**Orchestrating The Cloud With Kubernetes on GCP**

**MINI PROJECT REPORT**

**18CSE316J – Essentials in Cloud and Devops**

**LABORATORY**

**(2018 Regulation)**

**III Year/ VI Semester**

**Academic Year: 2022 -2023**

By

**SHIVAM SINGH (RA2011003010087)**

**Abstract**

Over the past decade there has been tremendous change and advancement in how applications are built and deployed by developers as a result of the ever-increasing demands from end users. "Orchestrating the Cloud with Kubernetes" is a book that provides an in-depth overview of Kubernetes, an open-source container orchestration platform that simplifies the deployment, scaling, and management of containerized applications. The book covers the basics of containerization, microservices architecture, and the advantages of using Kubernetes in a cloud environment. It also delves into the core Kubernetes concepts, including pods, deployments, services, and volumes, and provides practical guidance on how to deploy and manage applications using Kubernetes. Additionally, the book covers advanced topics such as security, networking, and monitoring, as well as the integration of Kubernetes with other cloud-native technologies. Overall, "Orchestrating the Cloud with Kubernetes" serves as a comprehensive guide for anyone looking to deploy and manage containerized applications in the cloud using Kubernetes.

**Index**

|  |  |  |
| --- | --- | --- |
| **Sr No.** | **Topic** | **Page No** |
| **1** | **Introduction** | **5** |
| **2** | **Literature Survey** | **10** |
| **3** | **Methodology** | **12** |
| **4** | **System Architecture** | **13** |
| **5** | **Workflow** | **16** |
| **6** | **CodePipeline** | **20** |
| **7** | **Deploying applications with Kubernetes** | **21** |
| **8** | **Deployment of other services** | **22** |
| **9** | **Output** | **34** |
| **10** | **Conclusion** | **36** |
| **11** | **Futureworks** | **37** |
| **12** | **References** | **38** |

**ABBREVIATIONS**

AWS : Amazon Web Services

IaC : Infrastructure as Code

Node.js : Node, a JavaScript runtime

npm : Node Package Manager, a package manager for Node.js

CLI : Command Line Interface

EC2 Elastic Compute Cloud

IAM Identity and Access Management

SG : Security Group

**INTRODUCTION**

**Aim :**

The aim of "Orchestrating the Cloud with Kubernetes" on Google Cloud Boost (GCB) is to provide a comprehensive and practical guide for developers and IT professionals to leverage Kubernetes on Google Cloud Platform (GCP). The book covers the basics of containerization and microservices, and then focuses on the core concepts of Kubernetes, including pods, deployments, services, and volumes, as well as advanced topics such as security, networking, and monitoring. Readers will learn how to use Kubernetes on GCP to deploy, scale, and manage containerized applications effectively.

The book aims to empower readers to take full advantage of Kubernetes' benefits, such as fault tolerance, high availability, and scalability, to build resilient and robust cloud-native applications. It also provides guidance on integrating Kubernetes with other GCP services, such as Google Cloud Storage, Google Cloud Load Balancing, and Google Cloud Logging, to create a complete cloud-native application ecosystem.

Overall, "Orchestrating the Cloud with Kubernetes" on GCB aims to be a practical and comprehensive resource for anyone looking to deploy and manage containerized applications on Google Cloud Platform using Kubernetes.

**Context of project :**

As businesses increasingly move towards cloud-based solutions and microservices architecture, the need for efficient deployment and management of containerized applications becomes essential. Kubernetes, an open-source container orchestration platform, has emerged as the go-to solution for simplifying the deployment, scaling, and management of containerized applications in a cloud environment. "Orchestrating the Cloud with Kubernetes" is a comprehensive guide to understanding and utilizing Kubernetes for cloud-native application management. This book covers the basics of containerization and microservices architecture, and then dives into the core concepts of Kubernetes, including pods, deployments, services, and volumes. With practical guidance on how to deploy and manage applications using Kubernetes, readers will be able to take full advantage of the platform's benefits. The book also explores advanced topics such as security, networking, and monitoring, making it a valuable resource for both beginners and experienced users. By the end of this book, readers will have a deep understanding of how to deploy and manage containerized applications in the cloud using Kubernetes.

**Objective :**

The objective of "Orchestrating the Cloud with Kubernetes" on Google Cloud Books (GCB) is to provide readers with a practical and comprehensive understanding of how to leverage Kubernetes on Google Cloud Platform (GCP) for deploying and managing containerized applications.

The book aims to help readers achieve the following objectives:

1. Understand the basics of containerization and microservices architecture, and why Kubernetes is an ideal solution for managing containerized applications in a cloud environment.

2. Gain a deep understanding of Kubernetes core concepts, such as pods, deployments, services, and volumes, and how to use them effectively on GCP

3. Learn how to deploy and manage containerized applications using Kubernetes on GCP, including scaling and rolling updates.

4. Explore advanced Kubernetes topics such as security, networking, and monitoring, and how to apply them to GCP.

5. Integrate Kubernetes with other GCP services to build a complete cloud-native application ecosystem, including Google Cloud Storage, Google Cloud Load Balancing, and Google Cloud Logging.

Overall, the objective of "Orchestrating the Cloud with Kubernetes" on GCB is to equip readers with the knowledge and skills to deploy, scale, and manage containerized applications effectively on Google Cloud Platform using Kubernetes.

**Goals:**

The goal of "Orchestrating the Cloud with Kubernetes" on Google Cloud Boost (GCB) is to help readers master the skills necessary to leverage Kubernetes effectively for deploying and managing containerized applications on Google Cloud Platform (GCP).

The book aims to achieve the following goals:

1. Provide a comprehensive and practical understanding of Kubernetes core concepts, such as pods, deployments, services, and volumes, and how to use them effectively on GCP.

2. Teach readers how to deploy and manage containerized applications on GCP using Kubernetes, including scaling and rolling updates.

3. Explore advanced Kubernetes topics such as security, networking, and monitoring, and how to apply them to GCP.

4. Demonstrate how to integrate Kubernetes with other GCP services to build a complete cloud-native application ecosystem, including Google Cloud Storage, Google Cloud Load Balancing, and Google Cloud Logging.

5. Help readers become proficient in using Kubernetes to build resilient and scalable cloud-native applications on GCP.

THE MAIN TASK ARE-

1. Create a Kubernetes cluster and launch Nginx container
2. Create Monolith pods and service
3. Allow traffic to the monolith service on the exposed nodeport
4. Adding Labels to Pods
5. Creating Deployments (Auth, Hello and Frontend)

Overall, the goal of "Orchestrating the Cloud with Kubernetes" on GCB is to equip readers with the knowledge and skills to leverage Kubernetes effectively on GCP, and enable them to build, deploy, and manage robust and scalable cloud-native applications.

