

Terraform Task-2

Task Description:

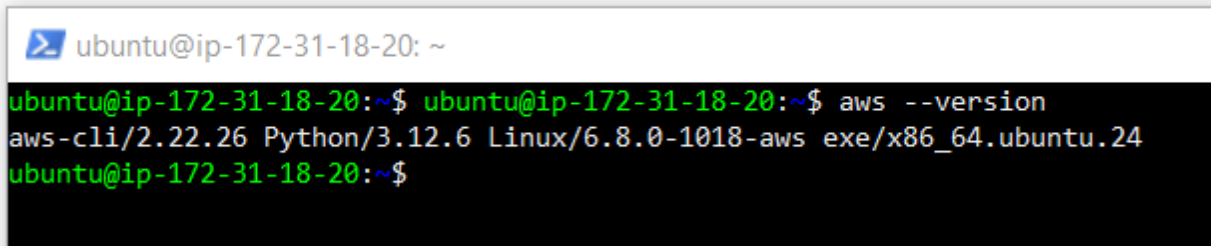
Create 2 EC2 instances on 2 different regions and install nginx using terraform script.

Techstacks needs to be used :

- AWS EC2
- Terraform
- AWS CLI

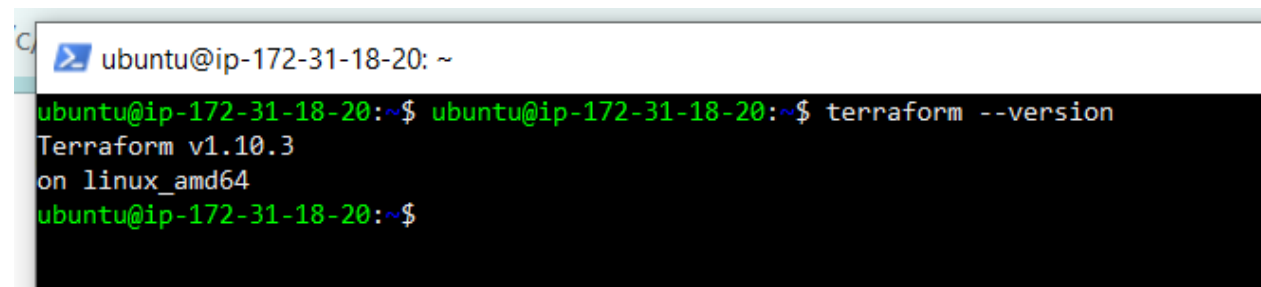
Step1

- Install AWS CLI

A terminal window with a dark background and light green text. The prompt is 'ubuntu@ip-172-31-18-20: ~'. The command 'aws --version' has been executed, and the output is 'aws-cli/2.22.26 Python/3.12.6 Linux/6.8.0-1018-aws exe/x86_64.ubuntu.24'.

```
ubuntu@ip-172-31-18-20: ~  
ubuntu@ip-172-31-18-20:~$ aws --version  
aws-cli/2.22.26 Python/3.12.6 Linux/6.8.0-1018-aws exe/x86_64.ubuntu.24  
ubuntu@ip-172-31-18-20:~$
```

- Install Terraform

A terminal window with a dark background and light green text. The prompt is 'ubuntu@ip-172-31-18-20: ~'. The command 'terraform --version' has been executed, and the output is 'Terraform v1.10.3 on linux_amd64'.

```
ubuntu@ip-172-31-18-20: ~  
ubuntu@ip-172-31-18-20:~$ terraform --version  
Terraform v1.10.3  
on linux_amd64  
ubuntu@ip-172-31-18-20:~$
```

Step2

- Set up IAM user and credential.

satthya-terraform

Info

Delete

Summary

ARN
arn:aws:iam::529088293565:user/satthya-terraform

Created
January 02, 2025, 14:42 (UTC+08:00)

Console access
Disabled

Last console sign-in
-

Access key 1
AKIAIAXWMA6VK6VEFV4DUF - Active
ⓘ Never used. Created today.

Access key 2
Create access key

Permissions

Groups

Tags
(1)

Security credentials

Last Accessed

Permissions policies (2)

Remove

Add permissions

Permissions are defined by policies attached to the user directly or through groups.

Search

Filter by Type
All types

< 1 > ⚙

☐

Policy name

▲

Type

▼

Attached via

☐

AdministratorAccess

AWS managed - job function

Directly

☐

AmazonEC2FullAccess

AWS managed

Directly

IAM > Users > satthya-terraform > Create access key

Access key created

This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

Step 1
Access key best practices & alternatives

Step 2 - optional
Set description tag

Step 3
Retrieve access keys

Retrieve access keys

Info

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key

AKIAIAXWMA6VK6VEFV4DUF

Secret access key

***** Show

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

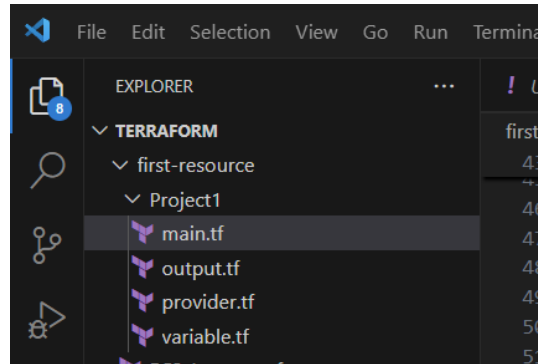
For more details about managing access keys, see the [best practices for managing AWS access keys](#).

