**PRACTICAL: 7**

**AIM:**

Kubernetes in Google Cloud: Kubernetes is the most popular container orchestration system and the Google Kubernetes Engine was designed specifically to support managed Kubernetes deployments in the Google Cloud. In this advanced-level quest, you will get hands-on practice configuring Docker images and containers, and deploying fully-fledged Kubernetes Engine applications. This quest lab will teach you the practical skills needed for integrating container orchestration into your own workflow.

**THEORY:**

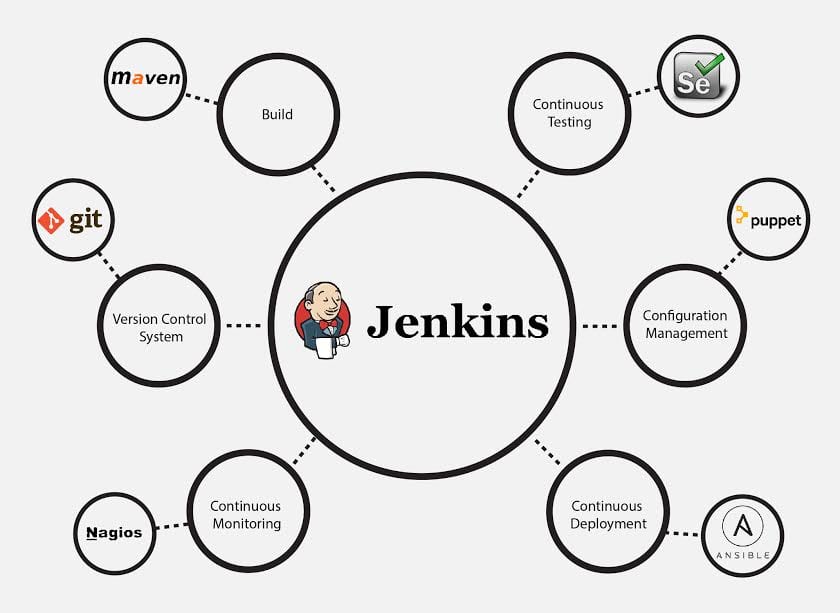
Docker is an open platform for developing, shipping, and running applications. With Docker, you can separate your applications from your infrastructure and treat your infrastructure like a managed application. Docker helps you ship code faster, test faster, deploy faster, and shorten the cycle between writing code and running code.

Docker does this by combining kernel containerization features with workflows and tooling that helps you manage and deploy your applications. Docker containers can be directly used in Kubernetes, which allows them to be run in the Kubernetes Engine with ease. After learning the essentials of Docker, you will have the skillset to start developing Kubernetes and containerized applications.

Kubernetes is an open source project (available on kubernetes.io) which can run on many different environments, from laptops to high-availability multi-node clusters, from public clouds to on-premise deployments, from virtual machines to bare metal.

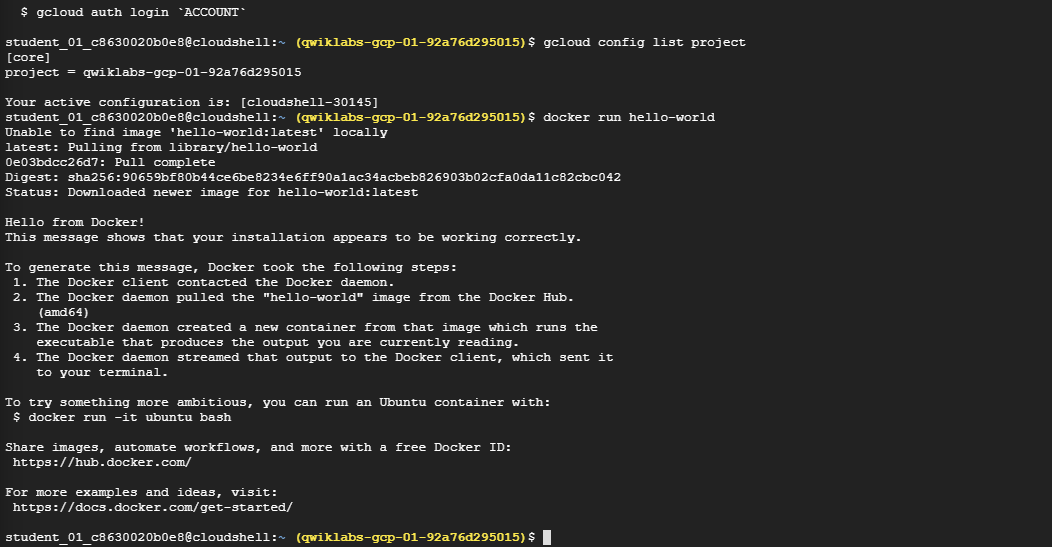
Dev Ops practices will regularly make use of multiple deployments to manage application deployment scenarios such as "Continuous Deployment", "Blue-Green Deployments", "Canary Deployments" and more.

Jenkins is a free and open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat.

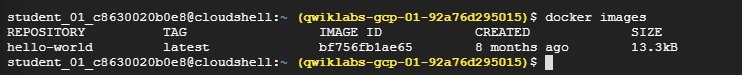


**OUTPUT:**

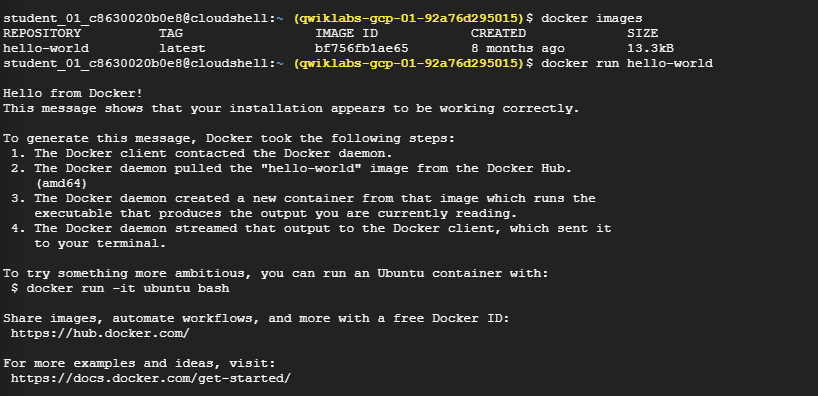
**7.1 - Introduction to Docker**



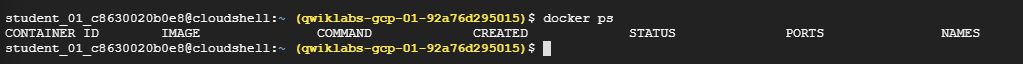
**Running docker hello-world**



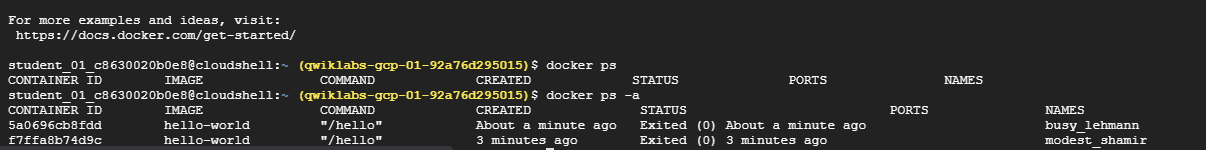
**Docker images**



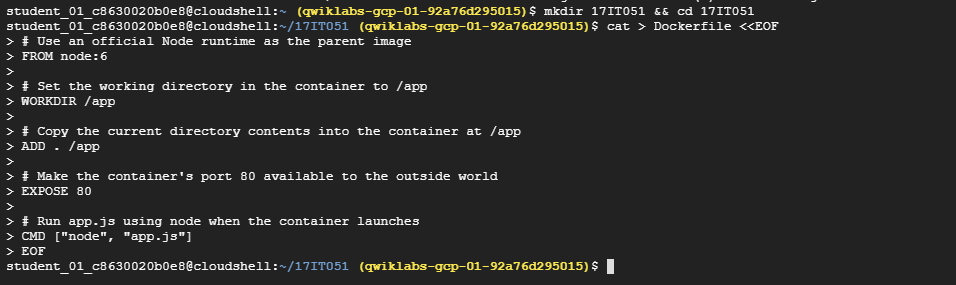
**Docker run hello-world again**



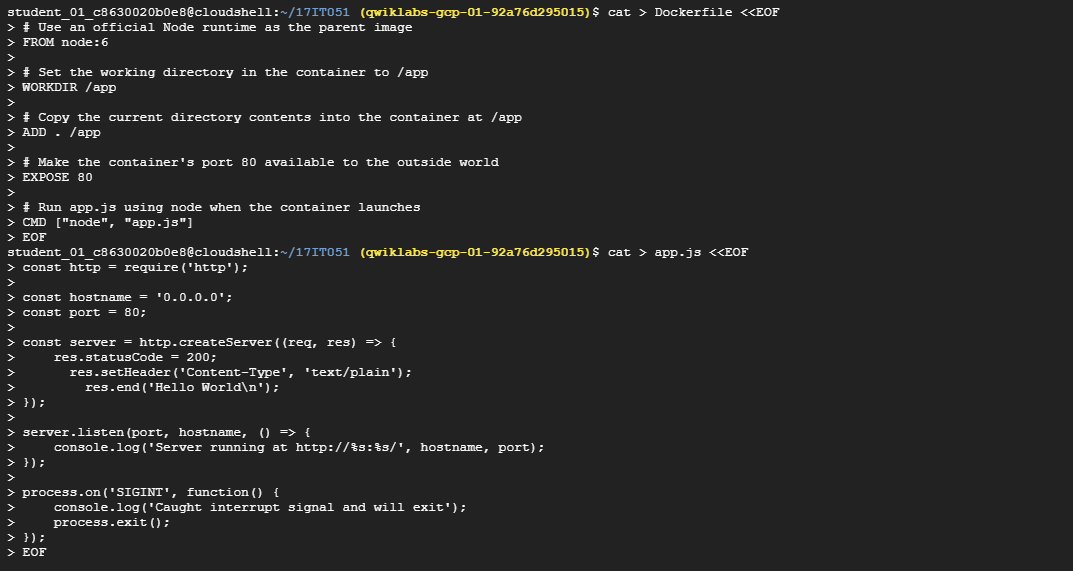
**Listing docker process**



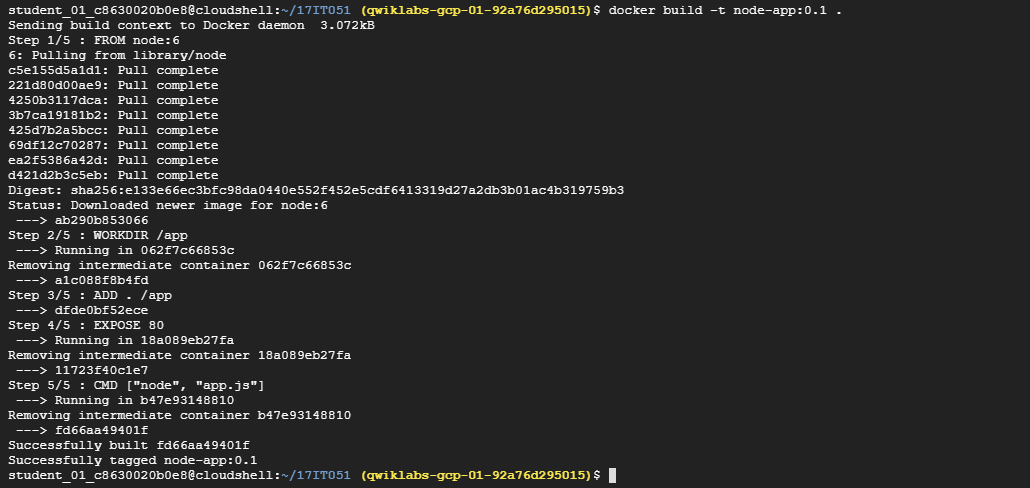
**Listing docker process with all flag**



