Lesson 10 Demo 03

Configuring Terraform State Backend Storage

Objective: To configure and manage Terraform state using the AWS S3 backend for ensuring reliable state storage and management

Tools required: Visual Studio Code

Prerequisites: Ensure you have created and implemented the AWS access key and secret key before starting this demo. Refer to Lesson 08 Assisted Practice 02 for detailed steps.

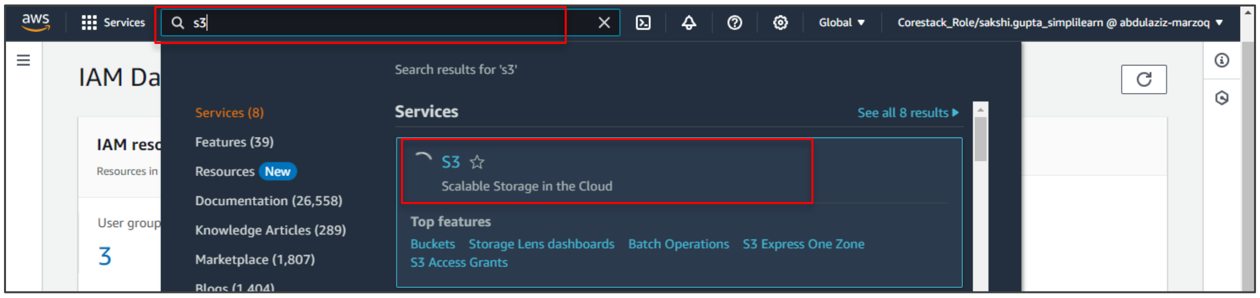
Note: The folder structure created in the previous demos is used here. It is also included in the resources section of LMS. Please refer Lesson_10_demo_01

Steps to be followed:

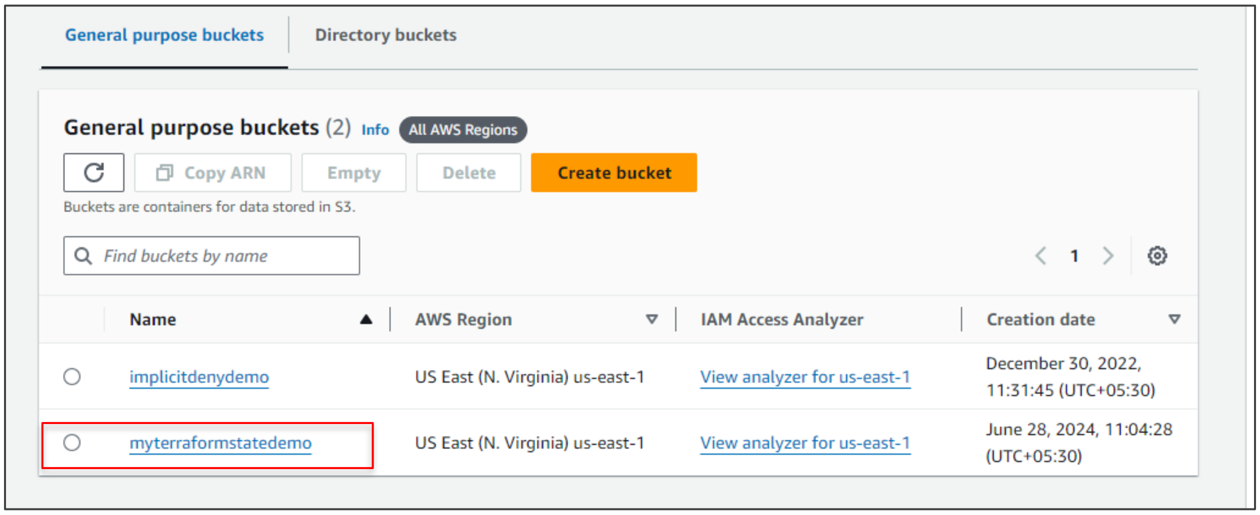
1. Enable versioning on the S3 bucket
2. Enable encryption on the S3 bucket
3. Enable locking for the S3 backend
4. Remove existing resources with the terraform destroy command

Step 1: Enable versioning on the S3 bucket

1.1 Log in to the AWS Management Console and navigate to the **S3** service using the search bar



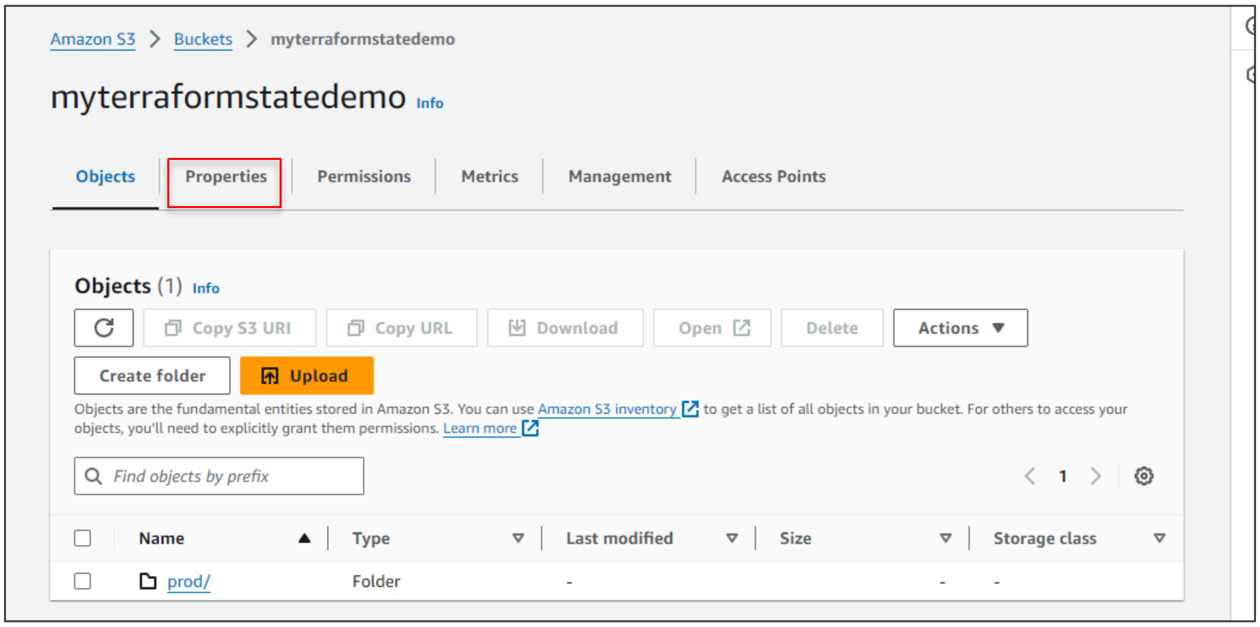
1.2 Click on the previously created bucket named **myterraformstatedemo**



Buckets are containers for data stored in S3.

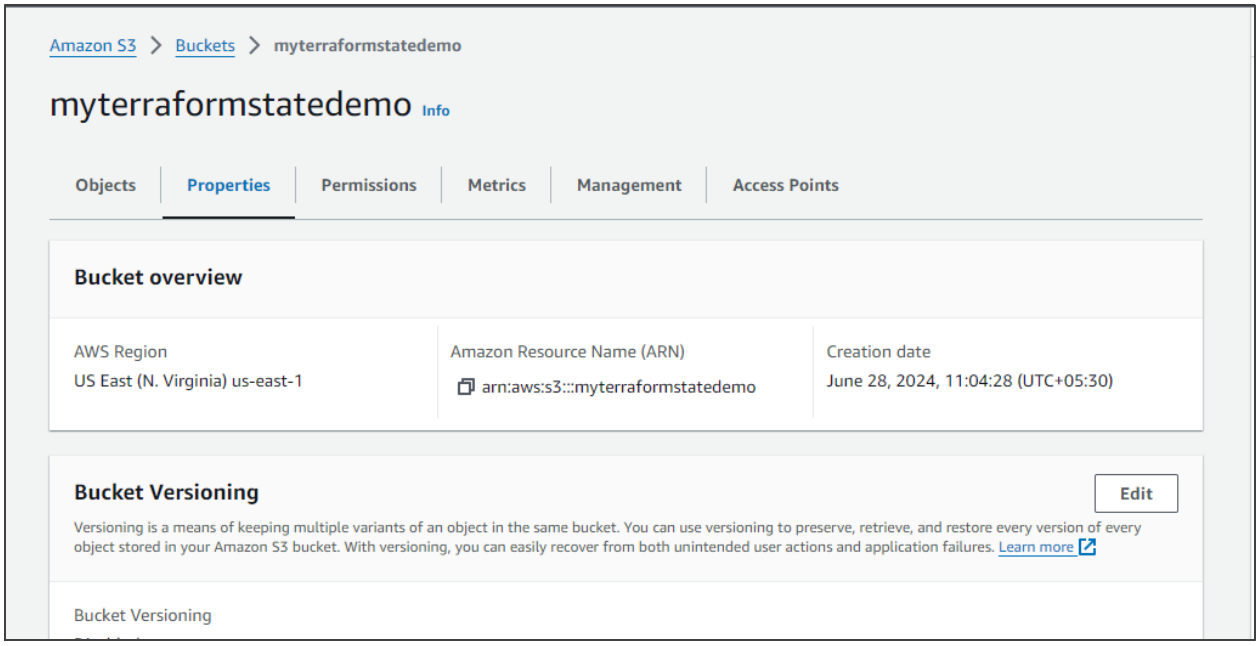
Name	AWS Region	IAM Access Analyzer	Creation date
implicitdenydemo	US East (N. Virginia) us-east-1	View analyzer for us-east-1	December 30, 2022, 11:31:45 (UTC+05:30)
myterraformstatedemo	US East (N. Virginia) us-east-1	View analyzer for us-east-1	June 28, 2024, 11:04:28 (UTC+05:30)

