SAMSUNG ELECTRONICS

Knox E-FOTA On-Premises

Installation and Initial Operation Guide

Version : 1.5

Last Update : Jan 2026

【Document History】

What	Ver.	When
I. **Updated:** 5.5 DB Backup & Restore	Ver1.5	Jan 2026
II. **Updated:** 4.7.2 DFM Core/Console Server 4.7.3 WEB Server 5.4. Updating the SSL Certificate when the old certificate is expired 5.7 Configurable device polling interval and postpone waiting time	Ver1.4	Dec 2025
III. **Updated to 1.0.1.10**	Ver1.3	May 2025
I. **Added:** Appendix F. Set E-FOTA agent config by managed Configuration II. **Updated:** 4.2.6 (STEP04) Install DFM Module Package 4.2.7 (STEP05) Load Container Image 4.3.6 (STEP04) Install DFM Module Package 4.3.7 (STEP05) Load Container Image 4.4.6 (STEP04) Install DFM Module Package 4.4.7 (STEP05) Load Container Image 4.6.18 (STEP04) Install DFM Module Package 4.6.19 (STEP05) Load Container Image 4.7.15 2) On DFM Serve <- added how to make pem file for device	Ver1.2	Dec 2024
I. **Added:** 4.4.12 (STEP10) Copy Background App files 4.4.13 (STEP11) Start-up Background App II. **Updated:** 4.2.5 (STEP03) Create Service Directories 4.2.6 (STEP04) Install DFM Module Package 4.4.9 (STEP07) Set-up Configuration(add license_app_ip set) 8.3 Terminate Services > add stop background App	Ver1.1	June 2024
Initial Release	Ver1.0	June 2020

Table of Contents

PART I: Getting Started 7

1. Introduction 8
   1.1. Purpose of this document 8

2. Environment Prerequisites 9
   2.1. Hardware Recommended 9
   2.2. Software Recommended 10
   2.2.1. Operating System 10
   2.2.2. Keepalived 10
   2.2.1. Podman 10
   2.2.3. Database (MySQL) 11
   2.2.4. HTTPS 11
   2.3. Recommendation Per each Product usage 11
   2.3.1. Product – “PoC” 11
   2.3.2. Product – “Commercial” 13

3. Deliverables 15
   3.1. DFM Modules 15
   3.2. Security Considerations 15
   3.2.1. HTTPS and Network encryption 16
   3.3. Supported Browser 16

PART II: Installation, and Validation 17

4. Installation & Configuration 18
   4.1. Check Pre-Config 18
   4.1.1 Check Web Server config 19
   4.1.2 Check DB(MySQL) Server config 22
   4.1.3 Check Firmware Storage(minio) Server config 22
   4.1.4 Check DFM Core/Console Server config 23
   4.1.5 Using the firewalld service 23
   4.1.6 Configure Access port 23
   4.2. DB(MySQL) Server 24
   4.2.1 (Prerequisites) Install Package 24
   4.2.2 (STEP01) Create Service Account and Login 25
   4.2.3 (STEP02) Prepare “Disk partition & mount” for DFM modules 26
   4.2.4 Permanently mount the disk 28
   4.2.5 (STEP03) Create Service Directories 29
   4.2.6 (STEP04) Install DFM Module Package 29
   4.2.7 (STEP05) Load Container Image 32
   4.2.8 (STEP06) Copy Configuration files 32
   4.2.9 (STEP07) Set-up Configuration 33
   4.2.10 (STEP9) Start-up and Initialize DB(MySQL) Server 39
   4.3. Firmware Storage (minio) Server 43
   4.3.1 (Prerequisites) Install Package 43
   4.3.2 (STEP01) Create Service Account and Login 43
   4.3.3 (STEP02) Prepare “Disk partition & mount” for DFM modules 45
   4.3.4 Permanently mount the disk 47
   4.3.5 (STEP03) Create Service Directories 47
   4.3.6 (STEP04) Install DFM Module Package 48
   4.3.7 (STEP05) Load Container Image 49

4.3.8 (STEP06) Copy Configuration files 50
4.3.9 (STEP07) Set-up Configuration 50
4.3.10 (STEP09) Start-up Firmware Storage(minio) Server 51
4.4. DFM Core/Console Server 52
4.4.1 (Prerequisites) Install Package 52
4.4.2 (STEP01) Create Service Account and Login 52
4.4.3 (STEP02) Prepare “Disk partition & mount” for DFM modules 53
4.4.4 Permanently mount the disk 55
4.4.5 (STEP03) Create Service Directories 56
4.4.6 (STEP04) Install DFM Module Package 56
4.4.7 (STEP05) Load Container Image 58
4.4.8 (STEP06) Copy Configuration files 59
4.4.9 (STEP07) Set-up Configuration 59
4.4.10 (STEP08) Configure HAProxy 61
4.4.11 (STEP09) Create Container Network 61
4.4.12 (STEP10) Copy Background App files 62
4.4.13 (STEP11) Start-up Background App 62
4.4.14 (STEP12) Start-up DFM Core/Console Server 62
4.5 Keepalived 63
4.5.1 (STEP01) Install package 63
4.5.2 (STEP02) Configure keepalived 63
4.5.3 (STEP03) Start-up keepalived 64
4.6 WEB Server 64
4.6.1 (Prerequisites) Install Package 64
4.6.2 (STEP01) Create Service Account and Login 65
4.6.3 (STEP02) Prepare “Disk partition & mount” for DFM modules 66
4.6.4 Permanently mount the disk 68
4.6.5 (STEP03) Create Service Directories 69
4.6.6 (STEP04) Install DFM Module Package 69
4.6.7 (STEP05) Load Container Image 71
4.6.8 (STEP06) Copy Configuration files 71
4.6.9 (STEP07) Set-up Configuration 72
4.6.10 (STEP08) Configure HAProxy 73
4.6.11 (STEP09) Create Container Network 75
4.6.12 (STEP10) Start up web server 75
4.7 Configure SSL 76
4.7.1 DB(MySQL) Server 76
4.7.2 DFM Core/Console Server 76
4.7.3 WEB Server 77
4.8 How to check Server Operation Status 82
PART III: Initial Operation 84

5. Service Operation 85
5.1. How to access the admin console page after installation 85
5.2. The Contents Upload 86
5.3. Troubleshooting and Logging during using the Service 86
5.4. Updating the SSL Certificate when the old certificate is expired 86
5.5 Configurable length of password digits 87
5.6 Configurable device group polling 95
5.7 Configurable device polling interval and postpone waiting time 96

6. When a Server is Rebooted 98
6.1. (STEP01) Login as the dedicated service account 98
6.2. (STEP02) Prepare “mount” for DFM modules 98

6.3 (STEP03) Start-up Database Server (MySQL) 100
6.4 (STEP04) Start-up Firmware Storage Server 101
6.5 (STEP05) Start-up DFM Core Server 103
6.6 (STEP06) Start-up DFM Admin Console Server 103
6.7 (STEP07) Start-up HAProxy Server 104
PART IV: Update the DFM Modules 105

7 Update the DFM Module 106
7.1 Container Image Update 106
7.1.1. DFM Database Update (MySQL) 106
7.1.2. DFM Firmware Storage Update (MinIO) 107
7.1.3. DFM Core Update 107
7.1.4. DFM Admin Console Update 108
7.1.5. HAProxy update 108
7.2 The Contents Update 109
PART V: Purge DFM Modules 110

8 Purge the DFM Modules 111
8.1 Terminate Services 111
8.2 Remove Service directory 112
PART VI: Install Case Scenario 113

9 How to install 2 servers 114
9.1 Create Service Directories 115
9.1.1. Web zone 115
9.1.2. App zone 115
9.1.3. Data zone 115
9.1.4. DB zone 115
9.2 Configurations 116
9.2.1. DB zone 116
9.2.2. Data zone 123
9.2.3. App zone 124
9.2.4. WEB zone 127
9.2.5. keepalived 128
9.3 Start-up services 128
PART VII: APPENDICES 130

APPENDICES 131
Appendix A. Terms and Abbreviations 131
Appendix B. How to terminate each DFM Module 132
Appendix C. Summary for Software (S/W) Recommendation 133
Appendix D. A Recommended Schedule for On-Site Installation by CSO/TEO 134
Appendix E. An Example of "Notice for Completion Installation" 136
Appendix F. Set E-FOTA agent config by managed Configuration 137
Appendix G. Set up mutual TLS 138

Tables of Figures & Tables

【**Figures**】

Fig 2-1 Knox E-FOTA On-Premises Product Arch for “PoC” 12
Fig 2-2 Knox E-FOTA On-Premises Product Arch for “Commercial” 14
Fig 3-1 Knox E-FOTA On-Premises Conceptual Architecture 15
Fig 4-1 Knox E-FOTA On-Premises Product Arch with config 19
Fig 4-2 IP-based Access Environment 20
Fig 4-3 Domain-Based Access Environment (Type A) 20
Fig 4-4 Domain-Based Access Environment (Type B) 21
Fig 4-5 Domain-Based Access Environment (Type C) 21
Fig 4-6 An Disk Partitions for DMF Module on DB(MySQL) server 26
Fig 4-7 An Disk Partitions for DMF Module on Firmware Storage(minio) server 46
Fig 4-8 An Disk Partitions for DMF Module on DFM core/console server 54
Fig 4-9 An Disk Partitions for DMF Module on WEB server 68
Fig 4-10 On Customer’s Load Balancer (Proxy) 80
Fig 4-11 On DFM Server 81
Fig 4-12 On DFM Server 83
Fig 5-1 The Admin Console for Knox E-FOTA On-Premises 87

【**Tables**】

Table 2-1 The Hardware Recommended for user work environment to this On-Premise 10
Table 2-2 The Software Recommended for user work environment to this On-Premise 10
Table 2-3 The Minimum Hardware Recommendation for “PoC” 11
Table 2-4 The Software Recommendation for “PoC” 12
Table 2-5 The Minimum Hardware Recommendation for “Commercial” 13
Table 2-6 Software Recommendation for “Commercial” 13

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	Red Hat Enterprise Linux 8.4, 9.2, or 9.6	
Docker Engine	Podman	
MySQL Edition	Community Edition	

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 Red Hat Enterprise Linux 8.4, 9.2, or 9.6 for the OS. It should be installed on 64-bit Intel x86, ARM, or MIPS architectures in order to support Podman.
Selinux mode in the Red hat OS supports permissive and enforcing modes, and the DFM Server can be installed in each mode.

2.2.2. Keepalived

Keepalived is a routing software for loadbalancing and high-availability to Linux system and Linux based infrastructures. Keepalived implements a set of checkers to dynamically and adaptively maintain and manage loadbalanced server pool according their health. On the other hand, high-availability is achieved by the Virtual Router Redundancy Protocol (VRRP). VRRP is a fundamental brick for router failover.
This is used for continuous use in the event of a server failure.

2.2.1. Podman

Podman is a daemonless container engine for developing, managing, and running OCI Containers on your Linux System. Containers can either be run as root or in rootless mode. Since Red Hat Enterprise Linux 8, Podman is officially supported. For Knox E-FOTA On-Premises, Podman is the recommended container tool for Red Hat users. Podman installation is described in Section 4.1

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	**Server CPU Cores**	1 CPU Cores	

(MySQL)	**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	Red Hat Enterprise Linux 8.4, 9.2, or 9.6.	
Docker Engine	Podman	
MySQL Edition	Community 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.

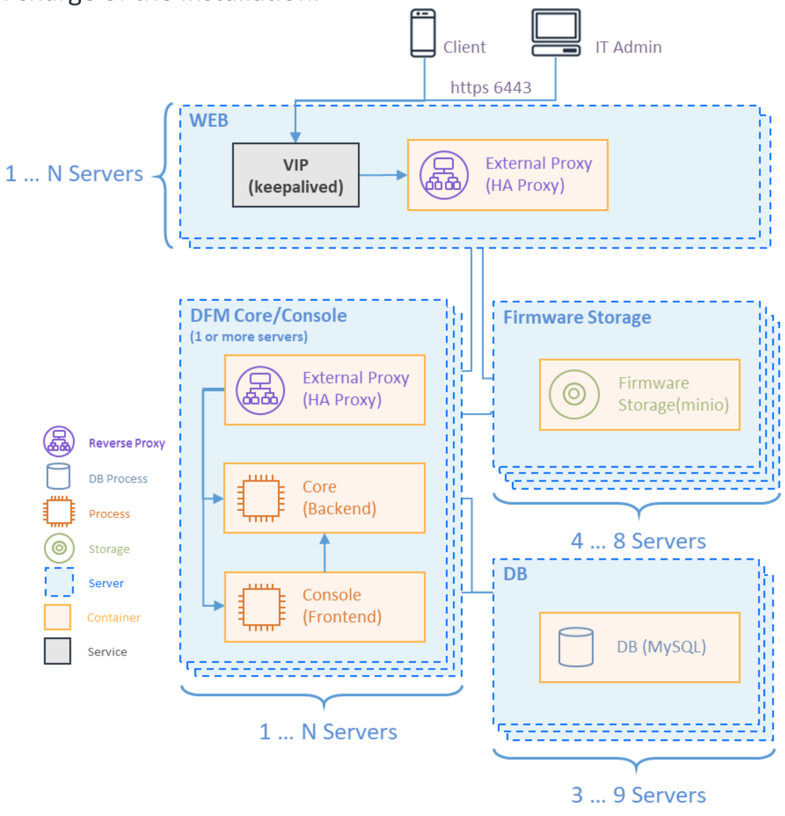


Fig 2-1 Knox E-FOTA On-Premises Product Arch for PoC

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 Red hat OS based this service package, depending on their service environment. Note that the customer must provide the service infrastructure to the Samsungrepresentative in charge of the installation.

Items	Recommended Value	Description
Operating System	Red Hat Enterprise Linux 8.4, 9.2, or 9.6	
Docker Engine	Podman	
MySQL Edition	Community Edition	

Table 2-6 Software Recommendation for “Commercial”

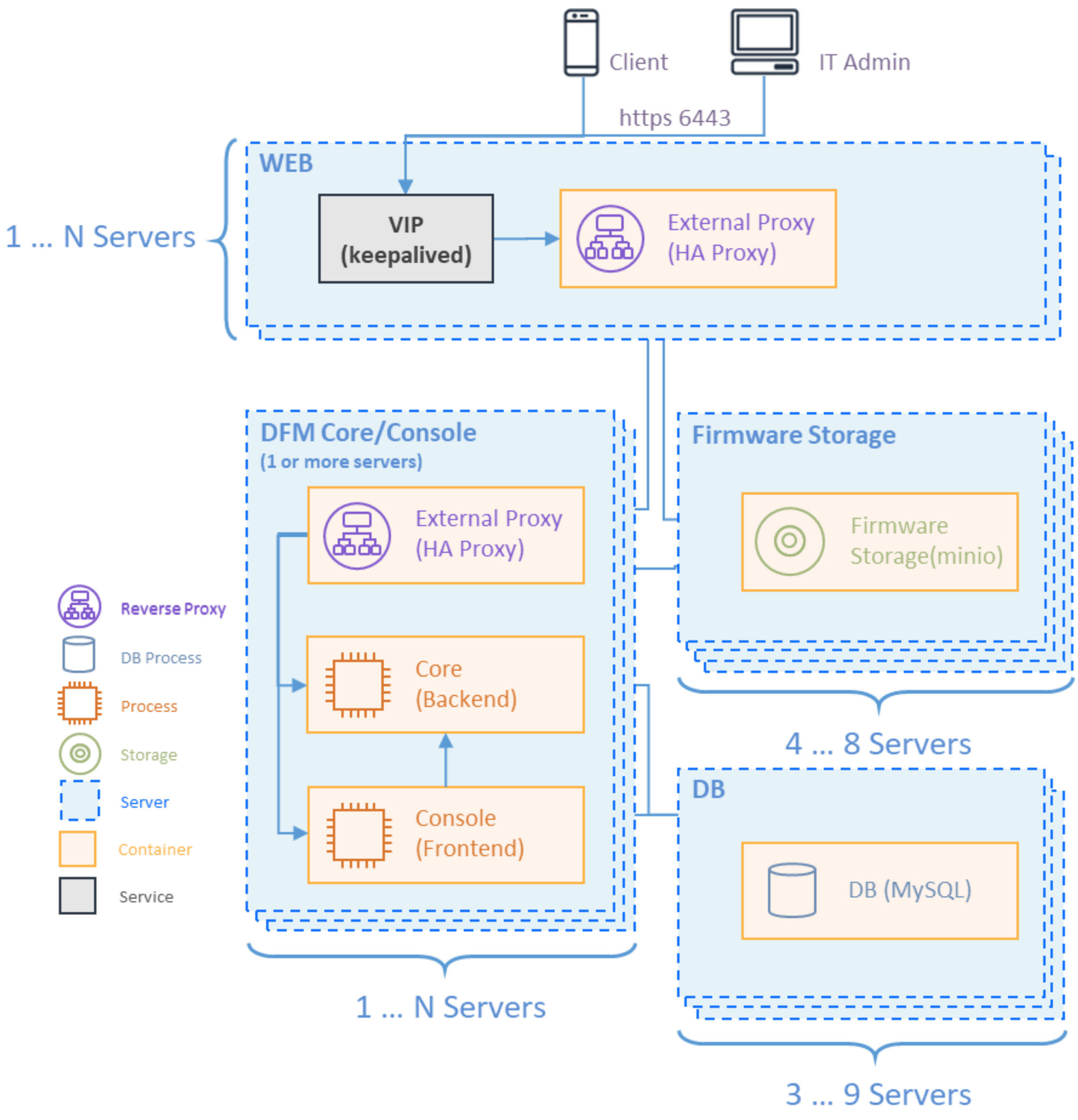


Fig 2-2 Knox E-FOTA On-Premises Product Arch for “Commercial”

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

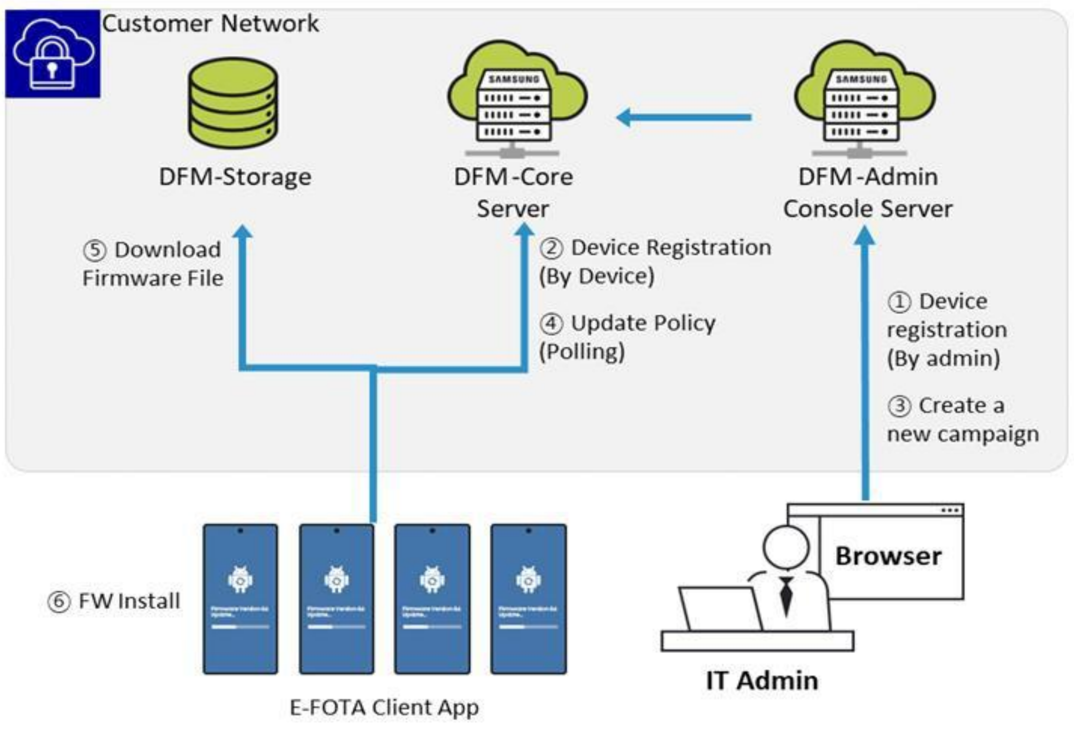


Fig 3-1 Knox E-FOTA On-Premises Conceptual Architecture

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.

Podman can run in root permission mode and rootless permission mode. If you run in root permission mode, you must attach “sudo” in front of the podman or dfm command. The following document is written based on rootless mode. If a command requires "sudo" because the system is running in root mode, "sudo is required in root mode." will be shown in the document.

DFM Modules must be installed on each of the 10 servers: the HA-Proxy server, DFM Core/Console server, Firmware Storage server (at least 4), and MySQL server (at least 3).

For the more details about each server specification, explain each server install information.

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 asa container. The config values determined for each step are set during the installation process for each server.

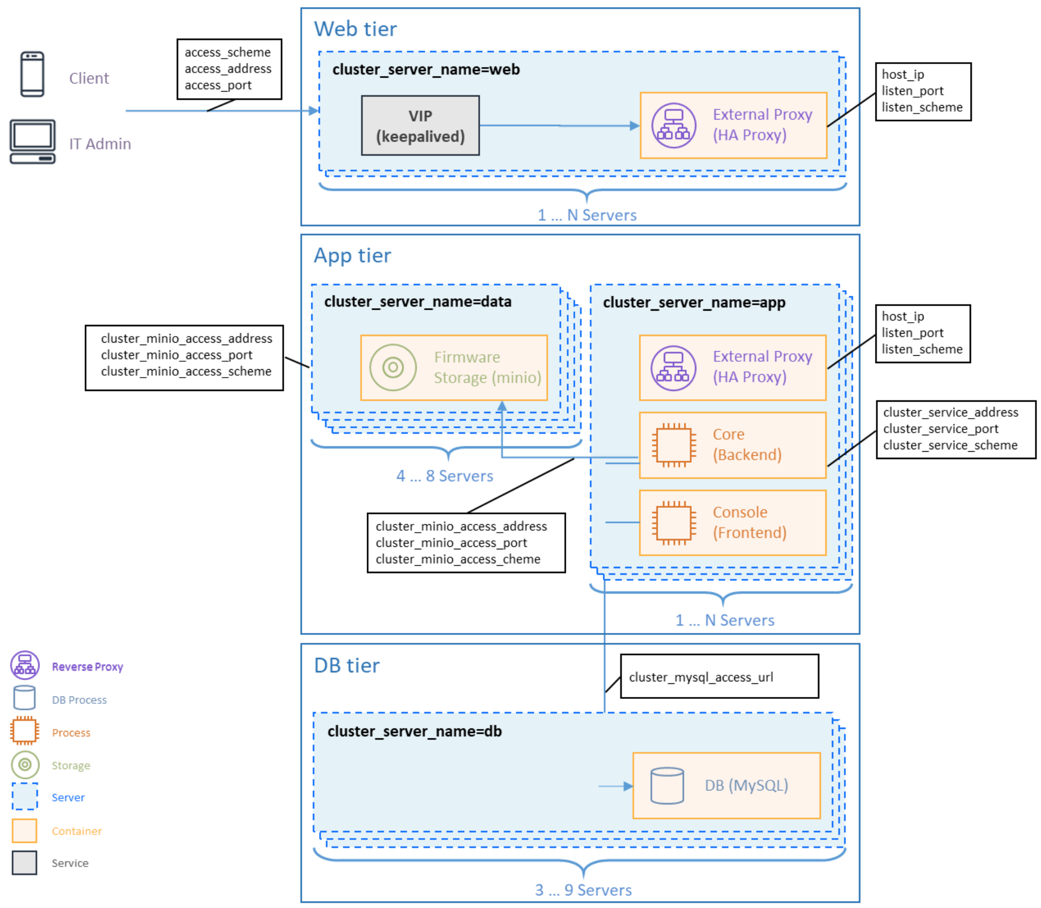


Fig 4-1 **Knox E-FOTA On-Premises product arch with config**

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.
It receives all requests through the VIP (Virtual IP) set in KEEPALIVED and forwards them to HAPROXY on the listen port set in “Web tier”.

