MetaDefender Cluster

v2.10.0

Contents

Installation 4

- Overview 4
- System requirements 5
- Installation 14
  - Recommended Setup 21
  - PostgreSQL 22
    - Windows 23
    - Linux 26
    - Air-gapped 29
  - Redis 37
    - Windows 37
    - Linux 42
    - Air-gapped 45
  - RabbitMQ 53
    - Windows 54
    - Linux 55
    - Air-gapped 61
  - MD Cluster File Storage 69
    - Windows 69
    - Linux 77
  - MD Cluster Identity Service 86
    - Windows 86
    - Linux 95
  - MD Cluster Control Center 103
    - Windows 103
    - Linux 115
  - MD Cluster Worker 126
  - Lake and Warehouse setup 135
- License activation 137
  - Online Activation 137
  - Offline Activation 139
  - License Management Server Activation 144
- Module update 146
- MetaDefender Cluster Compatibility Matrix 150
- MetaDefender Cluster Documentation 151

Configurations 152

- High Availability 152
  - High Availability support for File Storage 152
  - High Availability support for RabbitMQ 154
  - High Availability support for Redis 157
  - High Availability support for PostgreSQL Data lake 160
- System settings 168
  - Remote Support Package Gathering 168
  - Security 170
- File Storage 175

Operating 180

- Dashboard 180
- History 184
  - Processing History 184
  - Audit Log 186
- Inventory 187
  - Services 187
  - Workers 191
  - Modules 202
  - Licenses 208
  - Installers 215
- Workflow Management 216
  - Workflow Configuration Template 217
  - Security zone configuration 230
  - Workflow rule 230
- User Management 236
  - User directory 237
- Settings 245
  - Security 245
  - Module update 249
  - Data retention 250
  - Health check 251
  - Export 252

Upgrade 253

- System Upgrade 253

Performance 262

- Performance and Load Estimation 262

Container Deployment 271

Kubernetes 271
Kubernetes components 272
Quick start 274
Configuration 276
Enable HTTPs 276
Resource requirements 277
Upgrade 279
Scaling 280
High availability 282
Troubleshooting 288

Troubleshooting 292

Log Gathering in MetaDefender Cluster 292
Open Connection On PostgreSQL Server 294
What is the latest MetaDefender Cluster version? 295
Configure Firewall Rules to Allow Incoming Connections 297

Release Notes 307

Release notes 307
Archived release notes 309

Knowledge Base 336

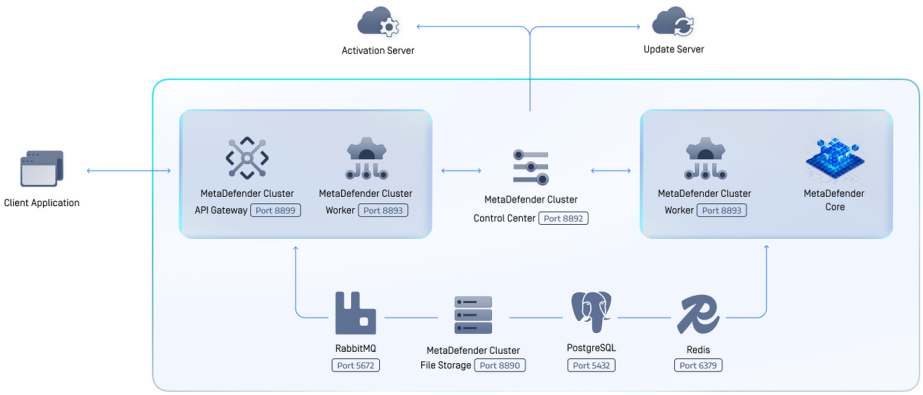
Use a RAM disk for MetaDefender Core temporary files 336

API References

API Gateway 341

Control Center 425

Overview



The MetaDefender Cluster (MD Cluster) is an approach to serve very large deployments while offering improved auto-scaling, high availability and fault tolerant capabilities for MetaDefender Core.

The MetaDefender Cluster consists of several components:

Component	Functionalities
MD Cluster Control Center	Assist administrators with user management, system health monitoring, and deploying or upgrading MetaDefender Core or MD Cluster API Gateway without any downtime.
MD Cluster Identity Service	Assist MD Cluster Control Center and MD Cluster API Gateway in client authentication, managing user activity sessions and authorization.
MD Cluster File Storage	Securely store and share files asynchronously across components in the cluster. The component manages the duration and duplication of files.
MD Cluster Worker	Deploy and monitor activities of MetaDefender Core and MD Cluster API Gateway.
MD Cluster API Gateway	Accept file scans, fetch scan statuses, and process download requests from clients.
MetaDefender Core	Scan the accepted files.

Component	Functionalities
RabbitMQ - Message broker	Receives tasks from MD Cluster API Gateway and forwards them to MetaDefender Core instances for processing.
Redis - Caching server	Store in-progress results in memory for rapid retrieval.
PostgreSQL - Database sever	Permanently store scan results, configuration and executive reports.

The MD Cluster offers users two distinct interfaces. The first is a RESTful interface provided by MD Cluster **API Gateway** for applications to upload files for scanning, retrieve scan status, download processed files, or abort file scanning. The other is a Web UI provided by MD Cluster **Control Center** for the system administrator to manage licenses and users, modify workflow configurations, monitor the overall system, and remotely deploy or upgrade MD Cluster API Gateway or MetaDefender Core.

When a file is submitted to the MD Cluster **API Gateway** for scanning, its body content is securely transmitted to MD Cluster File Storage for subsequent use. API Gateway submits a scan task in RabbitMQ queue, and responds to the application with `data_id`. The task is delivered to healthy MetaDefender Core instances and one of them will accept the task. The file corresponding to the task is transmitted from MD Cluster File Storage to the instance's local storage, and the processing of the file takes place. Scan results produced by the processing are continuously recorded in Redis for fast retrieval and are finally stored in the PostgreSQL database for long-term storage. If created, the sanitized or watermarked file is securely transmitted to MD Cluster File Storage for future download by MD Cluster API Gateway.

In certain rate situations, if one of the MetaDefender Core instances unexpectedly ceases operation, its 'broken' files are delivered to other MetaDefender Core instances for continued processing without the need for applications to resubmit the files. By leveraging MD Cluster File Storage and RabbitMQ, MetaDefender Core instances within MetaDefender Cluster can collaborate in distributing the workload of archive extraction, greatly decreasing the overall time required to process archive files while utilizing the resources much more efficiently.

Using the Web Console from by MD Cluster **Control Center**, the system administrator is able to adjust workflow settings centrally and, after which the updates are automatically synced across all MetaDefender Core instances. The administrator can scale out the number of MD Cluster API Gateway or MetaDefender Core instances if additional power is required. He or she can also upgrade the instances seamlessly while the file processing is occurring. All statistical data and health information for components, along with executive reports, can be accessed easily through the Web UI of MD Cluster Control Center.

System requirements

This page describes the software dependencies and recommended system requirements for deploying MetaDefender Cluster (MD Cluster) components on supported Windows and Linux platforms. Review these requirements before installation to ensure your environment is properly prepared for a stable and supported deployment.

Windows

PostgreSQL

- **Supported version:** 16.9 or later
- **System requirements:**
  - 8 CPU cores
  - 16 GB RAM
  - 5 Gbps network bandwidth
  - 1 TB SSD or NVMe storage

> **Info**
> For a deployment with 5 MD Core instances, each running 8 AV engines, approximately 4 GB of disk space is needed to store 1 million objects.

RabbitMQ

- **Supported version:** 4.2 or later
- **Dependencies:**
  - 64-bit Erlang/OTP. Please refer to the link for the Erlang-RabbitMQ compatibility matrix.
- **System requirements:**
  - 4 CPU cores
  - 8 GB RAM
  - 1 Gbps network bandwidth
  - 100 GB storage

MD Cluster File Storage

- **Supported version:** 2.6.0
- **Dependencies:**
  - Microsoft Visual C++ Redistributable 2019
  - Version 14.29.30139.0 or later
- **Recommended system requirements:**

- 8 CPU cores
- 8 GB RAM
- 5 Gbps network bandwidth
- 1 TB SSD or NVMe storage

Info

The available storage volume should be at least twice the total size of files submitted to the MD Cluster API Gateway at the same time.

If CDR or DLP engines are installed on MD Core instances, the required file storage volume should be doubled.

MD Cluster Control Center

- **Supported version:** 2.6.0
- **Dependencies:**
  - Microsoft Visual C++ Redistributable 2019
  - Version 14.29.30139.0 or later
- **Recommended system requirements:**
  - 4 CPU cores
  - 4 GB RAM
  - 1 Gbps network bandwidth
  - 100 GB storage

MD Cluster Identity Service

- **Supported version:** 2.6.0
- **Dependencies:**
  - Microsoft Visual C++ Redistributable 2019
  - Version 14.29.30139.0 or later
- **Recommended system requirements:**
  - 4 CPU cores
  - 4 GB RAM
  - 1 Gbps network bandwidth

- 50 GB storage

Worker hosting MD Cluster API Gateway

- **Supported version:** 2.6.0
- **Dependencies:**
  - Microsoft Visual C++ Redistributable 2019
  - Version 14.29.30139.0 or later
- **Recommended system requirements:**
  - 8 CPU cores
  - 16 GB RAM
  - 5 Gbps network bandwidth
  - 50 GB storage

Worker hosting MD Core

- **Supported version:** 2.6.0
- **Dependencies:**
  - Microsoft Visual C++ Redistributable 2019
  - Version 14.29.30139.0 or later
- **Recommended system requirements:**
  - Refer here for detailed requirements

Worker hosting MD Cluster Callback Service

- **Supported version:** 2.6.0
- **Dependencies:**
  - Microsoft Visual C++ Redistributable 2019
  - Version 14.29.30139.0 or later
- **Recommended system requirements:**
  - 4 CPU cores
  - 4 GB RAM

- 5 Gbps network bandwidth
- 50 GB storage

Worker hosting MD Cluster Download Service

- **Supported version:** 2.10.0
- **Dependencies:**
  - Microsoft Visual C++ Redistributable 2019
  - Version 14.29.30139.0 or later
- **Recommended system requirements:**
  - 4 CPU cores
  - 4 GB RAM
  - 5 Gbps network bandwidth
  - 50 GB storage

Notes

- MetaDefender Cluster requires **WMIC** to be enabled. Run the following command in Command Prompt as Administrator:

Code · powershell

```
DISM /Online /Add-Capability /CapabilityName:WMIC
```

Debian / Ubuntu / Red Hat / Rocky

PostgreSQL

- **Supported version:** 16.9 or later
- **System requirements:**
  - 8 CPU cores
  - 16 GB RAM
  - 5 Gbps network bandwidth
  - 1 TB SSD or NVMe storage

Info

For a deployment with 5 MD Core instances, each running 8 AV engines, approximately 4 GB of disk space is needed to store 1 million objects.

Redis

- **Supported version:** 7.2 or later
- **System requirements:**
  - 2 CPU cores
  - 32 GB RAM
  - 5 Gbps network bandwidth
  - 20 GB storage

RabbitMQ

- **Supported version:** 4.2 or later
- **Dependencies:**
  - 64-bit Erlang/OTP. Please refer to the link for the Erlang-RabbitMQ compatibility matrix.
- **System requirements:**
  - 4 CPU cores
  - 8 GB RAM
  - 1 Gbps network bandwidth
  - 100 GB storage

MD Cluster File Storage

- **Supported version:** 2.6.0
- **Required packages/tools:**
  - uuid
  - tar
  - lsb_release
- **Recommended system requirements:**

- 8 CPU cores
- 8 GB RAM
- 5 Gbps network bandwidth
- 1 TB SSD or NVMe storage

> **Info**
>
> The available storage volume should be at least twice the total size of files submitted to the MD Cluster API Gateway at the same time.
>
> If CDR or DLP engines are installed on MD Core instances, the required file storage volume should be doubled.

MD Cluster Control Center

- **Supported version:** 2.6.0
- **Required packages/tools:**
  - uuid
  - tar
  - lsb_release
- **Recommended system requirements:**
  - 4 CPU cores
  - 4 GB RAM
  - 1 Gbps network bandwidth
  - 100 GB storage

MD Cluster Identity Service

- **Supported version:** 2.6.0
- **Required packages/tools:**
  - uuid
  - tar
  - lsb_release
- **Recommended system requirements:**
  - 4 CPU cores

- 4 GB RAM
- 1 Gbps network bandwidth
- 50 GB storage

Worker hosting MD Cluster API Gateway

- **Supported version:** 2.6.0
- **Required packages/tools:**
  - uuid
  - tar
  - lsb_release
  - jq
  - curl
- **Recommended system requirements:**
  - 4 CPU cores
  - 8 GB RAM
  - 5 Gbps network bandwidth
  - 50 GB storage

Worker hosting MD Core

- **Supported version:** 2.6.0
- **Required packages/tools:**
  - uuid
  - tar
  - lsb_release
  - jq
  - curl
- **Recommended system requirements:**
  - Refer here for detailed requirements

Worker hosting MD Callback Service

- **Supported version:** 2.6.0
- **Required packages/tools:**
  - uuid
  - tar
  - lsb_release
  - jq
  - curl
- **Recommended system requirements:**
  - 4 CPU cores
  - 4 GB RAM
  - 5 Gbps network bandwidth
  - 50 GB storage

Worker hosting MD Cluster Download Service

- **Supported version:** 2.10.0
- **Required packages/tools:**
  - uuid
  - tar
  - lsb_release
  - jq
  - curl
- **Recommended system requirements:**
  - 4 CPU cores
  - 4 GB RAM
  - 5 Gbps network bandwidth
  - 50 GB storage

Notes

- MD Cluster requires `uuid`, `tar`, `curl` , `jq` to be installed. Run the following command in Terminal to install them:

```
Code · bash
```

```
# Debian/Ubuntu
sudo apt install -y uuid curl jq tar

# Red Hat/Rocky
sudo dnf install -y libuuid curl jq tar
```

- MD Cluster requires `lsb_release` on Rocky Linux. Run the following commands in Terminal to install it:

Code · bash

```
sudo dnf install -y yum-utils
sudo dnf config-manager --set-enabled devel
sudo dnf update -y
sudo dnf install -y redhat-lsb-core
```

Installation

Overview

This section describes how to install and set up **MetaDefender Cluster** (MD Cluster), a distributed system composed of multiple services, including:

- Redis
- RabbitMQ
- PostgreSQL
- MetaDefender Cluster File Storage (MD Cluster File Storage)
- MetaDefender Cluster Identity (MD Cluster Identity)
- MetaDefender Cluster Control Center (MD Cluster Control Center)
- MetaDefender Cluster Worker (MD Cluster Worker)

MD Cluster can be deployed on **virtual machines** or **physical servers**. After installation, all services are connected to **MD Cluster Control Center**, where licensing is activated and system health is monitored.

Architecture Overview

MD Cluster consists of the following components:

Component	Default port	Description
Redis	6379	Caching and fast data access
RabbitMQ	5672	Message broker

Component	Default port	Description
PostgreSQL	5432	Persistent data storage
MD Cluster File Storage	8890	File storage service
MD Cluster Identity Service	8891	Authentication and authorization service
MD Cluster Control Center	8892	Central control and management
MD Cluster Worker	8893	Instance lifecycle management

Info

All other services must be connected to MD Cluster Control Center for the system to function correctly.

Prerequisites

Ensure the following requirements are met before starting the installation.

Requirement	Description
Deployment type	Virtual machines or physical servers.
Privileges	**Administrator** privileges on Windows; **Root** or **sudo** access on Linux
Network	All nodes must be able to communicate with each other
Port	Required ports for Redis, RabbitMQ, PostgreSQL, and MD Cluster services must be open
Time synchronization	NTP must be configured across all nodes
Resources	Adequate CPU, RAM, and disk for each service

Info

Ensure that firewall rules allow communication between all services (Redis, RabbitMQ, PostgreSQL, and MD Cluster services).

Installation procedure

Step 1: Install infrastructure services

Install the required third-party services on designated nodes.

1. Install Redis on Windows, Linux, or Air-gapped.
2. Install RabbitMQ on Windows, Linux, or Air-gapped.
3. Install PostgreSQL on Windows, Linux, or Air-gapped.

Success

Ensure all services are installed and running before proceeding to the next step.

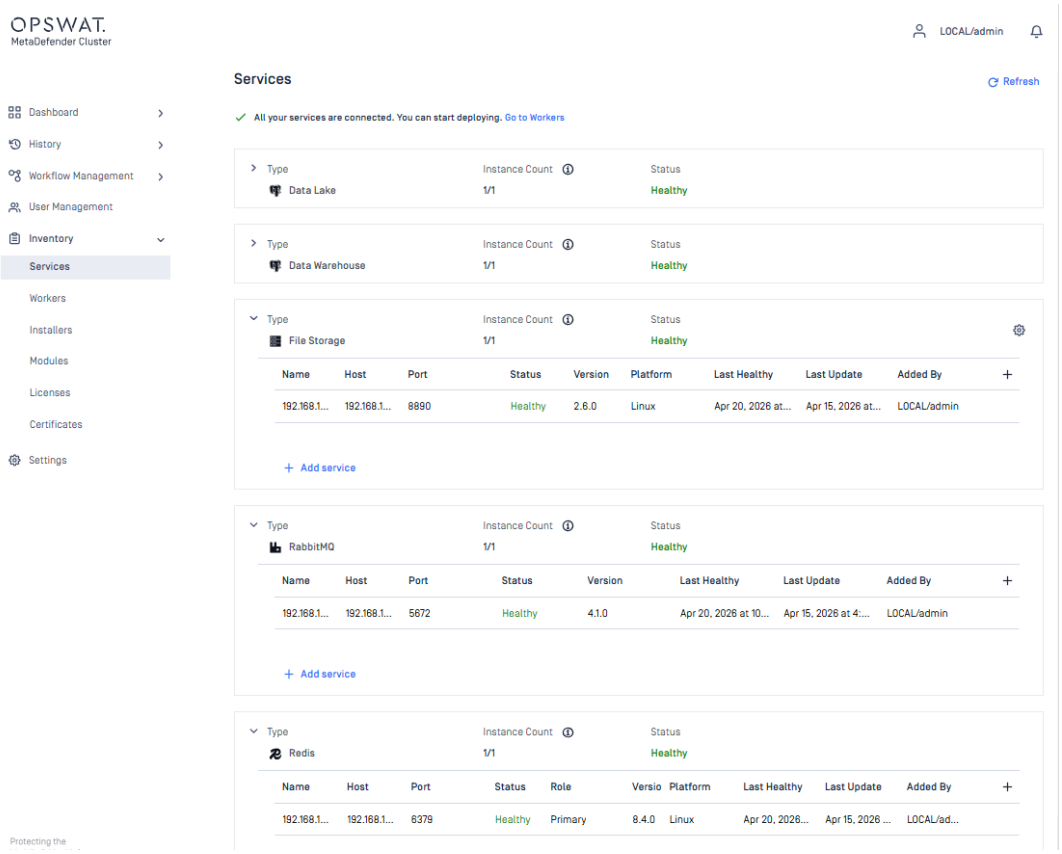
Step 2: Install and setup core MD Cluster services

1. Install MD Cluster File Storage on Windows or Linux.
2. Install MD Cluster Identity Service on Windows or Linux.
3. Install MD Cluster Control Center on Windows or Linux.
4. Setup Data Lake and Warehouse.

Step 3: Connect services to Control Center

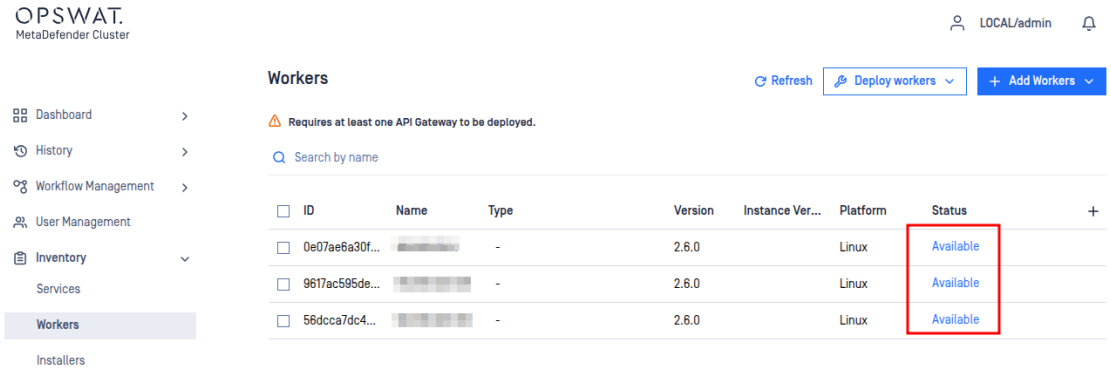
After installation, register all infrastructure services and MD Cluster File Storage to Control Center.

1. Sign in to MD Cluster **Control Center** console using your administrator account.
2. From the sidebar, go to `Inventory` > `Services`.
3. Follow the instructions to add the services.
4. Verify connectivity to
    1. PostgreSQL (including DataLake and Data Warehouse)
    2. File Storage
    3. RabbitMQ
    4. Redis



Step 4: Install and register MD Cluster Worker

1. Follow the instructions to obtain the installation script.
2. Access each target machine and run the script.
3. Sign in to MD Cluster **Control Center** console.
4. From the sidebar, go to `Inventory` > `Workers`.
5. Verify that the installed workers are showing as `Available`.



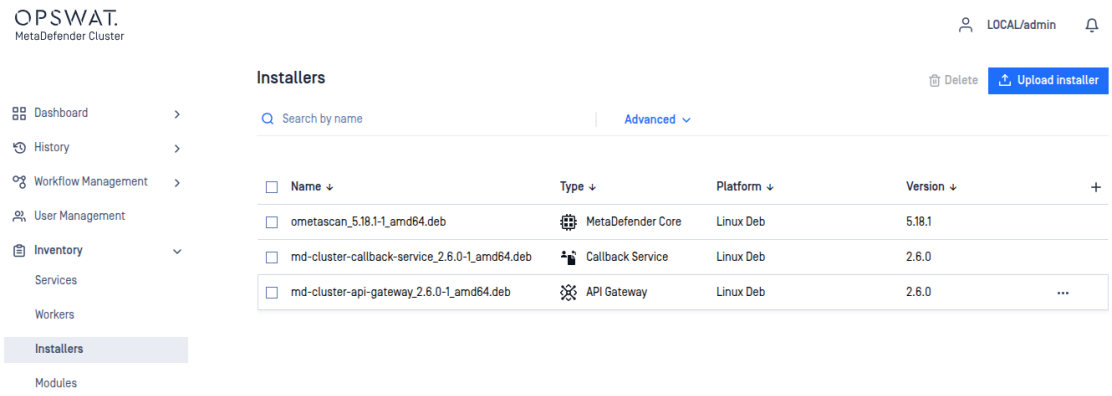
Step 5: Submit execution installers

Execution installers are deployable units managed by MD Cluster Worker, responsible for executing defined processing tasks within the system's main flow. These installers are:

- MetaDefender Core (MD Core)
- MetaDefender Cluster API Gateway (MD Cluster API Gateway)
- MetaDefender Cluster Callback Service (MD Cluster Callback Service)

Typical steps:

1. Sign in to MD Cluster Control Center console using your administrator account.
2. From the sidebar, go to `Inventory` > `Installers`.
3. Follow the instructions to upload the installers.

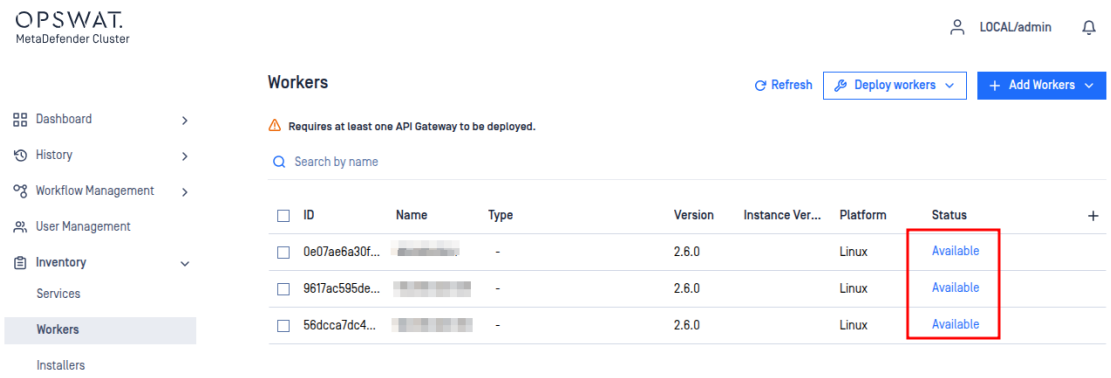


Info

While MD Core and MD Cluster API Gateway are **essential** to the main flow, MD Cluster Callback Service is **optional** and only needed when scan results are delivered via webhooks.

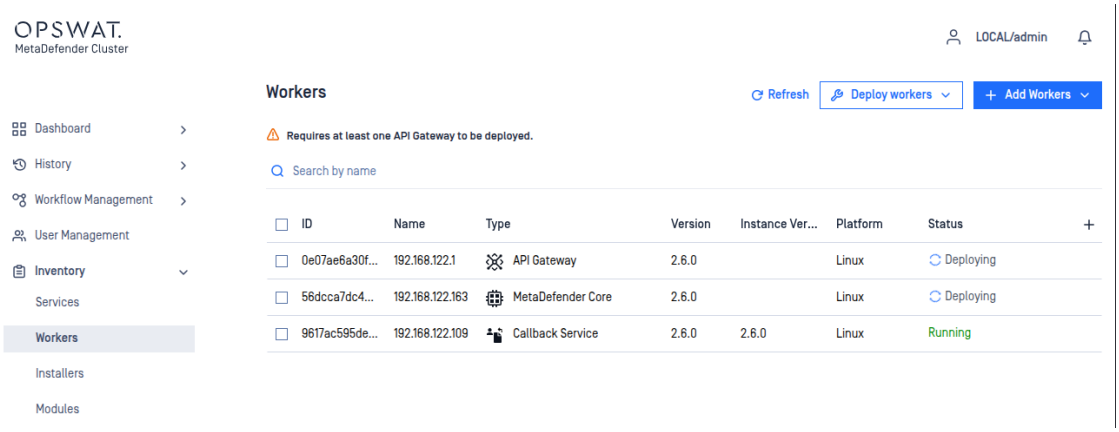
Step 6: Deploy execution instances

1. Sign in to MD Cluster Control Center console.
2. From the sidebar, go to `Inventory` > `Workers`.
3. Ensure that MD Cluster Workers are showing as `Available`.



4. Follow the instructions to deploy execution instances.

5. Verify that all instances are successfully deployed.



Info

At least one instance of **MD Cluster API gateway** and **MD Core** must be deployed on MD Cluster Workers for the system to function fully.

Step 7: Activate license

1. Sign in to MD Cluster **Control Center** console.

2. From the sidebar, go to `Inventory` > License

3. Follow the instructions to activate the product according to your license type.

4. Confirm activation.

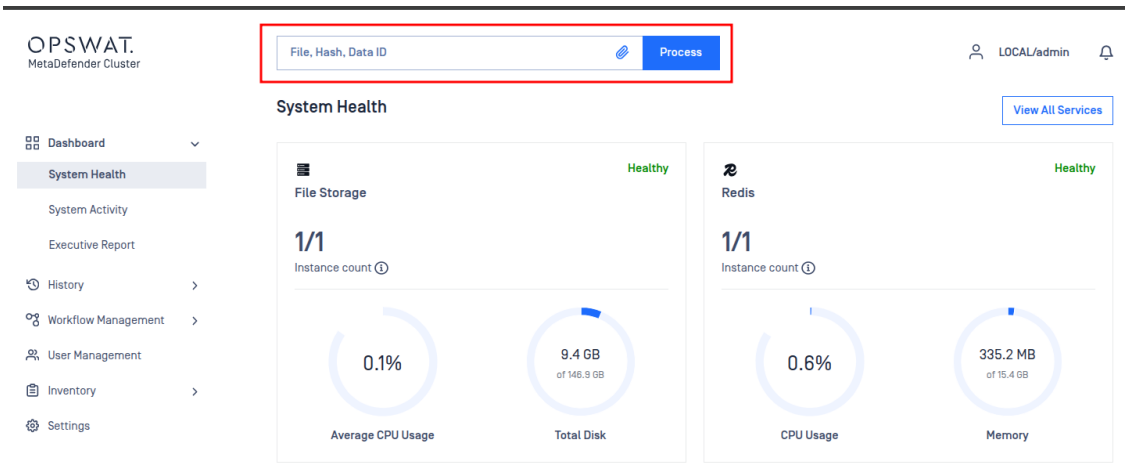


Step 8: Verify system activity

After licensing, verify system status.

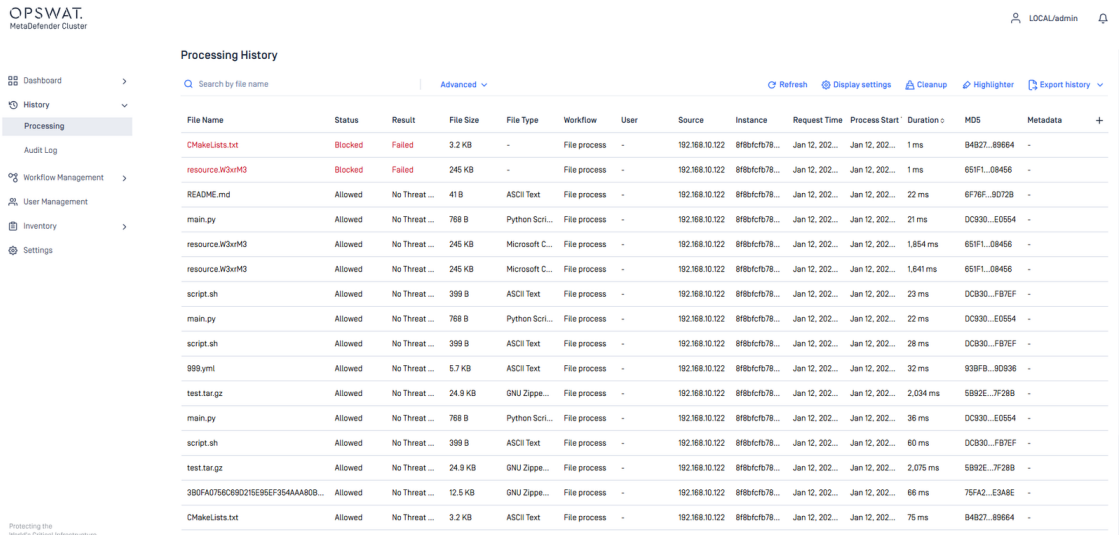
1. Sign in to the MD Cluster **Control Center** console.

2. Submit a file for scanning from the Control Center.



> Submitting files from the Control Center is intended only to verify that the system is working correctly. For actual scanning workloads, submit files through the API Gateway using POST – Analyze File (Asynchronous Mode).

3. Go to `History` > `Processing` and confirm that the files has been processed.



Log files

Platform	Service	Log location
Windows	MD Cluster Control Center	`C:\Program Files\OPSWAT\MetaDefender Cluster Control Center\data\log\control-center.log`
Linux	MD Cluster Control Center	`var/log/md-cluster-control-center/control-center.log`

Recommended Setup

While it is technically possible to install Redis Cache Server, RabbitMQ Message Broker, PostgreSQL Database Server, MetaDefender Cluster (MD Cluster) File Storage, and other components on the same machine, OPSWAT strongly recommends deploying them on separate servers to achieve better performance, scalability, and reliability.

Redis Caching Server

Redis consumes a significant amount of memory during operation. To ensure optimal performance, it should be deployed on a dedicated machine with high-capacity, high-speed RAM.

RabbitMQ Message Broker

RabbitMQ is a critical component of MD Cluster architecture. It is responsible for distributing tasks evenly across MetaDefender Core (MD Core) instances, rerouting failed tasks to healthy instances, and balancing workloads when additional Core instances are added to the environment.

Because of its central role in maintaining system stability and scalability, RabbitMQ should be hosted on a dedicated server.

Postgres Database Server

The MD Cluster database is divided into two primary components:

- **Data Lake** stores scan results and request-related information such as `data_id`, file hashes, and metadata. Since it is shared between MD Core and MD Cluster API Gateway instances, it should be hosted on high-speed, large-capacity storage with a reliable, high-bandwidth network connection.
- **Data Warehouse** periodically collects information from Data Lake and prepares data for executive reporting. Because executive reports may need to be retained for long periods and accessed by MD Cluster Control Center, Data Warehouse should also be deployed on a server with large storage capacity.

MetaDefender Cluster File Storage

MD Cluster File Storage is shared among MD Core and MD Cluster API Gateway instances. It stores submitted files from all connected instances and therefore requires substantial disk capacity.

Since all file-related traffic passes through this component, a high-speed network connection is essential. OPSWAT recommends hosting MD Cluster File Storage on Rocky Linux 9.0.

MetaDefender Cluster API Gateway

Due to operating system differences and NGINX support considerations between Windows and Linux, MD Cluster API Gateway should be deployed on a Linux server running Rocky Linux 9.0 to ensure high throughput for file scan submissions.

MetaDefender Core

One of the key strengths of MD Cluster is its support for hybrid deployments. MD Cluster API Gateway and MD Cluster File Storage can run on Linux servers, while MD Core instances can be deployed on either Windows or Linux.

This flexibility allows customers to choose the operating system that best fits their infrastructure and operational requirements.

Warning

It is recommended to deploy all MD Core instances on the same operating system, either Windows or Linux. Mixed operating system deployments are not supported.

PostgreSQL

Supported Version

Requires PostgreSQL version 16.9 or later.

Info

The instructions below are based entirely on the PostgreSQL document. The following installation guidelines describe how to install PostgreSQL 16.10.

Supported Operating Systems

OS	Version
Windows	Windows 11 23H2+, or Windows Server 2019+.
Linux	Ubuntu 22.04+, Ubuntu 24.04+, Debian 12+, Rocky 9+, or RHEL 9+.

Windows

Info

PostgreSQL version 16.9 or higher is required.

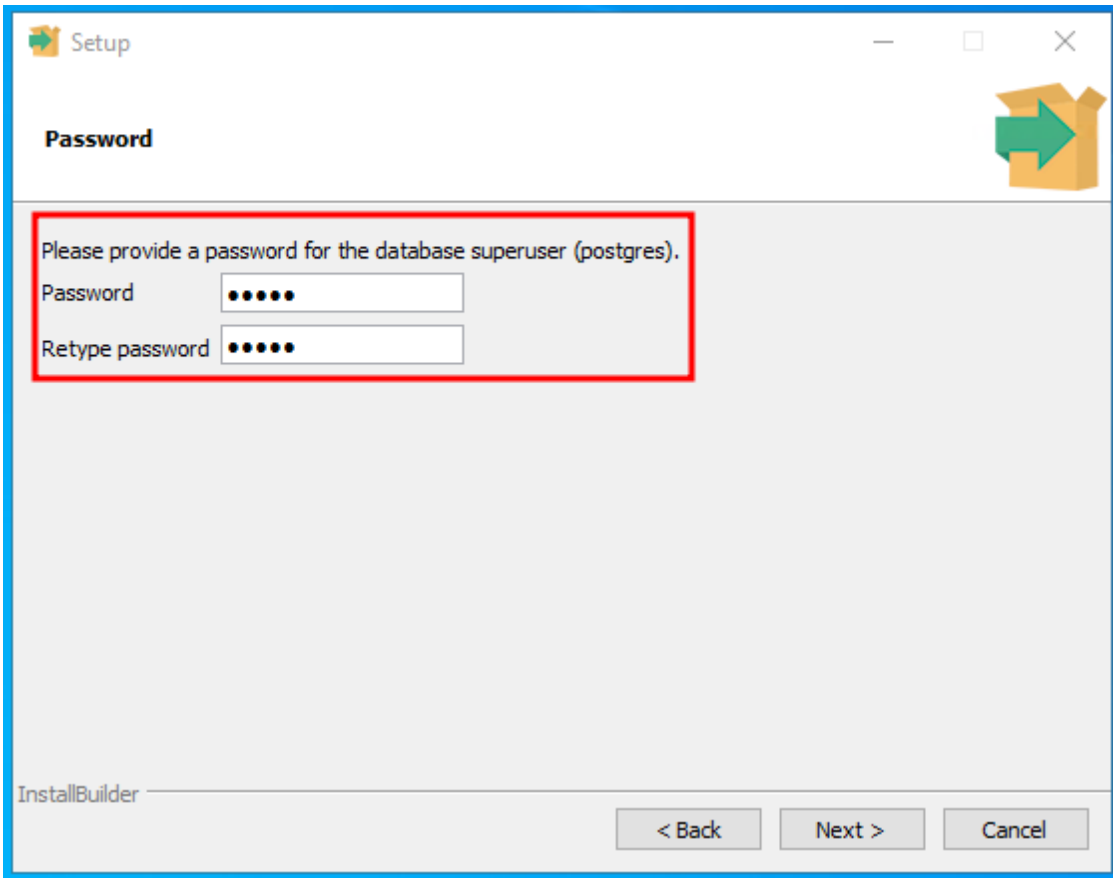
Prerequisites

Before installing the PostgreSQL service, ensure the following requirements are met.

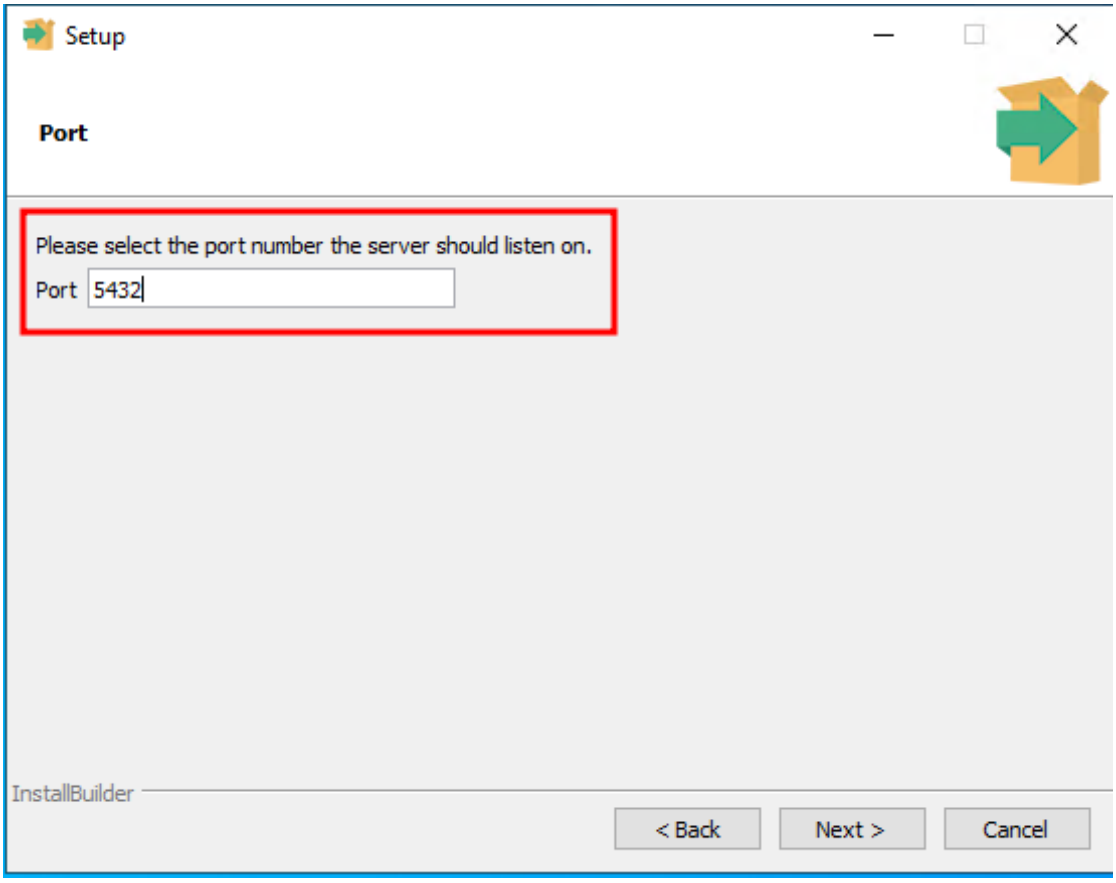
Requirement	Description
Operating System	Windows 11 23H2+, or Windows Server 2019+.
Privileges	Administrator privileges.
Installation package	`postgresql-<version>-windwos-x64.exe`
Hardware	Minimum 8 vCPU and 16 GB RAM.
Disk space	At least 1 TB of available storage on **SSD** or **NVMe** for optimal performance. Approximately 4 GB of storage is required per 1 million scanned objects for a deployment with 5 MD Core instances running 8 AV engines each.
Network access	Required port is open (default port: **5432**). A minimum network bandwidth of **1 Gbps** is required. A bandwidth of **5 Gbps or higher is strongly recommended** for production deployments.

Install PostgreSQL

1. Go to Download PostgreSQL page.
2. Choose and download the Windows x86-64 version.
3. Once the download is complete, run the installer.
4. When prompted, keep the **default installation directory**.
5. Choose the **default components** to install.
6. Set a password for the `postgres` user. Make sure to **save this password** for later use.



7. Select the port. Default is **5432**.



8. Choose the **default locale** for the new database cluster.
9. Complete the installation process.

Open connection

1. Locate the PostgreSQL data directory on your server.

Code · bash

```
C:\Program Files\PostgreSQL\<version>\data\
```

2. To allow remote connections, open the **postgresql.conf** file and configure the following setting:

postgresql.conf · markdown

```
listen_addresses = '*'
```

3. To allow MetaDefender Cluster services to access the database, open the **pg_hba.conf** file and add the following rule.

pg_hba.conf · markdown

```
host    all             all             0.0.0.0/0           scram-sha-256
```

4. Restart the PostgreSQL service to apply the changes.

Linux

> **Info**
> Requires PostgreSQL version 16.9 or later.

Prerequisites

Before installing the PostgreSQL service, ensure the following requirements are met.

Requirement	Description
Operating System	Debian 12+, Ubuntu 22.04+, Ubuntu 24.04+, Rocky Linux 9+, or RHEL 9+
Privileges	`root` or `sudo` privileges.
Hardware	Minimum 8 vCPU and 16 GB RAM.
Disk space	At least 1 TB of available storage on **SSD** or **NVMe** for optimal performance. Approximately 4 GB of storage is required per 1 million scanned objects for a deployment with 5 MD Core instances running 8 AV engines each.
Network access	Required port is open (default port: **5432**). A minimum network bandwidth of **1 Gbps** is required. A bandwidth of **5 Gbps or higher is strongly recommended** for production deployments.

Debian, Ubuntu

Install PostgreSQL

1. Add the official PostgreSQL repository.

Code · bash

```
sudo apt install -y postgresql-common
sudo /usr/share/postgresql-common/pgdg/apt.postgresql.org.sh

# Automated get VERSION_CODENAME
source /etc/os-release
echo "Using VERSION_CODENAME: $VERSION_CODENAME"

# Add PostgreSQL software repository
sudo tee /etc/apt/sources.list.d/pgdg.sources <<EOF
Types: deb deb-src
URIs: https://apt.postgresql.org/pub/repos/apt
Suites: $(echo $VERSION_CODENAME)-pgdg
Architectures: amd64
Components: main
Signed-By: /usr/share/postgresql-common/pgdg/apt.postgresql.org.asc
EOF

sudo apt update
```

2. Install PostgreSQL 16.

Code · bash

```
sudo apt install -y postgresql-16.10
```

3. Install additional supplied modules and extensions.

Code · bash

```
sudo apt install -y postgresql-contrib
```

4. Configures PostgreSQL to start automatically whenever the system boots.

Code · bash

```
sudo systemctl enable --now postgresql
```

5. Setup login password. Make sure to save this password for later use.

Code · bash

```
sudo -u postgres psql -c "ALTER USER postgres WITH PASSWORD '<your_password>';"
```

Open connection

1. Locate the PostgreSQL data directory on your server.

Code · bash

```
/etc/postgresql/<version>/main/
```

2. To allow remote connections, open the **postgresql.conf** file and configure the following setting:

postgresql.conf · markdown

```
listen_addresses = '*'
```

3. To allow MetaDefender Cluster services to access the database, open the `**pg_hba.conf**` file and add the following rule.

pg_hba.conf · markdown

```
host    all             all             0.0.0.0/0            scram-sha-256
```

4. Restart the PostgreSQL service to apply the changes.

Code · bash

```
sudo systemctl restart postgresql
```

Rocky, RHEL

Install PostgreSQL

1. Copy, paste, and run the relevant parts of the setup script:

Code · bash

```
sudo dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-9-x86_64/pgdg-
redhat-repo-latest.noarch.rpm
sudo dnf install -y postgresql16-server
sudo /usr/pgsql-16/bin/postgresql-16-setup initdb
sudo systemctl enable postgresql-16
sudo systemctl start postgresql-16
```

2. Install additional supplied modules and extensions.

Code · bash

```
sudo dnf install -y postgresql16-contrib
```

3. Setup login password. Make sure to save this password for later use.

Code · bash

```
sudo -u postgres psql -c "ALTER USER postgres WITH PASSWORD '<your_password>';"
```

Open connection

1. Locate the PostgreSQL data directory on your server.

Code · bash

```
/var/lib/pgsql/16/data
```

2. To allow remote connections, open the **postgresql.conf** file and configure the following setting:

postgresql.conf · markdown

```
listen_addresses = '*'
```

3. To allow MetaDefender Cluster services to access the database, open the `**pg_hba.conf**` file and add the following rule.

pg_hba.conf · markdown

```
host    all             all             0.0.0.0/0            scram-sha-256
```

4. Restart the PostgreSQL service to apply the changes.

Code · bash

```
sudo systemctl restart postgresql-16
```

Air-gapped

Info

Requires PostgreSQL version 16.9 or later.

Info

To install PostgreSQL in an air-gapped environment, download the required packages on a **preparation machine** and then transfer them to the **air-gapped server**. The preparation machine should have internet access and run the **same operating system version** as the target server.

Prerequisites

Before installing the PostgreSQL service, ensure the following requirements are met.

Requirement	Description
Operating System	Debian 12+, Ubuntu 22.04+, Rocky Linux 9+, or RHEL 9+.
Privileges	`root` or `sudo` privileges on both the preparation machine and the air-gapped server.
Hardware	Minimum 8 vCPU and 16 GB RAM.
Disk space	At least 1 TB of available storage on **SSD** or **NVMe** for optimal performance. Approximately 4 GB of storage is required per 1 million scanned objects for a deployment with 5 MD Core instances running 8 AV engines each.
Network access	Required port is open (default port: **5432**). A minimum network bandwidth of **1 Gbps** is required. A bandwidth of **5 Gbps or higher is strongly recommended** for production deployments.
Offline Package Transfer	A USB drive or other secure transfer medium is required to move the packages to the air-gapped server.

Debian, Ubuntu

Prepare packages

1. On the preparation machine, create file `prepare_postgresql.sh` with content:

Code · bash

```
#!/usr/bin/env bash
#
# Build an offline PostgreSQL install bundle for an air-gapped host.
#
# Dependency resolution: apt resolves the FULL closure as if nothing were
# installed (Dir::State::status -> an empty file), so the bundle is complete
# regardless of what's on this build host - the apt equivalent of dnf's
# --alldeps. The real system is never touched (download-only + a throwaway
# status file).

set -euo pipefail

# ---- configuration ------------------------------------------------------
PG_VERSION="${PG_VERSION:-16}"                    # PostgreSQL major version
INCLUDE_RECOMMENDS="${INCLUDE_RECOMMENDS:-yes}"   # yes = bundle Recommends too

PACKAGES=("postgresql-${PG_VERSION}" "postgresql-contrib")
ARCH="$(dpkg --print-architecture)"
. /etc/os-release
CODENAME="${VERSION_CODENAME:?cannot detect release codename}"
OUTDIR="$(pwd)/postgresql-offline"

# Empty status file = "nothing is installed" -> apt downloads every dependency.
EMPTY_STATUS="$(mktemp)"
trap 'rm -f "$EMPTY_STATUS"' EXIT

echo ">> target: PostgreSQL ${PG_VERSION} | ${ID:-?} ${CODENAME} | ${ARCH}"

# ---- host tooling -------------------------------------------------------
sudo apt-get update
sudo apt-get install -y --no-install-recommends \
  postgresql-common dpkg-dev ca-certificates

# ---- PGDG repository (key + sources, configured once) -------------------
# The official script configures BOTH the signing key and the deb822 sources
# file for the right codename/arch. Don't also hand-write sources.list - doing
# both is what causes "duplicate source" and key-mismatch errors.
echo "" | sudo /usr/share/postgresql-common/pgdg/apt.postgresql.org.sh
sudo apt-get update

# ---- download packages + full dependency closure ------------------------
rm -rf "$OUTDIR"
mkdir -p "$OUTDIR/partial"

REC_FLAG="--no-install-recommends"
[ "$INCLUDE_RECOMMENDS" = "yes" ] && REC_FLAG="--install-recommends"

# Preview the complete closure (every .deb, base libs included).
echo ">> resolving full dependency closure..."
apt-get install --print-uris -y $REC_FLAG \
  -o Dir::State::status="$EMPTY_STATUS" \
```

```
  "${PACKAGES[@]}" \
  | grep -oP "(?<=')[^']*\.deb" > "$OUTDIR/manifest.txt" || true
echo ">> $(wc -l < "$OUTDIR/manifest.txt") package file(s) to download"

# Download them. download-only + the empty status file means the real system
# is never modified, but every dependency is fetched into the bundle.
sudo apt-get install -y --download-only $REC_FLAG \
  -o Dir::State::status="$EMPTY_STATUS" \
  -o Dir::Cache::archives="$OUTDIR" \
  "${PACKAGES[@]}"

# apt leaves lock/partial artifacts behind - drop them and fix ownership.
sudo rm -rf "$OUTDIR/partial" "$OUTDIR/lock"
sudo chown -R "$(id -u):$(id -g)" "$OUTDIR"

# ---- turn the folder into a local apt repo ---------------------------------
# So the air-gapped install becomes a clean `apt-get install` that pulls only
# what the target actually needs (far safer than `dpkg -i *.deb`).
( cd "$OUTDIR" \
    && dpkg-scanpackages -m . /dev/null > Packages \
    && gzip -9c Packages > Packages.gz )

# ---- integrity manifest ----------------------------------------------------
( cd "$OUTDIR" && sha256sum *.deb > SHA256SUMS )

echo
echo ">> done."
echo ">> bundle: $OUTDIR"
echo ">> files:  $(ls "$OUTDIR"/*.deb | wc -l) .deb"
echo ">> size:   $(du -sh "$OUTDIR" | cut -f1)"
```

2. Run commands.

Code · bash

```
chmod +x prepare_postgresql.sh
./prepare_postgresql.sh
```

3. Copy the `postgresql-offline` folder to a USB drive or secure transfer medium.

4. Move it to the air-gapped server.

Install PostgreSQL

1. On the target server, insert the USB drive or secure transfer medium.

2. Copy `postgresql-offline` folder to `/opt/postgresql-offline` . E.g:

Code · bash

```
sudo cp -r postgresql-offline /opt/postgresql-offline
```

3. Run the commands below.

Code · bash

```
# Verify integrity:
cd /opt/postgresql-offline && sha256sum -c SHA256SUMS

# Register it as a local apt repo:
echo "deb [trusted=yes] file:/opt/postgresql-offline ./" \
   | sudo tee /etc/apt/sources.list.d/pg-offline.list

# Update using ONLY this source
sudo apt-get -o Dir::Etc::sourcelist="sources.list.d/pg-offline.list" \
             -o Dir::Etc::sourceparts="/dev/null" update
# Install
sudo apt-get install -y postgresql-16

# Configures PostgreSQL to start automatically whenever the system boots
sudo systemctl enable --now postgresql

# Setup login password
sudo -u postgres psql -c "ALTER USER postgres WITH PASSWORD '<your_password>';"
```

Open connection

1. Locate the PostgreSQL data directory on your server.

Code · bash

```
/etc/postgresql/16/main/
```

2. To allow remote connections, open the **postgresql.conf** file and configure the following setting:

postgresql.conf · markdown

```
listen_addresses = '*'
```

3. To allow MetaDefender Cluster services to access the database, open the **p g_hba.conf** file and add the following rule.

pg_hba.conf · markdown

```
hostssl    all             all             0.0.0.0/0            scram-sha-256
```

4. Restart the PostgreSQL service to apply the changes.

Code · bash

```
sudo systemctl restart postgresql
```

Rocky, RHEL

Prepare packages

1. On the preparation machine, create file `prepare_postgresql.sh` with content:

Code · bash

```
#!/usr/bin/env bash
#
# Build an offline PostgreSQL install bundle for an air-gapped Rocky/RHEL host.
#
# `dnf download --resolve --alldeps` pulls the FULL dependency closure
# regardless of what's installed on this build host, so a clean machine is NOT
# required for completeness. The cost is a larger bundle (it grabs base libs
# too). The local-repo install step keeps that safe: on the target you
# `dnf install <name>` and dnf installs only what's missing - it will NOT
# force-downgrade your base system.

set -euo pipefail

# ---- configuration ------------------------------------------------------
PG_VERSION="${PG_VERSION:-16}"          # PostgreSQL major version
PACKAGES=("postgresql${PG_VERSION}-server" "postgresql${PG_VERSION}-contrib")

ARCH="$(uname -m)"
EL_VERSION="$(rpm -E %{rhel})"
OUTDIR="$(pwd)/postgresql-offline"
REPO_RPM="https://download.postgresql.org/pub/repos/yum/reporpms/EL-${EL_VERSION}-${ARCH}/pgdg-
redhat-repo-latest.noarch.rpm"

echo ">> target: PostgreSQL ${PG_VERSION} | EL${EL_VERSION} | ${ARCH}"

# ---- build-host tooling -------------------------------------------------
sudo dnf install -y dnf-plugins-core createrepo_c

# ---- PGDG repository + disable the built-in AppStream module ------------
sudo dnf install -y "$REPO_RPM"
sudo dnf -qy module disable postgresql

# ---- download packages + full dependency closure ------------------------
```

```
# --resolve : also fetch dependencies
# --alldeps : fetch them even if already installed here (= complete bundle)
# Run as your normal user so the .rpm files aren't root-owned.
rm -rf "$OUTDIR"
mkdir -p "$OUTDIR"
dnf download --resolve --alldeps --downloaddir "$OUTDIR" "${PACKAGES[@]}"

# ---- turn the folder into a local dnf repo ----------------------------------
# So the air-gapped install is a clean `dnf install <name>` with ordering and
# dependency *selection* handled for you. Do NOT `dnf install *.rpm` - that
# tries to install every base lib in the bundle and can downgrade the target.
createrepo_c "$OUTDIR"

# ---- bundle the PGDG GPG key + integrity manifest ---------------------------
cp -f /etc/pki/rpm-gpg/*PGDG* "$OUTDIR"/ 2>/dev/null || true
( cd "$OUTDIR" && sha256sum *.rpm > SHA256SUMS )

echo
echo ">> done."
echo ">> bundle: $OUTDIR"
echo ">> files:  $(ls "$OUTDIR"/*.rpm | wc -l) .rpm"
echo ">> size:   $(du -sh "$OUTDIR" | cut -f1)"
```

2. Run commands.

Code · bash

```
chmod +x prepare_postgresql.sh
./prepare_postgresql.sh
```

3. Copy the postgresql-offline folder to a USB drive or secure transfer medium.

4. Move it to the air-gapped server.

Install PostgreSQL

1. On the target server, insert the USB drive or secure transfer medium.

2. Copy `postgresql-offline` folder to `/opt/postgresql-offline`. E.g:

Code · bash

```
sudo cp -r postgresql-offline /opt/postgresql-offline
```

3. Run the commands below.

Code · bash

```
# Verify integrity:
cd /opt/postgresql-offline && sha256sum -c SHA256SUMS

# Disable the built-in module here too (it conflicts with PGDG)
sudo dnf -qy module disable postgresql

# Install
sudo dnf install -y \
       --disablerepo='*' \
       --repofrompath=pgoffline,/opt/postgresql-offline \
       --setopt=pgoffline.gpgcheck=0 \
       postgresql16-server postgresql16-contrib

# Configures PostgreSQL to start automatically whenever the system boots
sudo /usr/pgsql-16/bin/postgresql-16-setup initdb

# Configures PostgreSQL to start automatically whenever the system boots
sudo systemctl enable --now postgresql-16

# Setup login password
sudo -u postgres psql -c "ALTER USER postgres WITH PASSWORD '<your_password>';"
```

Open connection

1. Locate the PostgreSQL data directory on your server.

Code · bash

```
/var/lib/pgsql/16/data
```

2. To allow remote connections, open the **postgresql.conf** file and configure the following setting:

postgresql.conf · markdown

```
listen_addresses = '*'
```

3. To allow MetaDefender Cluster services to access the database, open the **pg_hba.conf** file and add the following rule.

pg_hba.conf · markdown

```
host    all             all             0.0.0.0/0            scram-sha-256
```

4. Restart the PostgreSQL service to apply the changes.

Code · bash

```
sudo systemctl restart postgresql-16
```

Redis

> **Info**
>
> Requires Redis version 7.2 or later.
>
> The instructions below are based entirely on the Redis documentation or Memurai documentation.

Supported Operating Systems

Server	OS	Version
Memurai	Windows	Windows 11 23H2+, or Windows Server 2019+.
Redis	Windows	WLS - Ubuntu 24.04 (not recommended)
Redis	Linux	Ubuntu 22.04+, Ubuntu 24.04+, Debian 12+, Rocky 9+, or RHEL 9+.

Windows

> **Info**
>
> Requires Redis version 7.2 or later.

Prerequisites

Requirement	Description
Operating System	Windows 11 23H2+, or Windows Server 2019+.
Privileges	Administrator privileges.
Hardware	2 vCPU 32 GB RAM minimum.

Requirement	Description
Disk space	At least 50 GB of available storage.
Network access	Required port is open (default port: **6379**). A minimum network bandwidth of **1 Gbps** is required. A bandwidth of **5 Gbps or higher is strongly recommended** for production deployments.

Windows Subsystem for Linux (WSL)

Install Windows Subsystem

1. Open **PowerShell as Administrator**.
2. Run the following command:

Code · bash

```
wsl --install
```

3. After launching, **Ubuntu** will open automatically and ask you to create a **Linux username and password**.
4. Update the Linux packages.

Install Redis

1. Add the repository to the APT index, update it, and install Redis:

Code · bash

```
# add repository
sudo apt-get install lsb-release curl gpg
curl -fsSL https://packages.redis.io/gpg | sudo gpg --dearmor -o /usr/share/keyrings/redis-archive-keyring.gpg
sudo chmod 644 /usr/share/keyrings/redis-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/redis-archive-keyring.gpg] https://packages.redis.io/deb $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/redis.list
sudo apt-get update

# install Redis
sudo apt-get install \
            redis=6:8.6.4-1rl1~noble1 \
            redis-server=6:8.6.4-1rl1~noble1 \
            redis-tools=6:8.6.4-1rl1~noble1
```

```
sudo systemctl enable --now redis-server

# check redis-server status
sudo systemctl status redis-server
```

2. Access Redis configuration file `/etc/redis/redis.conf` for editing.

3. Comment out the `bind` setting and set `protected-mode` option to **no**.

Code · bash

```
...
# The following line should be commented
# bind 127.0.0.1
...
# The following line should be uncommented and set to no
protected-mode no
...
```

3. Restart Redis

Code · bash

```
sudo systemctl restart redis-server
```

4. Run the command with Redis CLI

Code · bash

```
redis-cli ping
```

5. Confirm that the response returned is PONG

Code

```
PONG
```

Memurai

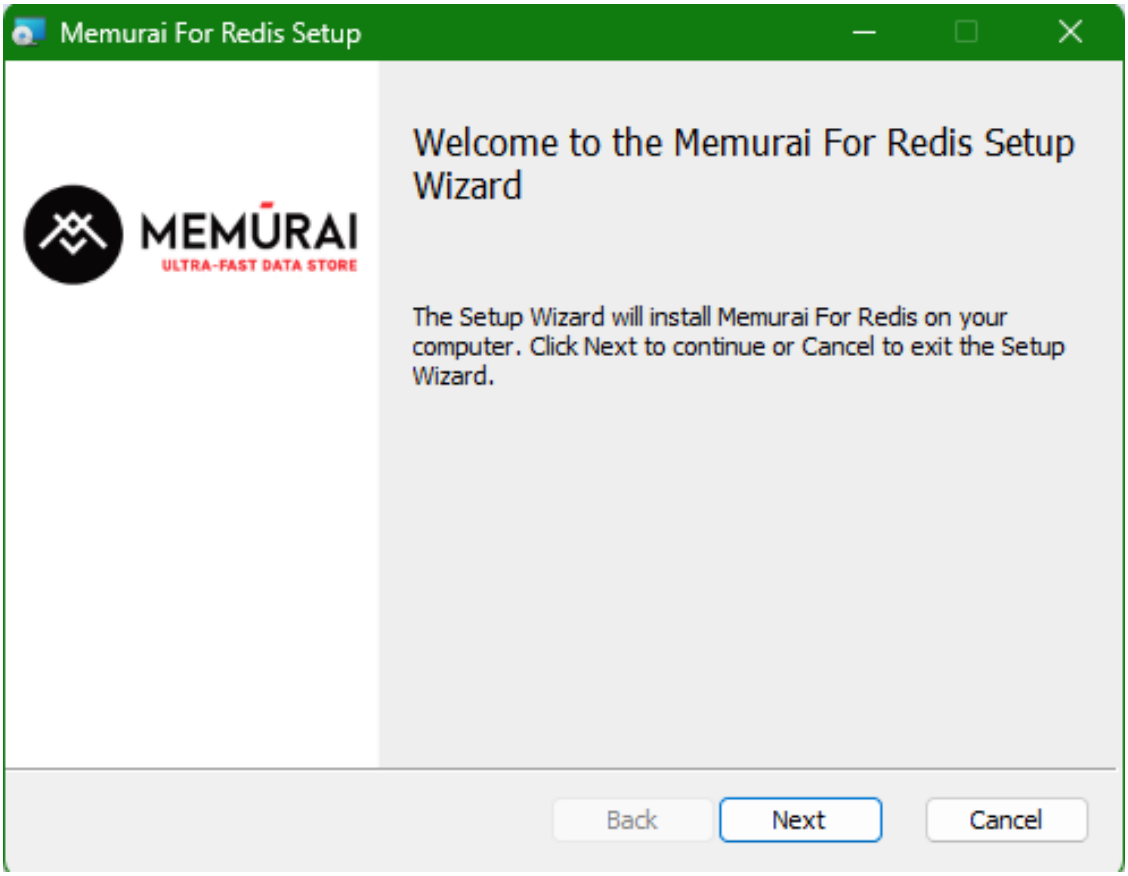
Download and install

1. Go to the official website: https://www.memurai.com/get-memurai

2. Choose the appropriate version.

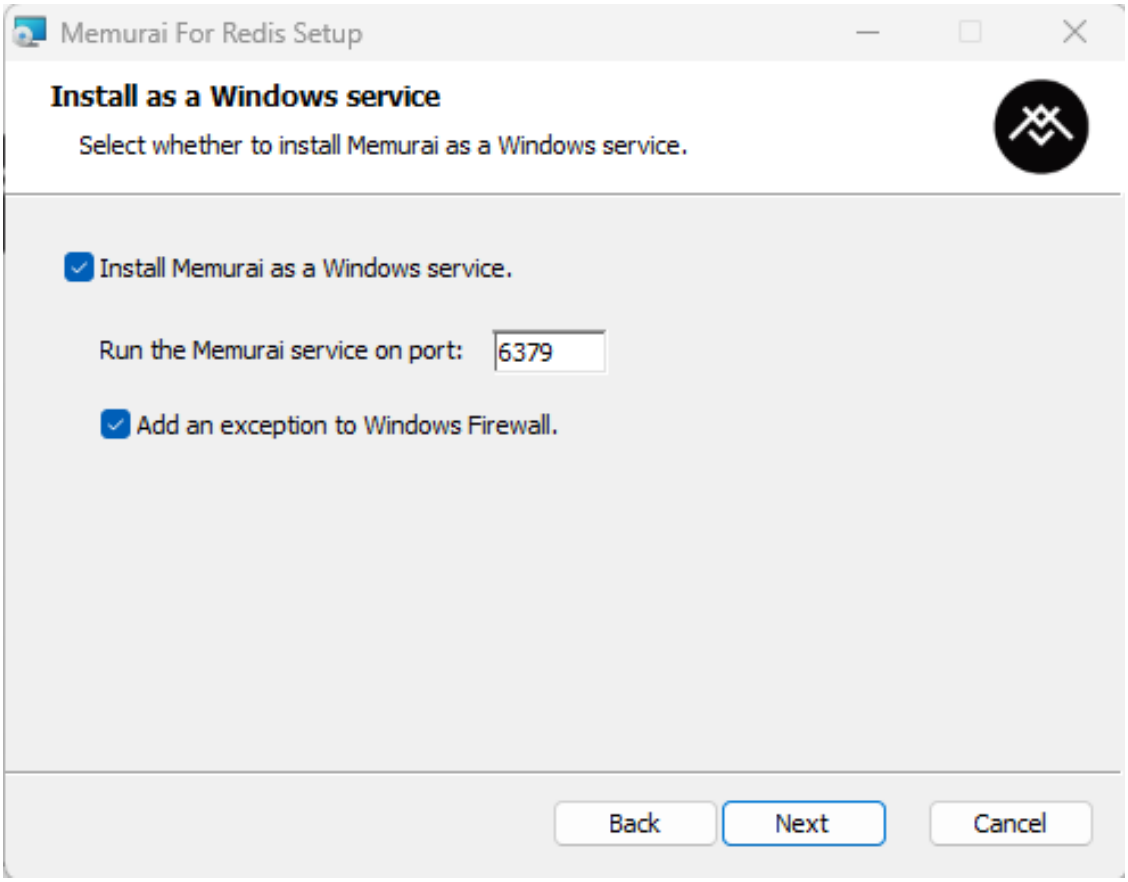
3. Download the Windows installer (.msi).

4. Double-click the downloaded .msi file.



5. Click `Next` and follow the installation wizard.

- Accept the license agreement.
- Choose the installation directory.
- Select port (default is 6379).
- Click `Install` to start installation process.



Verify Memurai installation

1. Open **Command Prompt** or **PowerShell**.
2. Run the command with Memurai CLI:

```bash
memurai-cli ping
```

3. Confirm that the response returned is **PONG**.

```bash
PONG
```

Configure Memurai

1. Access the Memurai configuration file in the installation directory, for example: `C:\Program Files\Memurai\memurai.conf`.
2. Comment out the `bind` setting and set `protected-mode` option to **no**.

```
memurai.conf
```

```
...
# The following line should be commented
# bind 127.0.0.1
...
# The following line should be uncommented and set to no
protected-mode no
...
```

3. Restart Memurai service.

Linux

> **Info**
>
> Requires Redis version 7.2 or later.

Prerequisites

Before installing the Redis service, ensure the following requirements are met.

Requirement	Description
Operating System	Debian 12+, Ubuntu 22.04+, Ubuntu 24.04+, Rocky Linux 9+, or RHEL 9+
Privileges	`root` or `sudo` privileges.
Hardware	2 vCPU 32 GB RAM minimum.
Disk space	At least 50 GB of available storage.
Network access	Required port is open (default port: **6379**). A minimum network bandwidth of **1 Gbps** is required. A bandwidth of **5 Gbps or higher is strongly recommended** for production deployments.

Debian/Ubuntu

1. Add the repository to the APT index, update it, and install Redis:

Code · bash

```
# add repository
sudo apt-get install lsb-release curl gpg

# get version code name. e.g: noble, jammy, ..
VERSION_CODENAME=$(lsb_release -cs)

curl -fsSL https://packages.redis.io/gpg | sudo gpg --dearmor -o /usr/share/keyrings/redis-
archive-keyring.gpg
sudo chmod 644 /usr/share/keyrings/redis-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/redis-archive-keyring.gpg] https://packages.redis.io/
deb $VERSION_CODENAME main" | sudo tee /etc/apt/sources.list.d/redis.list
sudo apt-get update

# install Redis
sudo apt-get install \
            redis=6:8.6.4-1rl1~${VERSION_CODENAME}1 \
            redis-server=6:8.6.4-1rl1~${VERSION_CODENAME}1 \
            redis-tools=6:8.6.4-1rl1~${VERSION_CODENAME}1
sudo systemctl enable --now redis-server

# check redis-server status
sudo systemctl status redis-server
```

2. Access Redis configuration file `/etc/redis/redis.conf` for editing.

3. Comment out the `bind` setting and set `protected-mode` option to **no**.

Code · bash

```
...
# The following line should be commented
# bind 127.0.0.1
...
# The following line should be uncommented and set to no
protected-mode no
...
```

4. Restart Redis.

Code · bash

```
sudo systemctl restart redis-server
```

5. Run the command with Redis CLI.

Code · bash

```
redis-cli ping
```

6. Confirm that the response returned is PONG.

```
Code

PONG
```

Rocky, RHEL

1. Create the file `/etc/yum.repos.d/redis.repo` with the following contents.

```
Code

[Redis]
name=Redis
baseurl=http://packages.redis.io/rpm/rockylinux9
enabled=1
gpgcheck=1
```

2. Run the commands.

```
Code · bash

curl -fsSL https://packages.redis.io/gpg > /tmp/redis.key
sudo rpm --import /tmp/redis.key
sudo dnf install -y redis-8.6.4-1.x86_64
sudo systemctl enable --now redis
sudo systemctl status redis
```

3. Access Redis configuration file `/etc/redis/redis.conf` for editing.

4. Comment out the `bind` setting and set `protected-mode` option to **no**.

```
Code · bash

...
# The following line should be commented
# bind 127.0.0.1
...
# The following line should be uncommented and set to no
protected-mode no
...
```

4. Restart Redis.

```bash
sudo systemctl restart redis
```

5. Run the command with Redis CLI.

```bash
redis-cli ping
```

6. Confirm that the response returned is PONG.

```
PONG
```

Air-gapped

> **Info**
> Requires Redis version 7.2 or later.

> **Info**
> To install Redis in an air-gapped environment, download the required packages on a **preparation machine** and then transfer them to the **air-gapped server**. The preparation machine should have internet access and run the **same operating system version** as the target server.

Prerequisites

Before installing the Redis service, ensure the following requirements are met.

Requirement	Description
Operating System	Debian 12+, Ubuntu 22.04+, Rocky Linux 9+, or RHEL 9+.
Privileges	`root` or `sudo` privileges on both the preparation machine and the air-gapped server.

Requirement	Description
Hardware	2 vCPU 32 GB RAM minimum.
Disk space	At least 50 GB of available storage.
Network access	Required port is open (default port: **6379**). A minimum network bandwidth of **1 Gbps** is required. A bandwidth of **5 Gbps or higher is strongly recommended** for production deployments.
Offline Package Transfer	A USB drive or other secure transfer medium is required to move the packages to the air-gapped server.

Debian, Ubuntu

Prepare packages

1. On the preparation machine, create file `prepare_redis.sh` with content:

Code · bash

```
#!/usr/bin/env bash
#
# Build an offline Redis install bundle for an air-gapped Ubuntu/Debian host.
#
# Dependency resolution: apt resolves the FULL closure as if nothing were
# installed (Dir::State::status -> an empty file), so the bundle is complete
# regardless of what's on this build host - the apt equivalent of dnf's
# --alldeps. The real system is never touched (download-only + throwaway status).

set -euo pipefail

# ---- configuration ------------------------------------------------------
# Pin the UPSTREAM Redis version (e.g. 8.6.4) - REQUIRED.
# The full apt version string - epoch, Debian revision and the codename suffix
# like "~noble1" - is detected automatically from the repo for THIS release,
# so the same script works unchanged on jammy / bookworm / etc.
REDIS_UPSTREAM="${REDIS_UPSTREAM:-8.6.4}"
INCLUDE_RECOMMENDS="${INCLUDE_RECOMMENDS:-yes}"   # yes = also bundle Recommends
PKGS=(redis redis-server redis-tools)

ARCH="$(dpkg --print-architecture)"
```

```
. /etc/os-release
CODENAME="${VERSION_CODENAME:?cannot detect release codename}"
OUTDIR="$(pwd)/redis-offline"

# Empty status file = "nothing is installed" -> apt downloads every dependency.
EMPTY_STATUS="$(mktemp)"
trap 'rm -f "$EMPTY_STATUS"' EXIT

echo ">> target: Redis ${REDIS_UPSTREAM:-latest} | ${ID:-?} ${CODENAME} | ${ARCH}"

# ---- build-host tooling + Redis repository ---------------------------------
sudo apt-get update
sudo apt-get install -y --no-install-recommends curl gpg ca-certificates dpkg-dev

curl -fsSL https://packages.redis.io/gpg \
  | sudo gpg --dearmor --yes -o /usr/share/keyrings/redis-archive-keyring.gpg
sudo chmod 644 /usr/share/keyrings/redis-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/redis-archive-keyring.gpg] https://packages.redis.io/
deb ${CODENAME} main" \
  | sudo tee /etc/apt/sources.list.d/redis.list >/dev/null
sudo apt-get update

# ---- auto-detect the exact, codename-correct version -----------------------
# packages.redis.io serves a per-codename repo, and Ubuntu's own universe also
# ships a (different, older) redis-server, so we match BOTH the upstream version
# and the "~<codename>" suffix to land on the right .deb. madison lists newest
# first, so head -n1 wins.
SPECS=()
if [ -n "$REDIS_UPSTREAM" ]; then
  esc="$(printf '%s' "$REDIS_UPSTREAM" | sed 's/[.[\*^$]/\\&/g')"
  REDIS_VERSION="$(
    apt-cache madison redis-server \
      | awk -F'|' '{gsub(/ /,"",$2); print $2}' \
      | grep -E "(^|:)${esc}-" \
      | grep -F "~${CODENAME}" \
      | head -n1
  )"
  if [ -z "$REDIS_VERSION" ]; then
    echo "ERROR: no Redis ${REDIS_UPSTREAM} build found for '${CODENAME}'." >&2
    echo "Available versions:" >&2
    apt-cache madison redis-server | awk -F'|' '{print "    " $2}' >&2
    exit 1
  fi
  echo ">> resolved version: ${REDIS_VERSION}"
  for p in "${PKGS[@]}"; do SPECS+=("${p}=${REDIS_VERSION}"); done
else
  echo "ERROR: REDIS_UPSTREAM is required" >&2
  exit 1
fi
```

```
REC_FLAG="--no-install-recommends"
[ "$INCLUDE_RECOMMENDS" = "yes" ] && REC_FLAG="--install-recommends"

# ---- download packages + full dependency closure ----------------------------
rm -rf "$OUTDIR"
mkdir -p "$OUTDIR/partial"

# Preview the complete closure (every .deb, base libs included).
echo ">> resolving full dependency closure..."
apt-get install --print-uris -y $REC_FLAG \
  -o Dir::State::status="$EMPTY_STATUS" \
  "${SPECS[@]}" \
  | grep -oP "(?<=')[^']*\.deb" > "$OUTDIR/manifest.txt" || true
echo ">> $(wc -l < "$OUTDIR/manifest.txt") package file(s) to download"

# download-only + the empty status file: every dependency is fetched, but the
# real system is never modified.
sudo apt-get install -y --download-only $REC_FLAG \
  -o Dir::State::status="$EMPTY_STATUS" \
  -o Dir::Cache::archives="$OUTDIR" \
  "${SPECS[@]}"

sudo rm -rf "$OUTDIR/partial" "$OUTDIR/lock"
sudo chown -R "$(id -u):$(id -g)" "$OUTDIR"

# ---- turn the folder into a local apt repo ----------------------------------
# So the air-gapped install becomes a clean `apt-get install` that pulls only
# what the target actually needs (far safer than `dpkg -i *.deb`).
( cd "$OUTDIR" \
    && dpkg-scanpackages -m . /dev/null > Packages \
    && gzip -9c Packages > Packages.gz )

# ---- integrity manifest -----------------------------------------------------
( cd "$OUTDIR" && sha256sum *.deb > SHA256SUMS )

echo
echo ">> done."
echo ">> bundle: $OUTDIR"
echo ">> files:  $(ls "$OUTDIR"/*.deb | wc -l) .deb"
echo ">> size:   $(du -sh "$OUTDIR" | cut -f1)"
```

2. Run commands.

Code · bash

```
chmod +x prepare_redis.sh
./prepare_redis.sh
```

3. Copy the `redis-offline` folder to a USB drive or secure transfer medium.

4. Move it to the air-gapped server.

Install Redis

1. On the target server, insert the USB drive or secure transfer medium.
2. Copy `redis-offline` folder to `/opt/redis-offline` . E.g:

Code · bash

```
sudo cp -r redis-offline /opt/redis-offline
```

2. Run the commands below.

Code · bash

```
# Verify integrity:
cd /opt/redis-offline && sha256sum -c SHA256SUMS

# Register
echo "deb [trusted=yes] file:/opt/redis-offline ./" | sudo tee /etc/apt/sources.list.d/redis-
offline.list

# Update using ONLY this source
sudo apt-get -o Dir::Etc::sourcelist="sources.list.d/redis-offline.list" -o
Dir::Etc::sourceparts="/dev/null" update

# Install Redis.
sudo apt-get install -y redis-server redis-tools
sudo systemctl enable --now redis-server
```

3. Access Redis configuration file /etc/redis/redis.conf for editing.
4. Comment out the bind setting and set protected-mode option to no.

Code · bash

```
...
# The following line should be commented
# bind 127.0.0.1
...
# The following line should be uncommented and set to no
protected-mode no
...
```

5. Restart Redis.

Code · bash

```
sudo systemctl restart redis-server
```

6. Run the command with Redis CLI.

Code · bash

```
redis-cli ping
```

7. Confirm that the response returned is PONG.

Code

```
PONG
```

Rocky, RHEL

Prepare packages

1. On the preparation machine, create file `prepare_redis.sh` with content:

Code · bash

```
#!/usr/bin/env bash
#
# Build an offline Redis install bundle for an air-gapped Rocky/RHEL host.
#
# Note: `dnf download --resolve --alldeps` pulls the FULL dependency closure
# regardless of what's installed on this build host, so a clean machine is NOT
# required for completeness. The cost is a larger bundle (it grabs base libs
# too). The local-repo install step below keeps that safe: on the target you
# `dnf install <name>` and dnf installs only what's missing - it will NOT
# force-downgrade your base system.

set -euo pipefail

# ---- configuration ------------------------------------------------------
# Pin the UPSTREAM Redis version (e.g. 8.6.4). Leave empty for the latest.
# The release tag (".el9") and arch are filled in automatically for THIS host.
REDIS_VERSION="${REDIS_VERSION:-8.6.4}"

ARCH="$(uname -m)"
EL_VERSION="$(rpm -E %{rhel})"
OUTDIR="$(pwd)/redis-offline"
```

```
echo ">> target: Redis ${REDIS_VERSION:-latest} | EL${EL_VERSION} | ${ARCH}"

# ---- build-host tooling -------------------------------------------------
sudo dnf install -y dnf-plugins-core createrepo_c

# ---- Redis repository + GPG key -----------------------------------------
# baseurl tracks the EL major version; gpgkey lets dnf verify on its own.
sudo tee /etc/yum.repos.d/redis.repo >/dev/null <<EOF
[redis]
name=Redis
baseurl=https://packages.redis.io/rpm/rockylinux${EL_VERSION}
enabled=1
gpgcheck=1
gpgkey=https://packages.redis.io/gpg
EOF

curl -fsSL https://packages.redis.io/gpg -o /tmp/redis.key
sudo rpm --import /tmp/redis.key
sudo dnf makecache || true

# ---- resolve the exact, EL-correct NEVRA ---------------------------------
# (release tag like "1.el9" and the arch are appended automatically; newest
#  matching release wins). On EL8 you'd also: sudo dnf module disable redis
SPEC="redis"
if [ -n "$REDIS_VERSION" ]; then
  esc="$(printf '%s' "$REDIS_VERSION" | sed 's/[.[\*^$]/\\&/g')"
  SPEC="$(
    dnf repoquery --quiet --repo=redis --arch="${ARCH},noarch" \
        --queryformat '%{name}-%{version}-%{release}.%{arch}' redis 2>/dev/null \
      | grep -E -- "-${esc}-" \
      | sort -V | tail -n1
  )"
  if [ -z "$SPEC" ]; then
    echo "ERROR: no Redis ${REDIS_VERSION} build found for EL${EL_VERSION}/${ARCH}." >&2
    echo "Available redis versions in the repo:" >&2
    dnf repoquery --quiet --repo=redis redis \
        --queryformat '    %{version}-%{release}.%{arch}' >&2
    exit 1
  fi
  echo ">> resolved: ${SPEC}"
fi

# ---- download package + full dependency closure --------------------------
# --resolve : also fetch dependencies
# --alldeps : fetch them even if already installed here (= complete bundle)
rm -rf "$OUTDIR"
mkdir -p "$OUTDIR"
dnf download --resolve --alldeps --downloaddir "$OUTDIR" "$SPEC"

# ---- turn the folder into a local dnf repo -------------------------------
# So the air-gapped install is a clean `dnf install <name>` with ordering and
```

```
# dependency *selection* handled for you. Do NOT `dnf install *.rpm` - that
# tries to install every base lib in the bundle and can downgrade the target.
createrepo_c "$OUTDIR"

# ---- bundle the GPG key + integrity manifest -------------------------------
cp -f /tmp/redis.key "$OUTDIR/RPM-GPG-KEY-redis"
( cd "$OUTDIR" && sha256sum *.rpm > SHA256SUMS )

echo
echo ">> done."
echo ">> bundle: $OUTDIR"
echo ">> files:  $(ls "$OUTDIR"/*.rpm | wc -l) .rpm"
echo ">> size:   $(du -sh "$OUTDIR" | cut -f1)"
```

2. Run commands.

Code · bash

```
chmod +x prepare_redis.sh
./prepare_redis.sh
```

3. Copy the redis folder to a USB drive or secure transfer medium.

4. Move it to the air-gapped server.

Install Redis

1. On the target server, insert the USB drive or secure transfer medium.

2. Copy `redis-offline` folder to `/opt/redis-offline`. E.g:

Code · bash

```
sudo cp -r redis-offline /opt/redis-offline
```

3. Run the commands below.

Code · bash

```
# Verify integrity
cd /opt/redis-offline && sha256sum -c SHA256SUMS

# Install from the local folder as a temporary repo
sudo dnf install -y \
     --disablerepo='*' \
     --repofrompath=redisoffline,/opt/redis-offline \
     --setopt=redisoffline.gpgcheck=0 \
```

```
      redis
```

3. Access Redis configuration file `/etc/redis/redis.conf` for editing.

4. Comment out the `bind` setting and set `protected-mode` option to **no**.

Code · bash

```
...
# The following line should be commented
# bind 127.0.0.1
...
# The following line should be uncommented and set to no
protected-mode no
...
```

3. Restart Redis.

Code · bash

```
sudo systemctl restart redis
```

4. Run the command with Redis CLI.

Code · bash

```
redis-cli ping
```

5. Confirm that the response returned is PONG.

Code

```
PONG
```

RabbitMQ

Info

Requires RabbitMQ version 4.2 or later.

RabbitMQ functions effectively only with specific supported versions of Erlang. Please refer to the link for the Erlang-RabbitMQ compatibility matrix.

The instructions below are based entirely on the RabbitMQ documentation. The following installation guidelines describe how to install RabbitMQ 4.2.5 with Erlang/OTP 27.3.4.9

Supported Operating Systems

OS	Version
Windows	Windows 11 23H2+, or Windows Server 2019+.
Linux	Ubuntu 22.04+, Ubuntu 24.04+, Debian 12+, Debian 13+, Rocky 9+, or RHEL 9+.

Windows

Info

Requires RabbitMQ version 4.2 or later.

Please refer to the link for the Erlang-RabbitMQ compatibility matrix.

Prerequisites

Before installing the RabbitMQ service, ensure the following requirements are met.

Requirement	Description
Operating System	Windows 11 23H2+, or Windows Server 2019+.
Privileges	Administrator privileges.
Hardware	4 vCPU and 8 GB RAM.
Disk space	At least 100 GB of available storage.
Network access	Required ports are open: • **5672** — AMQP client connections (default). • **15672** — Management UI (optional). A minimum network bandwidth of **1 Gbps** is required.

Install Erlang/OTP

1. Download Erlang/OTP 26.2.5.18.
2. Run the downloaded installer.

Install RabbitMQ

1. Download RabbitMQ 4.2.5.
2. Run installer.
3. Pick an install directory.
4. In Command Prompt, run the command as **Administrator** to set up user.

Code

```
cd <RabbitMQ_sbin_dir> # default: "C:\Program Files\RabbitMQ Server\rabbitmq_server-4.2.5\sbin"
rabbitmqctl.bat add_user <username> <password>
rabbitmqctl.bat set_permissions -p / <username> "." "." "."
rabbitmqctl.bat set_user_tags <username> administrator
```

Linux

Info

Requires RabbitMQ version 4.2 or later.

Please refer to the link for the Erlang-RabbitMQ compatibility matrix.

Prerequisites

Before installing the RabbitMQ service, ensure the following requirements are met.

Requirement	Description
Operating System	Debian 12+, Ubuntu 22.04+, Ubuntu 24.04+, Rocky Linux 9+, or RHEL 9+
Privileges	`root` or `sudo` privileges.
Hardware	4 vCPU and 8 GB RAM.
Disk space	At least 100 GB of available storage.
Network access	Required ports are open: • **5672** — AMQP client connections (default). • **15672** — Management UI (optional). A minimum network bandwidth of **1 Gbps** is required.

Requirement	Description

Ubuntu, Debian

Install Dependencies

Code · bash

```
sudo apt-get update -y
sudo apt-get install curl gnupg apt-transport-https wget -y
sudo apt-get install logrotate init-system-helpers adduser -y
```

Add Repository Signing Key

Code · bash

```
## Team RabbitMQ's signing key
curl -1sLf "https://keys.openpgp.org/vks/v1/by-
fingerprint/0A9AF2115F4687BD29803A206B73A36E6026DFCA" | sudo gpg --dearmor | sudo tee /usr/
share/keyrings/com.rabbitmq.team.gpg > /dev/null
```

Install Erlang/OTP

1. Add a repository file.

- Ubuntu 24.4

Code · bash

```
sudo tee /etc/apt/sources.list.d/rabbitmq.list <<EOF
## Modern Erlang/OTP releases
##
deb [arch=amd64 signed-by=/usr/share/keyrings/com.rabbitmq.team.gpg] https://deb1.rabbitmq.com/
rabbitmq-erlang/ubuntu/noble noble main
deb [arch=amd64 signed-by=/usr/share/keyrings/com.rabbitmq.team.gpg] https://deb2.rabbitmq.com/
rabbitmq-erlang/ubuntu/noble noble main

## Provides modern RabbitMQ releases
##
deb [arch=amd64 signed-by=/usr/share/keyrings/com.rabbitmq.team.gpg] https://deb1.rabbitmq.com/
rabbitmq-server/ubuntu/noble noble main
deb [arch=amd64 signed-by=/usr/share/keyrings/com.rabbitmq.team.gpg] https://deb2.rabbitmq.com/
rabbitmq-server/ubuntu/noble noble main
EOF
```

- Debian Bookworm

Code · bash

```
sudo tee /etc/apt/sources.list.d/rabbitmq.list <<EOF
## Modern Erlang/OTP releases
##
deb [arch=amd64 signed-by=/usr/share/keyrings/com.rabbitmq.team.gpg] https://deb1.rabbitmq.com/
rabbitmq-erlang/debian/bookworm bookworm main
deb [arch=amd64 signed-by=/usr/share/keyrings/com.rabbitmq.team.gpg] https://deb2.rabbitmq.com/
rabbitmq-erlang/debian/bookworm bookworm main

## Provides modern RabbitMQ releases
##
deb [arch=amd64 signed-by=/usr/share/keyrings/com.rabbitmq.team.gpg] https://deb1.rabbitmq.com/
rabbitmq-server/debian/bookworm bookworm main
deb [arch=amd64 signed-by=/usr/share/keyrings/com.rabbitmq.team.gpg] https://deb2.rabbitmq.com/
rabbitmq-server/debian/bookworm bookworm main
EOF
```

2. After updating the list of `apt` sources, it is necessary to run `apt-get update` .

Code · bash

```
sudo apt-get update -y
```

3. Install Erlang.

Code · bash

```
## Install Erlang packages
sudo apt-get install -y erlang-base \
                        erlang-asn1 \
                        erlang-crypto \
                        erlang-eldap \
                        erlang-ftp \
                        erlang-inets \
                        erlang-mnesia \
                        erlang-os-mon \
                        erlang-parsetools \
                        erlang-public-key \
                        erlang-runtime-tools \
                        erlang-snmp \
                        erlang-ssl \
                        erlang-syntax-tools \
                        erlang-tftp \
                        erlang-tools \
                        erlang-xmerl
```

Install RabbitMQ

1. Run the following commands.

Code · bash

```
# download the package
sudo apt-get -y install wget
sudo wget https://github.com/rabbitmq/rabbitmq-server/releases/download/v4.2.5/rabbitmq-server_4.2.5-1_all.deb

# install the package with dpkg
sudo dpkg -i rabbitmq-server_4.2.5-1_all.deb

sudo rm rabbitmq-server_4.2.5-1_all.deb

sudo systemctl enable --now rabbitmq-server
sudo systemctl status  rabbitmq-server
```

2. Set up user.

Code · bash

```
sudo rabbitmqctl add_user <username> <password>
sudo rabbitmqctl set_permissions -p / <username> "." "." "."
sudo rabbitmqctl set_user_tags <username> administrator
```

Rocky, RHEL

Install Dependencies

Code · bash

```
sudo dnf install -y logrotate wget
```

Import the Signing Keys

Code · bash

```
## primary RabbitMQ signing key
sudo rpm --import 'https://github.com/rabbitmq/signing-keys/releases/download/3.0/rabbitmq-release-signing-key.asc'
## modern Erlang repository
sudo rpm --import 'https://github.com/rabbitmq/signing-keys/releases/download/3.0/
```

```
cloudsmith.rabbitmq-erlang.E495BB49CC4BBE5B.key'
## RabbitMQ server repository
sudo rpm --import 'https://github.com/rabbitmq/signing-keys/releases/download/3.0/
cloudsmith.rabbitmq-server.9F4587F226208342.key'
```

Add Yum Repositories

Create the file `/etc/yum.repos.d/rabbitmq.repo` with the following content.

Code · bash

```
# In /etc/yum.repos.d/rabbitmq.repo

##
## Zero dependency Erlang RPM
##

[modern-erlang]
name=modern-erlang-el9
# Use a set of mirrors maintained by the RabbitMQ core team.
# The mirrors have significantly higher bandwidth quotas.
baseurl=https://yum1.rabbitmq.com/erlang/el/9/$basearch
        https://yum2.rabbitmq.com/erlang/el/9/$basearch
repo_gpgcheck=1
enabled=1
gpgkey=https://github.com/rabbitmq/signing-keys/releases/download/3.0/cloudsmith.rabbitmq-
erlang.E495BB49CC4BBE5B.key
gpgcheck=1
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300
pkg_gpgcheck=1
autorefresh=1
type=rpm-md

[modern-erlang-noarch]
name=modern-erlang-el9-noarch
# Use a set of mirrors maintained by the RabbitMQ core team.
# The mirrors have significantly higher bandwidth quotas.
baseurl=https://yum1.rabbitmq.com/erlang/el/9/noarch
        https://yum2.rabbitmq.com/erlang/el/9/noarch
repo_gpgcheck=1
enabled=1
gpgkey=https://github.com/rabbitmq/signing-keys/releases/download/3.0/cloudsmith.rabbitmq-
erlang.E495BB49CC4BBE5B.key
       https://github.com/rabbitmq/signing-keys/releases/download/3.0/rabbitmq-release-signing-
key.asc
gpgcheck=1
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

```
metadata_expire=300
pkg_gpgcheck=1
autorefresh=1
type=rpm-md


##
## RabbitMQ Server
##

[rabbitmq-el9]
name=rabbitmq-el9
baseurl=https://yum2.rabbitmq.com/rabbitmq/el/9/$basearch
        https://yum1.rabbitmq.com/rabbitmq/el/9/$basearch
repo_gpgcheck=1
enabled=1
# Cloudsmith's repository key and RabbitMQ package signing key
gpgkey=https://github.com/rabbitmq/signing-keys/releases/download/3.0/cloudsmith.rabbitmq-
server.9F4587F226208342.key
       https://github.com/rabbitmq/signing-keys/releases/download/3.0/rabbitmq-release-signing-
key.asc
gpgcheck=1
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300
pkg_gpgcheck=1
autorefresh=1
type=rpm-md

[rabbitmq-el9-noarch]
name=rabbitmq-el9-noarch
baseurl=https://yum2.rabbitmq.com/rabbitmq/el/9/noarch
        https://yum1.rabbitmq.com/rabbitmq/el/9/noarch
repo_gpgcheck=1
enabled=1
# Cloudsmith's repository key and RabbitMQ package signing key
gpgkey=https://github.com/rabbitmq/signing-keys/releases/download/3.0/cloudsmith.rabbitmq-
server.9F4587F226208342.key
       https://github.com/rabbitmq/signing-keys/releases/download/3.0/rabbitmq-release-signing-
key.asc
gpgcheck=1
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300
pkg_gpgcheck=1
autorefresh=1
type=rpm-md
```

Install Erlang

Code · bash

```
sudo dnf update -y
sudo dnf install -y erlang
```

Install RabbitMQ

1. Run the following commands.

Code · bash

```
rpm --import https://github.com/rabbitmq/signing-keys/releases/download/3.0/rabbitmq-release-signing-key.asc

sudo wget https://github.com/rabbitmq/rabbitmq-server/releases/download/v4.2.5/rabbitmq-server-4.2.5-1.el8.noarch.rpm

sudo dnf install -y rabbitmq-server-4.2.5-1.el8.noarch.rpm
sudo rm abbitmq-server-4.2.5-1.el8.noarch.rpm

sudo systemctl enable --now rabbitmq-server
sudo systemctl status  rabbitmq-server
```

2. Set up user.

Code · bash

```
sudo rabbitmqctl add_user <username> <password>
sudo rabbitmqctl set_permissions -p / <username> "." "." "."
sudo rabbitmqctl set_user_tags <username> administrator
```

Air-gapped

Info

Requires RabbitMQ version 4.2 or later, along with a compatible Erlang version.

Info

To install Redis in an air-gapped environment, download the required packages on a **preparation machine** and then transfer them to the **air-gapped server**. The preparation machine should have internet access and run the **same operating system version** as the target server.

Prerequisites

Before installing the RabbitMQ service, ensure the following requirements are met.

Requirement	Description
Operating System	Debian 12+, Ubuntu 22.04+, Rocky Linux 9+, or RHEL 9+.
Privileges	`root` or `sudo` privileges on both the preparation machine and the air-gapped server.
Hardware	4 vCPU and 8 GB RAM.
Disk space	At least 100 GB of available storage.
Network access	Required ports are open: • **5672** — AMQP client connections (default). • **15672** — Management UI (optional). A minimum network bandwidth of **1 Gbps** is required. A bandwidth of **5 Gbps or higher is strongly recommended** for production deployments.

Debian, Ubuntu

Prepare packages

1. On the preparation machine, create file `prepare_rabbitmq.sh` with content:

Code · bash

```
#!/usr/bin/env bash
#
# Build an offline RabbitMQ (+ Erlang) install bundle for an air-gapped
# Ubuntu/Debian host.
#
# Dependency resolution: apt resolves the FULL closure as if nothing were
# installed (Dir::State::status -> an empty file), so every Erlang module AND
# its base-library dependencies are fetched regardless of what's on this build
# host - the apt equivalent of dnf's --alldeps. The real system is never
# touched (download-only + throwaway status file).

set -euo pipefail

# ---- configuration ------------------------------------------------------
```

```
# Pin the UPSTREAM RabbitMQ version 4.2.5.
# The Debian revision ("-1") is detected automatically from the repo. apt then
# resolves a *compatible* Erlang version for you - don't pin Erlang by hand.
RABBITMQ_UPSTREAM="${RABBITMQ_UPSTREAM:-4.2.5}"
INCLUDE_RECOMMENDS="${INCLUDE_RECOMMENDS:-yes}"   # yes = also bundle Recommends

# RabbitMQ's recommended trimmed Erlang set (no GUI/wx). Named explicitly
# because some are runtime-needed but only Recommended, not hard Depends.
ERLANG_PKGS=(
  erlang-base erlang-asn1 erlang-crypto erlang-eldap erlang-ftp erlang-inets
  erlang-mnesia erlang-os-mon erlang-parsetools erlang-public-key
  erlang-runtime-tools erlang-snmp erlang-ssl erlang-syntax-tools
  erlang-tftp erlang-tools erlang-xmerl
)

ARCH="$(dpkg --print-architecture)"
. /etc/os-release
DISTRO="${ID:?cannot detect distro}"            # ubuntu | debian
CODENAME="${VERSION_CODENAME:?cannot detect release codename}"
KEYRING="/usr/share/keyrings/com.rabbitmq.team.gpg"
OUTDIR="$(pwd)/rabbitmq-offline"

# Empty status file = "nothing is installed" -> apt downloads every dependency.
EMPTY_STATUS="$(mktemp)"
trap 'rm -f "$EMPTY_STATUS"' EXIT

echo ">> target: RabbitMQ ${RABBITMQ_UPSTREAM:-latest} | ${DISTRO} ${CODENAME} | ${ARCH}"

# ---- build-host tooling -----------------------------------------------
sudo apt-get update
sudo apt-get install -y --no-install-recommends curl gpg ca-certificates dpkg-dev

# ---- RabbitMQ signing key + repositories ------------------------------
curl -1sLf "https://keys.openpgp.org/vks/v1/by-
fingerprint/0A9AF2115F4687BD29803A206B73A36E6026DFCA" \
  | sudo gpg --dearmor --yes -o "$KEYRING"
sudo chmod 644 "$KEYRING"

# Two mirrors each for Erlang and RabbitMQ; paths track distro + codename.
sudo tee /etc/apt/sources.list.d/rabbitmq.list >/dev/null <<EOF
## Modern Erlang/OTP releases
deb [arch=${ARCH} signed-by=${KEYRING}] https://deb1.rabbitmq.com/rabbitmq-erlang/${DISTRO}/
${CODENAME} ${CODENAME} main
deb [arch=${ARCH} signed-by=${KEYRING}] https://deb2.rabbitmq.com/rabbitmq-erlang/${DISTRO}/
${CODENAME} ${CODENAME} main

## Modern RabbitMQ releases
deb [arch=${ARCH} signed-by=${KEYRING}] https://deb1.rabbitmq.com/rabbitmq-server/${DISTRO}/
${CODENAME} ${CODENAME} main
deb [arch=${ARCH} signed-by=${KEYRING}] https://deb2.rabbitmq.com/rabbitmq-server/${DISTRO}/
${CODENAME} ${CODENAME} main
```

```
EOF

sudo apt-get update

# ---- resolve the exact rabbitmq-server version ------------------------------
RMQ_SPEC="rabbitmq-server"
if [ -n "$RABBITMQ_UPSTREAM" ]; then
  esc="$(printf '%s' "$RABBITMQ_UPSTREAM" | sed 's/[.[\*^$]/\\&/g')"
  RMQ_VER="$(
    apt-cache madison rabbitmq-server \
      | awk -F'|' '{gsub(/ /,"",$2); print $2}' \
      | grep -E "(^|:)${esc}-" \
      | head -n1
  )"
  if [ -z "$RMQ_VER" ]; then
    echo "ERROR: no rabbitmq-server ${RABBITMQ_UPSTREAM} found for ${CODENAME}." >&2
    echo "Available versions:" >&2
    apt-cache madison rabbitmq-server | awk -F'|' '{print "    " $2}' >&2
    exit 1
  fi
  echo ">> resolved rabbitmq-server: ${RMQ_VER}"
  RMQ_SPEC="rabbitmq-server=${RMQ_VER}"
fi

# rabbitmq-server is pinned; Erlang stays unpinned so apt picks compatible
# versions, and the whole set is solved together.
SPECS=("$RMQ_SPEC" "${ERLANG_PKGS[@]}")

REC_FLAG="--no-install-recommends"
[ "$INCLUDE_RECOMMENDS" = "yes" ] && REC_FLAG="--install-recommends"

# ---- download packages + full dependency closure ----------------------------
rm -rf "$OUTDIR"
mkdir -p "$OUTDIR/partial"

# Preview the complete closure (rabbitmq-server + all Erlang modules + base libs).
echo ">> resolving full dependency closure..."
apt-get install --print-uris -y $REC_FLAG \
  -o Dir::State::status="$EMPTY_STATUS" \
  "${SPECS[@]}" \
  | grep -oP "(?<=')[^']*\.deb" > "$OUTDIR/manifest.txt" || true
echo ">> $(wc -l < "$OUTDIR/manifest.txt") package file(s) to download"

# download-only + the empty status file: every dependency is fetched, but the
# real system is never modified.
sudo apt-get install -y --download-only $REC_FLAG \
  -o Dir::State::status="$EMPTY_STATUS" \
  -o Dir::Cache::archives="$OUTDIR" \
  "${SPECS[@]}"

sudo rm -rf "$OUTDIR/partial" "$OUTDIR/lock"
```

```
sudo chown -R "$(id -u):$(id -g)" "$OUTDIR"

# ---- turn the folder into a local apt repo --------------------------------
( cd "$OUTDIR" \
    && dpkg-scanpackages -m . /dev/null > Packages \
    && gzip -9c Packages > Packages.gz )

# ---- integrity manifest ---------------------------------------------------
( cd "$OUTDIR" && sha256sum *.deb > SHA256SUMS )

echo
echo ">> done."
echo ">> bundle: $OUTDIR"
echo ">> files:  $(ls "$OUTDIR"/*.deb | wc -l) .deb"
echo ">> size:   $(du -sh "$OUTDIR" | cut -f1)"
```

2. Run commands.

Code · bash

```
chmod +x prepare_rabbitmq.sh
./prepare_rabbitmq.sh
```

3. Copy the `rabbitmq-offline` folder to a USB drive or secure transfer medium.

4. Move it to the air-gapped server.

Install RabbitMQ

1. On the target server, insert the USB drive or secure transfer medium.

2. Copy `rabbitmq-offline` folder to `/opt/rabbitmq-offline`. E.g:

Code · bash

```
sudo cp -r rabbitmq-offline /opt/rabbitmq-offline
```

3. Run the commands below.

Code · bash

```
# Verify integrity:
cd /opt/rabbitmq-offline && sha256sum -c SHA256SUMS

# Register it as a local apt repo:
echo "deb [trusted=yes] file:/opt/rabbitmq-offline ./" \
       | sudo tee /etc/apt/sources.list.d/rabbitmq-offline.list
```

```
# Update using ONLY this source:
sudo apt-get -o Dir::Etc::sourcelist="sources.list.d/rabbitmq-offline.list" \
             -o Dir::Etc::sourceparts="/dev/null" update

# Install
sudo apt-get install -y rabbitmq-server
sudo systemctl enable --now rabbitmq-server
```

4. Set up user.

Code · bash

```
sudo rabbitmqctl add_user <username> <password>
sudo rabbitmqctl set_permissions -p / <username> "." "." "."
sudo rabbitmqctl set_user_tags <username> administrator
```

Rocky, RHEL

Prepare packages

1. On the preparation machine, create file `prepare_rabbitmq.sh` with content:

Code · bash

```
#!/usr/bin/env bash
#
# Offline RabbitMQ bundle via GitHub releases - use this when the
# yum1/yum2.rabbitmq.com mirrors are unreachable from your build host but
# github.com AND the Rocky repos ARE reachable.
#
# How dependency resolution works here: the two RabbitMQ/Erlang RPMs come from
# GitHub (not a repo), so we stage them into a temporary local repo, then let
# dnf resolve their FULL dependency closure against that staging repo + the
# Rocky repos. That pulls ncurses-libs, openssl-libs, socat, logrotate, etc.
#
# RabbitMQ 4.2.x runs on Erlang 26.2.x or 27.x - verify at
#   https://www.rabbitmq.com/docs/which-erlang
# and pick an available release from
#   https://github.com/rabbitmq/erlang-rpm/releases

set -euo pipefail

# ---- configuration (check the GitHub releases pages for exact asset names) --
RABBITMQ_VERSION="${RABBITMQ_VERSION:-4.2.5}"
RABBITMQ_DIST="${RABBITMQ_DIST:-el8}"       # server RPM is noarch; el8 installs fine on el9
ERLANG_VERSION="${ERLANG_VERSION:-27.2}"    # MUST be compatible with the RabbitMQ version
```

```
ERLANG_DIST="${ERLANG_DIST:-el9}"
ARCH="${ARCH:-$(uname -m)}"
OUTDIR="$(pwd)/rabbitmq-offline"

RMQ_URL="https://github.com/rabbitmq/rabbitmq-server/releases/download/v${RABBITMQ_VERSION}/
rabbitmq-server-${RABBITMQ_VERSION}-1.${RABBITMQ_DIST}.noarch.rpm"
ERL_URL="https://github.com/rabbitmq/erlang-rpm/releases/download/v${ERLANG_VERSION}/erlang-
${ERLANG_VERSION}-1.${ERLANG_DIST}.${ARCH}.rpm"

echo ">> RabbitMQ ${RABBITMQ_VERSION} (${RABBITMQ_DIST}) + Erlang ${ERLANG_VERSION}
(${ERLANG_DIST}/${ARCH})"

# ---- build-host tooling -------------------------------------------------
sudo dnf install -y createrepo_c dnf-plugins-core curl

STAGE="$(mktemp -d)"
trap 'rm -rf "$STAGE"' EXIT
rm -rf "$OUTDIR"
mkdir -p "$OUTDIR"

# ---- fetch the two RPMs from GitHub into a staging repo -----------------
# curl -f makes a 404 fail loudly instead of saving an HTML error page as .rpm.
echo ">> downloading Erlang..."
curl -fL "$ERL_URL" -o "$STAGE/erlang-${ERLANG_VERSION}-1.${ERLANG_DIST}.${ARCH}.rpm"
echo ">> downloading RabbitMQ server..."
curl -fL "$RMQ_URL" -o "$STAGE/rabbitmq-server-${RABBITMQ_VERSION}-1.
${RABBITMQ_DIST}.noarch.rpm"

# Verify they're real RPMs (not HTML error pages), then index as a local repo.
for f in "$STAGE"/*.rpm; do
  rpm -qp "$f" >/dev/null 2>&1 || { echo "ERROR: $f is not a valid RPM (bad URL?)"; exit 1; }
done
createrepo_c "$STAGE"

# ---- resolve + download the FULL dependency closure ---------------------
# --resolve --alldeps : pull every dependency even if installed here, so the
#                       bundle is complete for a clean target.
# The staging repo provides rabbitmq-server + erlang; the Rocky repos provide
# their base-OS dependencies. Versions are pinned so we don't pick up Rocky's
# old SIG rabbitmq/erlang by mistake.
echo ">> resolving full dependency closure..."
dnf download --resolve --alldeps \
  --repofrompath="rmqstage,$STAGE" \
  --setopt=rmqstage.gpgcheck=0 \
  --downloaddir "$OUTDIR" \
  "rabbitmq-server-${RABBITMQ_VERSION}" "erlang-${ERLANG_VERSION}"

# ---- local repo + integrity manifest ------------------------------------
createrepo_c "$OUTDIR"
( cd "$OUTDIR" && sha256sum *.rpm > SHA256SUMS )
```

```
echo
echo ">> done."
echo ">> bundle: $OUTDIR  ($(ls "$OUTDIR"/*.rpm | wc -l) RPMs, $(du -sh "$OUTDIR" | cut -f1))"
```

2. Run commands.

Code · bash

```
chmod +x prepare_rabbitmq.sh
./prepare_rabbitmq.sh
```

3. Copy the rabbitmq-offline folder to a USB drive or secure transfer medium.

4. Move it to the air-gapped server.

Install RabbitMQ

1. On the target server, insert the USB drive or secure transfer medium.

2. Copy `rabbitmq-offline` folder to `/opt/rabbitmq-offline`. E.g:

Code · bash

```
sudo cp -r rabbitmq-offline /opt/rabbitmq-offline
```

3. Run the commands below.

Code · bash

```
# Verify integrity:
cd /opt/rabbitmq-offline && sha256sum -c SHA256SUMS

sudo dnf install -y \
    --disablerepo='*' \
    --repofrompath=rmqoffline,/opt/rabbitmq-offline \
    --setopt=rmqoffline.gpgcheck=0 \
    rabbitmq-server
sudo systemctl enable --now rabbitmq-server
```

4. Set up user.

Code · bash

```
sudo rabbitmqctl add_user <username> <password>
sudo rabbitmqctl set_permissions -p / <username> "." "." "."
sudo rabbitmqctl set_user_tags <username> administrator
```

MD Cluster File Storage

Supported Operating Systems

OS	Version
Windows	Windows 11 23H2+, or Windows Server 2019+.
Linux	Ubuntu 22.04+, Ubuntu 24.04+, Debian 12+, Rocky 9+, or RHEL 9+.

Info

You can use **OpenSSL** or a similar tool (such as **ssh-keygen**) to generate an **X.509 public and private key pair**.

Windows

Overview

This section describes how to install and configure **MetaDefender (MD) Cluster File Storage** service on **Windows** system. After installation, MD Cluster Control Center can connect to MD Cluster File Storage and monitor its system health.

Prerequisites

Before installing the MD Cluster File Storage service, ensure the following requirements are met.

Requirement	Description
Operating System	Windows 11 23H2+, or Windows Server 2019+.
Privileges	Administrator privileges.
Installation package	`md-cluster-file-storage-<version>-1-x64.msi`
Network access	Required port is open (default port: **8890**). A minimum network bandwidth of **1 Gbps** is required. A bandwidth of **5 Gbps or higher is strongly recommended** in the following scenarios: • Load shared among MetaDefender Core instances for archive processing.

Requirement	Description
	• CDR or DLP is enabled.
Hardware	8 vCPU and 8GB RAM
Disk space	Use **SSD storage with high read/write throughput** for optimal performance. A minimum of **1 TB of available disk space** is required. If CDR or DLP is enabled: • Additional disk capacity may be required due to increased processing and storage demands. • It is recommended to enable data retention to manage stored files effectively.

Create the ignition file

Create an ignition file in YAML format. This file contains the credentials required for the service to connect to the system.

The file must include the following keys:

Key	Description
`secure.connection_key`	A **4–64 character alphanumeric string** (a–z, A–Z, 0–9) used by MD Cluster Control Center to connect to the server.
`secure.private_key`	The content of private key in X509 format.
`secure.certificate`	The content of certificate in X509 format.

Example ignition file:

Code · yaml

```yaml
secure:
  connection_key: "1234abcd" # [0-9a-zA-Z]{4,64}
  private_key: |
        -----BEGIN PRIVATE KEY-----
        MIIJQwIBADANBgkqhkiG9w0BAQEFAASCCS0wggkpAgEAAoICAQCjYtuWaICCY0tJ
        PubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8Di3DAmHK
```

```
        nSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTekLWcfI5ZZ
        toGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnHKT/m6DSU
        0xlZ13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6jzhNyMBT
        J1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQUjy5N8pS
        Np7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAzTK4l2pHN
        uC53QVc/EF++GBLAxmvCDq9ZpMIYi7OmzkkAKKC9Ue6Ef217LFQCFIBKIzv9cgi9
        fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqnV0nPN1IMSn
        zXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnWOFIaVrKWLzAtgzoGMTvP/Au
        ehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9psNcjTMaB
        QLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABAoICAFWe8MQZb37k2gdAV3Y6aq8f
        qokKQqbCNLd3giGFwYkezHXoJfg6Di7oZxNcKyw35LFEghkgtQqErQqo35VPIoH+
        vXUpWOjnCmM4muFA9/cX6mYMc8TmJsg0ewLdBCOZVw+wPABlaqz+0UOiSMMftpk9
        fz9JwGd8ERyBsT+tk3Qi6D0vPZVsC1KqxxL/cwIFd3Hf2ZBtJXe0KBn1pktWht5A
        Kqx9mld2Ovl7NjgiC1Fx9r+fZw/iOabFFwQA4dr+R8mEMK/7bd4VXfQ1o/QGGbMT
        G+ulFrsiDyP+rBIAaGC0i7gDjLAIBQeDhP409ZhswIEc/GBtODU372a2CQK/u4Q/
        HBQvuBtKFNkGUooLgCCbFxzgNUGc83GB/6IwbEM7R5uXqsFiE71LpmroDyjKTlQ8
        YZkpIcLNVLw0usoGYHFm2rvCyEVlfsE3Ub8cFyTFk50SeOcF2QL2xzKmmbZEpXgl
        xBHR0hjgon0IKJDGfor4bHO7Nt+1Ece8u2oTEKvpz5aIn44OeC5mApRGy83/0bvs
        esnWjDE/bGpoT8qFuy+0urDEPNId44XcJm1IRIlG56ErxC3l0s11wrIpTmXXckqw
        zFR9s2z7f0zjeyxqZg4NTPI7wkM3M8BXlvp2GTBIeoxrWB4V3YArwu8QF80QBgVz
        mgHl24nTg00UH1OjZsABAoIBAQDOxftSDbSqGytcWqPYP3SZHAWDA0O4ACEM+eCw
        au9ASutl0IDlNDMJ8nC2ph25BMe5hHDWp2cGQJog7pZ/3qQogQho2gUniKDifN77
        40QdykllTzTVROqmP8+efreIvqlzHmuqaGfGs5oTkZaWj5su+B+bT+9rIwZcwfs5
        YRINhQRx17qa++xh5mfE25c+M9fiIBTiNSo4lTxWMBShnK8xrGaMEmN7W0qTMbFH
        PgQz5FcxRjCCqwHilwNBeLDTp/ZECEB7y34khVh531mBE2mNzSVIQcGZP1I/DvXj
        W7UUNdgFwii/GW+6M0uUDy23UVQpbFzcV8o1C2nZc4Fb4zwBAoIBAQDKSJkFwwuR
        naVJS6WxOKjX8MCu9/cKPnwBv2mmI2jgGxHTw5sr3ahmF5eTb8Zo19BowytN+tr6
        2ZFoIBA9Ubc9esEAU8l3fggdfM82cuR9sGcfQVoCh8tMg6BP8IBLOmbSUhN3PG2m
        39I802u0fFNVQCJKhx1m1MFFLOu7lVcDS9JN+oYVPb6MDfBLm5jOiPuYkFZ4gH79
        J7gXI0/YKhaJ7yXthYVkdrSF6Eooer4RZgma62Dd1VNzSq3JBo6rYjF7Lvd+RwDC
        R1thHrmf/IXplxpNVkoMVxtzbrrbgnC25QmvRYc0rlS/kvM4yQhMH3eA7IycDZMp
        Y+0xm7I7jTT7AoIBAGKzKIMDXdCxBWKhNYJ8z7hiItNl1IZZMW2TPUiY0rl6yaCh
        BVXjM9W0r07QPnHZsUiByqb743adkbTUjmxdJzjaVtxN7ZXwZvOVrY7I7fPWYnCE
        fXCr4+IVpZI/ZHZWpGX6CGSgT6EOjCZ5IUufIvEpqVSmtF8MqfXO9o9uIYLokrWQ
        x1dBl5UnuTLDqw8bChq7O5y6yfuWaOWvL7nxI8NvSsfj4y635gIa/0dFeBYZEfHI
        UlGdNVomwXwYEzgE/c19ruIowX7HU/NgxMWTMZhpazlxgesXybel+YNcfDQ4e3RM
        OMz3ZFiaMaJsGGNf4++d9TmMgk4Ns6oDs6Tb9AECggEBAJYzd+SOYo26iBu3nw3L
        65uEeh6xou8pXH0Tu4gQrPQTRZZ/nT3iNgOwqu1gRuxcq7TOjt41UdqIKO8vN7/A
        aJavCpaKoIMowy/aGCbvAvjNPpU3unU8jdl/t08EXs79S5IKPcgAx87sTTi7KDN5
        SYt4tr2uPEe53NTXuSatilG5QCyExIELOuzWAMKzg7CAiIlNS9foWeLyVkBgCQ6S
        me/L8ta+mUDy37K6vC34jh9vK9yrwF6X44ItRoOJafCaVfGI+175q/eWcqTX4q+I
        G4tKls4sL4mgOJLq+ra50aYMxbcuommctPMXU6CrrYyQpPTHMNVDQy2ttFdsq9iK
        TncCggEBAMmt/8yvPflS+xv3kg/ZBvR9JB1In2n3rUCYYD47ReKFqJ03Vmq5C9nY
        56s9w7OUO8perBXlJYmKZQhO4293lvxZD2Iq4NcZbVSCMoHAUzhzY3brdgtSIxa2
        gGveGAezZ38qKIU26dkz7deECY4vrsRkwhpTW0LGVCpjcQoaKvymAoCmAs8V2oMr
        Ziw1YQ9uOUoWwOqm1wZqmVcOXvPIS2gWAs3fQlWjH9hkcQTMsUaXQDOD0aqkSY3E
        NqOvbCV1/oUpRi3076khCoAXI1bKSn/AvR3KDP14B5toHI/F5OTSEiGhhHesgRrs
        fBrpEY1IATtPq1taBZZogRqI3rOkkPk=
        -----END PRIVATE KEY-----
certificate: |
        -----BEGIN CERTIFICATE-----
```

```
MIIF5jCCA86gAwIBAgIJANq50IuwPFKgMA0GCSqGSIb3DQEBCwUAMIGGMQswCQYD
VQQGEwJHQjEQMA4GA1UECAwHRXJld2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEb
MBkGA1UECgwSbGlid2Vic29ja2V0cy10ZXN0MRIwEAYDVQQDDAlsb2NhbGhvc3Qx
HzAdBgkqhkiG9w0BCQEWEG5vbmVAaW52YWxpZC5vcmcwIBcNMTgwMzIwMDQxNjA3
WhgPMjExODAyMjQwNDE2MDdaMIGGMQswCQYDVQQGEwJHQjEQMA4GA1UECAwHRXJl
d2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEbMBkGA1UECgwSbGlid2Vic29ja2V0
cy10ZXN0MRIwEAYDVQQDDAlsb2NhbGhvc3QxHzAdBgkqhkiG9w0BCQEWEG5vbmVA
aW52YWxpZC5vcmcwggIiMA0GCSqGSIb3DQEBAQUAA4ICDwAwggIKAoICAQCjYtuW
aICCY0tJPubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8
Di3DAmHKnSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTek
LWcfI5ZZtoGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnH
KT/m6DSU0xlZ13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6
jzhNyMBTJ1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQ
Ujy5N8pSNp7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAz
TK4l2pHNuC53QVc/EF++GBLAxmvCDq9ZpMIYi7OmzkkAKKC9Ue6Ef217LFQCFIBK
Izv9cgi9fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqnV0
nPN1IMSnzXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnWOFIaVrKWLzAtgzo
GMTvP/AuehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9p
sNcjTMaBQLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABo1MwUTAdBgNVHQ4EFgQU
9mYU23tW2zsomkKTAXarjr2vjuswHwYDVR0jBBgwFoAU9mYU23tW2zsomkKTAXar
jr2vjuswDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAgEANjIBMrow
YNCbhAJdP7dhlhT2RUFRdeRUJD0IxrH/hkvb6myHHnK8nOYezFPjUlmRKUgNEDuA
xbnXZzPdCRNV9V2mShbXvCyiDY7WCQE2Bn44z26O0uWVk+7DNNLH9BnkwUtOnM9P
wtmD9phWexm4q2GnTsiL6Ul6cy0QlTJWKVLEUQQ6yda582e23J1AXqtqFcpfoE34
H3afEiGy882b+ZBiwkeV+oq6XVF8sFyr9zYrv9CvWTYlkpTQfLTZSsgPdEHYVcjv
xQ2D+XyDR0aRLRlvxUa9dHGFHLICG34Juq5Ai6lM1EsoD8HSsJpMcmrH7MWw2cKk
ujC3rMdFTtte83wF1uuF4FjUC72+SmcQN7A386BC/nk2TTsJawTDzqwOu/VdZv2g
1WpTHlumlClZeP+G/jkSyDwqNnTu1aodDmUa4xZodfhP1HWPwUKFcq8oQr148QYA
AOlbUOJQU7QwRWd1VbnwhDtQWXC92A2w1n/xkZSR1BM/NUSDhkBSUU1WjMbWg6Gg
mnIZLRerQCu1Oozr87rOQqQakPkyt8BUSNK3K42j2qcfhAONdRl8Hq8Qs5pupy+s
8sdCGDlwR3JNCMv6u48OK87F4mcIxhkSefFJUFII25pCGN5WtE4p5l+9cnO1GrIX
e2Hl/7M0c/lbZ4FvXgARlex2rkgS0Ka06HE=
-----END CERTIFICATE-----
```

Save the ignition file to the following path on the target machine:

Code · bash

```
C:\opswat\md_cluster_file_storage.yml
```

Info

The ignition file contains sensitive credentials. This file can be safely deleted any time after the installation is complete.

Install the service

1. Copy the installer file (`.msi`) to the target machine.
2. Open **PowerShell** with **Administrator privileges**.
3. Run the following command to start the installation in **silent mode**:

```
Code · powershell

msiexec.exe /i <md_cluster_file_storage_package>.msi /qn
```

Verify the service status

1. Open **PowerShell** and run the following command:

```
Code · powershell

Get-Service -Name md-cluster-file-storage
```

2. Check the **Running** Status in the output.
3. If the service is not running, start it manually:

```
Code · powershell

Start-Service -Name md-cluster-file-storage
```

Service management

Action	Command
Check service status	`Get-Service -Name md-cluster-file-storage`
Start service	`Start-Service -Name md-cluster-file-storage`
Stop service	`Stop-Service -Name md-cluster-file-storage`
Restart service	`Restart-Service -Name md-cluster-file-storage`

Customize the service configuration

During installation, MD Cluster File Storage service generates a configuration file at:

```bash
C:\Program Files\OPSWAT\MetaDefender Cluster File Storage\md_cluster_file_storage.yml
```

To customize the service behavior:

1. Open the configuration file in a text editor.
2. Modify the required settings according to your environment.
3. Save the changes.
4. Restart the service to apply the new settings.

