# C:\Users\jameswhi\Desktop\SDM Templates\3393d980-0436-4e28-b1fd-32c927c2f3c0\MSFT_logo_rgb_C-Wht_D.png

**Cloud DevOps Engineer**

**Project 1**

**(Using Conventional DevOps Tools)**

**Prepared By: Victor Majumder**

**Contents**

[1](#_Toc84201057)

[Business Requirement 4](#_Toc84201058)

[Architecture 4](#_Toc84201059)

[Solution 4](#_Toc84201060)

[Part 1 4](#_Toc84201061)

[Generate Service Principle for Terraform in Azure 4](#_Toc84201062)

[Installation Terraform in linux 5](#_Toc84201063)

[Install Azure cli 5](#_Toc84201064)

[Create required Vars.tf to create two VMs in Azure 5](#_Toc84201065)

[Create required terraform.tfvars 5](#_Toc84201066)

[Create main.tf to create two Linux VMs 6](#_Toc84201067)

[Init,Plan and Apply Terraform Script 6](#_Toc84201068)

[Manually Start Jenkins and configure required Plug-ins and Master Slave Configuration 6](#_Toc84201069)

[Part 2 – Phase 1 6](#_Toc84201070)

[Create Maven Project with Archetype as web application in eclipse 6](#_Toc84201071)

[Modify Index.jsp under src/main/webapp to display a custom message 6](#_Toc84201072)

[Run Maven clean install in eclipse to check the build and check for .war file in target folder 6](#_Toc84201073)

[Part 2 – Phase 2 7](#_Toc84201074)

[Generate Dockerfile under project folder of your app 7](#_Toc84201075)

[Modify FORM statement to use tomcat as base image 7](#_Toc84201076)

[Test the Dockerfile by running Docker build and create a container 7](#_Toc84201077)

[Access the application from container and check it 7](#_Toc84201078)

[Part 2 – Phase 3 7](#_Toc84201079)

[Create a github repository and copy repo URL 7](#_Toc84201080)

[In Eclipse convert the app in to a local repo from Team menu share Project Option 7](#_Toc84201081)

[Commit and Push the code to remote repo 7](#_Toc84201082)

[Part 2 – Phase 4 7](#_Toc84201083)

[Modify Project index.jsp, Commit and Push to remote repo 7](#_Toc84201084)

[Check for the change in remote Repo 8](#_Toc84201085)

[Part 2 – Phase 5 8](#_Toc84201086)

[In build server configure Ansible manually 8](#_Toc84201087)

[Modify ansible.cfg to use hosts file as inventory 8](#_Toc84201088)

[install python-pip in ansible server 8](#_Toc84201089)

[Using PIP install azure modules in Ansible server 8](#_Toc84201090)

[Use the same service Principle created for terraform for ansible to get authenticated to Azure 9](#_Toc84201091)

[Create a playbook1 to create a vm in azure 9](#_Toc84201092)

[Update the playbook2 to install Docker engine on the VM 9](#_Toc84201093)

[create a shell script to get VM ip and updating it in Inventory File 9](#_Toc84201094)

