**Capstone Project**

**Infra Optimization**

**Problem Statement**:

A popular payment application, **EasyPay**where users add money to their wallet accounts, faces an issue in its payment success rate. The timeout that occurs with the connectivity of the database has been the reason for the issue. While troubleshooting, it is found that the database server has several downtime instances at irregular intervals.

This situation compels the company to create their own infrastructure that runs in high-availability mode. Given that online shopping experiences continue to evolve as per customer expectations, the developers are driven to make their app more reliable, fast, and secure for improving the performance of the current system.

**Objective:**

* To create a cluster with EC2 instances attaching the load balancer and elastic IP
* Automate the provisioning of an EC2 instance using Terraform
* Install Docker and Kubernetes on the cluster
* Implement the network policies at the database pod to allow ingress traffic from the front-end application pod
* Create a new user with permissions to create, list, get, update, and delete pods
* Configure application on the pod
* Take snapshot of ETCD database
* Set criteria such that if the memory of CPU goes beyond 50%, environments automatically get scaled up and configured

**Methodology:**

# I. Using Terraform to create the infrastructure

Terraform is a platform-agnostic tool that allows you to build, change, and version infrastructure securely and efficiently. Terraform is an open-source tool and a Hashicorp-developed multi-cloud, infrastructure-as-code solution that uses the declarative Hashicorp Configuration Language.

Created the infrastructure with below resources:

* VPC
* Subnets
* Internet Gateway
* EC2 Instances
* Elastic IP
* Security Groups
* Application Load Balancer

# Steps:

## Installed Visual Studio Code app to use Terraform.

## Install awscli on the local machine

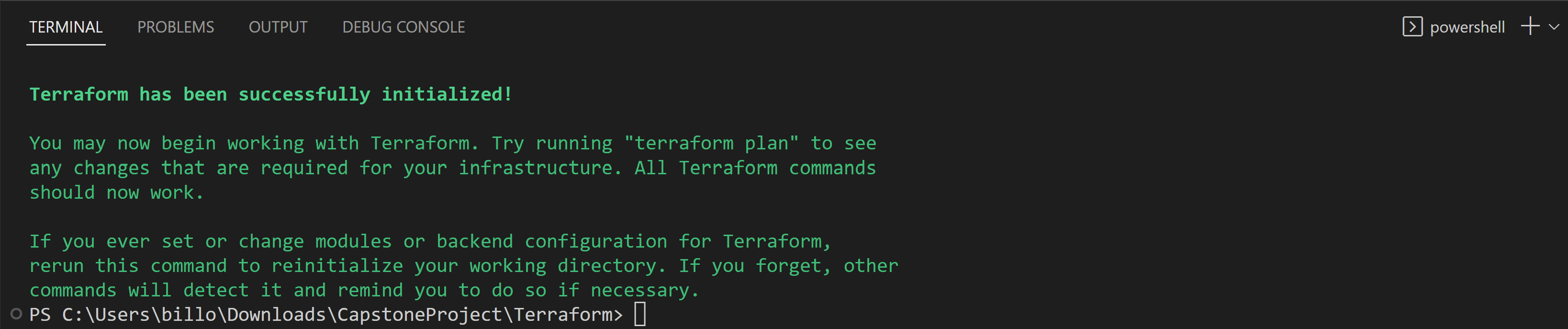


## Code Structure

Graphical user interface, text, application

Description automatically generated

## Initialize terraform:



**terraform init** is a command used in Terraform, an infrastructure as code (IaC) tool, to initialize a new or existing Terraform working directory.

This command must be run before any other Terraform commands in a given directory, as it ensures that the necessary dependencies and plugins are installed and available.

If you have an existing Terraform working directory, running terraform init can also update the provider plugins to the latest version and update the backend configuration if needed.

## Create an execution plan

Text

Description automatically generatedGraphical user interface, logo

Description automatically generated

**terraform plan** is a command used in Terraform, an infrastructure as code (IaC) tool, to create an execution plan for the infrastructure changes that are defined in your Terraform configuration files.

## Creating resources

Text

Description automatically generated

**Text

Description automatically generatedterraform apply** is a command used in Terraform, an infrastructure as code (IaC) tool, to apply the changes to the infrastructure that are defined in your Terraform configuration files. Before making any changes, Terraform will prompt you to confirm that you want to proceed with the changes. This helps ensure that you are aware of the changes that will be made to your infrastructure, and can prevent accidental or unwanted changes.

