Cloudera Migration Assistant 4.0.0

Administrator guide for Cloudera Migration Assistant

Date published: 2026-06-18
Date modified: 2026-06-18

https://docs.cloudera.com/

Legal Notice

Contents

Cloudera Migration Assistant deployment

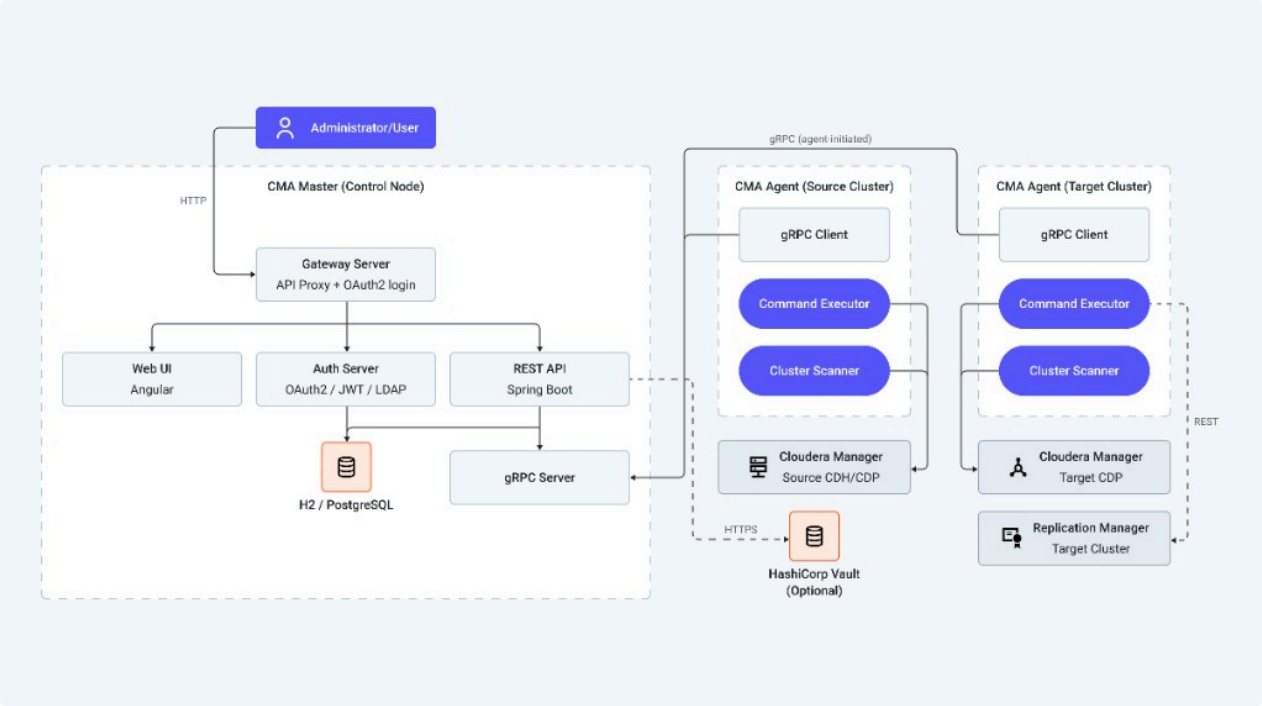
Cloudera Migration Assistant uses a Master-Agent architecture to orchestrate migrations across distributed clusters with gateway, REST, authentication, metadata services, and optional Vault integration.

Architecture overview

Cloudera Migration Assistant uses a CMA Master-Agent model: a central CMA Master orchestrates migrations while CMA Agents on source and target clusters execute scans and replication commands.

Architecture diagram

Figure 1: Cloudera Migration Assistant architecture



CMA Master

The **CMA Master** is the central control node deployed on a dedicated host or on a cluster node through Cloudera Manager.

Table 1: CMA Master components

Component	Description
Gateway Server	Entry point for routing API requests to Master, directing authentication to the Auth Server, and handling OAuth2 login and session management.
Web UI	Angular-based interface for managing clusters, creating migration plans, and monitoring execution.
REST API	Spring Boot endpoints for governing all migration operations.
Auth Server	OAuth2/JWT-based authentication service with LDAP integration and role-based access control (RBAC).

Component	Description
Database	Repository for storing cluster metadata, migration plans, scan results, credentials, and execution history. H2 (embedded) for evaluation, PostgreSQL for production.
gRPC Server	Communication framework for handling bidirectional streaming communication with CMA Agents.

CMA Master has the following key responsibilities:

- Provides the administrative web interface.
- Manages cluster registration (source and target).
- Stores and displays scan results.
- Orchestrates migration execution plans.
- Dispatches commands to agents and tracks execution status.
- Manages user authentication and authorization.
- Optionally stores secrets in HashiCorp Vault.

CMA Agent

The **CMA Agent** runs on or near the source or target clusters and runs commands locally. It is deployed as a Cloudera Manager add-on service.

Table 2: CMA Agent components

Component	Description
gRPC Client	Maintains persistent outbound connection to the Master
Command Executor	Runs scanning and migration commands, and Ansible playbooks
Cluster Scanner	Scans cluster resources (HDFS, Hive, Ozone, Ranger)
Cluster Service	Interfaces with Cloudera Manager APIs

CMA Agent has the following key responsibilities:

- Connects to and registers with the CMA Master (outbound connection)
- Runs commands received from the Master (scanning, migration, data replication)
- Uploads scan results to the Master
- Reports command execution status
- Interfaces directly with cluster services through Cloudera Manager

Cloudera Replication Manager

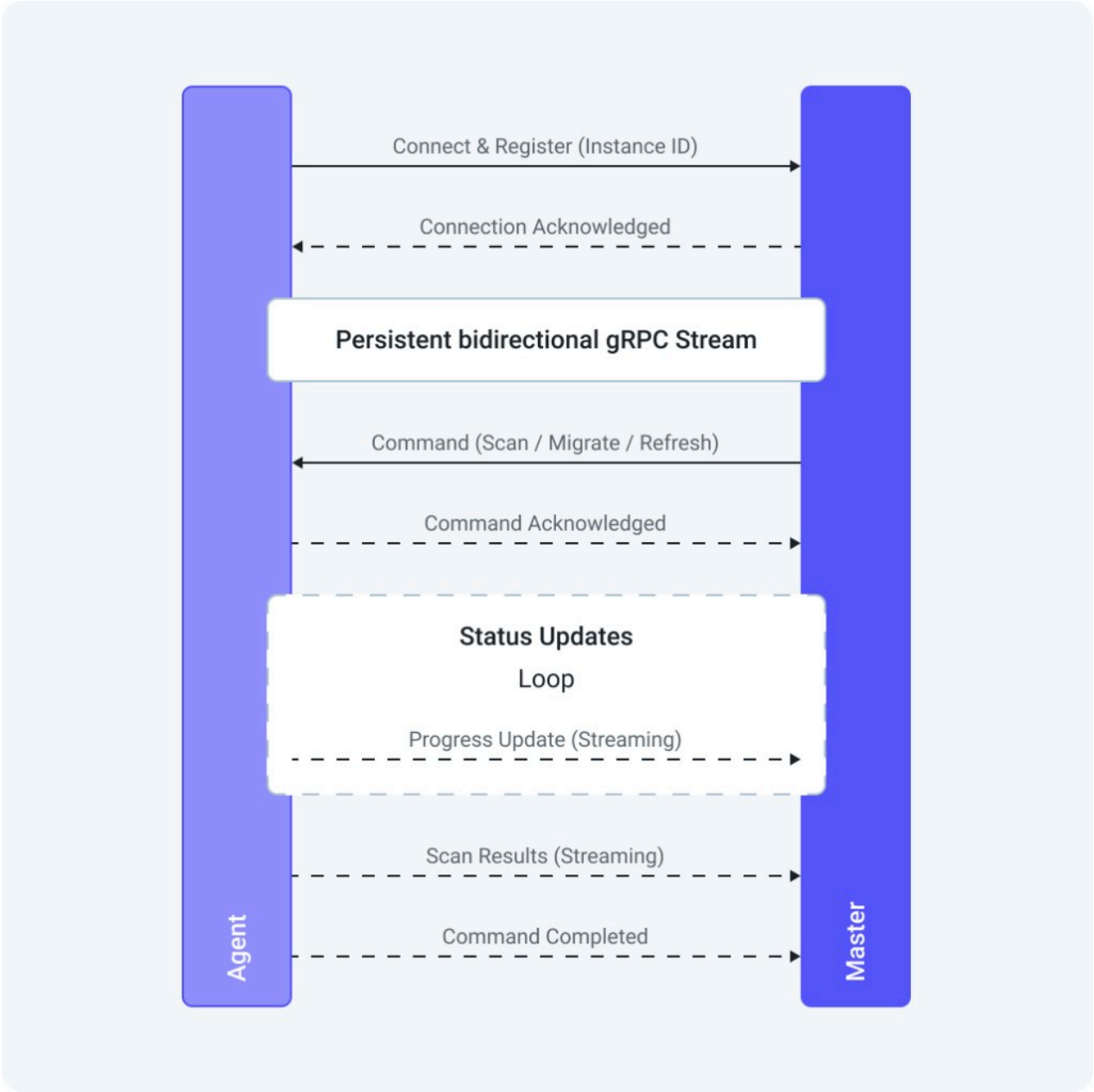
For Cloudera Base on premises to Cloudera Base on premises migrations, CMA requires a **Cloudera Replication Manager** (Cloudera Replication Manager) service deployed through Cloudera Manager parcel on the target cluster. The CMA Agent on the target cluster communicates with Cloudera Replication Manager over REST to create and manage replication policies for data migration (HDFS, HDFS to Ozone).

Cloudera Replication Manager must be installed on the **same node as Cloudera Manager** on the target cluster. It is deployed as a separate parcel alongside the CMA parcel. For HDFS to Ozone migrations, the target cluster must run **Cloudera Manager 7.13.2** or higher versions.

Note: Cloudera Replication Manager is required for Cloudera Base on premises to Cloudera Base on premises migrations, including HDFS to Ozone. For detailed setup and integration information, see *Cloudera Replication Manager Integration*.

Communication model

Figure 2: Communication between CMA Master and CMA Agent



The communication model for CMA Master and CMA Agent has the following key characteristics:

- **Agent-initiated** – Agents establish outbound connections to the Master to simplify firewall rules
- **Persistent** – Connection remains active and automatically reconnects on failure
- **Bidirectional** – Commands flow from the Master to the Agent, while results and status flow from the Agent to the Master
- **Concurrent** – Multiple agents can connect simultaneously

gRPC services

Table 3: gRPC services and their purpose

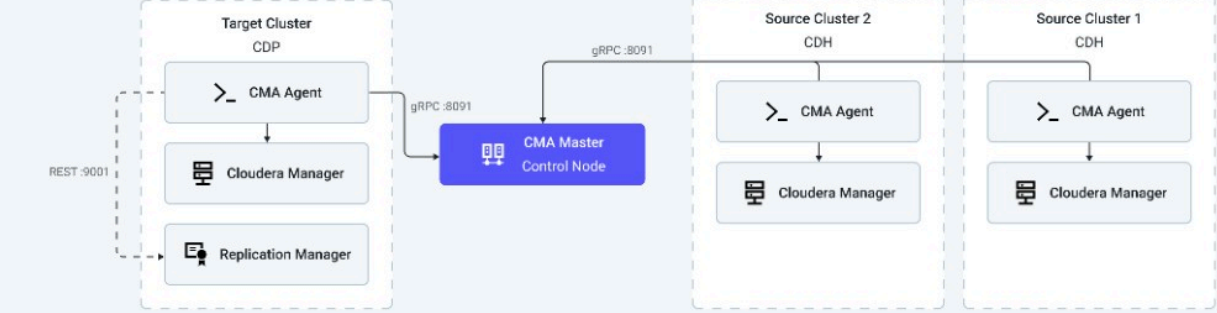
Service	Purpose
CommandService	Bidirectional streaming for command dispatch and acknowledgment
CommandStateService	Streaming for command status updates

Service	Purpose
ScanService	Streaming for scan result uploads
CmaInstanceService	Agent registration and instance management
ManagedClusterService	Cluster metadata synchronization

Distributed deployment

The CMA Master runs on a control node while CMA Agents deploy on each cluster through Cloudera Manager parcels. This is the recommended production topology. For installation procedures, see *Deploying the CMA server with parcels*.

Figure 3: Distributed deployment architecture



Distributed deployment provides the following benefits:

- Scalability for multiple clusters
- Reduced network latency for local operations
- Isolated execution environments
- Parallel scanning and migration across clusters

Cluster registration

In distributed deployments, cluster registration proceeds in the following order:

1. The administrator deploys CMA Agent service on the cluster through Cloudera Manager
2. The administrator configures the required properties using the Table 4: Required CMA Agent configuration properties in Cloudera Manager on page 8 and Table 5: Optional CMA Agent configuration properties on page 8tables.
3. The administrator starts the CMA Agent service.
4. The Cloudera Migration Assistant Agent automatically connects to the Cloudera Migration Assistant Master over gRPC.
5. The Cloudera Migration Assistant Agent registers the managed cluster with the Cloudera Migration Assistant Master.

6. Cluster is displayed in CMA Master UI ready for scanning.

Table 4: Required CMA Agent configuration properties in Cloudera Manager

Property	Description
cma_gateway_url	URL of the CMA Gateway, for example, https://master-host:8093. **Required when the CMA Agent is on a different cluster than the CMA Master.** The agent uses this URL to discover the Master host, gRPC port, and Auth Server URI automatically through the discovery endpoint (GET /api/agent/config). When connecting from a different cluster, set the **CMA Master** service dependency to none in Cloudera Manager.
cm_username	Cloudera Manager administrator username
cm_password	Cloudera Manager administrator password
cma_user_home	Home directory for the cma-agent user, for example, /home/cma-agent
cma_agent_client_id	OAuth2 client ID for agent-to-master authentication. Must match the value configured on the CMA Master. The default value is cma-agent.
cma_agent_client_secret	OAuth2 client secret for agent-to-master authentication. Must match the value configured on the CMA Master.

