Solution Guide

MayaData

Deploying CockroachDB using OpenEBS.

OpenEBS

MayaData

Contents

Overview

CockroachDB is a cloud-native SQL database for building global, scalable cloud services that survive disasters. It is a distributed SQL database built on a transactional and strongly-consistent key-value store. It scales horizontally; survives disk, machine, rack, and even datacenter failures with minimal latency disruption and no manual intervention; supports strongly-consistent ACID transactions; and provides a familiar SQL API for structuring, manipulating, and querying data.[1][2].

This guide explains the basic installation for CockroachDB operators on OpenEBS Local PV devices. The guide will also provide a way to monitor the health of cockroachDB using Prometheus and Grafana tools.

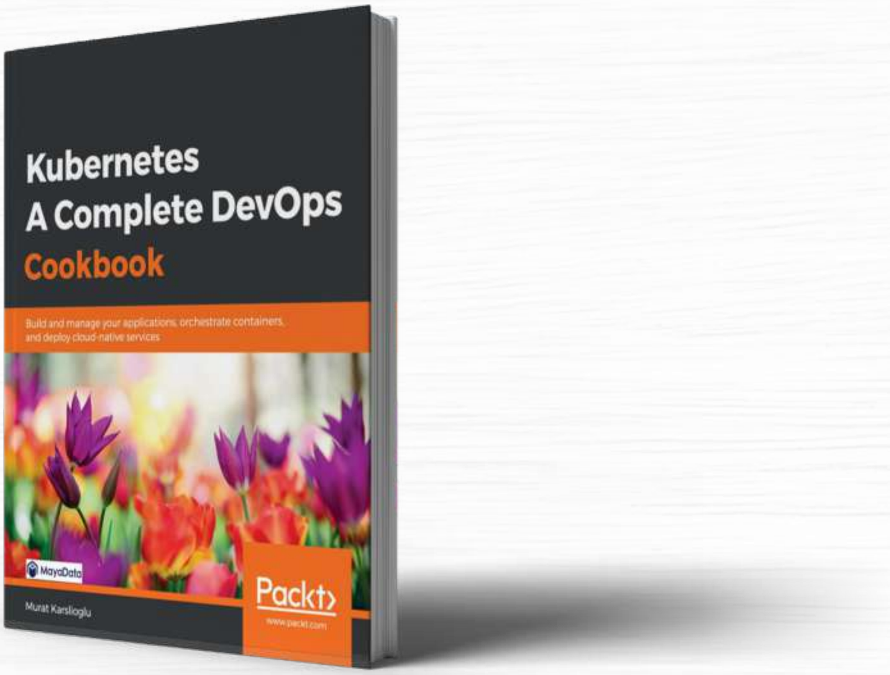
Before starting

You require an existing Kubernetes cluster. Kubernetes provides platform abstraction, cloud-native software runs, and behaves the same way on a managed Kubernetes service like AWS EKS, Google Cloud GKE, Microsoft AKS, DigitalOcean Kubernetes Service, or self-managed based on Red Hat OpenShift and Rancher. You can also use kubeadm, kubespray, minikube. Since you made it here, we assume you already have one configured.

MayaData team has proudly over 50 CKAs, years of experience building for enterprises, and running Kubernetes in production. If you need professional help to decide, we can connect you with one of our trusted partners. In case you want to learn more, just schedule a call [schedule a call] with us and we will send you a best-selling “Kubernetes - A Complete DevOps Cookbook,” also written by one of our own experts.

Free Book:

Kubernetes A Complete DevOps Cookbook


Kubernetes
A Complete DevOps
Cookbook
Build and manage your applications, orchestrate containers, and deploy cloud-native services
MayaData
Murat Karslioglu
Packt>
www.packt.com

Schedule a 15-minute call today to speak with one of our partners and receive a FREE copy of our new book.

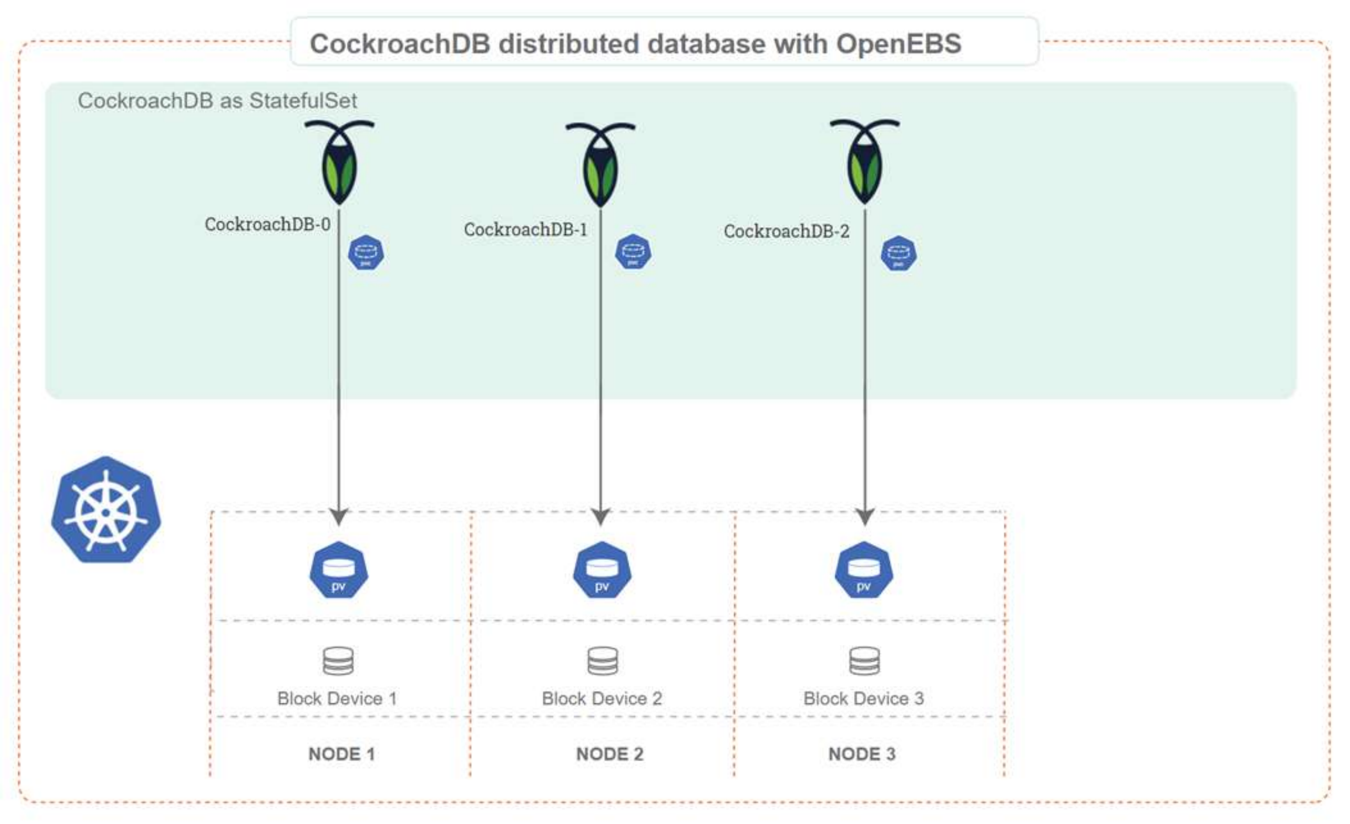
Perform pre-configuration

We will use GKE, where we will install CockroachDB with OpenEBS storage engine. The Local PV volume will be provisioned on a node where the CockroachDB pod is getting scheduled and uses one of the matching unclaimed block devices, which will then use the entire block device for storing data. No other application can use this device. If users have limited block devices attached to some nodes, they can use nodeSelector in the application YAML to provision applications on particular nodes where the available block device is present. The recommended configuration is to have at least three nodes and one unclaimed Local SSDs to be attached per node. Users can mention the required number of Local SSDs during the cluster creation time or provision the additional disks as described in the steps shown below.

As per CockroachDB's recommendation, it is better to use node-local storage instead of using external or replicated storage provisioners[2]. Since OpenEBS LocalPV Devices is using the unclaimed block device of the node where the application pod is getting scheduled, as mentioned above, it gives higher performance as compared to other storage provisioners.

Let's review our setup used for the configuration.

- 3 Nodes in GKE
- 4 vCPUs / node
- Ubuntu 18.04
- 16 GB memory / node
- 1 gpd with minimum 100Gi / node
- GCP instance type: e2-standard-4
- Kubernetes version: v1.18.15

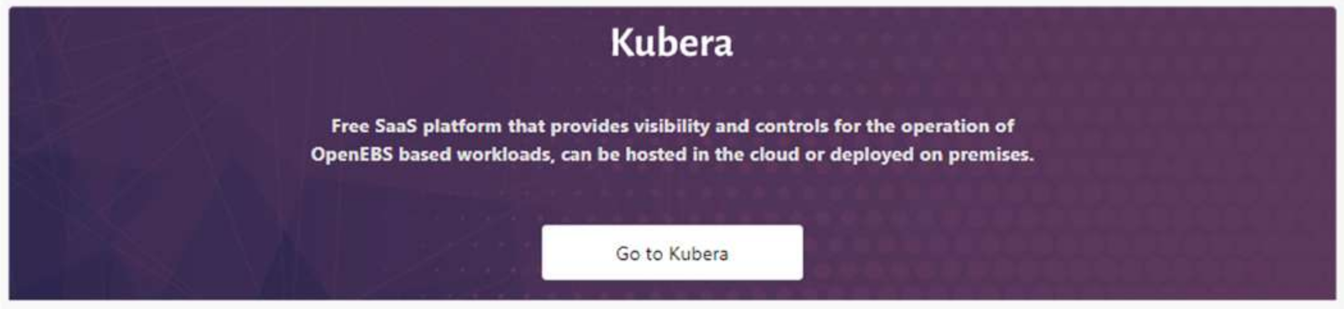


Getting Started with OpenEBS

Let's start the installation of OpenEBS using the Kubera platform.

