Capstone Project 1

Hatem El-Sebaaly

Project Description

DESCRIPTION

Create a DevOps infrastructure for an e-commerce application to run on high-availability mode.

Background of the 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.

Implementation requirements:

1. Create the cluster (EC2 instances with load balancer and elastic IP in case of AWS)
2. Automate the provisioning of an EC2 instance using Ansible or Chef Puppet
3. Install Docker and Kubernetes on the cluster
4. Implement the network policies at the database pod to allow ingress traffic from the front-end application pod
5. Create a new user with permissions to create, list, get, update, and delete pods
6. Configure application on the pod
7. Take snapshot of ETCD database
8. Set criteria such that if the memory of CPU goes beyond 50%, environments automatically get scaled up and configured

The following tools must be used:

Project Requirement

The following things to be kept in check:

1. You need to document the steps and write the algorithms in them.
2. The submission of your GitHub repository link is mandatory. In order to track your tasks, you need to share the link of the repository.
3. Document the step-by-step process starting from creating test cases, then executing them, and recording the results.
4. You need to submit the final specification document, which includes:

- Project and tester details
- Concepts used in the project
- Links to the GitHub repository to verify the project completion
- Your conclusion on enhancing the application and defining the USPs (Unique Selling Points)

Approach

- A Spring Data Rest simple application will be used for testing purposes
  - The application is called EasyPay
  - It allows you to use REST to add wallet and add dollars to add transaction amount in the balance
  - **https://github.com/hatemhobbies/easy_pay_rest_data_sample_postgres.git**
- Terraform is used to provision the infrastructure
  - **https://github.com/hatemhobbies/terraform_k8s_aws_postgres_spring_ansible.git**
  - **https://github.com/hatemhobbies/terraform_k8s_aws_postgres_spring_ansible.git/terraform**
- Ansible is used to install Kubernetes Controller and workers
  - One controller node is installed
  - The Kubernetes metrics are installed by ansible
  - The number of worker nodes is configurable
  - **https://github.com/hatemhobbies/terraform_k8s_aws_postgres_spring_ansible.git/ansible**
- Kubernetes
  - Role based access control will be added to showcase a pod-admin user
  - ETCD will be backed up in a snapshot
  - **https://github.com/hatemhobbies/terraform_k8s_aws_postgres_spring_ansible.git/k8s**

Prerequisite

To run terraform - you must have a shell script (e.g. aws_key.sh) with the following content (Replace with your key - you may need a token as well - please review AWS)

#!/bin/sh

export AWS_ACCESS_KEY_ID=ZZZZS7DTTTTTJXEEAAAA

export AWS_SECRET_ACCESS_KEY=8xxVAte+xOQ780ydtUDGat54GlgsNZrhaALAfhuL

export AWS_REGION=us-east-2

The EasyPay Spring Data Rest Application
(Description)

https://github.com/hatemhobbies/easy_pay_rest_data_sample_postgres.git

Description:

The application uses Spring (Boot, Data, Data Rest), maven, postgres docker and docker-compose.

Functionally, it exposes REST APIs with the following features

/wallets - used to create a wallet

/transactions - used to fill add /remove transaction from the wallet

Section 1 - Review of the application easypayservice (walletservice)

The EasyPay Spring Data Rest Application

(clone the github repository)

mkdir capstone_project1

cd capstone_project1

git clone https://github.com/hatemhobbies/easy_pay_rest_data_sample_postgres.git

```
hatem@Hatems-Priv-MBP ~ % cd capstone_project1
hatem@Hatems-Priv-MBP capstone_project1 % git clone https://github.com/hatemhobbies/easy_pay_rest_data_sample_postgres.git
Cloning into 'easy_pay_rest_data_sample_postgres'...
remote: Enumerating objects: 52, done.
remote: Counting objects: 100% (52/52), done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 52 (delta 15), reused 31 (delta 3), pack-reused 0
Receiving objects: 100% (52/52), 15.99 KiB | 5.33 MiB/s, done.
Resolving deltas: 100% (15/15), done.
hatem@Hatems-Priv-MBP capstone_project1 %
```

The EasyPay Spring Data Rest Application
(build the application)

mvn clean install -DskipTests

```
hatem@Hatems-Priv-MBP easy_pay_rest_data_sample_postgres % mvn clean install -skipTest=true
[ERROR] Error executing Maven.
[ERROR] The specified user settings file does not exist: /Users/hatem/capstone_project1/easy_pay_rest_data_sample_postgres/kipTest=true
hatem@Hatems-Priv-MBP easy_pay_rest_data_sample_postgres % mvn clean install  -DskipTests
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< com.easypay:wallet >------------------------
[INFO] Building wallet 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:3.2.0:clean (default-clean) @ wallet ---
[INFO] Deleting /Users/hatem/capstone_project1/easy_pay_rest_data_sample_postgres/target
[INFO]
[INFO] --- maven-resources-plugin:3.2.0:resources (default-resources) @ wallet ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Using 'UTF-8' encoding to copy filtered properties files.
[INFO] Copying 1 resource
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.10.1:compile (default-compile) @ wallet ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 5 source files to /Users/hatem/capstone_project1/easy_pay_rest_data_sample_postgres/target/classes
[INFO]
[INFO] --- maven-resources-plugin:3.2.0:testResources (default-testResources) @ wallet ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Using 'UTF-8' encoding to copy filtered properties files.
[INFO] Copying 1 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.10.1:testCompile (default-testCompile) @ wallet ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /Users/hatem/capstone_project1/easy_pay_rest_data_sample_postgres/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ wallet ---
[INFO] Tests are skipped.
[INFO]
[INFO] --- maven-jar-plugin:3.2.2:jar (default-jar) @ wallet ---
[INFO] Building jar: /Users/hatem/capstone_project1/easy_pay_rest_data_sample_postgres/target/wallet-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:2.7.5:repackage (repackage) @ wallet ---
[INFO] Replacing main artifact with repackaged archive
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ wallet ---
[INFO] Installing /Users/hatem/capstone_project1/easy_pay_rest_data_sample_postgres/target/wallet-0.0.1-SNAPSHOT.jar to /Users/hatem/.m
2/repository/com/easypay/wallet/0.0.1-SNAPSHOT/wallet-0.0.1-SNAPSHOT.jar
[INFO] Installing /Users/hatem/capstone_project1/easy_pay_rest_data_sample_postgres/pom.xml to /Users/hatem/.m2/repository/com/easypay/
wallet/0.0.1-SNAPSHOT/wallet-0.0.1-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.440 s
[INFO] Finished at: 2022-11-05T11:36:59-05:00
[INFO] ------------------------------------------------------------------------
hatem@Hatems-Priv-MBP easy_pay_rest_data_sample_postgres %
```

The EasyPay Spring Data Rest Application

(creating and publishing the docker image)

docker login

docker build . -t hatemhobbies/wallet:2.0

docker push hatemhobbies/wallet:2.0

```
hatem@Hatems-Priv-MBP easy_pay_rest_data_sample_postgres % docker login
docker build . -t hatemhobbies/wallet:2.0
docker push  hatemhobbies/wallet:2.0

Authenticating with existing credentials...
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/
[+] Building 1.7s (9/9) FINISHED
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 228B                                                                                0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [internal] load metadata for docker.io/library/openjdk:17-alpine                                                0.8s
 => [auth] library/openjdk:pull token for registry-1.docker.io                                                      0.0s
 => [internal] load build context                                                                                   0.4s
 => => transferring context: 40.11MB                                                                                0.4s
 => [1/3] FROM docker.io/library/openjdk:17-alpine@sha256:4b6abae565492dbe9e7a894137c966a7485154238902f2f25e9dbd9784383d81  0.0s
 => CACHED [2/3] RUN addgroup -S spring && adduser -S spring -G spring                                              0.0s
 => [3/3] COPY target/*.jar app.jar                                                                                 0.3s
 => exporting to image                                                                                              0.2s
 => => exporting layers                                                                                             0.2s
 => => writing image sha256:37c3014d9f58292ff18854ef2a8afc461531d917e6be240e5140a2ee1ace9aeb                        0.0s
 => => naming to docker.io/hatemhobbies/wallet:2.0                                                                  0.0s
The push refers to repository [docker.io/hatemhobbies/wallet]
3aa825d5d1fd: Pushed
2bc0e879a282: Layer already exists
34f7184834b2: Layer already exists
5836ece05bfd: Layer already exists
72e830a4dff5: Layer already exists
2.0: digest: sha256:9d2bcd0e71d6cf09e65ffb34b7881410cd911c70e57832b78c8461ddb43173c9 size: 1371
hatem@Hatems-Priv-MBP easy_pay_rest_data_sample_postgres %
```

The EasyPay Spring Data Rest Application

(Docker Compose for image testing)

We will use docker compose up and let it run postgres

and the app. Not the use of the environment variables in Spring

And Docker to specify the host, user and password.

```
# ==============================
# = DATA SOURCE
# ==============================
# Set here configurations for the database connection
spring.datasource.url=jdbc:postgresql://${POSTGRES_HOST:db}:5432/postgres
spring.datasource.username=${POSTGRES_USER:postgres}
spring.datasource.password=${POSTGRES_PASSWORD:postgres}
spring.datasource.driver-class-name=org.postgresql.Driver
# Keep the connection alive if idle for a long time (needed in production)
spring.datasource.testWhileIdle=true
spring.datasource.validationQuery=SELECT 1
# ==============================
# = JPA / HIBERNATE
```

28 lines (28 sloc) | 611 Bytes

```
version: '3.8'
services:
  db:
    image: postgres:14.1-alpine
    restart: always
    environment:
      - POSTGRES_HOST=localhost
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
    ports:
      - '5432:5432'
    volumes:
      - ./postgres-data:/var/lib/postgresql/data
    container_name: eppostgres
  wallet-app:
    image: hatemhobbies/wallet-service:2.0
    build:
      context: ./
      dockerfile: Dockerfile
    environment:
      -  POSTGRES_HOST=db
    depends_on:
      - db
    ports:
      - 8080:8080
```

The EasyPay Spring Data Rest Application

(Starting Docker compose)

docker compose up

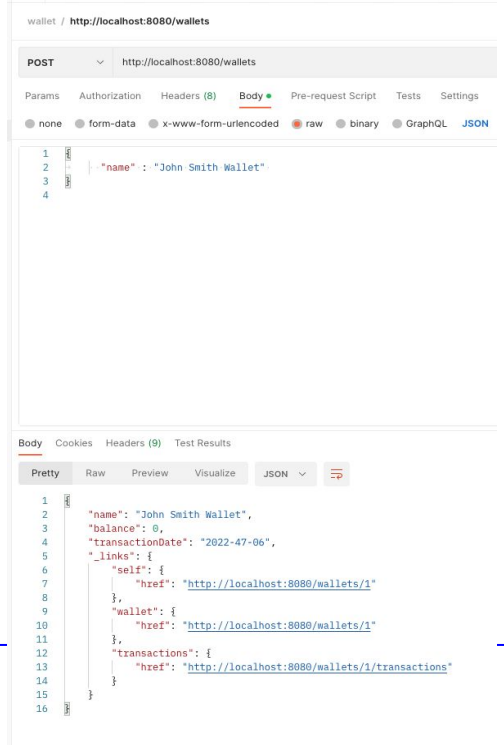
```
hatem@Hatems-Priv-MBP easy_pay_rest_data_sample_postgres % docker-compose up
[+] Running 9/10
 ⠿ wallet-app Error
 ⠿ db Pulled
   ⠿ 59bf1c3509f3 Pull complete
   ⠿ c50e01d57241 Pull complete
   ⠿ a0646b0f1ead Pull complete
   ⠿ 7433e5151e0c Pull complete
   ⠿ 8854018388d9 Pull complete
   ⠿ 8de463f7fd19 Pull complete
   ⠿ b39ee18abab9 Pull complete
   ⠿ 11d7473a0ff9 Pull complete
[+] Building 1.4s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 223B
```

```
=> => naming to docker.io/hatemhobbies/wallet-service:2.0
+] Running 3/3
 ⠿ Network easy_pay_rest_data_sample_postgres_default  Created
 ⠿ Container eppostgres                               Created
 ⠿ Container walletservice                            Created
ttaching to eppostgres, walletservice
ppostgres      | The files belonging to this database system will be owned by user "p
ppostgres      | This user must also own the server process.
ppostgres      |
ppostgres      | The database cluster will be initialized with locale "en_US.utf8".
ppostgres      | The default database encoding has accordingly been set to "UTF8".
ppostgres      | The default text search configuration will be set to "english".
ppostgres      |
ppostgres      | Data page checksums are disabled.
ppostgres      |
ppostgres      | fixing permissions on existing directory /var/lib/postgresql/data ..
ppostgres      | creating subdirectories ... ok
ppostgres      | selecting dynamic shared memory implementation ... posix
ppostgres      | selecting default max_connections ... 100
ppostgres      | selecting default shared_buffers ... 128MB
ppostgres      | selecting default time zone ... UTC
ppostgres      | creating configuration files ... ok
alletservice   |
alletservice   |   .   ____          _            __ _ _
alletservice   |  /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
alletservice   | ( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
alletservice   |  \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
alletservice   |   '  |____| .__|_| |_|_| |_\__, | / / / /
alletservice   |  =========|_|==============|___/=/_/_/_/
alletservice   |  :: Spring Boot ::                (v2.7.5)
alletservice   |
```

