

Infrastructure Automation using Terraform

Use S3 Bucket as Remote Backend



Table of Contents

Prerequisite:.....	3
Walkthrough:	3
Part 1: Initializing Terraform Directory	3
Part 2: Creating S3 Bucket and DynamoDB.....	6
Part 3: Add Backend Configuration.....	7
Part 4: Remove Backend Configuration	9
Part 5: Destroy Resources	11

Infrastructure Automation using Terraform – Lab Guide

This Activity demonstrates the usage of S3 Bucket as Remote Backend in Terraform. Backend is used to define the location where our state files will be stored.

Prerequisite:

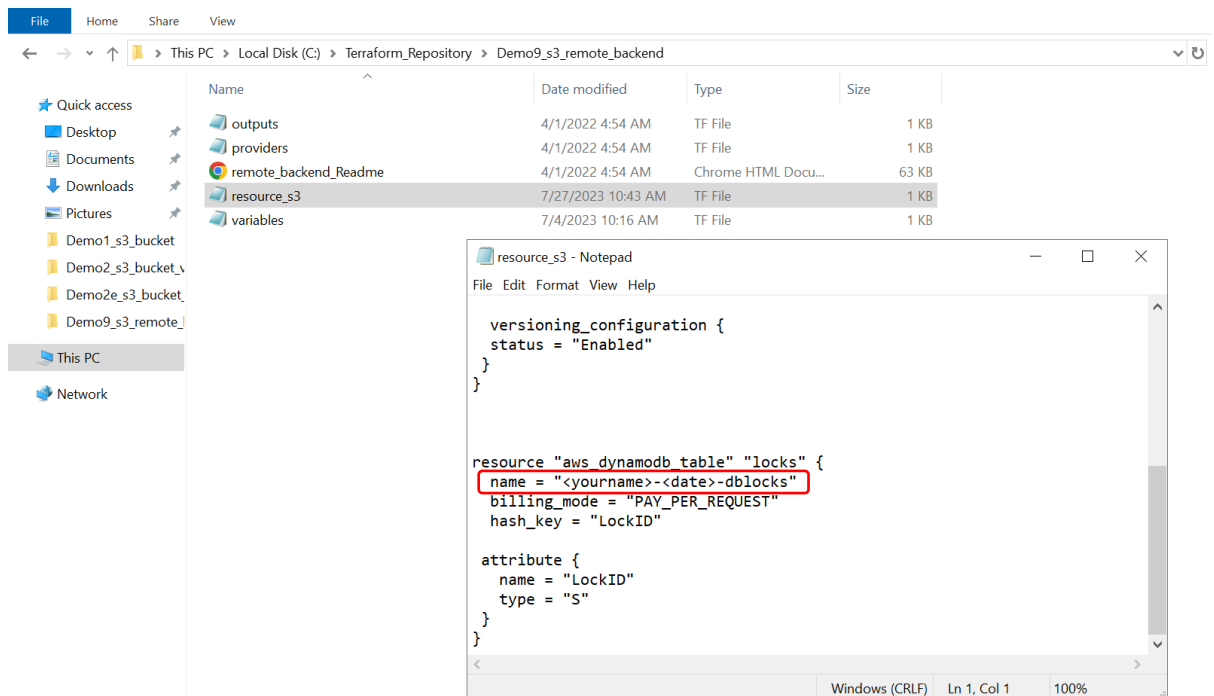
- 1) Download the zip file shared by the trainer and extract it.

Walkthrough:

1. Initializing Terraform Directory
2. Creating S3 Bucket and DynamoDB
3. Add Backend Configuration
4. Remove Backend Configuration
5. Destroy Resources