Download .csv file

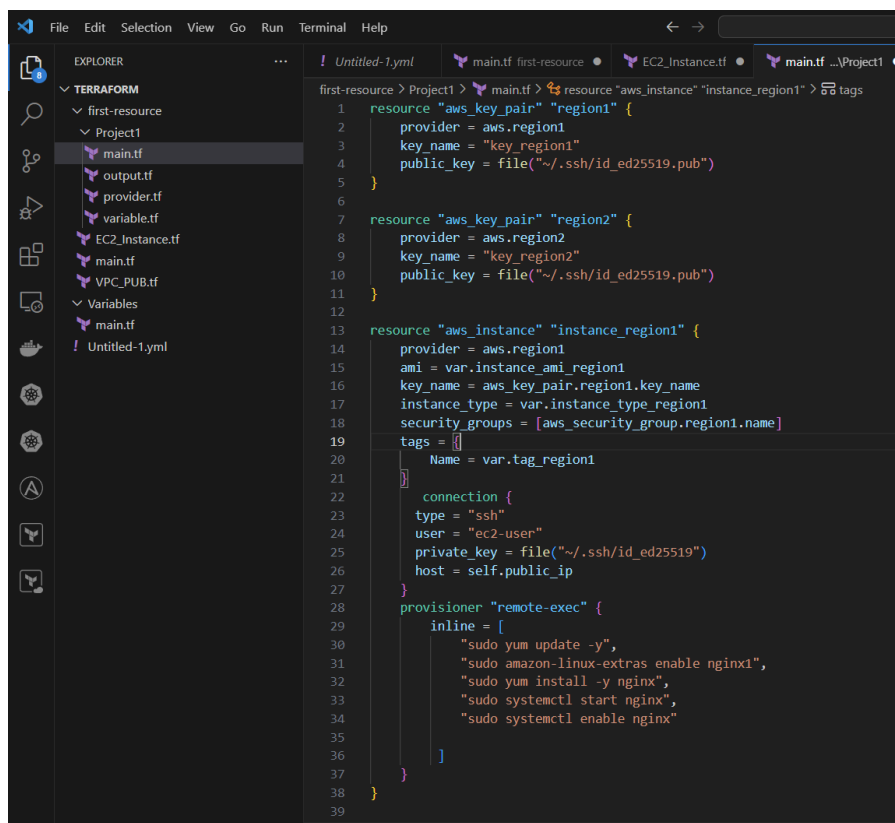
Done

Step3

- Create Terraform Configuration file.
 - Project1
 - main.tf
 - provider.tf
 - variable.tf
 - output.tf



 main.tf

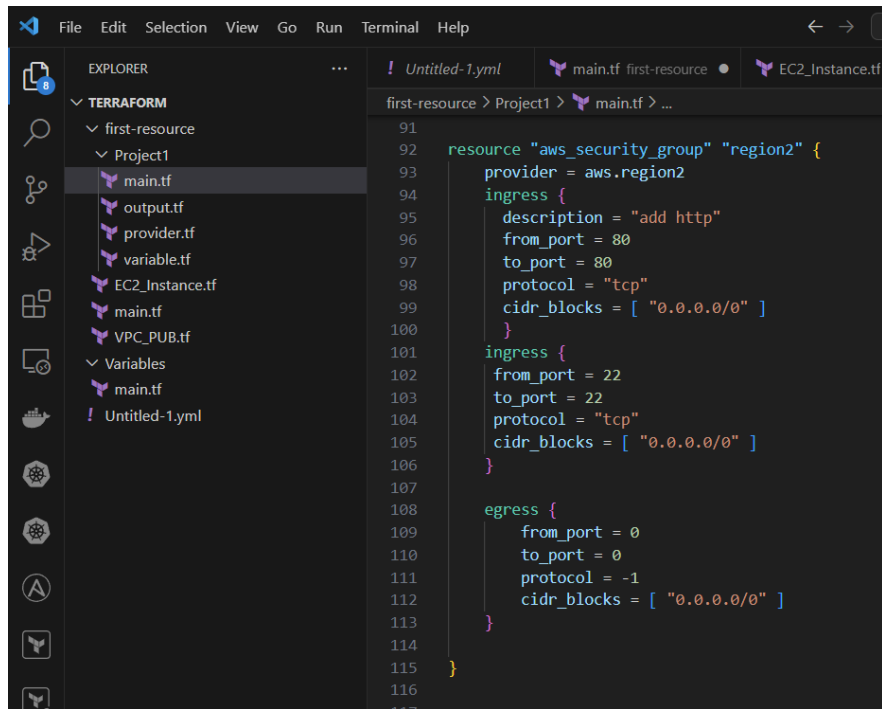


This screenshot shows the VS Code interface with the Terraform Explorer on the left and the main editor on the right. The Explorer shows a project structure under 'TERRAFORM' with folders 'first-resource' and 'Variables', and a 'Project1' folder containing files like 'main.tf', 'output.tf', 'provider.tf', 'variable.tf', 'EC2_Instance.tf', and 'VPC_PUB.tf'. The main editor displays the content of 'main.tf' for 'Project1', showing a Terraform resource definition for 'aws_instance' named 'instance_region2'. The configuration includes provider settings, ami, key_name, instance_type, security_groups, and tags. It also defines an SSH connection and a remote-exec provisioner to install and enable nginx.

```
41 resource "aws_instance" "instance_region2" {
42   provider = aws.region2
43   ami = var.instance_ami_region2
44   key_name = aws_key_pair.region2.key_name
45   instance_type = var.instance_type_region2
46   security_groups = [aws_security_group.region2.name]
47   tags = {
48     Name = var.tag_region2
49   }
50   connection {
51     type = "ssh"
52     user = "ec2-user"
53     private_key = file("~/ssh/id_ed25519")
54     host = self.public_ip
55   }
56   provisioner "remote-exec" {
57     inline = [
58       "sudo yum update -y",
59       "sudo amazon-linux-extras enable nginx1",
60       "sudo yum install -y nginx",
61       "sudo systemctl start nginx",
62       "sudo systemctl enable nginx"
63     ]
64   }
65 }
```