```powershell
Restart-Service -Name md-cluster-file-storage
```

Directory structure

- `C:\opswat\md_cluster_file_storage.yml`: Service Ignition file.
- `C:\Program Files\OPSWAT\MetaDefender Cluster File Storage\md_cluster_file_storage.yml`: Service configuration file.
- `C:\Program Files\OPSWAT\MetaDefender Cluster File Storage\data\log\file-storage.log`: Service log file.
- `C:\Program Files\OPSWAT\MetaDefender Cluster File Storage\data\storage`: Default storage directory.

Log files

To check the service logs, open the file `C:\Program Files\OPSWAT\MetaDefender Cluster File Storage\data\log\file-storage.log`.

To check logs using Event Viewer:

1. Open **Event Viewer**.
2. Navigate to **Windows Logs > Application**.
3. Look for events related to **MD Cluster File Storage** Service.

Uninstall the service

Open **PowerShell** and run the following command:

Code · powershell

```
Uninstall-Package -Name 'MetaDefender Cluster File Storage' -Force
```

> **Warning**
>
> If `storage.path` is not set in the Ignition file, the storage directory will be **deleted** when MD Cluster File Storage is uninstalled.

Troubleshooting

A. Service is not running

1. Check the service status

Code · powershell

```
Get-Service -Name md-cluster-file-storage
```

2. Start the service if it is not running:

Code · powershell

```
Start-Service -Name md-cluster-file-storage
```

B. Installation fails

Possible causes

- Insufficient privileges.
- Missing dependencies.

Solution

- Ensure the installation command is executed with **Administrator** privileges.
- Ensure dependencies are installed.

C. MD Cluster Control Center cannot connect to MD Cluster File Storage.

Possible causes

- Network connectivity issues.
- Firewall restrictions.

Solution

- Ensure MD Cluster Control Center has network connectivity to MD Cluster File Storage.
- Verify that firewall rules allow **inbound** and **outbound** connections.

Ignition file key reference

- `secure.connection_key` (**Required**)
  - Value type: string.
  - Description: Use a **4–64 character string** that contains only numbers (0–9) and letters (a–z, A–Z). This string is used by clients to connect to the server. Set this value as the `identity.connection_key` in the MD Cluster Control Center configuration file.
- `secure.private_key` (**Required**)
  - Value type: string.
  - Description: The content of private key in X509 format.
- `secure.certificate` (**Required**)
  - Value type: string.
  - Description: The content of certificate in X509 format.
- `storage.path` (optional)
  - Value type: string.
  - Description: Path to an existing directory where the MD Cluster File Storage server stores its files. On Linux, the server must have full permissions to access this directory.
- `rest.host` [optional)
  - Value type: string.
  - Description: IP address (V4/V6) or host where the server resides on. Default value is '*'. Notes: `'*'` allows the service to accept connections from all network interfaces. To bind the service to a specific interface, specify its IP address or domain name. For example, to listen on all IPv4 interfaces, set the host to `0.0.0.0`.
- `rest.port` [optional)
  - Value type: number.
  - Description: The port where the server resides on. Default value is `8890`.
- `log.streams[@].log_type` [optional)
  - Value type: string.
  - Description: Type of log device (`file`, or `syslog`)
- `log.streams[@].log_level` [optional)

- Value type: string.
- Description: Level of log message (`dump`, `debug`, `info`, `warning`, or `error`).

- `log.streams[@].log_path` [optional]
  - Value type: string.
  - Description: Location where logs are written. If `log.streams[@].log_type` is `"file"` then `log.streams[@].log_path` is the path to a file on file system where logs are written. If `log.streams[@].log_type` is `"syslog"` then
    - `log.streams[@].log_path` can be `[tcp/udp]://host:port` where host:port is the host and port to a remote syslog server that supports TCP or UDP protocol.
    - `log.streams[@].log_path` can be `"local"` to write log to local syslog server (Linux only).

> **Info**
>
> If `storage.path` is not set in the Ignition file, MD Cluster File Storage saves submitted files to the default storage directory at `C:\Program Files\OPSWAT\MetaDefender Cluster File Storage\data\storage`

Linux

Overview

This section describes how to install and configure **MetaDefender (MD) Cluster File Storage** service on supported **Linux** distributions. After installation, MD Cluster Control Center can connect to MD Cluster File Storage and monitor its system health.

Prerequisites

Before installing the MD Cluster File Storage service, ensure the following requirements are met.

Requirement	Description
Operating System	Ubuntu 22.04+, Ubuntu 24.04+, Debian 12+, Rocky 9+, or RHEL 9+.
Privileges	Root or sudo privileges.
Installation package	Debian/Ubuntu: `md-cluster-file-storage_<version>-1_amd64.deb` Rocky/RHEL: `md-cluster-file-storage-<version>-1.x86_64.rpm`
Network access	Required port is open (default port: **8890**).

Requirement	Description
	A minimum network bandwidth of **1 Gbps** is required. A bandwidth of **5 Gbps or higher is strongly recommended** in the following scenarios: • CDR or DLP is enabled. • Load shared among MetaDefender Core instances for archive processing.
Hardware	8 vCPU and 8GB RAM
Disk space	Use **SSD storage with high read/write throughput** for optimal performance. A minimum of **1 TB of available disk space** is required. If CDR or DLP is enabled: • Additional disk capacity may be required due to increased processing and storage demands. • It is recommended to enable data retention to manage stored files effectively.

Create the ignition file

Create an ignition file in YAML format. This file contains the credentials required for the service to connect to the system.

The file must include the following keys:

Key	Description
`secure.connection_key`	A **4–64 character alphanumeric string** (a–z, A–Z, 0–9) used by MD Cluster Control Center to connect to the server.
`secure.private_key`	The content of private key in X509 format.
`secure.certificate`	The content of certificate in X509 format.

Example ignition file:

Code · yaml

```
secure:
  connection_key: "1234abcd" # [0-9a-zA-Z]{4,64}
  private_key: |
        -----BEGIN PRIVATE KEY-----
        MIIJQwIBADANBgkqhkiG9w0BAQEFAASCCS0wggkpAgEAAoICAQCjYtuWaICCY0tJ
        PubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8Di3DAmHK
        nSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTekLWcfI5ZZ
        toGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnHKT/m6DSU
        0xlZ13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6jzhNyMBT
        J1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQUjy5N8pS
        Np7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAzTK4l2pHN
        uC53QVc/EF++GBLAxmvCDq9ZpMIYi7OmzkkAKKC9Ue6Ef217LFQCFIBKIzv9cgi9
        fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqnV0nPN1IMSn
        zXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnWOFIaVrKWLzAtgzoGMTvP/Au
        ehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9psNcjTMaB
        QLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABAoICAFWe8MQZb37k2gdAV3Y6aq8f
        qokKQqbCNLd3giGFwYkezHXoJfg6Di7oZxNcKyw35LFEghkgtQqErQqo35VPIoH+
        vXUpWOjnCmM4muFA9/cX6mYMc8TmJsg0ewLdBCOZVw+wPABlaqz+0UOiSMMftpk9
        fz9JwGd8ERyBsT+tk3Qi6D0vPZVsC1KqxxL/cwIFd3Hf2ZBtJXe0KBn1pktWht5A
        Kqx9mld2Ovl7NjgiC1Fx9r+fZw/iOabFFwQA4dr+R8mEMK/7bd4VXfQ1o/QGGbMT
        G+ulFrsiDyP+rBIAaGC0i7gDjLAIBQeDhP409ZhswIEc/GBtODU372a2CQK/u4Q/
        HBQvuBtKFNkGUooLgCCbFxzgNUGc83GB/6IwbEM7R5uXqsFiE71LpmroDyjKTlQ8
        YZkpIcLNVLw0usoGYHFm2rvCyEVlfsE3Ub8cFyTFk50SeOcF2QL2xzKmmbZEpXgl
        xBHR0hjgon0IKJDGfor4bHO7Nt+1Ece8u2oTEKvpz5aIn44OeC5mApRGy83/0bvs
        esnWjDE/bGpoT8qFuy+0urDEPNId44XcJm1IRIlG56ErxC3l0s11wrIpTmXXckqw
        zFR9s2z7f0zjeyxqZg4NTPI7wkM3M8BXlvp2GTBIeoxrWB4V3YArwu8QF80QBgVz
        mgHl24nTg00UH1OjZsABAoIBAQDOxftSDbSqGytcWqPYP3SZHAWDA0O4ACEM+eCw
        au9ASutl0IDlNDMJ8nC2ph25BMe5hHDWp2cGQJog7pZ/3qQogQho2gUniKDifN77
        40QdykllTzTVROqmP8+efreIvqlzHmuqaGfGs5oTkZaWj5su+B+bT+9rIwZcwfs5
        YRINhQRx17qa++xh5mfE25c+M9fiIBTiNSo4lTxWMBShnK8xrGaMEmN7W0qTMbFH
        PgQz5FcxRjCCqwHilwNBeLDTp/ZECEB7y34khVh531mBE2mNzSVIQcGZP1I/DvXj
        W7UUNdgFwii/GW+6M0uUDy23UVQpbFzcV8o1C2nZc4Fb4zwBAoIBAQDKSJkFwwuR
        naVJS6WxOKjX8MCu9/cKPnwBv2mmI2jgGxHTw5sr3ahmF5eTb8Zo19BowytN+tr6
        2ZFoIBA9Ubc9esEAU8l3fggdfM82cuR9sGcfQVoCh8tMg6BP8IBLOmbSUhN3PG2m
        39I802u0fFNVQCJKhx1m1MFFLOu7lVcDS9JN+oYVPb6MDfBLm5jOiPuYkFZ4gH79
        J7gXI0/YKhaJ7yXthYVkdrSF6Eooer4RZgma62Dd1VNzSq3JBo6rYjF7Lvd+RwDC
        R1thHrmf/IXplxpNVkoMVxtzbrrbgnC25QmvRYc0rlS/kvM4yQhMH3eA7IycDZMp
        Y+0xm7I7jTT7AoIBAGKzKIMDXdCxBWKhNYJ8z7hiItNl1IZZMW2TPUiY0rl6yaCh
        BVXjM9W0r07QPnHZsUiByqb743adkbTUjmxdJzjaVtxN7ZXwZvOVrY7I7fPWYnCE
        fXCr4+IVpZI/ZHZWpGX6CGSgT6EOjCZ5IUufIvEpqVSmtF8MqfXO9o9uIYLokrWQ
        x1dBl5UnuTLDqw8bChq7O5y6yfuWaOWvL7nxI8NvSsfj4y635gIa/0dFeBYZEfHI
        UlGdNVomwXwYEzgE/c19ruIowX7HU/NgxMWTMZhpazlxgesXybel+YNcfDQ4e3RM
        OMz3ZFiaMaJsGGNf4++d9TmMgk4Ns6oDs6Tb9AECggEBAJYzd+SOYo26iBu3nw3L
        65uEeh6xou8pXH0Tu4gQrPQTRZZ/nT3iNgOwqu1gRuxcq7TOjt41UdqIKO8vN7/A
        aJavCpaKoIMowy/aGCbvAvjNPpU3unU8jdl/t08EXs79S5IKPcgAx87sTTi7KDN5
        SYt4tr2uPEe53NTXuSatilG5QCyExIELOuzWAMKzg7CAiIlNS9foWeLyVkBgCQ6S
        me/L8ta+mUDy37K6vC34jh9vK9yrwF6X44ItRoOJafCaVfGI+175q/eWcqTX4q+I
        G4tKls4sL4mgOJLq+ra50aYMxbcuommctPMXU6CrrYyQpPTHMNVDQy2ttFdsq9iK
        TncCggEBAMmt/8yvPflS+xv3kg/ZBvR9JB1In2n3rUCYYD47ReKFqJ03Vmq5C9nY
        56s9w7OUO8perBXlJYmKZQhO4293lvxZD2Iq4NcZbVSCMoHAUzhzY3brdgtSIxa2
        gGveGAezZ38qKIU26dkz7deECY4vrsRkwhpTW0LGVCpjcQoaKvymAoCmAs8V2oMr
```

```
        Ziw1YQ9uOUoWwOqm1wZqmVcOXvPIS2gWAs3fQlWjH9hkcQTMsUaXQDOD0aqkSY3E
        NqOvbCV1/oUpRi3076khCoAXI1bKSn/AvR3KDP14B5toHI/F5OTSEiGhhHesgRrs
        fBrpEY1IATtPq1taBZZogRqI3rOkkPk=
        -----END PRIVATE KEY-----
  certificate: |
        -----BEGIN CERTIFICATE-----
        MIIF5jCCA86gAwIBAgIJANq50IuwPFKgMA0GCSqGSIb3DQEBCwUAMIGGMQswCQYD
        VQQGEwJHQjEQMA4GA1UECAwHRXJld2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEb
        MBkGA1UECgwSbGlid2Vic29ja2V0cy10ZXN0MRIwEAYDVQQDDAlsb2NhbGhvc3Qx
        HzAdBgkqhkiG9w0BCQEWEG5vbmVAaW52YWxpZC5vcmcwIBcNMTgwMzIwMDQxNjA3
        WhgPMjExODAyMjQwNDE2MDdaMIGGMQswCQYDVQQGEwJHQjEQMA4GA1UECAwHRXJl
        d2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEbMBkGA1UECgwSbGlid2Vic29ja2V0
        cy10ZXN0MRIwEAYDVQQDDAlsb2NhbGhvc3QxHzAdBgkqhkiG9w0BCQEWEG5vbmVA
        aW52YWxpZC5vcmcwggIiMA0GCSqGSIb3DQEBAQUAA4ICDwAwggIKAoICAQCjYtuW
        aICCY0tJPubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8
        Di3DAmHKnSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTek
        LWcfI5ZZtoGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnH
        KT/m6DSU0xlZ13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6
        jzhNyMBTJ1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQ
        Ujy5N8pSNp7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAz
        TK4l2pHNuC53QVc/EF++GBLAxmvCDq9ZpMIYi7OmzkkAKKC9Ue6Ef217LFQCFIBK
        Izv9cgi9fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqnV0
        nPN1IMSnzXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnWOFIaVrKWLzAtgzo
        GMTvP/AuehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9p
        sNcjTMaBQLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABo1MwUTAdBgNVHQ4EFgQU
        9mYU23tW2zsomkKTAXarjr2vjuswHwYDVR0jBBgwFoAU9mYU23tW2zsomkKTAXar
        jr2vjuswDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAgEANjIBMrow
        YNCbhAJdP7dhlhT2RUFRdeRUJD0IxrH/hkvb6myHHnK8nOYezFPjUlmRKUgNEDuA
        xbnXZzPdCRNV9V2mShbXvCyiDY7WCQE2Bn44z26O0uWVk+7DNNLH9BnkwUtOnM9P
        wtmD9phWexm4q2GnTsiL6Ul6cy0QlTJWKVLEUQQ6yda582e23J1AXqtqFcpfoE34
        H3afEiGy882b+ZBiwkeV+oq6XVF8sFyr9zYrv9CvWTYlkpTQfLTZSsgPdEHYVcjv
        xQ2D+XyDR0aRLRlvxUa9dHGFHLICG34Juq5Ai6lM1EsoD8HSsJpMcmrH7MWw2cKk
        ujC3rMdFTtte83wF1uuF4FjUC72+SmcQN7A386BC/nk2TTsJawTDzqwOu/VdZv2g
        1WpTHlumlClZeP+G/jkSyDwqNnTu1aodDmUa4xZodfhP1HWPwUKFcq8oQr148QYA
        AOlbUOJQU7QwRWd1VbnwhDtQWXC92A2w1n/xkZSR1BM/NUSDhkBSUU1WjMbWg6Gg
        mnIZLRerQCu1Oozr87rOQqQakPkyt8BUSNK3K42j2qcfhAONdRl8Hq8Qs5pupy+s
        8sdCGDlwR3JNCMv6u48OK87F4mcIxhkSefFJUFII25pCGN5WtE4p5l+9cnO1GrIX
        e2Hl/7M0c/lbZ4FvXgARlex2rkgS0Ka06HE=
        -----END CERTIFICATE-----
```

Save the ignition file to the following path on the target machine:

Code · bash

```
/etc/opswat/md_cluster_file_storage.yml
```

Info

The ignition file contains sensitive credentials. This file can be safely deleted any time after the installation is

complete.

Install the service

1. Copy the installer file (`.deb` or `.rpm`) to the target machine.
2. Open **Terminal**.
3. Run the following command to start the installation:

Code · bash

```
# Debian or Ubuntu
sudo apt install -y uuid tar
sudo dpkg -i <md_cluster_file_storage_package> || sudo apt install -f

# Rocky or RHEL
sudo dnf install -y yum-utils
sudo dnf config-manager --set-enabled devel
sudo dnf update -y
sudo dnf install -y redhat-lsb-core libuuid tar
sudo yum install <md_cluster_file_storage_package> -y
```

Verify the service status

1. Open **Terminal** and run the following command:

Code · bash

```
sudo systemctl status md-cluster-file-storage
```

2. Check the **active (running)** field in the output.
3. If the service is not running, start it manually:

Code · bash

```
sudo systemctl restart md-cluster-file-storage
```

4. To ensure the service starts automatically at system boot:

Code · bash

```
sudo systemctl enable md-cluster-file-storage
```

Service management

Action	Command
Check service status	`sudo systemctl status md-cluster-file-storage`
Start service	`sudo systemctl start md-cluster-file-storage`
Stop service	`sudo systemctl stop md-cluster-file-storage`
Restart service	`sudo systemctl restart md-cluster-file-storage`
Enable service at boot	`sudo systemctl enable md-cluster-file-storage`

Customize the service configuration

During installation, MD Cluster File Storage service generates a configuration file at:

Code · bash

```
/etc/md-cluster-file-storage/md_cluster_file_storage.yml
```

To customize the service behavior:

1. Open the configuration file in a text editor such as **nano**.

Code · bash

```
sudo nano /etc/md-cluster-file-storage/md_cluster_file_storage.yml
```

2. Modify the required settings according to your environment.

3. Save the changes.

4. Restart the service to apply the new settings.

Code · bash

```
sudo systemctl restart md-cluster-file-storage
```

Directory structure

- `/etc/opswat/mdcluster_file_storage.yml`: Service Ignition file.
- `/etc/md-cluster-file-storage/md_cluster_file_storage.yml`: Service configuration file.
- `/var/log/md-cluster-file-storage/`: Default log directory.
- `/var/lib/md-cluster-file-storage/storage`: Default storage directory.

Log files

To check the service logs, open the file: `/var/log/md-cluster-file-storage/file-storage.log`.

To check the system log, run the following command:

Code · bash

```
# Fetch by systemd-journald
sudo journalctl -r

# Ubuntu syslog
sudo cat /var/log/syslog

# Rocky or RHEL syslog
sudo cat /var/log/message
```

Uninstall the service

Code · bash

```
# Debian or Ubuntu
sudo apt purge <md_cluster_file_storage_package>

# Rocky or RHEL
sudo yum remove <md_cluster_file_storage_package>
```

Warning

If `storage.path` is not set in the Ignition file, the storage directory will be **deleted** when MD Cluster File Storage is uninstalled.

Troubleshooting

A. Service is not running

1. Check the service status

```bash
sudo systemctl status md-cluster-file-storage
```

2. Start the service if it is not running:

```bash
sudo systemctl start md-cluster-file-storage
```

B. Installation fails

Possible causes

- Insufficient privileges.
- Missing dependencies.

Solution

- Ensure the installation command is executed with **sudo**.
- Ensure dependencies are installed.

C. MD Cluster Control Center cannot connect to MD Cluster File Storage.

Possible causes

- Network connectivity issues.
- Firewall restrictions.

Solution

- Ensure MD Cluster Control Center has network connectivity to MD Cluster File Storage.
- Verify that firewall rules allow **inbound** and **outbound** connections.

Ignition file key reference

- `secure.connection_key` (**Required**)
  - Value type: string.

- Description: Use a **4–64 character string** that contains only numbers (0–9) and letters (a–z, A–Z). This string is used by clients to connect to the server. Set this value as the `identity.connection_key` in the MD Cluster Control Center configuration file.

- `secure.private_key` (**Required**)
  - Value type: string.
  - Description: The content of private key in X509 format.

- `secure.certificate` (**Required**)
  - Value type: string.
  - Description: The content of certificate in X509 format.

- `storage.path` (optional)
  - Value type: string.
  - Description: Path to an existing directory where the MD Cluster File Storage server stores its files. On Linux, the server must have full permissions to access this directory.

- `rest.host` [optional)
  - Value type: string.
  - Description: IP address (V4/V6) or host where the server resides on. Default value is '*'. Notes: `'*'` allows the service to accept connections from all network interfaces. To bind the service to a specific interface, specify its IP address or domain name. For example, to listen on all IPv4 interfaces, set the host to `0.0.0.0`.

- `rest.port` [optional)
  - Value type: number.
  - Description: The port where the server resides on. Default value is `8890`.

- `log.streams[@].log_type` [optional)
  - Value type: string.
  - Description: Type of log device (`file`, or `syslog`)

- `log.streams[@].log_level` [optional)
  - Value type: string.
  - Description: Level of log message (`dump`, `debug`, `info`, `warning`, or `error`).

- `log.streams[@].log_path` [optional)
  - Value type: string.
  - Description: Location where logs are written. If `log.streams[@].log_type` is `"file"` then `log.streams[@].log_path` is the path to a file on file system where logs are written. If `log.streams[@].log_type` is `"syslog"` then

- `log.streams[@].log_path` can be `[tcp/udp]://host:port` where host:port is the host and port to a remote syslog server that supports TCP or UDP protocol.
- `log.streams[@].log_path` can be `"local"` to write log to local syslog server (Linux only).

> **Info**
>
> If `storage.path` is not set in the Ignition file, MD Cluster File Storage saves submitted files to the default storage directory at `/var/lib/md-cluster-file-storage/storage`

MD Cluster Identity Service

Supported Operating Systems

OS	Version
Windows	Windows 11 23H2+, or Windows Server 2019+.
Linux	Ubuntu 22.04+, Ubuntu 24.04+, Debian 12+, Rocky 9+, or RHEL 9+.

Windows

Overview

This section describes how to install and configure **MetaDefender (MD) Cluster Identity Service** on **Windows** system. After installation, MD Cluster Control Center can connect to MD Cluster Identity and monitor its system health.

Prerequisites

Before installing the MD Cluster Identity Service, ensure the following requirements are met.

Requirement	Description
Operating System	Windows 11 23H2+, or Windows Server 2019+.
Privileges	Administrator privileges
Installation package	`md-cluster-identity-service-<version>-1-x64.msi`

Requirement	Description
Network access	Required port is open (default port: **8891**). A minimum network bandwidth of **1 Gbps** is required.
Hardware	4vCPU and 4GB RAM
Disk space	A minimum of **50 GB of available disk space** is required.

Create the ignition file

Create an ignition file in YAML format. This file contains the credentials required for the service to connect to the system.

The file must include the following keys:

Key	Description
`database.host`	IP address or domain name of the server hosting PostgreSQL.
`database.port`	IP address or domain name of the server hosting PostgreSQL.
`database.user`	PostgreSQL server user. SUPERUSER privileges are required to set up the database and extensions during the initial configuration.
`database.password`	PostgreSQL user password.
`secure.connection_key`	A **4–64 character alphanumeric string** (a–z, A–Z, 0–9) used by MD Cluster Control Center to connect to the server.
`secure.private_key`	The content of private key in X509 format.
`secure.certificate`	The content of certificate in X509 format.

Example ignition file:

Code · yaml

```
database:
  host: "posgresql_host"
  port: 5432
  user: "postgres"
  password: "admin"
secure:
  connection_key: "1234abcd" # [0-9a-zA-Z]{4,64}
  private_key: |
        -----BEGIN PRIVATE KEY-----
        MIIJQwIBADANBgkqhkiG9w0BAQEFAASCCS0wggkpAgEAAoICAQCjYtuWaICCY0tJ
        PubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8Di3DAmHK
        nSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTekLWcfI5ZZ
        toGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnHKT/m6DSU
        0xlZ13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6jzhNyMBT
        J1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQUjy5N8pS
        Np7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAzTK4l2pHN
        uC53QVc/EF++GBLAxmvCDq9ZpMIYi7OmzkkAKKC9Ue6Ef217LFQCFIBKIzv9cgi9
        fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqnV0nPN1IMSn
        zXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnWOFIaVrKWLzAtgzoGMTvP/Au
        ehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9psNcjTMaB
        QLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABAoICAFWe8MQZb37k2gdAV3Y6aq8f
        qokKQqbCNLd3giGFwYkezHXoJfg6Di7oZxNcKyw35LFEghkgtQqErQqo35VPIoH+
        vXUpWOjnCmM4muFA9/cX6mYMc8TmJsg0ewLdBCOZVw+wPABlaqz+0UOiSMMftpk9
        fz9JwGd8ERyBsT+tk3Qi6D0vPZVsC1KqxxL/cwIFd3Hf2ZBtJXe0KBn1pktWht5A
        Kqx9mld2Ovl7NjgiC1Fx9r+fZw/iOabFFwQA4dr+R8mEMK/7bd4VXfQ1o/QGGbMT
        G+ulFrsiDyP+rBIAaGC0i7gDjLAIBQeDhP409ZhswIEc/GBtODU372a2CQK/u4Q/
        HBQvuBtKFNkGUooLgCCbFxzgNUGc83GB/6IwbEM7R5uXqsFiE71LpmroDyjKTlQ8
        YZkpIcLNVLw0usoGYHFm2rvCyEVlfsE3Ub8cFyTFk50SeOcF2QL2xzKmmbZEpXgl
        xBHR0hjgon0IKJDGfor4bHO7Nt+1Ece8u2oTEKvpz5aIn44OeC5mApRGy83/0bvs
        esnWjDE/bGpoT8qFuy+0urDEPNId44XcJm1IRIlG56ErxC3l0s11wrIpTmXXckqw
        zFR9s2z7f0zjeyxqZg4NTPI7wkM3M8BXlvp2GTBIeoxrWB4V3YArwu8QF80QBgVz
        mgHl24nTg00UH1OjZsABAoIBAQDOxftSDbSqGytcWqPYP3SZHAWDA0O4ACEM+eCw
        au9ASutl0IDlNDMJ8nC2ph25BMe5hHDWp2cGQJog7pZ/3qQogQho2gUniKDifN77
        40QdykllTzTVROqmP8+efreIvqlzHmuqaGfGs5oTkZaWj5su+B+bT+9rIwZcwfs5
        YRINhQRx17qa++xh5mfE25c+M9fiIBTiNSo4lTxWMBShnK8xrGaMEmN7W0qTMbFH
        PgQz5FcxRjCCqwHilwNBeLDTp/ZECEB7y34khVh531mBE2mNzSVIQcGZP1I/DvXj
        W7UUNdgFwii/GW+6M0uUDy23UVQpbFzcV8o1C2nZc4Fb4zwBAoIBAQDKSJkFwwuR
        naVJS6WxOKjX8MCu9/cKPnwBv2mmI2jgGxHTw5sr3ahmF5eTb8Zo19BowytN+tr6
        2ZFoIBA9Ubc9esEAU8l3fggdfM82cuR9sGcfQVoCh8tMg6BP8IBLOmbSUhN3PG2m
        39I802u0fFNVQCJKhx1m1MFFLOu7lVcDS9JN+oYVPb6MDfBLm5jOiPuYkFZ4gH79
        J7gXI0/YKhaJ7yXthYVkdrSF6Eooer4RZgma62Dd1VNzSq3JBo6rYjF7Lvd+RwDC
        R1thHrmf/IXplxpNVkoMVxtzbrrbgnC25QmvRYc0rlS/kvM4yQhMH3eA7IycDZMp
        Y+0xm7I7jTT7AoIBAGKzKIMDXdCxBWKhNYJ8z7hiItNl1IZZMW2TPUiY0rl6yaCh
        BVXjM9W0r07QPnHZsUiByqb743adkbTUjmxdJzjaVtxN7ZXwZvOVrY7I7fPWYnCE
        fXCr4+IVpZI/ZHZWpGX6CGSgT6EOjCZ5IUufIvEpqVSmtF8MqfXO9o9uIYLokrWQ
        x1dBl5UnuTLDqw8bChq7O5y6yfuWaOWvL7nxI8NvSsfj4y635gIa/0dFeBYZEfHI
        UlGdNVomwXwYEzgE/c19ruIowX7HU/NgxMWTMZhpazlxgesXybel+YNcfDQ4e3RM
        OMz3ZFiaMaJsGGNf4++d9TmMgk4Ns6oDs6Tb9AECggEBAJYzd+SOYo26iBu3nw3L
        65uEeh6xou8pXH0Tu4gQrPQTRZZ/nT3iNgOwqu1gRuxcq7TOjt41UdqIKO8vN7/A
        aJavCpaKoIMowy/aGCbvAvjNPpU3unU8jdl/t08EXs79S5IKPcgAx87sTTi7KDN5
        SYt4tr2uPEe53NTXuSatilG5QCyExIELOuzWAMKzg7CAiIlNS9foWeLyVkBgCQ6S
```

```
        me/L8ta+mUDy37K6vC34jh9vK9yrwF6X44ItRoOJafCaVfGI+175q/eWcqTX4q+I
        G4tKls4sL4mgOJLq+ra50aYMxbcuommctPMXU6CrrYyQpPTHMNVDQy2ttFdsq9iK
        TncCggEBAMmt/8yvPflS+xv3kg/ZBvR9JB1In2n3rUCYYD47ReKFqJ03Vmq5C9nY
        56s9w7OUO8perBXlJYmKZQhO4293lvxZD2Iq4NcZbVSCMoHAUzhzY3brdgtSIxa2
        gGveGAezZ38qKIU26dkz7deECY4vrsRkwhpTW0LGVCpjcQoaKvymAoCmAs8V2oMr
        Ziw1YQ9uOUoWwOqm1wZqmVcOXvPIS2gWAs3fQlWjH9hkcQTMsUaXQDOD0aqkSY3E
        NqOvbCV1/oUpRi3076khCoAXI1bKSn/AvR3KDP14B5toHI/F5OTSEiGhhHesgRrs
        fBrpEY1IATtPq1taBZZogRqI3rOkkPk=
        -----END PRIVATE KEY-----
  certificate: |
        -----BEGIN CERTIFICATE-----
        MIIF5jCCA86gAwIBAgIJANq50IuwPFKgMA0GCSqGSIb3DQEBCwUAMIGGMQswCQYD
        VQQGEwJHQjEQMA4GA1UECAwHRXJld2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEb
        MBkGA1UECgwSbGlid2Vic29ja2V0cy10ZXN0MRIwEAYDVQQDDAlsb2NhbGhvc3Qx
        HzAdBgkqhkiG9w0BCQEWEG5vbmVAaW52YWxpZC5vcmcwIBcNMTgwMzIwMDQxNjA3
        WhgPMjExODAyMjQwNDE2MDdaMIGGMQswCQYDVQQGEwJHQjEQMA4GA1UECAwHRXJl
        d2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEbMBkGA1UECgwSbGlid2Vic29ja2V0
        cy10ZXN0MRIwEAYDVQQDDAlsb2NhbGhvc3QxHzAdBgkqhkiG9w0BCQEWEG5vbmVA
        aW52YWxpZC5vcmcwggIiMA0GCSqGSIb3DQEBAQUAA4ICDwAwggIKAoICAQCjYtuW
        aICCY0tJPubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8
        Di3DAmHKnSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTek
        LWcfI5ZZtoGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnH
        KT/m6DSU0xlZ13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6
        jzhNyMBTJ1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQ
        Ujy5N8pSNp7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAz
        TK4l2pHNuC53QVc/EF++GBLAxmvCDq9ZpMIYi7OmzkkAKKC9Ue6Ef217LFQCFIBK
        Izv9cgi9fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqnV0
        nPN1IMSnzXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnWOFIaVrKWLzAtgzo
        GMTvP/AuehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9p
        sNcjTMaBQLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABo1MwUTAdBgNVHQ4EFgQU
        9mYU23tW2zsomkKTAXarjr2vjuswHwYDVR0jBBgwFoAU9mYU23tW2zsomkKTAXar
        jr2vjuswDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAgEANjIBMrow
        YNCbhAJdP7dhlhT2RUFRdeRUJD0IxrH/hkvb6myHHnK8nOYezFPjUlmRKUgNEDuA
        xbnXZzPdCRNV9V2mShbXvCyiDY7WCQE2Bn44z26O0uWVk+7DNNLH9BnkwUtOnM9P
        wtmD9phWexm4q2GnTsiL6Ul6cy0QlTJWKVLEUQQ6yda582e23J1AXqtqFcpfoE34
        H3afEiGy882b+ZBiwkeV+oq6XVF8sFyr9zYrv9CvWTYlkpTQfLTZSsgPdEHYVcjv
        xQ2D+XyDR0aRLRlvxUa9dHGFHLICG34Juq5Ai6lM1EsoD8HSsJpMcmrH7MWw2cKk
        ujC3rMdFTtte83wF1uuF4FjUC72+SmcQN7A386BC/nk2TTsJawTDzqwOu/VdZv2g
        1WpTHlumlClZeP+G/jkSyDwqNnTu1aodDmUa4xZodfhP1HWPwUKFcq8oQr148QYA
        AOlbUOJQU7QwRWd1VbnwhDtQWXC92A2w1n/xkZSR1BM/NUSDhkBSUU1WjMbWg6Gg
        mnIZLRerQCu1Oozr87rOQqQakPkyt8BUSNK3K42j2qcfhAONdRl8Hq8Qs5pupy+s
        8sdCGDlwR3JNCMv6u48OK87F4mcIxhkSefFJUFII25pCGN5WtE4p5l+9cnO1GrIX
        e2Hl/7M0c/lbZ4FvXgARlex2rkgS0Ka06HE=
        -----END CERTIFICATE-----
```

Save the ignition file to the following path on the target machine:

Code · bash

```
C:\opswat\md_cluster_identity_service.yml
```

> **Info**
> The ignition file contains sensitive credentials. This file can be safely deleted any time after the installation is complete.

Install the service

1. Copy the installer file (`.msi`) to the target machine.
2. Open **PowerShell** with **Administrator privileges**.
3. Run the following command to start the installation in **silent mode**:

Code · powershell

```
msiexec.exe /i <md_cluster_identity_service_package>.msi /qn
```

Verify the service status

1. Open **PowerShell** and run the following command:

Code · powershell

```
Get-Service -Name md-cluster-identity-service
```

2. Check the **Running** Status in the output.
3. If the service is not running, start it manually:

Code · powershell

```
Start-Service -Name md-cluster-identity-service
```

Service management

Action	Command (PowerShell)
Check service status	`Get-Service -Name md-cluster-identity-service`

Action	Command (PowerShell)
Start service	`Start-Service -Name md-cluster-identity-service`
Stop service	`Stop-Service -Name md-cluster-identity-service`
Restart service	`Restart-Service -Name md-cluster-identity-service`

Customize the service configuration

During installation, MD Cluster Identity Service generates a configuration file at:

```
C:\Program Files\OPSWAT\MetaDefender Cluster Identity Service\md_cluster_identity_service.yml
```

To customize the service behavior:

1. Open the configuration file in a text editor.
2. Modify the required settings according to your environment.
3. Save the changes.
4. Restart the service to apply the new settings.

```powershell
Restart-Service -Name md-cluster-identity-service
```

Directory structure

- `C:\opswat\md_cluster_file_storage.yml`: Service Ignition file.
- `C:\Program Files\OPSWAT\MetaDefender Cluster Identity Service\md_cluster_file_storage.yml`: Service configuration file.
- `C:\Program Files\OPSWAT\MetaDefender Cluster Identity Service\data\log\identity-service.log` : Service log file.
- `C:\Program Files\OPSWAT\MetaDefender Cluster Identity Service\data\storage`: Default storage directory.
- `C:\Program Files\OPSWAT\MetaDefender Cluster Identity Service\data\log` Default log directory.

Log files

To check the service log, open the file `C:\Program Files\OPSWAT\MetaDefender Cluster Identity Service\data\log\identity-service.log`.

To check logs using Event Viewer:

1. Open **Event Viewer**.
2. Navigate to **Windows Logs > Application**.
3. Look for events related to **MD Cluster Identiy Service.**

Uninstall the service

Open **PowerShell** and run the following command:

Code · powershell

```
Uninstall-Package -Name 'MetaDefender Cluster Identity Service' -Force
```

Troubleshooting

A. Service is not running

1. Check the service status

Code · powershell

```
Get-Service -Name md-cluster-identity-service
```

2. Start the service if it is not running:

Code · powershell

```
Start-Service -Name md-cluster-identity-service
```

B. Installation fails

Possible causes

- Insufficient privileges.
- Missing dependencies.

Solution

- Ensure the installation command is executed with **Administrator** privileges.
- Ensure dependencies are installed.

C. MD Cluster Control Center cannot connect to MD Cluster Identity Service.

Possible causes

- Network connectivity issues.
- Firewall restrictions.

Solution

- Ensure MD Cluster Control Center has network connectivity to MD Cluster Identity Service.
- Verify that firewall rules allow **inbound** and **outbound** connections.

Ignition file key reference

- `secure.connection_key` (**Required**)
  - Value type: string.
  - Description: Use a **4–64 character string** that contains only numbers (0–9) and letters (a–z, A–Z). This string is used by clients to connect to the server. Set this value as the `identity.connection_key` in the MD Cluster Control Center configuration file.
- `secure.private_key` (**Required**)
  - Value type: string.
  - Description: The content of private key in X509 format.
- `secure.certificate` (**Required**)
  - Value type: string.
  - Description: The content of certificate in X509 format.
- `database.host` (**Required**)
  - Value type: string.
  - Description: IP address or domain name of the server hosting PostgreSQL.
- `database.port` (**Required**)
  - Value type: string.
  - Description: Port where the PostgreSQL server listens for client connections.
- `database.user` (**Required**)
  - Value type: string.

- Description: PostgreSQL server user. SUPERUSER privileges are required to set up the database and extensions during the initial configuration.

- `database.password` (**Required**)
  - Value type: string.
  - Description: PostgreSQL user password.

- `rest.host` [optional)
  - Value type: string.
  - Description: IP address (V4/V6) or host where the server resides on. Default value is '*'. Notes: `'*'` allows the service to accept connections from all network interfaces. To bind the service to a specific interface, specify its IP address or domain name. For example, to listen on all IPv4 interfaces, set the host to `0.0.0.0`.

- `rest.port` [optional)
  - Value type: number.
  - Description: The port where the server resides on. Default value is `8891`.

- `log.streams[@].log_type` [optional)
  - Value type: string.
  - Description: Type of log device (`file`, or `syslog`)

- `log.streams[@].log_level` [optional)
  - Value type: string.
  - Description: Level of log message (`dump`, `debug`, `info`, `warning`, or `error`).

- `log.streams[@].log_path` [optional)
  - Value type: string.
  - Description: Location where logs are written. If `log.streams[@].log_type` is `"file"` then `log.streams[@].log_path` is the path to a file on file system where logs are written. If `log.streams[@].log_type` is `"syslog"` then
    - `log.streams[@].log_path` can be `[tcp/udp]://host:port` where host:port is the host and port to a remote syslog server that supports TCP or UDP protocol.
    - `log.streams[@].log_path` can be `"local"` to write log to local syslog server (Linux only).

- `user.name` [optional)
  - Value type: string.
  - Description: Username for the initial administrator account.

- `user.password` [optional)

- Value type: string.
- Description: Password for the initial administrator account.

- `user.email` [optional)
  - Value type: string.
  - Description: Email address for the initial administrator account.

- `user.apikey` [optional)
  - Value type: string.
  - Description: API key for the initial administrator account.

Linux

Overview

This section describes how to install and configure **MetaDefender (MD) Cluster Identity Service** on supported **Linux** distributions. After installation, MD Cluster Control Center can connect to MD Cluster Identity and monitor its system health.

Prerequisites

Before installing the MD Cluster Identity Service, ensure the following requirements are met.

Requirement	Description
Operating System	Ubuntu 22.04+, Ubuntu 24.04+, Debian 12+, Rocky 9+, or RHEL 9+.
Privileges	Root or sudo privileges
Installation package	Debian/Ubuntu: `md-cluster-identity-service_<version>-1_amd64.deb` Rocky/RHEL: `md-cluster-identity-service-<version>-1.x86_64.rpm`
Network access	Required port is open (default port: **8891**). A minimum network bandwidth of **1 Gbps** is required.
Hardware	4vCPU and 4GB RAM
Disk space	A minimum of **50 GB of available disk space** is required.

Create the ignition file

Create an ignition file in YAML format. This file contains the credentials required for the service to connect to the system.

The file must include the following keys:

Key	Description
`database.host`	IP address or domain name of the server hosting PostgreSQL.
`database.port`	IP address or domain name of the server hosting PostgreSQL.
`database.user`	PostgreSQL server user. SUPERUSER privileges are required to set up the database and extensions during the initial configuration.
`database.password`	PostgreSQL user password.
`secure.connection_key`	A **4–64 character alphanumeric string** (a–z, A–Z, 0–9) used by MD Cluster Control Center to connect to the server.
`secure.private_key`	The content of private key in X509 format.
`secure.certificate`	The content of certificate in X509 format.

Example ignition file:

Code · yaml

```yaml
database:
  host: "posgresql_host"
  port: 5432
  user: "postgres"
  password: "admin"
secure:
  connection_key: "1234abcd" # [0-9a-zA-Z]{4,64}
  private_key: |
        -----BEGIN PRIVATE KEY-----
        MIIJQwIBADANBgkqhkiG9w0BAQEFAASCCS0wggkpAgEAAoICAQCjYtuWaICCY0tJ
        PubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8Di3DAmHK
        nSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTekLWcfI5ZZ
```

```
        toGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnHKT/m6DSU
        0xlZ13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6jzhNyMBT
        J1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQUjy5N8pS
        Np7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAzTK4l2pHN
        uC53QVc/EF++GBLAxmvCDq9ZpMIYi7OmzkkAKKC9Ue6Ef217LFQCFIBKIzv9cgi9
        fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqnV0nPN1IMSn
        zXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnWOFIaVrKWLzAtgzoGMTvP/Au
        ehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9psNcjTMaB
        QLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABAoICAFWe8MQZb37k2gdAV3Y6aq8f
        qokKQqbCNLd3giGFwYkezHXoJfg6Di7oZxNcKyw35LFEghkgtQqErQqo35VPIoH+
        vXUpWOjnCmM4muFA9/cX6mYMc8TmJsg0ewLdBCOZVw+wPABlaqz+0UOiSMMftpk9
        fz9JwGd8ERyBsT+tk3Qi6D0vPZVsC1KqxxL/cwIFd3Hf2ZBtJXe0KBn1pktWht5A
        Kqx9mld2Ovl7NjgiC1Fx9r+fZw/iOabFFwQA4dr+R8mEMK/7bd4VXfQ1o/QGGbMT
        G+ulFrsiDyP+rBIAaGC0i7gDjLAIBQeDhP409ZhswIEc/GBtODU372a2CQK/u4Q/
        HBQvuBtKFNkGUooLgCCbFxzgNUGc83GB/6IwbEM7R5uXqsFiE71LpmroDyjKTlQ8
        YZkpIcLNVLw0usoGYHFm2rvCyEVlfsE3Ub8cFyTFk50SeOcF2QL2xzKmmbZEpXgl
        xBHR0hjgon0IKJDGfor4bHO7Nt+1Ece8u2oTEKvpz5aIn44OeC5mApRGy83/0bvs
        esnWjDE/bGpoT8qFuy+0urDEPNId44XcJm1IRIlG56ErxC3l0s11wrIpTmXXckqw
        zFR9s2z7f0zjeyxqZg4NTPI7wkM3M8BXlvp2GTBIeoxrWB4V3YArwu8QF80QBgVz
        mgHl24nTg00UH1OjZsABAoIBAQDOxftSDbSqGytcWqPYP3SZHAWDA0O4ACEM+eCw
        au9ASutl0IDlNDMJ8nC2ph25BMe5hHDWp2cGQJog7pZ/3qQogQho2gUniKDifN77
        40QdykllTzTVROqmP8+efreIvqlzHmuqaGfGs5oTkZaWj5su+B+bT+9rIwZcwfs5
        YRINhQRx17qa++xh5mfE25c+M9fiIBTiNSo4lTxWMBShnK8xrGaMEmN7W0qTMbFH
        PgQz5FcxRjCCqwHilwNBeLDTp/ZECEB7y34khVh531mBE2mNzSVIQcGZP1I/DvXj
        W7UUNdgFwii/GW+6M0uUDy23UVQpbFzcV8o1C2nZc4Fb4zwBAoIBAQDKSJkFwwuR
        naVJS6WxOKjX8MCu9/cKPnwBv2mmI2jgGxHTw5sr3ahmF5eTb8Zo19BowytN+tr6
        2ZFoIBA9Ubc9esEAU8l3fggdfM82cuR9sGcfQVoCh8tMg6BP8IBLOmbSUhN3PG2m
        39I802u0fFNVQCJKhx1m1MFFLOu7lVcDS9JN+oYVPb6MDfBLm5jOiPuYkFZ4gH79
        J7gXI0/YKhaJ7yXthYVkdrSF6Eooer4RZgma62Dd1VNzSq3JBo6rYjF7Lvd+RwDC
        R1thHrmf/IXplxpNVkoMVxtzbrrbgnC25QmvRYc0rlS/kvM4yQhMH3eA7IycDZMp
        Y+0xm7I7jTT7AoIBAGKzKIMDXdCxBWKhNYJ8z7hiItNl1IZZMW2TPUiY0rl6yaCh
        BVXjM9W0r07QPnHZsUiByqb743adkbTUjmxdJzjaVtxN7ZXwZvOVrY7I7fPWYnCE
        fXCr4+IVpZI/ZHZWpGX6CGSgT6EOjCZ5IUufIvEpqVSmtF8MqfXO9o9uIYLokrWQ
        x1dBl5UnuTLDqw8bChq7O5y6yfuWaOWvL7nxI8NvSsfj4y635gIa/0dFeBYZEfHI
        UlGdNVomwXwYEzgE/c19ruIowX7HU/NgxMWTMZhpazlxgesXybel+YNcfDQ4e3RM
        OMz3ZFiaMaJsGGNf4++d9TmMgk4Ns6oDs6Tb9AECggEBAJYzd+SOYo26iBu3nw3L
        65uEeh6xou8pXH0Tu4gQrPQTRZZ/nT3iNgOwqu1gRuxcq7TOjt41UdqIKO8vN7/A
        aJavCpaKoIMowy/aGCbvAvjNPpU3unU8jdl/t08EXs79S5IKPcgAx87sTTi7KDN5
        SYt4tr2uPEe53NTXuSatilG5QCyExIELOuzWAMKzg7CAiIlNS9foWeLyVkBgCQ6S
        me/L8ta+mUDy37K6vC34jh9vK9yrwF6X44ItRoOJafCaVfGI+175q/eWcqTX4q+I
        G4tKls4sL4mgOJLq+ra50aYMxbcuommctPMXU6CrrYyQpPTHMNVDQy2ttFdsq9iK
        TncCggEBAMmt/8yvPflS+xv3kg/ZBvR9JB1In2n3rUCYYD47ReKFqJ03Vmq5C9nY
        56s9w7OUO8perBXlJYmKZQhO4293lvxZD2Iq4NcZbVSCMoHAUzhzY3brdgtSIxa2
        gGveGAezZ38qKIU26dkz7deECY4vrsRkwhpTW0LGVCpjcQoaKvymAoCmAs8V2oMr
        Ziw1YQ9uOUoWwOqm1wZqmVcOXvPIS2gWAs3fQlWjH9hkcQTMsUaXQDOD0aqkSY3E
        NqOvbCV1/oUpRi3076khCoAXI1bKSn/AvR3KDP14B5toHI/F5OTSEiGhhHesgRrs
        fBrpEY1IATtPq1taBZZogRqI3rOkkPk=
        -----END PRIVATE KEY-----
certificate: |
        -----BEGIN CERTIFICATE-----
        MIIF5jCCA86gAwIBAgIJANq50IuwPFKgMA0GCSqGSIb3DQEBCwUAMIGGMQswCQYD
```

```
VQQGEwJHQjEQMA4GA1UECAwHRXJld2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEb
MBkGA1UECgwSbGlid2Vic29ja2V0cy10ZXN0MRIwEAYDVQQDDAlsb2NhbGhvc3Qx
HzAdBgkqhkiG9w0BCQEWEG5vbmVAaW52YWxpZC5vcmcwIBcNMTgwMzIwMDQxNjA3
WhgPMjExODAyMjQwNDE2MDdaMIGGMQswCQYDVQQGEwJHQjEQMA4GA1UECAwHRXJl
d2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEbMBkGA1UECgwSbGlid2Vic29ja2V0
cy10ZXN0MRIwEAYDVQQDDAlsb2NhbGhvc3QxHzAdBgkqhkiG9w0BCQEWEG5vbmVA
aW52YWxpZC5vcmcwggIiMA0GCSqGSIb3DQEBAQUAA4ICDwAwggIKAoICAQCjYtuW
aICCY0tJPubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8
Di3DAmHKnSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTek
LWcfI5ZZtoGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnH
KT/m6DSU0xlZ13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6
jzhNyMBTJ1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQ
Ujy5N8pSNp7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAz
TK4l2pHNuC53QVc/EF++GBLAxmvCDq9ZpMIYi7OmzkkAKKC9Ue6Ef217LFQCFIBK
Izv9cgi9fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqnV0
nPN1IMSnzXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnWOFIaVrKWLzAtgzo
GMTvP/AuehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9p
sNcjTMaBQLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABo1MwUTAdBgNVHQ4EFgQU
9mYU23tW2zsomkKTAXarjr2vjuswHwYDVR0jBBgwFoAU9mYU23tW2zsomkKTAXar
jr2vjuswDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAgEANjIBMrow
YNCbhAJdP7dhlhT2RUFRdeRUJD0IxrH/hkvb6myHHnK8nOYezFPjUlmRKUgNEDuA
xbnXZzPdCRNV9V2mShbXvCyiDY7WCQE2Bn44z26O0uWVk+7DNNLH9BnkwUtOnM9P
wtmD9phWexm4q2GnTsiL6Ul6cy0QlTJWKVLEUQQ6yda582e23J1AXqtqFcpfoE34
H3afEiGy882b+ZBiwkeV+oq6XVF8sFyr9zYrv9CvWTYlkpTQfLTZSsgPdEHYVcjv
xQ2D+XyDR0aRLRlvxUa9dHGFHLICG34Juq5Ai6lM1EsoD8HSsJpMcmrH7MWw2cKk
ujC3rMdFTtte83wF1uuF4FjUC72+SmcQN7A386BC/nk2TTsJawTDzqwOu/VdZv2g
1WpTHlumlClZeP+G/jkSyDwqNnTu1aodDmUa4xZodfhP1HWPwUKFcq8oQr148QYA
AOlbUOJQU7QwRWd1VbnwhDtQWXC92A2w1n/xkZSR1BM/NUSDhkBSUU1WjMbWg6Gg
mnIZLRerQCu1Oozr87rOQqQakPkyt8BUSNK3K42j2qcfhAONdRl8Hq8Qs5pupy+s
8sdCGDlwR3JNCMv6u48OK87F4mcIxhkSefFJUFII25pCGN5WtE4p5l+9cnO1GrIX
e2Hl/7M0c/lbZ4FvXgARlex2rkgS0Ka06HE=
-----END CERTIFICATE-----
```

Save the ignition file to the following path on the target machine:

Code · bash

```
/etc/opswat/md_cluster_identity_service.yml
```

Info

The ignition file contains sensitive credentials. This file can be safely deleted any time after the installation is complete.

Install the service

1. Copy the installer file (`.deb` or `.rpm`) to the target machine.

2. Open **Terminal**.

3. Run the following command to start the installation:

Code · bash

```
# Debian or Ubuntu
sudo apt install -y uuid tar
sudo dpkg -i <md_cluster_identity_service_package> || sudo apt install -f

# Rocky or RHEL
sudo dnf install -y yum-utils
sudo dnf config-manager --set-enabled devel
sudo dnf update -y
sudo dnf install -y redhat-lsb-core libuuid tar
sudo yum install <md_cluster_identity_service_package> -y
```

Verify the service status

1. Open **Terminal** and run the following command:

Code · bash

```
sudo systemctl status md-cluster-identity-service
```

2. Check the **active (running)** field in the output.

3. If the service is not running, start it manually:

Code · bash

```
sudo systemctl restart md-cluster-identity-service
```

4. To ensure the service starts automatically at system boot:

Code · bash

```
sudo systemctl enable md-cluster-identity-service
```

Service management

Action	Command

Action	Command
Check service status	`sudo systemctl status md-cluster-identity-service`
Start service	`sudo systemctl start md-cluster-identity-service`
Stop service	`sudo systemctl stop md-cluster-identity-service`
Restart service	`sudo systemctl restart md-cluster-identity-service`
Enable service at boot	`sudo systemctl enable md-cluster-identity-service`

Customize the service configuration

During installation, MD Cluster Identity Service generates a configuration file at:

Code · bash

```
/etc/md-cluster-identity-service/md_cluster_identity_service.yml
```

To customize the service behavior:

1. Open the configuration file in a text editor such as **nano**.

Code · bash

```
sudo nano /etc/md-cluster-identity-service/md_cluster_identity_service.yml
```

2. Modify the required settings according to your environment.

3. Save the changes.

4. Restart the service to apply the new settings.

Code · bash

```
sudo systemctl restart md-cluster-identity-service
```

Directory structure

- `/etc/opswat/md_cluster_identity_service.yml`: Service Ignition file.
- `/etc/md-cluster-identity-service/md_cluster_identity_service.yml`: Service configuration file.
- `/var/log/md-cluster-identity-service/`: Default log directory.
- `/var/lib/md-cluster-identity-service/`: Contains persistent data required for the service to maintain state across reboots.

Log files

To check the service logs, open the file: `/var/log/md-cluster-identity-service/identity-service.log`

To check the system log, run the following in Terminal:

Code · bash

```
# Fetch by systemd-journald
sudo journalctl -r

# Ubuntu syslog
sudo cat /var/log/syslog

# Rocky or RHEL syslog
sudo cat /var/log/message
```

Uninstall the service

Code · bash

```
# Debian or Ubuntu
sudo apt purge <md_cluster_identity_service_package>

# Rocky or RHEL
sudo yum remove <md_cluster_identity_service_package>
```

Troubleshooting

A. Service is not running

1. Check the service status

Code · bash

```
sudo systemctl status md-cluster-identity-service
```

2. Start the service if it is not running:

```
Code · bash

sudo systemctl start md-cluster-identity-service
```

B. Installation fails

Possible causes

- Insufficient privileges.
- Missing dependencies.

Solution

- Ensure the installation command is executed with **sudo**.
- Ensure dependencies are installed.

C. MD Cluster Control Center cannot connect to MD Cluster Identity Service.

Possible causes

- Network connectivity issues.
- Firewall restrictions.
- PostgreSQL database connectivity issues

Solution

- Ensure MD Cluster Control Center has network connectivity to MD Cluster Identity Service.
- Verify that firewall rules allow **inbound** and **outbound** connections.
- Verify PostgreSQL database connectivity and credentials in the ignition file.

Ignition file key reference

- `database.host` (**Required**)
  - Value type: string.
  - Description: IP address or domain name of the server hosting PostgreSQL.
- `database.port` (**Required**)
  - Value type: integer.
  - Description: PostgreSQL server port (default: 5432).

- `database.user` (**Required**)
  - Value type: string.
  - Description: PostgreSQL server user. SUPERUSER privileges are required to set up the database and extensions during the initial configuration.
- `database.password` (**Required**)
  - Value type: string.
  - Description: PostgreSQL user password.
- `secure.connection_key` (**Required**)
  - Value type: string.
  - Description: Use a **4–64 character string** that contains only numbers (0–9) and letters (a–z, A–Z). This string is used by clients to connect to the server. Set this value as the `identity.connection_key` in the MD Cluster Control Center configuration file.
- `secure.private_key` (**Required**)
  - Value type: string.
  - Description: The content of private key in X509 format.
- `secure.certificate` (**Required**)
  - Value type: string.
  - Description: The content of certificate in X509 format.

MD Cluster Control Center

Supported Operating Systems

OS	Version
Windows	Windows 11 23H2+, Windows Server 2019+.
Ubuntu	Ubuntu 22.04+, Ubuntu 24.04+, Debian 12+, Rocky 9+, or RHEL 9+.

Windows

Overview

This section describes how to install and configure **MetaDefender (MD) Cluster Control Center** on **Windows**

systems. After installation, administrators can use MD Cluster Control Center to manage MD Cluster services from a centralized interface.

Prerequisites

Before installing the MD Cluster Control Center service, ensure the following requirements are met.

Requirement	Description
Operating System	Windows 11 23H2+, Windows Server 2019+.
Privileges	Administrator privileges
Installation package	`md-cluster-control-center-<version>-1-x64.msi`
Network access	Required port is open (default port: **8892**).
Dependencies	MD Cluster Identity Service and PostgreSQL must be installed and reachable from the Control Center host.
Hardware	4vCPU and 4GB RAM
Disk space	A minimum of **100 GB of available disk space** is required.

Create the ignition file

Create an ignition file in YAML format. This file contains the credentials and connection settings required for the service.

The file must include the following keys:

Key	Description
`identity.host`	IP address or domain name of the MD Cluster Identity Service host. **Do not** use `localhost` or `127.0.0.1`.
`identity.port`	Port used by MD Cluster Identity Service.
`identity.connection_key`	A **4-64 character alphanumeric string** (`a-z`, `A-Z`, `0-9`) that matches the Identity Service connection key.

Key	Description
`database.host`	IP address or domain name of the PostgreSQL server.
`database.port`	Port used by PostgreSQL.
`database.user`	PostgreSQL user. SUPERUSER privileges are required during the initial setup.
`database.password`	PostgreSQL user password.
`secure.encryption_key`	A **32-character** plain text key containing only lowercase letters and digits.

Example ignition file:

Code · yaml

```yaml
database:
  host: "your_postgres_host_ip"
  port: 5432
  user: "your_postgres_username"
  password: "your_postgres_admin_password"
identity:
  host: "your_md_cluster_identity_service_host_ip"
  port: 8891
  connection_key: "1234abcd"
secure:
  encryption_key: "12345678123456781234567812345678" # [a-z0-9]{32}
```

Save the ignition file to the following path on the target machine:

Code · powershell

```powershell
C:\opswat\md_cluster_control_center.yml
```

Info

The ignition file contains sensitive credentials. This file can be safely deleted any time after the installation is complete.

Install the service

1. Copy the installer file (`.msi`) to the target machine.

2. Open **PowerShell** with **Administrator privileges**.

3. Run the following command to start the installation in **silent mode**:

```powershell
msiexec.exe /i <md_cluster_control_center_package>.msi /qn
```

Verify the service status

1. Open **PowerShell** and run the following command:

```powershell
Get-Service -Name md-cluster-control-center
```

2. Check the **Running** status in the output.

3. If the service is not running, start it manually:

```powershell
Start-Service -Name md-cluster-control-center
```

Service management

Action	Command (PowerShell)
Check service status	`Get-Service -Name md-cluster-control-center`
Start service	`Start-Service -Name md-cluster-control-center`
Stop service	`Stop-Service -Name md-cluster-control-center`
Restart service	`Restart-Service -Name md-cluster-control-center`

Customize the service configuration

During installation, MD Cluster Control Center generates a configuration file at:

```
Code

C:\Program Files\OPSWAT\MetaDefender Cluster Control Center\md_cluster_control_center.yml
```

To customize the service behavior:

1. Open the configuration file in a text editor.
2. Modify the required settings according to your environment.
3. Save the changes.
4. Restart the service to apply the new settings.

```
Code · powershell

Restart-Service -Name md-cluster-control-center
```

Directory structure

- `C:\opswat\md_cluster_control_center.yml`: Service ignition file.
- `C:\Program Files\OPSWAT\MetaDefender Cluster Control Center\md_cluster_control_center.yml`: Service configuration file.
- `C:\Program Files\OPSWAT\MetaDefender Cluster Control Center\data\log\control-center.log`: Service log file
- `C:\Program Files\OPSWAT\MetaDefender Cluster Control Center\data\log`: Default log directory.

Log files

To check the service log, open the file `C:\Program Files\OPSWAT\MetaDefender Cluster Control Center\data\log\control-center.log`.

To check logs using Event Viewer:

1. Open **Event Viewer**.
2. Navigate to **Windows Logs > Application**.
3. Look for events related to **MD Cluster Control Center** service.

Uninstall the service

Open **PowerShell** and run the following command:

Code · powershell

```
Uninstall-Package -Name 'MetaDefender Cluster Control Center' -Force
```

Troubleshooting

A. Service does not start

1. Check the service status:

Code · powershell

```
Get-Service -Name md-cluster-control-center
```

2. Review the application log and Event Viewer entries for Control Center errors.

3. Verify that the configuration file contains valid values, then restart the service.

B. Installation fails

Possible causes:

- Insufficient privileges.
- Missing prerequisites.
- The installer package does not match the target platform.

Solution:

- Ensure the installation command is executed with **Administrator** privileges.
- Confirm MD Cluster Identity Service and PostgreSQL are available.
- Re-download or verify the installer package if the MSI exits unexpectedly.

C. Control Center cannot connect to Identity Service

Possible causes:

- `identity.host` is incorrect.
- Network connectivity issues.
- Firewall restrictions.
- `identity.connection_key` does not match the value configured on MD Cluster Identity Service.

Solution:

- Verify `identity.host`, `identity.port`, and `identity.connection_key` in the ignition or configuration file.
- Ensure MD Cluster Identity Service is running and reachable from the MD Cluster Control Center host.
- Verify Windows Firewall rules allow traffic between the two services.

- Do not use `localhost` or `127.0.0.1` for `identity.host` unless both services run on the same host and the deployment explicitly supports it.

D. Database connection fails

Possible causes:

- PostgreSQL is not running.
- Database credentials are incorrect.
- PostgreSQL is not reachable from MD Cluster Control Center host.

Solution:

- Verify PostgreSQL service status.
- Confirm `database.host`, `database.port`, `database.user`, and `database.password` are correct.
- Ensure firewall rules allow access to PostgreSQL.

Ignition file key reference

- `identity.host` (**Required**)
  - Value type: string.
  - Description: IP address or domain name of the server hosting MD Cluster Identity Service. **Avoid** using `localhost` or `127.0.0.1`.
- `identity.port` (**Required**)
  - Value type: number.
  - Description: Port where MD Cluster Identity Service listens for client connections.
- `identity.connection_key` (**Required**)
  - Value type: string.
  - Description: A **4-64 character** string that contains only digits (**0-9**) and letters (**a-z, A-Z**). This value must match the connection key configured on MD Cluster Identity Service.
- `database.host` (**Required**)
  - Value type: string.
  - Description: IP address or domain name of the PostgreSQL server.
- `database.port` (**Required**)
  - Value type: number.
  - Description: Port where PostgreSQL listens for client connections.

- `database.user` (**Required**)
  - Value type: string.
  - Description: PostgreSQL user account. SUPERUSER privilege is required during the initial setup.
- `database.password` (**Required**)
  - Value type: string.
  - Description: PostgreSQL user password.
- `secure.encryption_key` (**Required**)
  - Value type: string.
  - Description: A 32-character plain text key composed only of lowercase letters (**0-9]** and digits (**0-9**).
- `rest.port` (Optional)
  - Value type: number.
  - Description: Port where MD Cluster Control Center listens for incoming connections. Default value is `8892`.
- `rest.log_path` (Optional)
  - Value type: string.
  - Description: Location where REST logs are written.
- `rest.log_level` (Optional)
  - Value type: string.
  - Description: Level of REST log messages (`dump`, `debug`, `info`, `warning`, or `error`).
- `log.streams[@].log_type` (Optional)
  - Value type: string.
  - Description: Type of log device (`file` or `syslog`).
- `log.streams[@].log_level` (Optional)
  - Value type: string.
  - Description: Level of log message (`dump`, `debug`, `info`, `warning`, or `error`).
- `log.streams[@].log_path` (Optional)
  - Value type: string.
  - Description: Location where logs are written. If `log.streams[@].log_type` is `"file"`, the value is a file path. If `log.streams[@].log_type` is `"syslog"`, the value can be `[tcp/udp]://host:port` for a remote syslog server or `"local"` for the local syslog server on supported platforms.
- `service` (optional)

- Value type: object.
- Description: Configures service connections that are automatically created the first time MD Cluster Control Center starts. This section can include one or more of the following service types: `cache`, `datalake`, `warehouse`, `file_storage`, and `broker`.

- `service.cache` (optional) — Redis cache service
  - Value type: object.
  - Description: Defines one or more Redis cache service connections.
  - `service.cache.connections[@].display_name` (Required)
    - Value type: string.
    - Description: The display name of the connection. The value must not be empty.
  - `service.cache.connections[@].host` (Required)
    - Value type: string.
    - Description: The hostname or IP address of the Redis server.
  - `service.cache.connections[@].port` (Required)
    - Value type: number.
    - Description: The port used by the Redis server. A valid value ranges from **1** to **65535**.
  - `service.cache.connections[@].user` (optional)
    - Value type: string.
    - Description: The Redis username.
  - `service.cache.connections[@].password` (optional)
    - Value type: string.
    - Description: The Redis password.
  - `service.cache.connections[@].secure` (optional)
    - Value type: boolean.
    - Description: Enables a TLS connection to the Redis server. Defaults to false. When enabled, port must be the Redis TLS port.
  - `service.cache.connections[@].ssl_verify` (optional)
    - Value type: boolean.
    - Description: Verifies the Redis server's certificate. Defaults to true. Only applies when secure is enabled.
  - `service.cache.connections[@].ca_cert` (optional)
    - Value type: string.

  - Description: The path to the CA certificate file, read on the Control Center host. If empty, the system trust store is used.

- `service.datalake` (optional)
  - Value type: object.
  - Description: Defines one or more PostgreSQL Data Lake service connections.
  - `service.datalake.connections[@].display_name` (Required)
    - Value type: string.
    - Description: The display name of the connection. The value must not be empty.
  - `service.datalake.connections[@].host` (Required)
    - Value type: string.
    - Description: The hostname or IP address of the PostgreSQL server.
  - `service.datalake.connections[@].port` (Required)
    - Value type: number.
    - Description: The port used by the PostgreSQL server. A valid value ranges from **1** to **65535**.
  - `service.datalake.connections[@].user` (Required)
    - Value type: string.
    - Description: The PostgreSQL username.
  - `service.datalake.connections[@].password` (Required)
    - Value type: string.
    - Description: The password for the PostgreSQL user.

- `service.warehouse` (optional) — PostgreSQL warehouse service
  - Value type: object.
  - Description: Defines one or more PostgreSQL Warehouse service connections.
  - `service.warehouse.connections[@].display_name` (Required)
    - Value type: string.
    - Description: The display name of the connection. The value must not be empty.
  - `service.warehouse.connections[@].host` (Required)
    - Value type: string.
    - Description: The hostname or IP address of the PostgreSQL server.

- `service.warehouse.connections[@].port` (Required)
  - Value type: number.
  - Description: The port used by the PostgreSQL server. A valid value ranges from **1** to **65535**.
- `service.warehouse.connections[@].user` (Required)
  - Value type: string.
  - Description: The PostgreSQL username.
- `service.warehouse.connections[@].password` (Required)
  - Value type: string.
  - Description: The password for the PostgreSQL user.

- `service.file_storage` (optional)
  - Value type: object.
  - Description: Defines one or more File Storage service connections.
  - `service.file_storage.connections[@].display_name` (Required)
    - Value type: string.
    - Description: The display name of the connection. The value must not be empty.
  - `service.file_storage.connections[@].host` (Required)
    - Value type: string.
    - Description: The hostname or IP address of the File Storage server.
  - `service.file_storage.connections[@].port` (Required)
    - Value type: number.
    - Description: The port used by the File Storage server. A valid value ranges from **1** to **65535**.
  - `service.file_storage.connections[@].connection_key` (Required)
    - Value type: string.
    - Description: The connection key used to authenticate with the File Storage service.

- `service.broker` (optional)
  - Value type: object.
  - Description: Defines one or more RabbitMQ broker service connections.
  - `service.broker.connections[@].display_name` (Required)
    - Value type: string.

  - Description: The display name of the connection. The value must not be empty.

- `service.broker.connections[@].host` (Required)
  - Value type: string.
  - Description: The hostname or IP address of the RabbitMQ server.
- `service.broker.connections[@].port` (Required)
  - Value type: number.
  - Description: The port used by the RabbitMQ server. A valid value ranges from **1** to **65535**.
- `service.broker.connections[@].user` (Required)
  - Value type: string.
  - Description: The RabbitMQ username.
- `service.broker.connections[@].password` (Required)
  - Value type: string.
  - Description: The password for the RabbitMQ user.
- `service.broker.connections[@].secure` (Optional)
  - Value type: boolean.
  - Description: Enables a TLS connection to the RabbitMQ server. Defaults to false. When enabled, port must be the broker's TLS port.
- `service.broker.connections[@].ssl_verify` (Optional)
  - Value type: boolean.
  - Description: Verifies the RabbitMQ server's certificate. Defaults to true. Only applies when secure is enabled.
- `service.broker.connections[@].ca_cert` (Optional)
  - Value type: string.
  - Description: The path to the CA certificate file, read on the Control Center host. If empty, the system trust store is used.

- `access.allowed_directories` (optional)
  - Value type: array of strings.
  - Description: Specifies up to **16** local directories from which MD Cluster Control Center is permitted to read certificate files. Only files located in the configured directories or their subdirectories are allowed. Each directory must exist when MD Cluster Control Center starts. If any configured directory is invalid or cannot be resolved, MD Cluster Control Center fails to start.

Linux

Overview

This section describes how to install and configure **MetaDefender (MD) Cluster Control Center** on supported **Linux** distributions. After installation, administrators can use MD Cluster Control Center to manage MD Cluster services from a centralized interface.

Prerequisites

Before installing the MD Cluster Control Center service, ensure the following requirements are met.

Requirement	Description
Operating System	Ubuntu 22.04+, Ubuntu 24.04+, Debian 12+, Rocky 9+, or RHEL 9+.
Privileges	Root or sudo privileges
Installation package	Debian/Ubuntu: `md-cluster-control-center_<version>-1_amd64.deb` Rocky/RHEL: `md-cluster-control-center-<version>-1.x86_64.rpm`
Network access	Required port is open (default port: **8892**).
Dependencies	MD Cluster Identity Service and PostgreSQL must be installed and reachable from the Control Center host.
Hardware	4vCPU and 4GB RAM
Disk space	A minimum of **100 GB of available disk space** is required.

Create the ignition file

Create an ignition file in YAML format. This file contains the credentials and connection settings required for the service.

The file must include the following keys:

Key	Description
`identity.host`	IP address or domain name of the MD Cluster Identity

Key	Description
	Service host. **Do not** use `localhost` or `127.0.0.1`.
`identity.port`	Port used by MD Cluster Identity Service.
`identity.connection_key`	A **4-64 character alphanumeric string** (`a-z`, `A-Z`, `0-9`) that matches the Identity Service connection key.
`database.host`	IP address or domain name of the PostgreSQL server.
`database.port`	Port used by PostgreSQL.
`database.user`	PostgreSQL user. SUPERUSER privileges are required during the initial setup.
`database.password`	PostgreSQL user password.
`secure.encryption_key`	A **32-character** plain text key containing only lowercase letters and digits.

Example ignition file:

Code · yaml

```yaml
database:
  host: "your_postgres_host_ip"
  port: 5432
  user: "your_postgres_username"
  password: "your_postgres_admin_password"
identity:
  host: "your_md_cluster_identity_service_host_ip"
  port: 8891
  connection_key: "1234abcd"
secure:
  encryption_key: "12345678123456781234567812345678" # [a-z0-9]{32}
```

Save the ignition file to the following path on the target machine:

Code · bash

```bash
/etc/opswat/md_cluster_control_center.yml
```

Info

The ignition file contains sensitive credentials. This file can be safely deleted any time after the installation is complete.

Install the service

1. Copy the installer file (`.deb` or `.rpm`) to the target machine.
2. Open **Terminal**.
3. Run the following command to start the installation:

Code · bash

```
# Debian or Ubuntu
sudo apt -y install uuid
sudo dpkg -i <md_cluster_control_center_package> || sudo apt install -f

# Rocky or RHEL
sudo dnf install -y yum-utils
sudo dnf config-manager --set-enabled devel
sudo dnf update -y
sudo dnf install -y redhat-lsb-core libuuid tar
sudo yum install <md_cluster_control_center_package> -y
```

Verify the service status

1. Open **Terminal** and run the following command:

Code · bash

```
sudo systemctl status md-cluster-control-center
```

2. Check the **active (running)** field in the output.
3. If the service is not running, start it manually:

Code · bash

```
sudo systemctl restart md-cluster-control-center
```

4. To ensure the service starts automatically at system boot:

Code · bash

```
sudo systemctl enable md-cluster-control-center
```

Service management

Action	Command
Check service status	`sudo systemctl status md-cluster-control-center`
Start service	`sudo systemctl start md-cluster-control-center`
Stop service	`sudo systemctl stop md-cluster-control-center`
Restart service	`sudo systemctl restart md-cluster-control-center`
Enable service at boot	`sudo systemctl enable md-cluster-control-center`

Customize the service configuration

During installation, MD Cluster Control Center generates a configuration file at:

Code · bash

```
/etc/md-cluster-control-center/md_cluster_control_center.yml
```

To customize the service behavior:

1. Open the configuration file in a text editor such as **nano**.

Code · bash

```
sudo nano /etc/md-cluster-control-center/md_cluster_control_center.yml
```

2. Modify the required settings according to your environment.

3. Save the changes.

4. Restart the service to apply the new settings.

Code · bash

```
sudo systemctl restart md-cluster-control-center
```

Directory structure

- `/etc/opswat/md_cluster_control_center.yml`: Service ignition file.
- `/etc/md-cluster-control-center/md_cluster_control_center.yml`: Service configuration file.
- `/var/log/md-cluster-control-center/`: Default log directory.
- `/var/lib/md-cluster-control-center/`: Contains persistent data required for the service to maintain state across reboots.

Log files

To check the service logs, open the file: `/var/log/md-cluster-control-center/control-center.log`.

To check the system log, run the following in Terminal:

Code · bash

```
# Fetch by systemd-journald
sudo journalctl -r

# Ubuntu syslog
sudo cat /var/log/syslog

# Rocky or RHEL syslog
sudo cat /var/log/message
```

Uninstall the service

Code · bash

```
# Debian or Ubuntu
sudo apt purge <md_cluster_control_center_package>

# Rocky or RHEL
sudo yum remove <md_cluster_control_center_package>
```

Troubleshooting

A. Service does not start

1. Check the service status:

```bash
sudo systemctl status md-cluster-control-center
```

2. Review recent service logs:

```bash
sudo journalctl -u md-cluster-control-center -n 100 --no-pager
```

3. Verify that the configuration file syntax is valid and restart the service after correcting any issue.

B. Installation fails

Possible causes:

- Insufficient privileges.
- Missing package dependencies.
- Required `uuid` package is not installed.

Solution:

- Ensure the installation commands are executed with **sudo**.
- Install dependencies and rerun the package installation.
- Confirm the package file matches the Linux distribution in use.

C. Control Center cannot connect to Identity Service

Possible causes:

- `identity.host` is incorrect.
- Network connectivity issues.
- Firewall restrictions.
- `identity.connection_key` does not match the value configured on MD Cluster Identity Service.

Solution:

- Verify `identity.host`, `identity.port`, and `identity.connection_key` in the ignition or configuration file.
- Ensure MD Cluster Identity Service is running and reachable from MD Cluster Control Center host.
- Verify that firewall rules allow traffic between the two services.
- Do not use `localhost` or `127.0.0.1` for `identity.host` unless both services run on the same host and the

deployment explicitly supports it.

D. Database connection fails

Possible causes:

- PostgreSQL is not running.
- Database credentials are incorrect.
- PostgreSQL is not reachable from MD Cluster Control Center host.

Solution:

- Verify PostgreSQL service status.
- Confirm `database.host`, `database.port`, `database.user`, and `database.password` are correct.
- Ensure PostgreSQL accepts remote connections and firewall rules allow access.

Ignition file key reference

- `identity.host` (**Required**)
  - Value type: string.
  - Description: IP address or domain name of the server hosting MD Cluster Identity Service. **Avoid** using `localhost` or `127.0.0.1`.
- `identity.port` (**Required**)
  - Value type: number.
  - Description: Port where MD Cluster Identity Service listens for client connections.
- `identity.connection_key` (**Required**)
  - Value type: string.
  - Description: A **4-64 character** string that contains only digits (**0-9**) and letters (**a-z, A-Z**). This value must match the connection key configured on MD Cluster Identity Service.
- `database.host` (**Required**)
  - Value type: string.
  - Description: IP address or domain name of the PostgreSQL server.
- `database.port` (**Required**)
  - Value type: number.
  - Description: Port where PostgreSQL listens for client connections.
- `database.user` (**Required**)

  - Value type: string.
  - Description: PostgreSQL user account. SUPERUSER privilege is required during the initial setup.

- `database.password` (**Required**)
  - Value type: string.
  - Description: PostgreSQL user password.

- `secure.encryption_key` (**Required**)
  - Value type: string.
  - Description: A 32-character plain text key composed only of lowercase letters (**0-9]** and digits (**0-9**).

- `rest.port` (Optional)
  - Value type: number.
  - Description: Port where MD Cluster Control Center listens for incoming connections. Default value is `8892`.

- `rest.log_path` (Optional)
  - Value type: string.
  - Description: Location where REST logs are written.

- `rest.log_level` (Optional)
  - Value type: string.
  - Description: Level of REST log messages (`dump`, `debug`, `info`, `warning`, or `error`).

- `log.streams[@].log_type` (Optional)
  - Value type: string.
  - Description: Type of log device (`file` or `syslog`).

- `log.streams[@].log_level` (Optional)
  - Value type: string.
  - Description: Level of log message (`dump`, `debug`, `info`, `warning`, or `error`).

- `log.streams[@].log_path` (Optional)
  - Value type: string.
  - Description: Location where logs are written. If `log.streams[@].log_type` is `"file"`, the value is a file path. If `log.streams[@].log_type` is `"syslog"`, the value can be `[tcp/udp]://host:port` for a remote syslog server or `"local"` for the local syslog server on supported platforms.

- `service` (optional)
  - Value type: object.

- Description: Configures service connections that are automatically created the first time MD Cluster Control Center starts. This section can include one or more of the following service types: `cache`, `datalake`, `warehouse`, `file_storage`, and `broker`.

- `service.cache` (optional) — Redis cache service
  - Value type: object.
  - Description: Defines one or more Redis cache service connections.
  - `service.cache.connections[@].display_name` (Required)
    - Value type: string.
    - Description: The display name of the connection. The value must not be empty.
  - `service.cache.connections[@].host` (Required)
    - Value type: string.
    - Description: The hostname or IP address of the Redis server.
  - `service.cache.connections[@].port` (Required)
    - Value type: number.
    - Description: The port used by the Redis server. A valid value ranges from **1** to **65535**.
  - `service.cache.connections[@].user` (optional)
    - Value type: string.
    - Description: The Redis username.
  - `service.cache.connections[@].password` (optional)
    - Value type: string.
    - Description: The Redis password.
  - `service.cache.connections[@].secure` (optional)
    - Value type: boolean.
    - Description: Enables a TLS connection to the Redis server. Defaults to false. When enabled, port must be the Redis TLS port.
  - `service.cache.connections[@].ssl_verify` (optional)
    - Value type: boolean.
    - Description: Verifies the Redis server's certificate. Defaults to true. Only applies when secure is enabled.
  - `service.cache.connections[@].ca_cert` (optional)
    - Value type: string.
    - Description: The path to the CA certificate file, read on the Control Center host. If empty, the system trust

store is used.

- `service.datalake` (optional)
  - Value type: object.
  - Description: Defines one or more PostgreSQL Data Lake service connections.
  - `service.datalake.connections[@].display_name` (Required)
    - Value type: string.
    - Description: The display name of the connection. The value must not be empty.
  - `service.datalake.connections[@].host` (Required)
    - Value type: string.
    - Description: The hostname or IP address of the PostgreSQL server.
  - `service.datalake.connections[@].port` (Required)
    - Value type: number.
    - Description: The port used by the PostgreSQL server. A valid value ranges from **1** to **65535**.
  - `service.datalake.connections[@].user` (Required)
    - Value type: string.
    - Description: The PostgreSQL username.
  - `service.datalake.connections[@].password` (Required)
    - Value type: string.
    - Description: The password for the PostgreSQL user.
- `service.warehouse` (optional) — PostgreSQL warehouse service
  - Value type: object.
  - Description: Defines one or more PostgreSQL Warehouse service connections.
  - `service.warehouse.connections[@].display_name` (Required)
    - Value type: string.
    - Description: The display name of the connection. The value must not be empty.
  - `service.warehouse.connections[@].host` (Required)
    - Value type: string.
    - Description: The hostname or IP address of the PostgreSQL server.
  - `service.warehouse.connections[@].port` (Required)

  - Value type: number.
  - Description: The port used by the PostgreSQL server. A valid value ranges from **1** to **65535**.
- `service.warehouse.connections[@].user` (Required)
  - Value type: string.
  - Description: The PostgreSQL username.
- `service.warehouse.connections[@].password` (Required)
  - Value type: string.
  - Description: The password for the PostgreSQL user.

- `service.file_storage` (optional)
  - Value type: object.
  - Description: Defines one or more File Storage service connections.
  - `service.file_storage.connections[@].display_name` (Required)
    - Value type: string.
    - Description: The display name of the connection. The value must not be empty.
  - `service.file_storage.connections[@].host` (Required)
    - Value type: string.
    - Description: The hostname or IP address of the File Storage server.
  - `service.file_storage.connections[@].port` (Required)
    - Value type: number.
    - Description: The port used by the File Storage server. A valid value ranges from **1** to **65535**.
  - `service.file_storage.connections[@].connection_key` (Required)
    - Value type: string.
    - Description: The connection key used to authenticate with the File Storage service.

- `service.broker` (optional)
  - Value type: object.
  - Description: Defines one or more RabbitMQ broker service connections.
  - `service.broker.connections[@].display_name` (Required)
    - Value type: string.
    - Description: The display name of the connection. The value must not be empty.

- `service.broker.connections[@].host` (Required)
  - Value type: string.
  - Description: The hostname or IP address of the RabbitMQ server.
- `service.broker.connections[@].port` (Required)
  - Value type: number.
  - Description: The port used by the RabbitMQ server. A valid value ranges from **1** to **65535**.
- `service.broker.connections[@].user` (Required)
  - Value type: string.
  - Description: The RabbitMQ username.
- `service.broker.connections[@].password` (Required)
  - Value type: string.
  - Description: The password for the RabbitMQ user.
- `service.broker.connections[@].secure` (Optional)
  - Value type: boolean.
  - Description: Enables a TLS connection to the RabbitMQ server. Defaults to false. When enabled, port must be the broker's TLS port.
- `service.broker.connections[@].ssl_verify` (Optional)
  - Value type: boolean.
  - Description: Verifies the RabbitMQ server's certificate. Defaults to true. Only applies when secure is enabled.
- `service.broker.connections[@].ca_cert` (Optional)
  - Value type: string.
  - Description: The path to the CA certificate file, read on the Control Center host. If empty, the system trust store is used.

- `access.allowed_directories` (optional)
  - Value type: array of strings.
  - Description: Specifies up to **16** local directories from which MD Cluster Control Center is permitted to read certificate files. Only files located in the configured directories or their subdirectories are allowed. Each directory must exist when MD Cluster Control Center starts. If any configured directory is invalid or cannot be resolved, MD Cluster Control Center fails to start.

MD Cluster Worker

Supported Operating Systems

OS	Version
Windows	Windows 11 23H2+, Windows Server 2019+.
Ubuntu	Ubuntu 22.04+, Ubuntu 24.04+, Debian 12+, Rocky 9+, or RHEL 9+.

Overview

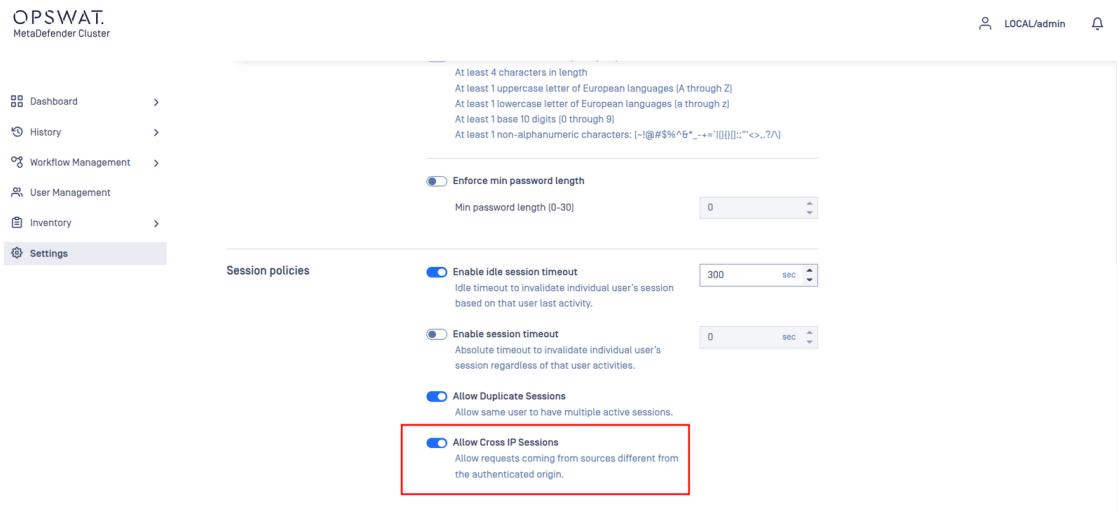
This section describes how to install a **MD Cluster Worker** on a target machine by retrieving and executing an installation script from **MD Cluster Control Center**. After installation, administrators can monitor the health status of the installed worker.

Prerequisites

Requirement	Description
Privileges	**Administrator** privileges on Windows; **Root** or **sudo** access on Linux
Network	MD Cluster Worker must be able to communicate with MD Cluster Control Center
Port	Default port 8893 must be open for MD Cluster Control Center to reach
Dependencies	Meet System requirements

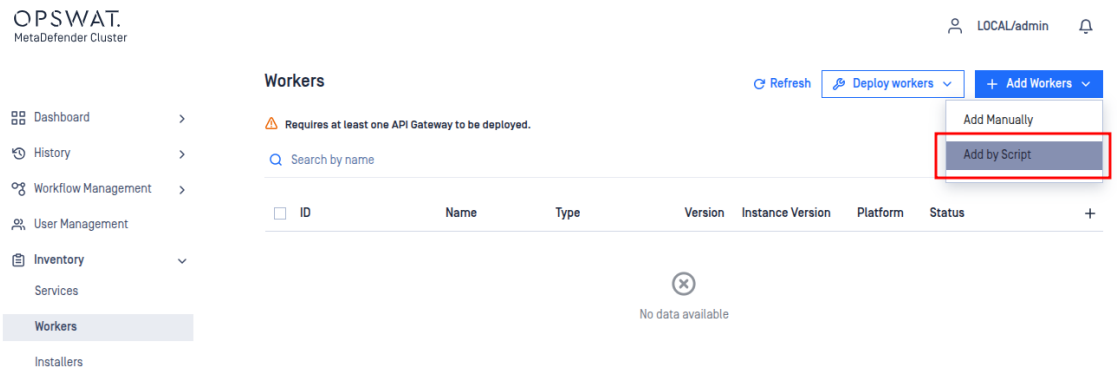
Allow Cross IP Sessions

1. Sign in to MD Cluster **Control Center** console using your administrator account.
2. From the sidebar, go to `Settings` > `Security`.
3. Toggle `Allow Cross IP Sessions`.



Retrieve the installation script

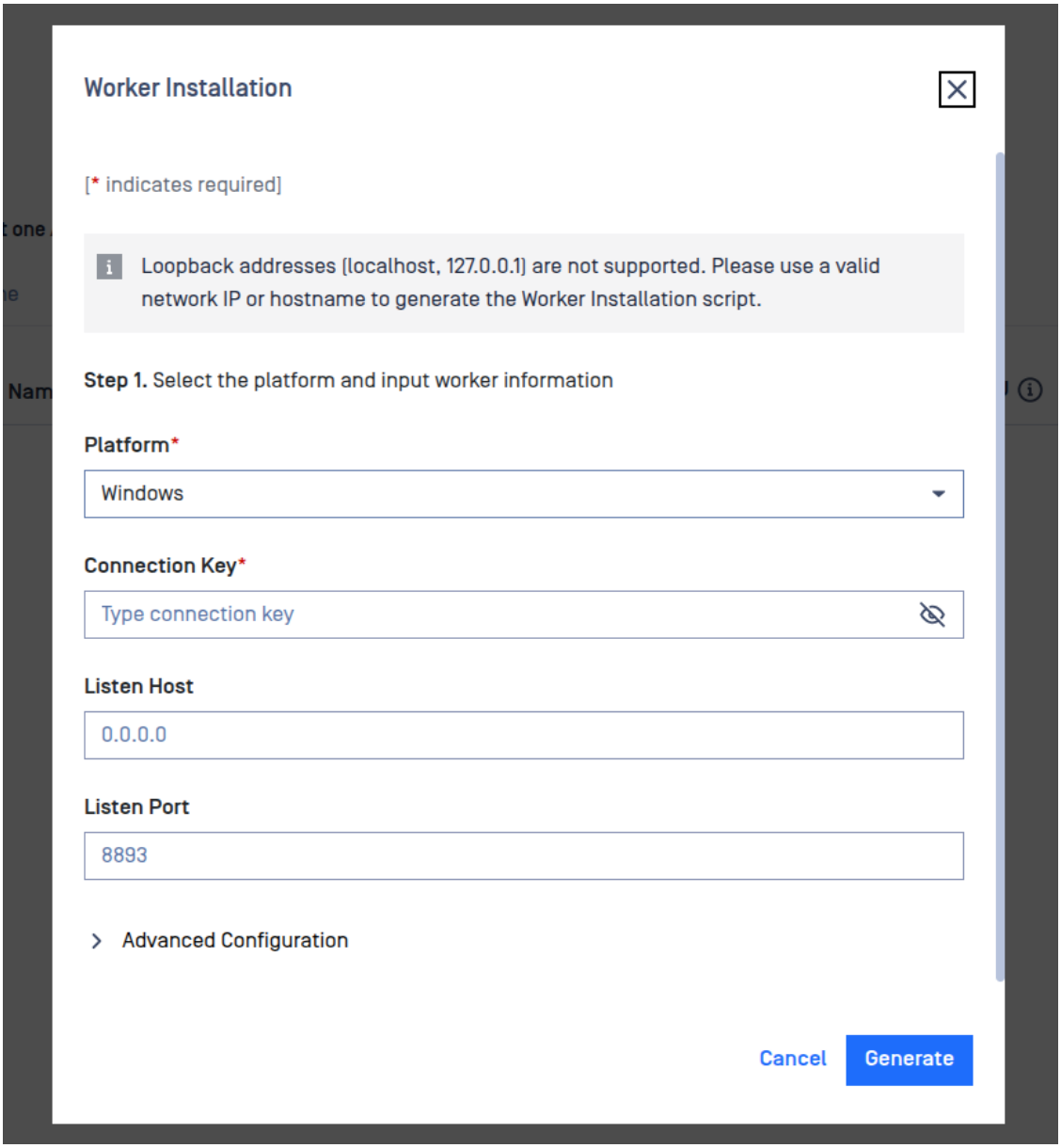
1. From the sidebar, go to to `Inventory` > `Workers`.
2. Expand `Add Workers` and select `Add by script`.



3. In Worker Installation, choose the appropriate platform (Windows or Linux).
4. Enter the connection key provided by your administrator and leave the other settings as default.

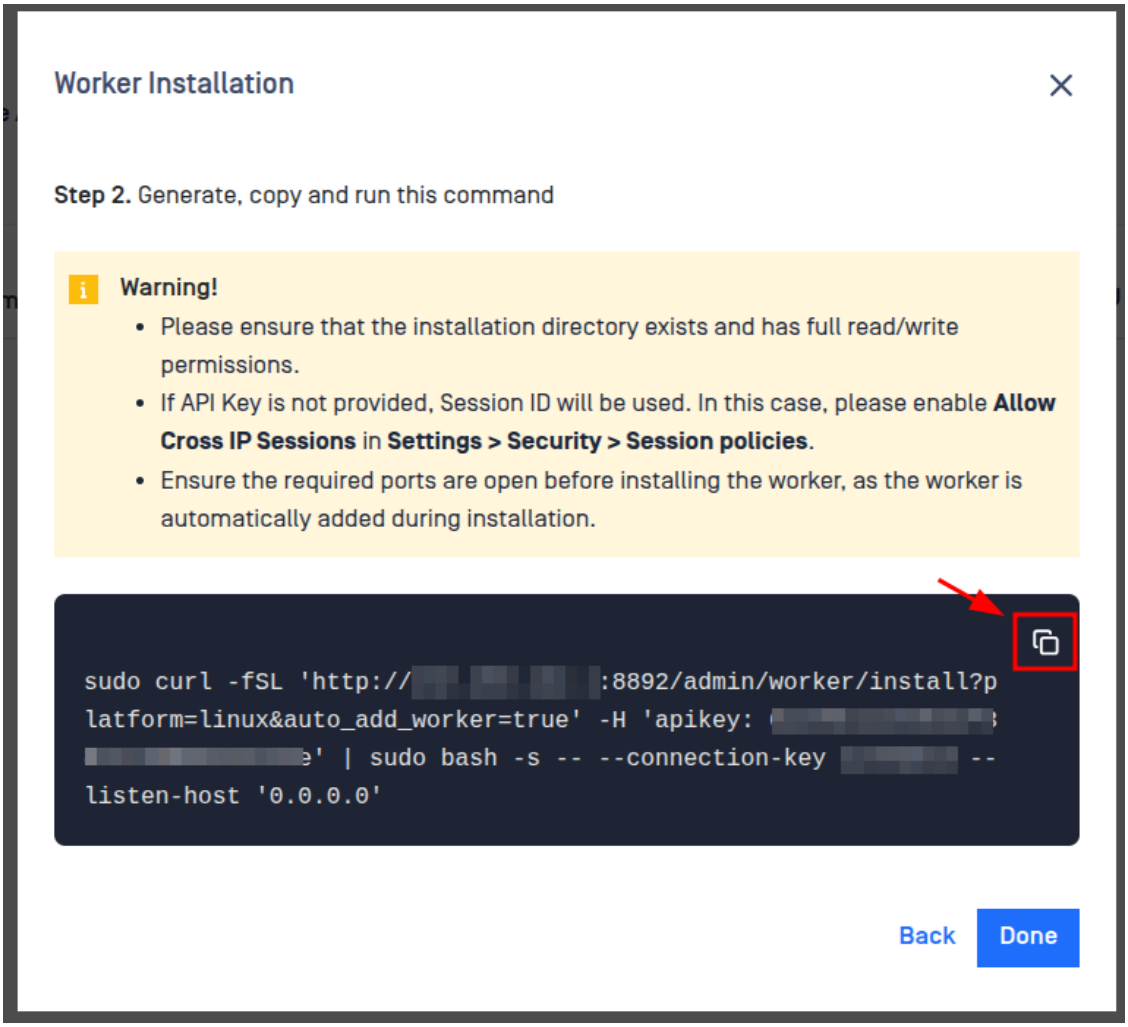
Warning

Avoid using **loopback** addresses (such as localhost, 127.0.0.1, or ::1) when generating the installation script.



5. Click `Generate`.

6. Copy the generated script.



Execute the installation script

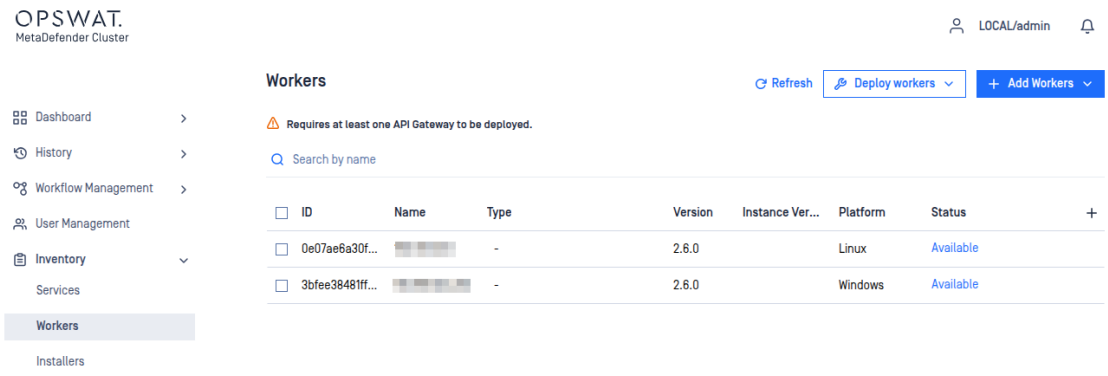
1. Access the target machine.
2. Run the copied command:
   - On Windows: open PowerShell as Administrator, paste and run the script.
   - On Linux: open a root shell (e.g., sudo su), paste and run the script.
3. Once the installation completes, note the Worker ID displayed on the screen.

Code · bash

```
...
[INFO] Worker registered successfully with Control Center. Worker ID:
3bfee38481ff482d8fa1b893afe79d77
```

Verify the installation

1. Sign in to MD Cluster **Control Center** console.
2. From the sidebar, go to `Inventory` > `Workers`.
3. Confirm that the worker with the previously collected Worker ID appears with the correct version and platform.



Service management

- Stop the service:

Code · bash

```
# PowerShell on Windows
> Stop-Service md-cluster-worker

# Terminal on Linux
$ sudo systemctl --machine=md-cluster-worker@.host --user status md-cluster-worker
```

- Start the service:

Code · bash

```
# PowerShell on Windows
> Start-Service md-cluster-worker

# Terminal on Linux
$ sudo systemctl --machine=md-cluster-worker@.host --user start md-cluster-worker
```

- Restart the service:

Code · bash

```
# PowerShell on Windows
> Restart-Service md-cluster-worker
```

```
# Terminal on Linux
$ sudo systemctl --machine=md-cluster-worker@.host --user restart md-cluster-worker
```

Customize the service configuration

During installation, MD Cluster Worker service generates a configuration file at

- Windows: `<installation_dir>\MetaDefender Cluster Worker\md_cluster_worker.yml`
- Linux: `<installation_dir>/data/config/md_cluster_worker.yml`

To customize the service behavior:

1. Open the configuration file in a text editor.
2. Modify the required settings according to your environment.
3. Save the changes.
4. Restart the service to apply the new settings.

Code · bash

```
# PowerShell on Windows
> Restart-Service md-cluster-worker

# Terminal on Linux
$ sudo systemctl --machine=md-cluster-worker@.host --user restart md-cluster-worker
```

Uninstall the service

1. Navigate to MD Cluster Worker installation directory.
2. Open PowerShell (Admin) on Windows or a sudo terminal on Linux.
3. Run command:

Code · bash

```
# PowerShell on Windows
> & '<path_to_worker_installed>\bin\uninstall.ps1'

# Terminal on Linux
$ sudo <path_to_worker_installed>/bin/uninstall.sh
```

Troubleshooting

CURL error 403 during installation

Possible causes:

1. The session has expired.
2. The user does not have deployment permission.

Solutions:

1. Regenerate the installation script and run it again.
2. Use an account with the required permissions or provide an API key with deployment access.

Configuration reference

Common configurations

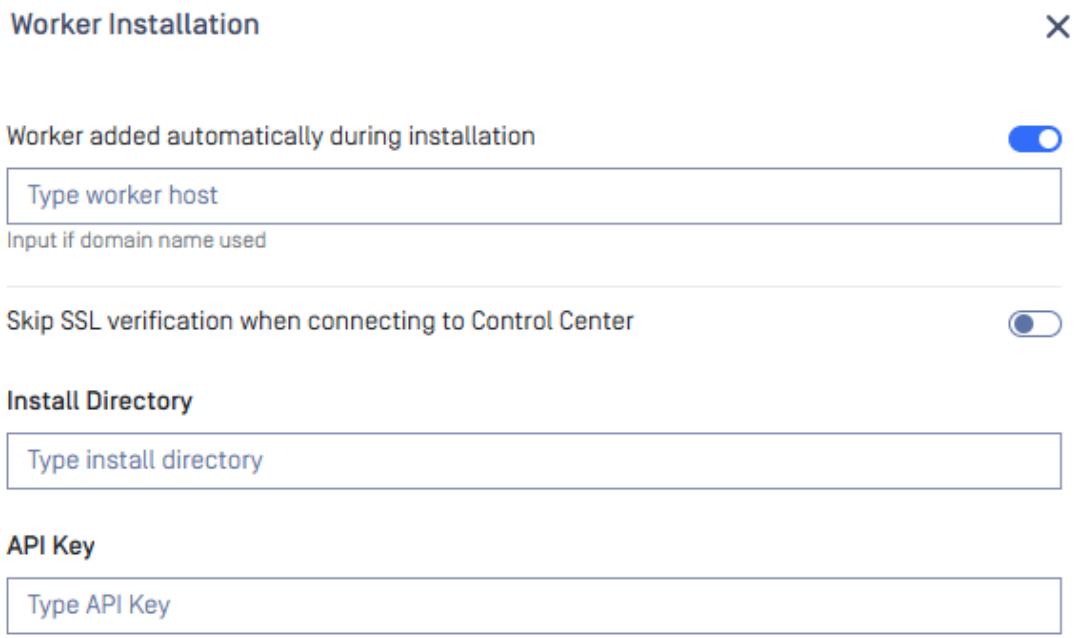
- `Platform` (Required)
  - Value: Windows or Linux
  - Description: The platform where the installation script will be run.
- `Connection Key` (Required)
  - Value: String
  - Description: A **4-64 character** string that contains only digits (`0-9`) and letters (`a-z`, `A-Z`) provided by system administrator.
- `Listen Host` (Optional)
  - Value: String
  - Description: The host/interface the MD Cluster Worker listens on. Default is 0.0.0.0 (all IPv4).
- `Listen Port` (Optional)
  - Value: Number
  - Description: The port number where MD Cluster Worker listens for requests. Default is 8893.



Advanced configurations

- `Worker added automatically during installation` (Optional)
  - Value: String
  - Description: The worker host is automatically detected from the IP address used to connect to MD Cluster Control Center. To register the worker with a DNS or domain name instead, specify the hostname here.
- `Skip SSL verification when connecting to Control Center` (Optional)
  - Value: Yes or No
  - Description: Use this option only for testing or when using self-signed certificates. For production, use a CA-signed certificate and keep SSL verification enabled.
- `Install Directory` (Optional)
  - Value: a valid directory path.
  - Description: Specifies the location where MD Cluster Worker will be installed. The default is `C:\Program Files\OPSWAT\MetaDefender Cluster Worker` on Windows and `/opt/md-cluster-worker/` on Linux.
- `API Key` (Optional)
  - Value: String
  - Description: Use an **API Key instead of** `Allow Cross IP Sessions` for authentication during installation. This is the recommended alternative when you don't want to enable Allow Cross IP Sessions.
- `Skip dependency installation` (Optional, Linux only] :
  - Value: Yes or No

- Description: Skip installing dependencies on Linux (e.g. when they are pre-installed or managed by your OS image).



Lake and Warehouse setup

Overview

This section describes how to set up the Data Lake and Data Warehouse in PostgreSQL. Once configured, execution services (such as MD Core, MD Cluster API Gateway and MD Cluster Callback Service) can store and retrieve scan results from them, and MD Cluster Control Center can access them to generate executive reports.

Prerequisites

1. **PostgreSQL Service**: Must be installed and running.
2. **MD Cluster Control Center**: Must be installed.
3. **Superuser Rights**: PostgreSQL user must have superuser privileges.

Assumption

It is assumed that PostgreSQL Server A is designated to host the Data Lake, while PostgreSQL Server B hosts the Data Warehouse. The Data Lake must be configured first, as the Data Warehouse will connect to it as its primary data source.

Windows

1. On the machine hosting MD Cluster **Control Center**, navigate to the folder:

Code · powershell

```
C:\Program Files\OPSWAT\MetaDefender Cluster Control Center
```

2. Run the following command to set up **Data Lake** on PostgreSQL Server A:

Code · powershell

```
md-cluster-dbready.exe --host=<postgres-host> --port=<postgres-port> --user=<postgres-user> --
password=<postgres-password> --target=lake
```

3. Run the following command to set up **Data Warehouse** on PostgreSQL Server B:

Code · powershell

```
md-cluster-dbready.exe --host=<postgres-host> --port=<postgres-port> --user=<postgres-user> --
password=<postgres-password> --lake-host=<lake-postgres-host> --lake-port=<lake-postgres-port>
--lake-user=<lake-postgres-user> --lake-password=<lake-postgres-password> --target=warehouse
```

Linux

1. On the machine hosting MD Cluster Control Center, navigate to the folder:

Code · bash

```
/usr/sbin
```

2. Run the following command to set up **Data Lake** on PostgreSQL Server A:

Code · bash

```
md-cluster-dbready --host=<postgres-host> --port=<postgres-port> --user=<postgres-user> --
password=<postgres-password> --target=lake
```

3. Run the following command to set up **Data Warehouse** on PostgreSQL Server B:

Code · bash

```
md-cluster-dbready --host=<postgres-host> --port=<postgres-port> --user=<postgres-user> --
```

```
password=<postgres-password> --lake-host=<lake-postgres-host> --lake-port=<lake-postgres-port>
--lake-user=<lake-postgres-user> --lake-password=<lake-postgres-password> --target=warehouse
```

Combined Lake and Warehouse

If the Data Lake and Data Warehouse are hosted on the same PostgreSQL instance, a combined setup command can be used:

Code · bash

