Project:- Terraform IaC for VM

Description

Use Terraform to provision infrastructure

Description:

Nowadays, infrastructure automation is critical. We tend to put the most emphasis on software development processes, but infrastructure deployment strategy is just as important. Infrastructure automation not only aids disaster recovery, but it also facilitates testing and development.

Your organization is adopting the DevOps methodology and in order to automate provisioning of infrastructure there's a need to set up a centralized server for Jenkins.

Terraform is a tool that allows you to provision various infrastructure components. Ansible is a platform for managing configurations and deploying applications. It means you'll use Terraform to build a virtual machine, for example, and then use Ansible to install the necessary applications on that machine.

Considering the Organizational requirement you are asked to automate the infrastructure using Terraform first and install other required automation tools in it.

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

Expected Deliverables:

- Launch an EC2 instance using Terraform
- Connect to the instance
- Install Git, Ansible, Jenkins, Java and Python in the instance

SOLUTION:

1. Create a key pair in AWS which will be used later. Download them and save it in your system.

2. Create an IAM User and generate public access and secret access key and save in your machine in notepad.

3. Create a directory for the terraform project. Create a terraform file for saving AWS credentials with extension .tf. Save AWS region, secret access key and public access key in this file.

4. Install AWS CLI in your machine. After that, With the AWS Configure command, Add the public and private access key, with the region and other details, as per need. These details will be considered as default details which terraform will use if some detail is not mentioned in the .tf file. These details will be saved in the AWS home directory in the AWS credentials.

5. After adding the details in the AWS configure file, you can remove key detail from the .tf file in the terraform directory, and give the AWS home root path instead, where your default keys and other credentials are saved. Generally the root path is —> [“~/.aws/credentials”] —> here you can find the AWS Credentials you have given.
   Please note-: Details provided in the .tf files will be considered first by the terraform. If some detail is not present there, then terraform will pick detail from AWS home directory .

6. Write the terraform file specifying the, AWS resource block in the terraform file, giving the instance type, name and other details.

7. Write the Security Group block in the terraform file, containing the port number, CIDR IP address details, ingress egress network, and other network security details, which needs to be specified.

8. Write the data you want to call within the instance. Here, we are installing git, ansible in our created instance. But prior to that, we will need to connect to our instance, via EDI key pair which was generated in the first step, so that we can access our instance and make changes on it. Mention the key name in the AWS resource block in the last tag.

9. Connect the security group with the AWS resource block.

10. Plan and apply the terraform. The instance will be successfully generated with the git and ansible installed on it.

    ==========PART 1 OF PROJECT COMPLETED! ==========
    PART 2: Not added in this project. But you can find the reference in the Ansible module Number 1 , 2 and 3, attached in the Ansible repository.
    ==========

11. After the instance has been generated, add the ansible in the controller machine. Create a user In the worker node (instance) and give relevant permissions. Perform the same steps in the controller machine.
    Generate the ssh key and copy the key on the worker node (instance) to connect with them via ssh.

