Machine Learning

Private Cloud Requirements

Date published: 2020-07-16
Date modified: 2024-05-30

https://docs.cloudera.com/

Legal Notice

Contents

Introduction to Private Cloud

With the Cloudera Machine Learning (CML) service, data scientists and partners can build and run machine learning experiments and workloads in a secure environment. CML on Private Cloud provides an identical experience to CML on Public Cloud, but running in your own on-premises data center.

Cloudera Machine Learning enables you to:

- Easily onboard a new tenant and provision an ML workspace in a shared OpenShift or ECS environment.
- Enable data scientists to access shared data on CDP Private Cloud Base and CDW.
- Leverage Spark-on-K8s to spin up and down Spark clusters on demand.

Cloudera Machine Learning requirements (OCP)

To launch the Cloudera Machine Learning service, the OpenShift Container Platform (OCP) host must meet several requirements. Review the following CML-specific software, NFS server, and storage requirements.

Requirements

Note:

Only the usage of SSD disks is supported with Private Cloud Data Services on OCP.

If necessary, contact your Administrator to make sure the following requirements are satisfied:

1. If you are using OpenShift, the installed OpenShift Container Platform must be version 4.10 or 4.8 (when upgrading from 1.4.x to 1.5.x, not performing a fresh installation). For ECS, refer to the *Hardware and Software Requirements* section in *CDP Private Cloud Experiences Installation Hardware Requirements and Managing a Private Cloud Experience Cluster 1.5.x.*
2. CML assumes it has cluster-admin privileges on the cluster.
3. Storage:
   a. Persistent volume block storage per ML Workspace: 600 GB minimum, 4.5 TB recommended.
   b. 1 TB of external NFS space recommended per Workspace (depending on user files). If using embedded NFS, 1 TB per workspace in addition to the 600 GB minimum, or 4.5 TB recommended block storage space.
   c. Access to NFS storage is routable from all pods running in the cluster.
   d. For monitoring, recommended volume size is 60 GB.
4. On OCP, CephFS is used as the underlying storage provisioner for any new internal workspace on PVC 1.5.x. A storage class named ocs-storagecluster-cephfs with csi driver set to "openshift-storage.cephfs.csi.ceph.com" must exist in the cluster for new internal workspaces to get provisioned.
5. A block storage class must be marked as default in the cluster. This may be rook-ceph-block, Portworx, or another storage system. Confirm the storage class by listing the storage classes (run `oc get sc`) in the cluster, and check that one of them is marked default.
6. If external NFS is used, the NFS directory and assumed permissions must be those of the cdsw user. For details see Using an External NFS Server in the Related information section at the bottom of this page.
7. If CML needs access to a database on the CDP Private Cloud Base cluster, then the user must be authenticated using Kerberos and must have Ranger policies set up to allow read/write operations to the default (or other specified) database.
8. Ensure that Kerberos is enabled for all services in the cluster. Custom Kerberos principals are not currently supported. For more information, see Enabling Kerberos for authentication.
9. Forward and reverse DNS must be working.
10. DNS lookups to sub-domains and the ML Workspace itself should work.

11. In DNS, wildcard subdomains (such as *.cml.yourcompany.com) must be set to resolve to the master domain (such as cml.yourcompany.com). The TLS certificate (if TLS is used) must also include the wildcard subdomains. When a session or job is started, an engine is created for it, and the engine is assigned to a random, unique subdomain.
12. The external load balancer server timeout needs to be set to 5 min. Without this, creating a project in an ML workspace with `git clone` or with the API may result in API timeout errors. For workarounds, see Known Issue DSE-11837.
13. If you intend to access a workspace over https, see Deploy an ML Workspace with Support for TLS.
14. For non-TLS ML workspaces, websockets need to be allowed for port 80 on the external load balancer.
15. Only a TLS-enabled custom Docker Registry is supported. Ensure that you use a TLS certificate to secure the custom Docker Registry. The TLS certificate can be self-signed, or signed by a private or public trusted Certificate Authority (CA).
16. On OpenShift, due to a Red Hat issue with OpenShift Container Platform 4.3.x, the image registry cluster operator configuration must be set to Managed.
17. Check if storage is set up in the cluster image registry operator. See Known Issues DSE-12778 for further information.

For more information on requirements, see CDP Private Cloud Base Installation Guide.

