Machine Learning 1.5.1

ML Workspaces (Private Cloud)

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

CLOUDƎRA

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>.<application domain>

      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*.

What to do next

> **Note:** Prometheus captures data for the previous 30 days.

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

Follow the supported and recommended upgrade paths.

Note: From Cloudera Machine Learning (CML) 1.5.0, 1.5.1, or 1.5.1 Cumulative Hotfix (CHF), upgrade directly to CML 1.5.2. CML 1.5.2 contains fixes for upgrade issues discovered with CML 1.5.1 and 1.5.2. For more details, see TSB 2024-735: Upgrading from affected CML Private Cloud versions is not recommended.

Supported upgrade paths from 1.4.1 and 1.5.0:

Current ML version	NFS	Supported upgrade version	Steps to upgrade
1.5.0	Internal/External	1.5.2	Click the upgrade button located next to the workspace name.
1.4.1	External	1.5.0	Click the upgrade button located next to the workspace name.
1.4.1	Internal	1.5.0	Upgrade manually to 1.5.0.

Upgrade CML workspaces version 1.4.1 to 1.5.0 (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.

   Note: ECS upgrades and supported registries

   - Upgrading a workspace from CDP version 1.4.1 to 1.5.1 is supported only on Embedded Registry
   - Upgrading a workspace from CDP version 1.5.0 to 1.5.1 is supported on Embedded Registry, Public/ Cloudera Default Registry, or External Registry.

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.

Upgrade CML workspaces version 1.4.1 to 1.5.0 (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.

Backing up ML workspaces

Cloudera Machine Learning makes it easy to create machine learning projects, jobs, experiments, ML models, and applications in workspaces. The data and metadata of these artifacts are stored in different types of storage systems in private clouds or in external NFS-backed workspaces outside of a private cloud.

You can backup an ML workspace, and restore it later. The backup preserves all files, models, applications and other assets in the workspace (files are note backed up by CML automatically for external NFS-based workspaces). All workspace backups can be viewed in the Workspace Backup Catalog UI.

The Backup and Restore feature gives you the ability to backup all of your data (except files in external NFS-backed workspaces) to protect your machine learning artifacts against disasters. If your Cloudera Machine Learning workspace is backed up, this feature lets you restore the saved data so that you can recover your ML artifacts as they were saved in the desired backup. The Backup and Restore feature gives the administrator the ability to take "on-demand" backups of CML workspaces. Core services running in the workspace are shut down during the backup process to ensure consistency in the backup data. It is recommended that backups are taken during off-peak hours to minimize user impacts.

The time required to complete backing up a workspace depends on the amount of data to copy. The backup process copies data from both block volumes and internal NFS. In general, the time taken to backup NFS is the dominant factor. You should regularly back up CML workspaces.

The time to backup NFS is highly dependent on the amount of data, and on the nature and number of files. Based on the amount of data, you can set a timeout value while taking backup. You can view the status of ongoing/old backups on CML workspace UI and backup catalog UI.

There is currently no restriction on the number of backups one can take, and the backup snapshots are retained indefinitely in the underlying private cloud cluster as long as the original workspace (from which this backup was

taken from) is not deleted.. CML workspace backup details are stored in the Workspace Backup Catalog UI in the CML control plane, and these entries may be listed, viewed, deleted or restored as desired.

Note: Deleting workspace backups from UI is not yet supported.

Restoring a backup overwrites the existing CML workspace (from which this backup was taken from) wherein the restored data is automatically imported. All the projects, jobs, applications, etc., that were in existence during the backup are automatically available in the new workspace. Restoring a CML backup overwrites the existing workspace with a new one and then launches restore jobs to create storage volumes from the backup snapshots. The restore process takes longer than a regular workspace provisioning operation due to the extra work in copying data from backup to the new storage volumes. Restores are always full-copy restores. The time to restore is dominated by NFS restoration, which takes at least as long as the time to backup the file system. The restored workspace is always created with the latest CML software version, which may be different from the CML version of the original workspace that was backed up.

Note: ML workspace Backup and Restore feature is available on both ECS and OCP, through the CML UI only.

Workspace Backup and Restore Prerequisites

To backup and restore workspaces, check that the following prerequisites are satisfied.

The following prerequisites apply to Backup functionality.

