

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=ZZZZS7DTTTTJXEEAAAAA  
  
export  
AWS_SECRET_ACCESS_KEY=8xxVAtexOQ780ydtUDGat54GlgsNZrhaALAfhuL  
  
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@Hatem's-Priv-MBP ~ % cd capstone_project1  
hatem@Hatem's-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@Hatem's-Priv-MBP capstone_project1 %
```

The EasyPay Spring Data Rest Application

(build the application)

```
mvn clean install -DskipTests
```

```
hatem@Hatem's-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@Hatem's-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/.m2/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]
```

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@Hatem's-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: 28                                         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@Hatem's-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.

```
1 # =====
2 # = DATA SOURCE
3 # =====
4 # Set here configurations for the database connection
5 spring.datasource.url=jdbc:postgresql://${POSTGRES_HOST:db}:5432/postgres
6 spring.datasource.username=${POSTGRES_USER:postgres}
7 spring.datasource.password=${POSTGRES_PASSWORD:postgres}
8 spring.datasource.driver-class-name=org.postgresql.Driver
9 # Keep the connection alive if idle for a long time (needed in production)
10 spring.datasource.testWhileIdle=true
11 spring.datasource.validationQuery=SELECT 1
12 # =====
13 # = JPA / HIBERNATE
14 # =====
```

28 lines (28 sloc) | 611 Bytes

```
1 version: '3.8'
2 services:
3   db:
4     image: postgres:14.1-alpine
5     restart: always
6     environment:
7       - POSTGRES_HOST=localhost
8       - POSTGRES_USER=postgres
9       - POSTGRES_PASSWORD=postgres
10    ports:
11      - '5432:5432'
12    volumes:
13      - ./postgres-data:/var/lib/postgresql/data
14    container_name: eppostgres
15  wallet-app:
16    image: hatemhobbies/wallet-service:2.0
17    build:
18      context: ../
19      dockerfile: Dockerfile
20    environment:
21      - POSTGRES_HOST=db
22    depends_on:
23      - db
24    ports:
25      - 8080:8080
```

The EasyPay Spring Data Rest Application **(Starting Docker compose)**

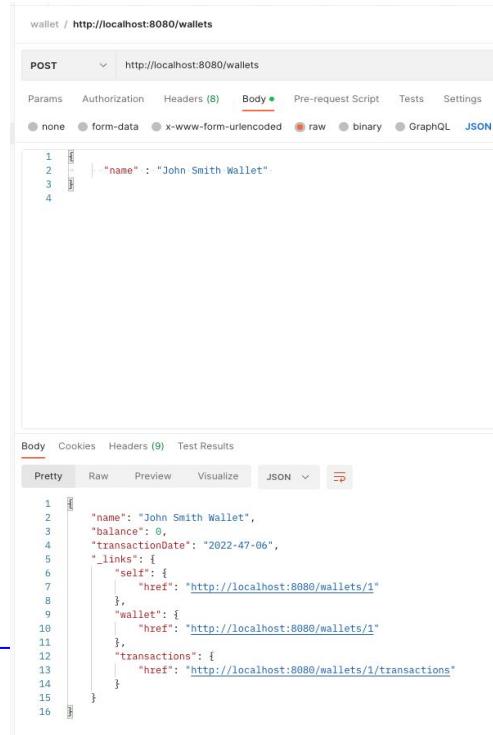
docker compose up

```
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
  # a06446b0flead 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
```

The EasyPay Spring Data Rest Application

(Testing using postman)

1) Create a wallet



wallet / http://localhost:8080/wallets

POST http://localhost:8080/wallets

Body

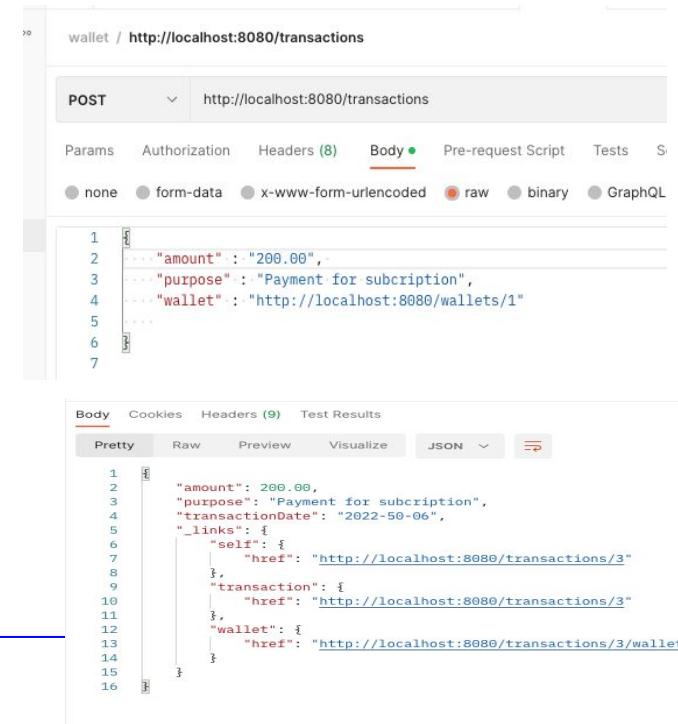
```
1
2   "name": "John Smith Wallet"
3
4
```

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON

```
1
2   "name": "John Smith Wallet",
3   "balance": 0,
4   "transactionDate": "2022-47-06",
5   "_links": {
6     "self": {
7       "href": "http://localhost:8080/wallets/1"
8     },
9     "wallet": {
10       "href": "http://localhost:8080/wallets/1"
11     },
12     "transactions": {
13       "href": "http://localhost:8080/wallets/1/transactions"
14     }
15   }
16 }
```

2) Add two transactions for 200\$ each



wallet / http://localhost:8080/transactions

POST http://localhost:8080/transactions

Body

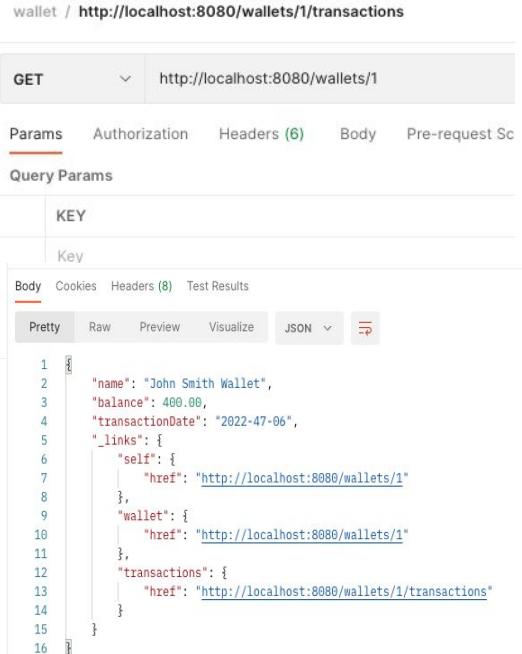
```
1
2   "amount": "200.00",
3   "purpose": "Payment for subscription",
4   "wallet": "http://localhost:8080/wallets/1"
5
6
7
8
9
10
11
12
13
14
15
16
```

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON

```
1
2   "amount": 200.00,
3   "purpose": "Payment for subscription",
4   "transactionDate": "2022-50-06",
5   "_links": {
6     "self": {
7       "href": "http://localhost:8080/transactions/3"
8     },
9     "transaction": {
10       "href": "http://localhost:8080/transactions/3"
11     },
12     "wallet": {
13       "href": "http://localhost:8080/wallets/3/wallet"
14     }
15   }
16 }
```

3) View wallet



wallet / http://localhost:8080/wallets/1/transactions

GET http://localhost:8080/wallets/1

Body

Query Params

KEY

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1
2   "name": "John Smith Wallet",
3   "balance": 400.00,
4   "transactionDate": "2022-47-06",
5   "_links": {
6     "self": {
7       "href": "http://localhost:8080/wallets/1"
8     },
9     "wallet": {
10       "href": "http://localhost:8080/wallets/1"
11     },
12     "transactions": {
13       "href": "http://localhost:8080/wallets/1/transactions"
14     }
15   }
16 }
```

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_0_ from "wallet" wallet0_ where wallet0_."id"=?
```

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**