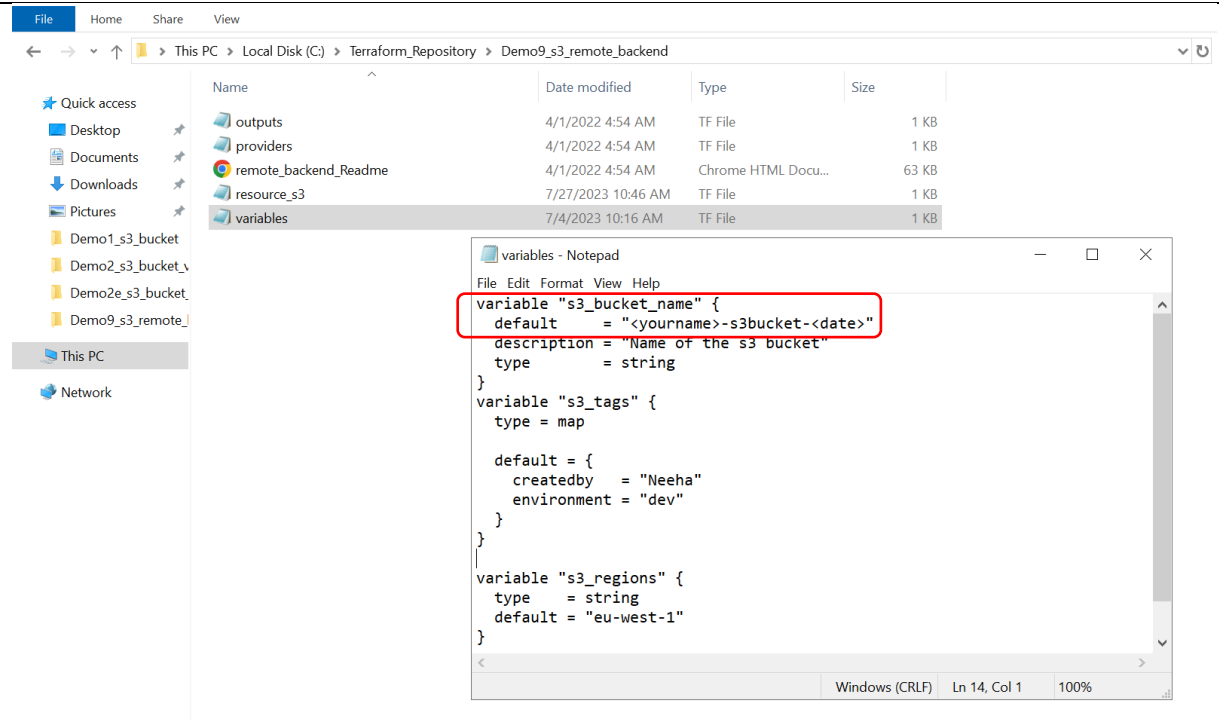
Part 1: Initializing Terraform Directory

- 1 Open the extracted folder and navigate to “.tf” files. Open “resource_s3.tf” and update the “name” argument in aws_dynamodb_table block.



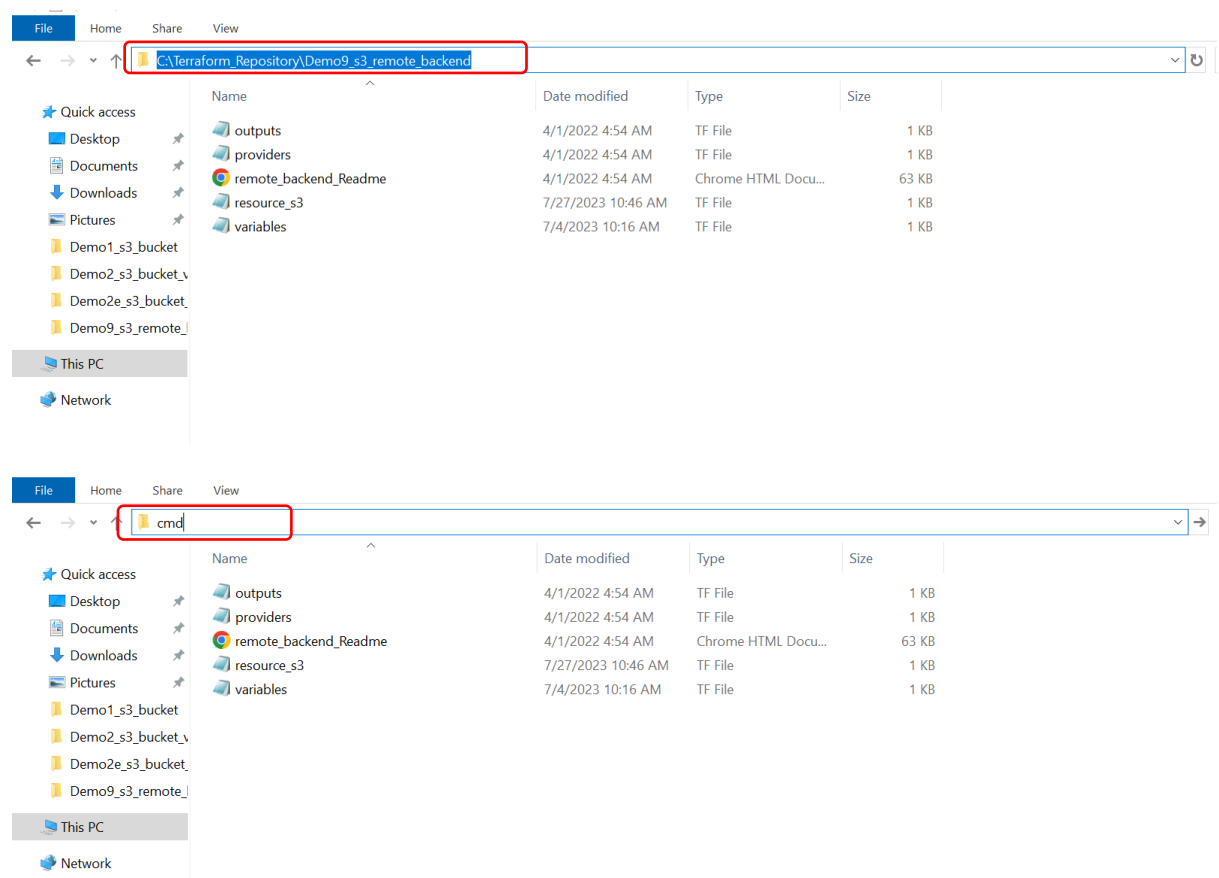
Open “variables.tf” and update “s3_bucket_name” variable.