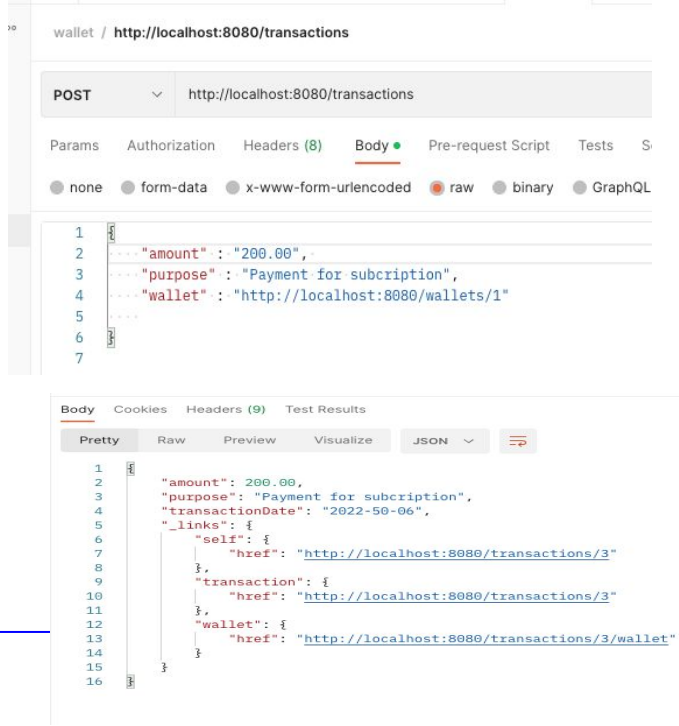
The EasyPay Spring Data Rest Application

(Testing using postman)

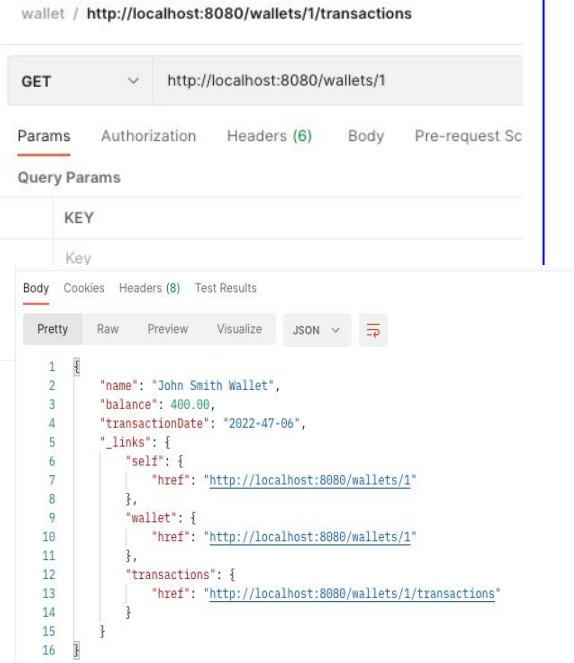
1) Create a wallet



2) Add two transactions for 200$ each



3) View wallet



Logs of inserts

```
ion in 2 ms
walletservice  | Hibernate: insert into "wallet" ("name", "transaction_date") values (?, ?)
walletservice  | Hibernate: insert into "transaction" ("amount", "purpose", "transaction_date", "wallet_id") values (?, ?, ?, ?)
walletservice  | Hibernate: select wallet0_."id" as id1_1_0_, wallet0_."name" as name2_1_0_, wallet0_."transaction_date" as transact3_1
_0_ from "wallet" wallet0_ where wallet0_."id"=?
walletservice  | Hibernate: select transactio0_."wallet_id" as wallet_i5_0_0_, transactio0_."id" as id1_0_0_, transactio0_."id" as id1
0_1_, transactio0_."amount" as amount2_0_1_, transactio0_."purpose" as purpose3_0_1_, transactio0_."transaction_date" as transact4_0_1
, transactio0_."wallet_id" as wallet_i5_0_1_ from "transaction" transactio0_ where transactio0_."wallet_id"=?
walletservice  | Hibernate: select wallet0_."id" as id1_1_0_, wallet0_."name" as name2_1_0_, wallet0_."transaction_date" as transact3_1
_0_ from "wallet" wallet0_ where wallet0_."id"=?
walletservice  | Hibernate: select transactio0_."wallet_id" as wallet_i5_0_0_, transactio0_."id" as id1_0_0_, transactio0_."id" as id1
0_1_, transactio0_."amount" as amount2_0_1_, transactio0_."purpose" as purpose3_0_1_, transactio0_."transaction_date" as transact4_0_1
, transactio0_."wallet_id" as wallet_i5_0_1_ from "transaction" transactio0_ where transactio0_."wallet_id"=?
walletservice  | Hibernate: insert into "transaction" ("amount", "purpose", "transaction_date", "wallet_id") values (?, ?, ?, ?)
walletservice  | Hibernate: select wallet0_."id" as id1_1_0_, wallet0_."name" as name2_1_0_, wallet0_."transaction_date" as transact3_1
```

Automated Integration Testing (using Mockito)

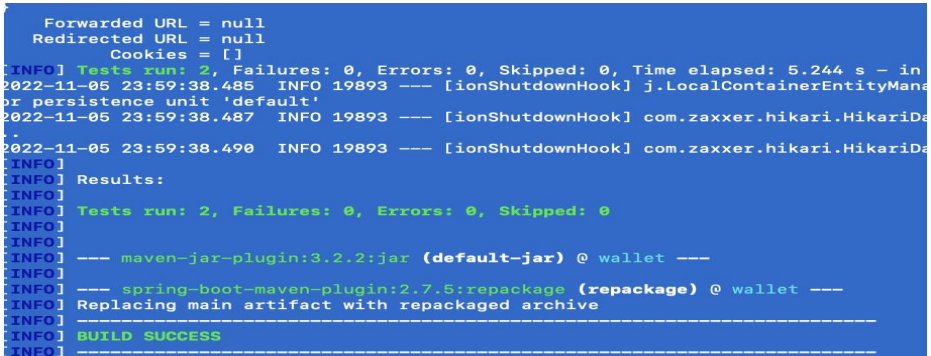
See the file

https://github.com/hatemhobbies/easy_pay_rest_data_sample_postgres/blob/main/src/test/java/com/easypay/wallet/WalletApplicationTests.java

Start the postgres container using

docker run --name empostgres --rm -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postgres -e POSTGRES_DB=postgres -e PGDATA=/var/lib/postgresql/data/pgdata -v /tmp/data:/var/lib/postgresql/data -p 5432:5432 -d postgres:14.1-alpine

Simply do **maven package**



Section 2 - Using terraform and ansible to install Kubernetes +Kubernetes metrics and then deploy postgres and wallet service as pods

Preview the project

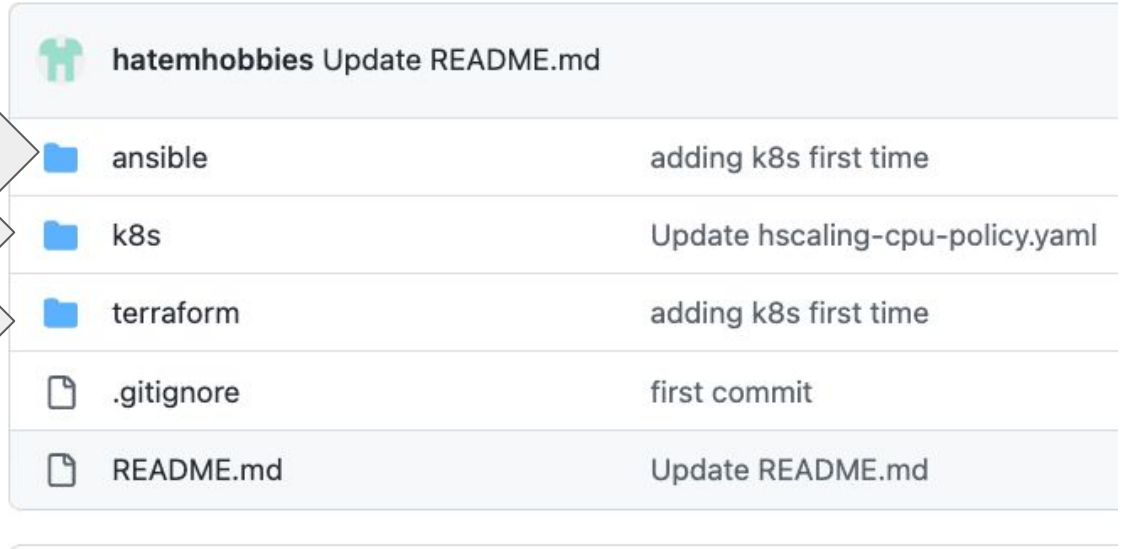
https://github.com/hatemhobbies/terraform_k8s_aws_postgres_spring_ansible.git

There are three important folders

Installs a controller and X number of worker nodes

Contains the yaml and scripts for k8s

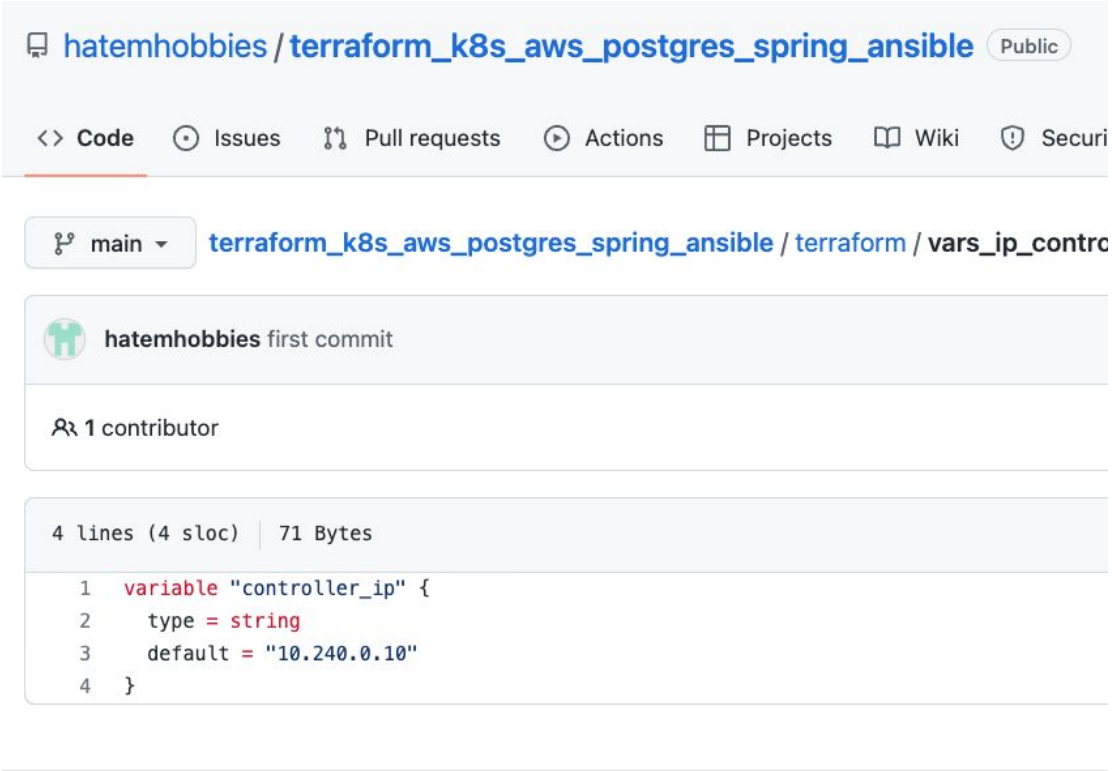
Provision the infra and launches ansible



