dremio

Dremio Software

Operating Dremio on Kubernetes

Introduction

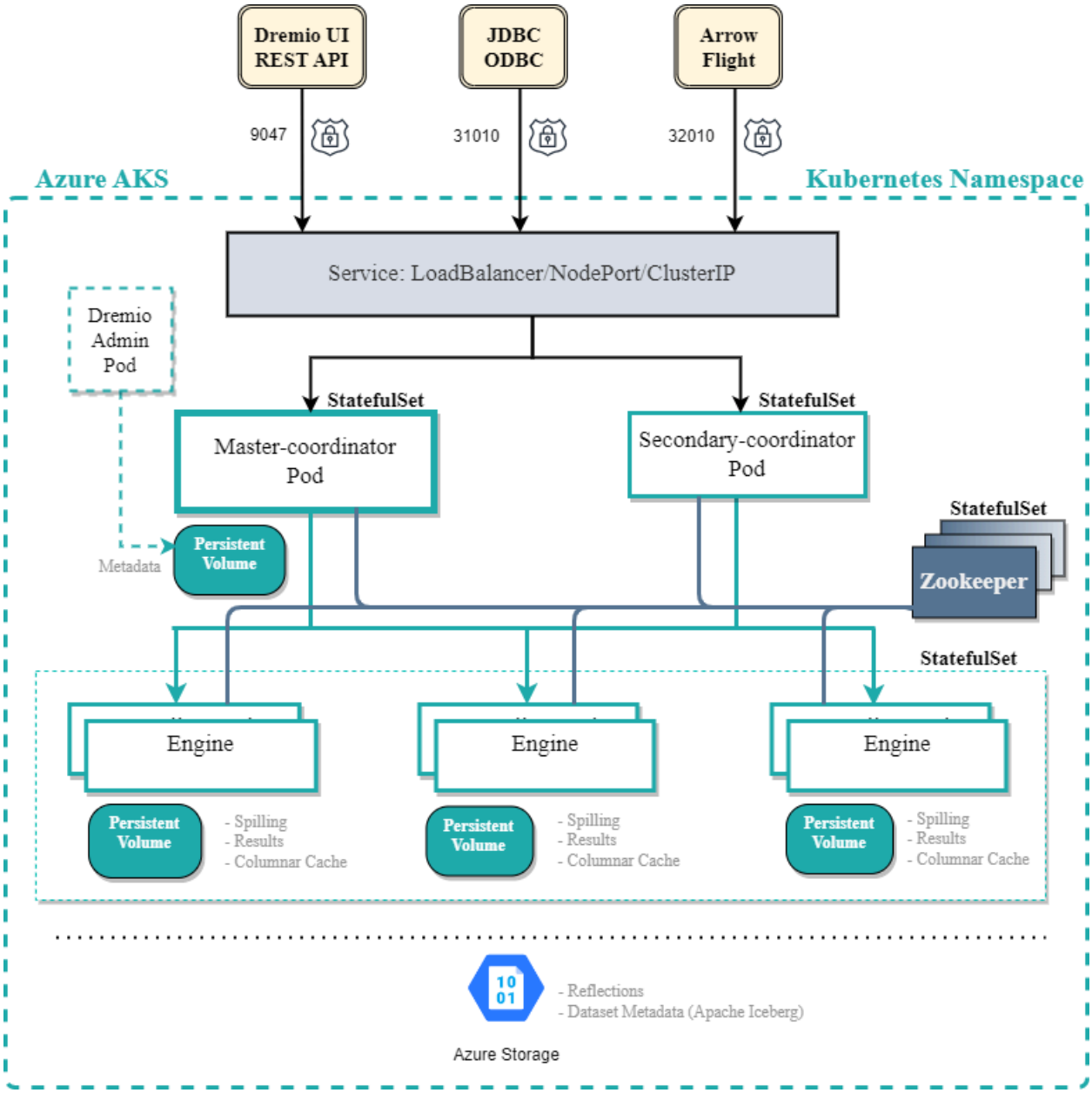
The document gives guidance on how to operate Dremio on Kubernetes. This includes

- Troubleshoot common issues
- Monitor network and node utilization
- Monitor disk performance (IOPS and queue depth)
- How to verify the healthiness of nodes and pods
- Best practices of how the Helm Charts should be updated

The intention of the document is not to cover how to operate Dremio in general. If you need this, please download the Operating Dremio Runbook.

Dremio Kubernetes Architecture

The architecture below shows an example of an Azure Kubernetes Service deployment.



The following table contains a high-level mapping of the standalone deployment components to Kubernetes terminology:

Standalone Deployment	Kubernetes
External load balancer for the Web-UI, ODBC and Arrow-Flight (Ports 9047, 31010, 32010)	The load balancer in Kubernetes is represented as a service component. A service can be a load balancer or just a cluster IP address. Kubernetes takes

	responsibility for provisioning the load balancer. Sometimes, no load balancer integration is available, especially in on-premise Kubernetes deployments. In this case, an external load balancer can be provided and bound to the local Kubernetes node ports.
The Dremio coordinator and executors run on standalone VMs and are started and managed with a systemctl service.	The Dremio coordinator and executors run in pods. This is similar to a Docker container instance. Pods can be moved to another node if a node fails. Pods are orchestrated and managed by Kubernetes. A pod itself is managed by a stateful set. This facilitates scaling the executor nodes and ensures all executor pods see the same configuration.
VMs usually have local disks, block devices, or NFS attached. These disks store Dremio's metadata or store caches for the executors.	Kubernetes has an abstraction layer for disks called Persistent Volume Claim (PVC). A PVC is created based on a storage class. These storage classes define the underlying implementation type, such as NFS or block storage. Avoid local storage, which can have negative impacts in case of failures and does not allow you to reassign pods to other nodes.
Dremio's distributed storage is stored on the data lake. This includes metadata, reflections, and uploads. No data is stored on the local disk.	There is no difference for Kubernetes; the distributed storage works the same: Dremio's distributed storage is hosted on the data lake. This includes metadata, reflections, and uploads. No data is stored on the local disk.

Check for Coordinator, Executor and Zookeeper Restarts

Under some circumstances, restarts for the coordinator, executors or Zookeeper can occur.

```
$ kubectl get pods

NAME                                READY   STATUS      RESTARTS       AGE
dremio-executor-0                   1/1     Running     1 (28d ago)    28d
dremio-executor-1                   1/1     Running     0              27d
dremio-master-0                     1/1     Running     1 (5h2m ago)   12h
zk-0                                1/1     Running     0              28d
```

Investigate the Cause for Restarts

Common cases for restarts are:

- Zookeeper can restart when the healthiness probe uses netcat to validate the connection. This approach is unreliable and often returns an error, even if Zookeeper is properly working and up and running. If the error is returned three times in a row, Kubernetes assumes that Zookeeper is down and restarts it. Please ensure that your Helm Charts have this fix included.
- Pods for the coordinator and executors can be OOM killed (out of memory). This mainly happens when limits have been added to the Helm Charts or the pod exceeds the capacity of the host node.
- Another cause for restarts on the coordinator can be when taking backups using the `dremio-admin` command. The `dremio-admin` command runs within the dremio-master-0 pod and will allocate additional memory. If the cumulative memory of the coordinator and dremio-admin process exceeds the defined limit or node capacity, the pod will also be OOM killed.
- The Java process could just crash. If the Java process stops due to a SEGFAULT, it will restart immediately.
- The Java process could get unresponsive due to significant GC pauses. The health probe regularly checks the health via a REST call. If it gets unresponsive, the process will be restarted. An indication for this could be a SIGTERM return code 143 (or SIGKILL return code 137). Kubernetes sends a termination signal to the container to restart it.
- The Java process could just exit with a lower return code. An example might be that Zookeeper went down and has no quorum; after a while, the Dremio coordinator process might fail and restart.

