

Infrastructure Automation using Terraform

Working with State Management Commands



Table of Contents

Prerequisite:.....	3
Walkthrough:	3
Part 1: Initializing Terraform Directory	3
Part 2: Working with State Management Commands	6
Part 3: Destroying Resources	9

Infrastructure Automation using Terraform – Lab Guide

This Activity demonstrates the working of State Management commands in Terraform.

Prerequisite:

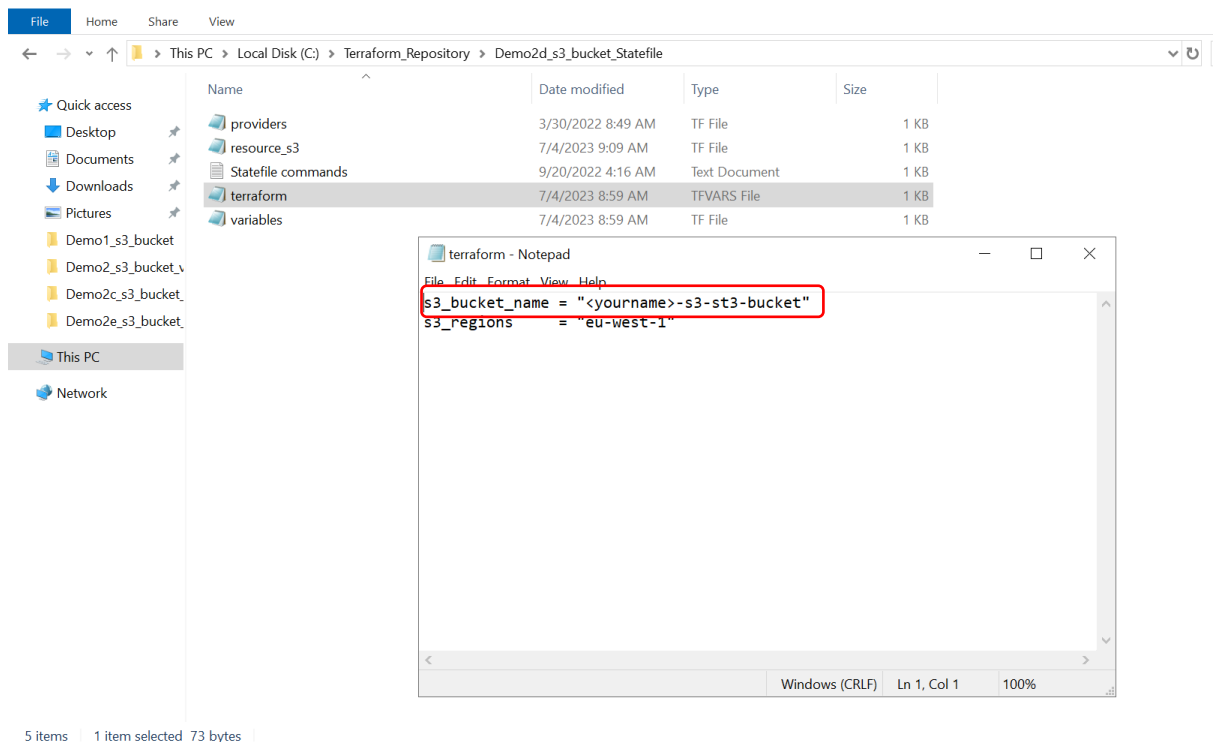
- 1) Download the zip file shared by the trainer and extract it.

Walkthrough:

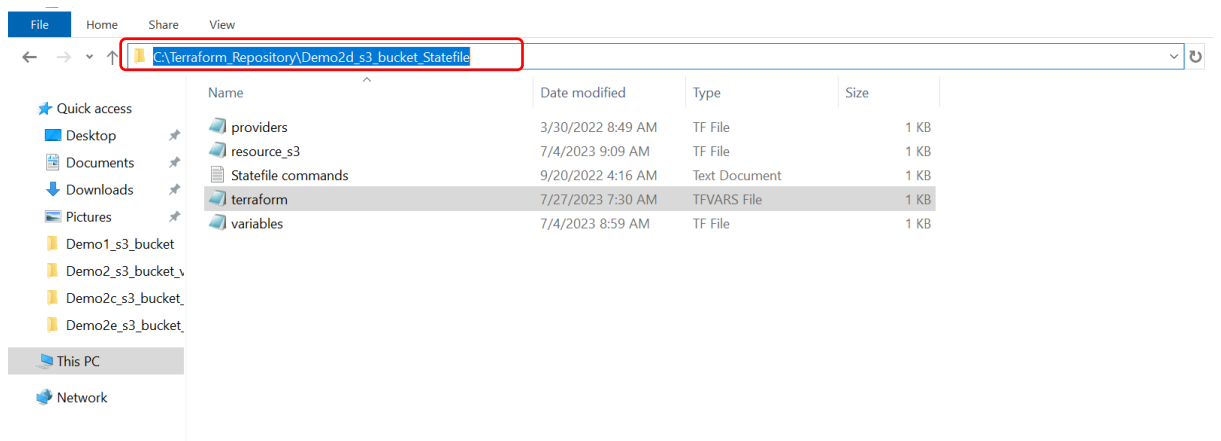
1. Initializing Terraform Directory
2. Working with the State Management Commands
3. Destroying Resources

