EDB

EDB Postgres Advanced Server

17.5

Integration Guide

ESKM

8.54.0

utimaco®

Imprint

Copyright 2025	Utimaco IS GmbH
	Germanusstr. 4
	D-52080 Aachen
	Germany
Phone	AMERICAS +1-844-UTIMACO (+1 844-884-6226)
	EMEA +49 800-627-3081
	APAC +81 800-919-1301
Internet	https://support.hsm.utimaco.com/
e-mail	support@utimaco.com
Document Version	1.0.0
Date	2025-07-18
Status	PUBLISHED
Document No.	IG-2025-0046

Table of Contents

1 Introduction

This guide is part of the information and support provided by Utimaco.

This guide explains the integration of Utimaco ESKM with EDB Postgres Advanced Server to protect sensitive data. Integrating EDB Postgres Advanced Server and Utimaco ESKM enhances data security by enabling centralized and secure management of encryption keys.

The guide walks through the necessary steps to integrate the Utimaco ESKM with EDB Postgres Advanced Server.

1.1 About This Guide

This guide describes how to integrate EDB Postgres Advanced Server with Utimaco ESKM to protect the database. It equips users with key data to facilitate effortless communication and authentication between ESKM and EDB Postgres Advanced Server, implementing the Key Management Interoperability Protocol (KMIP) and certificate-based authentication.

1.2 Target Audience

This guide is intended for EDB Postgres Advanced Server and Utimaco ESKM administrators.

1.3 Purpose of the Integration

The integration of EDB Postgres Advanced Server with Utimaco ESKM enhances the security and compliance of your data protection strategy. ESKM provides robust encryption key lifecycle management, including generation, storage, access control, and auditing.

The primary objective of this integration is to:

- Enhance data security by encrypting the data at rest.
- Leverage encryption capabilities of EDB Postgres within the Utimaco ESKM environment.

1.4 Document Conventions

The following conventions are used in this guide:

Convention	Use	Example
Bold	Items of the Graphical User Interface (GUI), e.g., menu options	Press **OK**
`Monospaced`	Code that is given for explanation or as an example, file paths	`chsm-create`
Italic	References and important terms	See *Sample Chapter* in the *CryptoServer - Sample Manual*

Table 1: Document Conventions

We use special icons to highlight the most important notes and information.

Here, you find important safety information that should be followed.

Here, you find additional notes or supplementary information.

This message marks the result expected after the successful execution of an instruction.

1.5 Abbreviations

Abbreviation	Meaning
ESKM	Enterprise Secure Key Manager
KMIP	Key Management Interoperability Protocol

Abbreviation	Meaning
EDB	Enterprise DB
TDE	Transparent Data Encryption
API	Application Programming Interface
CA	Certificate Authority
CN	Common Name
KMS	Key Management System
URL	Uniform Resource Locator
AES	Advanced Encryption Standard
CSR	Certificate Signing Request
RSA	Ron Rivest, Adi Shamir, and Leonard Adleman
SHA	Secure Hash Algorithm
TLS	Transport Layer Security
SSL	Secure Sockets Layer
EPEL	Extra Package for Enterprise Linux

Table 2: Abbreviations

2 Overview

2.1 EDB Postgres Advanced Server

EDB Postgres Advanced Server is a powerful, enterprise-ready version of Postgres developed by EnterpriseDB. It enhances the open-source Postgres database with advanced features like Oracle compatibility, high availability, performance tuning tools and features that make it more suitable for enterprise environments.

2.2 Utimaco ESKM (Enterprise Secure Key Manager)

Utimaco ESKM is a complete solution for generating, storing, serving, controlling, and auditing access to encryption keys. It enables you to protect and preserve access to business-critical, sensitive data-at-rest encryption keys, either locally or remotely. ESKM is offering industry-certified Key Management Interoperability Protocol (KMIP) with market-leading support for partner applications and pre-qualified solutions, integrating out-of-the-box with varied deployments, as well as custom integrations.

2.3 Joint Value Proposition

The integration of EDB Postgres with Utimaco ESKM delivers a comprehensive and secure data protection solution that provides advanced encryption key management. This joint solution empowers organizations to:

- Enhanced Security will be achieved by storing the keys separately from the data, reducing the data management risk. Supports key rotation.
- Centralized key management simplifies the operations across multiple databases and environments.
- Ensure end-to-end data security by encrypting data at rest.
- Simplify compliance with regulatory standards through centralized key lifecycle management, including generation, storage, access control, and auditing.
- Maximize operational efficiency by seamlessly leveraging EDB Postgres's encryption features within the ESKM environment.
- Strengthen resilience against data breaches and unauthorized access.

This integration empowers organizations to secure sensitive data with confidence, maintain operational agility, and simplify compliance.

3 Integration Requirements and Prerequisites

Ensure that the system environment you will be using meets the following hardware and software requirements.

This guide assumes that the user has already installed and configured the required software.

3.1 Tested Versions

The integration has been successfully tested with the Utimaco ESKM and EDB Postgres Advanced Server.

Operating System	EDB Postgres Advanced Server Version	Utimaco ESKM Version
Linux Server (Rocky Linux 9.5)	17.5, 16.9, 15.13	8.54.0

Table 3: Tested Versions

3.2 Supported Platforms

- Utimaco ESKM hardware appliance
- Utimaco ESKM virtual/cloud appliance

3.3 Software Requirements

Software	Software Requirements
Utimaco ESKM	8.54.0
EDB Postgres Advanced Server	17.5, 16.9, 15.13

Table 4: Software Requirements

3.4 Prerequisites

Before you begin, please ensure that you have installed/set up:

- **EDB Postgres Advanced Server**: Ensure the supported version is installed and properly configured.
- **Licensing**: A valid EDB Postgres license that supports API access and integration features.
- **ESKM:** Ensure that the latest version of ESKM is available.

4 Installation and Configuration

The following section outlines the procedures for configuring both the ESKM and EDB Postgres Advanced Server components for seamless integration.

4.1 Setting Up ESKM

The initial phase involves configuring ESKM before proceeding to EDB Postgres Advanced Server. For detailed configuration steps, see the *"ESKM_Installation and Replacement_Guide_8.54.0"* installation guide.

After successful installation and configuration, log in to ESKM.

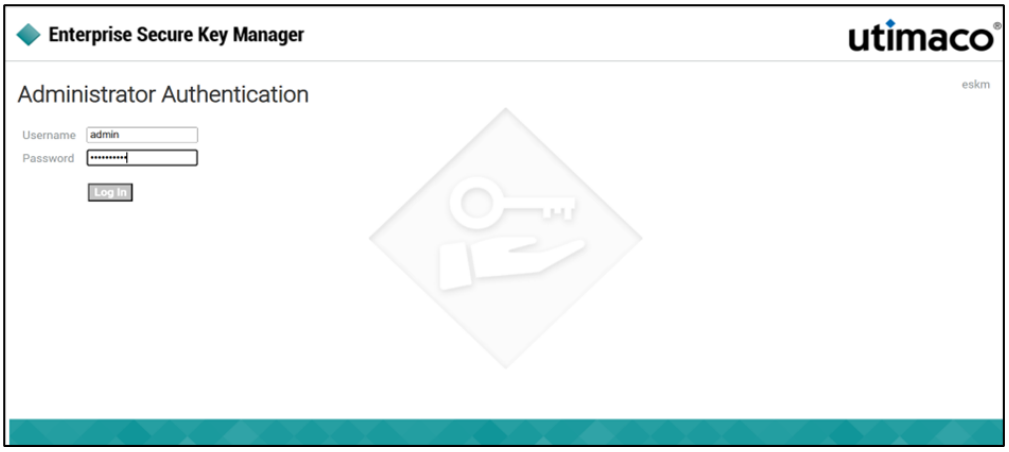

Figure 1 : ESKM log-in page

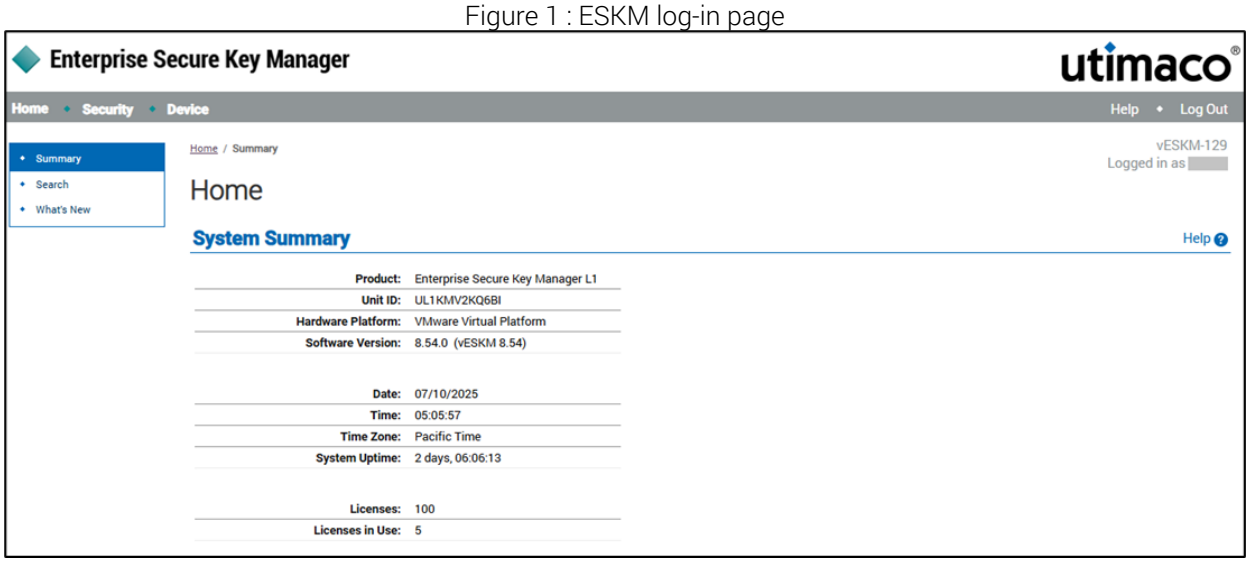

Figure 2 : ESKM Home page

4.1.1 Create a Local CA

Inside ESKM create a local CA by following the below steps:

1. Go to the **Security** tab.
2. From the left side panel, click on the **Certificates** option listed under **Certificates & CAs**.
3. Scroll down the main page and go to the **Create Certificate** section.
4. Enter a **Certificate Authority Name** and **Common Name**. These may have the same value, such as ESKM Local CA.
5. Enter your **Organizational information**.
6. Select the **Algorithm** (e.g., RSA-2048).
7. Click on **Self-signed Root CA** and enter the **CA Certification Duration** and **Maximum User Certificate Duration**. These values determine when the certificate must be renewed and should be set in accordance with your company's security policies.
   The default value for both is 3650 days or 10 years.
8. Click on **Create.**

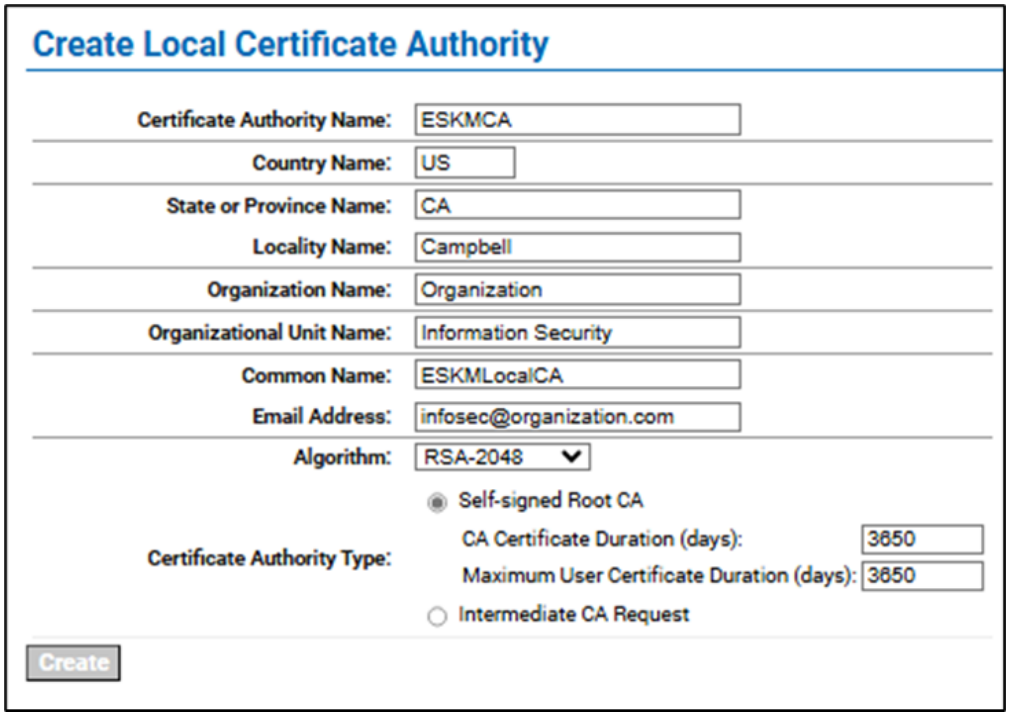

Figure 3 : Local CA creation with common name as 'ESKMLocalCA'

9. Click the **Local CAs** option listed under **Certificates & CAs** from the left-hand side panel to display the created local CA certificate.

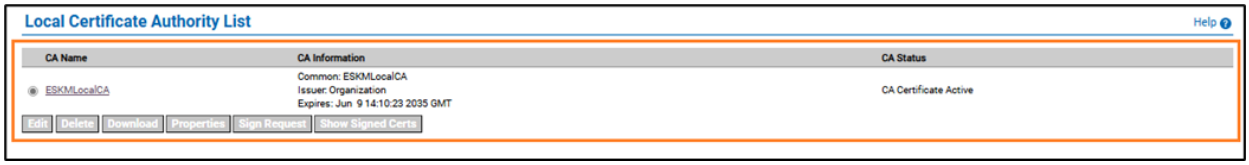

Figure 4 : Created Local CA

4.1.2 Create a Server Certificate

The client uses ESKM server certificates to authenticate the ESKM server during the TLS/SSL handshake.