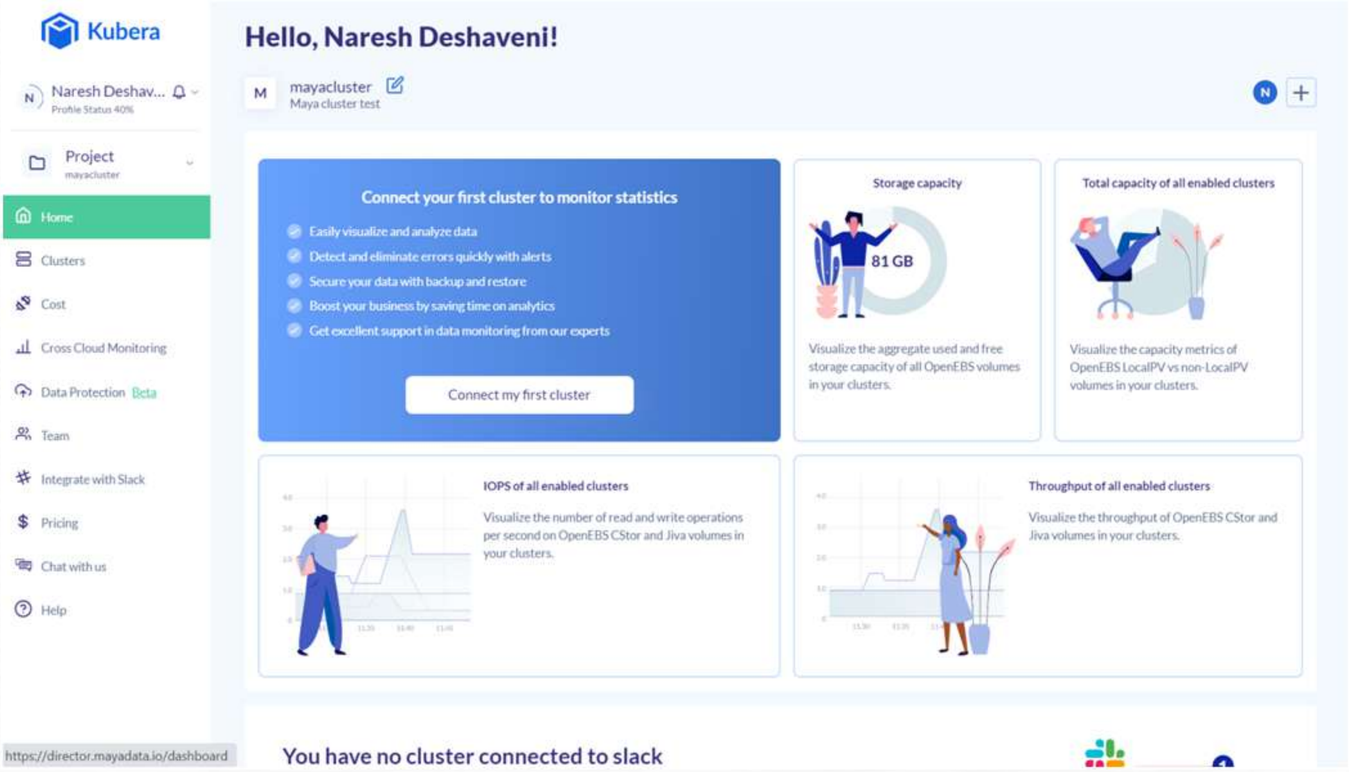
Installing OpenEBS using Kubera

Signup here for your free Kubera account. Then click on **Go to Kubera.**



Kubera

Free SaaS platform that provides visibility and controls for the operation of OpenEBS based workloads, can be hosted in the cloud or deployed on premises.



Hello, Naresh Deshaveni!

Connect your first cluster to monitor statistics

- Easily visualize and analyze data
- Detect and eliminate errors quickly with alerts
- Secure your data with backup and restore
- Boost your business by saving time on analytics
- Get excellent support in data monitoring from our experts

Storage capacity

Visualize the aggregate used and free storage capacity of all OpenEBS volumes in your clusters.

Total capacity of all enabled clusters

Visualize the capacity metrics of OpenEBS LocalPV vs non-LocalPV volumes in your clusters.

IOPS of all enabled clusters

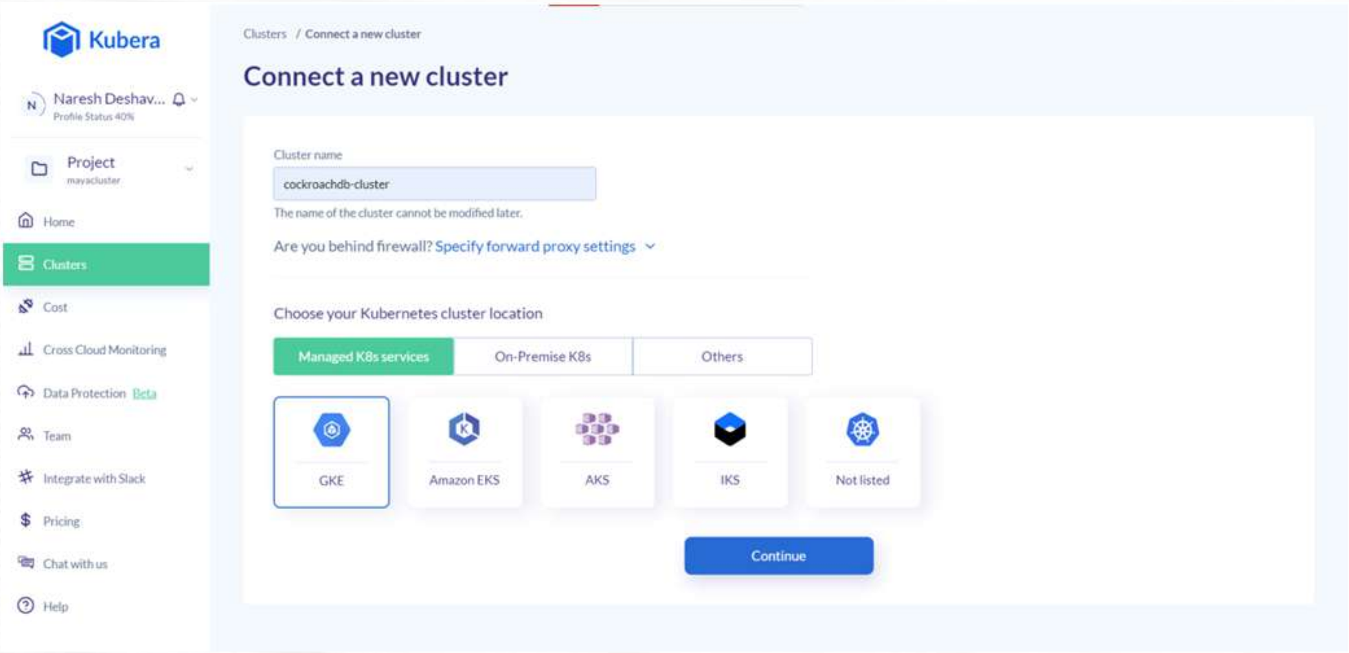
Visualize the number of read and write operations per second on OpenEBS CStor and Jiva volumes in your clusters.

Throughput of all enabled clusters

Visualize the throughput of OpenEBS CStor and Jiva volumes in your clusters.

You have no cluster connected to slack

Follow the instructions to connect your cluster to your Kubera account.



Clusters / Connect a new cluster

Connect a new cluster

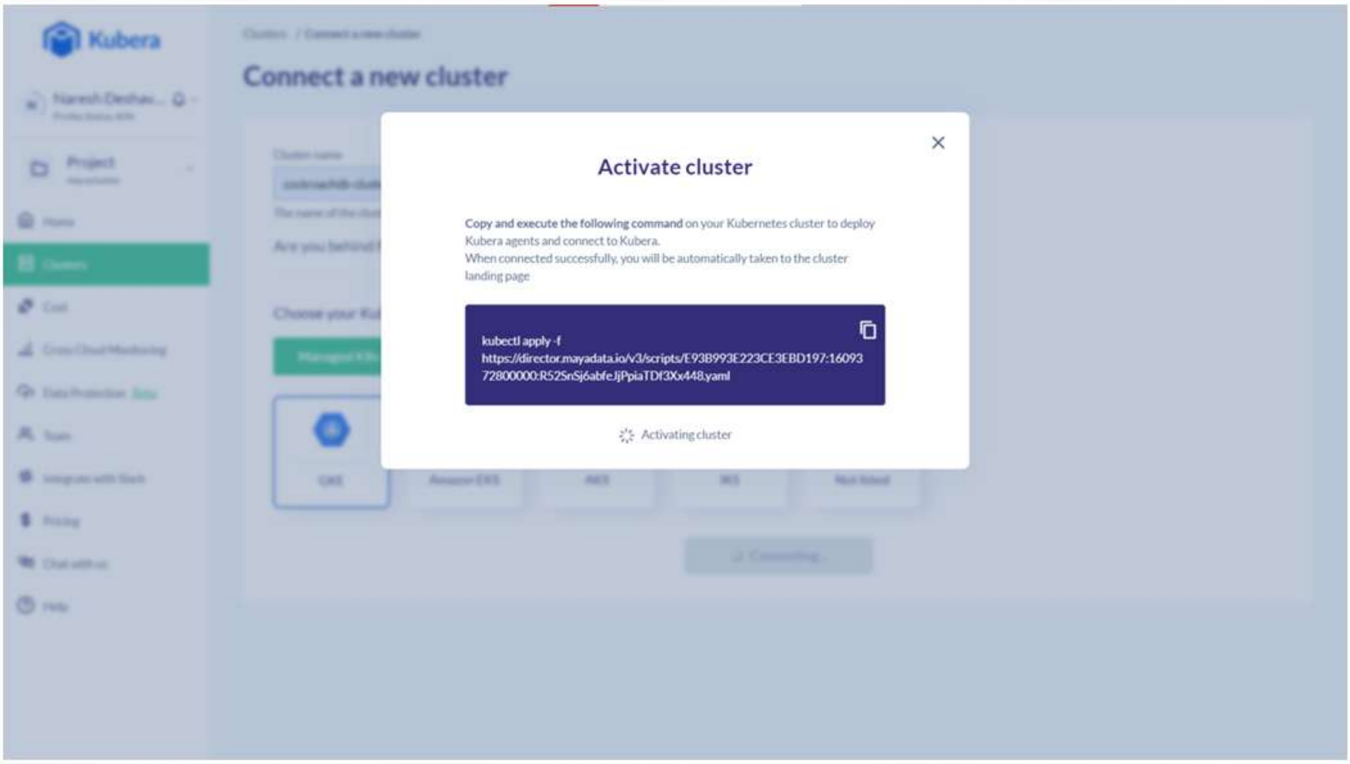
Cluster name: cockroachdb-cluster

The name of the cluster cannot be modified later.

Are you behind firewall? Specify forward proxy settings

Choose your Kubernetes cluster location

It will open a window with the command to connect your K8s cluster with the Kubera SaaS version. Copy and execute the command on your own Kubernetes cluster.



