Get a Shell to a Running Container

This page shows how to use `kubectl exec` to get a shell to a running container.

Before you begin

You need to have a Kubernetes cluster, and the kubectl command-line tool must be configured to communicate with your cluster. It is recommended to run this tutorial on a cluster with at least two nodes that are not acting as control plane hosts. If you do not already have a cluster, you can create one by using minikube or you can use one of these Kubernetes playgrounds:

- Killercoda
- Play with Kubernetes

Getting a shell to a container

In this exercise, you create a Pod that has one container. The container runs the nginx image. Here is the configuration file for the Pod:

application/shell-demo.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: shell-demo
spec:
  volumes:
  - name: shared-data
    emptyDir: {}
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - name: shared-data
      mountPath: /usr/share/nginx/html
  hostNetwork: true
  dnsPolicy: Default
```

Create the Pod:

```
kubectl apply -f https://k8s.io/examples/application/shell-demo.yaml
```

Verify that the container is running:

```
kubectl get pod shell-demo
```

Get a shell to the running container:

```
kubectl exec --stdin --tty shell-demo -- /bin/bash
```

> **Note:** The double dash (`--`) separates the arguments you want to pass to the command from the kubectl arguments.

In your shell, list the root directory:

```
# Run this inside the container
ls /
```

In your shell, experiment with other commands. Here are some examples:

```
# You can run these example commands inside the container
ls /
cat /proc/mounts
cat /proc/1/maps
apt-get update
apt-get install -y tcpdump
tcpdump
apt-get install -y lsof
lsof
apt-get install -y procps
ps aux
ps aux | grep nginx
```

Writing the root page for nginx

Look again at the configuration file for your Pod. The Pod has an `emptyDir` volume, and the container mounts the volume at `/usr/share/nginx/html`.

In your shell, create an `index.html` file in the `/usr/share/nginx/html` directory:

```
# Run this inside the container
echo 'Hello shell demo' > /usr/share/nginx/html/index.html
```

In your shell, send a GET request to the nginx server:

```
# Run this in the shell inside your container
apt-get update
apt-get install curl
curl http://localhost/
```

The output shows the text that you wrote to the `index.html` file:

```
Hello shell demo
```

When you are finished with your shell, enter `exit`.

```
exit # To quit the shell in the container
```

Running individual commands in a container

In an ordinary command window, not your shell, list the environment variables in the running container:

```
kubectl exec shell-demo -- env
```

Experiment with running other commands. Here are some examples:

```
kubectl exec shell-demo -- ps aux
kubectl exec shell-demo -- ls /
kubectl exec shell-demo -- cat /proc/1/mounts
```

Opening a shell when a Pod has more than one container

If a Pod has more than one container, use `--container` or `-c` to specify a container in the `kubectl exec` command. For example, suppose you have a Pod named my-pod, and the Pod has two containers named *main-app* and *helper-app*. The following command would open a shell to the *main-app* container.

```
kubectl exec -i -t my-pod --container main-app -- /bin/bash
```

> **Note:** The short options `-i` and `-t` are the same as the long options `--stdin` and `--tty`

What's next

- Read about kubectl exec

Feedback

Was this page helpful?

Yes No

Last modified September 19, 2023 at 11:12 PM PST: fix: update deprecated command (448734c716)