Part 1: Initializing Terraform Directory

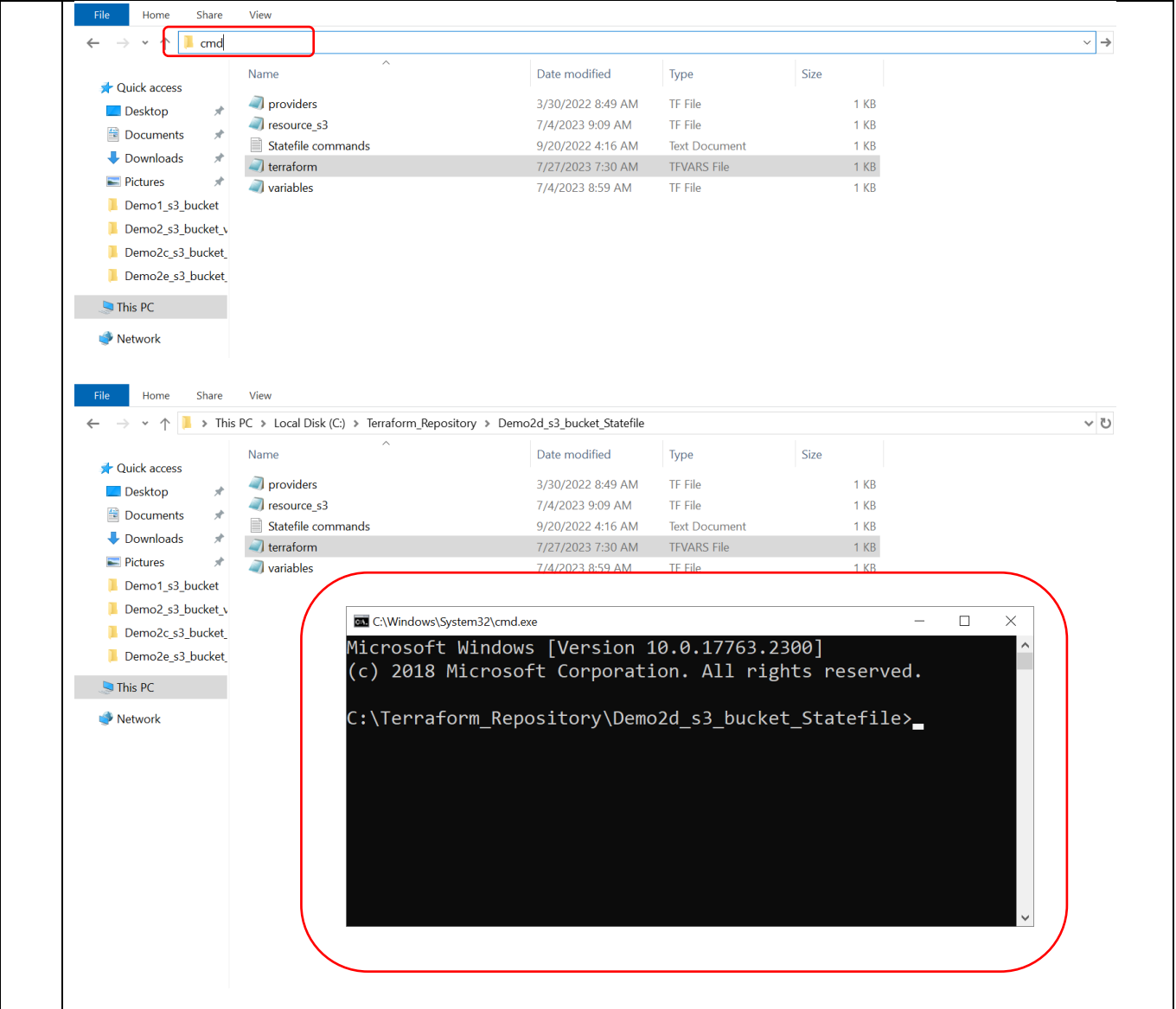
- 1 Open the extracted folder and navigate to “.tf” files. Open “terraform.tfvars” and update the “s3_bucket_name” argument.