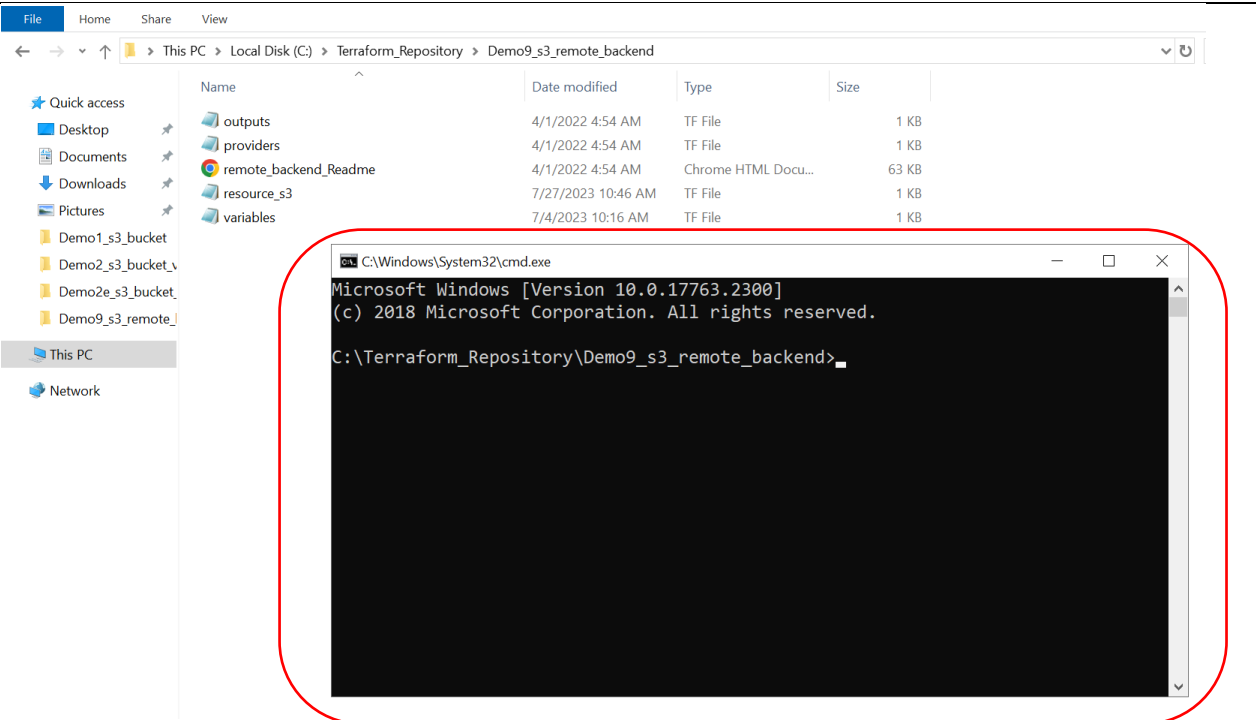
Infrastructure Automation using Terraform – Lab Guide



```
variable "s3_bucket_name" {  
  default = "<yourname>-s3bucket-<date>"  
  description = "Name of the s3 bucket"  
  type = string  
}  
variable "s3_tags" {  
  type = map  
  
  default = {  
    createdby = "Neeha"  
    environment = "dev"  
  }  
}  
variable "s3_regions" {  
  type = string  
  default = "eu-west-1"  
}
```