This screenshot shows the VS Code interface with the Terraform Explorer on the left and the main editor on the right. The Explorer shows a project structure under 'TERRAFORM' with folders 'first-resource' and 'Variables', and a 'Project1' folder containing files like 'main.tf', 'output.tf', 'provider.tf', 'variable.tf', 'EC2_Instance.tf', and 'VPC_PUB.tf'. The main editor displays the content of 'main.tf' for 'Project1', showing a Terraform resource definition for 'aws_security_group' named 'region1'. The configuration includes provider settings and two ingress rules: one for port 80 and another for port 22. It also includes an egress rule for all traffic.

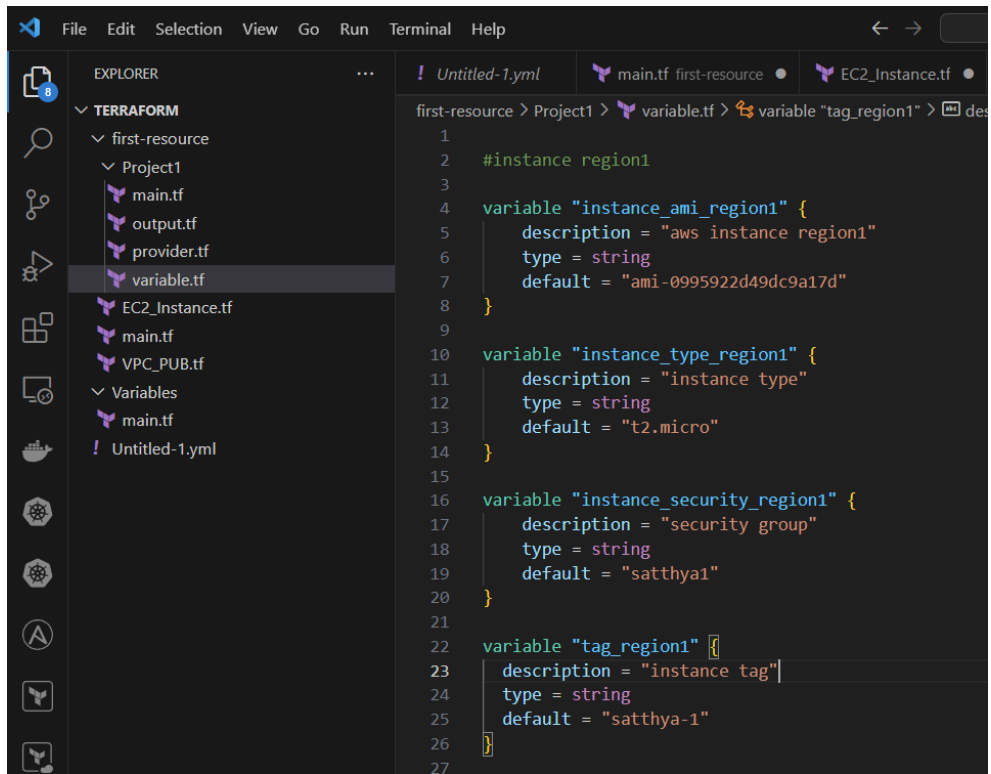
```
66 |
67 resource "aws_security_group" "region1" {
68   provider = aws.region1
69   ingress {
70     description = "add http"
71     from_port = 80
72     to_port = 80
73     protocol = "tcp"
74     cidr_blocks = [ "0.0.0.0/0" ]
75   }
76   ingress {
77     from_port = 22
78     to_port = 22
79     protocol = "tcp"
80     cidr_blocks = [ "0.0.0.0/0" ]
81   }
82   egress {
83     from_port = 0
84     to_port = 0
85     protocol = "-1"
86     cidr_blocks = [ "0.0.0.0/0" ]
87   }
88 }
89
90 }
```



The screenshot shows the VS Code interface with the Terraform Explorer on the left. The Explorer shows a project structure under 'TERRAFORM' with folders 'first-resource' and 'Project1'. Under 'Project1', there are files 'main.tf', 'output.tf', 'provider.tf', 'variable.tf', 'EC2_Instance.tf', and 'VPC_PUB.tf'. The 'main.tf' file is selected. The main editor shows the content of 'main.tf', which defines an 'aws_security_group' resource named 'region2'. The resource has a provider of 'aws.region2' and two ingress rules. The first ingress rule has a description of 'add http', from_port of 80, to_port of 80, protocol of 'tcp', and cidr_blocks of ['0.0.0.0/0']. The second ingress rule has a description of 'add https', from_port of 22, to_port of 22, protocol of 'tcp', and cidr_blocks of ['0.0.0.0/0']. There is also an egress rule with from_port of 0, to_port of 0, protocol of '-1', and cidr_blocks of ['0.0.0.0/0'].

