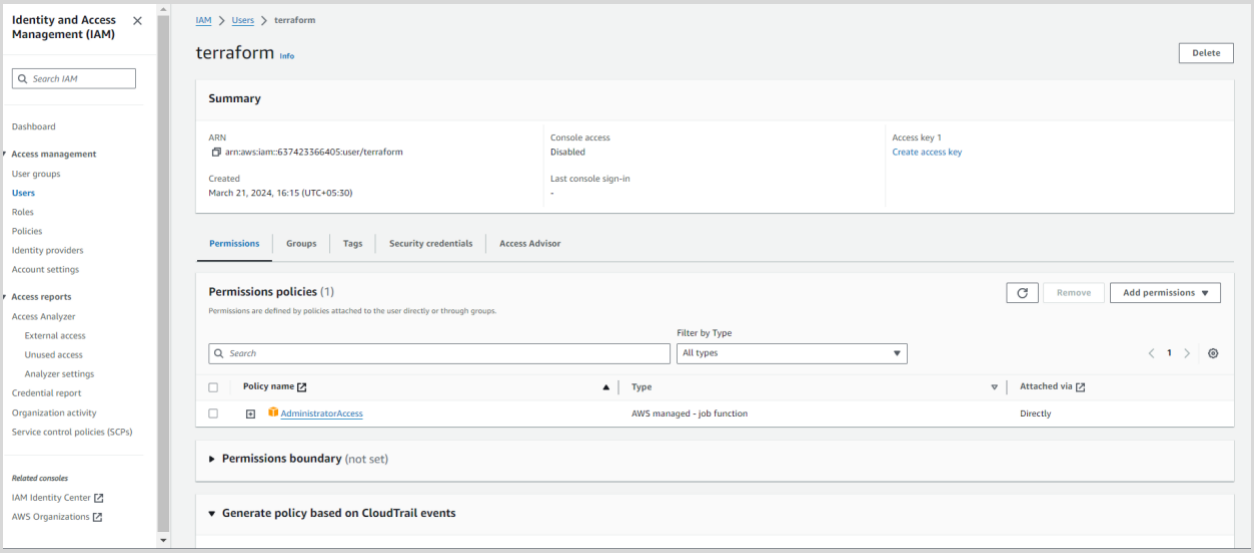
Problem Statement: To set up terraform environment, deploy, and destroy multiple EC2 instance using Terraform in a Cloud environment of your choice.

Tools : 1.Terraform 2.AWS EC2 3.AWS IAM

Solution: -

STEP 1:

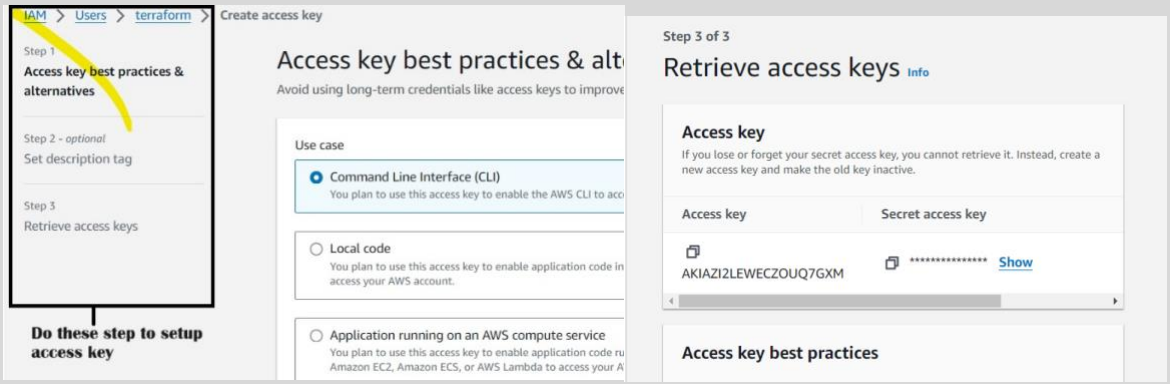
Create IAM user for the Terraform:



Create Access keys:

Access key: AKIAZI2LEWECQNK75HT6

Secrete access keys: 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



Now we will use EC2 instance for the setup so our local machine doesn't get hampered:

Connect to our EC2 instance:

```
PS C:\Users\INFINITY\Downloads> ssh -i .\ngnix.pem ec2-user@54.66.17.32
       #_
   ,     ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Thu Mar 21 11:05:09 2024 from 160.202.36.73
[ec2-user@ip-172-31-7-197 ~]$
```

STEP 2:

Now install Terraform in the machine:

Use yum-config-manager to add the official HashiCorp Linux repository:

```
sudo yum-config-manager --add-
repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
```

```
[root@ip-172-31-7-197 ~]# sudo yum-config-manager --add-repo https://rpm.rel
eases.hashicorp.com/AmazonLinux/hashicorp.reposudo yum-config-manager --add-
repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
Adding repo from: https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.r
eposudo
Adding repo from: https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.r
epo
[root@ip-172-31-7-197 ~]#
```

install:

Yum install terraform.

```
[root@ip-172-31-7-197 ~]# yum install terraform
Hashicorp Stable - x86_64                       11 MB/s | 1.4 MB     00:00
created by dnf config-manager from https:// 263  B/s | 321  B     00:01
Errors during downloading metadata for repository 'rpm.releases.hashicorp.c
m_AmazonLinux_hashicorp.reposudo':
```

STEP 3:

Now install AWS Cli to perform our terraform task on AWS:

yum install aws* -y

```
[root@ip-172-31-7-197 terraform]# yum install aws* -y
created by dnf config-manager from https:// 271  B/s | 333  B     00:01
Errors during downloading metadata for repository 'rpm.releases.hashicor
m_AmazonLinux_hashicorp.reposudo':
  - Status code: 404 for https://rpm.releases.hashicorp.com/AmazonLinux/
icorp.reposudo/repodata/repomd.xml (IP: 18.67.93.117)
Error: Failed to download metadata for repo 'rpm.releases.hashicorp.com_
onLinux_hashicorp.reposudo': Cannot download repomd.xml: Cannot download
odata/repomd.xml: All mirrors were tried
Ignoring repositories: rpm.releases.hashicorp.com_AmazonLinux_hashicorp.
sudo
Last metadata expiration check: 0:17:41 ago on Thu Mar 21 11:34:20 2024.
Package aws-cfn-bootstrap-2.0-29.amzn2023.noarch is already installed.
Package awscli-2-2.14.5-1.amzn2023.0.1.noarch is already installed
```

Configure our AWS CLI:

Provide access key and secret access key also in the aws cli configuration.

aws configure

Access key: AKIAZI2LEWECQNK75HT6

Secrete access keys: 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

```
[root@ip-172-31-7-197 terraform]# aws configure
AWS Access Key ID [None]: AKIAZI2LEWECQNK75HT6
AWS Secret Access Key [None]:  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
Default region name [None]:
Default output format [None]:
[root@ip-172-31-7-197 terraform]#
```

STEP 4:

Now let's create Terraform directory to store and perform tf files:

```
[root@ip-172-31-7-197 ~]# mkdir terraform
[root@ip-172-31-7-197 ~]# cd terraform
[root@ip-172-31-7-197 terraform]#
```

Create your tf file for ec2 instance to launch:

vim ec2.tf

```
[root@ip-172-31-7-197 terraform]# vim ec2.tf
[root@ip-172-31-7-197 terraform]#
```

Now lets write the code the :

```
provider "aws" {
    access_key = "AKIAZI2LEWECQNK75HT6"
    secret_key = "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"
    region = "ap-southeast-2"
}

resource "aws_instance" "ec2_G1" {
    ami = "ami-004c37117ce961527"
    instance_type = "t2.micro"

}
resource "aws_instance" "ec2_G2" {
    ami = "ami-004c37117ce961527"
    instance_type = "t2.micro"
}
resource "aws_instance" "ec2_G3" {
    ami ="ami-004c37117ce961527"
    instance_type = "t2.micro"
}
```

Here you can see I have take security group as an default and doesn't provide any specific ip taken it as an default allotted by the aws same as to subnet.

I have provided:

information of IAM

information AWS region which is in my case was Sydney.

Information of ami

Information of cpu which is in my case t2.micro

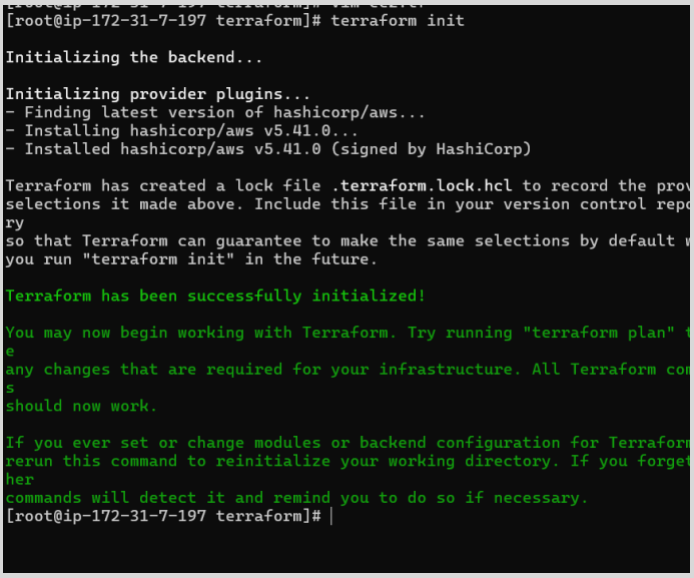
We will create 3 ec2 with name ec2_G1, ec2_G2, and ec2_G3.

```
provider "aws" {
    access_key = "AKIAZI2LEWECQNK75HT6"
    secret_key = "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"
    region = "ap-southeast-2"
}

resource "aws_instance" "ec2_G1" {
    ami = "ami-004c37117ce961527"
    instance_type = "t2.micro"

}
resource "aws_instance" "ec2_G2" {
    ami = "ami-004c37117ce961527"
    instance_type = "t2.micro"
}
resource "aws_instance" "ec2_G3" {
    ami ="ami-004c37117ce961527"
    instance_type = "t2.micro"
}
```

