Chanesh Mahadeo
Kura Labs
10/29/2022

Deployment 4 Documentation

Welcome to Kura Labs' deployment 4 documentation. In this deployment we are utilizing terraform in order to provision a vpc as well as deploying a web server to one of the subnets through the use of an ec2 instance.

Intro

To start we must create our Jenkins Server. For this deployment we must also installed terraform on our jenkins machine. To streamline this process I have appended terraform installation to a userdata script which also installs jenkins.

Userdata:

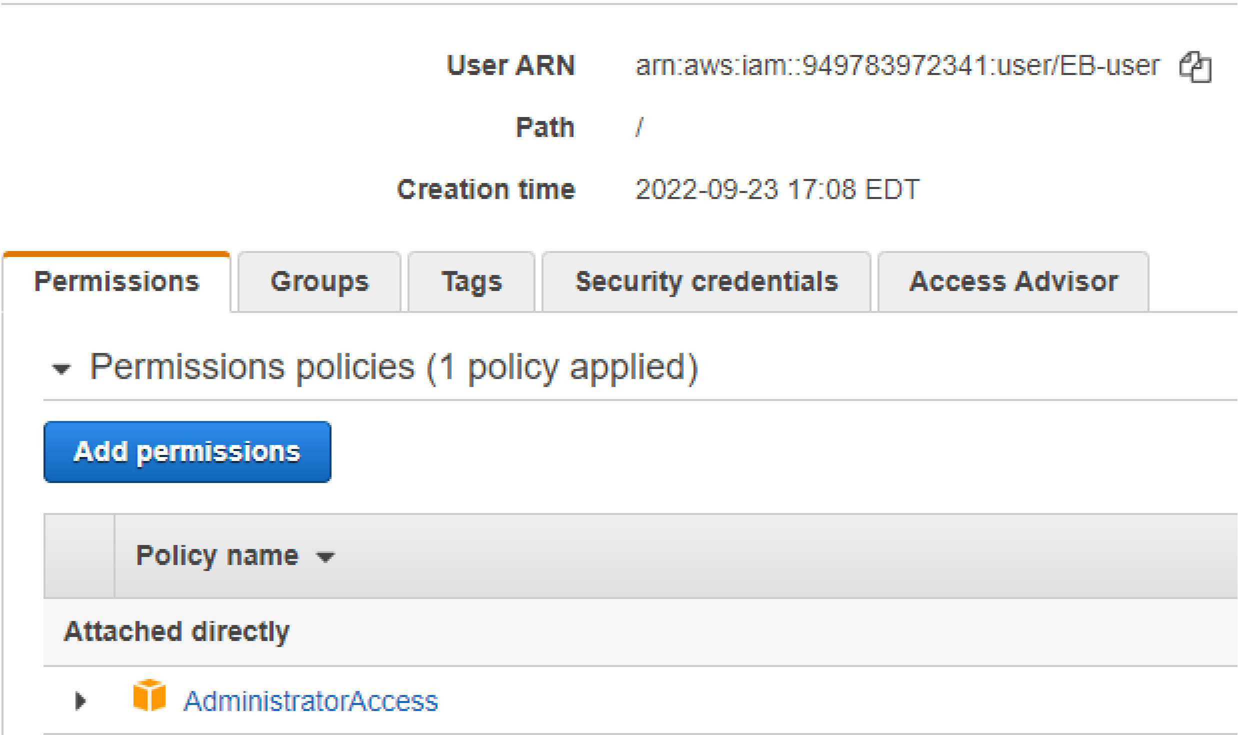
```
#!/bin/bash



if [ $UID != 0 ]; then

   echo "Run again with admin permissions"

   exit 1

fi

wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | gpg --batch
--yes --dearmor -o /usr/share/keyrings/jenkins.gpg && echo "Jenkins Keyring
Added"
sh -c 'echo deb [signed-by=/usr/share/keyrings/jenkins.gpg]
http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list' && echo "Jenkins Repo Added"
apt-get update

#Install java, Jenkins, pip and venv in that order

apt-get install default-jre -y && echo "Installed Java Runtime Engine" && apt-get
install jenkins -y && echo "Installed Jenkins" && apt-get install python3-pip -y
&& echo "Installed Python pip" && apt-get install python3.10-venv -y && echo
"Installed Python venv"
sudo apt-get update && sudo apt-get install -y gnupg software-properties-common

wget -O- https://apt.releases.hashicorp.com/gpg | \

   gpg --dearmor | \
   sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg
gpg --no-default-keyring \
   --keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \
   --fingerprint
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \
   https://apt.releases.hashicorp.com $(lsb_release -cs) main" | \
   sudo tee /etc/apt/sources.list.d/hashicorp.list
sudo apt update
sudo apt-get install terraform
#Start the Jenkins service
systemctl start jenkins && echo "Jenkins Started"
#successful
echo "Installation successful"
exit 0
```

