Cohesity Cloud Scale Technology Manual Deployment Guide for Kubernetes Clusters

Release 11.2

VERITAS

now part of Cohesity

Cohesity Cloud Scale Technology Manual Deployment Guide for Kubernetes Clusters

Last updated: 2026-05-28

Legal Notice

Cohesity Support

Reach Cohesity Support

There are several ways to create a Cohesity support case.

- Go to Cohesity Support, to search in our knowledge base; or contact us by phone - United States and Canada: 1-855-9CO-HESI (926-4374), option 2.
- Log in to the Cohesity Support Portal to create a new case.
- Click the (**?**) icon on the Cohesity UI and select Support Portal.

Support/Service Assistance

First, contact the Service Provider that you have contracted for service and support. If you work directly with Cohesity and have a product warranty/entitlement, repair pricing, or technical support-related question, see your options below:

- To find solutions to your product issues or for suggestions or best practices, visit the Cohesity Knowledge Base.
- Log in to the Cohesity Support Portal to create a new case.
- To monitor your open cases, log in to the portal and click the **Cases** tab on the home page. This page should have all the case statuses and updates. You can also view individual case status.

Cohesity Software Running on Partner Hardware

For Cohesity software running on qualified third-party hardware, the following support workflow applies:

1. The customer may contact Cohesity Support first if the issue cannot be determined as a hardware issue.

   Note: Cohesity cannot process hardware replacement requests for partner hardware.

   ---

2. Cohesity Support triages the issue. If it is a software issue, Cohesity Support continues to work on it.
3. If it is a hardware/firmware issue or is suspected to be a hardware/firmware issue, Cohesity provides information about the issue to the customer and requests that the customer open a support ticket with the appropriate partner.
4. If needed, Cohesity Support can join a three-way call with the partner and the customer.
5. The customer informs Cohesity Support on the progress of the partner’s case.

Contents

Chapter 1

Introduction

This chapter includes the following topics:

- About Cloud Scale deployment
- About NetBackup Snapshot Manager
- Required terminology
- User roles and permissions

About Cloud Scale deployment

Cohesity Cloud Scale Technology redefines data management for the next decade. It is a new generation of the proven NetBackup architecture. It is designed to operate cloud-natively and uses technologies such as containers and micro services along with IT techniques like elasticity and automation. These modern techniques enable NetBackup to operate cloud-natively and deliver a consistent experience across multiple clouds to return on investment (ROI), service resiliency, and security. These factors would automatically reduce the operational complexity and the costs involved.

Cloud Scale Technology provides the NetBackup deployment solution on the following Kubernetes clusters:

- Amazon Elastic Kubernetes (EKS) cluster, in the Amazon Web Services (AWS) Cloud
- Azure Kubernetes Services cluster (AKS), in the Azure Cloud

The solution facilitates an orchestrated deployment of the following components on Kubernetes clusters:

- **NetBackup**: You can deploy NetBackup on the Kubernetes clusters for scaling the capacity of the NetBackup host to serve a large number of requests concurrently running on the NetBackup primary server at its peak performance capacity.

- **MSDP Scaleout**: In addition to the NetBackup components namely the primary and media servers, the deduplication engine (1 to 16 replicas) may also be deployed.
  From NetBackup version 10.4 onwards, MSDP Scaleout is deployed as a component during the Cloud Scale solution deployment. Refer to the following section to deploy MSDP Scaleout separately without the environment operator or Helm charts:
  See "Deploying MSDP Scaleout" on page 427.
- **NetBackup Snapshot Manager**: You can deploy NetBackup Snapshot Manager with autoscaling capabilities for data movement.

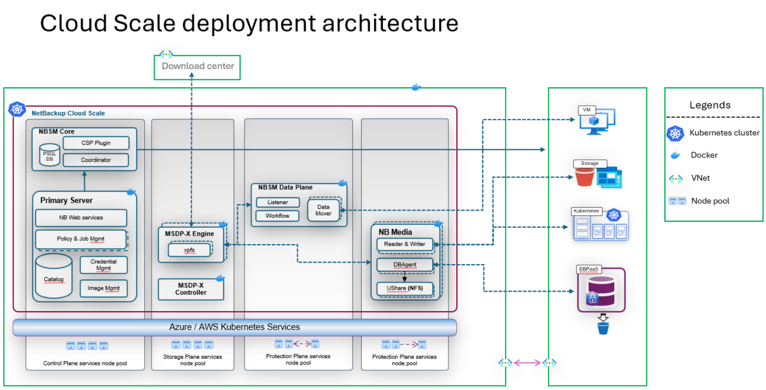
The guide describes a very comprehensive method to deploy, configure, and remove the Cloud Scale components using the environment operators.

Supported platforms

For AWS cloud: Amazon Elastic Kubernetes Service

For Azure cloud: Azure Kubernetes Service

Figure 1-1 Cloud Scale architecture before decoupling

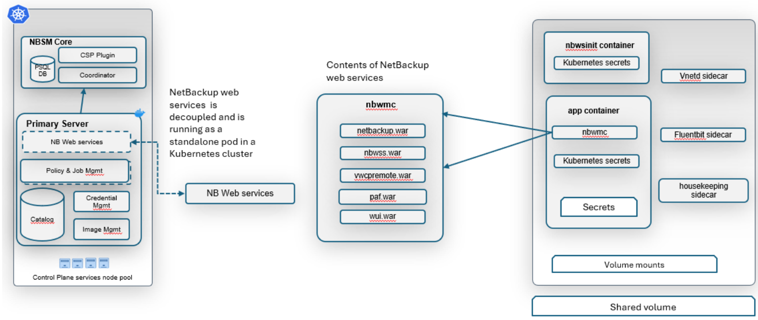


Decoupling of NetBackup web services from primary server

From Cloud Scale Technology version 10.5 and later, the NetBackup web services are decoupled (dissociated) from the primary server so that it can be containerized and executed independently as a standalone service pod in a Kubernetes cluster. All the other APIs and web UI behavior remains the same within the NetBackup cluster. The NetBackup web applications and the supporting infrastructure is deployed to `/usr/openv/wmc/` as part of a new Docker container image called `netbackup-ws-app`. This image (netbackup-ws-app) is deployed in a new separate Kubernetes pod within the NetBackup cluster.

The deployment of the web services pod is orchestrated using NetBackup operator framework which manages all of the NetBackup Kubernetes deployment artifacts within a Kubernetes cloud deployment.

Figure 1-2 Decoupled NetBackup web services architecture



- Components involved in the decoupled NetBackup web services architecture:
  - **netbackup.war**
  - **nbwss.war**
  - **webui.war**
  - **vwcpremote.war**
  - **paf.war**
- **nbwsinit container**: The `nbwsinit` container is the first file to execute during the deployment. It performs the environmental setup work described in the deployment section below. When the `nbwsinit` container stops, the rest of the containers start simultaneously and the `nbwsinit` container dissociates.
- **Volume mounts**: Components of volume mounts supports `vnet` proxy sidecar and the `nbwsapp` pod mounts the `/mnt/sock-data persistent` volume where the proxy files are used to exchange within sockets.
- **Shared volume mounts**: This volume represents the NetBackup primary catalogue data log.
- **Kubernetes Secrets**: Components of `Kubernetes secrets` is mounted and used by the `nbwsinit` container to establish trust with the `nbatd` pod in order to get the CA certificate as prerequisite to creating the web services certificates and trust stores.
- **Vnetd sidecar (Secure Communication)**: For Cloud Scale Technology deployment, the NetBackup web service is unable to make localhost connections.

It needs to connect remotely on the primary pod and the decoupled `nbatd` pod. For these remote service connections, it now needs to use the secure communication infrastructure provided by the `vnetd` inbound and outbound proxies.

- **Housekeeping sidecar**: The `nbwsapp` includes the `nbhousekeeping` sidecar container in its deployment to perform scheduled housekeeping tasks such as log file clean up, rotation and telemetry collection.
- **Fluentbit log collection sidecar**: The `nbwsapp` pod includes the `fluentbit` sidecar container in its deployment to perform regular log collection from the source container(s) and combine in a separate `fluentbit-collector` pod.

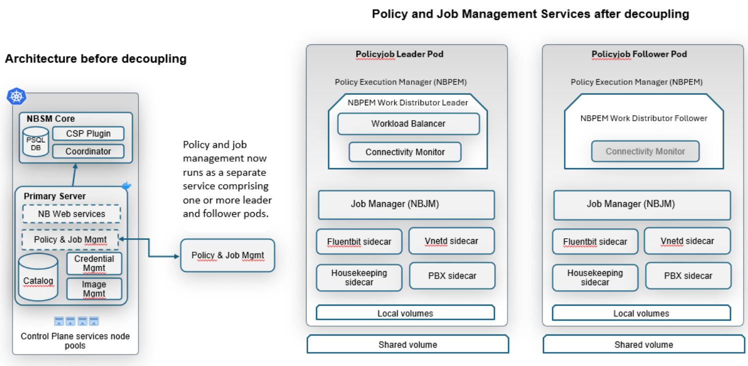
Decoupling of NetBackup Policy and Job Management from primary server

In Cloud Scale Technology, policy execution and job management (NBPEM and NBJM) is implemented as StatefulSet which can be deployed through the operators. Policy and job management capabilities are decoupled from the primary server and run as separate service in Kubernetes cluster.

See “Deploying the operators” on page 136.

NBPEM/NBJM is dependent on primary, web services and mqbroker pods. NBPEM/NBJM cluster comprises of one or more `policyjobmgr` pods and one or more `policyjob` pods. StatefulSet `policyjobmgr` represents the leaders whereas StatefulSet `policyjob` is associated with the followers. You can consider adding a secondary leader to guarantee continuous job execution if the primary leader is unavailable.

Figure 1-3 Policy and Job Management architecture



Note: In the above figure, **NB Web services** is a decoupled component. For more information refer to Decoupling of NetBackup web services from primary server.

A NBPEM/NBJM pod consists of the following containers:

- The main container for core policy and job management
- PBX sidecar
- Vnetd sidecar
- Fluentbit sidecar
- Housekeeping sidecar

nbpemreq

The `nbpemreq` (NetBackup Policy Execution Manager Requisition) command shows the distribution of policy/clients across the follower pods.

For Cloud Scale, it is recommended to run the `nbpemreq` command on the primary pod or on the policyjob leader.

It is recommended to use `-M` option with the pod name to direct the request to the specific pod. When `-M` option is not specified, the default target is one of the leaders.

For more information on `nbpemreq` command, refer to the *NetBackup™ Commands Reference Guide*.

Limitations and recommendations

- **Multiplexing (MPX)**: MPX is not supported in NBPEM/NBJM scale-out environment.
  Supporting MPX in any format could lead to unpredictable/unexpected/undesired behaviors. Hence it is recommended to disable the MPX support.
- **Session end and catalog backup**: For scale-out deployments (Kubernetes cluster), session end catalog backup schedules are not supported.
  For non scale-out deployments, session end catalog backup schedules and persistence would be moved to a Json formatted file.
- **Secondary leader**: The secondary leader's role is limited to forwarding media server calls to the follower pods. All communication from the media server is load balanced between primary and secondary leaders, ensuring uninterrupted job execution in case the primary leader is unavailable.
- **Scale-out/Scale-in**:
  - Auto-scaling is not supported. However, manual scaling of follower and leader is allowed.

- By default, Cloud Scale starts with a single follower pod; you can scale this out up to a maximum of 12 pods.
  By default, Cloud Scale starts with a single leader pod; you can scale this out up to a maximum of 6 pods.

  Warning: The users are advised to scale out the leader and follower services only if necessary, as this would consume more resources.

- Run the following command to control the follower replica count:
  ```
  helm upgrade cloudscale cloudscale-11.2-xxx.tgz -n <namespace>
  --reuse-values --set
  environment.primary.replicas.policyjob=<count>
  ```
- Run the following command to control the leader replica count:
  ```
  helm upgrade cloudscale cloudscale-11.2-xxx.tgz -n <namespace>
  --reuse-values --set
  environment.primary.replicas.policyjobmgr=<count>
  ```

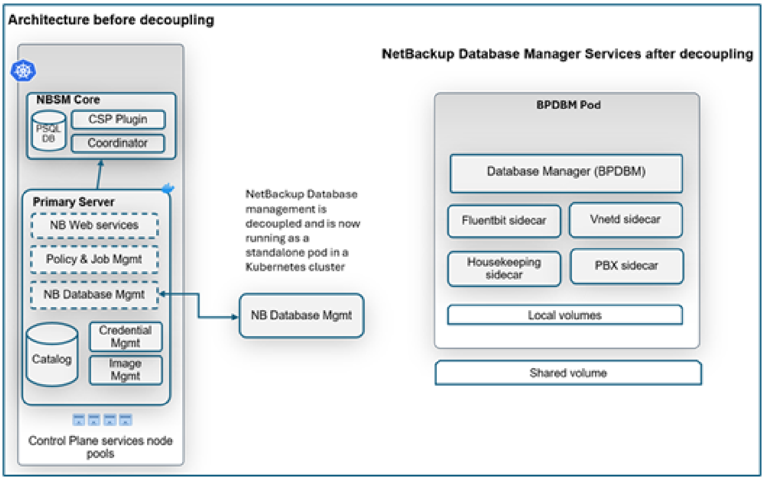
Decoupling of NetBackup Database Manager from primary server

In Cloud Scale Technology, NetBackup Database Manager (BPDBM) is implemented as a StatefulSet deployed by the NetBackup operator. BPDBM capabilities that is, catalog management, interface to policy, schedule, image and media databases and so on are decoupled from the primary server and reside in a separate Kubernetes service.

BPDBM service is dependent on web services and comprises a single BPDBM pod. Requests to the database manager from other NetBackup components are now routed to this pod.

Note: This release does not support scale-out of BPDM pod.

Figure 1-4 Decoupling of Database Manager architecture



Note: In the above figure, NetBackup web services and NBPEM/NBJM are decoupled components. For more information refer to the following section:

See "Decoupling of NetBackup Policy and Job Management from primary server" on page 15.

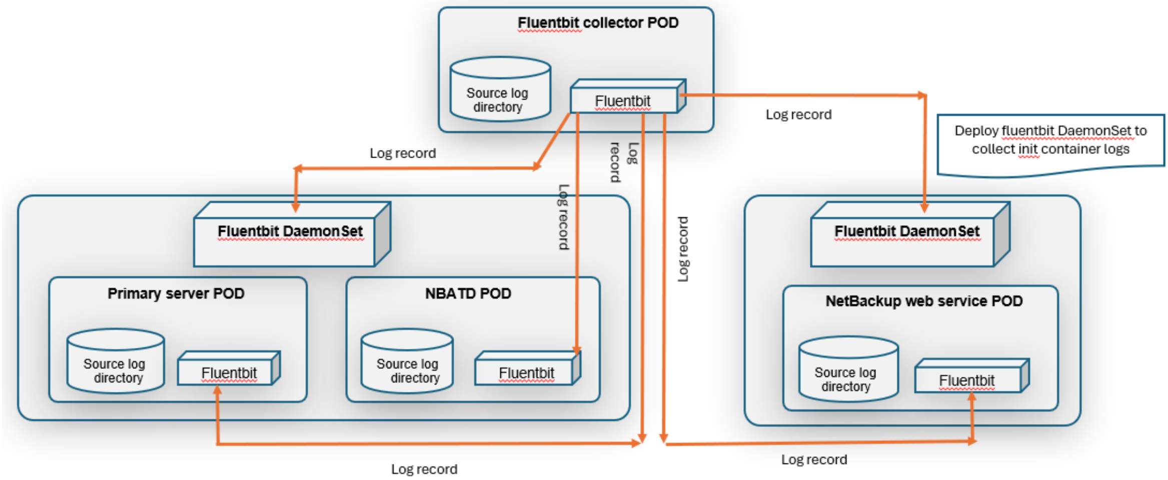
A BPDBM pod consists of the following containers:

- The main container for NetBackup database management (BPDBM)
- PBX sidecar
- Vnetd sidecar
- Fluentbit sidecar
- Housekeeping sidecar

Logging feature (fluentbit) in Cloud Scale

Logging feature is introduced in 10.5 and later version of NetBackup for Cloud Scale Technology which allows you to consolidate log files that have been distributed during the process of running NetBackup in a scale-out environment. It helps you to gather all the log files in one place, making them easier to access and use.

Figure 1-5 Architecture of logging feature (Fluentbit)



To deploy fluentbit for logging feature, following are the required components:

- **Collector pod**: The collector pod receives logs from the DaemonSet and Application sidecar containers. The collector pod itself consists of two containers:
  - **Fluentbit-collector:** This container is responsible for receiving the logs and then writing them to a central location based on the details such as date, namespace, pod, container and file path. The primary purpose of the collector is to consolidate and write the files to a centralized the destination.
  - **Log-Cleanup Sidecar** This container on the collector pod is responsible for cleaning up logs from the PVC (PersistentVolumeClaim) attached. There are variables that can be configured to determine retention and other parameters.
- **Sidecar sender**: This container is the Kubernetes sidecar container that runs with NetBackup application pods. The pods produce NetBackup application specific logs that are to be collected, and access to the location of the logs is shared with the fluentbit sidecar. It scrapes those logs and sends them to the Collector Pod. The logs are stored in a shared volume mounted at `/mnt/nblogs`.
- **DaemonSet sender**: DaemonSet senders in Kubernetes are the pods allocated to the nodes based on specific taints and tolerations. Nodes with certain taints reject DaemonSets without matching tolerations, while nodes with matching toleration are assigned DaemonSet sender pods. These sender pods have access to the container logs of all pods within the node. This allows the

DaemonSet sender to gather `stdout/stderr` logs for NetBackup applications as well as the Kubernetes infrastructure.

- **Log-Viewer**: The log-viewer pod (introduced in NetBackup version 11.0) serves as the primary pod for users to exec into the pod and view the logs that have been collected by the fluentbit logging system. It also hosts APIs that provides access to extract the collected logs.

  Note: By default, only stdout logs are downloaded. To include sidecar logs, the **fluentbit.enableSidecarsLogging** flag must be manually enabled.

  For more information on log extraction, see the following section:
  See "Extracting NetBackup logs" on page 229.

Known issue

An abrupt restart of the listener service (including the fluentbit sidecar) while jobs are running may occasionally result in duplicate log entries being sent to the **fluentbit-collector**.

Bootstrapper pod

After deploying the NetBackup Kubernetes cluster, you might encounter pods stuck in an 'Init' state due to bootstrapper pod failure. This pod is short-lived and doesn't remain active after failing. To identify the cause of failure, check the bootstrapper logs within the NetBackup Fluent-bit collector.

Refer to the section See "Troubleshooting issue for bootstrapper pod" on page 387. for more details.

Debug tool container for troubleshooting

The NetBackup Debug Tool Container provides a dedicated, ephemeral debugging environment for troubleshooting NetBackup Kubernetes deployments. It is designed to be attached dynamically to running pods or nodes using the `kubectl debug` command, without modifying or rebuilding the original NetBackup container images.

The debug tool container includes a comprehensive set of diagnostic and troubleshooting utilities including tools such as **atop**, **htop**, **gdb**, and **strace**. These utilities enable administrators to investigate runtime, performance, and process-level issues directly within the kubernetes environment. This approach is especially useful in restricted or air-gapped environments where installing tools at runtime is not feasible.

Because the container is ephemeral, it is automatically terminated after the debugging session ends. Each subsequent debugging attempt creates a new debug container instance, ensuring that production workloads remain unchanged during troubleshooting activities.

For running the debug tool, refer to the following section:

Note: For debugging pods or nodes (for example, using `kubectl debug`), a minimum kubectl client version 1.31 is required. Cohesity recommends using the latest stable kubectl release. The kubectl client version should remain within one minor version of the cluster's API server version, in accordance with the official kubernetes version skew policy.

See "Troubleshoot issues with debug tool" on page 372.

Monitoring health checker

Cloud Scale provides a built-in **Health Checker** to help monitor and validate the health of Kubernetes-based NetBackup deployments. The health checker is integrated with the `kubectl-cloudscale` plugin and runs a set of diagnostic checks to verify that Cloud Scale components are correctly installed, accessible, and properly configured across different lifecycle stages as follows:

- Pre-deployment
- Post-deployment
- Pre-upgrade

It reports OKAY, NOT_OKAY, or SKIPPED statuses to assist in early detection and troubleshooting of potential issues in Cloud Scale environments.

For detailed usage, supported checks, command options, and result interpretation, refer to the following section:

See "Overview" on page 194.

About NetBackup Snapshot Manager

You must deploy Snapshot Manager solution in your Kubernetes cluster environment. The Kubernetes cluster must be created with appropriate network and configuration settings. Before you deploy the solution, ensure that your environment meets the requirements.

See "Prerequisites for Snapshot Manager (AKS/EKS)" on page 41.

Common fluent-bit sidecar for datamover log collection

Starting from NetBackup version 11.1, the fluent-bit sidecar has been removed from the datamover pod. Log collection for datamover pods will now be handled by a single, centralized fluent-bit sidecar running as part of the `flexsnap-listener` pod.

Benefits

- Reduced resource usage (fewer sidecars per datamover instance).
- Simplified log collection and management.
- Consistent logging architecture across components.

Previous behavior (before 11.1 release)

- Each datamover pod launched with its own fluent-bit sidecar.
- This sidecar was responsible for collecting datamover logs and forwarding them to the collector pod.

Current behavior (from 11.1 release)

- The per-pod fluent-bit sidecar has been removed.
- A single fluent-bit sidecar, running continuously with the `flexsnap-listener` pod, now collects and forwards logs from all datamover pods.
- This implementation is similar to the existing fluent-bit sidecar setup used for NetBackup Primary and Media pods.
- An abrupt restart of the listener service, along with the fluent-bit sidecar, while jobs are in progress can result in duplicate log entries being relayed to the fluent-bit collector.

Required terminology

The table describes the important terms for NetBackup deployment on Kubernetes cluster. For more information visit the link to Kubernetes documentation.

Table 1-1 Important terms

Term	Description
Pod	A Pod is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers. For more information on Pods, see Kubernetes Documentation.

Table 1-1 Important terms *(continued)*

Term	Description
StatefulSet	StatefulSet is the workload API object used to manage stateful applications and it represents a set of Pods with unique, persistent identities, and stable hostnames. For more information on StatefulSets, see Kubernetes Documentation.
Job	Kubernetes jobs ensure that one or more pods execute their commands and exit successfully. For more information on Jobs, see Kubernetes Documentation.
ConfigMap	A ConfigMap is an API object used to store non-confidential data in key-value pairs. For more information on ConfigMaps, see Kubernetes Documentation.
Service	A Service enables network access to a set of Pods in Kubernetes. For more information on Service, see Kubernetes Documentation.
Persistent Volume Claim	A PersistentVolumeClaim (PVC) is a request for storage by a user. For more information on Persistent Volumes, see Kubernetes Documentation.
Persistent Volume	A PersistentVolume (PV) is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using storage classes. For more information on Persistent Volumes, see Kubernetes Documentation.
Custom Resource	A Custom Resource (CR) is an extension of the Kubernetes API that is not necessarily available in a default Kubernetes installation. For more information on Custom Resources, see Kubernetes Documentation.
Custom Resource Definition	The CustomResourceDefinition (CRD) API resource lets you define custom resources. For more information on CustomResourceDefinitions, see Kubernetes Documentation.
Secret	A Secret is an object that contains a small amount of sensitive data such as a password, a token, or a key. Such information might otherwise be put in a Pod specification or in a container image. For more information on Secrets, see Kubernetes Documentation.
ServiceAccount	A service account provides an identity for processes that run in a Pod. For more information on configuring the service accounts for Pods, see Kubernetes Documentation.

Table 1-1 Important terms *(continued)*

Term	Description
ClusterRole	An RBAC Role or ClusterRole contains rules that represent a set of permissions. Permissions are purely additive (there are no "deny" rules). For more information on ClusterRole, see Kubernetes Documentation.
ClusterRoleBinding	A role binding grants the cluster-wide permissions defined in a role to a user or set of users. For more information on ClusterRoleBinding, see Kubernetes Documentation.
Namespace	Kubernetes supports multiple virtual clusters backed by the same physical cluster. These virtual clusters are called namespaces. For more information on Namespaces, see Kubernetes Documentation.

User roles and permissions

Note the following for user authentication:

- An Administrator must define the custom user credentials by creating a secret; and then provide the secret name at the time of primary server deployment.
- A custom user is assigned the role of a NetBackup Security Administrator and can access the NetBackup Web UI after deployment.
- A custom user will be persisted during the pods restart or upgrade.
- For the custom user, you can change only the password after the deployment. The changed password will be persisted. If the username is changed after the deployment, an error message will be logged in the Operator pod.
- You can delete the secret after the primary server deployment. In that case, if you want to deploy or scale the media servers, you must create a new secret with the same username which was used in the primary server CR. The password can be the same or different. If you change the password, it is also changed in the primary server pod, and gets persisted.
- Do not create a local user in the pods (using the `kubectl exec` or `useradd` commands) as this user may or may not be persisted.
- The cloud provider user is supported through Single Sign-on (SSO). For the detailed user integration information, refer to the *NetBackup Administrator's Guide Volume I*.
- An **nbitanalyticsadmin** user is available in primary server container. This user is used as **Master Server User ID** while creating data collector policy for data collection on NetBackup IT Analytics portal.

- NetBackup runs most of the primary server services and daemons as non-root user **(nbsvcusr)**.
- ClusterRole named **netbackup-operator-netbackup-role** is set in the NetBackup Operator to define the cluster wide permissions to the resources.
- Appropriate roles and Kubernetes cluster specific permissions are set to the cluster at the time of cluster creation.
- After successful deployment of the primary and media servers, the operator creates a custom Kubernetes role with name `ResourceName-admin` whereas `Resource Name` is given in primary server or media server CR specification.

  The following permissions are provided in the respective namespaces:

Resource name	API group	Allowed operations
ConfigMaps	default	■ Create ■ Delete ■ Get ■ List ■ Patch ■ Update ■ Watch
Nodes	default	■ Get ■ List

  This role can be assigned to the NetBackup Administrator to view the pods that were created, and to execute into them. For more information on the access control, see Kubernetes Access Control Documentation.

  Note: One role would be created, only if primary and media servers are in same namespace with the same resource name prefix.

- (*AKS-specific only*) Your AKS cluster must have the RBAC enabled. To view the permissions set for the AKS cluster, use one of the following methods and verify if `enbleRBAC` is set to **true**:
  - Run the following command:

    ```
    az resource show -g <resource group name> -n <cluster name>
    --resource-type
    Microsoft.ContainerService/ManagedClusters --query
    properties.enableRBAC
    ```

  - Run the `az aks list` command.

- You can check the cluster's resource details at `resources.azure.com` and verify if `enableRBAC` is set to **true**.

Role-based authentication (RBAC)

NetBackup Operator deployment uses a `serviceAccount` and it must have the following permissions:

Table 1-2

Resource Name	API Group	Allowed Operations
ConfigMaps	default	■ Create ■ Delete ■ Get ■ List ■ Patch ■ Update ■ Watch
Nodes	default	■ Get ■ List
PersistentVolumeClaims	default	■ Create ■ Delete ■ Get ■ List ■ Patch ■ Update ■ Watch
(*AKS-specific only*) PersistentVolume	default	■ Create ■ Delete ■ Get ■ List ■ Patch ■ Update ■ Watch

Table 1-2 *(continued)*

Resource Name	API Group	Allowed Operations
Pods	default	■ Create ■ Delete ■ Get ■ List ■ Patch ■ Update ■ Watch
Pods/exec	default	■ Create ■ Get
Secret	default	■ Get ■ List ■ Watch
Services	default	■ Create ■ Delete ■ Get ■ List ■ Patch ■ Update ■ Watch
StatefulSet	app	■ Create ■ Delete ■ Get ■ List ■ Patch ■ Update ■ Watch
Jobs	batch	■ Create ■ Delete ■ Get ■ List

Table 1-2 *(continued)*

Resource Name	API Group	Allowed Operations
Primary servers	netbackup.veritas.com	■ Create ■ Delete ■ Get ■ List ■ Patch ■ Update ■ Watch
PrimaryServers/status	netbackup.veritas.com	■ Get ■ Patch ■ Update
Media servers	netbackup.veritas.com	■ Create ■ Delete ■ Get ■ List ■ Patch ■ Update ■ Watch
MediaServers/status	netbackup.veritas.com	■ Get ■ Patch ■ Update
Secrets	netbackup.veritas.com	Watch
Secrets/status	netbackup.veritas.com	■ Get ■ Patch ■ Update
Roles	rbac.authorization.k8s.io	■ Create ■ Get ■ List ■ Watch
Storageclasses	storage.k8s.io	■ Get ■ List

Table 1-2 *(continued)*

Resource Name	API Group	Allowed Operations
Deployment	app	■ Get ■ List ■ Update ■ Watch

Section 1

Configurations

Prerequisites

This chapter includes the following topics:

- Preparing the environment for NetBackup installation on Kubernetes cluster
- Prerequisites for Snapshot Manager (AKS/EKS)
- Prerequisites for Kubernetes cluster configuration
- Prerequisites for Cloud Scale configuration
- Prerequisites for deploying environment operators
- Prerequisites for using private registry

Preparing the environment for NetBackup installation on Kubernetes cluster

Ensure that the following prerequisites are met before proceeding with the deployment for AKS/EKS.

AKS-specific requirements

Use the following checklist to prepare the AKS for installation.

- Your Azure Kubernetes cluster must be created with appropriate network and configuration settings.
  For a complete list of supported Kubernetes cluster version, see the NetBackup Compatibility List for all Versions.

  Note: Azure Container Networking Interface (CNI) or Azure CNI overlay can be used as the network plugin for AKS.

- While creating the cluster, assign appropriate roles and permissions.

  Refer to the 'Concepts - Access and identity in Azure Kubernetes Services (AKS)' section in *Microsoft Azure Documentation*.
- Use an existing Azure container registry or create a new one. Your Kubernetes cluster must be able to access this registry to pull the images from the container registry. For more information on the Azure container registry, see 'Azure Container Registry documentation' section in *Microsoft Azure Documentation*.
- Deploying the Primary and Media server installation on the same node pool (node) is possible. For optimal performance, it is recommended to create separate node pools. Select the **Scale method** as **Autoscale**. The autoscaling feature allows your node pool to scale dynamically by provisioning and de-provisioning the nodes as required automatically.
- A dedicated node pool for Primary server must be created in Azure Kubernetes cluster.

  The following table lists the node configuration for the primary and media servers.

Node type		D16ds v4
Disk type		P30
vCPU		16
RAM		64 GiB
Total disk size per node (TiB)		1 TB
Number of disks/node		1
Cluster storage size	Small (4 nodes)	4 TB
	Medium (8 nodes)	8 TB
	Large (16 nodes)	16 TB

- For Cloud Scale environment, following are the recommended node configurations for primary node pool:

Node type	Standard_D8s_v5
CPU	8
RAM	32

- Another dedicated node pool must be created for Snapshot Manager (if it has to be deployed) with auto scaling enabled.

Following is the minimum configuration required for Snapshot Manager data plane node pool:

Node type	B4ms
RAM	8 GB
Number of nodes	Minimum 0 with auto scaling enabled.
Maximum pods per node	6 (system) + 4 (static pods) + RAM*2 (dynamic) = 26 pods or more

Following are the different scenario's on how the NetBackup Snapshot Manager calculates the number of job which can run at a given point in time, based on the above mentioned formula:

- For 2 CPU's and 8 GB RAM node configuration:

CPU	More than 2 CPU's
RAM	8 GB
Maximum pods per node	6 (system) + 4 (static pods) + 8*2 = 16 (dynamic pods) = 26 or more
Autoscaling enabled	Minimum number =1 and Maximum = 3

  Note: Above configuration will run 8 jobs per node at once.

- For 2/4/6 CPU's and 16 GB RAM node configuration:

CPU	More than 2/4/6 CPU's
RAM	16 GB
Maximum pods per node	6 (system) + 4 (Static pods) + 16*2=32 (Dynamic pods) = 42 or more
Autoscaling enabled	Minimum number =1 and Maximum = 3

  Note: Above configuration will run 16 jobs per node at once.

- All the nodes in the node pool must be running the Linux operating system. Linux based operating system is only supported with default settings.
- Taints and tolerations allows you to mark (taint) a node so that no pods can schedule onto it unless a pod explicitly *tolerates* the taint. Marking nodes instead of pods (as in node affinity/anti-affinity) is particularly useful for situations where most pods in the cluster must avoid scheduling onto the node.
  Taints are set on the node pool while creating the node pool in the cluster. Tolerations are set on the pods.
- If you want to use static private IPs and fully qualified domain names for the load balancer service, private IP addresses and FQDNs must be created in AKS before deployment.
- If you want to bind the load balancer service IPs to a specific subnet, the subnet must be created in AKS and its name must be updated in the **annotations** key in the **networkLoadBalancer** section of the custom resource (CR).
  For more information on the network configuration for a load balancer service, refer to the *How-to-Guide* section of the *Microsoft Azure Documentation*.
  For more information on managing the load balancer service, See "About the Load Balancer service" on page 205.
- Create a storage class with `Managed disc` storage type with `allowVolumeExpansion = true` and `ReclaimPolicy=Retain`. This storage class will be used for Primary server data and log volume. Media server storage details support azure disks only.
- Customer's Azure subscription should have **Network Contributor** role.
  For more information, see 'Azure built-in roles' section of *Microsoft Azure Documentation*.

EKS-specific requirements

1 Create a Kubernetes cluster with the following guidelines:

- Use Kubernetes version 1.27 onwards.
- AWS default CNI is used during cluster creation.
- Create a nodegroup with only one availability zone and instance type should be of at least **m5.4xlarge** configuration and select the size of attached EBS volume for each node more than 100 GB.
  The nodepool uses AWS manual or autoscaling group feature which allows your nodepool to scale by provisioning and de-provisioning the nodes as required automatically.

Note: All the nodes in node group must be running on the Linux operating system.

- Minimum required policies in IAM role:
  - AmazonEKSClusterPolicy
  - AmazonEKSWorkerNodePolicy
  - AmazonEC2ContainerRegistryPowerUser
  - AmazonEKS_CNI_Policy
  - AmazonEKSServicePolicy

2. Use an existing AWS Elastic Container Registry or create a new one and ensure that the EKS has full access to pull images from the elastic container registry.
3. It is recommended to create separate node pool for Media server installation with autoscaler add-on installed in the cluster. The autoscaling feature allows your node pool to scale dynamically by provisioning and de-provisioning the nodes as required automatically.
4. A dedicated node pool for Primary server must be created in Amazon Elastic Kubernetes Services cluster.

   The following table lists the node configuration for the primary and media servers.

Node type		m5.4xlarge
vCPU		16
RAM		64 GiB
Total disk size per node (TiB)		1 TB
Number of disks/node		1
Cluster storage size	Small (4 nodes)	4 TB
	Medium (8 nodes)	8 TB
	Large (16 nodes)	16 TB

5. Another dedicated node pool must be created for Snapshot Manager (if it has to be deployed) with auto scaling enabled.

Following is the minimum configuration required for Snapshot Manager data plane node pool:

Node type	t3.large
RAM	8 GB
Number of nodes	Minimum 1 with auto scaling enabled.
Maximum pods per node	Number of IPs required for Snapshot Manager data pool, must be greater than: the number of nodes (for node's own IP) + (RAM size per node * 2 * number of nodes) + (number of all kube-system pods running on all nodes) + static listener pod + number of nodes(for fluent daemonset) Number of IPs required for Snapshot Manager control pool, must be greater than: number of nodes (for node's own IP) + number of flexsnap pods(15) + number of flexsnap services (6) + nginx load balancer IP + no. of additional off host agents + operator + (number of all kube-system pods running on all nodes)

Following are the different scenario's on how the NetBackup Snapshot Manager calculates the number of job which can run at a given point in time, based on the above mentioned formula:

- For Cloud Scale environment
  Following is the recommended configuration for primary node pool:

Node type	m5.2xlarge
RAM	32 GB
CPU	8

- For DBPaaS Workload

Note: The following configuration is advised as the CPU credit limit was reached in the T-series workload.

Node type	m4.2xlarge
RAM	32 GB

- For 2 CPU's and 8 GB RAM node configuration:

CPU	More than 2 CPU's
RAM	8 GB
Maximum pods per node	Number of IPs required for Snapshot Manager data pool, must be greater than: number of nodes (for node's own IP) + (RAM size per node * 2 * number of nodes) + (number of all kube-system pods running on all nodes) + static listener pod + number of nodes(for fluent daemonset) Number of IPs required for Snapshot Manager control pool, must be greater than: number of nodes (for node's own IP) + number of flexsnap pods(15) + number of flexsnap services (6) + nginx load balancer IP + no. of additional off host agents + operator + (number of all kube-system pods running on all nodes)
Autoscaling enabled	Minimum number =1 and Maximum = 3

Note: Above configuration will run 8 jobs per node at once.

- For 2/4/6 CPU's and 16 GB RAM node configuration:

CPU	More than 2/4/6 CPU's
RAM	16 GB
Maximum pods per node	6 (system) + 4 (Static pods) + 16*2=32 (Dynamic pods) = 42 or more
Autoscaling enabled	Minimum number =1 and Maximum = 3

Note: Above configuration will run 16 jobs per node at once.

---

6. Taints and tolerations allows you to mark (taint) a node so that no pods can schedule onto it unless a pod explicitly *tolerates* the taint. Marking nodes instead of pods (as in node affinity/anti-affinity) is particularly useful for situations where most pods in the cluster must avoid scheduling onto the node.

   Taints are set on the node group while creating the node group in the cluster. Tolerations are set on the pods.

7. Deploy aws load balancer controller add-on in the cluster.

   For more information on installing the add-on, see 'Installing the AWS Load Balancer Controller add-on' section of the *Amazon EKS User Guide*.

8. Install cert-manager and trust-manager as follows:

   - Add helm repo using the following command:
     ```
     helm repo add jetstack https://charts.jetstack.io
     --force-update
     ```
   - Install cert-manager by using the following command:
     ```
     helm upgrade -i -n cert-manager cert-manager
     jetstack/cert-manager --version 1.18.2 --set
     webhook.timeoutSeconds=30 --set installCRDs=true --wait
     --create-namespace
     ```
     Create the deployment namespace: `kubectl create ns netbackup`
   - Install trust-manager by using the following command:
     ```
     helm upgrade -i --create-namespace --namespace trust-manager
     trust-manager jetstack/trust-manager --set
     app.trust.namespace=netbackup --version v0.19.0 --wait
     ```

9. The FQDN that will be provided in primary server CR specifications in networkLoadBalancer section must be DNS resolvable to the provided IP address.

10 Amazon Elastic File System (Amazon EFS) for shared persistence storage. To create EFS for primary server, see 'Create your Amazon EFS file system' section of the *Amazon EKS User Guide*.

EFS configuration can be as follow and user can update Throughput mode as required:

Performance mode: General Purpose

Throughput mode: Bursting (256 MiB/s)

Availability zone: Regional

Note: Throughput mode can be increased at runtime depending on the size of workloads and also if you are seeing performance issue you can increase the Throughput mode till 1024 MiB/s.

Note: To install the add-on in the cluster, ensure that you install the Amazon EFS CSI driver. For more information on installing the Amazon EFS CSI driver, see 'Amazon EFS CSI driver' section of the *Amazon EKS User Guide*.

---

11 If NetBackup client is outside VPC or if you want to access the WEB UI from outside VPC then NetBackup client CIDR must be added with all NetBackup ports in security group inbound rule of cluster. See “About the Load Balancer service” on page 205. for more information on NetBackup ports.

- To obtain the cluster security group, run the following command:
  ```
  aws eks describe-cluster --name <my-cluster> --query
  cluster.resourcesVpcConfig.clusterSecurityGroupId
  ```
- The following link helps to add inbound rule to the security group: 'Add rules to a security group' section of the *Amazon EKS User Guide*.

12 The EFS based PV must be specified for Primary server catalog volume with `ReclaimPolicy=Retain`.

Host-specific requirements

Use the following checklist to address the prerequisites on the system that you want to use as a NetBackup host that connects to the AKS/EKS cluster.

AKS-specific

- Linux operating system: For a complete list of compatible Linux operating systems, refer to the Software Compatibility List (SCL) at:
  NetBackup Compatibility List for all Versions

- Install Docker on the host to install NetBackup container images through tar, and start the container service.
  Install Docker Engine
- Prepare the host to manage the AKS cluster.
  - Install Azure CLI.
    For more information, see 'Install the Azure CLI on Linux' section of the *Microsoft Azure Documentation*.
  - Install Kubernetes CLI
    For more information, see 'Install and Set Up kubectl on Linux' section of the *Kubernetes Documentation*.
  - Log in to the Azure environment to access the Kubernetes cluster by running this command on Azure CLI:
    ```
    az login –identity
    az account set --subscription <subscriptionID>
    az aks get-credentials --resource-group
    <resource_group_name> --name <cluster_name>
    az resource list -n $cluster_name --query
    [*].identity.principalId --out tsv
    az role assignment create --assignee <identity.principalId>
    --role 'Contributor' --scope
    /subscriptions/$subscription_id/resourceGroups/NBUX-QA-BiDi-
    RG/providers/Microsoft.Network/virtualNetworks/NBUX-QA-BiDiNet01/subnets/$subnet
    az login --scope https://graph.microsoft.com//.default
    ```
  - Log in to the container registry:
    ```
    az acr login -n <container-registry-name>
    ```

EKS-specific

- Install AWS CLI.
  For more information on installing the AWS CLI, see Install or update the latest version of the AWS CLI' section of the *AWS Command Line Interface User Guide*.
- Install Kubectl CLI.
  For more information on installing the Kubectl CLI, see 'Installing kubectl' section of the *Amazon EKS User Guide*.
- Configure docker to enable the push of the container images to the container registry.
- Create the OIDC provider for the AWS EKS cluster.

For more information on creating the OIDC provider, see 'Create an IAM OIDC provider for your cluster' section of the *Amazon EKS User Guide*.

- Create an IAM service account for the AWS EKS cluster.
  For more information on creating an IAM service account, see 'Configuring a Kubernetes service account to assume an IAM role' section of the *Amazon EKS User Guide*.
- If an IAM role needs an access to the EKS cluster, run the following command from the system that already has access to the EKS cluster:
  `kubectl edit -n kube-system configmap/aws-auth`
  For more information on creating an IAM role, see Enabling IAM user and role access to your cluster.
- Login to the AWS environment to access the Kubernetes cluster by running the following command on AWS CLI:
  ```
  aws eks --region <region_name> update-kubeconfig --name
  <cluster_name>
  ```
- Free space of approximately 13GB on the location where you copy and extract the product installation TAR package file. If using docker locally, there should be approximately 8GB available on the `/var/lib/docker` location so that the images can be loaded to the docker cache, before being pushed to the container registry.
- AWS EFS-CSI driver should be installed for static PV/PVC creation of primary catalog volume.

Prerequisites for Snapshot Manager (AKS/EKS)

A working Azure Kubernetes cluster

- Azure Kubernetes cluster
  - Your Azure Kubernetes cluster must be created with appropriate network and configuration settings.
  - Availability zone for AKS cluster must be disabled.
  - Cert-manager and trust-manager must be installed.
  - Two storage classes with the following configurations are required:

Storage class field	Data	Log
provisioner	disk.csi.azure.com	file.csi.azure.com
storageaccounttype	Premium_LRS	Premium_LRS

Storage class field	Data	Log
reclaimPolicy	Retain	Retain
allowVolumeExpansion	True	True

- Azure container registry (ACR)
  Use existing ACR or create a new one. Your Kubernetes cluster must be able to access this registry to pull the images from ACR. For more information on Azure container registry, see 'Azure Container Registry documentation' section in *Microsoft Azure Documentation*.
- Node pool
  A dedicated data node pool must be created specifically for Snapshot Manager's data plane services (workflow and data movement services). Autoscaling for the Snapshot Manager's data node pool must be enabled to allow dynamic scaling based on workload demands. It is recommended to configure the Snapshot Manager's data node pool minimum node count to 0, and max node count more than 0, depending on the operational requirements.
  It is recommended to use NetBackup primary server's node group for Snapshot Manager's control plane services. The control node pool must have managed identity enabled. Users must assign the appropriate roles to allow Snapshot Manager to operate as expected.
- Client machine to access AKS cluster
  - A separate computer that can access and manage your AKS cluster and ACR.
  - It must have Linux operating system.
  - It must have Docker daemon, the Kubernetes command-line tool (kubectl), and Azure CLI installed.
    The Docker storage size must be more than 6 GB. The version of kubectl must be v1.19.x or later. The version of Azure CLI must meet the AKS cluster requirements.
  - If AKS is a private cluster, see Create a private Azure Kubernetes Service cluster.
- Static Internal IPs
  If the internal IPs are used, reserve the internal IPs (avoid the IPs that are reserved by other systems) for Snapshot Manager and add DNS records for all of them in your DNS configuration.
  The Azure static public IPs can be used but is not recommended.

If Azure static public IPs are used, create them in the node resource group for the AKS cluster. A DNS name must be assigned to each static public IP. The IPs must be in the same location of the AKS cluster.

- Ensure that the managed identity has the scope to connect to the resource group of the cluster created for cloud scale deployment.

A working AWS Kubernetes cluster

- AWS Kubernetes cluster
  - Your AWS Kubernetes cluster must be created with appropriate network and configuration settings.
  - Two storage classes with the following configuration is required:

Storage class field	Data	Log
provisioner	kubernetes.io/aws-ebs	efs.csi.aws.com
reclaimPolicy	Retain	Retain
allowVolumeExpansion	True	True

Note: It is recommended to use a separate EFS for Snapshot Manager deployment and primary server catalog.

- Amazon Elastic Container Registry (ECR)
  Use existing ECR or create a new one. Your Kubernetes cluster must be able to access this registry to pull the images from ECR.
- Node Group
  A dedicated data node group must be created specifically for Snapshot Manager's data plane services (workflow and data movement services). Autoscaling for the Snapshot Manager's data node group must be enabled to allow dynamic scaling based on workload demands. It is recommended to configure the Snapshot Manager's data node group minimum node count to 0, and max node count more than 0, depending on the operational requirements.
  It is recommended to use NetBackup primary server's node group for Snapshot Manager's control plane services. Users must assign the appropriate IAM role with required permission to control node group.
- Client machine to access EKS cluster
  - A separate computer that can access and manage your EKS cluster and ECR.

  - It must have Linux operating system.
  - It must have Docker daemon, the Kubernetes command-line tool (kubectl), and AWS CLI installed.
    The Docker storage size must be more than 6 GB. The version of kubectl must be v1.19.x or later. The version of AWS CLI must meet the EKS cluster requirements.
  - If EKS is a private cluster, see Creating an private Amazon EKS cluster.
- Static Internal IPs
  If the internal IPs are used, reserve the internal IPs (avoid the IPs that are reserved by other systems) for Snapshot Manager and add forward and reverse DNS records for all of them in your DNS configuration.
  The AWS static public IPs can be used but is not recommended.

Prerequisites for Kubernetes cluster configuration

This section describes the working, execution and status details of the Config-Checker utility, Data-Migration for AKS and the Webhooks validation for EKS.

Config-Checker utility

How does the Config-Checker utility work

The Config-Checker utility performs checks on the deployment environment to verify that the environment meets the requirements, before starting the primary server and media server deployments.

How does the Config-Checker works:

- **RetainReclaimPolicy check**:
  This check verifies that the storage classes used for PVC creation in the CR have reclaim policy as **Retain.** The check fails if any of the storage classes do not have the **Retain** reclaim policy.
  For more information, see the 'Persistent Volumes Reclaiming' section of the *Kubernetes Documentation.*
- **MinimumVolumeSize check**:
  This check verifies that the PVC storage capacity meets the minimum required volume size for each volume in the CR. The check fails if any of the volume capacity sizes does not meet the requirements.
  Following are the minimum volume size requirements:
  - Primary server:

    - Data volume size: 30Gi
    - Catalog volume size: 100Gi
    - Log volume size: 30Gi
  - Media server:
    - Data volume size: 50Gi
    - Log volume size: 30Gi

- **Provisioner check**:

  EKS-specific only

  - Primary server: This will verify that the storage type provided is Amazon Elastic Block Store (Amazon EBS) for data and log volume. If any other driver type is used, the Config-Checker fails.
  - Media server: This will verify that the storage type provided is Amazon Elastic Block Store (Amazon EBS) for data and log volume. Config-Checker fails if this requirement is not met for media server.

  AKS-specific only

  - This check verifies that the provisioner type used in defining the storage class is **Azure disk**, for the volumes in Media servers. If not the Config-Checker will fail. This check verifies that the provisioner type used in defining the storage class is not **Azure files** for the volumes in Media servers. That is data and log volumes in case of Media server.

- (*EKS-specific only*) **AWS Load Balancer Controller add-on check**:
  This check verifies if the AWS Load Balancer Controller add-on is installed in the cluster. This load balancer controller is required for load balancer in the cluster. If this check fails, user must deploy the AWS Load Balancer Controller add-on

- **Cluster Autoscaler**
  This autoscaler is required for autoscaling in the cluster. If autoscaler is not configured, then Config-Checker displays a warning message and continues with the deployment of NetBackup servers.

  (*EKS-specific only*) This check verifies if the AWS Autoscaler add-on is installed in the cluster. For more information, refer to 'Autoscaling' section of the *Amazon EKS User Guide*.

- **Volume expansion check**:
  This check verifies the storage class name given for Primary server data and log volume and for Media server data and log volumes has `AllowVolumeExpansion = true`. If Config-Checker fails with this check then it

gives a warning message and continues with deployment of NetBackup media servers.

Config-Checker execution and status details

Note the following points.

- Config-Checker is executed as a separate job in Kubernetes cluster for both the primary server and media server CRs respectively. Each job creates a pod in the cluster. Config-checker creates the pod in the operator namespace.

  Note: Config-checker pod gets deleted after 4 hours.

- Execution summary of the Config-Checker can be retrieved from the Config-Checker pod logs using the `kubectl logs <configchecker-pod-name> -n <operator-namespace>` command.
  This summary can also be retrieved from the operator pod logs using the `kubectl logs <operator-pod-name> -n <operator-namespace>` command.
- Following are the Config-Checker modes that can be specified in the Primary and Media CR:
  - Default: This mode executes the Config-Checker. If the execution is successful, the Primary and Media CRs deployment is started.

    Note: This is default mode of Config-Checker if not changed explicitly through CR specs.

  - Dryrun: This mode only executes the Config-Checker to verify the configuration requirements but does not start the CR deployment.
  - Skip: This mode skips the Config-Checker execution of Config-Checker and directly start the deployment of the respective CR.
- Status of the Config-Checker can be retrieved from the primary server and media server CRs by using the `kubectl describe <PrimaryServer/MediaServer> <CR name> -n <namespace>` command.
  For example, `kubectl describe primaryservers environment-sample -n test`
- Following are the Config-Checker statuses:
  - Success: Indicates that all the mandatory config checks have successfully passed.
  - Failed: Indicates that some of the config checks have failed.

- Running: Indicates that the Config-Checker execution is in progress.
- Skip: Indicates that the Config-Checker is not executed because the `configcheckmode` specified in the CR is skipped.

- If the Config-Checker execution status is Failed, you can check the Config-Checker job logs using `kubectl logs <configchecker-pod-name> -n <operator-namespace>`. Review the error codes and error messages pertaining to the failure and update the respective CR with the correct configuration details to resolve the errors.
  For more information about the error codes, refer to NetBackup™ Status Codes Reference Guide.
- If Config-Checker ran in **dryrun** mode and if user wants to run Config-Checker again with same values in Primary or Media server YAML as provided earlier, then user needs to delete respective CR of Primary or Media server. And then apply it again.
  - If it is primary server CR, delete primary server CR using the `helm uninstall cloudscale -n <namespace>` command.
  - Apply the CR again. Add the required data which was deleted earlier at correct location, save it and install chart using the `helm install cloudscale -n <namespace>` command.

Data-Migration for AKS

This section describes the working, execution and status details of the Data-Migration for AKS.

How Data-Migration works

Data migration job kicks-in whenever there is any change in the storage class name of the primary server's catalog, log and data volumes.

- Migration job is used to perform data transfer of Primary server's file system data from `Azure disks` to `Azure premium files` for existing NetBackup deployments.
- If user is deploying NetBackup for the first time, then it is considered as fresh installation and the user can directly utilize the `Azure premium files` for Primary server's catalog volume. Primary server log and data volume supports azure disks only.
- For existing NetBackup deployments, migration job would copy Primary server's old `Azure disk` catalog volume to new azure file volumes, except nbdb data, nbdb data will be copied to new azure disks based data volume. Logs can be migrated to new azure disk log volume.

- To invoke the migration job, the `Azure premium files` storage class must be provided in the `cloudscale-values.yaml` file for catalog volume. User can also provide new azure disks storage class for log volume and new azure disk based data volume must be provided in `cloudscale-values.yaml`.
- The migration status is updated to *Success* in primary server CRD post successful data migration.

Note: Migration will take longer time based on catalog data size.

Data-Migration execution and status details

Data migration is carried out in form of **job** in NetBackup Kubernetes cluster for only the primary server CR. There will be a migration job per primary volume for data migration which will be part of NetBackup environment namespace. Each job creates a pod in the cluster.

- Execution summary of the Data migration can be retrieved from the migration pod logs using the following command:

  ```
  kubectl logs <migration-pod-name> -n
  <netbackup-environment-namespace>
  ```

  This summary can also be retrieved from the operator pod logs using the following command:

  ```
  kubectl logs <netbackup-operator-pod-name> -n
  <netbackup-environment-namespace>
  ```

- Status of the data migration can be retrieved from the primary server CR by using the following command:

  ```
  kubectl describe <PrimaryServer> <CR name> -n
  <netbackup-environment-namespace>
  ```

  Following are the data migration statuses:

  - Success: Indicates all necessary conditions for the migration of the Primary server are passed.
  - Failed: Indicates some or all necessary conditions for the migration the Primary server are failed.
  - Running: Indicates migration is in running state for the Primary server.

- If the Data migration execution status is failed, you can check the migration job logs using the following command:

  ```
  kubectl logs <migration-pod-name> -n
  <netbackup-environment-namespace>
  ```

Review the error codes and error messages pertaining to the failure and update the primary server CR with the correct configuration details to resolve the errors. For more information about the error codes, refer to *NetBackup™ Status Codes Reference Guide*.

Webhooks validation for EKS

This section describes the working, execution and status details of the Webhooks validation for EKS.

How does the webhook validation works

- Webhooks are implemented to validate the CR input provided in the environment section of `cloudscale-values.yaml` file which is the interface of NetBackup installation on the EKS cluster.
- For each user input in the `samples/environment.yaml` file a validation webhook is implemented.
- If any of the input value is not in the required form, then webhooks displays an error and prevents the creation of an environment.
- For primary server deployment, following webhook validations have been implemented:
  - Validate RetainReclaimPolicy: This check verifies that the storage classes used for PVC creation in the CR have reclaim policy as **Retain.** The check fails if any of the webhook do not have the **Retain** reclaim policy.
  - Validate MinimumVolumeSize: This check verifies that the PVC storage capacity meets the minimum required volume size for each volume in the CR. The check fails if any of the volume capacity sizes does not meet the following requirements for Primary server.
    - Catalog volume size: 100Gi
    - Log volume size: 30Gi
    - Data volume size: 30Gi
  - Validate CSI driver: This will verify that the PV created is provisioned using the `efs.csi.aws.com` driver, that is, AWS Elastic file system (EFS) for volumes catalog. If any other driver type is used, the webhook fails.
  - Validate AWS Elastic file system (EFS) controller add-on: Verifies if the AWS Elastic file system (EFS) controller add-on is installed on the cluster. This AWS Elastic file system (EFS) controller is required to use EFS as persistence storage for pods which will be running on cluster. Webhooks will check the

EFS controller add-on is installed and it is running properly. If no, then validation error is displayed.

- AWS Load Balancer Controller add-on check: Verifies if the AWS load balancer controller add-on is installed on the cluster. This load balancer controller is required to use load balancer in the cluster. Webhooks will check the load balancer controller add-on is installed and it is running properly. If no, then a validation error is displayed.

Webhooks validation execution details

Note the following points.

- A Webhook is an HTTP call back: An HTTP POST that occurs when an event-notification is sent through HTTP POST. A web application implementing Webhooks will POST a message to a URL when certain tasks happen.
- Webhooks are triggered when the following command is used to create or update the environment, validating the creation of the environment CR from the user-supplied `cloudscale-values.yaml` file:

  ```
  helm upgrade --install cloudscale cloudscale.tgz -f
  cloudscalevalues.yaml -n <namespace>
  ```

- Webhook validates each check in sequence. Even if one of the validation fails then a validation error is displayed and the execution is stopped.
- The error must be fixed and the `cloudscale-values.yaml` file must be applied so that the next validation check is performed.
- The environment is created only after webhook validations are passed.

Prerequisites for Cloud Scale configuration

This section describes the best practices and recommendations around sizing for cloud and cluster specific to be followed before Cloud Scale deployment.

Cluster specific settings

Private cluster

- It is recommended to have a private Kubernetes cluster created for Cloud Scale deployment.
- Ensure that the control plane or API server of the private created Kubernetes cluster has an internal IP address.

Note: The use of private cluster ensures that the network traffic between your API server and node pools remain on the private network only.

Node group/Pool setting

- Select Linux based operating system for the control and data pool nodes.
  By default, Linux based operating system is supported only with default settings.
- NetBackup Cloud Scale does not depend on Kubernetes versions. Ensure that Kubernetes cluster exists before deploying Cloud Scale.

Autoscaling parameters

- The autoscaling value for the **cpdata** pool node must always be set to **True**.
- The minimum value of nodes in this node pool must be 0 and the maximum value can be obtained using the following formulae:
  `Number of max nodes = Number of parallel backup and snapshot jobs to run / Minimum of (RAM per node in GB, max_jobs setting)`

Maximum pods per node setting

- For Azure:
  `MAX pods per node = (RAM size in GB * 2) + number of Kubernetes and CSP pods[10] + 3(listener + fluent collector + fluentbit)`
- For AWS:
  Node size in AWS must be selected depending on ENIC available with the node type. For more information on changing the value of max pods per node in AWS, refer to AWS Documentation.

Note: If the max pods per node are not sufficient, then max jobs per node can be reduced as mentioned in the 'max_jobs tunable' content in the following section.

Pool settings

- **NetBackup pool**: Used for deployment of NetBackup primary services along with Snapshot Manager control plane services.
  Minimum CPU requirement and Node size RAM: 4 CPU and 16 GB RAM
- **cpdata pool**: Used for deployment of Snapshot Manager data plane (dynamically created) services.

Average size of VM to be backed up **	RAM requirement in GB's for cpdata node	Number of CPU's	Tunable
<= 2 TB	8	2	
2 TB > and < 4 TB	8	2	Max_jobs = 4
4 TB > and < 8 TB	16	4	Max_jobs = 5
8 TB > and < 16 TB	16	4	Max_jobs = 4
16 TB > and < 24 TB	24	4	Max_jobs = 3
24 TB > and < 32 TB	32	4	Max_jobs = 3

Note: ** If customer has distinct sizes of hosts to be protected then one should consider the higher sized VM's as an average size of the VM.

- **Media pool**: CPU requirement and Node size RAM: 4 CPU and 16 GB RAM
- **MSDP pool**: CPU requirement and Node size RAM: 4 CPU and 16 GB RAM
  - max_jobs tunable: The max_jobs tunable parameter is used to restrict the number of jobs that can run on single node of the Cloud Scale cpdata node which can be used to reduce the number of jobs a single node can run.

    The max_jobs must be updated as follows:

    ```
    $ Kubectl edit configmap flexsnap-conf -n <nbux ns>
    ```

    Add the following entry in `flexsnap.conf` section:

    ```
    [capability_limit]
    max_jobs=16
    ```

    For example,

    ```
    =====
    ~$ k describe cm flexsnap-conf
    Name: flexsnap-conf
    Namespace: nbux-002522
    Labels: <none>
    Annotations: <none>
    Data
    ====
    flexsnap.conf:
    ----
    [agent] id = agent.8308b7c831af4b0388fdd7f1d91541e0
    ```

```
[capability_limit]
max_jobs=16
=======
```

- Tuning account rate limit: For BFS performance improvement, the API limits per AWS account can be updated as per the following formulae:

  ```
  X  = Account rate limit
  V1 = Number of VM's
  S1 = schedules/day
  D1 = Data change rate TB/incremental backup
  X  = ((S1 * D1 * V1)/40) < 1 ? keep 1000 : go for (X=
  ((X+1)*1000)) requests/sec
  ```

  For example,

  - The default theoretical speed for the account is 43 TB/day (1000 request per sec x 86400 sec in a day x 512 KB block size).
  - For PP schedule frequency of 1 per day and each VM around 1 TB size.
  - Theoretical maximum for number of full/day if the backup window is the full day, then 43 VM/day can be backed up.
  - With 10% incremental changes everyday, the theoretical maximum for incremental is 380 incremental VM's/day with all incrementals having similar change rate. This incremental change does not consider obtaining the changed list and other pre and post backup functionality. If you consider this as taking 20% of time, then it would be around 250 incremental VMs/ day.

Cloud specific settings

For AWS

1. **Configuration of EBS endpoint**: Create VPC endpoint to EBS for reduced cost and better performance during backup from snapshot operation in AWS.

   For more information on creating AWS EBS direct interface endpoint, refer to the AWS Documentation.

2. **Configuration of S3 endpoint**, for S3 as the target for backup from snapshot operations:

   If the STU configured in context of backup job is S3, the configure VPC Gateway endpoint to S3 service. For more information on how to create S3 endpoint, refer to the AWS Documentation.

3. **Permissions and role assignment**: Before plugin configuration, the Kubernetes cluster requires permissions to be assigned to IAM of Primary nodepool. The IAM role must be assigned during the node pool creation as follows:
   - Create a nodepool with the name as **nbupool**.
   - Assign the appropriate IAM role to the **nbupool** nodepool.
   - If the IAM role is assigned after the plugin configuration, then restart the agent pod using the following command:

     ```
     $ Kubectl delete pod <flexsnap agent pod> -n <nbu namespace>
     ```

   Note: When upgrading to NetBackup version 11.0 or later, ensure that you add `eks:ListNodegroups` permission to the role.

   ---

4. **AWS EFS bursting mode**: Considering the performance and cost, it is recommended to select the **bursting** mode for the volume based on EFS (AWS).
5. **Recommended PV sizes**: If required the PV sizes (catalog, MSDP log PV's) can be altered after the Cloud Scale deployment.
6. **Expiring image before the default time**: If required the backup image can be expired within the default 24 hours of backup from the primary pod/container by running the following command:

   ```
   # bpexpdate -backupid <backup id> [-copy <number>] [-force]
   ```

For AKS

1. **Permissions and role assignment**: Before plugin configuration, the Kubernetes cluster requires permissions to be assigned to the System Managed Identity as follows:
   - Obtain the name of the infrastructure resource group for the Kubernetes cluster.
   - Enable the System Managed Identity on the identified nodepool (nbupool).
   - Assign the role having the Snapshot Manager permission.
2. **Recommended PV sizes**: If required the PV sizes (catalog, MSDP log PV's) can be altered after the Cloud Scale deployment.
3. **Expiring image before the default time**: If required the backup image can be expired within the default 24 hours of backup from the primary pod/container by running the following command:

```
# bpexpdate -backupid <backup id> [-copy <number>] [-force]
```

Prerequisites for deploying environment operators

Ensure that the following prerequisites are met before proceeding with the deployment.

- Taints and tolerations allows you to mark (taint) a node so that no pods can schedule onto it unless a pod explicitly *tolerates* the taint. Marking nodes instead of pods (as in node affinity/anti-affinity) is particularly useful for situations where most pods in the cluster must avoid scheduling onto the node.
  To use this functionality, user must create the node pool/node group with the following detail:
  - Add a label with certain key value. For example `key = nbpool, value = nbnodes`
  - Add a taint with the same key and value which is used for label in above step with effect as *NoSchedule*.
    For example, `key = nbpool, value = nbnodes, effect = NoSchedule`
- Install cert-manager and trust-manager as follows:
  - Install cert-manager by using the following command:
    ```
    helm upgrade -i -n cert-manager cert-manager
    jetstack/cert-manager --version 1.18.2 --set
    webhook.timeoutSeconds=30 --set installCRDs=true --wait
    --create-namespace
    ```
    Create the deployment namespace: `kubectl create ns netbackup`
    For details, see *cert-manager Documentation*.
  - Install trust-manager by using the following command:
    ```
    helm upgrade -i --create-namespace --namespace trust-manager
    trust-manager jetstack/trust-manager --set
    app.trust.namespace=netbackup --version v0.19.0 --wait
    ```
- A workstation or VM running Linux with the following:
  - Configure `kubectl` to access the cluster.
  - Install Azure/AWS CLI to access Azure/AWS resources.
  - Configure docker to be able to push images to the container registry.
  - Helm is installed and configured.
  - Free space of approximately 15.5 GB on the location where you copy and extract the product installation TAR package file. If using docker locally, there

should be approximately 15 GB available on the `/var/lib/docker` location so that the images can be loaded to the docker cache, before being pushed to the container registry.

AKS-specific

- A Kubernetes cluster in Azure Kubernetes Service in AKS with multiple nodes. Using separate node pool is recommended for the NetBackup servers, MSDP Scaleout deployments and for different media server objects. It is required to have separate node pool for Snapshot Manager data plane.
- Taints are set on the node pool while creating the node pool in the cluster. Tolerations are set on the pods.
- Enable AKS Uptime SLA. AKS Uptime SLA is recommended for a better resiliency. For information about AKS Uptime SLA and to enable it, see 'Azure Kubernetes Service (AKS) Uptime SLA' section in *Azure Kubernetes Service Documentation*.
- Access to a container registry that the Kubernetes cluster can access, like an Azure Kubernetes Service Container Registry.

EKS-specific

- A Kubernetes cluster in Amazon Elastic Kubernetes Service in EKS with multiple nodes. Using separate node group is recommended for the NetBackup servers, MSDP Scaleout deployments and for different media server objects. It is required to have separate node pool for Snapshot Manager data plane.
- Taints are set on the node group while creating the node group in the cluster. Tolerations are set on the pods.
- Access to a container registry that the Kubernetes cluster can access, like an Amazon Elastic Kubernetes Service Container Registry.
- AWS network load balancer controller add-on must be installed for using network load balancer capabilities.
- AWS EFS-CSI driver must be installed for statically provisioning the PV or PVC in EFS for primary server.

For more information on installing the load balancer add-on controller and EFS-CSI driver, See "About the Load Balancer service" on page 205.

Prerequisites for using private registry

For registries that require repositories to be created manually before pushing images, ensure that the necessary repositories are set up in advance. Additionally, when using a private registry with your Cloud Scale deployment, you must manually create a registry access secret in each relevant namespace, such as the operator and environment namespaces, to enable successful pushing and pulling of images and Helm charts.

To deploy trust-manager using private registry

1 Create kubernetes secret in trust-manager namespace using the following command:

```
kubectl create secret docker-registry demo-secret --namespace
trust-manager
--docker-server=nbk8s-bo.nbartifactory.rsv.ven.cohesity.com
--docker-username=<username_for_registry>
--docker-password=<password_for_registry>
```

2 Run the following command to download the helm chart:

```
helm repo add jetstack https://charts.jetstack.io
```

```
helm repo update
```

```
helm pull jetstack/trust-manager --version v0.19.0 (Download the
Helm chart for trust-manager version v0.19.0 locally.)
```

3 Push the required images to your private registry:

Note: Ensure that the docker login is done for the private registry prior to pushing the images.

```
Syntax (pull, tag, push):

docker pull <source-registry>/<image-name>:<tag>

docker tag <source-registry>/<image-name>:<tag>
<target-registry>/<image-name>:<tag>

docker push <target-registry>/<image-name>:<tag>
```

For example:

```
docker pull
quay.io/jetstack/cert-manager-package-debian:20210119.0
docker tag quay.io/jetstack/cert-manager-package-debian:20210119.0

nbk8s-bo.nbartifactory.rsv.ven.cohesity.com/cert-manager-package-debian:20210119.0
docker push
nbk8s-bo.nbartifactory.rsv.ven.cohesity.com/cert-manager-package-debian:20210119.0


docker pull quay.io/jetstack/trust-manager:v0.19.0
docker tag quay.io/jetstack/trust-manager:v0.19.0
nbk8s-bo.nbartifactory.rsv.ven.cohesity.com/trust-manager:v0.19.0
docker push
nbk8s-bo.nbartifactory.rsv.ven.cohesity.com/trust-manager:v0.19.0
```

4 Deploy trust-manager using the following command:

```
helm upgrade -i -n trust-manager trust-manager
./trust-manager-v0.19.0.tgz \

--set
image.repository=nbk8s-bo.nbartifactory.rsv.ven.cohesity.com/trust-manager
\

--set image.tag=v0.19.0 \

--set imagePullSecrets[0].name=demo-secret \

--set app.trust.namespace=nbux \ --wait
```

5 Run the following commands to list and verify if the trust-manager is installed:

```
helm list -n trust-manager

NAME             NAMESPACE       REVISION  UPDATED
STATUS     CHART                  APP VERSION
trust-manager    trust-manager   1         <Date and Time>
deployed   trust-manager-v0.19.0    v0.19.0

kubectl get pods -n trust-manager

NAME                               READY   STATUS     RESTARTS
AGE
trust-manager-####47cc6c-949v6   1/1     Running     0
26m
```

To deploy cert-manager using private registry

1 Create kubernetes secret in cert-manager namespace using the following command:

```
kubectl create secret docker-registry demo-secret --namespace
cert-manager
--docker-server=nbk8s-bo.nbartifactory.rsv.ven.cohesity.com
--docker-username=<username_for_registry>
--docker-password=<password_for_registry>
```

2 Run the following command to download the helm chart:

```
helm repo add jetstack https://charts.jetstack.io

helm repo update

helm pull jetstack/cert-manager --version 1.18.2 (Download the
Helm chart for cert-manager version v1.18.2 locally.)
```

3 Push the required images to your private registry:

Note: Ensure that the docker login is done for the private registry prior to pushing the images.

```
Syntax (pull, tag, push):

docker pull <source-registry>/<image-name>:<tag>

docker tag <source-registry>/<image-name>:<tag>
<target-registry>/<image-name>:<tag>

docker push <target-registry>/<image-name>:<tag>
```

For example,

```
docker pull quay.io/jetstack/cert-manager-controller:v1.18.2
docker pull quay.io/jetstack/cert-manager-webhook:v1.18.2
docker pull quay.io/jetstack/cert-manager-cainjector:v1.18.2

docker tag quay.io/jetstack/cert-manager-controller:v1.18.2
nbk8s-bo.nbartifactory.rsv.ven.cohesity.com/cert-manager-controller:v1.18.2
docker tag quay.io/jetstack/cert-manager-webhook:v1.18.2
nbk8s-bo.nbartifactory.rsv.ven.cohesity.com/cert-manager-webhook:v1.18.2

docker tag quay.io/jetstack/cert-manager-cainjector:v1.18.2
nbk8s-bo.nbartifactory.rsv.ven.cohesity.com/cert-manager-cainjector:v1.18.2


docker push
nbk8s-bo.nbartifactory.rsv.ven.cohesity.com/cert-manager-controller:v1.18.2

docker push
nbk8s-bo.nbartifactory.rsv.ven.cohesity.com/cert-manager-webhook:v1.18.2

docker push
nbk8s-bo.nbartifactory.rsv.ven.cohesity.com/cert-manager-cainjector:v1.18.2
```

4. Deploy cert-manager using the following command:

```
helm upgrade -i -n cert-manager cert-manager
./cert-manager-v1.18.2.tgz \
--set
image.repository=nbk8s-bo.nbartifactory.rsv.ven.cohesity.com/cert-manager-controller
\
--set image.tag=v1.18.2 \
--set
webhook.image.repository=nbk8s-bo.nbartifactory.rsv.ven.cohesity.com/cert-manager-webhook
\
--set webhook.image.tag=v1.18.2 \
--set
cainjector.image.repository=nbk8s-bo.nbartifactory.rsv.ven.cohesity.com/cert-manager-cainjector
\
--set cainjector.image.tag=v1.18.2 \
--set global.imagePullSecrets[0].name=demo-secret \
--set webhook.timeoutSeconds=30 \
--set installCRDs=true \
--wait
```

5. Run the following commands to list and verify if the cert-manager is installed:

```
helm list -n cert-manager

NAME            NAMESPACE       REVISION   UPDATED
STATUS     CHART                 APP VERSION
cert-manager     cert-manager    1         <Date and Time>
deployed   cert-manager-v1.18.2    v1.18.2
kubectl get pods -n trust-manager

NAME                                READY    STATUS     RESTARTS
 AGE
cert-manager-####4466d-lzgvg         1/1     Running     0
 14m
cert-manager-####d7754-jrd8n         1/1     Running     0
 14m
cert-manager-####d88bb-c79cc         1/1     Running     0
 14m
```

Chapter 3

Recommendations and Limitations

This chapter includes the following topics:

Recommendations of NetBackup deployment on Kubernetes cluster

Note the following recommendations:

- Do not delete the disk linked to PV used in primary server and media server CR deployment. This may lead to data loss.
- Ensure that in one cluster, only one NetBackup operator instance is running.
- Do not edit any Kubernetes resource created as part of primary server and media server custom resource. Update is supported through custom resource update only.
- Detailed primary server custom resource deployment and media server custom resource deployment logs are retrieved from NetBackup operator pod logs using the `kubectl logs <netbackup-operator-pod-name> -c netbackup-operator -n <netbackup operator-namespace>` command .
- Deploy primary server custom resource and media server custom resource in same namespace.

- Ensure that you follow the symbolic link and edit the actual persisted version of the file, if you want to edit a file having a symbolic link in the primary server or media server.
- Specify different block storage based volume to obtain good performance when the `nbdeployutil` utility does not perform well on the following respective storage types based volumes:
  (*AKS-specific*): Azure premium files
  (*EKS-specific*): Amazon elastic files
- Duplication job configuration recommendation:
  While configuring destination storage unit, manually select media servers that are always up, running and would never scale in (by the media server autoscaler). Number of media servers that are always up and running would be same as that of the value mentioned in **minimumReplicas** field in CR.
  When upgrading from older version of NetBackup 10.3, post upgrade ensure that you manually select media servers mentioned in **minimumReplicas** field in CR. If the value of **minimumReplicas** is not specified, the value will be set to the default value of 1.
- Adjust the value of **minimumReplicas** field and **maximum jobs per media servers** managed by the Cloud Scale Technology based on the backup environment and requirements.
- (***AKS-specific***)
  - Use Azure Premium storage for data volume in media server CR.
  - Use Azure Standard storage for log volume in media server CR.
  - For primary server catalog volume, use `Azure premium files` as storage type and for media server volumes, use `managed-disk` as storage type.
  - In case of upgrade and during migration, do not delete the `Azure premium files/Azure disk volume` linked to the old PV which is used in primary server CR deployment until the migration is completed successfully. Else this leads to data loss.
  - Do not skip the Config-Checker utility execution during NetBackup upgrade or data migration.