Finding the Gateway URL

In Cloudera Manager on the cluster where CMA Master is installed, open the CMA Master service and click the CMA UI link. The URL shown in your browser (up to the port number) is the Gateway URL. The format is https://<master-host>:8093 for HTTPS or http://<master-host>:8090 for HTTP. You can verify connectivity from the agent host by calling the following discovery endpoint:

```
curl -sk https://<master-host>:8093/api/agent/config
```

The response returns the Auth Server URI and gRPC endpoint that the agent will use.

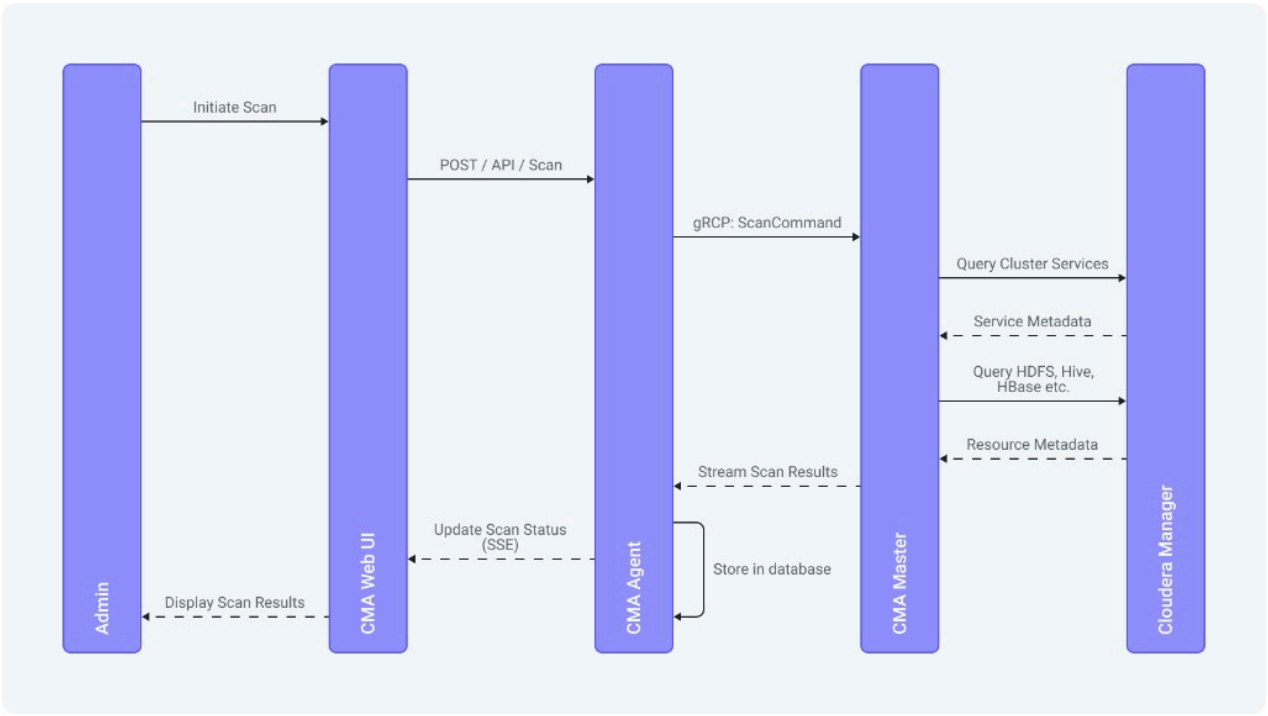
Note: The gateway URL is the only connection parameter required for deployment. The CMA Agent discovers all other system coordinates automatically. You can manually override individual values using cma_master_url (gRPC endpoint), --master-host, --grpc-port, or --auth-server-uri if automatic discovery is not suitable for your environment.

Table 5: Optional CMA Agent configuration properties

Property	Default	Description
cma_java_home	Auto-detected	JDK 17+ home directory
cma_python3_executable	python3	Python 3.11 executable path
cma_agent_http_port	9090	Agent HTTP port
cma_agent_https_port	9093	Agent HTTPS port
kerberos.auth.enable	false	Kerberos authentication flag
external_vault_url	-	External Vault URL (if not using CM-managed Vault)

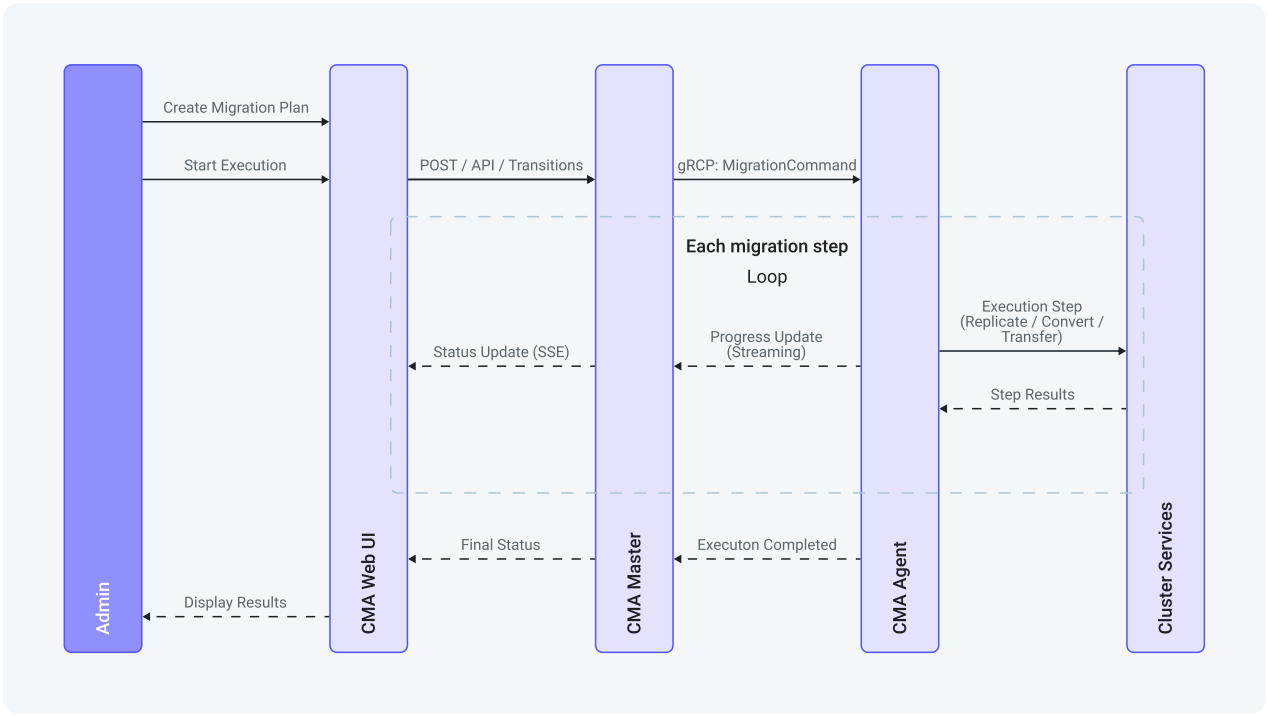
Scanning flow

Figure 4: Workload scanning end-to-end sequence flow



Migration flow

Figure 5: Migration plan execution sequence flow



Service ports

Table 6: HTTP and HTTP service ports

Service	HTTP port	HTTPS port	Purpose
Gateway Server	8090	8093	Web UI and REST API (entry point for users)
Master Server	8890	8893	Internal REST API
Master gRPC	8091	8891	Agent-to-Master communication (plaintext or TLS on the same port, depending on the configuration)
Auth Server	9000	9443	Authentication and token issuance
Agent Server	9090	9093	Agent local REST API
Cloudera Replication Manager	9011	-	Data replication policies (from Cloudera Base on premises to Base)

Agent network requirements

Table 7: Network requirements for the CMA Agent

Requirement	Description
Outbound CMA Master connectivity	gRPC connection to the CMA Master on port 8091
Cloudera Manager API access	HTTP or HTTPS connection to the Cloudera Manager API, typically port 7180 or 7183
SSH access	SSH connection to cluster nodes for Ansible-based operations
Cluster service access	Network access to HDFS, Hive Metastore, HBase, and Ozone

Firewall considerations

- Cloudera Migration Assistant Agents initiate all connections to the Master (outbound from agent network).
- Master does **not** need to initiate connections to agents.
- Only the Gateway Server port (8090 or 8093) is required to be exposed to administrators.
- Internal ports for the Master or the Auth Server can be restricted to the CMA host.
- All traffic is encrypted when TLS is enabled. Cloudera recommends enabling TLS for production use.

Authentication

- **User Authentication** – OAuth2/JWT through CMA Auth Server, with optional LDAP or Active Directory integration.
- **Agent Communication** – gRPC with TLS encryption. Agents authenticate to the Master using OAuth2 client credentials (cma_agent_client_id and cma_agent_client_secret). These must match on both the Master and Agent services. Cloudera recommends TLS encryption for production use. TLS is auto-bootstrapped in parcel deployments.
- **Internal Service Authentication** – The Gateway Server authenticates to the Auth Server using OAuth2 client credentials (cma_client_id and cma_client_secret), configured on the CMA Master.
- **Cluster Authentication** – Cloudera Manager credentials stored in the database or HashiCorp Vault.

Authorization

CMA implements role-based access control (RBAC).

Table 8: RBAC roles and permissions

Role	Permissions
ROLE_CMA_ADMIN	Full access to all features
ROLE_CMA_OPERATOR	• Full access on CMA Server • Read access on CMA Auth Server
ROLE_CMA_MIGRATOR	• Modify access on CMA Server • Read access on CMA Auth Server
ROLE_CMA_VIEWER	Read-only access

Credential management

- Credentials can be stored encrypted in the database or in **HashiCorp Vault**. Cloudera recommends using HashiCorp Vaults for production use.
- Vault integration uses KV v2 secret engine with short-lived access tokens.
- SSH keys used for Ansible-based migration operations.
- Cloudera Manager API credentials stored per-cluster.

Master HA

- Database can be externalized to PostgreSQL for durability and backup.
- Multiple Master instances are not currently supported (roadmap).
- Cloudera recommends performing regular backup of database and configuration.

Agent HA

- Agents automatically reconnect on connection loss.
- Commands are idempotent whenever possible.
- Status tracking allows resuming interrupted operations.

Troubleshooting

For common issues (agent connectivity, port conflicts, slow startup) and log file locations, see *Troubleshooting*.

Related Information

Deploying the server with parcels
Supported platforms and migration paths
Role-based Access Control
Enabling TLS/SSL for
Storing secrets in Vault
Scanning clusters

Deploying the CMA server with parcels

Install Cloudera Migration Assistant CSDs and parcels through Cloudera Manager, add CMA_MASTER and CMA_AGENT services on the right hosts, and configure OAuth and Gateway discovery for production clusters.

About this task

CMA is deployed as a parcel-based add-on service in Cloudera Manager and requires 1.5 GB of extra memory on the host. For architecture and network requirements, see *Cloudera Migration Assistant deployment*.

Dependencies

Ensure these components are installed on the CMA host:

- Python 3.11
- JDK 17+ with JAVA_HOME set

CMA automatically downloads required Python packages (Ansible, cryptography, database drivers, and others) at startup. In airgapped environments, pre-bundle packages in the CMA extras tarball.

Custom PyPI repository

By default, CMA downloads Python packages from the public PyPI index. In corporate environments with internal PyPI mirrors, set PyPI Index URL (cma_pip_index_url) in the CMA Master or CMA Agent service configuration — for example https://nexus.corp.com/repository/pypi/simple/. When a custom index URL is configured, CMA passes --trusted-host to pip automatically.

Note: The --trusted-host flag disables SSL certificate verification for pip downloads. If your internal PyPI mirror has a valid certificate from a trusted CA, configure the CA in the system trust store instead.

If JDK 17+ is not installed, download and install it on the host before deploying the service. When Java is on a default path (/usr/java/jdk-17, /usr/lib/jvm/java-17, or similar), you can omit Cloudera Migration Assistant Java Home.

For general instructions on adding an add-on service, see Extending Cloudera Manager — Add-on Services.

Download URLs

Cloudera Migration Assistant CSDs and parcels are published at https://archive.cloudera.com/cma/*VERSION*/, where *VERSION* is the Cloudera Migration Assistant release version (for example, 4.0.0.0).

To find the exact file names for your version, browse the archive:

- CSDs: https://archive.cloudera.com/cma/*VERSION*/csd/
- Parcels: https://archive.cloudera.com/cma/*VERSION*/parcels/

Table 9:

File	Description
CMA_MASTER-*VERSION-BUILD*.jar	CSD for the CMA Master service
CMA_AGENT-*VERSION-BUILD*.jar	CSD for the CMA Agent service

Note: Both CSD files are required on the Cloudera Manager Server host, even if you deploy only one service type on a given cluster.

Note: The CMA Agent defaults to HTTP port 9090 and HTTPS port 9093. Port 9090 commonly conflicts with HBase Thrift Server on the same host. If the agent fails to start with Port 9090 was already in use, change cma_agent_http_port and cma_agent_https_port in the agent service configuration. For details, see *Agent port conflict (port 9090 already in use)*.

Procedure

1. Download both Cloudera Migration Assistant CSD files to the /opt/cloudera/csd/ directory on the Cloudera Manager Server host.

   ```
   wget -P /opt/cloudera/csd/ https://archive.cloudera.com/cma/VERSION/csd/
   CMA_MASTER-VERSION-BUILD.jar
   ```

   ```
   wget -P /opt/cloudera/csd/ https://archive.cloudera.com/cma/VERSION/csd/
   CMA_AGENT-VERSION-BUILD.jar
   ```

   Cloudera Manager automatically detects the CSD files.