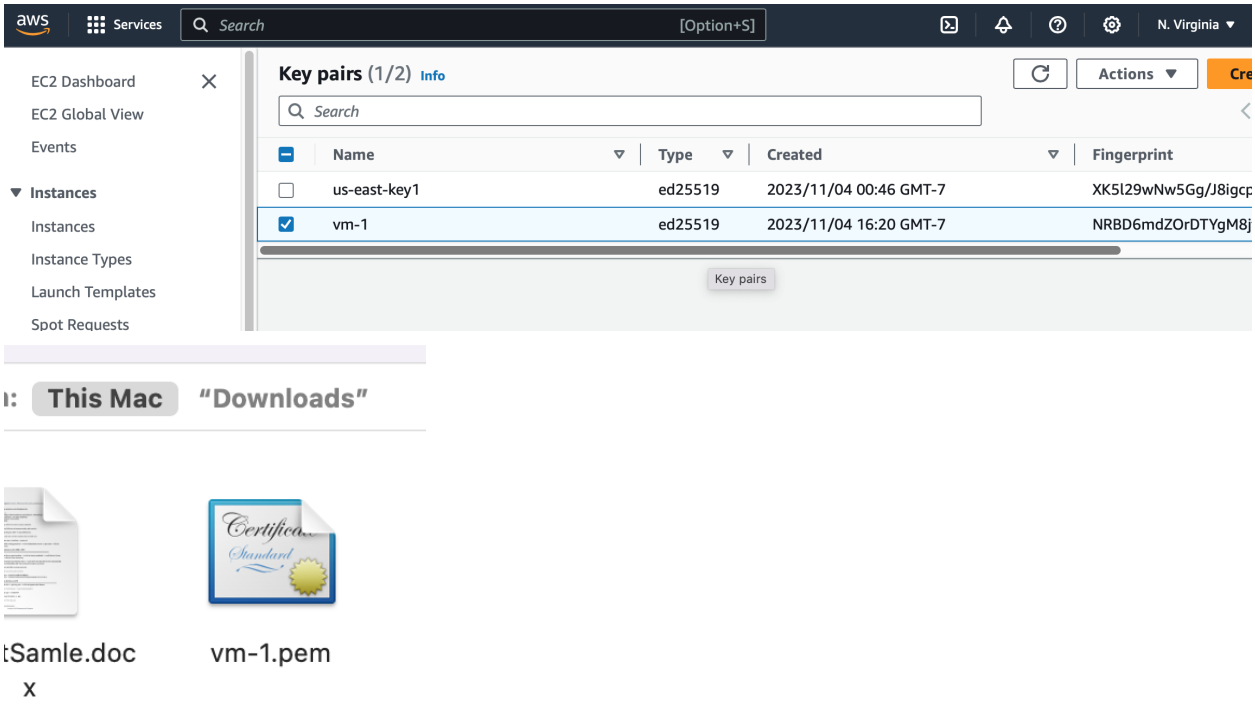
12. Add the worker node IP address in the host file of ansible and Ping the worker node to check if connection is successful or not. After successful connection, we will write ansible playbook to install java, python and Jenkins on our worker node.

13. Execute the playbook and check if the java, python and Jenkins were successfully installed or not on the worker node by checking their version, You can di it via ansible module or manually via worker node console.

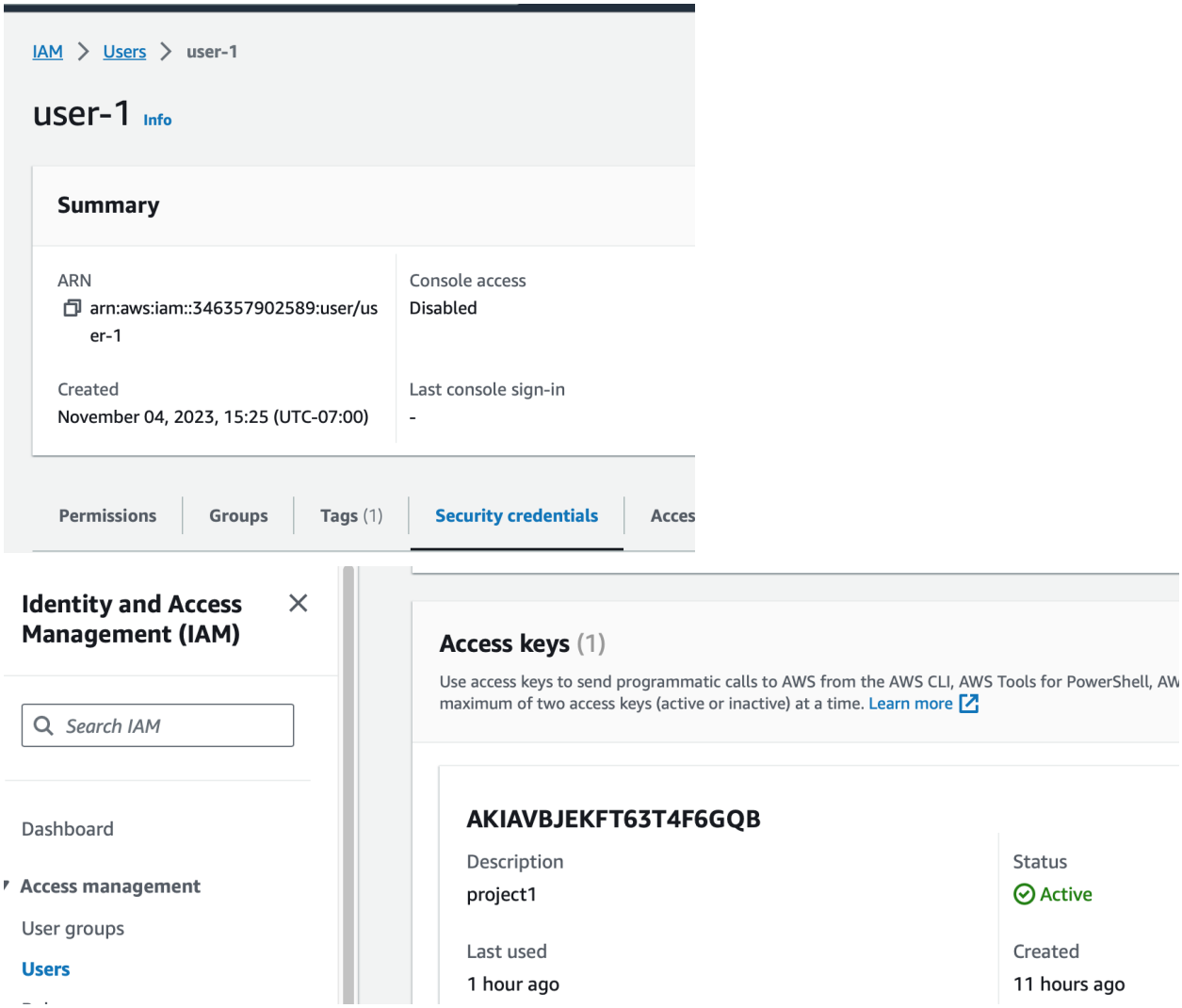
14. LINKS:-
    https://registry.terraform.io/providers/hashicorp/aws/latest/docs