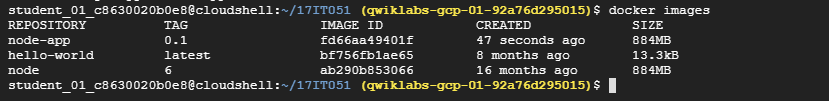
**Creating directory and navigation into it**



**Viewing docker file**



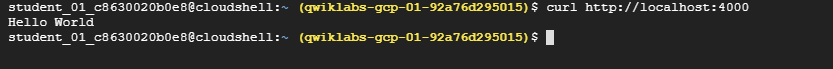
**Building docker**



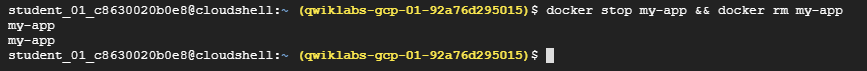
**Visualize docker image**



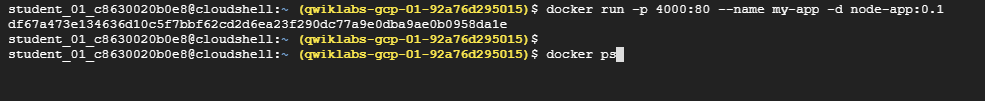
**Running docker**



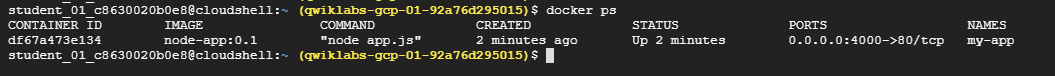
**Curl to IP on which docker activated**



