SAMSUNG ELECTRONICS

Knox E-FOTA On-Premises

Installation and Initial Operation Guide

Version : 1.1

Last Update : Apr 2023

【Document History】

What	*Ver.*	*When*
I. Added: 5.7 Configurable device group polling	**Ver1.1**	**Apr 2023**
Initial Release	**Ver1.0**	**June 2020**

Table of Contents

Tables of Figures & Tables

【Figures】

【Tables】

PART I: Getting Started

PART 1: Getting Started presents the purpose of this document, what customer infrastructure is recommended prior to the installation of the Knox E-FOTA On-Premises service, and provides an overview of deliverables that will be used during the installation.

1. Introduction

1.1. Purpose of this document

The purpose of this document is to present how to plan for, install, and configure the managed DFM module within the customer's network. This document includes information about how to install and configure the 3rd party software, such as Docker, and provides detailed descriptions of the commands used to perform its installations.

This document is intended **for the personnel who are in charge of performing the installation**.

In order to prepare the installer, this document includes the following tasks:

1.1.1 Evaluate the customer's network and hardware facilities

1.1.2 Introduce which modules will be installed to provide this service

1.1.3 Explain the install flow with DFM Modules

1.1.4 Explain how to configure the installed DFM Modules with the proper conditions

1.1.5 Explain how to test if the installed DFM Modules are running as expected

The server infrastructure, hereafter referred to as **DFM Modules**, will be installed on the customer's side by Samsung to service the Knox E-FOTA On-Premises environment.

We recommend "The 4-Days Installation" for this installation, as the customer should understand how they are using this service during this program (see "*Appendix D. A Recommended Schedule for On-Site Installation by CSO/TEO*").

2. Environment Prerequisites

This chapter presents the hardware, software and network facilities required by the DFM. To ensure proper support of E-FOTA On-Premise, the service must be installed upon the following recommended software and hardware infrastructure.

The following recommended items should be prepared by the customer prior to the installation of the Knox E-FOTA On-Premises service by Samsung personnel.

2.1. Hardware Recommended

The recommended user environment, including the network card, for the On-Premises Hardware (H/W) requirements are as follows (the customer can choose the correct value depending on the product type. See "***2.3 Recommendation Per each Product Usage***"):

Server	Items	Recommended value	Descripstion
WEB	**Server CPU Cores**	Above 1 or 2 CPU Cores	1 Cores is for PoC Product Above 2 Cores is for Commercial Product
	RAM	4 or 8 GB	4GB is for PoC Product 8GB is for Commercial Product
	Disk	128 GB or 256 GB SSD	For DFM Module 128GB (PoC), 256GB (Commercial Product)
		256 GB	For System region (OS and Rootfilesystem)
	Network Card	Above 10 Gbps	
DFM Core/Console	**Server CPU Cores**	Above 2 or 4 CPU Cores	2 Cores is for PoC Product Above 4 Cores is for Commercial Product
	RAM	8 or 16 GB	8GB is for PoC Product 16GB is for Commercial Product
	Disk	128GB or 256GB SSD	For DFM Module 128GBTB (PoC), 256GB (Commercial Product)
		256 GB	For System region (OS and Rootfilesystem)
	Network Card	Above 10 Gbps	
Firmware Storage (minio)	**Server CPU Cores**	Above 1 or 2 CPU Cores	1 Cores is for PoC Product Above 2 Cores is for Commercial Product
	RAM	4 or 8 GB	4GB is for PoC Product 8GB is for Commercial Product
	Disk	1TB or 2TB SSD	For DFM Module 1TB (PoC), 2TB (Commercial Product)
		256 GB	For System region (OS and Rootfilesystem)
	Network Card	Above 10 Gbps	
DB (MySQL)	**Server CPU Cores**	Above 1 or 2 CPU Cores	1 Cores is for PoC Product Above 2 Cores is for Commercial Product

	RAM	4 or 8 GB	4GB is for PoC Product 8GB is for Commercial Product
	Disk	128GB or 256GB SSD	For DFM Module 128GB (PoC), 256GB (Commercial Product)
		256 GB	For System region (OS and Rootfilesystem)
	Network Card	Above 10 Gbps	

Table 2-1 The Hardware Recommended for the Knox E-FOTA On-Premises user work environment

The recommendations in the above table are the minimum specifications to run this On-Premises Service. User performance expectations may require additional infrastructure resources that exceed the minimum specifications.

2.2. Software Recommended

The recommended user work environment, including the network, for this On-Premises Software (S/W) requirements are as follows:

Items	Recommended Value	Description
Operating System	Ubuntu Server 18.04.3 LTS	
Docker Engine	Community Edition (Ubuntu)	
MySQL Edition	Enterprise Edition	For Commercial Product

Table 2-2 The Software Recommended for the Knox E-FOTA On-Premises user work environment

Refer to "*Appendix C*" for a summary of software recommendations.

2.2.1. Operating System

By default, the DFM Server requires Ubuntu Server 18.04.3 LTS for the OS. It should be installed on 64-bit Intel x86, ARM, or MIPS architectures in order to support Docker.

2.2.2. Docker

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package. By doing so, thanks to the container, the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.

In a way, Docker is like a virtual machine. Unlike a virtual machine, however, rather than creating a whole virtual operating system, Docker allows applications to use the same Linux kernel as the system it's running on and only requires applications be shipped with things not already running on the host computer. This provides a significant performance boost and reduces the size of the application. In this On-Premises service, **the Community version** for Ubuntu will be using Docker. This version can be downloaded from "**download.docker.com**".

2.2.3. Database (MySQL)

The MySQL database contains all service-related data, including device models, their IDs, and policy dependencies in Campaigns.

2.2.4. HTTPS

To use the https protocol between Samsung mobile devices and the DFM Modules, the customer should prepare a DNS hostname (FQDN) and public (or private) SSL certificates.

2.3. Recommendation Per each Product usage

Knox E-FOTA On-Premises has 3 types of product use case architecture recommendations, including 2 Commercial and 1 POC architecture.

2.3.1. Product – "PoC"

The **PoC** product is recommended if a customer wants to use the on-premises service to understand its functions and product configuration clearly prior to purchasing a Commercial Product, along with a small number of devices (clients, until 300 devices). The PoC product can run on lower specification hardware than the Commercial product, but the table below contains the minimum specifications to be running Knox E-FOTA On-Premises. To ensure the service runs as expected, the customer should set up the infrastructure with higher specifications than shown below.

【Minimum H/W Recommendation】

Server	Items	Recommended value	Descripstion
WEB	**Server CPU Cores**	1 CPU Cores	
	RAM	4 GB	
	Disk	128 GB SSD	For DFM Module
		256 GB	For System region (OS and Rootfilesystem)
	Network Card	Above 10 Gbps	
DFM Core/Console	**Server CPU Cores**	2 CPU Cores	
	RAM	8 GB	
	Disk	128GB SSD	For DFM Module
		256 GB	For System region (OS and Rootfilesystem)
	Network Card	Above 10 Gbps	
	One or More DFM Core/Console Servers can be configured.		
Firmware Storage (minio)	**Server CPU Cores**	1 CPU Cores	
	RAM	4 GB	
	Disk	1TB SSD	
		256 GB	For System region (OS and Rootfilesystem)
	Network Card	Above 10 Gbps	
DB (MySQL)	**Server CPU Cores**	1 CPU Cores	
	RAM	4 or 8 GB	
	Disk	128GB SSD	For DFM Module
		256 GB	For System region (OS and Rootfilesystem)
	Network Card	Above 10 Gbps	

Table 2-3 The Minimum Hardware Recommendation for PoC

【S/W Recommendation】

Items	Recommended Value	Description
Operating System	Ubuntu Server 18.04.3 LTS	
Docker Engine	Community Edition (Ubuntu)	
MySQL Edition	Community Edition	For continuous Commercial support, recommend Enterprise Edition

Table 2-4 The Software Recommendation for "PoC"

The customer can purchase Ubuntu OS based on this service package, depending on their service environment. Note that the customer must provide the service infrastructure to the Samsung representative in charge of the installation.

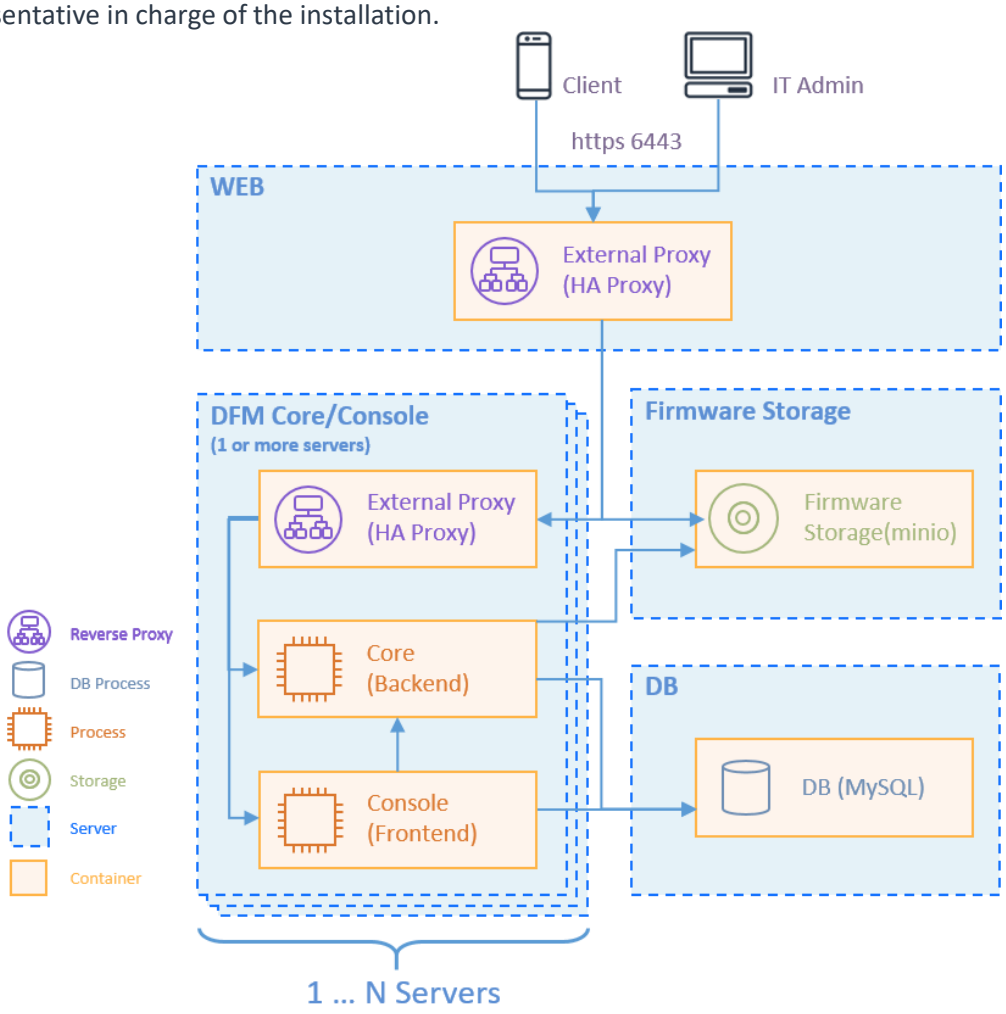

Fig 2-1 Knox E-FOTA On-Premises Product Arch for PoC

2.3.2. Product – "Commercial"

The **Commercial** product is recommended for customers who want to use this product with a maximum of 20,000 devices for device firmware updates over-the-air (FOTA), but it also supports more than 20,000 devices.

The recommended specification for the infrastructure is the minimum required to be running the service. To optimize performance expectations, the customer may need to provide infrastructure with higher specifications than the below table to the Samsung representative in charge of the installation.

【Minimum H/W Recommendation】

Server	Items	Recommended value	Descripstion
WEB	**Server CPU Cores**	2 CPU Cores	1 Cores is for PoC Product Above 2 Cores is for Commercial Product
	RAM	8 GB	
	Disk	256 GB SSD	For DFM Module
		256 GB	For System region (OS and Rootfilesystem)
	Network Card	Above 10 Gbps	
DFM Core/Console	**Server CPU Cores**	4 CPU Cores	
	RAM	16 GB	
	Disk	256GB SSD	For DFM Module
		256 GB	For System region (OS and Rootfilesystem)
	Network Card	Above 10 Gbps	
	One or More DFM Core/Console Servers can be configured.		
Firmware Storage (minio)	**Server CPU Cores**	2 CPU Cores	
	RAM	8 GB	
	Disk	2TB SSD	For DFM Module
		256 GB	For System region (OS and Rootfilesystem)
	Network Card	Above 10 Gbps	
DB (MySQL)	**Server CPU Cores**	2 CPU Cores	
	RAM	8 GB	
	Disk	256GB SSD	For DFM Module
		256 GB	For System region (OS and Rootfilesystem)
	Network Card	Above 10 Gbps	

Table 2-5 **The Minimum Hardware Recommendation for "Commercial"**

【S/W Recommendation】

The customer can purchase Ubuntu OS based this service package, depending on their service environment. Note that the customer must provide the service infrastructure to the Samsung representative in charge of the installation.

Items	Recommended Value	Description
Operating System	Ubuntu Server 18.04.3 LTS	
Docker Engine	Community Edition (Ubuntu)	
MySQL Edition	Enterprise Edition	

Table 2-6 Software Recommendation for "Commercial"

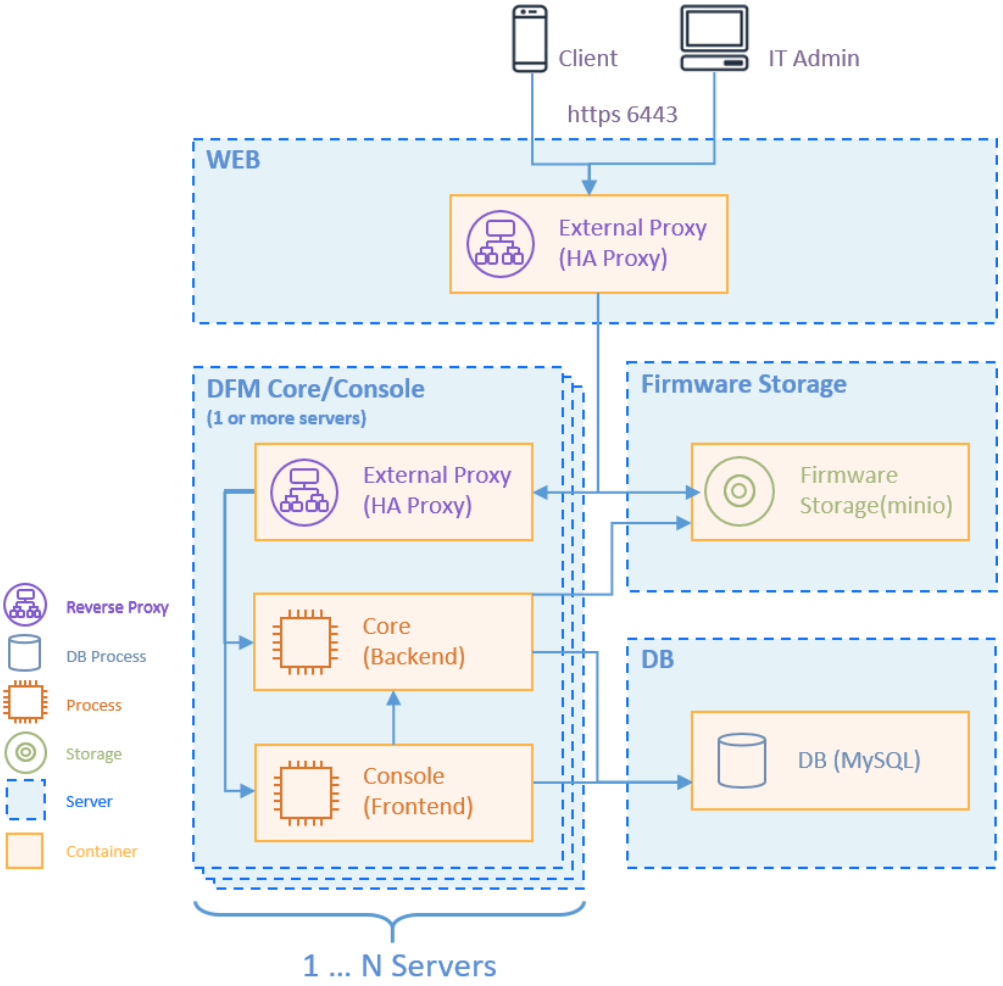

Fig 2-2 Knox E-FOTA On-Premises Product Arch for "Commercial"

3. Deliverables

This chapter describes the actions performed by Samsung to deliver the Knox E-FOTA On-Premises environment.

3.1. DFM Modules

The DFM Module consists of the following core modules:

3.1.1 **DFM Admin Console Server**: The Frontend module to provide IT admins with an accessible graphical user interface (GUI) on the Google Chrome browser.

3.1.2 **DFM Core Server**: The Backend module to manage device (client application) actions, integrated into the device using RESTful APIs from the client.

3.1.3 **DFM Database**: The MySQL-based database contains all service-related data, including device models, their IDs, and policy dependencies in Campaigns.

3.1.4 **DFM Firmware Storage Management**: The firmware files for downloaded files from the client application.

3.1.5 **Proxy:** This is used for redirection between outer and DFM modules, and for AP Gateway and TLS/SSL termination

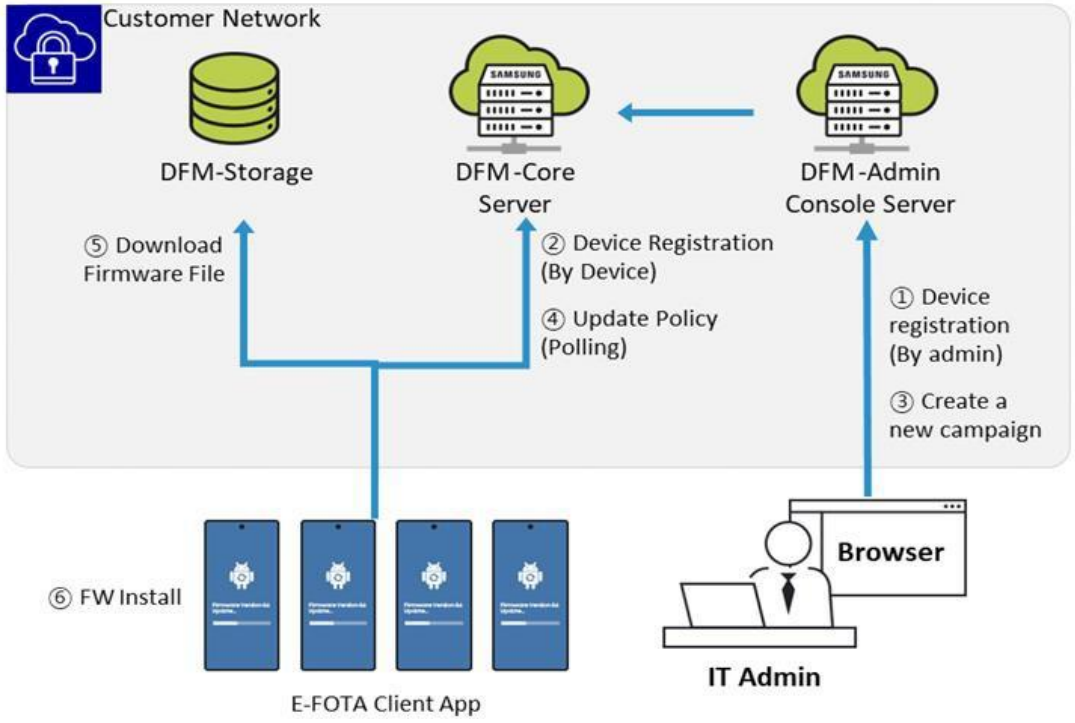

Fig 3-1 **Knox E-FOTA On-Premises Conceptual Architecture**

3.2. Security Considerations

In order to improve the default security of the Samsung deliverable, it must be implemented using the following standards.

3.2.1. HTTPS and Network encryption

The DFM Module uses HTTPS TLS-based encryption to enhance the security of transactions. The Transport Layer Security (TLS) protocol provides data encryption and verification between applications and servers in scenarios where data is being sent across an insecure network—for example, when working with the DFM Module.

