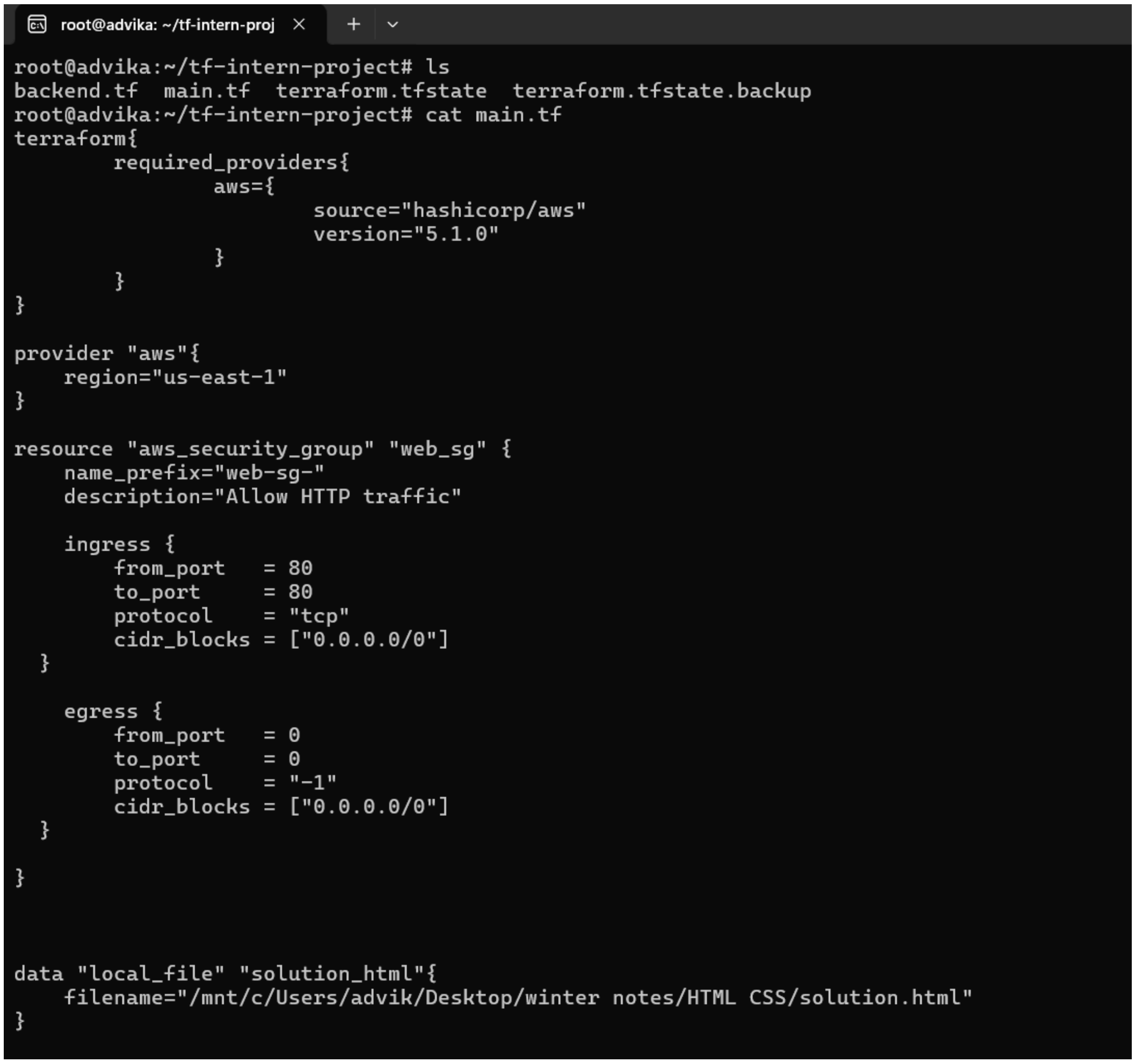
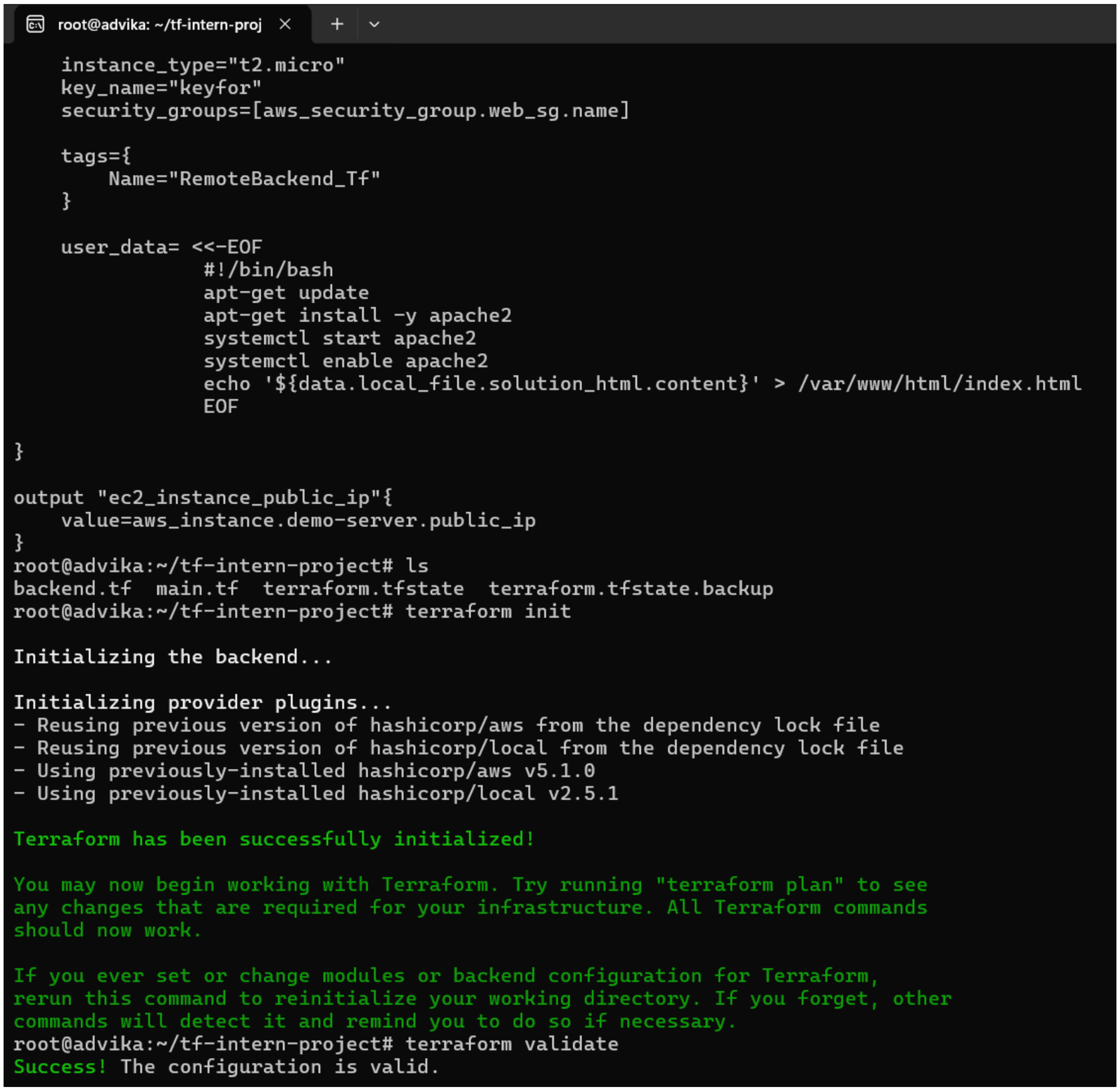
MODIFYING THE RESOURCES AND CHECKING THE UPDATION OF STATE FILE ON S3