2. Change the ownership of the CSD files.

```
chown cloudera-scm:cloudera-scm /opt/cloudera/csd/CMA_MA
STER-VERSION-BUILD.jar /opt/cloudera/csd/CMA_AGENT-VERSION-BUILD.jar
```

3. Restart Cloudera Manager for the changes to take effect.

```
systemctl restart cloudera-scm-server
```

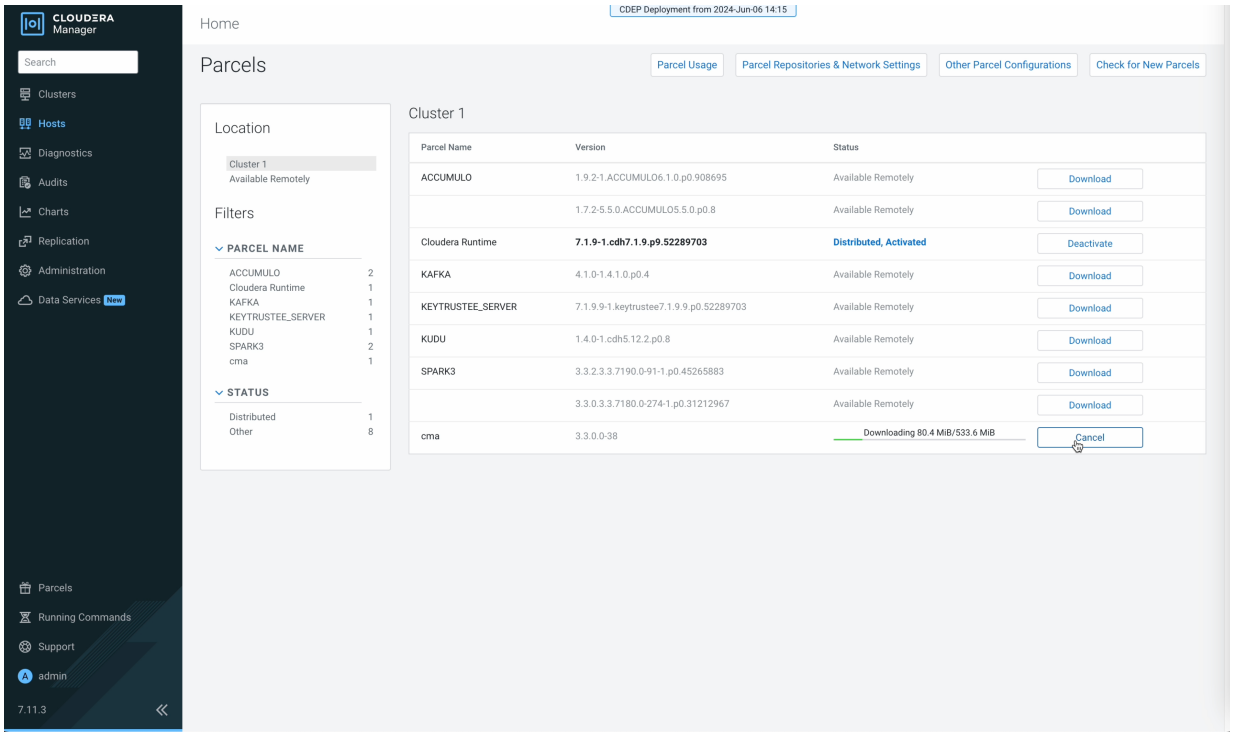
4. Log into Cloudera Manager.
5. Restart the Cloudera Management Service.
6. Go to Hosts Parcels .
7. Click Parcel Repositories & Network Settings.
8. Add the Remote Parcel Repository URL for Cloudera Migration Assistant.

```
https://archive.cloudera.com/cma/VERSION/parcels/
```

Note: Make sure that the Remote Parcel Repository URL uses an HTTPS link. To install a different version of the parcel, change the URL as needed.

9. Click Save & Verify Configuration to commit the change.
10. Click Close.
    You are redirected to the Parcels page.
11. Search for CMA, and click Download to download the parcel to the local repository.
12. After download completes, click Distribute to distribute the parcel to all clusters.

Figure 6: Distribute the CMA parcel

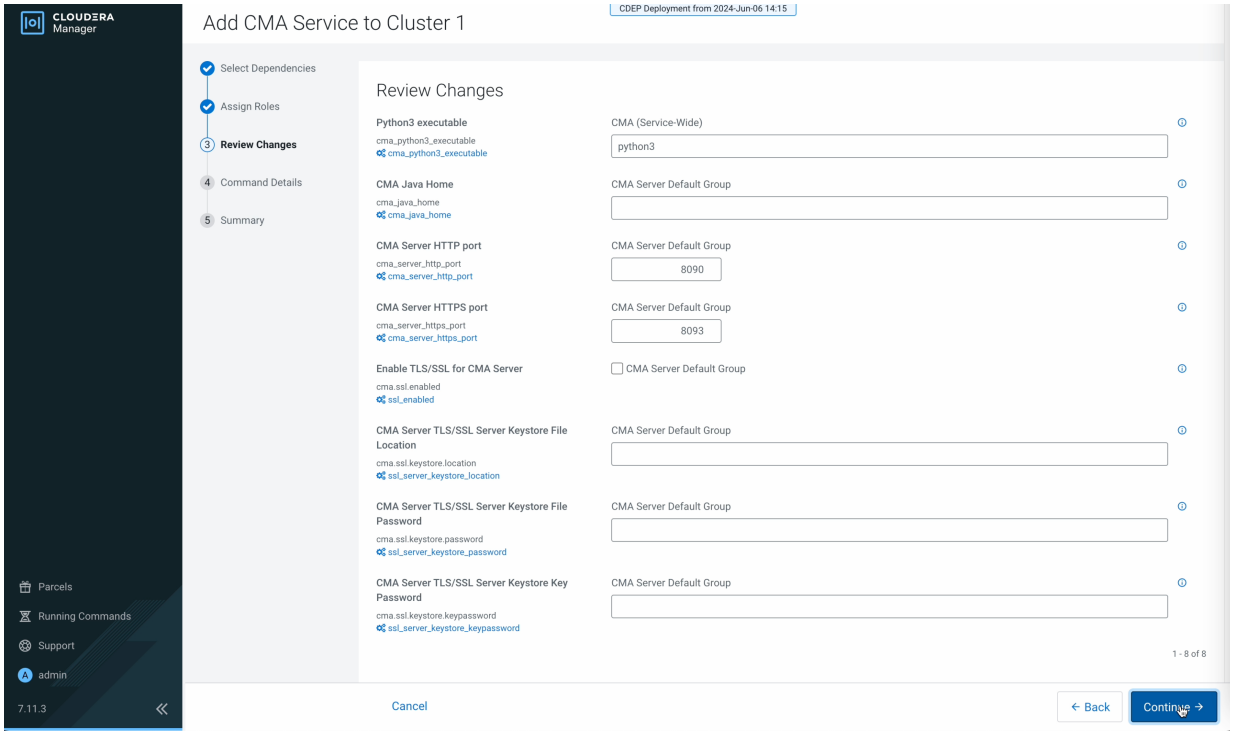


13. Click Activate to activate the parcel.
14. Click OK when confirmation is required.
15. Click Clusters in the left navigation pane.
16. Select Add Service from the drop-down menu to the right of your cluster.

17. From the list, select the CMA service type to add, then click Continue. The single CMA parcel provides two service types: CMA_MASTER (central control node) and CMA_AGENT (cluster-side executor).

    The **Add Service wizard** opens.

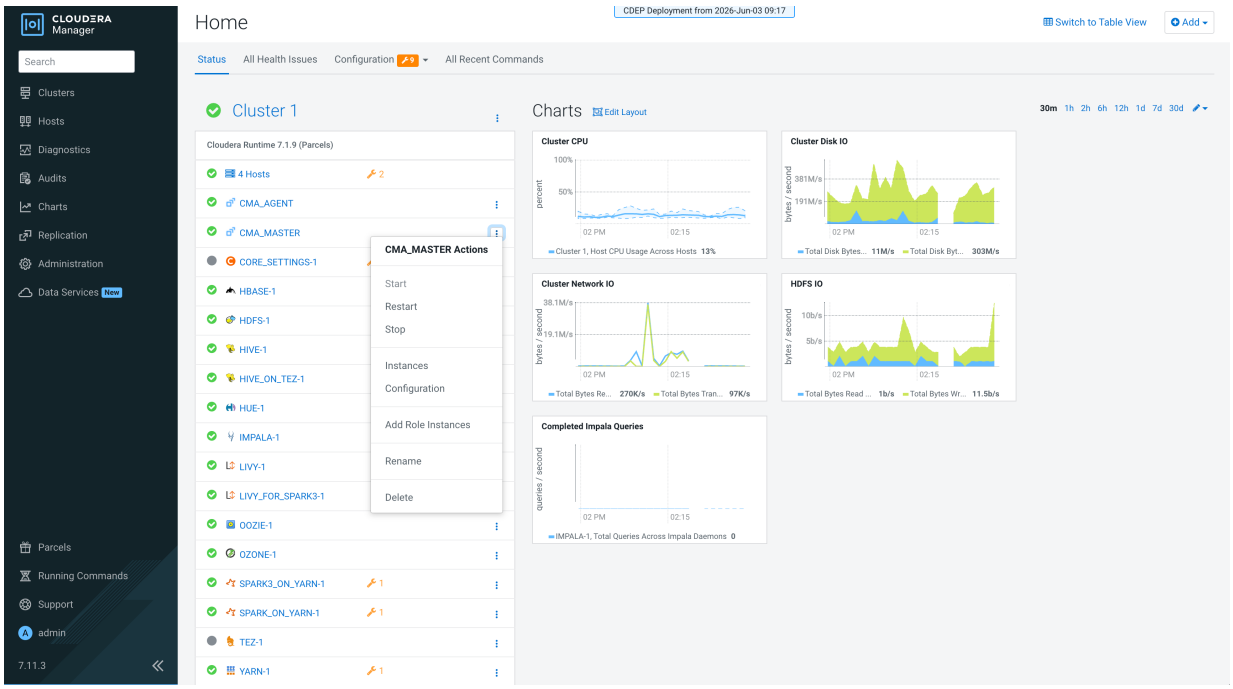
    Figure 7: Add CMA service



18. Assign the service roles to the hosts where Python 3.11 and JDK 17+ are installed, and click Continue.
19. Review service configurations and click Continue. At minimum, set OAuth2 client credentials (cma_client_secret, cma_agent_client_secret). If the CMA Agent is on a different cluster from the CMA Master, also set Gateway URL (cma_gateway_url).

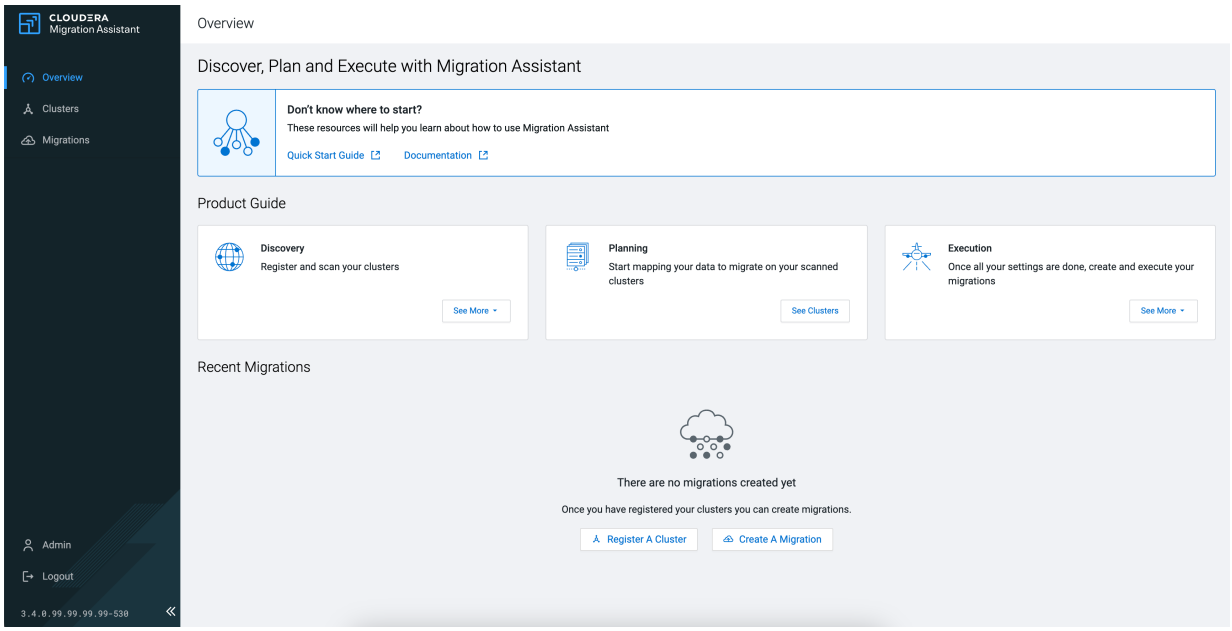
    The first run of the service starts. When the command finishes, the service is added to the cluster.

    Figure 8: CMA service started



20. Repeat steps 16–19 to add the other CMA service type if needed (for example, add CMA_AGENT after CMA_MASTER, or vice versa).
21. Go back to the cluster homepage, open the CMA Master service page, and click the Server UI tab to open the CMA UI.