Terraform folder

- admin_machine_setup.tf
- ami.tf
- aws_k8s_key.tf
- controller.tf
- load_balancer.tf
- main.tf
- sec_group.tf
- vars_ip_controller.tf
- vars_ip_pods.tf
- vars_ip_workers.tf
- vars_keys.tf
- vpc.tf
- workers.tf

The vars_ip_controller.tf controls the controller node ip



The vars_ip_workers.tf controls the worker nodes count

main ▾ terraform_k8s_aws_postgres_spring_ansible / terraform / vars_ip_workers.tf

hatemhobbies first commit

1 contributor

8 lines (7 sloc) | 103 Bytes

```
1  variable "worker_ips" {
2    type = list(string)
3
4    default = [
5      "10.240.0.20",
6      "10.240.0.21"
7    ]
8  }
```

controller.tf - provision the EC2 instance and launches ansible to install k8s cluster

main terraform_k8s_aws_postgres_spring_ansible / terraform / controller.tf

Go to file ···

hatemhobbies adding k8s first time — Latest commit 6c6a69d 4 days ago — History

1 contributor

39 lines (32 sloc) | 1.41 KB — Raw | Blame

```
resource "aws_instance" "k8s_controller" {
  depends_on                  = [aws_vpc.k8s]
  ami                         = "${data.aws_ami.ubuntu.id}"
  associate_public_ip_address = true
  key_name                    = "${aws_key_pair.k8s.key_name}"
  vpc_security_group_ids      = ["${aws_security_group.k8s.id}"]
  instance_type               = "t2.micro"
  private_ip                  = "${var.controller_ip}"
  user_data                   = "name=k8s_controller"
  subnet_id                   = "${aws_subnet.k8s.id}"
  source_dest_check           = false

  tags = {
    Name = "k8s_controller"
  }

  provisioner "remote-exec" {
    connection {
      host = self.public_ip
      user = "ubuntu"
      private_key = file("~/.ssh/k8s_rsa")
    }
    inline = ["echo 'Instance ${self.public_dns} is up!'"]
  }


  provisioner "local-exec" {
    command = <<EOT
      scp -o StrictHostKeyChecking=no  -i ~/.ssh/k8s_rsa*   ~/.ssh/k8s_rsa ubuntu@${aws_instance.k8s_controller.public_ip}:/home/ubuntu/.ssh
      scp -r -i ~/.ssh/k8s_rsa ../k8s  ubuntu@${aws_instance.k8s_controller.public_ip}:/home/ubuntu/k8s
    EOT
  }



  provisioner "local-exec" {
    command = "ANSIBLE_HOST_KEY_CHECKING=False ansible-playbook -T 300 -i '${self.public_ip},' --extra-vars 'private_ip=${self.private_ip} hostname=${split(".",
  }
}
```

The workers.tf builds the workers EC2 nodes and launch ansible to instal k8s and join th cluster

main terraform_k8s_aws_postgres_spring_ansible / terraform / workers.tf Go to file ···

hatemhobbies first commit Latest commit f19bece 5 days ago History

1 contributor

31 lines (27 sloc) | 1.13 KB Raw Blame

```
resource "aws_instance" "k8s_worker" {
  depends_on                  = [aws_instance.k8s_controller]
  count                       = "${length(var.worker_ips)}"
  ami                         = "${data.aws_ami.ubuntu.id}"
  associate_public_ip_address = true
  key_name                    = "${aws_key_pair.k8s.key_name}"
  vpc_security_group_ids      = ["${aws_security_group.k8s.id}"]
  instance_type               = "t2.micro"
  private_ip                  = "${var.worker_ips[count.index]}"
  user_data                   = "name=k8s_worker-${count.index+1}|pod-cidr=${var.worker_pod_cidrs[count.index]}"
  subnet_id                   = "${aws_subnet.k8s.id}"
  source_dest_check           = false

  tags = {
    Name = "k8s_worker-${count.index+1}"
  }

  provisioner "remote-exec" {
    connection {
      host = self.public_ip
      user = "ubuntu"
      private_key = file("~/.ssh/k8s_rsa")
    }

    inline = ["echo 'Instance ${self.public_dns} is up!'"]
  }

  provisioner "local-exec" {
    command = "ANSIBLE_HOST_KEY_CHECKING=False ansible-playbook -T 300 -i '${self.public_ip},' --private-key ~/.ssh/k8s_rsa ../ansible/k8s_node_playbook.yml"
  }
```

Important Notes - One button push does it all

1) Clone the repo from github

https://github.com/hatemhobbies/terraform_k8s_aws_postgres_spring_ansible.git

2) Make sure you have a private key / pub key pairs

ssh-keygen

- **Make sure not to require a password - name it k8s_rsa**
- **The public key generated will be k8s_rsa.pub and the private key is k8s_rsa**
- **Place these files in ~/.ssh**

3) cd terraform

4) terraform init

5) terraform apply -auto-approve

Auto - build infra, ansible installs k8s and then applies yaml

Output of **terraform apply -auto-approve**

1

```
hatem@Hatems-Priv-MBP terraform % source ../../aws_key.sh
hatem@Hatems-Priv-MBP terraform % pwd
/Users/hatem/Desktop/CapstoneProject/terraform_k8s_aws_postgres_spring_ansible/terraform
hatem@Hatems-Priv-MBP terraform %
```

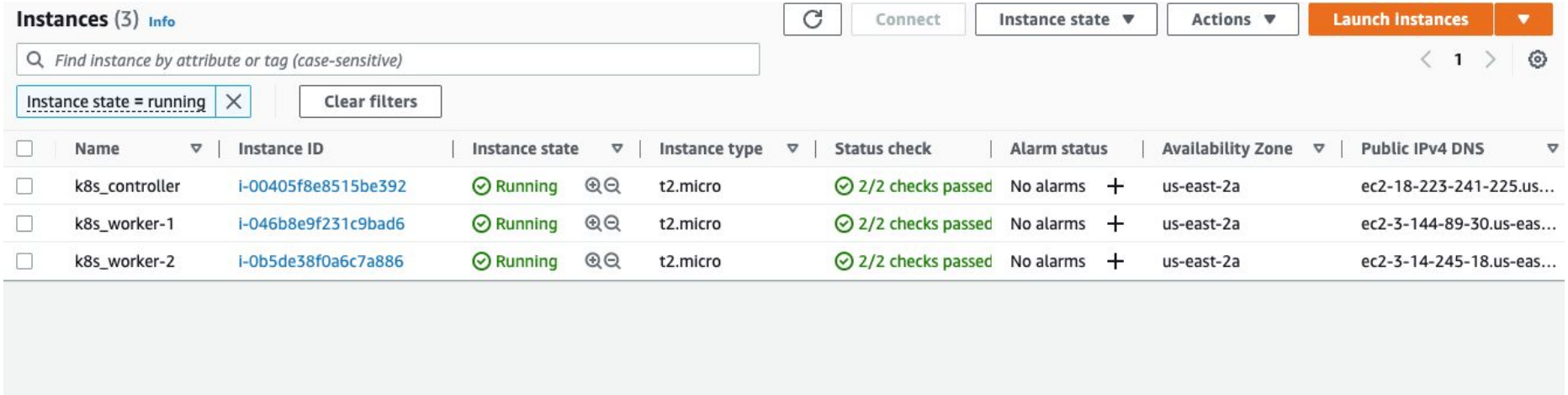
2

```
    + cidr_block                          = "10.240.0.0/24"
    + default_network_acl_id              = (known after apply)
    + default_route_table_id              = (known after apply)
    + default_security_group_id           = (known after apply)
    + dhcp_options_id                     = (known after apply)
    + enable_classiclink                  = (known after apply)
    + enable_classiclink_dns_support      = (known after apply)
    + enable_dns_hostnames                = true
    + enable_dns_support                  = true
    + id                                  = (known after apply)
    + instance_tenancy                    = "default"
    + ipv6_association_id                 = (known after apply)
    + ipv6_cidr_block                     = (known after apply)
    + ipv6_cidr_block_network_border_group = (known after apply)
    + main_route_table_id                 = (known after apply)
    + owner_id                            = (known after apply)
    + tags                                = {
        + "Name" = "kubernetes-simplilearn"
      }
    + tags_all                            = {
        + "Name" = "kubernetes-simplilearn"
      }
  }

# aws_vpc_dhcp_options.k8s will be created
+ resource "aws_vpc_dhcp_options" "k8s" {
    + arn             = (known after apply)
    + domain_name     = "us-east-2.compute.internal"
    + domain_name_servers = [
        + "AmazonProvidedDNS",
      ]
    + id              = (known after apply)
    + owner_id        = (known after apply)
    + tags            = {
        + "Name" = "kubernetes"
      }
    + tags_all        = {
        + "Name" = "kubernetes"
      }
  }

# aws_vpc_dhcp_options_association.k8s will be created
+ resource "aws_vpc_dhcp_options_association" "k8s" {
    + dhcp_options_id = (known after apply)
    + id              = (known after apply)
    + vpc_id          = (known after apply)
  }

# null_resource.fetch_kubeconfig will be created
+ resource "null_resource" "fetch_kubeconfig" {
    + id = (known after apply)
  }
```

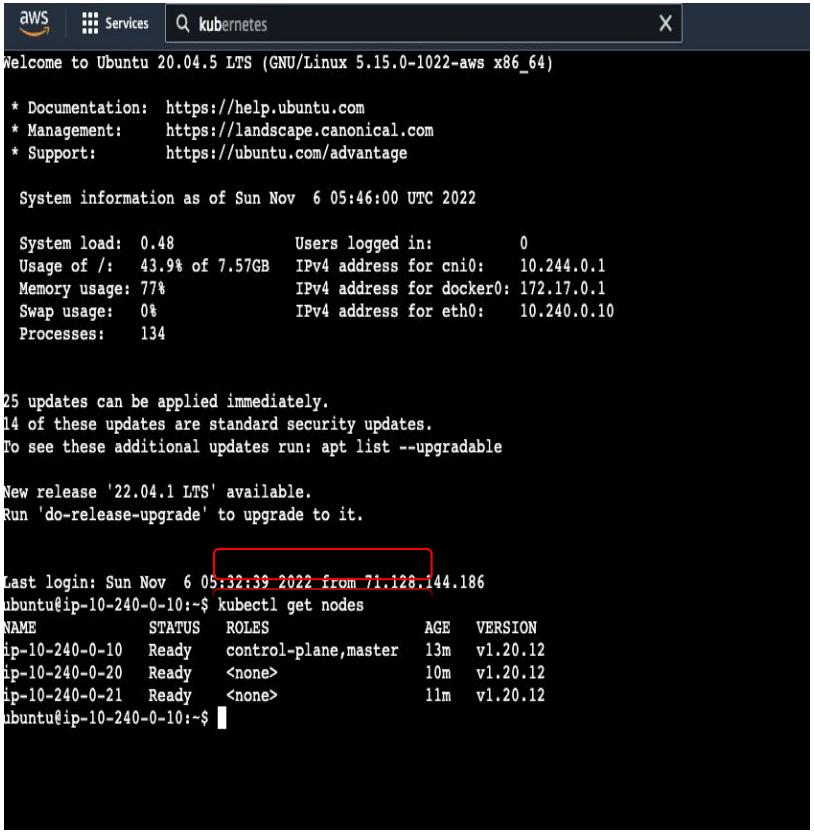
3

```
aws_instance.k8s_worker[0] (local-exec): RUNNING HANDLER [docker status] ********************************************
aws_instance.k8s_worker[0] (local-exec): ok: [3.144.89.30]

aws_instance.k8s_worker[0] (local-exec): PLAY RECAP *********************************************************************
aws_instance.k8s_worker[0] (local-exec): 3.144.89.30              : ok=16   changed=13   unreachable=0    failed=0    skipped=1
scued=0    ignored=0

aws_instance.k8s_worker[0]: Creation complete after 2m38s [id=i-046b8e9f231c9bad6]
null_resource.fetch_kubeconfig: Creating...
null_resource.fetch_kubeconfig: Provisioning with 'local-exec'...
null_resource.fetch_kubeconfig (local-exec): Executing: ["/bin/sh" "-c" "      mkdir -p ~/.kube\n      scp -i ~/.ssh/k8s_rsa ubuntu@
223.241.225:~/.kube/config ~/.kube/config\n      sed 's/server: .*$/server: https:\\/\\/18.223.241.225:6443/g' ~/.kube/config > ~/.k
/config.tmp && mv -f ~/.kube/config.tmp ~/.kube/config\n\n"]
null_resource.fetch_kubeconfig: Creation complete after 3s [id=5577006791947779410]

Apply complete! Resources: 17 added, 0 changed, 0 destroyed.
hatem@Hatems-Priv-MBP terraform %
```