Hardware requirements

Storage

The cluster must have persistent storage classes defined for both block and filesystem volumeModes of storage. Ensure that a block storage class is set up. The exact amount of storage classified as block or filesystem storage depends on the specific workload used:

- Machine Learning workload requirements for storage largely depend on the nature of your machine learning jobs. 4 TB of persistent volume block storage is required per Machine Learning Workspace instance for storing different kinds of metadata related to workspace configuration. Additionally, Machine Learning requires access to NFS storage routable from all pods running in the cluster (see below).
- Monitoring uses a large Prometheus instance to scrape workloads. Disk usage depends on scale of workloads. Recommended volume size is 60 GB.

	Local Storage (for example, ext4)	Block PV (for example, Ceph or Portworx)	NFS (for ML user project files)
Control Plane	N/A	250 GB	N/A
CML	N/A	1.5 TB per workspace	1 TB per workspace (dependent on size of ML user files)

NFS

Cloudera Machine Learning (CML) requires NFS 4.0 for storing project files and folders. NFS storage is to be used only for storing project files and folders, and not for any other CML data, such as PostgreSQL database and LiveLog.

ECS requirements for NFS Storage

Cloudera managed ECS deploys and manages an internal NFS server based on LongHorn which can be used for CML. This is the recommended option for CML on ECS clusters. CML requires nfs-utils in order to mount longhorn-nfs provisioned mounts.

CML requires the nfs-utils package be installed in order to mount volumes provisioned by longhorn-nfs. The nfs-utils package is not available by default on every operating system. Check if nfs-utils is available, and ensure that it is present on all ECS cluster nodes.

Alternatively, the NFS server can be external to the cluster, such as a NetApp filer that is accessible from the private cloud cluster nodes.

OpenShift requirements for NFS storage

An internal user-space NFS server can be deployed into the cluster which serves a block storage device (persistent volume) managed by the cluster's software defined storage (SDS) system, such as Ceph or Portworx. This is the recommended option for CML on OpenShift. Alternatively, the NFS server can be external to the cluster, such as a NetApp filer that is accessible from the private cloud cluster nodes. NFS storage is to be used only for storing project files and folders, and not for any other CML data, such as PostgreSQL database and LiveLog.

CML does not support shared volumes, such as Portworx shared volumes, for storing project files. A read-write-once (RWO) persistent volume must be allocated to the internal NFS server (for example, NFS server provisioner) as the persistence layer. The NFS server uses the volume to dynamically provision read-write-many (RWX) NFS volumes for the CML clients.

Related Information

CDP Private Cloud Base Installation Guide
CDP Private Cloud Experiences Installation Software Requirements
CDP Private Cloud Experiences Installation Hardware Requirements
Known Issues and Limitations
Deploy an ML Workspace with Support for TLS
Using an External NFS Server

Cloudera Machine Learning requirements (ECS)

There are minimal requirements when using Cloudera Machine Learning (CML) on Embedded Container Service (ECS).

The primary requirement is to have 1.5 TB of storage space.

Note: The ECS installation wizard offers a one-time option to download CML Docker images. You cannot revisit this option; post install of the cluster.

For further information, see the Hardware and Software Requirements section in *Installation using the Embedded Container Service (ECS)*.

Related Information

Installation using the Embedded Container Service (ECS)

Get started with CML on Private Cloud

To get started as a user with Cloudera Machine Learning on your Private Cloud, follow the steps described below. They will show you how to set up a Project and work on some data.

Before you begin

Make sure the Admin creates a new Workspace for you. If you are an Admin, see: Provision an ML Workspace.

Note: Make sure that an Admin user logs into the Workspace first.

Procedure

1. Log in to your workspace. On the Workspaces tab, click Launch Workspace.
2. Next, create a Project. See: Creating a Project.
3. Once you have a Project, run a Session to start your work. See: Launch a Session.
4. Test your access to the base cluster (Data Lake). See: CDP-DC cluster connectivity test.

5. You can then run a Model. Learn about Models here: Creating and Deploying a Model.
6. When you are finished with your workspace, your Admin can remove it, as described here: Removing ML Workspaces.

Test your connectivity to the CDP-DC cluster

Test that you can create a Project in your ML Workspace and access data that is stored in the data center cluster.

