Project Documentation

Project Title: Simpli-project **Completion Date**: 02nd August 2023

Project Summary:

Simpli-project is an excellent tool for developers who want to deploy their applications quickly and easily to the cloud. It automates many steps involved in deployment, so you can focus on developing your application instead of worrying about the infrastructure.

Development Platform: Terraform

Developer: Shubham Thorat
+91 7976438345

Content

Abstract

Project Title: Simpli-project

Simpli-project is a valuable tool for developers who want to deploy applications quickly, easily, and securely to the cloud. It automates many steps in the deployment process, is affordable, and can be scaled to meet the needs of growing businesses.

Here are some of the benefits of using Simpli-project:

- It can help you save time by automating many of the tasks involved in deploying applications to the cloud.
- It is easy to use, even for developers who are not familiar with cloud computing.
- It is secure, with Private keys and Public keys.
- It is scalable, so you can easily add more applications or users as your business grows.

If you are looking for a tool to help you deploy applications to the cloud, Simpli-project is a great option.

Tools Used

1. Terraform
2. AWS Cli
3. Git
4. Jenkins
5. Ansible
6. Docker

Problem Statement

Nowadays, developers need to know a lot of skills to deploy their applications on the cloud. This is because cloud computing is a complex and ever-changing field. Developers need to be able to understand and use a variety of cloud-based services, such as computing, storage, networking, and databases. They also need to be able to manage and monitor their applications in the cloud.

Here are some of the skills that developers need to know to deploy their applications on the cloud:

- Cloud computing fundamentals
- Cloud-based services
- Application management and monitoring
- Security and Compliance
- Networking and storage
- DevOps

Solution specification

This project automated essential steps by creating HCL scripts that automatically provision the server. These scripts also provide the necessary information to configure further, such as the default password for Jenkins and the public IP of the instance. Additionally, an Ansible script was created to automate deployment over Docker.

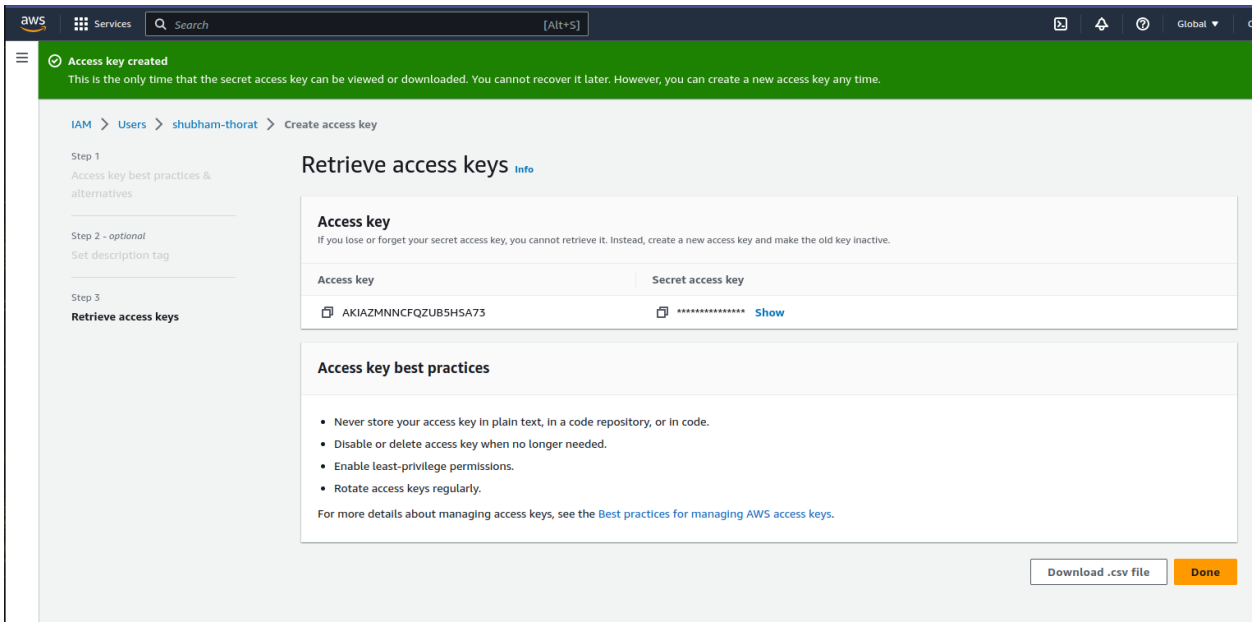
Development

AWS:

aws cli access is needed to create resources in aws cloud. Install aws cli and then get your access key and secret key from the aws console. And set up with the command in the screenshot below.

```
✘ henrry  master -  aws configure
AWS Access Key ID [****************7TNK]: ASIAZMNNCFQZZUWF7TNK
AWS Secret Access Key [****************aCTw]: jwLRRnrFSSWXm+8ARYqnc8q2E8t2R0+lM2mKaCTw
Default region name [us-east-1]: us-east-1
Default output format [None]:
```