2 Click on the Address bar and type cmd. Press Enter (It will open a command prompt from that location).



	
3	<p>Execute below command to initialize the current directory as Terraform directory which enables us to run terraform commands to manage Infrastructure.</p> <p>Command :</p> <pre>C:\Terraform_Repository\Demo9_s3_remote_backend>terraform init</pre> <p>Result :</p> <pre>C:\Terraform_Repository\Demo9_s3_remote_backend>terraform init Initializing the backend... Initializing provider plugins... - Finding latest version of hashicorp/aws... - Installing hashicorp/aws v5.9.0... - Installed hashicorp/aws v5.9.0 (signed by HashiCorp) Terraform has created a lock file .terraform.lock.hcl to record the provider selections it made above. Include this file in your version control repository so that Terraform can guarantee to make the same selections by default when you run "terraform init" in the future. Terraform has been successfully initialized! You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work. If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary. C:\Terraform_Repository\Demo9_s3_remote_backend></pre>
4	<p>Next execute below command to validate syntax and configuration of terraform configuration files. If everything is proper, it will return a success message otherwise it will display the errors.</p> <p>Command :</p>

	<pre>C:\Terraform_Repository\Demo9_s3_remote_backend>terraform validate</pre> <p>Result :</p> <pre>C:\Terraform_Repository\Demo9_s3_remote_backend>terraform validate Success! The configuration is valid. C:\Terraform_Repository\Demo9_s3_remote_backend>_</pre>
5	<p>Next run below command and observe the output. The output contains information depicting all the changes which will happen in the AWS cloud. It is like dry-run to ensure whatever we are trying to do using terraform commands is what we want.</p> <p>Command :</p> <pre>C:\Terraform_Repository\Demo9_s3_remote_backend>terraform plan -out "backend.tfplan"</pre> <p>Result :</p> <pre>C:\Terraform_Repository\Demo9_s3_remote_backend>terraform plan -out "backend.tfplan" Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols: + create Terraform will perform the following actions: # aws_dynamodb_table.locks will be created + resource "aws_dynamodb_table" "locks" { + arn = (known after apply) + billing_mode = "PAY_PER_REQUEST" + hash_key = "LockID" + id = (known after apply) + name = "neeha-2707-dblocks" + read_capacity = (known after apply) + stream_arn = (known after apply) + stream_label = (known after apply) + stream_view_type = (known after apply) + tags_all = (known after apply) + write_capacity = (known after apply) + attribute { + name = "LockID" + type = "S" + } + point_in_time_recovery { + enabled = (known after apply) + } }</pre>