Once the jenkins machine starts, use terraform –version in order to verify installation.

AWS IAM Configuration

We will be using an AWS IAM user in order to deploy our infrastructure to the cloud. To do this we must first create an IAM user that has the AdministratorAccess policy.

Example:



User ARN arn:aws:iam::949783972341:user/EB-user

Path /

Creation time 2022-09-23 17:08 EDT

Permissions policies (1 policy applied)

Policy name
Attached directly
AdministratorAccess

As you can see from this screenshot, you can use the same EB-user we created in an older deployment. You must have both keys for this user.

Configuring Jenkins

- Download the Jenkins Addon Pipeline Keep Running Step
- In the Jenkins Dashboard, click on manage Jenkins and then select Manage Credentials
- Now select Global
- On the right, Select Add Credentials
- Now enter the First credentials:
- ○ Select “Secret text” for Kind
- ○ Scope should be Global
- ○ Secret: Copy and Paste your aws access key
- ○ ID: AWS_ACCESS_KEY
- ○ Select Create
- Now enter the Second credentials:
- ○ Select “Secret text” for Kind
- ○ Scope should be Global
- ○ Secret: Copy and Paste your aws secret key
- ○ ID: AWS_SECRET_KEY
- ○ Select Create
- Before you build your pipeline and execute, observe the Jenkinsfile, initTerraform folder and all files in the initTerraform folder.
- Once you have successfully run your deployment and check your application. Add a destroy stage to the Jenkinsfile

- The updated jenkinsfile can be found at https://github.com/Chaneshm/kuralabs_deployment_4/blob/main/Jenkinsfile
- You may now build your pipeline

Changes

Some changes I made were to the file structure of the terraform files. Instead of having just three different files, I took inspiration from professional terraform file structures so that the code can be deciphered much easier than when there is one massive file. All other changes done in my previous deployments were also added here.