**Stopping and removing my-app**



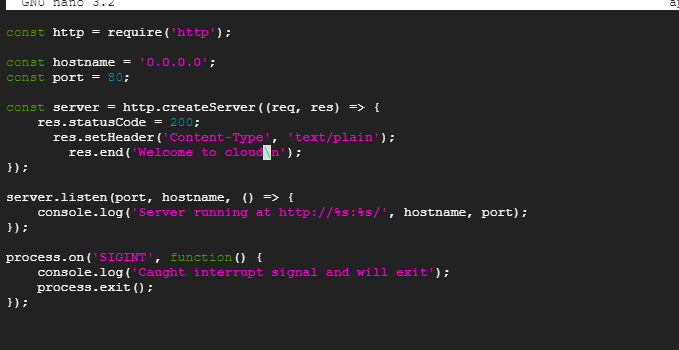
**Running docker again**



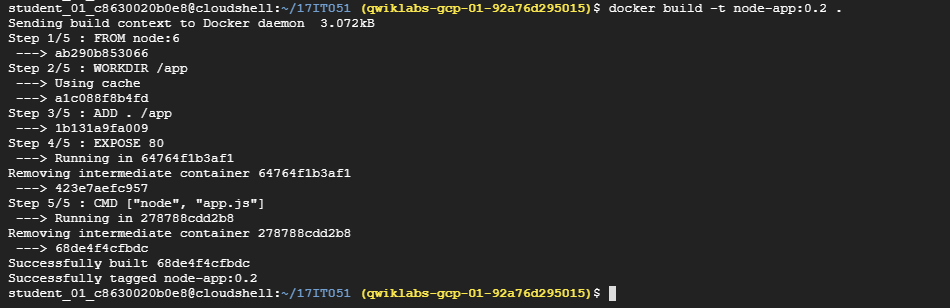
**Visualize docker process**



**Docker logs**



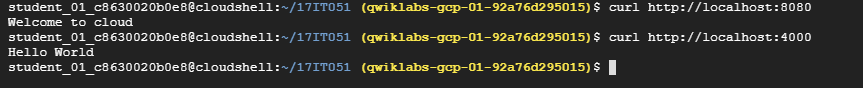
**Node server**



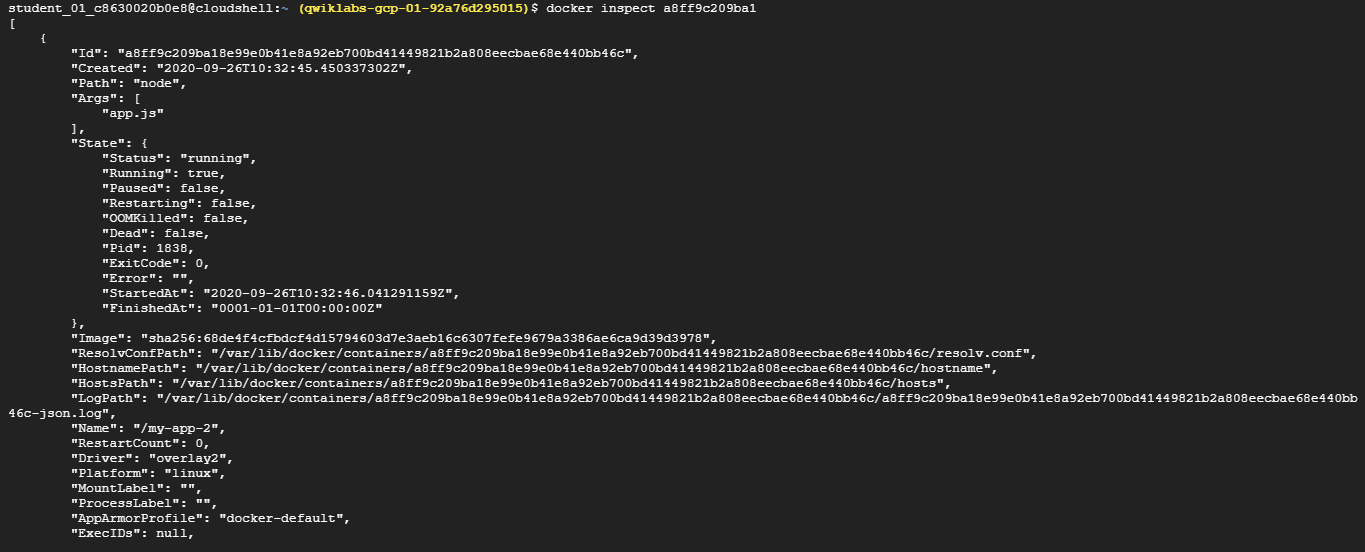
**Building docker**



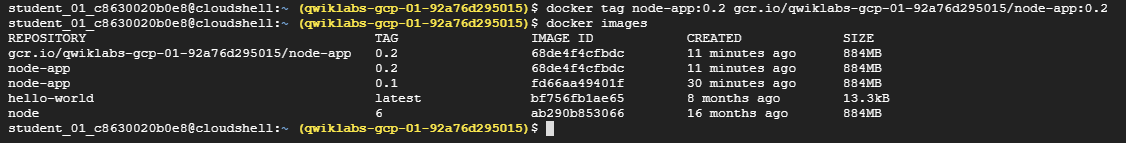
**Running docker again**



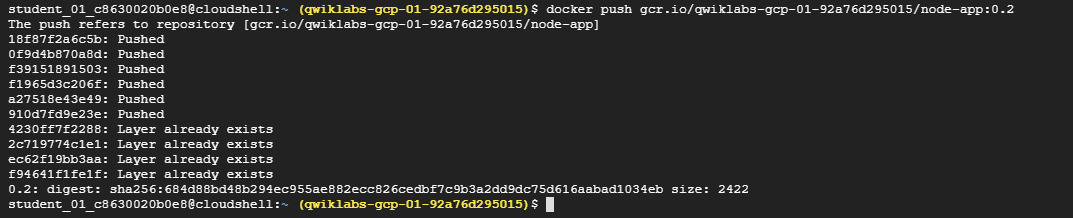
**Curl both ports 8080 and 4000**



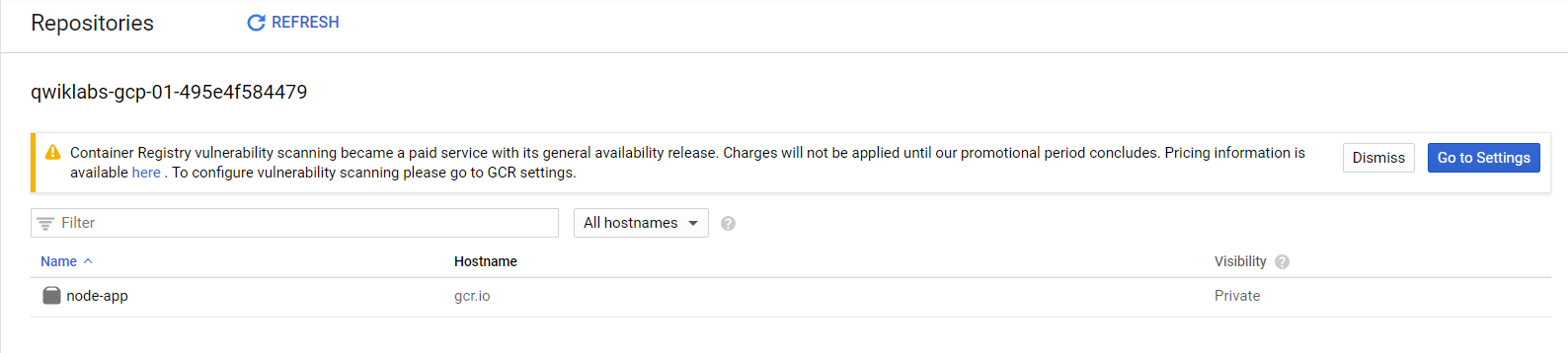
**Inspecting the docker**



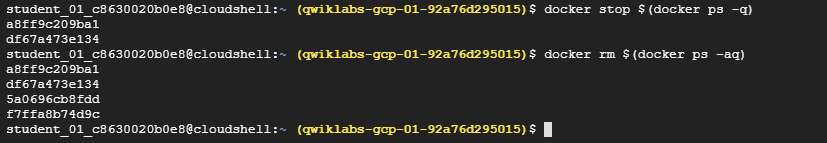
**Viewing all docker images again**



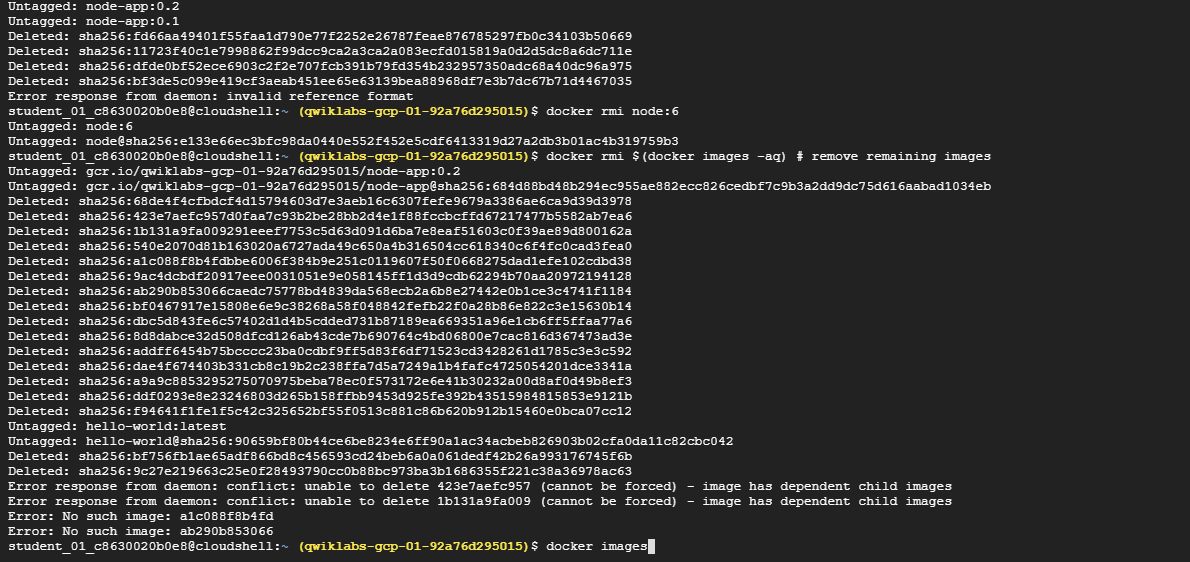
**Pushing into docker image**



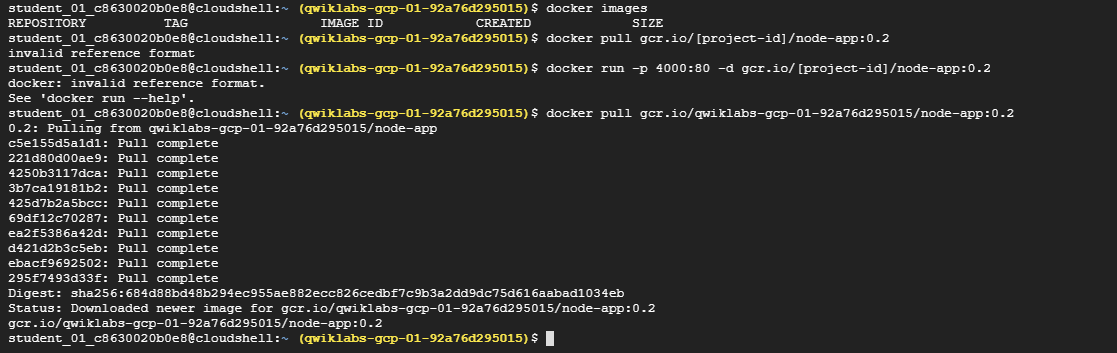
**Repositories dashboard contain node-app docker**



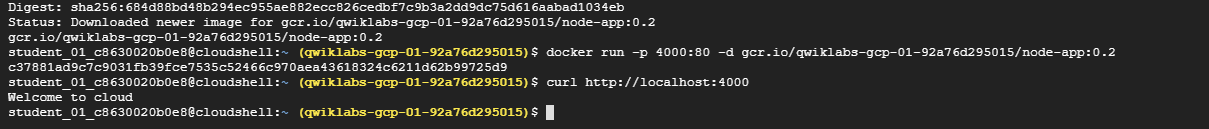
**Stopping docker and then removing using nesting commands**



**Removing remaining images**

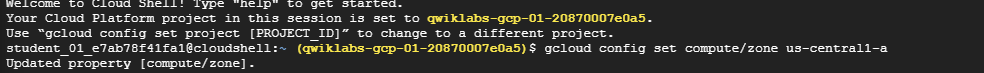


**Docker pulling**

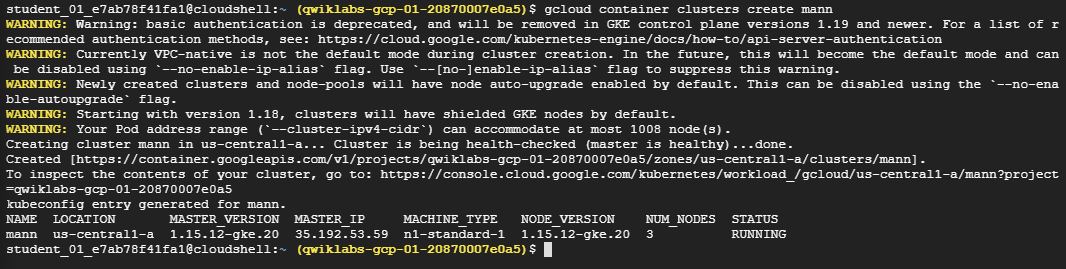


**Running docker again on port 4000**

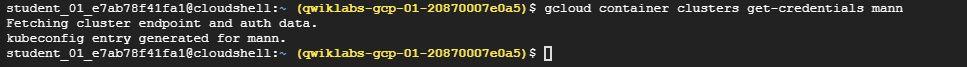
**7.2 - Kubernetes Engine: Qwik Start**



**Setting up compute zone to us-central-a**



**Creating cluster with name “mann”**



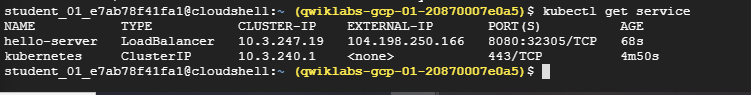
**Fetching cluster endpoint ad auth data**



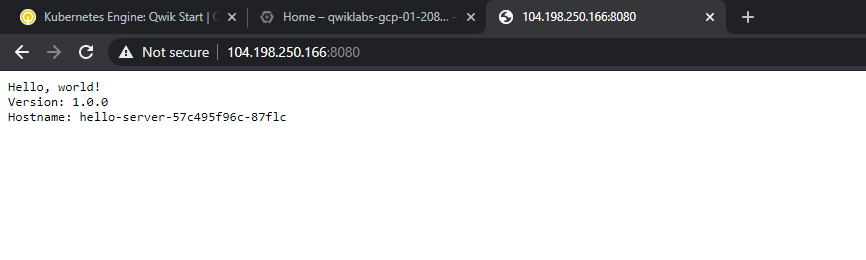
**Using cloud shell we deployed a hello-server server**



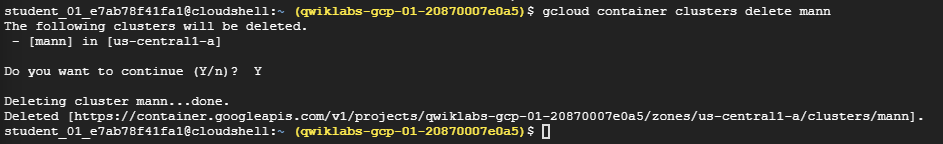
**Expose deployment on port 8000 with type load balancer**



**This command help to fetch all deployed instances**

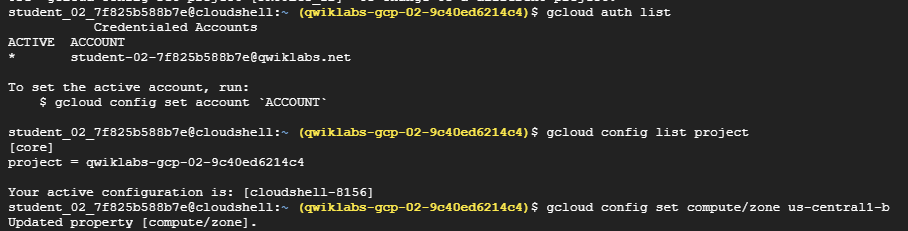


**Accessing using external IP just generated on previous step**

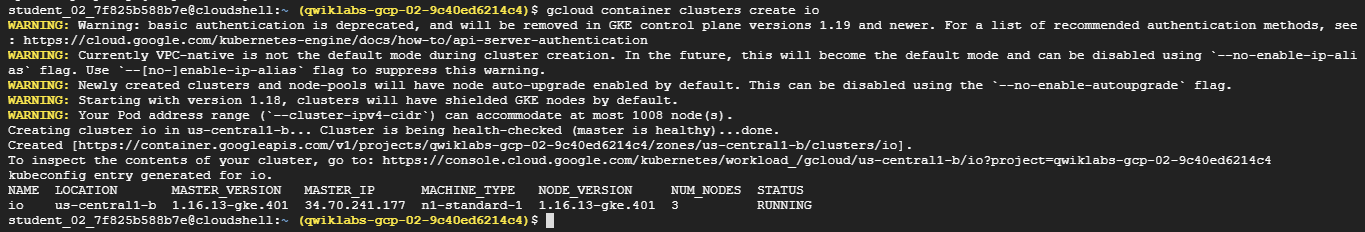


**Deleting the container generated on the very first step**

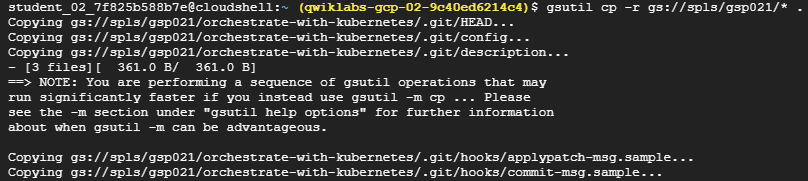
**7.3 - Orchestrating the Cloud with Kubernetes**



**Fetching auth list and project ID and set the compute engine zome to us-central1-b**



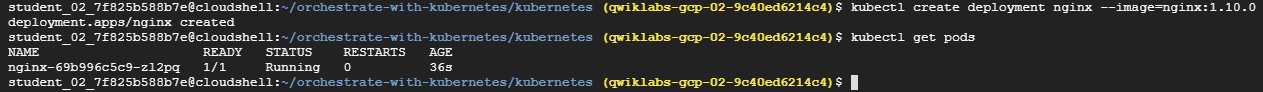
**Creating cluster with name io**



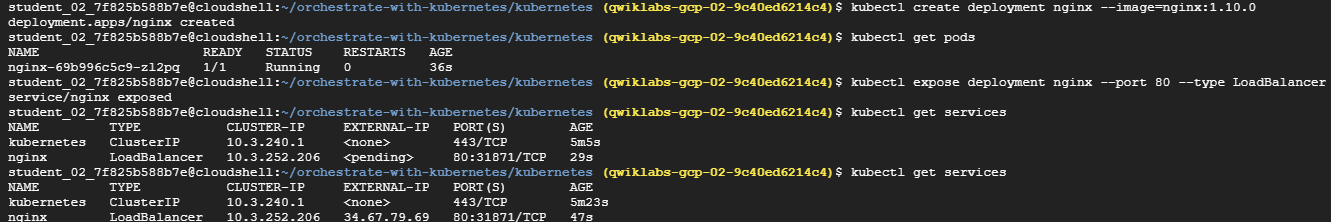
**Copying files with recursively flag**



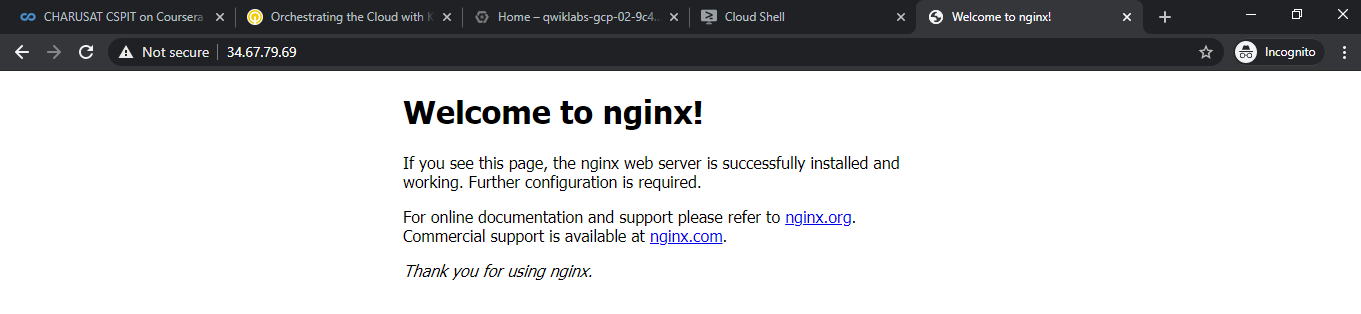
**Changing directory into the copied folder**



