Dayton Abbott

Student ID: 011125353

D387 – Advanced Java

Deploying A Multithreaded Spring Application to the Cloud Using AWS

1. Step one for deploying the application to the cloud using AWS is to make the application read from the environment, instead of from port 8080. In order to accomplish this the app component typescript file must be changed from the base URL of port 8080 to a HTTP client location path.
2. Next, a Dockerfile must be created in the “target” directory in order to build a docker image. The Docker file lists the base image, the source .jar file path, and the destination .jar file path to be created in the image. Lastly the Dockerfile contains an “entrypoint” clause which specifies the main application that should be run in the container.
3. With the Docker file created, the next step is to build an image of the Docker file. This is done by opening the Docker desktop client and then using a terminal to navigate to the project directory. In the appropriate project directory, the Docker command “docker build -t {filename}” builds a Docker image that can then be located in the Docker desktop client.
4. The next step would be to push the Docker image to DockerHub. This is done by logging in to a docker account on docker hub and generating an access token to use in the terminal. With the access code you can then login to your docker account in the terminal and tag your image with a name and associate it with your account. Lastly, you can push that image to docker hub using the name and your account name.
5. Moving over to AWS, the first step here would be to create a new root user account. Considering that I have already created one, the first step is to create a new IAMs user role of EC2 for SSM using my root user account.
6. With the EC2 role created, next I would navigate to the EC2 dashboard and launch an instance. This includes naming the instance, adding an RSA key pair for connecting to the instance, selecting AWS linux for the application and OS image, creating a security group that allows SSH, HTTPS, and HTTP traffic.
7. After creating an instance, the next step is to navigate to the AWS Systems Manager tab and open the Session Manager page. In Session Manager, I would start a new session using the EC2 instance created in the previous step.
8. Creating the session opens a linux instance where I would run the linux commands to install Docker: “sudo sh” and “yum install -y docker”. Then we start the docker service in the linux terminal using, “service docker start.”
9. The final stage of the deployment using AWS requires pulling the image from docker hub and running it in a container on the AWS EC2 instance. To do this I would log in to docker in the linux terminal using “docker login -u {username}” with my DockerHub username and the password from an access token. Once logged in to docker, I would pull the image from DockerHub using the repository name specified previously with the, “docker pull {username}/{reponame}” command. Finally, I would run the application using, “docker run –name {createappname} -d -p 80:8080 {username}/{reponame}”.
10. The application is now deployed through the AWS cloud and can be accessed via the public IPv4 address listed in AWS for the running instance.