EXECUTION :-

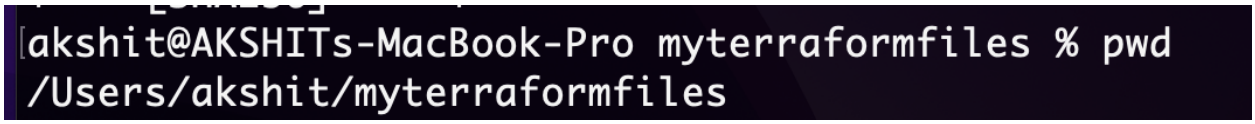
1. Create the key pair



2. Generate security key from IAM user



3. Add the keys credentials



—> vim aws_config.tf
In this file add the keys in this format
https://registry.terraform.io/providers/hashicorp/aws/latest/docs

```
provider "aws" {
  region     = "us-west-2"
  access_key = "my-access-key"
  secret_key = "my-secret-key"
}
```

4. You can save credentials in AWS configure as well

```
akshit@AKSHITs-MacBook-Pro myterraformfiles % aws configure
AWS Access Key ID [****************6GQB]:
AWS Secret Access Key [****************Hxf5]:
Default region name [None]:
Default output format [None]:
```

5. Here in shared_credentials_files we have not provided the secret access key directly but reference to the path of AWS home directory credentials file.

```
akshit@AKSHITs-MacBook-Pro myterraformfiles % cat aws_config.tf
provider "aws" {
  region     = "us-east-1"
  shared_credentials_files = ["~/.aws/credentials"]
}
```

6. Make a file with .tf extension with all the instance details

```
akshit@AKSHITs-MacBook-Pro myterraformfiles % cat aws_config.tf
provider "aws" {
  region     = "us-east-1"
  shared_credentials_files = ["~/.aws/credentials"]
}
```

7. Security Block

```
resource "aws_security_group" "Vm2SecurityGroup" {
  name        = "MySecurityGroup"
  description = "Allow TLS inbound traffic"

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

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

  egress {
    from_port        = 0
    to_port          = 0
protocol         = "-1"
    cidr_blocks      = ["0.0.0.0/0"]
  }

  tags = {
    Name = "Mynetwork"
  }
}
```

8. Add key and the script for installation. The AWS resource block will now looks like this

```
resource "aws_instance" "MyEC2machine" {
  ami           = "ami-0573324ffc6ebc574"
  instance_type = "t2.micro"
  tags = {
    Name = "virtualmachine2"
  }
 key_name = "vm-1"

 user_data = <<-EOF

      #!/bin/bash

        sudo apt update

        sudo apt install -y git ansible

   EOF

}
```

9. Connect resource block with network block.

```
resource "aws_network_interface_sg_attachment" "sg_attachment1" {
security_group_id = aws_security_group.Vm2SecurityGroup.id
network_interface_id = aws_instance.MyEC2machine.primary_network_interface_id
}
```

