Task 18: create 2 EC2 instance on 2 different regions and install nginx using terraform script

1. Installed terraform:

```
ubuntu@ip-172-31-35-127:~$ wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/h
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release
hashicorp.list
sudo apt update && sudo apt install terraform
--2024-07-27 10:35:14--  https://apt.releases.hashicorp.com/gpg
Resolving apt.releases.hashicorp.com (apt.releases.hashicorp.com)... 99.84.108.40, 99.84.108.74, 99.84.108.3, ...
Connecting to apt.releases.hashicorp.com (apt.releases.hashicorp.com)|99.84.108.40|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3980 (3.9K) [binary/octet-stream]
Saving to: 'STDOUT'
```

```
ubuntu@ip-172-31-35-127:~$ terraform --version
Terraform v1.9.3
on linux_amd64
```

2. Create a directory for write terraform file:

```
ubuntu@ip-172-31-35-127:~$ mkdir terraform
ubuntu@ip-172-31-35-127:~$ cd terraform/
ubuntu@ip-172-31-35-127:~/terraform$ vi main.tf
```

3. Main.tf file:

```
provider "aws" {
alias = "us_east_2"
region = "us-east-2"
}
provider "aws" {
alias = "us_west_2"
region = "us-west-2"
}
resource "aws_instance" "demo1" {
provider = aws.us_east_2
instance_type = "t2.micro"
ami = "ami-00db8dadb36c9815e"
user_data = <<-EOF
                #!/bin/bash
                sudo yum update -y
                sudo yum install nginx -y
                sudo systemctl start nginx
                sudo systemctl enable nginx
                EOF


tags = {
Name = "ec2-useast2"
}
}
```

```
resource "aws_instance" "demo2" {
provider = aws.us_west_2
instance_type = "t2.micro"
ami = "ami-074be47313f84fa38"
user_data = <<-EOF
              #!/bin/bash
              sudo yum update -y
              sudo yum install nginx -y
              sudo systemctl start nginx
              sudo systemctl enable nginx
              EOF

tags = {
Name = "ec2-uswest2"
}
}
```

Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Li

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI — Free tier eligible

ami-00db8dadb36c9815e (64-bit (x86), uefi-preferred) / ami-0fb5231409345e557 (64-bit (Arm), uefi)

Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Architecture	Boot mode	AMI ID	
64-bit (x86)	uefi-preferred	ami-00db8dadb36c9815e	Verified provider

Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Li

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI — Free tier eligible

ami-074be47313f84fa38 (64-bit (x86), uefi-preferred) / ami-07200707e433337ed (64-bit (Arm), uefi)

Virtualization: hvm ENA enabled: true Root device type: ebs

Amazon Linux 2023 AMI

Description

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Architecture	Boot mode	AMI ID	
64-bit (x86)	uefi-preferred	ami-074be47313f84fa38	Verified provider

4. Terraform init:

```
ubuntu@ip-172-31-35-127:~/terraform$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.60.0...
- Installed hashicorp/aws v5.60.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
```

5. Configure aws cli:

```
ubuntu@ip-172-31-35-127:~/terraform$ aws configure
AWS Access Key ID [None]: AKIASEMTU4DA4CF6MYF7
AWS Secret Access Key [None]: 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
Default region name [None]: us-west-1
Default output format [None]: json
ubuntu@ip-172-31-35-127:~/terraform$
```

6. Terraform plan:

```
ubuntu@ip-172-31-35-127:~/terraform$ terraform plan

Terraform used the selected providers to generate the following execution plan.
  + create

Terraform will perform the following actions:

  # aws_instance.demo1 will be created
  + resource "aws_instance" "demo1" {
      + ami                                  = "ami-00db8dadb36c9815e"
      + 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)
```

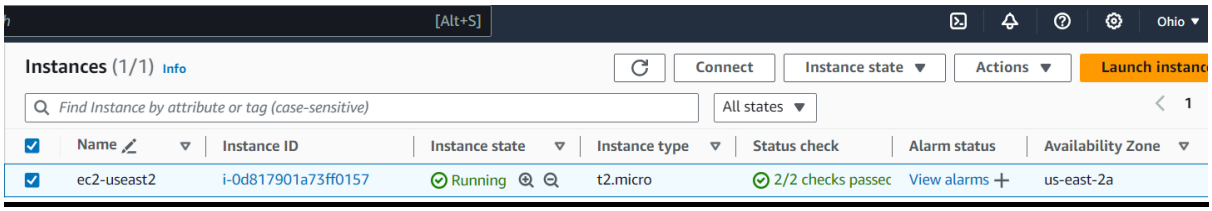
```
  # aws_instance.demo2 will be created
  + resource "aws_instance" "demo2" {
      + ami                                  = "ami-074be47313f84fa38"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (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"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
```