Procedure

1. Create a new Project, using the PySpark template.
2. Create a new file called testdata.txt (use this exact filename).
3. Add 2-3 lines of any text in the file to serve as sample data.
4. Run the following Spark commands to test the connection.

```
from pyspark.sql import SparkSession

# Instantiate Spark-on-K8s Cluster
spark = SparkSession\
.builder\
.appName("Simple Spark Test")\
.config("spark.executor.memory","8g")\
.config("spark.executor.cores","2")\
.config("spark.driver.memory","2g")\
.config("spark.executor.instances","2")\
.getOrCreate()

# Validate Spark Connectivity
spark.sql("SHOW databases").show()
spark.sql('create table testcml (abc integer)').show()
spark.sql('insert into table testcml  select t.* from (select 1) t').show(
)
spark.sql('select * from testcml').show()
# Stop Spark Session
spark.stop()
```

5. Run the following direct HDFS commands to test the connection.

```
# Run sample HDFS commands
# Requires an additional testdata.txt file to be created with sample data
 in project home dir
!hdfs dfs -mkdir /tmp/testcml/
!hdfs dfs -copyFromLocal /home/cdsw/testdata.txt /tmp/testcml/
!hdfs dfs -cat /tmp/testcml/testdata.txt
```

What to do next

If you get errors, then check with your Admin to make sure that your user ID is set up in the Hadoop Authentication settings to access the CDP-DC cluster, and that the correct Ranger permissions have been applied.

Differences Between Public and Private Cloud

There are some differences in Cloudera Machine Learning functionality between Public and Private Cloud.

Feature	Public Cloud	Private Cloud 1.5.x
CML application control plane (infrastructure containers and workload containers)	Control plane is hosted on public cloud servers.	Control plane is hosted on customer's cluster.
Storage - CML internal state data (database, images, logs)	EBS on AWS, Azure Disks on Azure.	Software Defined Storage System, such as Ceph or Portworx.
Storage - User project files	EFS on AWS, external NFS on Azure.	Internal NFS storage is recommended.
Autoscaling	CPU/GPU nodes scale up and down as needed.	Autoscaling concept is different; Private Cloud shares a pooled set of resources among workloads.
Logging	Per-workspace diagnostic bundles can be downloaded from the workspace.	Diagnostic bundles are not supported at Workspace level, but can be downloaded from the control plane at the cluster level.
Monitoring dashboards	Provides four dashboards.	Provides two dashboards, for K8s Container and K8s Cluster.
NFS support	AWS uses EFS; Azure requires external NFS.	Internal NFS is recommended, external NFS is supported.
TLS support	TLS access to workspaces is supported.	TLS access is supported, but requires manual setup of certificate and other steps.
Hadoop Authentication	Uses FreeIPA	User needs to provide credentials to communicate with the CDP Private Base cluster.
Remote Access	Available from each workspace.	Not available in the workspace. Instead, the environment's kubeconfig file may be downloaded from Environments using the Download Kubernetes configuration action for the specified environment.
Roles	MLAdmin, MLUser	The corresponding roles are: EnvironmentAdmin, EnvironmentUser

Limitations on Private Cloud

There are some limitations to keep in mind when you are working with Cloudera Machine Learning on Private Cloud.

The following features are not yet supported in CML Private Cloud:

- Logging is limited, and diagnostic bundles for each workspace cannot be downloaded from the workspace UI. Instead, diagnostic bundles for the entire cluster can be downloaded from the control plane.
- Monitoring on Private Cloud does not support node-level resource metrics, hence only K8s Cluster and K8s Container dashboards are available.
- CML does not support the NVIDIA Multi-Instance GPU (MIG) feature.

Network File System (NFS)

A Network File System (NFS) is a protocol to access storage on a network that emulates accessing storage in a local file system. CML requires an NFS server for storing project files and folders, and the NFS export must be configured before you provision the first CML workspace in the cluster.

There are many different products or packages that can create an NFS in your private network. A Kubernetes cluster can host an internal NFS server, or an external NFS server can be installed on another cluster that is accessible by the private cloud cluster nodes. NFS storage is used only for storing project files and folders, and not for any other CML data, such as PostgreSQL database and livelog files.

CML does not support shared volumes, such as Portworx shared volumes, for storing project files. A read-write-once (RWO) persistent volume must be allocated to the internal NFS server (for example, NFS server provisioner) as the persistence layer. The NFS server uses the volume to dynamically provision read-write-many (RWX) NFS volumes for the CML clients.

