

How to Deploy CockroachDB on Kubernetes

Learn how to deploy CockroachDB on Kubernetes with our step-by-step guide. Discover best practices, configuration tips, and troubleshooting advice for optimal performance.

Contents

01	Introduction	3
02	Prerequisites	3
03	Install the CockroachDB Operator	3
04	Create a CockroachDB Cluster	5
05	Access the CockroachDB Cluster	7
06	Conclusion	12

Introduction

CockroachDB is a distributed SQL database system designed to support data-intensive applications in production environments by replicating data to multiple nodes within a cluster. In case a single cluster node fails, CockroachDB elects new nodes with high availability and enhanced scaling to balance the database load. In addition, the Cockroach DB automated repair feature detects data inconsistencies and automatically corrects any corrupted databases to keep the cluster applications running.

This guide explains how to deploy CockroachDB on a Vultr Kubernetes Engine (VKE) cluster to enable scalability, fault tolerance, and management using the cluster SQL client or graphical interface.

Prerequisites

Before you begin:

- Deploy a [Vultr Kubernetes Engine \(VKE\) cluster](#) with at least `3` nodes, `8 GB` RAM and `4 VcPUs` in a supported Vultr Block Storage location.
- Deploy a Vultr [Ubuntu instance](#) to use as a management workstation.
- Access the workstation as a [non-root user with sudo privileges](#).
- Install and configure [Kubectl](#) to [access the VKE cluster](#).

Install the CockroachDB Operator

A CockroachDB Kubernetes operator manages the deployment process with the automation of tasks such as scaling and maintenance of Cockroach DB instances on each node within the cluster. Follow the steps below to install the CockroachDB operator and necessary dependencies in a new `cockroach-operator-system` namespace to run a CockroachDB cluster.

1. Install the latest Cockroach DB Operator Custom Resource Definitions (CRDs).

CONSOLE

```
$ kubectl apply -f https://raw.githubusercontent.com/cockroachdb/cockroach-operator/v2.12.0/install/crds.yaml
```

The above command installs the CockroachDB Operator CRDs version `v2.12.0`. Visit the [operator tags page to verify the latest version](#).

2. Install the CockroachDB Operator.

CONSOLE

```
$ kubectl apply -f https://raw.githubusercontent.com/cockroachdb/cockroach-operator/v2.12.0/install/operator.yaml
```

Output:

```
namespace/cockroach-operator-system created
serviceaccount/cockroach-operator-sa created
clusterrole.rbac.authorization.k8s.io/cockroach-operator-role created
clusterrolebinding.rbac.authorization.k8s.io/cockroach-operator-rolebinding
created
service/cockroach-operator-webhook-service created
deployment.apps/cockroach-operator-manager created
mutatingwebhookconfiguration.admissionregistration.k8s.io/cockroach-operator-
mutating-webhook-configuration created
validatingwebhookconfiguration.admissionregistration.k8s.io/cockroach-operator-
validating-webhook-configuration created
```

3. Wait for at least `2` minutes for the deployment process to complete. Then, view all pods in the CockroachDB Operator namespace `cockroach-operator-system`.

CONSOLE

```
$ kubectl get pods -n cockroach-operator-system
```

Verify that all pods are ready and running similar to the output below:

NAME	READY	STATUS	RESTARTS	AGE
cockroach-operator-manager-6fd5c68d58-drqf6	1/1	Running	0	5m11s

Create a CockroachDB Cluster

To create a CockroachDB cluster, specify the necessary storage volumes, CPU count, and memory limits to use with the database nodes. In a production environment, define a PVC resource to use Vultr Block Storage volumes with at least **60 GB** storage, **2** vCPUs, and **8 GB** memory. Follow the steps below to deploy a CockroachDB cluster with the minimum production values.

1. Create a new directory `cockroachdb-cluster` to store the cluster manifest files.

CONSOLE

```
$ mkdir cockroachdb-cluster
```

2. Change to the directory.

CONSOLE

```
$ cd cockroachdb-cluster
```

3. Create a new CockroachDB cluster resource file `cluster.yaml` to define the operator functions.

CONSOLE

```
$ nano cluster.yaml
```

4. Add the following contents to the file.

YAML

```
apiVersion: crdb.cockroachlabs.com/v1alpha1
kind: CrdbCluster
```

```
metadata:
  name: cockroachdb
spec:
  dataStore:
    pvc:
      spec:
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: "60Gi"
        volumeMode: Filesystem
  resources:
    requests:
      cpu: 2
      memory: 4Gi
    limits:
      cpu: 2
      memory: 8Gi
  tlsEnabled: true
  image:
    name: cockroachdb/cockroach:v23.1.11
  nodes: 3
  additionalLabels:
    crdb: vultr-cockroachdb
```

Save and close the file.