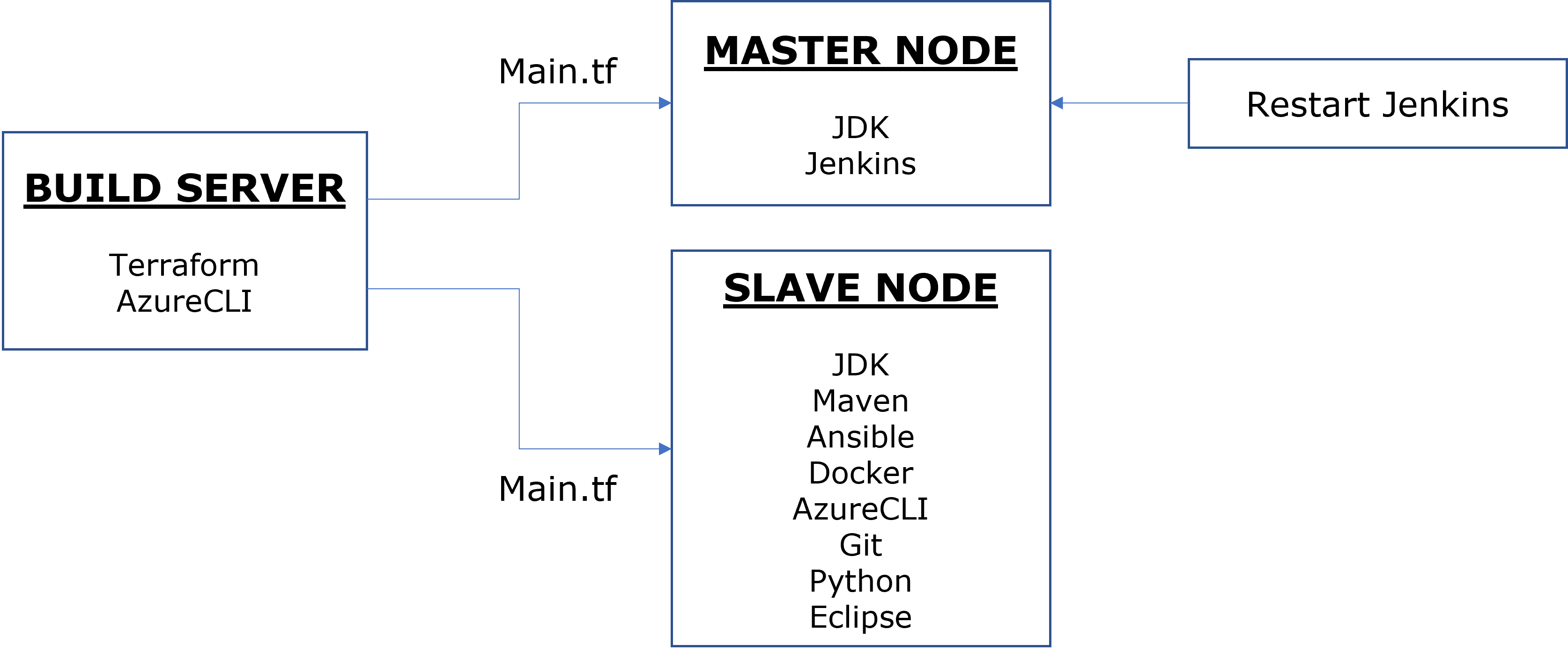
[Part 2 – Phase 7 9](#_Toc84201095)

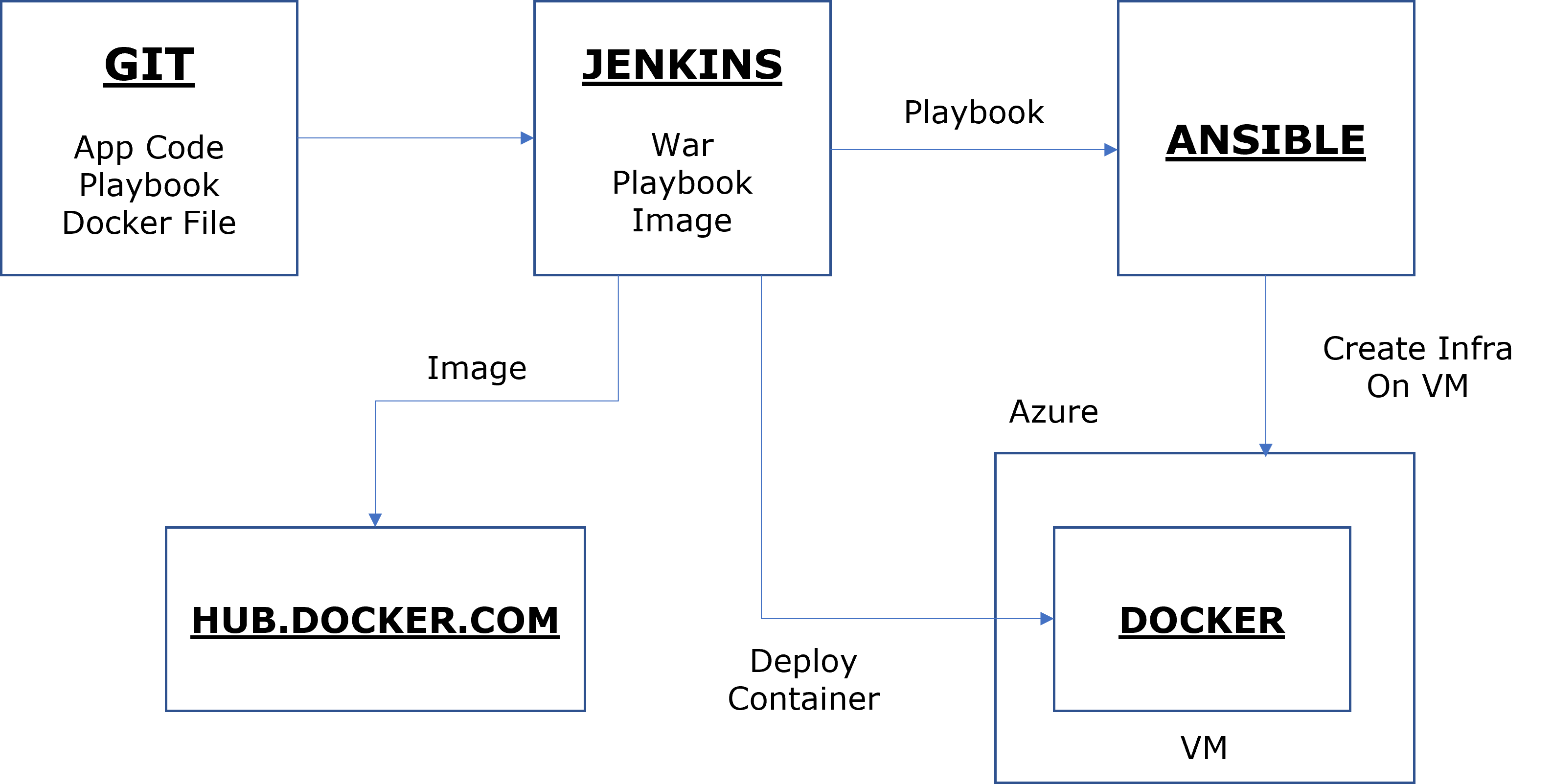
[From Build server run the playbook1,playbook2 and shell script to test for the required result 9](#_Toc84201096)

