COHESITY

Version 1.0

June 2024

IBM Security Guardium Key Lifecycle Manager to Manage Cohesity Encryption Keys

ABSTRACT

To ensure the security of your data, both in flight and at rest, Cohesity supports AES-256 software encryption using an internal Key Management Service (KMS) that automatically generates keys and stores them internally. However, where there is a need to have External Key Management System, for reasons like Compliance and Security Best Practices, Centralized policy and Control for encryption keys, Deployment Flexibility and separation of duties between Key and Storage Administrators, users may go with IBM Security Guardium Key Lifecycle Manager. This guide will walk you through the process of setting up IBM Security Guardium Key Lifecycle Manager 4.2.1 as your external KMS in Cohesity.

Table of Contents

Key Management Overview 3
KMIP and Certificate Requirements 3
Prerequisites 4
Encryption and Configuration Considerations 4
Set Up Cohesity-IBM KMS Integration 5
Configure Cohesity Cluster with a Client Certificate 6
Configure Cohesity Cluster with Self-Signed Certificate 6
Configure Cohesity Cluster with CA Signed Certificate 8
Configure IBM Security Guardium Key Lifecycle Manager KMIP 1.4 Settings 11
Create and Extract the IBM GKLM Server Certificate 17
Create and Extract Self-signed Server Certificate 18
Create and Extract CA-signed Server Certificate 22
Configure Cohesity Key Management Settings 26
Configure IBM Security Guardium Key Lifecycle Manager in Cohesity DataPlatform UI 26
Configure IBM Security Guardium Key Lifecycle Manager in Cohesity DataPlatform CLI 29
Update Cohesity Key Management Configuration 30
Update Cohesity KMS in Cohesity DataPlatform UI 30
Update Cohesity KMS in Cohesity DataPlatform CLI 30
Troubleshooting 32
About Errors 33
Troubleshoot Errors 34
Workflow ERRORs when KMS Server is Unreachable 35
Keychain Service 37
Restart the Keychain Service after Key Management Settings Update 37
Your Feedback 38
About the Authors 38
Document Version History 38

Key Management Overview

As organizations work to address a stream of security threats to the data they manage and store, encryption becomes critical to every aspect of their cyber defense strategies. Among its many security features, Cohesity supports using an external key management system (KMS). This guide gives you the information and procedures to configure and integrate Cohesity Data Cloud and IBM Security Guardium Key Lifecycle Manager.

When used with an external key manager like IBM Security Guardium Key Lifecycle Manager, the Cohesity cluster requests that the key manager creates a Key Encryption Key (KEK) for each Storage Domain in the cluster. These KEKs are used to encrypt and decrypt the Data Encryption Keys (DEKs) created and stored locally in the cluster. As the name implies, the DEKs are used to encrypt and decrypt data.

The Cohesity cluster retrieves the KEKs from IBM KMS after the system reboots or the keychain service restarts. If the IBM Security Guardium Key Lifecycle Manager is unavailable, the data in the Storage Domains remains encrypted and inaccessible.

Organizations should carefully evaluate their requirements and consider adopting external KMS for robust key management.

KMIP and Certificate Requirements

The Key Management Interoperability Protocol (KMIP) facilitates communication between the Cohesity cluster and the key server on the IBM Security Guardium Key Lifecycle Manager. KMIP requires Transport Layer Security (TLS) to provide a secure connection. X.509 certificates must exist for the key server and the Cohesity cluster. When you install IBM KMS, the internal Certificate Authority (CA) automatically generates and signs the key server certificate. You must create a client certificate for Cohesity using a tool like OpenSSL. This certificate may be self-signed or externally signed, but this characteristic must be a system-wide characteristic within the IBM KMS environment. In other words, all KMIP clients connecting to IBM KMS must either use self-signed or externally-signed certificates — they cannot use a mixture of both.

Prerequisites

Review the following prerequisites:

- The procedures in this guide assume a basic knowledge of IBM KMS concepts
- Licensing requirements for KMIP functionality
- Cohesity DataPlatform version 6.8.1 or later is installed and operational, and the cluster is configured to use encryption. You can only enable cluster-level encryption when you create the Cohesity cluster. If encryption is not enabled for a Cohesity cluster, you can enable encryption at the Storage Domain level. See the Setup Guide for your specific cluster model in the online Help.

  NOTE: When encryption is at Cluster level or Storage Domain in Cohesity DataPlatform, it cannot be disabled later.

- IBM KMS KMIP version 1.4.

Product/Solution	Feature	Functionality
IBM Security Guardium Key Lifecycle Manager 4.2.1	KMIP1_4	KMS

- IBM KMS is operational and is accessible by the Cohesity cluster on port 5696. For more information, refer Manage Application-Based Firewall Rules.
- Review your organization's security policies governing data-at-rest encryption specifically where they address preventing unauthorized access, regulatory compliance (HIPAA, GDPR, or PCI DSS), and so on.
- Ability to run OpenSSL or some other mechanism for generating a client certificate and private key in the Privacy Enhanced Mail (PEM) format on a Linux machine.

Encryption and Configuration Considerations

The following are some key points to understand regarding this integration:

- Once you enable encryption on a Cohesity cluster, you cannot disable it.
- Once you configure a Cohesity cluster to use an external Key Management System (KMS), you cannot return to using the internal KMS.
- With encryption enabled at the cluster level, all Storage Domains in the cluster are encrypted.
- The Cohesity cluster supports the configuration of one external KMS, and the IP address or FQDN of the KMS cannot be altered once configured.
- Once it establishes a TLS connection with IBM Security Guardium Key Lifecycle Manager, a Cohesity cluster never disconnects that connection. This results in persistent TLS connections.

Set Up Cohesity-IBM KMS Integration

The following tasks are required to integrate Cohesity with IBM Security Guardium Key Lifecycle Manager:

- Use OpenSSL to create a private key and client certificate for the Cohesity cluster.
- Use OpenSSL to extract the internal root CA certificate from IBM KMS.
- Configure IBM KMS to support the integration, including creating a host for the Cohesity cluster. The client certificate will be imported into the host.
- Configure the Cohesity cluster to use IBM KMS as its external Key Management System (KMS KMIP 1.4).

Configure Cohesity Cluster with a Client Certificate

The first step is to create a client certificate and private key for Cohesity that you will use when you configure IBM Security Guardium Key Lifecycle Manager with the Cohesity cluster. You can use OpenSSL or any similar tool to create them, as long as the certificate and key files are in the PEM format and created on a Linux machine.

There are two different types of client certificates:

Self-signed certificates. If your security policy allows for it, you can generate and sign your client certificate.

Externally-signed certificates. If your security policy calls for an external Certificate Authority (CA), use this option. To use it, you must have:

- IBM Security Guardium Key Lifecycle Manager configured to support system-wide use of externally signed certificates.
- Established the external CA as trusted by importing the necessary CA certificates.
- An external, trusted CA that is available to sign the Client Signing Request (CSR).

NOTE: The certificate files must be available on the machine where you plan to log in to Cohesity DataPlatform to configure the KMS settings.

Configure Cohesity Cluster with Self-Signed Certificate

To generate a self-signed certificate and private key for Cohesity:

1. Log in to the Cohesity Data Platform as documented in Using the Cohesity CLI.
2. Use the `genrsa` command to generate the private key that will be written to the key filename and length you specify.

```
$ openssl genrsa -out <key_file_name> <key_length>
```

NOTE: Key lengths less than 2048 bits are not secure.

3. Enter the following OpenSSL command to create the self-signed certificate per your security policy.

```
$ openssl req -new -x509 -key <key_file_name> -out <cert_file_name> -days
<validity_period>
```

4. Enter the requested information when prompted by OpenSSL:
   a. **Country Name**. Your two-letter country code.
   b. **State or Province Name**. The full name of your state.
   c. **Locality Name (eg, city)**. The full name of your city.
   d. **Organization Name**. The name of your organization.
   e. **Organizational Unit Name**. The name of your department.
   f. **Common Name**. The name must match the name of the host created on IBM Security Guardium Key Lifecycle Manager.

g. **Email Address**. (*Optional*) Your email address.
h. **Challenge password**. (*Optional*) Leave this blank; click **Enter**.
i. **Company name**. (*Optional*) Leave this blank; click **Enter**.

5. When the process is complete, you can find the generated file in the current directory.

```
[support@roboisntance ~]$ openssl req -new -x509 -key client_key.key -out
client_cert.crt -days 365
You are about to be asked to enter information that will be incorporated into your
certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []: California
Locality Name (eg, city) [Default City]:San Jose
Organization Name (eg, company) [Default Company Ltd]:Cohesity
Organizational Unit Name (eg, section) []:Solutions
Common Name (eg, your name or your server's hostname) []:roboisntance
Email Address []:
```

6. You must generate the file on a Linux machine, and if you're creating it manually, be careful that the file doesn't contain any extraneous characters.

```
[cohesity@roboisntance ~]$ cat client_cert.crt
-----BEGIN CERTIFICATE-----
MIIDxTCCAq2gAwIBAgIJAOSurkIpu6d4MA0GCSqGSIb3DQEBCwUAMHkxCzAJBgNV
BAYTAIVTMRMwEQYDVQQIDApDYWxpZm9ybmIhMREwDwYDVQQHDAhTYW4gSm9zZTER
MA8GAlUECgwIQ29oZXNpdHkxFDASBgNVBAsMC0VuZ2IuZWVyaW5nMRkwFwYDVQQD
DBBkc20uY29oZXNpdHkuY29tMB4XDTIwMDQxNzE3MDcwNVoXDTIxMDQxNzE3MDcw
NVoweTELMAkGAlUEBhMCVVMxEzARBgNVBAgMCkNhbGImb3JuaWExETAPBgNVBAcM
CFNhbiBKb3NIMREwDwYDVQQKDAhDb2hIc2I0eTEUMBIGAlUECwwLRW5naW5IZXJp
bmcxGTAXBgNVBAMMEGRzbS5jb2h!c2I0eS5jb20wggEiMA0GCSqGSIb3DQEBAQUA
A4IBDwAwggEKAoIBAQDMwMq7EW+ikiKZTyhvR0G2+/lJTy0HrI6bzyo3PRDXC+Mb
67yX3sCZDuAodY5bblXiyRoZrxFjwHi6xu0+mYedPhMFHSdo9CPQF2gx2GfWZtEB
FH0qZFZpXug3evqz2IOKBj8SLtFto4uKxSE9EfkMM/IezcP8JVsW7cERohuqc38V
BuLU7LH52vKH+oY0P5v0vpp8IPMJ2zB5vVSh97NTiF/Yo4sWBxeLBQfGmQjxLqkp
zSKGt7T3cle5d4AgPgITzLs6ia+XGQHvoZyIuYnD797hgoJfJZ2R82b8zJRms7IG
et8TXn!cic6+067It2z8fSveyBfRnDyphyWvKhpnAgMBAAGjUDBOMBOGAlUdDgQW
BBS74xB8ZVcniPRZgErlx601kO9+8jAfBgNVHSMEGDAWgBS74xB8ZVcniPRZgEr1
x601kO9+8jAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQByOZndS7Y8
tG2a2HAzPZB6p9DBDvpNGtT+kpXSjh5uoU4PdB2//k3zGr9UIE8TYBFbLlTv!tMn
908EVcmlbSYjnK+4k9LWOd0PWsLn4a9mFiTvIibatQ3CkjYZAv8jg2y6GI01s6wC
xvWun70jkTxjf2S4bT0ioI+jPp/p0d9oafa+AWAcIMieEm2Gh7n75dZRWQ6Ne7XI
xKk61kASKBiurnzRSvgaCBTTcfTG0fVhZWE7FFAf0C2vAMEFgiI8H65Lgw9f4Lmo
dngrIrD3JQms93QymIeyoF+wwrDsl0nOxONrgJ3oAEH+VdW37YSCR3WeTq46FW2e
5wBVA5TJIr9z
-----END CERTIFICATE-----
```

7. Replace the Cohesity cluster's current SSL certificate with the new self-signed certificate you created in the previous steps:

   a. Start the Cohesity CLI using the following command:

   ```
   cohesity@roboinstance ~]$ iris_cli
   ```

   b. When prompted, enter the Username and password you use to log into the Cohesity Cluster's User Interface. Once the password is successfully authenticated, the Cohesity CLI console opens.

   Replace the certificate by running the following command:

   ```
   cluster update-ssl-certificate ssl-certificate=<absolute path of the cert.pem
   file> ssl-cert-private-key=<absolute path of the key.pem file>

   Example:
   admin@127.0.0.1> cluster update-ssl-certificate ssl-
   certificate=/home/support/client_cert.pem ssl-cert-private-
   key=/home/support/client_key.pem
   ```

   c. Restart the UI and REST API Services using the following command:

   ```
   admin@127.0.0.1> cluster restart service-names=iris
   ```

   d. Wait for a minute before proceeding with the next command.

   ```
   admin@127.0.0.1> cluster restart service-names=bridge
   ```

   e. Restart the I/O Operations service using the following command:

Reference: Configuring Cohesity Cluster with a Self-Signed Certificate

Configure Cohesity Cluster with CA Signed Certificate

If you choose to use an externally signed certificate instead of a self-signed certificate, you need to do so with an external and trusted Certificate Authority.

To generate an externally signed certificate and private key:

1. Log in to the Cohesity Data Platform as documented in Using the Cohesity CLI.
2. Use the `genrsa` command to generate the private key that will be written to the key filename and length you specify. Create the key per your organization's security policy.

