## Overview

In this lab, you will use the kubectl command to make requests to the kubernetes API in order to run a simple application. You will then explore other basic capabilities of the kubectl command line.

* Run an application in Kubernetes
* Explore the kubectl CLI

[Overview](#_scot2uasg1t0)

[Instructions](#_8ehy52bbxj4o)

[Task 0: Run a simple app](#_q1eipr6h6290)

[Step 1: Explore the cli](#_z9ad9pzgs0f1)

[Step 2: Run an application](#_guqiq3ja6mzy)

[Step 3: Test the application](#_k687ds5gpm0q)

[Task 1: Scale-out the app](#_c0i533prk1n7)

[Step 1: Create 3 replicas](#_y2xjad3y77zo)

[Step 2: View resource usage](#_cf9082du9zm0)

[Task 2: Explore Kubectl](#_y77r1wwp2fyh)

[Step 1: View events](#_ui2lowwf6laf)

[Step 2: Poke around the container](#_8sigualwep4z)

[Task 3: Cleanup](#_o6p73anl9fq5)

[Step 1: Delete all resources](#_7pk4qwyoc2h0)

## 

## Instructions

Read this lab like a book, all text is there for a reason!

"→" denotes an action you must take

Use your favorite editor to edit files within the console. I suggest VI, nano, or emacs.

|  |
| --- |
| White boxes with black text denote commands and file contents |

|  |
| --- |
| Black boxes with green text denote example output |

### **Task 0: Run a simple app**

#### **Step 1: Explore the cli**

→ View the kubectl commands and options:

|  |
| --- |
| kubectl -h |

Many of these options we will revisit in later labs. For this lab, we are going to make use of the 'run', 'scale', 'logs', and 'exec' commands.

→ View global options for kubectl:

|  |
| --- |
| kubectl options |

You can also check out this cheatsheet of commands: <https://kubernetes.io/docs/reference/kubectl/cheatsheet/>

→ List all resources:

|  |
| --- |
| kubectl get all |

|  |
| --- |
| No resources found. |

We don't have any resources yet!

#### **Step 2: Run an application**

Kubectl run is similar to docker run, except at the whole cluster level instead of the individual node level.

→ View the help for the 'run' command:

|  |
| --- |
| kubectl run -h |

Notice there are a lot of examples and options! The run command is a convenient way to quickly run an application.

Unfortunately, in Kubernetes 1.18, we cannot use “run” any longer to create a “Deployment”, which will let us scale an application. Instead, we need to create a deployment manually.

→ Lets create one now:

|  |
| --- |
| kubectl create deployment nginx --image=nginx |

|  |
| --- |
| deployment.apps/nginx created |

This command created a Deployment object and a single Pod. We will learn more about Deployments later.

→ Now list the pods:

|  |
| --- |
| kubectl get pods |

|  |
| --- |
| NAME READY STATUS RESTARTS AGE  nginx-dbddb74b8-svt5z 1/1 Running 0 51s |

#### **Step 3: Test the application**

→ Let's get a little more information, add the '-o wide' option.

|  |
| --- |
| kubectl get pods -o wide |

|  |
| --- |
| NAME READY STATUS RESTARTS AGE IP NODE  NOMINATED NODE  nginx-dbddb74b8-svt5z 1/1 Running 0 64s 10.8.1.199 gke-standard-cluster-1-default-pool-1b3ee6a  7-sbq8 <none> |

This shows us information such as the pod IP and what node the pod is running on. We can get even more information with the 'describe' command. Copy the pod's name from the previous command output and run the following.

→ Replace <POD\_NAME>

|  |
| --- |
| kubectl describe pod <POD\_NAME> |

|  |
| --- |
| Name: nginx-dbddb74b8-svt5z  Namespace: ameade  Priority: 0  PriorityClassName: <none>  Node: gke-standard-cluster-1-default-pool-1b3ee6a7-sbq8/10.128.0.4  Start Time: Sat, 04 May 2019 20:32:56 +0000  Labels: pod-template-hash=dbddb74b8  run=nginx  ... |

