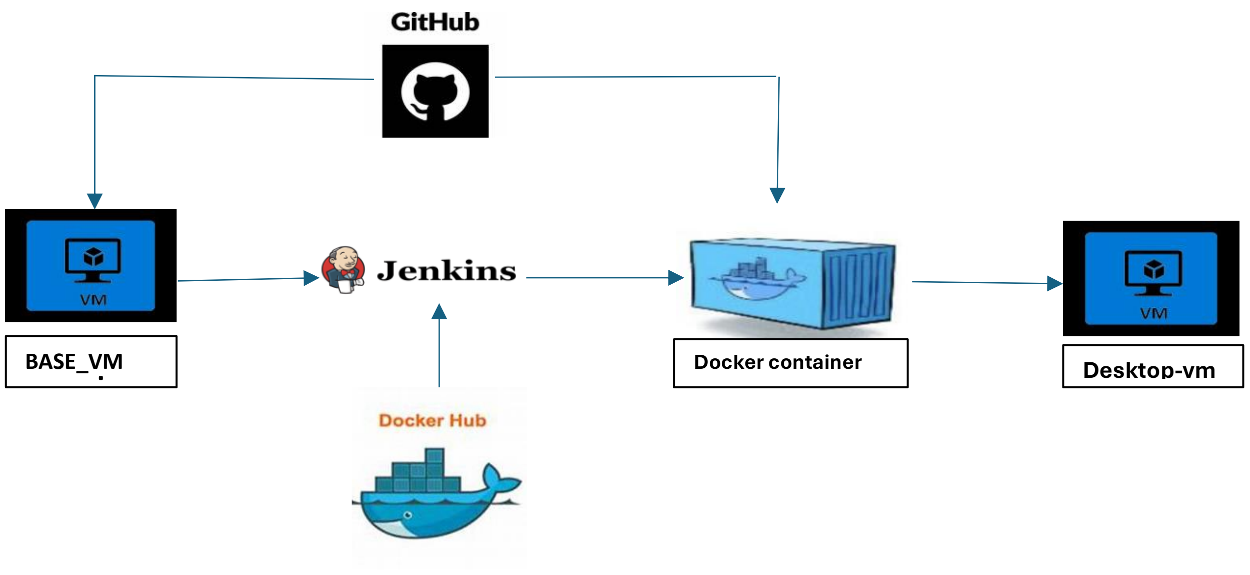
Automated Desktop Environment Deployment using Terraform, Ansible, Jenkins, and Docker

Project Description:



This project aims to automate the provisioning and configuration of infrastructure and software for setting up a remote desktop environment using Terraform, Ansible, Jenkins, and Docker. The process involves the following key steps:

- **Provision Jenkins Server:** Using Terraform, we will create the necessary infrastructure on Google Cloud Platform (GCP), including a Virtual Private Cloud (VPC), a Service Account, and a Virtual Machine (VM) instance. This VM will be configured as a Jenkins server, an open-source automation server designed to manage continuous integration and deployment pipelines.

- **Provision Ubuntu Desktop Server:** Using another Terraform script with a Remote backend block, we will provision a VM instance that connects to the previously created VPC infrastructure. This VM will run Ubuntu 20.04 (focal v20240519) and will serve as a remote desktop environment.
- **Configure Ubuntu Desktop Server:** An Ansible playbook is written to install necessary software on the Ubuntu Desktop server, including Python and Chrome Remote Desktop, and set up the desktop environment.
- **Execute Configuration via Jenkins:** The Jenkins server is configured to run jobs using Docker agent nodes. These jobs execute the Terraform scripts to provision the infrastructure and the Ansible playbook to configure the Ubuntu Desktop server.
- **Access Remote Desktop:** Finally, the desktop environment on the Ubuntu Desktop VM instance can be accessed remotely using Chrome Remote Desktop through a web browser.

Note: Refer the below GitHub Repository for entire Project's terraform Script, Playbook, Dockerfile, Jenkinsfile (pipeline) etc.,

https://github.com/bpurnachander/get-ubuntudesktop-iac.git

Pre-requisites:

1. Create Docker image from Dockerfile in above GitHub repo and push it to your Docker-Hub (or) Artifact Registry.
2. Create a Storage Bucket for Terraform Remote backend Setup.