To create an ESKM server certificate, perform the following steps:

1. Go to the **Security** tab.
2. Click on the **Certificates** option listed under **Certificates & CAs**.
3. Scroll down to the **Create Certificate** section.
4. Enter **Certificate Name**, **Country Name**, **State and Province Name**, **Locality Name**, **Organization Name,** and **Organization Unit Name**
5. Select **RSA-2408** from the **Algorithm** drop-down list.
6. Select the previously created CA certificate name from the **Local CA** drop-down list.
7. Select **Server** from the **Certificate Purpose** drop-down list.
8. Click on **Create.**

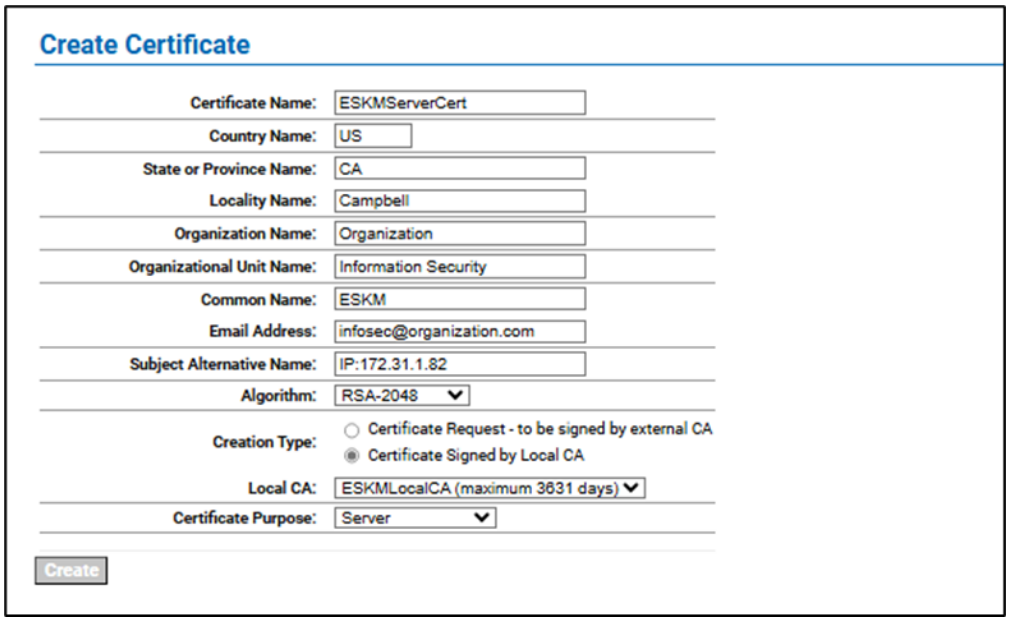

Figure 5 : Certificate creation

4.1.3 Configure the KMIP Server

1. Go to the **Device** tab.
2. From the left-hand side panel, click on the **KMIP Server** option listed under **Device Configuration**.
3. Click the **Edit** button on the main page.
4. Choose the created server certificate as the **Server Certificate** for the KMIP Server.
5. Click the **Save** button.

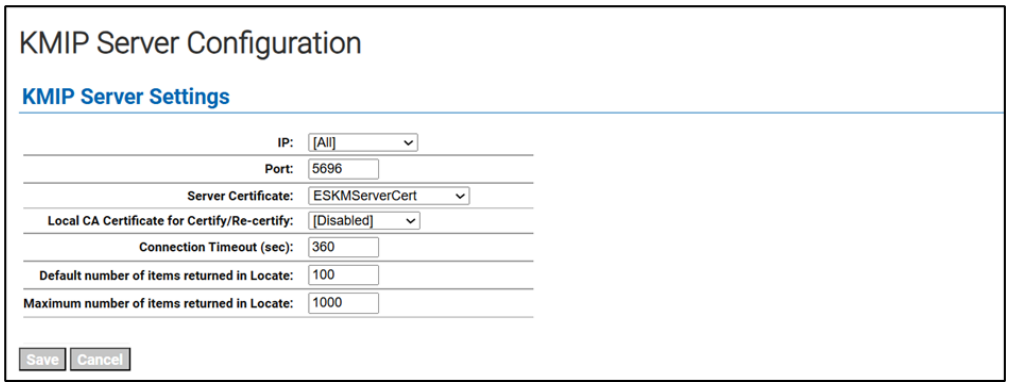

Figure 6 : KMIP Server Configuration

4.2 Setting up the EDB Postgres Advanced Server

Follow the steps below for setting up the EDB Postgres Advanced Server on a Linux server. The steps are based on the EDB Postgres Advanced Server *EDB Docs - EDB Postgres Advanced Server v17 - Installing EDB Postgres Advanced Server on AlmaLinux 9 or Rocky Linux 9 x86_64*.

1. Setting up the repository is a one-time task. Check if the repository already exists on the Linux machine.
   Use the commands below to confirm the repository exists. If no output is generated, the repository isn't installed.
   ```
   $ sudo dnf repolist | grep enterprisedb
   ```

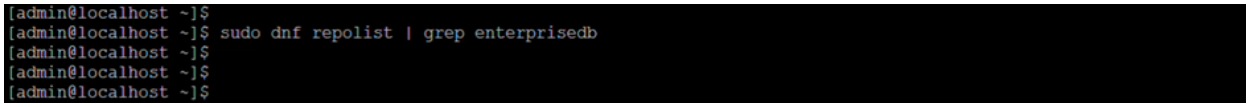

Figure 7 : Linux Terminal: command related to EDB repo check

2. Go to the EDB repository https://www.enterprisedb.com/repos-downloads.
   Accessing this repository requires an EDB user account, which can be created from the following URL: *Account management | EDB*.
   Select **Access EDB Repos 2.0** as highlighted in the figure below.

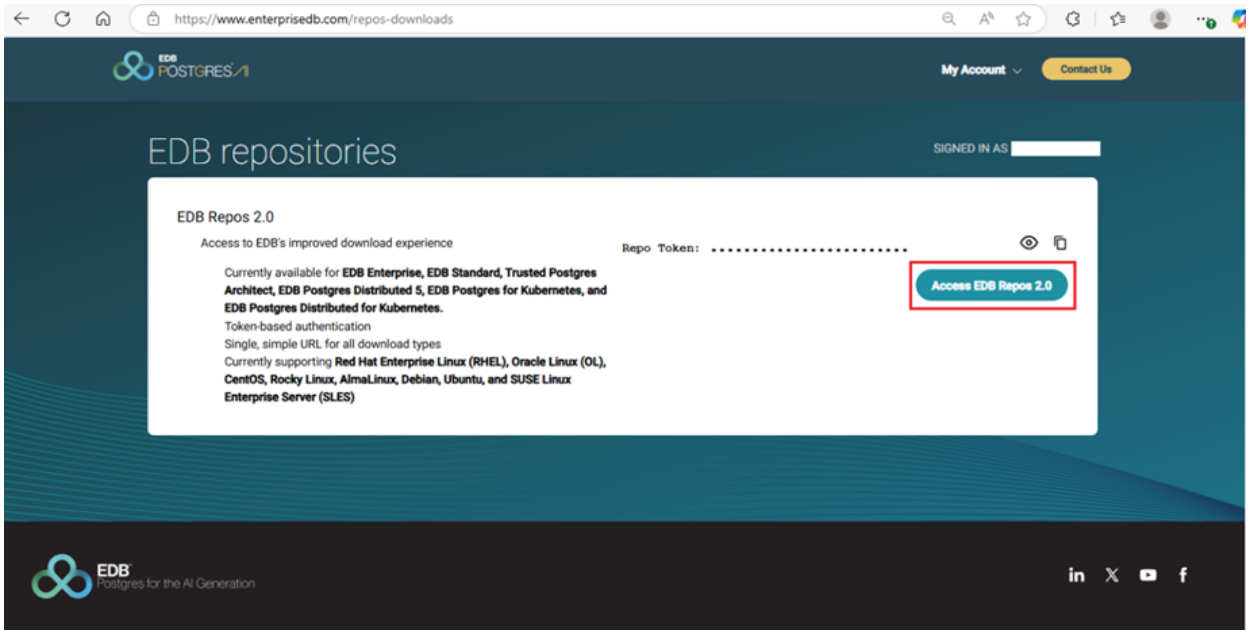

Figure 8 : EDB official website: Repo access

3. Select the required platform and software to download.
   a. Select the platform from the **Select Platform** drop-down list.

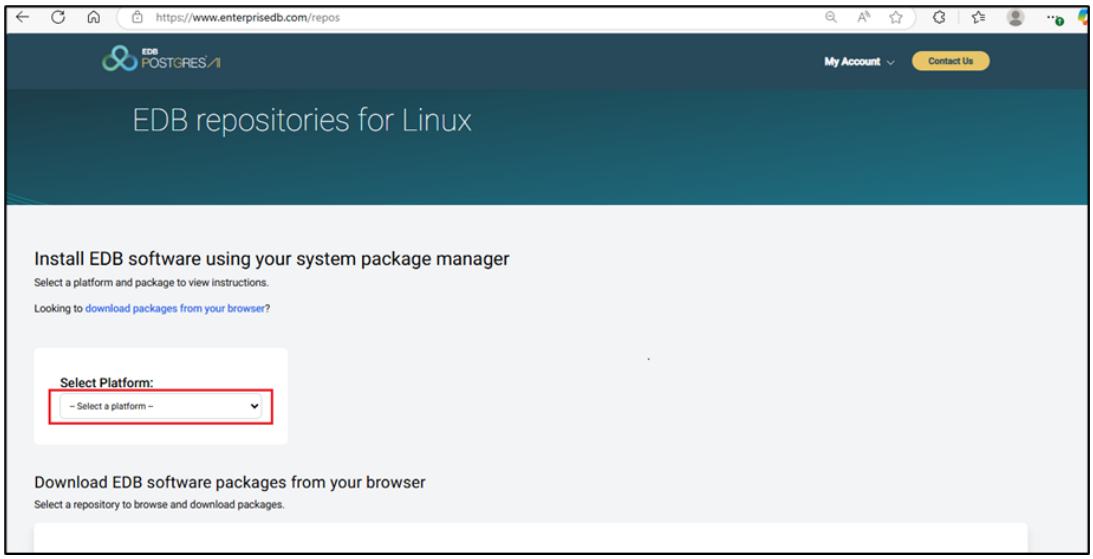

Figure 9 : EDB official website: Platform selection for EDB repo

b. Select Rocky Linux 9 from the drop-down list, as it is used for testing.

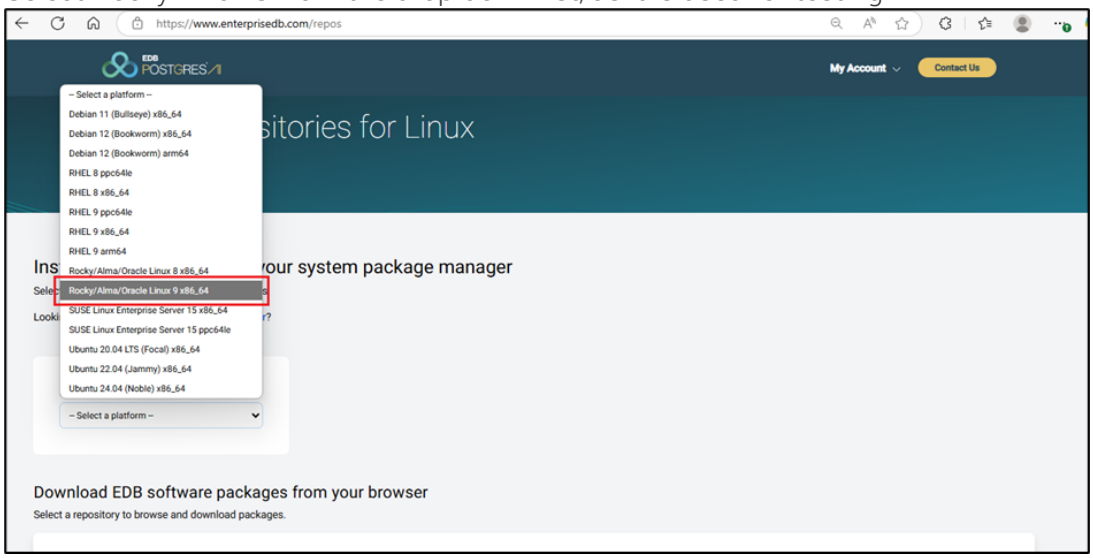

Figure 10 : EDB official website: Rocky Linux 9 selection

c. Select **Software EDB Postgres Advanced Server version 17,** as the figure below highlights.

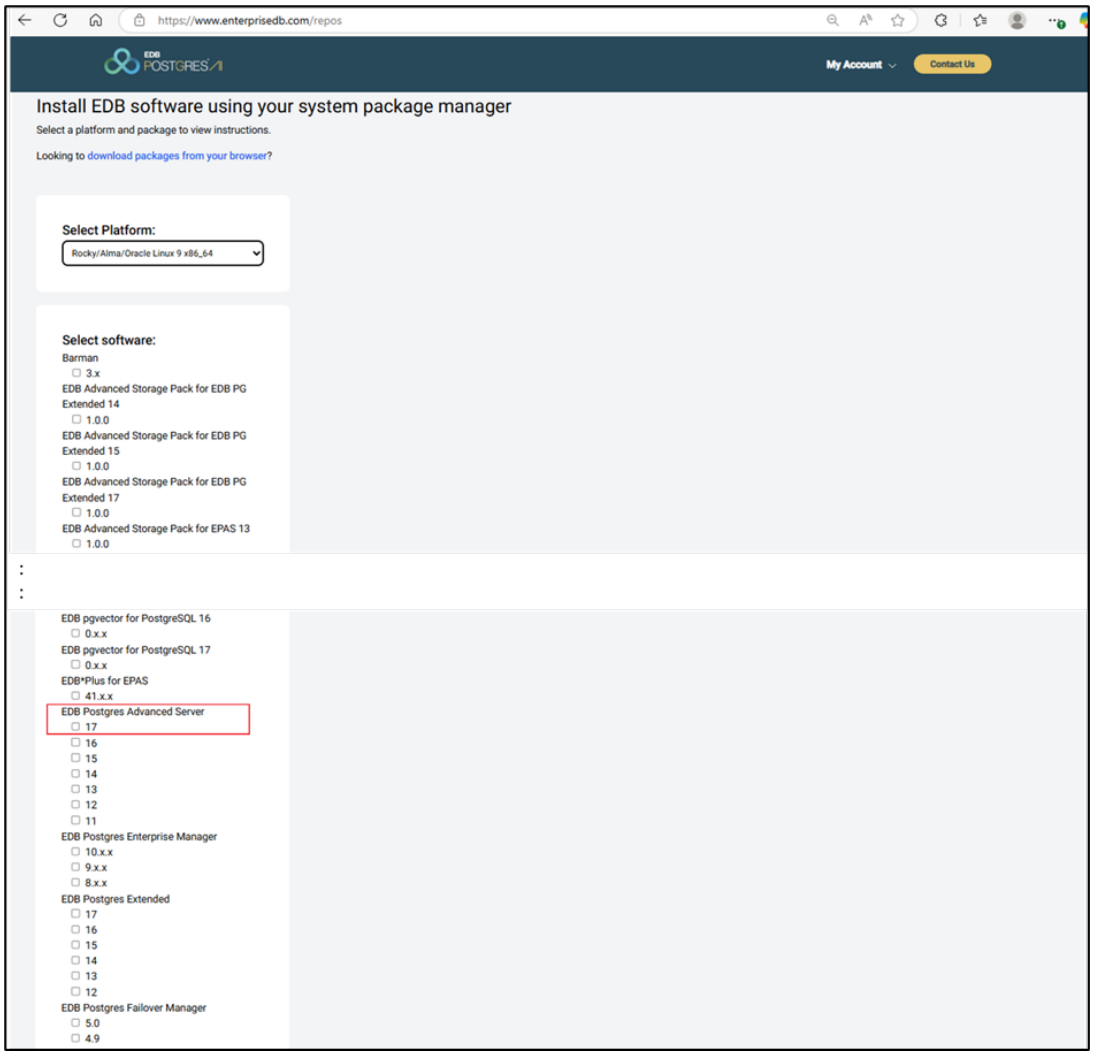

Figure 11 : EDB official website: Select the software 'EDB Postgres Advanced Server'

4. Set up the repository using the link from the EDB repository
    a. The repository URL is specified in the figure below.

