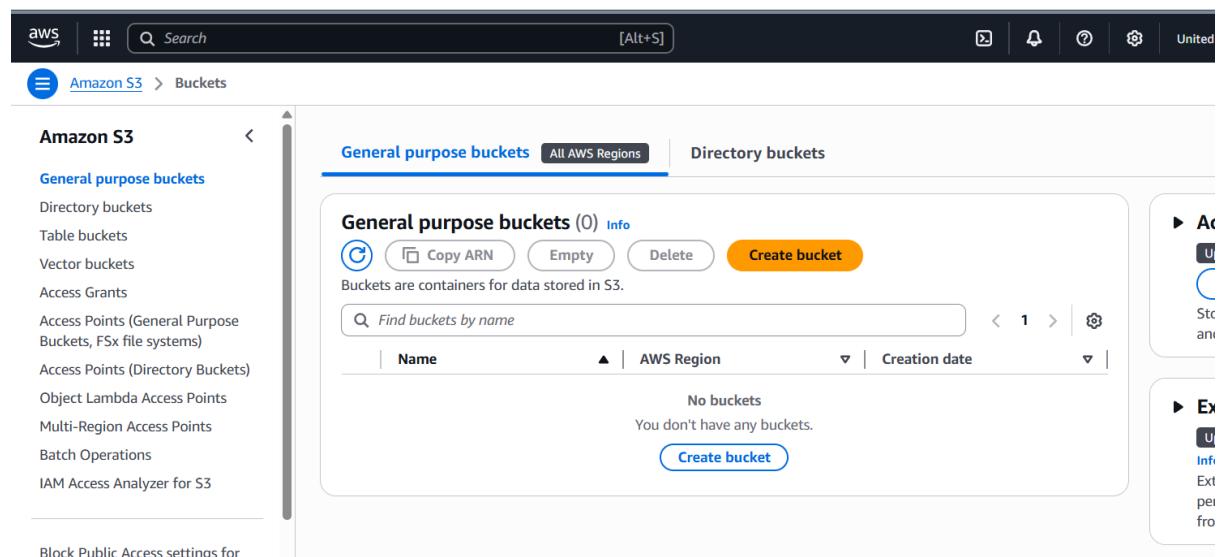


1. Watch the Terraform-05 video.

2. Execute the Script Shown in the Video

Create a s3 bucket by this script.

```
resource "aws_s3_bucket" "s3_bucket" {  
  
    bucket = "s3backend2345678"  
  
    acl = "private"  
  
}
```



The screenshot shows the AWS S3 console interface. The top navigation bar includes the AWS logo, search bar, and various icons. The main content area is titled 'General purpose buckets' with a sub-tab 'All AWS Regions'. A sub-header states 'Buckets are containers for data stored in S3.' Below this is a search bar labeled 'Find buckets by name'. A table header row includes columns for 'Name', 'AWS Region', and 'Creation date'. A large message 'No buckets' is displayed with the subtext 'You don't have any buckets.' and a prominent blue 'Create bucket' button. On the left, a sidebar lists other S3 features: 'General purpose buckets', 'Directory buckets', 'Table buckets', 'Vector buckets', 'Access Grants', 'Access Points (General Purpose Buckets, FSx file systems)', 'Access Points (Directory Buckets)', 'Object Lambda Access Points', 'Multi-Region Access Points', 'Batch Operations', and 'IAM Access Analyzer for S3'. At the bottom left, there's a link 'Block Public Access settings for'.

EXPLORER

TERRAFORM BASICS

- .terraform
- .terraform.lock.hcl
- main.tf 1, U
- terraformer.tfstate
- terraformer.tfstate.backup

main.tf > ...

```
1 resource "aws_s3_bucket" "s3_bucket" {  
2   bucket = "s3backend2345678"  
3   acl = "private"  
4 }  
5  
6
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Ashish\Desktop\Terraform basics> **terraform init**
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Reusing previous version of hashicorp/local from the dependency lock file
- Installing hashicorp/aws v6.21.0...
- Installed hashicorp/aws v6.21.0 (signed by HashiCorp)
- Using previously-installed hashicorp/local v2.5.3
Terraform has made some changes to the provider dependency selections recorded in the .terraform.lock.hcl file. Review those changes and commit them to your version control system if they represent changes you intended to make.
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 prompt you to run this command if necessary.

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Ashish\Desktop\Terraform basics> **terraform apply**

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated by colors:
+ create

Terraform will perform the following actions:

```
# aws_s3_bucket.s3_bucket will be created  
+ resource "aws_s3_bucket" "s3_bucket" {  
  + acceleration_status      = (known after apply)  
  + acl                      = "private"  
  + arn                      = (known after apply)  
  + bucket                   = "s3backend2345678"  
  + bucket_domain_name       = (known after apply)  
  + bucket_prefix             = (known after apply)  
  + bucket_region              = (known after apply)  
  + bucketRegionalDomainName = (known after apply)  
  + force_destroy            = false  
  + hosted_zone_id           = (known after apply)  
  + id                       = (known after apply)
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

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.s3_bucket: Creating...
aws_s3_bucket.s3_bucket: Creation complete after 7s [id=s3backend2345678]
```

Warning: Argument is deprecated
with aws_s3_bucket.s3_bucket,
on main.tf line 4, in resource "aws_s3_bucket" "s3_bucket":
4: acl = "private"

acl is deprecated. Use the aws_s3_bucket_acl resource instead.

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>

A bucket has been created.

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with 'Amazon S3' and a 'General purpose buckets' section containing links for Directory buckets, Table buckets, Vector buckets, Access Grants, Access Points (General Purpose Buckets, FSx file systems), Access Points (Directory Buckets), Object Lambda Access Points, Multi-Region Access Points, and Batch Operations. The main area is titled 'General purpose buckets (1/1)' and shows a table with one item:

Name	AWS Region	Creation date
s3backend2345678	US East (N. Virginia) us-east-1	November 14, 2025, 17:02:49 (UTC+05:30)

Create a dynamodb with this script.

```
resource "aws_dynamodb_table" "dynamodb-terraform-state-lock" {
    name = "terraform-state-lock-dynamo"
    hash_key = "LockID"
    read_capacity = 20
    write_capacity = 20
```

```

attribute {
  name = "LockID"
  type = "S"
}
}

```

The screenshot shows a terminal window with the following content:

```

main.tf 1, U X
main.tf > resource "aws_dynamodb_table" "dynamodb-terraform-state-lock"
1   resource "aws_s3_bucket" "s3_bucket" {
2
3     bucket = "s3backend2345678"
4     acl    = "private"
5   }
6   resource "aws_dynamodb_table" "dynamodb-terraform-state-lock" {
7     name        = "terraform-state-lock-dynamo"
8     hash_key   = "LockID"
9     read_capacity = 20
10    write_capacity = 20
11
12   attribute {
13     name = "LockID"
14     type = "S"
15   }
16
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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.
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_s3_bucket.s3_bucket: Refreshing state... [id=s3backend2345678]

Terraform used the selected providers to generate the following execution plan. Resource actions are
+ create

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
with aws_s3_bucket.s3_bucket,
on main.tf line 4, in resource "aws_s3_bucket" "s3_bucket":
  4:     acl = "private"

acl is deprecated. Use the aws_s3_bucket_acl resource instead.

(and one more similar warning elsewhere)
```

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_dynamodb_table.dynamodb-terraform-state-lock: Creating...
aws_dynamodb_table.dynamodb-terraform-state-lock: Still creating... [00m10s elapsed]
aws_dynamodb_table.dynamodb-terraform-state-lock: Creation complete after 12s [id=terraform-state-lock-dynamo]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

You can see a table has been created in dynamodb

The screenshot shows the AWS DynamoDB console. On the left, there's a navigation sidebar with options like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. The 'Tables' option is selected. The main area displays a table titled 'Tables (1/1)'. It shows one row for 'terraform-state-lock-dynamo', which is marked as 'Active' and has a LockID of 'S'. The table has columns for Name, Status, Partition key, Sort key, Indexes, Replication Regions, Deletion protection, and Favorite. There are buttons for Actions, Delete, and Create table. At the top right, it says 'Last updated November 14, 2025, 17:14 (UTC+5:30)'.

An empty table has been created.

This screenshot shows the detailed view of the 'terraform-state-lock-dynamo' table within the AWS DynamoDB console. The left sidebar remains the same. The main panel shows the table name 'terraform-state-lock-dynamo' and its status as 'Active'. Below this, there's a section titled 'Protect your DynamoDB table from accidental writes and deletes' with a note about Point-in-time recovery (PITR). The 'Settings' tab is selected, showing various configuration options: Partition key (LockID String), Sort key (-), Point-in-time recovery (PITR) (Off), Alarms (No active alarms), Average item size (0 bytes), Resource-based policy (Info), Capacity mode (Provisioned), Item count (0), Table status (Active), and Table size (0 bytes). There are also buttons for 'Edit PITR' and 'Get live item count'.

Giving s3 as backend for terraform.tf state file.

Giving this script.

```
terraform {
```

```
  backend "s3" {
```

```
    bucket = "s3backend2345678"
```

```
    dynamodb_table = "terraform-state-lock-dynamo"
```

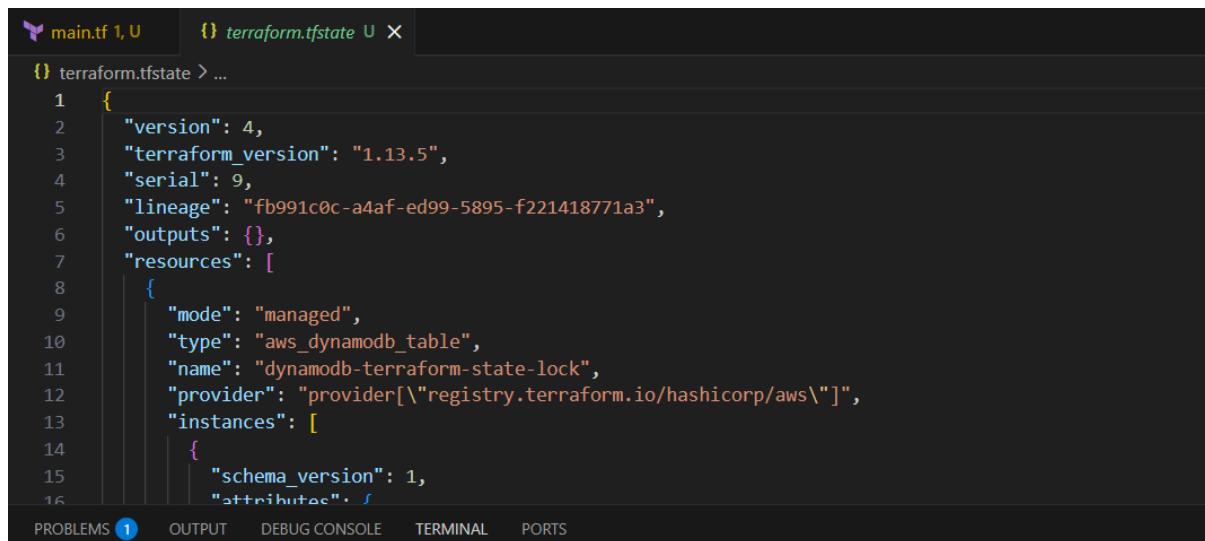
```
    key = "terraform.tfstate"
```