- Backup is enabled only for workspaces that are successfully installed.
- All workloads (sessions, jobs, applications, models) should be turned off manually by the user before taking backup. This will ensure consistency in the backup data.
- Core services running in the workspace are shut down during the backup process. So, during the backup process, the workspace will not be accessible to the customer.
- It is recommended that backups are taken during off-peak hours to minimize user impacts.
- Time to backup is proportional to the amount of data present in the workspace. So, give sufficient timeout when triggering backup.
- Backup is supported for both external and internal NFS-backed workspaces.
- Make sure enough disk space is available for taking workspace backup. Workspace backup generally takes an equivalent amount of storage space compared to the workspace itself.

The following prerequisites apply to Restore functionality.

- Workspace restore doesn't create a new workspace. It will instead replace the running workspace with an older backup.
- Restore process, overwrites the existing workspace with one of the older backups. This means that anything on the running workspace which is not backed up will get lost forever. So, make sure you really want to restore an older version of the workspace. If you want to save the current state before restore, you can do so by first taking a new backup and then proceeding with the restore.
- All workloads (sessions, jobs, applications, models) should be turned off manually by the user before triggering restore.
- All workloads (sessions, jobs, applications, models) should be turned on manually by the user after restore has completed.
- Time to restore is proportional to the amount of data present in the backup. In general, restoration will take at most 12 hours to complete.
- Always make sure that any ongoing backup for a workspace is completed (by looking at workspace status and backup event logs), before triggering restore for it.

Back up an ML workspace

Backing up an ML workspace preserves all files, models, applications, and other assets in the workspace, although files in external NFS-backed workspaces are not backed up by CML automatically.

Procedure

1. In the Workspaces UI, find the workspace to back up. The workspace must be in the Installation completed state, otherwise backup is disabled.
2. Enter the workspace, and manually stop all workloads (sessions, jobs, applications, and models).

    For external NFS backed workspaces, manually back up the configured external NFS data to another location. This manual backup of the NFS data will be used when this particular backup is restored in future. Ignore this step if the workspace is configured with internal NFS, as internal NFS data is backed up and restored automatically by CML.
3. In the Actions menu for that workspace, select Backup Workspace.
4. In the Backup Workspace modal, enter a Backup Name to identify the workspace, and enter an appropriate timeout value.
5. Click Backup to start the process.

Results

The workspace shuts down, and the backup process begins. The workspace state changes to reflect the ongoing backup progress. If necessary, click Cancel to cancel the backup process. The backup process can take some time to complete, depending on the amount of data to copy.

Note: The default timeout is 12 hours. The estimated time to complete a backup (from the cloud provider) is now periodically added to the event logs.

What to do next

Monitoring event logs

You can monitor the progress of the backup process by checking the event logs. In the Actions menu for the workspace, click View Event Logs, and then on the Events & Logs tab, click View Event Logs again for the latest backup event.

When the backup process completes, the workspace enters the Installation completed   state again.

If there were issues during backup, appropriate error messages will be displayed in the event logs. However the workspace will recover from failure and will be reverted back to the original state when backup was triggered.

Restore an ML workspace

Restoring a backup overwrites the existing CML workspace (from which the backup was taken from) and automatically imports the restored data. All of the projects, jobs, applications and so on in the original workspace are recreated in the new one.

About this task

Note: Restoring a workspace is a non-reversible operation. The restore process overwrites the existing workpace with older backup data. Any data in the running workspace that is not backed up will be lost. To save the current state, take a new backup before proceeding with the restore operation.

Procedure

1. In the Workspace Backups UI, find the workspace to restore. You can search for the workspace name or CRN. There can be multiple backups for a given workspace.

2. Enter the workspace, and manually stop all workloads (sessions, jobs, applications, and models).

   For external NFS backed workspaces, copy the manual backup of external NFS data (corresponding to this particular backup) to the configured external NFS export. Ignore this step if the workspace is configured with internal NFS, as internal NFS data is backed up and restored automatically by CML.

3. Look for the backup to restore, and click Restore. The restore process starts, and the workplace state changes to Creating workspace.

Results

The restore process can take some time, depending on the amount of data to copy. When it is complete, you can find the restored workspace in the Workspaces UI.

> **Note:** If there is an issue during the restore process, the event log will show the relevant error messages. In case of error, the workspace will not recover from the failure automatically and will not revert back to the original state prior to the restore operation.

What to do next

Monitoring event logs

You can monitor the progress of the backup process by checking the event logs. In the Actions menu for the workspace, click View Event Logs, and then on the Events & Logs tab, click View Event Logs again for the latest backup event.

When the backup process completes, the workspace enters the installation completed state again.

If there were issues during backup, appropriate error messages will be displayed in the event logs. However the workspace will recover from failure and will be reverted back to the original state when backup was triggered.