Fortanix®

Integration Guide

USING FORTANIX DATA SECURITY MANAGER WITH MINIO (KES SERVER)

VERSION 1.0

Fortanix, Inc. | 3910 Freedom Circle | Suite 104 | Santa Clara, CA 95052 | United States of America | +1 (650) 943-2484 |
info@fortanix.com | www.fortanix.com

TABLE OF CONTENTS

2.0 INTRODUCTION

This article describes how to integrate **Fortanix Data Security Manager (DSM)** with MinIO's **Key Encryption Service (KES) server** that uses Fortanix DSM as a persistent and secure key store. KES server runs inside Kubernetes and distributes cryptographic keys to Fortanix DSM applications. This article also contains the information that a user needs to:

- Configure Fortanix DSM
- Set up KES server

3.0 TERMINOLOGY REFERENCES

- **Fortanix Data Security Manager (DSM) –**

  Fortanix DSM is the cloud solution secured with Intel® SGX. With Fortanix DSM, you can securely generate, store, and use cryptographic keys and certificates, as well as secrets, such as passwords, API keys, tokens, or any blob of data.

- **KES – Key Encryption Service**

  KES Service is a simple, stateless, and distributed key management system for high-performance applications. It is designed to be run inside Kubernetes and distribute cryptographic keys to applications. It acts as a bridge between a central Key Management Service (KMS) and cloud-native applications.

4.0 ARCHITECTURE

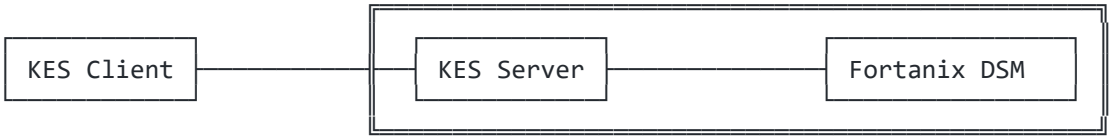

FIGURE 1: KES WITH DSM ARCHITECTURE

KES Server acts as a bridge between the Fortanix DSM and cloud-native applications.

Here Fortanix DSM is the central KMS that protects the master keys and acts as the root of trust in your infrastructure. Instead of deploying and managing one KMS per set of applications, when an

application wants to encrypt data, it can request a new DEK from a KES server or ask the KES server to decrypt an encrypted DEK. This way the load on the central KMS (Fortanix DSM) does not increase much because KES can serve the vast majority of application requests without talking to Fortanix DSM.

5.0 FORTANIX DATA SECURITY MANAGER CONFIGURATION

5.1 CREATE APPLICATION

First, register a new application that can authenticate and communicate to the Fortanix DSM instance. To do that,

1. Go to the **Apps** page in the Fortanix DSM UI and create a new App.

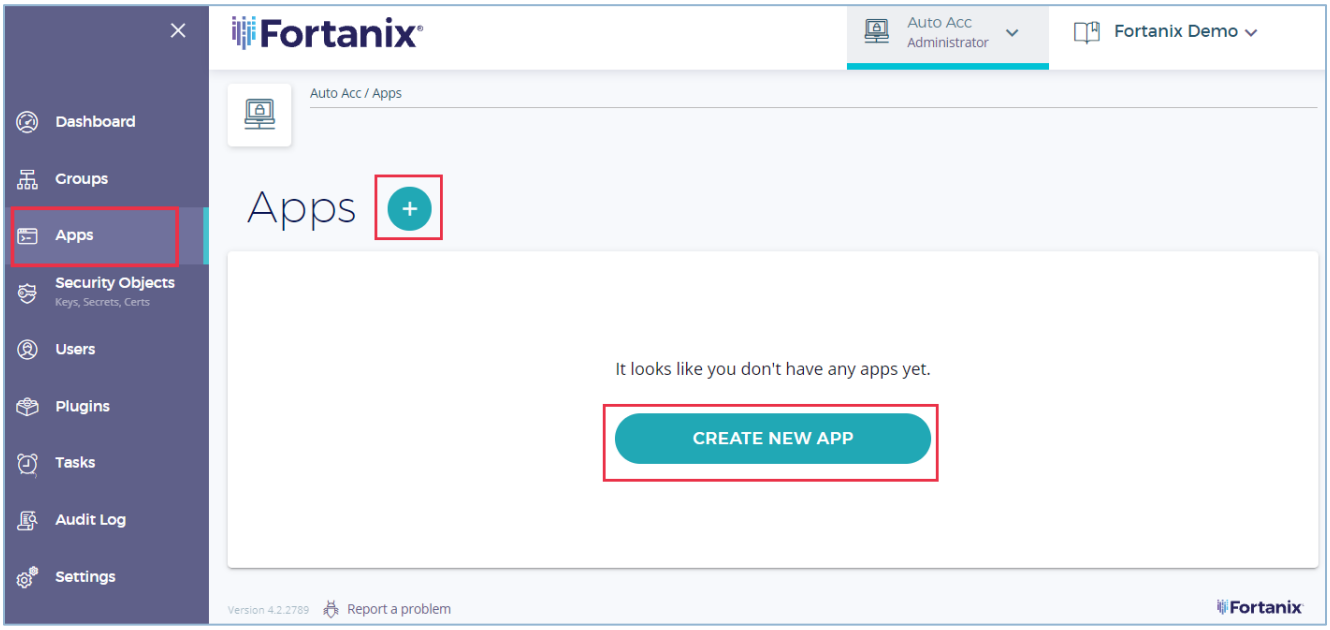

FIGURE 2: CREATE NEW APP

2. Enter a descriptive name for the app. For example, **KES.**
3. Select **REST** as the **Interface** and select **API Key** as the **Authentication method**.

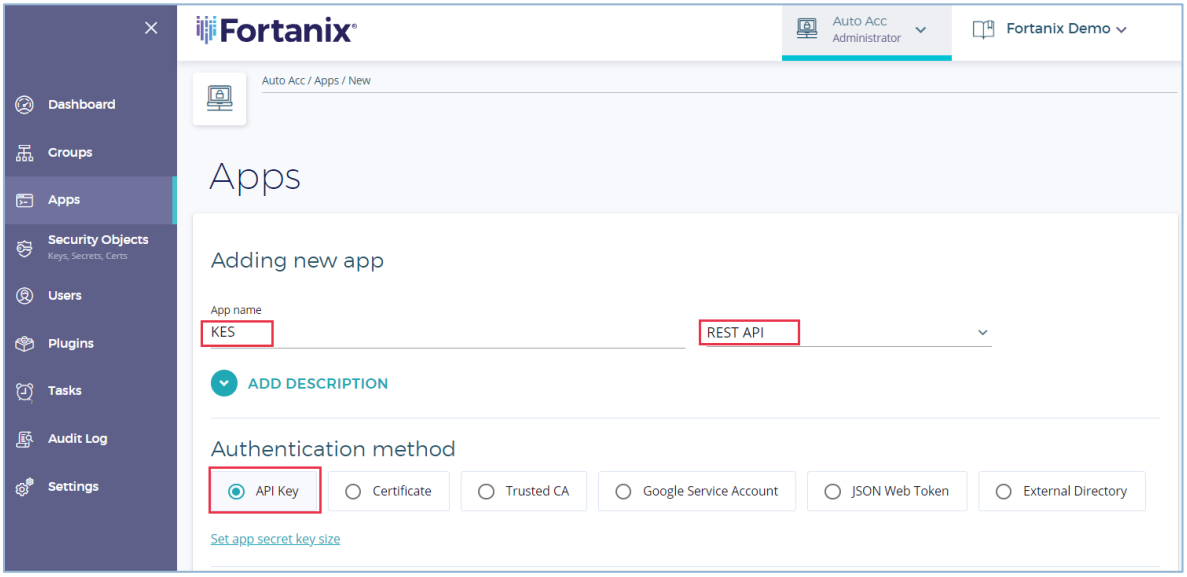

FIGURE 3: CREATE APP

5.2 ASSIGN APP TO GROUP

1. Next, assign the application to a group. This group will be the default group of the application. The newly created keys will belong to this group unless an explicit group ID is specified in the KES configuration file.

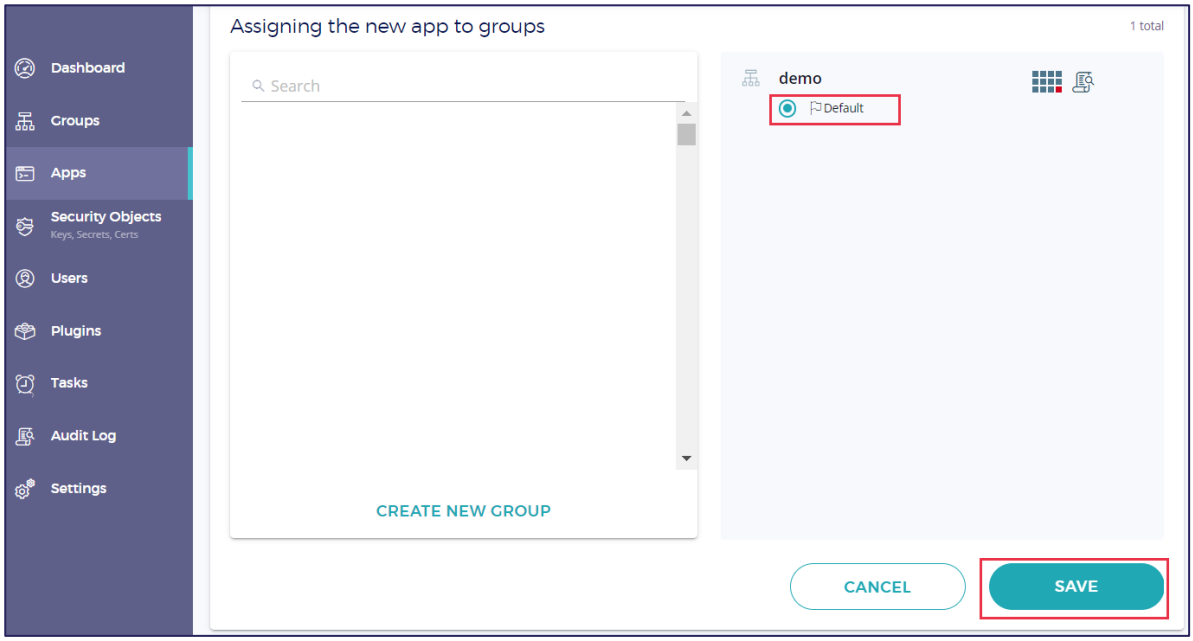

FIGURE 4: ASSIGN APP TO GROUP

2. Click **SAVE** to complete creating the application.

3. Click **COPY API KEY** to copy the application's API key. This key is the access credential to talk to Fortanix DSM as the application.