```
        Forwarded URL = null  
        Redirected URL = null  
        Cookies = []  
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 5.244 s - in  
2022-11-05 23:59:38.485 INFO 19893 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : persistence unit 'default'  
2022-11-05 23:59:38.487 INFO 19893 --- [ionShutdownHook] com.zaxxer.hikari.HikariData...  
2022-11-05 23:59:38.490 INFO 19893 --- [ionShutdownHook] com.zaxxer.hikari.HikariData...  
[INFO]  
[INFO] Results:  
[INFO]  
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0  
[INFO]  
[INFO] --- maven-jar-plugin:3.2.2:jar (default-jar) @ wallet ---  
[INFO] --- spring-boot-maven-plugin:2.7.5:repackage (repackage) @ wallet ---  
[INFO] Replacing main artifact with repackaged archive  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----
```

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

 hatemhobbies	Update README.md	
 ansible		adding k8s first time
 k8s		Update hscaling-cpu-policy.yaml
 terraform		adding k8s first time
 .gitignore		first commit
 README.md		Update README.md

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

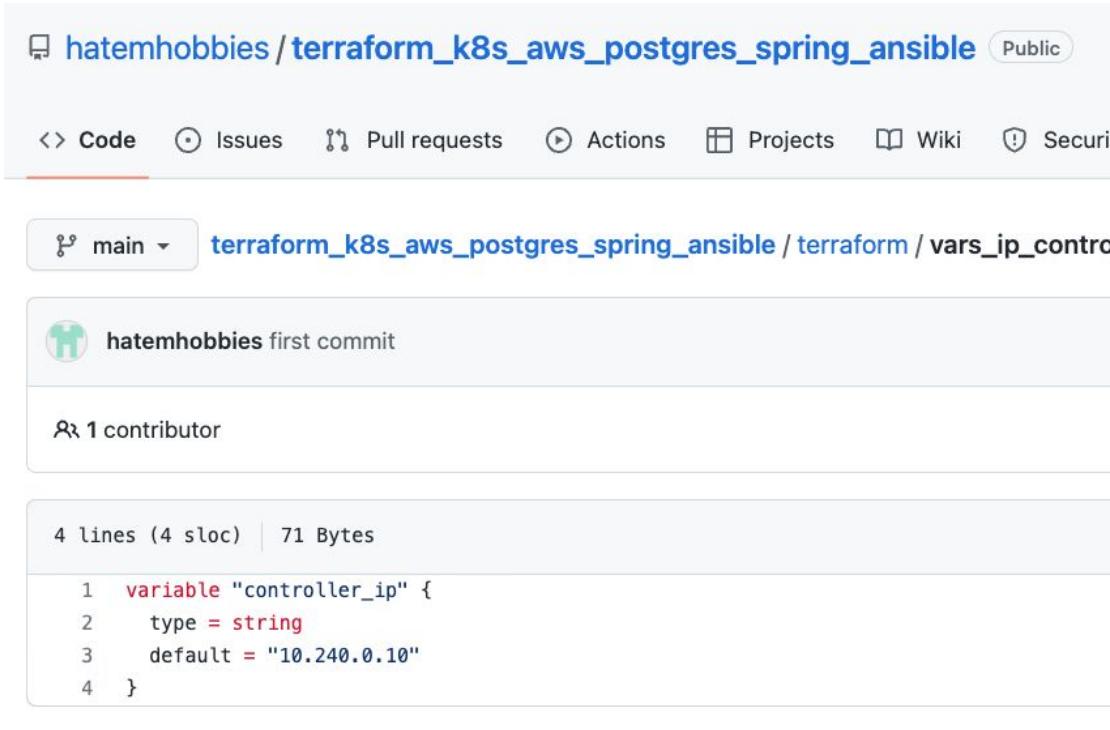
 vars_ip_workers.tf

 vars_keys.tf

 vpc.tf

 workers.tf

The vars_ip_controller.tf controls the controller node ip



The screenshot shows a GitHub repository page for `hatemhobbies / terraform_k8s_aws_postgres_spring_ansible`. The repository is public. The main navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, and Security. Below the navigation, a dropdown menu shows the branch `main` selected. The code editor displays the `vars_ip_controller.tf` file, which contains the following Terraform configuration:

```
1 variable "controller_ip" {
2   type = string
3   default = "10.240.0.10"
4 }
```

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

```
1 resource "aws_instance" "k8s_controller" {
2   depends_on = [aws_vpc.k8s]
3   ami         = "${data.aws_ami.ubuntu.id}"
4   associate_public_ip_address = true
5   key_name    = "${aws_key_pair.k8s.key_name}"
6   vpc_security_group_ids = ["${aws_security_group.k8s.id}"]
7   instance_type = "t2.micro"
8   private_ip    = "${var.controller_ip}"
9   user_data     = "name=k8s_controller"
10  subnet_id    = "${aws_subnet.k8s.id}"
11  source_dest_check = false
12
13  tags = {
14    Name = "k8s_controller"
15  }
16
17  provisioner "remote-exec" {
18    connection {
19      host = self.public_ip
20      user = "ubuntu"
21      private_key = file("~/ssh/k8s_rsa")
22    }
23    inline = ["echo 'Instance ${self.public_dns} is up!'"]
24  }
25
26
27  provisioner "local-exec" {
28    command = <<EOT
29    scp -o StrictHostKeyChecking=no -i ~/ssh/k8s_rsa* ~/ssh/k8s_rsa ubuntu@${aws_instance.k8s_controller.public_ip}:~/home/ubuntu/.ssh
30    scp -r -i ~/ssh/k8s_rsa ./k8s ubuntu@${aws_instance.k8s_controller.public_ip}:~/home/ubuntu/k8s
31    EOT
32  }
33
34
35
36  provisioner "local-exec" {
37    command = "ANSIBLE_HOST_KEY_CHECKING=False ansible-playbook -T 300 -i '${self.public_ip}', --extra-vars 'private_ip=${self.private_ip} hostname=${split('.', ${self.public_ip})[0]} ${self.private_ip}'"
38  }
39 }
```

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

By 1 contributor

31 lines (27 sloc) | 1.13 KB

Raw · Blame · ⚙ · ⌂ · ⏮