```
    region = "us-east-1"
```

```
}
```

```
}
```

If you open terraform state file you can see your content.



The screenshot shows a code editor interface with two tabs: 'main.tf 1, U' and 'terraform.tfstate 1, U'. The 'terraform.tfstate' tab is active, displaying the following JSON content:

```
1 {  
2   "version": 4,  
3   "terraform_version": "1.13.5",  
4   "serial": 9,  
5   "lineage": "fb991c0c-a4af-ed99-5895-f221418771a3",  
6   "outputs": {},  
7   "resources": [  
8     {  
9       "mode": "managed",  
10      "type": "aws_dynamodb_table",  
11      "name": "dynamodb-terraform-state-lock",  
12      "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",  
13      "instances": [  
14        {  
15          "schema_version": 1,  
16          "attributes": {}  
17        }  
18      ]  
19    }  
20  ]  
21}
```

At the bottom of the editor, there are tabs for 'PROBLEMS 1', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'.

After initializing you can't see your content.

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Do you want to copy existing state to the new backend?
Pre-existing state was found while migrating the previous "local" backend to the
newly configured "s3" backend. No existing state was found in the newly
configured "s3" backend. Do you want to copy this state to the new "s3"
backend? Enter "yes" to copy and "no" to start with an empty state.

Enter a value: yes

Releasing state lock. This may take a few moments...

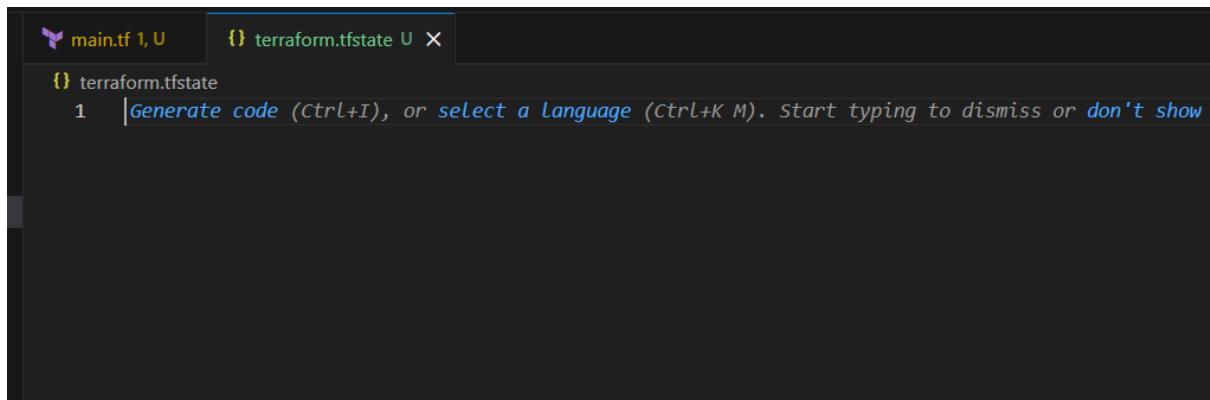
Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
```

If you open statefile.

It will be empty.



It will be stored in s3bucket go to your s3bucket and click on objects.

Our terraform statefile will be stored in s3.

The screenshot shows the AWS S3 console. The left sidebar lists 'General purpose buckets' and 'Amazon S3'. The main area shows a bucket named 's3backend2345678'. Inside, there is one object named 'terraform.tfstate' which is a tfstate file. The object details show it was last modified on November 14, 2025, at 17:25:47 (UTC+05:30) and is 4.9 KB in size, stored in the Standard storage class.

If you go to dynamodb, explore items you will see terraform statefile will be stored in your table.

The screenshot shows the AWS DynamoDB console. The left sidebar lists 'DynamoDB' features like Dashboard, Tables, and Explore items. Under Explore items, there is a table named 'terraform-state-lock-dynamo'. A single item is selected, showing fields 'LockID (String)' and 'Digest'. The item's value for LockID is 's3backend2345678/t...'. The table summary indicates 1 item returned.

If you add another resource

```
main.tf 1, U X
main.tf > terraform
17  terraform {
18    backend "s3" {
19      bucket = "s3backend2345678"
20      dynamodb_table = "terraform-state-lock-dynamodb"
21      key     = "terraform.tfstate"
22      region = "us-east-1"
23    }
24  }
25  resource "local_file" "name" {
26    filename = "pets.txt"
27    content = "I love cats"
28  }
29 }
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
with aws_s3_bucket.s3_bucket,
on main.tf line 4, in resource "aws_s3_bucket" "s3_bucket":
  4:   acl = "private"

acl is deprecated. Use the aws_s3_bucket_acl resource instead.

(and one more similar warning elsewhere)
```

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

```
local_file.name: Creating...
local_file.name: Creation complete after 0s [id=aa9f05f39211ea80c845af77b88de873f63b14af]
Releasing state lock. This may take a few moments...

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

```
main.tf 1, U    pets.txt U X
pets.txt
1 I love cats
```

terraform destroy it will delete all s3bucket,dynamodb.

Amazon S3 > Buckets

General purpose buckets

General purpose buckets (0)

No buckets

You don't have any buckets.

Create bucket

Creating an ec2 instance with terraform.

Go to e2-instance copy the ami id

EC2 > Instances > Launch an instance

Instance type

t3.micro

Family: t3 2 vCPU 1 GiB Memory Current generation: true

On-Demand Ubuntu Pro base pricing: 0.0139 USD per Hour

On-Demand SUSE base pricing: 0.0104 USD per Hour On-Demand Linux base pricing: 0.0104 USD per Hour

On-Demand RHEL base pricing: 0.0392 USD per Hour On-Demand Windows base pricing: 0.0196 USD per Hour

Key pair (login)

Can you use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Summary

Number of instances: 1

Software Image (AMI): Amazon Linux 2023 AMI 2023.9.2...read more
ami-0cae6d6fe6048ca2c

Virtual server type (instance type): t3.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

```
resource "aws_instance" "webserver" {
```

```
    ami = "ami-0cae6d6fe6048ca2c"
```

```
    instance_type = "t3.micro"
```

```
    key_name = "red"
```

```
    subnet_id = "subnet-0a192382de0e2bf6a"
```

```
    tags = {
```

```
        name = "first-server"
```

}

}

The screenshot shows a terminal window with the following content:

```
main.tf  pets.txt
main.tf > resource "aws_instance" "webserver"
1   resource "aws_instance" "webserver" {
2     ami = "ami-0cae6d6fe6048ca2c"
3     instance_type = "t3.micro"
4     key_name = "red"
5     subnet_id = "subnet-0a192382de0e2bf6a"
6     tags = {
7       name = "first-server"
8     }
9   }

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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.
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated
+ create

Terraform will perform the following actions:

# aws_instance.webserver will be created
+ resource "aws_instance" "webserver" {
  + ami                         = "ami-0cae6d6fe6048ca2c"
```

```
+ primary_network_interface (known after apply)  
+ private_dns_name_options (known after apply)  
+ root_block_device (known after apply)  
}
```

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.webserver: Creating...  
aws_instance.webserver: Still creating... [00m10s elapsed]  
aws_instance.webserver: Creation complete after 18s [id=i-0ad0218d81e082685]
```

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

PS C:\Users\Ashish\Desktop\Terraform basics>

An ec2 instance has been created here.

The screenshot shows the AWS EC2 Instances page. The left sidebar has 'EC2' selected under 'Instances'. The main area displays a table titled 'Instances (1/1) Info'. The table has columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4. One row is shown, corresponding to the instance created via Terraform, with the details: Name 'i-0ad0218d81e082685', Instance ID 'i-0ad0218d81e082685', Instance state 'Running', Instance type 't3.micro', Status check 'Initializing', Alarm status 'View alarms +', Availability Zone 'us-east-1a', and Public IPv4 listed as '-'. There are also 'Connect', 'Actions', and 'Launch instances' buttons at the top of the table.

```

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

PS C:\Users\Ashish\Desktop\Terraform basics> terraform destroy
aws_instance.webserver: Refreshing state... [id=i-0ad0218d81e082685]

Terraform used the selected providers to generate the following execution plan. Resource actions are as follows:
- destroy

Terraform will perform the following actions:

# # aws_instance.webserver will be destroyed
- resource "aws_instance" "webserver" {
    - ami                               = "ami-0cae6d6fe6048ca2c" -> null
    - arn                               = "arn:aws:ec2:us-east-1:235351028455:instance/i-0ad0218d81e082685" -> null
    - associate_public_ip_address       = false -> null
    - availability_zone                = "us-east-1a" -> null
    - disable_api_stop                 = false -> null
    - disable_api_termination          = false -> null
    - ebs_optimized                    = false -> null
    - force_destroy                     = false -> null
}

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

        # (1 unchanged attribute hidden)
    }
}

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

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_instance.webserver: Destroying... [id=i-0ad0218d81e082685]
aws_instance.webserver: Still destroying... [id=i-0ad0218d81e082685, 00m10s elapsed]
aws_instance.webserver: Still destroying... [id=i-0ad0218d81e082685, 00m20s elapsed]
aws_instance.webserver: Still destroying... [id=i-0ad0218d81e082685, 00m30s elapsed]
aws_instance.webserver: Still destroying... [id=i-0ad0218d81e082685, 00m40s elapsed]
aws_instance.webserver: Still destroying... [id=i-0ad0218d81e082685, 00m50s elapsed]
aws_instance.webserver: Still destroying... [id=i-0ad0218d81e082685, 01m00s elapsed]
aws_instance.webserver: Still destroying... [id=i-0ad0218d81e082685, 01m10s elapsed]
aws_instance.webserver: Still destroying... [id=i-0ad0218d81e082685, 01m20s elapsed]
aws_instance.webserver: Destruction complete after 1m25s

Destroy complete! Resources: 1 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>

```

```

main.tf  U X  instance_state.txt U
main.tf > resource "aws_instance" "webserver" > tags
1 resource "aws_instance" "webserver" {
2   ami           = "ami-0cae6d6fe6048ca2c"
3   instance_type = "t3.micro"
4   key_name      = "red"
5   subnet_id     = "subnet-0a192382de0e2bf6a"
6   tags = [
7     | name = "first-server"
8   ]
9   provisioner "local-exec" {
10    | command = "echo Instance ${aws_instance.webserver.public_ip} created! > instance_state.txt"
11  }

