Machine Learning 1.5.0

ML Workspaces (Private Cloud)

Date published: 2020-07-16
Date modified: 2023-01-31

https://docs.cloudera.com/

Legal Notice

Contents

Provision an ML Workspace

In CML on Private Cloud, the ML Workspace is where data scientists get their work done. After your Admin has created or given you access to an environment, you can set up a workspace.

Before you begin

The first user to access the ML workspace after it is created must have the EnvironmentAdmin role assigned.

Procedure

1. Log in to the CDP Private Cloud web interface using your corporate credentials or other credentials that you received from your CDP administrator.
2. Click ML Workspaces.
3. Click Provision Workspace. The Provision Workspace panel displays.
4. In Provision Workspace, fill out the following fields.
   a) Workspace Name - Give the ML workspace a name. For example, test-cml. Do not use capital letters in the workspace name.
   b) Select Environment - From the dropdown, select the environment where the ML workspace must be provisioned. If you do not have any environments available to you in the dropdown, contact your CDP admin to gain access.
   c) Namespace - Enter the namespace to use for the ML workspace.
   d) NFS Server - Select Internal to use an NFS server that is integrated into the Kubernetes cluster. This is the recommended selection at this time.

      The path to the internal NFS server is already set in the environment.
5. In Production Machine Learning, select to enable the following features.
   a) Enable Governance - Enables advanced lineage and governance features.

      Governance Principal Name - If Enable Governance is selected, you can use the default value of mlgov, or enter an alternative name. The alternative name must be present in your environment and be given permissions in Ranger to allow the MLGovernance service deliver events to Atlas.
   b) Enable Model Metrics - Enables exporting metrics for models to a PostgreSQL database.
6. In Other Settings, select to enable the following features.
   a) Enable TLS - Select this to enable https access to the workspace.
   b) Enable Monitoring - Administrators (users with the EnvironmentAdmin role) can use a Grafana dashboard to monitor resource usage in the provisioned workspace.
   c) CML Static Subdomain - This is a custom name for the workspace endpoint, and it is also used for the URLs of models, applications, and experiments. Only one workspace with the specific subdomain endpoint name can be running at a time. You can create a wildcard certificate for this endpoint in advance. The workspace name has this format: <static subdomain name>.<environment name>.<workload subdomain>.<base do main></cmd>

      

      Note: The endpoint name can have a maximum of 15 characters, using alphanumerics and hyphen or underscore only, and must start and end with an alphanumeric character.
7. Click Provision Workspace. The new workspace provisioning process takes several minutes.

What to do next

After the workspace is provisioned, you can log in by clicking the workspace name on the Machine Learning Workspaces page. The first user to log in must be the administrator.

Related Information

Monitoring ML Workspaces

Removing ML Workspaces

Monitoring ML Workspaces

This topic shows you how to monitor resource usage on your ML workspaces.

About this task

Cloudera Machine Learning leverages Prometheus and Grafana to provide a dashboard that allows you to monitor how CPU, memory, storage, and other resources are being consumed by ML workspaces. Prometheus is an internal data source that is auto-populated with resource consumption data for each workspace. Grafana is a monitoring dashboard that allows you to create visualizations for resource consumption data from Prometheus.

Each ML workspace has its own Grafana dashboard.

Before you begin

Required Role: MLAdmin

You need the MLAdmin role to view the Workspace details page.

Note: On Private Cloud, the corresponding role is EnvironmentAdmin.

Procedure

1. Log in to the CDP web interface.
2. Click ML Workspaces.
3. For the workspace you want to monitor, click Actions Open Grafana .

Results

CML provides you with several default Grafana dashboards:

- K8s Cluster: Shows cluster health, deployments, and pods
- K8s Containers: Shows pod info, cpu and memory usage
- K8s Node: Shows node cpu and memory usage, disk usage and network conditions
- Models: Shows response times, requests per second, cpu and memory usage for model replicas.

You might choose to add new dashboards or create more panels for other metrics. For more information, see the *Grafana documentation*.

