Deployment Application:

Project Blossom App

Domain front-end: <https://blossom.isolutions.top/>

Backend API:

- main service: <https://services.isolutions.top/>

- api-gateway: <https://api-gateway.isolutions.top/>

- notification-service: <https://notifications.isolutions.top/>

- integration-service: <https://integrations.isolutions.top/>

CI/CD: Jenkins: <http://103.250.78.50:8080>

Cloudwatch logs: <https://523695643279.signin.aws.amazon.com/console>

Docker hub:

<https://hub.docker.com/repository/docker/bwnguyenvu/swd392-notification/general>

<https://hub.docker.com/repository/docker/bwnguyenvu/swd392-be/general>

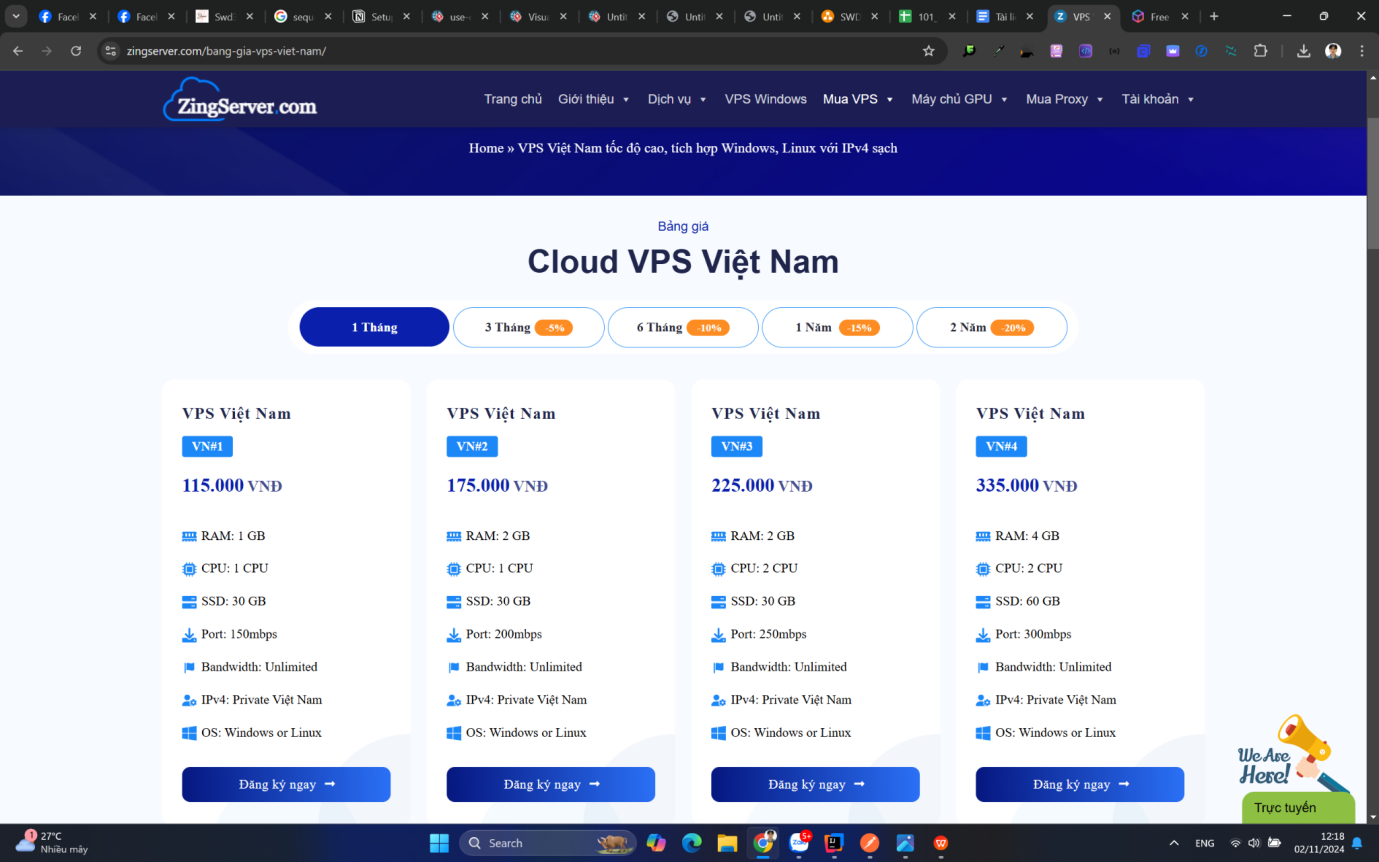
<https://hub.docker.com/repository/docker/bwnguyenvu/integration-service/general>

<https://hub.docker.com/repository/docker/bwnguyenvu/swd392-api-gateway/general>

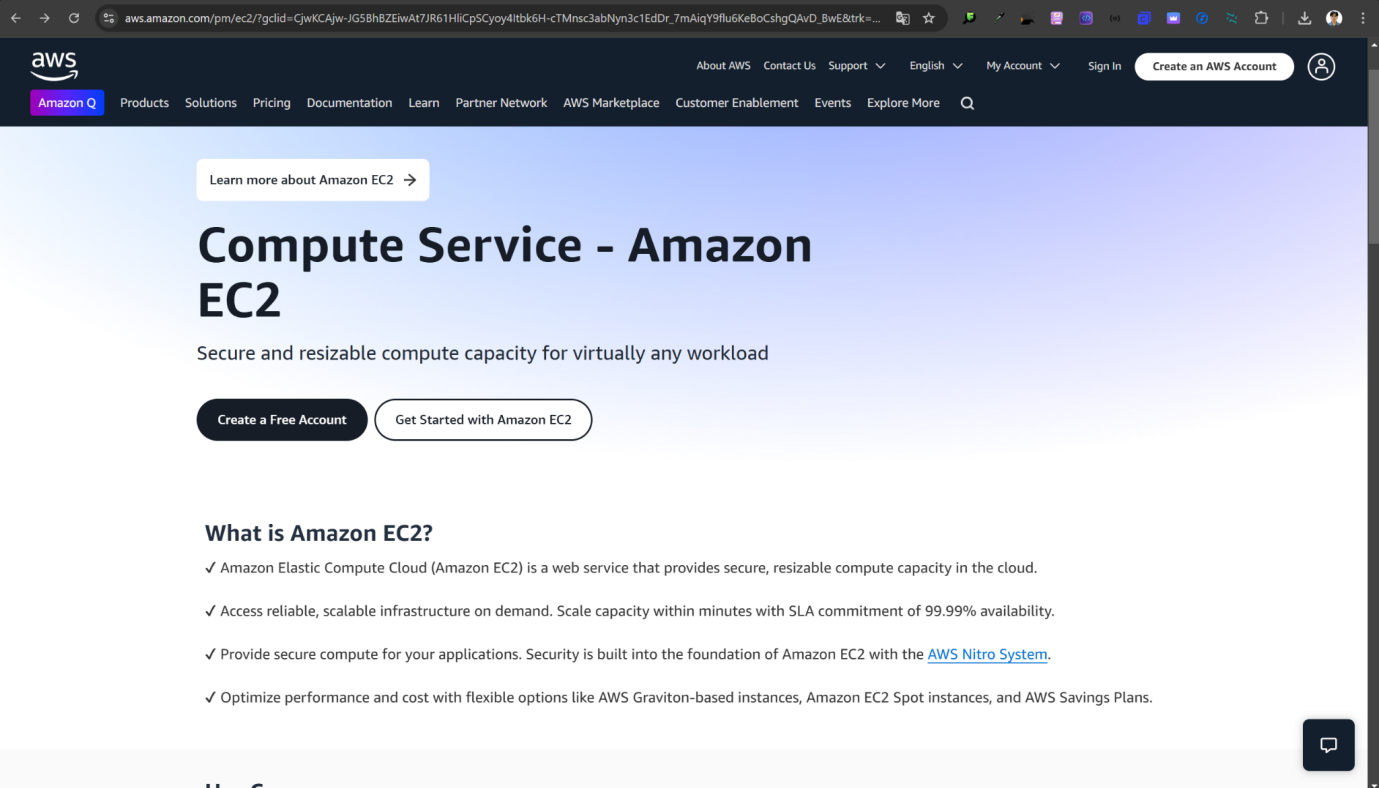
<https://hub.docker.com/r/phatnttse1923/swd392-client/tags>