```

```
resource "aws_instance" "webserver" {  
    ami = "ami-0cae6d6fe6048ca2c"  
    instance_type = "t3.micro"  
    key_name = "red"  
    subnet_id = "subnet-0a192382de0e2bf6a"  
    tags = {  
        name = "first-server"  
    }  
    provisioner "local-exec" {  
        command = "echo Instance  
${aws_instance.webserver.public_ip} created! >  
instance_state.txt"  
    }  
}
```

The screenshot shows the Visual Studio Code interface. The left pane displays the `main.tf` file with code for creating an AWS instance named "webserver". The right pane is a terminal window showing the output of the `terraform init` command.

```

main.tf
1 resource "aws_instance" "webserver" {
2   ami           = "ami-0cae6d6fe6048ca2c"
3   instance_type = "t3.micro"
4   key_name      = "red"
5   subnet_id     = "subnet-0a192382de0e2bf6a"
6   tags          = []
7   name          = "first-server"
8 }
9 provisioner "local-exec" {
10   command = "echo Instance ${aws_instance.webserver.public_ip} created! > instance_state.txt"
11 }
12

```

TERMINAL

```

PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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.
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply

```

The screenshot shows the terminal window from the previous screenshot, continuing the `terraform apply` process. It shows the creation of a network interface, the primary network interface, private DNS name options, and root block device. A confirmation message asks if the user wants to proceed, and the user enters "yes". The process then creates the AWS instance, provisions it with local exec, and executes the echo command as specified in the script.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

+ network_interface (known after apply)
+ primary_network_interface (known after apply)
+ private_dns_name_options (known after apply)
+ root_block_device (known after apply)
}

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.webserver: Creating...
aws_instance.webserver: Still creating... [00m10s elapsed]
aws_instance.webserver: Provisioning with 'local-exec'...
aws_instance.webserver (local-exec): Executing: ["cmd" "/C" "echo Instance created! > instance_state.txt"]
aws_instance.webserver: Creation complete after 18s [id=i-0c565d291d11e9f41]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>

```

You will get a message in the file that you given in the script.

The screenshot shows the VS Code interface with the "TERRAFORM BASICS" workspace selected in the Explorer sidebar. The main editor area displays two files: "main.tf" and "instance_state.txt".

File Explorer:

- .terraform
- .terraform.lock.hcl
- instance_state.txt
- main.tf
- { } terraform.tfstate
- terraformer.tfstate.backup

Code Editor:

main.tf

```
resource "local_file" "test" {
    filename = "pets.txt"
    content = "I love cats"
    provisioner "local-exec" {
        command = "echo created the file pets.txt > file.txt"
    }
}
```

instance_state.txt

```
1 Instance created!
2
```

The screenshot shows the AWS Management Console with the EC2 Instances page open. The left navigation menu shows "Instances" selected under "EC2". The main table displays one instance.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
i-0c565d291d11e9f41	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1a	-	-

The screenshot shows the VS Code interface with the "TERRAFORM BASICS" workspace selected in the Explorer sidebar. The main editor area displays two files: "main.tf" and "file.txt".

File Explorer:

- .terraform
- .terraform.lock.hcl
- file.txt
- main.tf
- { } terraform.tfstate
- terraformer.tfstate.backup

Code Editor:

main.tf

```
resource "local_file" "test" {
    filename = "pets.txt"
    content = "I love cats"
    provisioner "local-exec" {
        command = "echo created the file pets.txt > file.txt"
    }
}
```

file.txt

```
I love cats
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Installing hashicorp/local v2.5.3...
- Installed hashicorp/local v2.5.3 (signed by HashiCorp)
Terraform has made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.

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.
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions
+ create

Terraform will perform the following actions:
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

+ content_md5      = (known after apply)
+ content_sha1     = (known after apply)
+ content_sha256   = (known after apply)
+ content_sha512   = (known after apply)
+ directory_permission = "0777"
+ file_permission   = "0777"
+ filename          = "pets.txt"
+ id                = (known after apply)
}

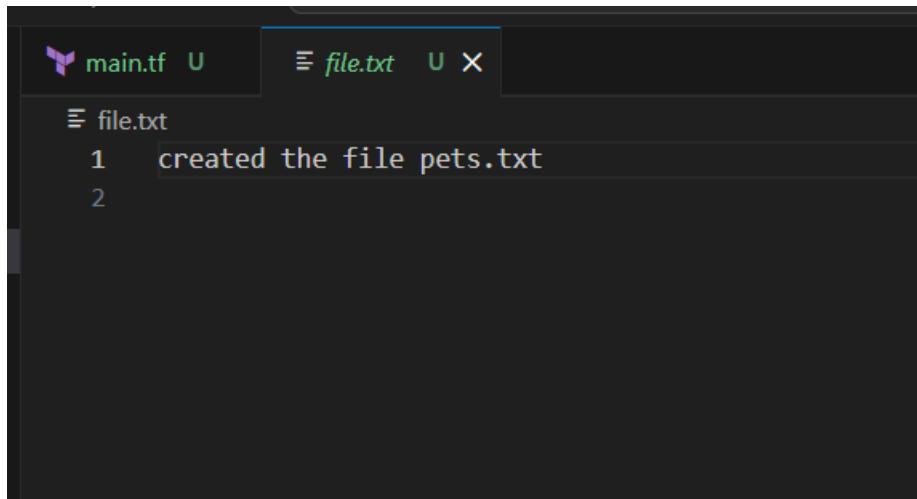
Plan: 1 to add, 0 to change, 0 to destroy.

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

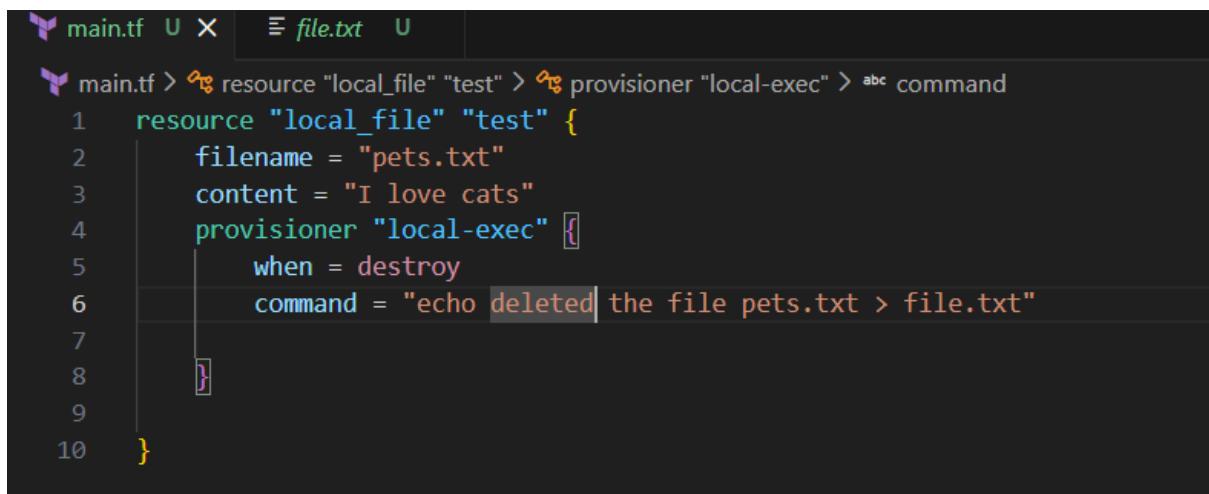
Enter a value: yes

local_file.test: Creating...
local_file.test: Provisioning with 'local-exec'...
local_file.test (local-exec): Executing: ["cmd" "/C" "echo created the file pets.txt > file.txt"]
local_file.test: Creation complete after 0s [id=aa9f05f39211ea80c845af77b88de873f63b14af]

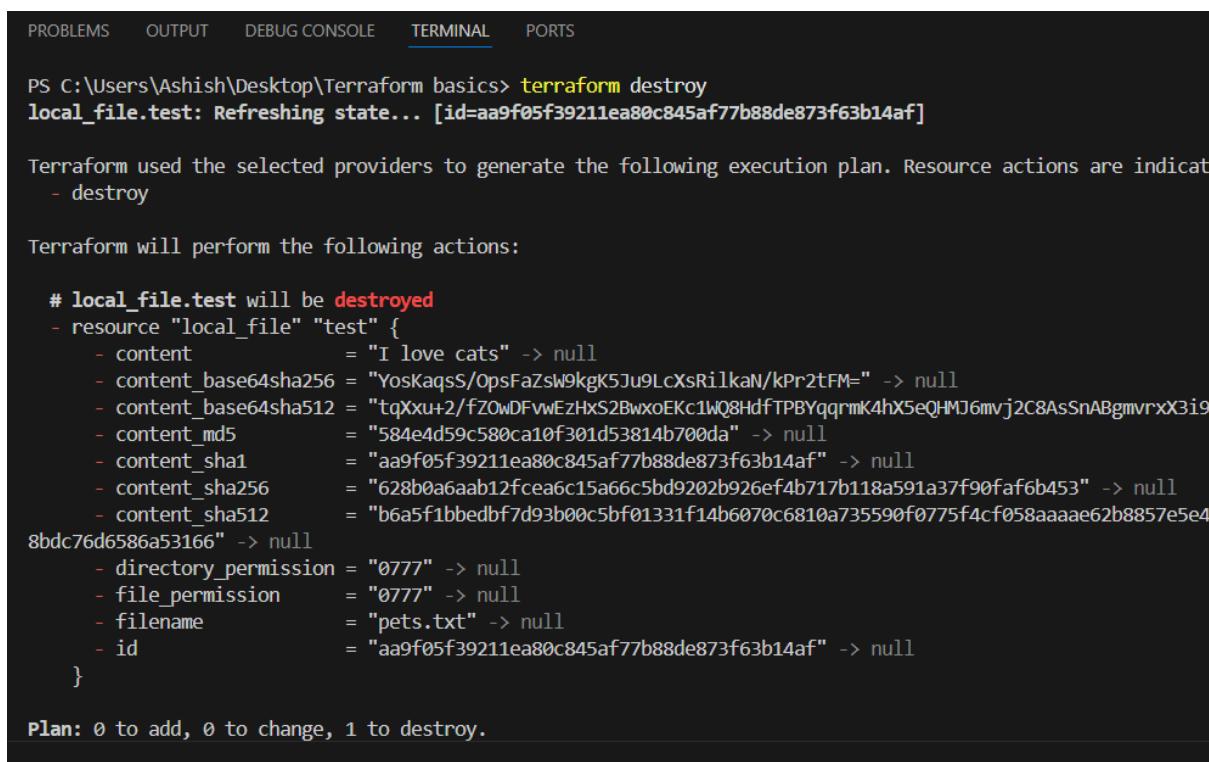
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```



```
main.tf  U   file.txt  U X
└── file.txt
    1     created the file pets.txt
    2
```



```
main.tf  U X   file.txt  U
└── main.tf >  resource "local_file" "test" >  provisioner "local-exec" > abc command
  1   resource "local_file" "test" {
  2     filename = "pets.txt"
  3     content = "I love cats"
  4     provisioner "local-exec" [
  5       when = destroy
  6       command = "echo deleted the file pets.txt > file.txt"
  7     ]
  8   }
  9
 10 }
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Ashish\Desktop\Terraform basics> terraform destroy
local_file.test: Refreshing state... [id=aa9f05f39211ea80c845af77b88de873f63b14af]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated by:
- destroy

Terraform will perform the following actions:

# local_file.test will be destroyed
- resource "local_file" "test" {
  - content          = "I love cats" -> null
  - content_base64sha256 = "YosKaqSS/OpsFaZsw9kgk5Ju9LcXsRilkaN/kPr2tFM=" -> null
  - content_base64sha512 = "tqXXu+2/fZOWDFvwEzHxS2BwxoEKc1wQ8HdFTPByqqrmK4hx5eQHMJ6mvj2C8AsSnABgmvrxx3i9
  - content_md5      = "584e4d59c580ca10f301d53814b700da" -> null
  - content_sha1     = "aa9f05f39211ea80c845af77b88de873f63b14af" -> null
  - content_sha256   = "628b0a6aab12fce46c15a66c5bd9202b926ef4b717b118a591a37f90faf6b453" -> null
  - content_sha512   = "b6a5f1bbbedbf7d93b00c5bf01331f14b6070c6810a735590f0775f4cf058aaaae62b8857e5e4
8bdc76d6586a53166" -> null
  - directory_permission = "0777" -> null
  - file_permission    = "0777" -> null
  - filename           = "pets.txt" -> null
  - id                 = "aa9f05f39211ea80c845af77b88de873f63b14af" -> null
}

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

The terminal window shows two tabs: 'main.tf' and 'file.txt'. The 'file.txt' tab contains the following text:

```
file.txt
1 deleted the file pets.txt
2
```

The terminal window shows two tabs: 'main.tf' and 'file.txt'. The 'main.tf' tab contains the following Terraform configuration:

```
main.tf
resource "local_file" "test" {
  filename = "pets.txt"
  content = "I love cats"
  provisioner "local-exec" {
    command = "echo created the file pets.txt > /tmp/file.txt"
  }
}
```

It is not created because in my pc there is no such path is available.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Ashish\Desktop\Terraform basics> **terraform apply**

Terraform used the selected providers to generate the following execution plan. Resource actions a
+ create

Terraform will perform the following actions:

```
# local_file.test will be created
+ resource "local_file" "test" {
    + content          = "I love cats"
    + content_base64sha256 = (known after apply)
    + content_base64sha512 = (known after apply)
    + content_md5      = (known after apply)
    + content_sha1     = (known after apply)
    + content_sha256   = (known after apply)
    + content_sha512   = (known after apply)
    + directory_permission = "0777"
    + file_permission   = "0777"
    + filename          = "pets.txt"
    + id                = (known after apply)
}
```

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

Do you want to perform these actions?

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

}

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

```
local_file.test: Creating...
local_file.test: Provisioning with 'local-exec'...
local_file.test (local-exec): Executing: ["cmd" "/c" "echo created the file pets.txt > /tmp/file.txt"]
local_file.test (local-exec): The system cannot find the path specified.
```

Error: local-exec provisioner error

```
with local_file.test,
on main.tf line 4, in resource "local_file" "test":
  4:   provisioner "local-exec" {
```

Error running command 'echo created the file pets.txt > /tmp/file.txt': exit status 1. Output: The

The screenshot shows a VS Code interface with two tabs open: 'main.tf' and 'file.txt'. The 'main.tf' tab contains the following Terraform code:

```
resource "local_file" "test" {
  filename = "pets.txt"
  content = "I love cats"
  provisioner "local-exec" {
    command = "echo created the file pets.txt > file.txt"
  }
}
```

The 'TERMINAL' tab shows the output of running 'terraform apply' in a Windows command prompt. It includes error messages about provisioning and a detailed execution plan for the 'local_file.test' resource.

```
on main.tf line 4, in resource "local_file" "test":
4:   provisioner "local-exec" {

Error running command 'echo created the file pets.txt > /tmp/file.txt': exit status 1. Output: The system cannot find the path specified.

PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
local_file.test: Refreshing state... [id=aa9f05f39211ea80c845af77b88de873f63b14af]

Terraform used the selected providers to generate the following execution plan. Resource actions are in
-/+ destroy and then create replacement

Terraform will perform the following actions:

# local_file.test is tainted, so must be replaced
-/+ resource "local_file" "test" {
  ~ content_base64sha256 = "YosKaqss/OpsFaZsw9kgK5Ju9LcXsRilkaN/kPr2tFM=" -> (known after apply)
  ~ content_base64sha512 = "tqXXu+2/fZOwDFvwEzHxS2BwxoEKc1WQ8HdfTPBYqqrmK4hX5eQHMJ6mvj2C8AsSnABgmvr
)
  ~ content_md5          = "584e4d59c580ca10f301d53814b700da" -> (known after apply)
  ~ content_sha1         = "aa9f05f39211ea80c845af77b88de873f63b14af" -> (known after apply)
  ~ content_sha256        = "628b0a6aab12fceaa6c15a66c5bd9202b926ef4b717b118a591a37f90faf6b453" -> (k
  ~ content_sha512        = "b6a5f1bbbedbf7d93b00c5bf01331f14b6070c6810a735590f0775f4cf058aaaae62b885
8bdc76d6586a53166" -> (known after apply)
  ~ id                   = "aa9f05f39211ea80c845af77b88de873f63b14af" -> (known after apply)
```

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

~ content_sha1          = "aa9f05f39211ea80c845af77b88de873f63b14af" -> (known after apply)
~ content_sha256         = "628b0a6aab12fce6c15a66c5bd9202b926ef4b717b118a591a37f90faf6b453" -> (known
~ content_sha512         = "b6a5f1bbbedbf7d93b00c5bf01331f14b6070c6810a735590f0775f4cf058aaaae62b8857e5e
8bcd76d6586a53166" -> (known after apply)
~ id                     = "aa9f05f39211ea80c845af77b88de873f63b14af" -> (known after apply)
# (4 unchanged attributes hidden)
}

Plan: 1 to add, 0 to change, 1 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

local_file.test: Destroying... [id=aa9f05f39211ea80c845af77b88de873f63b14af]
local_file.test: Destruction complete after 0s
local_file.test: Creating...
local_file.test: Provisioning with 'local-exec'...
local_file.test (local-exec): Executing: ["cmd" "/C" "echo created the file pets.txt > file.txt"]
local_file.test: Creation complete after 0s [id=aa9f05f39211ea80c845af77b88de873f63b14af]

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>

```

If you do terraform you will see some logs.

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Ashish\Desktop\Terraform basics> terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are
+ create

Terraform will perform the following actions:

# local_file.test will be created
+ resource "local_file" "test" {
    + content          = "I love cats"
    + content_base64sha256 = (known after apply)
    + content_base64sha512 = (known after apply)
    + content_md5      = (known after apply)
    + content_sha1     = (known after apply)
    + content_sha256   = (known after apply)
    + content_sha512   = (known after apply)
    + directory_permission = "0777"
    + file_permission   = "0777"
    + filename          = "pets.txt"
    + id                = (known after apply)
}

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

```

By passing this

- Set-Item -Path env:TF_LOG -value "TRACE"

After you will do terraform plan you will see more number of logs.

```
2025-11-14T19:48:08.376+0530 [TRACE] terraform.contextPlugins: Schema for provider "registry.terraform.io/hashicorp/local" is in the schema
2025-11-14T19:48:08.376+0530 [TRACE] AttachSchemaTransformer: attaching provider config schema to provider["registry.terraform.io/hashicorp/local"]
2025-11-14T19:48:08.376+0530 [TRACE] Executing graph transform *terraform.ModuleExpansionTransformer
2025-11-14T19:48:08.376+0530 [TRACE] Executing graph transform *terraform.ExternalReferenceTransformer
2025-11-14T19:48:08.376+0530 [TRACE] Executing graph transform *terraform.ReferenceTransformer
2025-11-14T19:48:08.376+0530 [DEBUG] ReferenceTransformer: "provider[\"registry.terraform.io/hashicorp/local\"]" references: []
2025-11-14T19:48:08.376+0530 [DEBUG] ReferenceTransformer: "local_file.test" references: []
2025-11-14T19:48:08.377+0530 [TRACE] Executing graph transform *terraform.AttachDependenciesTransformer
2025-11-14T19:48:08.377+0530 [TRACE] Executing graph transform *terraform.attachDataResourceDependsOnTransformer
2025-11-14T19:48:08.377+0530 [TRACE] Executing graph transform *terraform.DestroyEdgeTransformer
2025-11-14T19:48:08.377+0530 [TRACE] Executing graph transform *terraform.pruneUnusedNodesTransformer
2025-11-14T19:48:08.377+0530 [TRACE] Executing graph transform *terraform.TargetsTransformer
2025-11-14T19:48:08.377+0530 [TRACE] Executing graph transform *terraform.ForcedCBDTransformer
2025-11-14T19:48:08.377+0530 [TRACE] ForcedCBDTransformer: "local_file.test" (*terraform.NodeValidatableResource) has no CBD descendants
2025-11-14T19:48:08.377+0530 [TRACE] Executing graph transform *terraform.ephemeralResourceCloseTransformer
2025-11-14T19:48:08.377+0530 [TRACE] Executing graph transform *terraform.CloseProviderTransformer
2025-11-14T19:48:08.377+0530 [TRACE] Executing graph transform *terraform.CloseRootModuleTransformer
2025-11-14T19:48:08.377+0530 [TRACE] Executing graph transform *terraform.TransitiveReductionTransformer
2025-11-14T19:48:08.377+0530 [TRACE] Completed graph transform:
local_file.test - *terraform.NodeValidatableResource
  provider["registry.terraform.io/hashicorp/local"] - *terraform.NodeApplicableProvider
  provider["registry.terraform.io/hashicorp/local"] - *terraform.NodeApplicableProvider
  provider["registry.terraform.io/hashicorp/local"] (close) - *terraform.graphNodeCloseProvider
  local_file.test - *terraform.NodeValidatableResource
  root - *terraform.nodeCloseModule
  provider["registry.terraform.io/hashicorp/local"] (close) - *terraform.graphNodeCloseProvider
-----
2025-11-14T19:48:08.377+0530 [DEBUG] Starting graph walk: walkvalidate
2025-11-14T19:48:08.378+0530 [TRACE] vertex "provider[\"registry.terraform.io/hashicorp/local\"]": starting visit (*terraform.NodeApplicableProvider)
2025-11-14T19:48:08.378+0530 [TRACE] vertex "provider[\"registry.terraform.io/hashicorp/local\"]": belongs to
2025-11-14T19:48:08.378+0530 [DEBUG] created provider logger: level=trace
```

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
2025-11-14T19:48:08.965+0530 [INFO] backend/local: plan operation completed
2025-11-14T19:48:08.965+0530 [TRACE] LoadSchemas: retrieving schema for provider type "registry.terraform.io/hashicorp/local"
2025-11-14T19:48:08.965+0530 [TRACE] terraform.contextPlugins: Schema for provider "registry.terraform.io/hashicorp/local" he
2025-11-14T19:48:08.965+0530 [TRACE] LoadSchemas: retrieving schema for provisioner "local-exec"
2025-11-14T19:48:08.965+0530 [TRACE] terraform.contextPlugins: Initializing provisioner "local-exec" to read its schema

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the fol
+ create

Terraform will perform the following actions:

# local_file.test will be created
+ resource "local_file" "test" {
  + content          = "I love cats"
  + content_base64sha256 = (known after apply)
  + content_base64sha512 = (known after apply)
  + content_md5      = (known after apply)
  + content_sha1     = (known after apply)
  + content_sha256   = (known after apply)
  + content_sha512   = (known after apply)
  + directory_permission = "0777"
  + file_permission   = "0777"
  + filename         = "pets.txt"
  + id               = (known after apply)
}

Plan: 1 to add, 0 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
apply now.
2025-11-14T19:48:08.966+0530 [TRACE] statemgr.Filesystem: removing lock metadata file .terraform.tfstate.lock.info
2025-11-14T19:48:08.967+0530 [TRACE] statemgr.Filesystem: unlocked by closing terraform.tfstate
PS C:\Users\Ashish\Desktop\Terraform basics> []