Changed the *index.html* file while keeping everything else the same
The public IP now hits a *solution.html* file

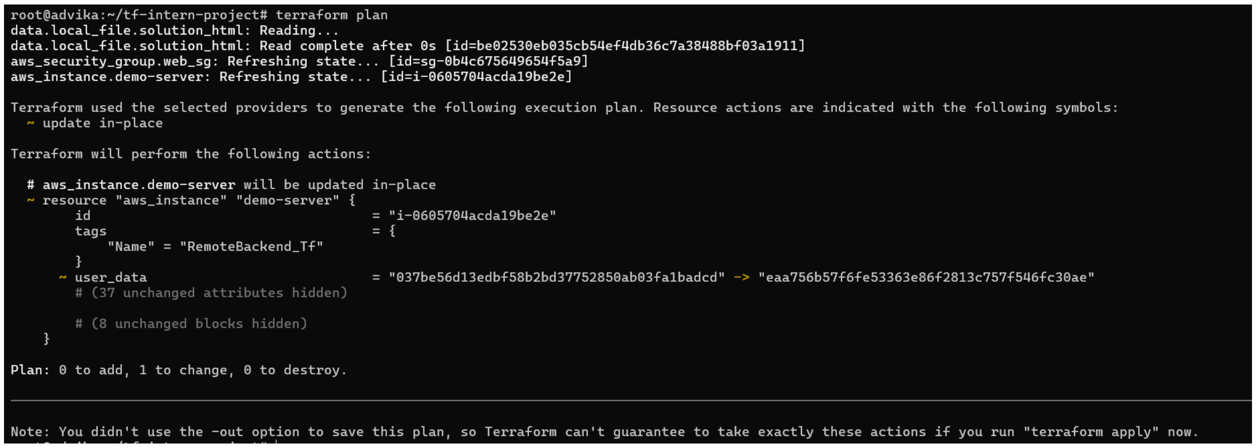


```
root@advika:~/tf-intern-project# ls
backend.tf  main.tf  terraform.tfstate  terraform.tfstate.backup
root@advika:~/tf-intern-project# cat main.tf
terraform{
        required_providers{
                aws={
                        source="hashicorp/aws"
                        version="5.1.0"
                }
        }
}

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

resource "aws_security_group" "web_sg" {
    name_prefix="web-sg-"
    description="Allow HTTP traffic"

    ingress {
        from_port   = 80
        to_port     = 80
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
  }

    egress {
        from_port   = 0
        to_port     = 0
        protocol    = "-1"
        cidr_blocks = ["0.0.0.0/0"]
  }

}



data "local_file" "solution_html"{
    filename="/mnt/c/Users/advik/Desktop/winter notes/HTML CSS/solution.html"
}
```

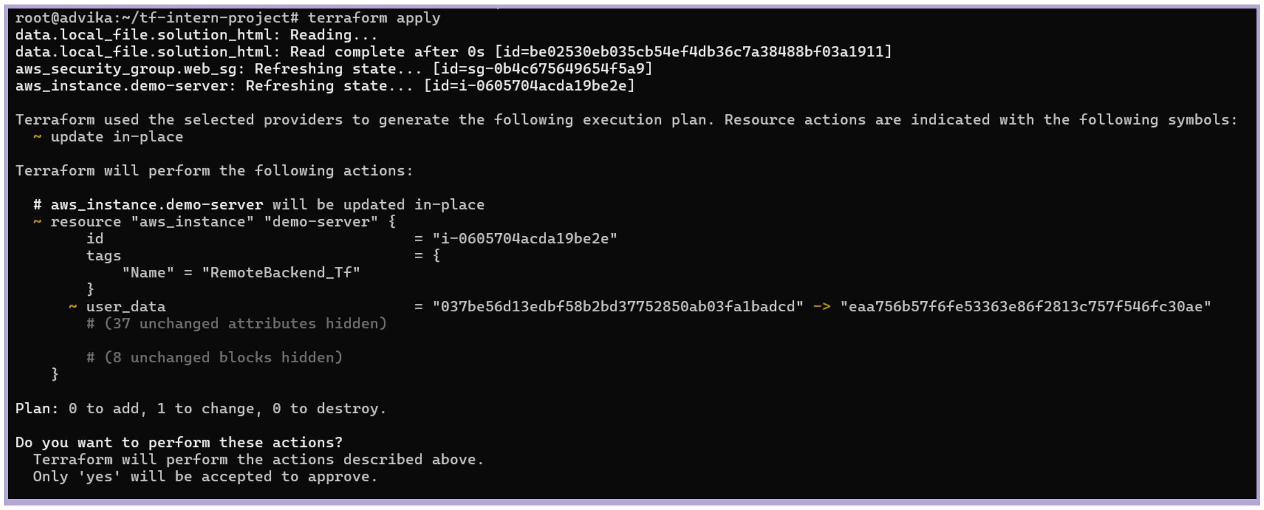


```
    instance_type="t2.micro"
    key_name="keyfor"
    security_groups=[aws_security_group.web_sg.name]

    tags={
        Name="RemoteBackend_Tf"
    }

    user_data= <<-EOF
                #!/bin/bash
                apt-get update
                apt-get install -y apache2
                systemctl start apache2
                systemctl enable apache2
                echo '${data.local_file.solution_html.content}' > /var/www/html/index.html
                EOF

}

output "ec2_instance_public_ip"{
    value=aws_instance.demo-server.public_ip
}
root@advika:~/tf-intern-project# ls
backend.tf  main.tf  terraform.tfstate  terraform.tfstate.backup
root@advika:~/tf-intern-project# terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/local from the dependency lock file
- Using previously-installed hashicorp/aws v5.1.0
- Using previously-installed hashicorp/local v2.5.1

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.
root@advika:~/tf-intern-project# terraform validate
Success! The configuration is valid.
```