```
md-cluster-dbready --host=<postgres-host> --port=<postgres-port> --user=<postgres-user> --
password=<postgres-password> --target=lake,warehouse
```

Warning

While this approach is simpler and faster to deploy, it is not recommended for large-scale or long-running systems due to potential performance and scalability limitations.

License activation

MetaDefender Cluster supports two types of license activations:

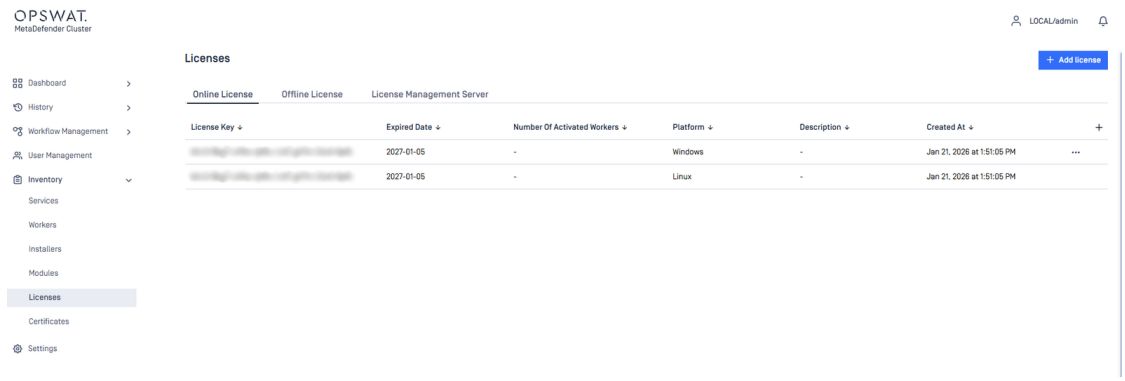
- Online Activation
- Offline Activation
- License Management Server Activation

Online Activation

MetaDefender Cluster (MD Cluster) supports seamless license activation for every deployed **MetaDefender Core** instance. The license key must be provided to MD Cluster **Control Center** and will be manually activated on each individual **MetaDefender Core** instance. If necessary, multiple license keys may also be supplied.

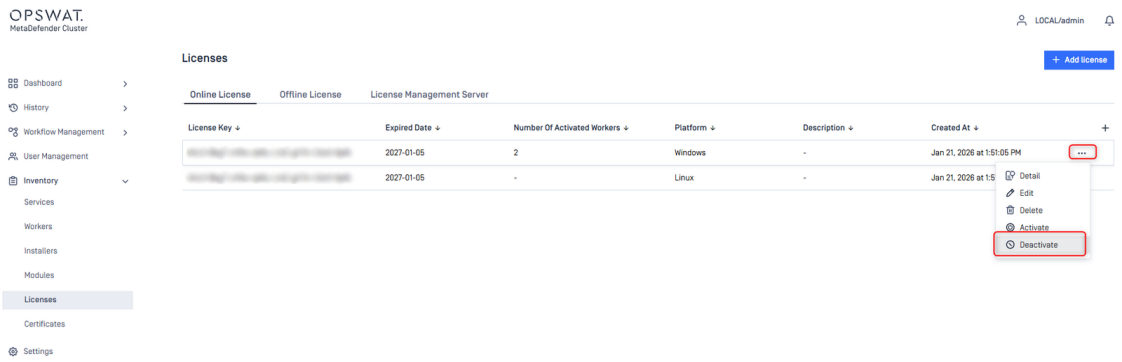
Adding License

1. Sign in to MD Cluster **Control Center** console.
2. Go to `Inventory` > `Licenses` and select `Add license`.
3. Input your license key and click `Add`.



License Activation

1. Sign in to MD Cluster **Control Center** console.
2. From the left side bar, go to `Inventory` > `Licenses`.
3. From the list of available licenses, choose the key you wish to use for activation.
4. Click `Activate` to apply the license key to the appropriate instance(s).



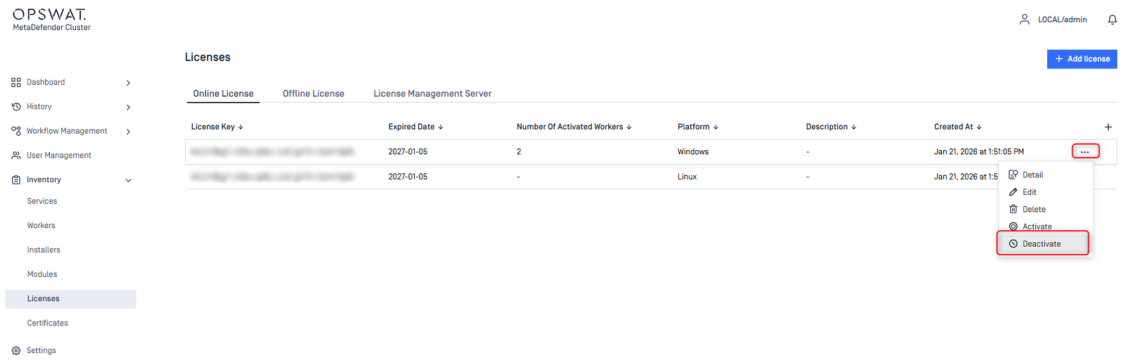
> **Info**
>
> Once `Activate` is clicked, the license key will be applied to all unlicensed **MetaDefender Core** instances. Ensure that your license quota is sufficient to cover all unlicensed instances.
>
> You can view the number of activated instances and available slots by selecting **Details** on the license key. At the moment, **Details** can only be viewed after activation is successful.

License Deactivation

Follow these steps to deactivate your license:

1. Sign in to MD Cluster **Control Center** console.
2. From the left side bar, select `Inventory` > `Licenses`.
3. From the list of available licenses, choose the key you wish to use for deactivation.
4. Click Deactivate to remove the license from all **MetaDefender Core** instances currently activated with the license

key.



Info

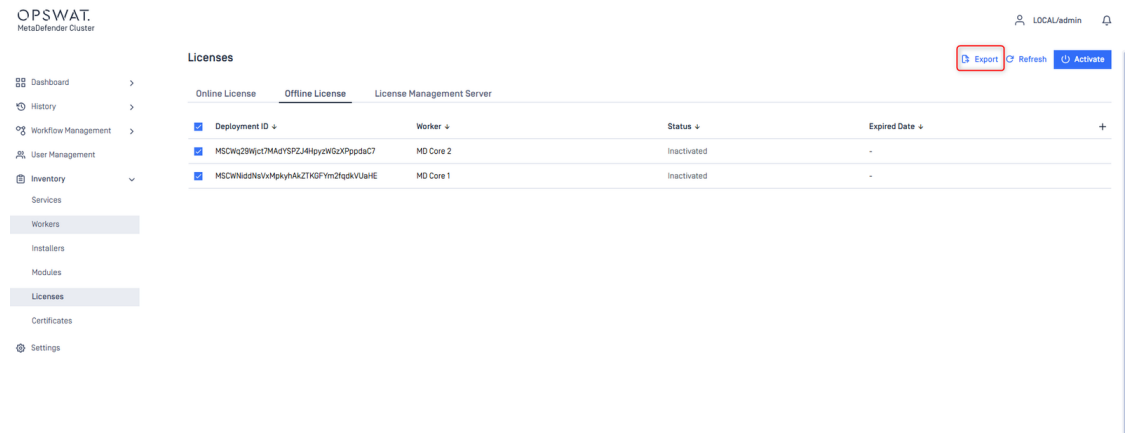
Once deactivated, the license slots will become available and can be reassigned when necessary.

When **MetaDefender Core** instances are undeployed through MD Cluster **Control Center**, their associated licenses are automatically deactivated.

Offline Activation

Collect Deployment IDs

1. Sign in to MetaDefender Cluster (MD Cluster) **Control Center** console.
2. Go to `Inventory` > `Licenses` and select `Offline License` tab.
3. Select Deployment IDs of MetaDefender Core instances you prefer to activate.
4. Press `Export` at the top right corner and save the exported file to your location of choice.



Info

MetaDefender Cluster **Control Center** only displays the Deployment IDs of MetaDefender Core instances that have not been activated thus far.

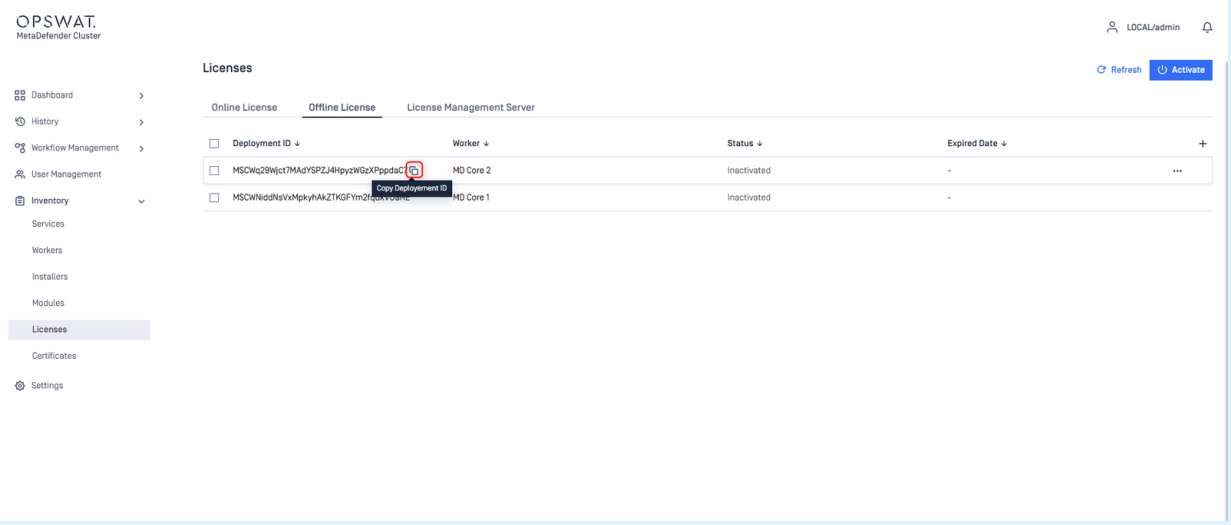
Info

The exported file includes a list of chosen Deployment IDs that will be used for activation in the subsequent stage.

Info

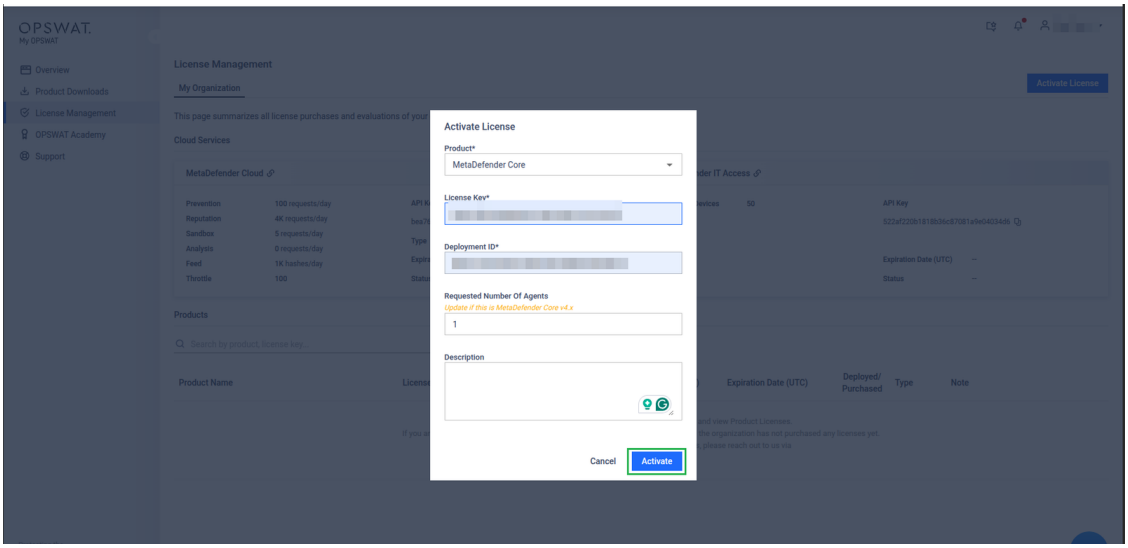
To collect the Deployment ID of a **single** MetaDefender Core instance, please:

1. Hover your mouse over the preferred Deployment ID to display the copy button.
2. Press the copy button.
3. Retain the copied Deployment ID and proceed to the next stage.



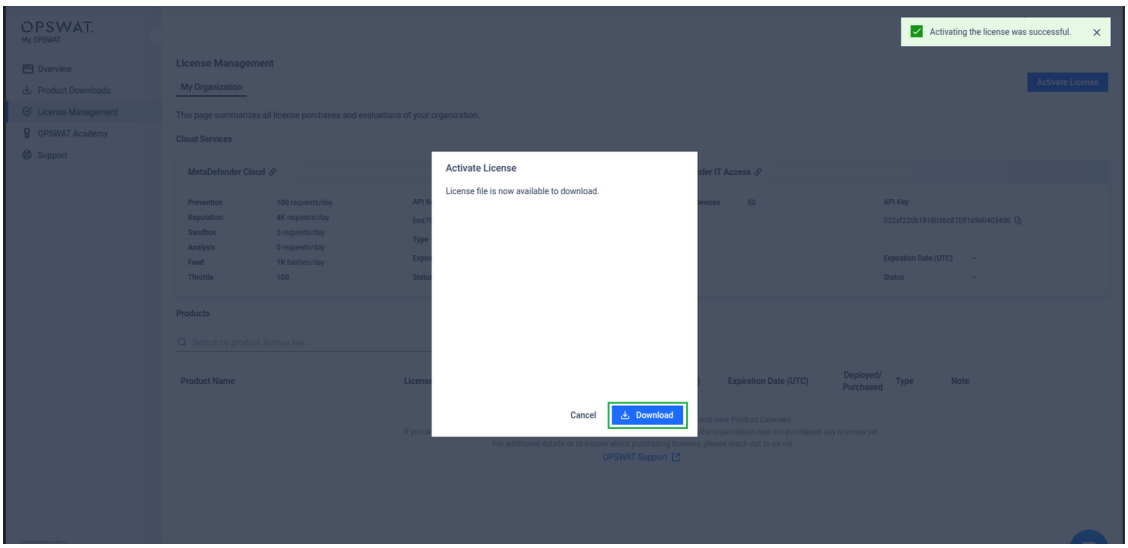
Activate license with Deployment ID

1. Sign in to MyOPSWAT with your account.
2. Navigate to `License Management` on the left side panel.
3. Click `Activate License`.
4. Fill out all necessary information, including your Activation Key, Deployment ID and selection of the Package you require.



5. Click `Activate`.

6. Click `Download` and store the license file to your secure location.



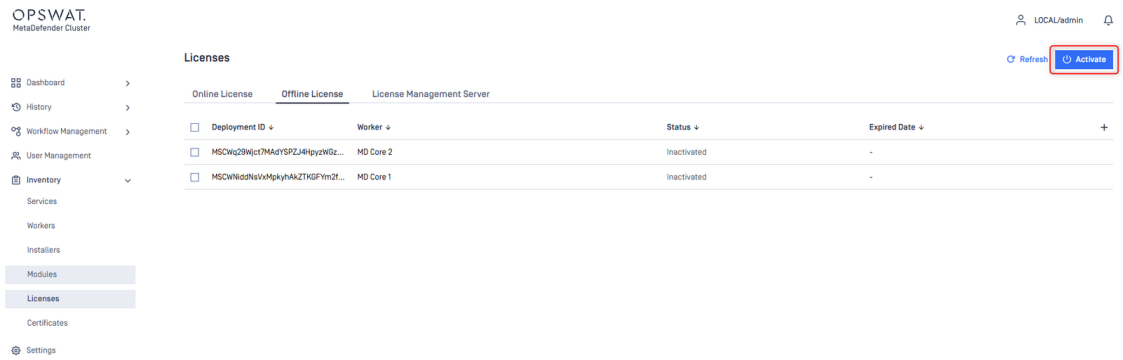
> **Info**
>
> The license file is associated with one unique Deployment ID. The users must carry out steps 3 to 6 for every deployment ID on their list.

Activate MetaDefender Core instances with license files

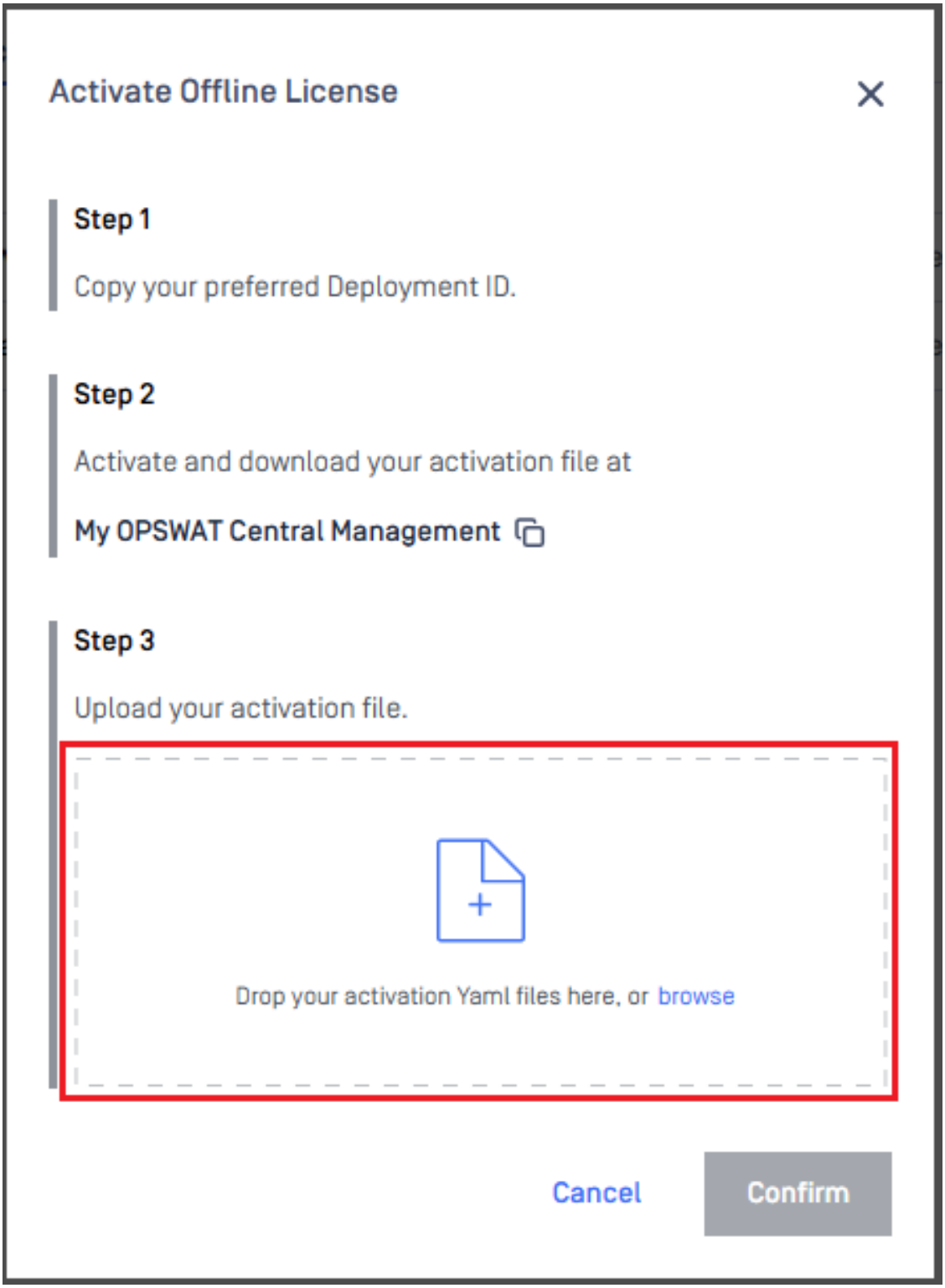
1. Sign in to MD Cluster **Control Center** console.

2. Go to `Inventory` > `Licenses` and select `Offline License` tab.

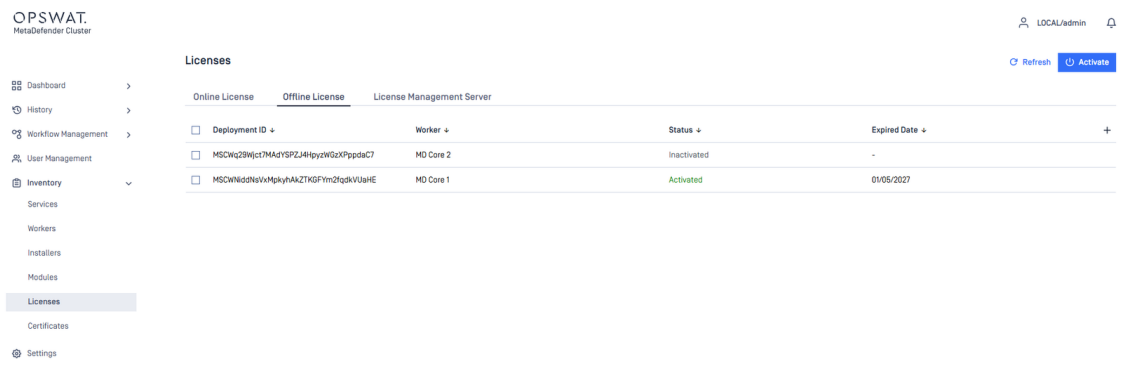
3. Click `Activate`.



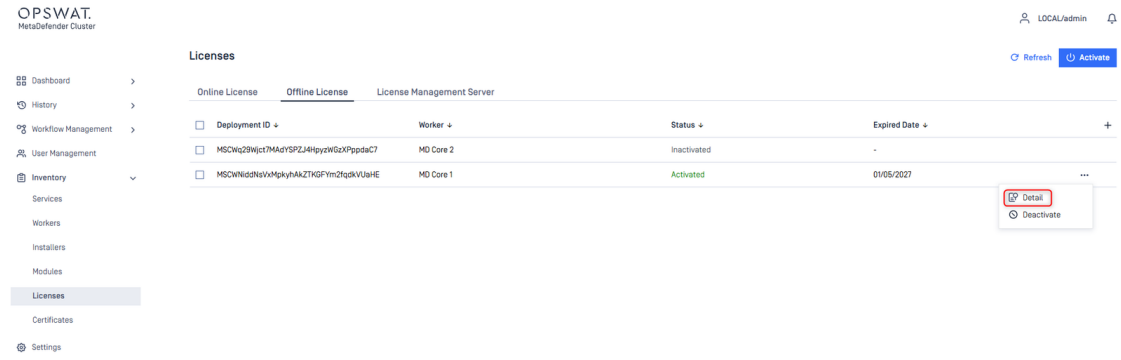
4. Drop the license files into the dash area for submission.



5. Click `Confirm` to complete.
6. MD Cluster **Control Center** activates MetaDefender Core instances associated with the provided license files and displays their activation status.



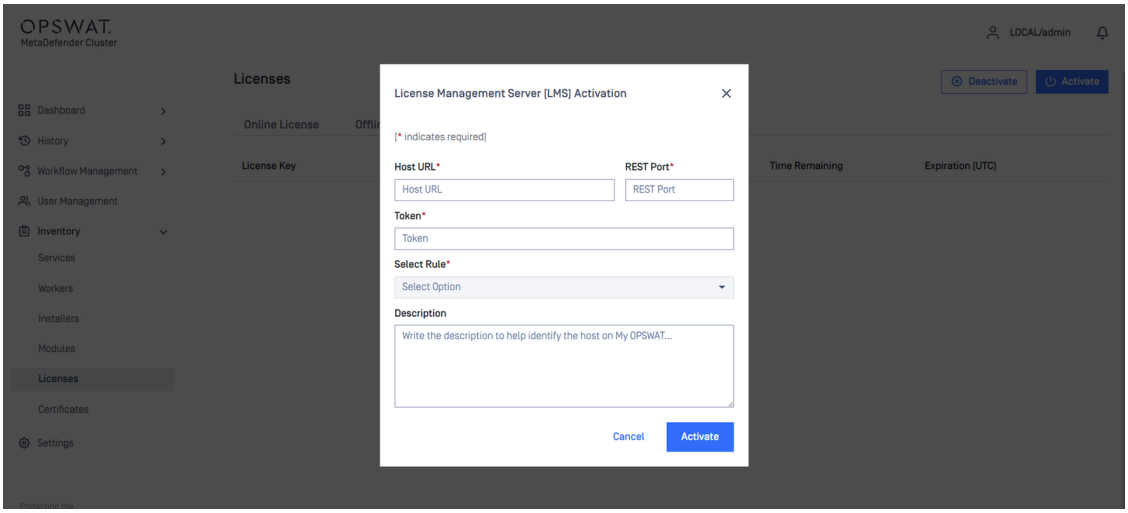
7. Select an activated MetaDefender Core instance and press `Details` to view the license details



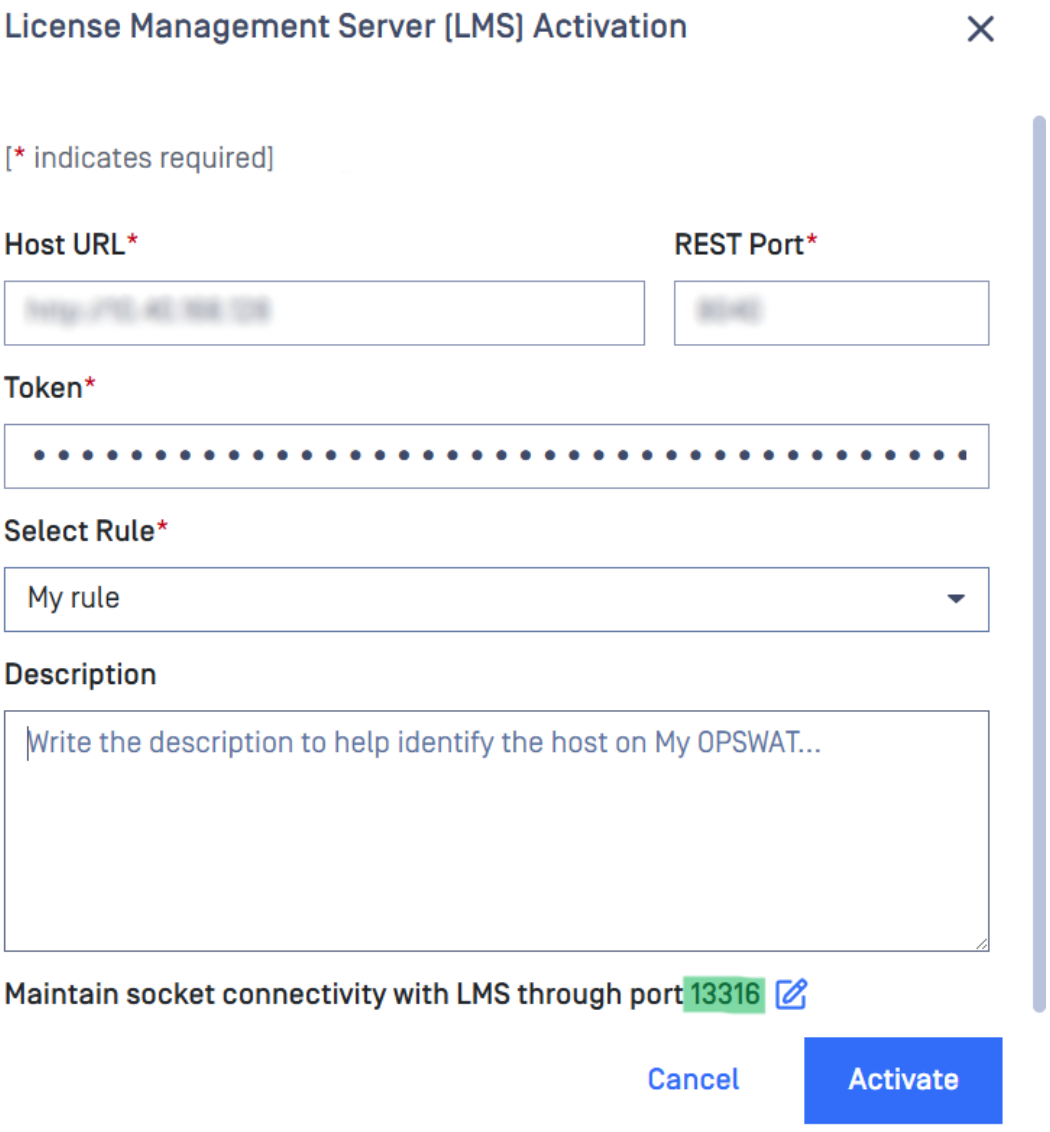
License Management Server Activation

Connect to License Management Server (LMS)

1. Sign in to MetaDefender Cluster **Control Center** console.
2. Go to `Inventory` > `Licenses` and select License Management Server tab.
3. Select `Activate` and provide the necessary information in the required fields:
    1. **Host URL**: The URL of the License Management Server to connect to.
    2. **REST Port**: The port of the License Management Server.
    3. **Token**: Access token obtained from the License Management Server.



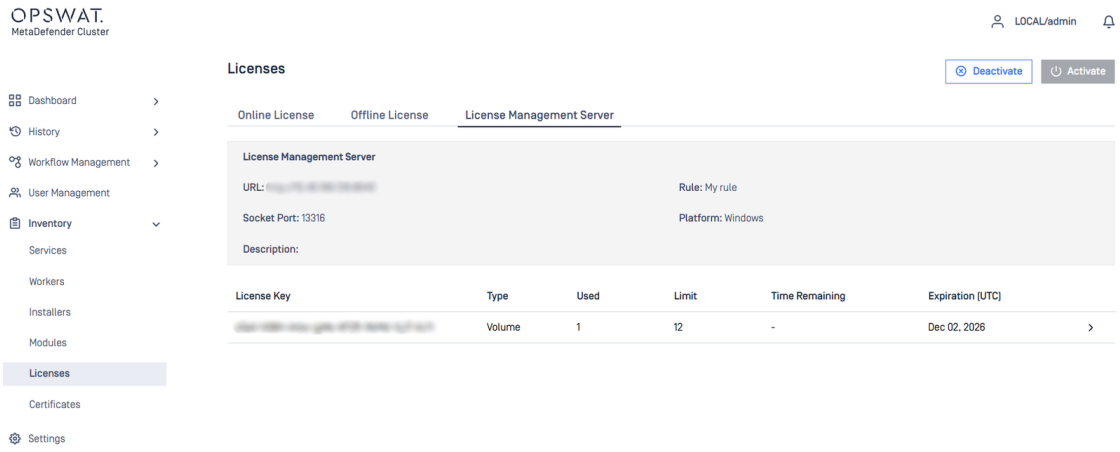
4. After input all required fields, the connection to LMS will be established and available rules can be selected under `Select Rules`.



Info

The port number defined in "Maintain socket connectivity with LMS through port `<port_number>`" is required to sustain connectivity between MD Cluster Control Center and License Management Server. In order for successfully activation, **confirm that the port is properly configured and allowed in firewall setting**.

5. Select the appropriate rule and choose Activate. Upon successful completion, the license details will be shown.



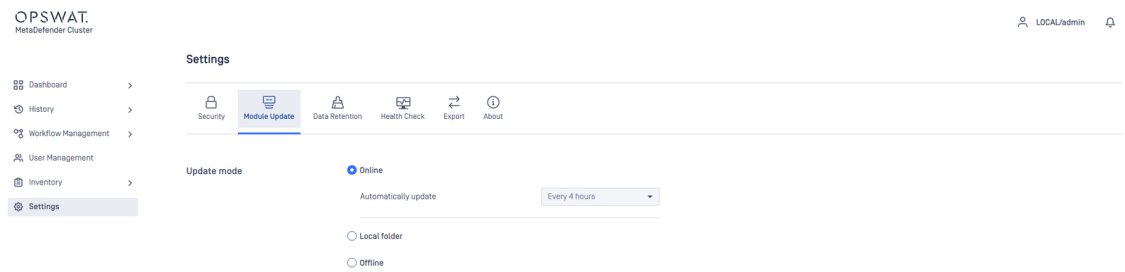
Info

Once activation is successful, the License Management Server will manage the license status of all MetaDefender Core instances. Please ensure sufficient quota is available.

6. Check instance status under `Inventory` > `Workers` and confirm all MetaDefender Core instances is activated successfully.

Module update

MetaDefender Cluster (MD Cluster) introduces three modes of Module Update. To switch between the modes, please sign in to MD Cluster **Control Center** console, navigate to `Settings` and select `Module Update` tab.

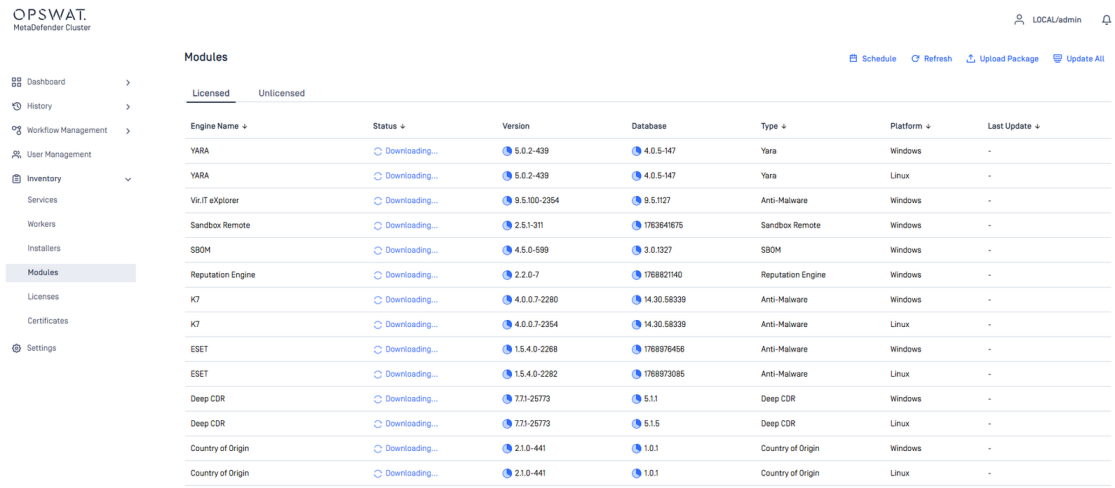


Info

Online update mode is enabled by default.

Online module update

In online update mode, MD Cluster **Control Center** will base its checks on the activated licenses to find and download the latest engine packages from OPSWAT online update infrastructure, repeating this process every four hours.



All downloaded engine packages are verified and stored in MD Cluster **File Storage** for licensed MetaDefender Core instances to pull, install, or upgrade on their end. Through this mechanism, the instances cease to independently pull the engine packages from the update infrastructure, conserving network bandwidth while enhancing their readiness.

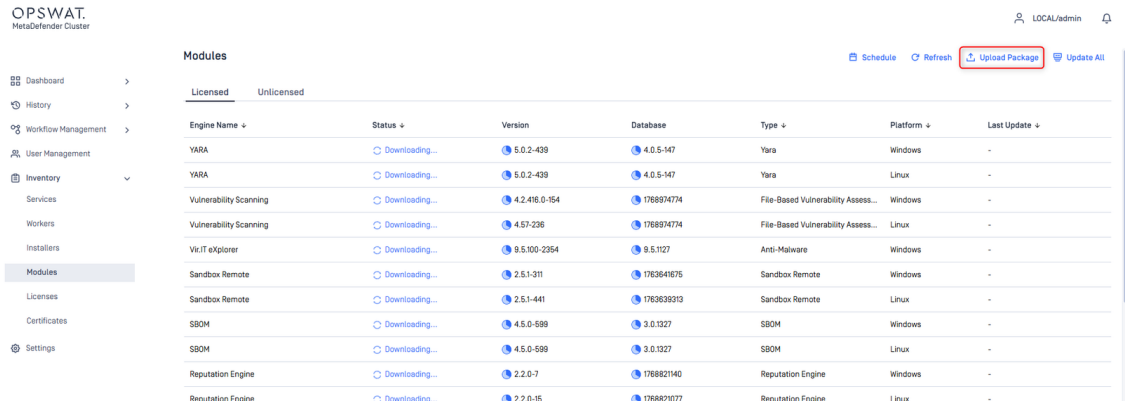
Offline module update

In offline update mode, administrators must download the licensed engine packages from **MetaDefender Update Downloader** and upload them manually to MD Cluster **Control Center**.

Info

Please reference here for more details about downloading engine packages from MetaDefender Update Downloader.

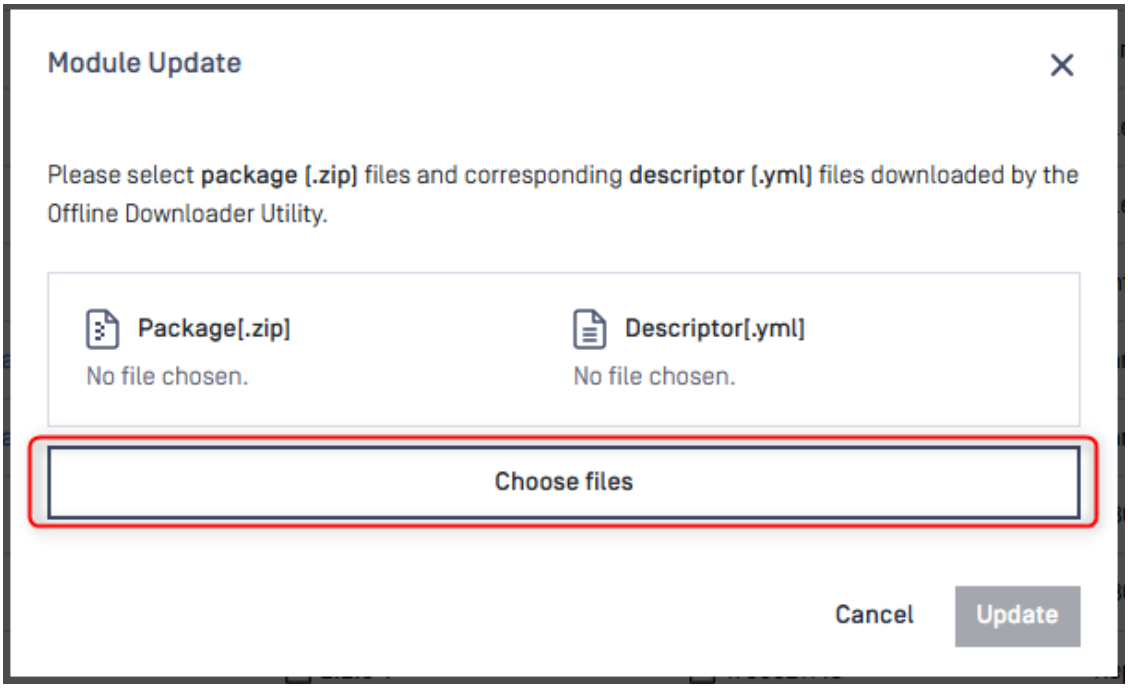
1. Sign in to MD Cluster **Control Center** console.
2. Go to `Inventory` > `Modules`.
3. Press `Upload Package` at the top right corner.



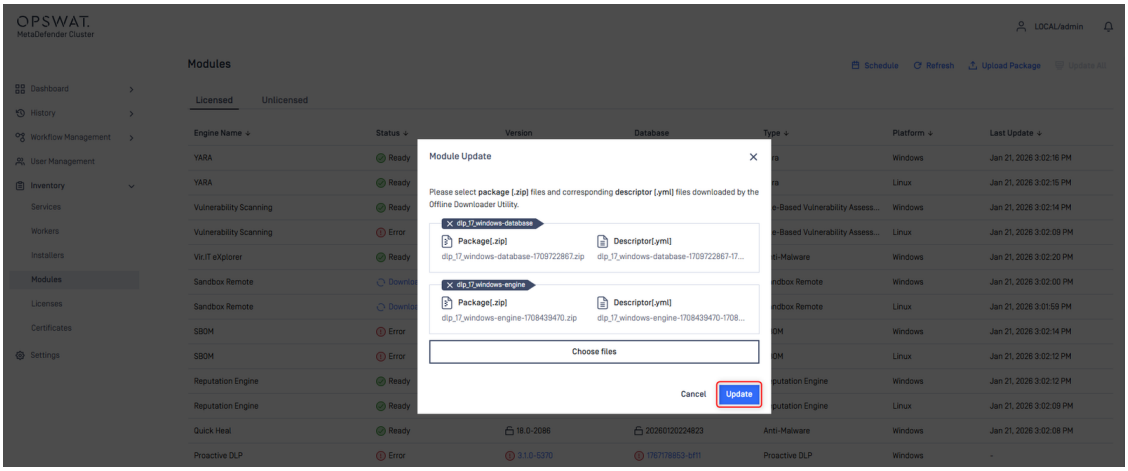
Info

`Update All` is always disabled if Offline Update mode is selected in `Settings` > `Module Update`.

4. Choose your engine package files.

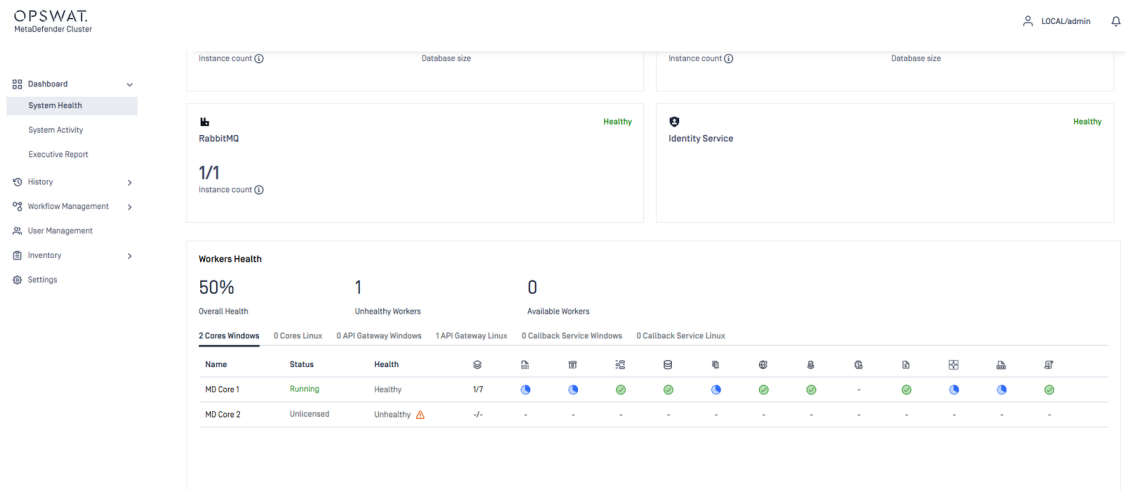


5. Click `Update` to submit the package files.



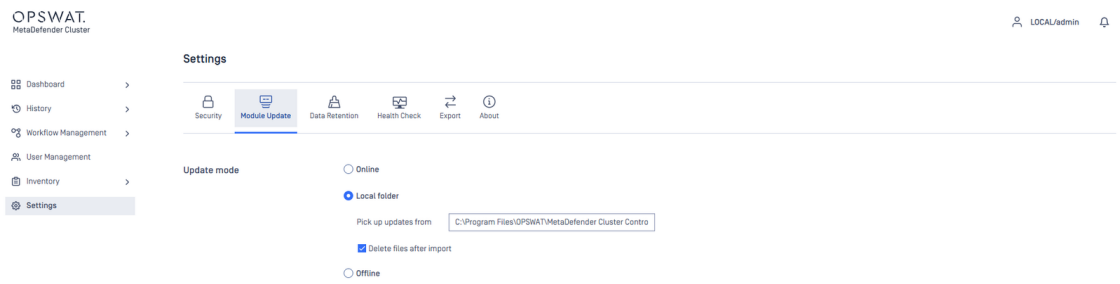
> **Info**
>
> Package files from various engines can be selected simultaneously.

6. Wait until engine packages are ready.
7. Engine update statuses on MetaDefender Core instances can be monitored in `Dashboard` > `System Health` > `Worker Health`.



Local folder update

In local folder update, the administrator can specify the path to a folder in `Pick updates from` option. MD Cluster **Control Center** autonomously gathers packages that are added to the folder, then verifies the packages and stores them in MD Cluster **File Storage** for licensed MetaDefender Core instances to retrieve, install, or update on their end.



The administrator has the option to select `Delete files after import` so that MD Cluster **Control Center** wipes all packages upon success.

Info

Network-mapped drives and UNC paths are permitted.

- On Windows, ensure that MD Cluster Control Center has permission to access to the folder.
- On Linux, ensure that the folder has read and execute permissions set (e.g., `chmod 755`).

MetaDefender Cluster Compatibility Matrix

Benefits of MetaDefender Cluster

By integrating with MetaDefender Cluster (MD Cluster), administrators can take advantage of simplified scalability, high availability, distributed extraction, and improved performance optimization — all through a single, unified MD Cluster API Gateway endpoint.

Version Compatibility Matrix

The table below shows the minimum supported product versions compatible with MD Cluster version 2.6.0, running MetaDefender Core (MD Core) version 5.18.1.

Product	Minimum Supported Version
MetaDefender Storage Security	4.4.0
MetaDefender Kiosk	4.8.1
MetaDefender Email Gateway Security	6.3.2
MetaDefender ICAP Server	5.13.0
MetaDefender Managed File Transfer	3.11.3

Product	Minimum Supported Version
MetaDefender Software Supply Chain	3.3.1
MetaDefender Endpoint	7.6.2605.999

Integration through MD Cluster API Gateway

To ensure seamless integration with MD Cluster, all products must communicate directly with **MD Cluster API Gateway** instead of connecting to individual MD Core instances.

All file scanning and result retrieval requests should be routed through MD Cluster API Gateway

1. File analysis
2. Batch analysis
3. Fetching scan results

MetaDefender Cluster Documentation

The users can consult this web page or, alternatively, they can download the manual in pdf format from the link below:

MetaDefender Cluster manual

- Download link
- SHA256:

High Availability

Overview

In MetaDefender Cluster (MD Cluster), critical components for its continuous operation include RabbitMQ, Redis, Postgres, and MD Cluster File Storage. Any disruption of these components will lead to an interruption in the scanning processes and result in a failed verdict for the processed files. To prevent the interruption, high-availability solutions must be implemented on the components.

A strategy for achieving high availability is the replication and redundancy of essential components. The key concept is that if a single component fails, the redundant system takes over seamlessly, avoiding any interruption in service. Following are guidelines to set up the high availability solution on individual components and apply them in MetaDefender Cluster.

- High availability support for MD Cluster File Storage.
- High availability support for RabbitMQ.
- High availability support for Redis.
- High availability support for Data lake.

High Availability support for File Storage

Key concept

The High Availability solution for MetaDefender Cluster (MD Cluster) **File Storage** is implemented in this manner:

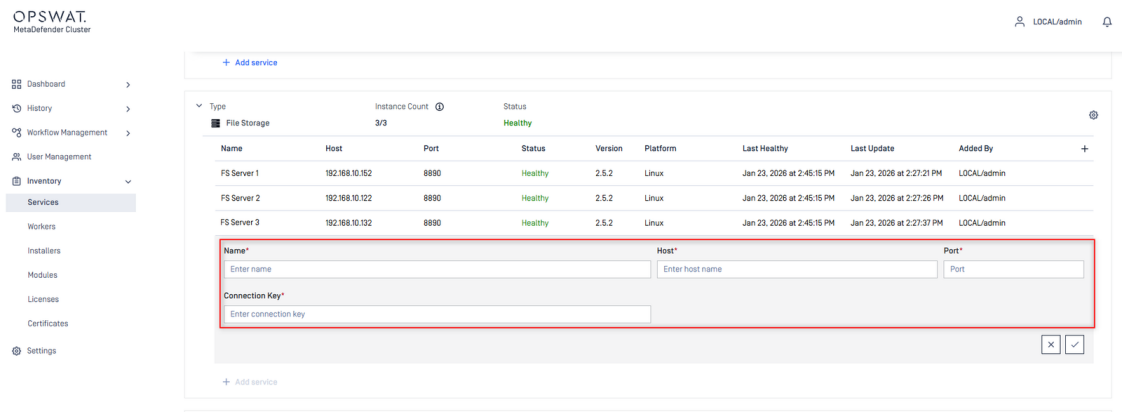
- A file is stored across multiple MD Cluster **File Storages** by MD Cluster **API Gateway** or **MetaDefender Core**.
- MD Cluster **API Gateway** or **MetaDefender Core** must request all MD Cluster **File Storage** instances for a file existence or a file download.

> **Info**
>
> A minimum of **three** MD Cluster File Storage instances must be installed on separate hosts for High Availability solution to function properly.
>
> All MD Cluster File Storage instances must be of an identical version.

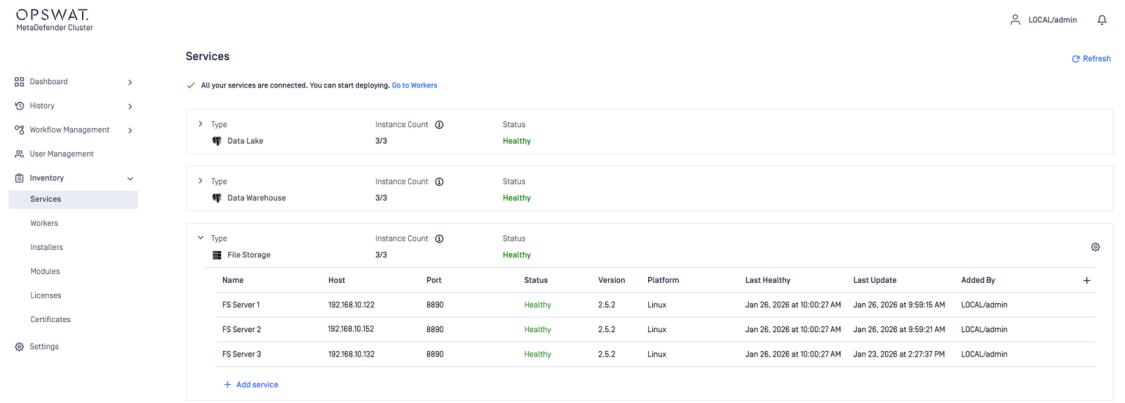
Setup Instructions

1. Setup MD Cluster **File Storage** instances on individual servers.
2. Sign to MD Cluster **Control Center** console with your Administrator account.
3. Navigate to `Inventory` > `Services`.
4. Expand the `File Storage Service` group.
5. Click `Add service`.

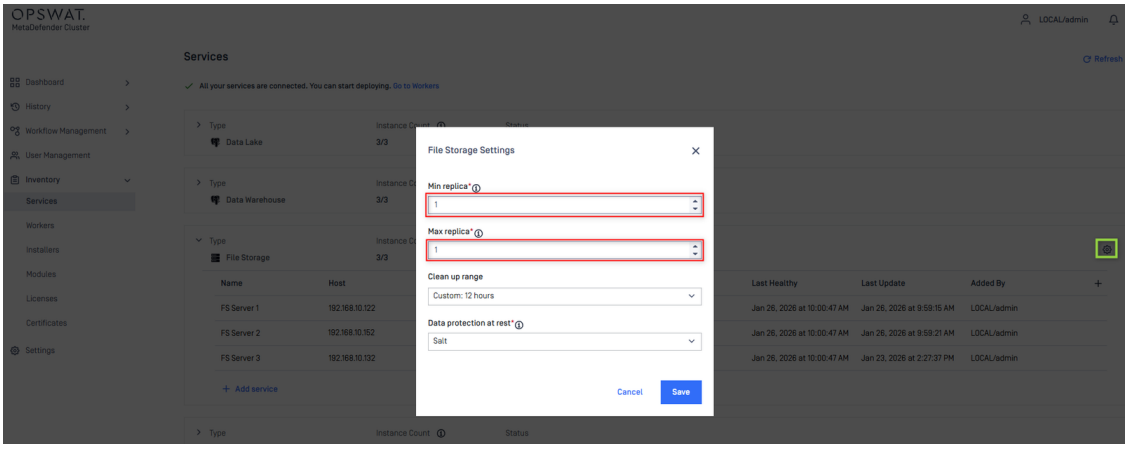
6. Enter the values for `Name`, `Host`, `Port` and `Connection Key` fields of individual MD Cluster File Storage instances set up in Step 1.
7. Click the `Check` icon in the bottom right to complete.



8. Ensure all MD Cluster `File Storage` instances are healthy and reachable by MD Cluster `Control Center`.



9. Click on the gear icon in the top left of `File Storage Service` group to configure the minimum and maximum replicas.



- Minimum replica: The minimum number of data copies that must be written for the operation to succeed.
- Maximum replica: The maximum of data copies stored across the system.

> **Info**
>
> To balance performance and high availability efficiency, the minimum and maximum replicas should be set to the following values:
>
> - `Min replica` = 2
> - `Max replica` = 3

10. Click `Save` to complete.

High Availability support for RabbitMQ

> **Info**
>
> A minimum of **three** RabbitMQ nodes must be installed on separate hosts for High Availability solution to function properly.
>
> An **odd number** of RabbitMQ nodes is required.
>
> All RabbitMQ nodes must be of an identical version.

RabbitMQ cluster

1. Install RabbitMQ nodes on servers.
2. Ensure each node can resolve its own hostname and those of the others.

- Start **Command Prompt on Windows** or **Terminal on Linux**, and run the following command to get hostname.

Code · bash

```
# Windows
> hostname

# Debian/Ubuntu or Red Hat/Rocky
$ hostname
```

- In **Command Prompt on Windows** or **Terminal on Linux** of any RabbitMQ node, and run the following command to ping to the other using its hostname.

Code · bash

```
# Windows
> ping <other_node_hostname>
```

```
# Debian/Ubuntu or Red Hat/Rocky
$ ping <other_node_hostname>
```

3. On each RabbitMQ nodes, open the following ports.

Default port	Process
5672	Used by MD Cluster **Control Center**, MD Cluster **API Gateway** and **MetaDefender Core**.
4369	Used by discovery daemon on each RabbitMQ nodes and `rabbitmqctl` tool.
25672	Used by each RabbitMQ nodes and `rabbitmqctl` tool to communicate to the other nodes.
15672	Used by `rabbitmq-management` plugin.

4. Verify that the Erlang cookies of all RabbitMQ nodes are identical.

Info

RabbitMQ nodes and `rabbitmqctl` tool use a cookie to determine whether they are allowed to communicate with each other. For two nodes to be able to communicate they must have the same shared secret called the Erlang cookie. The cookie is a string of alphanumeric characters up to 255 characters in size.

- In **Windows**, access the specified locations to check the cookie contents.

Type	Location
Server cookie	`C:\Windows\system32\config\systemprofile.erlang.cookie`
Command line cookie	`C:\Users%USERNAME%.erlang.cookie`

- In **Linux**, access the specified locations to check the cookie contents.

Type	Location
Server cookie	`/var/lib/rabbitmq/.erlang.cookie`

Type	Location
Command line cookie	`$HOME/.erlang.cookie`

5. Select one node to be the leader of RabbitMQ cluster.
6. In **Command Prompt on Windows** or **Terminal on Linux** of the server hosting the leader, run the following command to obtain its node name.

Code · bash

```
# Windows
> rabbitmqctl status

# Debian/Ubuntu or Red Hat/Rocky
$ rabbitmqctl status
```

7. In **Command Prompt on Windows** or **Terminal on Linux** of each member node server, run the following command to join the node to the same cluster as the leader.

Code · bash

```
# Windows
> rabbitmqctl stop_app
> rabbitmqctl reset
> rabbitmqctl join_cluster <leader_node_name>
> rabbitmqctl start_app

# Debian/Ubuntu or Red Hat/Rocky
$ rabbitmqctl stop_app
$ rabbitmqctl reset
$ rabbitmqctl join_cluster <leader_node_name>
$ rabbitmqctl start_app
```

8. In **Command Prompt on Windows** or **Terminal on Linux** of all nodes, ensure they are in the same cluster.

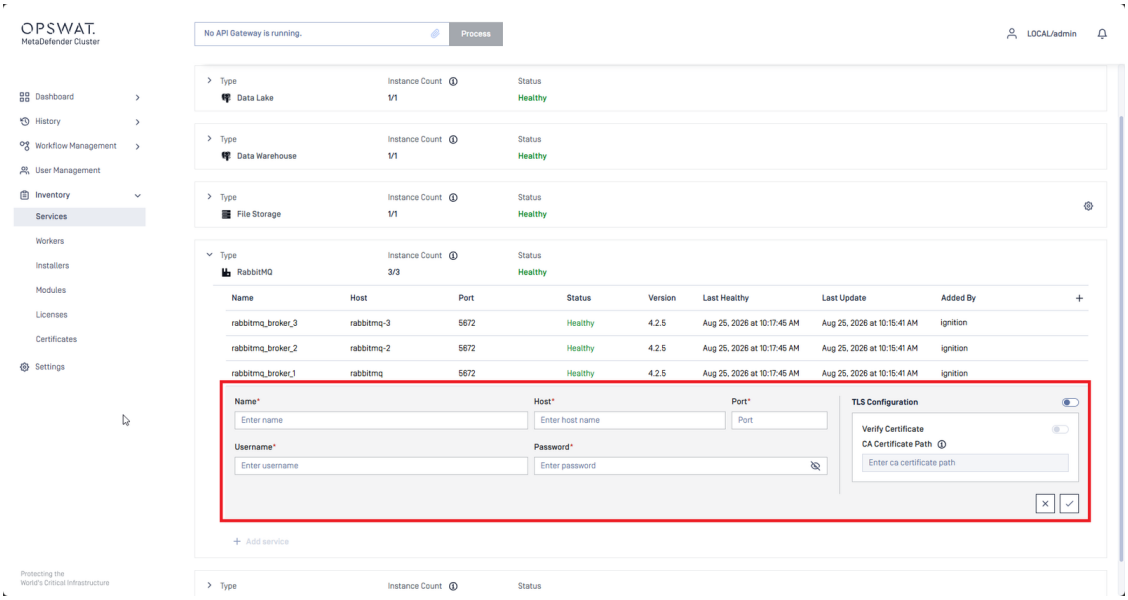
Code · bash

```
# Windows
> rabbitmqctl cluster_status

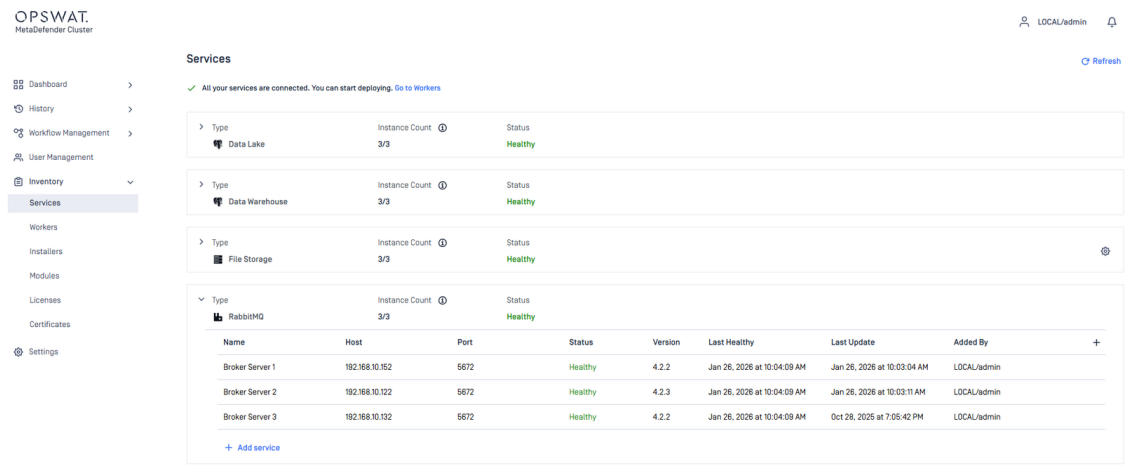
# Debian/Ubuntu or Red Hat/Rocky
$ rabbitmqctl cluster_status
```

Setup Instructions

1. Sign to MD Cluster **Control Center** console with your Administrator account.
2. Navigate to `Inventory` > `Services`.
3. Expand the `RabbitMQ` group.
4. Click `Add service`.
5. Enter the values for `Name`, `Host`, `Port`, `Username` and `Password` fields of individual RabbitMQ nodes set up in Build RabbitMQ cluster.



6. Click the Check icon in the bottom right to complete.
7. Ensure all RabbitMQ nodes are reachable by the MD Cluster **Control Center**.



High Availability support for Redis

> **Info**
>
> A minimum of **two** Redis instances must be installed on separate hosts for High Availability solution to function properly.

An **odd number** of **Redis Sentinels** should be installed.

Redis Sentinel

1. Install Redis instances on servers.

2. Select one instance as primary. In **Linux Terminal** of the other instances (replicas), run the following command:

Code · bash

```
# Debian/Ubuntu or Red Hat/Rocky
$ redis-cli replicaof <primary_host> <primary_port>
```

3. Build configuration file for **Redis Sentinel**.

Code · bash

```
# The port on which the Sentinel should run
port <SENTINEL_PORT>

# By default Redis does not run as a daemon. Use 'yes' if you need it.
# Note that Redis will write a pid file in /var/run/redis.pid when daemonized.
daemonize yes

sentinel monitor myprimary <PRIMARY_IP> <PRIMARY_PORT> 2
# sentinel monitor <master-name> <ip> <port> <quorum>
# quorum is the number of Sentinels that need to agree about the
# fact the master is not reachable, in order to really mark the master as
# failing, and eventually start a failover procedure if possible.


sentinel down-after-milliseconds myprimary 2000
# means sentinel will consider master down after 2 seconds

sentinel failover-timeout myprimary 4000
# means the chosen sentinel has 4 seconds to perform failover


sentinel parallel-syncs myprimary 2
# sets the number of replicas that can be reconfigured to use the new master
# after a failover at the same time. The lower the number, the more time it
# will take for the failover process to complete, however if the replicas are
# configured to serve old data, you may not want all the replicas to
# re-synchronize with the master at the same time. While the replication process is
# mostly non blocking for a replica, there is a moment when it stops to
# load the bulk data from the master. You may want to make sure only one
# replica at a time is not reachable by setting this option to the value of 1.
```

> **Info**
>
> Duplicate the configuration file and modify `SENTINEL_PORT` to the appropriate port that the Redis Sentinel instance listens on.

4. Install Redis Sentinel instances on servers with the corresponding configuration files.

Code · bash

```
# Debian/Ubuntu or Red Hat/Rocky
$ sudo redis-server </path/to/sentinel-config-file> --sentinel
```

5. Verify the Redis primary and its replicas. In **Linux Terminal** of any machine, run the following command:

Code · bash

```
# Debian/Ubuntu or Red Hat/Rocky
$ redis-cli -h <sentinel_host> -p <sentinel_port>

# Provides information about the Primary
> sentinel master myprimary

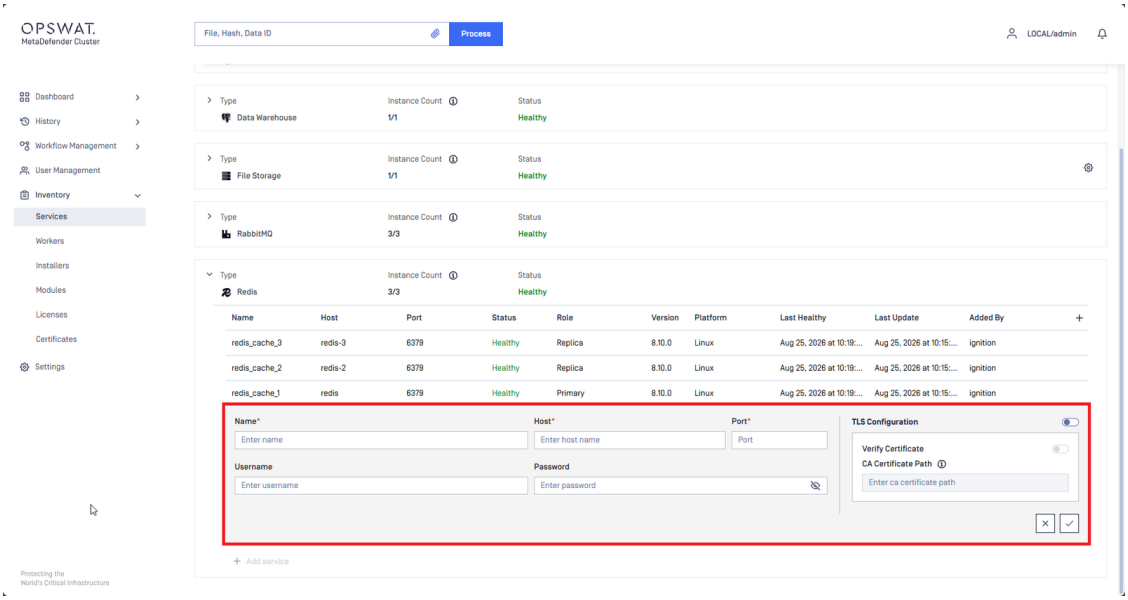
# Gives you information about the replicas connected to the Primary
> sentinel replicas myprimary

# Provides information on the other Sentinels
> sentinel sentinels myprimary

# Provides the IP address of the current Primary
> sentinel get-master-addr-by-name myprimary
```

Setup instructions

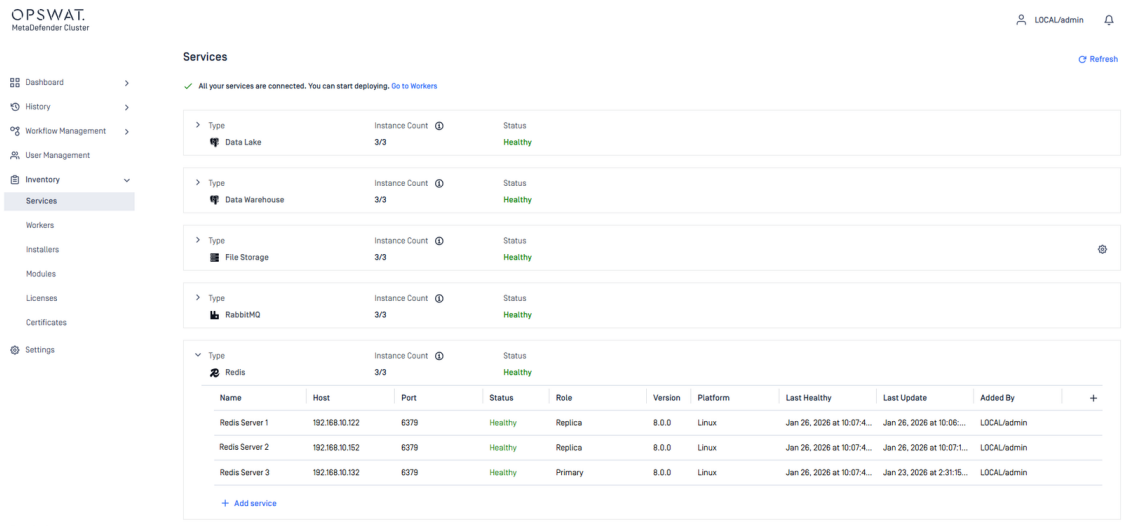
1. Sign to MD Cluster **Control Center** console with your Administrator account.
2. Navigate to `Inventory` > `Services`.
3. Expand the `Redis` group.
4. Click `Add service`.
5. Enter the values for `Name`, `Host`, `Port`, `Username` and `Password` fields of individual Redis instance.



> **Warning**
>
> MD Cluster Control Center **only accepts Redis** and **not Redis Sentinel**.

6. Click the Check icon in the bottom right to complete.

7. Ensure all RabbitMQ nodes are reachable by the MD Cluster **Control Center**.



High Availability support for PostgreSQL Data lake

Installation

> **Info**
>
> - Replication Manager is compatible solely with Linux-based operating systems.
> - The Replication Manager version in use must be compatible with the major version of the installed

PostgreSQL.

- All PostgreSQL servers must be of the same version and run on the same operating system.

High availability solution for PostgreSQL data lake requires a single primary server along with a minimum of two standby servers. Both PostgreSQL and Replication Manager must be installed on every server.

Warning

On the servers that target to run as standby:

- Do not create a PostgreSQL instance (i.e., do not execute `initdb` or any database creation scripts provided by packages).
- Ensure the destination data directory exists and is owned by the `postgres` system user.

1. Select your Linux distribution here and follow the steps to install PostgreSQL accordingly.
2. Follow the steps to install Replication Manager for PostgreSQL clusters - repmgr.

Primary configuration

1. Choose one of the installed servers to be the primary one.
2. Navigate to the folder containing `postgresql.conf` file and create a replication config file named `postgresql.replication.conf`.

Info

By default, `postgresql.conf` file is placed at

- `/var/lib/pgsql/<version>/data/` on Red Hat/Rocky.
- `/var/lib/postgresql/<version>/main` on Debian/Ubuntu.

Code · bash

```
# Enable replication connections; set this value to at least one more
# than the number of standbys which will connect to this server
# (note that repmgr will execute "pg_basebackup" in WAL streaming mode,
# which requires two free WAL senders).
#
# See: https://www.postgresql.org/docs/current/runtime-config-replication.html#GUC-MAX-WAL-
SENDERS

max_wal_senders = 10

# If using replication slots, set this value to at least one more
```

```
# than the number of standbys which will connect to this server.
# Note that repmgr will only make use of replication slots if
# "use_replication_slots" is set to "true" in "repmgr.conf".
# (If you are not intending to use replication slots, this value
# can be set to "0").
#
# See: https://www.postgresql.org/docs/current/runtime-config-replication.html#GUC-MAX-
REPLICATION-SLOTS

max_replication_slots = 10

# Ensure WAL files contain enough information to enable read-only queries
# on the standby.
#
# See: https://www.postgresql.org/docs/current/runtime-config-wal.html#GUC-WAL-LEVEL

wal_level = 'hot_standby'

# Enable read-only queries on a standby
#
# See: https://www.postgresql.org/docs/current/runtime-config-replication.html#GUC-HOT-STANDBY

hot_standby = on

# Enable WAL file archiving
#
# See: https://www.postgresql.org/docs/current/runtime-config-wal.html#GUC-ARCHIVE-MODE

archive_mode = on

# Set archive command to a dummy command; this can later be changed without
# needing to restart the PostgreSQL instance.
#
# See: https://www.postgresql.org/docs/current/runtime-config-wal.html#GUC-ARCHIVE-COMMAND

archive_command = '/bin/true'

# This config should be added if you plan to use repmgrd for
# automatic failover
# See: https://www.repmgr.org/docs/current/repmgrd-basic-configuration.html
shared_preload_libraries = 'repmgr'

wal_log_hints = on # for pg_rewind when rejoin
```

3. Add the replication configuration file name to the end of `postgresql.conf` file and save the modifications.

Code · bash

```
...
```

```
include 'postgresql.replication.conf'
```

4. In Terminal, run the following commands to create `repmgr` user and database.

Code · bash

```
$ createuser -s repmgr
$ createdb repmgr -O repmgr
```

Info

In this guideline, although the term `repmgr` is used for both user and database, any names can be used.

5. Edit `pg_hba.conf` file to configure the authentication.

Code · bash

```
# Ensure the repmgr user has appropriate permissions in pg_hba.conf
# and can connect in replication mode
# pg_hba.conf should contain entries similar to the following:
# Uncomment this if you want to access Postgresql database via pgadmin with user "postgres":
#host    all             postgres        0.0.0.0/0        scram-sha-256

local   replication   repmgr                              trust
host    replication   repmgr      127.0.0.1/32            trust
#or
host    replication   repmgr      0.0.0.0/0               trust

local   repmgr        repmgr                              trust
host    repmgr        repmgr      127.0.0.1/32            trust
#or
host    repmgr        repmgr      0.0.0.0/0               trust
```

6. Restart PostgreSQL server.

Code · bash

```
$ cd /path/to/pg_ctl
$ pg_ctl -D <postgresql_data_dir> restart
```

7. Create `repmgr.conf` file, fill out information in brackets and store it in a location of your choice.

Warning

`repmgr.conf` file should not be placed inside PostgreSQL data folder as it may be overwritten.

Code · bash

```
node_id=<any_node_id>
node_name=<any_node_name>
# connection info of the current node
conninfo='host=<host_address_of_node> user=repmgr dbname=repmgr connect_timeout=2'
data_directory='<postgres_data_dir>'
failover='automatic' # for repmgrd (automatic failover)
promote_command='<postgres_dir>/repmgr standby promote -f "<your_dir>/repmgr.conf" --log-level
INFO'
follow_command='<postgres_dir>/repmgr standby follow -f "<your_dir>/repmgr.conf" -W --log-level
INFO'
reconnect_attempts='5'
reconnect_interval='1'
monitor_interval_secs='1'
pg_bindir='<postgres_dir>'
# enable this so that repmgr only vote new primary
# when none of the standbys can connect to current primary
primary_visibility_consensus=true
```

Key	Red Hat/Rocky	Debian/Ubuntu
`postgres_data_dir`	`/var/lib/pgsql/<version>/data/`	`/var/lib/postgresql/<version>/main/`
`postgres_dir`	`/usr/pgsql-<version>/bin/`	`/usr/lib/postgresql/<version>/bin/`
`your_dir`	Directory to `repmgr.conf` file.	Directory to `repmgr.conf` file.

8. In Terminal, run the following commands to register the primary server.

Code · bash

```
$ cd path/to/repmgr
$ repmgr -f <repmgr_config_file_path> primary register

INFO: connecting to primary database...
NOTICE: attempting to install extension "repmgr"
NOTICE: "repmgr" extension successfully installed
NOTICE: primary node record (id: 1) registered
```

Standby configuration

1. Create `repmgr.conf` file and modify values of `node`, `node_name`, `conninfo` accordingly.
2. Store the file in your reference location.
3. Stop PostgreSQL server.

Code · bash

```
$ cd /path/to/pg_ctl
$ pg_ctl -D <postgresql_data_dir> stop
```

4. In Terminal, run the following commands to clone data from the primary server.

Code · bash

```
$ cd path/to/repmgr
$ repmgr -h <primary_server_host> \
         -U repmgr -d repmgr \ # primary repmgr <user> and <database>
         -f <standby_repmgr_config_file_path> \
         -c \ # fast checkpoint to speed up process
         standby clone \
         --dry-run # dry run to check if the primary can be cloned

$ repmgr -h <primary_server_host> \
         -U repmgr -d repmgr \ # primary repmgr <user> and <database>
         -f <standby_repmgr_config_file_path> \
         -c \ # fast checkpoint to speed up process
         standby clone
```

5. Start PostgreSQL server.

Code · bash

```
$ cd /path/to/pg_ctl
$ pg_ctl -D <postgresql_data_dir> start
```

6. In Terminal, run the following commands to register the standby server.

Code · bash

```
$ cd /path/to/repmgr
$ repmgr -f <standby_repmgr_config_file_path> \
            standby register
```

7. Check if the node was registered successfully.

Code · bash

```
$ cd /path/to/repmgr
$ repmgr -f /etc/repmgr.conf cluster show
```

Automatic failover

In Terminal, run the following command to start Replication manager daemon on all PostgreSQL servers (including primary and standbys)

Code · bash

```
$ cd /path/to/repmgr
$ repmgrd -f <repmgr_config_file_path>
```

Rejoin after a failure

Info

Replication manager daemon `repmgrd` does not automatically join a failed PostgreSQL server node to the cluster. Consequently, the cluster contains at least two primary nodes at one time, and the system administrator has to join the node to the cluster manually.

1. Ensure the failed PostgreSQL server is not running. Run the following command in Terminal to stop the server if it has, by chance, already been started by the Linux system and service manager.

Code · bash

```
$ cd /path/to/pg_ctl
$ pg_ctl -D <postgresql_data_dir> stop
```

2. In Terminal, run the following command to rejoin the server.

Code · bash

```
$ cd /path/to/repmgr
$ repmgr -f <repmgr_config_file> node rejoin \
         --force-rewind \ # use pg_rewind to help with diverge timeline
         -d 'host=<current_primary>  dbname=repmgr user=repmgr'
```

3. If the server rejoin fails, do register it as a standby. In Terminal, run the following command.

Code · bash

```
$ cd /path/to/repmgr
```

```
$ repmgr -h <current_primary_server_host> \
         -U repmgr -d repmgr \ # primary repmgr <user> and <database>
         -f <standby_repmgr_config_file_path> \
          -c \ # fast checkpoint to speed up process
          -F \ # this overwritten the the data folder if it was created
          standby clone \
```

4. Start PostgreSQL server.

Code · bash

```
$ cd /path/to/pg_ctl
$ pg_ctl -D <postgresql_data_dir> start
```

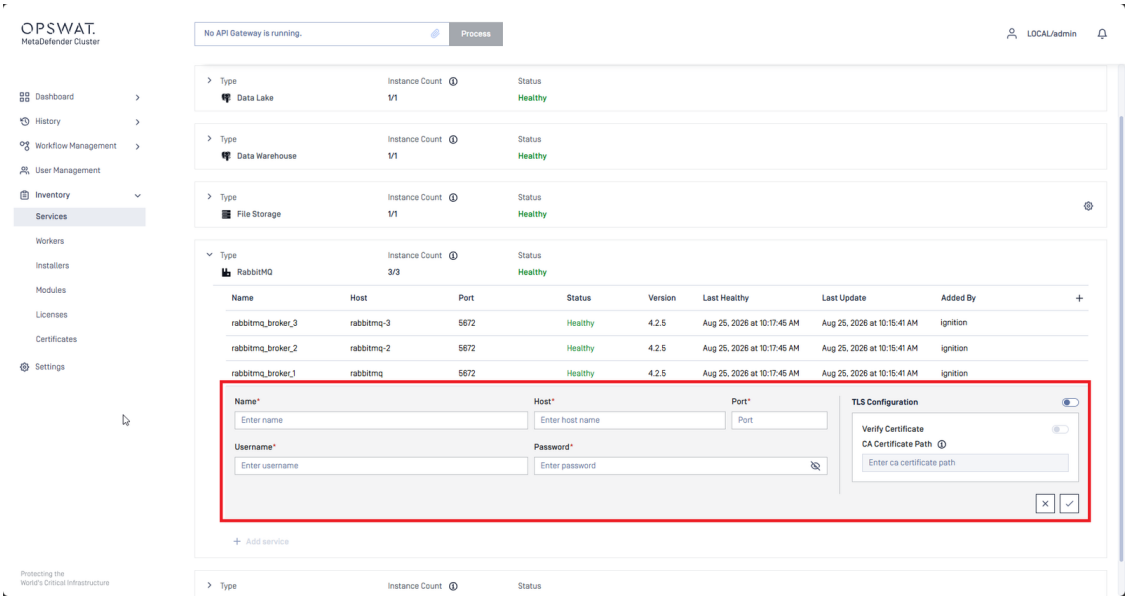
5. Force register the server as a standby.

Code · bash

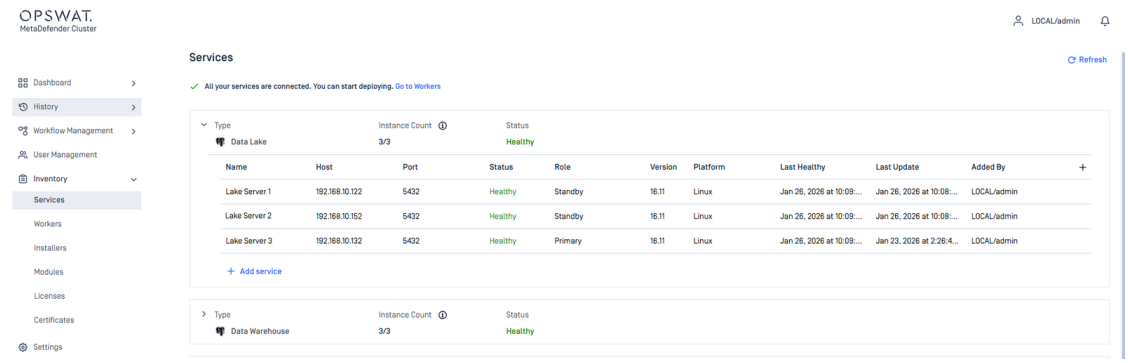
```
$ cd /path/to/repmgr
$ repmgr -f <standby_repmgr_config_file_path> \
         -F \ # forcefully overwrite an existing node record or user --force
         standby register
```

Setup instructions

1. Sign to MD Cluster **Control Center** console with your Administrator account.
2. Navigate to `Inventory` > `Services`.
3. Expand the `Data Lake` group.
4. Click `Add service`.
5. Enter the values for `Name`, `Host`, `Port`, `Username` and `Password` fields of individual PostgreSQL instance.
6. Click the Check icon in the bottom right to complete.



7. Ensure all PostgreSQL instances are reachable by the MD Cluster **Control Center.**



System settings

This section shows MetaDefender Cluster settings.

Remote Support Package Gathering

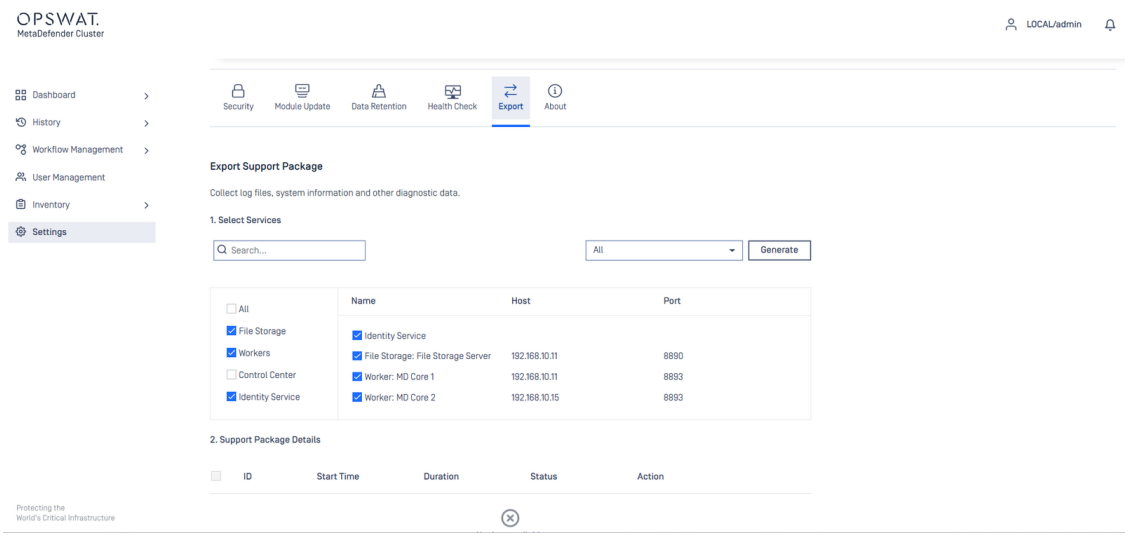
The support package contains log files and is essential for OPSWAT to troubleshoot issues. Since version **2.2.0**, it is now possible to gather a support package remotely via the web console of the MetaDefender Cluster Control Center.

> **Info**
>
> Ensure that all MD Cluster services are upgraded to version 2.2.0 or higher to fully support this feature.

Remote support package gathering steps:

1. Go to `Settings` > `Export` .

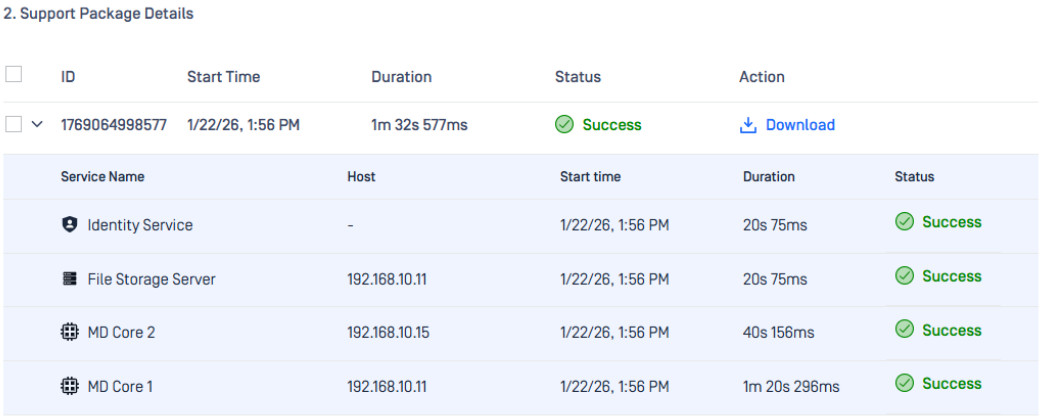
2. Select which MetaDefender Cluster services need to generate support package, then select `Generate`



Info

Select the MetaDefender Cluster Worker service to generate a support package for its deployed instance as well.

3. Wait for the generation process to complete successfully. Once it is done, the download button will appear, and the support packages will be ready for download.



Info

The size of the support package may vary depending on log size and the number of days for collection. If disk space is insufficient, certain log files may be excluded from the support package.

Warning

All support package files will be downloaded to the MetaDefender Cluster Control Center. Please monitor the disk space on the host running this service when using this functionality, as the size of log files can be very large.

4. Click `Download`

> **Info**
>
> Some services may fail due to connection issues or insufficient disk space. In such cases, only the successfully generated support packages will be available for download. Users can view detailed error information if failures occur.

Security

Setup HTTPS

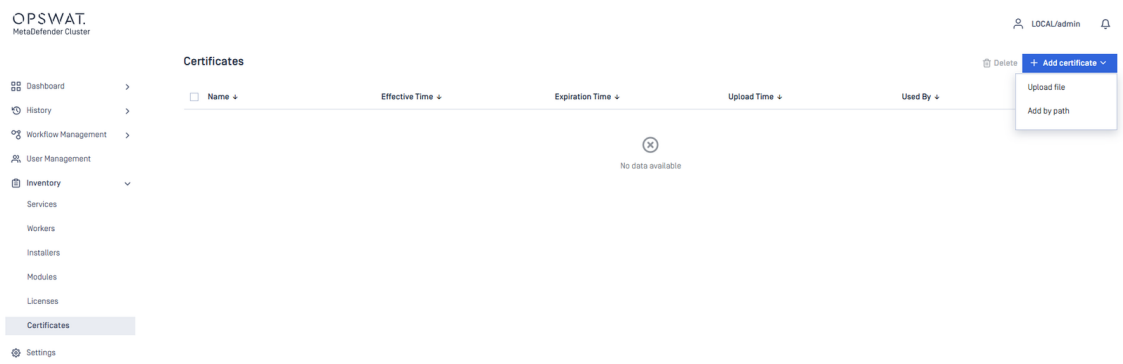
Transport Layer Security (TLS) is a cryptographic protocol that provides communications security over a computer network. Websites, like the Web Management Console, are able to use TLS to secure all communications between their servers and web browsers.

The TLS protocol aims primarily to provide confidentiality (privacy) and data integrity between two communicating computer applications.

> **Info**
>
> HTTPS is not enabled by default. As a consequence sessions between the wizard's backend and the browser may be insecure.

Steps to setup this feature:

1. Go to `Inventory`> `Certificates`
2. Click `Add certificate`
    1. To add a certificate using a file path, choose `Add by path` and enter the location of both the certificate and its corresponding private key file.
    2. To upload certificate file, select `Upload file`.



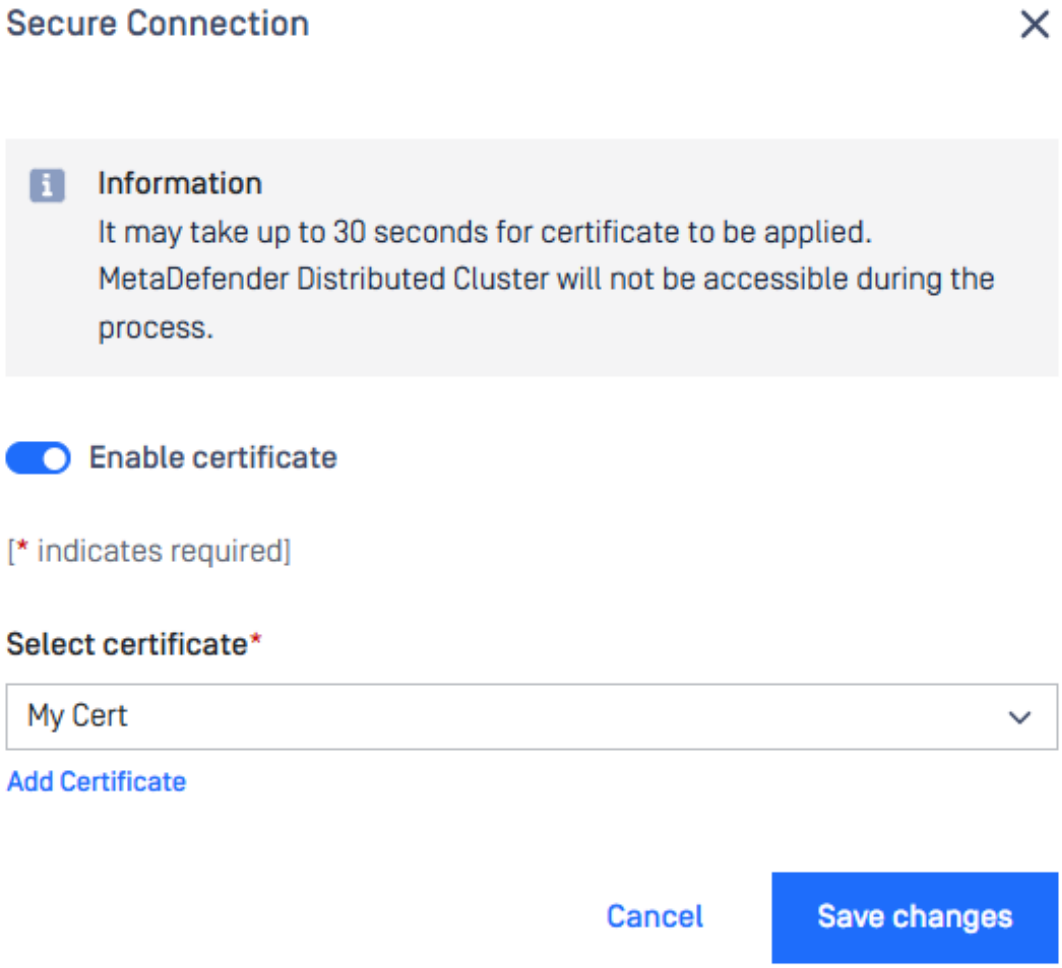
Certificate YML sample file:

Code · yaml

```
---
private_key: |
  -----BEGIN PRIVATE KEY-----
  MIIJQwIBADANBgkqhkiG9w0BAQEFAASCCS0wggkpAgEAAoICAQCjYtuWaICCY0tJ
  PubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8Di3DAmHK
  nSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTekLWcfI5ZZ
  toGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnHKT/m6DSU
  0xlZ13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6jzhNyMBT
  J1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQUjy5N8pS
  Np7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAzTK4l2pHN
  uC53QVc/EF++GBLAxmvCDq9ZpMIYi7OmzkkAKKC9Ue6Ef217LFQCFIBKIzv9cgi9
  fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqnV0nPN1IMSn
  zXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnWOFIaVrKWLzAtgzoGMTvP/Au
  ehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9psNcjTMaB
  QLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABAoICAFWe8MQZb37k2gdAV3Y6aq8f
  qokKQqbCNLd3giGFwYkezHXoJfg6Di7oZxNcKyw35LFEghkgtQqErQqo35VPIoH+
  vXUpWOjnCmM4muFA9/cX6mYMc8TmJsg0ewLdBCOZVw+wPABlaqz+0UOiSMMftpk9
  fz9JwGd8ERyBsT+tk3Qi6D0vPZVsC1KqxxL/cwIFd3Hf2ZBtJXe0KBn1pktWht5A
  Kqx9mld2Ovl7NjgiC1Fx9r+fZw/iOabFFwQA4dr+R8mEMK/7bd4VXfQ1o/QGGbMT
  G+ulFrsiDyP+rBIAaGC0i7gDjLAIBQeDhP409ZhswIEc/GBtODU372a2CQK/u4Q/
  HBQvuBtKFNkGUooLgCCbFxzgNUGc83GB/6IwbEM7R5uXqsFiE71LpmroDyjKTlQ8
  YZkpIcLNVLw0usoGYHFm2rvCyEVlfsE3Ub8cFyTFk50SeOcF2QL2xzKmmbZEpXgl
  xBHR0hjgon0IKJDGfor4bHO7Nt+1Ece8u2oTEKvpz5aIn44OeC5mApRGy83/0bvs
  esnWjDE/bGpoT8qFuy+0urDEPNId44XcJm1IRIlG56ErxC3l0s11wrIpTmXXckqw
  zFR9s2z7f0zjeyxqZg4NTPI7wkM3M8BXlvp2GTBIeoxrWB4V3YArwu8QF80QBgVz
  mgHl24nTg00UH1OjZsABAoIBAQDOxftSDbSqGytcWqPYP3SZHAWDA0O4ACEM+eCw
  au9ASutl0IDlNDMJ8nC2ph25BMe5hHDWp2cGQJog7pZ/3qQogQho2gUniKDifN77
  40QdykllTzTVROqmP8+efreIvqlzHmuqaGfGs5oTkZaWj5su+B+bT+9rIwZcwfs5
  YRINhQRx17qa++xh5mfE25c+M9fiIBTiNSo4lTxWMBShnK8xrGaMEmN7W0qTMbFH
  PgQz5FcxRjCCqwHilwNBeLDTp/ZECEB7y34khVh531mBE2mNzSVIQcGZP1I/DvXj
  W7UUNdgFwii/GW+6M0uUDy23UVQpbFzcV8o1C2nZc4Fb4zwBAoIBAQDKSJkFwwuR
  naVJS6WxOKjX8MCu9/cKPnwBv2mmI2jgGxHTw5sr3ahmF5eTb8Zo19BowytN+tr6
  2ZFoIBA9Ubc9esEAU8l3fggdfM82cuR9sGcfQVoCh8tMg6BP8IBLOmbSUhN3PG2m
  39I802u0fFNVQCJKhx1m1MFFLOu7lVcDS9JN+oYVPb6MDfBLm5jOiPuYkFZ4gH79
  J7gXI0/YKhaJ7yXthYVkdrSF6Eooer4RZgma62Dd1VNzSq3JBo6rYjF7Lvd+RwDC
  R1thHrmf/IXplxpNVkoMVxtzbrrbgnC25QmvRYc0rlS/kvM4yQhMH3eA7IycDZMp
  Y+0xm7I7jTT7AoIBAGKzKIMDXdCxBWKhNYJ8z7hiItNl1IZZMW2TPUiY0rl6yaCh
  BVXjM9W0r07QPnHZsUiByqb743adkbTUjmxdJzjaVtxN7ZXwZvOVrY7I7fPWYnCE
  fXCr4+IVpZI/ZHZWpGX6CGSgT6EOjCZ5IUufIvEpqVSmtF8MqfXO9o9uIYLokrWQ
  x1dBl5UnuTLDqw8bChq7O5y6yfuWaOWvL7nxI8NvSsfj4y635gIa/0dFeBYZEfHI
  UlGdNVomwXwYEzgE/c19ruIowX7HU/NgxMWTMZhpazlxgesXybel+YNcfDQ4e3RM
  OMz3ZFiaMaJsGGNf4++d9TmMgk4Ns6oDs6Tb9AECggEBAJYzd+SOYo26iBu3nw3L
  65uEeh6xou8pXH0Tu4gQrPQTRZZ/nT3iNgOwqu1gRuxcq7TOjt41UdqIKO8vN7/A
  aJavCpaKoIMowy/aGCbvAvjNPpU3unU8jdl/t08EXs79S5IKPcgAx87sTTi7KDN5
  SYt4tr2uPEe53NTXuSatilG5QCyExIELOuzWAMKzg7CAiIlNS9foWeLyVkBgCQ6S
  me/L8ta+mUDy37K6vC34jh9vK9yrwF6X44ItRoOJafCaVfGI+175q/eWcqTX4q+I
  G4tKls4sL4mgOJLq+ra50aYMxbcuommctPMXU6CrrYyQpPTHMNVDQy2ttFdsq9iK
```