**LITERATURE SURVEY**

"Orchestrating the Cloud with Kubernetes" is a popular book on the topic of Kubernetes and cloud-native application management. It has been well-received by readers and has received positive reviews for its comprehensive coverage of Kubernetes concepts, practical guidance, and hands-on examples.

Other books on Kubernetes and cloud-native application management that are worth mentioning include:

1. "Kubernetes: Up and Running" by Brendan Burns, Joe Beda, and Kelsey Hightower: This book is an excellent resource for those looking to get started with Kubernetes. It covers the basics of Kubernetes, including deployment, scaling, and management, and provides practical examples for deploying and managing applications on Kubernetes.

2. "Cloud Native Infrastructure" by Kris Nova and Justin Garrison: This book provides a comprehensive overview of cloud-native infrastructure, including Kubernetes and other cloud-native technologies. It covers topics such as infrastructure as code, containers, and microservices, and provides practical guidance on deploying and managing applications in a cloud-native environment.

3. "Kubernetes in Action" by Marko Lukša: This book provides a deep dive into Kubernetes concepts, including pods, services, and deployments, and provides practical guidance on how to deploy and manage applications using Kubernetes. It also covers advanced topics such as security and networking.

4. "The Kubernetes Book" by Nigel Poulton: This book is a beginner-friendly guide to Kubernetes, providing an overview of its core concepts and practical examples for deploying and managing applications on Kubernetes.

Overall, "Orchestrating the Cloud with Kubernetes" stands out for its comprehensive coverage of Kubernetes on Google Cloud Platform, making it an excellent resource for those looking to deploy and manage containerized applications on GCP using Kubernetes.

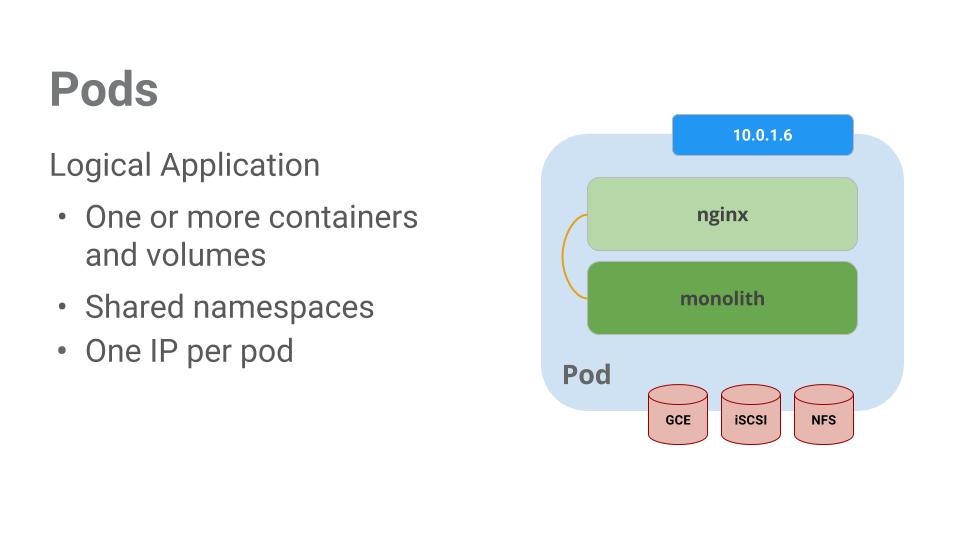
**METHODOLOGY**

* Kubernetes is an open source project (available on [kubernetes.io](http://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.
* Login on GCP activate cloud shell, Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources. When you are connected, you are already authenticated, and the project is set to your **PROJECT\_ID**. The output contains a line that declares the **PROJECT\_ID** for this session.
* For Google Kubernets Engine, In the cloud shell environment type the following command to set the zone and start up a cluster for use in this lab. Then ou are automatically authenticated to your cluster upon creation. If you lose connection to your Cloud Shell for any reason, run the gcloud container clusters get-credentials io command to re-authenticate. **Note:**It will take a while to create a cluster - Kubernetes Engine is provisioning a few Virtual Machines behind the scenes for you to play with! And so on ..

**SYSTEM ARCHITECTURE**

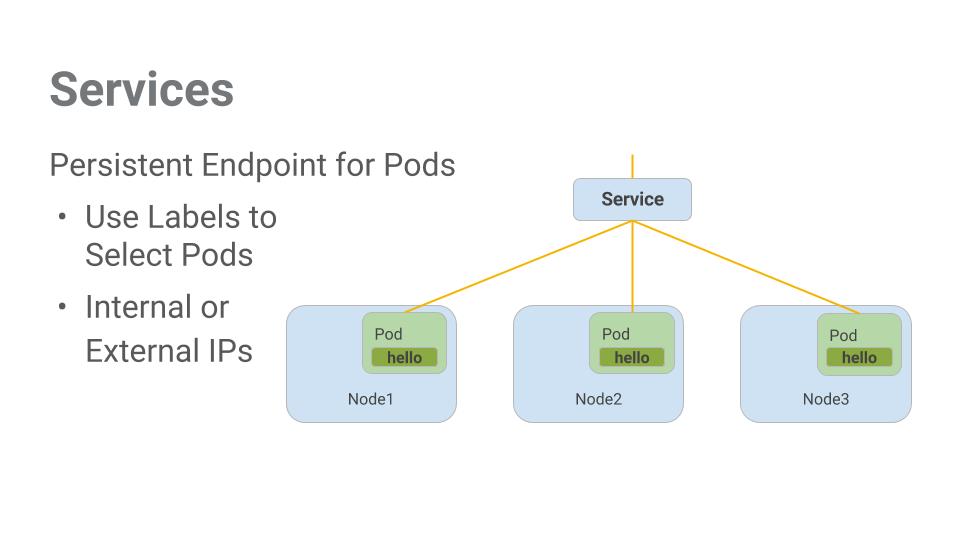
At the core of Kubernetes is the [Pod](http://kubernetes.io/docs/user-guide/pods/).

Pods represent and hold a collection of one or more containers. Generally, if you have multiple containers with a hard dependency on each other, you package the containers inside a single pod.



Pods also have [Volumes](http://kubernetes.io/docs/user-guide/volumes/). Volumes are data disks that live as long as the pods live, and can be used by the containers in that pod. Pods provide a shared namespace for their contents which means that the two containers inside of our example pod can communicate with each other, and they also share the attached volumes.

Pods also share a network namespace. This means that there is one IP Address per pod.



Pods aren't meant to be persistent. They can be stopped or started for many reasons - like failed liveness or readiness checks - and this leads to a problem:

What happens if you want to communicate with a set of Pods? When they get restarted they might have a different IP address.