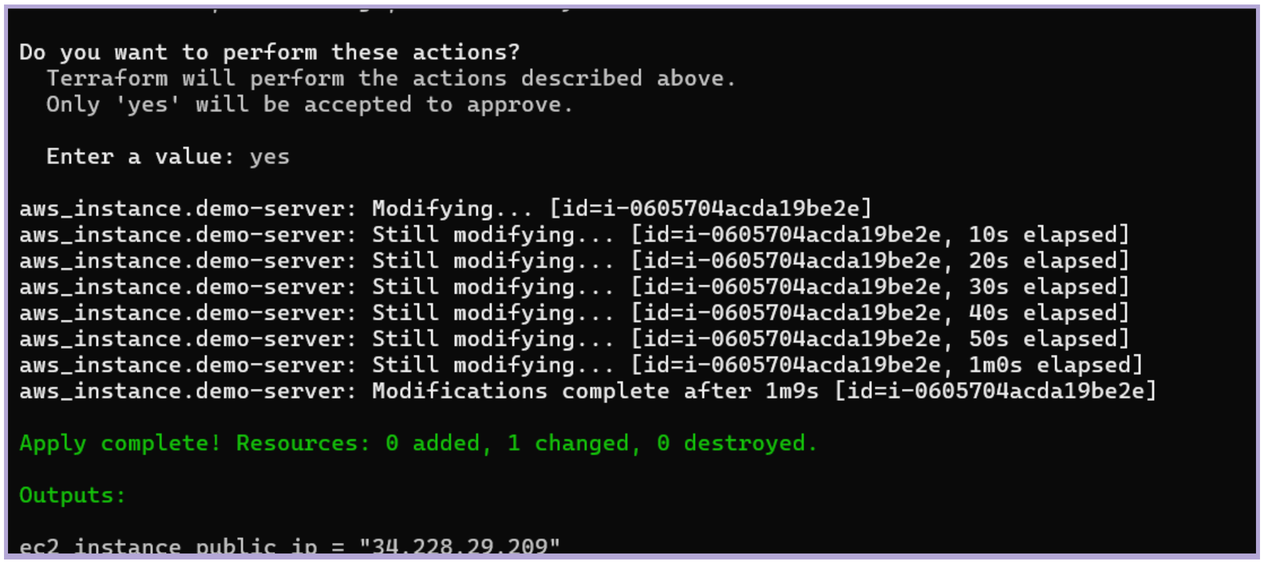
Only 1 change is planned and it is reflected so in the *terraform plan*