1.3 Click on **Properties**



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
prod/	Folder	-	-	-


Amazon S3 > Buckets > myterraformstatedemo

myterraformstatedemo Info

Bucket overview

AWS Region
US East (N. Virginia) us-east-1

Amazon Resource Name (ARN)
arn:aws:s3:::myterraformstatedemo

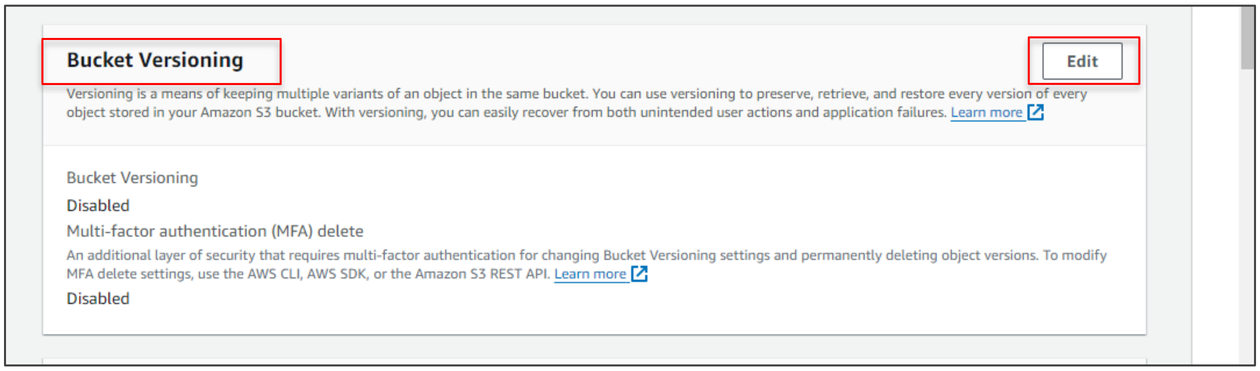
Creation date
June 28, 2024, 11:04:28 (UTC+05:30)

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

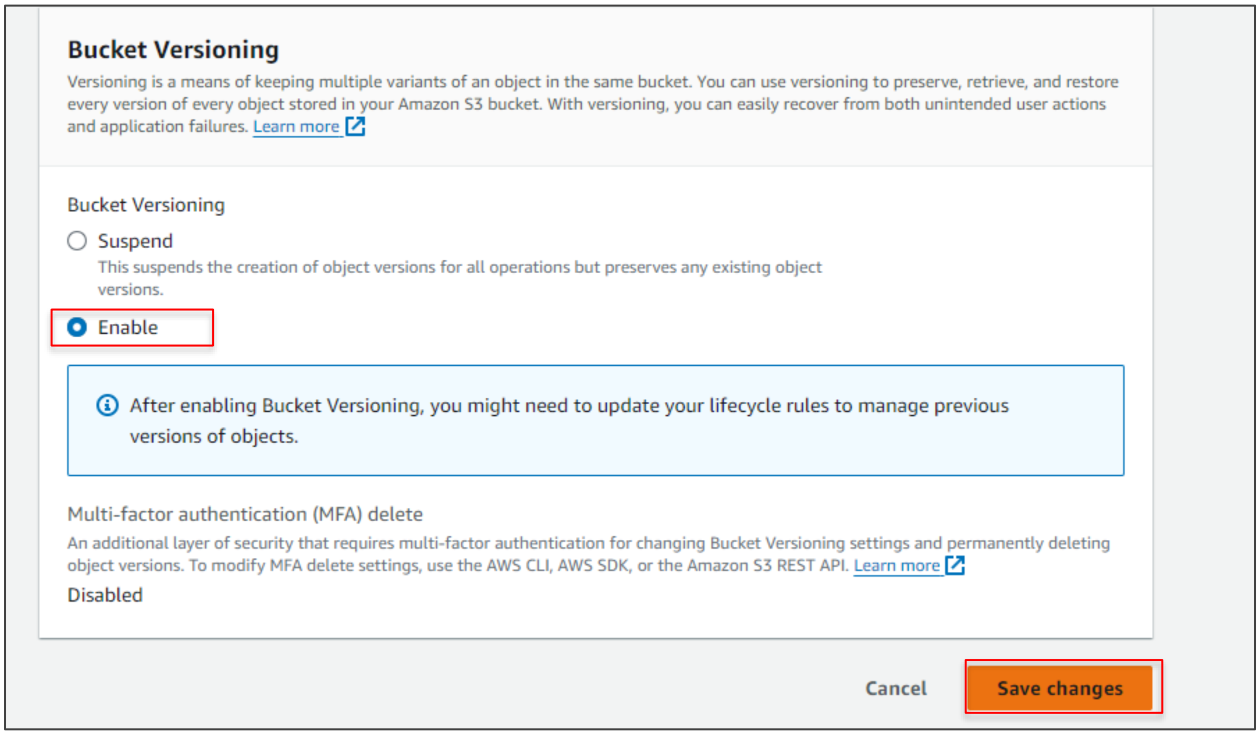
1.4 Scroll down to **Bucket Versioning** and click on **Edit**


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

1.5 Select **Enable** and click on **Save changes**



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

Suspend [✓ Enable]

This suspends the creation of object versions for all operations but preserves any existing object versions.

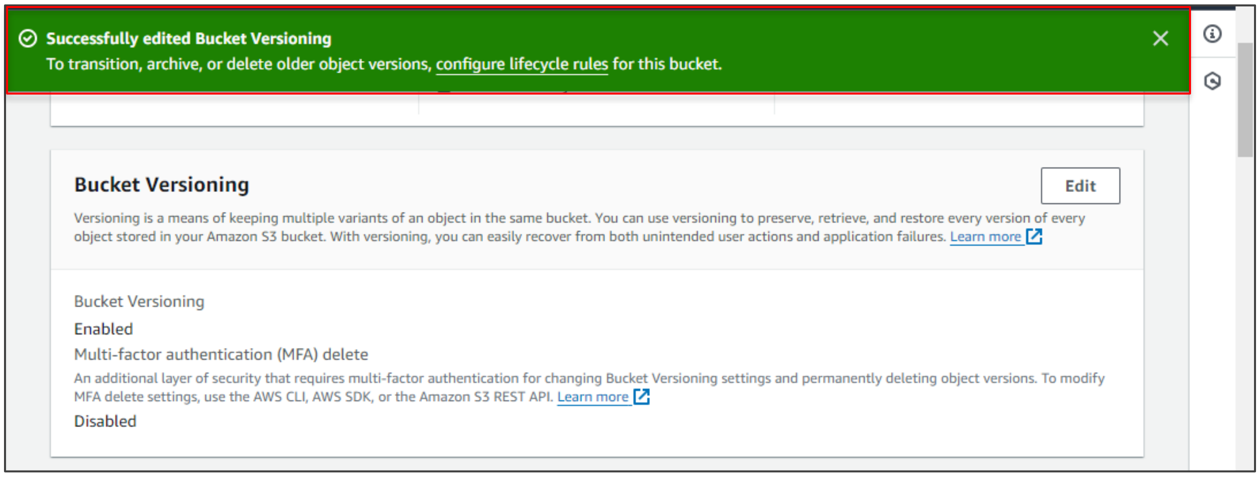
After enabling Bucket Versioning, you might need to update your lifecycle rules to manage previous versions of objects.

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

Versioning on the S3 bucket is successfully enabled, and the following message is shown:



Successfully edited Bucket Versioning

To transition, archive, or delete older object versions, configure lifecycle rules for this bucket.