   ```
   $ openssl genrsa -out <key_file_name> <key_length>
   ```

3. Enter the following OpenSSL command to initiate the CSR file generation process.

   ```
   $ openssl req -new -key <key_file-name> -out <csr_file_name>
   ```

4. Enter the requested information as prompted by OpenSSL:
   - **Country Name**. Your two-letter country code.
   - **State or Province Name**. The full name of your state.
   - **City Name**. The full name of your city.
   - **Organization Name**. The name of your organization.
   - **Organizational Unit Name**. The name of your department.
   - **Common Name**. The name must match the host's name, created on IBM Security Guardium Key Lifecycle Manager.
   - **Email**. (*Optional*) Your email address.
   - **Challenge password**. (*Optional*) Leave this blank; click **Enter**.
   - **Company name**. (*Optional*) Leave this blank; click **Enter**.
5. You can find the generated CSR file in the current directory when the process is complete.

```
[cohesity@roboinstance ~]$ openssl req -new -key cohesity241.key -out
cohesity241.csr
You are about to be asked to enter information that will be incorporated into your
certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []: California
Locality Name (eg, city) [Default City]:San Jose
Organization Name (eg, company) [Default Company Ltd]:Cohesity
Organizational Unit Name (eg, section) []:Engineering
Common Name (eg, your name or your server's hostname) []:dsm.cohesity.com
Email Address []:
```

6. Have a trusted CA sign the CSR and download or create a file that contains the certificate.

```
[cohesity@roboinstance ~]$ vi IBMKMS_root.crt
[cohesity@roboinstance ~]$ cat IBMKMS_root.crt
-----BEGIN CERTIFICATE-----
MIIENjCCAx6gAwIBAgIGALOmLIL9MA0GCSqGSIb3DQEBDAUAMIGjMSQwIgYDVQQD
ExtDRyBDQSBTIG9uIGRzbS5jb2hlc2l0eS5jb20xETAPBgNVBAsTCGNvaGVzaXR5
MREwDwYDVQQKEwhjb2hlc2l0eTEQMA4GAlUEBxMHU2FuSm9zZTELMAkGAlUECBMC
Q0ExKTAnBgkqhkiG9w0BCQEWGmZlcWlhbmcuemhhbmdAY29oZXNpdHkuY29tMQsw
CQYDVQQGEwJVUzAeFw0yMDA0MTUyMzQ5MThaFw0zMDA0MTcyMzQ5MThaMIGjMSQw
IgYDVQQDExtDRyBDQSBTIG9uIGRzbS5jb2hlc2l0eS5jb20xETAPBgNVBAsTCGNv
aGVzaXR5MREwDwYDVQQKEwhjb2hlc2l0eTEQMA4GAlUEBxMHU2FuSm9zZTELMAkG
AlUECBMCQ0ExKTAnBgkqhkiG9w0BCQEWGmZlcWlhbmcuemhhbmdAY29oZXNpdHku
Y29tMQswCQYDVQQGEwJVUzCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
AL5BAxvndW/VoHfpMQPFqw9nzvxddT3yfypbl07TZOeby7C2XHhuIneUzPPr7m/v
C8cTS0eqj75cj0SVwVkMzKiU2nHXxX3qAztnkHhlo5uryWsQk2Kls0fB3GecvjU2
5oDo9NOJqDoU29wKhjgZMAMv/SPIHYu1IfWT0CV5FfjtsdsDYINHiLdUxZIQJF42
r0aPQT0weiTG0V9bJj/ilA2hCUTyVVwXc4erEsDbeT9Sc7kKJ3QEGcBMcVtit5zM
AK5HhMq8T+oS9dhIyWBZpiAVp/rYE2ZfIU2FeplHJtf60zCpIgHr7N12lqviTbsK
svhyLt20teadRitSRlXGweMCAwEAAaNuMGwwEgYDVR0TAQH/BAgwBgEB/wIBADAO
BgNVHQ8BAf8EBAMCAQYwHQYDVR0OBBYEFKgZMHRJBV1MRLi+xHdp0vrjsQSwMCcG
AlUdIwQgMB6AFKgZMHRJBV1MRLi+xHdp0vrjsQSwggYAs6Ysgv0wDQYJKoZIhvcN
AQEMBQADggEBAEsQ29HfaHpHUFBj4Tn74XW0b9w9Pmk2u5AM13d9CU/oLYcMJn0L
tfavpnMgKPuDwQ/3gqgFDXiq40dq73fD+38A2wlcB/xZumQ6nGGuYGk2p2VVy6Cr
JlVD0pL4iwidQXMYJ7xDmL+TwHM8clwPiwxqI3HRNRQokqwl489VeptNMteInllM
IY28TZOn/T89OxUDxoqxIlXEu672qZq7kG/YpeeGG+0uwH8QZnxjogaIQPulTxJs
5tzcsFM7nRjE63Z14GUIsQElpXikSu9G8ljpGBmkfcJdaVIHSeRTB7NV5ZIj2NvA
UXm/RFvjTxh9blR5yTkdPUDEDgFdJWVA6Fw=
-----END CERTIFICATE-----
```

7. Replace the Cohesity cluster's current SSL certificate with the new self-signed certificate you created in the previous steps:

   a. Start the Cohesity CLI using the following command:

   ```
   [cohesity@roboinstance ~]$ iris_cli
   ```

   b. When prompted, enter the Username and password to log into the Cohesity Cluster's User Interface. Once the password is successfully authenticated, the Cohesity CLI console opens. Replace the certificate by running the following command:

   ```
   cluster update-ssl-certificate ssl-certificate=<absolute path of the cert.pem
   file> ssl-cert-private-key=<absolute path of the key.pem file>
   Example:
   admin@127.0.0.1> cluster update-ssl-certificate ssl-
   certificate=/home/support/cert.pem ssl-cert-private-key=/home/support/key.pem
   ```

   c. Restart the UI and REST API Services using the following command:

   ```
   admin@127.0.0.1> cluster restart service-names=iris
   ```

   d. It is recommended to wait a minute before proceeding with the following command.

   e. Restart the I/O Operations service using the following command:

   ```
   admin@127.0.0.1> cluster restart service-names=bridge
   ```

Reference: [Configuring Cohesity Cluster with a CA-Signed Certificate]

Configure IBM Security Guardium Key Lifecycle Manager KMIP 1.4 Settings

Equipped with your client certificate, private key, and the IBM Security Guardium Key Lifecycle Manager, either self-signed or CA-signed certificate, you are ready to configure the IBM Security Guardium Key Lifecycle Manager settings to support Cohesity for a Key Management Interoperability Protocol (KMIP) integration.

You can install IBM Security Guardium Key Lifecycle Manager on distributed platforms with IBM Security Guardium Key Lifecycle Manager traditional or on container platforms with IBM Security Guardium Key Lifecycle Manager container.

This Solutions Guide recommends installing via the Traditional method utilizing Silent mode.

In this example, a RHEL 8.4 Linux Server was chosen to install and configure the IBM KMS.

1. Download the software onto the server.

```
[root@lnxhost ~]# ls -l gklm_v421_linux_*
-rw-r--r--. 1 root root 4944673292 Mar  4 02:04 gklm_v421_linux_1.tar.gz
-rw-r--r--. 1 root root 1787450138 Mar  4 01:52 gklm_v421_linux_2.tar.gz
```

2. Extract the files, as shown below:

```
[root@lnxhost ~]# tar -xvf gklm_v421_linux_1.tar.gz
[root@lnxhost ~]# tar -xvf gklm_v421_linux_2.tar.gz

[root@lnxhost ~]# ls -l
drwxr-xr-x. 11 root root 4096 Mar 11 03:26 disk1
drwxr-xr-x.  3 root root   35 Nov 29 03:03 disk2
```

3. To perform a fresh installation, edit and update all the user inputs, such as the repository location where installation binaries are present, installation directory and user credentials, in the input response file **SKLM_Silent_Linux_Resp.xml**.

   To do so, switch to the 'disk1' folder, use any text editor (vi, for example), and edit as shown below. Before that, create the following users and groups and associate them appropriately as shown below:

```
[root@lnxhost ~]# groupadd -g 501 klmdb421
[root@lnxhost ~]# useradd -u 501 -g klmdb421 klmdb421
[root@lnxhost ~]# passwd klmdb421
New password:
Retype new password:

[root@lnxhost ~]# groupadd -g 502 SKLMAdmin
[root@lnxhost ~]# useradd -u 502 -g SKLMAdmin SKLMAdmin
[root@lnxhost ~]# passwd SKLMAdmin
New password:
Retype new password:
```

NOTE: SKLM admin password expects at least two numerals.

4. Ensure the **umask** is set to 022, as shown below:

```
[root@lnxhost disk1]# umask
0022
```

5. Disable SELinux and ensure the same, as shown below:

```
[root@lnxhost disk1]# vi /etc/selinux/config
      1
      2 # This file controls the state of SELinux on the system.
      3 # SELINUX= can take one of these three values:
      4 #     enforcing - SELinux security policy is enforced.
      5 #     permissive - SELinux prints warnings instead of enforcing.
      6 #     disabled - No SELinux policy is loaded.
      7 SELINUX=disabled
      8 # SELINUXTYPE= can take one of these three values:
      9 #     targeted - Targeted processes are protected,
     10 #     minimum - Modification of targeted policy. Only selected
   processes are protected.
     11 #     mls - Multi Level Security protection.
     12 SELINUXTYPE=targeted

[root@lnxhost ~]# reboot
```

Wait for the host to come back with up and running state

Check if SELinux is disabled, as shown below:

```
[root@lnxhost ~]# getenforce
Disabled

[root@lnxhost ~]# sestatus
SELinux status:                 disabled
```

6. Permissions on the /tmp directory of the system are set to 777, that is, full execute, read, and write permissions, as shown below:

```
[root@lnxhost ~]# chmod 777 /tmp
```

7. Run the **preReqCheck.sh** script to validate if all prerequisites are met before launching the install script, as shown below:

```
[root@lnxhost precheckscripts]# ./disk1/precheckscripts/preReqCheck.sh
OS: Linux
which: no preReqCheckLinux.sh in
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin)
Thu Mar 28 04:22:00 MDT 2024 - SKLM Prerequisite check started.
Db2 prerequisite check - PASSED.
Checking required shell - PASSED
Checking executable permissions - PASSED
Checking required umask - PASSED
Checking minimum required space - PASSED
Checking kernel parameters - PASSED
Checking SELinux - PASSED
Checking CPU speed - PASSED
Checking RAM - PASSED
```

```
Thu Mar 28 04:22:01 MDT 2024 - SKLM Prerequisite check - PASSED .
check /tmp/SKLMPrereqCheck.log for more details.
```

8. Updating the **SKLM_Silent_Linux_Resp.xml**. The contents that need modification are highlighted in blue and with an appropriate explanation.

```
[root@lnxhost disk1]# vi SKLM_Silent_Linux_Resp.xml
<?xml version='1.0' encoding='UTF-8'?>
<agent-input  clean='true' >
  <server>
```

a. Ensure the directory paths are correct.

```
 <repository location='/root/disk1/im'/>
 <repository location='/root/disk1/'/>
 </server>
```

b. Provide Custom or retain default and ensure path exists.

```
<profile id='IBM Installation Manager'
installLocation='/opt/IBM/InstallationManager/eclipse' kind='self'>
    <data key='eclipseLocation'
value='/opt/IBM/InstallationManager/eclipse'/>
    <data key='user.import.profile' value='false'/>
    <data key='cic.selector.os' value='linux'/>
    <data key='cic.selector.arch' value='x86_64'/>
    <data key='cic.selector.ws' value='gtk'/>
  </profile>
  <install modify='false'>
    <offering profile='IBM Installation Manager' id='com.ibm.cic.agent'
features='agent_core,agent_jre,agent_web' installFixes='none'/>
        <offering profile='IBM Db2 11.5.9.0'
id='com.ibm.gklm421.db2.lin.ofng' features='main.feature'
installFixes='none'/>
        <offering profile='WebSphere Liberty'
id='com.ibm.websphere.liberty.BASE' features='liberty'
installFixes='none'/>
        <offering profile='WebSphere Liberty' id='com.ibm.java.jdk.v8'
features='com.ibm.sdk.8' installFixes='none'/>
        <offering profile='IBM Security Guardium Key Lifecycle Manager
v4.2.1' id='com.ibm.gklm421.linux' features='main.feature'
installFixes='none'/>
  </install>
```

c. Provide Custom or retain default and ensure path exists.

```
<profile id='IBM Db2 11.5.9.0' installLocation='/opt/IBM/DB2GKLMV421'>
 <data key='eclipseLocation' value='/opt/IBM/DB2GKLMV421'/>
 <data key='user.import.profile' value='false'/>
 <data key='cic.selector.os' value='linux'/>
 <data key='cic.selector.arch' value='x86_64'/>
 <data key='cic.selector.ws' value='gtk'/>
```

d. Provide Custom user or retain default and ensure a user is created and functional on the host.

```
    <data key='user.DB2_ADMIN_ID,com.ibm.gklm421.db2.lin.ofng'
value='klmdb421'/>
    <data key='user.DB2_INSTANCE,com.ibm.gklm421.db2.lin.ofng'
value='klmdb421'/>
```

