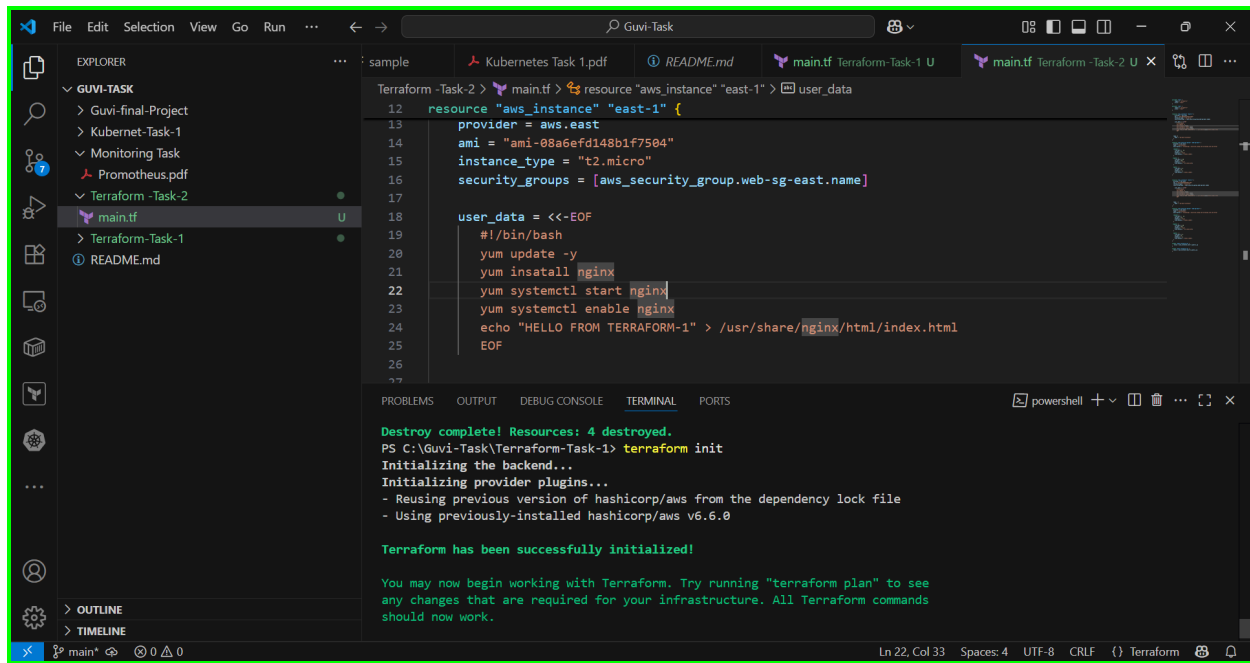


Create 2 EC2 instances on 2 different regions and install nginx using terraform script

The script file is attached in the same git repo, so as per the requirement the script is created and

Applying terraform init



The screenshot shows the Visual Studio Code interface with a Terraform script open in the editor. The script defines an AWS instance named 'east-1' in the 'east' region, using the 'ami-08a6efd148b1f7504' AMI and 't2.micro' instance type. It also includes a user_data block to install and start Nginx. The terminal window at the bottom shows the output of the 'terraform init' command, indicating that the backend is initialized and the provider plugins are ready.

```
resource "aws_instance" "east-1" {
  provider = aws.east
  ami = "ami-08a6efd148b1f7504"
  instance_type = "t2.micro"
  security_groups = [aws_security_group.web-sg-east.name]

  user_data = <<-EOF
  #!/bin/bash
  yum update -y
  yum install nginx
  yum systemctl start nginx
  yum systemctl enable nginx
  echo "HELLO FROM TERRAFORM-1" > /usr/share/nginx/html/index.html
  EOF
}
```

```
Destroy complete! Resources: 4 destroyed.
PS C:\Guvi-Task\Terraform-Task-1> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.6.0

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.
```

The terraform init is successful, now give terraform validate for the code

The screenshot shows a Visual Studio Code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named 'GUVI-TASK' with subfolders 'Guvi-final-Project', 'Kubernetes-Task-1', 'Monitoring Task', 'Prometheus.pdf', and 'Terraform -Task-2'. The 'Terraform -Task-2' folder is expanded, showing 'main.tf' and 'README.md'. The 'main.tf' file is open in the editor, showing a Terraform configuration for an AWS instance. The configuration includes a resource 'aws_instance' named 'east-1' with provider 'aws.east', ami 'ami-08a6efd148b1f7504', instance_type 't2.micro', and security_groups ['aws_security_group.web-sg-east.name']. It also includes a user_data block with a shell script to install and start nginx. The terminal at the bottom shows the command 'terraform validate' being executed, resulting in a success message: 'Success! The configuration is valid.'

```
12 resource "aws_instance" "east-1" {
13   provider = aws.east
14   ami = "ami-08a6efd148b1f7504"
15   instance_type = "t2.micro"
16   security_groups = [aws_security_group.web-sg-east.name]
17
18   user_data = <<-EOF
19   #!/bin/bash
20   yum update -y
21   yum insatall nginx
22   yum systemctl start nginx
23   yum systemctl enable nginx
24   echo "HELLO FROM TERRAFORM-1" > /usr/share/nginx/html/index.html
25   EOF
26
27
```

```
PS C:\Guvi-Task\Terraform-Task-1> terraform validate
Success! The configuration is valid.

PS C:\Guvi-Task\Terraform-Task-1>
```

Now give terraform plan to know is any change is done

The screenshot shows the same Visual Studio Code editor as before, but now the terminal displays the output of the 'terraform plan' command. The output indicates that the configuration is valid and shows the actions Terraform will perform: creating the 'aws_instance.east-1' resource. The output also shows the specific ami value for the instance: 'ami-08a6efd148b1f7504'.

```
PS C:\Guvi-Task\Terraform-Task-1> 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.east-1 will be created
+ resource "aws_instance" "east-1" {
+   ami = "ami-08a6efd148b1f7504"
```

Finally apply terraform apply to execute the script

The screenshot shows a VS Code editor with a file explorer on the left containing a project named 'GUVI-TASK'. The main editor displays a Terraform configuration file 'main.tf' for 'Terraform-Task-2'. The configuration defines an AWS instance 'east-1' with the following details:

- provider: aws.east
- ami: "ami-08a6efd148b1f7504"
- instance_type: "t2.micro"
- security_groups: [aws_security_group.web-sg-east.name]
- user_data: A script to update yum, install nginx, start and enable it, and echo a message.

The terminal at the bottom shows the command 'terraform apply' being executed. The output indicates that 4 resources will be added, with no changes or destructions. The instance IP addresses are listed as (known after apply).

On the apply is done it get create the instance and install the nfginx finally we will get the ips of the instance

This screenshot shows the continuation of the Terraform apply process. The terminal output shows the progress of creating two instances:

- aws_instance.west-1: Still creating... [00m20s elapsed]
- aws_instance.east-1: Still creating... [00m20s elapsed]
- aws_instance.west-1: Creation complete after 25s [id=i-063df1da5c4cc1608]
- aws_instance.east-1: Still creating... [00m30s elapsed]
- aws_instance.east-1: Creation complete after 35s [id=i-0fab98d4ae06ee9fb]

The final output shows 'Apply complete! Resources: 4 added, 0 changed, 0 destroyed.' and lists the IP addresses for both instances:

- east_instance_ip = "54.162.240.42"
- weat_instance_ip = "54.193.214.49"

Copy those IP and check in the browser we will nginx output from two different servers.

Terraform server - 1



Terraform server - 2