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

Enabled

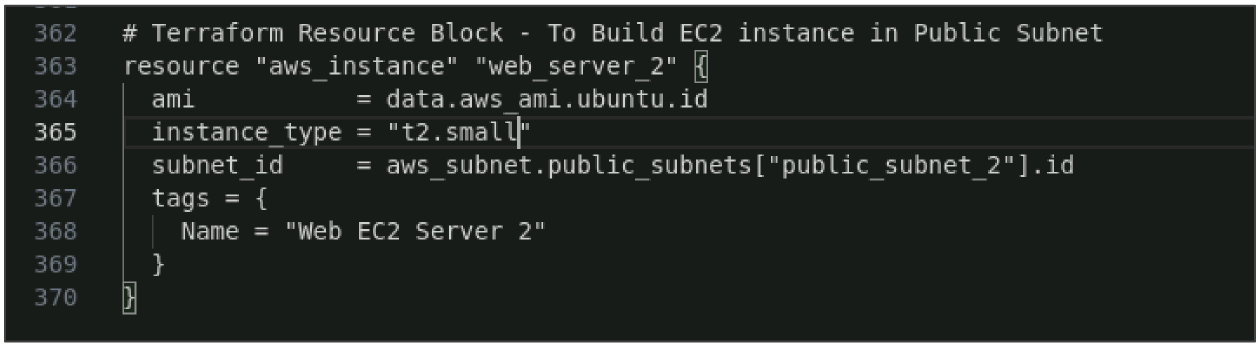
Multi-factor authentication (MFA) delete

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

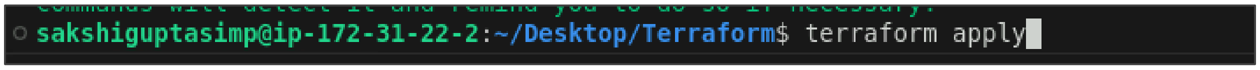
1.6 Make a configuration change in Terraform **main.tf** file to generate a new state version using the following code:

```
resource "aws_instance" "web_server" {
 ami         = "ami-12345678"
 instance_type = "t2.small"
 subnet_id     = "subnet-12345678"
 tags = {
   Name = "Web EC2 Server 2"
 }
}
```



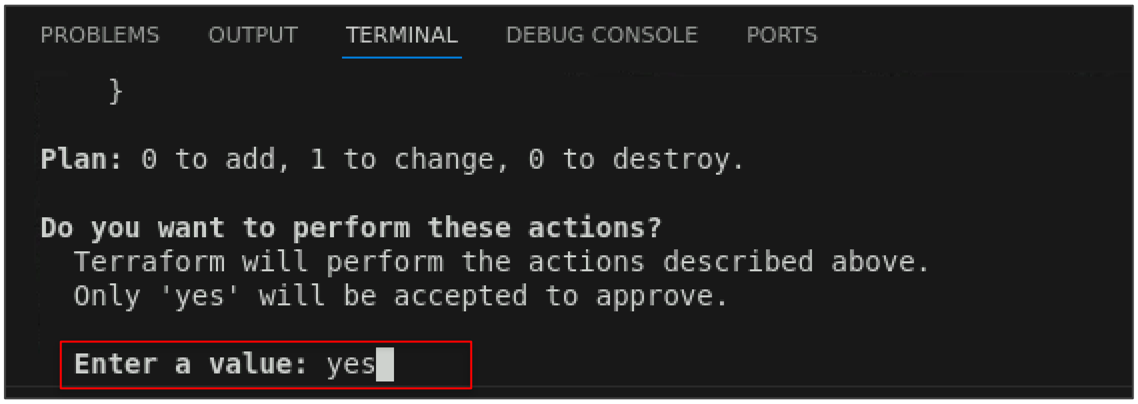
```
# Terraform Resource Block - To Build EC2 instance in Public Subnet
resource "aws_instance" "web_server_2" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t2.small"
  subnet_id     = aws_subnet.public_subnets["public_subnet_2"].id
  tags = {
    Name = "Web EC2 Server 2"
  }
}
```

1.7 Execute the following command in the terminal to apply the configuration changes: **terraform apply**



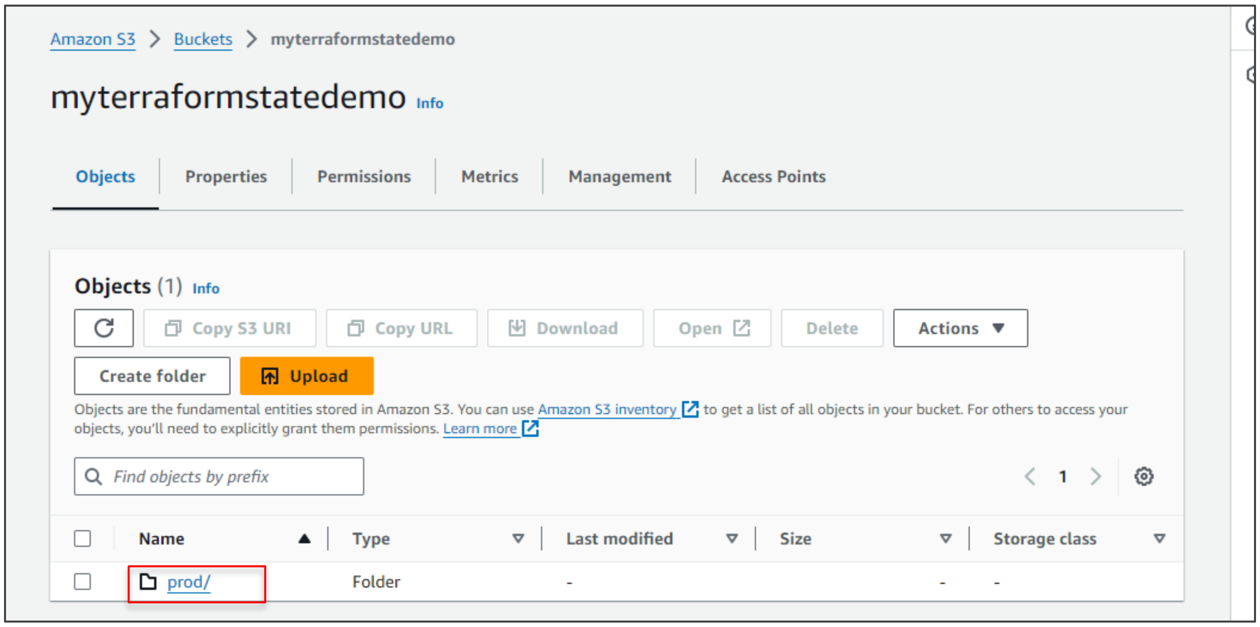
```
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform apply
```

1.8 When prompted, approve the changes by typing **yes**



```
    }

Plan: 0 to add, 1 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
```

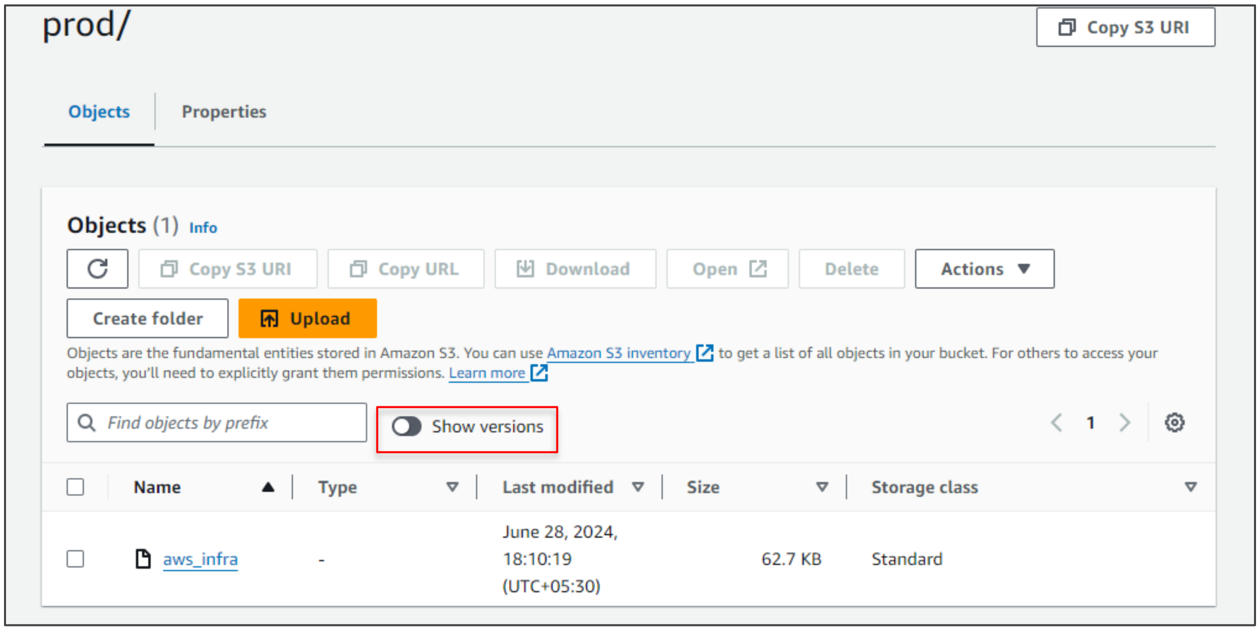
1.9 Go to the Amazon S3 management console and click on the folder named **/prod**