```
    TncCggEBAMmt/8yvPflS+xv3kg/ZBvR9JB1In2n3rUCYYD47ReKFqJ03Vmq5C9nY
    56s9w7OUO8perBXlJYmKZQhO4293lvxZD2Iq4NcZbVSCMoHAUzhzY3brdgtSIxa2
    gGveGAezZ38qKIU26dkz7deECY4vrsRkwhpTW0LGVCpjcQoaKvymAoCmAs8V2oMr
    Ziw1YQ9uOUoWwOqm1wZqmVcOXvPIS2gWAs3fQlWjH9hkcQTMsUaXQDOD0aqkSY3E
    NqOvbCV1/oUpRi3076khCoAXI1bKSn/AvR3KDP14B5toHI/F5OTSEiGhhHesgRrs
    fBrpEY1IATtPq1taBZZogRqI3rOkkPk=
    -----END PRIVATE KEY-----
  certificate: |
    -----BEGIN CERTIFICATE-----
    MIIF5jCCA86gAwIBAgIJANq50IuwPFKgMA0GCSqGSIb3DQEBCwUAMIGGMQswCQYD
    VQQGEwJHQjEQMA4GA1UECAwHRXJld2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEb
    MBkGA1UECgwSbGlid2Vic29ja2V0cy10ZXN0MRIwEAYDVQQDDAlsb2NhbGhvc3Qx
    HzAdBgkqhkiG9w0BCQEWEG5vbmVAaW52YWxpZC5vcmcwIBcNMTgwMzIwMDQxNjA3
    WhgPMjExODAyMjQwNDE2MDdaMIGGMQswCQYDVQQGEwJHQjEQMA4GA1UECAwHRXJl
    d2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEbMBkGA1UECgwSbGlid2Vic29ja2V0
    cy10ZXN0MRIwEAYDVQQDDAlsb2NhbGhvc3QxHzAdBgkqhkiG9w0BCQEWEG5vbmVA
    aW52YWxpZC5vcmcwggIiMA0GCSqGSIb3DQEBAQUAA4ICDwAwggIKAoICAQCjYtuW
    aICCY0tJPubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8
    Di3DAmHKnSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTek
    LWcfI5ZZtoGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnH
    KT/m6DSU0xlZ13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6
    jzhNyMBTJ1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQ
    Ujy5N8pSNp7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAz
    TK4l2pHNuC53QVc/EF++GBLAxmvCDq9ZpMIYi7OmzkkAKKC9Ue6Ef217LFQCFIBK
    Izv9cgi9fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqnV0
    nPN1IMSnzXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnWOFIaVrKWLzAtgzo
    GMTvP/AuehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9p
    sNcjTMaBQLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABo1MwUTAdBgNVHQ4EFgQU
    9mYU23tW2zsomkKTAXarjr2vjuswHwYDVR0jBBgwFoAU9mYU23tW2zsomkKTAXar
    jr2vjuswDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAgEANjIBMrow
    YNCbhAJdP7dhlhT2RUFRdeRUJD0IxrH/hkvb6myHHnK8nOYezFPjUlmRKUgNEDuA
    xbnXZzPdCRNV9V2mShbXvCyiDY7WCQE2Bn44z26O0uWVk+7DNNLH9BnkwUtOnM9P
    wtmD9phWexm4q2GnTsiL6Ul6cy0QlTJWKVLEUQQ6yda582e23J1AXqtqFcpfoE34
    H3afEiGy882b+ZBiwkeV+oq6XVF8sFyr9zYrv9CvWTYlkpTQfLTZSsgPdEHYVcjv
    xQ2D+XyDR0aRLRlvxUa9dHGFHLICG34Juq5Ai6lM1EsoD8HSsJpMcmrH7MWw2cKk
    ujC3rMdFTtte83wF1uuF4FjUC72+SmcQN7A386BC/nk2TTsJawTDzqwOu/VdZv2g
    1WpTHlumlClZeP+G/jkSyDwqNnTu1aodDmUa4xZodfhP1HWPwUKFcq8oQr148QYA
    AOlbUOJQU7QwRWd1VbnwhDtQWXC92A2w1n/xkZSR1BM/NUSDhkBSUU1WjMbWg6Gg
    mnIZLRerQCu1Oozr87rOQqQakPkyt8BUSNK3K42j2qcfhAONdRl8Hq8Qs5pupy+s
    8sdCGDlwR3JNCMv6u48OK87F4mcIxhkSefFJUFII25pCGN5WtE4p5l+9cnO1GrIX
    e2Hl/7M0c/lbZ4FvXgARlex2rkgS0Ka06HE=
    -----END CERTIFICATE-----
```

3. Go to `Settings` > `Security`
4. On the `Secure Connection` section, click `Details`
5. Select `Enable Certificate`, then select your certificate added in step 2.



Warning

Applying HTTPS settings may take some time. During this process, the MetaDefender Cluster Control Center web console will be temporarily unavailable.

Password policies

Password Policy settings are accessible under `Settings` > `Security`tab.

Info

These password policies changes only apply to new user creations and future password changes. Existing users' passwords are unaffected.

Local users' password can be enforced to meet requirements set by administrators, which includes following constraints:

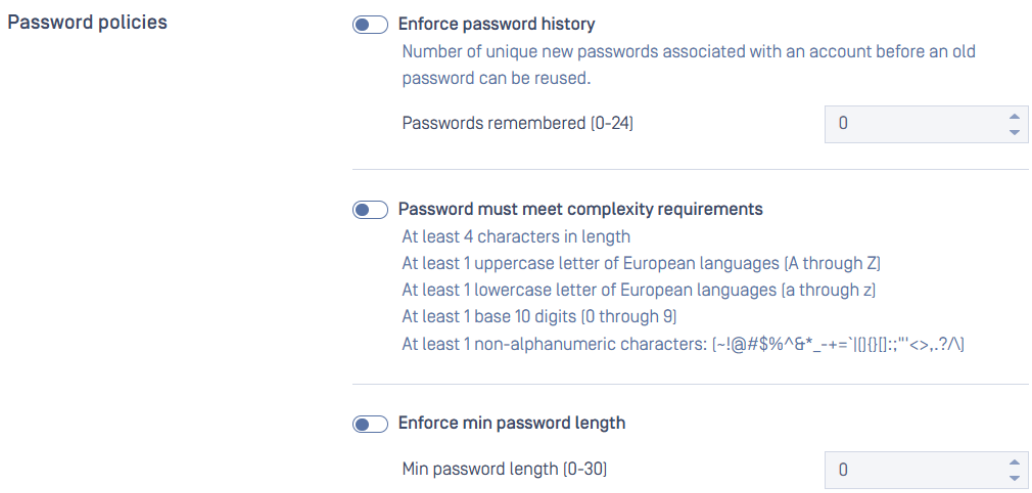
- **Enforce password policy:**
  - Determines the number of unique new passwords that must be associated with a user account before an old

password can be reused

- Range: [0-24]
- Default: 0 (to disable enforcement)

- **Minimum password length:**
  - The least number of characters that can make up a password for a user account
  - Range: [0-30]
  - Default: 0 (to disable enforcement)

- **Password must meet complexity requirements:**
  - Determines whether passwords must meet a series of guidelines that are considered important for a strong password.
  - Default: unchecked



Session policies

Administrators can enforce session policies for local users to ensure compliance with organizational requirements, using the following settings:

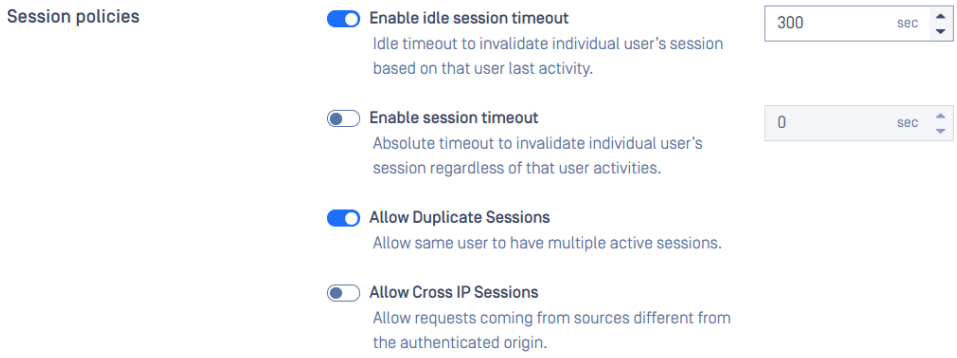
- **Enable idle session timeout:**
  - Idle timeout automatically terminates a user's session based on how long since their last recorded activity.
  - Default: 300 seconds.

- **Enable session timeout**
  - Absolute timeout terminates an individual user's session after a fixed duration, regardless of any user activity.
  - Default: 0 (to disable enforcement)

- **Allow Duplicate Sessions**

- Permit the same user to log in and operate multiple sessions at once.
- Default: Enabled.

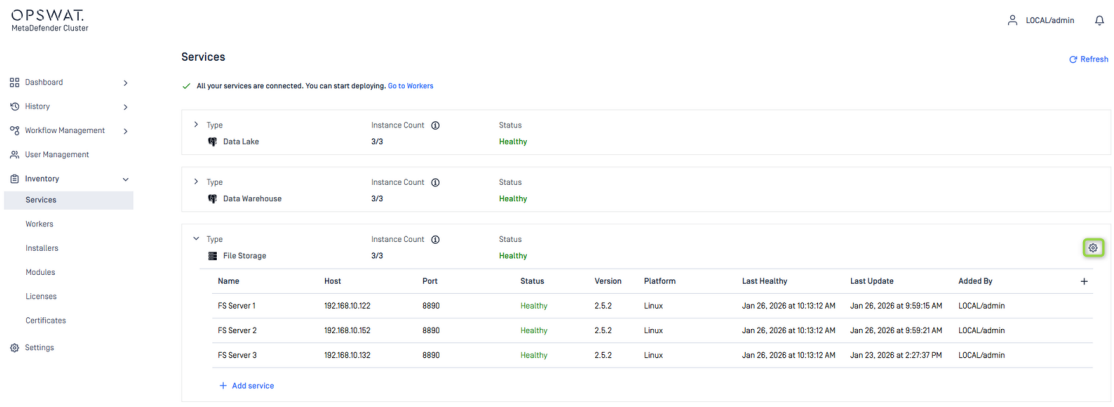
- **Allow Cross IP Sessions**
  - Permit requests from sources other than the authenticated origin.
  - Default: Disabled.



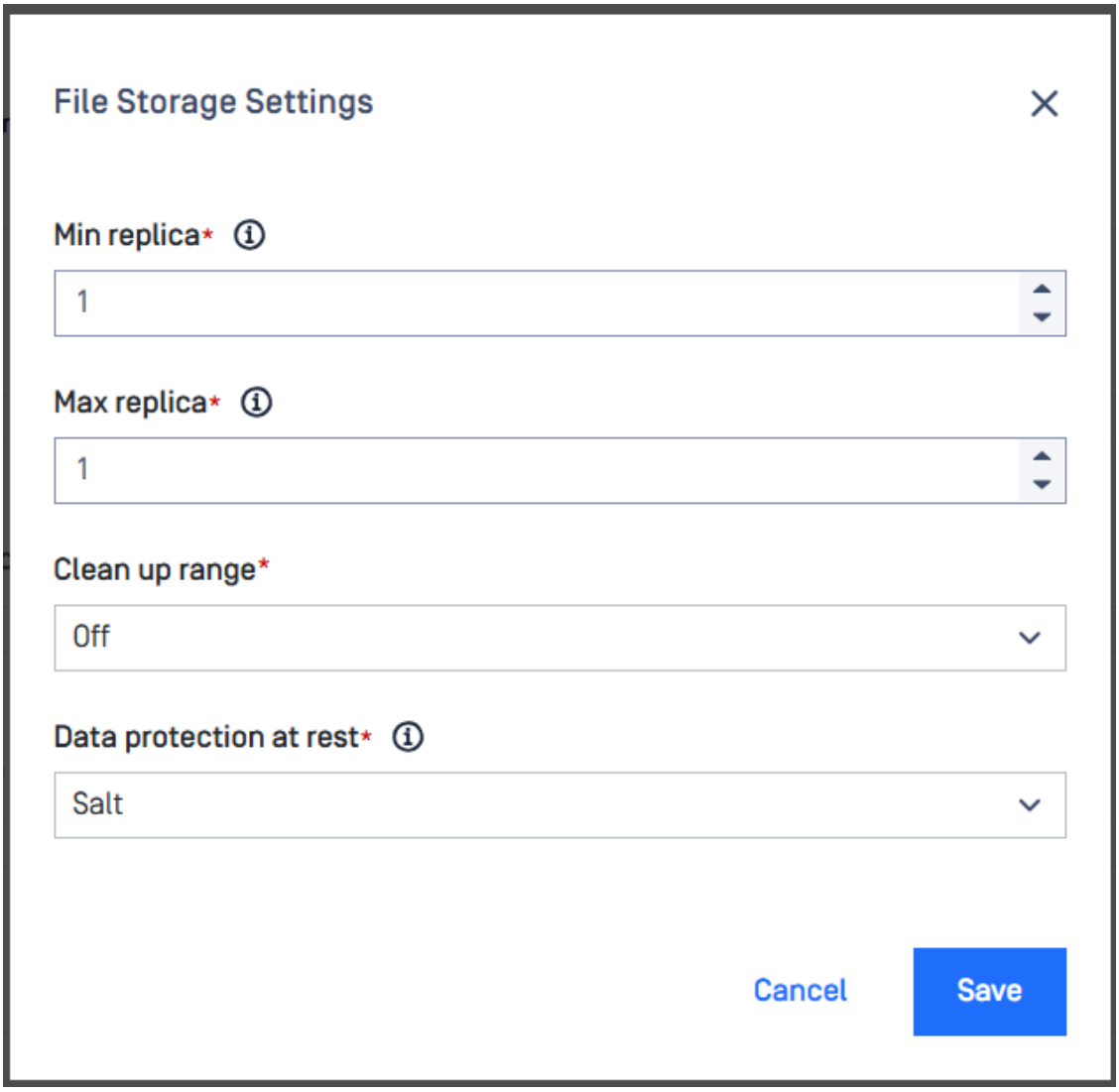
File Storage

MetaDefender Cluster (MD Cluster) introduces a built-in file storage server known as MD Cluster **File Storage**. The server stores and manages the live time of files and their duplications.

The administrator can set up MD Cluster to work with a single instance of MD Cluster **File Storage** or build a group of MD Cluster **File Storage** instances.



From `Inventory` > `Services` of the MD Cluster **Control Center** web console, the administrator can click on the gear icon in the top left corner of the `File Storage` group to access MD Cluster **File Storage** settings.



Multiple instances

When several instances of MD Cluster **File Storage** are added to the File Storage group, `Min Replica` and `Max Replica` enable the administrator to configure the operation of the storage group, as shown in the table below.

Setting	Behavior
`Min replica` = 1 `Max replica` = 1	Every file is stored without a backup across all File Storage servers. File Storage servers in the group implement a **Sharding** solution for file storage. Since there is no backup for any file, if one server in the group goes down, files managed by that server will be lost to the clients. This setup provides the best performance but also poses a high risk of data loss.
`Min replica` > 1 `Max replica` > `Min replica`	Every file is stored on at least `Min replica` number of File Storage servers and at most `Max-replica` number of servers. The setting provides **High**

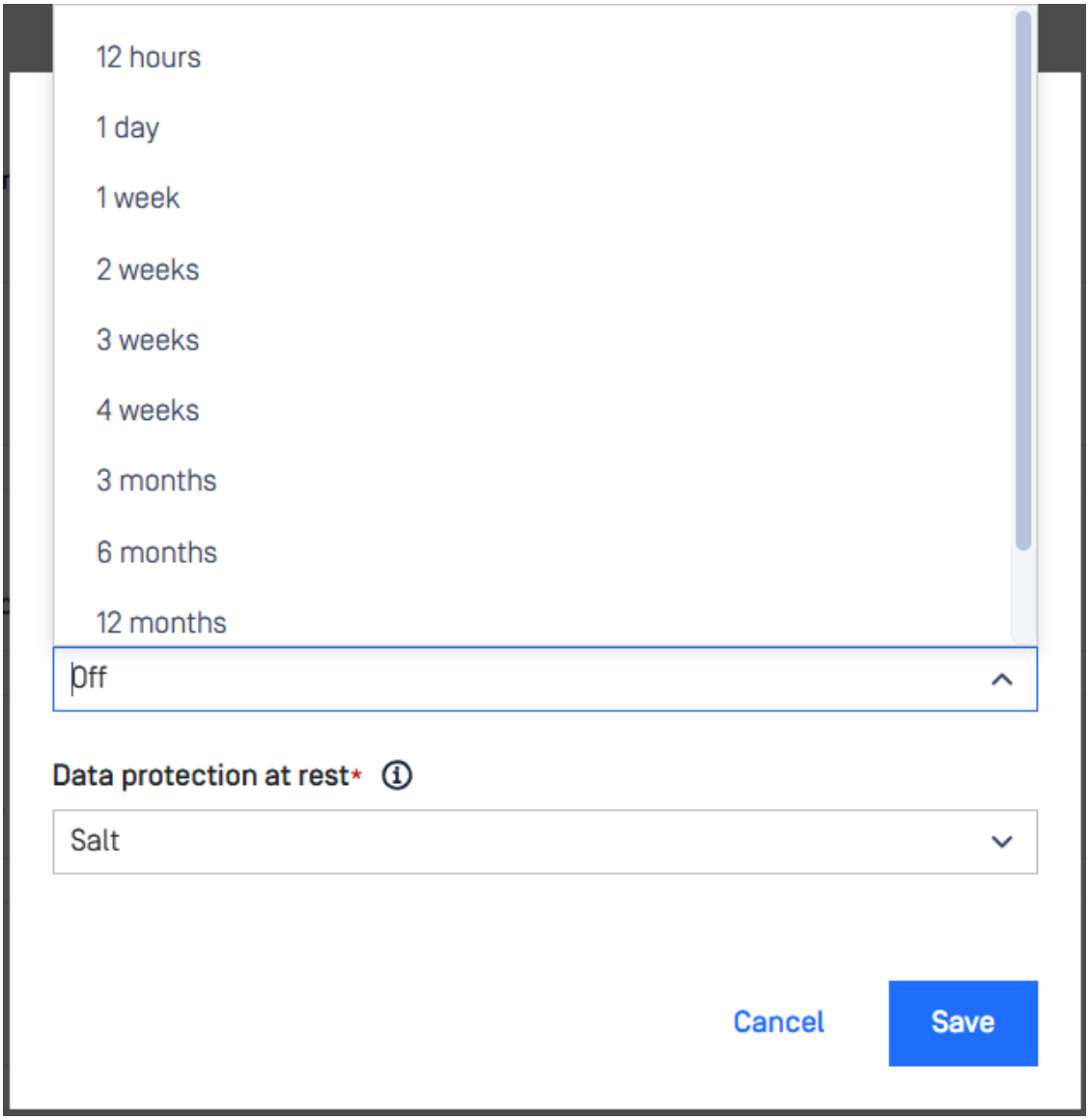
Setting	Behavior
	Availability support for File Storage. In most cases, `Min replica` and `Max replica` are configured to 2 and 3, creating a balance between performance and efficiency in High Availability.
`Min replica` > 1 `Max replica` = `Min replica`	Every file is fully stored on `Max replica` number of file storage servers and will not succeed if it can not be. This setting is the strictest among three options and should be considered carefully due to its impact on system performance.

Warning

Replication of a file across several MD Cluster File Storage servers significantly impacts the overall system performance. Hence, the number of replications must be evaluated thoughtfully.

Data retention

The administrator can configure data retention for files stored in the File Storage group with the `Clean up range` option. By default, this option is disabled. The administrator has the option to retain files for 12 hours, 1 day, 1 week, etc., starting from the present time.



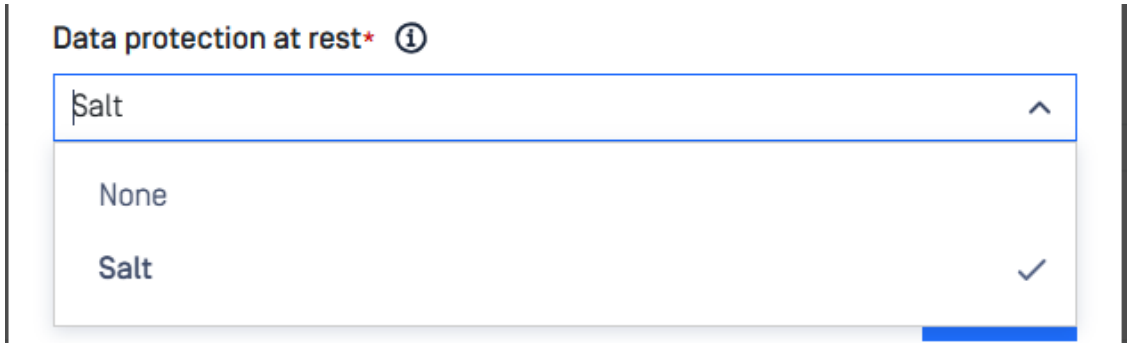
Info

Files eligible for retention include those produced by CDR, DLP, SBOM, Quarantine engines.

Package (.msi, .deb or .rpm) and module files are marked for cleanup manually and will never be affected by data retention.

Data protection

By default, all files stored on the MD Cluster File Storage server are XOR bitwise with a randomly generated binary string. This option is enabled by default to prevent the file from being executed successfully due to unexpected factors. The administrator can disable the option to optimize MD Cluster File Storage performance, though it may expose the system to security risks.


Data protection at rest*
Salt
None
Salt

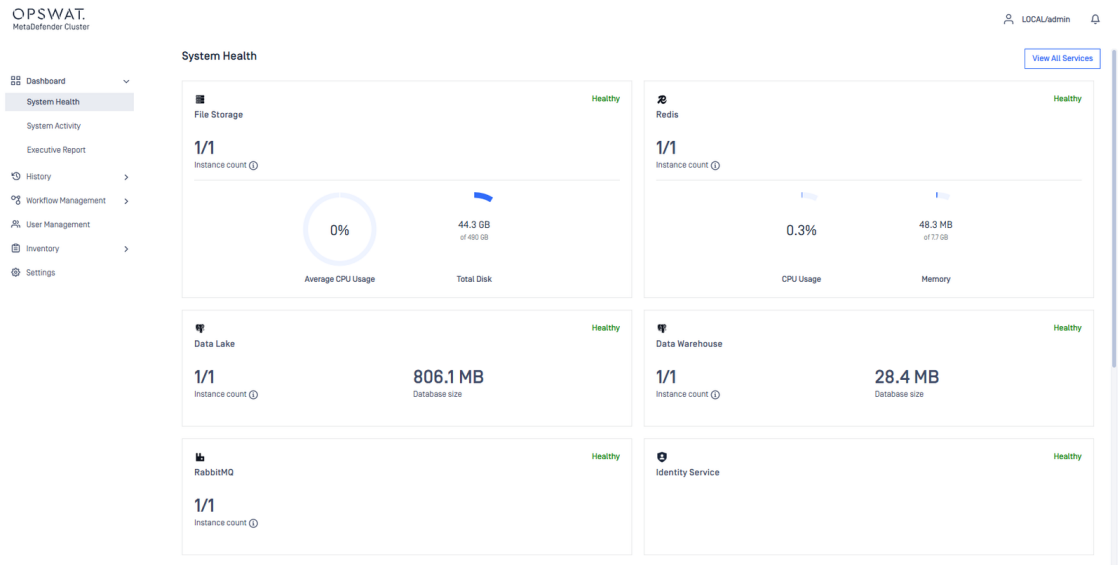
Dashboard

MetaDefender Cluster (MD Cluster) Control Center provides a Web-based user interface at default port 8892 for user to monitor the system health, system activity and executive report.

System Health

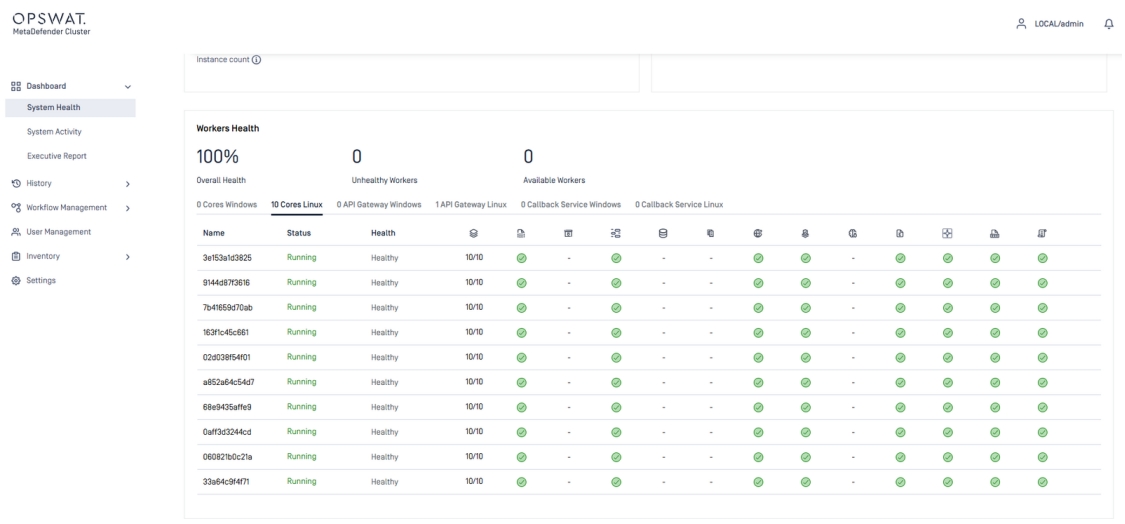
System Health show the status of all related services:

- File Storage shows the number of MD Cluster File Storage instances in File Storage cluster added to MD Cluster. The average CPU usage by the instances and the available over total disk are displayed below.
- Redis shows the number of Redis Caching services in MD Cluster. Only the CPU usage and free memory amount of the master Redis node are presented.
- RabbitMQ shows the number of RabbitMQ message brokers in the cluster.
- Data lake and warehouse shows the number of Postgres instances including the primary and its replicas. Database sizes are shown.
- Identity Service shows the health status of the MD Cluster authentication/authorization service.

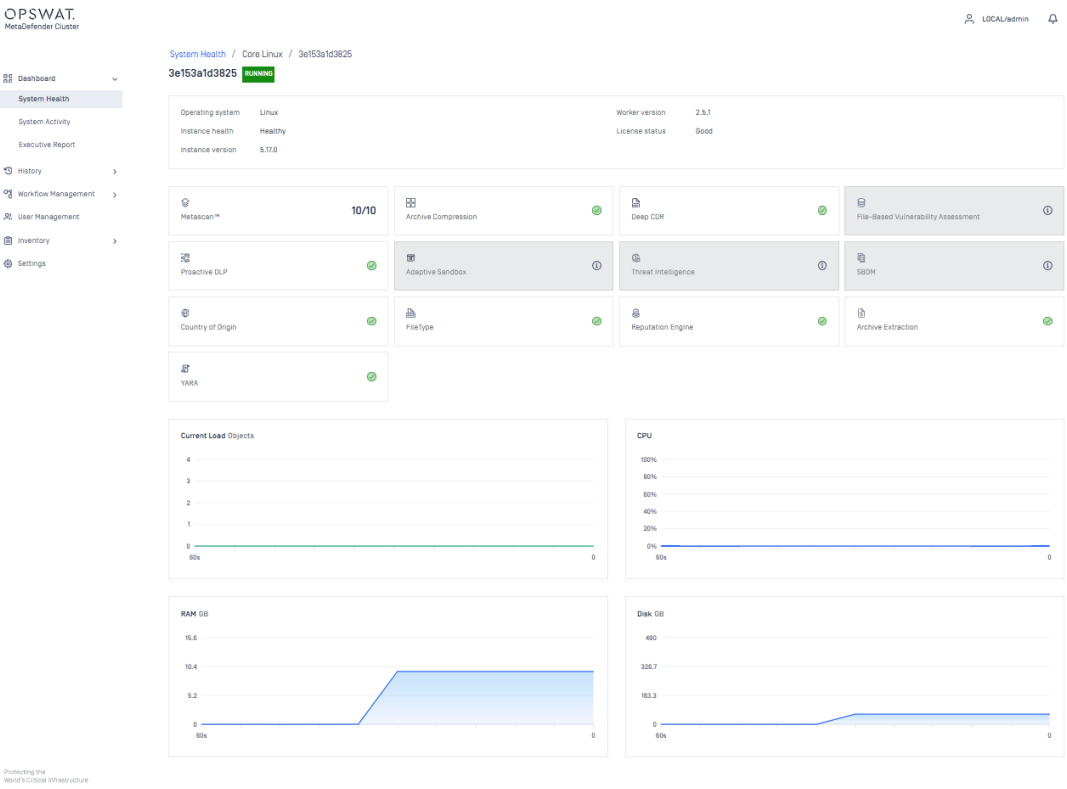


Worker Health

The overall health of all MD Cluster API Gateway, MetaDefender Core instances of platforms are shown here. The administrator can quickly monitor the number of AV engines installed on each MetaDefender Core instance.



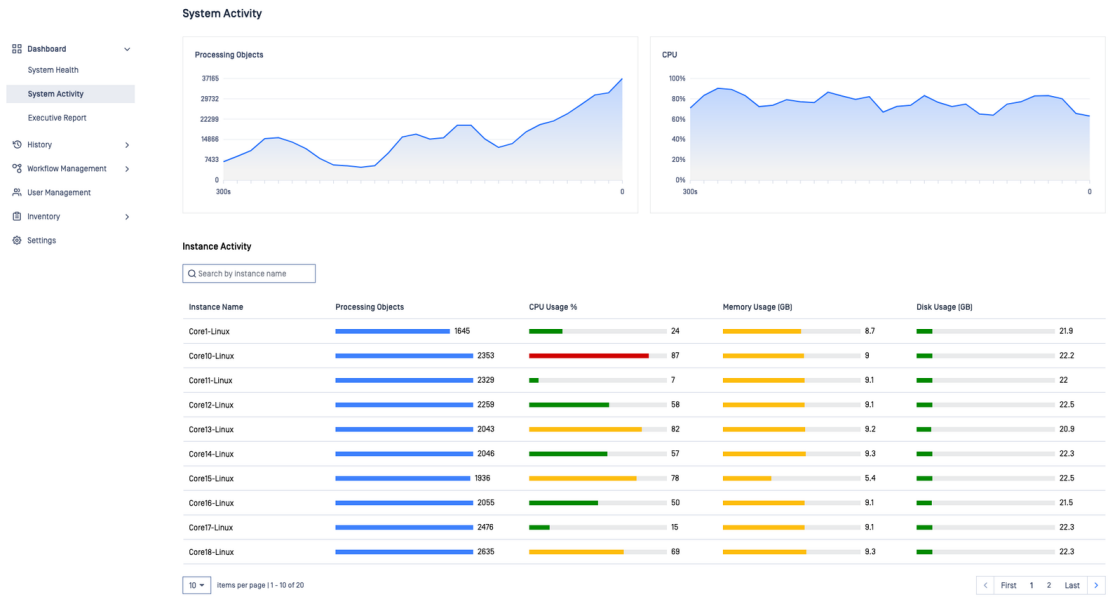
For more details of engines installed on any MetaDefender Core instance, the administrator can hit on the row of that instance. A new page show.



System Activity

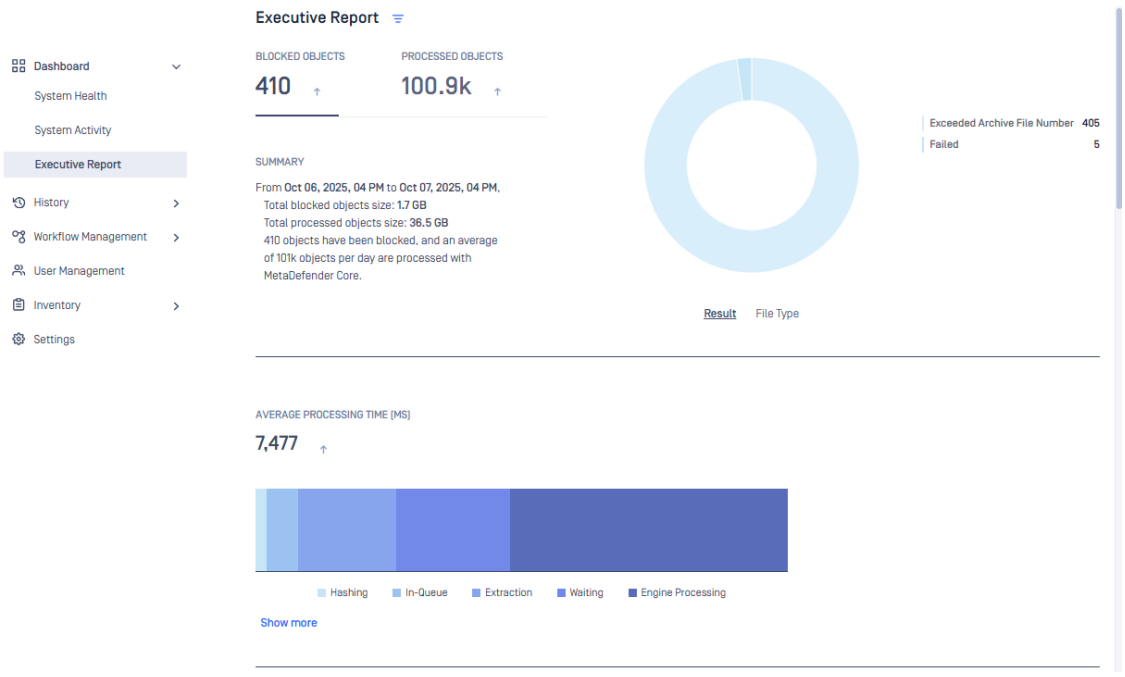
The overall System Activity of all MetaDefender Core instances are shown here. There are two sections of this page, System Activity and Instance Activity.

- System Activity shows the overall objects currently being processed, objects currently in-queue and average CPU usage of all MetaDefender Core instances.
- Instance Activity shows individual statistics of all MetaDefender Core instances. This includes Processing Objects, CPU Usage, Memory Usage and Disk Usage.

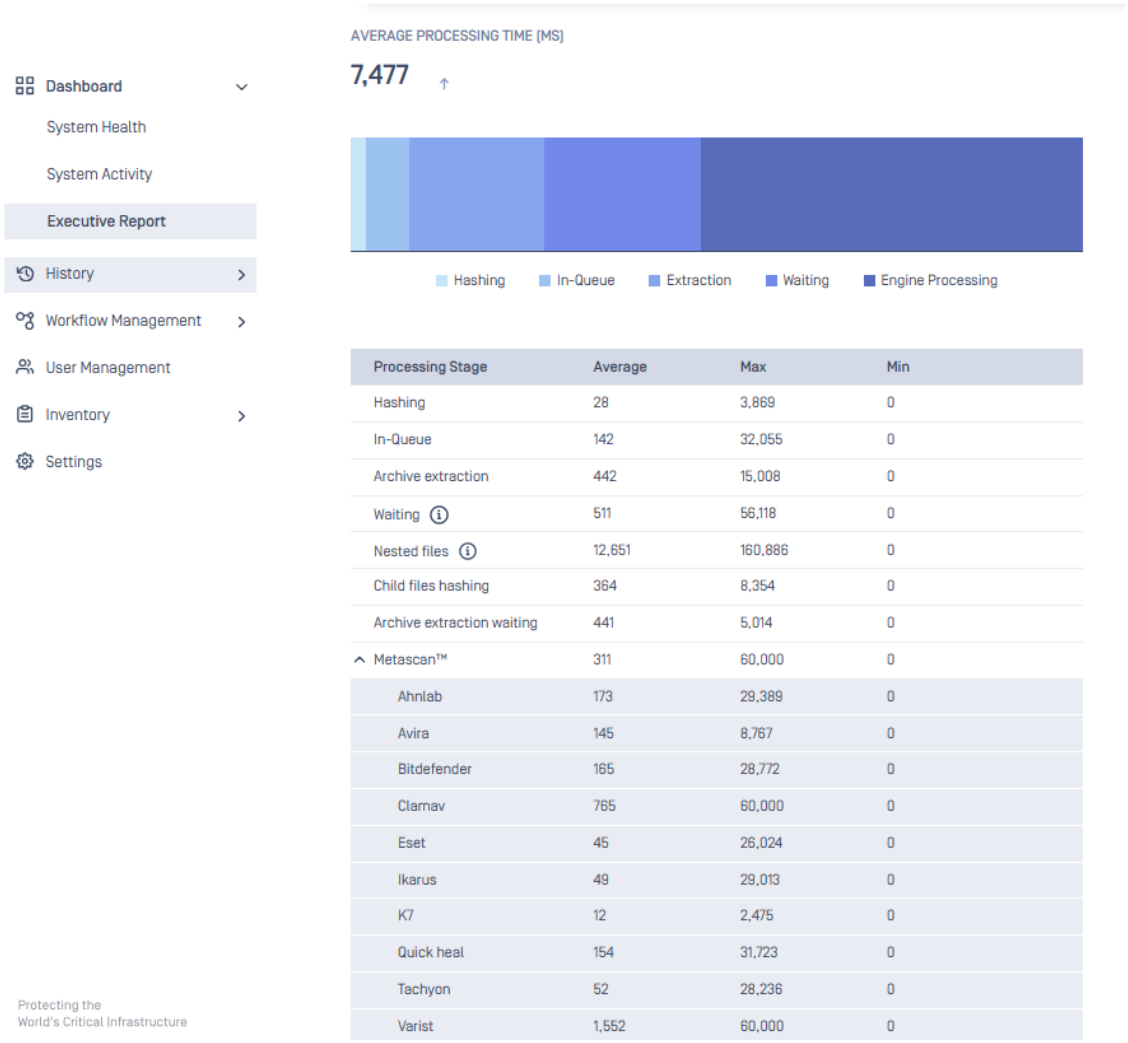


Executive Report

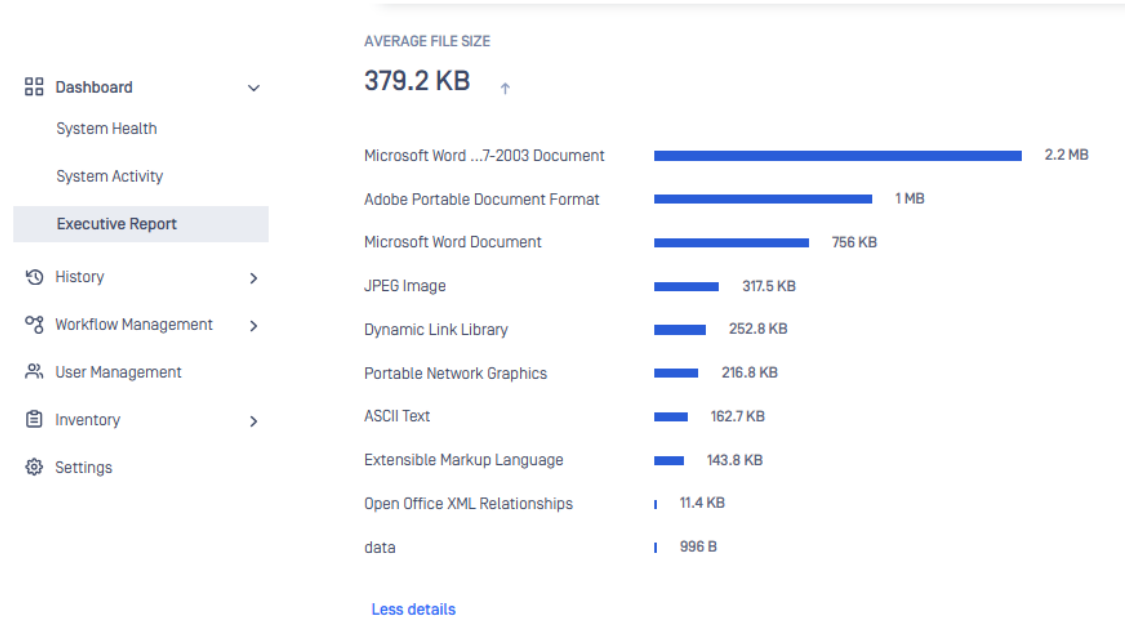
The Executive Report in MD Cluster provides statistical data on file scanning performance and metrics. This section will provide metrics and values to help you better understand your MetaDefender Cluster's performance.



Average processing time will provide detailed information regarding the Processing Stages. The Average, Min and Max will be shown for each stage.



Average file size will provide detailed information on the average file size of file types that were commonly scanned.



History

The History section contains detailed views of processing history and audit log history.

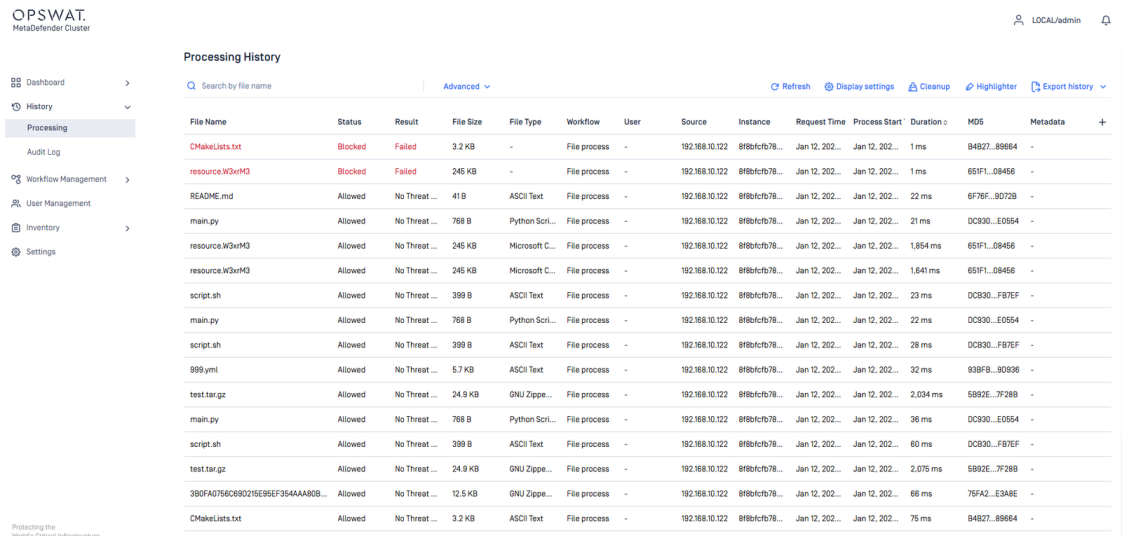
- Processing history.
- Audit log history.

Processing History

Processing history

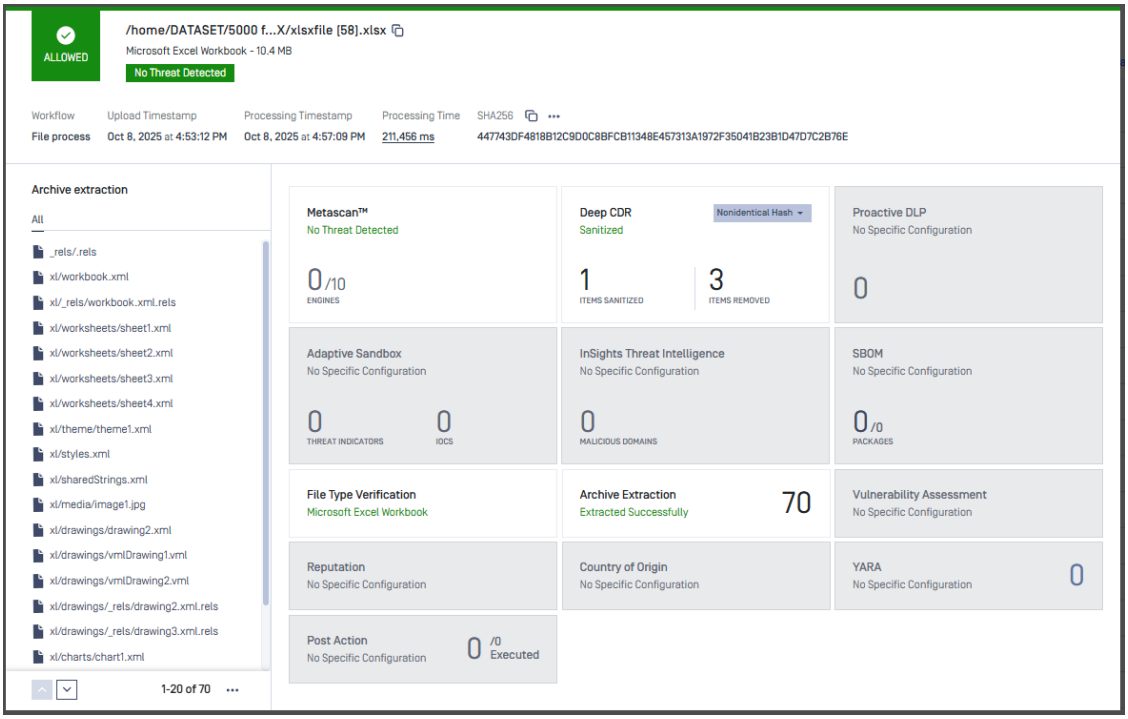
The Processing History section shows information on all scans made on MetaDefender Cluster (MD Cluster). Search and Filter are also supported against each scan result attribute. The user can search based on the following:

- MD5, SHA1, SHA256, or SHA512 hashes.
- File name (and you can limit search result for a specific scan result, and for specific username who submitted files).
- Source (IP address).
- User name.



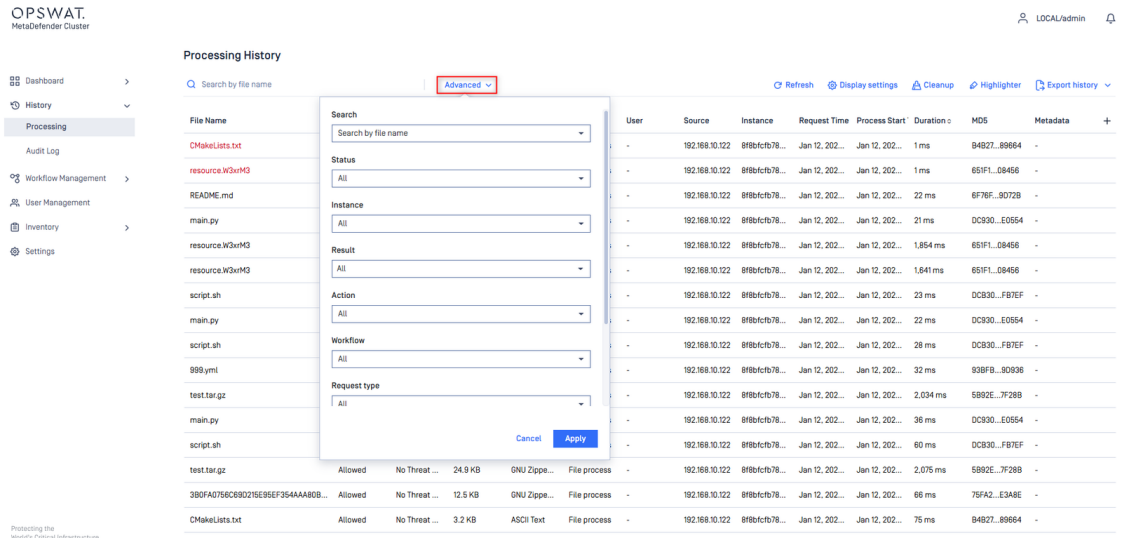
Scan result

Selecting a file in the Processing History will show the Scan Result of the file. Here the user will be able to see each of the Modules results.



Filter

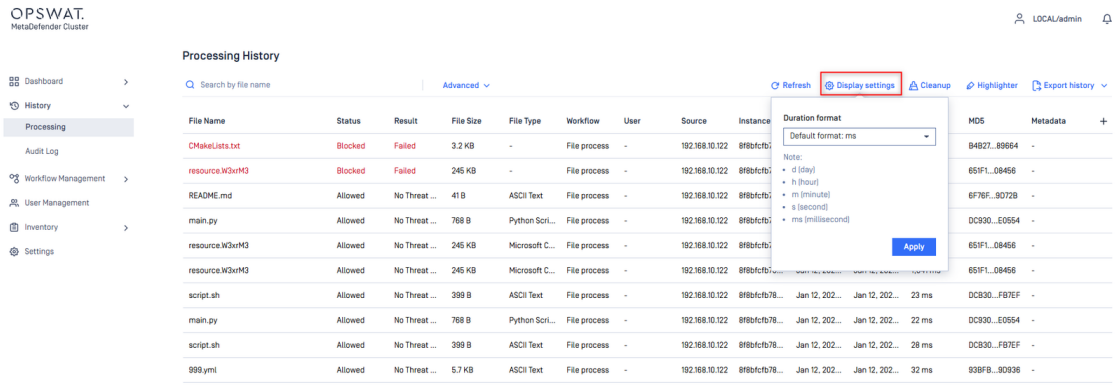
In Advanced Settings, there are multiple filtering options, such as the scan status, the instance that handled the file, the action taken on the file, the workflow used for processing the file, the date and time of the process, and more. Users can easily combine these to find their desired results.



Display settings

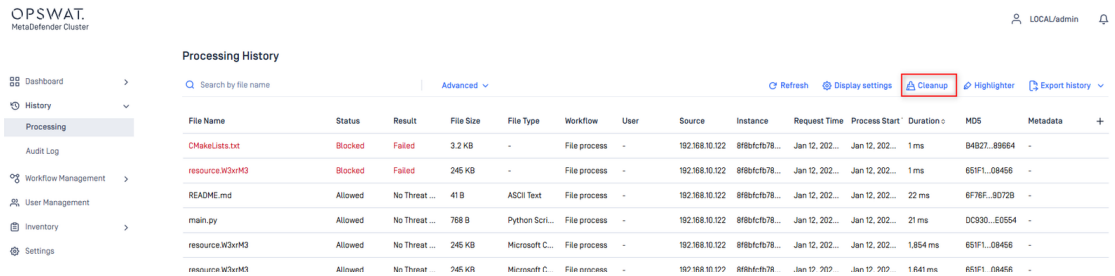
Display Settings will allow the user to change the format of the Duration column. There are 3 options to choose from:

- Default: Shown as milliseconds or ms.
- Time Format: Shown as date h:m:s.ms.
- Short Format: Shown as date h m s ms.



Cleanup

Cleanup allows the user to delete the Processing History based on Time Range.

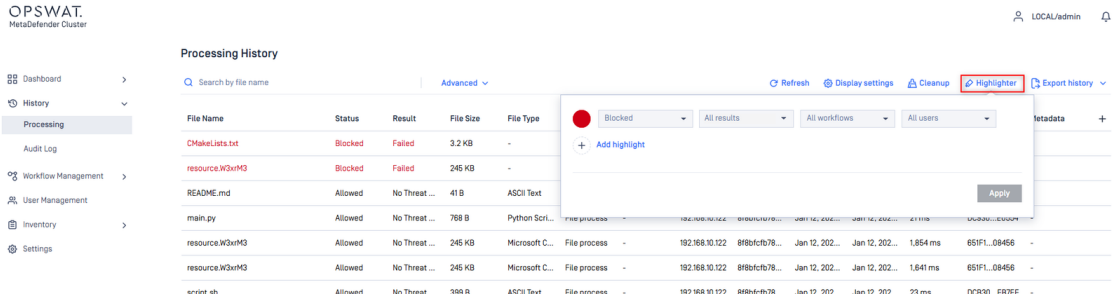


Callback and processing history retention

In some cases, a callback may not be delivered if its processing history record is removed by the retention policy before MD Cluster Callback Service processes the request. Without this record, the Callback Service cannot retrieve and send the scan result or update the callback delivery status.

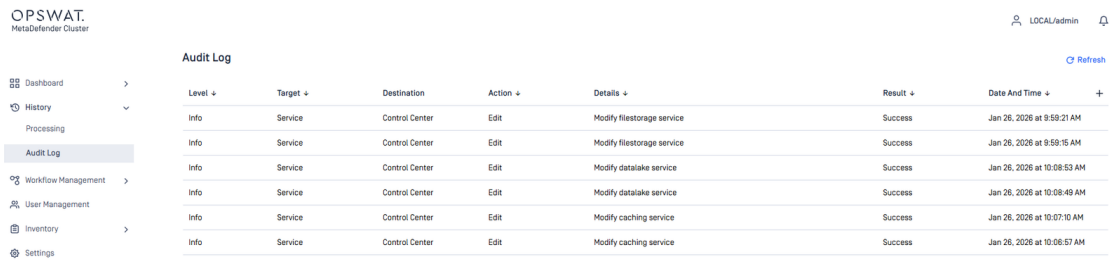
Highlight

Highlight allows the user to highlight a specific scan result based on color. This way, it will be easier to see visually for example, what is Blocked, Sanitized, Failed, etc...



Audit Log

This section provides application updates such as Deploying or Undeploying an instance, license activation, configuration change etc...



Inventory

The Inventory section contains all detailed and configurations for Services and Workers. This also incudes Module Updates, Licenses and Installers.

- Services.
- Workers.
- Modules.
- Licenses.
- Installers.

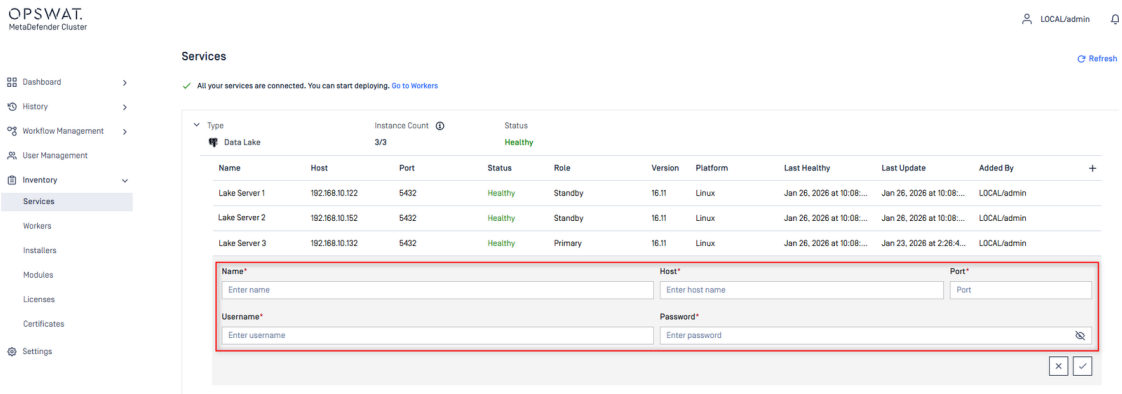
Services

This section will allow the user to add all necessary services for MetaDefender Cluster to fully function.

Data Lake and Data Warehouse

Add the Data Lake and Data Warehouse in this section. Fill out all required fields. If High Availability was configured for PostgreSQL, add them to this section as well.

> **Info**
>
> High Availability support for Postgres is only available for Data Lake



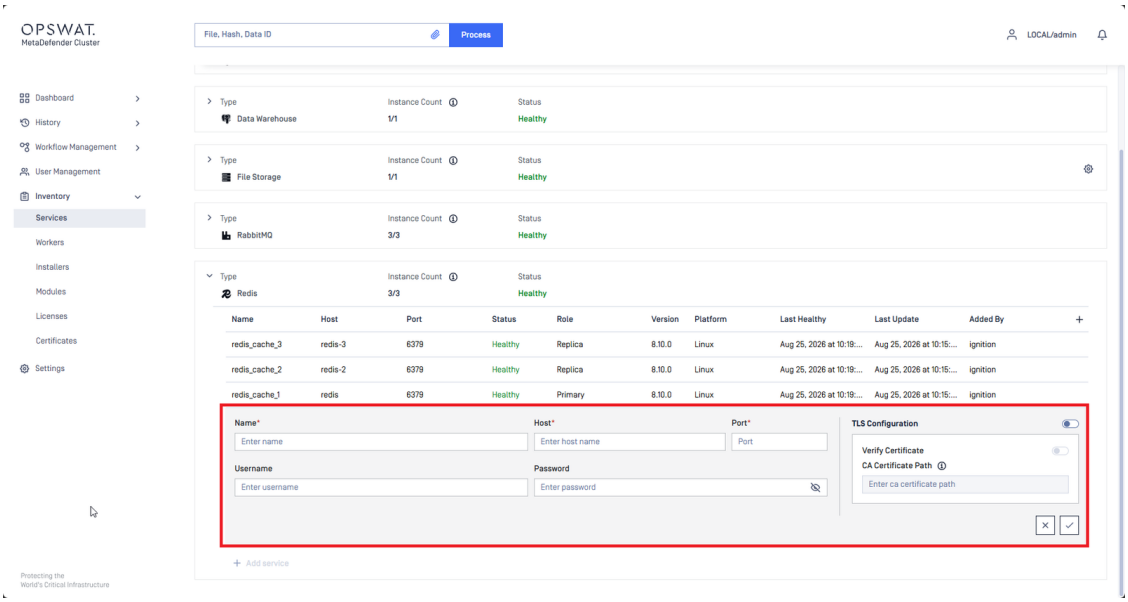
Redis

Add the Redis server in this section. Fill out all required fields.

To use an encrypted connection, enable **TLS Configuration** and set the port to the broker's TLS listener.

1. **Verify Certificate**: validates the broker's certificate. Turn it off to encrypt the traffic without verifying the peer.
2. **CA Certificate Path**: path to the CA certificate file. The file is read on the Control Center host, not on the Redis host. Leave it empty to use the operating system's trust store.

If High Availability was configured for Redis, add them to this section as well. TLS settings are per connection, so configure them on each node.



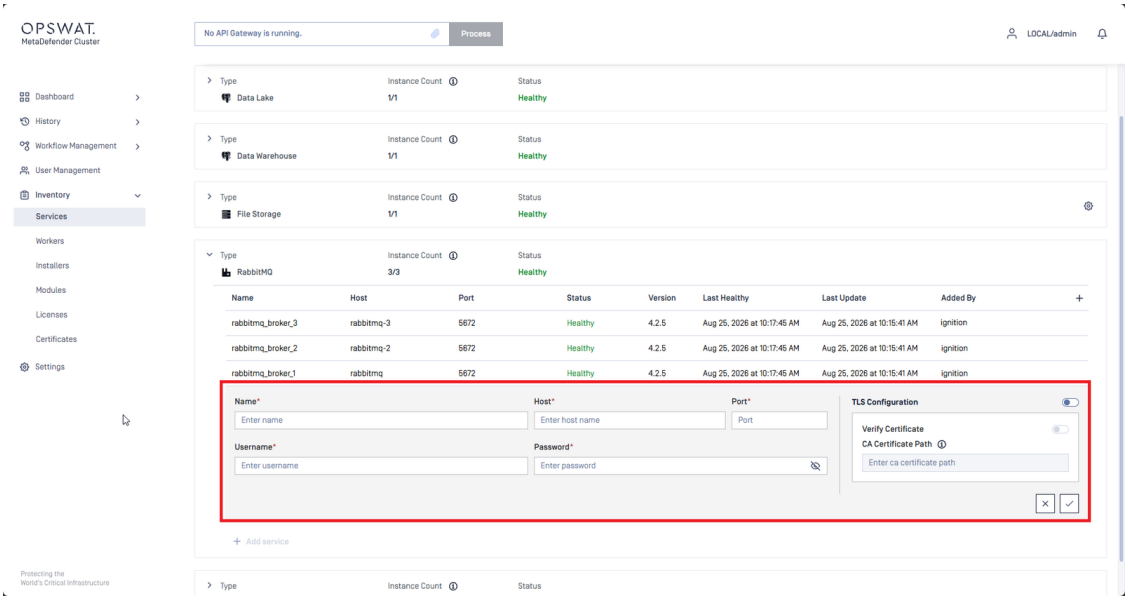
RabbitMQ

Add the RabbitMQ server in this section. Fill out all required fields.

To use an encrypted connection, enable **TLS Configuration** and set the port to the broker's TLS listener.

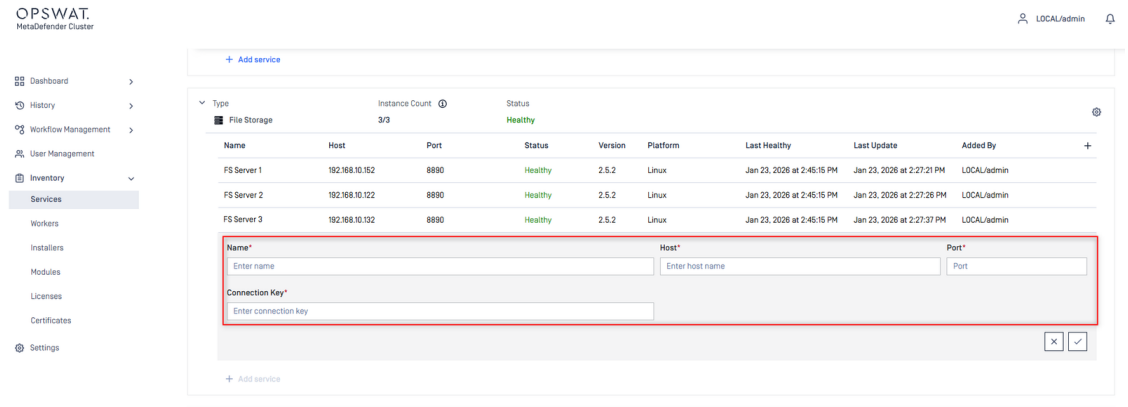
1. **Verify Certificate**: validates the broker's certificate. Turn it off to encrypt the traffic without verifying the peer.
2. **CA Certificate Path**: path to the CA certificate file. The file is read on the Control Center host, not on the RabbitMQ host. Leave it empty to use the operating system's trust store.

If High Availability was configured for RabbitMQ, add every node in this section as well. TLS settings are per connection, so configure them on each node.



File Storage

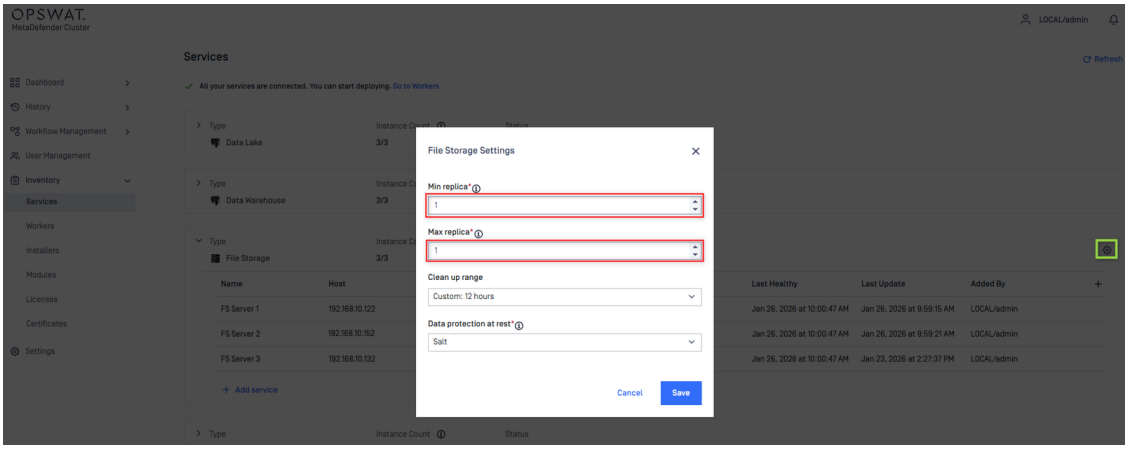
Add the File Storage server in this section. Fill out all required fields. If High Availability was configured for File Storage, add them to this section as well.



File Storage Settings

Values found in the File Storage Settings can be left in its default settings. However, when configuring High Availability for File Storage. Ensure that the Minimum replica and Maximum replica has been configured correctly. For more information on High Availability for File Storage, click here.

- Min replica: The minimum number of data copies that must be written for the operation to succeed.
- Max replica: The maximum of data copies stored across the system.
- Clean up range: The number of days, weeks or months until clean up of files that are sanitized, watermarked files and files processed by Proactive DLP.
- Data protection at rest: The protection mechanism is applied to the files stored in File Storage. By default, all files are salted before being they are saved to disk.



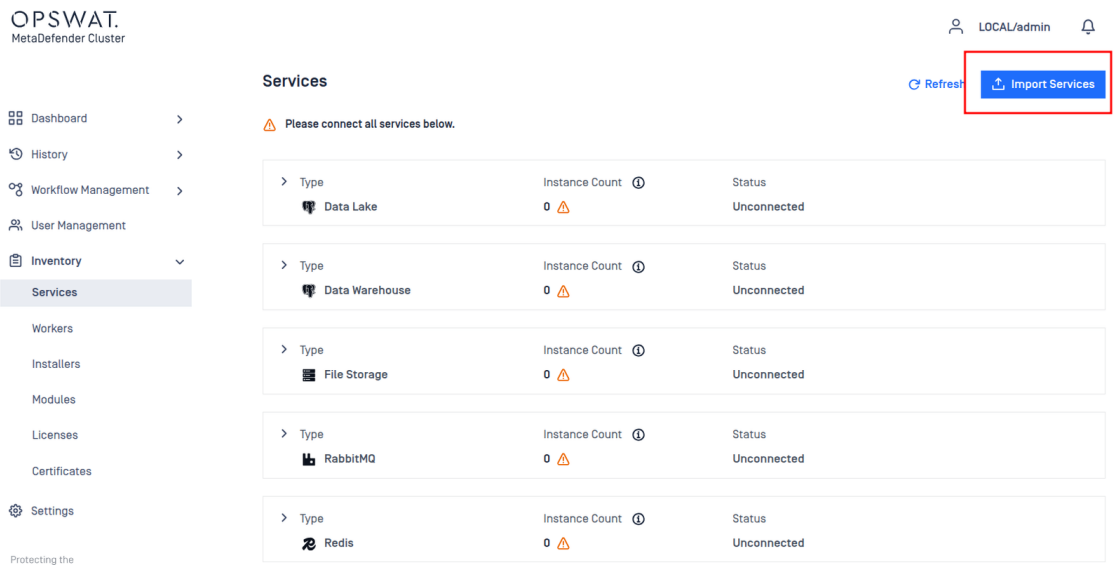
Import Services

The **Import Services** feature allows you to import connections for multiple service types—including **Data Lake**, **Data Warehouse**, **Redis**, **RabbitMQ**, and **File Storage**—from a single YAML file. This provides a faster alternative to creating each connection manually.

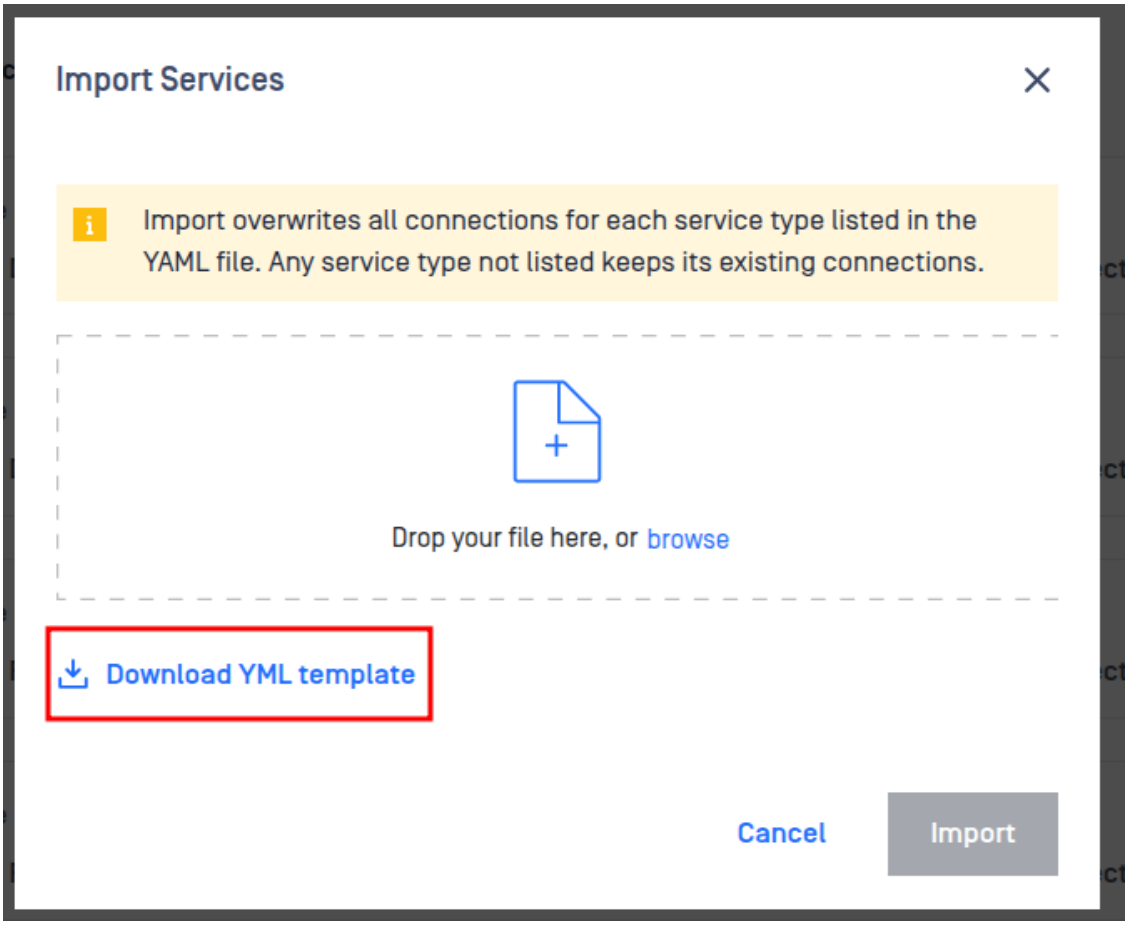
> **Info**
>
> *Importing services replaces all existing connections for each service type included in the YAML file. Connections for service types that are not included in the file remain unchanged*

To import services:

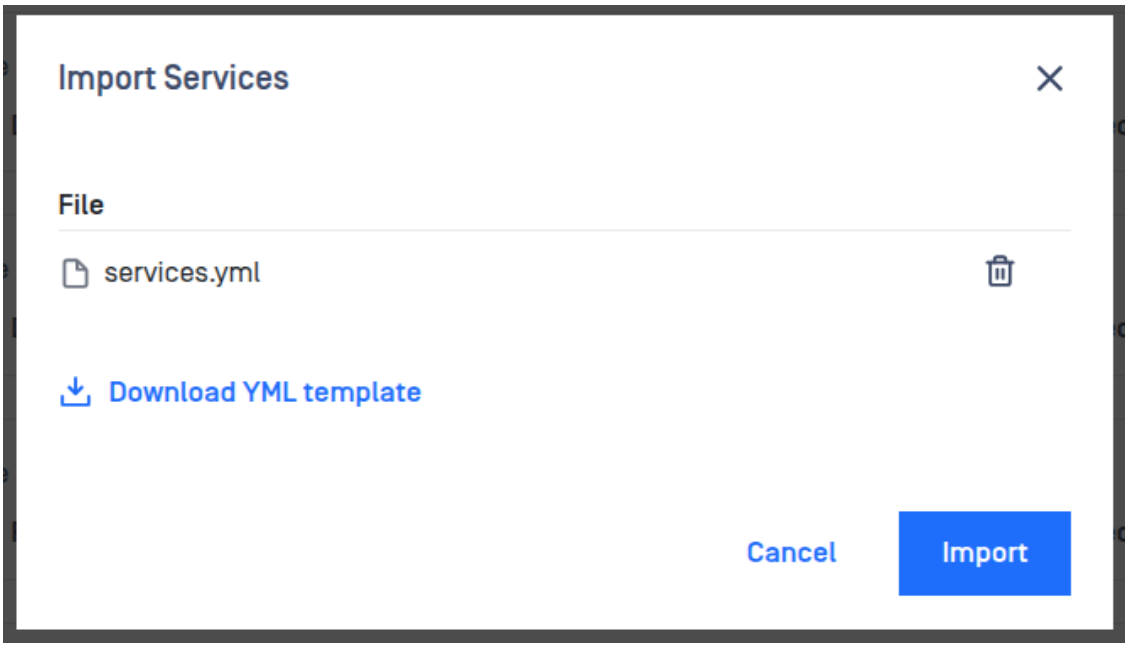
1. On the **Services** page, click **Import Services** in the upper-right corner.



2. In the **Import Services** dialog, click **Download YAML Template** to download a sample file. Update the file with the connection details for the service types you want to import, and remove any service type sections that you do not want to import.



3. Drag the completed YAML file into the upload area, or click **Browse** to select the file from your computer.



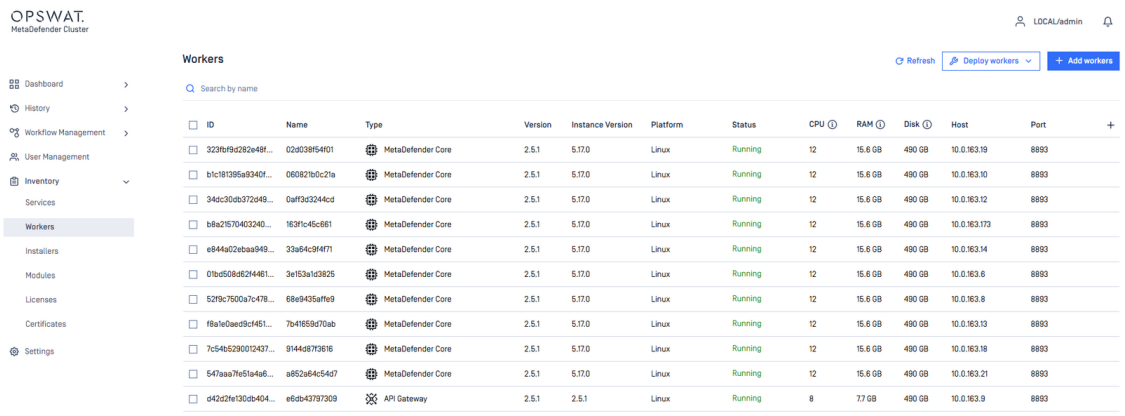
4. Click **Import** to apply the new service connections.

Workers

This section will allow the system administrator to add the MetaDefender Cluster (MD Cluster) Workers which will help deploy and monitor activities of MetaDefender Core, MD Cluster API Gateway and MD Cluster Callback Service.

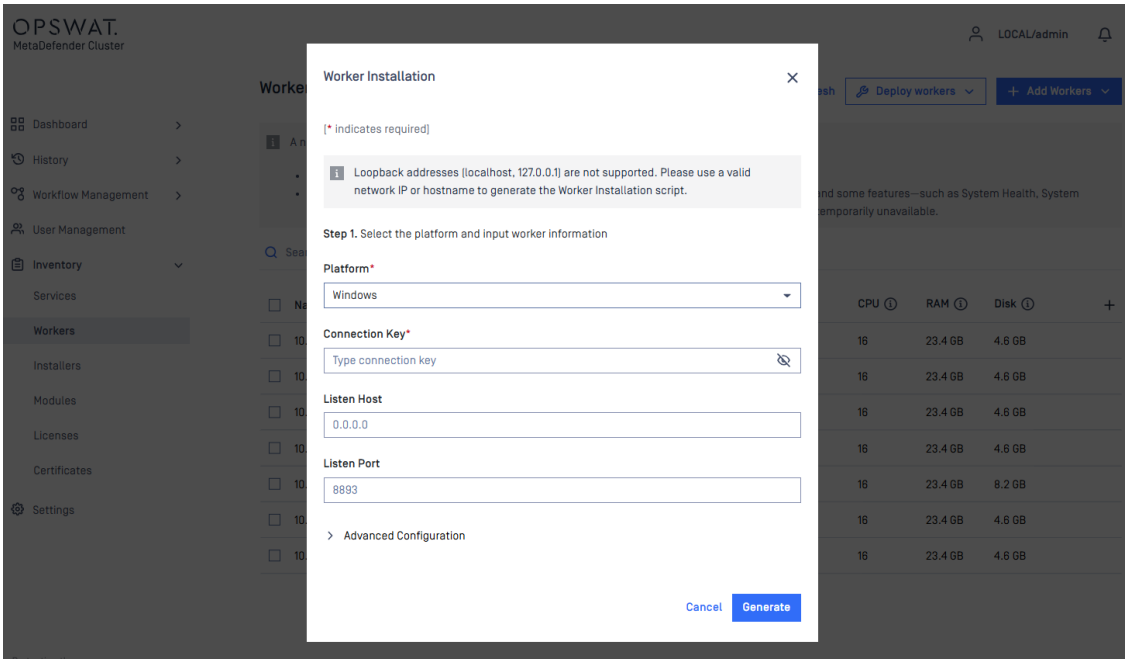
Info

MD Cluster Callback Service is optional. However, if the scan result needs to be sent to a Webhook. This service must be installed in order to use the Callback Service feature.



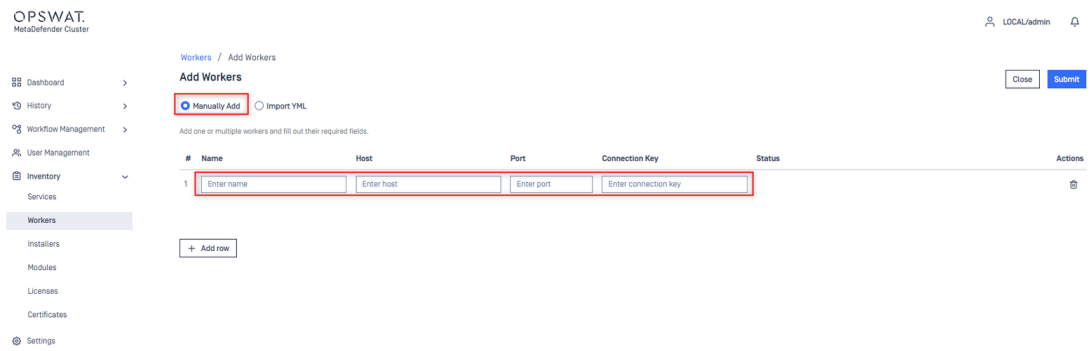
Automatically Deploy Workers

MD Cluster Workers can be automatically installed and enrolled using deployment scripts generated from MD Control Center. Administrators can follow the instructions to retrieve the installation script and then run the installation script on target hosts to deploy Workers with the required configuration and cluster enrollment settings. This automated approach simplifies large-scale deployments, reduces manual effort, and helps ensure consistent Worker configuration across all managed hosts.



Manually Add Workers

To manually add an MD Cluster Worker, enter the **Name**, **Host**, **Port**, and **Connection Key**. The **Port** and **Connection Key** must match the values configured in the Worker during installation. After completing the fields, select **Submit**. MD Cluster will validate the connection and indicate whether the Worker was added successfully.



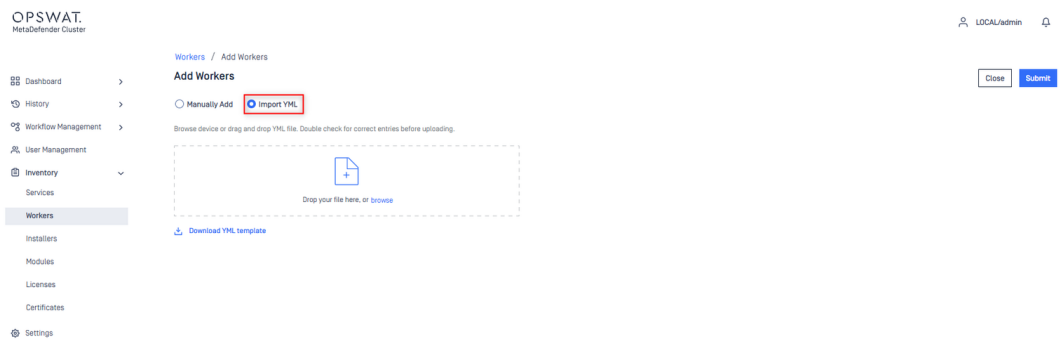
> **Info**
>
> Manual addition is intended for Workers that are already installed but not currently connected to MD Control Center. For new deployments, use the automated deployment process whenever possible.

Bulk Import Workers

System administrators can import MD Cluster Workers using a YAML file instead of adding them individually through the MD Cluster Control Center interface. To simplify the process, a downloadable template is available when selecting the **Import YAML** option. The following example demonstrates a YAML configuration that includes an MD Cluster API Gateway and three MetaDefender Core instances:

Code · yaml