HTTPS header fields are components of the header section of HTTPS request and response messages. They define the operating parameters of a HTTPS transaction. The load balancer and reverse proxy are in front of the DFM Module queries.

3.3. Supported Browser

PLEASE NOTE that this version of the DFM Console UI is designed for **Google Chrome** only.

PART II: Installation, and Validation

PART II: Installation and Validation describes how to install the Knox E-FOTA On-Premises service on the customer-provided infrastructure, and how to validate the installed service infrastructure.

4. Installation & Configuration

This chapter explains the first-time installation flow with the proper configuration conditions of the DFM Modules. Steps in this chapter run only once during initial installation.

DFM Modules must be installed on each of the 4 servers: the HA-Proxy server, DFM Core/Console server, Firmware Storage server, and MySQL server.

The **Docker Engine** must already be installed **prior to** following this installation process.

4.1. Check Pre-Config

Before installation, determine the config for each server.
The config for outside communication and the config for communication between the internal servers should be set respectively. The config for outside communication is determined through the web server config, and the config for communication between the internal servers is determined through each server config.
In this step, we will set up the initial configuration information needed for the DFM module to run as a container. The config values determined for each step are set during the installation process for each server.

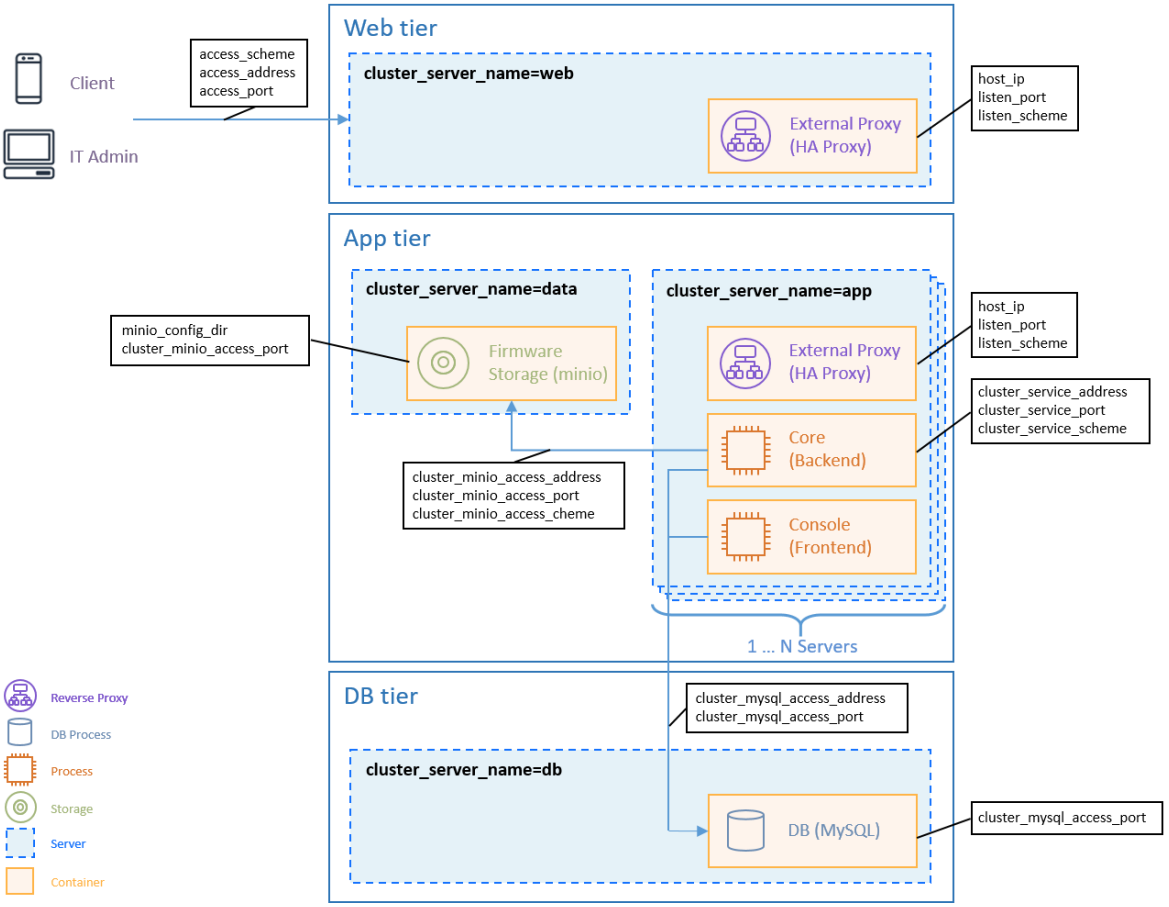

Fig 4-1 Knox E-FOTA On-Premises product arch with config

4.1.1 Check Web Server config

In this step, we will set up the initial configuration information needed for the DFM modules to perform outside communication.

【Configuration List】

- cluster_server_name: web(fixed value)
- host_ip: Static IP for DFM server.
- listen_port: External listen port at server for DFM module to be accessed.
- listen_scheme: url scheme(http or https) for DFM module to be accessed.
- access_address: domain-based or ip-based
- access_scheme: http or https
- access_port: public port
- public_endpoint: {access_scheme}://{access_address}:{access_port}

In order to properly configure this service after installation, check the customer's network environment in advance. Be sure to check and verify any port-forwarding mapping (NAT) in the customer's network.

Here are a few sample use cases:

【Use Case 1】 IP-based Access Environment

This environment reflects a real-world network environment. The host IP address is not the same, as the public IP address and the CP port number between the public network side and the customer internal network side (including DFM Modules) may be different.

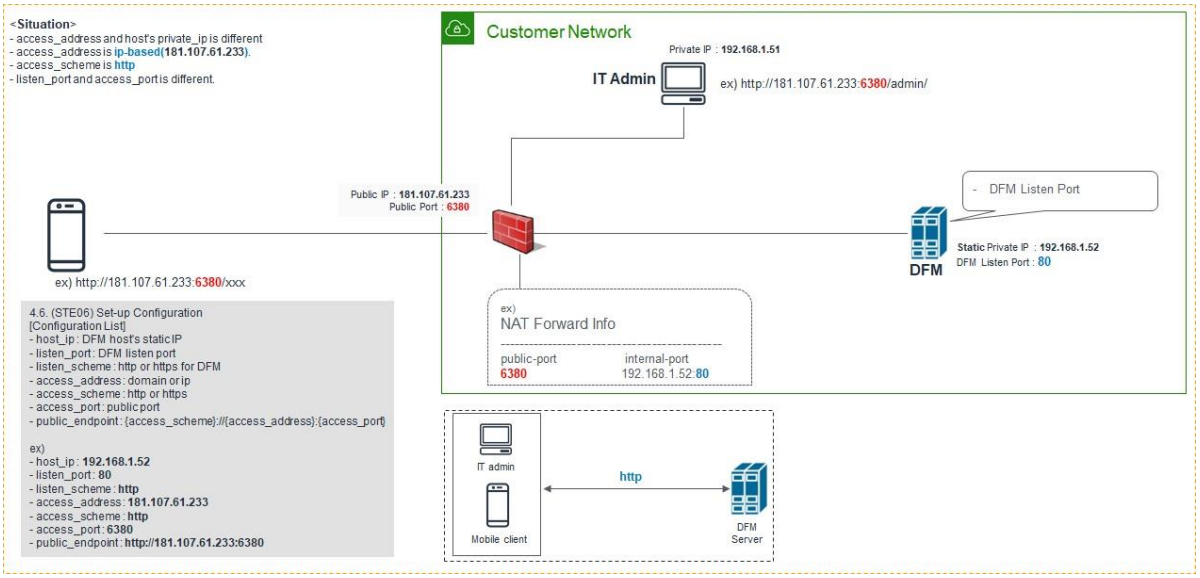

Fig 4-2 **IP-based Access Environment**

【Use Case 2】 Domain-based Access Environment

This environment represents a domain name-based network environment. You can check the network using the domain name instead of the IP address.

2-1. (Type A) Using **HTTP**

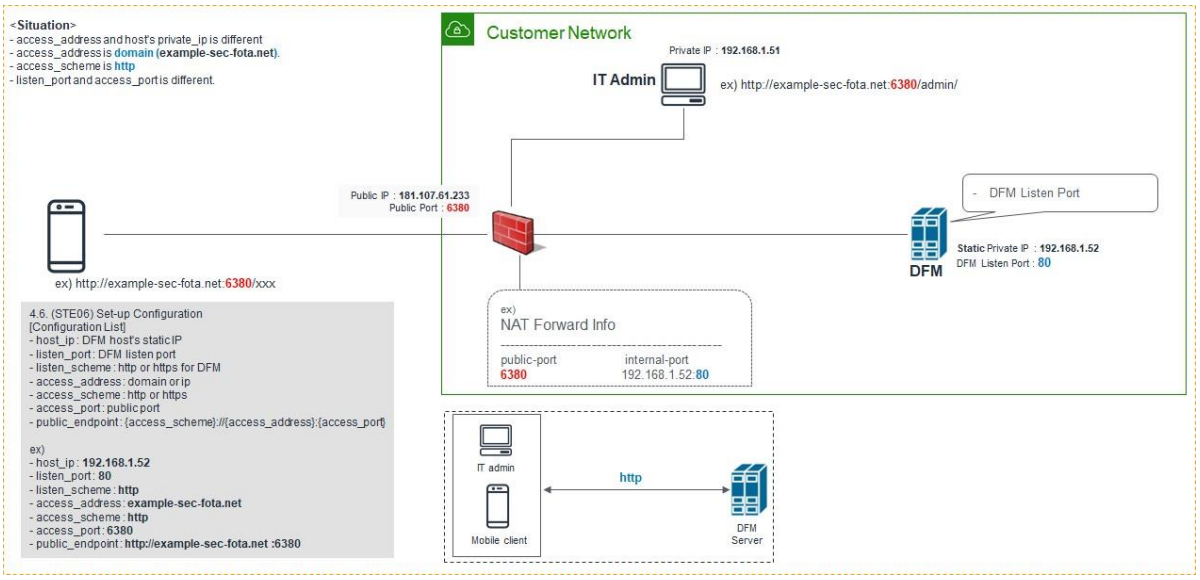

Fig 4-3 Domain-Based Access Environment (Type A)

2-2. (Type B) Using **HTTPS** - Customer's LB processes TLS/SSL Termination)

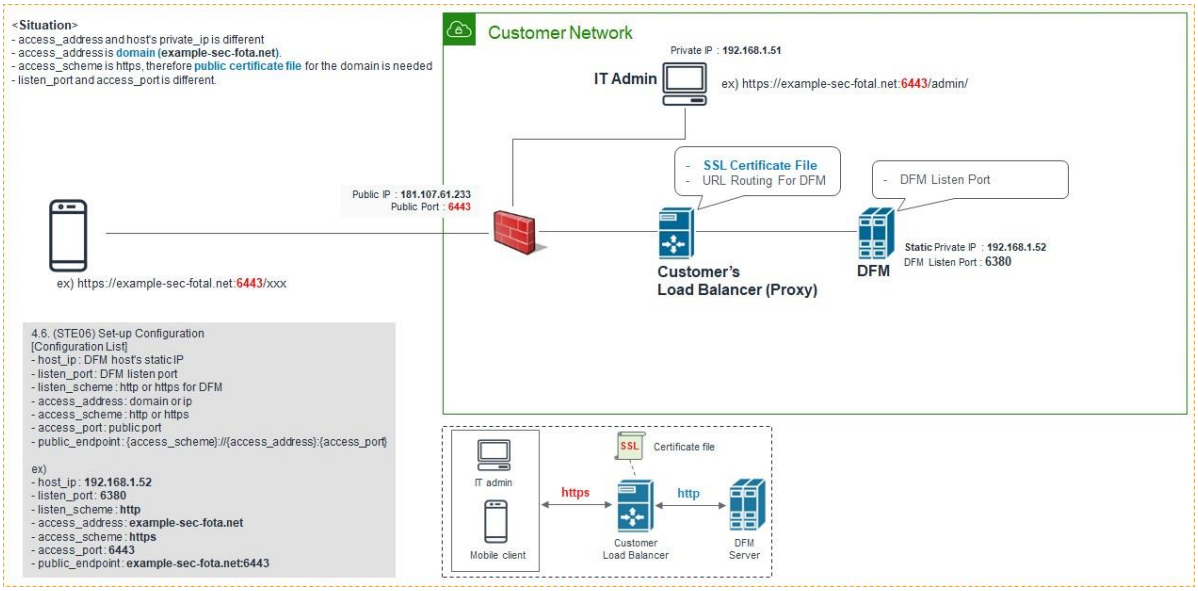

Fig 4-4 Domain-Based Access Environment (Type B)

2-3. (Type C) Using **HTTPS** - DFM processes TLS/SSL Termination

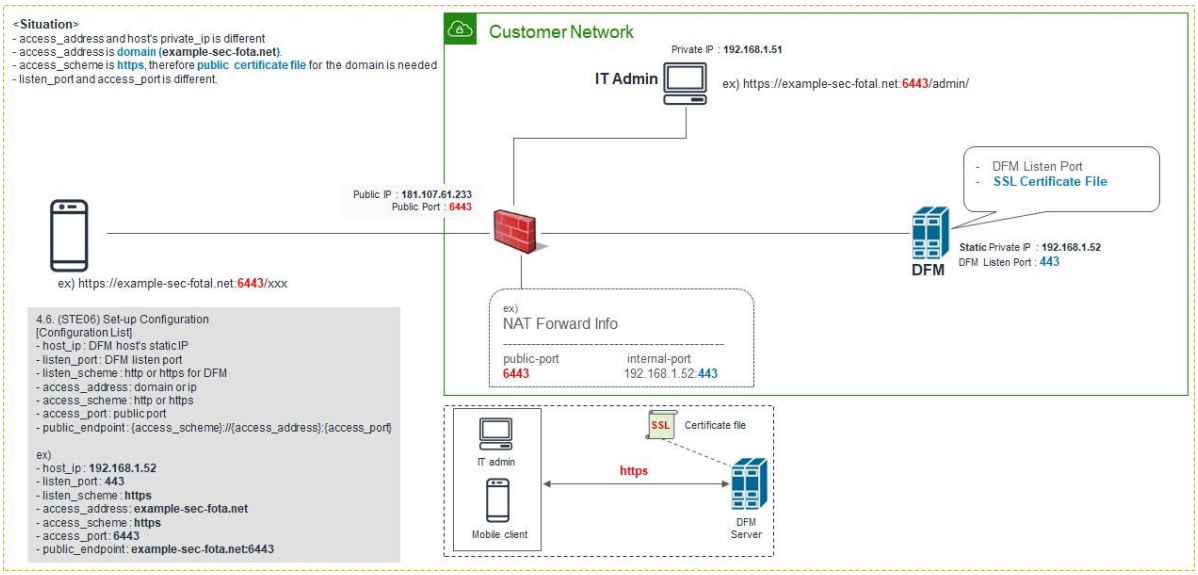

Fig 4-5 **Domain-Based Access Environment (Type C)**

The following is **an example** of how to execute the command to set the above configurations:

(CASE01) IP-Based

- host_ip: 192.168.1.52
- listen_port: 80
- listen_scheme: http
- access_address: 181.107.61.233
- access_scheme: http
- access_port: 6380

(CASE02) Domain-Based

⇦ (**Type ①**) Using **HTTP**

- host_ip: 192.168.1.52
- listen_port: 80
- listen_scheme: http
- access_address: example-sec-fota.net
- access_scheme: http
- access_port: 6380

⇦ (**Type ②**) Using **HTTPS** - Customer's LB processes TLS/SSL Termination

- host_ip: 192.168.1.52
- listen_port: 6380
- listen_scheme: http
- access_address: example-sec-fota.net
- access_scheme: https
- access_port: 6443

⇦ (**Type ③**) Using **HTTPS** - DFM processes TLS/SSL Termination

- host_ip: 192.168.1.52
- listen_port: 443
- listen_scheme: https
- access_address: example-sec-fota.net
- access_scheme: https
- access_port: 6443

4.1.2 Check DB(MySQL) Server config

The customer should decide whether to use a custom port on the DB(MySQL) server. If so, the following config needs to be set.

【Configuration List】

- host_ip: Static IP
- cluster_server_name: db (fixed value)
- cluster_mysql_access_port: port to connect to database from outside. (default : 3306)

4.1.3 Check Firmware Storage(minio) Server config

On the firmware storage (minio) server, the customer should decide:

1) Whether to use SSL

2) Whether to use a custom port

【Configuration List】

- host_ip: Static IP
- cluster_server_name: data (fixed value)
- cluster_minio_access_port: external access port (default : 9000)
- minio_config_dir: ssl file location (default : /dfm/minio/config)

4.1.4 Check DFM Core/Console Server config

On the MySQL server, the customer should decide:

1) Whether to use SSL between the Core server and Firmware storage server

2) Whether to use SSL between the Core server and HA Proxy server

3) Whether to use a custom port

【Configuration List】

- host_ip: Static IP
- listen_port: External listen port at server for DFM module to be accessed.
- listen_scheme: url scheme(http or https) for DFM module to be accessed.
- cluster_server_name: app (fixed value)
- cluster_service_address: “access_address” determined in 4.1.1
- cluster_service_port: “access_port” determined in 4.1.1
- cluster_service_scheme: “access_scheme” determined in 4.1.1
- cluster_minio_access_address: “host_ip” determined in 4.1.2
- cluster_minio_access_port: “access_port” determined in 4.1.2
- cluster_minio_access_scheme: https (if you decide to use ssl in 4.1.2)

- cluster_mysql_access_address: “host_ip” determined in 4.1.3
- cluster_mysql_access_port: “access_port” determined in 4.1.3

4.2. DB(MySQL) Server

4.2.1 (STEP01) Create Service Account and Login

The DFM Module is logged in with **a dedicated service account** and operates with the privileges of the account. Therefore, the dedicated service account has to be created in the server. The service account also needs the “**sudo**” privilege as a Docker requirement for command permissions. Ensure you add your service account into the Docker group.
We recommend that you create a service account before you start the installation.

The below shows you how to add your service account into the Docker group:

We assume that you are using the “**nightwatch**” account.

```
$ sudo usermod -aG docker {your-user}

Example)
 sudo usermod -aG docker nightwatch
```

4.2.2 (STEP02) Prepare “Disk partition & mount” for DFM modules

DFM module is installed in and operates in the below directory on the **dedicated disk**.

Therefore, we should check if the dedicated disk exists and the “partition & mount” is ready, in case the customer has not worked with the disk partition for the DFM module before.

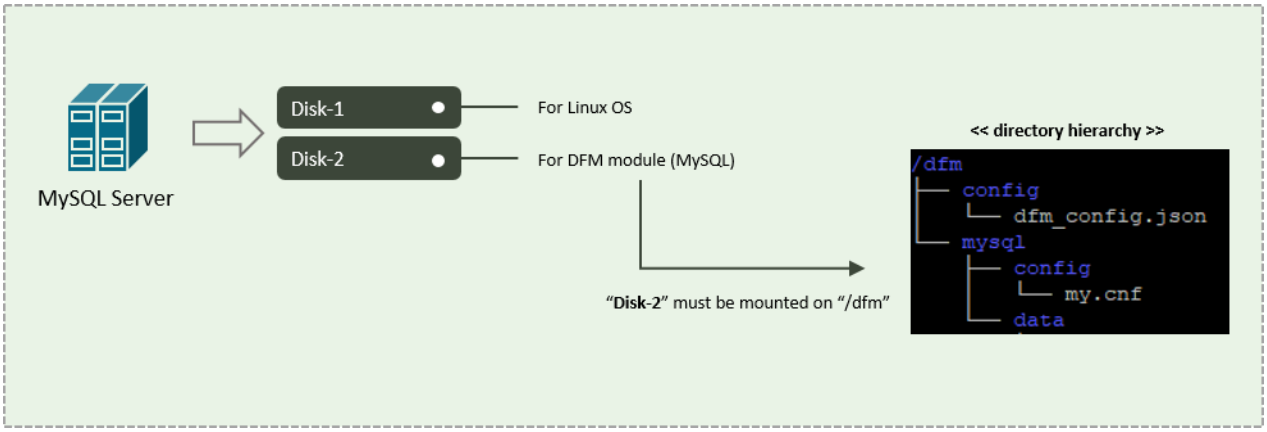

Fig 4-6 **An Disk Partitions for DMF Module on DB(MySQL) server**

For example, we assume that two disks (“sda” and “sdb”) exist.