```

Ln 5, Col 53 Spa

- Set-Item -Path env:TF_LOG -value "ERROR"
- terraform plan

you don't see any errors because there are no errors.

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
PS C:\Users\Ashish\Desktop\Terraform basics> Set-Item -Path env:TF_LOG -value "ERROR"
PS C:\Users\Ashish\Desktop\Terraform basics> terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the followin
+ create

Terraform will perform the following actions:

# local_file.test will be created
+ resource "local_file" "test" {
  + content          = "I love cats"
  + content_base64sha256 = (known after apply)
  + content_base64sha512 = (known after apply)
  + content_md5      = (known after apply)
  + content_sha1     = (known after apply)
  + content_sha256   = (known after apply)
  + content_sha512   = (known after apply)
  + directory_permission = "0777"
  + file_permission   = "0777"
  + filename         = "pets.txt"
  + id               = (known after apply)
}


```

If you provide error then it will show error.

The screenshot shows a code editor with two tabs: 'main.tf' and 'terraform.tfstate'. The 'main.tf' tab is active and displays the following Terraform code:

```
1 resource "local_file" "test" {
2     filename = "pets.txt"
3     content = "I love cats"
4     provisioner "local-exec" {
5         command = "echo created the file pets.txt > /tmp/file.txt"
6     }
7 }
```

Click on terraform apply you will see errors.

The screenshot shows a terminal window with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

local_file.test: Creating...
local_file.test: Provisioning with 'local-exec'...
local_file.test (local-exec): Executing: ["cmd" "/c" "echo created the file pets.txt > /tmp/file.txt"]
local_file.test (local-exec): The system cannot find the path specified.
2025-11-14T19:55:42.487+0530 [ERROR] vertex "local_file.test" error: local-exec provisioner error

Error: local-exec provisioner error

    with local_file.test,
    on main.tf line 4, in resource "local_file" "test":
    4:     provisioner "local-exec" {

Error running command 'echo created the file pets.txt > /tmp/file.txt': exit status 1. Output: The system cannot

PS C:\Users\Ashish\Desktop\Terraform basics>
```

- Set-Item -Path env:TF_LOG_PATH -value "terraform.log"
- Set-Item -Path env:TF_LOG -value "ERROR"
- terraform apply

if we have errors the it will be stored In permanent file.

main.tf U X {} terraform.tfstate U

```

main.tf > resource "local_file" "test" > provisioner "local-exec"
1   resource "local_file" "test" {
2     filename = "pets.txt"
3     content = "I love cats"
4     provisioner "local-exec" {
5       command = "echo created the file pets.txt > /tmp/file.txt"
6     }
7   }
8
9 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Ashish\Desktop\Terraform basics> Set-Item -Path env:TF_LOG_PATH -value "terraform.log"
PS C:\Users\Ashish\Desktop\Terraform basics> Set-Item -Path env:TF_LOG -value "ERROR"
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
local_file.test: Refreshing state... [id=aa9f05f39211ea80c845af77b88de873f63b14af]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated
-/+ destroy and then create replacement

Terraform will perform the following actions:

# local_file.test is tainted, so must be replaced
-/+ resource "local_file" "test" {
  ~ content_base64sha256 = "YosKqsS/OpsFaZsw9kgk5Ju9LcXsRilkaN/kPr2tFM=" -> (known after apply)
  ~ content_base64sha512 = "tqXXu+2/fZ0wDFvwEzHxS2BwxoEKc1WQ8HdfTPBYqqrmK4hX5eQHMJ6mvj2C8AsSnABgmvrX3i9x2
)
  ~ content_md5          = "584e4d59c580ca10f301d53814b700da" -> (known after apply)
  ~ content_sha1         = "aa9f05f39211ea80c845af77b88de873f63b14af" -> (known after apply)
  ~ content_sha256        = "628b0a6aab12fce46c15a66c5bd9202b926ef4b717b118a591a37f90faf6b453" -> (known af
  ~ content_sha512        = "b6a5f1bbbedbf7d93b00c5bf01331f14b6070c6810a735590f0775f4cf058aaaae62b8857e5e407
8bcd76d6586a53166" -> (known after apply)
  ~ id                   = "aa9f05f39211ea80c845af77b88de873f63b14af" -> (known after apply)
  # (4 unchanged attributes hidden)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

local_file.test: Destroying... [id=aa9f05f39211ea80c845af77b88de873f63b14af]
local_file.test: Destruction complete after 0s
local_file.test: Creating...
local_file.test: Provisioning with 'local-exec'...
local_file.test (local-exec): Executing: ["cmd" "/C" "echo created the file pets.txt > /tmp/file.txt"]
local_file.test (local-exec): The system cannot find the path specified.

Error: local-exec provisioner error

with local_file.test,
on main.tf line 4, in resource "local_file" "test":
 4:   provisioner "local-exec" {
```

```

Error running command 'echo created the file pets.txt > /tmp/file.txt': exit status 1.
```

PS C:\Users\Ashish\Desktop\Terraform basics>

```

EXPLORER
TERRAFORM BASICS
.terraform
.terraform.lock.hcl
file.txt
main.tf
pets.txt
terraform.log
terrafo rm.tfstate
terrafo rm.tfstate.backup

main.tf U
terraform.log U X

terraform.log
1 2025-11-14T19:59:41.818+0530 [ERROR] vertex "local_file.test" error: local-exec provisioner error
2

```

```

main.tf U X
main.tf > resource "local_file" "test" > provisioner "local-exec" > abc command

resource "local_file" "test" {
  filename = "pets.txt"
  content = "I love cats"
  provisioner "local-exec" {
    command = "echo created the file pets.txt > file.txt"
  }
}

```

- Set-Item -Path env:TF_LOG -value "TRACE"
- terraform apply

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Ashish\Desktop\Terraform basics> Set-Item -Path env:TF_LOG -value "TRACE"
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
local_file.test: Refreshing state... [id=aa9f05f39211ea80c845af77b88de873f63b14af]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
-/+ destroy and then create replacement

Terraform will perform the following actions:

# local_file.test is tainted, so must be replaced
-/+ resource "local_file" "test" {
  ~ content_base64sha256 = "YosKqsS/0psFaZsW9kgK5Ju9LcXsRilkaN/kPr2tFM=" -> (known after apply)
  ~ content_base64sha512 = "tqXXu+/fZ0wDFvwEzHxS2BwxoEKc1WQ8HdfTPBYqqrmK4hx5eQHMJ6mvj2C8AsSnABgmrxx3i9x21lhquxzg
)
  ~ content_md5      = "584e4d59c580ca10f301d53814b700da" -> (known after apply)
  ~ content_sha1     = "aa9f05f39211ea80c845af77b88de873f63b14af" -> (known after apply)
  ~ content_sha256   = "628b0a6aab12fce46c15a66c5bd9202b926ef4b717b118a591a37f90faf6b453" -> (known after apply)
  ~ content_sha512   = "b6a5f1bbbedbf7d93b00c5bf01331f14b6070c6810a735590f0775f4cf058aaaae62b8857e5e407309ea6be
8bcd76d6586a53166" -> (known after apply)
  ~ id               = "aa9f05f39211ea80c845af77b88de873f63b14af" -> (known after apply)
  # (4 unchanged attributes hidden)
}

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

Do you want to perform these actions?