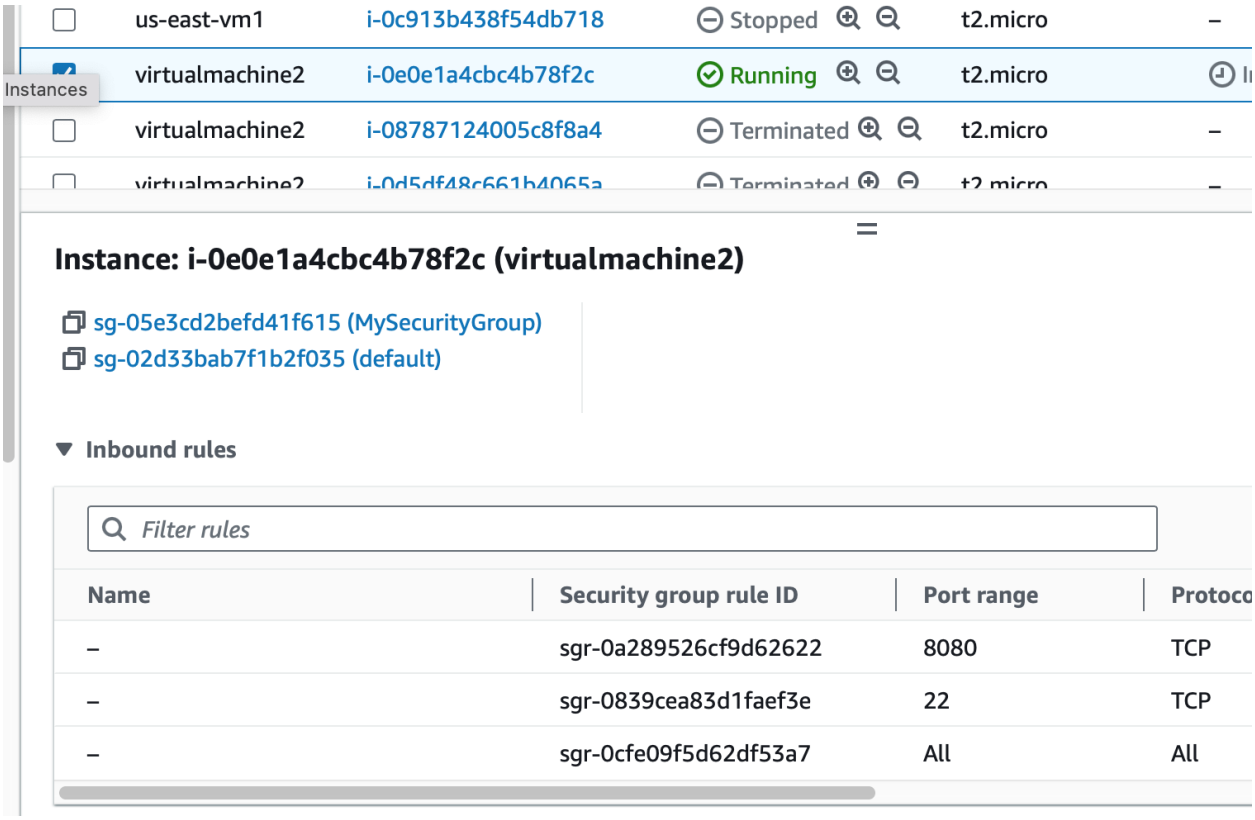
10. Apply terraform

```
akshit@AKSHITs-MacBook-Pro myterraformfiles % vim virtualmachine.tf
akshit@AKSHITs-MacBook-Pro myterraformfiles % terraform plan
aws_instance.MyEC2machine: Refreshing state... [id=i-0d5df48c661b4065a]
```

```
akshit@AKSHITs-MacBook-Pro myterraformfiles % terraform apply
aws_security_group.Vm2SecurityGroup: Refreshing state... [id=sg-05e3cd2befd41f615]
aws_instance.MyEC2machine: Refreshing state... [id=i-0d5df48c661b4065a]
```

```
aws_instance.MyEC2machine: Creation complete after 37s [id=i-0e0e1a4cbc4b78f2c]
aws_network_interface_sg_attachment.sg_attachment1: Creating...
aws_network_interface_sg_attachment.sg_attachment1: Creation complete after 1s [id=
]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```



```
No VM guests are running outdated hypervisor (qemu) binaries on this ho
ubuntu@ip-172-31-45-139:~$ ansible --version
ansible 2.10.8
  config file = None
  configured module search path = ['/home/ubuntu/.ansible/plugins/modul
  ansible python module location = /usr/lib/python3/dist-packages/ansib
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0]
ubuntu@ip-172-31-45-139:~$
```

i-0e0e1a4cbc4b78f2c (virtualmachine2)

PublicIPs: 52.90.184.237 PrivateIPs: 172.31.45.139