Figure 9: CMA landing page



22. Set the **CMA Master** service dependency to none in the CMA Agent service configuration in Cloudera Manager because Cloudera Manager cannot discover the Master automatically across clusters.
23. Set the **Gateway URL** to cma_gateway_url in the CMA Agent service configuration.

```
curl -sk https://<master-host>:8093/api/agent/config
```

Results

Table 10:

Service	Description
CMA_MASTER	The central control node, including the Gateway Server, Auth Server, Master Server roles. Deploy once, typically on a dedicated host or on the target cluster.
CMA_AGENT	The cluster-side executor, including the Agent Server role. Deploy on each source and target cluster that participates in the migration.

You can deploy both services on the same cluster if needed (for example, the target cluster runs both CMA Master and CMA Agent).

Clusters are registered automatically when CMA Agents are deployed and started. For more information, see *Registering Clusters*.

Agent configuration: When the CMA Agent is deployed on a **different cluster** from the CMA Master, you must perform the following tasks

The agent uses the Gateway URL to automatically discover the Master host, gRPC port, and Auth Server URI using the discovery endpoint (GET /api/agent/config).

To find the Gateway URL: In Cloudera Manager on the Master's cluster, open the **CMA Master** service and click the **CMA UI** link. The URL in your browser (up to the port) is the Gateway URL — typically https://<master-host >:8093 (HTTPS) or http://<master-host>:8090 (HTTP). You can verify it from the agent host with:

You can set up Vault to store secrets. For instructions, see *Storing secrets in Vault*.

By default, CMA uses an embedded H2 database. A database password is mandatory for all database types, including H2. Cloudera Manager generates credential files (db.properties and auth-db.properties) from the CSD service descriptor at deployment.

For production deployments, CMA supports **PostgreSQL** as an external database. For setup instructions, see *Configuring CMA with PostgreSQL*.

CMA uses the following sets of OAuth2 client credentials for trusted communication between its components:

- **Master credentials** (cma_client_id / cma_client_secret) — Used by the Gateway Server to authenticate with the Auth Server. Configured on the CMA Master service.
- **Agent credentials** (cma_agent_client_id / cma_agent_client_secret) — Used by CMA Agents to authenticate to the Master's gRPC server. These values must match on both the CMA Master and every CMA Agent.

Table 11:

Parameter	Default	Description
cma_client_id	cma	OAuth2 client ID for Gateway # Auth Server
cma_client_secret	—	OAuth2 client secret for Gateway # Auth Server (**required**)
cma_agent_client_id	cma-agent	OAuth2 client ID for Agent # Master gRPC
cma_agent_client_secret	—	OAuth2 client secret for Agent # Master gRPC

Set OAuth credentials in Cloudera Manager under the CMA Master and CMA Agent service configurations. The CSD generates credential files (master-auth.properties, agent-auth.properties) automatically.

Related Information

Registering clusters

Configuring with PostgreSQL

Storing secrets in Vault

Supported platforms and migration paths

Cloudera Migration Assistant 4.0 supports CDP Private Cloud Base to CDP Private Cloud Base migrations and related Cloudera Data Engineering and Git workflows.

Cloudera Migration Assistant 4.0

Cloudera Migration Assistant 4.0 supports **CDP Private Cloud Base to CDP Private Cloud Base** migrations. Public cloud migration support was removed and will be reintroduced in a future release with the new architecture.

Supported migration paths

Migration type	Source	Target	Requirements
HDFS to HDFS	CDP Base 7.1.7+	CDP Base 7.1.7+	Cloudera Replication Manager on target cluster
HDFS to Ozone	CDP Base 7.1.7+	CDP Base 7.1.9+	Cloudera Replication Manager on target cluster, Cloudera Manager 7.13.2+
Cloudera Data Engineering to Cloudera Workflow Orchestrator Git	Cloudera Data Engineering virtual cluster (CDP Public/Private Cloud)	Git repository (GitHub, GitLab, or plain Git)	Cloudera Data Engineering Agent with CDP API credentials, Git access token. Technical preview
Ranger policy discovery	CDP Base with Ranger	—	Technical preview; scan and browse only

Supported source platforms

Platform	Versions
Cloudera Data Platform (CDP) Private Cloud Base	7.1.7, 7.1.8, 7.1.9, 7.1.9 SP1
Cloudera Data Engineering (Cloudera Data Engineering)	Via CDP Control Plane API

Supported target platforms

Platform	Versions
CDP Private Cloud Base	7.1.7, 7.1.8, 7.1.9, 7.1.9 SP1
Git repositories	GitHub, GitHub Enterprise, GitLab, plain Git

Infrastructure requirements

Component	Requirement
Cloudera Manager (target)	7.13.1+ (7.13.2+ for HDFS to Ozone)
JDK	17+
Python	3.11
Operating system	RHEL/CentOS 7.x, 8.x, and 9.x
Browser	Chrome, Firefox (latest)

Cloudera Migration Assistant 3.5 and earlier

Earlier Cloudera Migration Assistant releases supported additional migration paths. See the product archive for release-specific documentation:

- Cloudera Migration Assistant 3.5.0 — HDP upgrades are not supported in Cloudera Migration Assistant 3.5; Cloudera Base 7.1.7|8|9 to Cloudera on cloud 7.2.18 migration (AWS, Azure)
- Cloudera Migration Assistant 3.4.0
- Cloudera Migration Assistant 3.3.0
- Cloudera Migration Assistant 3.2.0
- Cloudera Migration Assistant 3.0.0
- Cloudera Migration Assistant 2.x — HDP to CDP upgrades (various paths). See release notes for details.

CMA Role-based Access Control

CMA role-based access control (RBAC) manages user permissions within CMA components through the CMA Auth Server, which handles authentication and authorization using JSON Web Tokens (JWT).

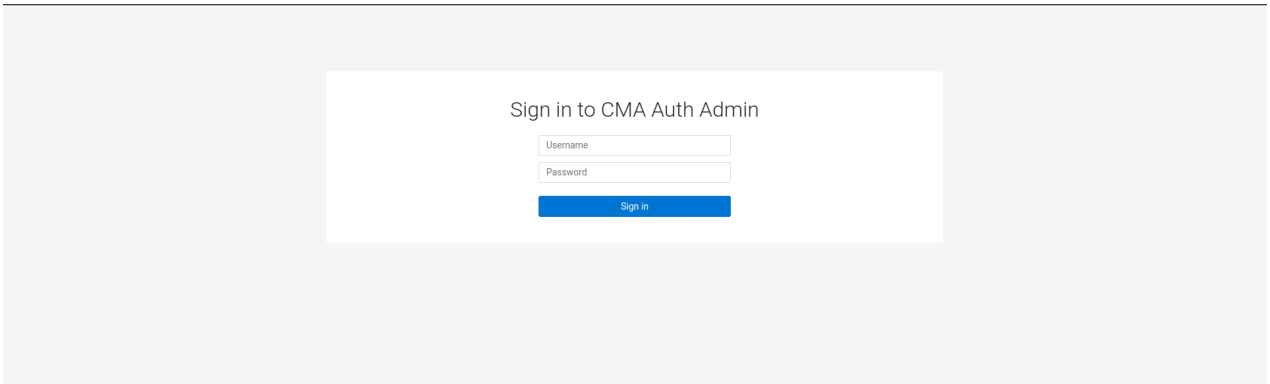
Administrators use this security model to define what actions users can perform within the CMA components.

Accessing the CMA Auth Server

By default, CMA Auth Server runs on port 9000 and is accessible at http://localhost:9000. In parcel deployments, users access CMA through the Gateway Server on port 8090, which proxies authentication requests to the Auth Server.

The server comes pre-configured with a default administrative account admin:admin with the ROLE_CMA_ADMIN role for full access to all features and settings within the CMA Auth Server.

Figure 10: CMA Auth Server log-in page

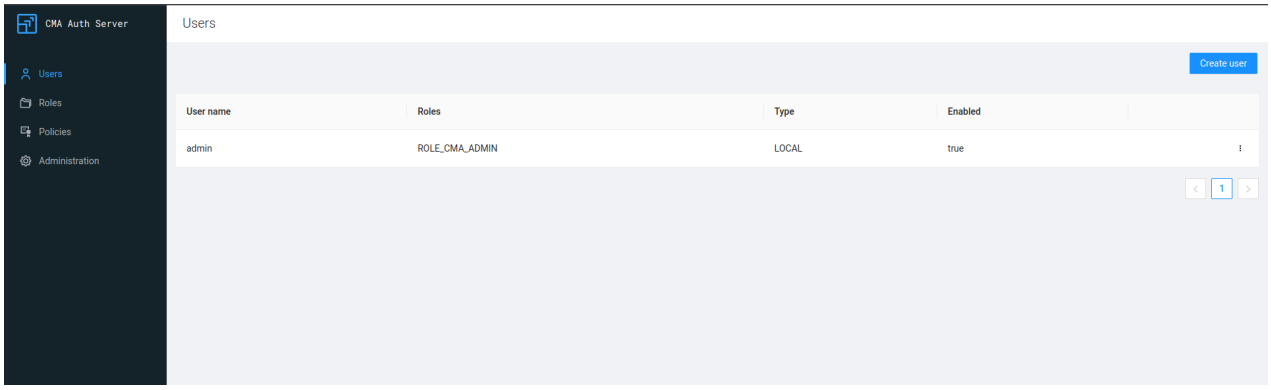


Note: Cloudera recommends changing the default password after initial login for security reasons.

User management

The CMA Auth Server provides a comprehensive interface for managing users through the Users tab in the left navigation pane.

Figure 11: CMA Auth Server Users tab



From this interface, administrators can perform the following actions:

- Create new users with specific roles
- Modify existing user details and role assignments
- Delete users who are no longer required
- Reset user passwords

Each user can be assigned one or more roles that determine their permissions within the CMA.

Important: The system prevents deletion of the currently logged-in user and the last administrator user.

Role management

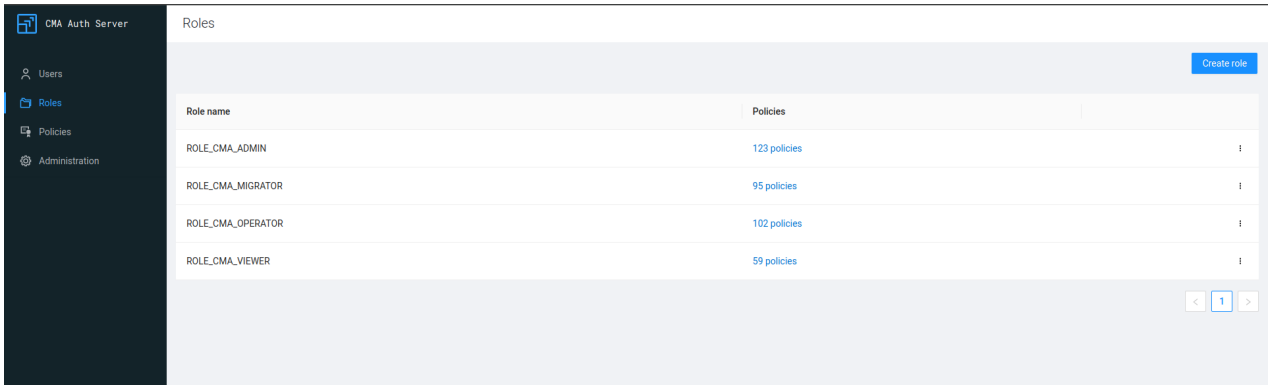
Roles define the set of actions a user can perform within CMA. CMA Auth Server comes with predefined roles that cover common use cases.

Table 12: Cloudera Migration Assistant roles with their access levels

Role	Description	Access level
ROLE_CMA_ADMIN	Administrative role	Full access to all features and settings
ROLE_CMA_MIGRATOR	Migrations role	• Modify access on CMA Server • Read access on CMA Auth Server
ROLE_CMA_OPERATOR	Operations role	• Full access on CMA Server • Read access on CMA Auth Server
ROLE_CMA_VIEWER	Viewer role	Read-only access to content

To manage roles, go to the Roles tab in the left navigation pane.

Figure 12: CMA Auth Server Roles tab



Important: The system prevents deletion of roles that are currently assigned to users. To remove a role, you must first remove it from all users.

Policy management

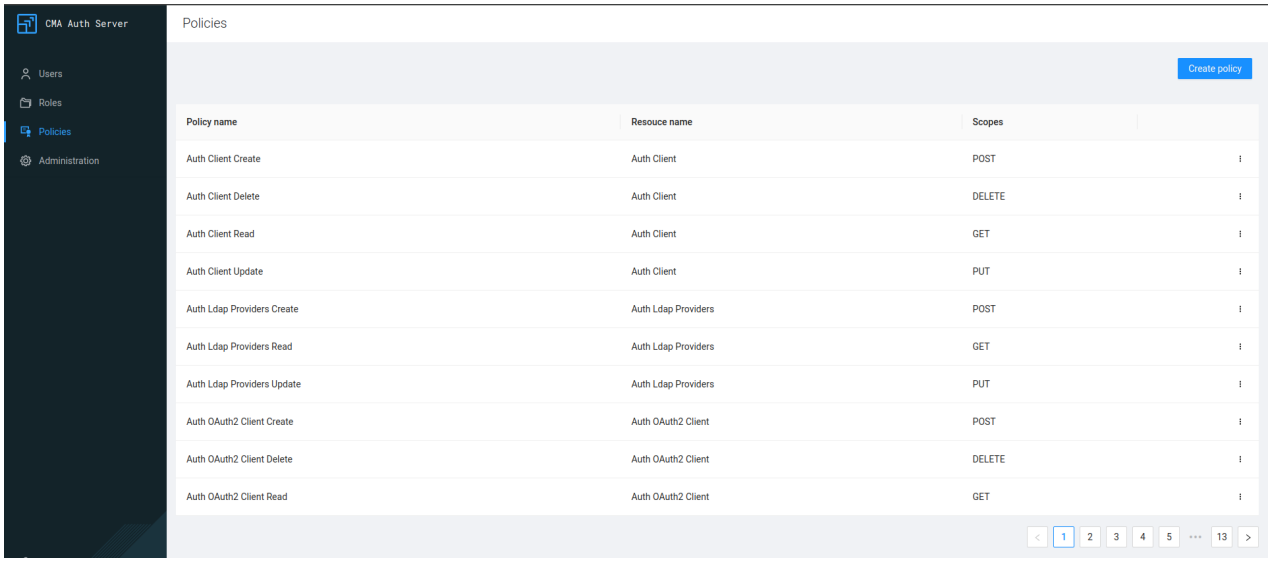
Policies in CMA Auth Server define the specific permissions for each resource and action (scope). CMA Auth includes a default set of policies covering all CMA resources and their available actions.

