Deploy CockroachDB on IBM Cloud Kubernetes

This page shows you how to manually deploy an insecure single-node CockroachDB cluster on IBM Cloud Kubernetes platform.

If you are only testing CockroachDB, or you are not concerned with protecting network communication with TLS encryption, working with insecure cluster will be fine.

> *TIP:*
> *To deploy a 30-day free CockroachCloud cluster instead of running CockroachDB yourself, see the Quickstart.*

Requirements

- You must have an active IBMCloud account.
- Enough credits to run a Kubernetes (k8s) cluster.
- Microsoft PowerShell (For Windows) or Bash Shell for Linux distribution.
- Carefully review the Production Checklist and recommended Topology Patterns.
- When deploying in a single availability zone:
    - To be able to tolerate the failure of any 1 node, use at least 3 nodes with the default 3-way replication factor. In this case, if 1 node fails, each range retains 2 of its 3 replicas, a majority.
    - To be able to tolerate 2 simultaneous node failures, use at least 5 nodes and increase the default replication factor for user data to 5. The

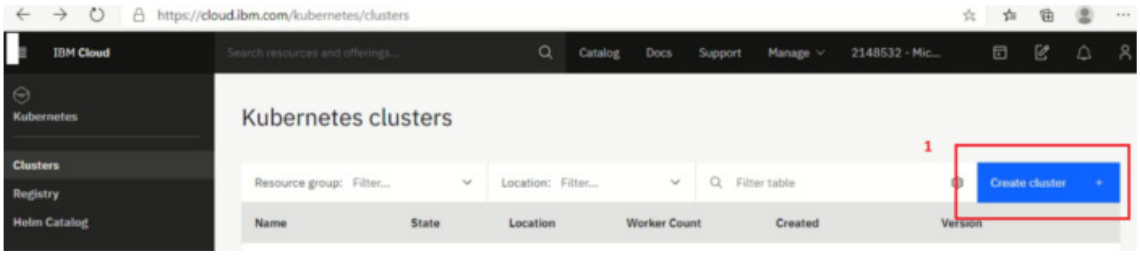
replication factor for important internal data is 5 by default, so no adjustments are needed for internal data. In this case, if 2 nodes fail at the same time, each range retains 3 of its 5 replicas, a majority.

- When deploying across multiple availability zones:
  - To be able to tolerate the failure of 1 entire AZ in a region, use at least 3 AZs per region and set `--locality` on each node to spread data evenly across regions and AZs. In this case, if 1 AZ goes offline, the 2 remaining AZs retain a majority of replicas.
  - To be able to tolerate the failure of 1 entire region, use at least 3 regions.

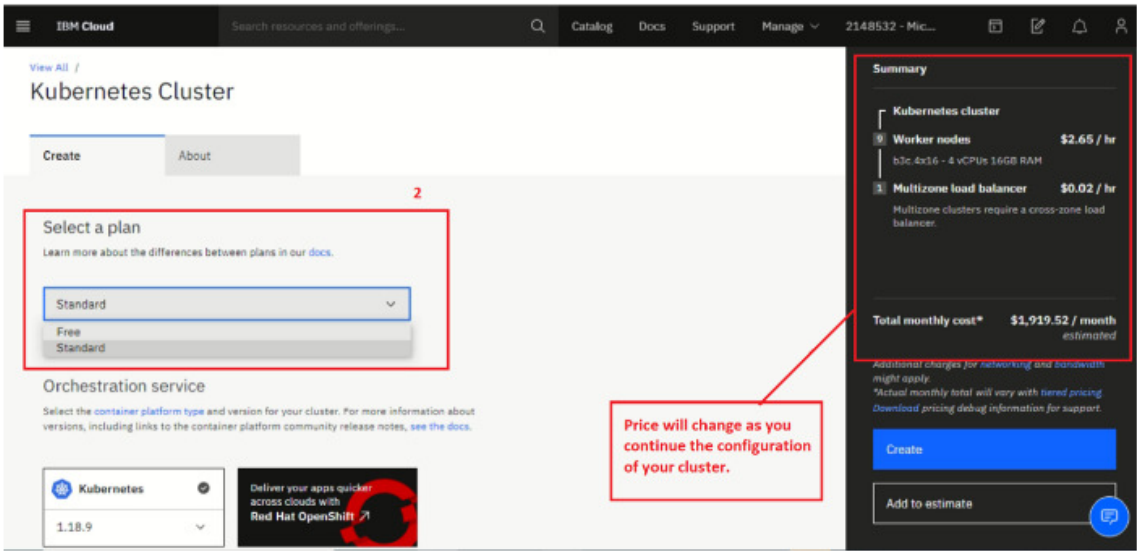
Step 1. Create a Kubernetes Cluster

After opening your IBM Cloud account, you need to set up K8s in a configuration which can support CockroachDB. The first step is to create a cluster with minimum resources:

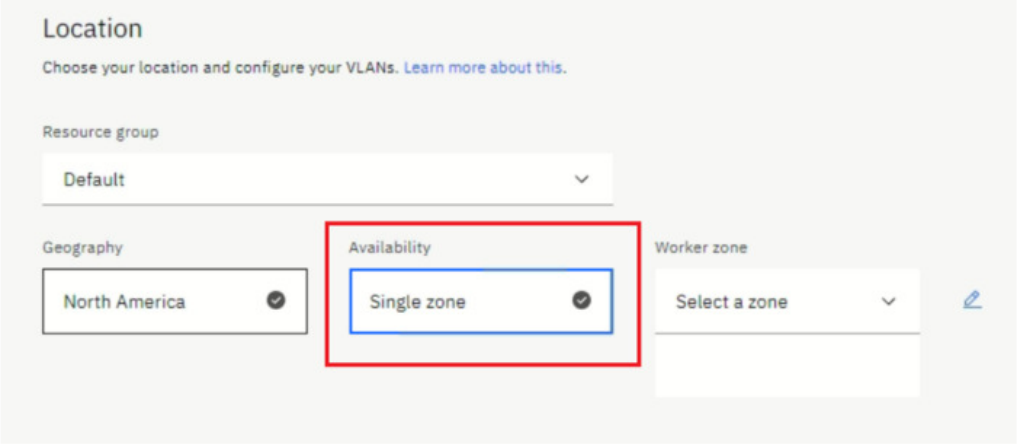
1. Go to the **main menu/Kubernetes/Cluster** and select "**Create Cluster**".



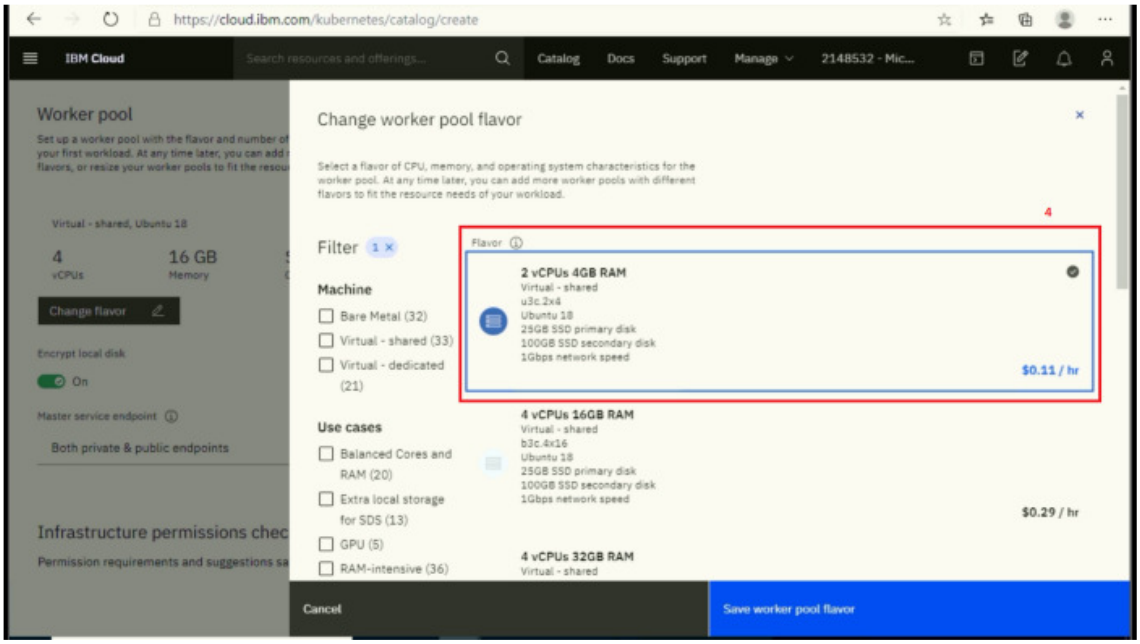
2 . You now have two options: a free cluster (will expire in 30 days) or 'Standard' cluster. When you choose 'Standard' watch for the price on the right hand side, as it will change as you continue the configuration as you scroll down. Use the latest/recommended version.