Part 2: Creating S3 Bucket and DynamoDB

1	<p>For creating resources , execute below command and observe the actions performed by the command.</p> <p>Command :</p> <pre>C:\Terraform_Repository\Demo9_s3_remote_backend>terraform apply "backend.tfplan"</pre> <p>Result :</p>
---	---

```
C:\Terraform_Repository\Demo9_s3_remote_backend>terraform apply "backend.tfplan"
aws_dynamodb_table.locks: Creating...
aws_s3_bucket.bckt: Creating...
aws_s3_bucket.bckt: Creation complete after 4s [id=neeha-s3bucket-2707]
aws_s3_bucket_versioning.versioning_example: Creating...
aws_s3_bucket_server_side_encryption_configuration.example: Creating...
aws_s3_bucket_server_side_encryption_configuration.example: Creation complete after 1s [id=neeha-s3bucket-2707]
aws_s3_bucket_versioning.versioning_example: Creation complete after 2s [id=neeha-s3bucket-2707]
aws_dynamodb_table.locks: Creation complete after 9s [id=neeha-2707-dblocks]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

Outputs:

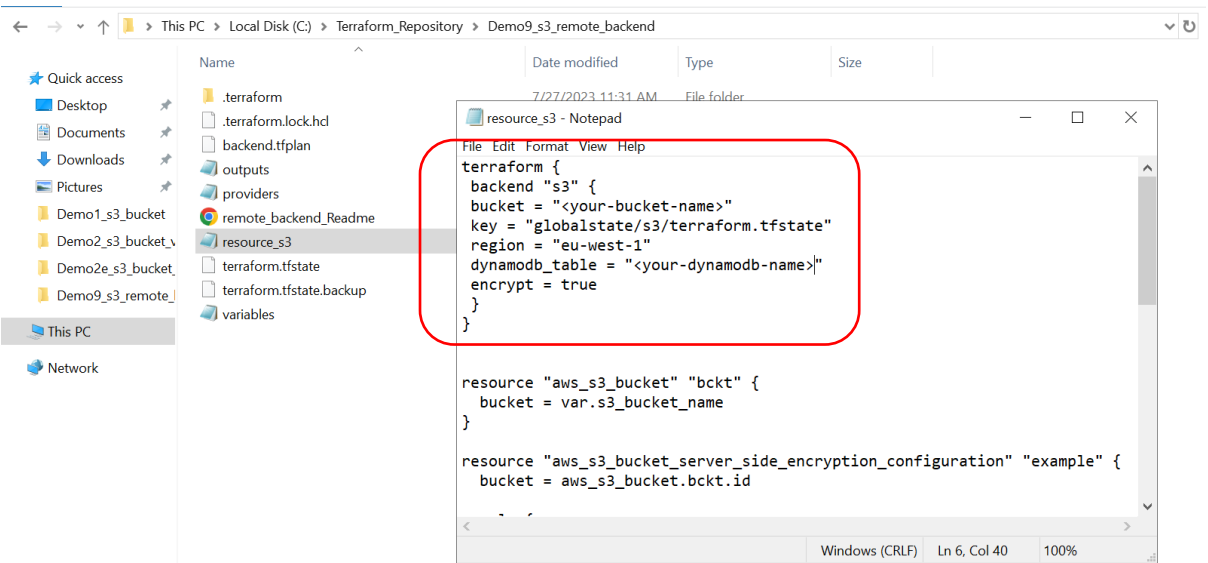
dynamodb_table_name = "neeha-2707-dblocks"
s3_bucket_arn = "arn:aws:s3:::neeha-s3bucket-2707"

C:\Terraform_Repository\Demo9_s3_remote_backend>_
```

Part 3: Add Backend Configuration

1

Add below backend block to "resource_s3.tf" file. Make sure that you are providing appropriate bucket and dynamodb_table value in the backend block.



```
terraform {
  backend "s3" {
    bucket = "<your-bucket-name>"
    key = "globalstate/s3/terraform.tfstate"
    region = "eu-west-1"
    dynamodb_table = "<your-dynamodb-name>"
    encrypt = true
  }
}

resource "aws_s3_bucket" "bckt" {
  bucket = var.s3_bucket_name
}

resource "aws_s3_bucket_server_side_encryption_configuration" "example" {
  bucket = aws_s3_bucket.bckt.id
}
```

2

To store state file in the mentioned s3 bucket, we must run below command again. Since we are changing backend location, terraform will ask for your approval. Please enter "yes".

Command :

```
C:\Terraform_Repository\Demo9_s3_remote_backend>terraform init
```

Result :

```
C:\Terraform_Repository\Demo9_s3_remote_backend>terraform init

Initializing the backend...
Acquiring state lock. This may take a few moments...
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: _
```

```
C:\Terraform_Repository\Demo9_s3_remote_backend>terraform init
```

Initializing the backend...

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

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 v5.9.0

Terraform has been successfully initialized!