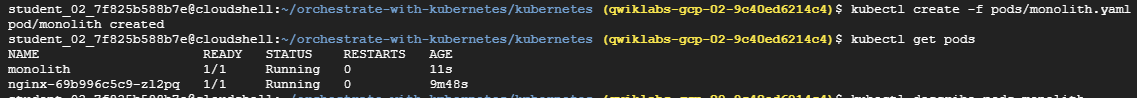
**Creating nginx deployment and fetching pods using get command**



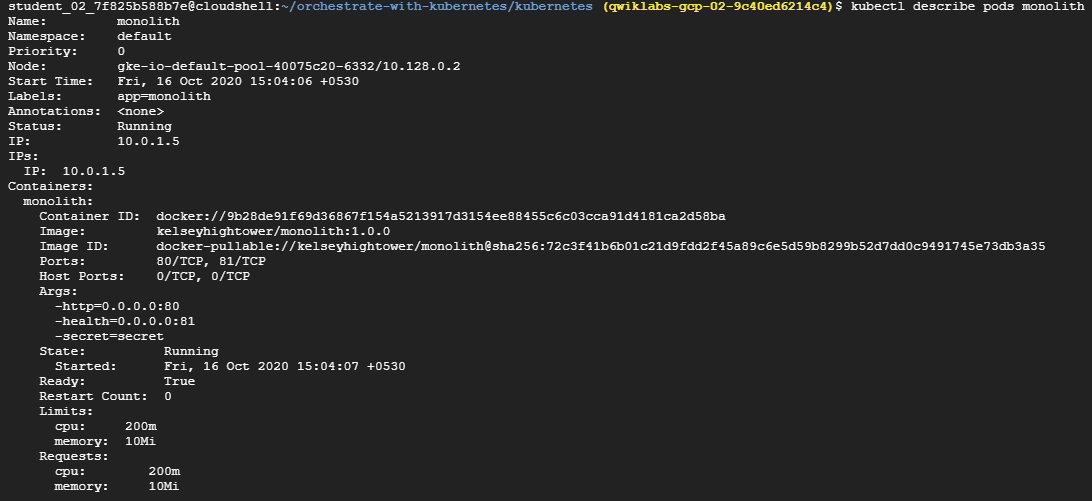
**Now exposing nginx on port 80 with type load balancer, and fetching service until external ip change from pending to some actual value**



**Using external IP, we able to see nginx default page**



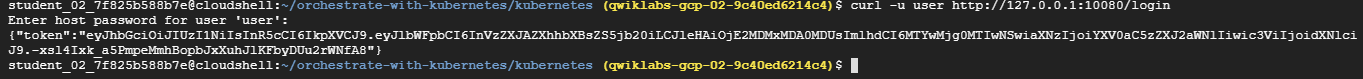
**Creating pods with predefined file monolith.yaml under pods directory**



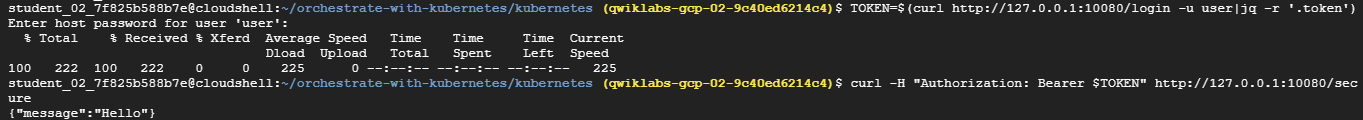
**Describing pods**



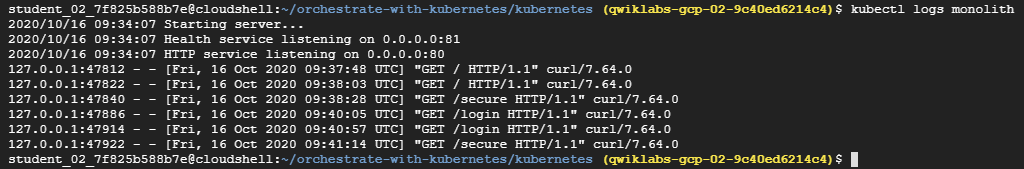
**Forwarding ports to 80**



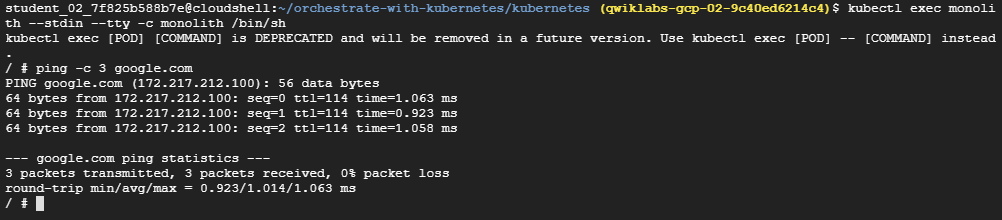
**Login with JWT token**



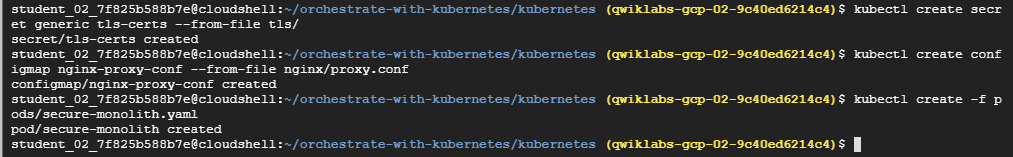
**Using token for accessing nonpublic contents**



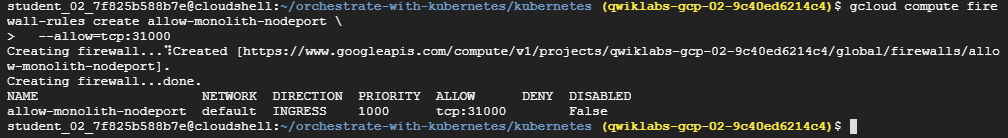
**Displaying monolith logs**



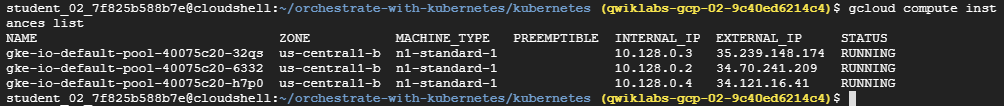
**Executing monolith**



**Creating secrets, configmap and pods**



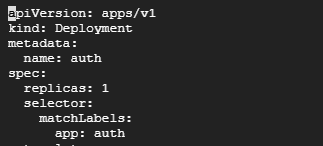
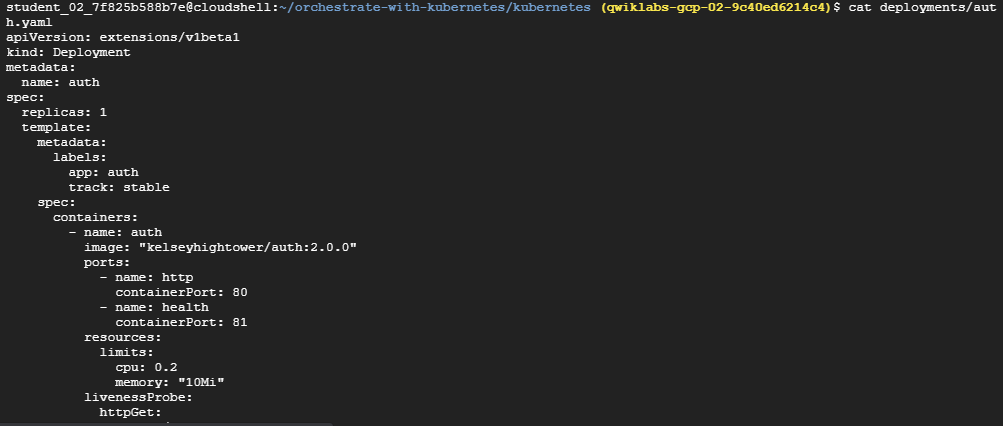
**Creating firewall rules**



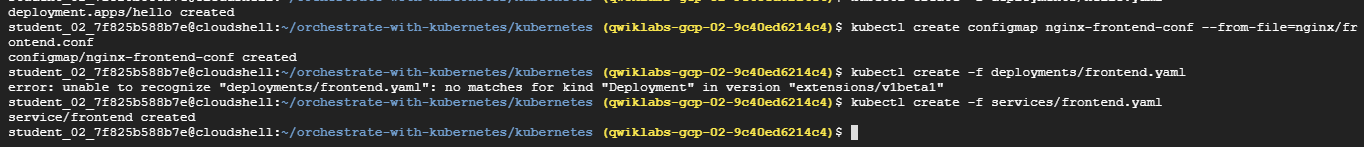
**Listing out instance**



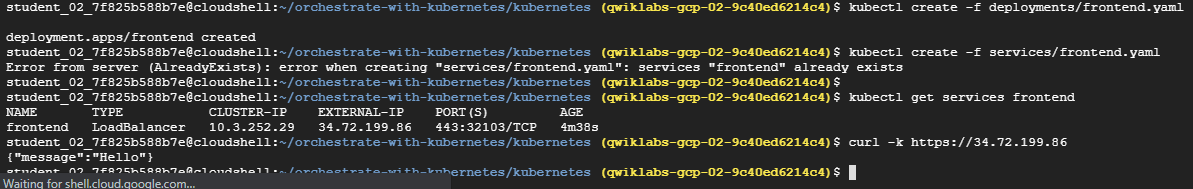
**Unable to get response from monolith server**



**Setting up appVersion to apps/v1**

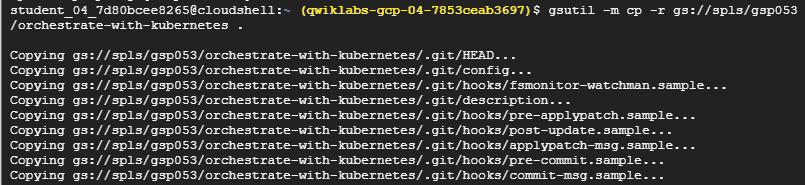


**Creating from the deployment frontend.yaml and services frontend.yaml**

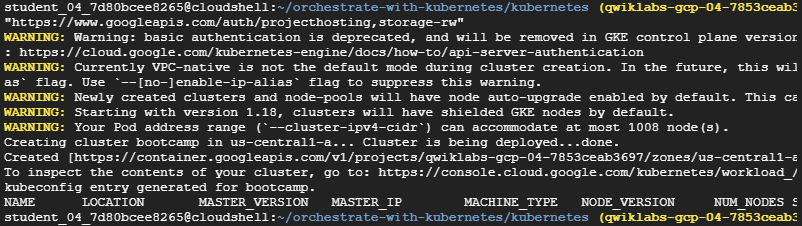


**Successfully receiving the message json object**

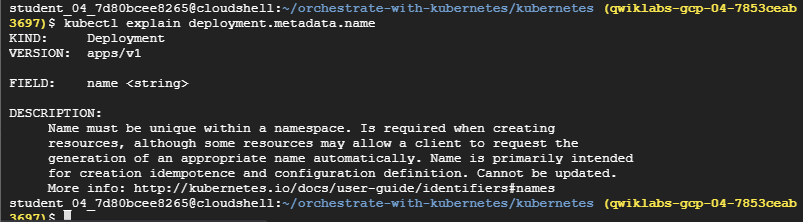
**7.4 - Managing Deployments Using Kubernetes Engine**