The above cluster configuration creates a CockroachDB cluster with **60 GB** storage per Vultr Block Storage volume, and a limit of **2 vCPUs**, **8 GiB** memory for each database node within the cluster.

5. Apply the configuration to create the CockroachDB cluster.

CONSOLE

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

Output:

```
crdbcluster.crdb.cockroachlabs.com/cockroachdb created
```

6. Wait at least **3** minutes for the CockroachDB cluster and Vultr Block Storage volumes creation process to complete. Then, view the CockroachDB cluster resource to verify its run-time status.

CONSOLE

```
$ kubectl get crdbcluster
```

Output:

NAME	AGE
cockroachdb	4m21s

7. View the cluster Pods to verify that all CockroachDB resources are ready and running.

CONSOLE

```
$ kubectl get pods
```

Your output should look like the one below:

NAME	READY	STATUS	RESTARTS	AGE
cockroachdb-0	1/1	Running	0	5m35s
cockroachdb-1	1/1	Running	0	5m35s
cockroachdb-2	1/1	Running	0	5m35s

Access the CockroachDB Cluster

CockroachDB is accessible within the cluster using the internal `cockroachdb-public` service address. To externally access the cluster, create a new internal `cockroach` binary pod to use the SQL client, and set up a new load balancer resource to access the graphical database console as described in the steps below.

1. List all cluster services to verify the available CockroachDB resources.

CONSOLE

```
$ kubectl get svc
```

Your output should look like the one below:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
cockroach-operator-webhook-service	ClusterIP	10.96.106.14	<none>
cockroachdb	ClusterIP	None	<none>
cockroachdb-public	ClusterIP	10.97.143.136	<none>

As displayed in the output above, pods within the same namespace communicate to CockroachDB resources using the `cockroachdb-public` service internal cluster IP address.

2. Create a new `cockroachdb` binary pod to access the CockroachDB SQL client.

CONSOLE

```
$ kubectl create -f https://raw.githubusercontent.com/cockroachdb/cockroach-operator/v2.12.0/examples/client-secure-operator.yaml
```

3. View the cluster pods to verify that the `cockroachdb-client-secure` SQL client is ready and running.

CONSOLE

```
$ kubectl get pods
```

Output:

NAME	READY	STATUS	RESTARTS	AGE
cockroachdb-0	1/1	Running	0	8m29s
cockroachdb-1	1/1	Running	0	8m29s

```
cockroachdb-2          1/1    Running  0          8m29s
cockroachdb-client-secure 1/1    Running  0          33s
```

4. Access the pod shell using the `cockroachdb-public` service as the host gateway to access CockroachDB.

CONSOLE

```
$ kubectl exec -it cockroachdb-client-secure -- ./cockroach
sql --certs-dir=/cockroach/cockroach-certs --
host=cockroachdb-public
```

Output:

```
#
# Welcome to the CockroachDB SQL shell.
# All statements must be terminated by a semicolon.
# To exit, type: \q.
#
# Server version: CockroachDB CCL v23.1.11 (x86_64-pc-linux-gnu, built 2023/09/27
01:53:43, go1.19.10) (same version as client)
# Cluster ID: ce9eaa9d-1689-4116-b5e0-10ca71874712
#
# Enter \? for a brief introduction.
#
root@cockroachdb-public:26257/defaultdb>
```

5. Create a sample CockroachDB database.

SQL

```
> CREATE DATABASE exampledb;
```

Output:

```
CREATE DATABASE

Time: 41ms total (execution 40ms / network 1ms)
```

6. Create a new CockroachDB database user with a strong password.

```
SQL
```

```
> CREATE USER example_admin WITH PASSWORD 'strong-pass';
```

Output:

```
CREATE ROLE
```

```
Time: 312ms total (execution 311ms / network 1ms)
```