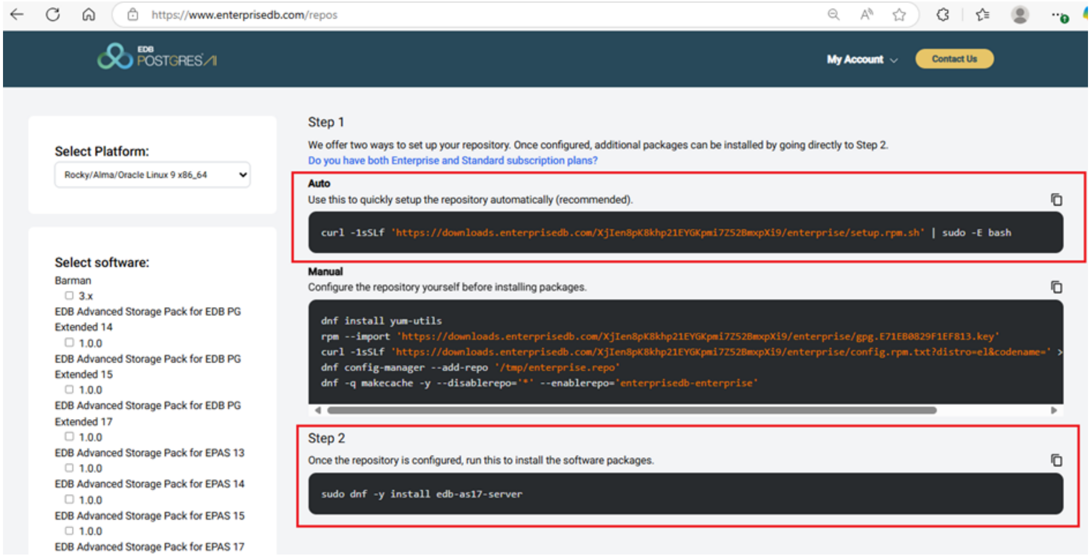

Figure 12 : EDB official website: EDB Postgres Advanced Server repo urls

b. Execute the command below on a Linux machine.

`$ curl -1sSLf 'https://downloads.enterprisedb.com/XjIen8pK8khp21EYGKpmi7Z52BmxpXi9/enterprise/setup.rpm.sh' | sudo -E bash`

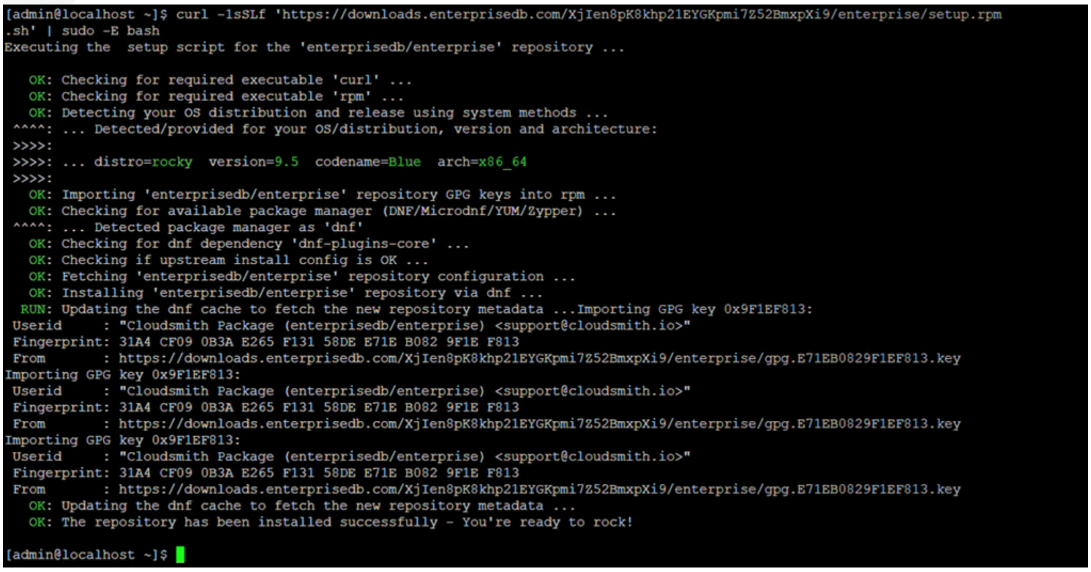

Figure 13 : Linux Terminal: EDB repository setup

5. Install the software packages.
   The command for installing the software package is `sudo dnf -y install edb-as<xx>-server`, where `<xx>` is the version of EDB Postgres Advanced Server. For

version 17, use the command below.

```
$ sudo dnf -y install edb-as17-server
```

```
[admin@localhost ~]$
[admin@localhost ~]$ sudo dnf -y install edb-as17-server
Last metadata expiration check: 0:01:58 ago on Thu 19 Jun 2025 11:04:45 PM PDT.
Dependencies resolved.
================================================================================
 Package                    Architecture   Version              Repository                 Size
================================================================================
Installing:
 edb-as17-server            x86_64         17.5.0-2.el9         enterprisedb-enterprise    11 k
Installing dependencies:
 edb-as17-server-client     x86_64         17.5.0-2.el9         enterprisedb-enterprise   1.9 M
 edb-as17-server-contrib    x86_64         17.5.0-2.el9         enterprisedb-enterprise   798 k
 edb-as17-server-core       x86_64         17.5.0-2.el9         enterprisedb-enterprise   7.5 M
 edb-as17-server-docs       x86_64         17.5.0-2.el9         enterprisedb-enterprise    19 k
 edb-as17-server-ecpg       x86_64         17.5.0-2.el9         enterprisedb-enterprise   1.4 M
 edb-as17-server-libs       x86_64         17.5.0-2.el9         enterprisedb-enterprise   653 k
 edb-as17-server-plperl     x86_64         17.5.0-2.el9         enterprisedb-enterprise    73 k
 edb-as17-server-plpython3  x86_64         17.5.0-2.el9         enterprisedb-enterprise   110 k
 edb-as17-server-pltcl      x86_64         17.5.0-2.el9         enterprisedb-enterprise    48 k
 lz4                        x86_64         1.9.3-5.el9          Atalla_base                58 k
Installing weak dependencies:
 beacon-agent               x86_64         1.226.16-1           enterprisedb-enterprise    84 M

Transaction Summary
================================================================================
Install  12 Packages

Total download size: 97 M
Installed size: 273 M
Downloading Packages:
(1/12): lz4-1.9.3-5.el9.x86_64.rpm                          276 kB/s |  58 kB     00:00
(2/12): edb-as17-server-17.5.0-2.el9.x86_64.rpm             8.6 kB/s |  11 kB     00:01

(3/12): edb-as17-server-client-17.5.0-2.el9.x86_64.rpm      802 kB/s | 1.9 MB     00:02
(4/12): edb-as17-server-contrib-17.5.0-2.el9.x86_64.rpm     492 kB/s | 798 kB     00:01
(5/12): edb-as17-server-docs-17.5.0-2.el9.x86_64.rpm         14 kB/s |  19 kB     00:01
(6/12): edb-as17-server-core-17.5.0-2.el9.x86_64.rpm        3.0 MB/s | 7.5 MB     00:02
(7/12): beacon-agent-1.226.16_linux_amd64.rpm                14 MB/s |  84 MB     00:06
(8/12): edb-as17-server-ecpg-17.5.0-2.el9.x86_64.rpm        773 kB/s | 1.4 MB     00:01
(9/12): edb-as17-server-libs-17.5.0-2.el9.x86_64.rpm        370 kB/s | 653 kB     00:01
(10/12): edb-as17-server-plperl-17.5.0-2.el9.x86_64.rpm      58 kB/s |  73 kB     00:01
(11/12): edb-as17-server-plpython3-17.5.0-2.el9.x86_64.rpm   89 kB/s | 110 kB     00:01
(12/12): edb-as17-server-pltcl-17.5.0-2.el9.x86_64.rpm       46 kB/s |  48 kB     00:01
--------------------------------------------------------------------------------
Total                                                        12 MB/s |  97 MB     00:07
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Installing       : edb-as17-server-libs-17.5.0-2.el9.x86_64               1/12
  Running scriptlet: edb-as17-server-libs-17.5.0-2.el9.x86_64               1/12
  Installing       : edb-as17-server-ecpg-17.5.0-2.el9.x86_64               2/12
  Installing       : edb-as17-server-client-17.5.0-2.el9.x86_64             3/12
  Running scriptlet: edb-as17-server-client-17.5.0-2.el9.x86_64             3/12
  Installing       : edb-as17-server-contrib-17.5.0-2.el9.x86_64            4/12
  Installing       : edb-as17-server-docs-17.5.0-2.el9.x86_64               5/12
  Installing       : beacon-agent-1.226.16-1.x86_64                         6/12
  Running scriptlet: edb-as17-server-core-17.5.0-2.el9.x86_64               7/12
  Installing       : edb-as17-server-core-17.5.0-2.el9.x86_64               7/12
  Running scriptlet: edb-as17-server-core-17.5.0-2.el9.x86_64               7/12
  Installing       : edb-as17-server-plperl-17.5.0-2.el9.x86_64             8/12
  Installing       : edb-as17-server-plpython3-17.5.0-2.el9.x86_64          9/12
  Installing       : edb-as17-server-pltcl-17.5.0-2.el9.x86_64             10/12
  Installing       : lz4-1.9.3-5.el9.x86_64                                11/12
  Installing       : edb-as17-server-17.5.0-2.el9.x86_64                   12/12
  Running scriptlet: edb-as17-server-17.5.0-2.el9.x86_64                   12/12
  Verifying        : lz4-1.9.3-5.el9.x86_64                                 1/12
  Verifying        : beacon-agent-1.226.16-1.x86_64                         2/12
  Verifying        : edb-as17-server-17.5.0-2.el9.x86_64                    3/12
  Verifying        : edb-as17-server-client-17.5.0-2.el9.x86_64             4/12
  Verifying        : edb-as17-server-contrib-17.5.0-2.el9.x86_64            5/12
  Verifying        : edb-as17-server-core-17.5.0-2.el9.x86_64               6/12
  Verifying        : edb-as17-server-docs-17.5.0-2.el9.x86_64               7/12
  Verifying        : edb-as17-server-ecpg-17.5.0-2.el9.x86_64               8/12
  Verifying        : edb-as17-server-libs-17.5.0-2.el9.x86_64               9/12
  Verifying        : edb-as17-server-plperl-17.5.0-2.el9.x86_64            10/12
  Verifying        : edb-as17-server-plpython3-17.5.0-2.el9.x86_64         11/12
  Verifying        : edb-as17-server-pltcl-17.5.0-2.el9.x86_64             12/12

Installed:
  beacon-agent-1.226.16-1.x86_64              edb-as17-server-17.5.0-2.el9.x86_64       edb-as17-server-client-17.5.0-2.el9.x86_64
  edb-as17-server-contrib-17.5.0-2.el9.x86_64 edb-as17-server-core-17.5.0-2.el9.x86_64  edb-as17-server-docs-17.5.0-2.el9.x86_64
  edb-as17-server-ecpg-17.5.0-2.el9.x86_64    edb-as17-server-libs-17.5.0-2.el9.x86_64  edb-as17-server-plperl-17.5.0-2.el9.x86_64
  edb-as17-server-plpython3-17.5.0-2.el9.x86_64 edb-as17-server-pltcl-17.5.0-2.el9.x86_64 lz4-1.9.3-5.el9.x86_64

Complete!
[admin@localhost ~]$
```

Figure 14 : Linux Terminal: Install the edb-as17-server software package

6. Install the EPEL repository.
   Use this command for EPEL installation.
   ```
   $ sudo dnf -y install epel-release
   ```