# Business Requirement

At Wolkentech Pvt Ltd, there was a separate team that provided dedicated Jenkins pipelines with a stable master-slave node setup, but the environment was only used for quality assurance (QA), staging, and production environments. The development environment was still very manual, and the team needed to automate it to gain as much flexibility as possible while accelerating the development effort. This is the reason they decided to build a CI/CD pipeline for DevOps. And the open source version of Jenkins was the obvious choice due to its flexibility, openness, powerful plugin-capabilities, and ease of use.

# Architecture





# Solution

* Build a multi-staged Java build pipeline that takes from the phases of pulling dependencies from JAR repositories like Maven, compiling Java codes, running the unit tests, packaging into a JAR/WAR file, and deploying to a cloud server.
* Construct a multi-pipeline automating the tasks of executing Ansible playbooks to deploy the required infrastructure for Application.
* Design a complete end-to-end DevOps pipeline that pulls the infrastructure resource files and configuration files stored in SCM like GitHub and executing the scripts through various runtime programs

# Part 1

## Generate Service Principle for Terraform in Azure

Azure Portal 🡪 Azure Active Directory 🡪 app registrations 🡪 new registration

Name: **Terraform**

Supported Account Types: **Accounts in this organizational directory only (single-tenant)**

Redirect URI: **you should choose "Web" for the URI type. the actual value can be left blank**

Register

Take note of the "Application (client) ID" and the "Directory (tenant) ID"

Azure Portal 🡪 Azure Active Directory 🡪 app registrations 🡪 select the new app created 🡪 Certificates & secrets 🡪 Click on New client secret button 🡪

Description: **Terraform\_Secret**

Expires: **Select the default one**

Add

Take a note of the secret ID

Subscription 🡪 Access control 🡪 Add role assignment 🡪 Specify a role 🡪 search for and select the name of the Service Principal created in Azure Active Directory to assign it this role - then press Save.

Add in env variables

ARM\_CLIENT\_ID="**XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX**"

ARM\_CLIENT\_SECRET=" **XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX**"

ARM\_SUBSCRIPTION\_ID=" **XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX**"

ARM\_TENANT\_ID=" **XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX**"

### Installation Terraform in linux

sudo wget <https://releases.hashicorp.com/terraform/1.0.3/terraform_1.0.3_linux_amd64.zip>

sudo unzip terraform\_1.0.3\_linux\_amd64.zip

sudo mv terraform /usr/bin/

terraform version

### Install Azure cli

curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash

## Create required Vars.tf to create two VMs in Azure

mkdir devops\_project\_1

vi vars.tf



## Create required terraform.tfvars

vi terraform.tfvars



## Create main.tf to create two Linux VMs

vi main.tf



## Init,Plan and Apply Terraform Script

terraform init

terraform validate

terraform plan

terraform apply

## Manually Start Jenkins and configure required Plug-ins and Master Slave Configuration

Login to masternode

Start jenkins

java -jar jenkins.war &

http://13.91.122.139:8080/ (http://<publicip\_of\_masternode>:8080/)

accept pluglins

create account

# Part 2 – Phase 1

## Create Maven Project with Archetype as web application in eclipse

mvn archetype:generate -DgroupId=Devops\_project\_1 -DartifactId=Devops\_project\_1 -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false

below project will be created

/home/adminuser/Devops\_project\_1

## Modify Index.jsp under src/main/webapp to display a custom message

cd src/main/webapp/

vi index.jsp

<html>

<body>

<h2>Hello This is a custom project for Devops Project 1!</h2>

</body>

</html>

## Run Maven clean install in eclipse to check the build and check for .war file in target folder

cd /home/adminuser/Devops\_project\_1

mvn clean install

cd /home/adminuser/Devops\_project\_1/target

you should see a .war file like “Devops\_project\_1.war”