# Step 1: Buy server

* Buy VPS at zingserver.com



* Buy EC2 at AWS



# 

# Step 2: Config server

Open port in VPS:

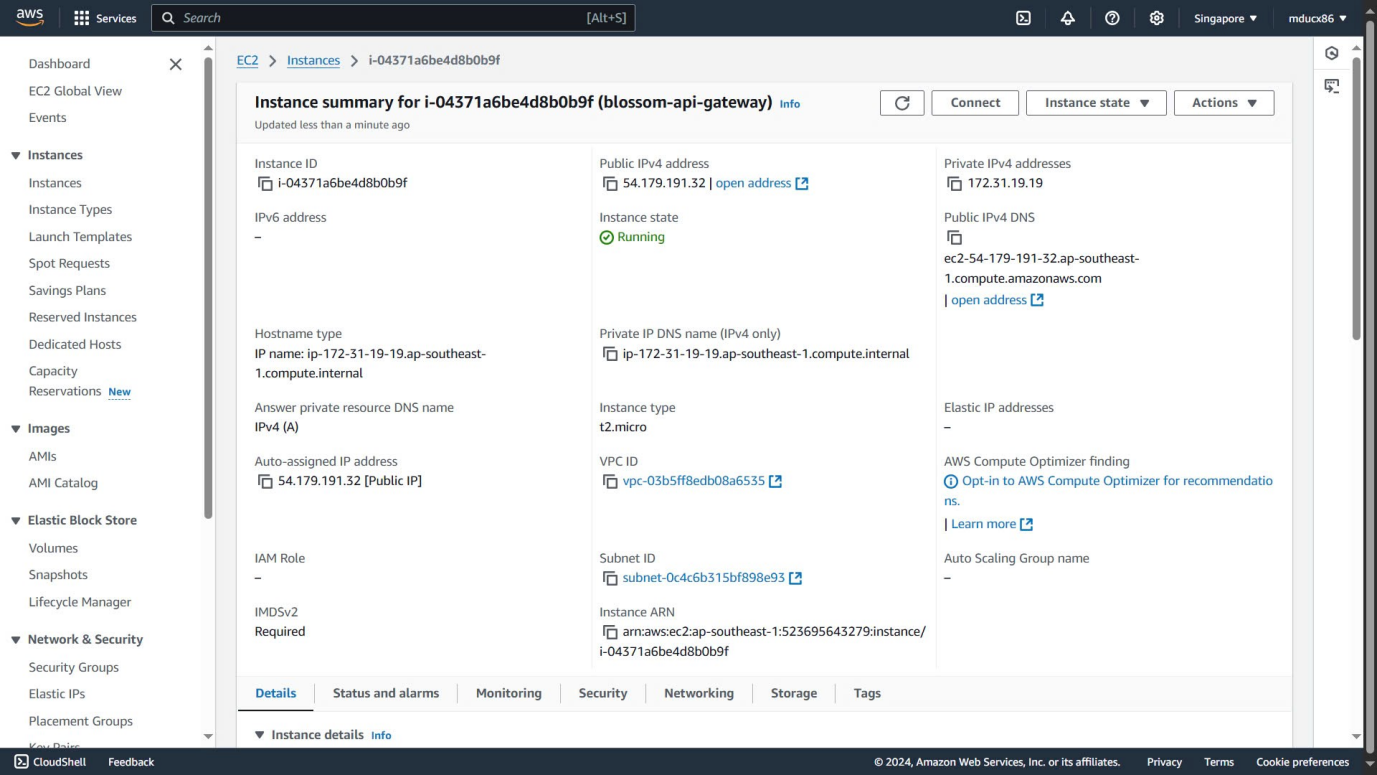
* Client: 3000
* Backend: MainService: 6868, Notification: 8082
* Message Broker: Kafka: 9092, Zookeeper: 2181
* Cache: Redis : 6379
* Database: PostgresQL: 5432
* CI/CD: Jenkins: 8080

Open port in EC2:

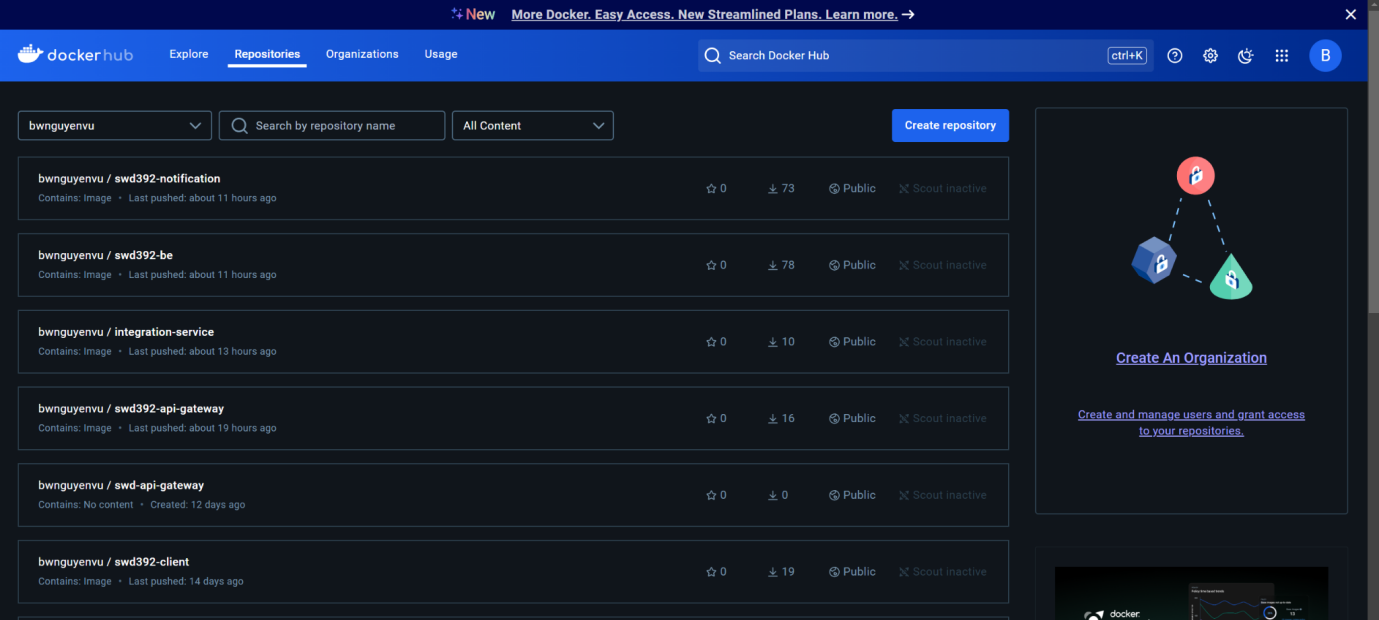
* Backend: Api-gateway: 8888, Integration: 8081