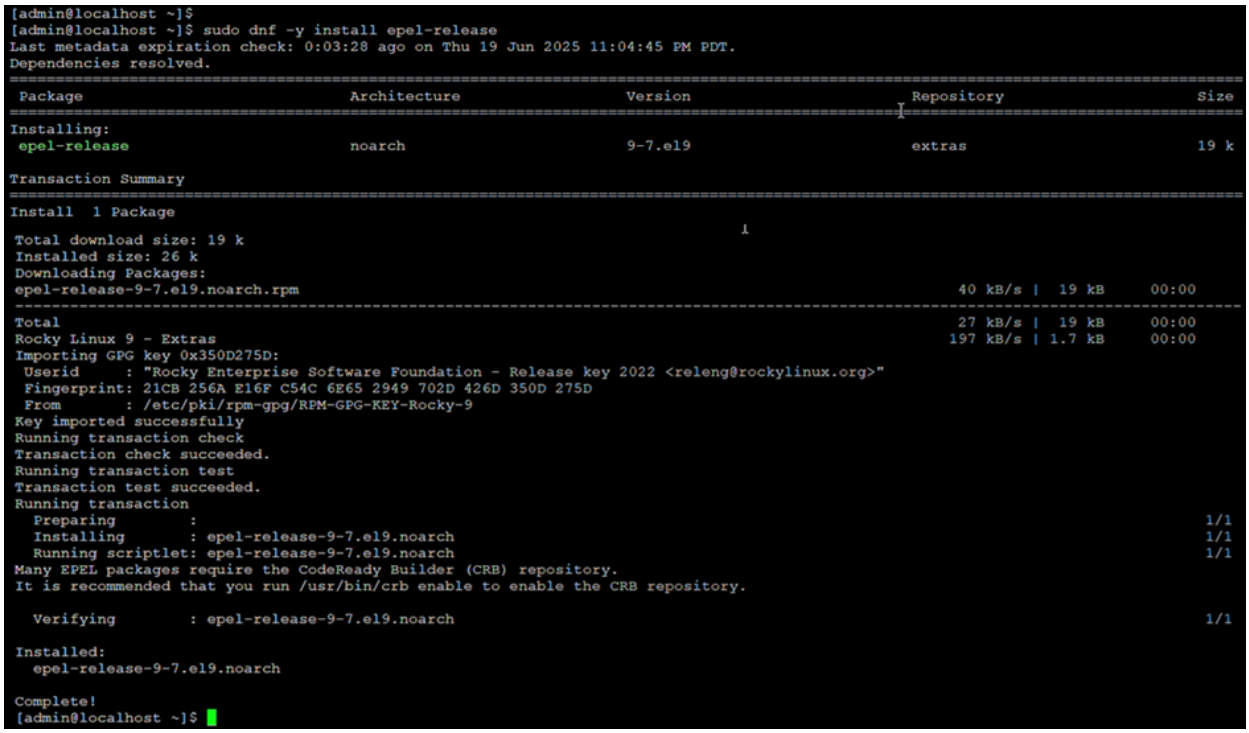

Figure 15 : Linux Terminal: installing EPEL repository

7. Enable additional repositories to resolve dependencies.
   Use this command to enable additional repositories.
   ```
   $ sudo dnf config-manager --set-enabled crb
   ```

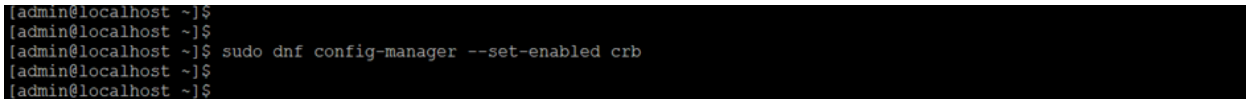

Figure 16 : Linux Terminal: enabling the additional repositories to resolve dependencies

8. Disable the built-in PostgreSQL module.
   Use this command to disable any existing PostgreSQL module.
   ```
   $ sudo dnf -qy module disable postgresql
   ```

9. Install the PostgreSQL packages
   Use this command to install PostgreSQL.
   ```
   $ sudo dnf -y install postgresql17-server postgresql17-contrib
   ```

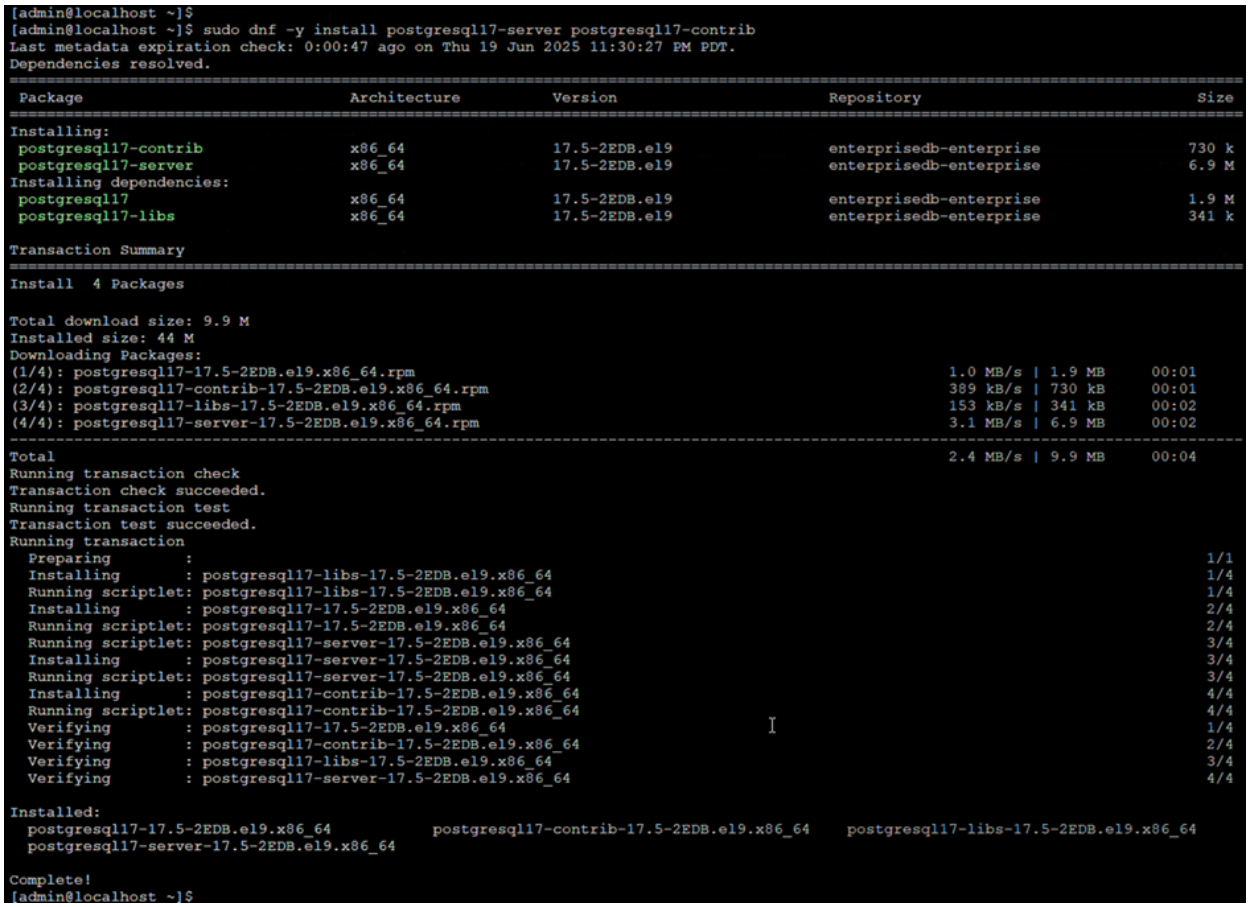

Figure 17 : Linux Terminal: installing the PostgreSQL package

10. Installing the server package creates an operating system user named `enterprisedb`. The user has no default password, so a password needs to be set using the `passwd` command.
`$ sudo passwd enterprisedb`

11. Make sure the user `enterprisedb` has sudo privileges.
`$ sudo usermod -aG wheel enterprisedb`

4.3 Install PyKMIP on the Postgres Linux server

PyKMIP is a Python implementation of the Key Management Interoperability Protocol. It is an open-source software. A Postgres Linux server can use it to interact with ESKM following the KMIP protocol.

PyKMIP must be installed as user `enterprisedb`. A user can be changed to `enterprisedb` with the command `sudo su - enterprisedb`.

4.3.1 Install PyKMIP Dependency Packages

1. Install Python on the Linux server if it has not already been installed.
   The `python` command can be used to verify the Python version.

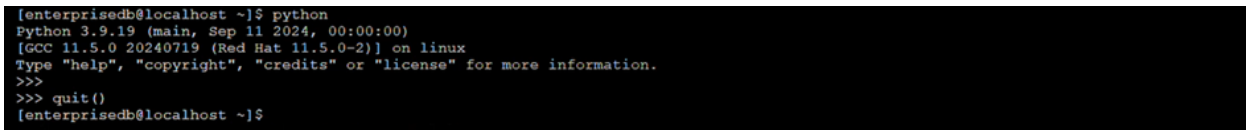

Figure 18 : Linux Terminal: Python version check

2. Install or upgrade PIP on the Linux server

   a. A PIP installation can be done using the following command.
      `$ sudo dnf install pip`

   b. A PIP pip upgrade can be done using the following command.
      `$ /usr/bin/python3.9 -m pip install --upgrade pip`

3. Upgrade *setuptools*.
   Use the following command to upgrade *setuptools*.
   $ `sudo pip3 install --upgrade setuptools`

4. Install tox.
   tox can be installed using the following command.
   `$ sudo pip3 install tox`

5. Install git.
   git can be installed using the following command.
   `$ sudo dnf install -y git`

6. Install *cryptography*.
   cryptography can be installed using the following command.
   `$ sudo pip3 install cryptography`

4.3.2 Build PyKMIP

1. Navigate to the home directory of the user `enterprisedb` using the `cd ~` command.

2. Clone the PyKMIP repository.
   `git clone https://github.com/openkmip/pykmip.git`

3. Check out the latest versions V0.10.0

   a. Change the directory to `pykmip`

   b. Check out v0.10.0: `git checkout tags/v0.10.0`

4. Install PyKMIP .
   `$ sudo python3.9 setup.py install`

5. Verify the installed PyKMIP version.
   `$ pip show pykmip`

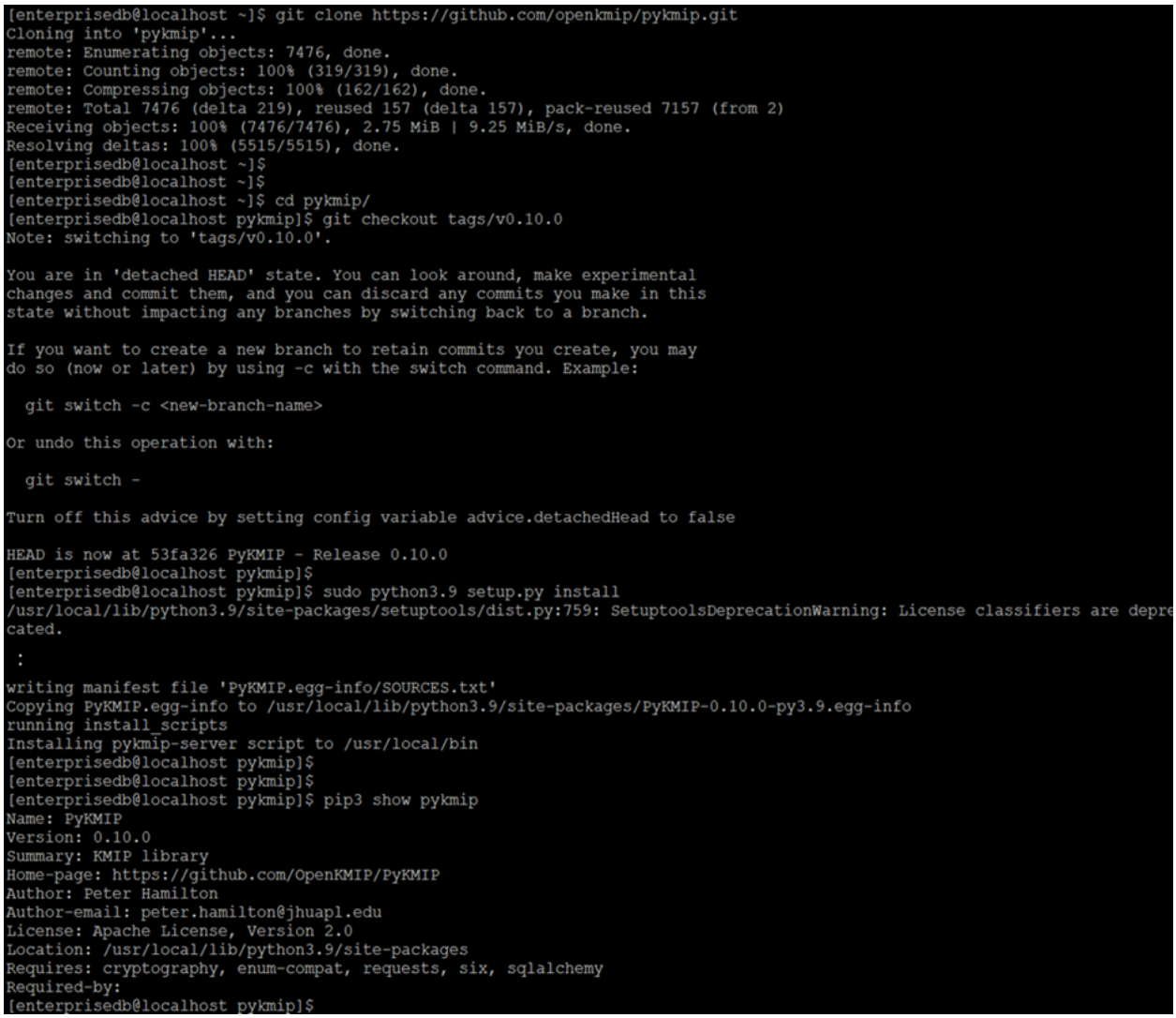

Figure 19 : Linux Terminal: PyKMIPinstallation

5 Integration Steps

One of the building blocks of data encryption in EDB Postgres is TDE (Transparent Data Encryption). It offers encryption at the file level, which solves the problem of protecting data at rest. The key for transparent data encryption is generated by `initdb` and stored in the file `pg_encryption/key.bin` under the data directory.

A wrap and an unwrap command need to be specified to secure the data encryption key, which provides TDE with a data encryption protection mechanism. The data encryption key will be protected using a wrapping key stored in a key management system. This second key is also called the key-wrapping key or master key. The Utimaco ESKM acts as the KMS here.

5.1 Establishing Trust Between the Utimaco ESKM and EDB Postgres

Certificates are required to facilitate the KMIP communications with the EDB Postgres and Utimaco ESKM.