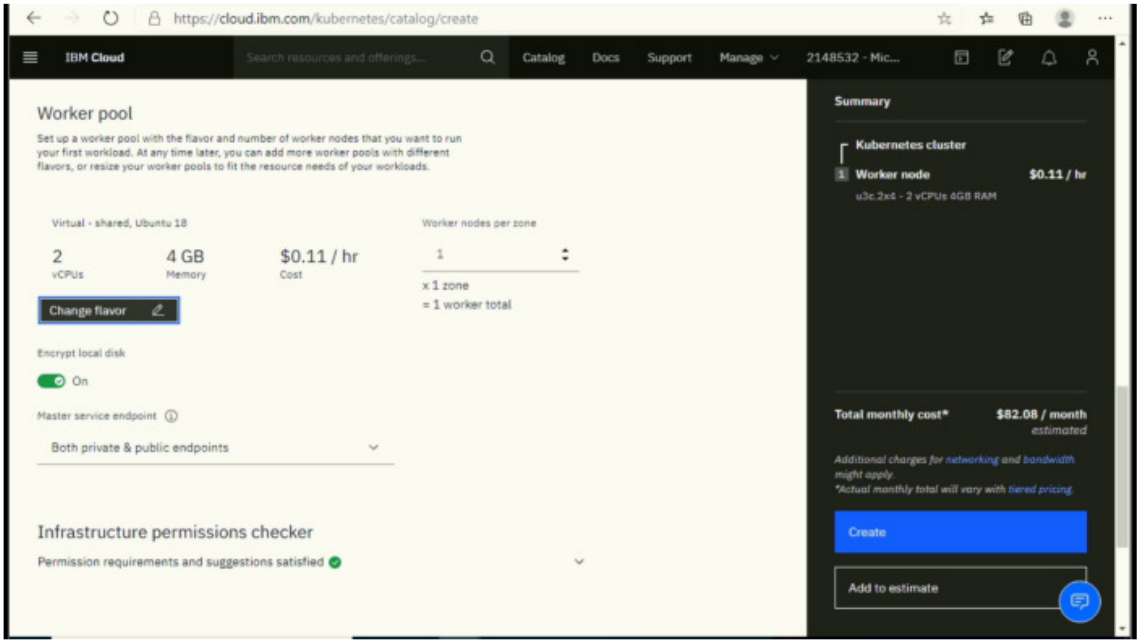


3. Classic infrastructure should be fine to choose. The next step is to configure the location settings, choosing between single or multi zone.

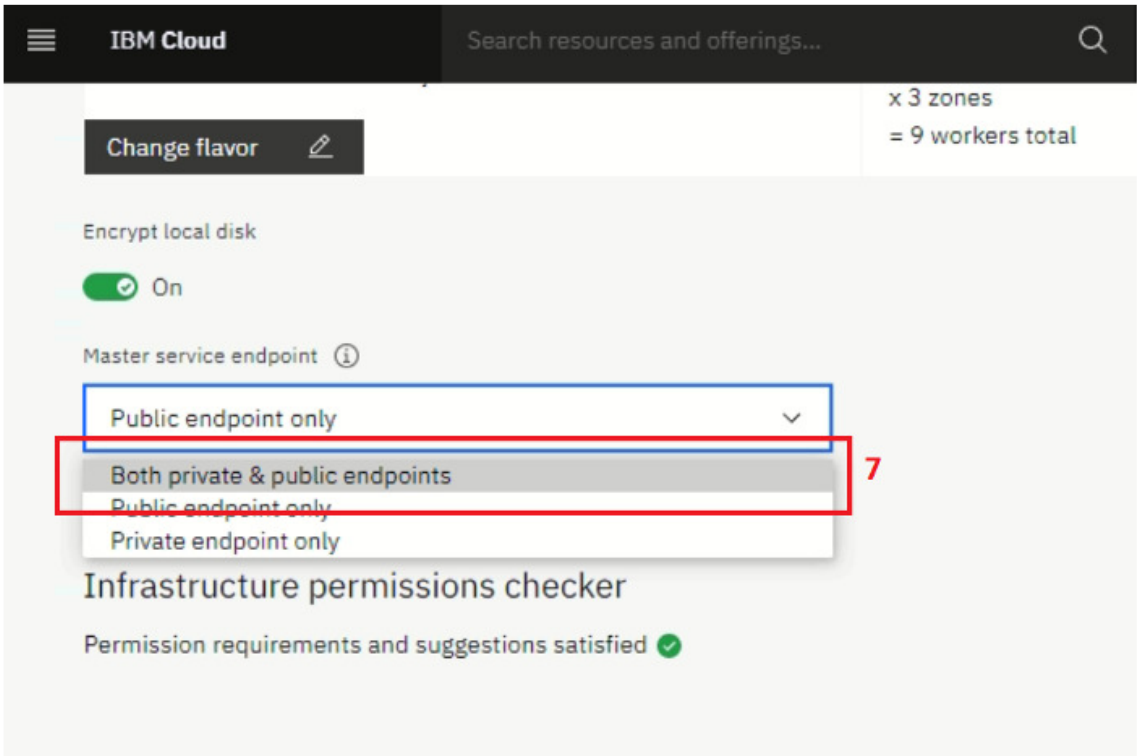


4 . Configure the Worker pool and choose the hardware which best suits your requirments.





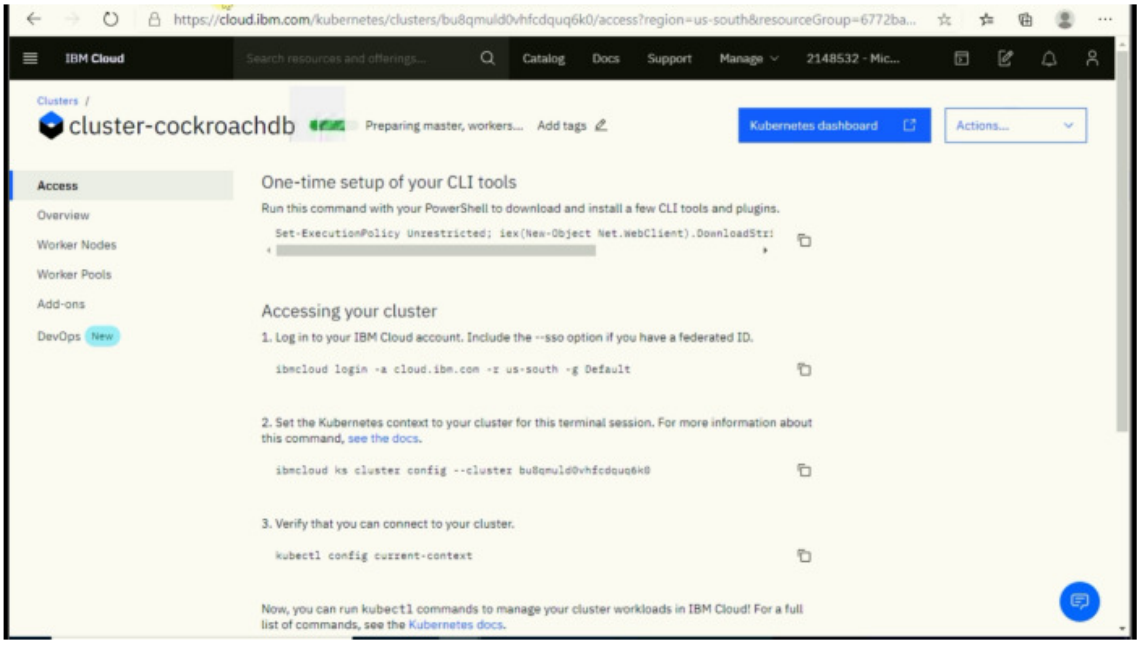
5. Make sure to choose both private and public endpoints to access clusters on the internet. It is remanded to choose “Encrypt local disk” to be secure.



6. Name your cluster.

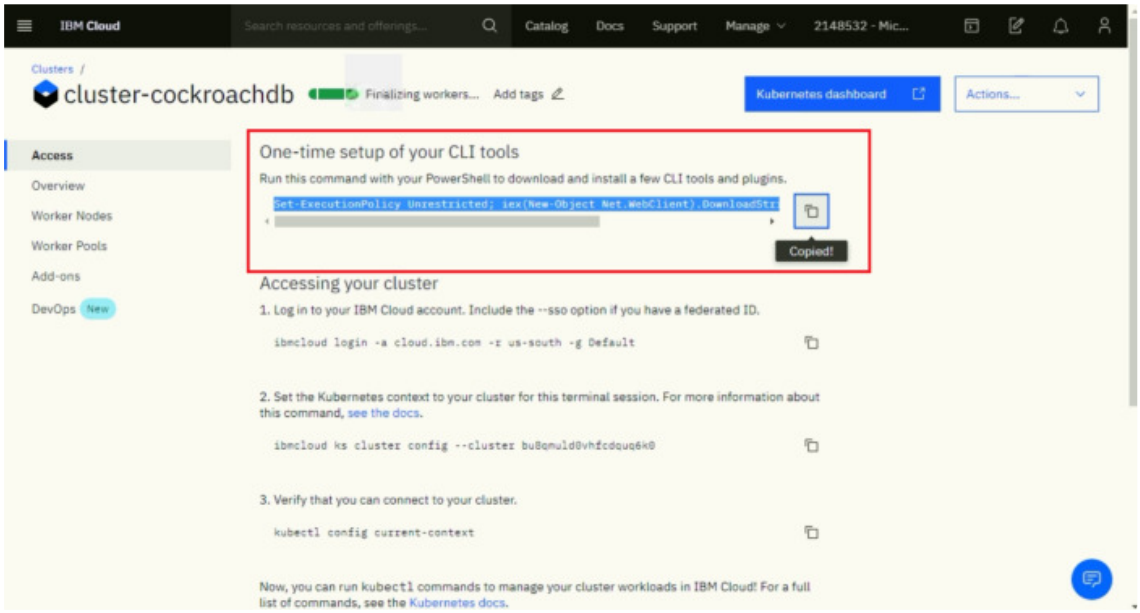
7. Finally - create our cluster. This process can take up to 10-20 minutes, depending on your configuration.