Verifying the cluster was installed and the wallet service as well (Using AWS console)

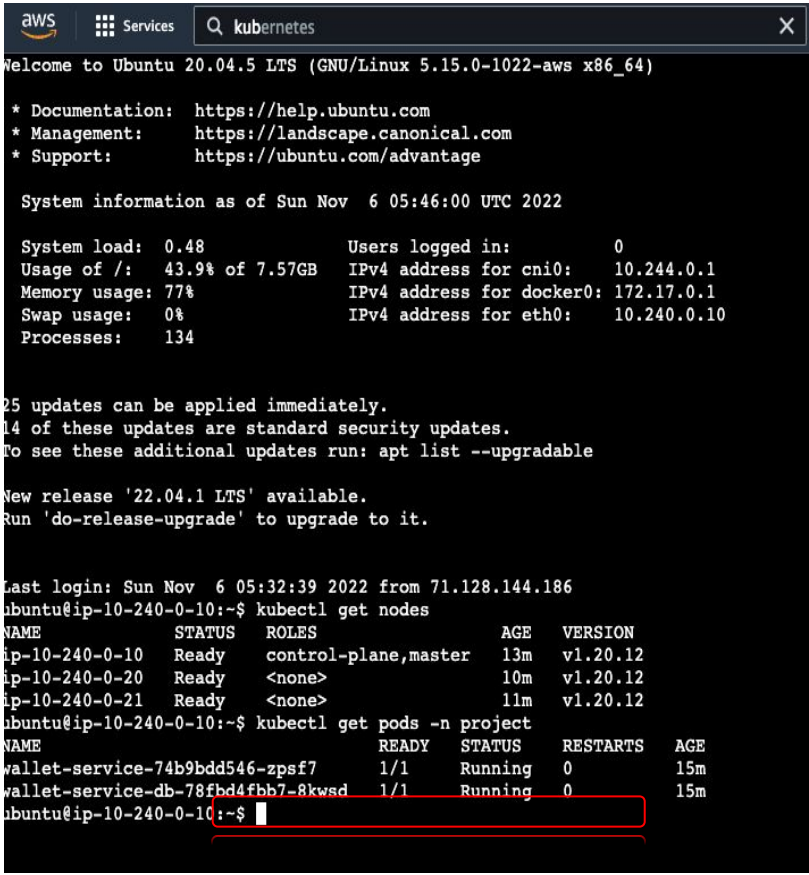


Login to the controller



walletservice

Postgres



kubectl top pods -A

```
ubuntu@ip-10-240-0-10:~$ kubectl top pods -A
NAMESPACE      NAME                                       CPU(cores)   MEMORY(bytes)
kube-flannel   kube-flannel-ds-ltcst                      4m           11Mi
kube-flannel   kube-flannel-ds-s8khv                      4m           10Mi
kube-flannel   kube-flannel-ds-thfj9                      4m           9Mi
kube-system    coredns-74ff55c5b-rkspb                    2m           8Mi
kube-system    coredns-74ff55c5b-sxjtg                    2m           9Mi
kube-system    etcd-ip-10-240-0-10                        9m           35Mi
kube-system    kube-apiserver-ip-10-240-0-10              41m          269Mi
kube-system    kube-controller-manager-ip-10-240-0-10     11m          55Mi
kube-system    kube-proxy-drccn                           1m           10Mi
kube-system    kube-proxy-j9zp8                           1m           13Mi
kube-system    kube-proxy-jpjlv                           1m           12Mi
kube-system    kube-scheduler-ip-10-240-0-10              2m           21Mi
kube-system    metrics-server-67f85b74f6-62tv6            3m           19Mi
project        wallet-service-74b9bdd546-zpsf7            1m           155Mi
project        wallet-service-db-78fbd4fbb7-8kwsd         1m           39Mi
ubuntu@ip-10-240-0-10:~$
```

kubectl get pvc-n project

```
ubuntu@ip-10-240-0-10:~$ kubectl get pvc -n project
NAME                                                STATUS   VOLUME                                              CAPACITY   ACCESS MODES   STORAGECLASS   AGE
wallet-service-postgres-data-persisent-volume-claim   Bound    wallet-service-postgres-data-persisent-volume   5Gi        RWO            manual         45m
wallet-service-source-code-persisent-volume-claim     Bound    wallet-service-source-code-persisent-volume     1Gi        RWO            manual         45m
ubuntu@ip-10-240-0-10:~$
```

kubectl get pv -n project

```
ubuntu@ip-10-240-0-10:~$ kubectl get pvc --all -n project
Error: unknown flag: --all
See 'kubectl get --help' for usage.
ubuntu@ip-10-240-0-10:~$ kubectl get pvc -n project
NAME                                                STATUS   VOLUME                                              CAPACITY   ACCESS MODES   STORAGECLASS   AGE
wallet-service-postgres-data-persisent-volume-claim   Bound    wallet-service-postgres-data-persisent-volume   5Gi        RWO            manual         45m
wallet-service-source-code-persisent-volume-claim     Bound    wallet-service-source-code-persisent-volume     1Gi        RWO            manual         45m
ubuntu@ip-10-240-0-10:~$ kubectl get pv -n project
NAME                                            CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                                                       STORAGECLASS
  REASON   AGE
wallet-service-postgres-data-persisent-volume   5Gi        RWO            Retain           Bound    project/wallet-service-postgres-data-persisent-volume-claim   manual
         46m
wallet-service-source-code-persisent-volume     1Gi        RWO            Retain           Bound    project/wallet-service-source-code-persisent-volume-claim     manual
         46m
ubuntu@ip-10-240-0-10:~$
```

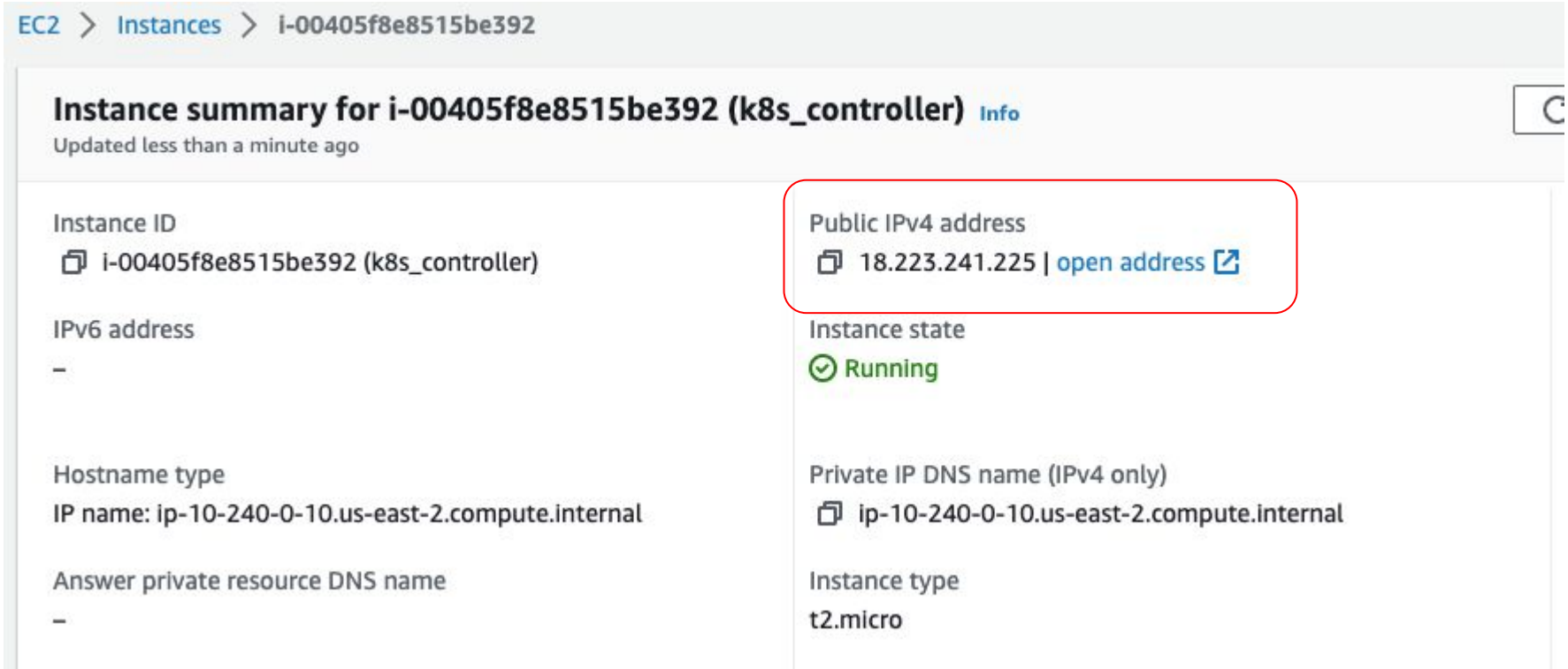
kubectl get replicaset -n project

```
ubuntu@ip-10-240-0-10:~$ kubectl get replicaset -n project
NAME                              DESIRED   CURRENT   READY   AGE
wallet-service-74b9bdd546         1         1         1       43m
wallet-service-db-78fbd4fbb7      1         1         1       43m
ubuntu@ip-10-240-0-10:~$
```

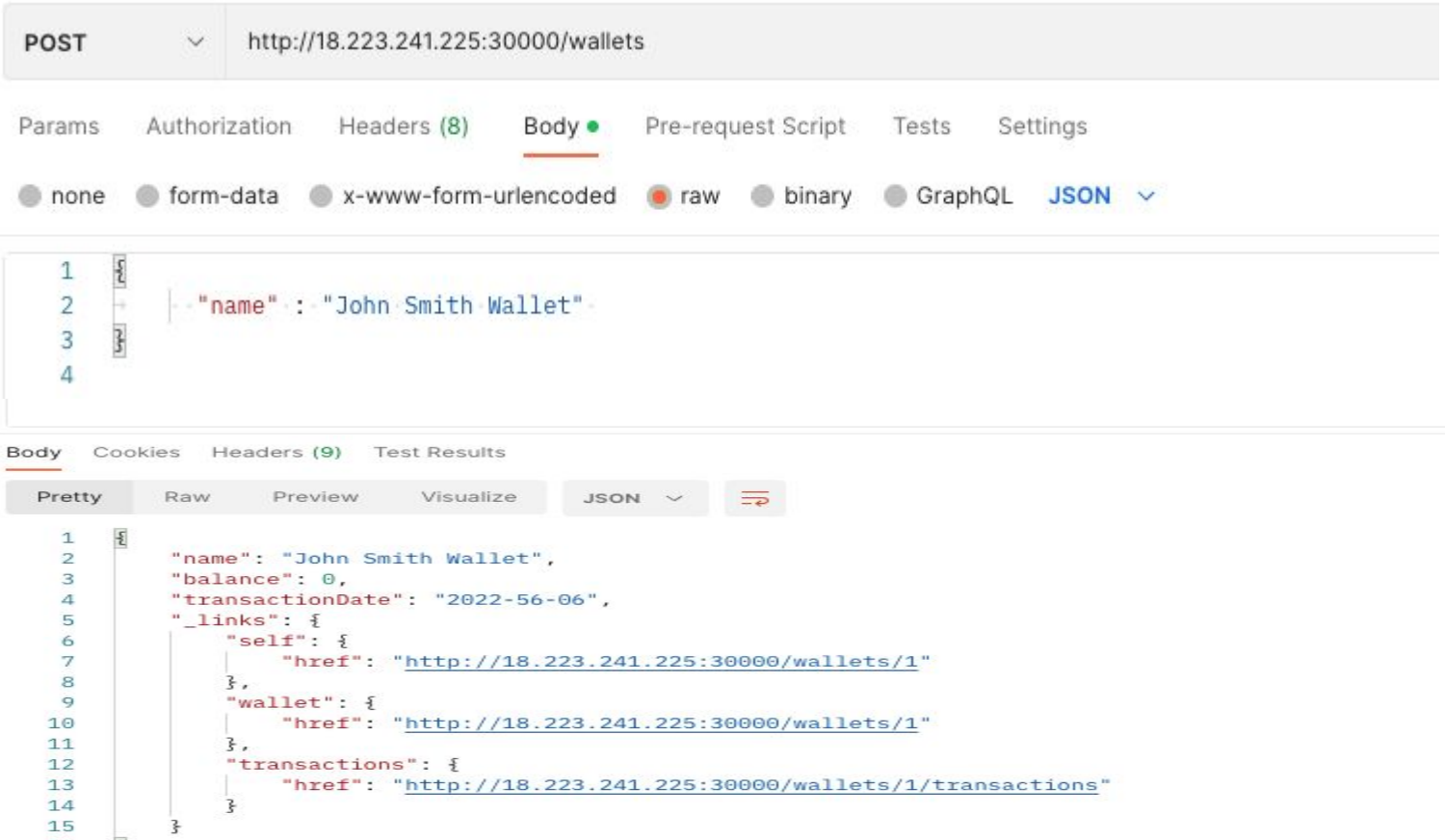
kubectl get replicaset -n project

```
ubuntu@ip-10-240-0-10:~$ kubectl get replicaset -n project
NAME                              DESIRED   CURRENT   READY   AGE
wallet-service-74b9bdd546         1         1         1       43m
wallet-service-db-78fbd4fbb7      1         1         1       43m
ubuntu@ip-10-240-0-10:~$
```

