ASSIGNMENT 1

Write Terraform script to do perform following tasks on AWS cloud Platform

Name: Raman Boora

SAP:500109408

R.NO: R2142221160

Batch:DevOps(HB1)

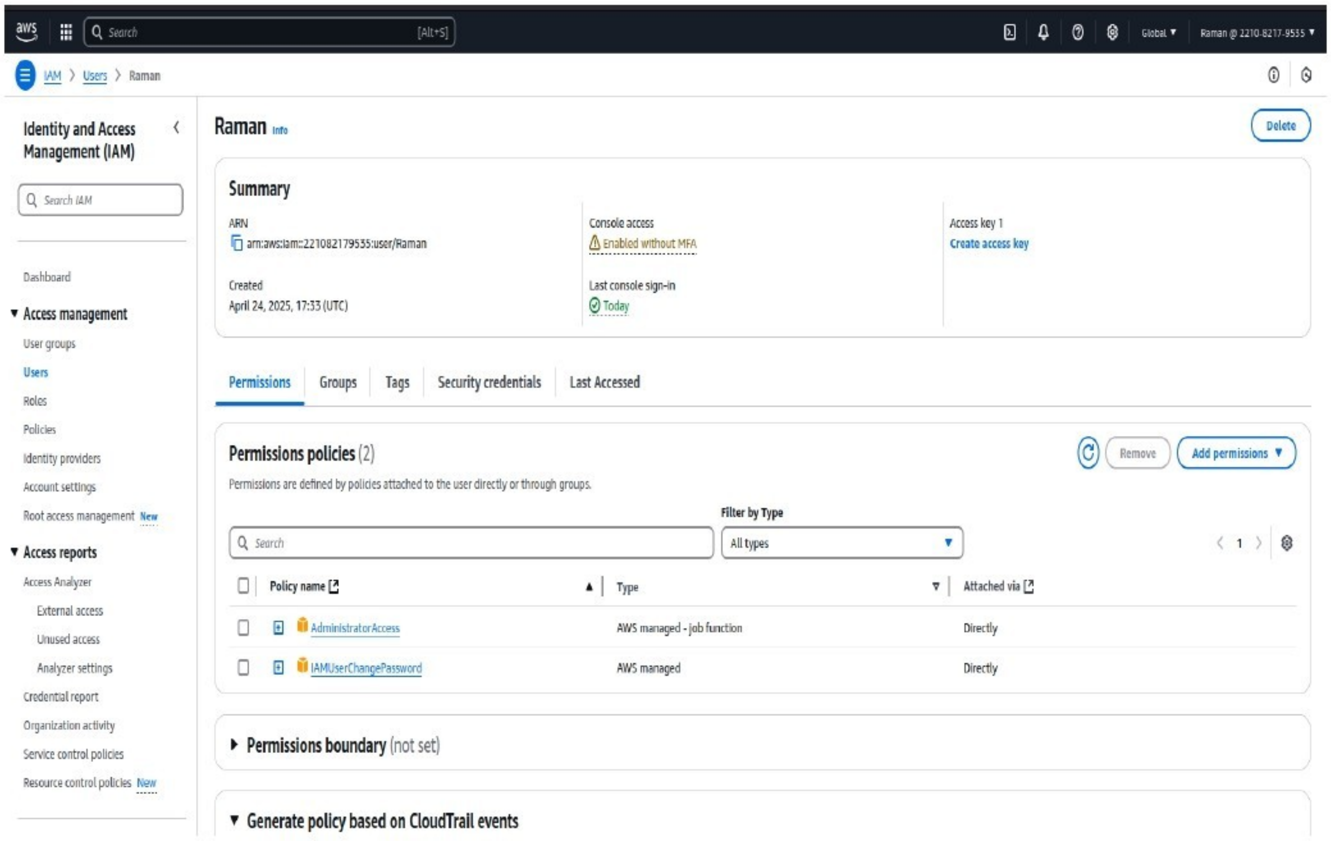
In this assignment, terraform — an open-source Infrastructure as Code (IaC) tool — is used to automate the provisioning of cloud resources on Amazon Web Services (AWS). The goal is to write a reusable, version-controlled script that sets up a basic cloud infrastructure in a matter of seconds.