【Configuration List】

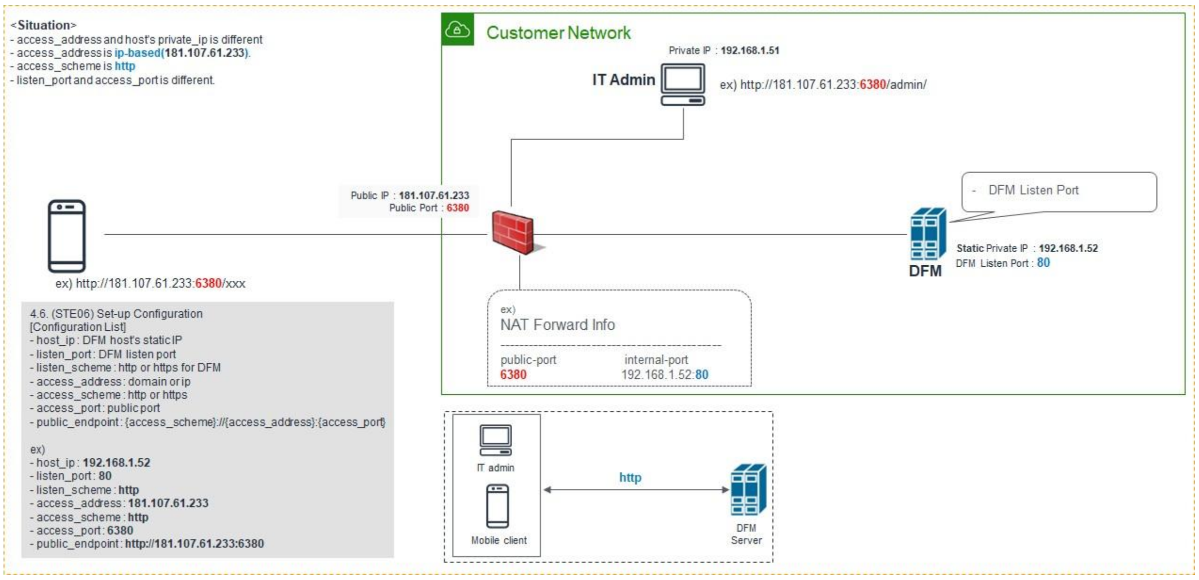
- cluster_server_name: web(fixed value)
- host_ip: Static IP for DFM server.
- listen_port: External listen port at server for DFM module to be accessed.
- listen_scheme: url scheme(http or https) for DFM module to be accessed.
- access_address: domain-based or ip-based
- access_scheme: http or https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



<Situation>
- access_address and host's private_ip is different
- access_address is ip-based(181.107.61.233).
- access_scheme is http
- listen_port and access_port is different.

4.6. (STE06) Set-up Configuration
[Configuration List]
- host_ip : DFM host's static IP
- listen_port : DFM listen port
- listen_scheme : http or https for DFM
- access_address : domain or ip
- access_scheme : http or https
- access_port : public port
- public_endpoint : {access_scheme}://{access_address}:{access_port}

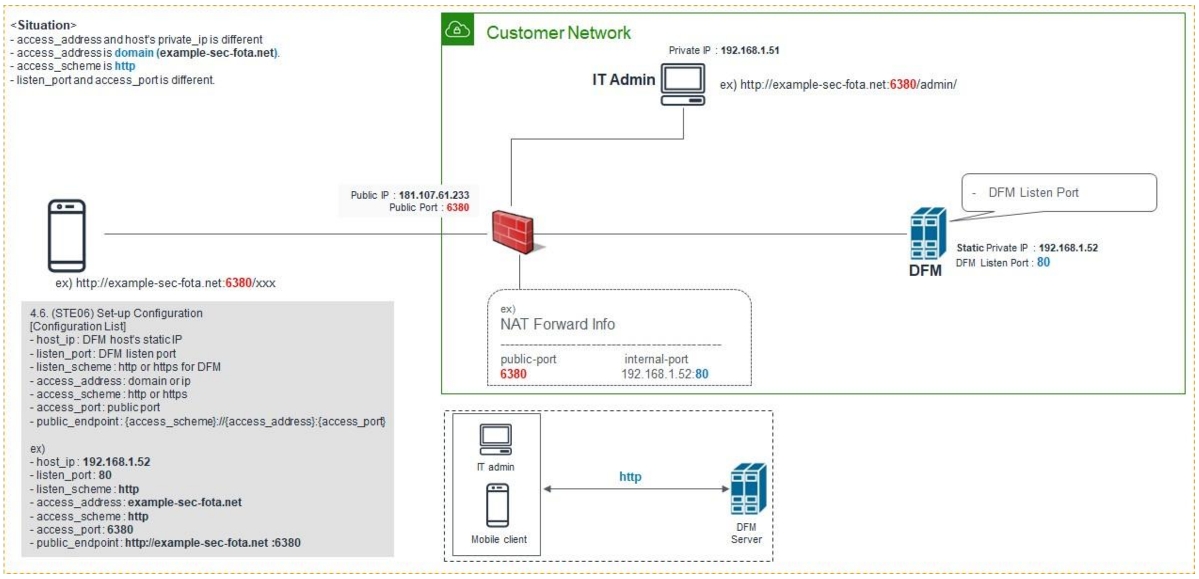
ex)
- host_ip : **192.168.1.52**
- listen_port : **80**
- listen_scheme : **http**
- access_address : **181.107.61.233**
- access_scheme : **http**
- access_port : **6380**
- public_endpoint : **http://181.107.61.233:6380**

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**



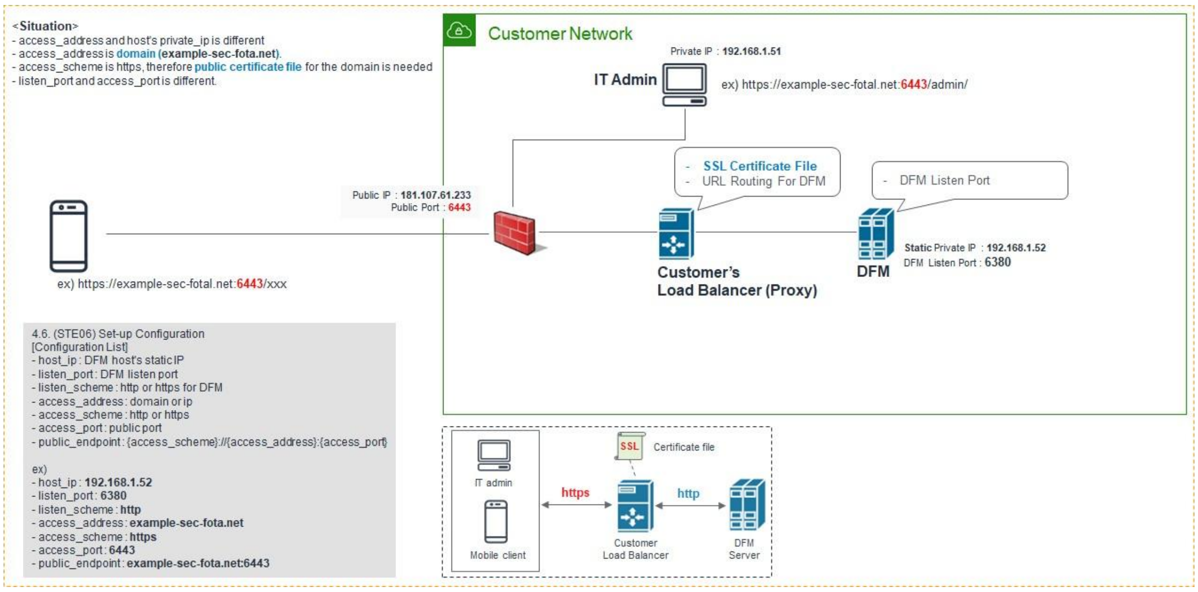
<Situation>
- access_address and host's private_ip is different
- access_address is domain (example-sec-fota.net).
- access_scheme is http
- listen_port and access_port is different.

4.6. (STE06) Set-up Configuration
[Configuration List]
- host_ip : DFM host's static IP
- listen_port : DFM listen port
- listen_scheme : http or https for DFM
- access_address : domain or ip
- access_scheme : http or https
- access_port : public port
- public_endpoint : {access_scheme}://{access_address}:{access_port}

ex)
- host_ip : **192.168.1.52**
- listen_port : **80**
- listen_scheme : **http**
- access_address : **example-sec-fota.net**
- access_scheme : **http**
- access_port : **6380**
- public_endpoint : **http://example-sec-fota.net :6380**

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

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



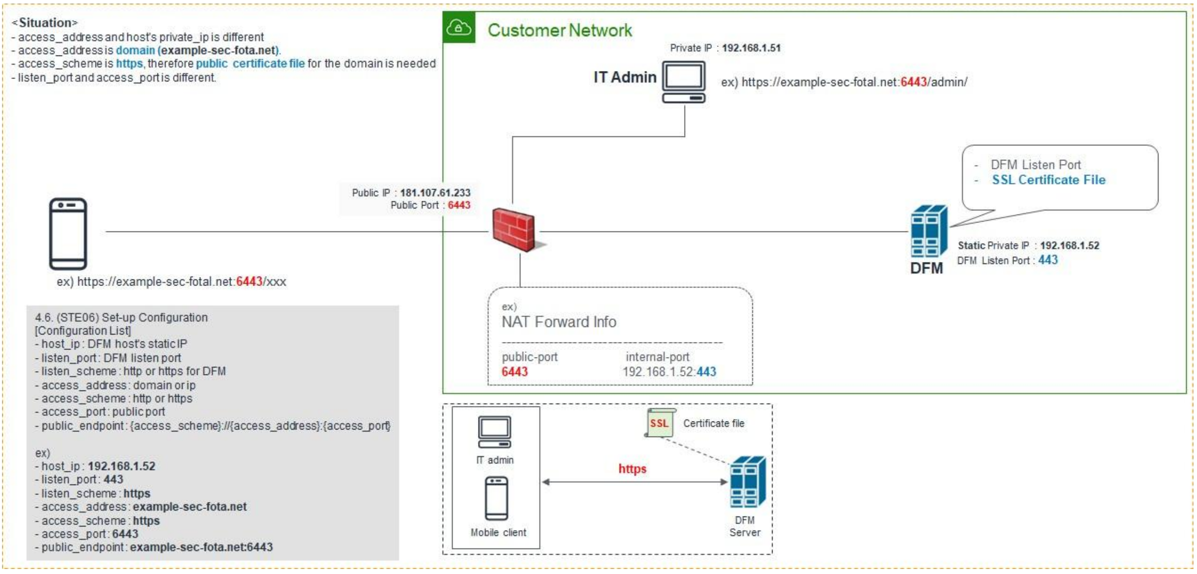
<Situation>
- access_address and host's private_ip is different
- access_address is domain (example-sec-fota.net).
- access_scheme is https, therefore public certificate file for the domain is needed
- listen_port and access_port is different.

4.6. (STE06) Set-up Configuration
[Configuration List]
- host_ip : DFM host's static IP
- listen_port : DFM listen port
- listen_scheme : http or https for DFM
- access_address : domain or ip
- access_scheme : http or https
- access_port : public port
- public_endpoint : {access_scheme}://{access_address}:{access_port}

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

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

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



<Situation>
- access_address and host's private_ip is different
- access_address is domain (example-sec-fota.net).
- access_scheme is https, therefore public certificate file for the domain is needed
- listen_port and access_port is different.

4.6. (STE06) Set-up Configuration
[Configuration List]
- host_ip : DFM host's static IP
- listen_port : DFM listen port
- listen_scheme : http or https for DFM
- access_address : domain or ip
- access_scheme : http or https
- access_port : public port
- public_endpoint : {access_scheme}://{access_address}:{access_port}

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

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

The following is <u>**an example**</u> 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)
```

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)
- cluster_minio_access_address: external access address(default: dfm-proxy)
- cluster_minio_scheme: url scheme(default: http)
- minio_config_dir: config 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: http
- cluster_mysql_access_url: access to database url
```

4.1.5 Using the firewalld service

If the firewalld service is in operation on the customer’s server, a service port must be added.

```
// List all allowed ports
sudo firewall-cmd --list-ports

// Add a port to the allowed ports to open it for incoming traffic
sudo firewall-cmd --add-port={port number}/tcp

// Make the new settings persistent
sudo firewall-cmd --runtime-to-permanent

Example)
dfm config get listen_port
443
dfm config get access_port
4443
sudo firewall-cmd --add-port=443/tcp --add-port=4443
sudo firewall-cmd --runtime-to-permanent
```

4.1.6 Configure Access port

In Red Hat OS, connection is restricted for ports below 1024.We need to set it to listen_port, which was set above:

```
sudo sysctl net.ipv4.ip_unprivileged_port_start={port number}


Example)
// check listen_port
dfm config get listen_port
443
// open port to listen_port
sudo sysctl net.ipv4.ip_unprivileged_port_start=443
```

4.2. DB(MySQL) Server

Use the Group Replication feature of MySQL enterprise to configure HA for the DB server.

There is a built-in group membership service that keeps the view of the group consistent and available for all servers at any given point in time. Servers can leave and join the group and the view is updated accordingly. Sometimes servers can leave the group unexpectedly, in which case the failure detection mechanism detects this and notifies the group that the view has changed. This is all automatic.
In single-primary mode, the group has a single primary server that is set to read-write mode. All the other members in the group are set to read-only mode.

The DFM solution is implemented in single-primary mode.

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 command permissions.

4.2.1 (Prerequisites) Install Package

If the installed Podman version is between 4.0 to 4.4 in Red Hat Enterprise Linux 8.4, 9.2, or 9.6, an additional package installation is required.

The following describes the package that needs to be installed:

podman: manage container tool
podman-plugins: the CNI plugins used to run podman
dnsmasq: use to find local dnsname

【Default】
sudo yum install -y podman
【Red hat 8.4, 9.2, or 9.6 podman 4.0 ~ 4.4】
sudo yum install -y podman podman-plugins dnsmasq

4.2.2 (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 command permission. Ensure you add your service account into the Wheel group.
We recommend that you create a service account before you start the installation.
The following command shows you how to add your service account into the Wheel group:
We assume that you are using the "**nightwatch**" account, and that the DFM Module is logged in with **a dedicated service account** and operates with the privileges of the account.

Add wheel group

```
sudo usermod --append -G wheel <username>

Example)
 sudo usermod --append -G wheel nightwatch
```

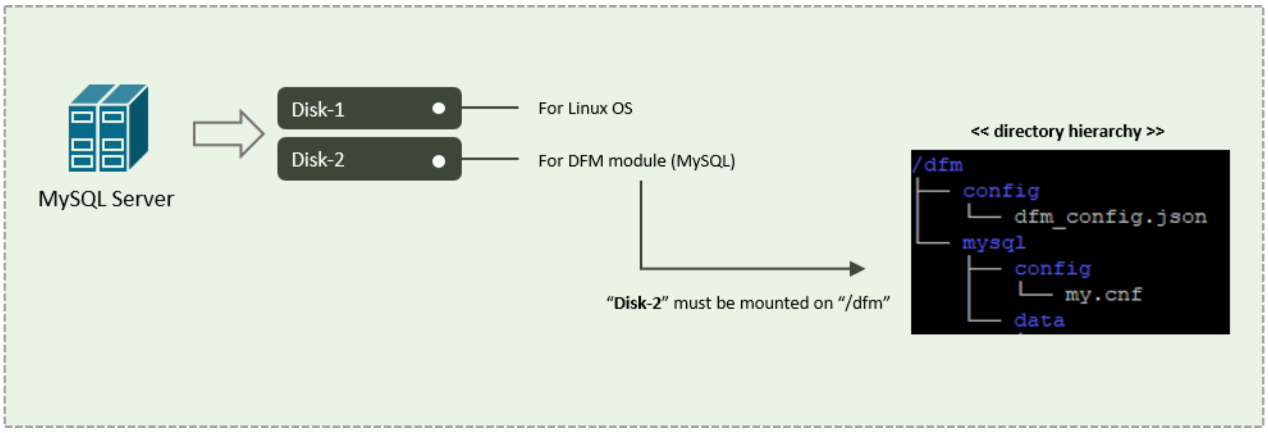
To connect using a created user:

```
ssh {your-user}@localhost [-p {port}]

Example)
 ssh nightwatch@localhost

or if you use port 9000
 ssh nightwatch@localhost -p 9000
```

4.2.3 (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.



```
/dfm
├── config
│   └── dfm_config.json
└── mysql
    ├── config
    │   └── my.cnf
    └── data
```

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

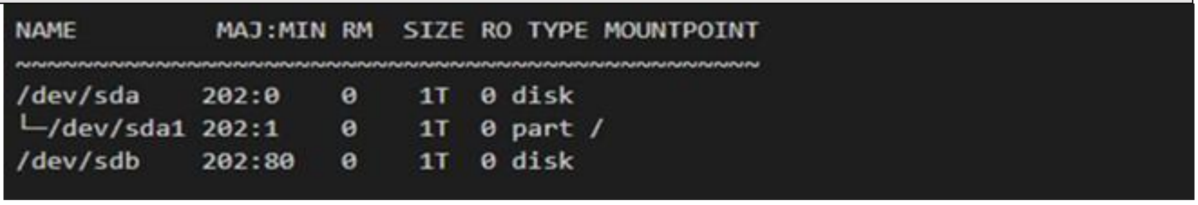
For example, we assume that two disks ("sda" and "sdb") exist.

【CASE01】 Disk is Ready

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

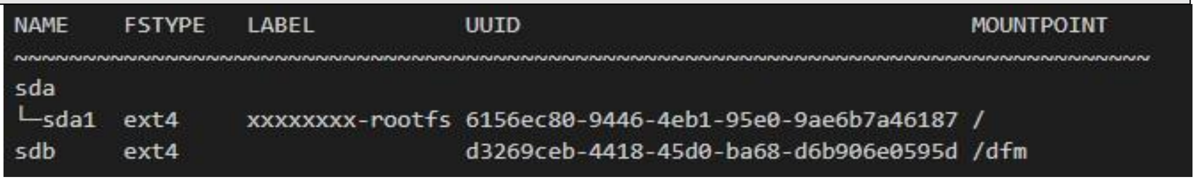
Now, let's check the disk information:

```
sudo lsblk -p
```



```
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
```

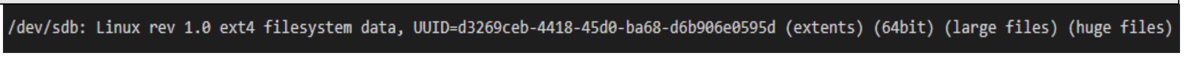
```
sudo lsblk -f
```



```
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
```

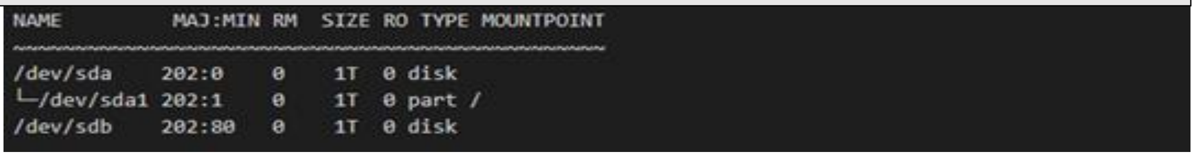


```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

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

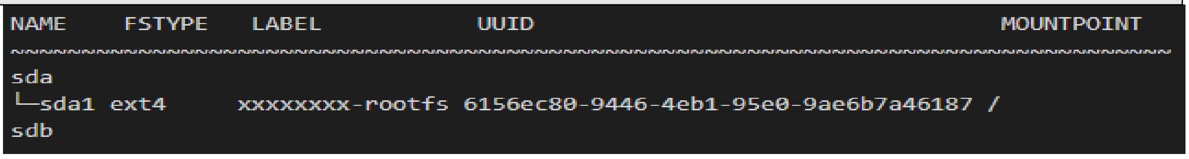
If the disk is not ready, it needs to be formatted and mounted on **/dfm**.
Now, let's check the disk information:

```
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
```

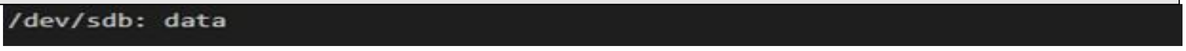
```
sudo lsblk -f
```



```
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
```

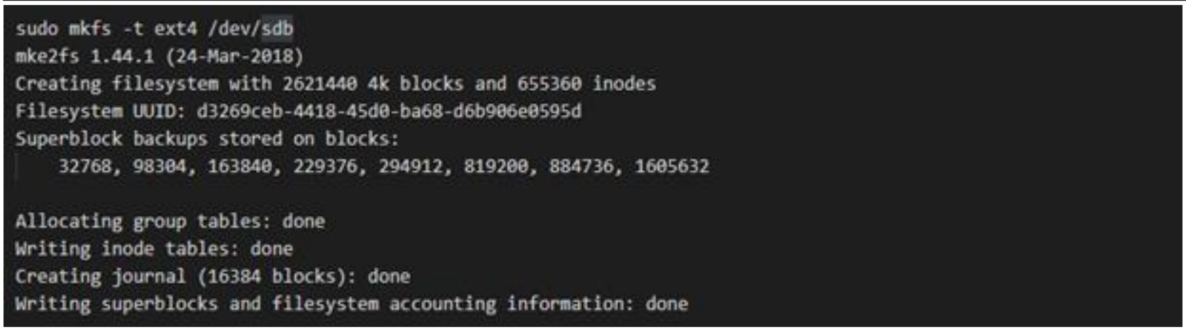


```
/dev/sdb: data
```

⇨ This means that the disk needs to be formatted

1) **Format with ext4 file-system**

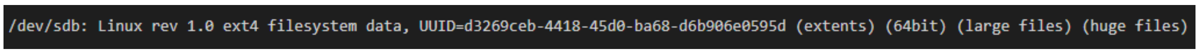
```
sudo file -s /dev/sdb
```



```
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
```



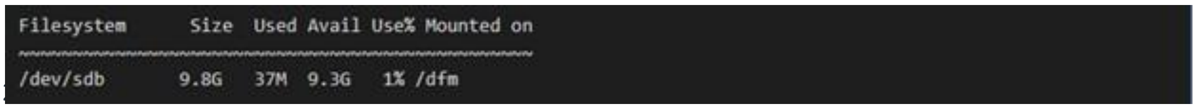
```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

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

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

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

4) **Verify**

```
df -h
```



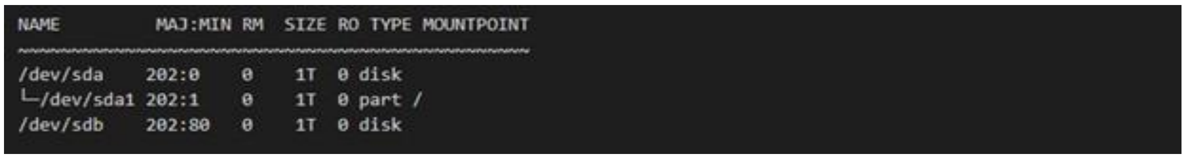
```
Filesystem      Size  Used Avail Use% Mounted on
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/dev/sdb        9.8G   37M  9.3G   1% /dfm
```

【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**.