# Part 2 – Phase 2

## Generate Dockerfile under project folder of your app

cd /home/adminuser/Devops\_project\_1

## Modify FORM statement to use tomcat as base image

vi Dockerfile

FROM tomcat:8.0-alpine

ADD target/Devops\_project\_1.war /usr/local/tomcat/webapps/

EXPOSE 8080

CMD [“catalina.sh”, “run”]

## Test the Dockerfile by running Docker build and create a container

sudo docker build -t devops\_project\_1\_app .

## Access the application from container and check it

sudo docker run -p 8080:8080 devops\_project\_1\_app

Run below to check your web app

<http://13.91.122.121:8080/Devops_project_1/> (http://<slave\_public\_ip>:8080/Devops\_project\_1/)

# Part 2 – Phase 3

## Create a github repository and copy repo URL

<https://github.com>

sign up or sign in

username: XXXXXXXX

email: [XXXXXXXX@gmail.com](mailto:XXXXXXXX@gmail.com)

create a new repository

repo URL: <https://github.com/XXXXXXXXXXX/devops_project_1.git>

## In Eclipse convert the app in to a local repo from Team menu share Project Option

git init

git status

git add .

git commit -m my-first-commit

## Commit and Push the code to remote repo

git remote add origin <https://github.com/XXXXXXXXXX/devops_project_1.git>

git push -u origin master

it will ask for username. Enter XXXXXXXXXX

it will ask for password. Create Personal Access Token on GitHub

From your GitHub account, go to Settings => Developer Settings => Personal Access Token => Generate New Token (Give your password) => Fillup the form => click Generate token => Copy the generated Token, it will be something like ghp\_sFhFsSHhTzMDreGRLjmks4Tzuzgthdvfsrta

# Part 2 – Phase 4

## Modify Project index.jsp, Commit and Push to remote repo

cd /home/adminuser/Devops\_project\_1/src/main/webapp

vi index.jsp

<html>

<body>

<h2>Hello This is a custom project for Devops Project 1 -- update for git test!</h2>

</body>

</html>

git add .

git commit -m my-third-commit

git push -u origin master

## Check for the change in remote Repo

You can see the index.jsp updated here

<https://github.com/XXXXXXXXXXXXX/devops_project_1/blob/master/src/main/webapp/index.jsp>

# Part 2 – Phase 5

## In build server configure Ansible manually

Login to build server(initial server from where terraform was executed)

Install python on server

sudo apt-get install -y python-pip

install ansible on server

$ sudo apt-get update

$ sudo apt-get install software-properties-common

$ sudo apt-add-repository ppa:ansible/ansible

$ sudo apt-get update

$ sudo apt-get install ansible

ansible –version

it gets installed in /etc/ansible

change ownership

sudo chown victor\_123: victor\_123/etc/ansible/ -R

## Modify ansible.cfg to use hosts file as inventory

sudo vi /etc/ansible/ansible.cfg

Enable the inventory

sudo vi /etc/ansible/hosts

[slavegroup]

slave ansible\_host=13.91.122.121

[slavegroup:vars]

ansible\_connection=ssh

ansible\_user=adminuser

generate SSH keys

ssh-keygen -m PEM -t rsa -b 4096

copy id\_rsa.pub and paste under authorized\_keys in slave node(also calling it as ansible node)

## install python-pip in ansible server

python-pip was already installed by terraform in part 1

## Using PIP install azure modules in Ansible server

Login to slave node

command to install azure modules

sudo pip install ansible[azure]

## Use the same service Principle created for terraform for ansible to get authenticated to Azure

mkdir .azure

ls -al

cd .azure/

touch credentials

vi credentials

[default]

subscription\_id= **XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX**

client\_id= **XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX**

secret= **XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX**

tenant= **XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX**

## Create a playbook1 to create a vm in azure



## Update the playbook2 to install Docker engine on the VM

Add below in /etc/ansible/hosts

[dockergroup]

docker ansible\_host=40.86.174.112

[dockergroup:vars]

ansible\_connection=ssh

ansible\_user=victor\_123



## create a shell script to get VM ip and updating it in Inventory File

# Part 2 – Phase 7

## From Build server run the playbook1,playbook2 and shell script to test for the required result

ansible-playbook playbook1.yaml

ansible-playbook playbook2.yaml