Sprint 3: Review

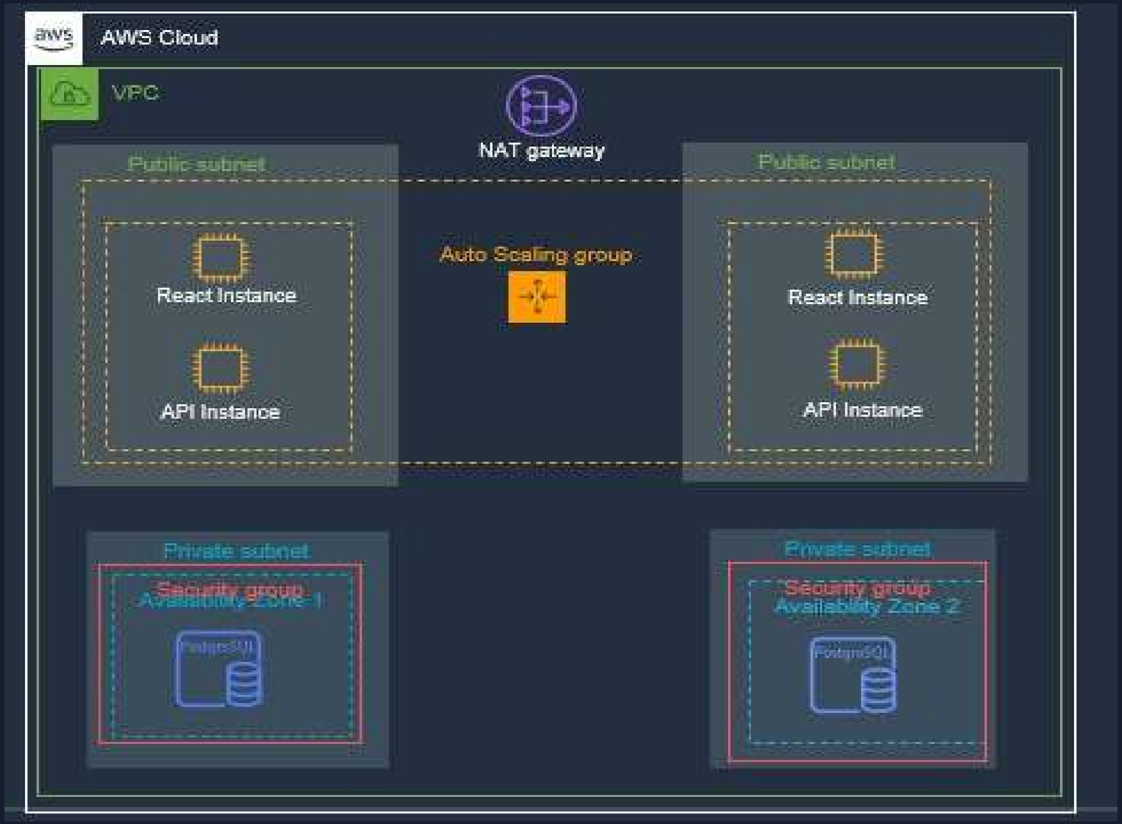
Team Jupiter: Yao Li, Li Chuan Xi Tracy, Hu Min Qiang Trevor, Shank Jay Spencer & Aditi Gupta

Jupiter: Contents to be covered

Jupiter: 1. Key features implemented

- With Terraform, we automate
  - Launch template for React and Api
  - Autoscaling
  - Load balancer
  - RDS
- With Microservice
  - Using gateway to route other service
  - Finishing token in back end for verify user under Microservice

Jupiter: 2. Architecture Diagram



Jupiter: 3. Terraform Implemented

1. Launch template for React and Api
2. RDS
3. Load Balancer
4. Auto Scaling

3-1. command

```
Docker container data volume    -v
```

```
sudo docker run -p 80:8080 springdocker -v /home/ec2-user/aws-bankend-api:/data/java/config -d
```

Summary: use the path in the Docker container link to the host's disk. Because need to
Set up a dynamic database connection for an api project

3-2. command

Command Line Redirection >

For bank api

```
sudo echo
"spring.datasource.url=jdbc:postgresql://'${aws_db_instance.smartbankdb2.endpoint}':5432/'${aw
s_db_instance.smartbankdb2.db_name}'
******
"
> application.properties
```

Summary: On a command line, redirection is the process of using the input/output of a file or command to use it as an input for another file
So with help of > tag Set up a dynamic database connection for bankend api project