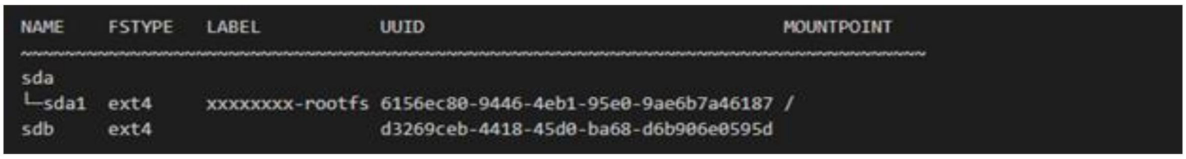
Now, let's check the disk information:

```
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
```

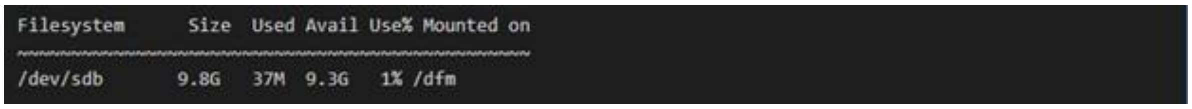
⇨ "sdb" " is formatted but Not yet mounted

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
```

4.2.4 Permanently mount the disk

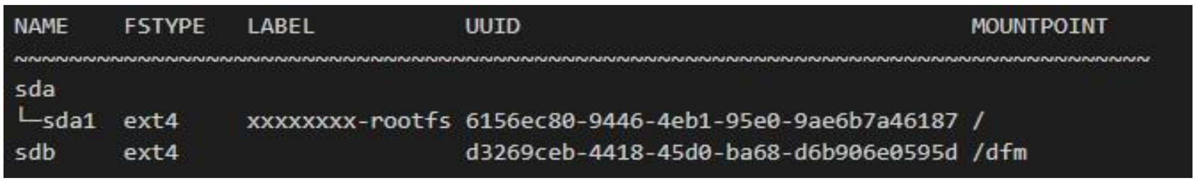
We recommend that the **customer's IT manager** sets the boot script so that **the dedicated disk** is auto-mounted when the server is booted.

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
```

4.2.5 (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
```

4.2.6 (STEP04) Install DFM Module Package

The DFM Module is delivered as a TAR archive file. 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) Container Image load**, **2) initializing MySQL DB**, and **3) Copying the config file** to the service directory.

- executable binary:
  - ⇒ /tmp/sec-dfm_1.0.1.11/dfm/bin/dfm
- docker images:
  - ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/haproxy-debian-{version}.tar
  - ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/minio-RELEASE. {version}.tar
  - ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/mysql-{version}.tar
  - ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/dfm-console-{version}.tar
  - ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/dfm-core-{version}.tar
- sql query file:
  - ⇒ /tmp/sec-dfm_1.0.1.11/dfm/mysql-query/init_db.sql
  - ⇒ /tmp/sec-dfm_1.0.1.11/dfm/mysql-query/init_dfm_core.sql
  - ⇒ /tmp/sec-dfm_1.0.1.11/dfm/mysql-query/init_dfm_console.sql
- mysql config file:
  - ⇒ /tmp/sec-dfm_1.0.1.11/dfm/ha/db-server/mysql/config/my.cnf
- dfm config file:
  - ⇒ /tmp/sec-dfm_1.0.1.11/dfm/ha/db-server/config/dfm_config.json

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

```
    tar -zxvf sec-dfm_{version}.tar.gz -C /tmp

  example)
    tar -zxvf  sec-dfm_1.0.1.11.tar.gz -C /tmp

  /tmp/sec-dfm_1.0.1.11/dfm/
  /tmp/sec-dfm_1.0.1.11/dfm/mysql-query/
  /tmp/sec-dfm_1.0.1.11/dfm/haproxy-config/
  ……………..
  /tmp/sec-dfm_1.0.1.11/dfm/bin/
  ……………..
```

Next, check if the necessary files exist:

```
1) check dfm file
ls -l /tmp/sec-dfm_1.0.1.11/dfm/bin/dfm

2) check images
ls /tmp/sec-dfm_1.0.1.11/dfm/images/ -l
total 971552
-rw-rw-r-- 1 dfm-console-1.0.1.11.tar
-rw-rw-r-- 1 dfm-core-1.0.1.11.tar
-rw-rw-r-- 1 haproxy-debian-2.2.33.tar
-rw-rw-r-- 1 minio-RELEASE.2022-04-30T22-23-53Z.tar
-rw-rw-r-- 1 mysql-8.0.36.tar

3) check sql query file
ls /tmp/sec-dfm_1.0.1.11/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/sec-dfm_1.0.1.11/dfm/ha/db-server/mysql/config/ -l
total 4
-rw-r--r-- my.cnf

5) dfm config file : dfm_config.json
ls /tmp/sec-dfm_1.0.1.11/dfm/ha/db-server/config/dfm_config.json
/tmp/sec-dfm_1.0.1.11/dfm/ha/db-server/dfm_config.json
```

4.2.7 (STEP05) Load Container Image

Next, register the Images that were unpacked at "**/tmp/sec-dfm_1.0.1.11/dfm/images**". The loaded Container Images are used when the container is driven. The following shows how to load Container image required for DB(MySQL) server using Podman commands: (sudo is required in root mode.)

```
podman load < /tmp/sec-dfm_1.0.1.11/dfm/images/mysql-{version}.tar
```

Next, check if the MySQL Podman image was loaded. Use the "Podman Images" command: (sudo is required in root mode.)

```
Example)
 podman images

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

4.2.8 (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 executable binary:

```
// copy executable binary
  sudo cp /tmp/sec-dfm_1.0.1.11/dfm/bin/dfm /usr/bin/ or cp /tmp/sec-
  dfm_1.0.1.11/dfm/bin/dfm /usr/local/bin

// Set executable
sudo chmod 755 /usr/bin/dfm or sudo chmod 755 /usr/local/bin/dfm
```

- **copy mysql config file**:

```
// copy configuration file
cp /tmp/sec-dfm_1.0.1.11/dfm/ha/db-server/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/sec-dfm_1.0.1.11/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
```

4.2.9 (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".

Proceed with the two steps below.

1) Configure dfm_config.json file. (cluster_server_name)
2) Configure mysql file. (my.cnf)

1) Configure dfm_config.json file. "cluster_server_name" is "db"

【Configuration List】

- cluster_server_name: db (fixed value)
- host_ip: static ip

The following shows the commands: (sudo is required in root mode.)

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

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
```

2) Configure my.cnf file

If each server IP is 192.168.0.4, 192.168.0.5, 192.168.0.6, configure it as shown below:

【Configuration List】

- cluster_server_name: db (fixed value)
- host_ip: static ip
- bind_address: static ip
- report_host: static ip ip
- server_id: numeric (Set to 1 if primary server)
- loose-group_replication_group_name: value of uuid
- loose-group_replication_local_address: static ip and port(ex: 192.168.0.4:33161)
- loose-group_replication_group_seeds: IP and port of the server you need to connect (ex: 192.168.0.4:33161)

```
[client]
default-character-set=utf8mb4


[mysql]
default-character-set=utf8mb4


[mysqld]
user=mysql
```

```
default-time-zone='+00:00'
event_scheduler = ON
general_log = 0
slow-query-log = 1
long_query_time = 4
lower_case_table_names = 1
collation-server = utf8mb4_unicode_ci
init-connect='SET NAMES utf8mb4'
character-set-server = utf8mb4
group_concat_max_len = 4096

port=33061
mysqlx_port=33071

bind-address="192.168.0.100"
report_host="192.168.0.100"

skip-name-resolve

# # Disable other storage engines #
disabled_storage_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"
 # # Replication configuration parameters #
server_id=1
gtid_mode=ON
enforce_gtid_consistency=ON
binlog_checksum=NONE
# Not needed from 8.0.21 # # Group Replication configuration #
plugin_load_add='group_replication.so'
loose-group_replication_group_name=" f8ad695d-1fc9-49d6-9b70-7e296c86dc03"
loose-group_replication_start_on_boot=off
loose-group_replication_local_address= "192.168.0.100:33161"
loose-group_replication_group_seeds= ""
loose-group_replication_bootstrap_group=off
loose-group-replication-ssl-mode=REQUIRED

loose-group_replication_recovery_use_ssl=ON
```

【UUID Generate】

All MySQL servers must have the same UUID value.

```
uuidgen
ex)
uuidgen
14cfe0c5-fca1-47cd-89eb-cd8d23393dab
```

Show an example of my.cnf settings for servers with IPs of 192.168.0.4, 192.168.0.5, 192.168.0.6, respectively, with the UUIDs generated as above.

【Example about 192.168.0.4 server my.cnf】

```
[client]
default-character-set=utf8mb4


[mysql]
default-character-set=utf8mb4


[mysqld]
user=mysql
default-time-zone='+00:00'
event_scheduler = ON
general_log = 0
slow-query-log = 1
long_query_time = 4
lower_case_table_names = 1
collation-server = utf8mb4_unicode_ci
init-connect='SET NAMES utf8mb4'
character-set-server = utf8mb4
group_concat_max_len = 4096


port=33061
mysqlx_port=33071


bind-address="192.168.0.4"
report_host="192.168.0.4"


skip-name-resolve
```

```
# # Disable other storage engines #
disabled_storage_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"
 # # Replication configuration parameters #
server_id=1
gtid_mode=ON
enforce_gtid_consistency=ON
binlog_checksum=NONE
# Not needed from 8.0.21 # # Group Replication configuration #
plugin_load_add='group_replication.so'
loose-group_replication_group_name="14cfe0c5-fca1-47cd-89eb-cd8d23393dab"
loose-group_replication_start_on_boot=off
loose-group_replication_local_address= "192.168.0.4:33161"
loose-group_replication_group_seeds= "192.168.0.4:33161, 192.168.0.5:33161,
192.168.0.6:33161"
loose-group_replication_bootstrap_group=off
loose-group-replication-ssl-mode=REQUIRED

loose-group_replication_recovery_use_ssl=ON
```

【Example about 192.168.0.5 server my.cnf】

```
[client]
default-character-set=utf8mb4

[mysql]
default-character-set=utf8mb4

[mysqld]
user=mysql
default-time-zone='+00:00'
event_scheduler = ON
general_log = 0
slow-query-log = 1
long_query_time = 4
lower_case_table_names = 1
collation-server = utf8mb4_unicode_ci
init-connect='SET NAMES utf8mb4'
character-set-server = utf8mb4
```

```
group_concat_max_len = 4096

port=33061
mysqlx_port=33071

bind-address="192.168.0.5"
report_host="192.168.0.5"

skip-name-resolve

# # Disable other storage engines #
disabled_storage_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"
 # # Replication configuration parameters #
server_id=2
gtid_mode=ON
enforce_gtid_consistency=ON
binlog_checksum=NONE
# Not needed from 8.0.21 # # Group Replication configuration #
plugin_load_add='group_replication.so'
loose-group_replication_group_name="14cfe0c5-fca1-47cd-89eb-cd8d23393dab"
loose-group_replication_start_on_boot=off
loose-group_replication_local_address= "192.168.0.5:33161"
loose-group_replication_group_seeds= "192.168.0.4:33161, 192.168.0.5:33161,
192.168.0.6:33161"
loose-group_replication_bootstrap_group=off
loose-group-replication-ssl-mode=REQUIRED

loose-group_replication_recovery_use_ssl=ON
```

【Example about 192.168.0.6 server my.cnf】

```
[client]
default-character-set=utf8mb4

[mysql]
default-character-set=utf8mb4

[mysqld]
```

```
user=mysql
default-time-zone='+00:00'
event_scheduler = ON
general_log = 0
slow-query-log = 1
long_query_time = 4
lower_case_table_names = 1
collation-server = utf8mb4_unicode_ci
init-connect='SET NAMES utf8mb4'
character-set-server = utf8mb4
group_concat_max_len = 4096

port=33061
mysqlx_port=33071

bind-address="192.168.0.6"
report_host="192.168.0.6"

skip-name-resolve

# # Disable other storage engines #
disabled_storage_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"
 # # Replication configuration parameters #
server_id=3
gtid_mode=ON
enforce_gtid_consistency=ON
binlog_checksum=NONE
# Not needed from 8.0.21 # # Group Replication configuration #
plugin_load_add='group_replication.so'
loose-group_replication_group_name="14cfe0c5-fca1-47cd-89eb-cd8d23393dab"
loose-group_replication_start_on_boot=off
loose-group_replication_local_address= "192.168.0.6:33161"
loose-group_replication_group_seeds= "192.168.0.4:33161, 192.168.0.5:33161,
192.168.0.6:33161"
loose-group_replication_bootstrap_group=off
loose-group-replication-ssl-mode=REQUIRED
```

```
loose-group_replication_recovery_use_ssl=ON
```

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) Set Group replication.
3) 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: (sudo is required in root mode.)

```
dfm start dfm-mysql
```

【Validation】

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

```
 # If it is red hat 8.4 version, run health check
 podman healthcheck run dfm-mysql


 podman ps -a

CONTAINER ID  IMAGE                                   STATUS                      NAMES
2cd1bae13406  localhost/mysql/enterprise-server:8.0   Up 3 seconds ago (starting)  dfm-mysql

 $ podman ps -a

CONTAINER ID  IMAGE                                   STATUS                      NAMES
2cd1bae13406  localhost/mysql/enterprise-server:8.0   Up 52 seconds ago (healthy)  dfm-mysql
```

If the status is healthy, run each of the following commands. (sudo is required in root mode.)

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

```
 podman 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) Set Group replication. (sudo is required in root mode.)
Check the group_user password. We assume that the password is "**1q2w3e4r**".
We use this command: CREATE USER group_user@'%' IDENTIFIED BY '{password}' REQUIRE SSL

【Primary server】

Proceed to the primary server, or 192.168.0.4 server in the example above, for group replication.

```
podman exec -i dfm-mysql mysql -uroot -p1q2w3e4r
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> SET SQL_LOG_BIN=0;
Query OK, 0 rows affected (0.00 sec)
mysql> CREATE USER group_user@'%' IDENTIFIED BY '1q2w3e4r' REQUIRE SSL;
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT REPLICATION SLAVE ON *.* TO group_user@'%';
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT CONNECTION_ADMIN ON *.* TO group_user@'%';
Query OK, 0 rows affected (0.00 sec)
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
mysql> SET SQL_LOG_BIN=1;
Query OK, 0 rows affected (0.00 sec)

mysql> CHANGE MASTER TO MASTER_USER='group_user', MASTER_PASSWORD='1q2w3e4r'\
FOR CHANNEL 'group_replication_recovery';
Query OK, 0 rows affected (0.02 sec)

mysql> SET GLOBAL group_replication_bootstrap_group=ON;
Query OK, 0 rows affected (0.00 sec)
mysql> START GROUP_REPLICATION;
Query OK, 0 rows affected (4.56 sec)
mysql> SET GLOBAL group_replication_bootstrap_group=OFF;
Query OK, 0 rows affected (0.00 sec)
```

【Secondary server】

Proceed to the secondary server, or 192.168.0.5, 192.168.0.6 server in the example above, for group replication.

```
podman exec -i dfm-mysql mysql -uroot -p1q2w3e4r
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> SET SQL_LOG_BIN=0;
Query OK, 0 rows affected (0.00 sec)
mysql> CREATE USER group_user@'%' IDENTIFIED BY '1q2w3e4r' REQUIRE SSL;
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT REPLICATION SLAVE ON *.* TO group_user@'%';
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT CONNECTION_ADMIN ON *.* TO group_user@'%';
Query OK, 0 rows affected (0.00 sec)
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
mysql> SET SQL_LOG_BIN=1;
Query OK, 0 rows affected (0.00 sec)

mysql> CHANGE MASTER TO MASTER_USER='group_user', MASTER_PASSWORD='1q2w3e4r'\
FOR CHANNEL 'group_replication_recovery';
Query OK, 0 rows affected (0.02 sec)

mysql> START GROUP_REPLICATION;
Query OK, 0 rows affected (4.56 sec)
```

【validation】

```
mysql> SELECT MEMBER_HOST, MEMBER_PORT, MEMBER_STATE, MEMBER_ROLE FROM
performance_schema.replication_group_members;

 +---------------------+---------------------+-----------------------+---------------------+
 | MEMBER_HOST| MEMBER_PORT | MEMBER_STATE | MEMBER_ROLE |
 +---------------------+---------------------+-----------------------+---------------------+
 |     192.168.0.4  |               33061 |        ONLINE      | PRIMARY         |
 |     192.168.0.5  |               33061 |        ONLINE      | SECONDARY     |
 |     192.168.0.6  |               33061 |        ONLINE      | SECONDARY     |
 +---------------------+---------------------+-----------------------+---------------------+
```

3) Run the SQL query file: we assume that the password is “1q2w3e4r” (sudo is required in root mode.)

Important! DFM db initialization must be performed on the primary server.

For the example above, we'll connect to server 192.168.0.4.

```
podman exec -i dfm-mysql mysql -uroot -p1q2w3e4r < /tmp/sec-dfm_1.0.1.11/dfm/mysql-query/init_db.sql
mysql: [Warning] Using a password on the command line interface can be insecure.

podman exec -i dfm-mysql mysql -uroot -p1q2w3e4r < /tmp/sec-dfm_1.0.1.11/dfm/mysql-query/init_dfm_core.sql
mysql: [Warning] Using a password on the command line interface can be insecure.

podman exec -i dfm-mysql mysql -uroot -p1q2w3e4r < /tmp/sec-dfm_1.0.1.11/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

A server for storing firmware files and client files.
A minimum of four and a maximum of eight servers are required for an HA configuration.
Each server is automatically synchronized and will not function properly if the time is different between servers.

4.3.1 (Prerequisites) Install Package

If the installed Podman version is between 4.0 to 4.4 in Red Hat Enterprise Linux 8.4, 9.2, or 9.6, an additional package installation is required.

The following describes the package that needs to be installed:

podman: manage container tool
podman-plugins: the CNI plugins used to run podman
dnsmasq: use to find local dnsname

【Default】

```
sudo yum install -y podman
```

【Red hat 8.4, 9.2, or 9.6 podman 4.0 ~ 4.4】

```
sudo yum install -y podman podman-plugins dnsmasq
```

4.3.2 (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 command permission. Ensure you add your service account into the Wheel group.
We recommend that you create a service account before you start the installation.
The following command shows you how to add your service account into the Wheel group:
We assume that you are using the "**nightwatch**" account, and that the DFM Module is logged in with **a dedicated service account** and operates with the privileges of the account.

Add wheel group

```
sudo usermod --append -G wheel <username>

Example)
 sudo usermod --append -G wheel nightwatch
```

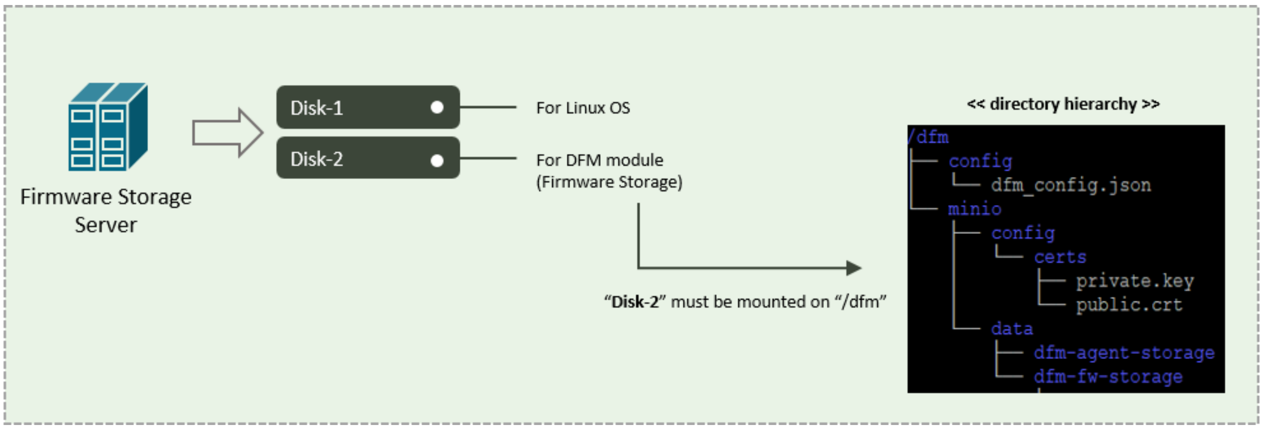
To connect using a created user:

```
ssh {your-user}@localhost [-p {port}]

Example)
 ssh nightwatch@localhost

or if you use port 9000
 ssh nightwatch@localhost -p 9000
```

4.3.3 (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.


Firmware Storage Server
Disk-1
Disk-2
For Linux OS
For DFM module (Firmware Storage)
“Disk-2” must be mounted on “/dfm”
<< directory hierarchy >>

```
/dfm
├─ config
│   └─ dfm_config.json
└─ minio
    ├─ config
    │   └─ certs
    │       ├─ private.key
    │       └─ public.crt
    └─ data
        ├─ dfm-agent-storage
        └─ dfm-fw-storage
```

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

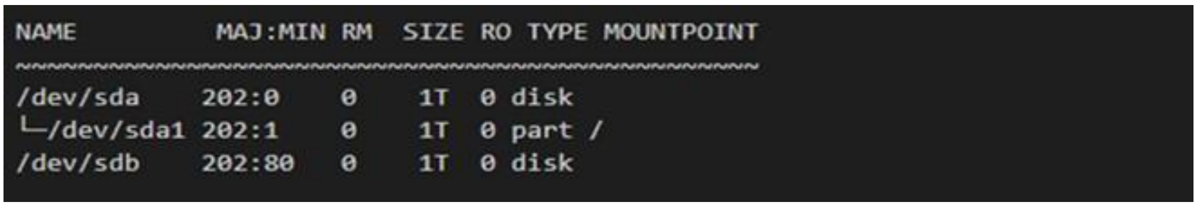
For example, we assume that two disks (“sda” and “sdb”) exist.

【CASE01】 Disk is Ready

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

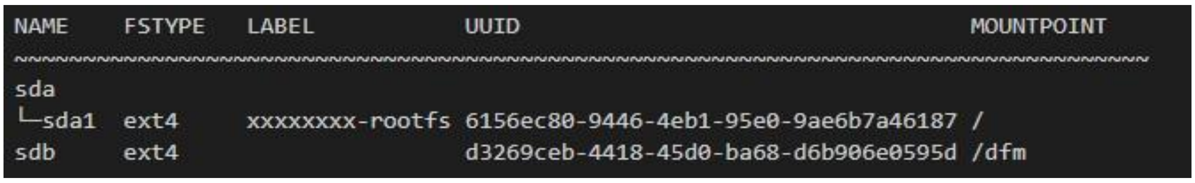
Now, let’s check the disk information:

```
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
```

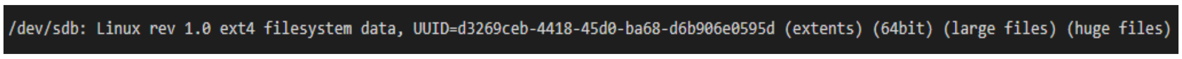
```
sudo lsblk -f
```



```
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
```



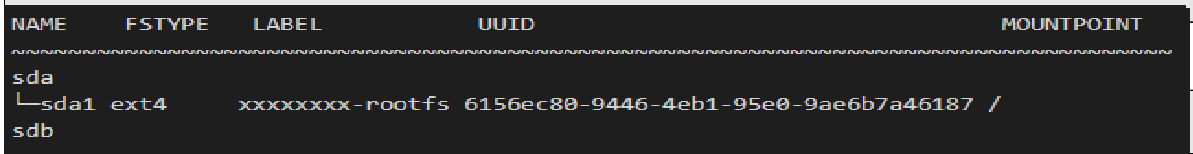
```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

【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:

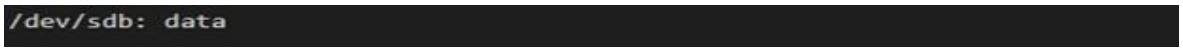
```
sudo lsblk -p
```



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

⇨ “sdb” is NOT formatted

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

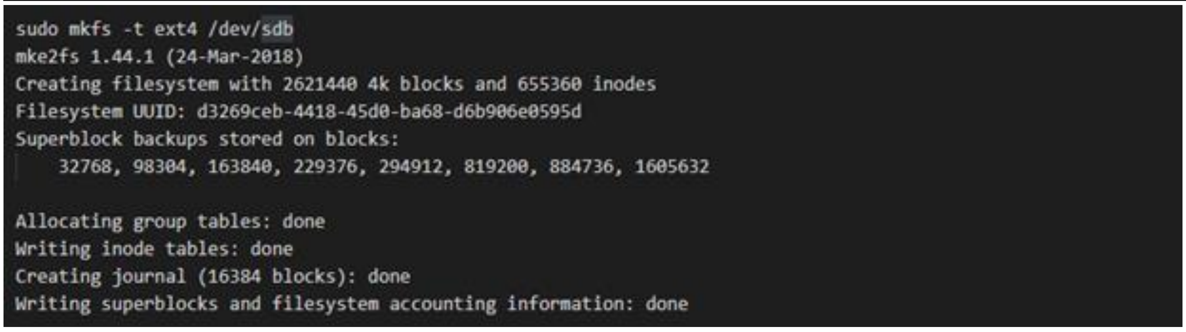


```
/dev/sdb: data
```

⇨ This means that the disk needs to be formatted

5) **Format with ext4 file-system**

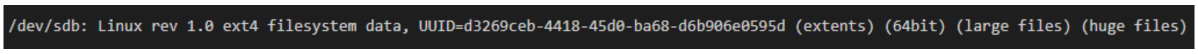
```
sudo file -s /dev/sdb
```



```
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
```



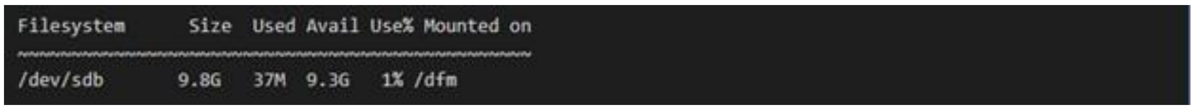
```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

7) **Mount “/dev/sdb” on /dfm**

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

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

8) **Verify**

```
df -h
```



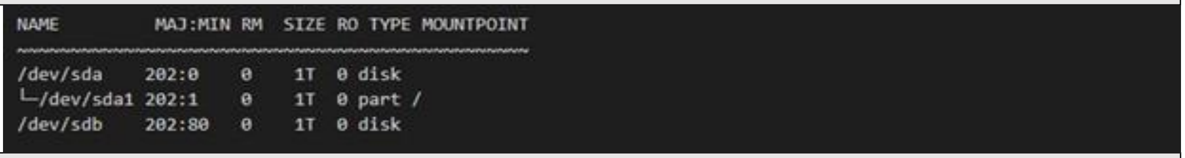
```
Filesystem      Size  Used Avail Use% Mounted on
----------------------------------------------------
/dev/sdb        9.8G   37M  9.3G   1% /dfm
```

【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
```



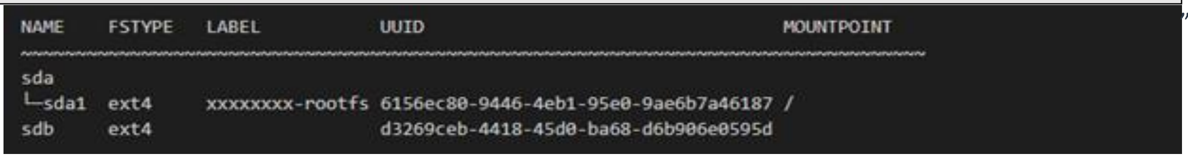
```
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
------------------------------------------------
/dev/sda     202:0    0    1T  0 disk
└─/dev/sda1  202:1    0    1T  0 part /
/dev/sdb     202:80   0    1T  0 disk
```

⇨ “

sdb”

```
sudo lsblk -f
```

” is



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

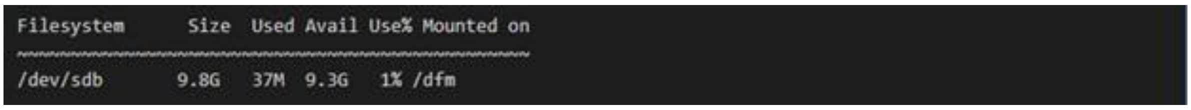
formatted but Not yet mounted

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
```

4.3.4 Permanently mount the disk

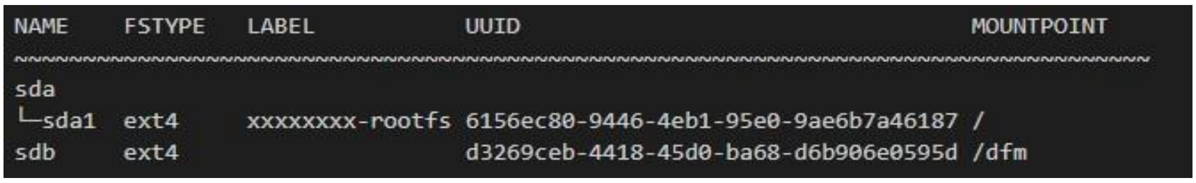
We recommend that the **customer's IT manager** sets the boot script so that **the dedicated disk** is auto-mounted when the server is booted.

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
```

4.3.5 (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

⇒ that is where efota client APK files and 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
sudo mkdir -p /dfm/minio/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/minio
sudo chown -R nightwatch:nightwatch /dfm/minio/config
sudo chown -R nightwatch:nightwatch /dfm/config
```

4.3.6 (STEP04) Install DFM Module Package

The DFM Module is delivered as a TAR archive file. 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) Container Image load**, and **2) Copying the config file** to the service directory.

- executable binary:
  ⇒ /tmp/sec-dfm_1.0.1.11/dfm/bin/dfm

- docker images:
  ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/haproxy-debian-{version}.tar
  ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/minio-RELEASE. {version}.tar
  ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/mysql-{version}.tar
  ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/dfm-console-{version}.tar
  ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/dfm-core-{version}.tar

- dfm config file:
  ⇒ /tmp/sec-dfm_1.0.1.11/dfm/ha/data-server/config/dfm_config.json

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

```
tar -zxvf sec-dfm_{version}.tar.gz -C /tmp

example)
tar -zxvf  sec-dfm_1.0.1.11.tar.gz -C /tmp

/tmp/sec-dfm_1.0.1.11/dfm/
/tmp/sec-dfm_1.0.1.11/dfm/mysql-query/
/tmp/sec-dfm_1.0.1.11/dfm/haproxy-config/
……………..
/tmp/sec-dfm_1.0.1.11/dfm/bin/
……………..
```

Next, check if the necessary files exist:

```
1) check dfm file
ls -l /tmp/sec-dfm_1.0.1.11/dfm/bin/dfm

2) check images
ls /tmp/sec-dfm_1.0.1.11/dfm/images/ -l
total 971552
-rw-rw-r-- 1 dfm-console-1.0.1.11.tar
-rw-rw-r-- 1 dfm-core-1.0.1.11.tar
-rw-rw-r-- 1 haproxy-debian-2.2.33.tar
-rw-rw-r-- 1 minio-RELEASE.2022-04-30T22-23-53Z.tar
-rw-rw-r-- 1 mysql-8.0.36.tar

3) dfm config file : dfm_config.json
ls /tmp/sec-dfm_1.0.1.11/dfm/ha/data-server/config/dfm_config.json
/tmp/sec-dfm_1.0.1.11/dfm/ha/data-server/config/dfm_config.json
```

4.3.7 (STEP05) Load Container Image

Next, register the Images that were unpacked at "**/tmp/sec-dfm_1.0.1.11/dfm/images**". The loaded container images are used when the container is driven. The following shows how to load container image required for Firmware storage (minio) server using Podman commands: (sudo is required in root mode.)

```
podman load -i /tmp/sec-dfm_1.0.1.11/dfm/images/minio-RELEASE.{version}.tar
```

Next, check if the minio image was loaded. Use the "Podman Images" command: (sudo is required in root mode.)

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

4.3.8 (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 executable binary:

```
// copy executable binary
  sudo cp /tmp/sec-dfm_1.0.1.11/dfm/bin/dfm /usr/bin/ or cp /tmp/sec-
  dfm_1.0.1.11/dfm/bin/dfm /usr/local/bin

// Set executable
sudo chmod 755 /usr/bin/dfm or sudo chmod 755 /usr/local/bin/dfm
```

- **copy dfm config file**:

```
// copy configuration file
cp /tmp/sec-dfm_1.0.1.11/dfm/ha/data-server/config/dfm_config.json /dfm/config

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

4.3.9 (STEP07) Set-up Configuration

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

【Configuration List】

- cluster_server_name: data (fixed value)
- cluster_minio_access_address: internal access address for each minio server(All minio server addresses separated by commas)
- cluster_minio_access_port: internal access port (default : 9000)
- minio_config_dir: ssl file location (default : /dfm/minio/config)

For example, a minio server is configured as a 192.168.0.10, 192.168.0.11, 192.168.0.12, 192.168.0.13 server.

The following shows the commands:

You can't use different ports for each server. If you use port 9001, you must set the other servers to port 9001 as well. Therefore, we recommend setting it up using the default port of 9000.

(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
```

Configure cluster_minio_access_address:

Enter the full server connection address, separated by spaces.

http://{minio server ip}/data1

```
dfm cluster config set cluster_minio_access_address=http://192.168.0.10/data1
http://192.168.0.11/data1 http://192.168.0.12/data1 http://192.168.0.12/data1
```

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: (sudo is required in root mode.)

```
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.

```
# If it is red hat 8.4 version, run health check
podman healthcheck run dfm-minio

podman ps -a

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

$ podman ps -a
CONTAINER ID    ~     STATUS                          NAMES
~~
af3949b8db9           Up 2 minutes (healthy)          dfm-minio
~~
```

4.4. DFM Core/Console Server

4.4.1 (Prerequisites) Install Package

If the installed Podman version is between 4.0 to 4.4 in Red Hat Enterprise Linux 8.4, 9.2, or 9.6, an additional package installation is required.

The following describes the package that needs to be installed:

podman: manage container tool
podman-plugins: the CNI plugins used to run podman
dnsmasq: use to find local dnsname

【Default】

```
sudo yum install -y podman
```

【Red hat 8.4, 9.2, or 9.6 podman 4.0 ~ 4.4】

```
sudo yum install -y podman podman-plugins dnsmasq
```

4.4.2 (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 command permission. Ensure you add your service account into the Wheel group.
We recommend that you create a service account before you start the installation.
The following command shows you how to add your service account into the Wheel group:
We assume that you are using the "**nightwatch**" account, and that the DFM Module is logged in with **a dedicated service account** and operates with the privileges of the account.

Add wheel group

```
sudo usermod --append -G wheel <username>

Example)
 sudo usermod --append -G wheel nightwatch
```

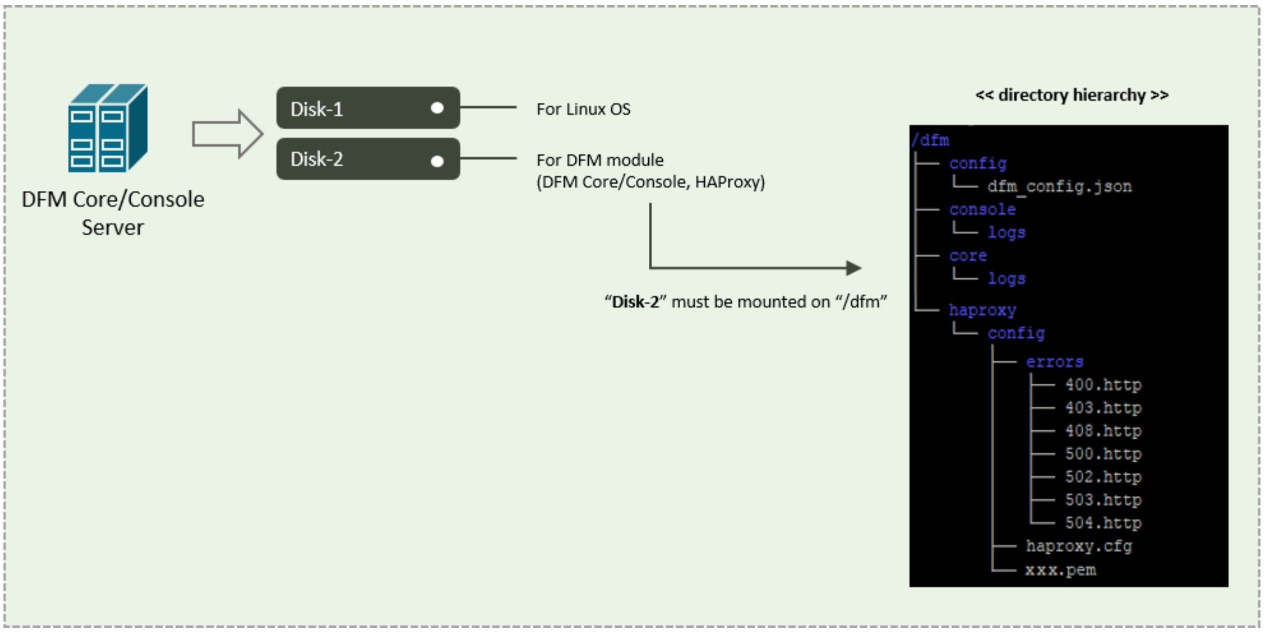
To connect using a created user:

```
ssh {your-user}@localhost [-p {port}]

Example)
 ssh nightwatch@localhost

or if you use port 9000
 ssh nightwatch@localhost -p 9000
```

4.4.3 (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.


DFM Core/Console Server
Disk-1
Disk-2
For Linux OS
For DFM module (DFM Core/Console, HAProxy)
“Disk-2” must be mounted on “/dfm”
<< directory hierarchy >>

```
/dfm
├─ config
│  └─ dfm_config.json
├─ console
│  └─ logs
├─ core
│  └─ logs
└─ haproxy
   └─ config
      ├─ errors
      │  ├─ 400.http
      │  ├─ 403.http
      │  ├─ 408.http
      │  ├─ 500.http
      │  ├─ 502.http
      │  ├─ 503.http
      │  └─ 504.http
      ├─ haproxy.cfg
      └─ xxx.pem
```

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

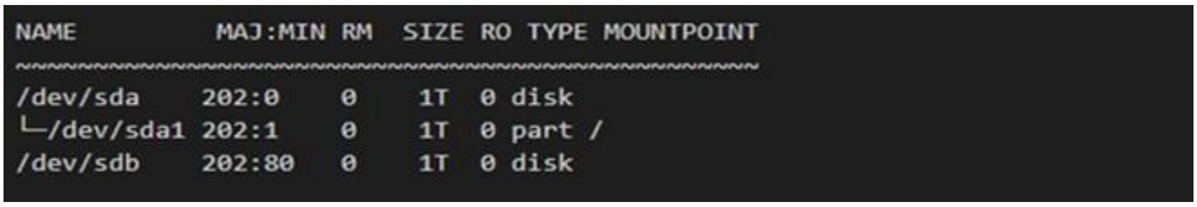
For example, we assume that two disks (“sda” and “sdb”) exist.

【CASE01】Disk is Ready

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

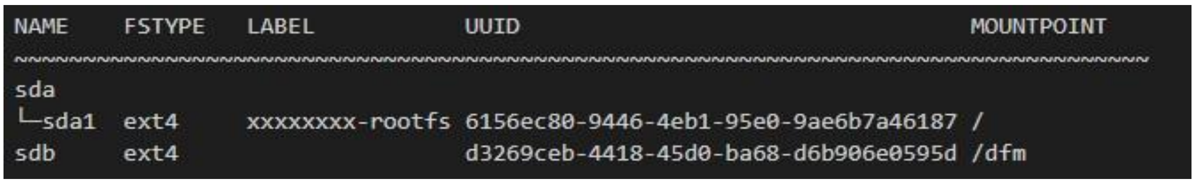
Now, let’s check the disk information:

```
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
```

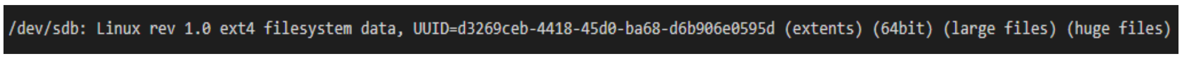
```
sudo lsblk -f
```



```
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
```



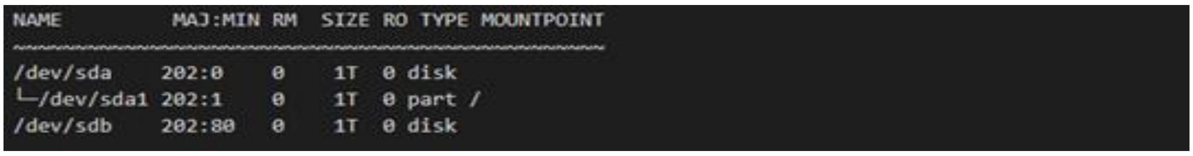
```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

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

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

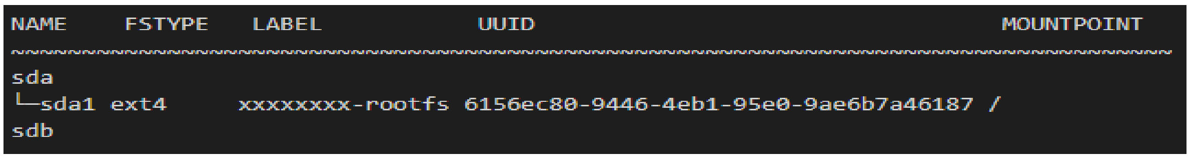
Now, let's check the disk information:

```
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
```

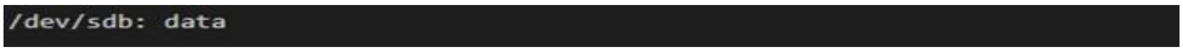
```
sudo lsblk -f
```



```
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
```

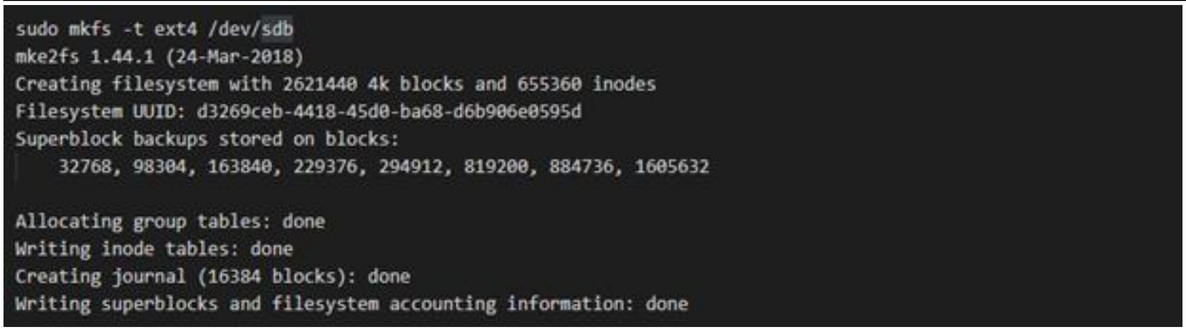


```
/dev/sdb: data
```

⇨ This means that the disk needs to be formatted

1) Format with ext4 file-system

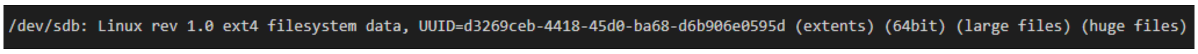
```
sudo file -s /dev/sdb
```



```
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
```



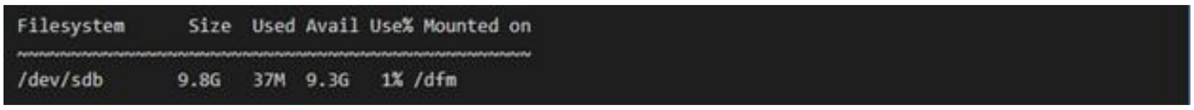
```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

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

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

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

4) Verify

```
df -h
```



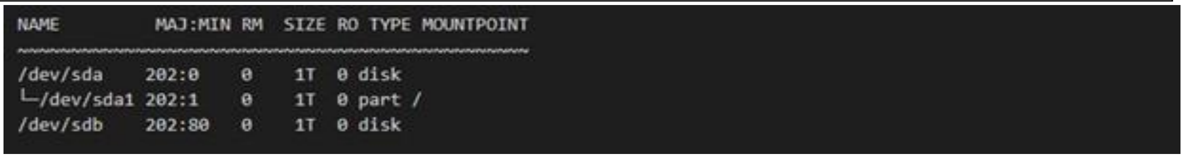
```
Filesystem      Size  Used Avail Use% Mounted on
-------------------------------------------------
/dev/sdb        9.8G   37M  9.3G   1% /dfm
```

【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**.