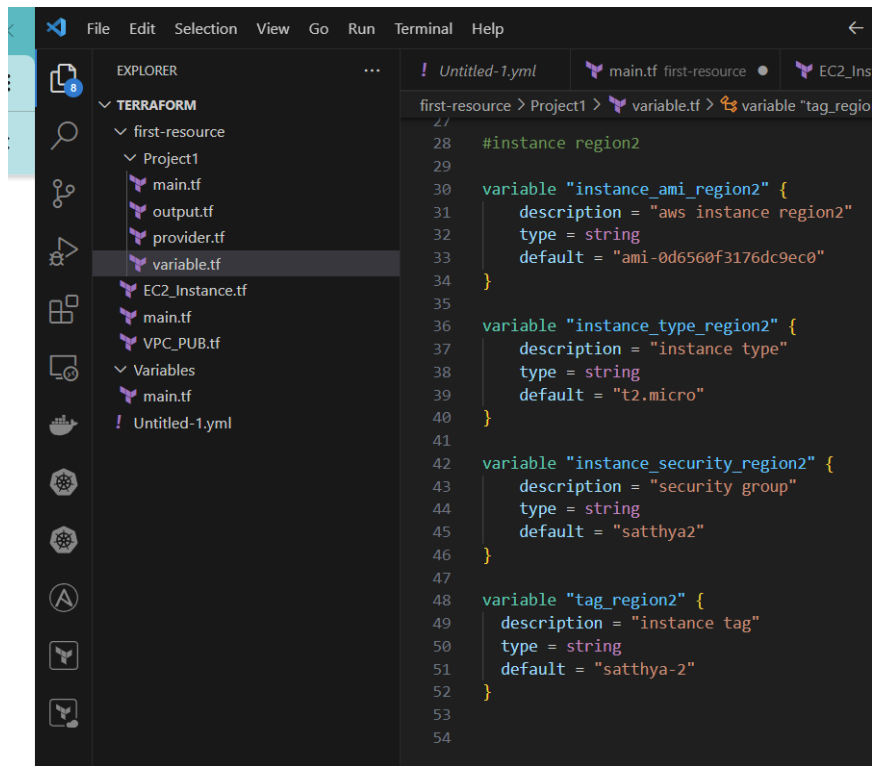
```
91
92 resource "aws_security_group" "region2" {
93     provider = aws.region2
94     ingress {
95         description = "add http"
96         from_port = 80
97         to_port = 80
98         protocol = "tcp"
99         cidr_blocks = [ "0.0.0.0/0" ]
100     }
101     ingress {
102         description = "add https"
103         from_port = 22
104         to_port = 22
105         protocol = "tcp"
106         cidr_blocks = [ "0.0.0.0/0" ]
107     }
108     egress {
109         from_port = 0
110         to_port = 0
111         protocol = "-1"
112         cidr_blocks = [ "0.0.0.0/0" ]
113     }
114 }
115
116
117
```

variable.tf



The screenshot shows the VS Code interface with the Terraform Explorer on the left. The Explorer shows a project structure under 'TERRAFORM' with folders 'first-resource' and 'Project1'. Under 'Project1', there are files 'main.tf', 'output.tf', 'provider.tf', 'variable.tf', 'EC2_Instance.tf', and 'VPC_PUB.tf'. The 'variable.tf' file is selected. The main editor shows the content of 'variable.tf', which defines four variables: 'instance_ami_region1', 'instance_type_region1', 'instance_security_region1', and 'tag_region1'. The 'instance_ami_region1' variable has a description of 'aws instance region1', type of 'string', and a default value of 'ami-0995922d49dc9a17d'. The 'instance_type_region1' variable has a description of 'instance type', type of 'string', and a default value of 't2.micro'. The 'instance_security_region1' variable has a description of 'security group', type of 'string', and a default value of 'satthya1'. The 'tag_region1' variable has a description of 'instance tag', type of 'string', and a default value of 'satthya-1'.

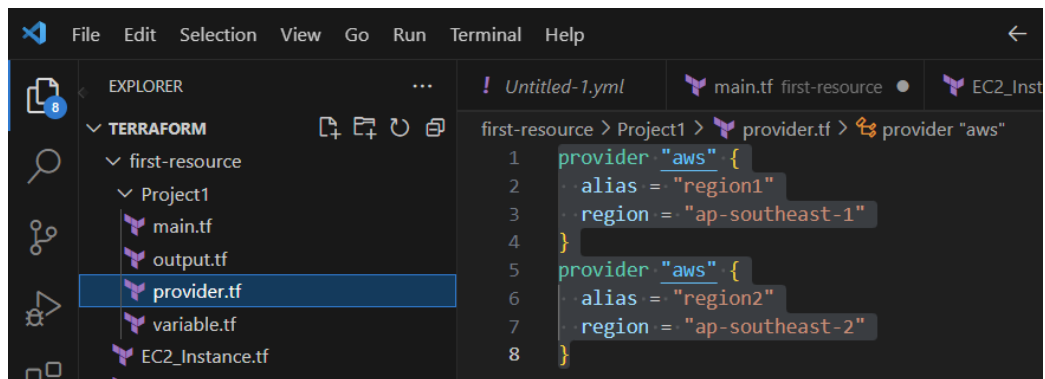
```
1
2 #instance region1
3
4 variable "instance_ami_region1" {
5     description = "aws instance region1"
6     type = string
7     default = "ami-0995922d49dc9a17d"
8 }
9
10 variable "instance_type_region1" {
11     description = "instance type"
12     type = string
13     default = "t2.micro"
14 }
15
16 variable "instance_security_region1" {
17     description = "security group"
18     type = string
19     default = "satthya1"
20 }
21
22 variable "tag_region1" {
23     description = "instance tag"
24     type = string
25     default = "satthya-1"
26 }
27
```



