

Infrastructure Automation using Terraform

Create a S3 Bucket – Displaying Output



Table of Contents

Prerequisite:.....	3
Walkthrough:	3
Part 1: Initializing Terraform Directory	3
Part 2: Creating S3 Bucket – Displaying Output.....	6
Part 3: Destroying S3 Bucket.....	6

Infrastructure Automation using Terraform – Lab Guide

This Activity demonstrates the creation of S3 Bucket in AWS using Terraform and using Output block to display information on the command line.

Prerequisite:

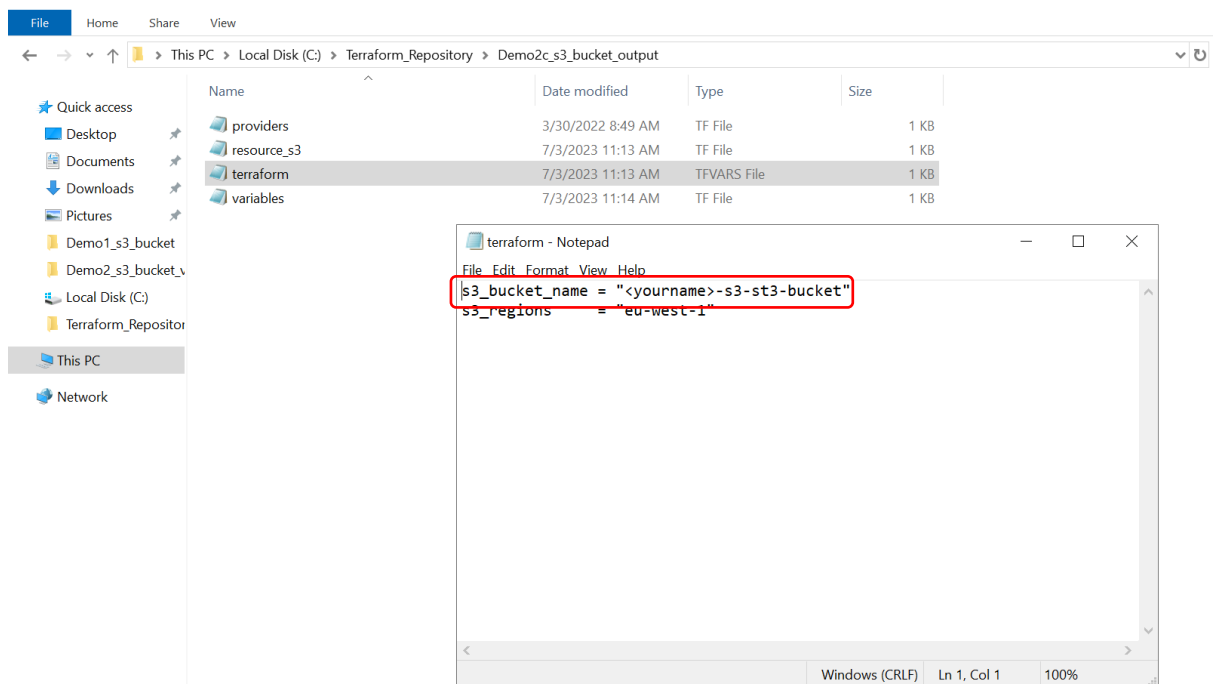
- 1) Download the zip file shared by the trainer and extract it.

Walkthrough:

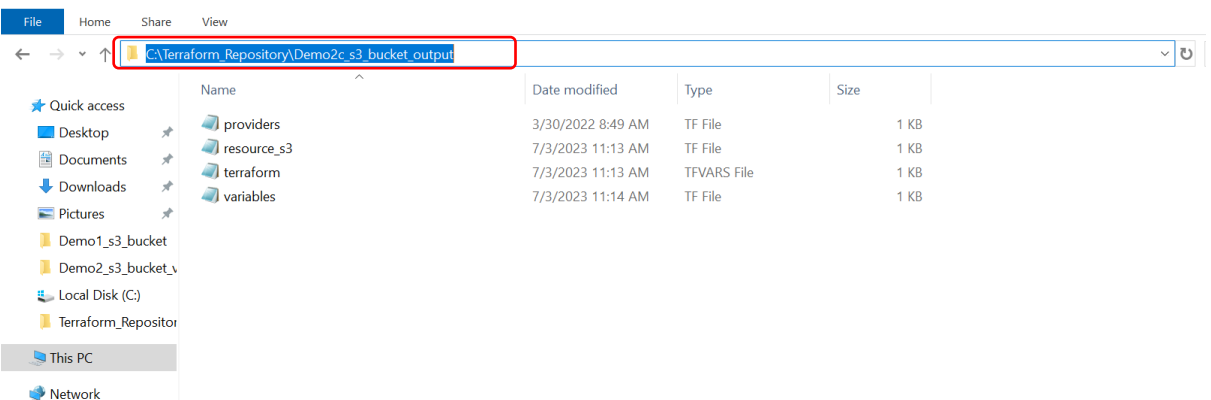
1. Initializing Terraform Directory
2. Creating S3 Bucket – Displaying Output
3. Destroying S3 Bucket

