

Project: Automate infrastructure using Terraform

Requirements: The UnD Fitness has decided to use the AWS (IaaS) infrastructure as code and (SaaS) software as service. This requires infrastructure to be run on the cloud (AWS) and to achieve this company is going to follow DevOps mythology.

As a DevOps engineer, you need to launch the EC2 instance and install Java, Jenkins, and Python on it using Terraform.

Tools required: Terraform, AWS account with security credentials, and Keypair.

Expected Deliverables:

- Launch an EC2 instance
- Connect to instance
- Install Jenkins, Java, and Python on the instance.

Step 1: Install the Terraform on the local\AWS machine.

Step 2: Create an IMA user on the AWS account.

Step 3: Write a terraform script.

Step 1 : Terraform Installation on Linux operating system.

1. Execute the following commands to install the terraform.

a. `$ yum install wget`

b. `$ wget`

https://releases.hashicorp.com/terraform/1.2.3/terraform_1.2.3_linux_arm64.zip

c. #Install unzip package.

d. `$ sudo apt-get install -y unzip`

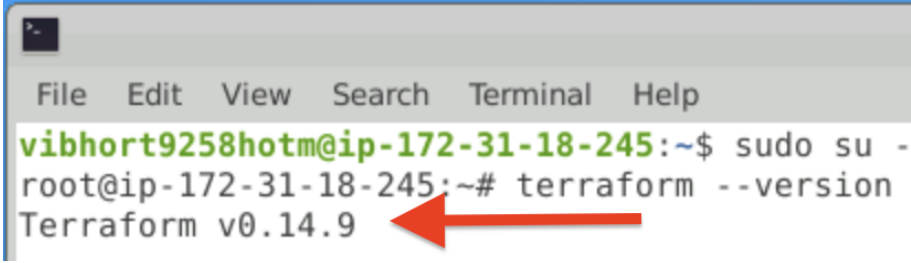
e. #Unzip the terraform_0.14.9_linux_amd64.zip file

f. `unzip terraform_1.2.3_linux_arm64.zip`

g. # Move the file terraform files to the given below location

h. `$ sudo mv terraform /usr/local/bin/`

i. `$ terraform -version`

j. 

```
vibhort9258hotm@ip-172-31-18-245:~$ sudo su -
root@ip-172-31-18-245:~# terraform --version
Terraform v0.14.9
```

Step 2: Create an IMA user and keypair.

1. Sign in as a root user on your AWS account.
2. Search for the IAM service and navigate to the Users screen.
3. Click on Add User to navigate to a user detail form. Provide all details, such as the username and access type and give the permission to the user and click on the add user.

4. The user has added and download the .csv file of Access key ID and Secret access key (keep it in a safe place or remember both Access key ID and Secret access key.)
5. Log in to the newly created account and navigate to the EC2 instance and select keypair from network & security.
6. Provide the name and select the RSA and .pem and create key pair and save the .pem file.

Step :3y

Write a script

1. Create a new directory and switch to the directory.
2. Create a new file with .tf extension.
3. Write a script

```
provider "aws" {
  region = "us-east-1"
  access_key = "AKIAT7DAGQZD3DXXXX"
  secret_key = "Yt7nXBDRu69jmjBcvRaaXrwfWMSI5rsFq2vOXXXXX"
}

resource "aws_security_group" "sg" {
  name = "allow_tls"
  description = "Allow TLS inbound traffic"

  ingress {
    description = "SSH"
    from_port = 22
    to_port = 22
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    description = "HTTP"
    from_port = 8080
    to_port = 8080
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port = 22
    to_port = 22
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

```

tags = {
  Name = "allow_tls"
}
}

resource "aws_instance" "instance"{
  ami = "ami-0cff7528ff583bf9a"
  instance_type = "t2.micro"
  key_name = "teraform"
  user_data = <<-EOF
#!/bin/bash
# Updating the newly install system(EC2)

  sudo yum update
# Wget Installation
  sudo yum install wget git -y
# java installation
  sudo yum install java-1.8.0-openjdk-devel -y
# Jenkins installation
  sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat/jenkins.repo
  sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key
  sudo yum upgrade -y
  sudo yum install jenkins -y
  sudo systemctl start jenkins
#python installation
  sudo yum install python37 -y

EOF
}

resource "aws_network_interface_sg_attachment" "sg_attachment" {
  security_group_id = aws_security_group.sg.id
  network_interface_id = aws_instance.instance.primary_network_interface_id
}

```

4. Run the \$ terraform init command

```
root@ip-172-31-18-245:~/project_teraform# terraform init
```

Initializing the backend...

Initializing provider plugins...

- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.20.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.