```
sudo apt install ansible        # version 2.10.
ubuntu@ip-172-31-45-139:~$ git --version
git version 2.34.1
```

Git and Ansible are successfully installed and instance has been successfully created.

==========PART 2 ==========

11. Connect via ssh.

On Worker Node

```
root@ip-172-31-17-14:~# adduser akshit
Adding user `akshit' ...
Adding new group `akshit' (1018) ...
Adding new user `akshit' (1018) with group `akshit' ...
Creating home directory `/home/akshit' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for akshit
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
root@ip-172-31-17-14:~# vim /etc/sudoers
root@ip-172-31-17-14:~# vim /etc/ssh/sshd_config
root@ip-172-31-17-14:~# systemctl restart sshd
```

On Controller machine

```
akshit@AKSHITs-MacBook-Pro ~ % ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/akshit/.ssh/id_rsa):
/Users/akshit/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/akshit/.ssh/id_rsa
Your public key has been saved in /Users/akshit/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:6PhOIFt4bPNHSTg6wT3l1MpM+KoF6/lbnjmjtRpe+uA akshit@AKSHITs-MacBook-Pro.local
The key's randomart image is:
+---[RSA 3072]----+
|       .o.       |
|   . ..=. .      |
|    o ==o.       |
|   o.o ==.       |
|  o Xo..S        |
|   *.Bo.         |
|  ...+* =        |
|    += Xo+       |
|     oE=*o       |
+----[SHA256]-----+
```

```
akshit@AKSHITs-MacBook-Pro ~ % ssh-copy-id akshit@18.212.205.25
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that
are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is
to install the new keys
akshit@18.212.205.25's password:

Number of key(s) added:        1

Now try logging into the machine, with:   "ssh 'akshit@18.212.205.25'"
and check to make sure that only the key(s) you wanted were added.
```

12.

```
nano ~/ansible/hosts
```

```
[webserver]
18.212.205.25
~
~
```

```
touch ~/ansible/hosts
```

```
akshit@AKSHITs-MacBook-Pro ~ % ansible -i ~/ansible/hosts webserver -m ping
18.212.205.25 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

check the hosts os distribution via fact variable in controller machine

```
akshit@AKSHITs-MacBook-Pro ansible % ansible -i hosts webserver -m setup -a "filter=ansible_os_family*"
18.212.205.25 | SUCCESS => {
    "ansible_facts": {
        "ansible_os_family": "Debian",
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false
}
```

As we can see, java, python and jenkins are not installed , already. So now, we will do it via playbook

```
ubuntu@ip-172-31-17-14:~$ jenkins --version
jenkins: command not found
ubuntu@ip-172-31-17-14:~$ python --version
Command 'python' not found, did you mean:
  command 'python3' from deb python3
  command 'python' from deb python-is-python3
ubuntu@ip-172-31-17-14:~$ java --version
Command 'java' not found, but can be installed with:
sudo apt install openjdk-11-jre-headless  # version 11.0.20.1+1-0ubuntu1~22.04, or
sudo apt install default-jre              # version 2:1.11-72build2
sudo apt install openjdk-17-jre-headless  # version 17.0.8.1+1~us1-0ubuntu1~22.04
sudo apt install openjdk-18-jre-headless  # version 18.0.2+9-2~22.04
sudo apt install openjdk-19-jre-headless  # version 19.0.2+7-0ubuntu3~22.04
sudo apt install openjdk-8-jre-headless   # version 8u382-ga-1~22.04.1
```

```
7d80d88dcb7b4fb3b73965c73cd76c22
akshit@AKSHITs-MacBook-Pro ~ % cd ~/ansible
akshit@AKSHITs-MacBook-Pro ansible % ls
hosts
akshit@AKSHITs-MacBook-Pro ansible % vim installationPlaybook.yml
```

```
- name: Installation packages playbook
  hosts: webserver
  become: true
  become_user: root
  tasks:
  - name: Apt_Update_repository
    command: apt-get update
  - name: install Java
    apt:
     name: openjdk-11-jdk
     state: present
  - name: install jenkins
    apt:
     name: jenkins
     state: present
  - name: start service jenkins
    service:
     name: jenkins
     state: started
  - name: install python
    apt:
     name: python
     state: present

~
~
~
```

After executing playbook , we can see the jenkins task is not completed , as we the package is not present in the apt repo of worker node, so we will have to add the url key manually in the worker nodes. beside that, let us check if python and jenkins have been successfully installed.

13. Check if packages are installed

Java is successfully installed. The python and jenkins will need the apt key url , which will be done manually.

So, we have successfully installed packages on our worker nodes via ansible.

==========