Install a local Kubernetes with MicroK8s

1. Overview

What is Kubernetes

Kubernetes clusters host containerised applications in a reliable and scalable way. Having DevOps in mind, Kubernetes makes maintenance tasks such as upgrades dead simple.

What is MicroK8s

MicroK8s is a CNCF certified upstream Kubernetes deployment that runs entirely on your workstation or edge device. Being a snap it runs all Kubernetes services natively (i.e. no virtual machines) while packing the entire set of libraries and binaries needed. Installation is limited by how fast you can download a couple of hundred megabytes and the removal of MicroK8s leaves nothing behind.

In this tutorial you'll learn how to…

- Get your Kubernetes cluster up and running
- Enable core Kubernetes addons such as dns and dashboard
- Control your cluster from the kubectl CLI client
- Deploy your first container workload

You will only need …

- A machine with Linux

Install a local Kubernetes with MicroK8s

2. Deploying MicroK8s

If you are using Ubuntu, the quickest way to get started is to install MicroK8s directly from the snap store by clicking the "Install" button.

However, you can also install MicroK8s from the command line:

```
sudo snap install microk8s --classic
```

If you are using a different Linux distribution, you will have to install snapd first. Refer to Snapd documentation for more information on installing snapd on your Linux distribution.

For other platforms (Windows, macOS, Raspberry Pi etc) and install methods, please see the MicroK8s documentation

MicroK8s is a snap and as such it is frequently updated to each release of Kubernetes. To

follow a specific upstream release series it's possible to select a channel during installation. For example, to follow the v1.18 series:

```
sudo snap install microk8s --classic --channel=1.18/stable
```

Channels are made up of a track and an expected level of MicroK8s' stability. Try `snap info microk8s` to see what versions are currently published. At the time of this writing we have:

```
channels:
  latest/stable:    v1.19.0  2020-09-08 (1668) 214MB classic
  latest/candidate: v1.19.0  2020-09-03 (1668) 214MB classic
  latest/beta:      v1.19.0  2020-09-03 (1668) 214MB classic
  latest/edge:      v1.19.0  2020-09-04 (1673) 214MB classic
  dqlite/stable:    –
  dqlite/candidate: –
  dqlite/beta:      –
  dqlite/edge:      v1.16.2  2019-11-07 (1038) 189MB classic
  1.19/stable:      v1.19.0  2020-09-03 (1667) 214MB classic
  1.19/candidate:   v1.19.0  2020-09-02 (1667) 214MB classic
  1.19/beta:        v1.19.0  2020-09-02 (1667) 214MB classic
  1.19/edge:        v1.19.0  2020-09-04 (1674) 214MB classic
  1.18/stable:      v1.18.8  2020-08-25 (1609) 201MB classic
  1.18/candidate:   v1.18.8  2020-08-17 (1609) 201MB classic
  1.18/beta:        v1.18.8  2020-08-17 (1609) 201MB classic
  1.18/edge:        v1.18.8  2020-08-13 (1609) 201MB classic
  1.17/stable:      v1.17.11 2020-08-25 (1608) 179MB classic
  1.17/candidate:   v1.17.11 2020-08-21 (1608) 179MB classic
  1.17/beta:        v1.17.11 2020-08-21 (1608) 179MB classic
  1.17/edge:        v1.17.11 2020-08-13 (1608) 179MB classic
  1.16/stable:      v1.16.14 2020-08-23 (1606) 179MB classic
  1.16/candidate:   v1.16.15 2020-09-04 (1671) 179MB classic
  1.16/beta:        v1.16.15 2020-09-04 (1671) 179MB classic
  1.16/edge:        v1.16.15 2020-09-02 (1671) 179MB classic
  1.15/stable:      v1.15.11 2020-03-27 (1301) 171MB classic
  1.15/candidate:   v1.15.11 2020-03-27 (1301) 171MB classic
  1.15/beta:        v1.15.11 2020-03-27 (1301) 171MB classic
  1.15/edge:        v1.15.11 2020-03-26 (1301) 171MB classic
  1.14/stable:      v1.14.10 2020-01-06 (1120) 217MB classic
  1.14/candidate:   ↑
  1.14/beta:        ↑
```

```
  1.14/edge:        v1.14.10 2020-03-26 (1303) 217MB classic
  1.13/stable:      v1.13.6  2019-06-06  (581) 237MB classic
  1.13/candidate:   ↑
  1.13/beta:        ↑
  1.13/edge:        ↑
  1.12/stable:      v1.12.9  2019-06-06  (612) 259MB classic
  1.12/candidate:   ↑
  1.12/beta:        ↑
  1.12/edge:        ↑
  1.11/stable:      v1.11.10 2019-05-10  (557) 258MB classic
  1.11/candidate:   ↑
  1.11/beta:        ↑
  1.11/edge:        ↑
  1.10/stable:      v1.10.13 2019-04-22  (546) 222MB classic
  1.10/candidate:   ↑
  1.10/beta:        ↑
  1.10/edge:        ↑
```