Policies include the following key characteristics:

- Default policies cover a single resource with a single scope.
- Custom policies support multiple scopes for specific resources.
- Existing policies allow modification or deletion as needed.

To manage policies, go to the Policies tab in the left navigation pane.

Figure 13: CMA Auth Server Policies tab

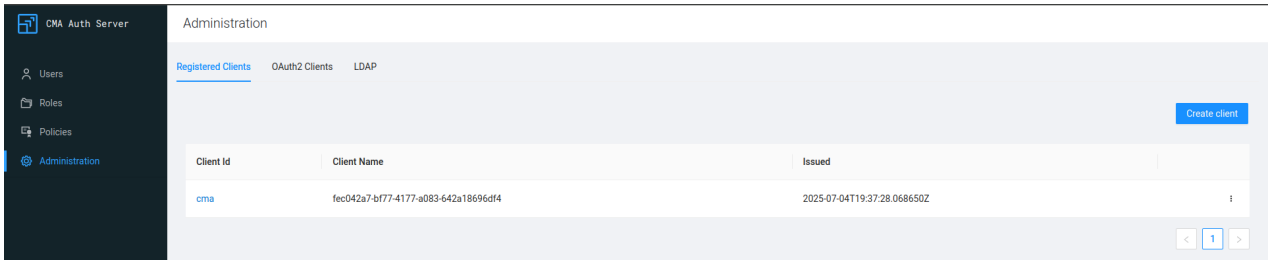


Important: The system prevents deletion of policies that are currently assigned to roles. To remove a policy, you must first remove it from all roles.

Client configuration

CMA Auth Server includes a built-in OAuth2 server that manages authorization for various components within the CMA. You can configure these settings in the Administration Clients section.

Figure 14: CMA Auth Server Clients tab



By default, the CMA Auth Server creates a cma client for the CMA Master that handles server authorization.

External OAuth2 integration

CMA Auth Server also supports integration with external OAuth2 providers, with current support for Okta. To configure these integrations, go to Administration OAuth2 Clients.

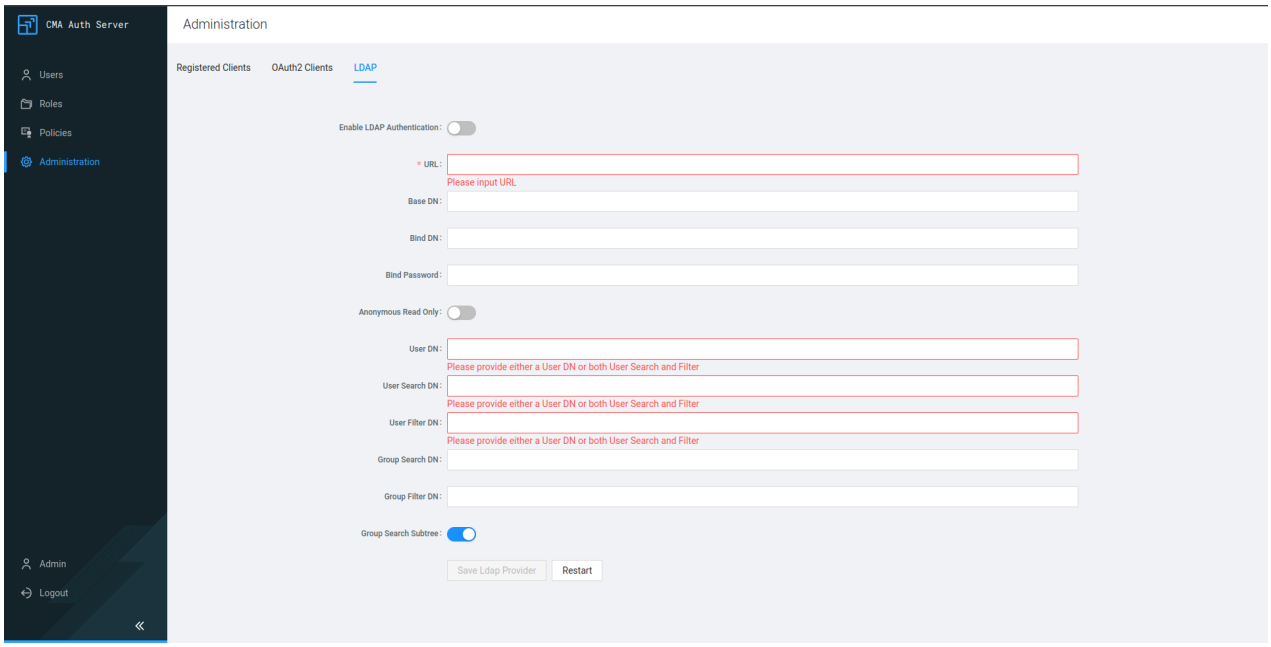
LDAP configuration

For enterprise environments, CMA Auth Server supports LDAP authentication, allowing users to authenticate using their existing directory credentials.

To configure LDAP, perform the following steps:

1. Go to the Administration LDAP .
2. Enter your LDAP server details and connection parameters.
3. Save changes.
4. Click the Restart button to apply changes.

Figure 15: CMA Auth Server LDAP tab



Note: After applying LDAP configuration changes, the CMA Auth Server restarts and you must sign in again.

Enabling TLS/SSL for CMA

Cloudera Migration Assistant uses TLS/SSL to encrypt communication between components. In parcel deployments, Cloudera Manager Auto-TLS provisions certificates automatically.

Cloudera Migration Assistant uses TLS/SSL to encrypt all communication between its components. In parcel deployments, TLS is configured automatically through Cloudera Manager's Auto-TLS feature — no manual steps are needed.

Auto-TLS (default)

When Auto-TLS is enabled in Cloudera Manager, all Cloudera Migration Assistant components (CMA Master, CMA Agent, Gateway, Auth Server) automatically receive keystores, truststores, and certificates provisioned by Cloudera Manager. No additional configuration is required.

gRPC TLS (Master-Agent communication)

Cloudera Migration Assistant encrypts the gRPC channel between the CMA Master and CMA Agents by default. All command dispatch, scan result streaming, and status updates are encrypted in transit.

When no external keystore is configured, the Master auto-generates a self-signed PKCS12 keystore (grpc.p12) at startup and distributes its public certificate to agents via the gateway discovery endpoint. Agents verify the certificate using HMAC-SHA256 signatures keyed to CMA_AGENT_CLIENT_SECRET before importing it into their truststore.

When Auto-TLS is active (Cloudera Manager-managed keystore), keystore generation is skipped and the existing Cloudera Manager keystore is used.

Certificate bootstrap on the agent

On first start, the CSD startup script automatically:

1. Fetches the gRPC certificate and gateway TLS certificate from GET /api/agent/config
2. Verifies each certificate's HMAC signature
3. Imports both into the agent truststore

A guard prevents unnecessary re-download on every restart.

Rotating the gRPC certificate

1. Set CMA_GRPC_ROTATE_CERT=true in the CMA Master service configuration in Cloudera Manager.
2. Restart the CMA Master. A new grpc.p12 is generated and the HMAC recomputed.
3. Restart each CMA Agent. Agents fetch the new certificate and rebuild their truststore.
4. Remove CMA_GRPC_ROTATE_CERT after the rotation is complete.

TLS configuration parameters

The following parameters are available in the CMA Master service configuration in Cloudera Manager:

Parameter	Default	Description
Minimum TLS version	TLSv1.3	Minimum TLS protocol version for HTTPS connections
TLSv1.2 ciphers	See Cloudera Manager defaults	Cipher suite preferences for TLS 1.2
TLSv1.3 ciphers	See Cloudera Manager defaults	Cipher suite preferences for TLS 1.3

Ports

Port	Protocol	Description
8090	HTTP	CMA web UI (redirects to HTTPS when TLS is enabled)
8093	HTTPS	CMA web UI (secure)
8891	gRPC + TLS	Master-Agent communication

Storing secrets in Vault [Technical Preview]

Store cluster credentials, SSH keys, and OAuth secrets in HashiCorp Vault instead of the CMA database so migration configuration files contain Vault paths and credentials are masked in the UI.

About this task

Note: This feature is in Technical Preview and is not ready for production deployments. Cloudera recommends trying this feature in test or development environments and encourages you to provide feedback on your experiences.

When Vault is enabled, CMA stores cluster credentials (Cloudera Manager passwords, SSH keys) as Vault KV v2 secrets. Configuration files generated during migration contain only Vault paths, not plaintext credentials. Credentials are masked in the CMA UI.

Note: The CMA-managed local Vault is in technical preview. For production, Cloudera recommends using an external Vault instance or a Cloudera Manager-managed Vault service.

What is stored in Vault

Secret type	Vault path	Stored when
Cloudera Manager credentials	secret/cma/clusters/<id>/cm	Cluster registration
SSH keys	secret/cma/clusters/<id>/ssh	SSH credential configuration
Database passwords	secret/cma/db	CMA startup (when Vault is enabled)
OAuth2 client secrets	secret/cma/auth	CMA startup (when Vault is enabled)

When Vault is not configured, credentials are stored encrypted in the CMA database (H2 or PostgreSQL). The encryption key is derived from the database password.

Parcel deployment options

You can enable Vault using a Cloudera Manager-managed Vault service or by running the Vault binary bundled with the CMA parcel.

Procedure

1. **Option 1 — CM-managed Vault service:** If the Vault CSD and parcel are deployed in Cloudera Manager, CMA can auto-detect the Vault service dependency.
   a) Deploy the Vault parcel and add the Vault service in Cloudera Manager.
   b) In CMA Master ServerConfiguration, set Vault Role ID (AppRole auth) and Vault Secret ID (AppRole auth).
   c) Restart the CMA Master service.
2. **Option 2 — External Vault using the bundled Vault binary:** Use the Vault binary and scripts shipped with the CMA parcel on the CMA Server host.

   Prerequisites: SSH (root) access to the CMA Server host; the CMA parcel is installed and activated; python3, wget, and unzip are available; outbound HTTPS access to releases.hashicorp.com for the Vault binary download.

   a) SSH to the CMA Server host and install and start Vault.

   ```
   export VAULT_HOME=/opt/cloudera/parcels/cma/vault-local
   export SCRIPT_PATH=$VAULT_HOME/scripts/bin
   export PYTHON_EXECUTABLE=python3
   cd $VAULT_HOME/scripts/bin
   bash vault-server.sh setup
   bash vault-server.sh start
   ```

   By default, Vault listens on http://127.0.0.1:8210. Change the port with the VAULT_SERVER_PORT environment variable.

   b) Configure AppRole authentication and note the role_id and secret_id values.

   ```
   export VAULT_ADDR=http://127.0.0.1:8210
   export VAULT_TOKEN=$(python3 -c "import json; print(json.load(open('$HOM
   E/.vault_data/conf/.vault'))['root_token'])")
   ```

```
VAULT_BIN=/opt/cloudera/parcels/cma/vault-local/vault/vault

$VAULT_BIN auth enable approle

$VAULT_BIN policy write cma-policy - <<'EOF'
path "secret/*" {
  capabilities = ["create", "read", "update", "delete", "list"]
}
EOF

$VAULT_BIN write auth/approle/role/cma-role \
  token_policies=cma-policy \
  token_ttl=1h \
  token_max_ttl=4h

$VAULT_BIN read auth/approle/role/cma-role/role-id
$VAULT_BIN write -f auth/approle/role/cma-role/secret-id
```

c) In CMA Master ServerConfiguration, set External Vault URL (for example http://127.0.0.1:8210), Vault Role ID (AppRole auth), and Vault Secret ID (AppRole auth).

d) Restart the CMA Master service from Cloudera Manager.

Results

Verify that the vault profile is active on the CMA Server host:

```
grep 'profiles are active' /var/log/cma/cma-server.log | tail -1
grep 'vault-ext' /var/log/cma/cma-server.log | tail -1
```

The first command should list vault among active profiles. The second should show Loaded extension: vault-ext v1.0 .0.

Note: After initial configuration, CMA Master Server and Vault Server roles can be started, stopped, or restarted separately in Cloudera Manager.

Managing Vault after setup

To restart Vault after a host reboot:

```
export VAULT_HOME=/opt/cloudera/parcels/cma/vault-local
export PYTHON_EXECUTABLE=python3
cd $VAULT_HOME/scripts/bin
bash vault-server.sh start
```

To stop Vault:

```
cd /opt/cloudera/parcels/cma/vault-local/scripts/bin
bash vault-server.sh stop
```

The Vault server runs as a background process. If the host reboots, restart Vault before starting CMA when the vault profile is active.

For Vault token file issues, see *Vault Troubleshooting*.

CMA database backends

Cloudera Migration Assistant database backends consists of an embedded H2 datastore suitable for evaluation environments, alongside PostgreSQL configuration for production clusters and split Master-Agent layouts.

Unless you explicitly configure an external database, CMA uses an embedded H2 datastore that installs itself on the first boot and requires no administrator provisioning. Liquibase migrations still run so schema versions stay aligned with the code release.

Cloudera recommends migrating to PostgreSQL 12+ when you need clustered durability, centralized backups, or when CMA Agents and CMA Masters must point at different JDBC endpoints. Depending on your topology, every role that persists state, including Master, Auth, Gateway, and Agent, requires direct JDBC connectivity to the target database.