【CASE01】 Disk is Ready

If the disks exist, we don’t need to format and mount them.

Now, let’s check the disk information:

```
sudo lsblk -p
```

```
/dev/sda     202:0    0    1T  0 disk
└─/dev/sda1  202:1    0    1T  0 part /
/dev/sdb     202:80   0    1T  0 disk
```

sudo lsblk -f

```
NAME    FSTYPE   LABEL          UUID                                   MOUNTPOINT
sda
└─sda1  ext4     xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb     ext4                    d3269ceb-4418-45d0-ba68-d6b906e0595d /dfm
```

⇨ "sdb" is already formatted and mounted on **/dfm**

sudo file -s /dev/sdb

```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

【CASE02】 Disk is NOT Ready: it is not formatted

If the disk is not ready, it needs to be formatted and mounted on **/dfm**.

Now, let's check the disk information:

sudo lsblk -p

```
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
/dev/sda     202:0    0    1T  0 disk
└─/dev/sda1 202:1    0    1T  0 part /
/dev/sdb     202:80   0    1T  0 disk
```

sudo lsblk -f

```
NAME    FSTYPE   LABEL          UUID                                   MOUNTPOINT
sda
└─sda1 ext4      xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb
```

⇨ "sdb" is NOT formatted

sudo file -s /dev/sdb

```
/dev/sdb: data
```

⇨ This means that the disk needs to be formatted

1) Format with ext4 file-system

sudo file -s /dev/sdb

```
sudo mkfs -t ext4 /dev/sdb
mke2fs 1.44.1 (24-Mar-2018)
Creating filesystem with 2621440 4k blocks and 655360 inodes
Filesystem UUID: d3269ceb-4418-45d0-ba68-d6b906e0595d
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

2) Check if the disk is formatted

```
sudo mkfs -t ext4 /dev/sdb
```

```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

3) Mount "/dev/sdb" on /dfm

```
// create directory to mount
sudo mkdir /dfm

// mount
sudo mount /dev/sdb /dfm
```

4) Verify

```
df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb        9.8G   37M  9.3G   1% /dfm
```

【CASE03】 Disk is NOT Ready: it is already formatted but not yet mounted on /dfm

If the disk is formatted but not yet mounted, it needs to be mounted on **/dfm**.

Now, let's check the disk information:

```
sudo lsblk -p
```

```
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
/dev/sda     202:0    0    1T  0 disk
└─/dev/sda1 202:1    0    1T  0 part /
/dev/sdb     202:80   0    1T  0 disk
```

```
sudo lsblk -f
```

```
NAME    FSTYPE   LABEL          UUID                                   MOUNTPOINT
sda
└─sda1  ext4     xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb     ext4                     d3269ceb-4418-45d0-ba68-d6b906e0595d
```

⇨ "sdb" " is formatted but Not yet mounted

1) Mount /dev/sdb on /dfm

```
// create directory to mount
sudo mkdir /dfm

// mount
sudo mount /dev/sdb /dfm
```

2) Verify

```
df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb        9.8G   37M  9.3G   1% /dfm
```

4.2.3 Permanently mount the disk

We recommend that the **customer's IT manager** sets the boot script so that **the dedicated disk** is auto-mounted when the server is booted.

If the **customer's IT manager** has not set the boot script for disk auto-mounting, you should proceed according to the command below.

*) If the settings are incorrect, booting may not be possible. The command below is for general situations, and options may differ depending on the customer's system and situation. Please refer to the "fstab" manual for details.

1) Check mount /dev/sdb on /dfm

```
sudo lsblk -f
```

```
NAME    FSTYPE   LABEL          UUID                                   MOUNTPOINT
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
sda
└─sda1  ext4     xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb     ext4                     d3269ceb-4418-45d0-ba68-d6b906e0595d /dfm
```

2) Edit /etc/fstab file

Add the content next to "sdb" to the new line.

```
vi /etc/fstab

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~
UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d    /dfm   ext4   defaults      0 0
```

4.2.4 (STEP03) Create Service Directories

A separated service directory configuration is required to install and operate the Samsung DFM Module. The service account must have "**read / write / execute**" permissions to the service directory. The service directory should be mounted in a different device location from the OS installation area.

【Service Directory List】

/ dfm/mysql/config

⇒ that is where the config file is referenced when the mysql server starts.

/dfm/mysql/data

⇒ that is where databases are created when mysql server runs.

/dfm/mysql/backups

⇒ that is where databases are backed-up when mysql server runs.

/dfm/config

⇒ that is where the config file contains the information needed to run the DFM module.

Now, let's create each service directory.

```
sudo mkdir -p /dfm/mysql/config
sudo mkdir -p /dfm/mysql/data
sudo mkdir -p /dfm/mysql/backups
sudo mkdir -p /dfm/config
```

Set the service account's permission for the created service directory.

We assume that you are using the "**nightwatch**" account.

```
sudo chown -R nightwatch:nightwatch /dfm
sudo chown -R nightwatch:nightwatch /dfm/mysql
sudo chown -R nightwatch:nightwatch /dfm/mysql/config
sudo chown -R nightwatch:nightwatch /dfm/mysql/data
sudo chown -R nightwatch:nightwatch /dfm/mysql/backups
sudo chown -R nightwatch:nightwatch /dfm/config
```

4.2.5 (STEP04) Install DFM Module Package

The DFM Module will either be delivered as a debian package or an rpm package tool. This package contains the following resources:

- executable binary (dfm): managed command to run DFM module
- docker images: docker image about DFM module
- sql query file: DFM module's DB data to initialize mysql
- mysql config file (my.cnf): config file for mysql
- dfm config file (dfm_config.json): config file for DFM module

To install these resources, the files have to be unpacked within the following locations by the host. The files will be used during: **1) Docker Image load**, **2) initializing MySQL DB**, and **3) Copying the config file** to the service directory.

- executable binary:
  ⇒ /usr/bin/dfm

- docker images:
  ⇒ /tmp/dfm/docker-images/haproxy-debian-2.1.4.tar
  ⇒ /tmp/dfm/docker-images/minio-RELEASE. 2020-06-01T17-28-03Z.tar
  ⇒ /tmp/dfm/docker-images/mysql-enterprise-server-8.0.20.tar
  ⇒ /tmp/dfm/docker-images/dfm-console-xxx.tar
  ⇒ /tmp/dfm/docker-images/dfm-core-xxx.tar

- sql query file:
  ⇒ /tmp/dfm/mysql-query/init_db.sql
  ⇒ /tmp/dfm/mysql-query/init_dfm_core.sql
  ⇒ /tmp/dfm/mysql-query/init_dfm_console.sql

- mysql config file:
  ⇒ /tmp/dfm/mysql-config/my.cnf

- dfm config file:
  ⇒ /tmp/dfm/ha/db-server/dfm_config.json

The following is a command showing how to install the debian package:

```
  sudo dpkg -i xxx.deb

example)
  sudo dpkg -i sec-dfm_1.0.1.5.deb
Selecting previously unselected package dfm.
(Reading database ... 973294 files and directories currently
installed.)
Preparing to unpack sec-dfm_1.0.1.5.deb ...
Unpacking dfm (1.0.1.5) ...
Setting up dfm (1.0.1.5) ...
```

Next, check if the necessary files exist:

```
1) check dfm file
ls /usr/bin/dfm
/usr/bin/dfm

2) check docker images
ls /tmp/dfm/docker-images/ -l
total 971552
-rw-rw-r-- 1 dfm-console-1.0.1.4.tar
-rw-rw-r-- 1 dfm-core-1.0.1.4.tar
-rw-rw-r-- 1 haproxy-debian-2.1.4.tar
-rw-rw-r-- 1 minio-RELEASE.2020-06-01T17-28-03Z.tar
-rw-rw-r-- 1 mysql-enterprise-server-8.0.20.tar

3) check sql query file
ls /tmp/dfm/mysql-query/ -l
total 2076
-rw-r--r-- 1 init_db.sql
-rw-r--r-- 1 init_dfm_console.sql
-rw-r--r-- 1 init_dfm_core.sql

4) check mysql config file : my.cnf
ls /tmp/dfm/mysql-config/ -l
total 4
-rw-r--r-- my.cnf

5) dfm config file : dfm_config.json
ls /tmp/dfm/ha/db-server/dfm_config.json
/tmp/dfm/ha/db-server/dfm_config.json
```

4.2.6 (STEP05) Load Docker Image

Next, register the Docker Images that were unpacked at "**/tmp/dfm/docker-images**". The loaded Docker Images are used when the container is driven. The following shows how to load Docker image required for DB(MySQL) server using Docker commands:

```
docker load < /tmp/dfm/docker-images/mysql-enterprise-server-8.0.20.tar
```

Next, check if the MySQL Docker image was loaded. Use the "Docker Images" command:

Example)

docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mysql/enterprise-server	8.0	f350b0949588	8 days ago	462MB

4.2.7 (STEP06) Copy Configuration files

After loading the Docker images, copy the following configuration files into the service directory from the unpacked resources directory.
We assume that you are using the "**nightwatch**" account.

- **copy mysql config file**:

```
// copy configuration file
cp /tmp/dfm/mysql-config/my.cnf /dfm/mysql/config

// Set the service account's permission to the configuration file.
sudo chown -R nightwatch:nightwatch /dfm/mysql/config
```

- **copy dfm config file**:

```
// copy configuration file
cp /tmp/dfm/ha/db-server/dfm_config.json /dfm/config

//Set the service account's permission to the configuration file.
sudo chown -R nightwatch:nightwatch /dfm/config
```

4.2.8 (STEP07) Set-up Configuration

In this step, we will set up the initial configuration needed to connect from another intenal server to DB(MySQL) server.
DB(MySQL) server requires "cluster_server_name" and "cluster_mysql_access_port" settings.

【Configuration List】

- cluster_server_name: db (fixed value)
- cluster_mysql_access_port: port to connect to database from outside. (default : 3306)
- host_ip: static ip

The following shows the commands:

(CASE01) Use default port

```
dfm cluster config set cluster_server_name=db
```

(CASE02) Use a port other than the default port (ex: Change port to 33066 from default port)

```
dfm cluster config set cluster_server_name=db
dfm cluster config set cluster_mysql_access_port =33066
```

Next, check if the configured value is correct. Use the "**dfm cluster config get {*key*}**" command:

```
Example)
dfm cluster config get cluster_server_name
  db
dfm cluster config get cluster_mysql_access_port
  3306
```

4.2.9 (STEP08) Create Container Network

The DFM Module is a process executed on a container basis, creating the Docker network required for communications among containers.

To create a network, use the following command:

```
dfm network create
```

【Validation】

Run the following command to see if "dfm-network" is visible.

```
dfm network ls
NETWORK ID          NAME                DRIVER              SCOPE
~~~~~~~~~~
e2697cd6621a        dfm-network         bridge              local
~~~~~~~~~~
```

4.2.10 (STEP9) Start-up and Initialize DB(MySQL) Server

In this stage, you will perform the following two steps:

1) Set DB root password
2) Initialize the SQL query file copied in "4.3 Installing the DFM Module Package" above, on the mysql server

To do this, you must first start the mysql server container.

The command to run the mysql server container is as follows:

```
dfm cluster start dfm-mysql
```

【Validation】

Make sure the MySQL container is in a healthy state. It may take some time until its state is healthy.

```
docker ps -a
CONTAINER ID   IMAGE                          STATUS                            PORTS                 NAMES
efded2363698   mysql/enterprise-server:8.0    Up 3 seconds (health: starting)   3306/tcp, 33060/tcp   dfm-mysql
~~
$ docker ps -a
CONTAINER ID   IMAGE                          STATUS                            PORTS                 NAMES
efded2363698   mysql/enterprise-server:8.0    Up 46 seconds (health: healthy)   3306/tcp, 33060/tcp   dfm-mysql
```

If the status is healthy, run each of the following commands.

1) Set DB root password : we assume that “pass-word” is “**1q2w3e4r**”
We use this command: ALTER USER 'root'@'localhost' IDENTIFIED BY '{password}'

```
docker exec -it dfm-mysql mysql -uroot
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 5.7.25-log MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY '1q2w3e4r';
Query OK, 0 rows affected (0.00 sec)

mysql> exit
```

2) Run sql query file : we assume that pass-word is “1q2w3e4r”

```
docker exec -i dfm-mysql mysql -uroot -p1q2w3e4r < /tmp/dfm/mysql-query/init_db.sql
mysql: [Warning] Using a password on the command line interface can be insecure.

docker exec -i dfm-mysql mysql -uroot -p1q2w3e4r < /tmp/dfm/mysql-query/init_dfm_core.sql
mysql: [Warning] Using a password on the command line interface can be insecure.

docker exec -i dfm-mysql mysql -uroot -p1q2w3e4r < /tmp/dfm/mysql-query/init_dfm_console.sql
mysql: [Warning] Using a password on the command line interface can be insecure.
$
```

4.3. Firmware Storage(minio) Server

4.3.1 (STEP01) Create Service Account and Login

The DFM Module is logged in with **a dedicated service account** and operates with the privileges of the account. Therefore, the dedicated service account has to be created in the server. The service account also needs the "**sudo**" privilege as a Docker requirement for command permissions. Ensure you add your service account into the Docker group.
We recommend that you create a service account before you start the installation.

The below shows you how to add your service account into the Docker group:

We assume that you are using the "**nightwatch**" account.

```
$ sudo usermod -aG docker {your-user}

Example)
  sudo usermod -aG docker nightwatch
```

4.3.2 (STEP02) Prepare "Disk partition & mount" for DFM modules

DFM module is installed in and operates in the below directory on the **dedicated disk**.

Therefore, we should check if the dedicated disk exists and the "partition & mount" is ready, in case the customer has not worked with the disk partition for the DFM module before.

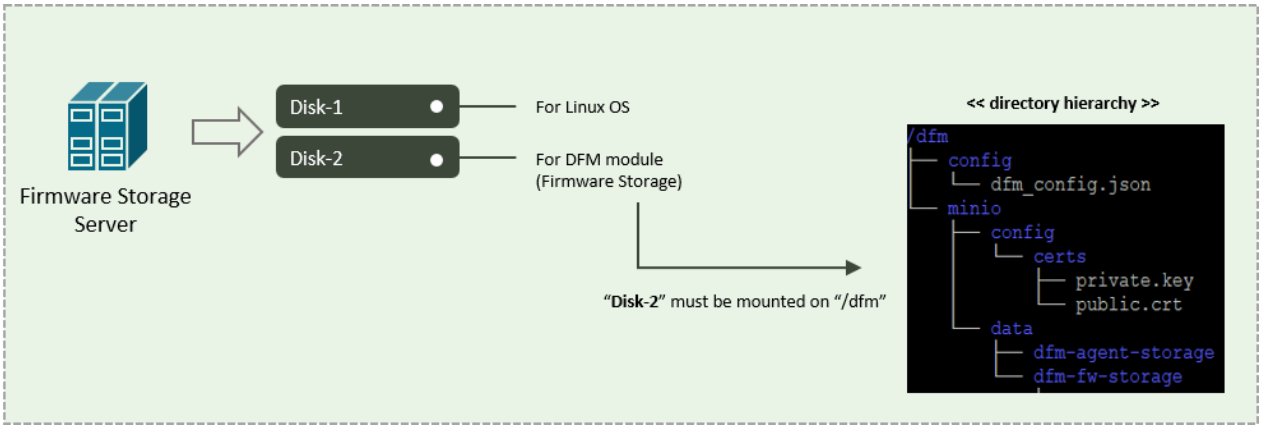

Fig 4-7 **An Disk Partitions for DMF Module on Firmware Storage(minio) server**

For example, we assume that two disks ("sda" and "sdb") exist.

【CASE01】 Disk is Ready

If the disks exist, we don't need to format and mount them.

Now, let's check the disk information:

```
sudo lsblk -p
```

```
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
/dev/sda     202:0    0    1T  0 disk
└─/dev/sda1 202:1    0    1T  0 part /
/dev/sdb     202:80   0    1T  0 disk
```

sudo lsblk -f

```
NAME    FSTYPE   LABEL          UUID                                   MOUNTPOINT
sda
└─sda1  ext4     xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb     ext4                    d3269ceb-4418-45d0-ba68-d6b906e0595d /dfm
```

⇨ “sdb” is already formatted and mounted on **/dfm**

sudo file -s /dev/sdb

```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

【CASE02】 Disk is NOT Ready: it is not formatted

If the disk is not ready, it needs to be formatted and mounted on **/dfm**.

Now, let’s check the disk information:

sudo lsblk -p

```
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
/dev/sda     202:0    0    1T  0 disk
└─/dev/sda1 202:1    0    1T  0 part /
/dev/sdb     202:80   0    1T  0 disk
```

sudo lsblk -f

```
NAME    FSTYPE   LABEL          UUID                                   MOUNTPOINT
sda
└─sda1 ext4      xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb
```

⇨ “sdb” is NOT formatted

sudo file -s /dev/sdb

```
/dev/sdb: data
```

⇨ This means that the disk needs to be formatted

5) Format with ext4 file-system

sudo file -s /dev/sdb

```
sudo mkfs -t ext4 /dev/sdb
mke2fs 1.44.1 (24-Mar-2018)
Creating filesystem with 2621440 4k blocks and 655360 inodes
Filesystem UUID: d3269ceb-4418-45d0-ba68-d6b906e0595d
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

6) Check if the disk is formatted

```
sudo mkfs -t ext4 /dev/sdb
```

```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

7) Mount "/dev/sdb" on /dfm

```
// create directory to mount
sudo mkdir /dfm

// mount
sudo mount /dev/sdb /dfm
```

8) Verify

```
df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb        9.8G   37M  9.3G   1% /dfm
```

【CASE03】 Disk is NOT Ready: it is already formatted but not yet mounted on /dfm

If the disk is formatted but not yet mounted, it needs to be mounted on **/dfm**.

Now, let's check the disk information:

```
sudo lsblk -p
```

```
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
/dev/sda     202:0    0    1T  0 disk
└─/dev/sda1 202:1    0    1T  0 part /
/dev/sdb     202:80   0    1T  0 disk
```

```
sudo lsblk -f
```

```
NAME    FSTYPE   LABEL          UUID                                   MOUNTPOINT
sda
└─sda1  ext4     xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb     ext4                     d3269ceb-4418-45d0-ba68-d6b906e0595d
```

⇨ "sdb" " is formatted but Not yet mounted

3) Mount /dev/sdb on /dfm

```
// create directory to mount
sudo mkdir /dfm

// mount
sudo mount /dev/sdb /dfm
```

4) Verify

```
df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb        9.8G   37M  9.3G   1% /dfm
```

4.3.3 Permanently mount the disk

We recommend that the **customer's IT manager** sets the boot script so that **the dedicated disk** is auto-mounted when the server is booted.

If the **customer's IT manager** has not set the boot script for disk auto-mounting, you should proceed according to the command below.

*) If the settings are incorrect, booting may not be possible. The command below is for general situations, and options may differ depending on the customer's system and situation. Please refer to the "fstab" manual for details.

3) Check mount /dev/sdb on /dfm

```
sudo lsblk -f
```

```
NAME    FSTYPE   LABEL          UUID                                   MOUNTPOINT
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
sda
└─sda1  ext4     xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb     ext4                     d3269ceb-4418-45d0-ba68-d6b906e0595d /dfm
```

4) Edit /etc/fstab file

Add the content next to "sdb" to the new line.

```
vi /etc/fstab

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~
UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d    /dfm    ext4   defaults       0 0
```

4.3.4 (STEP03) Create Service Directories

A separated service directory configuration is required to install and operate the Samsung DFM Module. The service account must have "**read / write / execute**" permissions to the service directory. The service directory should be mounted in a different device location from the OS installation area.

【Service Directory List】

/dfm/minio/data/dfm-agent-storage

⇒ that is where efota client APK files are uploaded when minio server runs.