5.1.1 Prepare Certificates Required for KMIP Communication

1. Generate a CSR on the Postgres Linux server.

    a. Change to the home directory of `enterprisedb`.

    b. Generate a private key using OpenSSL.
    ```
    $ openssl genrsa -out client1.key 2048
    ```

    c. Create a Certificate Signing Request (CSR) with the generated private key. Include necessary fields for CSR generation.
    ```
    $ openssl req -new -key client1.key -out client1.csr -sha256
    ```

    The figure below shows the generation of a private key `client1.key` and a CSR `client1.csr` with respective fields including `kmip_client_EDB` as `Common Name`.

```
[enterprisedb@localhost ~]$ cd ~
[enterprisedb@localhost ~]$
[enterprisedb@localhost ~]$ openssl genrsa -out client1.key 2048
[enterprisedb@localhost ~]$ ls client1.key
client1.key
[enterprisedb@localhost ~]$ cat << EOF | openssl req -new -key client1.key -out client1.csr -sha256
US
California
Campbell
Utimaco
Atalla
kmip_client_EDB
eskm@utimaco.com


EOF
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]:State or Province Name (full name) []:Locality Name (eg, city) [Default City]:Organizati
on Name (eg, company) [Default Company Ltd]:Organizational Unit Name (eg, section) []:Common Name (eg, your name or your s
erver's hostname) []:Email Address []:
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:An optional company name []:[enterprisedb@localhost ~]$
[enterprisedb@localhost ~]$
[enterprisedb@localhost ~]$ ls client1.csr
client1.csr
[enterprisedb@localhost ~]$
```

Figure 20 : Linux Terminal: CSR generation

2. Sign the CSR with the ESKM local CA and copy the signed certificate to the Postgres Linux server.
   The CSR now needs to be signed by the local CA.

   a. View the `client1.csr` file using the `cat` command `cat client1.csr` or open it using any text editor.

   b. Select the entire text and copy it to your clipboard.
   Make sure to include the first and last lines ("-----BEGIN CERTIFICATE REQUEST----- to -----END CERTIFICATE REQUEST-----").

   c. Log in to the Management Console and go to **Security > Certificates & CAs > Local CAs**.

   d. Select the CA used by your ESKM (in this case, ESKMLocalCA), and click on **Sign Request**. The Sign Request window will appear.

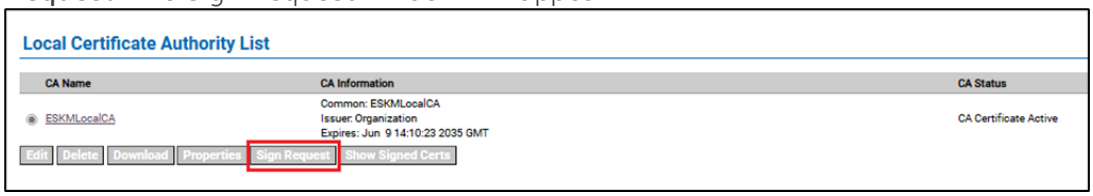

Figure 21 : ESKM server: Local CA sign request selection

   e. For certificate purpose, select **Client**.

   f. Paste the CSR text you copied into the **Certificate Request** window.

g. Click **Sign Request**.

Figure 22 : ESKM server: certificate signing with Local CA

h. The signed client certificate is displayed. Download the signed client certificate by clicking on **Download,** and copy it to the Postgres Linux server.

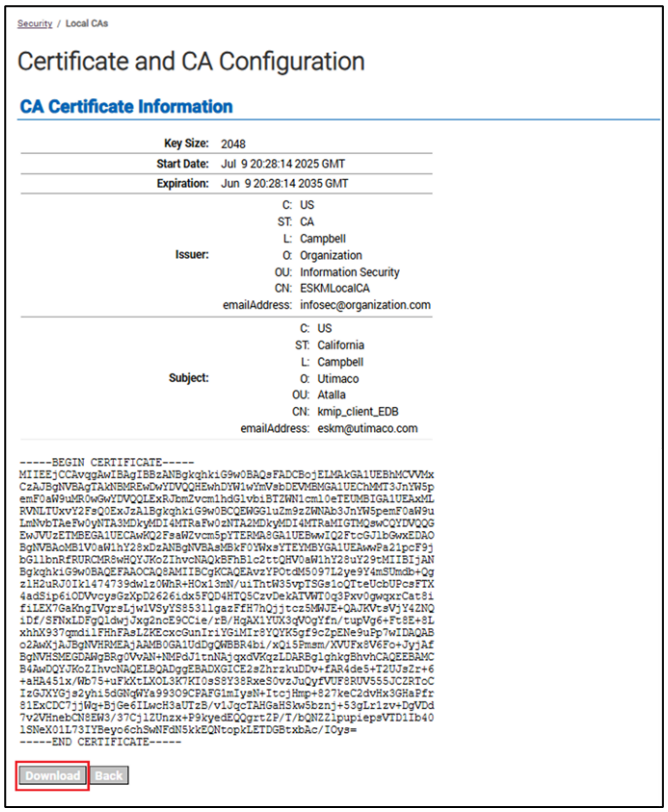

Figure 23 : ESKM server: signed certificate download

3. Copy the ESKM local CA to the Postgres Linux server.

   a. ESKM local CA (*ESKMLocalCA*) can be downloaded from the ESKM management console. Go to **Security > Certificates & CAs > Local CAs,** select the CA used by the ESKM (in this case, *ESKMLocalCA*), and click **Download**.

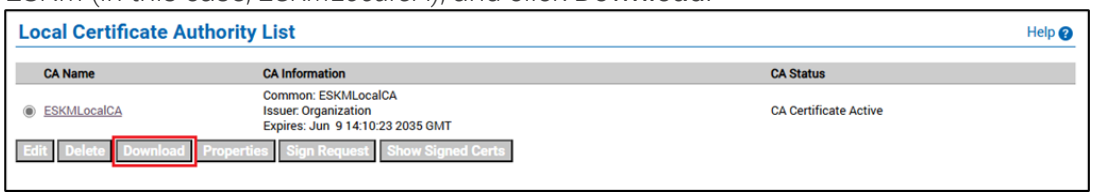

Figure 24 : ESKM server: Local CA download selection

   b. Copy the downloaded local CA to the Postgres Linux server.

4. Rename the ESKM local CA to `cacert.pem`.
   `$ cp ESKMLocalCA.crt cacert.pem`

5. Combine the private key and client certificate and rename to `client1.pem`.
   The created private key (`client1.key`) and the signed client certificate (`signed.crt`)

need to be combined and renamed to `client1.pem`.

```
$ cat client1.key signed.crt > client1.pem
```

5.1.2 Create a Local User

Perform the following steps to create a local user in the ESKM.

1. Go to the **Security** tab.
2. In the left side panel, click on **Local Users & Groups** under **Users & Groups**.
3. Click on **Local Users**.
4. Scroll down and click on **Add**.
5. Enter the **Common Name** (kmip_client_EDB) you provided during teh client certificate creation as **Username**.
6. Select **Enable KMIP**.
7. Select **KMIP** as **License Type**.
8. Paste the content of the downloaded client certificate into the **KMIP Client Certificate Contents** edit field.
9. Click on **Create**.

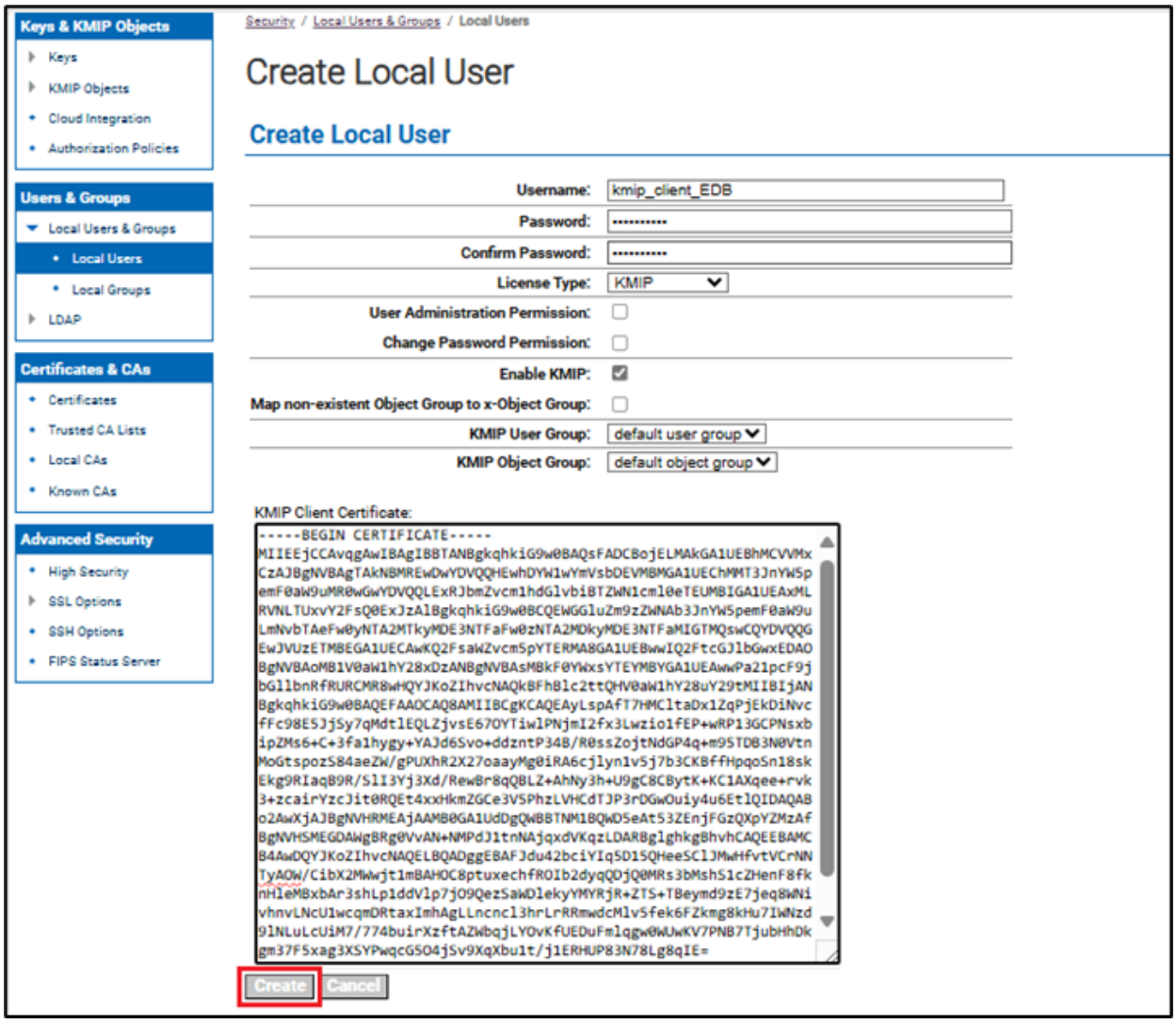

Figure 25 : ESKM server: Local user creation

10. The created user is displayed under **Local Users & Groups > Local Users.**

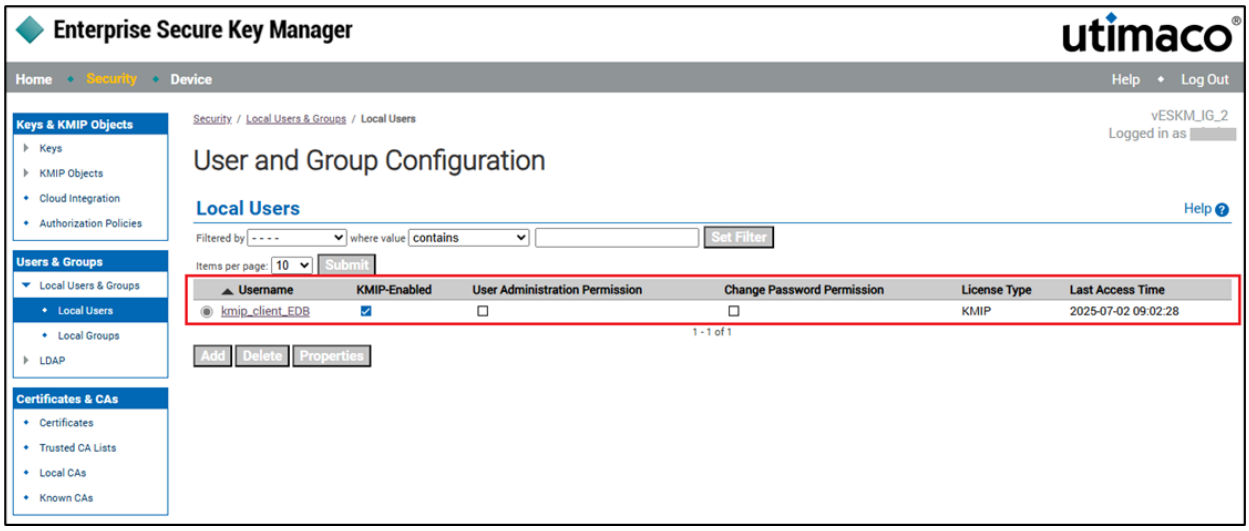

Figure 26 : ESKM server: created local user 'kmip_client_EDB'

5.2 Configure PyKMIP on the Postgres Linux Server

PyKMIP has to be configured to use the client certificate generated by ESKM.

1. Log in to the Postgres Linux server as user `enterprisedb`.
    a. Log in as user `enterprisedb` and go to the home directory of `enterprisedb`.
2. Install the KMIP client and the CA certificates.
    a. Create the `/etc/pykmip/certs` directory
    `$ sudo mkdir -p /etc/pykmip/certs`
    b. Copy the private key, and the client and CA certificates to `etc/pykmip/certs` and rename them as described here:
    `$ sudo cp client1.pem /etc/pykmip/certs/client_cert.pem`
    `$ sudo cp client1.key /etc/pykmip/certs/client_private_key.pem`
    `$ sudo cp cacert.pem /etc/pykmip/certs/server_ca_cert.pem`
    c. Assign read permission to all users of the private key, the client, and CA certificates.
    `$ sudo chmod a+r /etc/pykmip/certs/*`
3. Create/update the Python environment variable.

a. Open the `enterprisedb` user's `.bash_profile` file and add the following line to it:
`$ export PYTHONPATH=$HOME/pykmip`

b. Source the file so the environment variable becomes available:
`$ source ~/.bash_profile`

c. Check if the environment variable is set by executing the command below:.
`$ echo $PYTHONPATH`
Expected output: *var/lib/edb/pykmip*

4. Copy policy and configuration files to pykmip.

a. Copy the PyKMIP `policy.json` file to `/etc/pykmip/policy.json`.
`$ sudo cp ~/pykmip/examples/policy.json /etc/pykmip/policy.json`

b. Copy the PyKMIP `pykmip.conf` file to `/etc/pykmip/pykmip.conf`.
`$ sudo cp ~/pykmip/examples/pykmip.conf /etc/pykmip/pykmip.conf`

5. Edit the pykmip configuration file `/etc/pykmip/pykmip.conf`.

Fields	Field values
host	Provide ESKM IP address
port	No change required [5696].
keyfile	Private key is already copied.
certfile	client_cert is already copied.
cert_reqs	No change required.
ssl_version	Change protocol to `PROTOCOL_TLS`
ca_certs	CA cert is already copied.

do_handshake_on_connect	No change required.
suppress_ragged_eofs	No change required.
username	Comment this line.
password	Comment this line.