Below is an example of how to retrieve the restart reason. The example is shortened since the `kubectl describe` command returns a lot of information. In the example, we can see that Kubernetes terminated the container because it exceeded the node capacity. Limits are not set. Additionally, it might be possible to see some events on the pods. In this case, there are none. Events are only stored for a very short time (1 to 3 hours) in most Kubernetes systems.

```
$ kubectl describe pod dremio-master-0

Name:             dremio-master-0
Namespace:        default
...(shortened)
Status:           Running
IP:               10.240.0.25
IPs:
```

```
  IP:           10.240.0.25
Controlled By:  StatefulSet/dremio-master
Init Containers:
 ... (shortened)
Containers:
  dremio-master-coordinator:
    Container ID:  containerd://9d8275d316b3ec874ff403abdd2c12126923c6fcc11dcd6cf60f20bf381e5dbe
    Image:         yourregistry.azurecr.io/dremio-ee:24.2.6
    Image ID:
yourregistry.azurecr.io/dremio-ee@sha256:f0646a33998ffc0a3a0769ffa29829b439576a7bca3d5b703af280a
f8e427e24
    Ports:         9047/TCP, 31010/TCP, 32010/TCP, 45678/TCP, 45679/TCP
    Host Ports:    0/TCP, 0/TCP, 0/TCP, 0/TCP, 0/TCP
    Command:
      /opt/dremio/bin/dremio
    Args:
      start-fg
    State:          Running
      Started:      Tue, 02 Jan 2024 10:31:24 +0100
    Last State:     Terminated
      Reason:       OOMKilled
      Exit Code:    137
      Started:      Tue, 02 Jan 2024 03:01:36 +0100
      Finished:     Tue, 02 Jan 2024 10:31:24 +0100
    Ready:          True
    Restart Count:  1
    Requests:
      cpu:      2
      memory:   11Gi
    Readiness:  http-get http://:web/ delay=0s timeout=1s period=1s #success=1 #failure=300
    Startup:    http-get http://:web/ delay=0s timeout=1s period=1s #success=1 #failure=600
    Environment:
      DREMIO_MAX_HEAP_MEMORY_SIZE_MB:    9216
      DREMIO_MAX_DIRECT_MEMORY_SIZE_MB:  2048
      DREMIO_JAVA_SERVER_EXTRA_OPTS:     -Ddremio.log.path=/opt/dremio/data/log
... (shortened)
    Mounts:
... (shortened)
Volumes:
  dremio-master-volume:
    Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same
namespace)
    ClaimName:  dremio-master-volume-dremio-master-0
    ReadOnly:   false
...(shortened)
Events:                      <none>
```

The example below shows the termination reason for the executor. In this case, the Java Virtual Machine returned an error and terminated. Kubernetes did not terminate the process. A further investigation into Dremio's server.log would be required. A likely reason for a restart here might be that Zookeeper restarted and lost the quorum and connection. In this example cluster, there is only one Zookeeper deployed.

```
$ kubectl describe pod dremio-executor-0

Name:             dremio-executor-0
Namespace:        default
... (shortened)
Status:           Running
IP:               10.240.0.101
IPs:
  IP:           10.240.0.101
Controlled By:  StatefulSet/dremio-executor
Init Containers:
  ... (shortened)
Containers:
  dremio-executor:
    Container ID:  containerd://968827f376365d6df7396b758cc76b7d1636aca4d22c42682f945956584e4871
    Image:         yourregistry.azurecr.io/dremio-ee:24.2.6
    Image ID:
yourregistry.azurecr.io/dremio-ee@sha256:f0646a33998ffc0a3a0769ffa29829b439576a7bca3d5b703af280a
f8e427e24
    Ports:         45678/TCP, 45679/TCP, 9010/TCP
    Host Ports:    0/TCP, 0/TCP, 0/TCP
    Command:
      /opt/dremio/bin/dremio
    Args:
      start-fg
    State:          Running
      Started:      Tue, 05 Dec 2023 13:31:10 +0100
    Last State:     Terminated
      Reason:       Error
      Exit Code:    4
      Started:      Tue, 05 Dec 2023 13:29:37 +0100
      Finished:     Tue, 05 Dec 2023 13:31:09 +0100
    Ready:          True
    Restart Count:  1
    Requests:
      cpu:     14
      memory:  56Gi
    Environment:
      DREMIO_MAX_HEAP_MEMORY_SIZE_MB:    8192
      DREMIO_MAX_DIRECT_MEMORY_SIZE_MB:  49152
      DREMIO_JAVA_SERVER_EXTRA_OPTS:     -Ddremio.log.path=/opt/dremio/data/log
```

```
  ... (shortened)
Volumes:
  dremio-default-executor-volume:
    Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same
namespace)
    ClaimName:  dremio-default-executor-volume-dremio-executor-0
    ReadOnly:   false
  dremio-default-executor-c3-0:
    Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same
namespace)
    ClaimName:  dremio-default-executor-c3-0-dremio-executor-0
    ReadOnly:   false
  ... (shortened)
Events:                      <none>
```

Check Server Logs for Errors

Before logs can be used for analysis, they must be configured in the values.yml of the Helm Charts. Add the following configuration to the coordinator and executor section:

```
extraStartParams: >-
   -Ddremio.log.path=/opt/dremio/data/log
```

This example follows up on the termination above. The output of the `kubectl describe` command can be used to isolate the time range to look into the logs:

```
    Last State:     Terminated
      Reason:       Error
      Exit Code:    4
      Started:      Tue, 05 Dec 2023 13:29:37 +0100
      Finished:     Tue, 05 Dec 2023 13:31:09 +0100
```

The time of the `kubectl` command is in UTC and needs to be converted. For the `grep` command, the time above becomes (based on the range, you can add seconds as well):

```
Tue, 05 Dec 2023 13:31:09 +0100  ⇒  2023-12-05 12:31
```

Below you can see the command with the output. In this case, the cause for the restart was that the executor could not talk to the coordinator. The coordinator was unreachable because of a DNS issue within the cluster or the unavailability of the dremio-master-0 pod.

```
$ kubectl exec -it dremio-executor-0 -c dremio-executor -- bash -c "zgrep -E '2023-12-05 12:30|2023-12-05 12:31' -A 10 /opt/dremio/data/log/server.log /opt/dremio/data/log/archive/server*.log.gz"

/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz:2023-12-05 12:30:43,594 [FABRIC] WARN c.d.telemetry.api.metrics.Metrics - Removing old metric since name matched newly registered metric. Metric name: FABRIC-send-durations-ms
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz:2023-12-05 12:30:43,611 [FABRIC-2] INFO c.d.services.fabric.FabricClient - [FABRIC]: Channel closed null <--> null (fabric client)
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz:2023-12-05 12:30:43,612 [FABRIC-2] ERROR com.dremio.exec.rpc.BasicClient - Failed to establish connection
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-java.util.concurrent.ExecutionException: java.net.UnknownHostException: dremio-master-0.dremio-cluster-pod.default.svc.cluster.local: Name or service not known
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at io.netty.util.concurrent.DefaultPromise.get(DefaultPromise.java:374)
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at com.dremio.exec.rpc.BasicClient$ConnectionMultiListener$ConnectionEstablishmentListener.operationComplete(BasicClient.java:301)
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at com.dremio.exec.rpc.BasicClient$ConnectionMultiListener$ConnectionEstablishmentListener.operationComplete(BasicClient.java:289)
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at io.netty.util.concurrent.DefaultPromise.notifyListener0(DefaultPromise.java:590)
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at io.netty.util.concurrent.DefaultPromise.notifyListenersNow(DefaultPromise.java:557)
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at io.netty.util.concurrent.DefaultPromise.notifyListeners(DefaultPromise.java:492)
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at io.netty.util.concurrent.DefaultPromise.setValue0(DefaultPromise.java:636)
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at io.netty.util.concurrent.DefaultPromise.setFailure0(DefaultPromise.java:629)
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at io.netty.util.concurrent.DefaultPromise.setFailure(DefaultPromise.java:110)
--
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz:2023-12-05 12:30:48,613 [FABRIC] WARN c.d.telemetry.api.metrics.Metrics - Removing old metric since name matched newly registered metric. Metric name: FABRIC-send-durations-ms
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz:2023-12-05 12:30:48,618 [FABRIC-3] INFO c.d.services.fabric.FabricClient - [FABRIC]: Channel closed null <--> null (fabric client)
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz:2023-12-05 12:30:48,619 [FABRIC-3] ERROR com.dremio.exec.rpc.BasicClient - Failed to establish connection
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-java.util.concurrent.ExecutionException: java.net.UnknownHostException: dremio-master-0.dremio-cluster-pod.default.svc.cluster.local
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at io.netty.util.concurrent.DefaultPromise.get(DefaultPromise.java:374)
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at com.dremio.exec.rpc.BasicClient$ConnectionMultiListener$ConnectionEstablishmentListener.operationComplete(BasicClient.java:301)
```

```
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at
com.dremio.exec.rpc.BasicClient$ConnectionMultiListener$ConnectionEstablishmentListener.operatio
nComplete(BasicClient.java:289)
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at
io.netty.util.concurrent.DefaultPromise.notifyListener0(DefaultPromise.java:590)
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at
io.netty.util.concurrent.DefaultPromise.notifyListenersNow(DefaultPromise.java:557)
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at
io.netty.util.concurrent.DefaultPromise.notifyListeners(DefaultPromise.java:492)
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at
io.netty.util.concurrent.DefaultPromise.setValue0(DefaultPromise.java:636)
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at
io.netty.util.concurrent.DefaultPromise.setFailure0(DefaultPromise.java:629)
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz-    at
io.netty.util.concurrent.DefaultPromise.setFailure(DefaultPromise.java:110)
--
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz:2023-12-05 12:31:03,365 [main] INFO
c.d.s.s.LocalSchedulerService - Stopping SchedulerService
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz:2023-12-05 12:31:04,367 [main] INFO
c.d.s.s.LocalSchedulerService - Stopped SchedulerService
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz:2023-12-05 12:31:04,379 [main] INFO
c.d.s.o.LocalOwnershipServiceServer - LocalOwnershipService stopped
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz:2023-12-05 12:31:04,379 [main] INFO
c.d.datastore.NoopKVStoreProvider - Stopped NoopKVStoreProvider
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz:2023-12-05 12:31:04,389 [main] INFO
c.d.s.fabric.EnterpriseFabricServer - [FABRIC]: Server shutdown.
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz:2023-12-05 12:31:04,394 [main] INFO
c.d.s.c.TaskLeaderStatusListener - Stopping TaskLeaderStatusListener for: MASTER
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz:2023-12-05 12:31:04,394 [main] INFO
c.d.s.c.TaskLeaderStatusListener - Stopped TaskLeaderStatusListener for: MASTER
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz:2023-12-05 12:31:04,397 [main] INFO
c.d.s.coordinator.zk.ZKClusterClient - Stopping ZKClusterClient
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz:2023-12-05 12:31:04,405 [main] INFO
c.d.s.coordinator.zk.ZKClusterClient - Stopped ZKClusterClient
/opt/dremio/data/log/archive/server.2023-12-05.0.log.gz:2023-12-05 12:31:04,409 [main] ERROR
ROOT - Dremio is exiting. Failure while starting services.
```