Below screenshots shows how to retrieve keys:



Terraform:

Run bellow scripts through command line with commands

$terraform init

$terraform apply -auto-approve

$terraform init:

This command will download all the necessary providers which need to create resources mentioned in below scripts.

$terraform apply -auto-approve:

This command will run all below mentioned scripts and create resources mentioned in below scripts.

demo-sg.tf:

This script will create a security group in aws.In which all the inbound and outbound rules are specified as per requirements.so that we can access the app from outside.The open ports are 22,80,8080.

demo-key.tf:

This script will create a pair of private and public keys. The ke needs to connect with the server through ssh.

dem-ec2.tf :

This script will create an eac2 instance. Attach the key to the instance. And provision the Instance according to the shells scripts written in **installation-scripts.sh.**

Provisioner:

It is a block written in **dem-ec2.tf** which will help the resource created equipped with necessary tools. It will install **Jenkins**, **Ansible**, **Docker** in the instance. As well as install **Ansible plugin** into the jenkins.

For this I have set up the following files.

installation-jenkins.sh: This will install jenkins in the instance.

installation-ansible.sh: This will install ansible in the instance.

installation-docker.sh: This will install docker in the instance. As well as it will set the permissions for `/var/run/docker.sock` so that it won't throw an error while running the container from ansible.

setting.sh: This script will install **Ansible plugin** into the jenkins as well as it will show the default password so that we can unlock jenkins from the browser.

```
aws_instance.demo-instance (remote-exec): ************************************************************
aws_instance.demo-instance (remote-exec): Jenkins default password:  e508da091ee24c1785f96a44acf3f5e8
aws_instance.demo-instance (remote-exec): ************************************************************
aws_instance.demo-instance (remote-exec): Installing ansible from update center
```

Output block:

This block will print the ip of the instance so that we can ssh into the instance to troubleshoot or directly open jenkins gui from the browser with port 8080.

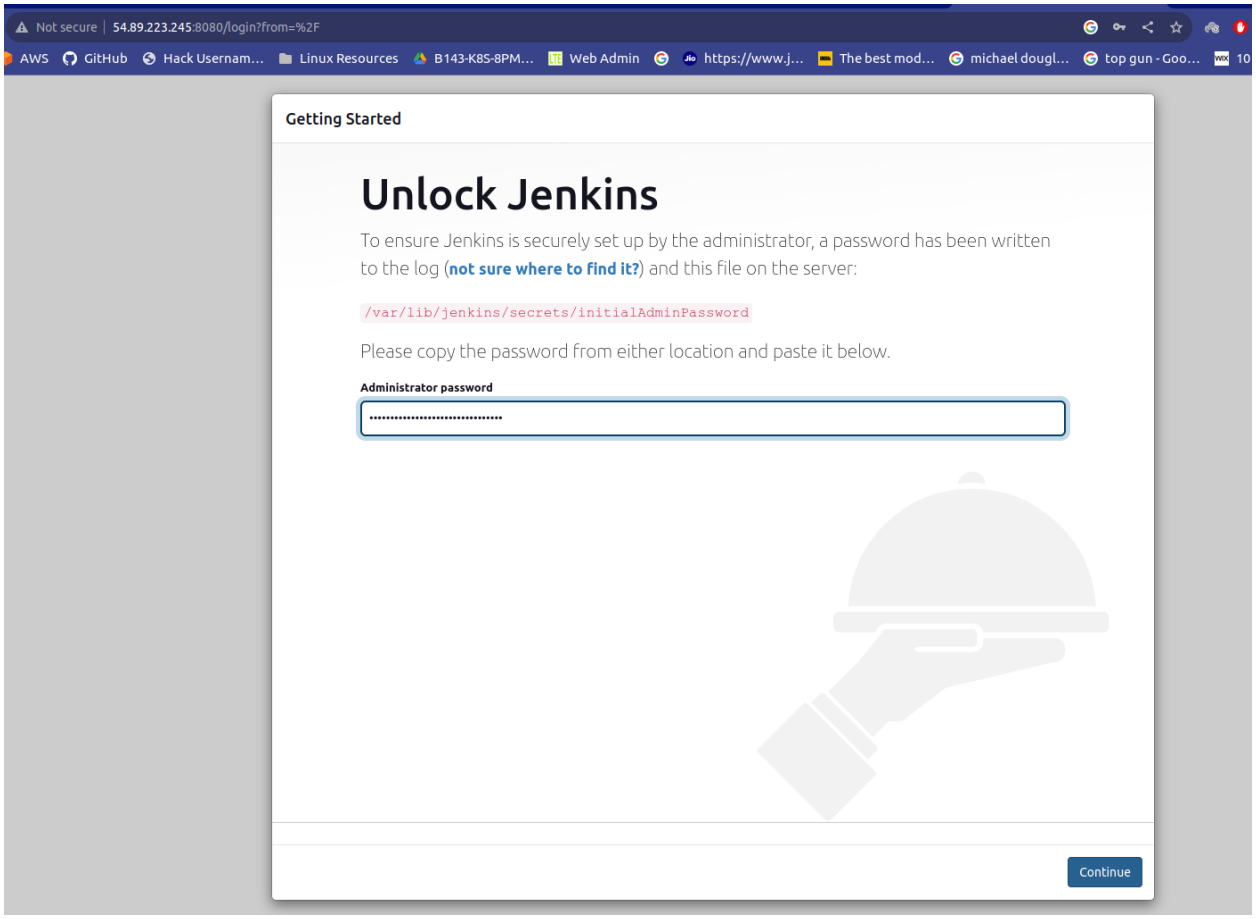
```
Apply complete! Resources: 5 added, 0 changed, 0 destroyed.

Outputs:

instance-ip = "54.89.223.245"
```