The tasks covered in this assignment include:

- Creating two EC2 instances of type t2.micro
- Provisioning a Customer Gateway and a VPN Gateway to simulate a VPN setup
- Creating an S3 bucket for cloud-based storage

1. Creating IAM User

An IAM user named Raman was created in the AWS Console with programmatic access. This user will be used to authenticate Terraform with AWS for infrastructure provisionin



Raman Info

Summary

ARN
arn:aws:iam::221082179535:user/Raman

Created
April 24, 2025, 17:33 (UTC)

Console access
Enabled without MFA

Last console sign-in
Today

Access key 1
Create access key

Permissions policies (2)

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

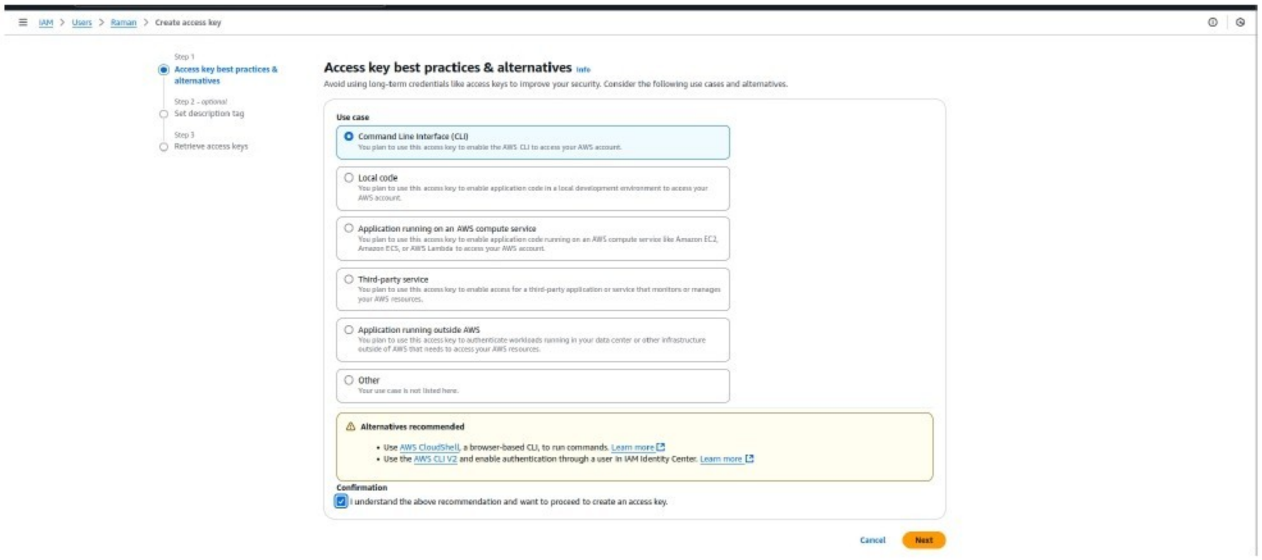
Policy name	Type	Attached via
AdministratorAccess	AWS managed - job function	Directly
IAMUserChangePassword	AWS managed	Directly

Permissions boundary (not set)

Generate policy based on CloudTrail events

2. Creating Access Key for IAM User

An Access Key and Secret Access Key were generated for the IAM user. These credentials were configured using the aws configure command to securely connect Terraform with the AWS account.


IAM > Users > Raman > Create access key
Step 1
Access key best practices & alternatives
Step 2 - optional
Set description tag
Step 3
Retrieve access keys