3-3. command

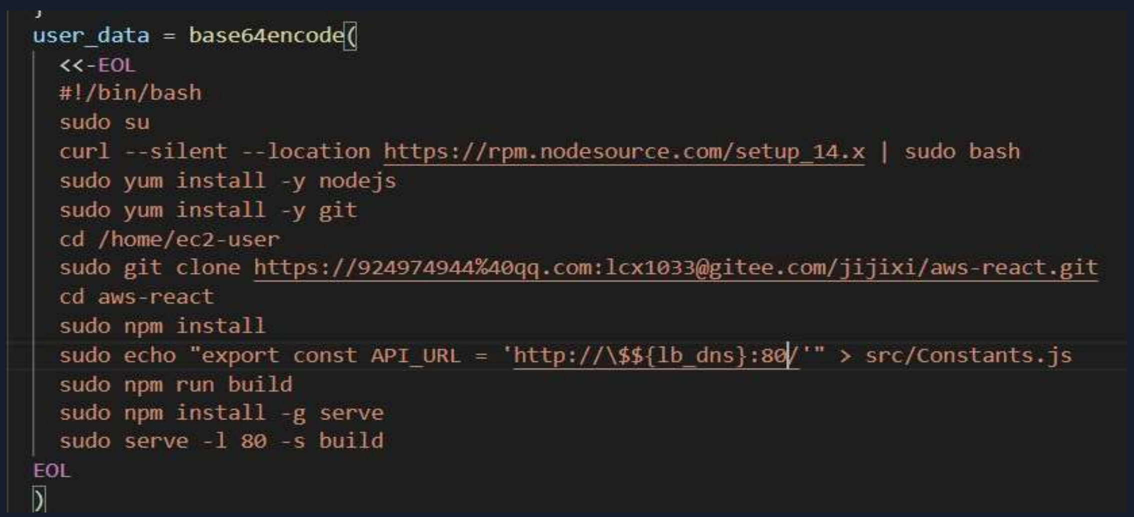
For front part

```
sudo echo "export const API_URL = 'http://${aws_instance.smartbankapi.public_ip}/'" >
src/Constants.js
```

Use > tag to set up a dynamic bankapi url for front part

3-4. command

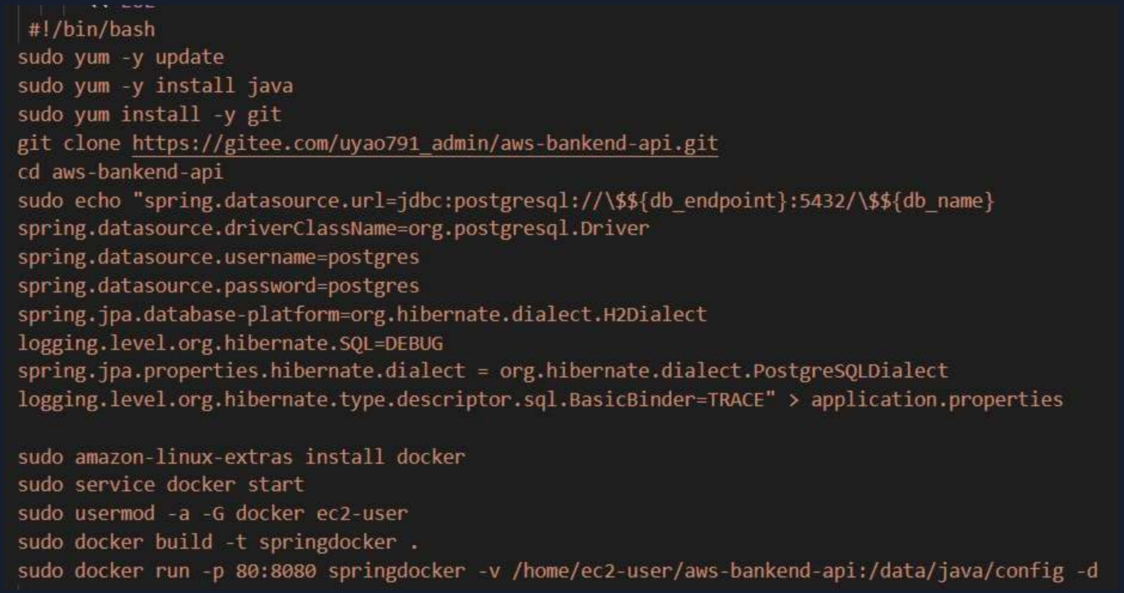
User data for React Launch Template



```
user_data = base64encode(
  <<-EOL
  #!/bin/bash
  sudo su
  curl --silent --location https://rpm.nodesource.com/setup_14.x | sudo bash
  sudo yum install -y nodejs
  sudo yum install -y git
  cd /home/ec2-user
  sudo git clone https://924974944%40qq.com:lcx1033@gitee.com/jijixi/aws-react.git
  cd aws-react
  sudo npm install
  sudo echo "export const API_URL = 'http://\$${lb_dns}:80/'" > src/Constants.js
  sudo npm run build
  sudo npm install -g serve
  sudo serve -l 80 -s build
EOL
)
```