**Copying files**



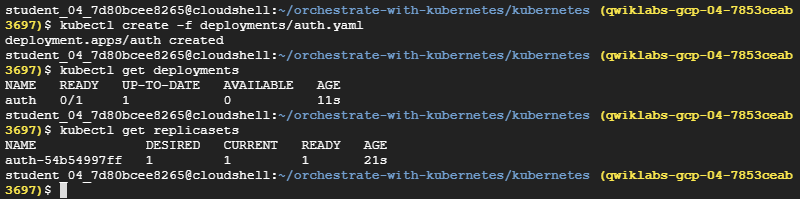
**Creating cluster and node-pools**



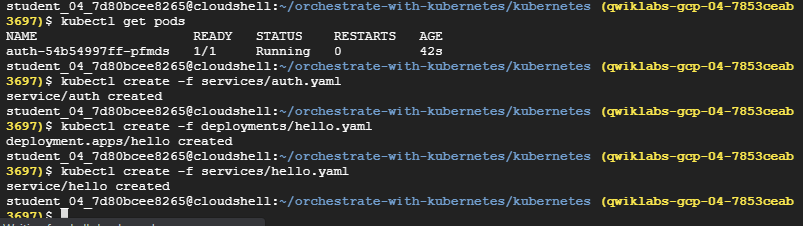
**Explain metadata.name**



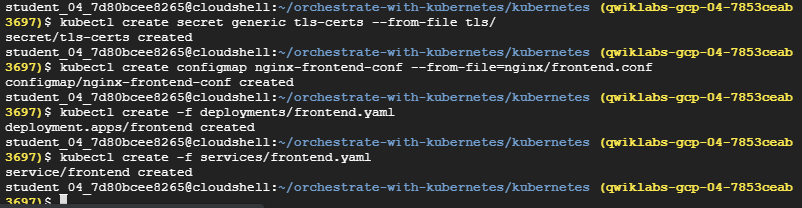
**Changing auth:1.0.0 to 2.0.0**



**Creating deployment with auth.yaml**

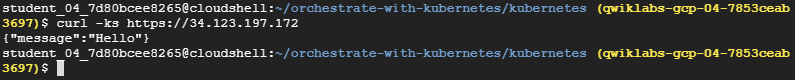


**Getting podsnad creating services and deployment from hello.yaml**

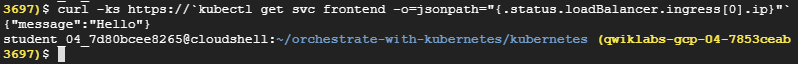


**Creating generic tls certificate, configmap for frontend**

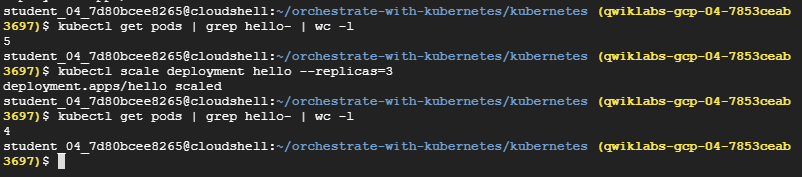




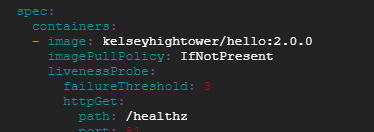
**Able access message from external IP of frontend load balancer**



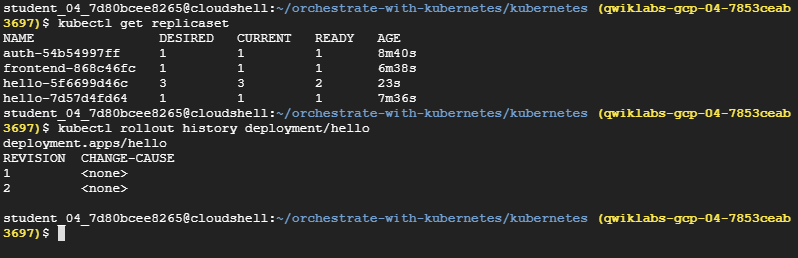
**Another way where json path contain exactly same IP**



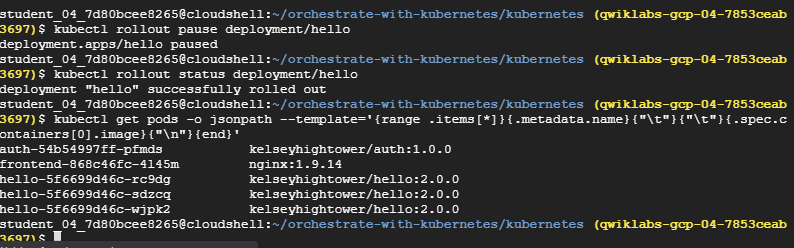
**Displaying WC(Word Count) command**



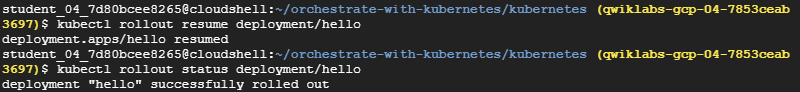
**Hello.yaml**



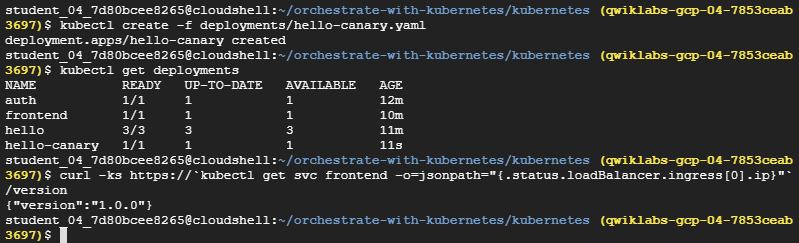
**Getting replicaset**



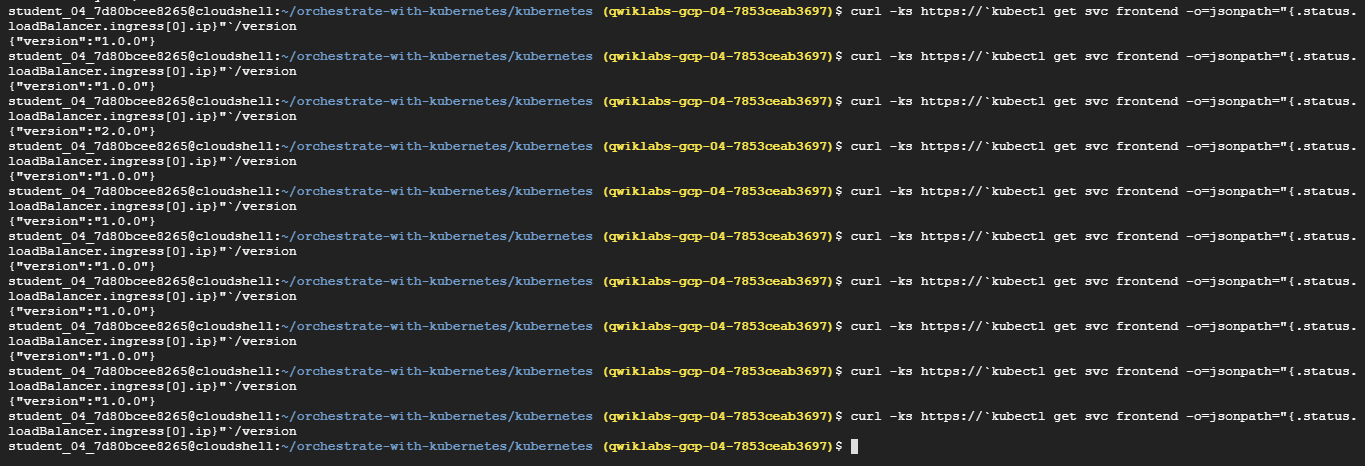
**Rolling out then executing roll out status**



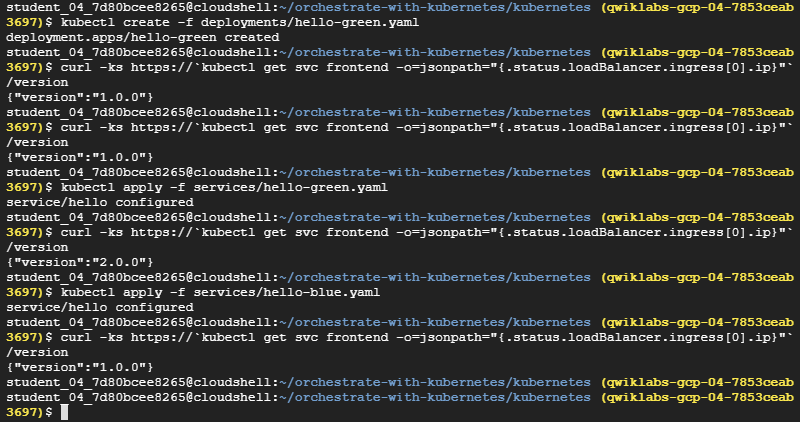
**Again, resuming the rollout**



**Finally create deployments**

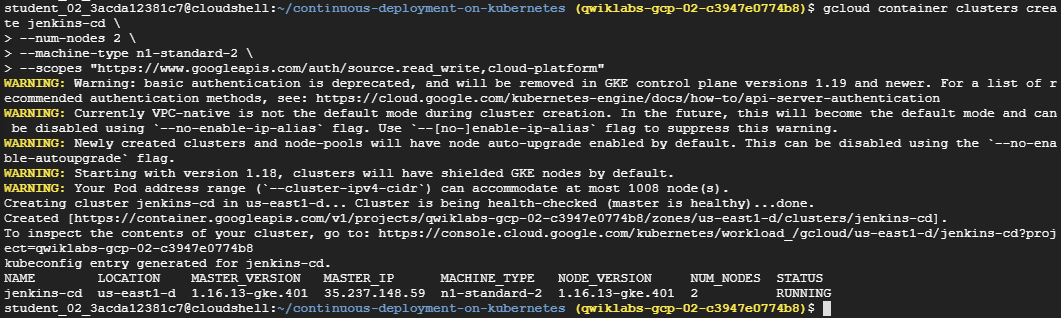


**Pinging up sometime it access 1.0.0 and some time 2.0.0**

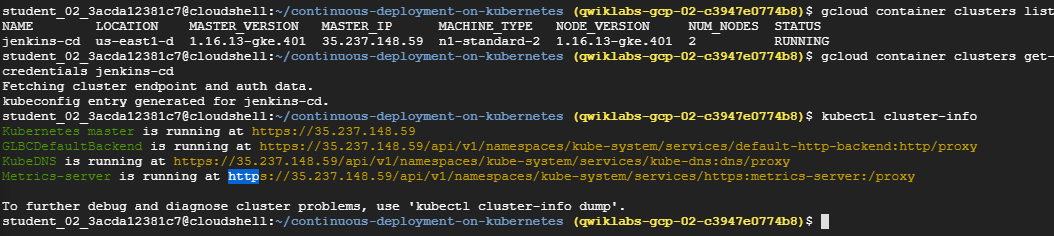


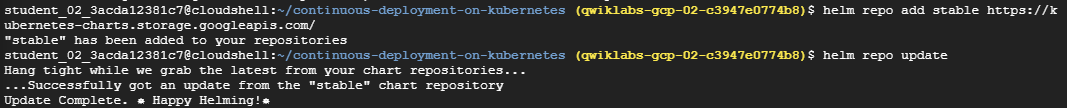
**Similarly, we are seeing version sometimes comes up 1.0.0 and sometimes 2.0.0**