An external NFS server option is currently the recommended option for Private Cloud production workloads. Not specifying an external NFS Server for your ML Workspace will use/require a deprecated internal NFS provisioner, which should only be used for small, proof-of-concept deployments. There are several options for setting up an internal NFS provisioner, described in the appendix. The Private Cloud Admin is responsible for setting up an NFS for use by your cluster.

Note: See *CDP Private Cloud Data Services Installation Software Requirements* for some information about installing NFS.

Related Information

CDP Private Cloud Experiences Installation Software Requirements

NFS Options for Private Cloud

Cloudera Machine Learning on Private Cloud requires a Network File System (NFS) server for storing project files and folders.

On ECS, NFS is part of the overall installation, and no additional setup steps are required.

You can use an NFS server that is external to the cluster, such as a NetApp Filer appliance. In this case, you must manually create a directory for each workspace. In this case, the NFS server must be configured before deploying the first CML workspace in the cluster. One important limitation is that CML does not support using shared volumes for storing project files.

Storage provisioner change On OCP, CephFS is used as the underlying storage provisioner for any new internal workspace on PVC 1.5.1. A storage class named "ocs-storagecluster-cephfs" with csi driver set to "openshift-storage.cephfs.csi.ceph.com" must exist in the cluster for new internal workspaces to get provisioned. Each workspace will have separate 1 TB internal storage.

On ECS, any new internal workspace on 1.5.1 will use Longhorn as the underlying storage provisioner. A storage class named "longhorn" with csi driver set to "driver.longhorn.io" must exist in the cluster for new internal workspaces to get provisioned. Each workspace will have separate 1 TB internal storage.

On either ECS or OCP, internal workspaces running on PVC 1.4.0/1.4.1 use NFS server provisioner as the storage provisioner. These workspaces when upgraded to 1.5.0 will continue to run with the same NFS server Provisioner. However, NFS server provisioner is now deprecated and is not supported in the 1.5.1 release.

Existing workspaces in 1.4.0/1.4.1 can be upgraded to 1.5.0 from PVC UI. After this, you can do one of the following:

- Migrate the 1.5.0 upgraded workspace from NFS server provisioner to Longhorn (ECS) / Cephfs (OCP) if you want to continue using the same workspace in PVC 1.5.1 as well
- Create a new 1.5.0 workspace and migrate the existing workloads to that before upgrading to the 1.5.1 release.

Note: There is no change in the underlying storage of external NFS backed workspaces and these can be simply upgraded to 1.5.1.

Internal Network File System on OCP

Learn about backing up and uninstalling an internal NFS server on OpenShift Container Platform.

Backing up Project Files and Folders

The block device backing the NFS server data must be backed up to protect the CML project files and folders. The backup mechanism would vary depending on the underlying block storage system and backup policies in place.

1. identify the underlying block storage to backup, first determine the NFS PV:

```
$ echo `kubectl get pvc -n cml-nfs -o jsonpath='{.items[0].spec.volumeNa
me}'`
       pvc-bec1de27-753d-11ea-a287-4cd98f578292
```

2. For Ceph, the RBD volume/image name is the name of the dynamically created persistent volume (pvc-3d3316b6-6cc7-11ea-828e-1418774847a1).

Ensure this volume is backed up using an appropriate backup policy. The Backup/Restore feature is not yet supported for workspaces with an internal NFS.

Uninstalling the NFS server on OpenShift

Uninstall the NFS server provisioner using either of the following commands.

Use this command if the NFS server provisioner was installed using oc and yaml files:

```
$ oc delete scc nfs-scc
$ oc delete clusterrole cml-nfs-nfs-server-provisioner
$ oc delete clusterrolebinding cml-nfs-nfs-server-provisioner
$ oc delete namespace cml-nfs
```

Use this command if the NFS server provisioner was installed using `Helm`:

```
$ helm tiller run cml-nfs -- helm delete cml-nfs --purge
$ oc delete scc nfs-scc securitycontextconstraints.security.openshift.io "n
fs-scc" deleted
```

Storage provisioner change

In CDP 1.5.0 on OCP, CML has changed the underlying storage provisioner for internal workspaces.

Any new internal workspace on 1.5.0 will use CephFS as the underlying storage provisioner. A storage class named "ocs-storagecluster-cephfs" with csi driver set to "openshift-storage.cephfs.csi.ceph.com" must exist in the cluster for new internal workspaces to get provisioned. Each workspace will have separate 1 TB internal storage.