A similar command could be used to grep for exceptions or errors:

```
$ kubectl exec -it dremio-master-0 -c dremio-master-coordinator -- bash -c "zgrep -E 'Exception'
-A 20 /opt/dremio/data/log/server.log /opt/dremio/data/log/archive/server*.log.gz"

/opt/dremio/data/log/server.log:java.lang.NullPointerException: Master coordinator is down
/opt/dremio/data/log/server.log- at
com.google.common.base.Preconditions.checkNotNull(Preconditions.java:921)
/opt/dremio/data/log/server.log- at
com.dremio.service.conduit.client.ConduitProviderImpl.getOrCreateChannelToMaster(ConduitProvider
```

```
Impl.java:180)
/opt/dremio/data/log/server.log- at
com.dremio.plugins.sysflight.SysFlightStoragePlugin.getFlightClient(SysFlightStoragePlugin.java:
109)
/opt/dremio/data/log/server.log- at
com.dremio.plugins.sysflight.SysFlightStoragePlugin.getFlightTableList(SysFlightStoragePlugin.ja
va:239)
/opt/dremio/data/log/server.log- at
com.dremio.plugins.sysflight.SysFlightStoragePlugin.listDatasetHandles(SysFlightStoragePlugin.ja
va:191)
... (shortened)
--
/opt/dremio/data/log/server.log:2024-01-02 04:46:40,174 [UserServer-1] WARN
c.d.exec.rpc.RpcExceptionHandler - Exception occurred with closed channel.  Connection:
/10.240.0.25:31010 <--> /10.240.0.156:38174 (user client)
/opt/dremio/data/log/server.log:io.netty.handler.codec.DecoderException:
io.netty.handler.ssl.NotSslRecordException: not an SSL/TLS record:
0300002f2ae00000000000436f6f6b69653a206d737473686173683d41646d696e697374720d0a0100080003000000
/opt/dremio/data/log/server.log- at
io.netty.handler.codec.ByteToMessageDecoder.callDecode(ByteToMessageDecoder.java:499)
/opt/dremio/data/log/server.log- at
io.netty.handler.codec.ByteToMessageDecoder.channelRead(ByteToMessageDecoder.java:290)
/opt/dremio/data/log/server.log- at
io.netty.channel.AbstractChannelHandlerContext.invokeChannelRead(AbstractChannelHandlerContext.j
ava:444)
/opt/dremio/data/log/server.log- at
io.netty.channel.AbstractChannelHandlerContext.invokeChannelRead(AbstractChannelHandlerContext.j
ava:420)
/opt/dremio/data/log/server.log- at
io.netty.channel.AbstractChannelHandlerContext.fireChannelRead(AbstractChannelHandlerContext.jav
a:412)
/opt/dremio/data/log/server.log- at
io.netty.channel.DefaultChannelPipeline$HeadContext.channelRead(DefaultChannelPipeline.java:1410
)
... (shortened)
```

Check Garbage Collection Logs for Long Pauses

Before the garbage can be monitored, it must be configured in the values.yml of the Helm Charts. Add the following configuration to the coordinator and executor sections:

```
extraStartParams: >-
   -Ddremio.log.path=/opt/dremio/data/log
   -Xloggc:/opt/dremio/data/log/gc-%t.log -XX:+PrintGCTimeStamps
```

The command below finds lines with garbage collection pauses larger than one second. We can also see very large pauses of up to 8 seconds. During this time, the entire Java process becomes unresponsive. If the pause exceeds 120 seconds, the pod will restart.

```
$ kubectl exec -it dremio-master-0 -c dremio-master-coordinator -- bash -c "grep 'real='
/opt/dremio/data/log/gc*.log | grep -v 'real=0'"

/opt/dremio/data/log/gc-2023-06-12_01-09-25.log: [Times: user=1.27 sys=0.02, real=1.09 secs]
/opt/dremio/data/log/gc-2023-06-12_01-09-25.log: [Times: user=8.27 sys=0.11, real=7.95 secs]
/opt/dremio/data/log/gc-2023-06-12_01-09-25.log: [Times: user=8.10 sys=0.07, real=7.76 secs]
/opt/dremio/data/log/gc-2023-07-10_01-07-35.log: [Times: user=3.09 sys=0.02, real=2.76 secs]
/opt/dremio/data/log/gc-2023-08-07_01-09-04.log: [Times: user=2.97 sys=0.05, real=2.70 secs]
/opt/dremio/data/log/gc-2023-08-07_01-09-04.log: [Times: user=1.15 sys=0.20, real=1.01 secs]
/opt/dremio/data/log/gc-2023-09-01_02-00-48.log: [Times: user=2.04 sys=0.08, real=1.81 secs]
/opt/dremio/data/log/gc-2023-09-01_02-00-48.log: [Times: user=8.41 sys=0.02, real=7.95 secs]
/opt/dremio/data/log/gc-2023-09-24_02-00-41.log: [Times: user=1.31 sys=0.02, real=1.15 secs]
/opt/dremio/data/log/gc-2023-09-24_02-00-41.log: [Times: user=8.62 sys=0.01, real=8.08 secs]
/opt/dremio/data/log/gc-2023-10-25_16-32-10.log: [Times: user=2.16 sys=0.04, real=1.96 secs]
/opt/dremio/data/log/gc-2023-10-25_16-32-10.log: [Times: user=8.86 sys=0.09, real=8.37 secs]
/opt/dremio/data/log/gc-2023-11-27_15-54-07.log: [Times: user=8.26 sys=0.12, real=7.89 secs]
/opt/dremio/data/log/gc-2023-11-27_15-54-07.log: [Times: user=1.62 sys=0.02, real=1.42 secs]
/opt/dremio/data/log/gc-2023-11-27_15-54-07.log: [Times: user=8.45 sys=0.10, real=8.03 secs]
/opt/dremio/data/log/gc-2023-11-27_15-54-07.log: [Times: user=7.87 sys=0.05, real=7.49 secs]
/opt/dremio/data/log/gc-2023-12-15_02-00-07.log: [Times: user=1.33 sys=0.04, real=1.21 secs]
/opt/dremio/data/log/gc-2023-12-15_02-00-07.log: [Times: user=8.65 sys=0.01, real=8.17 secs]
```