Related Information

Monitoring and Alerts

Removing ML Workspaces

This topic describes how to remove an existing ML workspace and clean up any cloud resources associated with the workspace. Currently, only CDP users with both the MLAdmin role and the EnvironmentAdmin account role can remove workspaces.

Procedure

1. Log in to the CDP web interface.
2. Click ML Workspaces.

3. Click on the Actions icon and select Remove Workspace.
   a) Force Delete - This property is not required by default. You should first attempt to remove your workspace with this property disabled.

      Enabling this property deletes the workspace from CDP but does not guarantee that the underlying kubernetes resources used by the workspace are cleaned up properly. Go to you kuberknetes administration console to make sure that the resources have been successfully deleted.
4. Click OK to confirm.

How to upgrade CML workspaces (ECS)

When you upgrade from Private Cloud version 1.4.1 to version 1.5.0, you need to manually upgrade ML workspaces that are running on ECS using internal NFS.

In ECS Private Cloud 1.5.0, the internal NFS implementation is changed from using an NFS provisioner for each workspace, to using a Longhorn Native RWX Volume.

On either ECS or OCP, internal workspaces on PVC 1.4.0/1.4.1 use the NFS server provisioner as a storage provisioner. This server provisioner still works in 1.5.0, however, it is deprecated, and will be removed in 1.5.1.

Existing workspaces in 1.4.1 need to be upgraded to 1.5.0. These workspaces use the older storage provisioner. You can do one of the following:

- Migrate the workspace to Longhorn before 1.5.1 is released, or:
- Create a new 1.5.0 workspace, and migrate the workloads to that workspace now.

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

The manual steps mentioned in this guide are required if an existing workspace backed by internal NFS (which was created on PVC 1.4.1 or below) needs to be migrated to Longhorn RWX.

1. Update ECS PVC to version 1.5.0.
2. Each existing ML workspace can now be upgraded, although this is optional. If you want to continue using your existing workspaces without upgrading them, then this procedure is not required. This is true for all existing workspaces (both internal and external NFS).
3. If you want to upgrade a workspace, then first determine whether the workspace is backed by internal or external NFS.
   a. If the existing workspace is backed by external NFS, you can simply upgrade the workspace from the UI. There is no need to follow the rest of this procedure.
   b. If the existing workspace is backed by internal NFS, then please follow this procedure to migrate to Longhorn RWX after the workspace upgrade.
4. Upgrade the workspace from CML UI.
5. Get the Kubeconfig for your Private Cloud cluster.
6. Try to suspend the workspace manually so that there are no read/write operations happening to the underlying NFS. Stop all your running workloads - sessions, jobs, application, deployments and so forth. Also, scale down ds-vfs and s2i-client deployments with these commands:
   a. kubectl scale -n <workspace-namespace> --replicas=0 deployment ds-vfs
   b. kubectl scale -n <workspace-namespace> --replicas=0 deployment s2i-client
7. Create a backup volume for the upgrade process. The backup can either be taken in the cluster itself or it can also be taken outside in an external NFS. Based on what you want, go ahead with either step a. or b. below.

Substitute your workspace details where indicated with angle brackets. Start by creating a backup.yaml file. Add the following content to the file and run it using the command: `kubectl apply -f ./backup.yaml`

a. Internal backup:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: projects-pvc-backup
  namespace: <existing-workspace-namespace>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 500Gi
  storageClassName: longhorn
```

b. External backup:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: projects-pvc-backup
spec:
  capacity:
    storage: 500Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  mountOptions:
    - nfsvers=3
  nfs:
    server: <your-external-nfs-address>
    path: <your-external-nfs-export-path>
  volumeMode: Filesystem

---

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: projects-pvc-backup
  namespace: <existing-workspace-namespace>
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 500Gi
  storageClassName: ""
  volumeName: projects-pvc-backup
  volumeMode: Filesystem
```