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

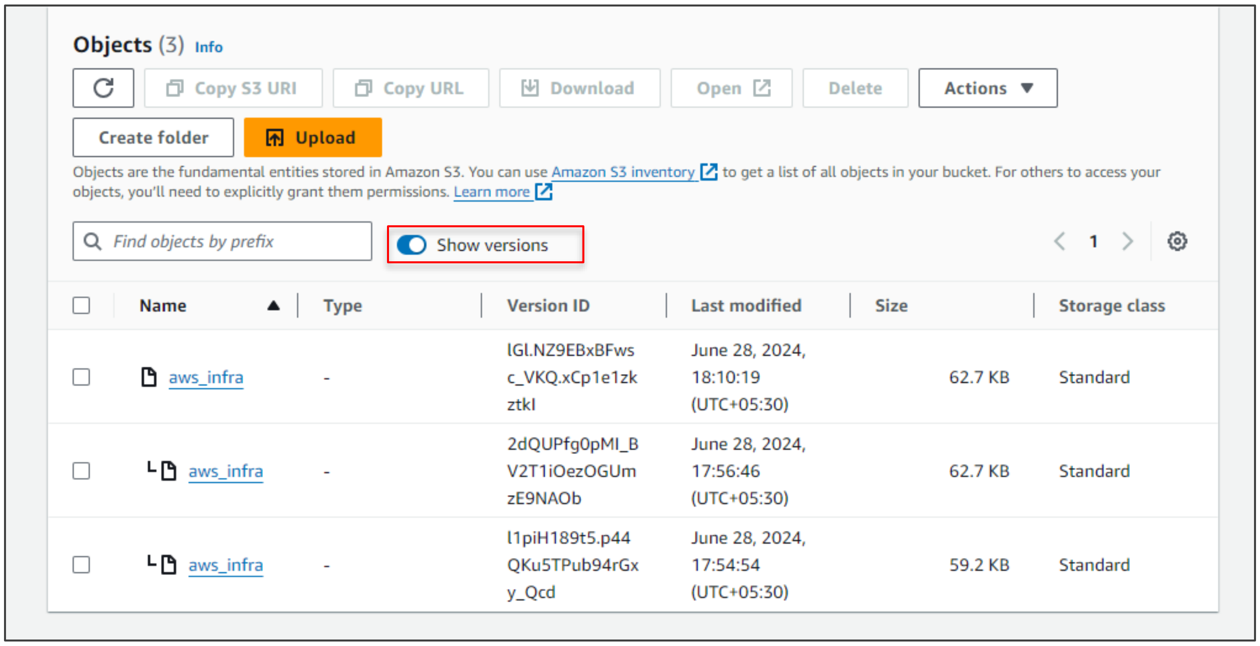
Name	Type	Last modified	Size	Storage class
prod/	Folder	-	-	-

1.10 Enable **Show versions** and the newly created versions of the Terraform configurations will appear



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
aws_infra	-	June 28, 2024, 18:10:19 (UTC+05:30)	62.7 KB	Standard

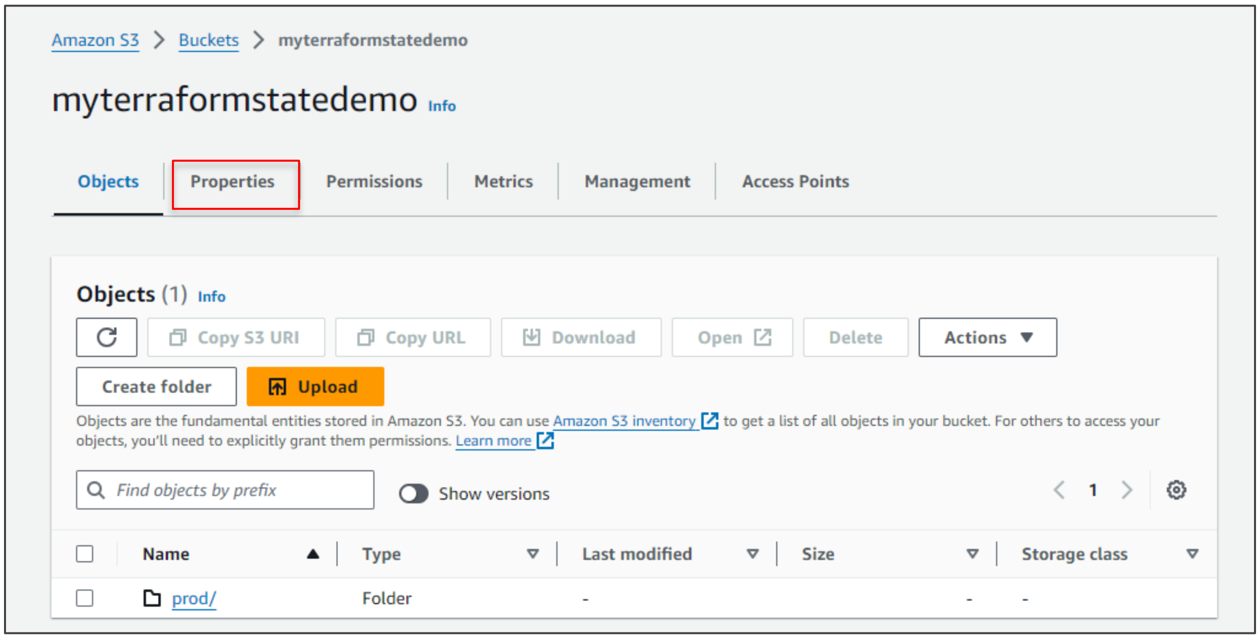


Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Version ID	Last modified	Size	Storage class
aws_infra	-	lGl.NZ9EBxBFwsc_VKQ.xCp1e1zkztkl	June 28, 2024, 18:10:19 (UTC+05:30)	62.7 KB	Standard
aws_infra	-	2dQUPfg0pMI_BV2T1iOezOGUmzE9NAOb	June 28, 2024, 17:56:46 (UTC+05:30)	62.7 KB	Standard
aws_infra	-	l1piH189t5.p44QKu5TPub94rGxy_Qcd	June 28, 2024, 17:54:54 (UTC+05:30)	59.2 KB	Standard

Step 2: Enable encryption on the S3 bucket

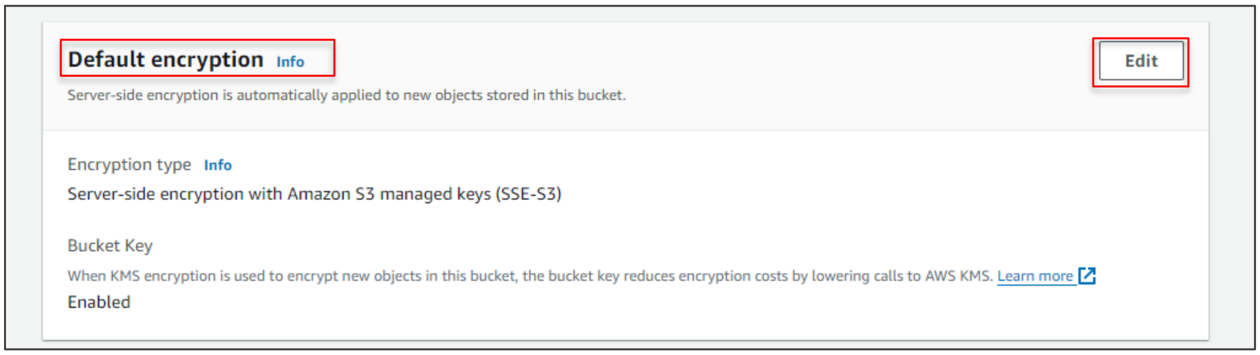
2.1 In the Amazon S3 management console, go into the created bucket named **myterraformstatedemo,** and click on **Properties**



