Cloudera Data Services on premises 1.5.5

Troubleshooting Cloudera Data Services on premises

Date published: 2023-12-16
Date modified: 2025-11-08

https://docs.cloudera.com/

Legal Notice

Contents

Cloudera Data Services on premises Troubleshooting

This section provides common issues you encounter during Cloudera Data Services on premises installation or upgrade along with the steps to troubleshoot them.

Data Services not functioning because the certificate generation fails

This section provides steps to troubleshoot scenario where data services are not functioning because the certificates are not getting generated.

Procedure

1. Validate cert-manager is installed in cert-manager namespace in ECS by running following command:

```
kubectl get ns | grep cert-manager
```

2. Validate all pods are up and running by executing the following command:

```
kubectl get pods -n cert-manager
```

Note: The pods should have the 'Running' or 'Completed' status.

The following section provides an overview of the pods running in the cert-manager namespace, outlining the specific function of each component. Check the logs of these pods to troubleshoot the cause for any certification failure:

- cert-manager: These pods are the core controllers for cert-manager. They are responsible for processing certificate resources, communicating with certificate authorities (like Let's Encrypt or Venafi), and ensuring that certificates are valid and up-to-date.
  - cdp-release-cert-manager-69ddcb8584-4k77q
  - cdp-release-cert-manager-69ddcb8584-zqh6g
- cainjector: The cainjector's primary role is to inject CA (Certificate Authority) bundles into webhook configurations and other resources. This ensures that the Kubernetes API server can securely communicate with the cert-manager webhooks for validation and mutation of resources.
  - cdp-release-cert-manager-cainjector-75b7947d85-m5crd
  - cdp-release-cert-manager-cainjector-75b7947d85-rkm26
- startupapicheck: This is a one-time job that runs when cert-manager is first deployed or upgraded. It verifies that the Kubernetes API is available and that the necessary cert-manager Custom Resource Definitions (CRDs) have been properly installed and are accessible. The Completed status indicates it has run successfully.
  - cdp-release-cert-manager-startupapicheck-5sz69
- webhook: The webhook pods provide an HTTP server that the Kubernetes API server can query. It has two main functions: to validate cert-manager resource configurations to ensure they are correct and to provide default values for certain fields in those resources. This helps prevent misconfigurations and simplifies the user experience.
  - cdp-release-cert-manager-webhook-6b6b564f77-m8lzz
  - cdp-release-cert-manager-webhook-6b6b564f77-q9mdq

    - cdp-release-cert-manager-webhook-6b6b564f77-srf2p
- Certificate Revocation Operator: This pod runs Cloudera's Certificate revocation operator. It is responsible for monitoring for certificate delete requests and communicating with the Venafi Trust Protection Platform to revoke certificates when they are no longer needed.
    - cdp-release-thunderhead-certrevoke-7d64fd4dbb-xtqhp

Deleting the CNI directory

First run command fails at setup storage step with an error. Pod is stuck in pending state on the host for a long time.

To fix this issue:

Delete the CNI directory on the host failing to launch pods:

```
ssh
              root@ecs-ha1-p-7.vpc.cloudera.com rm -rf /var/lib/cni
```

Restart the canal pod running on that host:

```
kubectl get pods -n kube-system -o wide | grep  ecs-ha1-p-7.vpc.cloudera.com
kube-proxy-ecs-ha1-p-7.vpc.cloudera.com                 1/1     Running
 0          11h   10.65.52.51    ecs-ha1-p-7.vpc.cloudera.com   <none>
     <none>
rke2-canal-llkc9                                        2/2     Running
0          11h   10.65.52.51    ecs-ha1-p-7.vpc.cloudera.com   <none>
    <none>
rke2-ingress-nginx-controller-dqtz8                     1/1     Running
 0          11h   10.65.52.51    ecs-ha1-p-7.vpc.cloudera.com   <none>
     <none>
kubectl delete pod rke2-canal-llkc9 -n kube-system
```

Refreshing the YuniKorn configuration

Sometimes it is possible for the scheduler state to go out of sync from the cluster state. This may result in pods in Pending and ApplicationRejected states, with pod events showing Placement Rule related errors. To recover from this, you may need to refresh the YuniKorn configuration.