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

~ content_md5      = "584e4d59c580ca10f301d53814b700da" -> (known after apply)
~ content_sha1    = "aa9f05f39211ea80c845af77b88de873f63b14af" -> (known after apply)
~ content_sha256   = "628b0a6aab12fceae6c15a66c5bd9202b926ef4b717b118a591a37f90faf6b453" -> (known after apply)
~ content_sha512   = "b6a5f1bbebf7d93b00c5bf01331f14b6070c6810a735590f0775f4cf058aaaae62b8857e5e4073098bdc76d6586a53166" -> (known after apply)
~ id              = "aa9f05f39211ea80c845af77b88de873f63b14af" -> (known after apply)
# (4 unchanged attributes hidden)
}

Plan: 1 to add, 0 to change, 1 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

local_file.test: Destroying... [id=aa9f05f39211ea80c845af77b88de873f63b14af]
local_file.test: Destruction complete after 0s
local_file.test: Creating...
local_file.test: Provisioning with 'local-exec'...
local_file.test (local-exec): Executing: ["cmd" "/C" "echo created the file pets.txt > file.txt"]
local_file.test: Creation complete after 0s [id=aa9f05f39211ea80c845af77b88de873f63b14af]

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics> 
```

Ln 2

```

main.tf  terraform.log  ...
terraform.log
1 2025-11-14T19:59:41.818+0530 [ERROR] vertex "local_file.test" error: local-exec provisioner error
2 2025-11-14T20:04:48.545+0530 [INFO] Terraform version: 1.13.5
3 2025-11-14T20:04:48.546+0530 [DEBUG] using github.com/hashicorp/go-tfe v1.74.1
4 2025-11-14T20:04:48.546+0530 [DEBUG] using github.com/hashicorp/hcl/v2 v2.24.0
5 2025-11-14T20:04:48.546+0530 [DEBUG] using github.com/hashicorp/terraform-svchost v0.1.1
6 2025-11-14T20:04:48.546+0530 [DEBUG] using github.com/zclconf/go-cty v1.16.3
7 2025-11-14T20:04:48.546+0530 [INFO] Go runtime version: go1.24.5
8 2025-11-14T20:04:48.546+0530 [INFO] CLI args: []string{"E:\\Terraform\\terraform.exe", "apply"}
9 2025-11-14T20:04:48.556+0530 [TRACE] Stdout is a terminal of width 143
10 2025-11-14T20:04:48.556+0530 [TRACE] Stderr is a terminal of width 143
11 2025-11-14T20:04:48.556+0530 [TRACE] Stdin is a terminal
12 2025-11-14T20:04:48.560+0530 [DEBUG] Attempting to open CLI config file: C:\\Users\\Ashish\\AppData\\Roaming\\terraform.rc
13 2025-11-14T20:04:48.560+0530 [DEBUG] File doesn't exist, but doesn't need to. Ignoring.
14 2025-11-14T20:04:48.564+0530 [DEBUG] ignoring non-existing provider search directory terraform.d/plugins
15 2025-11-14T20:04:48.564+0530 [DEBUG] ignoring non-existing provider search directory C:\\Users\\Ashish\\AppData\\Roaming\\terraform.d\\plugins
16 2025-11-14T20:04:48.564+0530 [DEBUG] ignoring non-existing provider search directory C:\\Users\\Ashish\\AppData\\Roaming\\Hashicorp\\Terra
17 2025-11-14T20:04:48.562+0530 [INFO] CLI command args: []string{"apply"}
18 2025-11-14T20:04:48.564+0530 [TRACE] Meta.Backend: no config given or present on disk, so returning nil config
19 2025-11-14T20:04:48.564+0530 [TRACE] Meta.Backend: backend has not previously been initialized in this working directory
20 2025-11-14T20:04:48.564+0530 [TRACE] Meta.Backend: using default local state only (no backend configuration, and no existing initiali
21 2025-11-14T20:04:48.564+0530 [TRACE] Meta.Backend: instantiated backend of type <nil>
22 2025-11-14T20:04:48.565+0530 [TRACE] providercache.fillMetaCache: scanning directory .terraform\\providers
23 2025-11-14T20:04:48.567+0530 [TRACE] getproviders.SearchLocalDirectory: found registry.terraform.io\\hashicorp\\aws v6.21.0 for windows
24 2025-11-14T20:04:48.568+0530 [TRACE] getproviders.SearchLocalDirectory: found registry.terraform.io\\hashicorp\\local v2.5.3 for windows
25 2025-11-14T20:04:48.568+0530 [TRACE] providercache.fillMetaCache: including .terraform\\providers\\registry.terraform.io\\hashicorp\\aws\\v6.21.0
26 2025-11-14T20:04:48.568+0530 [TRACE] providercache.fillMetaCache: including .terraform\\providers\\registry.terraform.io\\hashicorp\\local\\v2.5.3
27 2025-11-14T20:04:48.626+0530 [DEBUG] checking for provisioner in "."
28 2025-11-14T20:04:48.626+0530 [DEBUG] checking for provisioner in "E:\\Terraform"
29 2025-11-14T20:04:48.626+0530 [TRACE] Meta.Backend: backend <nil> does not support operations, so wrapping it in a local backend
30 2025-11-14T20:04:48.630+0530 [INFO] backend/local: starting Apply operation
31 2025-11-14T20:04:48.630+0530 [TRACE] backend/local: requesting state manager for workspace "default"
32 2025-11-14T20:04:48.630+0530 [TRACE] backend/local: state manager for workspace "default" will:
33  - read initial snapshot from terraform.tfstate
34  - write new snapshots to terraform.tfstate
35  - create any backup at terraform.tfstate.backup
36 2025-11-14T20:04:48.630+0530 [INFO] Backend local successfully locked for updates. "https://..."
```

Create an instance.

```

main.tf
1 resource "aws_instance" "webserver" {
2     ami = "ami-0cae6d6fe6048ca2c"
3     instance_type = "t3.micro"
4     key_name = "red"
5     subnet_id = "subnet-0a192382de0e2bf6a"
6     tags = {
7         name = "first-server"
8     }
9 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.21.0...
- Installed hashicorp/aws v6.21.0 (signed by HashiCorp)
Terraform has made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.

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.

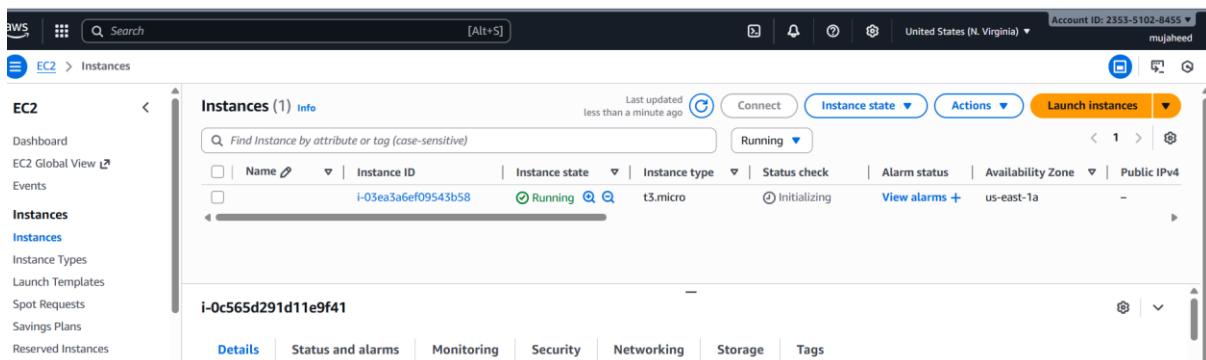
If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated
+ create

Terraform will perform the following actions:

# aws_instance.webserver will be created

```



- `terraform import aws_instance.<name> instance_id`

```
https://developer.hashicorp.com/terraform/tutorials/resource-addressing  
PS C:\Users\Ashish\Desktop\Terraform basics> terraform import aws_instance.manual i-0c565d291d11e9f41  
>>  
Error: resource address "aws_instance.manual" does not exist in the configuration.
```

Before importing this resource, please create its configuration in the root module. For example:

```
resource "aws_instance" "manual" {  
    # (resource arguments)  
}
```

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform import aws_instance.manual i-0c565d291d11e9f41  
>>
```

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init  
Initializing the backend...  
Initializing provider plugins...  
- Reusing previous version of hashicorp/aws from the dependency lock file  
- Using previously-installed hashicorp/aws v6.21.0
```

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.

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform import aws_instance.manual i-03ea3a6ef09543b58  
aws_instance.manual: Importing from ID "i-03ea3a6ef09543b58"...  
aws_instance.manual: Import prepared!  
  Prepared aws_instance for import  
aws_instance.manual: Refreshing state... [id=i-03ea3a6ef09543b58]
```

- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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.

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform import aws_instance.manual i-03ea3a6ef09543b58  
aws_instance.manual: Importing from ID "i-03ea3a6ef09543b58"...  
aws_instance.manual: Import prepared!  
  Prepared aws_instance for import  
aws_instance.manual: Refreshing state... [id=i-03ea3a6ef09543b58]
```

Import successful!

The resources that were imported are shown above. These resources are now in your Terraform state and will henceforth be managed by Terraform.

```
PS C:\Users\Ashish\Desktop\Terraform basics>
```

If you do `terraform destroy` instance will be deleted.

```

PS C:\Users\Ashish\Desktop\Terraform basics> terraform destroy
aws_instance.webserver: Refreshing state... [id=i-03ea3a6ef09543b58]
aws_instance.manual: Refreshing state... [id=i-03ea3a6ef09543b58]

Terraform used the selected providers to generate the following execution plan. Resource actions:
- destroy

Terraform will perform the following actions:

# aws_instance.manual will be destroyed
- resource "aws_instance" "manual" {
    - ami                               = "ami-0cae6d6fe6048ca2c" -> null
    - arn                               = "arn:aws:ec2:us-east-1:235351028455:instance/i-03ea3a6ef09543b58" -> null
    - associate_public_ip_address      = false -> null
    - availability_zone                = "us-east-1a" -> null
    - disable_api_stop                 = false -> null
    - disable_api_termination          = false -> null
    - ebs_optimized                    = false -> null
    - force_destroy                     = false -> null
    - get_password_data                = false -> null
}

```

Plan: 0 to add, 0 to change, 2 to destroy.

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

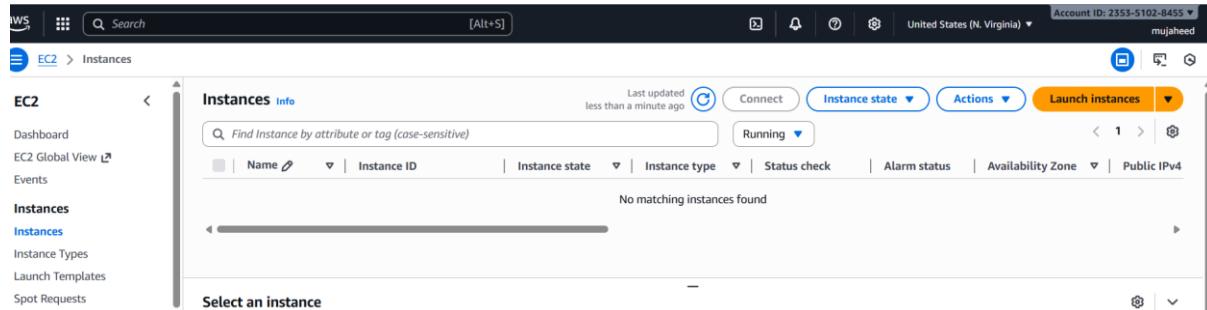
```