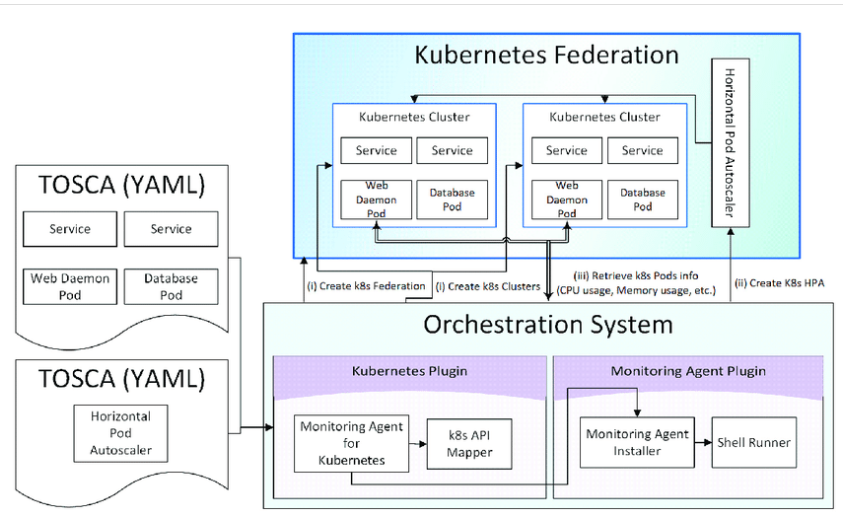
That's where [Services](http://kubernetes.io/docs/user-guide/services/) come in. Services provide stable endpoints for Pods.

Services use labels to determine what Pods they operate on. If Pods have the correct labels, they are automatically picked up and exposed by our services.

The level of access a service provides to a set of pods depends on the Service's type. Currently there are three types:

* ClusterIP (internal) -- the default type means that this Service is only visible inside of the cluster,
* NodePort gives each node in the cluster an externally accessible IP and
* LoadBalancer adds a load balancer from the cloud provider which forwards traffic from the service to Nodes within it.

**WORKFLOW**



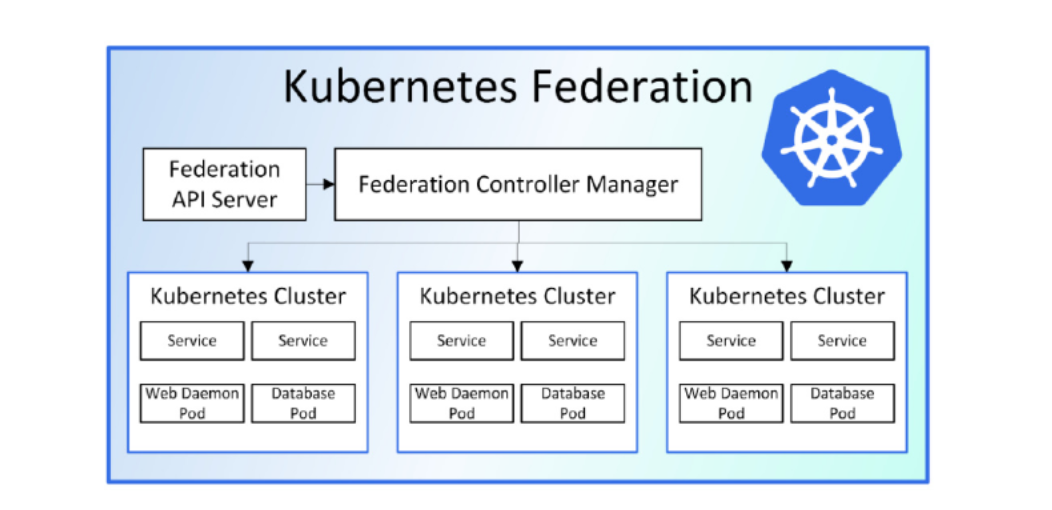
**Kubernetes, a container orchestration tool for automatically installing and managing Docker containers, has recently begun to support a federation function of multiple Docker container clusters. This technology, called Kubernetes Federation, allows developers to increase the responsiveness and reliability of their applications by distributing and federating container clusters to multiple service areas of cloud service providers. However, it is still a daunting task to manually manage federated container clusters across all the service areas or to maintain the entire topology of cloud applications at a glance. This research work proposes a method to automatically form and monitor Kubernetes Federation, given application topology descriptions in TOSCA (Topology and Orchestration Specification for Cloud Applications), by extending the orchestration tool that automatizes the modeling and instantiation of cloud applications. It also demonstrates the successful federation of the clusters according to the TOSCA specifications and verifies the auto-scaling capability of the configured system through a scenario in which the servers of a sample application are deployed and federated**.

Conﬁguring a cloud service requires the installation of various components and programs.For instance, conﬁguring a web server requires an operating system, web daemon, and a databaseserver. The conventional method of installing constituting parts on actual machines limits thecomputing resources to a particular service execution and does not allow the resources to be shared bydifferent services.Docker containerizes individual processes and allows them to be run within a separatedlightweight execution environment called a container. As the adoption of Docker has increased,the need to automatize the deployment and management of containerized applications has recentlyarisen . Developed in this context, Kubernetes is an open-source container orchestration toolthat automatically installs and manages a cluster of Docker containers. The service developer cancreate Docker images containing desired service elements and Kubernetes can deploy and manage thecomponents and their relationships. Kubernetes includes the following elements:

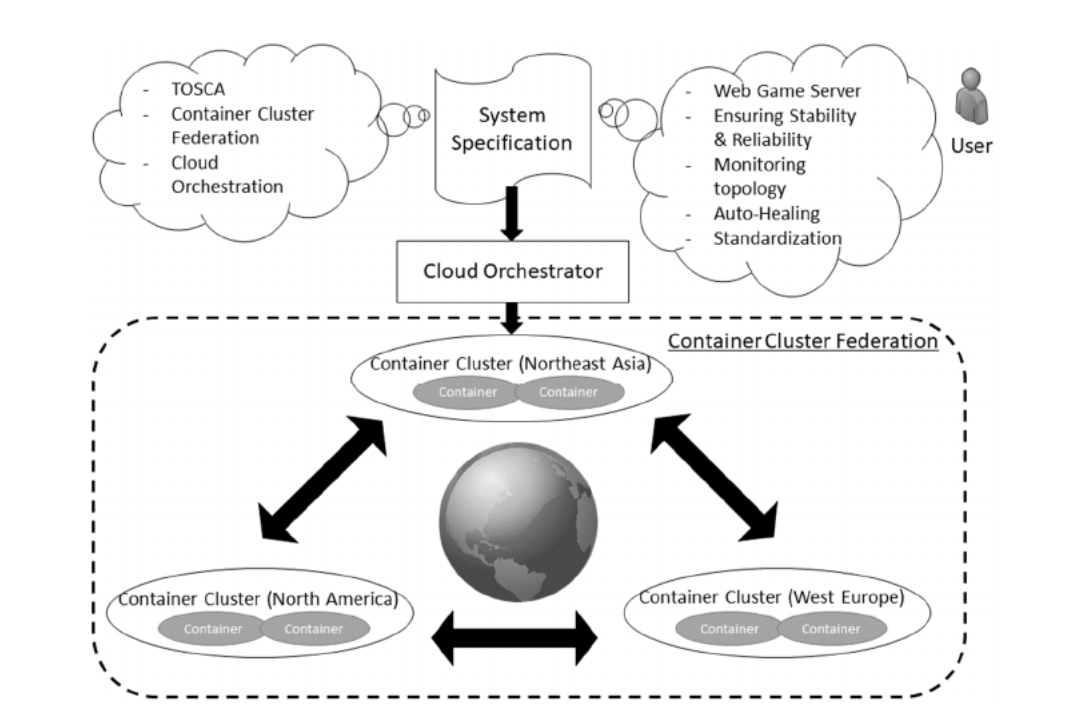
Kubernetes pod: this is an essential building block of Kubernetes, usually containing multipleDocker containers