ⓘ You may need to configure your firewall to allow pod-to-pod and pod-to-internet communication:

```
sudo ufw allow in on cni0 && sudo ufw allow out on cni0
sudo ufw default allow routed
```

Suggest changes ›

about 23 minutes to go

Install a local Kubernetes with MicroK8s

3. Enable addons

By default we get a barebones upstream Kubernetes. Additional services, such as dashboard, core-dns or local storage can be enabled by running the `microk8s enable` command:

```
microk8s enable dns dashboard storage
```

These addons can be disabled at anytime by running the `microk8s disable` command:

```
microk8s disable dns dashboard storage
```

With `microk8s status` you can see the list of available addons and the ones currently enabled.

List of the most important addons

- dns: Deploy DNS. This addon may be required by others, thus we recommend you always enable it.
- dashboard: Deploy kubernetes dashboard.
- storage: Create a default storage class. This storage class makes use of the hostpath-provisioner pointing to a directory on the host.
- ingress: Create an ingress controller.
- gpu: Expose GPU(s) to MicroK8s by enabling the nvidia-docker runtime and nvidia-device-plugin-daemonset. Requires NVIDIA drivers to be already installed on the host system.
- istio: Deploy the core Istio services. You can use the `microk8s istioctl` command to manage your deployments.
- registry: Deploy a docker private registry and expose it on localhost:32000. The storage addon will be enabled as part of this addon.

Suggest changes ›

about 21 minutes to go

Install a local Kubernetes with MicroK8s

4. Accessing the Kubernetes dashboard

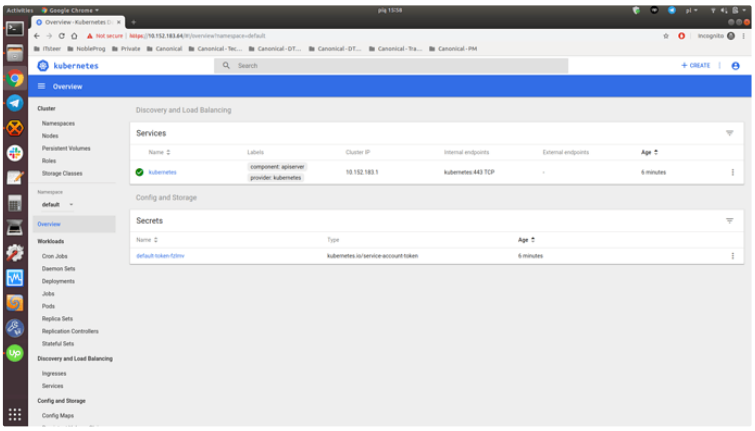
Now that we have enabled the dns and dashboard addons we can access the available dashboard. To do so we first check the deployment progress of our addons with `microk8s kubectl get all --all-namespaces`. It only takes a few minutes to get all pods in the "Running" state:

```
NAMESPACE     NAME                                         READY   STATUS    RESTARTS   AGE
kube-system   pod/calico-kube-controllers-847c8c99d-fmbsl  1/1     Running   0          92s
kube-system   pod/metrics-server-8bbfb4bdb-gwbch           1/1     Running   0          14s
kube-system   pod/dashboard-metrics-scraper-6c4568dc68-5xpbb  1/1     Running   0          14s
kube-system   pod/calico-node-sc2pv                        1/1     Running   0          92s
kube-system   pod/coredns-86f78bb79c-lfjtr                 1/1     Running   0          23s
kube-system   pod/kubernetes-dashboard-7ffd448895-7n5j2    1/1     Running   0          14s

NAMESPACE     NAME                                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                  AGE
default       service/kubernetes                  ClusterIP   10.152.183.1     <none>        443/TCP                  100s
kube-system   service/kube-dns                    ClusterIP   10.152.183.10    <none>        53/UDP,53/TCP,9153/TCP   23s
kube-system   service/metrics-server              ClusterIP   10.152.183.158   <none>        443/TCP                  15s
kube-system   service/kubernetes-dashboard        ClusterIP   10.152.183.64   <none>        443/TCP                 14s
kube-system   service/dashboard-metrics-scraper   ClusterIP   10.152.183.223   <none>        8000/TCP                 14s

NAMESPACE     NAME                         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
kube-system   daemonset.apps/calico-node   1         1         1       1            1           kubernetes.io/os=linux   95s

NAMESPACE     NAME                                       READY   UP-TO-DATE   AVAILABLE   AGE
kube-system   deployment.apps/calico-kube-controllers    1/1     1            1           95s
kube-system   deployment.apps/metrics-server             1/1     1            1           15s
kube-system   deployment.apps/dashboard-metrics-scraper  1/1     1            1           14s
kube-system   deployment.apps/coredns                    1/1     1            1           23s
kube-system   deployment.apps/kubernetes-dashboard       1/1     1            1           14s

NAMESPACE     NAME                                                 DESIRED   CURRENT   READY   AGE
kube-system   replicaset.apps/calico-kube-controllers-847c8c99d     1         1         1       93s
kube-system   replicaset.apps/metrics-server-8bbfb4bdb             1         1         1       15s
kube-system   replicaset.apps/dashboard-metrics-scraper-6c4568dc68  1         1         1       14s
kube-system   replicaset.apps/coredns-86f78bb79c                   1         1         1       23s
kube-system   replicaset.apps/kubernetes-dashboard-7ffd448895      1         1         1       14s
```