  (***EKS-specific***)
  - Use AWS Premium storage for data volume in media server CR.
  - Use AWS Standard storage for log volume in media server CR.
  - For primary server volume (catalog), use `Amazon EFS` as storage type. For media server, primary server volumes, log and data volumes use `Amazon EBS` as storage type.

- In case of upgrade and during migration, do not delete the `Amazon elastic files` linked to the old PV which is used in primary server CR deployment until the migration is completed successfully. Else this leads to data loss.

Limitations of NetBackup deployment on Kubernetes cluster

Note the following limitations:

- External Certificate Authority (ECA) is supported Web UI only (port 443).
  For more information on how to configure ECA for Web UI port 443, refer to the following section:
  See "Configuring an External Certificate Authority for Web UI port 443" on page 80.
- In case of load balancer service updating the CR with dynamic IP address to static IP address and vice versa is not allowed.
- Media server pods as NetBackup storage targets are not supported. For example, NetBackup storage targets like AdvancedDisk and so on are not supported on the media server pods.
- NetBackup Access Control (NBAC) and Enhanced Auditing (EA) configuration are not supported.
- Changes to the CorePattern which specifies the path used for storing core dump files in case of a crash are not supported. CorePattern can only be set during initial deployment.

AKS-specific

- As per Microsoft, since the NFS file share is in Premium account, the minimum file share size is 100GB. If you create a PVC with a small storage size, the following error message is displayed:
  `"failed to create file share ... size (5)...".`
- The IP displayed is of POD IP. NetBackup installing on container takes the IP address of POD as configured IP. Even logs would have POD IP reference at many places.
- (*Applicable only for media servers*) A storage class that has the storage type as `Azure file` is not supported. When the Config-Checker runs the validation for checking the storage type, the Config-Checker job fails if it detects the storage type as `Azure file`. But if the Config-Checker is skipped then this validation is not run, and there can be issues in the deployment. There is no workaround

available for this limitation. You must clean up the PVCs and CRs and reapply the CRs.

- Only `NFS` is supported as the protocol while performing data migration with `Azure premium files`.

EKS-specific

- (*Applicable only for media servers*) A storage class that has the storage type as `EFS` is not supported. When the Config-Checker runs the validation for checking the storage type, the Config-Checker job fails if it detects the storage type as `EFS`. But if the Config-Checker is skipped then this validation is not run, and there can be issues in the deployment. There is no workaround available for this limitation. You must clean up the PVCs and CRs and reapply the CRs.

Recommendations and limitations for Cloud Scale deployment

This section provides the list of recommendations and limitations for Cloud Scale to be noted.

Recommendations

- Ensure that NetBackup clients or media servers outside AKS/EKS cluster are DNS resolvable with NetBackup Primary server load balancer FQDN:
  - *For AKS*: For the DNS name, you can add DNS entries in Private DNS.
  - *For EKS*: For the DNS name, you can use the Private IP DNS name amazon provided, or you can create DNS and Reverse DNS entries under Route53.
- Ensure that you do not use primary server as a media server for use cases where data movement is required.
- To use a cloud bucket as storage, it is recommended to create a new DiskPool with the cloud volume or bucket configured under the `msdpscaleout` storage server.

Limitations

- Cloud Scale does not support **Native Multifactor authentication**.
- **DaemonSet scheduling**:
  With NetBackup 10.5 or later, fluentbit DaemonSet's collect every pod’s `stdout/stderr` logs for the nodes that the DaemonSets are scheduled on. However due to scheduling, some nodes would not get a DaemonSet pod on it due to the scheduling criteria (of having a major NetBackup pod on it). These pods on nodes without a schedule DaemonSet will not have their logs collected.

These logs are non-crucial logs (but important to be aware of) such as infrastructure logs that you encounter in different configurations. However those logs are still available via standard Kubernetes log commands.

- **Fluentbit TLS errors**:
  Following are the sample occasional logs that are commonly viewed on the fluentbit collector pod:

  ```
  [error] [input:forward:forward.0] could not accept new connection
  [error] [tls] error: unexpected EOF
  [error] [input:forward:forward.0] could not accept new connection
  [error] [tls] error: unexpected EOF
  [error] [input:forward:forward.0] could not accept new connection
  [error] [tls] error: unexpected EOF

  And this:
  [error] [/src/fluent-bit/src/tls/openssl.c:433 errno=104]
  Connection reset by peer
  [error] [tls] syscall error: error:00000005:lib(0):func(0):DH lib
  [error] [/src/fluent-bit/src/tls/openssl.c:433 errno=104]
  Connection reset by peer
  [error] [tls] syscall error: error:00000005:lib(0):func(0):DH lib
  [error] [/src/fluent-bit/src/tls/openssl.c:433 errno=104]
  Connection reset by peer
  [error] [tls] syscall error: error:00000005:lib(0):func(0):DH lib
  ```

- **Connection refused**:
  Following are the sample occasional error messages that appear on startup of the fluentbit DaemonSet's and sidecars:

  ```
  [warn] [net] getaddrinfo(host='nb-fluentbit-collector-svc', err=4):
   Domain name not found
  [warn] [net] getaddrinfo(host='nb-fluentbit-collector-svc', err=4):
   Domain name not found
  [error] [output:forward:forward.0] no upstream connections
  available
  [error] [output:forward:forward.0] no upstream connections
  available
  ```

  The above error messages are due to collector having networking issues or not being fully up and accepting connections yet. These messages can be ignored if they are brief and not continuous.

- Veritas Cloud Scale Technology deployment does not support DNAS.

Chapter 4

Configurations

This chapter includes the following topics:

- Contents of the TAR file
- Configuring the cloudscale-values.yaml file
- Configuring an External Certificate Authority for Web UI port 443
- Loading docker images
- Configuring NetBackup

Contents of the TAR file

Download the TAR file from the Cohesity download center.

The TAR file contains the following:

Table 4-1 TAR contents

Item	Description
OCI images in the `/images` directory	These docker image files that are loaded and then copied to the container registry to run in Kubernetes. They include NetBackup and MSDP Scaleout application images and the operator images.
`/helm` directory	Helm file.
Script files at `/scripts` directory	Contains `prep_operators_for_uprade.sh` script and patch file to be used during upgrades.
Sample product (.yaml) files at `/samples` directory	You can use these as templates to define your NetBackup environment.

Table 4-1 TAR contents *(continued)*

Item	Description
`README.md`	Readme file.
MSDP kubectl plug-in at `/bin/kubectl-msdp`	Used to deploy MSDP Scaleout separately without NetBackup operator. **Note:** Used for troubleshooting issues only.
Kubectl Cloud Scale plugin at `/bin/kubectl-cloudscale`	The binary to be used while deploying/upgrading/uninstalling Cloud Scale.

Configuring the `cloudscale-values.yaml` file

The `cloudscale-values.yaml` file lets you configure the postgresql (if using containerized postgres), fluentbit log collector, primary server, media servers, scale out MSDP Scaleout storage and Snapshot Manager servers. The file contains four sections, the first section is global - contains some common parameters, the second for postgresql, third for fluentbit, and fourth for the environment - which contains subsections for the primary, media, MSDP Scaleouts and Snapshot Manager servers.

Note: During configuration, ensure that the following restricted keywords are not used for the Environment custom resource name or other component name:

`primary`, `netbackup`, `media`, `msdp`, `cpserver`, `nbsm` or `flexsnap`

Installation may fail if the above keywords are used in the configuration.

The following configurations apply to all the components:

Table 4-2 Common environment parameters

Parameter	Description
name: **cloudscale-sample**	Specify the name of the cloudscale in your cluster.
namespace: **example-ns**	Specify the namespace where all the NetBackup resources are managed.

Table 4-2 Common environment parameters *(continued)*

Parameter	Description
(*AKS-specific*) containerRegistry: **example.azurecr.io** (*EKS-specific*) containerRegistry: **example.dkr.ecr.us-east-2 .amazonaws.com/exampleReg**	Specify a container registry that the cluster has access. NetBackup images are pushed to this registry.
imagePullSecrets	A comma-separated list of secret references used for pulling images from registries. For example: imagePullSecrets: **my-registry-secret**
tag: **11.2**	This tag is used for all images in the environment. Specifying a \`tag\` value on a sub-resource affects the images for that sub-resource only. For example, if you apply an EEB that affects only primary servers, you might set the \`primary.tag\` to the custom tag of that EEB. The primary server runs with that image, but the media servers and MSDP scaleouts continue to run images tagged \`11.0.0.1\`. Beware that the values that look like numbers are treated as numbers in YAML even though this field needs to be a string; quote this to avoid misinterpretation.
paused: **false**	Specify whether the NetBackup operator attempts to reconcile the differences between this YAML specification and the current Kubernetes cluster state. Only set it to true during maintenance.
configCheckMode: **default**	This controls whether certain configuration restrictions are checked or enforced during setup. Other allowed values are skip and dryrun.
corePattern: **/corefiles/core.%e.%p.%t**	Specify the path to use for storing core files in case of a crash.
(*AKS-specific*) loadBalancerAnnotations: service. beta.kubernetes.io/ azure-load-balancer- internal-subnet: *example-subnet* (*EKS-specific*) loadBalancerAnnotations: service.beta.kubernetes.io/aws-load-balancer-subnets: *example-subnet1 name*	Specify the annotations to be added for the network load balancer
emailServerConfigmapName	(*Optional*) Name of the configmap that contains required details to configure the email server in NetBackup.
drInfoSecretName	(*Optional*) Name of secret created to pass DR information such as **passphrase** and **e-mail address**. If user has not provided this secret, default catalog backup policy will not be created automatically by the NetBackup Operator.

Table 4-2 Common environment parameters *(continued)*

Parameter	Description
dbSecretName	Specify the name of the secret required for deployment of PostgreSQL as a container. This secret is created as part of the Helm installation of PostgreSQL container.
dbSecretProviderClass	(*Optional*) Specify the name of the SecretProvider class required for DBaaS deployment of PostgreSQL.

Note: (*EKS-specific*) If NetBackup is upgraded form 10.0.0.1, then delete the following configuration from the `cloudscale-values.yaml` file from **spec.loadBalancerAnnotations** section:

`service.beta.kubernetes.io/aws-load-balancer-target-group-attributes: preserve_client_ip.enabled=true`

The following section describes Snapshot Manager related parameters. You may also deploy without any Snapshot Manager. In that case, remove the cpServer section entirely from the configuration file.

Table 4-3 Snapshot Manager parameters

Parameter	Description
cpServer: -name	This specifies Snapshot Manager configurations. Currently only single instance of Snapshot Manager deployment is supported. It is also possible to have no Snapshot Managers configured; in this case, delete the cpServer section itself.
containerRegistry	(*Optional*) Specify a container registry that the cluster has access. Snapshot Manager images are pushed to this registry which overrides the one defined in **Common environment parameters** table above.
tag:	This tag overrides the one defined in **Common environment parameters** table above. The Snapshot Manager images are shipped with tags different from the NetBackup primary, media, and MSDP images.
networkLoadBalancer: annotations	Annotations to be provided to the network load balancer. All networkLoadBalancer annotations are supported. These values are merged with the values provided in the **loadBalancerAnnotations**. The duplicate values provided here, override the corresponding values in the **loadBalancerAnnotations**.
networkLoadBalancer: ipaddr	IP address to be assigned to the network load balancer.
networkLoadBalancer: fqdn	FQDN to be assigned to the network load balancer.

Table 4-3 Snapshot Manager parameters *(continued)*

Parameter	Description
log.capacity	Size for log volume.
log.storageClassName	Storage class for log volume. It must be EFS based storage class.
data.capacity	Size for data volume.
data.storageClassName	EBS based storage class for data volume.
controlPlane.nodePool	Name of the control plane node pool.
controlPlane.labelKey	Label and taint key of the control plane.
controlPlane.labeValue	Label and taint value of the control plane.
dataPlane.nodePool	Name of the data plane node pool.
dataPlane.labelKey	Label and taint key of the data plane.
dataPlane.labelValue	Label and taint value of the data plane.
proxySettings: vx_http_proxy:	Address to be used as the proxy for all HTTP connections. For example, `"http://proxy.example.com:8080/"`
proxySettings: vx_https_proxy:	Address to be used as the proxy for all HTTPS connections. For example, `"http://proxy.example.com:8080/"`
proxySettings: vx_no_proxy:	Address that are allowed to bypass the proxy server. You can specify host name, IP addresses and domain names in this parameter. For example, `"localhost,mycompany.com,169.254.169.254"`

The following configurations apply to the primary server. The values specified in the following table can override the values specified in the table above.

Table 4-4 Environment parameters for the primary server

Paragraph	Description
paused: *false*	Specifies whether the NetBackup operator attempts to reconcile the differences between this YAML specification and the current Kubernetes cluster state. Set it to *true* only during maintenance. This applies only to the primary server object. To pause reconciliation of the managed primary server, for example, you must set `spec.primary.paused`. Setting `spec.paused:true` ceases updates to the managed resources, including updates to their `paused` status. Entries in the media servers and MSDP scaleouts lists also support the `paused` field. The default value is *false*.
primary	Specifies attributes specific to the primary server resources. Every environment has exactly one primary server, so this section cannot be left blank.
name: primary-name	Set **resourceNamePrefix** to control the name of the primary server. The default value is the same as the environment's name.
tag: **11.2-*build_number***	To use a different image tag specifically for the primary server, uncomment this value and provide the desired tag. This overrides the tag specified in the common section.
nodeSelector: labelKey: kubernetes.io/os labelValue: linux	Specify a key and value that identifies nodes where the primary server pod runs. **Note:** This **labelKey** and **labelValue** must be the same label key:value pair used during cloud node creation which would be used as a toleration for primary server.

networkLoadBalancer:
(*AKS-specific*) annotations: service.beta. kubernetes.io / azure-load- balancer-internal- subnet: example- subnet
(*EKS-specific*) annotations: service.beta.kubernetes.io/aws-load-balancer-subnets: example-subnet1 name
`ipList:`
`        - ipAddr: 4.3.2.1`
`          fqdn: primary.example.com`
	Uncomment the annotations to specify additional primary server-specific annotations. These values are merged with the values given in the loadBalancerAnnotations above. Any duplicate values given here override the corresponding values above.
	Next, specify the hostname and IP address of the primary server.

Table 4-4 Environment parameters for the primary server *(continued)*

Paragraph	Description
credSecretName: **primary-credential-secret**	This determines the credentials for the primary server. Media servers use these credentials to register themselves with the primary server.
kmsDBSecret: **kms-secret**	Secret name which contains the Host Master Key ID (HMKID), Host Master Key passphrase (HMKpassphrase), Key Protection Key ID (KPKID) and Key Protection Key passphrase (KPKpassphrase) for NetBackup Key Management Service.
(*AKS specific*) autoVolumeExpansion	Enables automatic monitoring of the catalog volume when set to true. For more information, see Reducing catalog storage management.
catalog: capacity: *100Gi* (*AKS-specific*) storageClassName: **standard** (*EKS-specific*) storageClassName: **<EFS_ID>**	This storage applies to the primary server for the NetBackup catalog, log and data volumes. The primary server catalog volume must be at least 100 Gi.
log: capacity: **30Gi** (*AKS-specific*) storageClassName: **standard** (*EKS-specific*) storageClassName: **<EBS based storage class>**	Log volume must be at least 30Gi.
data: capacity: **30Gi** (*AKS-specific*) storageClassName: **standard** (*EKS-specific*) storageClassName: **<EBS based storage class>**	The primary server data volume must be at least 30Gi. **Note:** (*AKS-specific*) This storage applies to primary server data volume.

The following section describes the media server configurations. If you do not have a media server either remove this section from the configuration file entirely, or define it as an empty list.

Note: (*EKS-specific*) The environment name or media server name in `cloudscale-values.yaml` file must always be less than 22 characters.

Table 4-5 Media server related parameters

Parameters	Description
mediaServers: - name: media1	This specifies media server configurations. This is given as a list of media servers, but most environments will have just one, with multiple replicas. It's also possible to have zero media servers; in that case, either remove the media servers section entirely, or define it as an empty list: mediaServers: []
minimumReplicas	Describes the minimum number of replicas of the running media server. This is an optional field. If not specified, default value is 1. When **minimumReplica = 0**, user must change the minimum size of media nodepool to 0 through the portal. NetBackup 11.0 and later, now provides support for scaling down the **minimumReplica** value of media server custom resource to 0 to diverge from the default behavior. For more information refer to the following section: See “Elastic media server” on page 99.
replicas: *2*	Specifies the maximum number of replicas that the media server can scale up to. The value of replicas must be greater than the value of **minimumReplicas** for media autoscaler to work.
tag: **11.2-*build_number***	To use a different image tag specifically for the media servers, uncomment this value and provide the desired tag. This overrides the tag specified above in the common table.
nodeSelector: labelKey: *kubernetes.io/os* labelValue: *linux*	Specify a key and value that identifies nodes where media-server pods will run. **Note:** This **labelKey** and **labelValue** must be the same label key:value pair used during cloud node creation which would be used as a toleration for media server.
data: capacity: *50Gi* (*AKS-specific*) storageClassName: *managed-premium-nbux* (*EKS-specific*) storageClassName: *<EBS based storage class>*	This storage applies to the media server data volumes. The minimum data size for a media server is 50 Gi.

Table 4-5 Media server related parameters *(continued)*

Parameters	Description
log capacity: *30Gi* (*AKS-specific*) storageClassName: *managed-premium-nbux* (*EKS-specific*) storageClassName: *<EBS based storage class>*	This storage applies to the media server log volumes. Log volumes must be at least 30Gi.

(*EKS-specific*) **Note the following:**

To use gp3 (EBS based storage class), user must specify provisioner for storage class as `ebs.csi.aws.com` and must install EBS CSI driver. For more information on installing the EBS CSI driver, see Amazon EBS CSI driver. Example, for gp3 storage class:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: gp3
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
allowVolumeExpansion: true
provisioner: ebs.csi.aws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
```

The following section describes MSDP-related parameters. You may also deploy without any MSDP scaleouts. In that case, remove the msdpScaleouts section entirely from the configuration file.

Table 4-6 MSDP Scaleout related parameters

Parameter	Description
msdpScaleouts: - name: dedupe1	This specifies MSDP Scaleout configurations. This is given as a list, but it would be rare to need more than one scaleout deployment in a single environment. Use the `replicas` property below to scale out. It's also possible to have zero MSDP scaleouts; in that case, either remove the msdpScaleouts section entirely, or define it to an empty list: msdpScaleouts: []

Table 4-6 MSDP Scaleout related parameters *(continued)*

Parameter	Description
tag: '*21.1*'	This tag overrides the one defined in the table 1-3. It is necessary because the MSDP Scaleout images are shipped with tags different from the NetBackup primary and media images.
replicas: *1*	This is the scaleout size of this MSDP Scaleout component. It is a required value, and it must be between 4 and 16 inclusive. **Note:** Scale-down of the MSDP Scaleout replicas after deployment is not supported.
serviceIPFQDNs: ipAddr: 1.2.3.4 fqdn: dedupe1-1.example.com	These are the IP addresses and host names of the MSDP Scaleout servers. The number of the entries should match the number of the replicas specified above.
kms: keyGroup: *example-key-group*	Specifies the initial key group and key secret to be used for KMS encryption. When reusing storage from a previous deployment, the key group and key secret may already exist. In this case, provide the keyGroup only.
keySecret: *example-key-secret*	Specify keySecret only if the key group does not already exist.
(*AKS-specific*) loadBalancerAnnotations: service.beta.kubernetes .io/azure-load- balancer-internal: *true* (*EKS-specific*) loadBalancerAnnotations: service.beta.kubernetes .io/aws-load- balancer-internal: *true*	For MSDP scaleouts, the default value for the following annotation is \`false\`, which may cause the MSDP Scaleout services in this Environment to be accessible publicly: (*AKS-specific*): Azure-load-balancer-internal (*EKS-specific*): AWS-load-balancer-internal Ensure that they use private IP addresses, specify \`true\` here or in the loadBalancerAnnotations above in Table 1-3.
credential: secretName: *msdp-secret1*	This defines the credentials for the MSDP Scaleout server. It refers to a secret in the same namespace as this environment resource. Secret can be either of type 'Basic-auth' or 'Opaque'.

Table 4-6 MSDP Scaleout related parameters *(continued)*

Parameter	Description
autoDelete: *false*	Optional parameter. Default value is true. When set to true, the MSDP Scaleout operator deletes the MSDP secret after using it. In such case, the MSDP and primary secrets must be distinct. To use the same secret for both MSDP scaleouts and the primary server, set autoDelete to false.
catalog: capacity: *1Gi* (*AKS-specific*) storageClassName: *standard* (*EKS-specific*) storageClassName: *gp2*	This storage applies to MSDP Scaleout to store the catalog and metadata. The catalog size may only be increased for capacity expansion. Expanding the existing catalog volumes cause short downtime of the engines. Recommended size is 1/100 of backend data capacity.
dataVolumes: capacity: *5Gi* (*AKS-specific*) storageClassName: *standard* (*EKS-specific*) storageClassName: *gp2*	This specifies the data storage for this MSDP Scaleout resource. You may increase the size of a volume or add more volumes to the end of the list, but do not remove or re-order volumes. Maximum 16 volumes are allowed. Appending new data volumes or expanding existing ones will cause short downtime of the Engines. Recommended volume size is 5Gi-32Ti.
log: capacity: *20Gi* (*AKS-specific*) storageClassName: *standard* (*EKS-specific*) storageClassName: *gp2*	Specifies log volume size used to provision Persistent Volume Claim for Controller and MDS Pods. In most cases, 5-10 Gi capacity should be big enough for one MDS or Controller Pod to use.
nodeSelector: labelKey: *kubernetes.io/os* labelValue: *linux*	Specify a key and value that identifies nodes where MSDP Scaleout pods will run.
s3Credential: secretName: *s3-secret1*	This is an optional parameter. Defines the MSDP S3 root credentials for the MSDP Scaleout server. It refers to a secret in the same namespace as this environment resource. If the parameter is not specified, MSDP S3 feature is unavailable.
autoDelete: *false*	This is an optional parameter. Default value is true. When set to true, the MSDP Scaleout operator deletes the MSDP S3 credential secret after using it.

Table 4-6 MSDP Scaleout related parameters *(continued)*

Parameter	Description
s3Ip: ipAddr: 1.2.3.8 fqdn: dedupe1-s3.example.com	These are optional parameters. The IP address and host name of the S3 load balancer service. If the parameter is not specified, MSDP S3 feature is unavailable.

For more information on Snapshot Manager related parameters, refer to the following:

See "Installing the docker images for Snapshot Manager" on page 93.

Edit restricted parameters post deployment

Do not change these parameters post initial deployment. Changing these parameters may result in an inconsistent deployment.

Table 4-7 Edit restricted parameters post deployment

Parameter	Description
name	Specifies the prefix name for the primary, media, and MSDP Scaleout server resources.

(*AKS-specific*)
ipAddr, fqdn and
loadBalancerAnnotations
	The values against `ipAddr`, `fqdn` and `loadBalancerAnnotations` against following fields should not be changed post initial deployment. This is applicable for primary and MSDP Scaleout servers. For example:
	`- The loadBalancerAnnotations for loadBalancerAnnotations:`
	`service.beta.kubernetes.io/azure-load-balancer`
	`-internal-subnet: example-subnet`
	`service.beta.kubernetes.io/azure-load-balancer -internal:`
	`"true"`
	`The IP and FQDNs values defined for Primary, Media and`
	`MSDPScaleout ipList:`
	`  - ipAddr: 4.3.2.1  fqdn: primary.example.com`
	
	`serviceIPFQDNs:`
	`  - ipAddr: 1.2.3.4  fqdn: dedupe1-1.example.com`
	`  - ipAddr: 1.2.3.5  fqdn: dedupe1-2.example.com`
	`  - ipAddr: 1.2.3.6  fqdn: dedupe1-3.example.com`
	`  - ipAddr: 1.2.3.7  fqdn: dedupe1-4.example.com`

Table 4-7 Edit restricted parameters post deployment *(continued)*

Parameter	Description

(*EKS-specific*)
ipAddr, fqdn and
loadBalancerAnnotations
	The values against `ipAddr`, `fqdn` and `loadBalancerAnnotations` against following fields should not be changed post initial deployment. This is applicable for primary and MSDP Scaleout servers. For example:
	
	`- The loadBalancerAnnotations for loadBalancerAnnotations: service.beta.kubernetes.io/aws-load-balancer -internal-subnet: example-subnet service.beta.kubernetes.io/aws-load-balancer -internal: "true"`
	`- The IP and FQDNs values defined for Primary, Media and MSDPScaleout ipList:`
	`   - ipAddr: 4.3.2.1   fqdn: primary.example.com`
	
	`serviceIPFQDNs:`
	`   - ipAddr: 1.2.3.4   fqdn: dedupe1-1.example.com`
	`   - ipAddr: 1.2.3.5   fqdn: dedupe1-2.example.com`
	`   - ipAddr: 1.2.3.6   fqdn: dedupe1-3.example.com`
	`   - ipAddr: 1.2.3.7   fqdn: dedupe1-4.example.com`

Table 4-8 Snapshot Manager server related parameters

parameters	Description
nodeSelector: controlPlane: *cpcontrol* dataPlane: *cpdata*	Details of the label to be used for identification of Kubernetes nodes reserved for the Snapshot Manager Servers. If controlPlane is not specified here it will read it from *primary.nodeSelector*. In that case the nodepool should have appropriate taint and label added to it.[the nodepool name mentioned will have value of primary.nodeSelector.labelValue] **Note:** The nodepool name mentioned will have value of *primary.nodeSelector.labelValue*. The `flexsnap-listener` pod of the Snapshot Manager is migrated from dataPlane node pool (cpdata) to controlPlane node pool (primary). This reduces the requirement of separate node initially. The minimum node count for cpdata node pool is set to 0. As this node pool is not used initially.

Table 4-8 Snapshot Manager server related parameters *(continued)*

parameters	Description
data: capacity: *100Gi* (*AKS-specific*) storageClassName: *managed-premium* (*EKS-specific*) storageClassName: *<EBS based storage class>*	This storage applies to the Snapshot Manager server data volumes. The minimum data size for a Snapshot Manager server is 100 Gi.
log capacity: *5Gi* (*AKS-specific*) storageClassName: *managed-premium* (*EKS-specific*) storageClassName: *<EBS based storage class>*	This storage applies to the Snapshot Manager server log volumes. Log volumes must be at least 5 Gi.
networkLoadBalancer: (*AKS-specific*) annotations: - service.beta.kubernetes.io/ azure-load-balancer -internal-subnet: *example-subnet* (*EKS-specific*) *annotations: -service.beta.kubernetes.io/aws-load-balancer-subnets: example-subnet1 name* ipList: ipAddr: *4.3.2.2* fqdn: *media1-1.example.com* ipAddr: *4.3.2.3* cpServer: *media1-2.example.com*	Snapshot ManagerUncomment annotations to specify additional -server specific annotations. These values are merged with the values given in the spec.loadBalancerAnnotations. The duplicate values given here, override the corresponding values in the spec.loadBalancerAnnotations.

Configuring an External Certificate Authority for Web UI port 443

During Cloud Scale installation, a Cloud Scale environment is configured to use a certificate issued by the NetBackup Certificate Authority for Web UI (port 443). The following sections provide the steps to replace the default certificate with an External Certificate Authority (ECA) on Cloud Scale.

Prerequisites

Ensure that all the following files are available and ready before proceeding:

- `ca.pem`: A PEM-formatted file, containing the Root CA certificate from which the Web UI certificate was issued.
- `cert.pem`: A PEM-formatted file, that includes the Web UI certificate chain, consisting of the leaf certificate and any intermediate CA certificates.
- `privatekey.pem`: PKCS #8 PEM-formatted file, containing the encrypted private key for the Web UI certificate.
- `passphrase.txt,`: A plaintext file, containing the passphrase used to encrypt the private key. Ensure that the plaintext file contains only a single line with the passphrase and does not end with a new line.

Basic sanity verification for `ca.pem`**,** `cert.pem`**,** `privatekey.pem` **and** `passphrase.txt,` **files:**

1. Execute the following commands and ensure that the output of both the commands result in a matching value for `cert.pem` and `privatekey.pem` files:

   ```
   $ openssl x509 -noout -modulus -in cert.pem | openssl sha256
   ```

   ```
   $ openssl rsa -noout -modulus -in privatekey.pem -passin
   file:passphrase.txt | openssl sha256
   ```

2. Check the validity of the certificate and presence of the primary server name in the certificate by executing the following command to list the certificate details:

   ```
   $ openssl x509 -text -in cert.pem -noout
   ```

   Verify the **Not Before** and **Not After** fields display the certificate is valid and has not expired. Confirm that the primary server name appears in the **X509v3 Subject Alternative Name**. If **X509v3 Subject Alternative Name** is missing, then confirm that the primary server name appears in the **Subject** as a **CN**.

3. Execute the following command to verify that a complete certificate chain exists in `cert.pem` and `ca.pem` files:

   ```
   $ openssl verify -CAfile ca.pem -untrusted cert.pem cert.pem

   cert.pem: OK
   ```

Configuring an external certificate

Perform the steps described in the following procedure to configure external certificate for the first time.

Configure an external certificate for the first time

1. Log in to the host where the Kubernetes cluster is managed and has the `kubectl` command.
2. Execute the following commands to create the `tpcredentials` directory and copy the artifacts into the primary pod:

   ```
   $ kubectl exec -it <primary_pod_name> -n <namespace> -- mkdir -p
   /usr/openv/var/global/wsl/credentials/tpcredentials
   ```

   ```
   $ kubectl cp <path_to_ca.pem>
   <primary_pod_name>:/usr/openv/var/global/wsl/credentials/tpcredentials/ca.pem
   -n <namespace>
   ```

   ```
   $ kubectl cp <path_to_cert.pem>
   <primary_pod_name>:/usr/openv/var/global/wsl/credentials/tpcredentials/cert.pem
   -n <namespace>
   ```

   ```
   $ kubectl cp <path_to_privatekey.pem>
   <primary_pod_name>:/usr/openv/var/global/wsl/credentials/tpcredentials/privatekey.pem
   -n <namespace>
   ```

   ```
   $ kubectl cp <path_to_passphrase.txt>
   <primary_pod_name>:/usr/openv/var/global/wsl/credentials/tpcredentials/passphrase.txt
   -n <namespace>
   ```

3. Execute the following commands to ensure the directory and files are created with correct ownership. Ensure that the owner has full permissions on the directory and files.

   ```
   $ kubectl exec -it <primary_pod_name> -n <namespace> -- chown -R
   nbwebsvc:nbwebgrp
   /usr/openv/var/global/wsl/credentials/tpcredentials
   ```

   ```
   $ kubectl exec -it <primary_pod_name> -n <namespace> -- chmod
   u+rwx /usr/openv/var/global/wsl/credentials/tpcredentials
   ```

   ```
   $ kubectl exec -it <primary_pod_name> -n <namespace> -- chmod -R
   u+rw /usr/openv/var/global/wsl/credentials/tpcredentials
   ```

4. Restart the **nbwsapp** pod.

   Use the following commands to list all the pods from the namespace, then select the **nbwsapp** pod and delete it.

   ```
   $ kubectl get pods -n <namespace>
   ```

   ```
   $ kubectl delete pod <nbwsapp_pod_name> -n <namespace>
   ```

5 Use the following command to ensure that the **nbwsapp** pod is up and running:

```
$ kubectl get pods -n <namespace> | grep nbwsapp

<nbwsapp_pod_name>                       4/4     Running   0
       9m44s
```

6 Execute the following commands to create the keystore:

```
$ kubectl exec -it <nbwsapp_pod_name> -n <namespace> -- bash
```

```
$ cp
/usr/openv/var/global/wsl/credentials/tpcredentials/passphrase.txt
/usr/openv/var/global/wsl/credentials/tpcredentials/jkskey
```

```
$ /usr/openv/netbackup/bin/goodies/vxsslcmd pkcs12 -export -inkey
/usr/openv/var/global/wsl/credentials/tpcredentials/privatekey.pem
-in /usr/openv/var/global/wsl/credentials/tpcredentials/cert.pem
-out /tmp/cert.p12 -passin
file:/usr/openv/var/global/wsl/credentials/tpcredentials/passphrase.txt
-passout
file:/usr/openv/var/global/wsl/credentials/tpcredentials/jkskey
-name eca
```

Note: Ignore the following error message if displayed by the last command above:

```
unable to write 'random state'
```

```
$ ls -l /tmp/cert.p12
```

```
-rw-r--r-- 1 nbwebsvc nbwebgrp   4420 Sep 20 19:44 cert.p12
```

```
$ export KEYSTORE_PASS=$(cat
/usr/openv/var/global/wsl/credentials/tpcredentials/jkskey)
```

```
$ /usr/openv/java/jre/bin/keytool -storetype BCFKS -providerpath
/usr/openv/wmc/webserver/lib/ccj.jar -providerclass
com.safelogic.cryptocomply.jcajce.provider.CryptoComplyFipsProvider
-importkeystore -srckeystore /tmp/cert.p12 -srcstoretype pkcs12
-srcstorepass ${KEYSTORE_PASS} -destkeystore
/tmp/nbwebservice.bcfks -deststorepass ${KEYSTORE_PASS}
```

```
Importing keystore /tmp/cert.p12 to /tmp/nbwebservice.bcfks...
Entry for alias eca successfully imported.
Import command completed:  1 entries successfully imported, 0
entries failed or cancelled
```

```
$ mv /tmp/nbwebservice.bcfks
/usr/openv/var/global/wsl/credentials/tpcredentials/
```

```
keytool -storetype BCFKS -providerpath
/usr/openv/wmc/webserver/lib/ccj.jar -providerclass
```

```
com.safelogic.cryptocomply.jcajce.provider.CryptoComplyFipsProvider
-importcert -alias ca -file
/usr/openv/var/global/wsl/credentials/tpcredentials/ca.pem
-keystore
/usr/openv/var/global/wsl/credentials/tpcredentials/nbwebservice.bcfks
-srcstorepass ${KEYSTORE_PASS} -deststorepass ${KEYSTORE_PASS}
```

7 The new certificate will be automatically applied within 30 minutes, or you can restart the **requestrouter** pod to apply it immediately by using the following commands:

```
$ kubectl get pods -n <namespace>

$ kubectl delete pod <requestrouter_pod_name> -n <namespace>
```

Renewal or replacement of external certificate

Use the following steps to replace the existing external certificate with a new external certificate:

1. Log in to the host where the Kubernetes cluster is managed and has the `kubectl` command.

2. Execute the following commands to copy the new artifacts into the primary pod:

```
$ kubectl exec -it <primary_pod_name> -n <namespace> -- mkdir -p
/usr/openv/var/global/wsl/credentials/tpcredentials

$ kubectl cp <path_to_ca.pem>
<primary_pod_name>:/usr/openv/var/global/wsl/credentials/tpcredentials/ca.pem
-n <namespace>

$ kubectl cp <path_to_cert.pem>
<primary_pod_name>:/usr/openv/var/global/wsl/credentials/tpcredentials/cert.pem
-n <namespace>

$ kubectl cp <path_to_privatekey.pem>
<primary_pod_name>:/usr/openv/var/global/wsl/credentials/tpcredentials/privatekey.pem
-n <namespace>

$ kubectl cp <path_to_passphrase.txt>
<primary_pod_name>:/usr/openv/var/global/wsl/credentials/tpcredentials/passphrase.txt
-n <namespace>

$ kubectl exec -it <primary_pod_name> -n <namespace> -- mkdir -p
/usr/openv/var/global/wsl/credentials/tpcredentials/backup
```

```
$ kubectl exec -it <primary_pod_name> -n <namespace> -- mv
/usr/openv/var/global/wsl/credentials/tpcredentials/nbwebservice.bcfks
/usr/openv/var/global/wsl/credentials/tpcredentials/backup/
```

3. Execute the below commands to ensure the directory and files are created with correct ownership. Ensure that the owner has full permission on the directory and files.

```
$ kubectl exec -it <primary_pod_name> -n <namespace> -- chown -R
nbwebsvc:nbwebgrp
/usr/openv/var/global/wsl/credentials/tpcredentials
```

```
$ kubectl exec -it <primary_pod_name> -n <namespace> -- chmod
u+rwx /usr/openv/var/global/wsl/credentials/tpcredentials
```

```
$ kubectl exec -it <primary_pod_name> -n <namespace> -- chmod -R
u+rw /usr/openv/var/global/wsl/credentials/tpcredentials
```

4. Restart the **nbwsapp** pod.

   List all the pods from the namespace, select the **nbwsapp** pod and delete it.

```
$ kubectl get pods -n <namespace>
```

```
$ kubectl delete pod <nbwsapp_pod_name> -n <namespace>
```

5. Ensure that **nbwsapp** pod is up and running:

```
$ kubectl get pods -n <namespace> | grep nbwsapp
```

```
<nbwsapp_pod_name>                       4/4     Running   0
       9m44s
```

6. Execute the following commands to create the keystore:

```
$ kubectl exec -it <nbwsapp_pod_name> -n <namespace> -- bash
```

```
$ chmod 2750
/usr/openv/var/global/wsl/credentials/tpcredentials/backup/
```

```
$ /usr/openv/netbackup/bin/goodies/vxsslcmd pkcs12 -export -inkey
/usr/openv/var/global/wsl/credentials/tpcredentials/privatekey.pem
-in /usr/openv/var/global/wsl/credentials/tpcredentials/cert.pem
-out /tmp/cert.p12 -passin
file:/usr/openv/var/global/wsl/credentials/tpcredentials/passphrase.txt
-passout
file:/usr/openv/var/global/wsl/credentials/tpcredentials/jkskey
-name eca
```

Note: Ignore the following message from last command if it is seen:

unable to write 'random state'

```
$ ls -l /tmp/cert.p12
```

```
-rw-r--r-- 1 nbwebsvc nbwebgrp   4420 Sep 20 19:44 cert.p12
```

```
$ export KEYSTORE_PASS=$(cat
/usr/openv/var/global/wsl/credentials/tpcredentials/jkskey)
```

```
$ /usr/openv/java/jre/bin/keytool -storetype BCFKS -providerpath
/usr/openv/wmc/webserver/lib/ccj.jar -providerclass
com.safelogic.cryptocomply.jcajce.provider.CryptoComplyFipsProvider
-importkeystore -srckeystore /tmp/cert.p12 -srcstoretype pkcs12
-srcstorepass ${KEYSTORE_PASS} -destkeystore
/tmp/nbwebservice.bcfks -deststorepass ${KEYSTORE_PASS}
```

```
Importing keystore /tmp/cert.p12 to /tmp/nbwebservice.bcfks...
Entry for alias eca successfully imported.
Import command completed:  1 entries successfully imported, 0
entries failed or cancelled
```

```
$ mv /tmp/nbwebservice.bcfks
/usr/openv/var/global/wsl/credentials/tpcredentials/
```

7. The new certificate will be automatically applied within 30 minutes, or you can restart the **requestrouter** pod to apply it immediately.

```
$ kubectl get pods -n <namespace>
```

```
$ kubectl delete pod <requestrouter_pod_name> -n <namespace>
```

Removing the external certificate

Perform the following steps to remove the external certificate for the Web UI (port 443) and replace it with the NetBackup Certificate Authority issued certificate:

1. Log in to the host where the Kubernetes cluster is managed and has the `kubectl` command.

2. Execute the following commands to remove the ECAs:

```
$ kubectl exec -it <primary_pod_name> -n <namespace> -- rm -f
/usr/openv/var/global/wsl/credentials/tpcredentials/ca.pem
```

```
$ kubectl exec -it <primary_pod_name> -n <namespace> -- rm -f
/usr/openv/var/global/wsl/credentials/tpcredentials/cert.pem
```

```
$ kubectl exec -it <primary_pod_name> -n <namespace> -- rm -f
/usr/openv/var/global/wsl/credentials/tpcredentials/privatekey.pem
```

```
$ kubectl exec -it <primary_pod_name> -n <namespace> -- rm -f
/usr/openv/var/global/wsl/credentials/tpcredentials/passphrase.txt
```

```
$ kubectl exec -it <primary_pod_name> -n <namespace> -- rm -f
/usr/openv/var/global/wsl/credentials/tpcredentials/nbwebservice.bcfks
```

```
$ kubectl exec -it <primary_pod_name> -n <namespace> -- rm -f
/usr/openv/var/global/wsl/credentials/tpcredentials/jkskey
```

```
$ kubectl exec -it <primary_pod_name> -n <namespace> -- rm -rf
/usr/openv/var/global/wsl/credentials/tpcredentials
```

3. Restart the **requestrouter** pod.

   List all of the pods from the namespace, select the **requestrouter** pod and delete it.

   ```
   $ kubectl get pods -n <namespace>
   ```

   ```
   $ kubectl delete pod <requestrouter_pod_name> -n <namespace>
   ```

Note: No certificate configuration is required after completing a disaster recovery of the Cloud Scale environment, as the process automatically restores external certificates.

Loading docker images

This section provides the details for installing the docker images for the NetBackup, Snapshot Manager and MSDP Scaleout.

Installing the docker images for NetBackup

The NetBackup package `VRTSk8s-netbackup-<version>.tar.gz` for Kubernetes includes the following:

- A docker image for NetBackup operator
- Docker images for NetBackup: operator, main, media, mqbroker,nbatd, pbx, vnetd, bpdbm, ws, policyjobmgr, fluentbit, postgresql, log-viewer and requestrouter

To install the docker images

1. Download `VRTSk8s-netbackup-<version>.tar.gz` from the Cohesity site.
2. Run the following commands to load the docker images to the local docker instance:

```
$ docker load -i netbackup-fluentbit-<version>.tar.gz

$ docker load -i netbackup-fluentbit-log-cleanup-<version>.tar.gz

$ docker load -i netbackup-main-<version>.tar.gz

$ docker load -i netbackup-media-<version>.tar.gz

$ docker load -i netbackup-mqbroker-<version>.tar.gz

$ docker load -i netbackup-mqbroker-init-<version>.tar.gz

$ docker load -i netbackup-nbatd-init-<version>.tar.gz

$ docker load -i netbackup-nbatd-main-<version>.tar.gz

$ docker load -i netbackup-nbhousekeeping-<version>.tar.gz

$ docker load -i netbackup-operator-<version>.tar.gz

$ docker load -i netbackup-pbx-<version>.tar.gz

$ docker load -i netbackup-policyjobmgr-<version>.tar.gz

$ docker load -i netbackup-postgresql-<version>.tar.gz

$ docker load -i netbackup-postgresql-upgrade-<version>.tar.gz

$ docker load -i netbackup-requestrouter-<version>.tar.gz

$ docker load -i netbackup-vnetd-<version>.tar.gz

$ docker load -i netbackup-ws-app-<version>.tar.gz

$ docker load -i netbackup-ws-init-<version>.tar.gz

$ docker load -i netbackup-log-viewer-<version>.tar.gz

$ docker load -i netbackup-dbm-<version>.tar.gz
```

   Run the command `docker image ls` command to confirm that the NetBackup images are loaded properly to the docker cache.

   <*version*>: Represents the NetBackup product version.

3 Run the following commands to re-tag the images to associate them with your container registry, keep the image name and version same as original:

(*AKS-specific*): `$ REGISTRY=<example.azurecr.io> (Replace with your own container registry name)`

(*EKS-specific*): `$ REGISTRY=<<AccountID>.dkr.ecr.<region>.amazonaws.com`

```
$ docker tag localhost/netbackup/fluentbit:<version>
${REGISTRY}/netbackup/fluentbit:<version>

$ docker tag localhost/netbackup/fluentbit-log-cleanup:<version>
${REGISTRY}/netbackup/fluentbit-log-cleanup:<version>

$ docker tag localhost/netbackup/main:<version>
${REGISTRY}/netbackup/main:<version>

$ docker tag localhost/netbackup/media:<version>
${REGISTRY}/netbackup/media:<version>

$ docker tag localhost/netbackup/mqbroker:<version>
${REGISTRY}/netbackup/mqbroker:<version>

$ docker tag localhost/netbackup/mqbroker-init:<version>
${REGISTRY}/netbackup/mqbroker-init:<version>

$ docker tag localhost/netbackup/nbatd-init:<version>
${REGISTRY}/netbackup/nbatd-init:<version>

$ docker tag localhost/netbackup/nbatd-main:<version>
${REGISTRY}/netbackup/nbatd-main:<version>

$ docker tag localhost/netbackup/nbhousekeeping:<version>
${REGISTRY}/netbackup/nbhousekeeping:<version>

$ docker tag localhost/netbackup/operator:<version>
${REGISTRY}/netbackup/operator:<version>

$ docker tag localhost/netbackup/pbx:<version>
${REGISTRY}/netbackup/pbx:<version>

$ docker tag localhost/netbackup/policyjobmgr:<version>
${REGISTRY}/netbackup/policyjobmgr:<version>

$ docker tag localhost/netbackup/requestrouter:<version>
${REGISTRY}/netbackup/requestrouter:<version>

$ docker tag localhost/netbackup/vnetd:<version>
${REGISTRY}/netbackup/vnetd:<version>
```

```
$ docker tag localhost/netbackup/ws-app:<version>
${REGISTRY}/netbackup/ws-app:<version>

$ docker tag localhost/netbackup/ws-init:<version>
${REGISTRY}/netbackup/ws-init:<version>

$ docker tag localhost/netbackup/postgresql:<version>
${REGISTRY}/netbackup/postgresql:<version>

$ docker tag localhost/netbackup/postgresql-upgrade:<version>
${REGISTRY}/netbackup/postgresql-upgrade:<version>

$ docker tag localhost/netbackup/log-viewer:<version>$
{REGISTRY}/netbackup/log-viewer:<version>

$ docker tag localhost/netbackup/dbm:<version>
${REGISTRY}/netbackup/dbm:<version>
```

4 (*EKS-specific*) Login using the following command:

```
docker login -u AWS -p $(aws ecr get-login-password --region
<region-name>) <account-id>.dkr.ecr.<region-name>.amazonaws.com
```

If the repository is not created, then create the repository using the following command:

```
aws ecr create-repository --repository-name <image-name> --region
<region-name>
```

For example, `aws ecr create-repository --repository-name veritas/flexsnap-datamover --region us-east-2`

5 Run the following commands to push the images to the container registry:

```
$ docker push ${REGISTRY}/netbackup/fluentbit:<version>

$ docker push
${REGISTRY}/netbackup/fluentbit-log-cleanup:<version>

$ docker push ${REGISTRY}/netbackup/main:<version>

$ docker push ${REGISTRY}/netbackup/media:<version>

$ docker push ${REGISTRY}/netbackup/mqbroker:<version>

$ docker push ${REGISTRY}/netbackup/mqbroker-init:<version>

$ docker push ${REGISTRY}/netbackup/nbatd-init:<version>

$ docker push ${REGISTRY}/netbackup/nbatd-main:<version>

$ docker push ${REGISTRY}/netbackup/nbhousekeeping:<version>

$ docker push ${REGISTRY}/netbackup/operator:<version>

$ docker push ${REGISTRY}/netbackup/pbx:<version>

$ docker push ${REGISTRY}/netbackup/policyjobmgr:<version>

$ docker push ${REGISTRY}/netbackup/postgresql:<version>

$ docker push ${REGISTRY}/netbackup/postgresql-upgrade:<version>

$ docker push {REGISTRY}/netbackup/requestrouter:<version>

$ docker push ${REGISTRY}/netbackup/vnetd:<version>

$ docker push ${REGISTRY}/netbackup/ws-app:<version>

$ docker push ${REGISTRY}/netbackup/ws-init:<version>

$ docker push ${REGISTRY}/netbackup/log-viewer:<version>

$ docker push ${REGISTRY}/netbackup/main:<version>
```

Installing the docker images for Snapshot Manager

The Snapshot Manager package `netbackup-flexsnap-$(SNAPSHOT_MANAGER_VERSION).tar.gz` for Kubernetes includes the following:

- A docker image for Snapshot Manager operator
- 7 docker images for Snapshot Manager: flexsnap-fluentd, flexsnap-deploy, flexsnap-core, flexsnap-rabbitmq, flexsnap-nginx, flexsnap-postgresql, flexsnap-datamover

To install the docker images

1 Download `netbackup-flexsnap-$(SNAPSHOT_MANAGER_VERSION).tar.gz` from the Cohesity site.

2 Load the docker images to your docker storage.

```
docker load -i
netbackup-flexsnap-$(SNAPSHOT_MANAGER_VERSION).tar.gz
```

3 Tag the images.

```
$ docker tag
localhost/veritas/flexsnap-fluentd:${SNAPSHOT_MANAGER_VERSION}
${REGISTRY}/veritas/flexsnap-fluentd:${SNAPSHOT_MANAGER_VERSION}
```

```
$ docker tag
localhost/veritas/flexsnap-datamover:${SNAPSHOT_MANAGER_VERSION}
${REGISTRY}/veritas/flexsnap-datamover:${SNAPSHOT_MANAGER_VERSION}
```

```
$ docker tag
localhost/veritas/flexsnap-nginx:${SNAPSHOT_MANAGER_VERSION}
${REGISTRY}/veritas/flexsnap-nginx:${SNAPSHOT_MANAGER_VERSION}
```

```
$ docker tag
localhost/veritas/flexsnap-postgresql:${SNAPSHOT_MANAGER_VERSION}
${REGISTRY}/veritas/flexsnap-postgresql:${SNAPSHOT_MANAGER_VERSION}
```

```
$ docker tag
localhost/veritas/flexsnap-core:${SNAPSHOT_MANAGER_VERSION}
${REGISTRY}/veritas/flexsnap-core:${SNAPSHOT_MANAGER_VERSION}
```

```
$ docker tag
localhost/veritas/flexsnap-deploy:${SNAPSHOT_MANAGER_VERSION}
${REGISTRY}/veritas/flexsnap-deploy:${SNAPSHOT_MANAGER_VERSION}
```

```
$ docker tag
localhost/veritas/flexsnap-rabbitmq:${SNAPSHOT_MANAGER_VERSION}
${REGISTRY}/veritas/flexsnap-rabbitmq:${SNAPSHOT_MANAGER_VERSION}
```

Note: Ensure that you use the same tag as that of Snapshot Manager image version.

4 Push the images.

```
$ docker push
${REGISTRY}/veritas/flexsnap-rabbitmq:${SNAPSHOT_MANAGER_VERSION}

$ docker push
${REGISTRY}/veritas/flexsnap-fluentd:${SNAPSHOT_MANAGER_VERSION}

$ docker push
${REGISTRY}/veritas/flexsnap-datamover:${SNAPSHOT_MANAGER_VERSION}

$ docker push
${REGISTRY}/veritas/flexsnap-nginx:${SNAPSHOT_MANAGER_VERSION}

$ docker push
${REGISTRY}/veritas/flexsnap-postgresql:${SNAPSHOT_MANAGER_VERSION}

$ docker push
${REGISTRY}/veritas/flexsnap-core:${SNAPSHOT_MANAGER_VERSION}

$ docker push
${REGISTRY}/veritas/flexsnap-deploy:${SNAPSHOT_MANAGER_VERSION}
```

Configure Snapshot Manager

After you push the docker images to the following respective registry, then initialize Snapshot Manager (flexsnap) operator and configure Snapshot Manager:

The Snapshot Manager operator starts with NetBackup operator. For more information, refer to the following section:

See "Deploying the operators" on page 136.

Installing the docker images and binaries for MSDP Scaleout

The MSDP package `VRTSpddek.tar.gz` for Kubernetes includes the following:

- A docker image for MSDP operator
- 3 docker images for MSDP Scaleout: uss-controller, uss-mds, and uss-engine
- A kubectl plugin: kubectl-msdp

Note: The kubectl plugin is required only when MSDP Scaleout is deployed separately without the environment operator or Helm charts.

For more information, See "Installing the docker images and binaries for MSDP Scaleout (without environment operators or Helm charts)" on page 425.

To install the docker images and binaries for AKS

1. Download `VRTSpddek.tar.gz` from the Cohesity site.
2. Load the docker images to your docker storage.

   ```
   tar -zxvf VRTSpddek.tar.gz
   ```

   ```
   ls VRTSpddek-*/images/*.tar.gz|xargs -i docker load -i {}
   ```

3. Push the docker images to the ACR. Keep the image name and version same as original.

   ```
   docker login <your-acr-url>
   for image in msdp-operator uss-mds uss-controller uss-engine; do \
       docker image tag $image:<version> <your-acr-url>/$image:<version>; \
       docker push <your-acr-url>/$image:<version>; \
   done
   ```

To install the docker images and binaries for EKS

1. Download `VRTSpddek.tar.gz` from the Cohesity site.
2. Load the docker images to your docker storage.

   ```
   tar -zxvf VRTSpddek.tar.gz
   ```

   ```
   ls VRTSpddek-*/images/*.tar.gz|xargs -i docker load -i {}
   ```

3. Push the docker images to the ECR.
   - Log in.

     ```
     aws ecr get-login-password \
     --region <region> \
     | docker login \
     --username AWS \
     --password-stdin \
     <aws_account_id>.dkr.ecr.<region>.amazonaws.com
     ```

   - Create a repository.
     Refer to the "Creating a private repository" section of the *AWS documentation*.
   - Push the docker images to ECR. Keep the image name and version same as original.

     ```
     for image in msdp-operator uss-mds uss-controller
     uss-engine; do \
     docker image tag $image:<version> <your-ecr-url>/$image:<version>; \
     ```

```
docker push <your-ecr-url>/$image:<version>; \
done
```

Configuring NetBackup

Primary and media server CR

This section provides details of the primary and media server CR's.

For more information on managing the load balancer service, See “About the Load Balancer service” on page 205.

About primary server CR and media server CR

Primary server custom resource is used to deploy the NetBackup primary server and media server custom resource is used to deploy the NetBackup media server.

- After the operator is installed, update the custom resource YAMLs to deploy the primary server and media server CRs located in the `samples` folder.
- The primary server CRD and media server CRD are installed using the operators helm chart.
- **Name** used in the primary server and media server CRs must not be same. In the primary server CR the **Name** should not contain the word **media** and in the media server CR the **Name** should not contain the word **primary**.

  Note: After deployment, you cannot change the **Name** in primary server and media server CR.

- Before the CRs can be deployed, the utility called `Config-Checker` is executed that performs checks on the environment to ensure that it meets the basic deployment requirements. The config-check is done according to the **configCheckMode** and **paused** values provided in the custom resource YAML. See the section called “How does the Config-Checker utility work” on page 44.
- You can deploy the primary server and media server CRs in same namespace.
- (*AKS-specific*) Use the storage class that has the storage type as `Azure premium files` for the catalog volumes in the primary server CR, and the storage type as `Azure disk` for the data and log volumes in the media server CR and primary server CR.

(*EKS-specific*) Use the storage class that has the storage type as `Amazon elastic files` for the catalog volume in the primary server CR. For data and log volumes in the media server use the storage type as `EBS`.

- During fresh installation of the NetBackup servers, the value for **keep logs up to** under **log retention configuration** is set based on the log storage capacity provided in the primary server CR inputs. You may change this value if required.
  To update logs retention configuration, refer the steps mentioned in NetBackup™ Logging Reference Guide.
  - The NetBackup deployment sets the value as per the formula.
    Size of logs PVC/PV * 0.8 = Keep logs up value By default, the default value is set to 24GB.
    For example: If the user configures the storage size in the CR as 40GB (instead of the default 30GB) then the default value for that option become 32GB automatically based on the formula.

    Note: This value will get automatically updated to the value of `bp.conf` file on volume expansion.

- Deployment details of primary server and media server can be observed from the operator pod logs using the following command:
  ```
  kubectl logs <operator-pod-name> -c netbackup-operator -n
  <operator-namespace>
  ```

After installing primary server CR

Note the following points:

- (*AKS-specific*) The primary server CR will create a pod, a statefulset, a load balancer service, a configmap, a persistent volume claim for log volume, and a persistent volume claim for catalog volume.
  (*EKS-specific*) The primary server CR will create a pod, a statefulset, a load balancer service, a configmap and PVCs.
- Primary server can be considered as successfully installed and running when log-viewer, nbatd, nbmqbroker, nbwsapp, bpdbm, policyjob, policyjobmgr, primary, requestrouter pods successfully got into running state.
- You can access the NetBackup Web UI using the **primary server hostname** that was specified in the primary server CR status in **Primary server details** section.
  For example, if the primary server hostname is **nbu-primary**, then you can access the Web UI at *https://nbu-primary/webui/login*.

- Check the installation progress in the pod logs by using the following command:

  ```
  kubectl logs <request-router-pod-name> -n <namespace>
  ```

  Request router can be considered as successfully installed and running when the request router pod's state is **ready (1/1)**.

After Installing the media server CR

Note the following points:

- The media server CR will create a pod, a configmap, a loadbalancer service, a persistent volume claim for data volume, and a persistent volume claim for log volume.
- Initially pod will be in not ready state (0/2) when installation is going in the background. Check the pod logs for installation progress using the following command:

  ```
  kubectl logs <media-pod-name> -n <namespace>
  ```

  Media server can be considered as successfully installed and running when the media server pod's state is **ready (2/2)**.
- Details of media server name for each replica can be obtained from media server CR status by running the following command:

  ```
  kubectl describe <MediaServer_cr_name> -n <namespace>
  ```

Elastic media server

All the replicas for the media server are always up and running which incurs unnecessary cost to customers. The basic media server pod power management (Elastic media server) feature provides Auto scaling of media server replicas based on the CPU and memory usage as well as the jobs queued due to maximum jobs per media server settings to reduce the cost.

Note: For some of the cases such as import and duplication and so on, specifically selected elastic media server is ignored and is treated as any available media server. This is not applicable to cases where media server is used as backup host/client.

Enabling/disabling the auto scaling feature

For enabling/disabling the auto scaling feature, following media server CR inputs are required:

- **replicas**: Describes the maximum number of replicas that the media server can scale up to.

- **minimumReplicas**: Describes the minimum number of replicas of the media server running. This is an optional field. If not specified, the value for **minimumReplicas** field will be set to the default value of 1.
- From version 10.5 and later, along with CPU and memory usage, the media server scaleout is also seen if the jobs are found in queued state due to the maximum job per media server settings. To configure this setting, refer to the configuration parameter `bpsetconfig` below.

To enable the elasticity of media server, the value of **replicas** must be more than value of **minimumReplicas**.

To disable the autoscaling feature of media server, ensure that the value of **replicas** is equal to the value of **minimumReplicas**.

Note: The value of **replica** must be greater than 0 to enable the elasticity of media server.

NetBackup 11.0 and later, now provides support for scaling down the **minimumReplica** value of media server custom resource to 0 to diverge from the default behavior. After updating the value of **minimumReplica** there would be no media server pod running when there are no jobs running. This improves the total cost of ownership (TCO). The count of nodes reduces to 2 when the setup is idle. User must change the value of **minimumReplica** by editing the environment custom resource object.

- When **minimumReplica = 0**, user must change the minimum size of media nodepool to 0 through the portal.
- If no existing media pod or external media is available, all jobs that require storage interaction will trigger the creation of a new media pod by the NetBackup operator. A job remains in queue/active state waiting for resource with the following reason till the time new media pod is up and ready:

  ```
  Cloud scale media server is not available
  ```

  Primary server acting as media server will not be used in such cases.
  For more information on the above reason and the resolution for the same, refer to the following section:
  See "Job remains in queue for long time" on page 365.

Note: For certain jobs, example big data workloads, a specific media server is required. Users must configure these jobs with minimumReplica = 1 in the media server custom resource. Same applies to other cases where media server used as backup host.

Status attributes of elastic media server CR

Following table describes the ElasticityAttributes that describes the attributes associated with the media server autoscaler. These attributes are only applicable if autoscaler is running.

Fields	Description
ExpectedReplicas	Indicates the ideal number of replicas computed by media server autoscaler that must be running. **Note:** If autoscaler is disabled then **ExpectedReplicas** is equals to **minimumReplicas**.
ActiveReplicas	Indicates the actual number of replicas that must be running to complete the ongoing operations on the media servers. **Note:** If autoscaler is disabled then **ActiveReplicas** is equals to **minimumReplicas**.
NextIterationTime	Indicates the next iteration time of the media server autoscaler that is, the media server autoscaler will run after NextIterationTime only. Default value is empty.

Configuration parameters

- **ConfigMap**
  A new ConfigMap with name `nbu-media-autoscaler-configmap` is created during deployment and the key-value pairs would be consumed for tuning the media server autoscaler. This ConfigMap is common to all the media server CR objects and supports the following keys:

Parameters	Description
memory-low-watermark-in-percent	Low watermark for memory usage.
memory-high-watermark-in-percent	High watermark for memory usage.
cpu-low-watermark-in-percent	Low watermark for CPU usage.
cpu-high-watermark-in-percent	High watermark for CPU usage.
scaling-interval-in-seconds	Interval after which media server autoscaler should run.
stabilitywindow-time-in-seconds	CPU and memory usage is calculated between two time intervals. This key indicates the time interval to be considered for collecting usage.

Parameters	Description
stability-count	CPU and memory usages are calculated by averaging out on multiple readings. This key indicates the number of readings to be considered.
graceful-shutdown-interval-in-seconds	The time interval after which the media server autoscaler should run incase it is not able to scale in due to running jobs on media server pods.
delayed-scalein-notifications-interval-in-minutes	The time interval between two successive notifications in the event that a scale in does not occur.

Note: If you are upgrading to latest version, change the default values of the following parameters: **scaling-interval-in-seconds : “45” stabilitywindow-time-in-seconds : “5” stability-count : “3” graceful-shutdown-interval-in-seconds : “35” cpu-high-watermark-in-percent: "80"**

- **bpsetconfig**
  A new entry has been added in the primary server `bp.conf` that is consumed by media server autoscaler. This value applies to all the Cloud Scale Technology managed media servers.

Parameters	Description
MAX_JOBS_PER_K8SCLUSTER_MEDIA_SERVER	Maximum number of jobs that can run on each media server. This value can be set using bpsetconfig CLI. NetBackup Commands Reference Guide

- **Media server scaling**

Parameters	Description
Scale-out	If all the active media servers managed by the Cloud Scale Technology are at their capacity due to the maximum jobs per media server settings and if there are more jobs in queue, scale-out is performed and multiple replicas may get scaled out due to the media server settings. Additionally, if there are no jobs in queue due to this settings and if the CPU or memory consumption in the specified value provided in configMap but if any existing media server is idle that is, no jobs are running on it, then scale-out will not be performed. If all the existing media servers which are ready have jobs running on them, media server autoscaler will scale out a media server pod.
Scale-in	If the CPU and memory consumption is below the specified values provided in configMap, media server autoscaler will scale in the media server pods. Ensure that the running jobs are completed. **Note:** The scale-in does not happen until there are jobs in the queue due to the maximum job per media server settings.

Note:

The media server autoscaler scales out a single pod at a time in case a scale-out happens due to CPU and memory usage. It may exit from the multiple pods in case the scale-out happens due to the throttled jobs. The media server autoscaler can scale-in multiple pods at a time.

Note: If the scale-in does not happen due to background processes running on the media server, a notification would be sent on NetBackup Web UI after regular time interval as configured in the autoscaler ConfigMap. For more details, see the following section:

See "Troubleshooting AKS and EKS issues" on page 317.

The time taken for media server scale depends on the value of **scaling-interval-in-seconds** configuration parameter. During this interval, the jobs would be served by existing media server replicas based on NetBackup throttling parameters. For example, Maximum concurrent jobs in storage unit, Number of jobs per client, and so on.

Cluster's native autoscaler takes some time as per **scale-down-unneeded-time** attribute, which decides on the time a node should be unneeded before it is eligible to be scaled down. By default this is 10 minutes. To change this parameter, edit the cluster-autoscaler's current deployment settings using the following commands and then edit the existing value:

- AKS: `az aks update --resource-group $RESOURCE_GROUP_NAME --name $CLUSTER_NAME --cluster-autoscaler-profile scale-down-unneeded-time=5m`
- EKS: `kubectl -n kube-system edit deployment cluster-autoscaler`

Note the following:

- For scaled in media servers, certain resources and configurations are retained to avoid reconfiguration during subsequent scale out.
  - Kubernetes services, persistent volume claims and persistent volumes are not deleted for scaled in media servers.
- For scaled down media servers, the deleted media servers are also displayed on Web UI/API during the credential validation for database servers.

Handling of sudden incoming jobs

Based on the configured schedules, if a large number of jobs are expected to run at certain time, the maximum number of jobs per media server should be configured to ensure that the required number of media server pods are scaled out and the jobs are properly distributed.

For configuration related parameters, see 'Media server scaling' table in Configuration parameters

Elastic media servers and primary server certificate sharing

Starting from NetBackup version 11.0, elastic media servers have been enhanced to share the primary server certificate. This enhancement is a step towards creating a unified, logical media server entity in a Cloud Scale environment with the following changes:

- **Mapping**: Elastic media servers will be now mapped to the primary server's host ID.
- **Certificate renewal**: The primary server will now manage the certificate renewal for all media servers.
  The renewed certificate will now be shared across the elastic media servers.
- **Host ID management**: There will no longer be dedicated Host ID entries for elastic media servers, simplifying host ID management within the environment.

This design ensures consistency and streamlines the management of certificates and Host IDs, contributing to a more cohesive and scalable architecture for elastic media servers.

- **Post upgrade**: Upon upgrading to version 11.0.x.x.xxxx, the existing elastic media server certificates and their corresponding host IDs will be deleted.

  All media servers will then be mapped to the primary server's Host ID.

Chapter 5

Configuration of key parameters in Cloud Scale deployments

This chapter includes the following topics:

Tuning touch files

Some files are populated with default values, which are used to tune the performance parameters of NetBackup. Following table lists the files and a brief description of what each accomplishes:

Table 5-1 Files for tuning performance parameters of NetBackup

File location	Value set	Purpose
`/usr/openv/netbackup/db/config/DPS_PROXYDEFAULTRECVTMO`	800	To set the timeout value for the Data Protection Server (DPS) proxy. For more information, see *NetBackup Appliance Commands Reference Guide*.
`/usr/openv/netbackup/db/config/BPSCHED_THRESHOLD`	1000000	To control the frequency with which media servers send messages to the primary server to update job status in the activity monitor. For more information, see Knowledge base 100008521.
`/usr/openv/netbackup/db/config/RBALLOC_KBYTES_THRESHOLD`	25000000	To regulate the frequency with which media servers send disk storage capacity update (RBDB update) messages to the primary server. For more information, see Knowledge base 100008521.
`/usr/openv/netbackup/db/config/REPORT_RAW_KBS`	3600	To reduce traffic to the primary server. Not required when writing to MSDP server.
`/usr/openv/netbackup/db/config/DEFERRED_IMAGE_LIMIT`	64	To change the limit of number of backed up images. For more information, see *NetBackup Appliance Commands Reference Guide*.

Table 5-1 Files for tuning performance parameters of NetBackup *(continued)*

File location	Value set	Purpose
`/usr/openv/netbackup/db/config/SIZE_DATA_BUFFERS`	262144	To set the buffer size that the media server uses to buffer data between the network and the tape drive. It must be a multiple of 1024. For more information, see Knowledge base 100000498.
`/usr/openv/netbackup/MAX_ENTRIES_PER_ADD`	100000	To configure the number of metadata entries to be sent to the primary's catalog in each batch. For more information, see *NetBackup Backup Planning and Performance Tuning Guide*.

Setting maximum jobs per client

The maximum number of jobs that can run concurrently per client is set to 5. This value can be changed.

For more information on changing the value, refer to the *NetBackup Backup Planning and Performance Tuning Guide*.

Setting maximum jobs per media server

Cloud Scale Technology manages maximum number of jobs that runs on the media server. The default value of the job is 10000. It is recommended to change this value as per the backup environment. The setting can be added to the primary server's `bp.conf` file. To change the default value, the following parameter can be added or updated in the primary server `bp.conf` file using bpsetconfig CLI command as `MAX_JOBS_PER_K8SCLUSTER_MEDIA_SERVER = <value>`.

Refer to the NetBackup Commands Reference Guide.

To tune the value refer to the benchmarking figures from the section

---

Note: The value is not be applicable to any external media servers that are added in the Cloud Scale Technology environment. Also, the configuration is only applicable for backup and duplication jobs. Other types of jobs do not consider this limit and runs on the available media servers even if those are at capacity due to this limit.

Additionally, third-party monitoring tools like Azure insights, Prometheus, etc. can be used to monitor the usage of media server pods. If those pods are underutilized which means that the media server is capable to run number of jobs, thus the value for this setting can be increased. Similarly, if the pods are overutilized, the value for this setting can be decreased.

Enabling intelligent catalog archiving

Intelligent catalog archiving (ICA) is used to reduce the number of catalog files based on a specified retention period or file size. When you enable ICA, any catalog file that is older than the specified retention period value is removed from the catalog disk. The retention value is set to 2592000 (that is., 30 days). The advantage of ICA is that it shortens catalog backup time, by reducing the number of catalog files to be backed up.

To change the retention period, refer to the 'Enabling intelligent catalog archiving' section of the *NetBackup Administrator's Guide, Volume I*.

Enabling security settings

Changes to the security settings include the following:

- Disabling earlier versions of NetBackup 8.0 support for insecure connections.
  For more information on enabling this, refer to the 'How NetBackup 8.1 hosts communicate with NetBackup 8.0 and earlier hosts' section of *NetBackup Read This First Guide for Secure Communications*.
- Setting the data-in-transit encryption to **Preferred On**, which can be changed as listed in 'Configure the global data-in-transit encryption setting' section of *NetBackup Security and Encryption Guide*.

Configuring email server

Mail server can be configured during NetBackup installation by including the name of the **configMap** in the **emailServerConfigmapName** field of the `environment.Spec` in `helm/cloudscale/cloudscale-values.yaml` file.

- The **configMap** data must have entries in a `key: value` form to configure the mail server, as shown below for smtp field:

  ```
  emailServerConfigmap
  apiVersion: v1
  kind: ConfigMap
  metadata:
  name: configmail
  namespace: <netbackup-namespace>
  data:
    smtp: "xyz.abc.domain.com"
    smtp-use-starttls: ""
  ```

- If there is a specific parameter that needs to be set only (not value), a key can only be specified with the **smtp-use-starttls** field.

Perform the following to modify the mail server settings:

- Exec into the primary container using the following command:

  ```
  kubectl exec -it -n <namespace> <primaryServer-pod-name> -- bash
  ```

- Update the persistent file at `/mnt/nbdata/usr/openv/etc/mail.rc` to set the fields as required.

If the **emailServerConfigmapName** field in the `environment.Spec`, is left empty (""), then the mail server would not be configured automatically as a part of NetBackup installation. In such case, the user will have to configure it manually post installation by performing the following steps:

- Exec into the primary container using the following command:

  ```
  kubectl exec -it -n <namespace> <primaryServer-pod-name> -- bash
  ```

- Execute the following commands in the primary container:

  ```
  mv /etc/mail.rc /mnt/nbdata/usr/openv/etc/mail.rc
  ln -nsf /mnt/nbdata/usr/openv/etc/mail.rc /etc/mail.rc
  ```

- Update values for required fields at `/mnt/nbdata/usr/openv/etc/mail.rc`. An example of **mail.rc** file is shown below:

  ```
  mail.rc
  # mail server configuration
  set mail
  set mailserver=xyz.abc.domain.com:25
  set smtp-use-starttls
  ```

Reducing catalog storage management

(*For EKS-specific deployments*) The primary catalog storage is based on the Elastic File System, which provides unlimited storage (automatic expansion), thereby effacing the need to monitor it.

(*For AKS-specific deployments*) With this release, the fresh installation of Cloud Scale deployment would have the capability to self-regulate the primary server's catalog volume. The functionality can be controlled by changing the value of **allowVolumeExpansion** field in the primary server catalog storage section of `cloudscale-values.yaml` file.

If the **allowVolumeExpansion** field is set to,

- *true*, the operator automatically expands the catalog's capacity by 20%, every time the catalog reaches 80% of its usage. This check for catalog usage is only triggered every 10 hours.
- *false*, the automatic check and expansion of primary catalog usage will not happen, and if needed, the volume can be expanded. See "Expanding storage volumes" on page 164.

If required, the user can expand the catalog capacity by a value different than 20%. This can be achieved by performing the following:

- Set the value of **allowVolumeExpansion** field to *false* under primary's catalog storage.