Regardless of vendor, credential files (db.properties, auth-db.properties, Agent variants) rotate through Cloudera Manager for parcel installs through Cloudera Manager. Keep CMA_DBA_PASS or CM-safe equivalents synchronized whenever you change database passwords.

Related Information

Configuring with PostgreSQL

Configuring CMA with PostgreSQL

Provision PostgreSQL users, databases, and JDBC wiring for Cloudera Migration Assistant parcel deployments, including Liquibase schema management, deployment topologies, H2 cutovers, and troubleshooting.

Before you begin

- You must use PostgreSQL 12 or later.
- You must configure a PostgreSQL superuser account, for example, postgres, to run the setup script.
- You must install the psql client on the host where you run the setup script.
- You must configure network connectivity from the CMA Master and CMA Agent hosts to the PostgreSQL server.
- If CMA Agent and CMA Master use different PostgreSQL instances, for example, each on its own cluster node, you must run the setup script once per PostgreSQL instance.

About this task

Create CMA databases.

Run the setup script from the parcel directory on any host that has psql installed and can reach the PostgreSQL server:

```
/opt/cloudera/parcels/cma/bin/setup-cma-postgres.sh \
          --host <pg-host> \
          --admin-user postgres \
          --admin-password <superuser-password>
```

Note: The setup script always creates all three databases (cma, cma-auth, cma-agent). In a Master-Agent split deployment where the Agent uses a separate PostgreSQL instance, only the cma-agent database on the Agent's PostgreSQL is used. The unused cma and cma-auth databases can be safely ignored or dropped.

The script creates three databases and their owning users:

Database	User	Purpose
cma	cma	CMA Master
cma-auth	cma (shared)	CMA Auth Server
cma-agent	cma-agent	CMA Agent

Passwords are auto-generated and printed at the end. Save them for configuring CMA.

Setup Script Options

Option	Default	Description
--host	**(required)**	PostgreSQL hostname or IP
--admin-user	**(required)**	PostgreSQL superuser (e.g., postgres)
--admin-password	**(required)**	PostgreSQL superuser password
--port	5432	PostgreSQL port
--cma-user	cma	CMA Master database user
--cma-password	**(auto-generated)**	CMA Master database user password
--auth-user	*(same as --cma-user)*	CMA Auth database user
--auth-password	*(same as --cma-password)*	CMA Auth database user password
--agent-user	cma-agent	CMA Agent database user
--agent-password	**(auto-generated)**	CMA Agent database user password

Manual Database Creation

If you prefer to create the databases manually instead of using the setup script:

```
sudo -u postgres psql

        CREATE USER cma WITH PASSWORD 'your_secure_password';
        CREATE USER "cma-agent" WITH PASSWORD 'your_agent_password';

        CREATE DATABASE cma OWNER cma;
        CREATE DATABASE "cma-auth" OWNER cma;
        CREATE DATABASE "cma-agent" OWNER "cma-agent";

        GRANT ALL PRIVILEGES ON DATABASE cma TO cma;
        GRANT ALL PRIVILEGES ON DATABASE "cma-auth" TO cma;
        GRANT ALL PRIVILEGES ON DATABASE "cma-agent" TO "cma-agent";

        \c cma
        GRANT ALL ON SCHEMA public TO cma;
        \c cma-auth
        GRANT ALL ON SCHEMA public TO cma;
        \c cma-agent
        GRANT ALL ON SCHEMA public TO "cma-agent";
```

Configure CMA

Parcel Deployment (Cloudera Manager)

In Cloudera Manager, configure the following parameters for each CMA service:

CMA Master service configuration:

Parameter	Value
Database Type	postgres
Database Host	PostgreSQL hostname or IP
Database Port	5432 (or your custom port)
Database User	cma
Database Password	Password from setup script

CMA Agent service configuration:

Parameter	Value
Database Type	postgres
Database Host	PostgreSQL hostname or IP
Database Port	5432 (or your custom port)
Database User	cma-agent
Database Password	Password from setup script

Then restart both CMA services.

Environment variables reference

Parcel deployments map these environment variables from Cloudera Manager service configuration. Use the safety valve section below when additional variables are required for the Auth Server role.

CMA Master

Variable	Default	Description
CMA_DB_TYPE	h2	Database backend: h2 (embedded) or postgres (external PostgreSQL)
CMA_DB_HOST	—	PostgreSQL hostname (required when CMA_DB_TYPE=postgres)
CMA_DB_PORT	5432	PostgreSQL port
CMA_DB_NAME	cma	Database name for CMA Master
CMA_DBA_USER	cma	Database username
CMA_DBA_PASS	—	Database password (**mandatory** for all database types, including H2)

CMA Auth Server

Variable	Default	Description
CMA_AUTH_DB_HOST	CMA_DB_HOST	PostgreSQL hostname for auth server
CMA_AUTH_DB_PORT	CMA_DB_PORT	PostgreSQL port for auth server
CMA_AUTH_DB_NAME	cma-auth	Database name for auth server
CMA_AUTH_DBA_USER	CMA_DBA_USER	Auth database username
CMA_AUTH_DBA_PASS	CMA_DBA_PASS	Auth database password

CMA Agent

Variable	Default	Description
CMA_AGENT_DB_HOST	—	PostgreSQL hostname for agent (required when CMA_DB_TYPE=postgres)
CMA_AGENT_DB_PORT	5432	PostgreSQL port for agent
CMA_AGENT_DB_NAME	cma-agent	Database name for agent
CMA_AGENT_DBA_USER	cma-agent	Agent database username
CMA_AGENT_DBA_PASS	—	Agent database password (required when CMA_DB_TYPE=postgres)

Schema Management

CMA uses Liquibase for automatic schema management. When starting with the postgres profile, Liquibase creates all required tables, indexes, and seed data automatically. No manual schema creation is needed.

Deployment Topologies

Single cluster (Master + Agent co-located)

All three databases (cma, cma-auth, cma-agent) on a single PostgreSQL instance. Run the setup script once. Configure both services to point to the same PostgreSQL host.

Master-Agent split (separate clusters)

CMA Master on the target cluster, CMA Agent on the source cluster. Each can use a local PostgreSQL instance:

- **Target cluster PostgreSQL**: cma, cma-auth, cma-agent databases (the target Agent and Master share the same PostgreSQL)
- **Source cluster PostgreSQL**: cma-agent database only

Run the setup script on each PostgreSQL host. Configure the CMA Agent on the source cluster with the CMA Mast er URL pointing to the Master's gRPC port (default 8091) on the target cluster.

When the CMA Agent is on a different cluster from the Master, the CMA Master URL parameter must be set in the Agent's service configuration in Cloudera Manager (format: http://<master-host>:<grpc-port>).

Limitations

- **No H2-to-PostgreSQL migration**: Switching from H2 to PostgreSQL requires a fresh start. Existing H2 data cannot be migrated automatically.
- **Separate databases**: CMA Master, the Auth Server, and CMA Agent each use separate databases. All must be configured when using PostgreSQL.

Switching from H2 to PostgreSQL

When switching an existing CMA deployment from H2 to PostgreSQL, the Master starts with a fresh database. CMA Agents, however, retain their local H2 databases which contain stale server and cluster IDs referencing the old Master database. This causes agents to fail registration because those IDs no longer exist.

To resolve this, wipe the agent H2 databases on **every** agent host after switching the Master to PostgreSQL:

```
# On each CMA Agent host:
        # 1. Stop the CMA Agent service (via CM or systemctl)
        # 2. Delete the agent's local H2 database
        rm -f /var/lib/cma-agent/data/db/cma-agent.mv.db
        # 3. Start the CMA Agent service
```

After restart, each agent will re-register with the Master and receive new IDs.

Similarly, if the Master was previously running with H2 and accumulated data (/var/lib/cma/data/, /var/lib/cma/con fig/), clear those directories before switching to PostgreSQL to avoid H2 startup errors:

```
# On the CMA Master host (stop service first):
        sudo rm -rf /var/lib/cma/data/* /var/lib/cma/config/*
```

Troubleshooting

PostgreSQL superuser password not known

Cloudera Manager's PostgreSQL typically uses md5 authentication for all connections. If you don't have the postgres superuser password, you can temporarily enable peer authentication (OS-level user matching) to set one:

```
# Add peer auth as the first rule (takes priority)
        sudo sed -i '1i local   all   postgres   peer' /var/lib/pgsql/12/
data/pg_hba.conf
        sudo systemctl reload postgresql-12

        # Set a known password
        sudo -u postgres psql -c "ALTER USER postgres WITH PASSWORD 'new_pa
ssword';"
```

```
        # Remove the peer auth rule
        sudo sed -i '1d' /var/lib/pgsql/12/data/pg_hba.conf
        sudo systemctl reload postgresql-12
```

Remote connections rejected

If the CMA Agent connects to a PostgreSQL on a different host and gets "connection refused" or authentication errors, verify:

1. PostgreSQL listens on all interfaces (listen_addresses = '*' in

postgresql.conf)

1. pg_hba.conf allows remote connections:

```
host    all    all    0.0.0.0/0    md5
```

1. Reload PostgreSQL after changes: sudo systemctl reload postgresql-12

Connection refused or hostname resolution error

Verify the PostgreSQL host is reachable from the CMA host:

```
psql -h <pg-host> -p <pg-port> -U <user> -d postgres -c "SELECT 1"
```

Check that pg_hba.conf on the PostgreSQL server allows connections from the CMA hosts.

Missing Database Host or Password error on startup

When CMA_DB_TYPE is set to postgres, the Database Host and Database Password parameters are required. In Cloudera Manager, verify these are set in the CMA service configuration.

Parcel deployment: environment variable injection via safety valve

In parcel deployments, CMA reads database parameters from the CM-managed service configuration. If additional environment variables are needed (e.g., for the Auth Server), you can inject them using CM safety valves.

In Cloudera Manager, set the following for the **CMA_AUTH_SERVER** role:

CMA_AUTH_SERVER > Configuration > CMA_AUTH_SERVER role Environment Advanced Configuration Snippet (Safety Valve):

```
CMA_AUTH_DB_HOST=db.example.com
        CMA_AUTH_DB_PORT=5432
        CMA_AUTH_DB_NAME=cma-auth
        CMA_AUTH_DBA_USER=cma
        CMA_AUTH_DBA_PASS=your_secure_password
```

Safety valve values are injected directly into the Java process environment by CM, bypassing any shell script issues. Restart the CMA service after applying these changes.

Procedure

Perform the prerequisites, UI actions, commands, and validations described in this topic in the order they appear.

Integrating Cloudera Replication Manager

Install, flag, and register Cloudera Replication Manager so Cloudera Migration Assistant can delegate Base-to-Base replication policies for HDFS, Hive, HBase, and HDFS-to-Ozone scenarios.

About this task

CMA uses **Cloudera Replication Manager** (Cloudera Replication Manager) as the replication engine for Cloudera Base on premises to Cloudera Base on premises migrations.

Cloudera Replication Manager is a Cloudera service that creates and manages data replication policies between clusters. It supports the replication of the following data:

- HDFS data, including HDFS to Ozone
- Hive metadata and data
- HBase tables

CMA delegates all data replication operations to Cloudera Replication Manager through its REST API.

Cloudera Replication Manager is required for Cloudera Base on premises to Cloudera Base on premises migrations that involve data replication.

Table 13: Cloudera Replication Manager requirement by migration type

Migration type	Cloudera Replication Manager required?
HDFS replication (Base to Base)	Yes
HDFS to Ozone migration	Yes
Hive replication (Base to Base)	Yes
HBase replication (Base to Base)	Yes
Scanning clusters	No

Cloudera recommends deploying Cloudera Replication Manager (DMX) 1.2 or higher versions on the target cluster, on the same node as Cloudera Manager. It is installed as a separate Cloudera Manager parcel alongside the CMA parcel.

```
Target Cluster Node (Cloudera Manager host)
        ### CMA  (parcel)
        ### Cloudera Replication Manager (parcel)
```

Before you begin

- The Cloudera Manager version must be 7.13.2 or higher versions on the target cluster because of a requirement for HDFS to Ozone migrations.
- You must deploy Cloudera Cloudera Replication Manager on the same node as Cloudera Manager on the target cluster.
- The Cloudera Replication Manager REST API port **9011** must be accessible from the CMA Agent on the target cluster.
- You must deploy the CMA Agent and connect to the CMA Master on the target cluster.
- You must configure the Cloudera Manager administrator credentials for both source and target clusters in CMA.
- The following network requirements must be met:

Table 14: Network requirements

From	To	Port	Protocol	Purpose
CMA Agent (target)	Cloudera Replication Manager	9011	REST (HTTP)	Replication policy management
Cloudera Replication Manager	Source Cloudera Manager	7180/7183	HTTP/HTTPS	Source cluster access
Cloudera Replication Manager	Target Cloudera Manager	7180/7183	HTTP/HTTPS	Target cluster access

About this task

Procedure

1. Enable the HDFS to Ozone flag on the Cloudera Manager Server host.

   For HDFS to Ozone migrations, Cloudera Manager requires the CMF_FF_API_H2O_REPLICATION feature flag to be enabled. Without this flag, the Cloudera Replication Manager API endpoints for HDFS to Ozone replication are not available.

   Note: This flag is only required for HDFS to Ozone migrations. It is not required for HDFS-to-HDFS, Hive, or HBase replication.

   a) Append the following line to /etc/default/cloudera-scm-server on the Cloudera Manager Server host:

   ```
   export CMF_FF_API_H2O_REPLICATION=true
   ```

   b) Restart the Cloudera Manager Server.