Another example for the executor:

```
$ kubectl exec -it dremio-executor-0 -c dremio-executor -- bash -c "grep 'real='
/opt/dremio/data/log/gc*.log | grep -v 'real=0'"

/opt/dremio/data/log/gc-2022-08-24_20-59-45.log: [Times: user=3.98 sys=0.00, real=1.23 secs]
/opt/dremio/data/log/gc-2022-08-24_20-59-45.log: [Times: user=2.79 sys=0.00, real=1.02 secs]
/opt/dremio/data/log/gc-2022-08-24_20-59-45.log: [Times: user=3.80 sys=0.00, real=1.20 secs]
/opt/dremio/data/log/gc-2022-08-24_20-59-45.log: [Times: user=4.04 sys=0.00, real=1.49 secs]
/opt/dremio/data/log/gc-2022-08-24_20-59-45.log: [Times: user=3.88 sys=0.00, real=1.17 secs]
/opt/dremio/data/log/gc-2022-08-24_20-59-45.log: [Times: user=5.94 sys=0.00, real=1.28 secs]
/opt/dremio/data/log/gc-2022-08-24_20-59-45.log: [Times: user=6.21 sys=0.00, real=1.27 secs]
/opt/dremio/data/log/gc-2022-08-30_12-09-09.log: [Times: user=3.98 sys=0.00, real=1.25 secs]
/opt/dremio/data/log/gc-2022-08-30_12-09-09.log: [Times: user=3.40 sys=0.00, real=1.05 secs]
/opt/dremio/data/log/gc-2022-08-30_12-09-09.log: [Times: user=4.09 sys=0.00, real=1.19 secs]
/opt/dremio/data/log/gc-2022-08-30_12-09-09.log: [Times: user=3.45 sys=0.00, real=1.09 secs]
/opt/dremio/data/log/gc-2022-08-30_12-09-09.log: [Times: user=4.33 sys=0.00, real=1.25 secs]
/opt/dremio/data/log/gc-2022-08-30_12-09-09.log: [Times: user=3.84 sys=0.01, real=1.19 secs]
/opt/dremio/data/log/gc-2022-09-02_13-55-17.log: [Times: user=4.01 sys=0.00, real=1.50 secs]
/opt/dremio/data/log/gc-2022-09-02_13-55-17.log: [Times: user=3.88 sys=0.00, real=1.19 secs]
```

Verify the Healthiness of the Host Nodes

Verify Node Status

First, verify that all nodes are up and running and have the status 'ready':

```
$ kubectl get nodes
NAME                                STATUS   ROLES   AGE   VERSION
aks-agentpool-27649900-vmss000011   Ready    agent   41d   v1.27.3
aks-coordpool-35637789-vmss000029   Ready    agent   36d   v1.27.3
aks-executorpool-11591932-vmss00002q   Ready    agent   41d   v1.27.3
aks-executorpool-11591932-vmss00002u   Ready    agent   27d   v1.27.3
```

Connect to the Kubernetes Host Node

If you suspect issues with a specific host node, use the following command to connect to the host node. The hostname can be picked from the output of the `kubectl get nodes` command.

```
$ kubectl debug node/aks-coordpool-35637789-vmss000029 -it
--image=dremiops.azurecr.io/dremio-k8s-tools:v1.15
$ chroot /host
```

More information about the `kubectl debug` command can be found here.

Check Host Node Logs

Once you connected to the host node, use the following command:

```
$ grep -i -E "WARN|ERROR|FATAL" /var/log/messages

Jan  3 02:00:49 aks-coordpool-35637789-vmss000029 networkd-dispatcher[797]: ERROR:Failed to get
interface "azvc3f87939b22" status: Command '['/usr/bin/networkctl', 'status', '--no-pager',
'--no-legend', '--', 'azvc3f87939b22']' returned non-zero exit status 1.
Jan  3 02:01:18 aks-coordpool-35637789-vmss000029 containerd[2315]:
time="2024-01-03T02:01:18.158922580Z" level=warning msg="Failed to get podSandbox status for
container event for sandboxID
\"69b818abcec642e25f58794f4a4a5245fb94e32cb9ef6ef2022c674ed87e1a64\": an error occurred when try
to find sandbox: not found. Sending the event with nil podSandboxStatus."
Jan  3 08:32:41 aks-coordpool-35637789-vmss000029 python3[1874]: 2024-01-03T08:32:41.302609Z
INFO ExtHandler ExtHandler [HEARTBEAT] Agent WALinuxAgent-2.9.1.1 is running as the goal state
agent [DEBUG HeartbeatCounter: 1765;HeartbeatId:
6D50307C-3F16-474E-B837-D4C0CCFC0528;DroppedPackets: 0;UpdateGSErrors: 0;AutoUpdate: 1]
Jan  3 09:00:00 aks-coordpool-35637789-vmss000029 networkd-dispatcher[797]: WARNING:Unknown
index 216 seen, reloading interface list
```

```
Jan  3 09:00:13 aks-coordpool-35637789-vmss000029 containerd[2315]:
time="2024-01-03T09:00:13.090576251Z" level=warning msg="cleaning up after shim disconnected"
id=ddd369c814ebe960f336e0d1c7035eae1d047c1bb6cabd297de58aa6b2cfaa88 namespace=k8s.io
Jan  3 09:00:14 aks-coordpool-35637789-vmss000029 containerd[2315]:
time="2024-01-03T09:00:14.306173686Z" level=warning msg="cleaning up after shim disconnected"
id=8c3c49e02a5e931ee633425411fc1acc71ab638689acc07bf72a86998a5fbea5 namespace=k8s.io
Jan  3 09:00:14 aks-coordpool-35637789-vmss000029 networkd-dispatcher[797]: ERROR:Failed to get
interface "azv1e2219415b4" status: Command '['/usr/bin/networkctl', 'status', '--no-pager',
'--no-legend', '--', 'azv1e2219415b4']' returned non-zero exit status 1.
Jan  3 09:00:19 aks-coordpool-35637789-vmss000029 containerd[2315]:
time="2024-01-03T09:00:19.679888702Z" level=warning msg="Failed to get podSandbox status for
container event for sandboxID
\"331706f2c30211f84214798ee8c2561344ec84a1c76cdd24615811fa0267edcd\": an error occurred when try
to find sandbox: not found. Sending the event with nil podSandboxStatus."
...
```

Another option to check the logs could be via the `journalctl` command:

```
$ journalctl -p err -b

Nov 27 13:11:53 localhost.localdomain kernel: RETBleed: WARNING: Spectre v2 mitigation leaves
CPU vulnerable to RETBleed attacks, data leaks possible!
Nov 27 13:11:55 localhost.localdomain dhclient[494]: execve (/bin/true, ...): Permission denied
Nov 27 13:11:55 localhost.localdomain dhclient[495]: execve (/bin/true, ...): Permission denied
Nov 27 13:11:55 localhost.localdomain dhclient[490]: Timeout too large reducing to: 2147483646
(TIME_MAX - 1)
Nov 27 13:31:26 localhost.localdomain dhclient[533]: execve (/bin/true, ...): Permission denied
Nov 27 13:31:26 localhost.localdomain dhclient[534]: execve (/bin/true, ...): Permission denied
...
Jan 01 09:00:14 aks-coordpool-35637789-vmss000029 networkctl[3331613]: Interface
"azv31917621732" not found.
Jan 02 02:00:39 aks-coordpool-35637789-vmss000029 networkctl[401048]: Interface "azv8cb7e72e3a2"
not found.
Jan 02 02:00:57 aks-coordpool-35637789-vmss000029 kernel: Memory cgroup out of memory: Killed
process 3459472 (java) total-vm:12720132kB, anon-rss:9500160kB, file-rss:48848kB, shmem-rss:0kB,
UID:999 pgtables:19676kB oom_score_adj:296
Jan 02 02:00:57 aks-coordpool-35637789-vmss000029 networkctl[401599]: Interface "azvc4b5732e431"
not found.
Jan 02 09:31:24 aks-coordpool-35637789-vmss000029 kernel: Memory cgroup out of memory: Killed
process 403863 (java) total-vm:12445740kB, anon-rss:10685424kB, file-rss:48992kB, shmem-rss:0kB,
UID:999 pgtables:21956kB oom_score_adj:296
Jan 02 09:33:00 aks-coordpool-35637789-vmss000029 networkctl[967900]: Interface "azve5326b3b584"
not found.
Jan 02 09:34:36 aks-coordpool-35637789-vmss000029 networkctl[970484]: Interface "azv855170f72c0"
not found.
```