#Console command VPS is: sudo ufw allow {PORT}

If EC2 Configuration in EC2 Config add {Port}



# Step 3: Setup Docker Hub



Create repository : <https://hub.docker.com/repository/create?namespace=bwnguyenvu>

# Step 4: Setup DockerFile, docker-compose.yml, JenkinsFile

* docker-compose.yml

version: '3.9'

services:

notification-service:

build:

context: ./notification-service

ports:

- "8082:8082"

main-service:

build:

context: ./main-service

ports:

- "6868:6868"

api-gateway:

build:

context: ./api-gateway

ports:

- "8888:8888"

image: bwnguyenvu/swd392-api-gateway

integration-service:

build:

context: ./integration-service

ports:

- "8081:8081"

image: bwnguyenvu/integration-service:latest

postgres-db:

image: postgres:latest

environment:

POSTGRES\_USER: postgres

POSTGRES\_PASSWORD: 123

POSTGRES\_DB: resellingFlowerDev

ports:

- "5432:5432"

volumes:

- postgres\_data:/var/lib/postgresql/data

volumes:

postgres\_data:

* Notification DockerFile
* # Use an official OpenJDK runtime as a parent image
* FROM openjdk:17-jdk-slim
* # Set the working directory in the container
* WORKDIR /usr/src/app
* # Copy the jar file from the target directory to the working directory in the container
* COPY target/notification-service-0.0.1-SNAPSHOT.jar notification-service.jar
* # Expose the port the application runs on
* EXPOSE 8082
* # Run the jar file
* ENTRYPOINT ["java", "-jar", "notification-service.jar"]
* MainService DockerFile
* FROM openjdk:17-jdk-slim
* # Set the working directory in the container
* WORKDIR /usr/src/app
* # Copy the jar file from the target directory to the working directory in the container
* COPY target/myflower-0.0.1-SNAPSHOT.jar myflower-0.0.1-SNAPSHOT.jar
* ENV *AWS\_ACCESS\_KEY*=${*AWS\_ACCESS\_KEY*}
* ENV *AWS\_PRIVATE\_KEY*=${*AWS\_PRIVATE\_KEY*}
* ENV *AWS\_REGION*=${*AWS\_REGION*}
* ENV *AWS\_BUCKET\_NAME*=${*AWS\_BUCKET\_NAME*}
* ENV *CONNECTION\_STRING\_POSTGRES\_PROD*=${*CONNECTION\_STRING\_POSTGRES\_PROD*}
* ENV *POSTGRES\_USERNAME*=${*POSTGRES\_USERNAME*}
* ENV *POSTGRES\_PASSWORD*=${*POSTGRES\_PASSWORD*}
* ENV *EMAIL\_USERNAME*=${*EMAIL\_USERNAME*}
* ENV *EMAIL\_PASSWORD*=${*EMAIL\_PASSWORD*}
* ENV *KAFKA\_BOOTSTRAP\_SERVER*=${*KAFKA\_BOOTSTRAP\_SERVER*}
* ENV *REDIS\_HOST*=${*REDIS\_HOST*}
* ENV *REDIS\_PASSWORD*=${*REDIS\_PASSWORD*}
* ENV *GOOGLE\_CLIENT\_ID*=${*GOOGLE\_CLIENT\_ID*}
* ENV *GOOGLE\_CLIENT\_SECRET*=${*GOOGLE\_CLIENT\_SECRET*}
* ENV *JWT\_SECRET*=${*JWT\_SECRET*}
* ENV *PUBLIC\_API*=${*PUBLIC\_API*}
* # Expose the port the application runs on
* EXPOSE 6868
* # Run the jar file
* ENTRYPOINT ["java", "-jar", "myflower-0.0.1-SNAPSHOT.jar"]
* Integration DockerFile  
  # Use an official OpenJDK runtime as a parent image
* FROM openjdk:17-jdk-slim
* # Set the working directory in the container
* WORKDIR /usr/src/app
* # Copy the jar file from the target directory to the working directory in the container
* COPY target/integration-service-0.0.1-SNAPSHOT.jar integration-service-0.0.1-SNAPSHOT.jar
* ENV *CONNECTION\_STRING\_INTEGRATION\_POSTGRES\_PROD*=${*CONNECTION\_STRING\_INTEGRATION\_POSTGRES\_PROD*}
* ENV *POSTGRES\_USERNAME*=${*USERNAME\_INTEGRATION\_POSTGRES\_PROD*}
* ENV *POSTGRES\_PASSWORD*=${*PASSWORD\_INTEGRATION\_POSTGRES\_PROD*}
* ENV *KAFKA\_BOOTSTRAP\_SERVER*=${*KAFKA\_BOOTSTRAP\_SERVER*}
* ENV *REDIS\_HOST*=${*REDIS\_HOST*}
* ENV *REDIS\_PASSWORD*=${*REDIS\_PASSWORD*}
* ENV *REDIS\_PORT*=${*REDIS\_PORT*}
* ENV *REDIS\_DATABASE*=${*REDIS\_DATABASE*}
* # Expose the port the application runs on
* EXPOSE 8081
* # Run the jar file
* ENTRYPOINT ["java", "-jar", "integration-service-0.0.1-SNAPSHOT.jar"]
* API GATEWAY DOCKERFILE

# Use an official OpenJDK runtime as a parent image

FROM openjdk:17-jdk-slim

# Set the working directory in the container

WORKDIR /usr/src/app

# Copy the jar file from the target directory to the working directory in the container

COPY target/api-gateway-0.0.1-SNAPSHOT.jar api-gateway.jar

# Expose the port the application runs on

EXPOSE 8888

# Run the jar file

ENTRYPOINT ["java", "-jar", "api-gateway.jar"]

* JENKINSFILE

pipeline {

agent any

tools {

maven 'my-maven'