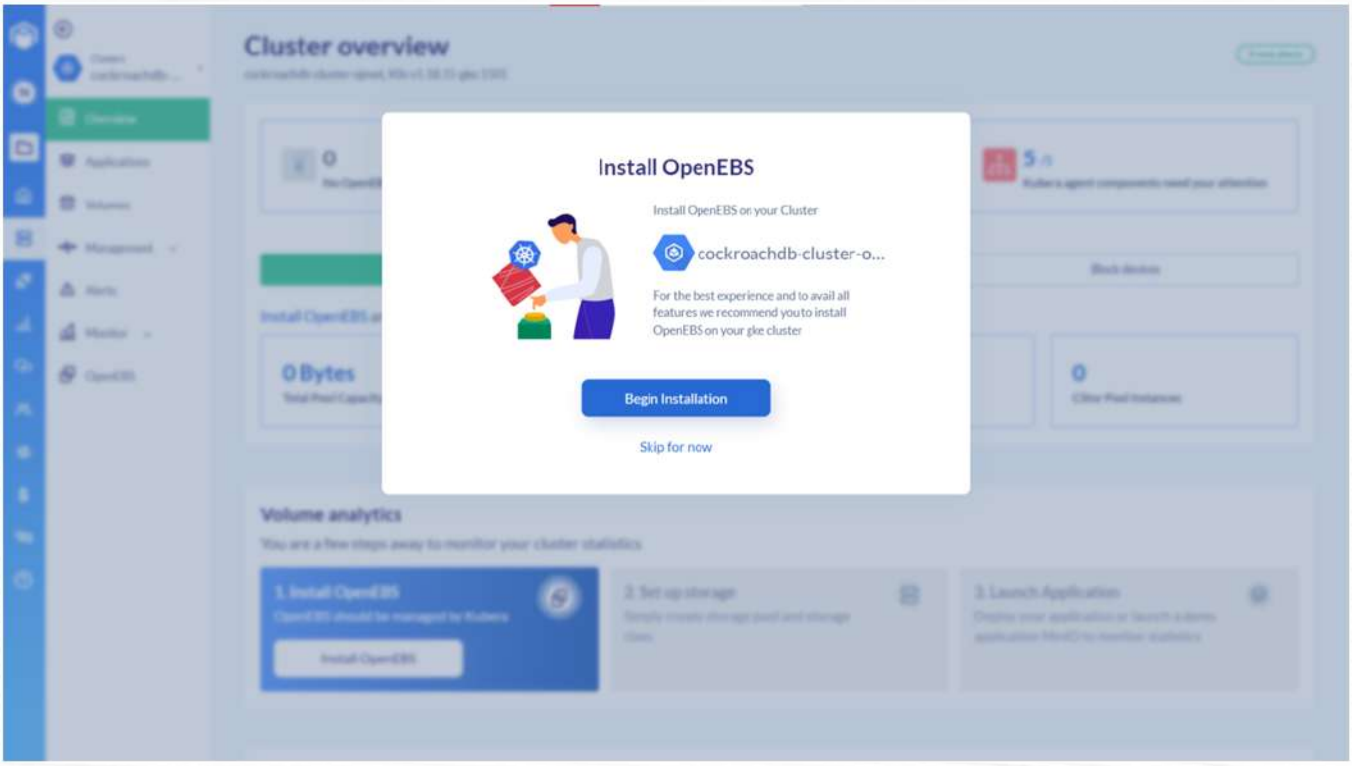
Activate cluster

Copy and execute the following command on your Kubernetes cluster to deploy Kubera agents and connect to Kubera.
When connected successfully, you will be automatically taken to the cluster landing page

```
kubectl apply -f
https://director.mayadata.io/v3/scripts/E93B993E223CE3EBD197:16093
72800000:R52SnSj6abfeJjPpiaTDf3Xx448.yaml
```

Activating cluster

If OpenEBS was already installed using Kubera in your cluster, skip this process. If OpenEBS was not installed using Kubera, then click on **Begin Installation**, which will lead to a page where you can choose how to install OpenEBS.



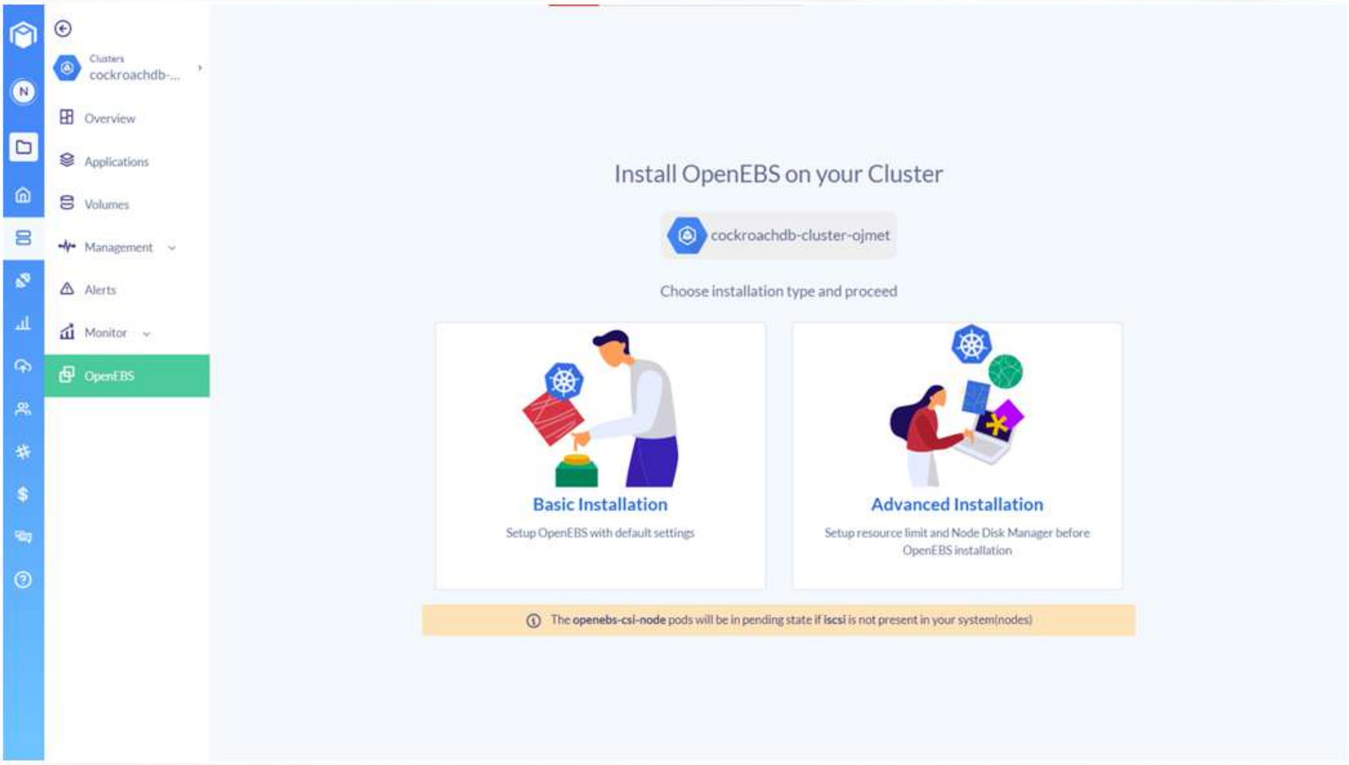
Install OpenEBS

Install OpenEBS on your Cluster

cockroachdb-cluster-o...

For the best experience and to avail all features we recommend you to install OpenEBS on your gke cluster

Follow the on-screen instructions titled **Basic Installation** for the default installation of OpenEBS Enterprise Edition on your K8s cluster.



Install OpenEBS on your Cluster

cockroachdb-cluster-ojmet

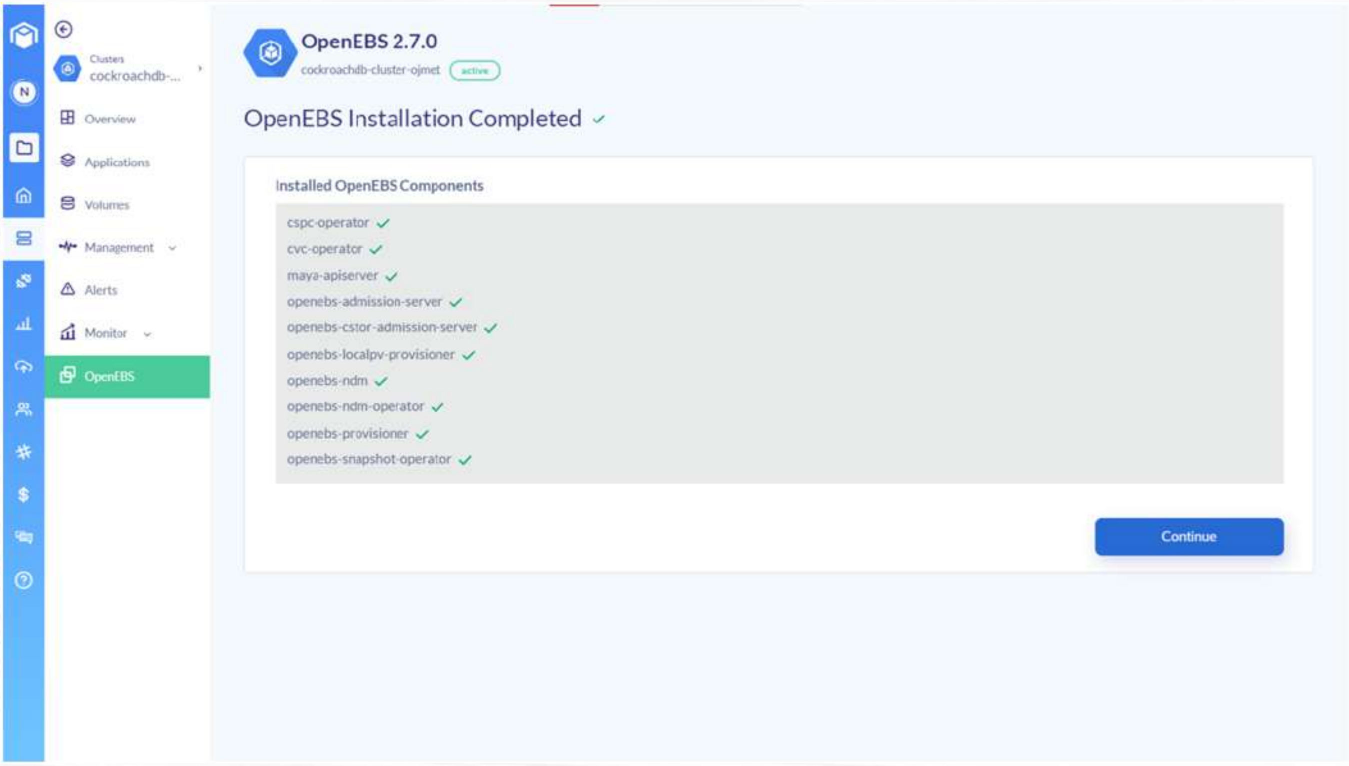
Choose installation type and proceed

Basic Installation
Setup OpenEBS with default settings

Advanced Installation
Setup resource limit and Node Disk Manager before OpenEBS installation

ⓘ The **openebs-csi-node** pods will be in pending state if **iscsi** is not present in your system(nodes)

Click on **Deploy OpenEBS** on the next screen and verify the installation status from the next screen. After successful installation of OpenEBS, click on **Continue.** If you run into any errors or have questions, **community support** for Kubera is available on Slack.