Alternatively, monitor the log live by using:

```
$ journalctl -f

Jan 03 12:28:25 aks-coordpool-35637789-vmss000029 kubelet[2475]: I0103 12:28:25.356976    2475
prober.go:107] "Probe failed" probeType="Startup" pod="default/dremio-master-0"
podUID=1ad50710-cf2d-4883-be58-49b94418fa3f containerName="dremio-master-coordinator"
probeResult=failure output="Get \"http://10.240.0.25:9047/\": dial tcp 10.240.0.25:9047:
connect: connection refused"
Jan 03 12:28:26 aks-coordpool-35637789-vmss000029 kubelet[2475]: I0103 12:28:26.356571    2475
prober.go:107] "Probe failed" probeType="Startup" pod="default/dremio-master-0"
podUID=1ad50710-cf2d-4883-be58-49b94418fa3f containerName="dremio-master-coordinator"
probeResult=failure output="Get \"http://10.240.0.25:9047/\": dial tcp 10.240.0.25:9047:
connect: connection refused"
Jan 03 12:28:29 aks-coordpool-35637789-vmss000029 kubelet[2475]: I0103 12:28:29.356478    2475
prober.go:107] "Probe failed" probeType="Startup" pod="default/dremio-master-0"
podUID=1ad50710-cf2d-4883-be58-49b94418fa3f containerName="dremio-master-coordinator"
probeResult=failure output="Get \"http://10.240.0.25:9047/\": dial tcp 10.240.0.25:9047:
connect: connection refused"
Jan 03 12:28:30 aks-coordpool-35637789-vmss000029 kubelet[2475]: I0103 12:28:30.356503    2475
prober.go:107] "Probe failed" probeType="Startup" pod="default/dremio-master-0"
podUID=1ad50710-cf2d-4883-be58-49b94418fa3f containerName="dremio-master-coordinator"
probeResult=failure output="Get \"http://10.240.0.25:9047/\": dial tcp 10.240.0.25:9047:
connect: connection refused"
Jan 03 12:28:31 aks-coordpool-35637789-vmss000029 kubelet[2475]: I0103 12:28:31.356885    2475
prober.go:107] "Probe failed" probeType="Startup" pod="default/dremio-master-0"
podUID=1ad50710-cf2d-4883-be58-49b94418fa3f containerName="dremio-master-coordinator"
probeResult=failure output="Get \"http://10.240.0.25:9047/\": dial tcp 10.240.0.25:9047:
connect: connection refused"
Jan 03 12:28:32 aks-coordpool-35637789-vmss000029 kubelet[2475]: I0103 12:28:32.771483    2475
kubelet.go:2447] "SyncLoop (probe)" probe="startup" status="started"
pod="default/dremio-master-0"
Jan 03 12:28:32 aks-coordpool-35637789-vmss000029 kubelet[2475]: I0103 12:28:32.788678    2475
kubelet.go:2447] "SyncLoop (probe)" probe="readiness" status="ready"
pod="default/dremio-master-0"
```

Verify Node Events, Capacity and Status

The following command returns all information about a node. Please see the comments inlined at the highlighted positions.

```
$ kubectl describe node aks-executorpool-11591932-vmss00002u

Name:               aks-executorpool-11591932-vmss00002u
Roles:              agent
Labels:             agentpool=executorpool
                    beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/instance-type=Standard_D16s_v3
```

```
                    beta.kubernetes.io/os=linux
                    failure-domain.beta.kubernetes.io/region=westeurope
                    failure-domain.beta.kubernetes.io/zone=westeurope-1
                    kubernetes.azure.com/agentpool=executorpool
                    kubernetes.azure.com/cluster=MC_ps-healthcheck_healthcheck-aks_westeurope

kubernetes.azure.com/consolidated-additional-properties=de803661-889b-11ee-9558-b24c5f993228

kubernetes.azure.com/kubelet-identity-client-id=56796114-33dd-41da-be42-cc3f7b74c1aa
                    kubernetes.azure.com/mode=user
                    kubernetes.azure.com/network-policy=none

kubernetes.azure.com/node-image-version=AKSUbuntu-2204gen2containerd-202310.31.0
                    kubernetes.azure.com/nodepool-type=VirtualMachineScaleSets
                    kubernetes.azure.com/os-sku=Ubuntu
                    kubernetes.azure.com/role=agent
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=aks-executorpool-11591932-vmss00002u
                    kubernetes.io/os=linux
                    kubernetes.io/role=agent
                    node-role.kubernetes.io/agent=
                    node.kubernetes.io/instance-type=Standard_D16s_v3
                    topology.disk.csi.azure.com/zone=westeurope-1
                    topology.kubernetes.io/region=westeurope
                    topology.kubernetes.io/zone=westeurope-1
Annotations:        node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Wed, 06 Dec 2023 14:53:14 +0100
Taints:             node.kubernetes.io/unreachable:NoExecute
                    node.kubernetes.io/unreachable:NoSchedule
Unschedulable:      false
Lease:
  HolderIdentity:  aks-executorpool-11591932-vmss00002u
  AcquireTime:     <unset>
  RenewTime:       Tue, 02 Jan 2024 17:12:49 +0100
Addresses:
  InternalIP:  10.240.0.179
  Hostname:    aks-executorpool-11591932-vmss00002u
Capacity:
  cpu:                16
  ephemeral-storage:  129886128Ki
  hugepages-1Gi:      0
  hugepages-2Mi:      0
  memory:             65843096Ki
  pods:               30
# What are the CPU, memory and number of pods that can be allocated to the node.
# The node capacity is slightly higher and above, but some resources need to be
# reserved for the OS and Kubernetes.
# Sometimes node count is lower, because there is only limited capacity on internal
```

```
# IP addresses.
Allocatable:
  cpu:                15740m
  ephemeral-storage:  119703055367
  hugepages-1Gi:      0
  hugepages-2Mi:      0
  memory:             59329432Ki
  pods:               30
System Info:
  Machine ID:                 a766b252d9d64f57ba95942eca9825e2
  System UUID:                22b29ffe-ffb4-4be1-82b1-5092d72a734b
  Boot ID:                    c939fdf0-0f87-4163-a2c1-ce4d1c4d9ace
  Kernel Version:             5.15.0-1051-azure
  OS Image:                   Ubuntu 22.04.3 LTS
  Operating System:           linux
  Architecture:               amd64
  Container Runtime Version:  containerd://1.7.5-1
  Kubelet Version:            v1.27.3
  Kube-Proxy Version:         v1.27.3
ProviderID:
azure:///subscriptions/73d6bf3b-48e3-4809-9e31-e0807af07cff/resourceGroups/mc_ps-healthcheck_hea
lthcheck-aks_westeurope/providers/Microsoft.Compute/virtualMachineScaleSets/aks-executorpool-115
91932-vmss/virtualMachines/102
Non-terminated Pods:          (7 in total)
# What else is running on the same node?
# This can be summed up to get an understanding how much resources are bound to
# system or other pods. Depending on other services like Microsoft Defender,
# any kind of policy enforcement or extensions, the number of pods can vary.
  Namespace                   Name                                                CPU Requests
CPU Limits  Memory Requests  Memory Limits  Age
  ---------                   ----                                                ------------
----------  ---------------  -------------  ---
  default                     dremio-executor-1                                   14 (88%)      0
(0%)      56Gi (98%)       0 (0%)         27d
  kube-system                 azure-ip-masq-agent-gl9fq                           100m (0%)
500m (3%)   50Mi (0%)        250Mi (0%)     27d
  kube-system                 cloud-node-manager-zdjlw                            50m (0%)      0
(0%)      50Mi (0%)        512Mi (0%)     27d
  kube-system                 csi-azuredisk-node-vw2jb                            30m (0%)      0
(0%)      60Mi (0%)        400Mi (0%)     27d
  kube-system                 csi-azurefile-node-dr4m8                            30m (0%)      0
(0%)      60Mi (0%)        600Mi (1%)     27d
  kube-system                 kube-proxy-8xg95                                    100m (0%)     0
(0%)      0 (0%)           0 (0%)         27d
  kube-system                 node-shell-da219e81-1187-4e7f-bd9b-62ab882c7223     0 (0%)        0
(0%)      0 (0%)           0 (0%)         40s
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource           Requests       Limits
```

```
  --------           --------       ------
  cpu                14310m (90%)   500m (3%)
  memory             57564Mi (99%)  1762Mi (3%)
  ephemeral-storage  0 (0%)         0 (0%)
  hugepages-1Gi      0 (0%)         0 (0%)
  hugepages-2Mi      0 (0%)         0 (0%)
# Node events. The node was not ready 37 seconds ago
Events:
  Type    Reason        Age   From             Message
  ----    ------        ----  ----             -------
  Normal  NodeNotReady  37s   node-controller  Node aks-executorpool-11591932-vmss00002u status
is now: NodeNotReady
```