  ```
  helm upgrade cloudscale cloudscale-11.2.tgz -n netbackup
  --reuse-values --set
  environment.primary.storage.catalog.autoVolumeExpansion=false
  ```

- Manually expand the catalog capacity to desired value using the following command:

  ```
  helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup
  --reuse-values --set
  environment.primary.storage.catalog.capacity=80Gi (or whatever the
  desired capacity is)
  ```

  See "Expanding storage volumes" on page 164.

- To enable automatic catalog volume usage monitoring, update the value of **allowVolumeExpansion** field in the environment to *true* after manually updating the capacity.

Configuring zone redundancy

This section describes configuring zone redundancy using the zone redundant storage for disk-based and file-based storage.

Disk-based storage

Azure-disk based storage

- Zone-redundant storage (ZRS) synchronously replicates an Azure managed disk across three Azure availability zones in the regions selected. This can be selected by setting **skuname: Premium_ZRS** in the `yaml` file for creating the storage class.
- ZRS disks are currently available in: Southeast Asia, Australia East, Brazil South, North Europe, West Europe, France Central, Japan East, Korea Central, Qatar Central, UK South, East US, East US 2, South Central US and West US 2.
- The following `yaml` file can be used:

```
aks-disk
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: <storage class name>
provisioner: disk.csi.azure.com
reclaimPolicy: Retain
allowVolumeExpansion: True
volumeBindingMode: Immediate
parameters:
  skuname: Premium_ZRS
```

EKS disk-based storage (EBS)

- Zone redundancy not supported. The snapshot backup to store backup at different S3 bucket can be used.
- Refer to the following document to take volume snapshot and restore it to the existing pod:
  Using Amazon EBS snapshots for persistent storage with your Amazon EKS cluster by leveraging add-ons

File-based storage

Azure file-based storage

- Zone-redundant storage (ZRS) replicates the storage account synchronously across three Azure availability zones in the primary region. This can be selected by setting **skuname: Premium_ZRS** in the `yaml` file for creating the storage class.

- ZRS for premium file shares is available in: Southeast Asia, Australia East, Brazil South, North Europe, West Europe, France Central, Japan East, Korea Central, Qatar Central, UK South, East US, East US 2, South Central US and West US 2.
- The following `yaml` file can be used:

```
aks-file
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: <name of storage class>
provisioner: file.csi.azure.com
reclaimPolicy: Retain
allowVolumeExpansion: True
volumeBindingMode: Immediate
parameters:
  skuName: Premium_ZRS
  protocol: nfs
```

EKS file-based storage (EFS)

- Zone redundancy is automatically supported within the region.
  For more information, see Resilience in Amazon Elastic File System.

Enabling client-side deduplication capabilities

Client-side deduplication or Client Direct is an easy way to improve the performance of your backups to an MSDP target. Backup performance can be improved by using the OpenStorage storage server instead of the media server to transfer the data during backups. For more information on Client Direct, see *NetBackup Deduplication Guide*.

This can be done by adding the **OST_CLIENT_DIRECT** new entry in the `/usr/openv/netbackup/bp.conf` file of the media server. The **OST_CLIENT_DIRECT** entry can have one of the following values:

- 0 - Client Direct will not be used as the data transfer method to transfer the data to the specified host.
- 1 - Client Direct will be preferred as the data transfer method to transfer the data to the specified host, contingent on Client Direct capability probes on the storage

server permitting the same. This is the default value set for Cloud Scale deployment.

- 2 - Client Direct will always be used as the data transfer method to transfer the data to the specified host.

To change the value of **OST_CLIENT_DIRECT** entry through the bpclient CLI, refer to the *NetBackup Commands Reference Guide*.

To enable client-side deduplication during restores, add **OST_CLIENT_DIRECT_RESTORE [=0/1/2]** entry to the `bp.conf` file of the media server(s). A new entry **OLD_VNETD_CALLBACK=YES** must be added to the `bp.conf` file of all the clients that would use the client-side dedupliction feature during restores, barring which restore jobs would fail. For more information, see *NetBackup Administrator's Guide, Volume I*.

The restore mode (for enabling/disabling Client Direct) can also be changed using the bpclient CLI as listed under **client_direct_restore** of the *NetBackup Commands Reference Guide*.

If Client Direct is enabled/disabled using the `bp.conf` file and through the bpclient CLI, then the configuration in bpclient would take precedence over the `bp.conf` file changes.

Parameters for logging (fluentbit)

The logging feature is introduced in NetBackup 10.5 to to consolidate log files that have been distributed during the process of running NetBackup in a scale-out environment. To configure the fluentbit, ensure that the following prerequisites are met:

Prerequisites for logging fluentbit)

Ensure that the following system requirements and prerequisites are met before proceeding with the deployment:

- **System requirements:** Cloud Scale deployment environment only
- **Application requirements:** Cloud Scale fluentbit pods and containers:
  - Fluentbit collector pod and containers
  - Fluentbit DaemonSet sender pods and containers
  - Fluentbit sidecar sender containers

To view the current values, execute the command:

```
helm get values cloudscale -n <netbackup namespace> > new-values.yaml
```

Update the `new-values.yaml` file to modify the required parameters under the fluentbit section based on your environment and configuration requirements.

To apply the new values of fluentbit, execute the command:

```
helm upgrade --install cloudscale cloudscale-11.2.tgz -f
new-values.yaml -n <netbackup namespace>
```

Table 5-2 Fluentbit collector configuration variables

Configuration value	Description
fluentbit.volume.pvcStorage	This is the size of the PVC created for the fluentbit collector to store logs.
collectorNodeSelector	This is how you can set the node selector for the fluentbit collector pod.

Table 5-3 Log Cleanup configuration variables

Configuration value	Description
fluentbit.volume.pvcStorage	This parameter describes the size of the PVC created for the fluentbit collector to store logs.
fluentbit.cleanup.retentionDays	(number of days) - Number of days to retain logs before cleaning them up. For example1 would mean 1 day which would mean that if the logs were created the day before they are to be kept that 1 day and no cleanup would occur. The next day the logs are 2 days old and with a setting of 1 they would be cleaned up. **Default:** retentionDays: 7
fluentbit.cleanup.retentionCleanupTime	(hh:mm) - Time of day to cleanup the logs. This is based on the local Kubernetes time zone. The retention cleanup occurs once daily. **Default:** retentionCleanupTime: 04:00
fluentbit.cleanup.utilizationCleanupFrequency	(number of minutes) - Number of minutes to wait between subsequent utilization cleanups. This is based on start time of the previous cleanup, not when the cleanup finishes. For example: `helm upgrade cloudscale cloudscale.tgz -n netbackup --reuse-values --set fluentbit.cleanup.utilizationCleanupFrequency=50` **Default:** utilizationCleanupFrequency: 60

Table 5-3 Log Cleanup configuration variables *(continued)*

Configuration value	Description
fluentbit.cleanup.highWatermark	(number percent) - This is the percentage of storage utilization that the utilization cleanup will start cleaning up data one day at a time. It will continue until it reaches the low watermark or only the current days logs remain. If only current days logs remain it will start cleaning up individual files of the current day based on last updated time. For example: `helm upgrade cloudscale cloudscale.tgz -n netbackup --reuse-values --set fluentbit.cleanup.highWatermark=80` **Default:** highWatermark: 90
fluentbit.cleanup.lowWatermark	(number percent) - This is the percentage of storage utilization that the utilization cleanup will stop cleaning up data once it gets below. **Default:** lowWatermark: 60
fluentbit.namespaces	This is a list of namespaces to collect stdout logs from. This is designed to allow you to filter out cluster logs and other logs not related to the NetBackup system in order to reduce network traffic and focus on relevant logs. **Default**: netbackup, netbackup-operator-system

Table 5-4 Fluentbit DaemonSet configuration variables

Configuration value	Description
tolerations	This is where you can set the tolerations of the daemonset pods to help determine which nodes they should be scheduled to.

Table 5-5 Resizing Log Volumes

Configuration value	Description

Resize Log Volumes:
 	
It is important that all the non-collector log volumes must remain the same size. This is because they are all using a common variable in the `bp.conf` file to determine when to be cleaned up and how much size to use.
	1 Except for the primary server, perform this action on every log volume. It opens the edit menu that allows you to change the volume size and save. Upon saving it takes few minutes to resize the volume. In order to resize the log volumes you must run the command:
	
	`$ kubectl edit pvc -n <netbackup namespace> <pvc name>`
	
	2 To resize the primary volume run:
	
	`helm upgrade cloudscale cloudscale.tgz -n netbackup --reuse-values --set environment.primary.storage.log.capacity=<expanded capacity>Gi`
	
	The new values take a few minutes to apply but it will change the log volume size as well as edit the associated `bp.conf` value.
	
	3 After executing the edit command, run the delete command to restart the pod.
	
	`kubectl delete pod -n <netbackup namespace> <application pod name>`

Table 5-6 Enabling the fluentbit sidecars logging flag

Configuration value	Description

fluentbit.enableSidecarsLogging	Enables or disables sidecar log collectors used to gather NetBackup and VxUL logs in addition to standard `stdout` logs.
	Default value is *false*. Only `stdout` logs are collected unless the flag is manually enabled.
	Use the following command to enable sidecar logging:
	`helm upgrade --install cloudscale cloudscale-11.2-4439002.tgz -n netbackup --reuse-values --set fluentbit.enableSidecarsLogging=true`
	The flag may be toggled from *false* to *true* or from *true* to *false* after deployment/upgrade completes.
	Use the following command to get the current cloud scale values:
	`helm get values cloudscale -n netbackup`
	Use the following command to verify the `configMap` update:
	`kubectl get configmap nbfluentbitsidecarconf -n netbackup -o yaml`

Managing media server configurations in Web UI

When media server certificates are shared with the primary server, then:

- Media servers would no longer have explicit Host IDs.
- Media servers would not appear in **Web UI > Hosts > Host properties** but would be visible in Java UI.

Editing media server properties

Perform the following steps to edit the properties of media server through Web UI:

1. Navigate to **Web UI > Hosts > Host properties > Configure hosts > Configure media server**.

2. Enter the **Media server name** and click on **Configure**.
3. Perform the required changes to the media server's configuration and click **Save** to apply the changes.

Note: Even after modifying the media server configurations, the media servers would not be visible in **Web UI > Hosts > Host properties**. However, the changes would have been applied successfully.

Using NVMe Local Storage for MSDP Cloud Cache

MSDP-C supports using local NVMe storage on Kubernetes worker nodes as the cloud cache for MSDP Engine Pods. Configuring cloud cache on NVMe disks can significantly improve performance and reduce cloud storage costs for cloud-backed MSDP deployments.

This section describes how to configure local NVMe storage on Kubernetes nodes for use as the cloud cache in an MSDP-C deployment. The configuration is performed by specifying an annotation on the MSDP Scaleout custom resource (CR) and updating the cloud cache configuration to use the NVMe-backed mount points.

Configurations

To improve MSDP-C performance and reduce costs, configure the following settings to use the NVMe local storage of Kubernetes nodes as the cloud cache:

1. Configure NVMe devices for MSDP engine pods:

   Local NVMe devices are configured through an annotation on the MSDPScaleout CR. The annotation specifies one or more block devices that are initialized and mounted for exclusive use by MSDP Engine Pods.

   - Create the MSDP node pool with a VM type having NVMe local storage.
     Example: For Azure cloud, use VM sizes from the L-series, such as **Standard_L8s_v3**, which provide NVMe local storage.
   - Specify the following annotation key-value pair for MSDP Scaleout in the Environment CR:
     `msdp.veritas.com/blkdev-params: <dev-params>`
     Where **`<dev-params>`** is specified in the following format:
     `<dev1>:<sub-mnt1>:<fstype1>;<dev2>:<sub-mnt2>:<fstype2>;...`
     Example 1: Single NVMe device: `msdp.veritas.com/blkdev-params: /dev/nvme0n1:nvme-mnt:xfs`

Example 2: Multiple NVMe devices: `msdp.veritas.com/blkdev-params: /dev/nvme0n1:nvme-mnt:xfs;/dev/nvme1n1:nvme-mnt2:xfs`

2. Initialize and mount NVMe device:

   The annotation key-value pair is used to specify the additional block devices exclusively used by the MSDP Engine Pods.

   During MSDP engine pod initialization:

   - Each device is formatted with the specified filesystem.
   - Each device is mounted in the engine container under `/bdev-mnts/` folder. For example, for `/dev/nvme0n1:nvme-mnt:xfs` annotation, the `/dev/nvme0n1` device is formatted as **xfs** and mounted at `/bdev-mnts/nvme-mnt`.

   Note: If the annotation is added after MSDP is already configured, the MSDP engine pods must be manually deleted. The NVMe devices are mounted only when the engine pods are restarted.

   ---

3. Configure Cloud cache:

   After the NVMe devices are mounted, update the cloud cache configuration so that cloud volumes use the NVMe-backed paths.

   Perform the following steps on each MSDP engine pod:

   - Update the cloud cache path by editing the cloud configuration file:
     `/msdp/data/dp1/pdvol/etc/puredisk/cloud.json`
     Change the value of cache path to a subdirectory under the NVMe mount point.
     For example, update the default `/msdp/data/dp1/pdvol/data/ds_3` cache path to NVMe based cache path as `/bdev-mnts/nvme-mnt/ds_3`.

     Note: You may also adjust the values of **DownloadDataCacheGB** and **DownloadMetaCacheGB** based on the available NVMe disk capacity.

   - Stop MSDP services: Disable health checks and stop MSDP services as follows:

     ```
     /usr/openv/pdde/health_check.sh disable
     /usr/openv/pdde/pdconfigure/pdde stop
     ```

   - Delete the existing cloud cache data by using the following command:

     ```
     rm -rf /msdp/data/dp1/pdvol/data/ds_3
     ```

- Restart the MSDP services by using the following command:

  `/usr/openv/pdde/health_check.sh enable`

  Wait for couple of minutes for the MSDP services to restart automatically and for the MSDP engine pods to return to the **Ready** state.

  To speed up service startup, you can manually start MSDP services as follows using the default root user:

  `/usr/openv/pdde/startup.sh`

Section 2

Deployment

Chapter 6

Deploying Cloud Scale

This chapter includes the following topics:

How to deploy Cloud Scale

Cloud Scale can be deployed using one of the following methods:

- Helm chart
- kubectl plugin

Table 6-1 describes a high level procedure for deploying Cloud Scale through Helm chart or kubectl plugin.

Table 6-1 Steps for deploying Cloud Scale (Helm chart or kubectl plugin)

Steps	Description
Type of deployment: Manual deployment using Cloud Scale Helm chart	
Step 1	Prerequisites for Cloud Scale deployment See "Prerequisites for Cloud Scale deployment" on page 124.

Table 6-1 Steps for deploying Cloud Scale (Helm chart or kubectl plugin) *(continued)*

Steps	Description
Step 2	Deploy the following operators: ■ msdp-operator ■ nb-operator ■ flexsnap operator See "Deploying the operators" on page 136.
Step 3	Deploy Cloud Scale environment using one of the following method: ■ Installing Cloud Scale environment using Helm charts ■ Single node Cloud Scale deployment See "Deploying Cloud Scale using Helm chart" on page 140.
Step 4	Verify if the Cloud Scale deployment is successful. See "Verifying Cloud Scale deployment" on page 152.
Type of deployment: Cloud Scale deployment using kubectl plugin	
Step 1	Prerequisites for Cloud Scale deployment See "Prerequisites for Cloud Scale deployment" on page 124.
Step 2	Deploying Cloud Scale using kubectl-cloudscale plugin See "Deploying Cloud Scale using kubectl plugin" on page 147.
Step 3	Verify if the Cloud Scale deployment is successful. See "Verifying Cloud Scale deployment" on page 152.

Prerequisites for Cloud Scale deployment

Ensure that the following steps are performed before deploying the operators:

1. Install cert-manager by using the following command:

   ```
   helm repo add jetstack https://charts.jetstack.io
   ```

   ```
   helm repo update jetstack
   ```

   ```
   helm upgrade -i -n cert-manager cert-manager jetstack/cert-manager
   \ --version 1.18.2 \ --set webhook.timeoutSeconds=30 \ --set
   installCRDs=true \ --wait --create-namespace
   ```

   For details, see cert-manager Documentation.

2. Create NetBackup namespace by using the following command:

   ```
   kubectl create ns netbackup
   ```

3. Install trust-manager by using the following command:

   ```
   kubectl create namespace trust-manager
   ```

   ```
   helm upgrade -i -n trust-manager trust-manager
   jetstack/trust-manager --set app.trust.namespace=netbackup
   --version v0.19.0 --wait
   ```

   For details, see trust-manager Documentation.

4 (If using private registry) Create kubernetes secret for using the private registries as follows:

Create operator namespace by using the following command:

```
kubectl create ns netbackup-operator-system
```

```
kubectl create secret docker-registry <secret-name> \ --namespace
<namespace> \ --docker-server=<container-registry-name> \
--docker-username=<service-principal-ID> \
--docker-password=<service-principal-password>
```

For example (AKS):

```
kubectl create secret docker-registry demo-secret
--namespace netbackup-operator-system
--docker-server=cpautomation.azurecr.io
--docker-username=c1d03169-6f35-4c29-b527-995bbad3b608
--docker-password=<password_here>
```

5 (For Cloud Scale) Create kubernetes secret for using the private registries as follows:

```
kubectl create secret docker-registry <secret-name> \ --namespace
<namespace> \ --docker-server=<container-registry-name> \
--docker-username=<service-principal-ID> \
--docker-password=<service-principal-password>
```

For example (AKS):

```
kubectl create secret docker-registry demo-secret
--namespace netbackup --docker-server=cpautomation.azurecr.io
--docker-username=c1d03169-6f35-4c29-b527-995bbad3b608
--docker-password=<password_here>
```

Air-Gapped installation procedure for cert-manager and trust-manager in AKS

This section provides step-by-step instructions for deploying **cert-manager** and **trust-manager** in an air-gapped or partially air-gapped Azure Kubernetes Service (AKS) environment. It enables installation without direct internet access, using JFrog Artifactory as the container registry (adaptable for ACR or ECR). The tools are installed in separate namespaces (**cert-manager** and **trust-manager**) for compatibility, with trust-manager configured to manage trust bundles in the NetBackup namespace.

Prerequisites

- Internet-connected and air-gapped environments
- Secure file transfer method between environments
- Access to a private container registry (for example, JFrog Artifactory)

- Docker/Podman and Helm installed on both environments
- `kubectl` configured for the AKS cluster
- Registry credentials

Air-Gapped installation procedure

1 Gather Helm charts locally (Internet-connected environment):

Download the cert-manager and trust-manager charts:

- ```
  # Add the Jetstack Helm repository
  helm repo add jetstack https://charts.jetstack.io
  --force-update

  # Update your local Helm repository indexes
  helm repo update

  # Pull the cert-manager chart
  helm pull jetstack/cert-manager --version v1.17.0 --untar
  --untardir ./cert-manager-chart

  # Pull the trust-manager chart
  helm pull jetstack/trust-manager --untar --untardir
  ./trust-manager-chart --replace
  ```

  This creates folders `./cert-manager-chart` and `./trust-manager-chart` containing the Helm chart files.

- Identify the container images in each chart:
  - Method 1: Generate and examine the manifests:

    ```
    helm template ./cert-manager-chart/cert-manager --namespace
     cert-manager > cert-manager-manifest.yaml
    helm template ./trust-manager-chart/trust-manager --namespace
     trust-manager > trust-manager-manifest.yaml

    # Find all images used in the manifests
    grep "image:" cert-manager-manifest.yaml
    grep "image:" trust-manager-manifest.yaml
    ```

    Example output:

    ```
    # cert-manager images
    image: "quay.io/jetstack/cert-manager-cainjector:v1.17.0"
    image: "quay.io/jetstack/cert-manager-controller:v1.17.0"
    image: "quay.io/jetstack/cert-manager-webhook:v1.17.0"
    ```
    ```

```
image:
"quay.io/jetstack/cert-manager-startupapicheck:v1.17.0"

# trust-manager images
image:
"quay.io/jetstack/trust-pkg-debian-bookworm:20230311.0"
image: "quay.io/jetstack/trust-manager:v0.16.0"
```

- Method 2: Examine the `values.yaml` files directly:
  For cert-manager: Look for `.image.repository`, `.image.tag`, and so on.
  For trust-manager: Look for `app.image.repository`, `app.image.tag`, and so on.
  The default images are typically pulled from `quay.io/jetstack/`.

2 Pull and save the images (Internet-connected environment):

```
# Pull cert-manager images
docker pull quay.io/jetstack/cert-manager-controller:v1.17.0
docker pull quay.io/jetstack/cert-manager-webhook:v1.17.0
docker pull quay.io/jetstack/cert-manager-cainjector:v1.17.0
docker pull quay.io/jetstack/cert-manager-startupapicheck:v1.17.0

# Pull trust-manager images
docker pull quay.io/jetstack/trust-manager:v0.16.0
docker pull quay.io/jetstack/trust-pkg-debian-bookworm:20230311.0

# Save all images into a single tarball
docker save \
  quay.io/jetstack/cert-manager-controller:v1.17.0 \
  quay.io/jetstack/cert-manager-webhook:v1.17.0 \
  quay.io/jetstack/cert-manager-cainjector:v1.17.0 \
  quay.io/jetstack/cert-manager-startupapicheck:v1.17.0 \
  quay.io/jetstack/trust-manager:v0.16.0 \
  quay.io/jetstack/trust-pkg-debian-bookworm:20230311.0 \
  -o cert-manager-trust-manager-images.tar
```

3 Copy the files to the offline/restricted host:

- Transfer the image tarball:
  Transfer `cert-manager-trust-manager-images.tar` file to your air-gapped environment using an approved method (for example, SCP, secure file transfer, or physical media).

- Transfer the Helm charts:
  Transfer the Helm charts to the air-gapped environment using one of the following approaches:
  - Option 1: Pack and transfer the extracted chart directories:

    ```
    # Create archive of the chart directories
    tar -czf helm-charts.tar.gz ./cert-manager-chart
    ./trust-manager-chart

    # Transfer helm-charts.tar.gz to the air-gapped environment
    # Then on the air-gapped host:
    tar -xzf helm-charts.tar.gz
    ```

  - Option 2: Create Helm package archives and transfer those:

    ```
    # Package the charts
    helm package ./cert-manager-chart/cert-manager -d ./packages
    helm package ./trust-manager-chart/trust-manager -d
    ./packages

    # Transfer the ./packages directory to the air-gapped
    environment
    ```

    Use the same secure file transfer method as used for the container images.

4 Load and push images into your private registry (Air-Gapped environment):

```
# Load the images into Docker
docker load -i cert-manager-trust-manager-images.tar

# Login to your private Artifactory registry
# You'll be prompted to enter your password securely
# If using ACR/ECR replace the docker login command below with
(Ex: az acr login -n myacr)
docker login my-artifactory.mycompany.com -u <your-username>

# Tag the images for your private registry (replace
my-artifactory.mycompany.com with your registry URL)
docker tag quay.io/jetstack/cert-manager-controller:v1.17.0 \

my-artifactory.mycompany.com/jetstack/cert-manager-controller:v1.17.0

docker tag quay.io/jetstack/cert-manager-webhook:v1.17.0 \

my-artifactory.mycompany.com/jetstack/cert-manager-webhook:v1.17.0

docker tag quay.io/jetstack/cert-manager-cainjector:v1.17.0 \

my-artifactory.mycompany.com/jetstack/cert-manager-cainjector:v1.17.0

docker tag quay.io/jetstack/cert-manager-startupapicheck:v1.17.0
 \

my-artifactory.mycompany.com/jetstack/cert-manager-startupapicheck:v1.17.0

docker tag quay.io/jetstack/trust-manager:v0.16.0 \
  my-artifactory.mycompany.com/jetstack/trust-manager:v0.16.0

docker tag quay.io/jetstack/trust-pkg-debian-bookworm:20230311.0
 \

my-artifactory.mycompany.com/jetstack/trust-pkg-debian-bookworm:20230311.0

# Push the images to your private registry
docker push
my-artifactory.mycompany.com/jetstack/cert-manager-controller:v1.17.0
docker push
my-artifactory.mycompany.com/jetstack/cert-manager-webhook:v1.17.0
```

```
docker push
my-artifactory.mycompany.com/jetstack/cert-manager-cainjector:v1.17.0
docker push
my-artifactory.mycompany.com/jetstack/cert-manager-startupapicheck:v1.17.0
docker push
my-artifactory.mycompany.com/jetstack/trust-manager:v0.16.0
docker push
my-artifactory.mycompany.com/jetstack/trust-pkg-debian-bookworm:20230311.0
```

5 Create image pull secret and update Helm charts:

- Create a kubernetes secret for registry authentication:

```
# Create the cert-manager namespace if it doesn't exist yet
kubectl create namespace cert-manager --dry-run=client -o yaml
 | kubectl apply -f -

# Create a secret with your registry credentials
# Create secret for cert-manager
kubectl create secret docker-registry
artifactory-registry-secret \
  --namespace cert-manager \
  --docker-server=my-artifactory.mycompany.com \
  --docker-username=<your-username> \
  --docker-password=<your-password> \
  --docker-email=<your-email>

# Create the same secret for trust-manager
kubectl create secret docker-registry
artifactory-registry-secret \
  --namespace trust-manager \
  --docker-server=my-artifactory.mycompany.com \
  --docker-username=<your-username> \
  --docker-password=<your-password> \
  --docker-email=<your-email>
```

- Update the Helm charts to use your private registry using one of the following options:
  - Option 1: Modify `values.yaml` files:
    For `cert-manager`, update the following sections in `cert-manager-chart/cert-manager/values.yaml` file:

```
image:
  repository:
my-artifactory.mycompany.com/jetstack/cert-manager-controller

  tag: v1.17.0
  # ...

webhook:
  image:
    repository:
my-artifactory.mycompany.com/jetstack/cert-manager-webhook
    tag: v1.17.0
  # ...

cainjector:
  image:
    repository:
my-artifactory.mycompany.com/jetstack/cert-manager-cainjector

    tag: v1.17.0
  # ...

  # Add imagePullSecrets
  imagePullSecrets:
  - name: artifactory-registry-secret
```

For trust-manager, update the following in trust-manager-chart/trust-manager/values.yaml:

```
app:
  image:
    repository:
my-artifactory.mycompany.com/jetstack/trust-manager
    tag: v0.16.0
  trust:
    namespace: netbackup
  # ...

pkgImporter:
  image:
    repository:
my-artifactory.mycompany.com/jetstack/trust-pkg-debian-bookworm

    tag: "20230311.0"
```

```
# Add imagePullSecrets
imagePullSecrets:
- name: artifactory-registry-secret
```

- Option 2: Use `--set` parameters during installation:
  Instead of modifying the `values.yaml` files directly, you can use `--set` parameters during installation.
  For more information, see the commented lines in the step 6 below.

6 Connect to and install on your AKS cluster (Air-Gapped environment):

- Authenticate to your AKS cluster:
  Before installing Helm charts, ensure that your `kubectl` is properly configured to communicate with your AKS cluster:

```
# If you have Azure CLI in your air-gapped environment
az login
az account set --subscription <your-subscription-id>
az aks get-credentials --resource-group <your-resource-group>
 --name <your-aks-cluster-name>

# OR if you've transferred a kubeconfig file
export KUBECONFIG=/path/to/your/kubeconfig

# Verify connectivity
kubectl cluster-info
kubectl get nodes
```

- Install `cert-manager`:
  Based on how you transferred the Helm charts, use one of the following approaches:

```
# Create cert-manager namespace kubectl create namespace
cert-manager
```

Option 1:

```
If you transferred the extracted chart directories
helm install cert-manager ./cert-manager-chart/cert-manager \

  --namespace cert-manager \
  --set crds.enabled=true
  # If you didn't modify values.yaml, add these overrides:
  # --set
image.repository=my-artifactory.mycompany.com/jetstack/cert-manager-controller
```

```
 \
  # --set image.tag=v1.17.0 \
  # --set
webhook.image.repository=my-artifactory.mycompany.com/jetstack/cert-manager-webhook
 \
  # --set webhook.image.tag=v1.17.0 \
  # --set
cainjector.image.repository=my-artifactory.mycompany.com/jetstack/cert-manager-cainjector
 \
  # --set cainjector.image.tag=v1.17.0 \
  # --set
startupapicheck.image.repository=my-artifactory.mycompany.com/jetstack/cert-manager-startupapicheck
 \
  # --set startupapicheck.image.tag=v1.17.0 \
  # --set imagePullSecrets[0].name=artifactory-registry-secret
```

Option 2:

```
If you transferred packaged charts (.tgz files)
helm install cert-manager ./packages/cert-manager-v1.17.0.tgz
 \
  --namespace cert-manager \
  --set crds.enabled=true
  # If you didn't modify values.yaml, add these overrides:
  # --set
image.repository=my-artifactory.mycompany.com/jetstack/cert-manager-controller
 \
  # --set image.tag=v1.17.0 \
  # --set
webhook.image.repository=my-artifactory.mycompany.com/jetstack/cert-manager-webhook
 \
  # --set webhook.image.tag=v1.17.0 \
  # --set
cainjector.image.repository=my-artifactory.mycompany.com/jetstack/cert-manager-cainjector
 \
  # --set cainjector.image.tag=v1.17.0 \
  # --set
startupapicheck.image.repository=my-artifactory.mycompany.com/jetstack/cert-manager-startupapicheck
 \
  # --set startupapicheck.image.tag=v1.17.0 \
  # --set imagePullSecrets[0].name=artifactory-registry-secret
```

- Install `trust-manager`:

Based on how you transferred the Helm charts, use one of the following approaches:

```
# Create trust-manager and netbackup namespace kubectl create
namespace trust-manager kubectl create namespace netbackup
```

Option 1:

```
If you transferred the extracted chart directories
helm install trust-manager ./trust-manager-chart/trust-manager
 \
  --namespace trust-manager \
  --set app.trust.namespace=netbackup \
  --wait
  # If you didn't modify values.yaml, add these overrides:
  # --set
app.image.repository=my-artifactory.mycompany.com/jetstack/trust-manager
 \
  # --set app.image.tag=v0.16.0 \
  # --set
pkgImporter.image.repository=my-artifactory.mycompany.com/jetstack/trust-pkg-debian-bookworm
 \
  # --set pkgImporter.image.tag=20230311.0 \
  # --set imagePullSecrets[0].name=artifactory-registry-secret
```

Option 2:

```
If you transferred packaged charts (.tgz files)
helm install trust-manager ./packages/trust-manager-v0.16.0.tgz \
  --namespace trust-manager \
  --set app.trust.namespace=netbackup \
  --wait
  # If you didn't modify values.yaml, add these overrides:
  # --set app.image.repository=my-artifactory.mycompany.com/jetstack/tr
  # --set app.image.tag=v0.16.0 \
  # --set pkgImporter.image.repository=my-artifactory.mycompany.com/jet
  # --set pkgImporter.image.tag=20230311.0 \
  # --set imagePullSecrets[0].name=artifactory-registry-secret
```

7 Verify the installation:

```
# Check if cert-manager pods are running
kubectl get pods -n cert-manager

# Check if trust-manager pod is running
kubectl get pods -n trust-manager -l app=trust-manager
```

Health checker for pre-deployment

Run pre-deployment health checks to validate Kubernetes cluster readiness and required configurations before deploying Cloud Scale (supported for Cloud Scale 11.2 and later).

For more information, See "Overview" on page 194.

Deploying the operators

To perform these steps, log on to the Linux workstation or VM where you have extracted the TAR file.

To deploy the operators

1. Use the following command to save the operators chart values to a file:

   ```
   helm show values operators-<version>.tgz operators-values.yaml
   ```

2. Use the following command to edit the chart values to fit your requirement:

   ```
   vi operators-values.yaml
   ```

3 Execute the following command to deploy the operators:

```
helm upgrade --install operators operators-<version>.tgz -f
operators-values.yaml --create-namespace --namespace
netbackup-operator-system
```

Or

If using the OCI container registry, use the following command:

```
helm upgrade --install operators
oci://abcd.veritas.com:5000/helm-charts/operators --version
<version> -f operators-values.yaml --create-namespace --namespace
netbackup-operator-system
```

Following is the output of the above command:

```
$ helm show values operators-11.2.x.x.xxxx.tgz >
operators-values.yaml
$
$ vi operators-values.yaml
$
$ helm upgrade --install operators operators-11.2.x.x.xxxx.tgz \
>         -f operators-values.yaml \
>         --create-namespace \
>         --namespace netbackup-operator-system
Release "operators" does not exist. Installing it now.
NAME: operators
LAST DEPLOYED: Tue Feb 27 00:01:29 2024
NAMESPACE: netbackup-operator-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Following is an example for `operators-values.yaml` file which includes the parameter for private registry support.

```
flexsnap-operator:
  image:
    name: veritas/flexsnap-deploy
    pullPolicy: Always
    tag: 11.2.0.0-1001-ar1
  namespace:
    labels: {}
  nodeSelector:
    node_selector_key: agentpool
```

```
    node_selector_value: nbuxpool
  replicas: 1
global:
  containerRegistry: name.azurecr.io
  operatorNamespace: netbackup-operator-system
  platform: aks
  storage:
    aks:
      storageAccountName: null
      storageAccountRG: null
    eks:
      fileSystemId: fs-id
  timezone: null
msdp-operator:
  corePattern: /core/core.%e.%p.%t
  image:
    name: msdp-operator
    pullPolicy: Always
    tag: 21.1-0002-ar1
  logging:
    age: 28
    debug: false
    num: 20
  namespace:
    labels:
      control-plane: controller-manager
  nodeSelector: {}
  replicas: 2
  resources:
    limits: {}
    requests:
      cpu: 150m
      memory: 150Mi
  storageClass:
    name: nb-disk-premium
    size: 5Gi
nb-operator:
  flexsnap-operator:
    image:
      tag: 11.2.0.0-1001-ar1
  image:
    name: netbackup/operator
    pullPolicy: Always
```

```
  tag: 11.2-0004-ar1
loglevel:
  value: "0"
msdp-operator:
  image:
    tag: 21.1-0002-ar1
namespace:
  labels:
    nb-control-plane: nb-controller-manager
nodeSelector:
  node_selector_key: agentpool
  node_selector_value: nbuxpool
```

Deploying Cloud Scale using Helm chart

This section describes the following deployment methods of Cloud Scale environment using Helm chart:

- Installing Cloud Scale environment
- Single node Cloud Scale deployment

Installing Cloud Scale environment

To install NetBackup using Helm charts on Kubernetes cluster, perform the following:

1. Download the NetBackup TAR package from Cohesity Download Center which contains the container images and the Helm chart for NetBackup Kubernetes application.
2. Deploy the operators. For more information on deploying the operators, refer to the following section:

   See "Deploying the operators" on page 136.
3. Perform the following steps to deploy the `cloudscale-values.yaml` file:
   - Use the following command to save the environment chart values to a file:

     `helm show values cloudscale-<version>.tgz > cloudscale-values.yaml`
   - Edit the chart values to fit the required environment:

     `vi cloudscale-values.yaml`
   - Use the following command to apply the `cloudscale-values.yaml` file:

```
helm upgrade --install cloudscale cloudscale-<version>.tgz -f
cloudscale-values.yaml --namespace netbackup
```

For example,

```
helm show values cloudscale-11.2.x.x.xxxx.tgz >
cloudscale-values.yaml
$
$ vi cloudscale-values.yaml
$
```

The output for the above example would be as follows:

```
environment:
  corePattern: '''/corefiles/core.%e.%p.%t'''
  cpServer:
    name: cpserver-1
    networkLoadBalancer:
      fqdn: nbux-xx-xxx-xx-xx.vxindia.cohesity.com
      ipAddr: xx.xxx.xx.xx
    nodeSelector:
      controlPlane:
        labelKey: pool
        labelValue: primary
      dataPlane:
        labelKey: pool
        labelValue: cp
    proxySettings: null
    storage:
      data:
        capacity: 30Gi
        storageClassName: nb-disk-standardssd
      log:
        capacity: 5Gi
        storageClassName: nb-file-premium
    tag: 11.2.0.0-1035-11.2-0045
  createServiceAccount: false
  drInfoSecretName: null
  emailServerConfigmapName: null
  enabled: true
  loadBalancerAnnotations:

service.beta.kubernetes.io/azure-load-balancer-internal-subnet:
LB-RESERVED
```

```
mediaServers:
- name: media1
  nodeSelector:
    labelKey: pool
    labelValue: media
  replicas: 1
  storage:
    data:
      capacity: 50Gi
      storageClassName: nb-disk-standardssd
    log:
      capacity: 5Gi
      storageClassName: nb-disk-standardssd
msdpScaleouts:
  annotations:
    msdp.veritas.com/baas: "false"
  credential:
    secretName: msdp-secret1
  ipList:
  - fqdn: nbux-xx-xxx-xx-xx.vxindia.cohesity.com
    ipAddr: xx.xxx.xx.xx
  kms:
    keyGroup: example-key-group
    keySecret: example-key-secret
  loadBalancerAnnotations: null
  name: dedupe1
  nodeSelector:
    labelKey: pool
    labelValue: msdp
  replicas: 1
  secret:
    credential:
      password: Y123abCdEf
      username: devuser
    kms:
      passphrase: Y123abC@dEf12345
      username: devuser
  storage:
    dataVolumes:
    - capacity: 5Gi
      storageClassName: nb-disk-premium
    log:
      capacity: 5Gi
```

```
        storageClassName: nb-disk-premium
    tag: 21.1-0033-11.2-0045
  name: nbu
  primary:
    credSecretName: primary-credential-secret
    kmsDBSecret: kms-secret
    networkLoadBalancer:
      ipList:
      - fqdn: nbux-xx-xxx-xx-xx.vxindia.cohesity.com
        ipAddr: xx.xxx.xx.xx
    nodeSelector:
      labelKey: pool
      labelValue: primary
    secret:
      credential:
        password: Gyp.s8m123
        username: devuser
      kms:
        HMKID: hmkid123
        HMKpassphrase: HMK@passphrase123
        KPKID: kpkid123
        KPKpassphrase: KPK@passphrase123
    storage:
      catalog:
        autoVolumeExpansion: false
        capacity: 100Gi
        storageClassName: nb-file-premium
      data:
        capacity: 30Gi
        storageClassName: nb-disk-standardssd
      log:
        capacity: 10Gi
        storageClassName: nb-disk-standardssd
  tag: 11.2-0045-11.2-0045
fluentbit:
  cleanup:
    highWatermark: 90
    image:
      name: netbackup/fluentbit-log-cleanup
      tag: 11.2-0045-11.2-0045
    lowWatermark: 60
    retentionCleanupTime: "04:00"
    retentionDays: 7
```

```
    utilizationCleanupFrequency: 60
  collectorNodeSelector:
    node_selector_key: pool
    node_selector_value: primary
  enabled: true
  image:
    name: netbackup/fluentbit
    pullPolicy: IfNotPresent
    tag: 11.2-0024-DR2
  metricsPort: 2020
  namespaces:
  - netback
  - netbackup-operator-system
  tolerations:
  - key: pool
    value: primary
  - key: pool
    value: msdp
  - key: pool
    value: cp
  - key: pool
    value: media
  volume:
    pvcStorage: 100Gi
    storageClassName: nb-disk-premium
global:
  containerRegistry: nbuk8sreg.azurecr.io
  dbSecretName: dbsecret
  environmentNamespace: netbackup
  timezone: null
postgresql:
  clientSecretName: postgresql-client-crt
  createCerts: true
  dataDir: /netbackup/postgresqldb
  dbAdminName: postgres
  dbPort: 13785
  defaultPostgresqlHostName: nb-postgresql
  enabled: true
  image:
    name: netbackup/postgresql
    pullPolicy: Always
    tag: 16.10.2.0-0001-11.2-0045
  initialDbAdminPassword: postgres
```

```
logDestination: file
nodeSelector:
  key: pool
  value: primary
pgbouncerPort: 13787
postgresqlUpgrade:
  env:
    dataDir: /netbackup/postgresqldb
  image:
    name: netbackup/postgresql-upgrade
    pullPolicy: Always
    tag: 16.10.2.0-0001-11.2-0045
  replicas: 1
  securityContext:
    runAsUser: 0
  volume:
    mountPathData: /netbackup/postgresqldb
    timezone: null
    volumeClaimName: nb-psql-pvc
replicas: 1
securityContext:
  runAsUser: 0
serverSecretName: postgresql-server-crt
service:
  serviceName: nb-postgresql
tolerations:
- key: pool
  value: primary
volume:
  mountPathData: /netbackup/postgresqldb
  pvcStorage: 30Gi
  secretMountPath: /netbackup/postgresql/keys/server
  storageClassName: nb-disk-premium
  volumeClaimName: nb-psql-pvc
  volumeDefaultMode: 416
```

Note: By default, Helm stores the input values in a Kubernetes secret. Hence to avoid the values being discovered by anyone, customers must reset the database password after Cloud Scale deployment.

Single node Cloud Scale Technology deployment

From version 10.5 or later, Cloud Scale Technology can be deployed on the single node of Kubernetes cluster. This deployment provides a cost-effective solution as it requires only one node in a Kubernetes cluster where all the Cloud Scale components get easily deployed.

In the single node deployment environment, all the micro services pods of the primary server share the same log volumes whereas the media server, storage server, and the cloud point server pods have their own log volumes. Only the media server deployment and MSDP server scale-out is not supported.

Prerequisites for single node Cloud Scale Technology deployment

For better single node Cloud Scale Technology deployment, follow the best practices and recommendations around sizing before Cloud Scale Technology deployment.

- Cluster auto scalar minimum and maximum node count value set to 1.
- **For AKS:** 16 CPU, 64 GB RAM and node type: standard D16_ds_V5, max pod per node is 60.
- **For EKS:** 16 vCPUs, 42 GiB RAM and node type: c5n.4xlarge

---

Note: Upgrading from the multi-node deployment to single node deployment is not supported whereas upgrading from the single node to single node deployment is supported.

Refer to the section See “Steps to deploy Cloud Scale in single node” on page 146. for the detail steps.

Steps to deploy Cloud Scale in single node

To deploy the Cloud Scale Technology on a single node in the Kubernetes cluster, perform the following steps:

- **Deploy operators**: To deploy operators, perform points mentioned in step 1.
- **Deploy Cloud Scale**: To perform settings in the Cloud Scale, check for points mentioned in step 4.

Steps for deployment

1 In the `operators-values.yaml file`, do the following changes:

  - Set 'replicas' to 1 in `msdp-operator`.

- Set value to `examplepool` for every `node_selector_value` in the nodeSelector section.
- Set value to `agentpool` for every `node_selector_key` in the nodeSelector section.
  To deploy the operators, refer to the section See "Deploying the operators" on page 136.

2. Do the following in the `cloudscale-values.yaml` file to deploy the Cloud Scale Technology:
   - Set value to **examplepool** for every **node_selector_value** and **labelValue** in the nodeSelector section.
   - Set value to "agentpool" for every `node_selector_key` and `labelKey` in the nodeSelector section.
   - Set 'replicas' to 1 for `mediaServers` and `msdpScaleouts` section.
   - Add the section in the `msdpScaleouts` section:

```
resources:
  requests:
    cpu: 6000m
    memory: 16384Mi
  limits:
    cpu: 9000m
    memory: 16384Mi
```

```
# save cloudscale chart values to a file
helm show values cloudscale-<version>.tgz > cloudscale-values.yaml

# edit chart values to fit your environment
vi cloudscale-values.yaml

helm upgrade --install cloudscale cloudscale-<version>.tgz -f  cloudscale
```

Deploying Cloud Scale using kubectl plugin

To deploy Cloud Scale using kubectl plugin

1. Download the NetBackup TAR package from Cohesity Download Center which contains the kubectl plugin.
2. Extract the contents of the TAR file.

3 Execute the following command:

```
./kubectl-cloudscale install
```

```
/xxx/xx/VRTSk8s-netbackup-11.2-xxxx/bin
kubectl-cloudscale  kubectl-msdp
[root@user bin]# ./kubectl-cloudscale -h
A kubectl plugin to manage cloudscale installation, upgrade and
uninstallation

Usage:
  kubectl-cloudscale [flags]
  kubectl-cloudscale [command]

Available Commands:
  help        Help about any command
  install     Install's the CloudScale setup
  uninstall   Uninstall the CloudScale setup.
  upgrade     Upgrades the CloudScale setup

Flags:
  -h, --help   help for kubectl-cloudscale

Use "kubectl-cloudscale [command] --help" for more information
about a command.
[root@user bin]#
```

4 Provide inputs for the installation that follow.

Note: You can skip the image validation for image tags by running the following command:

```
kubectl create configmap cs-image-validation-config -n
<netbackup-namespace> --from-literal=SKIP_IMAGE_VALIDATION=true
```

Note: The following CLI example is for AKS.

For AWS, eks-specific inputs would be required (such as file-system and so on).

```
Provide the file path of the extracted Cloud Scale folder:
/new-disk/VRTSk8s-netbackup-11.2-0004

Provide a namespace for the Environment: nb registry? (y/n): n

Provide the container registry where the Cloud Scale images are
stored (for example, example.azurecr.io or
123456789012.dkr.ecr.us-east-1.amazonaws.com):
nbuk8sreg.azurecr.io

Provide the image tag for NetBackup images: 11.2-0004-ar1
Validating images...

Provide the image tag for MSDP images: 21.1-0002-ar1
Validating images...

Provide the image tag for NetBackup Snapshot Manager images:
11.2.0.0-1001-ar1
Validating images...

Provide a name for the Environment: anuj

Provide the timezone (for example,
/usr/share/zoneinfo/Asia/Kolkata) or press enter to use the
default timezone:

Provide the node selector key value pair in the key=value format:
 agentpool=nbuxpool

Provide the node selector key value pair for the Media server in
```

```
 the key=value format: agentpool=mediapool

Provide the node selector key value pair for the MSDP server in
the key=value format: agenntpoool=msdpxpool

Provide the node selector key value pair for the data plane of
the NetBackup Snapshot Manager server in the key=value format:
agentpool = cppool

Provide the image tag for PostgreSQL images: 16.10.1.0-0001-ar1
Validating images...

Provide the prefix for the media server name: media

Specify the maximum number of replicas for the Media server: 1

Specify the scale-out size or replica for the MSDP server: 1

Provide the IP address for the primary server loadbalancer:
xx.xxx.xx.xx

Provide the FQDN for the Primary server loadbalancer:
nbux-xx-xxx-xx-xx.vxindia.cohesity.com

Provide the IP address for the MSDP server loadbalancer:
xx.xxx.xx.xx

Provide the FQDN for the MSDP server loadbalancer:
nbux-xx-xxx-xx-xx.vxindia.cohesity.com

Provide the IP address for NetBackup Snapshot Manager server
loadbalancer: xx.xxx.xx.xx

Provide the FQDN for NetBackup Snapshot Manager loadbalancer:
nbux-xx-xxx-xx-xx.vxindia.cohesity.com

Provide the username for the Primary server credentials: root

Provide the password for the Primary server credentials:

Confirm password:

Provide the HMK ID for the Primary server KMS database secret:
```

```
hmkid

Provide the HMK passphrase for the Primary server KMS database
secret:

Confirm password:

Provide the KPK ID for the Primary server KMS database secret:
kpkid

Provide the KPK passphrase for the Primary server KMS database
secret:

Confirm password:

Provide the username for the KMS database secret: devuser

Provide the passphrase for the KMS database secret:

Confirm password:

Provide the username for the MSDP server credentials: devuser

Provide the password for the MSDP server credentials:

Confirm password:

Specify the subnet name that will be used for the loadbalancer:
LB-RESERVED

Would you like to set up the proxy for CloudPoint? (y/n): n

Do you want to provide the information for the Disaster Recovery
 secret? (y/n): n

Do you want to set the email server configuration? (y/n): n
```

Verifying Cloud Scale deployment

Note: Cloud Scale deployment does not support ARM (Advanced RISC Machine) Architecture.

Execute the following command to verify if the Cloud Scale deployment is successful:

```
kubectl get
primaryserver,mediaserver,msdpscaleout,cpserver,environment -n
<netbackup-namespace>
```

The output should display the name and status of all the CRs. If the value of **STATUS** field for each CR is displayed as **Success**, then it indicates that the deployment is successful.

The output message for a successful Cloud Scale deployment is as follows:

```
# kubectl get
primaryserver,mediaserver,msdpscaleout,cpserver,environment -n
<netbackup-namespace>

NAME                                          TAG      AGE   STATUS
primaryserver.netbackup.veritas.com/gov-terraform1   11.0.x.x.xxxx
  20h   Success

NAME                                                       TAG
AGE   PRIMARY SERVER            STATUS
mediaserver.netbackup.veritas.com/gov-terraform1   11.0.x.x.xxxx
20h   10-0-131-11.usda.gov.in   Success

NAME                                          AGE   TAG    SIZE
READY
msdpscaleout.msdp.veritas.com/gov-terraform1   20h   21.0   1
1

NAME                                                  TAG          AGE
   STATUS
cpserver.netbackup.veritas.com/gov-terraform1   11.0.x.x.xxxx   20h
   Success

NAME                                                  READY   AGE
STATUS
environment.netbackup.veritas.com/gov-terraform1   4/4     20h
Success
```

For further confirmation, verify if the Web UI of the Primary Server is accessible through *https://<Primary Server's FQDN>/webui/login/*.

Post Cloud Scale deployment tasks

This section describes about restarting Cloud Scale technology services that must be performed after Cloud Scale deployment.

Health checker for post-deployment

Run post-deployment health checks to verify that Cloud Scale components are correctly installed, reachable, and properly configured after deployment (requires Cloud Scale 11.2 or later).

For more information, See “Overview” on page 194.

Restarting Cloud Scale Technology services

Up to NetBackup versions 10.4 in the Cloud Scale Technology, the primary server deployment was performed in a monolithic way with all the services running in a single primary pod. Utilities like `bp.kill_all` and `bp.start_all` were used to start or stop the primary pod services.

NBPEM/NBJM and BPDBM NetBackup services are decoupled from the primary server. As there is no separate script to restart the decoupled primary services, perform the following steps:

To restart the Cloud Scale Technology services

1. Navigate to the `VRTSk8s-netbackup-<version>/scripts` folder.
2. Run the `cloudscale_restart.sh` script as follows:

   ```
   ./cloudscale_restart.sh <action> <namespace>
   ```

   Provide the namespace and the required action:

   - **stop**: Stops all the services under primary server (waits until all the services are stopped).
   - **start**: Starts all the services and waits until the services are up and running under primary server.
   - **restart**: Stops the services and waits until all the services are down. Then starts all the services and waits until the services are up and running.

   Note: Ignore if policy job pod does not come up in running state. Policy job pod would start once primary services start.

Section 3

Monitoring and Management

Chapter 7

Monitoring NetBackup

This chapter includes the following topics:

Monitoring the application health

Kubernetes Liveness and Readiness probes are used to monitor and control the health of the NetBackup primary server and media server pods. The probes collectively also called as health probes, keep checking the availability and readiness of the pods, and take designated actions in case of any issues. The kubelet uses liveness probes to know when to restart a container, and readiness probes to know when a container is ready. For more information, refer to the Kubernetes documentation.

Configure Liveness, Readiness and Startup Probes | Kubernetes

The health probes monitor the following for the NetBackup deployment:

- Mount directories are present for the data/catalog at `/mnt/nbdata` and the log volume at `/mnt/nblogs`.
- `bp.conf` is present at `/usr/openv/netbackup`
- NetBackup services are running as expected.

Following table describes the actions and time intervals configured for the probes:

Table 7-1

Action	Description	Probe name	Primary server (seconds)	Media server (seconds)	Request router (seconds)
Initial delay	This is the delay that tells kubelet to wait for a given number of seconds before performing the first probe.	Readiness Probe	120	0	60
		Liveness Probe	300	90	120
Periodic execution time	This action specifies that kubelet should perform a probe every given number of seconds.	Readiness Probe	30	30	60
		Liveness Probe	90	90	60
Threshold for failure retries	This action specifies that kubelet should retry the probe for given number of times in case a probe fails, and then restart a container.	Readiness Probe	1	1	1
		Liveness Probe	5	5	5

Heath probes are run using the `nb-health` command. If you want to manually run the `nb-health` command, the following options are available:

- **Disable**
  This option disables the health check that will mark pod as not ready (0/1).
- **Enable**
  This option enables the already disabled health check in the pod. This marks the pod in ready state(1/1) again if all the NetBackup health checks are passed.
- **Deactivate**
  This option deactivates the health probe functionality in pod. Pod remains in ready state(1/1). This will avoid pod restarts due to health probes like liveness, readiness probe failure. This is the temporary step and not recommended to use in usual case.
- **Activate**
  This option activates the health probe functionality that has been deactivated earlier using the **deactivate** option.

You can manually disable or enable the probes if required. For example, if for any reason you need to exec into the pod and restart the NetBackup services, the health probes should be disabled before restarting the services, and then they should be enabled again after successfully restarting the NetBackup services. If you do not disable the health probes during this process, the pod may restart due to the failed health probes.

Note: It is recommended to disable the health probes only temporarily for troubleshooting purposes. When the probes are disabled, the web UI is not accessible in case of the primary server pod, and the media server pods cannot be scaled up. Then the health probes must be enabled again to successfully run NetBackup.

To disable or enable the health probes

1 Execute the following command in the Primary or media server pod as required:

```
kubectl exec -it -n <namespace> <primary/media-server-pod-name>
-- /bin/bash
```

2 To disable the probes, run the `/opt/veritas/vxapp-manage/nb-health disable` command. Then the pod goes into the **not ready (0/1)** state.

3 To enable the probes, run the "`/opt/veritas/vxapp-manage/nb-health enable`" command. Then the pod will be back into the ready (1/1) state.

You can check pod events in case of probe failures. To get more details use the `kubectl describe <primary/media-pod-name> -n <namesapce>` command.

To disable or enable the request router health probes

1 Execute the following command in the request router pod as required:

```
kubectl exec -it -n <namespace> <request-router-pod-name> --
/bin/bash
```

2 To disable the probes, run the `/opt/veritas/vxapp-manage/health.sh disable` command. Then the pod goes into the **not ready (0/1)** state.

3 To enable the probes, run the `/opt/veritas/vxapp-manage/health.sh enable` command. Then the pod will be back into the **ready (1/1)** state.

You can check pod events in case of probe failures. To get more details use the `kubectl describe -n` command.

Telemetry reporting

Telemetry reporting entries for the NetBackup deployment on AKS/EKS are indicated with the **AKS/EKS based deployments** text.

- By default, the telemetry data is saved at the `/var/veritas/nbtelemetry/` location. The default location will not persisted during the pod restarts.
- If you want to save telemetry data to persisted location, then execute the `kubectl exec -it -n <namespace> <primary/media_server_pod_name> - /bin/bash` command in the pod using the and execute telemetry command using `/usr/openv/netbackup/bin/nbtelemetry` with `--dataset-path=DESIRED_PATH` option.
  - Exec into the primary server pod using the following command:
    `kubectl exec -it -n <namespace> <primary/media_server_pod_name> -- /bin/bash`
  - Execute telemetry command using `/usr/openv/netbackup/bin/nbtelemetry` with `--dataset-path=DESIRED_PATH`

  Note: Here `DESIRED_PATH` must be `/mnt/nbdata or /mnt/nblogs`.

- Operator also collects certain telemetry data and it is dumped inside the primary server at location - `/usr/openv/var/global/telemetry/tmp` .

About NetBackup operator logs

Note the following about the NetBackup operator logs.

- NetBackup operator logs can be checked using the operator pod logs using the `kubectl logs <Netbackup-operator-pod-name> -c netbackup-operator -n <netbackup-opertaor-namespace>` command.
- NetBackup operator provides different log levels that can be changed before deployment of NetBackup operator.
  The following log levels are provided:
  - -1 - Debug
  - 0 - Info
  - 1 - Warn
  - 2 - Error

  By default, the log level is 0.
  It is recommended to use 0, 1, or 2 log level depending on your requirement.
  To change the log level modify the `operators-values.yaml` file and upgrade the operators using the following command:

`helm upgrade --install operators operators-<version>.tgz -f operators-values.yaml -n netbackup-operator-system`

- Config-Checker jobs that run before deployment of primary server and media server creates the pod. The logs for config checker executions can be checked using the `kubectl logs <configchecker-pod-name> -n <netbackup-operator-namespace>` command.
- Installation logs of NetBackup primary server and media server can be retrieved using any of the following methods:
  - Run the `kubectl logs <PrimaryServer/MediaServer-Pod-Name> -n <PrimaryServer/MediaServer namespace>` command.
  - Execute the following command in the primary server/media server pod and check the `/mnt/nblogs/setup-server.log` file:

    `kubectl exec -it <PrimaryServer/MediaServer-Pod-Name> -n <PrimaryServer/MediaServer-namespace> -- bash`
- (*AKS-specific*) Data migration jobs create the pods that run before deployment of primary server. The logs for data migration execution can be checked using the following command:

  `kubectl logs <migration-pod-name> -n <netbackup-environment-namespace>`
- Execute the following respective commands to check the event logs that shows deployment logs for PrimaryServer, MediaServer and Request Router:
  - For primary server: `kubectl describe PrimaryServer <PrimaryServer name> -n <PrimaryServer-namespace>`
  - For media server: `kubectl describe MediaServer<MediaServername> -n<MediaServer-namespace>`
  - For request router: `kubectl describe pod <request-router-pod-name> -n <PrimaryServer-namespace>`

Monitoring Primary/Media server CRs

You can view the status and other details of the primary server and media server CRs using the following commands:

- `kubectl get <PrimaryServer/MediaServer> -n <namespace>` or
- `kubectl describe <PrimaryServer/MediaServer> <CR name> -n <namespace>`

Following table describes the primary server CR and media server CR status fields:

Table 7-2 Primary server and media server CR status fields

Section	Field / Value	Description
Primary Server Details Only one hostname and IP address for the respective primary server.	Host Name	Name of the primary server that should be used to access the web UI.
	IP	IP address to access the primary server.
	Version	This indicates that the NetBackup primary server version is installed.

Table 7-2 Primary server and media server CR status fields *(continued)*

Section	Field / Value	Description	
Media Server Details Number of hostname and IP address is equal to the replica count mentioned CR spec.	Host Name	Name of the media server.	
	Version	This indicates that the NetBackup media server version is installed.	
	Attributes	Resource Name	Statefulset name of the respective server.
		Primary/Media server name	Name of the primary server or media server deployed.
		Config checker status	Indicates the status of the config checker as passed, failed, or skipped. For more information on the Config-Checker, See the section called "How does the Config-Checker utility work" on page 44.
		SHA Fingerprint	Represents the SHA key fingerprint of the NetBackup primary server. **Note:** SHAFingerprint represents the SHA256 CA certificate fingerprint of the primary server.

Table 7-2 Primary server and media server CR status fields *(continued)*

Section	Field / Value	Description
ElasticityAttributes		Describes the attributes associated with the media server autoscaler. For more information on the elasticity attributes, See "Elastic media server" on page 99.
Error Details	Code	A code assigned to an error encountered during the CR deployment or during the config-check operation. For more information on the error, refer to the *NetBackup Status Code Reference Guide*.
	Message	Message that describes the respective error code.

Table 7-2 Primary server and media server CR status fields *(continued)*

Section	Field / Value	Description
State	Success /Paused /Failed /Running	Current state of the custom resource, from one of the following: ■ Success: Indicates that the deployment of Primary/Media Custom Resource (CR) is successful. However, this does not mean that the installation of the NetBackup primary/media servers is successful. The primary or media server StatefulSets and/or the pods might or might not be in a **ready** state, irrespective of that the Primary/Media CR state will show as Success. ■ Paused: Indicates that the reconciler is in **paused** state and deployment of a CR is paused. ■ Failed: Indicates that the deployment of a CR failed with errors. However this does not mean failed installation of the NetBackup Server. Errors can be analyzed from the Operator logs or CR **describe**. ■ Running: Indicates that the CR deployment is in progress and the resources are getting created.
Events	INIT/FAILOVER/UPGRADE	The events like INIT, FAILOVER and UPGRADE are logged in here. The details of these events can also be added.

Table 7-2 Primary server and media server CR status fields *(continued)*

Section	Field / Value	Description
Services	■ Name string ■ State string ■ Ready string ■ ImageTag string ■ Service string	This field indicates the status of the pods, their running state, image tag used and name of the service.

Expanding storage volumes

You can update storage capacity of already created persistent volume claim for primary server and media server. Expanding storage volume for particular replica of respective CR object is not supported. In case of media server user needs to update volumes for all the replicas of particular media server object.

(*AKS-specific*) To expand storage capacity of catalog volume in primary server

1 Run the following command to edit the environment custom resource, update the storage capacity for respective volume:

```
helm upgrade cloudscale cloudscale.tgz -n <namespace>
--reuse-values --set
environment.primary.storage.catalog.capacity=<expandedcapacity>Gi
```

2 Save the changes.

PVC will expand as per the new size and it will be available to volume mounts in primaryServer pod.

To expand volume of data and log volumes for primary and media server

---

Note: (*EKS-specific*) Amazon EFS is an elastic file system, it does not enforce any file system capacity limits. The actual storage capacity value in persistent volumes and persistent volume claims is not used when creating the file system. However, because storage capacity is a required field in Kubernetes, you must specify a valid value. This value does not limit the size of your Amazon EFS file system.

---

1 Run the following command:

- For data volumes:

  ```
  helm upgrade cloudscale cloudscale.tgz -n
  <namespace>--reuse-values
  --setenvironment.primary.storage.data.capacity=<expandedcapacity>Gi
  ```

- For log volumes:

  ```
  helm upgrade cloudscale cloudscale.tgz -n
  <namespace>--reuse-values --set
  environment.primary.storage.log.capacity=<expandedcapacity>Gi
  ```

2. Run the `kubectl get pvc -n <namespace>` command and check for **capacity** column in result to check the persistent volume claim storage capacity is expanded.

3. (Optional) Update the log retention configuration for NetBackup depending on the updated storage capacity.

   For more information, refer to the `NetBackup™ Administrator's Guide, Volume I`

Allocating static PV for Primary and Media pods

For allocating static PV for primary and media pods, refer to the following respective sections:

- (AKS-specific) See "(AKS-specific) Allocating static PV for Primary and Media pods" on page 168.
- (EKS-specific) See "(EKS-specific) Allocating static PV for Primary and Media pods" on page 172.

Expanding log volumes for primary pods

To optimize and reduce the log sizes of the PV's of the decoupled services, execute the following steps:

To reduce the log size of the PV

1. **Stop the primary pods:**
   - Untar the .tar for the kubernetes package, and within the untar folder, navigate to `scripts/`.
   - Navigate to the scripts folder inside the build folder (For example: `VRTSk8s-netbackup-<version>/scripts`).
   - Run the script `cloudscale_restart.sh`, with stop (as the input for action) and the netbackup namespace (for namespace parameter).
     For example: `./cloudscale_restart.sh stop <namespace>` This script will pause the primary server CR and stop all the decoupled services.

2. **Manually delete the statefulsets:**

- After all the pods are scaled down, get the required list of statefulsets using the command:
  `kubectl get sts -n <netbackup_namespace> | grep -e "nbatd" -e "nbmqbroker" -e "nbwsapp" -e "bpdbm" -e "policyjob" -e "policyjobmgr" -e "primary"`
- Delete the statefulsets using the command:
  `kubectl delete statefulset <statefulset_names_obtained_above> -n <netbackup_namespace>`

3 **List the log PVCs for primary pods:**

- Execute the command:
  `kubectl get pvc -n <netbackup_namespace> | grep log | grep -ve "catalog" -ve "uss" -ve "media"`
  For example: `kubectl get pvc -n netbackup | grep log | grep -ve "catalog" -ve "uss" -ve "media"`

4 **Expand the log PVCs**

- Expand the capacities of the above obtained log PVCs using the command:
  `kubectl patch pvc <pvc_names_obtained_above> -n <netbackup_namespace> -p '{"spec":{"resources":{"requests":{"storage":"<Expanded_capacity>Gi"}}}}'`
  For example: `kubectl patch pvc logs-nbu-primary-0 -n netbackup -p '{"spec":{"resources":{"requests":{"storage":"35Gi"}}}}'`

5 **Expand log volume size in primary server CR**

- Execute the command: `helm upgrade cloudscale cloudscale.tgz -n <namespace>--reuse-values --set environment.primary.storage.log.capacity=35Gi`

6 **Start decoupled services** using the `cloudscale_restart.sh` script:

- Navigate to the scripts folder inside the build folder (For example: `VRTSk8s-netbackup-<version>/scripts`).
- Run the script `cloudscale_restart.sh` with start (as the input for action) and the netbackup namespace (for namespace parameter).
  For example: `./cloudscale_restart.sh start <namespace>`

Recommendation for media server volume expansion

All media server pods are terminated if media server volumes are expanded. Ensure that no jobs are running on media servers when volume must be expanded.

During media server volume expansion, it is recommended that the value of **replicas** must be equal to or greater than the media servers added in primary server. Navigate to **Storage** > **Media servers** to check the number of media servers added in primary server on Web UI.

Primary server is also a media server that must not be considered. External media servers must not be considered.

During volume expansion, if media servers added in primary server are more than the value of **replicas** (that is, user has reduced the value for replicas field) then the volumes of provided **replicas** value would be expanded. If the value for **replicas** is increased again irrespective of volume expansion and if media server must be scaled up, then all the media server pods would be terminated and would be re-deployed with all PVCs expanded. This might fail the backup jobs as media servers may be terminated.

Scenarios that can occur during media server volume expansion

Expanding the volume of a media server may trigger pod restarts to apply the changes, and existing PVCs can be updated to ensure that the media server pods have sufficient storage volumes available for their operation. These scenarios are described in detail as follows:

- **Pod restart on volume expansion**
  If the volume of a running media server's data/log volume is expanded, the pod associated with that volume will restart automatically to apply the changes. This is because Kubernetes dynamically manages volumes and mounts, so any changes to the underlying volume will require the associated pod to restart to pick up the changes.
- **Updating PVCs for insufficient volumes**
  If there are existing Persistent Volume Claims (PVCs) running but with insufficient amounts of log/data volume, they can be updated to meet the requirements of the media server pods.
  PVCs define the requirements for storage volumes used by pods. By updating the PVCs, you can specify larger volumes to accommodate the media server's needs.
  Once the PVCs are updated, Kubernetes will automatically resize the underlying volumes to match the new PVC specifications, ensuring that the media server pods have access to the required storage capacity.

(AKS-specific) Allocating static PV for Primary and Media pods

When you want to use a disk with specific performance parameters, you can statically create the PV and PVC. You must allocate static PV and PVC before deploying the NetBackup server for the first time.

To allocate static PV for Primary and Media pods

1 Create storage class in cluster as per recommendations.

See the section called "How does the Config-Checker utility work" on page 44. for storage class recommendation.

This newly created storage class name is used while creating PV and PVC's and should be mentioned for Catalog, Log, Data volume in the environment CR in primaryServer and mediaServer section at the time of deployment.

For more information on creating storage class, see Create a custom storage class.

For example,

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: managed-premium-retain
provisioner: disk.csi.azure.com
reclaimPolicy: Retain
allowVolumeExpansion: true
volumeBindingMode: Immediate
parameters:
  storageaccounttype: Premium_LRS
  kind: Managed
```

2 Calculate number of disks required.

The following persistent volumes are required by Primary and Media pods:

- Catalog, Data and Log volume disk for primary server.
- Data and Log volume disk per replica of media server.

Use the following format to form PVC names.

For primary server

- catalog-<resourceNamePrefix_of_primary>-primary-0
- data-<resourceNamePrefix_of_primary>-primary-0
- logs-<resourceNamePrefix_of_primary>-primary-0

For media server

- data-<resourceNamePrefix_of_media>-media-<media server replica number. Count starts from 0>
- logs-<resourceNamePrefix_of_media>-media-<media server replica number. Count starts from 0>

Example 1	If user wants to deploy a media server with replica count 3.
	
	Names of the Media PVC assuming resourceNamePrefix_of_media is **testmedia**.
		For this scenario, you must create total 8 disks, 8 PV and 8 PVCs.
		
		6 disks, 6 PV and 6 PVCs for media server.
		
		Following will be the names for:
		
		Primary server
		■ For catalog: catalog-testprimary-primary-0
		■ For data: data-testprimary-primary-0
		■ For logs: logs-testprimary-primary-0
		
		Media server
		■ For data:
		  ■ data-testmedia-media-0
		  ■ data-testmedia-media-1
		  ■ data-testmedia-media-2
		■ For log:
		  ■ logs-testmedia-media-0
		  ■ logs-testmedia-media-1
		  ■ logs-testmedia-media-2

Example 2	If user wants to deploy a media server with replica count 5 Names of the Media PVC assuming resourceNamePrefix_of_media is **testmedia**.	For this scenario, you must create 12 disks, 12 PV and 12 PVCs 10 disks, 10 PV and 10 PVCs for media server. Following will be the names for media server volumes For data: ■ data-testmedia-media-0 ■ data-testmedia-media-1 ■ data-testmedia-media-2 ■ data-testmedia-media-3 ■ data-testmedia-media-4 For log: ■ logs-testmedia-media-0 ■ logs-testmedia-media-1 ■ logs-testmedia-media-2 ■ logs-testmedia-media-3 ■ logs-testmedia-media-4

3 Create required number of Azure disks and save the ID of newly created disk.

For more information, see Azure Disk - Static

4 Create PVs for each disk and link the PVCs to respective PVs.

To create the PVs, specify the created storage class and diskURI (ID of the disk received in step 3) in the yaml file. The PV must be created using the **claimRef** field and provide PVC name for its corresponding namespace.

For example, if you are creating PV for catalog volume, storage required is 128GB, diskName is **primary_catalog_pv** and namespace is **test**. PVC named **catalog-testprimary-primary-0** is linked to this PV when PVC is created in the namespace test.

```
apiVersion: v1
      kind: PersistentVolume
      metadata:
        name: catalog
           spec:
               capacity:
                       storage: 128Gi
               accessModes:
                       - ReadWriteOnce
               azureDisk:
                        kind: Managed
                        diskName: primary_catalog_pv
                        diskURI:
/subscriptions/3247febe-4e28-467d-a65c-10ca69bcd74b/
resourcegroups/MC_NBU-k8s-network_xxxxxx_eastus/providers/Microsoft.Compute/disks/deepak_s_pv

               claimRef:
                          apiVersion: v1
                          kind: PersistentVolumeClaim
                          name: catalog-testprimary-primary-0
                          namespace: test
```

5 Create PVC with correct PVC name (step 2), storage class and storage.

For example,

```
apiVersion: v1
               kind: PersistentVolumeClaim
               metadata:
                 name: catalog-testprimary-primary-0
                 namespace: test
               spec:
                 storageClassName: "managed-premium-retain"
                 accessModes:
                        - ReadWriteOnce
                 resources:
                        requests:
                          storage: 128Gi
```

6 Perform step 4 and 5 for all of the following:

Primary server: Catalog, Data and Logs volume.

Media server: Logs and Data volume.

7 Deploy the Operator.

8 Use previously created storage class names for the volumes in primaryServer and mediaServer section in environment CR spec and deploy environment CR.

(EKS-specific) Allocating static PV for Primary and Media pods

When you want to use a disk with specific performance parameters, you can statically create the PV and PVC. You must allocate static PV and PVC before deploying the NetBackup server for the first time.

To allocate static PV for Media and Primary pods

1. Create storage class in cluster.

   See the section called “How does the Config-Checker utility work” on page 44.

   This newly created storage class name is used while creating PV and PVC's and should be mentioned for Catalog, Log, Data volume in the environment CR mediaServer section at the time of deployment.

   For more information on creating the storage class, see Storage class.

   For example,

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: gp2-reclaim
provisioner: kubernetes.io/aws-ebs
reclaimPolicy: Retain
allowVolumeExpansion: true
volumeBindingMode: Immediate
parameters:
  fsType: ext4
  type: gp2
```

2. Calculate number of disks required.

   The following persistent volumes are required by Media and Primary pods:

   - Catalog, Data and Log volume disk for primary server.
   - Data and Log volume disk per replica of media server.

   Use the following format to form PVC names.

   For primary server:

   - catalog-<resourceNamePrefix_of_primary>-primary-0
   - data-<resourceNamePrefix_of_primary>-primary-0
   - logs-<resourceNamePrefix_of_primary>-primary-0

   For media server

   - data-<resourceNamePrefix_of_media>-media-<media server replica number. Count starts from 0>
   - logs-<resourceNamePrefix_of_media>-media-<media server replica number. Count starts from 0>

<table>
<tr>
<td>Example 1</td>
<td>If user wants to deploy a media server with replica count 3.

Name of the Media PVC assuming resourceNamePrefix_of_media is testmedia.</td>
<td>For this scenario, you must create total 8 disks, 8 PV and 8 PVCs.

6 disks, 6 PV and 6 PVCs for media server.

Following will be the names for:

Primary server

■ For logs: logs-testprimary-primary-0
■ For catalog: catalog-testprimary-primary-0
■ For data: data-testprimary-primary-0

Media server

■ For data:
  ■ data-testmedia-media-0
  ■ data-testmedia-media-1
  ■ data-testmedia-media-2
■ For log:
  ■ logs-testmedia-media-0
  ■ logs-testmedia-media-1
  ■ logs-testmedia-media-2</td>
</tr>
<tr>
<td>Example 2</td>
<td>If user wants to deploy a media server with replica count 5

Names of the Media PVC assuming resourceNamePrefix_of_media is testmedia.</td>
<td>For this scenario, you must create 12 disks, 12 PV and 12 PVCs

10 disks, 10 PV and 10 PVCs for media server.

Following will be the names for primary server volumes

For data:

■ data-testmedia-media-0
■ data-testmedia-media-1
■ data-testmedia-media-2
■ data-testmedia-media-3
■ data-testmedia-media-4

For log:

■ logs-testmedia-media-0
■ logs-testmedia-media-1
■ logs-testmedia-media-2
■ logs-testmedia-media-3
■ logs-testmedia-media-4</td>
</tr>
</table>

3 Create the required number of AWS EBS volumes and save the VolumeId of newly created volumes.

For more information on creating EBS volumes, see EBS volumes.

(For Primary Server volumes) Create the required number of EFS. User can use single EFS to mount catalog of primary. For example, VolumeHandle in **PersistentVolume** spec will be as follows:

```
<file_system_id>:/catalog
```

4 Create PVs for each disks.

To create the PVs, specify the created storage class and VolumeId (ID of the EBS volumes received in step 3) in the yaml file. The PV must be created using the **claimRef** field and provide PVC name for its corresponding namespace.

For example, if you are creating PV for catalog volume, storage required is 128GB and namespace is **test**. PVC named **catalog-testprimary-primary-0** is linked to this PV when PVC is created in the namespace test.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: catalog
spec:
  accessModes:
  - ReadWriteMany
  awsElasticBlockStore:
    fsType: xfs
    volumeID: aws://us-east-2c/vol-xxxxxxxxxxxxxxxxx
  capacity:
    storage: 128Gi
  persistentVolumeReclaimPolicy: Retain
  storageClassName: gp2-retain
  volumeMode: Filesystem
  claimRef:
    apiVersion: v1
    kind: PersistentVolumeClaim
    name: catalog-testprimary-primary-0
    namespace: test
```

5 Create PVC with correct PVC name (step 2), storage class and storage.

For example,

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: catalog-testprimary-primary-0
namespace: test
spec:
  storageClassName: gp2-retain
accessModes:
  - ReadWriteMany
resources:
  requests:
  storage: 128Gi
```

6 Perform step 4 and 5 for all of the following:

Primary server: Catalog, Data and Logs volume.

Media server: Logs and Data volume.

7 Deploy the Operator.

8 Use previously created storage class names for the volumes in primaryServer and mediaServer section in environment CR spec and deploy environment CR.

Chapter 8

Monitoring Snapshot Manager

This chapter includes the following topics:

- Overview
- Configuration parameters
- Snapshot Manager manual certificate renewal in Cloud Scale

Overview

The status of Snapshot Manager deployment can be verified by using the following command:

```
kubectl describe cpserver -n $ENVIRONMENT_NAMESPACE
```

This displays the status of deployment as follows:

Status	Description
Running	Deployment is in progress.
Failed	Deployment has failed.
Success	Deployment is successful.
Paused	Reconciliation is paused.