Step1 – Create a Base VM – Instance with Ubuntu OS

Step2 – Get into the VM instance SSH – Terminal Install Terraform and Clone the below given GitHub – Repo to local

Run the below commands:

- git init
- git clone https://github.com/bpurnachander/get-ubuntudesktop-iac

Step3 – Now get into the cloned GitHub directory and Perform terraform lifecycle and create the Jenkins-Server infra - Run the below commands to perform:

- cd /get-ubuntudesktop-iac/ terraform-jenkins/
- terraform init
- terraform validate
- terraform plan
- terraform apply --auto-approve

  Note: make sure you have modified your GCP Project_Id

Step4 – Now Login to the Jenkins console and create a pipeline Job (pipeline-1)

Step5 – install Docker Plugin:

- Manage Jenkins -> plugins -> Available plugins – docker

Step6 – Go to credentials add your docker hub username and password

- Manage Jenkins -> credentials -> global -> add credentials -> kind (username & password) -> username & password (docker-hub username & password) -> Id (docker-hub) -> save

Step7 – Add credentials for user created by Docker image in the container

- Manage Jenkins --> credentials -> global -> add credentials -> kind (username & password) -> username (jenkins) -> password (password) -> Id (jenkins-user) -> save

Step8 – Add docker agent configuration to make container as a slave node

- Goto Manage Jenkins -> Cloud -> New cloud (give name) -> Select (docker) Create

Drop down docker cloud details:

- Docker host url = tcp:$JENKINS SERVER EXTERNAL IP:4243
- Click Test connection (you need to get docker version) -> select (Enabled)

Drop down Docker Agent templates - (List of Images to be launched as agents)

- Labels (docker) -> select (Enabled)
- Name (docker-agent)
- Docker Image (give your DOCKER_HUB REPO PATH)
- Select (Registry Authentication) -> Credentials (select DOCKER-HUB credential ID)
- Remote File System Root (/var/lib/jenkins)
- Usage (only build jobs label expression matching this node)
- Connect method (Use Configured SSH credentials)
- SSH credentials (Select Jenkins-user Credential ID)
- Host Key Verification Strategy (Non-Verifying verification strategy) ->save

Step9 – Now configure the pipeline as given in below steps:

a. Parameterize the Job: -> This project is parameterized (select) -> (select) choice Parameters

   -> Name = action

   -> Choices: apply
   Destroy

b. Pipeline Definition: Pipeline Script from SCM (select)

-> Repository URL (https://github.com/bpurnachander/get-ubuntudesktop-iac.git)

-> Branch Specifier (*/main) ->Script Path = (Jenkinsfile) -> save -> Build the Job

Step10 – Chrome Remote Desktop Setup:

a. Click the below link for Chrome Remote desktop site

   Remote Access - Chrome Remote Desktop (google.com)

b. Select (Setup Viva SSH) -> Begin -> Next -> Authorize -> copy the Debian Linux link

c. Now Login to Desktop-Server SSH Terminal as a Normal user.

   Now paste above copied Link here and Enter

   -> Set Hostname = (desktop)

   -> Enter a PIN of at least six digits:

   -> Enter the same PIN again:

d. Now get back into your Browser (chrome remote desktop)

   -> Select Remote Access -> Remote devices (you will find your node – desktop)

   -> Click on it and login with the 6-digit pin.

Troubleshooting Errors:

1. Connection refused:

Cause: Openssh-client not working properly.

Remedy: Wait for few seconds after VM is created.

2. Permission denied:

Cause: User has no permission to use private key.

Remedy: Check the user type and change permissions of private key.

3. Host key verification failed:

Cause: Unable to verify Host key.