Access key best practices & alternatives Info

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

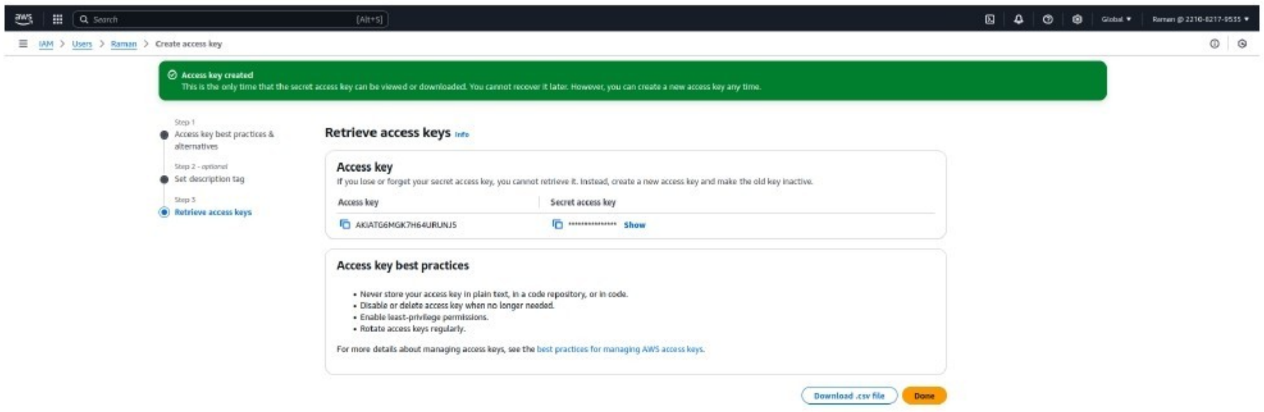
- [✓ Command Line Interface (CLI)] You plan to use this access key to enable the AWS CLI to access your AWS account.
- Local code: You plan to use this access key to enable application code in a local development environment to access your AWS account.
- Application running on an AWS compute service: You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.
- Third-party service: You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.
- Application running outside AWS: You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.
- Other: Your use case is not listed here.

Alternatives recommended

- Use AWS CloudShell, a browser-based CLI, to run commands. Learn more
- Use the AWS CLI V2 and enable authentication through a user in IAM Identity Center. Learn more

Confirmation

✓ I understand the above recommendation and want to proceed to create an 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.

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	Secret access key
AKIATG6MGK7H64URUNJ5	*************** Show