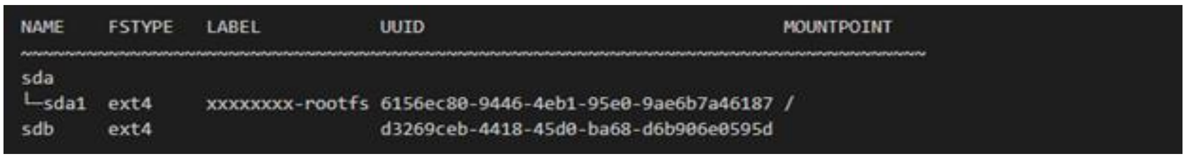
Now, let's check the disk information:

```
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
```

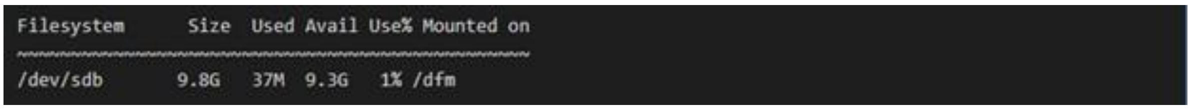
⇨ "sdb" " is formatted but Not yet mounted

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
```

4.4.4 Permanently mount the disk

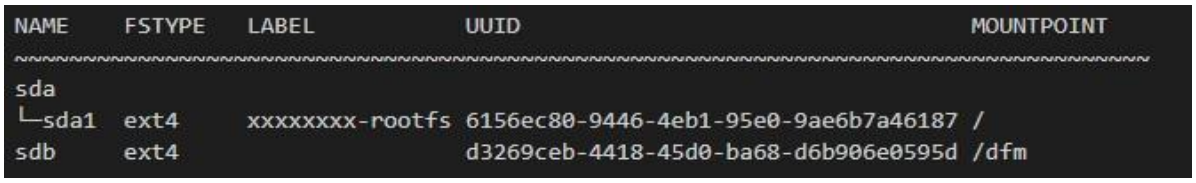
We recommend that the **customer's IT manager** sets the boot script so that **the dedicated disk** is auto-mounted when the server is booted.

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
```

4.4.5 (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
```

4.4.6 (STEP04) Install DFM Module Package

The DFM Module is delivered as a TAR archive file. 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) Container Image load**, **2) initializing MySQL DB**, and **3) Copying the config file** to the service directory.

```
- executable binary:
   ⇒ /tmp/sec-dfm_1.0.1.11/dfm/bin/dfm

- docker images:
   ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/haproxy-debian-{version}.tar
   ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/minio-RELEASE{version}.tar
   ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/mysql-{version}.tar
   ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/dfm-console-{version}.tar
   ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/dfm-core-{version}.tar

- haproxy config file:
   ⇒ /tmp/sec-dfm_1.0.1.11/dfm/ha/app-server/haproxy-config/haproxy.cfg

- dfm config file:
   ⇒ /tmp/sec-dfm_1.0.1.11/dfm/ha/app-server/config/dfm_config.json
```

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

```
    tar -zxvf sec-dfm_{version}-{package type}.tar.gz -C /tmp

   example)
    tar -zxvf  sec-dfm_1.0.1.11-root-enforcing.tar.gz -C /tmp

   /tmp/sec-dfm_1.0.1.11/dfm/
   /tmp/sec-dfm_1.0.1.11/dfm/mysql-query/
   /tmp/sec-dfm_1.0.1.11/dfm/haproxy-config/
   ................
   /tmp/sec-dfm_1.0.1.11/dfm/bin/
   ................
```

Next, check if the necessary files exist:

```
1) check dfm file
ls -l /tmp/sec-dfm_1.0.1.11/dfm/bin/dfm


2) check images
ls /tmp/sec-dfm_1.0.1.11/dfm/images/ -l
total 971552
-rw-rw-r-- 1 dfm-console-1.0.1.11.tar
-rw-rw-r-- 1 dfm-core-1.0.1.11.tar
-rw-rw-r-- 1 haproxy-debian-2.2.33.tar
-rw-rw-r-- 1 minio-RELEASE.2022-04-30T22-23-53Z.tar
```

```
-rw-rw-r-- 1 mysql-8.0.36.tar

4) check haproxy config file : haproxy.cfg
ls /tmp/sec-dfm_1.0.1.11/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/sec-dfm_1.0.1.11/dfm/ha/app-server/config/dfm_config.json
/tmp/sec-dfm_1.0.1.11/dfm/ha/app-server/config/dfm_config.json
```

4.4.7 (STEP05) Load Container Image

Next, register the Container Images that were unpacked at "**/tmp/sec-dfm_1.0.1.11/dfm/images**". The loaded Container Images are used when the container is driven. The following shows how to load Container Images required for DFM Core/Cosole server using Podman commands: (sudo is required in root mode.)

```
podman load -i /tmp/sec-dfm_1.0.1.11/dfm/images/haproxy-debian-{version}.tar
podman load -i /tmp/sec-dfm_1.0.1.11/dfm/images/dfm-core-{version}.tar
podman load -i /tmp/sec-dfm_1.0.1.11/dfm/images/dfm-console-{version}.tar
```

Next, check if the 3 Podman images were loaded. Use the "Podman Images" command: (sudo is required in root mode.)

```
Example)
podman images

 REPOSITORY                           TAG          IMAGE ID       CREATED       SIZE
 localhost/dfm-console                1.0.1.11     91343f0b589f   2 weeks ago   191 MB
 localhost/dfm-core                   1.0.1.11     0c838d5b5f1d   2 weeks ago   149 MB
 localhost/haproxytech/haproxy-debian 2.2.33       7ba2bc46a616   3 years ago   123 MB
```

4.4.8 (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 executable binary:

```
// copy executable binary
  sudo cp /tmp/sec-dfm_1.0.1.11/dfm/bin/dfm /usr/bin/ or cp /tmp/sec-
  dfm_1.0.1.11/dfm/bin/dfm /usr/local/bin

// Set executable
sudo chmod 755 /usr/bin/dfm or sudo chmod 755 /usr/local/bin/dfm
```

- **copy haproxy config file**:

```
// copy configuration file
cp /tmp/sec-dfm_1.0.1.11/dfm/ha/app-server/haproxy-config/haproxy.cfg
/dfm/haproxy/config

// copy error files
cp -rf /tmp/sec-dfm_1.0.1.11/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/sec-dfm_1.0.1.11/dfm/ha/app-server/config/dfm_config.json /dfm/config

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

4.4.9 (STEP07) Set-up Configuration

In this step, we will set up the initial configuration information needed for the DFM module to run asa Container.
Set the configuration reffering to the setting in **4.1. Check Pre-Config**.

【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(fixed: dfm-proxy)
- cluster_minio_access_port: firmware storage server port
- cluster_minio_access_scheme: firmware storage server protocol (http or https)

- cluster_mysql_access_url: database server address with port(ip or address)
- license_app_ip: license app access ip(if you do not need to change ip, the default value is set when input blank, sudo is required in root mode)

The following shows the commands:

cluster_mysql_access_address is set to the address used as an example when setting up the db server.

```
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=dfm-proxy
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_url=192.168.0.4:33061, 192.168.0.5:33061,
192.168.0.6:33061
dfm config set license_app_ip=
```

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
dfm-proxy

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_url
192.168.0.4:33061, 192.168.0.5:33061, 192.168.0.6:33061

dfm config get license_app_ip
100.0.0.1
```

4.4.10 (STEP08) Configure HAProxy

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

```
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
    balance leastconn
   # if minio server use ssl
   #server dfm-minio 192.168.0.11:9000 ssl verify none check
   # otherwise
    server dfm-minio-group-1 192.168.0.10:9000 check
    server dfm-minio-group-1 192.168.0.11:9000 check
    server dfm-minio-group-1 192.168.0.12:9000 check
    server dfm-minio-group-1 192.168.0.13:9000 check

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

4.4.11 (STEP09) Create Container Network

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

To create a network, use the following command: (sudo is required in root mode.)

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

4.4.12 (STEP10) Copy Background App files

Copy the following Background app files into theservice directoryfrom the unpacked resources directory. We assume that you are using the "**nightwatch**" account.

```
// copy background files
  cp /tmp/sec-dfm_1.0.1.11/dfm/licenseApp /dfm/background/licenseApp
  cp /tmp/sec-dfm_1.0.1.11/dfm/efota-license.service /etc/systemd/system/efota-
  license.service

// Set the service account's permission to the configuration file.
  sudo chmod 744 /dfm/background/licenseApp
  sudo chcon –t bin_t /dfm/background/licenseApp
```

4.4.13 (STEP11) Start-up Background App

In this stage, the installer starts the Background App for license check The command to run the background app is as follows:

```
sudo systemctl daemon-reload
sudo systemctl enable efota-license.service
sudo systemctl start efota-license.service
```

【Validation】

Make sure the Background app is running.

```
sudo systemctl status efota-license.service
Loaded: loaded (/etc/system/system/efota-license.service; enabled; vendor preset: enabled)
Active: active (running) since Tue 2024-XX-XX 06:39:10 UTC; 7s ago
Main PID: 2028 (licenseApp)
```

4.4.14 (STEP12) 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 HA proxy, Core and Console containers is as follows: (sudo is required in root mode.)

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

【Validation】

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

```
# If it is red hat 8.4 version, run health check
podman healthcheck run dfm-proxy
podman healthcheck run dfm-core
podman healthcheck run dfm-console
podman ps -a

Example)
$ podman 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
~~
$
$ podman 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 Keepalived

Install and set up keepalived on your web server before setting up your web server.
Assuming you have two web servers, this section describes the installation and setup process.

192.168.0.20(MASTER)
192.168.0.21(BACKUP)

4.5.1 (STEP01) Install package

Install the installation on each server with the yum package manager.

```
$ sudo yum install -y keepalived
```

4.5.2 (STEP02) Configure keepalived

【Configuration MASTER】

```
# vi /etc/keepalived/keepalived.conf
vrrp_instance VI_1 {
    state MASTER
    interface eth0
    virtual_router_id 51
    priority 255
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 12345
    }
    virtual_ipaddress {
        192.168.0.200/24
    }
}
```

【Configuration BACKUP】

```
# vi /etc/keepalived/keepalived.conf
vrrp_instance VI_1 {
      state BACKUP
      interface eth0
      virtual_router_id 51
      priority 255
      advert_int 1
      authentication {
           auth_type PASS
           auth_pass 12345
      }
      virtual_ipaddress {
           192.168.0.200/24
      }
}
```

4.5.3 (STEP03) Start-up keepalived

Start keepliaved on each server and verify that it is working properly.

```
sudo systemctl start keepalived
```

【Validation】

```
# 192.168.0.20(master server)
ip -br a

Example)

eth0 UP 192.168.0.20/24 192.168.0.200/24

# 192.168.0.21(backup server)
ip -br a

Example)

eth0 UP 192.168.0.21/24
```

4.6 WEB Server

4.6.1 (Prerequisites) Install Package

If the installed Podman version is between 4.0 to 4.4 in Red Hat Enterprise Linux 8.4, 9.2, or 9.6, an additional package installation is required.

The following describes the package that needs to be installed:

podman: manage container tool
podman-plugins: the CNI plugins used to run podman
dnsmasq: use to find local dnsname

```
【Default】
 sudo yum install -y podman

【Red hat 8.4 or 9.2 podman 4.0 ~ 4.4】
 sudo yum install -y podman podman-plugins dnsmasq
```

4.6.2 (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 command permission. Ensure you add your service account into the Wheel group.
We recommend that you create a service account before you start the installation.
The following command shows you how to add your service account into the Wheel group:
We assume that you are using the "**nightwatch**" account, and that the DFM Module is logged in with **a dedicated service account** and operates with the privileges of the account.

```
sudo usermod --append -G wheel <username>

Example)
 sudo usermod --append -G wheel nightwatch
```

Add wheel group

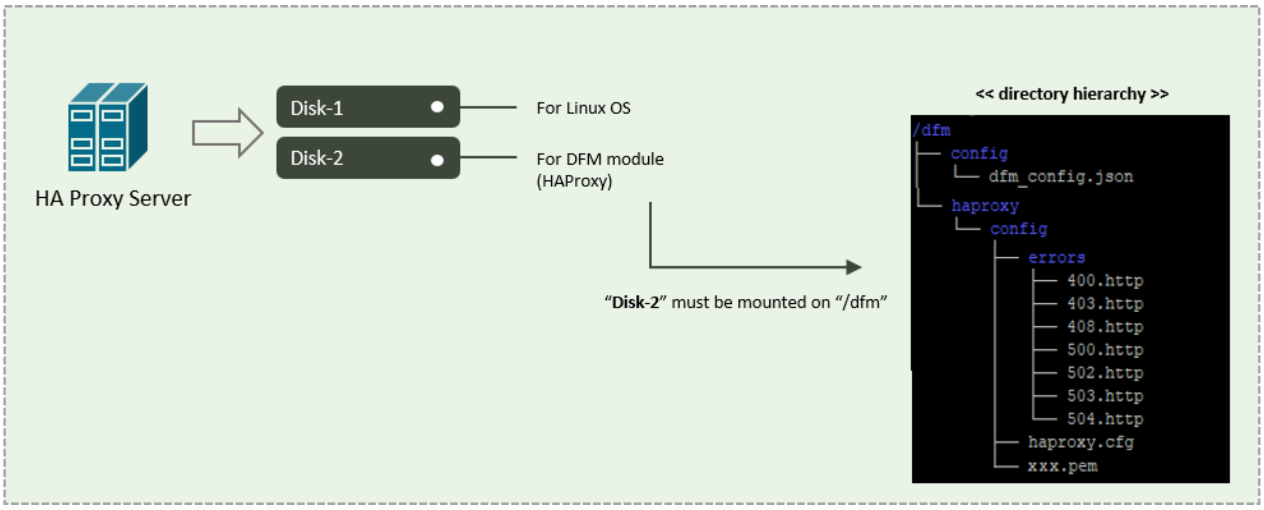
To connect using a created user:

```
ssh {your-user}@localhost [-p {port}]

Example)
 ssh nightwatch@localhost

or if you use port 9000
 ssh nightwatch@localhost -p 9000
```

4.6.3 (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.



```
/dfm
├── config
│   └── dfm_config.json
└── haproxy
    └── config
        ├── errors
        │   ├── 400.http
        │   ├── 403.http
        │   ├── 408.http
        │   ├── 500.http
        │   ├── 502.http
        │   ├── 503.http
        │   └── 504.http
        ├── haproxy.cfg
        └── xxx.pem
```

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

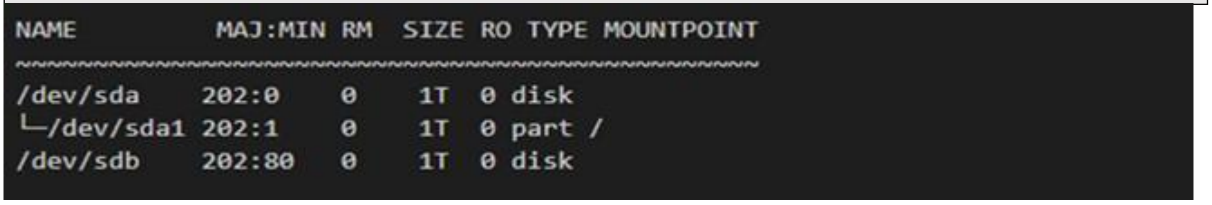
For example, we assume that two disks (“sda” and “sdb”) exist.

【CASE01】 Disk is Ready

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

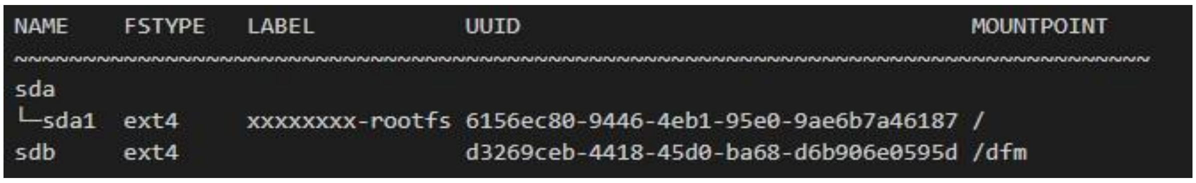
Now, let’s check the disk information:

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
```

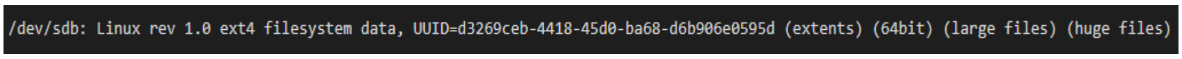
sudo lsblk -f



```
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



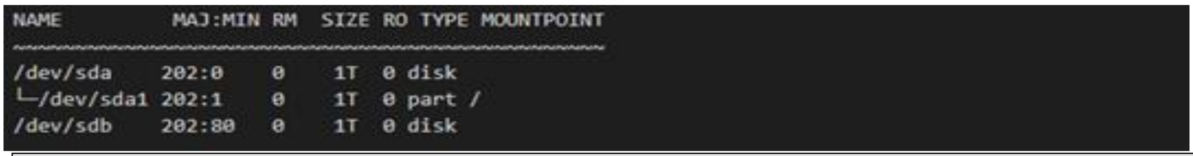
```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

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

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

```
sudo lsblk -p
```

Now, let’s check the disk information:

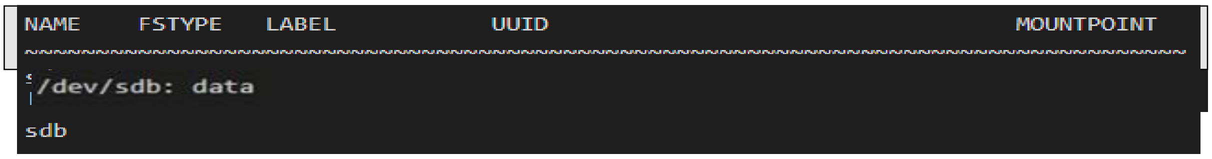


```
NAME          MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
/dev/sda     202:0    0    1T  0 disk
└/dev/sda1  202:1    0    1T  0 part /
/dev/sdb     202:80   0    1T  0 disk
```

```
sudo lsblk -f
```

⇨ “sdb” is NOT formatted

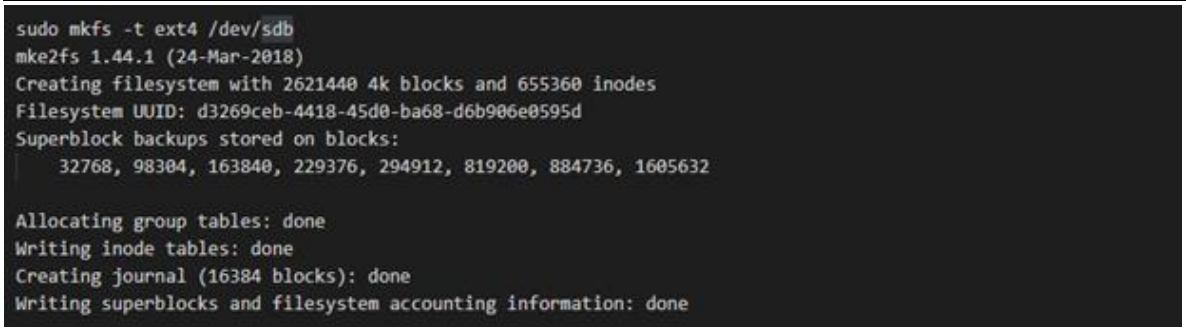
⇨ This means that the disk needs to be formatted



```
NAME      FSTYPE    LABEL           UUID                                MOUNTPOINT
/dev/sdb: data
sdb
```

5) **Format with ext4 file-system**

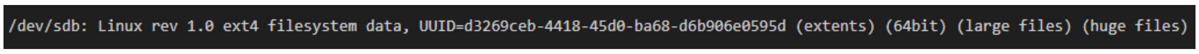
```
sudo file -s /dev/sdb
```



```
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
```



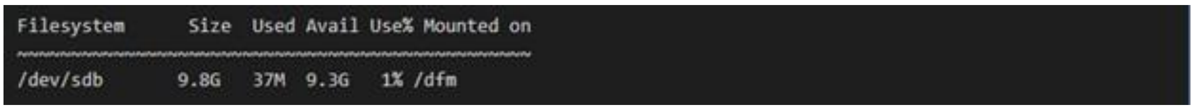
```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

7) **Mount “/dev/sdb” on /dfm**

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

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

8) **Verify**

```
df -h
```



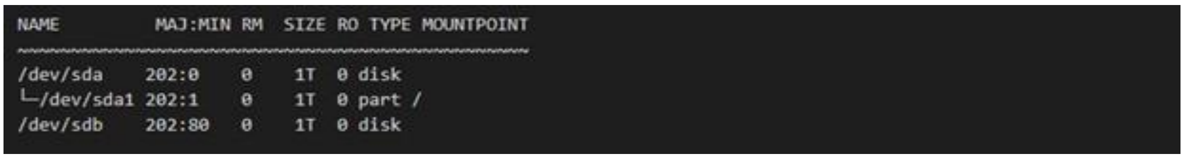
```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb        9.8G   37M  9.3G   1% /dfm
```

【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**.

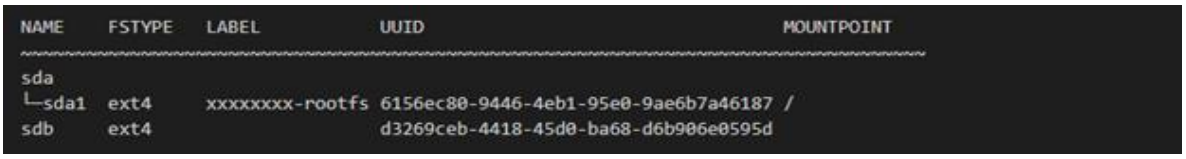
Now, let's check the disk information:

```
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
```

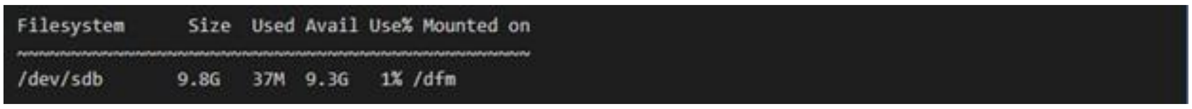
⇨ "sdb" " is formatted but Not yet mounted

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
```

4.6.4 Permanently mount the disk

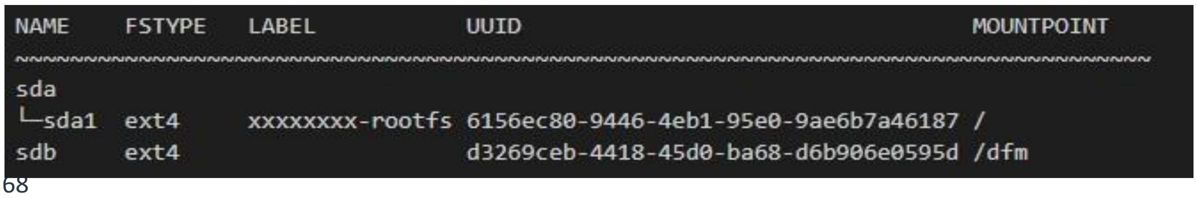
We recommend that the **customer's IT manager** sets the boot script so that **the dedicated disk** is auto-mounted when the server is booted.

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
```

4.6.5 (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
```

4.6.6 (STEP04) Install DFM Module Package

The DFM Module is delivered as a TAR archive file. 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) Image load**, and **2) Copying the config file** to the service directory.