Jenkins:

Our instance ready now we have to create the jenkins job. I have made it very easy. just follow the instructions.

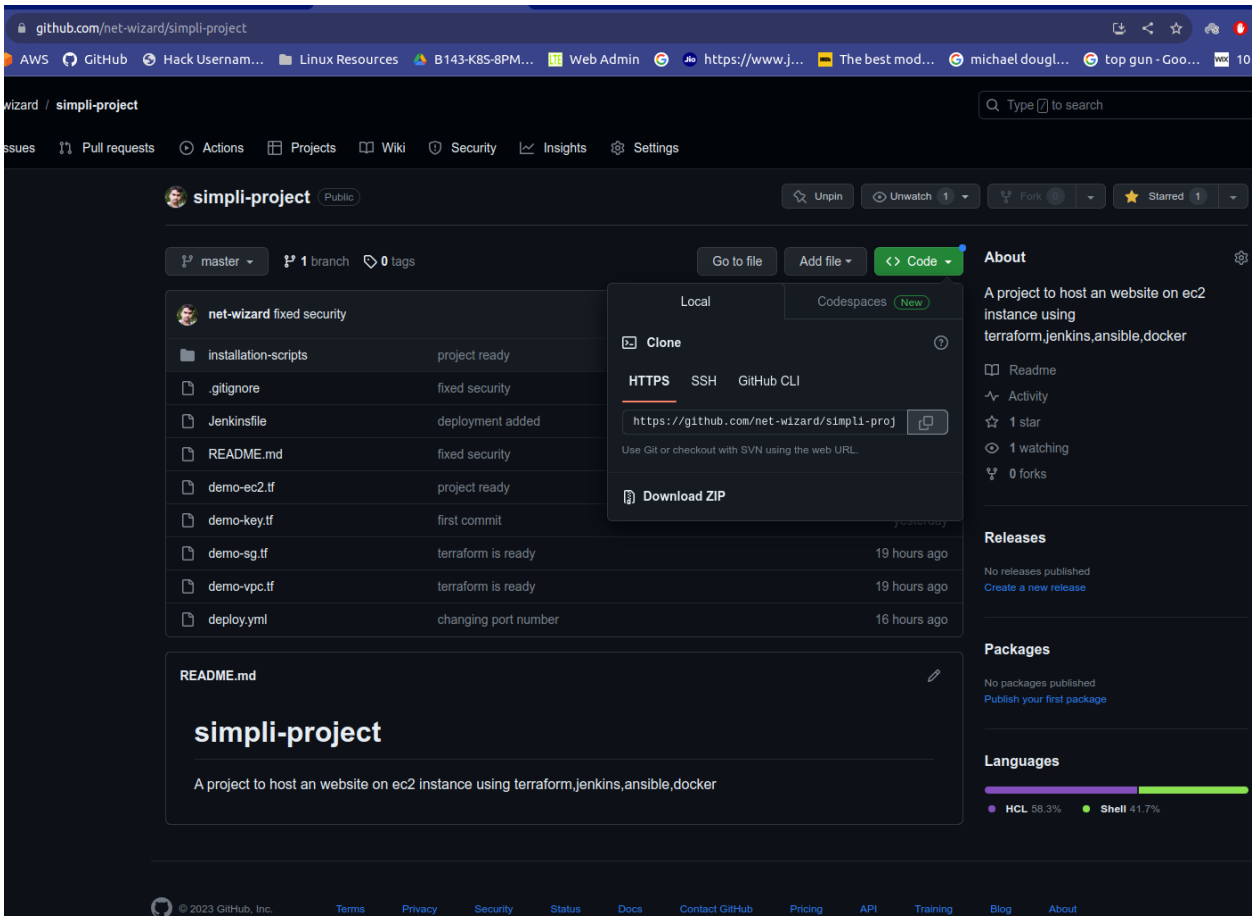
Step 1: Goto the browser. Put the ip from the output and give port 8080.
http://54.89.223.245:8080/ .