**terraform apply** is a critical command in the Terraform workflow, as it is used to actually apply the changes to the infrastructure. It is important to use this command with caution, and to thoroughly review the plan generated by terraform plan before applying any changes.

By now we have created all the resources that are needed for our execution.

# II. Using Ansible to Manage Infrastructure Configuration

Ansible is an open-source automation tool that is used for configuration management, application deployment, and orchestration. It is designed to make it easy to automate the deployment and management of complex IT infrastructure, including servers, applications, and networks.

Ansible uses a simple and human-readable YAML syntax to define automation tasks, making it easy to write and maintain playbooks (sets of tasks). Ansible can be used to automate a wide range of tasks, from simple software installations to complex, multi-tier applications.

# Steps:

## Using the Capstone DevOps AWS instance where I can use Ansible.

## Adding the Ansible hosts in /etc/ansible/hosts file.

1. Add the SSH key to all the instances in AWS.On all the nodes



Text

Description automatically generated

Ansible uses YAML files to define its automation tasks, which are called "playbooks". YAML is a human-readable data serialization language that is commonly used for configuration files and data exchange between applications.

In Ansible, YAML files are used to define the tasks that should be performed on the managed nodes, as well as any required configuration data or variables. Playbooks consist of a series of tasks, each of which defines the action to be performed, the target hosts or groups of hosts, and any required parameters or variables.

1. Playbook to install docker, kubeadm, kubectl:

Table

Description automatically generatedTable

Description automatically generated

1. This playbook installs docker, kubectl and kubeadm.

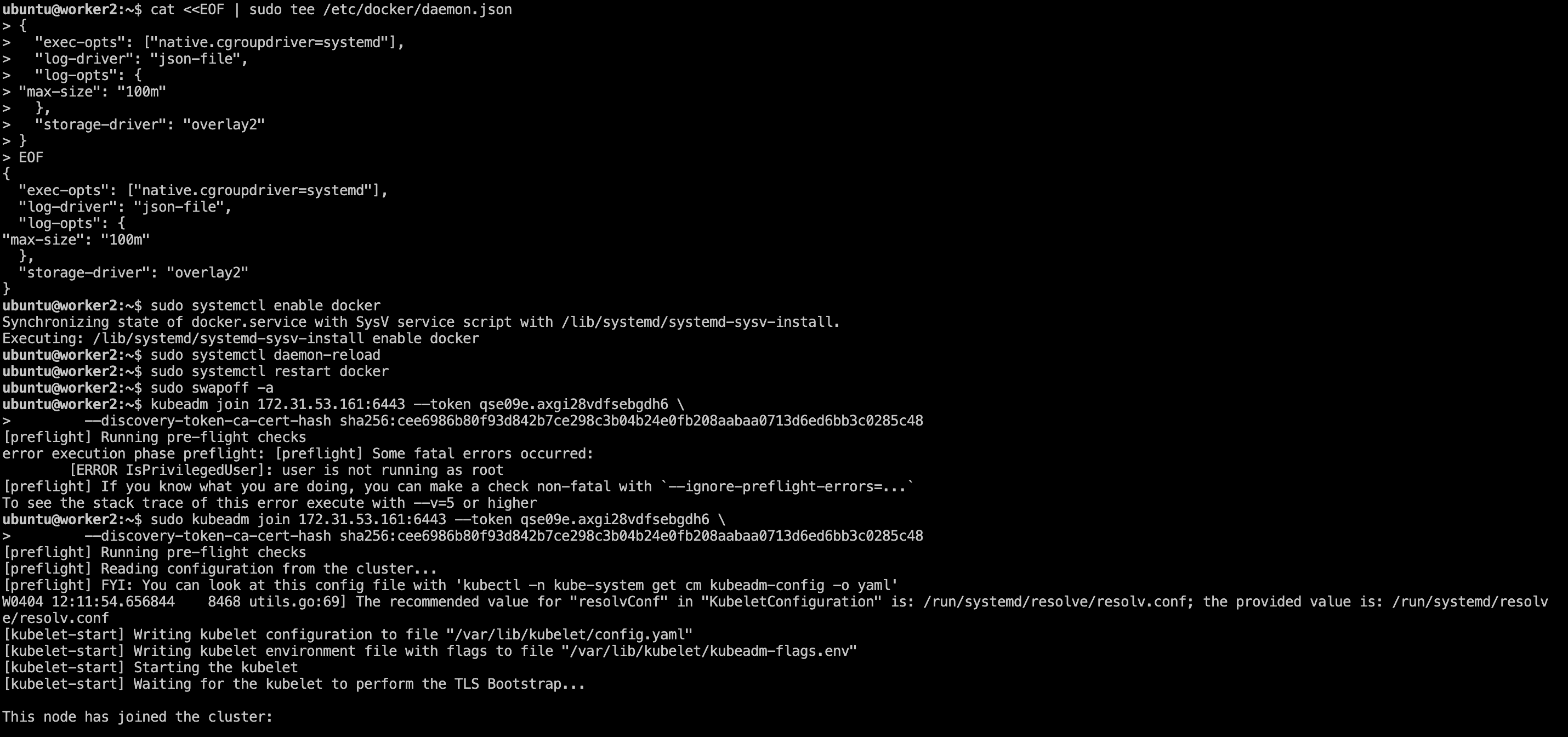
## Setting up and initializing the cluster in maser node