Part 1: Initializing Terraform Directory

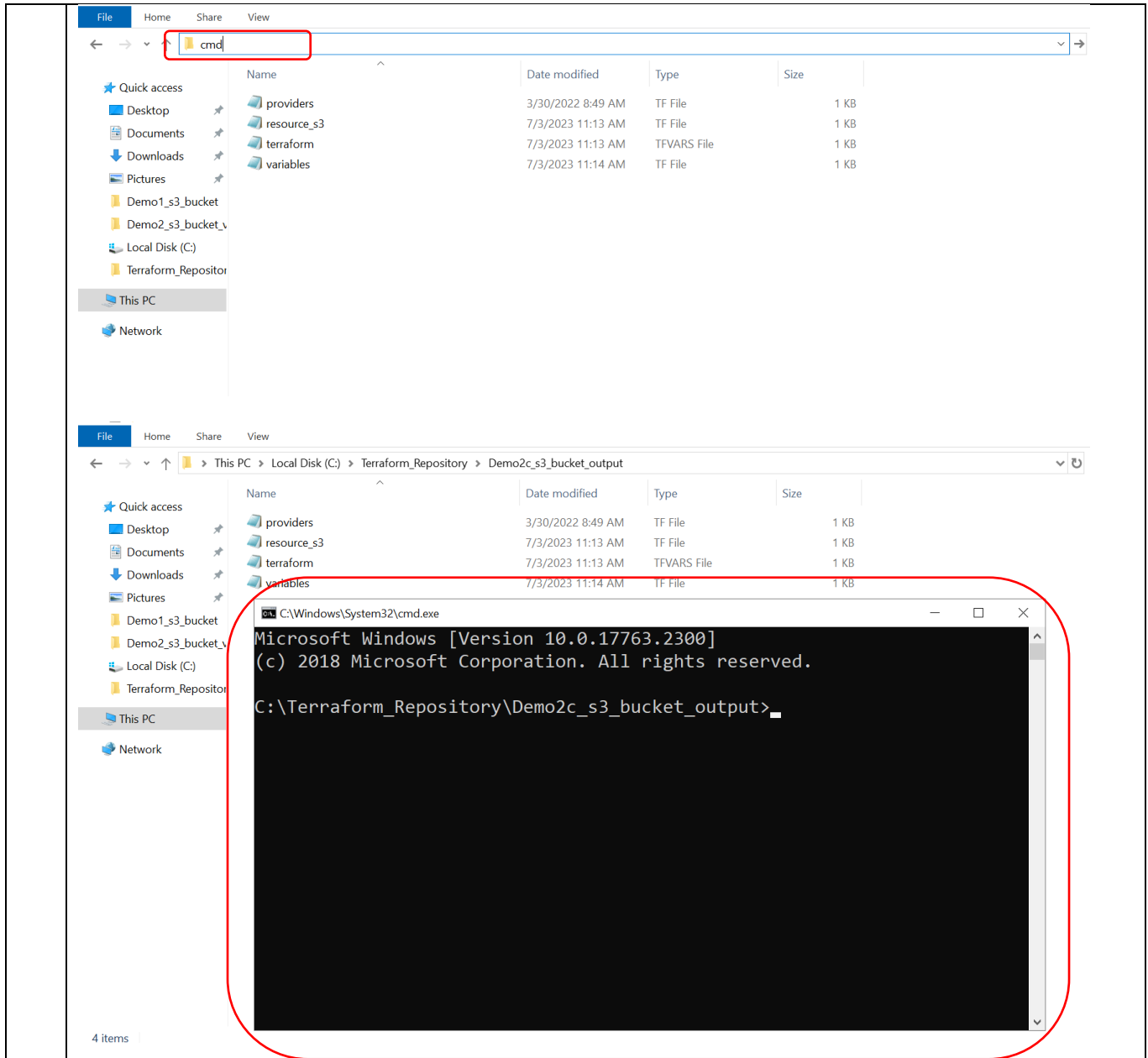
- 1 Open the extracted folder and navigate to “.tf” files. Open “terraform.tfvars” and update the “s3_bucket_name” argument.



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



Infrastructure Automation using Terraform – Lab Guide



3

Execute below command to initialize the current directory as Terraform directory which enables us to run terraform commands to manage Infrastructure.