8. Now, create a migrate.yaml file. Add the following content to the file. With the following Kubernetes job, create a backup of the existing workspace's NFS data to the volume that was created in the previous step. Run the job using the command: `kubectl apply -f ./migrate.yaml`

```
apiVersion: batch/v1
kind: Job
metadata:
  namespace: <existing-workspace-namespace>
  name: projects-pvc-backup
```

```
spec:
  completions: 1
  parallelism: 1
  backoffLimit: 10
  template:
    metadata:
      name: projects-pvc-backup
      labels:
        name: projects-pvc-backup
    spec:
      restartPolicy: Never
      containers:
        - name: projects-pvc-backup
          image: docker-private.infra.cloudera.com/cloudera_base/ubi8/c
ldr-ubi-minimal:8.6-751-fips-03062022
          tty: true
          command: [ "/bin/sh" ]
          args: [  "-c", "microdnf install rsync && rsync -P -a /mnt/old/
 /mnt/new && chown -R 8536:8536 /mnt/new;" ]
          volumeMounts:
            - name: old-vol
              mountPath: /mnt/old
            - name: new-vol
              mountPath: /mnt/new
      volumes:
        - name: old-vol
          persistentVolumeClaim:
            claimName: projects-pvc
        - name: new-vol
          persistentVolumeClaim:
            claimName: projects-pvc-backup
```

9. Monitor the previous job for completion. Logs can be retrieved using:

```
kubectl logs -n <workspace-namespace> -l job-name=projects-pvc-backup
```

You can check for job completion with:

```
kubectl get jobs -n <workspace-namespace> -l job-name=projects-pvc-backup
```

Once the job completes, move on to the next step.

10. Now delete the existing NFS volume for the workspace.

```
kubectl delete pvc -n <workspace-namespace> projects-pvc
kubectl patch pvc -n <workspace-namespace> projects-pvc -p '{"metadata":
{"finalizers":null}}'
```

11. Perform the following steps to modify underlying NFS from NFS provisioner to Longhorn RWX.

a. Get the release name for the workspace, using: `helm list -n <workspace-namespace>`. For example, in this case `mlx-workspace1` is the `release-name`.

```
helm list -n workspace1
WARNING: Kubernetes configuration file is group-readable. This is insecu
re. Location: ./../piyushecs
WARNING: Kubernetes configuration file is world-readable. This is insec
ure. Location: ./../piyushecs
NAME           NAMESPACE  REVISION UPDATED
   STATUS   CHART                    APP VERSION
mlx-workspace1 workspace1 4        2023-01-04 08:07:47.075343142 +0000
 UTC deployed cdsw-combined-2.0.35-b93
```

b. Save the existing Helm values.

```
helm get values <release-name> -n <workspace-namespace> -o yaml > old.ya
ml
```

c. Modify the `ProjectsPVCStorageClassName` in the old.yaml file to longhorn and add ProjectsPVCSize: 1Ti. For example. ProjectsPVCStorageClassName: longhorn-nfs-sc-workspace1 should be changed to Projects PVCStorageClassName: longhorn Also, add this to the file: ProjectsPVCSize: 1Ti

d. Get the GitSHA from old.yaml: `grep GitSHA old.yaml` For example: GitSHA: 2.0.35-b93

e. Get the release chart `cdsw-combined-<GitSHA>.tgz` This is available in dp-mlx-control-plane-app pod in the namespace at folder /app/service/resources/mlx-deploy/ Contact Cloudera support to download the chart if needed.

f. Delete the jobs and stateful sets (these are recreated after the helm install)

```
kubectl --namespace <workspace-namespace> delete jobs  --all
```

```
kubectl --namespace <workspace-namespace> delete statefulsets  --all
```

g. Do a Helm upgrade to the same release.

```
helm upgrade <release-name> <path to release chart (step e)> --install -
f ./old.yaml --wait  --namespace <workspace-namespace> --debug --timeout
 1800s
```

12. Scale down the ds-vfs and s2i-client deployments with the commands:

```
kubectl scale -n <workspace-namespace> --replicas=0 deployment ds-vfs
```

```
kubectl scale -n <workspace-namespace> --replicas=0 deployment s2i-client
```