## Text Description automatically generated

Text

Description automatically generated

## Joining the worker nodes.



# III. Using Kubernetes to Orchestrate Containers

Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. It provides a way to automate the deployment, scaling, and management of containerized applications, making it easier to manage and deploy applications in a containerized environment.

With Kubernetes, you can deploy and manage containerized applications across a cluster of servers, with features such as automatic scaling, self-healing, and load balancing.

Kubernetes was originally developed by Google, and is now maintained by the Cloud Native Computing Foundation (CNCF). Kubernetes allows developers to easily deploy, manage, and scale containerized applications across a cluster of servers.

It provides features such as automatic load balancing, self-healing, and horizontal scaling, which make it a popular choice for managing containerized workloads in production environments.

Kubernetes has become the de facto standard for container orchestration and is widely used in production environments for deploying and managing microservices-based architectures.

# Steps:

1. Check whether the cluster is running properly

Graphical user interface

Description automatically generated with low confidence

Now that we got everything set to deploy our application, let’s install the dependencies to package our application code.

1. Installing Java

Text

Description automatically generated

1. Install maven

Text, timeline

Description automatically generated

Text

Description automatically generated

1. Install Git

Text

Description automatically generated with medium confidence

1. Now the git repository where the application code exists

Text

Description automatically generated

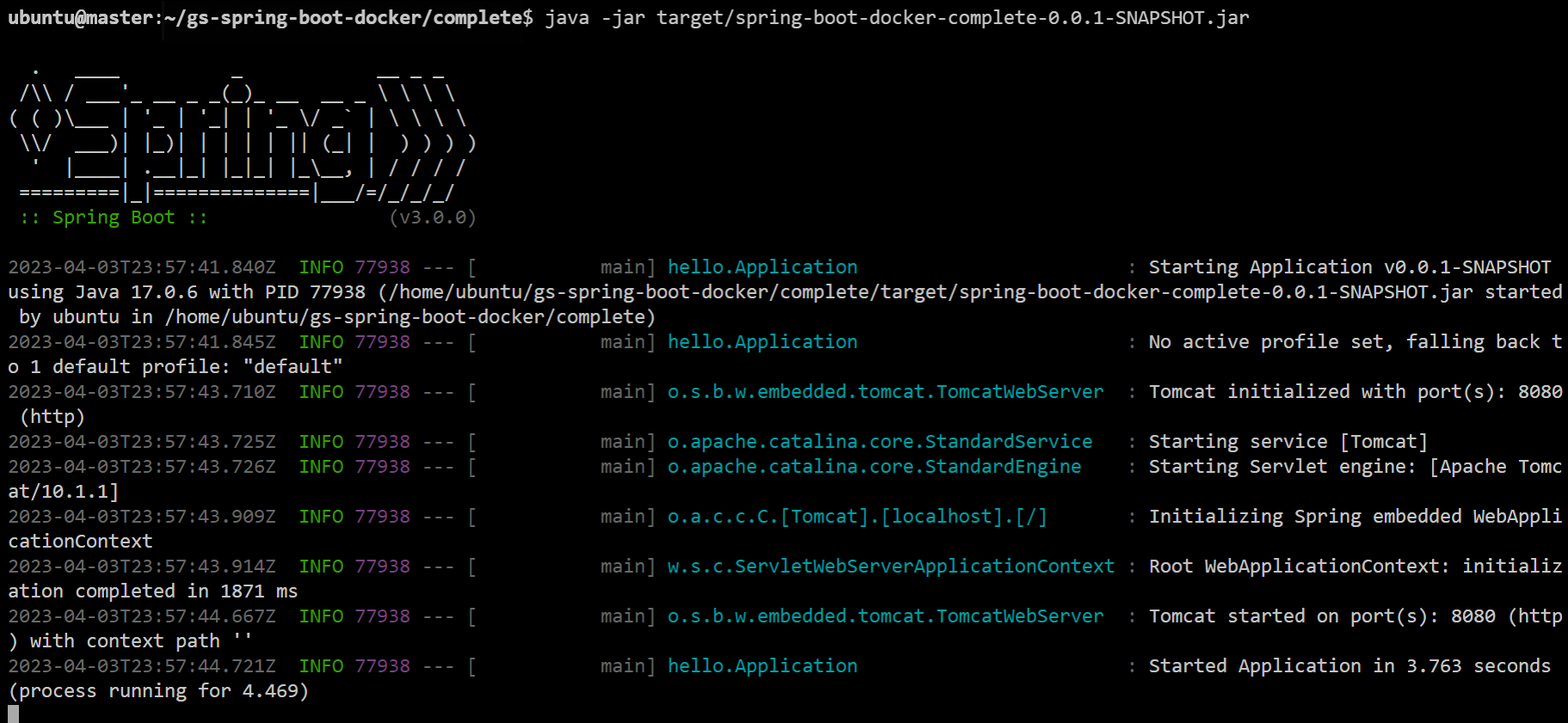
1. Now build a jar file



Text

Description automatically generated

1. Now run the java application



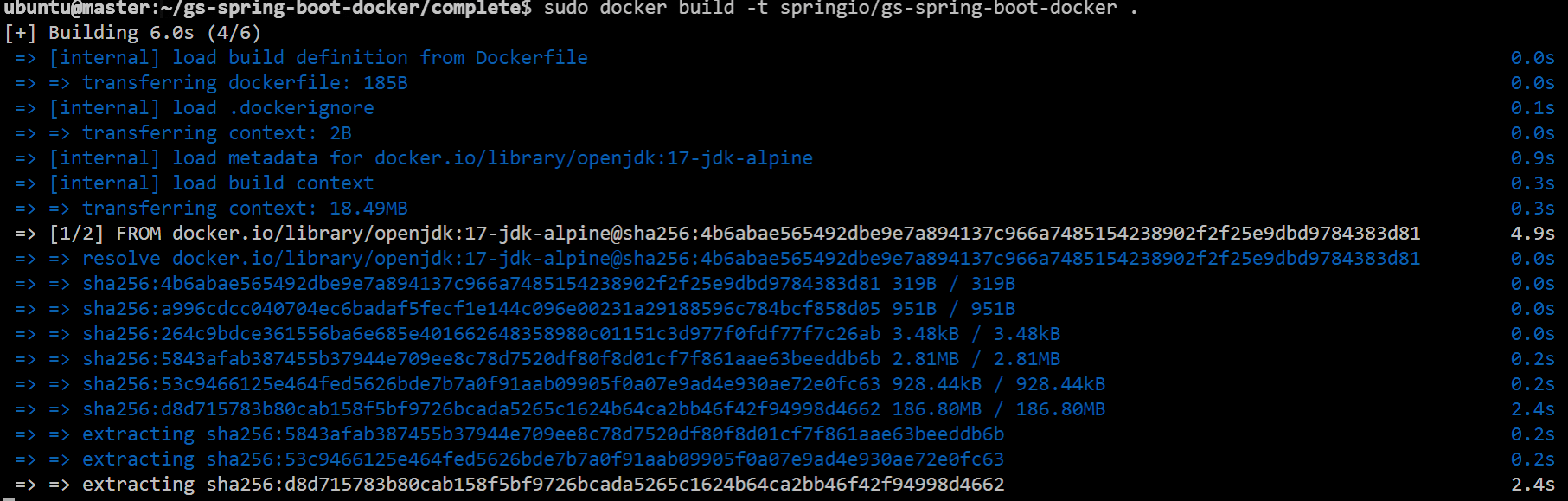
Now that our java application is running, let’s create a Dockerfile.

