Lab 3 - Delivery
Software Quality

Okikioluwa Ojo (100790236)

Date: Feb 28, 2024

GitHub & Video Link: https://github.com/okikio-school/SOFE3980U-Lab3

I don’t have a credit card, so I can’t actually use GCP, I had already communicated with the Prof. and had sent you an email with Prof. CC’d.

Docker is a containerization software that lets users replicate an exact replica of an environment cheaply and reliably.

A container is a running isolated environment where you can run software, containers run off a shared operating system and thus provide process and file system isolation but cannot replicate hardware virtualization, which is why containers are so cheap resource wise. An image is a pre-built container in the non-running state ready to be deployed and a registry is basically a store of images that one can pull from and use as a container. Docker is very lightweight compared to Virtual Machines (VMs) making it very cheap to run, but it’s also very portable so anyone can run the built isolated environment almost everywhere, a con though is that it’s not quite as secure as VMs as it’s running at a higher level of abstraction than VMs, by default containers use the host kernel available so they are vulnerable to security threats that can take advantage of that, but VMs run their own independent kernel making their isolation more complete but also more resource intensive.

Kubernetes (K8s) is a cloud orchestrator that lets applications scale to match the needs of customers, it does this using Containers, Pods, Services, and Deployments. Basically Kuberenetes lets you as a developer define how you want your cloud application to scale and it will handle the physical details. The core aspects of K8s are Nodes, and Clusters. A node represents an atomic computing unit (basically a computer or a server), a cluster is a collection of nodes working together, basically kubernetes scales by using all the nodes it has available in the cluster to match customer demand for scale. Pods contain multiple containers and also play a role as part of K8s when scaling, so basically you can deploy pods and based off of your K8s Deployment and ReplicaSets, as usage grows more Pods will also be deployed An example of one of the reasons Pods support multiple containers are for use cases like a Spring-Boot Server and a MySQL Database back-to-back as containers within a Pod (technically you would want your database to be a standalone Pod but hopefully you get the point), as the pod scales more Pods will be deployed on various nodes.

An interesting detail about Pods is that they're hot swappable, so you can actually deploy a new version of a Pod and it'll keep the old version until the criteria you've set for it at that point in time, the new version will then take over. Deployments are configuration you use to deploy new pods, you can specify the image you want for the pod, the size you want the Pod to scale up to, as well as what tolerations the Deployment has which helps identify and match with the correct Nodes that have been tainted to run the Pods, etc…

A Service is a networking primitive for K8s and lets you control routing, acts as a LoadBalancer, and enable discovery within a K8s cluster, e.g. you can choose how you want your deployment to be exposed which would define how a pod is accessible from inside and outside the cluster. Note: each cluster is basically treated as an independent computer from the rest and within each cluster all nodes are treated as if they're all connected as one giant computer.

For Lab 3, I ran the Lab on K3D as I didn't have a credit card to run GCP.

To actually run the program you would need to first install K3D and Docker Desktop

Then you need to create your cluster (Note: I'm assuming you're running this on WSL or directly on Linux)

```
k3d cluster delete multiserver &&
k3d cluster create multiserver
  --agents 1
  -p "30000-30100:30000-30100@agent:0"
  -p "4306:3306@agent:0"
  -p "8080:8080@agent:0"
  --k3s-arg '--service-node-port-range=30000-30100
@server:*'
```

You then build the image using this command

```
mvn clean package
docker build -t <path-to-repo>/binarycalculator --no-cache .
docker push <path-to-repo>/binarycalculator
```

> When I ran the command I replaced `<path-to-repo>/binarycalculator` with `okikioschool/binarycalculator`

To apply the image you then run

```
kubectl apply -f .
```

The IP to access the BinaryCalculator is

```
localhost:8080
```

The IP to access the MySQL server

```
localhost:4306
```