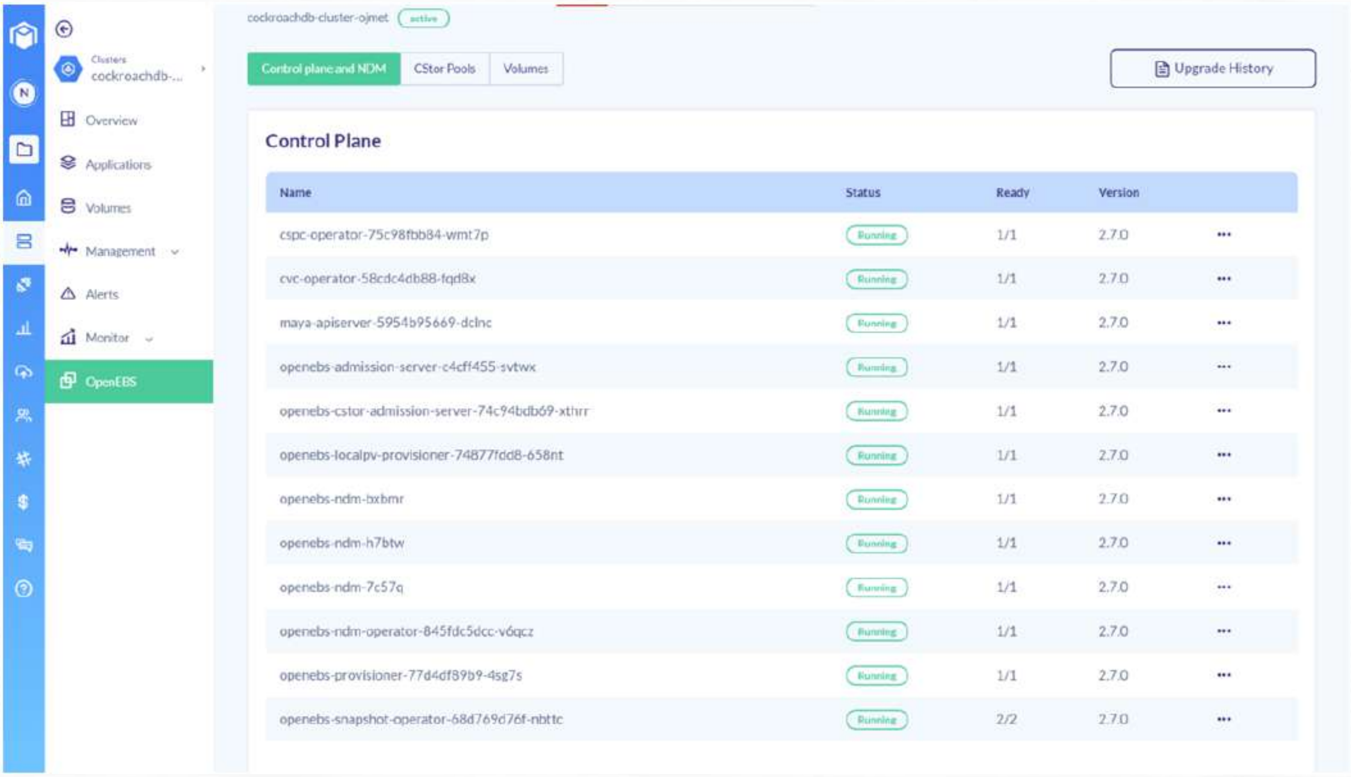
OpenEBS 2.7.0
cockroachdb-cluster-ojmet active

OpenEBS Installation Completed ✓

Installed OpenEBS Components

- cspc-operator ✓
- cvc-operator ✓
- maya-apiserver ✓
- openebs-admission-server ✓
- openebs-cstor-admission-server ✓
- openebs-localpv-provisioner ✓
- openebs-ndm ✓
- openebs-ndm-operator ✓
- openebs-provisioner ✓
- openebs-snapshot-operator ✓

Now, you will see OpenEBS control-plane has been enabled on your Kubernetes cluster.



Control Plane

Name	Status	Ready	Version	
cspc-operator-75c98fbb84-wmt7p	Running	1/1	2.7.0	•••
cvc-operator-58cdc4db88-fqd8x	Running	1/1	2.7.0	•••
maya-apiserver-5954b95669-dclnc	Running	1/1	2.7.0	•••
openebs-admission-server-c4cff455-svtwx	Running	1/1	2.7.0	•••
openebs-cstor-admission-server-74c94bdb69-xthrr	Running	1/1	2.7.0	•••
openebs-localpv-provisioner-74877fdd8-658nt	Running	1/1	2.7.0	•••
openebs-ndm-bxbmr	Running	1/1	2.7.0	•••
openebs-ndm-h7btw	Running	1/1	2.7.0	•••
openebs-ndm-7c57q	Running	1/1	2.7.0	•••
openebs-ndm-operator-845fdc5dcc-v6qcz	Running	1/1	2.7.0	•••
openebs-provisioner-77d4df89b9-4sg7s	Running	1/1	2.7.0	•••
openebs-snapshot-operator-68d769d76f-nbttc	Running	2/2	2.7.0	•••

Configuring GCP Project

If you are on GCP, you need to select your project before you can attach disks to the nodes.

```
$ gcloud config set project <your-project-name-here>
```

Create 1 100Gi disks for each node.

```
$ gcloud compute disks create disk-1 disk-2 disk-3 --size=100G
--zone=us-central1-c
```

Note: Provide the required size initially as currently Local PV volume will not allow you to expand the capacity later.

Attaching disks to each Node

Now, we will add 1 disk to each node. Disks will be later consumed by CockroachDB. This step can be done through your cloud vendor's web user interface, or if you are running in a VM, you can use your hypervisor to add 1 additional virtual device to each node. In this example, we have used GCP and added the disks using the gcloud CLI tool.

Get the list of Instance IDs per each Zone

```
$ gcloud compute instances list --zones us-central1-c
NAME  ZONE          MACHINE_TYPE   PREEMPTIBLE  INTERNAL_IP    EXTERNAL_IP
STATUS
gke-openebs-cockroachdb-default-pool-fbceb18c-j9pl  us-central1-c  e2-standard-4
10.128.0.62    35.224.42.110  RUNNING
gke-openebs-cockroachdb-default-pool-fbceb18c-kq41  us-central1-c  e2-standard-4
10.128.0.61    34.121.88.146  RUNNING
gke-openebs-cockroachdb-default-pool-fbceb18c-nh13  us-central1-c  e2-standard-4
10.128.15.192  35.184.99.128  RUNNING
```

Now, attach the disks to each node.

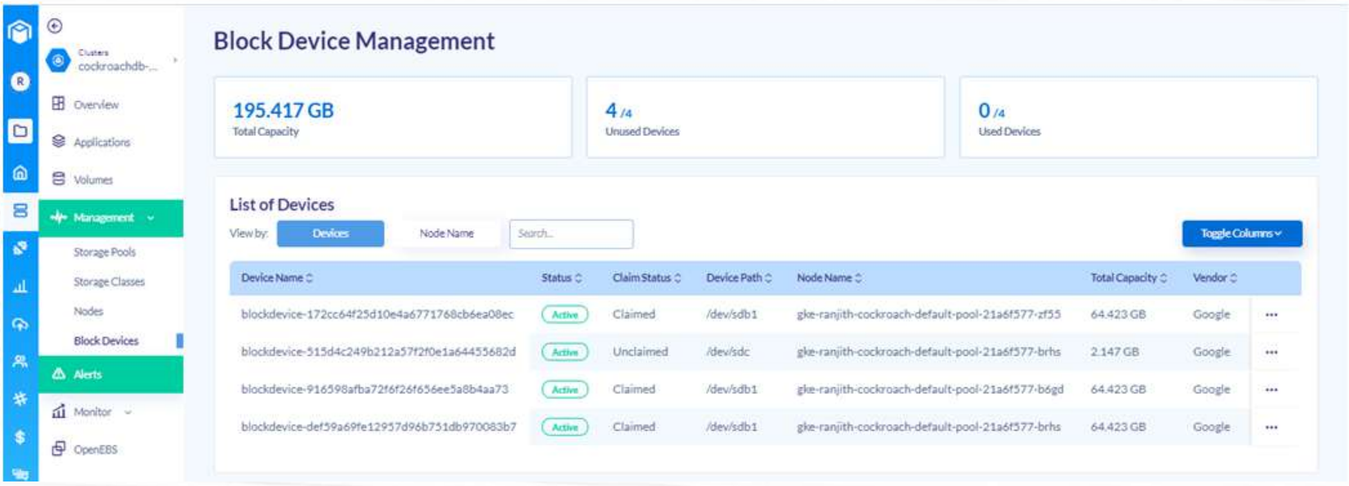
```
$ gcloud compute instances attach-disk gke-openebs-cockroachdb-default-pool-fbceb18c-j9pl  --disk disk-1 --device-name disk-1 --zone us-central1-c
```

```
$ gcloud compute instances attach-disk gke-openebs-cockroachdb-default-pool-fbceb18c-j9pl  --disk disk-2 --device-name disk-2 --zone us-central1-c
```

```
$ gcloud compute instances attach-disk gke-openebs-cockroachdb-default-pool-fbceb18c-kq41  --disk disk-3 --device-name disk-3 --zone us-central1-c
```

Verify the Block Device information

You can verify the attached Block Device information from Kubera portal under **Management > Block Devices** from the corresponding cluster page.



Block Device Management

195.417 GB
Total Capacity

4 /4
Unused Devices

0 /4
Used Devices

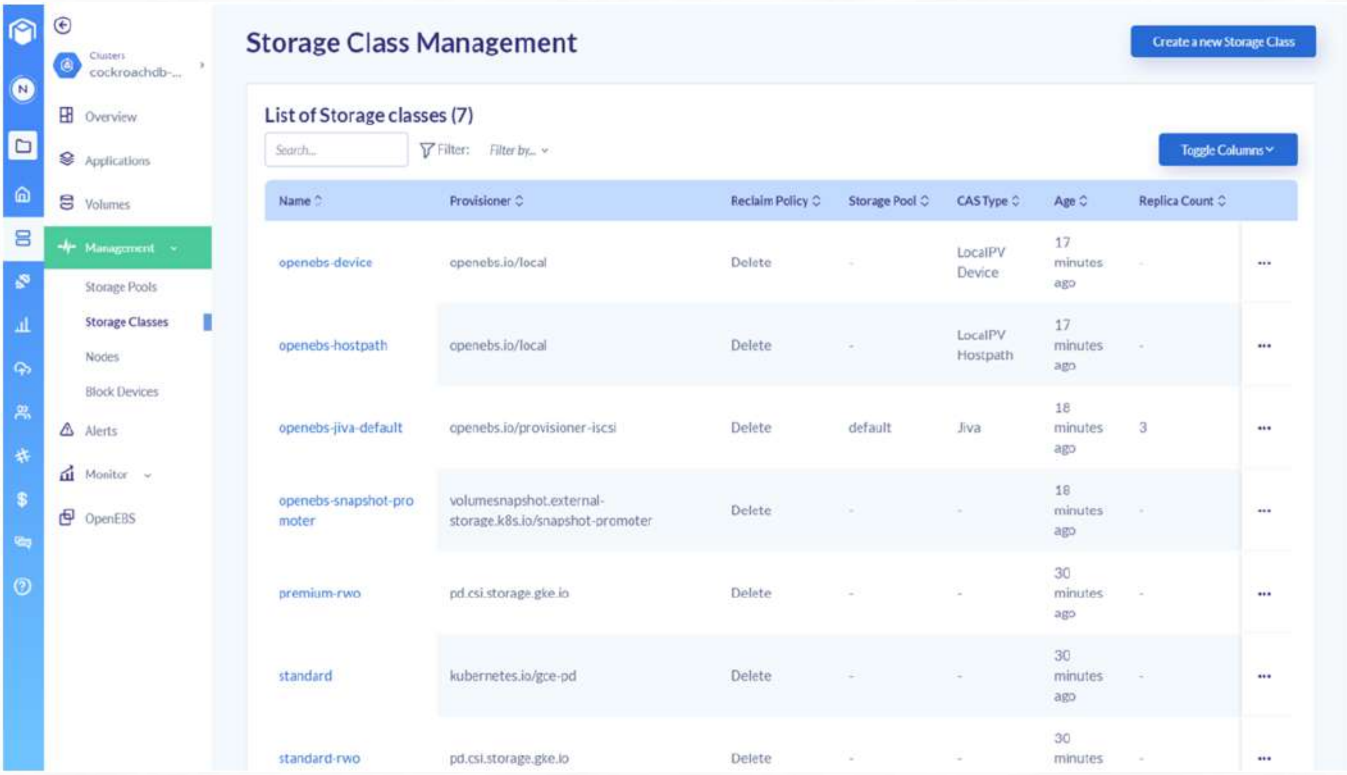
List of Devices

View by: Devices | Node Name | Search... | Toggle Columns

Device Name	Status	Claim Status	Device Path	Node Name	Total Capacity	Vendor	
blockdevice-172cc64f25d10e4a6771768cb6ea08ec	Active	Claimed	/dev/sdb1	gke-ranjith-cockroach-default-pool-21a6f577-zf55	64.423 GB	Google	•••
blockdevice-515d4c249b212a57f2f0e1a64455682d	Active	Unclaimed	/dev/sdc	gke-ranjith-cockroach-default-pool-21a6f577-brhs	2.147 GB	Google	•••
blockdevice-916598afba72f6f26f656ee5a8b4aa73	Active	Claimed	/dev/sdb1	gke-ranjith-cockroach-default-pool-21a6f577-b6gd	64.423 GB	Google	•••
blockdevice-def59a69fe12957d96b751db970083b7	Active	Claimed	/dev/sdb1	gke-ranjith-cockroach-default-pool-21a6f577-brhs	64.423 GB	Google	•••

Verify default Storage Class

You can verify the installed Storage Class information from Kubera portal under **Management > Storage Classes** from the corresponding cluster page.