e. Use command './disk1/im/tools/imcl encryptString <password string>' to encrypt the password and provide here.

```
    <data key='user.DB2_ADMIN_PWD,com.ibm.gklm421.db2.lin.ofng'
value='QTh/0AiFvrljhs9gnOYkGA=='/>
    <data key='user.CONFIRM_PASSWORD,com.ibm.gklm421.db2.lin.ofng'
value='QTh/0AiFvrljhs9gnOYkGA=='/>

    <data key='user.DB2_DB_HOME,com.ibm.gklm421.db2.lin.ofng'
value='/home/klmdb421'/>
    <data key='user.DB2_DB_NAME,com.ibm.gklm421.db2.lin.ofng'
value='KLMDB421'/>
    <data key='user.DB2_DB_PORT,com.ibm.gklm421.db2.lin.ofng'
value='50110'/>
    <data key='user.DB2_EXISTS,com.ibm.gklm421.db2.lin.ofng'
value='false'/>
    <data key='user.DB2_LOCATION,com.ibm.gklm421.db2.lin.ofng'
value='/opt/IBM/DB2GKLMV421'/>
    <data key='user.DB2_DB_LHOME,com.ibm.gklm421.db2.lin.ofng'
value='/home/klmdb421'/>
    <!--The DB2 Group name should not be more than 8 characters -->
    <data key='user.DB2_ADMIN_GRP,com.ibm.gklm421.db2.lin.ofng'
value='klmdb421'/>
    <data key='cic.selector.nl' value='en'/>
  </profile>
```

f. Provide Custom or retain default and ensure path exists.

```
<profile id='WebSphere Liberty'
installLocation='/opt/IBM/WebSphere/Liberty'>
    <data key='eclipseLocation' value='/opt/IBM/WebSphere/Liberty'/>

    <data key='user.import.profile' value='false'/>
    <data key='cic.selector.os' value='linux'/>
    <data key='cic.selector.arch' value='x86_64'/>
    <data key='cic.selector.ws' value='gtk'/>
    <data key='cic.selector.nl' value='en'/>
  </profile>
```

g. Provide Custom or retain default and ensure path exists.

```
<profile id='IBM Security Guardium Key Lifecycle Manager v4.2.1'
installLocation='/opt/IBM/GKLMV421'>
    <data key='eclipseLocation' value='/opt/IBM/GKLMV421'/>
    <data key='user.import.profile' value='false'/>
    <data key='cic.selector.os' value='linux'/>
    <data key='cic.selector.arch' value='x86_64'/>
    <data key='cic.selector.ws' value='gtk'/>
    <data key='user.IS_SILENT_MODE,com.ibm.gklm421.linux'
value='false'/>
```

h. Provide Custom user or retain default and ensure a user is created and functional on the host.

```
<data key='user.SKLM_ADMIN_USER,com.ibm.gklm421.linux'
value='SKLMAdmin'/>
```

i. Use command './disk1/im/tools/imcl encryptString <password string>' to encrypt the password and provide here

```
    <data key='user.SKLM_ADMIN_PASSWORD,com.ibm.gklm421.linux'
value='9YTRJMRIydDSdfhaHPs1ag=='/>
    <data key='user.SKLM_ADMIN_CONF_PWD,com.ibm.gklm421.linux'
value='9YTRJMRIydDSdfhaHPs1ag=='/>

    <data key='user.SKLM_APP_PORT,com.ibm.gklm421.linux' value='9443'/>
    <data key='cic.selector.nl' value='en'/>
  </profile>
```

j. Provide Custom or retain default and ensure path exists.

```
<preference name='com.ibm.cic.common.core.preferences.eclipseCache'
value='/opt/IBM/IBMIMShared'/>
<preference name='com.ibm.cic.common.core.preferences.connectTimeout'
value='30'/>
<preference name='com.ibm.cic.common.core.preferences.readTimeout'
value='45'/>
<preference
name='com.ibm.cic.common.core.preferences.downloadAutoRetryCount'
value='0'/>
<preference name='offering.service.repositories.areUsed' value='true'/>
<preference
name='com.ibm.cic.common.core.preferences.ssl.nonsecureMode'
value='false'/>
<preference
name='com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthent
ication' value='false'/>
<preference name='http.ntlm.auth.kind' value='NTLM'/>
<preference name='http.ntlm.auth.enableIntegrated.win32' value='true'/>
<preference
name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts'
value='true'/>
<preference name='com.ibm.cic.common.core.preferences.keepFetchedFiles'
value='false'/>
<preference name='PassportAdvantageIsEnabled' value='false'/>
<preference name='com.ibm.cic.common.core.preferences.searchForUpdates'
value='false'/>
<preference name='com.ibm.cic.agent.ui.displayInternalVersion'
value='false'/>
<preference name='com.ibm.cic.common.sharedUI.showErrorLog'
value='true'/>
<preference name='com.ibm.cic.common.sharedUI.showWarningLog'
value='true'/>
<preference name='com.ibm.cic.common.sharedUI.showNoteLog'
value='true'/>
</agent-input>
```

9. Save the file and exit. Use the below silent install script and the above-edited .xml by accepting the license.

```
[root@lnxhost disk1]# ./silent_install.sh SKLM_Silent_Linux_Resp.xml -
acceptLicense
No pre installed IBM Installation Manager found on the system.
Installing IBM Security Guardium Key Lifecycle Manager v4.2.1.0
Mon Mar 4 23:56:50 PST 2024 - SKLM Prerequisite check started.
Db2 prerequisite check - PASSED.
Checking required shell - PASSED
Checking executable permissions - PASSED
Checking required umask - PASSED
Checking minimum required space - PASSED
Checking kernel parameters - PASSED
Checking SELinux - PASSED
Checking CPU speed - PASSED
Checking RAM - PASSED
Mon Mar 4 23:56:51 PST 2024 - SKLM Prerequisite check - PASSED .
check /tmp/SKLMPrereqCheck.log for more details.
Installed com.ibm.cic.agent_1.9.2005.20230718_1844 to the
/opt/IBM/InstallationManager/eclipse directory.
Installed com.ibm.gklm421.db2.lin.ofng_11.5.9.0 to the
/opt/IBM/DB2GKLMV421 directory.
Installed com.ibm.websphere.liberty.BASE_23.0.9.20230904_1159 to the
/opt/IBM/WebSphere/Liberty directory.
Installed com.ibm.java.jdk.v8_8.0.8015.20231031_0036 to the
/opt/IBM/WebSphere/Liberty directory.
Installed com.ibm.gklm421.linux_4.2.1000.0 to the /opt/IBM/GKLMV421
directory.
WARNING: Support for using Java SE 7 and Java SE 7.1 with WebSphere
Liberty ended. The Liberty kernel can no longer run with Java SE 7 or Java
SE 7.1.
Installation process is complete. Please look into Installation Manager
logs for details.
```

IBM Security Guardium Key Lifecycle Manager is now installed SUCCESSFULLY to support KMIP communication with the Cohesity cluster. You are ready to configure the KMS settings in Cohesity DataPlatform.

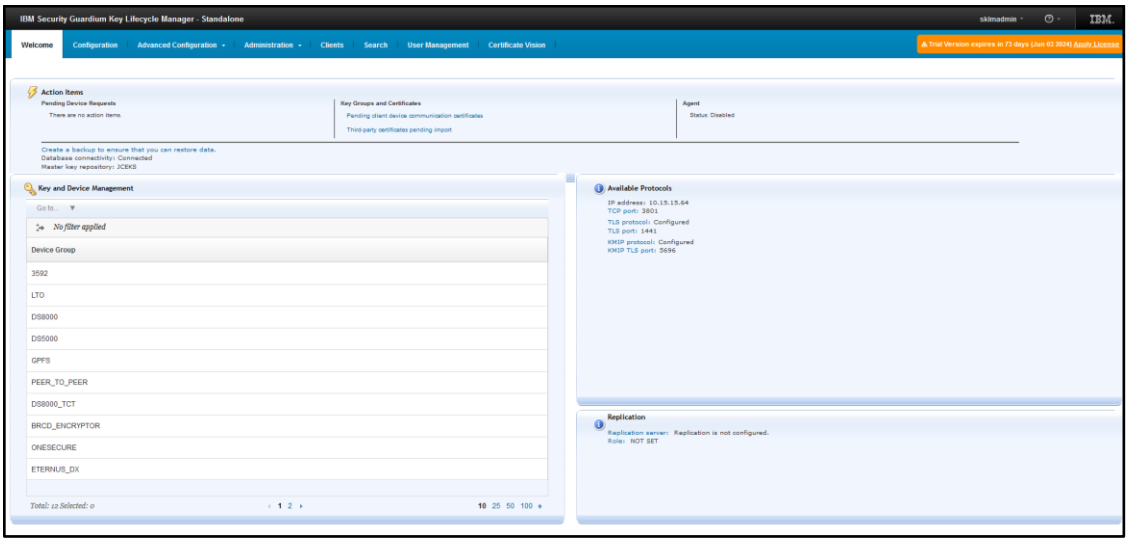
Create and Extract the IBM GKLM Server Certificate

Once you have generated the client certificate and private key, you must extract the IBM Security Guardium Key Lifecycle Manager certificate for import into the Cohesity cluster. You can extract it by logging into the IBM KMS GUI.

To extract the IBM KMS internal root CA certificate:

Log in to IBM Security Guardium Key Lifecycle Manager Web UI https://<IBM KMS Server address>:9443/ibm/SKLM/login.jsp

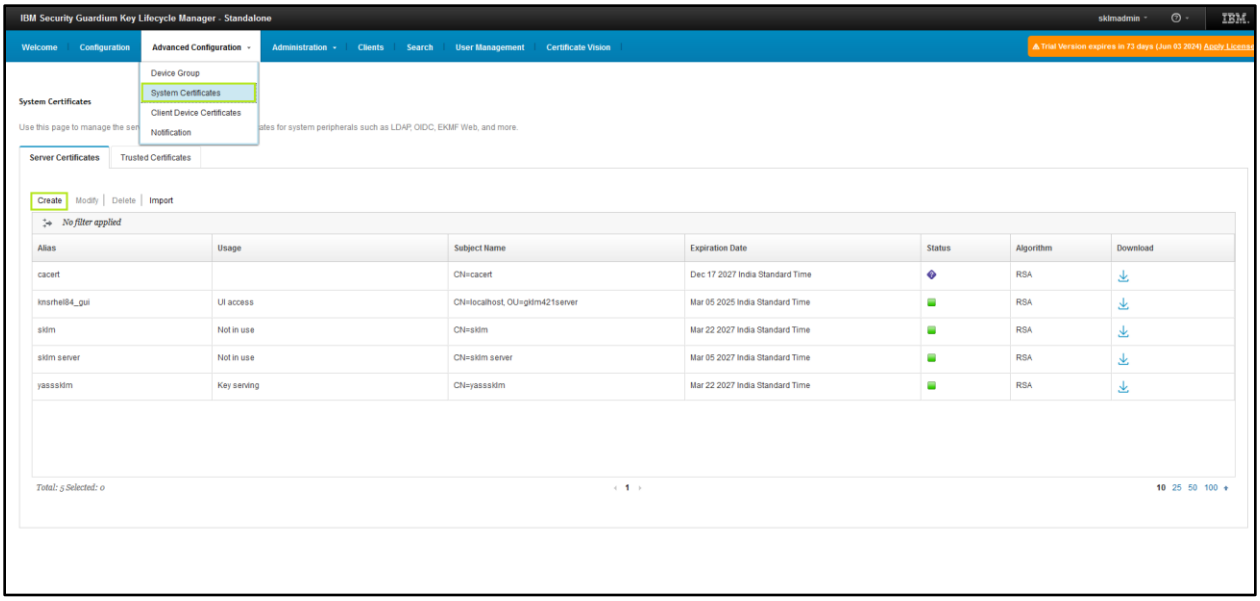
Once Logged in, the landing page will be as below:



Create and Extract Self-signed Server Certificate

To create a self-signed server certificate, complete the following steps:

1. Click **Advanced Configuration** > **System Certificates** and then click **Create**, as shown below:

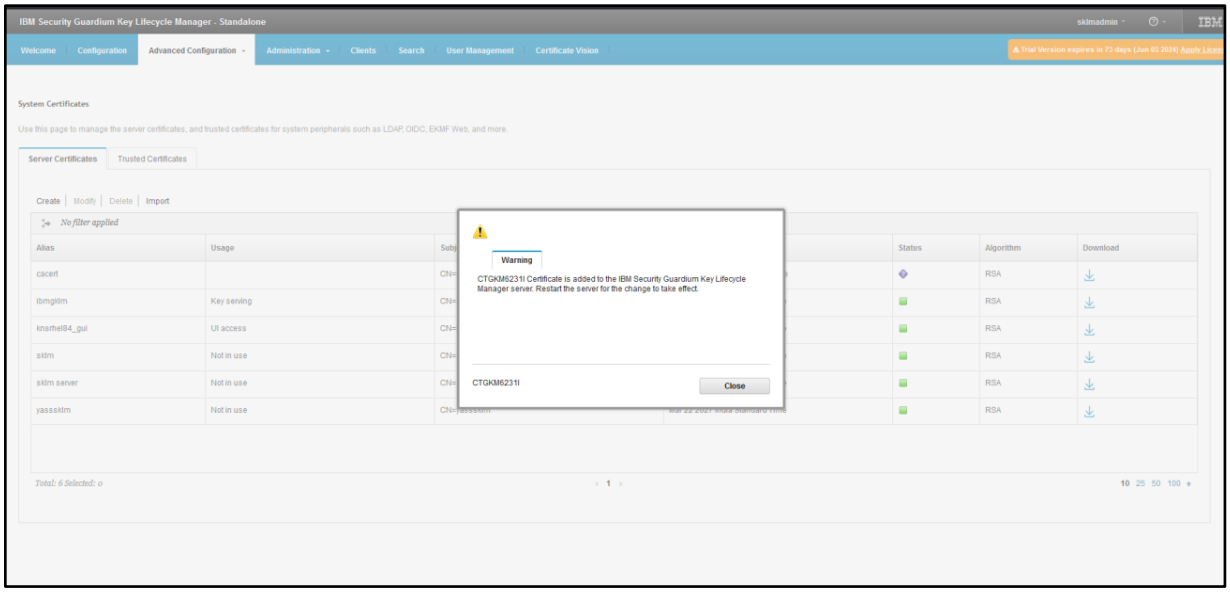


2. Select the Create a self-signed certificate option and complete the details, as shown below.

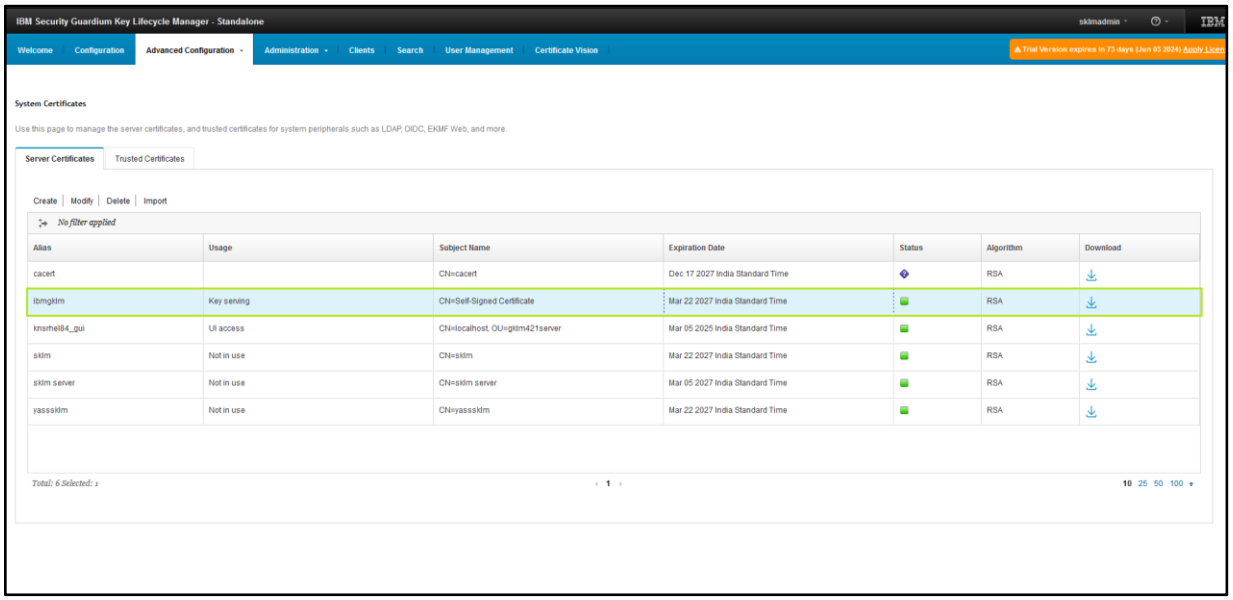
   The validity period determines how long the certificate is valid. By default, IBM Security Guardium Key Lifecycle Manager creates a 2048-bit RSA public-private key pair for server certificates with a validity of 3 years.

3. After all details are completed, click **Create Certificate**.

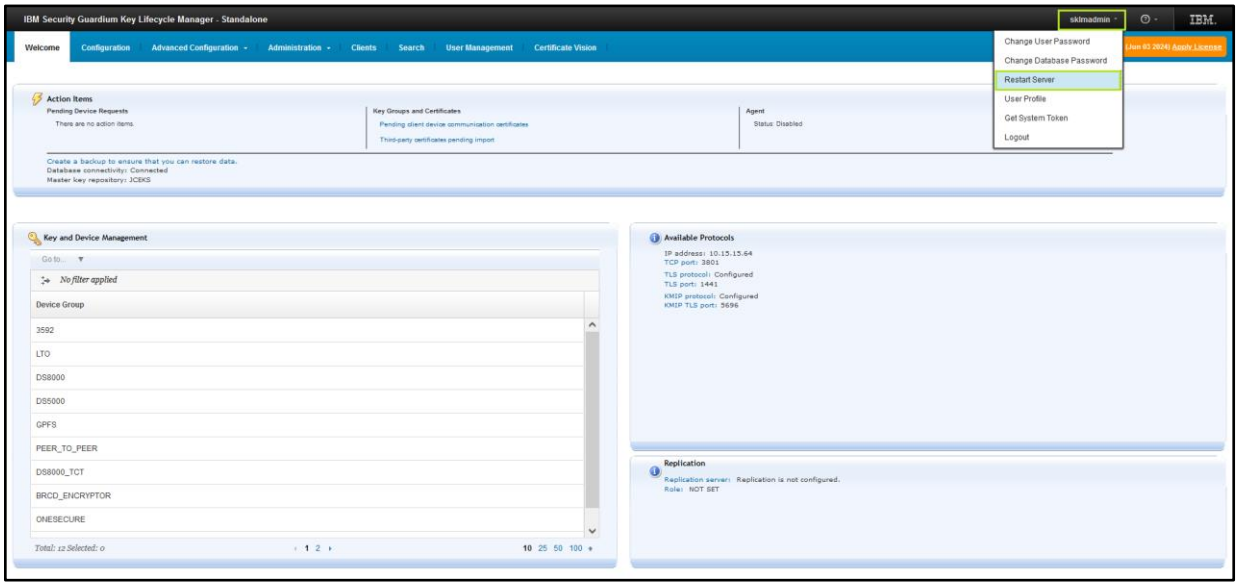
4. A message confirms that the certificate was successfully created, as shown below. Click **Close**.



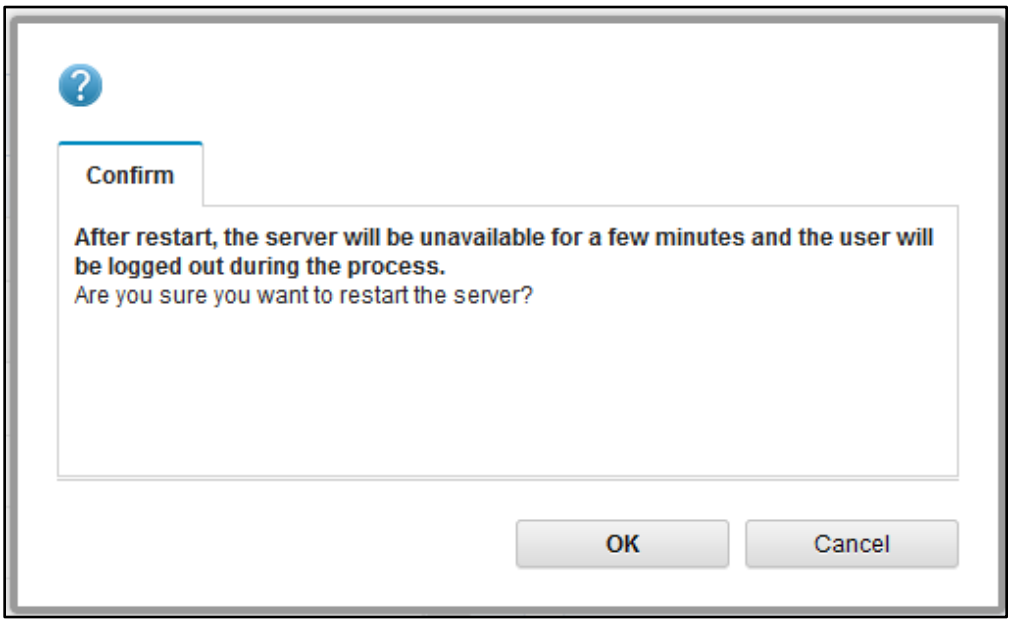
5. Verify the details of the new server certificate, as shown below:



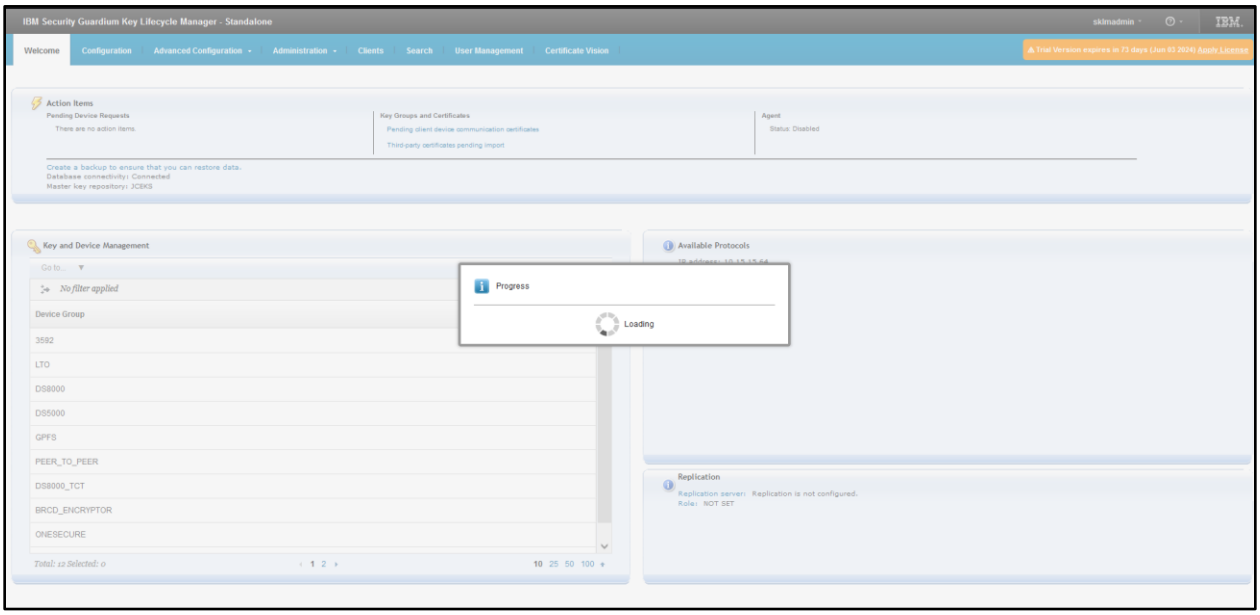
6. Restart the IBM Security Guardium Key Lifecycle Manager by selecting the logged-in user in the upper right corner. Click Restart Server, as shown below:



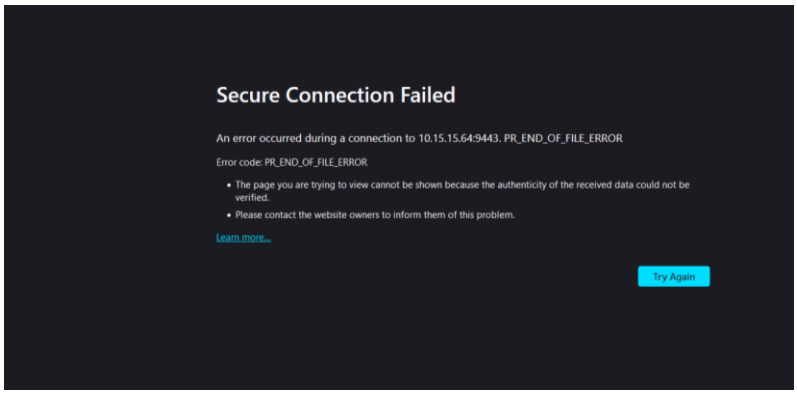
7. Click **OK** to confirm the restart of the IBM KMS server.



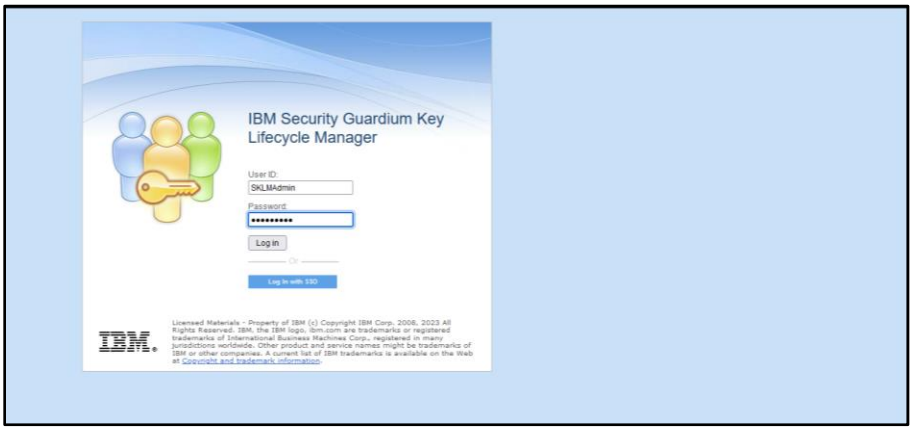
8. Monitor until the IBM Server comes up, as this may take a moment.