In the creation screen you already have some useful scripts for setting up CLI and accessing your cluster.

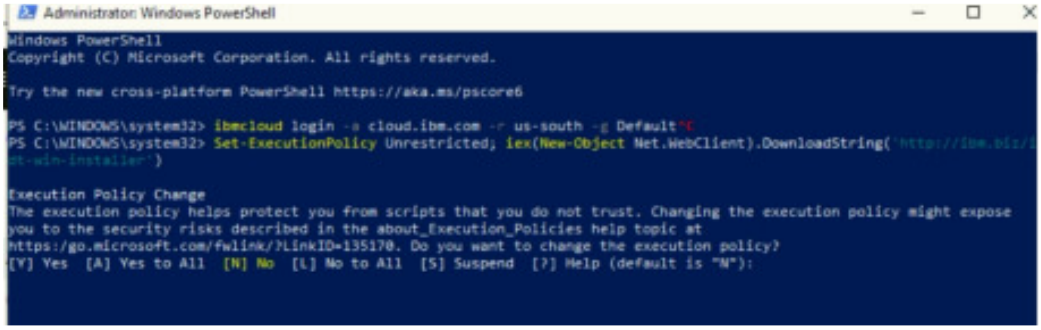


Step 2. Configure IBMCloud binaries an Helm

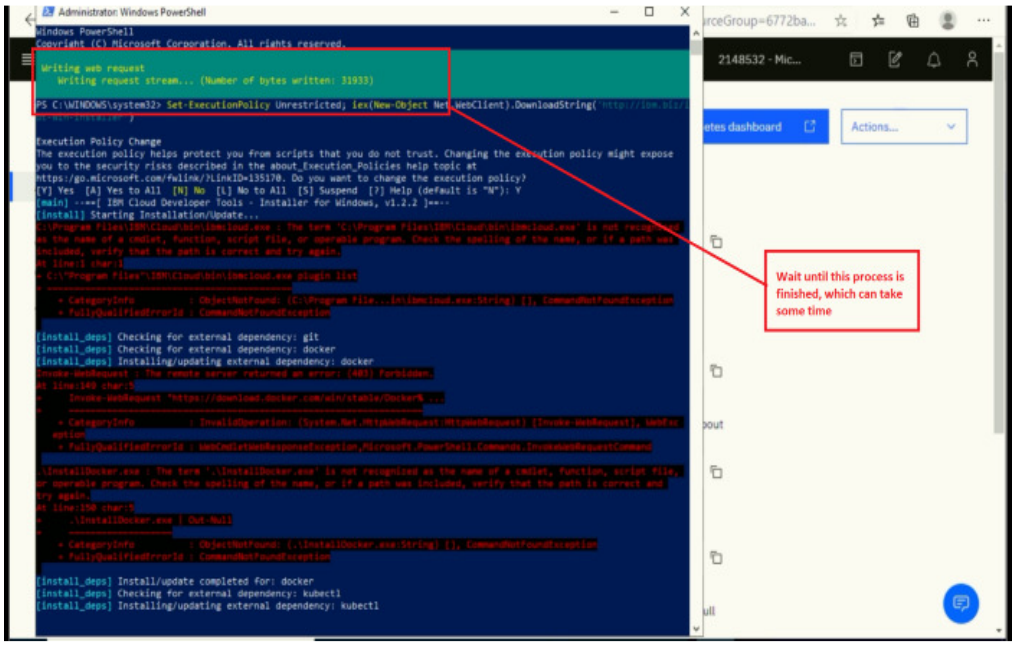
1. Copy the script from your screen and paste it in your PowerShell (Run as admin).



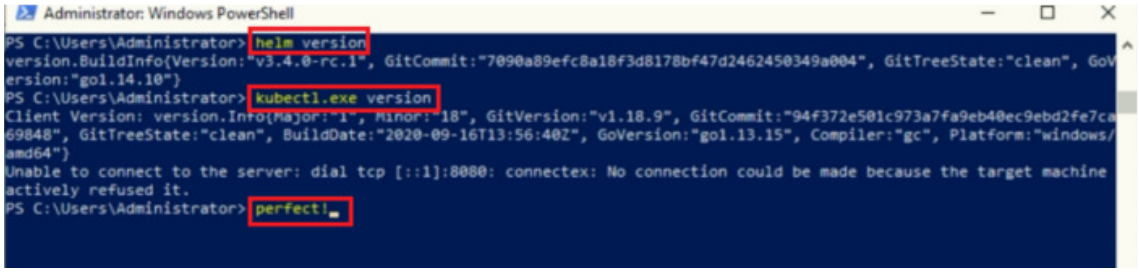
2. Press ‘Y’ when prompt for policy change.



3. Install all plugins (kubectl, Helm etc) when prompted.

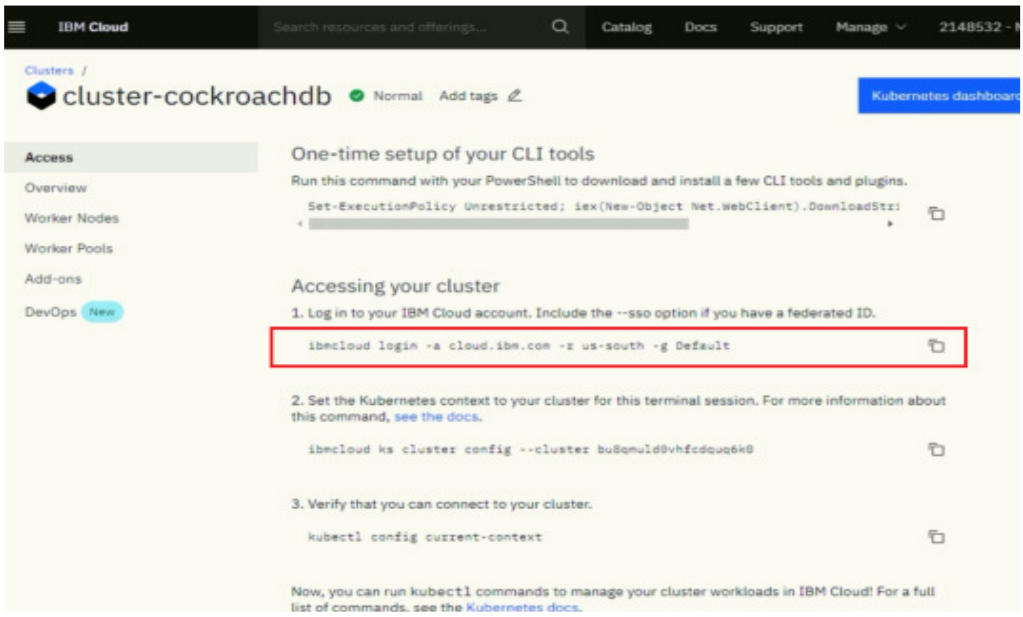


4. When setup is finished make sure you have helm and kubectl installed:



Step 3. Accessing your Cluster

1. Copy the second command from the Kubernetes installation page and execute on the terminal



2. Enter IBMCloud credentials:

```
PS C:\Users\Administrator\Desktop\helm-charts-master> ls


    Directory: C:\Users\Administrator\Desktop\helm-charts-master


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        10/22/2020   9:39 AM                build
d-----        10/22/2020   9:39 AM                cockroachdb
-a----        10/12/2020   1:16 PM          11357 LICENSE
-a----        10/12/2020   1:16 PM            194 Makefile
-a----        10/12/2020   1:16 PM            193 README.md


PS C:\Users\Administrator\Desktop\helm-charts-master> cd .\cockroachdb\
PS C:\Users\Administrator\Desktop\helm-charts-master\cockroachdb> ibmcloud login -a cloud.ibm.com

API endpoint: https://cloud.ibm.com

Email> [redacted]

Password>
Authenticating...
OK

Targeted account [redacted]

Targeted resource group Default

Targeted region us-south


API endpoint:      https://cloud.ibm.com
Region:            us-south
User:              [redacted]
Account:           [redacted]
Resource group:    Default
CF API endpoint:
Org:
Space:
```

3. Verify the cluster by running this command

```
Kubectl get nodes
```

```
PS C:\Users\Administrator\Desktop\helm-charts-master\cockroachdb> kubectl.exe get nodes
NAME            STATUS   ROLES    AGE   VERSION
10.177.20.40    Ready    <none>   32m   v1.18.9+IKS
PS C:\Users\Administrator\Desktop\helm-charts-master\cockroachdb>
```