Kubernetes dashboard

As we see above the kubernetes-dashboard service in the kube-system namespace has a ClusterIP of `10.152.183.64` and listens on TCP port `443`. The ClusterIP is randomly assigned, so if you follow these steps on your host, make sure you check the IP adress you got. Point your browser to `https://10.152.183.64:443` and you will see the kubernetes dashboard UI. To access the dashboard use the default token retrieved with:

```
token=$(microk8s kubectl -n kube-system get secret | grep default-token | cut -d " " -f1)
microk8s kubectl -n kube-system describe secret $token
```



Suggest changes ›

about 13 minutes to go

Install a local Kubernetes with MicroK8s

5. Host your first service in Kubernetes

We start by creating a microbot deployment with two pods via the kubectl cli:

```
microk8s kubectl create deployment microbot --image=dontrebootme/microbot:v1
microk8s kubectl scale deployment microbot --replicas=2
```

To expose our deployment we need to create a service:

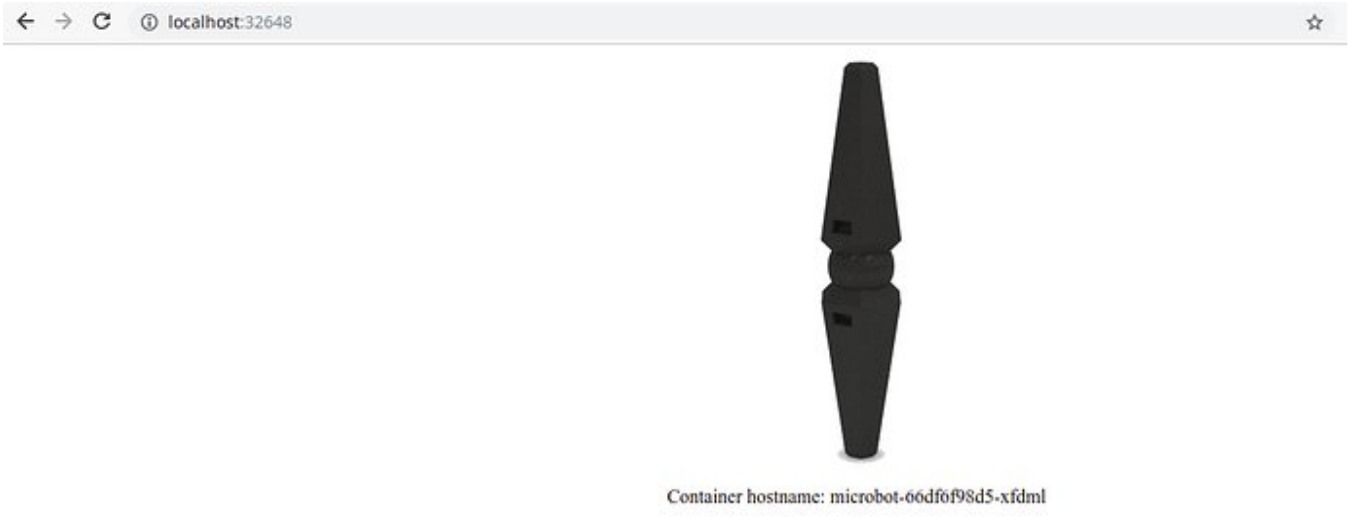
```
microk8s kubectl expose deployment microbot --type=NodePort --port=80 --name=micro
```