Configuration parameters

- Any configuration related parameter that must be added in `/cloudpoint/flexsnap.conf` file can be added in `flexsnap-conf` configmap by editing it as follows:

  `kubectl edit configmap flexsnap-conf -n $ENVIRONMENT_NAMESPACE`

  For example, for changing the log level from info to debug, add the following:

  ```
  [logging]
  level = debug
  ```

- Any configuration related parameter which needs to be added in `/cloudpoint/openv/netbackup/bp.conf` file can be added in `nbuconf` configmap by editing it as follows:

  `kubectl edit configmap nbuconf -n $ENVIRONMENT_NAMESPACE`

Snapshot Manager manual certificate renewal in Cloud Scale

Trigger mechanism

Certificate renewal is initiated manually by modifying the environment configuration file:

1. Under the `cpServer` section, set the `renew` flag to `True`.
2. The flexsnap operator continuously monitors this flag for changes.
3. When the flag transitions from `False to True`, the operator triggers a certificate renewal process.
4. To initiate renewal again in the future, users must first reset the flag to `False`, then set it back to `True`.

Certificate revocation and generation

Once the renewal is triggered:

1. The flexsnap operator starts a Kubernetes job with **flexsnap-certgen** as the entry point, passing the renew argument.
2. The **flexsnap-certgen** utility performs the following steps:
   - Iterates through a list of services (either from `/cloudpoint/keys` or the predefined service list in `flexsnap-certauth`).
   - Revokes and regenerates certificates for each service.
   - Creates new kubernetes secrets containing the regenerated certificates.

Chapter 9

Monitoring fluentbit

This chapter includes the following topics:

- Monitoring fluentbit for logging

Monitoring fluentbit for logging

All pods and containers that are part of the fluentbit logging system (including the fluentbit-collector, fluentbit sidecars, fluentbit Daemonsets and the Log-viewer) produce logs to stdout. These logs can be viewed via kubectl commands as described in the Table 9-1 or by standard log viewing and extraction options described in the 'Managing logging' chapter.

Using the following commands, you can debug the logs of the logging pods and containers:

Table 9-1 Commands for debugging the logs

Command	Usage
`$ kubectl describe pod -n <netbackup namespace> <pod name>`	To check the status of the pods and information about the sidecars.
`$ kubectl logs -n <netbackup namespace> <pod name>`	To read (view) the logs of the primary container of a pod including the fluentbit collector pod.
`$ kubectl logs -n <netbackup namespace> <pod name> -c <container name>`	If the container you are looking at is a sidecar you can specify the container with -c.

Note: The appearance of some logs depends on whether the infrastructure nodes are scheduled in the nodes with NetBackup services, that are picked up by the daemonset.

Monitoring MSDP Scaleout

This chapter includes the following topics:

About MSDP Scaleout status and events

The MSDP Scaleout CR status includes the readiness state, the storage space utilization (via PersistentVolumeClaim) of each Controller, MDS, and Engine pod.

In the initial configuration of MSDP Scaleout, the readiness state of each pod changes from "false" to "true" in the first few minutes. When the state of all the pods changes to "true", it indicates MSDP Scaleout is ready for use.

You can check the storage space utilization routinely to plan MSDP Scaleout autoscaling before the storage space runs out.

To check the MSDP Scaleout status and events

1 Check the status and the events under the namespace for MSDP Scaleout.

```
kubectl -n <sample-namespace> describe msdpscaleout
<sample-cr-name>
```

2 Check the MSDP Scaleout events.

```
kubectl -n <sample-namespace> get events
[--sort-by='{.lastTimestamp}']
```

3 Check the storage space utilization.

```
kubectl -n <sample-namespace> get msdpscaleout <sample-cr-name>
-o json
```

Example of the of the status format:

```
kubectl -n sample-cr-namespace get msdpscaleout sample-cr -o json
```

AKS:

```
{
  "controllers": [
    {
      "apiVersions": [
        "1.0"
      ],
      "name": "msdp-aks-demo-uss-controller",
      "nodeName": "aks-nodepool1-25250377-vmss000002",
      "productVersion": "15.1-0159",
      "pvc": [
        {
          "pvcName": "msdp-aks-demo-uss-controller-log",
          "stats": {
            "availableBytes": "10125.98Mi",
            "capacityBytes": "10230.00Mi",
            "percentageUsed": "1.02%",
            "usedBytes": "104.02Mi"
          }
        }
      ],
      "ready": "True"
    }
  ],
  "engines": [
    {
```

```
      "ip": "x.x.x.x",
      "name": "msdppods1.westus2.cloudapp.azure.com",
      "nodeName": "aks-nodepool1-25250377-vmss000003",
      "pvc": [
        {
          "pvcName": "msdppods1.westus2.cloudapp.azure.com-catalog",
          "stats": {
            "availableBytes": "20293.80Mi",
            "capacityBytes": "20470.00Mi",
            "percentageUsed": "0.86%",
            "usedBytes": "176.20Mi"
          }
        },
        {
          "pvcName": "msdppods1.westus2.cloudapp.azure.com-data-0",
          "stats": {
            "availableBytes": "30457.65Mi",
            "capacityBytes": "30705.00Mi",
            "percentageUsed": "0.81%",
            "usedBytes": "247.35Mi"
          }
        }
      ],
      "ready": "True"
    },
    ......
```

EKS:

```
# kubectl get -n demo msdpscaleout msdp-app -o json |jq .status
{
  "controllers": [
    {
      "apiVersions": [
        "1.0"
      ],
      "name": "msdp-app-uss-controller",
      "nodeName": "ip-x-x-x-x.ec2.internal",
      "productVersion": "16.0.1-0035",
      "pvc": [
        {
          "pvcName": "msdp-app-uss-controller-log",
          "stats": {
            "availableBytes": "9878.00Mi",
```

```
            "capacityBytes": "9951.27Mi",
            "percentageUsed": "0.58%",
            "usedBytes": "57.27Mi"
          }
        }
      ],
      "ready": "True"
    }
  ],
  "engines": [
    {
      "ip": "x.x.x.x",
      "name": "ip-x-x-x-x.ec2.internal",
      "nodeName": "ip-x-x-x-x.ec2.internal",
      "pvc": [
        {
          "pvcName": "ip-x-x-x-x.ec2.internal-catalog",
          "stats": {
            "availableBytes": "604539.68Mi",
            "capacityBytes": "604629.16Mi",
            "percentageUsed": "0.01%",
            "usedBytes": "73.48Mi"
          }
        },
        {
          "pvcName": "ip-x-x-x-x.ec2.internal-data-0",
          "stats": {
            "availableBytes": "4160957.62Mi",
            "capacityBytes": "4161107.91Mi",
            "percentageUsed": "0.00%",
            "usedBytes": "134.29Mi"
          }
        }
      ],
      "ready": "True"
    },
```


Monitoring with Amazon CloudWatch

You can use Amazon CloudWatch to collect Prometheus metrics to monitor pods in MSDP-X cluster.

To configure Amazon CloudWatch

1. Install the CloudWatch agent with Prometheus metrics collection on EKS.

   See AWS documentation.

2. Install the CloudWatch agent on EKS clusters. Select the EC2 launch type, and download the template YAML file `Prometheus-eks.yaml`.

3 Add the YAML file with the following sample configuration.

```
# create configmap for prometheus cwagent config
apiVersion: v1
data:
  # cwagent json config
  cwagentconfig.json: |
    {
      "logs": {
        "metrics_collected": {
          "prometheus": {
            "prometheus_config_path": "/etc/prometheusconfig/
             prometheus.yaml",
            "emf_processor": {
              "metric_declaration": [
                {
                  "source_labels": ["job"],
                  "label_matcher": "^msdpoperator-metrics",
                  "dimensions":[
                    ["ClusterName","NameSpace"]
                  ],
                  "metric_selectors": [
                    "msdpoperator_reconcile_failed",
                    "msdpoperator_operator_run",
                    "msdpoperator_diskFreeLess5GBEngines_total",
                    "msdpoperator_diskFreeMiBytesInEngine",
                    "msdpoperator_diskFreeLess10GBClusters_total",
                    "msdpoperator_totalDiskFreePercentInCluster",
                    "msdpoperator_diskFreePercentInEngine",
                    "msdpoperator_pvcFreePercentInCluster",
                    "msdpoperator_unhealthyEngines_total",
                    "msdpoperator_createdPods_total"
                  ]
                }
              ]
            }
          }
        },
        "force_flush_interval": 5
      }
    }
kind: ConfigMap
metadata:
```

```
  name: prometheus-cwagentconfig
  namespace: amazon-cloudwatch

---
# create configmap for prometheus scrape config
apiVersion: v1
data:
  # prometheus config
  prometheus.yaml: |
    global:
      scrape_interval: 1m
      scrape_timeout: 10s
    scrape_configs:
    - job_name: 'msdpoperator-metrics'

      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        insecure_skip_verify: true
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount
      /token

      kubernetes_sd_configs:
      - role: pod

      relabel_configs:
      - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io
        _scrape]
        action: keep
        regex: true
      - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io
        _path]
        action: replace
        target_label: __metrics_path__
        regex: (.+)
      - source_labels: [__address__, __meta_kubernetes_pod_annotation_
        prometheus_io_port]
        action: replace
        regex: ([^:]+)(?::\d+)?;(\d+)
        replacement: $1:$2
        target_label: __address__
      - source_labels: [__meta_kubernetes_namespace]
        action: replace
```

```
          target_label: NameSpace
        - source_labels: [__meta_kubernetes_pod_name]
          action: replace
          target_label: PodName

kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: amazon-cloudwatch
```

Table 10-1 lists the Prometheus metrics that MSDP Scaleout supports.

4 Apply the YAML file.

```
Kubectl apply -f Prometheus-eks.yaml
```

The default log groups name is `/aws/containerinsights/{cluster_name}/Prometheus`.

5 Create Amazon CloudWatch alarms.

See Using Amazon CloudWatch alarms in AWS documentation.

6 In the CloudWatch console, add the related log query. In the navigation pane, select **Log Insights**.

For example, the free space size of the MSDP scaleout cluster engines is lower than 1 GB in past 5 minutes. Select the log group from the drop-down list, and select the time duration 5m on the time line.

Log query:

```
fields @timestamp, @message
| filter msdpoperator_diskFreeMiBytesInEngine <= 100000
| sort @timestamp desc
```

If multiple MSDP scaleout clusters are deployed in the same EKS cluster, use the filter to search the results. For example, search the MSDP engines with the free space size lower than 1GB in the namespace **sample-cr-namespace**.

Log query:

```
fields @timestamp, @message
| filter msdpscalout_ns == "sample-cr-namespace"
| filter msdpoperator_diskFreeMiBytesInEngine <= 100000
| sort @timestamp desc
```

MSDP Scaleout supports the following Prometheus metrics:

Table 10-1 Supported Prometheus metrics list in MSDP Scaleout

Metrics	Type	Filters	Description
msdpoperator_reconcile_total	Counter	N/A	The total of the reconcile loops msdp-operator run.
msdpoperator_reconcile_failed	Counter	N/A	The total of the reconcile loops msdp-operator failed to run.
msdpoperator_operator_run	Counter	N/A	The total of the running operator.
msdpoperator_diskFree Less5GBEngines_total	Gauge	msdpscalout_ns	The checked number of the engines which have free spaces lower than 5GB.
msdpoperator_diskFree MiBytesInEngine	Gauge	msdpscalout_ns	The free space of current engine in MiBytes.
msdpoperator_diskFreeLess 10GBClusters_total	Gauge	msdpscalout_ns	The checked number of the msdpscaleout applications that have free spaces lower than 10GB.
msdpoperator_totalDiskFree PercentInCluster	Gauge	msdpscalout_ns	The percent of the msdpscaleout applications that have free spaces. For example, 0.95 means 95%
msdpoperator_diskFree PercentInEngine	Gauge	msdpscalout_ns	The percent of the current engines, which have free spaces.
msdpoperator_pvcFree PercentInCluster	Gauge	msdpscalout_ns, component	The percent of the used PVC, which have free spaces.
msdpoperator_unhealthy Engines_total	Gauge	msdpscalout_ns	The total of unhealthy engines.
msdopoerator_createdPods_total	Gauge	msdpscalout_ns, component	The total of created msdpscaleout pods.

Monitoring with Azure Container insights

You can use Azure Container insights to collect Prometheus metrics to monitor pods in MSDP Scaleout.

To configure Azure Container insights

1. Enable Azure Container insights.

   See Azure documentation.

2. Download the template ConfigMap YAML file and save it as `container-azm-ms-agentconfig.yaml`.

3. Add the YAML file with the following sample configuration:

```
prometheus-data-collection-settings: |-

[prometheus_data_collection_settings.cluster]
interval = "1m"

fieldpass = ["msdpoperator_reconcile_total",
             "msdpoperator_reconcile_failed",
             "msdpoperator_operator_run",
             "msdpoperator_diskFreeLess5GBEngines_total",
             "msdpoperator_diskFreeMiBytesInEngine",
             "msdpoperator_diskFreeLess10GBClusters_total",
             "msdpoperator_totalDiskFreePercentInCluster",
             "msdpoperator_diskFreePercentInEngine",
             "msdpoperator_pvcFreePercentInCluster",
             "msdpoperator_unhealthyEngines_total",
             "msdpoperator_createdPods_total"]

monitor_kubernetes_pods = true

# Add the namespace of MSDP operator in the follow list.
It's "msdp-operator-system" by default.

monitor_kubernetes_pods_namespaces =
["msdp-operator-system"]
```

   Table 10-2 lists the Prometheus metrics that MSDP Scaleout supports.

4 Apply the ConfigMap.

```
kubectl apply -f container-azm-ms-agentconfig.yaml
```

The configuration change takes a few minutes and all omsagent pods in the cluster restart.

The default namespace of prometheus metrics is **prometheus**.

5 Add alert rules for the integrated metrics.

Add related log query, add new alert rule for the selected query, and alert group/action for it.

For example,

If the free space size of the MSDP Scaleout engines is lower than 1 GB in past 5 minutes, alert the users.

Log query:

```
InsightsMetrics

| where Name == "msdpoperator_diskFreeMiBytesInEngine"
| where Namespace == "prometheus"
| where TimeGenerated > ago(5m)
| where Val <= 1000000
| where Val > 0
```

If multiple MSDP Scaleouts are deployed in the same AKS cluster, use the filter to search the results. For example, search the MSDP engines with the free space size lower than 1GB in the namespace **sample-cr-namespace**

Log query:

```
InsightsMetrics
| where Name == "msdpoperator_diskFreeMiBytesInEngine"
| where Namespace == "prometheus"
| where TimeGenerated > ago(5m)
| where Val <= 1000000
| where Val > 0
| extend Tags = parse_json(Tags)
| where Tags.msdpscalout_ns == "sample-cr-namespace"
```

MSDP Scaleout supports the following Prometheus metrics:

Table 10-2 Supported Prometheus metrics list in MSDP Scaleout

Metrics	Type	Filters	Description
msdpoperator_reconcile_total	Counter	N/A	The total of the reconcile loops msdp-operator run.
msdpoperator_reconcile_failed	Counter	N/A	The total of the reconcile loops msdp-operator failed to run.
msdpoperator_operator_run	Counter	N/A	The total of the running operator.
msdpoperator_diskFreeLess 5GBEngines_total	Gauge	InsightsMetrics.Tags.msdpscalout_ns	The checked number of the engines which have free spaces lower than 5GB.
msdpoperator_diskFreeMi BytesInEngine	Gauge	InsightsMetrics.Tags.msdpscalout_ns	The free space of current engine in MiBytes.
msdpoperator_diskFreeLess 10GBClusters_total	Gauge	InsightsMetrics.Tags.msdpscalout_ns	The checked number of the msdpscaleout apps which have free spaces lower than 10GB.
msdpoperator_totalDiskFree PercentInCluster	Gauge	InsightsMetrics.Tags.msdpscalout_ns	The percent of the msdpscaleout apps that have free spaces. For example, 0.95 means 95%
msdpoperator_diskFree PercentInEngine	Gauge	InsightsMetrics.Tags.msdpscalout_ns	The percent of the current engines, which have free spaces.
msdpoperator_pvcFree PercentInCluster	Gauge	InsightsMetrics.Tags.msdpscalout_ns, InsightsMetrics.Tags.component	The percent of the used PVC, which have free spaces.
msdpoperator_unhealthy Engines_total	Gauge	InsightsMetrics.Tags.msdpscalout_ns	The total of unhealthy engines.
msdpoperator_createdPods _total	Gauge	InsightsMetrics.Tags.msdpscalout_ns, InsightsMetrics.Tags.component	The total of created msdpscaleout pods.

The Kubernetes resources for MSDP Scaleout and MSDP operator

Do not change or delete the Kubernetes resources that MSDP deployment has created.

- Run the following command to find all the namespaced resources:

```
kubectl api-resources --verbs=list --namespaced=true -o name |
xargs -n 1 -i bash -c 'if ! echo {} |grep -q events; then kubectl
get --show-kind --show-labels --ignore-not-found -n <cr or operator
namespace> {}; fi'
```

- Run the following command to find commonly used namespace resources:

  ```
  kubectl get pod,svc,deploy,rs,pvc -n <cr or operator namespace>
  -o wide
  ```

- Run the following command to find the Kubernetes cluster level resources that belong to the CR:

  ```
  kubectl api-resources --verbs=list --namespaced=false -o name |
  xargs -n 1 -i bash -c 'kubectl get --show-kind --show-labels
  --ignore-not-found {} |grep [msdp-operator|<cr-name>]'
  ```

Chapter 11

Monitoring Health Checker

This chapter includes the following topics:

Overview

The **Health Checker** is a diagnostic capability integrated into the `kubectl-cloudscale` plugin. It executes a collection of validation checks to ensure that Cloud Scale Kubernetes infrastructure components are properly installed, reachable, and correctly configured. This tool helps both support teams and users quickly identify root causes during troubleshooting.

When to run health checks:

- Before deployment (pre-deployment)
- After deployment (post-deployment)
- Before performing upgrades (pre-upgrade)

You can execute all checks, target a specific category, or run individual checks by name depending on your requirements.

Prerequisites:

- Installed `kubectl` and `Helm charts`.
- A container runtime such as Podman or Docker.
- Access to a functional Kubernetes cluster.

Command:

Use the following command to run the health checks:

```
kubectl cloudscale health-check
```

You can run the health check command in multiple ways. For more information on the available options, See "Command options" on page 195. section.

Cloud Scale version compatibility

The Health Checker validates Cloud Scale deployments across different lifecycle phases. Version compatibility requirements vary based on when the checks are run.

- **Pre-deployment**: Supported for environments preparing to deploy Cloud Scale version 11.2 or later.
- **Post-deployment**: Requires Cloud Scale version 11.2 or later.
- **Pre-upgrade**: Requires Cloud Scale 10.5 or later.

When you run specific tests using the `-t/--test` option or test categories using the `-c/--test-category` option, each selection is validated for version compatibility before execution. If a selected test is incompatible with the current Cloud Scale version, the Health Checker reports an error and skips execution of that test.

Some health checks are platform-specific (for example, Amazon EKS or Azure AKS). If a check does not apply to the detected platform, it is automatically skipped.

Command options

The health checks can be performed by using the following command:

```
kubectl cloudscale health-check
```

You can run the health checker command using multiple options. The following table describes each available option and its function.

Table 11-1 Health checker command options

Command options	Function	Notes
`--list-all` Short form: `-l`	Lists all available health checks and the categories they belong to.	Discover which checks exist before running.

Table 11-1 Health checker command options *(continued)*

Command options	Function	Notes
`--test` Short form: `-t`	Runs health checks by name. Accepts a single name, a comma-separated list (for example, `test1`, `test2`), or all to execute every check.	Run specific checks or all checks by name. Use `-t -h` or `-t help` for detailed help.
`--test-category` Short form: `-c`	Runs all checks in a category: ■ Before deployment: `pre-deployment` ■ After deployment: `post-deployment` ■ Before performing upgrade: `pre-upgrade`	Run a full set of checks for a given phase. Use `-c -h` or `-c help` for detailed help.
`--namespace` Short form: `-n`	Specifies the Kubernetes namespace where health checks run.	■ `-t`: Required except when running pre-deployment tests only. ■ `-c`: Required for post-deployment and pre-upgrade; optional for pre-deployment. ■ `-n -h` or `-n help`: Use these options for detailed help.
`--help` Short form: `-h`	Shows help for the `health-check` command or, when used with a flag value (for example, `-t -h`), help for that specific flag.	Get usage instructions and examples.

Note: The `-t` and `-c` command options cannot be used together.

Example for health checker command options

The following table lists the examples for the various functions of the command options.

Table 11-2 Example for health checker command options

Functions	Short form of the command	Long form of the command
Help (main)	`kubectl-cloudscale health-check -h`	`kubectl-cloudscale health-check --help`
List all checks and categories	`kubectl-cloudscale health-check -l`	`kubectl-cloudscale health-check --list-all`
Test flag help	`kubectl-cloudscale health-check -t -h`	`kubectl-cloudscale health-check --test help`
Category flag help	`kubectl-cloudscale health-check -c -h`	`kubectl-cloudscale health-check --test-category help`
Namespace flag help	`kubectl-cloudscale health-check -n -h`	`kubectl-cloudscale health-check --namespace help`
Single test by name	`kubectl-cloudscale health-check -t SecretsValidation -n netbackup`	`kubectl-cloudscale health-check --test SecretsValidation --namespace netbackup`
Multiple tests (comma-separated)	`kubectl-cloudscale health-check -t SecretsValidation,ServiceImageTagCheck -n netbackup`	`kubectl-cloudscale health-check --test SecretsValidation,ServiceImageTagCheck --namespace netbackup`
All tests	`kubectl-cloudscale health-check -t all -n netbackup`	`kubectl-cloudscale health-check --test all --namespace netbackup`
By category (pre-deployment)	`kubectl-cloudscale health-check -c pre-deployment`	`kubectl-cloudscale health-check --test-category pre-deployment`
By category (post-deployment)	`kubectl-cloudscale health-check -c post-deployment -n netbackup`	`kubectl-cloudscale health-check --test-category post-deployment --namespace netbackup`

Table 11-2 Example for health checker command options *(continued)*

Functions	Short form of the command	Long form of the command
By category (pre-upgrade)	`kubectl-cloudscale health-check -c pre-upgrade -n netbackup`	`kubectl-cloudscale health-check --test-category pre-upgrade --namespace netbackup`

Note: Some health checks require user input (yes/no or text) and must be run in an interactive terminal.

Interpreting health check results

Each health check returns one of the following statuses:

- **OKAY**: The check passed. The component or configuration meets the required criteria.
- **NOT_OKAY**: The check failed. Review the error message and logs for detailed diagnostic information.
- **SKIPPED**: The check was not run because it was not applicable (for example, an EKS-specific check on an AKS cluster).

When multiple tests or an entire test category is executed, the Health Checker prints a summary at the end of the run that includes:

- Total number of checks
- Passed, failed, and skipped counts
- Total execution duration

Detailed logs for each run are written to the session log path, which is displayed at the end of the execution.

Note the following:

- Each health check has a default timeout of 5 minutes (300 seconds).
- For checks that require interactive input (yes/no responses or text prompts), there is no timeout while waiting for user input.
- Once user input is provided, the underlying test logic remains subject to the 5-minute timeout.

Chapter 12

Managing NetBackup

This chapter includes the following topics:

Managing NetBackup deployment using VxUpdate

VxUpdate package is not shipped with the NetBackup deployment package. You must download and add it in NetBackup primary server.

To manage NetBackup deployment using VxUpdate

1. Download the required VxUpdate package on the docker-host used to interact with Kubernetes cluster.
2. Copy the VxUpdate package to primary server pod using the `kubectl cp <path-vxupdate.sja> <primaryServerNamespace>/<primaryServer-pod-name>:<path-on-primary-pod>` command.
3. After VxUpdate package is available on primary server Pod, add it to NetBackup repository using any one of the following:
   - Select the VxUpdate package from the NetBackup Web UI.
   - Execute the following command in the primary server pod:

     ```
     kubectl exec -it -n <primaryserver-namespace>
     <primaryServer-pod-name> -- bash
     ```

     And run the following command:

     ```
     nbrepo -a <vxupdate-package-path-on-PrimaryPod>
     ```

After adding the VxUpdate package to `nbrepo`, this package is persisted even after pod restarts.

Updating the Primary/Media server CRs

After the successful deployment of the primary server and media server CRs, you can update the values of only selected specs by editing the respective environment custom resource.

Note: Updating the Kubernetes resources (pod, configmap, services, statefulset etc) created for the CRs is not recommended.

Following tables describe the specs that can be edited for each CR.

Table 12-1 Primary server CR

Spec	Description
paused	Specify *True* or *False* as a value, to temporarily stop the respective CR controller. *True*: Stop the controller. *False*: Resume the controller.
configCheckMode	Specify *default*, *dryrun* or *skip* as a value. See the section called "Config-Checker execution and status details" on page 46.
(*AKS-specific*) capacity	Catalog, log and data volume storage capacity can be updated.

Table 12-2 Media server CR

Spec	Description
paused	Specify *True* or *False* as a value, to temporarily stop the respective CR controller. *True*: Stop the controller. *False*: Resume the controller.

Table 12-2 Media server CR *(continued)*

Spec	Description
replicas	Specifies the maximum number of replicas that the media server can scale up to. **Note:** It is recommended not to reduce the number of maximum replicas. To reduce the maximum number of replicas, perform the media server decommissioning steps mentioned in References to nonexistent or decommissioned media servers remain in NetBackup.
minimumReplicas	Describes the minimum number of replicas of the media server running. This is an optional field. If not specified, the value for this field will be set to the default value of 1.
configCheckMode	Specify *default*, *dryrun* or *skip* as a value. See the section called “Config-Checker execution and status details” on page 46.
(*AKS-specific*) capacity	Catalog, log and data volume storage capacity can be updated.

If you edit any other fields, the deployment can go into an inconsistent state.

Additional steps

- Delete the Load Balancer service created for the media server by running the following commands:

  ```
  $ kubectl get service --namespace <namespce_name>
  $ kubectl delete service <service-name> --namespace <namespce_name>
  ```

- Identify and delete any outstanding persistent volume claims for the media server by running the following commands:

  ```
  $ kubectl get pvc --namespace <namespce_name>
  $ kubectl delete pvc <pvc-name>
  ```

- Locate and delete any persistent volumes created for the media server by running the following commands:

  ```
  $ kubectl get pv
  $ kubectl delete pv <pv-name> --grace-period=0 --force
  ```

Migrating the cloud node for primary or media servers

You can migrate the following respective cloud node for primary or media servers:

- (*AKS-specific*) node pool
- (*EKS-specific*) node group

To migrate the cloud node for primary or media servers

1. Run the following command:

   ```
   helm upgrade cloudscale cloudscale.tgz -n
   <namespace>--reuse-values --set
   environment.primary.nodeSelector.labelKey=<NewKey> --set
   environment.primary.nodeSelector.labelValue=<NewKey>
   ```

2. Change the node selector **labelKey** and **lableValue** to new values for primary/media server.
3. Save the environment CR.

This will change the statefulset for respective NetBackup server replica to 0 for respective server. This will terminate the pods. After successful migration, statefulset replicas will be set to original value.

Note: If you migrate any component (Primary Server, Media Server, MSDP, or cpdata) to a new node pool with different labels, ensure that you update the corresponding **key** and **value** fields in the tolerations section of the fluentbit chart/YAML file. For more information, refer to the 'Fluentbit DaemonSet configuration variables' table in the following section:

See "Parameters for logging (fluentbit)" on page 114.

Migrating cpServer controlPlane node

This section describes the steps for cpServer controlPlane node migration.

Note: It is recommended to use the same node pool for primary and cpServer controlPlane.

Steps for cpServer controlPlane node migration

1. Create new **nbunewpool** node pool with required data (with appropriate taints and labels).

   Note: Ensure that the new node pool uses the same subnets or compatible subnets in the same Availability Zones as the original deployment.

   ---

2. Validate subnet compatibility and network configuration:

   For AKS:

   - Verify Azure Files/NetApp Files availability across all target zones.
   - Confirm new node pool subnet and zone compatibility with existing storage configuration.
   - Check that virtual network peering and routing support the new node placement.

   For EKS:

   - Verify that AWS Load Balancer Controller and CNI are configured for the target subnets.
   - Confirm EFS mount targets exist in all Availability Zones where the new node pool will be deployed.
   - Check that target subnets are properly tagged for Kubernetes cluster discovery.
   - If EFS mount targets are missing in target AZs, create them before proceeding with migration.
   - Ensure that the security groups allow NFS traffic (port 2049) between nodes and EFS mount targets.

3. Set the required permissions and assign the role.

   - For AKS: For plugin configuration, the kubernetes cluster requires permissions to be assigned to the System Managed Identity as follows:
     - Obtain the name of the infrastructure resource group for the kubernetes cluster.
     - Enable the System Managed Identity on the identified nodepool (`nbunewpool`).
     - Assign the role having the Snapshot Manager permission.

- For EKS: For plugin configuration, the kubernetes cluster requires permissions to be assigned to IAM of primary nodepool. The IAM role must be assigned during the node pool creation as follows:
  - Assign the appropriate IAM role to the **nbunewpool** nodepool.

4 Configure Instance Metadata Service (IMDS) for AL2023 compatibility:

Critical for Amazon Linux 2023: AL2023 AMI for EKS nodes enforces `IMDSv2 hop limit = 1` by default, which prevents NetBackup containers from accessing AWS metadata service.

For more information, see Upgrade from Amazon Linux 2 to Amazon Linux 2023.

Note: If you migrate any component (Primary Server, Media Server, MSDP, or cpdata) to a new node pool with different labels, ensure that you update the corresponding **key** and **value** fields in the tolerations section of the fluentbit chart/YAML file. For more information, refer to the 'Fluentbit DaemonSet configuration variables' table in the following section:

See “Parameters for logging (fluentbit)” on page 114.

Chapter 13

Managing the Load Balancer service

This chapter includes the following topics:

About the Load Balancer service

Key features of the Load Balancer service:

Note: Load Balancer services are not created for media servers from NetBackup 10.5 release. If the setup is upgraded from a pre 10.4 release, the load balancer services are available post upgrade. It is recommended to clear the Load Balancer services. Refer to the following section:

See "Steps for upgrading Cloud Scale from multiple media load balancer to none" on page 211.

- Load balancer services are created in primary server deployment that allows you to access the NetBackup application from public domains.
- In the primary server CR spec, networkLoadBalancer section is used for handling the IP address and DNS name allocation for load balancer services. If **ipList** is provided in CR spec, IP address count must not be less than the count specified in **replica** for media server and for primary server, only one IP address must be mentioned.

- The networkLoadBalancer section can be used to provide static IP address and DNS name allocation to the Load Balancer services.
- (***AKS-specific***)
  - The networkLoadBalancer section can be used to provide static IP address and DNS name allocation to the loadbalancer services. For more information to create and use static loadbalancer, see Microsoft documentation.
    Static IP addresses and FQDN if used must be created before being used. Refer the below section:
    - Pre-allocation of static IP address and FQDN from resource group
      In this case, it is required to provide the network resource group in annotations. This resource group is the resource group of load balancer public IPs that are in the same resource group as the cluster infrastructure (node resource group). This static FQDN and IP address must be valid in case of pod failure or upgrades scenarios as well.
      In case user wants to use public load balancer, add **type: Public** in networkLoadBalancer section in primary and media server section in environment CR.
      - Example: In primary CR,

```
networkLoadBalancer:
  type: Public
  annotations:
  - service.beta.kubernetes.io/azure-load-balancer-
resource-group:<name of network resource-group>
  ipList:
  - fqdn: primary.eastus.cloudapp.azure.com
    ipAddr: 40.123.45.123
```

- (***EKS-specific***)
  - NetBackup supports the network load balancer with AWS Load Balancer scheme as **internet-facing**.
  - FQDN must be created before being used. Refer below sections for different allowed annotations to be used in CR spec.
  - User must add the following annotations:
    `service.beta.kubernetes.io/aws-load-balancer-subnets: <subnet1 name>`
    In addition to the above annotations, if required user can add more annotations supported by AWS. For more information, see AWS Load Balancer Controller Annotations.

For example:
CR spec in primary server,

```
networkLoadBalancer:
type: Private
annotations:
  service.beta.kubernetes.io/aws-load-balancer-subnets: <subnet1
 name>
  ipList:
  "10.244.33.27: abc.vxindia.cohesity.com"
```

The IP address, the subnet provided in ipList and annotations in networkLoadBalancer section in CR spec must belong to same availability zone that of the node group.

Note: The subnet provided here should be same as the one given in node pool used for primary server and media server.

If NetBackup client is outside VPC or to access Web UI from outside VPC, then client CIDR must be added with all NetBackup ports in security group rule of cluster. Run the following command, to obtain the cluster security group:

```
aws eks describe-cluster --name <my-cluster> --query
cluster.resourcesVpcConfig.clusterSecurityGroupId
```

For more information on cluster security group, see Amazon EKS security group requirements and considerations.

Add inbound rule to security group. For more information, see Add rules to a security group.

(*AKS-specific*) Preferred annotations

Table 13-1 Preferred annotations

Annotations	Value	Description
service.beta.kubernetes.io/ azure-load-balancer- internal	true or false	Specify whether the load balancer should be internal. Added by default when type is selected as **Private** in load balancer service annotations.

Table 13-1 Preferred annotations *(continued)*

Annotations	Value	Description
service.beta.kubernetes.io/ azure-load-balancer-internal-subnet	Name of the subnet	Specify which subnet the internal load balancer should be bound to.
service.beta.kubernetes.io/ azure-load-balancer -resource-group	Name of the resource group	Specify the resource group of load balancer public IPs that are not in the same resource group as the cluster infrastructure (node resource group).

Default ports used in the Load Balancer service

- Primary server:
  - 8443
    Used to access NetBackup Web UI from a machine which is not deployed in the same VPC as your EKS cluster.
    In this case, open the Primary Load Balancer’s Inbound security group to the machine’s IP/CIDR with TCP port 443 and 8443.
    For more information, refer to the "Update the security groups for your Network Load Balancer" section in the *AWS documentation*.
  - 1556
    Used as bidirectional port. Primary server to/from media servers and primary server to/from client require this TCP port for communication.
  - 443
    Used to inbound to vnet proxy tunnel on the primary server. Also, this is used Nutanix workload, communication from primary server to the deduplication media server.
  - 13781
    The MQBroker is listening on TCP port 13781. NetBackup client hosts - typically located behind a NAT gateway - be able to connect to the message queue broker (MQBroker) on the primary server.
  - 13724
    This port is required as a fall-back option when a legacy service cannot be reached through PBX and is required when using the Resilient Network feature.
- Snapshot Manager server:
  - 443

The Snapshot Manager user interface uses this port as the default HTTPS port.

- 5671
  The Snapshot Manager RabbitMQ server uses this port for internal service communications. This port must be open to support multiple agents, extensions, backup from snapshot, and restore from backup jobs.
- (*EKS-specific*) 2049
  It is used for Amazon EFS access.
  For more information, see Source ports for working with EFS.

  Note: Add the NFS rule that allows traffic on port 2049 directly to the cluster security group. The security group attached to EFS must also allow traffic from port 2049.

Notes for Load Balancer service

Note the following points:

- After deployment of primary server or media server, updating the DNS name, IP address and subnet through CR is not allowed.
- If mistakenly user has added wrong values:
  - User wants to update IP address and subnet, you must delete the CR and update the CR yaml and reapply it.
  - User wants to update the DNS name, you must delete the respective CR and delete the respective PVC and PV as well.

  Note: Be cautious while performing this step, this may lead to data loss.

- Before using the DNS and its respective IP address in CR yaml, you can verify the IP address and its DNS resolution using nslookup.
- If nslookup is done for loadbalancer IP inside the container, it returns the DNS in the form of **<svc name>.<namespace_name>.svc.cluster.local**. This is Kubernetes behavior. Outside the pod, the loadbalancer service IP address is resolved to the configured DNS. The `nbbptestconnection` command inside the pods can provide a mismatch in DNS names, which can be ignored.

Opening the ports from the Load Balancer service

In this deployment, most of the required ports are already opened from the NetBackup primary and media server load balancer services by default.

- If you want to use a specific workload and that needs specific ports, you must add those ports in the port specification of the load balancer service.
- In case of media server, you must add custom ports in the load balancer service of all the replicas. In case of scaling up the media server, user needs to explicitly add newly added custom ports in respective newly created load balancer services.
- In case custom ports are added in the load balancer service and the same load balancer service is deleted or created again, you must add respective custom ports again in the load balancer service specification.

For all three scenarios, perform the steps given in this section.

To open the ports from the Load Balancer service

1. Run the `kubectl get service -n <namespace>` command.

   This command lists all the services available in given namespace.

2. Edit the required primary load balancer service using `kubectl edit service <service-name> -n <namespace>` command.

   For example:

   - For primary server load balancer service:
     - Service name starts with **Name** of primary server like **<Name>-primary**. Edit the service with the `kubectl edit service <Name>-primary -n <namespace>` command.

   Note: The load balancer service with name **Name** used in primary sever and media server specification must be unique.

   ---

3 Add entry for new port in ports array in specification field of the service. For example, if user want to add 111 port, then add the following entry in ports array in specification field.

```
name: custom-111

    port: 111

    protocol: TCP

    targetPort: 111
```

4 Save the changes.

The service is updated and the new port is listed in ports list of the respective service when you run the `kubectl get service -n <namespace>` command.

Steps for upgrading Cloud Scale from multiple media load balancer to none

For version 10.5 and later, there is no need of load balancers for media servers. this section describes the post upgrade procedure to be performed for converting from multiple load balancers to none.

Steps to convert from multiple load balancers to none

1 After successfully upgrading the Cloud Scale Technology to 10.5 or later, all the load balancers associated with the media server are upgraded to version 10.5 or later.

2 Deactivate the policy and the schedules. Wait for the running jobs to complete.

3 Add the `media2` object in the mediaServers without te network load balancers.

4 Execute the following command:

```
helm get values cloudscale -n <namespace>
```

Note the details of the mediaserver section.

Then, add another mediaserver object with a different name, but with identical details as the first one:

```
helm upgrade cloudscale
VRTSk8s-netbackup-11.2.0.2/helm/cloudscale-11.2.0.2.tgz -n
<namespace> --reuse-values \
--set environment.mediaServers[0].name=media1 --set
environment.mediaServers[0].nodeSelector.labelKey=<original value>
 \
--set
environment.mediaServers[0].nodeSelector.labelValue=<original
value> --set environment.mediaServers[0].replicas=<original value>
 \
--set environment.mediaServers[0].minimumReplicas=<original value>
 --set environment.mediaServers[0].storage.data.capacity=<original
 value> \
--set environment.mediaServers[0].tag=11.2 --set
environment.mediaServers[0].storage.log.capacity=<original value>
 \
--set
environment.mediaServers[0].storage.data.storageClassName=<sc-name>
 \
--set
environment.mediaServers[0].storage.log.storageClassName=<sc-name>
 \
--set environment.mediaServers[1].name=media2 --set
environment.mediaServers[1].nodeSelector.labelKey=agentpool \
--set
environment.mediaServers[1].nodeSelector.labelValue=mediapool
--set environment.mediaServers[1].replicas=2 \
--set environment.mediaServers[1].minimumReplicas=2 --set
environment.mediaServers[1].storage.data.capacity=60Gi \
--set
environment.mediaServers[1].storage.data.storageClassName=<sc-name>
 \
--set environment.mediaServers[1].tag=11.2 --set
environment.mediaServers[1].storage.log.capacity=10Gi \
--set
environment.mediaServers[1].storage.log.storageClassName=<sc-name>
```

5. Once the `media2` is successfully added, check the status of the media servers, media2 pods and services. Wait for the all the Pods to come up with media server status as **success**.

   ```
   Get mediaservers
   Kubectl get mediaserver -n <namespace>
   media1 11.0.x-xx 79m nbux-xx-xxx-xx-xxx.vxindia.cohesity.com Success
   media2 11.0.x-xx 79m nbux-xx-xxx-xx-xxx.vxindia.cohesity.com Success
   ```

6. Ensure that all the user settings that were present for media1 pods are added manually to media2 pods also. For example, LogLevel, FIPS Mode, DNAT and so on.

7 Pause both the mediaserver objects using the following command:

```
helm upgrade cloudscale
VRTSk8s-netbackup-11.2/helm/cloudscale-11.2.tgz -n <namespace>
--reuse-values \

--set environment.mediaServers[0].name=media1 --set
environment.mediaServers[0].nodeSelector.labelKey=<original value>
\

--set
environment.mediaServers[0].nodeSelector.labelValue=<original
value>

--set environment.mediaServers[0].replicas=<original value> \

--set environment.mediaServers[0].minimumReplicas=<original value>

--set environment.mediaServers[0].storage.data.capacity=<original
value> \

--set
environment.mediaServers[0].storage.data.storageClassName=<sc-name>
\

--set environment.mediaServers[0].paused=true \

--set environment.mediaServers[0].tag=11.2

--set environment.mediaServers[0].storage.log.capacity=<original
value> \

--set
environment.mediaServers[0].storage.log.storageClassName=<sc-name>
\

--set environment.mediaServers[1].name=media2

--set environment.mediaServers[1].nodeSelector.labelKey=<original
value> \

--set
environment.mediaServers[1].nodeSelector.labelValue=mediapool

--set environment.mediaServers[1].replicas=<original value> \

--set environment.mediaServers[1].minimumReplicas=<original value>

--set environment.mediaServers[1].storage.data.capacity=<original
value> \
```

```
--set
environment.mediaServers[1].storage.data.storageClassName=<sc-name>
\

--set environment.mediaServers[1].tag=11.2

--set environment.mediaServers[1].storage.log.capacity=<original
value> \

--set
environment.mediaServers[1].storage.log.storageClassName=<sc-name>
\

--set environment.mediaServers[1].paused=true
```

8. Navigate to NetBackup web UI and select **Storage**. On the **Disk Storage**, select the **Storage server.** There are multiple entries for **media server** with and without the load balancers.
9. Delete the entry for `media1`. You have to remove all the entries of media servers containing FQDN.

10 Remove the first mediaserver object from the environment using the following command (change the index first mediaServer to index 0 mediaServer):

```
helm upgrade cloudscale
VRTSk8s-netbackup-11.2/helm/cloudscale-11.2.tgz -n <namespace>
--reuse-values \

--set environment.mediaServers[0].name=media2

--set environment.mediaServers[0].nodeSelector.labelKey=agentpool
\

--set
environment.mediaServers[0].nodeSelector.labelValue=mediapool

--set environment.mediaServers[0].replicas=2 \

--set environment.mediaServers[0].minimumReplicas=2

--set environment.mediaServers[0].storage.data.capacity=60Gi \

--set
environment.mediaServers[0].storage.data.storageClassName=<sc-name>
\

--set environment.mediaServers[0].tag=11.2

--set environment.mediaServers[0].storage.log.capacity=10Gi \

--set
environment.mediaServers[0].storage.log.storageClassName=<sc-name>
\

--set environment.mediaServers[0].paused=true
```

11 Check the `bp.conf` file.

12 Delete the `media1` entries (FQDN entries of media1) from the `bp.conf` file. Exec into the primary pod and open `/usr/openv/netbackup/bp.conf` and delete the entries and save the file.

13 Resume the `media2` using the following command:

```
helm upgrade cloudscale
VRTSk8s-netbackup-11.2/helm/cloudscale-11.2.tgz -n <namespace>
--reuse-values \

--set environment.mediaServers[0].name=media2

--set environment.mediaServers[0].nodeSelector.labelKey=agentpool
\ --set
environment.mediaServers[0].nodeSelector.labelValue=mediapool

--set environment.mediaServers[0].replicas=2 \

--set environment.mediaServers[0].minimumReplicas=2

--set environment.mediaServers[0].storage.data.capacity=60Gi \

--set
environment.mediaServers[0].storage.data.storageClassName=<sc-name>
\

--set environment.mediaServers[0].tag=11.2

--set environment.mediaServers[0].storage.log.capacity=10Gi \

--set
environment.mediaServers[0].storage.log.storageClassName=<sc-name>
\

--set environment.mediaServers[0].paused=false
```

14 Note the **minimum replica** value of `media2`. Change the minimum replica value using the following command:

```
kubectl get pvc -n nbu | grep -i data-media1-media | wc -l 2
```

Wait for the pod to be in running state and the status of the media server must be displayed as `success`.

Note: Although **minimum replica** value of `media server = 0` is supported in cloud scale, for PSF workloads (for example, MongoDB) at least one elastic media server must be running (that is, minimum replica of media server >= 1). The name of media server can be obtained from **Host mappings**

For elastic media server, this name is same as the media server pods.

---

15 Login into the primary server Pod and move the database from previous media server to new media server using the following command:

```
bpmedia -movedb -allvolumes -oldserver <old mediaserver name>
-newserver <new mediaserver name>
```

```
example: bpmedia -movedb -allvolumes -oldserver
nbux-xx-xxx-xx-xxx.vxindia.cohesity.com -newserver media2-media-0
```

Repeat this step for other media servers.

16 Using the command delete the alias or hostname and repeat this for each media server:

```
nbemmcmd -deletehost -machinename <old mediaserver name>
-machinetype media
```

```
example: nbemmcmd -deletehost -machinename
nbux-xx-xxx-xx-xxx.vxindia.cohesity.com -machinetype media
```

17 Modify the minimum replica value to the original value noted in step 13.

18 Add the entry of the previous media server object alias to the new media server in the **Host mappings**.

Note: When migrating from multiple load balancers to no load balancers in NetBackup version 11.0 or later, delete the stale host mapping entries as follows:

Navigate to, NetBackup Web UI ==> Security==> Host mappings ==> Select the **Actions** menu next to the primary host name ==> Manage mappings ==> Delete the old load balancer media name mapping and click on Save.

---

19 Restore the backup and check the status in the **Activity monitor** and to the location where the backup is restored.

Managing PostrgreSQL DBaaS

This chapter includes the following topics:

- Changing database server password in DBaaS
- Updating database certificate in DBaaS

Changing database server password in DBaaS

Note: When setting the PostgreSQL password in DBaaS, ensure that the password does not contain the following special characters:

equal (=), double quote ("), single quote ('), percentage (%), at sign (@), ampersand (&), question mark (?), underscore (_), and hash (#)

Azure-specific

1 Launch an Azure CLI pod into the AKS cluster using the following command:

```
$ kubectl run az-cli --image=mcr.microsoft.com/azure-cli:2.53.0
--command sleep infinity
```

Note: Access to Azure Key Vault is restricted to specific subnets. Passwords stored in Azure Key Vault can be easily updated from a pod running in AKS.

Connecting to Postgres database using Azure requires installing rdbms-connections. This functionality is applicable to azure-cli 2.53.0.

---

2 Exec into the Azure CLI pod as follows:

```
$ kubectl exec -it az-cli -- /bin/ash
```

3 From Azure CLI pod, log into Azure account:

```
$ az login --scope https://graph.microsoft.com//.default
```

4 (Optional) Create a key vault policy to allow the current user to retrieve the database credential.

Obtain the name of your resource group, key vault and ID of the current user by using the following respective commands:

- Resource group name:
  ```
  $ RESOURCE_GROUP=<resource_group_name>
  ```
- Key vault name:
  ```
  $ KEY_VAULT_NAME=$(az keyvault list --resource-group
  $RESOURCE_GROUP --resource-type vault | jq -r '.[].name')
  ```
- Current user ID name:
  ```
  $ USER_ID=$(az account show | jq -r '.user.name')
  ```

Create a key vault access policy as follows:

```
$ az keyvault set-policy -n $KEY_VAULT_NAME --upn $USER_ID
--resource-group $RESOURCE_GROUP --secret-permissions all
```

5 Obtain the login name for the key vault (DBADMINUSER):

```
$ DBADMINUSER=$(az keyvault secret show --vault-name
$KEY_VAULT_NAME --name dbadminlogin | jq -r .value)
```

6 Obtain the password for the key vault (OLD_DBADMINPASSWORD):

```
$ OLD_DBADMINPASSWORD=$(az keyvault secret show --vault-name
$KEY_VAULT_NAME --name dbadminpassword | jq -r .value)
```

7 Obtain the server name (DBSERVER):

```
DBSERVER=$(az postgres flexible-server list --resource-group
$RESOURCE_GROUP | jq -r '.[].name')
```

8 (*Optional*) Verify the current password encryption method by using the following command:

```
az postgres flexible-server execute -p "$OLD_DBADMINPASSWORD" -u
$DBADMINUSER -n $DBSERVER -d postgres -q "SELECT * from
azure_roles_authtype();" -o table
```

Following message is displayed:

```
Successfully connected to twilk-db.
Ran Database Query: 'SELECT * from azure_roles_authtype();'
Retrieving first 30 rows of query output, if applicable.
Closed the connection to twilk-db
Authtype    Rolename
----------  -------------------------
NOLOGIN     azuresu
NOLOGIN     pg_database_owner
NOLOGIN     pg_read_all_data
NOLOGIN     pg_write_all_data
NOLOGIN     pg_monitor
NOLOGIN     pg_read_all_settings
NOLOGIN     pg_read_all_stats
NOLOGIN     pg_stat_scan_tables
NOLOGIN     pg_read_server_files
NOLOGIN     pg_write_server_files
NOLOGIN     pg_execute_server_program
NOLOGIN     pg_signal_backend
NOLOGIN     azure_pg_admin
NOLOGIN     replication
MD5         nbdbadmin
```

To install `rdbms-connect` extension, ensure that you select the *Y* option. If installing the extension fails in the `az-cli` container, then some dependencies must be missing. Install the missing dependencies with `apk add gcc musl-dev` command and try again.

The rdbms-connect utility can also be installed using the `az extension add --name rdbms-connect` command.

The nbdbadmin auth type must be **SCRAM-256**. Resetting the password as follows will re-encrypt the password correctly.

9 Set the new password as follows:

Before setting the new password ensure that you know your database server name or obtain it by using the following command:

```
NEW_DBADMINPASSWORD="<new_password>"
```

Use the following command to set the new password:

```
az postgres flexible-server execute -p $OLD_DBADMINPASSWORD -u
$DBADMINUSER -n $DBSERVER -d postgres -q "ALTER USER\"nbdbadmin\"
WITH PASSWORD '$NEW_DBADMINPASSWORD';"
```

Or

If you are only trying to re-encrypt the current password without changing it, use the following command:

```
az postgres flexible-server execute -p $OLD_DBADMINPASSWORD -u
$DBADMINUSER -n $DBSERVER -d postgres -q "ALTER USER\"nbdbadmin\"
WITH PASSWORD '$OLD_DBADMINPASSWORD';"
```

Note: You can reset the flexible server password by using the following command. This command does not require az extension and potentially could be run outside of the az-cli container.

```
az postgres flexible-server update -g $RESOURCE_GROUP -n $DBSERVER
--admin-password <password>
```

---

10 Use the following command to verify if the password is using the correct encryption method (SCRAM-SHA-256):

```
az postgres flexible-server execute -p "$OLD_DBADMINPASSWORD" -u
$DBADMINUSER -n $DBSERVER -d postgres -q "SELECT * from
azure_roles_authtype();" -o table
```

```
Successfully connected to twilk-db.
Ran Database Query: 'SELECT * from azure_roles_authtype();'
Retrieving first 30 rows of query output, if applicable.
Closed the connection to twilk-db
Authtype    Rolename
----------  -------------------------
NOLOGIN     azuresu
NOLOGIN     pg_database_owner
NOLOGIN     pg_read_all_data
NOLOGIN     pg_write_all_data
NOLOGIN     pg_monitor
NOLOGIN     pg_read_all_settings
NOLOGIN     pg_read_all_stats
NOLOGIN     pg_stat_scan_tables
NOLOGIN     pg_read_server_files
NOLOGIN     pg_write_server_files
NOLOGIN     pg_execute_server_program
NOLOGIN     pg_signal_backend
NOLOGIN     azure_pg_admin
NOLOGIN     replication
SCRAM-256   nbdbadmin
```

11 Store the updated password in key vault:

Note: This step can be skipped if the password is not changed.

```
az keyvault secret set --vault-name $KEY_VAULT_NAME --name
dbadminpassword --value "$NEW_DBADMINPASSWORD"
```

12 (Optional) Delete the key vault access policy created in step 4 above:

```
$ az keyvault delete-policy -n $KEYVAULT --upn $USER_ID
```

13 Exit from the azure CLI pod:

```
$ exit
```

14 Delete the az CLI pod:

```
$ kubectl delete pod az-cli
```

15 (*Applicable only for an existing cloudscale deployment*) Restart the primary pod:

```
$ kubectl rollout restart "statefulset/${PRIMARY}" --namespace
"${NAMESPACE}"
```

In the above command,

- NAMESPACE is the namespace containing your NetBackup deployment
- PRIMARY is the name of primary pod's stateful set

Use the following command to obtain NAMESPACE and PRIMARY:

```
$ kubectl get --namespace "${NAMESPACE}" primaryserver -o
jsonpath='{.items[0].status.attributes.resourceName}'
```

AWS-specific

1 Use lambda function to change the password.

LAMBDA_ARN is the ARN of the password changing lambda function. This can be obtained from the lambda function page on AWS console.

NEW_PASSWORD is the new password to be used.

```
$ aws lambda invoke --function-name $LAMBDA_ARN \
--cli-binary-format raw-in-base64-out --payload
'{"password":"$NEW_PASSWORD"}' \ response_file
```

2 Wait for database to be available.

Obtain the POSTGRESQL_ID (database identifier)of your RDS Postgres database from the RDS database page of the AWS console, using the following command:

```
$ aws rds wait db-instance-available --db-instance-identifier
$POSTGRESQL_ID
```

3 Restart the primary pod:

```
$ kubectl rollout restart "statefulset/${PRIMARY}" --namespace
"${NAMESPACE}"
```

In the above command,

- NAMESPACE is the namespace containing your NetBackup deployment
- PRIMARY is the name of primary pod's stateful set

Use the following command to obtain NAMESPACE and PRIMARY:

```
$ kubectl get --namespace "${NAMESPACE}" primaryserver -o
jsonpath='{.items[0].status.attributes.resourceName}'
```

Containerized PostgreSQL

1. Exec into primary pod and change database password using the following command:

   ```
   $ kubectl exec -it <primary-pod-name> -n netbackup -- bash
   ```

   ```
   # /usr/openv/db/bin/nbdb_admin -dba "<new-password>"
   ```

   ```
   # exit
   ```

2. Update the database connection secret with new password:

   - Set the new password:

     ```
     $ kubectl patch secret <secret-name> -n netbackup -p
     '{"stringData": {"dbadminpassword": "<new-password>" }}'
     ```

   - Verify the new password:

     ```
     $ kubectl get secret <secret-name> -n netbackup -o
     jsonpath='{.data.dbadminpassword}' | base64 --decode
     ```

3. Restart the Postgres and primary pods:

   - Identify Postgres and primary statefulsets:

     ```
     $ kubectl get statefulset -n netbackup
     ```

   - Restart Postgres pod:

     ```
     $ kubectl rollout restart "statefulset/nb-postgresql" -n
     netbackup
     ```

   - Wait for the Postgres pod to restart:

     ```
     $ kubectl rollout status --watch --timeout=600s
     "statefulset/nb-postgresql" -n netbackup
     ```

   - Restart primary pod:

     ```
     $ kubectl rollout restart "statefulset/<primary-statefulset>"
     -n netbackup
     ```

   - Wait for primary pod to restart:

     ```
     $ kubectl rollout status --watch --timeout=600s
     "statefulset/<primary-statefulset>" -n netbackup
     ```

4. (*For Cloud Scale with decoupled web services only*) Restart the web services pod:

   - Identify nbwsapp statefulset:

     ```
     kubectl get statefulset -n netbackup
     ```

- Restart nbwsapp pod:

  ```
  kubectl rollout restart "statefulset/<nbwsapp-statefulset>"
  -n netbackup
  ```

- Wait for the nbswapp pod to restart:

  ```
  kubectl rollout status --watch --timeout=600s
  "statefulset/<nbwsapp-statefulset>" -n netbackup
  ```

5 (*For Cloud Scale with decoupled bpdbm services only*) Restart the bpdbm services pod:

- Identify bpdbm statefulset:

  ```
  kubectl get statefulset -n netbackup
  ```

- Restart bpdbm pod:

  ```
  kubectl rollout restart "statefulset/<bpdbm-statefulset>" -n
  netbackup
  ```

- Wait for the bpdbm pod to restart:

  ```
  kubectl rollout status --watch --timeout=600s
  "statefulset/<bpdbm-statefulset>" -n netbackup
  ```

Updating database certificate in DBaaS

Create Secret containing DBaaS CA certificates. NetBackup version 10.4 and later stores db cert in `db-cert` configMap instead of KeyVault/SecretsManager secret. The `db-cert` configMap is created by trust manager.

Note the following:

- Skip the steps in this section when using containerized Postgres.
- The steps in this section are only required for upgrading from 10.4 and 10.4.0.1 (non-decoupled services) to 10.5 and above (decoupled services).
- Upgrade from embedded to DBaaS is not supported.

1. Perform the following to create Secret containing DBaaS CA certificates:

   - **AKS-specific**:

     ```
     DIGICERT_ROOT_CA='/tmp/root_ca.pem'
     DIGICERT_ROOT_G2='/tmp/root_g2.pem'
     MS_ROOT_CRT='/tmp/ms_root.crt'
     COMBINED_CRT_PEM='/tmp/tls.crt'

     DIGICERT_ROOT_CA_URL="https://cacerts.digicert.com/DigiCertGlobalRootCA.crt.pem";
     DIGICERT_ROOT_G2_URL="https://cacerts.digicert.com/DigiCertGlobalRootG2.crt.pem";
     ```

```
MS_ROOT_CRT_URL="http://www.microsoft.com/pkiops/certs/Microsoft%20RSA%20Root%20Certificate%20Authority%202017.crt";


curl ${DIGICERT_ROOT_CA_URL} --output ${DIGICERT_ROOT_CA}
curl ${DIGICERT_ROOT_G2_URL} --output ${DIGICERT_ROOT_G2}
curl ${MS_ROOT_CRT_URL} --output ${MS_ROOT_CRT}

openssl x509 -inform DER -in ${MS_ROOT_CRT} -out
${COMBINED_CRT_PEM} -outform PEM
cat ${DIGICERT_ROOT_CA} ${DIGICERT_ROOT_G2} >>
${COMBINED_CRT_PEM}

kubectl -n netbackup create secret generic
postgresql-netbackup-ca --from-file ${COMBINED_CRT_PEM}
```

- **EKS-specific**:

```
TLS_FILE_NAME='/tmp/tls.crt'
PROXY_FILE_NAME='/tmp/proxy.pem'

rm -f ${TLS_FILE_NAME} ${PROXY_FILE_NAME}

DB_CERT_URL="https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem"
DB_PROXY_CERT_URL="https://www.amazontrust.com/repository/AmazonRootCA1.pem"


curl ${DB_CERT_URL} --output ${TLS_FILE_NAME}
curl ${DB_PROXY_CERT_URL} --output ${PROXY_FILE_NAME}

cat ${PROXY_FILE_NAME} >> ${TLS_FILE_NAME}

kubectl -n netbackup create secret generic
postgresql-netbackup-ca --from-file ${TLS_FILE_NAME}
```

2. Restart the primary pod using the following command:

```
kubectl rollout restart "statefulset/${PRIMARY}" --namespace
"${NAMESPACE}"
```

Chapter 15

Managing logging

This chapter includes the following topics:

- Viewing NetBackup logs
- Extracting NetBackup logs

Viewing NetBackup logs

To view NetBackup logs you can exec into the log-viewer pod and view the logs or extract them first and view them after they have been extracted. For more information on extracting the logs, see the following section:

See "Extracting NetBackup logs" on page 229.

To view NetBackup logs

1. First find the log-viewer pod: `$ kubectl get pod -n netbackup | grep log-viewer`

   ```
   nb-log-viewer-0 1/1 Running 0 7d
   ```

2. Exec into the log-viewer pod using the pod name from the previous command:

   ```
   $ kubectl exec -it -n netbackup nb-log-viewer-0 -- /bin/bash
   ```

3. Move into the fluentbit log location:

   ```
   $ cd /usr/openv/fluentbit/logs
   ```

4. Folders are labeled by date:

   ```
   $ ls
   ```

   ```
   2024-02-02 2024-02-03
   ```

5 Move into the folder of the date you are looking at and the logs are grouped by namespace:`$ cd 2024-02-03/`

```
$ ls
```

```
netbackup netbackup-operator-system kube-system trust-manager
```

6 Move into the folder of the namespace you are looking at and the logs are there labeled by pod name:

```
$ cd netbackup/
```

```
$ ls
```

```
<netbackup namespace>-nbatd-0 <netbackup namespace>-policyjob-0
<netbackup namespace>-primary-0 nb-fluentbit-daemonset-2l4sh
nb-fluentbit-daemonset-7ndsg nb-fluentbit-daemonset-qlv4m
<netbackup namespace>-nbwsapp-0 <netbackup namespace>-bpdbm-0
<netbackup namespace>-policyjobmgr-0
nb-fluentbit-collector-7dfc4d95b8-mzrkr
nb-fluentbit-aemonset-6nj66 nb-fluentbit-daemonset-9pslq
nb-fluentbit-daemonset-wpwp4
```

The log-viewer pod has `vi` built into it so you can use that to view the logs in the container itself.

7 From NetBackup version 11.0 and later, the flexsnap datamover logs can be viewed from the following fluenbit log locations:

- Flexsnap datamover STDOUT pod logs:

  ```
  cd
  /usr/openv/fluentbit/logs/<date>/<env_namespace>/flexsnap-datamover-<id>/flexsnap-datamover-<id>
  ```

- Datamover services logs:

  ```
  cd
  /usr/openv/fluentbit/logs/<date>/<env_namespace>/cloudpoint/openv/dm/datamover.<id>
  ```

Extracting NetBackup logs

You can extract NetBackup logs from the Cloud Scale environment using the following options:

- (*Recommended*) Use the NetBackup Troubleshooting APIs for log extraction requests
- Copy logs out of the log-viewer using the `kubectl` commands
- Copy logs out if the **log-viewer** or **nbwsapp** are not working

Extracting logs via NetBackup Troubleshooting APIs

NetBackup Cloud Scale Technology introduced log extraction APIs in version 11.0 to assist in extracting logs from a Cloud Scale environment.

Table 15-1 Extract logs via APIs

Action	Command / Description
In order to extract logs via the APIs you must go through a series of calls to create a request and extract the logs. All API endpoints require a valid JWT Authorization header.	**1** **Create Async Request**: Create the request. Use the filters described below to specify specific log types to extract. See Location Header for Async Request ID. POST /netbackup/troubleshooting/async-log-download-requests **2** **Get Async Request Status**: Check the status of the previous request. See Location Header for Async Result ID. GET /netbackup/admin/async-requests/<async_request_status_id> **3** **Get Async Request Result** (Usually done via redirect): Gets the Async Request Result. See Location Header for Log Download Request ID. GET /netbackup/troubleshooting/async-log-download-requests-results/<async_result_id> **4** **Get Log Download Request**: This retrieves the Log Download Request information. See **data.relationships.archive.links.related.href** for URL to download logs. GET /netbackup/troubleshooting/log-download-requests/<log_download_request_id> **5** **Download Log Download Request Archive**: This downloads the logs. /netbackup/troubleshooting/log-download-requests/<log_download_request_id>/logs/netbackup-logs.zip

Filtering: The **Create Async Request** endpoint has filters that can be applied to target specific logs that the user wants to gather.

Following filters exist:

globalFilter	■ startTime: The start timestamp to look for files. ■ endTime: The end timestamp to look for files. ■ namespaces: A list of kubernetes namespaces to extract logs from.
legacyLogFilter	directories: Legacy log directories to pull logs from.
unifiedLogFilter	fileIds: A list of UL file ID's to pull logs from.

It is recommended to use the provided filters to be specific about log request in order to avoid timeout. Default: 60 min

---

Note: All the filters are optional and have an `AND` relationship. This indicates that if you provide filters from multiple categories, they must match all the categories to be considered valid for extraction.

Limitations

Among the filtering options specified above, **globalFilter** is applicable to every log. But **legacyLogFilter** and **unifiedLogFilter** is applicable only to logs that are part of those specific categories.

For example, if you applied a date range via globalFilters and applied a legacyLogFilter and a unifiedLogFilter then you would get all legacy logs that fit the legacyLogFilter and the globalFilters and all unified logs that fit the unifiedLogFilter and the global filters.

By default, some of the logs have long names and long paths. This can cause issues on windows when extracting the logs on a windows computer. Hence the logs would be skipped and would not be a part of the resultant unzipped folder. Hence it is recommended that you extract these logs onto a Linux based file system.

Extract logs via kubectl cp commands

In order to extract the logs out of the log-viewer pod there are a few options. You can tar and compress the logs before extraction or extract them immediately.

1. (*Optional*) Tar up the files you want to extract. Select a folder and run:

   ```
   $ tar -cvf <name of tar> <name of folder>
   ```

2. Copy the files out of the container. Exit the container using the command:

   ```
   $ kubectl cp -n netbackup
   <pod-name>:/usr/openv/fluentbit/logs/<folder or tar> <output
   folder or tar>
   ```

3. (*Optional*) Extract the tar outside the container if necessary:

```
$ tar xvf <output tar>
```

Extracting logs if nbwsapp or log-viewer pods are not working

If **nbwsapp** or **log-viewer** pods are not working, then you need to extract logs from other pods. Use the following command to copy logs out of the pods:

```
$ kubectl cp -n netbackup <pod-name>:/usr/openv/fluentbit/logs/<folder
or tar> <output folder or tar>
```

The first pod to try must be fluentbit collector pod as it also mounts the file system storing all the Cloud Scale logs. If the fluentbit collector pod is not working you will need to copy the logs directly from individual application pods such as nbwsapp or primary. Logs within application pods are usually stored at `/mnt/nblogs` directory.

Chapter 16

Performing catalog backup and recovery

This chapter includes the following topics:

Backing up a catalog

You can backup a catalog by using one of the following methods:

- Automatically
- Manually

Automatic creation of catalog backup policy

You can backup a catalog by using the automatically created catalog backup policy which is applicable only for fresh/new deployment.

To backup a catalog automatically

1. A catalog backup policy can be automatically configured during a new installation. This can be done by supplying the DR Secret through the **drInfoSecret** field of the `environment.Spec` in `helm/values.yaml` file.
2. The **drInfoSecret** field must be created before deployment using the following command:

   ```
   kubectl create secret generic dr-info-secret --namespace
   <nbu-namespace> --from-literal=passphrase="Y@123abCdEf"
   --from-literal=emailAddress="abc@xyz.com"
   ```

   The passphrase field is compulsory.

3 Once catalog policy is created, configure **Recovery Vault** storage in the catalog backup policy. For more information, see *NetBackup Deduplication Guide*.

4 In the automatically configured catalog backup policy, the DR package path is set to `/mnt/nbdb/usr/openv/drpackage_<storage server name>`. If required, this can be changed by editing the policy from the Web UI.

5 If the email field is included in the DR Secret, then on running a catalog backup job, the created **DRPackages** would be sent through email. this is applicable only when the e-mail server is configured. See "Configuring email server" on page 109.

Manual creation of catalog backup policy

To backup a catalog manually

1 Exec into the primary server pod using the following command:

```
kubectl exec -it -n <namespace> <primary-pod-name> -- /bin/bash
```

2 Create a directory **DRPackages** at persisted location using `mkdir /mnt/nblogs/DRPackages`.

3 Change ownership of **DRPackages** folder to service user using `chown nbsvcusr:nbsvcusr /mnt/nblogs/DRPackages`.

4 Set the passphrase to be used at time of catalog recovery.

- Open NetBackup Administrator Console (Java UI).
- Navigate to **Security Management > Global Security Setting > Disaster Recovery**.
- In **Encryption for Disaster Recovery** section, add the passphrase, confirm passphrase, and save it.

5 Add respective external media server entry in host properties through **NetBackup Management > Host properties > Master Server > Add Additional server**.

Note: It is recommended to use an external media server for catalog backup and recovery.

---

6 Exec into the primary server pod using the following command:

```
kubectl exec -it -n <namespace> <primaryserver pod name> -- bash
```

Set the **KMS_CONFIG_IN_CATALOG_BKUP** configuration option to *1* in `/usr/openv/netbackup/bp.conf` file of primary server to include the KMS configuration as part of the disaster recovery package during catalog backup.

7 Restart the NetBackup services in primary and external media server.

- Exec into the primary server pod using the following command:
  `kubectl exec -it -n <namespace> <primary-pod-name> -- /bin/bash`
- Deactivate NetBackup health probes using the `/opt/veritas/vxapp-manage/nb-health deactivate` command.
- Run the `/usr/openv/netbackup/bin/bp.kill_all` command. After stopping all services restart the services using the `/usr/openv/netbackup/bin/bp.start_all` command.
- Activate NetBackup health probes using the `/opt/veritas/vxapp-manage/nb-health activate` command.
- Run the `/usr/openv/netbackup/bin/bp.kill_all` command. After stopping all services restart the services using the `/usr/openv/netbackup/bin/bp.start_all` command on the external media server.

8 Configure storage unit on earlier added external media server.

For more information, refer to the *NetBackup™ Administrator's Guide, Volume I*

Note: It is recommended to use AdvancedDisk or BasicDisk storage unit.

---

9 Configure NetBackup catalog backup policy.

Add package path as `/mnt/nblogs/DRPackages` while configuring the catalog backup policy.

10 Run the catalog backup job.

Restoring a catalog

You can restore a catalog. This section describes the procedures for restoring a catalog when catalog backup is taken on external media server or on MSDP-X and the primary server is corrupted.

Primary server corrupted

Primary server corrupted

- When catalog backup is taken on external media server
- When catalog backup is taken on MSDP-X

1. Copy DRPackages files (packages) located at `/mnt/nbdata/DRPackages/` (or wherever the DRPackage is located, on the primary pod) from the pod to the host machine from where Kubernetes Service cluster is accessed.

   Run the `kubectl cp <primary-pod-namespace>/<primary-pod-name>:/mnt/nbdata/DRPackages <Path_where_to_copy_on_host_machine>` command.

   Preserve the DRPackages as they will be required during Catalog recovery.