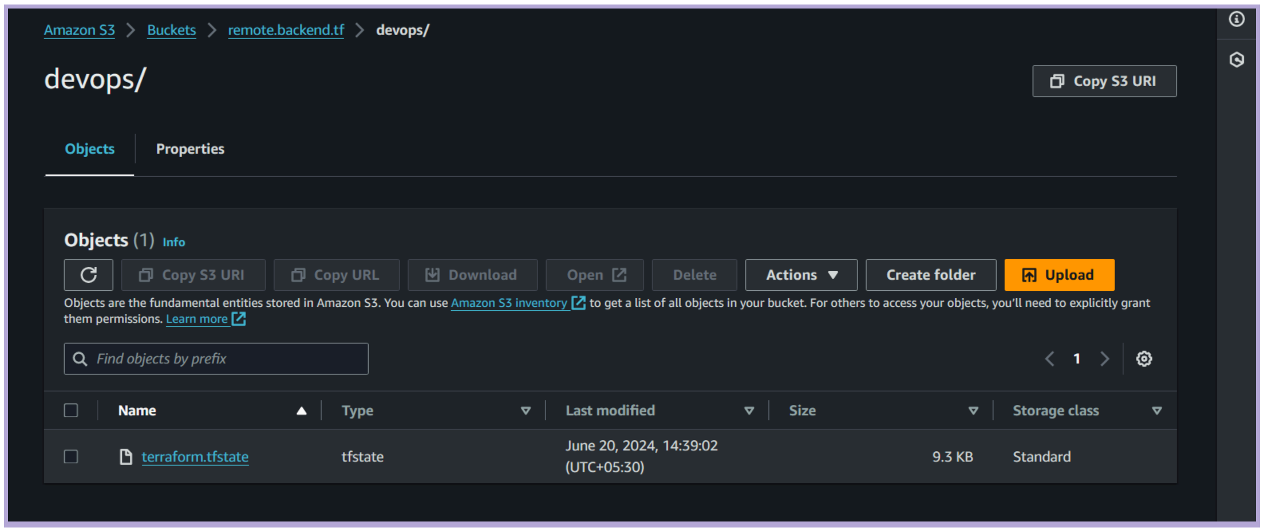
```
root@advika:~/tf-intern-project# terraform plan
data.local_file.solution_html: Reading...
data.local_file.solution_html: Read complete after 0s [id=be02530eb035cb54ef4db36c7a38488bf03a1911]
aws_security_group.web_sg: Refreshing state... [id=sg-0b4c675649654f5a9]
aws_instance.demo-server: Refreshing state... [id=i-0605704acda19be2e]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # aws_instance.demo-server will be updated in-place
  ~ resource "aws_instance" "demo-server" {
        id                                   = "i-0605704acda19be2e"
        tags                                 = {
            "Name" = "RemoteBackend_Tf"
        }
      ~ user_data                            = "037be56d13edbf58b2bd37752850ab03fa1badcd" -> "eaa756b57f6fe53363e86f2813c757f546fc30ae"
        # (37 unchanged attributes hidden)

        # (8 unchanged blocks hidden)
    }

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

─────────────────────────────────────────────────────────────────────────────

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.
```



```
root@advika:~/tf-intern-project# terraform apply
data.local_file.solution_html: Reading...
data.local_file.solution_html: Read complete after 0s [id=be02530eb035cb54ef4db36c7a38488bf03a1911]
aws_security_group.web_sg: Refreshing state... [id=sg-0b4c675649654f5a9]
aws_instance.demo-server: Refreshing state... [id=i-0605704acda19be2e]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # aws_instance.demo-server will be updated in-place
  ~ resource "aws_instance" "demo-server" {
        id                                   = "i-0605704acda19be2e"
        tags                                 = {
            "Name" = "RemoteBackend_Tf"
        }
      ~ user_data                            = "037be56d13edbf58b2bd37752850ab03fa1badcd" -> "eaa756b57f6fe53363e86f2813c757f546fc30ae"
        # (37 unchanged attributes hidden)

        # (8 unchanged blocks hidden)
    }

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.
```



```
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.demo-server: Modifying... [id=i-0605704acda19be2e]
aws_instance.demo-server: Still modifying... [id=i-0605704acda19be2e, 10s elapsed]
aws_instance.demo-server: Still modifying... [id=i-0605704acda19be2e, 20s elapsed]
aws_instance.demo-server: Still modifying... [id=i-0605704acda19be2e, 30s elapsed]
aws_instance.demo-server: Still modifying... [id=i-0605704acda19be2e, 40s elapsed]
aws_instance.demo-server: Still modifying... [id=i-0605704acda19be2e, 50s elapsed]
aws_instance.demo-server: Still modifying... [id=i-0605704acda19be2e, 1m0s elapsed]
aws_instance.demo-server: Modifications complete after 1m9s [id=i-0605704acda19be2e]

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.

Outputs:

ec2_instance_public_ip = "34.228.29.209"
```

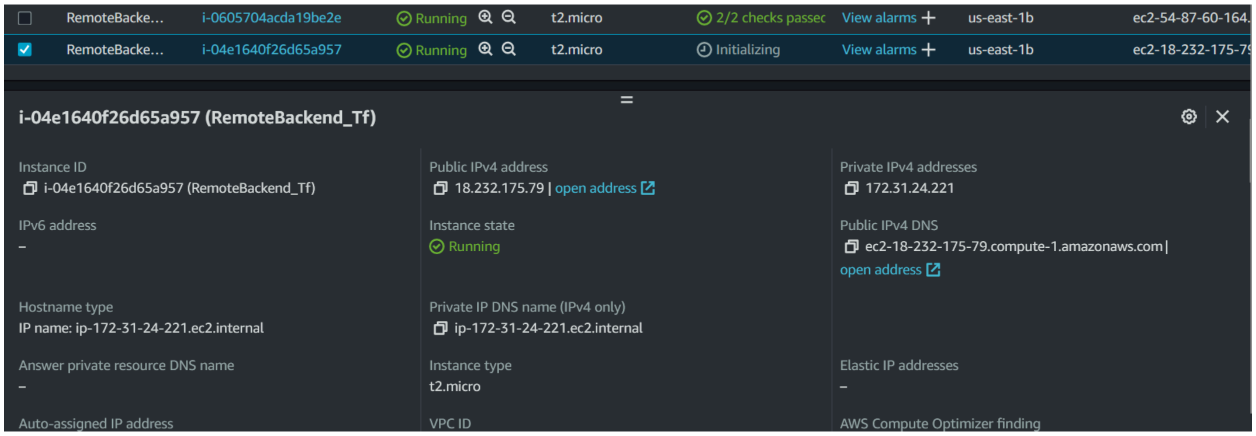


Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
terraform.tfstate	tfstate	June 20, 2024, 14:39:02 (UTC+05:30)	9.3 KB	Standard

The *terraform.tfstate* file is changed in place in the S3 bucket because no separate workspace has been allocated to it.

Tags were added to the previously created *.tfstate* file but it was completely overwritten by this newly created file.



```
☐  RemoteBacke...  i-0605704acda19be2e  Running  t2.micro  2/2 checks passec  View alarms +  us-east-1b  ec2-54-87-60-164.
☑  RemoteBacke...  i-04e1640f26d65a957  Running  t2.micro  Initializing       View alarms +  us-east-1b  ec2-18-232-175-79
```

i-04e1640f26d65a957 (RemoteBackend_Tf)