```
1 resource "aws_instance" "k8s_worker" {
2   depends_on          = [aws_instance.k8s_controller]
3   count               = "${length(var.worker_ips)}"
4   ami                 = "${data.aws_ami.ubuntu.id}"
5   associate_public_ip_address = true
6   key_name            = "${aws_key_pair.k8s.key_name}"
7   vpc_security_group_ids = ["${aws_security_group.k8s.id}"]
8   instance_type       = "t2.micro"
9   private_ip          = "${var.worker_ips[count.index]}"
10  user_data           = "name=k8s_worker-${count.index+1}|pod-cidr=${var.worker_pod_cidrs[count.index]}"
11  subnet_id          = "${aws_subnet.k8s.id}"
12  source_dest_check   = false
13
14  tags = {
15    Name = "k8s_worker-${count.index+1}"
16  }
17
18 provisioner "remote-exec" {
19   connection {
20     host = self.public_ip
21     user = "ubuntu"
22     private_key = file("~/ssh/k8s_rsa")
23   }
24
25   inline = ["echo 'Instance ${self.public_dns} is up!'"]
26 }
27
28 provisioner "local-exec" {
29   command = "ANSIBLE_HOST_KEY_CHECKING=False ansible-playbook -T 300 -i '${self.public_ip}', --private-key ~/ssh/k8s_rsa ../ansible/k8s_node_playbook.yml"
30 }
```

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@Hatem-Priv-MBP:~/Desktop/CapstoneProject/terraform_k8s_aws_postgres_spring_ansible/terraform$ source ../../aws_key.sh  
hatem@Hatem-Priv-MBP:~/Desktop/CapstoneProject/terraform_k8s_aws_postgres_spring_ansible/terraform$ pwd  
/Users/hatem/Desktop/CapstoneProject/terraform_k8s_aws_postgres_spring_ansible/terraform  
hatem@Hatem-Priv-MBP:~/Desktop/CapstoneProject/terraform_k8s_aws_postgres_spring_ansible/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_v4 = (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)  
}
```

```
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\nscp -i ~/.ssh/k8s_rsa ubuntu@223.241.225:~/kube/config ~/kube/config\nsed 's/server: .*/$server: https://\\18.223.241.225:6443/g' ~/kube/config > ~/kube/config.tmp && mv -f ~/kube/config.tmp ~/kube/config\\n"]  
null_resource.fetch_kubeconfig: Creation complete after 3s [id=5577006791947779410]
```

3

```
Apply complete! Resources: 17 added, 0 changed, 0 destroyed.  
hatem@Hatem-Priv-MBP:~/Desktop/CapstoneProject/terraform_k8s_aws_postgres_spring_ansible/terraform$
```

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

Instances (3) Info									
C Connect Instance state ▾ Actions ▾ Launch Instances ▼									
<input type="text"/> Find instance by attribute or tag (case-sensitive) Clear filters ◀ 1 ▶ ⚙️									
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	⋮
<input type="checkbox"/>	k8s_controller	i-00405f8e8515be392	Running ↻	t2.micro	2/2 checks passed	No alarms +	us-east-2a	ec2-18-223-241-225.us...	
<input type="checkbox"/>	k8s_worker-1	i-046b8e9f231c9bad6	Running ↻	t2.micro	2/2 checks passed	No alarms +	us-east-2a	ec2-3-144-89-30.us-eas...	
<input type="checkbox"/>	k8s_worker-2	i-0b5de38f0a6c7a886	Running ↻	t2.micro	2/2 checks passed	No alarms +	us-east-2a	ec2-3-14-245-18.us-eas...	