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
prod/	Folder	-	-	-

2.2 Scroll down to the **Default encryption** section and click on **Edit**



Default encryption Info

Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type Info

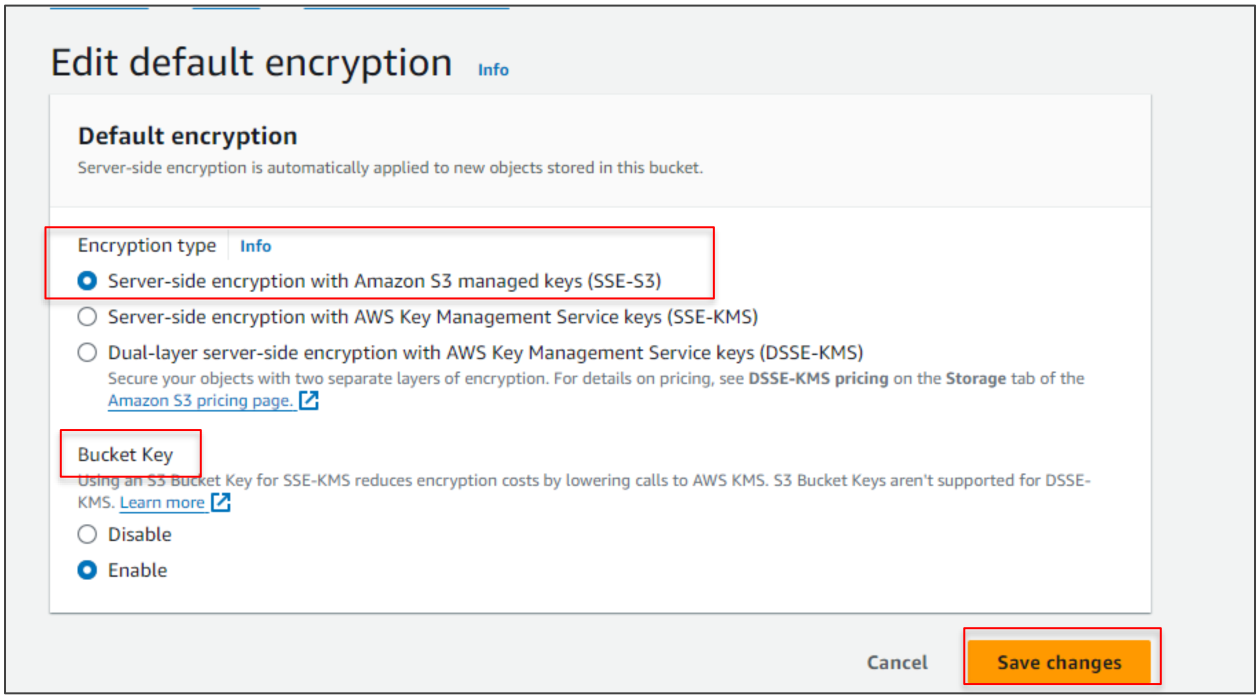
Server-side encryption with Amazon S3 managed keys (SSE-S3)

Bucket Key

When KMS encryption is used to encrypt new objects in this bucket, the bucket key reduces encryption costs by lowering calls to AWS KMS. Learn more

Enabled

2.3 Make sure the **Encryption type** is selected and the **Bucket Key** is enabled as shown in the following screenshot. Click on **Save changes**.



Edit default encryption Info

Default encryption

Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type Info

[✓ Server-side encryption with Amazon S3 managed keys (SSE-S3)]

Server-side encryption with AWS Key Management Service keys (SSE-KMS)

Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)

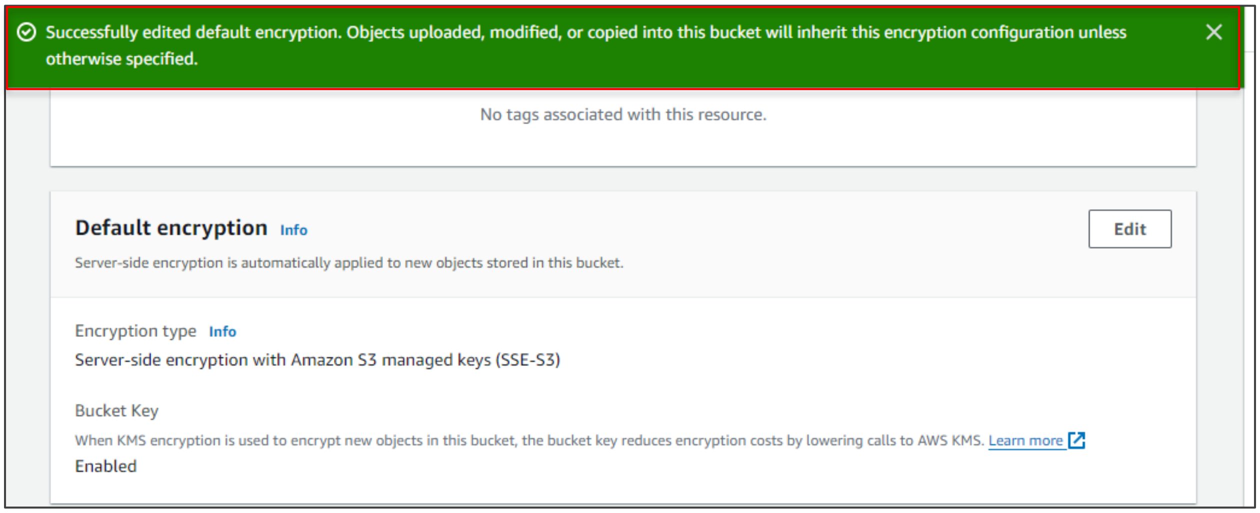
Secure your objects with two separate layers of encryption. For details on pricing, see **DSSE-KMS pricing** on the **Storage** tab of the Amazon S3 pricing page.

Bucket Key

Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. Learn more

Disable [✓ Enable]

Encryption on the S3 bucket is successfully enabled, and the following message will be displayed:



Successfully edited default encryption. Objects uploaded, modified, or copied into this bucket will inherit this encryption configuration unless otherwise specified.

No tags associated with this resource.

Default encryption Info

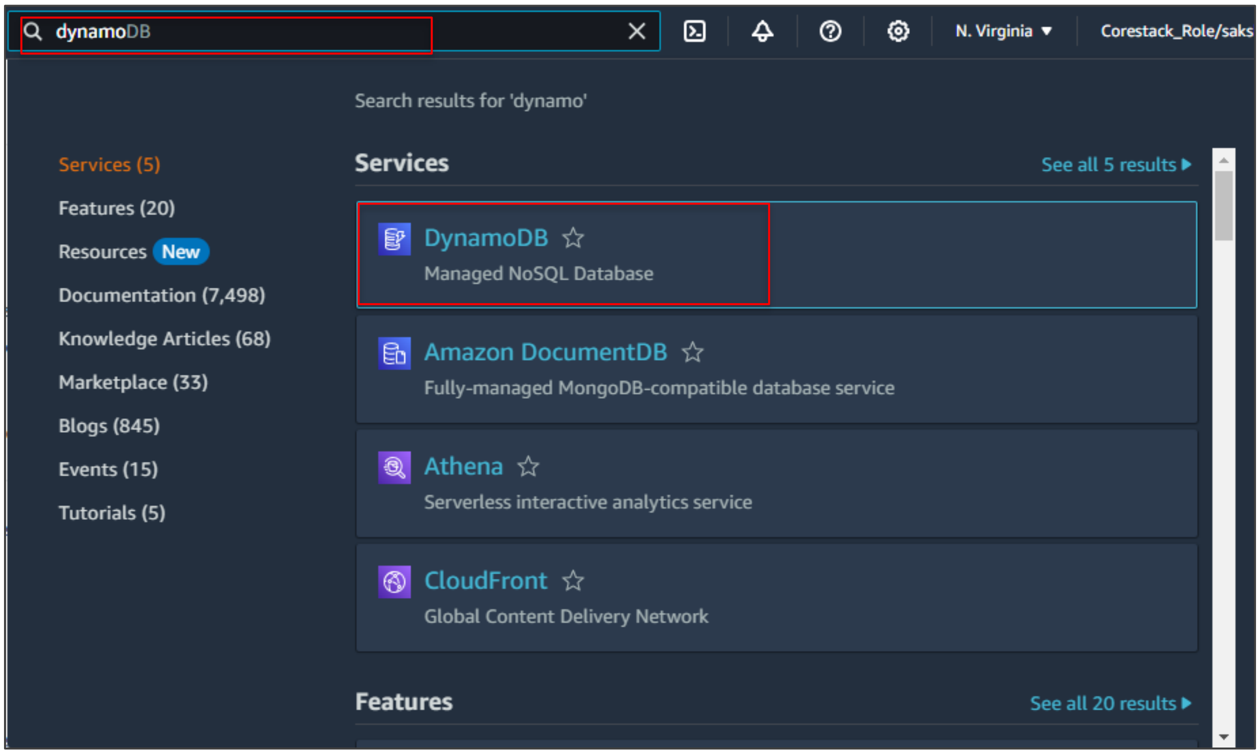
Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type Info
Server-side encryption with Amazon S3 managed keys (SSE-S3)

Bucket Key
When KMS encryption is used to encrypt new objects in this bucket, the bucket key reduces encryption costs by lowering calls to AWS KMS. Learn more
Enabled