- 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



The screenshot shows two windows. The top window is a File Explorer view of the 'cmd' directory, listing files: providers, resource_s3, Statefile commands, terraform, and variables. The bottom window is a File Explorer view of 'This PC > Local Disk (C:) > Terraform_Repository > Demo2d_s3_bucket_Statefile', listing the same files. A Command Prompt window is overlaid on the bottom File Explorer, showing the path 'C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>'.

Name	Date modified	Type	Size
providers	3/30/2022 8:49 AM	TF File	1 KB
resource_s3	7/4/2023 9:09 AM	TF File	1 KB
Statefile commands	9/20/2022 4:16 AM	Text Document	1 KB
terraform	7/27/2023 7:30 AM	TFVARS File	1 KB
variables	7/4/2023 8:59 AM	TF File	1 KB

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.17763.2300]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>
```

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\Demo2d_s3_bucket_Statefile>terraform init
```

Result :

	<pre> C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>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\Demo2d_s3_bucket_Statefile>_ </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\Demo2d_s3_bucket_Statefile>terraform validate </pre> <p>Result :</p> <pre> C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>terraform validate Success! The configuration is valid. C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>_ </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\Demo2d_s3_bucket_Statefile>terraform plan -out "s3.tfplan" </pre> <p>Result :</p>

	<pre> C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>terraform plan -out "s3.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_ebs_volume.myblockvolume will be created + resource "aws_ebs_volume" "myblockvolume" { + arn = (known after apply) + availability_zone = "eu-west-1a" + encrypted = true + final_snapshot = false + id = (known after apply) + iops = (known after apply) + kms_key_id = (known after apply) + size = 10 + snapshot_id = (known after apply) + tags_all = (known after apply) + throughput = (known after apply) + type = (known after apply) } # aws_s3_bucket.myblock will be created + resource "aws_s3_bucket" "myblock" { </pre>
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Part 2: Working with State Management Commands

1	<p>For creating resources , execute below command and observe the actions performed by the command.</p> <p>Command :</p> <pre> C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>terraform apply "s3.tfplan" </pre> <p>Result :</p> <pre> C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>terraform apply "s3.tfplan" aws_s3_bucket.myblock: Creating... aws_s3_bucket.mybucket: Creating... aws_ebs_volume.myblockvolume: Creating... aws_s3_bucket.myblock: Creation complete after 3s [id=terraform-20230727073441533700000001] aws_s3_bucket.mybucket: Creation complete after 3s [id=neeha-s3-st3-bucket] aws_ebs_volume.myblockvolume: Still creating... [10s elapsed] aws_ebs_volume.myblockvolume: Creation complete after 11s [id=vol-03314323e0e729ad3] Apply complete! Resources: 3 added, 0 changed, 0 destroyed. Outputs: bucketname = "neeha-s3-st3-bucket" bucketname2 = "terraform-20230727073441533700000001" volumename = "vol-03314323e0e729ad3" C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>_ </pre>
2	<p>If you have output blocks within your “.tf” files and you want to see what information they are storing, then execute below command.</p> <p>Command:</p> <pre> C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>terraform output </pre> <p>Result:</p>

	<pre>C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>terraform output bucketname = "neeha-s3-st3-bucket" bucketname2 = "terraform-20230727073441533700000001" volumename = "vol-03314323e0e729ad3" C:\Terraform_Repository\Demo2d_s3_bucket_Statefile></pre>
3	<p>For viewing the contents stored in State file or plan file, execute below command.</p> <p>Command: (For viewing State file Content)</p> <pre>C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>terraform show</pre> <p>Result:</p> <pre>C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>terraform show # aws_ebs_volume.myblockvolume: resource "aws_ebs_volume" "myblockvolume" { arn = "arn:aws:ec2:eu-west-1:659840170574:volume/vol-03314323e0e729ad3" availability_zone = "eu-west-1a" encrypted = true final_snapshot = false id = "vol-03314323e0e729ad3" iops = 100 kms_key_id = "arn:aws:kms:eu-west-1:659840170574:key/95519ebd-294f-4e3b-b789-93bb012d3cb0" multi_attach_enabled = false size = 10 tags_all = {} throughput = 0 type = "gp2" } # aws_s3_bucket.myblock: resource "aws_s3_bucket" "myblock" { arn = "arn:aws:s3:::terraform-20230727073441533700000001" bucket = "terraform-20230727073441533700000001" bucket_domain_name = "terraform-20230727073441533700000001.s3.amazonaws.com" bucket_prefix = "terraform-" bucket_regional_domain_name = "terraform-20230727073441533700000001.s3.eu-west-1.amazonaws.com" force_destroy = false hosted_zone_id = "Z1BKCTXD74EZPE" }</pre> <p>Command: (For viewing Plan file Content)</p> <pre>C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>terraform show "s3.tfplan"</pre> <p>Result:</p>