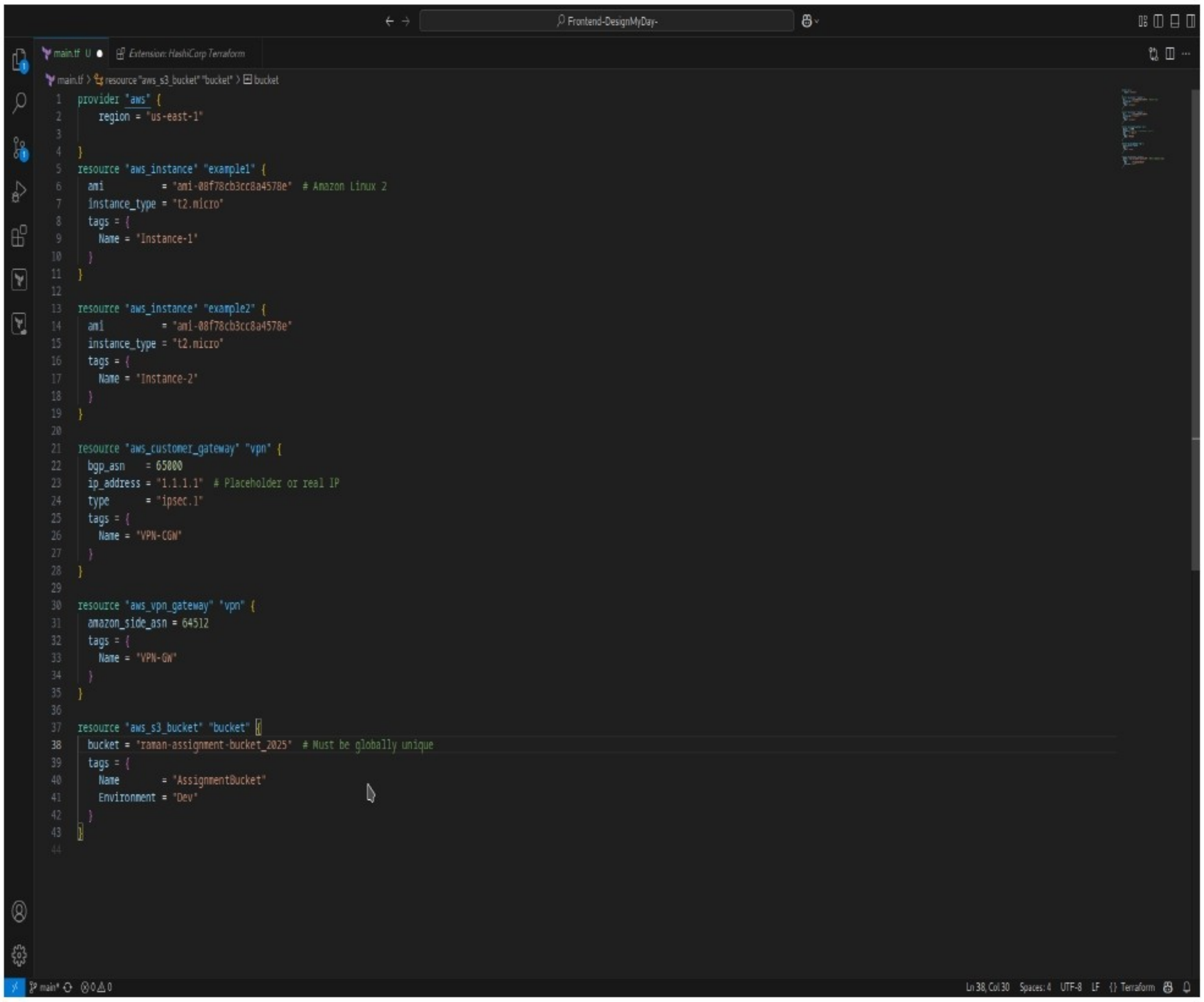
Access key best practices

- 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.

3. Writing Terraform Code in VS Code

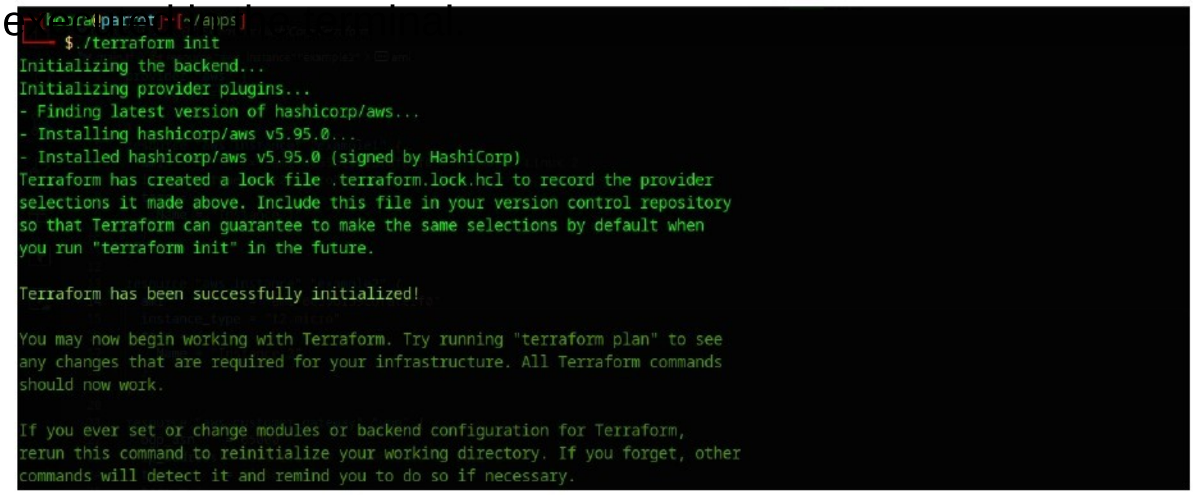
The Terraform configuration was written in VS Code. It include resources for EC2 instances, a VPN setup, and an S3 bucket —
all defined as Infrastructure as Code in a single main.tf file.