e. Creating a Dockerfile

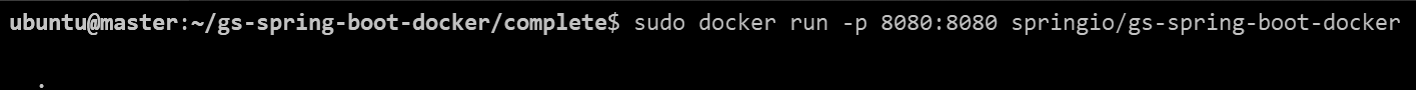
Text

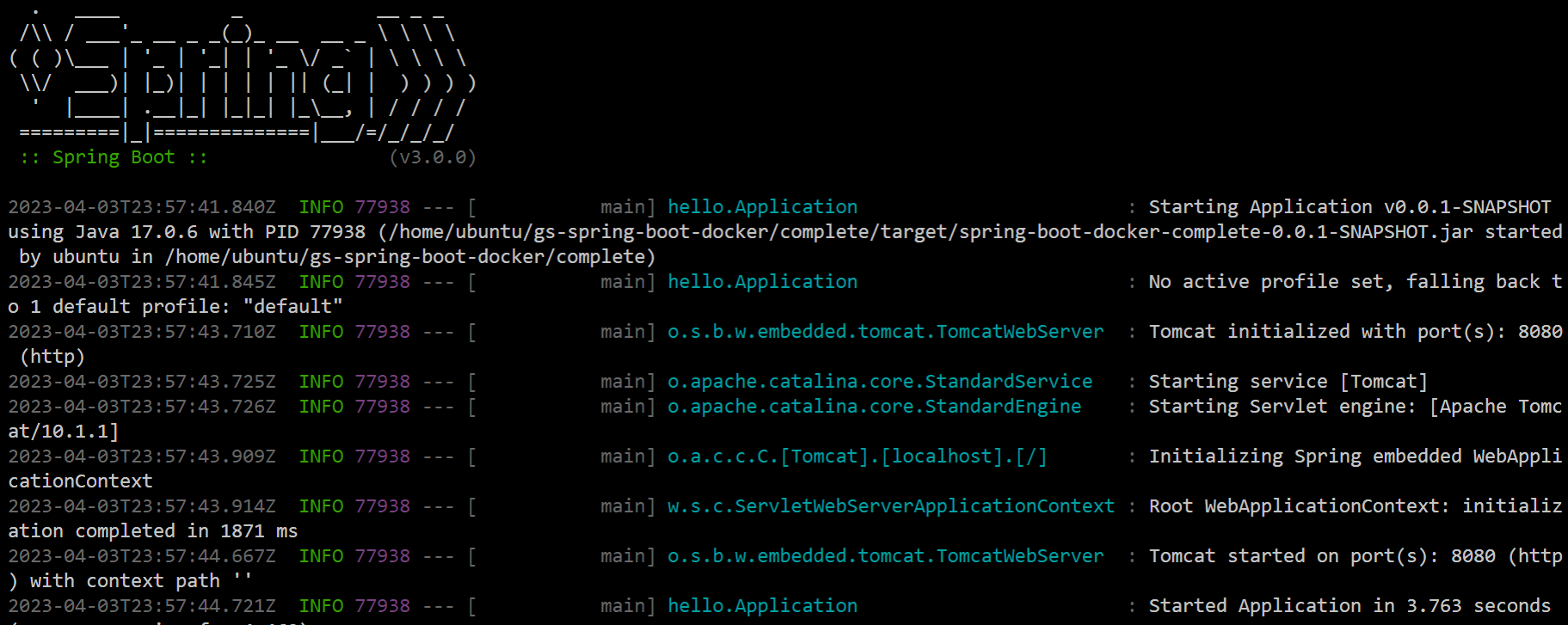
Description automatically generated

1. Building a docker image



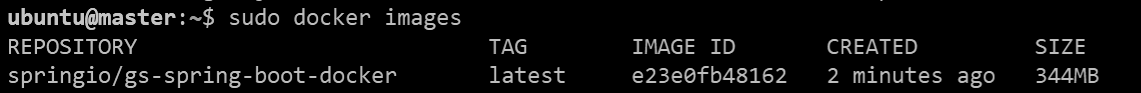
1. Running that docker image:





Now that the application is running inside the container, we can proceed to put the container to run in a pod.