```
root@ip-172-31-18-245:~/project_teraform#
```

a.

5. Check the plan with \$ terraform plan command
6. Finally, run the \$ terraform apply command to create an instance.

a.

```
aws_security_group.sg: Creating...
aws_instance.instance: Creating...
aws_security_group.sg: Creation complete after 2s [id=sg-0ff599c886f0c36e4]
aws_instance.instance: Still creating... [10s elapsed]
aws_instance.instance: Still creating... [20s elapsed]
aws_instance.instance: Still creating... [30s elapsed]
aws_instance.instance: Still creating... [40s elapsed]
aws_instance.instance: Creation complete after 41s [id=i-0657bbd3367645944]
aws_network_interface_sg_attachment.sg_attachment: Creating...
aws_network_interface_sg_attachment.sg_attachment: Creation complete after 1s [id=sg-0ff599c886
```

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

Step 4 Verify the instance and check the application.

1. Validation of instance

The screenshot shows the AWS Management Console for an EC2 instance named 'i-0657bbd3367645944'. The instance is in a 'Running' state, using a 't2.micro' instance type, and is located in the 'us-east-1b' availability zone. The instance is associated with the security group 'sg-0ff599c886f0c36e4'. The 'Inbound rules' section is expanded, showing a table of security group rules.

Security group rule ID	Port range	Protocol	Source	Security groups	default	allow
sgr-03e52ef6e262b4681	All	All	sg-0facd978bc4ccffb9	default	✓	-
sgr-048be9af3671f5cb5	8080	TCP	0.0.0.0/0	allow_tls	-	✓
sgr-0c20493ebee3d4fe1	22	TCP	0.0.0.0/0	allow_tls	-	✓

a.

2. SSH to the instance.

```

  _ |  _ |  )
  _ | (  _ | /
  _ | \  _ | _ |
                                     Amazon Linux 2 AMI

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-94-103 ~]$
```

a.

3. Install and verify the Java version

```

[ec2-user@ip-172-31-94-103 ~]$ java -version
openjdk version "1.8.0_312"
OpenJDK Runtime Environment (build 1.8.0_312-b07)
OpenJDK 64-Bit Server VM (build 25.312-b07, mixed mode)
```

a.

4. Install and Verify the Jenkins.

```

[root@ip-172-31-94-103 ~]# sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2022-07-04 09:36:33 UTC; 26s ago
     Main PID: 7329 (java)
    CGroup: /system.slice/jenkins.service
            └─7329 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=%C/jenkins/war --httpPort=8080
```

a.

⚠ Not secure | 54.89.137.43:8080/login?from=%2F

Getting Started

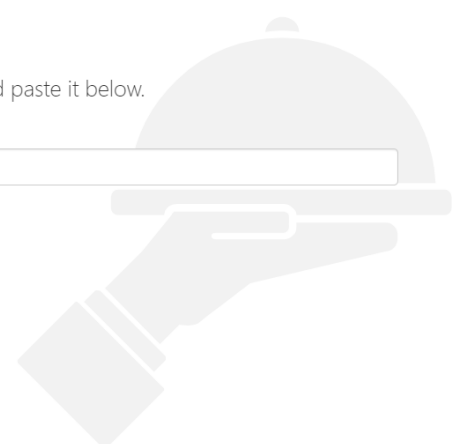
Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password



[Continue](#)

b.

5. Install and verify the Python version

```
[root@ip-172-31-94-103 ~]# python3 --version  
Python 3.7.10
```

- a.