}

stages {

stage('Setup Environment Variables') {

steps {

script {

withCredentials([

string(*credentialsId*: 'aws-access-key', *variable*: 'AWS\_ACCESS\_KEY'),

string(*credentialsId*: 'AWS\_PRIVATE\_KEY', *variable*: 'AWS\_PRIVATE\_KEY'),

string(*credentialsId*: 'AWS\_REGION', *variable*: 'AWS\_REGION'),

string(*credentialsId*: 'AWS\_BUCKET\_NAME', *variable*: 'AWS\_BUCKET\_NAME'),

string(*credentialsId*: 'CONNECTION\_STRING\_POSTGRES\_PROD', *variable*: 'CONNECTION\_STRING\_POSTGRES\_PROD'),

string(*credentialsId*: 'CONNECTION\_STRING\_POSTGRES\_DEV', *variable*: 'CONNECTION\_STRING\_POSTGRES\_DEV'),

string(*credentialsId*: 'POSTGRES\_USERNAME', *variable*: 'POSTGRES\_USERNAME'),

string(*credentialsId*: 'POSTGRES\_PASSWORD', *variable*: 'POSTGRES\_PASSWORD'),

string(*credentialsId*: 'PAYOS\_CLIENT\_ID', *variable*: 'PAYOS\_CLIENT\_ID'),

string(*credentialsId*: 'PAYOS\_API\_KEY', *variable*: 'PAYOS\_API\_KEY'),

string(*credentialsId*: 'PAYOS\_CHECKSUM\_KEY', *variable*: 'PAYOS\_CHECKSUM\_KEY'),

string(*credentialsId*: 'EMAIL\_USERNAME', *variable*: 'EMAIL\_USERNAME'),

string(*credentialsId*: 'EMAIL\_PASSWORD', *variable*: 'EMAIL\_PASSWORD'),

string(*credentialsId*: 'KAFKA\_BOOTSTRAP\_SERVER', *variable*: 'KAFKA\_BOOTSTRAP\_SERVER'),

string(*credentialsId*: 'REDIS\_HOST', *variable*: 'REDIS\_HOST'),

string(*credentialsId*: 'REDIS\_PORT', *variable*: 'REDIS\_PORT'),

string(*credentialsId*: 'REDIS\_PASSWORD', *variable*: 'REDIS\_PASSWORD'),

string(*credentialsId*: 'REDIS\_DATABASE', *variable*: 'REDIS\_DATABASE'),

string(*credentialsId*: 'GOOGLE\_CLIENT\_ID', *variable*: 'GOOGLE\_CLIENT\_ID'),

string(*credentialsId*: 'GOOGLE\_CLIENT\_SECRET', *variable*: 'GOOGLE\_CLIENT\_SECRET'),

string(*credentialsId*: 'JWT\_SECRET', *variable*: 'JWT\_SECRET'),

string(*credentialsId*: 'PUBLIC\_API', *variable*: 'PUBLIC\_API'),

string(*credentialsId*: 'CONNECTION\_STRING\_NOTIFICATION\_DEV', *variable*: 'CONNECTION\_STRING\_NOTIFICATION\_DEV'),

]) {

// This block sets up the environment variables

env.*AWS\_ACCESS\_KEY* = *AWS\_ACCESS\_KEY*

env.*AWS\_PRIVATE\_KEY* = *AWS\_PRIVATE\_KEY*

env.*AWS\_REGION* = *AWS\_REGION*

env.*AWS\_BUCKET\_NAME* = *AWS\_BUCKET\_NAME*

env.*CONNECTION\_STRING\_POSTGRES\_PROD* = *CONNECTION\_STRING\_POSTGRES\_PROD*

env.*CONNECTION\_STRING\_POSTGRES\_DEV* = *CONNECTION\_STRING\_POSTGRES\_DEV*

env.*POSTGRES\_USERNAME* = *POSTGRES\_USERNAME*

env.*POSTGRES\_PASSWORD* = *POSTGRES\_PASSWORD*

env.*PAYOS\_CLIENT\_ID* = *PAYOS\_CLIENT\_ID*

env.*PAYOS\_API\_KEY* = *PAYOS\_API\_KEY*

env.*PAYOS\_CHECKSUM\_KEY* = *PAYOS\_CHECKSUM\_KEY*

env.*EMAIL\_USERNAME* = *EMAIL\_USERNAME*

env.*EMAIL\_PASSWORD* = *EMAIL\_PASSWORD*

env.*KAFKA\_BOOTSTRAP\_SERVER* = *KAFKA\_BOOTSTRAP\_SERVER*

env.*REDIS\_HOST* = *REDIS\_HOST*

env.*REDIS\_PORT* = *REDIS\_PORT*

env.*REDIS\_PASSWORD* = *REDIS\_PASSWORD*

env.*REDIS\_DATABASE* = *REDIS\_DATABASE*

env.*GOOGLE\_CLIENT\_ID* = *GOOGLE\_CLIENT\_ID*

env.*GOOGLE\_CLIENT\_SECRET* = *GOOGLE\_CLIENT\_SECRET*

env.*JWT\_SECRET* = *JWT\_SECRET*

env.*PUBLIC\_API* = *PUBLIC\_API*

env.*CONNECTION\_STRING\_NOTIFICATION\_DEV* = *CONNECTION\_STRING\_NOTIFICATION\_DEV*

}

}

}

}

stage('Start Redis and Postgres Containers') {

steps {

script {

// Start Redis if not running

sh '''

if [ $(docker inspect -f '{{.State.Running}}' 03157b32e3f3) = "false" ]; then

docker start 03157b32e3f3;

fi

'''

// Start Postgres if not running

sh '''

if [ $(docker inspect -f '{{.State.Running}}' e311c112c83b) = "false" ]; then

docker start e311c112c83b;

fi

'''

}

}

}

stage('Start Zookeeper') {

steps {

script {

sh '''

if [ $(docker inspect -f '{{.State.Running}}' 4a53c8e9239e) = "false" ]; then

docker start 4a53c8e9239e;

fi

'''

}

}

}

stage('Build Maven for Main Service') {

steps {

checkout scmGit(*branches*: [[*name*: '\*/main']], *extensions*: [], *userRemoteConfigs*: [[*url*: 'https://github.com/BWNguyenVu/swd392-be.git']])

dir('./main-service') {

sh '''

mvn clean install -X \

-Daws.accessKey=${AWS\_ACCESS\_KEY} \

-Daws.privateKey=${AWS\_PRIVATE\_KEY} \

-Daws.region=${AWS\_REGION} \

-Daws.bucketName=${AWS\_BUCKET\_NAME} \

-Dspring.datasource.url=${CONNECTION\_STRING\_POSTGRES\_PROD} \

-Dspring.datasource.username=${POSTGRES\_USERNAME} \

-Dspring.datasource.password=${POSTGRES\_PASSWORD} \

-Dspring.mail.username=${EMAIL\_USERNAME} \

-Dspring.mail.password="${EMAIL\_PASSWORD}" \

-Dspring.kafka.bootstrap-servers=${KAFKA\_BOOTSTRAP\_SERVER} \

-Dspring.data.redis.host=${REDIS\_HOST} \

-Dspring.data.redis.password="${REDIS\_PASSWORD}" \

-Dspring.security.oauth2.client.registration.google.client-id=${GOOGLE\_CLIENT\_ID} \

-Dspring.security.oauth2.client.registration.google.client-secret=${GOOGLE\_CLIENT\_SECRET} \

-Djwt.secret=${JWT\_SECRET} \

-Dpublic.api.url=${PUBLIC\_API}

'''