Storage Class Management

List of Storage classes (7)

Name	Provisioner	Reclaim Policy	Storage Pool	CAS Type	Age	Replica Count	
openebs-device	openebs.io/local	Delete	-	LocalPV Device	17 minutes ago	-	...
openebs-hostpath	openebs.io/local	Delete	-	LocalPV Hostpath	17 minutes ago	-	...
openebs-jiva-default	openebs.io/provisioner-iscsi	Delete	default	Jiva	18 minutes ago	3	...
openebs-snapshot-promoter	volumesnapshot.external-storage.k8s.io/snapshot-promoter	Delete	-	-	18 minutes ago	-	...
premium-rwo	pd.csi.storage.gke.io	Delete	-	-	30 minutes ago	-	...
standard	kubernetes.io/gce-pd	Delete	-	-	30 minutes ago	-	...
standard-rwo	pd.csi.storage.gke.io	Delete	-	-	30 minutes	-	...

From the default StorageClasses, we will use openebs-device for providing persistent storage for running CockroachDB pods.

Installing CockroachDB Operator

In this section, we are installing the CockroachDB operator and then configuring CockroachDB cluster using OpenEBS LocalPV device as the storage engine.

RBAC policy configuration

CockroachDB requires cluster-admin privileges on GKE [4], hence we are going to configure the RBAC policies for the same

```
$ gcloud info | grep Account
  Account: [username@mayadata.io]
```

Create the cluster rolebinding

```
$ kubectl create clusterrolebinding $USER-cluster-admin-binding \
 --clusterrole=cluster-admin \
 --user=username@mayadata.io
clusterrolebinding.rbac.authorization.k8s.io/k8s-cluster-admin-binding created
```

Deploy CRD

We are going to use a Cockroachdb Operator. It is required to install the dependent CRDs to be deployed first.

```
$ kubectl apply -f https://raw.githubusercontent.com/cockroachdb/cockroach-
operator/master/config/crd/bases/crdb.cockroachlabs.com_crdbclusters.yaml
customresourcedefinition.apiextensions.k8s.io/crdbclusters.crdb.cockroachlabs
.com created
```

Deploy CockroachDB operator

Install CockroachDB operator using the following command.

```
$ kubectl apply -f
https://raw.githubusercontent.com/cockroachdb/cockroach-
operator/master/manifests/operator.yaml

clusterrole.rbac.authorization.k8s.io/cockroach-database-role created
serviceaccount/cockroach-database-sa created
clusterrolebinding.rbac.authorization.k8s.io/cockroach-database-rolebinding created
role.rbac.authorization.k8s.io/cockroach-operator-role created
clusterrolebinding.rbac.authorization.k8s.io/cockroach-operator-rolebinding created
clusterrole.rbac.authorization.k8s.io/cockroach-operator-role created
serviceaccount/cockroach-operator-sa created
rolebinding.rbac.authorization.k8s.io/cockroach-operator-default created
deployment.apps/cockroach-operator created
```

Check Operator deployment pod status

```
$ kubectl get pods

NAME                                                    READY   STATUS    RESTARTS  AGE
cockroach-operator-599465988d-k6ffx   1/1      Running   0               48s
```

CockroachDB cluster configuration

Download the cluster configuration file and make the necessary changes as per your requirement.

```
$ curl -O https://raw.githubusercontent.com/cockroachdb/cockroach-
operator/master/examples/example.yaml
```

We will update the storage class to use **openebs-device**, as shown below. Please note that for the production environment, make necessary other changes as per your requirement.

Sample **example.yaml** changes

```
apiVersion: crdb.cockroachlabs.com/v1alpha1
kind: CrdbCluster
metadata:
  name: cockroachdb
spec:
  dataStore:
    pvc:
      spec:
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: "60Gi"
        volumeMode: Filesystem
        storageClassName: openebs-device
   tlsEnabled: true
   image:
    name: cockroachdb/cockroach:v20.2.5
  nodes: 3
```

Apply the cluster configuration file

```
$ kubectl apply -f example.yaml
```

Check cluster pod status

```
$ kubectl  get pod,pv,pvc,sc
NAME                                    READY   STATUS     RESTARTS   AGE
pod/cockroach-operator-599465988d-fkgv6   1/1     Running   0          5m20s
pod/cockroachdb-0                          1/1     Running   0          2m17s
pod/cockroachdb-1                          1/1     Running   0          110s
pod/cockroachdb-2                          1/1     Running   0          81s
```

```
NAME                                                    CAPACITY   ACCESS MODES   RECLAIM POLICY
STATUS   CLAIM                          STORAGECLASS     REASON   AGE
persistentvolume/pvc-6f0a99a2-504a-4ab7-b865-200f96bfc6cb   60Gi       RWO
Delete          Bound    default/datadir-cockroachdb-1   openebs-device            104s
persistentvolume/pvc-a71b5078-f56f-4e1f-9237-43cfd854195e   60Gi       RWO
Delete          Bound    default/datadir-cockroachdb-0   openebs-device            2m12s
persistentvolume/pvc-de6ec858-0106-4454-8190-66cd2a9b465f   60Gi       RWO
Delete          Bound    default/datadir-cockroachdb-2   openebs-device 76s
```

```
NAME                                    STATUS   VOLUME                                  CAPACITY
ACCESS MODES   STORAGECLASS     AGE
persistentvolumeclaim/datadir-cockroachdb-0   Bound    pvc-a71b5078-f56f-4e1f-9237-
43cfd854195e   60Gi       RWO            openebs-device   2m19s
persistentvolumeclaim/datadir-cockroachdb-1   Bound    pvc-6f0a99a2-504a-4ab7-
b865-200f96bfc6cb   60Gi       RWO            openebs-device   111s
persistentvolumeclaim/datadir-cockroachdb-2   Bound    pvc-de6ec858-0106-4454-
8190-66cd2a9b465f   60Gi       RWO            openebs-device   82s
```

```
NAME                                        PROVISIONER
RECLAIMPOLICY   VOLUMEBINDINGMODE     ALLOWVOLUMEEXPANSION   AGE
storageclass.storage.k8s.io/openebs-device            openebs.io/local
Delete          WaitForFirstConsumer   false                 25m
storageclass.storage.k8s.io/openebs-hostpath           openebs.io/local
Delete          WaitForFirstConsumer   false                 25m
storageclass.storage.k8s.io/openebs-jiva-default       openebs.io/provisioner-iscsi
Delete          Immediate              false                 25m
storageclass.storage.k8s.io/openebs-snapshot-promoter   volumesnapshot.external-
storage.k8s.io/snapshot-promoter   Delete         Immediate            false              25m
storageclass.storage.k8s.io/premium-rwo               pd.csi.storage.gke.io
Delete          WaitForFirstConsumer   true                  38m
storageclass.storage.k8s.io/standard (default)         kubernetes.io/gce-pd
Delete          Immediate              true                  38m
storageclass.storage.k8s.io/standard-rwo              pd.csi.storage.gke.io
Delete          WaitForFirstConsumer   true                  38m
```

Accessing CockroachDB

After the pod status reaches running state, we can start using the database cluster. We will be using the built in sql-client for accessing and running some sql queries.

Enter into one of the cockroadb pod by using exec command

```
$ kubectl exec -it cockroachdb-2 -- ./cockroach sql --certs-dir cockroach-certs
#
# Welcome to the CockroachDB SQL shell.
# All statements must be terminated by a semicolon.
# To exit, type: \q.
#
# Server version: CockroachDB CCL v20.2.5 (x86_64-unknown-linux-gnu, built
2021/02/16 12:52:58, go1.13.14) (same version as client)
# Cluster ID: e51bfde5-2e75-4991-844e-d769f4b9b684
#
# Enter \? for a brief introduction.
#
root@:26257/defaultdb>
```

Run some basic SQL queries

```
root@:26257/defaultdb> CREATE DATABASE bank;
root@:26257/defaultdb> CREATE TABLE bank.accounts (id INT PRIMARY KEY, balance
DECIMAL);
root@:26257/defaultdb> INSERT INTO bank.accounts VALUES (1, 1000.50);
root@:26257/defaultdb> SELECT * FROM bank.accounts;
  id | balance
-----+----------
   1 | 1000.50
(1 row)
```

Create database user with password for accessing the database using web UI

```
root@:26257/defaultdb> CREATE USER roach WITH PASSWORD 'Q7gc8rEdS';
root@:26257/defaultdb> GRANT admin TO roach;
```

We are also going to create one more database, which we will use later for running benchmark load

```
root@:26257/defaultdb> CREATE DATABASE sbtest;
root@:26257/defaultdb> \q
```

In order to access the database , check the services.

```
$ kubectl get svc

NAME               TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)              AGE
cockroachdb          ClusterIP   None          <none>        26257/TCP,8080/TCP   5m3s
cockroachdb-public   ClusterIP   10.68.5.179   <none>        26257/TCP,8080/TCP   5m3s
kubernetes           ClusterIP   10.68.0.1     <none>        443/TCP              41m
```

For the demonstration purpose, we will be using NodePort for accessing the service.