Frontend-DesignMyDay-
main.tf U
Extension: HashiCorp Terraform
main.tf > resource "aws_s3_bucket" "bucket" > bucket
main*
Ln 38, Col 30 Spaces: 4 UTF-8 LF {} Terraform

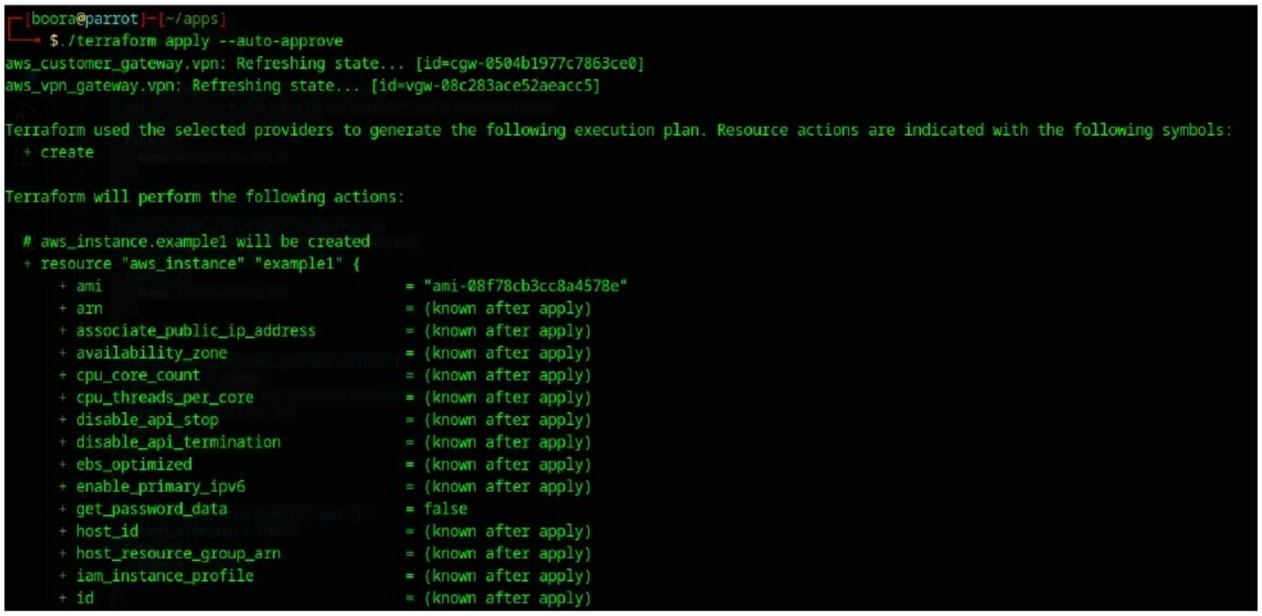
```
provider "aws" {
    region = "us-east-1"

}
resource "aws_instance" "example1" {
  ami           = "ami-08f78cb3cc8a4578e"  # Amazon Linux 2
  instance_type = "t2.micro"
  tags = {
    Name = "Instance-1"
  }
}

resource "aws_instance" "example2" {
  ami           = "ami-08f78cb3cc8a4578e"
  instance_type = "t2.micro"
  tags = {
    Name = "Instance-2"
  }
}

resource "aws_customer_gateway" "vpn" {
  bgp_asn    = 65000
  ip_address = "1.1.1.1"  # Placeholder or real IP
  type       = "ipsec.1"
  tags = {
    Name = "VPN-CGW"
  }
}

resource "aws_vpn_gateway" "vpn" {
  amazon_side_asn = 64512
  tags = {
    Name = "VPN-GW"
  }
}

resource "aws_s3_bucket" "bucket" {
  bucket = "raman-assignment-bucket_2025"  # Must be globally unique
  tags = {
    Name        = "AssignmentBucket"
    Environment = "Dev"
  }
}
```