Command :

```
C:\Terraform_Repository\Demo2c_s3_bucket_output>terraform init
```

Result :

	<pre> C:\Terraform_Repository\Demo2c_s3_bucket_output>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\Demo2c_s3_bucket_output>_ </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\Demo2c_s3_bucket_output>terraform validate </pre> <p>Result :</p> <pre> C:\Terraform_Repository\Demo2c_s3_bucket_output>terraform validate Success! The configuration is valid. C:\Terraform_Repository\Demo2c_s3_bucket_output>_ </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\Demo2c_s3_bucket_output>terraform plan -out "s3_locals.tfplan" </pre> <p>Result :</p>

	<pre>C:\Terraform_Repository\Demo2c_s3_bucket_output>terraform plan -out "s3_locals.tfplan"</pre> <p>Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:</p> <ul style="list-style-type: none"> + create <p>Terraform will perform the following actions:</p> <pre># aws_s3_bucket.mybucket will be created + resource "aws_s3_bucket" "mybucket" { + acceleration_status = (known after apply) + acl = (known after apply) + arn = (known after apply) + bucket = "neeha-s3-st3-bucket" + bucket_domain_name = (known after apply) + bucket_prefix = (known after apply) + bucket_regional_domain_name = (known after apply) + force_destroy = false + hosted_zone_id = (known after apply) + id = (known after apply) + object_lock_enabled = (known after apply) + policy = (known after apply) + region = (known after apply) + request_payer = (known after apply) + tags = { + "env" = "dev" } }</pre>
--	---

Part 2: Creating S3 Bucket – Displaying Output

1	<p>For creating a S3 Bucket , execute below command and observe the actions performed by the command.</p> <p>Command :</p> <pre>C:\Terraform_Repository\Demo2c_s3_bucket_output>terraform apply "s3_locals.tfplan"</pre> <p>Result :</p> <pre>C:\Terraform_Repository\Demo2c_s3_bucket_output>terraform apply "s3_locals.tfplan" aws_s3_bucket.mybucket: Creating... aws_s3_bucket.mybucket: Creation complete after 3s [id=neeha-s3-st3-bucket] Apply complete! Resources: 1 added, 0 changed, 0 destroyed.</pre> <div style="border: 1px solid red; padding: 5px; margin: 10px 0;"> <p>Outputs:</p> <pre>bucketarn = "arn:aws:s3:::neeha-s3-st3-bucket" bucketname = "neeha-s3-st3-bucket"</pre> </div> <pre>C:\Terraform_Repository\Demo2c_s3_bucket_output>_</pre> <p>NOTE : After the S3 Bucket got created, you will be able to notice that some information related to S3 Bucket is getting displayed on terminal. Whatever details we want to display on the terminal, we can define them in output block.</p>
---	--

Part 3: Destroying S3 Bucket

1	<p>Execute below command to destroy the S3 Bucket 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 S3 Bucket, provide "yes".</p> <p>Command:</p> <pre>C:\Terraform_Repository\Demo2c_s3_bucket_output>terraform destroy</pre> <p>Result:</p>
---	---

```
C:\Terraform_Repository\Demo2c_s3_bucket_output>terraform destroy
aws_s3_bucket.mybucket: Refreshing state... [id=neeha-s3-st3-bucket]

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_s3_bucket.mybucket will be destroyed
- resource "aws_s3_bucket" "mybucket" {
  - arn                        = "arn:aws:s3:::neeha-s3-st3-bucket" -> null
  - bucket                    = "neeha-s3-st3-bucket" -> null
  - bucket_domain_name       = "neeha-s3-st3-bucket.s3.amazonaws.com" -> null
  - bucket_regional_domain_name = "neeha-s3-st3-bucket.s3.eu-west-1.amazonaws.com" -> null
  - force_destroy            = false -> null
  - hosted_zone_id           = "Z1BKCTXD74EZPE" -> null
  - id                       = "neeha-s3-st3-bucket" -> null
  - object_lock_enabled      = false -> null
  - region                   = "eu-west-1" -> null
  - request_payer            = "BucketOwner" -> null
  - tags                     = {
    - "env" = "dev"
  } -> null
  - tags_all                  = {
    - "env" = "dev"
  }
}
```

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

Changes to Outputs:

```
- bucketarn = "arn:aws:s3:::neeha-s3-st3-bucket" -> null
- bucketname = "neeha-s3-st3-bucket" -> 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, 1 to destroy.

Changes to Outputs:

```
- bucketarn = "arn:aws:s3:::neeha-s3-st3-bucket" -> null
- bucketname = "neeha-s3-st3-bucket" -> 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.

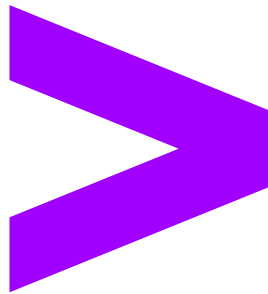
Enter a value: yes

aws_s3_bucket.mybucket: Destroying... [id=neeha-s3-st3-bucket]

aws_s3_bucket.mybucket: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.

C:\Terraform_Repository\Demo2c_s3_bucket_output>



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