Step 3: Enable locking for the S3 backend

3.1 Go to the AWS Console, search for **dynamoDB** in the search bar, and select the **DynamoDB** service

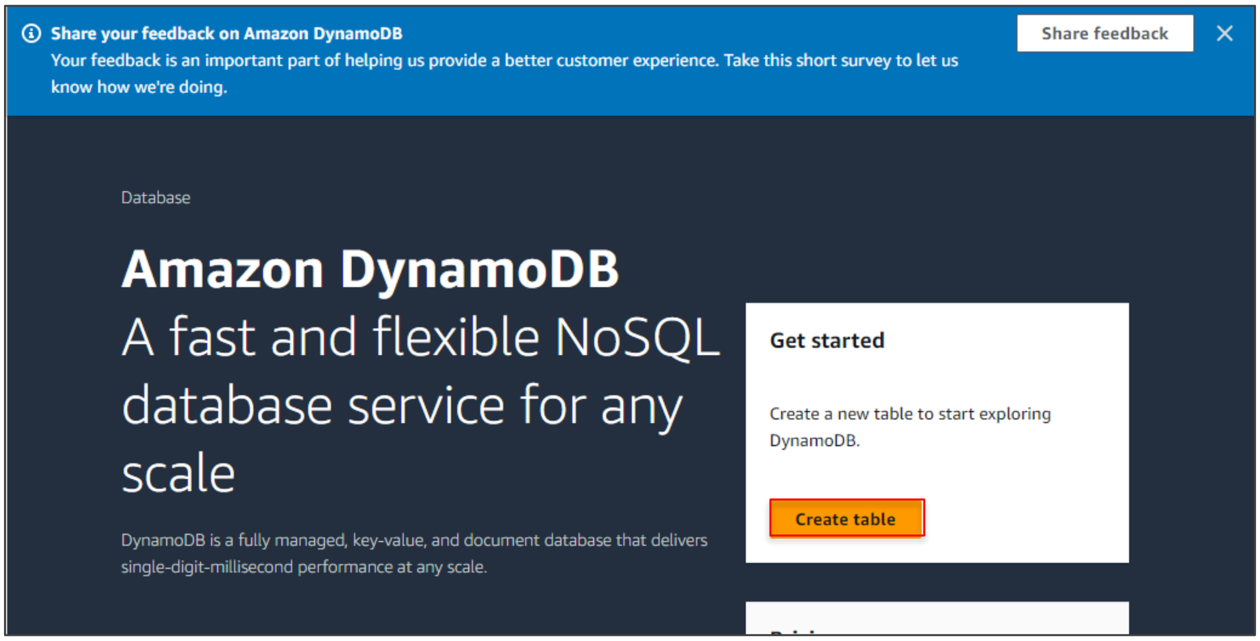


dynamoDB

Search results for 'dynamo'

Services
- DynamoDB — Managed NoSQL Database
- Amazon DocumentDB — Fully-managed MongoDB-compatible database service
- Athena — Serverless interactive analytics service
- CloudFront — Global Content Delivery Network

3.2 Click on **Create table**



Share your feedback on Amazon DynamoDB
Your feedback is an important part of helping us provide a better customer experience. Take this short survey to let us know how we're doing.

Database

Amazon DynamoDB
A fast and flexible NoSQL database service for any scale

DynamoDB is a fully managed, key-value, and document database that delivers single-digit-millisecond performance at any scale.

Get started

Create a new table to start exploring DynamoDB.

3.3 Name the table as **terraform-locks** and add the **Partition key** as **LockID**

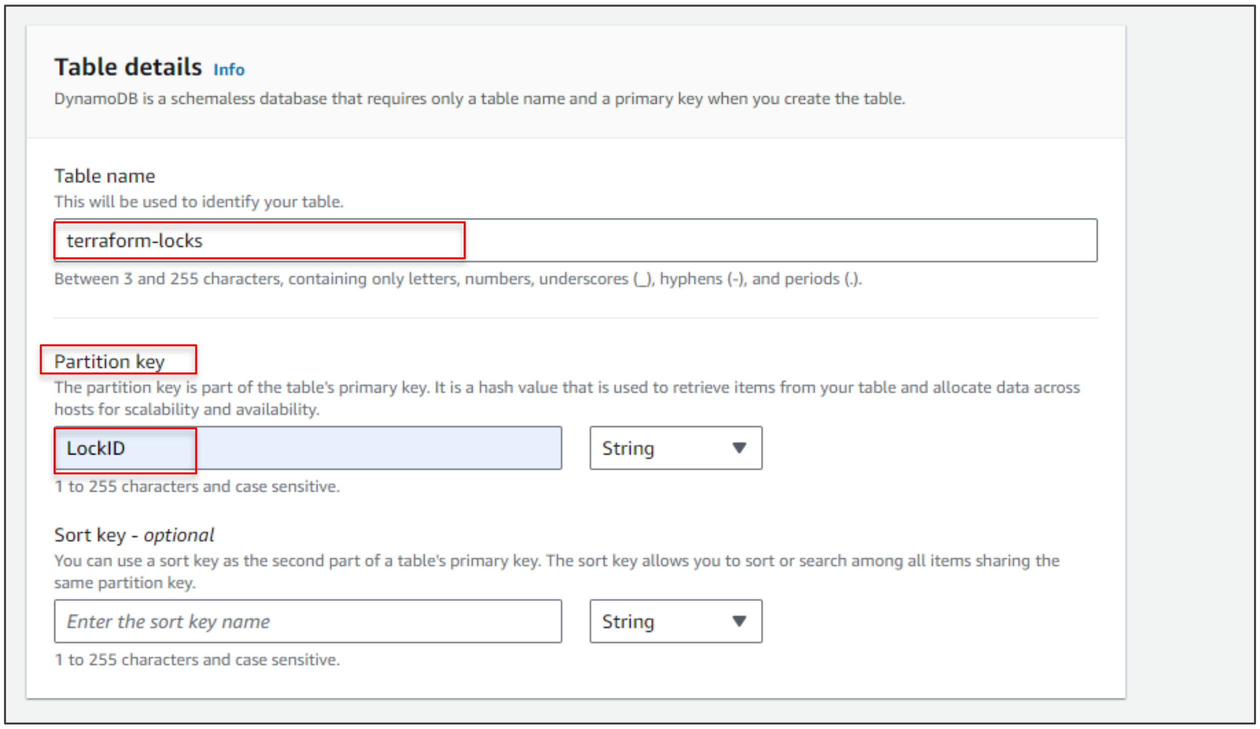


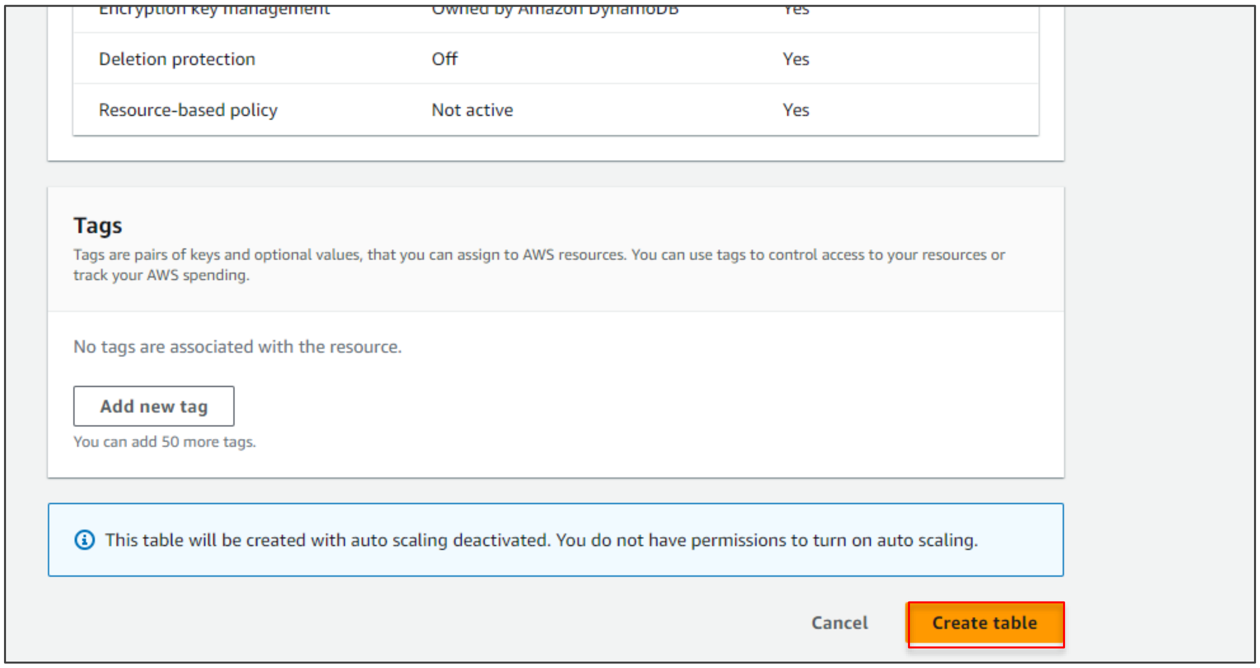
Table details Info
DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.
terraform-locks
Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.
LockID | String
1 to 255 characters and case sensitive.

Sort key - *optional*
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.
Enter the sort key name | String
1 to 255 characters and case sensitive.