4. Running terraform init, plan, and apply

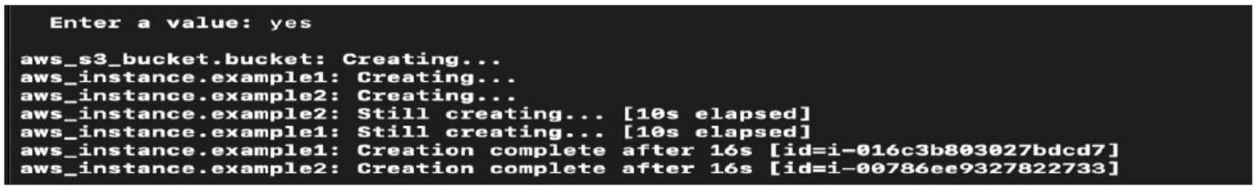
Terraform CLI commands were used to initialize the working directory, preview the execution plan, and apply the infrastructure changes. This screenshot shows the commands being executed in the terminal.



```
┌─[boora@parrot]─[~/apps]
└──╼ $./terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.95.0...
- Installed hashicorp/aws v5.95.0 (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.
```



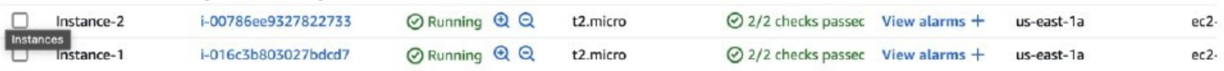
```
┌─[boora@parrot]─[~/apps]
└──╼ $./terraform apply --auto-approve
aws_customer_gateway.vpn: Refreshing state... [id=cgw-0504b1977c7863ce0]
aws_vpn_gateway.vpn: Refreshing state... [id=vgw-08c283ace52aeacc5]

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.example1 will be created
  + resource "aws_instance" "example1" {
      + ami                                  = "ami-08f78cb3cc8a4578e"
      + 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)
      + enable_primary_ipv6                  = (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)
```



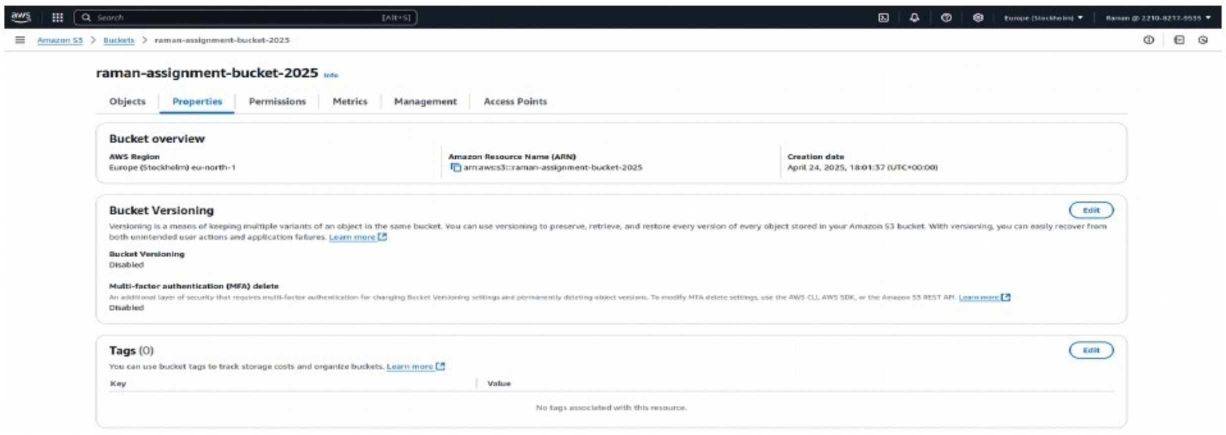
```
  Enter a value: yes

aws_s3_bucket.bucket: Creating...
aws_instance.example1: Creating...
aws_instance.example2: Creating...
aws_instance.example2: Still creating... [10s elapsed]
aws_instance.example1: Still creating... [10s elapsed]
aws_instance.example1: Creation complete after 16s [id=i-016c3b803027bdcd7]
aws_instance.example2: Creation complete after 16s [id=i-00786ee9327822733]
```

5. AWS Console Showing Resources Created

This screenshot captures the AWS Console showing all the resources created — including EC2 instances, VPN components, and the S3 bucket — confirming successful provisioning using Terraform.