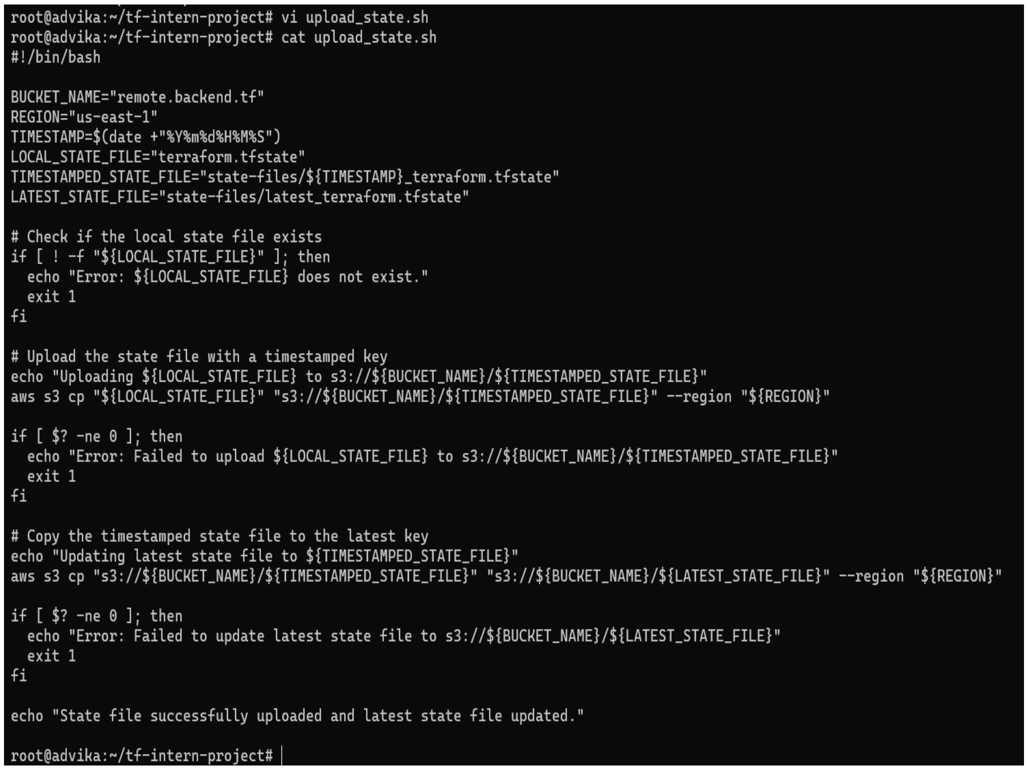
Instance ID i-04e1640f26d65a957 (RemoteBackend_Tf)	Public IPv4 address 18.232.175.79 | open address	Private IPv4 addresses 172.31.24.221
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-18-232-175-79.compute-1.amazonaws.com | open address
Hostname type IP name: ip-172-31-24-221.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-24-221.ec2.internal	
Answer private resource DNS name –	Instance type t2.micro	Elastic IP addresses –
Auto-assigned IP address	VPC ID	AWS Compute Optimizer finding

Two instances of the same name were created by AWS because the .html it was hitting was different and therefore the resource essentially changed.

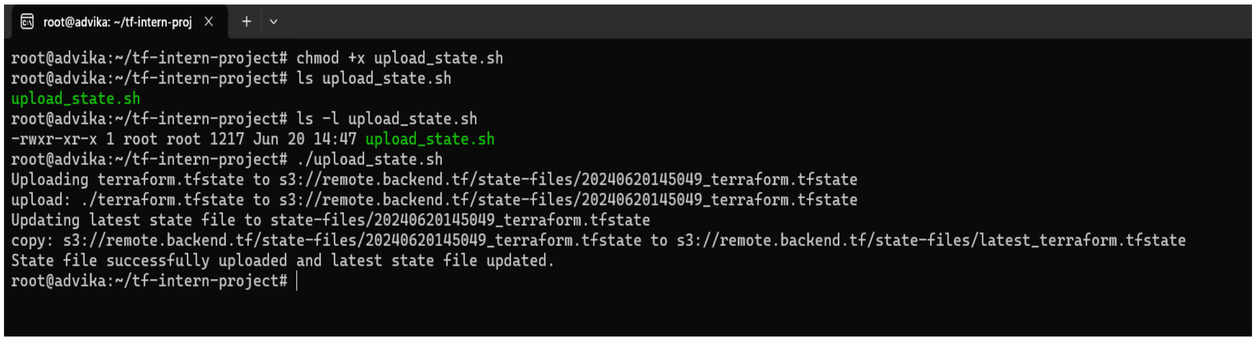
NAME OF .tfstate FILE/PREFIX SHOULD BE THE TIMESTAMP OF CREATION

Terraform's backend configuration does not support dynamic expressions (like timestamps) directly

I wrote a bash script for automating the renaming of the file, the following script made empty copies of the tfstate file renamed to the timestamps