Login to the controller

```
AWS Services Q kubernetes X
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-1022-aws x86_64)

* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage

System information as of Sun Nov 6 05:46:00 UTC 2022

System load: 0.48      Users logged in: 0
Usage of /: 43.9% of 7.57GB  IPv4 address for cni0: 10.244.0.1
Memory usage: 77%        IPv4 address for docker0: 172.17.0.1
Swap usage: 0%          IPv4 address for eth0: 10.240.0.10
Processes: 134

25 updates can be applied immediately.
14 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Nov 6 05:32:39 2022 from 71.128.144.186
ubuntu@ip-10-240-0-10:~$ kubectl get nodes
NAME     STATUS   ROLES          AGE    VERSION
ip-10-240-0-10 Ready    control-plane,master 13m   v1.20.12
ip-10-240-0-20 Ready    <none>        10m   v1.20.12
ip-10-240-0-21 Ready    <none>        11m   v1.20.12
ubuntu@ip-10-240-0-10:~$
```

```
aws Services Q kubernetes X
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-1022-aws x86_64)

* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage

System information as of Sun Nov 6 05:46:00 UTC 2022

System load: 0.48      Users logged in: 0
Usage of /: 43.9% of 7.57GB  IPv4 address for cni0: 10.244.0.1
Memory usage: 77%        IPv4 address for docker0: 172.17.0.1
Swap usage: 0%          IPv4 address for eth0: 10.240.0.10
Processes: 134

25 updates can be applied immediately.
14 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Nov 6 05:32:39 2022 from 71.128.144.186
ubuntu@ip-10-240-0-10:~$ kubectl get nodes
NAME     STATUS   ROLES          AGE    VERSION
ip-10-240-0-10 Ready    control-plane,master 13m   v1.20.12
ip-10-240-0-20 Ready    <none>        10m   v1.20.12
ip-10-240-0-21 Ready    <none>        11m   v1.20.12
ubuntu@ip-10-240-0-10:~$ kubectl get pods -n project
NAME                  READY   STATUS    RESTARTS   AGE
wallet-service-74b9bdd546-zpsf7   1/1     Running   0          15m
wallet-service-db-78fb4fb7-8kwsd  1/1     Running   0          15m
ubuntu@ip-10-240-0-10:~$
```

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-78fb4fb7-8kwsd   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
wallet-service-postgres-data-persistent-volume-claim  Bound   wallet-service-postgres-data-persistent-volume
wallet-service-source-code-persistent-volume-claim      Bound   wallet-service-source-code-persistent-volume
CAPACITY        ACCESS MODES  STORAGECLASS  AGE
5Gi            RWO          manual         45m
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
wallet-service-postgres-data-persistent-volume-claim  Bound   wallet-service-postgres-data-persistent-volume
wallet-service-source-code-persistent-volume-claim      Bound   wallet-service-source-code-persistent-volume
CAPACITY        ACCESS MODES  STORAGECLASS  AGE
5Gi            RWO          manual         45m
1Gi            RWO          manual         45m
ubuntu@ip-10-240-0-10:~$ kubectl get pv -n project
NAME           CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM
REASON        AGE
wallet-service-postgres-data-persistent-volume    5Gi       RWO        Retain        Bound   project/wallet-service-postgres-data-persistent-volume-claim
46m
wallet-service-source-code-persistent-volume     1Gi       RWO        Retain        Bound   project/wallet-service-source-code-persistent-volume-claim
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-78fdb4fbb7 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-78fdb4fbb7 1         1         1     43m
ubuntu@ip-10-240-0-10:~$ █
```

Using postman - getting public IP

EC2 > Instances > i-00405f8e8515be392

Instance summary for i-00405f8e8515be392 (k8s_controller) Info

Updated less than a minute ago

Instance ID

i-00405f8e8515be392 (k8s_controller)

IPv6 address

-

Hostname type

IP name: ip-10-240-0-10.us-east-2.compute.internal

Answer private resource DNS name

-

Public IPv4 address

18.223.241.225 | [open address](#) 

Instance state

 Running

Private IP DNS name (IPv4 only)

ip-10-240-0-10.us-east-2.compute.internal

Instance type

t2.micro

Using postman - Creating a wallet

POST <http://18.223.241.225:30000/wallets>

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```
1
2   "name" : "John Smith Wallet".
3
4
```

Body Cookies Headers (9) Test Results 🌐 Status: 201 Created Time:

Pretty Raw Preview Visualize **JSON**

```
1
2   "name": "John Smith Wallet",
3   "balance": 0,
4   "transactionDate": "2022-56-06",
5   "_links": {
6     "self": {
7       "href": "http://18.223.241.225:30000/wallets/1"
8     },
9     "wallet": {
10       "href": "http://18.223.241.225:30000/wallets/1"
11     },
12     "transactions": {
13       "href": "http://18.223.241.225:30000/wallets/1/transactions"
14     }
15 }
```

Using postman - Creating transactions

wallet / <http://18.223.241.225:30000/transactions>

POST [▼](#) <http://18.223.241.225:30000/transactions>

Params Authorization Headers (8) **Body** [●](#) Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON** [▼](#)

```
1
2   "amount": "200.00",
3   "purpose": "Payment for subscription",
4   "wallet": "http://18.223.241.225:30000/wallets/1"
5
6
7
```

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON [▼](#)

```
1
2   "amount": 200.00,
3   "purpose": "Payment for subscription",
4   "transactionDate": "2022-02-06",
5   "_links": {
6     "self": {
7       "href": "http://18.223.241.225:30000/transactions/1"
8     },
9     "transaction": {
10       "href": "http://18.223.241.225:30000/transactions/1"
11     },
12     "wallet": {
13       "href": "http://18.223.241.225:30000/transactions/1/wallet"
14     }
15   }
16 }
```

Using postman - Viewing the wallet

The screenshot shows the Postman application interface. At the top, there is a header bar with several tabs labeled "http" and "GET http". Below the header, the URL "wallet / http://18.223.241.225:30000/wallets/1/transactions" is displayed. The main area shows a "GET" request to the same URL. Below the request, there are tabs for "Params", "Authorization", "Headers (6)", "Body", "Pre-request Script", "Tests", and "Settings". Under "Params", there is a table titled "Query Params" with one row containing "Key" and "Value". In the "Body" section, there are tabs for "Pretty", "Raw", "Preview", "Visualize", and "JSON". The "JSON" tab is selected, showing a JSON response with line numbers from 1 to 20. The response contains an object with "_embedded" and "transactions" properties.

```
1  {
2      "_embedded": {
3          "transactions": [
4              {
5                  "amount": 200.00,
6                  "purpose": "Payment for subscription",
7                  "transactionDate": "2022-03-06",
8                  "_links": {
9                      "self": {
10                          "href": "http://18.223.241.225:30000/transactions/2"
11                      },
12                      "transaction": {
13                          "href": "http://18.223.241.225:30000/transactions/2"
14                      },
15                      "wallet": {
16                          "href": "http://18.223.241.225:30000/transactions/2/wallet"
17                      }
18                  },
19              },
20          ]
21      }
22  }
```

Section 3 - Role Based Access Control

Setting up a pod-admin user with limited access

k8s was copied to the controller

 [hatemhobbies / terraform_k8s_aws_postgres_spring_ansible](#) Public

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

 main  [terraform_k8s_aws_postgres_spring_ansible / k8s /](#)

	hatemhobbies Update hscaling-cpu-policy.yaml
..	
	etcd_backup adding k8s first time
	kustomize Update hscaling-cpu-policy.yaml
	rbac Fixed sed
	customized-metrics-server.yaml adding k8s first time



AWS controller node

```
Processing triggers for man-db (2.9.1-1) ...
ubuntu@ip-10-240-0-10:~$ tree .
.
└── k8s
    ├── customized-metrics-server.yaml
    ├── etcd_backup
    │   ├── README.md
    │   ├── backup_etcd.sh
    │   └── download_etcdctl.sh
    ├── kustomize
    │   ├── README.md
    │   └── base
    │       ├── db
    │       │   ├── deployment.yaml
    │       │   ├── hscaling-cpu-policy.yaml
    │       │   ├── hscaling-memory-policy.yaml
    │       │   ├── network-policy.yaml
    │       │   ├── persistent-volume-claim.yaml
    │       │   ├── persistent-volume.yaml
    │       │   └── service.yaml
    │       ├── kustomization.yaml
    │       └── namespace.yaml
    └── web
        └── service.yaml
.
└── dev
    ├── db
    │   └── persistent-volume-host-path.yaml
    └── kustomization.yaml
.
└── web
    ├── deployment.yaml
    ├── persistent-volume-claim.yaml
    └── persistent-volume.yaml
.
└── rbac
    ├── 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

10 directories, 27 files
ubuntu@ip-10-240-0-10:~$
```

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/0=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=LS0tLS1CRUdJTibDRVJUSUZJQ0FURSBDRVNVNULS0tLS0KTU1JQ2F6Q0NB  
qQU5CZ2txaGtpRz13MEJBUUVGQUFPQ0FROEFNSU1CQ2dLQ0FRRUFwYVNENFlXS1h4dU  
WFUyMTkKSDlnM2J6SDNIVFZodkIvTC9QcER5VmzDmg3T1RnNWs2QVcxOHaWFFzbzU  
09ZTDZCUU9WNGJiNDhYMOU5Z0FOCkNIVUFKNWpMYkJ4RGVtVndhYkdodThyZDdRZEQv  
dsZ05qbVU4TExVT1RkdEtTSFlTZUh60WdkeWloMERvVlZBaQpvaC9mNVFJREFRQUJvC  
3SULxd3I3TlJQeEllaHcxSDFMZE5RmI2QW1vMVAwbGlyazRXM1Q5M3VoMHFXWmtKUz  
dgphvFBZXQ1dHdaL0pPdkJkQn1BT1pqaHY3bTRpVWRUODhkNRiNlhkMU0zbEJvcmb  
1FoR2FuTTBJb1MvZFhac2gKc0xmRUv1ZldxSnJOelkxYmQvOTE0W1YrTER1VE9CVGV  
OK  
+ sed -i s/_CSR__/LS0tLS1CRUdJTibDRVJUSUZJQ0FURSBDRVNVNULS0tLS0  
Qpa04wTUlJQk1qQU5CZ2txaGtpRz13MEJBUUVGQUFPQ0FROEFNSU1CQ2dLQ0FRRUFw  
V0TjJNZG9VRG5SWFUyMTkKSDlnM2J6SDNIVFZodkIvTC9QcER5VmzDmg3T1RnNWs2Q  
ERVJnR2ZCdDJYv09ZTDZCUU9WNGJiNDhYMOU5Z0FOCkNIVUFKNWpMYkJ4RGVtVndhY  
Ylh3U2hPWVRCL2dsZ05qbVU4TExVT1RkdEtTSFlTZUh60WdkeWloMERvVlZBaQpvaC  
0t4ZEI5d2Q5Mvp3SULxd3I3TlJQeEllaHcxSDFMZE5RmI2QW1vMVAwbGlyazRXM1Q5  
M2eS9Pd0p0Vj1UdgphvFBZXQ1dHdaL0pPdkJkQn1BT1pqaHY3bTRpVWRUODhkNRiN  
TVGRINWhZYnR6V1FoR2FuTTBJb1MvZFhac2gKc0xmRUv1ZldxSnJOelkxYmQvOTE0W1  
OK
```

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-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-qvnx9    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 ....
```

```
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
```

```
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-qvnx9    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
```