3.4 Retain the other settings as default and click on **Create table**



Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes
Resource-based policy	Not active	Yes

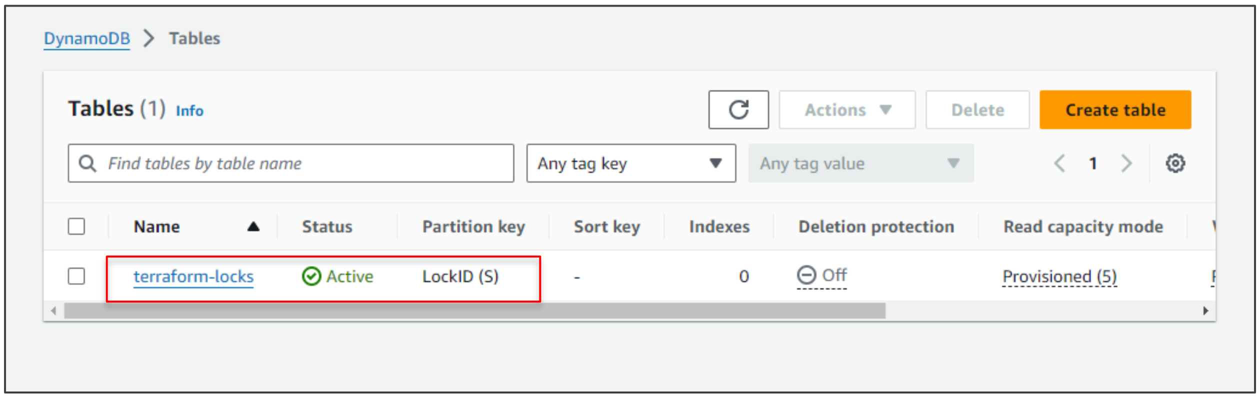
Tags

Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

You can add 50 more tags.

This table will be created with auto scaling deactivated. You do not have permissions to turn on auto scaling.

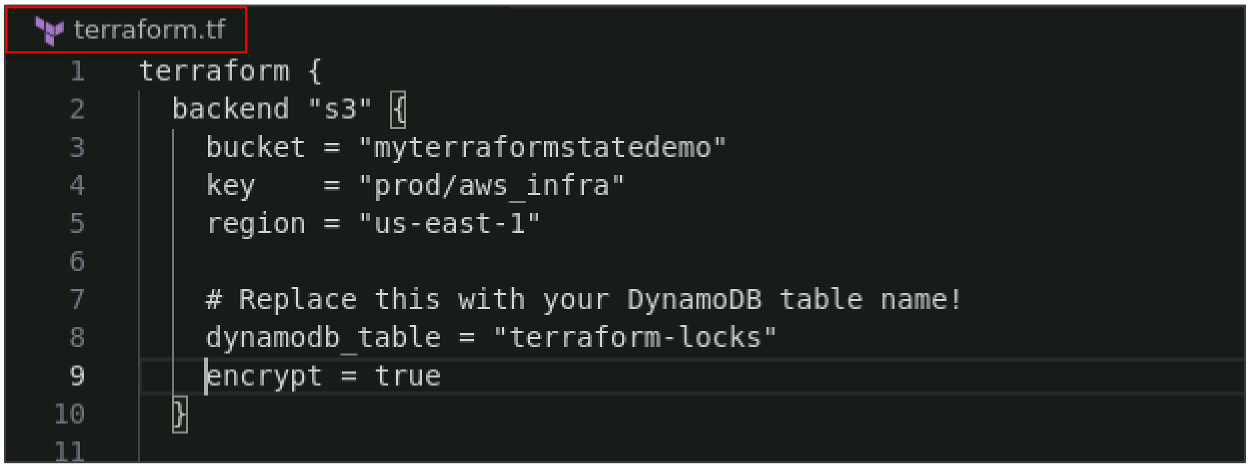


	Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode
	terraform-locks	Active	LockID (S)	-	0	Off	Provisioned (5)

The table will be created as shown above.

3.5 Update the backend configuration in **terraform.tf** to use the DynamoDB table with the following code:

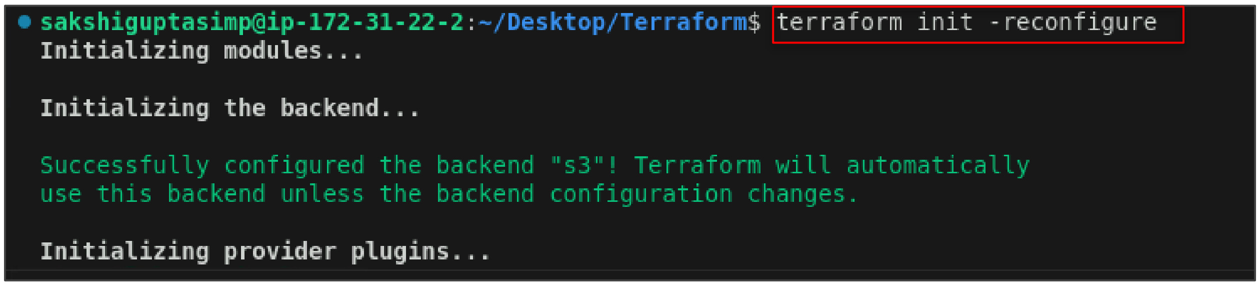
```
terraform {
  backend "s3" {
    bucket        = "myterraformstatedemo"
    key           = "prod/aws_infra"
    region        = "us-east-1"
    dynamodb_table = "terraform-locks"
    encrypt       = true
  }
}
```



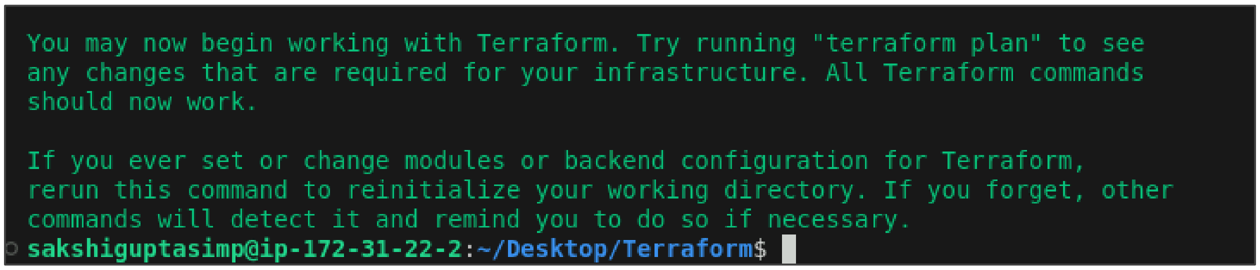
```
terraform {
  backend "s3" {
    bucket = "myterraformstatedemo"
    key    = "prod/aws_infra"
    region = "us-east-1"

    # Replace this with your DynamoDB table name!
    dynamodb_table = "terraform-locks"
    encrypt = true
  }
```

3.6 Reinitialize the backend using the following command:

terraform init -reconfigure



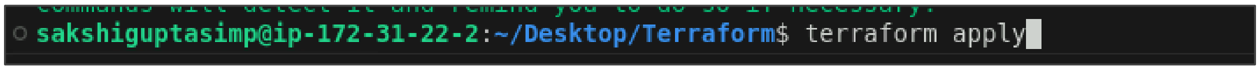
```
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform init -reconfigure
Initializing modules...

Initializing the backend...

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing provider plugins...
```



```
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.
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$
```

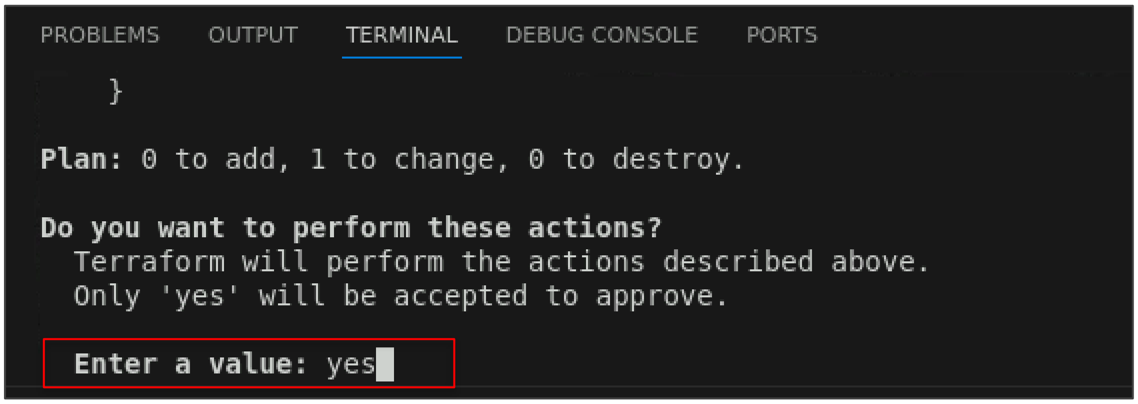
3.7 Use the following command to apply the configuration changes:

terraform apply



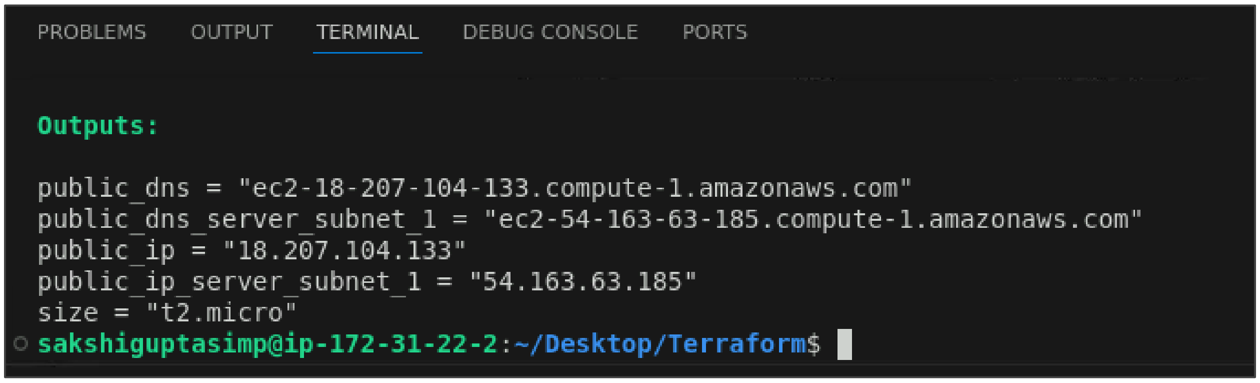
```
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform apply
```