Step 4. Deploying CockroachDB via Helm

1. Install the Helm client (version 3.0 or higher) and add the `cockroachdb` chart repository:

```
helm repo add cockroachdb https://charts.cockroachdb.com/
"cockroachdb" has been added to your repositories
```

2. Update your Helm chart repositories to ensure that you're using the latest CockroachDB chart:

```
helm repo update
```

3. Modify our Helm chart's values.yaml parameters for your deployment scenario.

   Create a my-values.yaml file to override the defaults in values.yaml, substituting your own values in this example based on the guidelines below.

```
Statefulset:
  replicas: 1
  resources:
    limits:
      memory: "8Gi"
    requests:
      memory: "8Gi"
conf:
  cache: "2Gi"
  max-sql-memory: "2Gi"
tls:
  enabled: false
```

   a. To avoid running out of memory when CockroachDB is not the only pod on a Kubernetes node, you *must* set memory limits explicitly. This is because CockroachDB does not detect the amount of memory allocated to its pod when run in Kubernetes. We recommend setting conf.cache and conf.max-sql-memory each to 1/4 of the memory allocation specified in statefulset.resources.requests and statefulset.resources.limits.
   b. You may want to modify storage.persistentVolume.size and storage.persistentVolume.storageClass for your use case. This chart defaults to 100Gi of disk space per pod. For more details on customizing disks for performance, see these instructions.

NOTE:

If necessary, you can expand disk size after the cluster is live.

   c. For a insecure deployment, set tls.enabled to false.

4. Install the CockroachDB Helm chart.

   Provide a "release" name to identify and track this particular deployment of the chart, and override the default values with those in my-values.yaml.

NOTE:

This tutorial uses my-release as the release name. If you use a different value, be sure to adjust the release name in subsequent commands. Also be sure to start and end the name with an alphanumeric character and otherwise use lowercase alphanumeric characters, -, or . so as to comply with CSR naming requirements.

```
helm install my-release --values my-values.yaml cockroachdb/cockroachdb
```

Behind the scenes, this command uses our cockroachdb-statefulset.yaml file to create the StatefulSet that automatically creates 1 pod, each with a CockroachDB node running inside it, and always binds the persistent storage on restart.

5. Confirm that one pods is Running successfully:

```
kubectl get pods
NAME                          READY     STATUS     RESTARTS   AGE
my-release-cockroachdb-0          0/1       Running    0          6m
```

9. Confirm that the persistent volumes and corresponding claims were created successfully for the pod:

```
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM
POLICY   STATUS    CLAIM                                   STORAGECLASS
REASON    AGE
pvc-71019b3a-fc67-11e8-a606-080027ba45e5   100Gi      RWO            Delete
Bound     default/datadir-my-release-cockroachdb-0   standard                11m
```

TIP:

The StatefulSet configuration sets all CockroachDB nodes to log to stderr, so if you ever need access to a pod/node's logs to troubleshoot, use kubectl logs <podname> rather than checking the log on the persistent volume.

Step 8. Use the built-in SQL Client

To use the built-in SQL client, you need to launch a pod that runs indefinitely with the `cockroach` binary inside it, get a shell into the pod, and then start the built-in SQL client.

1. From your local workstation get a shell into the pod and start the insecure CockroachDB built-in SQL client:

```
kubectl exec -it cockroachdb-client-secure -- ./cockroach sql --insecure --host=my-release-cockroachdb-public

# Welcome to the cockroach SQL interface.
# All statements must be terminated by a semicolon.
# To exit: CTRL + D.
#
# Client version: CockroachDB CCL v19.1.0 (x86_64-unknown-linux-gnu, built 2019/04/29 18:36:40, go1.11.6)
# Server version: CockroachDB CCL v19.1.0 (x86_64-unknown-linux-gnu, built 2019/04/29 18:36:40, go1.11.6)

# Cluster ID: 256a8705-e348-4e3a-ab12-e1aba96857e4
#
# Enter \? for a brief introduction.
#
root@my-release-cockroachdb-public:26257/defaultdb>
```

3. Run some basic CockroachDB SQL statements:

```
> CREATE DATABASE bank;

> CREATE TABLE bank.accounts (id INT PRIMARY KEY, balance DECIMAL);

> INSERT INTO bank.accounts VALUES (1, 1000.50);

> SELECT * FROM bank.accounts;

        id | balance

        +----+---------+

        1 | 1000.50

         (1 row)
```

4. Create a user with a password:

```
> CREATE USER roach WITH PASSWORD 'Q7gc8rEdS';
```

5. Exit the SQL shell and pod:

```
> \q
```

TIP:

This pod will continue running indefinitely, so any time you need to reopen the built-in SQL client or run any other cockroach client commands (e.g., cockroach node), repeat step 2 using the appropriate cockroach command.

If you'd prefer to delete the pod and recreate it when needed, run kubectl delete pod cockroachdb-client-secure.

Step 9. Access the Admin UI

To access the cluster's Admin UI:

1. On secure clusters, certain pages of the Admin UI can only be accessed by admin users.
   Get a shell into the pod and start the CockroachDB built-in SQL client:

   ```
   kubectl exec -it cockroachdb-client-secure -- ./cockroach sql --insecure --host=my-release-cockroachdb-public
   ```

2. Assign roach to the admin role (you only need to do this once):

   ```
   GRANT admin TO roach;
   ```

3. Exit the SQL shell and pod:

   ```
   > \q
   ```

4. In a new terminal window, port-forward from your local machine to one of the pods:

   ```
   kubectl port-forward my-release-cockroachdb-0 8080
   ```

```
Forwarding from 127.0.0.1:8080 -> 8080
```

5. Go to https://localhost:8080 and log in with the username and password you created earlier.

6. In the UI, verify that the cluster is running as expected:
   - Click **View nodes list** on the right to ensure that all nodes successfully joined the cluster.
   - Click the **Databases** tab on the left to verify that `bank` is listed.

Step 10. Monitor the cluster

Despite CockroachDB's various built-in safeguards against failure, it is critical to actively monitor the overall health and performance of a cluster running in production and to create alerting rules that promptly send notifications when there are events that require investigation or intervention.

Step 11. Scale the cluster

In order to scale the cockroachdb cluster increase the ***statefulset.replicas*** from ***my-values.yaml*** file and upgrade the helm chart

```
helm upgrade --install my-release --values my-values.yaml cockroachdb/cockroachdb
```

Step 12. Use the database

Now that your deployment is working, you can:

1. Implement your data model.
2. Create users and grant them privileges.
3. Connect your application. Be sure to connect your application to the load balancer, not to a CockroachDB node.

You may also want to adjust the way the cluster replicates data. For example, by default, a multi-node cluster replicates all data 3 times; you can change this replication factor or create additional rules for replicating individual databases and tables differently. For more information, see Configure Replication Zones.

WARNING:
When running a cluster of 5 nodes or more, it's safest to increase the replication factor for important internal data to 5, even if you do not do so for user data. For the cluster as a whole to remain available, the ranges for this internal data must always retain a majority of their replicas.

See also

- Production Checklist
- Manual Deployment
- Orchestrated Deployment
- Monitoring and Alerting
- Performance Benchmarking
- Performance Tuning
- Local Deployment

Deploy CockroachDB on IBM Cloud Kubernetes

This page shows you how to manually deploy a secure multi-node CockroachDB cluster on IBM Cloud Kubernetes platform.

If you are only testing CockroachDB, or you are not concerned with protecting network communication with TLS encryption, you can use an insecure cluster instead.

> *TIP:*
> *To deploy a 30-day free CockroachCloud cluster instead of running CockroachDB yourself, see the Quickstart.*

Requirements

- You must have an active IBMCloud account.
- Enough credits to run a Kubernetes (k8s) cluster.
- Microsoft PowerShell (For Windows) or Bash Shell for Linux distribution.
- Carefully review the Production Checklist and recommended Topology Patterns.
- When deploying in a single availability zone:
    - To be able to tolerate the failure of any 1 node, use at least 3 nodes with the default 3-way replication factor. In this case, if 1 node fails, each range retains 2 of its 3 replicas, a majority.
    - To be able to tolerate 2 simultaneous node failures, use at least 5 nodes and increase the default replication factor for user data to 5. The

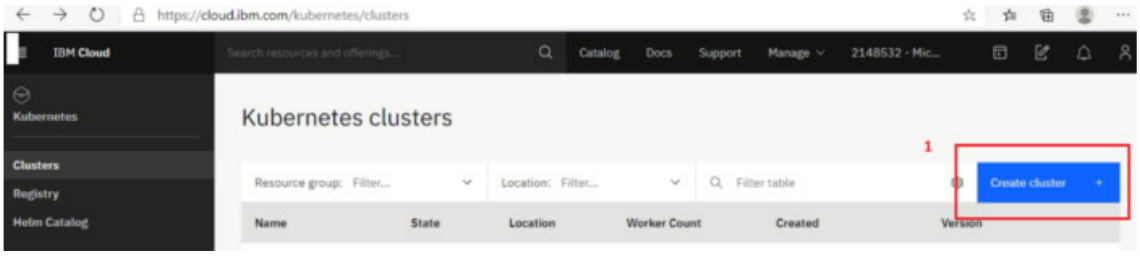
replication factor for important internal data is 5 by default, so no adjustments are needed for internal data. In this case, if 2 nodes fail at the same time, each range retains 3 of its 5 replicas, a majority.