.•Kubernetes node: this represents a VM (Virtual Machine) or physical machine where theKubernetes pods are run.

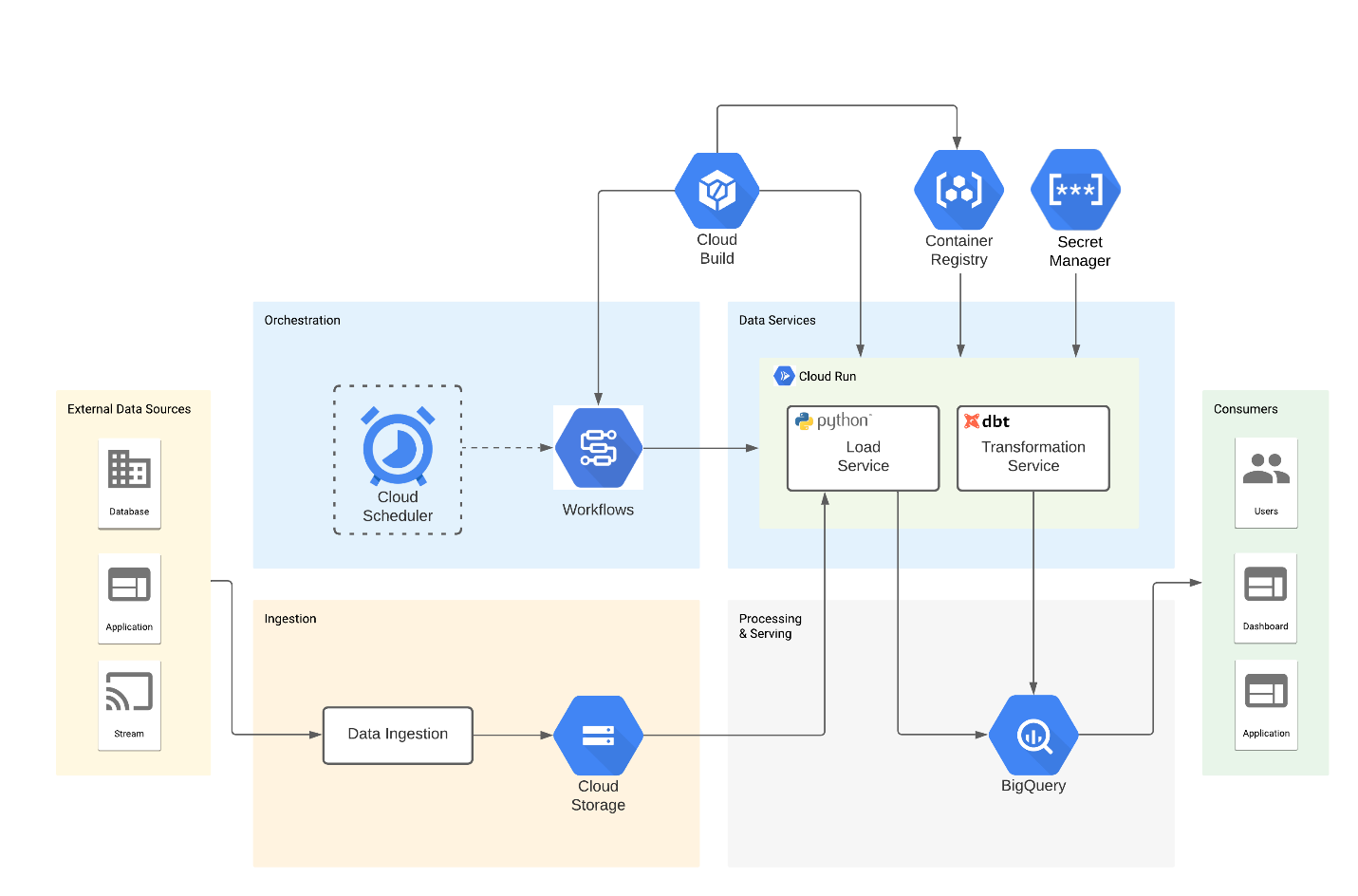
•Kubernetes cluster: this consists of a set of worker nodes that cooperate to run applications as asingle unit. Its master node coordinates all activities within the cluster.



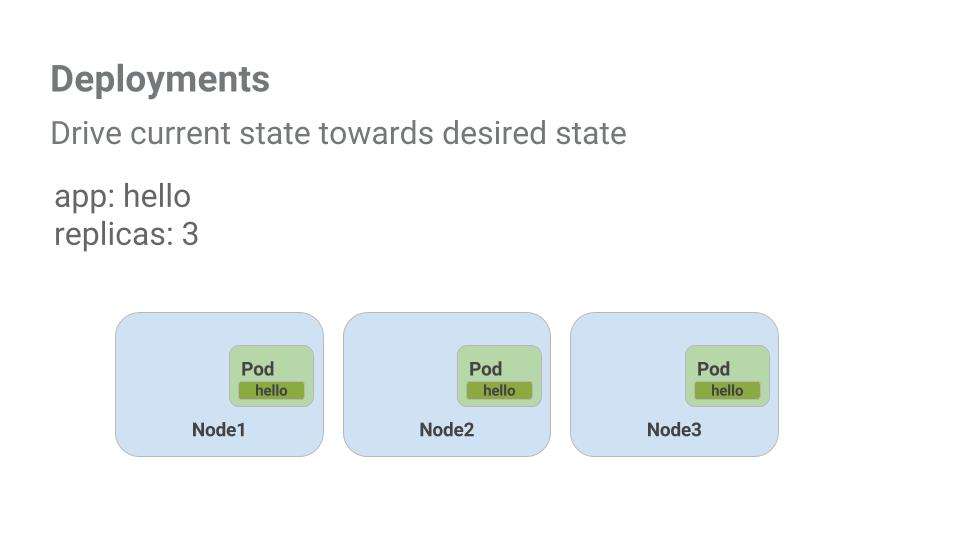
Kubernetes Federation: this is a cluster of clusters, i.e., viewed as a backbone cluster thatcombines multiple Kubernetes clusters. For example, when one Kubernetes cluster is running onGoogle Cloud Platform in Tokyo, Japan, and another is running in Oregon, U.S., one Kubernetesfederation might be conﬁgured such that if there is a problem in the Oregon platform, the Tokyocluster would be able to take over the share of the faulty platform, thereby increasing the resiliencyof the service. Figure 1shows an example of the Kubernetes Federation architecture. Kubernetesprovides a ﬂexible, loosely coupled mechanism for service delivery. The federation applicationprogram interface (API) server interacts with the cluster through the federation controller manager.The master is responsible for exposing the API, scheduling the deployments, and overall clustermanagement. The interaction with the Kubernetes cluster is done through the federation controllermanager using the federation API server.

