Troubleshooting Kubernetes Applications

29m 28s left

Open Cloud Environment
17%
Ready to login
Average setup time: 2m 26s

Open Code Environment
100%
Setup completed

Credentials

Account ID
63476765 Copy

Username
student Copy

Password
Ca1_nwSu Copy

Region
US West 2 Copy

PEM Download PPK Download

Bridge Connection
Loading...

Introduction

In this lab step, you will add a couple more tools to your troubleshooting toolbelt that can be used mainly for troubleshooting applications in Kubernetes but are useful in a variety of situations.

Instructions

1. Create a pod specification file for a simple application:

Copy code

```
cat > app.yaml <<EOF
apiVersion: v1
kind: Pod
metadata:
  name: legacy
spec:
  containers:
  - name: loop
    image: alpine:3.7
    command:
    - /bin/sh
    - -ec
    - while true; do echo hi >> /logs; sleep 2; done
EOF
```

The pod simply outputs messages to a log file every two seconds. This is simulating a legacy application that does not write log messages to standard output, which is where `kubectl logs` expects messages to be.

2. Create the application pod:

Copy code

```
kubectl create -f app.yaml
```

Support

3. View the logs seen by `kubectl`:

Copy code

output, so no logs are recorded.

4. Use the copy command to retrieve the log file from in the container:

Copy code

```
kubectl cp legacy:/logs logs
```

The syntax specifies the source first and the destination second. The path that begins with a pod name is the path in the container. The `cp` command can copy files and directories to and from the host and container.

5. View the contents of the copied log file:

Copy code

```
more logs
```

```
hi
hi
hi
hi
hi
```

Alternative ways to get at the logs are to use `exec` to run commands to view the logs from inside the container, or to use a volume to have the log file mounted on the host file system.

6. Create another pod specification for a pod that will consume a lot of CPU resources:

Copy code

```
cat > load.yaml <<EOF
apiVersion: v1
kind: Pod
metadata:
  name: load
spec:
  containers:
  - name: cpu-load
    image: vish/stress
    args:
    - -cpus
    - "2"
EOF
```

The `stress` image runs a binary that can consume a varying amount of

e two CPUs,

Copy code

```
kubectl create -f load.yaml
```

The pod simulates a situation where you notice degraded performance in the cluster. When there are many pods and nodes in the cluster, it is difficult to determine which pod or pods are causing the degradation by inspecting each node and pod individually.

8. Display the help document for the `top` command:

Copy code

```
kubectl top --help
```

The `top` command is similar to the native Linux `top` command for measuring CPU and memory usage of processes. You can monitor at the node or pod level.

9. List and watch the resource consumption of pods:

Copy code

```
watch kubectl top pods
```

```
NAME      CPU(cores)   MEMORY(bytes)
legacy    1m           0Mi
load      1943m        1Mi
```

The **load** pod is using nearly one full core (**1943m**illi-cores). Having such an unrestricted pod in your cluster could wreak havoc.

Note: It can take a minute for the new pod to appear in the list.

Press *ctrl+c* to stop watching the pods.

10. Create a similar pod specification except with a CPU resource limit and request for half of a CPU:

Copy code

```
cat > load-limited.yaml <<EOF
apiVersion: v1
kind: Pod
```

```
      limits:
        cpu: "0.5"
      requests:
        cpu: "0.5"
EOF
```

The `resources` key is added to specify the limit and request.

11. Create the pod with the CPU limit:

Copy code

```
kubectl create -f load-limited.yaml
```

12. Wait a minute for the new pod's metrics to be collected, and then display the pod usage:

Copy code

```
watch kubectl top pods
```

```
NAME           CPU(cores)   MEMORY(bytes)
legacy         1m           1Mi
load           1257m        1Mi
load-limited   499m         1Mi
```

Using requests and limits for CPU and memory can prevent performance issues, and allow the scheduler to make the best use of the cluster's resources.

Press *ctrl+c* to stop watching the pods.

Summary

In this lab step, you learned how to copy files to and from pod containers, which can be useful when working with legacy applications. You also saw how to diagnose performance degradations, and how to set resource limits to bring the cluster's consumption back to within an expected range.

VALIDATION CHECKS

1 Checks Start check