**7.5 - Continuous Delivery with Jenkins in Kubernetes Engine**

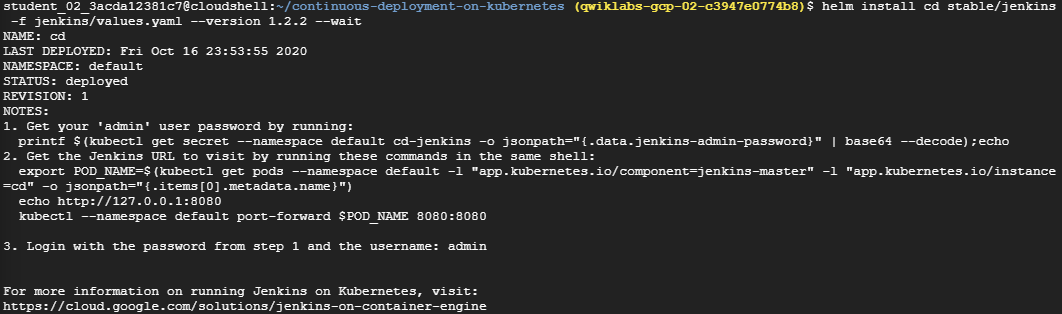


**Creating cluster with name Jenkins-cd**

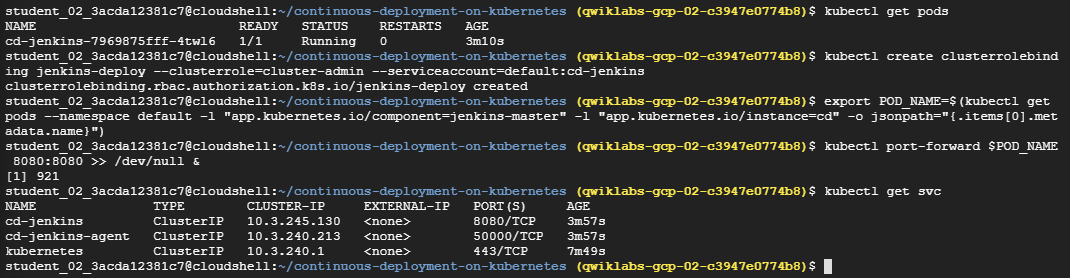


**Fetching cluster endpoint and auth data**

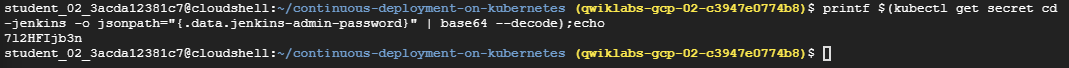
**Using helm added stable in the repository**



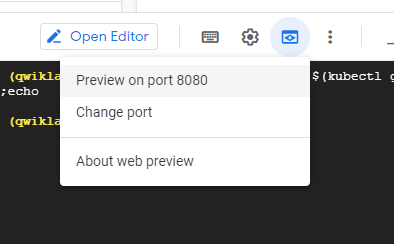
**Installing Jenkins**

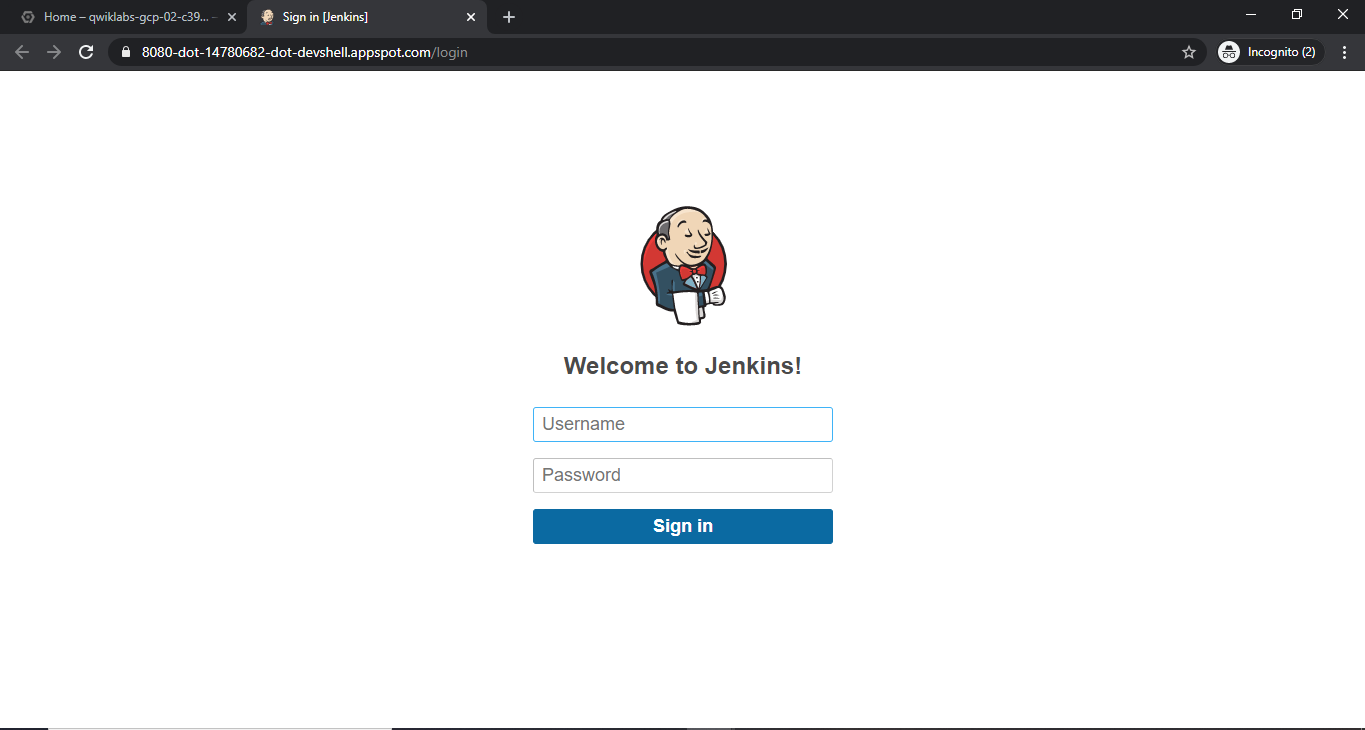


**Getting all 3 kubernetes engine**

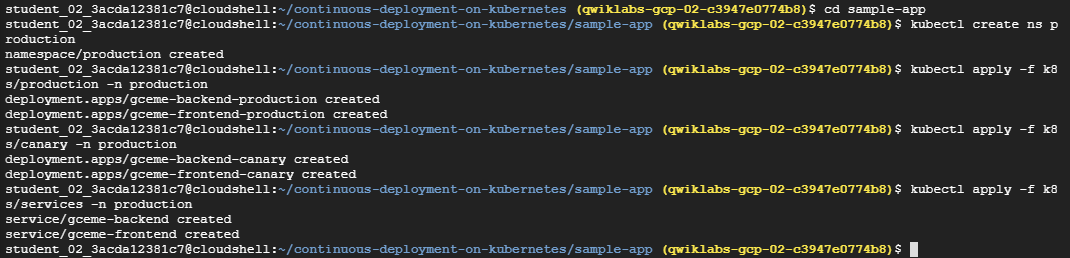


**Printing secret**

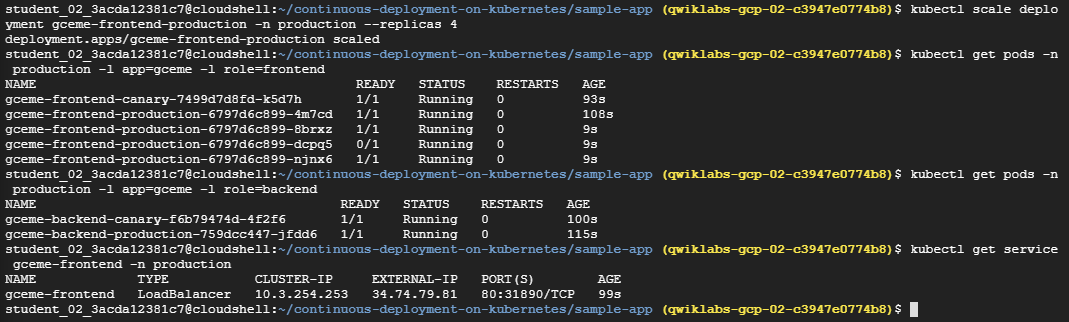


**Previewing on port 8080**

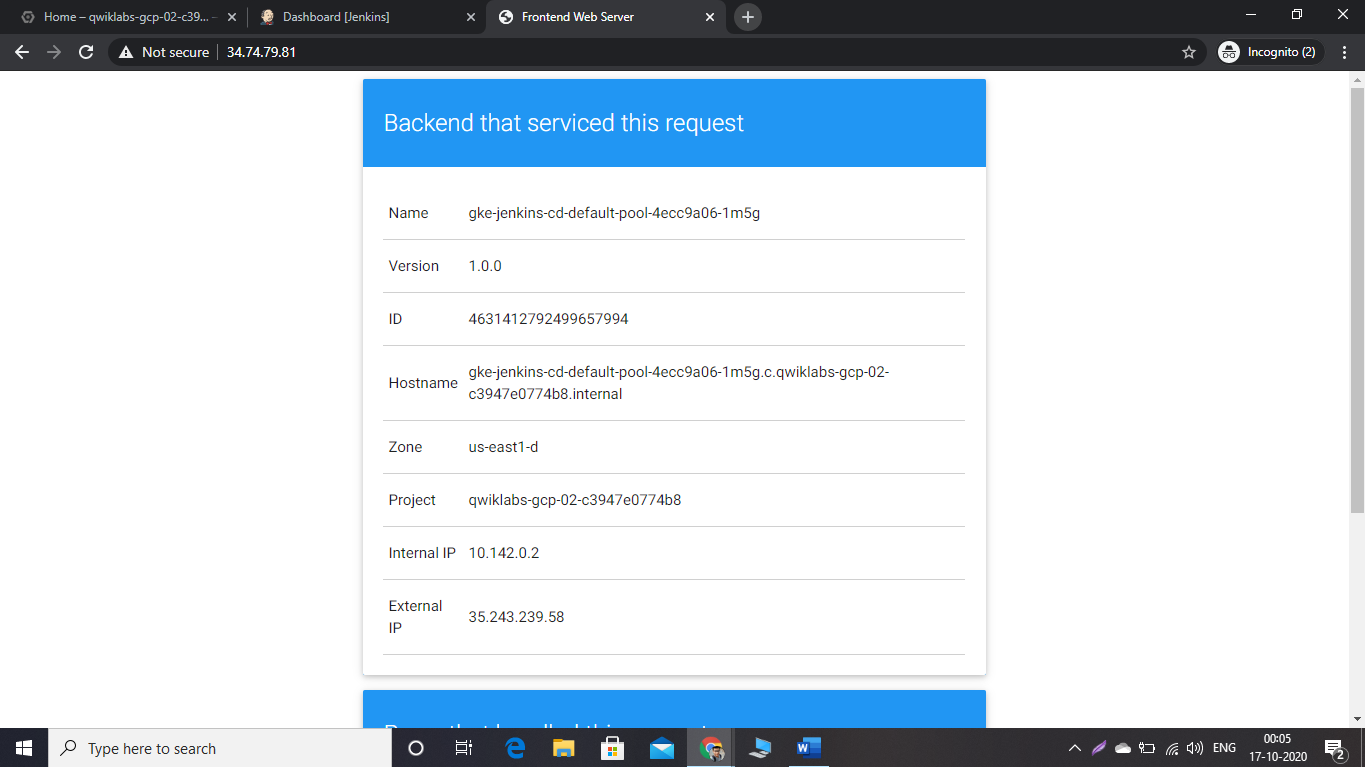
**Jenkins is not accessible used exact same user and password generated in previous step**