Verify Kubernetes Events

The following command returns cluster wide events. Unfortunately, this kind of event is only stored for a very short time based on the Kubernetes configuration and setup (usually 1 to 3 hours). It is a good idea to store the events in a third-party system.

```
$ kubectl get events

LAST SEEN   TYPE      REASON                   OBJECT
MESSAGE
60m         Normal    NodeNotReady             node/aks-executorpool-11591932-vmss00002u   Node
aks-executorpool-11591932-vmss00002u status is now: NodeNotReady
59m         Warning   ContainerdStart          node/aks-executorpool-11591932-vmss00002u
Starting containerd container runtime...
59m         Warning   KubeletIsDown            node/aks-executorpool-11591932-vmss00002u   Node
condition KubeletProblem is now: True, reason: KubeletIsDown, message: "Kubelet service is not
running"
# Node is not ready to host any pods. This could have caused a service disruption.
59m         Warning   KubeletIsDown            node/aks-executorpool-11591932-vmss00002u
Kubelet service is not running
59m         Warning   ContainerRuntimeIsDown   node/aks-executorpool-11591932-vmss00002u   Node
condition ContainerRuntimeProblem is now: True, reason: ContainerRuntimeIsDown, message:
"containerd service is not running"
59m         Warning   ContainerRuntimeIsDown   node/aks-executorpool-11591932-vmss00002u
containerd service is not running
59m         Normal    Starting                 node/aks-executorpool-11591932-vmss00002u
Starting kubelet.
59m         Warning   InvalidDiskCapacity      node/aks-executorpool-11591932-vmss00002u
invalid capacity 0 on image filesystem
59m         Normal    NodeHasSufficientMemory  node/aks-executorpool-11591932-vmss00002u   Node
aks-executorpool-11591932-vmss00002u status is now: NodeHasSufficientMemory
59m         Normal    NodeHasNoDiskPressure    node/aks-executorpool-11591932-vmss00002u   Node
aks-executorpool-11591932-vmss00002u status is now: NodeHasNoDiskPressure
59m         Normal    NodeHasSufficientPID     node/aks-executorpool-11591932-vmss00002u   Node
```

```
aks-executorpool-11591932-vmss00002u status is now: NodeHasSufficientPID
59m         Normal    NodeAllocatableEnforced   node/aks-executorpool-11591932-vmss00002u
Updated Node Allocatable limit across pods
59m         Warning   Rebooted                  node/aks-executorpool-11591932-vmss00002u   Node
aks-executorpool-11591932-vmss00002u has been rebooted, boot id:
2f7c27bc-5a8d-45ba-9664-d977d7175051
59m         Normal    NodeReady                 node/aks-executorpool-11591932-vmss00002u   Node
aks-executorpool-11591932-vmss00002u status is now: NodeReady
59m         Normal    Starting                  node/aks-executorpool-11591932-vmss00002u
59m         Normal    KubeletIsUp               node/aks-executorpool-11591932-vmss00002u   Node
condition KubeletProblem is now: False, reason: KubeletIsUp, message: "kubelet service is up"
59m         Normal    ContainerRuntimeIsUp      node/aks-executorpool-11591932-vmss00002u   Node
condition ContainerRuntimeProblem is now: False, reason: ContainerRuntimeIsUp, message:
"container runtime service is up"
# An executor pod was hosted on the node and this could have led to failing
# queries 60 minutes ago.
60m         Warning   NodeNotReady              pod/dremio-executor-1                       Node
is not ready
59m         Normal    TaintManagerEviction      pod/dremio-executor-1
Cancelling deletion of Pod default/dremio-executor-1
59m         Normal    SandboxChanged            pod/dremio-executor-1                       Pod
sandbox changed, it will be killed and re-created.
```

Verify IOPS Limits for Disks

Often, there is the question “Do I have enough IOPS on my disk, or do the processes and users wait?”. Some cloud providers provide embedded monitoring, which shows the IOPS consumption. Another useful metric is “queue depth”. Based on the number of caches, but usually if queue depth exceeds the value of 2 to 4, the IOPS have been exceeded. Please remember that monitoring tools often collect data in intervals and might not be able to see the peaks.

There are also ways to monitor a specific disk on the host node. Here is an example of how to monitor IOPS and queue depths on the Kubernetes host node:

First, get the coordinator or executor persistent volume name:

```
$ kubectl get pv

NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS
CLAIM                                                 STORAGECLASS   REASON   AGE
pvc-3af20ee0-fa18-4a04-bde5-41a193703021   8Gi        RWO            Delete           Bound
default/datadir-zk-0                                  managed                         36d
pvc-4346121a-3fe6-4981-8dbf-ed748f284c5d   128Gi      RWO            Delete           Bound
default/dremio-default-executor-c3-0-dremio-executor-1     managed-premium            49d
```

```
# We want to monitor the IOPS and Queue Depth of this disk.
pvc-9323b1fe-82f3-45d0-87f9-9ad9769116f3   128Gi      RWO            Delete           Bound
default/dremio-master-volume-dremio-master-0                  managed-premium            630d
pvc-be340d30-5427-4570-b824-95ab985a6601   128Gi      RWO            Delete           Bound
default/dremio-default-executor-c3-0-dremio-executor-0     managed-premium            497d
pvc-e3fb87d2-6af7-48a5-88f6-4ec8ee44ebe1   128Gi      RWO            Delete           Bound
default/dremio-default-executor-volume-dremio-executor-1   managed-premium            49d
pvc-ed96fa9c-7416-41a5-9db9-67e3c786e7a7   128Gi      RWO            Delete           Bound
default/dremio-default-executor-volume-dremio-executor-0   managed-premium            497d
```

Find out the host node of the pod:

```
$ kubectl get pods -o wide
NAME                                  READY   STATUS      RESTARTS   AGE     IP
NODE                                 NOMINATED NODE   READINESS GATES
dremio-executor-0                     1/1     Running     0          5h52m   10.240.0.11
aks-executorpool-11591932-vmss00002q   <none>           <none>
dremio-executor-1                     1/1     Running     0          29m     10.240.0.129
aks-executorpool-11591932-vmss00002v   <none>           <none>
# The pod is hosted on aks-coordpool-35637789-vmss000029
dremio-master-0                       1/1     Running     0          5h52m   10.240.0.29
aks-coordpool-35637789-vmss000029      <none>           <none>
zk-0                                  1/1     Running     0          30d     10.240.0.175
aks-agentpool-27649900-vmss00001l      <none>           <none>
```

Connect to the host node using:

```
$ kubectl debug node/aks-coordpool-35637789-vmss000029 --profile=netadmin -it
--image=dremiops.azurecr.io/dremio-k8s-tools:v1.15
```

Find the right disk which needs to be monitored. The disk name `pvc-9323b1fe-82f3-45d0-87f9-9ad9769116f3` is from the `kubectl get pv` command above. So we can see the disk we are looking for is sdc.

```
$ mount | grep pvc-9323b1fe-82f3-45d0-87f9-9ad9769116f3
/dev/sdc on
/var/lib/kubelet/pods/532a7907-99ae-491d-a210-3e9853fe0fc0/volumes/kubernetes.io~csi/pvc-9323b1f
e-82f3-45d0-87f9-9ad9769116f3/mount type ext4 (rw,relatime)
/dev/sdc on
/var/lib/kubelet/pods/532a7907-99ae-491d-a210-3e9853fe0fc0/volumes/kubernetes.io~csi/pvc-9323b1f
e-82f3-45d0-87f9-9ad9769116f3/mount type ext4 (rw,relatime)
```

