Assignment-DevOps

Deploying a 2-tier Application on AWS Cloud Using Terraform and Jenkins

Dr. Suchintan Mishra
Siksha 'O' Anusandhan University
Bhubaneswar

Problem Statement

Using a Jenkins Pipeline, create 3 stages. Each of these stages has a specific job. The stages are named as such:

1. **Create_Infra**
2. **Deploy_Apps**
3. **Test_Solution**

All the stages subsequently use other technologies such as terraform, AWS, Docker and bash scripting.

Stage-1: Create_Infra

- Use Terraform to create a Public VPC and 2 subnets in it, Namely PUBLIC and PRIVATE subnet. Add all other AWS services in such a way that resources in the PUBLIC subnet are accessible through routes and the PRIVATE subnet resources are restricted.
- Inside the PUBLIC subnet launch and instance called FRONTEND. In the PRIVATE subnet launch another instance called BACKEND.
- Test if the instances can communicate with each other (Although BACKEND is in private subnet, instances within a VPC are able to communicate)
- Using Terraform Provisioner send a script named frontend.sh to FRONTEND and backend.sh to BACKEND.

Creating a docker Application

- **Create a 2-tier application (preferably on git) using any language and tools, run and test the application.**
- **The application must have a frontend and a database connected to it in the backend. It must allow the user to enter some details in the frontend and store the same in a row in the database.**
- **Containerize the application in such a way that the frontend and the backend can be connected on different systems (test this using ec2 instances first then containerize)**
- **Upload the application to DockerHub and save the pull request command.**

Satge 2: **Deploy_Apps**

- **Using terraform provisioners execute the scripts in respective systems:**

frontend.sh: Installs and Configures docker in the FRONTEND instance and runs the containerized frontend in it using the pull request command in previous slide.

backend.sh: Installs and Configures docker in BACKEND instance and runs the containerized backend in it using the pull request command in previous slide.

- **Use remote-exec provisioner to find out if docker has been installed and the application is running in the local system.**

Stage 3: Test_Solution

- **Using terraform output save the public DNS or Public IP of the FRONTEND and display it as the stage is executed.**
- **Using terraform outputs and variables display the exact address with port number for the frontend form application.**
- **Curl the public DNS or IP to check if the frontend containerized application is working.**

Manual Testing

Manually test if the application is running correctly. In the frontend enter some details for an user (Capture a screenshot of it)

Now dive into the database and check if the line has been added to the database in the BACKEND instance or not (Capture a screenshot of it)

Deliverables

1. **Jenkinsfile - With steps in each of the 3 stages mentioned above.**
2. **Terraform file(s) that are required in the process.**
3. **Containerized Application Link (provide your link from dockerhub)**
4. **frontend.sh**
5. **backend.sh**
6. **A Textfile containing terraform outputs.**
7. **Screenshots of Manual Testing**
8. **And any other files that have been used.**