A CLOUD GURU

Lesson Guide - Running a Pod from a Podman-generated Kubernetes YAML File Using Kubernetes

Once you have used Podman to generate a Kubernetes YAML configuration file, you will want to use this file to run a pod. In this lesson, we will examine how to run the pod in a Kubernetes environment, using the YAML configuration file we generated with Podman. Upon completion of this lesson, you will be able to use Kubernetes to execute the pod we defined in the YAML file.

Resources

MicroK8s

How to Install MicroK8s on Red Hat Enterprise Linux

Instructions

It's time to test our YAML file!

So far, we've had great success with our `test-pod` YAML file. It was quick and easy to generate and ran great using `podman play kube`. Now it's time for the final exam. Let's take our YAML file to Kubernetes.

Will it run?

Commands Covered

- `microk8s kubectl get pod`: displays one or more resources
- `microk8s kubectl create`: creates a resource from a file or from `stdin`
- `microk8s kubectl describe pod`: shows details of a specific resource or group of resources
- `microk8s kubectl delete pods`: deletes resources by filenames, stdin, resources and names, or by resources and label selector

Run Our `nginx` Pod Using MicroK8s

We're going to use MicroK8s to stand up our `nginx` pod, using our `test-pod.yml` YAML file.

First, let's check to see if we have any Kubernetes pods:

```
microk8s kubectl get pod
```

We don't have any pods yet. Let's create one!

We'll stand up our `nginx` pod, using our `test-pod.yml` file:

```
microk8s kubectl create -f test-pod.yml
```

Checking again to see if we have any Kubernetes pods:

```
microk8s kubectl get pod
```

We see our `test-pod` pod starting.

To get more information on the details and status of our pod:

```
microk8s kubectl describe pod test-pod | more
```

Checking again to see if we have any Kubernetes pods:

```
microk8s kubectl get pod
```

We see our `test-pod` pod!

Checking to see if we can access `nginx` in our pod:

```
curl -s http://localhost:8080
```

We get the stock `nginx` index page. We can now check with a web browser, using our hostname and port `8080`.

Let's remove our `test-pod` pod:

```
microk8s kubectl delete pods --all
```

Checking again to see if we have any Kubernetes pods:

```
microk8s kubectl get pod
```

We're all cleaned up!

Great work, Cloud Guru! You just ran a pod from a YAML file using MicroK8s!

Notes

Recording - Environment used: Cloud Playground - Medium 3 unit RHEL 8 Cloud Server

Environment Setup:

Create your Cloud Playground server and log in.

Install MicroK8s

Add the EPEL Repository for RHEL 8:

```
sudo dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-
latest-8.noarch.rpm
```

```
sudo dnf -y upgrade
```

Install Snap:

```
sudo yum -y install snapd
```

Enable the `snapd.socket`:

```
sudo systemctl enable --now snapd.socket
```

Enable classic Snap support:

```
sudo ln -s /var/lib/snapd/snap /snap
```

Install MicroK8s:

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

Configure the `cloud_user` user to use MicroK8s:

```
sudo usermod -a -G microk8s cloud_user
sudo chown -f -R cloud_user ~/.kube
```

Now, you must log out and log back in.

Start MicroK8s and enable `dns` and `dashboard`:

```
microk8s start
```

```
microk8s enable dns dashboard storage
```

Create Your YAML File

Create a file named `test-pod.yml` with the contents below:

```
# Generation of Kubernetes YAML is still under development!
#
# Save the output of this file and use kubectl create -f to import
# it into Kubernetes.
#
# Created with podman-2.2.1
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2021-03-23T21:15:58Z"
  labels:
    app: test-pod
  name: test-pod
spec:
  containers:
  - command:
    - nginx
    - -g
    - daemon off;
    env:
    - name: PATH
      value: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
    - name: TERM
      value: xterm
    - name: container
      value: podman
    - name: NJS_VERSION
      value: 0.5.2
    - name: PKG_RELEASE
      value: 1~buster
    - name: NGINX_VERSION
```

```
      value: 1.19.8
    - name: HOSTNAME
      value: test-pod
    image: docker.io/library/nginx:latest
    name: test-nginx
    ports:
    - containerPort: 80
      hostPort: 8080
      protocol: TCP
    resources: {}
    securityContext:
      allowPrivilegeEscalation: true
      capabilities: {}
      privileged: false
      readOnlyRootFilesystem: false
      seLinuxOptions: {}
    workingDir: /
  restartPolicy: Always
status: {}
---
metadata:
  creationTimestamp: null
spec: {}
status:
  loadBalancer: {}
```

You're ready to go!