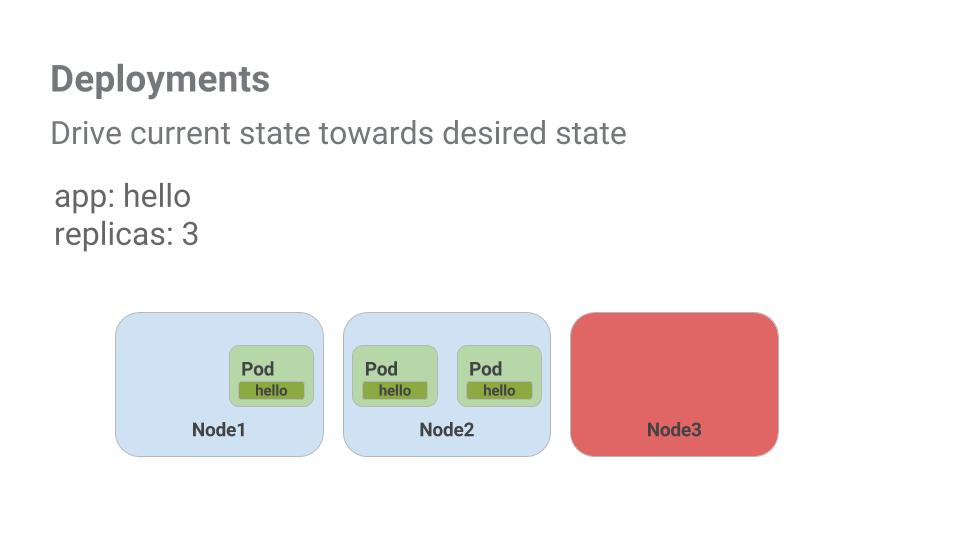


**CODEPIPELINE**



## Deploying applications with Kubernetes





**DEPLOYMENT**

1. Login on gcp
2. Open cloud shell
3. For Google Kubernetes Engine enter the following command on cloud shell

gcloud config set compute/zone us-central1-b

1. For start up cluster for enter the following command

gcloud container clusters create io

1. For sample code, Copy the source code from the Cloud Shell command line:

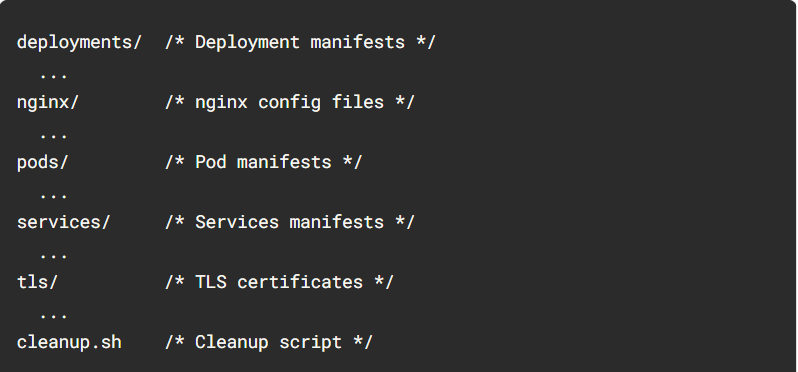
gsutil cp -r gs://spls/gsp021/\* .

1. Change into the directory needed for this lab:

cd orchestrate-with-kubernetes/Kubernetes

1. List the files to see what you're working with: ls

Output -----



kubectl create command.

1. Use it to launch a single instance of the nginx container:

kubectl create deployment nginx --image=nginx:1.10.0

Kubernetes has created a deployment -- more about deployments later, but for now all you need to know is that deployments keep the pods up and running even when the nodes they run on fail.

In Kubernetes, all containers run in a pod.

1. Use the kubectl get pods command to view the running nginx container:

kubectl get pods

1. Once the nginx container has a Running status you can expose it outside of Kubernetes using the kubectl expose command:

kubectl expose deployment nginx --port 80 --type LoadBalancer

content\_copy

So what just happened? Behind the scenes Kubernetes created an external Load Balancer with a public IP address attached to it. Any client who hits that public IP address will be routed to the pods behind the service. In this case that would be the nginx pod.

1. List our services now using the kubectl get services command:

kubectl get services

**Note:** It may take a few seconds before the ExternalIP field is populated for your service. This is normal -- just re-run the kubectl get services command every few seconds until the field populates.

1. Add the External IP to this command to hit the Nginx container remotely:

curl http://<External IP>:80

And there you go! Kubernetes supports an easy to use workflow out of the box using the kubectl run and expose commands.

**For Create Monolith pods and service**

Pods can be created using pod configuration files. Take a moment to explore the monolith pod configuration file.