Table 5: pykmip.config file entries and changes to be made

5.3 Key Creation Script Modification for Key Activation

A Python script `create.py` for key creation is provided in `pykmip pykmip/kmip/demos/pie/create.py`. It needs to be modified to activate the key. The changes to be made are shown in the figure below.

```
[admin@localhost Source]$ diff -Nurb create.py Modified/create.py
--- create.py   2025-06-27 07:48:07.000000000 -0400
+++ Modified/create.py  2025-06-22 08:20:00.000000000 -0400
@@ -56,5 +56,6 @@
             )
             logger.info("Successfully created symmetric key with ID: "
                         "{0}".format(uid))
+            client.activate(uid)
         except Exception as e:
             logger.error(e)
[admin@localhost Source]$
```

Figure 27 : Diff of create.py file modification

5.4 Install edb-tde-kmip-client and Check Prerequisites

5.4.1 Install edb-tde-kmip-client

Install the `edb-tde-kmip-client` on the Postgres Linux server as user `enterprisedb`.

```
$ sudo dnf install -y edb-tde-kmip-client
```

```
[enterprisedb@localhost ~]$
[enterprisedb@localhost ~]$ sudo dnf install -y edb-tde-kmip-client
enterprisedb-enterprise                                          669  B/s | 659  B     00:00
enterprisedb-enterprise-noarch                                   812  B/s | 659  B     00:00
enterprisedb-enterprise-source                                   832  B/s | 659  B     00:00
Dependencies resolved.
================================================================================================================
 Package                    Architecture      Version              Repository                              Size
================================================================================================================
Installing:
 edb-tde-kmip-client        noarch            1.0-1.el9            enterprisedb-enterprise-noarch          14 k

Transaction Summary
================================================================================================================
Install  1 Package

Total download size: 14 k
Installed size: 20 k
Downloading Packages:
edb-tde-kmip-client-1.0-1.el9.noarch.rpm                         11 kB/s |  14 kB     00:01
----------------------------------------------------------------------------------------------------------------
Total                                                            11 kB/s |  14 kB     00:01
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                        1/1
  Installing       : edb-tde-kmip-client-1.0-1.el9.noarch                                   1/1
  Verifying        : edb-tde-kmip-client-1.0-1.el9.noarch                                   1/1

Installed:
  edb-tde-kmip-client-1.0-1.el9.noarch

Complete!
[enterprisedb@localhost ~]$
```

Figure 28 : Linux Terminal: install edb-tde-kmip-client

5.4.2 Check the Prerequisite

1. Key creation

    a. Create an AES 256 key using pykmip. The below steps are used for key creation.

    ```
    $ cd ~/pykmip/
    ```

    ```
    $ python3.9 ./kmip/demos/pie/create.py -a AES -l 256
    ```

```
[enterprisedb@localhost ~]$
[enterprisedb@localhost ~]$
[enterprisedb@localhost ~]$ cd ~/pykmip/
[enterprisedb@localhost pykmip]$ python3.9 ./kmip/demos/pie/create.py -a AES -l 256
2025-07-03 03:56:13,203 - demo - INFO - Successfully created symmetric key with ID: 12d8888e-a382-4762-b357-6050fff6f84e
[enterprisedb@localhost pykmip]$
[enterprisedb@localhost pykmip]$
```

Figure 29 : Linux server: key creation command execution

    b. Verify the created logs in ESKM. For the steps to verify ESKM logs, see Log location and interpretation.

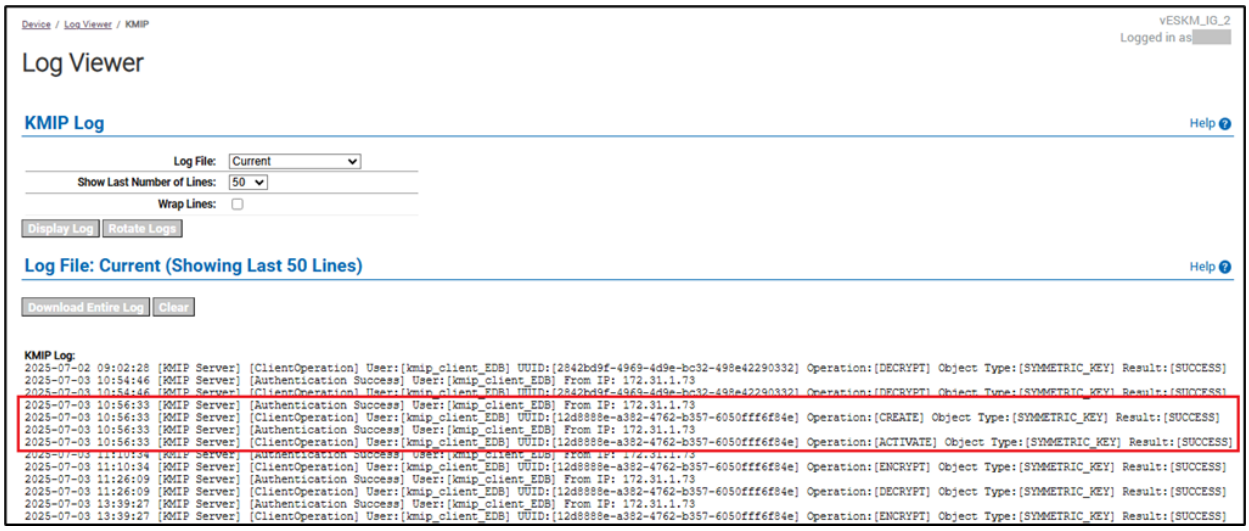

Figure 30 : ESKM server: KMIP logs with key creation

2. Verify encryption using the created key.
   Encrypt the string *utimaco* using the created key and store the cipher in `$HOME/test.bin`.

```
printf utimaco | python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py \
encrypt \
--out-file=$HOME/test.bin \
--pykmip-config-file=/etc/pykmip/pykmip.conf \
--key-uid='12d8888e-a382-4762-b357-6050fff6f84e' \
--variant=pykmip
```

3. Verify decryption with the same key.
   Decrypt the cipher in `$HOME/test.bin` using the same key.

```
python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py decrypt \
--in-file=$HOME/test.bin \
--pykmip-config-file=/etc/pykmip/pykmip.conf \
--key-uid='12d8888e-a382-4762-b357-6050fff6f84e' \
--variant=pykmip
```

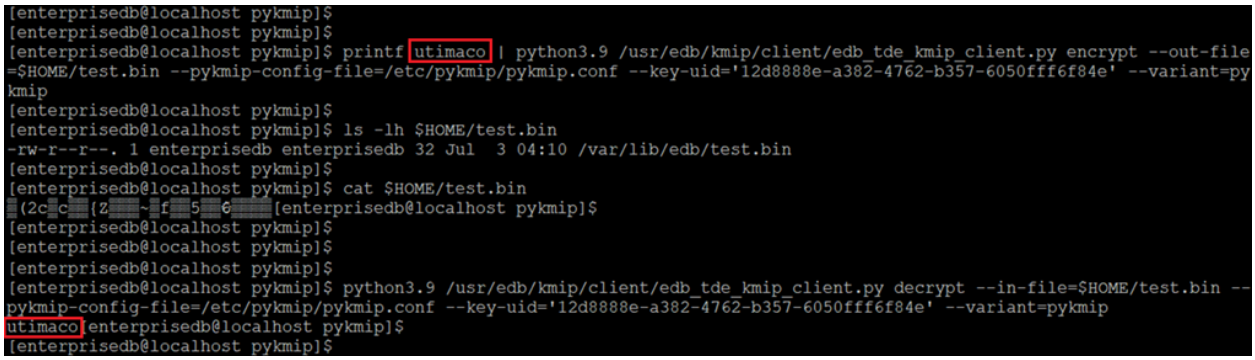

Figure 31 : Linux server: verifying encryption and decryption of a string

5.5 Perform the Initial Configuration and Create an Encrypted Postgres Database

1. Log in to the Postgres Linux server as user `enterprisedb`.

2. Go to the `/bin` directory. In this case the `/bin` directory path is `/usr/edb/as17/bin`.

3. Set the key wrap and key unwrap commands.

   a. Set `PGDATAKEYWRAPCMD`: shell command to encrypt the data encryption key.
   `export PGDATAKEYWRAPCMD='python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py \`
   `encrypt \`
   `--out-file=%p \`
   `--pykmip-config-file=/etc/pykmip/pykmip.conf \`
   `--key-uid="12d8888e-a382-4762-b357-6050fff6f84e" \`
   `--variant=pykmip'`

   b. Set `PGDATAKEYUNWRAPCMD`: shell command to decrypt the data encryption key when the database starts.
   `export PGDATAKEYUNWRAPCMD='python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py \`
   `decrypt \`

```
--pykmip-config-file=/etc/pykmip/pykmip.conf \
--key-uid="12d8888e-a382-4762-b357-6050fff6f84e" \
--in-file=%p --variant=pykmip'
```

c. Verify the key wrap and key unwrap variables set.

```
$ env | grep PGDATAKEY
```

```
[enterprisedb@localhost pykmip]$
[enterprisedb@localhost pykmip]$
[enterprisedb@localhost pykmip]$ export PGDATAKEYWRAPCMD='python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py encrypt --
out-file=%p --pykmip-config-file=/etc/pykmip/pykmip.conf --key-uid="12d8888e-a382-4762-b357-6050fff6f84e" --variant=pykmip
'
[enterprisedb@localhost pykmip]$
[enterprisedb@localhost pykmip]$ export PGDATAKEYUNWRAPCMD='python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py decrypt
--pykmip-config-file=/etc/pykmip/pykmip.conf --key-uid="12d8888e-a382-4762-b357-6050fff6f84e" --in-file=%p --variant=pykmi
p'
[enterprisedb@localhost pykmip]$
[enterprisedb@localhost pykmip]$ env | grep PGDATAKEY
PGDATAKEYWRAPCMD=python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py encrypt --out-file=%p --pykmip-config-file=/etc/pyk
mip/pykmip.conf --key-uid="12d8888e-a382-4762-b357-6050fff6f84e" --variant=pykmip
PGDATAKEYUNWRAPCMD=python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py decrypt --pykmip-config-file=/etc/pykmip/pykmip.c
onf --key-uid="12d8888e-a382-4762-b357-6050fff6f84e" --in-file=%p --variant=pykmip
[enterprisedb@localhost pykmip]$
[enterprisedb@localhost pykmip]$
```

Figure 32 : Linux server: key wrap and key unwrap command set

4. Perform the initial configuration of the database.
   The database configuration can be done with the following command.

```
$ /usr/edb/as17/bin/initdb -D /var/lib/edb/as17/data -y
```

```
[enterprisedb@localhost bin]$
[enterprisedb@localhost bin]$ /usr/edb/as17/bin/initdb -D /var/lib/edb/as17/data -y
The files belonging to this database system will be owned by user "enterprisedb".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.
Transparent data encryption is enabled (128 bits).

fixing permissions on existing directory /var/lib/edb/as17/data ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default "max_connections" ... 100
selecting default "shared_buffers" ... 128MB
selecting default time zone ... America/Los_Angeles
creating configuration files ... ok
setting up data encryption ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
creating edb sys ... ok
loading edb contrib modules ...

 :
 :
installing extension edb_dblink_libpq ... ok
installing extension edb_dblink_oci ... ok
snap_tables.sql
snap_functions.sql
dblink_ora.sql
sys_stats.sql
ok
finalizing initial databases ... ok
syncing data to disk ... ok

initdb: warning: enabling "trust" authentication for local connections
initdb: hint: You can change this by editing pg_hba.conf or using the option -A, or --auth-local and --auth-host, the next
 time you run initdb.

Success. You can now start the database server using:

    /usr/edb/as17/bin/pg_ctl -D /var/lib/edb/as17/data -l logfile start

[enterprisedb@localhost bin]$
```

Figure 33 : Linux server: initial configuration of the Postgres database

5. Start the database server.
   The database server can be started with the following command, and the output log can be passed to `$HOME/log`.
   `$ /usr/edb/as17/bin/pg_ctl -D /var/lib/edb/as17/data -l $HOME/logfile start`

```
[enterprisedb@localhost bin]$
[enterprisedb@localhost bin]$
[enterprisedb@localhost bin]$ /usr/edb/as17/bin/pg_ctl -D /var/lib/edb/as17/data -l $HOME/logfile start
waiting for server to start.... done
server started
[enterprisedb@localhost bin]$
[enterprisedb@localhost bin]$
```

Figure 34 : Linux server: Postgres database start

6. Verify the `data_encryption_key_unwrap_command` in the `postgresql.conf` file.
   The `data_encryption_key_unwrap_command` is set with `PGDATAUNWRAPCMD` should be present in the `/var/lib/edb/as17/data/postgresql.conf` file.

Figure 35 : Linux server: data_encryption_key_unwrap_command in postgresql.conf

7. Ensure encryption is enabled.
   Execute the following command and confirm '*Data encryption version*' and '*Data encryption key length*' are set.
   `$ /usr/edb/as17/bin/pg_controldata /var/lib/edb/as17/data`

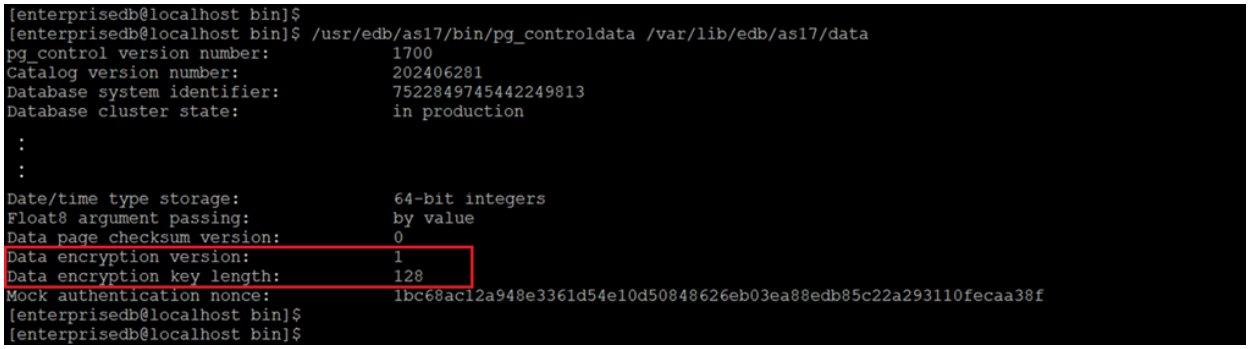

Figure 36 : Linux server: check if encryption is enabled

8. Create a database for the `enterprisedb` user to do the testing.
   The command for creating database `hr` as user `enterprisedb` is:
   `$ /usr/edb/as17/bin/createdb --owner enterprisedb hr`

9. Connect to the `hr` database in `psql`.
   `$ /usr/edb/as17/bin/psql hr`

10. Create columns.
    The tables are created with the `CREATE TABLE` command.
    Here is an example for creating table 'dept':
    ```
    hr=# CREATE TABLE public.dept (deptno numeric(2) NOT NULL CONSTRAINT
    dept_pk PRIMARY KEY, dname varchar(14) CONSTRAINT dept_dname_uq UNIQUE,
    loc varchar(13));
    ```

11. Insert values into the table.
    here is an example for inserting values into the table 'dept':
    ```
    hr=# INSERT INTO dept VALUES (10,'ACCOUNTING','NEW YORK');
    hr=# INSERT into dept VALUES (20,'RESEARCH','DALLAS';
    ```

12. View the table data.
    The table data can be viewed by selecting the values from the table with the command below.
    ```
    hr=# SELECT * FROM dept;
    ```

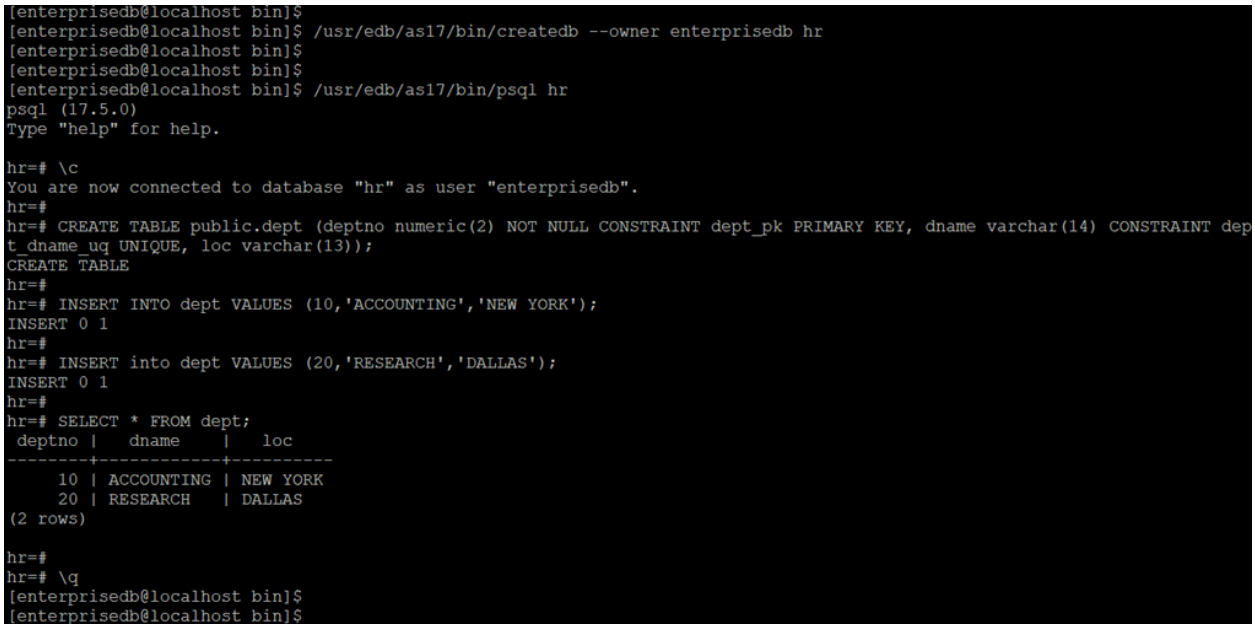

Figure 37 : Linux server: configure and create a database as an example

6 Verification and Testing

In this chapter, we will verify whether the integration between Utimaco ESKM and EDB Postgres Advanced Server is functioning as expected. This includes checking connectivity, validating encryption workflows, and ensuring both systems communicate correctly. By the end of this section, you should be able to confirm that the integration is successfully established and operational.

6.1 Restart the Postgres Database and Verify the ESKM Logs

1. Stop the Postgres database.
   `$ /usr/edb/as17/bin/pg_ctl -D /var/lib/edb/as17/data -l $HOME/logfile stop`

2. Start the Postgres database and view the database.
   `$ /usr/edb/as17/bin/pg_ctl -D /var/lib/edb/as17/data -l $HOME/logfile start`

```
[enterprisedb@localhost bin]$
[enterprisedb@localhost bin]$
[enterprisedb@localhost bin]$ /usr/edb/as17/bin/pg_ctl -D /var/lib/edb/as17/data -l $HOME/logfile stop
waiting for server to shut down.... done
server stopped
[enterprisedb@localhost bin]$
[enterprisedb@localhost bin]$
[enterprisedb@localhost bin]$ /usr/edb/as17/bin/pg_ctl -D /var/lib/edb/as17/data -l $HOME/logfile start
waiting for server to start..... done
server started
[enterprisedb@localhost bin]$
[enterprisedb@localhost bin]$ /usr/edb/as17/bin/psql hr
psql (17.5.0)
Type "help" for help.

hr=# SELECT * FROM dept;
 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
(2 rows)

hr=#
hr=# \q
[enterprisedb@localhost bin]$
```

Figure 38 : Linux server: steps for verification

3. Verify the ESKM KMIP logs.
   The steps for ESKM KMIP logs are mentioned in Log rotation and interpretation.
   The figure below shows that the master key is used for decryption when the database is started.

Figure 39 : ESKM server: ESKM KMIP logs of key used for decryption

6.2 Restart the Postgres Database when ESKM is Down

The key is fetched on the database's startup. We test the data encryption by the key stored in the ESKM by disabling the ESKM KMIP service and then attempting to start the database. Since the Postgres Linux server is unable to connect with the ESKM KMIP service, the system should fail and report an error.

1. Stop the Postgres database.
   Stop the database with this command.
   `$ /usr/edb/as17/bin/pg_ctl -D /var/lib/edb/as17/data -l $HOME/logfile stop`
   Postgres Linux server logs:
   ```
   [enterprisedb@localhost bin]$ /usr/edb/as17/bin/pg_ctl -D /var/lib/edb/as17/data -l $HOME/logfile  stop
   waiting for server to shut down.... done
   server stopped
   [enterprisedb@localhost bin]$
   ```

2. Stop the ESKM KMIP service.
   The ESKM KMIP service can be stopped from Utimaco ESKM by following these steps:

   a. Open the Utimaco ESKM page, log in, and click on the **Device** tab.

b. Click on **Services** under **Maintenance**.

c. Select **KMIP Server** under **Services Configuration**.

d. Click on **Stop**.

e. Click on **Confirm** on the **Confirmation Required** page.

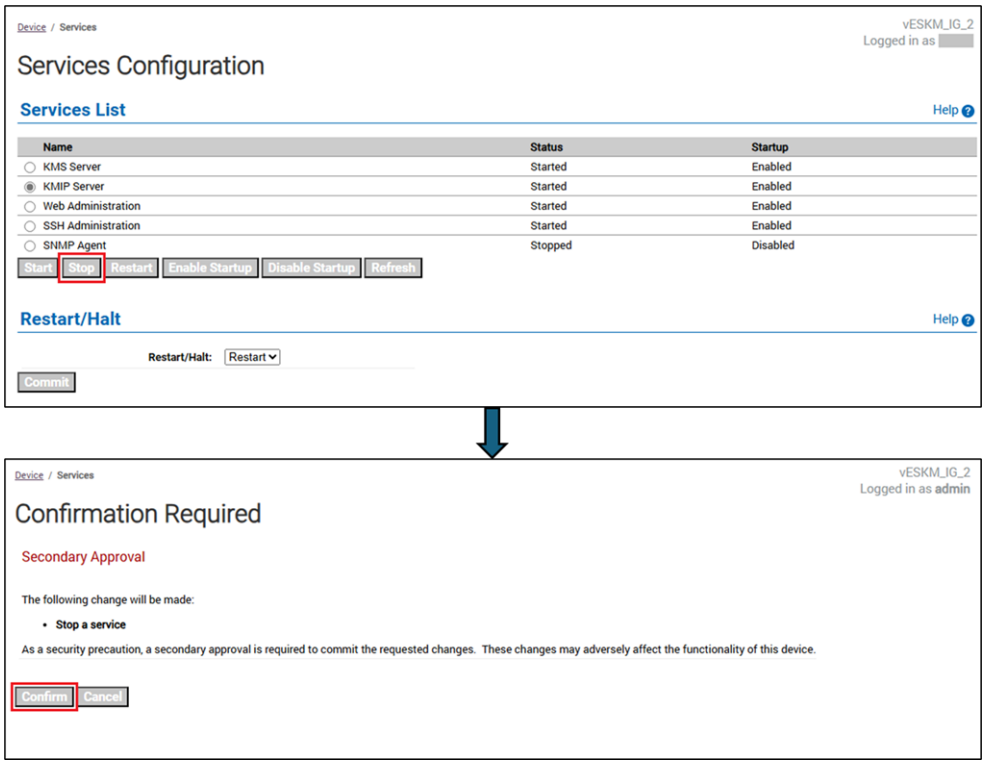

Figure 40 : ESKM server: screen flow of stopping the KMIP service

3. Start the Postgres database.

```
$ /usr/edb/as17/bin/pg_ctl -D /var/lib/edb/as17/data -l $HOME/logfile
start
```

Postgres Linux server logs:

```
[enterprisedb@localhost bin]$ /usr/edb/as17/bin/pg_ctl -D /var/lib/edb/
as17/data -l $HOME/logfile start
waiting for server to start.... stopped waiting
pg_ctl: could not start server
Examine the log output.
[enterprisedb@localhost bin]$
```

4. Verify the Postgres database log file.
   The log file is `$HOME/logfile`. The log confirms that the Postgres Linux server cannot reach the ESKM server, so the database cannot start.

   Postgres Linux server logs:

```
[enterprisedb@localhost bin]$ cat $HOME/logfile
An error occurred while connecting to appliance 172.31.1.82: [Errno 111] Connection refused
could not open client connection: [Errno 111] Connection refused
Traceback (most recent call last):
File "/usr/edb/kmip/client/edb_tde_kmip_client.py", line 111, in <module>
main()
File "/usr/edb/kmip/client/edb_tde_kmip_client.py", line 69, in main
with pykmip_client:
File "/var/lib/edb/pykmip/kmip/pie/client.py", line 1745, in __enter__
self.open()
File "/var/lib/edb/pykmip/kmip/pie/client.py", line 173, in open
self.proxy.open()
File "/var/lib/edb/pykmip/kmip/services/kmip_client.py", line 285, in open
six.reraise(*last_error)
File "/usr/lib/python3.9/site-packages/six.py", line 709, in reraise
raise value
File "/var/lib/edb/pykmip/kmip/services/kmip_client.py", line 274, in open
self.socket.connect((self.host, self.port))
File "/usr/lib64/python3.9/ssl.py", line 1376, in connect
self._real_connect(addr, False)
File "/usr/lib64/python3.9/ssl.py", line 1363, in _real_connect
super().connect(addr)
ConnectionRefusedError: [Errno 111] Connection refused
2025-07-03 07:42:42 PDT FATAL:  could not run command "python3.9   /usr/edb/kmip/client/edb_tde_kmip_client.py decrypt --pykmip-config-file=/etc/pykmip/pykmip.conf --key-uid="12d8888e-a382-4762-b357-6050fff6f84e" --in-file=pg_encryption/key.bin --variant=pykmip": child process exited with exit code 1
```

```
2025-07-03 07:42:42 PDT LOG:  database system is shut down
[enterprisedb@localhost bin]$
```

5. Start the ESKM KMIP service.
   We can stop the ESKM KMIP service from Utimaco ESKM by following these steps:

   a. Open the Utimaco ESKM page, log in, and click on the **Device** tab.

   b. Click on **Services** under **Maintenance**.

   c. Select **KMIP Server** under **Services Configuration**.

   d. Click on **Start**.

   e. Click on **Confirm** on the **Confirmation Required** page.

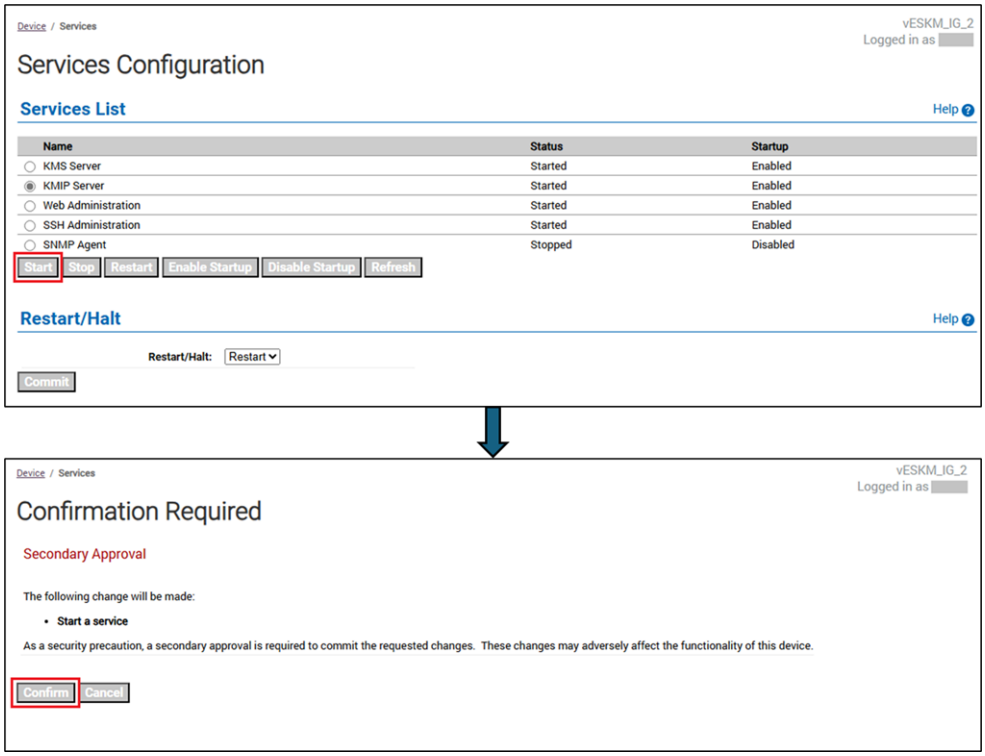

Figure 41 : ESKM server: screen flow of starting the KMIP service

6. Start the Postgres database.
   The database can be started using this command.
   `$ /usr/edb/as17/bin/pg_ctl -D /var/lib/edb/as17/data -l $HOME/logfile start`
   The logs confirm that the database start was successful.