aws_instance.manual: Destroying... [id=i-03ea3a6ef09543b58]
aws_instance.webserver: Destroying... [id=i-03ea3a6ef09543b58]
aws_instance.manual: Still destroying... [id=i-03ea3a6ef09543b58, 00m10s elapsed]
aws_instance.webserver: Still destroying... [id=i-03ea3a6ef09543b58, 00m10s elapsed]
aws_instance.webserver: Still destroying... [id=i-03ea3a6ef09543b58, 00m20s elapsed]
aws_instance.manual: Still destroying... [id=i-03ea3a6ef09543b58, 00m20s elapsed]
aws_instance.manual: Still destroying... [id=i-03ea3a6ef09543b58, 00m30s elapsed]
aws_instance.webserver: Still destroying... [id=i-03ea3a6ef09543b58, 00m30s elapsed]
aws_instance.manual: Still destroying... [id=i-03ea3a6ef09543b58, 00m40s elapsed]
aws_instance.webserver: Still destroying... [id=i-03ea3a6ef09543b58, 00m40s elapsed]
aws_instance.webserver: Still destroying... [id=i-03ea3a6ef09543b58, 00m50s elapsed]
aws_instance.manual: Still destroying... [id=i-03ea3a6ef09543b58, 00m50s elapsed]
aws_instance.manual: Destruction complete after 53s
aws_instance.webserver: Destruction complete after 54s

```

Destroy complete! Resources: 2 destroyed.

PS C:\Users\Ashish\Desktop\Terraform basics>

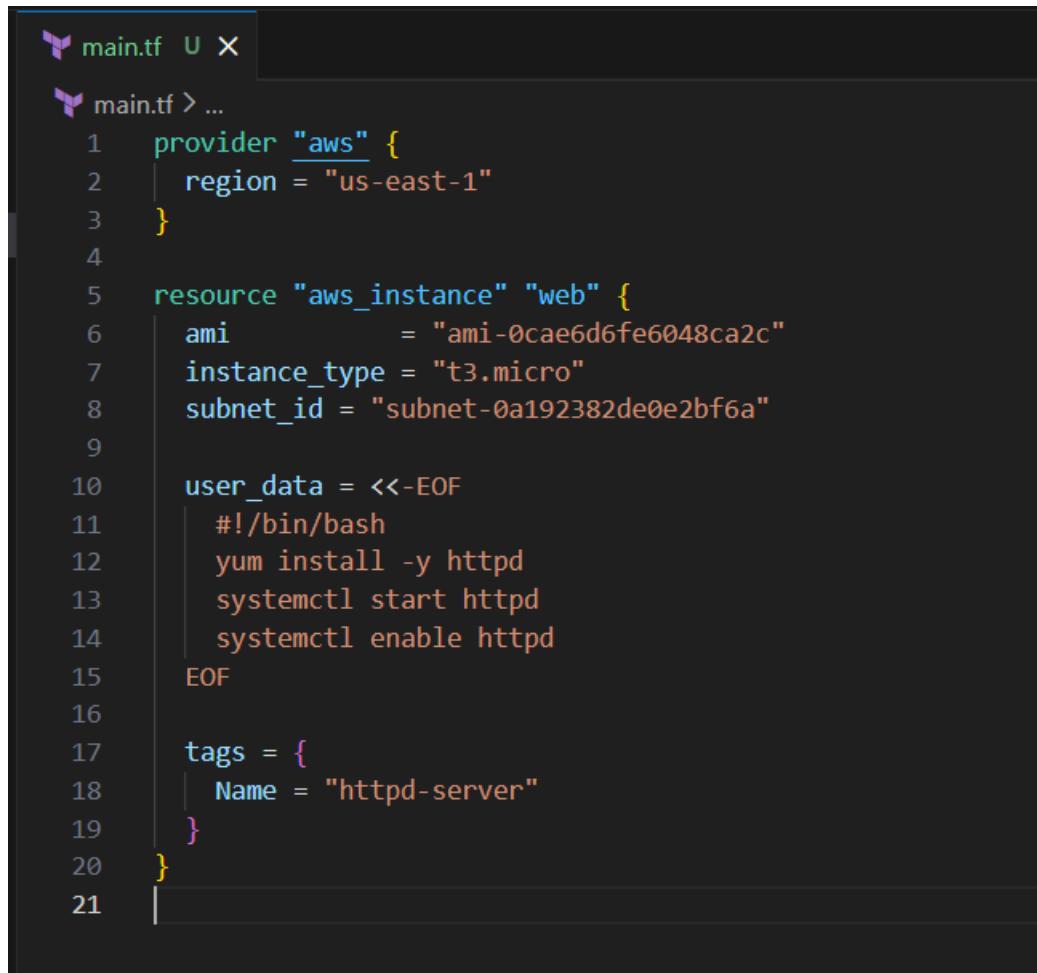


3. Create one EC2 instance with httpd installed using a Terraform script.

```
provider "aws" {  
    region = "us-east-1"  
}  
  
resource "aws_instance" "web" {  
    ami          = "ami-0cae6d6fe6048ca2c"  
    instance_type = "t3.micro"  
    subnet_id = "subnet-0a192382de0e2bf6a"  
  
    user_data = <<-EOF  
        #!/bin/bash  
        yum install -y httpd  
        systemctl start httpd  
        systemctl enable httpd  
    EOF  
  
    tags = {  
        Name = "httpd-server"  
    }  
}
```

}

- terraform init
- terraform apply



The screenshot shows a code editor window with a dark theme. The title bar says "main.tf" with a "U" icon and a close button. Below the title bar, there's a navigation bar with a tree icon and "main.tf > ...". The main editor area contains 21 numbered lines of Terraform code:

```
1 provider "aws" {
2   region = "us-east-1"
3 }
4
5 resource "aws_instance" "web" {
6   ami           = "ami-0cae6d6fe6048ca2c"
7   instance_type = "t3.micro"
8   subnet_id    = "subnet-0a192382de0e2bf6a"
9
10  user_data = <<-EOF
11    #!/bin/bash
12    yum install -y httpd
13    systemctl start httpd
14    systemctl enable httpd
15  EOF
16
17  tags = {
18    Name = "httpd-server"
19  }
20}
21
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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.
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply

Terraform used the selected providers to generate the following execution plan. Resources:
+ create

Terraform will perform the following actions:

# aws_instance.web will be created
+ resource "aws_instance" "web" {
    + ami                                     = "ami-0cae6d6fe6048ca2c"
    + arn                                     = (known after apply)
    + associate_public_ip_address             = (known after apply)
    + availability_zone                      = (known after apply)
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

+ root_block_device (known after apply)
+ private_dns_name_options (known after apply)

+ root_block_device (known after apply)
}

+ root_block_device (known after apply)
}

}

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

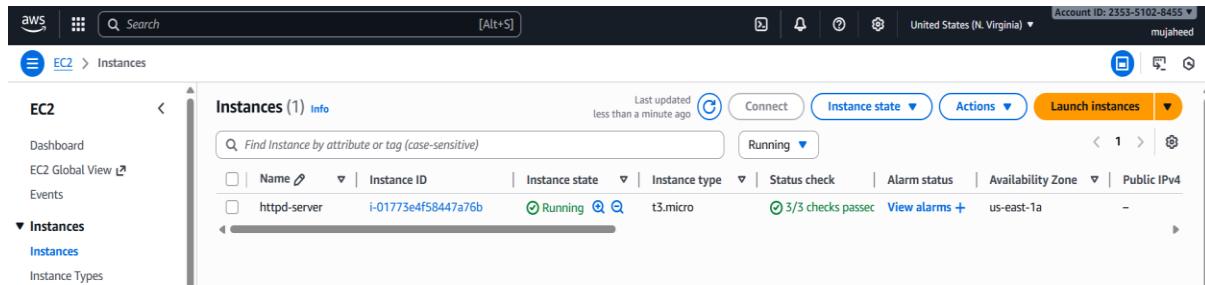
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_instance.web: Creating...
aws_instance.web: Still creating... [00m10s elapsed]
aws_instance.web: Creation complete after 17s [id=i-01773e4f58447a76b]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics> []
```

An instance will be created by using terraform with the userdata httpd init.



The screenshot shows the AWS EC2 Instances page. The left sidebar has 'EC2' selected under 'Instances'. The main area shows a table titled 'Instances (1) Info' with one row. The row details a single instance named 'httpd-server' with Instance ID 'i-01773e4f58447a76b', which is 'Running'. It is a 't3.micro' instance in 'us-east-1a' availability zone. A status check shows '3/3 checks passed'. There are buttons for 'Connect', 'Actions', and 'Launch instances' at the top right of the table.

4. Set up S3 as backend for task 3.

By adding resource to that script we can create a s3 bucket as backend.

```
provider "aws" {  
    region = "us-east-1"  
  
}  
  
resource "aws_instance" "web" {  
    ami      = "ami-0cae6d6fe6048ca2c"  
    instance_type = "t3.micro"  
    subnet_id = "subnet-0a192382de0e2bf6a"  
  
    user_data = <<-EOF  
    #!/bin/bash  
    yum install -y httpd
```

```
systemctl start httpd
systemctl enable httpd
EOF
```

```
tags = {
    Name = "httpd-server"
}
}

resource "aws_s3_bucket" "s3_bucket" {
    bucket = "s3backend234567812"
    acl = "private"
}
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0
```

```
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.
```

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_instance.web: Refreshing state... [id=i-01773e4f58447a76b]
```

```
Terraform used the selected providers to generate the following execution plan.
+ create
```

```
  Terraform will perform the following actions:
```

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Enter a value: yes

aws_s3_bucket.s3_bucket: Creating...
aws_s3_bucket.s3_bucket: Creation complete after 6s [id=s3backend234567812]
|
aws_s3_bucket.s3_bucket: Creating...
aws_s3_bucket.s3_bucket: Creation complete after 6s [id=s3backend234567812]

aws_s3_bucket.s3_bucket: Creating...
aws_s3_bucket.s3_bucket: Creation complete after 6s [id=s3backend234567812]

Warning: Argument is deprecated
with aws_s3_bucket.s3_bucket,
on main.tf line 24, in resource "aws_s3_bucket" "s3_bucket":
24:   acl = "private"

acl is deprecated. Use the aws_s3_bucket_acl resource instead.

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics> 
```

A bucket has been created as backend for that httpd server.

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with navigation links for Amazon S3, General purpose buckets, Directory buckets, Table buckets, Vector buckets, Access Grants, Access Points (General Purpose Buckets, FSx file systems), Access Points (Directory Buckets), Object Lambda Access Points, Multi-Region Access Points, Batch Operations, and IAM Access Analyzer for S3. The main content area is titled 'General purpose buckets' and shows one bucket named 's3backend234567812'. Below the bucket name are buttons for 'Copy ARN', 'Empty', 'Delete', and 'Create bucket'. A note says 'Buckets are containers for data stored in S3.' At the bottom, there's a table with columns for Name, AWS Region, and Creation date, showing the details for the single bucket listed.

Name	AWS Region	Creation date
s3backend234567812	US East (N. Virginia) us-east-1	November 15, 2025, 10:02:47 (UTC+05:30)

5. Set up DynamoDB locking for task 3.

Creating dynamodb for the httpd and s3 backend.