1. Check the docker image name or ID:



Now that we have the docker images ready, shall run the pod with that image.

1. Creating a namespace capstone

Text

Description automatically generated

This namespace is for java application and database pods to run.

## Creating a namespace mynamespace

Text

Description automatically generated

This name space is created to show that resources in this namespace will not be able to access the

database pods.

## Creating a dummy pod

Text

Description automatically generated

## Creating application pod

Text

Description automatically generated

This deployment across cluster with specific labels is used by the network policy, and prove that pods within the capstone namespace can communicate with each other.

## Creating a database pod

Text

Description automatically generated

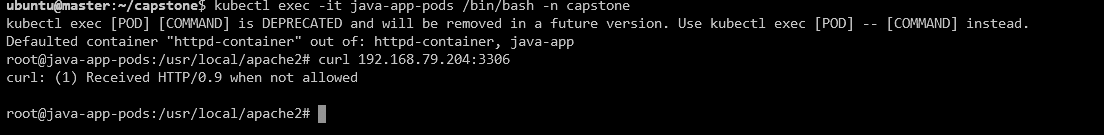
## Now checking the pods:

Text

Description automatically generated

As we can see all the containers are running inside the pods. Now we shall test the network and connectivity of the pods.

## Before applying the network policy check whether java-app-pods are communicating with database pods



## Also, mypod is communicating with database pods without the network policy.

Text

Description automatically generated

## Applying the network policy:

Text

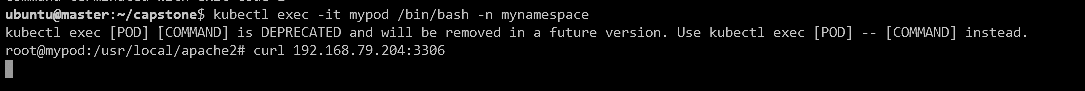
Description automatically generated

After applying the network policy, java-app-pods are able to communicate with database pods.

Text

Description automatically generated

After applying the network policy, mypod is not able to communicate with the database pods.



This confirms that the after applying the network policy only the application pod is able to communicate with database pod.

## Creating a new user

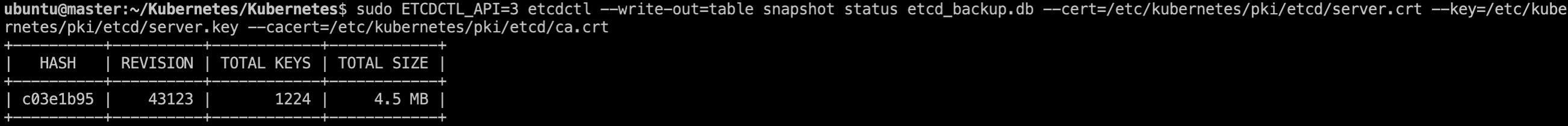
Text

Description automatically generated

This confirms that user david is able to only create, get, list, delete and update the pods. Any other resources access is forbidden for david.

## Take a snapshot of the ETCD

The ETCD snapshot is take and is saved in /tmp/myback



## To check the CPU utilization, lets install the metrics-server

Text

Description automatically generated

As the metrics server is installed, let’s check the utilization.

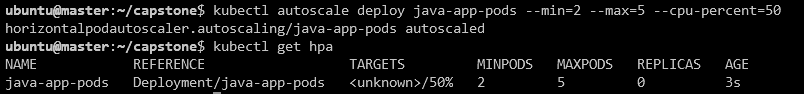
## Check the utilization

Text

Description automatically generated

## Now let’s set the criteria to be automatically be scaled up with the utilization goes beyond 50%.

## Horizontal Autoscaling



So, whenever the CPU utilization goes beyond 50% automatically it will be scaled up to 5 pods, if the utilization below 50% it will just the two pods running.

This completes our Capstone Project!