Procedure

1. Run the following commands to scale down the YuniKorn pods:

   ```
   kubectl scale deployment  yunikorn-admission-controller --replicas=0 -n
   yunikorn
   kubectl scale deployment  yunikorn-scheduler --replicas=0 -n yunikorn
   ```

   The yunikorn-scheduler and yunikorn-admission-controller pods are managed by the yunikorn-scheduler and yuni korn-admission-controller deployments in the yunikorn namespace, so you can scale down these deployments to 0.
2. Run the following command to delete the yunikorn-configs ConfigMap:

   ```
   kubectl delete cm yunikorn-configs -n yunikorn
   ```
3. Run the following commands to restart the resource-pool-manager pod:

   ```
   kubectl scale deployment cdp-release-resource-pool-manager --replicas=0 -n
    <cdp-namespace>
   ```

```
kubectl scale deployment cdp-release-resource-pool-manager --replicas=1 -n
 <cdp-namespace>
```

The resource-pool-manager pod is managed by the cdp-release-resource-pool-manager deployment in your CDP control plane namespace, so you can scale that deployment down to 0 and then scale it back up to 1.

4. Run the following commands to scale up the YuniKorn pods:

```
kubectl scale deployment  yunikorn-scheduler --replicas=1 -n yunikorn
kubectl scale deployment  yunikorn-admission-controller --replicas=1 -n yu
nikorn
```

The yunikorn-scheduler and yunikorn-admission-controller pods are managed by the yunikorn-scheduler and yunikorn-admission-controller deployments in the yunikorn namespace, so you can scale up these deployments to 1.

Results

The preceding steps will refresh the YuniKorn configuration for the applicable control plane.

After the YuniKorn restart, Pending pods will be picked up and recovered automatically, but pods left in the ApplicationRejected state may need to be redeployed. If the pod is managed by a deployment, you can simply delete the pod. If the pod is unmanaged, you must delete and redeploy the pod.

Testing Longhorn health post Cloudera Embedded Container Service upgrade

Post Cloudera Embedded Container Service upgrade Longhorn health test fails and the `helm-install-longhorn` pod gets in crashloop state.

To fix this issue, run the following command:

```
#Get the history of longhorn helm chart so that we can identify the chart fo
r which installation is failing. #
              helm history longhorn -n longhorn-system
              REVISION    UPDATED                     STATUS          CHART
            APP VERSION    DESCRIPTION
              1           Wed Feb 28 05:32:47 2024    deployed        longh
orn-1.4.2    v1.4.2         Install complete
              2           Wed Feb 28 09:28:39 2024    uninstalling    longh
orn-1.5.4    v1.5.4         Deletion in progress (or silently failed)

              #The actual chart is saved as kubernetes secret. List the lon
ghorn helm chart saved as secrets.#
              kubectl get secrets  -n longhorn-system
              NAME                              TYPE                 DATA
 AGE
              basic-auth                        Opaque               1
15h
              chart-values-longhorn             Opaque               0      1
0h
              longhorn-webhook-ca               kubernetes.io/tls    2
 15h
              longhorn-webhook-tls              kubernetes.io/tls    2
 15h
              sh.helm.release.v1.longhorn.v1    helm.sh/release.v1   1
15h
              sh.helm.release.v1.longhorn.v2    helm.sh/release.v1   1      2
1m
```

```
              #We want to delete the latest chart i.e. sh.helm.release.v1.l
onghorn.v2. Save the back up of the secret as yaml before deleting. #
              kubectl get secrets sh.helm.release.v1.longhorn.v2 -n longho
rn-system -o yaml > sh.helm.release.v1.longhorn.v2.yaml
              #Save the back up of the default values passed along with the
 helm chart while installing.#
              helm get values --revision=2 longhorn -n longhorn-system >
defaultSettings.yaml

              #Find all jobs in longhorn-system and delete those. These jobs
 will be re-triggered as part of the manual patch.#
              kubectl get jobs -n longhorn-system
              NAME                    COMPLETIONS   DURATION   AGE
              helm-install-longhorn   0/1           9h         9h
              longhorn-post-upgrade   1/1           11m        10h
              longhorn-uninstall      0/1           10h        10h

              #Delete the latest longhorn chart#
              kubectl delete job helm-install-longhorn longhorn-uninstall l
onghorn-post-upgrade -n longhorn-system
              kubectl delete secret sh.helm.release.v1.longhorn.v2 -n longho
rn-system

              #Apply the longhorn chart from the parcel directory.#
               kubectl patch HelmChart longhorn -n longhorn-system --type=m
erge --patch-file /opt/cloudera/parcels/ECS/longhorn/longhorn.yaml
```

Restarting the Docker Servers

Docker servers fail to come up after starting the cluster post hosts reboot.