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

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	59m	kubernetes.io/kube-apiserver-client-kubelet	system:bootstrap:uiph2o	Approved,Issued
pod-admin	2m56s	kubernetes.io/kube-apiserver-client	kubernetes-admin	Approved,Issued

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

```
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

# per kubectl
# permission denied
kubectl --kubeconfig=${user}.conf get pods -A
+ user
+ name
CCDATA='client-certificate-data:.*'
CKDATA='client-key-data:.*'
E1GSUNBVETLS0tLQo=/g
+ kubectl --kubeconfig=pod-admin.conf get pods -A
Error from server (Forbidden): pods is forbidden: User "pod-admin" cannot list resource "pods" in API group "" at the cluster scope
+ kubectl --kubeconfig=pod-admin.conf get pods -n project
Z04KamlVOEXMVU9U2HRLU0hZU2V1eJlnzHitaDEBElZWQW1vaC9mbNVFUREFRQUJBb0lCQVFQVpkczRSnd2Qnh1QOp5bXgyL1M2RFVpb1YwemFXQPI2Y2ZVT1J0NTcxYkx2ddZCa0dPNHh1V1EUQ31CenpnB2kzMGd6dnJ1Mf1T
MFNaCnpLaE04L3hj2kYyeVRVdmFLTWsN0pUNUVnSUJOYjN6c2FLMFhDRGpKa29qHNHBbwVUZC9hTxppSzBUNVNMYMuGKaU5mdVdhanJoTDgrQ0F2dEFkeTLYSVNTNjRJUDBvaDVUSno3WhVhb0lzSkp5WdcySElzzXA4K05hd215
RjDlV2puVlpMeWrjcEl4T2ZTalZzNTJLWLv5zctCb1j3
|SEh1bHJGd1NckcvrTGd1b0pmSl1J6SDhJK1NeblRyUm1J
|ZjUxUTTFKvQzendifYKppZERTdzaYQj1VOpflrn1A3lZq4
|ZTh2SGhwQU5ObjJKYUzvRXZbb0dCQuP6NUNUUW1TwjJI
|Snp2L2FCV0QvsTRNRMvBSt1QcFNvDBxSzVlv3FOY2to
|MOVpZzFVNUx0L2cyYjLRN11KeHBmz31Ybm1Wdh5bFdr
|dXB1Tf3aJv1vkhXyzVaOpvxVdh2ddY0T7d0awpJbzhl
|dUVE Smg2ZD1lemdvM0dQduwzbThxaUVNk1JkNGRkU1zv
# permission should be OK
kubectl --kubeconfig=${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-78fdb4fbb7-8kwsd 1/1     Running   0          63m
wallet-service-74b9bdd546-zpsf7   1/1     Running   0          63m
wallet-service-db-78fdb4fbb7-8kwsd 1/1     Running   0          63m
```

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
```

```
E1GSUNBVEUtLS0tLQo=/g
+ kubectl --kubeconfig=project-admin.conf get pods -A
Error from server (Forbidden): pods is forbidden: User "project-admin" cannot list resource "pods" in API group "" at the cluster scope
+ kubectl --kubeconfig=project-admin.conf get pods -n project
```

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

```
+ kubectl --kubeconfig=project-admin.conf get pods -n project
NAME                  READY   STATUS    RESTARTS   AGE
wallet-service-74b9bdd546-zpsf7   1/1     Running   0          63m
wallet-service-db-78fb4fbb7-8kwsd  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 threshold)

[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

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-78fb4fb7  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