2. Change CR spec from *paused: false* to *paused: true* in primary, mediaServers, and msdpScaleouts sections in environment object using the following command:

```
helm upgrade cloudscale <path to cloudscale.tar.gz>
--reuse-values  \
--set environment.primary.paused=true \
--set environment.msdpscaleout.paused=true \
--set environment.mediaServers[0].name=<original value> --set
environment.mediaServers[0].paused=true \
--set environment.mediaServers[0].nodeSelector.labelKey=<original
 value> --set
environment.mediaServers[0].nodeSelector.labelValue=<original
value> \
--set environment.mediaServers[0].replicas=<original value> --set
 environment.mediaServers[0].minimumReplicas=<original value> \
--set environment.mediaServers[0].storage.data.capacity=<original
 value>Gi \
--set
environment.mediaServers[0].storage.data.storageClassName=<original
 sc name> \
--set environment.mediaServers[0].storage.log.capacity=<original
 value>Gi \
--set
environment.mediaServers[0].storage.log.storageClassName=<original
 sc name> \
-n netbackup
```

3 Delete all statefulSets, deployments, and any existing jobs associated with the primary decoupled services.

```
kubectl get sts -n netbackup
NAME                   READY   AGE
flexsnap-rabbitmq      1/1     5h24m
msdpx-uss-controller   1/1     5h32m
nb-postgresql          1/1     5h53m
nbu-bpdbm              0/1     5h49m
nbu-log-viewer         0/1     5h49m
nbu-nbatd              1/1     5h49m
nbu-nbmqbroker         0/1     5h49m
nbu-nbwsapp            0/1     5h49m
nbu-policyjob          0/1     5h49m
nbu-policyjobmgr       0/1     5h49m
nbu-primary            0/1     5h49m

kubectl get deployment -n netbackup
NAME                                                 READY
UP-TO-DATE   AVAILABLE   AGE
flexsnap-agent                                       1/1     1
       1           5h23m
flexsnap-agent-2580d680823a4cfaa54c9fe9fdff8c84      1/1     1
       1           3h2m
flexsnap-api-gateway                                 1/1     1
       1           5h23m
flexsnap-certauth                                    1/1     1
       1           5h26m
flexsnap-coordinator                                 1/1     1
       1           5h23m
flexsnap-listener                                    1/1     1
       1           5h23m
flexsnap-nginx                                       1/1     1
       1           5h23m
flexsnap-notification                                1/1     1
       1           5h23m
flexsnap-scheduler                                   1/1     1
       1           5h23m
nb-fluentbit-collector                               1/1     1
       1           5h54m
nbu-requestrouter                                    1/1     1
       1           5h46m
```

```
kubectl delete sts nbu-log-viewer nbu-bpdbm nbu-nbatd
nbu-nbmqbroker nbu-nbwsapp nbu-policyjob nbu-policyjobmgr
nbu-primary -n netbackup
statefulset.apps/nbu-bpdbm deleted
statefulset.apps/nbu-log-viewer deleted
statefulset.apps/nbu-nbatd deleted
statefulset.apps/nbu-nbmqbroker deleted
statefulset.apps/nbu-nbwsapp deleted
statefulset.apps/nbu-policyjob deleted
statefulset.apps/nbu-policyjobmgr deleted
statefulset.apps/nbu-primary deleted

kubectl delete deployment nbu-requestrouter -n netbackup
deployment.apps/nbu-requestrouter deleted

kubectl delete job nbu-bootstrapper -n netbackup
job.batch "nbu-bootstrapper" deleted from netbackup namespace
```

Delete the db-cert configmap, it will be re-created by the bundle:

```
~/VRTSk8s-netbackup-<version>/helm$ kubectl get bundle -n
netbackup
NAME      CONFIGMAP TARGET   SECRET TARGET   SYNCED   REASON
AGE
db-cert   dbcertpem                          True     Synced
3h54m

~/VRTSk8s-netbackup-<version>/helm$ kubectl get cm -n netbackup
NAME                             DATA   AGE
appconf                          1      3h30m
cs-config                        1      4h12m
db-cert                          1      3h54m
svcmapping                       1      3h51m
~/VRTSk8s-netbackup-<version>/helm$ kubectl delete cm db-cert -n
 netbackup
configmap "db-cert" deleted from netbackup namespace
~/VRTSk8s-netbackup-<version>/helm$ kubectl get cm -n netbackup
NAME                             DATA   AGE
appconf                          1      3h30m
cs-config                        1      4h13m
db-cert                          1      7s
flexsnap-conf                    1      3h30m
flexsnap-deployment-conf-cp      24     3h30m
```

4 Remove the PVs and PVC's associated with the primary server decoupled pods.

```
~/Ampol$ kubectl delete pvc catalog-nbu-primary-0
data-nbu-primary-0 logs-nbu-primary-0 nbatd-data nbatd-logs
nbmqbroker-log policy ws-log bpdbm-logs -n netbackup
persistentvolumeclaim "bpdbm-log" deleted from netbackup namespace
persistentvolumeclaim "catalog-nbu-primary-0" deleted from
netbackup namespace
persistentvolumeclaim "data-nbu-primary-0" deleted from netbackup
 namespace
persistentvolumeclaim "logs-nbu-primary-0" deleted from netbackup
 namespace
persistentvolumeclaim "nbatd-data" deleted from netbackup
namespace
persistentvolumeclaim "nbatd-logs" deleted from netbackup
namespace
persistentvolumeclaim "nbmqbroker-log" deleted from netbackup
namespace
persistentvolumeclaim "policyjob-log-nbu-policyjob-0" deleted
from netbackup namespace
persistentvolumeclaim "policyjobmgr-log-nbu-policyjobmgr-0"
deleted from netbackup namespace
persistentvolumeclaim "ws-log" deleted from netbackup namespace
```

```
~/Ampol/DR$ kubectl get pv -n netbackup | grep Released
pvc-0bd4e64b-e089-493a-8ff1-e2eeafa7a5e9   30Gi       RWO
     Retain           Released   netbackup/logs-nbu-primary-0
                                             nb-
pvc-1a887c36-b860-424a-b35a-fc21313ef532   30Gi       RWO
     Retain           Released
netbackup/policyjob-log-nbu-policyjob-0
            nb-
pvc-1dd6720d-6cce-47f8-950c-5067a7dc384a   30Gi       RWO
     Retain           Released   netbackup/ws-log
                                             nb-
pvc-2be78c3f-df8d-4724-b72d-dbb60a44452a   30Gi       RWO
     Retain           Released   netbackup/nbmqbroker-log
                                             nb-
pvc-733438f4-da39-4031-a164-8a9706534ecd   110Gi      RWX
     Retain           Released   netbackup/catalog-nbu-primary-0
                                             nb-
```

```
pvc-9d421d55-e136-4de7-aa48-3026e931ccb4   30Gi       RWO
     Retain           Released   netbackup/data-nbu-primary-0
                                             nb-
pvc-c9e78f63-4ab3-4515-b41b-8711ef61ec10   2Gi        RWO
     Retain           Released   netbackup/nbatd-data
                                             nb-
pvc-cfe6ec5c-8367-42c7-b3a9-d99fc7087504   30Gi       RWO
     Retain           Released
netbackup/policyjobmgr-log-nbu-policyjobmgr-0
            nb-
pvc-ab1e8d27-2da2-4340-aae4-c4bb702db7ce   30Gi       RWO
     Retain           Released   netbackup/bpdbm-logs
                                            nb-



~/Ampol/DR$ kubectl delete pv
pvc-0bd4e64b-e089-493a-8ff1-e2eeafa7a5e9
pvc-1a887c36-b860-424a-b35a-fc21313ef532 pvc-1dd6720d-6cce-47f8
pvc-733438f4-da39-4031-a164-8a9706534ecd
pvc-9d421d55-e136-4de7-aa48-3026e931ccb4
pvc-c9e78f63-4ab3-4515-b41b-8711ef61ec10
pvc-cfe6ec5c-8367-42c7-b3a9-d99fc7087504 pvc-fe1e8d2
pvc-ab1e8d27-2da2-4340-aae4-c4bb702db7ce
persistentvolume "pvc-0bd4e64b-e089-493a-8ff1-e2eeafa7a5e9"
deleted
persistentvolume "pvc-1a887c36-b860-424a-b35a-fc21313ef532"
deleted
persistentvolume "pvc-1dd6720d-6cce-47f8-950c-5067a7dc384a"
deleted
persistentvolume "pvc-2be78c3f-df8d-4724-b72d-dbb60a44452a"
deleted
persistentvolume "pvc-733438f4-da39-4031-a164-8a9706534ecd"
deleted
persistentvolume "pvc-9d421d55-e136-4de7-aa48-3026e931ccb4"
deleted
persistentvolume "pvc-c9e78f63-4ab3-4515-b41b-8711ef61ec10"
deleted
persistentvolume "pvc-cfe6ec5c-8367-42c7-b3a9-d99fc7087504"
deleted
persistentvolume "pvc-fe1e8d27-2da2-4340-aae4-c4bb702db7ce"
deleted
persistentvolume "pvc-ab1e8d27-2da2-4340-aae4-c4bb702db7ce"
deleted
```

5 Perform the following:

- Change CR spec `paused: true` to `paused: false` for the environment using the following command:

  ```
  helm upgrade cloudscale <path to cloudscale.tar.gz>
  --reuse-values --set environment.paused=true -n netbackup
  ```

- Reinstall the PostgreSQL latest release:

  - Save the `dbSecret.yaml` file on a persistent location using the following command:

    ```
    kubectl get secret dbsecret -n netbackup -o yaml >
    dbSecret.yaml
    ```

  - Uninstall the PostgreSQL using the following command:

    ```
    helm upgrade cloudscale <path to cloudscale.tar.gz>
    --reuse-values --set postgresql.enabled=false -n netbackup
    ```

    Wait for the postgres pod to terminate.

  - Recreate the secret from the details saved above:

    ```
    kubectl apply -f dbSecret.yaml
    ```

  - Reinstall the PostgreSQL using the following command:

    ```
    helm upgrade cloudscale <path to cloudscale.tar.gz>
    --reuse-values --set postgresql.enabled=true -n netbackup
    ```

6 Retag and push the `netbackup/main` image with a unique tag (distinct from other decoupled pods, such as MR1), and add the `serviceImage` section under the primary environment with `bootstrapper.main: <new-image-tag>` using the following command:

```
helm upgrade cloudscale <path to cloudscale.tar.gz> --reuse-values
\ --set
environment.primary.serviceImageTag."bootstrapper\.main"="<newImage
tag>" -n netbackup
```

Change CR spec `paused: true` to `paused: false` in the primary server and environment sections in the environment using the following command:

```
helm upgrade cloudscale <path to cloudscale.tar.gz> --reuse-values
\ --set environment.primary.paused=false --set
environment.paused=false -n netbackup
```

7 After the primary server pod is in ready state, change CR spec from `paused: false` to `paused: true` in primary server section in environment object using the following command:

```
helm upgrade cloudscale <path to cloudscale.tar.gz> --reuse-values
\ --set environment.primary.paused=true -n netbackup
```

8 Once the primary server is up and running perform the following:

- Execute the `kubectl exec -it -n <namespace> <primary-pod-name> -- /bin/bash` command to exec into the primary pod.
- Increase the debug logs level on primary server (set *VERBOSE = 6* in `bp.conf` file).
- Create a **DRPackages** directory at persisted location using the following command:
  `mkdir /mnt/nbdata/usr/openv/drpackage`
- Deactivate NetBackup health probes using the `/opt/veritas/vxapp-manage/nb-health` deactivate command.
- Stop the NetBackup services using the following command:
  `/usr/openv/netbackup/bin/bp.kill_all`
  Check if all processes are terminated correctly using the `/usr/openv/netbackup/bin/bpps` command.
- Perform the following steps for NBATD pod recovery:
  - Create the `DRPackages` directory on persisted location `/mnt/nblogs/` in nbatd pod by executing the following command:
    ```
    kubectl exec -it -n <namespace> <nbatd-pod-name> --/bin/bash
    mkdir /mnt/nblogs/DRPackages
    ```
  - Copy DR files which were saved when performing DR backup to nbatd pod at `/mnt/nblogs/DRPackages` using the following command:
    ```
    kubectl cp <Path_of_DRPackages_on_host_machine>
    <nbatd-pod-namespace>/<nbatd-pod-name>:/mnt/nblogs/DRPackages
    ```
  - Execute the following steps in the nbatd pod:
    - Execute the `kubectl exec -it -n <namespace> <nbatd-pod-name> --/bin/bash` command.
    - Deactivate nbatd health probes using the `/opt/veritas/vxapp-manage/nbatd_health.sh disable` command.
    - Stop the nbatd service using `/opt/veritas/vxapp-manage/nbatd_stop.sh 0` command.

    - Execute the `/opt/veritas/vxapp-manage/nbatd_identity_restore.sh -infile /mnt/nblogs/DRPackages/ (DR package name)` command.

- Copy back the earlier copied disaster recovery files to primary pod at `/mnt/nbdata/usr/openv/drpackage` location using the following command:

  `kubectl cp <Path_of_DRPackages_on_host_machine> <primary-pod-namespace>/<primary-pod-name>:/mnt/nbdata/usr/openv/drpackage`

- Change the ownership of files in `/mnt/nbdb/usr/openv/drpackage` using the `chown nbsvcusr:nbsvcusr <file-name>` command.

- Execute the `/usr/openv/netbackup/bin/admincmd/nbhostidentity -import -infile /mnt/nbdb/usr/openv/drpackage/.drpkg` command.

- Clear NetBackup host cache, run the `bpclntcmd -clear_host_cache` command.

- Restart the pods as follows:

    - Navigate to the `VRTSk8s-netbackup-<version>/scripts` folder.

    - Run the `cloudscale_restart.sh` script with Restart option as follows:

      `./cloudscale_restart.sh` *`<action>`* *`<namespace>`*

      Provide the namespace and the required action:

      stop: Stops all the services under primary server (waits until all the services are stopped).

      start: Starts all the services and waits until the services are up and running under primary server.

      restart: Stops the services and waits until all the services are down. Then starts all the services and waits until the services are up and running.

- Refresh the certificate revocation list using the `/usr/openv/netbackup/bin/nbcertcmd -getcrl` command.

9 (***Applicable for catalog backup taken on external media server***)

- Add respective media server entry in host properties using NetBackup Administration Console as follows:

  Navigate to **NetBackup Management > Host properties > Master Server > Add Additional server** and add media server.

- Restart the NetBackup services in primary server pod and external media server

- Execute the `kubectl exec -it -n <namespace> <primary-pod-name> -- /bin/bash` command in the primary server pod.
- Run the `/usr/openv/netbackup/bin/bp.kill_all` command. After stopping all services restart the same using the `/usr/openv/netbackup/bin/bp.start_all` command.
- External media server: Run the `/usr/openv/netbackup/bin/bp.kill_all` command. After stopping all services restart the services using the `/usr/openv/netbackup/bin/bp.start_all` command on the external media server.

- Scaleout the msdp nodepool and unpause the **msdpScaleout** using the following command:

```
helm upgrade cloudscale <path to cloudscale.tar.gz>
--reuse-values --set environment.msdpscaleout.paused=false -n
netbackup
```

  Ensure that the msdpScaleout CR is in a ready state before moving ahead.

- Scaleout the media nodepool and unpause the media object using the following command:

```
helm upgrade cloudscale <path to cloudscale.tar.gz>
--reuse-values \
--set environment.mediaServers[0].name=<original name> --set
environment.mediaServers[0].paused=false \
--set
environment.mediaServers[0].nodeSelector.labelKey=<original
name> --set
environment.mediaServers[0].nodeSelector.labelValue=<original
 value> \
--set environment.mediaServers[0].replicas=<original value>
--set environment.mediaServers[0].minimumReplicas=<original
value> \
--set
environment.mediaServers[0].storage.data.capacity=<original
value>Gi \
--set
environment.mediaServers[0].storage.data.storageClassName=<original
 sc name> \
--set
environment.mediaServers[0].storage.log.capacity=<original
value>Gi \
--set
```

```
environment.mediaServers[0].storage.log.storageClassName=<original
 sc name> \
-n netbackup
```

- Perform catalog recovery:
  - Exec into primary pod and run `bprecover -wizard` command.
  - Reset the debug logs level on primary server (remove VERBOSE = 6 from `bp.conf` file).
  - Restart the pods as follows:
    Navigate to the `VRTSk8s-netbackup-<version>/scripts` folder.
    Run the `cloudscale_restart.sh` script with Restart option as follows:
    `./cloudscale_restart.sh <action> <namespace>`
    Provide the namespace and the required action.

10 (***Applicable for catalog backup taken on MSDP-X***)

- From Web UI, allow reissue of token from primary server for MSDP only as follows:
  Navigate to **Security > Host Mappings** for the MSDP storage server and select **Allow Auto reissue Certificate**.
- Run the primary server reconciler as follows:
  - Run the following command to pause the primary server CR:
    ```
    helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup
    --reuse-values --set environment.primary.paused=true
    ```
  - Run the following command to un-pause the primary server CR:
    ```
    helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup
    --reuse-values --set environment.primary.paused=false
    ```
- Scaleout the msdp nodepool and unpause the **msdpScaleout** using the following command:
  ```
  helm upgrade cloudscale <path to cloudscale.tar.gz>
  --reuse-values --set environment.msdpscaleout.paused=false -n
  netbackup
  ```
  Wait for the msdpScaleout CR to be ready, before moving ahead.
- Scaleout the media nodepool and unpause the media object using the following command:
  ```
  helm upgrade cloudscale <path to cloudscale.tar.gz>
  --reuse-values \
  --set environment.mediaServers[0].name=<original name> --set
  ```

```
environment.mediaServers[0].paused=false \
--set
environment.mediaServers[0].nodeSelector.labelKey=<original
name> --set
environment.mediaServers[0].nodeSelector.labelValue=<original
 value> \
--set environment.mediaServers[0].replicas=<original value>
--set environment.mediaServers[0].minimumReplicas=<original
value> \
--set
environment.mediaServers[0].storage.data.capacity=<original
value>Gi \
--set
environment.mediaServers[0].storage.data.storageClassName=<original
 sc name> \
--set
environment.mediaServers[0].storage.log.capacity=<original
value>Gi \
--set
environment.mediaServers[0].storage.log.storageClassName=<original
 sc name> \
-n netbackup
```

Ensure that the media pods are up and running before proceeding further.

- Verify if MSDP installation is successful and default MSDP storage server, STU and disk pool is created with old names. This takes some time.
  If the storage server and disk pool are not created in the Web UI after waiting for some time, you can manually create the Storage Server (STS) using the following commands:
  Create STS: `nbdevconfig -creatests -storage_server <server name> -stype <server type> -media_server <media server>`
  Add/STS and alias (using tpconfig): `tpconfig -add -storage_server <server name> -stype <server type> -sts_user_id <user ID> -password <password>`
  After completing these steps:
  - Refresh the Web UI and verify that the storage server is listed.
  - Create a disk pool on the storage server using the same name as before.
- Perform from step 2 in the following section:
  See "Scenario 2: MSDP Scaleout and its data is lost and the NetBackup primary server was destroyed and is re-installed" on page 435.

- Perform catalog recovery:
  - Exec into primary pod and run `bprecover -wizard` command.
  - Reset the debug logs level on primary server (remove VERBOSE = 6 from `bp.conf` file).
  - Restart the pods as follows:
    Navigate to the `VRTSk8s-netbackup-<version>/scripts` folder.
    Run the `cloudscale_restart.sh` script with Restart option as follows:
    `./cloudscale_restart.sh <action> <namespace>`
    Provide the namespace and the required action.

Section 4

Maintenance

Chapter 17

PostgreSQL DBaaS Maintenance

This chapter includes the following topics:

Configuring maintenance window for PostgreSQL database in AWS

You can enable a maintenance window for an Amazon RDS for PostgreSQL database instance by using one of the following methods:

- AWS Management Console
- AWS RDS Command Line Interface (CLI)
- RDS API

Using the AWS Management Console

1. Open the Amazon RDS console.
2. In the left navigation pane, navigate to **Databases**.
3. Select the database instance for which you want to enable a maintenance window and click **Modify**.
4. Under the **Maintenance** window section, select a day and time for the maintenance window.
5. Select **Continue** and then **Modify** the database instance.

Using the AWS RDS CLI

1. Open the Amazon RDS CLI.
2. Use the following command to enable a maintenance window for a database instance:

   ```
   aws rds modify-db-instance --db-instance-identifier mydbinstance
   --preferred-maintenance-window Mon:06:00-Mon:09:00
   ```

Using the RDS API

1. Use the **ModifyDBInstance** action and set the **PreferredMaintenanceWindow** field in the request parameter.
2. Send the request to the RDS API endpoint.

   You should be able to set the maintenance window to a specific day and time of your choice, provided is at least 30 minutes in the future and not within the next 24 hours. The maintenance window is specified in the **ddd:hh24:mi-ddd:hh24:mi** format, where:

   ddd is the three-letter abbreviation for the day of the week

   hh24 is the hour in 24-hour format

   mi is the minute

For more information refer to the "Maintaining a DB instance" section of the *Amazon RD User Guide*.

Setting up alarms for PostgreSQL DBaaS instance

This section describes the steps to create alarms for PostgreSQL flexible server using Azure and AWS portals.

Setting alarms for PostgreSQL DBaaS instance metrics in Azure

The alert triggers when the value of a specified metric crosses the assigned threshold. The alert triggers when the condition is met and when that condition is no longer being met.

Create an alert rule on a metrics from the Azure portal

1. In the Azure portal, select the Azure Database for PostgreSQL server you want to monitor.
2. Under the **Monitoring** section of the sidebar, select **Alerts**.
3. Select **Create alert rule** or **+**.

4 On the **Create an alert rule** page, under the **Condition** tab select **Add condition**.

5 Select a metric from the list of signals to be alerted on from the **Select a signal** page.

6 Configure the alert logic including the Condition (for example, Greater than), Threshold (for example, 85 percent), Time Aggregation, Period of time the metric rule must be satisfied before the alert triggers (for example, over the last 30 minutes), and Frequency.

7 Select **Next: Actions >**.

8 Under the **Actions** section, select **Create action group** to create a new group to receive notifications on the alert.

9 Fill in the **Basics** form with a name, display name, subscription, and resource group. Select **Next: Notifications >**.

10 Configure an **Email/SMS message/Push/Voice** action type by providing the details for all the required fields and then click **OK**.

11 (Optional) Select **Next: Actions >** to add actions based on the alerts.

12 Select **Review+Create** to review the information and create.

13 Provide the alert details. and select **Next/Review+Create**.

If the alert triggers, an email would be sent to the provided email id in the notification section.

For more information, refer to the PostgreSQL section of the *Azure documentation*.

Setting alarms for PostgreSQL DBaaS instance metrics in AWS

The following procedure is an example for creating an alarm that notifies the users if they run out of storage space on their RDS instance.

Create an alert rule on a metrics from the AWS portal

1 Open the CloudWatch console.

2 In the navigation pane, navigate to **Alarms** > **All Alarms.**

3 Select **Create alarm** > on the **Select metric** page > search for **FreeStorageSpace** metric > select **RDS** > select **Per-Database metrics**.

For the instance that you want to monitor, select the DB instance identifier **FreeStorageSpace** metric.

4 Click on **Select metric**.

5 In the **Specify metric and conditions** step, provide the required details.

6 In the **Configure actions** step, provide the required details.

After the details are provided and you click on **Create topic**, a confirmation email would be sent to the provided email address.

Open the email notification that you received from AWS Notifications, and then click on **Confirm subscription**.

7 Return to the **Configure actions** page in the CloudWatch console and click **Next**.

8 Enter a name and description for your alarm, and then click **Next**.

9 Review the preview of your metric, and then select **Create alarm**.

Note: It is a best practice to create a second, critical alarm for a lower threshold. For example, set your first alarm for 25 GB, and the second critical alarm to 10 GB.

For more information on creating alarms for other critical metrics, refer to the *Amazon RDS User Guide*.

To configure event notifications for Amazon RDS

1 Open the CloudWatch console.

2 In the search bar, search for **Simple Notification Services** and select **Topics** from the left pane.

3 Select **Create Topic** > **Standard**.

Enter the required details and click **Create.**

4 Select **Create Subscription** > **Email from** the list of protocols.

Enter the email address on which you want to receive notifications.

Note: Confirm the created subscription.

---

5 Navigate to **Amazon RDS console** > **Event subscriptions** > **Create event subscription**.

Enter the Name, select the ARN for the SNS topic, and select Instances as the Source type.

Select specific instances and select your instance.

6 Navigate to **Select specific event categories** > select **Maintenance** > **Create**.

For more information, refer to the RDS maintenance section of the *Amazon Knowledge Center*.

Chapter 18

Patching mechanism for primary, media servers, fluentbit pods, and postgres pods

This chapter includes the following topics:

Overview

NetBackup version 10.5 and later provides support for patching the following containers:

- Primary (main) and its other containers
- Media
- Fluentbit container
- Fluentbit cleanup container
- PostgreSQL pods

Patching introduces the ability for customers to patch images in a Kubernetes native way by specifying the image tags for respective containers using the **serviceImageTag** field of the environment. With **serviceImageTag** only primary server and media server of the environment can be patched.

Note: Ensure that the required patch images are pushed in the registry.

The poddependency-init image is present inside the operator image, hence to patch poddependency-init image, operator image must be pushed to the registry.

Patching of primary containers

This section describes the procedure for patching the following primary containers:

- Patching particular pod of primary server
- Patching Init containers for all pods of primary server
- Patching Sidecar containers for all pods of primary server

Patching particular pod of primary server

To patch the primary's main container with a particular image:

```
helm upgrade cloudscale cloudscale-11.2.x-xxxxxxx.tgz -n netbackup
--reuse-values --set
environment.primary.serviceImageTag."primary\.main=<new_img_tag>"
```

Table 18-1 Primary container keywords and examples

Profile name	Main container keyword	Init container keyword	Sidecar container keyword
Primary	`primary.main` For example, netbackup/main:<*version*>-patch	`primary.pod-dependency-init`	`primary.fluentbit`
Nbatd	`nbatd.main` For example, netbackup/nbatdmain:<*version*>-patch	`nbatd.nbatd-init` For example, netbackup/nbatd-init:<*version*>-patch	■ `nbatd.pbx` For example, netbackup/pbx:<*version*>-patch ■ `nbatd.fluentbit` ■ `nbatd.nbhousekeeping` For example, netbackup/nbhousekeeping:<*version*>-patch

Table 18-1 Primary container keywords and examples *(continued)*

Profile name	Main container keyword	Init container keyword	Sidecar container keyword

Mqbroker	`nbmqbroker.main`
	For example,
	`netbackup/mqbroker:`*<version>*`-patch`
		■ `nbmqbroker.nbmqbrokerinit`
		For example,
		`netbackup/mqbroker-init:`*<version>*`-patch`
		■ `nbmqbroker.pod-dependency-init`
			`nbmqbroker.fluentbit`

Webservice	`nbwsapp.main`
	For example,
	`netbackup/ws-app:`*<version>*`-patch`
		■ `nbwsapp.nbwsinit`
		For example,
		`netbackup/ws-init:`*<version>*`-patch`
		■ `nbwsapp.pod-dependency-init`
			■ `nbwsapp.secure-comms`
			For example,
			`netbackup/vnetd:`*<version>*`-patch`
			■ `nbwsapp.nbhousekeeping`
			For example,
			`netbackup/nbhousekeeping:`*<version>*`-patch`
			■ `nbwsapp.fluentbit`

Policyjob	`policyjob.main`
	For example,
	`netbackup/policyjobmgr:`*<version>*`-patch`
		`policyjob.pod-dependency-init`	■ `policyjob.pbx`
			For example,
			`netbackup/pbx:`*<version>*`-patch`
			■ `policyjob.fluentbit`
			■ policyjob .secure-comms
			For example,
			`netbackup/vnetd:`*<version>*`-patch`
			■ policyjob .nbhousekeeping
			For example,
			`netbackup/nbhousekeeping:`*<version>*`-patch`

Policyjobmgr	`policyjobmgr.main`
	For example,
	`netbackup/policyjobmgr:`*<version>*`-patch`
		`policyjobmgr.pod-dependency-init`	■ `policyjobmgr.pbx`
			For example,
			`netbackup/pbx:`*<version>*`-patch`
			■ `policyjobmgr .secure-comms`
			For example,
			`netbackup/vnetd:`*<version>*`-patch`
			■ `policyjobmgr .nbhousekeeping`
			For example,
			`netbackup/nbhousekeeping:`*<version>*`-patch`
			■ `policyjobmgr.fluentbit`

Table 18-1 Primary container keywords and examples *(continued)*

Profile name	Main container keyword	Init container keyword	Sidecar container keyword

Requestrouter	`requestrouter.main`
	For example,
	`netbackup/requestrouter:`*<version>*`-patch`
		`requestrouter.pod-dependency-init`	

Bootstrapper	`bootstrapper.main`
	For example,
	`netbackup/main:`*<version>*`-patch`
		`bootstrapper.pod-dependency-init`	`bootstrapper.fluentbit`

Log-viewer	`log-viewer.main`
	For example,
	`netbackup/log-viewer:`*<version>*`-patch`
		`log-viewer.pod-dependency-init`	

Patching Init containers for all pods of primary server:

Run the following command to patch the pod dependency `init` container of all pods:

```
helm upgrade cloudscale cloudscale-11.2.tgz -n netbackup
--reuse-values --set
environment.primary.serviceImageTag."pod-dependency-init=11.0.1-0100-h"
```

Patching Sidecar containers for all pods of primary server

Use this when you want to patch a specific Sidecar container for all the profiles (wherever applicable) of the primary server.

Sidecar containers keyword:

- fluentbit
- pbx
- secure-comms
- nbhousekeeping

To patch the sidecar containers:

```
helm upgrade --install cloudscale cloudscale-11.2.tgz --reuse-values
--set
environment.primary.serviceImageTag."primary\.fluentbit=11.2-tagged"
```

Patching of media containers

This section describes the procedure for patching of the following media containers:

- Patching for media server
- Patching Init containers for media server
- Patching Sidecar containers for media server:

Patching for media server

Profile name	Main container keyword	Init container keyword	Sidecar container keyword
Media	media.main	media.pod-dependency-init	media.fluentbit

When patching media servers, you must provide details for the entire media server object (since it is a list, not a single object). Ensure that the existing attribute values are not modified unintentionally.

To patch the media container:

```
helm upgrade cloudscale
VRTSk8s-netbackup-11.1.1-0118/helm/cloudscale-11.1.1-118.tgz -n nbux
 --reuse-values \
--set environment.mediaServers[0].name=<existing value>--set
environment.mediaServers[0].nodeSelector.labelKey=<existing value>\
--set environment.mediaServers[0].nodeSelector.labelValue=<existing
 value>--set environment.mediaServers[0].replicas=<existing value>\
--set environment.mediaServers[0].minimumReplicas=<existing
value>--set
environment.mediaServers[0].storage.data.capacity=<existing value>\
--set
environment.mediaServers[0].storage.data.storageClassName=<existing
 value> \
--set environment.mediaServers[0].tag=<existing value> --set
environment.mediaServers[0].storage.log.capacity=<existing value>\
--set
environment.mediaServers[0].storage.log.storageClassName=<existing
value>
--set
environment,mediaServers[0].serviceI.serviceImageTag."media\.main=11.2-tag"
```

Patching Init containers for media server

To patch the Init containers:

```
helm upgrade cloudscale
 VRTSk8s-netbackup-11.1.1-0118/helm/cloudscale-11.1.1-118.tgz-n
nbux--reuse-values \
--set environment.mediaServers[0].name=<existing value> \
--set environment.mediaServers[0].nodeSelector.labelKey=<existing
value>\
--set environment.mediaServers[0].nodeSelector.labelValue=<existing
 value> \
--set environment.mediaServers[0].replicas=<existing value>\
--set environment.mediaServers[0].minimumReplicas=<existing value>
\
--set environment.mediaServers[0].storage.data.capacity=<existing
value>\
--set
environment.mediaServers[0].storage.data.storageClassName=<existingvalue>
 \
--set environment.mediaServers[0].tag=<existing value> \
--set environment.mediaServers[0].storage.log.capacity=<existing
value>\
--set
environment.mediaServers[0].storage.log.storageClassName=<existing
value> \
--set
environment,mediaServers[0].serviceImageTag."media\.pod-dependency-init=11.2"
```

Patching Sidecar containers for media server

To patch the Sidecar containers:

```
helm upgrade cloudscale
 VRTSk8s-netbackup-11.1.1-0118/helm/cloudscale-11.1.1-118.tgz-n
nbux--reuse-values \
--set environment.mediaServers[0].name=<existing value> \
--set environment.mediaServers[0].nodeSelector.labelKey=<existing
value>\
--set environment.mediaServers[0].nodeSelector.labelValue=<existing
 value> \
--set environment.mediaServers[0].replicas=<existing value>\
--set environment.mediaServers[0].minimumReplicas=<existing value>
\
--set environment.mediaServers[0].storage.data.capacity=<existing
value>\
--set
environment.mediaServers[0].storage.data.storageClassName=<existingvalue>
```

```
 \
--set environment.mediaServers[0].tag=<existing value> \
--set environment.mediaServers[0].storage.log.capacity=<existing
value>\
--set
environment.mediaServers[0].storage.log.storageClassName=<existing
value> \
--set
environment,mediaServers[0].serviceImageTag."media\.fluentbit=11.2"
```

Patching of fluentbit collector pods

From version 11.0 and later, you can also patch the fluentbit collector pods using the fluentbit containers. The fluentbit collector, daemonset, and sidecar are the same container images.

Fluentbit container patch image example:

`netbackup/fluentbit:11.2.x.x.xxxx-patch`

Fluentbit cleanup container patch image example:

`netbackup/fluentbit-log-cleanup:11.2.x.x.xxxx-patch`

To patch the fluentbit collector pods

1. Using the following command, get the currently configured values. Save the values in a file.

   ```
   helm get values -n <namespace> cloudscale | grep -v
   "USER-SUPPLIEDVALUES:" > <values file name>
   ```

2. Update the output file to reflect new image tags:

   - Edit and update the tags in below json path file for the fluentbit-collector and execute the command:

     `fluentbit.image.tag to 11.2.x.x.xxxx-patch (or actual patch tag)`

   - Edit and update the tags in below json path file for the fluenbit-log-cleanup image and execute the command:

`fluentbit.cleanup.image.tag to 11.2.x.x.xxxx-patch (or actual patch tag)`

3. To update the deployment, run the helm command:

```
helm upgrade --install -n <namespace> cloudscale <cloudscale
tgz>-f <values file name>
```

Note: The tarball you need for this command is the one containing the helm templates, not the docker images. It is usually named `fluentbit-<version>.tgz`.

Update containerized PostgreSQL pod

The procedure describes the steps to update the image for NetBackup PostgreSQL pod in 11.0 or later.

To update the containerized PostgreSQL pod for NetBackup version 11.0 or later

Consider a scenario - In 11.2.x.x.xxxx deployment with netbackup-postgresql version 16.10.x.x, upgrade to netbackup-postgresql version 16.10.x.x using tag 11.2.x.x.xxxx

1. Suspend the backup job processing using the command:

   `nbpemreq -suspend_scheduling`

2. Load, tag, and push the new NetBackup PostgreSQL image using the commands:

   To load the PostgreSQL image`docker load -i VRTSnetbackup-postgresql-16.4.0002.tar.gz`

   To load the PostgreSQL-upgrade images`docker load -i VRTSnetbackup-postgresql-upgrade-16.4-0002.tar.gz`

   To tag the new images`docker tag netbackup/postgresql:16.4-0002 exampleacr.azurecr.io/netbackup/postgresql:11.2.x.x.xxxx`

   `docker tag localhost/netbackup/postgresql-upgrade:16.4-0002 exampleacr.azurecr.io/netbackup/postgresql-upgrade:11.2.x.x.xxxx_patch2`

   To push the PostgreSQL image`docker push exampleacr.azurecr.io/netbackup/postgresql:11.2.x.x.xxxx`

   `docker push exampleacr.azurecr.io/netbackup/postgresql-upgrade:11.2.x.x.xxxx-patch2`

3 Upgrade the PostgreSQL using Helm chart:

To save PostgreSQL chart values to a file: `helm get values cloudscale -n <namespace> > csvalues.yaml vi csvalues.yaml`

Edit the image tags for postgresql and postgresql-upgrade:

```
vi csvalues.yaml
```

Upgrade the PostgreSQL: `helm upgrade --install cloudscale <cloudscale tgz> -n <namespace> -f csvalues.yaml`

4 Resume the backup job processing using the command:

```
nbpemreq -resume_scheduling
```

Chapter 19

Upgrading

This chapter includes the following topics:

Upgrading Cloud Scale Technology

This section describes a high level procedure for upgrading Cloud Scale Technology.

Table 19-1 Steps for upgrading Cloud Scale Technology

Steps	Description
Step 1	Ensure that all the prerequisites are met as mentioned in the following section: See "Prerequisites for Cloud Scale Technology upgrade" on page 267.
Step 2	Upgrade the following add-ons: ■ cert-manager ■ trust-manager

Table 19-1 Steps for upgrading Cloud Scale Technology *(continued)*

Steps	Description

Step 3
 	
(*Applicable only when using non-official Veritas registry*)
	Upload the new images to your private registry.
	
	■ Extract contents of `VRTSk8s-netbackup-<version>.tar` file as follows:
	`sudo mkdir /path/to/extracted`
	`sudo chmod -R 777 /path/to/extracted`
	`tar -xvf /path/to/VRTSk8s-netbackup-<version>.tar -C /path/to/extracted`
	`cd /path/to/extracted/VRTSk8s-netbackup-<version>/scripts`
	■ Run one of the following command (update the registry/build information only):
	`REGISTRY='XXXXXXXXXX.dkr.ecr.us-east-1.amazonaws.com' PKG_ROOT='/path/to/extracted/VRTSk8s-netbackup-<version>' ./tag_push_images.sh`
	`REGISTRY='XXXXXXXXXX.azurecr.io' TAR_FILE='/path/to/VRTSk8s-netbackup-<version>.tar' ./tag_push_images.sh`
	For example,
	`REGISTRY='XXXXXXXXXX.azurecr.io' TAR_FILE='/home/azureuser/VRTSk8s-netbackup-11.2.x.x.xxxx.tar' ./tag_push_images.sh`
	
	For more information, refer to the following section:
	
	See "Loading docker images" on page 88.

Step 4	Log into the primary server and use the following command to suspend the backup job processing:
	
	`kubectl exec -it pod/<primary-pod-name> -n netbackup -- bash`
	
	`nbpemreq -suspend_scheduling`

Table 19-1 Steps for upgrading Cloud Scale Technology *(continued)*

Steps	Description

Step 5
(*Applicable only for upgrade of DBaaS 10.4*)
	Upgrade PostgreSQL DBaaS version from 14 to 16 for Azure/AWS:
	For Azure: Execute the `kubectl` command into 10.4 primary pod and create the `/tmp/grant_admin_option_to_roles.sql` file.
	Execute the following command to execute `grant_admin_option_to_roles.sql` file:
	`/usr/openv/db/bin/psql "host=$(< /tmp/.nb-pgdb/dbserver) port=$(< /tmp/.nb-pgdb/dbport) dbname=NBDB user=$(< /tmp/.nb-pgdb/dbadminlogin) password=$(< /tmp/.nb-pgdb/dbadminpassword) sslmode=verify-full sslrootcert='/tmp/.db-cert/dbcertpem'" -f /tmp/grant_admin_option_to_roles.sql`
	`/*`
	`Azure PostgreSQL upgrade from 14 to 16 does not grant the NetBackup database administrator role the ADMIN OPTION for NetBackup roles.`
	`This script will grant the NetBackup database administrator role the ADMIN OPTION so that it can manage NetBackup roles.`
	`*/`
	
	`GRANT ADTR_MAIN TO current_user WITH ADMIN OPTION;`
	`GRANT AUTH_MAIN TO current_user WITH ADMIN OPTION;`
	`GRANT DARS_MAIN TO current_user WITH ADMIN OPTION;`
	`GRANT DBM_MAIN TO current_user WITH ADMIN OPTION;`
	`GRANT EMM_MAIN TO current_user WITH ADMIN OPTION;`
	`GRANT JOBD_MAIN TO current_user WITH ADMIN OPTION;`
	`GRANT PEM_MAIN TO current_user WITH ADMIN OPTION;`
	`GRANT RB_MAIN TO current_user WITH ADMIN OPTION;`
	`GRANT SLP_MAIN TO current_user WITH ADMIN OPTION;`
	`GRANT NBPGBOUNCER TO current_user WITH ADMIN OPTION;`
	`GRANT NBWEBSVC TO current_user WITH ADMIN OPTION;`
	`GRANT AZ_DBA TO current_user WITH  ADMIN OPTION;`
	Exit 10.4 primary pod. Ready for 10.4 with PostgreSQL to 10.5 with PostgreSQL 16 upgrade.
	Upgrade Azure PostgreSQL version from 14 to 16 using Azure portal.

Table 19-1 Steps for upgrading Cloud Scale Technology *(continued)*

Steps	Description
	For AWS: Upgrade AWS PostgreSQL RDS version from 14 to 16 using AWS Management Console. Navigate to RDS page, select the database instance and click **Modify** to change the engine version. For more information, see Upgrading the PostgreSQL DB engine for Amazon RDS.

Table 19-1 Steps for upgrading Cloud Scale Technology *(continued)*

Steps	Description

Step 6
(*This step is applicable only when upgrading from version 10.3 and 10.3.0.1*)
	Perform the following (AWS/Azure) to create Secret containing DBaaS CA certificates:
	For AWS:
	`TLS_FILE_NAME='/tmp/tls.crt'`
	`PROXY_FILE_NAME='/tmp/proxy.pem'`
	
	`rm -f ${TLS_FILE_NAME} ${PROXY_FILE_NAME}`
	
	`DB_CERT_URL="https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem"`
	`DB_PROXY_CERT_URL="https://www.amazontrust.com/repository/AmazonRootCA1.pem"`
	
	`curl ${DB_CERT_URL} --output ${TLS_FILE_NAME}`
	`curl ${DB_PROXY_CERT_URL} --output ${PROXY_FILE_NAME}`
	
	`cat ${PROXY_FILE_NAME} >> ${TLS_FILE_NAME}`
	
	`kubectl -n netbackup create secret generic postgresql-netbackup-ca --from-file ${TLS_FILE_NAME}`

Table 19-1 Steps for upgrading Cloud Scale Technology *(continued)*

Steps	Description

	For Azure:
	
	`DIGICERT_ROOT_CA='/tmp/root_ca.pem'`
	`DIGICERT_ROOT_G2='/tmp/root_g2.pem'`
	`MS_ROOT_CRT='/tmp/ms_root.crt'`
	`COMBINED_CRT_PEM='/tmp/tls.crt'`
	
	`DIGICERT_ROOT_CA_URL="https://cacerts.digicert.com/DigiCertGlobalRootCA.crt.pem";`
	`DIGICERT_ROOT_G2_URL="https://cacerts.digicert.com/DigiCertGlobalRootG2.crt.pem";`
	`MS_ROOT_CRT_URL="http://www.microsoft.com/pkiops/certs/Microsoft%20RSA%20Root%20Certificate%20Authority%202017.crt";`
	
	`curl ${DIGICERT_ROOT_CA_URL} --output ${DIGICERT_ROOT_CA}`
	`curl ${DIGICERT_ROOT_G2_URL} --output ${DIGICERT_ROOT_G2}`
	`curl ${MS_ROOT_CRT_URL} --output ${MS_ROOT_CRT}`
	
	`openssl x509 -inform DER -in ${MS_ROOT_CRT} -out ${COMBINED_CRT_PEM} -outform PEM`
	`cat ${DIGICERT_ROOT_CA} ${DIGICERT_ROOT_G2} >> ${COMBINED_CRT_PEM}`
	
	`kubectl -n netbackup create secret generic postgresql-netbackup-ca --from-file ${COMBINED_CRT_PEM}`

Step 7	Create db-cert bundle if it does not exists.
Step 8	Upgrade Cloud Scale. See “Upgrade Cloud Scale using the kubectl plugin” on page 270.

Prerequisites for Cloud Scale Technology upgrade

Before upgrading Cloud Scale deployment, ensure that:

- NetBackup Cloud Scale does not depend on Kubernetes versions.
  For more information on upgrading the cluster hosting Cloud Scale, see the following section:
  See “Upgrade the cluster” on page 268.
- Ensure that all the prerequisites mentioned in the *Prerequisites* chapter are met.
- Ensure that catalog backup is done.
- Take snapshots of persistent volumes.

- Download the latest patch version.
- Push the container images to registry.
  See "Loading docker images" on page 88.
- Helm charts for operators and Postgres are available from a public or private registry.
- Images for operators, and Cloud Scale services are available from a public or private registry.

Note: During the upgrade process, ensure that the cluster nodes are not scaled down to 0 or restarted.

Health checker for pre-upgrade

Run pre-upgrade health checks to identify compatibility or configuration issues before upgrading an existing Cloud Scale deployment (supported for Cloud Scale 10.5 and later).

For more information, See "Overview" on page 194.

Upgrade the cluster

Before upgrading the kubernetes cluster, ensure that the NetBackup Catalog is successfully backed up on cloud LSU (MSDP-C).

To upgrade AKS or EKS cluster hosting Cloud Scale

1. Pause NetBackup job scheduling and allow all running jobs to be completed.
2. Stop the primary services as follows:
   - For NetBackup version 10.5 and above, run the following script with a stop option:
     `/VRTSk8s-netbackup-<version>/scripts$ ./cloudscale_restart.sh stop`
   - For NetBackup version lower than 10.5:
     - Get the environment name using the following command:
       `kubectl get environment -n <namespace>`
     - Pause using the following command:
       `kubectl patch environment <env_name> -n <namespace>--type=merge -p='{"spec": {"primary": {"paused": true}}}`
   - Scale down the primary pod to 0 as follows:

Get primary sts name:

```
kubectl get sts -n <namespace> | grep "\-primary"
```

Scale to 0:

```
kubectl scale --replicas=0 sts <sts name> -n <namespace>
```

3. Upgrade cluster version from AKS/EKS portal.
4. For EKS, update the add-ons by going to the add-on section (new defaults which are shown).

   If any problem occurs during the add-on upgrade, use the following command if there is no customization made as part of the add-on upgrade:

   ```
   aws eks update-addon \ --cluster-name <cluster_name> \
   --addon-name <addon_name> \ --addon-version <addon_version> \
   --resolve-conflicts OVERWRITE
   ```

5. Upgrade all node pools version. This can be done in parallel with a console.
6. For multiple cluster version upgrades, repeat steps 3 to 5.
7. Once the cluster, the add-on and the node pool are upgraded to the required version, perform the following:
   - For NetBackup version 10.5 and above: run the following command to restart:

     ```
     ./cloudscale_restart.sh start
     ```

   - For NetBackup version lower than 10.5:

     ```
     kubectl patch environment <env_name> -n <namespace>--type=merge
     -p='{"spec": {"primary": {"paused": true}}}
     ```

8. Resume NetBackup job scheduling.

Upgrade Cloud Scale

This section describes the available upgrade methods based on user requirements:

- Upgrade Cloud Scale using the kubectl plugin
- Manual upgrade of Cloud Scale using the Superchart

Note: When upgrading from Cloud Scale version 10.4 or earlier to version 11.1 or later, the deployment must first be manually upgraded to version 10.5 or 11.0. The final upgrade to version 11.1 or later must be completed exclusively via the kubectl plugin.

Upgrade Cloud Scale using the kubectl plugin

Note the following:

- During upgrade ensure that the value of minimumReplica of media server CR is same as that of media server before upgrade.
- Upgrading a Cloud Scale deployment using the `kubectl` plugin is supported for version 10.5 or later.
- By default, **fluentbit.enableSidecarsLogging** is set to **false**. As a result, only stdout logs will be collected and available for download unless the flag is manually enabled.

Upgrading Cloud Scale deployment using kubectl plugin

1. Before you proceed with upgrade, ensure that the following prerequisites are met:
   - Infrastructure readiness: The Kubernetes cluster and Cloud Scale environment is up and running.
   - Required container images: All Cloud Scale related images of the version you would like to upgrade to are pushed to your container registry.
   - Helm setup:
     - Helm is installed and configured.
     - The `jetstack` repository is added:`helm repo add jetstack https://charts.jetstack.io`
     - The cert-manager and trust-manager charts are installed via helm.

   Also review the prerequisites in the following section:

See "Prerequisites for Cloud Scale Technology upgrade" on page 267.

2 Execute the binary at `bin/` with the upgrade option using the following command:

```
./kubectl-cloudscale upgrade
```

3 Follow the upgrade steps as follows:

```
/root/VRTSk8s-netbackup-11.2.0.2-0019/bin/kubectl-cloudscale
upgrade Cloud Scale Upgrade
        Before you proceed with installation, please ensure the
following prerequisites are in place:
        1. Infrastructure readiness:
           - The Kubernetes cluster is up and running
           - Cloud Scale environment is up and running
        2. Required container images:
           - All Cloud Scale related images of the version you
would like to upgrade to are pushed to your container registry
        3. Helm setup:
           - Helm is installed and configured
           - The "jetstack" repository is added:
                  helm repo add jetstack
https://charts.jetstack.io
    - The cert-manager and trust-manager charts are installed via
 helm
 4. Container registry is logged in to this machine
 Also review the 'Prerequisites for Cloud Scale Technology
upgrade' section in the 'Cohesity Cloud Scale Technology Manual
Deployment Guide for Kubernetes Clusters' document for additional
 required steps.
        Once everything is ready, you can safely continue with
the upgrade.
Would you like to continue? (y/n): y
Checking if the input file already exists.
Input file is not present.
Provide the file path of the extracted Cloud Scale folder:
/root/VRTSk8s-netbackup-11.2.0.2-0019/

Provide a namespace for the Environment: netbackup Environment
is ready for upgrade

*******************************Checking for
cert-manager******************************
Checking if Cert Manager is installed or not Cert-manager is
installed with version 'v1.13.3'. It is recommended to upgrade
cert-manager to version v1.18.2 post Cloud Scale deployment

*******************************Checking for
trust-manager******************************
```

```
Checking if Trust Manager is installed or not Trust-manager is
installed with version ''v0.7.0''. It is recommended to upgrade
trust-manager to version v0.19.0 post Cloud Scale deployment

Provide the image tag for the NetBackup images that need to be
upgraded: 11.2.0.2-0019
Validating images...

Provide the image tag for the MSDP images that need to be
upgraded: 21.1.0.2-0013
Validating images...

Provide the image tag for the NetBackup Snapshot Manager images
that need to be upgraded: 11.2.0.2-3006
Validating images...

Operator Namespace: netbackup-operator-system
Upgrade of operators started...
The operators chart has been installed successfully.

Provide the image tag for the PostgreSQL images that need to be
upgraded: 16.11.2.0-0001
Validating images...

Current version of cloudscale: 11.1

Starting upgrade of Helm-based CloudScale environment in namespace
 netbackup
Successfully upgraded CloudScale Helm release to version 4
Successfully initiated the upgrade for Helm-based CloudScale
environment in netbackup namespace.
Ensure that all pods are up and running before using the
NetBackup.
For kubectl plugin specific logs, check the log at
/root/.cloudscale/setup-cloudscale.log
```

Post upgrade

- Post upgrade, the flexsnap-listener pod would be migrated to cp control nodepool as per the node selector settings in the environment CR. To reduce the TCO, user can change the minimum size of CP data nodepool to 0 through the portal.

- Post upgrade, for cost optimization, user has the option to change the value of **minimumReplica** of media server CR to 0. User can change the minimum size of media nodepool to 0 through the portal.
- (*For Azure*) Post upgrade, patch StorageClass and PV with `nconnect=4` for better NFS performance.
  The `nconnect` mount option in Azure is a client-side NFS feature that enables multiple TCP connections between an NFS client and the NFS endpoint. This improves large-scale performance by reducing the number of clients needed to reach the maximum bandwidth of high-capacity SSD file shares. Azure Files supports up to 16 nconnect channels, with `nconnect=4` recommended for optimal performance. For this reason, it is recommended to set `nconnect=4` in the StorageClass specification for Cloud Scale.

To patch StorageClass and PV with `nconnect=4` for better NFS performance

1. Patch the StorageClass (affects newly created PVs):
   - Identify the StorageClass used by the NFS-mounted volume.
     Use the following command to list the PVCs in the netbackup namespace:
     ```
     $ kubectl get pvc -n netbackup
     ```
     Example output:
     ```
     NAME                                                    STATUS   VOLUME
                                                  CAPACITY   ACCESS MODES
       STORAGECLASS       VOLUMEATTRIBUTESCLASS
     AGEcatalog-nbu-primary-0                                  Bound
     pvc-37f2d9be-84e8-4e6c-8463-dceb16642370   110Gi      RWX
             nb-file-premium   <unset>
     19dcertauth-pvc                                           Bound
     pvc-defd5f28-0be5-4e70-8453-0e2472bfca06   1Gi        RWO
             nb-disk-premium   <unset>
     19dcloudpoint-pvc                                         Bound
     pvc-e910737e-4d69-4e0b-8fc5-d57e5787eace   5Gi        RWX
             nb-file-premium   <unset>
     19ddata-flexsnap-rabbitmq-0                               Bound
     pvc-794271f2-5440-47b8-b285-2929b0ab6669   5Gi        RWO
             nb-disk-premium   <unset>                 19d
     ```
     From this output, the PVC **catalog-nbu-primary-0** uses the StorageClass `nb-file-premium`, so this is the StorageClass we will patch.
   - Patch the StorageClass.
     Apply the `nconnect=4` mount option:
     ```
     kubectl patch storageclass nb-file-premium --type merge -p
     '{"mountOptions":["nconnect=4"]}'
     ```

The following content must be displayed:

```
storageclass.storage.k8s.io/nb-file-premium patched
```

- Verify the StorageClass patch.

  ```
  kubectl get sc nb-file-premium -o yaml | sed -n
  '/mountOptions/,$p'
  ```

  The following content must be displayed:

  ```
  mountOptions:
  - nconnect=4
  ```

2 Patch the existing PV (applies to currently mounted volumes):

- Use the following command to export the PV name:

  ```
  export PV=pvc-37f2d9be-84e8-4e6c-8463-dceb16642370
  ```

- Use the following command to check whether the PV already has the mount option:

  ```
  kubectl get pv "$PV" -o json | jq -e '.spec.mountOptions //
  [] | index("nconnect=4")'
  ```

  If output is **null**, the option is not present.

- Use the following command to patch the PV:

  ```
  kubectl patch pv "$PV" --type merge -p '{"spec":
  {"mountOptions": ["nconnect=4"]}}'
  ```

  Example output:

  ```
  persistentvolume/<PV_NAME> patched
  ```

3 Restart pods to apply the new NFS mount option:

Use the following command to delete each MediaServer replica pod:

```
export POD=media1-media-0 kubectl delete pod $POD -n netbackup
```

The pod will restart and re-mount with the updated NFS settings.

4 Verify `nconnect=4` inside the pod:

Once the pod has restarted, run:

```
mount | grep nfs
```

Example output:

```
... nconnect=4 ...
```

5 Apply changes to primary nodes:

- Scale down the Primary nodepool to 0: This forces NFS unmounting. Then scale it back up so that all primary pods remount the storage with the updated `nconnect=4` option.
- Verify from primary pods:

  `mount | grep nfs`

  Example output:

  ```
  ... nconnect=4 ...
  ```

6. Verify from all other related pods:

   For example, `kubectl exec -it nbu-nbatd-0 -n netbackup -- mount | grep nfs`

   The `nconnect=4` must be visible on on all relevant NFS mount paths.

Manual upgrade of Cloud Scale using the Superchart

The following steps are applicable for performing a manual upgrade from Cloud Scale version 11.1 to later versions (11.1.0.2 and above) using Superchart.

Upgrading Cloud Scale deployment using Superchart

1. Deactivate all policy and suspend the scheduling.
2. Run the following command and check the operators release version:

   ```
   helm ls -n netbackup-operator-system
   ```

3. Upload images for operators and Cloud Scale services to a public or private registry.

4 Upgrade operators by specifying the new tags.

Run the helm command from the following location:

`/VRTSk8s-netbackup-<version>/helm/`

Extract the `operators-values.yaml` file from the Helm package, make modifications, and upgrade the operators.

Run the following command with `--reuse-values` option for upgrading the operators:

```
helm upgrade operators operators-<version>.tgz -n
netbackup-operator-system --reuse-values

--set msdp-operator.image.tag=<msdp image tag>

--set nb-operator.image.tag=<primary image tag>

--set nb-operator.msdp-operator.image.tag=<msdp image tag>

--set nb-operator.flexsnap-operator.image.tag=<nbsm image tag>

--set flexsnap-operator.image.tag=<NBSM image tag>
```

5 Use the following command to verify the operator pod status:

```
kubectl get all -n netbackup-operator-system
```

6 Depending on the environment, execute the following steps to upgrade the Cloud Scale deployment (fluentbit, postgresql and the environment):

For non-DBaaS environment: If the service image tag exists in version 11.1, perform the following steps for the PrimaryServer pods.

- Run the following command:
  ```
  helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup
  --reuse-values --set environment.primary.paused=true
  ```
- Remove the serviceImagetag section by using the `kubectl edit` command.
- Run the following command:
  ```
  helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup
  --reuse-values --set environment.primary.tag=new-tag --set
  environment.primary.paused=false
  ```

For example:

```
 helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup
--reuse-values \
                --set fluentbit.image.tag=<primary image tag>
--set fluentbit.cleanup.image.tag=<primary image tag>  \
                --set postgresql.image.tag=<postgres image tag>
```

```
--set postgresql.postgresqlUpgrade.image.tag=<postgres image tag>
 \
               --set environment.primary.tag=<primary image tag>
  \
                --set environment.mediaServers[0].name=media1 \
                --set environment.mediaServers[0].replicas=1 \
                --set
environment.mediaServers[0].nodeSelector.labelKey=agentpool  \
                --set
environment.mediaServers[0].nodeSelector.labelValue=mediapool  \

                --set
environment.mediaServers[0].storage.data.capacity=50Gi \
                --set
environment.mediaServers[0].storage.data.storageClassName=managed-csi-hdd
 \
                --set
environment.mediaServers[0].storage.log.capacity=30Gi  \
                --set
environment.mediaServers[0].storage.log.storageClassName=managed-csi-hdd
  \
                --set environment.mediaServers[0].tag=<primary
image tag>  \
                --set environment.msdpScaleouts.tag=<MSDP image
tag> \
                --set environment.cpServer.tag=<NBSM image tag>
```

For DBaaS environment: If the service image tag exists in version 11.1, perform the following steps for the PrimaryServer pods.

- Run the following command:

  `helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup --reuse-values --set environment.primary.paused=true`

- Remove the serviceImagetag section by using the `kubectl edit` command.

- Run the following command:

  `helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup --reuse-values --set environment.primary.tag=new-tag --set environment.primary.paused=false`

For example:

```
helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup
--reuse-values  \
                --set fluentbit.image.tag=<primary image tag>
```

```
--set fluentbit.cleanup.image.tag=<primary image tag>  \
               --set environment.primary.tag=<primary image tag>
  \
                --set environment.mediaServers[0].name=media1 \
                --set environment.mediaServers[0].replicas=1 \
                --set
environment.mediaServers[0].nodeSelector.labelKey=agentpool  \
                --set
environment.mediaServers[0].nodeSelector.labelValue=mediapool  \

                --set
environment.mediaServers[0].storage.data.capacity=50Gi \
                --set
environment.mediaServers[0].storage.data.storageClassName=managed-csi-hdd
 \
                --set
environment.mediaServers[0].storage.log.capacity=30Gi  \
                --set
environment.mediaServers[0].storage.log.storageClassName=managed-csi-hdd
  \
                --set environment.mediaServers[0].tag=<primary
image tag>  \
                --set environment.msdpScaleouts.tag=<MSDP image
tag>r \
                --set environment.cpServer.tag=<NBSM image tag>
```

7 Wait for the Environment CR to reach the **Ready** status. During this period, pods may restart and new services may start.

```
kubectl get environment -n namespace
```

8 Log into the primary server using the `kubectl exec -it pod/,primary-podname> -n netbackup -- bash` command and resume the backup job processing by using the following command:

```
nbpemreq -resume_scheduling
```

Configuring NetBackup IT Analytics for NetBackup deployment

The NetBackup IT Analytics configuration is applicable only for users who are upgrading from NetBackup version 10.2.

We can configure data collector on the primary server pod or on a separate host machine. Following are the steps for respective configurations.

Note: From NetBackup version 10.3 or later, Cloud Scale release data collector on primary server pod is supported.

Configuring NetBackup IT Analytics for NetBackup deployment by configuring data collector on a primary server pod

Configure data collector on the primary server pod.

To configure NetBackup IT Analytics for NetBackup deployment

1. Create DNS server entry in such a way that single IP must be resolvable to two FQDNs.

   IP: 1.2.3.4 must be resolved to the following FQDNs:

   ```
   itanalyticsportal.<yourdomain>
   itanalyticsagent.<yourdomain>
   ```

   Note the following:

   - If the IT Analytics Portal URL is `itanalyticsportal.<yourdomain>`, then ensure to add the DNS entries for the following hostnames:

     ```
     itanalyticsportal.<yourdomain>
     itanalyticsagent.<yourdomain>
     ```

   - If the IT Analytics Portal URL is `aptareportal.<yourdomain>`, then ensure to add the DNS entries for the following hostnames:

     ```
     aptareportal.<yourdomain>
     aptareagent.<yourdomain>
     ```

2. Collect the `<your-collector-name>.key` file for the new data collector by accessing the IT Analytics portal link and creating a collector and copy it to the host machine from where Kubernetes cluster is accessed.

   For more information, refer to the NetBackup IT Analytics User Guide.

3. Execute the following command in the primary server pod:

   ```
   kubectl exec -it -n <namespace> <primaryServer-pod-name> -- bash
   ```

4. In case if the data-receiver is configured with self-signed certificate (https). User must add the certificate in the data collector.

   For more information, refer to the Configure the Data Collector to trust the certificate.

5 Create a new folder **analyticscollector** at persisted location (for example, `/mnt/nbdata/`) using the following commands:

```
cd "/mnt/nbdata/"
mkdir analyticscollector
```

6 Exit from the container and copy the `<your-collector-name>.key` file to `/mnt/nbdata/analyticscollector` using the `kubectl cp <keyfile-name> <namespace>/<primary-pod-name>:/mnt/nbdata/analyticscollector` command.

7 Execute the following command to exec into the primary server pod:

```
kubectl exec -it -n <namespace> <primaryServer-pod-name> -- bash
```

8 Navigate to `/usr/openv/analyticscollector/installer/` location and perform the following:

- Open the `responsefile.sample` and add the following parameters:

```
Configuration for reference:
NetBackup.docx COLLECTOR_NAME=<your-collector-name>
COLLECTOR_PASSCODE=<your-password>
DR_URL=<http>/<https>://itanalyticsagent.<yourdomain>
COLLECTOR_KEY_PATH=/mnt/nbdata/analyticscollector/<your-collector-name>.key
HTTP_PROXY_CONF=N
HTTP_PROXY_ADDRESS=
HTTP_PROXY_PORT=
HTTPS_PROXY_ADDRESS=
HTTPS_PROXY_PORT=
PROXY_USERNAME=
PROXY_PASSWORD=
PROXY_EXCLUDE=
```

- Run `./dc_installer.sh -c /usr/openv/analyticscollector/installer/responsefile.sample` command to connect data collector with IT Analytics portal

9 Validate data collector integration with IT Analytics by performing the following:

- Navigate to `/usr/openv/analyticscollector/mbs/bin/` location.
- Run the following command:
  `./checkinstall.sh`

If data collector is configured with portal, it will display as **SUCCESSFUL**.

10 Check the data collector services status by running the following command and ensure that the following data collector services are up and running:

```
/usr/openv/analyticscollector/mbs/bin/aptare_agent status
```

Output of the above command:

```
IT Analytics WatchDog is running (pid: 13312).
IT Analytics MetaDataCollector is stopped.
IT Analytics EventDataCollector is stopped.
IT Analytics DataCollector process is running (pid: 13461).
IT Analytics On-demand process is running (pid: 13463).
IT Analytics Message Relay Server process is running (pid: 13471)
```

For more information about IT Analytics data collector policy, see NetBackup IT Analytics User Guide.

Cloud Scale Disaster Recovery

This chapter includes the following topics:

Cluster backup

AKS cluster

Once NetBackup is up and running and the basic sanity test is done to ensure that the cluster is in a working state, then save the template of the cluster through CLI as follows:

1. Ensure that you are logged in through Azure CLI with subscription set to where the cluster is present:

   `az group export --resource-group <resource_group> --resource-ids <resourceID> --include-parameter-default-value`

   Or

   `az group export --resource-group <resource_group> --resource-ids /subscriptions/${subscription_id}/`

```
resourceGroups/$resource_group/providers/Microsoft.ContainerService/
managedClusters/$cluster_name --include-parameter-default-value
```

Here,

Parameter	Description
resource_group	Resource group where the cluster is present.
resourceID	Cluster's resource ID. This can be obtained from the azure portal by navigating to the cluster and clicking on the JSON link present on the top corner of the Overview.
subscription_id	Subscription ID where cluster is present.
cluster_name	Cluster name that needs to be saved.

1. Copy the JSON output to a file and save it as a JSON file. For example, `template.json`
2. Ensure that `template.json` file is saved and is safe. This is required later to recover the cluster.
3. Note the Azure Container Registry name which is attached to the cluster.

EKS cluster

Note: User must save the output of every command in different file.

1. Get the cluster details.
   - Use the following command to obtain the information of the cluster:
     `aws eks describe-cluster --name <cluster name>`
   - Obtain the detailed information about each subnet which was received as output of the above command:
     `aws ec2 describe-subnets --subnet-ids <subnetID-1> <subnetID-2>`
   - Obtain the IAM role name from `describe-cluster` command and using the following command save the policies attached to it:
     `aws iam list-attached-role-policies --role-name <role name>`
   - Obtain the security group ID from `describe-cluster` command and using following command save the details of security-groups:
     `aws ec2 describe-security-groups --group-ids <Security group ID>`
2. Get the nodegroup details.

- Use the following command to get the list of all node groups:

  ```
  aws eks list-nodegroups --cluster-name <cluster-name>
  ```

- Obtain the information of a particular nodegroup from above list using the following command:

  ```
  aws eks describe-nodegroup --nodegroup-name <nodegroup-name>
  --cluster-name <cluster-name>
  ```

  User must execute this command for each node group in the cluster to obtain the details of each nodegroup.

3. Get the EFS ID details.

   NetBackup uses two EFS IDs (Primary and Snapshot Manager server):

   Get EFS ID of Primary server: spec > primary > storage > catalog > storageClassName

   Get EFS ID of Snapshot Manager server: Get storage class name from spec > cpserver > storage > log > storageClassName.

   Describe this storage class using the following command and note the fileSystemId:

   ```
   kubectl describe sc storageClassName
   ```

   Use following commands to describe EFS:

   ```
   aws efs describe-file-systems --file-system-id <EFS ID>
   ```

   Run the above EFS command for Primary and Snapshot Manager server EFS ID. The above EFS command does not describe the mount targets which was used while creating EFS. To get mount targets details use the following command:

   ```
   aws efs describe-mount-targets --file-system-id <EFS ID>
   ```

   Note the non-default parameters with which EFS was created.

4. Get the list of Add-on by using the following command:

   ```
   aws eks list-addons --cluster-name <cluster-name>
   ```

Environment backup

1. Note down the MSDP operator Namespace, NodeSelector, StorageClassName, Tolerations and Image tag as follows:

   Obtain the name of the msdp operator statefulset using the following command:

   ```
   kubectl get statefulset -n <msdp-operator-system-namespace>
   ```

Use the following command to backup MSDP operator Image tag, Tolerations, and NodeSelector:

```
kubectl get sts <msdp-operator-statefulset-name> -n
<msdp-operator-sample-namespace> -o=jsonpath='{"Namespace
:"}{$.metadata.namespace}{$"\nImage
:"}{$.spec.template.spec.containers[0].image}{$"\nNodeSelector
:"}{$.spec.template.spec.nodeSelector}{$"\nTolerations
:"}{$.spec.template.spec.tolerations[2]}{$"\nStorageClassName
:"}{$.spec.volumeClaimTemplates[0].spec.storageClassName}{$"\n"}'
```

From the output, note down the Image tag, StorageClassName, Tolerations and NodeSelector:

```
Sample Output:
Namespace :msdp-operator-system
Image :nbuk8sreg.azurecr.io/msdp-operator:21.0
NodeSelector :{"agentpool":"nbuxpool"}
Tolerations
:{"key":"agentpool","operator":"Equal","value":"nbuxpool"}
StorageClassName :nb-disk-premium
```

If toleration is not provided for msdp operator, then use the following command:

```
kubectl get sts <msdp-operator-statefulset-name> -n
<msdp-operator-sample-namespace> -o=jsonpath='{"Namespace
:"}{$.metadata.namespace}{$"\nImage
:"}{$.spec.template.spec.containers[0].image}{$"\nNodeSelector
:"}{$.spec.template.spec.nodeSelector}{$"\nStorageClassName
:"}{$.spec.volumeClaimTemplates[0].spec.storageClassName}{$"\n"}'
```

```
Sample Output:
Namespace :msdp-operator-system
Image :nbuk8sreg.azurecr.io/msdp-operator:21.0
NodeSelector :{"agentpool":"nbuxpool"}
StorageClassName :nb-disk-premium
```

2. Backup the above msdp-operator storageClass using the following command:

```
kubectl get sc <msdp-operator-storageclass-name> -o yaml >
msdpopstorageclass_backup.yaml
```

3. Note down the NetBackup operator Namespace, NodeSelector, Tolerations and Image tag as follows:

   Obtain the name of the NetBackup operator deployment using the following command:

```
kubectl get deployment -n <netbackup-operator-system-namespace>
```

Use the following command to backup NetBackup operator Image tag, Tolerations, and NodeSelector:

```
kubectl get deployment <netbackup-operator-deployment-name> -n
<netbackup-operator-system-namespace> -o=jsonpath='{"Namespace
:"}{$.metadata.namespace}{$"\nImage
:"}{$.spec.template.spec.containers[0].image}{$"\nNodeSelector
:"}{$.spec.template.spec.nodeSelector}{$"\nTolerations:
"}{$.spec.template.spec.tolerations}{$"\n"}'
```

From the output, note down the Image tag, Tolerations and NodeSelector:

```
Sample Output:
Namespace :netbackup-operator-system
Image :nbuk8sreg.azurecr.io/netbackup/operator:11.2.x.x.xxxx
NodeSelector :{"agentpool":"agentpool"}
Tolerations:
[{"key":"agentpool","operator":"Equal","value":"agentpool"}]
```

4. Note down the flexsnap-operator Namespace, NodeSelector, Tolerations and Image tag as follows:

   Obtain the name of the flexsnap-operator deployment using the following command:

   ```
   kubectl get deployment -n <netbackup-operator-system-namespace>
   ```

   Use the following command to backup flexsnap operator Image tag, Tolerations, and NodeSelector:

   ```
   kubectl get deployment <flexsnap-operator-deployment-name> -n
   <netbackup-operator-system-namespace> -o=jsonpath='{"Namespace
   :"}{$.metadata.namespace}{$"\nImage
   :"}{$.spec.template.spec.containers[0].image}{$"\nNodeSelector
   :"}{$.spec.template.spec.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms[0].matchExpressions[0]}{$"\nTolerations
    :"}{$.spec.template.spec.tolerations}{$"\n"}'
   ```

   From the output, note down the Image tag, Tolerations and NodeSelector:

   ```
   Sample Output:
   Namespace :netbackup-operator-system
   Image :nbuk8sreg.azurecr.io/veritas/flexsnap-deploy:11.2.x.x.xxxx
   NodeSelector
   :{"key":"agentpool","operator":"In","values":["agentpool"]}
   ```

```
Tolerations
:[{"effect":"NoSchedule","key":"agentpool","operator":"Equal","value":"agentpool"}]
```

5. (*For DBaaS*) Note the FQDN of the Postgres server created.
6. (*Applicable only if containerized DB is used*) Note the Postgres unified container image tag, containerPort:

   ```
   kubectl get statefulset.apps/nb-postgresql -n <sample-namespace>
   -o=jsonpath='{$"\nImage
   :"}{$.spec.template.spec.containers[0].image}{$"\ncontainerPort
   :"}{$.spec.template.spec.containers[0].ports[0].containerPort}{$"\n"}'
   ```

   Sample output:

   ```
   Image
   :cpautomation.azurecr.io/netbackup/postgresql:16.10.1.0-0001-DR1


   containerPort :13787
   ```

7. Obtain the fluentbit image tags and nodeselector using the following command:

   ```
   kubectl get deployment.apps/nb-fluentbit-collector -n netbackup
   -o=jsonpath='{$"\nImage
   :"}{$.spec.template.spec.containers[0].image}{$"\nImage2
   :"}{$.spec.template.spec.containers[1].image}{"\n"}'
   ```

   Sample output:

   ```
   Image1:cpautomation.azurecr.io/netbackup/fluentbit:11.2.x-xxxx
   Image2:cpautomation.azurecr.io/netbackup/fluentbit-log-cleanup:11.2.x-xxxx
   ```

8.
   - Take backup of `operator-values.yaml` file using the following command:

     ```
     helm get values operators -n netbackup-operator-system >
     operator-values.yaml
     ```

     Or

     Save the `operator-values.yaml` file which was used during operator helm chart installation.
   - Take backup of `cloudscale-values.yaml` file using the following command:

     ```
     helm get values cloudscale -n netbackup >
     cloudscale-values.yaml
     ```

     Or

     Save the `cloudscale-values.yaml` file that is generated during installation using the `kubectl-plugin`.

9. Note down the values of spec, cpServer, storage, log, storageClassName from the `cloudscale-values.yaml` file using the following command:

   ```
   kubectl get sc nb-file-premium -o yaml >
   CPServerLog_storageclass_backup.yaml
   ```

10. Search the storageClassName in `cloudscale-values.yaml` file using the `kubectl get sc nb-disk-standardssd -o yaml > storageclass_backup.yaml` command and provide these storageclasses name in the following command:

    For example, `nb-disk-standardssd`

    ```
    kubectl get sc <storageclass-name1> <storageclass-name2>
    <storageclass-name3> -o yaml > storageclass_backup.yaml
    ```

11. Note down and save the required values (names) of the secrets obtained from `cloudscale-values.yaml` file in the above step:

    For example, `credSecretName: primary-credential-secret`

    Save the secrets yaml file as follows:

    ```
    kubectl get secret <secret-name1> <secret-name2> <secret-name3>
    -n <sample-namespace> -o yaml > secret_backup.yaml
    ```

    For example, `kubectl get secret primary-credential-secret kms-secret example-key-secret -n example-ns -o yaml > secret_backup.yaml`

    Note: (*For DBaaS*) The `primary-credential-secret` and `kms-secret` key values will not be there in `cloudscale-values.yaml` file that have been backed up. Use the helm command to get values from the above step. By default helm would be using the values provided during the deployment.

    ---

12. Save the secrets named as Msdp credential and drInfoSecret during creation. As the operator would delete these secrets after using it.

    - MSDP credential: Step 2 in the following section:
      See “Configuring MSDP Scaleout” on page 419.
    - drInfoSecret: Step 2 in the following section:
      See the section called “Manual creation of catalog backup policy” on page 234.

13. (*For DBaaS*) Note the password changed during DBaaS cluster deployment:

    - (*For Azure*) To check the OLD_ DBADMINPASSWORD execute the following command after executing into the primary pod:

```
[root@user-primary-0 mnt]# ls -a
.  ..  atdata  .db-cert  nbdata  nbdb  .nb-kmsdb  nblogs
.nb-pgdb  nbu-primary-env  .nb-user  .passphrase-secret
.password-secret  podemptydir  .token-secret
[root@user-primary-0 mnt]# cd .nb-pgdb/
[root@user-primary-0 .nb-pgdb]# ls
dbadminlogin  dbadminpassword  dbport  dbserver  pgbouncerport
[root@user-primary-0 .nb-pgdb]# cat dbadminlogin
dbadminlogin
[root@user-primary-0 .nb-pgdb]# cat dbadminpassword
5MrJrGaLSnDKxTJ0UiPvotqmbqSzqU5a
[root@user-primary-0 .nb-pgdb]# cat dbport
5432
[root@user-primary-0 .nb-pgdb]# cat dbserver
user-postgres.postgres.database.azure.com
[root@user-primary-0 .nb-pgdb]# cat pgbouncerport
6432
DBADMINPASSWORD
DBADMINPASSWORD DBADMINPASSWORD
```

- (*For EKS*) Login to AWS UI, navigate to Secrets Manager and find adminSecret. Naming convention for admin secrets are as follows:

  `admin-secret-<use cluster name remove prefix eks->`

14. Note the values (names) of the secretProviderClass.

    For example, `dbSecretProviderClass: db-secret-provider-class`

    Save the `secretProviderClass.yaml` file using the following command:

    ```
    kubectl get secretproviderclass <secretproviderclass-name> -n
    <sample-namespace> -o yaml> secretproviderclass_backup.yaml
    ```

    Note: The dbSecretProviderClass is an optional field. If it is not present in the `cloudscale-values.yaml` file, then skip this step.

15. Note the required values (names)and sSave internal configmap yaml using the following command:

    ```
    kubectl get configmap nbu-media-autoscaler-configmap flexsnap-conf
    nbuconf cs-config -n <sample-namespace> -o yaml >
    internalconfigmap_backup.yaml
    ```

Note: The `nbu-media-autoscaler-configmap` is an optional internal configmap. If it is not present in namespace, then remove `nbu-media-autoscaler-configmap` from the above command.

16. Take a note of cert-manager and trust-manager as the same versions of cert-manager and trust-manager should be used when deploying them during recovery.
17. (*If MSDP-C is configured*) Note the details of cloud STU used for MSDP storage, such as name of bucket, volume, credential and the respective details added through Credential management in UI.
18. (*Applicable only for DBaaS based deployment environment*) Snapshot Manager backup steps:

    For AKS

    - Search the disk (PV) to which psql pvc is attached in Azure cloud portal and click on **Create snapshot** in the different resource group other than the cluster infra resource group and note down this resource group. Wait for the resource to be available.

      Note: Snapshot must be created in resource group in different availability zone to take care of the recovery in case of zone failures/corrupted.

      Save the `pgsql-pv.yaml` file:

      ```
      kubectl get pv | grep psql-pvc
      pvc-079b631e-a905-4586-80b5-46acc7011669 30Gi RWO Retain Bound
      nbu/psql-pvc managed-csi-hdd 3h10m
      kubectl describe pv <PV which is bound to psql-pvc> >
      pgsql-pv.yaml
      ```

      For example, `kubectl describe pv pvc-079b631e-a905-4586-80b5-46acc7011669 > pgsql-pv.yaml`

    - Note down the snapshot id, which would be used to create a disk from snapshot during recovery.

      Note: Disk Snapshot must be taken after every plugin addition as the latest database is required to recover all the plugins during Database recovery.