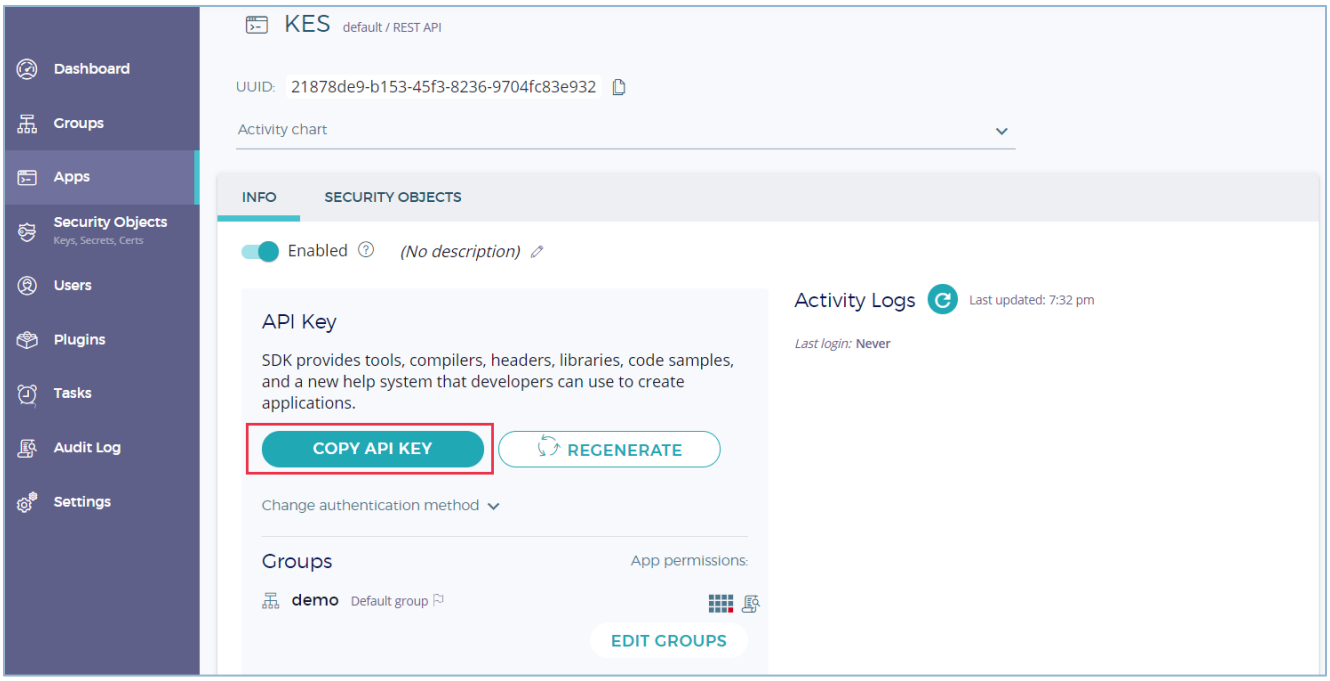

FIGURE 5: COPY API KEY

6.0 KES SERVER SETUP

First, you need to generate a TLS private key and certificate for the KES server. A KES server can only be run with TLS - since secure-by-default. Here we use self-signed certificates for simplicity. For a production setup, we highly recommend using a certificate signed by CA (for example, your internal CA or a public CA like Let's Encrypt).

6.1 GENERATE A TLS PRIVATE KEY AND CERTIFICATE FOR THE KES SERVER

The following command will generate a new TLS private key `server.key` and a X.509 certificate `server.cert` that is self-signed and issued for the IP `127.0.0.1` and DNS name `localhost` (as SAN). You may want to customize the command to match your setup.

```
kes tool identity new --server --key server.key --cert server.cert --ip
"127.0.0.1" --dns localhost
```

Any other tooling for X.509 certificate generation works as well. For example, you could use `openssl`:

```
$ openssl ecparam -genkey -name prime256v1 | openssl ec -out server.key

$ openssl req -new -x509 -days 30 -key server.key -out server.cert \
   -subj "/C=/ST=/L=/O=/CN=localhost" -addext "subjectAltName =
IP:127.0.0.1"
```

6.2 CREATE PRIVATE KEY AND CERTIFICATE FOR YOUR APPLICATION

```
kes tool identity new --key=app.key --cert=app.cert app
```

You can compute the app identity using:

```
kes tool identity of app.cert
```

6.3 CREATE CONFIGURATION FILE

Now, you have defined all entities in your demo setup. Wire everything together by creating the config file `server-config.yml`:

```
 address: 0.0.0.0:7373
 root:    disabled  # We disable the root identity since we don't need
 it in this guide

 tls:
 key : server.key
 cert: server.cert

 policy:
   my-app:
      allow:
```

```
      - /v1/key/create/my-app*
      - /v1/key/generate/my-app*
      - /v1/key/decrypt/my-app*
    identities:
    - ${APP_IDENTITY}

  keystore:
    fortanix:
      sdkms:
        endpoint: "<your-fortanix-sdkms-endpoint>"    # Use your
Fortanix instance endpoint.
        credentials:
         key: "<your-api-key>" # Insert the application's API key
```

6.4 START THE KES SERVER

Finally, start the KES Server in a new window/tab.

```
export APP_IDENTITY=$(kes tool identity of app.cert)

kes server --config=server-config.yml --auth=off
```

Where, `--auth=off` is required since your `root.cert` and `app.cert` certificates are self-signed.

6.5 CONNECT TO THE SERVER

In the previous window/tab, you can now connect to the server using the following commands:

```
export KES_CLIENT_CERT=app.cert
export KES_CLIENT_KEY=app.key
kes key create -k my-app-key
```

Where, `-k` is required because your are using self-signed certificates.

6.6 DECRYPT DATA ENCRYPTION KEYS

Finally, you can derive and decrypt the data keys from the previously created `my-app-key`.

```
kes key derive -k my-app-key
{
     plaintext : ...
     ciphertext: ...
}
```

```
kes key decrypt -k my-app-key <base64-ciphertext>
```

7.0 DOCUMENT INFORMATION

7.1 DOCUMENT LOCATION

The latest published version of this document is located at the URL:

https://support.fortanix.com/hc/en-us/articles/4408466794772-Using-Fortanix-Data-Security-Manager-with-MinIO-KES-Server-

7.2 DOCUMENT UPDATES

This document will typically be updated on a periodic review and update cycle.

For any urgent document updates, please send an email to: support@fortanix.com