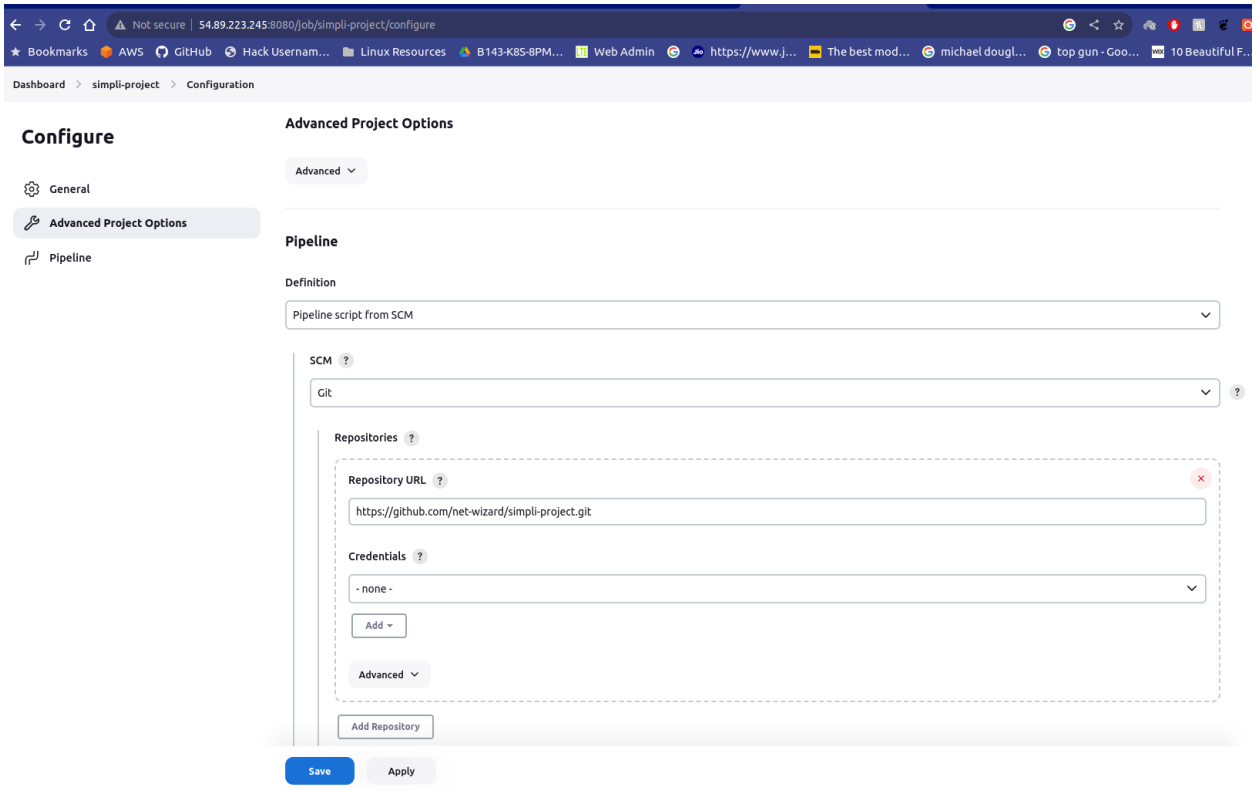
Step 2: Copy the password from the output generated from **$terraform apply -auto-approve** and paste in the jenkins browser. Install suggested plugins. And set up the user.