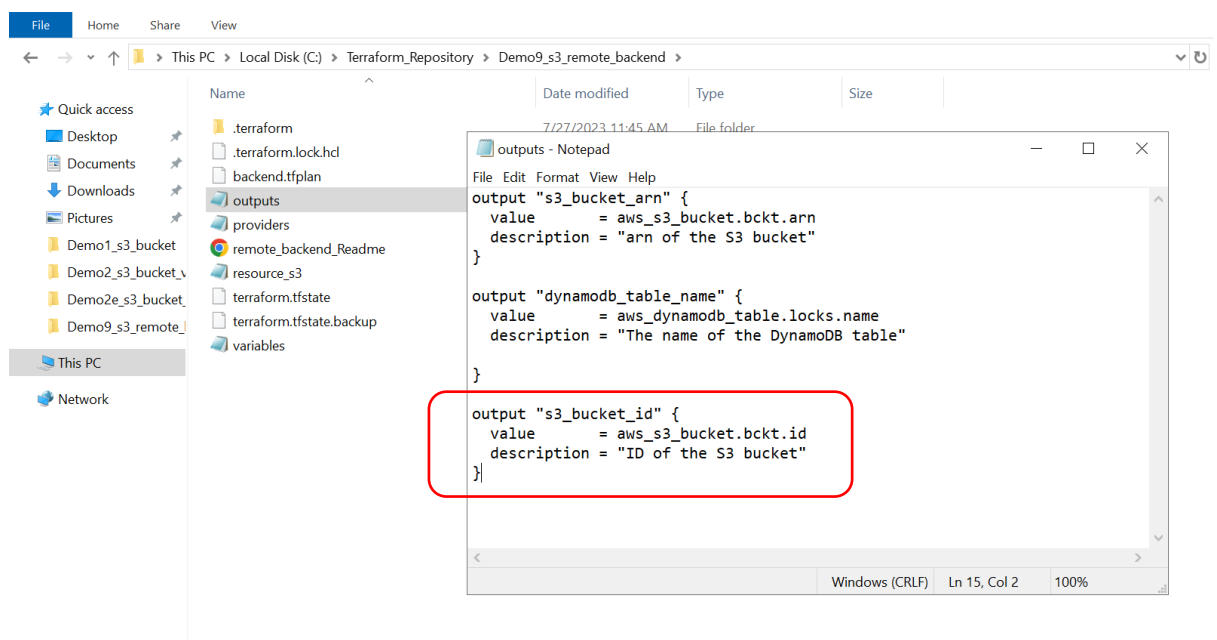
You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

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.

```
C:\Terraform_Repository\Demo9_s3_remote_backend>
```

3

To explore more, edit "output.tf" and run below commands to observe how state file is getting stored in s3 bucket.



Command :

```
C:\Terraform_Repository\Demo9_s3_remote_backend>terraform plan -out "backend.tfplan"
```

Result :


```
C:\Terraform_Repository\Demo9_s3_remote_backend>terraform plan -out "backend.tfplan"
Acquiring state lock. This may take a few moments...
aws_dynamodb_table.locks: Refreshing state... [id=neeha-2707-dblocks]
aws_s3_bucket.bckt: Refreshing state... [id=neeha-s3bucket-2707]
aws_s3_bucket_versioning.versioning_example: Refreshing state... [id=neeha-s3bucket-2707]
aws_s3_bucket_server_side_encryption_configuration.example: Refreshing state... [id=neeha-s3bucket-2707]
```

Note: Objects have changed outside of Terraform

Terraform detected the following changes made outside of Terraform since the last "terraform apply":

```
# aws_dynamodb_table.locks has been changed
~ resource "aws_dynamodb_table" "locks" {
  id           = "neeha-2707-dblocks"
  name         = "neeha-2707-dblocks"
  + tags       = {}
  # (9 unchanged attributes hidden)

  # (3 unchanged blocks hidden)
}
# aws_s3_bucket.bckt has been changed
~ resource "aws_s3_bucket" "bckt" {
  id           = "neeha-s3bucket-2707"
  + tags       = {}
  # (10 unchanged attributes hidden)
```

Command :

```
C:\Terraform_Repository\Demo9_s3_remote_backend>terraform apply "backend.tfplan"
```