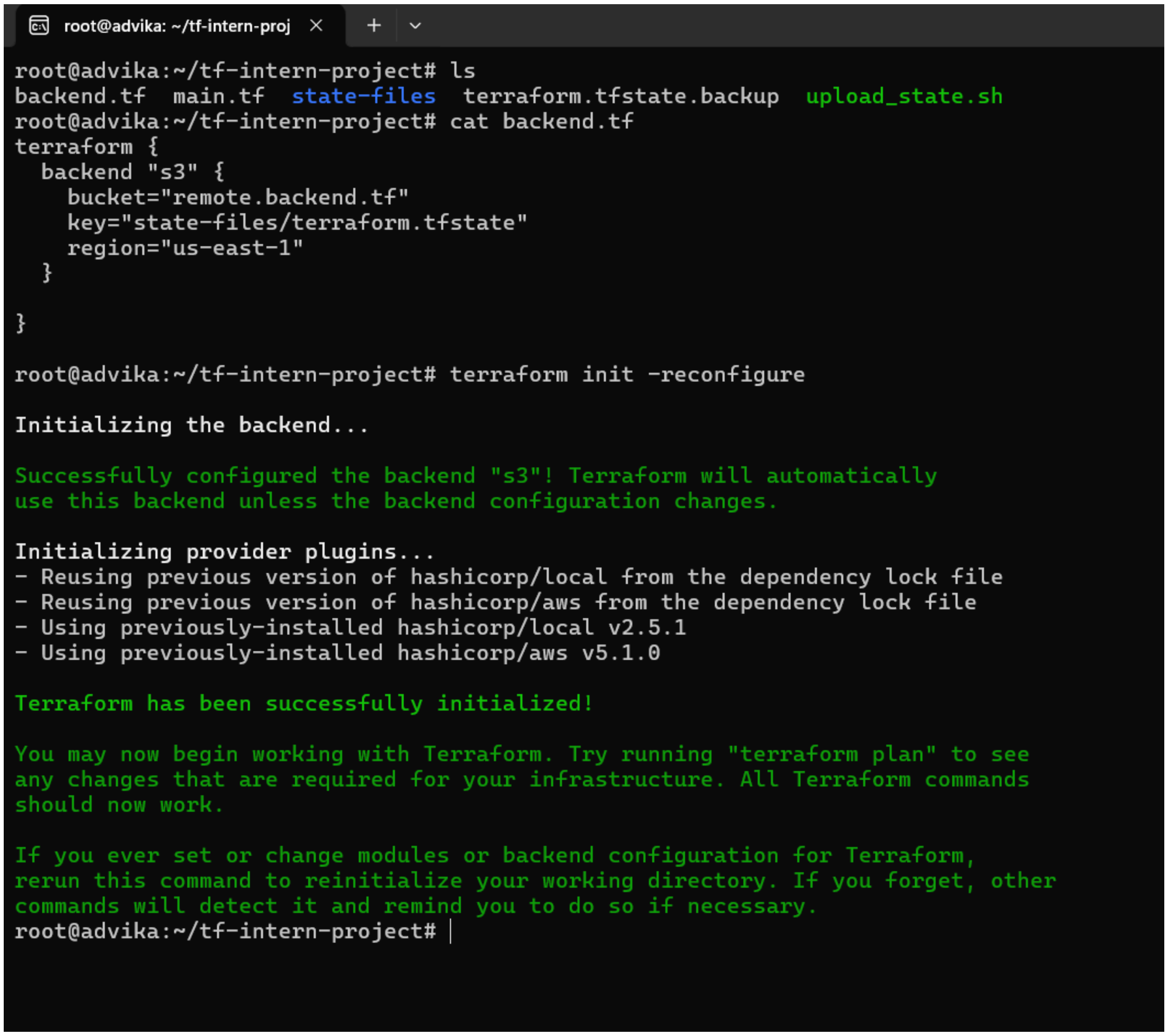


```
root@advika:~/tf-intern-project# vi upload_state.sh
root@advika:~/tf-intern-project# cat upload_state.sh
#!/bin/bash

BUCKET_NAME="remote.backend.tf"
REGION="us-east-1"
TIMESTAMP=$(date +"%Y%m%d%H%M%S")
LOCAL_STATE_FILE="terraform.tfstate"
TIMESTAMPED_STATE_FILE="state-files/${TIMESTAMP}_terraform.tfstate"
LATEST_STATE_FILE="state-files/latest_terraform.tfstate"

# Check if the local state file exists
if [ ! -f "${LOCAL_STATE_FILE}" ]; then
  echo "Error: ${LOCAL_STATE_FILE} does not exist."
  exit 1
fi

# Upload the state file with a timestamped key
echo "Uploading ${LOCAL_STATE_FILE} to s3://${BUCKET_NAME}/${TIMESTAMPED_STATE_FILE}"
aws s3 cp "${LOCAL_STATE_FILE}" "s3://${BUCKET_NAME}/${TIMESTAMPED_STATE_FILE}" --region "${REGION}"

if [ $? -ne 0 ]; then
  echo "Error: Failed to upload ${LOCAL_STATE_FILE} to s3://${BUCKET_NAME}/${TIMESTAMPED_STATE_FILE}"
  exit 1
fi

# Copy the timestamped state file to the latest key
echo "Updating latest state file to ${TIMESTAMPED_STATE_FILE}"
aws s3 cp "s3://${BUCKET_NAME}/${TIMESTAMPED_STATE_FILE}" "s3://${BUCKET_NAME}/${LATEST_STATE_FILE}" --region "${REGION}"

if [ $? -ne 0 ]; then
  echo "Error: Failed to update latest state file to s3://${BUCKET_NAME}/${LATEST_STATE_FILE}"
  exit 1
fi

echo "State file successfully uploaded and latest state file updated."

root@advika:~/tf-intern-project# 
```



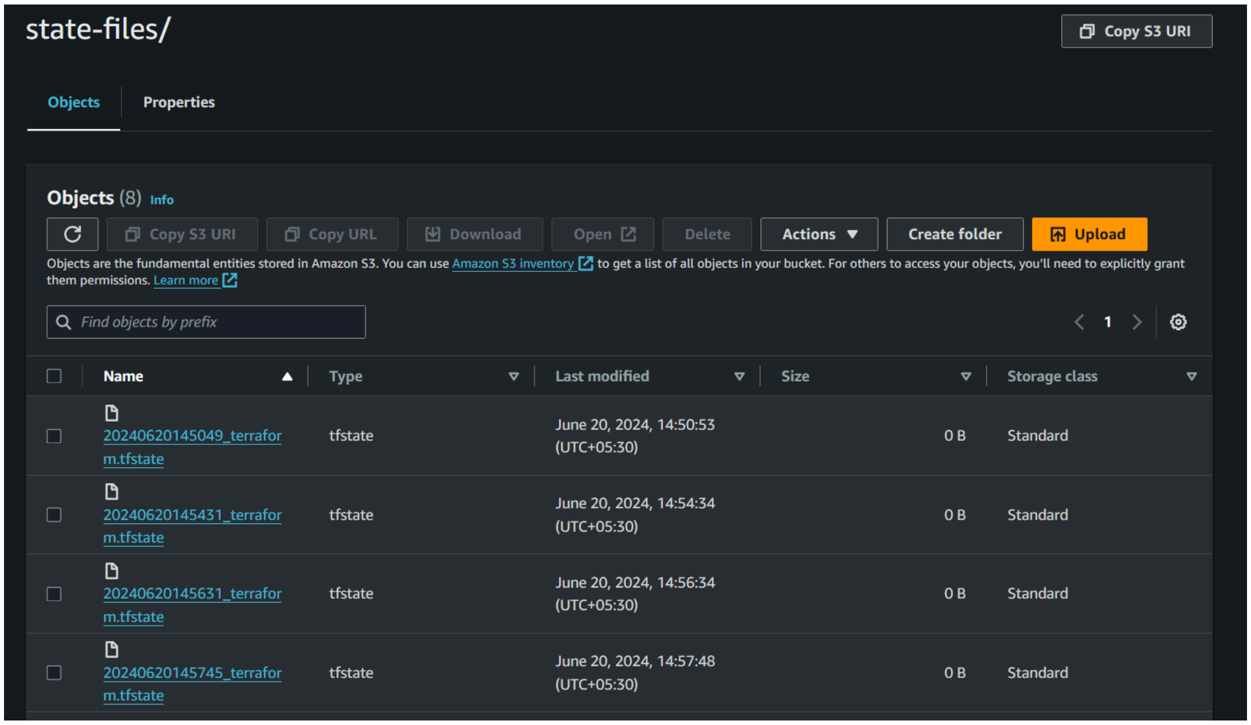
```
root@advika:~/tf-intern-project# chmod +x upload_state.sh
root@advika:~/tf-intern-project# ls upload_state.sh
upload_state.sh
root@advika:~/tf-intern-project# ls -l upload_state.sh
-rwxr-xr-x 1 root root 1217 Jun 20 14:47 upload_state.sh
root@advika:~/tf-intern-project# ./upload_state.sh
Uploading terraform.tfstate to s3://remote.backend.tf/state-files/20240620145049_terraform.tfstate
upload: ./terraform.tfstate to s3://remote.backend.tf/state-files/20240620145049_terraform.tfstate
Updating latest state file to state-files/20240620145049_terraform.tfstate
copy: s3://remote.backend.tf/state-files/20240620145049_terraform.tfstate to s3://remote.backend.tf/state-files/latest_terraform.tfstate
State file successfully uploaded and latest state file updated.
root@advika:~/tf-intern-project# 
```