3-5. command

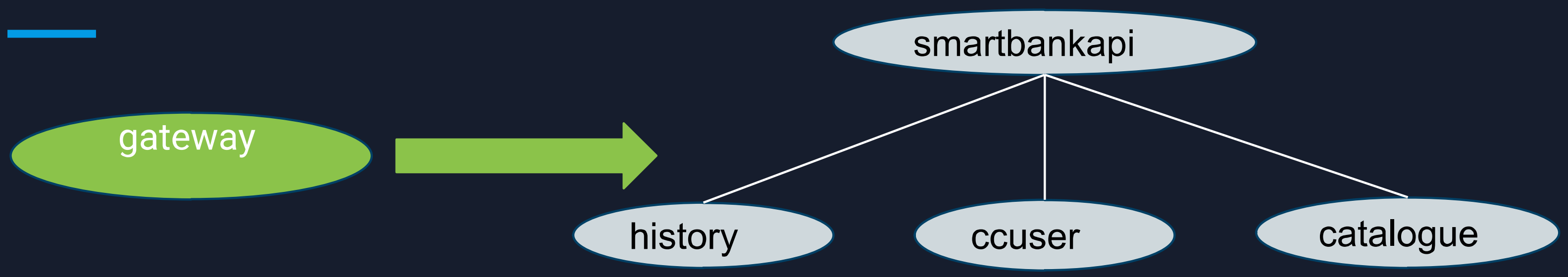
User data for API Launch Template



```
#!/bin/bash
sudo yum -y update
sudo yum -y install java
sudo yum install -y git
git clone https://gitee.com/uyao791_admin/aws-bankend-api.git
cd aws-bankend-api
sudo echo "spring.datasource.url=jdbc:postgresql://\$${db_endpoint}:5432/\$${db_name}
spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.username=postgres
spring.datasource.password=postgres
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
logging.level.org.hibernate.SQL=DEBUG
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE" > application.properties

sudo amazon-linux-extras install docker
sudo service docker start
sudo usermod -a -G docker ec2-user
sudo docker build -t springdocker .
sudo docker run -p 80:8080 springdocker -v /home/ec2-user/aws-bankend-api:/data/java/config -d
```

Jupiter: 4-1. Microservice implemented

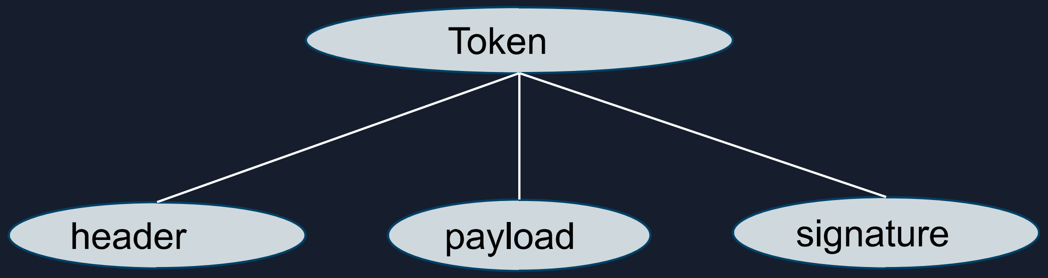


Use the spring-cloud gateway to route all the sub service , and protected real server url not to being exposed to outside

The sub services just use http to communicate with each other ,we will apply the SQS to decouple

the services from each others next week ,then every service just need to focus there part.

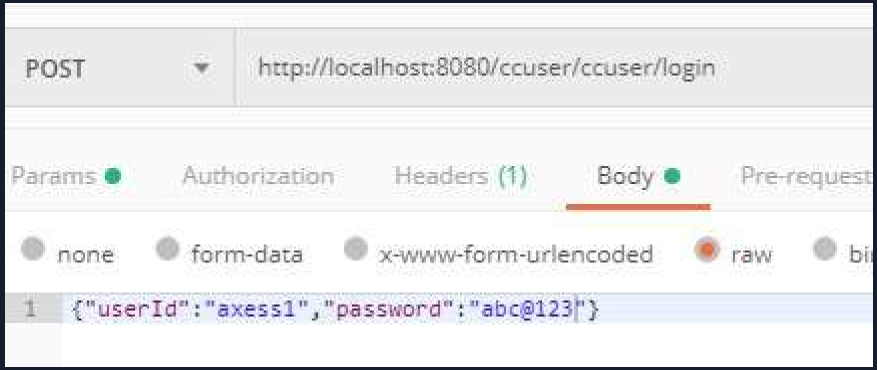
Jupiter: 4-2 JWT（JSON WEB TOKEN）



Header :Declare type such as the JWT and Declare the encryption algorithm such as SHA256
To Encrypted base64 format
Payload : to store the data in base64 format
Signature : include header and payload in Encrypted base64 format and secret