- executable binary:

  ⇒ /tmp/sec-dfm_1.0.1.11/dfm/bin/dfm

- docker images:

```
    ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/haproxy-debian-{version}.tar
    ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/minio-RELEASE. {version}.tar
    ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/mysql-{version}.tar
    ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/dfm-console-{version}.tar
    ⇒ /tmp/sec-dfm_1.0.1.11/dfm/images/dfm-core-{version}.tar


 - haproxy config file:
    ⇒ /tmp/sec-dfm_1.0.1.11/dfm/ha/web-server/haproxy-config/haproxy.cfg

 - dfm config file:
    ⇒ /tmp/sec-dfm_1.0.1.11/dfm/ha/web-server/dfm_config.json
```

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

```
     tar -zxvf sec-dfm_{version}.tar.gz -C /tmp

    example)
     tar -zxvf  sec-dfm_1.0.1.11.tar.gz -C /tmp

    /tmp/sec-dfm_1.0.1.11/dfm/
    /tmp/sec-dfm_1.0.1.11/dfm/mysql-query/
    /tmp/sec-dfm_1.0.1.11/dfm/haproxy-config/
    ……………..
    /tmp/sec-dfm_1.0.1.11/dfm/bin/
    ……………..
```

Next, check if the necessary files exist:

```
1) check dfm file
ls -l /tmp/sec-dfm_1.0.1.11/dfm/bin/dfm


2) check images
ls /tmp/sec-dfm_1.0.1.11/dfm/images/ -l
total 971552
-rw-rw-r-- 1 dfm-console-1.0.1.11.tar
-rw-rw-r-- 1 dfm-core-1.0.1.11.tar
-rw-rw-r-- 1 haproxy-debian-2.2.33.tar
-rw-rw-r-- 1 minio-RELEASE.2022-04-30T22-23-53Z.tar
-rw-rw-r-- 1 mysql-8.0.36.tar


4) check haproxy config file : haproxy.cfg
ls /tmp/sec-dfm_1.0.1.11/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/sec-dfm_1.0.1.11/dfm/ha/web-server/config/dfm_config.json
/tmp/sec-dfm_1.0.1.11/dfm/ha/app-server/config/dfm_config.json
```

4.6.7 (STEP05) Load Container Image

Next, register the Container Images that were unpacked at "**/tmp/sec-dfm_1.0.1.11/dfm/images**". The loaded Container Images are used when the container is driven. The following shows how to load Container Image required for WEB server using Podman commands:

```
podman load -i /tmp/sec-dfm_1.0.1.11/dfm/images/haproxy-debian-{version}.tar
```

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

```
Example)
 podman images
REPOSITORY                  TAG                IMAGE ID        CREATED        SIZE
haproxytech/haproxy-debian  2.2.33             88bf690bd83f     6 days ago     99.7MB
```

4.6.8 (STEP06) Copy Configuration files

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

- copy executable binary:

```
// copy executable binary
  sudo cp /tmp/sec-dfm_1.0.1.11/dfm/bin/dfm /usr/bin/ or cp /tmp/sec-
  dfm_1.0.1.11/dfm/bin/dfm /usr/local/bin

// Set executable
sudo chmod 755 /usr/bin/dfm or sudo chmod 755 /usr/local/bin/dfm
```

- **copy haproxy config file**:

```
// copy configuration file
cp /tmp/sec-dfm_1.0.1.11/dfm/ha/web-server/haproxy-config/haproxy.cfg
/dfm/haproxy/config

// copy error files
cp -rf /tmp/sec-dfm_1.0.1.11/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/sec-dfm_1.0.1.11/dfm/ha/web-server/config/dfm_config.json /dfm/config

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

4.6.9 (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
```

4.6.10 (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 all minio server ip and cluster_minio_access_port set in the Firmware Storage(minio) server.
Set up to “http-request set-header Host **dfm-proxy:{cluster_minio_acces_port}**”

```
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 dfm-proxy:9000
```

```
#if minio server use ssl
#server dfm-minioproxy 192.168.0.7:9000 ssl verify none check
#otherwise
server dfm-minio-group-1 192.168.0.10:9000 check
 server dfm-minio-group-1 192.168.0.11:9000 check
 server dfm-minio-group-1 192.168.0.12:9000 check
 server dfm-minio-group-1 192.168.0.13:9000 check

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

4.6.11 (STEP09) Create Container Network

The DFM Module is a process executed on a container basis, creating the Podman network requiredfor communications among containers.

To create a network, use the following command: (sudo is required in root mode.)

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

4.6.12 (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: (sudo is required in root mode.)

```
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.

```
# If it is red hat 8.4 version, run health check
podman healthcheck run dfm-proxy
podman ps -a

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

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

4.7 Configure SSL

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

4.7.1 DB(MySQL) Server

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

4.7.2 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 **example-sec-fota.net**

· cert.pem

· chain.pem

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

⇨ cat cert.pem chain.pem > fullchain.pem

· privkey.pem

⇨ sudo -E bash -c 'cat fullchain.pem privkey.pem > example-sec-fota.net.pem'

‘**example-sec-fota.net.pem**’ is the unified certificate file

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

We assume that you are using the “**nightwatch**” account.
The certificate file is bound to “/usr/local/etc/haproxy/” in the haproxy container.
So only the certificate file (.pem) name needs to be changed, and not the path.

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

* To set up and use mutual TLS, see Appendix G. Set up mutual TLS.

- Restart the container (HA proxy) (sudo is required in root mode.)

```
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) (sudo is required in root mode.)

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

4.7.3 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)

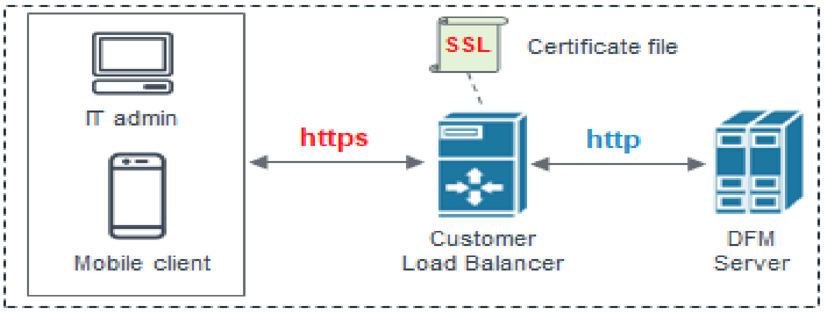


Fig 4-10 On Customer's Load Balancer (Proxy)

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

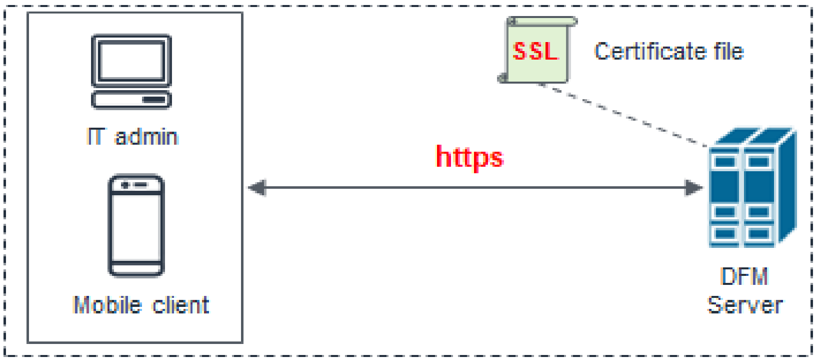


Fig 4-11 **On DFM Server**

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 **example-sec-fota.net**

  · cert.pem

  · chain.pem

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

  ⇨ cat cert.pem chain.pem > fullchain.pem

  · privkey.pem
  ⇨ sudo -E bash -c 'cat fullchain.pem privkey.pem > example-sec-fota.net.pem'
  ⇨ ‘**example-sec-fota.net.pem**’ is the unified certificate file
- Also you can make a pem file for devices. Copy the “chain.pem” file to create a new file named “efota.pem”
  ⇨ cp chain.pem efota.pem

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

The certificate file is bound to “/usr/local/etc/haproxy/” in the haproxy container.

So only the certificate file (.pem) name needs to be changed, and not the path.

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

* To set up and use mutual TLS, see Appendix G. Set up mutual TLS.

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

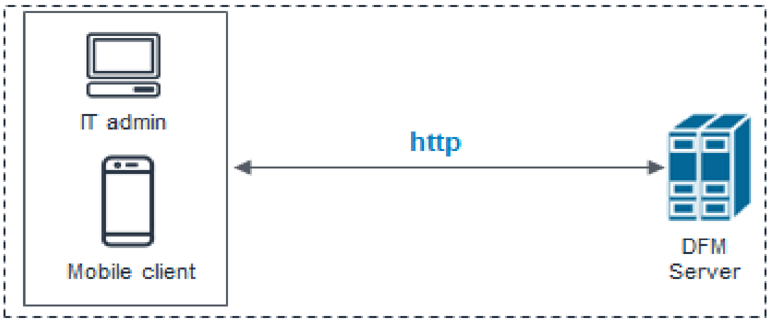


Fig 4-12 **On DFM Server**

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

4.8 How to check Server Operation Status

Finally, the installer has completed the installation of the on-premises service-based Podman, 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. (sudo is required in root mode.)

```
podman ps -a

Example)
1) MySQL Server (1 Container)
podman ps -a
  CONTAINER ID      ~     STATUS                       ~      NAMES
d882c61ba91c            Up 15 hours (healthy)              dfm-mysql

 2) Firmware Storage Server (1 Container)
 podman ps -a
CONTAINER ID          ~  STATUS                        ~  NAMES
af3949b8db98             Up 6 minutes (healthy)           dfm-minio

 3) DFM Core/Console Server (3 Containers)
 podman 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)
 podman 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.

We assume that you are using the "**nightwatch**" account.
If you do not enable linger on the account you added dfm — in this case, "nightwatch" — the account will drop the dfm containers and E-FOTA will stop working.
The command to enable linger is:

```
loginctl enable-linger <username>

Example)
loginctl enable-linger nightwatch
```

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)

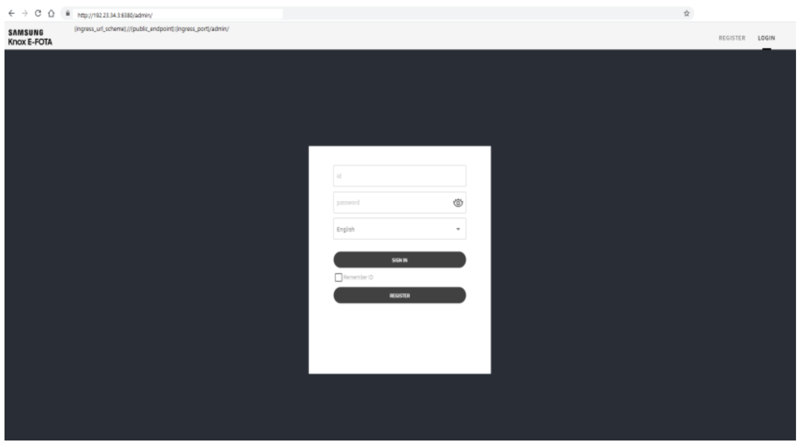

http://192.23.34.3:6380/admin/
{ingress_url_scheme}://{public_endpoint}:{ingress_port}/admin/
SAMSUNG
Knox E-FOTA
REGISTER
LOGIN
id
password
English
SIGN IN
Remember ID
REGISTER

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: (sudo is required in root mode.)

```
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
```

* To set up and use mutual TLS, see Appendix G. Set up mutual TLS.

【STEP03】 Restart proxy

The command to restart the proxy Server container is as follows: (sudo is required in root mode.)

```
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 its state is healthy.

```
# If it is red hat 8.4 version, run health check
podman healthcheck run dfm-proxy

podman ps -a

CONTAINER ID   IMAGE                                          STATUS                              NAMES
e80b80bdba55  localhost/haproxytech/haproxy-debian:2.2.33      Up 3 seconds ago (starting)          dfm-proxy

$ podman ps -a

CONTAINER ID   IMAGE                                          STATUS                              NAMES
e80b80bdba55  localhost/haproxytech/haproxy-debian:2.2.33      Up 52 seconds ago (healthy)          dfm-proxy
```

5.5 DB Backup & Restore

This section describes how to back up and restore the DB to ensure service continuity.
Important! backup & restore must be performed on the primary server

5.5.1. Back up a MySQL Server Instance

I. *To BACK UP a MySQL Server instance running in a Podman container using MySQL Shell Backup with Podman:*

1. On the same host where the MySQL Server container is running, start another container with an image of MySQL to perform a backup with the MySQL Shell Backup. 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 (*/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 podman image currently installed is localhost/mysql:8.0.36.

We assume that you are using the "**nightwatch**" account. (sudo is required in root mode.)

2. Create account for backup

```
podman exec -i dfm-mysql-1 mysql -uroot -p1q2w3e4r -e "ALTER TABLE dfm_console_db.databasechangelog
ADD PRIMARY KEY (ID);"
podman exec -i dfm-mysql-1 mysql -uroot -p1q2w3e4r -e "CREATE USER 'backup'@'%' IDENTIFIED BY
'1q2w3e4r';"
podman exec -i dfm-mysql-1 mysql -uroot -p1q2w3e4r -e "GRANT SELECT, SHOW VIEW, TRIGGER, EVENT, LOCK
TABLES, RELOAD, PROCESS, REPLICATION CLIENT ON *.* TO 'backup'@'%';"
```

【Basic Command】

```
podman run --rm --network host -v =/path-on-host-machine/datadir/:/backups localhost/mysql:8.0.36 mysqlsh --no-
defaults -u{user} -p{password} -P{Port number} -e "util.dumpInstance('/backups', {threads: 16, compression: 'zstd'})"
```

【Example】

```
podman run --rm --network host -v /dfm/db-server-1/mysql/backups:/backups localhost/mysql:8.0.36
mysqlsh --no-defaults -ubackup -p1q2w3e4r –P33061 -e "util.dumpInstance('/backups', {threads: 16,
compression: 'zstd'})"
```

```
[efotadm@xx mysql]$ podman run --rm --network host -v /dfm/db-server-1/mysql/backups:/backups
localhost/mysql:8.0.36 mysqlsh --no-defaults -ubackup -p1q2w3e4r -P33061 -e "util.dumpInstance('/backups', {threads:
16, compression: 'zstd'})"
Cannot set LC_ALL to locale en_US.UTF-8: No such file or directory
WARNING: Using a password on the command line interface can be insecure.
Initializing...
NOTE: Backup lock is not available to the account 'backup'@'%' and DDL changes will not be blocked. The dump may fail
with an error if schema changes are made while dumping.
Acquiring global read lock
Global read lock acquired
Initializing - done
Gathering information...
2 out of 6 schemas will be dumped and within them 75 tables, 0 views.
4 out of 7 users will be dumped.
Gathering information - done
All transactions have been started
Global read lock has been released
Writing global DDL files
Writing users DDL
Writing schema metadata...
Writing DDL...
Writing table metadata...
Running data dump using 16 threads.
Dumping data...
NOTE: Table statistics not available for `dfm_console_db`.`tb_system`, chunking operation may be not optimal. Please
consider running 'ANALYZE TABLE `dfm_console_db`.`tb_system`;' first.
NOTE: Table statistics not available for `dfm_console_db`.`databasechangeloglock`, chunking operation may be not
optimal. Please consider running 'ANALYZE TABLE `dfm_console_db`.`databasechangeloglock`;' first.
NOTE: Table statistics not available for `dfm_console_db`.`tb_group`, chunking operation may be not optimal. Please
consider running 'ANALYZE TABLE `dfm_console_db`.`tb_group`;' first.
NOTE: Table statistics not available for `dfm_console_db`.`tb_content_resource`, chunking operation may be not optimal.
Please consider running 'ANALYZE TABLE `dfm_console_db`.`tb_content_resource`;' first.
NOTE: Table statistics not available for `dfm_core_db`.`tczs_fw_scnr`, chunking operation may be not optimal. Please
consider running 'ANALYZE TABLE `dfm_core_db`.`tczs_fw_scnr`;' first.
Writing schema metadata - done
Writing DDL - done
Writing table metadata - done
Starting data dump
Dumping data - done
Dump duration: 00:00:00s
Total duration: 00:00:00s
Schemas dumped: 2
Tables dumped: 75
Uncompressed data size: 500.34 KB
Compressed data size: 59.08 KB
Compression ratio: 8.5
Rows written: 1458
Bytes written: 59.08 KB
Average uncompressed throughput: 500.34 KB/s
Average compressed throughput: 59.08 KB/s
```

3. 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 backup files is created and can be found in the host directory mounted in the last step for storing backups:

```
ls /dfm/db-server-1/mysql/backups/
```

5.5.2. Restore a MySQL Server Instance

II. ***To RESTORE a MySQL Server instance in a Podman container using MySQL Shell with Podman:***

1. Stop the every MySQL Server container, which also stops the MySQL Server runninginside:mounted in the last step for storing backups: (sudo is required in root mode.)

```
podman stop dfm-mysql-1
podman stop dfm-mysql-2
…
```

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

```
sudo rm -rf /dfm/db-server-1/mysql/data/*
sudo rm -rf /dfm/db-server-2/mysql/data/*
…
```

3. Restart the every server container (sudo is required in root mode.)

```
dfm cluster start dfm-mysql -f /dfm/db-server-1/config/dfm_config.json -n dfm-mysql-1
dfm cluster start dfm-mysql -f /dfm/db-server-2/config/dfm_config.json -n dfm-mysql-2
…
```

4. Set DB root password(we assume that “password” is “**1q2w3e4”**) and and set the ‘local_infile’ for restore

```
podman exec -i dfm-mysql-1 mysql -uroot -e "ALTER USER 'root'@'localhost' IDENTIFIED BY '1q2w3e4r';"
podman exec -i dfm-mysql-1 mysql -uroot -p1q2w3e4r -e "set global local_infile = 1;"
```

5. Create account for restore

```
podman exec -i dfm-mysql-1 mysql -uroot -p1q2w3e4r -e "CREATE USER 'restore'@'%' IDENTIFIED BY
'1q2w3e4r';"
podman exec -i dfm-mysql-1 mysql -uroot -p1q2w3e4r -e "GRANT CREATE, DROP, ALTER, INDEX, INSERT,
UPDATE, DELETE, SELECT, TRIGGER, EVENT, REFERENCES ON *.* TO 'restore'@'%';"
```

6. Set Group replication

【Primary server】

Proceed to the primary server for group replication.

```
podman exec -it dfm-mysql-1 mysql -uroot -p1q2w3e4r
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> SET SQL_LOG_BIN=0;
Query OK, 0 rows affected (0.00 sec)
mysql> CREATE USER group_user@'%' IDENTIFIED BY '1q2w3e4r' REQUIRE SSL;
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT REPLICATION SLAVE ON *.* TO group_user@'%';
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT CONNECTION_ADMIN ON *.* TO group_user@'%';
Query OK, 0 rows affected (0.00 sec)
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
mysql> SET SQL_LOG_BIN=1;
Query OK, 0 rows affected (0.00 sec)

mysql> CHANGE MASTER TO MASTER_USER='group_user', MASTER_PASSWORD='1q2w3e4r'\
FOR CHANNEL 'group_replication_recovery';
Query OK, 0 rows affected (0.02 sec)

mysql> SET GLOBAL group_replication_bootstrap_group=ON;
Query OK, 0 rows affected (0.00 sec)
mysql> START GROUP_REPLICATION;
Query OK, 0 rows affected (4.56 sec)
mysql> SET GLOBAL group_replication_bootstrap_group=OFF;
Query OK, 0 rows affected (0.00 sec)
```

【Secondary server】

Proceed to the secondary server for group replication.

```
podman exec -it dfm-mysql-2 mysql -uroot -p1q2w3e4r
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> SET SQL_LOG_BIN=0;
Query OK, 0 rows affected (0.00 sec)
mysql> CREATE USER group_user@'%' IDENTIFIED BY '1q2w3e4r' REQUIRE SSL;
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT REPLICATION SLAVE ON *.* TO group_user@'%';
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT CONNECTION_ADMIN ON *.* TO group_user@'%';
Query OK, 0 rows affected (0.00 sec)
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
mysql> SET SQL_LOG_BIN=1;
Query OK, 0 rows affected (0.00 sec)

mysql> CHANGE MASTER TO MASTER_USER='group_user', MASTER_PASSWORD='1q2w3e4r'\
FOR CHANNEL 'group_replication_recovery';
Query OK, 0 rows affected (0.02 sec)

mysql> START GROUP_REPLICATION;
Query OK, 0 rows affected (4.56 sec)
```

【validation】

```
mysql> SELECT MEMBER_HOST, MEMBER_PORT, MEMBER_STATE, MEMBER_ROLE FROM
performance_schema.replication_group_members;

 +---------------------+---------------------+----------------------+---------------------+
 | MEMBER_HOST| MEMBER_PORT | MEMBER_STATE | MEMBER_ROLE |
 +---------------------+---------------------+----------------------+---------------------+
 |     192.168.0.4  |              33061 |         ONLINE       | PRIMARY            |
 |     192.168.0.5  |              33061 |         ONLINE       | SECONDARY       |
 |     192.168.0.6  |              33061 |         ONLINE       | SECONDARY       |
 +---------------------+---------------------+----------------------+---------------------+
```

7. Start a container with an image of MySQL to perform the restore with the MySQL Shell. Bind-mount the server's data directory and the storage folder for the backups, like what we did when we backed up the server: (sudo is required in root mode.)

【Basic Command】

```
podman run --rm --network host -v /dfm/mysql/backups:/backups localhost/mysql:8.0.36 mysqlsh --no-defaults -u{user} -p{password} –P{Port number} -e "util.loadDump('/backups', {threads: 16})"
```

【Example】

```
podman run --rm --network host -v /dfm/db-server-1/mysql/backups:/backups localhost/mysql:8.0.36 mysqlsh --no-defaults -urestore -p1q2w3e4r -P33061 -e "util.loadDump('/backups', {threads: 16})"
[efotadm@xx ~]$ podman run --rm --network host -v /dfm/db-server-1/mysql/backups:/backups
localhost/mysql:8.0.36 mysqlsh --no-defaults -urestore -p1q2w3e4r -P33061 -e
"util.loadDump('/backups', {threads: 16})"
Cannot set LC_ALL to locale en_US.UTF-8: No such file or directory
WARNING: Using a password on the command line interface can be insecure.
Loading DDL and Data from '/backups' using 16 threads.
Opening dump...
Target is MySQL 8.0.36. Dump was produced from MySQL 8.0.36
Scanning metadata...
Scanning metadata - done
Checking for pre-existing objects...
Executing common preamble SQL
Executing DDL...
Executing DDL - done
Executing view DDL...
Executing view DDL - done
Loading data...
Starting data load
Loading data - done
```

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

8. Off the 'local_infile' option

```
podman exec -it dfm-mysql-1 mysql -uroot -p1q2w3e4r -e "set global local_infile = 0;"
```

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 (sudo is required in root mode.)

```
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 (sudo is required in root mode.)

```
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.

```
# If it is red hat 8.4 version, run health check
podman healthcheck run dfm-console

podman ps -a

CONTAINER ID   IMAGE                                    STATUS                          NAMES
8a3e2f4452e8   localhost/dfm-console:1.0.1.11            Up 3 seconds ago (starting)      dfm-console

$ podman ps -a

CONTAINER ID   IMAGE                                    STATUS                          NAMES
8a3e2f4452e8   localhost/dfm-console:1.0.1.11            Up 52 seconds ago (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: (sudo is required in root mode.)

```
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: (sudo is required in root mode.)

```
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 its state is healthy.