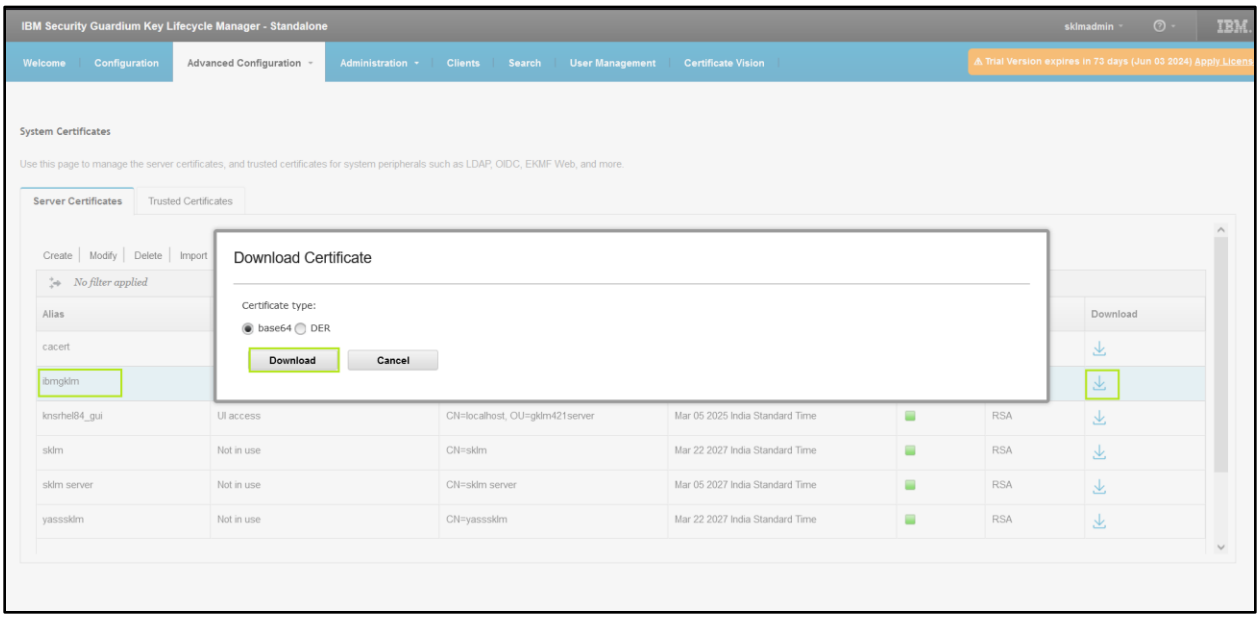
9. Observe that the web UI connection is lost, and the IBM Server comes up.



10. Observe the web UI connection resumes as IBM KMS comes up.



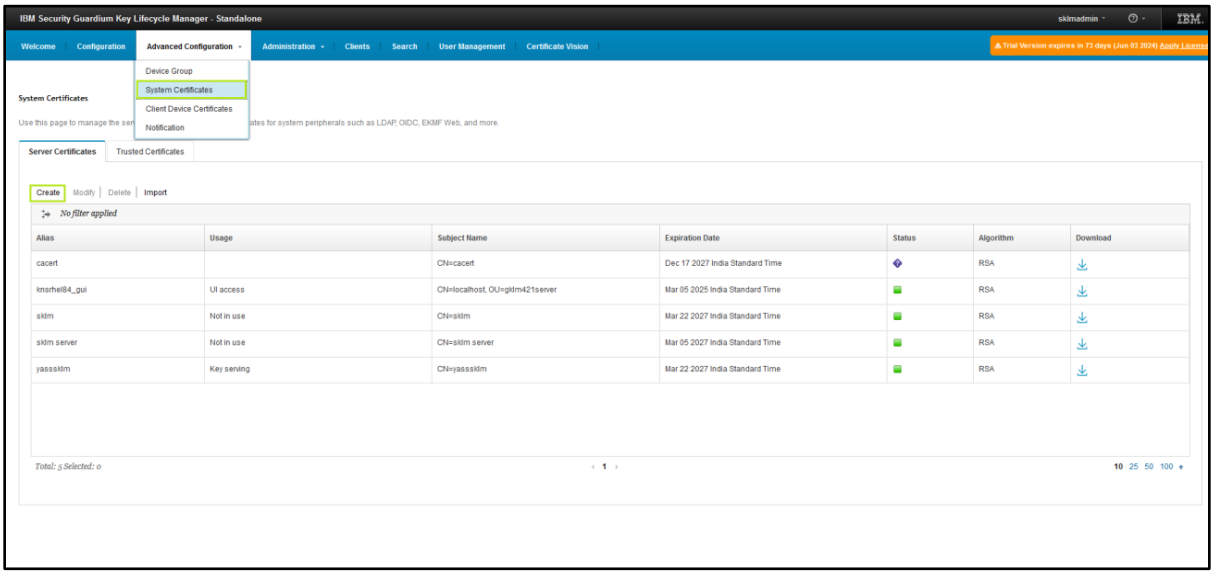
11. Download the self-signed certificate and keep it for the next steps detailed in the section below.



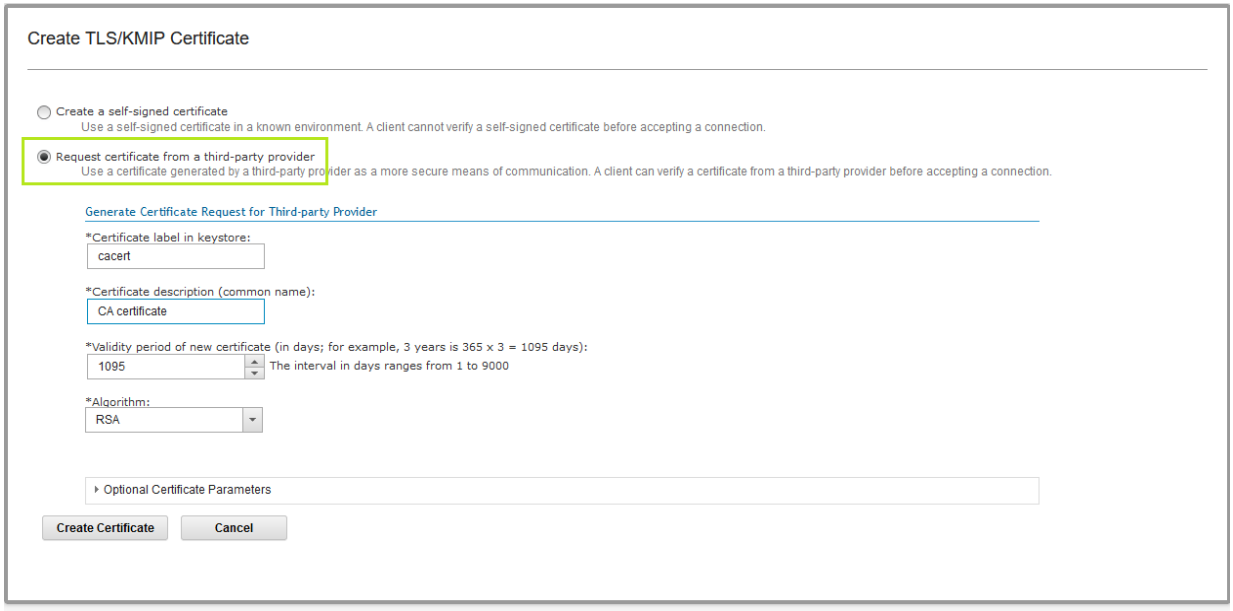
Create and Extract CA-signed Server Certificate

To create a certificate request for a signed CA server certificate, complete the following steps:

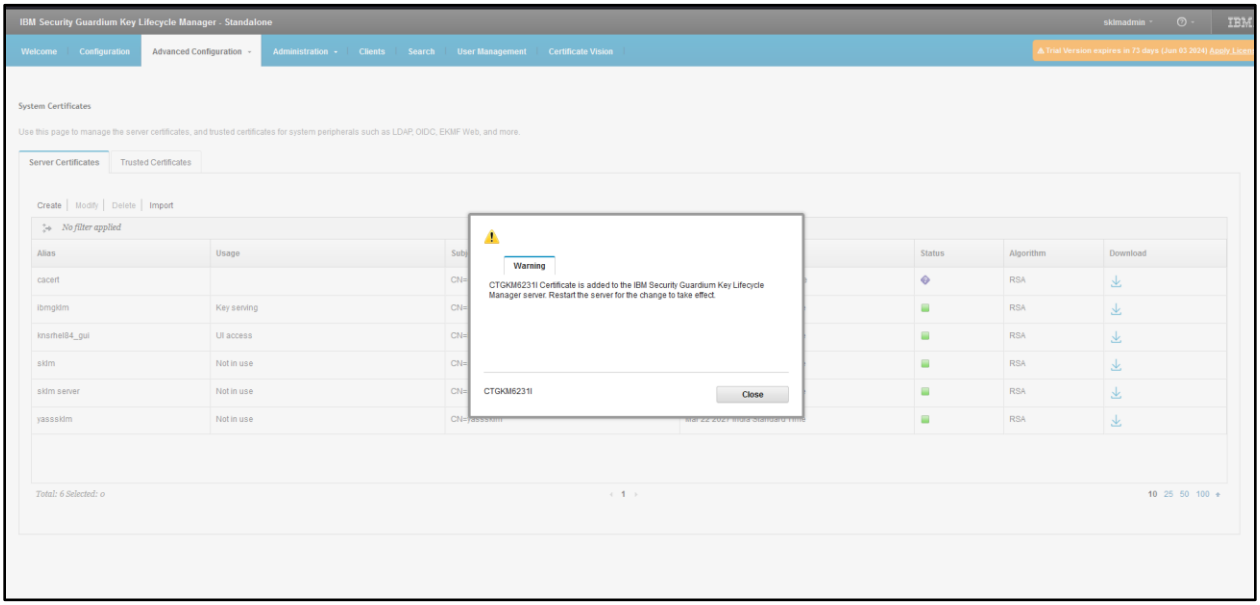
1. Click **Advanced Configuration** > **System Certificates** and then click **Create**.



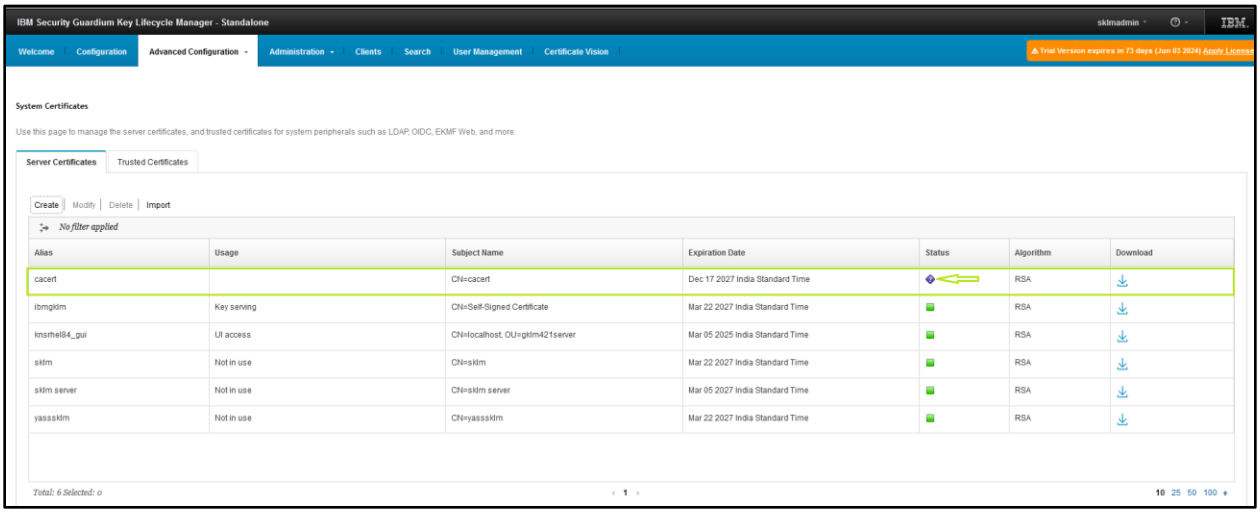
2. Select 'Request certificate from a third-party provider,' specify the certificate details and validity period, and then click **Create Certificate**.



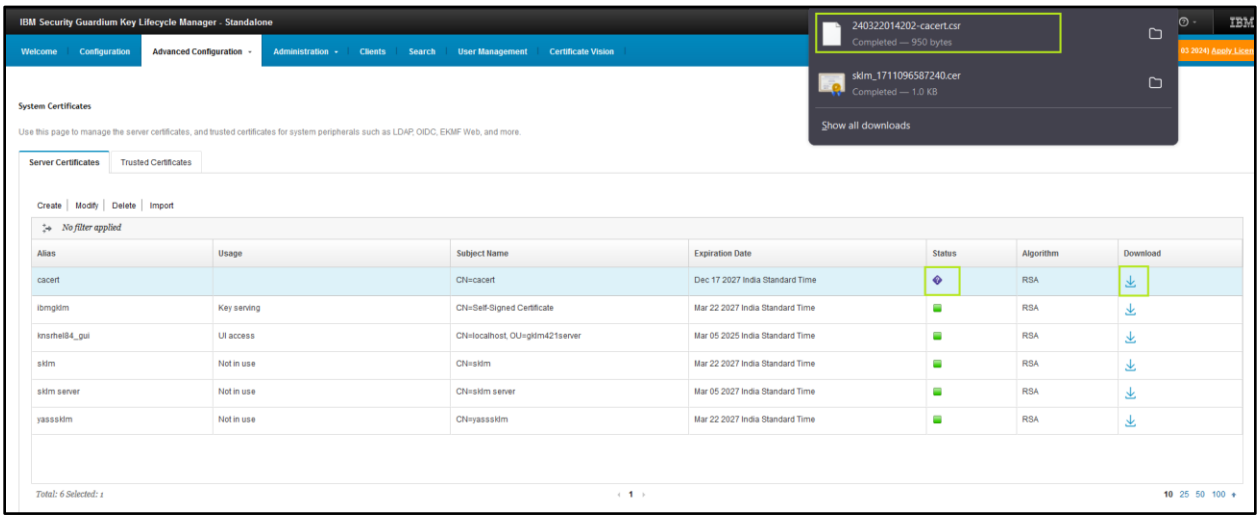
3. A message that confirms the successful certificate signing request is displayed, as shown below. Click **Close**.