```yaml
- display_name: API-Gateway
  host: 10.1.100.100
  port: 8893
  connection_key: 1234abcd
- display_name: Core-1
  host: 10.1.100.101
  port: 8893
  connection_key: 1234abcd
- display_name: Core-2
  host: 10.1.100.103
  port: 8893
  connection_key: 1234abcd
- display_name: Core-3
  host: 10.1.100.104
  port: 8893
  connection_key: 1234abcd
```



Info

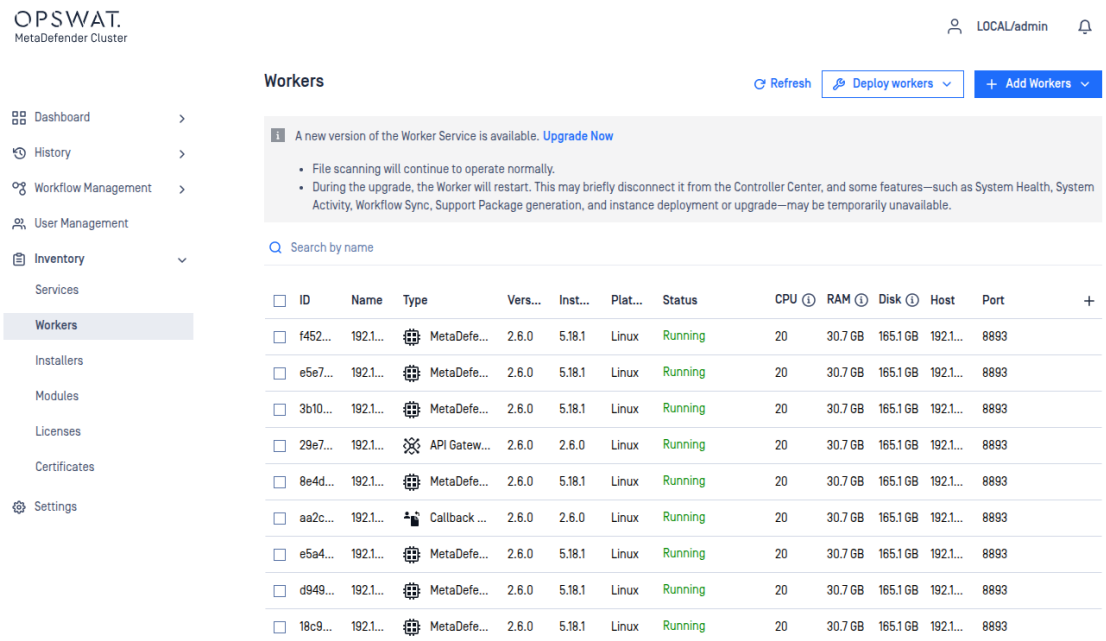
Importing a list of Workers is intended for Workers that are already installed but not currently registered with MD Control Center. This method provides a convenient way to add multiple existing Workers at once. For new deployments, use the automated deployment process whenever possible.

Upgrade Workers

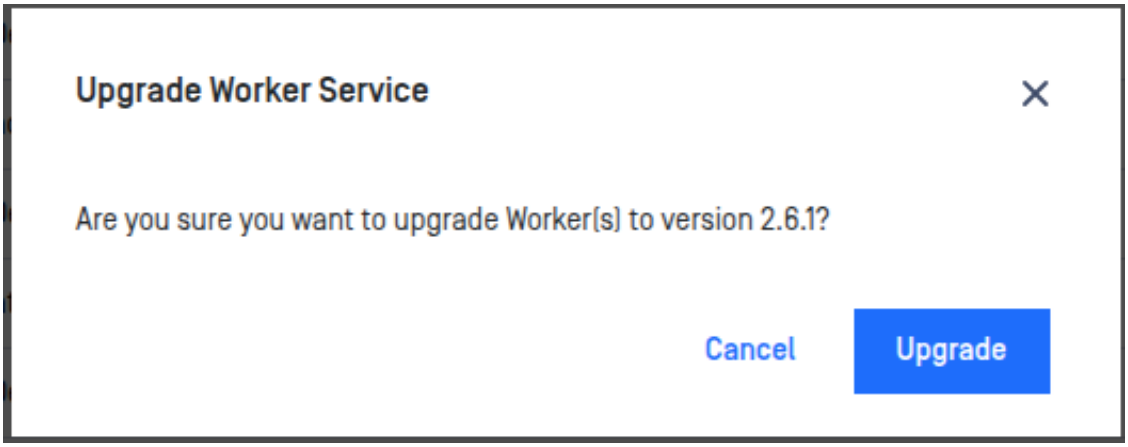
MD Cluster Worker is included with MD Cluster Control Center. Whenever the Control Center is upgraded to a newer version, a corresponding version of MD Cluster Worker also becomes available for centralized upgrade directly from the Control Center console.

To upgrade Workers:

1. Navigate to `Inventory`> `Workers`.
2. Look for the notification banner displayed on the page indicating that a Worker upgrade is available.
3. Click `Upgrade Now`.

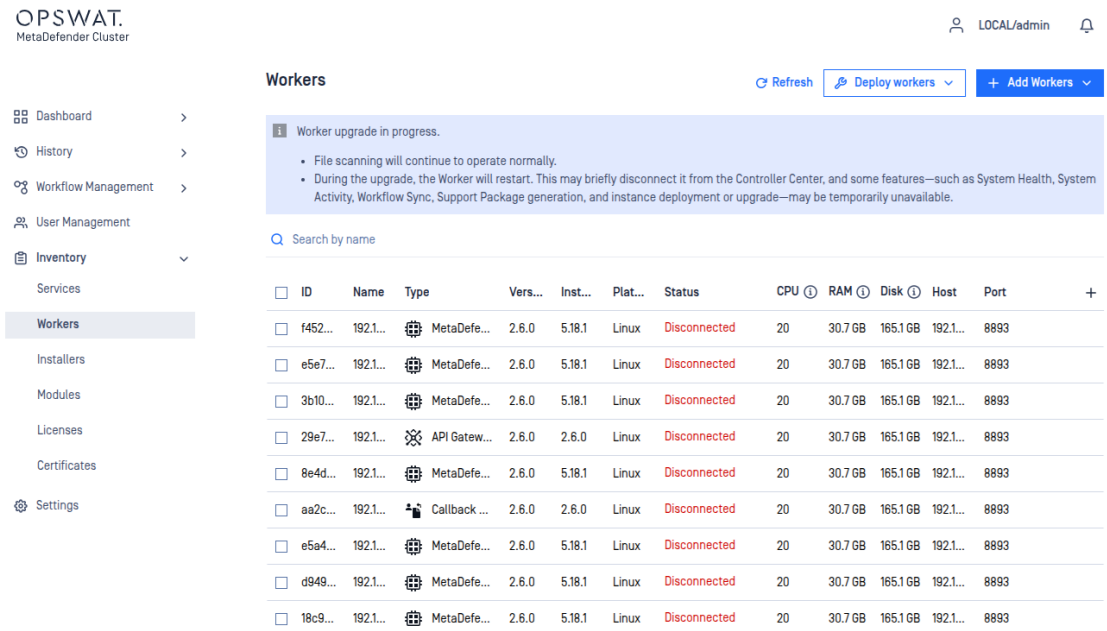


4. Confirm the action to begin the upgrade.



During the upgrade process:

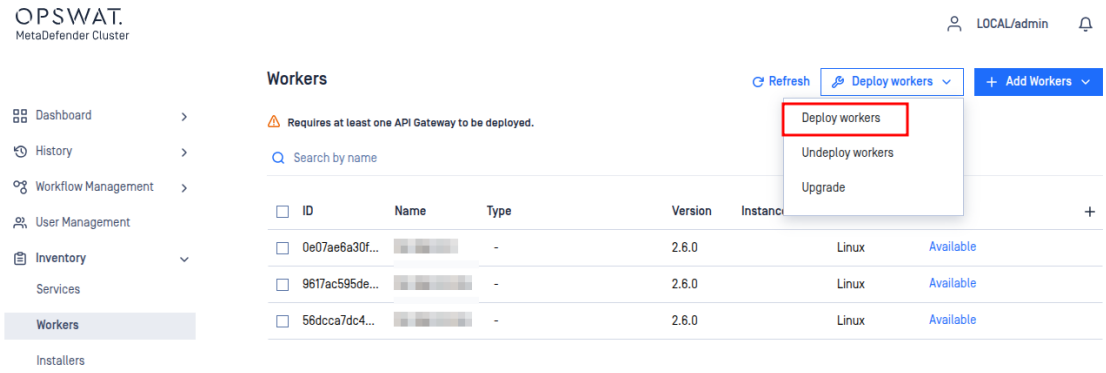
- The banner displays the message: **"Worker upgrade in progress."**
- Worker instances may temporarily show a **Disconnected** status while restarting.
- File scanning operations continue normally during the upgrade.



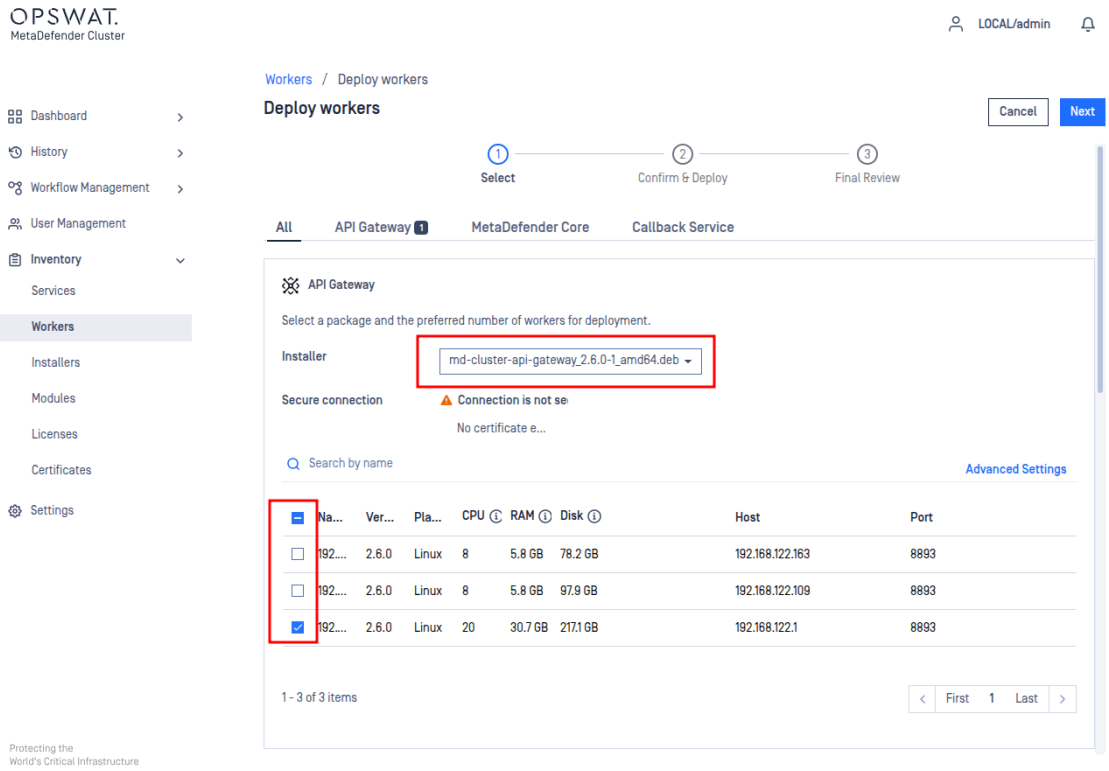
After the upgrade completes successfully, all Workers display the new version number (for example, **2.6.1**) in the **Version** column.

Deploy Instances

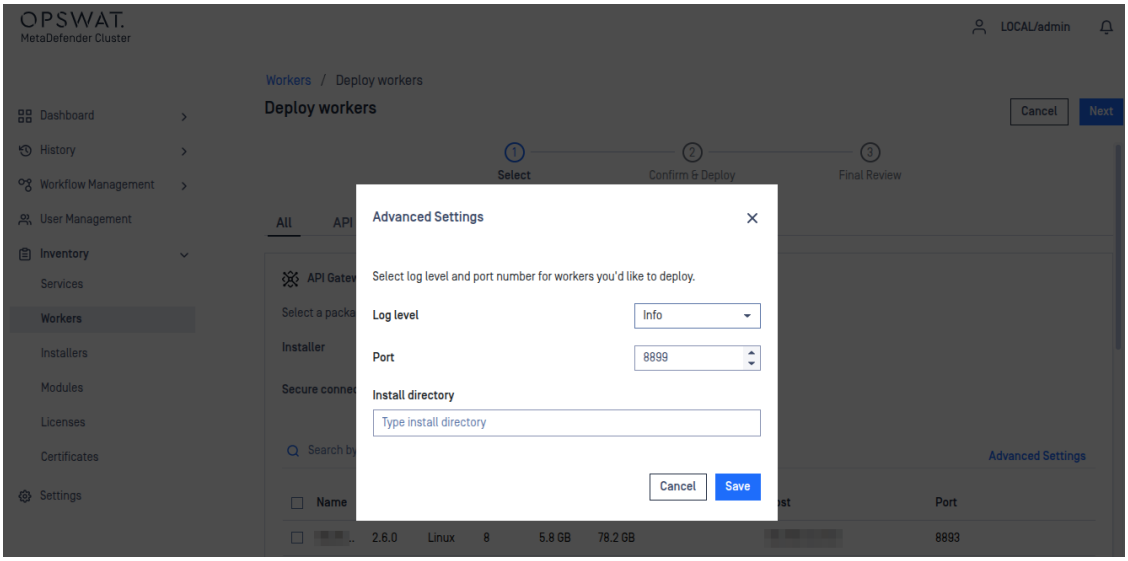
Once MD Cluster Workers have been added and their status are shown as `Available`. The system administrator can now deploy MetaDefender Core, MD Cluster API Gateway, MD Cluster Callback Service, and Download Service separately or all at the same time.



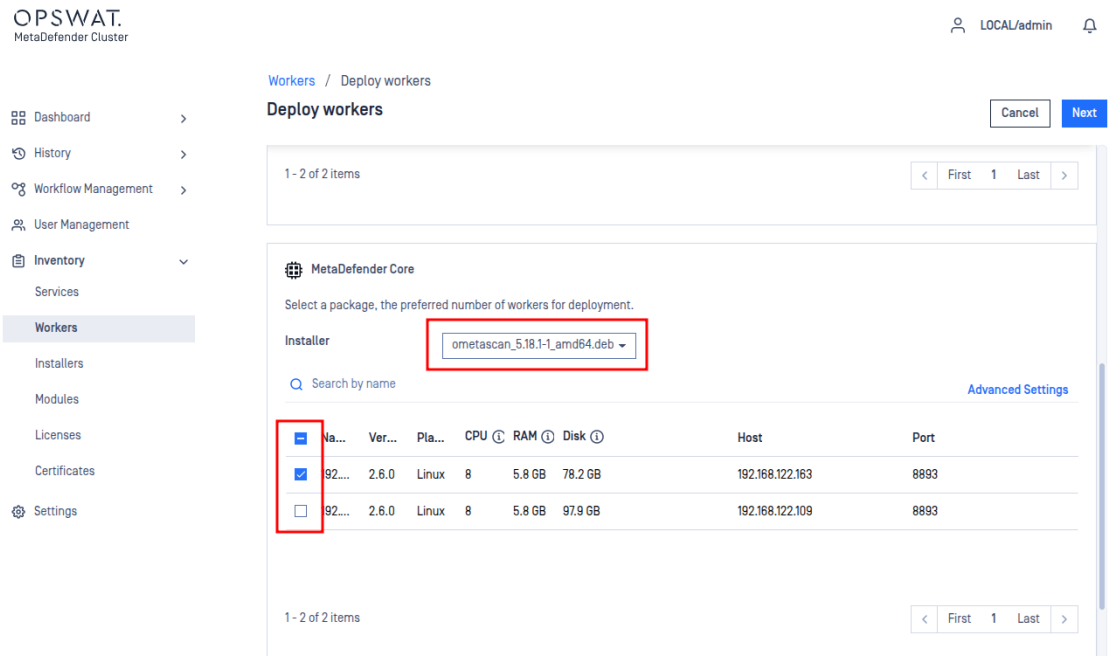
To deploy the MD Cluster API Gateway. Simply choose the version of MD Cluster API Gateway and select an available MD Cluster Worker.



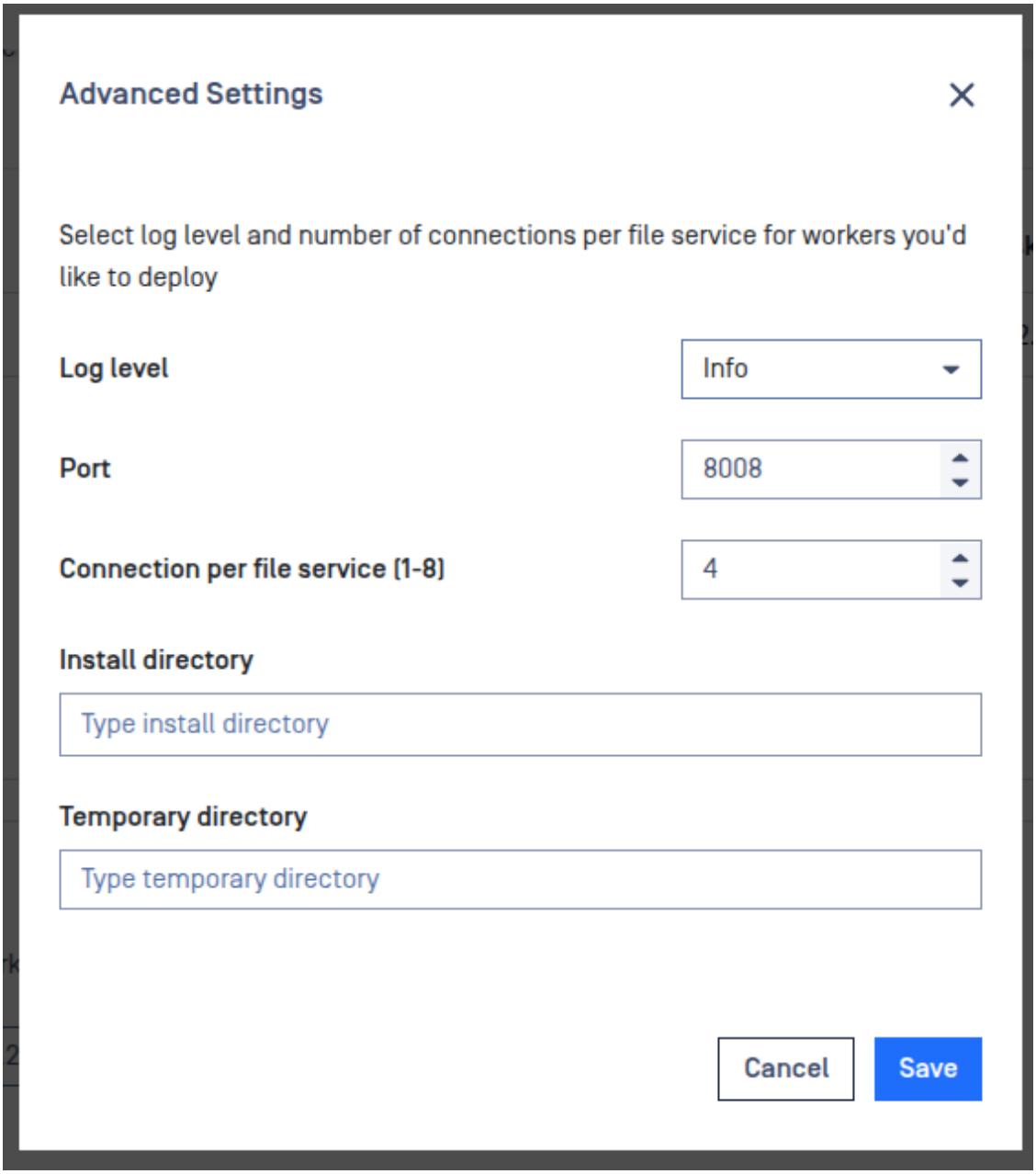
The Log level, Port, and custom Install directory that will be assigned to MD Cluster API Gateway can be set by clicking `Advanced Settings`.



The system administrator can choose the version of MetaDefender Core should be deployed on available MD Cluster Workers.



The system administrator can also modify the Log level, Port, custom Install directory, custom temporary directory, and Connection per file service of MetaDefender Core that will be deployed by selecting `Advanced Settings`.



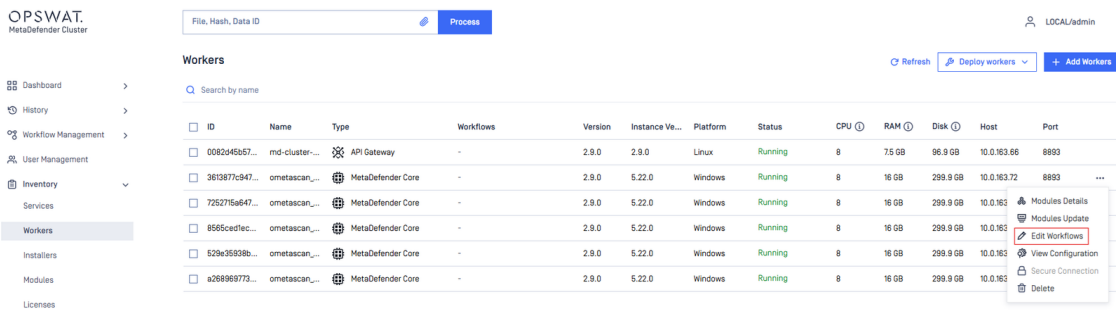
- `Installation directory`: Existing absolute path on the MD Cluster Worker host where this instance will be installed. Leave empty to use the default path.
- `Temporary directory`: Existing writable absolute path on the MD Cluster Worker host for MD Core temporary scan files. On Linux, the path can be up to 67 characters. Leave this field empty to use the default path. You can also use a RAM disk.

Click `Next` to confirm the deployment and `Finish` to start deploying on selected MD Cluster Workers.

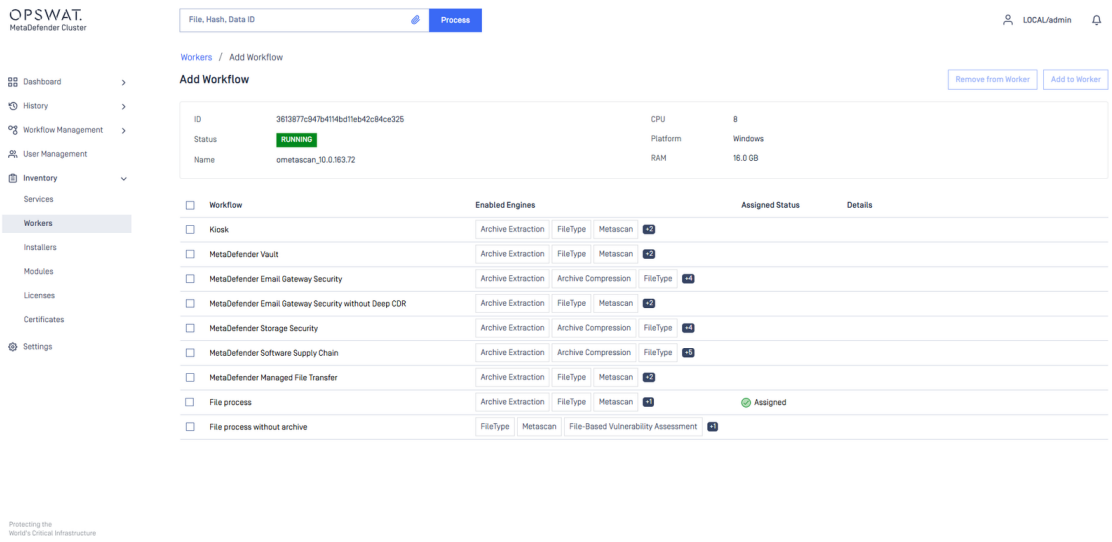
Assign Workflows to Instances

By default, a deployed MetaDefender Core instance processes files for every workflow. From the **Workers** page, you can restrict an instance to specific workflows, which is the same setting as **Assigned Instances** on the workflow itself. The **Workflows** column shows the workflows each instance is currently assigned to.

1. In the MetaDefender Core row, click the three-dot menu and select **Edit Workflows**..



2. Check the instance details at the top of the page (ID, name, status, CPU, RAM, and platform) to confirm you are editing the right instance, then select the workflows to change.
3. Click **Add to Worker** to assign the selected workflows, or **Remove from Worker** to unassign them. The **Assigned Status** column shows the result for each workflow.

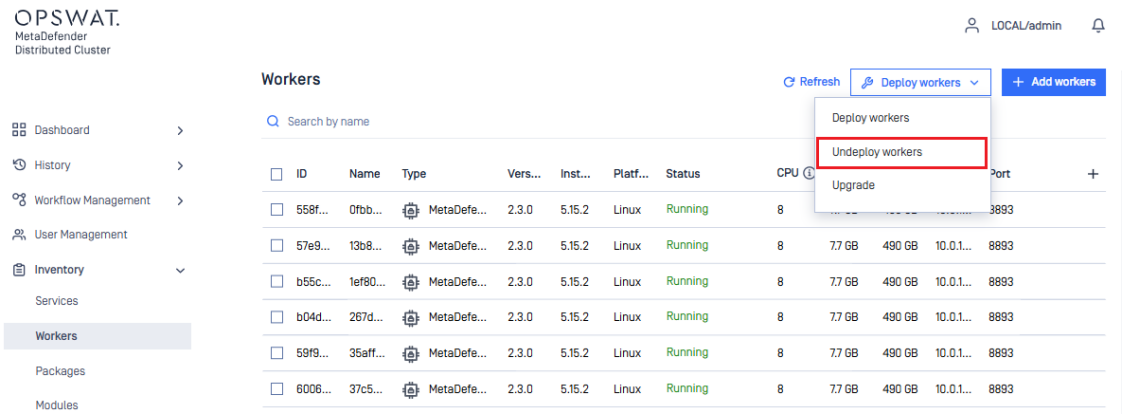


> Only MetaDefender Core instances can be assigned to workflows, so **Edit Workflows** is not offered for the API Gateway or the Callback Service. Assignment requires MetaDefender Core 5.22.0 or later.

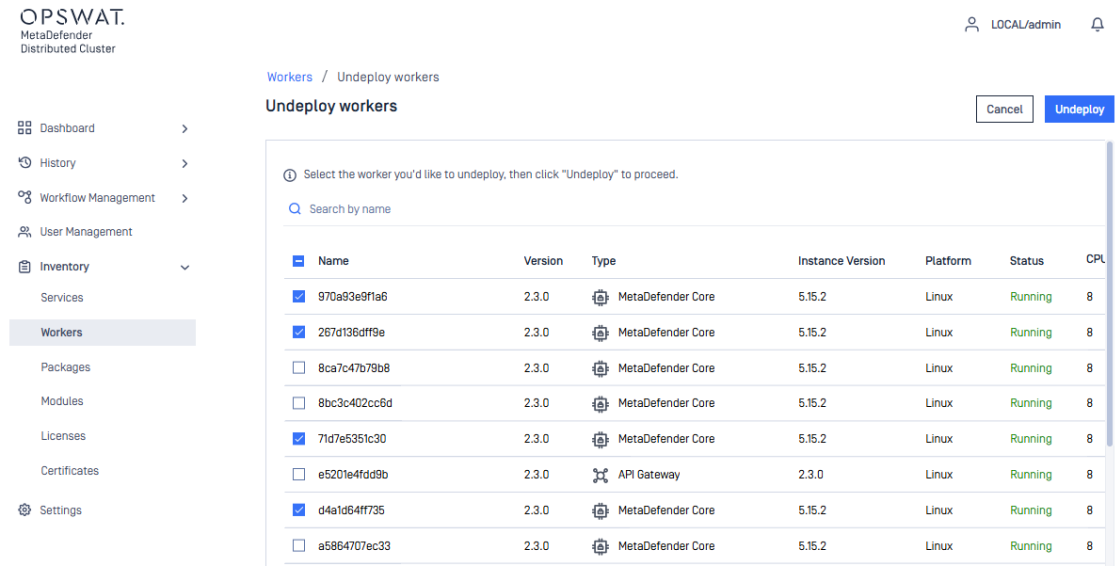
> A MetaDefender Cluster Worker cannot be undeployed or deleted while workflows are assigned to it. Remove all workflow assignments first.

Release Instances

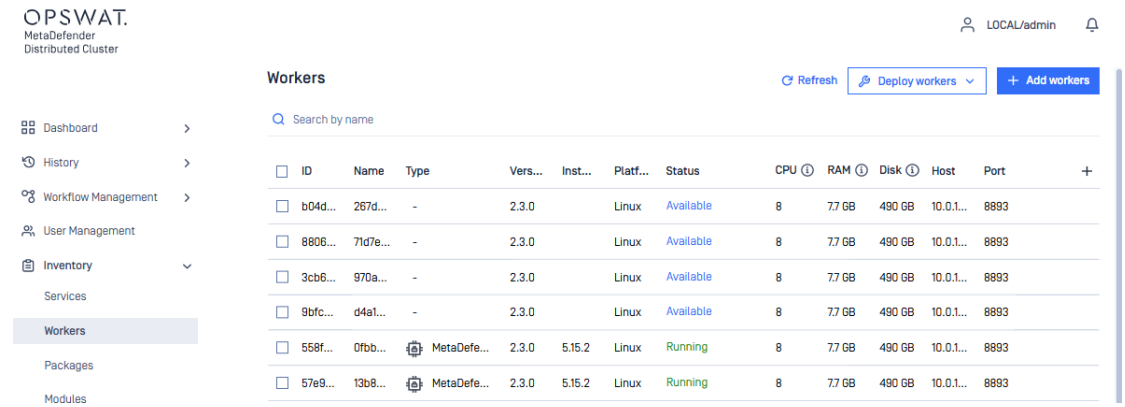
To release MD Cluster API Gateway, MD Cluster Callback Service, or MetaDefender Core on MD Cluster Workers, the system administrator can select `Undeploy` workers in the top right corner.



From the list of hosting MD Cluster Worker instances, the system administrator can choose instances to release MD Cluster API Gateway, MD Cluster Callback Service or MetaDefender Core.

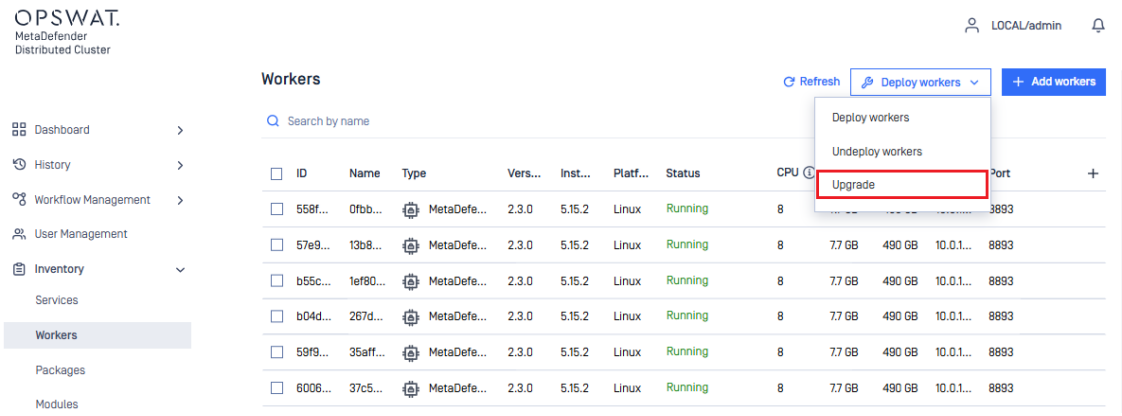


When the `Undeploy` button is selected. MD Cluster API Gateway, MD Cluster Callback Service, and/or MetaDefender Core on selected MD Cluster Worker instances are uninstalled. Once uninstalled, the MD Cluster Worker instances will become available to deploy new MD Cluster API Gateway, MD Cluster Callback Service or MetaDefender Core.

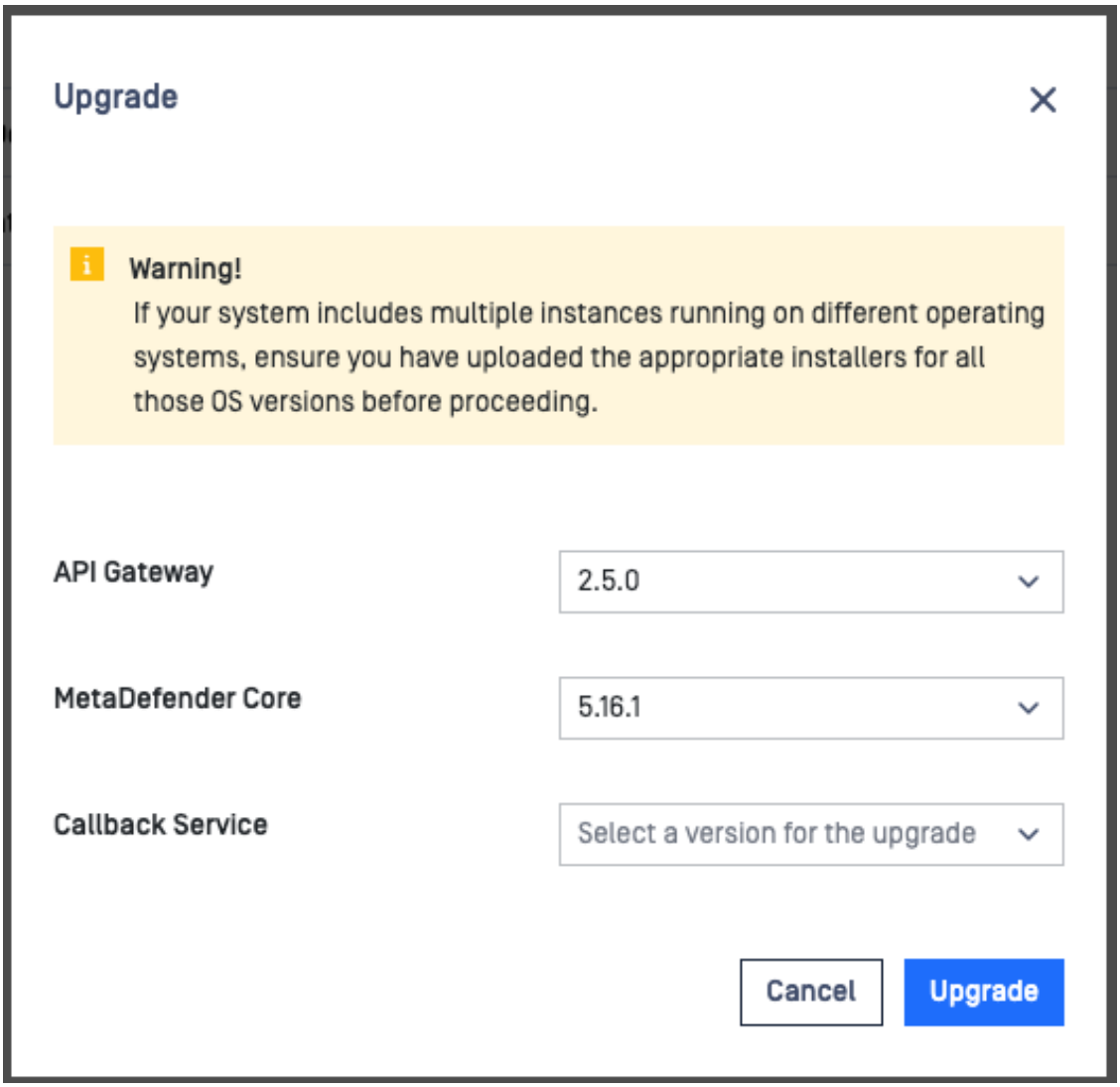


Upgrade Instances

When a new version of MD Cluster API Gateway, MD Cluster Callback Service or MetaDefender Core is available. And all three installers have been uploaded to the Installers section. The system administrator can perform an upgrade for the three products by selecting the `Deploy Workers` menu and then `Upgrade`.



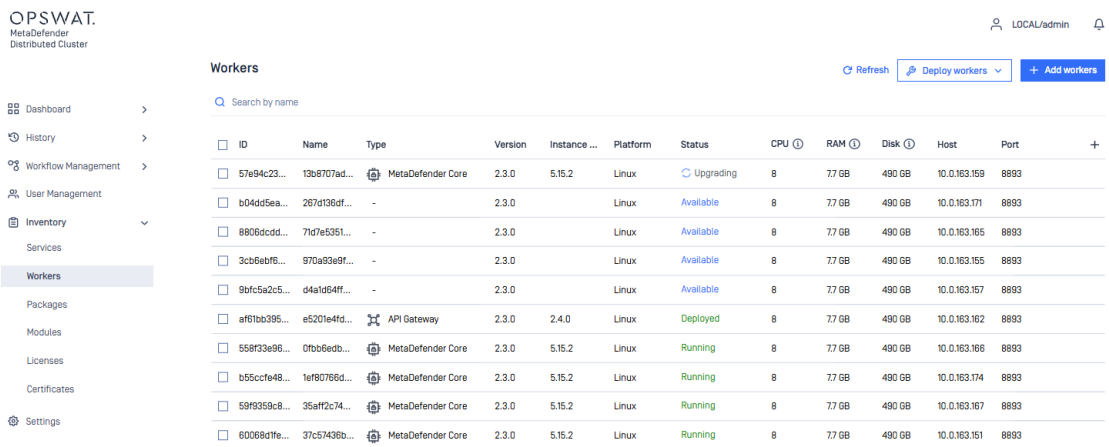
The system administrator can then choose the new version of MD Cluster API Gateway, MD Cluster Callback Service and MetaDefender Core to perform the upgrade.



The new versions of MD Cluster API Gateway, MD Cluster Callback Service, or MetaDefender Core are streamed to the MD Cluster Workers and remote upgrade process will takes place automatically.

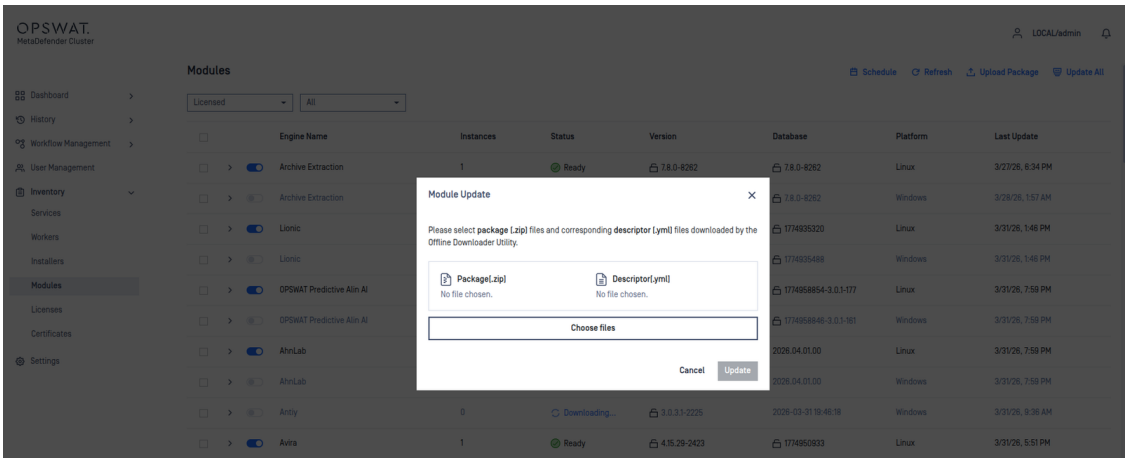
Info

If there are scans currently active. MetaDefender Core will isolate itself by no longer taking scan request and finishes its current scans before upgrading. Upgrades are performed one MetaDefender Core at a time so that scans are uninterrupted.



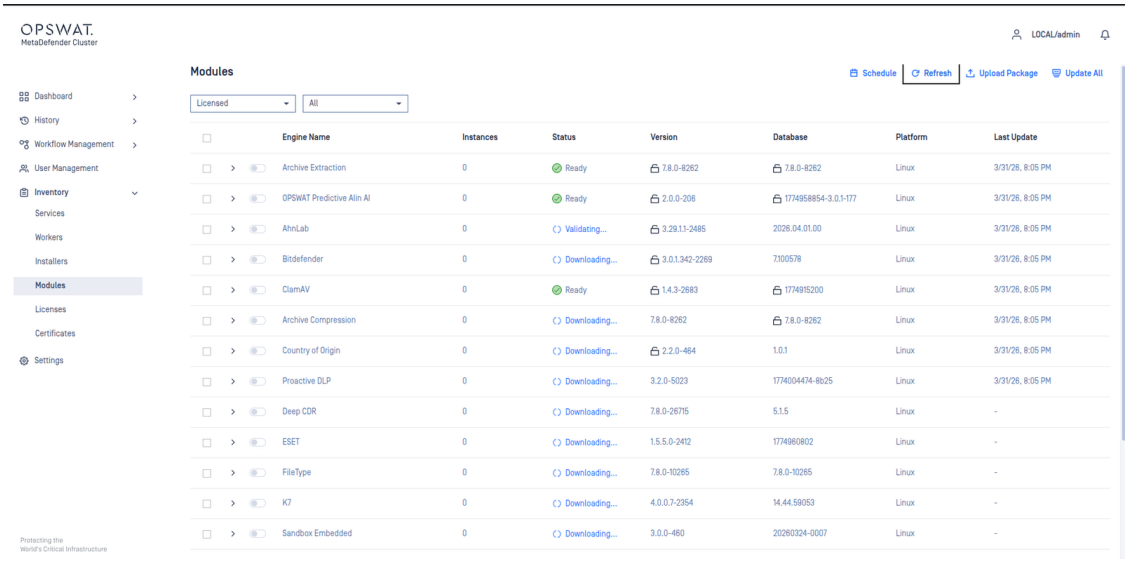
Modules

The Modules section provides visibility to the modules that are available to MetaDefender Core to use when deployed. Modules are downloaded by MetaDefender Cluster (MD Cluster) Control Center or if deploying in an Offline Environment, Modules can be uploaded manually by the system administrator when using MetaDefender Update Downloader.



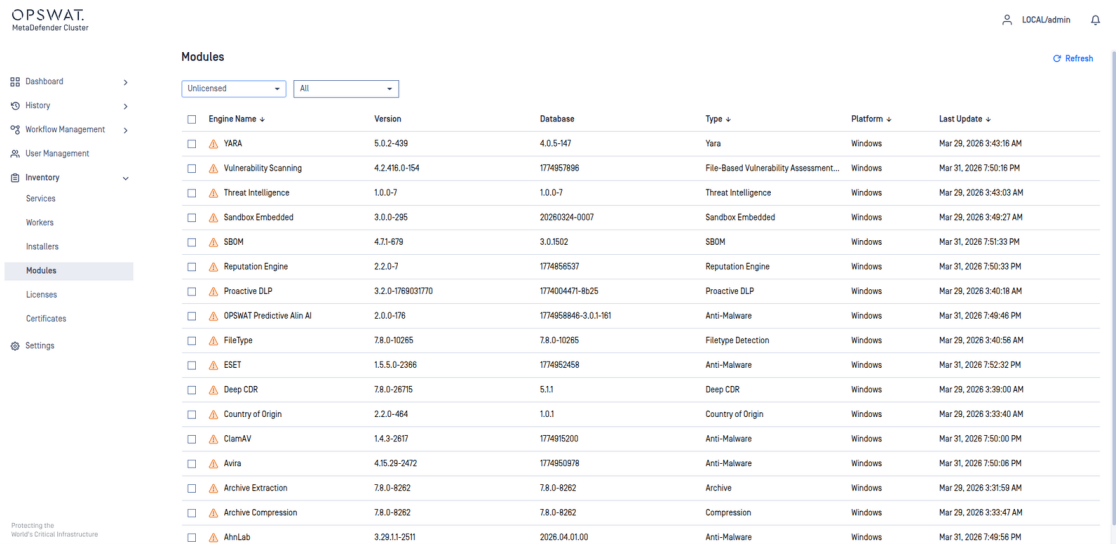
Licensed

This section of the Modules will show all the modules that are available and will be used when MetaDefender Core is deployed. The modules shown in this section are based on the MetaDefender Core license. Modules are automatically downloaded if the MD Cluster Control Center has access to the internet.



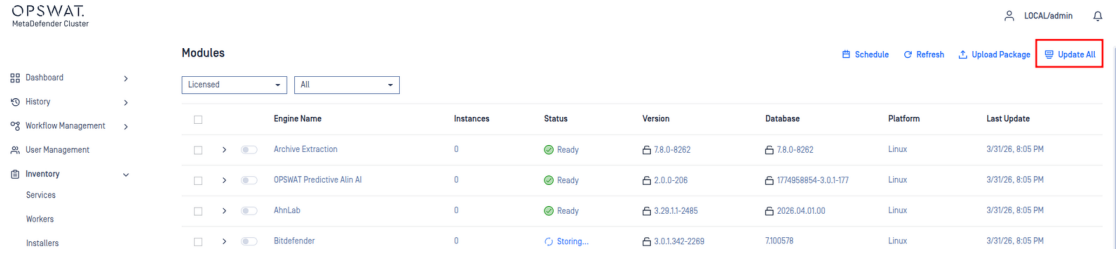
Unlicensed

This section of Modules will show all the modules that were previously licensed. From here, since the modules are no longer licensed they can be removed by selecting all the modules and clicking `Delete`.



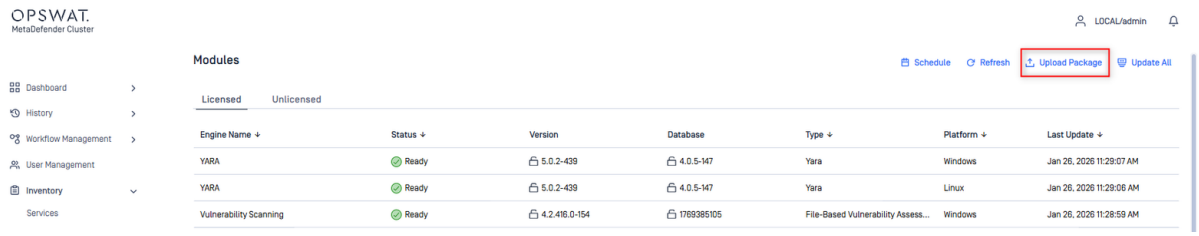
Manual Updates

The system administrator can trigger an update by simply clicking the `Update All` button.

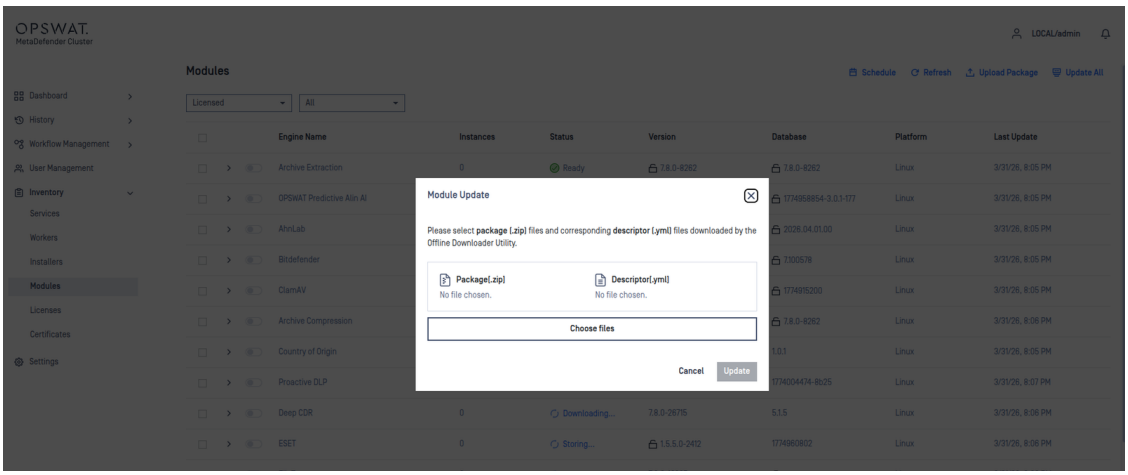


If MD Cluster Control Center does not have internet access, the system administrator will need to use **MetaDefender Update Downloader** to obtain the required modules. After downloading, these modules can be manually uploaded to

MD Cluster Control Center using the `Upload Packages` option.



Choose the package files and proceed with the upload.



Module Update Configuration

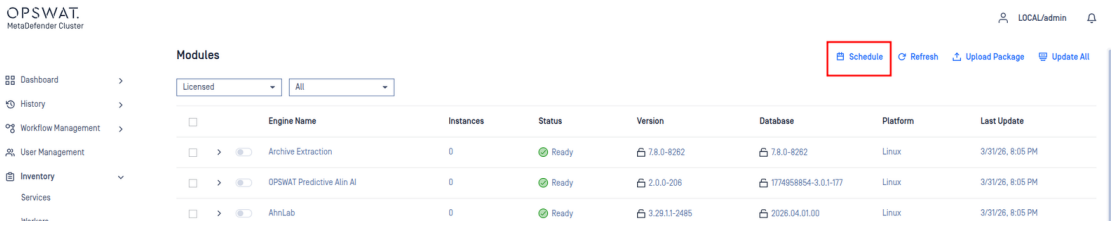
The system administrator can select an update mechanism for the Modules.

- **Online:** Automatic Updates which will download modules from the internet.
- **Local Folder:** MD Cluster Control Center will pick up Module updates from a specific folder.
- **Offline:** Disable Automatic Updates.

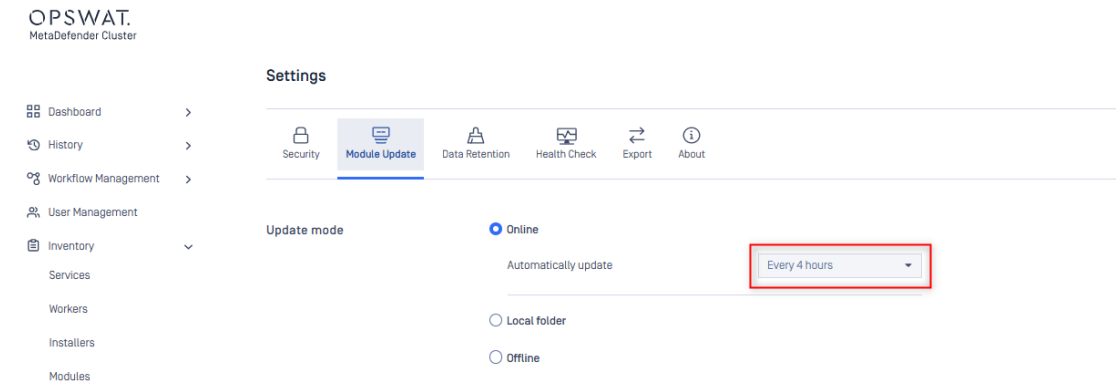
Online

Choosing the Online method will allow MD Cluster Control Center to perform an automatic update by downloading the modules directly from the internet.

> **Info**
>
> MD Cluster Control Center needs access to https://update.dl.opswat.com in order to receive module updates.



MD Cluster Control Center will periodically check the latest version of modules every 4 hours.



Warning

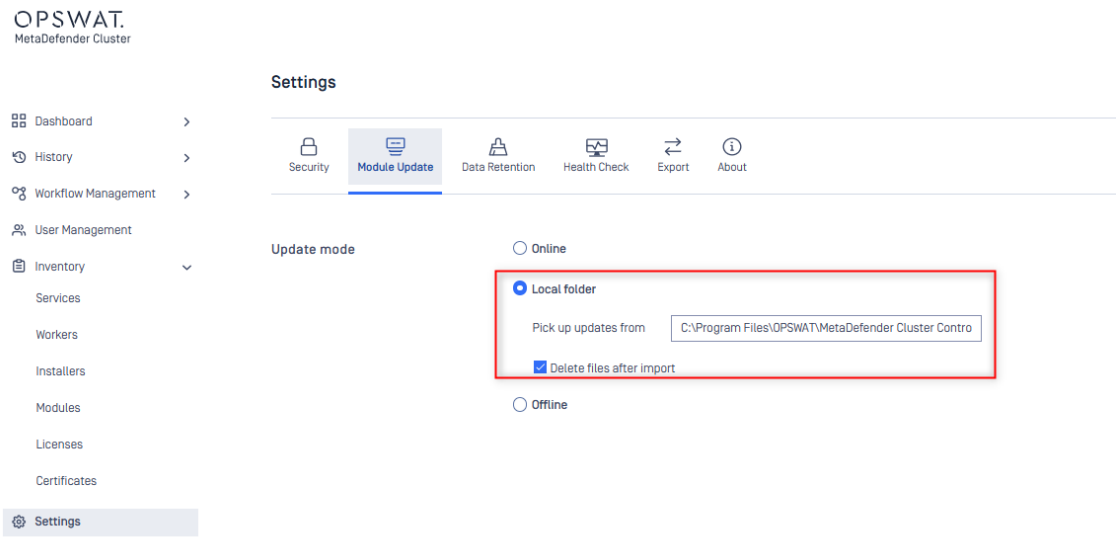
When using https://update.dl.opswat.com, whitelisting by IP address on your firewall may not work reliably over time. This is because OPSWAT uses a CDN (Content Delivery Network) to deliver updates faster worldwide, and the IP addresses of the edge servers can change periodically.

Local Folder

Choosing Local Folder method will allow MD Cluster Control Center to monitor any changes to the specified folder. If there are any changes, MD Cluster Control Center will apply the new module updates.

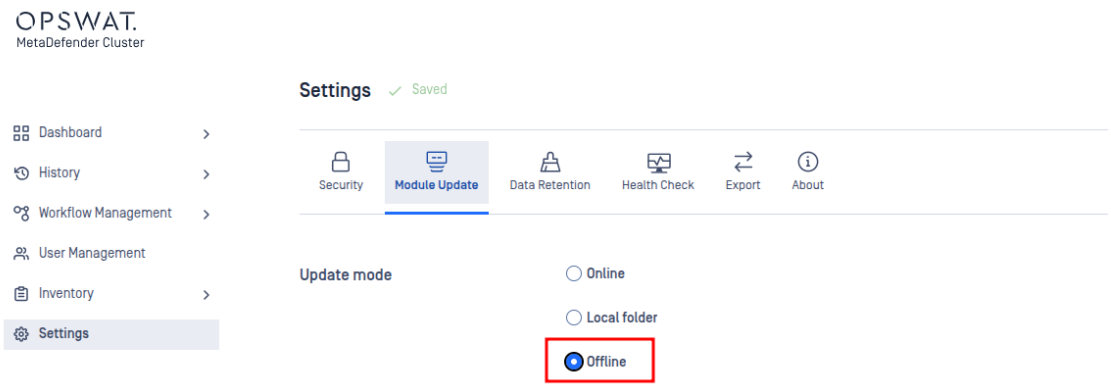
Info

MetaDefender Update Downloader must be used to download the modules.



Offline

Choosing Offline method will turn off the module update mechanism.

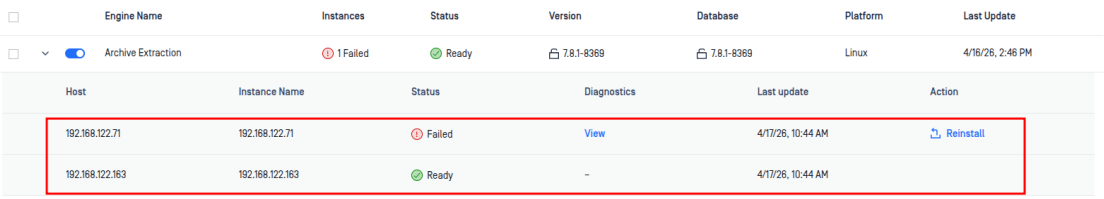


Engine management & engine advanced settings

MD Cluster Control Center centrally manages and synchronizes engine and module configurations across all MD Core instances. Any changes made in MD Cluster Control Center—such as enabling or disabling an engine, pinning or unpinning, editing, deleting or restoring, or reinstalling—are automatically propagated to every MD Core in the cluster, ensuring a consistent module state throughout the system.

Engine status

System administrators can monitor the real-time status of engines across the system.



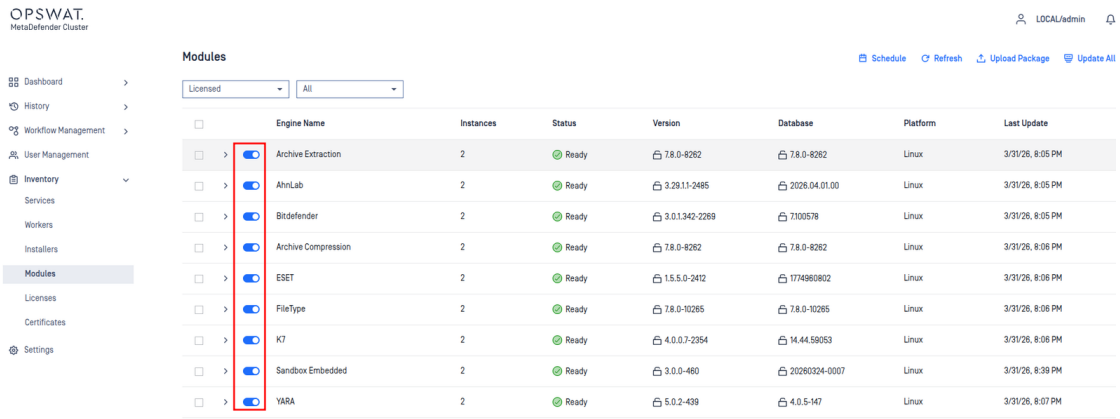
Each engine displays its deployment status across multiple instances, as outlined below:

Status	Description	Action
Ready	The engine is fully initialized, up to date, and ready to scan files.	No action needed.
Inactive	The engine is installed but pending startup.	No action needed; It will start automatically.
Disabled	The engine has been manually turned off by an administrator.	Enable it if scanning is required.
Failed	A critical error occurred during startup or scanning, often due to dependency issues.	Check the **Diagnostics** column or system logs, resolve the issue, then click **Reinstall**.

Status	Description	Action
Wait for installation	The system is currently downloading the engine binaries.	No action needed; just wait for it to finish.
In Progress	The engine is initializing its services and loading updates.	No action needed; wait until the process completes.
Mismatched	There is a version mismatch between this MD Core and the MD Cluster Control Center.	No action needed; the engine will sync automatically within a few minutes.

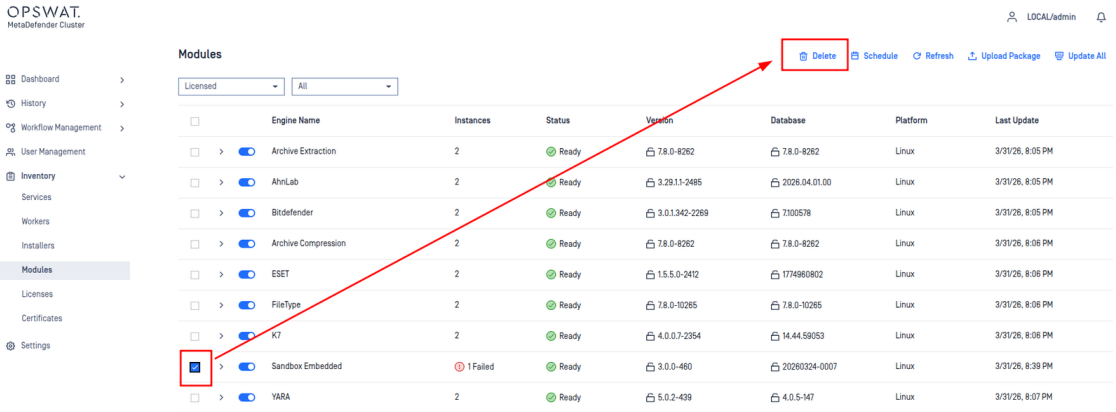
Enable & Disable engines

An engine can be disabled by switching off the toggle at the beginning of its row. Turn it back on the same way whenever needed.



Remove engines

In some cases, an engine may become unstable and remain in a failed state. When that happens, you can remove it by selecting the engine and clicking `Delete`.

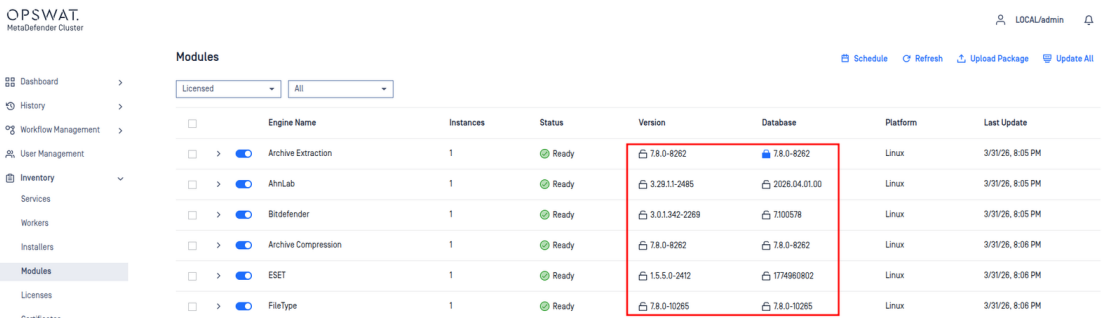


Lock & Unlock engines

You can lock an engine and its database to prevent new updates from being applied, even when auto-update is enabled.

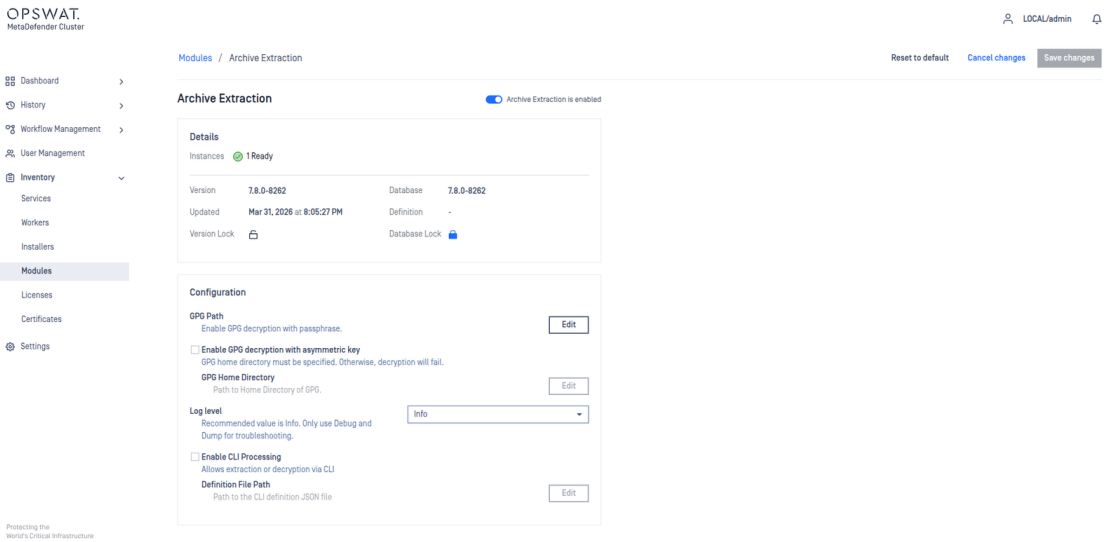
Info

- You can lock the engine and its database separately for the same engine.
- Once locked, that part (engine or database) won't receive any updates—even if the `Update All` button is triggered.



Engine advanced settings

Some engines offer advanced settings that you can configure.



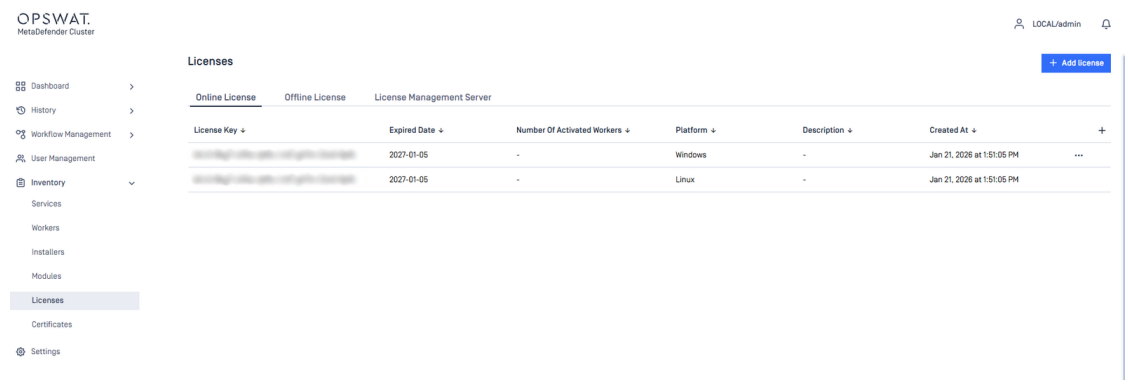
After selecting your preferred options, click `Save Changes` and your updates will be automatically synchronized across all MD Core instances.

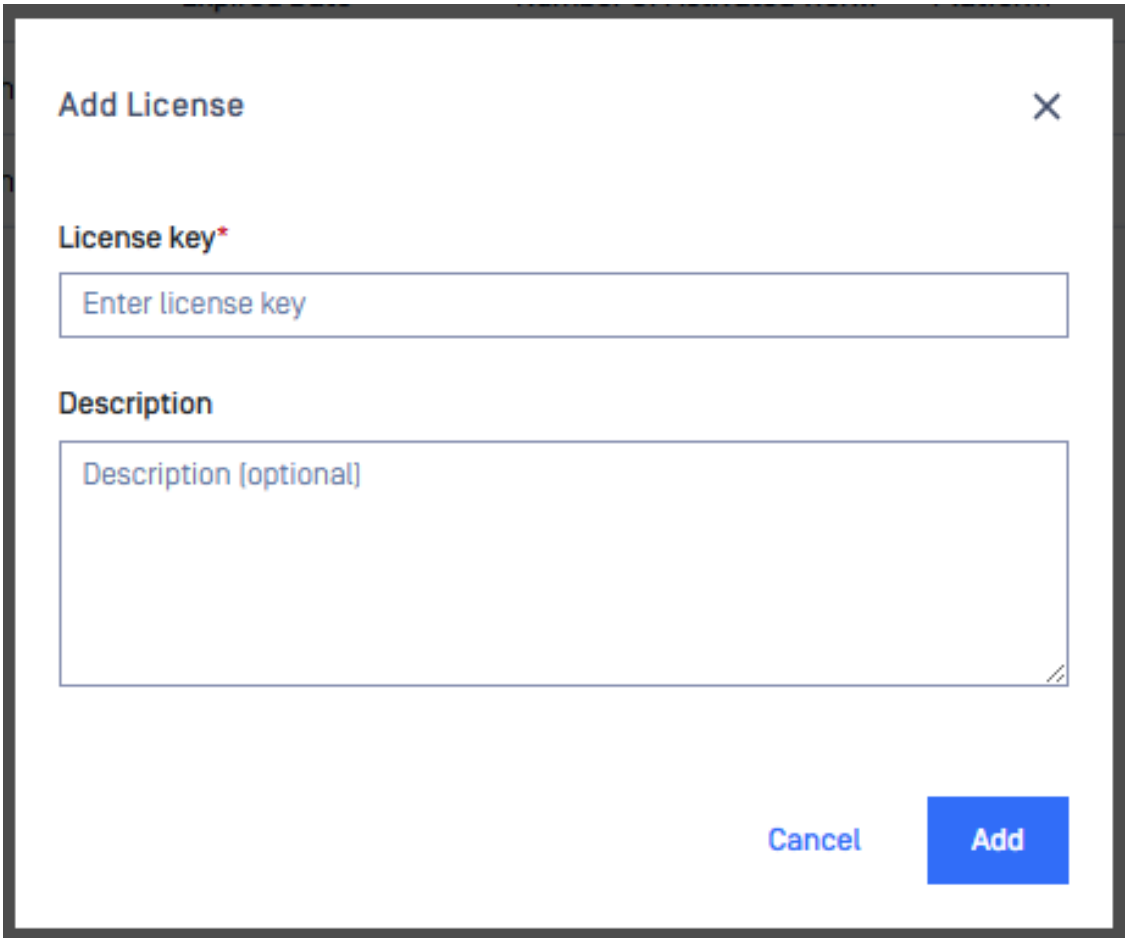
Licenses

MetaDefender Core licenses can be added to this section. The system administrator can add multiple licenses but only one license can be used to activate MetaDefender Core.

Adding a License

To add your license, simply select `Add License` and enter the license key then select `Add`.





Online activation

MD Cluster Control Center will connect directly to the OPSWAT licensing server if connected to the internet.

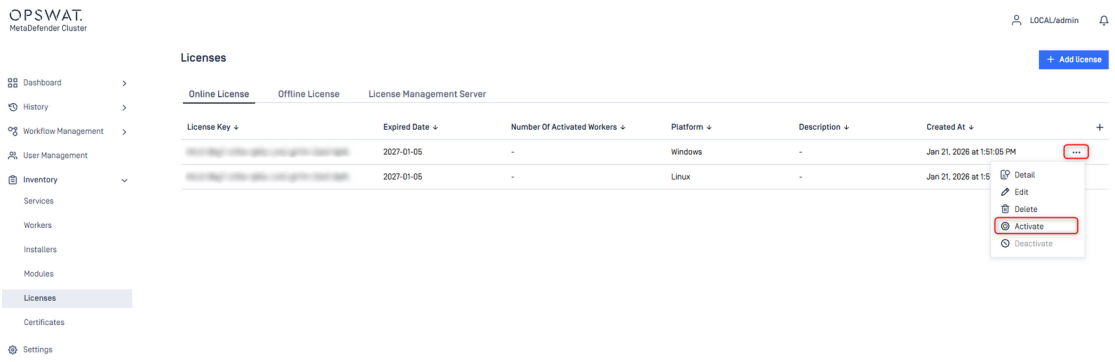
Warning

MD Cluster Control Center will need access to the following host for license activation:

- https://activation.dl.opswat.com Note: *IP address-based whitelisting on your firewall might fail after some time since OPSWAT uses CDN (Content Delivery Network) to faster delivery updates over the world, and IP*

address of edge servers might change over time.

To activate the MetaDefender Core instances simply select the three dots to the right of the license and select `Activate`.



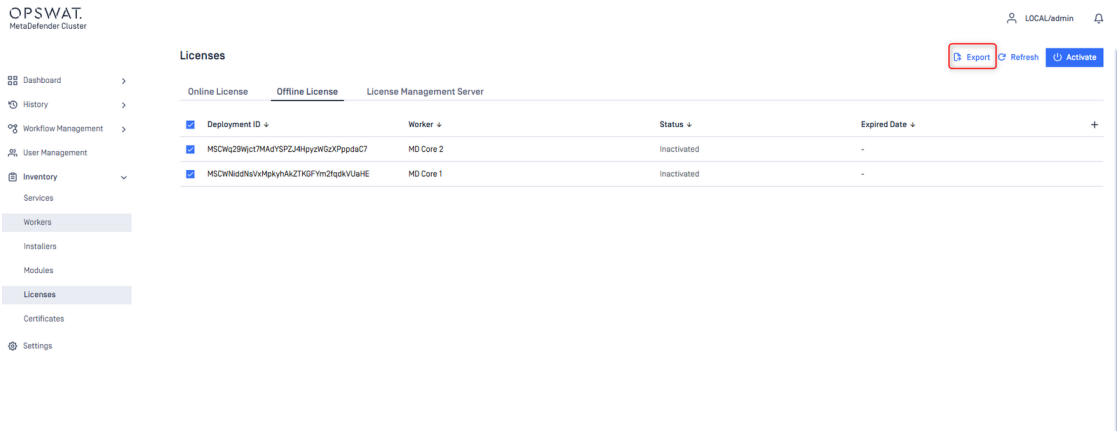
Offline activation

With no internet connection on MD Cluster Control Center, MetaDefender Core instances can be activated indirectly from a different machine that has internet connection. The system administrator must obtain all DeploymentID's of the MetaDefender Cores and then activate each DeploymentID separately to obtain their license file from My OPSWAT.

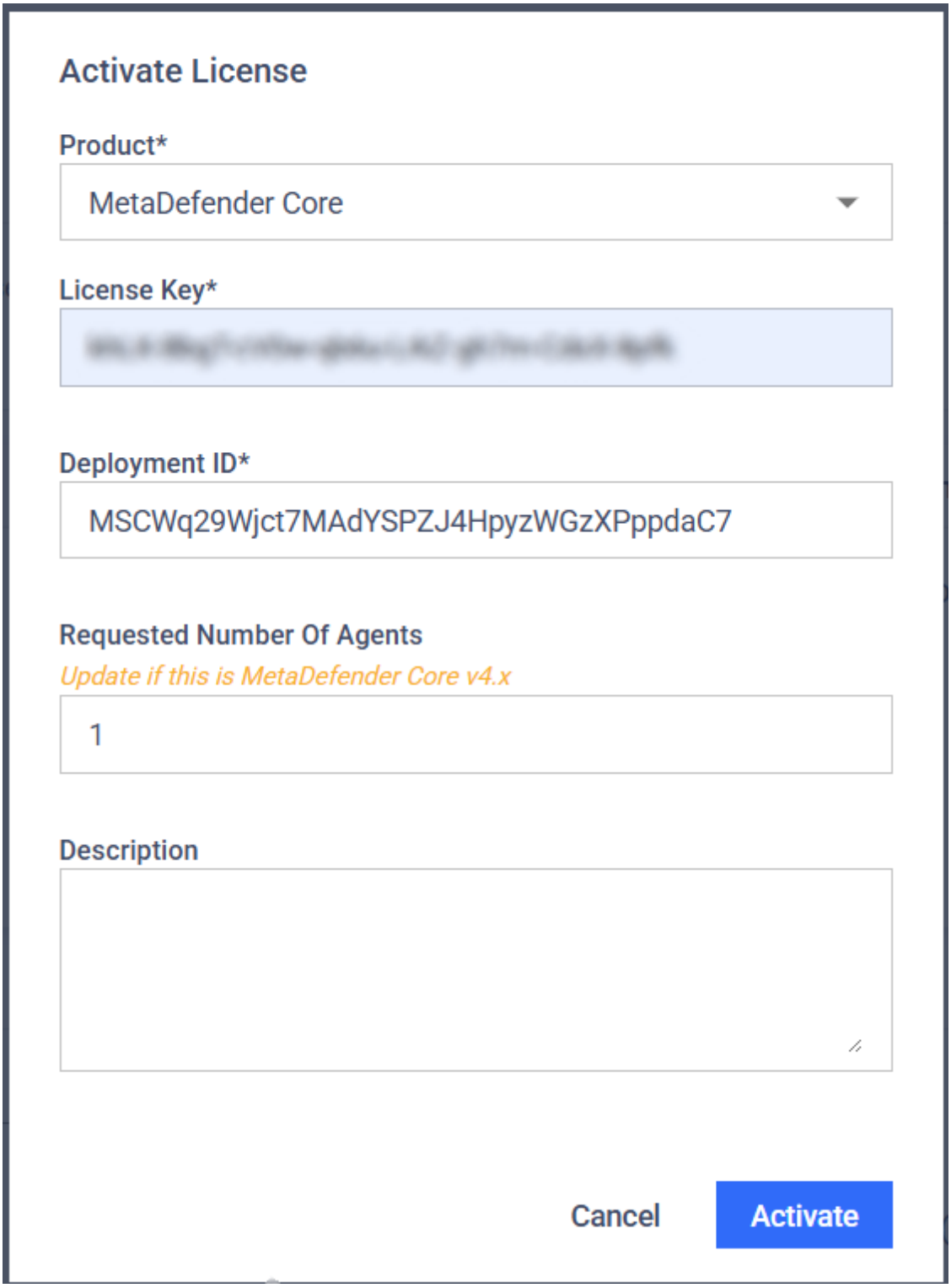
Once the system administrator obtains all the license files. Simply activate the MetaDefender Core instances on MD Cluster Control Center.

Follow the steps below to activate MetaDefender Core:

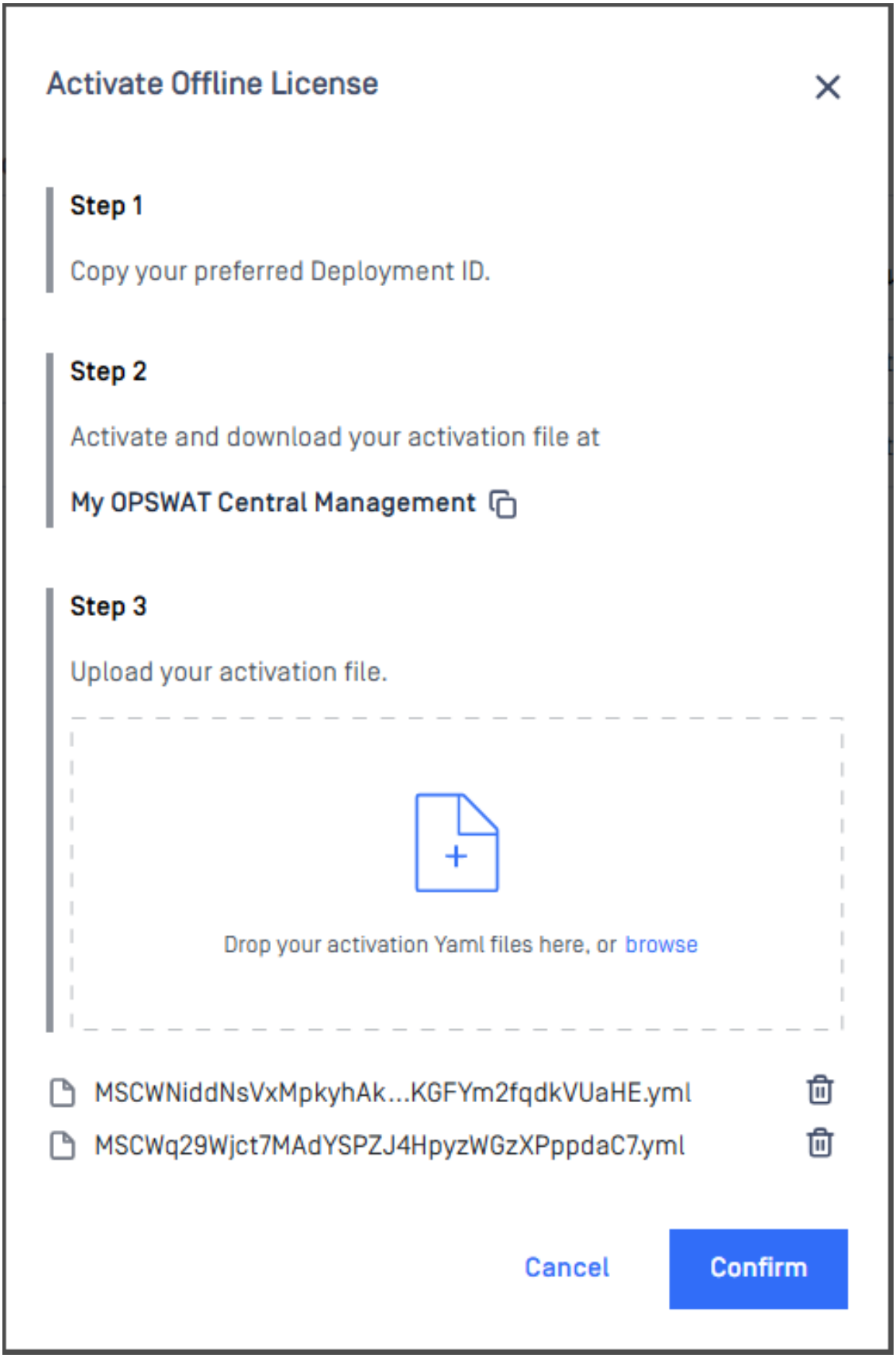
- Select a list of Deployment ID to activate then select `Export`. This will allow you to download a text file containing all the DeploymentID's that were selected.



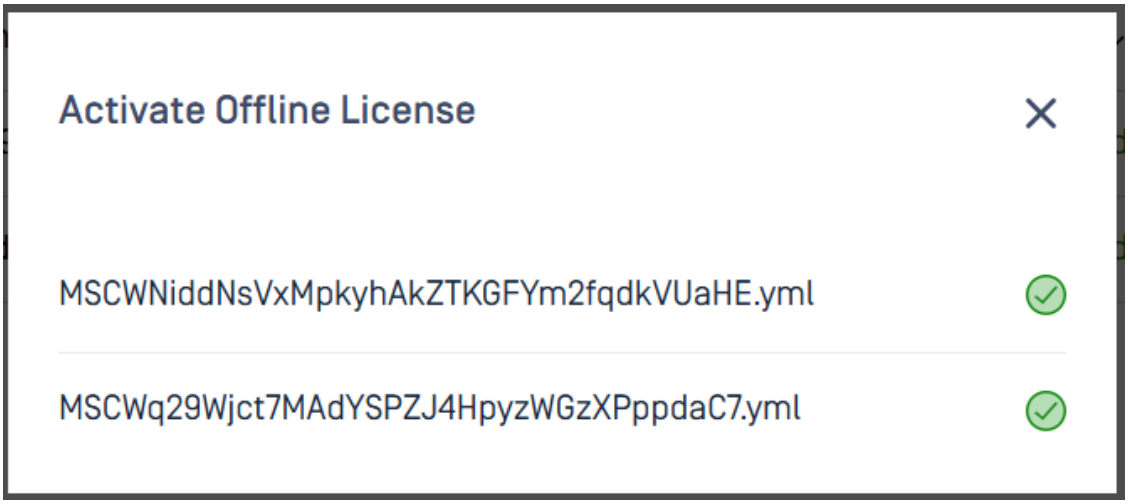
- Open the exported file to reveal all the selected DeploymentID.
- Sign in to My OPSWAT and then head to `License Management`-> `Activate License` and for each DeploymentID, click `Activate` and download the license files.



- Once all the license files are obtained. Head back to MD Cluster Control Center -> `Inventory` -> `Licenses` -> `Offline Licenses`. Click `Activate` to upload activation files to activate.



- By clicking `Confirm`, MD Cluster **Control Center** will verify the validity of the activation files and start activating MetaDefender Core accordingly.



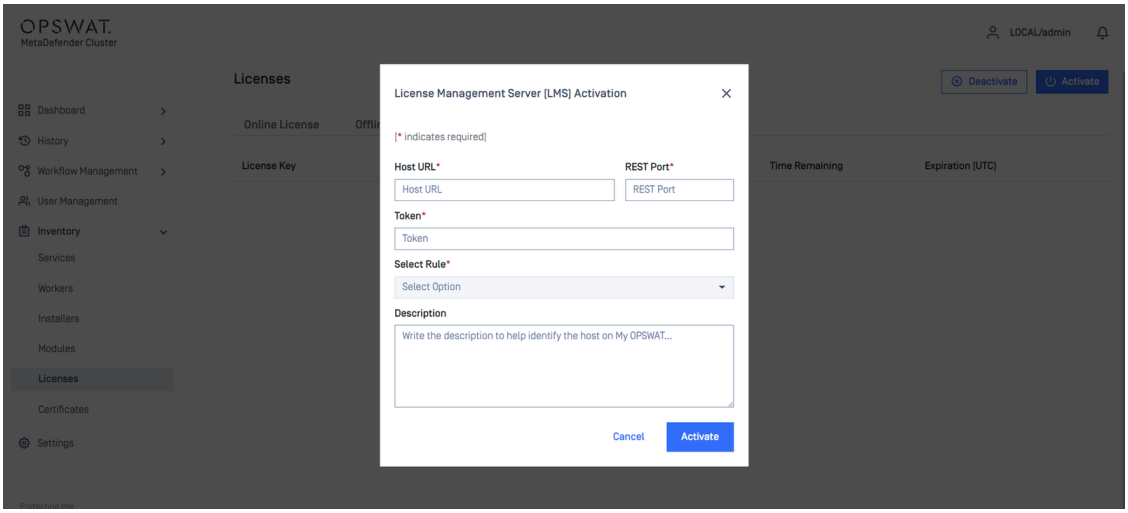
License Management Server activation

Using this licensing model, the License Management Server will act as a dedicated server that manages all license operations for MetaDefender Core. It is deisgned for organizations that prefer or required on-prem or cloud based solutions.

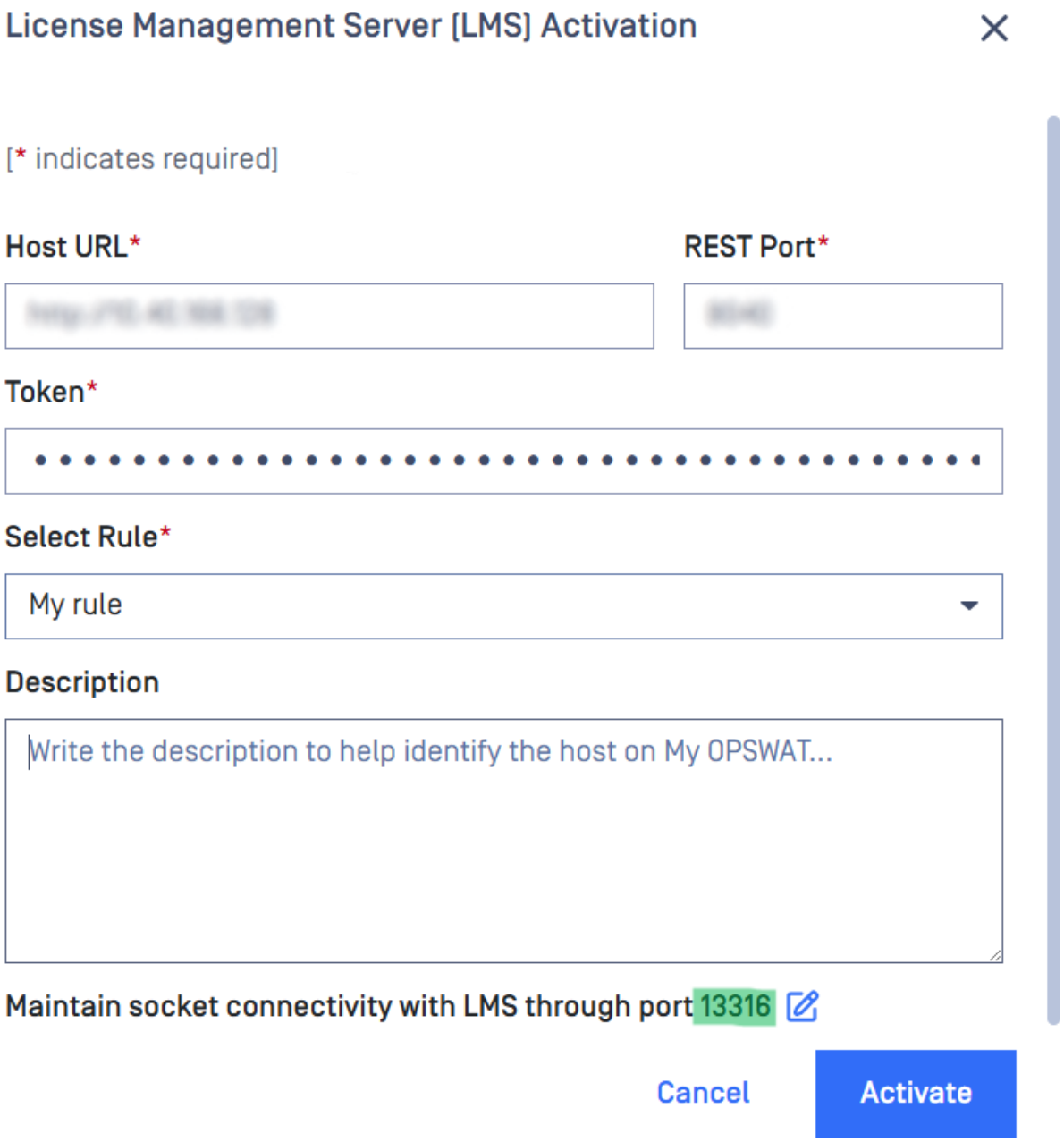
The License Management Server will handle the full license lifecycle - from activation to deactivation, renewal.

Follow the steps below to connect and activate with License Management Server:

- Sign in to MetaDefender Cluster **Control Center** console.
- Go to `Inventory` > `Licenses` and select License Management Server tab.
- Select `Activate` and provide the necessary information in the required fields:
  - **Host URL**: The URL of the License Management Server to connect to.
  - **REST Port**: The port of the License Management Server.
  - **Token**: Access token obtained from the License Management Server.



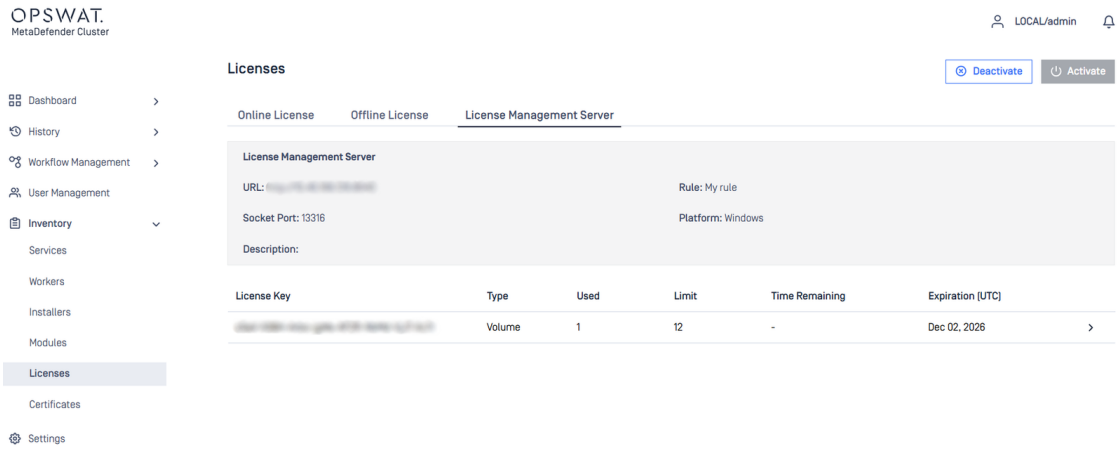
- After input all required fields, the connection to LMS will be established and available rules can be selected under `Select Rules`.



Info

The port number defined in "Maintain socket connectivity with LMS through port `<port_number>`" is required to sustain connectivity between the Control Center and License Management Server. In order for successfully activation, **confirm that the port is properly configured and allowed in firewall setting**.

- Select the appropriate rule and choose Activate. Upon successful completion, the license details will be shown.



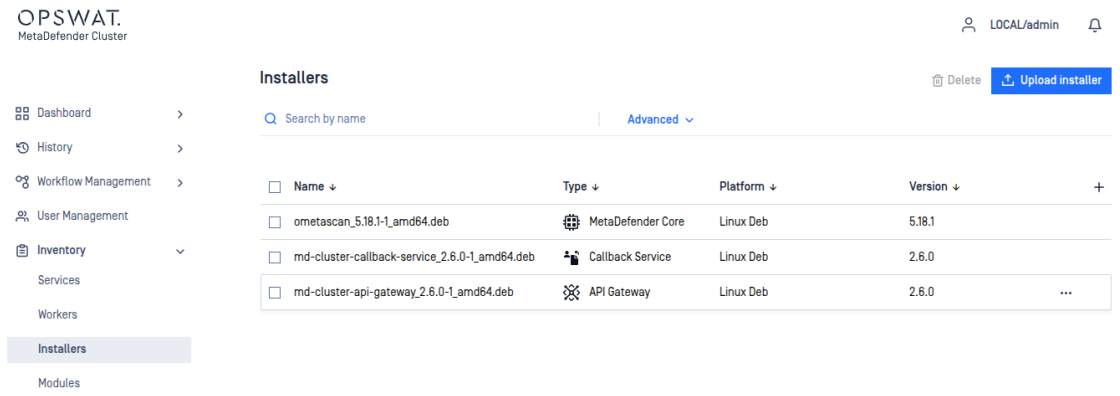
Info

Once activation is successful, the License Management Server will manage the license status of all MetaDefender Core instances. Please ensure sufficient quota is available.

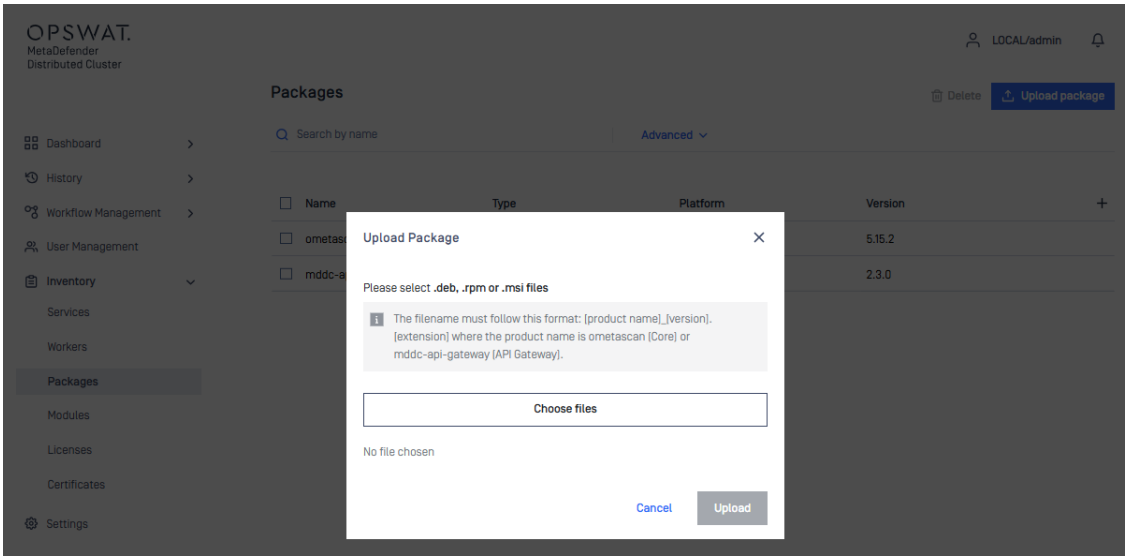
- Check instance status under `Inventory` > `Workers` and confirm all MetaDefender Core instances is activated successfully.

Installers

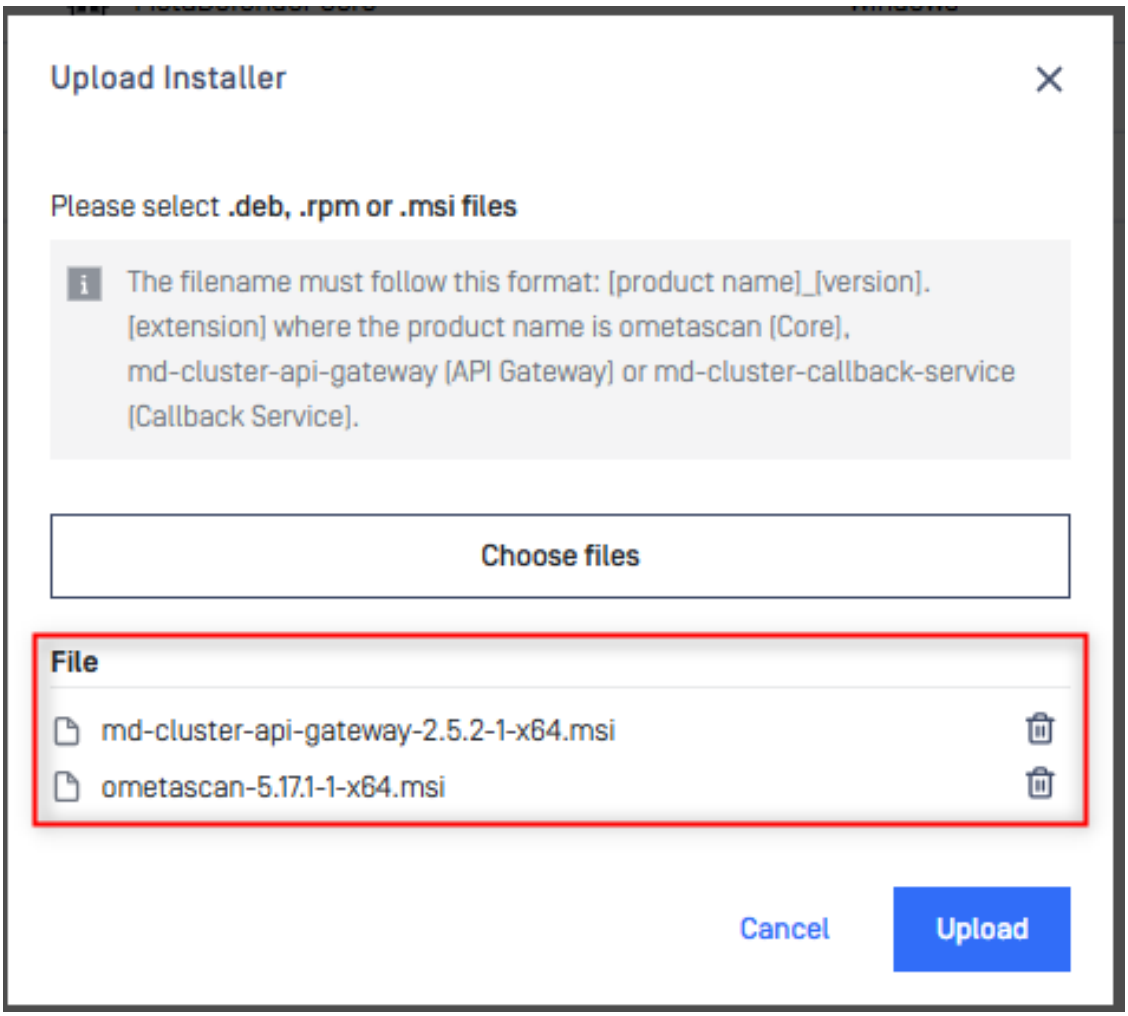
The Installers section lists all MetaDefender Cluster (MD Cluster) API Gateway, MD Cluster Callback Service and MetaDefender Core installers that have been recorded so far. These installers are used when deplying to a Worker.



Once OPSWAT releases a new version of the MD Cluster API Gateway, MD Cluster Callback Service and MetaDefender Core. The system administrator needs to download the installers from the OPSWAT portal and upload them to the MD Cluster Control Center by clicking `Upload Package` button in the top right corner.



The system administrator can upload several installer files at once.



Installer files are uploaded to the MD Cluster File Storage and will be ready for deployment or upgrading.

Workflow Management

This section explains how MetaDefender Cluster (MD Cluster) selects and applies scanning workflows based on

client source IP addresses and REST API requests. It also describes how workflow rules determine scan processing behavior and validation.

How scan profiles are selected

MD Cluster can use different scanning profiles for different clients. The profile selection is based on the client's source IP address.

If multiple scanning profiles are available for a client, the client can specify which profile to use in the request. If no profile is specified, MD Cluster applies the first matching profile defined in the Workflow rules.

How file scans are processed through the REST API

When MD Cluster API Gateway receives a scan request through the REST API, it checks the client's source IP address against the configured workflow rules. The first matching rule determines which workflow is applied to the scan request.

If the REST API request explicitly specifies a workflow, that workflow must also match an existing rule for the client. Otherwise, the scan request is rejected.

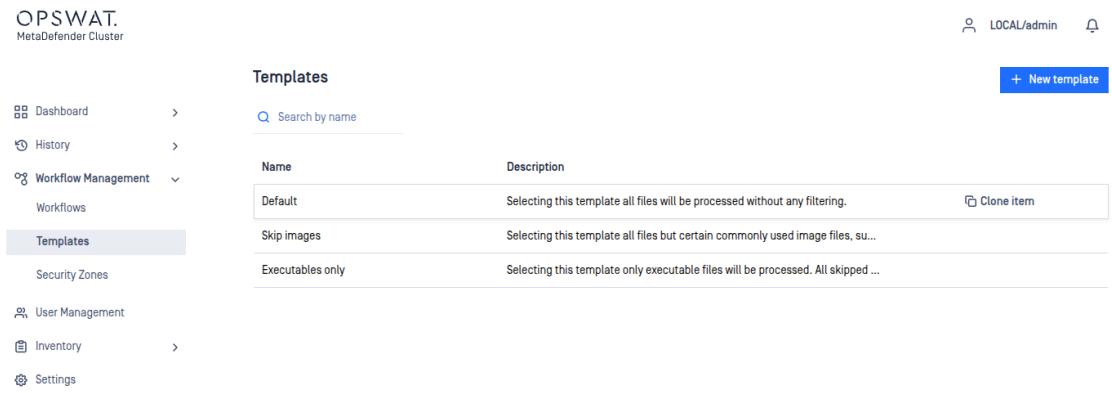
Workflow Configuration Template

The Workflow templates page is found under `Workflow Management` > `Templates` after successful login.

These workflow templates define the processing methods that can be used by the rules.

MetaDefender Core comes with predefined workflow templates that can not be modified, however they can be copied and the created workflow templates are fully customizable.

> **Info**
>
> It is highly recommended to use less workflow template and rather more rules based on the workflow templates.



Workflow templates

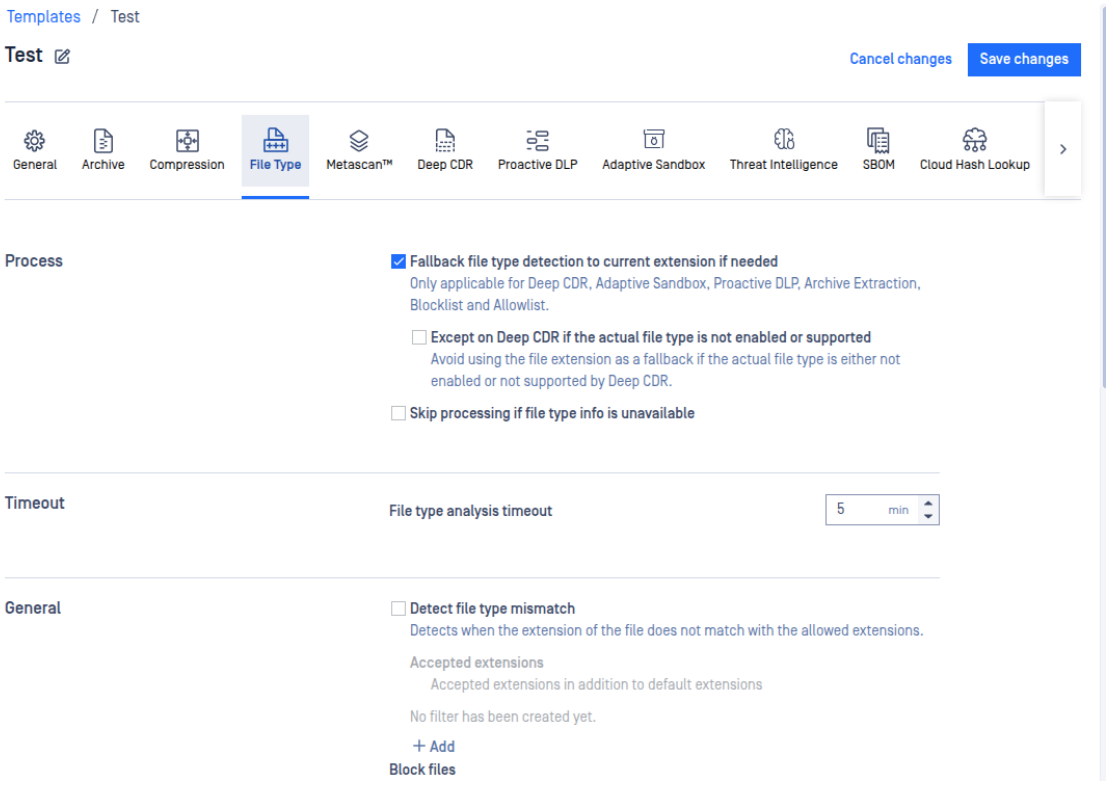
When clicking on a workflow template a windows pops up showing different tabs related to the workflow templates different kind of properties.

File Type

For all information about features powered by File Type, please learn more at File Type Verification Engine.

Detect file type mismatch can be enabled to identify files where the file extension differs from the actual file type.

Other configurations of File Type engine can be found at File type configurations.

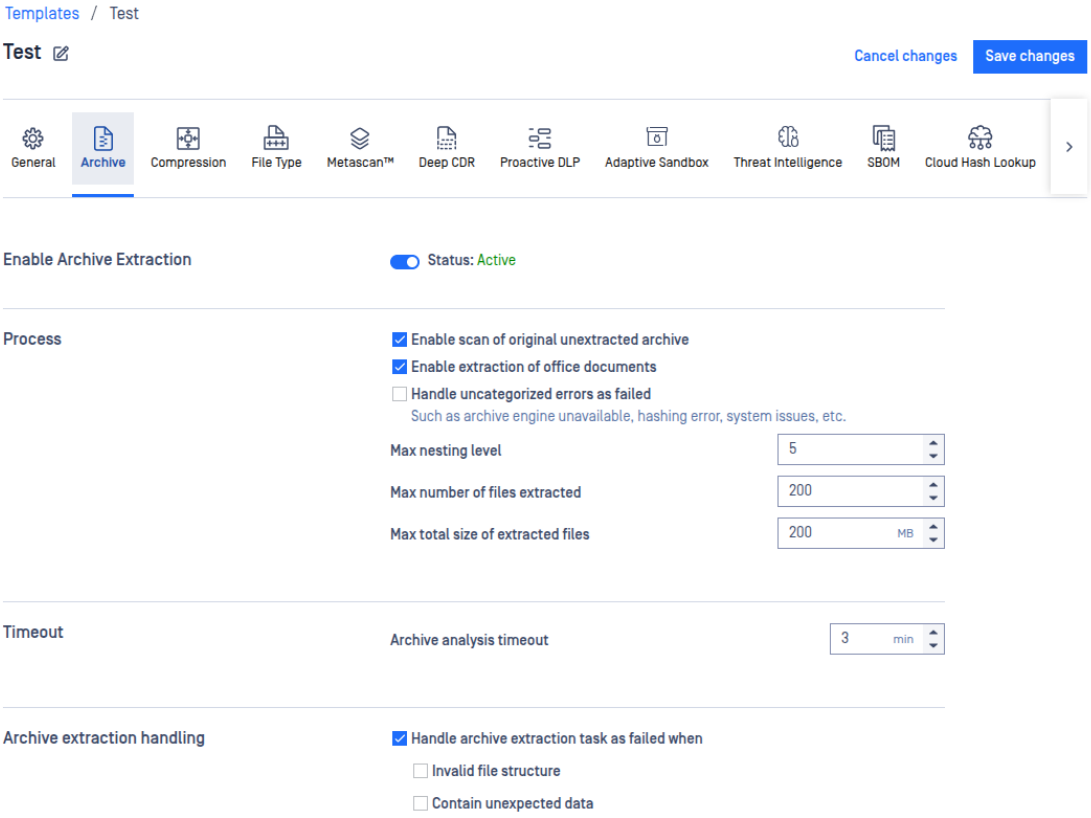


Archive

On the Archive tab the archive handling can be enabled or disabled as well as other parameters can be set.

The max recursion level defines how deep extraction should go into the archive, the number of maximum extracted files also can be set as well as the overall maximum size of these files.

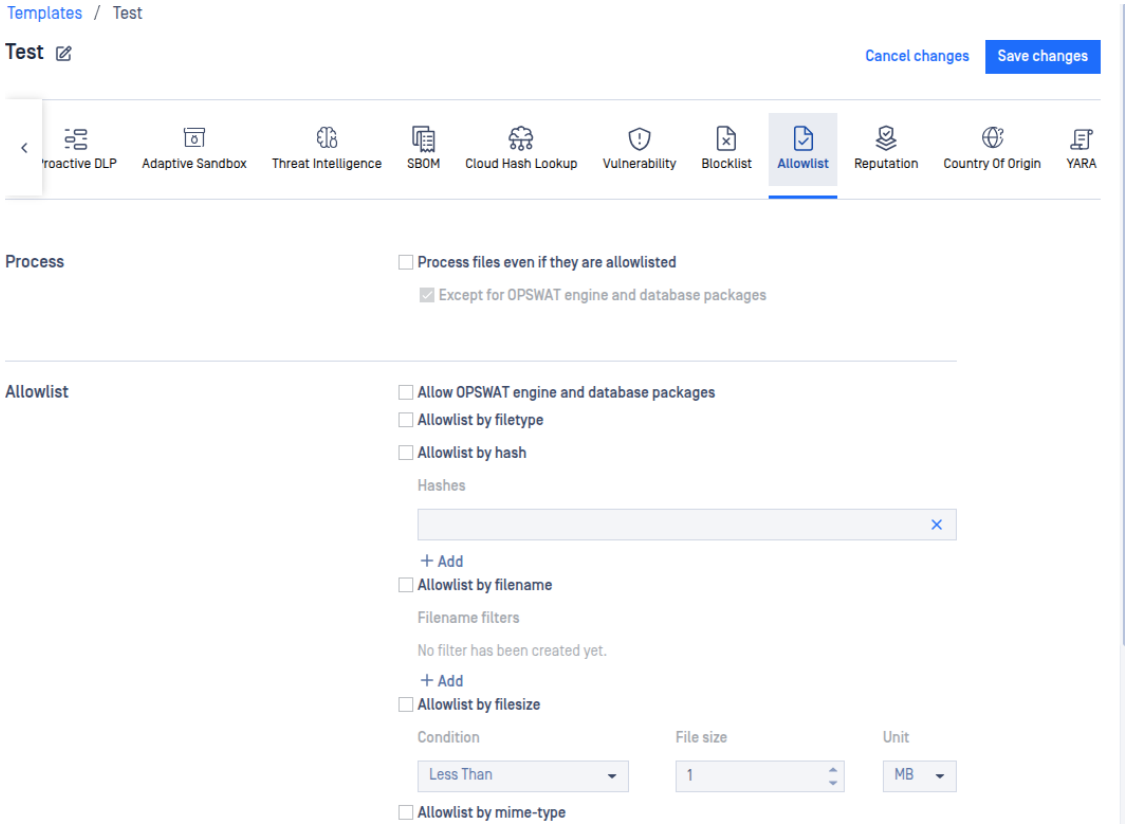
It is also possible to disable scanning the archive itself, and a timeout for the whole process can be set as well.



Blocklist

Info

Since MetaDefender Core 4.19.0, user is allowed to configure to exclude child extracted files from archive file type for being blocklisted.

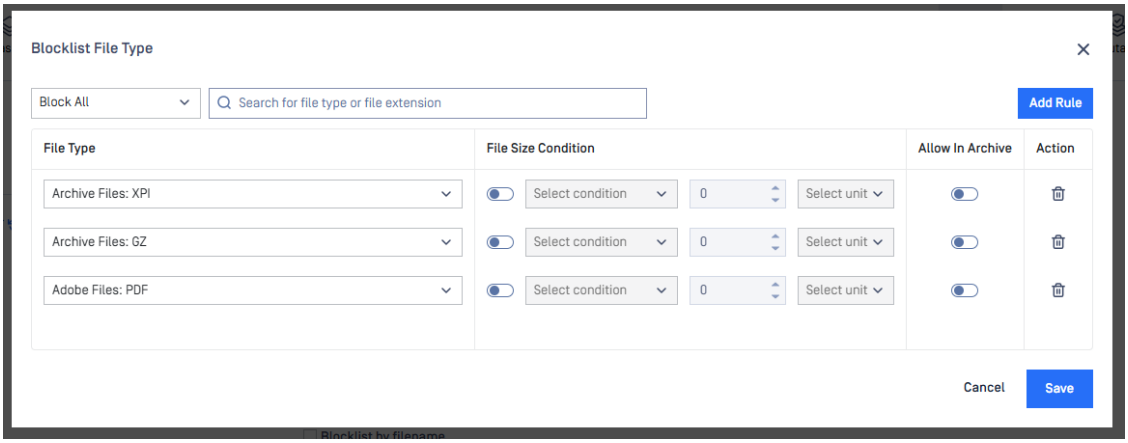


During scan, it is possible to create blocklists where files depending on their file type, checksum (hash), filename, extension, file size or MIME-TYPE. All of these can be stored in the fields on the Blocklist tab. Administrators can select tactic "Block All" or "Block All Except". Also it is available to blocklist all the files coming from the same group, such as Executable Files, Media Files and others.

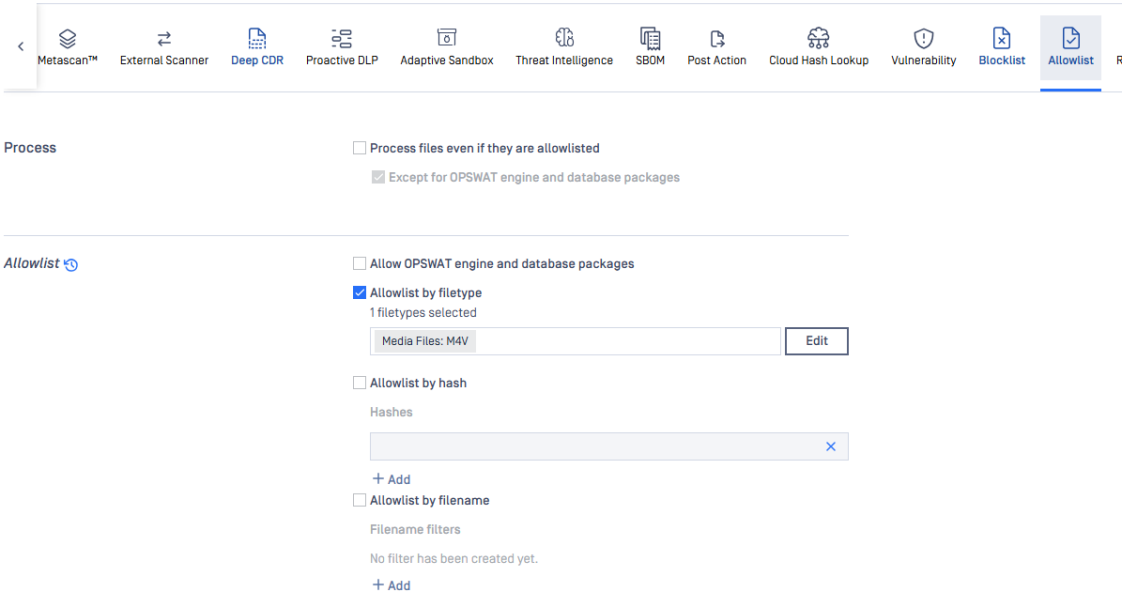
When filtering by mime-type or filename, the filter is handled as a regular expression.

Exceptions can be defined in **Exceptions (by mime-type)** section using regular exceptions. For instance, if all office files have to be blocked except docx files, then **Office documents** group should be chosen and `^application/vnd.openxmlformats-officedocument.wordprocessingml.document$` expression should be given as exception.

More details: Blocklist by filetype configuration



Allowlist



During scan, it is possible to create allowlists where files depending on their file type, checksum (hash), filename, extension, file size or MIME-TYPE. All of these can be stored in the fields on the Allowlist tab. Also it is available to allowlist all the files coming from the same group, such as Text Files, Office Documents and others.

When filtering by mime-type or filename, the filter is handled as a regular expression.

Exceptions can be defined in **Exceptions (by mime-type)** section using regular exceptions. For instance, if all office files have to be allowed except docx files, then **Office documents** group should be chosen and `^application/vnd.openxmlformats-officedocument.wordprocessingml.document$` expression should be given as exception.

Metascan

SKIP ANY FURTHER PROCESSINGS WITH AV SCANNING RESULT[S]: When enabled, this setting will halt further file processing if the aggregated antivirus (AV) scan result matches any value in the selected list. This can help optimize performance by avoiding redundant processing steps when a conclusive AV result has been obtained.

SUSPICIOUS DETECTED HANDLED AS: By enabled, you are able to decide if Suspicious result on any particular engine is considered as Infected or No Threat Found result, and it will take consideration into overall process result which also is constraint by threat detected threshold setting.

SCAN FAILURE THRESHOLD: It is possible to enable and set a threshold value for failed engine results. If the number of failed results for the currently scanned object reaches this threshold, the overall result will be marked as failed. This threshold value will not influence the scan result if there are other outcomes, such as "suspicious", except in cases where the result is categorized as "infected".

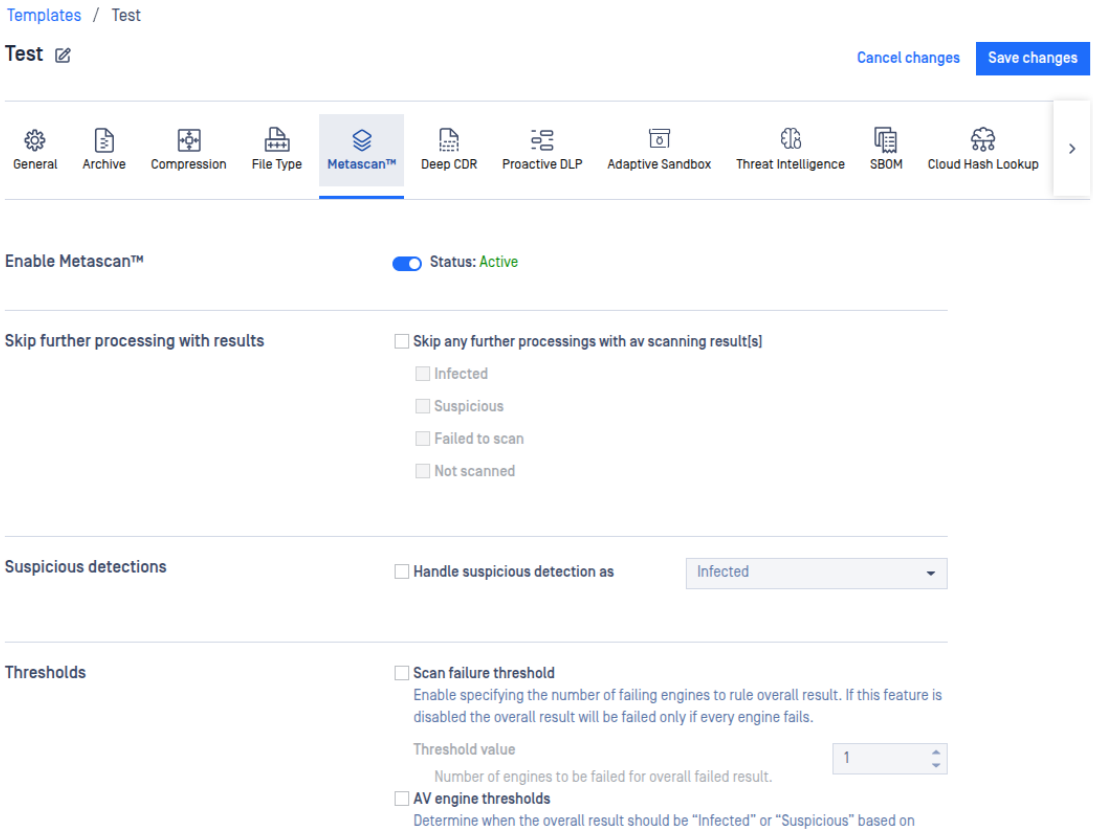
THREAT DETECTED THRESHOLD: When checked, this setting supports two configuration options INFECTED LIMIT and the SUSPICIOUS LIMIT when SUSPICIOUS DETECTED HANDLED AS is disabled, and its handling logic will be described as following:

- If the number of infected engine results is between these values the overall result will be suspicious.
- If the INFECTED LIMIT is reached the overall result will be always infected.
- If none of them is reached the overall result will be the highest priority engine result (infected results are ignored).

Nevertheless, if **SUSPICIOUS DETECTED HANDLED AS** is also enabled, regardless of handling as Infected or No Threat Found, SUSPICIOUS LIMIT setting are no longer taken into account. The following is the handling logic:

- If the INFECTED LIMIT is reached, the overall result will be always infected.
- Otherwise, the overall result will be the highest priority engine result (infected results are ignored).

PER ENGINE TIMEOUT / GLOBAL SCAN TIMEOUT: The timeout for the different engines and the whole scanning process also can be set. The maximum allowed size of scanned objects can be set also on this tab as well.

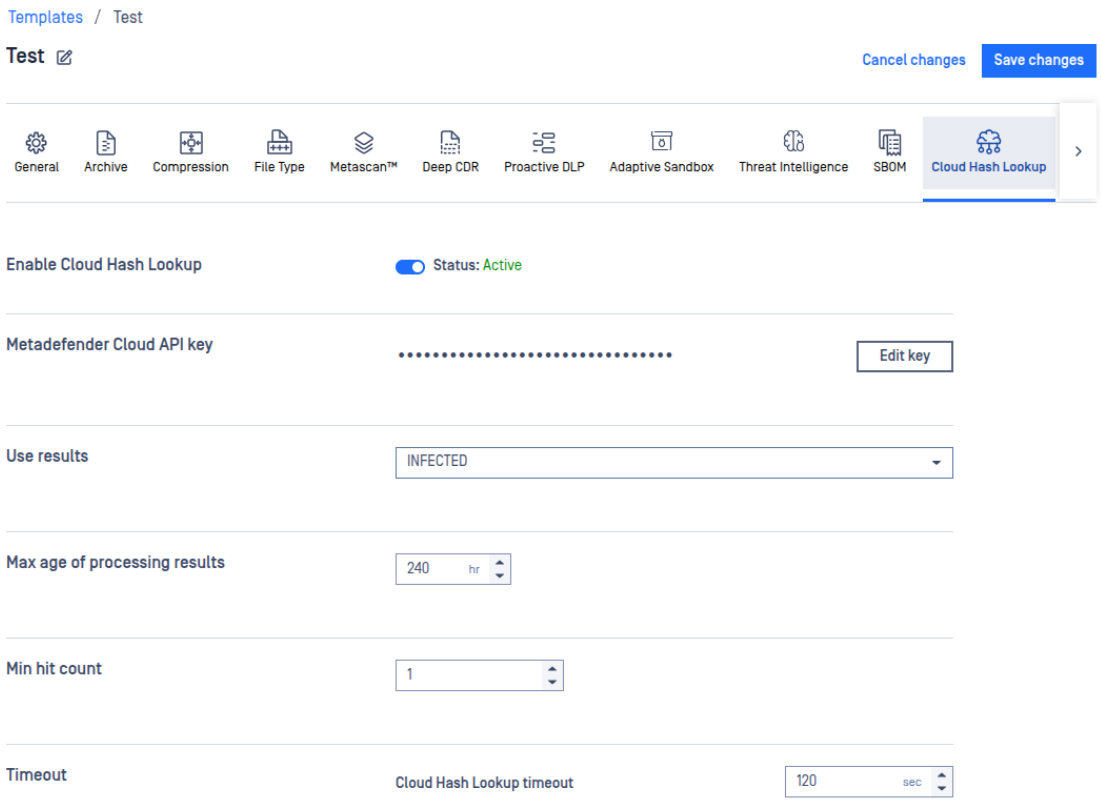


Cloud Hash Lookup

When MetaDefender Cloud workflow element is enabled, online database of MetaDefender Cloud will be used as source for hash lookups.

Available options:

1. **Use results**: INFECTED or ALL RESULTS If INFECTED is chosen, then only that result will be accepted as result, otherwise all type of results will be taken into account.
2. **MetaDefender Cloud API key**: An API key is necessary to have access to the MetaDefender Cloud database. API Key Information can be found on https://metadefender.opswat.com, under **Account Information** page.
3. **Maximum age of scan results**: Only results that are not older than what is set here will be considered as a valid result.
4. **Excluded engines' name**: Name of the engines whose results are not to be taken into account.
5. **Minimum hit count**: To consider a verdict as a valid one, there should be at least as many result for a hash as it has been set here. (If **Use result** is set to INFECTED, then only infected results will be counted in.)
6. **Time out**: The time interval within which the response should be received from MetaDefender Cloud.



Deep CDR

By enabling Deep CDR, one can convert from a set of supported file types into another (or the same). By doing so lot of vulnerabilities can be got rid out of rendering the resulting file be more safe. Both the types to be sanitized and the target file type can be set. To set the file types that you want to sanitize you should tick on corresponding checkboxes. In addition, you can also tick on "**ENABLE FOR ALL FILE TYPES**" to choose all supported file types. File name from sanitized files can be defined by using "OUTPUT FILENAME FORMAT" field. For usage and meanings of variables, please refer to Setup output file name page.

By default, MetaDefender Core allows files, where sanitization fails. This behavior can be overridden enabling "BLOCK FILES IF **SANITIZATION FAILS OR TIMES OUT**".

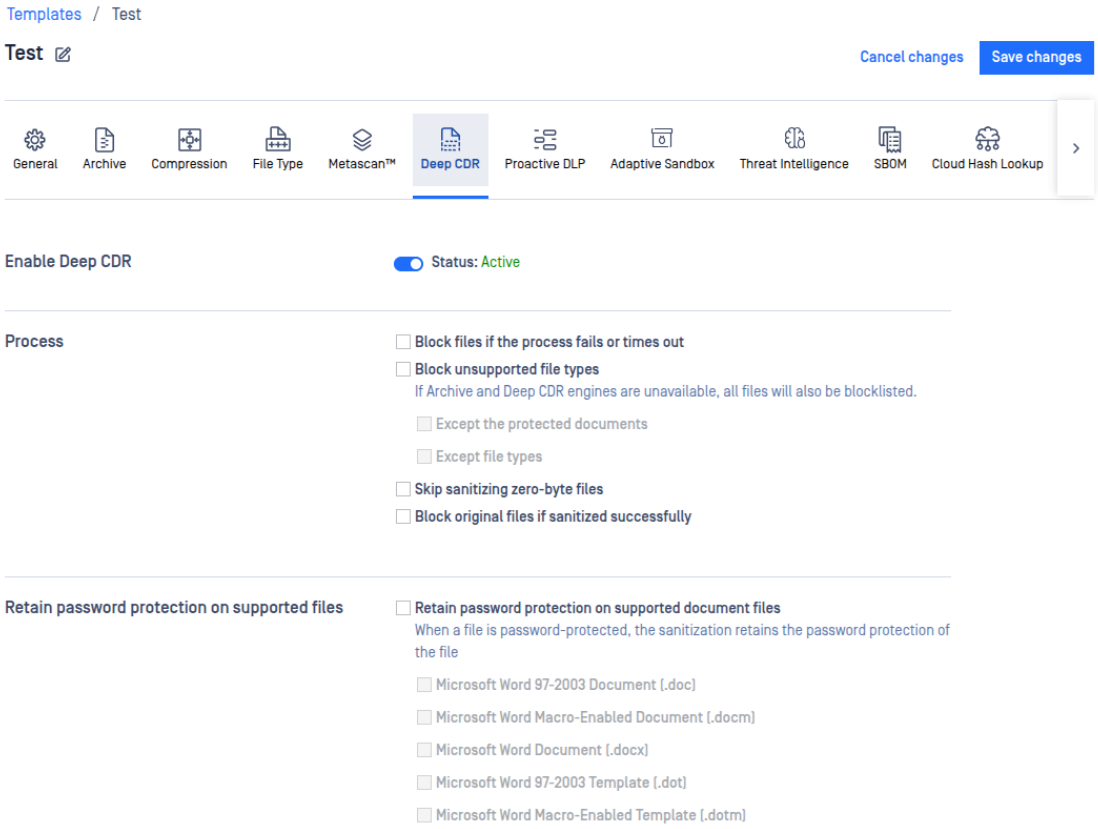
The maximum allowed time for data sanitization to be made can be configured through the "**DEEP CDR TIMEOUT**" and "**TRY COUNT**" options, where first one means that data sanitization should finish within the configured time frame, otherwise abort the conversion and latter means the number of times product should retry in case of a failed conversion.

When "**DISTINGUISH PARTIAL ARCHIVE SANITIZATION RESULT**" checked, MetaDefender Core will return "Partial Sanitization" processing result for Deep CDR when only some of child files in original archive files are sanitized successfully.

Beware, however, that possible data loss or change may occur during conversion, thus this feature is disabled by default.

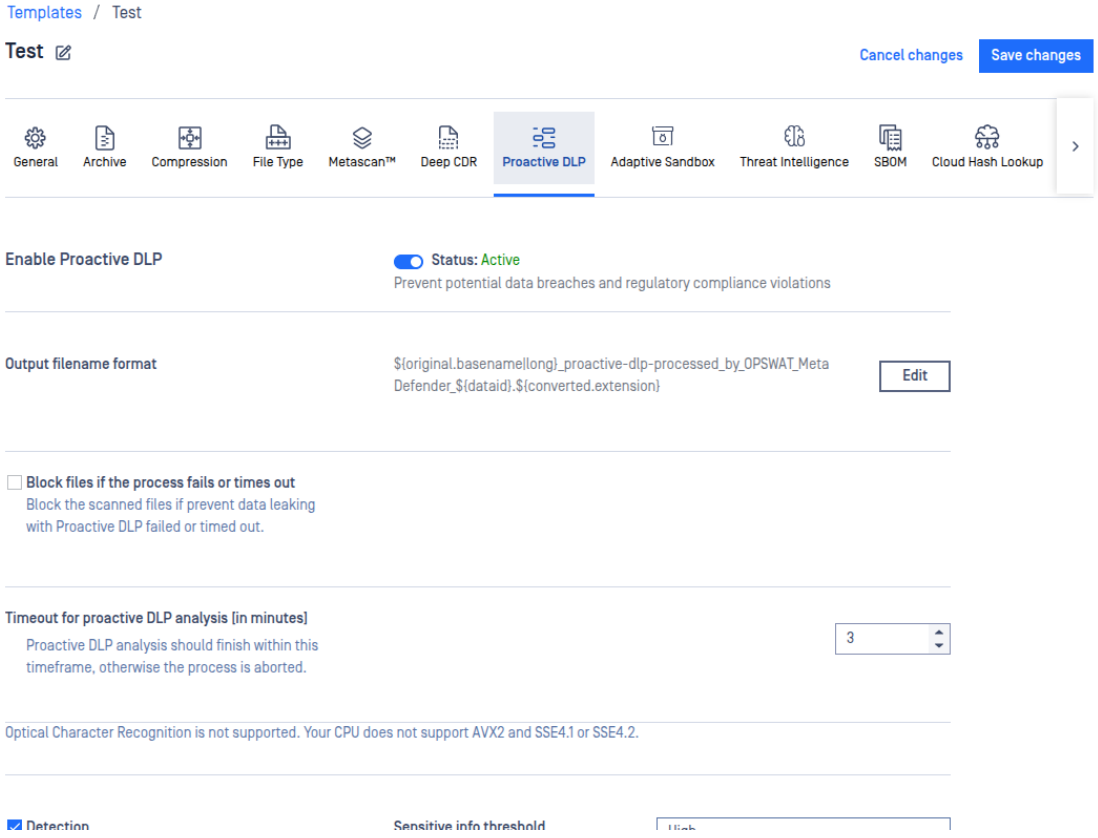
Result of sanitization can be either downloaded on the scan page or retrieved the data ID via REST. See Fetch processing result. Note that /hash API does not provide such information.

Length of time the system stores sanitized files can be set in **Settings > Data retention**.



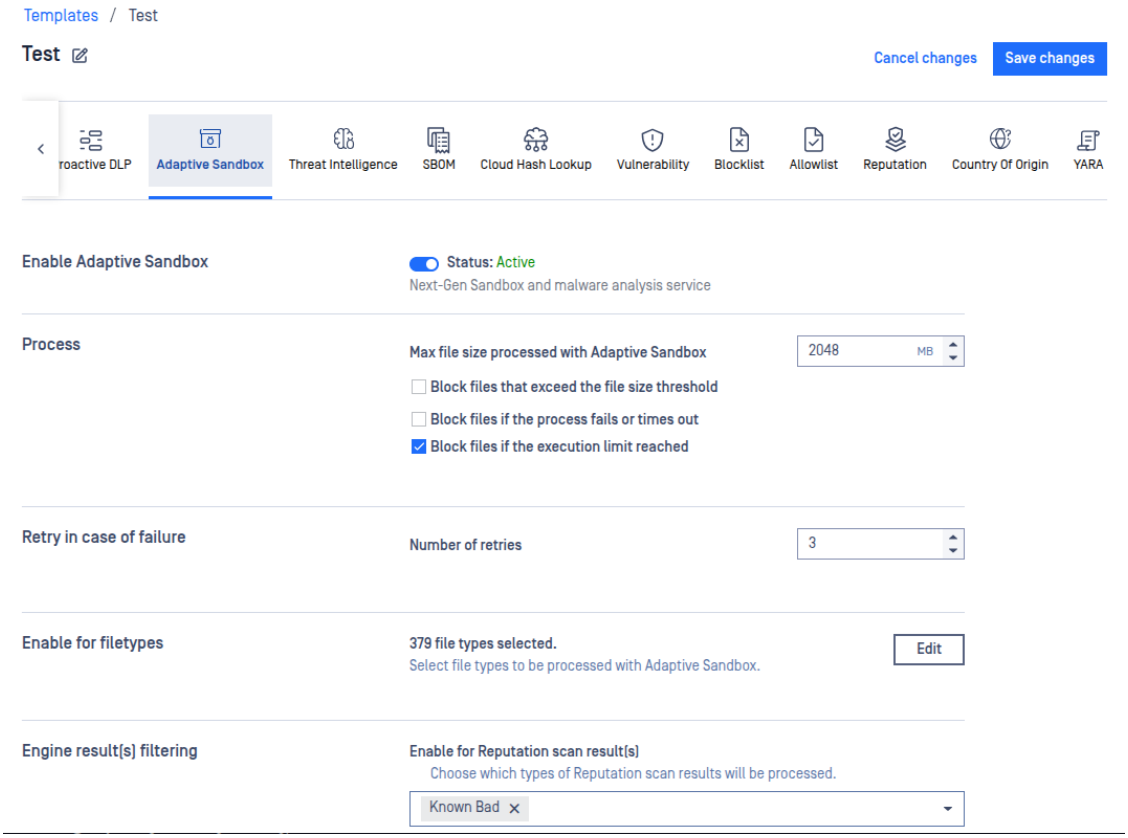
Proactive DLP

For all information about features powered by Proactive DLP, please learn more at Proactive DLP overview.



Adaptive Sandbox

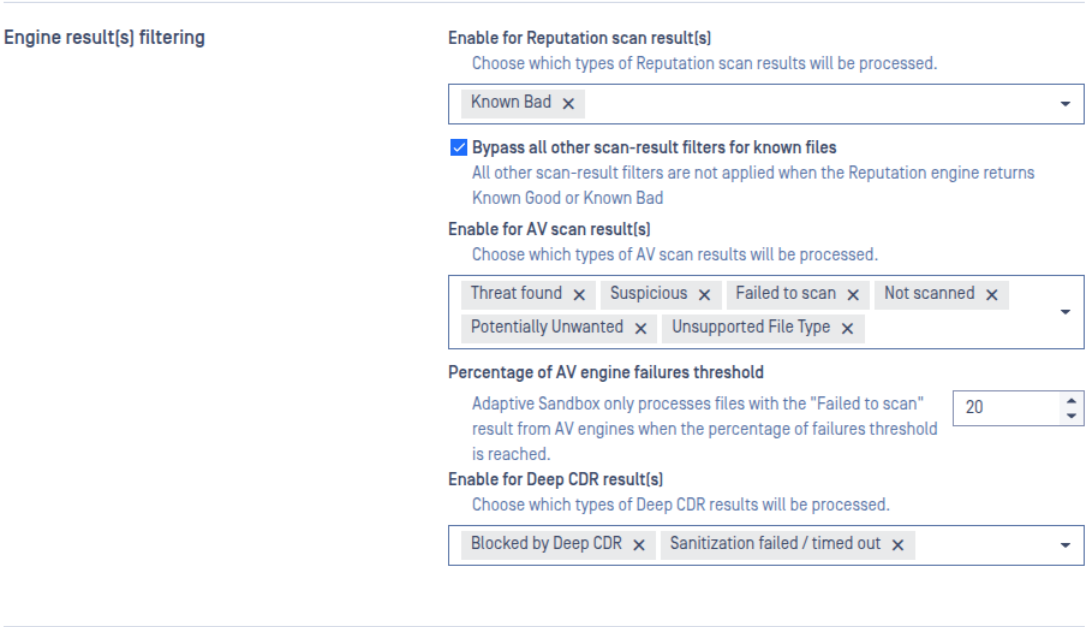
For all information about features powered by Adaptive Sandbox, please learn more at Adaptive Sandbox overview.



With File type Filtering configuration, you can configure Adaptive Sandbox engine to run only for the selected file types. All the supported file types are selected by default.

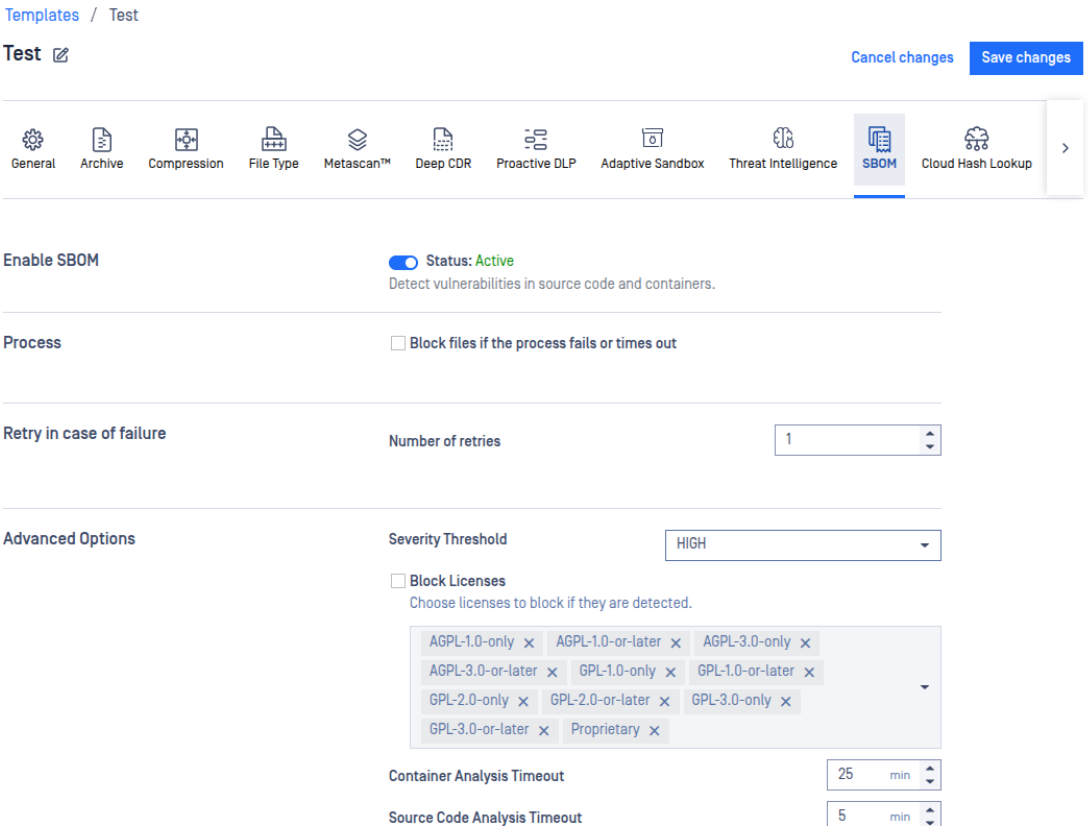
With the Result Filtering configuration, you can configure Adaptive Sandbox engine to process files based on

- All Reputation results, including Known Bad or Known Good.
- All Metascan AV scan results, including No Threat Found or any threats found. By default, No Threat Found is not selected.
- All Deep CDR results, regardless of successful sanitization or failures. By default, only sanitization failures are selected.



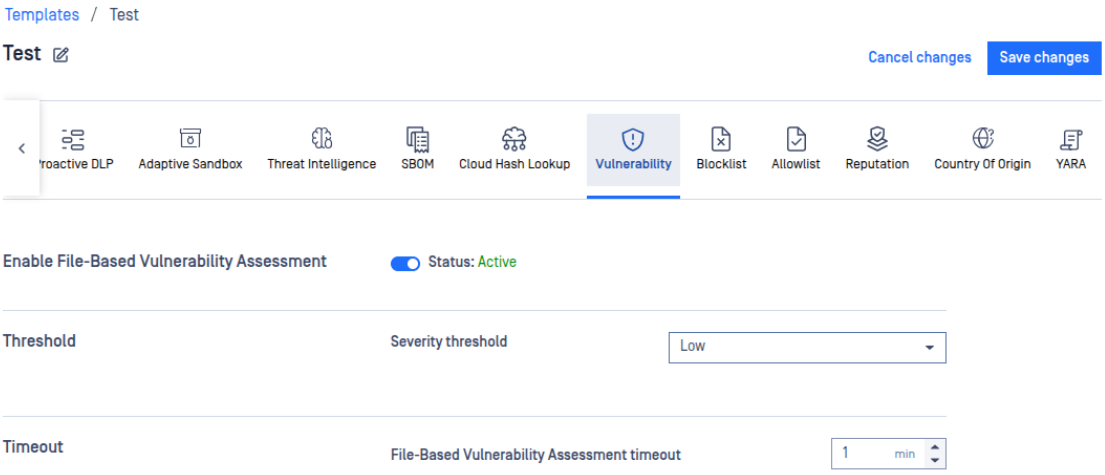
SBOM

For all information about features powered by SBOM, please learn more at Software Bill of Materials (SBOM).



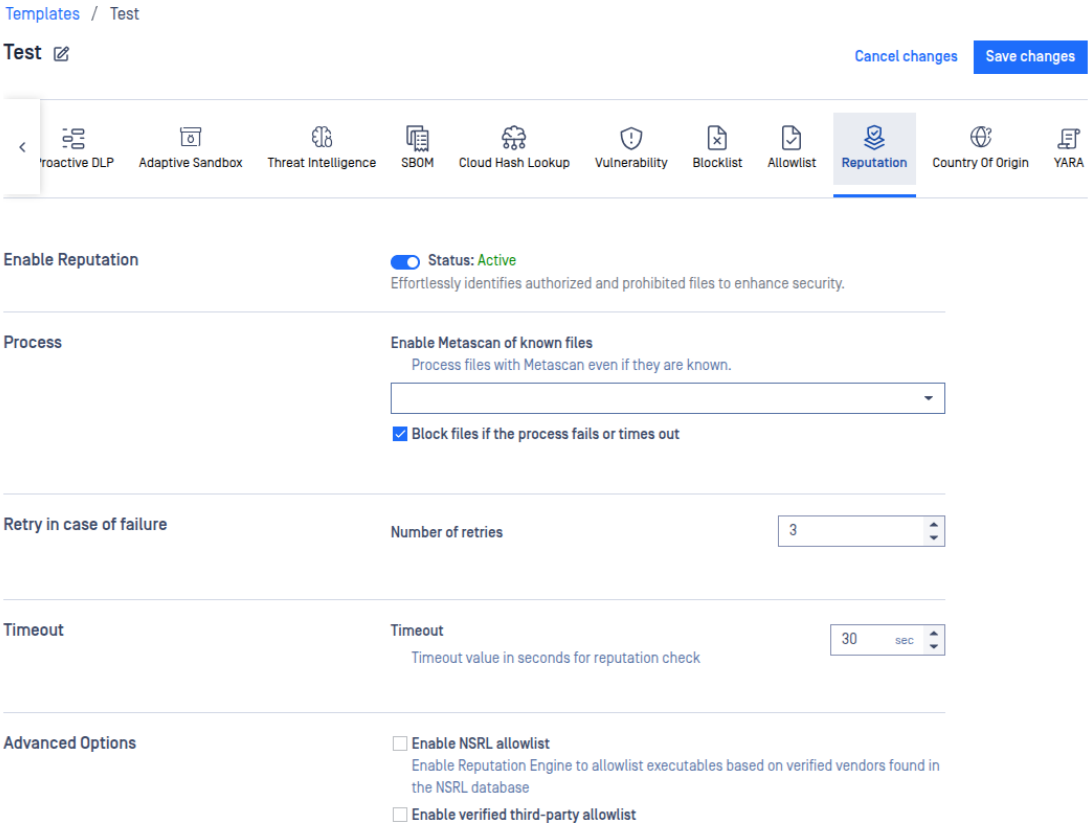
Vulnerability

File-based Vulnerability Assessment scans and analyzes binaries and installers to detect known application vulnerabilities. Administrators has an ability to configure severity threshold and timeout settings.



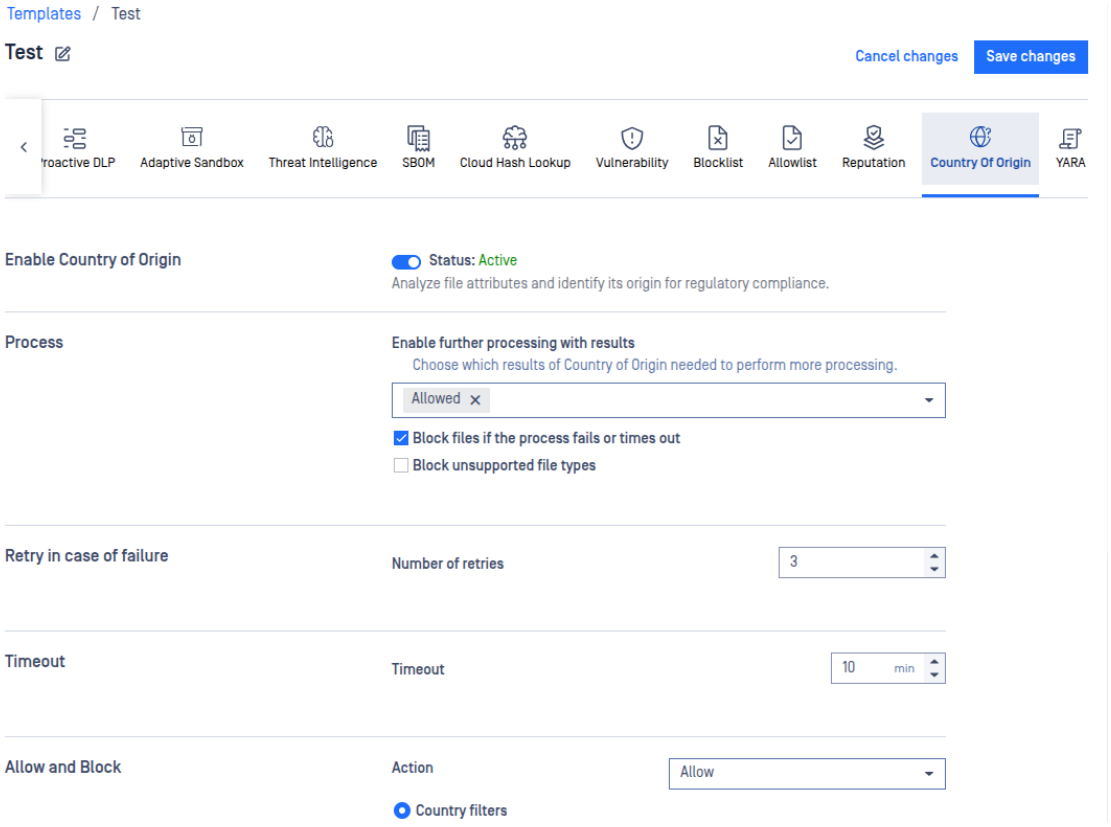
Reputation

For all information about features powered by Adaptive Sandbox, please learn more at Reputation overview.



Country of Origin

For all information about features powered by Country of Origin, please learn more at Country of Origin



General

By enabling '**Fallback filetype detection to current extension if needed**' (enabled by default), file type detection can use the extension of the currently processed file as a helping hand. For example this could be useful, when analyzing CSV files.

Setting '**Enable file hash for only selected types**' provides a list of supported hash types.

- By default, MD5, SHA1 and SHA256 are selected. SHA512 is unselected to avoid affecting processing performance, despite not much considerable.
- SHA256 cannot be deselected.
- File-based Vulnerability Assessment engine requires SHA1. Therefore, when SHA1 is deselected, it is expected to receive the result "Skipped the assessment" from the engine.

File download timeout: Scan-from-link is the feature supporting users to process file by just specifying a direct download link via the header "downloadfrom", instead of sending a payload in HTTP(S) request body. You are able to define timeout for file downloading process of this feature.

By enabling '**Global Timeout**' (disabled by default), application provides a global timeout for entire processing applied to every scan request.

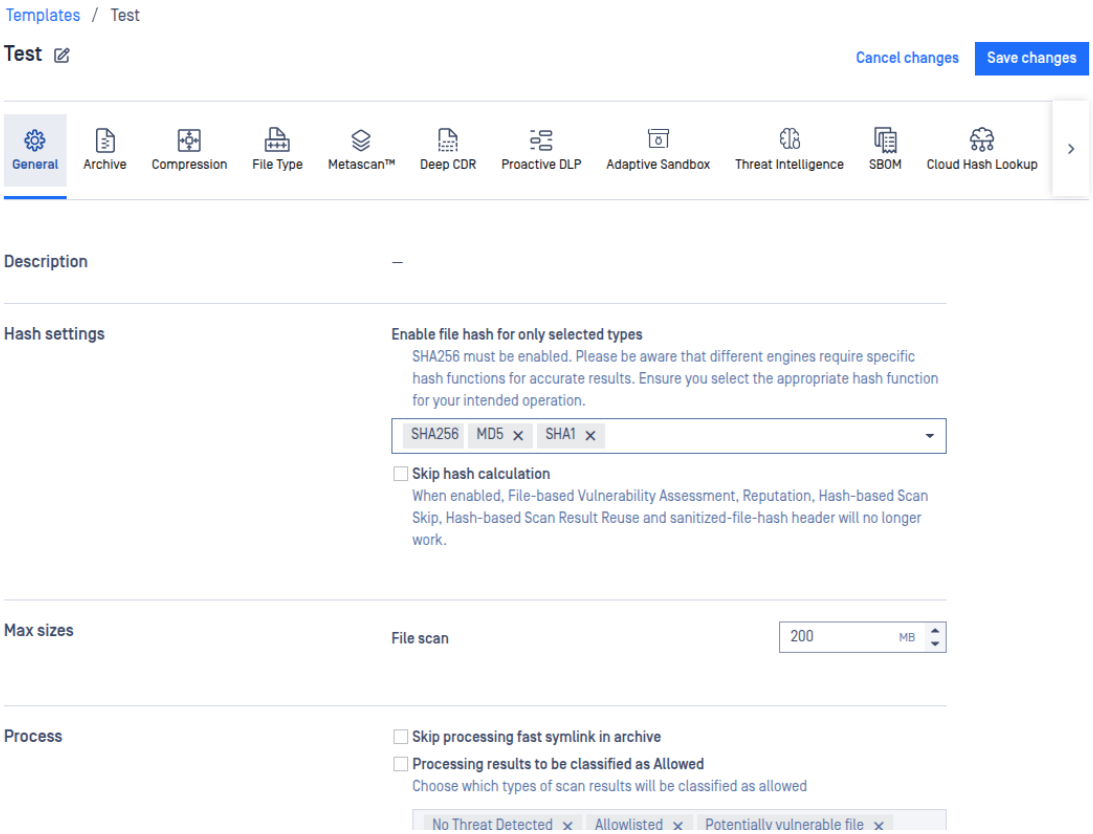
- This is only applicable to Asynchronous mode.
- When a scan request is timed out by this global processing timeout setting, the scan request will be blocked with "Global timeout exceeded" as blocked reason.
- **Note:** *This global processing timeout, when enabled, it should be greater or equal to the sum of all other engine related timeout settings such as Metascan timeout, Deep CDR timeout etc.*

By enabling '**OVERRIDE SCAN RESULTS CLASSIFIED AS ALLOWED**' it is possible to overwrite the default behavior of MetaDefender and determine which scan verdicts should result as allowed.

Scan results checked are marked as allowed.

By default only following verdicts result in allowed status:

- No Threat Detected
- Skipped Clean
- Potentially Vulnerable File
- Yara Rule Matched

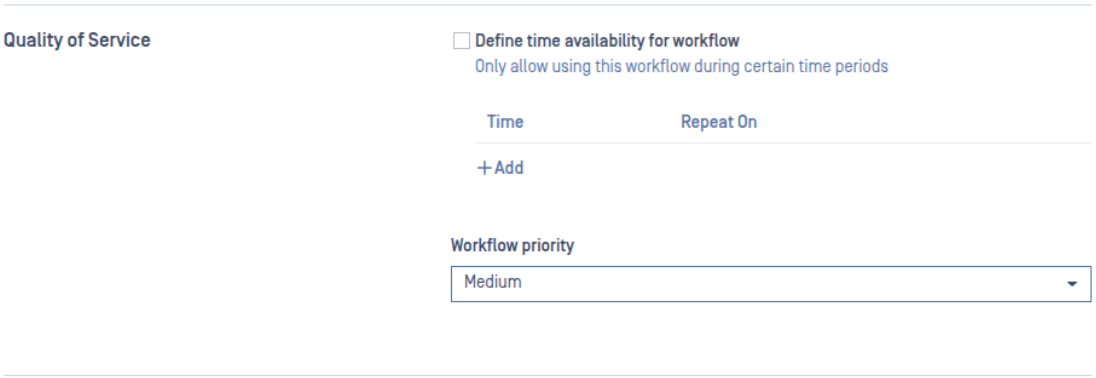


By enabling '**Define time availability for workflow**', it is possible for clients to define time frames in which a specific workflow is ready to use. MetaDefender Core only accepts requests of the workflow rule if they are within available time frames defined by the rule and responds HTTP code 403 otherwise.

By selecting one of values of '**Workflow priority**', it is possible for clients to prioritize a task coming to engine.

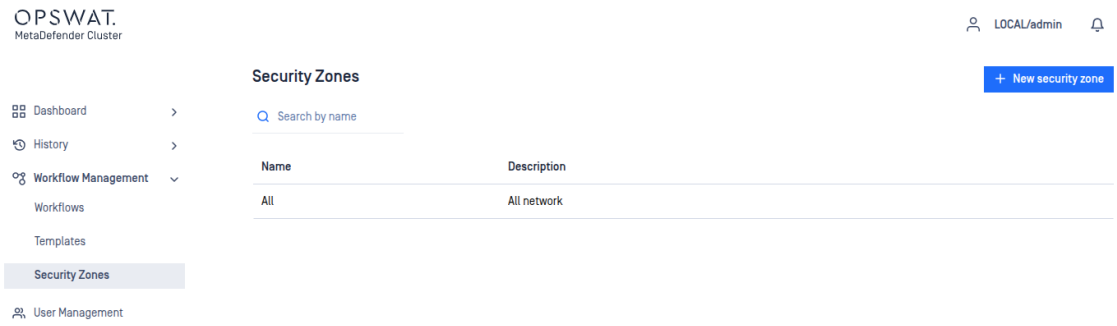
- If there are several tasks of the same priority waiting for an engine to process, the first comes, the first is processed by the engine.
- If tasks of different priorities are waiting for an engine to process, every two tasks of high priority processed by the engine will give a chance to one single task of next lower priority to be processed.

By default, priority of all tasks coming to an engine is Medium.



Security zone configuration

The Security zone page is found under `Workflow Management` > `Security Zones` after successful login.



The following actions are available:

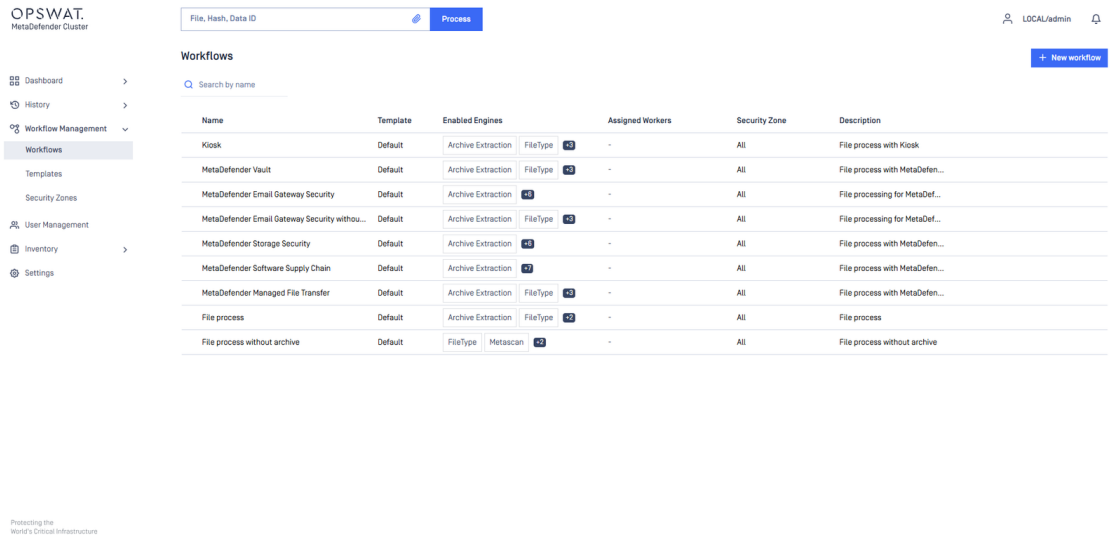
- New zones can be added
- Existing zones can be viewed
- Existing zones can be modified
- Existing zones can be deleted

Each zone contains a name, description and multiple network masks. Both IPv4 and IPv6 network zones are supported Workflow Configuration Template.

Workflow rule

The Workflow rule page is found under `Workflow Management`>`Workflows` after successful login.

The rules represent different processing profiles.



The following actions are available:

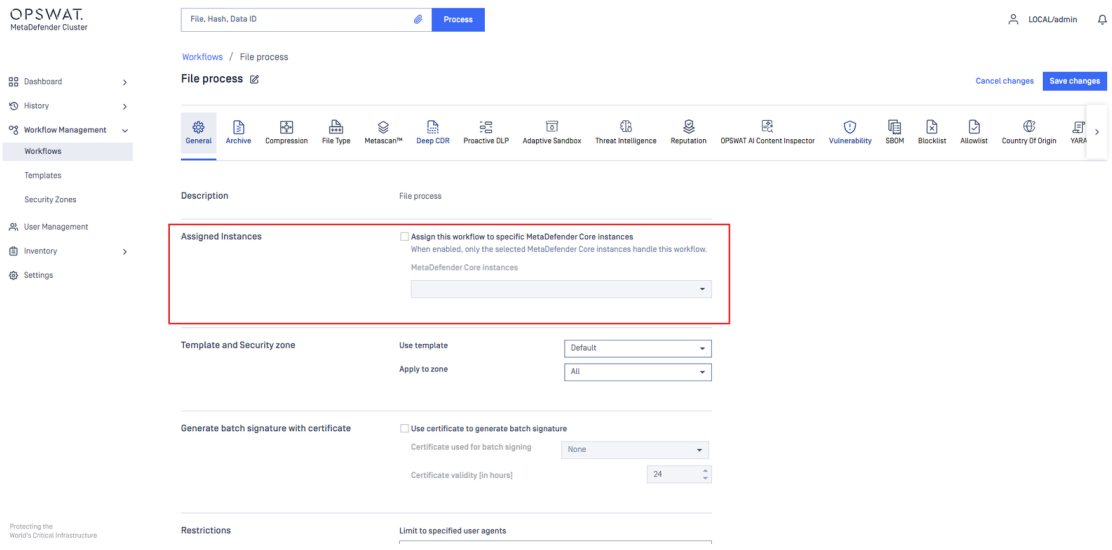
- New rules can be added
- Existing rules can be viewed
- Existing rules can be modified
- Existing rules can be deleted

Rules combine workflow templates and security zones and describe which workflows are available in a specified security zone. Multiple rules can be added for the same security zone.

Assigned Instances

By default, every MetaDefender Core instance in the cluster can process files for any workflow.

Enable **Assign this workflow to specific MetaDefender Core instances** on the **General** tab to reserve a workflow for a chosen set of instances. For example, to keep large archives on dedicated hardware, or to serve separate channels from separate instances inside a single cluster.



When the option is enabled, pick one or more instances from **MetaDefender Core instances**. Each instance is listed as `<display name> - N CPUs, R GB RAM`, so a workflow can be matched to instances of the right capacity. At least one instance must be selected.

- Files matched to this workflow are processed only by the selected instances. Instances that are not selected never receive them.
- While a selected instance has nothing of its own to process, it also handles workflows that have no assigned instances, so its capacity is not wasted.
- Workflows with the option disabled are unchanged: any MetaDefender Core instance can process them.
- The **Assigned Workers** column on the **Workflows** list shows the instances each workflow is assigned to.

The same assignment can be made from the instance's side — see **Inventory > Workers**

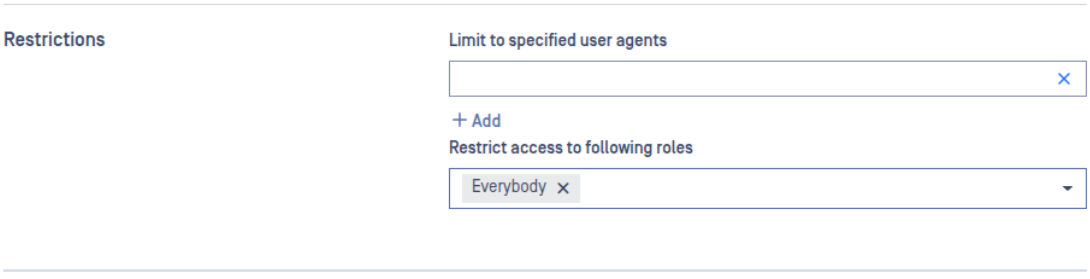
> Only MetaDefender Core instances can be assigned; the API Gateway and the Callback Service cannot. Assignment requires MetaDefender Core 5.22.0 or later. Earlier versions do not appear in the list.

> While files matched to the assigned workflow are still being processed, the option cannot be turned off and the workflow cannot be deleted. Wait for processing to finish and try again.

> In Kubernetes deployments, a MetaDefender Core instance loses its workflow assignments when it is terminated or redeployed. The workflows must then be reassigned to the instance.

Restriction

Processing result reports are generated dynamically based on the permissions defined in the selected Role settings.



Two special roles are also available:

- **Every authenticated** — applies to any logged-in user.
- **Everybody** — applies to all users, including unauthenticated users.

Users who are not assigned to any role specified in the rule cannot view scan results. Access can be further limited by using the **RESTRICT ACCESS TO FOLLOWING ROLES** setting.

You can also override workflow template settings for a specific rule by selecting the corresponding tab and changing individual properties. These changes apply only to the selected rule and do not modify the original workflow template.

This allows multiple rules to use the same workflow template while applying different custom settings. Any properties that are not overridden continue to use the values defined in the original template.

Rules are evaluated in order. MD Cluster uses the first rule that matches the client's source IP address. If no matching rule is found, the MD Cluster API Gateway denies the scan request.

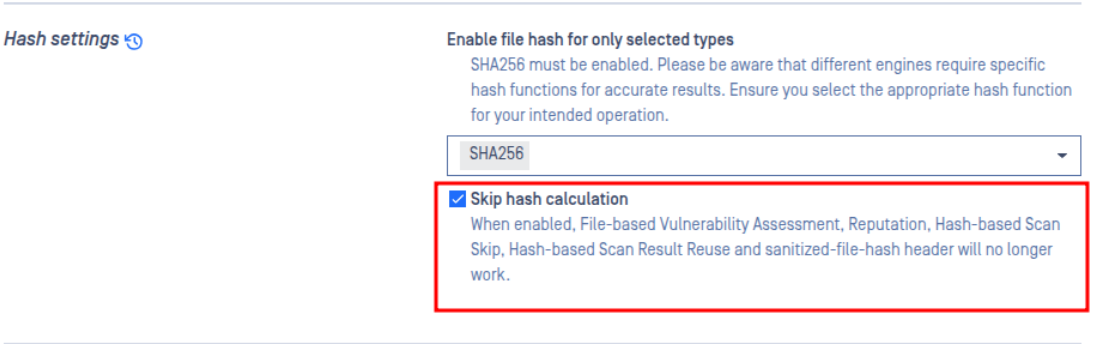
Skip hash calculation

Info

The **SKIP HASH CALCULATION** option can be enabled in the **General** tab of a workflow rule. This feature is disabled by default.

When enabled, MD Cluster skips calculating file hashes during processing. As a result, the following features become unavailable because they rely on file hash values:

- **File-based Vulnerability Assessment** — requires the file hash as input.
- **Reputation checks** — require the file hash as input.
- `sanitized-file-hash` **response header** — contains the SHA256 hash of the sanitized file.
- **Scan result reuse for identical files** — depends on the SHA256 hash to identify matching files.



This feature allows MD Cluster to skip hash calculation for all processed files, including files inside archives.

It is primarily intended for large file processing scenarios, where skipping hash generation can significantly reduce overall processing time.

When this option is enabled:

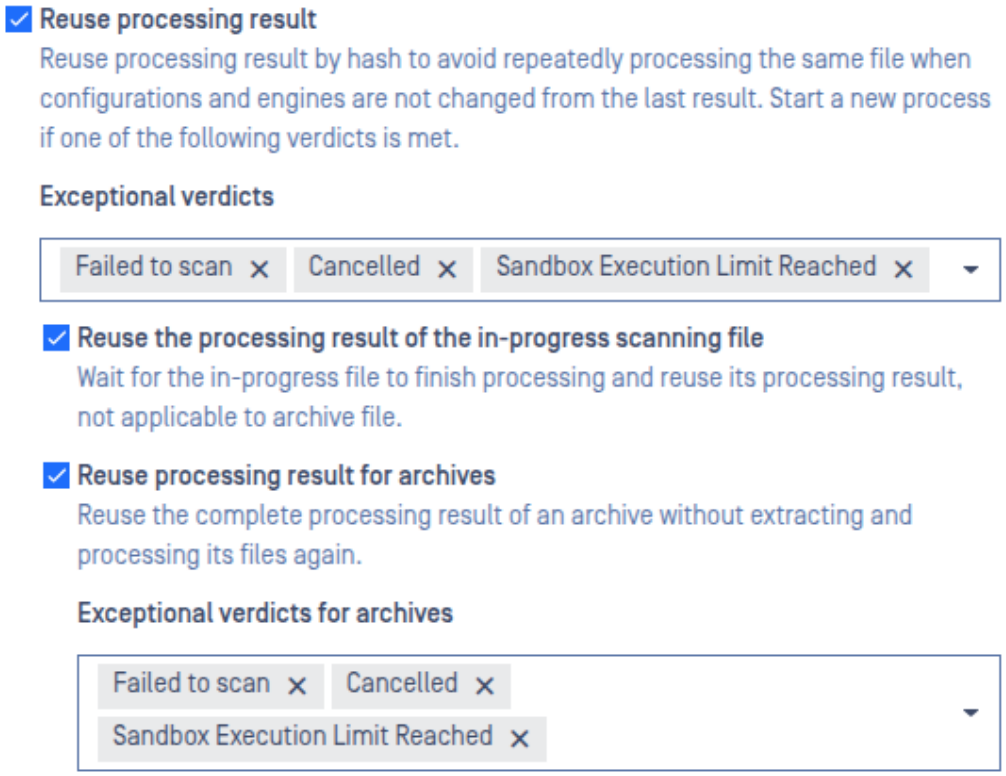
- The `md5`, `sha1`, `sha256`, and `sha512` fields in the JSON scan result are returned as empty values.
- The `process_info.skip_hash` field is set to `true`, which can be used by client integrations to identify that hash

calculation was skipped.

Reuse scan results for the same hash

The **Reuse Existing Results** option, available in the **General** section of a MD Cluster workflow rule, allows authorized users to automatically reuse eligible processing results for files with the same hash. This setting is disabled by default.

When enabled, MD Cluster can reuse results from an already processed file for other matching requests that are still in progress, helping reduce processing time and improve overall performance.



MD Cluster also allows customers to control when previously processed results can be reused. In some cases, customers may prefer to start a new scan instead of reusing an earlier result, such as when a previous scan ended with a verdict like **Failed to scan**.

A scan request is eligible to reuse the results of an earlier request with the same file hash only when all of the following conditions are met:

- **Reuse processing result** is enabled.
- **Skip hash calculation** is disabled.
- The earlier scan was submitted either by the same user or by an anonymous user.
- Both scan requests use the same workflow rule.
- The earlier scan verdict is not included in the workflow rule's configured "skip reuse" verdict list.
- The following request headers match the earlier request:

- `metadata`
- `engines-metadata`

- No scan engines or engine databases have been updated since the earlier request started.
- No engine configurations have changed since the earlier request started.
- No workflow configurations have changed since the earlier request started.
- If the earlier request produced a **Deep CDR**-sanitized file, that file should still be retained in the MD Cluster File Storage.
- If the earlier request produced a **Proactive DLP**-processed file, that file should still be retained in the MD Cluster File Storage.
- If **Discard processed files when the original file is blocked** is enabled and the processed files were removed, the previous result cannot be reused.
- The list of detected possible file types must match between the earlier and current requests.
- The earlier result is still available for reuse, either in the current instance's local reuse pool or in the Redis global reuse pool for up to 50 minutes.

By default, processing-result reuse applies to individual files. To reuse results for archive files as well, enable **Reuse Processing Result for Archives**. This can significantly reduce processing time when the same archive is submitted again.

When this option is enabled:

- You can separately configure which previous verdicts must not be reused for archives.
- Reuse also applies to nested archives. If a submitted archive contains a nested archive that was processed previously, the stored result for that nested archive can be reused while the remaining contents of the outer archive are processed normally.

In addition to the configured verdict exclusions and the general reuse conditions described above, an archive is re-scanned in the following cases:

Re-scanned Condition	Why the archive is re-scanned
The archive, or any file inside it, is **password-protected or encrypted**.	A stored result cannot currently validate the password provided with a new request. The archive must therefore be extracted and scanned again.
A stored archive result that exceeded the **expansion-factor limit** is reused as a nested archive inside another archive.	The expansion-factor limit depends on the size of the top-level file, so the result may be different when the archive appears in a different position.
The previous archive extraction **timed out**.	Extraction timeouts can depend on system load and may not occur again. For this reason, timeout results are treated as non-repeatable and the archive is scanned again.

Re-scanned Condition	Why the archive is re-scanned
A **skip-by-hash allowlist or blocklist entry** that affected a file inside the archive has **changed or expired**.	The stored decision for that file is no longer current, so the archive is scanned again using the latest allowlist and blocklist configuration.

Shared Archive Processing

MD Cluster can share archive processing tasks among multiple MD Core instances. Distributing the workload across the cluster helps optimize resource usage and improves performance when handling large archive files.

This feature is enabled by default and can be configured in the **General** section of an MD Cluster workflow rule.

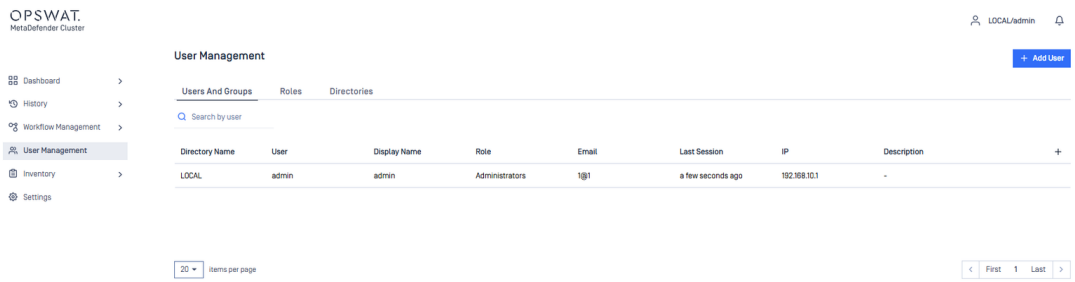


Archive processing includes both extraction and compression. When an archive file is processed, extraction tasks are distributed across available MD Core instances. If the **Compression Engine** is enabled, compression tasks are distributed in the same manner, allowing MD Cluster to process large archive files more efficiently and scale more effectively.

User Management

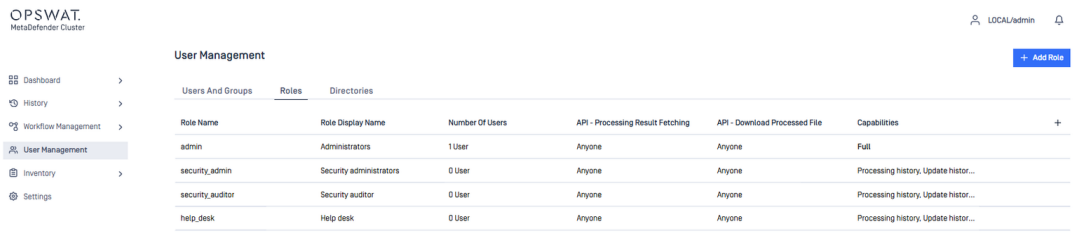
Users And Groups

This section lists the existing users for MetaDefender Cluster. A user can be added in this section and can be assigned a specific role.



Roles

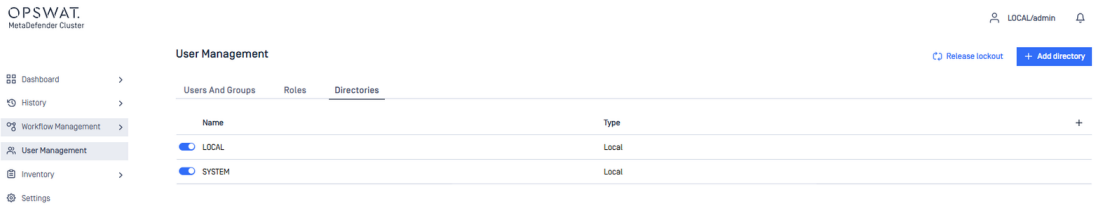
This section list the existing roles available for MetaDefender Cluster. Each role has specific permission and can be assigned to a user.



Directories

This section lists the existing directories for MetaDefender Cluster. Each directory has a specific permission that will enforce the following login policies:

- Number of failed logins before lockout.
- Lockout time (minutes).



User directory

User directories

User directories organize users into separate authentication domains and allow you to manage login behavior for each directory independently.

The **User directories** page lets you:

- Create Local, Active Directory, or LDAP user directories
- Configure login policies for Local user directories
- Enable or disable user directories
- Edit or remove existing user directories

Default user directories

After installation, MetaDefender Cluster (MD Cluster) Control Center creates the following local user directories by default.

User directory type	Name	Failed login attempts before lockout	Lockout duration (minutes)
Local	LOCAL	3	5

User directory type	Name	Failed login attempts before lockout	Lockout duration (minutes)
Local	SYSTEM	0	0

MD Cluster Control Center supports the following user directory types:

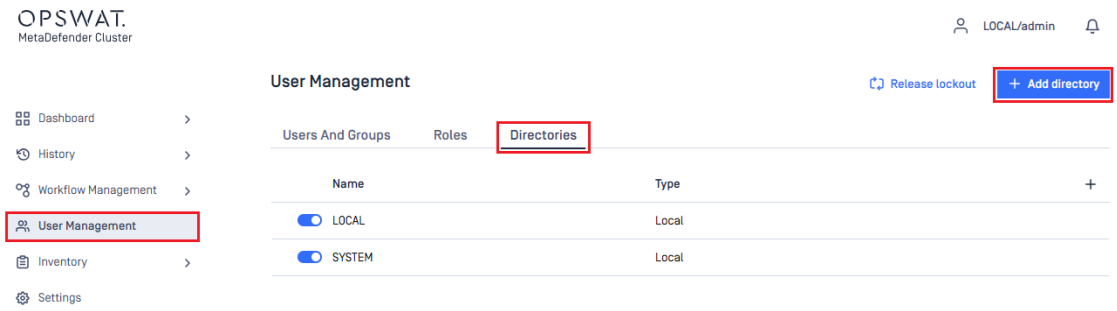
- Local
- Active Directory
- LDAP

Local user directories

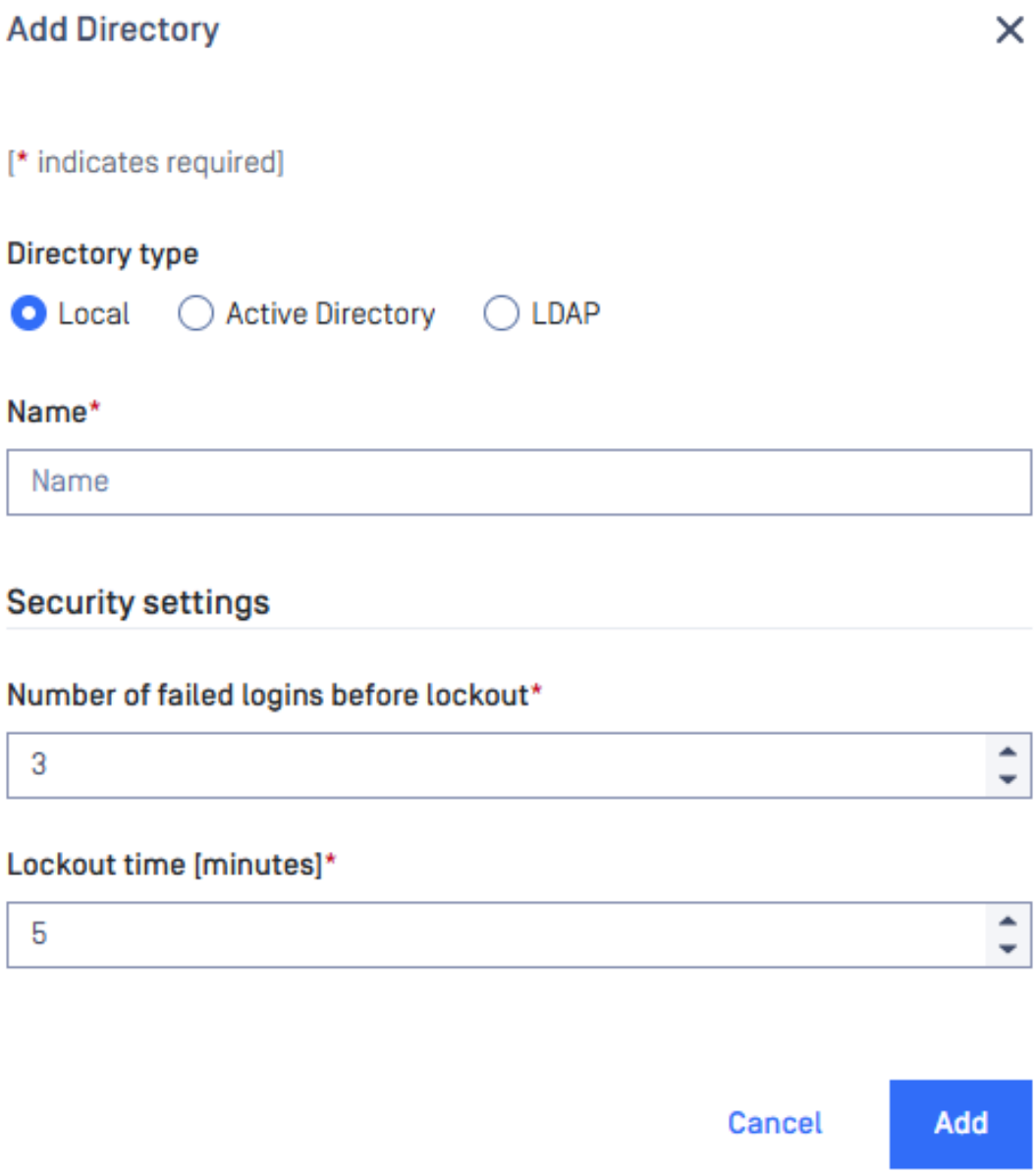
A Local user directory stores user accounts directly in MD Cluster Control Center and MD Cluster API Gateway.

Create a Local user directory

1. From the navigation pane, select **User Management**.
2. On the **Directories** tab, click **Add Directory**.



3. Select **Local** as the **User Directory Type**.
4. Enter a name for the new user directory.
5. Configure the security settings as needed.
6. Click **Add** to create the user directory.



Security settings

- **Failed login attempts before lockout** – The number of consecutive failed login attempts allowed before an account is locked.
- **Lockout duration (minutes)** – How long the account remains locked.
- The account is automatically unlocked when the lockout period expires.
- Administrators with the appropriate permissions can manually unlock an account before the timeout expires by clicking **RELEASE LOCKOUT**.
- **Enable enhanced password policy** – Enables the enhanced password policy for users in the directory. For more information, see Password Policy.

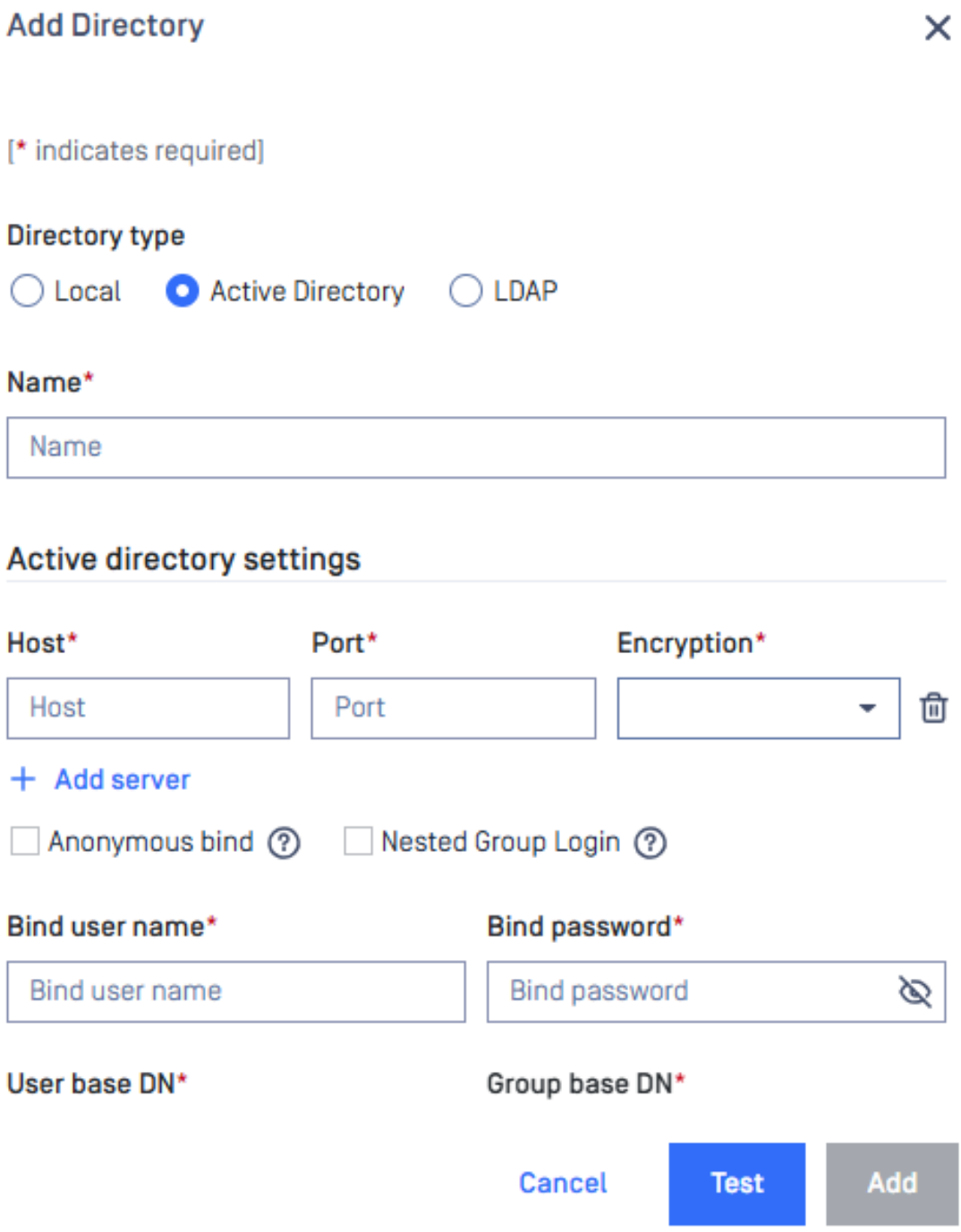
Active Directory user directories

An Active Directory user directory allows users from Microsoft Active Directory to authenticate to MD Cluster Control Center.

Authentication policies such as account lockout and password requirements are managed by Active Directory rather than MD Cluster.

Configure an Active Directory user directory

1. From the navigation pane, select **User Management**.
2. On the **Directories** tab, click **Add Directory**.
3. Select **Active Directory** as the **User Directory Type**.
4. Enter a name for the new user directory.
5. Configure the required Active Directory connection settings.
6. Click **Test** to verify the connection.
7. If the connection test succeeds, click **Add** to create the user directory.



Active directory settings

Configure one or more Active Directory servers by providing the following information:

- **Host** – Hostname or IP address of the Active Directory server.
- **Port** – Port used to connect to the Active Directory server.
- **Encryption** – Connection security protocol. Select one of the available options:
  - **None** – Connect without encryption.
  - **StartTLS** – Upgrade an LDAP connection to use TLS.

  - **SSL** – Establish an encrypted connection using LDAPS.

- **Anonymous Bind** – Allows MD Cluster Control Center to connect without providing bind credentials. Enable this only if your Active Directory server permits anonymous directory searches.
- **Nested Group Login** – Enables authentication for users who belong to nested Active Directory groups.
- **Bind User Name** – Distinguished Name (DN) of the account used to search the Active Directory.
- **Bind Password** – Password for the bind account.
- **User Base DN** – Distinguished Name (DN) that defines where MD Cluster Control Center begins searching for user accounts.
- **Group Base DN** – Distinguished Name (DN) that defines where MD Cluster Control Center begins searching for groups.

For guidance on selecting the appropriate Base DN values, see **Active Directory/LDAP attributes**.

> **Info**
> Use the most specific **User Base DN** and **Group Base DN** possible. Limiting the search scope improves authentication performance and reduces the load on the Active Directory server.

> **Info**
> Whenever possible, select **StartTLS** or **SSL** for **Encryption** instead of **None**. Also ensure that the certificate authority (CA) used to issue the Active Directory server certificate is trusted by the MD Cluster Control Center server.

LDAP user directories

An LDAP user directory allows users stored in an LDAP directory service to authenticate to MD Cluster Control Center.

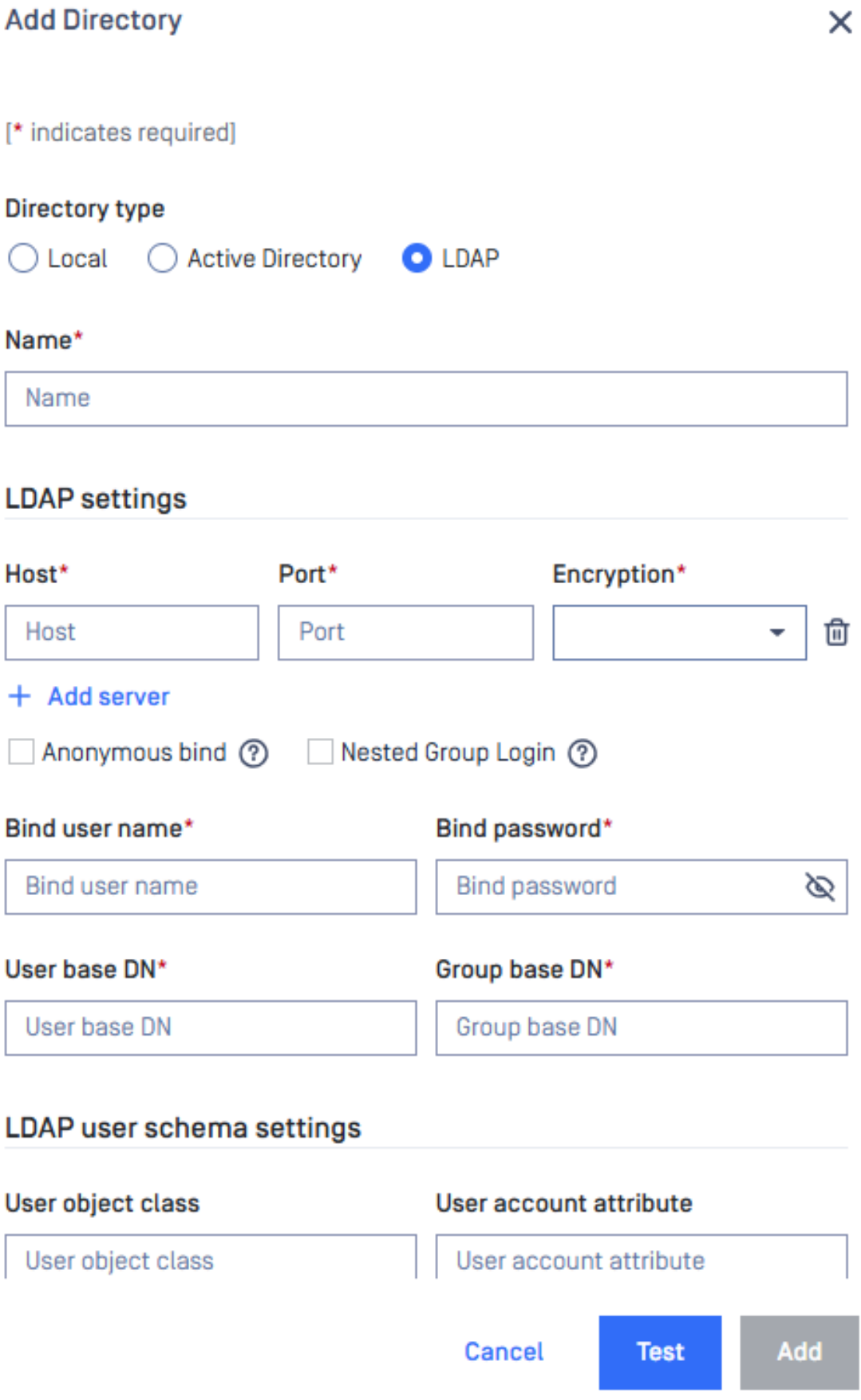
Authentication policies are managed by the LDAP server rather than MD Cluster Control Center.

Configure an Active Directory user directory

1. From the navigation pane, select **User Management**.

2.

On the **Directories** tab, click **Add Directory**. 3. Select **LDAP** as the **User Directory Type**. 4. Enter a name for the new user directory. 5. Configure the required Active Directory connection settings. 6. Click **Test** to verify the connection. 7. If the connection test succeeds, click **Add** to create the user directory.


Add Directory
[* indicates required]
Directory type
Local
Active Directory
LDAP
Name*
Name
LDAP settings
Host*
Port*
Encryption*
Host
Port
Add server
Anonymous bind
Nested Group Login
Bind user name*
Bind password*
Bind user name
Bind password
User base DN*
Group base DN*
User base DN
Group base DN
LDAP user schema settings
User object class
User account attribute
User object class
User account attribute
Cancel
Test
Add

LDAP directory settings

Configure one or more LDAP servers by providing the following information:

- **Host** – Hostname or IP address of the Active Directory server.
- **Port** – Port used to connect to the Active Directory server.
- **Encryption** – Connection security protocol. Select one of the available options:
  - **None** – Connect without encryption.
  - **StartTLS** – Upgrade an LDAP connection to use TLS.
  - **SSL** – Establish an encrypted connection using LDAPS.
- **Anonymous Bind** – Allows MD Cluster Control Center to connect without providing bind credentials. Enable this only if your Active Directory server permits anonymous directory searches.
- **Nested Group Login** – Enables authentication for users who belong to nested Active Directory groups.
- **Bind User Name** – Distinguished Name (DN) of the account used to search the Active Directory.
- **Bind Password** – Password for the bind account.
- **User Base DN** – Distinguished Name (DN) that defines where MD Cluster Control Center begins searching for user accounts.
- **Group Base DN** – Distinguished Name (DN) that defines where MD Cluster Control Center begins searching for groups.

Info

Use the most specific **User Base DN** and **Group Base DN** possible. Limiting the search scope improves authentication performance and reduces the load on the Active Directory server.

Info

Whenever possible, select **StartTLS** or **SSL** for **Encryption** instead of **None**. Also ensure that the certificate authority (CA) used to issue the Active Directory server certificate is trusted by the MD Cluster Control Center server.

Limitations

For security reasons, MetaDefender Cluster prevents administrators from performing the following actions:

- Disable the user directory that contains their own account.
- Delete their own user account.
- Delete the user directory that contains their own account.

For example, the built-in **admin** user cannot disable or delete the **LOCAL** user directory because the account belongs to that directory.

Settings

Additional MetaDefender Cluster Control Control settings can be found in this section. This ranges from Data Retention, Generating Support Packages, Password and Session policies, Module Updates and Health Checks.

- Security.
- Module update
- Data Retention.
- Health check.
- Export.

Security

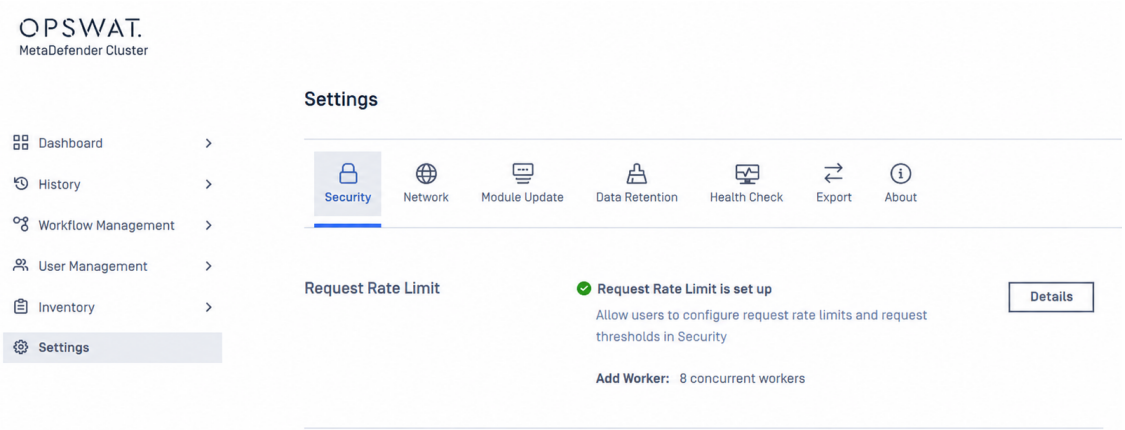
This section will allow the system administrator to enable secure connections to MetaDefender Cluster Control Center if required. In addition, Request rate limit, Password and Session policies are set in this section.

Request Rate Limit

The **Request Rate Limit** feature helps protect MD Cluster Control Center from excessive administrative requests by limiting the number of concurrent operations the system accepts. This prevents bursts of requests from overwhelming system resources and helps maintain stable performance.

Click **Details** to configure request rate limits for supported administrative operations. Each operation can be enabled or disabled independently, and its maximum number of concurrent requests can be adjusted to match your deployment requirements.

After you save the configuration, the new request rate limits may take up to **30 seconds** to take effect.



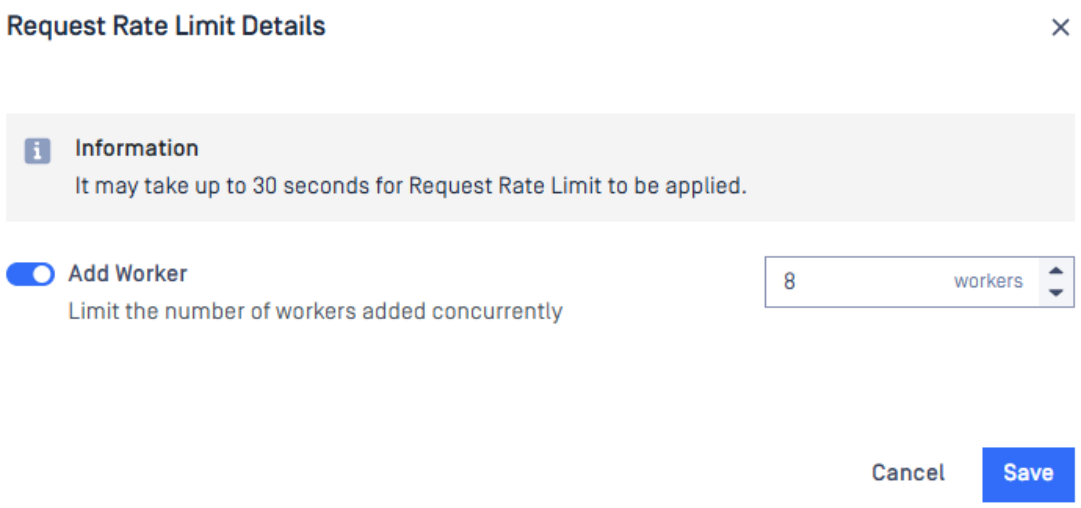
Add Worker

Limits the number of **Add Worker** operations that can run concurrently.

When enabled, specify the maximum number of Worker registration requests that MD Cluster Control Center can

process at the same time. Requests that exceed the configured limit are rejected, and users are prompted to try again later.

The updated limit may take up to **30 seconds** to take effect after it is saved.



Password Policy

> **Info**
>
> These password policies changes only apply to new user creations and future password changes. Existing users' passwords are unaffected.

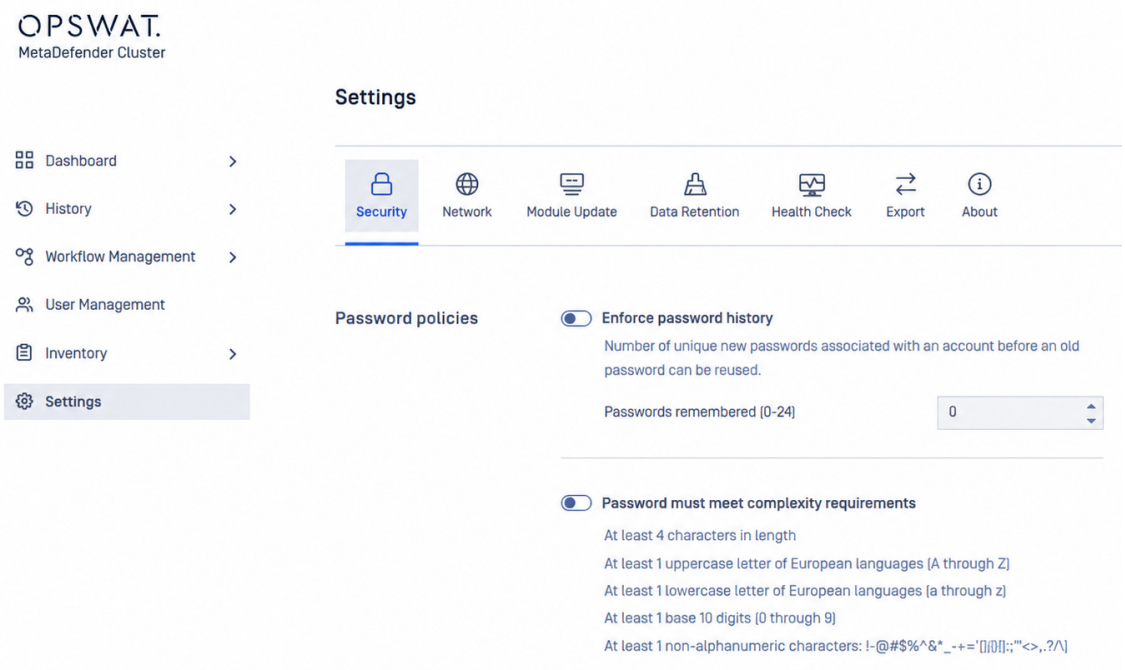
Local users' password can be enforced to meet requirements set by administrators, which includes following constraints:

- **Enforce password policy:**
  - Determines the number of unique new passwords that must be associated with a user account before an old password can be reused.
  - Range: [0-24].
  - Default: 0 (to disable enforcement).
- **Password must meet complexity requirements:**
  - Determines whether passwords must meet a series of guidelines that are considered important for a strong password.
  - Default: unchecked

> **Complexity requirements**
>
> - At least 4 characters in length.

- At least 1 uppercase letter of European languages (A through Z).
- At least 1 lowercase letter of European languages (a through z).
- At least 1 base 10 digits (0 through 9).
- At least 1 non-alphanumeric characters (special characters): (~!@#$%^&*_-+=`|(){}[]:;'"<>,.?/).

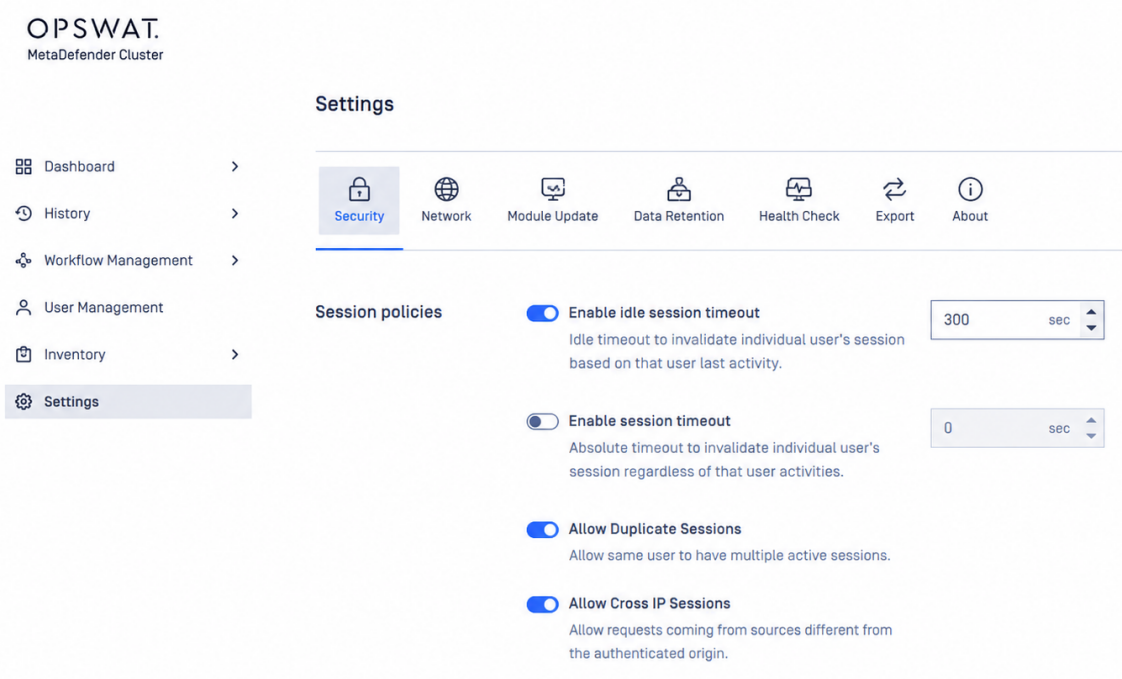
- **Minimum password length:**
  - The least number of characters that can make up a password for a user account.
  - Range: [0-30].
  - Default: 0 (to disable enforcement).



Session Policy

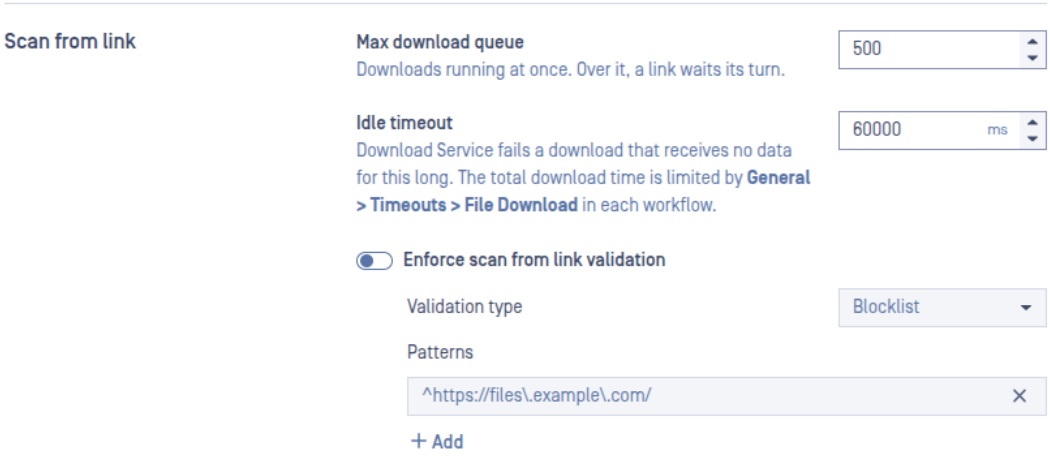
Session policies control how user sessions are created and maintained in MD Cluster. You can configure session expiration, allow or prevent multiple concurrent sessions for the same user, and control whether requests from different IP addresses are accepted within an authenticated session.

Changes take effect for new sessions. Existing sessions continue to use the settings that were in effect when they were established.



- **Idle session timeout**: Idle timeout to invalidate individual user's session based on that user last activity.
- **Session timeout**: Absolute timeout to invalidate individual user's session regardless of that user activities.
- **Allow Duplicate Sessions**: Allow same user to have multiple active sessions.
- **Allow Cross IP Sessions**: Allow requests coming from sources different from the authenticated origin.

Scan from link



These settings control how MetaDefender Cluster downloads files that are submitted by link (the `downloadfrom` header of `POST /file`). They apply to the whole cluster.

- **Max download queue**: the number of downloads that can run at the same time.
  - When the limit is reached, a new link is **not** refused. It waits in the queue and starts when a running download finishes.
  - Use a lower value to limit the network and disk load on the Download Service. Use a higher value to download

more files in parallel.

- **Idle timeout**: How long a download can go without receiving any data before it stops.
  - The value is in **milliseconds**.
  - The timer restarts every time data arrives. A slow download that still receives data does not stop.
  - When the timeout is reached, the download result is `Download Failed` with the error detail `Server took too long to respond`.

> The idle timeout does not limit the total download time. The total time is limited by the workflow setting **General > Timeouts > File download**. When that limit is reached, the result is Download Timeout.

- **Enforce scan from link validation**: Turn this on to allow or block links by pattern. When it is off, every link is downloaded, and the validation type and patterns are ignored.

Type	A link is downloaded when
Blocklist	It matches none of the patterns.
Allowlist	It matches at least one pattern.

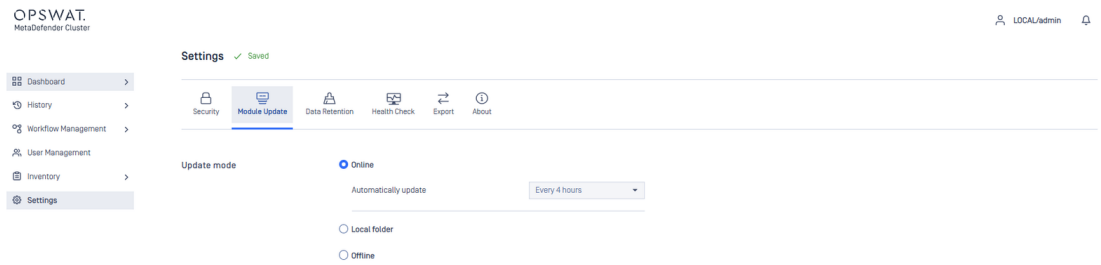
- **Patterns**: Each pattern is a regular expression.

Module update

This section will allow the system administrator to select one of three update modes offered by MetaDefender Cluster (MD Cluster).

Online

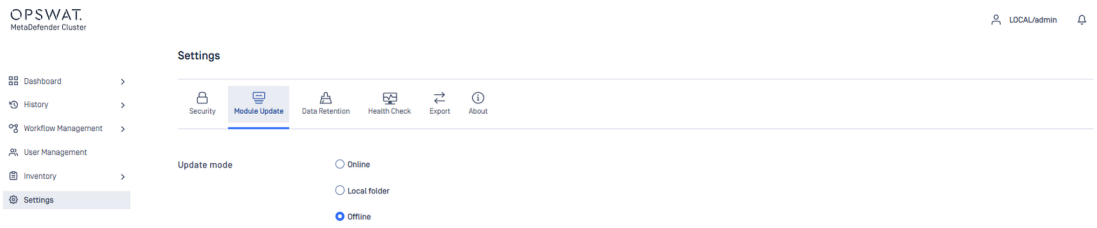
In this mode, MD Cluster **Control Center** will autonomously download the latest module packages from OPSWAT online update infrastructure, repeating this process every four hours by default.



Offline

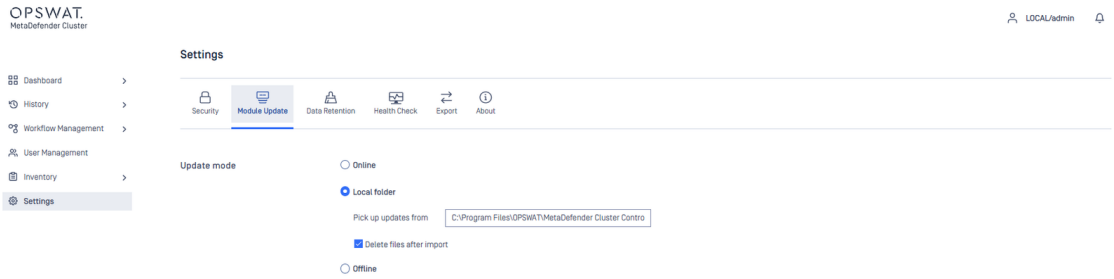
In this mode, the administrators must download the licensed engine packages from **MetaDefender Update Downloader** and upload them manually to MD Cluster **Control Center**.

Refer to the steps for manually uploading of the packages in MD Cluster **Control Center**.



Local folder

In this mode, the administrators need to input the path to a folder where module packages will be added. MD Cluster **Control Center** subsequently gathers the packages from there and starts the module update process.



`Delete files after import` option can be selected so that MD Cluster **Control Center** can wipe all packages upon success.

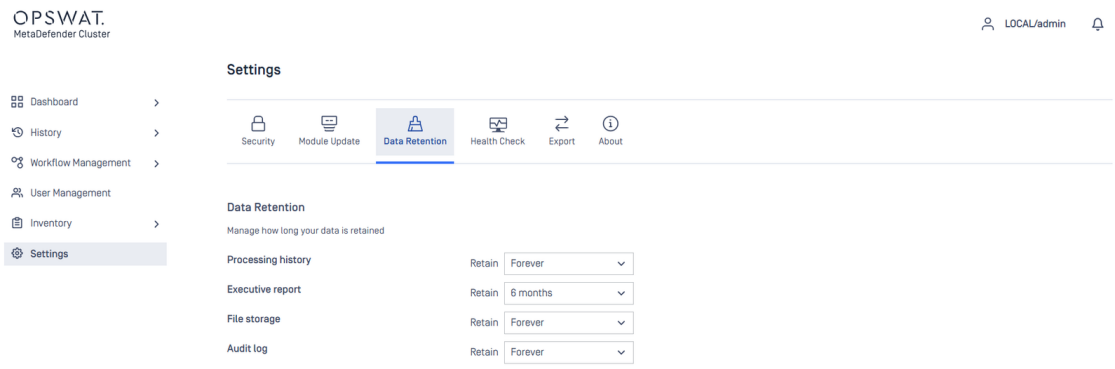
Data retention

This setting enables users to define the retention period for specific data types, helping optimize system storage and maintain efficiency.

Available Data Categories

1. **Processing History:** History of scan results.
2. **Executive Report:** Statistical summaries and insights.
3. **File Storage**: System-generated files (e.g., sanitized or watermarked versions).
4. **Audit Log:** Detailed logs of user activities and system events.

In case you do not want to enable automatic clean up, set the value to off. This will prevent automatic removal.

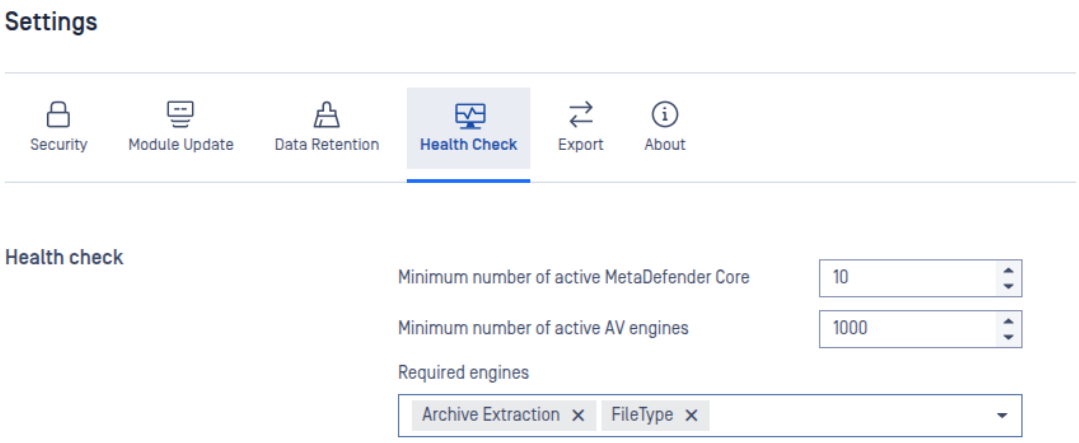


Warning

Disabling automatic clean-up may lead to data accumulation, which can affect system performance and increase storage costs.

Health check

The **Health Check** configuration page in OPSWAT MetaDefender Cluster allows administrators to define the minimum operational thresholds required for the system to be considered "**healthy**" and ready to process traffic from MD API Gateway for submitting files to scanning. By setting these parameters, you ensure that the system maintains your required baseline before accepting and scanning files.



Configuration Parameters

- **Minimum number of active MetaDefender Core:** The minimum number of MD Core servers that must be healthy. (Minimum value: 1)
- **Minimum number of active AV engines:** The minimum total AV engines that must be operational across MD Core instances. (Minimum value: 1)
- **Required engines:** Specific mandatory engines (e.g., `FileType`, `Archive Extraction`, `AhnLab`) that **must** be active, regardless of the overall engine count.

System Behavior on Failure

If any of the above conditions are not met, MD Cluster is considered unhealthy. As a result, the **MD API Gateway will return an error** to the client if they attempt to submit a file for scanning.

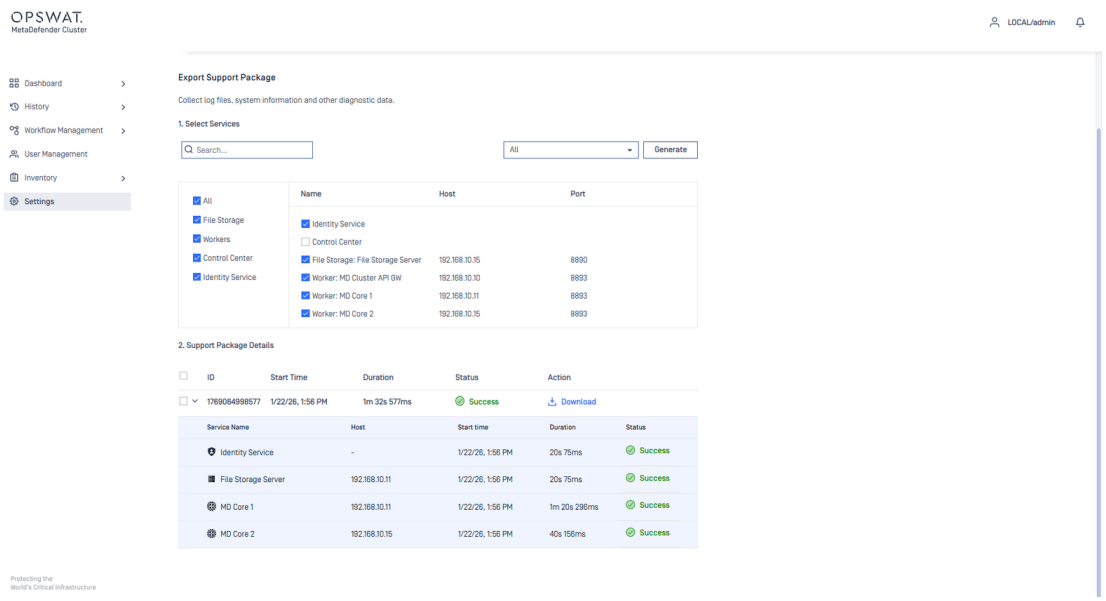
> **Info**
>
> The MD API Gateway may receive a health status signal later than expected due to the polling intervals between scheduled health checks.

Export

This section will allow the system administrator to export MetaDefender Cluster (MD Cluster) support packages. The support package contains all relevant information to help us diagnose the issue of MetaDefender Cluster.

> **Warning**
>
> MD Cluster Support Package does not include logs from services like Redis, RabbitMQ and PostgreSQL. To obtain the service logs, please click here.

- Simply select all to select all MD Cluster components and click the `Generate` button. A new ID and notification will be shown in the `Support Package Details.`



- Once the support package generation is finished, simply select `Download` in the `Action` column in the `Support Package Details.`

System Upgrade

Upgrading MetaDefender Cluster (MD Cluster) requires careful planning because not all services affect the system in the same way.

Some components can be upgraded while file scanning continues normally, while others are critical infrastructure services that affect authentication, queue management, metadata handling, and file availability across the entire cluster.

To reduce operational risk, the upgrade procedure is divided into two stages:

Stage	Description	Downtime Impact
Live Upgrade	Upgrades application-level services while the cluster continues processing files	Minimal or no disruption
Critical Service Upgrade	Upgrades shared infrastructure services that the cluster depends on	Maintenance window required

Before You Begin

Before starting the upgrade:

- Verify backups are available for:
  - PostgreSQL
  - MD Cluster File Storage
- Confirm all cluster nodes are healthy
- Review available disk space and system resources
- Notify users about potential service interruptions
- Schedule a maintenance window for Stage 2 upgrades

For production deployments, OPSWAT strongly recommends validating the upgrade procedure in a staging environment first.

Why the Upgrade Is Split Into Two Stages

MD Cluster includes both application services and shared infrastructure services. Application services can typically be upgraded with minimal impact to running scans. Critical infrastructure services are different.

Services such as:

- PostgreSQL
- Redis
- File Storage
- Identity Service

are shared dependencies used throughout the cluster. These services support:

- Authentication and authorization
- Queue coordination
- Metadata storage
- Workflow synchronization
- File access and retrieval
- Internal service communication

Because these components affect the entire cluster, they require additional care and should only be upgraded during a planned maintenance window.

Stage 1 — Live Upgrade

This stage upgrades services that can be updated while file scanning continues.

Although some management functionality may be temporarily unavailable, active file processing is not interrupted.

Services included:

- MD Cluster Identity Service
- MD Cluster Control Center
- MD Cluster Worker
- MD Core
- MD Cluster API Gateway
- MD Cluster Callback Service

---

1. Upgrade MD Cluster Identity Service

The MD Cluster Identity Service controls authentication and API authorization throughout the cluster.

Expected Impact:

- API authentication becomes temporarily unavailable.
- New authenticated requests cannot be submitted.

- Existing running scans continue processing.

Upgrade Procedure:

1. Download the MD Cluster Identity Service installer package from My OPSWAT.

2. Run the appropriate command for your operating system.

Code · bash