/dfm/minio/data/dfm-fw-storage

⇒ that is where firmware binary files are uploaded when minio server runs.

/dfm/minio/config/certs

⇒ that is where private key and public key are located by default when minio server communicate by ssl.

/dfm/config

⇒ that is where the config file contains the information needed to run the DFM module.

Now, let's create each service directory.

```
sudo mkdir -p /dfm/minio/data/dfm-agent-storage
sudo mkdir -p /dfm/minio/data/dfm-fw-storage
sudo mkdir -p /dfm/minio/config/certs
sudo mkdir -p /dfm/config
```

Set the service account's permission for the created service directory.

We assume that you are using the "**nightwatch**" account.

```
sudo chown -R nightwatch:nightwatch /dfm
sudo chown -R nightwatch:nightwatch /dfm/minio
sudo chown -R nightwatch:nightwatch /dfm/minio/config/certs
sudo chown -R nightwatch:nightwatch /dfm/minio/data/dfm-agent-storage
sudo chown -R nightwatch:nightwatch /dfm/minio/data/dfm-fw-storage
sudo chown -R nightwatch:nightwatch /dfm/config
```

4.3.5 (STEP04) Install DFM Module Package

The DFM Module will either be delivered as a debian package or an rpm package tool. This package contains the following resources:

- executable binary (dfm): managed command to run DFM module
- docker images: docker image about DFM module
- dfm config file (dfm_config.json): config file for DFM module

To install these resources, the files have to be unpacked within the following locations by the host. The files will be used during: **1) Docker Image load**, **2) initializing MySQL DB**, and **3) Copying the config file** to the service directory.

- executable binary:
  ⇒ /usr/bin/dfm

- docker images:
  ⇒ /tmp/dfm/docker-images/haproxy-debian-2.1.4.tar
  ⇒ /tmp/dfm/docker-images/minio-RELEASE.2020-06-01T17-28-03Z.tar
  ⇒ /tmp/dfm/docker-images/mysql-enterprise-server-8.0.20.tar
  ⇒ /tmp/dfm/docker-images/dfm-console-xxx.tar
  ⇒ /tmp/dfm/docker-images/dfm-core-xxx.tar

- dfm config file:
  ⇒ /tmp/dfm/ha/data-server/dfm_config.json

The following is a command showing how to install the debian package:

```
sudo dpkg -i xxx.deb

example)
sudo dpkg -i sec-dfm_1.0.1.5.deb
Selecting previously unselected package dfm.
(Reading database ... 973294 files and directories currently
installed.)
Preparing to unpack sec-dfm_1.0.1.5.deb ...
Unpacking dfm (1.0.1.5) ...
Setting up dfm (1.0.1.5) ...
```

Next, check if the necessary files exist:

```
1) check dfm file
ls /usr/bin/dfm
/usr/bin/dfm

2) check docker images
ls /tmp/dfm/docker-images/ -l
total 971552
-rw-rw-r-- 1 dfm-console-1.0.1.4.tar
-rw-rw-r-- 1 dfm-core-1.0.1.4.tar
-rw-rw-r-- 1 haproxy-debian-2.1.4.tar
-rw-rw-r-- 1 minio-RELEASE.2020-06-01T17-28-03Z.tar
-rw-rw-r-- 1 mysql-enterprise-server-8.0.20.tar

3) dfm config file : dfm_config.json
ls /tmp/dfm/ha/data-server/dfm_config.json
/tmp/dfm/ha/data-server/dfm_config.json
```

4.3.6 (STEP05) Load Docker Image

Next, register the Docker Images that were unpacked at "**/tmp/dfm/docker-images**". The loaded Docker Images are used when the container is driven. The following shows how to load Docker Image required for Firmware storage(minio) server using Docker commands:

```
docker load < /tmp/dfm/docker-images/minio-RELEASE.2020-06-01T17-28-03Z.tar
```

Next, check if the minio Docker image was loaded. Use the "Docker Images" command:

Example)

```
docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
minio/minio	RELEASE.2020-06-01T17-28-03Z	2f89782ec9dc	8 days ago	56.7MB

4.3.7 (STEP06) Copy Configuration files

After loading the Docker images, copy the following configuration files into the service directory from the unpacked resources directory.
We assume that you are using the "**nightwatch**" account.
- **copy dfm config file**:

```
// copy configuration file
cp /tmp/dfm/ha/data-server/dfm_config.json /dfm/config

//Set the service account's permission to the configuration file.
sudo chown -R nightwatch:nightwatch /dfm/config
```

4.3.8 (STEP07) Set-up Configuration

In this step, we will set up the initial configuration information needed for the DFM module to run as a Container.

【Configuration List】

- cluster_server_name: data (fixed value)
- cluster_minio_access_port: external access port (default : 9000)
- minio_config_dir: ssl file location (default : /dfm/minio/config)

The following shows the commands:

(CASE01) Use default port,

```
dfm cluster config set cluster_server_name=data
```

(CASE02) Use a port other than the default port (ex: Change port to 9001 from default port)

```
dfm cluster config set cluster_server_name=data
dfm cluster config set cluster_minio_access_port=9001
```

Next, check if the configured value is correct. Use the "**dfm cluster config get {*key*}**" command:

```
dfm cluster config get cluster_server_name
```

```
 data

dfm cluster config get cluster_minio_access_port
 9000

dfm cluster config get minio_config_dir
 /dfm/minio/config
```

4.3.9 (STEP08) Create Container Network

The DFM Module is a process executed on a container basis, creating the Docker network required for communications among containers.

To create a network, use the following command:

```
dfm network create
```

【Validation】

Run the following command to see if "dfm-network" is visible.

```
 dfm network ls
NETWORK ID          NAME                DRIVER              SCOPE
~~~~~~~~~~
e2697cd6621a        dfm-network         bridge              local
~~~~~~~~~~
```

4.3.10 (STEP09) Start-up Firmware Storage(minio) Server

In this stage, the installer starts the storage server that manages the firmware binary.

The command to run the Firmware Storage Server container is as follows:

```
dfm cluster start dfm-minio
```

【Validation】

Make sure the Minio container is in a healthy state. It may take some time until its state is healthy.

```
docker ps -a

Example)
$ docker ps -a
CONTAINER ID    ~    STATUS                          ~    NAMES
~~
af3949b8db98    Up 4 seconds (health: starting)          dfm-minio
~~
$

$ docker ps -a
CONTAINER ID    ~    STATUS                          ~    NAMES
```

```
~~
af3949b8db9       Up 2 minutes (healthy)                dfm-minio
~~
```

4.4. DFM Core/Console Server

4.4.1 (STEP01) Create Service Account and Login

The DFM Module is logged in with **a dedicated service account** and operates with the privileges of the account. Therefore, the dedicated service account has to be created in the server. The service account also needs the "**sudo**" privilege as a Docker requirement for command permissions. Ensure you add your service account into the Docker group.
We recommend that you create a service account before you start the installation.

The below shows you how to add your service account into the Docker group:

We assume that you are using the "**nightwatch**" account.

```
$ sudo usermod -aG docker {your-user}

Example)
 sudo usermod -aG docker nightwatch
```

4.4.2 (STEP02) Prepare "Disk partition & mount" for DFM modules

DFM module is installed in and operates in the below directory on the **dedicated disk**.

Therefore, we should check if the dedicated disk exists and the "partition & mount" is ready, in case the customer has not worked with the disk partition for the DFM module before.

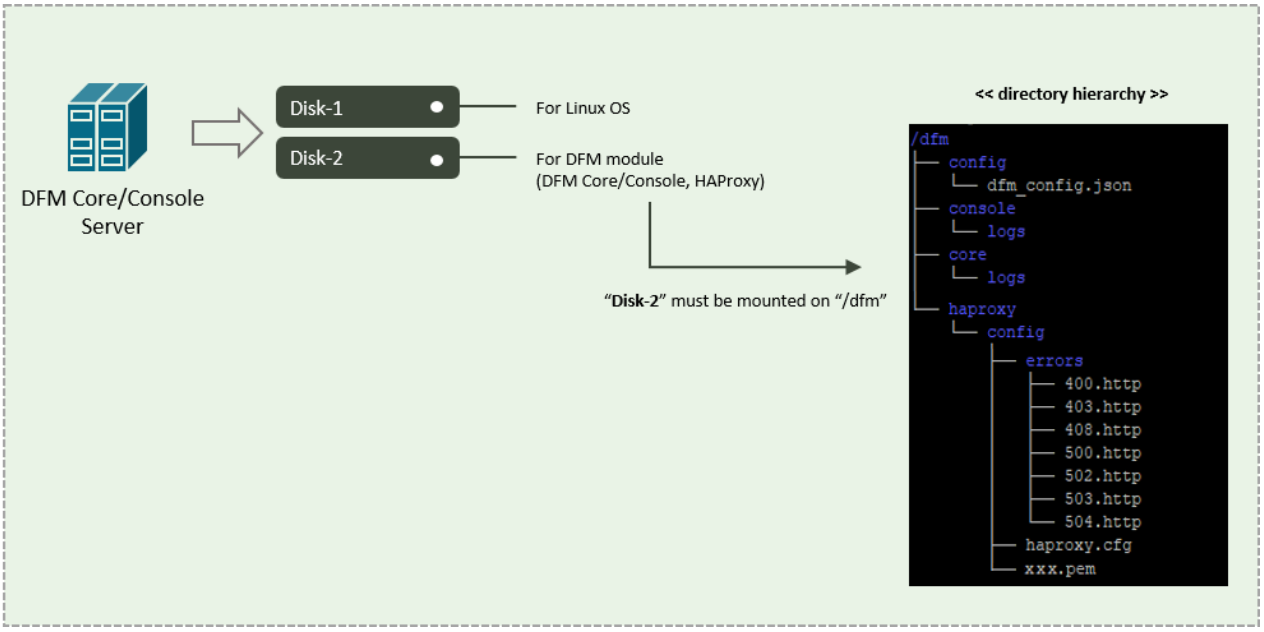

Fig 4-8 An Disk Partitions for DMF Module on DFM core/console server

For example, we assume that two disks ("sda" and "sdb") exist.

【CASE01】 Disk is Ready

If the disks exist, we don't need to format and mount them.
Now, let's check the disk information:

```
sudo lsblk -p
```

```
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
/dev/sda     202:0    0    1T  0 disk
└─/dev/sda1  202:1    0    1T  0 part /
/dev/sdb     202:80   0    1T  0 disk
```

```
sudo lsblk -f
```

```
NAME    FSTYPE   LABEL          UUID                                 MOUNTPOINT
sda
└─sda1  ext4     xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb     ext4                     d3269ceb-4418-45d0-ba68-d6b906e0595d /dfm
```

⇨ "sdb" is already formatted and mounted on **/dfm**

```
sudo file -s /dev/sdb
```

```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

【CASE02】 Disk is NOT Ready: it is not formatted

If the disk is not ready, it needs to be formatted and mounted on **/dfm**.
Now, let's check the disk information:

```
sudo lsblk -p
```

```
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
/dev/sda     202:0    0    1T  0 disk
└─/dev/sda1  202:1    0    1T  0 part /
/dev/sdb     202:80   0    1T  0 disk
```

```
sudo lsblk -f
```

```
NAME    FSTYPE   LABEL          UUID                                 MOUNTPOINT
sda
└─sda1  ext4     xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb
```

⇨ "sdb" is NOT formatted

```
sudo file -s /dev/sdb
```

```
/dev/sdb: data
```

⇨ This means that the disk needs to be formatted

9) Format with ext4 file-system

```
sudo file -s /dev/sdb
```

```
sudo mkfs -t ext4 /dev/sdb
mke2fs 1.44.1 (24-Mar-2018)
Creating filesystem with 2621440 4k blocks and 655360 inodes
Filesystem UUID: d3269ceb-4418-45d0-ba68-d6b906e0595d
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

10) Check if the disk is formatted

```
sudo mkfs -t ext4 /dev/sdb
```

```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

11) Mount "/dev/sdb" on /dfm

```
// create directory to mount
sudo mkdir /dfm

// mount
sudo mount /dev/sdb /dfm
```

12) Verify

```
df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb        9.8G   37M  9.3G   1% /dfm
```

【CASE03】 Disk is NOT Ready: it is already formatted but not yet mounted on /dfm

If the disk is formatted but not yet mounted, it needs to be mounted on **/dfm**.

Now, let's check the disk information:

```
sudo lsblk -p
```

```
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
/dev/sda     202:0    0    1T  0 disk
└─/dev/sda1 202:1    0    1T  0 part /
/dev/sdb     202:80   0    1T  0 disk
```

```
sudo lsblk -f
```

```
NAME    FSTYPE   LABEL          UUID                                   MOUNTPOINT
sda
└─sda1  ext4     xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb     ext4                     d3269ceb-4418-45d0-ba68-d6b906e0595d
```

⇨ "sdb" " is formatted but Not yet mounted

5) Mount /dev/sdb on /dfm

```
// create directory to mount
sudo mkdir /dfm

// mount
sudo mount /dev/sdb /dfm
```

6) Verify

```
df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb        9.8G   37M  9.3G   1% /dfm
```

4.4.3 Permanently mount the disk

We recommend that the **customer's IT manager** sets the boot script so that **the dedicated disk** is auto-mounted when the server is booted.

If the **customer's IT manager** has not set the boot script for disk auto-mounting, you should proceed according to the command below.

*) If the settings are incorrect, booting may not be possible. The command below is for general situations, and options may differ depending on the customer's system and situation. Please refer to the "fstab" manual for details.

5) Check mount /dev/sdb on /dfm

```
sudo lsblk -f
```

```
NAME    FSTYPE    LABEL           UUID                                   MOUNTPOINT
sda
└─sda1  ext4      xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb     ext4                      d3269ceb-4418-45d0-ba68-d6b906e0595d /dfm
```

6) Edit /etc/fstab file

Add the content next to "sdb" to the new line.

```
vi /etc/fstab

~
~
UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d    /dfm   ext4   defaults      0 0
```

4.4.4 (STEP03) Create Service Directories

A separated service directory configuration is required to install and operate the Samsung DFM Module. The service account must have "**read / write / execute**" permissions to the service directory. The service directory should be mounted in a different device location from the OS installation area.

【Service Directory List】

/dfm/haproxy/config
⇒ that is where the config file is referenced when haproxy server starts.
/dfm/console/logs
⇒ that is where log files are generated when admin console server runs.
/dfm/core/logs
⇒ that is where log files are generated when core server runs.
/dfm/config
⇒ that is where the config file contains the information needed to run the DFM module.

Now, let's create each service directory.

```
sudo mkdir -p /dfm/haproxy/config
sudo mkdir -p /dfm/console/logs
sudo mkdir -p /dfm/core/logs
sudo mkdir -p /dfm/config
```

Set the service account's permission for the created service directory.

We assume that you are using the "**nightwatch**" account.

```
sudo chown -R nightwatch:nightwatch /dfm
sudo chown -R nightwatch:nightwatch /dfm/console/logs
sudo chown -R nightwatch:nightwatch /dfm/core/logs
sudo chown -R nightwatch:nightwatch /dfm/haproxy/config
sudo chown -R nightwatch:nightwatch /dfm/config
```

4.4.5 (STEP04) Install DFM Module Package

The DFM Module will either be delivered as a debian package or an rpm package tool. This package contains the following resources:

- executable binary (dfm): managed command to run DFM module
- docker images: docker image about DFM module
- haproxy config file (haproxy.cfg): config file for haproxy
- dfm config file (dfm_config.json): config file for DFM module

To install these resources, the files have to be unpacked within the following locations by the host. The files will be used during: **1) Docker Image load**, **2) initializing MySQL DB**, and **3) Copying the config file** to the service directory.

- executable binary:
  ⇒ /usr/bin/dfm

- docker images:
  ⇒ /tmp/dfm/docker-images/haproxy-debian-2.1.4.tar
  ⇒ /tmp/dfm/docker-images/minio-RELEASE.2020-06-01T17-28-03Z.tar
  ⇒ /tmp/dfm/docker-images/mysql-enterprise-server-8.0.20.tar

⇒ /tmp/dfm/docker-images/dfm-console-xxx.tar

⇒ /tmp/dfm/docker-images/dfm-core-xxx.tar

- haproxy config file:

⇒ /tmp/dfm/app-server/haproxy-config/haproxy.cfg

- dfm config file:

⇒ /tmp/dfm/app-server/dfm_config.json

The following is a command showing how to install the debian package:

```
sudo dpkg -i xxx.deb

example)
sudo dpkg -i sec-dfm_1.0.1.5.deb
Selecting previously unselected package dfm.
(Reading database ... 973294 files and directories currently
installed.)
Preparing to unpack sec-dfm_1.0.1.5.deb ...
Unpacking dfm (1.0.1.5) ...
Setting up dfm (1.0.1.5) ...
```

Next, check if the necessary files exist:

```
1) check dfm file
ls /usr/bin/dfm
/usr/bin/dfm

2) check docker images
ls /tmp/dfm/docker-images/ -l
total 971552
-rw-rw-r-- 1 dfm-console-1.0.1.4.tar
-rw-rw-r-- 1 dfm-core-1.0.1.4.tar
-rw-rw-r-- 1 haproxy-debian-2.1.4.tar
-rw-rw-r-- 1 minio-RELEASE.2020-06-01T17-28-03Z.tar
-rw-rw-r-- 1 mysql-enterprise-server-8.0.20.tar


4) check haproxy config file : haproxy.cfg
ls /tmp/dfm/ha/app-server/haproxy-config/ -l
total 12
drwxrwxr-x errors
-rw-rw-r--    haproxy.cfg

5) dfm config file : dfm_config.json
```

```
ls /tmp/dfm/ha/app-server/dfm_config.json
/tmp/dfm/ha/app-server/dfm_config.json
```

4.4.6 (STEP05) Load Docker Image

Next, register the Docker Images that were unpacked at "**/tmp/dfm/docker-images**". The loaded Docker Images are used when the container is driven. The following shows how to load Docker Images required for DFM Core/Cosole server using Docker commands:

```
docker load < /tmp/dfm/docker-images/haproxy-debian-2.1.4.tar
docker load < /tmp/dfm/docker-images/dfm-core-{version}.tar
docker load < /tmp/dfm/docker-images/dfm-console-{version}.tar
```

Next, check if the 3 Docker images were loaded. Use the "Docker Images" command:

Example)

docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
dfm-core	1.0.1.5	62c236d15854	About an hour ago	124MB
dfm-console	1.0.1.5	d63dec253531	About an hour ago	169MB
haproxytech/haproxy-debian	2.1.4	88bf690bd83f	6 days ago	99.7MB

4.4.7 (STEP06) Copy Configuration files

After loading the Docker images, copy the following configuration files into the service directory from the unpacked resources directory.
We assume that you are using the "**nightwatch**" account.
- **copy haproxy config file**:

```
// copy configuration file
cp /tmp/dfm/ha/app-server/haproxy-config/haproxy.cfg /dfm/haproxy/config

// copy error files
cp -rf /tmp/dfm/ha/app-server/haproxy-config/errors/ /dfm/haproxy/config

//Set the service account's permission to the configuration file.
sudo chown -R nightwatch:nightwatch /dfm/haproxy/config
```

- **copy dfm config file**:

```
// copy configuration file
cp /tmp/dfm/ha/app-server/dfm_config.json /dfm/config

//Set the service account's permission to the configuration file.
sudo chown -R nightwatch:nightwatch /dfm/config
```

4.1. Check Pre-Config

4.4.8 (STEP07) Set-up Configuration

In this step, we will set up the initial configuration information needed for the DFM module to run as a Container.
Set the configuration reffering to the setting in .