Instance-2	i-00786ee9327822733	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-
Instance-1	i-016c3b803027bdcd7	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-



raman-assignment-bucket-2025

Bucket overview

AWS Region	Amazon Resource Name (ARN)	Creation date
Europe (Stockholm) eu-north-1	arn:aws:s3:::raman-assignment-bucket-2025	April 24, 2025, 18:01:37 (UTC+00:00)

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

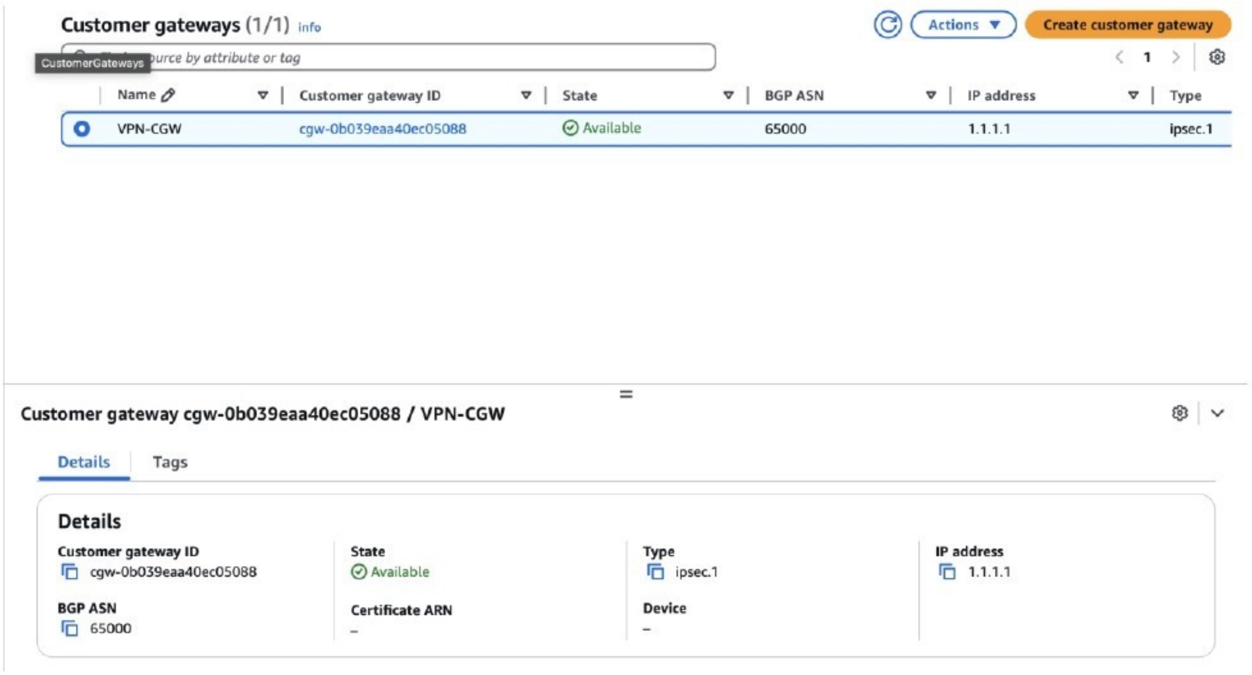
Bucket Versioning
Disabled

Multi-factor authentication (MFA) delete
An additional layer of security that requires multi-factor authentication for changing Bucket Versioning settings and permanently deleting object versions. To modify MFA delete settings, use the AWS CLI, AWS SDK, or the Amazon S3 REST API. Learn more
Disabled

Tags (0)

You can use bucket tags to track storage costs and organize buckets. Learn more

Key	Value
No tags associated with this resource.	



Customer gateways (1/1) info

Name	Customer gateway ID	State	BGP ASN	IP address	Type
VPN-CGW	cgw-0b039eaa40ec05088	Available	65000	1.1.1.1	ipsec.1

Customer gateway cgw-0b039eaa40ec05088 / VPN-CGW

Details

Customer gateway ID	State	Type	IP address
cgw-0b039eaa40ec05088	Available	ipsec.1	1.1.1.1
BGP ASN	**Certificate ARN**	**Device**	
65000	–	–	