I also created an alt directory on my local system that got mapped to the S3 bucket in the backend named as *state-files*



```
root@advika:~/tf-intern-project# ls
backend.tf  main.tf  state-files  terraform.tfstate.backup  upload_state.sh
root@advika:~/tf-intern-project# cat backend.tf
terraform {
  backend "s3" {
    bucket="remote.backend.tf"
    key="state-files/terraform.tfstate"
    region="us-east-1"
  }

}

root@advika:~/tf-intern-project# terraform init -reconfigure

Initializing the backend...

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

Initializing provider plugins...
- Reusing previous version of hashicorp/local from the dependency lock file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/local v2.5.1
- Using previously-installed hashicorp/aws v5.1.0

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.

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.
root@advika:~/tf-intern-project# 
```

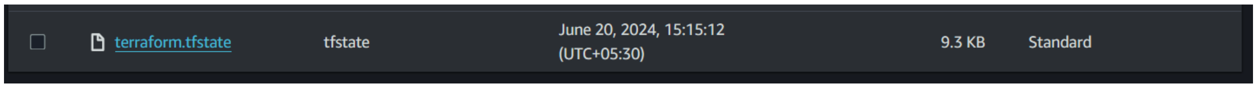
These 0 B files were generated with the timestamps as YYYYMMDDHHMMSS.



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

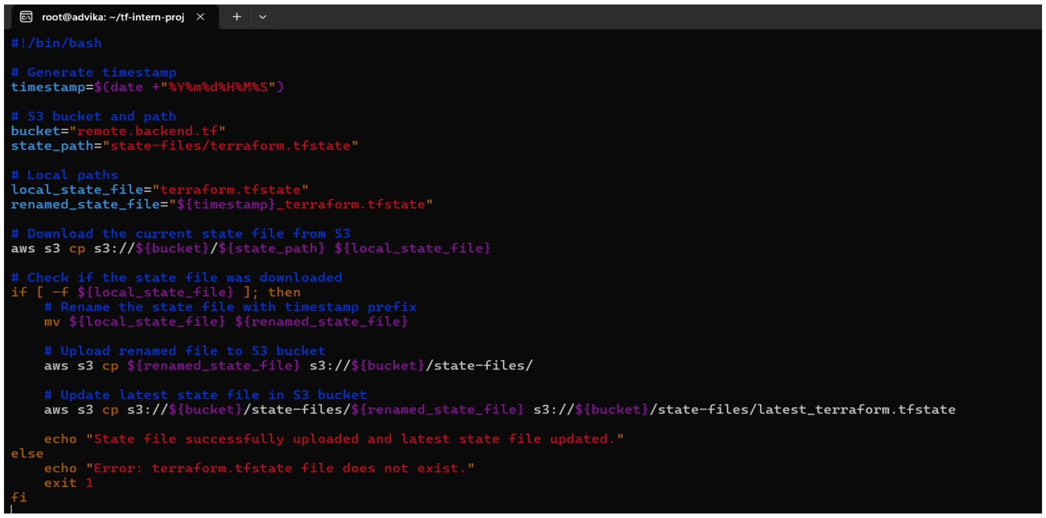
Name	Type	Last modified	Size	Storage class
20240620145049_terraform.tfstate	tfstate	June 20, 2024, 14:50:53 (UTC+05:30)	0 B	Standard
20240620145431_terraform.tfstate	tfstate	June 20, 2024, 14:54:34 (UTC+05:30)	0 B	Standard
20240620145631_terraform.tfstate	tfstate	June 20, 2024, 14:56:34 (UTC+05:30)	0 B	Standard
20240620145745_terraform.tfstate	tfstate	June 20, 2024, 14:57:48 (UTC+05:30)	0 B	Standard

On re-running the set of terraform commands, (*init, validate, plan* and *apply),* an original *terraform.tfstate* file was generated and uploaded to S3 straight without being saved to the local directory.



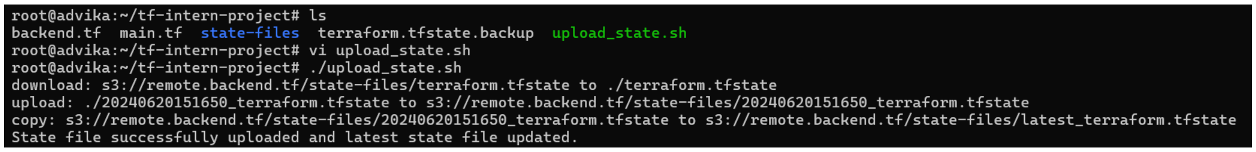
terraform.tfstate	tfstate	June 20, 2024, 15:15:12 (UTC+05:30)	9.3 KB	Standard

Terraform stored the state directly in the S3 bucket due to the backend configuration. Therefore to manipulate the state file, I downloaded it from s3, renamed it and uploaded it back.
For this, I had to modify my *upload_state.sh* bash script again.