Remedy: echo 'host_key_checking = False' | sudo tee -a /etc/ansible/ansible.cfg

4. Error: Jenkins doesn't have label 'docker-agent'

Remedy: Check Slave User credentials

Terraform Errors Troubleshooting

5. Error 409: Resource Already Exists
6. Error 400: Syntax Error

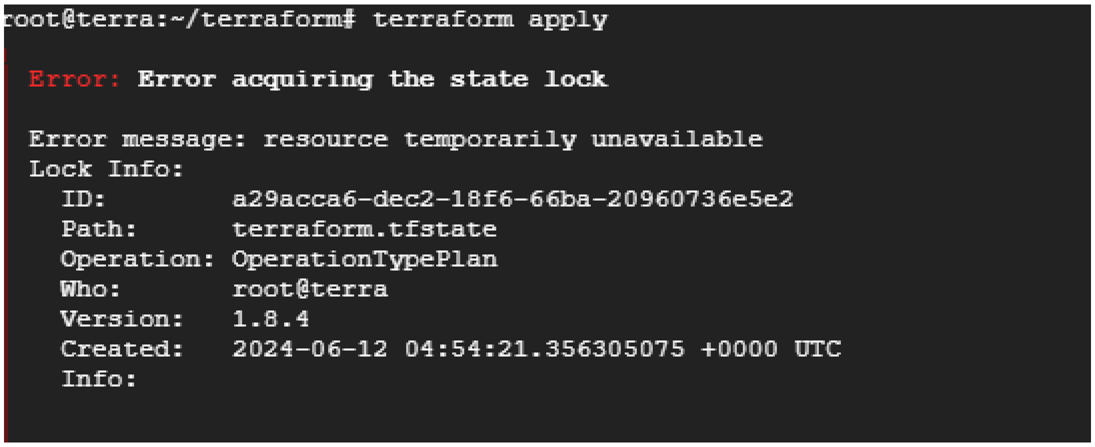
Remedy: Check the Syntax in Terraform Script

7. Erro 404: API error

Remedy: Enable API for the Resources to be Created

8. Error: Terraform State Lock

Remedy-1: terraform force-unlock Command - you can forcefully unlock the state using the terraform force-unlock command with the lock ID provided



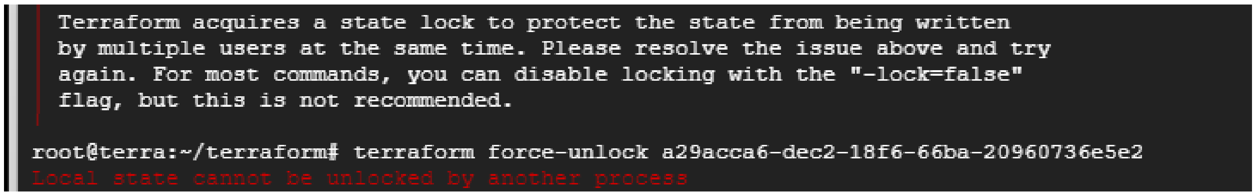
```
root@terra:~/terraform# terraform apply

 Error: Error acquiring the state lock

 Error message: resource temporarily unavailable
 Lock Info:
   ID:        a29acca6-dec2-18f6-66ba-20960736e5e2
   Path:      terraform.tfstate
   Operation: OperationTypePlan
   Who:       root@terra
   Version:   1.8.4
   Created:   2024-06-12 04:54:21.356305075 +0000 UTC
   Info:
```

- terraform force-unlock a29acca6-dec2-18f6-66ba-20960736e5e2

Error: Local State File Cannot be Unlocked by another Process



```
  Terraform acquires a state lock to protect the state from being written
  by multiple users at the same time. Please resolve the issue above and try
  again. For most commands, you can disable locking with the "-lock=false"
  flag, but this is not recommended.

root@terra:~/terraform# terraform force-unlock a29acca6-dec2-18f6-66ba-20960736e5e2
Local state cannot be unlocked by another process
```

Remedy-2: Remove the Lock File Manually

Since you are dealing with a local state, you can manually delete the lock file to remove the lock. Follow these steps:

- cd /Path/to/terraform/
- rm .terraform.tfstate.lock.info

Remedy-3: Terminate the Running Terraform Processes

- ps aux | grep terraform
- kill -9 <PID>
- For eg: kill -9 1425

Now Retry the Terraform Command: terraform apply (state lock will be unlocked)