【Configuration List】

- host_ip: Static IP for DFM server.
- listen_port: External listen port at server for DFM module to be accessed.
- listen_scheme: url scheme(http or https) for DFM module to be accessed.
- cluster_server_name: app
- cluster_service_address: address to access admin console (ip or address)
- cluster_service_port: port to access admin console
- cluster_service_scheme: protocol to access admin console (http or https)
- cluster_minio_access_address: firmware storage server address(ip or address)
- cluster_minio_access_port: firmware storage server port
- cluster_minio_access_scheme: firmware storage server protocol (http or https)
- cluster_mysql_access_address: database server address(ip or address)
- cluster_mysql_access_port: database server port

The following shows the commands:

```
dfm cluster config set host_ip=192.168.1.52
dfm cluster config set listen_port=80
dfm cluster config set listen_scheme=http
dfm cluster config set cluster_server_name=app
dfm cluster config set cluster_service_address=181.107.61.233
dfm cluster config set cluster_service_port=6380
dfm cluster config set cluster_service_scheme=http
dfm cluster config set cluster_minio_access_address=192.168.0.11
dfm cluster config set cluster_minio_access_port=9000
dfm cluster config set cluster_minio_access_scheme=http
dfm cluster config set cluster_mysql_access_address=192.168.0.30
dfm cluster config set cluster_mysql_access_port=3306
```

Next, check if the configured value is correct. Use the "**dfm cluster config get {*key*}**" command:

Example)

```
dfm cluster config get host_ip
192.168.1.52

dfm cluster config get listen_port
80

dfm cluster config get listen_scheme
http

dfm cluster config get cluster_server_name
app

dfm cluster config get cluster_service_address
181.107.61.233

dfm cluster config get cluster_service_port
6380

dfm cluster config get cluster_service_scheme
http
```

```
dfm cluster config get cluster_minio_access_address
192.168.0.11

dfm cluster config get cluster_minio_access_port
9000

dfm cluster config get cluster_minio_access_scheme
http

dfm cluster config get cluster_mysql_access_address
192.168.0.30

dfm cluster config get cluster_mysql_access_port
3306
```

4.4.9 (STEP08) Configure HAProxy

In this step, set up for communication between minio server and DFM core/console server. Change the value according to cluster_minio_access_address and cluster_minio_access_port set in minio server.

【UseCase1】When using ssl connection to Firmware storage server (minio)

vi /dfm/haproxy/config/haproxy.cfg

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
backend dfmMinioReplaceHostBackend
   mode http
   option httpchk GET /minio/health/live
   http-check expect status 200
   default-server inter 5s fall 3 rise 2
   # if minio server use ssl
   server dfm-minio 192.168.0.11:9000 ssl verify none check
   # otherwise
   #server dfm-minio 192.168.0.11:9000 check
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

【UseCase2】When NOT using ssl connection to Firmware storage server (minio)

vi /dfm/haproxy/config/haproxy.cfg

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
backend dfmMinioReplaceHostBackend
   mode http
   option httpchk GET /minio/health/live
   http-check expect status 200
   default-server inter 5s fall 3 rise 2
   # if minio server use ssl
   #server dfm-minio 192.168.0.11:9000 ssl verify none check
   # otherwise
   #server dfm-minio 192.168.0.11:9000 check
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

4.4.10 (STEP09) Create Container Network

The DFM Module is a process executed on a container basis, creating the Docker network required for communications among containers.

To create a network, use the following command:

```
dfm network create
```

【Validation】

Run the following command to see if "dfm-network" is visible.

```
dfm network ls
NETWORK ID          NAME                DRIVER              SCOPE
~~~~~~~~~~~
e2697cd6621a        dfm-network         bridge              local
~~~~~~~~~~~
```

4.4.11 (STEP10) Start-up DFM Core/Console Server

In this stage, the installer starts the storage server that manages the firmware binary.

The command to run the Core, Console and HA proxy containers is as follows:

```
dfm cluster start dfm-core
dfm cluster start dfm-console
dfm cluster start dfm-proxy
```

【Validation】

Make sure the 3 containers are in a healthy state. It may take some time until its state is healthy.

```
docker ps -a

Example)
$ docker ps -a
CONTAINER ID    ~    STATUS                          ~    NAMES
~~
cbdd8728e551    Up 4 seconds (health: starting)          dfm-core
c1e2cc5634a8    Up 4 seconds (health: starting)          dfm-console
c88feb369b2c    Up 4 seconds (health: starting)          dfm-proxy
~~
$
$ docker ps -a
CONTAINER ID    ~    STATUS                          ~    NAMES
~~
cbdd8728e551    Up 2 minutes (healthy)                   dfm-core
c1e2cc5634a8    Up 2 minutes (healthy)                   dfm-console
c88feb369b2c    Up 2 minutes (healthy)                   dfm-proxy

~~
```

4.5. WEB Server

4.5.1 (STEP01) Create Service Account and Login

The DFM Module is logged in with **a dedicated service account** and operates with the privileges of the account. Therefore, the dedicated service account has to be created in the server. The service account also needs the "**sudo**" privilege as a Docker requirement for command permissions. Ensure you add your service account into the Docker group.
We recommend that you create a service account before you start the installation.

The below shows you how to add your service account into the Docker group:

We assume that you are using the "**nightwatch**" account.

```
$ sudo usermod -aG docker {your-user}

Example)
  sudo usermod -aG docker nightwatch
```

4.5.2 (STEP02) Prepare "Disk partition & mount" for DFM modules

DFM module is installed in and operates in the below directory on the **dedicated disk**.

Therefore, we should check if the dedicated disk exists and the "partition & mount" is ready, in case the customer has not worked with the disk partition for the DFM module before.

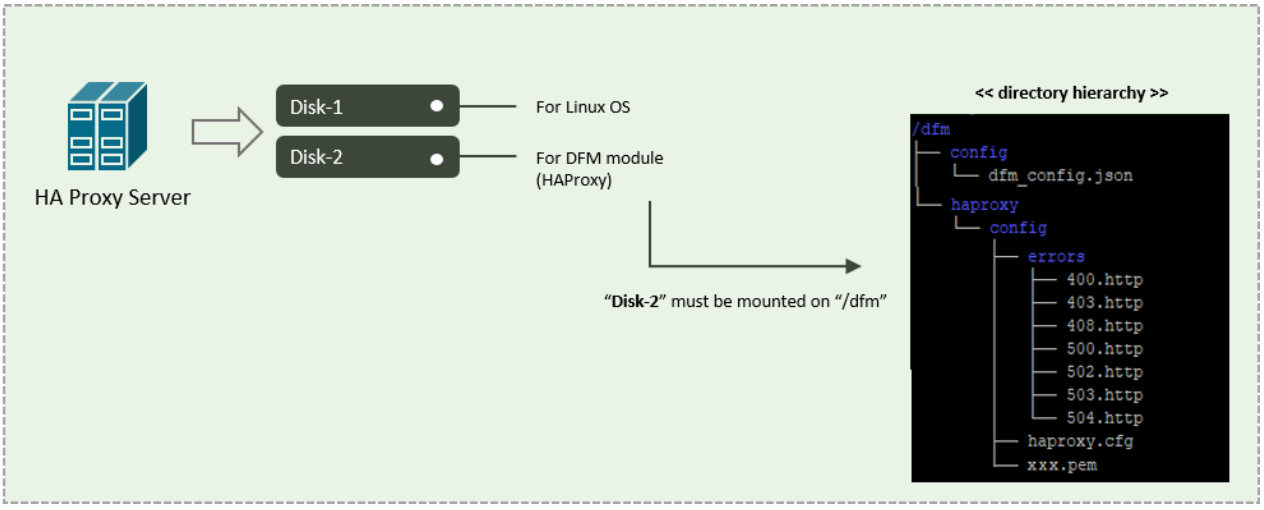

Fig 4-9 **An Disk Partitions for DMF Module on WEB server**

For example, we assume that two disks ("sda" and "sdb") exist.

【CASE01】 Disk is Ready

If the disks exist, we don't need to format and mount them.

Now, let's check the disk information:

```
sudo lsblk -p
```

```
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
/dev/sda     202:0    0    1T  0 disk
└─/dev/sda1  202:1    0    1T  0 part /
/dev/sdb     202:80   0    1T  0 disk
```

```
sudo lsblk -f
```

```
NAME    FSTYPE   LABEL          UUID                                   MOUNTPOINT
sda
└─sda1  ext4     xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb     ext4                    d3269ceb-4418-45d0-ba68-d6b906e0595d /dfm
```

⇨ “sdb” is already formatted and mounted on **/dfm**

```
sudo file -s /dev/sdb
```

```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

【CASE02】 Disk is NOT Ready: it is not formatted

If the disk is not ready, it needs to be formatted and mounted on **/dfm**.

Now, let’s check the disk information:

```
sudo lsblk -p
```

```
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
/dev/sda     202:0    0    1T  0 disk
└─/dev/sda1 202:1    0    1T  0 part /
/dev/sdb     202:80   0    1T  0 disk
```

```
sudo lsblk -f
```

```
NAME    FSTYPE   LABEL          UUID                                   MOUNTPOINT
sda
└─sda1 ext4      xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb
```

⇨ “sdb” is NOT formatted

```
sudo file -s /dev/sdb
```

```
/dev/sdb: data
```

⇨ This means that the disk needs to be formatted

13) Format with ext4 file-system

```
sudo file -s /dev/sdb
```

```
sudo mkfs -t ext4 /dev/sdb
mke2fs 1.44.1 (24-Mar-2018)
Creating filesystem with 2621440 4k blocks and 655360 inodes
Filesystem UUID: d3269ceb-4418-45d0-ba68-d6b906e0595d
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

14) Check if the disk is formatted

```
sudo mkfs -t ext4 /dev/sdb
```

```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

15) Mount "/dev/sdb" on /dfm

```
// create directory to mount
sudo mkdir /dfm

// mount
sudo mount /dev/sdb /dfm
```

16) Verify

```
df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb        9.8G   37M  9.3G   1% /dfm
```

【CASE03】 Disk is NOT Ready: it is already formatted but not yet mounted on /dfm

If the disk is formatted but not yet mounted, it needs to be mounted on **/dfm**.

Now, let's check the disk information:

```
sudo lsblk -p
```

```
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
/dev/sda     202:0    0    1T  0 disk
└─/dev/sda1 202:1    0    1T  0 part /
/dev/sdb     202:80   0    1T  0 disk
```

```
sudo lsblk -f
```

```
NAME    FSTYPE   LABEL          UUID                                   MOUNTPOINT
sda
└─sda1  ext4     xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb     ext4                     d3269ceb-4418-45d0-ba68-d6b906e0595d
```

⇨ "sdb" " is formatted but Not yet mounted

7) Mount /dev/sdb on /dfm

```
// create directory to mount
sudo mkdir /dfm

// mount
sudo mount /dev/sdb /dfm
```

8) Verify

```
df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb        9.8G   37M  9.3G   1% /dfm
```

4.5.3 Permanently mount the disk

We recommend that the **customer's IT manager** sets the boot script so that **the dedicated disk** is auto-mounted when the server is booted.

If the **customer's IT manager** has not set the boot script for disk auto-mounting, you should proceed according to the command below.

*) If the settings are incorrect, booting may not be possible. The command below is for general situations, and options may differ depending on the customer's system and situation. Please refer to the "fstab" manual for details.

7) Check mount /dev/sdb on /dfm

```
sudo lsblk -f
```

```
NAME    FSTYPE   LABEL          UUID                                   MOUNTPOINT
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
sda
└─sda1  ext4     xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb     ext4                     d3269ceb-4418-45d0-ba68-d6b906e0595d /dfm
```

8) Edit /etc/fstab file

Add the content next to "sdb" to the new line.

```
vi /etc/fstab

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~
UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d    /dfm   ext4   defaults      0 0
```

4.5.4 (STEP03) Create Service Directories

A separated service directory configuration is required to install and operate the Samsung DFM Module. The service account must have "**read / write / execute**" permissions to the service directory. The service directory should be mounted in a different device location from the OS installation area.

【Service Directory List】

/dfm/haproxy/config

⇒ that is where the config file is referenced when haproxy server starts.

/dfm/config

⇒ that is where the config file contains the information needed to run the DFM module.

Now, let's create each service directory.

```
sudo mkdir -p /dfm/haproxy/config
sudo mkdir -p /dfm/config
```

Set the service account's permission for the created service directory.

We assume that you are using the "**nightwatch**" account.

```
sudo chown -R nightwatch:nightwatch /dfm
sudo chown -R nightwatch:nightwatch /dfm/haproxy
```

```
sudo chown -R nightwatch:nightwatch /dfm/haproxy/config
sudo chown -R nightwatch:nightwatch /dfm/config
```

4.5.5 (STEP04) Install DFM Module Package

The DFM Module will either be delivered as a debian package or an rpm package tool. This package contains the following resources:

- executable binary (dfm): managed command to run DFM module
- docker images: docker image about DFM module
- haproxy config file (haproxy.cfg): config file for haproxy
- dfm config file (dfm_config.json): config file for DFM module

To install these resources, the files have to be unpacked within the following locations by the host. The files will be used during: **1) Docker Image load**, **2) initializing MySQL DB**, and **3) Copying the config file** to the service directory.

- executable binary:
  ⇒ /usr/bin/dfm

- docker images:
  ⇒ /tmp/dfm/docker-images/haproxy-debian-2.1.4.tar
  ⇒ /tmp/dfm/docker-images/minio-RELEASE.2020-06-01T17-28-03Z.tar
  ⇒ /tmp/dfm/docker-images/mysql-enterprise-server-8.0.20.tar
  ⇒ /tmp/dfm/docker-images/dfm-console-xxx.tar
  ⇒ /tmp/dfm/docker-images/dfm-core-xxx.tar

- haproxy config file:
  ⇒ /tmp/dfm/ha/web-server/haproxy-config/haproxy.cfg

- dfm config file:
  ⇒ /tmp/dfm/ha/web-server/dfm_config.json

The following is a command showing how to install the debian package:

```
sudo dpkg -i xxx.deb

example)
sudo dpkg -i sec-dfm_1.0.1.5.deb
Selecting previously unselected package dfm.
(Reading database ... 973294 files and directories currently
installed.)
Preparing to unpack sec-dfm_1.0.1.5.deb ...
Unpacking dfm (1.0.1.5) ...
Setting up dfm (1.0.1.5) ...
```

Next, check if the necessary files exist:

```
1) check dfm file
```

```
ls /usr/bin/dfm
/usr/bin/dfm

2) check docker images
ls /tmp/dfm/docker-images/ -l
total 971552
-rw-rw-r-- 1 dfm-console-1.0.1.4.tar
-rw-rw-r-- 1 dfm-core-1.0.1.4.tar
-rw-rw-r-- 1 haproxy-debian-2.1.4.tar
-rw-rw-r-- 1 minio-RELEASE.2020-06-01T17-28-03Z.tar
-rw-rw-r-- 1 mysql-enterprise-server-8.0.20.tar


4) check haproxy config file : haproxy.cfg
ls /tmp/dfm/ha/web-server/haproxy-config/ -l
total 12
drwxrwxr-x errors
-rw-rw-r--    haproxy.cfg

5) dfm config file : dfm_config.json
ls /tmp/dfm/ha/web-server/dfm_config.json
/tmp/dfm/ha/app-server/dfm_config.json
```

4.5.6 (STEP05) Load Docker Image

Next, register the Docker Images that were unpacked at "**/tmp/dfm/docker-images**". The loaded Docker Images are used when the container is driven. The following shows how to load Docker Image required for WEB server using Docker commands:

```
docker load < /tmp/dfm/docker-images/haproxy-debian-2.1.4.tar
```

Next, check if the Docker image was loaded. Use the "Docker Images" command:

Example)

```
docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
haproxytech/haproxy-debian	2.1.4	88bf690bd83f	6 days ago	99.7MB

4.5.7 (STEP06) Copy Configuration files

After loading the Docker images, copy the following configuration files into the service directory from the unpacked resources directory.
We assume that you are using the "**nightwatch**" account.

- **copy haproxy config file**:

```
// copy configuration file
```

```
cp /tmp/dfm/ha/web-server/haproxy-config/haproxy.cfg /dfm/haproxy/config

// copy error files
cp -rf /tmp/dfm/ha/web-server/haproxy-config/errors/ /dfm/haproxy/config

//Set the service account's permission to the configuration file.
sudo chown -R nightwatch:nightwatch /dfm/haproxy/config
```

- **copy dfm config file**:

```
// copy configuration file
cp /tmp/dfm/ha/web-server/dfm_config.json /dfm/config

//Set the service account's permission to the configuration file.
sudo chown -R nightwatch:nightwatch /dfm/config
```

4.5.8 (STEP07) Set-up Configuration

In this step, we will set up the initial configuration information needed for the DFM module to run as a Container.

【Configuration List】

- host_ip: Static IP for DFM server.
- listen_port: External listen port at server for DFM module to be accessed.
- listen_scheme: url scheme(http or https) for DFM module to be accessed.
- access_address: domain-based or ip-based
- access_scheme: http or https
- access_port: public port
- public_endpoint: {access_scheme}://{access_address}:{access_port}

The following is **an example** of how to execute the command to set the above configurations:

The following shows the commands:

```
dfm cluster config set host_ip=192.168.1.52
dfm cluster config set listen_port=80
dfm cluster config set listen_scheme=http
dfm cluster config set access_address=181.107.61.233
dfm cluster config set access_scheme=http
dfm cluster config set access_port=6380
```

Next, check if the configured value is correct. Use the "**dfm cluster config get {*key*}**" command:

```
Example)
dfm cluster config get host_ip
   192.168.1.52

dfm cluster config get listen_port
   80

dfm cluster config get listen_scheme
```

```
    http

dfm cluster config get access_address
    181.107.61.233

dfm cluster config get access_scheme
    http

dfm cluster config get access_port
    6380
```

4.5.9 (STEP08) Configure HAProxy

In this step, you will set up communication to the DFM core/console server.
Change the value according to listen_ip and listen_port set in the DFM core/console server.

【UseCase1】One DFM core/console server is used.

1) If you are **not** using SSL to connect:
Set up to “server dfm-coreproxy_1 **{listen_ip}**:**{listen_port}** check”

```
vi /dfm/haproxy/config/haproxy.cfg

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
backend dfmCoreProxyBackend
    balance roundrobin
    mode http
    option httpchk GET /admin/health/live
    http-check expect status 200
    default-server inter 5s fall 3 rise 2
    cookie SERVER insert indirect nocache

    # if core server use ssl
    #server dfm-coreproxy_1 192.168.0.3:443 ssl verify none check  cookie dfm-coreproxy_1
    #server dfm-coreproxy_2 192.168.0.4:443 ssl verify none check  cookie dfm-coreproxy_2
    # otherwise
    server dfm-coreproxy_1 192.168.0.3:80 check  cookie dfm-coreproxy_1
    #server dfm-coreproxy_2 192.168.0.4:80 check  cookie dfm-coreproxy_2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

2) If you are using SSL to connect:
Set up to “server dfm-coreproxy_1 **{listen_ip}**:**{listen_port}** ssl verify none check”

```
vi /dfm/haproxy/config/haproxy.cfg

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
backend dfmCoreProxyBackend
    balance roundrobin
    mode http
    option httpchk GET /admin/health/live
    http-check expect status 200
    default-server inter 5s fall 3 rise 2
    cookie SERVER insert indirect nocache

    # if core server use ssl
```

```
   server dfm-coreproxy_1 192.168.0.3:443 ssl verify none check cookie dfm-coreproxy_1
   #server dfm-coreproxy_2 192.168.0.4:443 ssl verify none check cookie dfm-coreproxy_2
   # otherwise
   #server dfm-coreproxy_1 192.168.0.3:80 check cookie dfm-coreproxy_1
   #server dfm-coreproxy_2 192.168.0.4:80 check cookie dfm-coreproxy_2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

【UseCase2】Three DFM core/console servers are used.