At 1 contributor

19 lines (19 sloc) | 408 Bytes [Raw](#)

```
1 apiVersion: autoscaling/v2beta2
2 kind: HorizontalPodAutoscaler
3 metadata:
4   name: wallet-service-db-memory-scaler
5   namespace: project
6
7   maxReplicas: 4
8   minReplicas: 1
9   metrics:
10    - type: Resource
11      resource:
12        name: memory
13        target:
14          type: Utilization
15          averageValue: 50Mi
16   scaleTargetRef:
17     apiVersion: apps/v1
18     kind: Deployment
19     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-znref7   1/1     Running   0          79m
wallet-service-db-78fb4fbb7-8kwsd  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-78fb4fbb7   1         1         1         81m
ubuntu@ip-10-240-0-10:~$
```

kubectl exec -it -n project pods/wallet-service-db-78fb4fbb7-8kwsd -- /bin/bash

```
ubuntu@ip-10-240-0-10:~/k8s/rbac$ kubectl exec -it -n project pods/wallet-service-db-78fb4fbb7-8kwsd -- /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-78fb4fbb7-8kwsd -- /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 &
```

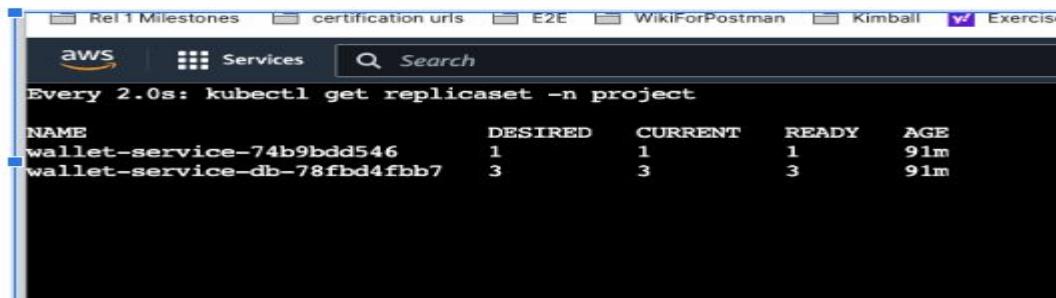
Effect of the load - the database pod scaled up

```
Every 2.0s: kubectl get replicaset -n project
```

NAME	DESIRED	CURRENT	READY	AGE
wallet-service-74b9bdd546	1	1	1	86m
wallet-service-db-78fb4fbb7	4	4	4	86m

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)



```
Re:1 Milestones certification urls E2E WikiForPostman Kimball Exercise
aws Services Search
Every 2.0s: kubectl get replicaset -n project
NAME          DESIRED  CURRENT  READY   AGE
wallet-service-74b9bdd546  1         1         1      91m
wallet-service-db-78fbfd4fbb7  3         3         3      91m
```

```
[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

The screenshot shows a GitHub repository page for a public repository named `hatemhobbies / terraform_k8s_aws_postgres_spring_ansible`. The repository has a single branch named `main`. The file structure under `k8s / etcd_backup /` includes `README.md`, `backup_etcd.sh`, and `download_etcdctl.sh`. The `README.md` file contains instructions for taking an ETCD backup snapshot.

README.md

ETCD take backup snapshot

This section has two scripts: one to download etcdctl and the other to actually do the backup

Usage:

```
sh ./download_etcdctl.sh  
sh ./backup_etcd.sh
```

```
rm -rf *.tar.gz etcd-*/
```

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/etcdd
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/etcdd
+ '[' -d /tmp/etcdd ']'
+ mkdir -p /tmp/etcdd
+ cd /tmp/etcdd
+ 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/kubernetes/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

```
ws_internet_gateway.k8s: Still destroying... [id=igw-05d31ed]
ws_instance.k8s_controller: Still destroying... [id=i-00405f]
ws_internet_gateway.k8s: Still destroying... [id=igw-05d31ed]
ws_instance.k8s_controller: Still destroying... [id=i-00405f]
ws_internet_gateway.k8s: Still destroying... [id=igw-05d31ed]
ws_internet_gateway.k8s: Destruction complete after 1m20s
ws_instance.k8s_controller: Still destroying... [id=i-00405f]
ws_instance.k8s_controller: Destruction complete after 41s
ws_key_pair.k8s: Destroying... [id=k8s]
ws_subnet.k8s: Destroying... [id=subnet-000353a4da3db18c7]
ws_security_group.k8s: Destroying... [id=sg-0fc2a792497c28a5]
ws_key_pair.k8s: Destruction complete after 0s
ws_subnet.k8s: Destruction complete after 0s
ws_security_group.k8s: Destruction complete after 0s
ws_vpc.k8s: Destroying... [id=vpc-0c3793eb20bda78b4]
ws_vpc.k8s: Destruction complete after 1s

Destroy complete! Resources: 17 destroyed.
atem@Hatems-Priv-MBP terraform %
```

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