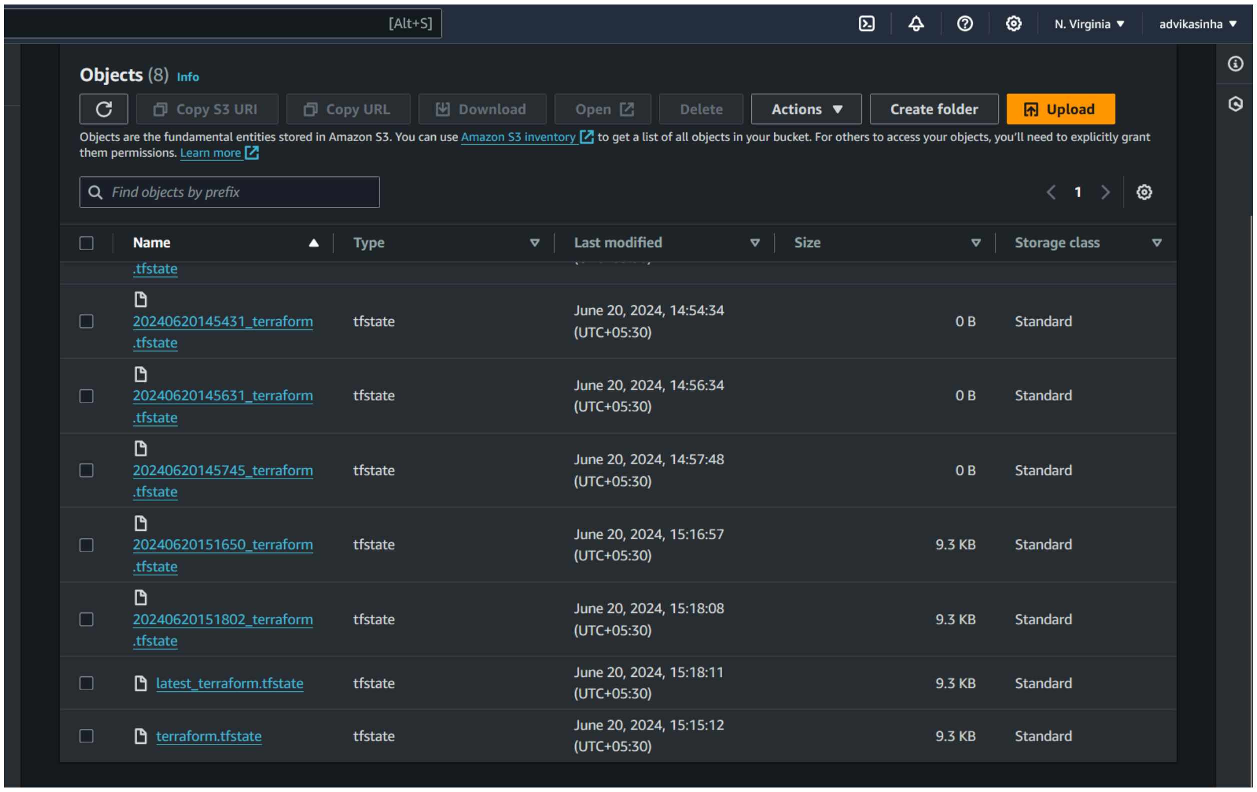


```
#!/bin/bash

# Generate timestamp
timestamp=$(date +"%Y%m%d%H%M%S")

# S3 bucket and path
bucket="remote.backend.tf"
state_path="state-files/terraform.tfstate"

# Local paths
local_state_file="terraform.tfstate"
renamed_state_file="${timestamp}_terraform.tfstate"

# Download the current state file from S3
aws s3 cp s3://${bucket}/${state_path} ${local_state_file}

# Check if the state file was downloaded
if [ -f ${local_state_file} ]; then
    # Rename the state file with timestamp prefix
    mv ${local_state_file} ${renamed_state_file}

    # Upload renamed file to S3 bucket
    aws s3 cp ${renamed_state_file} s3://${bucket}/state-files/

    # Update latest state file in S3 bucket
    aws s3 cp s3://${bucket}/state-files/${renamed_state_file} s3://${bucket}/state-files/latest_terraform.tfstate

    echo "State file successfully uploaded and latest state file updated."
else
    echo "Error: terraform.tfstate file does not exist."
    exit 1
fi
```

On running the shell script by *./upload_state.sh* command, the state files got successfully created and renamed



```
root@advika:~/tf-intern-project# ls
backend.tf  main.tf  state-files  terraform.tfstate.backup  upload_state.sh
root@advika:~/tf-intern-project# vi upload_state.sh
root@advika:~/tf-intern-project# ./upload_state.sh
download: s3://remote.backend.tf/state-files/terraform.tfstate to ./terraform.tfstate
upload: ./20240620151650_terraform.tfstate to s3://remote.backend.tf/state-files/20240620151650_terraform.tfstate
copy: s3://remote.backend.tf/state-files/20240620151650_terraform.tfstate to s3://remote.backend.tf/state-files/latest_terraform.tfstate
State file successfully uploaded and latest state file updated.
```



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
.tfstate				
20240620145431_terraform.tfstate	tfstate	June 20, 2024, 14:54:34 (UTC+05:30)	0 B	Standard
20240620145631_terraform.tfstate	tfstate	June 20, 2024, 14:56:34 (UTC+05:30)	0 B	Standard
20240620145745_terraform.tfstate	tfstate	June 20, 2024, 14:57:48 (UTC+05:30)	0 B	Standard
20240620151650_terraform.tfstate	tfstate	June 20, 2024, 15:16:57 (UTC+05:30)	9.3 KB	Standard
20240620151802_terraform.tfstate	tfstate	June 20, 2024, 15:18:08 (UTC+05:30)	9.3 KB	Standard
latest_terraform.tfstate	tfstate	June 20, 2024, 15:18:11 (UTC+05:30)	9.3 KB	Standard
terraform.tfstate	tfstate	June 20, 2024, 15:15:12 (UTC+05:30)	9.3 KB	Standard

The 9.3 KB files are the renamed, non-empty, state files created to reflect the state of the terraform object at the timestamp.