    For EKS

- Describe the PV attached to psql-pvc and save the VolumeID (for example, `vol-xxxxxxxxxxxxxxx`), storage class name and availability zone (AZ) from the output of following command:

  ```
  kubectl get pv | grep psql-pvc
  pvc-079b631e-a905-4586-80b5-46acc7011669 30Gi RWO Retain Bound
  nbu/psql-pvc managed-csi-hdd 3h10m
  kubectl describe pv <PV which is bound to psql-pvc> >
  pgsql-pv.yaml
  ```

  For example, `kubectl describe pv pvc-079b631e-a905-4586-80b5-46acc7011669 > pgsql-pv.yaml`
- Search above VolumeID in the **EC2 management console > Elastic Block Store > Volumes** in AWS cloud portal.
- Create snapshot (expand the **Actions** drop down) from the volume and wait for the completion. Note down the snapshot id (for example, `snap-xxxxxxxxxxxx`)

  Note: Disk Snapshot must be taken after every plugin addition as the latest database is required to recover all the plugins during Database recovery.

19. Take the backup of catalog policy at `/mnt/nbdata/DrPackages` on the master server.

Note: For manual deployment using Helm charts, ensure that you save the `operators-values.yaml` and `cloudscale-values.yaml` files. These files are used at the time of recovery.

Cluster recovery

This section describes the procedure for manual recovery of AKS and EKS clusters.

If initial deployment was done using Terraform, then cluster can be deployed using Terraform again.

Manual recovery of AKS cluster

Before performing the recovery of AKS cluster, template must be created.

Template creation

1 Modify the `template.json` file saved earlier by removing the following sections:

Under resources section for cluster > properties:

- ```
  windowsProfile": {
                      "adminUsername": "<Default username>",
                      "enableCSIProxy": true
                  },
  ```

- ```
  identityProfile": {
                      "kubeletidentity": {
                          "resourceId":
  "[parameters('userAssignedIdentities_<clustername>_<master
  node pool name>_externalid')]",
                          "clientId": "<CLIENT ID>",
                          "objectId": "<OBJECT ID>"
                      }
                  },
  ```

- Delete the resource of type `Microsoft.ContainerService/managedClusters/privateEndpointConnections:`

  ```
  {
  "type":
  "Microsoft.ContainerService/managedClusters/privateEndpointConnections",


       "apiVersion": "2023-08-02-preview",
               .


     .
     .
  }
  ```

- Delete the resource of type `Microsoft.ContainerService/managedClusters/agentPools/machines:`

  ```
  {
        "apiVersion": "2025-10-02-preview",
        "dependsOn": [
          ------
          ------
          --------
        "properties": {},
        "type":
  ```

```
"Microsoft.ContainerService/managedClusters/agentPools/machines"
}
```

2 If cluster recovering is in different availability zone, then set appropriate zone for the cluster and nodepools in `template.json` file.

```
"availabilityZones": [
"<zone number>"

],
```

3 Save the `template.json` file to be used later for restore.

Note: As and when Microsoft Azure updates, there may be more changes required in the template. For more information, refer to Use Azure portal to export a template

---

AKS cluster recovery

1 Delete the cluster that needs to be recovered (so that same name, IPs, and so on can be used).

2 Re-create cluster using templates saved earlier as follows:

- Ensure that you are logged in through Azure CLI with subscription set to where the cluster is present:

  ```
  az deployment group create --resource-group <resource_group>
  --name <deployment_name> --template-file template.json
  ```

  Here,

  resource_group: Resource group where cluster is present.

  deployment_name: Deployment name is the name with which you may look for the deployment under the deployments section of the resource group in portal.

- Once deployment is complete, a successful completion without any errors, with a deployment summary JSON.

- Connect to cluster and perform the remaining recovery options through CLI:

  - Connect to cluster:

    ```
    az login
    az account set --subscription <subscriptionID>
    az aks get-credentials --resource-group <resource_group>
    --name <cluster-name>
    kubelogin convert-kubeconfig -l azurecli
    ```

Here,
subscriptionID: Subscription ID where cluster is present.
resource_group: Resource group where cluster is present.
cluster-name: Name of the cluster.

- Attach ACR:

  ```
  az aks update -n <cluster-name> -g <resource_group>
  --attach-acr <ContainerRegistry>
  ```

  Here,
  ContainerRegistry: Name of Azure container registry where images are pushed.
  resource_group: Resource group where cluster is present.
  cluster-name: Name of the cluster.

- Authorize cluster to access Virtual Networks:
  If authorization is done through cluster service principal, then perform the following steps:

  - Get Service Principal ID:

    ```
    az resource list -n <cluster_name> --query
    [*].identity.principalId --out tsv
    ```

  - Role assignment:

    ```
    az role assignment create --assignee
    <clusterServicePrincipal> --role '<nbux-deployment-role>'
     --scope
    /subscriptions/<subscriptionID>/resourceGroups/<resource_group>/providers/Microsoft.Network/virtualNetworks/<vnet_pods>/subnets/<subnet_pods>
    ```

    ```
    az role assignment create --assignee
    <clusterServicePrincipal> --role '<nbux-deployment-role>'
     --scope
    /subscriptions/<subscriptionID>/resourceGroups/<resource_group>/providers/Microsoft.Network/virtualNetworks/<vnet_loadBalancer>/subnets/<subnet_loadBalancer_ips>
    ```

    Here,
    clusterServicePrincipal: Service Principal ID of cluster.
    nbux-deployment-role: Custom Role that has necessary permissions for NetBackup deployment.
    subscriptionID: Subscription ID where virtual network is present.
    resource_group: Resource group where virtual network is present.

  - If User Managed Identity or System Managed Identity is attached to scale sets then re-attach the same identity to scale sets.

  - Refresh the credentials of cluster as follows:

```
az aks get-credentials --resource-group <resource_group>
--name <cluster_name>
```

Here,

resource_group: Resource group where cluster is present.

cluster-name: Name of the cluster.

Manual recovery of EKS cluster

Note: In recovery steps keep all commands output files handy. Refer appropriate files saved during backup steps for respective command.

For IAM role and security group, user can refer to files which were created during backup steps of IAM roles and security group.

EKS cluster recovery

1. Cluster recovery:

   To create a new cluster, user can refer to the following fields from output of the `aws eks describe-cluster --name <Cluster name>` command:

   Name, **Kubernetes version 3**, **Cluster service role**, **tags**, **VPC**, **Subnet, Security Group** and **Cluster endpoint access**

2. Nodegroup recovery:

   To create a node group for a new cluster, user can refer to the following fields from output of the `aws eks describe-nodegroup --nodegroup-name "<nodegroup-name>" --cluster-name <cluster-name>` command:

   Nodegroup name, **Cluster name**, **Scaling config**, **Instance type**, **Node role**, **Disk size**, **Labels**, **Taints** , **Tags** and **Subnet**

3 File system:

To create new file system storage, user can refer to the following fields from the output of the `aws efs describe-file-systems --file-system-id <EFS ID>` and `aws efs describe-mount-targets --file-system-id <EFS ID>` commands:

Name, **Virtual Private Cloud (VPC)**, **Performance mode**, **Throughput mode** , **Provisioned Throughput (applicable only if Throughput mode is provisioned)**, **Network access**, **Virtual Private Cloud (VPC)** (**Mount targets** and **AZ, SubnetID, IPAddr, SecurityGroup**)

4 Add-ons

Once cluster is up and running, user must install add-ons listed by using the following command:

```
aws eks list-addons --cluster-name <cluster-name>
```

In addition to listed add-on, user must install **AWS load balancer controller add-on** and **Amazon EFS CSI driver**.

Note: If deploying in different availability zone, then choose subnet from that availability zone.

Note: If there is an availability zone failure or corruption in the AKS/EKS cluster and not able to recover the AKS/EKS cluster, then perform the procedure in the following section to recover Cloud Scale:

See "Cloud Scale recovery" on page 297.

Cloud Scale recovery

Perform the following procedure to recover Cloud Scale with the preserved configuration when there is an availability zone failure or cluster corruption.

There is a difference in term of subnet span for Azure and Amazon as follows:

- Azure: The subnet spans all availability zones in a region. So, for deploying in different availability zone same subnets can be used.
- Amazon: A subnet must live within a single availability zone. So, different subnet must be used corresponding to the availability zone being recovered.

Note: As part of recovery, volumes would be deleted and hence NetBackup logs that existed before disaster recovery would not be available in the fluent bit collector pod.

To recover Cloud Scale

1 Disaster recovery of Infra:

Perform the following steps for disaster recovery of Infra:

- **For deployment done using Terraform script**: Run the Terraform script for **base** and **addon** again with same inputs which was saved during successful deployment. For post deployment modifications done, update the additional infra details manually.

  Note: If deploying in EKS and in different availability zone, then use subnet corresponding to the new availability zone but ensure that FQDN remains same even though IP may be different.

- **For manual deployments or upgrade from previous deployments**: Perform the following steps:
  - Recover from the saved template or information:
    For AKS: Modify the template and then recover AKS cluster from the template.
    For EKS: Perform the EKS cluster recovery steps.
    See "Cluster recovery" on page 292.
  - Install cert-manager (refer to the following section) and ensure that the prerequisites listed in the following section are met:
    See "Preparing the environment for NetBackup installation on Kubernetes cluster" on page 31.

Note: If user needs to destroy the setup created, user must verify if the required permissions are provided to the key vault created during cluster creation phase. If the required permissions are not provided then the destroy command will display the following error:

```
does not have secrets get permission on key vault
```

---

2 Recovery of Cloud Scale:

Start deployment and recovery using the steps from the following section:

See " Environment Disaster Recovery" on page 299.

Environment Disaster Recovery

1. Ensure that the Cloud Scale deployment has been cleaned up in the cluster.

   Perform the following to verify the cleanup process:

   - Ensure that the namespace associated with Cloud Scale deployment are deleted by using the following command:

     `kubectl get ns`

   - Confirm that storageclass, pv, clusterroles, clusterrolebindings, crd's associated with Cloud Scale deployment are deleted by using the following command:

     `kubectl get sc,pv,pvc`

2. (*For EKS*) If deployment is in different AZ, update the subnet name in `cloudscale-values.yaml` file.

   For example, if earlier subnet name was **subnet-az1** and new subnet is **subnet-az2**, then in `cloudscale-values.yaml` file, there would be a section for `loadBalancerAnnotations` as follows:

   ```
   loadBalancerAnnotations:
             service.beta.kubernetes.io/aws-load-balancer-subnets:
    subnet-az1
   ```

   Update the name to new subnet name as follows:

   ```
   loadBalancerAnnotations:
             service.beta.kubernetes.io/aws-load-balancer-subnets:
    subnet-az2
   ```

   Update all IPs used for Primary, MSDP, Media and Snapshot Manager server in respective section.

   Note: Change of FQDN is not supported.

   The following example shows how to change the IP for Primary server:

   Old entry in `cloudscale-values.yaml` file:

   ```
   ipList:
               - ipAddr: 12.123.12.123
                 fqdn: primary.netbackup.com
   ```

   Update the above old entry as follows:

```
ipList:
            - ipAddr: 34.245.34.234
              fqdn: primary.netbackup.com
```

Similarly perform the above given procedure in the example (Primary server) for MSDP, Media and Snapshot Manager server.

3. Ensure that the iplist listed in Primary, Media, MSDP and Snapshot Manager server sections of `cloudscale-values.yaml` file that was saved during backup must be free and resolvable. If deployment is in different AZ, then FQDN must be same, but IP can be changed, hence ensure that same FQDN's can map to different IP.
4. (*For EKS*) Update spec > priamryServer > storage > catalog > storageClassName with new EFS ID which is created for primary in `cloudscale-values.yaml` file.
5. Ensure that nodeSelector is present in the `cloudscale-values.yaml` and `operators-values.yaml` files that were noted down during backup must be present in the cluster with required configurations.
6. Perform the steps in the following section for deploying DBaaS:

   See "DBaaS Disaster Recovery" on page 310.
7. Verify that the values in `secret_backup.yaml`, `storageclass_backup.yaml`, `CPServerLog_storageclass_backup.yaml`, and `msdpopstorageclass_backup.yaml` are correct. If a non-default StorageClass or updated passwords were used during deployment, ensure that these updated values are also included in `cloudscale-values.yaml` file.

   (*For DBaaS*) Confirm that the values in `secretproviderclass_backup.yaml` file are correctly reflected under global → dbsecretProviderClass in the backup `cloudscale-values.yaml` file, and update the admin secret ARN in `cloudscale-values.yaml` file with the new ARN provided in the AWS console.

   For AKS: Update the following values in `cloudscale-values.yaml` using the keyvault created in the earlier steps:

```
#      username: XXXXX
  #     client_id: XXXXX
  #     keyvault_name: XXXXX
  #     cloud_name: XXXXX
  #     tenant_id: XXXXX
  #     login_secret: XXXXX
  #     pass_secret: XXXXX
  #     host_secret: XXXXX
```

```
#     pgbouncer_port_secret: XXXXX
#     port_secret: XXXXX
```

Value mapping and commands:

- client_id: Retrieve the client ID using the following command:
  `az aks show -g "${GROUP_NAME}" -n "${AKS_NAME}" --query addonProfiles.azureKeyvaultSecretsProvider.identity.clientId -o tsv 2>/dev/null`
- tenant_id: Retrieve the tenant ID: `az account show --query 'tenantId' -o tsv`
- username: Set this value to the client_id.
- kevault: To list the available secrets in the Key Vault: `az keyvault secret list --vault-name <KV_NAME> --query "[].name" -o tsv))`

8. Install cert-manager (while restore cert-manager version should be same as given during backup):

   ```
   helm repo add jetstack https://charts.jetstack.io --force-update
   ```

   ```
   helm upgrade -i -n cert-manager cert-manager jetstack/cert-manager
   --version 1.18.2 --set webhook.timeoutSeconds=30 --set
   installCRDs=true --wait --create-namespace
   ```

9. Create namespace that is present in `cloudscale-values.yaml` file:

   ```
   kubectl create ns netbackup
   ```

10. Install trust-manager (while restore trust-manager version should be same as given while backup setup has been deployed):

    ```
    kubectl create namespace trust-manager
    ```

    ```
    helm upgrade -i -n trust-manager trust-manager
    jetstack/trust-manager --set app.trust.namespace=netbackup
    --version v0.19.0 --wait
    ```

11. (*For EKS*) Update the EFS ID in the backed-up `nb-file-premium` StorageClass to the new EFS ID. Then, install the operator using the `operator-values.yaml` file that was backed up during the disaster recovery backup procedure:

    ```
    helm upgrade --install operators operators-<version>.tgz -f
    operator-values.yaml --create-namespace --namespace
    netbackup-operator-system
    ```

12. (*Required only for DBaaS deployment*) Snapshot Manager restore steps:

    For AKS

- Navigate to the snapshot resource created during backup and **Create a disk** under the recovered cluster infra resource group (for example, `MC_<clusterRG>_<cluster name>_<cluster_region>`).
- Note down the resource ID of this disk (navigate to the **Properties** of the disk). It can be obtained from portal/az cli.
  Format of resource ID:`/subscriptions/<subscription id>/resourceGroups/<MC_<clusterRG>_<cluster name>_<cluster_region>/providers/Microsoft.Compute/disks>/<disk name>`
- Create static PV using the resource ID of backed up disk. Copy the below yaml and update the pv name, size of the disk, namespace and storage class name in `pgsql-pv.yaml` file and apply the yaml:

  `pgsql-pv.yaml`

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: <pv name>
spec:
  capacity:
    storage: <size of the disk>
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: <storage class name>
  claimRef:
    name: psql-pvc
    namespace: <environment namespace>
  csi:
    driver: disk.csi.azure.com
    readOnly: false
    volumeHandle: <Resorce ID of the Disk>
```

  Example of `pgsql-pv.yaml` file:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: psql-pv
spec:
  capacity:
    storage: 30Gi
  accessModes:
```

```
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: gp2-immediate
  claimRef:
    name: psql-pvc
    namespace: nbux
  csi:
    driver: disk.csi.azure.com
    readOnly: false
    volumeHandle:
[illegible]
```

Create psql-pv using the following command:

```
kubectl apply -f <path_to_psql_pv.yaml> -n
<environment-namespace>
```

- Ensure that the newly created PV is in *Available* state before restoring the Snapshot Manager server as follows:

  ```
  kubectl get pv | grep psql-pvc
  >> psql-pv 30Gi RWO managed-premium-disk Available nbu/psql-pvc
  50s
  ```

For EKS

- Navigate to the **EC2 > Snapshots** in AWS Console and click on the **Create volume from the snapshot** (expand the **Actions** drop down) which is taken in backup step 2 in same availability zone (AZ) of volume attached to psql-pvc (mentioned in step 1 of backup steps).
  Note down the volumeID (for example, **vol-xxxxxxxxxxxxxxx**).
- In case deployment is in different availability zone (AZ), user must change the availability zone (AZ) for volume and update the volumeID accordingly.
- Create static PV using the backed up volumeID. Copy the below yaml and update the pv name, size of the disk, namespace and storage class name in `pgsql-pv.yaml` file and apply the yaml:

  pgsql-pv.yaml

  ```
  apiVersion: v1
  kind: PersistentVolume
  metadata:
    name: <pv name>
  spec:
    accessModes:
    - ReadWriteOnce
    awsElasticBlockStore:
  ```

```
    fsType: <fs type>
    volumeID: <backed up volumeID>    # append this
aws://az-code/ , for e.g. aws://us-east-2b/ at the starting
  capacity:
    storage: 30Gi
  claimRef:
    apiVersion: v1
    kind: PersistentVolumeClaim
    name: psql-pvc
    namespace: <netbackup namespace>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: <storage class name>
  volumeMode: Filesystem
```

Sample yaml for `pgsql-pv.yaml` file:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: psql-pv
spec:
  accessModes:
  - ReadWriteOnce
  awsElasticBlockStore:
    fsType: ext4
    volumeID: aws://us-east-2b/vol-0d86d2ca38f231ede
  capacity:
    storage: 30Gi
  claimRef:
    apiVersion: v1
    kind: PersistentVolumeClaim
    name: psql-pvc
    namespace: nbu
  persistentVolumeReclaimPolicy: Retain
  storageClassName: gp2-immediate
  volumeMode: Filesystem
```

Create psql-pv using the following command:

```
kubectl apply -f <path_to_psql_pv.yaml> -n
<netbackup-namespace>
```

```
kubectl get pv | grep psql-pvc
```

- Ensure that the newly created PV is in *Available* state before restoring the Snapshot Manager server as follows:

```
kubectl get pv | grep psql-pvc
>>> psql-pv 30Gi RWO gp2-immediate Available nbu/psql-pvc 50s
```

13. Update the `cloudscale-values.yaml` file as follows:
    - Add *paused:true* for MSDP, Media Server in `cloudscale-vales.yaml` file and modify the file as `environment→ mediaServers→ paused: true; environment→ msdpScaleouts→ paused: true`

      Note: (*For DBaaS*) Ensure that `createServiceAccount` is set to *false* in `cloudscale-values.yaml` file.

    - Do not install cpServer.
      Create a copy of the `cloudscale-values.yaml` file and name it `cloudscale-values-copy.yaml`. Store this copy, as it would be required during the Snapshot Manager installation.
      Remove the entire cpServer section from the `cloudscale-values.yaml` file, and add *disabled: true* under the `cpServer` section.
14. Install Cloud Scale using the updated `cloudscale-values.yaml` file that is created in the above step.

    ```
    helm upgrade --install cloudscale cloudscale-<version>.tgz -f
    cloudscale-values.yaml --create-namespace --namespace netbackup
    ```
15. Once primary erver is up and running, perform the following:
    - Execute `kubectl exec -it -n <namespace> <primary-pod-name> -- /bin/bash` command to exec into the primary pod.
      Increase the debug logs level on primary server (set VERBOSE = 6 in `bp.conf` file).
    - Create a directory DRPackages at persisted location using `mkdir /mnt/nbdata/usr/openv/drpackage` command.
    - Deactivate NetBackup health probes using the `/opt/veritas/vxapp-manage/nb-health deactivate` command.
16. (*For containerized Postgres*) Execute the following command in the NetBackup namespace to scale down PostgreSql sts Replicas to 0:

    ```
    kubectl scale sts nb-postgresql -n netbackup --replicas=0
    ```

    (*For DBaaS*) Scale down or power off the DBaaS Server/Instance.

17. Exec into primary pod using `kubectl exec -it -n <namespace> <primary-pod-name> -- /bin/bash` command and stop the NetBackup services using the following command:

    `/usr/openv/netbackup/bin/bp.kill_all`

    Check if all processes are terminated correctly using `/usr/openv/netbackup/bin/bpps` command.

    If some processes remain, use the following command to forcefully terminate them:

    `kill -9 <process id>`

18. Perform the following steps for NBATD pod recovery:

    - Create the `DRPackages` directory on persisted location `/mnt/nblogs/` in nbatd pod by executing the following command:

      `kubectl exec -it -n <namespace> <nbatd-pod-name> --/bin/bash`

      `mkdir /mnt/nblogs/DRPackages`

    - Copy DR files which were saved when performing DR backup to nbatd pod at `/mnt/nblogs/DRPackages` using the following command:

      `kubectl cp <Path_of_DRPackages_on_host_machine> <nbatd-pod-namespace>/<nbatd-pod-name>:/mnt/nblogs/DRPackages`

    - Execute the following steps in the nbatd pod:

      - Execute the `kubectl exec -it -n <namespace> <nbatd-pod-name> --/bin/bash` command.
      - Deactivate nbatd health probes using the `/opt/veritas/vxapp-manage/nbatd_health.sh disable` command.
      - Stop the nbatd service using `/opt/veritas/vxapp-manage/nbatd_stop.sh 0` command.
      - Execute the `/opt/veritas/vxapp-manage/nbatd_identity_restore.sh -infile /mnt/nblogs/DRPackages/ (DR package name)` command.

19. Copy back the earlier copied disaster recovery files to primary pod at `/mnt/nbdata/usr/openv/drpackage` location using the following command:

    `kubectl cp <Path_of_DRPackages_on_host_machine> <primary-pod-namespace>/<primary-pod-name>:/mnt/nbdata/usr/openv/drpackage`

20. Execute the following steps after executing into the primary server pod:

    - Change the ownership of files in `/mnt/nbdb/usr/openv/drpackage` using the `chown nbsvcusr:nbsvcusr <file-name>` command.

- Execute the `/usr/openv/netbackup/bin/admincmd/nbhostidentity -import -infile /mnt/nbdb/usr/openv/drpackage/.drpkg` command.
- Clear NetBackup host cache, run the `bpclntcmd -clear_host_cache` command.
- Restart the pods as follows:
  - Navigate to the `VRTSk8s-netbackup-<version>/scripts` folder.
  - Run the `cloudscale_restart.sh` script with Restart option as follows:
    `./cloudscale_restart.sh` *`<action>`* *`<namespace>`*
    Provide the namespace and the required action:
    stop: Stops all the services under primary server (waits until all the services are stopped).
    start: Starts all the services and waits until the services are up and running under primary server.
    restart: Stops the services and waits until all the services are down. Then starts all the services and waits until the services are up and running.

  Note: Ignore if policy job pod does not come up in running state. Policy job pod would start once primary services start.

- Refresh the certificate revocation list using the `/usr/openv/netbackup/bin/nbcertcmd -getcrl` command.

Note: The primaryserver CR is automatically unpaused by the `cloudscale_restart.sh` script. However, in the Cloud Scale helm chart, the primary component remains paused (as configured in the above step). To unpause it in the helm chart, execute the following command:

```
helm upgrade cloudscale <path to cloudscale.tar.gz> --reuse-values
\ --set environment.primary.paused=false -n netbackup
```

---

21. Run the primary server reconciler.

    This can be done by changing primary spec's **paused** field to **true** using the following command and save it:

    ```
    helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup
    --reuse-values --set environment.primary.paused=true
    ```

    Then, to enable the reconciler to run, the primary's paused field in spec should again be set to false.

The SHA fingerprint will get updated in the primary CR's status.

22. Allow Auto reissue certificate from primary for MSDP, Media and Snapshot Manager server from Web UI.

    In Web UI, navigate to Security > Host Mappings > for the MSDP Storage Server, click on the 3 dots on the right > check **Allow Auto reissue Certificate**. Repeat this for media servers and Snapshot Manager server entries also.

23. Apply the `msdp-cred-secret` backed up during the disaster recovery backup.

    Change **paused** field to *false* for MSDP using the following command:

    ```
    helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup
    --reuse-values --set environment.msdpScaleouts.paused=false
    ```

24. Once MSDP is up and running, add the cloud provider credentials, from where the S3 bucket has been configured as described in "Add a credential in NetBackup' section of the *NetBackup™ Web UI Administrator's Guide*.

25. If the LSU cloud alias does not exist, you can use the following command to add it.

    ```
    /usr/openv/netbackup/bin/admincmd/csconfig cldinstance -as -in
    <instance-name> -sts <storage-server-name> -lsu_name <lsu-name>
    ```

    When MSDP Scaleout is up and running, re-use the cloud LSU on NetBackup primary server.

    ```
    /usr/openv/netbackup/bin/admincmd/nbdevconfig -setconfig
    -storage_server <STORAGESERVERNAME> -stype PureDisk -configlist
    <configuration file>
    ```

    Credentials, bucket name, and sub bucket name must be the same as the recovered Cloud LSU configuration in the previous MSDP Scaleout deployment.

    Configuration file template:

    ```
    V7.5 "operation" "reuse-lsu-cloud" string
    V7.5 "lsuName" "LSUNAME" string
    V7.5 "cmsCredName" "XXX" string
    V7.5 "lsuCloudAlias" "<STORAGESERVERNAME_LSUNAME>" string
    V7.5 "lsuCloudBucketName" "XXX" string
    V7.5 "lsuCloudBucketSubName" "XXX" string
    V7.5 "lsuKmsServerName" "XXX" string
    ```

Note: For Veritas Alta Recovery Vault Azure storage, the `cmsCredName` is a credential name and `cmsCredName` can be any string. Add recovery vault credential in the CMS using the NetBackup Web UI and provide the credential name for `cmsCredName`. For more information, see *About Veritas Alta Recovery Vault Azure* topic in *NetBackup Deduplication Guide*.

---

26. Change **paused = false** for media server using the following CLI and wait for media pods to come up and running:

```
helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup
--reuse-values \>                                 --set
environment.mediaServers[0].name=media1 --set
environment.mediaServers[0].replicas=1 \>
          --set
environment.mediaServers[0].nodeSelector.labelKey=agentpool \>
                              --set
environment.mediaServers[0].nodeSelector.labelValue=mediapool  \>
                                --set
environment.mediaServers[0].storage.data.capacity=50Gi \>
                          --set
environment.mediaServers[0].storage.data.storageClassName=nb-disk-standardssd
\>                                   --set
environment.mediaServers[0].storage.log.capacity=5Gi \>
                        --set
environment.mediaServers[0].storage.log.storageClassName=nb-disk-standardssd
 \>                                  --set
environment.mediaServers[0].tag=11.2-0016-DR1 \>
                --set environment.mediaServers[0].paused=false
```

27. Perform full Catalog Recovery:

    Pause the environment reconciler using the `helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup --reuse-values --set environment.paused=true` command.

    Use one of the following option to perform catalog recovery:

    - Trigger a Catalog Recovery from the Web UI.
      Or
    - Exec into primary pod and run the `bprecover -wizard` command.

    If Multi Factor Authentication (MFA) was enabled, perform the following additional steps:

- Disable MFA: In Web UI, navigate to **Global Security Setting > Security Controls > Enforce multifactor authentication** and disable it.
- Exec into primary POD and execute the following command to reset MFA for user provided in primary secret (in this case it is nbuser):
  `nbseccmd -resetMFA -userinfo unixpwd:nbuxqa-summiteers-xx-xxx-xx-xxx.vxindia.cohesity.com:nbuser`
- Un pause the environment reconciler: `helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup --reuse-values --set environment.paused=false`
- Confirm **modified time** of access keys are modified to latest once:
  In Web UI, navigate to **Security > Access Key**, and ensure that the modified Access Keys are created by `nb-operator`.

28. Once recovery is completed, restart the NetBackup services by running the following script:

    `cloudscale_restart.sh`

29. Activate NetBackup health probes using the `/opt/veritas/vxapp-manage/nb-health activate` command.
30. Install Snapshot Manager server:
    - Add **cpServer** section in `cloudscale-values.yaml` file and ensure that the **disabled** field is set to *false* for the **cpSever** section.
    - Install **cpServer** in the environment using the following command:
      `helm upgrade --install cloudscale cloudscale-<version>.tgz -f cloudscale-values.yaml --namespace netbackup`
    - Wait for the Snapshot Manager pods to come up and in running state.
31. Validate if the environment is up and running.

DBaaS Disaster Recovery

For Azure

1. Run following commands after providing the required values:

```
export SERVER_NAME=<Postgress Server Name can be found from azure
 UI> #Change IT
export NAMESPACE=netbackup #Change IT
export AKS_SUBNET_NAME=<vnet_name from applied TF-Var file>
#Change IT
```

```
export KV_NAME=<Key Vault Name can be found from azure UI> #Change
 IT
export AKS_NAME=<aks_name from applied TF-Var file> #Change IT
export GROUP_NAME=<new_rg_name from applied TF-Var file>
export PG_SUBNET_NAME=<db_subnet_name from applied TF-Var file>
export LOCATION="<location can be found from azure UI>"
export VNET_RESOURCE_GROUP=<vnet_rg_name from applied TF-Var file>
export VNET_NAME=<vnet_name from applied TF-Var file>
export TAGS=""
export PSQL_DNS_ZONE_NAME=<can be found from azure UI go to
postgres server then networking and use the name of private DNS
Zone being used>
export PRIVATE_DNS_LINK_NAME=<dns_to_vnet_link_name from applied
 TF-Var file>
export DB_LOGIN_NAME="dbadminlogin"
export DB_SECRET_NAME="dbadminpassword"
export DB_SERVER_NAME="dbserver"
export SECRET_PROVIDER_CLASS_NAME="dbsecret-spc"
export DB_PG_BOUNCER_PORT_NAME="pgbouncerport"
export DB_PORT_NAME="dbport"
export DB_CERT_NAME="dbcertpem"
export CLIENT_ID=$(az aks show -g "${GROUP_NAME}" -n "${AKS_NAME}"
 --query
addonProfiles.azureKeyvaultSecretsProvider.identity.clientId -o
tsv 2>/dev/null)
export TENANT_ID=$(az account show --query 'tenantId' -o tsv)
export
DB_CERT_URL="https://cacerts.digicert.com/DigiCertGlobalRootCA.crt.pem"
export TLS_FILE_NAME='/tmp/tls.crt'
```

2. Run the following command:

```
export KEYVAULT_ID=$(az keyvault show --name "${KV_NAME}"
--resource-group "${GROUP_NAME}" --query id --output tsv)
```

```
az postgres flexible-server parameter set --resource-group
"${GROUP_NAME}" --server "${SERVER_NAME}" --name
require_secure_transport --value off
```

3. Run the following commands:

```
TLS_FILE_NAME='/tmp/tls.crt'
PROXY_FILE_NAME='/tmp/proxy.pem'

rm -f ${TLS_FILE_NAME} ${PROXY_FILE_NAME}
```

```
DB_CERT_URL="https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem"
DB_PROXY_CERT_URL="https://www.amazontrust.com/repository/AmazonRootCA1.pem"


curl ${DB_CERT_URL} --output ${TLS_FILE_NAME}
curl ${DB_PROXY_CERT_URL} --output ${PROXY_FILE_NAME}

cat ${PROXY_FILE_NAME} >> ${TLS_FILE_NAME}

kubectl -n netbackup create secret generic postgresql-netbackup-ca
 --from-file ${TLS_FILE_NAME}
```

4. Reset the password and use the same one used at the time of backup.

   For more information on resetting the password refer to the *Azure-specific* procedure in the following section:

   See "Changing database server password in DBaaS" on page 219.

For AWS

1. Create Service Account for service access:

```
# create secret access policy
cat <<EOF > /tmp/db-secret-access-policy.json
{
  "Version": "2012-10-17",
  "Statement": [ {
     "Effect": "Allow",
     "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
     ],
     "Resource": [
        <admin-secret-arn>,
        <cert-secret-arn>
     ]
    } ]
 }
EOF


$ aws iam create-policy \
  --policy-name db-secret-access-policy \
```

```
  --policy-document file:///tmp/db-secret-access-policy.json

# create SA and link it with IAM Policy
eksctl create iamserviceaccount \
  --override-existing-serviceaccounts \
  --approve \
  --config-file - <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: $EKS_CLUSTER_NAME
  region: $REGION
  tags:
    OWNER: $OWNER
iam:
  withOIDC: true
  serviceAccounts:
    - metadata:
        name: db-access
        namespace: netbackup
      attachPolicyARNs:
        - $SECRET_ACCESS_POLICY_ARN
      permissionsBoundary: $PERMISSIONS_BOUNDARY
EOF
```

2. Run the following command:

```
TLS_FILE_NAME='/tmp/tls.crt'
PROXY_FILE_NAME='/tmp/proxy.pem'

rm -f ${TLS_FILE_NAME} ${PROXY_FILE_NAME}

DB_CERT_URL="https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem"
DB_PROXY_CERT_URL="https://www.amazontrust.com/repository/AmazonRootCA1.pem"


curl ${DB_CERT_URL} --output ${TLS_FILE_NAME}
curl ${DB_PROXY_CERT_URL} --output ${PROXY_FILE_NAME}

cat ${PROXY_FILE_NAME} >> ${TLS_FILE_NAME}

kubectl -n netbackup create secret generic postgresql-netbackup-ca
 --from-file ${TLS_FILE_NAME}
```

3. Perform the steps listed in the *AWS-specific* procedure in the following section to change password and replace with password saved during backup phase:

   See "Changing database server password in DBaaS" on page 219.

Uninstalling

This chapter includes the following topics:

- Uninstalling Cloud Scale Technology

Uninstalling Cloud Scale Technology

Execute the binary at `bin/` with the uninstall option using the following command:

```
./kubectl-cloudscale uninstall
```

The output would be displayed as follows:

```
Are you sure you want to uninstall NetBackup CloudScale? This action
 cannot be undone. (y/n): y

Fetching namespaces from installed Helm charts...
Both Cloud Scale and Operator charts found. Proceeding with
uninstallation of all associated components.

Uninstalling Cloud Scale from namespace: netbackup
Uninstalled cloudscale helm chart from the namespace: 'netbackup'

Waiting for all pods to terminate in namespace: netbackup

All pods in netbackup namespace are terminated.


Uninstalling Operator from namespace: netbackup-operator-system
Uninstalled operators helm chart from the namespace:
'netbackup-operator-system'

Waiting for all pods to terminate in namespace:
```

```
netbackup-operator-system

All pods in netbackup-operator-system namespace are terminated.

Deleting PVCs in namespace: netbackup

Deleting PVCs in namespace: netbackup-operator-system

Waiting for PVCs to be deleted in namespace: netbackup

Waiting for PVCs to be deleted in namespace: netbackup-operator-system

Deleting namespaces and waiting for deletion...
Deleting namespace: netbackup

Waiting for netback namespace to terminate...

Deleting namespace: netbackup-operator-system

Waiting for netbackup-operator-system namespace to terminate...

Full uninstallation completed successfully.
PVs are not deleted, please delete PVs manually.
```

To locate and delete any persistent volumes created by the deployment, run the following commands:

```
$ kubectl get pv

$ kubectl delete pv <pv-name> --grace-period=0 --force
```

Note: Some storage drivers may cause persistent volumes to remain stuck in the terminating state.

To resolve this issue, remove the finalizer, using the `$ kubectl patch pv <pv-name> -p '{"metadata":{"finalizers":null}}` command.

Chapter 22

Troubleshooting

This chapter includes the following topics:

- Troubleshooting AKS and EKS issues
- Troubleshooting AKS-specific issues
- Troubleshooting EKS-specific issues
- Troubleshooting issue for bootstrapper pod
- Troubleshooting issues for kubectl plugin

Troubleshooting AKS and EKS issues

This section lists the troubleshooting issues and their respective workaround that are common for the following cloud providers:

- Azure Kubernetes Services cluster
- Amazon Elastic Kubernetes cluster

View the list of operator resources

To view all the operator resources, execute the following command on Kubernetes cluster:

```
$ kubectl get all -n netbackup-operator-system
```

The output should be something like this:

```
   NAME                                                          READY
   STATUS    RESTARTS       AGE
pod/flexsnap-operator-7d45568767-n9g27                         1/1
  Running      0          18h
pod/msdp-operator-controller-manager-0                         2/2
```

```
  Running     0          43m
pod/msdp-operator-controller-manager-1                      2/2
  Running     0          44m
pod/netbackup-operator-controller-manager-6cbf85694f-p97sw   2/2
  Running     0          42m

NAME                                                      TYPE
CLUSTER-IP       EXTERNAL-IP    PORT(S)   AGE
service/msdp-operator-controller-manager-metrics-service
ClusterIP   10.96.144.99    <none>   8443/TCP   3h6m
service/msdp-operator-webhook-service
ClusterIP   10.96.74.75     <none>    443/TCP   3h6m

service/netbackup-operator-controller-manager-metrics-service
ClusterIP   10.96.104.94    <none>    8443/TCP  93m
service/netbackup-operator-webhook-service           ClusterIP
10.96.210.26    <none>    443/TCP   93m


NAME
               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/msdp-operator-controller-manager
                1/1     1            1          3h6m
deployment.apps/netbackup-operator-controller-manager-operator-controller-manager
  1/1     1            1          93m


NAME
DESIRED   CURRENT   READY   AGE
replicaset.apps/msdp-operator-controller-manager-65d8fd7c4d
 1         1         1     3h6m
replicaset.apps/netbackup-operator-controller-manager-55d6bf59c8
 1         1         1     93m
```

Verify that both pods display **Running** in the Status column and both deployments display **2/2** in the **Ready** column.

View the list of product resources

To view the list of product resources run the following command:

```
$ kubectl get --namespace <namespace>
all,environments,primaryservers,mediaservers,msdpscaleouts
```

The output should look like the following:

```
NAME                                          READY   STATUS
RESTARTS      AGE
pod/flexsnap-agent-5c87569449-rsg8r           1/1     Running   0
           19h
pod/flexsnap-api-gateway-5656578645-dcqkg     1/1     Running   0
           19h
pod/flexsnap-certauth-5f8d7f76bf-br9jd        1/1     Running   0
           19h
pod/flexsnap-coordinator-5df8df9f9b-8tmzm     1/1     Running   2
(66m ago)   19h
pod/flexsnap-listener-6f6b55447b-hnffv        1/1     Running   0
           19h
pod/flexsnap-nginx-86f8b9c8b4-szt8n           1/1     Running   3
(66m ago)   19h
pod/flexsnap-notification-7f6867467c-fd4wf    1/1     Running   2
(66m ago)   19h
pod/flexsnap-rabbitmq-0                       1/1     Running   0
           19h
pod/flexsnap-scheduler-85f56fd599-hn9fr       1/1     Running   2
(66m ago)   19h
pod/media1-media-0                            2/2     Running   0
           55m
pod/msdpx-uss-agent-859jc                     1/1     Running   0
           67m
pod/msdpx-uss-controller-0                    1/1     Running   0
           19h
pod/msdpx-uss-mds-1                           1/1     Running   0
           66m
pod/nb-fluentbit-collector-54c59d8c65-7ds58   2/2     Running   0
           19h
pod/nb-fluentbit-daemonset-j2smx              1/1     Running   0
           64m
pod/nb-fluentbit-daemonset-pvgd5              1/1     Running   0
           67m
pod/nb-fluentbit-daemonset-wmd56              1/1     Running   0
           67m
pod/nb-postgresql-0                           1/1     Running   0
           69m
pod/nbu-bpdbm-0                               5/5     Running   0
           69m
pod/nbu-log-viewer-0                          1/1     Running   0
```

```
           69m
pod/nbu-nbatd-0                              4/4     Running   0
           69m
pod/nbu-nbmqbroker-0                         2/2     Running   0
           69m
pod/nbu-nbwsapp-0                            4/4     Running   0
           69m
pod/nbu-policyjob-0                          5/5     Running   0
           69m
pod/nbu-policyjobmgr-0                       5/5     Running   0
           69m
pod/nbu-primary-0                            2/2     Running   0
           69m
pod/nbu-requestrouter-66f9cbbbd6-mk599       1/1     Running   0
           19h
pod/nbux-xx-xxx-xx-xxx.vxindia.cohesity.com    2/2     Running   0
             65m

NAME
  TYPE           CLUSTER-IP      EXTERNAL-IP    PORT(S)



                                                        AGE
service/flexsnap-api-gateway
  ClusterIP      10.10.76.161    <none>         8472/TCP



                                                        7d2h
service/flexsnap-certauth
  ClusterIP      10.10.92.5      <none>         9000/TCP



                                                        7d2h
service/flexsnap-nginx
  LoadBalancer   10.10.113.13    10.244.33.81
443:31203/TCP,5671:30421/TCP



          7d2h
service/flexsnap-rabbitmq
  ClusterIP      10.10.126.245   <none>         5671/TCP
```

```
                                                          7d2h
service/ip-10-244-33-80-host-nbux-10-244-33-80-vxindia-veritas-com
  LoadBalancer   10.10.251.63    10.244.33.80
[illegible]
   7d2h
service/log-viewer
  ClusterIP      10.10.43.73     <none>         8080/TCP



                                                          7d2h
service/media1-media-0
  ClusterIP      10.10.190.216   <none>         13724/TCP,1556/TCP



                                                          7d2h
service/msdpx-uss-controller
  ClusterIP      10.10.103.214   <none>         10100/TCP



                                                          7d2h
service/msdpx-uss-mds
  ClusterIP      None            <none>         2379/TCP,2380/TCP



                                                          7d2h
service/msdpx-uss-mds-client
  ClusterIP      10.10.20.57     <none>         2379/TCP



                                                          7d2h
service/nb-fluentbit-collector-svc
  ClusterIP      None            <none>
24224/TCP,24225/TCP,24226/TCP,24227/TCP,24228/TCP,24229/TCP



          7d2h
service/nb-postgresql
  ClusterIP      None            <none>         13787/TCP



                                                          7d2h
service/nbatd
  ClusterIP      10.10.194.78    <none>         1556/TCP
```

```
                                                          7d2h
service/nbmqbroker
  ClusterIP      10.10.18.54     <none>
13781/TCP,13777/TCP,13780/TCP


          7d2h
service/nbu-policyjob-0
  ClusterIP      10.10.109.211   <none>         13724/TCP,1556/TCP


                                                          7d2h
service/nbu-policyjobmgr-0
  ClusterIP      10.10.200.164   <none>         13724/TCP,1556/TCP


                                                          7d2h
service/nbu-primary
  LoadBalancer   10.10.2.26      10.244.33.79
13781:32084/TCP,13724:30101/TCP,1556:32450/TCP,443:30736/TCP


          7d2h
service/nbwsapp
  ClusterIP      10.10.38.84     <none>
443/TCP,1556/TCP,1558/TCP


          7d2h
service/bpdbm
  ClusterIP      10.10.100.211   <none>         13724/TCP,1556/TCP
7d2h
service/policyjobmgr
  ClusterIP      10.10.112.28    <none>         13724/TCP,1556/TCP


                                                          7d2h
service/primary
  ClusterIP      10.10.21.220    <none>
13781/TCP,13724/TCP,1556/TCP,443/TCP
```

```
          7d2h

NAME                                    DESIRED   CURRENT   READY
 UP-TO-DATE   AVAILABLE   NODE SELECTOR         AGE
daemonset.apps/msdpx-uss-agent          1         1         1
 1            1           agentpool=msdpxpool   7d2h
daemonset.apps/nb-fluentbit-daemonset   3         3         3
 3            3           <none>                7d2h

NAME                                     READY   UP-TO-DATE
AVAILABLE   AGE
deployment.apps/flexsnap-agent           1/1     1            1
       7d2h
deployment.apps/flexsnap-api-gateway     1/1     1            1
       7d2h
deployment.apps/flexsnap-certauth        1/1     1            1
       7d2h
deployment.apps/flexsnap-coordinator     1/1     1            1
       7d2h
deployment.apps/flexsnap-listener        1/1     1            1
       7d2h
deployment.apps/flexsnap-nginx           1/1     1            1
       7d2h
deployment.apps/flexsnap-notification    1/1     1            1
       7d2h
deployment.apps/flexsnap-scheduler       1/1     1            1
       7d2h
deployment.apps/nb-fluentbit-collector   1/1     1            1
       7d2h
deployment.apps/nbu-requestrouter        1/1     1            1
       7d2h

NAME                                               DESIRED   CURRENT
   READY   AGE
replicaset.apps/flexsnap-agent-5c87569449           1         1
     1       7d2h
replicaset.apps/flexsnap-api-gateway-5656578645     1         1
     1       7d2h
replicaset.apps/flexsnap-certauth-5f8d7f76bf        1         1
     1       7d2h
replicaset.apps/flexsnap-coordinator-5df8df9f9b     1         1
     1       7d2h
```

```
replicaset.apps/flexsnap-listener-6f6b55447b        1          1
     1       7d2h
replicaset.apps/flexsnap-nginx-86f8b9c8b4           1          1
     1       7d2h
replicaset.apps/flexsnap-notification-7f6867467c    1          1
     1       7d2h
replicaset.apps/flexsnap-scheduler-85f56fd599       1          1
     1       7d2h
replicaset.apps/nb-fluentbit-collector-54c59d8c65   1          1
     1       7d2h
replicaset.apps/nbu-requestrouter-66f9cbbbd6        1          1
     1       7d2h

NAME                                    READY   AGE
statefulset.apps/flexsnap-rabbitmq      1/1     7d2h
statefulset.apps/msdpx-uss-controller   1/1     7d2h
statefulset.apps/nb-postgresql          1/1     7d2h
statefulset.apps/nbu-log-viewer         1/1     7d2h
statefulset.apps/nbu-nbatd              1/1     7d2h
statefulset.apps/nbu-nbmqbroker         1/1     7d2h
statefulset.apps/nbu-bpdbm              1/1     7d2h
statefulset.apps/nbu-nbwsapp            1/1     7d2h
statefulset.apps/nbu-policyjob          1/1     7d2h
statefulset.apps/nbu-policyjobmgr       1/1     7d2h
statefulset.apps/nbu-primary            1/1     7d2h

NAME                                    READY   AGE    STATUS
environment.netbackup.veritas.com/nbu   4/4     7d2h   Success

NAME                                      TAG               AGE
STATUS
primaryserver.netbackup.veritas.com/nbu   11.0-0031-new31   7d2h
Success

NAME                                        TAG               AGE
 PRIMARY SERVER                           STATUS
mediaserver.netbackup.veritas.com/media1   11.0-0031-new31   7d2h
 nbux-xx-xxx-xx-xx.vxindia.cohesity.com   Success

NAME                                   AGE    TAG               SIZE
   READY
msdpscaleout.msdp.veritas.com/msdpx   7d2h   21.0-0024-new31   1
```

```
   1
$
```

An environment is deployed successfully if all pods and environment CR display status as "Success".

View operator logs

If environment deployment status is not successful, check operator logs for errors.

Command for MSDP Scaleout operator logs

```
$ kubectl logs pod/msdp-operator-controller-manager-65d8fd7c4d-whqpm
manager -n netbackup-operator-system -c manager
```

Command for NetBackup operator logs

```
$ kubectl logs
pod/netbackup-operator-controller-manager-55d6bf59c8-vltmp
netbackup-operator -n netbackup-operator-system
```

View primary logs

To view primary server logs execute the following command to get a shell to the running container.

```
$ kubectl exec --stdin --tty pod/<primary-server-pod-name> -n
<namespace> -- /bin/bash
```

Once in the primary server shell prompt, to see the list of logs, run:

```
ls /usr/openv/logs/
```

Socket connection failure

Socket connection failure can happen because of the following reasons:

- Long processing delays
- Azure/AWS connection reset (default 4 minutes)
- Load on CPU or Memory pressure
- IO saturation and throttling under load

If there are problems with the TCP stacks on the hosts, network between the hosts, or unusual long processing delays, then the connection may drop and the TCP stack on the host is unaware of the situation.

The following error is displayed in the web UI under job details:

```
db_FLISTsend failed: unexpected message received (43)
 *** - Error bptm (pid=14599) get_string() failed,
Connection reset by peer (104), network read error
*** - Info bptm (pid=14599) EXITING with status 42 <----------
*** - Info nbux-systest-media-1 (pid=14599)
StorageServer=PureDisk:nbux-systest-media-1;
Report=PDDO Stats for (nbux-systest-media-1):
scanned: 4195521 KB, CR sent: 171002 KB, CR sent over FC: 0 KB,
dedup: 95.9%, cache disabled, where dedup space saving:6.6%,
compression space saving:89.3%
*** - Info bpbkar (pid=19109) done. status: 42: network read failed
```

To resolve this issue, update the `sysctl.conf` values for NetBackup servers deployed on the Kubernetes cluster.

NetBackup image sets following values in `sysctl.conf` during Kubernetes deployment:

- net.ipv4.tcp_keepalive_time = 180
- net.ipv4.tcp_keepalive_intvl = 10
- net.ipv4.tcp_keepalive_probes = 20
- net.ipv4.ip_local_port_range = 14000 65535

These settings are persisted at the location `/mnt/nbdata/etc/sysctl.conf`.

Modify the values in `/mnt/nbdata/etc/sysctl.conf` and restart the pod. The new values are reflected after the pod restart.

If external media servers are used, perform the steps in the following order:

1. Add the following in `/usr/openv/netbackup/bp.conf`:

   HOST_HAS_NAT_ENDPOINTS = YES

2. Add the following sysctl configuration values in `etc/sysctl.conf` on external media servers to avoid any socket connection issues:

   - net.ipv4.tcp_keepalive_time = 180
   - net.ipv4.tcp_keepalive_intvl = 10
   - net.ipv4.tcp_keepalive_probes = 20
   - net.ipv4.ip_local_port_range = 14000 65535
   - net.core.somaxconn = 4096

3. Save the setting using the `sysctl -p` command.

Resolving an issue where external IP address is not assigned to a NetBackup server's load balancer services

The issue can be because of one of the following reasons:

- The **resourcePrefixName** mentioned in custom resource is not unique and valid.
- The static IP is provided in **networkLoadBalancer** section in CR but it is not created in Azure/AWS.
- The static IP is provided in **networkLoadBalancer** section in CR is already in use.
- The subnet or resource group is mentioned in annotations of **networkLoadBalancer** section in CR spec, the IP address is not available in given subnet or resource group.
- The RBAC permissions in your cluster for the given subnet or resource group are not assigned properly for allocating IP addresses.

To resolve an issue where external IP address is not assigned to a NetBackup server's load balancer services

1. Check the event logs of load balancer service using the `kubectl describe service <svc-name> -n <namespace>` command for detailed error information.
2. Run the `helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup --reuse-values --set` command.
3. Depending on the output of Step 1 and the reason for the issue, perform the required steps and update the environment CR using Step 2 to resolve the issue.

Resolving the issue where the NetBackup server pod is not scheduled for long time

The NetBackup server (primary server and media server) pods are stuck in Pending state. The issue can be because of one of the following reasons:

- Insufficient resource allocation.
- Persistent volume claims are not bound to persistent volume.

If nodes are not available, pod remains in pending state with event logs indicating nodes are scaling up, if auto scaling is configured in cluster.

To resolve the issue where the NetBackup server pod is not scheduled for long time

1. Check the pod event details for more information about the error using `kubectl describe <PrimaryServer/MediaServer_Pod_Name> -n <namespace>` command.
2. Depending on the output of the command and the reason for the issue, perform the required steps and update the environment CR to resolve the issue.

Resolving an issue where the Storage class does not exist

The Config-Checker checks if the storage class name given in primary server/media server CR is available in the cluster.

The following error is displayed:

```
Error: ERROR Storage class with the <storageClassName> name does not exist.
```

After fixing this error, primary server or media server CR does not require any changes. If you want to reflect the changes and invoke the NetBackup operator reconciler loop immediately, pause the reconciler of the custom resource by changing the *paused: false* value to *paused: true* in the primaryServer or mediaServer section by using the following command:

```
helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup
--reuse-values --set environment.primary.paused=true
```

Again change the value to *paused: false* (un pause) in the primaryServer or mediaServer section by using the following command:

```
helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup
--reuse-values --set environment.primary.paused=false
```

To resolve an issue where the Storage class does not exist

1. Create storage class with the same name given in primary server or media server CR with ReclaimPolicy as **Retain** in the cluster.

   To create storage class, refer to the following link:

   Azure Kubernetes Service storage classes

   Amazon Elastic Kubernetes Service storage classes

   In this scenario, no change in primary server or media server CR is required. As a result, reconciler loop is not invoked immediately.

2. To invoke the reconciler loop again, delete the respective CR by changing the *paused: false* value to *paused: true* in the primaryServer or mediaServer section in environment CR by using the following command:

   ```
   helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup
   --reuse-values --set environment.primary.paused=true
   ```

   Save the changes.

   Again change the value to *paused: false* in the primaryServer or mediaServer section by using the following command:

   ```
   helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup
   --reuse-values --set environment.primary.paused=false
   ```

   Save the changes.

Resolving an issue where the primary server or media server deployment does not proceed

The primary server or media server deployment does not proceed even if **configcheckmode = default** in primary server or media server CR spec and no other child resources are created. It is possible that the Config-Checker job has failed some of the configuration checks.

To resolve an issue where the primary server or media server deployment does not proceed

1. Check the status of Config-Checker **Configcheckerstatus** mentioned in primary server or media server CR status using the `kubectl describe <PrimaryServer/MediaServer> <CR name> -n <namespace>` command.

   If the state is **failed**, check the Config-Checker pod logs.

2. Retrieve the Config-Checker pod logs using the `kubectl logs <config-checker-pod-name> -n <operator-namespace>` command.

   Config-Checker pod name can be in the following format:

   `<serverType>-configchecker-<configcheckermode>-randomID`, for example if its Config-Checker for primary server with **configcheckermode = default**, pod name is `primary-configcehcker-default-dhg34`.

3. Depending on the error in the pod logs, perform the required steps and edit the environment CR to resolve the issue.

4. Data migration jobs create the pods that run before deployment of primary server. Data migration pod exist after migration for one hour only if data migration job failed. The logs for data migration execution can be checked using the following command:

   ```
   kubectl logs <migration-pod-name> -n
   <netbackup-environment-namespace>
   ```

   User can copy the logs to retain them even after job pod deletion using the following command:

   ```
   kubectl logs <migration-pod-name> -n
   <netbackup-environment-namespace> > jobpod.log
   ```

Resolving an issue of failed probes

If a pod is not in a ready state for a **long** time, the `kubectl describe pod/<podname> -n <namespace>` command displays the following errors:

- ```
  Readiness probe failed: The readiness of
  the external dependencies is not set.

  Server setup is still in progress.
  ```

- ```
  Liveness probe failed: bpps command did
  not list nbwmc process. nbwmc is not alive.

  The Primary server is unhealthy.
  ```

To resolve an issue of failed probes

1. If you are deploying NetBackup on Kubernetes Cluster for the first time, check the installation logs for detailed error.

   Use any of the following methods:

   - Execute the following command in the respective primary server or media server pod and check the logs in `/mnt/nblogs/setup-server.logs`:

     ```
     kubectl exec -it -n <namespace> <pod-name> -- /bin/bash
     ```

   - Run the `kubectl logs pod/<podname> -n <namespace>` command.

2. Check pod events for obtaining more details for probe failure using the following command:

   ```
   kubectl describe pod/<podname> -n <namespace>
   ```

   Kubernetes will automatically try to resolve the issue by restarting the pod after liveness probe times out.

3. Depending on the error in the pod logs, perform the required steps or contact technical support.

Resolving issues when media server PVs are deleted

Media server describe events display the following error:

```
Warning  FailedAttachVolume  55m (x13 over 66m)
attachdetach-controller  AttachVolume.Attach failed for volume
"pvc-e16fd50a-1b6e-4b5b-ab53-76553ea87507" : rpc error: code =
NotFound desc = Volume not found, failed with error: Retriable: false,
 RetryAfter: 0s, HTTPStatusCode: 404, RawError:
{"error":{"code":"ResourceNotFound","message":"The Resource
'Microsoft.Compute/disks/pvc-e16fd50a-1b6e-4b5b-ab53-76553ea87507'
under resource group 'mc_nbu-k8s-network_nbux-deepak_eastus' was not
 found. For more details please go to
https://aka.ms/ARMResourceNotFoundFix"}}"
```

NetBackup media server and NetBackup primary server were in running state. Media server persistent volume claim or media server pod is deleted. In this case, reinstallation of respective media server can cause the issue.

To resolve the issues

1. Open the NetBackup Web UI using primary server hostname given in the primary server CR status.
2. Pause the media server reconciler by setting the value as *paused: true* in **mediaServer** section in environment CR using the following command:

   ```
   helm upgrade cloudscale
   VRTSk8s-netbackup-11.2/helm/cloudscale-<version>.tgz -n
   <namespace> --reuse-values \

   --set environment.mediaServers[0].name=media1

   --set environment.mediaServers[0].nodeSelector.labelKey=<original
   value> \

   --set
   environment.mediaServers[0].nodeSelector.labelValue=<original
   value>

   --set environment.mediaServers[0].replicas=<original value> \

   --set environment.mediaServers[0].minimumReplicas=<original value>
   --set environment.mediaServers[0].storage.data.capacity=<original
   value> \

   --set
   environment.mediaServers[0].storage.data.storageClassName=<sc-name>
   \

   --set environment.mediaServers[0].tag=11.2

   --set environment.mediaServers[0].storage.log.capacity=<original
   value> \

   --set
   environment.mediaServers[0].storage.log.storageClassName=<sc-name>
   \

   --set environment.mediaServers[0].paused=true
   ```

   The values for `<original value>` in the above command can be viewed using the `helm get values cloudscale -n netbackup` command prior to invoking the commands in the step 2 above.

   Save the changes.
3. Delete respective media server pod using `kubectl delete <pod-name> -n <namespace>` command.

4 Delete data and logs PVC for respective media server only using the `kubectl delete pvc <pvc-name> -n <namespace>` command.

5 Un pause the media server reconciler by changing the value as *paused: false* in **mediaServer** section in environment CR using the following command:

```
helm upgrade cloudscale
VRTSk8s-netbackup-11.2/helm/cloudscale-<version>.tgz -n
<namespace> --reuse-values \

--set environment.mediaServers[0].name=media1

--set environment.mediaServers[0].nodeSelector.labelKey=<original
value> \

--set
environment.mediaServers[0].nodeSelector.labelValue=<original
value>

--set environment.mediaServers[0].replicas=<original value> \

--set environment.mediaServers[0].minimumReplicas=<original value>

--set environment.mediaServers[0].storage.data.capacity=<original
value> \

--set
environment.mediaServers[0].storage.data.storageClassName=<sc-name>
\

--set environment.mediaServers[0].tag=11.2

--set environment.mediaServers[0].storage.log.capacity=<original
value> \

--set
environment.mediaServers[0].storage.log.storageClassName=<sc-name>
\

--set environment.mediaServers[0].paused=false
```

Save the changes.

Resolving an issue related to insufficient storage

Setup-server.logs of NetBackup primary server displays an error.

Insufficient storage on the node can cause this issue. Minimum hardware requirements for NetBackup may not be completed. During fresh deployment of primary server, the following error is displayed in `/mnt/nblogs/setup-server.logs`:

```
DBSPAWN ERROR: -86
Not enough memory to start
```

To resolve an issue related to insufficient storage

1. Create a new nodepool with the hardware specifications as mentioned in the *Cloud Scale Technology Manual Deployment Guide for Kubernetes Clusters.*.
2. Run the `kubectl get nodes` command to ensure that the nodes from the newly created nodepool are used in your cluster.
3. Update the **nodeSelector** spec in primary section using the following command:

   ```
   helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup
   --reuse-values --set
   environment.primary.nodeSelector.labelKey=agentpool --set
   environment.primary.nodeSelector.labelValue=nbupool
   ```

Resolving an issue related to invalid nodepool

Invalid nodepool is mentioned in primary server or media server CR nodeSelector spec. Due to this, primary server or media server pod fails to schedule.

The following error is displayed:

```
Error: Did not match Pod’s node affinity/selector.
```

To resolve an issue related to invalid nodepool

1. For media server CR, ensure that you copy spec information of the media server CR. The spec information is used to reapply the media server CR.
2. Edit the environment CR using the following command in **primary/mediaServer** section in environment CR update the **nodeSelector**, and save the changes

   ```
   helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup
   --reuse-values --set
   environment.primary.nodeSelector.labelKey=agentpool --set
   environment.primary.nodeSelector.labelValue=nbupoo
   ```

Resolve an issue related to KMS database

Installation logs at `/mnt/nblogs/setup-server.logs` display an error message with other details. In this scenario, you must configure KMS manually.

```
Error: Failed to create KMS database
```

To resolve this issue, execute the following command in the primary server pod:

```
kubectl exec -it -n <namespace> <primary-server-pod-name> -- /bin/bash
```

Refer the NetBackup Security and Encryption Guide for configure KMS manually:

For other troubleshooting issue related to KMS, refer the NetBackup Troubleshooting Guide.

Resolve an issue related to pulling an image from the container registry

Primary or media server failed to deploy with **ImagePullBackOff** error. If the pod Status field displays **ImagePullBackOff**, it means that the pod could not start because Kubernetes cannot pull a container image. A misspelled registry or image name or image registry being not reachable can cause a **ImagePullBackOff** status.

Run the `$ k get all -n netbackup-operator-system` command.

The output should look like:

```
NAME                         READY   STATUS             RESTARTS   AGE
pod/msdp-operator
-controller-manager-
65d8fd7c4d-bsgms             2/2     Running            0          7m9s

pod/netbackup-operator
-controller-manager-
5df6f58b9b-6ftt9             1/2     ImagePullBackOff   0          13s
```

For additional details, use the following command:

```
$ kubectl describe pod/<pod_name> -n netbackup-operator-system
```

Resolve this issue using any of the following methods:

- Check if image name and tag are correct. If not, edit and update the environment CR using the `helm upgrade cloudscale cloudscale-<version>.tgz -n netbackup --reuse-values --set environment.primary.tag=<tag>` command with correct image name and tag, and then save the changes.
- Check if the user is authorized and has permissions to access the Azure/AWS container registry.

Resolving an issue related to recovery of data

If a PVC is deleted or the namespace where primary or media server is deployed, is deleted or deployment setup is uninstalled, and you want to recover the previous data, attach the primary server and media server PVs to its corresponding PVCs.

In case of recovering data from PV, you must use the same environment CR specs that are used at the time of previous deployment. If any spec field is modified, data recovery may not be possible.

To resolve an issue related to recovery of data

1 Run the `kubectl get PV` command.

2 From the output list, note down PV names and its corresponding claim (PVC name and namespace) that are relevant from previous deployment point of view.

3 Set claim ref for the PV to null using the `kubectl patch pv <pv name> -p '{"spec":{"claimRef": null}}'` command.

For example, `kubectl patch pv pvc-4df282e2-b65b-49b8-8d90-049a27e60953 -p '{"spec":{"claimRef": null}}'`

4 Run the `kubectl get PV` command and verify bound state of PVs is **Available**.

5 For the PV to be claimed by specific PVC, add the **claimref spec** field with PVC name and namespace using the `kubectl patch pv <pv-name> -p '{"spec":{"claimRef": {"apiVersion": "v1", "kind": "PersistentVolumeClaim", "name": "<Name of claim i.e. PVC name>", "namespace": "<namespace of pvc>"}}}'` command.

For example,

```
kubectl patch pv <pv-name> -p '{"spec":{"claimRef": {"apiVersion":
"v1", "kind": "PersistentVolumeClaim", "name":
"data-testmedia-media-0", "namespace": "test"}}}'
```

While adding claimRef add correct PVC names and namespace to respective PV. Mapping should be as it was before deletion of the namespace or deletion of PVC.

6 Deploy environment CR that deploys the primary server and media server CR internally.

Check primary server status

Check the primary server custom resource status, with the command:

```
$ kubectl get primaryserver.netbackup.veritas.com/environment-sample
-n <namespace>
```

```
NAME                TAG    AGE    STATUS
environment-sample  10.0   3h1m   Failed
```

If the output shows STATUS as *Failed* as in the example above, check the primary pod log for errors with the command:

```
$ kubectl logs pod/environment-sample-primary-0 -n <namespace>
```

Pod status field shows as pending

If the pod Status field shows Pending state, it indicates that Kubernetes is not able to schedule the pod. To check use the following command:

```
$ kubectl get all -n netbackup-operator-system
```

The output is something like:

```
NAME                         READY   STATUS    RESTARTS   AGE
pod/msdp-operator-
controller-manager-
65d8fd7c4d-bsgms               2/2     Running   0          12m

pod/netbackup-
operator-controller-
manager-6c9dc8d87f
-pq8mr                         0/2     Pending   0          15s
```

For more details use the following pod describe command:

```
$ kubectl describe
pod/netbackup-operator-controller-manager-6c9dc8d87f-pq8mr -n
netbackup-operator-system
```

The output is something like:

```
  Type     Reason            Age                  Message
  ----     ------            ----                    -------
  Warning  FailedScheduling  56s (x3 over 2m24s)    0/4 nodes are
                                                    available:1 node(s)
                                                    had taint {node-
                                                    role.kubernetes.
                                                    io/master: }, that
                                                    the pod didn't
                                                    tolerate, 3 node(s)
                                                    didn't match
                                                    Pod's node
                                                    affinity/selector.
```

To resolve this issue, you can edit the operator deployment using the following command and verify the nodeSelector:

```
kubectl edit deployment netbackup-operator-controller-manager -n
netbackup-operator-system
```

Ensure that the container is running the patched image

There are three copies of the container image present in the Kubernetes environment during deployment or patching.

The first image copy is created on a local docker instance during image load operation. To check this copy, perform the following:

1 Run:

```
$ docker load -i images/pdk8soptr-20.5.tar.gz
```

Sample output:

```
Loaded image: msdp-operator:20.5
```

2 Taking the image name from step 1, run:

```
$ docker image ls | grep msdp-operator
```

Sample output:

```
msdp-operator 20.5 353d2bd50105 2 days ago 480 MB
```

3 Taking the value from step 2, run:

```
$ docker inspect 353d2bd50105 | jq .[].Id

"sha256:353d2bd50105cbc3c61540e10cf32a152432d5173bb6318b8e"
```

The second copy is created in the following respective registry's:

- (*AKS-specific*): Azure Container Registry (ACR)
- (*EKS-specific*): Amazon Elastic Container Registry (ECR)

To check this copy, perform the following:

1. Keep the image name and version same as original, run:

   (*AKS-specific*): `$ docker image tag msdp-operator:20.5 testregistry.azurecr.io/msdp-operator:20.5`

   (*EKS-specific*): `$ docker image tag msdp-operator:20.5 046777922665.dkr.ecr.us-east-2.amazonaws.com/nbuxeksdeploy.aws.io/msdp-operator:20.5`

2. Run:

   ```
   $ docker image ls | grep msdp-operator
   ```

   Sample output:

   (***AKS-specific***):

   ```
   msdp-operator  20.5 353d2bd50105 2 days ago 480 MB
    tregistry.azurecr.io/msdp-operator 20.5 353d2bd50105 2 days ago
    480 MB
   ```

   (***EKS-specific***):

   ```
   msdp-operator  20.5 353d2bd50105 2 days ago 480 MB
   msdp-operator            20.5 046777922665.dkr.ecr.us-east-2.
   amazonaws.com/nbuxeksdeploy.aws.io/msdp-operator
   20.5 353d2bd50105 2 days ago 480 MB /msdp-operator 20.5
   353d2bd50105 2 days ago 480 MB
   ```

3. To push the image to the registry, run:

   (*AKS-specific*): `$ docker push testregistry.azurecr.io/msdp-operator`

   (*EKS-specific*): `$ docker push testregistry.<account id>.dkr.ecr.<region>.amazonaws.com/<registry>:<tag>.io/msdp-operator`

   The push refers to a repository *[testregistry.azurecr.io/msdp-operator]*

   - (*AKS-specific*): *[testregistry.azurecr.io/msdp-operator]*
   - (*EKS-specific*): *[046777922665.dkr.ecr.us-east-2.amazonaws.com/nbuxeksdeploy.aws.io/msdp-operator]*

   0a504041c925: Layer already exists

   ```
   20.5: digest:
   sha256:d294f260813599562eb5ace9e0acd91d61b7dbc53c3 size:
    2622
   ```

4 To verify local image digest after the push operation, run:

```
$ docker inspect 353d2bd50105 | jq .[].RepoDigests
```

Sample output:

(***AKS-specific***):

```
[
   "testregistry.azurecr.io/msdp-operator@sha256:
d294f260813599562eb5ace9e0acd91d61b7dbc53c3"
 ]
```

(***EKS-specific***):

```
[
   "testregistry.<account
id>.dkr.ecr.<region>.amazonaws.com/<registry>:<tag>.io/
msdp-operator@sha256: d294f260813599562eb5ace9e0acd91d61b7dbc53c3"

 ]
```

5 To verify image presence in the registry, run:

(*AKS-specific*): `$ az acr repository list --name testregistry`

(*EKS-specific*): `$ aws ecr describe-repositories --repository-names "veritas/main_test1"`

Sample output:

(***AKS-specific***):

```
[
  "msdp-operator",
]
```

(***EKS-specific***):

```
"repositories": [
        {
           "repositoryArn": "arn:aws:ecr:us-east-2:046777922665:

repository/veritas/main_test1",
            "registryId": "046777922665",
            "repositoryName": "veritas/main_test1",
            "repositoryUri": "046777922665.dkr.ecr.us-east-2.

amazonaws.com/veritas/main_test1",
            "createdAt": "2022-04-13T07:27:52+00:00",
            "imageTagMutability": "MUTABLE",
            "imageScanningConfiguration": {
                "scanOnPush": false
            },
            "encryptionConfiguration": {
                "encryptionType": "AES256"
            }
        }
    ]
```

6 To verify image digest in registry, run:

(*AKS-specific*): `$ az acr repository show -n testregistry --image msdp-operator:20.5`

(*EKS-specific*): `$ aws ecr describe-images --registry-id 046777922665 --repository-name "veritas/main_test1" --image-ids imageTag=latestTest5`

Sample output:

(***AKS-specific***):


```
{
   "changeableAttributes": {
     "deleteEnabled": true,
     "listEnabled": true,
     "readEnabled": true,
     "writeEnabled": true
   },
   "createdTime": "2022-02-01T13:43:26.6809388Z",
   "digest": "sha256:d294f260813599562eb5ace9e0acd91d61b7dbc53c3",
   "lastUpdateTime": "2022-02-01T13:43:26.6809388Z",
   "name": "20.5",
   "signed": false
 }
```


(***EKS-specific***):


```
"imageDetails": [
        {
            "registryId": "046777922665",
            "repositoryName": "veritas/main_test1",
            "imageDigest":
"sha256:d0095074286a50c6bca3daeddbaf264cf4006a92fa3a074daa4739cc995b36f8",
            "imageTags": [
                "latestTest5"
            ],
            "imageSizeInBytes": 38995046,
            "imagePushedAt": "2022-04-13T15:56:07+00:00",
            "imageManifestMediaType": "application/vnd.docker.
distribution.manifest.v2+json",
            "artifactMediaType": "application/vnd.docker.container.image.v1+json"
        }
    ]
```

The third copy is located on a Kubernetes node running the container after it is pulled from the registry. To check this copy, perform the following:

1 Run;

```
$ kubectl get nodes -o wide
```

(***AKS-specific***):

```
NAME            STATUS  VERSION   INTERNAL-IP   OS-IMAGE

aks-agentpool-7601-vmss000 Ready v1.21.7 10.240.0.4 Ubuntu 20.54.6
 LTS
```

(***EKS-specific***):

```
NAME            STATUS  VERSION   INTERNAL-IP   OS-IMAGE

eks-agentpool-7601-vmss000 Ready v1.21.7 10.240.0.4 Ubuntu 20.54.6
 LTS
```

2 Use kubectl debug to run a container on the node:

(*AKS-specific*): `$ kubectl debug node/aks-nodepool1-7601-vmss000-it --image=mcr.microsoft.com/aks/fundamental/base-ubuntu:v0.0.11 root@aks-agentpool-7601-vmss000:/#`

(*EKS-specific*): `$ kubectl debug node/eks-nodepool1-7601-vmss000-it --image=mcr.microsoft.com/eks/fundamental/base-ubuntu:v0.0.11 root@eks-agentpool-7601-vmss000:/#`

3 You can interact with the node session from the privileged container:

```
chroot /host
```

4 Verify the presence of the image:

```
/usr/local/bin/crictl image | grep msdp
```

Sample output:

(***AKS-specific***):

```
testregistry.azurecr.io/msdp-operator  20.5  353d2bd50105c  182MB
```

(***EKS-specific***):

```
<account id>.dkr.ecr.<region>.amazonaws.com/msdp-operator  20.5
 353d2bd50105c  182MB
```

5 Verify the image ID on the Kubernetes node, run:

```
/usr/local/bin/crictl inspecti 353d2bd50105c | jq .[].id
```

Sample output

```
"sha256:353d2bd50105cbc3c61540e10cf32a152432d5173bb6318b8e"
null
```

6 Verify the image digest on the Kubernetes node, run:

```
/usr/local/bin/crictl inspecti 353d2bd50105c | jq .[].repoDigests
```

Sample output

(***AKS-specific***):

```
[
  "testregistry.azurecr.io/msdp-operator@sha256:
d294f260813599562eb5ace9e0acd91d61b7dbc53c3"
]
null
```

(***EKS-specific***):

```
[
  "<account
id>.dkr.ecr.<region>.amazonaws.com/msdp-operator@sha256:
d294f260813599562eb5ace9e0acd91d61b7dbc53c3"
]
null
```

How to make sure that you are running the correct image

Use the steps given above to identify image ID and Digest and compare with values obtained from the registry and the Kubernetes node running the container.

Note: MSDP Scaleout images (uss-engine, uss-mds, uss-controller, msdp-operator) use IfNotPresent imagePullPolicy. A unique image tag is required in order for a Kubernetes node to pull an updated image.

Getting EEB information from an image, a running container, or persistent data

To view the list of installed EEBs, run the `nbbuilder` script provided in the EEB file archive.

(*AKS-specific*): `$ bash nbbuilder.sh -registry_name testregistry.azurecr.io -list_installed_eebs -nb_src_tag=10.3 -msdp_src_tag=18.0`

(*EKS-specific*): `$ bash nbbuilder.sh -registry_name <account id>.dkr.ecr.<region>.amazonaws.com -list_installed_eebs -nb_src_tag=10.3 -msdp_src_tag=18.0`

Sample output:

```
Wed Feb 2 20:48:13 UTC 2022: Listing strings for EEBs
installed in <account id>.dkr.ecr.<region>.amazonaws.com/netbackup/main:10.3.
EEB_NetBackup_10.3Beta6_PET3980928_SET3992004_EEB1
EEB_NetBackup_10.3Beta6_PET3980928_SET3992021_EEB1
EEB_NetBackup_10.3Beta6_PET3980928_SET3992022_EEB1
EEB_NetBackup_10.3Beta6_PET3980928_SET3992023_EEB1
EEB_NetBackup_10.3Beta6_PET3992020_SET3992019_EEB2
EEB_NetBackup_10.3Beta6_PET3980928_SET3992009_EEB2
EEB_NetBackup_10.3Beta6_PET3980928_SET3992016_EEB1
EEB_NetBackup_10.3Beta6_PET3980928_SET3992017_EEB1
Wed Feb 2 20:48:13 UTC 2022: End
Wed Feb 2 20:48:13 UTC 2022: Listing strings for EEBs
installed in <account id>.dkr.ecr.<region>.amazonaws.com/uss-controller:18.0.
EEB_MSDP_18.0_PET3980928_SET3992007_EEB1
EEB_MSDP_18.0_PET3992020_SET3992019_EEB2
EEB_MSDP_18.0_PET3980928_SET3992010_EEB2
Wed Feb 2 20:48:14 UTC 2022: End
Wed Feb 2 20:48:14 UTC 2022: Listing strings for EEBs
installed in <account id>.dkr.ecr.<region>.amazonaws.com/uss-engine:18.0.
EEB_MSDP_18.0_PET3980928_SET3992006_EEB1
EEB_MSDP_18.0_PET3980928_SET3992023_EEB1
EEB_MSDP_18.0_PET3992020_SET3992019_EEB2
EEB_MSDP_18.0_PET3980928_SET3992009_EEB2
EEB_MSDP_18.0_PET3980928_SET3992010_EEB2
EEB_MSDP_18.0_PET3980928_SET3992018_EEB1
Wed Feb 2 20:48:14 UTC 2022: End
Wed Feb 2 20:48:14 UTC 2022: Listing strings for EEBs
installed in <account id>.dkr.ecr.<region>.amazonaws.com/uss-mds:18.0.
EEB_MSDP_18.0_PET3980928_SET3992008_EEB1
EEB_MSDP_18.0_PET3992020_SET3992019_EEB2
EEB_MSDP_18.0_PET3980928_SET3992010_EEB2
Wed Feb 2 20:48:15 UTC 2022: End
```

Alternatively, if the `nbbuilder` script is not available, you can view the installed EEBs by executing the following command:

```
$ docker run --rm <image_name>:<image_tag> cat
/usr/openv/pack/pack.summary
```

Sample output:

```
EEB_NetBackup_10.1Beta6_PET3980928_SET3992004_EEB1
EEB_NetBackup_10.3Beta6_PET3980928_SET3992021_EEB1
EEB_NetBackup_10.3Beta6_PET3980928_SET3992022_EEB1
EEB_NetBackup_10.3Beta6_PET3980928_SET3992023_EEB1
EEB_NetBackup_10.3Beta6_PET3992020_SET3992019_EEB2
EEB_NetBackup_10.3Beta6_PET3980928_SET3992009_EEB2
EEB_NetBackup_10.3Beta6_PET3980928_SET3992016_EEB1
EEB_NetBackup_10.3Beta6_PET3980928_SET3992017_EEB1
```

To view all EEBs installed in a running container, run:

```
$ kubectl exec --stdin --tty <primary-pod-name> -n <namespace> --
cat /usr/openv/pack/pack.summary
```

Note: The pack directory may be located in different locations in the `uss-*` containers. For example: `/uss-controller/pack` , `/uss-mds/pack`, `/uss-proxy/pack`.

To view a list of installed data EEBs from a running container, run:

```
$ kubectl exec --stdin --tty <primary-pod-name> -n <namespace> --
cat /mnt/nbdata/usr/openv/pack/pack.summary
```

Resolving the certificate error issue in NetBackup operator pod logs

Following error is displayed in NetBackup operator pod logs when the primary server certificate is changed:

```
ERROR controller-runtime.manager.controller.environment
Error defining desired resource {"reconciler group": "netbackup.veritas.com",
"reconciler kind": "Environment", "name": "test-delete", "namespace":
"netbackup-environment",
"Type": "MSDPScaleout", "Resource": "dedupe1", "error": "Unable to get primary host
UUID:
Get \"https://nbux-xx-xxx-xx-xx.vxindia.cohesity.com:1556/netbackup/config/hosts\":
x509: certificate signed by unknown authority (possibly because of \"crypto/rsa:
verification error\" while trying to verify candidate authority certificate \"nbatd\")"}
```

To resolve this issue, restart the NetBackup operator by deleting the NetBackup operator pod using the following command:

```
kubectl delete <Netbackup-operator-pod-name> -n <namespace>
```

Issues with NetBackup Certificate Authority (NBCA)

This section provides the issues and their respective workaround for the issues observed with NBCA.

Certificate bundle upload fails on primary server pod

The following error appears in the primary server pod **nbcert** logs when the certificate bundle upload to the NetBackup web services pod fails:

```
Failed to push cert bundle to WS pod
    Invalid X-NBU-CERT-BUNDLE-SECRET
    Failed to create tar buffer
    Failed to compress bundle file
```

To resolve this issue:

1. Ensure that the NetBackup web services pod is running and is in a healthy state before retrying the certificate bundle upload.
2. If the following error is reported in the **nbcert** logs, verify that the NetBackup operator is running and not reporting errors:

   ```
   Invalid X-NBU-CERT-BUNDLE-SECRET
   ```

   To regenerate the POST token, delete the secret using the following command:

   ```
   kubectl delete secret nb-post-secret -n <namespace>
   ```

3. Restart the primary server pod to trigger a retry of the certificate bundle upload.
4. If the following errors are reported in the **nbcert** logs, verify that sufficient disk space is available on the primary server pod:

   ```
   Failed to create tar buffer
   Failed to compress bundle file
   ```

   After resolving any disk space issues, retry the certificate bundle push using the following command:

   ```
   nbcertcmd -pushCertBundle
   ```

Standard pods using old or missing certificates

One or more standard NetBackup pods are unable to establish secure communication due to missing or outdated security certificates.

The following errors are reported in the **nbcert** logs of the **secure-comm** container in **media**, **bpdbm**, **ws**, or **policyjob** pods:

```
x509: certificate signed by unknown authority
    Invalid X-NBU-CERT-BUNDLE-SECRET
    Failed to download certificate bundle from WS pod with error code
```

To resolve this issue:

1. Check the primary server pod **nbcert** logs to confirm that the certificate bundle push completed successfully:

   ```
   Successfully pushed cert bundle to WS pod
   ```

2. On the affected pod, initiate an immediate certificate pull using the following command:

   ```
   nbcertcmd -pullCertBundle
   ```

3. If the Invalid X-NBU-CERT-BUNDLE-SECRET error is present in the **nbcert** logs, verify that the NetBackup operator is running and not reporting errors.

   If the issue persists, delete the FETCH token secret to force regeneration by using the following command:

   ```
   kubectl delete secret nb-fetch-secret -n <namespace>
   ```

4. If the certificate issue continues, restart the affected pod to retry certificate retrieval.

5. As a fallback, certificate retrieval is automatically retried every 4 hours.

Pod startup hangs with an error

A NetBackup pod does not complete startup and remains in a waiting state.

The following error message is displayed:

```
Waiting for secret files to be available
```

To resolve this issue:

1. Ensure that the required Kubernetes secrets exist and are populated in the correct namespace:

   ```
   kubectl get secret nb-post-secret nb-fetch-secret nb-ca-cert -n
   <namespace>
   ```

2. Ensure that the **nbatd** pod is running.

3 Restart the NetBackup operator to force secret reconciliation:

```
kubectl delete pod <netbackup-operator-pod> -n
<operator-namespace>
```

4 Restart the affected pod once secrets are available.

Missing CA certificate in the nb-ca-cert Kubernetes secret

This issue occurs when the CA certificate is not populated in the Kubernetes **nb-ca-cert** secret.

To resolve this issue:

1 Verify that the **nbatd** pod is running by using the following command:

```
kubectl get pods -n <namespace> | grep nbatd
```

2 Verify that the required Kubernetes secret exists and is populated by using the following command:

```
kubectl get secret nb-ca-cert -n <namespace>
```

3 Restart the NetBackup operator to trigger a retry of CA certificate extraction.

4 Restart the affected pods after the CA certificate is successfully populated in the secret.

Certificate issues after upgrade or primary failover

After a NetBackup upgrade or primary server failover, certificate-related errors as follows may be observed in the **nbcert** logs:

```
x509: certificate signed by unknown authority
    Error: Failed to upload CA certificate bundle to CMS
    touch file not found
```

To resolve this issue:

1 Push certificates again from the new primary server by using the following command:

```
nbcertcmd -pushCertBundle
```

2 Restart the primary pod to reinitialize certificate distribution.

3 Verify that certificate files no longer reference NFS paths.

Pod restart failure due to liveness probe time-out

As part of liveness probe for primary and media pods, a health script runs inside the container to check the NetBackup health status.

When there is an issue with a container related to a full disk, CPU, or memory pressure, the liveness probe gets timed out because of no response from the health script. As a result, the Pod does not restart.

To resolve this issue, restart the Pod manually. Delete the Pod using the `kubectl delete pod/<podname> -n <namespace>` command.

The Pod is deleted and Kubernetes creates another Pod.

NetBackup messaging queue broker take more time to start

This issue is due to **nbmqbroker** service taking more time to start.

To resolve this issue, perform the following steps

1. Exec into the respective Primary Server pod using the following command:

   ```
   kubectl exec -it <pod-name> -n <namespace> -- /bin/bash
   ```

2. Check the nbmqbroker service logs which are in `/usr/openv/mqbroker/logs/` folder.

   If the value of **checking service start status count:** is more than the 75 then **nbmqbroker** would take more time to start.

3. Stop the nbmqbroker service using the following command:

   ```
   /usr/openv/mqbroker/bin/nbmqbroker stop
   ```

4. Open the `/usr/openv/mqbroker/bin/nbmqbroker` file.

5. Increase the value of **total_time** and **sleep_duration** and save the file.

6. Start the mqbroker service using the following command:

   ```
   /usr/openv/mqbroker/bin/nbmqbroker start
   ```

   If the Primary Server pod gets restarted then the user must perform the same above steps to increase the values of **total_time** and **sleep_duration**, as these values will not get persisted after pod restart.

Host mapping conflict in NetBackup

The following error message is displayed due to host mapping conflict in NetBackup:

```
..exited with status 7659: Connection cannot be established because
the host validation failed on the target host
```

In kubernetes deployment, communication to pod goes through multiple layers that is, load balancer, nodes and pod. In certain setups during communication host may get associated with certain IP and would be changed. That IP may get associated with some different pod and which causes conflict. The host mapping entries is in the form of `"::ffff:<ip address>"`

To resolve the issue of host mapping conflict in NetBackup

1. To resolve the conflict issue, refer to Mappings for Approval tab section of the *NetBackup Security and Encryption Guide*.
2. To remove the entries that are not valid, refer to Removing host ID to host name mappings section of the *NetBackup Security and Encryption Guide*.

Issue with capacity licensing reporting which takes longer time

The `nbdeployutil` utility does not perform well on EFS or Azure files based volumes. Specify different block storage based volume to get good performance.

To resolve the issue, perform the following:

1. For running report manually, pass `--parentdir /mnt/nbdb/<FOLDER_NAME>` to `nbdeployutil` command.
2. For changing `parentdir` to scheduled capacity reporting, provide a custom value in `nbdeployutilconfig.txt` file.
3. Create/Edit the `nbdeployutilconfig.txt` file located at `/usr/openv/var/global/` by adding the following entry:

```
[NBDEPLOYUTIL_INCREMENTAL]
PARENTDIR=/mnt/nbdb/<FOLDER_NAME>
```

Local connection is getting treated as insecure connection

The following error message is displayed when un-necessary audit events are get logged only when reverse dns lookup is enabled for primary and media Load Balancer service:

```
Host 'eaebbef2-57bc-483b-8146-1f6616622276' is trying to connect to
host '<serverName>.abc.com'. The connection is dropped, because the
host '<serverName>.abc.com' now appears to be NetBackup 8.0 or earlier
```

Primary and media servers are referred with multiple IP's inside the pod (pod IP/LoadBalancer IP). With reverse name lookup of IP enabled, NetBackup treats the local connection as remote insecure connection.

To resolve the audit events issue, disable the reverse name lookup of primary and media Load Balancer IP.

Backing up data from Primary server's `/mnt/nbdata/` directory fails with primary server as a client

Backing up data from Primary server's `/mnt/nbdata/` directory fails with primary server as a client.

From NetBackup version 10.1 onwards, the `/mnt/nbdata` directory would be on the following respective files:

- (*AKS-specific*): Azure file share
- (*EKS-specific*): EFS

The `/mnt/nbdata` directory of primary server is mounted using the NFS protocol.

Hence for backing up the data from `/mnt/nbdata` directory, change the policy attribute for such policies.

To resolve this issue, enable the **Follow NFS** check box in policy attribute.

Storage server not supporting Instant Access capability on Web UI after upgrading NetBackup

After you upgrade NetBackup 10.0 (10.0.0.1 or 10.1), 10.1.1, 10.2 to NetBackup 10.3and later or MSDP Scaleout to 18.0 or later, if you:

- find the storage server not supporting Instant Access capability on Web UI
  Or
- fail to select MSSQL recovery point to create MSSQL Instant Access on Web UI

Then perform the following steps manually to refresh the Instant Access capability in NetBackup:

1. Login to NetBackup primary server.
2. Execute the following commands to refresh the MSDP capabilities on NetBackup primary server:

   ```
   nbdevconfig -getconfig
   ```

   ```
   nbdevconfig -setconfig
   ```

   For example,

   ```
   /usr/openv/netbackup/bin/admincmd/nbdevconfig -getconfig -stype
   PureDisk -storage_server [storage server] >
   /tmp/tmp_pd_config_file
   ```

```
/usr/openv/netbackup/bin/admincmd/nbdevconfig -setconfig
-storage_server [storage server] -stype PureDisk -configlist
/tmp/tmp_pd_config_file
```

3. Restart the NetBackup Web Management Console service (nbwmc) on NetBackup primary server.

   For example,

   ```
   /usr/openv/netbackup/bin/nbwmc terminate
   ```

   ```
   /usr/openv/netbackup/bin/nbwmc start
   ```

Taint, Toleration, and Node affinity related issues in cpServer

The cpServer control pool pod is in pending state

If one of the following cpServer control pool pod is in pending state, then perform the steps that follow:

```
flexsnap-agent, flexsnap-api-gateway, flexsnap-certauth,
flexsnap-coordinator, flexsnap-idm, flexsnap-nginx,
flexsnap-notification, flexsnap-scheduler, flexsnap-listener,
flexsnap-postgresql, flexsnap-rabbitmq, flexsnap-fluentd-,
flexsnap-fluentd
```

1. Obtain the pending pod's toleration and affinity status using the following command:

   ```
   kubectl get pods <pod name>
   ```

2. Check if the node-affinity and tolerations of pod are matching with:

   - fields listed in **cpServer.nodepool.controlpool** or **primary.nodeselector** in the `cloudscale-values.yaml`file.
   - taint and label of node pool, mentioned in **cpServer.nodeselector.controlpool** or **primary.nodeselector** in the `cloudscale-values.yaml` file.

If all the above fields are correct and matching and still the control pool pod is in pending state, then the issue may be due to all the nodes in nodepool running at maximum capacity and cannot accommodate new pods. In such case the nodepool must be scaled properly.

The cpServer data pool pod is in pending state

If one of the following cpServer data pool pod is in pending state, then perform the steps that follow:

`flexsnap-workflow,flexsnap-datamover`

1. Obtain the pending pod's toleration and affinity status using the following command:

   ```
   kubectl get pods <pod name>
   ```

2. Check if the node-affinity and tolerations of pod are matching with:
   - fields listed in **cpServer.nodepool.datapool** in the `environment.yaml` file.
   - taint and label of node pool, mentioned in **cpServer.nodeselector.datapool** in the `cloudscale-values.yaml` file.

If all the above fields are correct and matching and still the control pool pod is in pending state, then the issue may be due to all the nodes in nodepool running at maximum capacity and cannot accommodate new pods. In such case the nodepool must be scaled properly.

The Snapshot Manager operator (flexsnap-operator) pod is in pending state

1. Obtain the pending pod's toleration and affinity status using the following command:

   ```
   kubectl get pods <pod name>
   ```

2. Check if the node-affinity and tolerations of pod are matching with:
   - fields listed in **operators-values.yaml** file.
   - taint and label of node pool, mentioned in above values.

If all the above fields are correct and matching and still the control pool pod is in pending state, then the issue may be due to all the nodes in nodepool running at maximum capacity and cannot accommodate new pods. In such case the nodepool must be scaled properly.

Nodes configured with incorrect taint and label

If the nodes are configured with incorrect taint and label values, the user can edit them using the following command provided for AKS as an example:

```
az aks nodepool update \ --resource-group <resource_group> \
--cluster-name <cluster_name> \ --name <nodepool_name> \ --node-taints
<key>=<value>:<effect> \ --no-wait
```

```
az aks nodepool update \ --resource-group <resource_group> \
--cluster-name <cluster_name> \ --name <cluster_name> \ --labels
<key>=<value>
```

Operations performed on cpServer in cloudscale-values.yaml file are not reflected

Operations such as add/remove/comment/uncomment performed on cpServer in `cloudscale-values.yaml` file are not reflected even after applying them. The reasons and solutions for the same are as follow:

- Check if the action is reflected in cpServer CRO (Custom Resource Object) by using the following command:

  ```
  kubectl describe cpserver n $ENVIRONMENT_NAMESPACE
  ```

  If changes are not reflected then , check environment operator logs and if changes are reflected then follow the next steps.
- Check if the flexsnap operator is running by using the following command:

  ```
  kubectl get pods -n $OPERATOR_NAMESPACE | grep flexsnap-operator
  | awk '{printf $1" " }
  ```
- The flexsnap operator is running and is already processing the event (Update, Upgrade, Create, Delete).
  - To check logs of running operator, use the following command:

    ```
    kubectl logs -f $(kubectl get pods -n $OPERATOR_NAMESPACE |
    grep flexsnap-operator | awk '{printf $1" " }')
    ```
  - If you still want to go ahead with new action, you can stop the processing of the current event so that the new events are processed. To do so delete the flexsnap operator pod using the following command:

    ```
    kubectl delete pod $(kubectl get pods -n $OPERATOR_NAMESPACE |
    grep flexsnap-operator | awk '{printf $1" " }')
    ```

    This will re-create the flexsnap-operator pod which will be ready to serve new events.

    Note: The newly created pod might have missed the event which was performed before re-creation of pod. In this case you may have to reapply `cloudscale-values.yaml`.

Elastic media server related issues

This section provides the issues and their respective workaround for the issues observed in elastic media server feature.

- **Jobs are going in queued state**

  This behavior can be observed in 10.5 or later versions. If the jobs are unexpectedly going in queued state -

- Check the Activity monitor logs for the specific job. Verify if it is due to maximum count being reached for the media server.
- Alternatively, operator logs can be checked and it may show following messages -

```
2024-06-18T18:16:33Z    INFO    Jobs are queued on all media servers due
2024-06-18T18:16:33Z    INFO    Found 19 jobs in queue due to job based
```

If the jobs are in queue due to this reason, the media server pods will scale out if the media server autoscaler is ON and the jobs will get assigned to scaled out pods. For more details on this configuration parameters, refer See "Elastic media server" on page 99..

- **Issue with autoscaler for scaling in the media server pods**
  This issue is observed when there is no load or only few jobs are running even when there are maximum number of media server pods/nodes that are in running state.
  - Verify if media server autoscaler is trying to scale-in but unable to shutdown the media server pods which are marked to be scaled-in.
  - Verify if there are any jobs or bpps processes running on the media pods with the higher indexed running pod by referring to the NetBackup operator logs as mentioned below:

```
2023-03-01T08:14:56.470Z        INFO
controller-runtime.manager.controller.mediaserver      Running
 jobs 0:  on Media Server
nbux-xx-xxx-xx-xx.vxindia.cohesity.com.        {"reconciler
group": "netbackup.veritas.com", "reconciler kind":
"MediaServer", "name": "media1", "namespace":
"netbackup-environment", "Media Server":
"nbux-xx-xxx-xx-xx.vxindia.cohesity.com"}
2023-03-01T08:14:56.646Z        INFO
controller-runtime.manager.controller.mediaserver       bpps
processes running status. false:  on Media Server
nbux-xx-xxx-xx-xx.vxindia.cohesity.com.  {"reconciler group":
"netbackup.veritas.com", "reconciler kind": "MediaServer",
"name": "media1", "namespace": "netbackup-environment", "Media
 Server": "nbux-xx-xxx-xx-xx.vxindia.cohesity.com"}
```

Perform the following to know which bpps processes are running and are not allowing to scale-in the media server pod:

  - Login to NetBackup Web UI portal.

- Check the notifications tab for any notifications of **Media server elasticity event** category. The notification has the list of additional process running on specific media server. User must wait until the process listed in the additional process running exits.

  Alternatively, user can also see the list of processes in the NetBackup operator logs as follows:

  ```
  2023-07-11T13:33:44.142Z  INFO
  controller-runtime.manager.controller.mediaserver
  Following processes are still running : bpbkar test1, bpbkar
   test2    {"reconciler group": "netbackup.veritas.com",
  "reconciler kind": "MediaServer", "name": "test-media-server",
   "namespace": "netbackup-environment"}
  ```

- Verify if there are any jobs in queued state due to the maximum jobs per media server setting. If yes, the scale-in should happen when there are no more queued jobs.

- **Duplication job is not getting completed**

  Duplication job is not getting completed (in turn scale-down is not happening) because **MachineAdministrativePause** is set on the media server as part of scale down while duplication is still running.

  While configuring destination storage unit, manually select media servers which are always up, running and would not be able to scale-in anytime. Number of media server which are always running would be same as that mentioned against **minimumReplicas** field in CR. Above recommendation also applies when upgrading from older version to NetBackup version 10.3. Post-upgrade manually select media servers that are mentioned against**minimumReplicas** field in CR during upgrade. Default **minimumReplicas** is 1.

- **Error while connecting to media server**

  For scaled in media servers, certain resources and configurations are retained to avoid reconfiguration during subsequent scale out. Post entries for scaled in media servers are not removed from NetBackup primary server and hence if those media servers are used for any operation, connectivity issue is observed.

  Workaround:

  It is recommenced to use media servers that are always up, running and would never scale in (by the media server autoscaler). Number of media servers that are always up and running would be same as that of the value mentioned in **minimumReplicas** field in CR.

Failed to register Snapshot Manager with NetBackup

NetBackup cloud scale installation failed due to Snapshot Manager registration failure with NetBackup.

The environment CustomResourceDefinitions (CRD) marks the NetBackup installation as failed due to the failure status of cpServer:

- Case 1: The status of cpServer CRD is displayed as failed with the following event:

  ```
  Failure OperationFailed 46m flexsnap-operator [Snapshot Manager
  registration with NetBackup] failed = Failed to register Snapshot
  Manager with NetBackup
  ```

  Or

- The `flexsnap-operator` encountered the following log:

  ```
  Mar 17 00:56:39 aks-cpcontrol-15585928-vmss000000
  flexsnap-operator[9] MainThread flexsnap.util: ERROR - Failed to
  register Snapshot Manager to NetBackup!
  {'errorCode': 9875, 'errorMessage': 'Failed to add preconfigured
  plugins from Snapshot Manager.', 'attributeErrors': {},
  'fileUploadErrors': [], 'errorDetails':
  []}. Response status code: 500

  Mar 17 00:57:00 aks-cpcontrol-15585928-vmss000000
  flexsnap-operator[9] MainThread flexsnap.util: ERROR - Failed to
  register Snapshot Manager to NetBackup!
  {'errorCode': 9858, 'errorMessage': 'Failed to add NetBackup with
   Snapshot Manager.', 'attributeErrors': {}, 'fileUploadErrors':
  [], 'errorDetails': []}.
  Response status code: 500

  Mar 17 01:00:14 aks-cpcontrol-15585928-vmss000000
  flexsnap-operator[9] MainThread flexsnap.util: ERROR - Failed to
  register Snapshot Manager to NetBackup!
  {'errorCode': 9896, 'errorMessage': 'Failed to add NetBackup with
   Snapshot Manager due to the failure in certificate generation.',
   'attributeErrors': {},
  'fileUploadErrors': [], 'errorDetails': []}. Response status code:
   500
  ```

Workaround:

Manually register Snapshot Manager with NetBackup by performing the following steps:

- Navigate to **NetBackup UI > Workload > Cloud > Snapshot Manager** and click on **Add**.
- Enter the values for FQDN of Snapshot Manager and the port (Default: 443).
- Click **Save**.

Note: Even after Snapshot Manager is registered with NetBackup manually the status of cpServer CRD would be displayed as failed. This status does not affect the working of Snapshot Manager.

Post Kubernetes cluster restart, flexsnap-listener pod went into CrashLoopBackoff state or pods were unable to connect to flexsnap-rabbitmq

After Kubernetes cluster restart, `flexsnap-listener` pod went into **CrashLoopBackoff** state or pods such as `flexsnap-coordinator`, `flexsnap-agent` and so on were unable to connect to `flexsnap-rabbitmq`.

This issue can be resolved by restarting the NetBackup Snapshot Manager services in the following order using the `kubectl delete <pod-name> -n <namespace>` command:

`flexsnap-certauth`

`flexsnap-rabbitmq`

`flexsnap-postgres`

`flexsnap-listener` and so on.

Post Kubernetes cluster restart, issues observed in case of containerized Postgres deployment

After Kubernetes cluster restart in case of containerized Postgres deployment, couple of `flexsnap` pods went into **CrashLoopBackoff** state and primary pod was not started successfully.

This issue arises due to certificate expiration when the cluster was down. Perform the following to ensure that the issue occurred was due to certificate expiration:

- Run the following command to exec into the `nb-postgresql` pod:

  ```
  kubectl -n <namespace> exec pod/nb-postgresql-0 -it bash
  ```

  Run the following command from within the Postgres pod

```
psql "host=nb-postgresql port=13787 dbname=postgres user=postgres
 sslmode=verify-full
sslcert='/netbackup/postgresql/keys/server/tls.crt'
sslkey='/netbackup/postgresql/keys/server/tls.key'
sslrootcert=/netbackup/postgresql/keys/server/ca.crt"
```

Following error message is displayed:

```
psql: error: connection to server at "nb-postgresql" (10.240.3.27),
 port 13787 failed: SSL error: certificate verify failed.
```

- Run the following command to check the `ca.crt` server certificate:
  - Get the `ca.crt` of **postgresql-server-crt**:
    `kubectl -n <namespace> get secret postgresql-server-crt -o "jsonpath={.data['ca\.crt']}" | base64 -d > server_ca.crt`
  - Get the `ca.crt` of **postgresql-netbackup-ca**:
    `kubectl -n <namespace> get secret postgresql-netbackup-ca -o "jsonpath={.data['ca\.crt']}" | base64 -d > ca.crt`
  - Get current date on the system:
    `date`
    Following message is displayed:
    ```
    Tue Feb 27 17:07:14 UTC 2024
    ```
  - Check expiry of `ca.crt`:
    `openssl x509 -enddate -noout -in ca.crt`
    Following message is displayed:
    ```
    notAfter=Feb 27 17:52:58 2024 GMT
    ```
  - Check expiry of `server_ca.crt`:
    `openssl x509 -enddate -noout -in server_ca.crt`
    Following message is displayed:
    ```
    notAfter=Feb 27 16:57:58 2024 GMT
    ```

  The above displayed messages indicate that `ca.crt` of **postgresql-server-crt** is referring to an expired certificate.

Resolution:

Perform the following to resolve the certificate issue:

1. Delete the certificate secrets:

```
kubectl -n <namespace> delete secret postgresql-netbackup-ca
postgresql-server-crt postgresql-client-crt
```

2. Execute the following commands to ensure that the correct certificate is referred:
   - Get the `ca.crt` of **postgresql-server-crt**:

     ```
     kubectl -n <namespace> get secret postgresql-server-crt -o
     "jsonpath={.data['ca\.crt']}" | base64 -d > server_ca.crt
     ```

   - Get the `ca.crt` of **postgresql-netbackup-ca**:

     ```
     kubectl -n <namespace> get secret postgresql-netbackup-ca -o
     "jsonpath={.data['ca\.crt']}" | base64 -d > ca.crt
     ```

   - Get current date on the system:

     ```
     date
     ```

     Following message is displayed:

     ```
     Tue Feb 27 17:11:59 UTC 2024
     ```

   - Check expiry of `ca.crt`:

     ```
     openssl x509 -enddate -noout -in ca.crt
     ```

     Following message is displayed:

     ```
     notAfter=Feb 27 18:11:38 2024 GMT
     ```

   - Check expiry of `server_ca.crt`:

     ```
     openssl x509 -enddate -noout -in server_ca.crt
     ```

     Following message is displayed:

     ```
     notAfter=Feb 27 18:11:38 2024 GMT
     ```

   The above displayed messages indicate that `ca.crt` of **postgresql-server-crt** is referring to the correct certificate.

3. Run the following command to exec into the `nb-postgresql` pod:

   ```
   kubectl -n <namespace> exec pod/nb-postgresql-0 -it bash
   ```

   Run the following command from within the Postgres pod

   ```
   psql "host=nb-postgresql port=13787 dbname=postgres user=postgres
    sslmode=verify-full
   sslcert='/netbackup/postgresql/keys/server/tls.crt'
   sslkey='/netbackup/postgresql/keys/server/tls.key'
   sslrootcert=/netbackup/postgresql/keys/server/ca.crt"
   ```

   Following message is displayed:

```
Password for user postgres:
```

Request router logs

Request router log levels

Request router provides different log levels that can be changed after deployment. The following log levels are provided:

1	trace
2	debug
3	info
4	warning\warn
5	error
6	critical

By default, the log level is 1.

Steps to increase the request router log level:

1. Execute the following command to log in to the primary pod:

   ```
   kubectl exec -it -n <primary-server-namespace> <primary-pod-name>
   -- /bin/bash
   ```

2. To set the request router log level, run the following command once in the primary server shell prompt:

   ```
   vxlogcfg -a -p 51216 -o 527 -s DebugLevel=<log-level-value>
   ```

   For example, to set the log level to critical, run the following command:

   ```
   vxlogcfg -a -p 51216 -o 527 -s DebugLevel=6
   ```

View request router logs

Request router pod logs are collected by fluentbit collector pod.

The fluentbit collector writes request router logs to the log volume using the following directory convention:

```
/usr/openv/fluentbit/logs/<date>/<namespace>/<request-router-pod-name>/<container-name>/<log
file>
```

To view request router logs, execute the following command to get a shell to the running fluentbit collector container:

```
kubectl exec -it -n <primary-server-namespace>
<fluentbit-collector-pod-name> -- /bin/bash
```

Once in the fluentbit collector pod shell prompt, run the following command to view the list of logs:

```
ls
/usr/openv/fluentbit/logs/<date>/<namespace>/<request-router-pod-name>/<container-name>/
```

Issues with NBPEM/NBJM

- In the activity monitory, details would be visible as the job starts, which policyjob follower pod the job would be executed on.
- The `nbpemreq` command can be used to verify the assignments. If the command is run on the primary or the leader, it will display information on the WorkDistributor leader. The **–M** option can be specified with the pod name to get the information on the specific pod. If no **–M** option is specified when executing the command from the primary pod, then the default gets routed to the leader pod.
- Logging details: The `bp.conf` and `nblog.conf` files are shared between all pods. NBPEM and NBJM verbose levels can be set using Web UI. The verbose level applies to all pods instances. Viewing logs and extracting logs from `fluentbit` in detail is mentioned under **Common Operations**. Commands for increasing logs verbose are listed under **Usability and Supportability**.
- Debugging resources:
  - nbpem binary path: `/usr/openv/netbackup/bin/nbpem`
  - nbjm binary path: `/usr/openv/netbackup/bin/nbjm`
  - libnbsubscriber path: `/usr/openv/lib/libnbsubscriber.so` (In the primary main container)
  - libmqclient path: `/usr/openv/lib/libmqclient.so`
  - bprdproxy path: `/usr/openv/netbackup/bin/bprdproxy` (In the primary main container)

Issues with logging feature for Cloud Scale

Useful commands required during troubleshooting

- To get the list of nodes:

  ```
  $ kubectl get nodes
  ```

- To view the information (such as Taints applied), describe the node:

```
$ kubectl describe node <node name>
```

- To view which nodes a pod is assigned to:

  ```
  $ kubectl get pods -A -o wide
  ```

  This command displays the extra data including node.

- To obtain the information about the fluentbit DaemonSet's, run the describe command on fluentbit DaemonSet's:

  ```
  $ kubectl describe ds nb-fluentbit-daemonset -n netbackup
  ```

  This command displays how many DaemonSet's are scheduled.

DaemonSet's not being assigned

If the taints and tolerations are not configured properly, a lack of DaemonSet's being assigned is seen and the container and pod logs would not be considered. This issue is due to tolerations not setup in the `values.yaml` file or tolerations not added.

The values can be viewed using the following commands:

- To get DaemonSet's in NetBackup namespace:

  ```
  $ kubectl get ds -n <netbackup namespace>
  ```

- To view tolerations:

  ```
  $ kubectl edit ds -n <netbackup namespace> nb-fluentbit-daemonset
  ```

  The tolerations can be found in the vi menu that is opened. If no change is required then do not save any changes.

Permission issue

Following error message appears when fluentbit scans the location for logs which has permission issues:

```
[error] [input:tail:tail.0] read error, check permissions:
/mnt/nblogs/*/*/*.log
[ warn] [input:tail:tail.0] error scanning path: /mnt/nblogs/*/*/*.log
[error] [input:tail:tail.0] read error, check permissions:
/mnt/nblogs/*/*/*/*.log
[ warn] [input:tail:tail.0] error scanning path:
/mnt/nblogs/*/*/*/*.log
[error] [input:tail:tail.0] read error, check permissions:
/mnt/nblogs/*/*/*.log
[ warn] [input:tail:tail.0] error scanning path: /mnt/nblogs/*/*/*.log
```

The above error messages are displayed in the sidecar logs which can be found in the collector pod as they are picked up by the DaemonSet's and stored under the pod that the sidecar resides in. Some application logs associated with the sidecar may be missing from the collector if this error occurs.

Workaround:

Exec into the sidecar and determine which folder has permission issues.

Issue while adding or configuring the toleration into fluentbit configuration file on AKS or EKS cluster

If you add an incorrect labels into .yaml file, it will end up having no demonset running for that node and eventually logs would not be collected for pods that are present on that node.

Issue with Agentpool

If any of the Cloud Scale node is configured on the Agentpool (systempool) in case of Azure, then the demonset would not be able to collect the logs.

The flexsnap-listener pod is unable to communicate with RabbitMQ

After successfully deploying Cloud Scale, if the RabbitMQ server is deleted, it gets recreated with a new server IP.

However, the `flexsnap-nginx` server configuration does not get updated to reflect the new RabbitMQ server IP. Hence, the `flexsnap-listener` pod is unable to communicate with RabbitMQ through `flexsnap-nginx` server.

Workaround:

To resolve the issue, perform the following:

- After RabbitMQ is recreated, restart the `flexsnap-nginx` server to update its configuration.
- Restart `flexsnap-listener` server to get the listener pod in running state.

Job remains in queue for long time

- Job remains in queue for long time with 'Cloud scale media server is not available' reason as follows:

```
awaiting resource abc-stu. Waiting for resources.
        Reason: Cloud scale media server is not available., Media
server: media1-media-0,
        Robot Type(Number): NONE(N/A), Media ID: N/A, Drive Name:
N/A,
        Volume Pool: N/A, Storage Unit: default_stu_xyz.abc.com,
Drive Scan Host: N/A,
         Disk Pool: default_dp_stu_xyz.abc.com, Disk Volume:
PureDiskVolume
```

The above issue occurs due to one of the following reason while creating STU:

- While selecting media server, **Manually select** option is selected and specific elastic media server or primary server is selected explicitly.
- While selecting media server, **Allow NetBackup to automatically select** option is selected and primary server as only media server is listed in media server list.

Workaround:

To resolve the issue, perform the following:

- Edit the respective storage unit, if **Manually select** option is selected for media server. Change the option to **Allow NetBackup to automatically select**.
- If non default storage server is used and while creating stu, **Allow NetBackup to automatically select** option is selected and primary server is listed in media server list as the only media server, then edit the respective storage server and add external or elastic media server in the media server list and remove the primary server.

- Job remains in queue for long time with "Media server is currently not connected to master server" or "Disk media server is not active" due to the following reasons:
  - At least one elastic media server is 'Offline'.
  - Primary server is not present in Media server of default storage server when **minimumReplica** value is set to 0.
  - ```
    awaiting resource default_stu_abc.com. Waiting for resources.
    Reason: Media server is currently not connected to
    master server, Media server: media1-media-0,
    Robot Type(Number): NONE(N/A), Media ID: N/A, Drive
    Name: N/A,
    Volume Pool: NetBackup, Storage Unit:
    default_stu_abc.com, Drive Scan Host: N/A,
    Disk Pool: default_dp_nbux-abc.com, Disk Volume:
    PureDiskVolume
    ```
  - ```
    awaiting resource default_stu_abc.com. Waiting for resources.
    Reason: Disk media server is not active, Media server:
    media1-media-0,
    Robot Type(Number): NONE(N/A), Media ID: N/A, Drive
    Name: N/A,
    Volume Pool: NetBackup, Storage Unit:
    default_stu_abc.com, Drive Scan Host: N/A,
    ```

```
Disk Pool: default_dp_abc.com, Disk Volume:
PureDiskVolume
```

Workaround: Perform the following respective workaround depending on the reason of the issue:

Issue	Workaround

At least one elastic media server is 'Offline'	***Workaround 1***:
	Change the value of **minimumReplica** to a value greater than the current value of **minimumReplica** and wait for new media server pod to be in **ready** state.
	Then value of **minimumReplica** can be changed to original value of **minimumReplica**.
	Use the following command to update the value of **minimumReplica** in mediaServer section:
	`helm upgrade cloudscale VRTSk8s-netbackup-11.0.1-0118/helm/cloudscale-11.0.1-118.tgz -n nbux --reuse-values \`
	`--set environment.mediaServers[0].name=media1 --set environment.mediaServers[0].nodeSelector.labelKey=agentpool \`
	`--set environment.mediaServers[0].nodeSelector.labelValue=mediapool --set environment.mediaServers[0].replicas=2 \`
	`--set environment.mediaServers[0].minimumReplicas=2 --set environment.mediaServers[0].storage.data.capacity=60Gi \`
	`--set environment.mediaServers[0].storage.data.storageClassName=gp2-ebs-storage-class \`
	`--set environment.mediaServers[0].tag=11.0.1-0118 --set environment.mediaServers[0].storage.log.capacity=10Gi \`
	`--set environment.mediaServers[0].storage.log.storageClassName=gp2-ebs-storage-class`
	Save the changes.

Issue	Workaround

	Workaround 2:
	If media server pod is not running, change the 'Offline' media server state to 'Deactivated' state as follows:
	Run `PATCH /config/media-servers/{hostName}` API to change the 'machineState' of Offline media server to **CEMM_MACHINE_ADMINISTRATIVE_PAUSE_SET**.
	`Or`
	Execute the following command to exec in primary server pod:
	`kubectl exec -it -n <namespace> <primaryServer-pod-name> -- bash`
	Execute the following command for Offline media server:
	`nbemmcmd -updatehost -machinename <offline-media-server-hostname> -machinestateop set_admin_pause -machinetype media -masterserver <primary-server-hostname>`
	Note: For elasticity, scaled in media servers are marked as **Deactivated** by media server elasticity.

Issue	Workaround

Primary server is not present in Media server of default storage server when **minimumReplica** value is set to 0 and the following error appears in netbackup-operator-pod logs:
 	
`Error in registering additional media servers in storage server. Please add manually.`
	Run the following command to obtain the netbackup-operator-pod logs:
	
	`kubectl logs <netbackup-operator-pod-name> -c netbackup-operator -n <netbackup operator-namespace>`
	
	To resolve the issue, perform one of the following:
	■ Set the value of **minimumReplica** to a value greater than 0 and wait for atleast one media server pod to be in ready state.
	■ After media server pod goes into running state then the value of **minimumReplica** can be set to 0.
	Use the following command to update the value of **minimumReplica** in mediaServer section:
	
	`helm upgrade cloudscale VRTSk8s-netbackup-11.0.1-0118/helm/cloudscale-11.0.1-118.tgz -n nbux --reuse-values \`
	`--set environment.mediaServers[0].name=media1 --set environment.mediaServers[0].nodeSelector.labelKey=agentpool \`
	`--set environment.mediaServers[0].nodeSelector.labelValue=mediapool --set environment.mediaServers[0].replicas=2 \`
	`--set environment.mediaServers[0].minimumReplicas=2 --set environment.mediaServers[0].storage.data.capacity=60Gi \`
	`--set environment.mediaServers[0].storage.data.storageClassName=gp2-ebs-storage-class \`
	`--set environment.mediaServers[0].tag=11.0.1-0118 --set environment.mediaServers[0].storage.log.capacity=10Gi \`
	`--set environment.mediaServers[0].storage.log.storageClassName=gp2-ebs-storage-class`
	
	Save the changes.
	
	Note: For a value of **minimumReplica** greater than 0, the primary server is not present in Media servers of default storage server.

Extracting logs if the nbwsapp or log-viewer pods are down

In order to extract the logs out of Cloud Scale when the **nbwsapp** or **log-viewer** pods are down, copy the logs out of the fluentbit-collector pod. You can tar and compress the logs before extraction or extract them immediately.

1. (*Optional*) Tar up the files you want to extract. Select a folder and run the following command:

```
$ tar -cvf <name of tar> <name of folder>
```

For example, `$ tar -cvf 2024-02-03.tar 2024-02-03/`

2. Copy the files out of the container. Exit the container with a simple exit command and run the following command:

```
$ kubectl cp -n netbackup
<pod-name>:/usr/openv/fluentbit/logs/<folder or tar> <output
folder or tar>
```

3. (*Optional*) Extract the tar outside the container if necessary using the following command:

```
$ tar xvf <output tar>
```

If the fluentbit-collector pod is down, then logs can be extracted directly from the log locations on the application pods using the similar commands.

Helm installation failed with bundle error

Helm installation fails to continue with the following error message:

```
Unable to continue with install: Bundle "db-cert" in namespace ""
exists and cannot be imported into the
current release: invalid ownership metadata; label validation error:
 missing key "app.kubernetes.io/managed-by": must be set to "Helm";
annotation validation
error: missing key "meta.helm.sh/release-name": must be set to
<release-name>"; annotation validation error: missing key
"meta.helm.sh/release-namespace": must be set to <NetBackup namespace>
```

This issue occurs when `db-cert` bundle is not deleted from the `""` empty namespace after Cloud Scale deployment is uninstalled.

To resolve the issue, perform the following:

1 Use the following command to check if the `db-cert` bundle exists:

```
kubectl get bundle -n ""
```

2 Use the following command to delete the `db-cert` bundle:

```
kubectl delete bundle db-cert -n ""
```

Deployment fails with private container registry and Postgres fails to pull the images

When using third-party registry for container images, the following error message is displayed for some images:

```
Warning Failed 8m1s (x5 over 10m) kubelet Failed to pull image
"reg3rdpartyshan.azurecr.io/netbackup/postgresql-upgrade:16.9.1.0-0003":
 failed to pull and unpack image
"reg3rdpartyshan.azurecr.io/netbackup/postgresql-upgrade:16.9.1.0-0003":
 failed to resolve reference
"reg3rdpartyshan.azurecr.io/netbackup/postgresql-upgrade:16.9.1.0-0003":
 failed to authorize: failed to fetch anonymous token:
 unexpected status from GET request to 401:
https://reg3rdpartyshan.azurecr.io/oauth2/token?scope=repository%3Anetbackup%2Fpostgresql-upgrade%3Apull&service=reg3rdpartyshan.azurecr.io
Unauthorized
Warning Failed 8m1s (x5 over 10m) kubelet Error: ErrImagePull
```

Workaround:

This is a transient error. Delete the pod. After the pod is up and running, ensure that you do not see the same error again.

Troubleshoot issues with debug tool

Some tools within the NetBackup Debug Tool Container may fail to execute or return incomplete results when attached to certain pods or nodes. It is recommended to use `--profile=sysadmin` and `--share-processes=true` options to have the same privileges as the original container or node.

Perform the following:

1. When launching the debug container, ensure that it is configured with elevated privileges and appropriate namespace sharing.
2. Push the image to the registry: The debug tool image is not pushed to the container registry by the image-pushing script and must be pushed manually by the user.
   - Navigate to the `VRTSk8s-netbackup-<version>/images` directory.
   - Load the debug tool image:

     ```
     # docker load -i netbackup-debugtool-<version>.tar.gz
     ```
   - Tag the image for registry:

     ```
     # docker tag localhost/netbackup/debugtool:<version>
     <registry>/netbackup/debugtool:<version>
     ```

- Push the image to the registry:

  ```
  # docker push <registry>/netbackup/debugtool:<version>
  ```

3. Using the debug tool:

   - Use the following command to debug a pod:

     ```
     # kubectl debug pod/<pod name> -it --image=<debugtool image>
     --target=<conatiner name> -n <namespace> --profile=sysadmin
     --share-processes=true
     ```

     For example, `# kubectl debug pod/nbu-primary-0 -it --image=nbuk8sreg.azurecr.io/netbackup/debugtool:11.2 --target=nbu-primary -n netbackup --profile=sysadmin --share-processes=true`

     Note: When you exit the debug tool container, the container terminates. Running the command again creates and attaches a new debug tool container.

   - Use the following command to debug a node:

     ```
     # kubectl debug node/<node name> -it --image=<debugtool image>
     --profile=sysadmin --share-processes=true
     ```

     For example, `# kubectl debug node/aks-nbuxpool-26991557-vmss00001t -it --image=nbuk8sreg.azurecr.io/netbackup/debugtool:11.2 --profile=sysadmin --share-processes=true`

Cloud Scale upgrade shows environment failed or partially ready

During a Cloud Scale upgrade, the environment may report a partial READY state (for example, `3/4 READY`) or appear as Failed.

The status can also fluctuate between phases such as:

- Upgrading MSDPScaleouts
- Upgrading MediaServers

At the same time:

- PrimaryServer and MediaServer components may report Success
- MSDPScaleout may remain not fully READY

Additional indicators include:

- Operator logs showing:

- Not all MSDP resources are ready
- Failed to get media servers registered to storage server
- HTTP timeout to storage-server API

- Primary logs may show NBSL / RDSM errors for the same period, then succeed later.

The Cloud Scale upgrade workflow depends on MSDPScaleout availability, including storage services and RDSM. While MSDP engines are starting, restarting, or temporarily unstable, upgrade reconcilers retry dependent operations. During this phase, transient errors can be reported until all required components become fully healthy.

Workaround:

Perform the following:

1. Check Environment, MSDPScaleout, PrimaryServer, and MediaServer resources together rather than relying on the Environment status alone.
2. Allow time for the upgrade to progress and recheck the status; in many cases, the condition resolves automatically once MSDP becomes stable.
3. If the issue persists: verify that all MSDP pods and engines are running and stable.
   - Verify that all MSDP pods and engines are running and stable.
   - Correlate NetBackup operator logs with primary server logs for storage or RDSM-related errors.
4. Escalate the issue if:
   - MSDPScaleout remains not Ready.
   - MSDP pods are crash-looping.
   - PrimaryServer or MediaServer custom resources remain Failed and API calls never succeed.

NetBackup Snapshot Manager auto-registration fails after successful Cloud Scale deployment

NetBackup Snapshot Manager deploys successfully in a Cloud Scale environment, but does not auto-register with the NetBackup primary server.

On a fresh NetBackup Snapshot Managerdeployment, all Kubernetes resources may appear healthy. The environment and cpServer custom resources show a **Success** state, and all flexsnap pods are in **Running** state. However, the Snapshot

Manager does not appear in the NetBackup primary server's list of registered Snapshot Managers.

This issue occurs when one or more flexsnap container images take longer than expected to pull during the first deployment. If the image pull exceeds the internal 10-minute health-check window, the operator marks the deployment as failed before reaching the Snapshot Manager registration step. As a result, registration is skipped even though the deployment completes successfully.

Run the following commands to confirm the issue. If one of the following command displays the failure indicators, the deployment is affected.

- Check deployment configmap status:

  ```
  kubectl get cm flexsnap-deployment-conf-<cpserver-name> -n
  <env-namespace> -o yaml | grep -i status
  ```

  Failure indicator:

  ```
  STATUS: Failed
  ```

- Check cpServer registration annotations:

  ```
  kubectl get cpserver <cpserver-name> -n <env-namespace> -o
  jsonpath='{.metadata.annotations}{"\n"}'
  ```

  Failure indicator:

  ```
  <empty output>
  ```

Workaround:

Remove and recreate the cpServer configuration by updating the environment custom resource. This forces the operator to re-run the full install workflow. Because the container images are already cached from the first attempt, the second run completes within the health-check window and successfully performs auto-registration.

1. Back up the environment configuration:

   - Save a copy of the environment custom resource using the following command:

     ```
     kubectl get environment <environment-name> -n <env-namespace>
     -o yaml > environment-backup.yaml
     ```

   - Verify that the flexsnap operator is running using the following command:

     ```
     kubectl get pods -n netbackup-operator-system | grep
     flexsnap-operator
     ```

2. Remove the cpServer specification:

   Edit the environment custom resource using the following command:

```
kubectl edit environment <environment-name> -n <env-namespace>
```

Delete the entire `spec.cpserver` section. Save and exit.

3. Wait for cpServer cleanup to complete.

   The operator will run the uninstall path on the cpServer. This scales down and removes `flexsnap-*` resources.

4. Re-add the cpServer specification:

   Edit the environment custom resource again using the following command:

   ```
   kubectl edit environment <environment-name> -n <env-namespace>
   ```

   Reinsert the `spec.cpserver` section using the values from `environment-backup.yaml` file. Save and exit.

5. Wait for cpServer reinstall to complete and verify that cpServer state is **Success**:

   ```
   kubectl get cpserver -n <env-namespace>
   ```

Verification:

After redeployment completes, confirm that Snapshot Manager registration is successful.

1. Verify deployment configmap status:

   ```
   kubectl get cm flexsnap-deployment-conf-<cpserver-name> -n
   <env-namespace> -o yaml | grep -i status
   ```

   Expected output:

   ```
   STATUS: Success
   ```

2. Verify cpServer registration annotations:

   ```
   kubectl get cpserver <cpserver-name> -n <env-namespace> -o
   jsonpath='{.metadata.annotations}{"\n"}'
   ```

   Expected output:

   ```
   netbackup.veritas.com/registration-status: Success
   ```

No usage report is available to download from the Web UI

Usage reports are not available for download from the NetBackup Web UI. Reports may be generated at a location that is not accessible to the Web UI.

This issue can occur if the **REPORTDIR** parameter is not set in the `nbdeployutilconfig.txt` file or is configured to a non-shared location.

To resolve this issue:

1. Edit the following file:

   `/usr/openv/var/global/nbdeployutilconfig.txt`

2. Under the **[NBDEPLOYUTIL_INCREMENTAL]** section, set the **REPORTDIR** parameter to a valid shared location.

   For example, `REPORTDIR=/mnt/nbdata/usr/openv/var/global/incremental`

3. Save the file.

4. Wait for the next scheduled reporting cycle to generate the reports in the configured **REPORTDIR** location.

Media server remains in running state after Helm upgrade

After triggering a Helm upgrade for the media server, the media server custom resource (CR) does not transition to a successful state and continues to remain in the *Running* state.

This issue can occur when the `environment.mediaServers[0].name` value specified during the Helm upgrade does not match the existing media server name. Due to this mismatch, a new media server is created with a different name, but it is not properly registered with the storage server.

In this scenario, the NetBackup operator logs display an error similar to the following:

```
Error in registering additional media servers in storage server.
Please add manually{"controller": "mediaserver", "controllerGroup":
 "netbackup.veritas.com", "controllerKind":
"MediaServer","MediaServer": {"name":"new1","namespace":"netbackup"},
 "namespace": "netbackup", "name": "new1","reconcileID":
"6f45dee4-ff55-43b1-bb49-0482d5abb638","storage server id":
"PureDisk:<storage server name>","error": "failed to update additional
 media server details for storage server: Failed to update storage
server."}
```

Workaround:

To resolve this issue, perform the following:

- Log in to the primary server pod:

  `kubectl exec -it -n <namespace> <primary-server-pod-name> -- bash`

- Manually register the media server with the storage server using the `tpconfig` command:

  ```
  tpconfig -add -storage_server <server_name> -stype <server_type>
  -sts_user_id <user_ID> -password <password>
  ```

- Wait for the new media server to transition to the *Success* state.
- Re-run the Helm upgrade command using the correct media server name to restore the original configuration.

Troubleshooting AKS-specific issues

This section lists the additional troubleshooting issues and their respective workaround that are specific to Azure Kubernetes Services cluster in the Azure Cloud.

Data migration unsuccessful even after changing the storage class through the storage yaml file

To resolve the data migration unsuccessful issue

1. Check if the previous/older migration PVC is still present.
2. Check if an existing migration job is in pending/running state.
3. Migration will not be triggered if storage class name is not changed for volumes (catalog) of primary server.
4. Check if the updated storage yaml file is set to use any other storage class other than `Azure premium files` for catalog.
5. Check if the protocol version for catalog storage class is set to **NFS** to allow successful migration.

6 Check if the status of migration reported by the migration pod in the logs is other than *MigrationStatus: Passed*.

7 Ensure that the User's Azure subscription has **Classic Network Contributor** role which is also a pre-requisite for migration/upgrade.

Note: If reconciler is called while migration PVC exists the invocation will be failed, customers must wait for the completion of a migration job if an existing migration job is running and they can also monitor the migration job pods to check if there are any issues with the migration job. In order to resolve any problems encountered during existing migration job pod they may choose to delete the migration job pod manually. If the migration job pod does not exist, then customer may delete the migration PVC.

Host validation failed on the target host

The following error message is displayed when host mapping conflicts with NetBackup:

```
Error in log: ..exited with status 7659: Connection cannot be
established because the host validation failed on the target host
```

In Kubernetes deployment, communication to pod happens through multiple layers that is, load balancer, nodes and pod. In certain setups during communication host may get associated with certain IP and that gets changed in course of time. That IP may get associated with some different pod and can cause conflict. The host mapping entries is in the form of `:ffff:<ip address>`.

To resolve this conflict issue

1 To verify the conflict entries, see **Mappings for Approval tab** section of *NetBackup™ Security and Encryption Guide*.

2 Remove the entries that are not valid.

For more information, see **Removing host ID to host name mappings** section of *NetBackup™ Security and Encryption Guide*

Primary pod goes in non-ready state

When jobs are running and if failover or upgrade of NetBackup primary server is triggered then primary pod may go in non-ready state. To confirm the issue,

- Exec into primary pod by running the following command:

```
kubectl exec -it <primary_pod_name> -n <namespace> -- bash
```

- Open the logs using the `cat /mnt/nblogs/setup-server.log` command and verify the last lines:

```
Softlink /mnt/nbdata/usr/openv/netbackup/db/rb.db does not exist or broken.
Error: Issue detected while validating soft links, restarting the pod may fix this.
NetBackup health check disabled.
```

To resolve this issue, delete the corrupted database and correct symlink as follows:

1. Exec into primary pod by running the following command:

   `kubectl exec -it <primary_pod_name> -n <namespace> – bash`

2. In primary pod run the following commands in order:

```
# /opt/veritas/vxapp-manage/nb-health disable
# bp.kill_all
# mv -f /mnt/nbdata/usr/openv/netbackup/db/rb.db /mnt/nbdb/usr/openv/netbackup/db/rb.db
# ln -sf /mnt/nbdb/usr/openv/netbackup/db/rb.db /mnt/nbdata/usr/openv/netbackup/db/rb.db
# chown -h nbsvcusr:nbsvcusr /mnt/nbdata/usr/openv/netbackup/db/rb.db
# bp.start_all
# /opt/veritas/vxapp-manage/nb-health enable
```

Troubleshooting EKS-specific issues

This section lists the additional troubleshooting issues and their respective workaround that are specific to Amazon Elastic Kubernetes cluster in the Amazon Web Services Cloud.

Resolving the primary server connection issue

NetBackup operator pod logs displays the following error:

```
Failed to connect the Primary server. "error":

"Get \"https://abc.xyz.com:*/netbackup/security/cacert\":

dial tcp: i/o timeout
```

The above error appears due to the following reasons:

- Operator, primary server and media server pod must have been started in different availability zones.
- Load balancer IP address and node on which primary server pod have started are in different availability zones.

- Load balancer created for NetBackup load balancer service is in failed state.
- FQDN to IP address given in networkLoadBalancer section in CR spec is not DNS resolvable.

To resolve the primary server connection issue, perform the following

1. Delete the primary server and media server CR instance using the following command:

   ```
   helm uninstall cloudscale -n <env_namespace>
   ```

   Caution: Be cautious while performing this step.

2. Fix the issue and provide appropriate details in CR specs in `cloudscale-values.yaml` file.
3. Redeploy the NetBackup by reapplying the `cloudscale-values.yaml` file using the following command:

   ```
   helm upgrade --install cloudscale cloudscale-<version>.tgz -f
   cloudscale-values.yaml --namespace <env_namespace>
   ```

NetBackup Snapshot Manager deployment on EKS fails

NetBackup Snapshot Manager deployment on EKS fails as cpServer fails to register Snapshot Manager to NetBackup.

This issue can be resolved by increasing the limit of inbound/outbound rules per Security Group. For more information, refer to the following link:

How do I increase my security group rule quota in Amazon VPC?

Wrong EFS ID is provided in `cloudscale-values.yaml` file

- Following error message is displayed when wrong EFS ID is provided in `cloudscale-values.yaml` file:

```
"samples/cloudscale-values.yaml": admission webhook
"environment2-validating-webhook.netbackup.veritas.com" denied the
request: Environment change rejected by validating webhook: EFS ID
provided for Catalog storage is not matching with EFS ID of already
created persistent volume for Primary servers Catalog volume. Old
EFS ID fs-0bf084568203f1c27 : New EFS ID fs-0bf084568203f1c29
```

Above error can appear due to the following reasons:

- During upgrade, if EFS ID provided is different from the EFS ID used in the previous version deployment.
- During fresh deployment, if user manually creates PV and PVC with EFS ID and provides different EFS ID in `cloudscale-values.yaml` file.

To resolve this issue, perform the following:

1. Identify the correct EFS ID used for PV and PVC.
   - Previously used EFS ID can be retrieved from PV and PVC by using the following steps:
     ```
     kubectl get pvc -n <namespace>
     ```
   - From the output, copy the name of catalog PVC which is of the following format:
     ```
     catalog-<resource name prefix>-primary-0
     ```
   - Describe catalog PVC using the following command:
     ```
     kubectl describe pvc <pvc name> -n <namespace>
     ```
     Note down the value of **Volume** field from the output.
   - Obtain the PV name using the following command:
     ```
     kubectl get pv -n <namespace>
     ```
   - Describe PV using the following command:
     ```
     kubectl describe pv <value of Volume obtained from above step>
     ```
     Note down the value of **VolumeHandle** field from the output. This is the EFS ID used previously.
2. Provide correct EFS ID in the `cloudscale-values.yaml` file and apply the `cloudscale-values.yaml` using the following command:
   ```
   helm upgrade --install cloudscale cloudscale-<version>.tgz -f
   cloudscale-values.yaml --namespace <env_namespace>
   ```

Primary pod is in ContainerCreating state

Primary pod is in ContainerCreating state for a long period due to the following reasons:

- Wrong EFS ID
- Format of the EFS ID is incorrect

To resolve this issue, perform the following:

1. Describe primary pod using the following command:

   `kubectl describe <name of primary server pod> -n <namespace>`

2. Depending on the following appropriate scenario, fix the error from the output under the Event section:

   - If the event log has an error related to incorrect EFS ID or incorrect format, then update the `cloudscale-values.yaml` file with the correct EFS ID and perform the below steps.
     Or
   - If the event log has an error other than the error related to incorrect EFS ID, then analyze and fix the error and perform the below steps.

3. After fixing the error, clean the environment using the following command:

   `helm uninstall operators -n <netbackupoperator-system>`

4. Delete PV and PVC created for primary server only by using the following command:

   `kubectl delete environment <environmentCR-name> -n <namespace>`

   Describe the PVC for primary server which has the following format and obtain the corresponding PV name:

   ```
   catalog-<resource name prefix of primary>-primary-0
   data-<resource name prefix of primary>-primary-0
   logs-<resource name prefix of primary>-primary-0
   ```

   Delete PVC and PV names using the following commands: For PVC: kubectl delete pvc <pvc name> -n <namespace> For PV: kubectl delete pv <pv name>

   - PVC: `kubectl delete pvc <pvc name> -n <namespace>`
   - PV: `kubectl delete pv <pv name>`

5. Deploy NetBackup operator again and then deploy Cloud Scale.

Webhook displays an error for PV not found

Following error message is displayed when the user creates PVC with **claimRef** and fails to create PV during deployment:

```
error when creating "samples/environment.yaml": admission webhook
"environment2-validating-webhook.netbackup.veritas.com" denied the
request: Environment change rejected by validating webhook: PVC exist
for Catalog Volume. PersistentVolume pv-catalog-nbuxsystest-primary-0
```

```
does not exist for primary server's Catalog volume : PersistentVolume
"pv-catalog-nbuxsystest-primary-0" not found.
```

This issue can be resolved by creating PV and apply `cloudscale-values.yaml` file again.

Cluster Autoscaler initialization issue

Cluster Autoscaler on EKS fails to initialize due to missing RBAC permissions for the `VolumeAttachment` resource in the `storage.k8s.io` API group.

This issue occurs because Kubernetes version 1.33 requires additional permissions that are not included in the default cluster autoscaler **ClusterRole**. During initialization, the cluster autoscaler status remains **Initializing**, and pod logs display the following error:

```
failed to list *v1.VolumeAttachment: volumeattachments.storage.k8s.io
 is forbidden: User
"system:serviceaccount:kube-system:cluster-autoscaler" cannot list
resource "volumeattachments"
in API group "storage.k8s.io" at the cluster scope
```

To resolve the Cluster Autoscaler initialization issue on Kubernetes 1.33 (EKS):

1. Identify the **ClusterRole** used by the cluster autoscaler.

   Run the following commands to locate the associated **ClusterRoleBinding** and view its details:

   ```
   kubectl get clusterrolebinding | grep cluster-autoscaler
   ```

   ```
   kubectl describe clusterrolebinding <binding-name>
   ```

2. Add the `volumeattachments` resource permission to the **ClusterRole**. Find the `storage.k8s.io` rule in the **ClusterRole** and patch it to include `volumeattachments`:

   ```
   kubectl get clusterrole <cluster-autoscaler-role> -o yaml | grep
   -n "storage.k8s.io" -A 10
   ```

   ```
   kubectl patch clusterrole <cluster-autoscaler-role> --type='json'
   -p='[
     {
       "op": "add",
       "path": "/rules/0/resources/-",
       "value": "volumeattachments"
     }
   ]'
   ```

3. Restart the Cluster Autoscaler deployment to apply the updated permissions.

   ```
   kubectl rollout restart deployment/cluster-autoscaler -n
   kube-system
   ```

4. Verify that the Cluster Autoscaler status is Running.

   ```
   kubectl describe cm cluster-autoscaler-status -n kube-system
   ```

   ```
   autoscalerStatus: Running
   ```

5 Run the following command to confirm if the updated permissions include `volumeattachments`:

```
kubectl get clusterrole <cluster-autoscaler-role> -o yaml | grep
-A 15 "storage.k8s.io"
```

```
- apiGroups:
  - storage.k8s.io
  resources:
  - storageclasses
  - csinodes
  - csidrivers
  - csistoragecapacities
  - volumeattachments
  verbs:
  - watch
  - list
  - get
```

6 Run the following command to check the Cluster Autoscaler logs to confirm normal operation and ensure that no further permission errors are reported:

```
kubectl -n kube-system logs -l app=cluster-autoscaler -f
```

Catalog backup job fails with an error (Status 9202)

After upgrading to NetBackup Cloud Scale 11.1 or later, the catalog backup job fails with the following error message:

```
Error nbpem (pid=120) catalog backup exited with status 9202
(The service user account does not have the write permissions on the
 specified path.)
```

With version 11.1, the NetBackup Database Manager (BPDBM) has been moved from the primary pod to its own scaled-out service pod. As a result, certain volumes that were previously accessible to the primary pod are no longer available to BPDBM, which means that it cannot write to the following paths anymore:

- `/mnt/nblogs`
- `/mnt/nbdb`

Workaround:

Update the Disaster Recovery package destination path in your Catalog Backup policy to use a valid volume that BPDBM can access. Change the path to a location on the following volume:

```
/mnt/nbdata
```

Troubleshooting issue for bootstrapper pod

When inside the NetBackup fluentbit collector, you may see multiple bootstrapper log folders. This is expected behavior because Kubernetes jobs add unique IDs to the pod's name every time the pod is rescheduled or redeployed.

To ensure that you are looking at the most recent bootstrapper logs, check the date when the bootstrapper log folder was created by running the following command:

```
ls -lt
```

To initialize the bootstrapper pod to run again after addressing the failure, execute the following steps:

- Execute the following command:

  ```
  $ kubectl get jobs -n <netbackup namespace> NAME
  ```

- Output is:

  ```
  COMPLETIONS                                          DURATION
           AGE
  job.batch/<netbackup namespace>-bootstrapper   0/1
           61m              61m
  ```

- Extract the name provided by the kubectl command in step 1 and run:

  ```
  $ kubectl delete job.batch/<netbackup namespace>-bootstrapper -n
  <netbackup namespace>
  ```

- After the command completes successfully, the bootstrapper pod getting executed in the environment.

  ```
  $ kubectl get pods -n <netbackup namespace>
  bash-4.4$ kubectl get pods -n netbackup
  NAME                                                        READY
  STATUS     RESTARTS   AGE
  <netbackup namespace>-bootstrapper-qf5xt                    0/2
  Running    0          4m47s
  <netbackup namespace>-nbatd-0                               1/4
  Running    0          4m37s
  <netbackup namespace>-nbmqbroker-0                          0/2
  Init:0/2   0          4m31s
  <netbackup namespace>-bpdbm-0                               0/4
  Init:0/2   0          4m26s
  <netbackup namespace>-nbwsapp-0                             0/4
  ```

```
Init:0/2   0          4m26s
<netbackup namespace>-policyjob-0                          0/5
Init:0/1   0          4m26s
<netbackup namespace>-policyjobmgr-0                       0/5
Init:0/1   0          4m26s
<netbackup namespace>-primary-0                            0/2
Init:0/1   0          4m26s
<netbackup namespace>-requestrouter-6db64979c8-r49jm   0/1
Init:0/1   0          2m26s
nb-fluentbit-collector-587d75547-8n4qj                     2/2
Running    0          6m22s
nb-fluentbit-daemonset-j59k8                               1/1
Running    0          3m47s
nb-fluentbit-daemonset-plxvk                               1/1
Running    0          6m22s
nb-postgresql-0                                            1/1
Running    0          6m17s
```

Troubleshooting issues for kubectl plugin

Primary key not recreated after manual deletion from secrets/sp-keys

If the primary key is manually deleted by editing the `secrets/sp-keys` in the environment, the key is not automatically recreated by the system. This results in missing or invalid service principal keys, which can affect environment operations.

Workaround:

Perform the following steps to manually recreate the deleted key:

1. Pause the environment reconciler using helm command to set the following parameter:

   `paused: true`

2. Delete the existing secret by running the following command:

   `- kubectl delete secrets/sp-keys -n netbackup`

3. Delete all the API keys using the following API's call with a valid JWT token:

   - Trigger `GET netbackup/security/service-principal-configs` API.
   - Find all the service principal configurations where **"servicePrincipalType"="netbackup-operator"** and note the value of **servicePrincipalId** for all these service principal configurations.

- Trigger `DELETE security/service-principal-configs/{id}` API for each **servicePrincipalId** captured above.

4. Restart the operator pod by deleting the operator pod:

   ```
   - kubectl delete pod <operator-pod-name> -n
   netbackup-operator-system
   ```

5. Un-pause the environment reconciler using helm command to set the following parameter:

   ```
   paused: false
   ```

Cloud Scale installation interrupted midway

During the Cloud Scale installation, if the process breaks or stops midway, the deployment does not complete successfully and must be resumed from the point of failure.

Workaround:

To resume the installation process:

1. Run the following command:

   ```
   kubectl-cloudscale install
   ```

2. When prompted to resume the installation, type **yes** to continue from where it stopped.

The plugin will automatically read the previously saved inputs and resume the installation from the point of interruption.

Unable to re-enter inputs during Cloud Scale upgrade

If a user enters an incorrect Cloud Scale folder path or provides any wrong input during the upgrade process, there is no option to correct it. Re-triggering the plugin simply skips to the next question without allowing the user to re-enter inputs.

Workaround:

To reset and re-enter all upgrade inputs:

1. Delete the following file:

   ```
   rm /home/<user-name>/.cloudscale/upgrade.csv
   ```

2. Re-run the upgrade command:

   ```
   kubectl-cloudscale upgrade
   ```

   The plugin will prompt for all required user inputs again.

   Following is an example of the logs:

```
*******************************Checking for
cert-manager********************************
{Component: Installation of helm chart, ComponentName:
jetstack/cert-manager}
INFO: 2025/09/26 12:38:32 logger.go:132: Checking if Cert Manager
 is installed or not {Component: Get pod status, ComponentName:
Dependency of Cloudscale component}
INFO: 2025/09/26 12:38:32 logger.go:132: Waiting 10 seconds before
 retrying... {Component: Get pod status, ComponentName: Dependency
 of Cloudscale component}
INFO: 2025/09/26 12:38:50 logger.go:132: Cloudscale Upgrade

Before you proceed with upgrade, please ensure the following
prerequisites are in place:

    1. Infrastructure readiness:
       - The Kubernetes cluster is up and running
       - Cloudscale environment is up and running

    2. Required container images:
       - All Cloudscale-related images of the version you would
like to upgrade to
         are pushed to your container registry

    3. Helm setup:
       - Helm is installed and configured
       - The "jetstack" repository is added:
         helm repo add jetstack https://charts.jetstack.io
       - The cert-manager and trust-manager charts are installed
 via helm

Also review the 'Prerequisites for Cloud Scale Technology upgrade'
 section in the
'NetBackup™ Deployment Guide for Kubernetes Clusters' document
for additional required steps.

Once everything is ready, you can safely continue with the
upgrade.
{Component: CloudScale, ComponentName: Upgrade of CloudScale}
INFO: 2025/09/26 12:38:50 logger.go:132: Would you like to
continue? (y/n):  {Component: CloudScale, ComponentName: Upgrade
 of CloudScale}
INFO: 2025/09/26 12:39:00 logger.go:132: Helm Version:
```

```
version.BuildInfo{Version:"v3.18.6",
GitCommit:"b76a950f6835474e0906b96c9ec68a2eff3a6430",
GitTreeState:"clean", GoVersion:"go1.24.6"}
{Component: Precheck Config, ComponentName: Installation of
CloudScale}
INFO: 2025/09/26 12:39:00 logger.go:132: kubectl Version: Client
 Version: v1.34.1
Kustomize Version: v5.7.1
Server Version: v1.32.6
{Component: Precheck Config, ComponentName: Installation of
CloudScale}
INFO: 2025/09/26 12:39:00 logger.go:132:
**************************************************
{Component: CloudScale Upgrade, ComponentName: Input
Configuration}
INFO: 2025/09/26 12:39:00 logger.go:129:
Checking if the input file already exists.