Internal workspaces running on PVC 1.4.0/1 use NFS server provisioner as the storage provisioner. These workspaces when upgraded to 1.5.0 will continue to run with NFS Provisioner. However, NFS server provisioner is deprecated now and will not be supported in 1.5.1 release. So, customers are expected to migrate their 1.5.0 upgraded workspace from NFS server provisioner to CephFS if they want to continue using the same workspace in 1.5.1 as well. If not, then customer should create a new 1.5.0 workspace and migrate their existing workloads to that before 1.5.1 release.

Note: There is no change in the underlying storage of external NFS backed workspaces and these can be simply upgraded to 1.5.0.

Portworx storage provisioner

CML now supports Portworx as storage backend for workspaces. New CML workspaces can be configured to use Portworx volumes for storing data.

Steps:

1. Create a Portworx storage class in your OCP cluster manually. This storage class should have the provisioner field set to either `pxd.portworx.com` or `kubernetes.io/portworx-volume`.

   Note: Although `kubernetes.io/portworx-volume` will work, this provisioner is deprecated in upstream Kubernetes and might be removed in the future (See: Volumes). So, `pxd.portworx.com` is the preferred provisioner to use. Also the storage class created must have the parameter `allow_all_ips: "true"` (or have parameter `allow_ips` set to the Kubernetes node in CIDR notation, such as `allow_ips: 10.17.0.0/16`).

   Sample storage classes are:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: sample-portworx-storage-class-k8s
parameters:
repl: "3"
allow_all_ips: "true"
provisioner: kubernetes.io/portworx-volume
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: Immediate

---


apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: sample-portworx-storage-class-pxd
parameters:
repl: "2"
sharedv4: "true"
sharedv4_svc_type: ClusterIP
allow_all_ips: "true"
provisioner: pxd.portworx.com
reclaimPolicy: Retain
allowVolumeExpansion: true
volumeBindingMode: Immediate

---

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: sample-portworx-storage-class-pxd
parameters:
repl: "2"
sharedv4: "true"
sharedv4_svc_type: ClusterIP
allow_ips: 10.17.0.0/16
provisioner: pxd.portworx.com
reclaimPolicy: Retain
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

2. In the CML workspace provisioning page, enable Set Custom NFS Storage Class Name". Enter the name of storage class created manually above into the Storage Class Name text box.
3. Enter the rest of the workspace details and submit. This will create a Portworx-backed CML workspace.

Related Information

Using an External NFS Server

Internal Network File System on ECS

On ECS, NFS is part of the overall installation, and no additional setup steps are required.

The internal NFS does not have a backup feature.

Storage provisioner change

On ECS, Longhorn is used as the underlying storage provisioner for any new internal workspace on CDP 1.5.0. A storage class named longhorn with csi driver set to driver.longhorn.io must exist in the cluster for new internal workspaces to get provisioned. Each workspace will have separate 1 TB internal storage.

Internal workspaces running on CDP 1.4.0 and 1.4.1 use the NFS server provisioner as the storage provisioner. These workspaces when upgraded to 1.5.0 will continue to run with NFS Provisioner. However, NFS server provisioner is deprecated now and will not be supported in the 1.5.1 release. So, customers are expected to migrate their 1.5.0 upgraded workspace from NFS server provisioner to Longhorn if they want to continue using the same workspace in 1.5.1 as well. If not, then customers should create a new 1.5.0 workspace and migrate their existing workloads to that before the 1.5.1 release.

- **Note:** There is no change in the underlying storage of external NFS backed workspaces and these can be simply upgraded to 1.5.0.

Using an External NFS Server

You can install an NFS server that is external to the cluster.

About this task

Currently, NFS version 4.1 is the recommended protocol to use for CML. The NFS client within CML must be able to mount the NFS storage with default options, and also assumes these export options:

```
rw,sync,no_root_squash,no_all_squash,no_subtree_check
```

Before you begin

Before creating a CML workspace, the storage administrator must create a directory that will be exported to the cluster for storing ML project files for that workspace. Either a dedicated NFS export path, or a subdirectory in an existing export must be specified for each workspace.

Each CML workspace needs a unique directory that does not have files in it from a different or previous workspace. For example, if 10 CML workspaces are expected, the storage administrator will need to create 10 unique directories. Either one NFS export and 10 subdirectories within it need to be created, or 10 unique exports need to be created.

