Introduction to Kubernetes Workshop Exercises

Students will have access to a small Kubernetes cluster on GKE, with connection instructions given to you on the day.

You will only have kubectl access to it, not SSH, and therefore the exercises below that relate to Docker and systemd will be marked with “DEMONSTRATION” and will be demonstrated by the instructor.

1. What is a Container? [DEMONSTRATION]

a. Containers are “just” processes

   i. Run a container for our example

      ```
      docker container run -d nginx
      ```

   ii. Look at the operating system process tree

      ```
      ps faux
      ```

   iii. Look for `/usr/bin/dockerd` and observe the process tree under it. What is going on here? What are the chain of processes above the (expected) two nginx processes?

b. Containers are processes with some isolation

   i. List the namespaces related to our container (requires modern systemd)

      ```
      sudo lsns | grep nginx
      ```

   ii. Note that the pid column is the parent of the two nginx processes. After Docker sets up the container’s namespaces, they are inherited by child processes

   iii. Note that most processes don’t have namespace output in this tool. Most processes don’t use namespaces. But containers do!

   iv. List the cgroups on the system

```
systemd-cgls
```

v. Note that cgroups are arranged in a hierarchy; cgroups, like namespaces are inherited. cgroups are divisions of system resources and are called “slices”. We won’t go into detail about this now

2. The Kubernetes API

a. Install kubectl if you don’t have it already

https://kubernetes.io/docs/tasks/tools/install-kubectl/

b. Configure your kubectl client to connect to your provided cloud cluster

Fill in the form here to get access to your k8s cluster:

https://goo.gl/forms/BQpDqYFa2R4nUgu73

Once you have it downloaded, set up an alias to use that config file:

```
alias kubectl='kubectl --kubeconfig=/path/to/kubeconfig/yml'
```

Note that you will need to run this command again if you open another terminal

c. View and inspect a running workload

i. Run some workloads for us to interact with and inspect

```
kubectl create -f exercise-2b.yaml
```

ii. List pods

```
kubectl get pods
```

iii. View logs of a pod

```
kubectl logs demo-api-****
```

iv. View more details about a pod

```
kubectl describe pod demo-api-****
```

v. Kill a pod and watch Kubernetes reschedule it

```
kubectl delete pod demo-api-****
```

```
kubectl get pods
```

vi. Observe the pod getting rescheduled. Think about how life might have been as a sysadmin before container schedulers!

3. Container Basics with Docker [DEMONSTRATION]

a. Interacting with containers and images with Docker

i. Working with containers

1. Run a container

```
docker container run alpine “hello docker”
```

2. List running containers

```
docker container ls
```

3. List stopped containers

```
docker container ls -a
```

4. If a container is just a process, what is a stopped container?

5. Remove a container

```
docker container rm <INSERT ID>
```

```
docker container run -d alpine ping 8.8.8.8
```

```
docker container rm <NEW CONTAINER ID>
```

6. Note that you can’t delete a running container (without forcing it)

7. Stop the container

```
docker container stop <NEW CONTAINER ID>
```

8. Note that you can still view the logs; Docker caches them

```
docker container logs <NEW CONTAINER ID>
```

9. Remove the stopped container

```
docker container rm <NEW CONTAINER ID>
```

10. View containers

```
docker container ls -a
```

ii. Working with images

1. List images

```
docker image ls
```

2. Pull an image

```
docker pull lukebond/demo-api
```

3. Run a container from an image you pulled and expose the port

```
docker container run -d -e \
  MESSAGE="Hello from Docker" \
  -p 9000:9000 lukebond/demo-api
```

4. Access your service

```
curl localhost:9000
```

b. Building Docker images

i. Create a Dockerfile based on nginx that adds an html file. The image's documentation will prove useful: https://hub.docker.com/_/nginx/

ii. Build the image, tag it and launch a container from it, hitting the exposed port to test it

4. Kubernetes Networking [DEMONSTRATION]

a. SSH into one of your nodes that has a pod running on it and inspect the network interfaces

   `ip addr`

   Note the Docker bridge, and the veth* entries towards the bottom- the IP address, the interface number, and the number of the interface to which it's connected (should be one number apart). Compare the IP address with that of the pod via `kubectl`.

b. Get a terminal into the container using docker exec, and inspect the interfaces in there:

   `docker exec -it <CONTAINER ID> bash`

   `ip addr`

   Notice the IP address of the pod, the interface number, and the number of the interface to which it's connected. What is happening here?

c. Create exercise-4d.yaml, a single pod with two containers in it. Inspect the namespaces on the host, and the network interfaces inside and outside the container. What do you notice about it?

5. Kubernetes Core Components

a. Clone the following repository locally to get the YAML files you'll need for this exercise

   `git clone https://github.com/controlplaneio`

b. Create a pod from exercise-5-1.yaml

   `kubectl create -f exercise-5-1.yaml`

c. List the running pods, waiting for the nginx pod to complete startup

```
kubectl get pods
```

d. Fetch the logs of the nginx pod

```
kubectl logs nginx
```

e. Get more detail about the running pod

```
kubectl describe pod nginx
```

f. Delete the pod

```
kubectl delete pod
```

g. Notice that it doesn't get re-created

```
kubectl get pods
```

h. Create some pods, this time with a ReplicaSet

```
kubectl create -f exercise-5-2.yaml
```

i. List the running pods, waiting for all pods to complete startup

```
kubectl get pods
```

j. Delete one of the pods

```
kubectl delete pod nginx-1
```

k. Notice that it gets recreated

```
kubectl get pods
```

l. Scale the replica set and watch the number of pods change

```
kubectl scale --replicas=1
```

m. Delete the deployment to make space

```
kubectl delete -f exercise-5-2.yaml
```

n. Write a deployment YAML file that creates a replicaset of three instances of "lukebond/demo-api-redis", one instance of "redis" and

a service for demo-api-redis. Use “kubectl port-forward” to forward a local port to the service and curl the results.

6. Stretch Goal Exercises

a. https://www.katacoda.com/courses/kubernetes/guestbook

b. https://www.katacoda.com/courses/kubernetes/liveness-readiness-healthchecks

c. https://www.katacoda.com/courses/kubernetes/managing-secrets

d. https://www.katacoda.com/courses/kubernetes/storage-introduction