STEP 5:

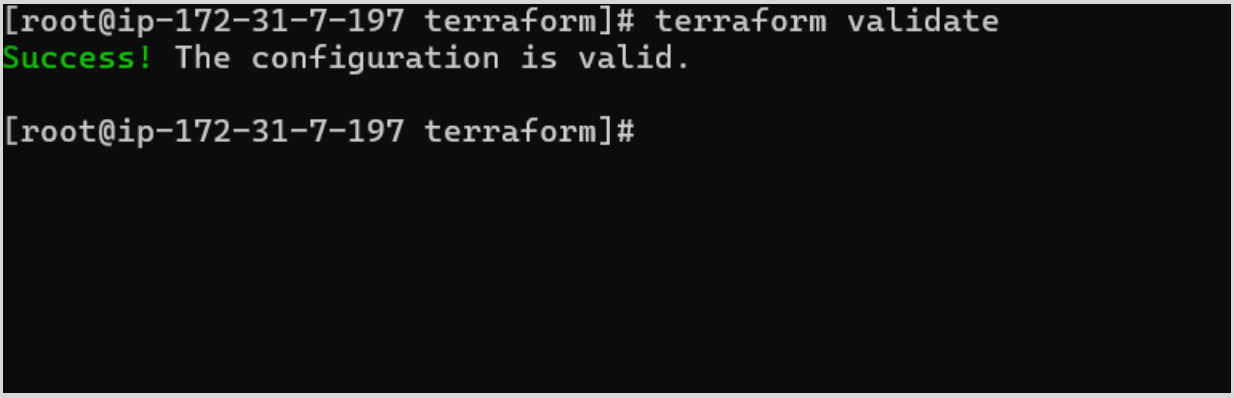
Now lets perform terraform commands in the file :

terraform init



Successfully validate our Code:

terraform validate



Now Perform PLAN command to know what will be create by applying the code:

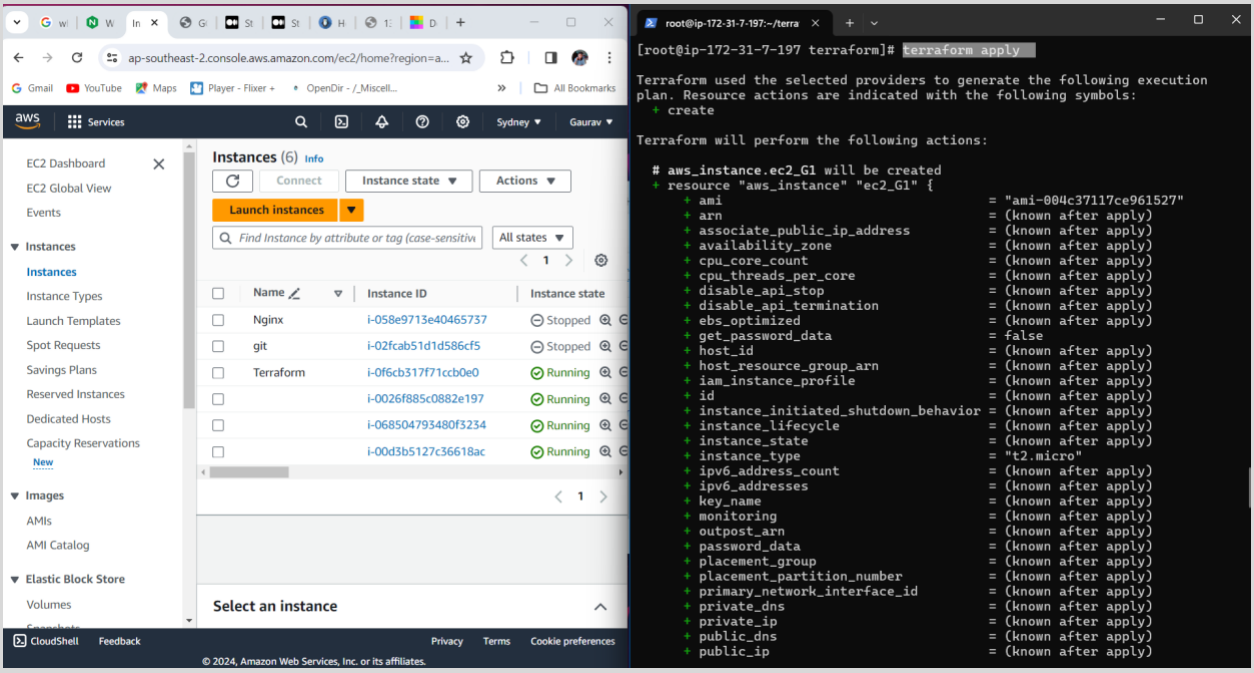
terraform plan

```
[root@ip-172-31-7-197 terraform]# terraform plan

Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.ec2_G1 will be created
  + resource "aws_instance" "ec2_G1" {
      + ami                                  = "vpc-034875c5ff5a740b1"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
```