1) If you are **not** using SSL to connect:
Set up to “server dfm-coreproxy_{number} **{listen_ip}**:**{listen_port}** check”

vi /dfm/haproxy/config/haproxy.cfg

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
backend dfmCoreProxyBackend
   balance roundrobin
   mode http
   option httpchk GET /admin/health/live
   http-check expect status 200
   default-server inter 5s fall 3 rise 2
   cookie SERVER insert indirect nocache

   # if core server use ssl
   #server dfm-coreproxy_1 192.168.0.3:443 ssl verify none check cookie dfm-coreproxy_1
   #server dfm-coreproxy_2 192.168.0.4:443 ssl verify none check cookie dfm-coreproxy_2
   # otherwise
   server dfm-coreproxy_1 192.168.0.3:80 check cookie dfm-coreproxy_1
   server dfm-coreproxy_2 192.168.0.4:80 check cookie dfm-coreproxy_2
   server dfm-coreproxy_3 192.168.0.5:80 check cookie dfm-coreproxy_3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

2) If you are using SSL to connect:
Set up to “server dfm-coreproxy_{number} **{listen_ip}**:**{listen_port}** ssl verify none check”

vi /dfm/haproxy/config/haproxy.cfg

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
backend dfmCoreProxyBackend
   balance roundrobin
   mode http
   option httpchk GET /admin/health/live
   http-check expect status 200
   default-server inter 5s fall 3 rise 2
   cookie SERVER insert indirect nocache

   # if core server use ssl
   server dfm-coreproxy_1 192.168.0.3:443 ssl verify none check dfm-coreproxy_1
   server dfm-coreproxy_2 192.168.0.4:443 ssl verify none check dfm-coreproxy_2
   server dfm-coreproxy_3 192.168.0.5:443 ssl verify none check dfm-coreproxy_3
   # otherwise
   #server dfm-coreproxy_1 192.168.0.3:80 check dfm-coreproxy_1
   #server dfm-coreproxy_2 192.168.0.4:80 check dfm-coreproxy_2
   #server dfm-coreproxy_3 192.168.0.5:80 check dfm-coreproxy_3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Set up communication to the Firmware Storage(minio) server
Change the value according to cluster_minio_access_address and cluster_minio_access_port set in the Firmware Storage(minio) server.
Set up to “http-request set-header Host

{cluster_minio_access_address}:{cluster_minio_acces_port}"

【UseCase1】If you are using SSL to connect:
Set up to "server dfm-minioproxy_**{cluster_minio_access_address}:{cluster_minio_acces_port}** ssl verify none check"

vi /dfm/haproxy/config/haproxy.cfg

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
backend dfmMinioProxyBackend
   mode http
   option httpchk GET /minio/health/live

   http-check expect status 200
   default-server inter 5s fall 3 rise 2
   http-request set-header Host 192.168.0.7:9900

   #if minio server use ssl
   server dfm-minioproxy 192.168.0.7:9000 ssl verify none check
   #otherwise
   #server dfm-minioproxy 192.168.0.7:9000 check
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

【UseCase2】If you are **not** using SSL to connect:
Set up to "server dfm-minioproxy_**{cluster_minio_access_address}:{cluster_minio_acces_port}** check"

vi /dfm/haproxy/config/haproxy.cfg

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
backend dfmMinioProxyBackend
   mode http
   option httpchk GET /minio/health/live

   http-check expect status 200
   default-server inter 5s fall 3 rise 2
   http-request set-header Host 192.168.0.7:9000

   #if minio server use ssl
   #server dfm-minioproxy 192.168.0.7:9000 ssl verify none check
   #otherwise
   server dfm-minioproxy 192.168.0.7:9000 check
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

4.5.10 (STEP09) Create Container Network

The DFM Module is a process executed on a container basis, creating the Docker network required for communications among containers.

To create a network, use the following command:

dfm network create

【Validation】

Run the following command to see if "dfm-network" is visible.

```
dfm network ls
NETWORK ID          NAME                DRIVER              SCOPE
~~~~~~~~~~~
e2697cd6621a        dfm-network         bridge              local
~~~~~~~~~~~
```

4.5.11 (STEP10) Start up web server

In this step, the installer starts the storage server that manages the firmware binary.
The command to run HA proxy containers is as follows:

```
dfm cluster start dfm-proxy
```

【Validation】

Make sure the 3 containers are in a healthy state. It may take some time until its state is healthy.

```
docker ps -a

Example)
$ docker ps -a
CONTAINER ID    ~    STATUS                          ~    NAMES
~~
c88feb369b2c    Up 4 seconds (health: starting)          dfm-proxy
~~
$

$ docker ps -a
CONTAINER ID    ~    STATUS                          ~    NAMES
~~
c88feb369b2c    Up 2 minutes (healthy)                   dfm-proxy
~~
```

4.6. Configure SSL

In this step, you can set the configuration for SSL on each server when you want to communicate using SSL between servers.

4.6.1 DB(MySQL) Server

MySQL communicates with SSL by default. No other options are provided due to MySQL policy.

4.6.2 Firmware Storage(minio) Server

The following steps are to configure the SSL settings of the firmware storage(minio) server.

1) Certificate preparation on the **firmware storage(minio) server.**
If you are using a self-signed certificate, you can create it using the command below.

```
openssl req -new -x509 -nodes -days 365 -keyout private.key -out public.crt
```

2) Copy the certificate and restart the container(minio) on the **firmware storage(minio) server.**

- Certificate copy and setup

```
cp private.key /dfm/minio/config/cert
cp public.crt /dfm/minio/config/cert
sudo chown nightwatch:nightwatch /dfm/minio/config/private.key
sudo chown nightwatch:nightwatch /dfm/minio/config/public.crt
sudo chmod 600 /dfm/minio/config/private.key
sudo chmod 600 /dfm/minio/config/public.crt
```

- Set the DFM configuration

Please ensure you enter the following config value carefully:

--log-driver json-file --log-opt max-size=5m --log-opt max-file=10 --env='MINIO_BROWSER=off' --restart=always --security-opt=no-new-privileges --health-cmd='curl --fail -k https://127.0.0.1:9000/minio/health/live || exit 1'

```
dfm cluster config set opt_minio_run_cmd="--log-driver json-file --log-opt max-size=5m --log-opt max-file=10 --env='MINIO_BROWSER=off' --restart=always --security-opt=no-new-privileges --health-cmd='curl --fail -k https://127.0.0.1:9000/minio/health/live || exit 1'"
```

- Restart the container:

```
dfm cluster restart dfm-minio
```

- Check the service:

```
docker logs dfm-minio


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Endpoint:  https://100.0.0.2:9000  https://127.0.0.1:9000


Object API (Amazon S3 compatible):
   Go:       https://docs.min.io/docs/golang-client-quickstart-guide
   Java:     https://docs.min.io/docs/java-client-quickstart-guide
   Python:   https://docs.min.io/docs/python-client-quickstart-guide
   JavaScript: https://docs.min.io/docs/javascript-client-quickstart-guide
   .NET:     https://docs.min.io/docs/dotnet-client-quickstart-guide
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

3) Set up the “**Web server”** and restart the container (HA proxy):

- Set up the web server

Modify “ip” and “port” to match the firmware storage server settings.

(Be careful when changing the value of “http-request set-header Host 192.168.0.7:9900”.)

```
vi /dfm/haproxy/config/haproxy.cfg
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
backend dfmMinioProxyBackend
   mode http
   option httpchk GET /minio/health/live

   http-check expect status 200
   default-server inter 5s fall 3 rise 2
```

```
    http-request set-header Host 192.168.0.7:9900

    #if minio server use ssl
    server dfm-minioproxy 192.168.0.7:9000 ssl verify none check
    #otherwise
    #server dfm-minioproxy 192.168.0.7:9000 check
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

- Restart the container (HA proxy)

```
dfm cluster restart dfm-proxy
```

4) Set up the "**DFM core/console server"** and restart the containers (core, HA proxy)

```
vi /dfm/haproxy/config/haproxy.cfg
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
backend dfmMinioReplaceHostBackend
    mode http
    option httpchk GET /minio/health/live
    http-check expect status 200
    default-server inter 5s fall 3 rise 2
    # if minio server use ssl
    server dfm-minio 192.168.0.11:9000 ssl verify none check
    # otherwise
    #server dfm-minio 192.168.0.11:9000 check
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

- Set up the config

【Configuration List】

```
  - cluster_minio_access_address: Firmware storage server address(ip or address)
  - cluster_minio_access_port: Firmware storage server port
  - cluster_minio_access_scheme: https
```

Ex) When Firmware storage server "ip" is set to 192.168.0.11 and "port" is set to 9000.

The following shows the commands:

```
Example)
dfm cluster config set cluster_minio_access_address=192.168.0.11
dfm cluster config set cluster_minio_access_port=9000
dfm cluster config set cluster_minio_access_scheme=https
```

Next, check if the configured value is correct. Use the "**dfm cluster config get {*key*}**" command:

```
Example)
dfm cluster config get cluster_minio_access_address
    192.168.0.11

dfm cluster config get cluster_minio_access_port
    9000

dfm cluster config get cluster_minio_access_scheme
    https
```

- Restart the containers (core, HA proxy)

```
dfm cluster restart dfm-core
dfm cluster restart dfm-proxy
```

4.6.3 DFM Core/Console Server

1) Certificate preparation on the DFM core/console server

The following assumes that the "**example-sec-fota.net.pem**" file is the public certificate issued by the customer. The public certificate must be copied into haproxy's config folder, and the "haproxy.cfg" file must be edited to change the bind port information and certificate configuration.

- The **crt** parameter identifies the location of the **PEM-formatted** SSL certificate
- **This certificate file** should contain **both the public certificate and private key**
- How to generate the unified certificate for the issued certificate file:

  For example: we assume that you have the below 4 files and the domain's name is **onptest.samsung-efota-test**

  · cert.pem

  · chain.pem

  · fullchain.pem: cert.pem and chain.pem combined

  · privkey.pem

  ⇨ sudo -E bash -c 'cat fullchain.pem privkey.pem > onptest.samsung-efota-test.pem'

'**onptest.samsung-efota-test.pem**' is the unified certificate file

2) Copy the certificate and restart the container (HA proxy) on the DFM core/console server

Be sure to uncomment the "**bind *:443 …**" line in the haproxy.cfg file:

```
cp example-sec-fota.net.pem /dfm/haproxy/config
sudo chown nightwatch:nightwatch /dfm/haproxy/config/example-sec-fota.net.pem
sudo chmod 600 /dfm/haproxy/config/example-sec-fota.net.pem
vi /dfm/haproxy/config/haproxy.cfg
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
frontend fe_web
    bind *:80
    bind *:443 ssl crt /usr/local/etc/haproxy/example-sec-fota.net.pem
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

- Restart the container (HA proxy)

```
dfm cluster restart dfm-proxy
```

3) Set up the "**Web server"** and restart the container (HA proxy)

Set up to "server dfm-coreproxy_{number} **{listen_ip}**:**{listen_port}** ssl verify none check"

```
vi /dfm/haproxy/config/haproxy.cfg

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
backend dfmCoreProxyBackend
```

```
    balance roundrobin
    mode http
    option httpchk GET /admin/health/live
    http-check expect status 200
    default-server inter 5s fall 3 rise 2
    cookie SERVER insert indirect nocache

    # if core server use ssl
    server dfm-coreproxy_1 192.168.0.3:443 ssl verify none check cookie dfm-coreproxy_1
    server dfm-coreproxy_2 192.168.0.4:443 ssl verify none check cookie dfm-coreproxy_2
    server dfm-coreproxy_3 192.168.0.5:443 ssl verify none check cookie dfm-coreproxy_3
    # otherwise
    #server dfm-coreproxy_1 192.168.0.3:80 check cookie dfm-coreproxy_1
    #server dfm-coreproxy_2 192.168.0.4:80 check cookie dfm-coreproxy_2
    #server dfm-coreproxy_3 192.168.0.5:80 check cookie dfm-coreproxy_3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

- Restart the container (HA proxy)

```
dfm cluster restart dfm-proxy
```

4.6.4 WEB Server

If the external connection type is "**https**", the customer must prepare **1)** the access domain they were issued, **2)** a public certificate for the domain in advance. If the customer is using IP address-based addressing rather than DNS, **this step may be skipped**.

If "ingress_url_scheme" is set to "**https**" on the "***4.7. (STEP07) Set-up Configuration***", this step must be completed.

I. HTTPS Handling

There are two possibilities for TLS/SSL Termination:

1) On Customer's Load Balancer (Proxy)

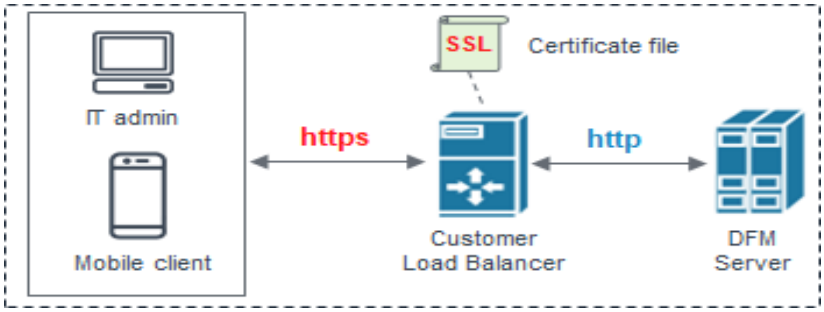

Fig 4-10 On Customer's Load Balancer (Proxy)

In this case, the customer's IT manager will operate "public certificate" on its own Load Balancer.

1. Web server configuration

Be careful to comment the "**bind *:443 …**" line in the haproxy.cfg file:

```
vi /dfm/haproxy/config/haproxy.cfg
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
frontend fe_web
    bind *:80
```

```
    #bind *:443 ssl crt /usr/local/etc/haproxy/example-sec-fota.net.pem
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

2. DFM core/console server configuration

Be careful to uncomment "**#http-response replace-valueLocation (.*) https://%[var(txn.host)]/admin/ if logout_path_set**" line in the haproxy.cfg file:

```
vi /dfm/haproxy/config/haproxy.cfg
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

backend dfmConsoleBackend
        mode http
        acl logout_path_set var(txn.path) path /admin/logout
        http-request set-header X-Forwarded-Port
        %[dst_port]
        http-request add-header X-Forwarded-Proto https if { ssl_fc }

        option httpchk GET
        /admin/health/livehttp-check expect
        status 200
        default-server inter 5s fall 3 rise 2

        # if DFM Server is behind customer's Load-Balancer and also customer's Load-Balancer provides ssl termination.
        http-response replace-value Location (.*) https://%[var(txn.host)]/admin/ if logout_path_set
        # otherwise
        #http-response replace-value Location (.*) %[var(txn.scheme)]://%[var(txn.host)]/admin/ if logout_path_set

        server dfm-console dfm-console:10050 check resolvers docker init-addr libc,none
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Since the DFM server can no longer add "Location Header" in response, the **Customer's Load Balancer must provide the corresponding function**. If the Load Balancer does not provide this function, the user cannot log out after logging into the "admin console webpage" on the DFM.

2) On DFM Server

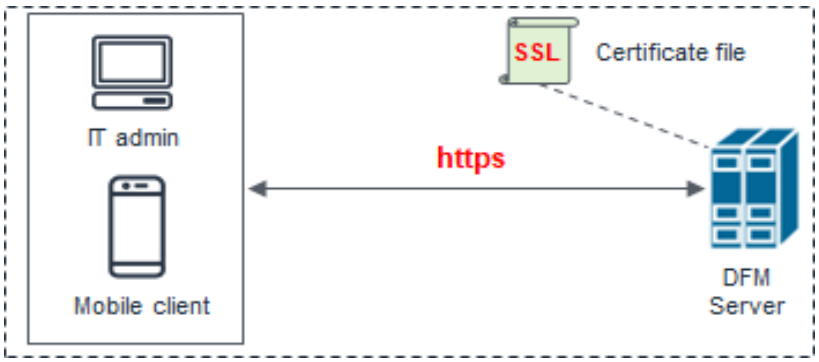

Fig 4-11 On DFM Server

In this case, we need to configure TLS/SSL on our DFM Server. Follow the below steps to do

so.

The following assumes that the "**example-sec-fota.net.pem**" file is the public certificate issued by the customer. The public certificate must be copied into haproxy's config folder, and the "haproxy.cfg" file must be edited to change the bind port information and certificate configuration.

- The **crt** parameter identifies the location of the **PEM-formatted** SSL certificate
- **This certificate file** should contain **both the public certificate and private key**
- How to generate the unified certificate for the issued certificate file:

  For example: we assume that you have the below 4 files and the domain's name is **onptest.samsung-efota-test**

  · cert.pem

  · chain.pem

  · fullchain.pem: cert.pem and chain.pem combined

  · privkey.pem

  ⇨ sudo -E bash -c 'cat fullchain.pem privkey.pem > onptest.samsung-efota-test.pem'

  ⇨ '**onptest.samsung-efota-test.pem**' is the unified certificate file

We assume that you are using the "**nightwatch**" account:

Be careful to uncomment the "**bind *:443 …**" line and uncomment the "**#http-response replace-value Location (.*) https://%[var(txn.host)]/admin/ if logout_path_set**" line in the haproxy.cfg file:

1. Web server configuration

Be careful to uncomment the "**bind *:443 …**" line in the haproxy.cfg file:

```
cp example-sec-fota.net.pem /dfm/haproxy/config
sudo chown nightwatch:nightwatch /dfm/haproxy/config/example-sec-fota.net.pem
sudo chmod 600 /dfm/haproxy/config/example-sec-fota.net.pem
vi /dfm/haproxy/config/haproxy.cfg
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
frontend fe_web
    bind *:80
    bind *:443 ssl crt /usr/local/etc/haproxy/example-sec-fota.net.pem
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