Result :

```
C:\Terraform_Repository\Demo9_s3_remote_backend>terraform apply "backend.tfplan"
Acquiring state lock. This may take a few moments...
Releasing state lock. This may take a few moments...

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

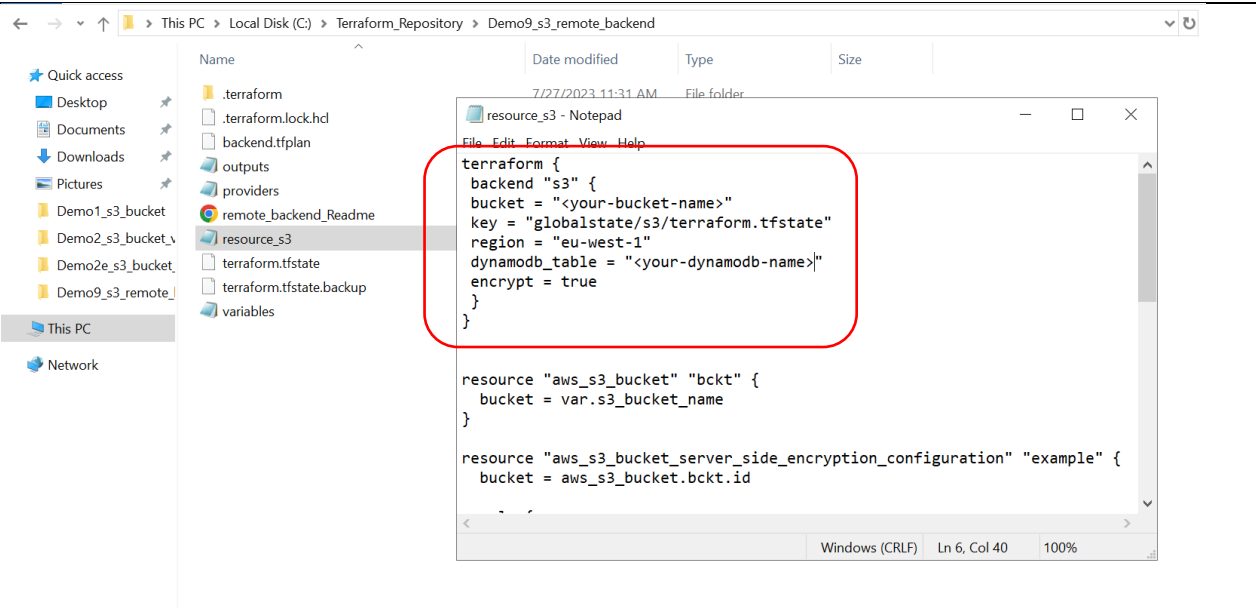
Outputs:

dynamodb_table_name = "neeha-2707-dblocks"
s3_bucket_arn       = "arn:aws:s3:::neeha-s3bucket-2707"
s3_bucket_id        = "neeha-s3bucket-2707"

C:\Terraform_Repository\Demo9_s3_remote_backend>_
```

Part 4: Remove Backend Configuration

- 1 Remove below backend block from "resource_s3.tf" file.

	 <pre> terraform { backend "s3" { bucket = "<your-bucket-name>" key = "globalstate/s3/terraform.tfstate" region = "eu-west-1" dynamodb_table = "<your-dynamodb-name>" encrypt = true } } resource "aws_s3_bucket" "bckt" { bucket = var.s3_bucket_name } resource "aws_s3_bucket_server_side_encryption_configuration" "example" { bucket = aws_s3_bucket.bckt.id </pre>
2	<p>Since we have removed remote backend information, we must execute below command to configure backend. It will ask for your approval, please provide "yes".</p> <p>Command :</p> <pre>C:\Terraform_Repository\Demo9_s3_remote_backend>terraform init -migrate-state</pre> <p>Result :</p> <pre> C:\Terraform_Repository\Demo9_s3_remote_backend>terraform init -migrate-state Initializing the backend... Terraform has detected you're unconfiguring your previously set "s3" backend. Acquiring state lock. This may take a few moments... Do you want to copy existing state to the new backend? Pre-existing state was found while migrating the previous "s3" backend to the newly configured "local" backend. No existing state was found in the newly configured "local" backend. Do you want to copy this state to the new "local" backend? Enter "yes" to copy and "no" to start with an empty state. Enter a value: </pre>

	<pre> Initializing the backend... Terraform has detected you're unconfiguring your previously set "s3" backend. Acquiring state lock. This may take a few moments... Do you want to copy existing state to the new backend? Pre-existing state was found while migrating the previous "s3" backend to the newly configured "local" backend. No existing state was found in the newly configured "local" backend. Do you want to copy this state to the new "local" 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 unset the backend "s3". Terraform will now operate locally. Initializing provider plugins... - Reusing previous version of hashicorp/aws from the dependency lock file - Using previously-installed hashicorp/aws v5.9.0 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. C:\Terraform_Repository\Demo9_s3_remote_backend>_ </pre>
3	<p>Run below command to verify backend configuration.</p> <p>Command :</p> <pre>C:\Terraform_Repository\Demo9_s3_remote_backend>terraform plan -out "backend.tfplan"</pre> <p>Result :</p> <pre> C:\Terraform_Repository\Demo9_s3_remote_backend>terraform plan -out "backend.tfplan" aws_dynamodb_table.locks: Refreshing state... [id=neeha-2707-dblocks] aws_s3_bucket.bckt: Refreshing state... [id=neeha-s3bucket-2707] aws_s3_bucket_server_side_encryption_configuration.example: Refreshing state... [id=neeha-s3bucket-2707] aws_s3_bucket_versioning.versioning_example: Refreshing state... [id=neeha-s3bucket-2707] No changes. Your infrastructure matches the configuration. Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed. C:\Terraform_Repository\Demo9_s3_remote_backend>_ </pre>