   ```
   sudo systemctl restart cloudera-scm-server
   ```

   c) Wait for Cloudera Manager to become available again on port 7180 before proceeding with the CMA setup.
2. Register the Cloudera Replication Manager instance in the Cloudera Migration Assistant UI to allow CMA Agents to locate and communicate with the replication service.
   a) In the CMA Web UI, go to Settings.
   b) Go to the Cloudera Replication Manager App Access section.
   c) Configure your connection fields using one of the following validation methods:
      - If CMA has already detected a Cloudera Replication Manager service on one of the registered clusters, select it from the Suggestions drop-down list to auto-fill the form.
      - Manually enter your target replication parameters in the following fields:
        - Cloudera Replication Manager App URL — The full URL of the Cloudera Replication Manager REST API, including protocol, host, and port, for example, https://rm-host.example.com:9011.
        - Cloudera Replication Manager App username — The username for authenticating with the Cloudera Replication Manager API.
        - Cloudera Replication Manager App password — The password for authenticating with the Cloudera Replication Manager API.
   d) Click Save and Push to Agents.

Results

CMA saves the Cloudera Replication Manager credentials and pushes them to all connected CMA Agents. The Agents then use these credentials to register the managed clusters with Cloudera Replication Manager and to manage replication policies during migration execution.

Note: If no Cloudera Replication Manager App is detected on your clusters, the **Settings** page displays the No Cloudera Replication Manager App found on your clusters. warning. In this case, verify that the Cloudera Replication Manager is correctly installed and running on the target cluster before proceeding.

The integration subsequently automates the following background operations without manual user intervention:

- Automated cluster registration

  When a migration plan initializes, the CMA Master sends an internal RM_CLUSTER_REGISTRATION command to the target CMA Agent. The Agent automatically calls the Cloudera Replication Manager REST API endpoint (/cm/registerClusters) to match the source and target cluster pairs, and writes the credentials to the /store/credential endpoint. If a cluster pair is already registered, the agent skips this step automatically.
- Migration execution interactivity

During active migration runtime, the target CMA Agent leverages the connection to create replication policies, manage HDFS and Ozone snapshots for efficient incremental delta transfers, poll policy statuses for real-time UI tracking, and clean up expired policies upon completion.

All Cloudera Replication Manager interactions are performed by the CMA Agent on the target cluster and reported back to CMA Master for tracking and display in the UI.

Table 15: Troubleshooting the integration

Issue	Possible cause	Resolution
Replication policy creation fails	Clusters not registered in Cloudera Replication Manager	Check CMA Agent logs for registration errors, then verify if Cloudera Replication Manager is running on port 9011.
Registration fails with credential error	Incorrect CM credentials in CMA	Update Cloudera Manager credentials in CMA for the affected cluster.
Cloudera Replication Manager not reachable from Agent	Network/firewall issue	Ensure that port 9011 is open between the Agent and Cloudera Replication Manager on the target cluster.
HDFS to Ozone replication not available	CM version too old	Upgrade Cloudera Manager on the target cluster to 7.13.2 or higher versions.

Related Information

Deployment

Registering clusters

Managing Ozone storage [Technical Preview]

Cloudera Migration Assistant provides comprehensive management of Apache Ozone storage structures on Cloudera Base clusters. You can scan existing Ozone storage, browse volumes and buckets, create new storage structures manually or by importing a YAML layout definition, and migrate data between HDFS and Ozone or between Ozone locations.

About this task

Note: This feature is in Technical Preview and is not ready for production deployments. Cloudera recommends trying this feature in test or development environments and encourages you to provide feedback on your experiences.

Ozone is a scalable, distributed object store optimized for big data workloads. In CMA, Ozone storage is organized in the following three-level hierarchy:

- **Volumes** – Top-level containers that provide namespace isolation. Each volume has an owner, a namespace quota (number of buckets), and a space quota.
- **Buckets** – Storage containers within a volume that hold keys (files and folders). Buckets have configurable layout types, replication settings, quotas, encryption, and snapshot options.
- **Folders** – Directory structures within a bucket that organize data. Folders can be nested to create complex hierarchies.

CMA allows you to manage this hierarchy through the UI and API, track changes through server-sent events (SSE), and apply the storage layout to the actual Ozone cluster through the CMA Agent.

Before you begin

- You must set up Cloudera Migration Assistant correctly. For instructions, see *Deploying Cloudera Migration Assistant*.

- The destination cluster must have Apache Ozone installed and the Ozone service must be running.
- You must register a Cloudera Base in CMA. If you do not have a cluster yet, for instructions, see *Registering clusters*.
- You must configure Kerberos authentication correctly if the cluster is secured. CMA uses the configured migration user principal for Ozone operations.

Procedure

1. Scan Ozone storage to discover existing volumes, buckets, and folders.
   a) Click on the cluster you want to scan on the **Clusters** page.
   b) Click Start Scanning to open **Scan Settings**.
   c) Select Ozone scan.

      Important:

      The Ozone service must be available on the cluster before initiating the scan. CMA checks service availability at scan time and reports an error if the Ozone service is not responding.

   d) Click Scan selected.

      You are redirected to the scanning progress page where you can monitor whether the scan completed successfully or encountered an error.
2. Browse the Ozone cluster in the cluster view.

   The scan results display volumes, buckets, and folder hierarchies. For each bucket, CMA displays the layout type, replication settings, quotas, and size information. For each volume, CMA displays the namespace quota, space quota, and owner.

   a) View all volumes on the cluster with the (owner, quotas, and bucket count attributes.
   b) Drill into a volume to view its buckets with the layout, replication type, quotas, encryption key, and GDPR enforcement attributes.
   c) Drill into a bucket to browse its folder hierarchy, including file sizes and ownership.
   d) View Ozone snapshots for buckets that have snapshots enabled.
   e) Filter folders by keyword, label, or related service such as Hive tables or databases that reference specific Ozone paths.
3. Create a volume.
   a) Go to the Ozone storage view for the cluster.
   b) Click Add Volume.
   c) Provide the following properties:

      ### Table 16: Volume configuration properties

Property	Description
Name	Name of the volume (required)
Namespace quota	Maximum number of buckets allowed in the volume
Space quota	Maximum storage space for the volume, for example, 10TB
User	Owner of the volume

   d) Click Create.
4. Update a volume.
   a) Select the volume you want to modify.
   b) Edit the mutable properties (namespace quota, space quota, user).
   c) Click Save.
5. Delete a volume.
   a) Select the volume you want to delete.

b) Click Delete.

Warning: Deleting a volume removes all its child buckets and folders. This action cannot be undone.

6. Create a bucket.
   a) Go to the volume where you want to create the bucket.
   b) Click Add Bucket.
   c) Provide the following properties:

Table 17: Bucket configuration properties

Property	Description
Name	Name of the bucket (required)
Layout	Bucket layout type. Possible values are FILE_SYSTEM_OPTIMIZED, OBJECT_STORE, or LEGACY
Replication type	Type of the replication. Can be RATIS or EC (Erasure Coding)
Replication	For RATIS: ONE or THREE. For EC: codec-data-parity-chunksize format (for example, RS-3-2-1024k)
Namespace quota	Maximum number of keys allowed in the bucket
Space quota	Maximum storage space for the bucket (for example, 1TB)
User	Owner of the bucket
Bucket encryption key	Name of the encryption key registered in the Ozone Key Management Server
Enforce GDPR	Enable Transparent Data Encryption for GDPR compliance
Create snapshot	Enable snapshot creation for the bucket

   d) Click Create.
7. Update a bucket.
   a) Select the bucket you want to modify.
   b) Edit the mutable properties.
   c) Click Save.
8. Delete a bucket.
   a) Select the bucket you want to delete.
   b) Click Delete.

Warning: Deleting a bucket removes all its child folders. This action cannot be undone.

9. Create a folder.
   a) Go to the bucket where you want to create a folder.
   b) Click Add Folder.
   c) Provide the folder name.
   d) Click Create.

   Non-existent intermediate folders are created automatically.
10. Rename a folder.
   a) Select the folder you want to rename.
   b) Provide the new name.
   c) Click Save.
11. Delete a folder.
   a) Select the folder you want to delete.

b) Click Delete.

Known limitations

No input validation

CMA does not validate layout definitions for volumes, buckets, or quotas.

No feasibility pre-check

CMA does not verify whether the specified settings are feasible on the target cluster.

Limited migration testing

Ozone Layout Provisioning has not been tested as part of the end-to-end migration workflow.

Use this feature for evaluation purposes only. Behavior and APIs may change in future releases.

CMA provides a **Storage Layout Editor** for designing and deploying Ozone storage structures.

Both the **Storage Layout Editor** and the **Ozone browser** use colors and text styles to indicate the status of volumes, buckets, and folders. The two views display different stages of the same lifecycle.

The **Storage Layout Editor** displays both unsaved (local) changes and changes that were saved to the database but not yet deployed to the cluster.

Table 18: Storage Layout Editor statuses

Status	Color	Style	Description
New or locally modified (unsaved)	Light red	Normal	Node was created or edited in the editor but not yet saved to the database.
Locally deleted (unsaved)	Light red	Strikethrough	Node was deleted in the editor but the deletion has not been saved yet.
Saved, pending CREATE	Red	Normal	Node was saved to the database and is queued for deployment to the cluster.
Saved, pending MODIFY	Default	Italic	Node was saved to the database and is queued for deployment to the cluster.
Saved, pending DELETE	Default	Strikethrough	Node was saved as a deletion and is queued for removal from the cluster.
Saved, no pending event	Default	Normal	Node is deployed and up to date.

The **Ozone browser** only displays nodes that have been saved to the database. Unsaved editor changes are not visible here. If you have unsaved changes in the editor, the browser shows the last saved state of those nodes until you save.

Table 19: Ozone browser statuses

Status	Color	Style	Description
Pending CREATE (saved, not yet deployed)	Red	Normal	Node was created and saved, awaiting deployment to the cluster.
Pending MODIFY (saved, not yet deployed)	Default	Italic	Node was modified and saved, awaiting deployment to the cluster.
Pending DELETE (saved, not yet deployed)	Default	Strikethrough	Node is queued for deletion. All children of a pending-delete node are also displayed with strikethrough.

Status	Color	Style	Description
No pending event	Default	Normal	Node is deployed and up to date.

Note: The strikethrough style in the browser propagates to all children of a node marked for deletion, so you can see the full impact of a pending delete operation before it is deployed.

12. Plan and deploy an Ozone storage layout using the **Storage Layout Editor**.

a) Go to the Layout tab in the cluster view to access the **Storage Layout Editor**.

Design the layout structure in the Plan tab and apply the layout to the cluster in the Deploy tab.

The Plan tab displays the following panels.

- Left panel (**Available Templates**) — Contains reusable volume structures you can drag into the editor.
- Right panel (**Editor**) — Is the working area where you build the target Ozone structure.

13. Add volumes and buckets manually.

a) Select the cluster on the **Clusters** page.
b) Go to the Layout tab.
c) In the Plan tab, click the Add Volume button at the top of the editor panel.
d) Provide the volume properties in the drawer that opens, then click Create.
e) To add buckets to a volume, select the volume node and click Add Bucket.
f) To add folders to a bucket, select the bucket node and click Add Folder.
g) Click Save Plan in the toolbar to save all pending changes to the database.

The **Available Templates** panel on the left displays reusable volume structures. You can use the following template types:

- System templates — Built-in templates provided by Cloudera Migration Assistant. These cannot be deleted.
- User templates — Templates you create from volumes in the editor. These can be deleted.

14. Search and filter templates.

Use the Search field at the top of the **Available Templates** panel to filter templates by name. Use the Type drop-down list to filter by template type.

15. Drag a template into the editor.

a) In the **Available Templates** panel, click the template name to expand it and see its volumes, buckets, and folders.
b) Drag the node you want (volume, bucket, or folder) by its drag handle into the editor panel on the right.

You can drag any node from a template directly into the editor to add it to the layout. Volumes, buckets, and folders can all be dragged individually using the drag handle (the grip icon on the right side of each node). When you drag a node, the full subtree (all its children) moves with it.

16. Save a volume as a template.

You can save any volume in the editor as a template to reuse it later. Saving a volume as a template stores the entire volume structure, including its buckets and folders, in the **Available Templates** panel.

a) In the editor, hover over the volume node you want to save.
b) Click Add to Template that is the download icon that is displayed on hover.
c) Provide a name for the template.
d) Click Save.

The template is displayed in the **Available Templates** panel and is stored in the database.

17. Delete a user template.

a) In the **Available Templates** panel, hover over the template name.
b) Click the icon next to the template name.
c) Click Delete to confirm the deletion.

18. Import a layout from YAML or JSON.

You can define a complete Ozone storage structure in a YAML or JSON file and import it into the editor.

This example shows a YAML file defining volumes, buckets, and folders in a hierarchical structure.

```
volumes:
          - name: "my-volume"
            namespace-quota: 100
            space-quota: "10TB"
            user: "admin"
            buckets:
              - name: "my-bucket"
                create-snapshot: true
                namespace-quota: 1000
                space-quota: "1TB"
                user: "admin"
                layout: "FILE_SYSTEM_OPTIMIZED"
                replication-type: "RATIS"
                replication: "THREE"
                bucketkey: "my-encryption-key"
                enforcegdpr: false
                folders:
                  - name: "data"
                    folders:
                      - name: "raw"
                      - name: "processed"
                  - name: "logs"
```

a) In the Plan tab, click Import Structure in the toolbar.
b) Upload the YAML or JSON file or paste the YAML or JSON content.
c) Click Upload.

CMA validates the structure and loads the volumes, buckets, and folders into the editor. Validation errors are displayed if the YAML or JSON is malformed or contains invalid values.

d) Click Save Plan to save the imported structure to the database.

19. Deploy the layout to the cluster.

a) Click the Deploy tab in the **Layout** page.

The **Deployment Queue** table displays all saved volumes, buckets, and folders that have pending changes. The CREATE, MODIFY, or DELETE events are not yet applied to the cluster.

b) Review the pending changes in the **Volumes**, **Buckets**, and **Folders** tables.