- When deploying across multiple availability zones:
  - To be able to tolerate the failure of 1 entire AZ in a region, use at least 3 AZs per region and set `--locality` on each node to spread data evenly across regions and AZs. In this case, if 1 AZ goes offline, the 2 remaining AZs retain a majority of replicas.
  - To be able to tolerate the failure of 1 entire region, use at least 3 regions.

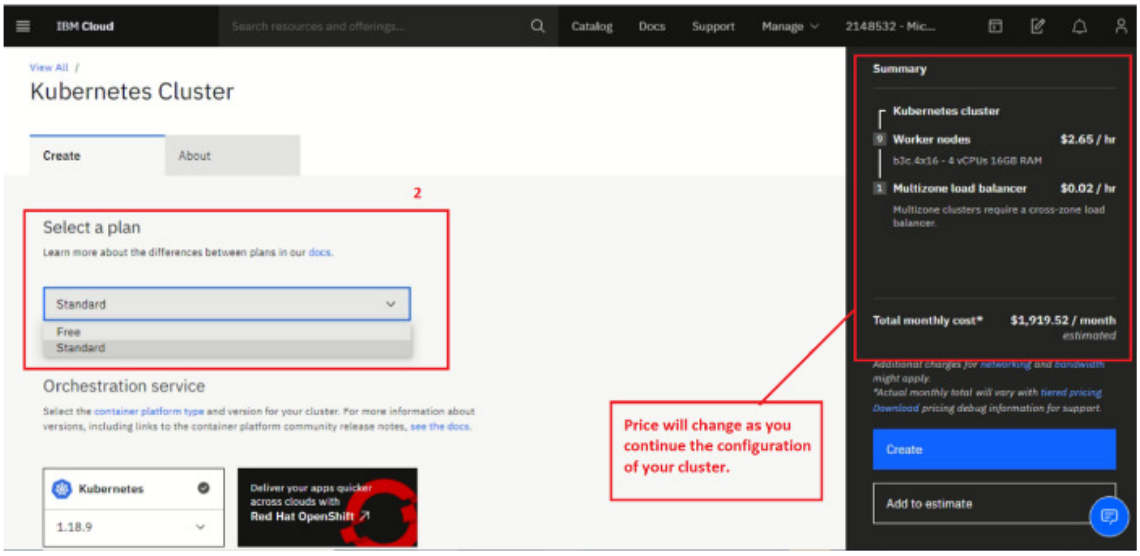
Step 1. Create a Kubernetes Cluster

After opening your IBM Cloud account, you need to set up K8s in a configuration which can support CockroachDB. The first step is to create a cluster with minimum resources:

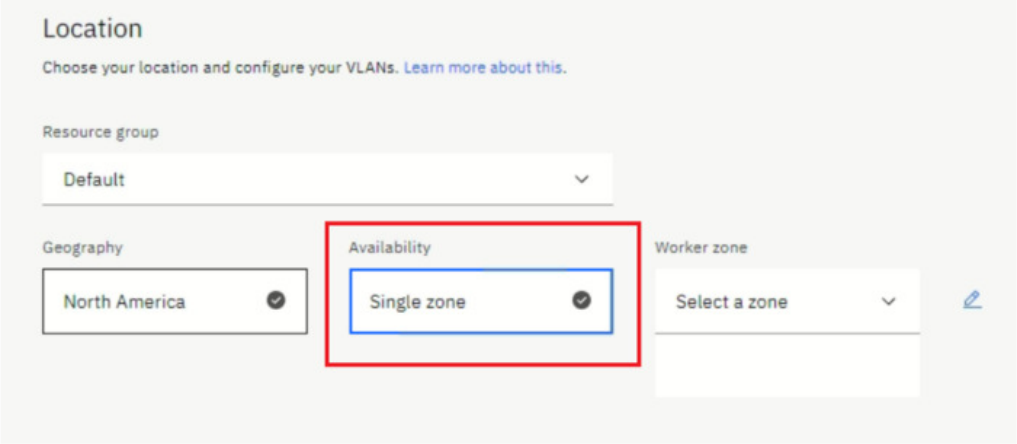
1. Go to the **main menu/Kubernetes/Cluster** and select "**Create Cluster**".



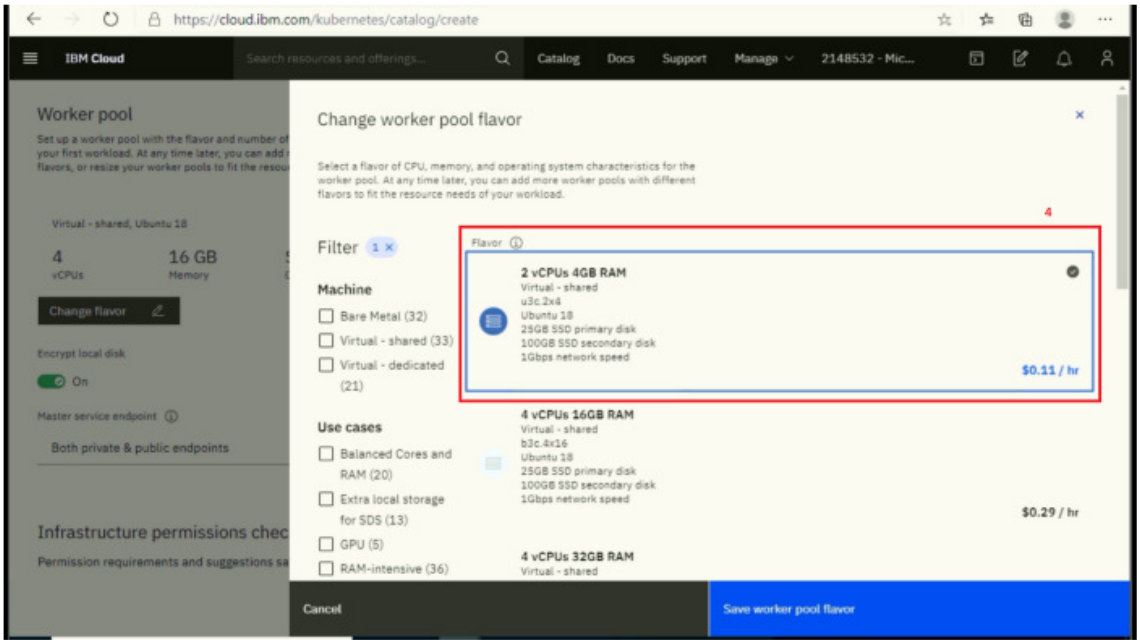
2 . You now have two options: a free cluster (will expire in 30 days) or 'Standard' cluster. When you choose 'Standard' watch for the price on the right hand side, as it will change as you continue the configuration as you scroll down. Use the latest/recommended version.

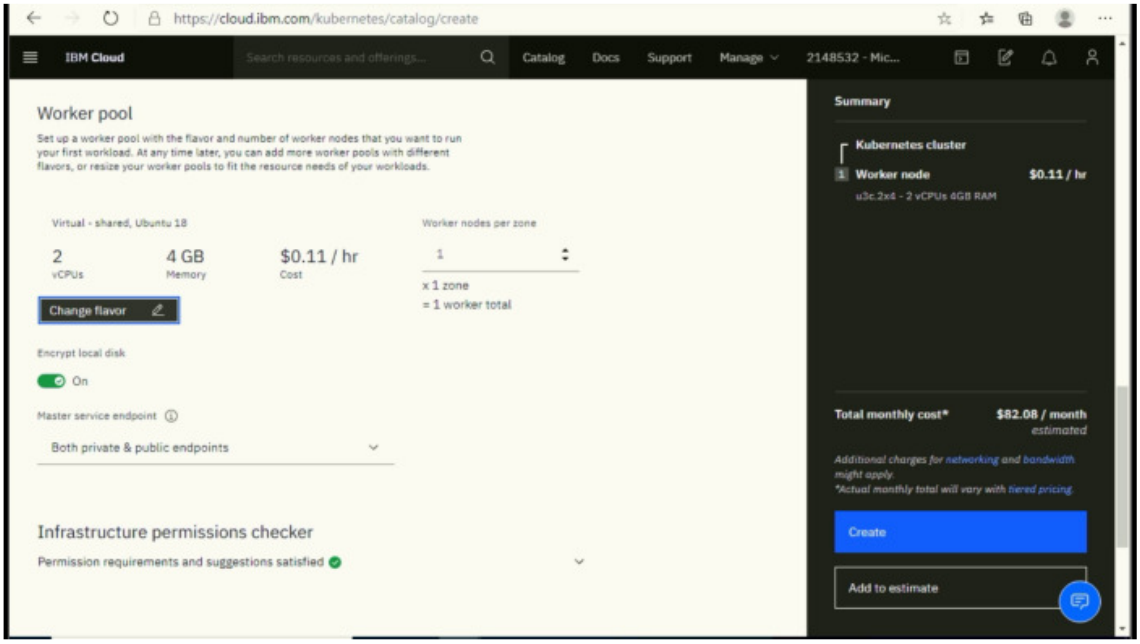


3. Classic infrastructure should be fine to choose. The next step is to configure the location settings, choosing between single or multi zone.