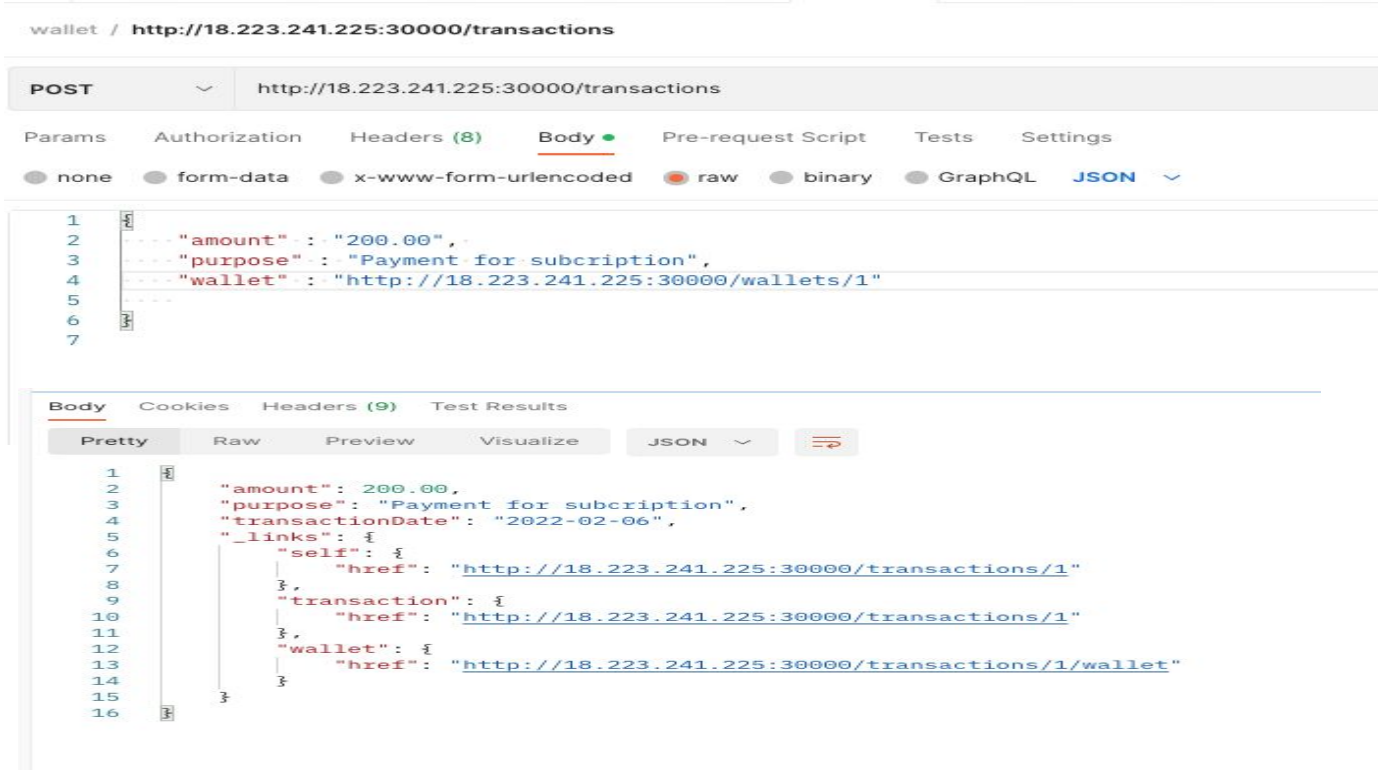
Using postman - getting public IP



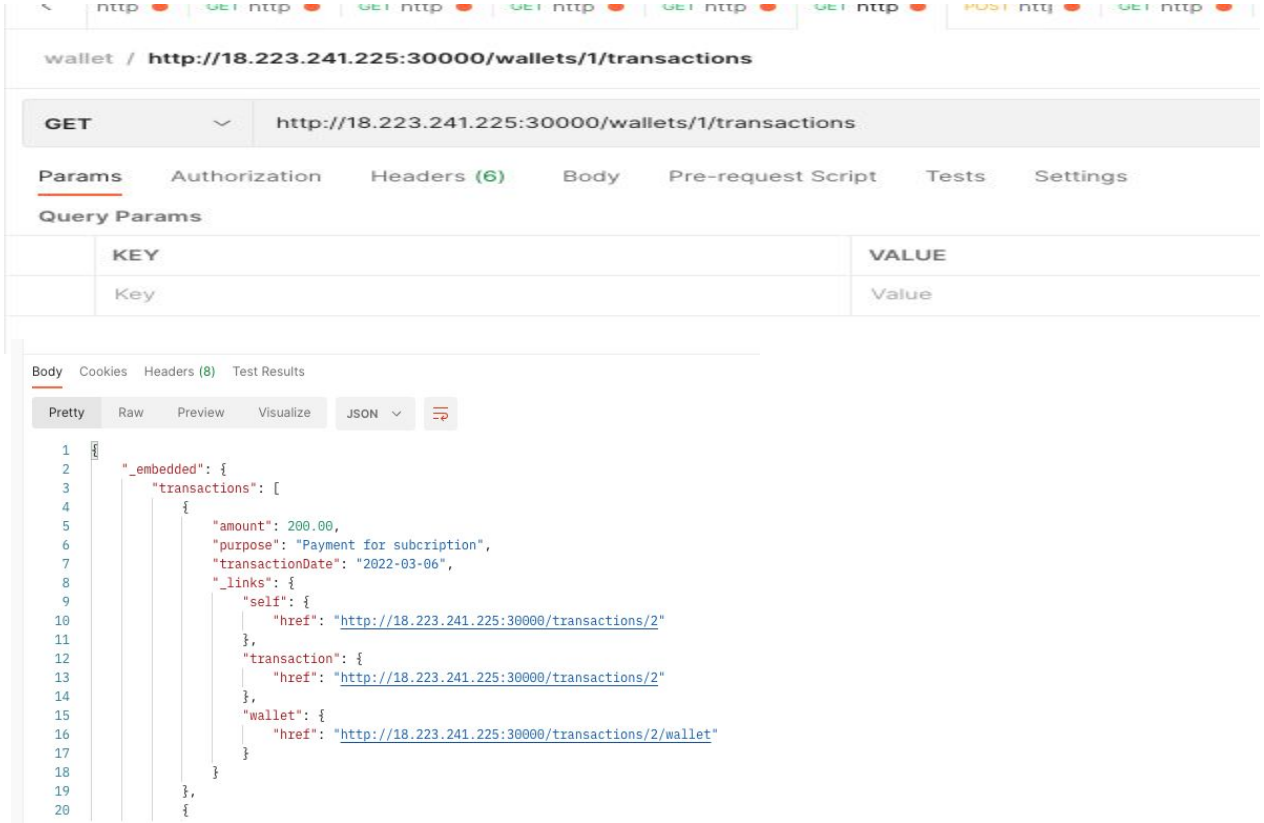
Using postman - Creating a wallet



Using postman - Creating transactions



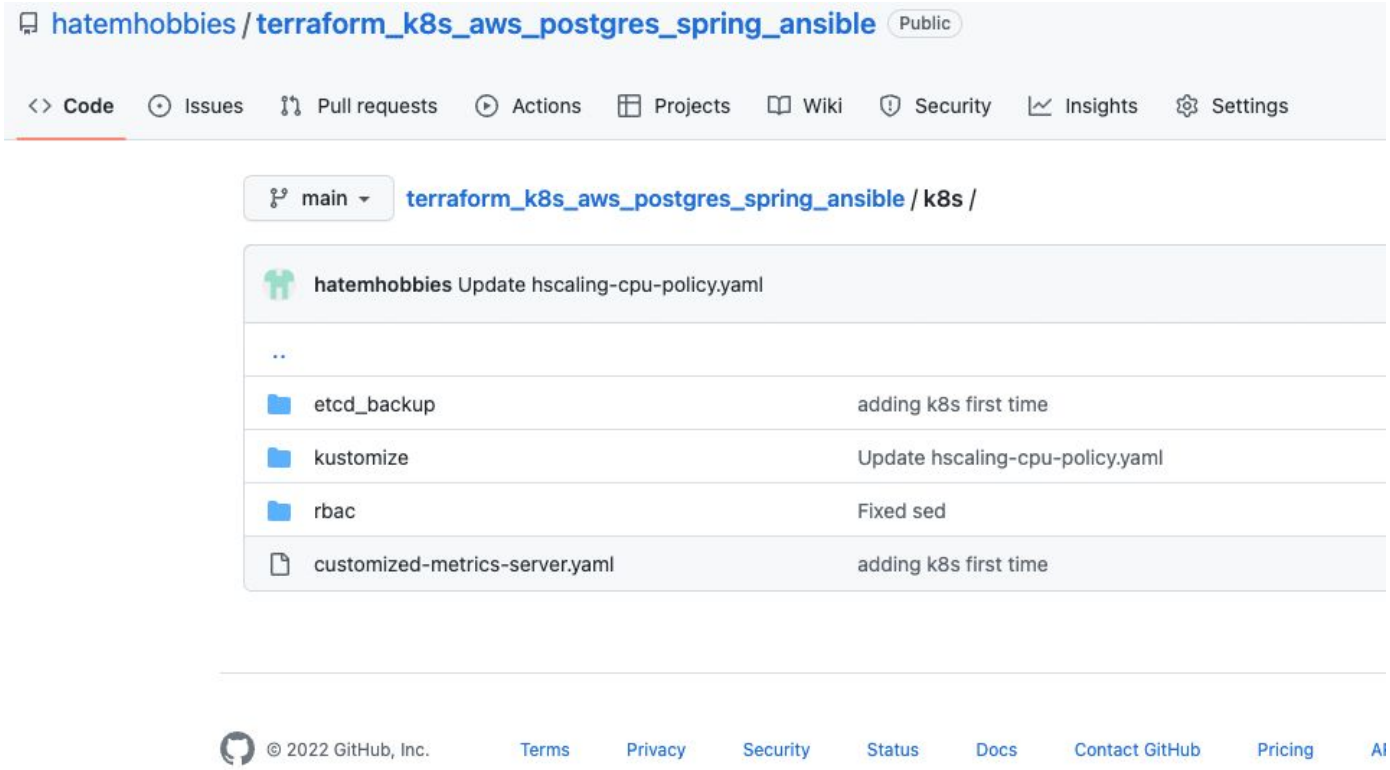
Using postman - Viewing the wallet



Section 3 - Role Based Access Control

Setting up a pod-admin user with limited access

k8s was copied to the controller



AWS controller node

AWS controller node

```
ubuntu@ip-10-240-0-10:~$ cd k8s
ubuntu@ip-10-240-0-10:~/k8s$ cd rbac
ubuntu@ip-10-240-0-10:~/k8s/rbac$ ls
README.md  create-certificate.sh  create-pod-user-config.sh  pod-admin-csr.yaml  pod-admin-role.yaml  pod-admin-roleBinding.yaml
ubuntu@ip-10-240-0-10:~/k8s/rbac$ tree
.
├── README.md
├── create-certificate.sh
├── create-pod-user-config.sh
├── pod-admin-csr.yaml
├── pod-admin-role.yaml
├── pod-admin-roleBinding.yaml
└── rbac_apply.sh

0 directories, 7 files
ubuntu@ip-10-240-0-10:~/k8s/rbac$
```

Creating certificate

```
ubuntu@ip-10-240-0-10:~/k8s/rbac$ cat create-certificate.sh
#!/bin/bash

set -x

openssl genrsa -out  pod-admin.key 2048
openssl req -new -key pod-admin.key -out pod-admin.csr -subj "/CN=pod-admin/O=project"

# Replace the CSR in the csr.yaml
CSR=$(cat pod-admin.csr | base64 | tr -d '\n')
sed -i "s/__CSR___/${CSR}/g" pod-admin-csr.yaml
ubuntu@ip-10-240-0-10:~/k8s/rbac$
```

```
ubuntu@ip-10-240-0-10:~/k8s/rbac$ cat create-certificate.sh
#!/bin/bash

set -x

openssl genrsa -out  pod-admin.key 2048
openssl req -new -key pod-admin.key -out pod-admin.csr -subj "/CN=p

# Replace the CSR in the csr.yaml
CSR=$(cat pod-admin.csr | base64 | tr -d '\n')
sed -i "s/__CSR___/${CSR}/g" pod-admin-csr.yaml
ubuntu@ip-10-240-0-10:~/k8s/rbac$ sh ./create-certificate.sh
+ openssl genrsa -out pod-admin.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
..............+++++
...+++++
e is 65537 (0x010001)
+ openssl req -new -key pod-admin.key -out pod-admin.csr -subj /CN=
+ base64
+ cat pod-admin.csr
+ tr -d \n
+ CSR=LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0KTUlJQ2F6Q0NBV
qQU5CZ2txaGtpRzl3MEJBUUVGQUFPQ0FROEFNSUlCQ2dLQ0FRRUFwYVNENFlXS1h4dU
WFUyMTkKSDlnM2J6SDNIVFZodkIvTC9QcER5VmxZdmg3T1RnNWs2QVcxOHdaWFFzbzU
09ZTDZCUU9WNGJiNDhYM0U5Z0FOCkNIVUFKNWpMYkJ4RGVtVndhYkdodThyZDdRZEQv
dsZ05qbVU4TExVT1RkdEtTSFlTZUh6OWdkeW1oMERvVlZBaQpvaC9mNVFJREFRQUJvC
3SU1xd3I3TlJQeE1laHcxSDFMZEs5RmI2QWlvMVAwbGlyazRXM1Q5M3VoMHFXWmtKUz
dgphVnFBZXQ1dHdaL0pPdkJkQnlBT1pqaHY3bTRpVWRUODhkNXRiNlhkMU0zbEJvcmE
lFoR2FuTTBJb1MvZFhac2gKc0xmRUVlZldxSnJOelkxYmQvOTE0WlYrTER1VE9CVGVM
0K
+ sed -i s/__CSR___/LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0
QpaV04wTUlJQklqQU5CZ2txaGtpRzl3MEJBUUVGQUFPQ0FROEFNSUlCQ2dLQ0FRRUFw
V0TjJNZG9VRG5SWFUyMTkKSDlnM2J6SDNIVFZodkIvTC9QcER5VmxZdmg3T1RnNWs2Q
ERVJnR2ZCdDJYV09ZTDZCUU9WNGJiNDhYM0U5Z0FOCkNIVUFKNWpMYkJ4RGVtVndhYk
Ylh3U2hPWVRCL2dsZ05qbVU4TExVT1RkdEtTSFlTZUh6OWdkeW1oMERvVlZBaQpvaC9
0t4ZEI5d2Q5MVp3SU1xd3I3TlJQeE1laHcxSDFMZEs5RmI2QWlvMVAwbGlyazRXM1Q5
M2eS9Pd0p0VjlUdgphVnFBZXQ1dHdaL0pPdkJkQnlBT1pqaHY3bTRpVWRUODhkNXRiN
TVGRINWhZYnR6VlFoR2FuTTBJb1MvZFhac2gKc0xmRUVlZldxSnJOelkxYmQvOTE0Wl
```

rbac_apply.sh

```
ubuntu@ip-10-240-0-10:~/k8s/rbac$ cat rbac_apply.sh
kubectl apply -f pod-admin-csr.yaml
kubectl get csr

kubectl certificate approve pod-admin

echo "waiting ...."

sleep 20

kubectl apply -f pod-admin-role.yaml

kubectl get roles -n project

kubectl apply -f pod-admin-roleBinding.yaml

kubectl get rolebindings -n project

kubectl describe rolebinding pod-admin-role-rolebindi

kubectl auth can-i list pods --as pod-admin -n project

kubectl auth can-i create pods --as pod-admin -n project
ubuntu@ip-10-240-0-10:~/k8s/rbac$
```

```
ubuntu@ip-10-240-0-10:~/k8s/rbac$ cat rbac_apply.sh
kubectl apply -f pod-admin-csr.yaml
kubectl get csr

kubectl certificate approve pod-admin

echo "waiting ...."

sleep 20

kubectl apply -f pod-admin-role.yaml

kubectl get roles -n project

kubectl apply -f pod-admin-roleBinding.yaml

kubectl get rolebindings -n project

kubectl describe rolebinding pod-admin-role-rolebinding -n project

kubectl auth can-i list pods --as pod-admin -n project

kubectl auth can-i create pods --as pod-admin -n project
ubuntu@ip-10-240-0-10:~/k8s/rbac$ sh ./rbac_apply.sh
certificatesigningrequest.certificates.k8s.io/pod-admin created
NAME        AGE   SIGNERNAME                                    REQUESTOR                    CONDITION
csr-bhcrd   58m   kubernetes.io/kube-apiserver-client-kubelet   system:node:ip-10-240-0-10   Approved,Issued
csr-qvgx9   55m   kubernetes.io/kube-apiserver-client-kubelet   system:bootstrap:uiph2o      Approved,Issued
csr-vlnk7   55m   kubernetes.io/kube-apiserver-client-kubelet   system:bootstrap:uiph2o      Approved,Issued
pod-admin   3s    kubernetes.io/kube-apiserver-client           kubernetes-admin             Pending
certificatesigningrequest.certificates.k8s.io/pod-admin approved
waiting ....
```

rbac_apply.sh cont

```
waiting ....
role.rbac.authorization.k8s.io/pod-admin-role created
NAME             CREATED AT
pod-admin-role   2022-11-06T06:30:48Z
rolebinding.rbac.authorization.k8s.io/pod-admin-role-rolebinding created
NAME                          ROLE                   AGE
pod-admin-role-rolebinding    Role/pod-admin-role    1s
Name:         pod-admin-role-rolebinding
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  pod-admin-role
Subjects:
  Kind  Name       Namespace
  ----  ----       ---------
  User  pod-admin
yes
yes
ubuntu@ip-10-240-0-10:~/k8s/rbac$
```

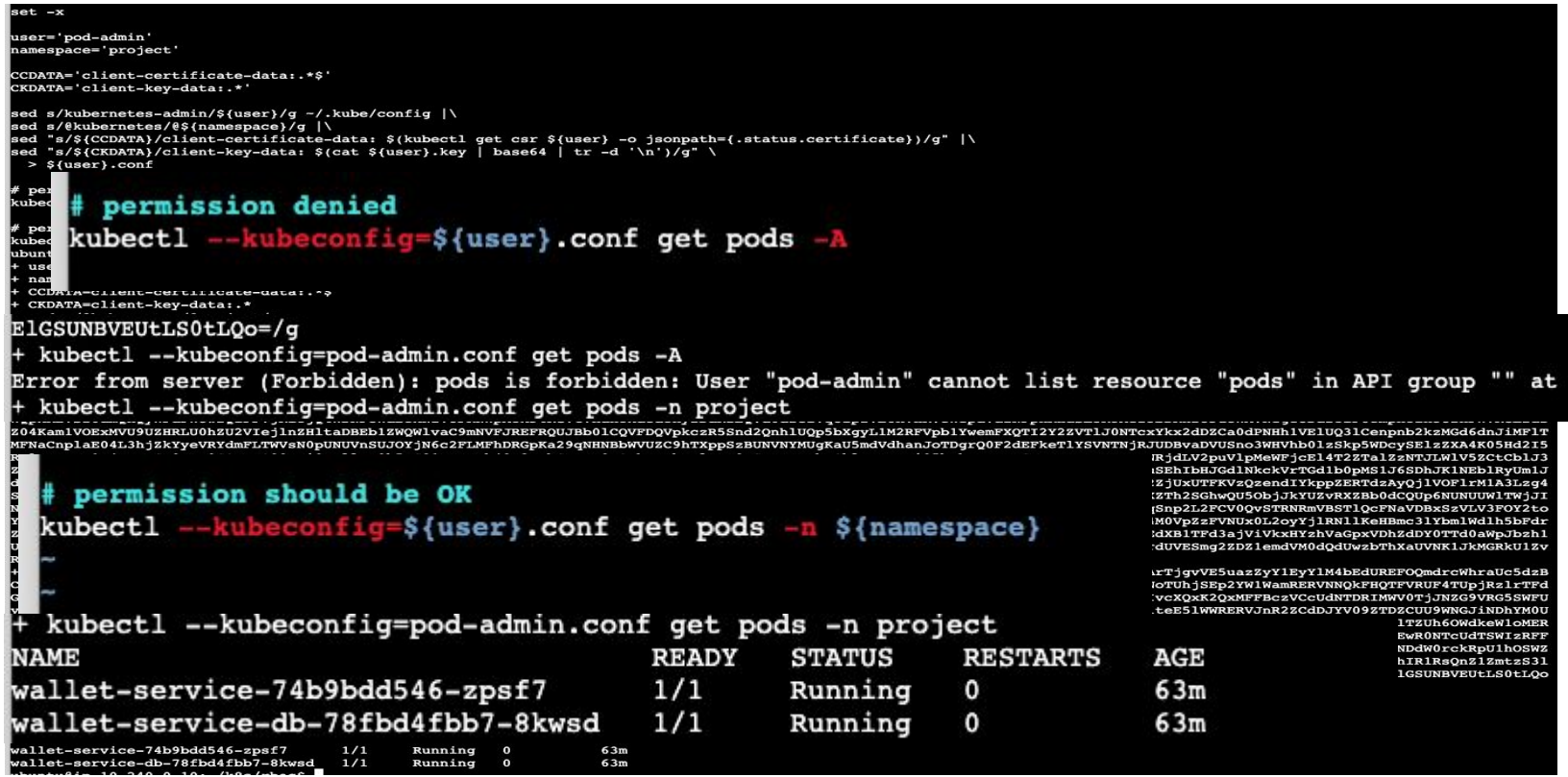
kubectl get csr -n project

```
ubuntu@ip-10-240-0-10:~/k8s/rbac$ vi rbac_apply.sh
ubuntu@ip-10-240-0-10:~/k8s/rbac$ kubectl get csr -n project
NAME        AGE     SIGNERNAME                                    REQUESTOR                   CONDITION
csr-bhcrd   60m     kubernetes.io/kube-apiserver-client-kubelet   system:node:ip-10-240-0-10   Approved,Issued
csr-qvgx9   58m     kubernetes.io/kube-apiserver-client-kubelet   system:bootstrap:uiph2o      Approved,Issued
csr-vlnk7   58m     kubernetes.io/kube-apiserver-client-kubelet   system:bootstrap:uiph2o      Approved,Issued
pod-admin   2m56s   kubernetes.io/kube-apiserver-client           kubernetes-admin            Approved,Issued
ubuntu@ip-10-240-0-10:~/k8s/rbac$
```

create-pod-user-config.sh

```
#!/bin/bash

set -x

user='pod-admin'
namespace='project'

CCDATA='client-certificate-data:.*$'
CKDATA='client-key-data:.*'

sed s/kubernetes-admin/${user}/g ~/.kube/config |\
sed s/@kubernetes/@${namespace}/g |\
sed "s/${CCDATA}/client-certificate-data: $(kubectl get csr ${user} -o jsonpath={.status.certificate})/g" |\
sed "s/${CKDATA}/client-key-data: $(cat ${user}.key | base64 | tr -d '\n')/g" \
  > ${user}.conf

# permission denied
kubectl --kubeconfig=${user}.conf get pods -A

# permission should be OK
kubectl --kubeconfig=${user}.conf get pods -n ${namespace}ubuntu@ip-10-240-0-10:~/k8s/rbac$
```

sh ./create-pod-user-config.sh



Summary on RBAC for pod-admin

- Pod-admin user is setup to have access only to the project namespace
- Hence -A rejects the access whereas -n project allows it

```
# permission denied
kubectl --kubeconfig=${user}.conf get pods -A
```

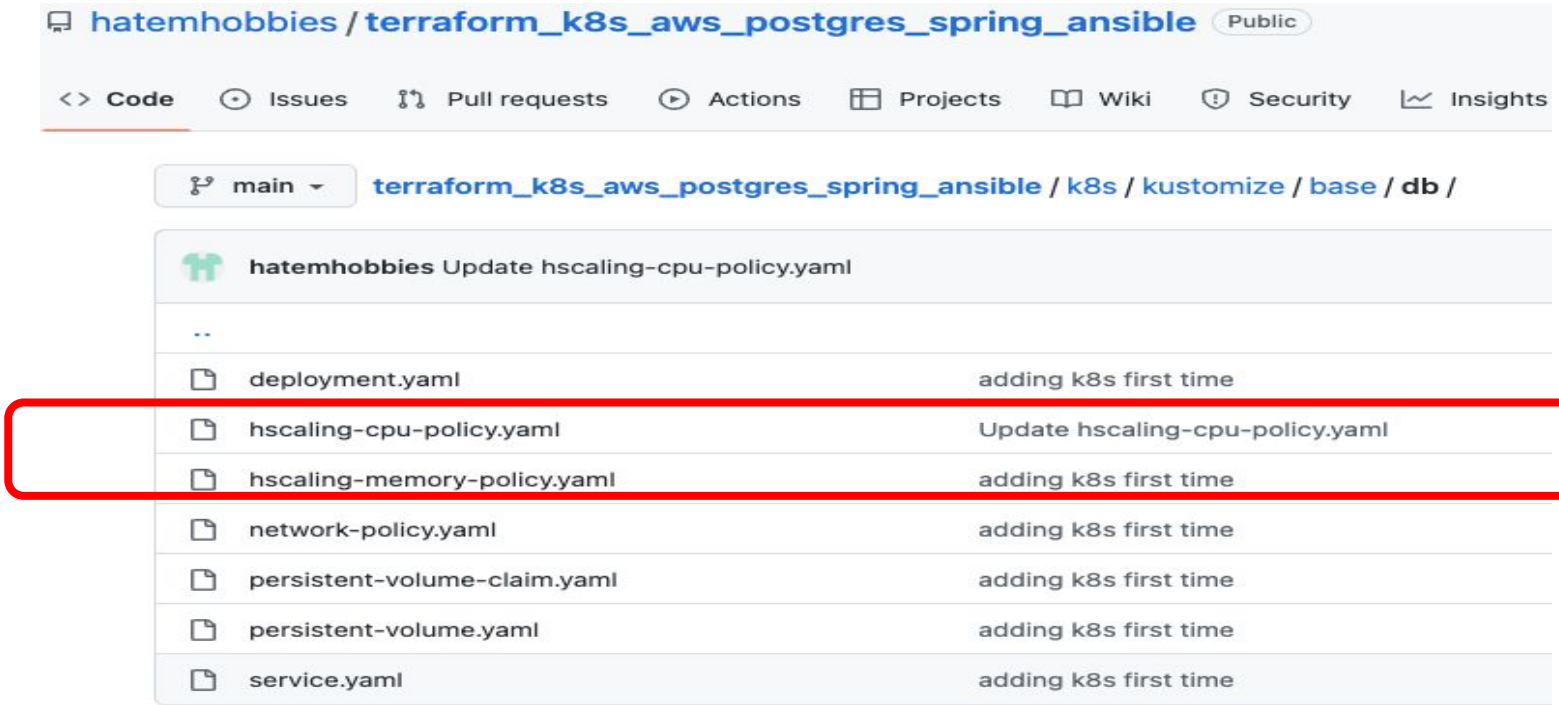
```
ElGSUNBVEUtLS0tLQo=/g
+ kubectl --kubeconfig=pod-admin.conf get pods -A
Error from server (Forbidden): pods is forbidden: User "pod-admin" cannot list resource "pods" in API group "" at the cluster scope
+ kubectl --kubeconfig=pod-admin.conf get pods -n project
```

```
# permission should be OK
kubectl --kubeconfig=${user}.conf get pods -n ${namespace}
~
~
```

```
+ kubectl --kubeconfig=pod-admin.conf get pods -n project
NAME                                  READY   STATUS    RESTARTS   AGE
wallet-service-74b9bdd546-zpsf7       1/1     Running   0          63m
wallet-service-db-78fbd4fbb7-8kwsd    1/1     Running   0          63m
```

Section 4 - Simulating load on the worker CPU

(The YAML are setup to scale up if CPU or memory hit the threshhold)


hatemhobbies / terraform_k8s_aws_postgres_spring_ansible
Public
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
main
terraform_k8s_aws_postgres_spring_ansible / k8s / kustomize / base / db /
hatemhobbies Update hscaling-cpu-policy.yaml
..
deployment.yaml
adding k8s first time
hscaling-cpu-policy.yaml
Update hscaling-cpu-policy.yaml
hscaling-memory-policy.yaml
adding k8s first time
network-policy.yaml
adding k8s first time
persistent-volume-claim.yaml
adding k8s first time
persistent-volume.yaml
adding k8s first time
service.yaml
adding k8s first time

Examining hscaling-cpu-policy.yaml

main ▾ terraform_k8s_aws_postgres_spring_ansible / k8s / kustomize / base / db / **hscaling-cpu-policy.yaml**

hatemhobbies Update hscaling-cpu-policy.yaml La

1 contributor

13 lines (13 sloc) | 441 Bytes

```
1   apiVersion: autoscaling/v1
2   kind: HorizontalPodAutoscaler
3   metadata:
4     name: wallet-service-db-cpu-scaler
5     namespace: project
6   spec:
7     maxReplicas: 4 # maximum replicas of pods
8     minReplicas: 1
9     scaleTargetRef:
10      apiVersion: apps/v1
11      kind: Deployment
12      name: wallet-service-db # TARGET name of the deployment the autoscaler need to be run on
13    targetCPUUtilizationPercentage: 30 # CPU maximum amount of use on the pod is set to 30%.
```

```
ubuntu@ip-10-240-0-10:~$ kubectl get replicaset -n project
NAME                          DESIRED   CURRENT   READY   AGE
wallet-service-74b9bdd546     1         1         1       81m
wallet-service-db-78fbd4fbb7  1         1         1       81m
ubuntu@ip-10-240-0-10:~$
```

Examining hscaling-memory-policy.yaml

main terraform_k8s_aws_postgres_spring_ansible / k8s / kustomize / base / db / hscaling-memory-policy.yaml

hatemhobbies adding k8s first time — Latest commit 6c6a6

1 contributor

19 lines (19 sloc) | 408 Bytes — Raw

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: wallet-service-db-memory-scaler
  namespace: project
spec:
  maxReplicas: 4
  minReplicas: 1
  metrics:
    - type: Resource
      resource:
        name: memory
        target:
          type: Utilization
          averageValue: 50Mi
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: wallet-service-db
```

Login to the database pod so that we can push the CPU

```
ubuntu@ip-10-240-0-10:~/k8s/rbac$ kubectl get pods -n project
NAME                                    READY   STATUS    RESTARTS   AGE
wallet-service-74b9bdd546-zpsf7         1/1     Running   0          79m
wallet-service-db-78fbd4fbb7-8kwsd      1/1     Running   0          79m
ubuntu@ip-10-240-0-10:~/k8s/rbac$
```

```
ubuntu@ip-10-240-0-10:~$ kubectl get replicaset
No resources found in default namespace.
ubuntu@ip-10-240-0-10:~$ kubectl get replicaset -n project
NAME                                DESIRED   CURRENT   READY   AGE
wallet-service-74b9bdd546           1         1         1       81m
wallet-service-db-78fbd4fbb7        1         1         1       81m
ubuntu@ip-10-240-0-10:~$
```

kubectl exec -it -n project pods/wallet-service-db-78fbd4fbb7-8kwsd -- /bin/bash

```
ubuntu@ip-10-240-0-10:~/k8s/rbac$ kubectl exec -it -n project pods/wallet-service-db-78fbd4fbb7-8kwsd -- /bin/bash

bash-5.1#
bash-5.1#
```

Creating loads

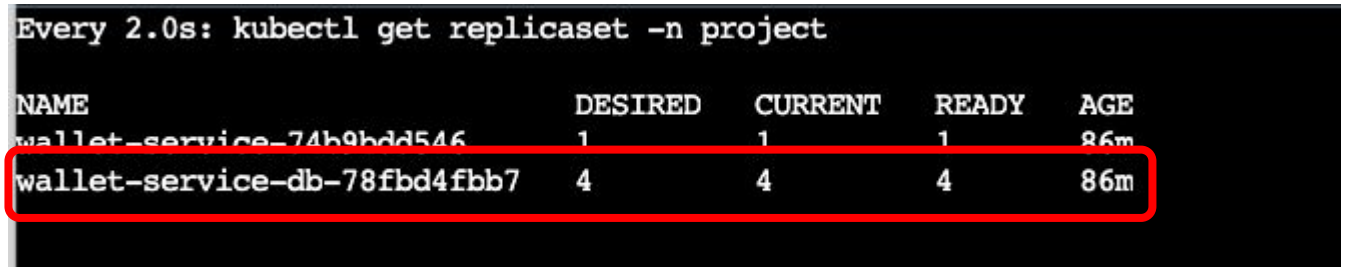
We will execute the following many times in the pod

while true; do /bin/true; done &

Executing loads

```
ubuntu@ip-10-240-0-10:~/k8s/rbac$ kubectl exec -it -n project pods/wallet-service-db-78fbd4fbb7-8kwsd   -- /bin/bash


bash-5.1#
bash-5.1# while true; do /bin/true; done &
[1] 173
bash-5.1# while true; do /bin/true; done &
[2] 2053
bash-5.1# while true; do /bin/true; done &
[3] 2955
bash-5.1# while true; do /bin/true; done &
[4] 3946
bash-5.1# while true; do /bin/true; done &
[5] 4750
bash-5.1# while true; do /bin/true; done &
[6] 5520
bash-5.1# while true; do /bin/true; done &
[7] 6256
bash-5.1# while true; do /bin/true; done &
[8] 7022
bash-5.1# while true; do /bin/true; done &
```

Effect of the load - the database pod scaled up

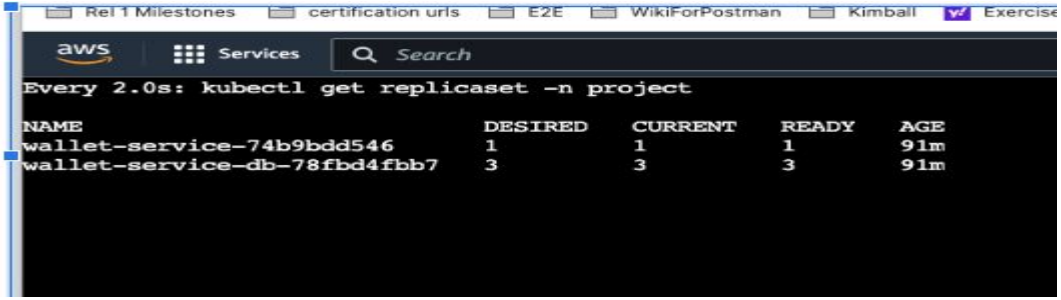


Scale down after the load is slowed

Reducing the load by doing fg and control C to kill the programs causing the load until no bg (background the extra cpu heavy programs)

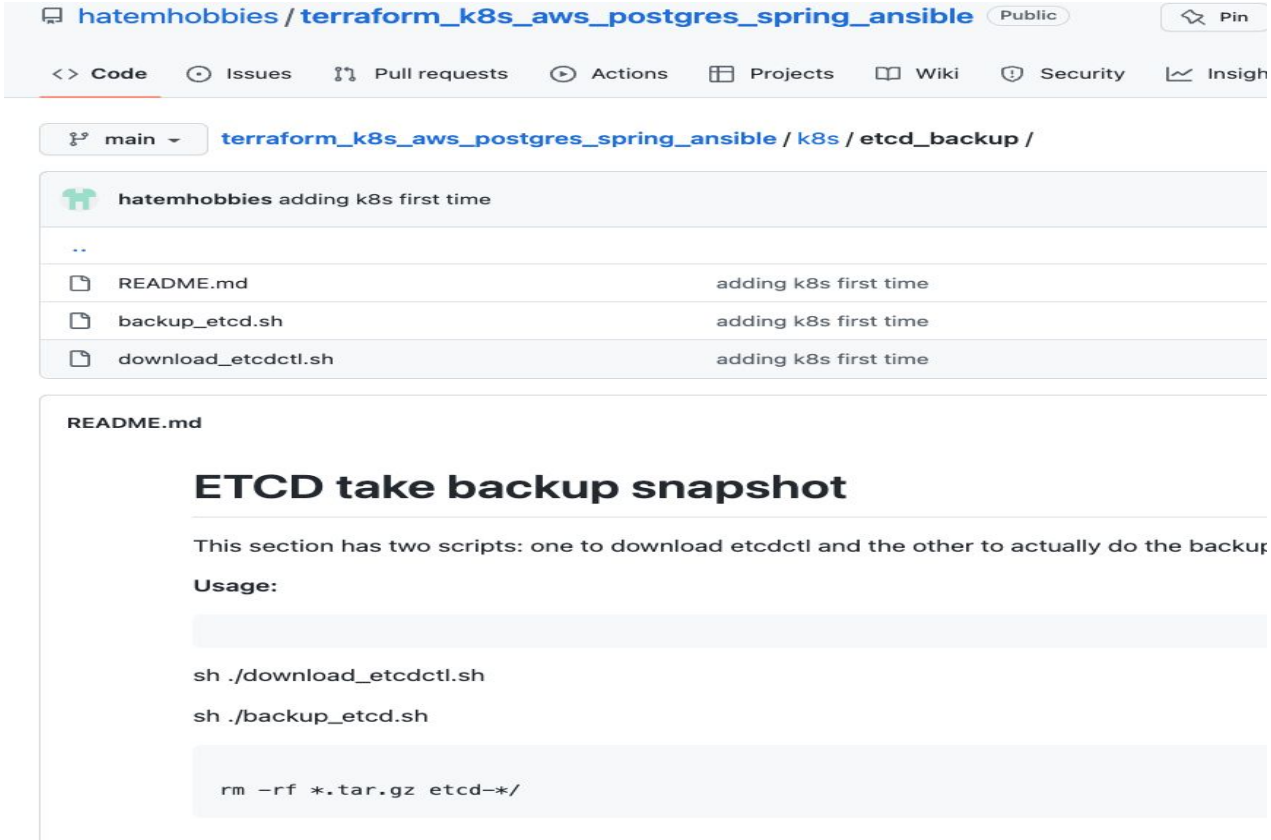
```
[9] 7877
bash-5.1# fg
while true; do
    /bin/true;
done
^C^C
bash-5.1#
```

```
    /bin/true;
done
^C
bash-5.1# ^C
bash-5.1# fg
bash: fg: current: no such job
bash-5.1#
```



Section 5 - Saving a snapshot for the ETCD database

Scripts to install ETCD and then do a snapshot backup



Installing ETCD using download_etcdctl.sh

main ▾ terraform_k8s_aws_postgres_spring

hatemhobbies adding k8s first time

1 contributor

25 lines (19 sloc) | 482 Bytes

```
1   #!/bin/bash
2
3   set -x
4
5   # Create temp folder
6   DOWNLOAD_DIR=/tmp/etcd
7   [ -d ${DOWNLOAD_DIR} ] || mkdir -p ${DOWNLOAD_
8   cd ${DOWNLOAD_DIR}
9
10  # Get the latest from the releases folder
11  curl -s https://api.github.com/repos/etcd-io/e
12    grep browser_download_url |\
13    grep linux-amd64 |\
14    cut -d '"' -f 4 | wget -qi -
15
16  # Extract the archive
17  tar xvf *.tar.gz
18
19  # Install in /usr/local/bin
20  cd etcd-*/
21  sudo mv etcd* /usr/local/bin/
22
23  # Cleanup
24  cd ..
25  rm -rf *.tar.gz etcd-*/
```

```
ubuntu@ip-10-240-0-10:~$ cd k8s
ubuntu@ip-10-240-0-10:~/k8s$ ls
customized-metrics-server.yaml  etcd_backup  kustomize  rbac
ubuntu@ip-10-240-0-10:~/k8s$ cd etcd_backup/
ubuntu@ip-10-240-0-10:~/k8s/etcd_backup$ ls
README.md  backup_etcd.sh  download_etcdctl.sh
ubuntu@ip-10-240-0-10:~/k8s/etcd_backup$ ./download_etcdctl.sh
+ DOWNLOAD_DIR=/tmp/etcd
+ '[' -d /tmp/etcd ']'
+ mkdir -p /tmp/etcd
+ cd /tmp/etcd
+ grep browser_download_url
+ grep linux-amd64
+ cut -d '"' -f 4
+ wget -qi -
+ curl -s https://api.github.com/repos/etcd-io/etcd/releases/latest
+ tar xvf etcd-v3.4.22-linux-amd64.tar.gz
etcd-v3.4.22-linux-amd64/
```

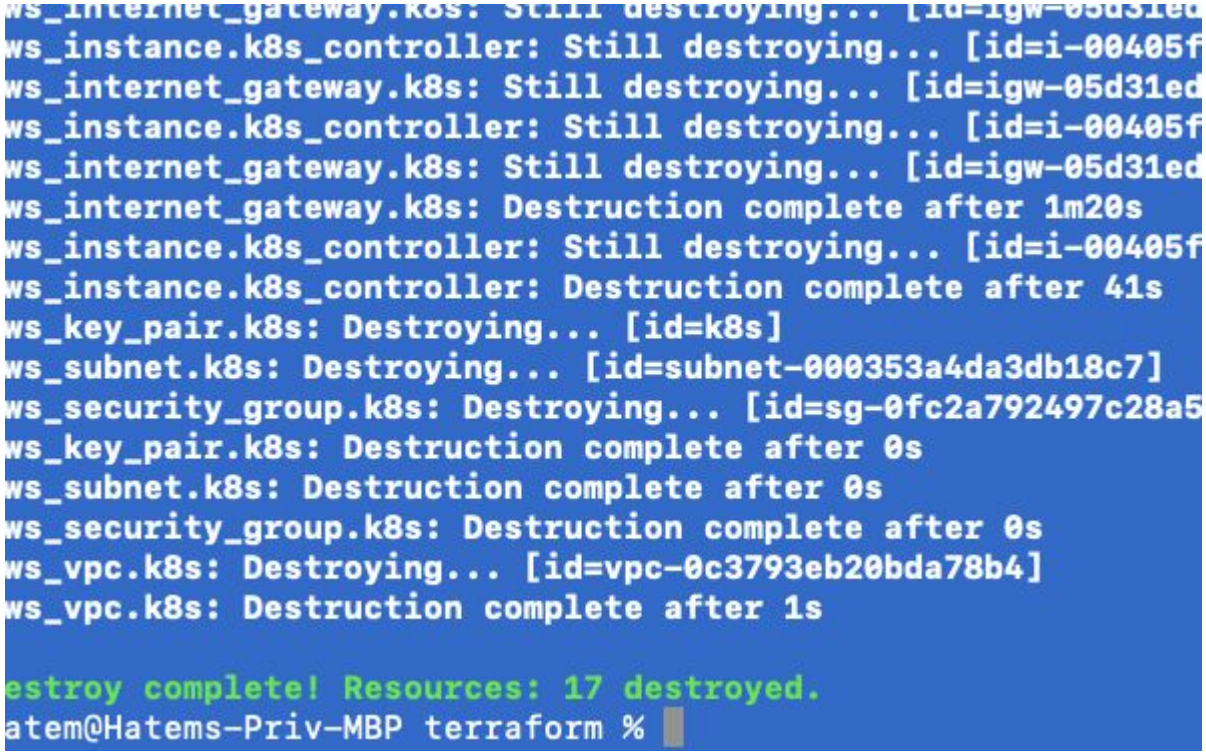
```
etcd-v3.4.22-linux-amd64/Documentation/v2/upgrade_2_3.md
+ cd etcd-v3.4.22-linux-amd64/
+ sudo mv etcd etcdctl /usr/local/bin/
+ cd ..
+ rm -rf etcd-v3.4.22-linux-amd64.tar.gz etcd-v3.4.22-linux-amd64/
ubuntu@ip-10-240-0-10:~/k8s/etcd_backup$
```

Backup of ETCD

```
ubuntu@ip-10-240-0-10:~/k8s/etcd_backup$ ./backup_etcd.sh
+ ETC_BACKUP_DIR=/tmp/etcd
+ '[' -d /tmp/etcd ']'
+ cd /tmp/etcd
+ K8S_CERT_FOLDER=/etc/kubernetes/pki/etcd
+ sudo ETCDCTL_API=3 etcdctl snapshot save snapshot.db --cacert /etc/kubernetes/pki/etcd/ca.crt --cert /etc/kubernetes/pki/etcd/server.crt --key /etc/kub
ernetes/pki/etcd/server.key
{"level":"info","ts":1667718694.1467674,"caller":"snapshot/v3_snapshot.go:119","msg":"created temporary db file","path":"snapshot.db.part"}
{"level":"info","ts":"2022-11-06T07:11:34.232Z","caller":"clientv3/maintenance.go:200","msg":"opened snapshot stream; downloading"}
{"level":"info","ts":1667718694.2330723,"caller":"snapshot/v3_snapshot.go:127","msg":"fetching snapshot","endpoint":"127.0.0.1:2379"}
{"level":"info","ts":"2022-11-06T07:11:34.338Z","caller":"clientv3/maintenance.go:208","msg":"completed snapshot read; closing"}
{"level":"info","ts":1667718694.3500473,"caller":"snapshot/v3_snapshot.go:142","msg":"fetched snapshot","endpoint":"127.0.0.1:2379","size":"3.5 MB","took
":0.196633267}
{"level":"info","ts":1667718694.3505018,"caller":"snapshot/v3_snapshot.go:152","msg":"saved","path":"snapshot.db"}
Snapshot saved at snapshot.db
ubuntu@ip-10-240-0-10:~/k8s/etcd_backup$
```

```
terraform_1893433510.sh
ubuntu@ip-10-240-0-10:~/k8s/etcd_backup$ ls /tmp/etcd
snapshot.db
ubuntu@ip-10-240-0-10:~/k8s/etcd_backup$
```

terraform destroy -auto-approve



Section 6 -Conclusion and Unique Selling Points

Conclusion

The goal of the project was to use EC2, terraform, ansible, docker and kubernetes to deploy an easy pay service (wallet service). These tools were leveraged to fully automate the process (A la one button push and the k8s cluster as well as the app will be deployed).

An admin user with access only to the ‘project’ namespace was created. ETCD was deployed and a backup was done.

The most important unique selling value is the scaling up when the CPU or memory went up beyond the threshold for the database. Scale down is also automatic.

Future enhancements

- Use GitHub Actions to automate the pipeline
- Use Helm
- Use AWS EKS

Unique selling point

- Total automation using terraform for provisioning the infra
- The integration of ansible to apply the installation of k8s cluster on the controller
- The ansible includes automatically installing all the workers and joining the cluster
- Uses Kustomize to deal with the entire deployment (automated)
- Factored in the provisions to do Dev, QA, and Prod environments
- Policy to control scaling based on CPU and memory
- A Spring Data REST