13. Copy the data from the backup into this upgraded workspace. In order to do this, create a migrate2.yaml file. Add the following content to the file. Run the job using the command `kubectl apply -f ./migrate2.yaml`

```
apiVersion: batch/v1
kind: Job
metadata:
  namespace: <existing-workspace-namespace>
  name: projects-pvc-backup2
spec:
  completions: 1
  parallelism: 1
  backoffLimit: 10
  template:
    metadata:
      name: projects-pvc-backup2
      labels:
        name: projects-pvc-backup2
    spec:
      restartPolicy: Never
      containers:
        - name: projects-pvc-backup2
          image: docker-private.infra.cloudera.com/cloudera_base/ubi8/c
ldr-ubi-minimal:8.6-751-fips-03062022
          tty: true
          command: [ "/bin/sh" ]
          args: [ "-c", "microdnf install rsync && rsync -P -a /mnt/old/ /
mnt/new && chown -R 8536:8536 /mnt/new;" ]
          volumeMounts:
            - name: old-vol
              mountPath: /mnt/old
```

```
            - name: new-vol
              mountPath: /mnt/new
      volumes:
        - name: old-vol
          persistentVolumeClaim:
            claimName: projects-pvc-backup
        - name: new-vol
          persistentVolumeClaim:
            claimName: projects-pvc
```

14. Monitor the job above for completion. Logs can be retrieved using:

```
kubectl logs -n <workspace-namespace> -l job-name=projects-pvc-backup2
```

You can check for job completion with:

```
kubectl get jobs -n <workspace-namespace> -l job-name=projects-pvc-backup2
```

Once the job completes, move on to the next step.

15. After the above job is completed, scale up `ds-vfs` and `s2i-client` using the command:

```
kubectl scale -n <workspace-namespace> --replicas=1 deployment ds-vfs
```

and

```
kubectl scale -n <workspace-namespace> --replicas=1 deployment s2i-client
```

16. The upgraded workspace is ready to use. In case you want to delete the backup, then delete the existing backup volume for the workspace using these commands:

```
kubectl delete pvc -n <workspace-namespace> projects-pvc-backup
kubectl patch pvc -n <workspace-namespace> projects-pvc-backup -p '{"met
adata":{"finalizers":null}}'
```

Note: Taking backup of the existing workspace will take additional space on either PVC cluster (internal backup) or external NFS storage (external backup). So, customers can clear this backup once their workspace is properly migrated.

How to upgrade CML workspaces (OCP)

When you upgrade from Private Cloud version 1.4.1 to version 1.5.0, you need to manually upgrade ML workspaces that are running on OCP using internal NFS.

In OCP Private Cloud 1.5.0, the internal NFS implementation is changed from using an NFS provisioner for each workspace, to using a CephFS Volume.

On either ECS or OCP, internal workspaces on PVC 1.4.0/1.4.1 use the NFS server provisioner as a storage provisioner. This server provisioner still works in 1.5.0, however, it is deprecated, and will be removed in 1.5.1.

Existing workspaces in 1.4.1 need to be upgraded to 1.5.0. These workspaces use the older storage provisioner. You can do one of the following:

- Migrate the workspace to CephFS before 1.5.1 is released, or:
- Create a new 1.5.0 workspace, and migrate the workloads to that workspace now.

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

The manual steps mentioned in this guide are required if an existing workspace backed by internal NFS (which was created on Private Cloud 1.4.1 or below) needs to be migrated to Longhorn RWX.