```
# If it is red hat 8.4 version, run health check
podman healthcheck run dfm-core


podman ps -a


CONTAINER ID  IMAGE                              STATUS                      NAMES
120be188f49f   localhost/dfm-core:1.0.1.11        Up 3 seconds ago (starting)   dfm-core


$ podman ps -a


CONTAINER ID  IMAGE                              STATUS                      NAMES
120be188f49f  localhost/dfm-core:1.0.1.11         Up 52 seconds ago (healthy)   dfm-core
```

5.7 Configurable device polling interval and postpone waiting time

The installer can change the default value of the device polling interval and postpone the waiting time. (default polling_interval_register=86400, default polling_interval_default=86400, default_waiting_time=30)

【STEP01】Stop DFM Core

The command to stop the DFM Core Server container is: (sudo is required in root mode.)

```
dfm cluster terminate dfm-core
```

【STEP02】Set up the device polling interval and postpone waiting time

The polling_interval_register can only be an integer.
The polling_interval_default is allowed from 60 to 86400.(Values lower than 60 will be set to 60)
The postpone waiting time limit is allowed from 1 to 7200 .

```
dfm config set polling_interval_register=86400
dfm config set polling_interval_default =86400
dfm config set default_waiting_time=30
```

【STEP03】Check the device polling interval and postpone waiting time

```
dfm config get polling_interval_register
86400
dfm config get polling_interval_default
86400
dfm config get default_waiting_time
30
```

【STEP04】Restart DFM Core

The command to restart the DFM Core Server container is: (sudo is required in root mode.)

```
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 its state is healthy.

```
# If it is red hat 8.4 version, run health check
podman healthcheck run dfm-core

podman ps -a

CONTAINER ID  IMAGE                              STATUS                      NAMES
120be188f49f  localhost/dfm-core:1.0.1.11         Up 3 seconds ago (starting)   dfm-core

$ podman ps -a

CONTAINER ID  IMAGE                              STATUS                      NAMES
120be188f49f  localhost/dfm-core:1.0.1.11         Up 52 seconds ago (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**.

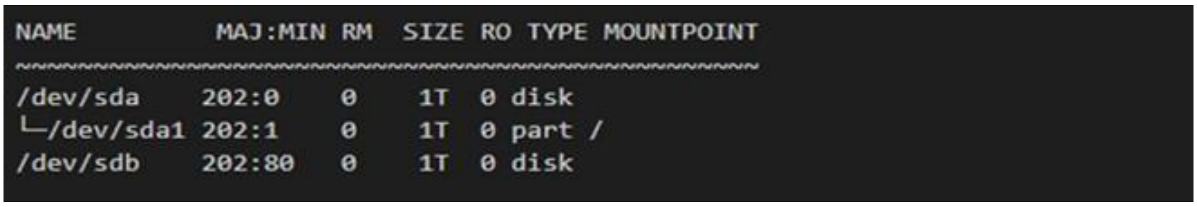
For example, we assume that two disks ("sda" and "sdb") exist.

【CASE01】 Disk is Ready

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

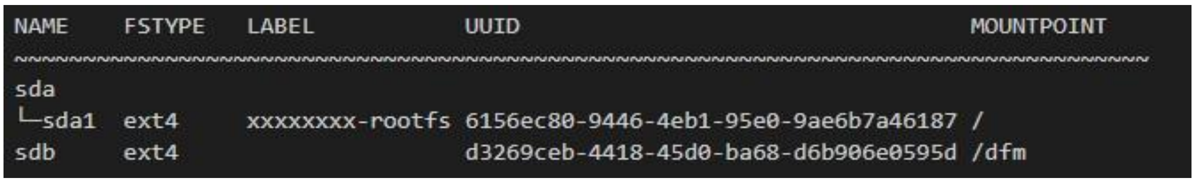
Now, let's check the disk information:

```
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
```

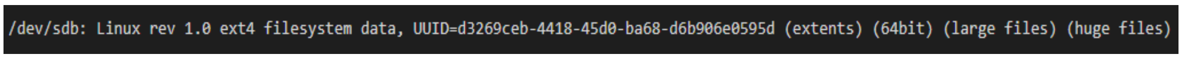
```
sudo lsblk –f
```



```
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
```



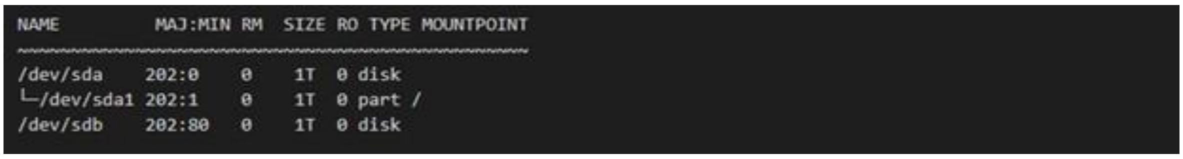
```
/dev/sdb: Linux rev 1.0 ext4 filesystem data, UUID=d3269ceb-4418-45d0-ba68-d6b906e0595d (extents) (64bit) (large files) (huge files)
```

【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**.

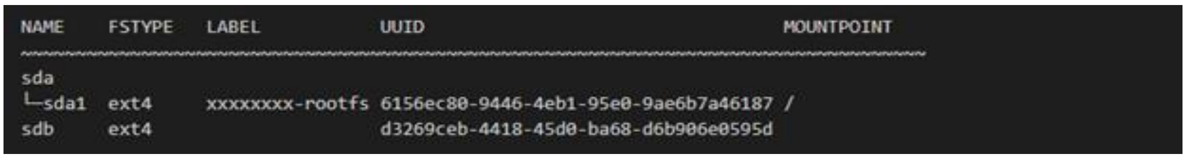
Now, let's check the disk information.

```
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
```

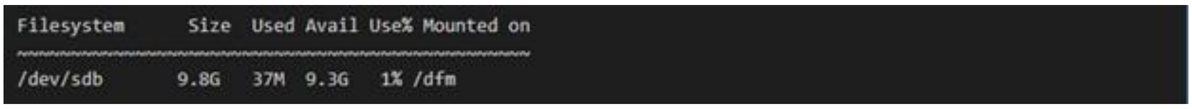
⇨ "sdb" " is formatted but not yet mounted

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 Database Server (MySQL)

After the system is rebooted, restart MySQL using the following command: (sudo is required in root mode.)

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

To rejoin group replication after a restart:

6.3.1 Delete and create a data folder.
6.3.2 Start MySQL.
6.3.3 Start the group replication.

【Delete and create a data folder】

```
# delete folder
sudo rm -rf /dfm/mysql/data

# make directory
mkdir -p /dfm/mysql/data
```

【Start MySQL】

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

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

```
# If it is red hat 8.4 version, run health check
podman healthcheck run dfm-mysql

podman ps -a

CONTAINER ID  IMAGE                                      STATUS                        NAMES
2cd1bae13406  localhost/mysql/enterprise-server:8.0      Up 3 seconds ago (starting)   dfm-mysql

$ podman ps -a

CONTAINER ID  IMAGE                                      STATUS                        NAMES
2cd1bae13406  localhost/mysql/enterprise-server:8.0      Up 52 seconds ago (healthy)   dfm-mysql
```

【Start group replication】

```
podman exec -i dfm-mysql mysql -uroot -p1q2w3e4r
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> SET SQL_LOG_BIN=0;
Query OK, 0 rows affected (0.00 sec)
mysql> CREATE USER group_user@'%' IDENTIFIED BY '1q2w3e4r' REQUIRE SSL;
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT REPLICATION SLAVE ON *.* TO group_user@'%';
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT CONNECTION_ADMIN ON *.* TO group_user@'%';
Query OK, 0 rows affected (0.00 sec)
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
mysql> SET SQL_LOG_BIN=1;
Query OK, 0 rows affected (0.00 sec)

mysql> CHANGE MASTER TO MASTER_USER='group_user', MASTER_PASSWORD='1q2w3e4r'\
FOR CHANNEL 'group_replication_recovery';
Query OK, 0 rows affected (0.02 sec)

mysql> START GROUP_REPLICATION;
Query OK, 0 rows affected (4.56 sec)
```

【validation】

```
mysql> SELECT MEMBER_HOST, MEMBER_PORT, MEMBER_STATE, MEMBER_ROLE FROM
performance_schema.replication_group_members;

 +---------------------+---------------------+-----------------------+---------------------+
 | MEMBER_HOST| MEMBER_PORT | MEMBER_STATE | MEMBER_ROLE |
 +---------------------+---------------------+-----------------------+---------------------+
 |     192.168.0.4  |            33061 |        ONLINE      | PRIMARY         |
 |     192.168.0.5  |            33061 |        ONLINE      | SECONDARY      |
 |     192.168.0.6  |            33061 |        ONLINE      | SECONDARY      |
 +---------------------+---------------------+-----------------------+---------------------+
```

6.4(STEP04) Start-up Firmware Storage Server

After the system is rebooted, restart Minio. The command to run Minio server container is as follows: (sudo is required in root mode.)

After the restart, proceed in the order below to synchronise between minio server.

```
# Delete and create data folder.
rm -rf /dfm/minio/data
mkdir -p /dfm/minio/data

#restart minio service
dfm cluster restart dfm-minio

#load to minio mc images for syncronise between minio server.
podman load -i /tmp/dfm/docker-images/minio_mc.tar

#start minio syncronise
dfm cluster sync dfm-minio

The command is running
Added `dfm` successfully.
[Green  ->  Green] ** system:disk-format **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/config/config.json **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/config/iam/format.json **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/buckets/.bloomcycle.bin **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/buckets/.usage-cache.bin **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/buckets/.usage.json **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/buckets/dfm-agent-storage/.metadata.bin **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/buckets/dfm-agent-storage/.usage-cache.bin **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/buckets/dfm-fw-storage/.metadata.bin **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/buckets/dfm-fw-storage/.usage-cache.bin **
[Green  ->  Green] dfm-fw-storage/
Healed: 0/0 objects; 0 B in 1s
[Green  ->  Green] ** system:disk-format **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/config/config.json **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/config/iam/format.json **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/buckets/.bloomcycle.bin **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/buckets/.usage-cache.bin **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/buckets/.usage.json **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/buckets/dfm-agent-storage/.metadata.bin **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/buckets/dfm-agent-storage/.usage-cache.bin **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/buckets/dfm-fw-storage/.metadata.bin **
[Green  ->  Green] ** system:bucket-metadata:.minio.sys/buckets/dfm-fw-storage/.usage-cache.bin **
[Green  ->  Green] dfm-agent-storage/
Healed: 0/0 objects; 0 B in 1s
```

【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.

```
# If it is red hat 8.4 version, run health check
podman healthcheck run dfm-minio


podman ps -a

CONTAINER ID  IMAGE                                          STATUS                              NAMES
c8f8232a8ca1   localhost/minio/minio:RELEASE.2020-06-01T17-28-03Z    Up 3 seconds ago (starting)     dfm-minio
```

```
$ podman ps -a

CONTAINER ID  IMAGE                                          STATUS                       NAMES
c8f8232a8ca1  localhost/minio/minio:RELEASE.2020-06-01T17-28-03Z  Up 52 seconds ago (healthy)  dfm-minio
```

6.5(STEP05) Start-up DFM Core Server

After the system is rebooted, restart DFM Core. The command to run the core server container is as follows: (sudo is required in root mode.)

```
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.

```
# If it is red hat 8.4 version, run health check
podman healthcheck run dfm-core

podman ps -a

CONTAINER ID     IMAGE                          STATUS                          NAMES
120be188f49f     localhost/dfm-core:1.0.1.11    Up 3 seconds ago (starting)     dfm-core

podman ps -a

CONTAINER ID     IMAGE                          STATUS                          NAMES
120be188f49f     localhost/dfm-core:1.0.1.11    Up 3 seconds ago (healthy)      dfm-core
```

6.6(STEP06) 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: (sudo is required in root mode.)

```
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.

```
# If it is red hat 8.4 version, run health check
podman healthcheck run dfm-console

podman ps -a

CONTAINER ID     IMAGE                  STATUS                          NAMES
8a3e2f4452e8     localhost/dfm-         Up 3 seconds ago (starting)     dfm-console
                 console:1.0.1.11

podman ps -a

CONTAINER ID     IMAGE                  STATUS                          NAMES
8a3e2f4452e8     localhost/dfm-         Up 3 seconds ago (healthy)      dfm-console
```

6.7 (STEP07) Start-up HAProxy Server

After the system is rebooted, restart HAProxy. The command to run the HAProxy server container is as follows: (sudo is required in root mode.)

```
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.

```
# If it is red hat 8.4 version, run health check
podman healthcheck run dfm-proxy

podman ps -a

CONTAINER ID   IMAGE                                         STATUS                          NAMES
e80b80bdba55   localhost/haproxytech/haproxy-debian:2.2.33   Up 3 seconds ago (starting)     dfm-proxy

podman ps -a

CONTAINER ID   IMAGE                                         STATUS                          NAMES
e80b80bdba55   localhost/haproxytech/haproxy-debian:2.2.33   Up 3 seconds ago (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.

7.1Container Image Update

If there is an updated DFM Module, it is released as a container 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:

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

It should be updated as per the following steps. (sudo is required in root mode.)

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

```
dfm cluster terminate dfm-core
dfm cluster terminate dfm-console
dfm cluster terminate dfm-mysql
```

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

```
podman load -i mysql-enterprise-server-xx.xx.xx.tar
```

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

```
dfm config set mysql_img_rep=localhost/mysql/enterprise-server
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

- 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.

```
podman 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: (sudo is required in root mode.)

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

【STEP01】Stop the MinIO server.

```
dfm cluster terminate dfm-minio
```

【STEP02】Load the released Podman Image.

```
podman load -i minio-xx.xx.xx.tar
```

【STEP03】Change the repository and tag's configuration

```
dfm config set minio_img_rep=localhost/minio/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.

```
podman 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: (sudo is required in root mode.)

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

【STEP01】Stop the running core server.

```
dfm cluster terminate dfm-core
```

【STEP02】Load the released Podman image.

```
podman load -i 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.

```
podman 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: (sudo is required in root mode.)

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

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

```
dfm cluster terminate dfm-console
```

【STEP02】Load the released Podman image.

```
podman load -i 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 Server

- Admin Console Server

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

【Validation】

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

```
podman 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: (sudo is required in root mode.)

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

【STEP01】 Stop the running haproxy server.

```
dfm cluster terminate dfm-proxy
```

【STEP02】 Load the released Podman image.

```
podman load -i haproxy-debian-xx.xx.xx.tar
```

【STEP03】 Change the repository and tag's configuration

```
dfm config set haproxy_img_rep=localhost/haproxytech/haproxy-debian
dfm config set haproxy_img_tag=xx.xx.xx
```

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

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

【STEP05】 Start-up 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.

```
podman 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.

8.1Terminate Services

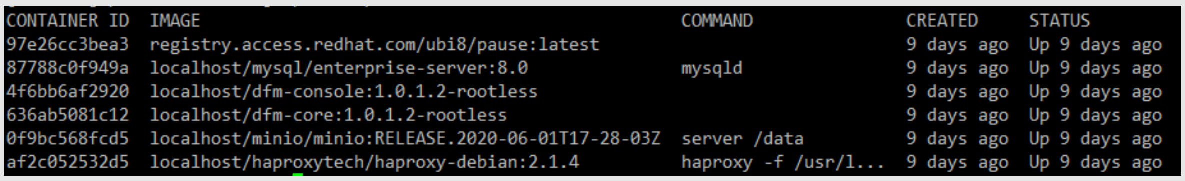
If there are active services, terminate them. (sudo is required in root mode.)

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

```
podman ps –a
```

Example)

```
podman ps –a
```



```
CONTAINER ID  IMAGE                                          COMMAND              CREATED     STATUS
97e26cc3bea3  registry.access.redhat.com/ubi8/pause:latest                        9 days ago  Up 9 days ago
87788c0f949a  localhost/mysql/enterprise-server:8.0          mysqld               9 days ago  Up 9 days ago
4f6bb6af2920  localhost/dfm-console:1.0.1.2-rootless                              9 days ago  Up 9 days ago
636ab5081c12  localhost/dfm-core:1.0.1.2-rootless                                 9 days ago  Up 9 days ago
0f9bc568fcd5  localhost/minio/minio:RELEASE.2020-06-01T17-28-03Z  server /data    9 days ago  Up 9 days ago
af2c052532d5  localhost/haproxytech/haproxy-debian:2.1.4     haproxy -f /usr/l... 9 days ago  Up 9 days ago
```

1. DFM Database (MySQL)

   Stop the server with the following command:

   ```
   dfm cluster terminate dfm-mysql
   ```

2. DFM Firmware Storage (MinIO)

   Stop the server with the following command:

   ```
   dfm cluster terminate dfm-minio
   ```

3. DFM Core Server

   Stop the server with the following command:

   ```
   dfm cluster terminate dfm-core
   ```

4. DFM Admin Console Server

   Stop the server with the following command:

   ```
   dfm cluster terminate dfm-console
   ```

5. DFM HAProxy Server

   Stop the server with the following command:

   ```
   dfm cluster terminate dfm-proxy
   ```

6. Check if all services are removed.

   Check with the following

   ```
   podman ps -a
   ```

command:

7. Stop Background App and check

Stop background app with the following command:

```
sudo systemctl stop efota-license.service
```

Check with the following command:

```
sudo systemctl status efota-license.service
  Loaded: loaded (/etc/system/system/efota-license.service; enabled; vendor preset: enabled)
  Active: inactive (dead) since Tue 2024-XX-XX 06:39:10 UTC; 7s ago
```

8.2 Remove Service directory

Remove old data using the following:

Remove all directory in /dfm

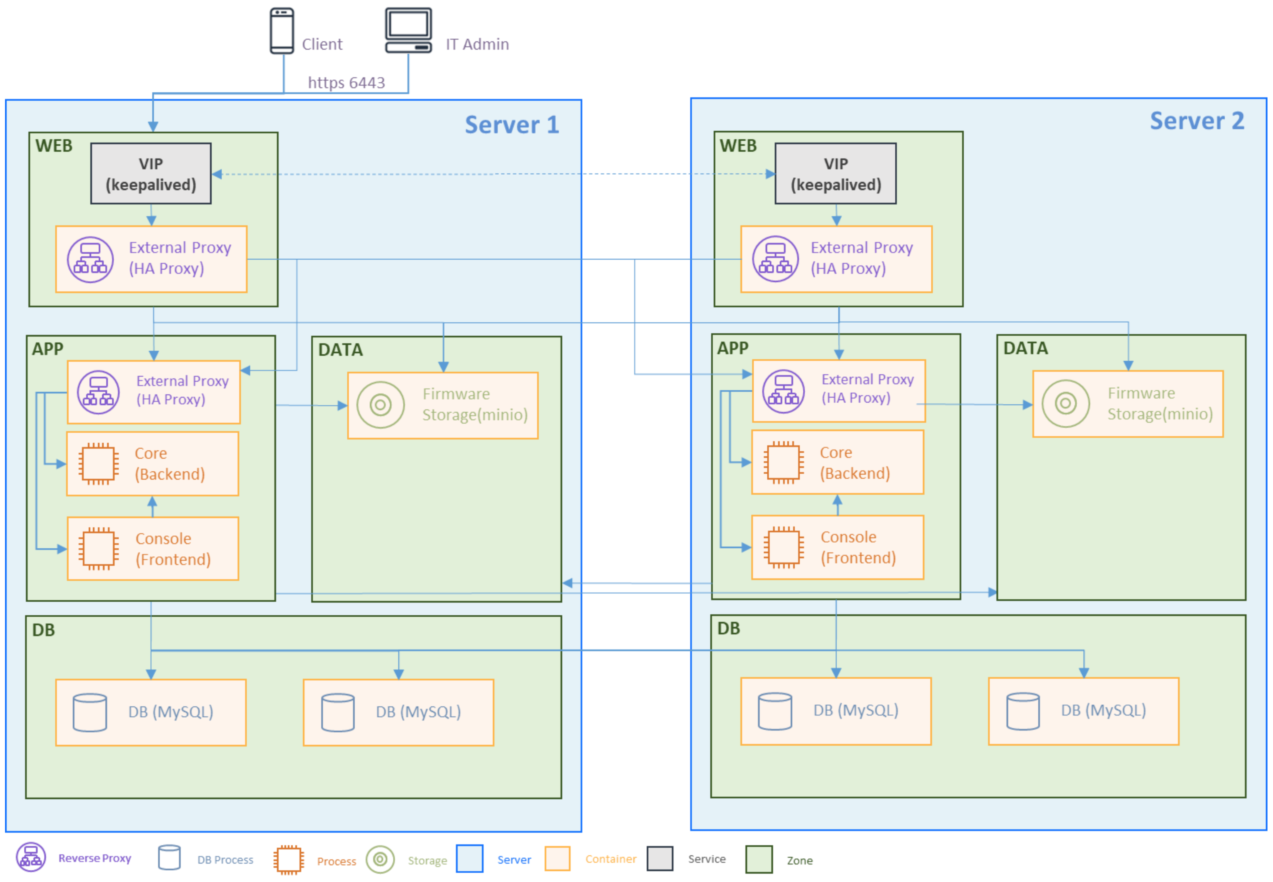
```
cd /dfm
sudo rm -rf *
```

PART VI: Install Case Scenario

This part describes a scenario where you are installing on two servers.

9 How to install 2 servers

The full diagram is shown below.



Initial requests are received via keepliaved's VIP (Virtual IP).
It then passes the request from the HAProxy in the web zone to the HAProxy in the app zone on server 1 and server 2 via load balancing (L7).
The minio in the data zone should have four independent storage spaces under it. In this scenario, we have four folders with independent storage.
The MySQL containers in the db zone are serviced by 2 per server. The minimum requirement for group replication is 3, so we go with an even number. Since it is in single-primary mode, write/read is handled by one server.

The procedure below describes what happens after the DFM package is installed.
It is assumed that
the IP of server 1 is 192.168.0.10 and
the IP of server 2 is 192.168.0.11.

9.1 Create Service Directories

9.1.1. Web zone

```
mkdir -p /dfm/web-server/config
mkdir -p /dfm/web-server/haproxy/config
mkdir -p /dfm/web-server/haproxy/errors
```

9.1.2. App zone

```
mkdir -p /dfm/app-server/config
mkdir -p /dfm/app-server/haproxy/config
mkdir -p /dfm/app-server/haproxy/errors
mkdir -p /dfm/app-serer/core/logs
mkdir -p /dfm/app-server/console/logs
```

9.1.3. Data zone

```
mkdir -p /dfm/data-server/config
mkdir -p /dfm/data-server/minio/config
mkdir -p /dfm/data-server/minio/data
mkdir -p /dfm/data-server/minio/data2
mkdir -p /dfm/data-server/minio/data3
mkdir -p /dfm/data-server/minio/data4
```

9.1.4. DB zone

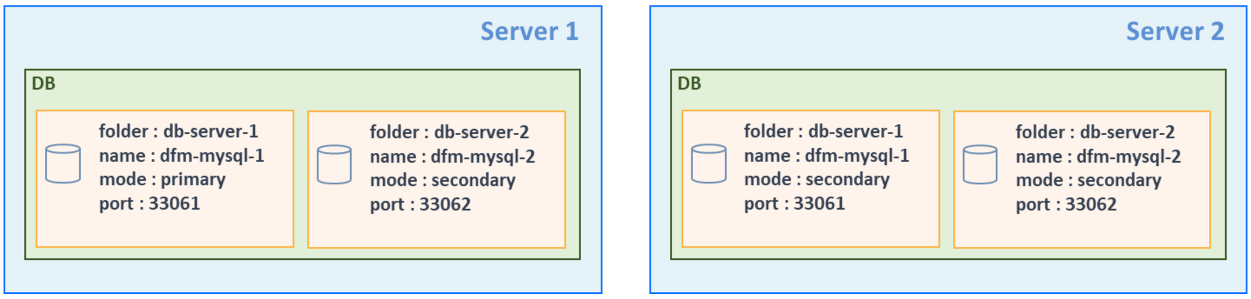
Since it’s running in a different container, we’ll create two different folders to make it work.

```
mkdir -p /dfm/db-server-1/config
mkdir -p /dfm/db-server-1/mysql/config
mkdir -p /dfm/db-server-1/mysql/data

mkdir -p /dfm/db-server-2/config
mkdir -p /dfm/db-server-2/mysql/config
mkdir -p /dfm/db-server-2/mysql/data
```

9.2 Configurations

9.2.1. DB zone

Describes the necessary settings for file copying and group replication.
The image below shows db-server-1 (folder name), dfm-mysql-1 (container name), primary (mode), and 33061 (service port).
It is located in the db-server-1 folder on server 1 under th container service name dfm-mysql-1.