In production environment either use loadbalancer or ingress services as per your requirement

Create a new node port service using the following.

```
$ cat cockroachdb-public-node-port.yaml
apiVersion: v1
kind: Service
metadata:
  name: cockroachdb-public-nodeport
  namespace: default
spec:
  ports:
  - name: grpc
    port: 26257
    protocol: TCP
    targetPort: 26257
  - name: http
    port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
    app.kubernetes.io/component: database
    app.kubernetes.io/instance: cockroachdb
    app.kubernetes.io/name: cockroachdb
  sessionAffinity: None
  type: NodePort
```

Apply the nodeport service

```
$ kubectl apply -f cockroachdb-public-node-port.yaml
```

Get the services status

```
$ kubectl get svc
NAME                        TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)                         AGE
cockroachdb                   ClusterIP   None          <none>        26257/TCP,8080/TCP
6m57s
cockroachdb-public            ClusterIP   10.68.5.179   <none>        26257/TCP,8080/TCP
6m57s
cockroachdb-public-nodeport   NodePort    10.68.4.195   <none>
26257:30324/TCP,8080:31937/TCP   5s
kubernetes                    ClusterIP   10.68.0.1     <none>        443/TCP                         43m
```

Verify that the cockroachDB Dashboard is accessible using web interface

```
https://<any_node_external-ip>:<NodePort>
```

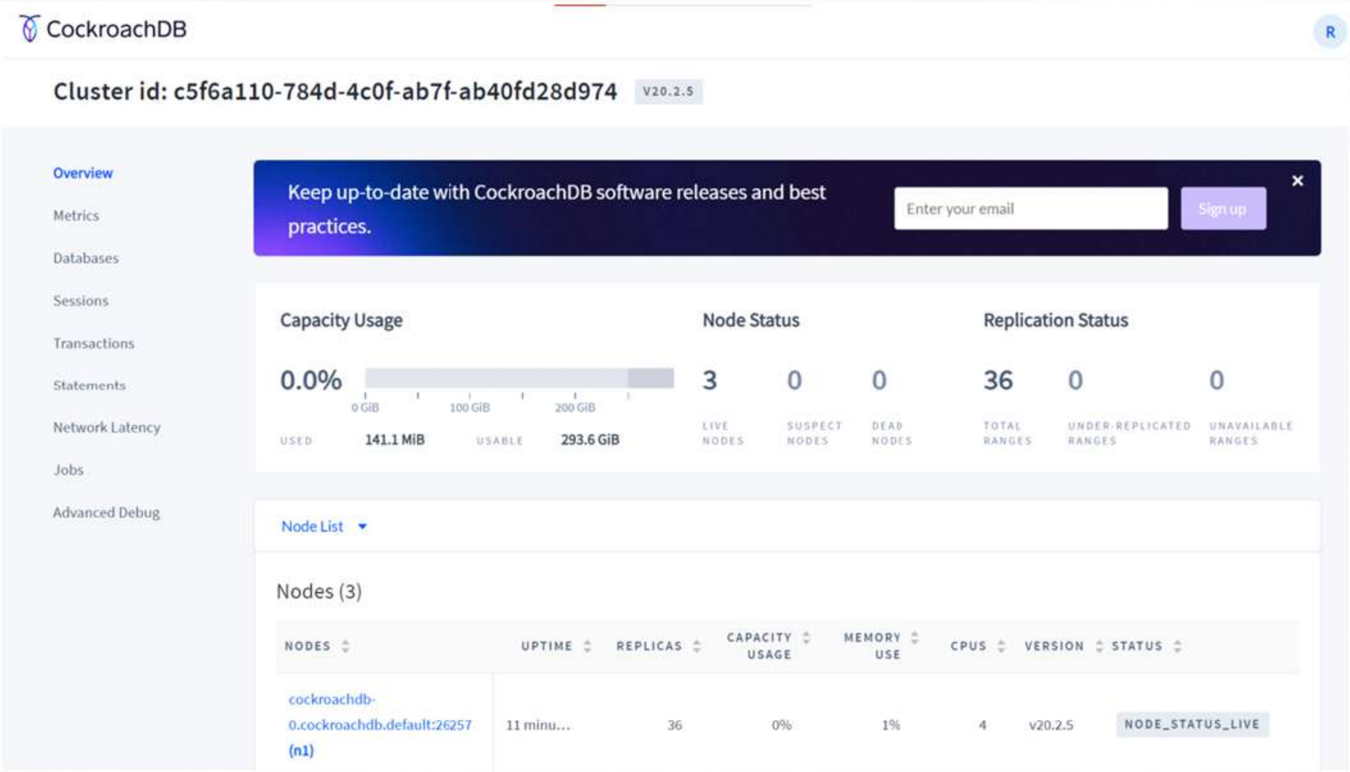
Example:

```
https://35.224.42.110:31937
```

Login credentials for the web UI

Username: roach

Password: Q7gc8rEdS



Cluster id: c5f6a110-784d-4c0f-ab7f-ab40fd28d974 V20.2.5

Keep up-to-date with CockroachDB software releases and best practices.

Capacity Usage

0.0% — 0 GiB, 100 GiB, 200 GiB

USED 141.1 MiB USABLE 293.6 GiB

Node Status

3 LIVE NODES

0 SUSPECT NODES

0 DEAD NODES

Replication Status

36 TOTAL RANGES

0 UNDER-REPLICATED RANGES

0 UNAVAILABLE RANGES

Nodes (3)

NODES	UPTIME	REPLICAS	CAPACITY USAGE	MEMORY USE	CPUS	VERSION	STATUS
cockroachdb-0.cockroachdb.default:26257 (n1)	11 minu...	36	0%	1%	4	v20.2.5	NODE_STATUS_LIVE

Monitoring CockroachDB

CockroachDB generates detailed time series metrics of each cluster node. Prometheus can be used for scrapping these metrics and grafana can be used for plotting graphs for the same.

Set up Prometheus and Grafana

In this section, we will install Prometheus Operator and use cockroachdb Service Monitor.We will install the community edition of Prometheus operator using Helm.This will install both Prometheus and Grafana.

Download Prometheus operator using Helm v3.

```
$ helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
$ helm repo update
$ kubectl create namespace monitoring
```

The following command will install both Prometheus and Grafana components.

```
$ helm install prometheus prometheus-community/kube-prometheus-stack
--namespace monitoring
```

Note: Check compatibility for your Kubernetes version and Prometheus stack from here.

Verify if Prometheus related pods are installed successfully:

```
$ kubectl get pods -n monitoring
NAME                                                  READY   STATUS    RESTARTS   AGE
alertmanager-prometheus-kube-prometheus-alertmanager-0   2/2
Running   0          54s
prometheus-grafana-6f5448f95b-qqsvc                      2/2     Running   0
59s
prometheus-kube-prometheus-operator-8556f58759-hpldk     1/1
Running   0          59s
prometheus-kube-state-metrics-6bfcd6f648-r89kk           1/1     Running
0          59s
prometheus-prometheus-kube-prometheus-prometheus-0       2/2
Running   1          53s
prometheus-prometheus-node-exporter-766l9                1/1     Running   0
59s
prometheus-prometheus-node-exporter-8q6pm                1/1     Running
0          60s
prometheus-prometheus-node-exporter-lst6v                1/1     Running   0
60s
```

Verify if Prometheus related services are installed successfully:

```
$ kubectl get svc -n monitoring

NAME                              TYPE       CLUSTER-IP    EXTERNAL-IP   PORT(S)
AGE
alertmanager-operated                   ClusterIP   None         <none>
9093/TCP,9094/TCP,9094/UDP   97s
prometheus-grafana                     ClusterIP   10.68.1.15     <none>       80/TCP
103s
prometheus-kube-prometheus-alertmanager   ClusterIP   10.68.11.16    <none>
9093/TCP                  103s
prometheus-kube-prometheus-operator       ClusterIP   10.68.10.115   <none>
443/TCP                   103s
prometheus-kube-prometheus-prometheus     ClusterIP   10.68.1.120    <none>
9090/TCP                  103s
prometheus-kube-state-metrics           ClusterIP   10.68.3.147    <none>       8080/TCP
103s
prometheus-operated                    ClusterIP   None         <none>       9090/TCP
97s
prometheus-prometheus-node-exporter       ClusterIP   10.68.6.139    <none>
9100/TCP                  103s
```

Change `prometheus-prometheus-oper-prometheus` service to LoadBalancer/NodePort from ClusterIP. This change is for accessing Prometheus service from your Web browser.

```
$ kubectl patch svc prometheus-kube-prometheus-prometheus -n monitoring -p '{"spec":
{"type": "NodePort"}}'
```

Change `prometheus-grafana` service to LoadBalancer/NodePort from ClusterIP. This change is for accessing Grafana service from your Web browser.

```
$ kubectl patch svc prometheus-grafana -n monitoring -p '{"spec":
 {"type": "NodePort"}}'
```

Note: If the user needs to access Prometheus and Grafana outside the network, the service type can be changed or a new service should be added to use LoadBalancer or create Ingress resources for production deployment.

For ease of simplicity in testing the deployment, we are going to use NodePort. Please be advised to consider using LoadBalancer or Ingress, instead of NodePort, for production deployment.

```
$ kubectl get svc -n monitoring

NAME                               TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)
AGE
alertmanager-operated                    ClusterIP   None          <none>
9093/TCP,9094/TCP,9094/UDP   11m
prometheus-grafana                       NodePort    10.68.1.15     <none>       80:31360/TCP
11m
prometheus-kube-prometheus-alertmanager   ClusterIP   10.68.11.16    <none>
9093/TCP                    11m
prometheus-kube-prometheus-operator      ClusterIP   10.68.10.115   <none>
443/TCP                     11m
prometheus-kube-prometheus-prometheus     NodePort    10.68.1.120    <none>
9090:32515/TCP              11m
prometheus-kube-state-metrics            ClusterIP   10.68.3.147    <none>       8080/TCP
11m
prometheus-operated                      ClusterIP   None          <none>       9090/TCP
11m
prometheus-prometheus-node-exporter      ClusterIP   10.68.6.139    <none>
9100/TCP                    11m
```

Installing Cockroachdb Service Monitor

We will label CockroachDB service, so that only cockroachdb (and not cockroachdb-public or cockroachdb-public-nodeport) service is monitored by the Prometheus.

```
$ kubectl label svc cockroachdb prometheus=cockroachdb
```

Deploy the following cockroachdb service monitor

```
$ cat cockroachdb-prometheus-sm.yaml
# Select any services with the prometheus:cockroachdb label
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: cockroachdb
  labels:
    app: cockroachdb
    prometheus: cockroachdb
    release: prometheus
spec:
  selector:
    matchLabels:
      prometheus: cockroachdb
  namespaceSelector:
    matchNames:
    - default
  endpoints:
  - port: http
    path: /_status/vars
    tlsConfig:
      # The HTTPS certs are signed by the kubernetes internal
      # certificate authority.
      caFile: "/var/run/secrets/kubernetes.io/serviceaccount/ca.crt"
      insecureSkipVerify: true
      # This overrides the hostname verification check for the admin
      # UI port to match our quickstart secure-mode cluster setup.
      serverName: "127.0.0.1"
```

Please note that CockroachDB pod internally uses a self signed certificate with CA Cockroach CA and prometheus uses cert generated by kubernetes. For this deployment guide, we have added

```
insecureSkipVerify: true .
```

Please consider using the appropriate CA certs for production environments.

Apply the service monitor for CockroachDB.

```
$ kubectl apply -f cockroachdb-prometheus-sm.yaml

servicemonitor.monitoring.coreos.com/cockroachdb created
```

Verify if the service monitor for CockroachDB is created successfully.

```
$ kubectl get servicemonitor
NAME          AGE
cockroachdb   55s
```

Launch Grafana using External IP of prometheus-grafana with port 80 on your browser, similar to the format here http://:<80>. This is applicable if the service is being created using Load Balancer. If it is NodePort, then use :

```
<External IP of Node>:< Nodeport of prometheus-grafana>.
```

Example:

```
http://35.224.42.110:31360/
```

Username: admin **Password:** prom-operator

Password can be obtained by using the command

```
(kubectl get secret \
    --namespace monitoring prometheus-grafana \
    -o jsonpath="{.data.admin-password}" \
    | base64 --decode ; echo
)
```

Add the following dashboards to Grafana for various metrics such as Run time info, storage level info, SQL info, Replica info, etc, by uploading them using the 'Upload JSON option and selecting the prometheus **as datasource.**

1. **Runtime dashboard: node status, including uptime, memory, and cpu.**

```
https://raw.githubusercontent.com/cockroachdb/cockroach/master/monitoring/grafana-dashboards/runtime.json
```

2. **Storage dashboard: storage availability.**

```
https://raw.githubusercontent.com/cockroachdb/cockroach/master/monitoring/grafana-dashboards/storage.json
```

3. **SQL dashboard: sql queries/transactions.**

```
https://raw.githubusercontent.com/cockroachdb/cockroach/master/monitoring/grafana-dashboards/sql.json
```

4. **Replicas dashboard: replica information and operations.**

```
https://raw.githubusercontent.com/cockroachdb/cockroach/master/monitoring/grafana-dashboards/replicas.json
```

Benchmarking

Let's create a SysBench pod for the performance benchmark of the CockroachDB database.

```
$ kubectl run -it --rm sysbench-client --image=perconalab/sysbench:latest --restart=Never -- bash
If you don't see a command prompt, try pressing Enter.

root@sysbench-client:/sysbench#
```

The above command will create a temporary pod for SysBench. This pod will be used to run the benchmark commands. In this example, we are using the cockroachdb-public **service name as the cockroachdb host in the test command.**

Prepare the data

Ensure that the database has already been created before running the tests. In this example, we have created a database called "sbtest" in the previous section and used it in the performance benchmark tests. Please remember to use the corresponding CockroachDB password throughout the performance benchmark tests.