For example, to use a dedicated NFS share for a workspace named “workspace1” from NFS server “nfs_server”, do the following:

Procedure

1. Create NFS export directory “/workspace1”.
2. Change ownership for the exported directory
   a) CML accesses this directory as a user with a UID and GID of 8536. Therefore, run chown 8536:8536 /workspace1
   b) Make the export directory group-writeable and set the GID:
   chmod g+srwx /workspace1
3. Provide the NFS export path nfs_server:/workspace1 when prompted by the CML Control Plane App while creating the workspace.

4. To use a subdirectory in an existing NFS share, say nfs_server:/export, do the following:
   a) Create a subdirectory /export/workspace1
   b) Change ownership: chown 8536:8536 /export/workspace1
   c) Set GID and make directory group writeable: chmod g+srwx /export/workspace1
   d) Provide the export path nfs_server:/export/workspace1 when prompted by the CML Control Plane App.

NFS share sizing

CML workloads are sensitive to latency and IO/s instead of throughput.

The minimum recommended file share size is 100 GB. The file share must support online volume capacity expansion. It must provide at least the following performance characteristics:

IO / s	3100
Throughput rate	110.0 MiBytes/s

Deploy an ML Workspace with Support for TLS

You can provision an ML workspace with TLS enabled, so that it can be accessed via https.

Before you begin

You need to obtain a certificate from the Certificate Authority used by your organization. This may be an internal certificate authority.

Additionally, you need a computer with CLI access to the cluster, and with kubectl installed.

Note:

When provisioning CML, we recommend that the ML workspace be given a static subdomain name. This subdomain is used in the URL for the ML Workspace. A workspace domain is structured as https://<workspace-subdomain>.<cluster>.<company>.com. Workloads created in an ML workspace are containers provisioned in kubernetes and must be addressable to the user. To do this, CML creates a unique subdomain. The URL for the workload is structured as https://<workload-subdomain>.<workspace-subdomain>.<cluster>.<company>.com. Because the workload subdomain is randomly generated, for TLS to work, an ML workspace needs to have a wildcard SAN entry in the TLS certificate and its formed like SAN:*.<workspace-subdomain>.<cluster>.<company>.com. By using unique subdomains, the ML Workspace is able to securely serve each interactive workload with proper isolation and protect it from code injection attacks such as Cross Site Scripting.

Procedure

1. Provision the ML Workspace. Follow the procedure Provisioning ML Workspaces.

Note: Ensure you select Enable TLS.

2. Obtain the .crt and .key files for the certificate from your Certificate Authority.

   The certificate URL is generally of the form: <workspaceid>.<cluster>.<domain>.com. Assuming an example URL for the certificate of ml-30b43418-53c.cluster.yourcompany.com, check that the certificate correctly shows the corresponding Common Name (CN) and Subject Alternative Names (SAN):

   - CN: ml-30b43418-53c.cluster.yourcompany.com
   - SAN: *.ml-30b43418-53c.cluster.yourcompany.com
   - SAN: ml-30b43418-53c.cluster.yourcompany.com

   Note: If you want to install a new signed certificate, you must regenerate a new certificate from your Certificate Authority. It is impossible to secure multi-level subdomains with a single wildcard certificate. If a wildcard certificate is issued for *.mydomain.tld, so that it can secure only the first-level subdomains of *.mydomain.com, then you will need another wildcard certificate for *.sub1.mydomain.tld.

3. Create a Kubernetes secret inside the previously provisioned ML workspace namespace, and name the secret cml-tls-secret.

   On a machine with access to the .srt and .key files above, and access to the OpenShift cluster, run this command: kubectl create secret tls cml-tls-secret --cert=<pathtocrt.crt> --key=<pathtokey.key> -o yaml --dry-run | kubectl -n <cml-workspace-namespace> create -f -

   You can replace or update certificates in the secret at any time.

4. In Admin Security Root CA configuration , add the root CA certificate to the workspace.

   For example: https://ml-def88113-acd.cluster.yourcompany.com/administration/security"

Results

The command creates routes to reflect the new state of ingress and secret, and enables TLS.

Replace a Certificate

You can replace a certificate in a deployed namespace.