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The 'TERRAFORM' folder is expanded, showing a hierarchy: 'first-resource' > 'Project1' > 'main.tf', 'output.tf', 'provider.tf', and 'variable.tf'. The 'variable.tf' file is selected and its content is displayed in the main editor. The code defines several variables for region2, including instance_ami, instance_type, instance_security, and tag_region2.

```
27
28 #instance region2
29
30 variable "instance_ami_region2" {
31   description = "aws instance region2"
32   type = string
33   default = "ami-0d6560f3176dc9ec0"
34 }
35
36 variable "instance_type_region2" {
37   description = "instance type"
38   type = string
39   default = "t2.micro"
40 }
41
42 variable "instance_security_region2" {
43   description = "security group"
44   type = string
45   default = "satthya2"
46 }
47
48 variable "tag_region2" {
49   description = "instance tag"
50   type = string
51   default = "satthya-2"
52 }
53
54
```

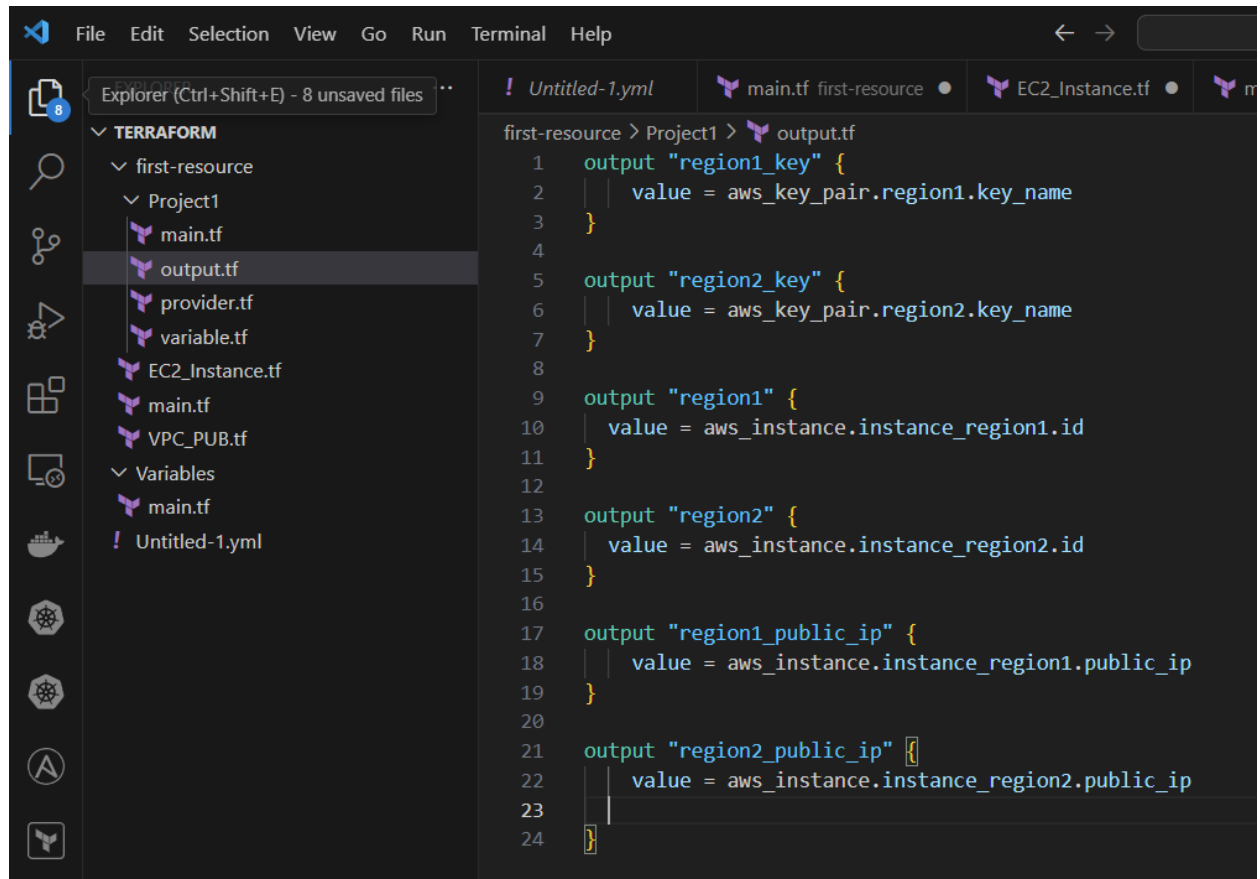
 provider.tf



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The 'TERRAFORM' folder is expanded, showing a hierarchy: 'first-resource' > 'Project1' > 'main.tf', 'output.tf', 'provider.tf', and 'variable.tf'. The 'provider.tf' file is selected and its content is displayed in the main editor. The code defines two AWS providers for different regions.

```
1 provider "aws" {
2   alias = "region1"
3   region = "ap-southeast-1"
4 }
5
6 provider "aws" {
7   alias = "region2"
8   region = "ap-southeast-2"
9 }
```