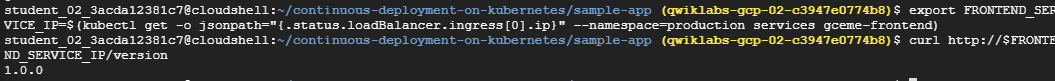
**All production created**

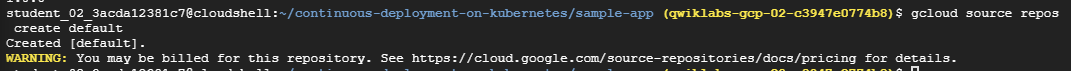


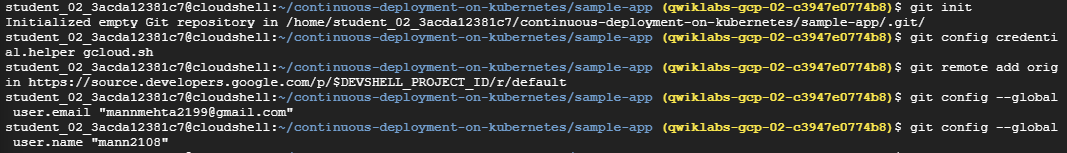
**Gceme-frontend Deployment**



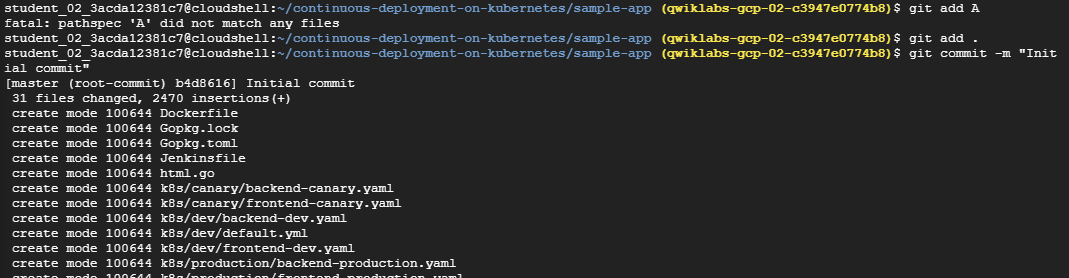
**Frontend webserver blue**

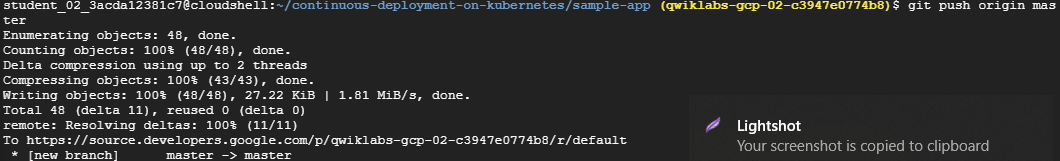




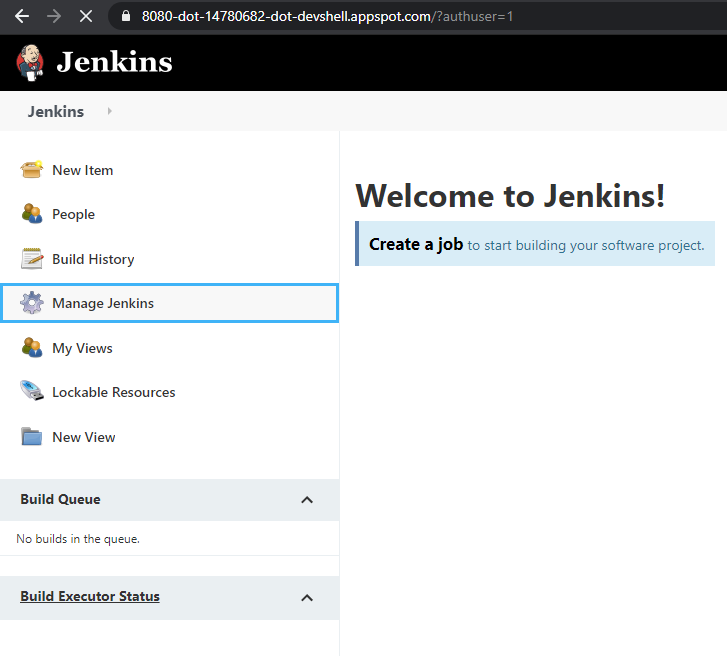


**Export frontend server, sorce repo created, initializing git and added global user name and email**

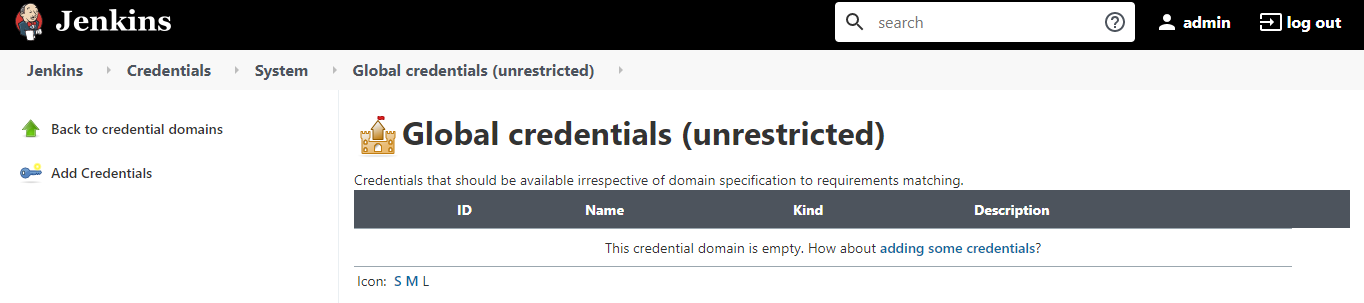




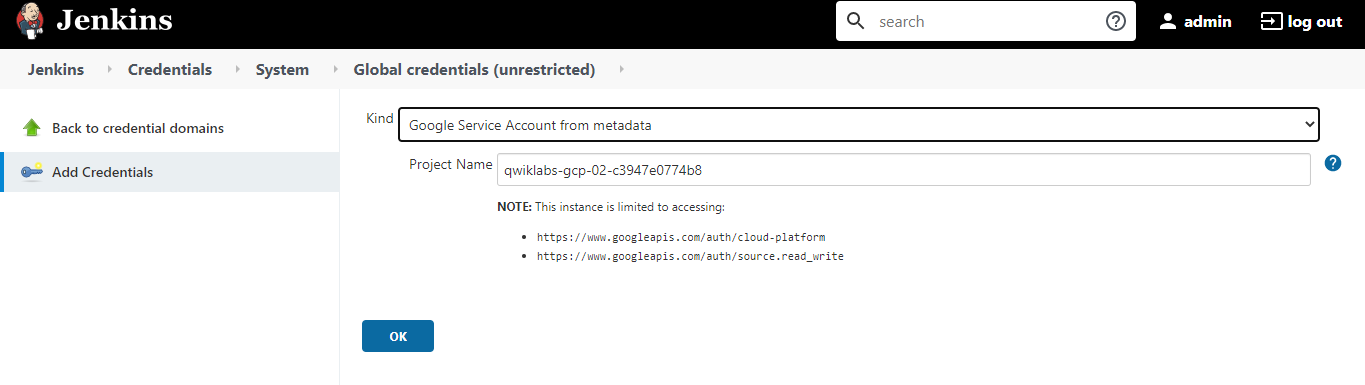
**Git add, commit and push**

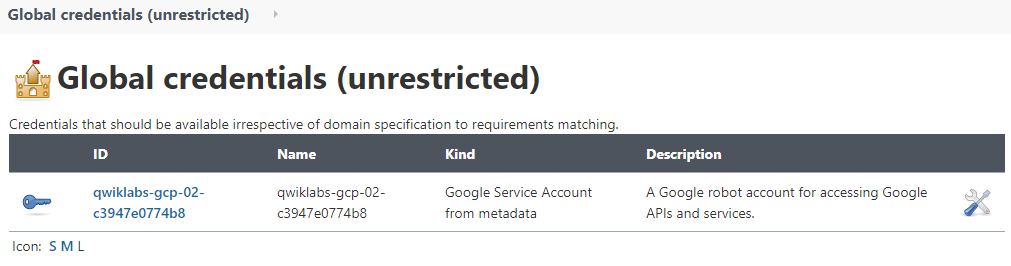


**Navigating to manage Jenkins**

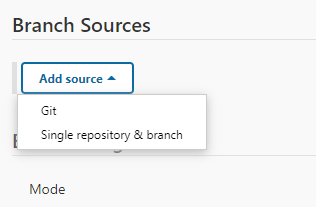


**Adding new credentials**

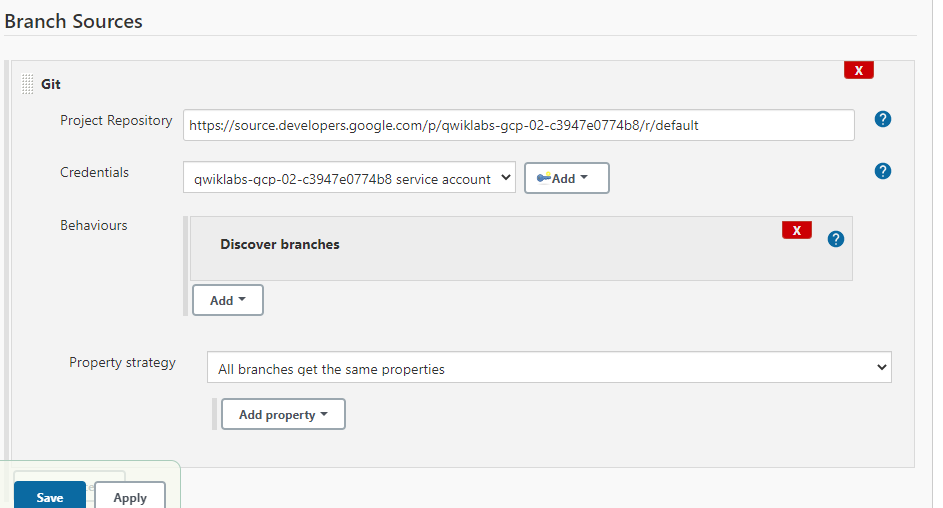




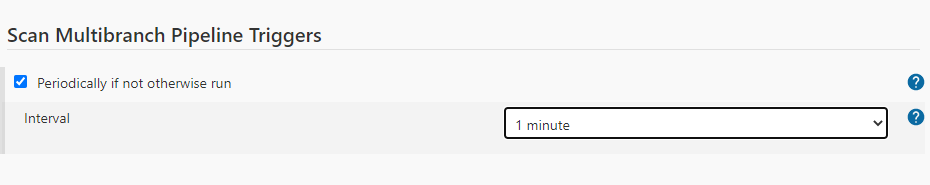
**Credentials created**



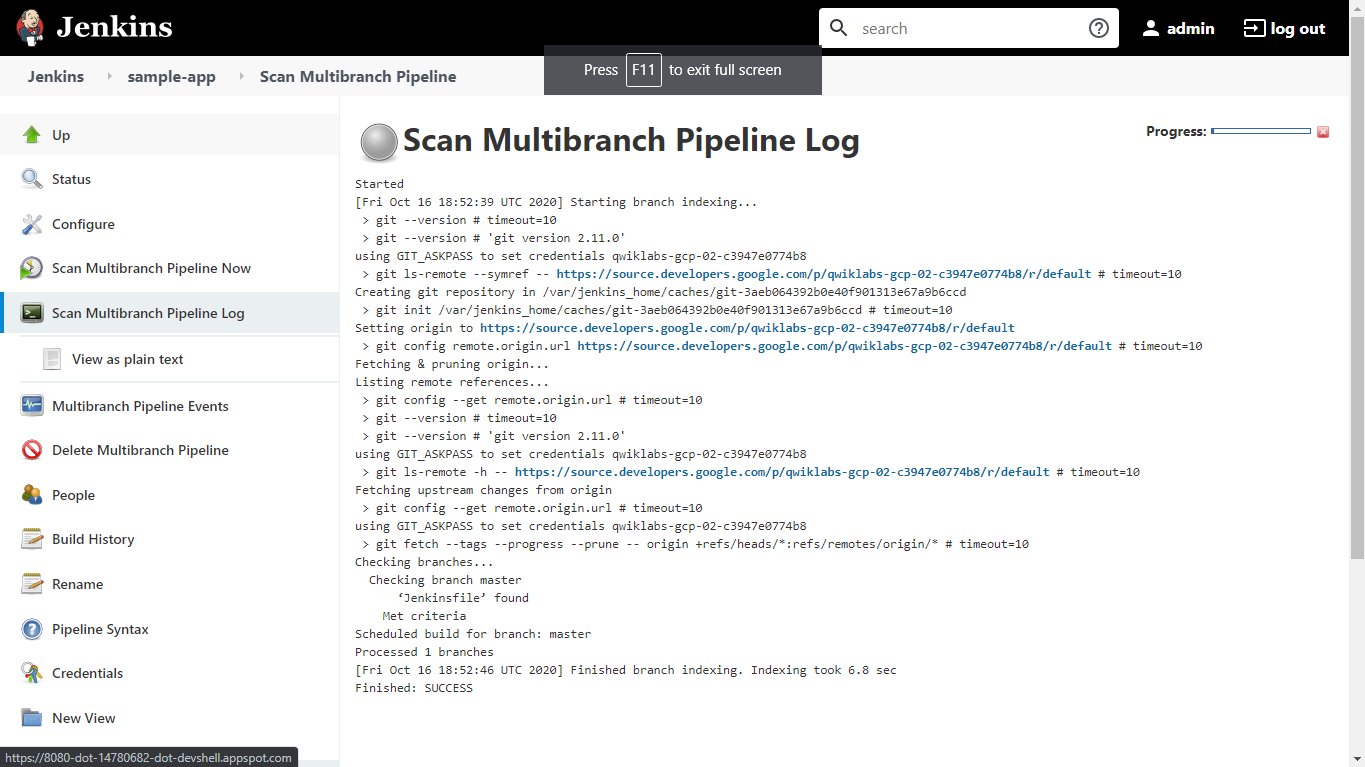
**Adding git as branch source**



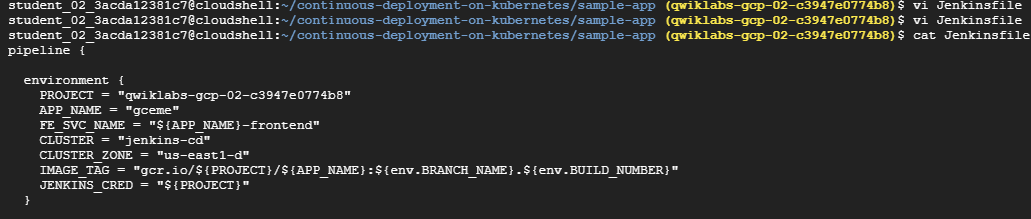
**Assigning git repo and credentials**



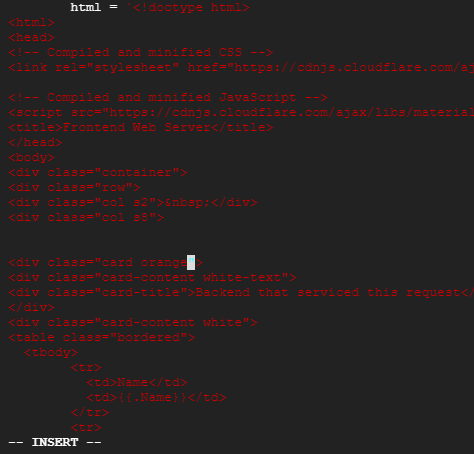
**Scan multibranch triggers**



**Scan multibranch pipeline logs**



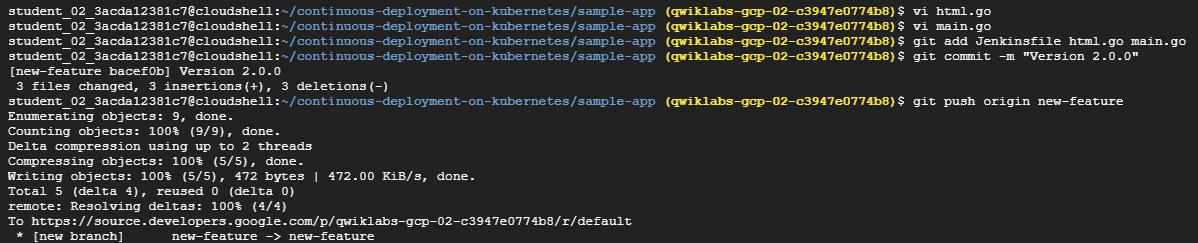
**viewing jenkinfile**



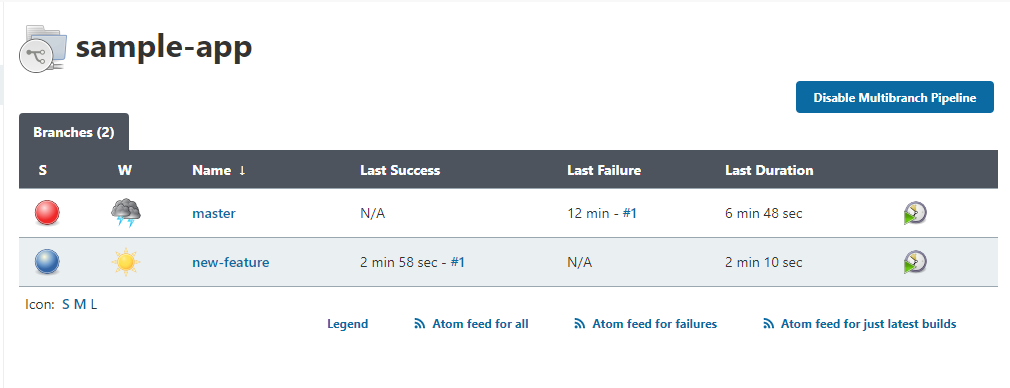
**Here we change card color blue to orange**



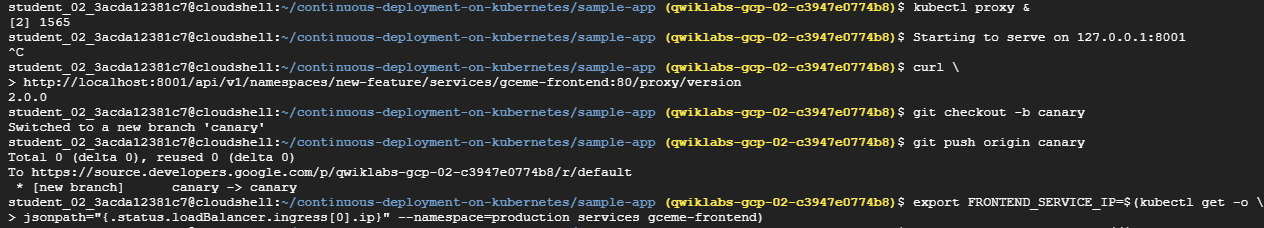
**Changing version 1.0.0 to 2.0.0**

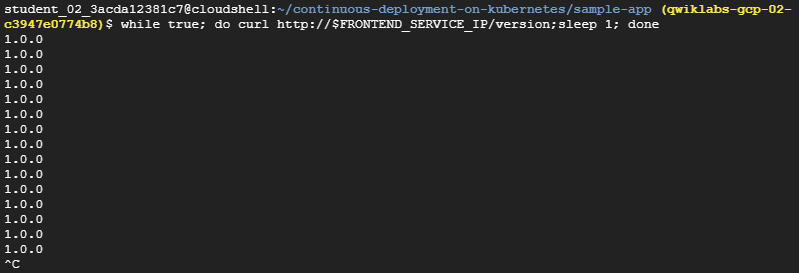


**After adding git commit and push**

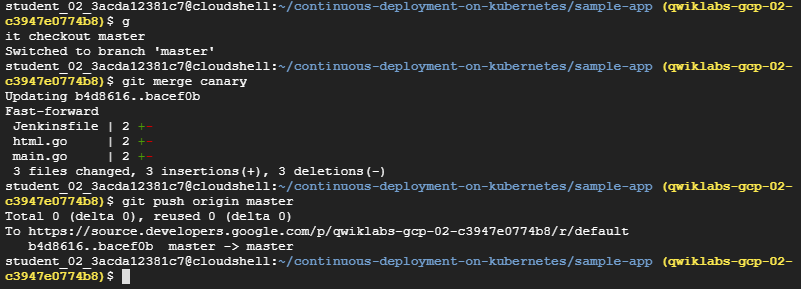


**Jenkins autodetect changes in branches**





**Curling infinitely until stop manually**



**Checking out to master and then git commit, and push master**

**LATEST APPLICATIONS:**

Kubernetes applications are enterprise-ready containerized solutions with prebuilt deployment templates, featuring portability, simplified licensing, and consolidated billing. They can be run on Anthos, in the cloud, on-premises, or on Kubernetes clusters hosted in other environments.