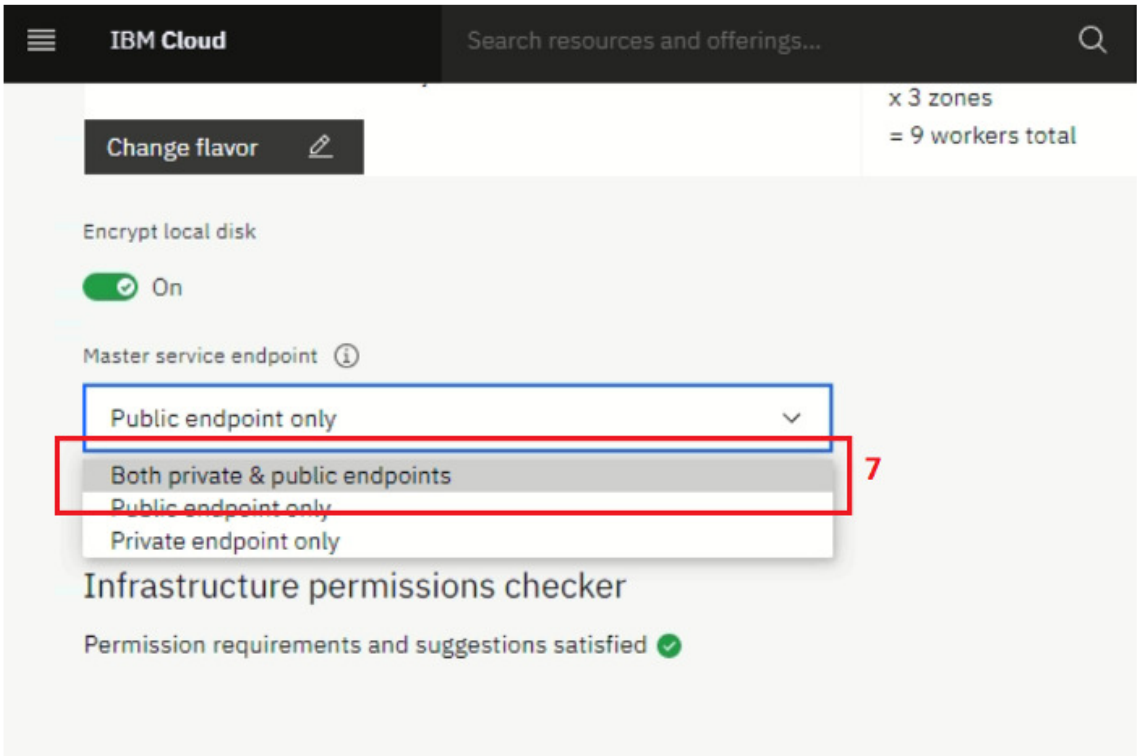


4 . Configure the Worker pool and choose the hardware which best suits your requirments.





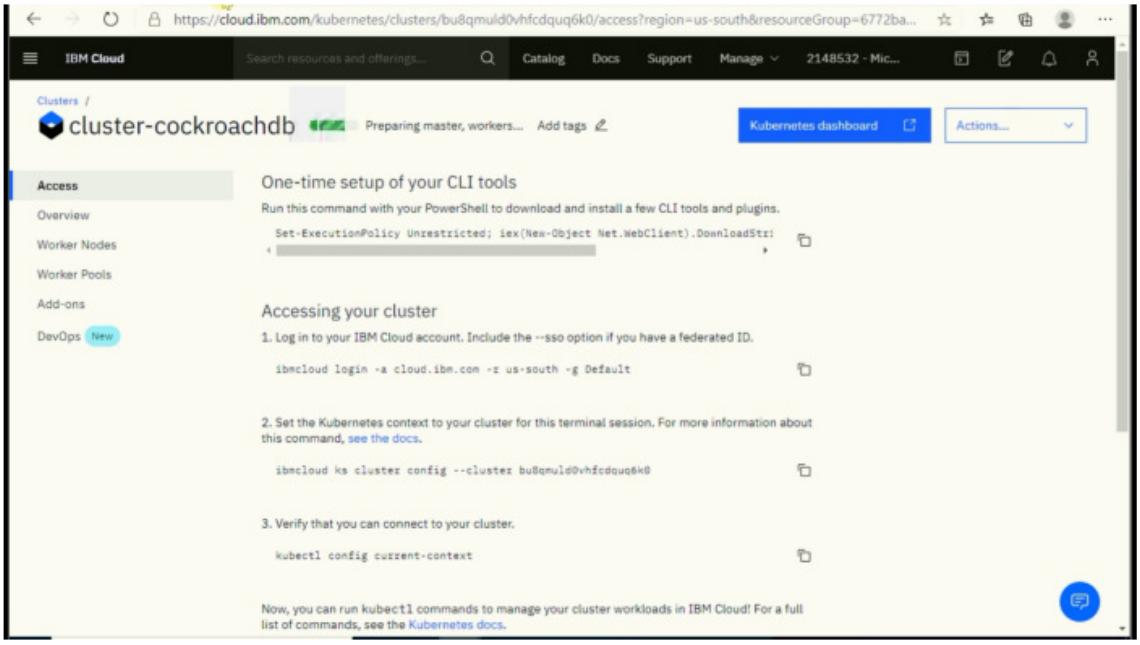
5. Make sure to choose both private and public endpoints to access clusters on the internet. It is remanded to choose “Encrypt local disk” to be secure.



6. Name your cluster.

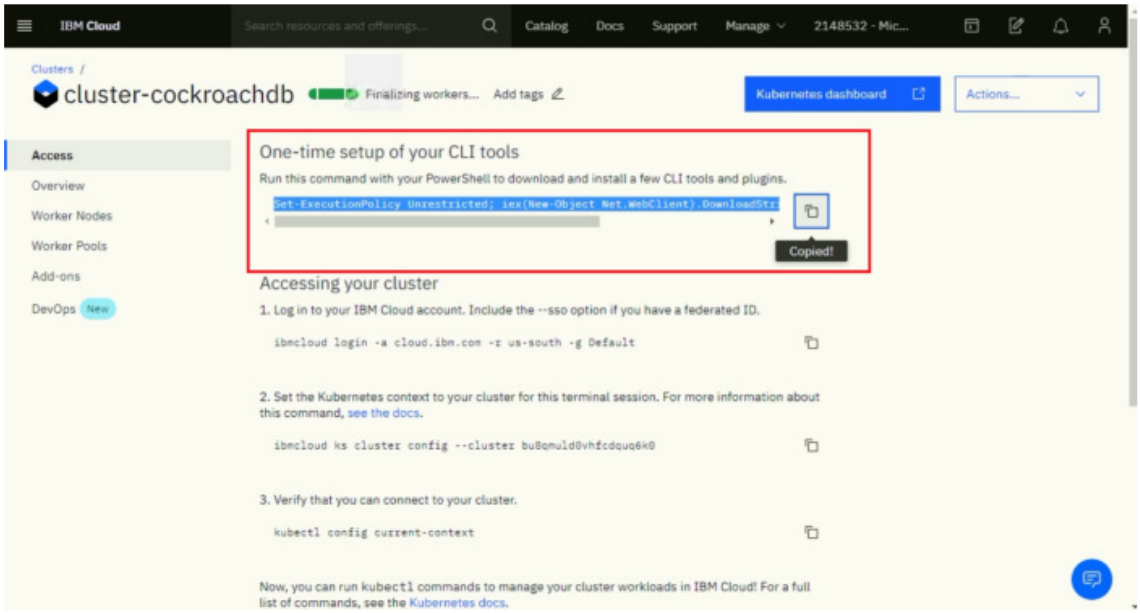
7. Finally - create our cluster. This process can take up to 10-20 minutes, depending on your configuration.

In the creation screen you already have some useful scripts for setting up CLI and accessing your cluster.