```
# Windows
> msiexec.exe /i <md_cluster_identity_package>.msi /qn

# Debian/Ubuntu
$ sudo dpkg -i <md_cluster_identity_package> || sudo apt install -f

# Rocky/RHEL
$ sudo dnf install <md_cluster_identity_package> -y
```

3. Verify the service status.

Code · bash

```
# Windows - PowerShell
> Get-Service md-cluster-identity-service

# Debian/Ubuntu/Rocky/RHEL
$ sudo systemctl status md-cluster-identity-service
```

---

2. Upgrade MD Cluster Control Center

The MD Cluster Control Center upgrade only affects the management interface.

Expected Impact:

- The web console may be temporarily unavailable.
- File scanning and processing continue normally.

Upgrade Procedure:

1. Download the MD Cluster Control Center installer package from My OPSWAT.

2. Run the appropriate command for your operating system.

Code · bash

```
# Windows
msiexec.exe /i <md_cluster_control_center_package>.msi /qn
```

```
# Debian/Ubuntu
sudo dpkg -i <md_cluster_control_center_package> || sudo apt install -f

# Rocky/RHEL
sudo dnf install <md_cluster_control_center_package> -y
```

3. Verify the service status.

Code · bash

```
# Windows PowerShell
Get-Service md-cluster-control-center

# Linux
sudo systemctl status md-cluster-control-center
```

---

3. Upgrade MD Cluster Worker

MD Cluster Worker can be upgraded without interrupting active file scans.

Expected Impact:

The following worker-related features may be temporarily unavailable during the upgrade:

- System Health
- System Activity
- Workflow Synchronization
- Support Package generation
- Automated deployment operations

Upgrade Procedure:

1. Sign in to MD Cluster Control Center console using your administrator account.
2. Go to `Inventory` > `Workers` .
3. In the notification banner, click `Upgrade Now`, then confirm the upgrade action.
4. Monitor the upgrade progress and verify that all Workers are running the same latest version.

See Upgrade Workers for more details.

---

4. Upgrade Execution Instances

This step includes upgrading:

- MD Core
- MD Cluster API Gateway
- MD Cluster Callback Service

Expected Impact:

- MD Core instances can be upgraded gradually using a rolling upgrade.
- Single-instance MD Cluster API Gateway or Callback Service deployments may experience a brief interruption.

Upgrade Procedure:

1. Download the MD Core, MD Cluster API Gateway or MD Cluster Callback Service installer packages from My OPSWAT.
2. Sign in to the Control Center console using your administrator account.
3. Navigate to `Inventory`>`Installers`.
4. Click `Upload Package` and upload the installer packages. For detailed upload instructions, see here.
5. Go to `Inventory`>`Workers`.
6. Expand `Deploy Workers` and select `Upgrade`.
7. Select the appropriate version of MD Core or MD Cluster API Gateway or MD Cluster Callback Service, then click `Upgrade` to start the process. See Upgrade Instances for more information.

Stage 2 — Critical Infrastructure Upgrade

This stage upgrades services that the entire cluster depends on.

These services must be upgraded carefully because interruptions can affect cluster-wide functionality, including authentication, queue processing, metadata operations, and file availability.

A maintenance window is required before continuing.

The following services are considered critical infrastructure components within MD Cluster:

Service	Purpose
PostgreSQL	Stores cluster configuration, operational state, processing result and executive data
RabbitMQ	Coordinate tasks between components
Redis	Handles caching, and internal coordination
MD Cluster File Storage	Stores files, scan artifacts, and deployment packages

Before Upgrading

We recommend completing the following checks before proceeding:

- Stop submitting new scan requests
- Allow all in-progress scans to complete
- Confirm the cluster is idle
- Ensure PostgreSQL, RabbitMQ, Redis, and MD Cluster File Storage are healthy before proceeding

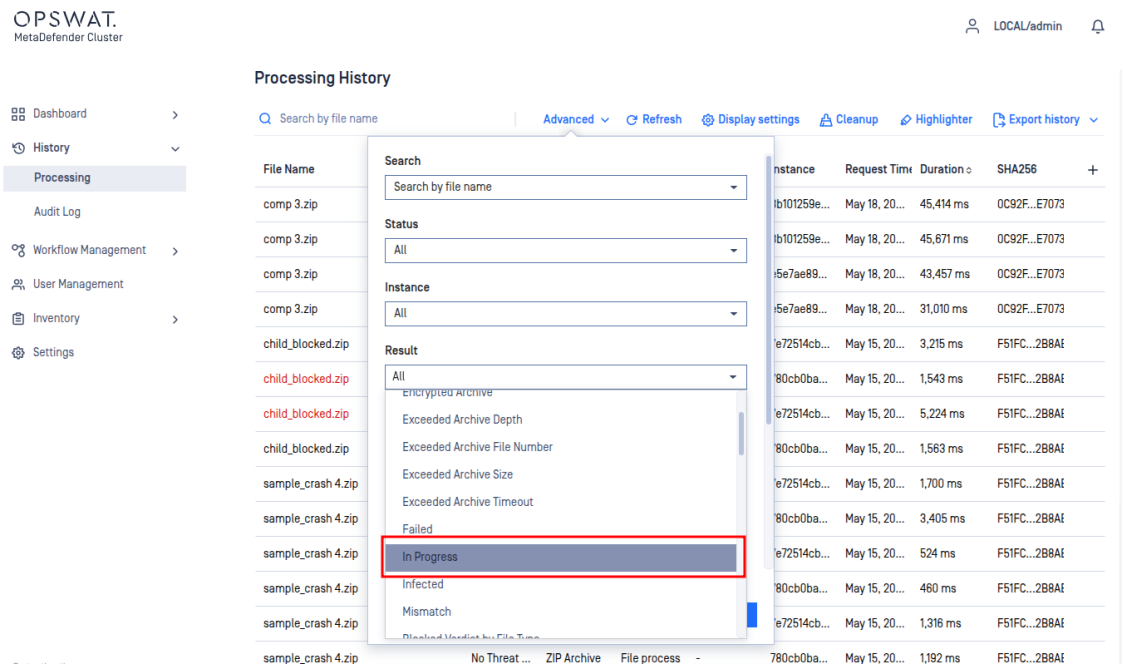
> **Info**
>
> Interrupting critical infrastructure services while scans are still processing may lead to failed requests, incomplete workflows, temporary data inconsistency, or service interruptions across the cluster.

---

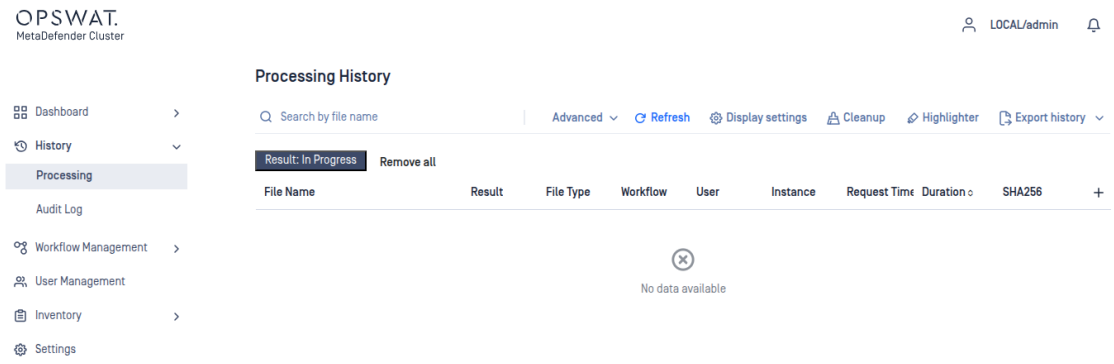
1. Verify That Processing Has Completed

Before starting Stage 2:

1. Log in to the Control Center web console.
2. Navigate to `Dashboard` > `Processing History` .
3. Filter requests by **In-Progress.**



4. Wait until all requests are completed.



> **Warning**
>
> Only continue after the cluster is no longer processing files.

---

2. Upgrade MD Cluster File Storage

MD Cluster File Storage is one of the most critical services in MD Cluster.

Most cluster operations depend on MD Cluster File Storage, including:

- File scanning
- File retrieval
- Module distribution
- Instance deployment

Because of this dependency, MD Cluster File Storage should always be upgraded last.

Expected Impact

- File processing may become temporarily unavailable
- Scan operations may pause
- Internal services depending on shared storage may be interrupted

Upgrade Procedure

1. Download the MD Cluster File Storage installer package from My OPSWAT.
2. Run the appropriate command for your operating system.

Code · bash

```
# Windows
msiexec.exe /i <md_cluster_file_storage_package>.msi /qn

# Debian/Ubuntu
```

```
sudo dpkg -i <md_cluster_file_storage_package> || sudo apt install -f

# Rocky/RHEL
sudo dnf install <md_cluster_file_storage_package> -y
```

3. Verify the service status.

Code · bash

```
# Windows PowerShell
Get-Service md-cluster-file-storage

# Linux
sudo systemctl status md-cluster-file-storage
```

3. Upgrade Lake and Warehouse

1. On the machine hosting MD Cluster **Control Center**, navigate to the folder:

Code · bash

```
# Windows
cd 'C:\Program Files\OPSWAT\MetaDefender Cluster Control Center'

# Linux
cd /usr/sbin
```

2. Run the following command to upgrade **Data Lake**:

Code · bash

```
md-cluster-dbready --host=<postgres-host> --port=<postgres-port> --user=<postgres-user> --
password=<postgres-password> --target=lake
```

3. Run the following command to upgrade **Data Warehouse**:

Code · bash

```
md-cluster-dbready --host=<postgres-host> --port=<postgres-port> --user=<postgres-user> --
password=<postgres-password> --lake-host=<lake-postgres-host> --lake-port=<lake-postgres-port>
--lake-user=<lake-postgres-user> --lake-password=<lake-postgres-password> --target=warehouse
```

Info

If the Data Lake and Data Warehouse are hosted on the same PostgreSQL instance, a combined setup

command can be used:

```
md-cluster-dbready --host=<postgres-host> --port=<postgres-port> --user=<postgres-user> --password=<postgres-password> --target=lake,warehouse
```

After the Upgrade

After completing all upgrade stages:

- Verify all services are healthy
- Confirm workers reconnect successfully
- Run test scan requests
- Validate API accessibility
- Review cluster logs for warnings or migration failures

Best Practices

- Perform upgrades during low-traffic periods
- Always back up PostgreSQL, Redis, RabbitMQ and MD Cluster File Storage before maintenance
- Avoid leaving the cluster in a mixed-version state
- Upgrade critical infrastructure services carefully and sequentially
- Monitor cluster health throughout the upgrade process

Info

Critical infrastructure services are foundational to cluster stability. Careful planning and controlled upgrades help prevent extended outages and operational inconsistencies.

Performance and Load Estimation

Disclaimer

These results should be viewed as guidelines and not performance guarantees, since there are many variables that affect performance (file set, network configurations, hardware characteristics, etc.). If throughput is important to your implementation, OPSWAT recommends site-specific benchmarking before implementing a production solution.

Factors that affect performance

- MetaDefender Cluster version
- MetaDefender Core version
- MetaDefender Cluster API Gateway, and File Storage version
- Workflow settings
- System environment (CPU, RAM, hard disk, and network bandwidth)
- Dataset
  - Encrypted or not
  - File types
    - Different file types (e.g., document, image, executable)
    - Archive file or compound document format files
  - File size
- Client tool (the tool submits files to the system)

Performance metrics

While processing files on the system, service performance is measured by various metrics. Some of them are commonly used to define performance levels, including:

Performance metrics	Description
Processed objects per hour	Total individual files that the system has to process. These could be separate **original files** submitted to the system or **extracted files inside an archive**. The number of processed objects is considered to be a more accurate throughput metric for measuring performance.

Performance metrics	Description
Total duration	Total duration is a typical performance metric for measuring the time it takes to complete the processing of an entire dataset.
Resource utilization	The average CPU and memory that the system consumes while processing the dataset.

How test results are calculated

Performance (mainly scanning speed) is measured by throughput rather than unit speed. For example, if it takes 10 seconds to process 1 object, and it also takes 10 seconds to process 10 objects, then performance is quantified as 1 second per object, rather than 10 seconds.

- total time / total number of objects processed: 10 seconds / 10 objects = 1 second / object.

Dataset

File category	File type	Number of files	Total size	Average file size
Document	DOC	3,820	534 MB	0.14 MB
Medium archive files	RPM CAB EXE	50	Compressed size: 2.8 GB Extracted size: 12.09 GB	Compressed size: 56.02 MB Extracted size: 294 MB
Big archive files	CAB	7	Compressed size: 5.1 GB Extracted size: 217 GB	Compressed size: 715 MB Extracted size: 30.4 GB

Environment

MD Cluster system

	MD Core	File Storage	API Gateway	PostgreSQL	RabbitMQ	Redis
OS	Windows Server 2022	Rocky Linux 9	Rocky Linux 9	Rocky Linux 9	Rocky Linux 9	Rocky Linux 9

	MD Core	File Storage	API Gateway	PostgreSQL	RabbitMQ	Redis
vCPU	8	16	4	4	4	4
Memory	16GB	32GB	8GB	8GB	8GB	32GB
Disk size	300GB	150GB	100GB	100GB	80GB	80GB
Disk I/O	600 MB/s	600 MB/s	600 MB/s	600 MB/s	600 MB/s	600 MB/s
Network bandwidth	10 Gbps	10 Gbps	10 Gbps	10 Gbps	10 Gbps	10 Gbps

Client tool

	Detail
OS	Rocky Linux 9
vCPU	4
Memory	10GB
Disk size	80GB
Disk I/O	600 MB/s
Network bandwidth	10 Gbps

Product information

- MetaDefender Core v5.20.0
- Engines:
  - Metascan 8: Ahnlab, Avira, ClamAV, ESET, Bitdefender, K7, Quick Heal, VirIT Explorer
  - Archive v8.0.0
  - File type analysis v8.0.0
- MD Cluster Control Center v2.7.0
- MD Cluster Identity Service v2.7.0

- MD Cluster API Gateway v2.7.0
- MD Cluster File Storage v2.7.0
- PostgreSQL v16.10
- RabbitMQ v3.13.7
- Redis v8.2.3

Workflow settings

Archive Extraction settings

- Max recursion level: 99999999
- Max number of extracted files: 99999999
- Max total size of extracted files: 99999999
- Timeout: 10 minutes
- Handle archive extraction task as Failed: true
  - Extracted partially: true

Metascan settings

- Max file size: 99999999
- Scan timeout: 10 minutes
- Per engine scan timeout: 1 minutes

General settings

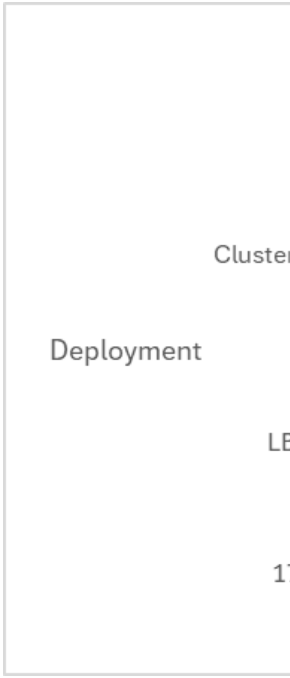
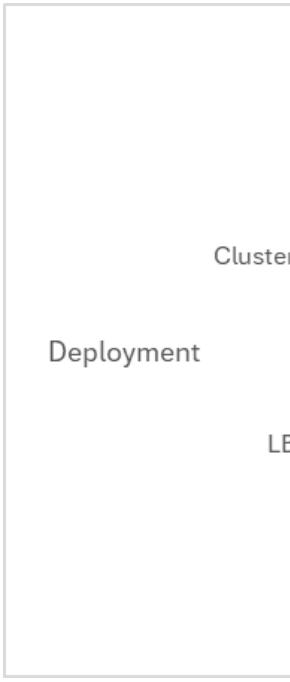
- Turn off data retention
- Turn off engine update
- Scan queue: 1000 (for Load Balancer deployment)

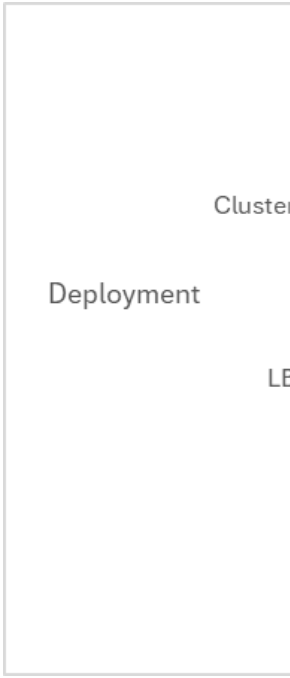
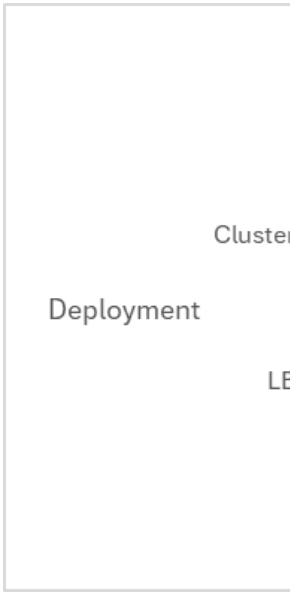
Performance results

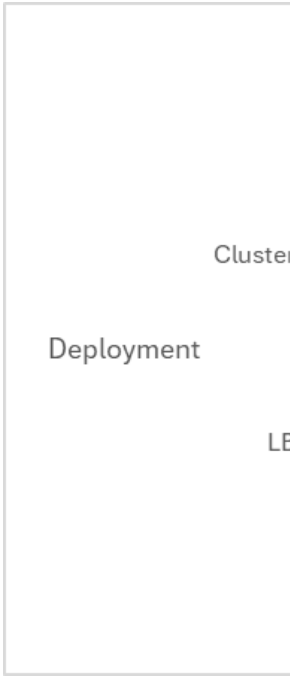
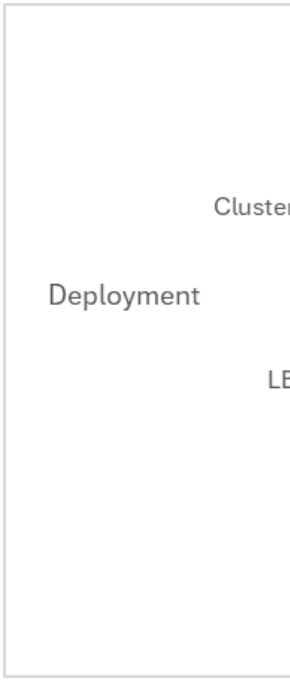
Load-balance deployment vs MD Cluster deployment

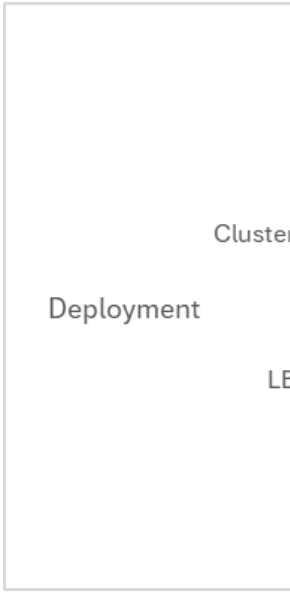
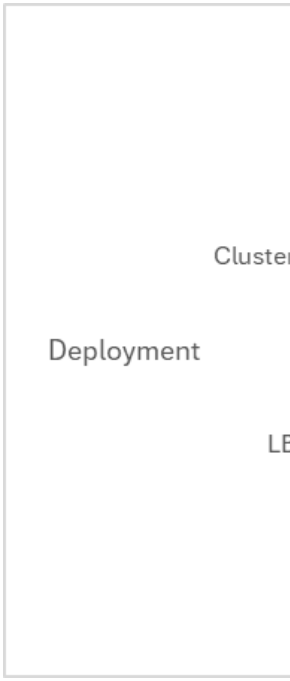
Multiple tests are conducted using **5** MetaDefender Core instances across two deployment types, MetaDefender Cluster (Cluster) and Load Balancer (LB), to determine the superiority of the MD Cluster in 3 different datasets.

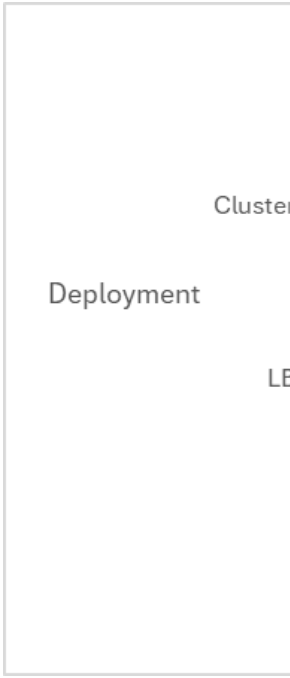
Scenario	Result
Aggressively submitted 2M non-archive files at a rate of 800 files per second.	

Scenario	Result
	 

Scenario	Result
	
Submitted 400 medium archive files at a rate of 1 files per second.	

Scenario	Result
	 
Submitted 7 large CAB files **Archive distribution (MD Cluster only)** In workflow, setting "Load shared among MetaDefender Core instances for archive processing" is enabled.	

Scenario	Result
☑ **Load shared among MetaDefender Core instances for archive processing** Applicable to Distributed Cluster deployment, nested files in archive could be processed in multiple MetaDefender Core instances, recommended when processing mostly big archive files.	 

Scenario	Result
	

Kubernetes

This guide provides installation and operating instructions for running **MetaDefender Cluster** on Kubernetes.

Supported Kubernetes versions

Version	Notes
1.31.x	Fully supported and validated.
Earlier versions	Not validated. The chart uses only stable APIs (`apps/v1`, `v1`), but no testing is performed below 1.31.

Guide structure

Start with Quick start for a first deployment. The other pages are reference material and can be read in any order once the cluster is running.

Page	What it covers
Quick start	Prerequisites and the step-by-step install
Kubernetes components	The workloads and images the chart creates, service discovery, the shared ConfigMap and Secret, and the pod security model
Enable HTTPs	Serving HTTPS from MetaDefender Cluster Control Center and MetaDefender Cluster API Gateway
Resource requirements	CPU, memory and ephemeral storage sizing
Scaling	Adding and removing MetaDefender Cluster Workers and MetaDefender Cluster File Storage replicas
Upgrade	Upgrading the core services and the deployed instances
High Availability	Clustered PostgreSQL, RabbitMQ and Redis, and multiple MetaDefender Cluster File Storage replicas
Troubleshooting	Symptoms, checks and log collection

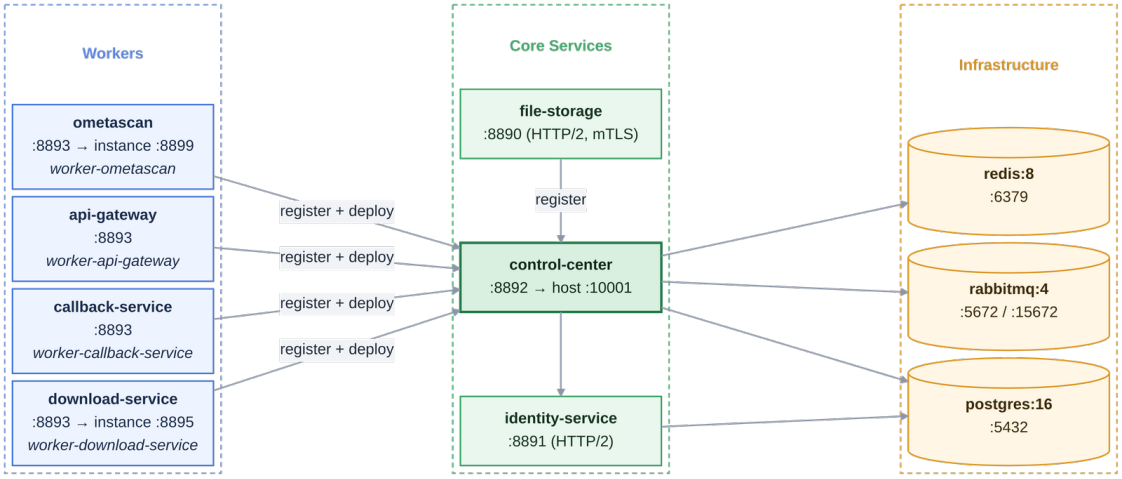
Kubernetes components

This page describes what the `mdcluster` chart creates and how configuration is shared between them.

Workloads

Product component	Kubernetes object	Workload	Service and port
MetaDefender Cluster Control Center	`control-center`	Deployment, always 1	`NodePort` by default · `8892`
MetaDefender Cluster Identity Service	`identity-service`	Deployment, always 1	`ClusterIP` · `8891`
MetaDefender Cluster File Storage	`file-storage`	StatefulSet	Headless · `8890`
MetaDefender Cluster Worker running MetaDefender Core	`ometascan`	StatefulSet	Shared headless `worker` · instance `8008`
MetaDefender Cluster Worker running the MetaDefender Cluster API Gateway	`api-gateway`	StatefulSet	Shared headless `worker`, plus its own `api-gateway` Service · instance `8899`
MetaDefender Cluster Worker running the MetaDefender Cluster Callback Service	`callback-service`	StatefulSet	Shared headless `worker` · instance `8894`
MetaDefender Cluster Worker running the MetaDefender Cluster Download Service	`download-service`	StatefulSet	Shared headless `worker` · instance `8895`

Architecture



Shared configuration

The chart creates two objects that the Control Center, Identity Service, File Storage and every worker pod mount as environment variables. (The bundled PostgreSQL, Redis and RabbitMQ pods do not get them; they read only the few credential values they need.)

Object	Rendered from	Contains
`mdcluster-config` ConfigMap	the `env:` block in your values file	Hostnames, ports, timeouts, file copy settings
`mdcluster-secrets` Secret	the `secrets:` block in your values file	Passwords, connection keys, the encryption key, the license key

Those seven pods pull in both objects wholesale with `envFrom`, so adding a key to `env:` or `secrets:` makes it visible to all of them without touching any template.

The bundled infrastructure works differently. None of the three mounts either object; each pod is given only the individual keys it needs, by reference:

Pod	Reads from the Secret
`postgres`	`CONTROL_CENTER_DB_USER`, `CONTROL_CENTER_DB_PASSWORD` — which is why those two also bootstrap its superuser
`rabbitmq`	`RABBITMQ_USER`, `RABBITMQ_PASSWORD`
`redis`	Nothing. The bundled Redis runs without authentication; `REDIS_USER` / `REDIS_PASSWORD` exist for pointing at an **external** Redis that requires it

Pod	Reads from the Secret

Quick start

Prerequisites

- **Kubernetes 1.31.x** with a `kubectl` context pointing at the cluster
- **Helm 3.x.**
- Access to a **container registry** holding the MetaDefender Cluster images.
- A valid MetaDefender **license key**.
- Requires LoadBalancer service support
- A default `StorageClass`, if you enable persistence for PostgreSQL or MetaDefender (MD) Cluster File Storage

Do not use Network File System (NFS) for persistent volumes. NFS can corrupt the MD Cluster File Storage metadata database and the PostgreSQL data directory. Longhorn is a tested alternative for on-cluster storage.

Step 1 — Add the chart repository

The charts are published in the OPSWAT Kubernetes chart repository:

The repository can be cloned locally:

Code · bash

```
git clone https://github.com/OPSWAT/metadefender-k8s.git metadefender

cd metadefender/helm_charts
```

Step 2 — Prepare a values file

Create your own values file — this guide calls it `override-values.yaml`

Code · yaml

```
# override-values.yaml

# --- Images ----------------------------------------------------------------
DOCKER_REPO: 'opswat/metadefendercluster-debian'
# imagePullSecrets:
#   - name: regcred
```

```
# --- Workers: replica count per instance type -------------------------------
workers:
  ometascan:
    replicas: 1

  api-gateway:
    replicas: 1
    # --- Expose API Gateway service by load balancer ---
    service:
      type: LoadBalancer

  callback-service:
    replicas: 1

  download-service:
    replicas: 1

secrets:
  # --- Bootstrap administrator account ---
  ADMIN_USER: '<admin-account>'
  ADMIN_PASSWORD: '<admin-password>'
  ADMIN_EMAIL: '<admin-email>'
  CONTROL_CENTER_ENCRYPTION_KEY: '<encryption-key>' # 32 alphanumeric characters
  LICENSE_KEY: '<license-key>'

# --- Expose Control Center service by load balancer ---
# control-center:
#   service:
#     type: LoadBalancer

# --- Persistence ------------------------------
postgres:
  enabled: true
  persistence:
    enabled: true
    size: 100Gi
  maxConnections: 400 # Max number of connections from the clients

file-storage:
  replicas: 1
  persistence:
    enabled: true
    size: 100Gi

# Container-level.
securityContext:
  readOnlyRootFilesystem: true # Change to false if not using a read-only file system.
```

Step 3 — Install

Code · bash

```
helm install mdcluster ./mdcl uster -f ./override-values.yaml --wait --timeout 5m
```

Step 4 — Verify

Code · bash

```
kubectl get pods -w
kubectl get svc -w
```

- Access MD Cluster Control Center UI using external IP with port `8892`.
- Post file to MD Cluster using external IP of MD Cluster API Gateway with port `8899`.

Configuration

All configuration is supplied through Helm values. This page covers the settings most deployments touch. For the exhaustive list, read the chart's own defaults — every block carries an inline comment:

Code · bash

```
helm show values ./mdcluster
```

Only the keys you set are overridden; everything else falls back to the chart default. Put them in your `override-values.yaml` (see Quick start) and apply with:

Code · bash

```
helm upgrade mdcluster ./mdcluster --install -f ./override-values.yaml
```

Where to configure what

To configure	See
HTTPS for the MetaDefender Cluster Control Center or API Gateway	Enable HTTPs
CPU, memory and ephemeral storage	Resource requirements

Enable HTTPs

Step 1 — Create TLS secret

Create cert and key with x509 format and add TLS secret:

Code · bash

```
kubectl create secret tls mdcluster-control-center-tls \
  --cert=<control-center-cert-file> --key=<control-center-key-file>

kubectl create secret tls mdcluster-api-gateway-tls \
  --cert=<api-gateway-cert-file> --key=<api-gateway-key-file>
```

Step 2 — Create TLS secret

Update `override-values.yaml` in Quick start

Code · yaml

```
control-center:
  tls:
    enabled: true
    secretName: mdcluster-control-center-tls

workers:
  api-gateway:
    tls:
      enabled: true
      secretName: mdcluster-api-gateway-tls
```

Step 3 — Apply config

Code · bash

```
helm upgrade mdcluster ./mdcluster --install -f ./override-values.yaml
```

Resource requirements

By default, the charts set no resource requests or limits. For any environment carrying real traffic, set requests and limits explicitly.

The published baseline

Per-component minimums are maintained in System requirements:

Component	CPU	Memory	Storage

Component	CPU	Memory	Storage
`postgres`	8	16Gi	1Ti SSD/NVMe
`redis`	2	32Gi	20Gi
`rabbitmq`	4	8Gi	100Gi
`file-storage`	8	8Gi	1Ti SSD/NVMe
`control-center`	4	4Gi	100Gi
`identity-service`	4	4Gi	50Gi
`api-gateway`	4	8Gi	50Gi
`callback-service`	4	4Gi	50Gi
`download-service`	4	4Gi	50Gi
`ometascan`	See the MetaDefender Core requirements		

Assigning CPU and memory

1. Requests and limits use the standard Kubernetes `resources` block. Every component in both charts accepts it. Update `override-values.yaml` in Quick start

Code · yaml

```yaml
# --- Instance ---
workers:
  ometascan:
    resources:
      requests: { cpu: "4", memory: 4Gi, ephemeral-storage: 200Gi }
      limits: { cpu: "8", memory: 16Gi, ephemeral-storage: 200Gi }

  # The API Gateway and Callback Service have the same allowance.
  api-gateway:
    resources:
      requests: { cpu: "4", memory: 4Gi, ephemeral-storage: 50Gi }
      limits: { cpu: "4", memory: 4Gi, ephemeral-storage: 50Gi }

  callback-service:
```

```
    resources:
      requests: { cpu: "4", memory: 8Gi, ephemeral-storage: 50Gi }
      limits: { cpu: "4", memory: 8Gi, ephemeral-storage: 50Gi }

  download-service:
    resources:
      requests: { cpu: "4", memory: 4Gi, ephemeral-storage: 50Gi }
      limits: { cpu: "4", memory: 4Gi, ephemeral-storage: 50Gi }

# --- Services ---
control-center:
  resources:
    requests: { cpu: "2", memory: 4Gi, ephemeral-storage: 100Gi }
    limits:   { cpu: "4", memory: 4Gi, ephemeral-storage: 100Gi }

identity-service:
  resources:
    requests: { cpu: "1", memory: 1Gi, ephemeral-storage: 50Gi }
    limits:   { cpu: "2", memory: 2Gi, ephemeral-storage: 50Gi }

file-storage:
  resources:
    requests: { cpu: "2", memory: 4Gi }
    limits:   { cpu: "8", memory: 8Gi }
  persistence:
    enabled: true
    size: 100Gi

postgres:
  resources:
    requests: { cpu: "4", memory: 8Gi }
    limits:   { cpu: "8", memory: 8Gi }
  persistence:
    enabled: true
    size: 100Gi
```

2. Apply config

Code · bash

```
helm upgrade mdcluster ./mdcluster --install -f ./override-values.yaml
```

Upgrade

MetaDefender Cluster upgrades are Helm upgrades. A `helm upgrade` covers the core services and the workers together — or, by pinning individual `version` keys, one component at a time.

Step 1 — Clone the new version

Code · bash

```
# Clone new
git clone https://github.com/OPSWAT/metadefender-k8s.git metadefender

# Or pull
git pull -r

cd metadefender/helm_charts/mdcluster
```

Step 2 — Upgrade

Option 1: Upgrade everything

Code · bash

```
helm upgrade mdcluster ./mdcluster --reuse-values \
  --set MDCLS_VERSION=<new_version>
```

Option 2: Upgrade one component

Code · bash

```
# component = control-center | file-storage | identity-service
helm upgrade mdcluster ./mdcluster --reuse-values \
  --set <component>.version=<new_version>
```

Code · bash

```
# instance_type = ometascan | api-gateway | callback-service
helm upgrade mdcluster ./mdcluster --reuse-values \
  --set workers.<instance_type>.version=<new_version>
```

Scaling

Two kinds of scaling apply to MetaDefender Cluster, and they work very differently.

MetaDefender Cluster Workers scale freely and without downtime. They register themselves with the MetaDefender Cluster Control Center on start and unregister on stop, so adding or removing replicas needs no coordination.

MetaDefender Cluster File Storage does not. The MetaDefender Cluster Control Center is given its list of MetaDefender Cluster File Storage endpoints at chart render time, so changing the replica count requires a `helm upgrade` — and a planned downtime window.

MetaDefender Cluster Control Center and MetaDefender Cluster Identity Service are singletons and cannot be scaled.

MetaDefender Cluster Workers

Scale each instance type independently, according to which resource is the bottleneck: `ometascan` for scan throughput, `api-gateway` for file submission capacity, `callback-service` for webhook delivery.

Code · bash

```
helm upgrade mdcluster ./mdcluster --reuse-values \
  --set workers.ometascan.replicas=5 \
  --set workers.api-gateway.replicas=3 \
  --set workers.callback-service.replicas=3 \
  --set workers.download-service.replicas=3
```

MetaDefender Cluster File Storage

Never scale MetaDefender Cluster File Storage in. Files are distributed across the instances, not copied to all of them, so removing a replica removes the MetaDefender Cluster Control Center's only route to every file that lived there. With the default `FILE_STORAGE_MAX_REPLICA: 1` each file has exactly one copy, so scaling in loses access to a share of your stored files immediately — and the MetaDefender Cluster Control Center reports nothing wrong.

Scaling **out** is safe, but offline: it needs a maintenance window with all MetaDefender Cluster Worker pods stopped. Size MetaDefender Cluster File Storage at install time so you do not need either operation.

Scaling MetaDefender Cluster File Storage requires **downtime**.

Procedure

1. **Scale all MetaDefender Cluster Worker instances to zero.** MetaDefender Cluster File Storage cannot be reconfigured while MetaDefender Cluster workers are running.

   Code · bash

   ```
   helm upgrade mdcluster ./mdcluster --reuse-values \
     --set workers.ometascan.replicas=0 \
     --set workers.api-gateway.replicas=0 \
     --set workers.callback-service.replicas=0 \
     --set workers.download-service.replicas=0

   # Wait for every worker pod to finish draining
   ```

2. **Scale MetaDefender Cluster File Storage:**

   Code · bash

   ```
   helm upgrade mdcluster ./mdcluster --reuse-values \
   ```

```
  --set file-storage.replicas=5 \
  --set env.FILE_STORAGE_MIN_REPLICA=2 \
  --set env.FILE_STORAGE_MAX_REPLICA=2
```

3. **Scale instance again:**

Code · bash

```
helm upgrade mdcluster ./mdcluster --reuse-values \
  --set workers.ometascan.replicas=5 \
  --set workers.api-gateway.replicas=3 \
  --set workers.callback-service.replicas=3 \
  --set workers.download-service.replicas=3
```

Finally, confirm on the **Services** page of the console that every MetaDefender Cluster File Storage replica is listed and healthy.

Two settings control how many copies of each file are stored - see Configuration.

High availability

The bundled `postgres`, `redis` and `rabbitmq` are **single-pod** deployments. They exist so that a fresh install works out of the box and matches the Docker Compose stack — not for production. Each is a single point of failure whose loss stops the cluster.

MetaDefender Cluster Workers already running continue to serve traffic through a MetaDefender Cluster Control Center outage.

Choosing an approach

In production, prefer a **managed service** from your cloud provider. Azure, AWS and Google Cloud all offer PostgreSQL-compatible databases with built-in high availability, backups and dynamic scaling — considerably less to operate than a database cluster you run yourself.

If the dependency must live inside the Kubernetes cluster, the options below are tested:

Dependency	On-cluster solution
PostgreSQL	CloudNativePG operator, 3 instances
Redis	dandydeveloper/redis-ha (Redis + Sentinel), 3 nodes, HAProxy enabled
RabbitMQ	RabbitMQ Cluster Operator, 3 nodes

Namespace and DNS. The Service names below assume the HA components run in the same namespace as the

mdcluster release. If they live elsewhere, use the fully qualified form `<service>.<namespace>.svc.cluster.local`.

RabbitMQ

For production, use the official RabbitMQ Cluster Operator.

The Bitnami community chart also works and is quicker to stand up, but note that it now ships images from the `bitnamilegacy` repository and is best-effort and unsupported.

Create `bitnami_rabbitmq.yaml`:

Code · yaml

```yaml
replicaCount: 3

persistence:
  enabled: true
  size: 50Gi

auth:
  username: '<rabbitmq-user>'
  password: '<rabbitmq-password>'
  erlangCookie: '<erlang-cookie>'

image:
  registry: docker.io
  repository: bitnamilegacy/rabbitmq

global:
  security:
    allowInsecureImages: true

# resources:
#   requests: { cpu: "4", memory: "8Gi" }
#   limits:   { cpu: "4", memory: "8Gi" }
```

Code · bash

```bash
helm repo add bitnami https://charts.bitnami.com/bitnami
helm upgrade --install md-rabbitmq-ha bitnami/rabbitmq \
  --version 16.0.14 -f ./bitnami_rabbitmq.yaml
```

This creates pods `md-rabbitmq-ha-0`, `-1` and `-2`, reachable through the headless Service `md-rabbitmq-ha-headless`.

Redis

Create `dandy_redis.yaml`:

Code · yaml

```yaml
persistentVolume:
  enabled: false
redis:
  disableCommands: null      # keep FLUSHDB/FLUSHALL enabled — the scan engines use them
haproxy:
  enabled: true              # required: gives one writable endpoint at md-redis-ha-haproxy:6379
  # resources:
  #   requests: { cpu: "2", memory: "8Gi" }
  #   limits:   { cpu: "2", memory: "8Gi" }
# sentinel:
#   resources:
#     requests: { cpu: "100m", memory: "256Mi" }
#     limits:   { cpu: "200m", memory: "512Mi" }
```

Code · bash

```bash
helm repo add dandydev https://dandydeveloper.github.io/charts
helm upgrade --install md-redis-ha dandydev/redis-ha -f ./dandy_redis.yaml
```

This creates pods `md-redis-ha-server-0`, `-1` and `-2`, reachable through the `md-redis-ha` Service.

Front the cluster with a single writable endpoint instead. Either enable the chart's HAProxy (haproxy.enabled=true, giving one endpoint at md-redis-ha-haproxy:6379), or use a managed Redis that presents a single writable address.

PostgreSQL

CloudNativePG runs PostgreSQL as an operator-managed cluster with automatic failover.

Create the credentials Secret

`cloudpg_secret.yaml`:

Code · yaml

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: md-postgres-ha-secret
type: kubernetes.io/basic-auth
stringData:
  username: '<db-user>'      # must match the role name below
  password: '<db-password>'  # must match every *_DB_PASSWORD / *_PASSWORD value
```

Define the cluster

`cloudpg_cluster.yaml`:

Code · yaml

```yaml
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
  name: md-postgres-ha
spec:
  instances: 3
  storage:
    size: 100Gi
    # storageClass: ""
  # resources:
  #   requests: { cpu: "4", memory: "8Gi" }
  #   limits:   { cpu: "4", memory: "8Gi" }
  managed:
    roles:
      - name: '<db-user>'
        passwordSecret:
          name: md-postgres-ha-secret
        login: true
        createdb: true       # required: the Control Center creates the four cluster databases
        superuser: true
  postgresql:
    parameters:
      log_min_duration_statement: "300"   # log queries slower than 300ms
      log_line_prefix: "%t [%p]: [%l-1] user=%u,db=%d,app=%a,client=%h "
      log_checkpoints: "off"
      log_connections: "off"
      log_disconnections: "off"
      log_lock_waits: "on"
```

The role needs `createdb` and `superuser` because the MetaDefender Cluster Control Center's database initialisation step creates the MetaDefender Cluster databases.

Deploy

Install the operator with cluster-admin privileges — the manifest creates cluster-scoped resources:

Code · shell

```shell
kubectl apply --server-side -f \
  https://raw.githubusercontent.com/cloudnative-pg/cloudnative-pg/release-1.30/releases/
cnpg-1.30.0.yaml
```

Then apply the Secret and the cluster:

Code · shell

```
kubectl apply -f cloudpg_secret.yaml
kubectl apply -f cloudpg_cluster.yaml
```

CloudNativePG exposes three Services: `md-postgres-ha-rw` (the current primary, read-write), `md-postgres-ha-ro` (replicas) and `md-postgres-ha-r` (any instance). **MetaDefender Cluster must use -rw.** The endpoint follows the primary automatically on failover.

MetaDefender Cluster File Storage

File Storage is part of the `mdcluster` chart rather than an external dependency, but it holds submitted files and their metadata, so it needs its own treatment.

Configuration

Code · yaml

```
env:
  FILE_STORAGE_MIN_REPLICA: 2
  FILE_STORAGE_MAX_REPLICA: 2

file-storage:
  replicas: 3
  persistence:
    enabled: true     # one PVC per replica; false means ephemeral emptyDir
    size: 100Gi       # per replica
```

`FILE_STORAGE_MIN_REPLICA` and `FILE_STORAGE_MAX_REPLICA` control how many copies of each file are stored. They are a redundancy factor, not a limit on how many instances the MetaDefender Cluster Control Center connects to.

Setting	Meaning
`env.FILE_STORAGE_MIN_REPLICA`	Minimum copies of each file required for a write to succeed — **and** minimum healthy instances.
`env.FILE_STORAGE_MAX_REPLICA`	How many copies of each file to write

This must hold:

`FILE_STORAGE_MIN_REPLICA` <= `FILE_STORAGE_MAX_REPLICA` <= `file-storage.replicas`

Requirements

Persistence must be enabled. With `persistence.enabled: false` each replica uses an ephemeral `emptyDir` and loses its files on restart, which defeats the point of running several.

Each replica needs its own ReadWriteOnce volume. Do not point the replicas at a single shared `ReadWriteMany` volume — each instance owns its own metadata database and expects exclusive access.

Spread the replicas across nodes. By default nothing prevents all MetaDefender Cluster File Storage pods from landing on the same node, which would make node loss a total outage. Add anti-affinity:

Code · yaml

```yaml
file-storage:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - topologyKey: kubernetes.io/hostname
          labelSelector:
            matchLabels:
              app: file-storage
```

Use `requiredDuringScheduling` only if you have at least as many eligible nodes as replicas; otherwise the surplus pods stay `Pending`. Use `preferredDuringScheduling` for a best-effort spread. For zone-level fault tolerance, combine this with a zone-aware `StorageClass` — a pod cannot move to another zone if its volume is pinned to the first one.

Point MetaDefender Cluster at the external infrastructure

Add the following to your values file. Passwords **must match** the RabbitMQ chart auth and the CloudNativePG role Secret exactly — a mismatch produces authentication failures at startup rather than a clear configuration error.

Code · yaml

```yaml
# Disable the bundled single-pod infrastructure
postgres: { enabled: false }
redis:    { enabled: false }
rabbitmq: { enabled: false }

secrets:
  CONTROL_CENTER_ENCRYPTION_KEY: '<32-character-key>'
  ADMIN_APIKEY: '<control-center-api-key>'
  # LICENSE_KEY: '<license-key>'

  # PostgreSQL — all four pairs must match the CloudNativePG role
  CONTROL_CENTER_DB_USER: '<db-user>'
  CONTROL_CENTER_DB_PASSWORD: '<db-password>'
  IDENTITY_DB_USER: '<db-user>'
  IDENTITY_DB_PASSWORD: '<db-password>'
  DATALAKE_USER: '<db-user>'
  DATALAKE_PASSWORD: '<db-password>'
  WAREHOUSE_USER: '<db-user>'
```

```
  WAREHOUSE_PASSWORD: '<db-password>'

  # RabbitMQ — must match bitnami_rabbitmq.yaml auth
  RABBITMQ_USER: '<rabbitmq-user>'
  RABBITMQ_PASSWORD: '<rabbitmq-password>'

  # REDIS_USER / REDIS_PASSWORD — only if your Redis requires auth

env:
  # PostgreSQL: always the read-write endpoint
  CONTROL_CENTER_DB_HOST: 'md-postgres-ha-rw'
  CONTROL_CENTER_DB_PORT: 5432
  IDENTITY_DB_HOST: 'md-postgres-ha-rw'
  IDENTITY_DB_PORT: 5432
  DATALAKE_SERVICES:  'md-postgres-ha-rw:5432'
  WAREHOUSE_SERVICES: 'md-postgres-ha-rw:5432'

  # RabbitMQ: all three brokers
  RABBITMQ_SERVICES: 'md-rabbitmq-ha-0.md-rabbitmq-ha-headless:5672,md-rabbitmq-ha-1.md-
rabbitmq-ha-headless:5672,md-rabbitmq-ha-2.md-rabbitmq-ha-headless:5672'

  # Redis: the single writable HAProxy endpoint, never the individual nodes
  REDIS_SERVICES: 'md-redis-ha-haproxy:6379'

  # File Storage: must be <= file-storage.replicas
  FILE_STORAGE_MIN_REPLICA: 2
  FILE_STORAGE_MAX_REPLICA: 2

# Multiple replicas of the stateful cluster services
file-storage:
  replicas: 3
  persistence: { enabled: true, size: 100Gi }

workers:
  ometascan:        { replicas: 3 }
  api-gateway:      { replicas: 2 }
  callback-service: { replicas: 2 }
```

Install as usual:

Code · bash

```
helm install mdcluster ./mdcluster -f ./override-values.yaml
```

Troubleshooting

Use this guide when a MetaDefender (MD) Cluster service does not start, keeps restarting, or is not working as expected. Start with the quick checks below, then collect the relevant logs and configuration if you need help from OPSWAT Support.

> *The commands on this page use the current Kubernetes namespace. If MetaDefender Cluster is deployed in another namespace, add `--namespace <namespace>` to each command or switch to that namespace before you begin.*

Start with a quick health check

Check the status of all pods and review recent Kubernetes events:

Code · bash

```
kubectl get pods -o wide
```

Look for pods that are not `Running` or `Ready`, frequent restarts, scheduling failures, storage problems, or failed health checks.

For more information about a specific pod, run:

Code · bash

```
kubectl describe pod <pod-name>
```

The **Events** section at the end of the output often explains why the pod cannot start or become ready.

Common pod states

Pod status	What to check
`Pending`	Run `kubectl describe pod <pod-name>` and check for insufficient CPU or memory, unbound volumes, or scheduling restrictions.
`CrashLoopBackOff`	Check both the current and previous container logs. The previous logs usually contain the error that caused the last restart.
`ImagePullBackOff`	Check the image name, registry access, image pull secret, and network connectivity to the registry.
`Running` but not `Ready`	Review the pod events and logs for failed readiness checks or unavailable dependencies.

Review service logs

Follow the logs for a Deployment:

Code · bash

```
kubectl logs -f deploy/control-center
kubectl logs -f deploy/identity-service
```

Follow the logs for a specific StatefulSet pod:

Code · bash

```
kubectl logs -f file-storage-<pod_index>
kubectl logs -f ometascan-<pod_index>
```

If a container has restarted, view the logs from its previous run:

Code · bash

```
kubectl logs <pod-name> --previous
```

For pods with more than one container, include logs from every container:

Code · bash

```
kubectl logs <pod-name> --all-containers
```

Press **Ctrl+C** to stop following live logs.

Check the active configuration

Review the configuration currently used by the deployment:

Code · bash

```
kubectl get configmap mdcluster-config -o yaml
kubectl get secret mdcluster-secrets -o jsonpath='{.data}' | jq 'keys'
helm list
helm get values <release-name> --all
```

The Secret command displays key names only; it does not decode or print their values.

`helm get values` shows the values stored in the Helm release. These values may differ from your local values file, especially after an upgrade performed with `--reuse-values`.

Collect information for OPSWAT Support

The easiest way to collect complete MD Cluster logs is to use Export in MD Cluster Control Center.

If MD Cluster Control Center is unavailable, you can collect Kubernetes diagnostics manually. The following

commands save the output in your current directory:

Code · bash

```
# Workload and storage status
kubectl get all,pvc -o wide > cluster-state.txt

# Pod details and recent events
kubectl describe pods > pod-details.txt
kubectl get events --sort-by=.metadata.creationTimestamp > cluster-events.txt

# Active configuration (Secret values are not included)
kubectl get configmap mdcluster-config -o yaml > config.yaml
helm get values <release-name> --all > values-<release-name>.yaml

# The latest 5,000 log lines from every pod and container
for p in $(kubectl get pods -o name); do
  kubectl logs "$p" --all-containers --tail=5000 \
    > "logs-${p#pod/}.txt" 2>&1
done
```

If an affected pod has restarted, also collect its previous logs:

Code · bash

```
kubectl logs <pod-name> --all-containers --previous > logs-<pod-name>-previous.txt 2>&1
```

Include the following details in your support request:

- MD Cluster version (`MDCLS_VERSION`)
- Kubernetes version (`kubectl version`)
- Helm release names (`helm list`)
- Whether PostgreSQL, Redis, RabbitMQ, and MD Cluster File Storage are bundled or externally managed
- Name of the affected service or pod
- Approximate time of the issue, including the time zone
- Steps to reproduce the issue, if known
- Any configuration, upgrade, or infrastructure changes made before the issue started

Log Gathering in MetaDefender Cluster

Download support packages

From the web console of MetaDefender Cluster (MD Cluster), the administrator can easily download the support packages of the following services:

- MD Cluster **Control Center**
- MD Cluster **Identity Service**
- MD Cluster **File Storage**
- MD Cluster **Worker** including MD Cluster **API Gateway**, MD Cluster **Callback Service** or **MetaDefender Core** deployed by the worker.

Please refer to Remote Support Package Gathering for more information.

Collect service logs

Logs from the services Redis, RabbitMQ, and PostgreSQL need to be collected manually.

Redis - Caching Server

> **Info**
> Redis caching server is officially supported on Linux.

1. Run Terminal as root privilege (`sudo`).
2. Open Redis config file `/etc/redis/redis.conf` in edit mode e.g.:

Code · bash

```
$ vi /etc/redis/redis.conf
```

3. Find and replace `logfile` directive with your desired location.

Code · bash

```
logfile "<path/to/your/redis/log>.log"
```

4. Save the file, and restart Redis daemon.

Code · bash

```
$ sudo systemctl restart redis
```

5. Find and collect Redis log `<path/to/your/redis/log>.log`

RabbitMQ - Message Broker Server

Windows

1. Locate and collect RabbitMQ log files that match the pattern `%APPDATA%\RabbitMQ\log\rabbit@<computer name>.log`.
2. Locate and collect RabbitMQ upgrade log files that match the pattern `%APPDATA%\RabbitMQ\log\rabbit@<computer name>_upgrade.log`.

Linux

1. Run terminal as root privilege (`sudo`).
2. Run following command to retrieve RabbitMQ log location:

Code · bash

```
$ rabbitmq-diagnostics -q log_location
```

3. Access RabbitMQ log folder and find log files:
   - `rabbit@<computer name>.log`
   - `rabbit@<computer name>_upgrade.log`

PostgreSQL - Database Server

Windows

1. Locate and collect log files that match the pattern `C:\Program Files\PostgreSQL\12\data\log with names postgresql-<yyyy-mm-dd>_<HHMMSS>.log`

Linux

1. Run terminal as root privilege (`sudo`).
2. Open the PostgreSQL config file `/etc/postgresql/12/main/postgresql.conf` in edit mode e.g.:

Code · bash

```
$ vi /etc/postgresql/12/main/postgresql.conf
```

3. Find and turn `logging_collector` directive on :

```
Code · bash

logging_collector = on
```

4. Save the file and restart PostgreSQL daemon, e.g.:

```
Code · bash

$ sudo systemctl restart postgresql
```

5. Locate and collect log files that match the pattern `/var/lib/postgresql/12/main/log/postgresql-<yyyy-mm-dd>_<HHMMSS>.log`.

Open Connection On PostgreSQL Server

> **Info**
> Just in case the firewall is enabled, please also ensure that you configure your firewall rules properly for the connections between PostgreSQL server and the services of MetaDefender Cluster, which include MetaDefender Core services.

> **Info**
> The guide here assumes we are using an SSL connection with PostgreSQL. With a non-SSL connection, please use `host` instead.

Windows

1. Locate and modify `pg_hba.conf` configuration file within the PostgreSQL data directory. For example**:** `C:\Program Files\PostgreSQL\16\data\pg_hba.conf`.

```
pg_hba.conf · markdown

hostssl    all             all             0.0.0.0/0            scram-sha-256
```

In the above example, all source addresses from MetaDefender Cluster and MetaDefender Core services are permitted. Refer here for more details.

2. Locate and modify `postgresql.conf` configuration file within the PostgreSQL data directory. For example: `C:\Program Files\PostgreSQL\<version>\data\postgresql.conf`.

postgresql.conf · markdown

```
listen_addresses = '*'
```

The configuration above directs PostgreSQL server to permit incoming connections from all sources associated with MetaDefender Cluster and MetaDefender Core services**.** Learn more from here.

Linux

1. Locate and modify the `pg_hba.conf` configuration file within the PostgreSQL data directory. For example**:** `/var/lib/pgsql/<version>/data/pg_hba.conf`.

pg_hba.conf · markdown

```
hostssl    all             all             0.0.0.0/0            scram-sha-256
```

In the above example, all source addresses from MetaDefender Cluster and MetaDefender Core services are permitted. Refer here for more details.

2. Locate and modify the `postgresql.conf` configuration file within the PostgreSQL data directory. For example: `/var/lib/pgsql/<version>/data/postgresql.conf`.

postgresql.conf · markdown

```
listen_addresses = '*'
```

The configuration above directs PostgreSQL server to permit incoming connections from all sources associated with MetaDefender Cluster and MetaDefender Core services**.** Learn more from here.

What is the latest MetaDefender Cluster version?

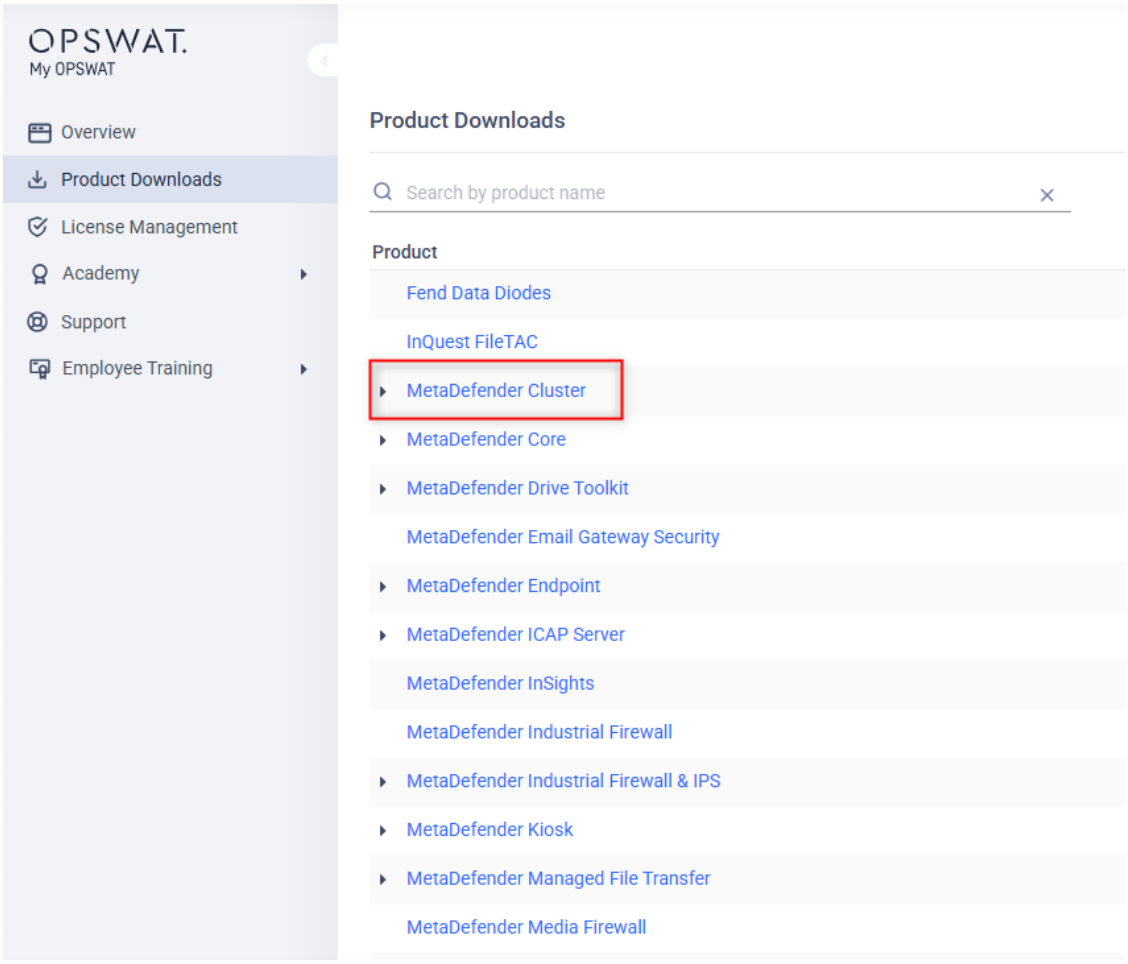
Check Your Version:

This article applies to all MetaDefender Cluster V2 releases deployed on Windows and Linux systems.

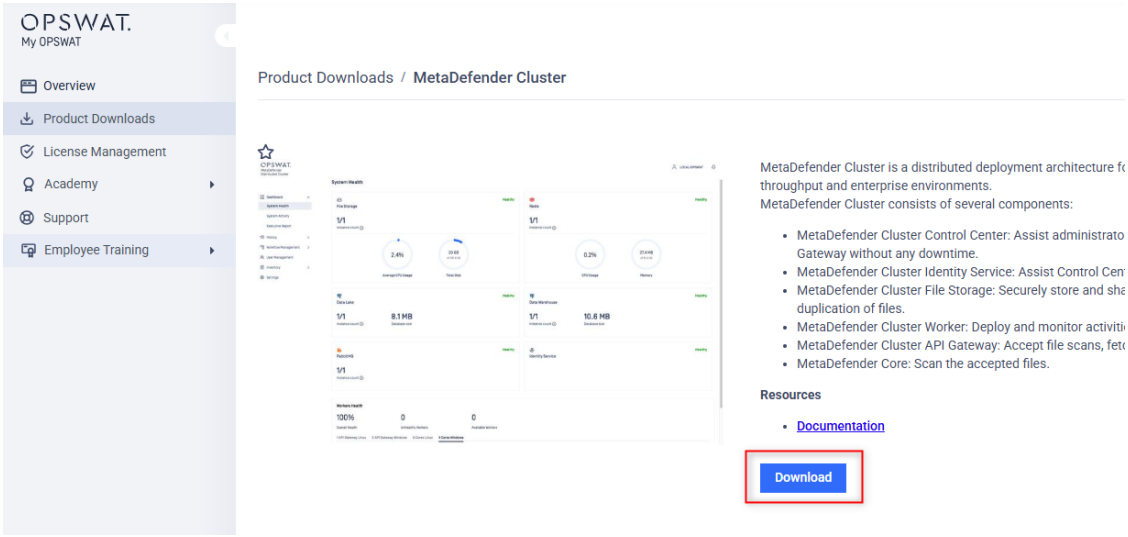
Browse MetaDefender Cluster Release Notes for details on the latest version, version history, version comparisons and release notes.

Alternatively, go directly through the My OPSWAT for a list of available downloads.:

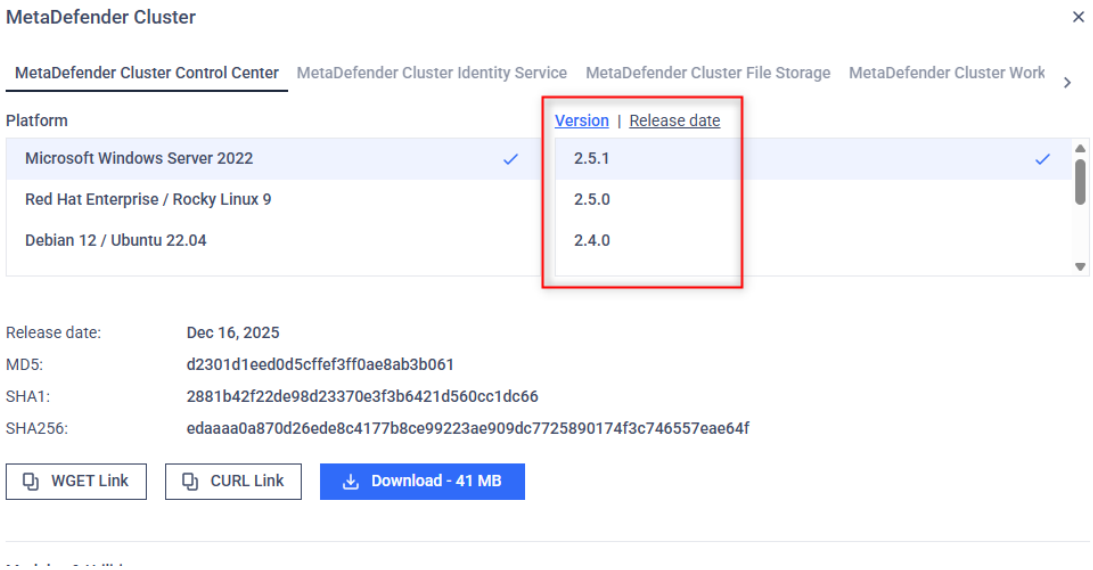
1. Login to My OPSWAT portal
2. Click on “MetaDefender Cluster”



3. Click on “Download” button



4. In the new window opened, always the version from the top is the latest version



Pro Tip:

OPSWAT highly recommends that users upgrade to the latest version of MetaDefender Cluster. Current versions incorporate state-of-the-art features and necessary bug fixes, and are freely available for all license holders to download.

Support:

Feel free to contact OPSWAT support for further guidance through your Upgrade process. Please follow these instructions on How To Create a Support Package, before logging a Support Ticket with the OPSWAT team.

Configure Firewall Rules to Allow Incoming Connections

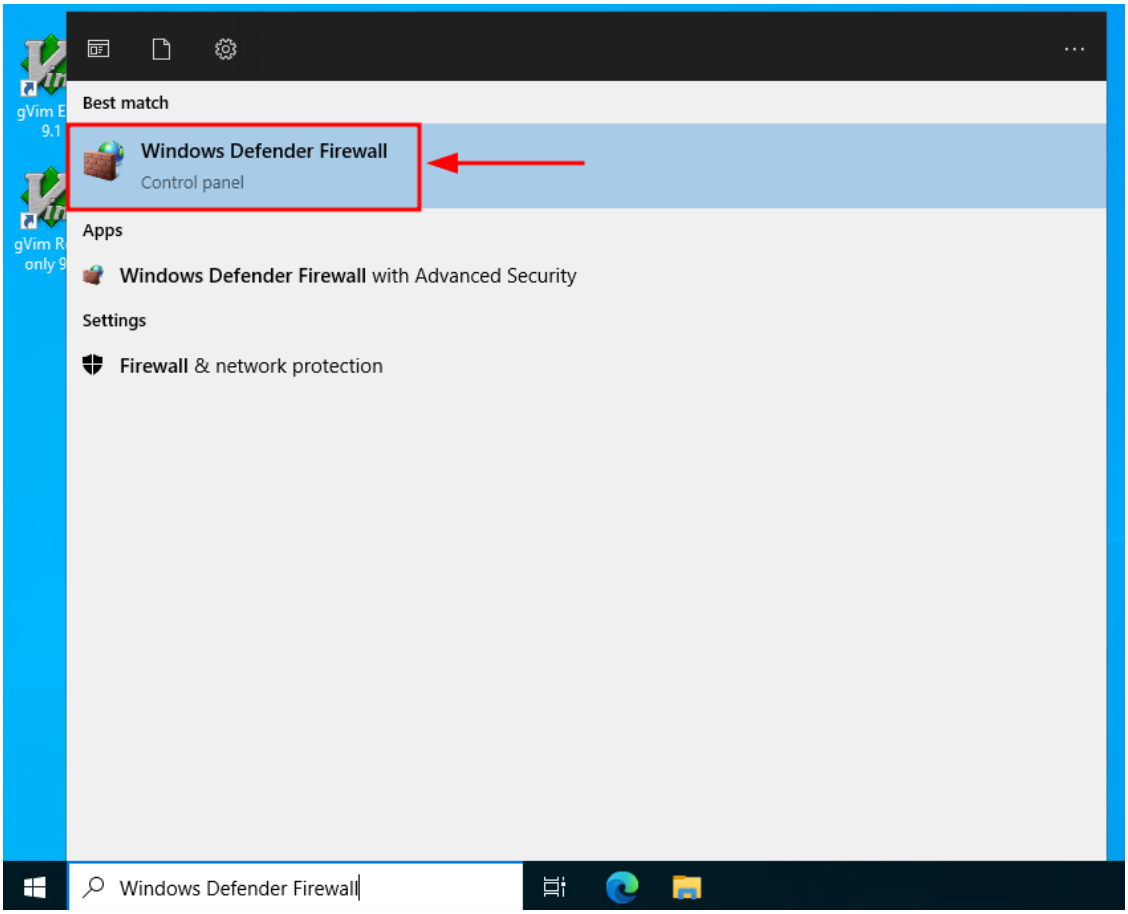
Some deployments require inbound network access on specific TCP ports. If the operating system firewall blocks these ports, clients or services may be unable to connect.

This guide explains how to allow inbound TCP connections using several commonly used firewall solutions.