4. Validate the status of the created certificate, which is Pending, as shown below:



5. You can download the CSR file from the IBM Security Guardium Key Lifecycle Manager GUI by clicking the link below.



6. Get the certificate signing request file signed by the trusted CA.
7. Go to the Welcome page in the IBM Security Guardium Key Lifecycle Manager GUI and check the Action Items.
8. Click Third-party certificates pending import.
9. On the Import page, select the Pending certificate and click Import.
10. Click Browse, select the signed certificate, click Select, and then click Import.
11. After importing the signed certificate, the status of the server certificate changes to Valid. Purple diamond shape with question mark icon to opaque/solid Green square icon.
12. Restart the IBM Security Guardium Key Lifecycle Manager Server.

To establish TLS communication, you need to download the IBM Security Guardium Key Lifecycle Manager Server certificate to get it trusted on the client side.

Complete the following steps to export and download the TLS/KMIP Server certificate:

- Log in to IBM Security Guardium Key Lifecycle Manager GUI and browse to Advanced
- Configuration > Server certificates page.
- Select the server certificate, which is marked in Use.
- Click the download icon.
- Click Download to download the exported certificate on your local machine for the next steps detailed in the section.

Configure Cohesity Key Management Settings

The final step is configuring the KMS settings on the Cohesity cluster. Once the configuration has been saved, the cluster establishes a TLS session with IBM Security Guardium Key Lifecycle Manager. After a few minutes of internal processing, it requests that keys be created for internal use and the existing Default Storage Domain.

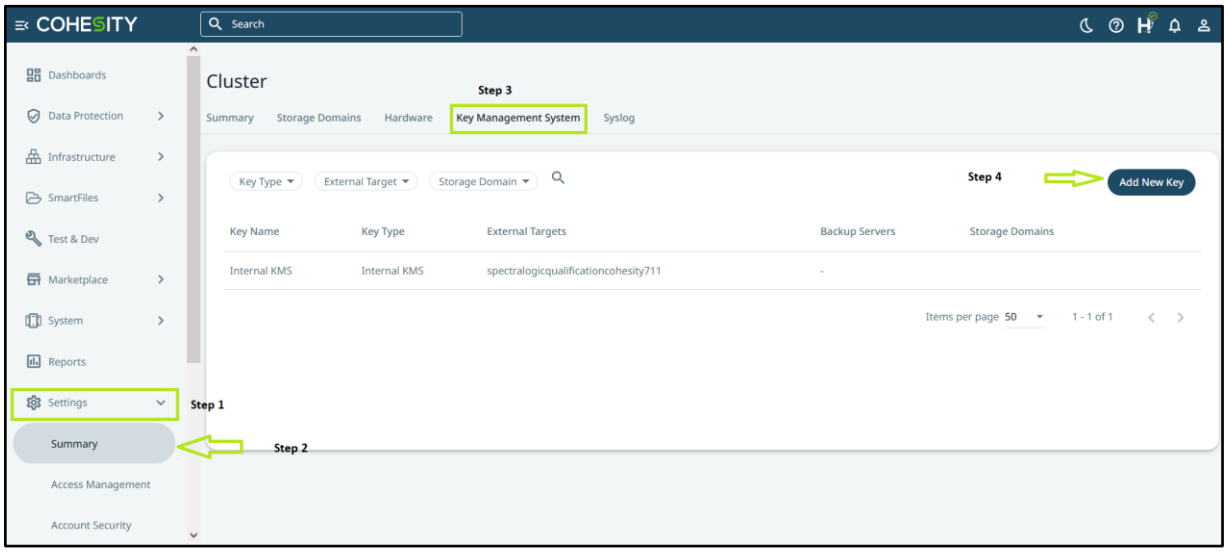
NOTE: The IBM Security Guardium Key Lifecycle Manager IP address cannot be changed once it is configured.

You can configure IBM Security Guardium Key Lifecycle Manager in the Cohesity DataPlatform browser UI or the Cohesity DataPlatform command-line interface (CLI).

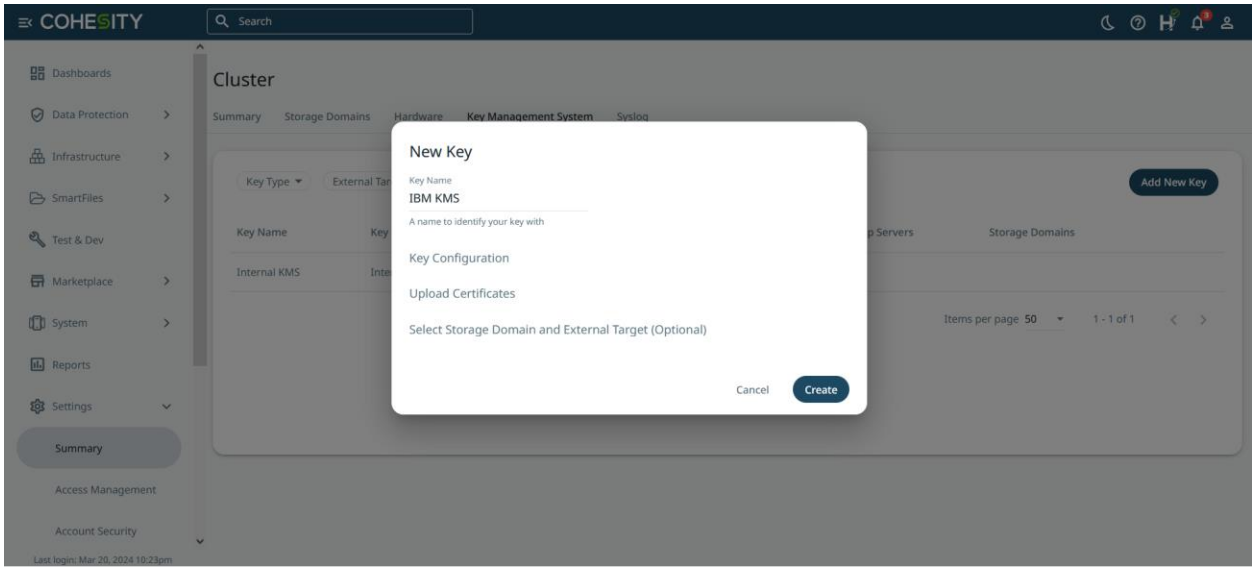
Configure IBM Security Guardium Key Lifecycle Manager in Cohesity DataPlatform UI

To configure IBM Security Guardium Key Lifecycle Manager in the Cohesity UI:

1. Log in to Cohesity DataPlatform.
2. Navigate to **Settings** > **Summary**.
3. On the Cluster Summary page, click the **Key Management System** tab and **Add New Key**.



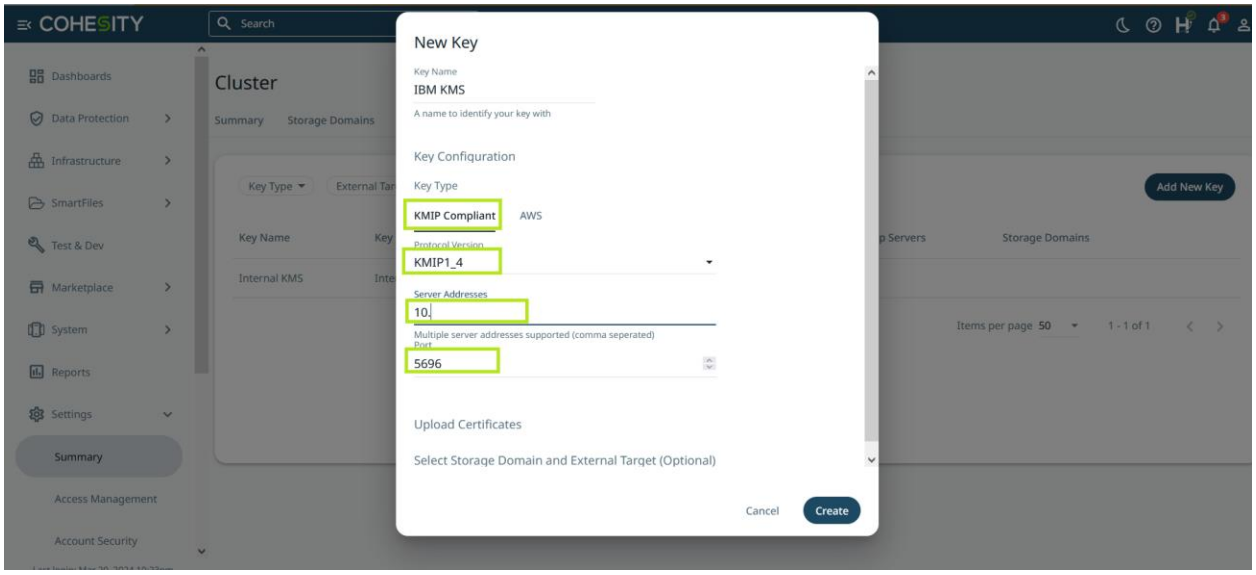
4. In the **New Key** form, type a key name for your IBM KMS key under **Key Name**.



5. Click and expand the **Key Configuration** tab. Under KMIP Compliant:
   - **Protocol Version**. The version of KMIP is to be used.

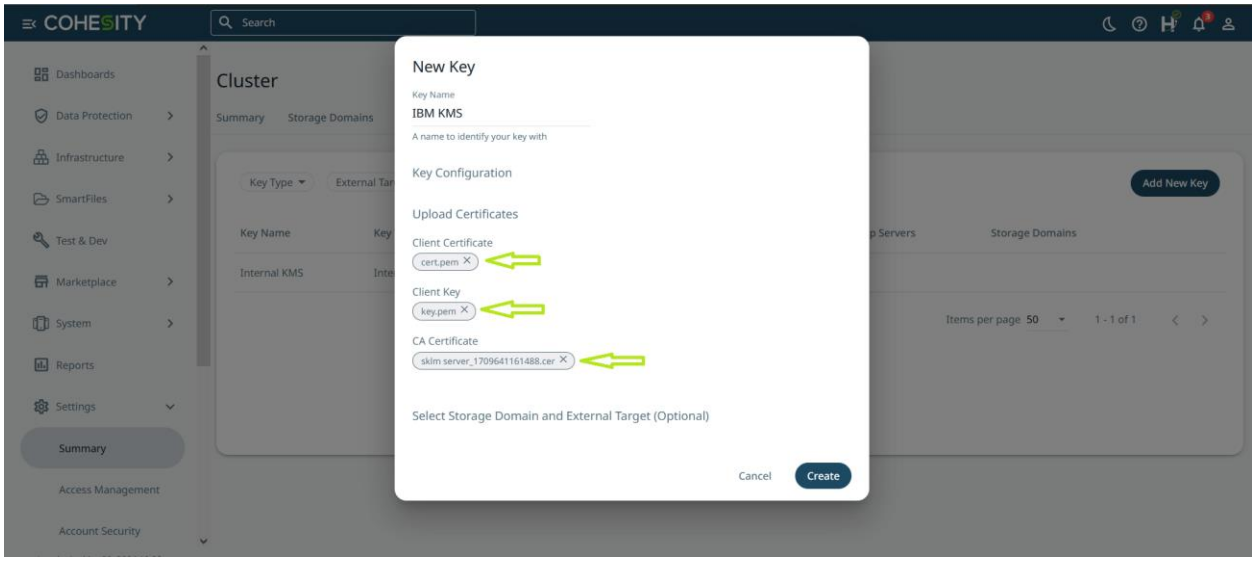
     NOTE: Acceptable options are KMIP1_2, KMIP1_3 or KMIP1_4. For IBM KMS, choose KMIP1_4.

   - **Server Addresses:** Type the IBM KMS's IP address.
   - **Port**: Keep the default port 5696.