Now apply and create the instances:

terraform apply

As you can see we have create three instances and they are running.



STEP 6:

Command to destroy to specific instance from the instances that was launch:

terraform destroy -target **Name_of_the_instances**

```
[root@ip-172-31-7-197 terraform]# terraform destroy -target aws_instance.ec2_G1
aws_instance.ec2_G1: Refreshing state... [id=i-068504793480f3234]

Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.ec2_G1 will be destroyed
  - resource "aws_instance" "ec2_G1" {
      - ami                                  = "ami-004c37117ce961527" -> null
      - arn                                  = "arn:aws:ec2:ap-southeast-2:637423366405:instance/i-068504793480f3234" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "ap-southeast-2b" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 1 -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
      - ebs_optimized                        = false -> null
      - get_password_data                    = false -> null
      - hibernation                          = false -> null
      - id                                   = "i-068504793480f3234" -> null
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
      - instance_type                        = "t2.micro" -> null
      - ipv6_address_count                   = 0 -> null
      - ipv6_addresses                       = [] -> null
      - monitoring                           = false -> null
      - placement_partition_number           = 0 -> null
      - primary_network_interface_id         = "eni-051ed872466e00ca4" -> null
      - private_dns                          = "ip-172-31-8-148.ap-southeast-2.compute.internal" -> null
      - private_ip                           = "172.31.8.148" -> null
      - public_dns                           = "ec2-54-253-24-128.ap-southeast-2.compute.amazonaws.com" -> null
      - public_ip                            = "54.253.24.128" -> null
      - secondary_private_ips                = [] -> null
      - security_groups                      = [
          - "default",
        ] -> null
      - source_dest_check                    = true -> null
```

To destroy all the instances:

terraform destroy

```
[root@ip-172-31-7-197 terraform]# terraform destroy
aws_instance.ec2_G3: Refreshing state... [id=i-00d3b5127c36618ac]
aws_instance.ec2_G2: Refreshing state... [id=i-0026f885c0882e197]

Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.ec2_G2 will be destroyed
  - resource "aws_instance" "ec2_G2" {
      - ami                                  = "ami-004c37117ce961527" ->
ll
      - arn                                  = "arn:aws:ec2:ap-southeast-
37423366405:instance/i-0026f885c0882e197" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "ap-southeast-2b" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 1 -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
      - ebs_optimized                        = false -> null
      - get_password_data                    = false -> null
      - hibernation                          = false -> null
      - id                                   = "i-0026f885c0882e197" -> n
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
      - instance_type                        = "t2.micro" -> null
      - ipv6_address_count                   = 0 -> null
      - ipv6_addresses                       = [] -> null
      - monitoring                           = false -> null
      - placement_partition_number           = 0 -> null
      - primary_network_interface_id         = "eni-0917d43cac2353481" ->
ll
      - private_dns                          = "ip-172-31-1-120.ap-southe
-2.compute.internal" -> null
      - private_ip                           = "172.31.1.120" -> null
      - public_dns                           = "ec2-3-25-80-138.ap-southe
-2.compute.amazonaws.com" -> null
      - public_ip                            = "3.25.80.138" -> null
```

We have successfully destroy all the instance that was created .

Conclusion:

In summary, setting up a Terraform environment to deploy and destroy multiple EC2 instances in a chosen cloud environment offers valuable insights into modern cloud infrastructure management. This project emphasizes the efficiency, scalability, and consistency achieved through infrastructure as code principles. By mastering Terraform, individuals can automate resource provisioning, mitigate errors, and adapt swiftly to evolving business requirements. Such proficiency is vital in maximizing the benefits of cloud technologies and advancing in the dynamic realm of DevOps practices.

Pardon me if i missed any step in between.

THANK YOU