 output.tf



The screenshot shows the Visual Studio Code interface with a Terraform project. The Explorer sidebar on the left shows a directory structure under 'TERRAFORM' with folders 'first-resource' and 'Variables'. The 'first-resource' folder contains 'Project1', which includes 'main.tf', 'output.tf' (selected), 'provider.tf', and 'variable.tf'. Other files in the project include 'EC2_Instance.tf', 'main.tf', and 'VPC_PUB.tf'. The main editor window displays the content of 'output.tf' with the following HCL code:

```
1  output "region1_key" {
2    |  value = aws_key_pair.region1.key_name
3  }
4
5  output "region2_key" {
6    |  value = aws_key_pair.region2.key_name
7  }
8
9  output "region1" {
10   |  value = aws_instance.instance_region1.id
11 }
12
13 output "region2" {
14   |  value = aws_instance.instance_region2.id
15 }
16
17 output "region1_public_ip" {
18   |  value = aws_instance.instance_region1.public_ip
19 }
20
21 output "region2_public_ip" {
22   |  value = aws_instance.instance_region2.public_ip
23 }
24 }
```

Step4

- Run terraform configuration file.

```
ubuntu@ip-172-31-19-68: ~/guvi3
```

```
ubuntu@ip-172-31-19-68:~/guvi3$ ll
total 24
drwxrwxr-x  2 ubuntu ubuntu 4096 Jan  7 06:05 ./
drwxr-x--- 12 ubuntu ubuntu 4096 Jan  7 06:05 ../
-rw-rw-r--  1 ubuntu ubuntu 2703 Jan  7 06:04 main.tf
-rw-rw-r--  1 ubuntu ubuntu  442 Jan  7 06:04 output.tf
-rw-rw-r--  1 ubuntu ubuntu  134 Jan  7 06:04 provider.tf
-rw-rw-r--  1 ubuntu ubuntu  981 Jan  7 06:05 variable.tf
ubuntu@ip-172-31-19-68:~/guvi3$
```

- terraform init

```
ubuntu@ip-172-31-19-68: ~/guvi3
ubuntu@ip-172-31-19-68:~/guvi3$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.82.2...
- Installed hashicorp/aws v5.82.2 (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.
ubuntu@ip-172-31-19-68:~/guvi3$
```

- terraform fmt

```
ubuntu@ip-172-31-19-68: ~/guvi3
ubuntu@ip-172-31-19-68:~/guvi3$ terraform fmt
main.tf
output.tf
provider.tf
variable.tf
ubuntu@ip-172-31-19-68:~/guvi3$
```


- terraform plan

ubuntu@ip-172-31-19-68: ~/guvi3

```
+ tags_all          = (known after apply)
+ vpc_id            = (known after apply)
}

# aws_security_group.region2 will be created
+ resource "aws_security_group" "region2" {
+   arn              = (known after apply)
+   description      = "Managed by Terraform"
+   egress           = [
+     {
+       + cidr_blocks = [
+         + "0.0.0.0/0",
+       ]
+       + from_port   = 0
+       + ipv6_cidr_blocks = []
+       + prefix_list_ids = []
+       + protocol     = "-1"
+       + security_groups = []
+       + self         = false
+       + to_port      = 0
+       # (1 unchanged attribute hidden)
+     },
+   ]
+   id              = (known after apply)
+   ingress         = [
+     {
+       + cidr_blocks = [
+         + "0.0.0.0/0",
+       ]
+       + from_port   = 22
+       + ipv6_cidr_blocks = []
+       + prefix_list_ids = []
+       + protocol     = "tcp"
+       + security_groups = []
+       + self         = false
+       + to_port      = 22
+       # (1 unchanged attribute hidden)
+     },
+     {
+       + cidr_blocks = [
+         + "0.0.0.0/0",
+       ]
+       + description = "add http"
+       + from_port   = 80
+       + ipv6_cidr_blocks = []
+       + prefix_list_ids = []
+       + protocol     = "tcp"
+       + security_groups = []
+       + self         = false
+       + to_port      = 80
+     },
+   ]
+   name            = (known after apply)
+   name_prefix     = (known after apply)
+   owner_id        = (known after apply)
+   revoke_rules_on_delete = false
+   tags_all        = (known after apply)
+   vpc_id          = (known after apply)
}
```

Plan: 6 to add, 0 to change, 0 to destroy.

Changes to Outputs:

```
+ region1          = (known after apply)
+ region1_key      = "key_region1"
+ region1_public_ip = (known after apply)
+ region2          = (known after apply)
+ region2_key      = "key_region2"
+ region2_public_ip = (known after apply)
```