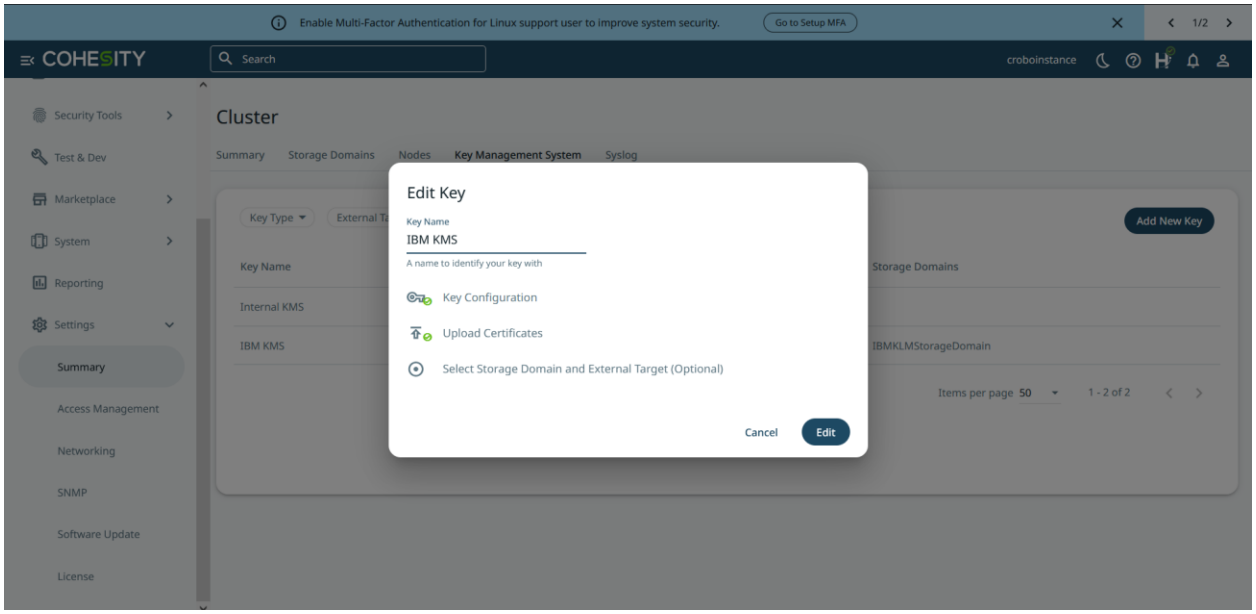
6. Click and expand the **Upload Certificates** tab.
   - **Client Certificate**. Select the client certificate file that you created above.
   - **Client Key**. Select the private key file that you created.
   - **Self-Signed or CA Signed Certificate**. Select the Self-Signed or CA Signed certificate file that you extracted.

- **Select Storage Domain and External Target** (Optional).

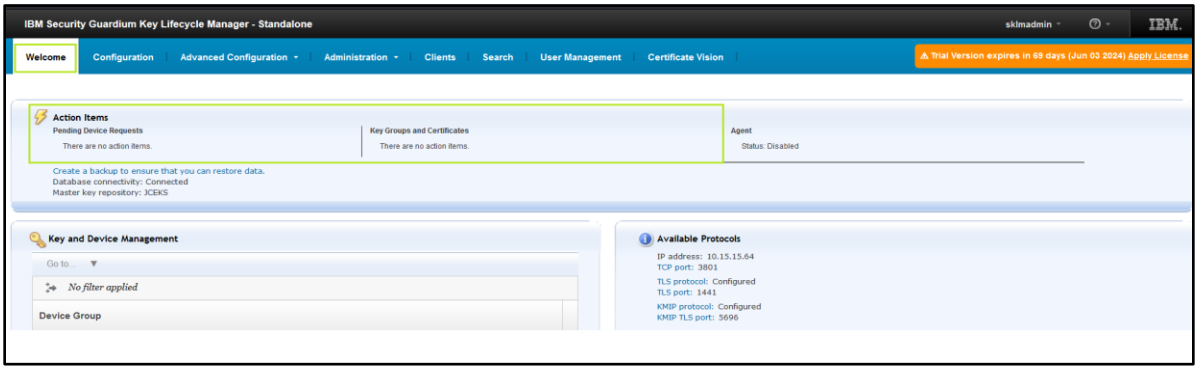


7. Click **Create**.

   The Cohesity cluster immediately attempts to establish a TLS session with IBM Security Guardium Key Lifecycle Manager and initiate KMIP communication. If you get a **KMS Validation Error**, see Troubleshooting.



8. To complete the flow, log in to the IBM KMS server UI, go to **Welcome > Action Items,** look for pending devices, and click **Accept or Modify and Accept**. If you modify the pending device information, make the necessary changes and click **Accept**.



Configure IBM Security Guardium Key Lifecycle Manager in Cohesity DataPlatform CLI

You can also configure IBM Security Guardium Key Lifecycle Manager in the Cohesity DataPlatform command-line interface (CLI). See Using the Cohesity DataPlatform CLI in the online Help for the full list of commands.

To configure IBM Security Guardium Key Lifecycle Manager in the Cohesity DataPlatform CLI:

1. SSH to the cluster using the following command:

```
ssh support@<ip address of node>
```

2. In the CLI, use the **`kms create-kmip`** command.

```
admin@127.0.0.1> kms create-kmip help
  DESCRIPTION
    Create a new kmip KMS.

  PARAMS
    Ca-certificate-path [string] required  File path to ca-certificate.
    Client-certificate  [string] required  File path to client-certificate.
    Client-key          [string] required  File path to client-key.
    Ip                  [string] required  IP address or FQDN of the KMS server.
    Key-name            [string] required  Name of the KMS Key.
    Kmip-protocol-version     [string] required  kmip-protocol-version
    Port                [int]   required  KMS Port. Default KMIP port is 5696.
    Vault-ids           [list of ints] optional  Comma separated list of vault ids.
    View-box-ids [list of ints]optional Comma separated list of Storage Domain IDs.
```

3. After successfully connecting with IBM Security Guardium Key Lifecycle Manager, Cohesity automatically creates one key for a default Storage Domain and other keys for internal use.
If you see the above keys in the GUI, the KEKs have migrated from your Cohesity cluster to the IBM Security Guardium Key Lifecycle Manager. This completes the integration of IBM Security Guardium Key Lifecycle Manager with the Cohesity cluster.

Update Cohesity Key Management Configuration

If your KMS configuration has changed because of expired certificates or a change in the KMIP protocol version, you can update the Key Management System settings in Cohesity DataPlatform. As with the initial Cohesity KMS configuration, you can update these in the Cohesity DataPlatform browser UI or in the Cohesity DataPlatform CLI.

Important: If you modify the Key Management System settings on the Cohesity cluster at any point after initial configuration, you must manually restart the keychain service. See Keychain Service below.

Update Cohesity KMS in Cohesity DataPlatform UI

To update the Cohesity KMS settings using the browser UI:

1. Navigate to **Settings** > **Summary** and click the **Key Management System** tab.
2. On the **Key Management System** page, you can edit the configured KMS details such as **Protocol Version**, **Server IP** and the associated **certificates**. When you are done, click **Edit.**

Update Cohesity KMS in Cohesity DataPlatform CLI

You can also update the Cohesity KMS settings in the Cohesity DataPlatform CLI. For the full list of commands, see Using the Cohesity DataPlatform CLI in the online Help.

To update the Cohesity KMS settings using the Cohesity DataPlatform CLI:

1. SSH to the cluster using the following command:

```
ssh support@<ip address of node>
```

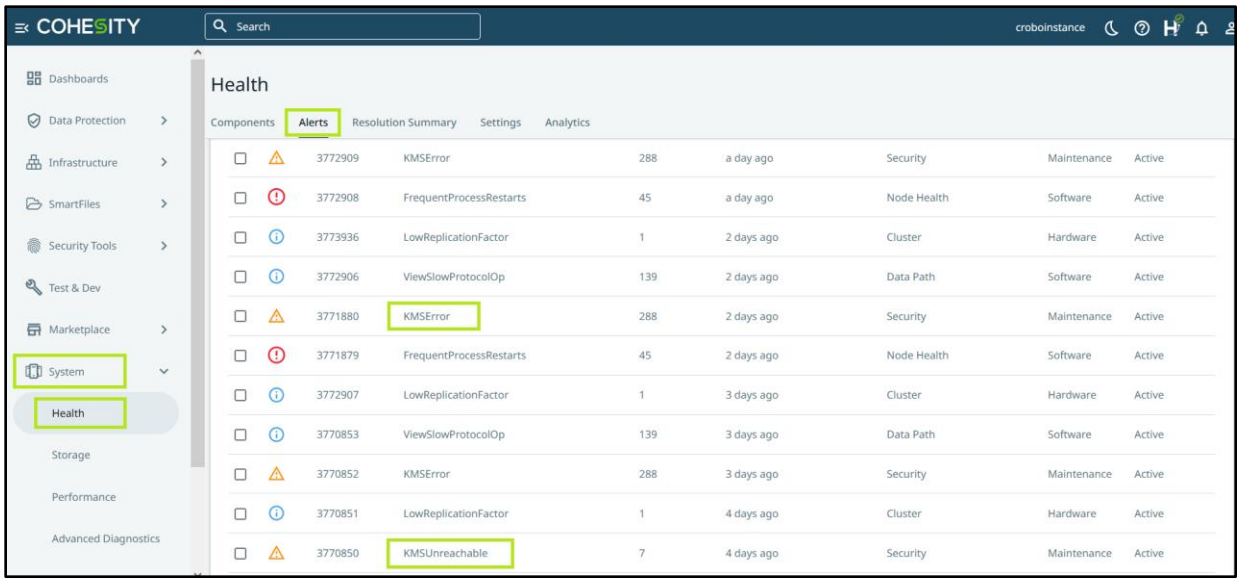
2. In the CLI, use the **`kms update-kmip`** command.

```
admin@127.0.0.1> kms update-kmip help
  DESCRIPTION
    Update an existing kmip KMS.

  PARAMS
    Ca-certificate-path [string]     optional  File path to ca-certificate.
    Client-certificate  [string]     optional  File path to client-certificate.
    Client-key          [string]     optional  File path to client-key.
    Confirm             [bool]       optional  Confirming to update KMS server.
Please make sure the new updated information is for thte same KMS server configured
before.
    Id                  [int]        required  The Id of a KMS server.
    Ip                  [string]     optional  IP address or FQDN of the KMS
server.
    Key-name            [string]     optional  Name of the KMS Key.
    Kmip-protocol-version     [string]     optional  kmip-protocol-version
    Port                [int]        optional  KMS Port. Default KMIP port is 5696.
    Vault-ids           [list of ints]      optional  Comma separated list of vault
ids.
    View-box-ids        [list of ints]      optional  Comma separated list of
Storage Domain IDs.

  RULES
    All params from the set: client-key, ca-certificate-path need to be specified
    At least one of ip, port, client-certificate, client-key, view-box-ids, vault-
ids should be specified.
```

NOTE: If at any point after initial configuration you modify the Key Management System settings on the Cohesity cluster, you must manually restart the keychain service. See Keychain Service below.

Troubleshooting

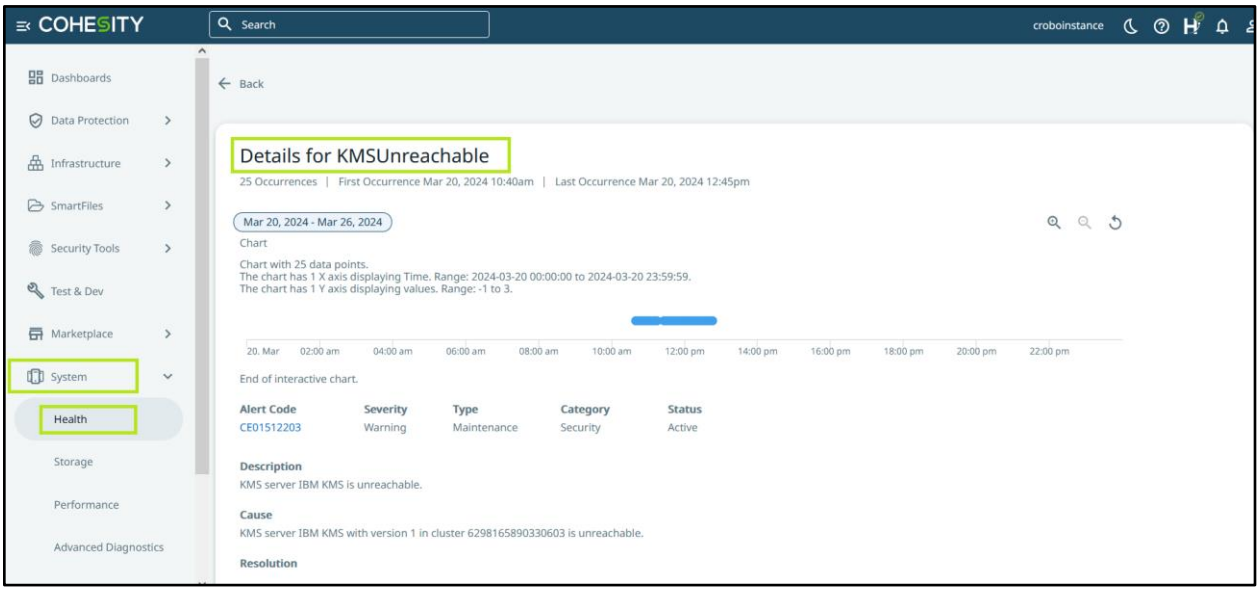
You might encounter errors while configuring KMS, Storage Domain settings, when the Certificate expires, or during workflows like Backup, Recovery, and Clone. Users can monitor the hygiene of KMS by visiting **System > Health > Alerts.**