The root password used in the following command can be obtained from the previous section.

Run the following tests from the SysBench pod.

```
root@sysbench-client:/sysbench# pass=Q7gc8rEdS
root@sysbench-client:/sysbench# # cocroachdb init
root@sysbench-client:/sysbench# sysbench --db-driver=pgsql  --tables=10 --
table_size=1000000  --pgsql-host=cockroachdb-public --pgsql-port=26257 --pgsql-
user=roach --pgsql-password=$pass  --time=0 --events=10000000 --report-interval=1   --
threads=128 oltp_write_only prepare
```

Sample output:

```
sysbench 1.0.20 (using bundled LuaJIT 2.1.0-beta2)

Initializing worker threads...

Creating table 'sbtest4'...
Creating table 'sbtest1'...
Creating table 'sbtest2'...
Creating table 'sbtest5'...
Creating table 'sbtest6'...
Inserting 1000000 records into 'sbtest1'
Inserting 1000000 records into 'sbtest4'
Inserting 1000000 records into 'sbtest5'
Creating table 'sbtest8'...
Inserting 1000000 records into 'sbtest6'
Creating table 'sbtest9'...
Creating table 'sbtest10'...
Creating table 'sbtest3'...
Creating table 'sbtest7'...
```

continued to the next page..

```
Inserting 1000000 records into 'sbtest9'
Inserting 1000000 records into 'sbtest2'
Inserting 1000000 records into 'sbtest8'
Inserting 1000000 records into 'sbtest7'
Inserting 1000000 records into 'sbtest10'
Inserting 1000000 records into 'sbtest3'
Creating a secondary index on 'sbtest5'...
Creating a secondary index on 'sbtest1'...
Creating a secondary index on 'sbtest6'...
Creating a secondary index on 'sbtest4'...
Creating a secondary index on 'sbtest2'...
Creating a secondary index on 'sbtest9'...
Creating a secondary index on 'sbtest3'...
Creating a secondary index on 'sbtest8'...
Creating a secondary index on 'sbtest7'...
Creating a secondary index on 'sbtest10'...
```

In the following series of commands, we are going to generate Read Only, Write Only and Read/Write traffic using different concurrent client threads.

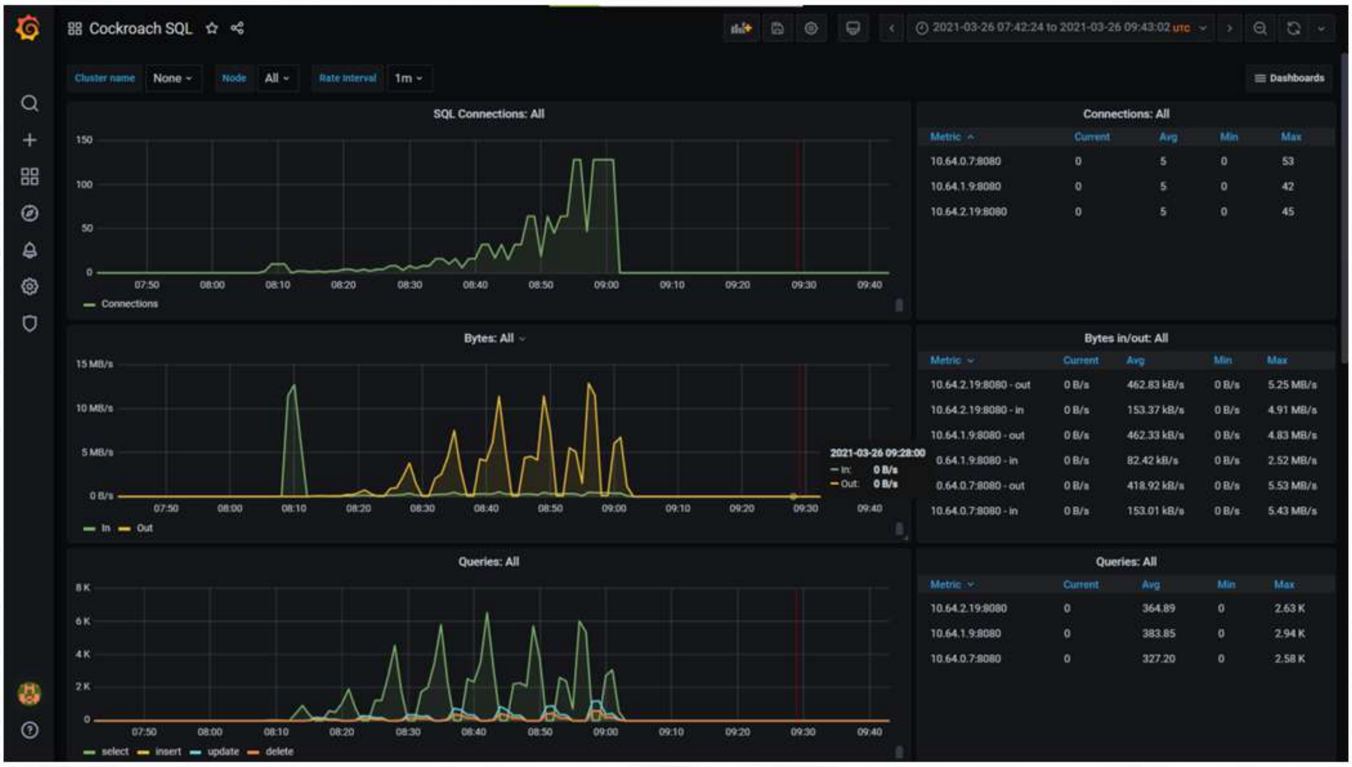
```
root@sysbench-client:/sysbench# pass=Q7gc8rEdS
root@sysbench-client:/sysbench# timeinterval=120
root@sysbench-client:/sysbench# cooloff=15
root@sysbench-client:/sysbench# logfile="cockroachdb-benchmark.txt"
root@sysbench-client:/sysbench# for i in 2 4 8 16 32 64 128
do
```

```
echo "Number of threads $i" >> $logfile
date >> $logfile
sysbench oltp_read_only  --db-driver=pgsql  --tables=10 --table_size=1000000  --pgsql-
host=cockroachdb-public --pgsql-port=26257 --pgsql-user=roach --pgsql-
password=$pass  --time=0 --events=10000000 --report-interval=1   --threads=$i --
time=$timeinterval run  >> $logfile
```

```
sleep $cooloff
sysbench oltp_write_only --db-driver=pgsql  --tables=10 --table_size=1000000  --pgsql-
host=cockroachdb-public --pgsql-port=26257 --pgsql-user=roach --pgsql-
password=$pass  --time=0 --events=10000000 --report-interval=1   --threads=$i --
time=$timeinterval run   >> $logfile
```

```
sleep $cooloff
sysbench oltp_read_write --db-driver=pgsql  --tables=10 --table_size=1000000  --pgsql-
host=cockroachdb-public --pgsql-port=26257 --pgsql-user=roach --pgsql-
password=$pass  --time=0 --events=10000000 --report-interval=1   --threads=$i --
time=$timeinterval run   >> $logfile
sleep 30
done
```

Following is Grafana screenshots after the benchmark runs



Cluster name: None · Node: All · Rate interval: 1m

Connections: All

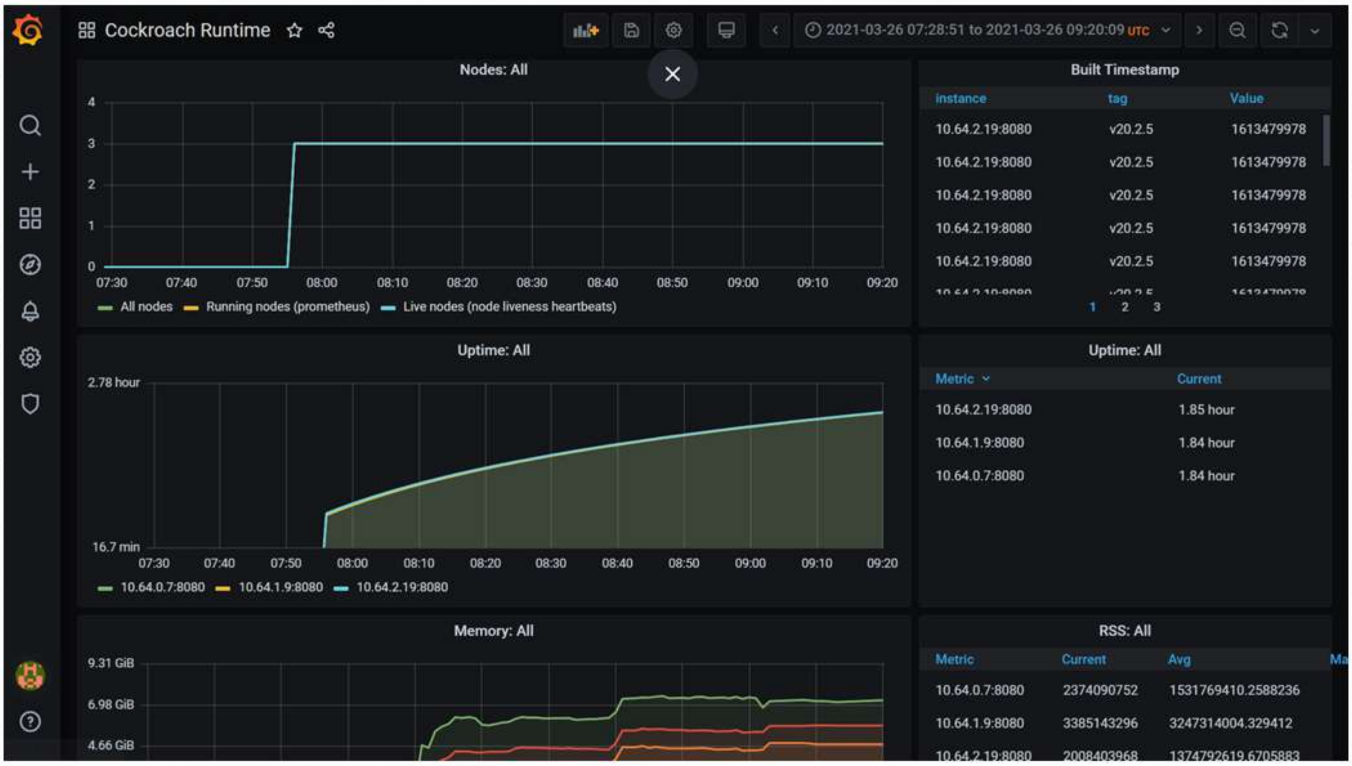
Metric	Current	Avg	Min	Max
10.64.0.7:8080	0	5	0	53
10.64.1.9:8080	0	5	0	42
10.64.2.19:8080	0	5	0	45