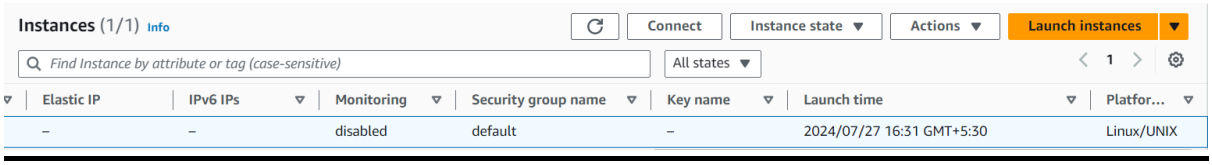
7. Terraform apply:

```
ubuntu@ip-172-31-35-127:~/terraform$ terraform apply

Terraform used the selected providers to generate the following execution plan.
  + create

Terraform will perform the following actions:

  # aws_instance.demo1 will be created
  + resource "aws_instance" "demo1" {
      + ami                                  = "ami-00db8dadb36c9815e"
      + 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)
```

```
aws_instance.demo1: Creating...
aws_instance.demo2: Creating...
aws_instance.demo1: Still creating... [10s elapsed]
aws_instance.demo2: Still creating... [10s elapsed]
aws_instance.demo1: Still creating... [20s elapsed]
aws_instance.demo2: Still creating... [20s elapsed]
aws_instance.demo1: Still creating... [30s elapsed]
aws_instance.demo2: Still creating... [30s elapsed]
aws_instance.demo1: Creation complete after 32s [id=i-0d817901a73ff0157]
aws_instance.demo2: Still creating... [40s elapsed]
aws_instance.demo2: Creation complete after 43s [id=i-08019211f56ef731f]

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

8. Ec2 created in region us-east-2:





Open the port no:80

i-0d817901a73ff0157 (ec2-useast2)

sg-0f13fd7f6a7e7d650 (default)

Inbound rules

Filter rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups
–	sgr-0cbbe56456f901f6f	All	All	sg-0f13fd7f6a7e7d650	default
–	sgr-00b81e7d6c77ee4ad	80	TCP	0.0.0.0/0	default

Nginix installed in us-east-2:

Not secure 3.138.181.145

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

9. Ec2 created in region us-west-2:

[Alt+S] Oregon

Instances (1/1) Info

Connect | Instance state | Actions | Launch instance

Find Instance by attribute or tag (case-sensitive) | All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
ec2-uswest2	i-08019211f56ef731f	Running	t2.micro	Initializing	View alarms	us-west-2b

Instances (1/1) Info

Connect | Instance state | Actions | Launch instances

Find Instance by attribute or tag (case-sensitive) | All states

Elastic IP	IPv6 IPs	Monitoring	Security group name	Key name	Launch time	Platfor...
–	–	disabled	default	–	2024/07/27 16:31 GMT+5:30	Linux/UNIX

Open the port no:80 :

i-08019211f56ef731f (ec2-uswest2)

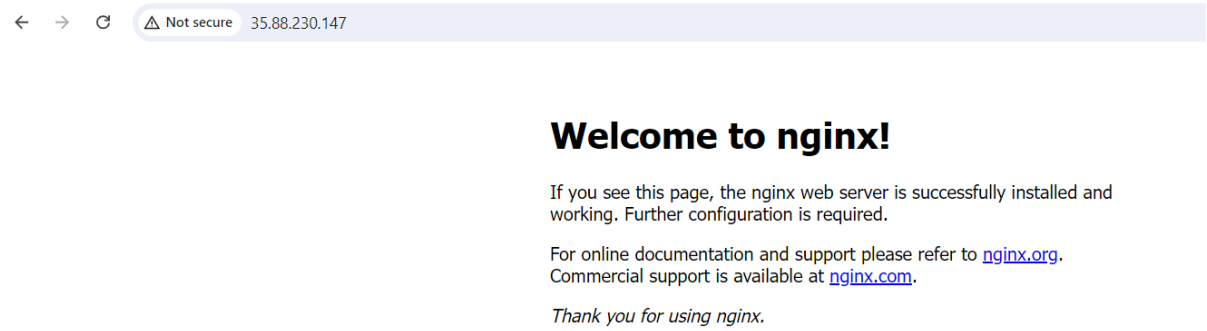
sg-0fa97a999f440a222 (default)

Inbound rules

Filter rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups
–	sgr-0c688763554346258	All	All	sg-0fa97a999f440a222	default
–	sgr-0289b71250bd67581	80	TCP	0.0.0.0/0	default

Nginix installed in us-west-2:



Main.tf file:

```
provider "aws" {
alias = "us_east_1"
region = "us-east-1"
}
provider "aws" {
alias = "us_west__2"
region = "us-west-2"
}
resource "aws_instance" "demo1" {
provider = aws.us_east_1
instance_type = "t2.micro"
ami = "ami-0427090fd1714168b"
user_data = <<-EOF
        #!/bin/bash
        sudo yum update -y
        sudo yum install nginx -y
        sudo systemctl start nginx
        sudo systemctl enable nginx
        EOF
```

```
tags = {
Name = "ec2-useast1"
}
}
resource "aws_instance" "demo2" {
provider = aws.us_west_2
instance_type = "t2.micro"
ami = "ami-074be47313f84fa38"
user_data = <<-EOF
        #!/bin/bash
        sudo yum update -y
        sudo yum install nginx -y
        sudo systemctl start nginx
        sudo systemctl enable nginx
        EOF

tags = {
Name = "ec2-uswest2"
}
}
```