Part 5: Destroy Resources

1	<p>Execute below command to destroy the resources which we have created in previous step. After you execute below command, it will show you what changes will be done and before doing those changes it will ask for your approval. So, if you want to proceed with destroying resources, provide "yes".</p> <p>Command:</p> <pre>C:\Terraform_Repository\Demo9_s3_remote_backend>terraform destroy</pre> <p>Result:</p>
---	---

```
C:\Terraform_Repository\Demo9_s3_remote_backend>terraform destroy
aws_dynamodb_table.locks: Refreshing state... [id=neeha-2707-dblocks]
aws_s3_bucket.bckt: Refreshing state... [id=neeha-s3bucket-2707]
aws_s3_bucket_versioning.versioning_example: Refreshing state... [id=neeha-s3bucket-2707]
aws_s3_bucket_server_side_encryption_configuration.example: Refreshing state... [id=neeha-s3bucket-2707]

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

Terraform will perform the following actions:

# aws_dynamodb_table.locks will be destroyed
- resource "aws_dynamodb_table" "locks" {
  - arn = "arn:aws:dynamodb:eu-west-1:659840170574:table/neeha-2707-dblocks" -> null
  - billing_mode = "PAY_PER_REQUEST" -> null
  - deletion_protection_enabled = false -> null
  - hash_key = "LockID" -> null
  - id = "neeha-2707-dblocks" -> null
  - name = "neeha-2707-dblocks" -> null
  - read_capacity = 0 -> null
  - stream_enabled = false -> null
  - table_class = "STANDARD" -> null
  - tags = {} -> null
  - tags_all = {} -> null
  - write_capacity = 0 -> null

  - attribute {
    - name = "LockID" -> null
    - type = "S" -> null
  }
}
```

Plan: 0 to add, 0 to change, 4 to destroy.

Changes to Outputs:

```
- dynamodb_table_name = "neeha-2707-dblocks" -> null
- s3_bucket_arn = "arn:aws:s3:::neeha-s3bucket-2707" -> null
- s3_bucket_id = "neeha-s3bucket-2707" -> null
```

Do you really want to destroy all resources?

Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value:

Plan: 0 to add, 0 to change, 4 to destroy.

Changes to Outputs:

```
- dynamodb_table_name = "neeha-2707-dblocks" -> null
- s3_bucket_arn = "arn:aws:s3:::neeha-s3bucket-2707" -> null
- s3_bucket_id = "neeha-s3bucket-2707" -> null
```

Do you really want to destroy all resources?

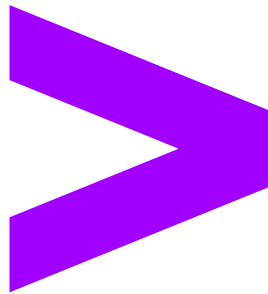
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

```
aws_s3_bucket_versioning.versioning_example: Destroying... [id=neeha-s3bucket-2707]
aws_s3_bucket_server_side_encryption_configuration.example: Destroying... [id=neeha-s3bucket-2707]
aws_dynamodb_table.locks: Destroying... [id=neeha-2707-dblocks]
aws_s3_bucket_versioning.versioning_example: Destruction complete after 0s
aws_s3_bucket_server_side_encryption_configuration.example: Destruction complete after 1s
aws_s3_bucket.bckt: Destroying... [id=neeha-s3bucket-2707]
aws_s3_bucket.bckt: Destruction complete after 1s
aws_dynamodb_table.locks: Destruction complete after 2s
```

Destroy complete! Resources: 4 destroyed.

C:\Terraform_Repository\Demo9_s3_remote_backend>



Copyright © 2023 Accenture
All rights reserved.
Accenture and its logo are trademarks of Accenture.