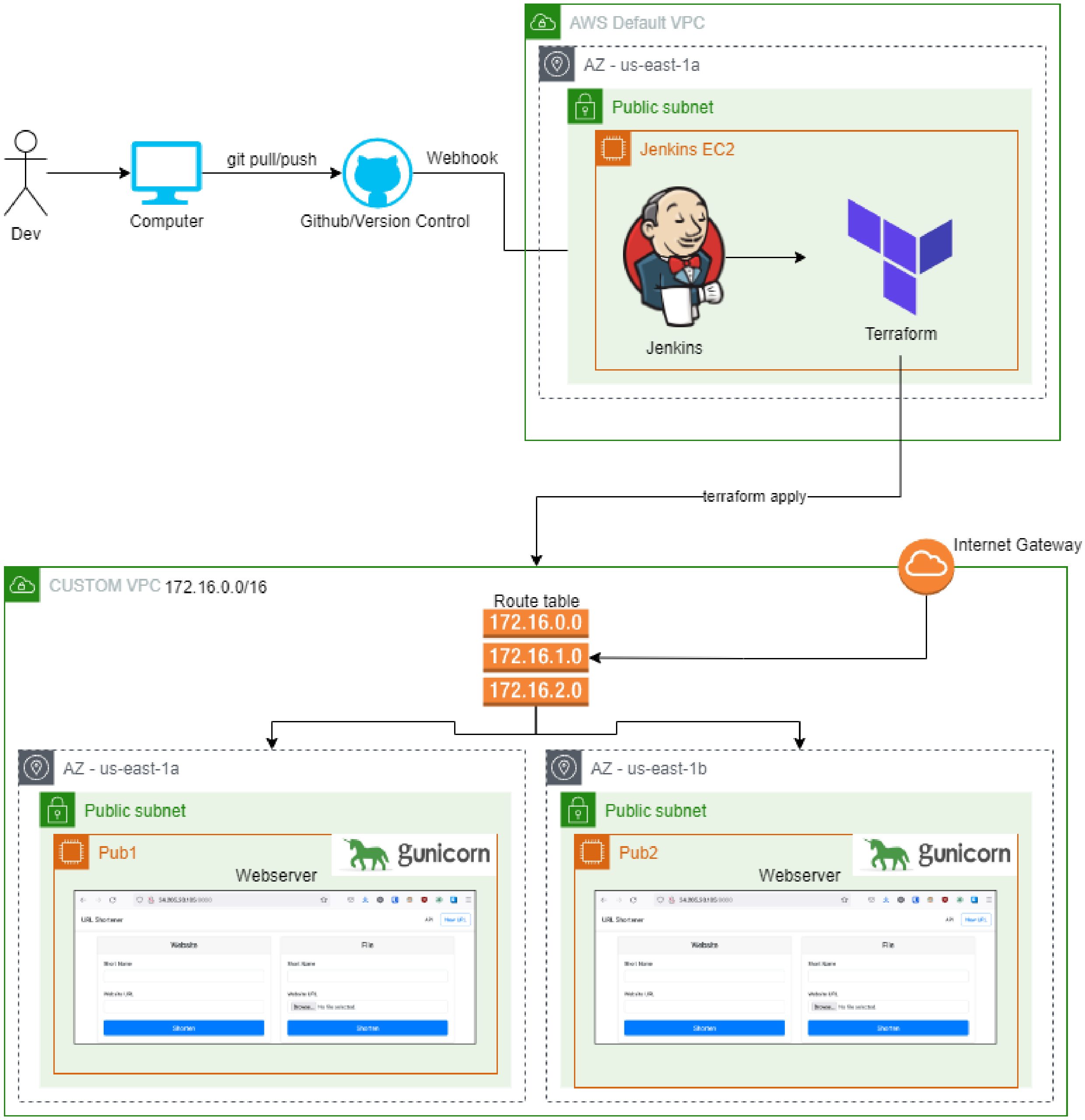
Issues

Some issues I had with this deployment initially were due to my own changes. I did not know how to properly pass variables through my new terraform file structure and so I did not have successful builds due to this. Also there are Jenkins addons which I had to install manually which are required so that the pipeline does not throw an error.

Improvements

Some improvements that I would add onto this pipeline is somehow streamlining the ENTIRE Jenkins configuration. That way, the user would only have to worry about perhaps inputting their credentials manually and I could maybe have a script configure all of Jenkins automatically. I assume there will be a tool to do this in the future so that processes like this are only that much more streamlined.

Pipeline Diagram


AWS Default VPC
AZ - us-east-1a
Public subnet
Jenkins EC2
Dev
Computer
git pull/push
Github/Version Control
Webhook
Jenkins
Terraform
terraform apply
Internet Gateway
CUSTOM VPC 172.16.0.0/16
Route table
172.16.0.0
172.16.1.0
172.16.2.0
AZ - us-east-1a
Public subnet
Pub1
gunicorn
Webserver
URL Shortener
API
New URL
Website
File
Short Name
Website URL
Browse...
No file selected.
Shorten
Shorten
AZ - us-east-1b
Public subnet
Pub2
gunicorn
Webserver
URL Shortener
API
New URL
Website
File
Short Name
Website URL
Browse...
No file selected.
Shorten
Shorten