Jupiter: 4-3 Login flow diagram

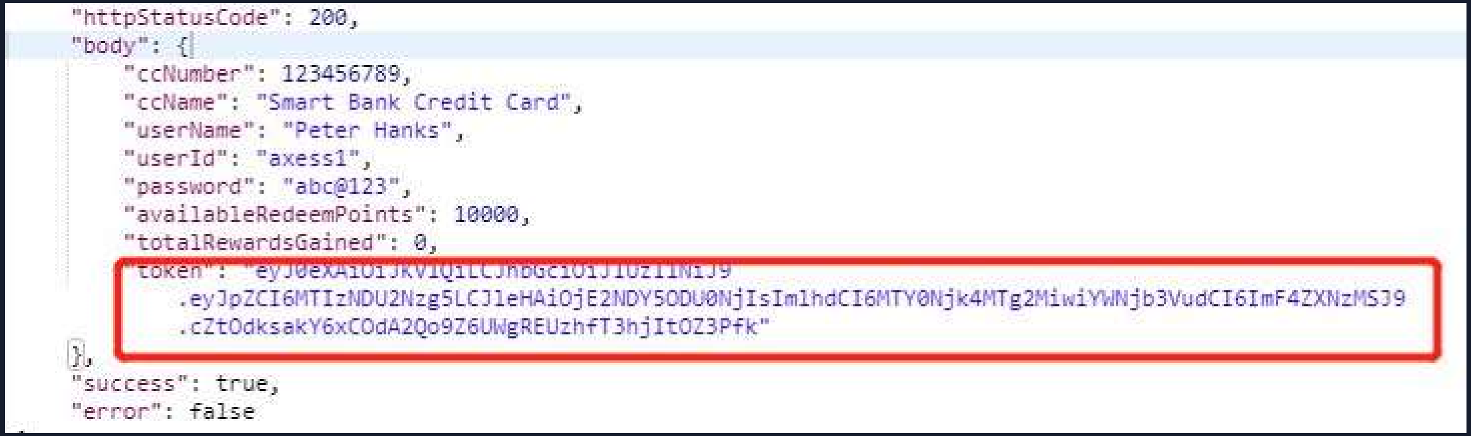
Request for login logic


POST http://localhost:8080/ccuser/ccuser/login

```
{"userId":"axess1","password":"abc@123"}
```

Jupiter: 4-4 the flow diagram of JWT

Response for login logic



```
    "httpStatusCode": 200,
    "body": {
        "ccNumber": 123456789,
        "ccName": "Smart Bank Credit Card",
        "userName": "Peter Hanks",
        "userId": "axess1",
        "password": "abc@123",
        "availableRedeemPoints": 10000,
        "totalRewardsGained": 0,
        "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9
            .eyJpZCI6MTIzNDU2Nzg5LCJleHAiOjE2NDY5ODU0NjIsImlhdCI6MTY0Njk4MTg2MiwiYWNjb3VudCI6ImF4ZXNzMSJ9
            .cZtOdksakY6xCOdA2Qo9Z6UWgREUzhfT3hjItOZ3Pfk"
    },
    "success": true,
    "error": false
```

It will Generate the token for front end ,so front end can use this token to request the backend api

Jupiter: 5. Challenges faced

1. Using the necessary parameter to create the db under particular VPC and subnet group.

2. Routing the VPC, Subnets, Internet Gateway, NAT Gateway, Route Tables with ALB, ASG and DB

Jupiter: 6. Features for the upcoming Sprint

1. To include CloudWatch within launch template to log application logs.
2. Push all the sub service to the EKS
3. Apply the SQS to the bankapi

Jupiter: 7 Learnings from Sprint 3

Shank: Infrastructure as Code (Terraform) can be tough initially when destroying resources that are very connected and applied manually and gets better over time

Trevor: Script using terraform for the first time, it's challenging, but fruitful

Tracy: Though terraform script is challenging for me, learned how to use terraform basically.

Yao :Terraform can help us to Improve work efficiency