1. Update OCP Private Cloud to version 1.5.0.
2. Each existing ML workspace can now be upgraded, although this is optional. If you want to continue using your existing workspaces without upgrading them, then this procedure is not required. This is true for all existing workspaces (both internal and external NFS).
3. If you want to upgrade a workspace, then first determine whether the workspace is backed by internal or external NFS.
   a. If the existing workspace is backed by external NFS, you can simply upgrade the workspace from the UI. There is no need to follow the rest of this procedure.
   b. If the existing workspace is backed by internal NFS, then please follow this procedure to migrate to CephFS after the workspace upgrade.
4. Upgrade the workspace from CML UI.
5. Get the Kubeconfig for your Private Cloud cluster.
6. Try to suspend the workspace manually so that there are no read/write operations happening to the underlying NFS. Stop all your running workloads - sessions, jobs, application, deployments and so forth. Also, scale down ds-vfs and s2i-client deployments with these commands:
   a. kubectl scale -n <workspace-namespace> --replicas=0 deployment ds-vfs
   b. kubectl scale -n <workspace-namespace> --replicas=0 deployment s2i-client
7. Create a backup volume for the upgrade process. The backup can either be taken in the cluster itself or it can also be taken outside in an external NFS. Based on what you want, go ahead with either step a. or b. below. Substitute your workspace details where indicated with angle brackets. Start by creating a backup.yaml file. Add the following content to the file and run it using the command: `kubectl apply -f ./backup.yaml`
   a. Internal backup:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: projects-pvc-backup
  namespace: <existing-workspace-namespace>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1 Ti
  storageClassName: longhorn
```

   b. External backup:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: projects-pvc-backup
spec:
  capacity:
    storage: 1 Ti
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  mountOptions:
    - nfsvers=3
  nfs:
    server: <your-external-nfs-address>
    path: <your-external-nfs-export-path>
  volumeMode: Filesystem
```

```
---

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: projects-pvc-backup
  namespace: <existing-workspace-namespace>
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1 Ti
  storageClassName: ""
  volumeName: projects-pvc-backup
  volumeMode: Filesystem
```

8. Now, create a migrate.yaml file. Add the following content to the file. With the following Kubernetes job, create a backup of the existing workspace's NFS data to the volume that was created in the previous step. Run the job using the command: `kubectl apply -f ./migrate.yaml`

```
apiVersion: batch/v1
kind: Job
metadata:
  namespace: <existing-workspace-namespace>
  name: projects-pvc-backup
spec:
  completions: 1
  parallelism: 1
  backoffLimit: 10
  template:
    metadata:
      name: projects-pvc-backup
      labels:
        name: projects-pvc-backup
    spec:
      restartPolicy: Never
      containers:
        - name: projects-pvc-backup
          image: docker-private.infra.cloudera.com/cloudera_base/ubi8/c
ldr-ubi-minimal:8.6-751-fips-03062022
          tty: true
          command: [ "/bin/sh" ]
          args: [  "-c", "microdnf install rsync && rsync -P -a /mnt/old/
 /mnt/new && chown -R 8536:8536 /mnt/new;" ]
          volumeMounts:
            - name: old-vol
              mountPath: /mnt/old
            - name: new-vol
              mountPath: /mnt/new
      volumes:
        - name: old-vol
          persistentVolumeClaim:
            claimName: projects-pvc
        - name: new-vol
          persistentVolumeClaim:
            claimName: projects-pvc-backup
```

9. Monitor the previous job for completion. Logs can be retrieved using:

```
kubectl logs -n <workspace-namespace> -l job-name=projects-pvc-backup
```

You can check for job completion with:

```
kubectl get jobs -n <workspace-namespace> -l job-name=projects-pvc-backup
```

Once the job completes, move on to the next step.

10. Now delete the existing NFS volume for the workspace.

```
kubectl delete pvc -n <workspace-namespace> projects-pvc
kubectl patch pvc -n <workspace-namespace> projects-pvc -p '{"metadata":
{"finalizers":null}}'
```

11. Perform the following steps to modify underlying NFS from NFS provisioner to Longhorn RWX.

a. Get the release name for the workspace, using: `helm list -n <workspace-namespace>`. For example, in this case `mlx-workspace1` is the `release-name`.

```
helm list -n workspace1
WARNING: Kubernetes configuration file is group-readable. This is insecu
re. Location: ./../piyushecs
WARNING: Kubernetes configuration file is world-readable. This is insec
ure. Location: ./../piyushecs
NAME           NAMESPACE  REVISION UPDATED
   STATUS   CHART                    APP VERSION
mlx-workspace1 workspace1 4        2023-01-04 08:07:47.075343142 +0000
 UTC deployed cdsw-combined-2.0.35-b93
```