Step 3: Create a new item from the Jenkins Dashboard, put the project name and select the project type as pipeline.

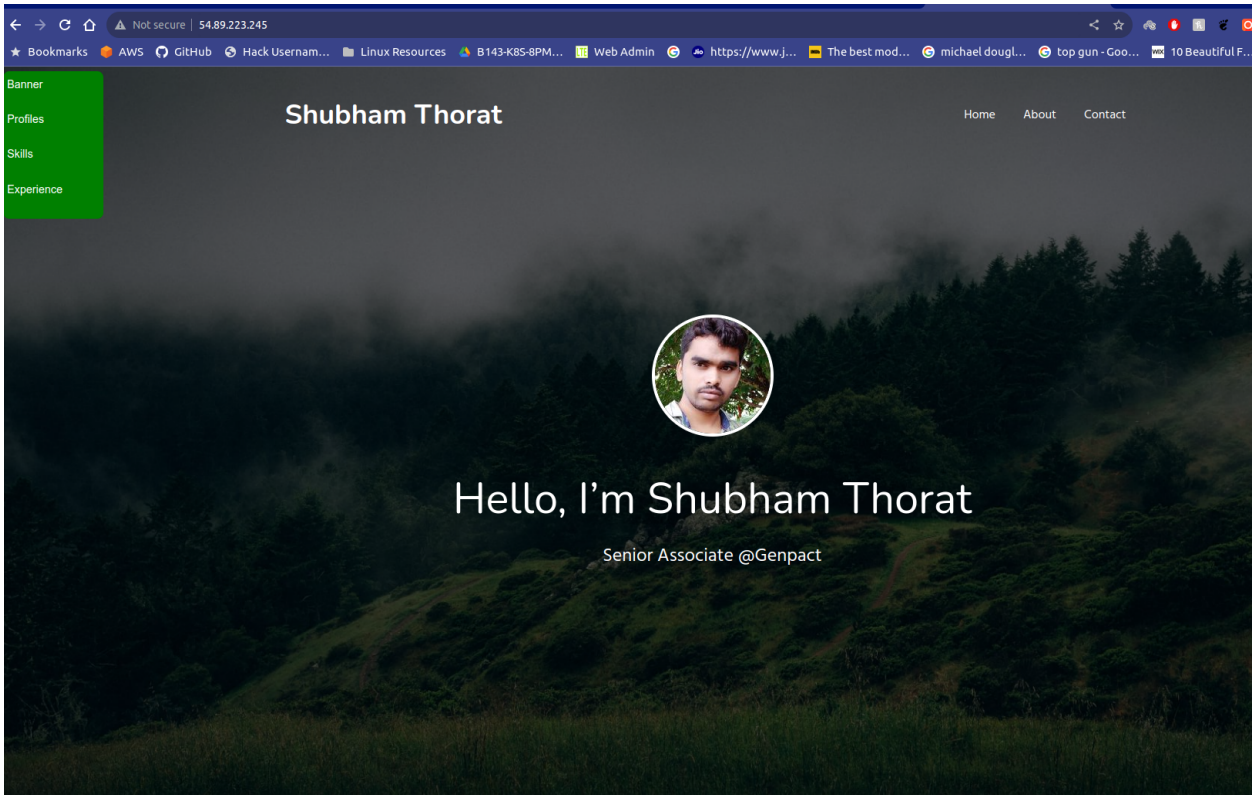
Step 4: In the configure window click on pipeline. It will scroll down to the pipeline. Select the Pipeline definition as Pipeline script from SCM. Select SCM as Git and provide the repository url from github repository and save. And lastly click on **Build now.**





It will fetch all the files from the github repository along with the Jenkins pipeline. The pipeline will be located in the root as filename **Jenkinsfile.** In Jenkinsfile I have written a pipeline script with two stages. In one stage it will fetch the code from the github repository (though only Jenkinsfile is required, In future if we need to build container from the code in repository it will be helpful) and in second Stage there is a ansible syntax to run the **deploy.yml** Which will pull the container from docker hub and run it.

And that's it. Your project is deployed. Go to the Browser, put the public ip and your website will appear.



Ansible:

Ansible here is used to pull the container from the docker hub.
You can modify the docker Image name and port according to your image.

Flow Diagram

Workflow: