Name: Akanksha Shinde   Class: D15C   Roll No: 53

13. Basic Infrastructure Management with Terraform

● Concepts Used: Terraform, AWS EC2, and S3.
● Problem Statement: "Use Terraform to create an EC2 instance and an S3 bucket. Store
the EC2 instance's IP address in the S3 bucket."
● Tasks:
○ Write a simple Terraform script to provision an EC2 instance and an S3 bucket.
○ Use Terraform outputs to extract the EC2 instance's IP address.

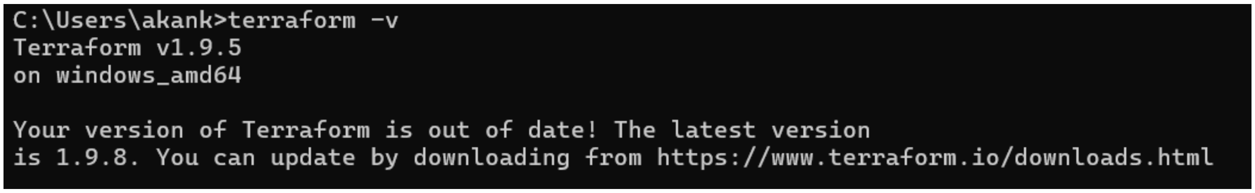
Introduction -

Case Study Overview:
This case study focuses on using Terraform to manage cloud infrastructure. The task involves creating an AWS EC2 instance and an S3 bucket using Terraform and storing the EC2 instance's IP address in the S3 bucket.

Key Feature and Application:
The key feature of this case study is infrastructure-as-code (IaC) using Terraform. Terraform allows automation of cloud infrastructure management, making tasks like provisioning an EC2 instance and managing S3 buckets easier, more efficient, and repeatable.

Step-by-Step Explanation -

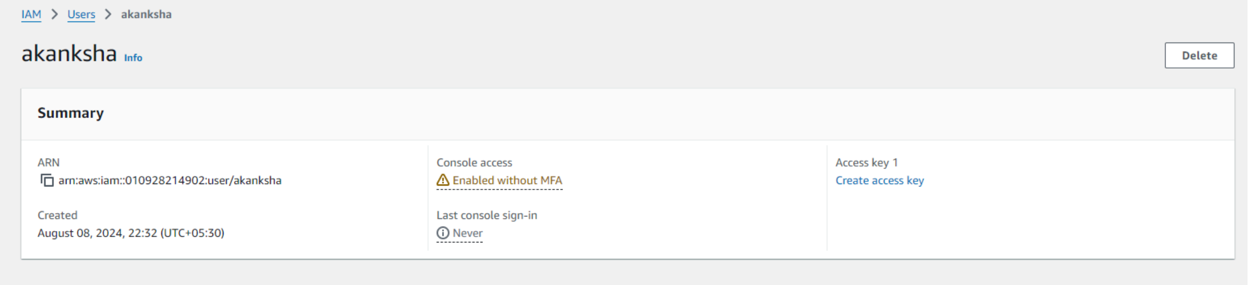
Step 1: Install Terraform and then check in the terminal.



```
C:\Users\akank>terraform -v
Terraform v1.9.5
on windows_amd64

Your version of Terraform is out of date! The latest version
is 1.9.8. You can update by downloading from https://www.terraform.io/downloads.html
```

Step 2: Set up AWS Credentials.

We will need an IAM user so preferably a personal AWS account is needed.
Get the Access Key ID and a Secret Access Key.



akanksha Info

Summary

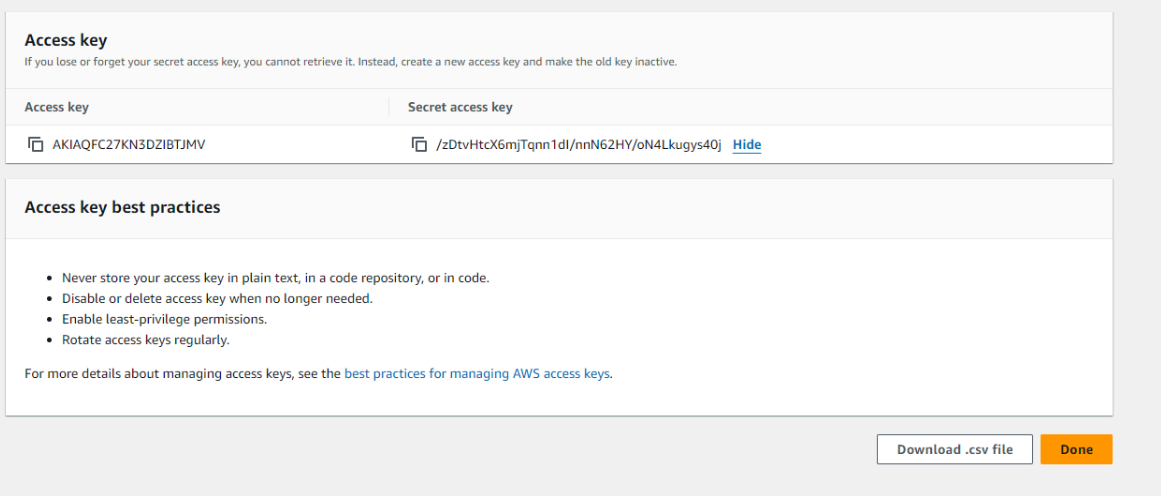
ARN
arn:aws:iam::010928214902:user/akanksha

Created
August 08, 2024, 22:32 (UTC+05:30)

Console access
⚠ Enabled without MFA

Last console sign-in
ⓘ Never

Access key 1
Create access key



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
AKIAQFC27KN3DZIBTJMV	/zDtvHtcX6mjTqnn1dl/nnN62HY/oN4Lkugys40j Hide

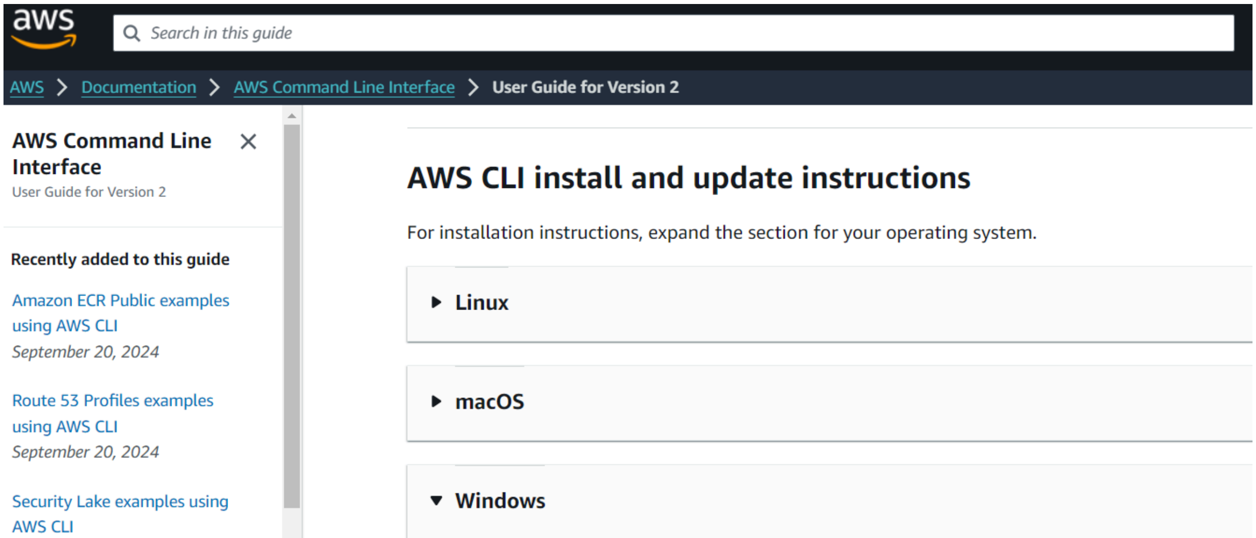
Access key best practices

- 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.

You will need AWS CLI to add these credentials and access the aws services from the local machine command prompt so install AWS CLI.

https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html



aws
Search in this guide
AWS > Documentation > AWS Command Line Interface > User Guide for Version 2

AWS Command Line Interface
User Guide for Version 2

Recently added to this guide

Amazon ECR Public examples using AWS CLI
September 20, 2024

Route 53 Profiles examples using AWS CLI
September 20, 2024

Security Lake examples using AWS CLI

AWS CLI install and update instructions

For installation instructions, expand the section for your operating system.

▸ Linux

▸ macOS

▾ Windows

Step 3: Write the Terraform Script

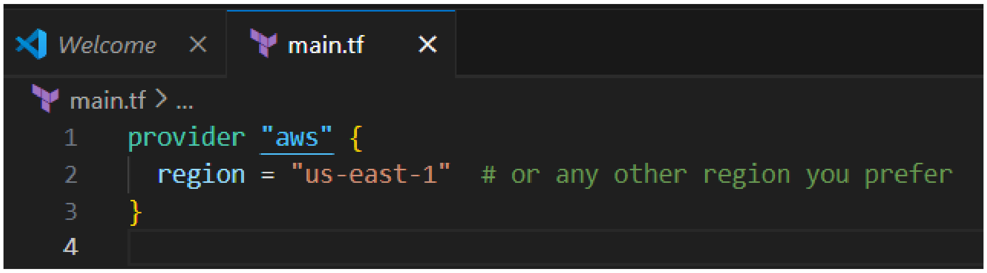
Make the folder and inside it make the main.tf file and write the code in it using IDE I used VScode.

Give AWS provider means that it will tell terraform that you are using AWS cloud provider out of many such existing like GCP, Alibaba etc.

Script:

Set region

```
provider "aws" {
  region = "us-east-1"  # or any other region you prefer
}
```



```
provider "aws" {
  region = "us-east-1"  # or any other region you prefer
}
```

Task 1: creating EC2 instance

```
resource "aws_instance" "my_ec2" {
  ami           = "ami-0c55b159cbfafe1f0"  # Amazon Linux 2 AMI for us-east-1
  instance_type = "t2.micro"               # Free-tier eligible instance type
}
```

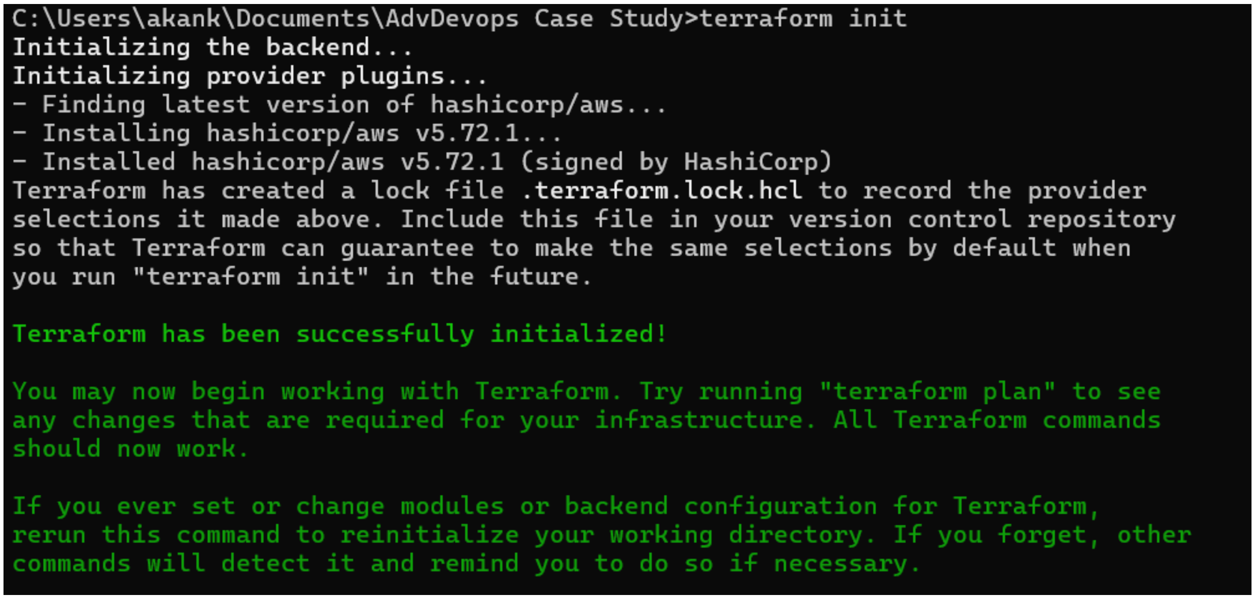
Task 2: Creating s3 bucket

```
# Create S3 bucket
resource "aws_s3_bucket" "my_bucket" {
  bucket = "my-terraform-bucket-akanksha-shinde"
}
```

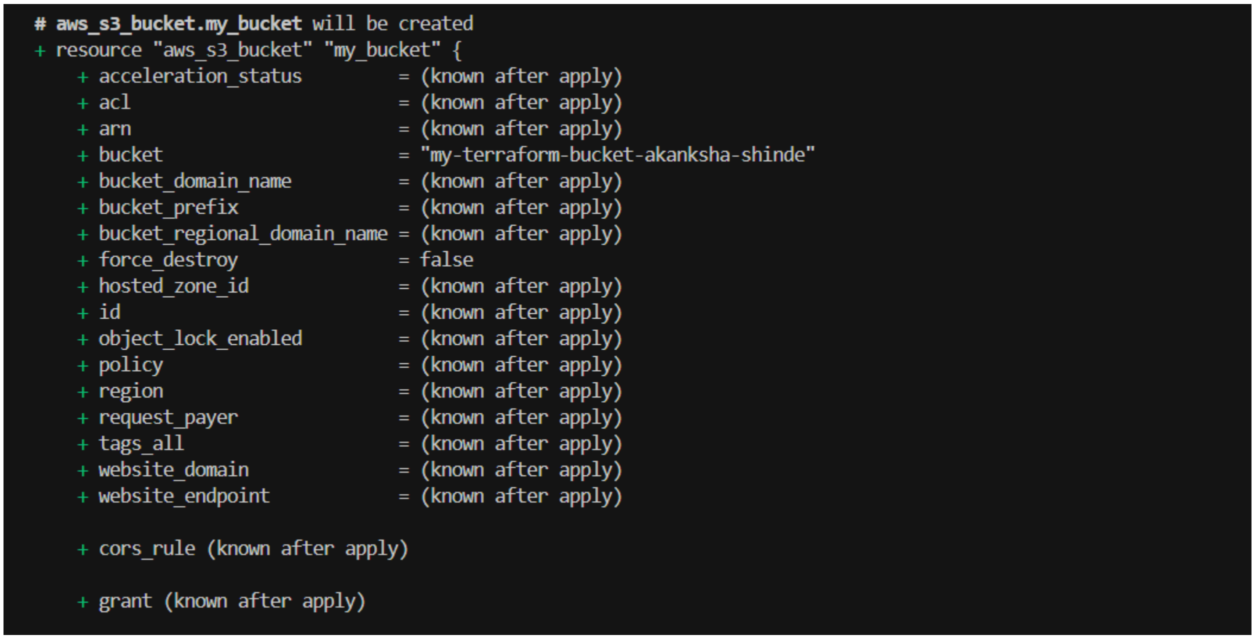
For output the EC2 instance's IP address

```
output "instance_ip" {
  value = aws_instance.my_ec2.public_ip
}
```

Step 4: Initialize and Apply Terraform
Open a terminal in the folder where your main.tf file is located and run terraform init.



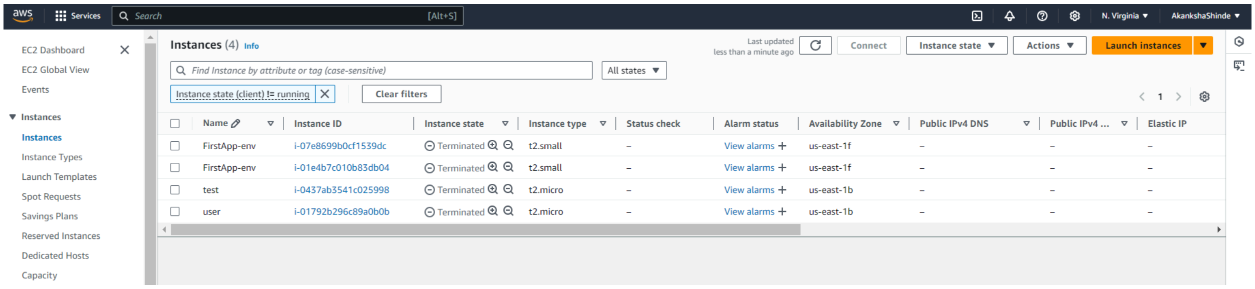
```
C:\Users\akank\Documents\AdvDevops Case Study>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.72.1...
- Installed hashicorp/aws v5.72.1 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl 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.
```



```
  # aws_s3_bucket.my_bucket will be created
  + resource "aws_s3_bucket" "my_bucket" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "my-terraform-bucket-akanksha-shinde"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags_all                    = (known after apply)
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + cors_rule (known after apply)

      + grant (known after apply)
```

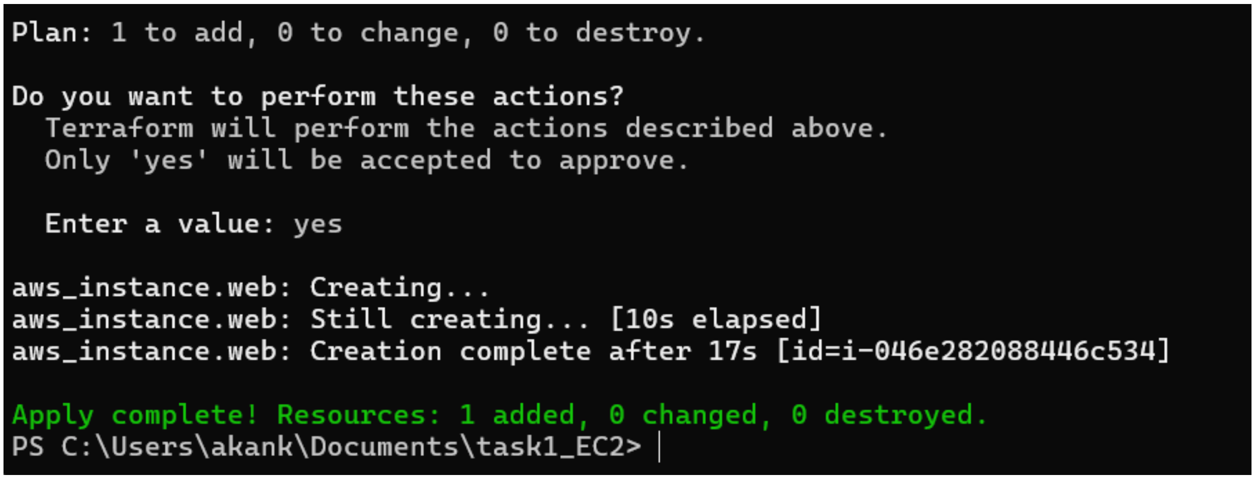
Task1: Create EC2 instance using Terraform.

Before creating ec2 instance



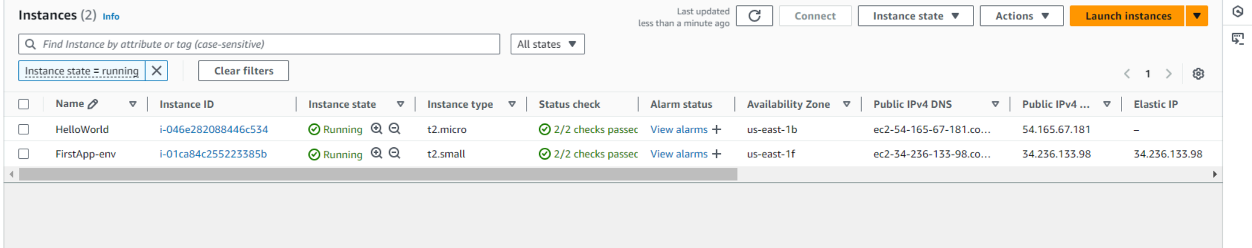
☐	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
☐	FirstApp-env	i-07e8699b0cf1539dc	Terminated	t2.small	–	View alarms +	us-east-1f	–	–	–
☐	FirstApp-env	i-01e4b7c010b83db04	Terminated	t2.small	–	View alarms +	us-east-1f	–	–	–
☐	test	i-0437ab3541c025998	Terminated	t2.micro	–	View alarms +	us-east-1b	–	–	–
☐	user	i-01792b296c89a0b0b	Terminated	t2.micro	–	View alarms +	us-east-1b	–	–	–

Now after running the scripts in terraform the ec2 instance has been successfully created.



```
Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.web: Creating...
aws_instance.web: Still creating... [10s elapsed]
aws_instance.web: Creation complete after 17s [id=i-046e282088446c534]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\akank\Documents\task1_EC2>
```

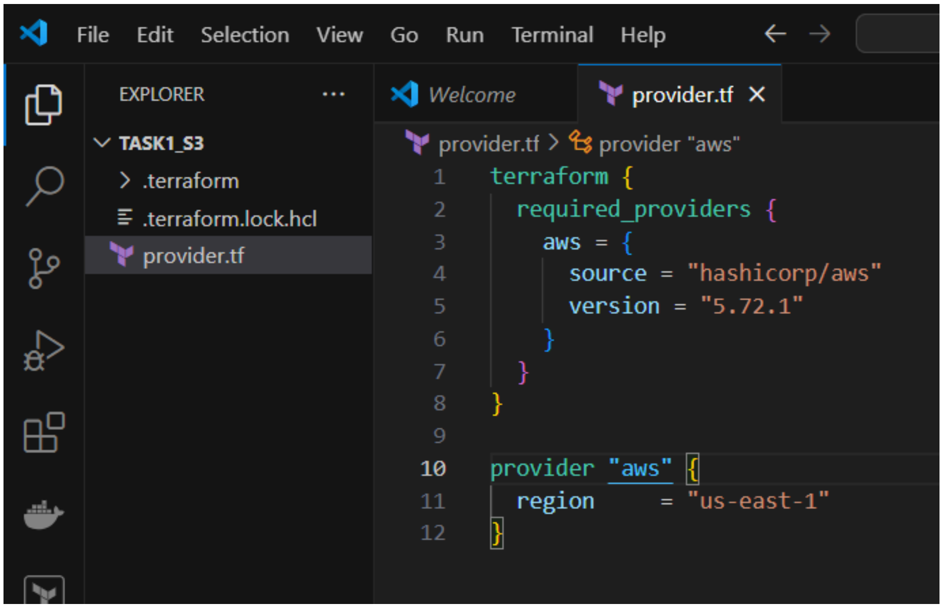
So ec2 instance named HelloWorld has been created once console page has been refreshed.



☐	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
☐	HelloWorld	i-046e282088446c534	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b	ec2-54-165-67-181.co...	54.165.67.181	–
☐	FirstApp-env	i-01ca84c255223385b	Running	t2.small	2/2 checks passec	View alarms +	us-east-1f	ec2-34-236-133-98.co...	34.236.133.98	34.236.133.98

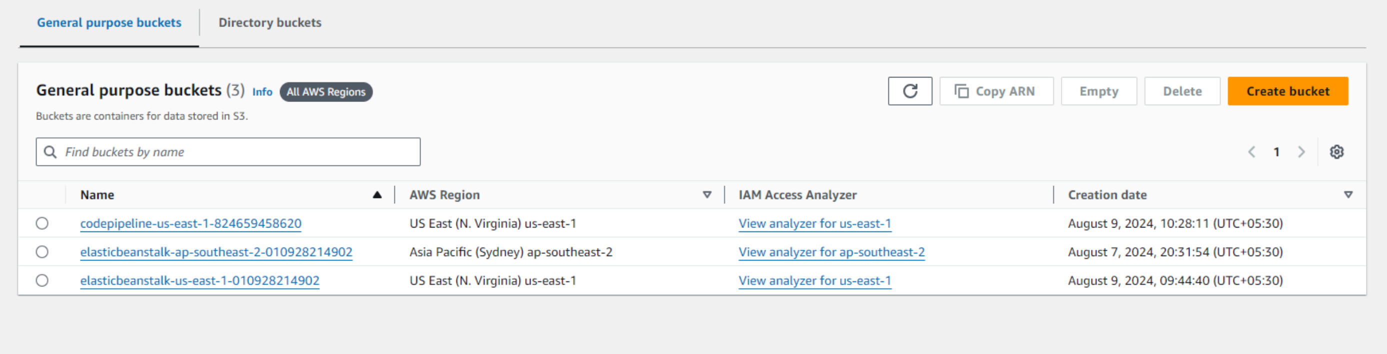
Task1 : Create S3 Bucket using Terraform.

Run the following script in IDE



```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.72.1"
    }
  }
}

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

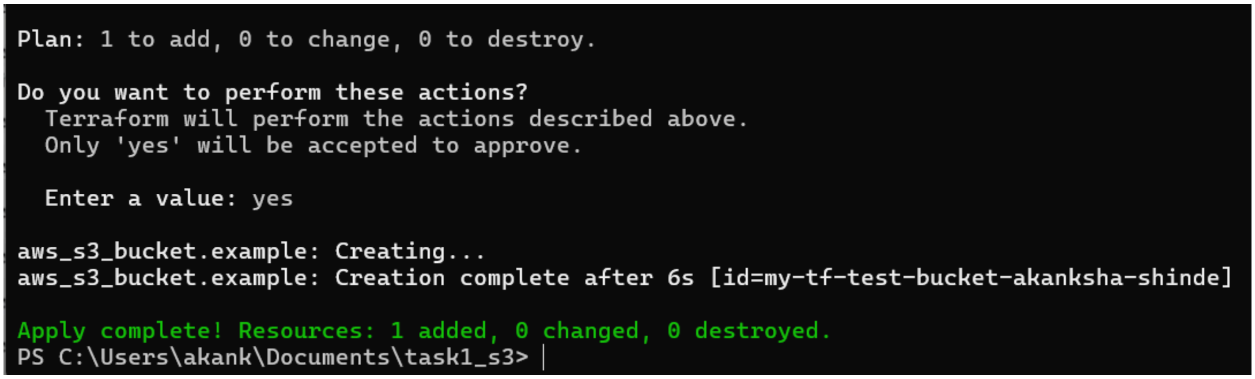
Before



General purpose buckets (3)

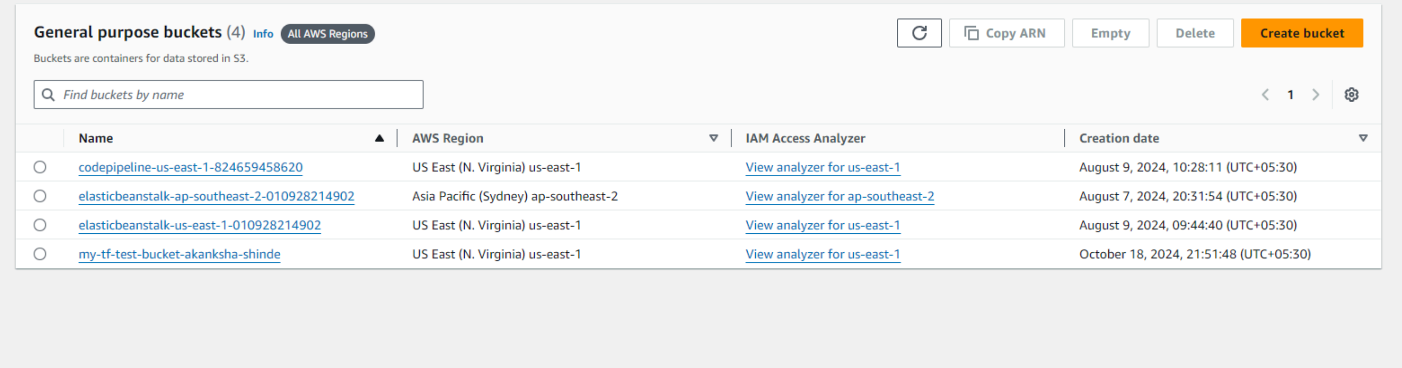
Buckets are containers for data stored in S3.

	Name	AWS Region	IAM Access Analyzer	Creation date
○	codepipeline-us-east-1-824659458620	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 9, 2024, 10:28:11 (UTC+05:30)
○	elasticbeanstalk-ap-southeast-2-010928214902	Asia Pacific (Sydney) ap-southeast-2	View analyzer for ap-southeast-2	August 7, 2024, 20:31:54 (UTC+05:30)
○	elasticbeanstalk-us-east-1-010928214902	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 9, 2024, 09:44:40 (UTC+05:30)



```
Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_s3_bucket.example: Creating...
aws_s3_bucket.example: Creation complete after 6s [id=my-tf-test-bucket-akanksha-shinde]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\akank\Documents\task1_s3>
```

After running script

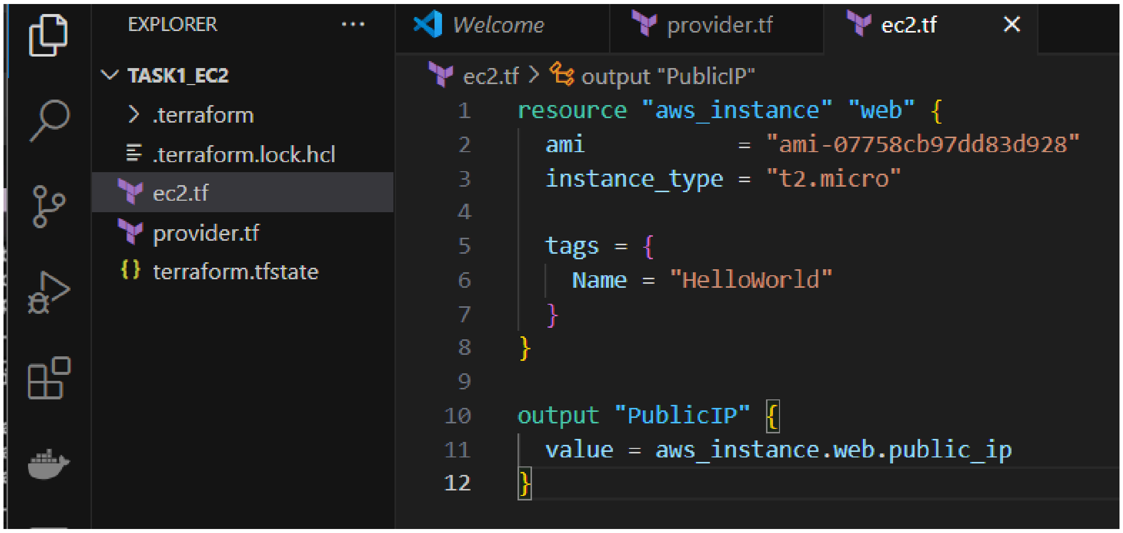


General purpose buckets (4) Info All AWS Regions
Buckets are containers for data stored in S3.
Copy ARN
Empty
Delete
Create bucket
Find buckets by name
1

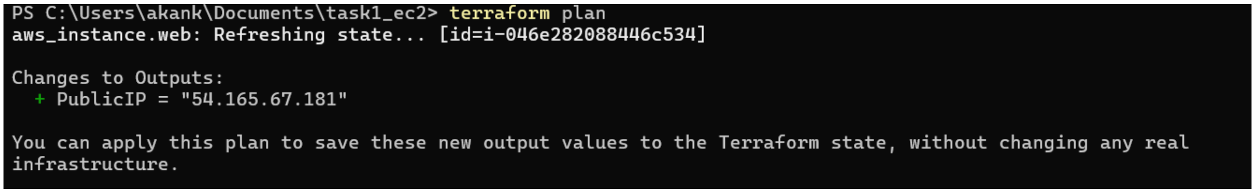
	Name	AWS Region	IAM Access Analyzer	Creation date
○	codepipeline-us-east-1-824659458620	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 9, 2024, 10:28:11 (UTC+05:30)
○	elasticbeanstalk-ap-southeast-2-010928214902	Asia Pacific (Sydney) ap-southeast-2	View analyzer for ap-southeast-2	August 7, 2024, 20:31:54 (UTC+05:30)
○	elasticbeanstalk-us-east-1-010928214902	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 9, 2024, 09:44:40 (UTC+05:30)
○	my-tf-test-bucket-akanksha-shinde	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 18, 2024, 21:51:48 (UTC+05:30)

Task 2: Use Terraform outputs to extract the EC2 instance's IP address.

As I have created the ec2 instance above now in the same script add the below script to get the output of IP address of ec2 instance.

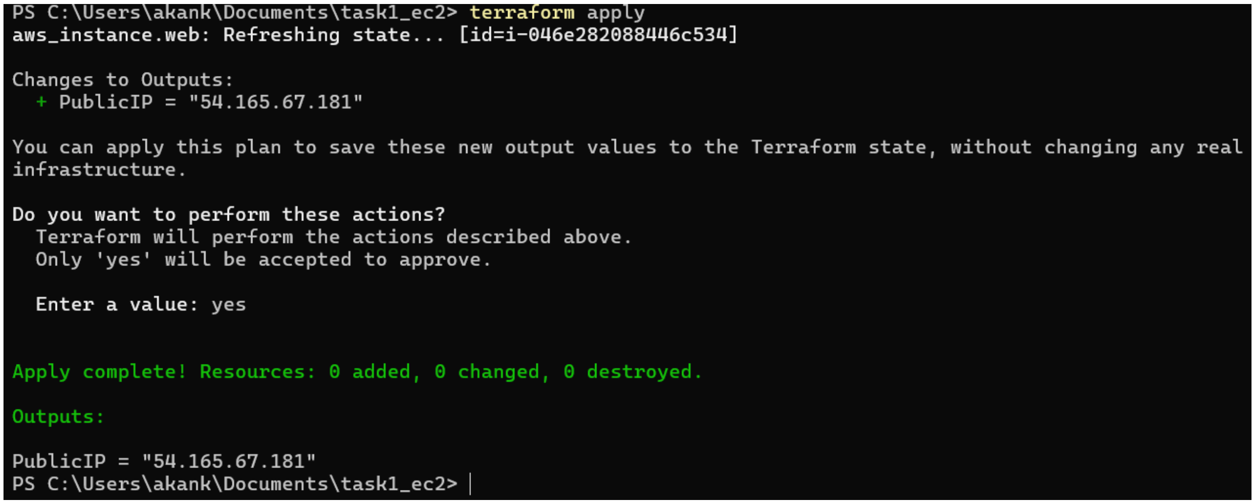


```
resource "aws_instance" "web" {
  ami           = "ami-07758cb97dd83d928"
  instance_type = "t2.micro"

  tags = {
    Name = "HelloWorld"
  }
}

output "PublicIP" {
  value = aws_instance.web.public_ip
}
```



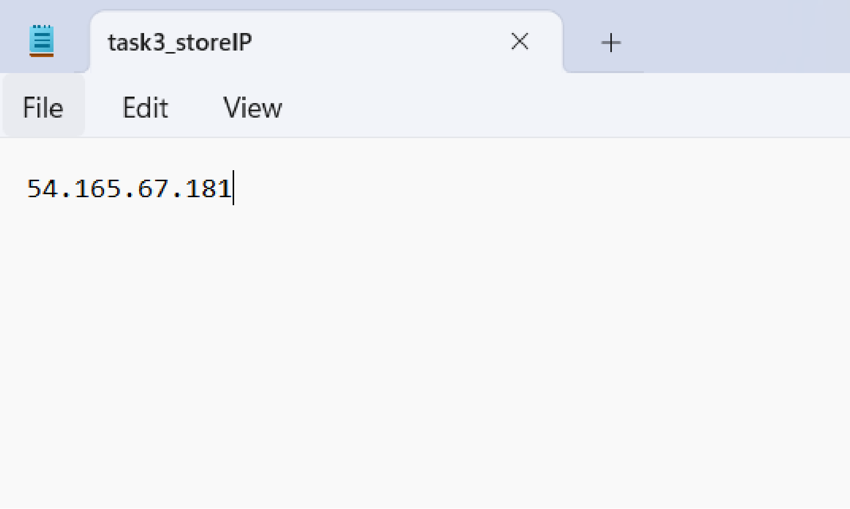
```
PS C:\Users\akank\Documents\task1_ec2> terraform plan
aws_instance.web: Refreshing state... [id=i-046e282088446c534]

Changes to Outputs:
  + PublicIP = "54.165.67.181"

You can apply this plan to save these new output values to the Terraform state, without changing any real
infrastructure.
```

Hence after terraform apply we get the Output IP address of ec2 instance that we created.



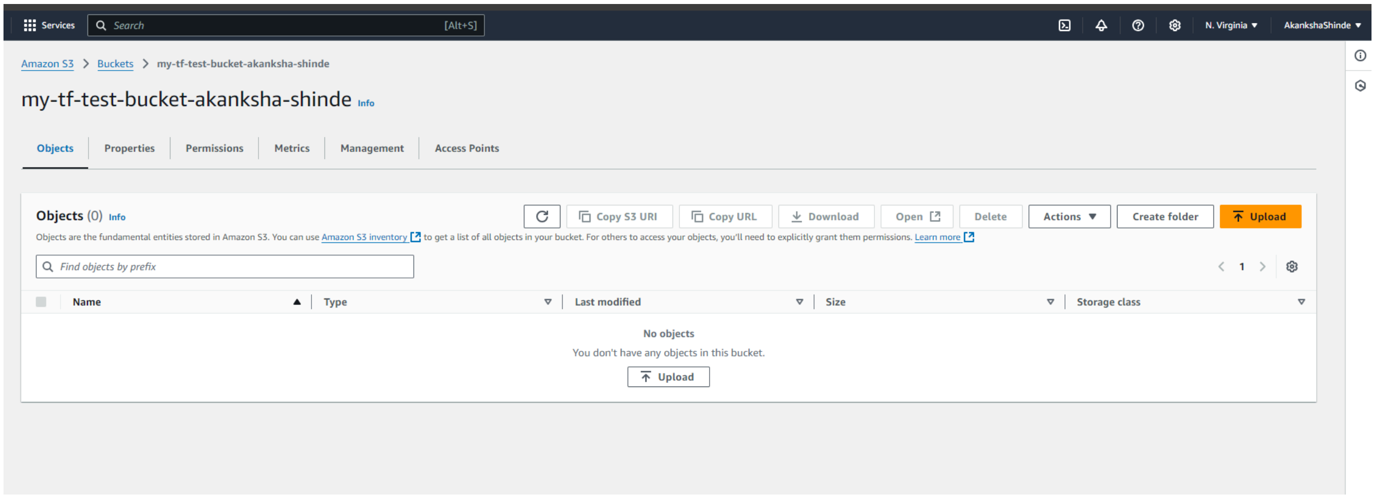
```
PS C:\Users\akank\Documents\task1_ec2> terraform apply
aws_instance.web: Refreshing state... [id=i-046e282088446c534]

Changes to Outputs:
  + PublicIP = "54.165.67.181"

You can apply this plan to save these new output values to the Terraform state, without changing any real
infrastructure.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes


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

Outputs:

PublicIP = "54.165.67.181"
PS C:\Users\akank\Documents\task1_ec2> 
```

Task 3: Store the IP address in a text file in the S3 bucket.



```
54.165.67.181
```

Go to S3 bucket.



my-tf-test-bucket-akanksha-shinde Info

Objects (0) Info

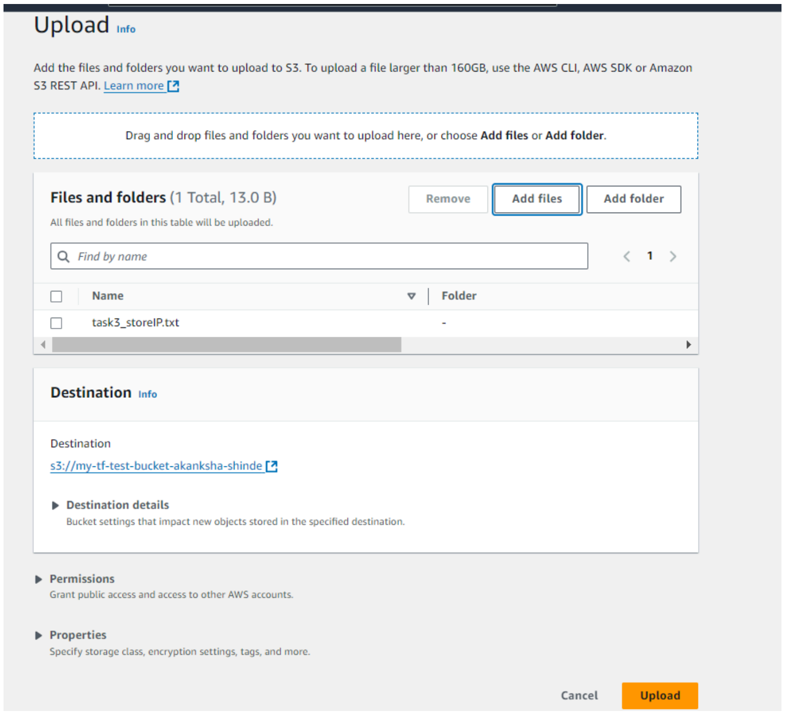
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name ▲	Type ▽	Last modified ▽	Size ▽	Storage class ▽

No objects

You don't have any objects in this bucket.

Upload the file



Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. Learn more

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

Files and folders (1 Total, 13.0 B)

All files and folders in this table will be uploaded.

	Name	Folder
□	task3_storeIP.txt	-

Destination Info

Destination

s3://my-tf-test-bucket-akanksha-shinde

Destination details

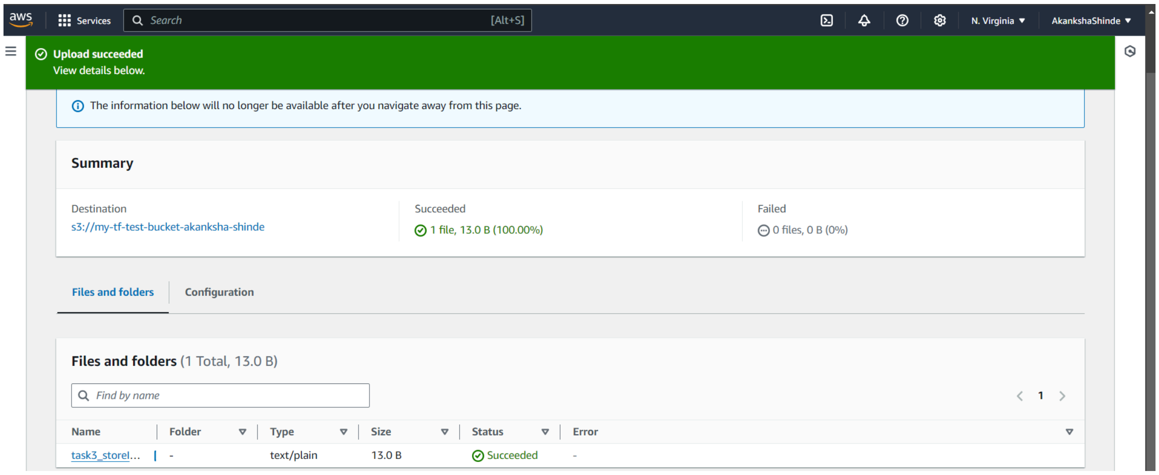
Bucket settings that impact new objects stored in the specified destination.

Permissions

Grant public access and access to other AWS accounts.

Properties

Specify storage class, encryption settings, tags, and more.



Upload succeeded

View details below.

The information below will no longer be available after you navigate away from this page.

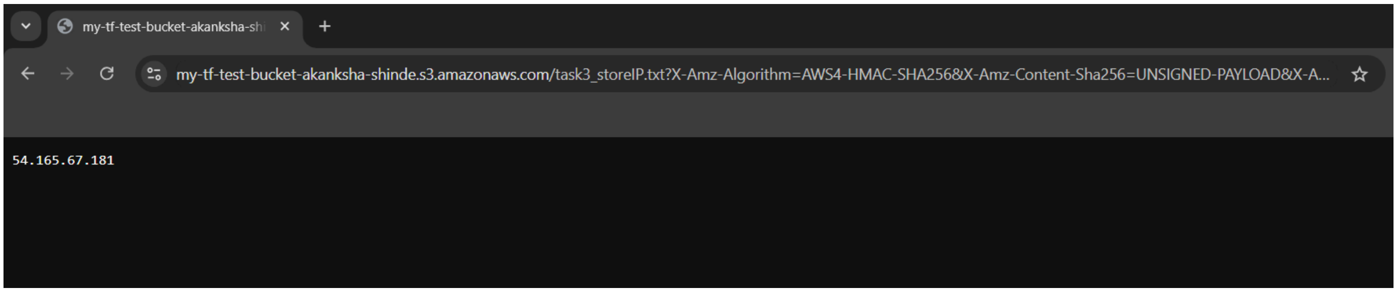
Summary

Destination	Succeeded	Failed
s3://my-tf-test-bucket-akanksha-shinde	✓ 1 file, 13.0 B (100.00%)	0 files, 0 B (0%)

Files and folders (1 Total, 13.0 B)

Name	Folder	Type	Size	Status	Error
task3_storeI...	-	text/plain	13.0 B	✓ Succeeded	-

Thus after pasting the url in the browser we will be seeing the EC2 instance’s IP address.



```
my-tf-test-bucket-akanksha-shinde.s3.amazonaws.com/task3_storeIP.txt?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Content-Sha256=UNSIGNED-PAYLOAD&X-A...
```

```
54.165.67.181
```

Conclusion

In this case study, I successfully used Terraform to automate the creation and management of basic cloud infrastructure on AWS. By usingTerraform’s infrastructure-as-code capabilities I was able to create an EC2 instance and an S3 bucket and automate the extraction and storage of the EC2 instance’s public IP address in the S3 bucket. This process demonstrated the power of automation in reducing manual work ensuring consistent and repeatable deployments. Through this practical experience I gained a deeper understanding of Terraform’s potential to cloud infrastructure management making it an essential tool.