**Planet Scale -** Designed on the same principles that allows Google to run billions of containers a week, Kubernetes can scale without increasing your ops team.

**Never Outgrow -** Whether testing locally or running a global enterprise, Kubernetes flexibility grows with you to deliver your applications consistently and easily no matter how complex your need is.

**Run Anywhere -** Kubernetes is open source giving you the freedom to take advantage of on-premises, hybrid, or public cloud infrastructure, letting you effortlessly move workloads to where it matters to you.

**LEARNING OUTCOME:**

7.1 – Learn to build, run and debug docker containers. Pulling docker image from docker hub and google container registry similarly pushing docker images to google container registry.

7.2 - Creates Kubernetes engine cluster, new deployment server, create Kubernetes services and finally cleaning up or delete the cluster.

7.3 – Create Kubernetes cluster and launch nginx container, create monolith pods and services, allow traffic to the monolith nodeport, adding labels to pods and finally creating deployments (Auth, Hello and Frontend)

7.4 – Canary deployment, a canary deployment, or canary release, is a deployment pattern that allows you to roll out new code/features to a subset of users as an initial test

7.5 – Create Kubernetes cluster, configure and install Jenkins, create production and canary deployments and finally creates repository.

**REFERENCE:**

1. <https://google.qwiklabs.com/quests/29>