Warning: Deployment is irreversible. After clicking Deploy to Ozone, rollback is not possible. Return to the Plan tab to make modifications before deploying.

c) Click Deploy to Ozone.

CMA sends the deploy command to the CMA Agent on the cluster. Monitor progress in the Commands tab on the cluster view. The agent processes the changes in the following order:

a. Create, update, or delete volumes.
b. Create, update, or delete buckets.
c. Create, update, or delete folders.

20. Use collections with Ozone locations.

You can organize Ozone locations into collections using labels for visual organization. This is useful when preparing for migration.

a) Go to the Ozone storage view and navigate to the folders you want to organize.
b) Select the Ozone locations.
c) Click Add to collection and select or create a collection.

Collections with Ozone locations can be added to migration plans for HDFS-to-Ozone or Ozone-to-Ozone migration.

21. Migrate HDFS to Ozone.

CMA supports migrating data from HDFS to Ozone storage using Cloudera Replication Manager policies. The migration process consists of the following sequence:

a. Ensure that the source cluster HDFS data is scanned and the destination cluster Ozone storage is set up.
b. Create a migration plan that includes HDFS locations from the source cluster.
c. In the structure mapping, map the source HDFS paths to destination Ozone paths (volume/bucket/folder).
d. Create an execution from the plan and click Run.

CMA uses the Cloudera Replication Manager Java API to create replication policies that copy data from HDFS to the mapped Ozone locations on the destination cluster.

22. Create folders on the destination cluster during migration planning.

a) In the migration plan split view, go to the destination cluster Ozone storage.
b) Click Create Folder on the destination side.
c) Provide the folder path.
d) Click Create.

The folder is created in the CMA data model and applied to the cluster during execution.

What to do next

Ozone Storage Layout Editor — UI reference

Note: This feature is in Technical Preview and is not ready for production deployments. Cloudera recommends trying this feature in test or development environments and encourages you to provide feedback on your experiences.

Known limitations

No input validation

CMA does not validate layout definitions for volumes, buckets, or quotas.

No feasibility pre-check

CMA does not verify whether the specified settings are feasible on the target cluster.

Limited migration testing

Ozone Layout Provisioning has not been tested as part of the end-to-end migration workflow.

Use this feature for evaluation purposes only. Behavior and APIs may change in future releases.

Internal reference. This section documents detailed UI interactions for the Ozone Storage Layout Editor. For the main workflow, see the steps above.

Using templates

The Available Templates panel shows reusable volume structures. System templates are built-in and cannot be deleted. User templates are created from volumes in the editor and can be deleted.

Use Search to filter templates by name and the Type dropdown to filter by template type. Drag any node (volume, bucket, or folder) by its drag handle into the editor; the full subtree moves with it.

To save a volume as a template, hover over the volume, click Add to Template, provide a name, and click Save. To delete a user template, hover over the template name and click the delete icon.

Importing a layout from YAML or JSON

Define volumes, buckets, and folders in a hierarchical YAML or JSON file and import it into the editor from the Plan tab using Import Structure. Upload a file or paste content, then click Upload. CMA validates the structure and loads nodes into the editor. Click Save Plan to persist the imported structure.

Deploying the layout to the cluster

After saving a plan, open the Deploy tab. The Deployment Queue table lists volumes, buckets, and folders with pending CREATE, MODIFY, or DELETE events. Review pending changes, then click Deploy to Ozone.

Caution: Deployment is irreversible. Return to the Plan tab to make modifications before deploying.

CMA sends the deploy command to the CMA Agent, which processes volumes, then buckets, then folders. Monitor progress in the Commands tab on the cluster view.

Node status colors and styles

The Storage Layout Editor and Ozone browser use colors and text styles to indicate node status. In the editor, light red indicates unsaved local changes; red or italic or strikethrough styles indicate saved changes pending deployment. In the browser, red indicates pending CREATE; italic indicates pending MODIFY; strikethrough indicates pending DELETE.

Setting up Cloudera Data Engineering to Cloudera Workflow Orchestrator Git migration [Technical Preview]

CMA can migrate Airflow DAGs from Cloudera Data Engineering virtual clusters into version-controlled Cloudera Workflow Orchestrator Git repositories. The migration produces a Pull Request (GitHub), Merge Request (GitLab), or branch (plain Git), which you review and merge — the source Cloudera Data Engineering environment is never modified.

About this task

Note: This feature is in Technical Preview and is not ready for production deployments. Cloudera recommends trying this feature in test or development environments and encourages you to provide feedback on your experiences.

Before migration engineers can use this feature, you must configure the CMA Agent with credentials for both the Cloudera Control Plane (Cloudera Data Engineering source) and the target Cloudera Workflow Orchestrator Git repository. CMA then automatically registers the required clusters.

When the CMA Agent starts with the required configuration, it automatically registers the following additional managed clusters in CMA Master:

- Cloudera Data Engineering cluster (<clusterName>-cde) — Represents the Cloudera Data Engineering environment. The agent uses Cloudera Control Plane API credentials to discover Cloudera Data Engineering services, virtual clusters, and Airflow DAGs.
- Git cluster (<clusterName>-git) — Represents the target Cloudera Workflow Orchestrator Git repository. The agent uses the repository URL and access token to scan the repository and create Pull Requests or Merge Requests.

Both clusters are displayed in the CMA Master UI and can be used as source and target in a migration plan.

Known limitations

- **No SSH support** – Plain Git connections support only HTTPS repository URLs.

- **No incremental migration** – Each migration execution creates a full Pull Request or Merge Request containing all mapped DAGs.
- **No automated rollback** – Once a Pull Request or Merge Request is created, it must be manually closed or reverted in the Git provider.
- **Single branch per plan** – All DAGs in a migration plan target one source branch (cma/<planName>).

Before you begin

- You must deploy CMA in parcel deployment mode. For more information, see *Deployment*.
- You must install and run the CMA Agent service on the cluster (Public Cloud or Cloudera platform Private Cloud Base).
- The CMA Agent must have network access to the Cloudera Control Plane API endpoint.
- The CMA Agent must have network access to Cloudera Data Engineering.
- The CMA Agent must have outbound HTTPS access to the Git provider (GitHub, GitLab, or self-hosted).
- You must have Cloudera Control Plane access key credentials (access key ID and private key) for the Control Plane.
- You must have a personal access token (PAT) for the target Cloudera Workflow Orchestrator Git repository with permissions to read the repository and create Pull Requests or Merge Requests.
- The target Cloudera Workflow Orchestrator Git repository must exist and the base branch, for example, main, must be present.
- If a firewall is in place between CMA components and external services, the following ports must be open:

Table 20: Network requirements

Source	Destination	Port	Protocol	Purpose
CMA Agent (Cloudera Data Engineering)	Cloudera Control Plane API	443	HTTPS	Discover Cloudera Data Engineering services and virtual clusters
CMA Agent (Cloudera Data Engineering)	Cloudera Data Engineering Knox gateway	443	HTTPS	Fetch DAG metadata and download DAG files
CMA Agent (Git)	GitHub or GitLab or self-hosted Git	443	HTTPS	Scan repository, create Pull Requests or Merge Requests
CMA Agent (Cloudera Data Engineering and Git)	CMA Master gRPC port	8091	gRPC	Agent-to-Master communication

Procedure

1. Enable the cma.cde-git-migration-enabled flag.
   a) In the CMA Master UI, go to Settings.
   b) Click the Feature Flags tab.
   c) Locate the cma.cde-git-migration-enabled flag and set it to true.
   d) Click Save.
   e) Go to the CMA Agent service in Cloudera Manager.
   f) Click the Configuration tab.
   g) Search for CMA_AGENT_SERVER_role_env_safety_valve.
   h) Add the Cloudera Control Plane and Git repository environment variables listed in the tables in the next step.
   i) Click Save to save the advanced configuration snippet configuration.
   j) Restart the CMA Agent service.
   k) In the CMA Master UI, go to Clusters.
   l) Confirm that two new clusters are displayed. If the clusters are not displayed within a few minutes, check the CMA Agent log for registration errors.

2. Configure the CMA Agent advanced configuration snippet in Cloudera Manager on the cluster where the CMA Agent is deployed:

   Cloudera Data Engineering and Git credentials are passed to the CMA Agent through the Cloudera Manager CMA_AGENT_SERVER_role_env_safety_valve advanced configuration snippet. These environment variables are read at startup. The agent uses them to register the Cloudera Data Engineering and Git clusters.

   The Cloudera Control Plane settings allow the CMA Agent to authenticate with the Cloudera Control Plane and the Cloudera Data Engineering Knox gateway.

   Table 21: Cloudera Control Plane environment variables

Variable	Description	Example
CMA_AGENT_CLUSTER_CLOUD_CONTROL_DATACENTER	Cloudera Control Plane API endpoint URL.	https://api.us-west-1.cdp.cloudera.com
CMA_AGENT_CLUSTER_CLOUD_CONTROL_ACCESS_KEY_ID	Cloudera Control Plane access key ID for Control Plane authentication. Obtained from the Cloudera Management Console in User Management Access Keys .	a1b2c3d4-...
CMA_AGENT_CLUSTER_CLOUD_CONTROL_PRIVATE_KEY	Cloudera Control Plane private key corresponding to the access key ID.	<YOUR_CDP_PRIVATE_KEY>
CMA_AGENT_CLUSTER_CLOUD_CONTROL_USERNAME	Cloudera Data Engineering username. Must be an **LDAP user**. Used by the CMA Agent to authenticate against the Cloudera Data Engineering Knox gateway when fetching DAG metadata.	admin
CMA_AGENT_CLUSTER_CLOUD_CONTROL_PASSWORD	Cloudera Data Engineering password corresponding to the username.	<CDE_PASSWORD>

   Note: CMA_AGENT_CLUSTER_CLOUD_CONTROL_USERNAME and CMA_AGENT_CLUSTER_CLOUD_CONTROL_PASSWORD are Cloudera Data Engineering credentials, not Cloudera Control Plane credentials. They are used only for Knox authentication against Cloudera Data Engineering virtual cluster APIs.

   The Cloudera Workflow Orchestrator Git repository settings allow the Git Agent to scan the target repository and create Pull Requests or Merge Requests.

   Table 22: Cloudera Workflow Orchestrator Git repository environment variables

Variable	Description	Example
CMA_AGENT_CLUSTER_GIT_REPOURL	HTTPS URL of the target Git repository. SSH URLs are not supported.	https://github.com/your-org/your-repo.git
CMA_AGENT_CLUSTER_GIT_TOKEN	Personal access token for the Git repository. For GitHub, the token requires the repo scope. For GitLab, it requires api or read_repository + write_repository.	<YOUR_GIT_PAT>
CMA_AGENT_CLUSTER_GIT_BRANCH	Base branch name. CMA creates a source branch named cma/<planName> from this branch and opens a Pull Request or Merge Request back to it. Defaults to main if not set.	main

3. Save and restart the CMA Agent.

   After the CMA Agent restarts, it registers both clusters with CMA Master automatically. No manual steps are required in the CMA UI.

4. Verify the cluster registration.

Table 23: Verification values for auto-registered clusters

Cluster Name	Platform Label	Purpose
<clusterName>-cde	Cloudera Data Services	Cloudera Data Engineering source that is used for scanning DAGs
<clusterName>-git	Git	Git target that is used for scanning the repository and creating Pull Requests and Merge Requests

Results

CMA automatically selects the correct Git provider based on the repository URL. You do not need to configure the provider type manually in most cases.

Table 24: Git provider auto-detection rules

Detection rule	Provider	Change type
URL contains github.com	GitHub	Pull Request
URL contains gitlab.com	GitLab	Merge Request
HTTP probe responds to /api/v4/version	GitLab (self-hosted)	Merge Request
HTTP probe responds to /api/v3/meta	GitHub Enterprise	Pull Request
None of the above	Plain Git	Branch push

To override auto-detection, set the git.providerType cluster configuration property to one of GITHUB, GITLAB, or JGIT.

Note: Plain Git (the fallback provider) only supports HTTPS. SSH-based Git URLs are not supported.

Table 25: Troubleshooting the migration

Symptom	Possible cause	Resolution
Cloudera Data Engineering or Git clusters are not displayed in the CMA UI	CMA Agent did not restart, or advanced configuration snippet variables are missing or misspelled.	Restart the CMA Agent service and check the Agent log for Cluster registration failed entries.
Cloudera Data Engineering scan fails with authentication error	CMA_AGENT_CLUSTER_CLOUD_CONTROL_USERNAME or CMA_AGENT_CLUSTER_CLOUD_CONTROL_PASSWORD are incorrect, or the Cloudera Data Engineering Knox gateway is unreachable.	Verify that the Cloudera Manager credentials are correct and that the Agent can reach the Cloudera Data Engineering Knox gateway on port 443.
Cloudera Data Engineering scan fails and no Cloudera Data Engineering are services found	CMA_AGENT_CLUSTER_CLOUD_CONTROL_DATACENTER points to the wrong region, or the access key credentials are invalid.	Verify the Control Plane URL and confirm that the access key is valid in the Cloudera Management Console.
Git scan fails with authentication error	CMA_AGENT_CLUSTER_GIT_TOKEN is missing, expired, or lacks the required permissions.	Regenerate the PAT in the Git provider and update CMA_AGENT_CLUSTER_GIT_TOKEN in the advanced configuration snippet.
Git scan fails, the branch is not found	CMA_AGENT_CLUSTER_GIT_BRANCH refers to a branch that does not exist.	Create the branch in the Git repository, or update the variable to point to an existing branch.
Cloudera Data Engineering cluster registered but no virtual clusters are displayed after the scan	A specific virtual cluster could not be reached (Knox auth failure or empty vcApiUrl).	Check the Cloudera Data Engineering scan command log in the CMA UI for SKIPPED entries with the affected virtual cluster name.

Related Information

Deployment
Registering clusters
Migrating Airflow DAGs from to a Git Repository [Technical Preview]