→ Lets test the app by using curl, replace <IP> with the pod ip from the previous command

|  |
| --- |
| curl <IP> |

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>Welcome to nginx!</title>  <style>  body {  width: 35em;  margin: 0 auto;  font-family: Tahoma, Verdana, Arial, sans-serif;  }  </style>  </head>  <body>  <h1>Welcome to nginx!</h1>  <p>If you see this page, the nginx web server is successfully installed and  working. Further configuration is required.</p>  <p>For online documentation and support please refer to  <a href="http://nginx.org/">nginx.org</a>.<br/>  Commercial support is available at  <a href="http://nginx.com/">nginx.com</a>.</p>  <p><em>Thank you for using nginx.</em></p>  </body>  </html> |

→ We can see in the application logs that the app received our request.

|  |
| --- |
| kubectl logs <POD\_NAME> |

|  |
| --- |
| 10.142.0.20 - - [04/May/2019:20:35:19 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.52.1" "-" |

### **Task 1: Scale-out the app**

#### **Step 1: Create 3 replicas**

→ scale up

|  |
| --- |
| kubectl scale deployment nginx --replicas=3 |

|  |
| --- |
| deployment.extensions/nginx scaled |

→ List the pods.

|  |
| --- |
| kubectl get pods -o wide |

|  |
| --- |
| NAME READY STATUS RESTARTS AGE IP NODE  NOMINATED NODE  nginx-dbddb74b8-456sj 1/1 Running 0 8s 10.8.1.201 gke-standard-cluster-1-default-pool-1b3ee6  a7-sbq8 <none>  nginx-dbddb74b8-btmmh 1/1 Running 0 8s 10.8.1.200 gke-standard-cluster-2-default-pool-1b3ee6  gd-feg2 <none>  nginx-dbddb74b8-svt5z 1/1 Running 0 4m4s 10.8.1.199 gke-standard-cluster-3-default-pool-1b3ee6  bb-cfc9 <none> |

Notice how the pods are likely running on different nodes. The default behavior for kubernetes is to spread identical pods out across nodes. This improves availability in case a node is lost.

#### **Step 2: View resource usage**

→ Let's check out the resource usage of our pods:

|  |
| --- |
| kubectl top pod |

|  |
| --- |
| NAME CPU(cores) MEMORY(bytes)  nginx-dbddb74b8-456sj 0m 1Mi  nginx-dbddb74b8-btmmh 0m 1Mi  nginx-dbddb74b8-svt5z 0m 1Mi |

They are not using a lot of resources, so we likely didn't need to scale out, but that is okay because we didn't need to reserve or allocate any resources. If the containers don't use the resources, then any other containers on the cluster can.

### **Task 2: Explore Kubectl**

Lets check out some of the other kubectl commands.

#### **Step 1: View events**

You can see all the actions taken by kubernetes by listing events. This can be very helpful for debugging why a container is crashing or won't start at all.

→ list events

|  |
| --- |
| kubectl get events |

|  |
| --- |
| LAST SEEN TYPE REASON KIND MESSAGE  4m56s Normal Pulled Pod Successfully pulled image "nginx"  4m55s Normal Created Pod Created container  4m55s Normal Started Pod Started container  4m58s Normal SuccessfulCreate ReplicaSet Created pod: nginx-dbddb74b8-svt5z  62s Normal SuccessfulCreate ReplicaSet Created pod: nginx-dbddb74b8-456sj  62s Normal SuccessfulCreate ReplicaSet Created pod: nginx-dbddb74b8-btmmh  4m58s Normal ScalingReplicaSet Deployment Scaled up replica set nginx-dbddb74b8 to 1  63s Normal ScalingReplicaSet Deployment Scaled up replica set nginx-dbddb74b8 to 3 |

#### **Step 2: Poke around the container**

We can run commands from within our container with the 'exec' command.

Lets print the process id of our nginx process.

→ Run command in pod

|  |
| --- |
| kubectl exec <POD\_NAME> -- cat /run/nginx.pid |

|  |
| --- |
| 1 |

It's got a PID of 1. This is because our container is in its own Linux PID namespace and can only see our nginx process.

For further debugging a container, we can actually 'attach' to a container as if we were in the same isolation as our application. We do this by running a shell command with the '-it' flags.

→ open a shell

|  |
| --- |
| kubectl exec -it <POD\_NAME> /bin/bash |

|  |
| --- |
| root@nginx-dbddb74b8-456sj:/# |

We are now 'in' the container. There are limited commands here because we do not have a lot installed within the nginx image.

→ Exit the container

|  |
| --- |
| exit |

### **Task 3: Cleanup**

#### **Step 1: Delete all resources**

→

|  |
| --- |
| kubectl delete all --all |