About this task

Procedure

1. Obtain the new certificate .crt and .key files.
2. Run this command (example): kubectl create secret tls cml-tls-secret --cert=<pathtocrt.crt> --key=<pathtokey.key> -o yaml --dry-run | kubectl -n <cml-workspace-namespace> replace -f -

What to do next

The certificate of an existing session does not get renewed. The new certificate only applies to newly created sessions.

Deploy an ML Workspace with Support for TLS on ECS

On ECS, you can provision an ML workspace with TLS enabled, so that the workspace is accessible via https.

About this task

You need to obtain a certificate from the Certificate Authority used by your organization. This may be an internal certificate authority. Additionally, you need a computer with CLI access to the cluster, and with `kubectl` installed.

Note:

When provisioning CML, we recommend that the ML workspace be given a static subdomain name. This subdomain is used in the URL for the ML Workspace. A workspace domain is structured as https://<workspace-subdomain>.<cluster>.<company>.com. Workloads created in an ML workspace are containers provisioned in kubernetes and must be addressable to the user. To do this, CML creates a unique subdomain. The URL for the workload is structured as https://<workload-subdomain>.<workspace-subdomain>.<cluster>.<company>.com. Because the workload subdomain is randomly generated, for TLS to work, an ML workspace needs to have a wildcard SAN entry in the TLS certificate and its formed like SAN:*.<workspace-subdomain>.<cluster>.<company>.com. By using unique subdomains, the ML Workspace is able to securely serve each interactive workload with proper isolation and protect it from code injection attacks such as Cross Site Scripting.

Procedure

1. Provision an ML workspace. See *Provision an ML Workspace* for more information.

   Note: Ensure you select Enable TLS.

2. Obtain the .crt and .key files for the certificate from your Certificate Authority.

   The certificate URL is generally of the form: <workspaceid>.<cluster>.<domain>.com. Assuming an example URL for the certificate of ml-30b43418-53c.apps.cluster.yourcompany.com, check that the certificate correctly shows the corresponding Common Name (CN) and Subject Alternative Names (SAN):

   - CN: ml-30b43418-53c.apps.cluster.yourcompany.com
   - SAN: *.ml-30b43418-53c.apps.cluster.yourcompany.com
   - SAN: ml-30b43418-53c.apps.cluster.yourcompany.com

3. Create or replace a Kubernetes secret inside the previously provisioned ML workspace namespace, then automatically upload the certificate.

   Login to the Ecs Server role host and execute the following commands to accomplish these steps:

   a) cd /opt/cloudera/parcels/ECS/bin/

   b) ./cml_utils.sh -h

      Optional: A helper prompt appears, with explanation for the next command.

   c) ./cml_utils.sh upload-cert -n <namespace> -c <path_to_cert> -k <path_to_key>

      For example: ./cml_utils.sh upload-cert -n bb-tls-1 -c /tmp/ws-cert.crt -k /tmp/ws-key.key

      Note: To find the <namespace> of the workspace, go to the Machine Learning Workspaces UI, and in the Actions menu for the workspace, select View Workspace Details. Namespace is shown on the Details tab.

4. In Site Administration Security Root CA configuration , add the root CA certificate to the workspace.

   For example: https://ml-def88113-acd.apps.nf-01.os4cluster.yourcompany.com/administration/security

GPU node setup

In Kubernetes, you can taint nodes to affect how the node is scheduled. You can ensure that nodes that have a GPU are reserved exclusively for CML workloads that require a GPU.

To reserve a GPU node, assign a taint to the node.

Openshift

On Openshift, specify the node taint nvidia.com/gpu: true:NoSchedule for any nodes that host GPUs and are required to be used only for GPU workloads.

ECS

On ECS, set the node taint nvidia.com/gpu: true:NoSchedule in one of the following three ways:

1. During ECS installation: After adding the GPU host(s) to Cloudera Manager but prior to creation of the ECS cluster, visit the Host Configuration page, select the Dedicated GPU Node for Data Services checkbox and Save the configuration. Repeat for all hosts on which the taint is desired. Then, proceed with installation via the Add Cluster wizard.
2. During ECS upgrade: After upgrading Cloudera Manager (if applicable), set the host configuration as described above on one or more hosts in the ECS cluster. Then, proceed with upgrade via the Upgrade Cluster wizard.
3. Independently of ECS install or upgrade: Set the host configuration as described above on one or more hosts in the ECS cluster. Redeploy the client configuration on the ECS cluster. Finally, run the Reapply All Settings to Cluster command on the ECS service, which can be found in the Service Actions menu.