}

}

}

stage('Build Maven for Notification Service') {

steps {

dir('./notification-service') {

sh '''

mvn clean install -X \

-Dspring.mail.username=${EMAIL\_USERNAME} \

-Dspring.mail.password="${EMAIL\_PASSWORD}" \

-Dspring.kafka.bootstrap-servers=${KAFKA\_BOOTSTRAP\_SERVER} \

-Dpublic.api.url=${PUBLIC\_API} \

-Dspring.datasource.url=${CONNECTION\_STRING\_NOTIFICATION\_DEV} \

-Dspring.datasource.username=${POSTGRES\_USERNAME} \

-Dspring.datasource.password=${POSTGRES\_PASSWORD} \

'''

}

}

}

stage('Build Docker Image for Main Service') {

steps {

script {

sh 'sudo docker build -t bwnguyenvu/swd392-be ./main-service'

}

}

}

stage('Build Docker Image for Notification Service') {

steps {

script {

sh 'sudo docker build -t bwnguyenvu/swd392-notification ./notification-service'

}

}

}

stage('Push Image to Docker Hub for Main Service') {

steps {

script {

withCredentials([string(*credentialsId*: 'dockerhub-pwd', *variable*: 'dockerhubpwd')]) {

sh 'docker login -u bwnguyenvu -p ${dockerhubpwd}'

}

sh 'docker push bwnguyenvu/swd392-be'

}

}

}

stage('Push Image to Docker Hub for Notification Service') {

steps {

script {

sh 'docker push bwnguyenvu/swd392-notification'

}

}

}

stage('Pull Docker Image for Main Service') {

steps {

script {

sh 'sudo docker pull bwnguyenvu/swd392-be:latest'

}

}

}

stage('Pull Docker Image for Notification Service') {

steps {

script {

sh 'sudo docker pull bwnguyenvu/swd392-notification:latest'

}

}

}

stage('Run Docker Container for Main Service') {

steps {

script {

sh 'sudo docker stop swd392-container || true'

sh 'sudo docker rm swd392-container || true'

sh '''

sudo docker run -d -p 6868:6868 \

--name swd392-container \

-e AWS\_ACCESS\_KEY=${AWS\_ACCESS\_KEY} \

-e AWS\_PRIVATE\_KEY=${AWS\_PRIVATE\_KEY} \

-e AWS\_REGION=${AWS\_REGION} \

-e AWS\_BUCKET\_NAME=${AWS\_BUCKET\_NAME} \

-e CONNECTION\_STRING\_POSTGRES\_PROD=${CONNECTION\_STRING\_POSTGRES\_PROD} \

-e CONNECTION\_STRING\_POSTGRES\_DEV=${CONNECTION\_STRING\_POSTGRES\_DEV} \

-e POSTGRES\_USERNAME=${POSTGRES\_USERNAME} \

-e POSTGRES\_PASSWORD=${POSTGRES\_PASSWORD} \

-e PAYOS\_CLIENT\_ID=${PAYOS\_CLIENT\_ID} \

-e PAYOS\_API\_KEY=${PAYOS\_API\_KEY} \

-e PAYOS\_CHECKSUM\_KEY=${PAYOS\_CHECKSUM\_KEY} \

-e EMAIL\_USERNAME=${EMAIL\_USERNAME} \

-e EMAIL\_PASSWORD="${EMAIL\_PASSWORD}" \

-e KAFKA\_BOOTSTRAP\_SERVER=${KAFKA\_BOOTSTRAP\_SERVER} \

-e REDIS\_HOST=${REDIS\_HOST} \

-e REDIS\_PORT=${REDIS\_PORT} \

-e REDIS\_PASSWORD="${REDIS\_PASSWORD}" \

-e REDIS\_DATABASE=${REDIS\_DATABASE} \

-e GOOGLE\_CLIENT\_ID=${GOOGLE\_CLIENT\_ID} \

-e GOOGLE\_CLIENT\_SECRET=${GOOGLE\_CLIENT\_SECRET} \

-e JWT\_SECRET=${JWT\_SECRET} \

-e PUBLIC\_API=${PUBLIC\_API} \

bwnguyenvu/swd392-be:latest

'''

}

}

}

stage('Run Docker Container for Notification Service') {

steps {

script {

sh 'sudo docker stop swd392-notification-container || true'

sh 'sudo docker rm swd392-notification-container || true'

sh '''

sudo docker run -d -p 8082:8082 \

--name swd392-notification-container \

-e EMAIL\_USERNAME=${EMAIL\_USERNAME} \

-e EMAIL\_PASSWORD="${EMAIL\_PASSWORD}" \

-e KAFKA\_BOOTSTRAP\_SERVER=${KAFKA\_BOOTSTRAP\_SERVER} \

-e PUBLIC\_API=${PUBLIC\_API} \

-e CONNECTION\_STRING\_NOTIFICATION\_DEV=${CONNECTION\_STRING\_NOTIFICATION\_DEV} \

-e POSTGRES\_USERNAME=${POSTGRES\_USERNAME} \

-e POSTGRES\_PASSWORD=${POSTGRES\_PASSWORD} \

bwnguyenvu/swd392-notification:latest

'''