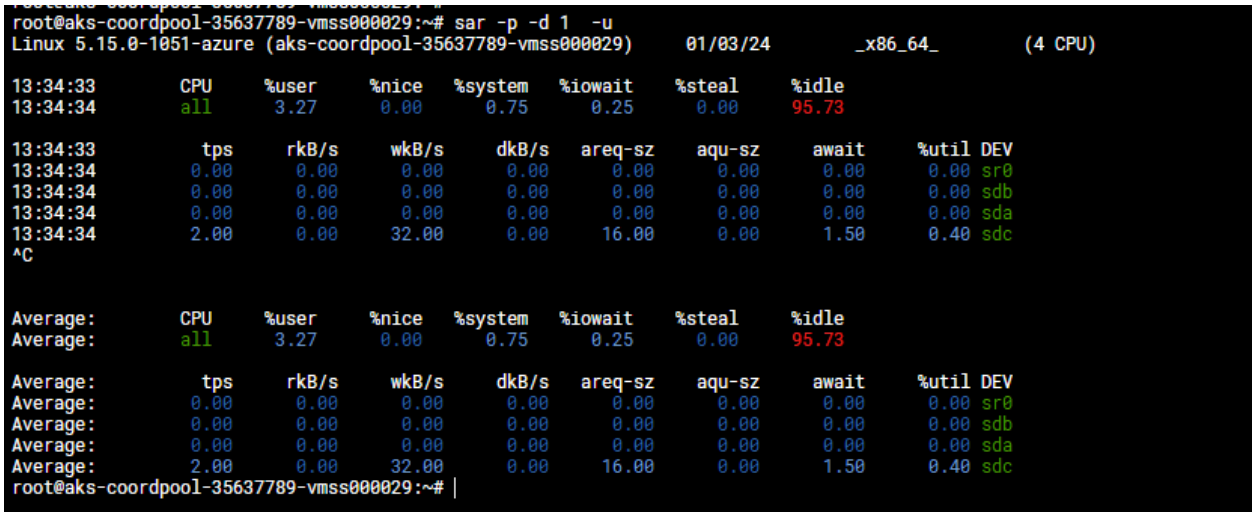
The command below returns the disk statistics every second. Once the command gets canceled using Ctrl+C, it prints the average for the entire time.

The two most important fields are:

- **tps**: Total number of transfers per second issued to physical devices. (IOPS)
- **aqu-sz**: The average queue length of the requests issued to the device. If the value exceeds 7, you might consider adding more IOPS to the disk.

```
$ sar -p -d 1  -u

Linux 5.15.0-1051-azure (aks-coordpool-35637789-vmss000029)     01/03/24        _x86_64_        (4 CPU)

13:34:33        CPU     %user     %nice   %system   %iowait    %steal     %idle
13:34:34        all      3.27      0.00      0.75      0.25      0.00     95.73

13:34:33          tps     rkB/s     wkB/s     dkB/s   areq-sz    aqu-sz     await     %util DEV
13:34:34         0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00 sr0
13:34:34         0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00 sdb
13:34:34         0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00 sda
13:34:34         2.00      0.00     32.00      0.00     16.00      0.00      1.50      0.40 sdc

Average:        CPU     %user     %nice   %system   %iowait    %steal     %idle
Average:        all      3.27      0.00      0.75      0.25      0.00     95.73

Average:          tps     rkB/s     wkB/s     dkB/s   areq-sz    aqu-sz     await     %util DEV
Average:         0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00 sr0
Average:         0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00 sdb
Average:         0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00 sda
Average:         2.00      0.00     32.00      0.00     16.00      0.00      1.50      0.40 sdc
```

Here is a nicer screenshot of the output above:



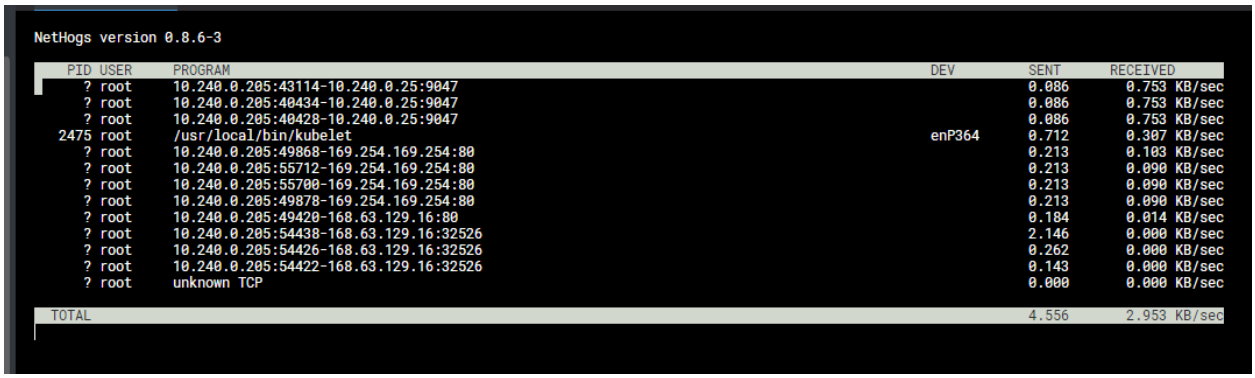
Monitor Network Throughput

Connect to the host node using the following command. Please note that the command requires the profile `netadmin`.

```
$ kubectl debug node/aks-coordpool-35637789-vmss000029 --profile=netadmin -it
--image=dremiops.azurecr.io/dremio-k8s-tools:v1.15
```

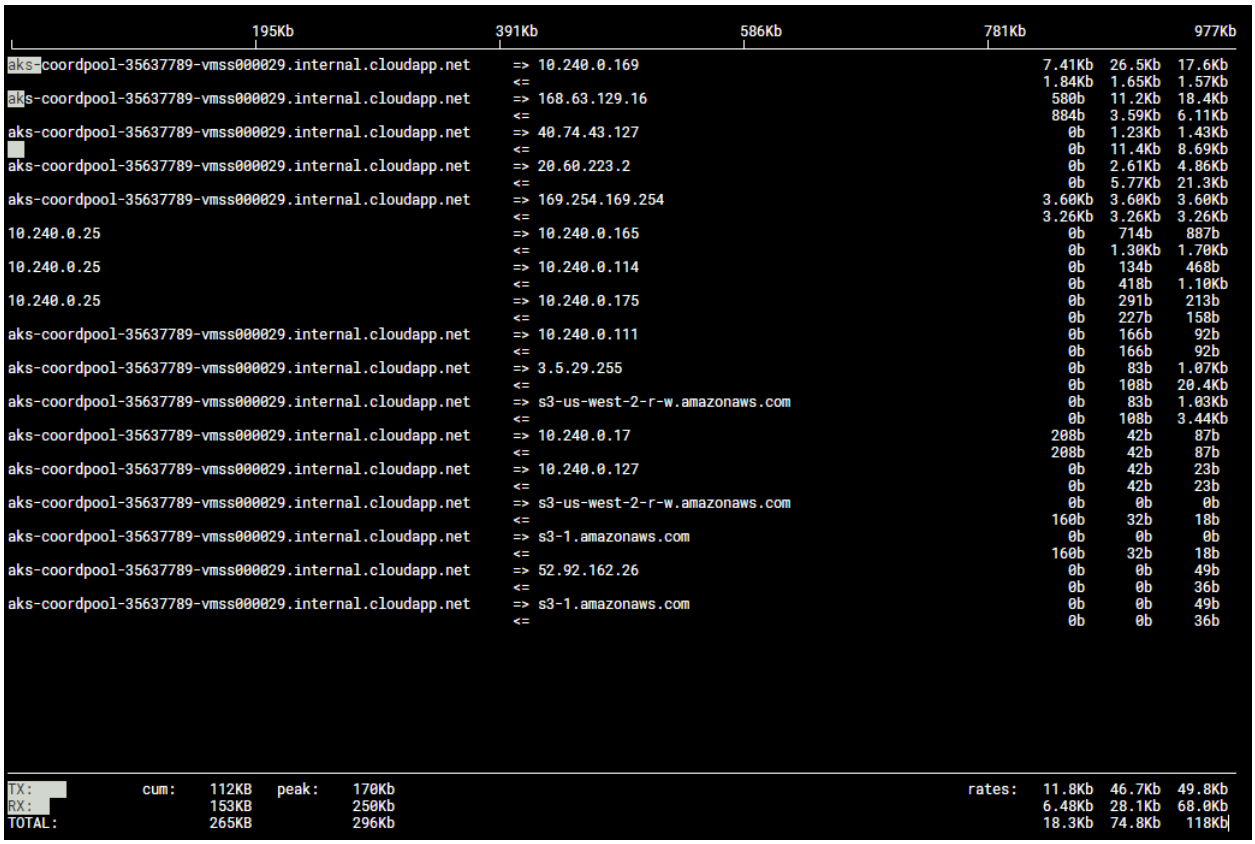
`Nethogs` could be used to monitor the throughput of the processes on a node. Start `nethogs` using:

```
$ nethogs
```



Alternatively, `iftop` can be used to monitor the traffic of a Kubernetes node.

```
$ iftop
```