After a few minutes our cluster looks like this:

```
> microk8s kubectl get all --all-namespaces
NAMESPACE     NAME                                         READY   STATUS    R
kube-system   pod/calico-kube-controllers-847c8c99d-fmbsl   1/1     Running   0
kube-system   pod/metrics-server-8bbfb4bdb-gwbch           1/1     Running   0
kube-system   pod/dashboard-metrics-scraper-6c4568dc68-5xpbb   1/1     Running   0
kube-system   pod/calico-node-sc2pv                        1/1     Running   0
kube-system   pod/coredns-86f78bb79c-lfjtr                 1/1     Running   0
kube-system   pod/kubernetes-dashboard-7ffd448895-7n5j2    1/1     Running   0
default       pod/microbot-5f5499d479-vznng                1/1     Running   0
default       pod/microbot-5f5499d479-6kq5r                1/1     Running   0

NAMESPACE     NAME                                TYPE        CLUSTER-IP       EXT
default       service/kubernetes                  ClusterIP   10.152.183.1     <no
kube-system   service/kube-dns                    ClusterIP   10.152.183.10    <no
kube-system   service/metrics-server              ClusterIP   10.152.183.158   <no
kube-system   service/kubernetes-dashboard        ClusterIP   10.152.183.64    <non
kube-system   service/dashboard-metrics-scraper   ClusterIP   10.152.183.223   <no
default       service/microbot-service            NodePort    10.152.183.69    <non

NAMESPACE     NAME                         DESIRED   CURRENT   READY   UP-TO-DATE
kube-system   daemonset.apps/calico-node   1         1         1       1

NAMESPACE     NAME                                    READY   UP-TO-DATE   AVA
kube-system   deployment.apps/calico-kube-controllers   1/1     1            1
kube-system   deployment.apps/metrics-server            1/1     1            1
kube-system   deployment.apps/dashboard-metrics-scraper   1/1     1            1
kube-system   deployment.apps/coredns                   1/1     1            1
kube-system   deployment.apps/kubernetes-dashboard      1/1     1            1
default       deployment.apps/microbot                  2/2     2            2

NAMESPACE     NAME                                               DESIRED   CUR
kube-system   replicaset.apps/calico-kube-controllers-847c8c99d   1         1
kube-system   replicaset.apps/metrics-server-8bbfb4bdb           1         1
kube-system   replicaset.apps/dashboard-metrics-scraper-6c4568dc68   1         1
kube-system   replicaset.apps/coredns-86f78bb79c                 1         1
kube-system   replicaset.apps/kubernetes-dashboard-7ffd448895    1         1
default       replicaset.apps/microbot-5f5499d479                2         2
```

We have now two microbot pods and the `service/microbot-service` is the last in the services list. Our service has a ClusterIP through which we can access it. Notice, however, that our service is of type NodePort. This means that our deployment is also available on a

port on the host machine; that port is randomly selected and in this case it happens to be `32648`. All we need to do is to point our browser to `http://localhost:32648`.



Suggest changes ›

about 6 minutes to go

Install a local Kubernetes with MicroK8s

6. Integrated commands

There are many commands that ship with MicroK8s. We've only seen the essential ones in this tutorial. Explore the others at your own convenience:

- microk8s status: Provides an overview of the MicroK8s state (running / not running) as well as the set of enabled addons
- microk8s enable: Enables an addon
- microk8s disable: Disables an addon
- microk8s kubectl: Interact with kubernetes
- microk8s config: Shows the kubernetes config file
- microk8s istioctl: Interact with the istio services; needs the istio addon to be enabled
- microk8s inspect: Performs a quick inspection of the MicroK8s intallation
- microk8s reset: Resets the infrastructure to a clean state
- microk8s stop: Stops all kubernetes services
- microk8s start: Starts MicroK8s after it is being stopped

Suggest changes ›

about 1 minutes to go

Install a local Kubernetes with MicroK8s

7. That's all folks!

Congratulations! You have made it!

Until next time, stop all MicroK8s services:

```
microk8s stop
```

Where to go from here?

- Learn more about MicroK8s
- Tell us what you think and fill in feature requests
- Discover Kubernetes opportunities with Canonical
- Try Charmed Kubernetes
- Contact us

Was this tutorial useful?

Suggest changes ›

about 0 minutes to go