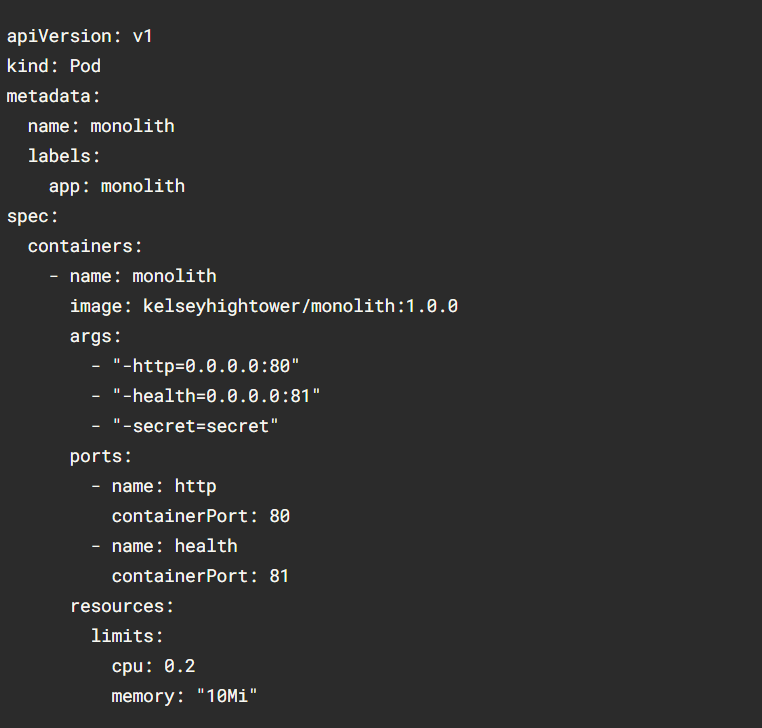
1. Go to directory:

cd ~/orchestrate-with-kubernetes/kubernetes

1. Run the following:

cat pods/monolith.yaml

Output



There's a few things to notice here. You'll see that:

* Your pod is made up of one container (the monolith).
* You're passing a few arguments to our container when it starts up.
* You're opening up port 80 for http traffic.

1. Create the monolith pod using kubectl:

kubectl create -f pods/monolith.yaml

1. Examine your pods. Use the kubectl get pods command to list all pods running in the default namespace:

kubectl get pods

**Note:** It may take a few seconds before the monolith pod is up and running. The monolith container image needs to be pulled from the Docker Hub before you can run it.

1. Once the pod is running, use kubectl describe command to get more information about the monolith pod:

kubectl describe pods monolith

## **Interacting with pods**

1. Open a second Cloud Shell terminal. Now you have two terminals, one to run the kubectl port-forward command, and the other to issue curl commands.
2. In the **2nd terminal**, run this command to set up port-forwarding:

kubectl port-forward monolith 10080:80

1. Now in the **1st terminal** start talking to your pod using curl:

curl http://127.0.0.1:10080

Yes! You got a very friendly "hello" back from your container.

1. Now use the curl command to see what happens when you hit a secure endpoint:

curl http://127.0.0.1:10080/secure

1. Try logging in to get an auth token back from the monolith:

curl -u user http://127.0.0.1:10080/login

1. At the login prompt, use the super-secret password "password" to login.

Logging in caused a JWT token to print out.

1. Since Cloud Shell does not handle copying long strings well, create an environment variable for the token.

TOKEN=$(curl http://127.0.0.1:10080/login -u user|jq -r '.token')

1. Enter the super-secret password "password" again when prompted for the host password.
2. Use this command to copy and then use the token to hit the secure endpoint with curl:

curl -H "Authorization: Bearer $TOKEN" http://127.0.0.1:10080/secure

At this point you should get a response back from our application, letting us know everything is right in the world again.

1. Use the kubectl logs command to view the logs for the monolith Pod.

kubectl logs monolith

1. **Open a 3rd terminal** and use the -f flag to get a stream of the logs happening in real-time:

kubectl logs -f monolith

1. Now if you use curl in the **1st terminal** to interact with the monolith, you can see the logs updating (in the **3rd terminal**):

curl http://127.0.0.1:10080

1. Use the kubectl exec command to run an interactive shell inside the Monolith Pod. This can come in handy when you want to troubleshoot from within a container:

kubectl exec monolith --stdin --tty -c monolith -- /bin/sh

1. For example, once you have a shell into the monolith container you can test external connectivity using the ping command:

ping -c 3 google.com

1. Be sure to log out when you're done with this interactive shell.

exit

## **For  Services**

1. If you've changed directories, make sure you return to the ~/orchestrate-with-kubernetes/kubernetes directory:

cd ~/orchestrate-with-kubernetes/kubernetes

1. Explore the monolith service configuration file:

cat pods/secure-monolith.yaml

1. Create the secure-monolith pods and their configuration data:

kubectl create secret generic tls-certs --from-file tls/

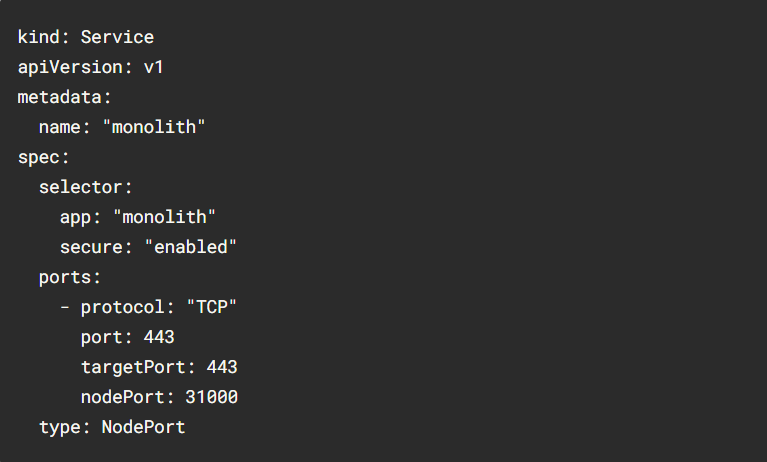
kubectl create configmap nginx-proxy-conf --from-file nginx/proxy.conf

kubectl create -f pods/secure-monolith.yaml

Now that you have a secure pod, it's time to expose the secure-monolith Pod externally.To do that, create a Kubernetes service.

1. Explore the monolith service configuration file:

cat services/monolith.yaml



1. Use the kubectl create command to create the monolith service from the monolith service configuration file:

kubectl create -f services/monolith.yaml

1. Use the gcloud compute firewall-rules command to allow traffic to the monolith service on the exposed nodeport:

gcloud compute firewall-rules create allow-monolith-nodeport \

--allow=tcp:31000

## **Adding labels to pods**

1. You can see that you have quite a few pods running with the monolith label.

kubectl get pods -l "app=monolith"

1. But what about "app=monolith" and "secure=enabled"?

kubectl get pods -l "app=monolith,secure=enabled"

1. Use the kubectl label command to add the missing secure=enabled label to the secure-monolith Pod. Afterwards, you can check and see that your labels have been updated.

