RMIT University Vietnam

Course: COSC2767 Systems Deployment and Operations

Semester: 2022B

Assessment: Assignment 1

Author: Du Duc Manh

ID: 3878480

Created date: 12/07/2022

Last modified: 26/07/2022

Acknowledgement: AWS, AWS Java OpenJDK 11, Maven 3.8.6, Tomcat 10.0.22, fontawesome, font-mfizz-2.4.1.

About this project

This project aims to practice deploying a Java Webapp to an AWS EC2 instance through port 8080 using the Tomcat service. The tool for building the WAR file is Maven. The project use AWS CloudFormation to create resources:

- An EC2 instance with a bash script in the UserData property. The instance will run that script in the
- first launch. The VPC and subnet is not declared so AWS will assign the default VPC and a public subnet inside it.
- A key to be associated with the EC2 instance. Since it is created by using CloudFormation, the private key will be stored as a secure string in AWS System Manager Parameter Store.
  A security group that allows inbound data to port 22 for SSH and port 8080 for Tomcat.

The structure of the project

The “deploy-ec2-with-sg.yaml” under the bash-script folder is used to launch AWS resources. AWS CloudFormation uses YAML or JSON files as templates to declare resources.

The project is launched using Linux. AWS CLI needs to be installed to launch the project through the Terminal. See how to install here: https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html. To launch the template from a CLI, put the access key, secret access key, and session token in the credentials file under “.aws” directory. The region to launch the resources could be put there too. Else, you can use "aws configure" to input the mentioned information.

In the script, there is a "git clone" command to download the remote private repo. Therefore, there is a parameter called Token to access the Github access token stored in AWS System Manager Parameter Store. The token must be saved in AWS System Manager Parameter Store before running the AWS CloudFormation template as CloudFormation will check for it and will not start if it cannot find the parameter.

UserData

```
UserData:
  Fn::Base64:
    !Sub |
      #!/bin/bash
      sudo su -
      sudo amazon-linux-extras install -y java-openjdk11
      sudo yum install -y git
      curl "https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/apache-maven-3.8.6-bin.tar.gz" --output /opt/maven.tar.gz
      cd /opt
      tar -xvzf /opt/maven.tar.gz
      curl "https://dlcdn.apache.org/tomcat/tomcat-10/v10.0.22/bin/apache-tomcat-10.0.22.tar.gz" --output /opt/tomcat.tar.gz
      tar -xvzf /opt/tomcat.tar.gz
      mv apache-maven-3.8.6 maven
      mv apache-tomcat-10.0.22 tomcat
      M2=/opt/maven
      M2_HOME=/opt/maven/bin
      JAVA_HOME=$(find /usr/lib -name java-11*)
      export PATH=$PATH:$HOME/bin:$JAVA_HOME:$M2:$M2_HOME
      git clone 'https://rider3458:${Token}@github.com/rider3458/COSC2767_Assignment1_BashScript_AWS_s3878480.git'
      cd COSC2767_Assignment1_BashScript_AWS_s3878480/personalWebsite
      mvn package
      cp target/personalWebsite.war /opt/tomcat/webapps
      /opt/tomcat/bin/startup.sh
```

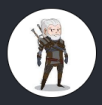
The UserData accepts base64-encoded values. Therefore, the intrinsic function “Fn::Base64:” is used to encode the string. The “!Sub” function is used so the value from the Parameter can be referenced. Here, the value of the parameter “Token” is needed in the UserData.

In the first launch, the script will be executed. Initially, it downloads Java OpenJDK 11 version by AWS and git. It will install Tomcat and Maven compressed files from official websites of them and unarchive the files in "/opt". The directory “/opt” is a placed for installing programs that does not require the build process, which makes it suitable for installing the files for Tomcat and Maven. The files will be decompressed and ready to use when the full directory of the files are called. Then, the paths for Java OpenJDK and Maven will be added to the environment PATH so Maven can run in the webapp root folder to build the WAR artifact. The command “find /usr/lib -name java-11*” will find the path of Java OpenJDK under the “/usr/lib” directory and input it to the variable JAVA. With all needed services installed, the repo will be cloned, and Maven will package the webapp. The artifact will be moved to "webapps" folder under the Tomcat directory. Finally, the instance run a script called "startup.sh" of tomcat to run the server. The website will be hosted under this URL: "URI of the EC2 Instance":8080/"name of the project directory"

Behind this website:

- Maven 3.8.6
- Tomcat 10.0.22
- AWS EC2

Image of the website



Name

Du Duc Manh

 # SID

s3878480

Education

IT (Cloud Technologies) at RMIT University Vietnam

</> IT Skills

ReactJS, React Native, Java, Python, PHP, HTML, CSS, MongoDB, MySQL, AWS, Linux

About this project

This project aims to practice deploying a Java Webapp to an AWS EC2 instance through port 8080 using the Tomcat service. The tool for building the WAR file is Maven. The project use AWS CloudFormation to create resources:

1. An EC2 instance with a bash script in the UserData property. The instance will run that script in the first launch.
2. A key to be associated with the EC2 instance. Since it is created by using CloudFormation, the private key will be stored as secure string in AWS System Manager Parameter Store.
3. A security group that allows inbound data to port 22 for SSH and port 8080 for Tomcat.

In the script, there is a "git clone" command to download the remote private repo. Therefore, there is a parameter called Token to access the Github access token stored in AWS System Manager Parameter Store. The token must be saved in AWS System Manager Parameter Store before running the AWS CloudFormation template as CloudFormation will check for it and will not start if it cannot find the parameter.

In the first launch, the script will be executed. Initially, it downloads Java OpenJDK 11 version by AWS and git. It will install Tomcat and Maven from official websites of them and unarchive the files in "/opt". Then, the paths for Java OpenJDK and Maven will be added to the environment PATH so Maven can run in the webapp root folder to build the WAR artifact. With all needed services installed, the repo will be cloned and Maven will package the webapp. The artifact will be moved to "webapps" folder under the Tomcat directory. Finally, the instance run a script called "startup.sh" of tomcat to run the server. The website will be hosted under this URL: "URI of the EC2 Instance":8080/"name of the project directory"

Behind this website:

1. Maven 3.8.6
2. Tomcat 10.0.22
3. AWS EC2