INFO: 2025/09/26 12:39:00 logger.go:132: Data is being read from
 an input file that is present.
{Component: CloudScale, ComponentName: Input Configuration}
INFO: 2025/09/26 12:39:00 logger.go:132: The following values
were loaded from the file:
{Component: CloudScale Upgrade, ComponentName: Input
Configuration}
```

Plugin upgrade interrupted or skipped validation checks

- The plugin was cancelled while checking for Cert Manager installation. When the upgrade was triggered again, it skipped the Cert Manager validation step and proceeded to the next question instead of restarting the validation process.
  Workaround:
  Delete the following file and re-run the upgrade:
  ```
  rm /home/<user-name>/.cloudscale/upgrade.csv
  ```
- The plugin was cancelled before completing the operator upgrade. When the upgrade was triggered again, it failed due to the Helm release being stuck in a pending-upgrade state with the following error:
  ```
  Operator Namespace : netbackup-operator-system
  Upgrade of operators started...
  Helm upgrade of operators failed with error: another operation
  (install/upgrade/rollback) is in progress
  Operators chart failed to upgrade.
  ```

```
Error while upgrading operators chart : another operation
(install/upgrade/rollback) is in progress
Upgrade failed:
```

Workaround:
To resolve this issue:

- Check for pending Helm releases:
  `helm ls -A --pending`
- If the operator's release is in a pending-upgrade state, list previous revisions:
  `helm history operators --namespace <operator_namespace>`
- Identify a revision with the status **deployed** or **superseded** and roll back to that version:
  ```
  helm rollback operators <REVISION> --namespace
  <operator_namespace>
  ```
  This clears the pending-upgrade lock caused by the interrupted upgrade.
- Once the rollback completes, rerun the plugin upgrade command:
  `kubectl-cloudscale upgrade`
- When prompted to resume the installation, type **yes** to continue from where it stopped.

Plugin crash after operator upgrade

The plugin crashed while annotating the environment resources immediately after the operator upgrade.

Workaround:

1. Trigger the upgrade process again using the `kubectl-cloudscale` plugin:

   ```
   kubectl-cloudscale upgrade
   ```

2. When prompted to resume the installation, type **yes** to continue the upgrade from where it stopped.

Plugin crash before Helm triggers Cloud Scale upgrade

If the plugin crashes at any point before Helm triggers the Cloud Scale upgrade command, the upgrade process is interrupted and cannot complete successfully.

Workaround:

1. Trigger the upgrade process using the `kubectl-cloudscale` plugin:

   ```
   kubectl-cloudscale upgrade
   ```

2. When prompted to resume the installation, type **yes** to continue the upgrade from where it stopped.

Media server pod restarting due to read-only mount

The media server pod is continuously restarting. Upon inspection, it is observed that the mount point `/mnt/nblogs` inside the media pod is in a read-only state, which is likely causing the restarts.

Perform the following steps to verify if the mount point is read-only

1. Exec the following command into the media pod:

   ```
   kubectl exec -it <media-pod-name> -- bash
   ```

2. Run the following command to check the mount point permissions:

   ```
   cd /mnt/nblogs/fluentbit
   ```

   ```
   touch test
   ```

   Expected output (in case of an issue):

   ```
   touch: cannot touch 'test': Read-only file system
   ```

   The above error message confirms that the `/mnt/nblogs` mount is mounted as read-only.

Workaround:

Run the following command to manually restart the media pod by deleting the affected media pod:

```
kubectl delete pod <media-pod-name>
```

Image validation fails after providing image tag input

Image validation can fail due to one or more of the following reasons:

- Invalid container registry name.
  An incorrect or non-existent registry name was provided.
  Example:

  ```
  Image validation failed, failed to configure transport: error
  pinging v2 registry:
  Get "https://wrong.azurecr.io/v2/": dial tcp: lookup
  wrong.azurecr.io on 168.63.129.16:53: no such host
  ```

- Container registry not logged in on the host VM.
  The container registry is not logged in using the same user account that runs the plugin on the host VM.
  Example:

```
Image validation failed,
Get
"https://CloudscaleACRshantaram.azurecr.io/v2/netbackup/dbm/manifests/11.2.0.2-0013":

unauthorized: authentication required
```

- Incorrect or non-existent image tags.
  The specified image tags do not exist or have not been pushed to the container registry.
  Example:

```
 Image validation failed, no such manifest:
nbuk8sreg.azurecr.io/netbackup/dbm:wrong
```

Workaround: Ensure that all inputs are correct and that the container registry is logged in properly.

To skip image validation for tags, run the following command:

```
kubectl create configmap cs-image-validation-config -n
<netbackup-namespace> --from-literal=SKIP_IMAGE_VALIDATION=true
```

Appendix A

CR template

This appendix includes the following topics:

- Secret
- MSDP Scaleout CR

Secret

Note: This content applies only to standalone MSDP scale-out installations. When MSDP scale-out is deployed as part of a Cloud Scale environment, these steps are not required.

The Secret is the Kubernetes security component that stores the MSDP credentials that are required by the CR YAML.

```
# The secret is used to store the MSDP credential, which is required
by the CR YAML as follows.
# This part should be created separately and not be part of CR Template.
# The secret should have a "username" and a "password" key-pairs with the
corresponding username and password values.
# Please follow MSDP guide for the rules of the credential.
#   https://support.cohesity.com/s/article/article-100048511
#   The pattern is "^[\\w!$+\\-,.:;=?@[\\]`{}\\|~]{1,62}$"
# We can create the secret directly via kubectl command:
#   kubectl create secret generic sample-secret --namespace
sample-namespace \
#   --from-literal=username=<username> --from-literal=password=<password>
# Alternatively, we can create the secret with a YAML file in the
following format.
apiVersion: v1
```

```
kind: Secret
metadata:
  name: sample-secret
  # The namespace needs to be present.
  namespace: sample-namespace
stringData:
  # Please follow MSDP guide for the credential characters and length.
  #   https://support.cohesity.com/s/article/article-100048511
  #   The pattern is "^[\\w!$+\\-,.:;=?[\\]`{}\\|~]{1,62}$"
  username: xxxx
  password: xxxxxx
```

MSDP Scaleout CR

- The CR name must be fewer than 40 characters.
- The MSDP credentials stored in the Secret must match MSDP credential rules.
  See Deduplication Engine credentials for NetBackup
- MSDP CR cannot be deployed in the namespace of MSDP operator. It must be in a separate namespace.
- You cannot reorder the IP/FQDN list. You can update the list by appending the information.
- You cannot change the storage class name.

  The storage class must be backed with:
  - AKS: Azure disk CSI storage driver "disk.csi.azure.com"
  - EKS: Amazon EBS CSI driver "ebs.csi.aws.com"
- You cannot change the data volume list other than for storage expansion. It is append-only and storage expansion only. Up to 16 data volumes are supported.
- Like the data volumes, the catalog volume can be changed for storage expansion only.
- You cannot change or expand the size of the log volume by changing the MSDP CR.
- You cannot enable NBCA after the configuration.
- Once KMS and the OST registration parameters set, you cannot change them.
- You cannot change the core pattern.

See "MSDP Scaleout CR template for EKS" on page 404.

See "MSDP Scaleout CR template for AKS" on page 397.

MSDP Scaleout CR template for AKS

```
# The MSDPScaleout CR YAML
apiVersion: msdp.veritas.com/v1
kind: MSDPScaleout
metadata:
  # The CR name should not be longer than 40 characters.
  name: sample-app
  # The namespace needs to be present for the CR to be created in.
  # It's not allowed to deploy the CR in the same namespace with MSDP
operator.
  namespace: sample-namespace
spec:
  # Your ACR URL where the docker images can be pulled from by the
AKS cluster on demand
  # The allowed length is in range 1-255
  # It's optional for BYO. The code does not check the presence or
validation.
  # User needs to specify it correctly if it's needed.
  containerRegistry: sample.azurecr.io
  #
  # The MSDP version string. It's the tag of the MSDP docker images.
  # The allowed length is in range 1-64
  version: "sample-version-string"
  #
  # Size defines the number of Engine instances in MSDP Scaleout.
  # The allowed size is between 1-16
  size: 4
  #
  # The IP and FQDN pairs are used by the Engine Pods to expose the MSDP
services.
  # The IP and FQDN in one pair should match each other correctly.
  # They must be pre-allocated.
  # The item number should match the number of Engine instances.
  # They're not allowed to be changed or re-ordered. New items can be
appended for scaling out.
  # The first FQDN is used to configure the storage server in NetBackup,
automatically if autoRegisterOST is enabled,
  # or manually by the user if not.
  serviceIPFQDNs:
    # The pattern is IPv4 or IPv6 format
```

```
  - ipAddr: "sample-ip1"
    # The pattern is FQDN format. `^[a-z][a-z0-9-.]{1,251}[a-z0-9]$`
    fqdn: "sample-fqdn1"
  - ipAddr: "sample-ip2"
    fqdn: "sample-fqdn2"
  - ipAddr: "sample-ip3"
    fqdn: "sample-fqdn3"
  - ipAddr: "sample-ip4"
    fqdn: "sample-fqdn4"
  #
 # # s3ServiceIPFQDN is the IP and FQDN pair to expose the S3 service from the MSDP
instance.
  # # The IP and FQDN in one pair should match each other correctly.
  # # It must be pre-allocated.
  # # It is not allowed to be changed after deployment.
  # s3ServiceIPFQDN:
  #   # The pattern is IPv4 or IPv6 format
  #   ipAddr: "sample-s3-ip"
  #   # The pattern is FQDN format.
  #   fqdn: "sample-s3-fqdn"
  #
  # Optional annotations to be added in the LoadBalancer services for the
Engine IPs.
  # In case we run the Engines on private IPs, we need to add some
customized annotations to the LoadBalancer services.
  # See https://docs.microsoft.com/en-us/azure/aks/internal-lb
  # It's optional. It's not needed in most cases if we're
with public IPs.
  # loadBalancerAnnotations:
  #   service.beta.kubernetes.io/azure-load-balancer-internal: "true"
  #
  # SecretName is the name of the secret which stores the MSDP credential.
  # AutoDelete, when true, will automatically delete the secret specified
by SecretName after the
  # initial configuration. If unspecified, AutoDelete defaults to true.
  # When true, SkipPrecheck will skip webhook validation of the MSDP
credential. It is only used in data re-use
  # scenario (delete CR and re-apply with pre-existing data) as the
secret will not take effect in this scenario. It
  # can't be used in other scenarios. If unspecified, SkipPrecheck
defaults to false.
  credential:
    # The secret should be pre-created in the same namespace which has
```

```
the MSDP credential stored.
    # The secret should have a "username" and a "password" key-pairs
with the corresponding username and password values.
    # Please follow MSDP guide for the rules of the credential.
    #   https://support.cohesity.com/s/article/article-100048511
    # A secret can be created directly via kubectl command or with the
equivalent YAML file:
    #   kubectl create secret generic sample-secret --namespace
sample-namespace \
    #   --from-literal=username=<username> --from-literal=password=
<password>
    secretName: sample-secret
    # Optional
    # Default is true
    autoDelete: true
    # Optional
    # Default is false.
    # Should be specified only in data re-use scenario (aka delete and
re-apply CR with pre-existing data)
    skipPrecheck: false
  #

# s3Credential:

  #   # Use this option in conjunction
with KMS option enabled.

  #   # The secret should be pre-created in the same namespace that the
MSDP cluster is deployed.

  #   # The secret should have an "accessKey" and a "secretKey" key-pairs
with the corresponding accessKey and secretKey values.

  #   # A secret can be created directly via kubectl-msdp command:

  #   #   kubectl-msdp generate-s3-secret --namespace <namespace>
--s3secret <s3SecretName>

  #   secretName: s3-secret

  #   # Optional

  #   # Default is true
```

```
  #   autoDelete: true

  #   # Optional

  #   # Default is false.

  #   # Should be specified only in data re-use scenario (aka delete and
re-apply CR with pre-existing data)

  #   skipPrecheck: false

  # Paused is used for maintenance only. In most cases you don't need
to specify it.
  # When it's specified, MSDP operator stops reconciling the corresponding
MSDP-X (aka the CR).
  # Optional.
  # Default is false
  # paused: false
  #
  # The storage classes for logVolume, catalogVolume and dataVolumes should
be:
  #   - Backed with Azure disk CSI driver "disk.csi.azure.com" with the
managed disks, and allow volume
  #     expansion.
  #   - The Azure in-tree storage driver "kubernetes.io/azure-disk" is not
supported. You need to explicitly
  #     enable the Azure disk CSI driver when configuring your AKS cluster,
or use k8s version v1.21.x which
  #     has the Azure disk CSI driver built-in.
  #   - In LRS category.
  #   - At least Standard SSD for dev/test, and Premium SSD or Ultra Disk
for production.
  #   - The same storage class can be used for all the volumes.
  #   -
  #
  # LogVolume is the volume specification which is used to provision a
volume of an MDS or Controller
  # Pod to store the log files and core dump files.
  # It's not allowed to be changed.
  # In most cases, 5-10 GiB capacity should be big enough for one MDS or
Controller Pod to use.
  logVolume:
```

```
    storageClassName: sample-azure-disk-sc1
    resources:
      requests:
        storage: 5Gi
  #
  # CatalogVolume is the volume specification which is used to provision a
volume of an MDS or Engine
  # Pod to store the catalog and metadata. It's not allowed to be changed
unless for capacity expansion.
  # Expanding the existing catalog volumes expects short downtime of the
Engines.
  # Please note the MDS Pods don't respect the storage request in
CatalogVolume, instead they provision the
  # volumes with the minimal capacity request of 500MiB.
  catalogVolume:
    storageClassName: sample-azure-disk-sc2
    resources:
      requests:
        storage: 600Gi
  #
  # DataVolumes is a list of volume specifications which are used to
provision the volumes of
  # an Engine Pod to store the MSDP data.
  # The items are not allowed to be changed or re-ordered unless for
capacity expansion.
  # New items can be appended for adding more data volumes to each
Engine Pod.
  # Appending new data volumes or expanding the existing data volumes
expects short downtime of the Engines.
  # The allowed item number is in range 1-16. To allow the other MSDP-X
Pods (e.g. Controller, MDS) running
  # on the same node, the item number should be no more than "<the maximum
allowed volumes on the node> - 5".
  # The additional 5 data disks are for the potential one MDS Pod, one
Controller Pod or one MSDP operator Pod
  # to run on the same node with one MSDP Engine.
  dataVolumes:
    - storageClassName: sample-azure-disk-sc3
      resources:
        requests:
          storage: 8Ti
    - storageClassName: sample-azure-disk-sc3
      resources:
```

```
        requests:
          storage: 8Ti
  #
  # NodeSelector is used to schedule the MSDPScaleout Pods on the specified
nodes.
  # Optional.
  # Default is empty (aka all available nodes)
  nodeSelector:
    # e.g.
    # agentpool: nodepool2
    sample-node-label1: sampel-label-value1
    sample-node-label2: sampel-label-value2
  #
  # NBCA is the specification for MSDP-X to enable NBCA SecComm
for the Engines.
  # Optional.
  nbca:
    # The master server name
    # The allowed length is in range 1-255
    masterServer: sample-master-server-name
    # The CA SHA256 fingerprint
    # The allowed length is 95
    cafp: sample-ca-fp
    # The NBCA authentication/reissue token
    # The allowed length is 16
    # For security consideration, a token with maximum 1 user allowed and
valid for 1 day should be sufficient.
    token: sample-auth-token

    # # S3TokenSecret is the secret name that holds NBCA authentication/reissue token
for MSDP S3 service.
    # # It is used to request NBCA certificate for S3 service.
    # # It must be set if MSDP S3 service is enabled.
    # # The allowed length is in range 1-255
    # # For security consideration, a token with maximum 1 user allowed and valid for
1 day should be sufficient.
    # s3TokenSecret: sample-auth-token-secret-for-s3
  #
  # KMS includes the parameters to enable KMS for the Engines.
  # We support to enable KMS in init or post configuration.
  # We don't support to change the parameters once they have been set.
  # Optional.
  kms:
```

```
    # As either the NetBackup KMS or external KMS (EKMS) is configured or
registered on NetBackup master server, then used by
    # MSDP by calling the NetBackup API, kmsServer is the NetBackup master
server name.
    kmsServer: sample-master-server-name
    keyGroup: sample-key-group-name
  #
  # autoRegisterOST includes the parameter to enable or disable the
automatic registration of
  # the storage server, the default disk pool and storage unit when MSDP-X
configuration finishes.
  autoRegisterOST:
    # If it is true, and NBCA is enabled, the operator would register the
storage server,
    # disk pool and storage unit on the NetBackup primary server, when the
MSDP CR is deployed.
    # The first Engine FQDN is the storage server name.
    # The default disk pool is in format "default_dp_<firstEngineFQDN>".
    # The default storage unit is in format "default_stu_<firstEngineFQDN>".
    # The default maximum concurrent jobs for the STU is 240.
    # In the CR status, field "ostAutoRegisterStatus.registered" with value
True, False or Unknown indicates the registration state.
    # It's false by default.
    # Note: Please don't enable it unless with NB_9.1.2_0126+.
    enabled: true
  #
  # CorePattern is the core pattern of the nodes where the MSDPScaleout
Pods are running.
  # It's path-based. A default core path "/core/core.%e.%p.%t" will be
used if not specified.
  # In most cases, you don't need to specify it.
  # It's not allowed to be changed.
  # Optional.
  # corePattern: /sample/core/pattern/path
  #
  # tcpKeepAliveTime sets the namespaced sysctl parameter
net.ipv4.tcp_keepalive_time in Engine Pods.
  # It's in seconds.
  # The minimal allowed value is 60 and the maximum allowed value is 1800.
  # A default value 120 is used if not specified. Set it to 0 to disable
the option.
  # It's not allowed to change unless in maintenance mode (Paused=true),
and the change will not apply until the Engine Pods get restarted
```

```
  # For AKS deployment in P release, please leave it unspecified or specify
it with a value smaller than 240.
  # tcpKeepAliveTime: 120
  #
  # TCPIdleTimeout is used to change the default value for Azure Load
Balancer rules and Inbound NAT rules.
  # It's in minutes.
  # The minimal allowed value is 4 and the maximum allowed value is 30.
  # A default value 30 minutes is used if not specified. Set it to 0 to
disable the option.
  # It's not allowed to change unless in maintenance mode (Paused=true),
and the change will not apply
  # until the Engine Pods and the LoadBalancer services get recreated.
  # For AKS deployment in P release, please leave it unspecified or specify
it with a value larger than 4.
  # tcpIdleTimeout: 30
```

MSDP Scaleout CR template for EKS

```
# The MSDPScaleout CR YAML
# notes:
# The CR name should be <= 40 characters.
# The MSDP credential stored in the Secret should match MSDP credential
rules defined in https://www.veritas.com/content/support/en_US/article.
100048511
apiVersion: msdp.veritas.com/v1
kind: MSDPScaleout
metadata:
  # The CR name should not be longer than 40 characters.
  name: sample-app
  # The namespace needs to be present for the CR to be created in.
  # It is not allowed to deploy the CR in the same namespace with MSDP
operator.
  namespace: sample-namespace
spec:
  # Your Container Registry(ECR for AWS EKS) URL where
the docker images can be pulled from the k8s cluster on demand
  # The allowed length is in range 1-255
  # It is optional for BYO. The code does not check the presence or
validation.
  # User needs to specify it correctly if it is needed.
  containerRegistry: sample.url
  #
```

```
  # The MSDP version string. It is the tag of the MSDP docker images.
  # The allowed length is in range 1-64
  version: "sample-version-string"
  #
  # Size defines the number of Engine instances in the MSDP-X cluster.
  # The allowed size is between 1-16
  size: 4
  #
  # The IP and FQDN pairs are used by the Engine Pods to expose the
MSDP services.
  # The IP and FQDN in one pair should match each other correctly.
  # They must be pre-allocated.
  # The item number should match the number of Engine instances.
  # They are not allowed to be changed or re-ordered. New items can be
appended for scaling out.
  # The first FQDN is used to configure the storage server in NetBackup,
automatically if autoRegisterOST is enabled,
  # or manually by the user if not.
  serviceIPFQDNs:
    # The pattern is IPv4 or IPv6 format
    - ipAddr: "sample-ip1"
      # The pattern is FQDN format.
      fqdn: "sample-fqdn1"
    - ipAddr: "sample-ip2"
      fqdn: "sample-fqdn2"
    - ipAddr: "sample-ip3"
      fqdn: "sample-fqdn3"
    - ipAddr: "sample-ip4"
      fqdn: "sample-fqdn4"
  #
  # # s3ServiceIPFQDN is the IP and FQDN pair to expose the S3 service from the MSDP instance.
  # # The IP and FQDN in one pair should match each other correctly.
  # # It must be pre-allocated.
  # # It is not allowed to be changed after deployment.
  # s3ServiceIPFQDN:
  #   # The pattern is IPv4 or IPv6 format
  #   ipAddr: "sample-s3-ip"
  #   # The pattern is FQDN format.
  #   fqdn: "sample-s3-fqdn"
  # Optional annotations to be added in the LoadBalancer services for the
Engine IPs.
  # In case we run the Engines on private IPs, we need to add some
customized annotations to the LoadBalancer services.
```

```
  # loadBalancerAnnotations:
  #   # If it's an EKS environment, specify the following annotation
to use the internal IPs.
  #   # see https://docs.microsoft.com/en-us/amazon/aws/internal-lb
  #   service.beta.kubernetes.io/aws-load-balancer: "true"
  #   # If the internal IPs are in a different subnet as the EKS cluster,
the following annotation should be
  #   # specified as well. The subnet specified must be in the same virtual
network as the EKS cluster.
  #   service.beta.kubernetes.io/aws-load-balancer-internal-subnet:
"apps-subnet"
  #
  #   # If your cluster is EKS, the following annotation item is required.
  #   # The subnet specified must be in the same VPC as your EKS.
  #   service.beta.kubernetes.io/aws-load-balancer-subnets: "subnet-04c47
28ec4d0ecb90"
  #
  # SecretName is the name of the secret which stores the MSDP credential.
  # AutoDelete, when true, will automatically delete the secret specified
by SecretName after the
  # initial configuration. If unspecified, AutoDelete defaults to true.
  # When true, SkipPrecheck will skip webhook validation of the MSDP
credential. It is only used in data re-use
  # scenario (delete CR and re-apply with pre-existing data) as the secret
will not take effect in this scenario. It
  # cannot be used in other scenarios. If unspecified, SkipPrecheck defaults
to false.
  credential:
    # The secret should be pre-created in the same namespace which has the
MSDP credential stored.
    # The secret should have a "username" and a "password" key-pairs with
the corresponding username and password values.
    # Please follow MSDP guide for the rules of the credential.
    #   https://support.cohesity.com/s/article/article-100048511
    # A secret can be created directly via kubectl command or with the
equivalent YAML file:
    #   kubectl create secret generic sample-secret --namespace sample-
namespace \
    #   --from-literal=username=<username> --from-literal=password=
<password>
    secretName: sample-secret
    # Optional
    # Default is true
```

```
    autoDelete: true
    # Optional
    # Default is false.
    # Should be specified only in data re-use scenario (aka delete and
re-apply CR with pre-existing data)
    skipPrecheck: false
  #
# s3Credential:

  #   # Use this option in conjunction with KMS option enabled.

  #   # The secret should be pre-created in the same namespace that the MSDP cluster is deployed

  #   # The secret should have an "accessKey" and a "secretKey" key-pairs with the corresponding

  #   # A secret can be created directly via kubectl-msdp command:

  #   #   kubectl-msdp generate-s3-secret --namespace <namespace> --s3secret <s3SecretName>

  #   secretName: s3-secret

  #   # Optional

  #   # Default is true

  #   autoDelete: true

  #   # Optional

  #   # Default is false.

  #   # Should be specified only in data re-use scenario (aka delete and re-apply CR with pre-ex

  #   skipPrecheck: false
  # Paused is used for maintenance only. In most cases you do not need
to specify it.
  #
  # When it is specified, MSDP operator stops reconciling the corresponding
MSDP-X cluster (aka the CR).
  # Optional.
  # Default is false
  # paused: false
  #
```

```
  # The storage classes for logVolume, catalogVolume and dataVolumes
should be:
  #   - Backed with AWS disk CSI driver "disk.csi.aws.com" with the
managed disks, and allow volume
  #     expansion.
  #   - The AWS in-tree storage driver "kubernetes.io/aws-disk" is not
supported. You need to explicitly
  #     enable the AWS disk CSI driver when configuring your EKS cluster,
or use k8s version v1.21.x which
  #     has the AWS disk CSI driver built-in.
  #   - In LRS category.
  #   - At least Standard SSD for dev/test, and Premium SSD or Ultra Disk
for production.
  #   - The same storage class can be used for all the volumes.
  #
  # LogVolume is the volume specification which is used to provision a
volume of an MDS or Controller
  # Pod to store the log files and core dump files.
  # It is not allowed to be changed.
  # In most cases, 5-10 GiB capacity should be big enough for one MDS or
Controller Pod to use.
  logVolume:
    storageClassName: sample-AWS-disk-sc1
    resources:
      requests:
        storage: xGi
  #
  # CatalogVolume is the volume specification which is used to provision a
volume of an MDS or Engine
  # Pod to store the catalog and metadata. It is not allowed to be changed
unless for capacity expansion.
  # Expanding the existing catalog volumes expects short downtime of the
 Engines.
  # Please note the MDS Pods do not respect the storage request in
CatalogVolume, instead they provision the
  # volumes with the minimal capacity request of 500MiB.
  catalogVolume:
    storageClassName: sample-AWS-disk-sc2
    resources:
      requests:
        storage: xxxGi
  #
  # DataVolumes is a list of volume specifications which are used to
```

```
provision the volumes of
  # an Engine Pod to store the MSDP data.
  # The items are not allowed to be changed or re-ordered unless for
capacity expansion.
  # New items can be appended for adding more data volumes to each
Engine Pod.
  # Appending new data volumes or expanding the existing data volumes
expects short downtime of the Engines.
  # The allowed item number is in range 1-16. To allow the other MSDP-X
Pods (e.g. Controller, MDS) running
  # on the same node, the item number should be no more than "<the
maximum allowed volumes on the node> - 5".
  # The additional 5 data disks are for the potential one MDS Pod, one
Controller Pod or one MSDP operator Pod
  # to run on the same node with one MSDP Engine.
  dataVolumes:
    - storageClassName: sample-aws-disk-sc3
      resources:
        requests:
          storage: xxTi
    - storageClassName: sample-aws-disk-sc3
      resources:
        requests:
          storage: xxTi
  #
  # NodeSelector is used to schedule the MSDPScaleout Pods on the
specified nodes.
  # Optional.
  # Default is empty (aka all available nodes)
  nodeSelector:
    # e.g.
    # agentpool: nodegroup2
    sample-node-label1: sampel-label-value1
    sample-node-label2: sampel-label-value2
  #
  # NBCA is the specification for the MSDP-X cluster to enable NBCA
SecComm for the Engines.
  # Optional.
  nbca:
    # The master server name
    # The allowed length is in range 1-255
    masterServer: sample-master-server-name
    # The CA SHA256 fingerprint
```

```
    # The allowed length is 95
    cafp: sample-ca-fp
    # The NBCA authentication/reissue token
    # The allowed length is 16
    # For security consideration, a token with maximum 1 user allowed
and valid for 1 day should be sufficient.
    token: sample-auth-token

    # # S3TokenSecret is the secret name that holds NBCA authentication/reissue token for MSDP S
    # # It is used to request NBCA certificate for S3 service.
    # # It must be set if MSDP S3 service is enabled.
    # # The allowed length is in range 1-255
    # # For security consideration, a token with maximum 1 user allowed and valid for 1 day shou
    # s3TokenSecret: sample-auth-token-secret-for-s3
  #
  # KMS includes the parameters to enable KMS for the Engines.
  # We support to enable KMS in init or post configuration.
  # We do not support to change the parameters once they have been set.
  # Optional.
  kms:
    # As either the NetBackup KMS or external KMS (EKMS) is configured
or registered on NetBackup master server, then used by
    # MSDP by calling the NetBackup API, kmsServer is the NetBackup master
server name.
    kmsServer: sample-master-server-name
    keyGroup: sample-key-group-name
  #
  # autoRegisterOST includes the parameter to enable or disable the
automatic registration of
  # the storage server, the default disk pool and storage unit when
MSDP-X configuration finishes.
  # We do not support to change autoRegisterOST.
  autoRegisterOST:
    # If it is true, and NBCA is enabled, the operator would register
the storage server,
    # disk pool and storage unit on the NetBackup primary server, when
the MSDP CR is deployed.
    # The first Engine FQDN is the storage server name.
    # The default disk pool is in format "default_dp_<firstEngineFQDN>".
    # The default storage unit is in format "default_stu_<firstEngineFQDN>".
    # The default maximum number of concurrent jobs for the STU is 240.
    # In the CR status, field "ostAutoRegisterStatus.registered" with
value True, False or Unknown indicates the registration state.
```

```
    # It is false by default.
    enabled: true
  #
  # CorePattern is the core pattern of the nodes where the MSDPScaleout
Pods are running.
  # It is path-based. A default core path "/core/core.%e.%p.%t" will be
used if not specified.
  # In most cases, you do not need to specify it.
  # It is not allowed to be changed.
  # Optional.
  # corePattern: /sample/core/pattern/path
  #
  # tcpKeepAliveTime sets the namespaced sysctl parameter net.ipv4.tcp_
keepalive_time in Engine Pods.
  # It is in seconds.
  # The minimal allowed value is 60 and the maximum allowed value is 1800.
  # A default value 120 is used if not specified. Set it to 0 to disable
the option.
  # It is not allowed to change unless in maintenance mode (paused=true),
and the change will not apply until the Engine Pods get restarted.
  # For EKS deployment in 10.1 release, please leave it unspecified or
specify it with a value smaller than 240.
  # tcpKeepAliveTime: 120
  #
  # TCPIdleTimeout is used to change the default value for AWS Load
Balancer rules and Inbound NAT rules.
  # It is in minutes.
  # The minimal allowed value is 4 and the maximum allowed value is 30.
  # A default value 30 minutes is used if not specified. Set it to 0 to
disable the option.
  # It is not allowed to change unless in maintenance mode (paused=true),
and the change will not apply until the Engine Pods and the LoadBalancer
services get recreated.
  # For EKS deployment in 10.1 release, please leave it unspecified or
specify it with a value larger than 4.
  # tcpIdleTimeout: 30
```

MSDP Scaleout

This appendix includes the following topics:

About MSDP Scaleout

MSDP Scaleout is based on MSDP. It empowers MSDP with high resilience and scalability capabilities to simplify management and reduce total cost of ownership.

It runs on multiple nodes to represent a single storage pool for NetBackup and other Cohesity products to use. You can seamlessly scale out and scale up a MSDP Scaleout on demand. MSDP Scaleout automatically does failure detection and repair in the background.

The core MSDP services run on each node to expose the storage optimized services, and manage a part of the cluster level data and metadata. Each MSDP Scaleout node is called MSDP engine.

MSDP Scaleout components

Following are the MSDP Scaleout components:

- MDS (MetaData service)
  MDS is an independent and stackable service that provides a single system view of MSDP Scaleout. It's an etcd cluster running inside the MDS pods. These pods run on different AKS or EKS nodes. The pod name has a format of **<cr-name>-uss-mds-<1,2...>**.
  The number of pods that get created depends on the number of MSDP Scaleout engines in a cluster. These pods are controlled by the MSDP operator.
  - 1 or 2 MSDP Scaleout engines: 1 pod
  - 3 or 4 MSDP Scaleout engines: 3 pods
  - 5 or more MSDP Scaleout engines: 5 pods
- MSDP Scaleout Controller
  Controller is a singleton service and the entry point of MSDP Scaleout that monitors and repairs MSDP Engines. It controls and manages the application-level business of the MSDP Scaleout. The Deployment object name has a format of **<cr-name>-uss-controller**. It is controlled by the MSDP operator.
- MSDP Scaleout Engine
  MSDP Engines provide the ability to write deduplicated data to the storage. The name of a MSDP engine pod is the corresponding FQDN of the static IP that is specified in the CR. Each MSDP engine pod has MSDP services such as spad, spoold, and ocsd running. They are controlled by the MSDP operator.

Prerequisites for MSDP Scaleout (AKS\EKS)

A working Azure Kubernetes cluster

- Azure Kubernetes cluster
  - Your Azure Kubernetes cluster must be created with appropriate network and configuration settings.
    For a complete list of supported Azure Kubernetes service versions, refer to the Software Compatibility List (SCL) at:NetBackup Compatibility List.
  - Availability zone for AKS cluster must be disabled.
  - Cert-manager and trust-manager must be installed.
  - A Kubernetes Secret that contains the MSDP credentials.
    See “ Secret” on page 395.
  - Enable AKS Uptime SLA.

AKS Uptime SLA is recommended for a better resiliency.
For information about AKS Uptime SLA and to enable it, see Azure Kubernetes Service (AKS) Uptime SLA.

- At least one storage class is backed with Azure disk CSI storage driver "disk.csi.azure.com", and allows volume expansion. It must be in LRS category with Premium SSD. For example, the built-in storage class "managed-csi-premium". It is recommended that the storage class has "Retain" reclaim.

- Azure container registry (ACR)
  Use existing ACR or create a new one. Your Kubernetes cluster must be able to access this registry to pull the images from ACR. For more information on Azure container registry, see 'Azure Container Registry documentation' section in *Microsoft Azure Documentation*.
- Node Pool
  You must have a dedicated node pool for MSDP Scaleout created. The Azure autoscaling allows your node pool to scale dynamically as required. It is recommended that you set the minimum node number to 1 or more to bypass some limitations in AKS.
  Azure availability zone must be disabled. If Azure autoscaling is not enabled, ensure that the node number is not less than MSDP Scaleout size.
- Client machine to access AKS cluster
  - A separate computer that can access and manage your AKS cluster and ACR.
  - It must have Linux operating system.
  - It must have Docker daemon, the Kubernetes command-line tool (kubectl), and Azure CLI installed.
    The Docker storage size must be more than 6 GB. The version of kubectl must be v1.19.x or later. The version of Azure CLI must meet the AKS cluster requirements.
  - If AKS is a private cluster, see Create a private Azure Kubernetes Service cluster.
- Static Internal IPs
  If the internal IPs are used, reserve the internal IPs (avoid the IPs that are reserved by other systems) for MSDP Scaleout and add DNS records for all of them in your DNS configuration.
  The Azure static public IPs can be used but is not recommended.

If Azure static public IPs are used, create them in the node resource group for the AKS cluster. A DNS name must be assigned to each static public IP. The IPs must be in the same location of the AKS cluster.

- Ensure that the managed identity has the scope to connect to the resource group of the cluster created for cloud scale deployment.

A working AWS Kubernetes cluster

- AWS Kubernetes cluster
  - Your AWS Kubernetes cluster must be created with appropriate network and configuration settings.
    For a complete list of supported AWS Kubernetes service versions, refer to the Software Compatibility List (SCL) at:NetBackup Compatibility List
  - A Kubernetes Secret that contains the MSDP credentials.
    See “ Secret” on page 395.
  - The node group in EKS should not cross availability zone.
  - At least one storage class that is backed with Amazon EBS CSI storage driver **ebs.csi.aws.com** or with the default provisioner **kubernetes.io/aws-ebs**, and allows volume expansion. The built-in storage class is **gp2**. It is recommended that the storage class has "Retain" reclaim policy.
  - AWS Load Balancer controller must be installed on EKS.
- Amazon Elastic Container Registry (ECR)
  Use existing ECR or create a new one. Your Kubernetes cluster must be able to access this registry to pull the images from ECR.
- Node Group
  You must have a dedicated node group for MSDP Scaleout created. The node group should not cross availability zone.
  The AWS Auto Scaling allows your node group to scale dynamically as required. If AWS Auto Scaling is not enabled, ensure the node number is not less than MSDP Scaleout size.
  It is recommended that you set the minimum node number to 1 or more to bypass some limitations in EKS.
  User must assign same IAM role with required permissions to both the node groups, that is control node group and data node group.
- Client machine to access EKS cluster
  - A separate computer that can access and manage your EKS cluster and ECR.

- It must have Linux operating system.
- It must have Docker daemon, the Kubernetes command-line tool (kubectl), and AWS CLI installed.
  The Docker storage size must be more than 6 GB. The version of kubectl must be v1.19.x or later. The version of AWS CLI must meet the EKS cluster requirements.
- If EKS is a private cluster, see Creating an private Amazon EKS cluster.

Existing NetBackup environment

MSDP Scaleout connects to the existing NetBackup environment with the required network ports 1556 and 443. The NetBackup primary server must be 10.0 or later. The NetBackup environment can be anywhere, locally or remotely. It may or may not be in AKS/EKS cluster. It may or may not be in the same AKS/EKS cluster.

If the NetBackup servers are on Azure/AWS cloud, besides the NetBackup configuration requirements, the following settings are recommended. They are not MSDP-specific requirements, they just help your NetBackup environment run smoothly on Azure/AWS cloud.

- Add the following in `/usr/openv/netbackup/bp.conf`

  ```
  HOST_HAS_NAT_ENDPOINTS = YES
  ```

- Tune sysctl parameters as follows:

  ```
  net.ipv4.tcp_keepalive_time=120

  net.ipv4.ip_local_port_range = 14000 65535

  net.core.somaxconn = 1024
  ```

  Tune the max open files to 1048576 if you run concurrent jobs.

Limitations in MSDP Scaleout

MSDP Scaleout has the following limitations:

- It is not fully compliant with Federal Information Processing Standards (FIPS).
  The internal services MSDP operator, MSDP Controller, and MDS of a MSDP Scaleout are not compliant with FIPS.
  MSDP is FIPS compliant. For more information, see the *NetBackup Deduplication Guide*.
- Supports only NBCA. Does not support ECA.

- Node group cross availability zone is not supported for EKS.
- Limited node failure tolerance.
  Backup and restore can fail if AKS/EKS node fails. If MSDP operator detects the MSDP Scaleout pod failure, it attempts to restart it and perform a repair operation automatically. The repair operation can be delayed if AWS or Azure infrastructure or Kubernetes do not allow the pod to be restarted.
  An Azure or AWS EBS volume cannot be attached to two different nodes at the same time. When the node which a volume is attached to fails, MSDP operator cannot run the same pod with the same Azure or AWS EBS volume on another node until the failed node is repaired or deleted by AKS or EKS.
  A node auto-repair may take more than 20 minutes to finish. In some cases, it may be necessary to bring the node backup manually.
  See Azure Kubernetes Service (AKS) node auto-repair
  See Amazon Elastic Kubernetes Service (EKS) Documentation
- IPv6 is not supported.
- Changes to MSDP Scaleout credential autoDelete, which allows automatic deletion of credential after use, is not supported. The autoDelete value can only be set during initial deployment.

MSDP Scaleout configuration

Note: This chapter is for deploying a standalone MSDP Scalout without the environment operator or Helm charts.

Initializing the MSDP operator

Run the following command to initialize MSDP operator.

- For AKS
  ```
  kubectl msdp init -i <acr-url>/msdp-operator:<version> -s
  <storage-class-name> [-l agentpool=<nodepool-name>]
  ```
- For EKS
  ```
  kubectl msdp init -i <ecr-url>/msdp-operator:<version> -s
  <storage-class-name> [-l agentpool=<nodegroup-name>]
  ```

You can use the following `init` command options.

Table B-1 init command options

Option	Description
`-i`	MSDP operator images on your ACR or ECR.
`-s`	The storage class name.
`-l`	Node selector of the MSDP operator. By default, each node pool has a unique label with key-value pair. ■ AKS: *agentpool=<nodepool-name>* ■ EKS: *agentpool=<nodegroup-name>* If you have assigned a different and cluster-wise unique label for the node pool, you can use that instead of *agentpool*.
`-c`	Core pattern of the operator pod. Default value: "/core/core.%e.%p.%t"
`-d`	Enable debug-level logging in MSDP operator.
`-a`	The maximum number of days to retain the old log files. Range: 1-365 Default value: 28
`-u`	The maximum number of old log files to retain. Range: 1-20 Default value: 20
`-n`	Namespace scope for this request. Default value: msdp-operator-system
`-o`	Generate MSDP operator CRD YAML.
`-h`	Help for the `init` command.

This command installs Custom Resource Definitions (CRD) **msdpscaleouts.msdp.veritas.com** and deploys MSDP operator in the Kubernetes environment. MSDP operator runs with Deployment Kubernetes workload type with single replica size in the default namespace **msdp-operator-system**.

MSDP operator also exposes the following services:

- Webhook service

The webhook service is consumed by Kubernetes api-server to mutate and validate the user inputs and changes of the MSDP CR for the MSDP Scaleout configuration.

- Metrics service
  AKS: The metric service is consumed by Kubernetes/AKS for Azure Container Insight integration.
  EKS: The metric service is consumed by Kubernetes/EKS for Amazon CloudWatch integration.

You can deploy only one MSDP operator instance in a Kubernetes cluster.

Run the following command to check the MSDP operator status.

```
kubectl -n msdp-operator-system get pods -o wide
```

Configuring MSDP Scaleout

After you push the docker images to ACR and initialize MSDP operator, configure MSDP Scaleout.

To configure MSDP Scaleout

1. Create a dedicated namespace for MSDP Scaleout to run.

   ```
   kubectl create ns <sample-namespace>
   ```

2. Create an MSDP Scaleout Secret. The Secret is used in CR.

   ```
   kubectl apply -f <secret-yaml-file>
   ```

   See “ Secret” on page 395.

3. (Optional) Create an MSDP S3 root credential secret. The secret is used in CR.

   ```
   $ kubectl msdp generate-s3-secret --namespace <sample-namespace>
   --s3secret <s3-secret-name>
   ```

4. Display the custom resource (CR) template.

   ```
   kubectl msdp show -c
   ```

5. Save the CR template.

   ```
   kubectl msdp show -c -f <file path>
   ```

6. Edit the CR file in the text editor.

7 Apply the CR file to the cluster.

Caution: Add `MSDP_SERVER = <first Engine FQDN>` in `/usr/openv/netbackup/bp.conf` file on the NetBackup primary server before applying the CR YAML.

```
kubectl apply -f <sample-cr-yaml>
```

8 Monitor the configuration progress.

```
kubectl get all -n <namespace> -o wide
```

In the **STATUS** column, if the readiness state for the controller, MDS and engine pods are all **Running**, it means that the configuration has completed successfully.

In the **READY** column for engines, 2/2 or 3/3 indicates that the engine configuration has completed successfully.

9 If you specified `spec.autoRegisterOST.enabled: true` in the CR, when the MSDP engines are configured, the MSDP operator automatically registers the storage server, a default disk pool, and a default storage unit in the NetBackup primary server.

A field **ostAutoRegisterStatus** in the Status section indicates the registration status. If **ostAutoRegisterStatus.registered** is **True**, it means that the registration has completed successfully.

You can run the following command to check the status:

```
kubectl get msdpscaleouts.msdp.veritas.com -n <sample-namespace>
```

You can find the storage server, the default disk pool, and storage unit on the Web UI of the NetBackup primary server.

Configuring the MSDP cloud in MSDP Scaleout

After you configure the local LSU, you can also configure MSDP cloud in MSDP Scaleout.

For more information about MSDP cloud support, see the *NetBackup Deduplication Guide*.

Using MSDP Scaleout as a single storage pool in NetBackup

If you did not enable automatic registration of the storage server (autoRegisterOST) in the CR, you can configure it manually using the NetBackup Web UI.

See " MSDP Scaleout CR" on page 396.

To use MSDP Scaleout as a single storage pool in NetBackup

1. Follow the OpenStorage wizard with storage type **PureDisk** to create the storage server using the first Engine FQDN.

   MSDP storage server credentials are defined in the Secret resource.

   For more information, see *Create a Cloud storage, OpenStorage, or AdvancedDisk storage server* topic of the *NetBackup Web UI Administrator's Guide*.

2. Follow the MSDP wizard to create the disk pool.

   For more information, see *Create a disk pool* topic of the *NetBackup Web UI Administrator's Guide*.

3. Follow the MSDP wizard to the storage unit.

   For more information, see *Create a disk pool* topic of the *NetBackup Web UI Administrator's Guide*.

You can use MSDP Scaleout like the legacy single-node MSDP.

Using S3 service in MSDP Scaleout

Ensure that S3 credential secret field and S3 service IP/FQDN fields are updated in the environment YAML file or MSDP Scaleout CR YAML file.

See "Enabling MSDP S3 service after MSDP Scaleout is deployed" on page 422.

To use S3 service in MSDP Scaleout

1. Check the S3 service status.

   Run the following command to check if S3 service is configured or not:

   ```
   $ kubectl get msdpscaleouts.msdp.veritas.com/<MSDP Scaleout CR
   name> -n <sample-namespace> -o=jsonpath={.status.s3srvConfigured}
   ```

   If the command output is true, S3 service is configured and ready for use. Otherwise, wait for the flag to be true. The flag changes to true automatically after all MSDP Scaleout resources are ready.

2. Use the following URL to access S3 service in MSDP Scaleout:

   ```
   https://<MSDP-S3-FQDN>
   ```

Limitations:

- S3 service in MSDP Scaleout only supports NBCA certificates.

You can use the CA certificate in NetBackup primary server to bypass SSL warnings when accessing S3 service. The CA certificate file path is `/usr/openv/var/webtruststore/cacert.pem`.

For example, when using AWS CLI you can use `-ca-bundle` parameter to specify CA certificate file path to bypass SSL warnings.

- The region name of MSDP S3 service is the LSU name that is used to store S3 data. Set the default region name to **PureDiskVolume** to use the MSDP local LSU to store the S3 data.
- Recommend 500~800 concurrent requests based on your Kubernetes node's performance.

Enabling MSDP S3 service after MSDP Scaleout is deployed

If MSDP S3 service is not enabled during the initial deployment, you can enable it later.

Identify the MSDP Scaleout deployment method of your deployment. Some of the steps in this procedure differ based on the MSDP Scaleout deployment method.

- MSDP Scaleout is deployed with environment YAML in the following scenarios:
  - NetBackup and MSDP are deployed together in the same cluster
  - Cloud Scale is deployment using the Terraform
  - Cloud Scale is deployment using the Maketplace
- MSDP Scaleout is deployed with MSDP Scaleout YAML in the following scenarios:
  - NetBackup and MSDP are not deployed together in the same cluster

To enable MSDP S3 service

1. Generate S3 root credential. Run the following command to generate the secret:

   ```
   $ kubectl msdp generate-s3-secret --namespace <sample-namespace>
   --s3secret <s3-secret-name>
   ```

   Ensure that you save the S3 credential at a secure place after it is generated for later use.

   If MSDP kubectl plug-in is not installed, copy MSDP kubectl plug-in from the operator TAR folder to a directory from where you access the cluster host. This directory can be configured in the PATH environment variable so that kubectl can load MSDP kubectl as a plug-in automatically.

   For example,

   ```
   $ cp ./VRTSk8s-netbackup-<version>-0065/bin/kubectl-msdp
   /usr/local/bin/
   ```

2. Input S3 credential field and S3 IP/FQDN fields in existing CR resources.

   - If the MSDP Scaleout is deployed with environment YAML, run the following command to update the `spec.msdpScaleouts[<index>].s3Credential` and `spec.msdpScaleouts[<index>].s3Ip` fields in the existing CR resources:

     ```
     $ kubectl edit environments.netbackup.veritas.com
     <environmentCR_name> -n <sample-namespace>
     ```

     Content format:

     ```
     msdpScaleouts:

       - credential:
           autoDelete: true
           secretName: msdp-creds
           skipPrecheck: false

         s3Credential:
           secretName: <s3secretName>

         s3Ip:
           ipAddr: <s3IpAddress>
           fqdn: <s3Fqdn>
     ```

   - If the MSDP Scaleout is deployed with MSDP Scaleout YAML, run the following command to update the `spec.s3Credential` and `spec.s3ServiceIPFQDN` fields in the existing CR resources:

```
$ kubectl edit msdpscaleouts.msdp.veritas.com.msdp.veritas.com
<MSDP Scaleout CR name> -n <sample-namespace>
```

Content format:

```
spec:

    credential:
      autoDelete: true
      secretName: msdp-creds
      skipPrecheck: false

    s3Credential:
      secretName: <s3secretName>

    s3ServiceIPFQDN:
      ipAddr: <s3IpAddress>
      fqdn: <s3Fqdn>
```

3 Provide the NetBackup token for MSDP S3.

If the MSDP Scaleout is deployed using helm command, skip this step. This step is done automatically by the environment operator.

If the MSDP Scaleout is deployed with MSDP Scaleout YAML, create a token from NetBackup web UI. Navigate to **Security > Tokens > Add**. Run the following command to create a token secret in Kubernetes:

```
$ kubectl create secret generic <S3-token-secret-name> --namespace
<sample-namespace> --from-literal=token=<token-value>
```

Run the following command to update the `spec.nbca.s3TokenSecret` field in the existing CR resources:

```
$ kubectl edit msdpscaleout <MSDP Scaleout CR name> -n
<sample-namespace>
```

Content format:

```
spec:
    nbca:
      s3TokenSecret: <S3-token-secret-name>
```

Wait for a few minutes. MSDP operator enables S3 service automatically.

4 Run the following command to check the S3 service status:

```
$ kubectl get msdpscaleouts.msdp.veritas.com/<MSDP Scaleout CR
name> -o=jsonpath={.status.s3srvConfigured}
```

If the command output is true, S3 service is configured and ready for use.

Installing the docker images and binaries for MSDP Scaleout (without environment operators or Helm charts)

The MSDP package `VRTSpddek.tar.gz` for Kubernetes includes the following:

- A docker image for MSDP operator
- 3 docker images for MSDP Scaleout: uss-controller, uss-mds, and uss-engine
- A kubectl plugin: kubectl-msdp

To install the docker images and binaries for AKS

1. Download `VRTSpddek.tar.gz` from the Cohesity site.
2. Load the docker images to your docker storage.

   ```
   tar -zxvf VRTSpddek.tar.gz
   ```

   ```
   ls VRTSpddek-*/images/*.tar.gz|xargs -i docker load -i {}
   ```

3. Copy MSDP kubectl plugin to a directory from where you access AKS or EKS host. This directory can be configured in the PATH environment variable so that kubectl can load kubectl-msdp as a plugin automatically.

   For example,

   ```
   cp ./VRTSpddek-*/bin/kubectl-msdp /usr/local/bin/
   ```

4. Push the docker images to the ACR. Keep the image name and version same as original.

   ```
   docker login <your-acr-url>
   for image in msdp-operator uss-mds uss-controller uss-engine; do \
       docker image tag $image:<version> <your-acr-url>/$image:<version>; \
       docker push <your-acr-url>/$image:<version>; \
   done
   ```

To install the docker images and binaries for EKS

1. Download `VRTSpddek.tar.gz` from the Cohesity site.
2. Load the docker images to your docker storage.

   ```
   tar -zxvf VRTSpddek.tar.gz
   ```

   ```
   ls VRTSpddek-*/images/*.tar.gz|xargs -i docker load -i {}
   ```

3. Copy MSDP kubectl plugin to a directory from where you access AKS or EKS host. This directory can be configured in the PATH environment variable so that kubectl can load kubectl-msdp as a plugin automatically.

   For example,

   ```
   cp ./VRTSpddek-*/bin/kubectl-msdp /usr/local/bin/
   ```

4. Push the docker images to the ECR.

   - Log in.

     ```
     aws ecr get-login-password \
     --region <region> \
     | docker login \
     --username AWS \
     ```

```
--password-stdin \
<aws_account_id>.dkr.ecr.<region>.amazonaws.com
```

- Create a repository.
  Refer to the "Creating a private repository" section of the *AWS documentation*.
- Push the docker images to ECR. Keep the image name and version same as original.

```
for image in msdp-operator uss-mds uss-controller
uss-engine; do \
docker image tag $image:<version> <your-ecr-url>/$image:<version>; \
docker push <your-ecr-url>/$image:<version>; \
done
```

Deploying MSDP Scaleout

You must deploy MSDP Scaleout solution in your Kubernetes Cluster environment. AKS or EKS must be created with appropriate network and configuration settings.

You can have multiple MSDP Scaleout deployments in the same cluster. Ensure that each MSDP Scaleout deployment runs in a dedicated namespace on a dedicated node pool.

Before you deploy the solution, ensure that your environment meets the requirements.

See "Prerequisites for MSDP Scaleout (AKS\EKS)" on page 413.

Table B-2 MSDP Scaleout deployment steps

Step	Task	Description
Step 1	Install the docker images and binaries.	See "Installing the docker images and binaries for MSDP Scaleout (without environment operators or Helm charts)" on page 425.
Step 2	Initialize MSDP operator.	See "Initializing the MSDP operator" on page 417.
Step 3	Configuring MSDP Scaleout.	See "Configuring MSDP Scaleout" on page 419.

Table B-2 MSDP Scaleout deployment steps *(continued)*

Step	Task	Description
Step 4	Use MSDP Scaleout as a single storage pool in NetBackup.	See "Using MSDP Scaleout as a single storage pool in NetBackup" on page 420.

Managing MSDP Scaleout

Adding MSDP engines

You can add new MSDP engines by updating MSDP Scaleout in Cloud Scale helm release. You can add maximum 16 MSDP engines.

Prerequisites:

- Allocate new static IP/FQDN pairs in the same node resource group.
- The node number must not be less than the MSDP Scaleout size that you plan to change.

To add the MSDP engines by updating the Cloud Scale helm release

1. Run the following command to obtain the Cloud Scale release name:

   ```
   Helm ls -n <namespace>
   ```

2. Navigate to the helm directory within the extracted tar location.
3. Run the following command to add new engine:

   ```
   helm upgrade <release name> cloudscale-<version>.tgz -n
   <namespace> --reuse-values

   --set environment.msdpScaleouts.replicas=<total no, including new
   engine>

   --set environment.msdpScaleouts.ipList[0].fqdn=<existing engine
   FQDN>

   --set environment.msdpScaleouts.ipList[0].ipAddr=<existing engine
   IP>

   --set environment.msdpScaleouts.ipList[1].fqdn=<FQDN for new
   engine to be added>

   --set environment.msdpScaleouts.ipList[1].ipAddr=<IP for new
   engine to be added>
   ```

The MSDP Scaleout services are not interrupted when MSDP engines are added.

Adding data volumes

You can add the data volumes by updating the Cloud Scale helm release.

To add the data volumes by updating the Cloud Scale helm release

1 Run the following command to obtain the Cloud Scale release name:

```
Helm ls -n <namespace>
```

2 Navigate to the helm directory within the extracted tar location.

3 Run the following command to add new data volume:

```
helm upgrade <cloudscale release name> cloudscale-<version>.tgz
-n netbackup --reuse-values

--set
environment.msdpScaleouts.storage.dataVolumes[0].capacity=<existing
volume capacity>

--set
environment.msdpScaleouts.storage.dataVolumes[0].storageClassName=<<existing
volume storage class>

--set
environment.msdpScaleouts.storage.dataVolumes[1].capacity=<capacity
for new volume>

--set
environment.msdpScaleouts.storage.dataVolumes[1].storageClassName=<storage
class for new volume>
```

Each MSDP engine can support up to 16 data volumes.

It is recommended that you use the same data volume size if you add multiple volumes.

Note: Due to some Kubernetes restrictions, MSDP operator restarts the engine pods for attaching the existing and new volumes, which can cause the short downtime of the services.

Expanding existing data or catalog volumes

You can expand the existing data or catalog volumes by updating the Cloud Scale helm release.

To expand the data or catalog volumes by updating the Cloud Scale helm release

1. Navigate to the helm directory within the extracted tar location.
2. Run the following command to obtain the Cloud Scale release name:

   ```
   Helm ls -n <namespace>
   ```

3. Run the following command to expand the data volume:

   ```
   helm upgrade <cloudscale helm release> cloudscale-<version>.tgz
   -n netbackup --reuse-values

   --set
   environment.msdpScaleouts.storage.dataVolumes[0].capacity=<new-value>

   --set
   environment.msdpScaleouts.storage.dataVolumes[0].storageClassName=<new-value>
   ```

4. Run the following command to expand the catalog volume:

   ```
   helm upgrade <cloudscale helm release> cloudscale-<version>.tgz
   -n netbackup --reuse-values

   --set
   environment.msdpScaleouts.storage.catalog[0].capacity=<new-value>

   --set
   environment.msdpScaleouts.storage.catalog[0].storageClassName=<new-value>
   ```

Sometimes Azure disk or Amazon EBS CSI driver may not respond the volume expansion request promptly. In this case, the operator retries the request by adding 1 byte to the requested volume size to trigger the volume expansion again. If it is successful, the actual volume capacity could be slightly larger than the requested size.

Due to the limitation of Azure disk or Amazon CSI storage driver, the engine pods need to be restarted for resizing the existing volumes. This can cause the short downtime of the services.

MSDP Scaleout does not support the following:

- Cannot shrink the volume size.
- Cannot change the existing data volumes other than for storage expansion.
- Cannot expand the log volume size. You can do it manually. See "Manual storage expansion" on page 431.
- Cannot expand the data volume size for MDS pods. You can do it manually. See "Manual storage expansion" on page 431.

Manual storage expansion

You also can manually expand storage size by expanding PVC size.

To expand the data or catalog volumes

1. Run the following command to obtain the Cloud Scale release name:

   ```
   Helm ls -n <namespace>
   ```

2. Navigate to the helm directory within the extracted tar location.
3. Run the following command to stop MSDP operator from reconciling and repairing the pods automatically:

   ```
   helm upgrade <release-name>cloudscale-<version>.tgz -n netbackup
   --reuse-values --set environment.msdpScaleouts.paused=true
   ```

4. Patch the corresponding PVCs.

   ```
   kubectl patch pvc <pvc-name> --type merge --patch '{"spec":
   {"resources": {"requests": {"storage": "<requested-size>"}}}}'
   -n <sample-namespace>
   ```

5. Run the following command to continue to reconcile and repair the pods automatically:

   ```
   helm upgrade <release-name>cloudscale-<version>.tgz -n netbackup
   --reuse-values --set environment.msdpScaleouts.paused=false
   ```

Note: If you add new MSDP Engines later, the new Engines will respect the CR specification only. Your manual changes would not be respected by the new Engines.

MSDP Scaleout scaling recommendations

Following are the scaling recommendations for the MSDP Scaleout:

- Allocate the data volumes of the similar sizes for MSDP to have better load balancing performance.
- Each data volume size is more than 4 TB.
- Have multiple data volumes for each engine to gain better throughput.
- Split a bigger backup policy to smaller ones
  In most cases, one backup job goes to one MSDP engine at the same time even if multistream is enabled for the backup policy. If the current MSDP engine, which is taking a backup job hits the high space watermark, the following backup data would be sent to a second MSDP engine. If the backup data is too big for up to 2 MSDP engines to persist, the backup job fails. When more MSDP engines

are added, the backup jobs may not be evenly balanced on each MSDP engine at the first a few hours or days. If the situation keeps longer beyond your expectation, consider to re-plan the backup policies, by splitting a bigger backup policy to two smaller ones, to help MSDP Scaleout to balance the new backup jobs more faster.

- After scaling up, the memory and CPU of the existing node pool may not meet the performance requirements anymore. In this case, you can add more memory and CPU by upgrading to the higher instance type to improve the existing node pool performance or create another node pool with higher instance type and update the node-selector for the CR accordingly. If you create another node pool, the new node-selector does not take effect until you manually delete the pods and deployments from the old node pool, or delete the old node pool directly to have the pods re-scheduled to the new node pool.
- Ensure that each AKS or EKS node supports mounting the number of data volumes plus 5 of the data disks.
  For example, if you have 16 data volumes for each engine, then each your AKS or EKS node should support mounting at least 21 data disks. The additional 5 data disks are for the potential MDS pod, Controller pod or MSDP operator pod to run on the same node with MSDP engine.

MSDP Cloud backup and disaster recovery

For information about MSDP cloud backup and disaster recovery, see MSDP Cloud section of the *NetBackup Deduplication Guide*.

Note: In case of disaster recovery of NetBackup environment (that is, primary, media and MSDP), perform the primary catalog recovery first and then proceed with MSDP disaster recovery steps.See "Backing up a catalog" on page 233.

About the reserved storage space

About 1 TB storage space is reserved by default on each MSDP engine for each cloud LSU.

The 1 TB storage space is selected from one of the data volumes of every engine. It requires each engine at least has one data volume, which has more than 1 TB available storage space, when a cloud LSU is to be configured. Otherwise, the configuration of the cloud LSU fails.

Cloud LSU disaster recovery

Scenario 1: MSDP Scaleout and its data is lost and the NetBackup primary server remains unchanged and works well

1. Redeploy MSDP Scaleout on a cluster by using the same CR parameters and NetBackup re-issue token.
2. If the LSU cloud alias does not exist, you can use the following command to add it.

   ```
   /usr/openv/netbackup/bin/admincmd/csconfig cldinstance -as -in
   <instance-name> -sts <storage-server-name> -lsu_name <lsu-name>
   ```

   When MSDP Scaleout is up and running, re-use the cloud LSU on NetBackup primary server.

   ```
   /usr/openv/netbackup/bin/admincmd/nbdevconfig -setconfig
   -storage_server <STORAGESERVERNAME> -stype PureDisk -configlist
   <configuration file>
   ```

   Credentials, bucket name, and sub bucket name must be the same as the recovered Cloud LSU configuration in the previous MSDP Scaleout deployment.

   Configuration file template:

   ```
   V7.5 "operation" "reuse-lsu-cloud" string
   V7.5 "lsuName" "LSUNAME" string
   V7.5 "cmsCredName" "XXX" string
   V7.5 "lsuCloudAlias" "<STORAGESERVERNAME_LSUNAME>" string
   V7.5 "lsuCloudBucketName" "XXX" string
   V7.5 "lsuCloudBucketSubName" "XXX" string
   V7.5 "lsuKmsServerName" "XXX" string
   ```

   Note: For Veritas Alta Recovery Vault Azure storage, the `cmsCredName` is a credential name and `cmsCredName` can be any string. Add recovery vault credential in the CMS using the NetBackup web UI and provide the credential name for `cmsCredName`. For more information, see *About Veritas Alta Recovery Vault Azure* topic in *NetBackup Deduplication Guide*.

   ---

3. On the first MSDP Engine of MSDP Scaleout, run the following command for each cloud LSU:

   ```
   sudo -E -u msdpsvc /usr/openv/pdde/pdcr/bin/cacontrol --catalog
   clouddr <LSUNAME>
   ```

4. Restart the MSDP services in the MSDP Scaleout.

   Option 1: Manually delete all the MSDP engine pods.

   ```
   kubectl delete pod <sample-engine-pod> -n <sample-cr-namespace>
   ```

   Option 2: Stop MSDP services in each MSDP engine pod. MSDP service starts automatically.

   ```
   kubectl exec <sample-engine-pod> -n <sample-cr-namespace> -c
   uss-engine -- /usr/openv/pdde/pdconfigure/pdde stop
   ```

   Note: After this step, the MSDP storge server status may appear as `down` on the NetBackup primary server. The status changes to `up` automatically after the MSDP services are restarted in a few minutes.

   If the status does not change, run the following command on the primary server to update MSDP storage server status manually:

   ```
   /usr/openv/volmgr/bin/tpconfig -update -storage_server
   <storage-server-name> -stype PureDisk -sts_user_id
   <storage-server-user-name> -password <storage-server-password>
   ```

   ---

5. If MSDP S3 service is configured, restart MSDP S3 service after MSDP services are restarted.

   ```
   kubectl exec <sample-engine-pod> -n <sample-cr-namespace> -c
   uss-engine -- systemctl restart pdde-s3srv
   ```

Scenario 2: MSDP Scaleout and its data is lost and the NetBackup primary server was destroyed and is re-installed

1. Redeploy MSDP Scaleout on a cluster by using the same CR parameters and new NetBackup token.
2. When MSDP Scaleout is up and running, reuse the cloud LSU on NetBackup primary server.

   ```
   /usr/openv/netbackup/bin/admincmd/nbdevconfig -setconfig
   -storage_server <STORAGESERVERNAME> -stype PureDisk -configlist
   <configuration file>
   ```

   Credentials, bucket name, and sub bucket name must be the same as the recovered Cloud LSU configuration in previous MSDP Scaleout deployment.

   Configuration file template:

   ```
   V7.5 "operation" "reuse-lsu-cloud" string
   V7.5 "lsuName" "LSUNAME" string
   V7.5 "cmsCredName" "XXX" string
   V7.5 "lsuCloudAlias" "<STORAGESERVERNAME_LSUNAME>" string
   V7.5 "lsuCloudBucketName" "XXX" string
   V7.5 "lsuCloudBucketSubName" "XXX" string
   V7.5 "lsuKmsServerName" "XXX" string
   ```

   If KMS is enabled, setup KMS server and import the KMS keys.

   If the LSU cloud alias does not exist, you can use the following command to add it.

   ```
   /usr/openv/netbackup/bin/admincmd/csconfig cldinstance -as -in
   <instance-name> -sts <storage-server-name> -lsu_name <lsu-name>
   ```

   Note: For Veritas Alta Recovery Vault Azure storage, the `cmsCredName` is a credential name and `cmsCredName` can be any string. Add recovery vault credential in the CMS using the NetBackup web UI and provide the credential name for `cmsCredName`. For more information, see *About Veritas Alta Recovery Vault Azure* topic in *NetBackup Deduplication Guide*.

   ---

3. On the first MSDP Engine of MSDP Scaleout, run the following command for each cloud LSU:

   ```
   sudo -E -u msdpsvc /usr/openv/pdde/pdcr/bin/cacontrol --catalog
   clouddr <LSUNAME>
   ```

4 Restart the MSDP services in the MSDP Scaleout.

Option 1: Manually delete all the MSDP engine pods.

```
kubectl delete <sample-engine-pod> -n <sample-cr-namespace>
```

Option 2: Stop MSDP services in each MSDP engine pod.

```
kubectl exec <sample-engine-pod> -n <sample-cr-namespace> -c
uss-engine -- /usr/openv/pdde/pdconfigure/pdde stop
```

Note: After this step, the MSDP storge server status may appear as `down` on the NetBackup primary server. The status changes to `up` automatically after the MSDP services are restarted in a few minutes.

If the status does not change, run the following command on the primary server to update MSDP storage server status manually:

```
/usr/openv/volmgr/bin/tpconfig -update -storage_server
<storage-server-name> -stype PureDisk -sts_user_id
<storage-server-user-name> -password <storage-server-password>
```

---

5 If MSDP S3 service is configured, restart MSDP S3 service after MSDP services are restarted.

```
kubectl exec <sample-engine-pod> -n <sample-cr-namespace> -c
uss-engine -- systemctl restart pdde-s3srv
```

6 Create disk pool for the cloud LSU on NetBackup server.

7 Do two-phase image importing.

See the *NetBackup Administrator's Guide, Volume I*

For information about other DR scenarios, see *NetBackup Deduplication Guide*.

Recovering MSDP S3 IAM configurations from cloud LSU

If MSDP S3 is enabled, you can recover the MSDP S3 IAM configurations from the cloud LSU that is recovered from the disaster.

To recover the MSDP S3 IAM configurations from the cloud LSU

- Run the following command.

  ```
  $ kubectl exec -it <first-MSDP-engine-FQDN> -n <sample-namespace>
  -c uss-engine -- /usr/openv/pdde/vxs3/cfg/script/s3srv_config.sh
  --recover-iam-config <LSU name>
  ```

  The command displays the IAM configurations in the cloud LSU and current IAM configurations.

  The following warning appears:

  ```
  WARNING: This operation overwrites current IAM configurations
  with the IAM configurations in cloud LSU.
  ```

  To overwrite the current IAM configurations, type the following and press **Enter**.

  ```
  overwrite-with-<cloud_LSU_name>
  ```

  Note: Do not run `/usr/openv/pdde/vxs3/cfg/script/s3srv_config.sh --reset-iam-root` command before this command. It overwrites the IAM configurations in the cloud LSU.

MSDP multi-domain support

An MSDP storage server is configured in a NetBackup media server. The NetBackup media servers and clients in the NetBackup domain can use this storage server. By default, the NetBackup media servers and clients cannot directly use an MSDP storage server from another NetBackup domain. For example, NetBackup media servers or clients cannot backup data to an MSDP storage server from another NetBackup domain.

To use an MSDP storage server from another NetBackup domain, the MSDP storage server must have multiple MSDP users. Then NetBackup media servers or clients can use the MSDP storage server from another NetBackup domain by using a different MSDP user. Multiple NetBackup domains can use the same MSDP storage server but each NetBackup domain must use a different MSDP user to access that MSDP storage server.

For more information, See *NetBackup Deduplication Guide*.

When you add a new MSDP user, the command `spauser` must be executed in the first MSDP engine of MSDP Scaleout, not on any of the NetBackup servers.

Ensure that you run MSDP commands with non-root user **msdpsvc** after logging into an engine pod.

For example, `sudo -E -u msdpsvc /usr/openv/pdde/pdcr/bin/spauser`

Configuring Auto Image Replication

The backups that are generated in one NetBackup domain can be replicated to storage in one or more target NetBackup domains. This process is referred to as Auto Image Replication (A.I.R.).

You can configure Auto Image Replication in NetBackup, which uses MSDP Scaleout storage servers.

To configure Auto Image Replication

1. Sign in to the NetBackup web UI on both the replication source and the target domain.
2. For the replication source and the target domain, add the other NetBackup primary server as the trusted primary server.

   For more information, see the *NetBackup Web UI Administrator's Guide*.
3. For the replication source domain, get the MSDP_SERVER name from the NetBackup web UI.

   Navigate to **Storage > Disk storage**. Then click the **Storage servers** tab.
4. Add MSDP_SERVER in the primary server of replication target domain. Log in to the target primary server and run the following command:

   ```
   echo "MSDP_SERVER = <Source MSDP server name>" >>
   /usr/openv/netbackup/bp.conf
   ```
5. Get the token from the target domain NetBackup web UI.

   Navigate to **Security > Tokens**. Enter the token name and other required details. Click **Create**.

   For more information, see the *NetBackup Web UI Administrator's Guide*.
6. Add replication targets for the disk pool in the replication source domain.

   Open **Storage > Disk storage**. Then click the **Storage servers** tab.

   On the **Disk pools** tab, click on the disk pool link.

   Click **Add** to add the replication target.
7. In the **Add replication targets** page:
   - Select the replication target primary server.
   - Provide the target domain token.
   - Select the target volume.

- Provide the target storage credentials.

Click **Add**.

About MSDP Scaleout logging and troubleshooting

- AKS troubleshooting
  See AKS troubleshooting page of *Azure documentation*.
- EKS troubleshooting
  See Amazon EKS troubleshooting page of the *AWS documentation*.
- Logs and core dumps files in MSDP Scaleout
  MSDP Operator, Controller, and MDS pod logs are stored in */log* location of the pods.
- Collect the logs and inspection information
  You can collect the logs and inspection information for MSDP Scaleout for troubleshooting purpose.
  See "Collecting the logs and the inspection information" on page 439.

Collecting the logs and the inspection information

You can collect the logs and inspection information for MSDP Scaleout for troubleshooting purpose.

Run the command `kubectl msdp collect-logs`

For example, `kubectl msdp collect-logs -o <output path> [-n <MSDP operator namespace>] [-c <MSDP applications namespace(s)>]`

Table B-3 collect-logs command options

Option	Description
`-c`	Comma-separated namespaces of MSDP applications. **Note:** If not specified, it collects MSDP applications of all namespaces.
`-f`	Output format of logs/core files/MSDP history files. Available options: targz: Copy logs/core files/MSDP history files from containers and compress them by tar/gzip. raw: Copy logs/core files/MSDP history files from containers as same format in the containers. Default value: targz

Table B-3 collect-logs command options *(continued)*

Option	Description
`-n`	Namespace of MSDP operator. Default value: msdp-operator-system
`-o`	Output path of the log file.

MSDP Scaleout maintenance

Pausing the MSDP Scaleout operator for maintenance

For maintenance purpose, if you want the operator to stop reconciling the resources of one CR but do not affect the resources of the other CRs, you can pause the MSDP Scaleout operator.

To pause the MSDP Scaleout operator, run the following command from helm directory within the extracted tar location:

```
helm upgrade <release-name>cloudscale-<version>.tgz -n netbackup
--reuse-values --set environment.msdpScaleouts.paused=true
```

Do not forcibly delete the deployment resource of MSDP Scaleout operator.

Logging in to the pods

You can log in to the pods for the maintenance purpose.

To log in to the pod, run the `kubectl` executable file.

Run MSDP commands with non-root user **msdpsvc** after logging in to an engine pod.

For example, `sudo -E -u msdpsvc <command>`

The MSDP Scaleout services in an engine pods are running with non-root user **msdpsvc**. If you run the MSDP Scaleout services or commands with the root user, MSDP Scaleout may stop working due to file permissions issues.

Reinstalling MSDP Scaleout operator

When you undeploy MSDP Scaleout operator, the MSDP Scaleout CRD is removed from the cluster. It also deletes all the existing MSDP Scaleout on the cluster. The PVC for the operator logs is also deleted. However, the MSDP Scaleout critical data and metadata is not deleted.

To reinstall MSDP Scaleout operator

1. Navigate to `cd $NB_DIR/bin` directory.

   where NB_DIR is the directory where the tar file has been unzipped.

   Run the following command to delete the MSDP Scaleout operator:

   ```
   ./kubectl-msdp delete -n netbackup-operator-system
   ```

   `-n`: Namespace scope for this request.

2 Run the following command to redeploy the operator.

```
# helm upgrade --install operators operators-<version>.tgz \
      --create-namespace \
      --namespace netbackup-operator-system \
      --set global.platform=${platform} \
      --set global.operatorNamespace=netbackup-operator-system \

      --set global.containerRegistry="$REGISTRY" \
      --set global.storage.eks.fileSystemId=${EFS_ID} \
      --set msdp-operator.image.name="$MSDP_OPERATOR_IMAGE_NAME"
 \
      --set msdp-operator.image.tag="$MSDP_OPERATOR_IMAGE_TAG" \

      --set msdp-operator.storageClass.name=nb-disk-standardssd
\
      --set msdp-operator.storageClass.size=5Gi \
      --set msdp-operator.logging.debug=false \
      --set msdp-operator.logging.age=28 \
      --set msdp-operator.logging.num=20 \
      --set
msdp-operator.nodeSelector."${MSDP_NODE_SELECTOR_KEY//./\\.}"="${MSDP_NODE_SELECTOR_VALUE}"
 \
      --set nb-operator.image.name="$OPERATOR_IMAGE_NAME" \
      --set nb-operator.image.tag="$OPERATOR_IMAGE_TAG" \
      --set nb-operator.loglevel.value="0" \
      --set
nb-operator.nodeSelector.node_selector_key="$MEDIA_NODE_SELECTOR_KEY"
 \
      --set
nb-operator.nodeSelector.node_selector_value="$MEDIA_NODE_SELECTOR_VALUE"
 \
      --set
flexsnap-operator.image.name="$FLEXSNAP_OPERATOR_IMAGE_NAME" \
      --set
flexsnap-operator.image.tag="$FLEXSNAP_OPERATOR_IMAGE_TAG" \
      --set
flexsnap-operator.nodeSelector.node_selector_key="$MEDIA_NODE_SELECTOR_KEY"
 \
      --set
flexsnap-operator.nodeSelector.node_selector_value="$MEDIA_NODE_SELECTOR_VALUE"
```

3. If the reclaim policy of the storage class is **Retain**, run the following command to restart the existing MSDP Scaleout. MSDP Scaleout starts with the existing data/metadata.

   ```
   kubectl apply -f <your-cr-yaml>
   ```

Migrating the MSDP Scaleout to another node pool

You can migrate an existing MSDP Scaleout on another node pool in case of the Kubernetes infrastructure issues.

To migrate the MSDP Scaleout to another node pool

1. Ensure that no job running related to MSDP Scaleout that is going to migrate.
2. Navigate to the helm directory within the extracted tar location.
3. Run the following command to get the Cloud Scale release name:

   ```
   Helm ls -n <namespace>
   ```

4. Run the following command to migrate MSDP Scaleout to new node pool:

   ```
   helm upgrade <release-name> cloudscale-<version>.tgz -n netbackup
   --reuse-values

   --set environment.msdpScaleouts.nodeSelector.labelKey=<new-value>

   --set
   environment.msdpScaleouts.nodeSelector.labelValue=<new-value>
   ```

5. After the helm upgrade, existing pods are terminated and restarted one at a time, and the pods are re-scheduled for the new node pool automatically.

   Note: Controller pods are temporarily unavailable when the MDS pod restarts. Do not delete pods manually.

6. Re run the following command to update the MSDP Scaleout operator with new node pool:

   ```
   # helm upgrade --install operators
   ```

7. If node selector does not match any existing nodes at the time of change, you see the message on the console.

   If auto scaling for node is enabled, it may resolve automatically as the new nodes are made available to the cluster. If invalid node selector is provided, pods may go in the pending state after the update. In that case, run the command above again.

   Do not delete the pods manually.