# Lab 5 – Kubernetes

## 1 - Creating a cluster and deploying containers

### Cluster Creation

Text

Description automatically generated

This took a few minutes. This command created a cluster for us to deploy containers in. We did leave some versioning flags and num-containers flags but I would assume the defaults are fine. From my knowledge of Kubernetes I would assume that we created a config file and passed it off to the Master Kubernetes API server to be used as the ‘default settings’ for the creation of our pods on the worker nodes.

### Pulling from cloud repo



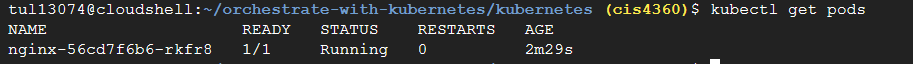
We used gsutil cp -r to copy files from a cloud repo to our machine. Here are the contents of that copied folder.

### K8s deploy



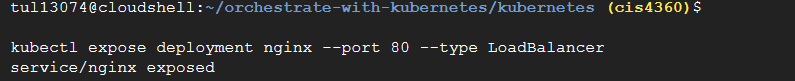
We deploy an nginx container using create deployment. This is likely to solve the issue of the short lifetime of pods. If a pod dies in this mode **it** will be restarted.

### K8s ‘ps’



Much like docker ps, we see all active pods on our process here.

### K8s port exposure and services



We have exposed the **service** on a port 80. It is important that we have exposed a **service** and not a pod. Since a pod has a short lifespan, its IP will change frequently. Thus, we use services that stand in front of a pod regardless how many times it may die. This allows us to attach a hard IP to the pod. Moreover services contain LoadBalancers as shown above.

### View Service IP and curl

Text

Description automatically generated

As I said above we gain the IP of the service and fetch what’s on the server via curl. It is a static HTML page supplied by the image we pulled from gcloud repo via gsutil cp.

## 2 - Pods

### Viewing a Monolith Configuration

Text

Description automatically generated

This is a true monolith since we have one container for one *Application.* Moreover, we can expose ports and limit resources for the container here.

### Creation from YAML and Pod description Text Description automatically generated

We create a pod via passing the config we saw earlier to the master Kubernetes node. The -f stands for FILENAME or path in this case.

We then ask the k8s process to show us active services to view our successful creation. We also ask for a description of the pod. The docs for this command state

“This command joins many API calls together to form a detailed description of a given resource or group of resources.”

I would assume that this would call on the etcd key-value storage for specific pod information.

## 3 - Interacting with Pods

### Port Forwarding & more curl

Text

Description automatically generated 

We port-forward to allow computers on other networks to discover our service that says “Hello”.

### Attempt to reach secure endpoint

Text

Description automatically generated

We fail to reach the secure endpoint meant for monitoring the site because we do not possess any type of valid “ID”. Thus we use our *password* and procure a token to log in and view that our pod can also handle secure exposures.

### Logging

Text

Description automatically generated

We specify the pod name and not the container name. This is because we only have single container active (monolith).

The port was accessed by 3 different machines as shown by their unique IP addresses.

### Interacting **directly with** a pod

A computer screen capture

Description automatically generated with medium confidence

Here I interact with a pod using bash after attaching my STDIN as my GCP terminal (which is really just a virtual computer). We have to tell the engine to specifically interact with the monolith container.

## 4 - Services

### Creation

#### Viewing **secure** pod configuration

Text

Description automatically generated

We have already used this command before on the monolith.yaml

#### Creating a secure pod

A screenshot of a computer

Description automatically generated with medium confidence

We notice the secure pod is now online. We need to create a services to expose its function to the outside world. Here is the service configuration:

Text

Description automatically generated

I had to use vim since cat was ignoring the new lines.  
The metadata allows it to know which pods to bind to, in this case we have the name and the label of the pod the same value to bind them.

#### K8S create

Text

Description automatically generated

After creating his service we notice that kubectl create is a all-in-one type command. It can handle the creation of pods and services at this point given a correct config file.

### Allow Node access

#### Firewall Creation

Text

Description automatically generated

The creation of our firewall can allow our service to direct traffic in/out.

#### List of active Machines

Text

Description automatically generated

#### curl Secure Pod

Graphical user interface, text

Description automatically generated

I stopped the connection because it took way too long.

#### Question

Graphical user interface, text, application, email

Description automatically generated

$ kubectl describe services monolith

Text

Description automatically generated with low confidence

We notice that we have no label on his pod and we know that services are bounded to pods via a lab. Thus, we hit the service but the service is not attached to a pod and it does not respond anything back to us.

This idea is further supported by this:

A screenshot of a computer

Description automatically generated

The service exists for a label of “monolith” with secure enabled, yet none exist. -l is label tag for the label query string.

#### Labeling our Secure pod with `secure-enabled`

A screenshot of a computer

Description automatically generated

Now that we have labeled our secure pod, we can attach our service. Given that the pod has this tag its likely that the service has already discovered this pod.

#### Endpoints



We used the *describe* command to get all data on our pod and viewed its endpoints. Now it has one.

#### Retrial of hitting an endpoint



Now the message is returned to us.

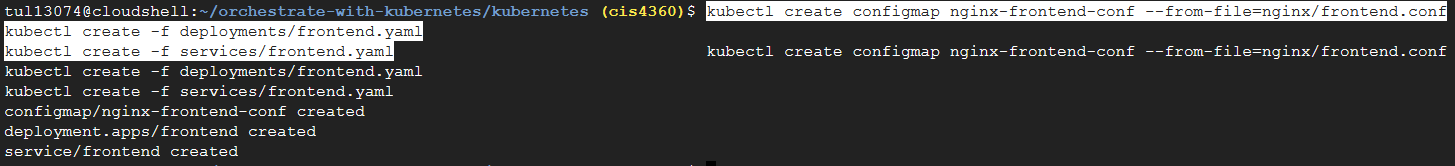
## 5 – Deployment

### Creating Auth



Used && to chain the creation of a deployment object and service. We note from looking at the deployments/auth.yaml and seeing kind: deployment that deployment is a type of Kubernetes object.

### Creating Frontend and its services



The frontend service is important since it will be directing traffic.

### curl hello

A screenshot of a computer

Description automatically generated with medium confidence

This last task was a great example of how unstable these pods can be. I tried many times to curl, and it would error out. I investigated the running pods and saw that the frontend pod had been restarted 7 times in 10 mins.