At times the Docker server may fail to come up and return the following error message:

```
/var/run/docker.sock: Is a directory
```

To fix this issue, remove the /var/run/docker.sock directory on the Docker server role host and then restart the Docker server role.

ECS Installation failure due to External Registry issues

During a fresh ECS install, if you encounter image pull issues with a custom docker registry set up, here are some steps to check if your custom registry is configured properly.

During a fresh ECS install, installation failure when pulling images typically indicates incorrect setup of the external or custom registry.

Istio image is the first to be pulled from the custom registry.

Describing the pod may look similar to:

Failed to pull image “registry.ecs.internal/cloudera_thirdparty/gloo_mesh/istio-4d37697f9711/pilot:1.24.2-solo-distroless”... failed to do request: Head: https://registry.ecs.internal/cloudera_thirdparty.... : dial tcp: lookup registry.ecs.internal on …

Run the ECS Refresh command to update the docker pull secrets for the custom registry:

Go to ECS Service Actions > Refresh ECS

Procedure

1. Navigate to Cloudera Manager UI ->Containerized cluster (ECS Cluster Name) ->Configuration .

2. Search for "docker" and verify the following parameters: External Container Registry User, External Container Registry Password, External Container Registry, External Docker Registry Certificate (PEM format). Only TLS-enabled custom Docker Registry is supported.
3. If any configurations are incorrect, update and save the changes.
4. Restart the ECS cluster to apply the changes(All ECS server and agents service need to be restarted. Note that hosts restart is not required.)
5. Once the command succeeds, navigate to Cloudera Manager UI -> Running Commands , find the failed First Run command, and click Resume to proceed with the installation process.

Run ECS Refresh Command

While updating the docker username/password, ensure to refresh ECS. Without Refresh ECS, the username/password will not be updated in the cdp-private-installer-docker-pull-secret secret objects.

Procedure

Run the ECS Refresh command and update the docker pull secrets for the custom registry: Go to ECS Server Actions > Refresh ECS

How to fix errors detected by Pre-Upgrade Checks

This section provides troubleshooting steps for common errors detected during the pre-upgrade checks.

Download Upgrade Validator

Problem: "Failed to Distribute to an ecs host"

Steps to resolve:

1. Verify if the Cloudera Manager agent on the failed host is having any issues communicating with the Cloudera Manager server.
2. Click View Command Details # Retry the command again if you have verified issues seen in the logs from the above steps are resolved.
3. Check the Cloudera Manager server logs for any error logs.

Control Plane Health Checks

Vault

Problem: "Failed to get Vault pod name/No Running Vault Pod Found"

No running Vault pod found in the specified namespace.

Steps to resolve:

1. Check if Vault pods are running:

```
kubectl get pods -n vault-system -l app.kubernetes.io/name=vault
```

2. If no pods are running, restart Vault:

```
kubectl rollout restart statefulset vault -n vault-system
```

Problem: "Failed to Execute Command in Vault Pod"

Failed to execute command in pod:

```
<error_message>, stderr: <stderr_output>
```

Steps to resolve:

1. Restart Vault by executing the following command:

```
kubectl rollout restart statefulset vault -n vault-system
```

Problem: "Vault is sealed"

Vault is in a sealed state.

Steps to resolve:

1. Log in to the Cloudera Manager Admin Console.
2. Navigate to the ECS Cluster and select the ECS cluster in Cloudera Manager UI.
3. Access Actions: Click on the Actions menu.
4. Unseal the Vault: Choose Unseal Vault from the Actions menu.

Longhorn

Problem: "Failed to list Longhorn volumes"

The Longhorn volume resource could not be retrieved due to API server issues or Longhorn component failures.

Steps to resolve:

1. Verify that the Longhorn volumes are accessible:

```
kubectl get volumes.longhorn.io -n longhorn-system
```

2. Check if the Longhorn manager and related pods are running:

```
kubectl get pods -n longhorn-system
```

3. If any Longhorn pods are in CrashLoopBackOff or Pending states, restart them:

```
kubectl delete pod <pod-name> -n longhorn-system
```

4. If the issue persists, restart ECS services.

Problem: Unhealthy Longhorn volumes detected

Some Longhorn volumes are in a degraded, detached, or error state.