7. Grant administrative database privileges to the user.

```
SQL
```

```
> GRANT admin TO example_admin;
```

8. Exit the pod shell.

```
SQL
```

```
> \q
```

9. Create a new load balancer resource using the `cockroachdb-public` service to expose the CockroachDB graphical console with an external IP Address.

```
CONSOLE
```

```
$ kubectl expose service cockroachdb-public --  
type=LoadBalancer --name=cockroachdb-external
```

10. Wait for at least 3 minutes to attach a Vultr LoadBalancer to the service. Then, view the cluster services to verify the assigned public IP Address.

```
CONSOLE
```

```
$ kubectl get svc
```

Your output should look like the one below:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
cockroachdb	ClusterIP	None	<none>
PORT(S)		AGE	
8080/TCP,26257/TCP		37m	26258/TCP,
cockroachdb-external	LoadBalancer	10.103.172.147	192.168.0.34
26258:31007/TCP,8080:32289/TCP,26257:30791/TCP		12m	
cockroachdb-public	ClusterIP	10.97.75.3	<none>
8080/TCP,26257/TCP		37m	26258/TCP,
kubernetes	ClusterIP	10.96.0.1	<none>
TCP		55m	443/

11. Access the CockroachDB port `8080` using your external service IP Address in a new web browser session. Replace `192.168.0.34` with your actual load balancer IP address.

```
https://192.168.0.34:8080
```

CockroachDB

Log in to the DB Console

Username

Password

[Log in](#)

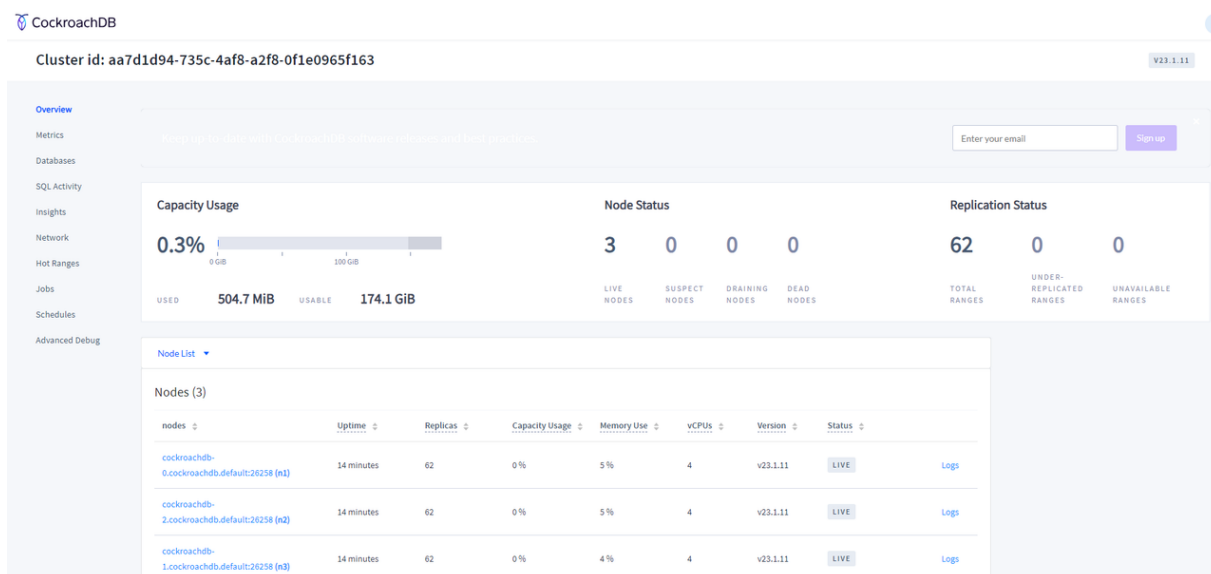
A user with a password is required to log in to the DB Console on secure clusters.

Create a user with this SQL command:

```
CREATE USER craig WITH PASSWORD 'cockroach';
```

[Read more about configuring login](#)

12. Log in to the database console using the `example_admin` user and password you created earlier in the SQL client.
13. Verify the CockroachDB cluster status, nodes, and databases in the graphical management console to integrate other cluster applications.



Conclusion

You have installed CockroachDB on a Vultr Kubernetes Engine (VKE) cluster and accessed the database using a graphical management console. Depending on your cluster applications, you can integrate internal and external applications to the database with automatic failover capabilities to ensure high application availability in case a cluster node fails. For more information about CockroachDB, visit the [official documentation](#).



VULTR