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

ubuntu@ip-172-31-19-68: ~/guvi3\$

- terraform apply

ubuntu@ip-172-31-19-68: ~/guvi3

```
aws_instance.instance_region1 (remote-exec): Preparing : 1/1
aws_instance.instance_region1 (remote-exec): Running scriptlet: nginx-filesy 1/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [ ] 1/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [= ] 1/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [== ] 1/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [=== ] 1/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [==== ] 1/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [===== ] 1/7
aws_instance.instance_region1 (remote-exec): Installing : nginx-filesy 1/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [ ] 2/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [==== ] 2/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [===== ] 2/7
aws_instance.instance_region1 (remote-exec): Installing : nginx-mimety 2/7
aws_instance.instance_region1 (remote-exec): Installing : libun [ ] 3/7
aws_instance.instance_region1 (remote-exec): Installing : libun [= ] 3/7
aws_instance.instance_region1 (remote-exec): Installing : libun [== ] 3/7
aws_instance.instance_region1 (remote-exec): Installing : libun [==== ] 3/7
aws_instance.instance_region1 (remote-exec): Installing : libun [===== ] 3/7
aws_instance.instance_region1 (remote-exec): Installing : libunwind-1. 3/7
aws_instance.instance_region1 (remote-exec): Installing : gperf [ ] 4/7
aws_instance.instance_region1 (remote-exec): Installing : gperf [= ] 4/7
aws_instance.instance_region1 (remote-exec): Installing : gperf [== ] 4/7
aws_instance.instance_region1 (remote-exec): Installing : gperf [=== ] 4/7
aws_instance.instance_region1 (remote-exec): Installing : gperf [==== ] 4/7
aws_instance.instance_region1 (remote-exec): Installing : gperf [===== ] 4/7
aws_instance.instance_region1 (remote-exec): Installing : gperftools-1 4/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [ ] 5/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [= ] 5/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [== ] 5/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [=== ] 5/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [==== ] 5/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [===== ] 5/7
aws_instance.instance_region1 (remote-exec): Installing : nginx-core-1 5/7
aws_instance.instance_region1 (remote-exec): Installing : gener [ ] 6/7
aws_instance.instance_region1 (remote-exec): Installing : gener [==== ] 6/7
aws_instance.instance_region1 (remote-exec): Installing : gener [===== ] 6/7
aws_instance.instance_region1 (remote-exec): Installing : generic-logo 6/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [ ] 7/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [== ] 7/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [=== ] 7/7
aws_instance.instance_region1 (remote-exec): Installing : nginx [===== ] 7/7
aws_instance.instance_region1 (remote-exec): Installing : nginx-1:1.26 7/7
aws_instance.instance_region1 (remote-exec): Running scriptlet: nginx-1:1.26 7/7
aws_instance.instance_region1 (remote-exec): Verifying : generic-logo 1/7
aws_instance.instance_region1 (remote-exec): Verifying : gperftools-1 2/7
aws_instance.instance_region1 (remote-exec): Verifying : libunwind-1. 3/7
aws_instance.instance_region1 (remote-exec): Verifying : nginx-1:1.26 4/7
aws_instance.instance_region1 (remote-exec): Verifying : nginx-core-1 5/7
aws_instance.instance_region1 (remote-exec): Verifying : nginx-filesy 6/7
aws_instance.instance_region1 (remote-exec): Verifying : nginx-mimety 7/7

aws_instance.instance_region1 (remote-exec): Installed:
aws_instance.instance_region1 (remote-exec): generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
aws_instance.instance_region1 (remote-exec): gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64
aws_instance.instance_region1 (remote-exec): libunwind-1.4.0-5.amzn2023.0.2.x86_64
aws_instance.instance_region1 (remote-exec): nginx-1:1.26.2-1.amzn2023.0.1.x86_64
aws_instance.instance_region1 (remote-exec): nginx-core-1:1.26.2-1.amzn2023.0.1.x86_64
aws_instance.instance_region1 (remote-exec): nginx-filesystem-1:1.26.2-1.amzn2023.0.1.noarch
aws_instance.instance_region1 (remote-exec): nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch

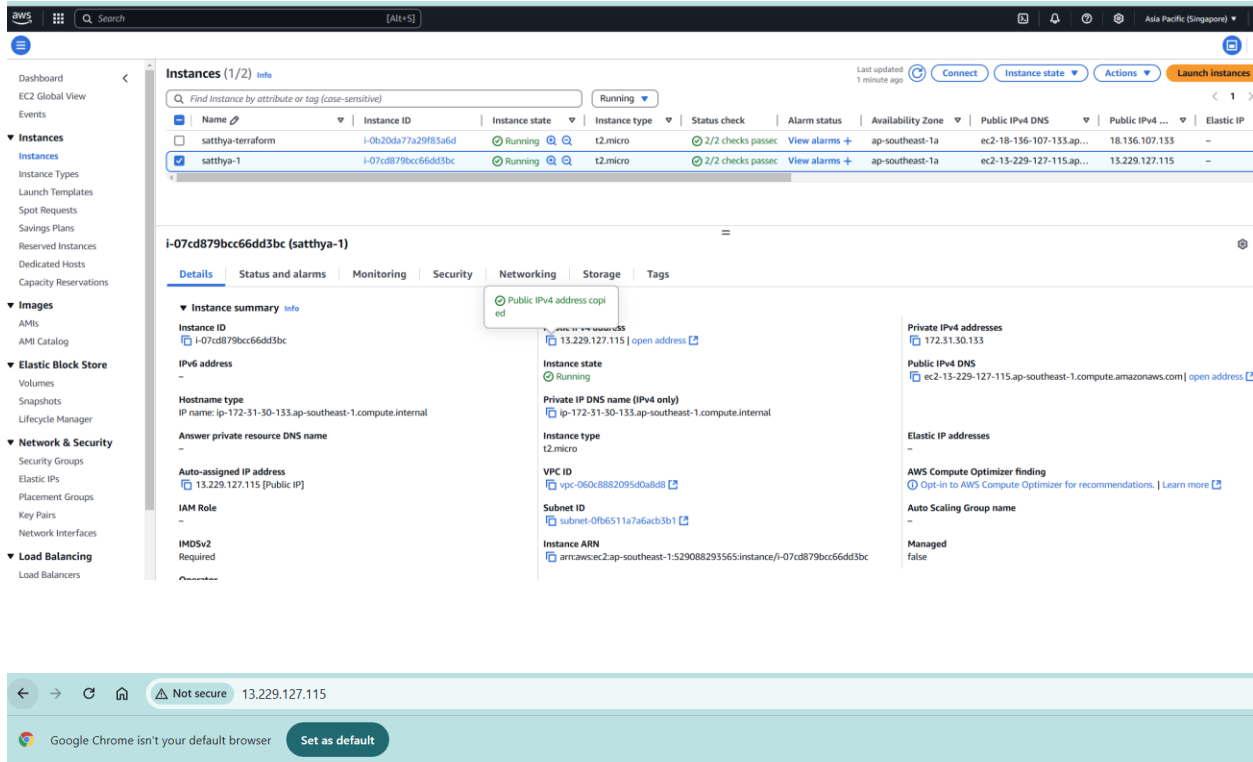
aws_instance.instance_region1 (remote-exec): Complete!
aws_instance.instance_region1 (remote-exec): Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service
aws_instance.instance_region1: Creation complete after 1m14s [id=i-07cd879bcc66dd3bc]

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.

Outputs:
region1 = "i-07cd879bcc66dd3bc"
region1_key = "key_region1"
region1_public_ip = "13.229.127.115"
region2 = "i-0f1596fc00e74494c"
region2_key = "key_region2"
region2_public_ip = "54.206.100.166"
ubuntu@ip-172-31-19-68:~/guvi3$
```