Steps to resolve:

1. Check the status of Longhorn volumes:

```
kubectl get volumes.longhorn.io -n longhorn-system -o jsonpath='{range
.items[*]}{.metadata.name} {.status.robustness}{"\n"}{end}'
```

2. If a volume is degraded or faulted, check volume details:

```
kubectl describe volumes.longhorn.io <volume-name> -n longhorn-system
```

3. If a volume is detached, attempt to attach it:

```
kubectl patch volumes.longhorn.io <volume-name> -n longhorn-system --typ
e='merge' -p '{"spec":{"frontend":"blockdev"}}'
```

4. Restart the Longhorn manager if necessary:

```
kubectl rollout restart daemonset longhorn-manager -n longhorn-system
```

Problem: Longhorn PVCs are not bound

PersistentVolumeClaims (PVCs) using Longhorn are not in a Bound state.

Steps to reolve:

1. List all PVCs in the cluster:

```
kubectl get pvc --all-namespaces
```

2. If a PVC is stuck in Pending state, check the corresponding PersistentVolume (PV):

```
kubectl get pv
```

3. Describe the problematic PVC to identify binding issues:

```
kubectl describe pvc <pvc-name> -n <namespace>
```

4. If the PV is Released but not Available, manually delete it:

```
kubectl delete pv <pv-name>
```

5. Restart the Longhorn services if necessary:

```
kubectl rollout restart daemonset longhorn-manager -n longhorn-system
```

RKE2

Problem: "Failed to check API server"

API server is unreachable.

Steps to resolve:

1. Verify the API server endpoint:

```
kubectl cluster-info
```

2. Check if the API server pods are running:

```
kubectl get pods -n kube-system -l component=kube-apiserver
```

3. Stop Cloudera Embedded Container Service.
4. Reboot hosts.

5. Start Cloudera Embedded Container Service.

   The start command fails with the following error message:

   Timed out waiting for kube-apiserver to be ready

   Option 1:

   Start each master role instance individually without waiting for each node to be up and running.

   Option 2:

   If Option 1 does not work, follow the steps from SUSE to recover the cluster:https://docs.rke2.io/backup_restore#cluster-reset

---

Problem: One or more nodes are in a NotReady state.

Steps to resolve:

1. Check node status:

   ```
   kubectl get nodes -o wide
   ```

2. Describe the problematic node:

   ```
   kubectl describe node <node-name>
   ```

3. Stop all roles on the affected host.
4. If the node is not required, remove it from the cluster.
5. Reboot the host.
6. Restart all roles on the host.

Pods in kube-system are unhealthy

Problem: Critical control plane or system pods are failing.

Steps to resolve:

1. Check pod statuses in kube-system:

   ```
   kubectl get pods -n kube-system
   ```

2. If any pods are in CrashLoopBackOff, check logs:

   ```
   kubectl logs <pod-name> -n kube-system
   ```

3. Restart unhealthy pods:

   ```
   kubectl delete pod <pod-name> -n kube-system
   ```

4. If the issue persists, restart ECS services.

Problem: Control Plane Pod XXX Readiness, containers with unready status

Steps to resolve:

1. Try to scale down and scale up the controller of the pod (Typically deployment or Replicaset).
2. If the problem persists, look for the reason of failure with the following commands:

   ```
   kubectl get events -n <pod-namespace>
   ```

```
kubectl describe pod <pod-name> -n <pod-namespace>
kubectl logs <pod-name> -n <pod-namespace> -c <container-name>
```

Docker Registry Health Checks

Problem: Docker registry connection failed and it's been verified the new ecs-toleration-webhook image required for upgrade is in docker registry.

You may see an error message similar to:

```
2025/02/27 13:11:04 Docker registry connection failed. Unable to obtain ecs-
toleration-webhook manifest: Status Code: 400, Response: {"errors":
[{"code":"MANIFEST_INVALID","message":"manifest invalid","detail":{"DriverNa
me":"filesystem","Enclosed":
{"Op":"mkdir","Path":"/var/lib/registry/docker/registry/v2/blobs/sha256/a3/
a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4","Err":28}}
}]}
```

This error is due to the fact that your docker registry host is running out of disk space. When we request information about an image manifest, if the information is not available in cache, then the registry would need to construct it again, and during that process it may need to create temporary files or directories. To resolve this issue, either add more disk space to your docker registry host or clean up the registry and re-download the required images.