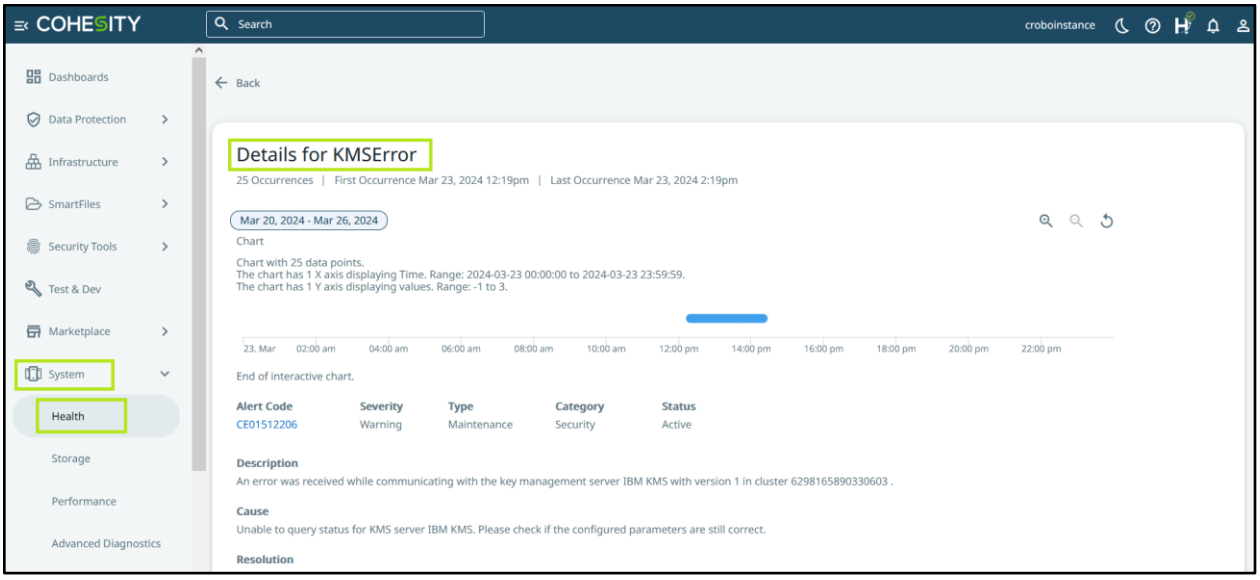


About Errors

- KMS unreachable error when the KMS is not reachable.
- KMS Certification Error when a Certificate expires.
- KMS Error when there are errors in communication, authentication, insufficient privileges, KMS setup, etc.

If the Cohesity cluster cannot communicate with the IBM Security Guardium Key Lifecycle Manager, the following generic KMS validation error appears:





Troubleshoot Errors

To troubleshoot errors, take one or more of the following steps:

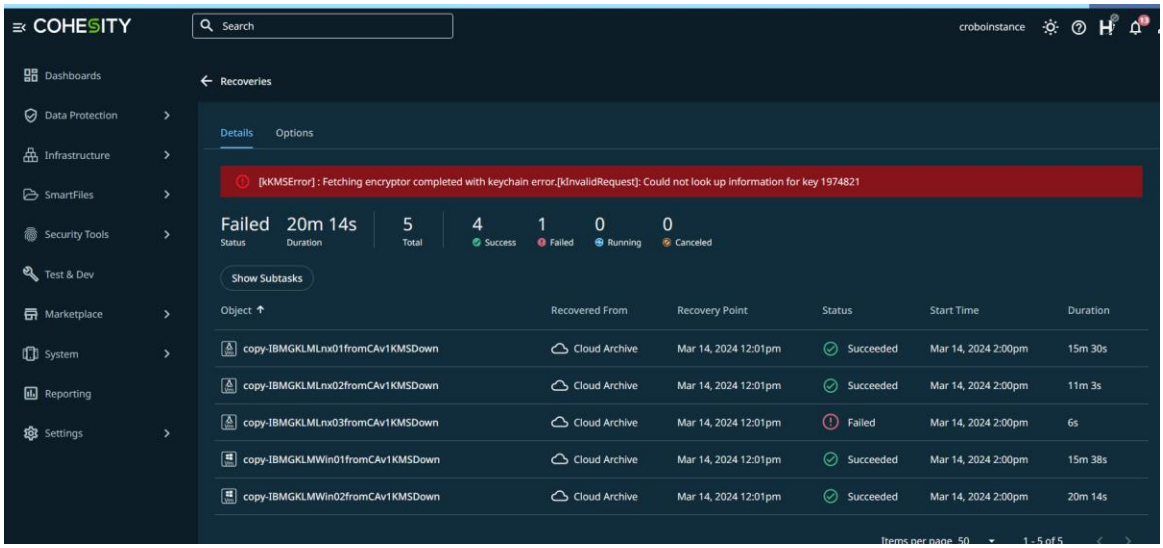
1. Verify correct addressing and basic network connectivity between IBM Security Guardium Key Lifecycle Manager and the Cohesity cluster.
2. Verify port 5696 is configured on the Cohesity DataPlatform KMS settings page and that firewalls are open for that port.
3. If any of the uploaded certificate files or private key file on the Cohesity DataPlatform KMS settings page were created on a Windows system, recreate them on a Linux system.

   NOTE: The Cohesity KMS client only accepts an SSL certificate that contains a Unix-style newline character, which is '\n'. Format your certificates accordingly — in Windows, replace '\r\n' with '\n' and on Mac OS, replace '\r' with '\n' — and then load the certificates.
4. Verify that the CA certificate uploaded on the Cohesity DataPlatform KMS settings page is the internal root CA certificate from IBM Security Guardium Key Lifecycle Manager. It should *not* be the IBM Security Guardium Key Lifecycle Manager's server certificate. The Cohesity cluster needs the root CA certificate to validate the server certificate that is delivered to it while establishing a TLS session.
5. If you are using an externally signed client certificate, ensure all CA chain certificates have been imported into IBM Security Guardium Key Lifecycle Manager via the **System > KMIP Trusted CA Certificates** page.
6. If you are using a self-signed client, ensure no KMIP Trusted CA Certificates are imported into IBM Security Guardium Key Lifecycle Manager. If there are, you'll have to create an externally signed client certificate signed by one of those trusted CAs.
7. Verify the host configuration on IBM Security Guardium Key Lifecycle Manager for the Cohesity cluster:
8. The **Host Name** must match the **Common Name** in the client certificate.
9. The client certificate must be uploaded into the host configuration on IBM Security Guardium Key Lifecycle Manager, and its **Certificate Fingerprint** should be visible on the host configuration page.
10. The KMIP **Registration Allowed** and **Communication Enabled** settings should be checked.
11. Proper licensing must be in place.
12. The error is that the TLS session with IBM Security Guardium Key Lifecycle Manager has been dropped due to inactivity. The Cohesity cluster will immediately take action to re-establish the connection, but only after you see the KMS is unreachable error message. To remedy this, simply click the Create Storage Domain button to try again. If the problem was a dropped TLS session, it should work the second time.
13. Suppose the problem was not just the lack of a TLS session, and there is indeed a connectivity issue of some type. In that case, you will either continue to see the KMS is unreachable error or possibly the internal error message below. Ensure the connection is established again.

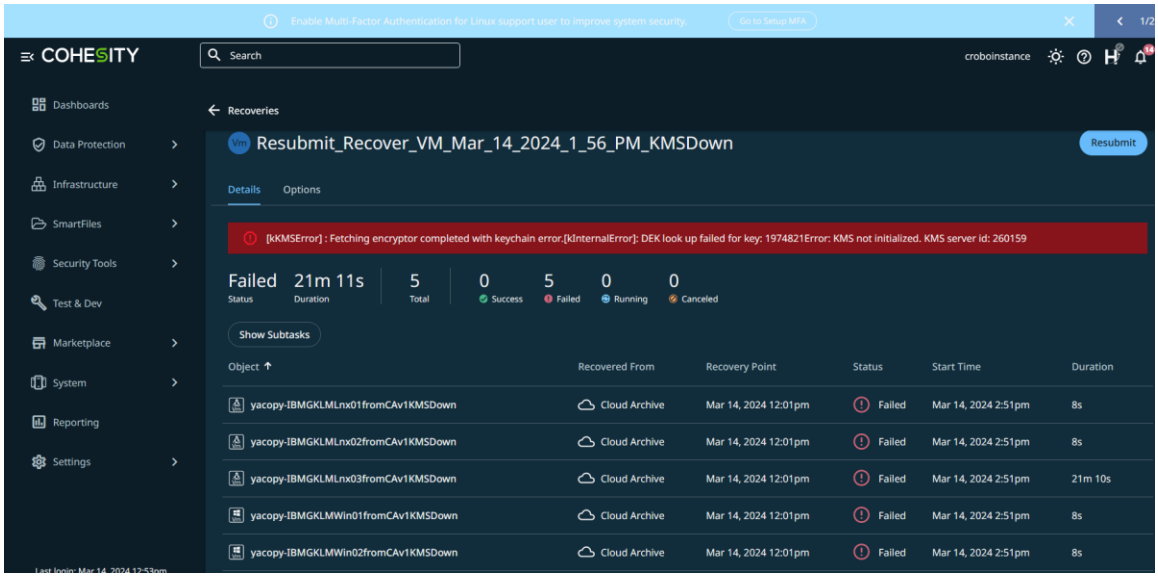
Workflow ERRORs when KMS Server is Unreachable

Below are a few examples of symptoms noticed when KMS Server is Unreachable or Down for some reason.

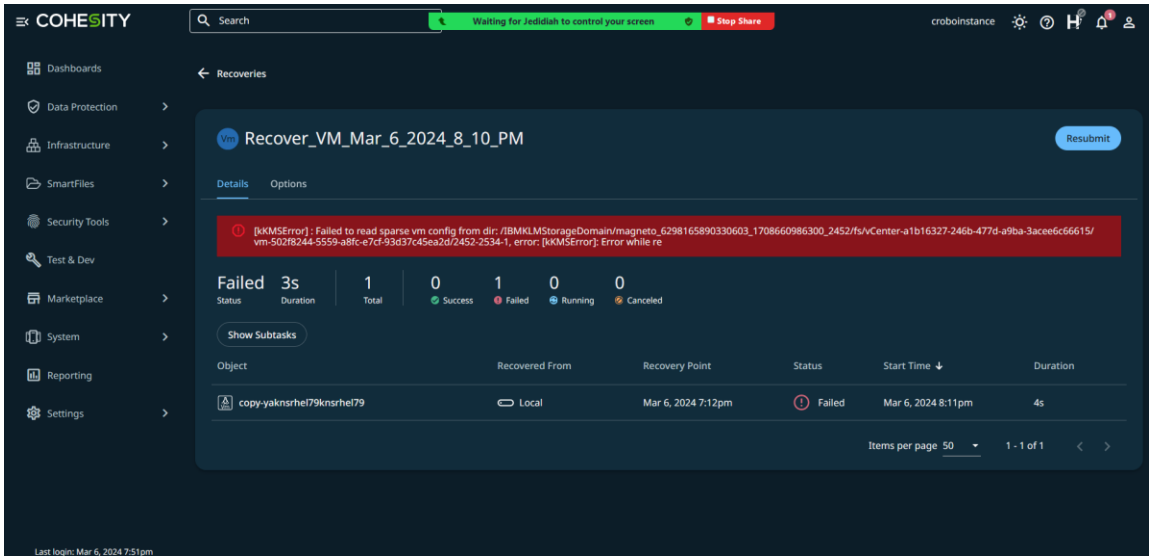
1. Recover of VMs from backup when KMS Server is down:



2. Recover of VM from Cloud Archive v1 when KMS is down:



3. Another attempt at recovery of VM:



Keychain Service

The keychain service on Cohesity DataPlatform is responsible for communicating with the IBM Security Guardium Key Lifecycle Manager.

An error log for the keychain service exists in the `/logs` directory of the Cohesity instance. It is accessible by SSHing into the instance to access the Linux OS. The filenames are `keychain_exec.ERROR`, `keychain_exec.INFO` and `keychain_exec.FATAL`.

Restart the Keychain Service after Key Management Settings Update

If you update the Key Management settings after initially configuring them, you must restart the keychain service for the new settings to take effect.

To restart the Cohesity keychain service in the Cohesity DataPlatform CLI,

1. Start the Cohesity DataPlatform CLI remotely or locally using the following command.

```
$ iris_cli -server <x.x.x.x> -username=<username> -password=<password>
```

2. Issue the following command:

```
$ cluster restart service-names="keychain"
```

See Restart the Cohesity Keychain Service in the online Help for more.

Your Feedback

Was this document helpful? Send us your feedback!

About the Authors

Shivananda K N Senior Product Solutions Architect, Product Solutions WWFO, at Cohesity. In his role, Shivananda focuses on enterprise data protection, solution validation, solution testing, solution qualification, and software usability.

Other essential contributors included:

- Jedidiah Sonavane, Product Solutions Architect at Cohesity
- Luke Walker, Lead Product Manager at Cohesity

Document Version History

VERSION	DATE	DOCUMENT HISTORY
1.0	June 2024	First release

ABOUT COHESITY

Cohesity is a leader in AI-powered data security and management. Aided by an extensive ecosystem of partners, Cohesity makes it easier to protect, manage, and get value from data – across the data center, edge, and cloud. Cohesity helps organizations defend against cybersecurity threats with comprehensive data security and management capabilities, including immutable backup snapshots, AI-based threat detection, monitoring for malicious behavior, and rapid recovery at scale. Cohesity solutions are delivered as a service, self-managed, or provided by a Cohesity-powered partner. Cohesity is headquartered in San Jose, CA, and is trusted by the world's largest enterprises, including six of the Fortune 10 and 42 of the Fortune 100.

Visit our website and blog, follow us on Twitter and LinkedIn and like us on Facebook.