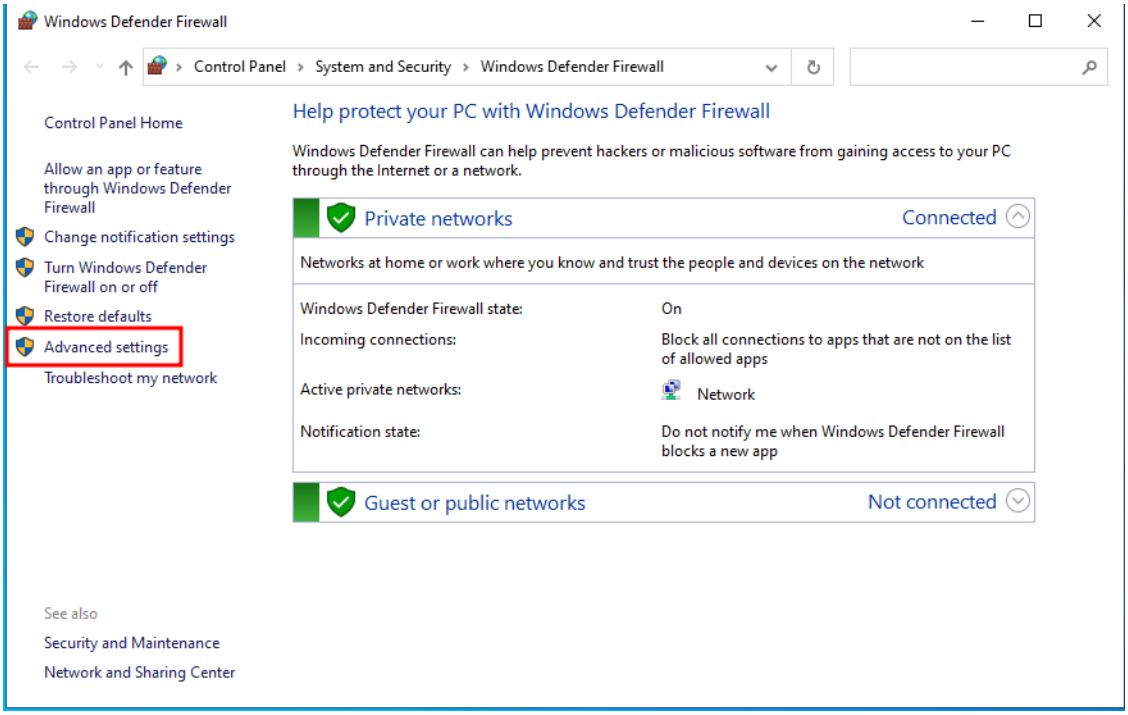
Windows Defender Firewall

Option 1: Using the Windows Defender Firewall Console

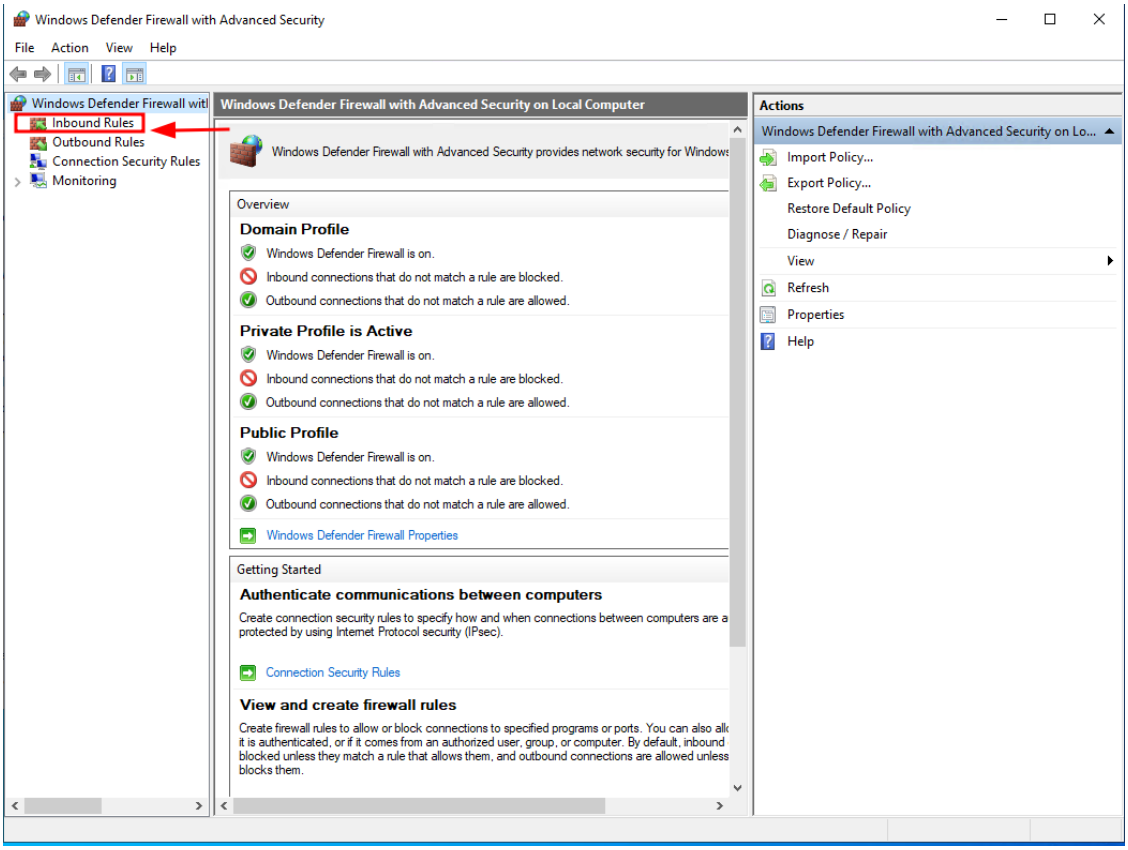
1. Open **Windows Defender Firewall** from the Windows Start menu or search bar.



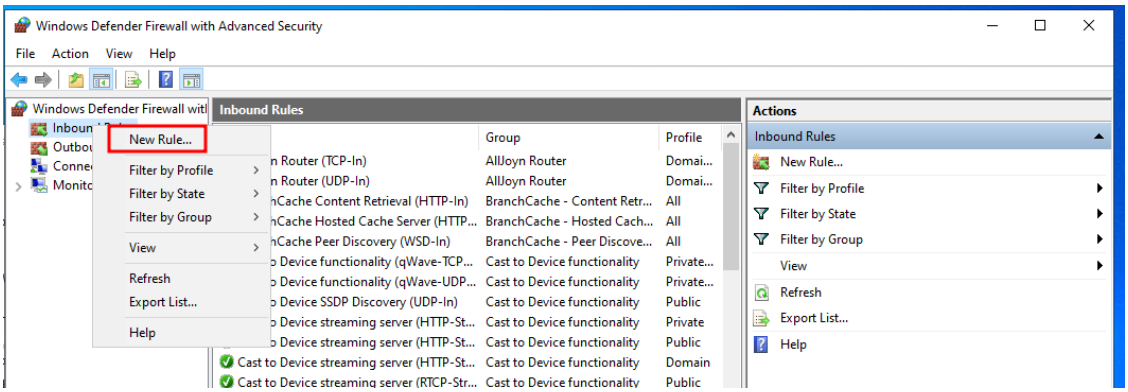
2. Select **Advanced settings** to open **Windows Defender Firewall with Advanced Security**.



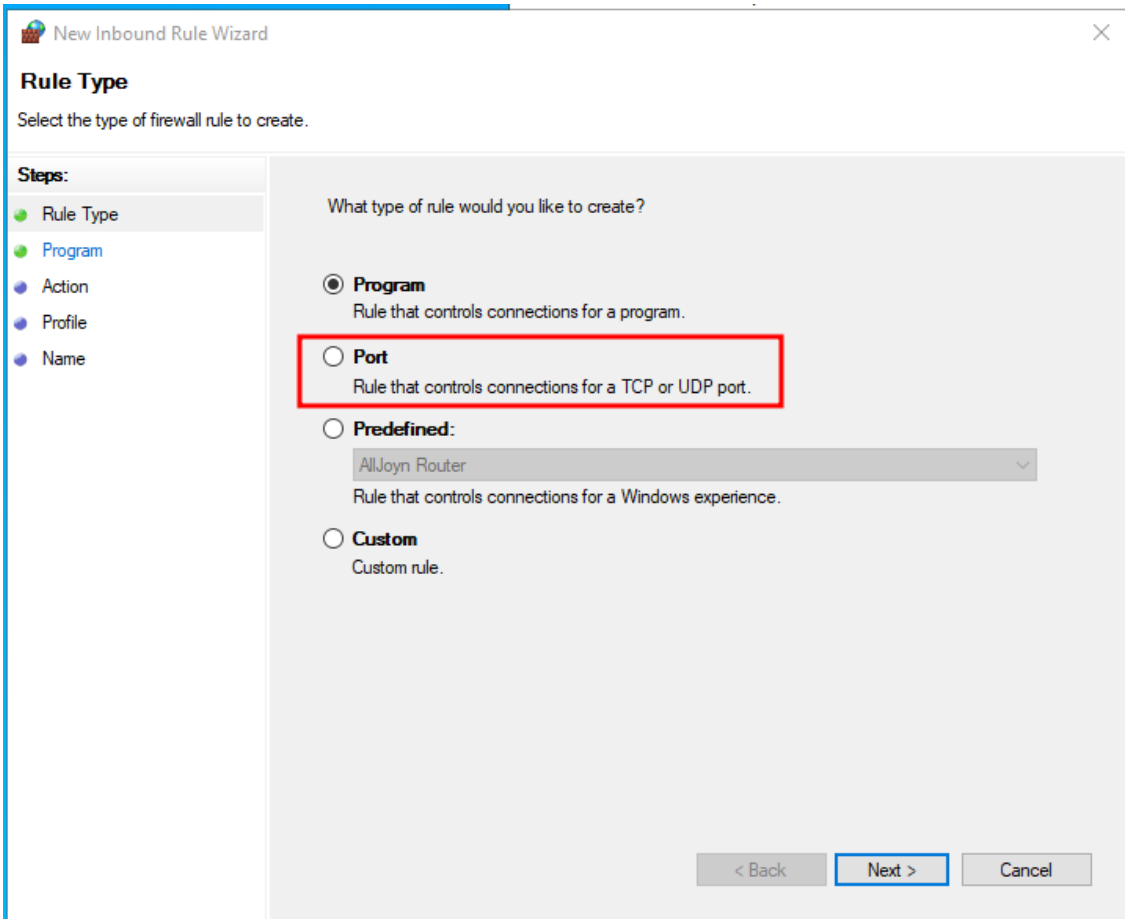
3. Select **Inbound Rules**.



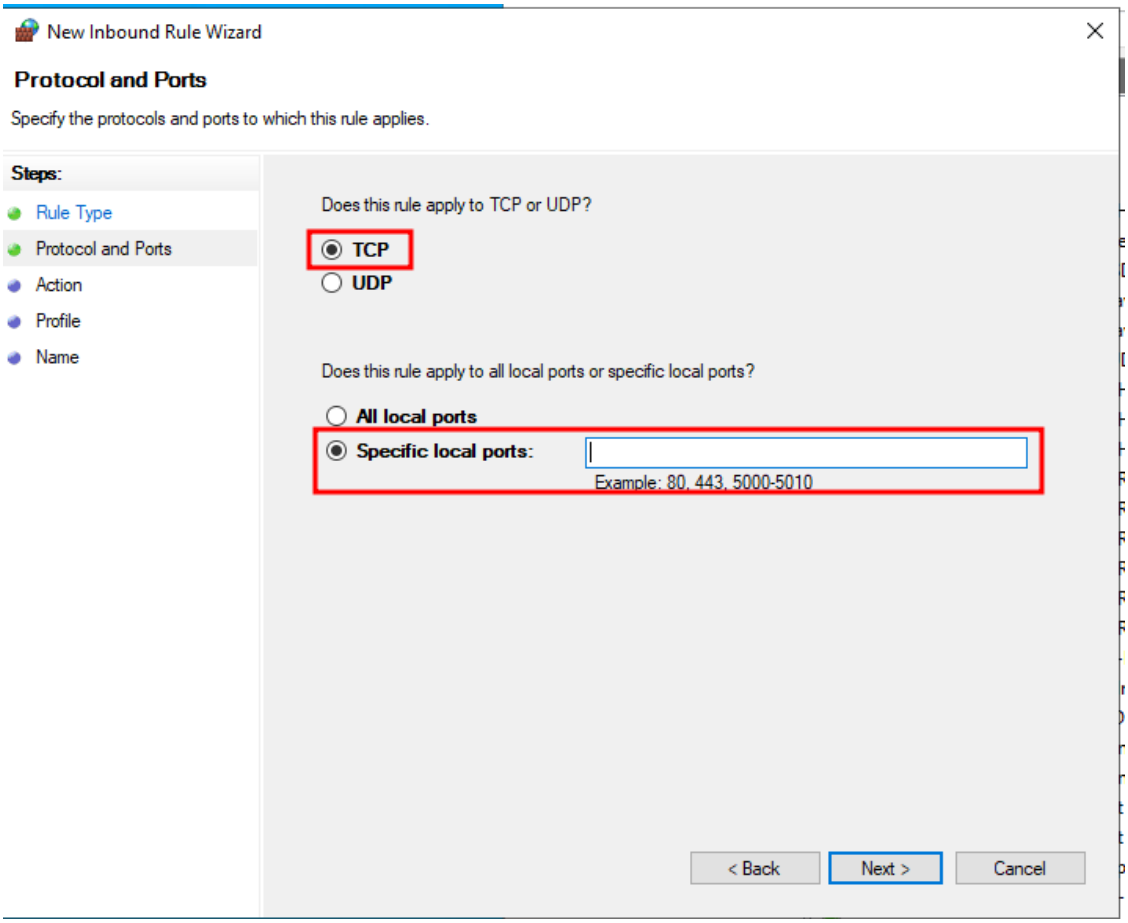
4. In the **Actions** at the right-hand pane, click **New Rule...**.



5. Select **Port** and click **Next**.

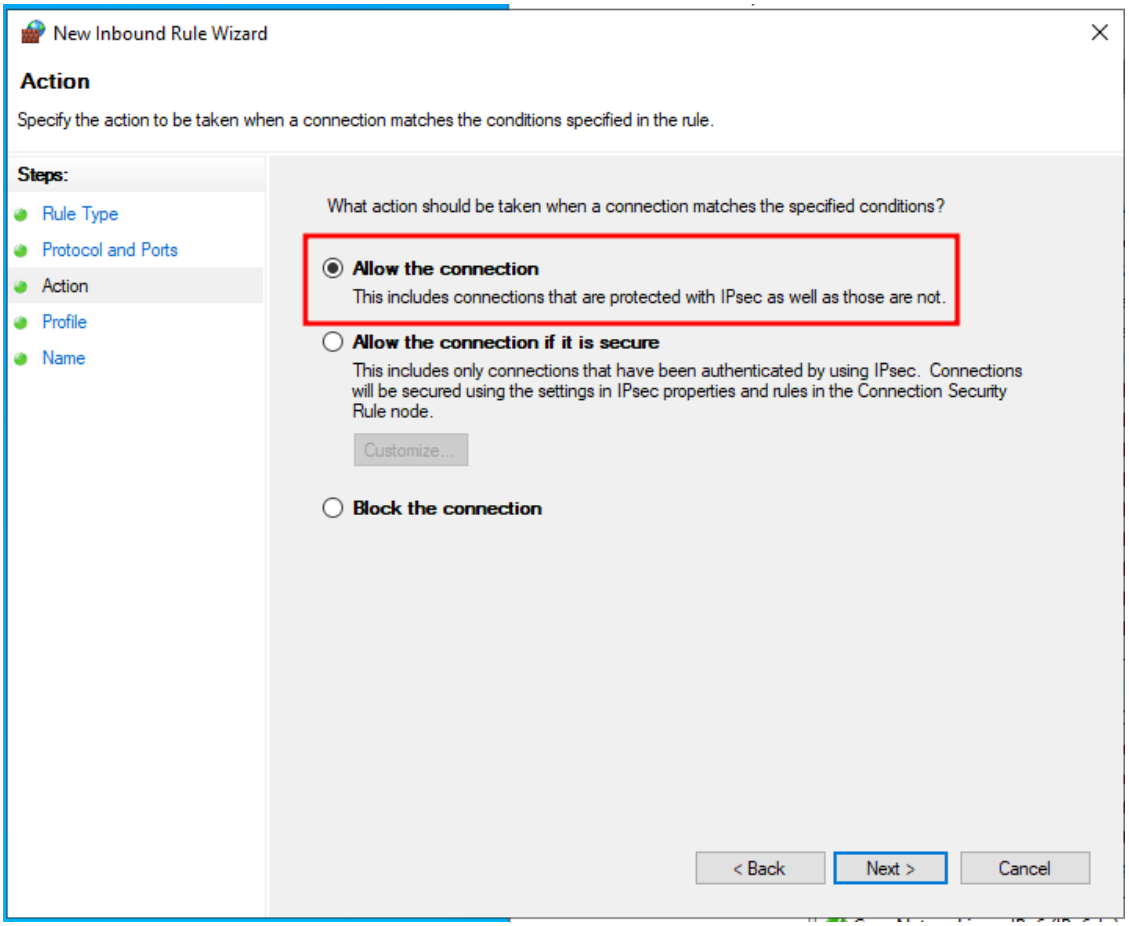


6. Select **TCP**, specify the required port number, and click **Next**.

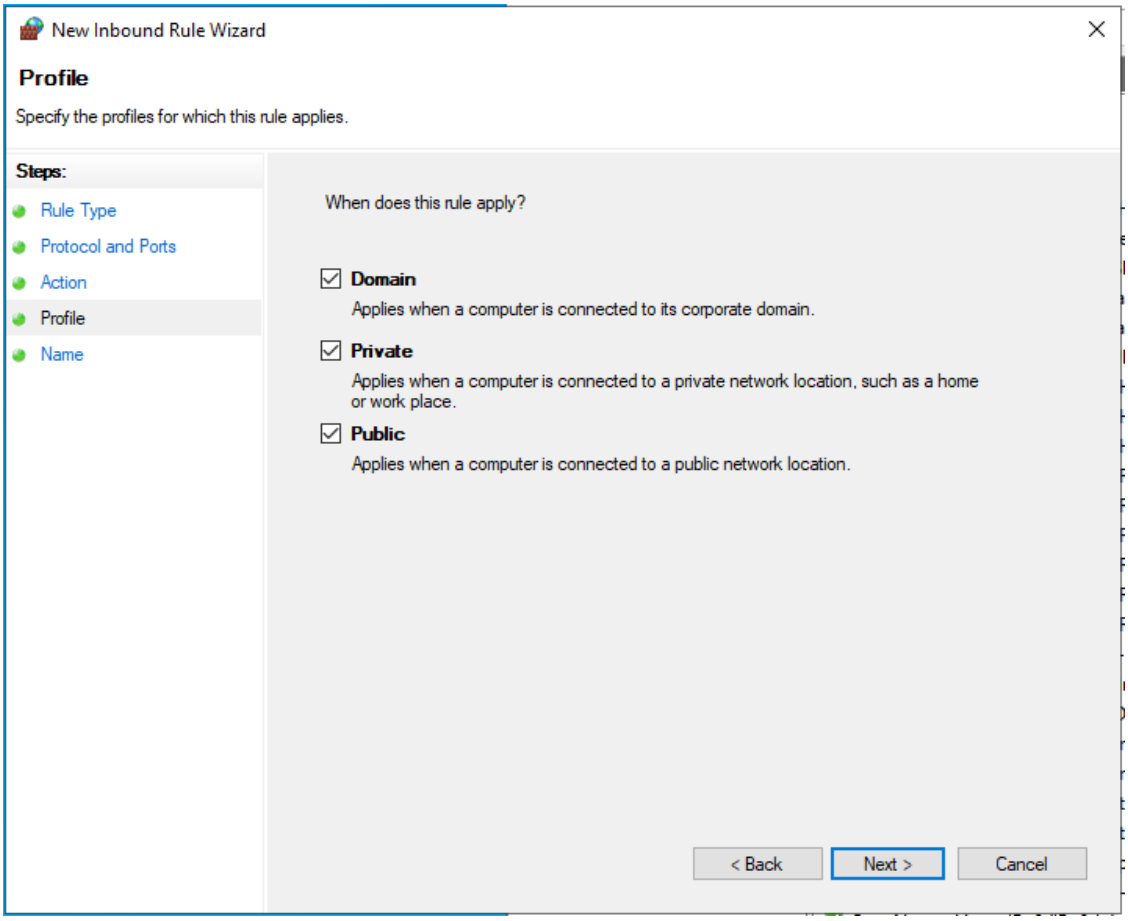


If you are unsure which protocol or port to use, refer to your application's documentation.

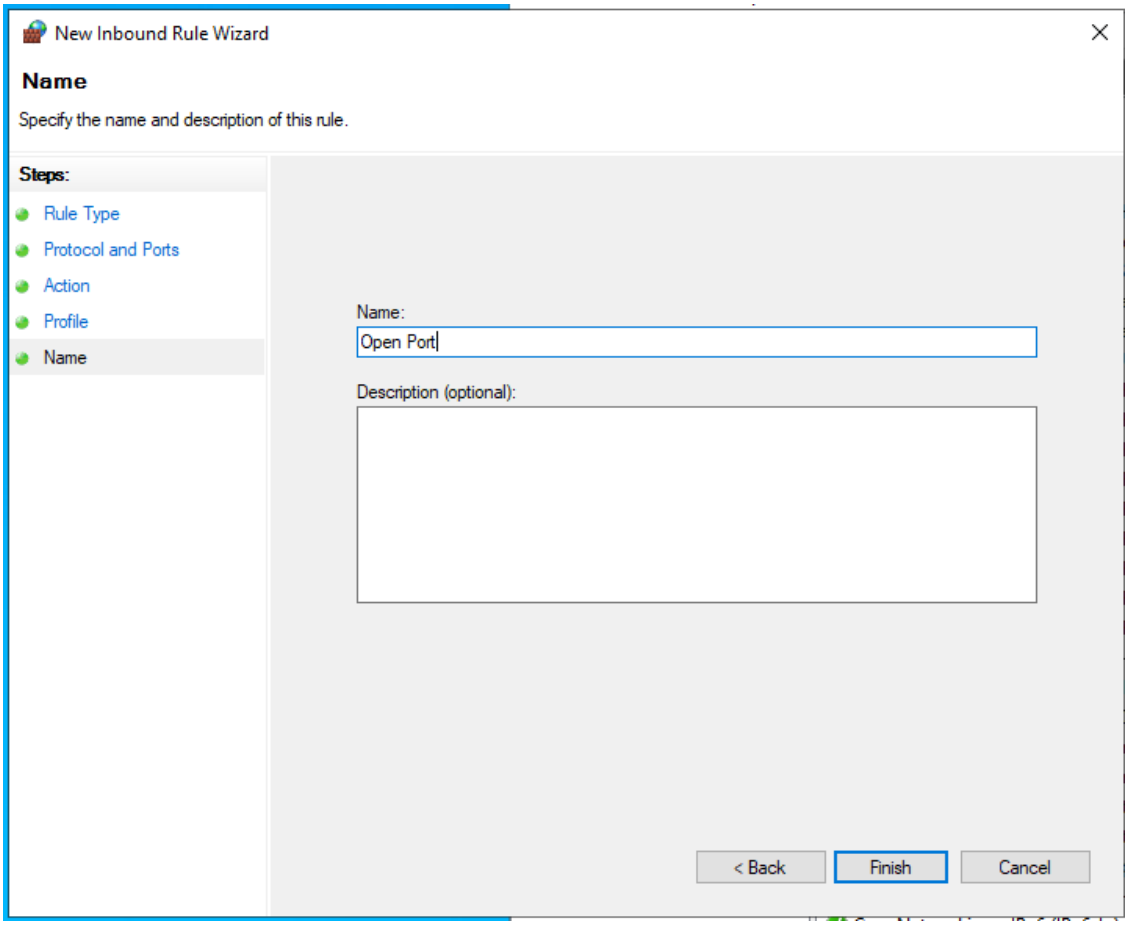
7. Select **Allow the connection**, then click **Next**.



8. Accept the default settings and click **Next**.



9. Enter a descriptive name for the rule (e.g., "Open Port”) and click **Finish**.



Option 2: Using PowerShell

1. Run **PowerShell** as **Administrator**.

2. Run the following command, replacing the placeholders with the appropriate values:

Code · powershell

```
New-NetFirewallRule `
  -DisplayName "<Rule Name>" `
  -Direction Inbound `
  -Action Allow `
  -Protocol TCP `
  -LocalPort <Port Number>
```

Uncomplicated Firewall (UFW)

1. Verify that **UFW** is running.

Code · bash

```
sudo ufw status
```

Expected output:

```
Code · bash

Status: active
```

2. Allow a TCP port, replacing the placeholders with the appropriate value.

```
Code · bash

sudo ufw allow <your port>/tcp
```

Example output:

```
Code · bash

Rule added
Rule added (v6)
```

3. Verify the firewall rule.

```
Code · bash

sudo ufw status
```

Confirm that the specified port appears in the list of allowed rules.

Firewalld

1. Verify that **firewalld** is running:

```
Code · bash

sudo firewall-cmd --state
```

Expected output:

```
Code · bash

running
```

2. Allow a TCP port. Add a permanent firewall rule:

Code · bash

```
sudo firewall-cmd --permanent --add-port=80/tcp
```

3. Reload the firewall configuration to apply the change:

JavaScript · bash

```
sudo firewall-cmd --reload
```

Python · python

```
print("hello")
```

4. Verify the firewall rule:

Code · bash

```
sudo firewall-cmd --list-ports
```

Confirm that the specified port is included in the list of allowed ports.

Release notes

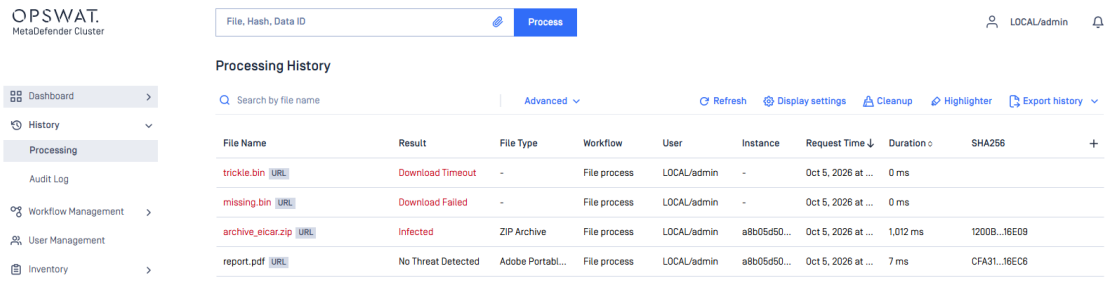
Version	**2.10.0**
Release date	5 October 2026
Scope	MetaDefender Cluster 2.10.0 introduces file scanning from URLs, extends processing result reuse to archives, and adds support for OPSWAT Alin AI Deflection. This release also improves Docker deployment, Kubernetes license management, and reliability during file processing.

New Features, Improvements and Enhancements

Scan Files from URLs

You can now submit a file for scanning by providing its URL. The **MetaDefender Cluster Download Service** downloads the file and makes it available for processing. The service also supports Docker deployments, allowing you to use URL-based scanning in these environments.

If the download times out or fails, the result shows **Download Timeout** or **Download Failed** to help you identify the problem. You can configure your workflow to classify these outcomes as Allowed.



Scan Files from URLs

The Reuse Processing Results option now supports archive files. When an archive file qualifies for reuse, MetaDefender Cluster uses its previous processing results to avoid repeating the same work. Reuse also covers CDR and DLP compression and supports results with the Exceed Archive Depth verdict.

image

OPSWAT Alin AI Deflection Integration

MetaDefender Cluster now supports the OPSWAT Alin AI Deflection engine. Deflection information is available in scan results, Processing History, and Executive Reports, so you can review these outcomes alongside other processing results.

Further Enhancements

1. Kubernetes License Management

You can now update licenses through helm upgrade. When a MetaDefender Core instance is uninstalled, its license slot is released so another instance can use it.

2. Dependency Updates

- Updated nghttp2 to version **1.70.0**.
- Updated SQLite to version **3.53.4**.

Security Enhancements

- Updated NGINX to version **1.30.5** to fix a buffer overflow vulnerability.
- Updated OpenSSL to version **3.5.9** to address security issues involving memory handling, denial-of-service, and timing side channels.

RESTful API

- `POST` `/file` now supports URL-based scanning through the **MetaDefender Cluster Download Service** using the **downloadfrom** header, with an optional **redirect-support** header for URLs returning 3xx responses without a Location header.
- The following APIs now support **Download Timeout** and **Download Failed** verdicts in their JSON responses for files submitted by URL:
  - `GET` `/file/{data_id}`
  - `GET` `/file/batch/{batch_id}`
  - `POST` `/file/batch/{batch_id}/close`
  - `GET` `/file/{data_id}/blocked-leaves`
  - `GET` `/stat/log/scan`
- When OPSWAT Alin AI Deflection is enabled, the JSON responses from `GET` `/file/{data_id}` and `GET` `/hash/{hash_id}` now include deflection information in the `deflection_info` field.

Bug Fixes

- Fixed an issue where MetaDefender Core instances could unexpectedly stop receiving new scan tasks.
- Fixed an issue where a parent scan was marked as finished while its child objects remained stuck in In Progress.

- Fixed an MetaDefender Cluster API Gateway crash when `GET` `/file/{data_id}` was called after a TLS-enabled Redis connection disconnected and reconnected.

Known Limitations

- MD Cluster Control Center does not detect duplicate workers when one is added using its IP address and another using its hostname (or vice versa). Registering the same worker multiple times may lead to incorrect license usage calculations.
- In Kubernetes deployments, terminating a MetaDefender Core instance removes it from the processing history filters and leaves an empty slot in its assigned workflows. After redeploying the instance, reassign those workflows to it.

Archived release notes

Version v2.9.1

Release Date: 15 September 2026

Improvements

Improved MD Core resource management when scanning large archives to reduce excessive memory and disk usage and prevent archive processing from affecting other scans. Archive extraction now uses dedicated processing resources and releases threads while waiting, improving stability under heavy archive workloads.

Security Enhancements

- Upgraded OpenSSL to version **3.5.8** to address known security vulnerabilities.

Bug Fixes

- Fixed an issue that could make MD Cluster Workers unusable on non-English Windows installations due to locale-dependent CPU metric collection.
- Fixed an issue that could cause Debian-based Docker image builds to fail when retrieving the `libtiff5` package required by Deep CDR.
- Fixed issues in MD Cluster File Storage gzip processing that could cause processing to hang on malformed data, accept truncated compressed content, or decompress data without a size limit.

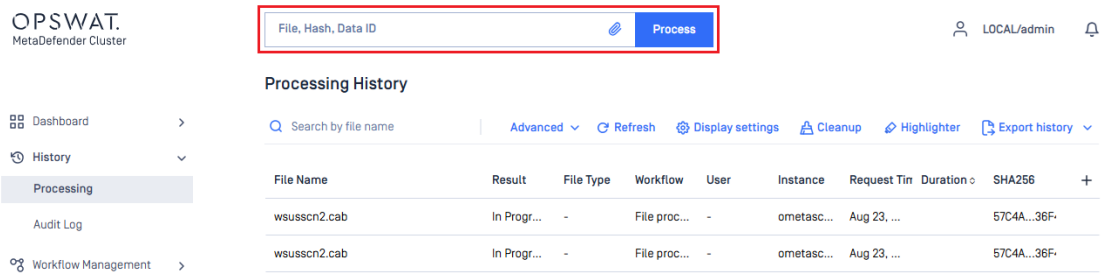
Version v2.9.0

Release Date: 26 August 2026

Scan and Access Results from Control Center

Users can now submit files for scanning directly from MetaDefender (MD) Cluster Control Center, view scan results, and look up previous results by data ID or file hash.

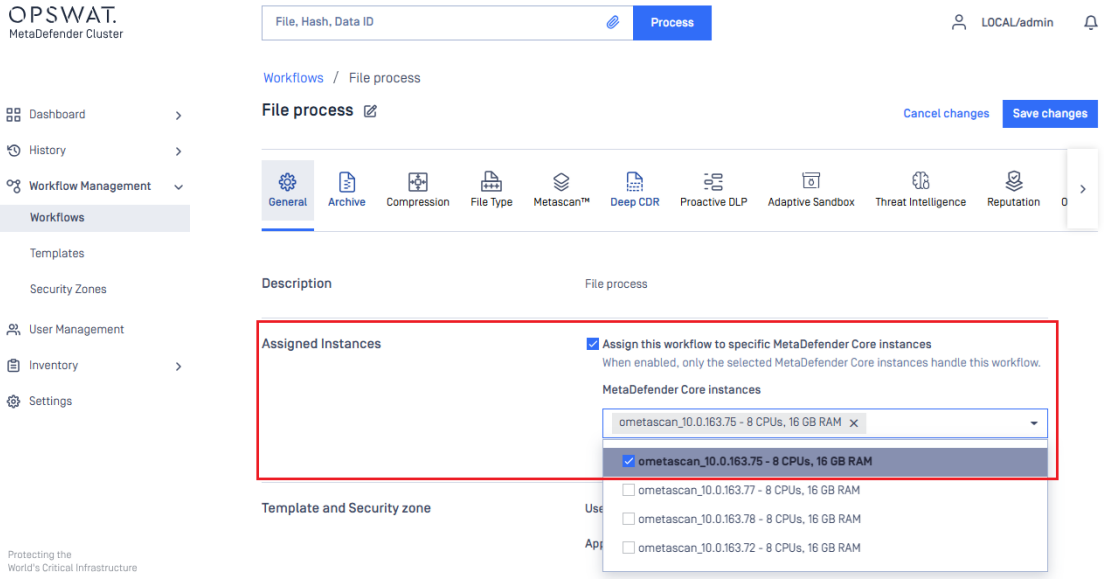
When available, sanitized or processed files can also be downloaded directly from MD Cluster Control Center. Anonymous scanning is supported when anonymous access is enabled.



Assign Specific MD Core Workers to Workflows

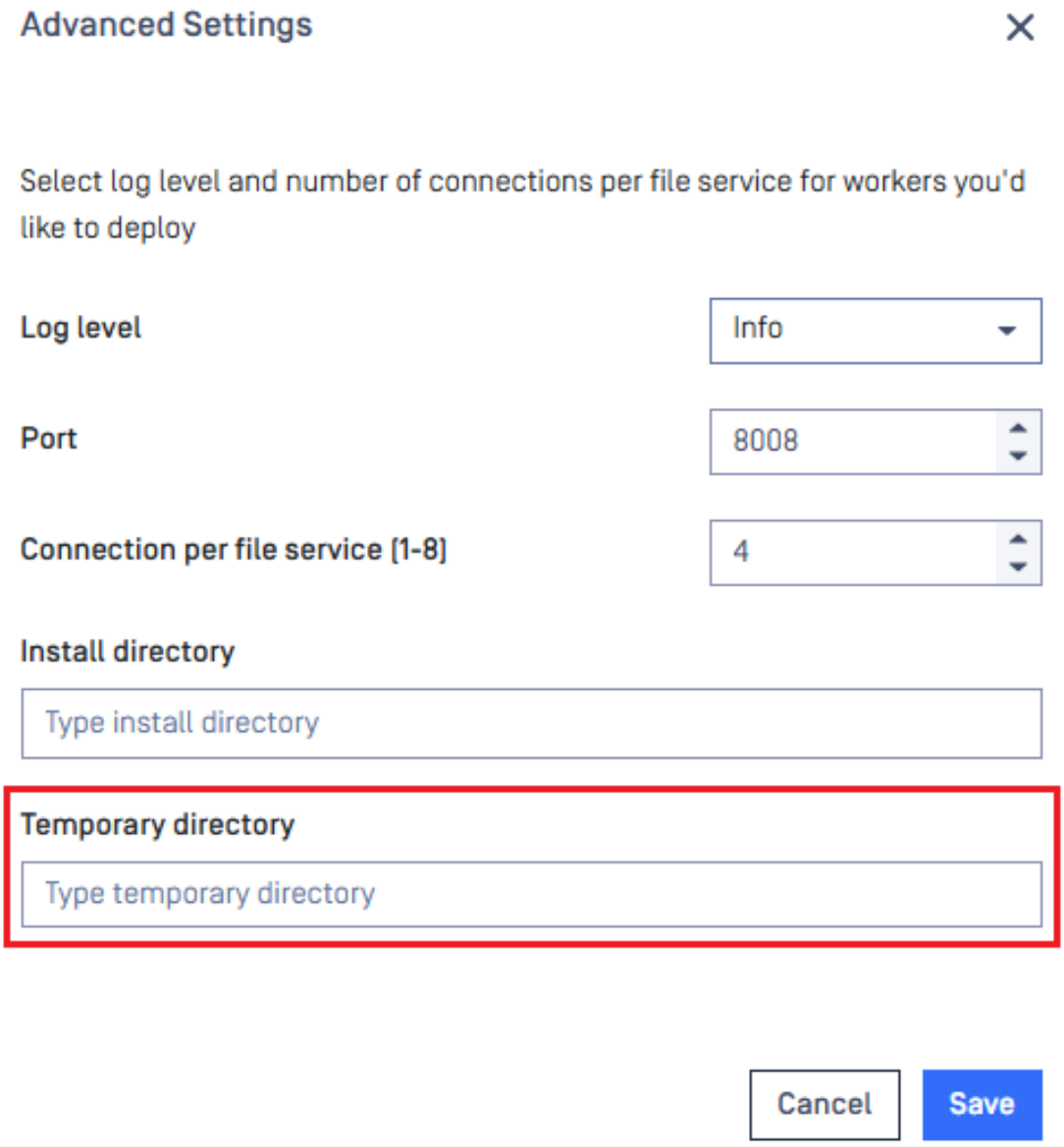
Administrators can now assign specific MD Core to individual workflows, providing greater control over which MD Core instances process requests for each workflow.

Assigned instances are displayed with CPU and memory information, making it easier to identify and manage the appropriate processing instances.



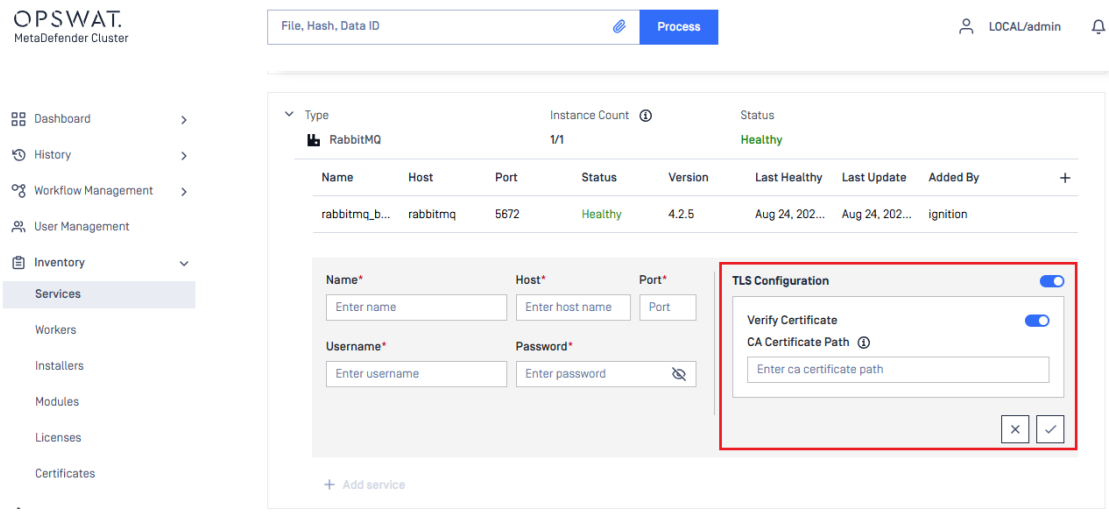
RAM Disk Support

MD Cluster now supports configuring MD Core working and temporary directories to use a RAM disk. This can reduce disk I/O and improve processing performance in environments with sufficient memory.



TLS Configuration for RabbitMQ and Redis

MD Cluster Control Center now provides TLS configuration options when adding or editing RabbitMQ and Redis service connections. Administrators can enable TLS, configure certificate verification, and specify a CA certificate path directly from Control Center.



Further Enhancements

- Kubernetes and container deployments are easier to manage with a unified Helm chart, bundled MD Core installers, simpler service import, and improved graceful shutdown.
- HTTPS is now easier to configure for Control Center and API Gateway on Kubernetes.
- Users can now select which Processing History fields to export to CSV.
- Proactive DLP now displays regex patterns imported from TSV files and lets users select which patterns to use.
- The minimum number of active AV engines can now be set to `0`, supporting Deep CDR, Proactive DLP, and other deployments that do not require antivirus engines.

Security Enhancements

- MD Cluster containers on Kubernetes now run as non-root users, with privilege escalation disabled and read-only root file systems.
- Bundled components have been updated for better security, stability, and compatibility: PostgreSQL 18.6, nghttp2 1.70.0, and SQLite 3.53.4.

RESTful API

- Added support for the `client-identity` header in the `POST` `/file` API endpoint provided by MD Cluster API Gateway.

Bug Fixes

- Fixed an issue where invalid or malformed `metadata` headers were not validated correctly or could return incorrect error messages.
- Improved MD Cluster File Storage cleanup to ensure orphaned physical files are removed correctly and do not accumulate over time.

Version v2.8.0

Release Date: 30 July 2026

Kubernetes deployment

MetaDefender Cluster can now be deployed and operated on Kubernetes, enabling scalable, cloud-native deployments across modern infrastructure.

High availability

MetaDefender Cluster now supports high-availability deployments on Kubernetes, with HA deployment options for PostgreSQL, RabbitMQ, Redis and MetaDefender Cluster File Storage.

Staged upgrades

MetaDefender Cluster now supports staged upgrades through separate Helm charts for platform services and worker instances, reducing service disruption during upgrades.

Official Docker images

Official Docker images are now available for all MetaDefender Cluster services, simplifying containerized deployments.

Graceful worker shutdown

MetaDefender Cluster Worker instances running on Kubernetes now complete in-progress processing before shutting down, helping prevent interrupted scans during scaling and upgrades.

Terraform deployment

Infrastructure as Code (IaC) support has been introduced using Terraform, allowing consistent deployment of MetaDefender Cluster across AWS, Microsoft Azure, and Google Cloud Platform. Production-ready Terraform templates are included to simplify provisioning and deployment.

Further Enhancements

- Improved container resource monitoring by accurately reporting Docker and Kubernetes CPU and memory limits instead of host resources.
- Improved stability for long-running service operations by preventing idle network timeouts through firewalls and load balancers.

Security Enhancements

- Upgraded library for vulnerability fixes: PostgreSQL to 18, NGINX to 1.30.4, OpenSSL to 3.5.7, SQLite to 3.53.3, 7-Zip to 26.02, yaml-cpp to 0.9.0, nghttp2 to 1.69.0, and Angular to 20.x.

RESTful API

- A new `GET` `/readyz` endpoint is available to allow Kubernetes and external orchestration systems to determine whether a service is ready to receive traffic.

Bug Fixed

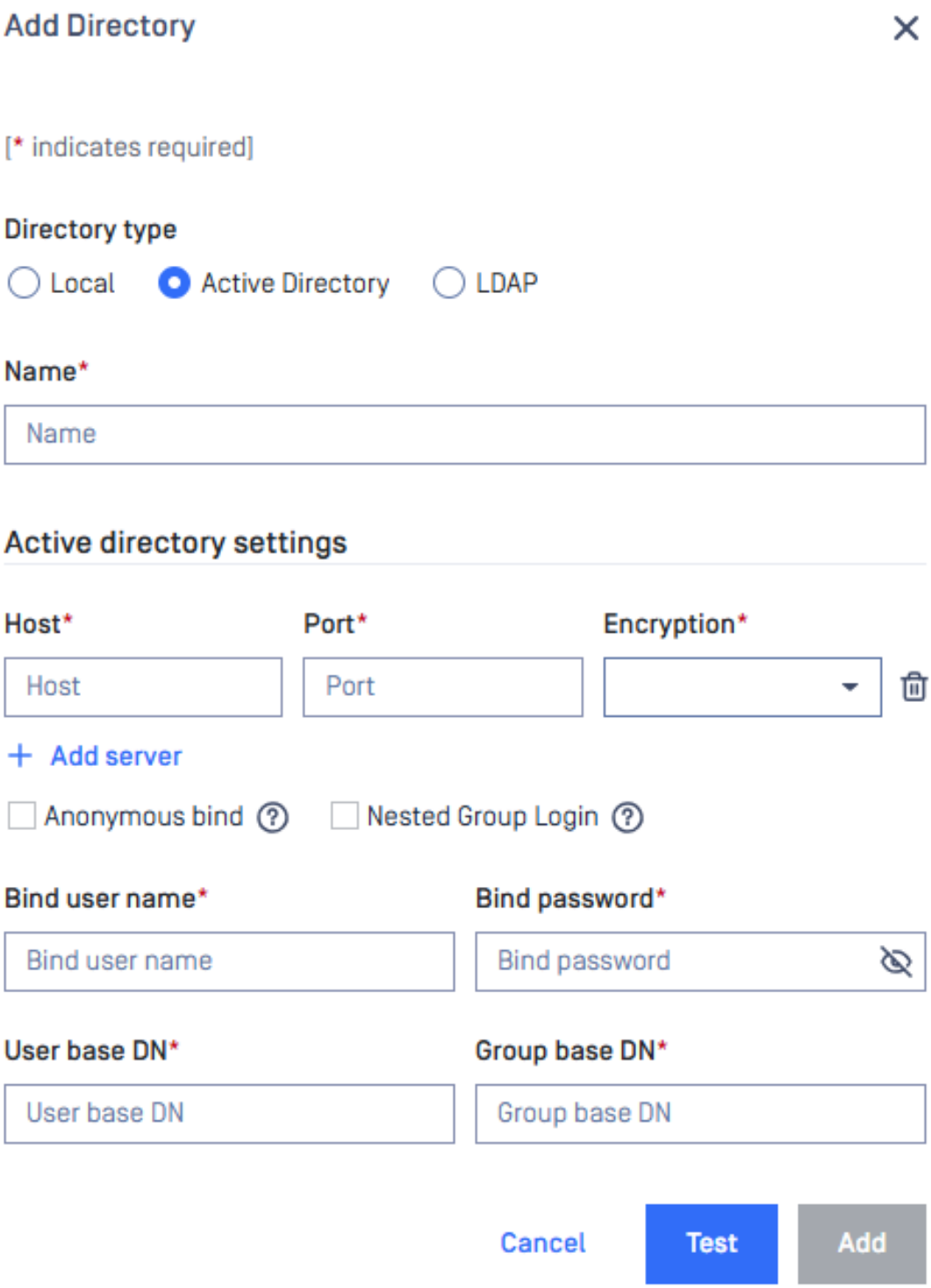
- Fixed an issue where the `GET` `/file` API incorrectly returned **404 Not Found** for files scanned by users that had later been deleted. The API now returns the scan result with an empty user field.

Version v2.7.0

Release Date: 30 June 2026

Active Directory (AD) and LDAP Authentication

MD Cluster 2.7.0 now supports user authentication through Active Directory (AD) and LDAP. Users authenticated via AD/LDAP can also generate and manage their own API keys.



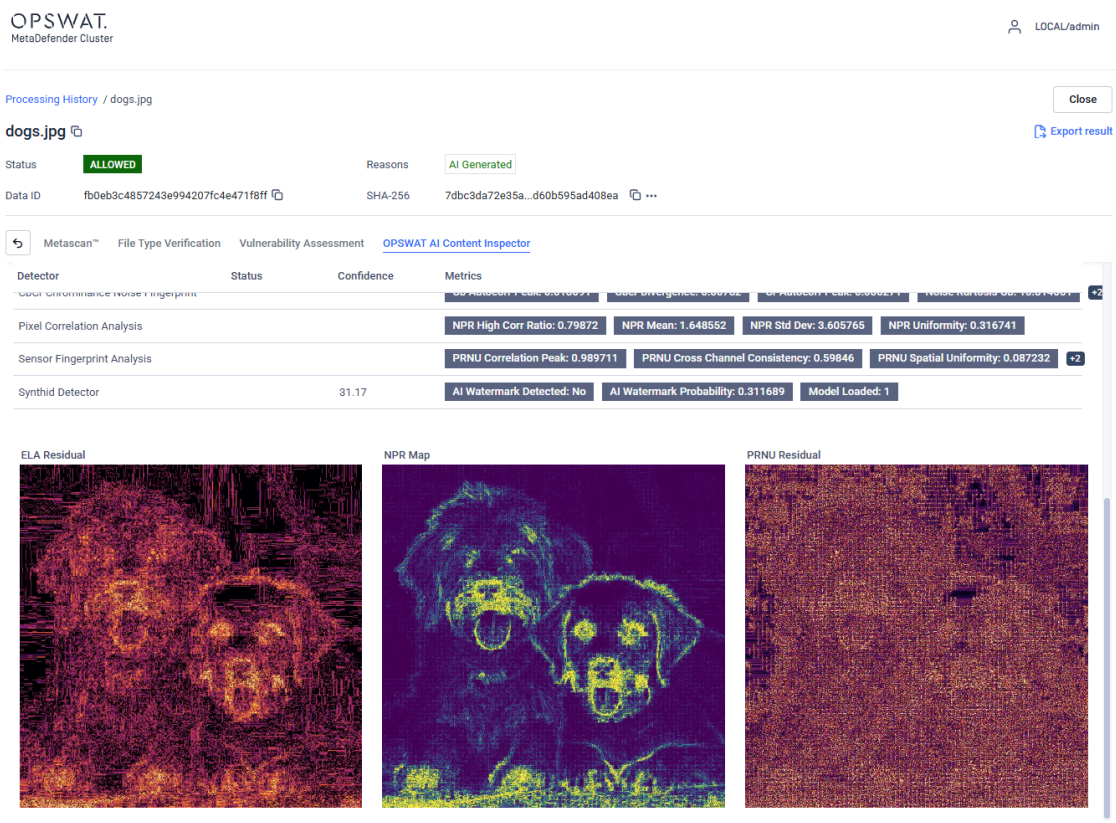
Service Connection Configuration via Ignition File

MD Cluster 2.7.0 allows service connections to be configured through the Ignition file during MD Cluster Control Center installation, simplifying automated deployments and initial setup.

```
log:⟺
rest:⟺
database:⟺
identity:⟺
secure:⟺
service:
  cache:
    connections:
      - display_name: "redis_cache"
        host: "192.168.xxx.xxx"
        port: 6379
  file_storage:
    connections:
      - display_name: "file_storage"
        host: "192.168.xxx.xxx"
        port: 8890
        connection_key: "1234abcd"
  broker:
    connections:
      - display_name: "rabbitmq_broker"
        host: "192.168.xxx.xxx"
        port: 5672
        user: "admin"
        password: "admin"
```

OPSWAT AI Content Inspector Integration

MD Cluster 2.7.0 introduces integration with OPSWAT AI Content Inspector, enabling AI-powered content analysis as part of the MD Cluster workflow.



Further Enhancements

- Service Connection Import via YAML
- Processing History Filtering by File Extension
- Certificate Renaming
- Heartbeat Support for All Services

Security Enhancements

- Introduced a more secure API key format. API keys are now 36 hexadecimal characters (`0-9`, `a-f`), containing at least 10 letters, at least 10 digits, and no more than three consecutive characters of the same type.
- Fixed a potential denial-of-service issue caused by unlimited worker addition.
- Restricted local file paths when reading certificates to improve security.
- Fixed potential SQL injection vulnerabilities.
- Restricted callback destinations to help prevent unauthorized callback requests.
- Upgraded NGINX to v1.30.3.
- Upgraded OpenSSL to v3.5.7.

RESTful API

- Added support for API endpoints to retrieve blocked leaves: GET /file/{data_id}/blocked-leaves and GET /hash/{md5|sha1|sha256|sha512}/blocked-leaves.
- The POST /file API endpoint now accepts only requests with the `application/octet-stream` content type.

Bug Fixes

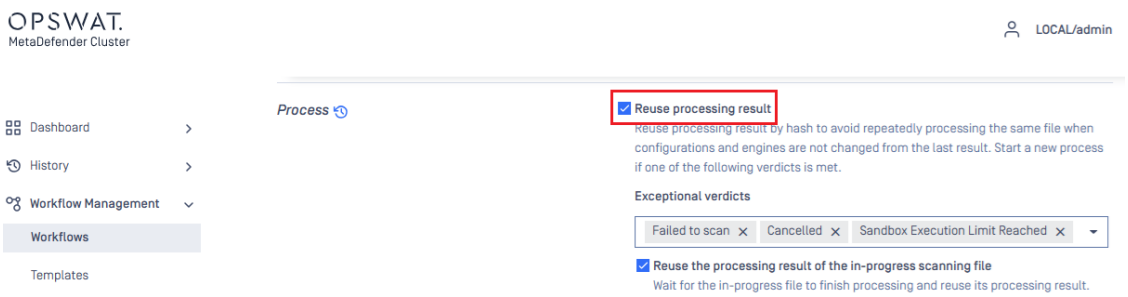
- Fixed an issue where a deployed MD Core could fail to start after being forcefully stopped or when its master Nginx process was terminated.
- Fixed an issue that could cause distributed archive extraction to hang when processing certain archive files.
- Fixed an issue where MD Cluster Control Center and Identity Service could fail to start if the PostgreSQL password contained special characters.

Version v2.6.1

Release Date: 21 May 2026

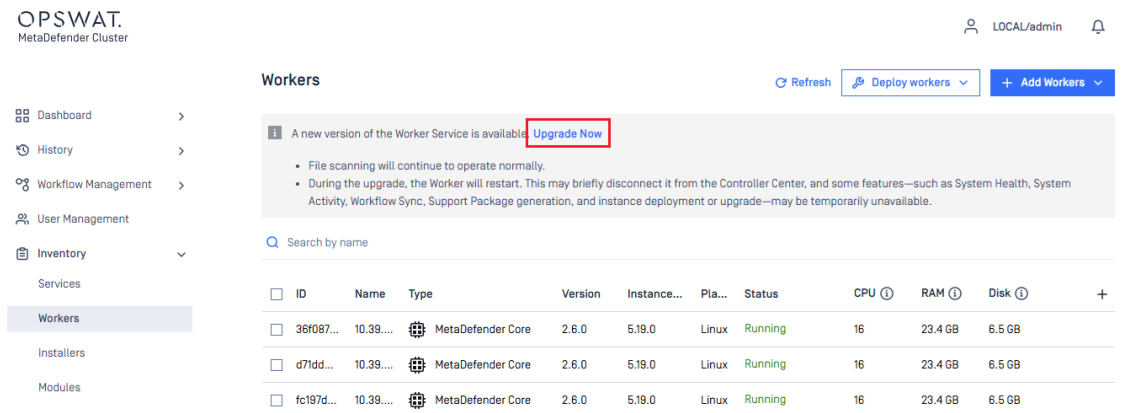
Cross-Instance Processing Result Reuse by File Hash

MD Cluster 2.6.1 supports cross-instance processing result reuse by file hash. When the same file is submitted, an MD Core instance can reuse existing processing results instead of processing the file again. If the original file is still being processed, the requesting instance waits for completion before reusing the result. This helps reduce bandwidth usage and improves resource efficiency, especially for complex tasks.



Remote Upgrade Support for MD Cluster Worker

MD Cluster 2.6.1 now supports remote upgrades for MD Cluster Workers directly from Control Center. System administrators can trigger upgrades centrally, and all connected MD Cluster Workers will be upgraded automatically without requiring manual intervention.



RESTful API

- Added `blocklisted_reason` and `allowlisted_reason` fields to the response of `GET /file/{data_id}`, `{% badge type="info" text="GET" /%}/file/batch/{batch_id}` and `GET /hash/{md5|sha1|sha256|sha512}`.

- Added support for the `{% badge type="info" text="GET" /%}/file/{data-id}/extraction-errors` API endpoint.

Further Enhancements

- Search Executive Reports by Workflow Rule Name
- Separate Protected File Verdicts
- Extraction Issue Reporting
- Archive Nesting Visibility
- Two-Year Executive Report Access

Security Enhancements

- Upgraded NGINX to v1.30.1.
- Upgraded PostgreSQL to v14.23.
- Access to available rules, password policy settings, and active performance reports in MD Cluster Control Center can now be restricted based on user permissions.

Bug fixes

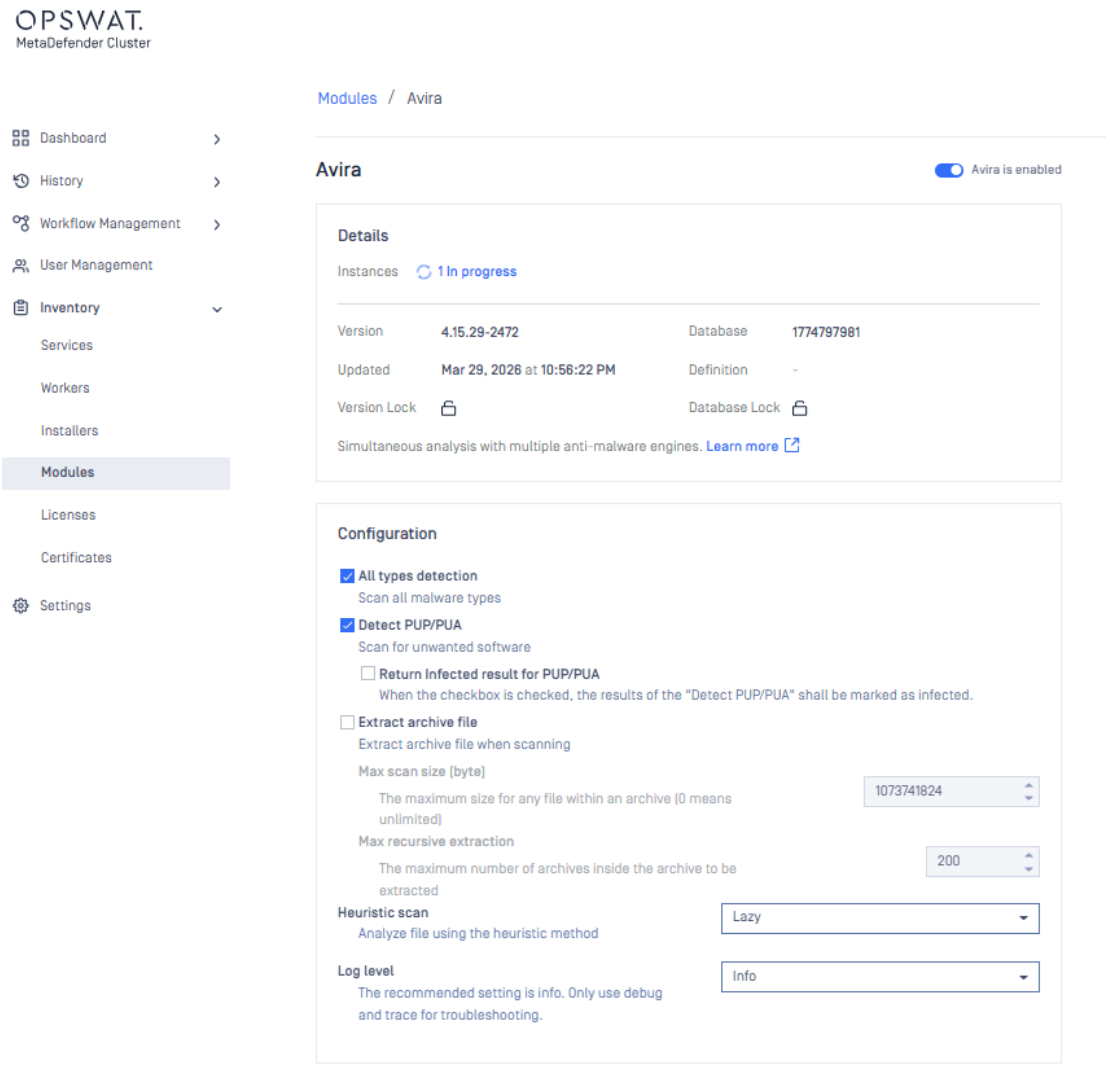
- Fixed an issue where MD Core support packages larger than 4 GB could not be generated.
- Fixed an issue that prevented an engine from being disabled on MD Cluster Control Center while it was waiting for installation in MD Core instance.
- Fixed an issue where passwords were not retained in files generated through distributed compression.

Version v2.6.0

Release Date: 15 April 2026

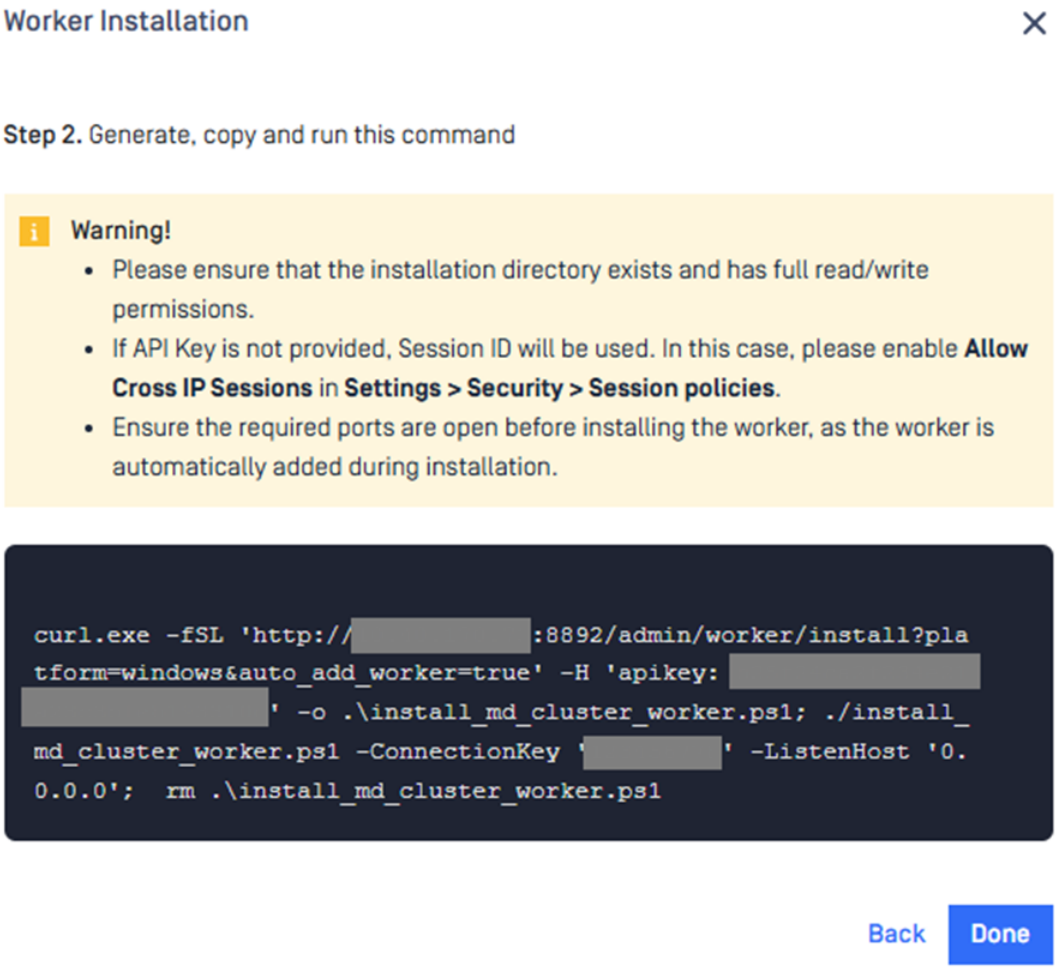
Centralized engine configuration for MetaDefender Core

System administrators can now manage engine settings for all MD Core instances from a single location. They can enable, disable, or remove engines centrally, and any changes are automatically synchronized across all instances.



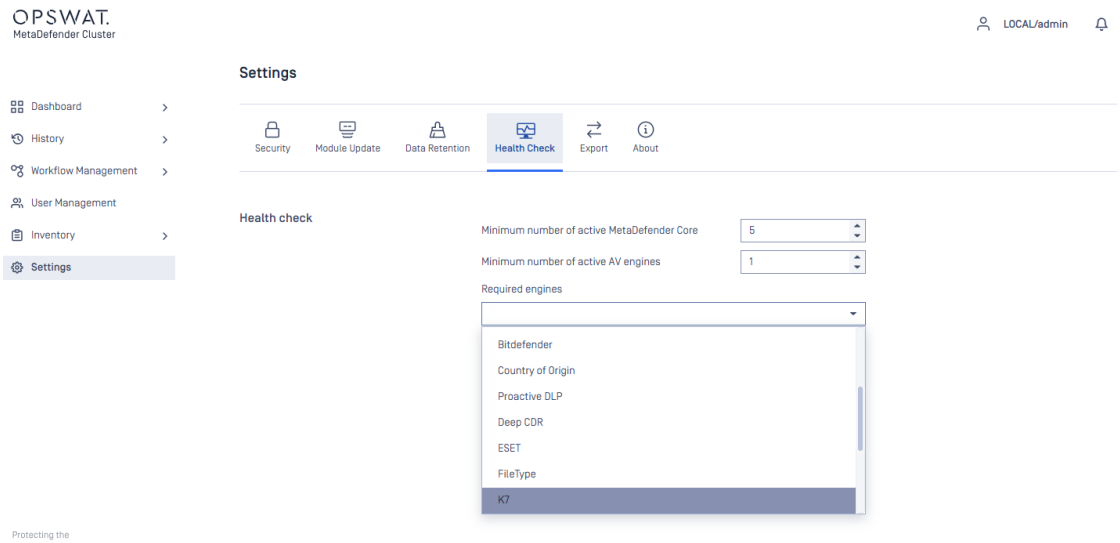
Simplified MetaDefender Worker installation

MetaDefender Cluster Worker is now bundled with MetaDefender Cluster Control Center. Users can install MetaDefender Cluster Worker through a single script available in Control Center.



Configurable minimum engine and MD Core instance requirements

Allow users to configure the minimum number of active MetaDefender Core, active AV engines, and required engines. This gives administrators more control over system readiness.



Redesigned distributed extraction flow

The distributed extraction flow has been redesigned to make better use of system resources while improving overall product stability.

Distributed compression

Distributed compression is now supported in MetaDefender Cluster, allowing archive files to be generated across multiple MetaDefender Core instances.

Further Enhancements

- Faster MetaDefender Cluster File Storage
- Faster executive report retrieval
- Custom installation directory support
- Display of Total Extracted File Size
- Display of stop reason
- Improved audit log search and filtering
- File size filter for Processing History
- MetaDefender Aether integration

Security Enhancements

- Upgraded library for vulnerability fixes: Nginx 1.29.7, 7zip 26.0.0, OpenSSL 3.5.6, Sqlite 3.51.2,PostgreSQL 14.22, zlib 1.3.2, Nghttp2 1.68.1, tikio-rs/bytes 1.11.1, quinn-proto 0.11.14, and webpki 0.103.10.
- Address a high-severity SQL injection vulnerability in the Control Center search functionality. The fix ensures that all search inputs are properly validated and handled using secure query mechanisms to prevent injection attacks.
- Fix a Denial-of-Service (DoS) vulnerability affecting endpoints that support resource listing and allow clients to specify the number of resources to be returned in a single request.

Bug fixes

- Fix an issue where some fields were missing in the JSON response of `GET /file/{batch_id}` when the request was returned through MetaDefender Cluster API Gateway.
- Fix a potential MetaDefender Cluster Worker crash that could occur when DEBUG logging is enabled.
- Fix an issue that prevented Proactive DLP engine from being deployed on Windows due to the long file path limitation.

Version v2.5.2

Release Date: 26 January 2026

RESTful API

- Added support for the `current_finished_files` field in the responses of `GET /batch/{batch_id}` and `GET /file/{batch_id}`.

Further Enhancements

- Optimized data lake queries to speed up retrieval of processing history records.
- Prevented multi-platform MD Core deployment to ensure consistent processing results.
- Added IPv6 support for essential MD Cluster services to improve modern network compatibility.
- Enabled fetching and installing OPSWAT Predictive Alin AI packages on MD Core for automated engine setup.

Security Enhancements

- Upgraded Nginx to 1.29.4 for vulnerability fixes.
- Addressed a security vulnerability that could permit unauthenticated remote code execution within MD Cluster Control Center and API Gateway.
- Addressed a vulnerability in user management operations that could allow SQL injection and potentially cause Denial of Service.

Bug Fixes

- Corrected inaccurate display of outdated scan engines in System Health.
- Resolved a problem that allowed selection of the License Management Service Activation rule even when using an invalid token.
- Fixed an issue where an MD Core instance’s active license could become permanently inactive after switching to a different license.
- Corrected a problem that allowed system admins to deactivate an online license that had already expired.

Version v2.5.1

Release Date: 16 December 2025

Support licensing with License Management Server (LMS)

New licensing management model for MetaDefender Cluster to allow license management server to manage the product's license status.

Further Enhancements

- Improve the performance of fetching processing history.
- Support sorting processing history by duration.
- Enhance product stability.

Security Enhancements

Upgraded library for vulnerability fixes:

- PostgreSQL v14.20

Bug Fixes

- Fixed an issue where certain archive file remained stuck in the in-progress state.
- Fixed an issue that led to the process start time of batch is invalid.

- Fixed a crash in MD Cluster Control Center when adding a worker using IPv6.
- Addressed incorrect status reporting of MD Cluster Worker when a machine was unexpectedly powered off.
- Fixed an issue where the MD Cluster Worker was unable to upgrade its hosted MD Core.

Version v2.5.0

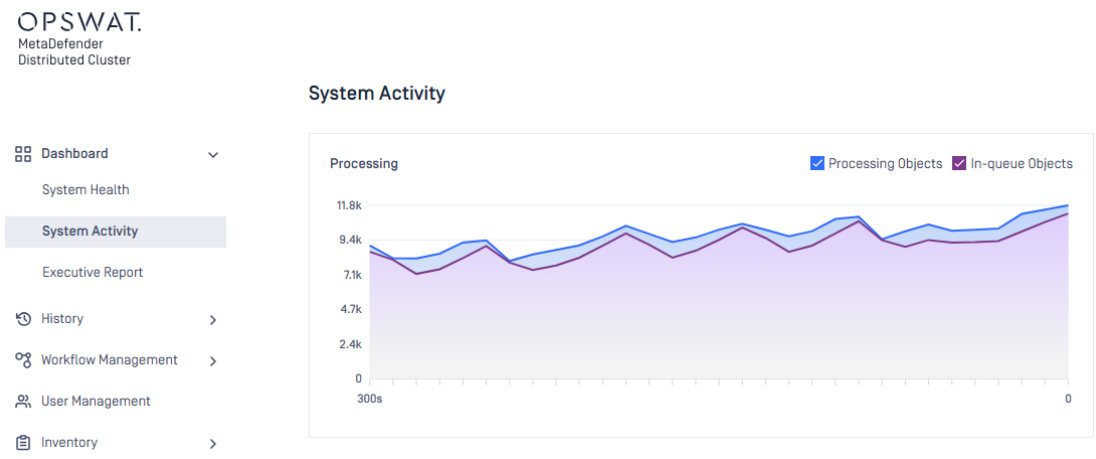
Release Date: 30 October 2025

MetaDefender Cluster license

New licensing model requires a MetaDefender Cluster license.

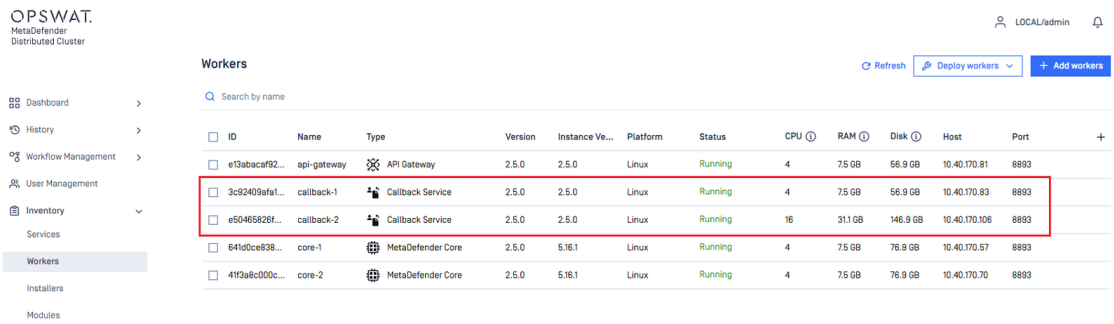
Chart for in-queue objects in System Activity

A new chart in System Activity shows the overall count of objects pending processing.



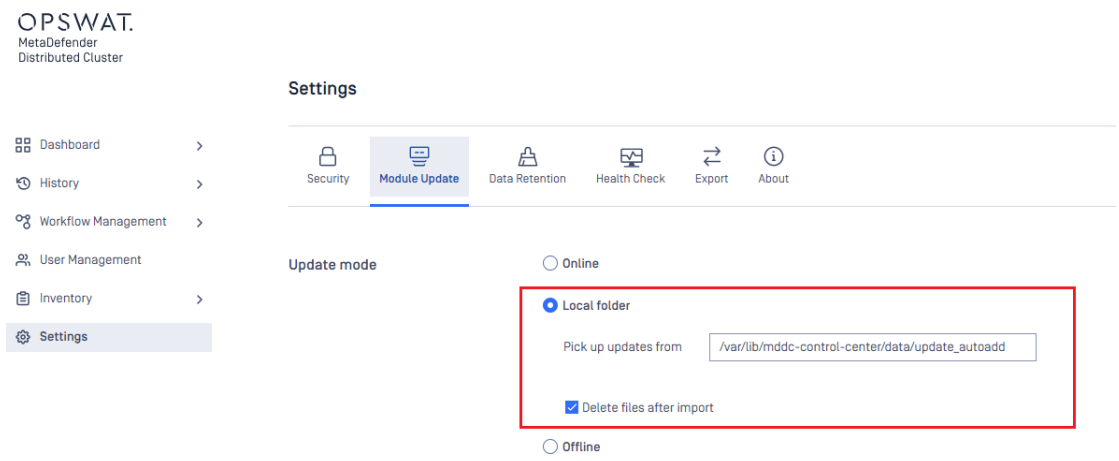
Dedicated callback service

A new optional service is introduced to dedicate sending the results of finished scan requests to the webhook server. The service can be scaled out easily and upgraded seamlessly.



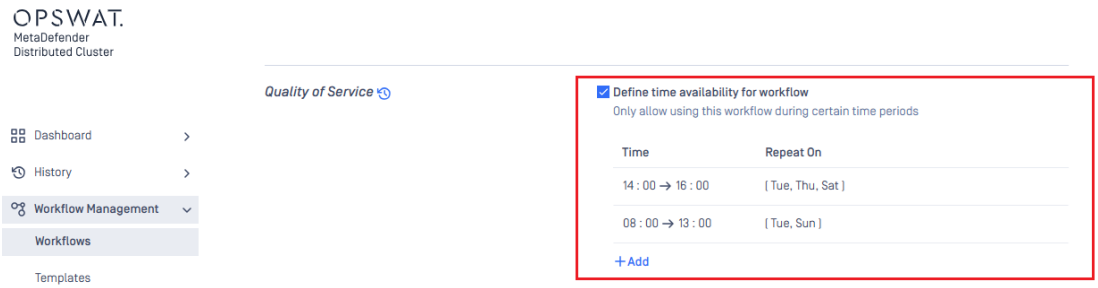
Engine package update initiated by folder monitoring

Modification of engine packages in a given folder can be monitored and notified to MetaDefender Core instances for engine updates.



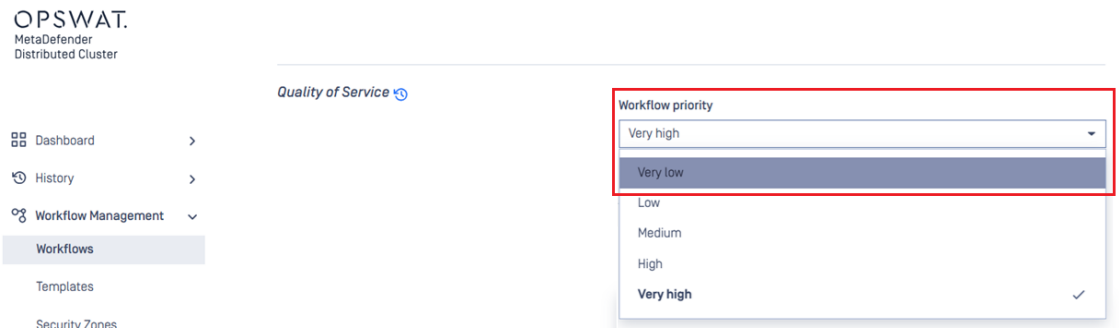
Time availability for scan request acceptance

MetaDefender Cluster API Gateway will accept scanning requests from clients during a pre-define time windows.



Workflow priority

MetaDefender Cluster Control Center allows system administrators to configure the priority of a specific workflow.



RESTful API

MetaDefender Cluster API Gateway:

- Introduce `metadata` request header to include vulnerability details in the response of `GET /batch/{batch_id}/certificate` API.
- Introduce `GET /file/webhook/{data_id}` API to retrieve the callback status of a scan request using `data_id`.
- Support `callbackurl` header in requests to `POST /file` API.
- Include callback service status in `GET /readyz` API.

MetaDefender Cluster Control Center:

- Introduce PUT and GET `/admin/config/sessioncookie` to modify and get session cookie attributes.
- Drop support for `license_id` field in POST `/admin/worker/deploy` API.
- Support callback service installer for POST `/admin/installer` API.
- Introduce a new `callback-service` type for the APIs: POST `/admin/worker/deploy`, POST `/admin/worker/upgrade` and GET `/admin/worker/upgrade/version`.

Further Enhancements

- Improve the performance of gathering materials for the executive report.
- Capability to configure the communication port of MetaDefender Core during deployment.
- Support instance name filtering for processing history exported in STIX/CSV format.
- Include metrics for waiting time related to file type detection in the executive report.

Security Enhancements

Upgraded library for vulnerability fixes:

- OpenSSL 3.5.4

Bug Fixes

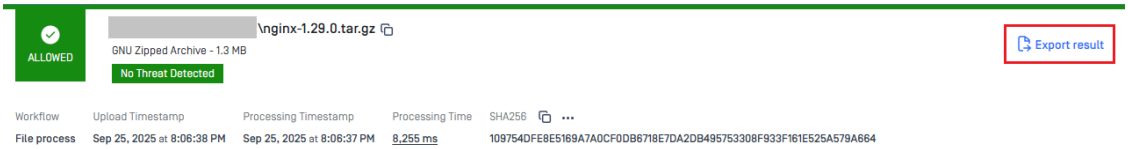
- Fixed the issue that led to the disappearance of MetaDefender Core instances from Instance Activity during high load.
- Fixed the issue that led to MetaDefender Core instances losing their licenses after the upgrade.
- Fixed the issue that caused GET `/file/{data_id}` API to return a zero `upload_time`.
- Fixed the issue that led to the omission of scan results for files within a batch when a filter was applied.

Version v2.4.0

Release Date: 30 September 2025

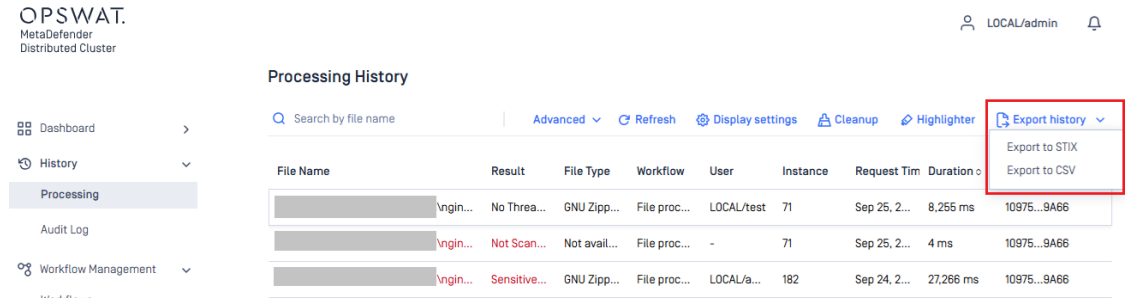
Export scan result in JSON format

From MetaDefender Cluster (MD Cluster) Control Center, users can export scan result in JSON format.



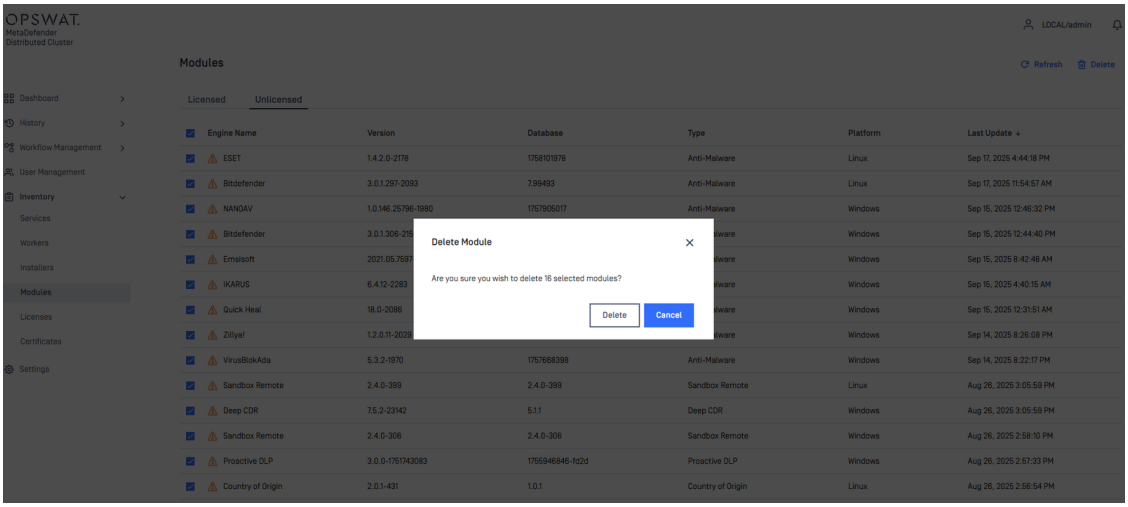
Export processing history in STIX or CSV format

Processing history can be exported in STIX or CSV format from MD Cluster Control Center.



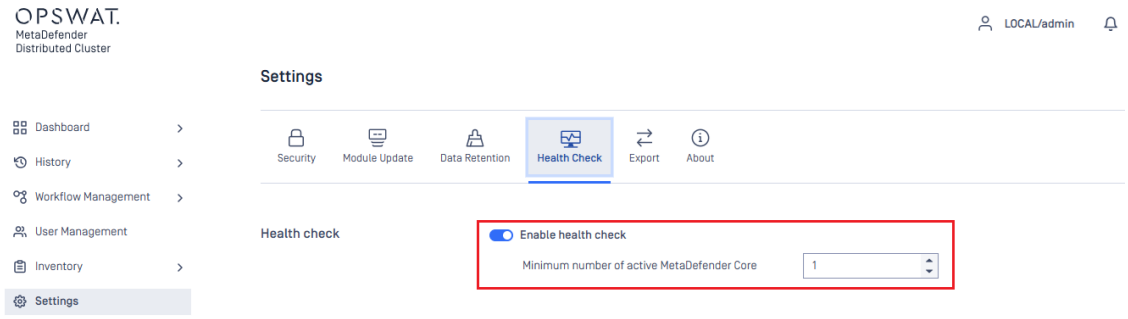
Remove abandoned module packages

Abandoned module packages can be selected and removed on web console of Control Center.



Customize the system health check

System administrators can enable the health check option and set the minimum number of required MetaDefender Core instances in the Health Check settings of the MD Cluster Control Center.



RESTful API

- Introduce a new API endpoint in MetaDefender Cluster API Gateway to verity if the system is ready for new scan requests `GET /readyz`.
- Introduce a new field, `dlp_wait_time`, in the response of `GET /file/{data_id}` API requested from MetaDefender Cluster API Gateway.
- Include `username` field in the response of `GET /file/{data_id}`, `GET /file/batch/{batch_id}` and `GET /has`

`h/{md5|sha1|sha256|sha512}`.

Further Enhancements

- Verify the minimum version requirement when adding a new instance of Redis, RabbitMQ, and PostgreSQL to MetaDefender Cluster Control Center.
- Improve storing scan results from AV engines to MetaDefender Cluster Data Lake.

Security Enhancements

Upgraded library for vulnerability fixes:

- OpenSSL 3.5.2

Bug Fixes

- Fixed the issue that caused occasional service crashes when halted.
- Fixed the issue that made it impossible to close a batch if its name contained special characters.
- Fixed the issue that led to the batch name not appearing in the UI of MD Cluster Control Center.
- Fixed the issue that caused the COO engine to fail or time out during installation.
- Fixed the issue that caused the executive report to eventually miss data.

Known limitations

MetaDefender Core becomes unlicensed following the MD Cluster Worker upgrade.

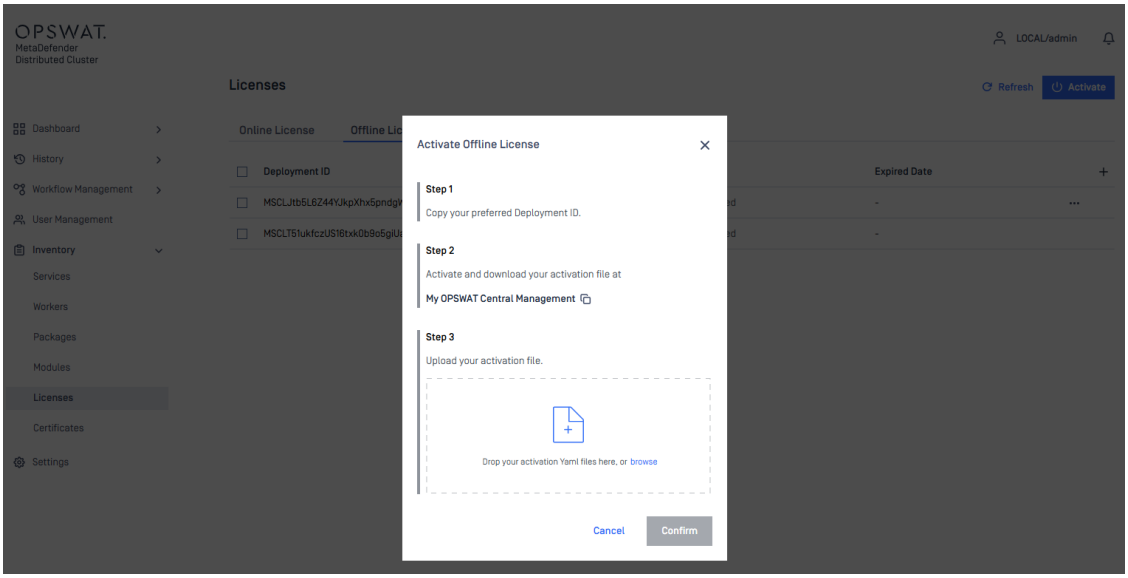
- If online activation is used, follow Online Activation activate MetaDefender Core again.
- Otherwise, follow Offline Activation.

Version v2.3.0

Release Date: 28 August 2025

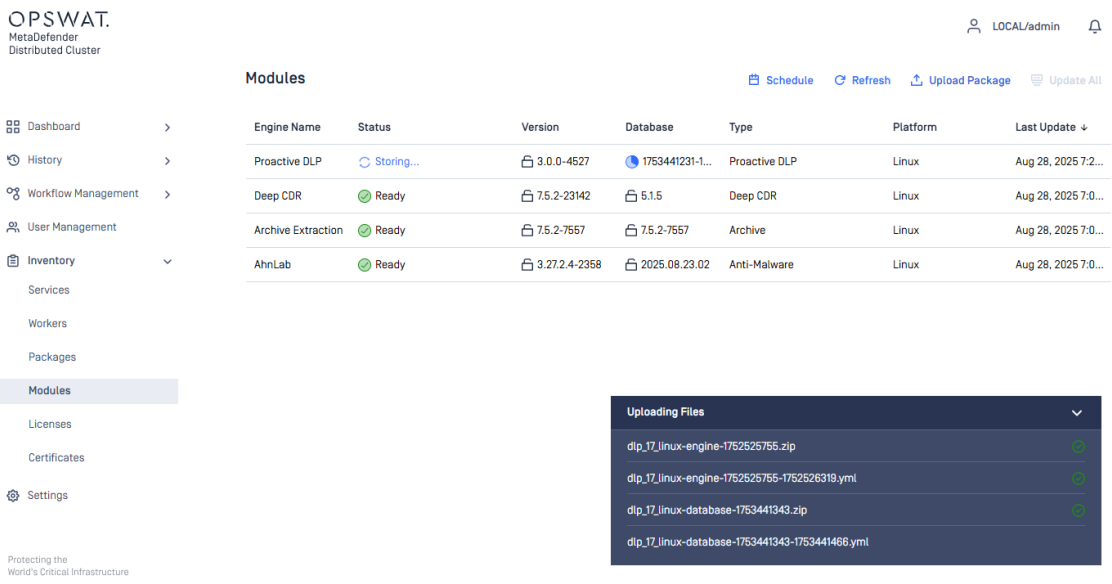
Offline License Activation

Offline license activation of MetaDefender Core instances within the cluster is supported for air-gapped environment.



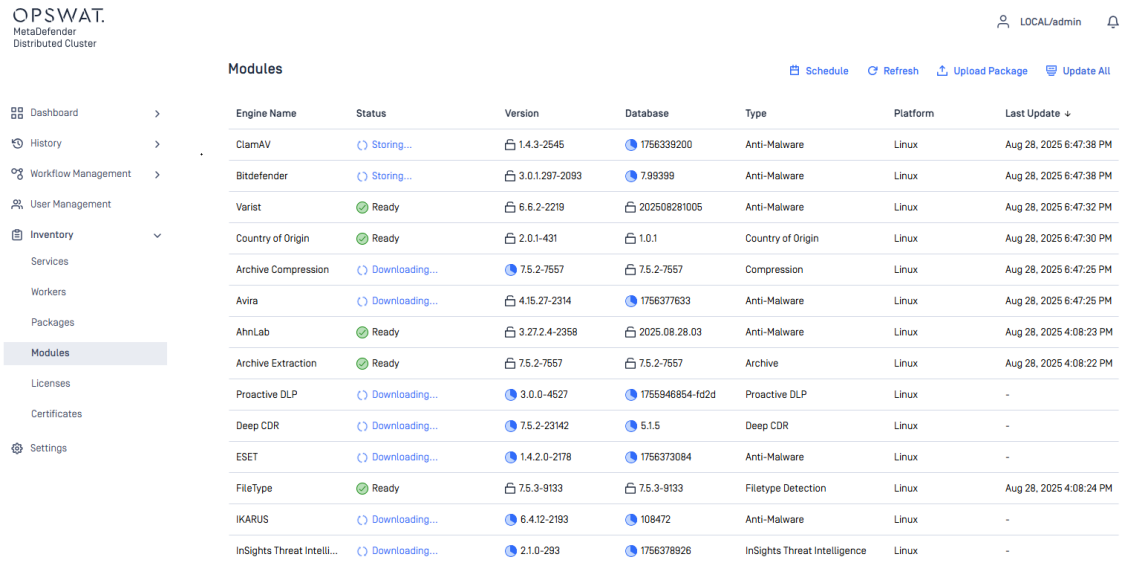
Offline Engine Package Upload

In offline environment, administrators can manually upload engine packages to MetaDefender Cluster Control Center.



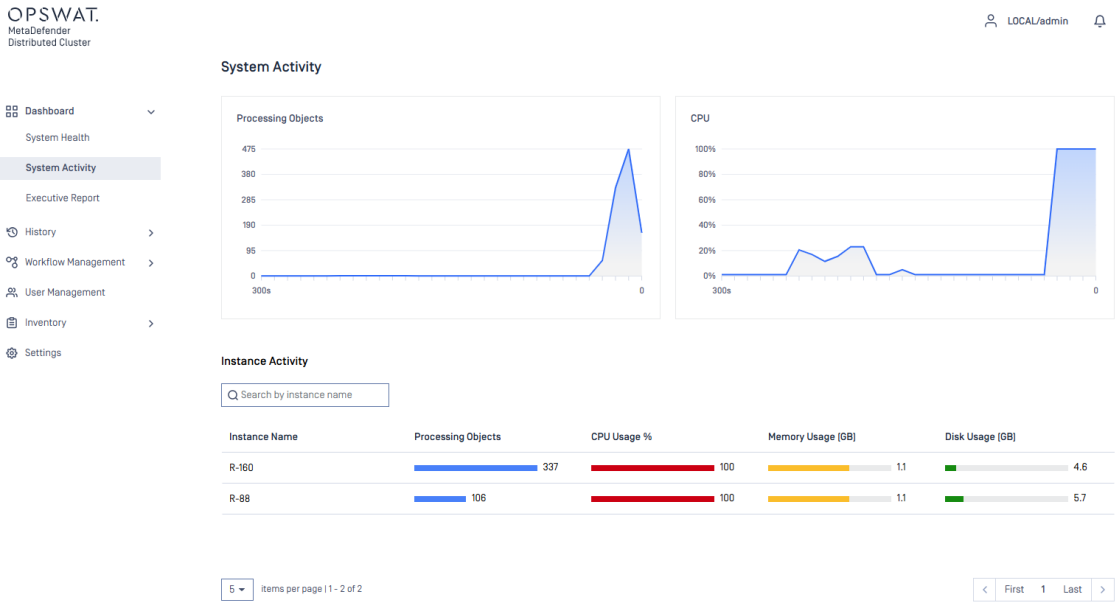
Centralized Online Engine Package Update

At update time, engine packages are obtained once from the cloud and shared with all MetaDefender Core instances within the cluster to retrieve, install, or update.



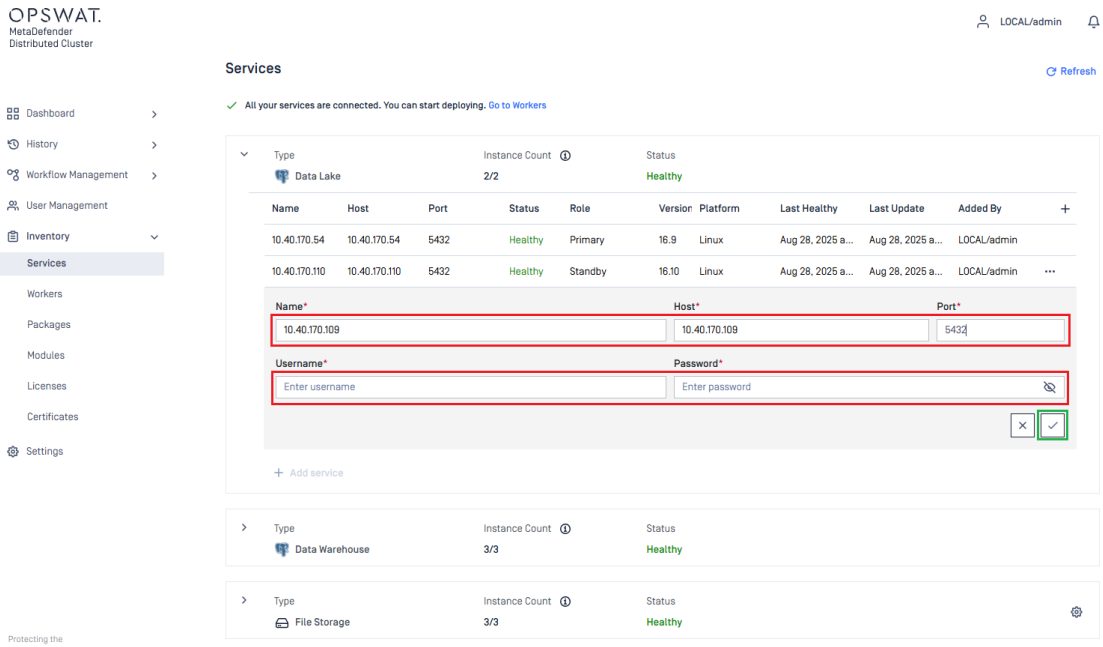
System and Instance Activities

System Activity page allows administrators to track the total processing objects, average CPU usage of the entire system, processed objects and other resources consumed by individual MetaDefender Core instances, all in one place.



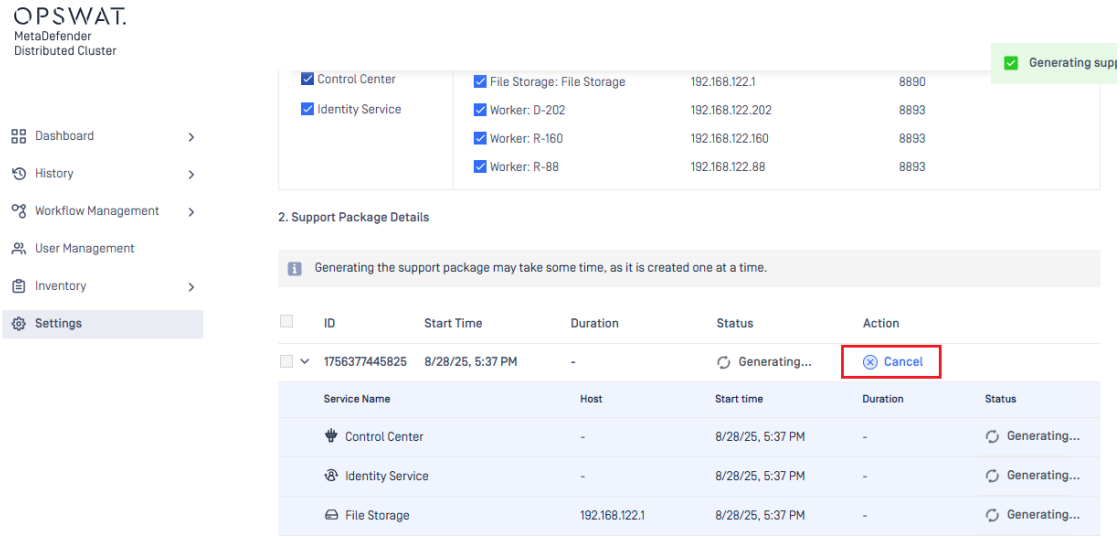
High Availability support for PostgreSQL Data Lake

Administrators can add multiple PostgreSQL instances to ensure high availability for Data Lake.



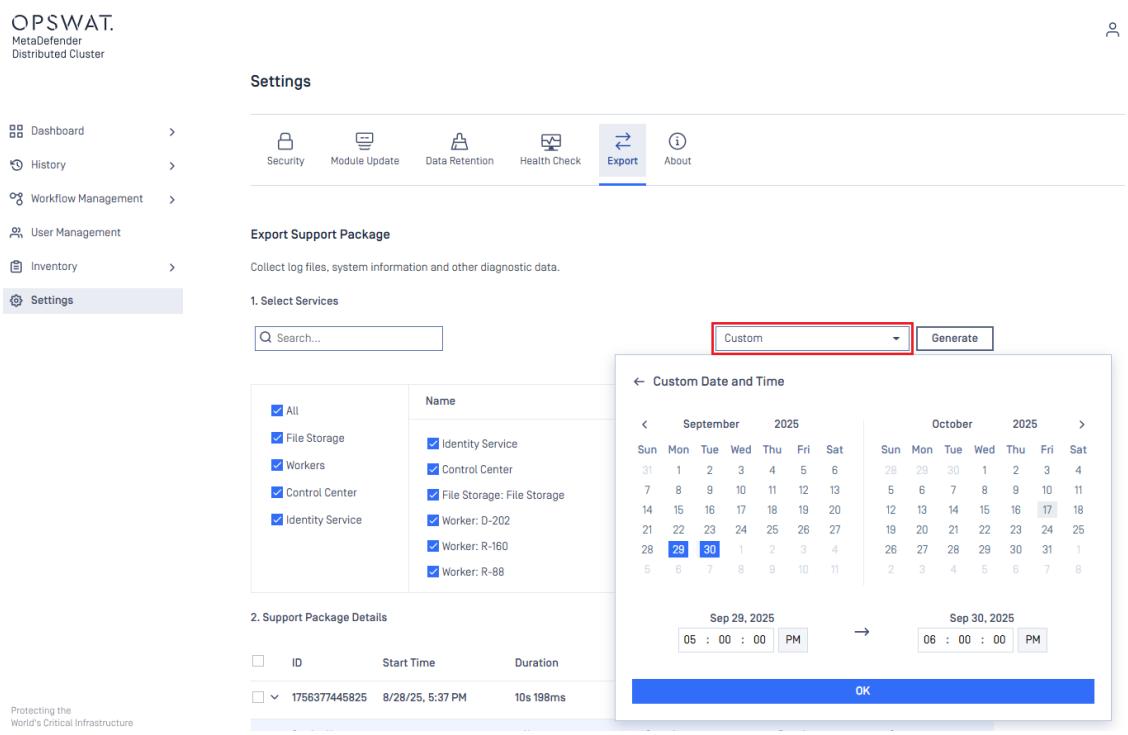
Cancellation of Remote Support Package Gathering

Administrators have the option to cancel the gathering of Remote Support Package from the MetaDefender Cluster console at any point. The cancellation is carried out at the earliest opportunity.



Support Package Gathering within a specified timeframe

MetaDefender Cluster enables administrators to define a time period for gathering support packages.



RESTful API

- Introduce a new API endpoint in MetaDefender Cluster API Gateway to fetch information on the longest expiry active license `GET` `/admin/license`.
- Introduce a new field, `filetype_wait_time`, in the response of `GET` `/file/{data_id}` API requested from MetaDefender Cluster API Gateway.

Security Enhancements

Upgraded libraries for vulnerability fixes:

- 7zip 25.01
- Nginx 1.28.0
- PostgreSQL 14.19

Bug Fixes

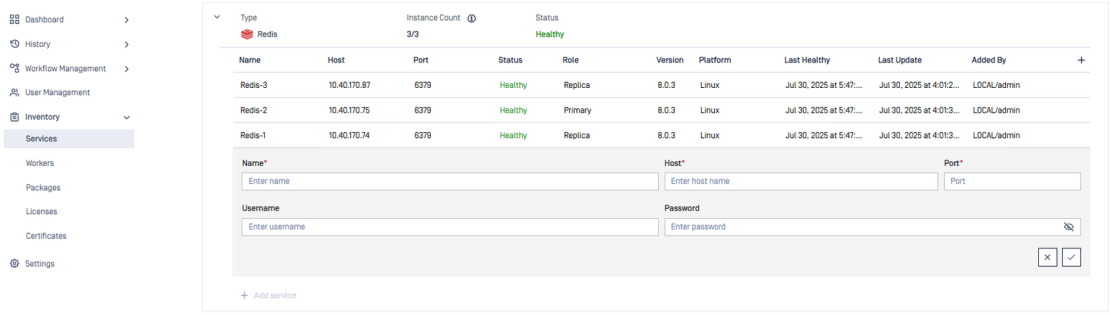
- Fixed an issue that caused the MetaDefender Cluster Control Center to occasionally crash upon stopping.
- Fixed an issue that caused session timeouts to exceed the configured duration.

Version v2.2.0

Release Date: 30 July 2025

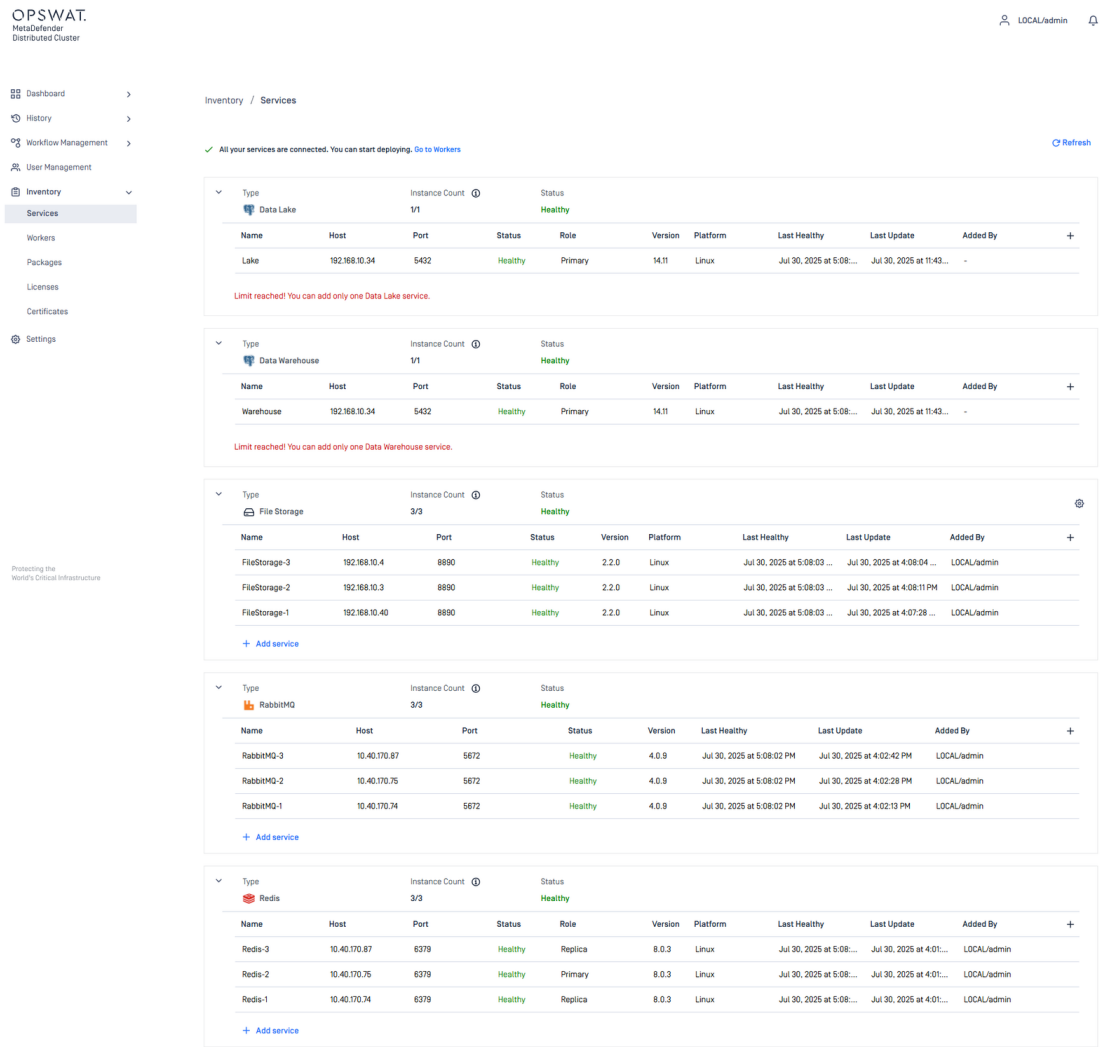
Support adding Redis Caching server with username and password

Administrator can add Redis Caching server including username and password to MetaDefender Cluster.



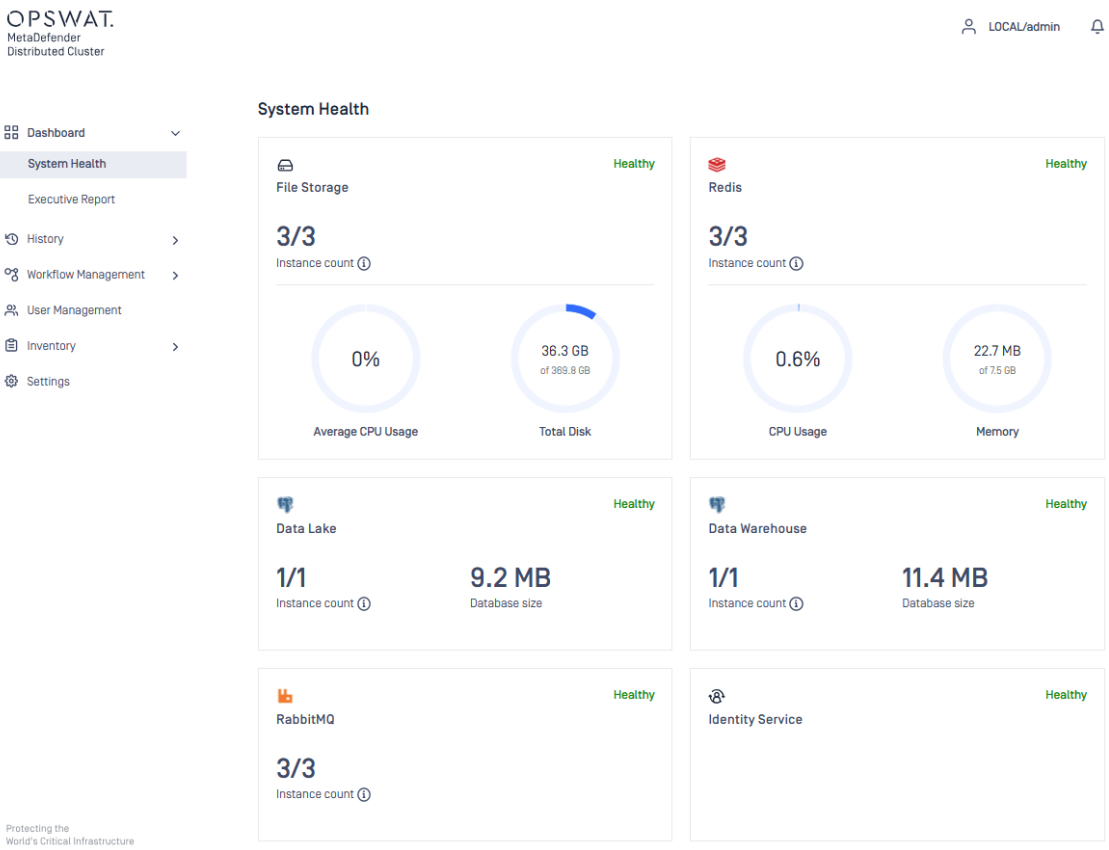
High Availability support for Redis Caching Server, RabbitMQ Message Broker and MetaDefender Cluster File Storage

Administrators can add multiple instances of Redis, RabbitMQ and MetaDefender Cluster File Storage for High Availability support.



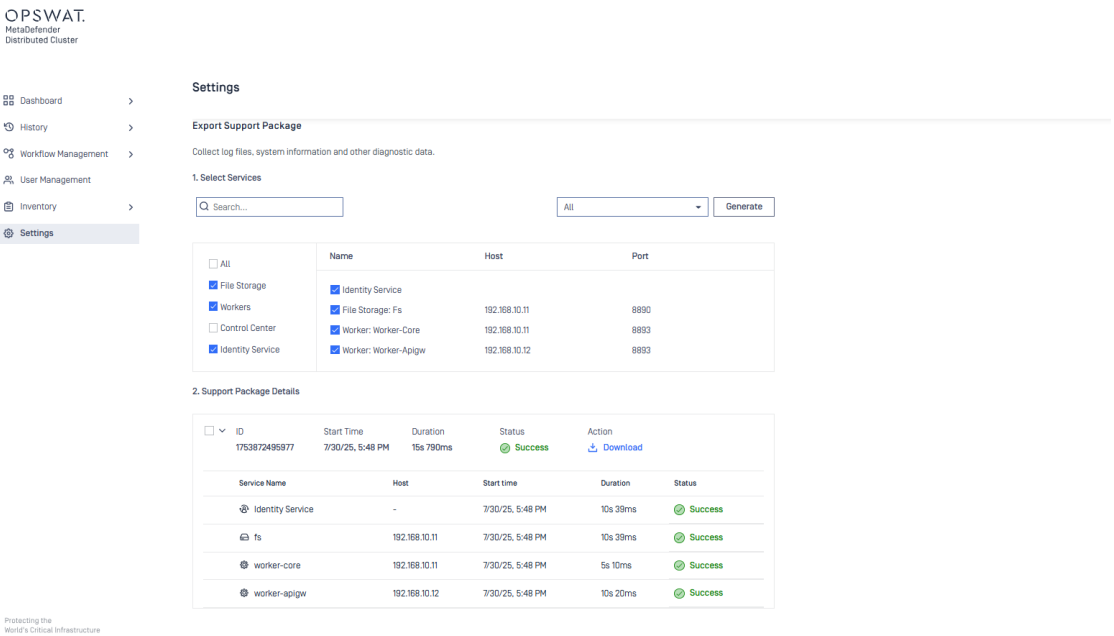
New Dashboard for High Availability support

A new dashboard is designed to show brief information about critical components of MetaDefender Cluster.



Remote Support Package Gathering

A feature to collect support packages remotely by the web console of MetaDefender Cluster Control Center.



RESTful API

- Introduce a new API endpoint in MetaDefender Cluster API Gateway to fetch a list of active MetaDefender Core

instances and their details: GET /stat/nodes.

- Introduce a new API endpoint in MetaDefender Cluster API Gateway to fetch a list of active engines and their properties GET /stat/engines.

Security Enhancements

Upgraded libraries for vulnerability fixes:

- Angular v19
- SQLite 3.47.2

Bug Fixes

- Fixed an issue where MetaDefender Cluster API Gateway always responded with HTTP code 500 when clients attempt to call the API `GET` `/hash/{md5|sha1|sha256|sha512}` with `rule` header.
- Fixed an issue that caused MetaDefender Cluster File Storage to peg at 100% CPU in certain cases.
- Fixed an issue that prevented MetaDefender Cluster components from restarting after an upgrade to the newer version on Windows Server 2025.
- Fixed an issue that prevented MetaDefender Cluster File Storage from upgrading to the newer version on Rocky.

Version v2.1.0

Release Date: 03 July 2025

Update product name

Introduce **MetaDefender Cluster** as the new product name.

RESTful API

- Support `include-inprogress` header in `GET` `/hash/{md5|sha1|sha256|sha512}` to fetch the scan status of the latest request by hash, including incomplete ones.
- Support `Content-Encoding` header in `POST` `/file`.

Further Enhancements

- Turn `storage.path` key in the ignition file for MetaDefender Cluster File Storage into an optional setting.
- Correct the returned HTTP code when the user signs in to or signs out from MetaDefender Cluster Control Center with the wrong information.
- Correct the returned HTTP code when `POST` `/file` is called with wrong API key.

Security Enhancements

Upgraded libraries for vulnerability fixes:

- PostgreSQL v14.18
- yaml-cpp v0.8.0

Bug Fixes

- Fixed an issue where MetaDefender Cluster Control Center can't show scan results of in-progress files on the web console.
- Fixed an issue that caused MetaDefender Cluster API Gateway and MetaDefender Core to be impossible to deploy after a deployment failure.
- Fixed an issue that caused MetaDefender Cluster File Storage to crash while downloading file when DEBUG log is enabled.
- Fixed an issue that caused MetaDefender Cluster File Storage impossible to do data retention with the setting.
- Fixed an issue that caused the MetaDefender Cluster Worker to fail to upgrade due to an engine crash while MetaDefender Core was hosted on the worker.
- Fixed an issue that caused the system administrator to be unable to access the Module page in the MetaDefender Cluster Control Center web console.

Use a RAM disk for MetaDefender Core temporary files

A RAM disk stores data in memory instead of on a physical disk. You can use one for the MetaDefender (MD) Core temporary directory to reduce disk I/O and improve access to temporary scan files.