3.8 When prompted, approve the changes by typing **yes**



PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE PORTS

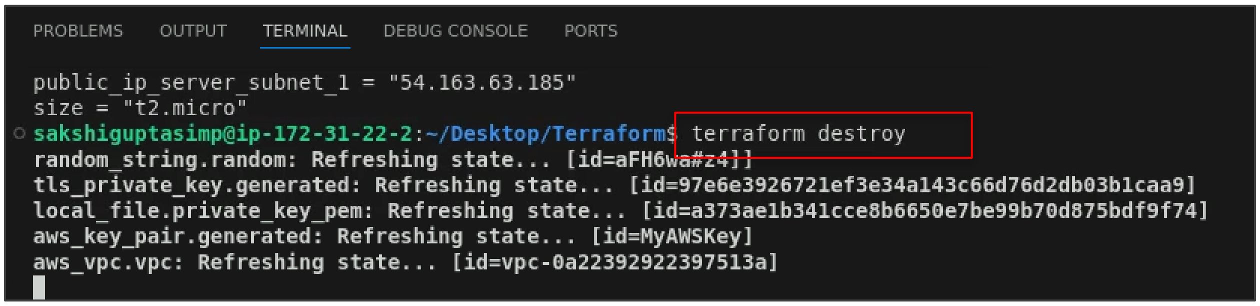
```
    }

Plan: 0 to add, 1 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
```



```
Outputs:

public_dns = "ec2-18-207-104-133.compute-1.amazonaws.com"
public_dns_server_subnet_1 = "ec2-54-163-63-185.compute-1.amazonaws.com"
public_ip = "18.207.104.133"
public_ip_server_subnet_1 = "54.163.63.185"
size = "t2.micro"
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$
```

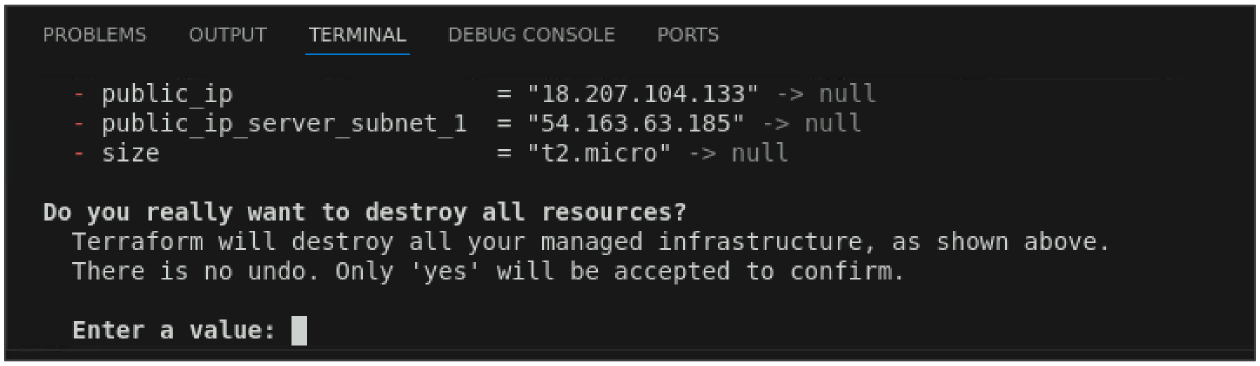
Step 4: Remove existing resources with the terraform destroy command

4.1 Clean up and destroy all managed infrastructure to prepare for the next demo using the following command:
terraform destroy

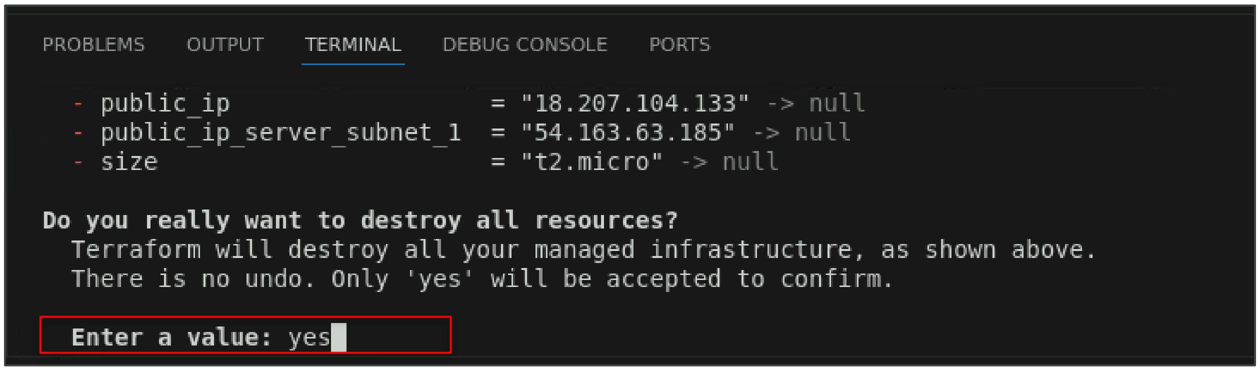


```
public_ip_server_subnet_1 = "54.163.63.185"
size = "t2.micro"
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform destroy
random_string.random: Refreshing state... [id=aFH6wa#z4]]
tls_private_key.generated: Refreshing state... [id=97e6e3926721ef3e34a143c66d76d2db03b1caa9]
local_file.private_key_pem: Refreshing state... [id=a373ae1b341cce8b6650e7be99b70d875bdf9f74]
aws_key_pair.generated: Refreshing state... [id=MyAWSKey]
aws_vpc.vpc: Refreshing state... [id=vpc-0a22392922397513a]
```

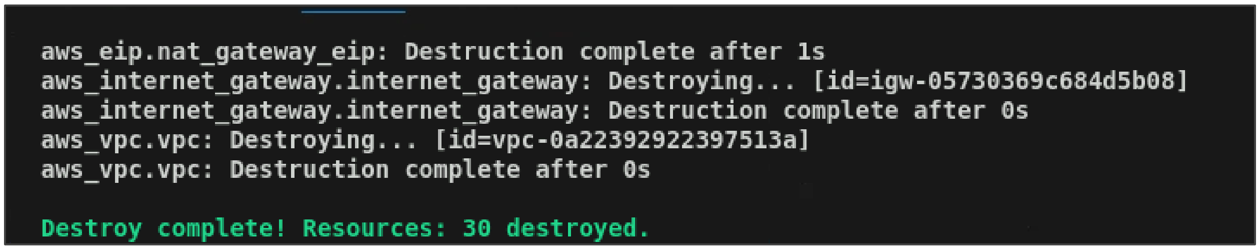
4.2 When prompted, approve the destroy by typing **yes**



```
  - public_ip                   = "18.207.104.133" -> null
  - public_ip_server_subnet_1   = "54.163.63.185" -> null
  - size                        = "t2.micro" -> null

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value:
```

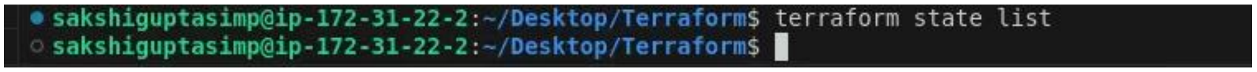


```
  - public_ip                   = "18.207.104.133" -> null
  - public_ip_server_subnet_1   = "54.163.63.185" -> null
  - size                        = "t2.micro" -> null

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes
```



```
aws_eip.nat_gateway_eip: Destruction complete after 1s
aws_internet_gateway.internet_gateway: Destroying... [id=igw-05730369c684d5b08]
aws_internet_gateway.internet_gateway: Destruction complete after 0s
aws_vpc.vpc: Destroying... [id=vpc-0a22392922397513a]
aws_vpc.vpc: Destruction complete after 0s

Destroy complete! Resources: 30 destroyed.
```

4.3 Validate the destruction using the following command:
terraform state list



```
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform state list
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ 
```

Since there are no files present, this indicates that the infrastructure has been successfully cleaned and destroyed.

By following these steps, you have successfully configured and managed Terraform state using the AWS S3 backend for ensuring reliable state storage and management.