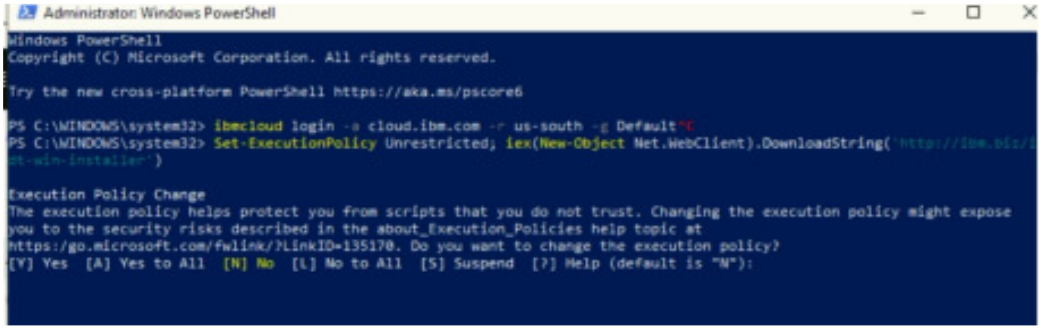


Step 2. Configure IBMCloud binaries an Helm

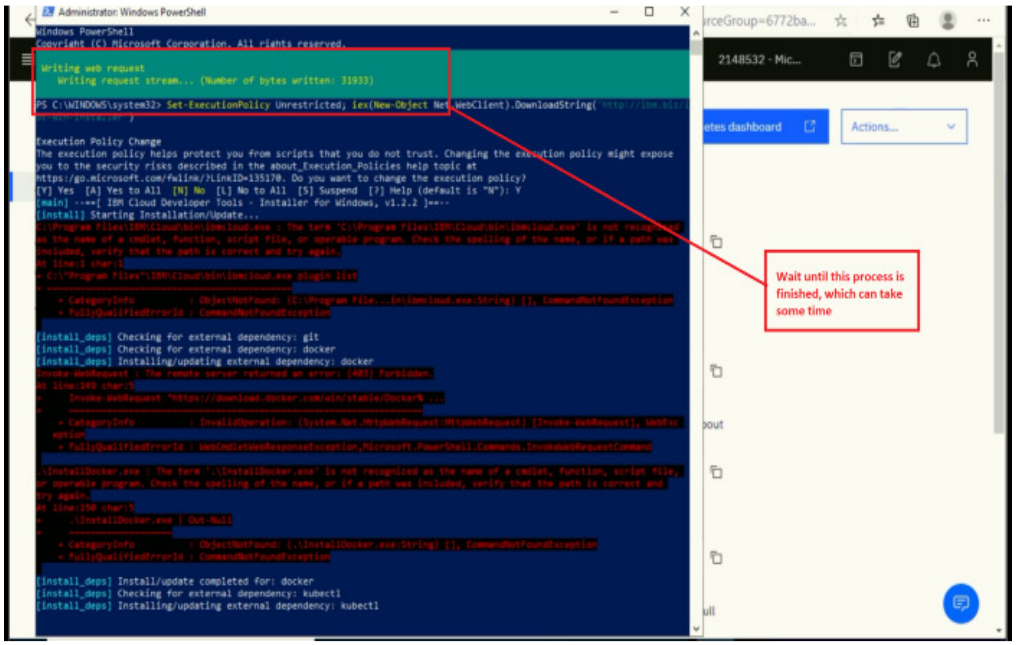
1. Copy the script from your screen and paste it in your PowerShell (Run as admin).



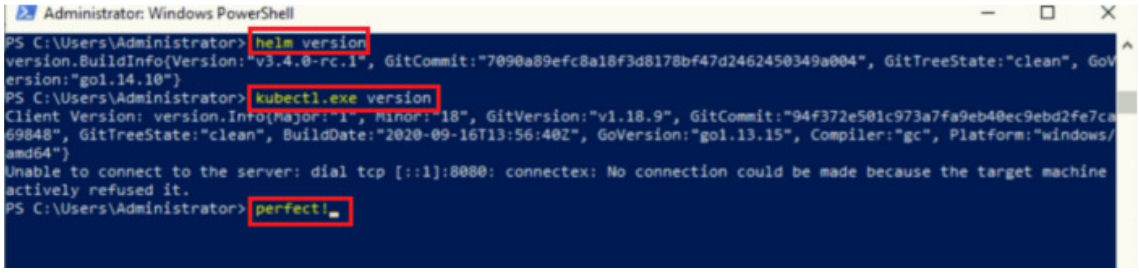
2. Press ‘Y’ when prompt for policy change.



3. Install all plugins (kubectl, Helm etc) when prompted.

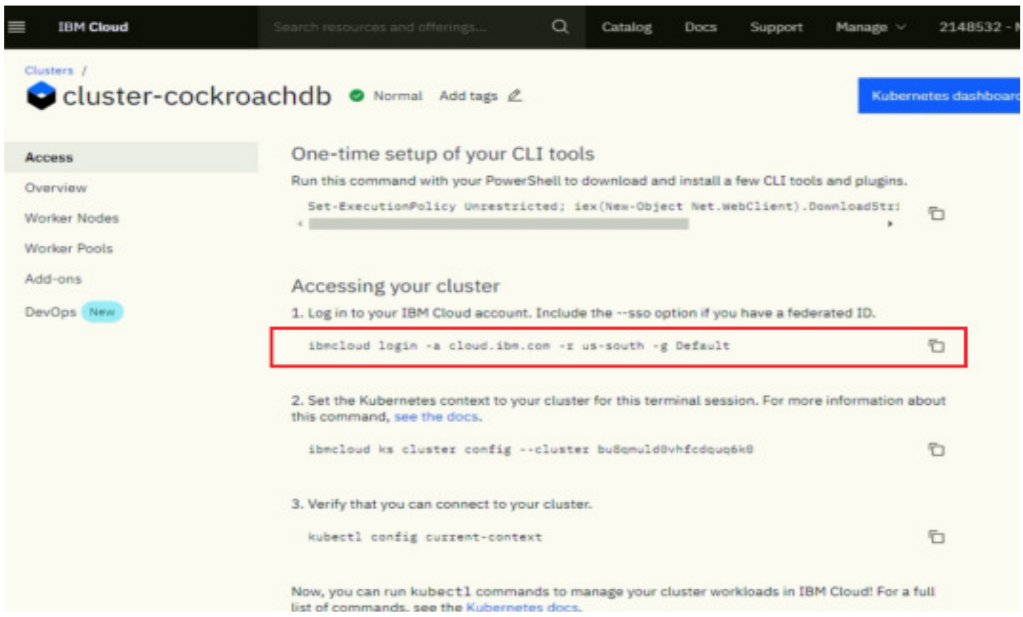


4. When setup is finished make sure you have helm and kubectl installed:



Step 3. Accessing your Cluster

1. Copy the second command from the Kubernetes installation page and execute on the terminal



2. Enter IBMCloud credentials:

```
PS C:\Users\Administrator\Desktop\helm-charts-master> ls


    Directory: C:\Users\Administrator\Desktop\helm-charts-master


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        10/22/2020   9:39 AM                build
d-----        10/22/2020   9:39 AM                cockroachdb
-a----        10/12/2020   1:16 PM          11357 LICENSE
-a----        10/12/2020   1:16 PM            194 Makefile
-a----        10/12/2020   1:16 PM            193 README.md


PS C:\Users\Administrator\Desktop\helm-charts-master> cd .\cockroachdb\
PS C:\Users\Administrator\Desktop\helm-charts-master\cockroachdb> ibmcloud login -a cloud.ibm.com

API endpoint: https://cloud.ibm.com

Email> [redacted]

Password>
Authenticating...
OK

Targeted account [redacted]

Targeted resource group Default

Targeted region us-south


API endpoint:      https://cloud.ibm.com
Region:            us-south
User:              [redacted]
Account:           [redacted]
Resource group:    Default
CF API endpoint:
Org:
Space:
```

3. Verify the cluster by running this command

```
Kubectl get nodes
```

```
PS C:\Users\Administrator\Desktop\helm-charts-master\cockroachdb> kubectl.exe get nodes
NAME            STATUS   ROLES    AGE   VERSION
10.177.20.40    Ready    <none>   32m   v1.18.9+IKS
PS C:\Users\Administrator\Desktop\helm-charts-master\cockroachdb>
```

Step 4. Deploying CockroachDB via Helm

1. Install the Helm client (version 3.0 or higher) and add the `cockroachdb` chart repository:

```
helm repo add cockroachdb https://charts.cockroachdb.com/
"cockroachdb" has been added to your repositories
```

2. Update your Helm chart repositories to ensure that you're using the latest CockroachDB chart:

```
helm repo update
```

3. Modify our Helm chart's values.yaml parameters for your deployment scenario.

   Create a my-values.yaml file to override the defaults in values.yaml, substituting your own values in this example based on the guidelines below.

```
statefulset:
  resources:
    limits:
      memory: "8Gi"
    requests:
      memory: "8Gi"
conf:
  cache: "2Gi"
  max-sql-memory: "2Gi"
tls:
  enabled: true
```

   a. To avoid running out of memory when CockroachDB is not the only pod on a Kubernetes node, you *must* set memory limits explicitly. This is because CockroachDB does not detect the amount of memory allocated to its pod when run in Kubernetes. We recommend setting conf.cache and conf.max-sql-memory each to 1/4 of the memory allocation specified in statefulset.resources.requests and statefulset.resources.limits.

TIP:

For example, if you are allocating 8Gi of memory to each CockroachDB node, allocate 2Gi to cache and 2Gi to max-sql-memory.

   b. You may want to modify storage.persistentVolume.size and storage.persistentVolume.storageClass for your use case. This chart defaults to 100Gi of disk space per pod. For more details on customizing disks for performance, see these instructions.

NOTE:

If necessary, you can expand disk size after the cluster is live.

   c. For a secure deployment, set tls.enabled to true.

4. Install the CockroachDB Helm chart.

Provide a "release" name to identify and track this particular deployment of the chart, and override the default values with those in `my-values.yaml`.