Bytes in/out: All

Metric	Current	Avg	Min	Max
10.64.2.19:8080 - out	0 B/s	462.83 kB/s	0 B/s	5.25 MB/s
10.64.2.19:8080 - in	0 B/s	153.37 kB/s	0 B/s	4.91 MB/s
10.64.1.9:8080 - out	0 B/s	462.33 kB/s	0 B/s	4.83 MB/s
0.64.1.9:8080 - in	0 B/s	82.42 kB/s	0 B/s	2.52 MB/s
0.64.0.7:8080 - out	0 B/s	418.92 kB/s	0 B/s	5.53 MB/s
10.64.0.7:8080 - in	0 B/s	153.01 kB/s	0 B/s	5.43 MB/s

Queries: All

Metric	Current	Avg	Min	Max
10.64.2.19:8080	0	364.89	0	2.63 K
10.64.1.9:8080	0	383.85	0	2.94 K
10.64.0.7:8080	0	327.20	0	2.58 K



Cockroach Runtime

2021-03-26 07:28:51 to 2021-03-26 09:20:09 UTC

Built Timestamp

instance	tag	Value
10.64.2.19:8080	v20.2.5	1613479978
10.64.2.19:8080	v20.2.5	1613479978
10.64.2.19:8080	v20.2.5	1613479978
10.64.2.19:8080	v20.2.5	1613479978
10.64.2.19:8080	v20.2.5	1613479978
10.64.2.19:8080	v20.2.5	1613479978

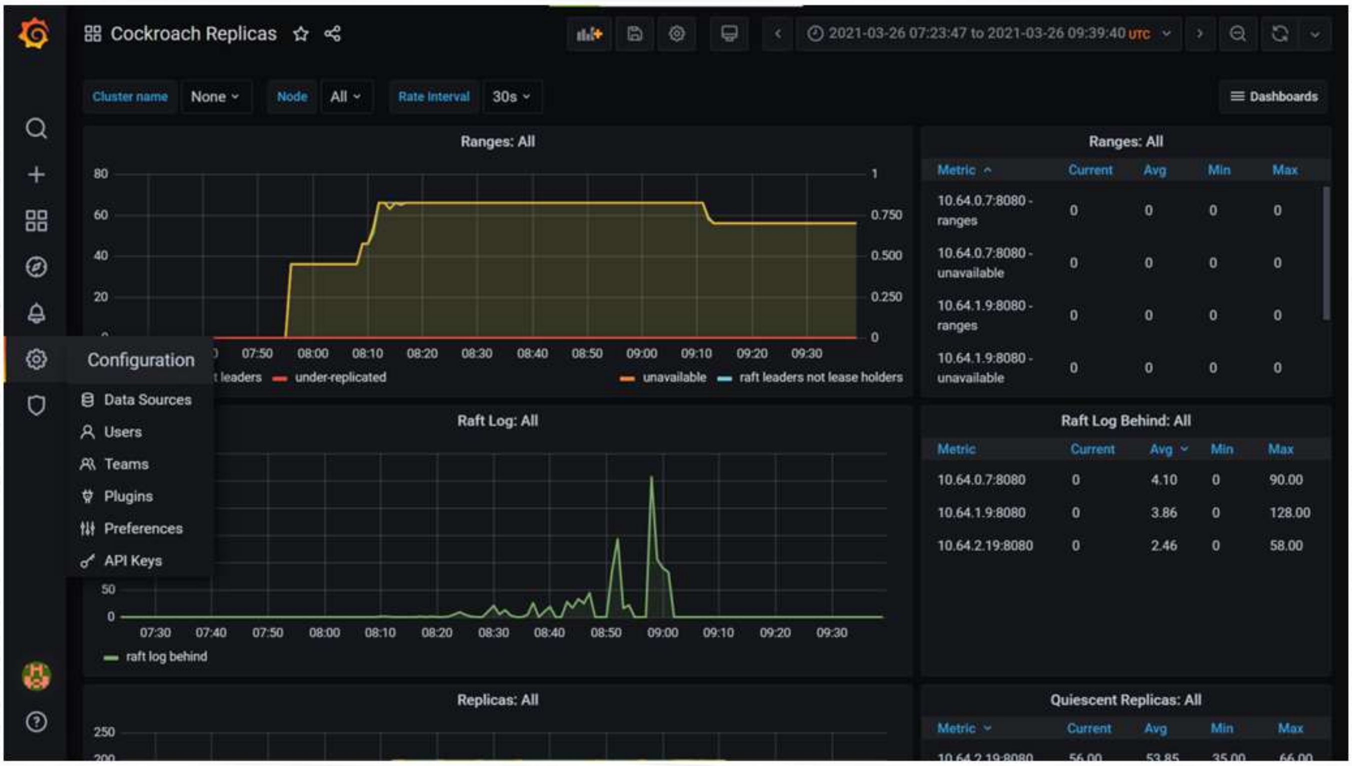
1 2 3

Uptime: All

Metric	Current
10.64.2.19:8080	1.85 hour
10.64.1.9:8080	1.84 hour
10.64.0.7:8080	1.84 hour

RSS: All

Metric	Current	Avg	Ma
10.64.0.7:8080	2374090752	1531769410.2588236	
10.64.1.9:8080	3385143296	3247314004.329412	
10.64.2.19:8080	2008403968	1374792619.6705883	



Cockroach Replicas

2021-03-26 07:23:47 to 2021-03-26 09:39:40 UTC

Cluster name: None
Node: All
Rate interval: 30s

Ranges: All

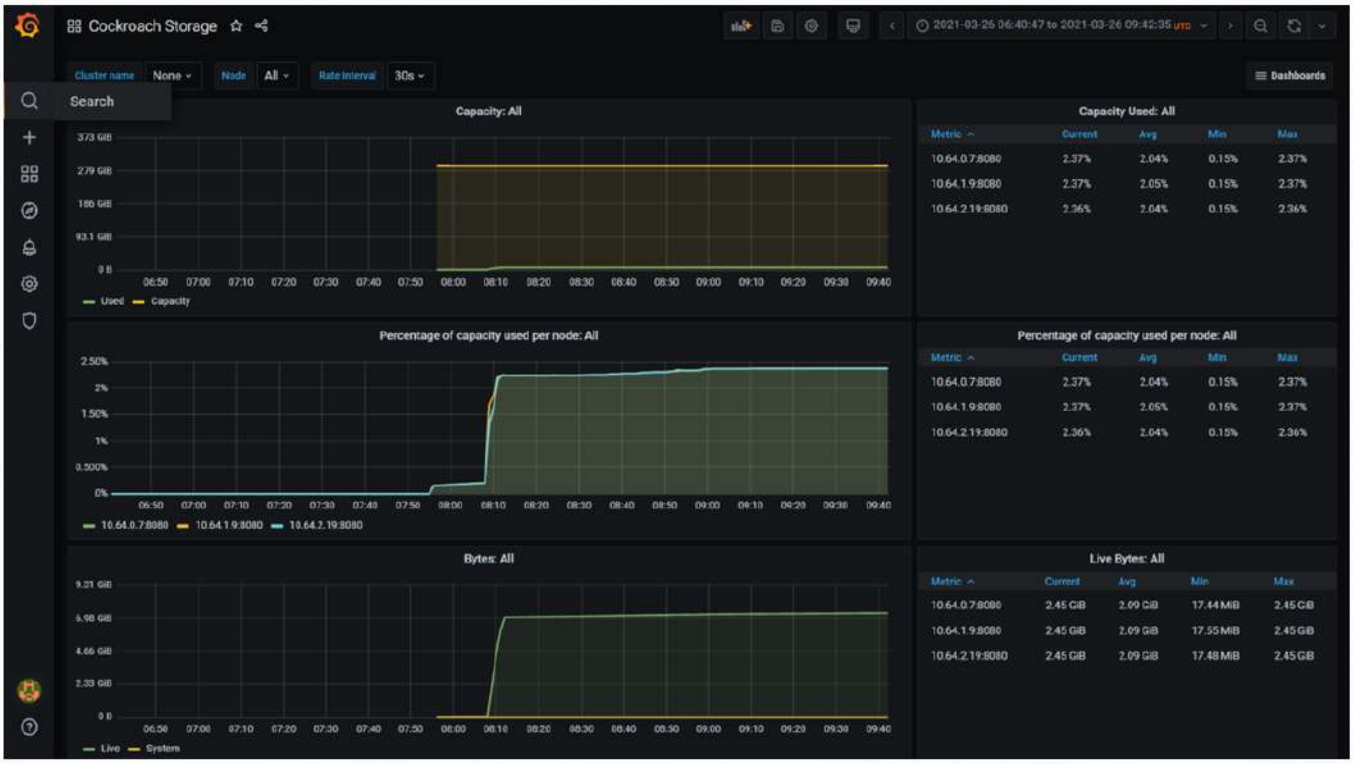
Metric	Current	Avg	Min	Max
10.64.0.7:8080 - ranges	0	0	0	0
10.64.0.7:8080 - unavailable	0	0	0	0
10.64.1.9:8080 - ranges	0	0	0	0
10.64.1.9:8080 - unavailable	0	0	0	0

Raft Log Behind: All

Metric	Current	Avg	Min	Max
10.64.0.7:8080	0	4.10	0	90.00
10.64.1.9:8080	0	3.86	0	128.00
10.64.2.19:8080	0	2.46	0	58.00

Quiescent Replicas: All

Metric	Current	Avg	Min	Max
10.64.2.19:8080	56.00	53.85	35.00	66.00



Capacity Used: All

Metric	Current	Avg	Min	Max
10.64.0.7:8080	2.37%	2.04%	0.15%	2.37%
10.64.1.9:8080	2.37%	2.05%	0.15%	2.37%
10.64.2.19:8080	2.36%	2.04%	0.15%	2.36%

Percentage of capacity used per node: All

Metric	Current	Avg	Min	Max
10.64.0.7:8080	2.37%	2.04%	0.15%	2.37%
10.64.1.9:8080	2.37%	2.05%	0.15%	2.37%
10.64.2.19:8080	2.36%	2.04%	0.15%	2.36%

Live Bytes: All

Metric	Current	Avg	Min	Max
10.64.0.7:8080	2.45 GiB	2.09 GiB	17.44 MiB	2.45 GiB
10.64.1.9:8080	2.45 GiB	2.09 GiB	17.55 MiB	2.45 GiB
10.64.2.19:8080	2.45 GiB	2.09 GiB	17.48 MiB	2.45 GiB

Conclusion

As described at the beginning of this guide, CockroachDB is a distributed SQL database built on a transactional and strongly-consistent key-value store. Since it is a stateful application, in this guide we have used OpenEBS LocalPV to provide node local storage to the CockroachDB statefulset deployment. We used Kubera to deploy OpenEBS on the k8 cluster. We showed how to check metrics and monitoring of CockroachDB instances using Prometheus and Grafana.

www.mayadata.io

MayaData

linkedin.com/company/mayadata/

twitter.com/MayaData

blog.mayadata.io/