2. DFM core/console server configuration

Be careful to uncomment the "**#http-response replace- value Location (.*) https://%[var(txn.host)]/admin/ if logout_path_set**" line in the haproxy.cfg file:

```
vi /dfm/haproxy/config/haproxy.cfg
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
backend dfmConsoleBackend
      mode http
      acl logout_path_set var(txn.path) path
```

```
        /admin/logout http-request set-header X-
        Forwarded-Port %[dst_port]
        http-request add-header X-Forwarded-Proto https if { ssl_fc }

        option httpchk GET
        /admin/health/livehttp-check
        expect status 200
        default-server inter 5s fall 3 rise 2

        # if DFM Server is behind customer's Load-Balancer and also customer's Load-Balancer provides ssl
        termination.#http-response replace-value Location (.*) https://%[var(txn.host)]/admin/ if logout_path_set
        # otherwise
        http-response replace-value Location (.*) %[var(txn.scheme)]://%[var(txn.host)]/admin/ if logout_path_set

        server dfm-console dfm-console:10050 check resolvers docker init-addr libc,none
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

II. HTTP Handling

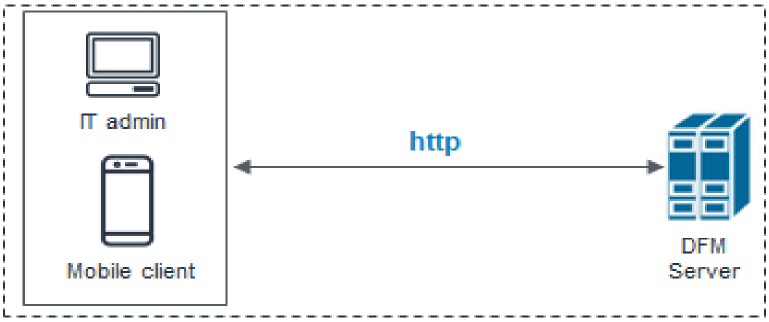

Fig 4-12 **On DFM Server**

Be careful to comment out the "**bind *:443 …**" line in the haproxy.cfg file in a HTTP-only (non HTTPS) configuration.

1. Web server configuration

vi /dfm/haproxy/config/haproxy.cfg

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
frontend fe_web
    bind *:80
    #bind *:443 ssl crt /usr/local/etc/haproxy/example-sec-fota.net.pem
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

2. DFM core/console configuration

vi /dfm/haproxy/config/haproxy.cfg

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
backend dfmConsoleBackend
        mode http
        acl logout_path_set var(txn.path) path
        /admin/logout http-request set-header X-
        Forwarded-Port %[dst_port]
        http-request add-header X-Forwarded-Proto https if { ssl_fc }

        option httpchk GET
        /admin/health/livehttp-check
        expect status 200
        default-server inter 5s fall 3 rise 2

        # if DFM Server is behind customer's Load-Balancer and also customer's Load-Balancer provides ssl
        termination.#http-response replace-value Location (.*) https://%[var(txn.host)]/admin/ if logout_path_set
        # otherwise
        http-response replace-value Location (.*) %[var(txn.scheme)]://%[var(txn.host)]/admin/ if logout_path_set

        server dfm-console dfm-console:10050 check resolvers docker init-addr libc,none
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

4.7. How to check Server Operation Status

Finally, the installer has completed the installation of the on-premises service-based Docker, and the service is now ready for use. However, we first need to validate whether the above five containers are running in a healthy state.

To check the status of the containers, use the command shown below. If every status returns healthy, the service is ready for operation.

docker ps -a

Example)
1) MySQL Server (1 Container)
docker ps -a

CONTAINER ID	STATUS	NAMES
d882c61ba91c	Up 15 hours (healthy)	dfm-mysql

2) Firmware Storage Server (1 Container)
docker ps -a

CONTAINER ID	STATUS	NAMES
af3949b8db98	Up 6 minutes (healthy)	dfm-minio

3) DFM Core/Console Server (3 Containers)
docker ps -a

CONTAINER ID	STATUS	NAMES
07ffa549f3cf	Up 2 minutes (healthy)	dfm-console
a470bb8bb995	Up 5 minutes (healthy)	dfm-core
e10be66fe8bc	Up 3 minutes (healthy)	dfm-proxy

4) HA Proxy Server (1 Container)
docker ps -a

CONTAINER ID	STATUS	NAMES
e10be66fe8bc	Up 3 minutes (healthy)	dfm-proxy

Here, the health status means:

Healthy(0): Normal
Unhealthy(1): Abnormal
Starting (2): Starting

When the installer checks the health status after the installation is completed, if the status is not “Normal”, the installer must redo the installation. If the installation is unsuccessful after several tries, please contact the Samsung engineering team.

PART III: Initial Operation

PART III describes how to operate the Knox E-FOTA On-Premises service upon completion of the service installation on the customer's infrastructure.

5. Service Operation

This chapter explains how to check the operation status of each DFM Server, and how to use the service properly.

5.1. How to access the admin console page after installation

If you completed every installation step, go to the admin page to check whether the DFM Service was successfully installed and is working as expected.

【URL to the admin site】

{access_scheme}://{access_address}:{access_port}/admin/

⇒ Refer to "***4.5.7. (STEP07) Set-up Configuration***".

In this guide, we are using the URL and other information as follows:

- host_ip : 192.168.1.52
- listen_port : 80
- listen_scheme : http
- access_address : 181.107.61.233
- access_scheme : http
- access_port : 6380

【Account & Initial Password (PWD)】

⇒ Account will be: **admin**

⇒ Initial PWD will be: **admin12#**

*) *This PWD is created by Samsung, so **change the password** after you sign in.*

【Example】 http://192.168.1.52:6380/admin/ (using a new **Chrome** browser)

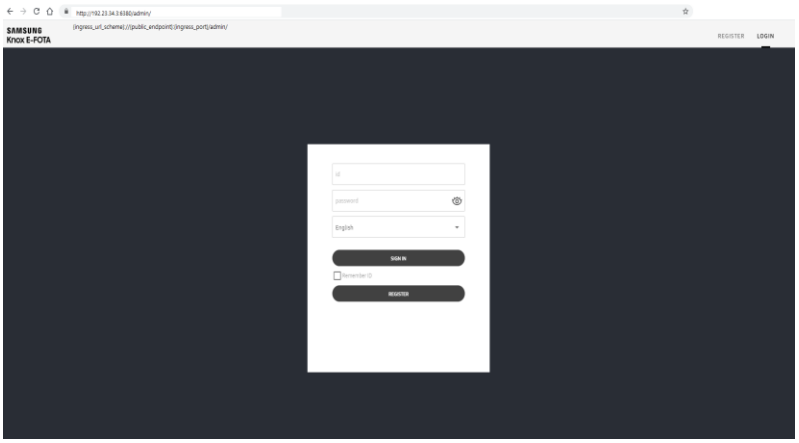

Fig 5-1 The Admin Console for Knox E-FOTA On-Premises

5.2. The Contents Upload

In order to use this service, IT admins must upload the contents (such as license and firmware) properly (please refer to the "**Knox E-FOTA On-Premises User Manual**" provided).

5.3. Troubleshooting and Logging during using the Service

While using this service, any issues should first be addressed on the site to avoid service disruptions from the issues. In order to support issue analysis, Samsung provides the "**TS & Logging Guide for Knox E-FOTA On-Premises**" guide for reference.

5.4. Updating the SSL Certificate when the old certificate is expired

SSL certificates have an expiration date. When the expiration date for the certificate approaches, the customer must reissue the certificate from the certificate signing authority before the current certificate expires.
This work must be done on the web server.

There are two possibilities for TLS/SSL Termination.

1) On Customer's Load Balancer (proxy)
   We don't need to update the certificate file.
   Refer to (**[Use Case 2]:Type B**) in "***4.6.4. Web server***"

2) On DFM Server
   We need to update the certificate file on the DFM Server.
   Refer to (**[Use Case 2]:Type C**) in "***4.6.4. Web server***"

We assume that the newly certificate file is "**new-example-fota.net.pem**", and we also assume that you are using the "**nightwatch**" service account.

【STEP01】 Stop Proxy

The command to stop the proxy Server container is as follows:

```
dfm terminate dfm-proxy
```

【STEP02】 Copy the newly certificate

```
cp new-example-fota.net.pem /dfm/haproxy/config
sudo chown nightwatch:nightwatch /dfm/haproxy/config/new-example-fota.net.pem
sudo chmod 600 /dfm/haproxy/config/new-example-fota.net.pem
vi /dfm/haproxy/config/haproxy.cfg

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
frontend fe_web
    bind *:80
    bind *:443 ssl crt /usr/local/etc/haproxy/new-example-sec-fota.net.pem
```

【STEP03】 Restart proxy

The command to restart the proxy Server container is as follows:

```
dfm cluster start dfm-proxy
```

To make sure that the HAProxy container is in a healthy state, run the following command. It may take some time until the state shows as healthy.

```
docker ps -a

docker ps -a
CONTAINER ID   ~   STATUS                          ~   NAMES
~~
e10be66fe8bc       Up 18 seconds (health: starting)      dfm-proxy
~~
$

docker ps -a
CONTAINER ID   ~   STATUS                  ~   NAMES
~~
e10be66fe8bc       Up 3 minutes (healthy)        dfm-proxy
~~
$
```

5.5. DB Backup & Restore.

This section describes how to back up and restore the DB to ensure service continuity.
This work must be done on the DB(MySQL) server.

5.5.1. Backup a MySQL Server Instance

I. *To BACKUP a MySQL Server instance running in a Docker container using MySQL Enterprise Backup with Docker:*

1. On the same host where the MySQL Server container is running, start another container with an image of MySQL Enterprise Edition to perform a backup with the MySQL Enterprise Backup command " `backup-to-image`". Provide access to the server's data directory using the bind mount we created in the last step. Also, mount a host directory (*/path-on-host-machine/backups/* in this example) onto the storage folder for backups in the container (*/data/backups* in the example) to persist the backups we are creating.

Here is a sample command for this step. Here, we assume that 'root' account's password is **1q2w3e4r** and that the MySql docker image currently installed is *mysql/enterprise-server:8.0*.

【Basic Command】

```
docker run \
  --mount type=bind,src=/path-on-host-machine/datadir/,dst=/var/lib/mysql \
  --mount type=bind,src=/path-on-host-machine/backups/,dst=/data/backups \
  --rm mysql/enterprise-server:8.0 \
  mysqlbackup -u{user} -p{password} --backup-dir=/tmp/backup-tmp --with-timestamp \
  --backup-image=/data/backups/{db-file-name}.mbi backup-to-image
```

【Example】

```
docker run \
  --mount type=bind,src=/dfm/mysql/data,dst=/var/lib/mysql \
  --mount type=bind,src=/dfm/mysql/backups,dst=/data/backups \
  --rm mysql/enterprise-server:8.0 \
  mysqlbackup -uroot -p1q2w3e4r --backup-dir=/tmp/backup-tmp --with-timestamp \
  --backup-image=/data/backups/db-2020-05-20.mbi backup-to-image

  [Entrypoint] MySQL Docker Image 8.0.20-1.1.16
  MySQL Enterprise Backup Ver 8.0.20-commercial for Linux on x86_64 (MySQL Enterprise - Commercial)
  Copyright (c) 2003, 2020, Oracle and/or its affiliates. All rights reserved.

  Oracle is a registered trademark of Oracle Corporation and/or its
  affiliates. Other names may be trademarks of their respective
  owners.

  Starting with following command line ...
    mysqlbackup -uroot -pxxxxxxxx --backup-dir=/tmp/backups --with-timestamp
              --backup-image=/data/backups/db-2020-05-20.mbi  backup-to-image

  IMPORTANT: Please check that mysqlbackup run completes successfully.
                  At the end of a successful 'backup-to-image' run mysqlbackup
                  prints "mysqlbackup completed OK!".

  200520 01:40:40 MAIN       INFO: Starting to log actions.
  200520 01:40:40 MAIN       INFO: No SSL options specified.
  .....
```

```
200520 01:40:42 MAIN    INFO: Backup image created successfully.
200520 01:40:42 MAIN    INFO: Image Path = /data/backups/db-2020-05-20.mbi
200520 01:40:42 MAIN    INFO: MySQL binlog position: filename binlog.000005, position 530056

----------------------------------------
     Parameters Summary
----------------------------------------
     Start LSN                  : 32781824
     End LSN                    : 32848770
----------------------------------------

mysqlbackup completed OK!
```

It is important to check the end of the output by **mysqlbackup** to make sure the backup has been completed successfully.

2. The container exits once the backup job is finished and, with the *--rm* option used to start it, it is removed after it exits. An image backup is created and can be found in the host directory mounted in the last step for storing backups:

```
ls /dfm/mysql/backups/
db-2020-05-20.mbi
```

5.5.2. Restore a MySQL Server Instance

II. *To RESTORE a MySQL Server instance in a Docker container using MySQL Enterprise Backup with Docker:*

1. Stop the MySQL Server container, which also stops the MySQL Server running inside:

mounted in the last step for storing backups:

```
docker stop dfm-mysql
```

2. On the host, delete all contents in the bind mount for the MySQL Server data directory:

```
sudo rm -rf /dfm/mysql/data/*
```

3. Start a container with an image of MySQL Enterprise Edition to perform the restore with the MySQL Enterprise Backup command copy-back-and-apply-log. Bind-mount the server's data directory and the storage folder for the backups, like what we did when we backed up the server:

【Basic Command】

```
docker run \
  --mount type=bind,src=/path-on-host-machine/datadir/,dst=/var/lib/mysql \
  --mount type=bind,src=/path-on-host-machine/backups/,dst=/data/backups \
  --rm mysql/enterprise-server:8.0 \
  mysqlbackup --backup-dir=/tmp/backup-tmp --with-timestamp \
  --datadir=/var/lib/mysql/ --backup-imag=/data/backups/{db-file-name-to-restore}.mbi copy-back-and-apply-log
```

【Example】

```
docker run \
    --mount type=bind,src=/dfm/mysql/data,dst=/var/lib/mysql \
    --mount type=bind,src=/dfm/mysql/backups,dst=/data/backups \
    --rm mysql/enterprise-server:8.0 \
    mysqlbackup --backup-dir=/tmp/backup-tmp --with-timestamp \
    --datadir=/var/lib/mysql --backup-image=/data/backups/db-2020-05-20.mbi copy-back-and-apply-log


MySQL Enterprise Backup    Ver 8.0.20-commercial for Linux on x86_64 (MySQL Enterprise - Commercial)
Copyright (c) 2003, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Starting with following command line ...
  mysqlbackup -uroot -pxxxxxxxx --backup-dir=/tmp/backup-tmp
              --with-timestamp --datadir=/var/lib/mysql
              --backup-image=/data/backups/db-2020-05-20.mbi copy-back-and-apply-log

IMPORTANT: Please check that mysqlbackup run completes successfully.
                   At the end of a successful 'copy-back-and-apply-log' run mysqlbackup
                   prints "mysqlbackup completed OK!".

200520 02:01:08 MAIN        INFO: Starting to log actions.
[Entrypoint] MySQL Docker Image 8.0.20-1.1.16
.......
200520 02:01:10 PCR1        INFO: We were able to parse ibbackup_logfile up to
                lsn 32848770.
200520 02:01:10 PCR1        INFO: Last MySQL binlog file position 0 530056, file name binlog.000005
200520 02:01:10 PCR1        INFO: The first data file is '/var/lib/mysql/ibdata1'
                                  and the new created log files are at '/var/lib/mysql'
200520 02:01:10 MAIN        INFO: Apply-log operation completed successfully.
200520 02:01:10 MAIN        INFO: Full Backup has been restored successfully.

mysqlbackup completed OK! with 3 warnings
```

The container exits once the backup job is finished and, with the *--rm* option used when starting it, it is removed after it exits.

4. Restart the server container, which also restarts the restored server:

```
docker restart dfm-mysql
```

5.6 Configurable length of password digits

This work must be done on the DFM Core/Console server.
The installer can change this default value of a minimum and maximum length of password digits.

(default password_min_length=8, default password_max_length=12)

【STEP01】 Stop DFM Admin Console

The command to stop the DFM Admin Console Server container is as follows

```
dfm terminate dfm-console
```

【STEP02】 Set-up the length of the password digits

The minimum length of password is allowed from 8 to 20.
The max length of password is allowed from 12 to 30.

```
dfm config set password_min_length=8
dfm config set password_min_length=20
```

【STEP03】 Check the length of the password digits

```
dfm config get password_min_length
8

dfm config get password_max_length
20
```

【STEP04】 Restart DFM Admin Console

The command to restart the DFM Admin Console Server container is as follows

```
dfm cluster start dfm-console
```

To make sure that the DFM Admin Console container is in a healthy state, run the following command. It maytake some time until the state shows as healthy.

```
    docker ps -a

    docker ps -a
  CONTAINER ID   ~   STATUS                          ~              NAMES

~~ e92be16ye8bc      Up 18 seconds (health: starting)              dfm-console
~~
$

   docker ps -a
  CONTAINER ID   ~   STATUS                          ~   NAMES

~~ e92be16ye8bc      Up 3 minutes (healthy)            dfm-console
~~
$
```

5.7 Configurable device group polling

The installer can change the default value of the device group. (default device_group_enable=false, device_group_max_limit=20000)
This function is used to distribute a large number of devices when serving, and all the devices are distributed across 60 groups.

【STEP01】 Stop DFM Core

The command to stop the DFM Core Server container is as follows:

```
dfm terminate dfm-core
```

【STEP02】 Set up the device group polling

The allowed values of “device group enable” are “true” or “false”.
The device group max limit is 20000.

```
dfm config set device_group_enable =true
dfm config set device_group_max_limit =20000
```

【STEP03】 Check the device group polling

```
dfm config get device_group_enable
true

dfm config get device_group_max_limit
20000
```

【STEP04】 Restart the DFM Core

The command to restart the DFM Core Server container is as follows:

```
dfm cluster start dfm-core
```

To make sure that the DFM Admin Console container is in a healthy state, run the following command. It may take some time until the state shows as healthy.

```
    docker ps -a

    docker ps -a
   CONTAINER ID   ~    STATUS                          ~              NAMES

~~ e92be16ye8bc       Up 18 seconds (health: starting)               dfm-core
~~
$

   docker ps -a
  CONTAINER ID    ~    STATUS                          ~    NAMES

~~ e92be16ye8bc       Up 3 minutes (healthy)              dfm-core
~~
$
```

6. When a Server is Rebooted

This chapter explains the steps to restart the DFM Modules if the server is rebooted, to ensure the service can run properly.

The steps to start the DFM Module server are as follows:

6.1. (STEP01) Login as the dedicated service account

The DFM Module is logged in with a dedicated service account and operates with the privileges of the account (see, "***4.1. (STEP01) Create Service Account and Login***").

6.2. (STEP02) Prepare "mount" for DFM modules

The DFM module is installed and operates in the below directory on the **dedicated disk**.

The customer **may NOT configure** the auto-mount on the dedicated disk. For such cases, it is necessary to manually mount the dedicated disk on **/dfm**.

Prepare "mount" reffering to image of **(STEP02)** for each server written in **4. Installation & Configuration**.

For example, we assume that two disks ("sda" and "sdb") exist.

【CASE01】 Disk is Ready

If the disk is ready, we don't need to mount it.

Now, let's check the disk information:

```
sudo lsblk –p
```

```
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
/dev/sda     202:0    0    1T  0 disk
└─/dev/sda1 202:1    0    1T  0 part /
/dev/sdb     202:80   0    1T  0 disk
```

sudo lsblk –f

```
NAME    FSTYPE   LABEL          UUID                                   MOUNTPOINT
sda
└─sda1  ext4     xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb     ext4                     d3269ceb-4418-45d0-ba68-d6b906e0595d /dfm
```

⇨ “sdb” is already formatted and mounted on **/dfm**

sudo file –s /dev/sdb

```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

【CASE02】 Disk is NOT Ready: it is already formatted but not yet mounted on /dfm

If the disk is formatted but not yet mounted, it needs to be mounted on **/dfm**.
Now, let’s check the disk information.

sudo lsblk –p

```
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
/dev/sda     202:0    0    1T  0 disk
└─/dev/sda1 202:1    0    1T  0 part /
/dev/sdb     202:80   0    1T  0 disk
```

sudo lsblk –f

```
NAME    FSTYPE   LABEL          UUID                                   MOUNTPOINT
sda
└─sda1  ext4     xxxxxxxx-rootfs 6156ec80-9446-4eb1-95e0-9ae6b7a46187 /
sdb     ext4                     d3269ceb-4418-45d0-ba68-d6b906e0595d
```

⇨ “sdb” ” is formatted but not yet mounted

1) Mount /dev/sdb on /dfm

```
// create directory to mount
sudo mkdir /dfm

// mount
sudo mount /dev/sdb /dfm
```

2) Verify

```
df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb        9.8G   37M  9.3G   1% /dfm
```

6.3. (STEP03) Start up Docker

After the system is rebooted, check whether the Docker engine is running.

```
sudo systemctl status docker
```

```
docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
 Drop-In: /etc/systemd/system/docker.service.d
          └─http-proxy.conf, https-proxy.conf
 Active: active (running) since Fri 2020-02-07 13:12:39 KST; 3 weeks 2 days ago
   Docs: https://docs.docker.com
```

If the **Active** value is not "**active (running)**", Docker is not yet running.

If the Docker engine is not running, run it using the following command.

```
$ sudo systemctl start docker
```

6.4. (STEP04) Start-up Database Server (MySQL)

After the system is rebooted, restart MySQL using the following command:

```
dfm cluster restart dfm-mysql
```

【Validation】

Run the following command to ensure the MySOL container is in a healthy state. It may take some time until its state is healthy.

```
docker ps -a

Example)
$ docker ps -a
CONTAINER ID   ~   STATUS                          ~   NAMES
d882c61ba91c       Up 4 seconds (health: starting)      dfm-mysql

docker ps -a
CONTAINER ID   ~   STATUS                          ~   NAMES
d882c61ba91c       Up 2 minutes (healthy)               dfm-mysql
```

6.5. (STEP05) Start-up Firmware Storage Server

After the system is rebooted, restart Minio. The command to run Minio server container is as follows:

```
dfm cluster restart dfm-minio
```

【Validation】

Run the following command to make sure the Minio container is in a healthy state. It may take some time until its state is healthy.

```
docker ps -a

Example)
docker ps -a
CONTAINER ID   ~   STATUS                          ~   NAMES
~~
af3949b8db98      Up 4 seconds (health: starting)      dfm-minio
~~

docker ps -a
CONTAINER ID   ~   STATUS                          ~   NAMES
~~
af3949b8db98      Up 2 minutes (healthy)               dfm-minio
~~
```

6.6. (STEP06) Start-up DFM Core Server

After the system is rebooted, restart DFM Core. The command to run the core server container is as follows:

```
dfm cluster restart dfm-core
```

【Validation】

Run the following command to make sure the core container is in a healthy state. It takes some time until its state is healthy.

```
docker ps -a

Example)
docker ps -a
CONTAINER ID   ~   STATUS                          ~   NAMES
~~
a470bb8bb995      Up 4 seconds (health: starting)      dfm-core
~~
$

docker ps -a
CONTAINER ID   ~   STATUS                          ~   NAMES
~~
a470bb8bb995      Up 2 minutes (healthy)               dfm-core
```

```
~~
```

6.7. (STEP07) Start-up DFM Admin Console Server

After the system is rebooted, restart DFM Admin. The command to run the admin server container is as follows:

```
dfm cluster restart dfm-console
```

【Validation】

Run the following command to ensure the admin container is in a healthy state. It takes some time until its state is healthy.

```
docker ps -a

Example)
docker ps -a
CONTAINER ID      ~      STATUS                          ~                NAMES
~~
07ffa549f3cf            Up 7 seconds (health: starting)                  dfm-console
~~

docker ps -a
CONTAINER ID      ~      STATUS                          ~      NAMES
~~
07ffa549f3cf            Up 2 minutes (healthy)                 dfm-console
~~
```

6.8. (STEP08) Start-up HAProxy Server

After the system is rebooted, restart HAProxy. The command to run the HAProxy server container is as follows:

```
dfm cluster restart dfm-proxy
```

【Validation】

Run the following command to make sure the HAProxy container is in a healthy state. It takes some time until its state is healthy.

```
docker ps -a

Example)
docker ps -a
CONTAINER ID      ~      STATUS                    ~              NAMES
~~
e10be66fe8bc            Up 18 seconds (health: starting)         dfm-proxy
~~
```

```
docker ps -a
CONTAINER ID   ~   STATUS                    ~   NAMES
~~
e10be66fe8bc       Up 3 minutes (healthy)        dfm-proxy
~~
```

PART IV: Update the DFM Modules

PART IV: Update the DFM Modules describes how to update the Knox E-FOTA On-Premises service if there are any updates within the service resources.