> ***NOTE:***
>
> *This tutorial uses my-release as the release name. If you use a different value, be sure to adjust the release name in subsequent commands. Also be sure to start and end the name with an alphanumeric character and otherwise use lowercase alphanumeric characters, -, or . so as to comply with CSR naming requirements.*

```
helm install my-release --values my-values.yaml cockroachdb/cockroachdb
```

Behind the scenes, this command uses our cockroachdb-statefulset.yaml file to create the StatefulSet that automatically creates 3 pods, each with a CockroachDB node running inside it, where each pod has distinguishable network identity and always binds back to the same persistent storage on restart.

5. As each pod is created, it issues a Certificate Signing Request, or CSR, to have the CockroachDB node's certificate signed by the Kubernetes CA. You must manually check and approve each node's certificate, at which point the CockroachDB node is started in the pod.

   a. Get the names of the `Pending` CSRs:

```
kubectl get csr

NAME                                                  AGE       REQUESTOR
CONDITION

default.client.root                                   21s       system:serviceaccount:default:my-
release-cockroachdb   Pending

default.node.my-release-cockroachdb-0   15s
system:serviceaccount:default:my-release-cockroachdb   Pending

default.node.my-release-cockroachdb-1   16s
system:serviceaccount:default:my-release-cockroachdb   Pending

default.node.my-release-cockroachdb-2   15s
system:serviceaccount:default:my-release-cockroachdb   Pending
```

   If you do not see a Pending CSR, wait a minute and try again.

   b. Examine the CSR for the first pod:

```
kubectl describe csr default.node.my-release-cockroachdb-0
Name:              default.node.my-release-cockroachdb-0
Labels:            <none>
Annotations:       <none>
CreationTimestamp:  Mon, 10 Dec 2018 05:36:35 -0500
Requesting User:    system:serviceaccount:default:my-release-cockroachdb
Status:            Pending
Subject:
  Common Name:    node
  Serial Number:
  Organization:   Cockroach
Subject Alternative Names:
        DNS Names:     localhost
                   my-release-cockroachdb-0.my-release-
cockroachdb.default.svc.cluster.local
                   my-release-cockroachdb-0.my-release-cockroachdb
                   my-release-cockroachdb-public
                   my-release-cockroachdb-public.default.svc.cluster.local
        IP Addresses:  127.0.0.1
Events:  <none>
```

c. If everything looks correct, approve the CSR for the first pod:

```
kubectl certificate approve default.node.my-release-cockroachdb-0
certificatesigningrequest.certificates.k8s.io/default.node.my-release-
cockroachdb-0 approved
```

d. Repeat steps 2 and 3 for the other 2 pods.

6. Confirm that three pods are Running successfully:

```
kubectl get pods
NAME                          READY     STATUS     RESTARTS   AGE
```

```
my-release-cockroachdb-0          0/1     Running   0          6m
my-release-cockroachdb-1          0/1     Running   0          6m
my-release-cockroachdb-2          0/1     Running   0          6m
my-release-cockroachdb-init-hxzsc   0/1     Init:0/1   0          6m
```

7. Approve the CSR for the one-off pod from which cluster initialization happens:

```
kubectl certificate approve default.client.root
certificatesigningrequest.certificates.k8s.io/default.client.root approved
```

8. Confirm that CockroachDB cluster initialization has completed successfully, with the pods for CockroachDB showing `1/1` under `READY` and the pod for initialization showing `COMPLETED` under `STATUS`:

```
kubectl get pods
NAME                          READY    STATUS     RESTARTS   AGE
my-release-cockroachdb-0          1/1     Running   0          8m
my-release-cockroachdb-1          1/1     Running   0          8m
my-release-cockroachdb-2          1/1     Running   0          8m
my-release-cockroachdb-init-hxzsc   0/1     Completed   0          1h
```

9. Confirm that the persistent volumes and corresponding claims were created successfully for all three pods:

```
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM
POLICY   STATUS    CLAIM                                  STORAGECLASS
REASON    AGE
pvc-71019b3a-fc67-11e8-a606-080027ba45e5   100Gi      RWO            Delete
Bound     default/datadir-my-release-cockroachdb-0   standard                11m
pvc-7108e172-fc67-11e8-a606-080027ba45e5   100Gi      RWO            Delete
Bound     default/datadir-my-release-cockroachdb-1   standard                11m
```

```
pvc-710dcb66-fc67-11e8-a606-080027ba45e5   100Gi      RWO           Delete
Bound     default/datadir-my-release-cockroachdb-2   standard               11m
```

TIP:

The StatefulSet configuration sets all CockroachDB nodes to log to stderr, so if you ever need access to a pod/node's logs to troubleshoot, use kubectl logs <podname> rather than checking the log on the persistent volume.

Step 8. Use the built-in SQL Client

To use the built-in SQL client, you need to launch a pod that runs indefinitely with the `cockroach` binary inside it, get a shell into the pod, and then start the built-in SQL client.

1. From your local workstation, use our `client-secure.yaml` file to launch a pod and keep it running indefinitely.
   a. Download the file:

      ```
      https://raw.githubusercontent.com/cockroachdb/cockroach/master/cloud/kubernetes/client-secure.yaml
      ```

   b. In the file, change `serviceAccountName: cockroachdb` to `serviceAccountName: my-release-cockroachdb`.

   c. Use the file to launch a pod and keep it running indefinitely:

      ```
      kubectl create -f client-secure.yaml
      ```

2. Get a shell into the pod and start the CockroachDB built-in SQL client:

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

   # Welcome to the cockroach SQL interface.
   # All statements must be terminated by a semicolon.
   # To exit: CTRL + D.
   #
   # Client version: CockroachDB CCL v19.1.0 (x86_64-unknown-linux-gnu, built 2019/04/29 18:36:40, go1.11.6)
   ```

```
# Server version: CockroachDB CCL v19.1.0 (x86_64-unknown-linux-gnu, built
2019/04/29 18:36:40, go1.11.6)

# Cluster ID: 256a8705-e348-4e3a-ab12-e1aba96857e4
#
# Enter \? for a brief introduction.
#
root@my-release-cockroachdb-public:26257/defaultdb>
```

3. Run some basic CockroachDB SQL statements:

```
> CREATE DATABASE bank;

> CREATE TABLE bank.accounts (id INT PRIMARY KEY, balance DECIMAL);

> INSERT INTO bank.accounts VALUES (1, 1000.50);

> SELECT * FROM bank.accounts;

      id | balance

      +----+---------+

      1 | 1000.50

       (1 row)
```

4. Create a user with a password:

```
> CREATE USER roach WITH PASSWORD 'Q7gc8rEdS';
```

5. Exit the SQL shell and pod:

```
> \q
```

TIP:

This pod will continue running indefinitely, so any time you need to reopen the built-in SQL client or run any other cockroach client commands (e.g., cockroach node), repeat step 2 using the appropriate cockroach command.

If you'd prefer to delete the pod and recreate it when needed, run kubectl delete pod cockroachdb-client-secure.

Step 9. Access the Admin UI

To access the cluster's Admin UI:

1. On secure clusters, certain pages of the Admin UI can only be accessed by `admin` users.
   Get a shell into the pod and start the CockroachDB built-in SQL client:

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

2. Assign `roach` to the `admin` role (you only need to do this once):

```
GRANT admin TO roach;
```

3. Exit the SQL shell and pod:

```
> \q
```

4. In a new terminal window, port-forward from your local machine to one of the pods:

```
kubectl port-forward my-release-cockroachdb-0 8080

Forwarding from 127.0.0.1:8080 -> 8080
```

5. Go to https://localhost:8080 and log in with the username and password you created earlier.

6. In the UI, verify that the cluster is running as expected:
   - Click **View nodes list** on the right to ensure that all nodes successfully joined the cluster.
   - Click the **Databases** tab on the left to verify that the bank is listed.

Step 10. Monitor the cluster

Despite CockroachDB's various built-in safeguards against failure, it is critical to actively monitor the overall health and performance of a cluster running in production and to create alerting rules that promptly send notifications when there are events that require investigation or intervention.

Step 11. Scale the cluster

In order to scale the cockroachdb cluster increase the ***statefulset.replicas*** from ***my-values.yaml*** file and upgrade the helm chart

```
helm upgrade --install my-release --values my-values.yaml cockroachdb/cockroachdb
```

Step 12. Use the database

Now that your deployment is working, you can:

1. Implement your data model.
2. Create users and grant them privileges.
3. Connect your application. Be sure to connect your application to the load balancer, not to a CockroachDB node.

You may also want to adjust the way the cluster replicates data. For example, by default, a multi-node cluster replicates all data 3 times; you can change this replication factor or create additional rules for replicating individual databases and tables differently. For more information, see Configure Replication Zones.

> WARNING:
> When running a cluster of 5 nodes or more, it's safest to increase the replication factor for important internal data to 5, even if you do not do so for user data. For the cluster as a whole to remain available, the ranges for this internal data must always retain a majority of their replicas.

See also

- Production Checklist
- Manual Deployment
- Orchestrated Deployment
- Monitoring and Alerting

- Performance Benchmarking
- Performance Tuning
- Local Deployment