}

}

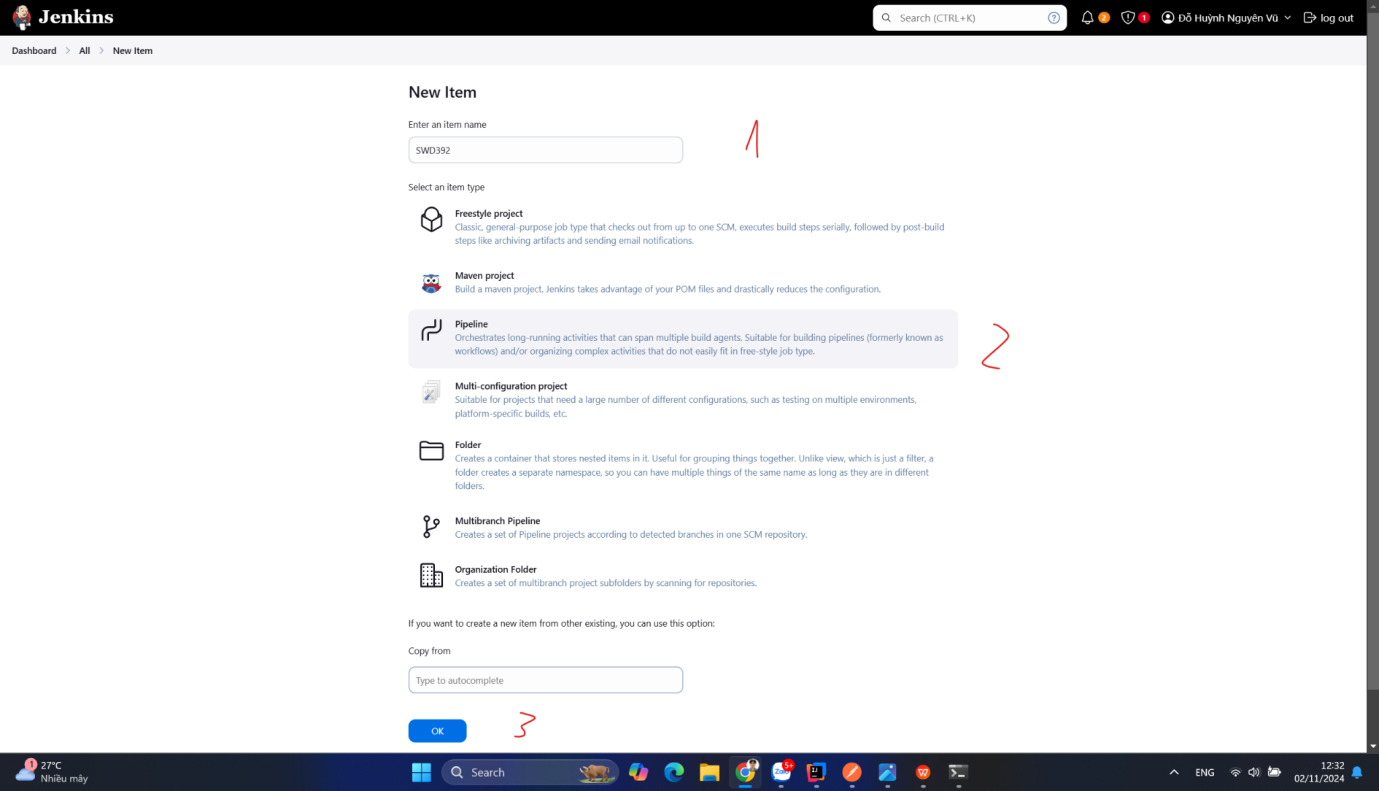
}

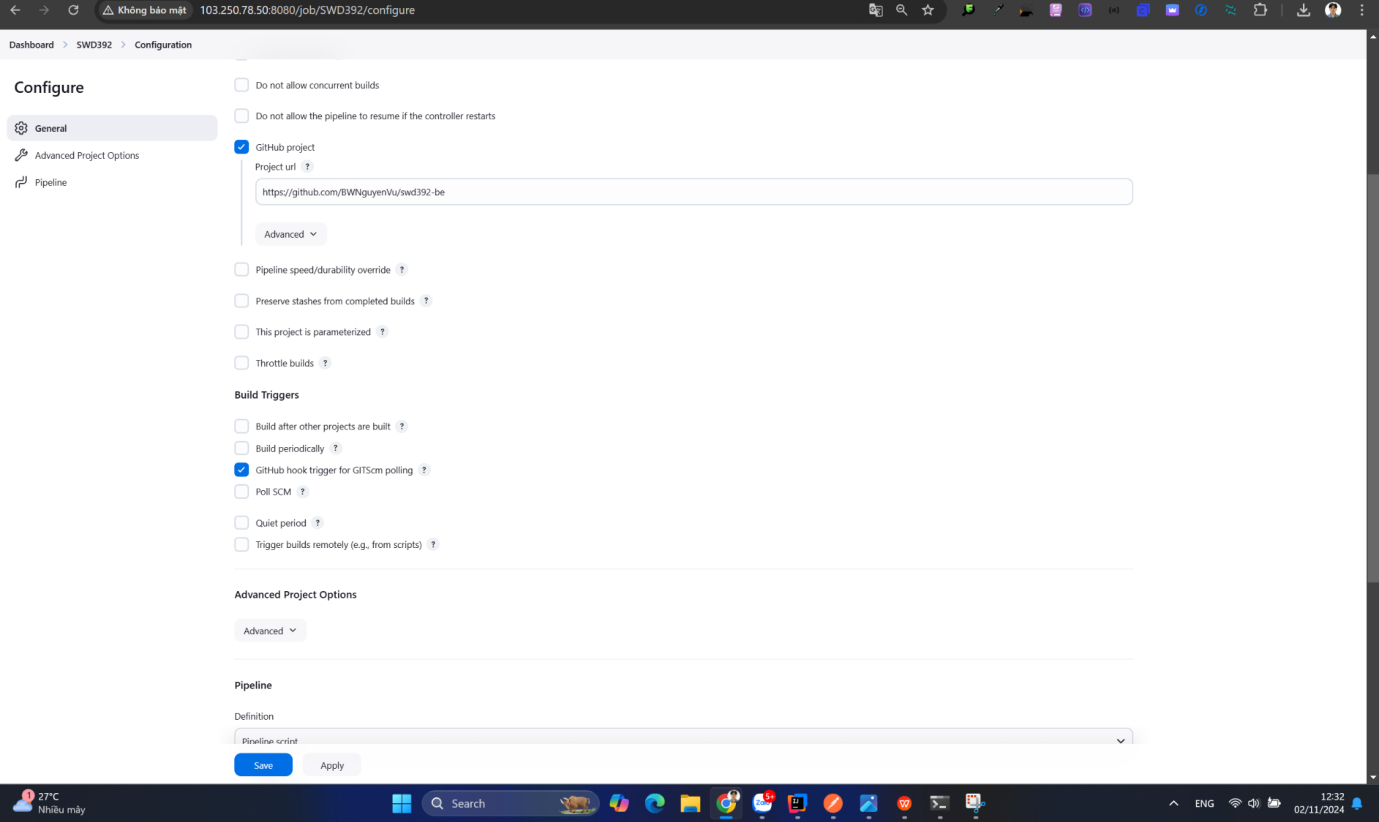
}

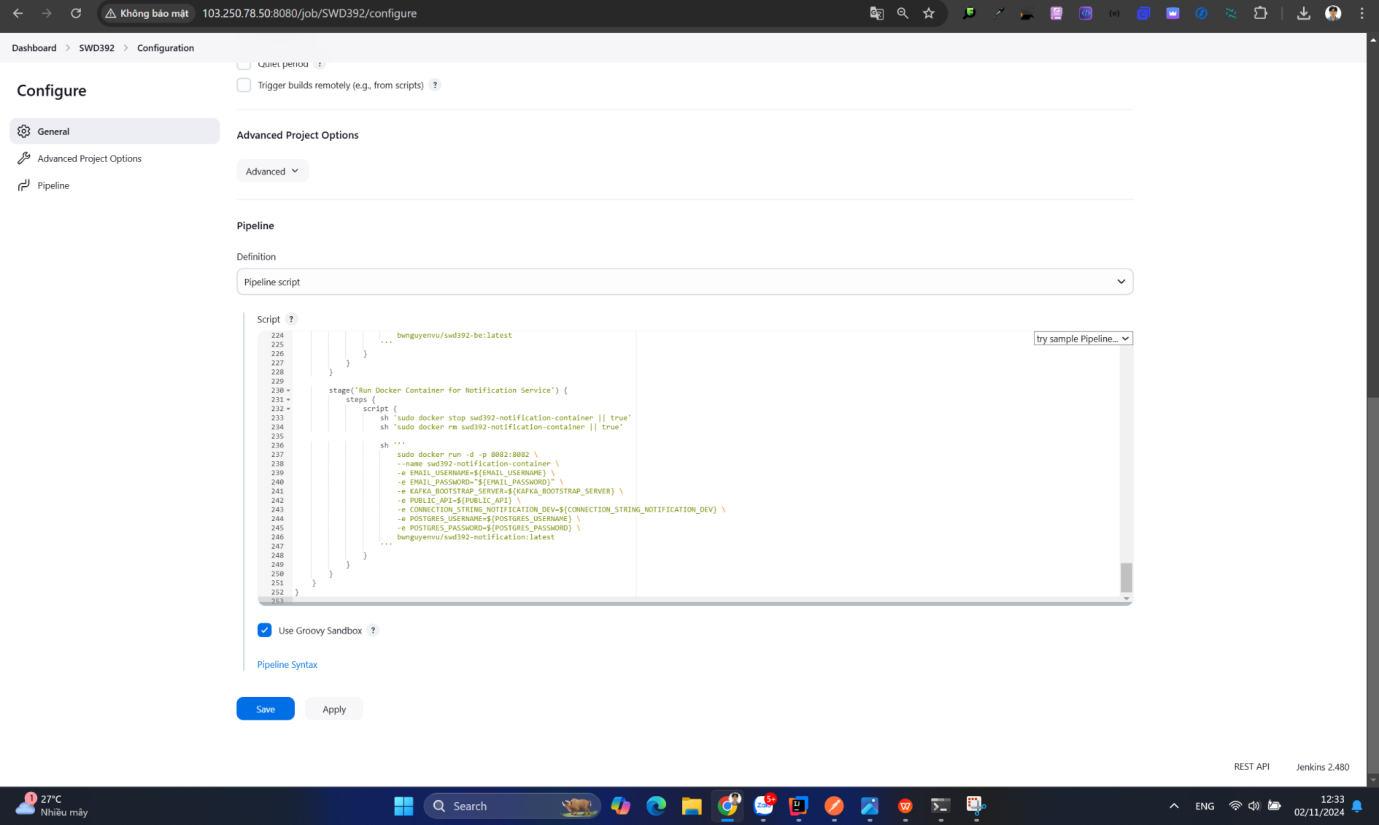
}

# Step 5: Setup Jenkins & Github Webhook

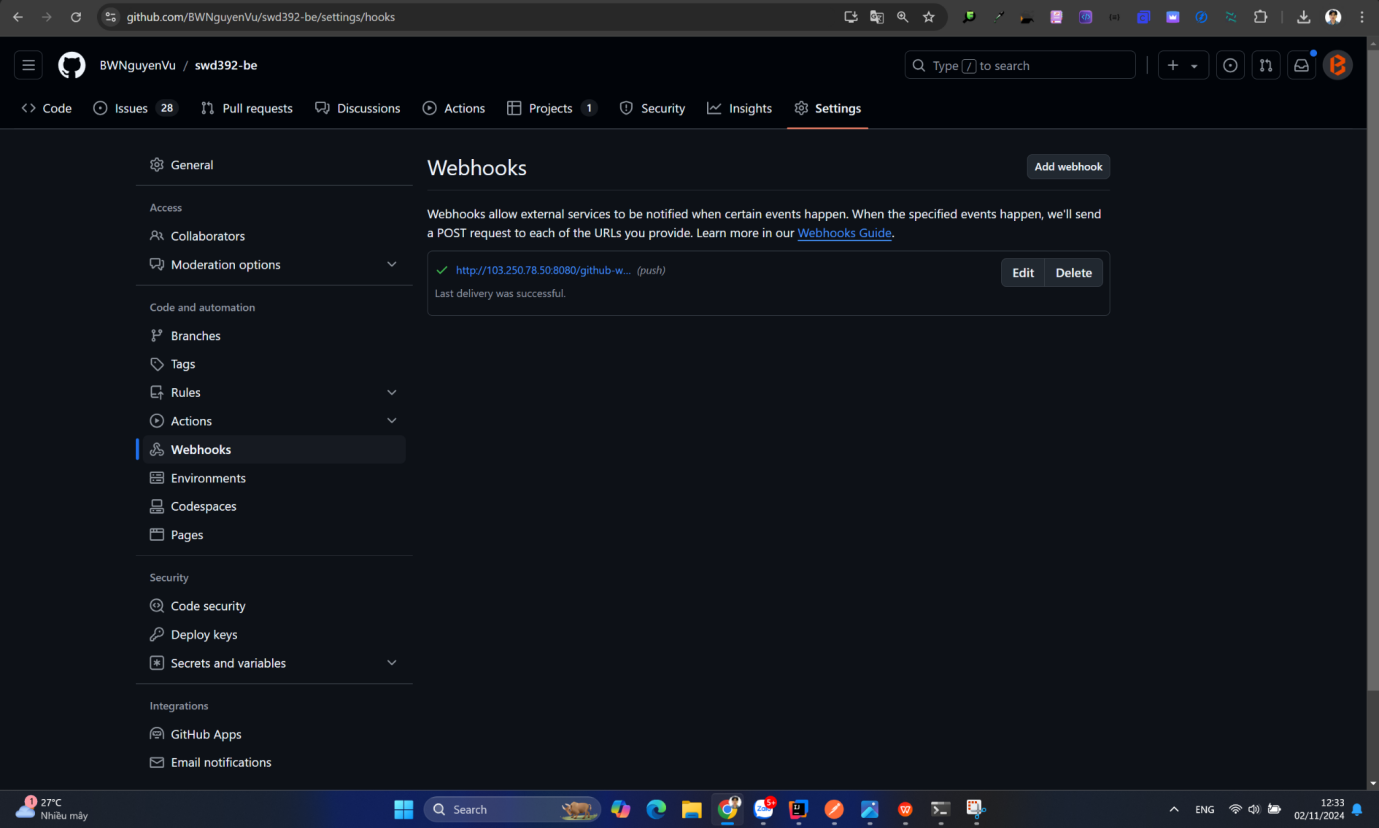
## Setup Jenkins



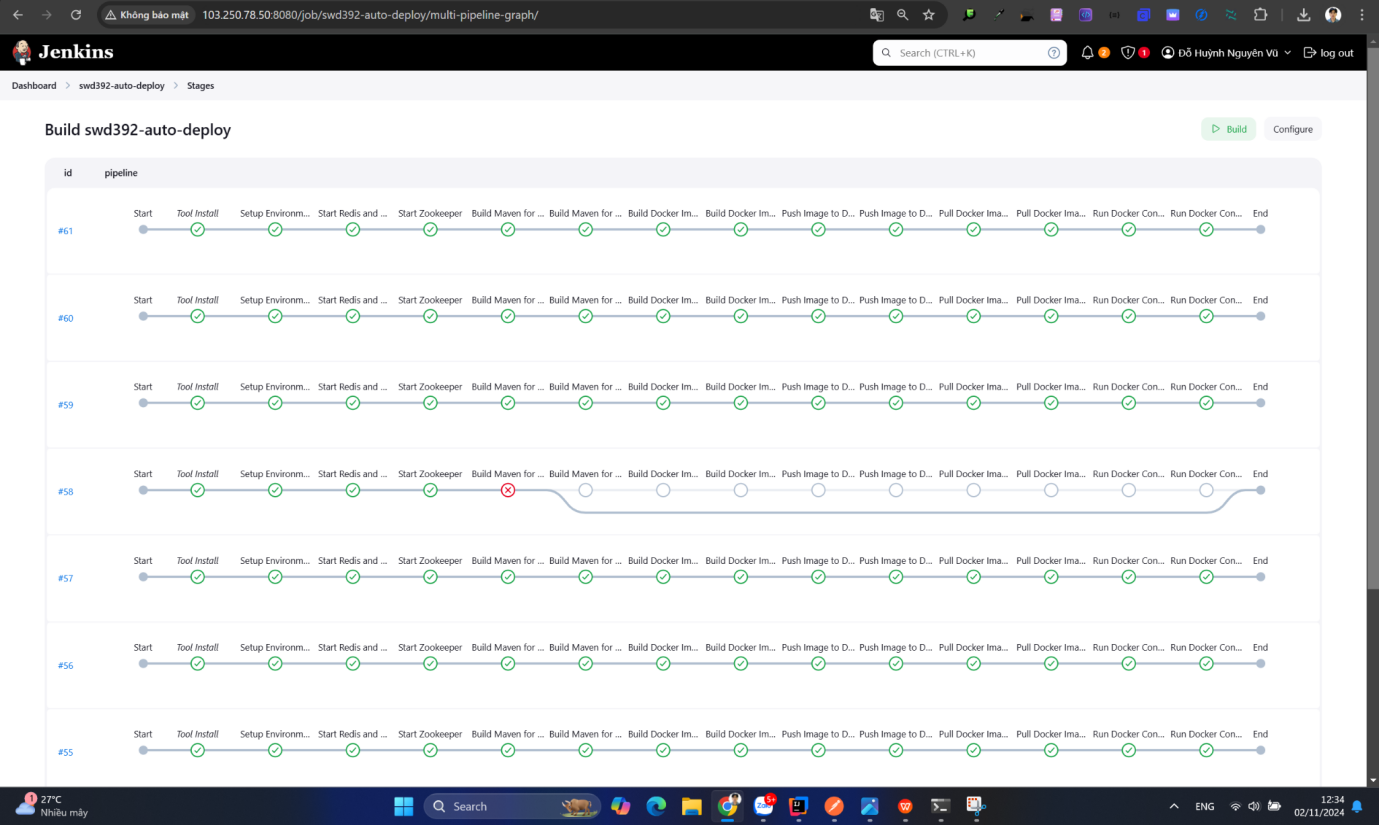




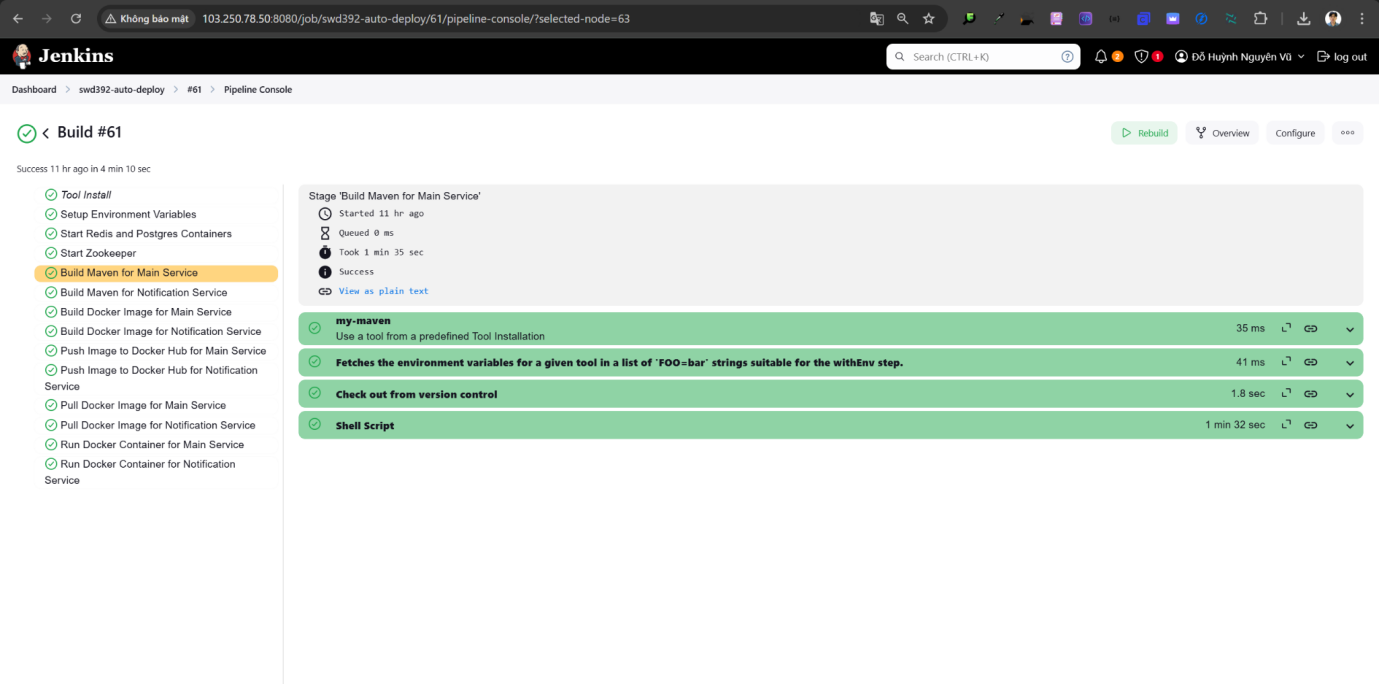
## Setup WEBHOOK



## View Stage



## View Stage Details



# Setup Kafka & Zookeeper

1. Zookeeper

wget<https://downloads.apache.org/zookeeper/zookeeper-3.8.4/apache-zookeeper-3.8.4-bin.tar.gz>

tar -xzf apache-zookeeper-3.8.4-bin.tar.gz