b. Save the existing Helm values.

```
helm get values <release-name> -n <workspace-namespace> -o yaml > old.ya
ml
```

c. Modify the `ProjectsPVCStorageClassName` in the old.yaml file to longhorn and add ProjectsPVCSize: 1Ti. For example. ProjectsPVCStorageClassName: longhorn-nfs-sc-workspace1 should be changed to Projects PVCStorageClassName: ocs-storagecluster-cephfs Also, add this to the file: ProjectsPVCSize: 1Ti

d. Get the GitSHA from old.yaml: `grep GitSHA old.yaml` For example: GitSHA: 2.0.35-b93

e. Get the release chart `cdsw-combined-<GitSHA>.tgz` This is available in dp-mlx-control-plane-app pod in the namespace at folder /app/service/resources/mlx-deploy/ Contact Cloudera support to download the chart if needed.

f. Delete the jobs and stateful sets (these are recreated after the helm install)

```
kubectl --namespace <workspace-namespace> delete jobs  --all
```

```
kubectl --namespace <workspace-namespace> delete statefulsets  --all
```

g. Do a Helm upgrade to the same release.

```
helm upgrade <release-name> <path to release chart (step e)> --install -
f ./old.yaml --wait  --namespace <workspace-namespace> --debug --timeout
 1800s
```

12. Scale down the ds-vfs and s2i-client deployments with the commands:

```
kubectl scale -n <workspace-namespace> --replicas=0 deployment ds-vfs
```

```
kubectl scale -n <workspace-namespace> --replicas=0 deployment s2i-client
```

13. Copy the data from the backup into this upgraded workspace. In order to do this, create a migrate2.yaml file. Add the following content to the file. Run the job using the command `kubectl apply -f ./migrate2.yaml`

```
apiVersion: batch/v1
kind: Job
metadata:
  namespace: <existing-workspace-namespace>
  name: projects-pvc-backup2
spec:
  completions: 1
  parallelism: 1
  backoffLimit: 10
  template:
    metadata:
      name: projects-pvc-backup2
      labels:
        name: projects-pvc-backup2
    spec:
      restartPolicy: Never
      containers:
        - name: projects-pvc-backup2
          image: docker-private.infra.cloudera.com/cloudera_base/ubi8/c
ldr-ubi-minimal:8.6-751-fips-03062022
          tty: true
          command: [ "/bin/sh" ]
          args: [ "-c", "microdnf install rsync && rsync -P -a /mnt/old/ /
mnt/new && chown -R 8536:8536 /mnt/new;" ]
          volumeMounts:
            - name: old-vol
              mountPath: /mnt/old
            - name: new-vol
              mountPath: /mnt/new
      volumes:
        - name: old-vol
          persistentVolumeClaim:
            claimName: projects-pvc-backup
        - name: new-vol
          persistentVolumeClaim:
            claimName: projects-pvc
```

14. Monitor the job above for completion. Logs can be retrieved using:

```
kubectl logs -n <workspace-namespace> -l job-name=projects-pvc-backup2
```

You can check for job completion with:

```
kubectl get jobs -n <workspace-namespace> -l job-name=projects-pvc-backup2
```

Once the job completes, move on to the next step.

15. After the above job is completed, scale up `ds-vfs` and `s2i-client` using the command:

```
kubectl scale -n <workspace-namespace> --replicas=1 deployment ds-vfs
```

and

```
kubectl scale -n <workspace-namespace> --replicas=1 deployment s2i-client
```

16. The upgraded workspace is ready to use. In case you want to delete the backup, then delete the existing backup volume for the workspace using these commands:

```
kubectl delete pvc -n <workspace-namespace> projects-pvc-backup
```

```
kubectl patch pvc -n <workspace-namespace> projects-pvc-backup -p '{"met
adata":{"finalizers":null}}'
```

Note: Taking backup of the existing workspace will take additional space on either Private Cloud cluster (internal backup) or external NFS storage (external backup). So, customers can clear this backup once their workspace is properly migrated.