kubectl label pods secure-monolith 'secure=enabled'

kubectl get pods secure-monolith --show-labels

1. Now that your pods are correctly labeled, view the list of endpoints on the monolith service:

kubectl describe services monolith | grep Endpoints

1. Test this out by hitting one of our nodes again.

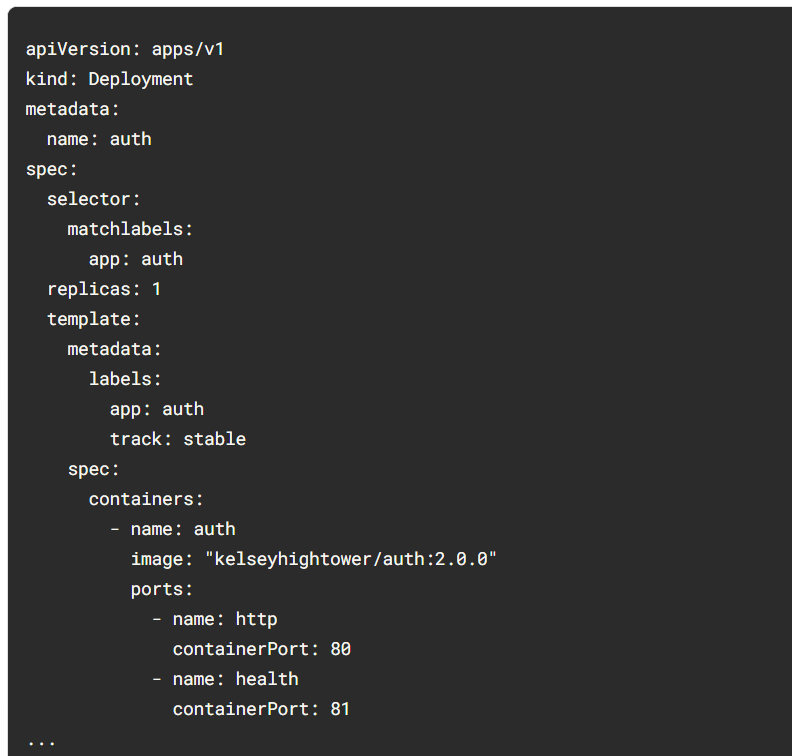
gcloud compute instances list

curl -k https://<EXTERNAL\_IP>:31000

## **Creating Deploying applications with Kubernetes**

1. Get started by examining the auth deployment configuration file.

cat deployments/auth.yaml



1. Anyway, go ahead and create your deployment object:

kubectl create -f deployments/auth.yaml

1. It's time to create a service for your auth deployment. Use the kubectl create command to create the auth service:

kubectl create -f services/auth.yaml

1. Now do the same thing to create and expose the hello deployment:

kubectl create -f deployments/hello.yaml

kubectl create -f services/hello.yaml

1. And one more time to create and expose the frontend Deployment.

kubectl create configmap nginx-frontend-conf --from-file=nginx/frontend.conf

kubectl create -f deployments/frontend.yaml

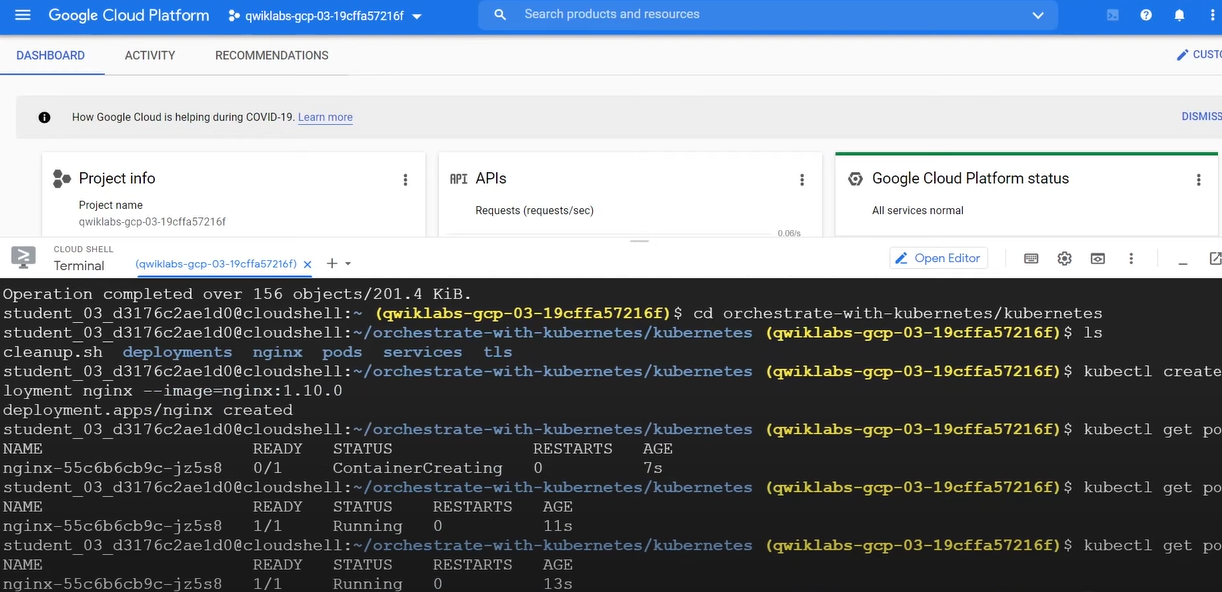
kubectl create -f services/frontend.yaml

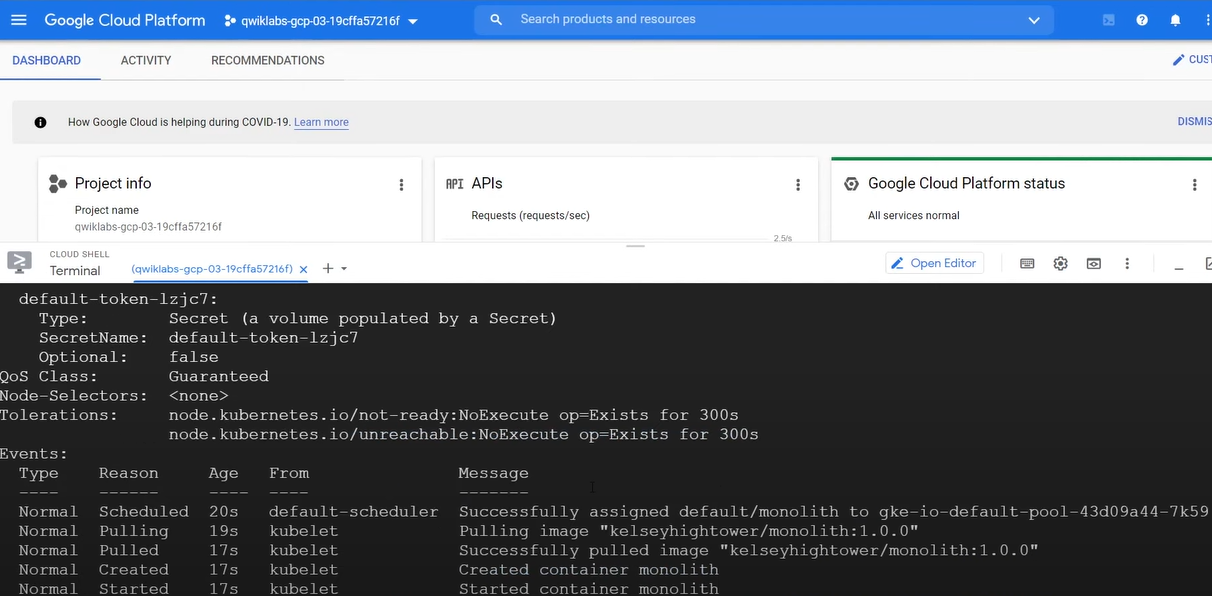
1. Interact with the frontend by grabbing its External IP and then curling to it:

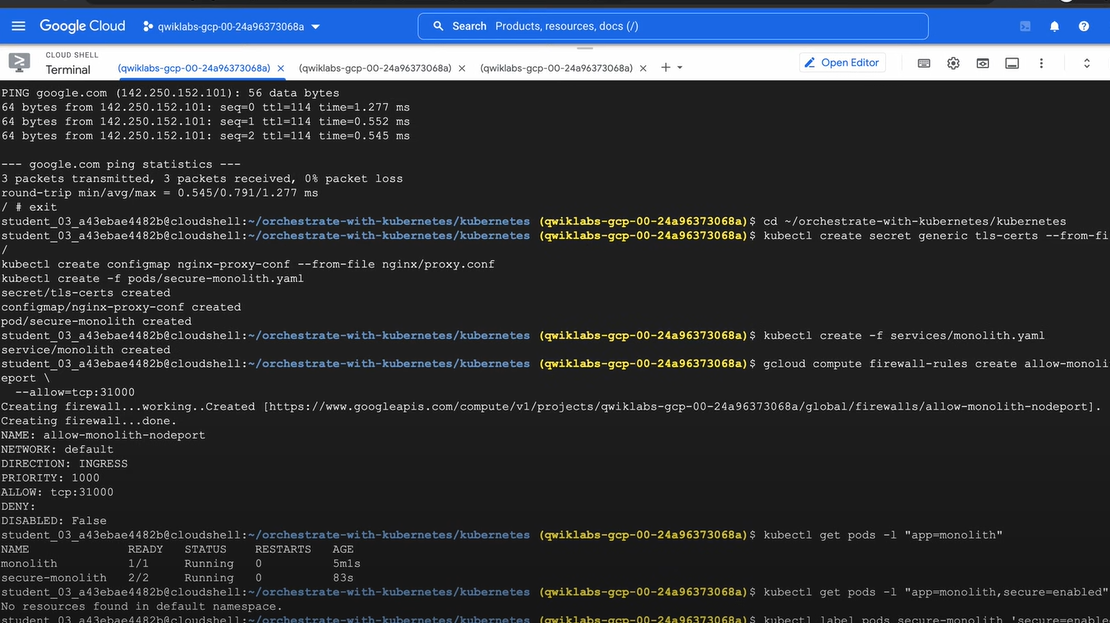
kubectl get services frontend

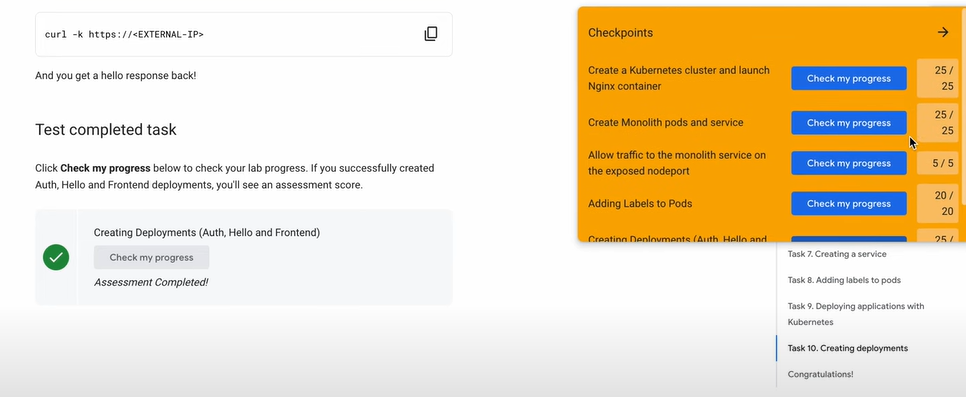
curl -k https://<EXTERNAL-IP>

**OUTPUT**



****

****

****

**CONCLUSION**

In conclusion, "Orchestrating the Cloud with Kubernetes" is a comprehensive guide to Kubernetes and its application to cloud-native application management on Google Cloud Platform (GCP). The book provides a practical and hands-on approach to learning Kubernetes, with a focus on real-world examples and exercises.

By following the methodology outlined in the book, readers can gain a thorough understanding of Kubernetes and its application to managing containerized applications on GCP. They can learn how to deploy and manage containerized applications, scale applications, and roll out updates, as well as integrate Kubernetes with other GCP services such as Google Cloud Storage, Google Cloud Load Balancing, and Google Cloud Logging.

"Orchestrating the Cloud with Kubernetes" is an essential resource for developers and IT professionals who are looking to build and manage cloud-native applications on GCP using Kubernetes. The book provides a practical and comprehensive guide to Kubernetes and its application to cloud-native application management on GCP, and is sure to be a valuable resource for anyone looking to work with Kubernetes in a cloud environment.

**FUTURE WORKS**

As Kubernetes continues to evolve and gain popularity as a container orchestration tool, there are several areas of future work for "Orchestrating the Cloud with Kubernetes" lab that can be explored, including:

1. Kubernetes on multi-cloud: While the book focuses on using Kubernetes on Google Cloud Platform, there is increasing interest in deploying Kubernetes across multiple cloud providers. Future work could explore best practices and strategies for deploying Kubernetes on multiple cloud providers and managing hybrid cloud environments.

2. Kubernetes for machine learning: Kubernetes has become an increasingly popular tool for managing machine learning workloads. Future work could explore how to deploy and manage machine learning workloads on Kubernetes, including integration with popular machine learning frameworks such as TensorFlow and PyTorch.

3. Kubernetes for edge computing: As more computing moves to the edge, there is a growing need for Kubernetes to be deployed on edge devices. Future work could explore how to deploy and manage Kubernetes on edge devices, as well as how to manage Kubernetes workloads that span both cloud and edge environments.

4. Security in Kubernetes: Kubernetes has a rich set of security features, but there is still room for improvement. Future work could explore best practices and strategies for securing Kubernetes clusters, including how to integrate Kubernetes with popular security tools such as Istio and Envoy.

**REFERENCES**

Google Docs

<https://kubernetes.io/>

Google Cloud Boost

Aws docs

<https://docs.aws.amazon.com/>

CloudGuru

<https://acloudguru.com/>

Medium

<https://medium.com/>