【STEP01】Copy required files.

```
# copy to dfm-mysql-1 files
cp /tmp/dfm/ha/db-server/config/dfm_config.json /dfm/db-server-1/config/
cp /tmp/dfm/ha/db-server/mysql/config/my.cnf /dfm/db-server-1/mysql/config/

#copy to dfm-mysql-2 files
cp /tmp/dfm/ha/db-server/config/dfm_config.json /dfm/db-server-2/config/
cp /tmp/dfm/ha/db-server/mysql/config/my.cnf /dfm/db-server-2/mysql/config/
```

【STEP02】Configure my.cnf file. (dfm-mysql-1 on server 1)

The loose-group_replication_group_name setting refers to the generation of the UUID.

```
# config

vi /dfm/db-server-1/mysql/config/my.cnf

  [client]
  default-character-set=utf8mb4

  [mysql]
  default-character-set=utf8mb4

  [mysqld]
  user=mysql
  default-time-zone='+00:00'
  event_scheduler = ON
  general_log = 0
```

```
slow-query-log = 1
long_query_time = 4
lower_case_table_names = 1
collation-server = utf8mb4_unicode_ci
init-connect='SET NAMES utf8mb4'
character-set-server = utf8mb4
group_concat_max_len = 4096

port=33061
mysqlx_port=33071

bind-address="192.168.0.10"
report_host="192.168.0.10"

skip-name-resolve

# # Disable other storage engines #
disabled_storage_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"
 # # Replication configuration parameters #
server_id=1
gtid_mode=ON
enforce_gtid_consistency=ON
binlog_checksum=NONE
# Not needed from 8.0.21 # # Group Replication configuration #
plugin_load_add='group_replication.so'
loose-group_replication_group_name="14cfe0c5-fca1-47cd-89eb-cd8d23393dab"
loose-group_replication_start_on_boot=off
loose-group_replication_local_address= "192.168.0.10:33161"
loose-group_replication_group_seeds= "192.168.0.10:33161, 192.168.0.10:33261,
192.168.0.11:33161, 192.168.0.11:33261"
loose-group_replication_bootstrap_group=off
loose-group-replication-ssl-mode=REQUIRED
loose-group_replication_recovery_use_ssl=ON
```

【STEP03】Configure my.cnf file. (dfm-mysql-2 on server 1)

The loose-group_replication_group_name setting refers to the generation of the UUID.

```
# config
vi /dfm/db-server-2/mysql/config/my.cnf
```

```
[client]
default-character-set=utf8mb4

[mysql]
default-character-set=utf8mb4

[mysqld]
user=mysql
default-time-zone='+00:00'
event_scheduler = ON
general_log = 0
slow-query-log = 1
long_query_time = 4
lower_case_table_names = 1
collation-server = utf8mb4_unicode_ci
init-connect='SET NAMES utf8mb4'
character-set-server = utf8mb4
group_concat_max_len = 4096

port=33062
mysqlx_port=33072

bind-address="192.168.0.10"
report_host="192.168.0.10"

skip-name-resolve

# # Disable other storage engines #
disabled_storage_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"
 # # Replication configuration parameters #
server_id=2
gtid_mode=ON
enforce_gtid_consistency=ON
binlog_checksum=NONE
# Not needed from 8.0.21 # # Group Replication configuration #
plugin_load_add='group_replication.so'
loose-group_replication_group_name="14cfe0c5-fca1-47cd-89eb-cd8d23393dab"
loose-group_replication_start_on_boot=off
loose-group_replication_local_address= "192.168.0.10:33261"
```

```
loose-group_replication_group_seeds= "192.168.0.10:33161, 192.168.0.10:33261, 192.168.0.11:33161,
192.168.0.11:33261"
loose-group_replication_bootstrap_group=off
loose-group-replication-ssl-mode=REQUIRED
loose-group_replication_recovery_use_ssl=ON
```

【STEP04】Configure my.cnf file. (dfm-mysql-1 on server 2)

The loose-group_replication_group_name setting refers to the generation of the UUID.

```
# config
vi /dfm/db-server-1/mysql/config/my.cnf

 [client]
 default-character-set=utf8mb4

 [mysql]
 default-character-set=utf8mb4

 [mysqld]
 user=mysql
 default-time-zone='+00:00'
 event_scheduler = ON
 general_log = 0
 slow-query-log = 1
 long_query_time = 4
 lower_case_table_names = 1
 collation-server = utf8mb4_unicode_ci
 init-connect='SET NAMES utf8mb4'
 character-set-server = utf8mb4
 group_concat_max_len = 4096

 port=33061
 mysqlx_port=33071

 bind-address="192.168.0.11"
 report_host="192.168.0.11"

 skip-name-resolve
```

```
  # # Disable other storage engines #
  disabled_storage_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"
   # # Replication configuration parameters #
  server_id=3
  gtid_mode=ON
  enforce_gtid_consistency=ON
  binlog_checksum=NONE
  # Not needed from 8.0.21 # # Group Replication configuration #
  plugin_load_add='group_replication.so'
  loose-group_replication_group_name="14cfe0c5-fca1-47cd-89eb-cd8d23393dab"
  loose-group_replication_start_on_boot=off
  loose-group_replication_local_address= "192.168.0.11:33161"
  loose-group_replication_group_seeds= "192.168.0.10:33161, 192.168.0.10:33261, 192.168.0.11:33161,
  192.168.0.11:33261"
  loose-group_replication_bootstrap_group=off
  loose-group-replication-ssl-mode=REQUIRED
  loose-group_replication_recovery_use_ssl=ON
```

【STEP05】Configure my.cnf file. (dfm-mysql-2 on server 2)
The loose-group_replication_group_name setting refers to the generation of the UUID.

```
# config
vi /dfm/db-server-2/mysql/config/my.cnf

  [client]
  default-character-set=utf8mb4

  [mysql]
  default-character-set=utf8mb4

  [mysqld]
  user=mysql
  default-time-zone='+00:00'
  event_scheduler = ON
  general_log = 0
  slow-query-log = 1
  long_query_time = 4
  lower_case_table_names = 1
  collation-server = utf8mb4_unicode_ci
```

```
init-connect='SET NAMES utf8mb4'
character-set-server = utf8mb4
group_concat_max_len = 4096

port=33062
mysqlx_port=33072

bind-address="192.168.0.11"
report_host="192.168.0.11"

skip-name-resolve

# # Disable other storage engines #
disabled_storage_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"
 # # Replication configuration parameters #
server_id=1
gtid_mode=ON
enforce_gtid_consistency=ON
binlog_checksum=NONE
# Not needed from 8.0.21 # # Group Replication configuration #
plugin_load_add='group_replication.so'
loose-group_replication_group_name="14cfe0c5-fca1-47cd-89eb-cd8d23393dab"
loose-group_replication_start_on_boot=off
loose-group_replication_local_address= "192.168.0.11:33261"
loose-group_replication_group_seeds= "192.168.0.10:33161, 192.168.0.10:33261, 192.168.0.11:33161,
192.168.0.11:33261"
loose-group_replication_bootstrap_group=off
loose-group-replication-ssl-mode=REQUIRED
loose-group_replication_recovery_use_ssl=ON
```

【STEP06】modify the dfm_config file (dfm-mysql-1 on server 1)
mysql_config_dir: mysql config file path
mysql_data_dir: mysql data folder path

```
vi /dfm/db-server-1/config/dfm_config.json

 {
   "base_dir': "/",
   ~~~~~~~~~~~~~~~~~
   “mysql_config_dir” : “/dfm/db-server-1/mysql/config/my.cnf”
   “mysql_data_dir” : “/dfm/db-server-1/mysql/data”
   ~~~~~~~~~~~~~~
 }
```

【STEP07】modify the dfm_config file (dfm-mysql-2 on server 1)
mysql_config_dir: mysql config file path
mysql_data_dir: mysql data folder path

```
vi /dfm/db-server-2/config/dfm_config.json

  {
    “base_dir’: “/”,
    ~~~~~~~~~~~~~~~~~
    “mysql_config_dir” : “/dfm/db-server-2/mysql/config/my.cnf”
    “mysql_data_dir” : “/dfm/db-server-2/mysql/data”
    ~~~~~~~~~~~~~~
  }
```

【STEP08】modify the dfm_config file (dfm-mysql-1 on server 2)
mysql_config_dir: mysql config file path
mysql_data_dir: mysql data folder path

```
vi /dfm/db-server-1/config/dfm_config.json

  {
    “base_dir’: “/”,
    ~~~~~~~~~~~~~~~~~
    “mysql_config_dir” : “/dfm/db-server-1/mysql/config/my.cnf”
    “mysql_data_dir” : “/dfm/db-server-1/mysql/data”
    ~~~~~~~~~~~~~~
  }
```

【STEP09】modify the dfm_config file (dfm-mysql-2 on server 2)
mysql_config_dir: mysql config file path
mysql_data_dir: mysql data folder path

```
vi /dfm/db-server-2/config/dfm_config.json

  {
     "base_dir': "/",
     ~~~~~~~~~~~~~~~~~
     "mysql_config_dir" : "/dfm/db-server-2/mysql/config/my.cnf"
     "mysql_data_dir" : "/dfm/db-server-2/mysql/data"
     ~~~~~~~~~~~~~~
  }
```

9.2.2. Data zone

Setting up the minio service to work is done by creating a total of four folders.
It operates on the default port of 9000 and cannot operate on a different port between servers (9000 for server 1 and 9090 for server 2).
If created, the folder name will be set to the value incremented by 1 in the order set in minio_data_idr when bound to the container.
For example, if you bind a folder created with data, data1, data2, and data3, it will be named
data -> data1, data1 -> data2, data2 -> data3, data3 -> data4.

【STEP01】Copy the required files

```
# copy files
cp /tmp/dfm/ha/data-server/config/dfm_config.json /dfm/data-server/config/
```

【STEP02】Modify the dfm_config file

- minio_data_dir: minio data folder path (absolute path)
- cluster_minio_config_dir: minio config folder path
- cluster_minio_access_address: access address minio server(space-separated access point)

```
vi /dfm/data-server/config/dfm_config.json

  {
     "base_dir': "/",
     ~~~~~~~~~~~~~~~~~
     "minio_data_dir" : "/dfm/data-server/minio/data, /dfm/data-server/minio/data1, /dfm/data-
  server/minio/data2, /dfm/data-server/minio/data3"
     "cluster_minio_access_address" : "http://192.168.0.10/data1 http://192.168.0.10/data2
  http://192.158.0.10/data3 http://192.168.0.10/data4 http://192.168.0.11/data1 http://192.168.0.11/data2
  http://192.158.0.11/data3 http://192.168.0.11/data4"
     "cluster_minio_config_dir" : "/dfm/data-server/minio/config"
     ~~~~~~~~~~~~~~
```

```
  }
```

9.2.3. App zone

【STEP01】Copy required file

```
# copy file for haproxy
cp -r /tmp/dfm/ha/app-server/haproxy-config/* /dfm/app-server/haproxy/config/

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

【STEP02】Modify the dfm_config file

For the settings below, except cluster_minio_access_address, please refer to setion 4.4.9 (STEP07) Set-up Configuration.
If you set listen_port and access_port to 80 when the external access port (cluster_service_port) is port 80, you will get a port conflict.
The cluster_minio_access_address value should contain the name of the haproxy container in the app zone.
Here, we write it as dfm-minio-app because we creating it with the dfm-minio-app name.

```
# for server 1
vi /dfm/app-server/config/dfm_config.json


  {
     “host_ip’: “192.168.0.10”,
     “listen_port”:”10010”,
     “listen_scheme”:”http”,
     “access_scheme”:”http”,
     “access_address”:”192.168.0.10”,
     “access_port”:” 10010”,
     ~~~~~~~~~~~~~~~~~~
     “console_log_dir” : “/dfm/app-server/console/logs”,
     “core_log_dir” : “/dfm/app-server/core/logs”,
     “haproxy_config_dir” : “/dfm/app-server/haproxy/config”,
     ~~~~~~~~~~~~~~~
     “cluster_service_address” : “http://efota-test.com”,
     “cluster_service_scheme” : “http”,
     “cluster_service_port” : “80”,
     “cluster_minio_access_address” : “dfm-proxy-app”,
     “cluster_minio_access_port” : “9000”,
     “cluster_minio_access_scheme” : “http”,
```

```
    “cluster_mysql_access_url” :
  “192.168.0.10:33061,192.168.0.10:33062,192.168.0.11:33061,192.168.0.11:33062”,
    ~~~~~~~~~~~~~~
  }

# for server 2
vi /dfm/app-server/config/dfm_config.json

  {
    “host_ip’: “192.168.0.11”,
    “listen_port”:”10010”,
    “listen_scheme”:”http”,
    “access_scheme”:”http”,
    “access_address”:”192.168.0.11”,
    “access_port”:” 10010”,
    ~~~~~~~~~~~~~~~~~
    “console_log_dir” : “/dfm/app-server/console/logs”,
    “core_log_dir” : “/dfm/app-server/core/logs”,
    “haproxy_config_dir” : “/dfm/app-server/haproxy/config”,
    ~~~~~~~~~~~~~~
    “cluster_service_address” : “efota-test.com”,
    “cluster_service_scheme” : “http”,
    “cluster_service_port” : “80”,
    “cluster_minio_access_address” : “dfm-proxy-app”,
    “cluster_minio_access_port” : “9000”,
    “cluster_minio_access_scheme” : “http”,
    “cluster_mysql_access_url” :
  “192.168.0.10:33061,192.168.0.10:33062,192.168.0.11:33061,192.168.0.11:33062”,
    ~~~~~~~~~~~~~~
  }
```

【STEP03】Create the network for app zone

Create a new dfm-network for the app zone.

- name: dfm-network-app
- subnet: 100.0.1.0/24

```
# create network
dfm cluster network create -n dfm-network-app -f /dfm/app-server/config/dfm_config.json -
subnet 100.0.1.0/24

The dfm network was created with the name "dfm-network-app".
```

```
# check network
docker network ls

NETWORK ID     NAME                DRIVER   SCOPE
515b8fe23711   dfm-network-app     bridge   local
```

【STEP04】Modify the haproxy.cfg file

Modify the server IP for minio access.
The dns1 value changes according to the subnet value of the network created in the step above. For example, if your network created 100.0.1.0, you should set it to 100.0.1.1.

```
vi /dfm/app-server/haproxy/config/haproxy.cfg

resolvers docker
  nameserver dns1 100.0.1.1:53
  resolve_retries 3
  timeout resolve 1s
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~{
    backend dfmMinioReplaceHostBackend
      mode http
      option httpchk GET /minio/health/live
      http-check expect status 200
      default-server inter 5s fall 3 rise 2
      balance leastconn

      # if minio server use ssl
      #server dfm-minio 192.168.1.60:9000 ssl verify none check
      # otherwise
      server dfm-minio-group-1 192.168.0.10:9000 check
      server dfm-minio-group-2 192.168.0.11:9000 check
```

9.2.4. WEB zone

【STEP01】Copy the required file

```
# copy file for haproxy
cp -r /tmp/dfm/ha/web-server/haproxy-config/* /dfm/web-server/haproxy/config/

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

【STEP02】Modify the haproxy.cfg file

Proceed to 4.6.8 and 4.6.9 for further explain.

```
vi /dfm/web-server/config/dfm_config.json

  {
     “base_dir’: “/”,
     “host_ip’: “192.168.0.10”,
     “listen_port”:”80”,
     “listen_scheme”:”http”,
     “access_scheme”:”http”,
     “access_address”:” efota-test.com”,
     “access_port”:” 80”,
     “haproxy_config_dir”:” /dfm/web-server/haproxy/config”,
     ~~~~~~~~~~~~~~~~~
  }
```

【STEP03】Modify the haproxy.cfg file

Proceed to 4.6.8 and 4.6.9 for further explain.

```
vi /dfm/web-server/haproxy/config/haproxy.cfg

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  backend dfmCoreProxyBackend
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     server dfm-coreproxy_1 10.31.0.237:10010 check cookie dfm-coreproxy_1
     server dfm-coreproxy_2 10.31.0.150:10010 check cookie dfm-coreproxy_2
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  backend dfmMinioProxyBackend
     mode http
     option httpchk GET /minio/health/live
```

```
http-check expect status 200
default-server inter 5s fall 3 rise 2
#set up point
http-request set-header Host dfm-proxy-app:9000
 server dfm-minio-1 192.168.0.10:9000 check
server dfm-minio-2 192.168.0.11:9000 check
```

9.2.5. keepalived

Proceed to 4.5 for more details.

9.3Start-up services

You will need to use the modified dfm command to run the same DFM service on the same server.
Each dfm command requires you to import the dfm_config.json file, which is the configuration file for the individual dfm service, upon execution.
For example, for the dfm-mysql-1 server, the /dfm/db-server-1/config/dfm_config.json file must be imported when running the dfm command.

dfm cluster {[start|…] run type} {[dfm-mysql|…] service name} -f {service config file path} -n {service another name} --network {dfm network name} --subnet {subnet mask}

- run type : start, terminate, restart
- service name : dfm-mysql, dfm-minio, dfm-core, dfm-console, dfm-proxy
- service config file path: dfm_config.json file absolute path(default: /dfm/config/dfm_config.json)
- service another name: container execute name. (default: same name as service name)
- dfm network name: the additional network name(default: dfm-network)
- subnet mask: parameters required when createing a network (default: 100.0.0.0/24)

```
dfm cluster start dfm-mysql -f /dfm/db-server-1/config/dfm_config.json -n dfm-mysql-1
```

【STEP01】Start the Mysql service (server1, server2)

```
dfm cluster start dfm-mysql -f /dfm/db-server-1/config/dfm_config.json -n dfm-mysql-1
dfm cluster start dfm-mysql -f /dfm/db-server-2/config/dfm_config.json -n dfm-mysql-2
```

【STEP02】Configure group replication.
See the group replication topic in 4.2.9

【STEP03】Start minio service.(server1, server2)

```
dfm cluster start dfm-minio -f /dfm/data-server/config/dfm_config.json
```

【STEP04】Start haproxy service on data zone (server1, server2)

```
dfm cluster start dfm-proxy -f /dfm/app-server/config/dfm_config.json -n dfm-proxy-app --network
dfm-network-app
```

【STEP05】Start core, console services on data zone (server1, server2)

```
dfm cluster start dfm-core -f /dfm/app-server/config/dfm_config.json --network dfm-network-app
dfm cluster start dfm-console -f /dfm/app-server/config/dfm_config.json --network dfm-network-
app
```

【STEP06】Start haproxy service on web zone (server1, server2)

```
dfm cluster start dfm-proxy -f /dfm/web-server/config/dfm_config.json
```

【STEP07】Start keepalived (server1, server2)

See section 4.5.3 for more information.

PART VII: 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 cluster terminate dfm-mysql
   ```

2. DFM Firmware Storage (MinIO)

   Stop the server with the following command:

   ```
   dfm cluster terminate dfm-minio
   ```

3. DFM Core Server

   Stop the server with the following command:

   ```
   dfm cluster terminate dfm-core
   ```

4. DFM Admin Console Server

   Stop the server with the following command:

   ```
   dfm cluster terminate dfm-console
   ```

5. DFM HAProxy Server

   Stop the server with the following command:

   ```
   dfm cluster 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**	Red hat	8.4, 9.2, or 9.6	Enterprise **(Paid)**	
	Container	Podman	over 4.0		
	Database	MySQL	Community Edition	Community (Free)	https://www.mysql.com/products/
PoC	**Server OS**	Red hat	8.4, 9.2, or 9.6	Community (free)	
	Container	Podman	Over 4.0		
	Database	MySQL	Community Edition	Community (Free)	

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) Name:______________________________ Signature:___________________________ Date: ______________________________	**< Customer Name >** Name:______________________________ Signature:___________________________ 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.

Appendix F. Set E-FOTA agent config by managed Configuration

The Knox E-FOTA On-Premises agent requires your server URL and TLS certificate to connect to your on-premises server. The Knox E-FOTA On-Premises agent supports a managed configuration which lets you configure connection and authentication details needed for the app to communicate with your on-premises server. Follow your EMM's documentation on how to configure an app's managed configuration.

Reference: https://developer.android.com/work/managed-configurations

The managed configuration has the following fields:

Field	Description
server_url	*string* The URL of your Knox E-FOTA On-Premises server
pem	*string* Your server's TLS certificate
client_cert_pem	*string* (Only used for mutual TLS) Client certificate used by clients to authenticate themselves to your server
client_key_pem	*string* (Only used for mutual TLS) Encrypted private key used by clients for signing
client_pem_password	*string* (Only used for mutual TLS) Password for the encrypted private key

Appendix G. Set up mutual TLS

To use mutual TLS, you must create the client certificate and its private key:

1. Create your private key:

```
openssl genrsa -out key.pem 4096
```

2. Create your certificate:

```
openssl req -x509 -new -key key.pem -out cert.pem -days 365
```

3. Encrypt your private key:

```
openssl pkcs8 -topk8 -in key.pem -out key.pem -v2 aes-128-cbc
```

4. Combine the two files to create efota_client.pem:

```
cat key.pem cert.pem > efota_client.pem
```

5. Create a copy of cert.pem named client.pem for HAProxy:

```
copy cert.pem client.pem
```

6. Move client.pem to set up HAProxy

```
copy client.pem /dfm/haproxy/config
```

7. Add the settings below to haproxy.cfg:

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

frontend fe_web

    bind *:80
    bind *:443 ssl crt /usr/local/etc/haproxy/example-sec-fota.net.pem ca-file /usr/local/etc/haproxy/client.pem verity optional

    # monitoring uri
       monitor-uri /health

       http-request capture req.hdr(Host) len 100

       acl acl_dfm_device path_reg ^/dfm/device/v1/*
       acl has_client_cert ssl_fc_has_crt eq 1
       http-request deny if acl_dfm_device !has_client_cert
       use_backend dfmCoreBackend if acl_dfm_device
```

8. To apply the client certificate to your devices, enter the password used to encrypt the private key in the second line of the efota_config file, then push the efota.pem, efota_client.pem, and efota_config files to the Download folder on each of your devices. You can also use the managed configuration. For details, see the Admin guide.

< EOF (End Of File) >