Because a RAM disk uses system memory and is cleared whenever the operating system restarts, plan its size carefully. Make sure the host still has enough memory for MD Core, the operating system, and any other services.

Before you begin, choose values for the following placeholders:

Placeholder	Description	Example
`<ramdisk_size>`	Amount of memory allocated to the RAM disk	`16G`
`<ramdisk_path>`	Directory used as the MD Core temporary directory	Linux: `/mnt/mdcore-ramdisk` Windows: `R:\mdcore-temp`
`<ramdisk_drive>`	Windows drive letter assigned to the RAM disk	`R:`

Set up the RAM disk

Linux

Linux provides two in-memory file systems: **tmpfs** and **ramfs**.

Aspect	tmpfs	ramfs
Size limit	Enforces the limit set by `size=`. The default is half of the host's physical RAM.	Does not enforce a size limit and continues to grow while memory is available.
Behavior when full	New writes fail with a `No space left on device` error. The scan may fail, but the host remains available.	Continues to consume memory and may cause the operating system to terminate processes.
Disk usage in `df`	Shows the configured size and available space.	Does not provide useful capacity information.

> ***We recommend tmpfs.***

Option 1: Set up tmpfs - Recommended

1. Create the mount point:

```
Code · bash

sudo mkdir -p <ramdisk_path>
```

2. Add the following entry to `/etc/fstab`:

```
Code

tmpfs  <ramdisk_path>  tmpfs  defaults,size=<ramdisk_size>,mode=0755  0  0
```

3. Reload the systemd configuration and mount the file system:

```
Code · bash

sudo systemctl daemon-reload
sudo mount -a
```

4. Confirm that the RAM disk is mounted with the expected size:

```
Code · bash

df -h <ramdisk_path>
```

Option 2: Set up ramfs

Use this option only if you have deliberately chosen **ramfs**. The configuration does not include `<ramdisk_size>` because `ramfs` cannot enforce a size limit.

1. Create the mount point:

```
Code · bash

sudo mkdir -p <ramdisk_path>
```

2. Add the following entry to `/etc/fstab`:

```
Code

ramfs  <ramdisk_path>  ramfs  defaults,mode=0755  0  0
```

3. Reload the systemd configuration and mount the file system:

```bash
sudo systemctl daemon-reload
sudo mount -a
```

4. Confirm that the RAM disk is mounted:

```bash
findmnt <ramdisk_path>
```

Windows

On Windows, use a virtual disk driver to create the RAM disk. Because its contents are lost at every restart, configure Task Scheduler to recreate it automatically when Windows starts.

1. Install and test ImDisk

Install the ImDisk Virtual Disk Driver. Then open Command Prompt as an administrator and run the following commands:

```powershell
imdisk -a -s <ramdisk_size> -m <ramdisk_drive> -p "/fs:ntfs /q /y"
mkdir "<ramdisk_path>"
```

Confirm that the new drive and temporary directory are available before continuing.

2. Create a startup script

Save the following commands in a file such as `C:\OPSWAT\create-ramdisk.cmd`:

```
@echo off
if not exist <ramdisk_drive>\ imdisk -a -s <ramdisk_size> -m <ramdisk_drive> -p "/fs:ntfs /q /y"
if not exist "<ramdisk_path>" mkdir "<ramdisk_path>"
```

The existence checks make the script safe to run more than once.

3. Run the script at startup

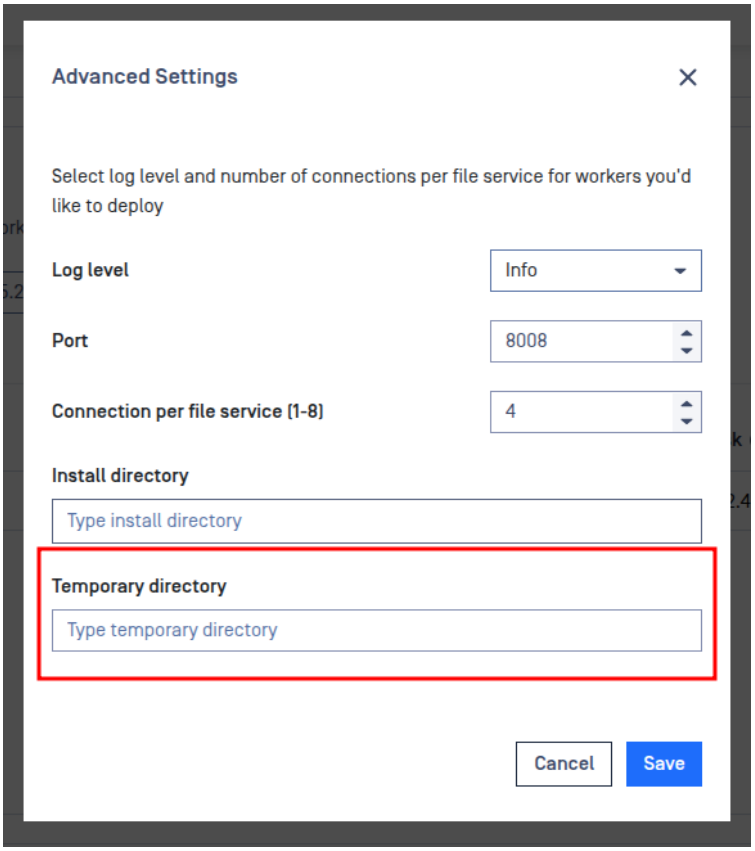
In Task Scheduler, create a new basic task with the following settings:

Setting	Value
User account	`SYSTEM`
Run with highest privileges	Enabled. Creating a virtual disk requires elevated privileges.
Run whether the user is logged on or not	Enabled. The disk must be available even when no user is signed in.
Trigger	At startup, with a delay of approximately 30 seconds to allow the ImDisk driver to initialize.
Action	Select **Start a program**, and fill in **Program/script** with `C:\OPSWAT\create-ramdisk.cmd`
Additional setting	Enable **Allow task to be run on demand** so that you can test it without restarting Windows.

Run the task once on demand and confirm that it creates both the drive and `<ramdisk_path>`.

Configure MetaDefender Core to use the RAM disk

When you deploy the MetaDefender Core instance, open **Advanced Settings** and set **Temporary directory** to `<ramdisk_path>`. For deployment instructions, see Deploy Instances.



After deployment, submit a test scan and confirm that MD Core creates temporary data on the RAM disk.

What happens after a restart

A RAM disk is empty after the operating system restarts, so the temporary subdirectory created by MD Cluster no longer exists. When the MD Cluster Worker service starts, it automatically attempts to recreate the directory for approximately 20 seconds. This delay allows time for the RAM disk to become available during startup.

The following behavior is expected:

- If the RAM disk becomes available within approximately 20 seconds, the MD Cluster Worker recreates the directory and starts the MD Core instance.
- If the RAM disk takes longer to mount, the instance remains stopped. Restart the MD Cluster Worker service after the RAM disk becomes available.
- If the RAM disk does not mount, the instance remains unhealthy. MD Cluster does not create the directory on regular storage, which prevents temporary scan data from being written to the wrong device.
- On Linux, MD Core may briefly appear unhealthy immediately after a restart and then recover automatically. This happens when MD Core starts before the temporary directory has been recreated and MD Cluster restarts it.
- On Windows, the MD Core instance normally starts after the RAM disk and its temporary directory have been recreated, so you should not see this temporary unhealthy state.

API Reference

API Gateway

API Version: v2.10.0

Developer Guide

This is the API documentation for *MetaDefender Cluster API Gateway Public API*. If you would like to evaluate or have any questions about this documentation, please contact us via our Contact Us form.

How to Interact with MetaDefender Cluster API Gateway using REST API

MetaDefender Cluster API Gateway is used to submit files for analysis, retrieve scan results, manage file processing, download processed files, and manage file batches. OPSWAT recommends using the JSON-based REST API. The available methods are documented below.

Note: MetaDefender Cluster API doesn't support chunk upload, however is recommended to stream the files to MetaDefender Cluster API Gateway as part of the upload process.

File Analysis Process

MetaDefender Cluster is a system with multiple components that work together to utilize the power of multiple MetaDefender Core instances. The system is designed to handle large volumes of files and provide high throughput for file analysis. The system can be deployed in a distributed manner, allowing for horizontal scaling and load balancing across multiple MetaDefender Core instances.

Below is a brief description of the API integration flow:

1. Upload a file for analysis to MetaDefender Cluster API Gateway (POST /file), which returns the data_id: File Analysis.

2. The following method can be used to retrieve the analysis report:

   - **Polling**: Fetch the result with previously received data_id (GET /file/{data_id} resource) until scan result belonging to data_id doesn't reach the 100 percent progress_percentage: (Fetch analysis result)

   Note: Too many data_id requests can reduce performance. It is enough to just check every few hundred milliseconds.

3. Retrieve the analysis results anytime after the analysis is completed with hash for files (md5, sha1, sha256, sha512) by calling Fetch analysis result by hash.

- The hash can be found in the scan results

4. Retrieve processed file (sanitized, redacted, watermarked, etc.) after the analysis is complete.

   Note: Based on the configured retention policy, the files might be available for retrieval at a later time.

OPSWAT provides some sample codes on GitHub to make it easier to understand how the MetaDefender REST API works.

CONTACT

NAME: API Support
EMAIL: feedback@opswat.com
URL: https://github.com/OPSWAT/metadefender-core-openapi3
Terms of service: https://onlinehelp.opswat.com/policies/

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
apikey	apiKey	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

API

1. ANALYSIS

File analysis APIs

Submit each file to MetaDefender Cluster API Gateway individually or group them in batches. Each file submission will return a data_id which will be the unique identifier used to retrieve the analysis results.

Note*:* MetaDefender API doesn't support chunk upload. You shouldn't load the file in memory, is recommended to stream the files to MetaDefender Cluster API Gateway as part of the upload process.

`1.1 POST /file`

Analyze File (Asynchronous mode)

Scanning a file using a specified workflow. Scan is done asynchronously and each scan request is tracked by data id of which result can be retrieved by API Fetch Scan Result.

Note: Chunked transfer encoding (applying header Transfer-Encoding: Chunked) is **not supported** on /file API. API Gateway **only accepts** file uploads with a known content length and a content type of application/octet-stream

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
filename	string		The name of the submitted file
user_agent	string		user_agent header used to identify (and limit) access to a particular rule. For rule selection, `rule` header should be used.
rule	string		Select rule for the analysis, if no header given the default rule will be selected (URL encoded UTF-8 string of rule name)
batch	string		Batch id to scan with, coming from `Initiate Batch` (If it is not given, it will be a single file scan.)
archivepwd	string		Password for archive (URL encoded UTF-8 string) Multiple passwords is also supported, format: archivepwdX * X: Could be empty * When having value, X must be a number >= 1 For example: * archivepwd1: "fox" * archivepwd2: "cow" * archivepwd3: "bear"
content-encoding	string		Content encoding of the file. This header is used to specify the encoding of the file content. The value should be a valid content encoding type, such as "base64", "gzip". This header is optional and can be omitted if the encoding is not applicable.
metadata	json		Could be utilized for: * Additional parameter for pre-defined post actions and external scanners (as a part of STDIN input). * Customized macro variable for watermarking text (Proactive DLP engine feature). * Additional context / verbose information for each file submission (appended into JSON response scan result). It is strongly recommended to apply URL encoding before sending `metadata` to Metadefender Core to prevent unexpected issues related to encoding errors or unsafe characters.

NAME	TYPE	EXAMPLE	DESCRIPTION
`engines-metadata`	`json`		Since MetaDefender Core 5.0.0, preferred context / verbose information can be sent to the engines. Please see the below pages for the details: * [File Type engine](https://docs.opswat.com/mdcore/utilities-engines/supported-engines-metadata) (supported since Core 5.2.1) * [Archive engine](https://docs.opswat.com/mdcore/utilities-engines/supported-engines-metadata-header) (supported since Core 5.4.1) * [Deep CDR](https://docs.opswat.com/mdcore/deep-cdr/supported-engines-metadata-json) * [Proactive DLP](https://docs.opswat.com/mdcore/proactive-dlp/supported-engines-metadata-json)
`callbackurl`	`uri`		Client's URL where MetaDefender Cluster Callback Service will notify scan result back to whenever scan is finished (webhook model). * Format: <protocol://><ip | domain>:<port></path> * Example: http://10.0.1.100:8081/listenback * Supported protocol: HTTP / HTTPS * Supported host types: domain name, IPv4 (IPv6 not supported) * Method: POST > _**Note**_: The Callback URL is only supported when MetaDefender Cluster Callback Service is deployed, and MetaDefender Core version must be 5.16.1 or higher.

NAME	TYPE	EXAMPLE	DESCRIPTION
downloadfrom	uri		Format: \<protocol://\>\<ip | domain\>:\<port\>\</path\> Scan a file from a download link, instead of uploading the file in the request body. MetaDefender Cluster Download Service downloads the file from the link, then sends it for scanning. > _**Note**_: To use this header, you need: > * MetaDefender Cluster Download Service deployed. > * MetaDefender Cluster version 2.10.0 or later. > * MetaDefender Core version 5.23.0 or later. > > If no Download Service instance is available, the request fails with HTTP 503. How it works: * Do not send a request body. `Content-Length` must be `0` or not set. If you send a body, the request fails with HTTP 400. * The response returns a `data_id` right away. The download continues in the background. * To see the download progress and result, check `download_info` in [Fetch Analysis Result](/docs/mdcore/metadefender-distributed-cluster/ref#fileanalysisget). * If you do not set the `filename` header, the file name is the last part of the link path. Supported: * Protocols: HTTP and HTTPS. * Single file scan, with or without webhook. * Batch scan: add the `downloadfrom` header to each file request in the batch. The batch request itself needs nothing extra. * If you cancel the batch, all files in the batch that are still downloading are also cancelled. * Settings in the workflow ("General" tab): * Max sizes > URL file download * Timeouts > File download * Settings in Settings > Scan From Link: * Max download queue: the number of downloads that run at once. Other links wait their turn. * Idle timeout: a download fails if it receives no data for this long. * Enforce scan from link validation: allow or block links that match the given patterns. * Size check before download: MetaDefender Cluster sends an HTTP `HEAD` request to the link. If the file is larger than the max URL file download size or the max file scan size, the download is refused. To skip this check, use the `skip-head-request` header.

NAME	TYPE	EXAMPLE	DESCRIPTION
`redirect-support`	`boolean`		Use only with the `downloadfrom` header. Follow redirects of the download link. * Default: `false`. If the link redirects, the request fails with HTTP 400 (`Redirect link not supported.`). * `true`: Follow up to 5 redirects.
`skip-head-request`	`boolean`		Use only with the `downloadfrom` header. Skip the HTTP `HEAD` request that checks the file size before the download. Use it when the download server does not support `HEAD` requests. * Default: `false`. Send the `HEAD` request first, then download. * `true`: Download the file directly. The size limits are still checked during the download.
`global-timeout`	`integer`		This custom global timeout (in seconds) will override the global timeout predefined in corresponding workflow rule.

NAME	TYPE	EXAMPLE	DESCRIPTION
client-identity	json		The `client-identity` header identifies the clients involved in processing a file, providing greater visibility into its processing stages. The header value is a JSON object containing a clients array. Each `client` object supports the following fields: * `deployment_id`: *REQUIRED*. Unique identifier of the deployment. * `user_name`: *REQUIRED*. Name of the user or system that initiated the scan. * `product_type`: *REQUIRED*. Type of product or system performing the scan. * `version`: *OPTIONAL*. Version of the product or system. * `host`: *OPTIONAL*. Hostname or IP address of the client machine. * `timestamp`: *REQUIRED*. Time at which the scan was initiated, expressed in milliseconds since the Unix epoch. Example: ```json { "clients": [{ "deployment_id": "abcdef1234567890", "user_name": "Gemini", "product_type": "MetaDefender Kiosk", "version": "1.0.0", "host": "client.example.com", "timestamp": 1744949754890 }] } ``` URL-encode the JSON value before sending the header to MetaDefender Cluster to avoid issues caused by unsafe characters or incorrect encoding. Omit the header when client identity tracking is not required.

RESPONSE

STATUS CODE - 200: Successful file submission

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
data_id*	string	Unique submission identifier. Use this value to reference the submission.

EXAMPLE:


```
{
```

```
 "data_id": "61dffeaa728844adbf49eb090e4ece0e"
}
```

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or
out of range, etc). The request is rejected before the file is processed, so no `data_id` is created.

When the `client-identity` header is present but its value is invalid, `err` is one of:

`err`	Cause
`Header 'client-identity' is empty`	The header was sent without a value.
`Header 'client-identity' must be less than or equal 4096 characters`	The header value is longer than 4096 characters.
`Header 'client-identity' contains non-printable character`	The value contains an ASCII control character, either directly or through an escape.
`Header 'client-identity' is not in JSON format`	The value could not be URL-decoded, is not valid JSON, or is not a JSON object.
`Header 'client-identity' does not contain a 'clients' key or empty 'clients'`	`clients` is missing, is not an array, or is an empty array.
`Header 'client-identity' contains an invalid data`	An entry of `clients` is not a JSON object.
`Header 'client-identity' does not contain required key '<key>'`	An entry is missing `deployment_id`, `user_name`, `product_type` or `timestamp`.
`Header 'client-identity' contains a non-string '<key>'`	`deployment_id`, `user_name` or `product_type` is not a string.
`Header 'client-identity' contains an empty '<key>'`	`deployment_id`, `user_name` or `product_type` is blank.
`Header 'client-identity' contains an invalid 'deployment_id'`	`deployment_id` holds anything other than letters and digits.
`Header 'client-identity' contains a non-integer 'timestamp'`	`timestamp` is not an integer.
`Header 'client-identity' contains a negative 'timestamp'`	`timestamp` is zero or negative.

Every entry of `clients` is checked, not only the first one, and the first rule that fails is the one reported.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Header 'client-identity' contains an invalid 'deployment_id'"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 411: Content-Length header is missing from the request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

EXAMPLE:

```
{
 "err": "Missing Content-Length header."
}
```

STATUS CODE - 422: Body input is empty.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

EXAMPLE:

```
{
 "err": "File is empty."
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

STATUS CODE - 503: Service is unavailable, `err` is one of:
* `Server is too busy. Try again later.`
* `Failed to request because no MetaDefender Cluster Download Service instance is available` - the `downloadfrom` header was given but no Download Service instance is healthy.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

EXAMPLE 1:

```
"busy"
```

EXAMPLE 2:

```
"noDownloadService"
```

1.2 GET /file/{data_id}

Fetch Analysis Result

Retrieve scan results.

Scan is done asynchronously and each scan request is tracked by a data ID.

Initiating file scans and retrieving the results need to be done using two separate API calls. This request needs to be made multiple times until the scan is complete. Scan completion can be traced using scan_results.progress_percentage value from the response.

Note: The REST API also supports pagination for archive file result. A completed response description with archive detection:

- extracted_files: information about extracted files
  - files_extracted_count: the number of extracted files
  - files_in_archive: array of files in archive
    - detected_by: number of engines reported threat
    - scanned_with: number of engines used for scanning the file

- first_index: it tells that from which file (index of the file, 0 is the first) the result JSON contains information about extracted files. (default=0, min=0)
- page_size: it tells how many files the result JSON contains information about (default=50, min=0, max=2000). So by default, the result JSON contains information about the first 50 extracted files.
- worst_data_id: data id of the file that has the worst result in the archive
- total_reused_files: Indicates how many extracted files reused results from a previous analysis instead of being processed again. This field is available starting with MetaDefender Cluster 2.10.0. A value of 0 means no extracted files were reused. If the entire archive is reused, original_data_id is set and files_in_archive is empty, while the archive totals are still included in the response.

- scan_results
  - last_file_scanned (stored only in memory, not in database): If available, the name of the most recent processed file

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string		Unique submission identifier. Use this value to reference the submission.

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
first	integer >=0		The `first` item order in the list child files of archive file
size	integer between 0 and 2000		The number of items to be fetched next, counting from the item order indicated in `first` header. The default value is 50, and the maximum value is 2000.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
user_agent	string		user_agent header used to identify (and limit) access to a particular rule. For rule selection, `rule` header should be used.

RESPONSE

STATUS CODE - 200: Entire analysis report generated by MetaDefender Core

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
data_id	string	data identifier of the requested file
deflected_engines	array	
deflection_info	object	
final_verdict	object	
verdict	enum	ALLOWED: clean, malicious, unknown, Cancelled Deflection engine verdict. Values: - clean: the file is considered clean. - malicious: the file is considered malicious. - unknown: the engine could not confidently classify the file. - Cancelled: processing was cancelled before a verdict was produced.
confidence	integer	between 0 and 100 Confidence of the verdict, as a percentage (0-100).
detail	object	
data_id	string	data identifier of the analyzed file.
file_type	string	Detected file type of the analyzed file.
result_template_hash	string	Hash of the result template used internally for Deflection rendering and coordination.
err_code	integer	Returned only when Deflection failed to produce a verdict (`final_verdict` is then absent). Common values: * 1: Deflection failed. * 5: Deflection timed out. * 6: The file type is not supported by Deflection. * 9: The global processing timeout was exceeded.
err_details	string	Returned only when Deflection failed. Human-readable reason of the failure.
err_category	string	Returned only when Deflection failed and the engine reported a failure category.
dlp_info	object	

NAME	TYPE	DESCRIPTION
certainty	`enum`	**ALLOWED:** `Very Low, Low, Medium, High, Very High` Describes how certain the hit is, possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
errors	`object`	
filename	`string`	Output processed file name (pre-configured on engine settings under Core's worflow rule)
hits	`object`	
ccn	`object`	
display_name	`string`	Credit Card Number, Social Security Number, or in case of RegEx, the name of the rule that has been given by the user
hits	`array`	
after	`string`	The context after the matched data.
before	`string`	The context before the matched data.
certainty	`enum`	**ALLOWED:** `Very Low, Low, Medium, High, Very High` The text version of "certainty_score", possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
certainty_score	`integer`	Is defined by the relevance of the given hit in its context. It is calculated based on multiple factors such as the number of digits, possible values: [0-100]
hit	`string`	The matched data.
location	`string`	The location of the hit that is found in a file.
severity	`enum`	**ALLOWED:** `0, 1` (NOTE: this field is deprecated): can be 0 (detected) or 1 (suspicious).
tryRedact	`boolean`	If file was redacted or not.
metadata_removal	`object`	
result	`enum`	**ALLOWED:** `removed, not removed, failed to remove` Result of the metadata removal process, possible values: * `removed` * `not removed` * `failed to remove`
redact	`object`	
result	`enum`	**ALLOWED:** `redacted, not redacted, failed to redact` Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`

NAME	TYPE	DESCRIPTION
severity	`enum`	**ALLOWED:** `0, 1` (NOTE: this field is deprecated): represents the severity of the data loss, possible values: * `0` - Certainly is data loss * `1` - Might be data loss
verdict	`enum`	**ALLOWED:** `0, 1, 2, 3, 4` The overall result for the scanned file. Possible values: * `0` - Clean * `1` - Found matched data * `2` - Suspicious * `3` - Failed * `4` - Not scanned
watermark	`object`	
result	`enum`	**ALLOWED:** `added, not added, failed to add` Result of the watermarking process, possible values: * `added` * `not added` * `failed to add`
download_info	`object`	
error_detail	`string`	Revealed detailed reason why the download failed. Only present when the download did not succeed.
progress	`number`	Only applicable when "status" is `Downloading`, indicates download finished percentage, in a range of [0, 99]. * Once hitting 100, the status will be changed to `Download Success`. * or other problematic status (`Download Cancelled`, `Download Failed`, `Download Timeout`) if the download stopped unexpectedly. * Stays at 0 when the download server does not declare the file size (no `Content-Length`); check `downloaded` instead.
downloaded	`integer`	Only applicable when "status" is `Downloading`, indicates the number of bytes downloaded so far.

NAME	TYPE	DESCRIPTION
status	`enum`	**ALLOWED:** `Downloading, Download Success, Download Failed, Download Timeout, Download Cancelled` Indicates download status, which could be either - `Downloading` - Check `progress` and `downloaded` key values for actual download progress ```json "download_info": { "downloaded": 268435456, "progress": 5, "status": "Downloading", "url": "http://192.168.200.97:8080/5gb.zip" } ``` - `Download Success` ```json "download_info": { "status": "Download Success", "url": "https://secure.eicar.org/eicar.com" } ``` - `Download Failed` - Check `error_detail` key value for an error explanation - The scan result is `Download Failed` ```json "download_info": { "error_detail": "Connection error", "status": "Download Failed", "url": "http://192.168.200.97:8080/2gb.zip" } ``` - `Download Timeout` - Expecting to occur when the download takes longer than the file download timeout configured in the workflow (General tab > Timeouts > File download) - The scan result is `Download Timeout` ```json "download_info": { "status": "Download Timeout", "url": "http://192.168.200.97:8080/2gb.zip" } ``` - `Download Cancelled` - Expecting to occur when user explicitly cancelled that file scan request, or batch request that the scan belongs to ```json "download_info": { "status": "Download Cancelled", "url": "http://192.168.200.97:8080/5gb.zip" } ```
url	`string`	Original download link which was specified in HTTP(S) request's `downloadfrom` header
extraction_info	`object`	
decrypted_status	`enum`	**ALLOWED:** `Success, Failed` Indicate that decryption phase is successful or not.
err_category	`string`	Error category
err_code	`integer`	Error code

NAME	TYPE	DESCRIPTION
err_details	string	Error message
is_encrypted_file	boolean	Indicate if file is password-protected or not.
total_reused_extraction_errors	integer	Indicates how many extraction errors were reused from a previous analysis instead of being detected during the current analysis. This can include errors from the archive itself or from nested archives. A value of 0 means no extraction errors were reused. This field is available starting with MetaDefender Cluster 2.10.0.
file_info	object	
display_name	string	The filename reported via `filename` header.
file_size	integer	Total file size in bytes.
file_type	string	The filetype using mimetype.
file_type_description	string	The filetype in human readable format.
md5	string	File's MD5 hash.
sha1	string	File's SHA1 hash.
sha256	string	File's SHA256 Hash.
sha512	string	File's SHA512 Hash.
signer_infos	array	
digest_algorithm	string	Digest algorithm.
digest_encryption_algorithm	string	Encryption algorithm.
issuer	string	Entity that develops and registers the certificate.
serial_number	string	Serial number of the certificate.
vendor_name	string	Entity that is issued a certificate and utilize it for creating a digital signature.
version	string	Version of X.509 that is used in the certificate. This version field is zero-based. * 0: v1 * 1: v2 * 2: v3
type_category	array	
receive_data_timestamp	string	The timestamp when upload progress started (first byte received) (in milliseconds)
upload_time	integer	Total time elapsed for upload process (in milliseconds).
upload_timestamp	string	The timestamp when upload progress finished (all bytes received) (in milliseconds)
filetype_info	object	
file_info*	object	
description*	string	File type description
detected_by	string	Analyzer that detected the file type
encrypted*	boolean	File is password-protected or not

NAME	TYPE	DESCRIPTION
extensions*	string	File type extension
groupID*	string	File type category
groupIDs*	array	
group_description	string	File type category description
likely_type_ids	array	
score*	integer	Likelihood score of the file type
typeID*	string	File type ID
type*	string	MIME type
typeID*	string	File type ID
type_ids*	array	
final_verdict	object	
verdict*	enum	**ALLOWED:** `allowed, blocked` Final verdict of the file type analysis.
verdict_explanation*	string	Explanation of the final verdict.
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.
other_detections	array	Other file type detections.
result_template_hash	string	SHA256 Hash of user-interface template. For web console only.
spoofing_info	object	
detection_result	string	Result of the spoofing detection.
result_explanation	string	Explanation of the spoofing detection result.
result_overview	string	Overview of the spoofing detection result.
opswatfilescan_info	object	
original_data_id	string	The data ID of the previous analysis whose result was reused. This field is present only when the result was returned from an earlier analysis through the workflow rule's Reuse processing result option (`process_info.reuse_processing_result`) instead of being processed again. For archive files, archive result reuse must also be enabled with `process_info.reuse_processing_result.include_archive`. Available starting with MetaDefender Cluster 2.10.0.
process_info	object	
blocked_reason	string	Provides the reason why the file is blocked (if so).
blocked_reasons	array	
file_type_skipped_scan	boolean	Indicates if the input file's detected type was configured to skip scanning.
hash_time	integer	Total time elapsed for computing hashes (in milliseconds).
outdated_data	array	
processing_time	integer	Total time elapsed during processing file (in milliseconds).
processing_time_details	object	

NAME	TYPE	DESCRIPTION
av_scan_time	integer	AV engines' processing time.
cdr_time	integer	Deep CDR engine's sanitization time.
dlp_time	integer	Proactive DLP engine's processing time.
extraction_time	integer	Archive extraction engine's processing time.
filetype_time	integer	FileType engine's processing time.
opswatfilescan_time	integer	OPSWAT Filescan engine's processing time.
others_time	integer	Total time elapsed for following processing tasks in the product (in milliseconds): * Decryption time (if receiving an encrypted file) * External scanner (if configured) * Post action (if configured) * Other internal processing time among components in the product
parse_dgsg_time	integer	Digital signature analyzing time.
vul_time	integer	Vulnerability engine's lookup time.
yara_time	integer	YARA engine's processing time.
filetype_wait_time	integer	FileType engine's wait time.
profile	string	The used rule name.
progress_percentage	integer	Percentage of processing completed (from 1-100).
queue_time	integer	Total time elapsed for file processing task was waiting in MetaDefender Core's queue until being picked up (queue_time = start_time - upload_timestamp) (in milliseconds).
result	string	The final result of processing the file (Allowed / Blocked / Processing).
user_agent	string	Identifier for the REST Client that calls the API.
username	string	User identifier who submitted scan request earlier.
verdicts	array	
post_processing	object	
actions_failed	string	Empty string if no action failed or list of failed actions, separated by "|".
actions_ran	string	List of successful actions, separated by "|". Empty string if otherwise.
converted_destination	string	Contains the name of the sanitized file.
converted_to	string	Contains target type name of sanitization.
copy_move_destination	string	Contains target type name of sanitization.
sanitization_details	object	
cdr_wait_time	integer	The time in milliseconds that the CDR process took to complete.
description	string	Action was successful or not.
details	array	
action*	enum	**ALLOWED:** `sanitized, removed` The type of action that was performed

NAME	TYPE	DESCRIPTION
count	integer	The number of objects that were sanitized/removed.
details	object	
action	enum	ALLOWED: sanitized, removed The type of action that was performed
count	integer	The number of objects that were sanitized/removed.
object_details	array	
object_name	string	The object type that was sanitized/removed.
description	string	Action was successful or not.
file_name	string	If an embedded file was sanitized.
object_details	array	
object_name*	string	The object type that was sanitized/removed.
failure_category	string	Deep CDR errors are classified into different categories. For more details, please find [Troubleshooting sanitization failures](https://docs.opswat.com/mdcore/deep-cdr/troubleshooting-sanitization-failures)
result	enum	ALLOWED: Sanitized, Sanitized failed, Sanitized skipped The result of the CDR process. - **Sanitized**: the file was successfully sanitized. - **Sanitized failed**: the file could not be sanitized due to an error during the process. - **Sanitized skipped**: the file was skipped from sanitization. Common reasons include the file being digitally signed or other policy-based exclusions.
result_template_hash	string	The hash value of the result template, which is used for displaying results on the Core UI and for internal communication between MetaDefender Core and the Deep CDR engine. This value is intended for system use only and is not required for external integration.
sanitized_file_info	object	
file_size	integer	Size of sanitized file in bytes.
sha256	string	SHA256 hash of sanitized file.
verdict	enum	ALLOWED: blocked, allowed The verdict of the CDR process. - **blocked**: the file is recommended for blocking by Deep CDR. - **allowed**: the file is recommended for allowing by Deep CDR as it found no reason to recommend blocking it.
verdict_explanations	array	
client_identity	object	
clients	array	
deployment_id*	string	Unique identifier for the deployment.
user_name*	string	Name of the user or system that initiated the scan.
product_type*	string	Type of product or system that is performing the scan.

NAME	TYPE	DESCRIPTION
version	string	Version of the product or system.
host	string	Hostname or IP address of the client machine.
timestamp*	integer	Timestamp of when the scan was initiated, in milliseconds since epoch.
scan_results	object	
data_id	string	Data ID of the requested file
progress_percentage	integer	Track analysis progress until reaches 100.
scan_all_result_a	enum	**ALLOWED:** `No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Scan Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, In Progress, Skip Processing Fast Symlink` The overall scan result as string
scan_all_result_i	enum	**ALLOWED:** `0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 67, 68, 69, 70, 71, 255, 1014` The overall scan result as index in the Processing Results table.
start_time	string	Timestamp when the scanning process starts.
total_avs	integer	Total number of scanning engines used as part of this analysis.
total_time	integer	Total time elapsed during scan (in milliseconds).
scan_details	object	
ClamAV	object	
def_time	string	The database definition time for this engine
eng_id	string	The unique identification string for the engine
location	string	Where this engine is deployed (local/remote).
scan_result_i	integer	Scan result as index in the Processing Results table
scan_time	integer	The time elapsed during scan with this engine (in milliseconds).
threat_found	string	The threat name, IF scan result is Infected or Suspicious. Otherwise empty string or error message from the engine.

NAME	TYPE	DESCRIPTION
wait_time	integer	Time elapsed between sending file to Core and receiving the result from the engine (in milliseconds).
vulnerability_info	object	
result	object	
code	integer	The result code for vulnerability check, 0 means a successful check
hash	string	The file's SHA1 hash value
method	enum	ALLOWED: `50700` The method used by OESIS Framework, it should be 50700 every time.
timestamp	string	Timestamp of the request issued
timing	integer	The vulnerability check's duration in milliseconds
detected_product	object	
has_kb	boolean	Indicates whether any KBs or MSBs exist for this hash
has_vulnerability	boolean	Indicates whether any vulnerabilities have been associated with the particular product
is_current	boolean	True if this product's patch level is current, defaults to true
product	object	
id	integer	The OPSWAT product id
name	string	The product name
remediation_link	string	A link where product updates or patches can be obtained
severity	enum	ALLOWED: `LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN` String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`
sig_name	string	Product signature descriptor
signature	integer	OPSWAT signature id
vendor	object	
id	integer	The OPSWAT vendor id
name	string	The vendor name
version	string	The installed product version
version_data	object	
count_behind	integer	The number of patches behind of the installed product
feed_id	integer	The remote feed ID used to determine patch level
version	string	The current version of the product in the remote feed
vulnerabilites	array	

NAME	TYPE	DESCRIPTION
description	string	A text description of the specific vulnerability
details	object	
cpe	string	A CPE product reference
cve	string	A CVE identification string
cvss	object	
access-complexity	string	A CVSS access-complexity descriptor
access-vector	string	A CVSS access-vector descriptor
authentication	string	A CVSS authentication descriptor
availability-impact	string	A CVSS availability impact descriptor
confidentiality-impact	string	A CVSS confidentiality impact descriptor
generated-on-epoch	string	An epoch timestamp indicating CVSS generation time
integrity-impact	string	A CVSS integrity impact descriptor
score	string	A CVSS 10-point severity score
source	string	A CVSS source descriptor
cwe	string	A CWE group identification string
last_modified_epoch	string	An epoch timestamp indicating source last update time
published-epoch	string	An epoch timestamp indicating source publishing time
references	array	
severity	enum	**ALLOWED:** `LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN` String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`
severity_index	integer	A 5 point scale numerical description of Severity level with 5 being greatest and 0 being unknown
static_id	integer	An OPSWAT identifier for the vulnerability
verdict	integer	The vulnerability check's duration in milliseconds * `0` - No Vulnerability Found * `1` - Vulnerability Found * `3` - Failed * `16` - Processing Timed Out
yara	object	
hits	object	

NAME	TYPE	DESCRIPTION
verdict	enum	**ALLOWED:** `0, 1, 2, 3, 4` The overall result for the analyzed file. Value will be one of the following: | index | status | |---|---| | 0 | Clean | | 1 | Found matched data | | 2 | Suspicious | | 3 | Failed | | 4 | Not scanned |

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
```

```
        }
```

1.3 GET /hash/{md5|sha1|sha256|sha512}

Fetch Analysis Result By Hash

Retrieve analysis result by hash

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*md5|sha1|sha256|sha512	string		Hash value to search. This can be md5, sha1, sha256, sha512

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
first	integer >=0		The `first` item order in the list child files of archive file
size	integer between 0 and 2000		The number of items to be fetched next, counting from the item order indicated in `first` header. The default value is 50, and the maximum value is 2000.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
rule	string		Select rule for the analysis, if no header given the default rule will be selected (URL encoded UTF-8 string of rule name)
selfonly	boolean		Useful to archive hash lookup. Allow specifying to only perform hash lookup against the original archive file self only, and skip searching all child files result within the original archive. Default value is false.

NAME	TYPE	EXAMPLE	DESCRIPTION
timerange	integer		Scoping down the recent number of hours that hash lookup task should start from till now, instead of searching the entire scan history in MetaDefender Cluster database. Default value is 0. That means no time scope.
include-inprogress	boolean		False (default): API will return "Not Found" if the verdict is in progress. True: If the queried hash has a completed processing result before, API will return the completed processing result. If this hash doesn't have any completed processing result, API will return this In-progress result.

RESPONSE

STATUS CODE - 200: Get information of file

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
data_id	string	data identifier of the requested file
deflected_engines	array	
deflection_info	object	
final_verdict	object	
verdict	enum	**ALLOWED:** clean, malicious, unknown, Cancelled Deflection engine verdict. Values: - clean: the file is considered clean. - malicious: the file is considered malicious. - unknown: the engine could not confidently classify the file. - Cancelled: processing was cancelled before a verdict was produced.
confidence	integer	between 0 and 100 Confidence of the verdict, as a percentage (0-100).
detail	object	
data_id	string	data identifier of the analyzed file.
file_type	string	Detected file type of the analyzed file.
result_template_hash	string	Hash of the result template used internally for Deflection rendering and coordination.

NAME	TYPE	DESCRIPTION
err_code	integer	Returned only when Deflection failed to produce a verdict (`final_verdict` is then absent). Common values: * 1: Deflection failed. * 5: Deflection timed out. * 6: The file type is not supported by Deflection. * 9: The global processing timeout was exceeded.
err_details	string	Returned only when Deflection failed. Human-readable reason of the failure.
err_category	string	Returned only when Deflection failed and the engine reported a failure category.
dlp_info	object	
certainty	enum	**ALLOWED:** `Very Low, Low, Medium, High, Very High` Describes how certain the hit is, possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
errors	object	
filename	string	Output processed file name (pre-configured on engine settings under Core's worflow rule)
hits	object	
ccn	object	
display_name	string	Credit Card Number, Social Security Number, or in case of RegEx, the name of the rule that has been given by the user
hits	array	
after	string	The context after the matched data.
before	string	The context before the matched data.
certainty	enum	**ALLOWED:** `Very Low, Low, Medium, High, Very High` The text version of "certainty_score", possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
certainty_score	integer	Is defined by the relevance of the given hit in its context. It is calculated based on multiple factors such as the number of digits, possible values: [0-100]
hit	string	The matched data.
location	string	The location of the hit that is found in a file.
severity	enum	**ALLOWED:** `0, 1` (NOTE: this field is deprecated): can be 0 (detected) or 1 (suspicious).
tryRedact	boolean	If file was redacted or not.
metadata_removal	object	

NAME	TYPE	DESCRIPTION
result	`enum`	**ALLOWED:** `removed, not removed, failed to remove` Result of the metadata removal process, possible values: * `removed` * `not removed` * `failed to remove`
redact	`object`	
result	`enum`	**ALLOWED:** `redacted, not redacted, failed to redact` Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`
severity	`enum`	**ALLOWED:** `0, 1` (NOTE: this field is deprecated): represents the severity of the data loss, possible values: * `0` - Certainly is data loss * `1` - Might be data loss
verdict	`enum`	**ALLOWED:** `0, 1, 2, 3, 4` The overall result for the scanned file. Possible values: * `0` - Clean * `1` - Found matched data * `2` - Suspicious * `3` - Failed * `4` - Not scanned
watermark	`object`	
result	`enum`	**ALLOWED:** `added, not added, failed to add` Result of the watermarking process, possible values: * `added` * `not added` * `failed to add`
download_info	`object`	
error_detail	`string`	Revealed detailed reason why the download failed. Only present when the download did not succeed.
progress	`number`	Only applicable when "status" is `Downloading`, indicates download finished percentage, in a range of [0, 99]. * Once hitting 100, the status will be changed to `Download Success`. * or other problematic status (`Download Cancelled`, `Download Failed`, `Download Timeout`) if the download stopped unexpectedly. * Stays at 0 when the download server does not declare the file size (no `Content-Length`); check `downloaded` instead.
downloaded	`integer`	Only applicable when "status" is `Downloading`, indicates the number of bytes downloaded so far.

NAME	TYPE	DESCRIPTION
status	`enum`	**ALLOWED:** `Downloading, Download Success, Download Failed, Download Timeout, Download Cancelled` Indicates download status, which could be either - `Downloading` - Check `progress` and `downloaded` key values for actual download progress ```json "download_info": { "downloaded": 268435456, "progress": 5, "status": "Downloading", "url": "http://192.168.200.97:8080/5gb.zip" } ``` - `Download Success` ```json "download_info": { "status": "Download Success", "url": "https://secure.eicar.org/eicar.com" } ``` - `Download Failed` - Check `error_detail` key value for an error explanation - The scan result is `Download Failed` ```json "download_info": { "error_detail": "Connection error", "status": "Download Failed", "url": "http://192.168.200.97:8080/2gb.zip" } ``` - `Download Timeout` - Expecting to occur when the download takes longer than the file download timeout configured in the workflow (General tab > Timeouts > File download) - The scan result is `Download Timeout` ```json "download_info": { "status": "Download Timeout", "url": "http://192.168.200.97:8080/2gb.zip" } ``` - `Download Cancelled` - Expecting to occur when user explicitly cancelled that file scan request, or batch request that the scan belongs to ```json "download_info": { "status": "Download Cancelled", "url": "http://192.168.200.97:8080/5gb.zip" } ```
url	`string`	Original download link which was specified in HTTP(S) request's `downloadfrom` header
extraction_info	`object`	
decrypted_status	`enum`	**ALLOWED:** `Success, Failed` Indicate that decryption phase is successful or not.
err_category	`string`	Error category
err_code	`integer`	Error code

NAME	TYPE	DESCRIPTION
err_details	string	Error message
is_encrypted_file	boolean	Indicate if file is password-protected or not.
total_reused_extraction_errors	integer	Indicates how many extraction errors were reused from a previous analysis instead of being detected during the current analysis. This can include errors from the archive itself or from nested archives. A value of 0 means no extraction errors were reused. This field is available starting with MetaDefender Cluster 2.10.0.
file_info	object	
display_name	string	The filename reported via `filename` header.
file_size	integer	Total file size in bytes.
file_type	string	The filetype using mimetype.
file_type_description	string	The filetype in human readable format.
md5	string	File's MD5 hash.
sha1	string	File's SHA1 hash.
sha256	string	File's SHA256 Hash.
sha512	string	File's SHA512 Hash.
signer_infos	array	
digest_algorithm	string	Digest algorithm.
digest_encryption_algorithm	string	Encryption algorithm.
issuer	string	Entity that develops and registers the certificate.
serial_number	string	Serial number of the certificate.
vendor_name	string	Entity that is issued a certificate and utilize it for creating a digital signature.
version	string	Version of X.509 that is used in the certificate. This version field is zero-based. * 0: v1 * 1: v2 * 2: v3
type_category	array	
receive_data_timestamp	string	The timestamp when upload progress started (first byte received) (in milliseconds)
upload_time	integer	Total time elapsed for upload process (in milliseconds).
upload_timestamp	string	The timestamp when upload progress finished (all bytes received) (in milliseconds)
filetype_info	object	
file_info*	object	
description*	string	File type description
detected_by	string	Analyzer that detected the file type
encrypted*	boolean	File is password-protected or not

NAME	TYPE	DESCRIPTION
extensions*	string	File type extension
groupID*	string	File type category
groupIDs*	array	
group_description	string	File type category description
likely_type_ids	array	
score*	integer	Likelihood score of the file type
typeID*	string	File type ID
type*	string	MIME type
typeID*	string	File type ID
type_ids*	array	
final_verdict	object	
verdict*	enum	**ALLOWED:** `allowed, blocked` Final verdict of the file type analysis.
verdict_explanation*	string	Explanation of the final verdict.
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.
other_detections	array	Other file type detections.
result_template_hash	string	SHA256 Hash of user-interface template. For web console only.
spoofing_info	object	
detection_result	string	Result of the spoofing detection.
result_explanation	string	Explanation of the spoofing detection result.
result_overview	string	Overview of the spoofing detection result.
opswatfilescan_info	object	
original_data_id	string	The data ID of the previous analysis whose result was reused. This field is present only when the result was returned from an earlier analysis through the workflow rule's Reuse processing result option (`process_info.reuse_processing_result`) instead of being processed again. For archive files, archive result reuse must also be enabled with `process_info.reuse_processing_result.include_archive`. Available starting with MetaDefender Cluster 2.10.0.
process_info	object	
blocked_reason	string	Provides the reason why the file is blocked (if so).
blocked_reasons	array	
file_type_skipped_scan	boolean	Indicates if the input file's detected type was configured to skip scanning.
hash_time	integer	Total time elapsed for computing hashes (in milliseconds).
outdated_data	array	
processing_time	integer	Total time elapsed during processing file (in milliseconds).
processing_time_details	object	

NAME	TYPE	DESCRIPTION
av_scan_time	integer	AV engines' processing time.
cdr_time	integer	Deep CDR engine's sanitization time.
dlp_time	integer	Proactive DLP engine's processing time.
extraction_time	integer	Archive extraction engine's processing time.
filetype_time	integer	FileType engine's processing time.
opswatfilescan_time	integer	OPSWAT Filescan engine's processing time.
others_time	integer	Total time elapsed for following processing tasks in the product (in milliseconds): * Decryption time (if receiving an encrypted file) * External scanner (if configured) * Post action (if configured) * Other internal processing time among components in the product
parse_dgsg_time	integer	Digital signature analyzing time.
vul_time	integer	Vulnerability engine's lookup time.
yara_time	integer	YARA engine's processing time.
filetype_wait_time	integer	FileType engine's wait time.
profile	string	The used rule name.
progress_percentage	integer	Percentage of processing completed (from 1-100).
queue_time	integer	Total time elapsed for file processing task was waiting in MetaDefender Core's queue until being picked up (queue_time = start_time - upload_timestamp) (in milliseconds).
result	string	The final result of processing the file (Allowed / Blocked / Processing).
user_agent	string	Identifier for the REST Client that calls the API.
username	string	User identifier who submitted scan request earlier.
verdicts	array	
post_processing	object	
actions_failed	string	Empty string if no action failed or list of failed actions, separated by "|".
actions_ran	string	List of successful actions, separated by "|". Empty string if otherwise.
converted_destination	string	Contains the name of the sanitized file.
converted_to	string	Contains target type name of sanitization.
copy_move_destination	string	Contains target type name of sanitization.
sanitization_details	object	
cdr_wait_time	integer	The time in milliseconds that the CDR process took to complete.
description	string	Action was successful or not.
details	array	
action*	enum	**ALLOWED:** `sanitized, removed` The type of action that was performed

NAME	TYPE	DESCRIPTION
count	integer	The number of objects that were sanitized/removed.
details	object	
action	enum	ALLOWED: sanitized, removed The type of action that was performed
count	integer	The number of objects that were sanitized/removed.
object_details	array	
object_name	string	The object type that was sanitized/removed.
description	string	Action was successful or not.
file_name	string	If an embedded file was sanitized.
object_details	array	
object_name*	string	The object type that was sanitized/removed.
failure_category	string	Deep CDR errors are classified into different categories. For more details, please find [Troubleshooting sanitization failures](https://docs.opswat.com/mdcore/deep-cdr/troubleshooting-sanitization-failures)
result	enum	ALLOWED: Sanitized, Sanitized failed, Sanitized skipped The result of the CDR process. - **Sanitized**: the file was successfully sanitized. - **Sanitized failed**: the file could not be sanitized due to an error during the process. - **Sanitized skipped**: the file was skipped from sanitization. Common reasons include the file being digitally signed or other policy-based exclusions.
result_template_hash	string	The hash value of the result template, which is used for displaying results on the Core UI and for internal communication between MetaDefender Core and the Deep CDR engine. This value is intended for system use only and is not required for external integration.
sanitized_file_info	object	
file_size	integer	Size of sanitized file in bytes.
sha256	string	SHA256 hash of sanitized file.
verdict	enum	ALLOWED: blocked, allowed The verdict of the CDR process. - **blocked**: the file is recommended for blocking by Deep CDR. - **allowed**: the file is recommended for allowing by Deep CDR as it found no reason to recommend blocking it.
verdict_explanations	array	
client_identity	object	
clients	array	
deployment_id*	string	Unique identifier for the deployment.
user_name*	string	Name of the user or system that initiated the scan.
product_type*	string	Type of product or system that is performing the scan.

NAME	TYPE	DESCRIPTION
version	string	Version of the product or system.
host	string	Hostname or IP address of the client machine.
timestamp*	integer	Timestamp of when the scan was initiated, in milliseconds since epoch.
scan_results	object	
data_id	string	Data ID of the requested file
progress_percentage	integer	Track analysis progress until reaches 100.
scan_all_result_a	enum	**ALLOWED:** `No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Scan Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, In Progress, Skip Processing Fast Symlink` The overall scan result as string
scan_all_result_i	enum	**ALLOWED:** `0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 67, 68, 69, 70, 71, 255, 1014` The overall scan result as index in the Processing Results table.
start_time	string	Timestamp when the scanning process starts.
total_avs	integer	Total number of scanning engines used as part of this analysis.
total_time	integer	Total time elapsed during scan (in milliseconds).
scan_details	object	
ClamAV	object	
def_time	string	The database definition time for this engine
eng_id	string	The unique identification string for the engine
location	string	Where this engine is deployed (local/remote).
scan_result_i	integer	Scan result as index in the Processing Results table
scan_time	integer	The time elapsed during scan with this engine (in milliseconds).
threat_found	string	The threat name, IF scan result is Infected or Suspicious. Otherwise empty string or error message from the engine.

NAME	TYPE	DESCRIPTION
wait_time	integer	Time elapsed between sending file to Core and receiving the result from the engine (in milliseconds).
vulnerability_info	object	
result	object	
code	integer	The result code for vulnerability check, 0 means a successful check
hash	string	The file's SHA1 hash value
method	enum	ALLOWED: `50700` The method used by OESIS Framework, it should be 50700 every time.
timestamp	string	Timestamp of the request issued
timing	integer	The vulnerability check's duration in milliseconds
detected_product	object	
has_kb	boolean	Indicates whether any KBs or MSBs exist for this hash
has_vulnerability	boolean	Indicates whether any vulnerabilities have been associated with the particular product
is_current	boolean	True if this product's patch level is current, defaults to true
product	object	
id	integer	The OPSWAT product id
name	string	The product name
remediation_link	string	A link where product updates or patches can be obtained
severity	enum	ALLOWED: `LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN` String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`
sig_name	string	Product signature descriptor
signature	integer	OPSWAT signature id
vendor	object	
id	integer	The OPSWAT vendor id
name	string	The vendor name
version	string	The installed product version
version_data	object	
count_behind	integer	The number of patches behind of the installed product
feed_id	integer	The remote feed ID used to determine patch level
version	string	The current version of the product in the remote feed
vulnerabilites	array	

NAME	TYPE	DESCRIPTION
description	string	A text description of the specific vulnerability
details	object	
cpe	string	A CPE product reference
cve	string	A CVE identification string
cvss	object	
access-complexity	string	A CVSS access-complexity descriptor
access-vector	string	A CVSS access-vector descriptor
authentication	string	A CVSS authentication descriptor
availability-impact	string	A CVSS availability impact descriptor
confidentiality-impact	string	A CVSS confidentiality impact descriptor
generated-on-epoch	string	An epoch timestamp indicating CVSS generation time
integrity-impact	string	A CVSS integrity impact descriptor
score	string	A CVSS 10-point severity score
source	string	A CVSS source descriptor
cwe	string	A CWE group identification string
last_modified_epoch	string	An epoch timestamp indicating source last update time
published-epoch	string	An epoch timestamp indicating source publishing time
references	array	
severity	enum	**ALLOWED:** `LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN` String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`
severity_index	integer	A 5 point scale numerical description of Severity level with 5 being greatest and 0 being unknown
static_id	integer	An OPSWAT identifier for the vulnerability
verdict	integer	The vulnerability check's duration in milliseconds * `0` - No Vulnerability Found * `1` - Vulnerability Found * `3` - Failed * `16` - Processing Timed Out
yara	object	
hits	object	

NAME	TYPE	DESCRIPTION
verdict	enum	**ALLOWED:** `0, 1, 2, 3, 4` The overall result for the analyzed file. Value will be one of the following: | index | status | |---|---| | 0 | Clean | | 1 | Found matched data | | 2 | Suspicious | | 3 | Failed | | 4 | Not scanned |

STATUS CODE - 404: Invalid hash format

1.4 GET /hash/{md5|sha1|sha256|sha512}/blocked-leaves

Retrieve blocked leaf files from an archive by hash

Returns the deepest blocked files found within an original archive, identified by the archive's hash value. A leaf file is a file that has no successfully extracted child files. If no blocked leaf files are found, the response returns an empty array. A maximum of 100 blocked leaf files is returned in a single request.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*md5|sha1|sha256|sha512	string	8101abae27be4d63859c55d9e0ed0135	Hash value of the archive. Supported hash types are MD5, SHA-1, SHA-256, and SHA-512.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
rule	string		Select rule for the analysis, if no header given the default rule will be selected (URL encoded UTF-8 string of rule name)

NAME	TYPE	EXAMPLE	DESCRIPTION
timerange	integer		Limits the hash lookup to files analyzed within the specified number of hours. A value of `0` (default) searches the entire analysis history.

RESPONSE

STATUS CODE - 200: Successfully retrieved the list of blocked leaf files.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
details	array	
blocked_reasons	array	
engines	array	
engine	string	Name of the scanning engine.
threat_name	string	Threat detected by the engine.
data_id	string	Unique identifier of the blocked leaf file.
display_name	string	Name of the blocked leaf file.
file_type	string	MIME type of the file.
parent_id	string	Data identifier of the original archive.
path	string	Full path of the blocked leaf file within the archive.
process_info	object	
post_processing	object	
actions_failed	string	Failed post-processing actions, separated by `|`. Empty if none.
actions_ran	string	Successful post-processing actions, separated by `|`. Empty if none.
verdicts	array	
yara_info	object	
hits	array	
limit_reached	boolean	Indicates whether more than 100 blocked leaf files were found.
total	integer	Number of blocked leaf files returned in the `details` array.

EXAMPLE 1:

```
"sample"
```

STATUS CODE - 400: The `timerange` header value is invalid.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

STATUS CODE - 404: The specified hash was not found, or the file is still being processed.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

1.5 GET /file/rules

Fetching Available Analysis Rules

Retrieve all available rules with their custom configurations. Fetching available processing rules.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication. Only those rules are returned, that: * Match the apikey's role sent using the apikey header, or * Are not restricted to a specific role.
user_agent	string		The user agent string value sent in the header (specified by the client). Only those rules are returned, that: * Match the client's user agent sent using the user_agent header, or * Are not restricted to a specific user agent. For details see KB article [What are Security Policies and how do I use them?](https://onlinehelp.opswat.com/corev4/What_are_Security_Policies_and_how_do_I_use_them_.html).

RESPONSE

STATUS CODE - 200: Returns the list of available rules.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
max_file_size	integer	The maximum allowed file size (in bytes) for this rule.
name	string	A unique identifier for identify in the used rule for a scan..
global_timeout	object	
value	integer	The timeout value in seconds.
enabled	boolean	Indicates whether the global timeout is enabled.

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

1.6 GET /file/converted/{data_id}

Download Sanitized Files

Retrieve sanitized file based on the data_id

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string	8101abae27be4d63859c55d9e0ed0135	The data_id comes from the result of `Analyze a file`. In case of sanitizing the content of an archive, the data_id of contained file can be found in `Fetch analysis result`.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Returns the sanitized content.

RESPONSE MODEL - application/octet-stream

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

1.7 GET /file/download/{data_id}

Download either sanitized files or DLP processed files

Retrieve sanitized file based on the data_id. In case there's no sanitized file, and DLP processed file is available, user will retrieve DLP processed file.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string	8101abae27be4d63859c55d9e0ed0135	The data_id comes from the result of `Analyze a file`. In case of sanitizing the content of an archive, the data_id of contained file can be found in `Fetch analysis result`.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Returns the sanitized or DLP processed content.

RESPONSE MODEL - application/octet-stream

STATUS CODE - 404: File could not be found

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "File could not be found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

1.8 POST /file/{data_id}/cancel

Cancel File Analysis

When cancelling a file analysis, the connected analysis (e.g. files in an archive) that are still in progress will be cancelled also.

The cancelled analysis will be automatically closed.

When the file was submitted with the downloadfrom header and is still being downloaded, the download is stopped and download_info.status becomes Download Cancelled.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string		Unique submission identifier. Use this value to reference the submission.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Analysis was sucessfully cancelled.

RESPONSE MODEL - application/json

EXAMPLE:

```
{
 "<<data_id>>": "cancelled"
}
```

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Data ID not found (invalid id) or Requests resource was not found

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

1.9 GET /file/{data_id}/blocked-leaves

Retrieve blocked leaf files from an archive

Returns the deepest blocked files found within an original archive, identified by the archive's data_id. A leaf file is a file that has no successfully extracted child files. If no blocked leaf files are found, the response returns an empty array. A maximum of 100 blocked leaf files is returned in a single request.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string	8101abae27be4d63859c55d9e0ed0135	The `data_id` returned by the **Analyze a file** API. When analyzing or sanitizing an archive, the `data_id` of files contained within the archive can be obtained from the **Fetch analysis result** API.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Successfully retrieved the list of blocked leaf files.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
details	array	
blocked_reasons	array	
engines	array	
engine	string	Name of the scanning engine.
threat_name	string	Threat detected by the engine.
data_id	string	Unique identifier of the blocked leaf file.
display_name	string	Name of the blocked leaf file.
file_type	string	MIME type of the file.
parent_id	string	Data identifier of the original archive.
path	string	Full path of the blocked leaf file within the archive.
process_info	object	
post_processing	object	
actions_failed	string	Failed post-processing actions, separated by `|`. Empty if none.
actions_ran	string	Successful post-processing actions, separated by `|`. Empty if none.
verdicts	array	
yara_info	object	
hits	array	
limit_reached	boolean	Indicates whether more than 100 blocked leaf files were found.
total	integer	Number of blocked leaf files returned in the `details` array.

EXAMPLE 1:

```
"sample"
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

1.10 GET /file/{data_id}/extraction-errors

Fetch the Top 100 Extraction Errors in an Archive

Returns up to 100 extraction errors found while processing the archive identified by data_id, including errors from nested archives. The response includes:

- total: Number of archives (root and nested) that reported extraction errors.

- root_archive: Extraction error details for the top-level archive. Omitted if no root-level errors exist.
- nested_archives: Extraction error details for nested archives.

If no extraction errors are found:

- total is 0.
- nested_archives may be empty.
- root_archive may be omitted.

If the archive result was reused from a previous analysis, the API returns the extraction errors from that analysis. You can identify this case when original_data_id is set in the GET /file/{data_id} response. The first segment of each archive_path is updated to use the name of the requested file. This behavior is available starting with MetaDefender Cluster 2.10.0.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string	8101abae27be4d63859c55d9e0ed0135	Data identifier of the original (root) archive file.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Extraction error information for the specified archive.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
total	integer	Total count of archives (root and nested) that have extraction errors.
root_archive	object	

NAME	TYPE	DESCRIPTION
err_category	string	High-level error category.
err_details	string	Detailed diagnostic message.
err_description	string	Human-readable description of the error.
nested_archives	array	
data_id	string	Data identifier of the nested archive.
archive_path	array	
err_category	string	High-level error category.
err_details	string	Detailed diagnostic message.
err_description	string	Human-readable description of the nested archive issue.

EXAMPLE 1:

```
"sample"
```

STATUS CODE - 400: Invalid request (e.g. malformed data_id or bad query parameters).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Description of the error.

EXAMPLE:

```
{
 "err": "Invalid query parameters"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 423: The file is still being processed; extraction errors not finalized yet.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Description of the error.

EXAMPLE:

```
{
 "err": "File is on processing. Abort"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

1.11 GET /file/webhook/{data_id}

Query webhook status

Prior to being notified when webhook mode is enabled, the client can request MetaDefender Cluster API Gateway for the file processing webhook status at any time.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string		The `data_id` of the file to query.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Webhook status is fetched successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
data_id	string	The file submission identifier
request_time	string	A timestamp when the request has been made.
status_code	integer	What was the returned HTTP status code. * `200` - Callback was sent successfully * `403` - ContentAccessDenied. The access to the remote content was denied (similar to HTTP(S) error 401). * `404` - ContentNotFoundError. The remote content was not found at the server (similar to HTTP(S) error 404). * `408` - TimeoutError. The connection to the remote server timed out. * `503` - HostNotFoundError. The remote host name was not found (invalid hostname). * `520` - RemoteHostClosedError. The remote server closed the connection prematurely, before the entire reply was received and processed. * `444` - Other error types.
url	string	What was the called URL (should match the `callbackurl` header).

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

2. AUTH

Authentication APIs

User authentication is done via username & password.

2.1 POST /login

Login

Initiate a new session. Required for using protected REST APIs.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
user*	string	Username
password*	string	User's password

EXAMPLE:

```
{
 "user": "admin",
 "password": "admin"
}
```

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
oms-csrf-token*	string	The randomly generated token used to prevent CSRF attacks
session_id*	string	The apikey used to make API calls which requires authentication

STATUS CODE - 403: Invalid credentials

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	<error message> will describe the incident. More details would be logged in MetaDefender Cluster services logs

EXAMPLE:

```
{
 "err": "Failed to login"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

2.2 POST /logout

Logout

Destroy session for not using protected REST APIs.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
response*	string	

STATUS CODE - 400: Bad Request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err*	string	

STATUS CODE - 403: Invalid user information.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err*	string	

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

3. BATCH

Group the analysis requests in batches. Supported with endpoints: MetaDefender Cluster API Gateway.

3.1 POST /file/batch

Initiate Batch

Create a new batch and retrieve the batch_id

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
rule	string		Select rule for the analysis, if no header given the default rule will be selected (URL encoded UTF-8 string of rule name)
user_agent	string		user_agent header used to identify (and limit) access to a particular rule. For rule selection, `rule` header should be used.
user-data	string		Name of the batch (max 1024 bytes, URL encoded UTF-8 string).

NAME	TYPE	EXAMPLE	DESCRIPTION
client-identity	json		The `client-identity` header identifies the clients involved in processing a file, providing greater visibility into its processing stages. The header value is a JSON object containing a clients array. Each `client` object supports the following fields: * `deployment_id`: *REQUIRED*. Unique identifier of the deployment. * `user_name`: *REQUIRED*. Name of the user or system that initiated the scan. * `product_type`: *REQUIRED*. Type of product or system performing the scan. * `version`: *OPTIONAL*. Version of the product or system. * `host`: *OPTIONAL*. Hostname or IP address of the client machine. * `timestamp`: *REQUIRED*. Time at which the scan was initiated, expressed in milliseconds since the Unix epoch. Example: ```json { "clients": [{ "deployment_id": "abcdef1234567890", "user_name": "Gemini", "product_type": "MetaDefender Kiosk", "version": "1.0.0", "host": "client.example.com", "timestamp": 1744949754890 }] } ``` URL-encode the JSON value before sending the header to MetaDefender Cluster to avoid issues caused by unsafe characters or incorrect encoding. Omit the header when client identity tracking is not required.

RESPONSE

STATUS CODE - 200: Batch created successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
batch_id*	string	The batch identifier used to submit files in the batch and to close the batch.

EXAMPLE:


```
{
 "batch_id": "74c85f475147439bac4d33b181853923"
```

```
}
```

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or
out of range, etc). The request is rejected before the file is processed, so no `data_id` is created.

When the `client-identity` header is present but its value is invalid, `err` is one of:

`err`	Cause
`Header 'client-identity' is empty`	The header was sent without a value.
`Header 'client-identity' must be less than or equal 4096 characters`	The header value is longer than 4096 characters.
`Header 'client-identity' contains non-printable character`	The value contains an ASCII control character, either directly or through an escape.
`Header 'client-identity' is not in JSON format`	The value could not be URL-decoded, is not valid JSON, or is not a JSON object.
`Header 'client-identity' does not contain a 'clients' key or empty 'clients'`	`clients` is missing, is not an array, or is an empty array.
`Header 'client-identity' contains an invalid data`	An entry of `clients` is not a JSON object.
`Header 'client-identity' does not contain required key '<key>'`	An entry is missing `deployment_id`, `user_name`, `product_type` or `timestamp`.
`Header 'client-identity' contains a non-string '<key>'`	`deployment_id`, `user_name` or `product_type` is not a string.
`Header 'client-identity' contains an empty '<key>'`	`deployment_id`, `user_name` or `product_type` is blank.
`Header 'client-identity' contains an invalid 'deployment_id'`	`deployment_id` holds anything other than letters and digits.
`Header 'client-identity' contains a non-integer 'timestamp'`	`timestamp` is not an integer.
`Header 'client-identity' contains a negative 'timestamp'`	`timestamp` is zero or negative.

Every entry of `clients` is checked, not only the first one, and the first rule that fails is the one reported.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Header 'client-identity' contains an invalid 'deployment_id'"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

3.2 POST /file/batch/{batchId}/close

Close Batch

The batch will be closed and files can no longer be added to the current batch.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*batchId	string		The batch identifier used to submit files in the batch and to close the batch.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Batch successfully closed.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
batch_files	object	
batch_count	integer	The total number of files/entries in the batch.
current_finished_files	integer	The total number of files/entries that have been fully processed in the batch.
files_in_batch	array	
data_id	string	Unique identifer for the file.
detected_by	integer	Total number of engines that detected this file.
display_name	string	The filename reported via `filename` header.
file_size	integer	Total file size in bytes.
file_type	string	The filetype using mimetype.
file_type_description	string	The filetype in human readable format.
process_info	object	
blocked_reason	string	Provides the reason why the file is blocked (if so).
progress_percentage	integer	Percentage of processing completed (from 1-100).
result	string	The final result of processing the file (Allowed / Blocked / Processing).
verdicts	array	
client_identity	object	
clients	array	
deployment_id*	string	Unique identifier for the deployment.
user_name*	string	Name of the user or system that initiated the scan.
product_type*	string	Type of product or system that is performing the scan.
version	string	Version of the product or system.
host	string	Hostname or IP address of the client machine.
timestamp*	integer	Timestamp of when the scan was initiated, in milliseconds since epoch.
progress_percentage	integer	Track analysis progress until reaches 100.

NAME	TYPE	DESCRIPTION
scan_all_result_a	enum	**ALLOWED:** No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Scan Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, In Progress, Skip Processing Fast Symlink The overall scan result as string
scan_all_result_i	enum	**ALLOWED:** 0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 67, 68, 69, 70, 71, 255, 1014 The overall scan result as index in the Processing Results table.
scanned_with	integer	The total number of engines used to analyze this file.
first_index	integer	The starting index in the batch. Used for pagination.
page_size	integer	The number of entries per page.
batch_id	string	The batch unique identifer
is_closed	boolean	The batch status (open/close).
process_info	object	
blocked_reason	string	Provides the reason why the file is blocked (if so).
file_type_skipped_scan	boolean	Indicates if the input file's detected type was configured to skip scanning.
profile	string	The used rule name.
result	string	The final result of processing the file (Allowed / Blocked / Processing).
user_agent	string	Identifier for the REST Client that calls the API.
username	string	User identifier who submitted scan request earlier.
client_identity	object	
clients	array	
deployment_id*	string	Unique identifier for the deployment.
user_name*	string	Name of the user or system that initiated the scan.
product_type*	string	Type of product or system that is performing the scan.
version	string	Version of the product or system.
host	string	Hostname or IP address of the client machine.

NAME	TYPE	DESCRIPTION
timestamp*	integer	Timestamp of when the scan was initiated, in milliseconds since epoch.
scan_results	object	
batch_id	string	The batch unique identifer
scan_all_result_a	enum	ALLOWED: No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Scan Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, In Progress, Skip Processing Fast Symlink The overall scan result as string
scan_all_result_i	enum	ALLOWED: 0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 67, 68, 69, 70, 71, 255, 1014 The overall scan result as index in the Processing Results table.
start_time	string	Timestamp when the scanning process starts.
total_avs	integer	Total number of scanning engines used as part of this analysis. Not like files, batch is not processed by engine, so this value is always 0.
total_time	integer	Total time elapsed during scan (in milliseconds).
user_data	string	Metadata submitted at batch creation.

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

3.3 GET /file/batch/{batchId}

Status of Batch Analysis

Retrieve status report for the entire batch

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*batchId	string		The batch identifier used to submit files in the batch and to close the batch.

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
first	integer >=0		The first item order in the list of files in this batch
size	integer between 0 and 2000		The number of items to be fetched next, counting from the item order indicated in `first` header. The default value is 50, and the maximum value is 2000.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Batch progress paginated report (50 entries/page).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
batch_files	object	
batch_count	integer	The total number of files/entries in the batch.
current_finished_files	integer	The total number of files/entries that have been fully processed in the batch.
files_in_batch	array	
data_id	string	Unique identifer for the file.
detected_by	integer	Total number of engines that detected this file.
display_name	string	The filename reported via `filename` header.
file_size	integer	Total file size in bytes.
file_type	string	The filetype using mimetype.
file_type_description	string	The filetype in human readable format.

NAME	TYPE	DESCRIPTION
process_info	`object`	
blocked_reason	`string`	Provides the reason why the file is blocked (if so).
progress_percentage	`integer`	Percentage of processing completed (from 1-100).
result	`string`	The final result of processing the file (Allowed / Blocked / Processing).
verdicts	`array`	
client_identity	`object`	
clients	`array`	
deployment_id*	`string`	Unique identifier for the deployment.
user_name*	`string`	Name of the user or system that initiated the scan.
product_type*	`string`	Type of product or system that is performing the scan.
version	`string`	Version of the product or system.
host	`string`	Hostname or IP address of the client machine.
timestamp*	`integer`	Timestamp of when the scan was initiated, in milliseconds since epoch.
progress_percentage	`integer`	Track analysis progress until reaches 100.
scan_all_result_a	`enum`	**ALLOWED:** `No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Scan Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, In Progress, Skip Processing Fast Symlink` The overall scan result as string
scan_all_result_i	`enum`	**ALLOWED:** `0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 67, 68, 69, 70, 71, 255, 1014` The overall scan result as index in the Processing Results table.
scanned_with	`integer`	The total number of engines used to analyze this file.
first_index	`integer`	The starting index in the batch. Used for pagination.
page_size	`integer`	The number of entries per page.
batch_id	`string`	The batch unique identifer
is_closed	`boolean`	The batch status (open/close).

NAME	TYPE	DESCRIPTION
process_info	`object`	
blocked_reason	`string`	Provides the reason why the file is blocked (if so).
file_type_skipped_scan	`boolean`	Indicates if the input file's detected type was configured to skip scanning.
profile	`string`	The used rule name.
result	`string`	The final result of processing the file (Allowed / Blocked / Processing).
user_agent	`string`	Identifier for the REST Client that calls the API.
username	`string`	User identifier who submitted scan request earlier.
client_identity	`object`	
clients	`array`	
deployment_id*	`string`	Unique identifier for the deployment.
user_name*	`string`	Name of the user or system that initiated the scan.
product_type*	`string`	Type of product or system that is performing the scan.
version	`string`	Version of the product or system.
host	`string`	Hostname or IP address of the client machine.
timestamp*	`integer`	Timestamp of when the scan was initiated, in milliseconds since epoch.
scan_results	`object`	
batch_id	`string`	The batch unique identifer
scan_all_result_a	`enum`	**ALLOWED:** `No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Scan Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, In Progress, Skip Processing Fast Symlink` The overall scan result as string
scan_all_result_i	`enum`	**ALLOWED:** `0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 67, 68, 69, 70, 71, 255, 1014` The overall scan result as index in the Processing Results table.
start_time	`string`	Timestamp when the scanning process starts.
total_avs	`integer`	Total number of scanning engines used as part of this analysis. Not like files, batch is not processed by engine, so this value is always 0.

NAME	TYPE	DESCRIPTION
total_time	integer	Total time elapsed during scan (in milliseconds).
user_data	string	Metadata submitted at batch creation.

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

3.4 GET /file/batch/{batchId}/certificate

Download Signed Batch Result

Download digitally signed status report for the entire batch

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*batchId	string		The batch identifier used to submit files in the batch and to close the batch.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
metadata	json		In JSON format, this can be used to: Include additional information in the response YML. Currently, one supported field in the metadata is `include_vul_info`, which can be set to `true` or `false` to indicate whether vulnerability processing information should be included or not. It is strongly recommended to apply URL encoding before sending `metadata` to Metadefender Cluster API Gateway to prevent unexpected issues related to encoding errors or unsafe characters.

RESPONSE

STATUS CODE - 200: Signed batch result and certificate are sent back in response body (YAML format).

RESPONSE MODEL - application/x-yaml

EXAMPLE:


```
"--- batch_id: 092876200fb54cfb80b6e3332c410ae9 user_data: the user data from the header from
batch creation cert_sha1_fingerprint: <some cert serial value> batch_files:\n  batch_count:
1\n  files_in_batch:\n  - data_id: 9112b225f0634f189a2bb46ec1a7826f\n    display_name:
New%20Text%20Document.txt\n    file_size: 5\n    scan_all_result_i: 0\n    process_info:\n
blocked_reason:\n      result: Allowed\n    md5: 42b130c3ce46e058f30712838cebf420\n    sha1:
ed94baf55ca851055fb76045f6949bca2f865605\n    sha256:
f4191b3ec6ce93aaf712919a38e52815c5da9c91d2b141df920bc8bcb5cbb8e3\n    sha512: \"\"\n
vulnerabilities:\n      - cve: CVE-2021-45463\n        cvss:\n          score: 6.8\n
cvss_3_0:\n          base_score: 7.8\n      - cve: CVE-2018-12713\n        cvss:\n
score: 6.4\n        cvss_3_0:\n          base_score: 9.1\nprocess_info:\n  blocked_reason:\n
file_type_skipped_scan: false\n  profile: File scan\n  result: Allowed\n  user_agent:
webscan\nscan_results:\n  scan_all_result_a: No Threat Detected\n  scan_all_result_i: 0\n
start_time: 2017-05-23T11:22:03.010Z\n  total_avs: 14\n  total_time: 995\n...\n--- signature:
881d22220c4ca0557d7c7d5c5794d53a8a2780997cd65b27b6e7f1c099a15de03dbcb5edbeaea7aafa6099fab37be
07017b39e3e3a7d66c550f44eb59a096c54d5b9555cb28198546fbec57c33b717751d333a09733d95dd876e2798d0
44c8caef828f4352b91f9a6d057253bb1a9461e0e0e0bf4313a80895998d645bebc81841ff3499589c80ffc4e8a19
0d1ec9b3e4126d86659d303b0e1f22d9289c9c4671d35532b55ad4620e048a78bb405b573897da63efdd5f036692c
934a82d9bdc9b9862e7fea5e8abeeb1444be0689d50373c5c0632484950c0fe0337ed5f91bdf26986f7cff8aa3431
bf4bc948fc127c16ba13ec679fe9f67e7586075c1f467454fa8cf40e9cd501291c95d862eb16f4477c17d1711294f
0ff2b3a1140bd53dbd1fbb0846af6062e9e4e2e1a09af3448503ed11e342164e535fc268bf7d8fbc28ed946cd2bb8
ea075f2295d2fa8392076d41608c3b5decf8fab3a5ec7de190f07583331e0517e5f361735cd59326622dc8b07b10a
464028de781a063e408f918c1d5534329140f4e4dc1a717d808d6784410410b00d36cb9a345f5bbc11fa1c58ee28f
8e7b863f3ea2c923ec5fb2ac29eaa4ddc0d6d9dfd3f16a97f207dc2858410a577c7f4a92ff01bad3229f5fcdb08e2
1df9869a113272aa9d96bfdfe8bfb3a50414c174e16a3504e5780c2718779b0757298546f287ef7ea86e67510d48a
8 certificate: |\n  -----BEGIN CERTIFICATE-----\n
MIIGJzCCBA+gAwIBAgIBATANBgkqhkiG9w0BAQUFADCBsjELMAkGA1UEBhMCRlIx\n
DzANBgNVBAgMBkFsc2FjZTETMBEGA1UEBwwKU3RyYXNib3VyZzEYMBYGA1UECgwP\n
d3d3LmZyZWVsYW4ub3JnMRAwDgYDVQQLDAdmcmVlbGFuMS0wKwYDVQQDDCRGcmVl\n
bGFuIFNhbXBsZSBDZXJ0aWZpY2F0ZSBBdXRob3JpdHkxIjAgBgkqhkiG9w0BCQEW\n
E2NvbnRhY3RAZnJlZWxhbi5vcmcwHhcNMTIwNDI3MTAzMTE4WhcNMjIwNDI1MTAz\n
MTE4WjB+MQswCQYDVQQGEwJGUjEPMA0GA1UECAwGQWxzYWNlMRgwFgYDVQQKDA93\n
d3cuZnJlZWxhbi5vcmcxEDAOBgNVBAsMB2ZyZWVsYW4xDjAMBgNVBAMMBWFsaWNl\n
MSIwIAYJKoZIhvcNAQkBFhNjb250YWN0QGZyZWVsYW4ub3JnMIICIjANBgkqhkiG\n
9w0BAQEFAAOCAg8AMIICCgKCAgEA3W29+ID6194bH6ejLrIC4hb2Ugo8v6ZC+Mrc\n
k2dNYMNPjcOKABvxxEtBamnSaeU/IY7FC/giN622LEtV/3oDcrua0+yWuVafyxmZ\n  yTKUb4/GUgafRQPf/
eiX9urWurtIK7XgNGFNUjYPq4dSJQPPhwCHE/LKAykWnZBX\n
RrX0Dq4XyApNku0IpjIjEXH+8ixE12wH8wt7DEvdO7T3N3CfUbaITl1qBX+Nm2Z6\n  q4Ag/
u5rl8NJfXg71ZmXA3XOj7zFvpyapRIZcPmkvZYn7SMCp8dXyXHPdpSiIWL2\n  uB3KiO4JrUYvt2GzLBUThp+lNSZaZ/
Q3yOaAAUkOx+1h08285Pi+P8lO+H2Xic4S\n
vMq1xtLg2bNoPC5KnbRfuFPuUD2/3dSiiragJ6uYDLOyWJDivKGt/72OVTEPAL9o\n
6T2pGZrwbQuiFGrGTMZOvWMSpQtNl+tCCXlT4mWqJDRwuMGrI4DnnGzt3IKqNwS4\n
Qyo9KqjMIPwnXZAmWPm3FOKe4sFwc5fpawKO01JZewDsYTDxVj+cwXwFxbE2yBiF\n
z2FAHwfopwaH35p3C6lkcgP2k/zgAlnBluzACUI+MKJ/G0gv/uAhj1OHJQ3L6kn1\n  SpvQ41/
ueBjlunExqQSYD7GtZ1Kg8uOcq2r+WISE3Qc9MpQFFkUVllmgWGwYDuN3\n
Zsez95kCAwEAAaN7MHkwCQYDVR0TBAIwADAsBglghkgBhvhCAQ0EHxYdT3BlblNT\n
TCBHZW5lcmF0ZWQgQ2VydGlmaWNhdGUwHQYDVR0OBBYEFFlfyRO6G8y5qEFKikl5\n
ajb2fT7XMB8GA1UdIwQYMBaAFCNsLT0+KV14uGw+quK7Lh5sh/JTMA0GCSqGSIb3\n
DQEBBQUAA4ICAQAT5wJFPqervbja5+90iKxi1d0QVtVGB+z6aoAMuWK+qgi0vgvr\n
mu9ot2lvTSCSnRhjeiP0SIdqFMORmBtOCFk/kYDp9M/91b+vS+S9eAlxrNCB5VOf\n  PqxEPp/wv1rBcE4GBO/
```

```
c6HcFon3F+oBYCsUQbZDKSSZxhDm3mj7pb67FNbZbJIzJ\n
70HDsRe2O04oiTx+h6g6pW3cOQMgIAvFgKN5Ex727K4230B0NIdGkzuj4KSML0NM\n
slSAcXZ41OoSKNjy44BVEZv0ZdxTDrRM4EwJtNyggFzmtTuV02nkUj1bYYYC5f0L\n
ADr6s0XMyaNk8twlWYlYDZ5uKDpVRVBfiGcq0uJIzIvemhuTrofh8pBQQNkPRDFT\n  Rq1iTo1Ihhl3/
Fl1kXk1WR3jTjNb4jHX7lIoXwpwp767HAPKGhjQ9cFbnHMEtkro\n
RlJYdtRq5mccDtwT0GFyoJLLBZdHHMHJz0F9H7FNk2tTQQMhK5MVYwg+LIaee586\n
CQVqfbscp7evlgjLW98H+5zylRHAgoH2G79aHljNKMp9BOuq6SnEglEsiWGVtu2l\n  hnx8SB3sVJZHeer8f/
UQQwqbAO+Kdy70NmbSaqaVtp8jOxLiidWkwSyRTsuU6D8i\n
DiH5uEqBXExjrj0FslxcVKdVj5glVcSmkLwZKbEU1OKwleT/iXFhvooWhQ==\n  -----END CERTIFICATE-----
\n...\n"
```

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

`3.5 POST /file/batch/{batchId}/cancel`

Cancel Batch

When cancelling a batch, the connected analysis that are still in progress will be cancelled also.

The cancelled batch will be closed.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*`batchId`	`string`		The batch identifier used to submit files in the batch and to close the batch.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
`apikey`	`string`		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Batch cancelled.

RESPONSE MODEL - application/json

EXAMPLE:

```
{
 "<<batch_id>>": "cancelled"
}
```

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Batch not found (invalid id)

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

---

4. LICENSE

Retrieve the current license information.

4.1 GET /admin/license

Get current license information

Fetch details about the longest expiry active license among all activated licenses.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Information about the licensed product (product type, number of activations, deploymentId, expiration date and days left)

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
days_left	integer	Number of days left before expiration
expiration	string	Expiration date in MM/DD/YYYY format.
licensed_engines	array	
licensed_to	string	Name of the entity to which the license is issued.
max_agent_count	string	Total number of deployed MetaDefender Agents attached to this MetaDefender Core instance.
online_activated	boolean	Track online/offline activation mode
product_id	string	Official MetaDefender base SKU licensed.

NAME	TYPE	DESCRIPTION
product_name	string	Official MetaDefender base product name licensed.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

5. STATS

Health check and statistics about MetaDefender Core instance usage.

5.1 GET /stat/engines

Engine Status

Return the status of the latest engines between the MetaDefender Core instances.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: An array with all the engines and their details.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
abandoned	boolean	Indicates if this engine is abandoned.
active	boolean	If used by at least one engine
def_time	string	The database definition time for this engine
download_progress	integer	The percentage progress of download
download_time	string	When this engine downloaded from the update server.
eng_id	string	Engine internal ID
eng_name	string	Engine name
eng_type	string	Engine type in human readable form

NAME	TYPE	DESCRIPTION
eng_ver	`string`	Engine's version (format differs from one engine to another).
engine_type	`enum`	**ALLOWED:** `av, archive, filetype` Engine's type: * av * archive * filetype
notified_messages	`array`	A list of messages from engine.
pinned	`boolean`	Indicate if this engine is pinned.
state	`enum`	**ALLOWED:** `downloading, downloaded, staging, production, removed, temporary failed, permanently failed, content invalid, download failed` Status of the engine: * downloading * downloaded * staging * production * removed * temporary failed * permanently failed * content invalid * download failed
type	`string`	The type of information, whether it is engine or engine's database.

`5.2 GET /stat/nodes`

Instance Status Overview

Retrieve status details of all available MetaDefender Core instances.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	`string`		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Status details of MetaDefender Core instances.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
external_nodes_allowed	boolean	Indicates whether external nodes can connect; always true.
max_node_count	integer	Total number of available MetaDefender Core instances.
statuses	array	
address	string	Location of the Core instance; currently always return empty string.
available_mem	integer	The number of available RAM in this system.
cpu_cores	integer	The number of CPU Cores allocated to this Core instance.
current_processing_files	integer	Number of objects currently being processed by the Core instance.
engines	array	
active	boolean	If used by at least one engine
db_ver	string	The database version for this engine
def_time	string	The database definition time for this engine
download_time	string	The database download time for this engine
eng_name	string	Engine name
eng_ver	string	Engine's version (format differs from one engine to another).
engine_type	enum	ALLOWED: `av, archive, filetype` Engine's type: * av * archive * filetype
id	string	Engine internal ID
issues	array	A list of all potential problems on this engine.
free_disk_space	integer	Reported available disk on Core instance (in bytes).
id	string	Identifier of the worker that deployed this Core instance.
info_disk_space	array	
dirs	array	list of directories used by MetaDefender Core.
free	integer	Free space on the disk (in bytes).
location	string	Disk location.
total	integer	Total space on the disk (in bytes).
issues	array	
description	string	Text detailing the issue.
severity	string	How important is the reported issue.
load	integer	Current CPU utilization on Core instance (in percentage).
os	string	Current used OS
scan_queue	integer	Number of objects currently being processed by the Core instance.

NAME	TYPE	DESCRIPTION
scan_queue_details	object	
archive_scan_queue_ratio	number	Ratio of archive scan queue, always -1 for Core in Cluster mode.
available_slots	integer	The number of slots is available, always -1 for Core in Cluster mode.
extracted_file_slots	integer	Number of child files being processing
file_slots	integer	Number of files taken from REST by the current Core instance
total_scan_queue	integer	Total scan queue, always -1 for Core in Cluster mode.
total_disk_space	integer	The amount of disk space is allocated on Core instance (in Byte).
total_mem	integer	How much memory is allocated on Core instance (in MB).
total_scan_queue	integer	The maximum queue size is allowed, always -1 for Core in Cluster mode.
uptime	integer	How long this Core is in operation (in second).
version	string	Product version

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

5.3 GET /readyz

Get health check status

Fetch current status of system health.

REQUEST

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
`verbose`	`boolean`	true	Optional. Show detailed result of system health.

RESPONSE

STATUS CODE - 200: System is currently healthy.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
status*	`boolean`	System-wide status, indicate if all components are healthy.
scan_queue*	`object`	
number_in_queue*	`integer`	Number of objects being processed by the system.
status*	`boolean`	The operational status of the scan process; true if the system contains the required minimum of healthy MetaDefender Core instances.
license*	`object`	
status*	`enum`	ALLOWED: `expired, invalid, ok` License status.
components*	`object`	
status*	`boolean`	Aggregate component status.
clusterdb	`object`	
status	`boolean`	Configuration database overall status.
detail	`string`	Status detail message.
datalake	`object`	
status	`boolean`	DataLake overall status.
detail	`string`	Status detail message
caching	`object`	
status	`boolean`	Caching overall status.
detail	`string`	Status detail message.
broker	`object`	

NAME	TYPE	DESCRIPTION
status	boolean	Broker overall status.
detail	string	Status detail message.
filestorage	object	
status	boolean	MetaDefender Cluster File storage overall status.
detail	string	Status detail message.
identity	object	
status	boolean	MetaDefender Cluster Identity service overall status.
detail	string	Status detail message.
ometascan*	object	
status*	boolean	MetaDefender Core overall status.
detail	string	Detail message.
instance	array	
callback-service*	object	
status*	boolean	Callback service overall status.
instance	array	
download-service*	object	
status*	boolean	Download Service overall status. `true` when at least one instance is healthy, or when no instance is deployed.
instance	array	

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

STATUS CODE - 503: System is currently unhealthy.

API Reference

Control Center

API Version: v2.10.0

Developer Guide

This is the API documentation for *MetaDefender Cluster Control Center Public API*. If you would like to evaluate or have any questions about this documentation, please contact us via our Contact Us form.

How to Interact with MetaDefender Cluster Control Center using REST API

The MetaDefender Cluster Control Center empowers administrators and system engineers to efficiently manage system operations, including:

1. Establishing and maintaining essential service connections.

2. Deploying and managing MetaDefender Core, MetaDefender Cluster API Gateway instances.
3. Managing licenses.
4. Administering user accounts and access controls.
5. Configuring and enforcing security protocols.
6. Monitoring the overall system health and system performance.

OPSWAT recommends using the JSON-based REST API. The available methods are documented below.

OPSWAT provides some sample codes on GitHub to make it easier to understand how the MetaDefender REST API works.

CONTACT

NAME: API Support
EMAIL: feedback@opswat.com
URL: https://github.com/OPSWAT/metadefender-core-openapi3
Terms of service: https://onlinehelp.opswat.com/policies/

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
apikey	apiKey	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

API

1. USER MANAGEMENT

User management APIs

The APIs for manage users and user directories.

`1.1 GET /admin/user`

List all users

Returns a list of all users in the server.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: List of users retrieved successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
api_key	string	Associated apikey with this user
directory_id	integer	To which User Directories belongs to (LOCAL/System/etc.)
display_name	string	Which is the name showed in the Management Console
email	string	User's email address
id	integer	User's unique identifier

NAME	TYPE	DESCRIPTION
name	`string`	User's full name
description	`string`	User's description, 256 characters maximum
roles	`array`	
ui_settings	`object`	

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

1.2 POST /admin/user

Create user

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
api_key	`string`	Associated apikey with this user
directory_id	`integer`	To which User Directories belongs to (LOCAL/System/etc.)
display_name	`string`	Which is the name showed in the Management Console
email	`string`	User's email address
name	`string`	User's full name
description	`string`	User's description, 256 characters maximum
roles	`array`	
ui_settings	`object`	
password	`string`	The user's password

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	`string`		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
api_key	`string`	Associated apikey with this user
directory_id	`integer`	To which User Directories belongs to (LOCAL/System/etc.)
display_name	`string`	Which is the name showed in the Management Console
email	`string`	User's email address
name	`string`	User's full name
description	`string`	User's description, 256 characters maximum
roles	`array`	
ui_settings	`object`	

STATUS CODE - 400: Bad Request (e.g. invalid header, invalid request body).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

1.3 DELETE /admin/user/{user_id}

Delete a user

Delete a user by id from the system.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

EXAMPLE:

```
{
 "result": "Success"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Item does not exist"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not allowed"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

1.4 POST /user/changepassword

Change Password for local user

Modify the current password for the user identified by apikey

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
old_password	string	The current password in plain text
new_password	string	The new password in plain text

EXAMPLE:

```
{
 "old_password": "admin",
 "new_password": "123456"
}
```

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
result	string	

STATUS CODE - 400: Bad Request (e.g. invalid header, invalid request body).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

2. ADMIN

Admin specific API requests.

2.1 GET /admin/userdirectory

List all user directories

Retrieve a list of all user directories configured in the system.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: List of user directories retrieved successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
id	integer	Internal used identifier
name	string	Name of the user directory
type	string	Type of the user directory (e.g., LDAP, Local, etc.)
enabled	boolean	If the user directory is enabled or not
lockout_attempts	integer	Number of failed login attempts before the user is locked out
lockout_timeout	integer	Time in seconds before the user can try to log in again after being locked out

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

2.2 POST /admin/role

Create new role

Add a new user role to the system.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
name	string	The name identifier of the role
display_name	string	The extended name showed in the Management Console.
rights	object	
cert	array	
configlog	array	
engines	array	
license	array	
retention	array	
rule	array	
scanlog	array	
update	array	
updatelog	array	
users	array	
workflow	array	
zone	array	
healthcheck	array	
fetch	array	
download	array	
deployment	array	
service	array	
packageupload	array	

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
name	string	The name identifier of the role
display_name	string	The extended name showed in the Management Console.
rights	object	
cert	array	
configlog	array	
engines	array	
license	array	
retention	array	
rule	array	
scanlog	array	
update	array	
updatelog	array	
users	array	
workflow	array	
zone	array	
healthcheck	array	
fetch	array	
download	array	
deployment	array	
service	array	
packageupload	array	
editable*	boolean	If the role can be altered or not.
id*	integer	Internal used identifier

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

2.3 DELETE /admin/role/{role_id}

Delete a role

Delete a role by id from the system.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

EXAMPLE:

```
{
 "result": "Success"
}
```

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Item does not exist"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not allowed"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

3. AUTH

Authentication APIs

User authentication is done via username & password.

3.1 POST /login

Login

Initiate a new session. Required for using protected REST APIs.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
user*	string	Username
password*	string	User's password

EXAMPLE:

```
{
 "user": "admin",
 "password": "admin"
}
```

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
oms-csrf-token*	string	The randomly generated token used to prevent CSRF attacks
session_id*	string	The apikey used to make API calls which requires authentication

STATUS CODE - 403: Invalid credentials

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	<error message> will describe the incident. More details would be logged in MetaDefender Cluster services logs

EXAMPLE:

```
{
 "err": "Failed to login"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

3.2 POST /logout

Logout

Destroy session for not using protected REST APIs.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
response*	string	

STATUS CODE - 400: Bad Request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err*	string	

STATUS CODE - 403: Invalid user information.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err*	string	

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4. CONFIG

Configure the product through APIs (especially the Settings). Will require admin apikey..

4.1 PUT /admin/config/auditlog/cleanup

Audit clean up

Setting audit record cleanup time (cleanup records older than).

Note: The cleanup range is defined in hours.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted. _**Note**_: If `cleanuprange` is `0`, the cleanup functionality will be disabled.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.2 PUT /admin/config/filestorage/cleanup

File storage clean up

Setting file storage clean up time (clean up records older than).

Note:The clean up range is defined in hours.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted. _**Note**_: If `cleanuprange` is `0`, the cleanup functionality will be disabled.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.3 PUT /admin/config/warehouse/cleanup

Executive report clean up

Setting executive report clean up time (clean up records older than).

Note:The clean up range is defined in hours.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted. _**Note**_: If

NAME	TYPE	DESCRIPTION
		`cleanuprange` is `0`, the cleanup functionality will be disabled.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.4 PUT /admin/config/scanhistory/cleanup

Processing history clean up

Setting processing history clean up time (clean up records older than).

Note:The clean up range is defined in hours.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted. _**Note**_: If `cleanuprange` is `0`, the cleanup functionality will be disabled.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.5 GET /admin/config/healthcheck

Fetch Health Check Setting

Retrieves the current health check configuration, including the number of active MetaDefender Core (ometascan) instances required and the list of engines available together with their selection state.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
number_active_ometascan	integer	>=1 Minimum number of active MetaDefender Core (ometascan) instances required to be considered healthy. Must be at least 1.
number_active_av_engines	integer	>=1 Minimum number of active anti-virus engines required to be considered healthy. Must be at least 1.
engines	array	
display_name	string	Human readable name of the engine.
id	string	Internal engine identifier.
is_selected	boolean	Whether the engine is selected as required for the health check.

EXAMPLE 1:

```
"sample"
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.6 POST /admin/config/healthcheck

Configure Health Check Setting

Updates the health check configuration. The required engines listed in the request body will be marked as selected; all other engines will be de-selected.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
number_active_ometascan	integer	>=1 Minimum number of active MetaDefender Core (ometascan) instances required to be considered healthy. Must be at least 1.
number_active_av_engines	integer	>=1 Minimum number of active anti-virus engines required to be considered healthy. Must be at least 1.
required_engines	array	

EXAMPLE 1:

```
"sample"
```

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully. Returns the updated health check configuration.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
number_active_ometascan	integer	>=1 Minimum number of active MetaDefender Core (ometascan) instances required to be considered healthy. Must be at least 1.
number_active_av_engines	integer	>=1 Minimum number of active anti-virus engines required to be considered healthy. Must be at least 1.
engines	array	
display_name	string	Human readable name of the engine.
id	string	Internal engine identifier.
is_selected	boolean	Whether the engine is selected as required for the health check.

EXAMPLE 1:

```
"sample"
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.7 PUT /admin/config/session

Session settings

Configure settings for session generated upon a successful login See more at Login

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
absoluteSessionTimeout	integer	The interval (in milliseconds) for overall session length timeout (regardless of activity).
allowCrossIpSessions	boolean	Allow requests from the same user to come from different IPs.
allowDuplicateSession	boolean	Allow same user to have multiple active sessions.
sessionTimeout	integer	The interval (in milliseconds) for the user's session timeout, based on last activity. Timer starts after the last activity for the apikey.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
absoluteSessionTimeout	integer	The interval (in milliseconds) for overall session length timeout (regardless of activity).
allowCrossIpSessions	boolean	Allow requests from the same user to come from different IPs.
allowDuplicateSession	boolean	Allow same user to have multiple active sessions.
sessionTimeout	integer	The interval (in milliseconds) for the user's session timeout, based on last activity. Timer starts after the last activity for the apikey.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.8 GET /admin/config/sessioncookie

Get session cookie attributes

Getting session cookie attributes

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
samesite	number	SameSite attribute accepts three values: * `Strict` - cookies will only be sent in a first-party context, not be sent along with requests initiated by third party websites. * `Lax` - cookies are not sent on normal cross-site subrequests, but are sent when a user is navigating to the origin site. * `None` - cookies will be sent in all contexts. Default value: `Lax`

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
```

```
}
```

4.9 PUT /admin/config/sessioncookie

Update session cookie attributes

Modifying session cookie attributes

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
samesite	number	SameSite attribute accepts three values: * `Strict` - cookies will only be sent in a first-party context, not be sent along with requests initiated by third party websites. * `Lax` - cookies are not sent on normal cross-site subrequests, but are sent when a user is navigating to the origin site. * `None` - cookies will be sent in all contexts. Default value: `Lax`

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
samesite	number	SameSite attribute accepts three values: * `Strict` - cookies will only be sent in a first-party context, not be sent along with requests initiated by third party websites. * `Lax` - cookies are not sent on normal cross-site subrequests, but are sent when a user is navigating to the origin site. * `None` - cookies will be sent in all contexts. Default value: `Lax`

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

5. INSTALLERS

Upload and manage installers for the MetaDefender Core and MetaDefender Cluster API Gateway.

5.1 GET /admin/installer

Get uploaded installers

Retrieve information about an uploaded installer.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
```

```
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

5.2 POST /admin/installer

Upload installer

Upload installers for the MetaDefender Core, MetaDefender Cluster API Gateway and

MetaDefender Cluster Callback Service.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
*filename	string		The name of the installer file to upload. **Note**: Ensure the filename remains same with the original MY OPSWAT download (e.g: ometascan-5.15.0-1-x64.msi)

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
installer_id	string	Unique identifier of the uploaded installer.

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

5.3 DELETE /admin/installer/{installer_id}

Delete an uploaded installer

Delete an uploaded installer.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
result	string	Success message.

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

6. SERVICES

Add essential services and view connection status.

6.1 GET /admin/service

Get the status of all essential services.

Retrieve the status of all added services within the MetaDefender Cluster system.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Details of all added services and their status.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
service_type	object	
healthy_instances	number	Number of healthy instances for the service.
overall_status	string	Aggregated status across all instances of the service.
overall_status_description	string	Description of the overall status.
instances	array	
service_id	string	Unique service identifier.
message	string	Optional status/message.
display_name	string	Human friendly name. Defaults to "host:port" if absent.

NAME	TYPE	DESCRIPTION
status_description	string	Human readable status explanation.
host	string	Hostname or IP.
port	number	Service's port.
version	string	Service version (semantic or other).
added_by	string	User or system that registered the service.
last_update	number	Unix epoch milliseconds of last update.
last_healthy	number	Unix epoch milliseconds of last confirmed healthy state.
detail	object	
cpu_usage	number	CPU usage (implementation specific units).
platform	string	Operating system/platform the service is running on.
role	string	Service role (e.g. primary, secondary).
db_size	number	Database size in bytes.
ram	object	
total_bytes	number	Total RAM available.
used_bytes	number	RAM currently in use.
disk	object	
total_bytes	number	Total disk space available.
used_bytes	number	Disk space currently in use.

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

6.2 POST /admin/service

Connect and check essential services status.

Establish connections and retrieve the status of essential services within the MetaDefender Cluster system.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
ONE:OF	object	
OPTION:1	object	
host*	string	the host address of the service.
port*	integer	the port number of the service.
connection_key*	string	the connection key for the service.
OPTION:2	object	
host*	string	the host address of the service
port*	integer	the port number of the service
user*	string	the user name for the service.
password*	string	the password for the service.
OPTION:3	object	
host*	string	the host address of the service.
port*	integer	the port number of the service.
user	string	the user name for the service.
password	string	the password for the service.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to add service was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
result	string	the result of the service addition, can be either "ok" or "error"
service_id	string	The unique identifier of the service if result is "ok"
detail	string	The error details if result is "error"

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

6.3 PUT /admin/service/{service_id}

Edit service details.

Update the display name and/or configuration details for a specific service. **Note**: Service configuration cannot be modified after instances have been deployed.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
display_name	string	Display name for the service.
config	object	
host	string	the host address of the service
port	integer	the port number of the service
user	string	the user name for the service. Applicable for type `caching`, `broker`, `datalake`, and `warehouse`
password	string	the password for the service. Applicable for type `caching`, `broker`, `datalake`, and `warehouse`
connection_key	string	the connection key for the service. Applicable for type `filestorage`

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to add service was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
service_id	string	Unique service identifier.
message	string	Optional status/message.
display_name	string	Human friendly name. Defaults to "host:port" if absent.
status_description	string	Human readable status explanation.

NAME	TYPE	DESCRIPTION
host	`string`	Hostname or IP.
port	`number`	Service's port.
version	`string`	Service version (semantic or other).
added_by	`string`	User or system that registered the service.
last_update	`number`	Unix epoch milliseconds of last update.
last_healthy	`number`	Unix epoch milliseconds of last confirmed healthy state.
detail	`object`	
cpu_usage	`number`	CPU usage (implementation specific units).
platform	`string`	Operating system/platform the service is running on.
role	`string`	Service role (e.g. primary, secondary).
db_size	`number`	Database size in bytes.
ram	`object`	
total_bytes	`number`	Total RAM available.
used_bytes	`number`	RAM currently in use.
disk	`object`	
total_bytes	`number`	Total disk space available.
used_bytes	`number`	Disk space currently in use.

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

6.4 DELETE /admin/service/{service_id}

Disconnect to service and remove their configurations.

Remove the connection and configuration details for a specific service. **Note**: Service configuration cannot be deleted after instances have been deployed.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to remove service was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
service_id	string	

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

6.5 GET /admin/service/{service_type}

Get status for a specific service.

Retrieve the current status of a specific service, including all instance details. **Note**: The service_type must be one of: datalake, warehouse, caching, broker, filestorage.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to get service type was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
overall_status	string	Aggregated status for the service type (e.g. healthy, degraded, down)
overall_status_description	string	Human readable description of the aggregated status
healthy_instances	number	Count of instances currently considered healthy
instances	array	
service_id	string	Unique service identifier.
message	string	Optional status/message.
display_name	string	Human friendly name. Defaults to "host:port" if absent.
status_description	string	Human readable status explanation.
host	string	Hostname or IP.
port	number	Service's port.
version	string	Service version (semantic or other).
added_by	string	User or system that registered the service.
last_update	number	Unix epoch milliseconds of last update.
last_healthy	number	Unix epoch milliseconds of last confirmed healthy state.
detail	object	
cpu_usage	number	CPU usage (implementation specific units).
platform	string	Operating system/platform the service is running on.
role	string	Service role (e.g. primary, secondary).

NAME	TYPE	DESCRIPTION
db_size	number	Database size in bytes.
ram	object	
total_bytes	number	Total RAM available.
used_bytes	number	RAM currently in use.
disk	object	
total_bytes	number	Total disk space available.
used_bytes	number	Disk space currently in use.

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
```

```
 "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

6.6 GET /admin/service/{service_type}/setting

Get service settings

Retrieve the current configuration settings for a specific service. **Note**: Supported only for the filestorage service type.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to retrieve service settings was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
max_replica	`number`	Maximum number of replicas, default is 1
min_replica	`number`	Minimum number of replicas, default is 1
cleanuprange	`number`	Cleanup interval in hours, default is 0
storage	`object`	
type	`string`	Storage backend type, can be `salt` or `none`
config	`object`	

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

6.7 PUT /admin/service/{service_type}/setting

Edit setting of service

Update the configuration settings for a specific service. **Note**: Supported only for the filestorage service type.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
max_replica	number	Maximum number of replicas, default is 1
min_replica	number	Minimum number of replicas, default is 1
cleanuprange	number	Cleanup interval in hours, default is 0
storage	object	
type	string	Storage backend type, can be `salt` or `none`
config	object	

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to add service was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
result	string	Success message

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7. STATS

Readiness probing for the MetaDefender Cluster Control Center.

7.1 GET /readyz

Get readiness status

Report whether the MetaDefender Cluster Control Center is ready to accept administrative operations such as uploading installers, deploying instances and managing MetaDefender Cluster Workers. Intended as a readiness probe, so it requires no authentication. Returns 200 when every backing service is healthy, and 503 when at least one is not, meaning those operations may fail or be rejected until it recovers. The response body is empty unless verbose=true is supplied.

REQUEST

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
verbose	boolean	true	Optional. Return the per-component readiness detail instead of an empty body.

RESPONSE

STATUS CODE - 200: The MetaDefender Cluster Control Center is ready; every backing service is healthy.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
status*	boolean	Overall readiness. `true` only when every component below is healthy.
components*	object	
status*	boolean	Aggregate component status, identical to the top-level `status`.
clusterdb*	object	
status*	boolean	Configuration database overall status.

NAME	TYPE	DESCRIPTION
detail*	string	Reason the component is unhealthy. Empty when healthy.
caching*	object	
status*	boolean	Caching service overall status.
detail*	string	Reason the component is unhealthy. Empty when healthy.
broker*	object	
status*	boolean	Message broker overall status.
detail*	string	Reason the component is unhealthy. Empty when healthy.
filestorage*	object	
status*	boolean	MetaDefender Cluster File Storage overall status.
detail*	string	Reason the component is unhealthy. Empty when healthy.
datalake*	object	
status*	boolean	Data lake overall status.
detail*	string	Reason the component is unhealthy. Empty when healthy.
datawarehouse*	object	
status*	boolean	Data warehouse overall status.
detail*	string	Reason the component is unhealthy. Empty when healthy.
identity*	object	
status*	boolean	MetaDefender Cluster Identity Service overall status.
detail*	string	Reason the component is unhealthy. Empty when healthy.

EXAMPLE:


```
{
 "components": {
  "clusterdb": {
   "status": true,
   "detail": ""
  },
  "caching": {
   "status": true,
   "detail": ""
  },
  "broker": {
   "status": true,
   "detail": ""
  },
  "filestorage": {
   "status": true,
   "detail": ""
  },
  "datalake": {
   "status": true,
   "detail": ""
```

```
    },
    "datawarehouse": {
     "status": true,
     "detail": ""
    },
    "identity": {
     "status": true,
     "detail": ""
    },
    "status": true
   },
   "status": true
  }
```


STATUS CODE - 503: The MetaDefender Cluster Control Center is not ready; at least one backing service is unhealthy. Administrative operations may fail until it recovers.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
status*	boolean	Overall readiness. `true` only when every component below is healthy.
components*	object	
status*	boolean	Aggregate component status, identical to the top-level `status`.
clusterdb*	object	
status*	boolean	Configuration database overall status.
detail*	string	Reason the component is unhealthy. Empty when healthy.
caching*	object	
status*	boolean	Caching service overall status.
detail*	string	Reason the component is unhealthy. Empty when healthy.
broker*	object	
status*	boolean	Message broker overall status.
detail*	string	Reason the component is unhealthy. Empty when healthy.
filestorage*	object	
status*	boolean	MetaDefender Cluster File Storage overall status.
detail*	string	Reason the component is unhealthy. Empty when healthy.
datalake*	object	
status*	boolean	Data lake overall status.
detail*	string	Reason the component is unhealthy. Empty when healthy.
datawarehouse*	object	
status*	boolean	Data warehouse overall status.
detail*	string	Reason the component is unhealthy. Empty when healthy.
identity*	object	

NAME	TYPE	DESCRIPTION
status*	boolean	MetaDefender Cluster Identity Service overall status.
detail*	string	Reason the component is unhealthy. Empty when healthy.

EXAMPLE:


```
{
 "components": {
  "clusterdb": {
   "status": true,
   "detail": ""
  },
  "caching": {
   "status": false,
   "detail": "service manager is not available"
  },
  "broker": {
   "status": true,
   "detail": ""
  },
  "filestorage": {
   "status": true,
   "detail": ""
  },
  "datalake": {
   "status": true,
   "detail": ""
  },
  "datawarehouse": {
   "status": true,
   "detail": ""
  },
  "identity": {
   "status": true,
   "detail": ""
  },
  "status": false
 },
 "status": false
}
```

8. WORKERS

Connect, deploy, undeploy and manage workers.

8.1 GET /admin/worker

List connected workers

Retrieve a list of currently connected MetaDefender Cluster Worker services.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: A list of connected workers.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
worker_id	string	Unique identifier of the worker.
display_name	string	Display name for the worker.
platform	string	Operating system / platform of the worker.
os	string	Operating system details of the worker.
package_type	string	The deployment package type.
hardware	object	
cpu	object	
count	integer	Number of CPU cores.

NAME	TYPE	DESCRIPTION
model	`string`	CPU model name.
usage	`number`	CPU usage percentage.
disk	`object`	
available_bytes	`integer`	Available disk space in bytes.
total_bytes	`integer`	Total disk space in bytes.
memory	`object`	
available_bytes	`integer`	Available memory in bytes.
total_bytes	`integer`	Total memory in bytes.
user_name	`string`	Name of the user who added the worker.
host	`string`	The address (IP or hostname) of the worker.
port	`integer`	Port on which the worker is listening.
status	`string`	Current status of worker.
status_description	`string`	The description of worker's status
version	`string`	The version of the worker.
deployment_info	`object`	
type	`enum`	**ALLOWED:** `ometascan, api-gateway, callback-service, download-service` Deployment type.
installer_id	`string`	Identifier of the installer.
version	`string`	The instance version.
deployer_name	`string`	Name of the user who deployed the instance.
custom_config	`object`	

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

8.2 POST /admin/worker

Connect to workers

Connect to MetaDefender Cluster Worker services.

REQUEST

REQUEST BODY - application/json

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to add service was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
display_name*	string	Display name for the worker.
host*	string	The address of the worker.
port*	number	The port on which the worker is listening.
result*	enum	ALLOWED: ok, failed Connection attempt result.
worker_id	string	Present only when result = ok. Unique identifier of the worker.
error	string	Present only when result = failed. Error message.

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

8.3 DELETE /admin/worker

Disconnect from workers

Disconnect from specified MetaDefender Cluster Worker services.

REQUEST

REQUEST BODY - application/json

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to disconnect workers was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
worker_id	enum	ALLOWED: Deleted Disconnection status of the worker.

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

8.4 GET /admin/worker/instance/deploy/available/{installer_id}

Get available workers by installer_id.

Retrieve the list of available workers eligible for deployment for the specified installer ID.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to get available workers was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
worker_id	string	Unique identifier of the worker.
display_name	string	Display name for the worker.
platform	string	Operating system / platform of the worker.
os	string	Operating system details of the worker.
package_type	string	The deployment package type.
hardware	object	
cpu	object	
count	integer	Number of CPU cores.
model	string	CPU model name.
usage	number	CPU usage percentage.
disk	object	
available_bytes	integer	Available disk space in bytes.
total_bytes	integer	Total disk space in bytes.
memory	object	
available_bytes	integer	Available memory in bytes.
total_bytes	integer	Total memory in bytes.
user_name	string	Name of the user who added the worker.
host	string	The address (IP or hostname) of the worker.
port	integer	Port on which the worker is listening.
status	string	Current status of worker.
status_description	string	The description of worker's status
version	string	The version of the worker.
deployment_info	object	
type	enum	**ALLOWED:** `ometascan, api-gateway, callback-service, download-service` Deployment type.
installer_id	string	Identifier of the installer.
version	string	The instance version.
deployer_name	string	Name of the user who deployed the instance.
custom_config	object	

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

8.5 POST /admin/worker/instance/deploy

Deploy workers

Deploy the selected installer on one or more selected workers.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
ONE:OF	`object`	
OPTION:1	`object`	
type*	`enum`	**ALLOWED:** `ometascan`
installer_id*	`string`	Identifier of the installer.
worker*	`array`	
config	`object`	
log_level	`enum`	**DEFAULT:**`info` **ALLOWED:** `debug, info, warning, error`
connection_per_file_service	`integer`	`>=1` **DEFAULT:**`4`
install_dir	`string`	Optional. Custom install directory on the worker. It must already exist.
OPTION:2	`object`	
type*	`enum`	**ALLOWED:** `api-gateway`
installer_id*	`string`	Identifier of the installer.
worker*	`array`	
cert	`string`	

NAME	TYPE	DESCRIPTION
		Certificate name (default empty). Only for api-gateway.
config	object	
port	integer	between 1 and 65535 DEFAULT:8899
log_level	enum	DEFAULT:info ALLOWED: debug, info, warning, error
install_dir	string	Optional. Custom install directory on the worker. It must already exist.
OPTION:3	object	
type*	enum	ALLOWED: callback-service
installer_id*	string	Identifier of the installer.
worker*	array	
config	object	
log_level	enum	DEFAULT:info ALLOWED: debug, info, warning, error
install_dir	string	Optional. Custom install directory on the worker. It must already exist.
OPTION:4	object	
type*	enum	ALLOWED: download-service
installer_id*	string	Identifier of the installer.
worker*	array	
config	object	
port	integer	between 1 and 65535 DEFAULT:8895
log_level	enum	DEFAULT:info ALLOWED: debug, info, warning, error
install_dir	string	Optional. Custom install directory on the worker. It must already exist.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to add service was successful

RESPONSE MODEL - application/json

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or

out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

8.6 DELETE /admin/worker/instance/deploy

Undeploy workers

Undeploy the specified workers.

REQUEST

REQUEST BODY - application/json

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to undeploy workers was successful

RESPONSE MODEL - application/json

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

8.7 POST /admin/worker/instance/upgrade

Upgrade deployed instances

Upgrade the deployed instances managed by the worker to a newer version

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
version*	`string`	Target version to upgrade to.
type*	`enum`	**ALLOWED:** `ometascan, api-gateway, callback-service, download-service` Worker deployment type.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	`string`		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to upgrade workers was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
result	`string`	

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

8.8 GET /admin/worker/instance/upgrade/version

Get upgradable instance version

Retrieve a list of available versions of MetaDefender Core, MetaDefender Cluster API Gateway, Callback Service and Download Service for upgrading.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: A list of available versions for upgrading.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ometascan*	array	
api-gateway*	array	
callback-service*	array	
download-service*	array	

STATUS CODE - 400: Bad Request (e.g. header is invalid, apikey is missing or invalid, parameter value is invalid or out of range, etc).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```