Result

Region1



The screenshot displays the AWS Management Console interface. On the left, the navigation menu includes options like Dashboard, EC2 Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces, Load Balancing, and Load Balancers. The main content area shows the 'Instances' page with a table of instances. The instance 'satthya-1' is selected, and its details are displayed. The instance is running and has a public IP address of 13.229.127.115. The browser address bar shows the URL '13.229.127.115'.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
satthya-terraform	i-0b20da77a29f85a6d	Running	t2.micro	2/2 checks passed	View alarms +	ap-southeast-1a	ec2-18-136-107-135.ap...	18.156.107.133	-
satthya-1	i-07cd879bcc66dd3bc	Running	t2.micro	2/2 checks passed	View alarms +	ap-southeast-1a	ec2-13-229-127-115.ap...	13.229.127.115	-

i-07cd879bcc66dd3bc (satthya-1)

Instance summary

Instance ID: i-07cd879bcc66dd3bc

IP v6 address: -

Hostname type: IP name: ip-172-31-30-133.ap-southeast-1.compute.internal

Answer private resource DNS name: -

Auto-assigned IP address: 13.229.127.115 [Public IP]

IAM Role: -

IMDSv2: Required

Instance state: Running

Private IP DNS name (IPv4 only): ip-172-31-30-133.ap-southeast-1.compute.internal

Instance type: t2.micro

VPC ID: vpc-060c8882095d0a8b0

Subnet ID: subnet-0fb6511a7af6acb3b1

Instance ARN: arn:aws:ec2:ap-southeast-1:529088293565:instance/i-07cd879bcc66dd3bc

Private IPv4 addresses: 172.31.30.133

Public IPv4 DNS: ec2-13-229-127-115.ap-southeast-1.compute.amazonaws.com

Elastic IP addresses: -

AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations. [Learn more]

Auto Scaling Group name: -

Managed: false

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.



Dashboard

EC2 Global View

Events

Instances

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Images

AMIs

AMI Catalog

Elastic Block Store

Volumes

Snapshots

Lifecycle Manager

Network & Security

Security Groups

Elastic IPs

Placement Groups

Key Pairs

Network Interfaces

Load Balancing

Load Balancers

Target Groups

Instances (1/1) info

Find instance by attribute or tag (case-sensitive)

Running

Last updated less than a minute ago

Connect

Instance state

Actions

Launch instances

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 I
sathya-2	i-0f1596fc00e74494c	Running	t2.micro	2/2 checks passed	View alarms	ap-southeast-2a	ec2-54-206-100-166.ap...	54.206.100.166	-	-

i-0f1596fc00e74494c (sathya-2)

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary info

Instance ID

i-0f1596fc00e74494c

IPv6 address

-

Hostname type

IP name: ip-172-31-1-177.ap-southeast-2.compute.internal

Answer private resource DNS name

-

Auto-assigned IP address

54.206.100.166 [Public IP]

IAM Role

-

IMDSv2

Required

Public IPv4 address

54.206.100.166 | open address

Instance state

Running

Private IP DNS name (IPv4 only)

ip-172-31-1-177.ap-southeast-2.compute.internal

Instance type

t2.micro

VPC ID

vpc-0a493270a94f62fa2

Subnet ID

subnet-031467c1b9ec29328

Instance ARN

arn:aws:ec2:ap-southeast-2:529088293565:instance/i-0f1596fc00e74494c

Private IPv4 addresses

172.31.1.177

Public IPv4 DNS

ec2-54-206-100-166.ap-southeast-2.compute.amazonaws.com | open address

Elastic IP addresses

-

AWS Compute Optimizer finding

Opt-in to AWS Compute Optimizer for recommendations. | Learn more

Auto Scaling Group name

-

Managed

false

Incidents | Sime Darby

Catalog Tasks | Sime Darby

9CTA580230087 | Catalog T...

Instances | EC2 | ap-southe...

Verint | Calendar

ChatGPT

Mail | sathya mahendran

Welcome to nginx!

→

Not secure

54.206.100.166

Google Chrome isn't your default browser

Set as default

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.