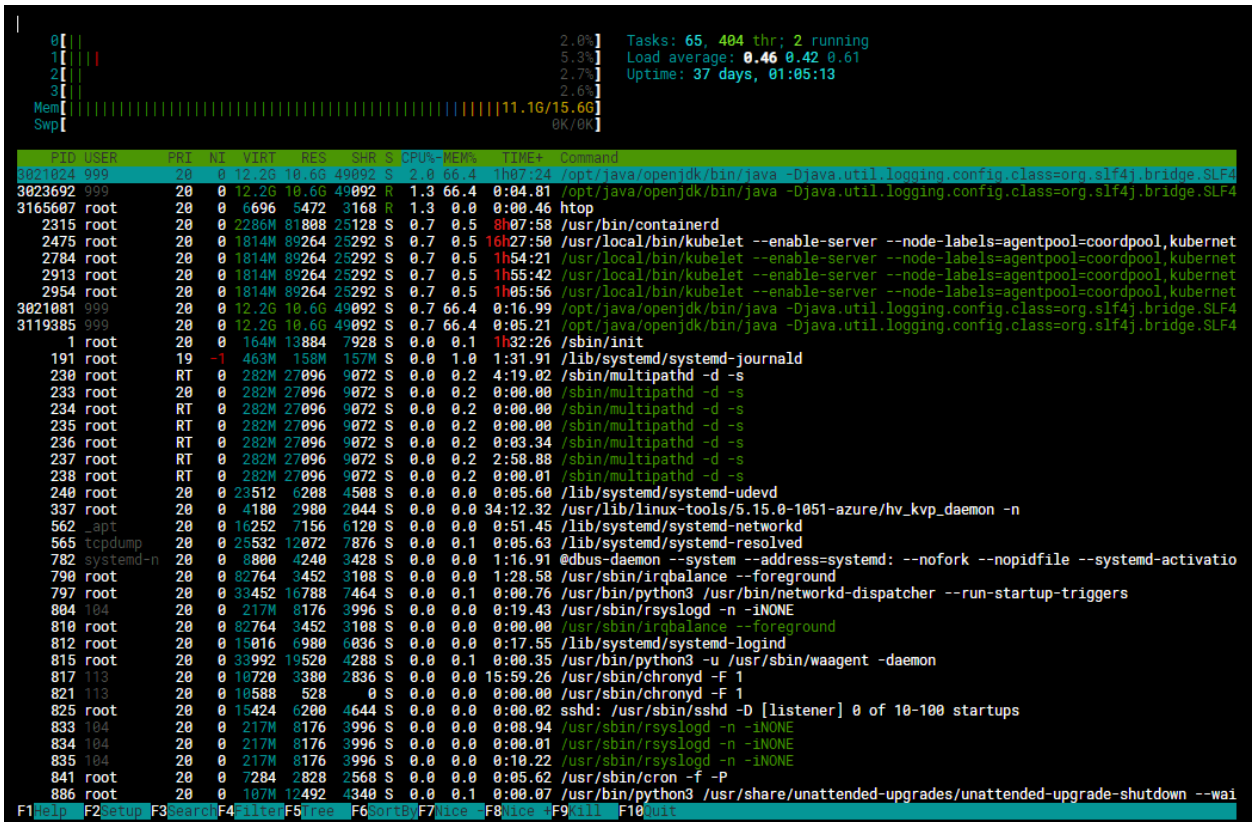
Monitor Node CPU and Memory Usage

Connect to the host node using the following command.

```
$ kubectl debug node/aks-coordpool-35637789-vmss000029 --profile=netadmin -it
--image=dremiops.azurecr.io/dremio-k8s-tools:v1.15
```

Start `htop` using the following command:

```
$ htop
```



Ensure and Validate Coordinator Backups

Ensure you run daily backups of Dremio's Metadata store, which is hosted on the Dremio coordinator/master node. This contains all the metadata and the work you have put into your SQL worksheet and virtual datasets.

A common way of running backups in Kubernetes is by using Kubernetes CronJobs. Regularly validate that backups are completed successfully.

Generally, there are two ways of doing backups using a Kubernetes CronJob:

- The `dremio-admin` command can backup the files directly to the data lake. The advantage is that this approach stores the backup in a completely different technology outside the Kubernetes cluster. Using this approach can put lots of pressure on the coordinator node. The backups should only run when the cluster utilization is low.
- Kubernetes Volume Snapshots can be used. Snapshots are an efficient way and should not affect the operating business. Snapshots are registered within the same Kubernetes cluster. So if a Kubernetes cluster gets lost, the snapshots might be too.

To learn more about the implementation details of backups, please follow this link.

Keep Kubernetes Up-To-Date and Avoid EOL

The cloud vendors move quickly forward with new Kubernetes versions. It is important to keep Kubernetes up-to-date to retrieve the latest security updates.

How to determine the Kubernetes version

This command shows which Kubernetes version is used on your Kubernetes cluster:

```
$ kubectl get nodes

NAME                                STATUS   ROLES   AGE   VERSION
aks-agentpool-27649900-vmss000011   Ready    agent   41d   v1.27.3
aks-coordpool-35637789-vmss000029   Ready    agent   36d   v1.27.3
aks-executorpool-11591932-vmss00002q   Ready    agent   41d   v1.27.3
aks-executorpool-11591932-vmss00002u   Ready    agent   27d   v1.27.3
```

Supported Cloud Kubernetes Versions / EOL

You should also ensure that the used Kubernetes version is not end-of-life so that you will still receive support and security updates. The cloud vendors provide excellent documentation about the lifecycle of their Kubernetes offerings:

- Azure - AKS
- AWS - EKS
- Google Cloud - GKE
- On-premises - Please consult your Kubernetes vendor.

Kubernetes Upgrade Instructions

- Azure - AKS
  AWS - EKS
- Google Cloud - GKE
- On-premises - Please consult your Kubernetes vendor.

Keep Dremio Up-To-Date

Update Dremio Helm Charts

The Dremio Helm Charts can be found here.

The Helm Charts are maintained together with Dremio Software release cycles. You should check this Git repository regularly, especially before performing a Dremio upgrade.

> ⚠ NOTICE
> **Important:** In nearly all cases, customers maintain a copy of this repository in their version control system. Please clone the repository directly from the official repository and push the entire history into your own Git repository. This keeps the history, including all Git commit ids, and allows you to easily merge changes from the original repository. Please do not use the download function in GitHub or copy the plain files into your repository because this will break the entire Git history and it becomes impossible to merge in changes.

This is how changes from the original repository can be merged into yours:

```
# Add the original dremio-cloud-tools repository
$ git remote add dremio git@github.com:dremio/dremio-cloud-tools.git

# Show the added repository
$ git remote -v
dremio  git@github.com:dremio/dremio-cloud-tools.git (fetch)
dremio  git@github.com:dremio/dremio-cloud-tools.git (push)
origin  git@ssh.dev.azure.com:v3/myorg/myproject/dremio-cloud-tools (fetch)
origin  git@ssh.dev.azure.com:v3/myorg/myproject/dremio-cloud-tools (push)

# Fetch remote changes from the original repository
$ git fetch dremio
remote: Enumerating objects: 65, done.
remote: Counting objects: 100% (61/61), done.
remote: Compressing objects: 100% (50/50), done.
remote: Total 65 (delta 26), reused 36 (delta 11), pack-reused 4
Unpacking objects: 100% (65/65), 38.79 KiB | 182.00 KiB/s, done.
From github.com:dremio/dremio-cloud-tools
 * [new branch]      master     -> dremio/master

# Merge the changes from the origin repository
$ git merge dremio/master
Auto-merging charts/dremio_v2/templates/dremio-master.yaml
Auto-merging charts/dremio_v2/templates/dremio-executor.yaml
Auto-merging charts/dremio_v2/templates/dremio-coordinator.yaml
Auto-merging charts/dremio_v2/config/logback.xml

# Push the changes into your repository
$ git push origin master
```

Update the Dremio Helm Deployment

> ⚠ NOTICE
> **Important:** Before performing any upgrade, a backup must be taken. This can be done via volume snapshots or using the `dremio-admin` command. **An upgrade cannot be rolled back to a former version without having any backup.**

This is an example of a backup that can be taken in an Azure deployment:

```
$ kubectl exec -it dremio-master-0 -c dremio-master-coordinator -- bash -c
"DREMIO_JAVA_CLIENT_EXTRA_OPTS=-Xmx4G /opt/dremio/bin/dremio-admin backup -l -d
dremioAzureStorage://:///dremio-backup"
```

After taking a backup, the upgrade can be performed. More information on how to upgrade Helm Charts can be found here.