nohup bin/zkServer.sh start > zkServer.log 2>&1 &

**nohup apache-zookeeper-3.8.4-bin/bin/zkServer.sh start > zkServer.log 2>&1 &**

1. Kafka

* wget<https://downloads.apache.org/kafka/3.8.0/kafka_2.13-3.8.0.tgz> tar -xzf kafka\_2.13-3.8.0.tgz cd kafka\_2.13-3.8.0  
    
   listeners=PLAINTEXT://103.250.78.50:9092  
    
   advertised.listeners=PLAINTEXT://103.250.78.50:9092

nohup kafka\_2.13-3.8.0/bin/kafka-server-start.sh kafka\_2.13-3.8.0/config/server.properties > kafka.log 2>&1 &

# Setup HTTPS SSL

### **Configure Nginx for HTTPS**

sudo nano /etc/nginx/sites-available/default

[Update config](https://www.notion.so/Update-config-11f9bab2636f809da7b4ebdfd3a82131?pvs=21)

sudo nginx -t sudo systemctl restart nginx

sudo apt install python3-certbot-nginx

#setup new domain

sudo certbot --nginx

### **Step 3 - Download Certbot [01:20]**

apt-get install -y snapd; snap install --classic certbot;

### **Step 4 - Create Standalone Certificates [02:34]**

**Ensure port 80 and 443 are open on your server**.

Create standalone certificates with this command:

certbot certonly --standalone