   Postgres Linux server logs:

```
[enterprisedb@localhost bin]$ /usr/edb/as17/bin/pg_ctl -D /var/lib/edb/
as17/data -l $HOME/logfile start
waiting for server to start.... done
server started
[enterprisedb@localhost bin]$
```

7. Verify the ESKM KMIP logs.
   See Log location and interpretation. The figure shows that the master key is used for decryption after the KMIP service and database are started.

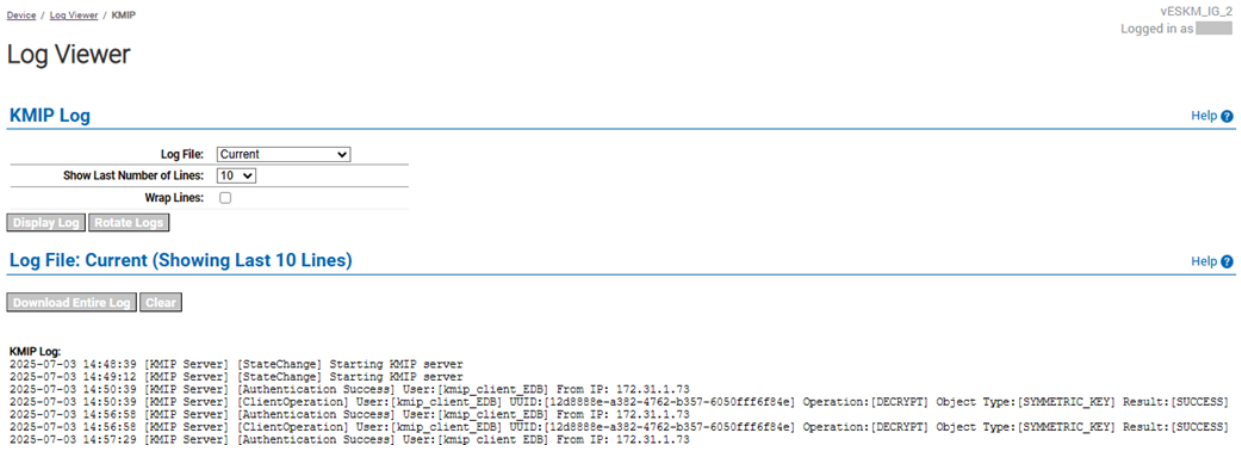

Figure 42 : ESKM server: ESKM KMIP logs of the key used for decryption

6.3 Key Rotation

Key rotation can be verified by changing the master key. A new key will be created, manually run the unwrap command specifying the old key and then feed the result into the wrap command specifying the new key. These operations can be performed when the database is running.

1. Create a new master key using pykmip.
   New key can be created by executing the *create.py* script.

   ```
   $ cd ~/pykmip/
   $ python3.9 ./kmip/demos/pie/create.py -a AES -l 256
   ```

   Postgres Linux server logs:

   ```
   [enterprisedb@localhost bin]$ cd ~/pykmip/
   [enterprisedb@localhost pykmip]$ python3.9 ./kmip/demos/pie/create.py -a
   ```

```
AES -l 256
2025-07-03 07:57:09,343 - demo - INFO - Successfully created symmetric
key with ID: 441555e4-e977-48bd-a710-d9942daf0fa3
[enterprisedb@localhost pykmip]$
```

2. Save the original `key.bin` in the `pg_encryption` directory.
   ```
   $ cd $PGDATA/pg_encryption
   $ cp key.bin key.bin.original
   ```

3. Run the `unwrap` command specifying the old key. Then feed the result into the `wrap` command, specifying the new key.
   This is the command with old key `uuid` and new key `uuid`:
   ```
   $ python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py decrypt --
   pykmip-config-file=/etc/pykmip/pykmip.conf --key-uid="12d8888e-
   a382-4762-b357-6050fff6f84e" --in-file=key.bin --variant=pykmip |
   python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py encrypt --out-
   file=key.bin --pykmip-config -file=/etc/pykmip/pykmip.conf --key-
   uid="441555e4-e977-48bd-a710-d9942daf0fa3" --variant=pykmip
   ```

4. Edit the `data_encryption_key_unwrap_command` with the new key in the `postgresql.conf` file.
   Update the new key `uuid` for the `data_encryption_key_unwrap_command` in `postgresql.conf` file.

   Linux Postgres server logs to confirm changes:
   ```
   [enterprisedb@localhost pg_encryption]$ grep -rn
   "data_encryption_key_unwrap_command" /var/lib/edb/as17/data/
   postgresql.conf
   124:#data_encryption_key_unwrap_command = 'python3.9 /usr/edb/kmip/
   client/edb_tde_kmip_client.py decrypt --pykmip-config-file=/etc/pykmip/
   pykmip.conf --key-uid="12d8888e-a382-4762-b357-6050fff6f84e" --in-
   file=%p --variant=pykmip'
   126:data_encryption_key_unwrap_command = 'python3.9 /usr/edb/kmip/
   client/edb_tde_kmip_client.py decrypt --pykmip-config-file=/etc/pykmip/
   pykmip.conf --key-uid="441555e4-e977-48bd-a710-d9942daf0fa3" --in-
   ```

```
file=%p --variant=pykmip'
[enterprisedb@localhost pg_encryption]$
```

5. Restart the Postgres database.
   Stop and start the Postgres database. Both operations should be successful.

   Linux Postgres server logs:

```
[enterprisedb@localhost]$ /usr/edb/as17/bin/pg_ctl -D /var/lib/edb/as17/
data -l $HOME/logfile stop
waiting for server to shut down.... done
server stopped
[enterprisedb@localhost]$
[enterprisedb@localhost]$ /usr/edb/as17/bin/pg_ctl -D /var/lib/edb/as17/
data -l $HOME/logfile start
waiting for server to start.... done
server started
[enterprisedb@localhost]$
```

6. View the Postgres database.
   Connect the database `hr` with `/usr/edb/as17/bin/psql hr`.

   Linux Postgres server logs:

```
[enterprisedb@localhost pg_encryption]$ /usr/edb/as17/bin/psql hr
psql (17.5.0)
Type "help" for help.

hr=#
hr=# SELECT * FROM dept;
deptno |   dname    |   loc
--------+------------+----------
    10 | ACCOUNTING | NEW YORK
   20 | RESEARCH   | DALLAS
(2 rows)
hr=#
hr=# \q
[enterprisedb@localhost pg_encryption]$
```

7. Verify the ESKM KMIP logs.
   See Log location and interpretation. The figure shows that the new master key is used for decryption when the database is started.

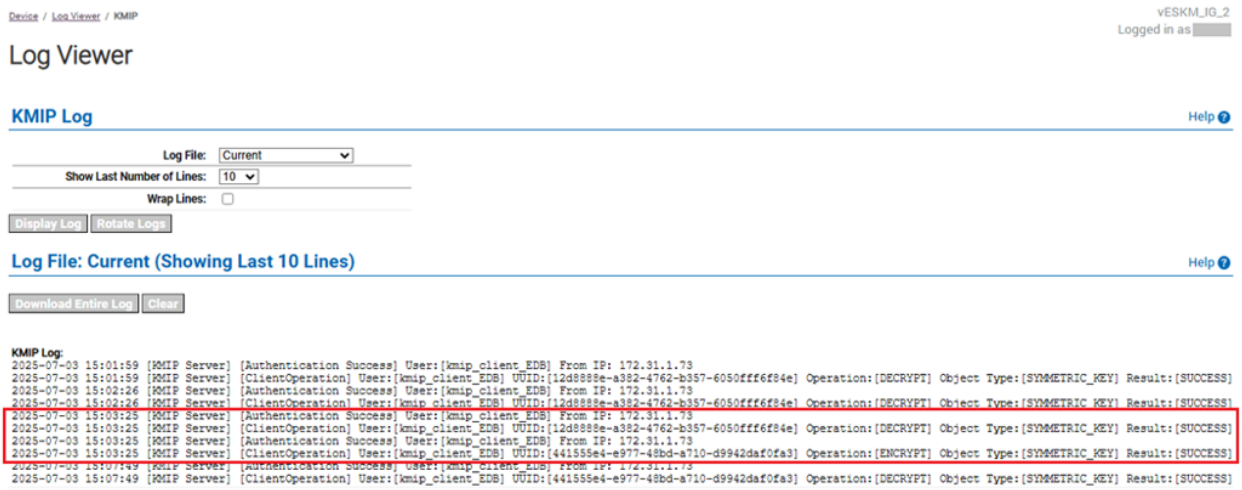

Figure 43 : ESKM server: KMIP log of new key used for decryption

6.4 Log Locations and Interpretation

Verify the logs from Utimaco ESKM by following these steps.

1. Open the Utimaco ESKM page and click on the **Device** tab.
2. Click on **Log Viewer** under **Logs & Statistics**.
3. Click on **KMIP** under **Log Viewer**.
4. Review logs related to the encryption and decryption operations performed.

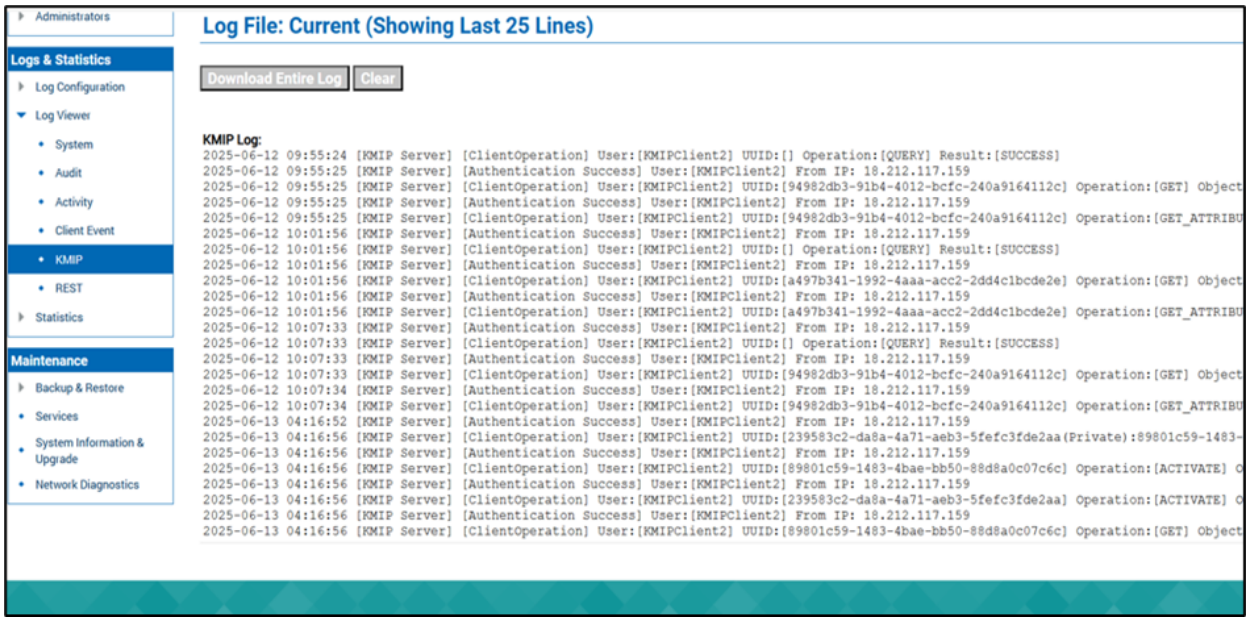

Figure 44 : ESKM server: ESKM KMIP logs

7 Troubleshooting

7.1 Common Issues

If the KMIP server certificate configured in ESKM does not meet the requirements mentioned in the prerequisites, then the KMIP connection will fail.

1. Verify the certificates used for KMIP communication. If the KMIP server certificate configured in ESKM does not meet the requirements mentioned in the prerequisites, the KMIP connection will fail.

2. Make sure the ESKM is accessible from the Postgres Linux server.
    - Use the `ping` command for a ESKM connectivity check.
        - `ping <ESKM ip address>`
    - Use the `netcat` command to check connectivity to the ESKM IP and port.
        - `nc -zv <ESKM ip address> 5696`

7.2 Contact and Support Information

7.2.1 Utimaco Technical Support

For technical questions, contact Utimaco Technical Support:

- E-mail: support-atalla@utimaco.com
- Telephone: 800-500-7858 (U.S.A.) +1-916-414-0216 (International)
- Website: https://support.utimaco.com/

Before contacting Utimaco with your questions, collect the following information:

- Product model names and numbers
- Technical support registration number or NonStop system number (if applicable)
- Service Agreement ID number (SAID)
- Product serial numbers

- Error messages
- Software version number

7.2.2 24-hour Support

24-hour emergency support is available to those customers who have valid service contracts. Use this service for product and system emergencies that occur after normal working hours or on weekends and U.S. holidays. Questions about product installation and setup are supported during normal working hours.

For 24-hour emergency support call: 800-500-7858 (U.S.A.) +1-916-414-0216 (International).

8 Appendices

8.1 References

This document serves as a comprehensive guide for integrating Utimaco's ESKM module with EDB Postgres Advanced Server.

For more information on other Utimaco products and offerings, please visit the official *Utimaco Website*.