```
provider "aws" {
```

```
  region = "us-east-1"
```

```
}
```

```
resource "aws_instance" "web" {
```

```
  ami      = "ami-0cae6d6fe6048ca2c"
```

```
  instance_type = "t3.micro"
```

```
  subnet_id = "subnet-0a192382de0e2bf6a"
```

```
  user_data = <<-EOF
```

```
#!/bin/bash
```

```
yum install -y httpd
```

```
systemctl start httpd
```

```
systemctl enable httpd
```

```
EOF
```

```
tags = {
```

```
  Name = "httpd-server"
```

```
}
```

```
}
```

```
resource "aws_s3_bucket" "s3_bucket" {
```

```
  bucket = "s3backend234567812"
```

```
  acl = "private"
```

```
}
```

```
resource "aws_dynamodb_table" "dynamodb-terraform-state-lock" {
```

```
  name = "terraform-state-lock-dynamo"
```

```
  hash_key = "LockID"
```

```
  read_capacity = 20
```

```
  write_capacity = 20
```

```
attribute {
```

```
  name = "LockID"
```

```
  type = "S"
```

```
}
```

```
}
```

```
main.tf 1, U X
main.tf > ...
21   resource "aws_s3_bucket" "s3_bucket" {
25   }
26   resource "aws_dynamodb_table" "dynamodb-terraform-state-lock" {
27     name = "terraform-state-lock-dynamo"
28     hash_key = "LockID"
29     read_capacity = 20
30     write_capacity = 20
31
32     attribute {
33       name = "LockID"
34       type = "S"
35     }
36 }

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Ashish\Desktop\Terraform basics>
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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.
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_s3_bucket.s3_bucket: Refreshing state... [id=s3backend234567812]
aws_instance.web: Refreshing state... [id=i-01773e4f58447a76b]

Terraform used the selected providers to generate the following execution plan. Resource actions are
+ create
```

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

with aws_s3_bucket.s3_bucket,
on main.tf line 24, in resource "aws_s3_bucket" "s3_bucket":
24:   acl = "private"

acl is deprecated. Use the aws_s3_bucket_acl resource instead.

(and one more similar warning elsewhere)

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_dynamodb_table.dynamodb-terraform-state-lock: Creating...
aws_dynamodb_table.dynamodb-terraform-state-lock: Still creating... [00m10s elapsed]
aws_dynamodb_table.dynamodb-terraform-state-lock: Creation complete after 18s [id=terrafo

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

A dynamodb has been created.

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

with aws_s3_bucket.s3_bucket,
on main.tf line 24, in resource "aws_s3_bucket" "s3_bucket":
24:     acl = "private"

acl is deprecated. Use the aws_s3_bucket_acl resource instead.

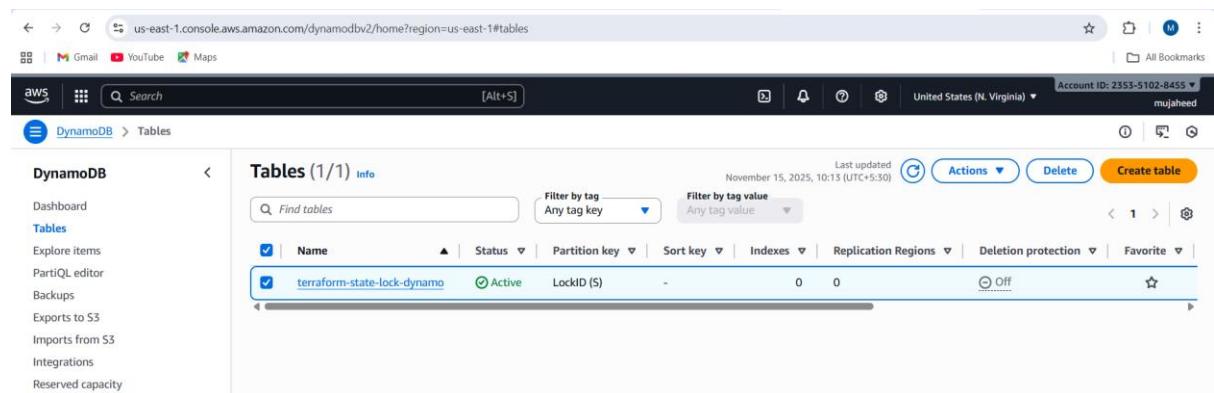
(and one more similar warning elsewhere)

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_dynamodb_table.dynamodb-terraform-state-lock: Creating...
aws_dynamodb_table.dynamodb-terraform-state-lock: Still creating... [00m10s elapsed]
aws_dynamodb_table.dynamodb-terraform-state-lock: Creation complete after 18s [id=terraform

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics> 
```



The screenshot shows the AWS DynamoDB console interface. On the left, there's a sidebar with options like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, and Reserved capacity. The main area is titled 'Tables (1/1) Info'. It displays a table with one row. The row details are: Name (terraform-state-lock-dynamo), Status (Active), Partition key (LockID (\$)), Sort key (-), Indexes (0), Replication Regions (0), Deletion protection (Off), and Favorite (star icon). There are also 'Actions' and 'Delete' buttons at the top right of the table view.

Giving s3 as backend for terraform statefile.tf

```
provider "aws" {
```

```
    region = "us-east-1"
```

```
}
```

```
resource "aws_instance" "web" {  
    ami      = "ami-0cae6d6fe6048ca2c"  
    instance_type = "t3.micro"  
    subnet_id = "subnet-0a192382de0e2bf6a"
```

```
user_data = <<-EOF  
#!/bin/bash  
yum install -y httpd  
systemctl start httpd  
systemctl enable httpd  
EOF
```

```
tags = {  
    Name = "httpd-server"  
}  
}  
  
resource "aws_s3_bucket" "s3_bucket" {  
  
    bucket = "s3backend234567812"  
    acl = "private"  
}
```

```
resource "aws_dynamodb_table" "dynamodb-terraform-state-lock" {  
    name = "terraform-state-lock-dynamo"  
    hash_key = "LockID"  
    read_capacity = 20  
    write_capacity = 20  
  
    attribute {  
        name = "LockID"  
        type = "S"  
    }  
}  
  
terraform {  
    backend "s3" {  
        bucket = "s3backend234567812"  
        dynamodb_table = "terraform-state-lock-dynamo"  
        key   = "terraform.tfstate"  
        region = "us-east-1"  
    }  
}
```

The screenshot shows a VS Code interface with two tabs open: 'main.tf' and 'terraform.tfstate'. The code editor displays Terraform configuration for an AWS DynamoDB table and its state backend.

```
main.tf
26 resource "aws_dynamodb_table" "dynamodb-terraform-state-lock" {
27   ...
28 }
29
30 terraform {
31   backend "s3" {
32     bucket = "s3backend234567812"
33     dynamodb_table = "terraform-state-lock-dynamo"
34     key      = "terraform.tfstate"
35     region  = "us-east-1"
36   }
37 }
```

The terminal window shows the execution of the 'terraform init' command:

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Do you want to copy existing state to the new backend?
Pre-existing state was found while migrating the previous "local" backend to the
newly configured "s3" backend. No existing state was found in the newly
configured "s3" backend. Do you want to copy this state to the new "s3"
backend? Enter "yes" to copy and "no" to start with an empty state.

Enter a value: yes

Releasing state lock. This may take a few moments...

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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.
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

commands will detect it and remind you to do so if necessary.

PS C:\Users\Ashish\Desktop\Terraform basics> **terraform apply**

Acquiring state lock. This may take a few moments...

aws_dynamodb_table.dynamodb-terraform-state-lock: Refreshing state... [id=terraform-sta

aws_s3_bucket.s3_bucket: Refreshing state... [id=s3backend234567812]

aws_instance.web: Refreshing state... [id=i-01773e4f58447a76b]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no

Warning: Argument is deprecated

with aws_s3_bucket.s3_bucket,
on main.tf line 24, in resource "aws_s3_bucket" "s3_bucket":
24: acl = "private"

acl is deprecated. Use the aws_s3_bucket_acl resource instead.

(and one more similar warning elsewhere)

Releasing state lock. This may take a few moments...

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Acquiring state lock. This may take a few moments...

aws_dynamodb_table.dynamodb-terraform-state-lock: Refreshing state... [id=terra

aws_s3_bucket.s3_bucket: Refreshing state... [id=s3backend234567812]

aws_instance.web: Refreshing state... [id=i-01773e4f58447a76b]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and

Warning: Argument is deprecated

with aws_s3_bucket.s3_bucket,
on main.tf line 24, in resource "aws_s3_bucket" "s3_bucket":
24: acl = "private"

acl is deprecated. Use the aws_s3_bucket_acl resource instead.

(and one more similar warning elsewhere)

Releasing state lock. This may take a few moments...

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

PS C:\Users\Ashish\Desktop\Terraform basics>

A table has been created in the dynamodb.

The screenshot shows the AWS DynamoDB 'Explore items' interface. On the left, a sidebar lists 'DynamoDB' options like Dashboard, Tables, and Explore items. Under 'Explore items', 'terraform-state-lock-dynamo' is selected. The main area displays a table named 'terraform-state-lock-dynamo'. A search bar at the top has 'Scan' and 'Query' buttons. Below the search bar, there's a 'Select a table or index' dropdown set to 'Table - terraform-state-lock-dynamo' and a 'Select attribute projection' dropdown set to 'All attributes'. A 'Filters - optional' section with 'Run' and 'Reset' buttons is shown. A green status bar at the bottom indicates 'Completed - Items returned: 1 - Items scanned: 1 - Efficiency: 100% - RCU consumed: 0.5'. The table itself has two columns: 'LockID (String)' and 'Digest'. The first row shows 's3backend23456781...' and '0b2ee50ea33dcc3050f4b258dde18b42'.

Our state file has been stored in dynamodb.

The screenshot shows the AWS DynamoDB 'Edit item' interface for the 'terraform-state-lock-dynamo' table. It displays the same table structure as the previous screenshot. The 'Attributes' table has two rows: one for 'LockID' (partition key) with value 's3backend234567812/terraform.tfstate-md5' and another for 'Digest' with value '0b2ee50ea33dcc3050f4b258dde18b42'. Buttons for 'Cancel', 'Save', and 'Save and close' are at the bottom. The status bar at the bottom right shows 'Account ID: 2353-5102-8455' and 'mujaheed'. At the very bottom, a terminal window shows the command 'terraform state save' being run.