7. Update the DFM Module

This chapter explains how to update the DFM Modules in operation, such as a fetch version. In order to properly update each module, the updater must first stop the module based on the related command (see, ***Appendix B***).

During the update, the Knox E-FOTA On-Premises service may not be available.

The DFM Module is logged in with a dedicated service account and operates with the privileges of the account. Ensure you log in with the account you previously used for installation.

7.1. Docker Image Update

If there is an updated DFM Module, it is released as a Docker Image Package and packed as a tar file. In the release, the Docker Image contains repository and tag information as well.

7.1.1. DFM Database Update (MySQL)

This work must be done on DFM Core/Console server and DB(MySQL) server.
For example, assume that the released **MySQL** image information is as follows:

- docker image: dfm-mysql-xx.xx.xx.tar
- repository: dfm-mysql
- tag: xx.xx.xx

It should be updated as per the following steps.

【STEP01】 Stop the running DFM Core Server, Admin Console Server, and Mysql Server.

1) Terminate the service on DFM Core/Console server.

```
dfm terminate dfm-core
dfm terminate dfm-console
```

2) Terminate the service on DB(MySQL) server.

```
dfm terminate dfm-mysql
```

【STEP02】 Load the released Docker Image.

```
docker load < dfm-mysql-xx.xx.xx.tar
```

【STEP03】 Change the repository and tag's configuration

```
dfm config set mysql_img_rep=dfm-mysql
dfm config set mysql_img_tag=xx.xx.xx
```

【STEP04】 Confirm the changed repository and tag's configuration

```
dfm config get mysql_img_rep
dfm config get mysql_img_tag
```

【STEP05】 Start-up Server

1) Start the service on DFM Core/Console server.

```
dfm cluster start dfm-core
dfm cluster start dfm-console
```

【Validation】

Run the following command to ensure the mysql container is in a healthy state. It takes

some time until its state is healthy.

```
docker ps -a
```

2) Start the service on DB(MySQL) server.

```
dfm cluster start dfm-mysql
```

【Validation】
Run the following command to ensure the mysql container is in a healthy state. It takes some time until its state is healthy.

```
docker ps -a
```

7.1.2. DFM Firmware Storage Update (MinIO)

This work must be done on the firmware Storage(minio) server.
For example, assume that the released **MinIO** image information is as follows:

- docker image : dfm-minio-xx.xx.xx.tar
- repository : dfm-minio
- tag : xx.xx.xx

【STEP01】 Stop the MinIO server.

```
dfm terminate dfm-minio
```

【STEP02】 Load the released Docker Image.

```
docker load < dfm-minio-xx.xx.xx.tar
```

【STEP03】 Change the repository and tag's configuration

```
dfm config set minio_img_rep=dfm-minio
dfm config set minio_img_tag=xx.xx.xx
```

【STEP04】 Confirm the changed repository and tag's configuration

```
dfm config get minio_img_rep
dfm config get minio_img_tag
```

【STEP05】 Start-up Server

- MinIO Server

```
dfm cluster start dfm-minio
```

【Validation】
Run the following command to ensure the mysql container is in a healthy state. It takes some time until its state is healthy.

```
docker ps -a
```

7.1.3. DFM Core Update

This work must be done on DFM Core/Console server.
For example, assume that the released **Core** image information is as follows:

- docker image : dfm-core-xx.xx.xx.tar
- repository : dfm-core
- tag : xx.xx.xx

【STEP01】 Stop the running core server.

```
dfm terminate dfm-core
```

【STEP02】 Load the released docker image.

```
docker load < dfm-core-xx.xx.xx.tar
```

【STEP03】 Change the repository and tag's configuration

```
dfm config set core_img_rep=dfm-core
dfm config set core_img_tag=xx.xx.xx
```

【STEP04】 Confirm the changed repository and tag's configuration

```
dfm config get core_img_rep
dfm config get core_img_tag
```

【STEP05】 Start-up Server

- DFM Core Server

  ```
  dfm cluster start dfm-core
  ```

  【Validation】
  Run the following command to ensure the mysql container is in a healthy state. It takes some time until its state is healthy.

  ```
  docker ps -a
  ```

7.1.4. DFM Admin Console Update

The following work must be done on the DFM Core/Console server.
For example, assume that the released **Admin** image information is as follows:

- docker image : dfm-console-xx.xx.xx.tar
- repository : dfm-console
- tag : xx.xx.xx

【STEP01】 Stop the running core, admin and mysql servers.

```
dfm terminate dfm-console
```

【STEP02】 Load the released docker image.

```
docker load < dfm-console-xx.xx.xx.tar
```

【STEP03】 Change the repository and tag's configuration

```
dfm config set console_img_rep=dfm-console
dfm config set console_img_tag=xx.xx.xx
```

【STEP04】 Confirm the changed repository and tag's configuration

```
dfm config get console_img_rep
dfm config get console_img_tag
```

【STEP05】 Start up the server

- Admin Console Server

  ```
  dfm cluster start dfm-console
  ```

  【Validation】
  Run the following command to ensure the mysql container is in a healthy state. It takes some time until its state is healthy.

```
docker ps -a
```

7.1.5. HAProxy update

The following work must be done on the web server and the DFM Core/Console server.
For example, assume that the released **HAProxy** image information is as follows:

- docker image : dfm-haproxy-xx.xx.xx.tar
- repository : dfm-haproxy
- tag : xx.xx.xx

【**STEP01**】 Stop the running haproxy server.

```
dfm terminate dfm-proxy
```

【**STEP02**】 Load the released docker image.

```
docker load < dfm-haproxy-xx.xx.xx.tar
```

【**STEP03**】 Change the repository and tag's configuration

```
dfm config set haproxy_img_rep=dfm-haproxy
dfm config set haproxy_img_tag=xx.xx.xx
```

【**STEP04**】 Confirm the changed repository and tag configuration

```
dfm config get haproxy_img_rep
dfm config get haproxy_img_tag
```

【**STEP05**】 Start up the server

- HAProxy Server

```
dfm cluster start dfm-proxy
```

【Validation】
Run the following command to ensure the HAProxy container is in a healthy state. It may take some time until its state is healthy.

```
docker ps -a
```

7.2. The Contents Update

In order to use this service, IT admins must upload the contents (such as the license and firmware) properly. Please refer to the "**Knox E-FOTA On-Premises User Manual**" provided.

PART V: Purge DFM Modules

This section, which covers purging the DFM Modules, describes how to erase all installed services when you want to delete the existing installed modules.

Please note that doing so **erases all existing data**.

After completing these actions, you can reinstall the DFM modules without any interference from the old installation (see 4.3. (STEP03) Create Service Directories).

8. Purge the DFM Modules

This chapter explains how to purge the installed DFM Modules.

The DFM Module is logged in with a dedicated service account and operates with the privileges of the account. Log in with the account you used during the installation.

8.1. Purge the installed Debian package

【STEP01】 Check if the installed dfm debian package exists.

dpkg -l | grep sec-dfm

example-1) when installed pkg exists

```
$ dpkg -l | grep sec-dfm
ii  sec-dfm        1.0.0.5     all Samsung Enterprise fota dfm package
$
```

example-2) when installed pkg does Not exist

```
$ dpkg -l | grep sec-dfm
$
```

【STEP02】 If the installed dfm debian package exists, remove it.

sudo dpkg -P sec-dfm

example)

```
$ sudo dpkg -P sec-dfm
(Reading database ... 300748 files and directories currently installed.)
Removing sec-dfm (1.0.0.5) ...
$
```

8.2. Terminate Services

Terminate services each server.
If there are active services, terminate them.

1) WEB Server

【STEP01】 Check if there is any running or exited services. If they exist, we need to terminate them.

```
docker ps -a

example)
docker ps -a
CONTAINER ID   IMAGE                                  ~~ STATUS                    ~~ NAMES
879ab3603220   dfm-console:2.0.0-20200625045619       ~~ Up 3 hours (healthy)   ~~   dfm-console
2966ea3ab692   dfm-core:1.0.0.5                       ~~ Up 3 hours (healthy)   ~~   dfm-core
b6ed98da1101   haproxytech/haproxy-debian:2.1.4       ~~ Up 3 hours (healthy)   ~~    dfm-proxy
b10b70f135d0   minio/minio:RELEASE.2020-06-01T17-28-03Z  ~~ Up 3 hours (healthy)   ~~   dfm-minio
63c384eb0d5c   mysql/enterprise-server:8.0             ~~ Up 3 hours (healthy)   ~~   dfm-mysql
```

1. DFM HAProxy Server

   Stop the server with the following command:

   ```
   dfm terminate dfm-proxy
   ```

2. Check if all services are removed.

   Check with the following command:

   ```
   docker ps -a
   ```

2) DFM Core/Console Server

【STEP01】 Check if there is any running or exited services. If they exist, we need to terminate them.

```
docker ps -a

example)
docker ps -a
CONTAINER ID   IMAGE                                  ~~ STATUS                    ~~ NAMES
879ab3603220   dfm-console:2.0.0-20200625045619       ~~ Up 3 hours (healthy)   ~~   dfm-console
2966ea3ab692   dfm-core:1.0.0.5                       ~~ Up 3 hours (healthy)   ~~   dfm-core
b6ed98da1101   haproxytech/haproxy-debian:2.1.4       ~~ Up 3 hours (healthy)   ~~    dfm-proxy
b10b70f135d0   minio/minio:RELEASE.2020-06-01T17-28-03Z  ~~ Up 3 hours (healthy)   ~~   dfm-minio
63c384eb0d5c   mysql/enterprise-server:8.0             ~~ Up 3 hours (healthy)   ~~   dfm-mysql
```

1. DFM HAProxy Server

   Stop the server with the following command:

   ```
   dfm terminate dfm-proxy
   ```

2. DFM Core Server

   Stop the server with the following command:

   ```
   dfm terminate dfm-core
   ```

3. DFM Admin Console Server

   Stop the server with the following command:

   ```
   dfm terminate dfm-console
   ```

4. Check if all services are removed.

Check with the following command:

```
docker ps -a
```

3) Firmware Storage(minio) Server

【STEP01】 Check if there is any running or exited services. If they exist, we need to terminate them.

```
docker ps -a

example)
docker ps -a
CONTAINER ID   IMAGE                                    ~~ STATUS                  ~~ NAMES
879ab3603220   dfm-console:2.0.0-20200625045619         ~~ Up 3 hours (healthy)    ~~  dfm-console
2966ea3ab692   dfm-core:1.0.0.5                         ~~ Up 3 hours (healthy)    ~~  dfm-core
b6ed98da1101   haproxytech/haproxy-debian:2.1.4         ~~ Up 3 hours (healthy)    ~~  dfm-proxy
b10b70f135d0   minio/minio:RELEASE.2020-06-01T17-28-03Z ~~ Up 3 hours (healthy)    ~~  dfm-minio
63c384eb0d5c   mysql/enterprise-server:8.0              ~~ Up 3 hours (healthy)    ~~  dfm-mysql
```

1. DFM HAProxy Server

Stop the server with the following command:

```
dfm terminate dfm-minio
```

2. Check if all services are removed.

```
docker ps -a
```

Check with the following command:

4) DB(MySQL) server

【STEP01】 Check if there is any running or exited services. If they exist, we need to terminate them.

```
docker ps -a

example)
docker ps -a
CONTAINER ID   IMAGE                                    ~~ STATUS                  ~~ NAMES
879ab3603220   dfm-console:2.0.0-20200625045619         ~~ Up 3 hours (healthy)    ~~  dfm-console
2966ea3ab692   dfm-core:1.0.0.5                         ~~ Up 3 hours (healthy)    ~~  dfm-core
b6ed98da1101   haproxytech/haproxy-debian:2.1.4         ~~ Up 3 hours (healthy)    ~~  dfm-proxy
b10b70f135d0   minio/minio:RELEASE.2020-06-01T17-28-03Z ~~ Up 3 hours (healthy)    ~~  dfm-minio
63c384eb0d5c   mysql/enterprise-server:8.0              ~~ Up 3 hours (healthy)    ~~  dfm-mysql
```

1. DFM Database (MySQL)

   Stop the server with the following command:

```
dfm terminate dfm-mysql
```

2. Check if all services are removed.

   Check with the following command:

```
ps -a
```

8.3. Remove Service directory

Remove old data using the following:

Remove all directory in /dfm

```
cd /dfm
sudo rm -rf *
```

PART VI: APPENDICES

PART IV: APPENDICES presents more in-depth explanations for each item.

APPENDICES

Appendix A. Terms and Abbreviations

This chapter outlines the terms and abbreviations used in this guide.

App: Application
CAT: Category Codes
CSO/TEO: Customer Service Operation/Technical Engineer for On-Premises
CM: Commercial Type Product
DE: Docker Enterprise
DFM: Device Firmware Management
DNS: Domain Name Server
E2E: End to End
E-FOTA: Enterprise – Firmware over the Air
FYI: For Your Information
HA: High Availability
H/W: Hardware
ID: Identification
KE: Knox E-FOTA (Brand)
LB: Load Balancer
NAT: Network Address Translation
OS: Operating System
PoC: Proof of Concept
PWD: Password
SSL: Secure Sockets Layer
TLS: Transport Layer Security, successor to SSL
UI: User Interface

Appendix B. How to terminate each DFM Module

These commands should not be used in normal operation, as stopping a module can seriously impact how the service runs. Use this command for updates, such as when there is a fetch version delivery.

1. DFM Database (MySQL)

   Stop the server with the following command:

   ```
   dfm terminate dfm-mysql
   ```

2. DFM Firmware Storage (MinIO)

   Stop the server with the following command:

   ```
   dfm terminate dfm-minio
   ```

3. DFM Core Server

   Stop the server with the following command:

   ```
   dfm terminate dfm-core
   ```

4. DFM Admin Console Server

   Stop the server with the following command:

   ```
   dfm terminate dfm-console
   ```

5. DFM HAProxy Server

   Stop the server with the following command:

   ```
   dfm terminate dfm-proxy
   ```

Appendix C. Summary for Software (S/W) Recommendation

Read more about detailed recommendations in "*2.3. Recommendation Per each Product usage*".

Product	Category	S/W	Version	Supported Options	Additional Info
CM	**Server OS**	Ubuntu	18.04.3 LTS	Enterprise (**Paid**)	https://assets.ubuntu.com/v1/1a8fb1b3-UA-I_datasheet_2019-Oct.pdf?_ga=2.267414477.2124202676.1591159591-176408230.1591159591
	Container	Docker Engine	Community Edition	Community (Free)	https://info.mirantis.com/l/530892/2018-04-12/37s6c/530892/93926/Mirantis_Support_Subscription_Brochure.pdf
	Database	MySQL	Enterprise Edition	Enterprise (**Paid**)	https://www.mysql.com/products/
PoC	**Server OS**	Ubuntu	18.04.3 LTS	Community (free)	
	Container	Docker Engine	Community Edition	Community (Free)	
	Database	MySQL	Community Edition	Community (Free)	If a customer wants to continue using the Commercial (**CM**) product after PoC ends, recommend **Enterprise Edition** for both **Server OS** and **Database** at the start of the PoC

Appendix D. A Recommended Schedule for On-Site Installation by CSO/TEO

This recommended schedule can be used by the CSO/TEO while they are doing the on-site installation. The detailed schedule can be freely modified.

We recommend "The 4-Day Installation", as the customer should understand how they are using the Knox E-FOTA On-Premises service during this program. A training session should be included to support this purpose as well.

Day	Actions	Program
Day1	Check the customer's infrastructures (such as H/W and S/W) to install the service on, based on the prerequisites (see "*2.3 Recommendation Per each Product usage*")	1. Introduce each other 2. Introduce "The 4-Days Installation" program 3. Introduce the Knox E-FOTA On-Premises Service (using "**Knox E-FOTA On-Premises Service Intro 2020.pdf**" 4. Check the customer's infrastructures 1) H/W recommendation, such as Server CPU cores, RAM, Disk, Network Card 2) S/W recommendation, such as Operating System, Docker Engine, MySQL Edition, and whether those have been installed by the customer 3) Get public certificate files for https 4) Get port number (6443) for https 5. Wrap-up
Day2	Perform the installation based on this guide (see "*4. Installation & Configuration*")	1. Introduce the program to Installation 2. Start Installation 3. Configure the DFM service infrastructure 4. Check the service operation via the Web Console 5. Wrap-up
Day3	Perform an acceptance test through E2E with devices	1. Introduce how to do an E2E test with devices 2. Introduce how to use the service Web Console (using "**Knox E-FOTA On-Premises User Guide.pdf**, and **Knox E-FOTA On-Premises User Guide for Device.pdf**") 3. Upload the License into the Server 4. Upload the Firmware deltas (Contents for FOTA) 5. Upload the device information using during the test 6. Create the Campaign 7. Perform E2E test with devices 8. Wrap-up
Day4	Introduce Operation and Maintenance procedures (Get document for "**The Confirmation of Installation Process End**" from the Customer)	1. Introduce the steps and how to perform them if there is an issue ☜ Using "**TS & Logging Guide for Knox E-FOTA On-Premises.pdf**" 2. Introduce how to raise issues ☜ Using "**Issue raising process**" 3. Introduce service operation steps ☜ Using "**Service Operation Guide**" 4. Sign the "Notice for Completion Installation"

		☜ Refer to "**Appendix E**" (*Installation and Initial Operation Guide for Knox E-FOTA On-Premises.pdf*) 5. Wrap-up

Appendix E. An Example of "Notice for Completion Installation"

Notice for Completion Installation

Dear < Customer Name >,

This form is to sign-off completion of your project with us. Kindly complete as best as possible and send back to us.

PRODUCT: Knox E-FOTA One On-premise	MANAGER NAME:
START DATE:	COMPLETION DATE:
June 1 2020 ~ June 4 2020	

DELIVERABLES:

1. **Device Client**
   It means Client application running on Samsung mobile devices. It is responsible for interacting with the E-FOTA (Enterprise-Firmware Over The Air) Server, including binary package download, and installer activation for the binary package.

2. **Device Firmware Management (DFM)**
   It is a main module for E-FOTA, including managed devices to FOTA, creation and management of FOTA Campaigns, and Firmware binaries for devices.
   It is consist of followings:
   1) DFM Core – It consists of Core Backend and Front End for Administrators
   2) DB (MySQL) – It is a data base for system operation
   3) Storage – It is a storage for Firmware binaries

3. **Installed in Customer's Environment**
   It depends on the contraction.
   1) Pre-Prod Environment (1 Set)
   2) Prod Environment (1 Set)

CUSTOMER'S COMMENTS:

REMARK:

By signing this document, I acknowledge that I have delivered all the stated deliverables.	By signing this document, I acknowledge that I have received all the stated deliverables.
Samsung (subsidiary office name)	**< Customer Name >**
Name:_____	Name:_____
Signature:_____	Signature:_____
Date: _____	Date: _____

We recommend that you complete and send this form within 5 working days. However, if after this period we do not receive the completed form, we shall assume that the project has been signed off by you and no further action will be required of you.

< EOF (End Of File) >