	<pre> C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>terraform show "s3.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_ebs_volume.myblockvolume will be created + resource "aws_ebs_volume" "myblockvolume" { + arn = (known after apply) + availability_zone = "eu-west-1a" + encrypted = true + final_snapshot = false + id = (known after apply) + iops = (known after apply) + kms_key_id = (known after apply) + size = 10 + snapshot_id = (known after apply) + tags_all = (known after apply) + throughput = (known after apply) + type = (known after apply) } # aws_s3_bucket.myblock will be created + resource "aws_s3_bucket" "myblock" { </pre>
4	<p>For listing out all the resources, whose information is stored in State file. Execute below command.</p> <p>Command:</p> <pre> C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>terraform state list </pre> <p>Result:</p> <pre> C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>terraform state list aws_ebs_volume.myblockvolume aws_s3_bucket.myblock aws_s3_bucket.mybucket C:\Terraform_Repository\Demo2d_s3_bucket_Statefile> </pre>
5	<p>For viewing detailed information of each resource, Execute below command.</p> <p>Command:</p> <pre> C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>terraform state show "aws_ebs_volume.myblockvolume" </pre> <p>Result:</p> <pre> C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>terraform state show "aws_ebs_volume.myblockvolume" # aws_ebs_volume.myblockvolume: resource "aws_ebs_volume" "myblockvolume" { arn = "arn:aws:ec2:eu-west-1:659840170574:volume/vol-03314323e0e729ad3" availability_zone = "eu-west-1a" encrypted = true final_snapshot = false id = "vol-03314323e0e729ad3" iops = 100 kms_key_id = "arn:aws:kms:eu-west-1:659840170574:key/95519ebd-294f-4e3b-b789-93bb012d3cb0" multi_attach_enabled = false size = 10 tags_all = {} throughput = 0 type = "gp2" } C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>_ </pre>

Part 3: Destroying Resources

- 1 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”.

Command:

```
C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>terraform destroy
```

Result:

```
C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>terraform destroy
aws_ebs_volume.myblockvolume: Refreshing state... [id=vol-03314323e0e729ad3]
aws_s3_bucket.myblock: Refreshing state... [id=terraform-20230727073441533700000001]
aws_s3_bucket.mybucket: Refreshing state... [id=neeha-s3-st3-bucket]
```

Note: Objects have changed outside of Terraform

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

```
# aws_s3_bucket.myblock has been changed
~ resource "aws_s3_bucket" "myblock" {
  id           = "terraform-20230727073441533700000001"
+ tags        = {}
  # (11 unchanged attributes hidden)

  # (3 unchanged blocks hidden)
}
# aws_ebs_volume.myblockvolume has been changed
~ resource "aws_ebs_volume" "myblockvolume" {
  id           = "vol-03314323e0e729ad3"
+ tags        = {}
  # (11 unchanged attributes hidden)
}
```

Plan: 0 to add, 0 to change, 3 to destroy.

Changes to Outputs:

```
- bucketname = "neeha-s3-st3-bucket" -> null
- bucketname2 = "terraform-20230727073441533700000001" -> null
- volumename = "vol-03314323e0e729ad3" -> 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, 3 to destroy.

Changes to Outputs:
  - bucketname = "neeha-s3-st3-bucket" -> null
  - bucketname2 = "terraform-20230727073441533700000001" -> null
  - volumename = "vol-03314323e0e729ad3" -> null

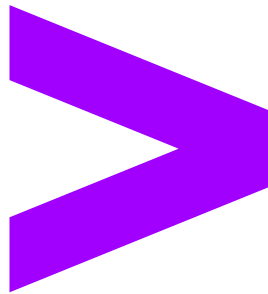
Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_s3_bucket.mybucket: Destroying... [id=neeha-s3-st3-bucket]
aws_s3_bucket.myblock: Destroying... [id=terraform-20230727073441533700000001]
aws_ebs_volume.myblockvolume: Destroying... [id=vol-03314323e0e729ad3]
aws_s3_bucket.mybucket: Destruction complete after 1s
aws_s3_bucket.myblock: Destruction complete after 1s
aws_ebs_volume.myblockvolume: Still destroying... [id=vol-03314323e0e729ad3, 10s elapsed]
aws_ebs_volume.myblockvolume: Destruction complete after 11s

Destroy complete! Resources: 3 destroyed.

C:\Terraform_Repository\Demo2d_s3_bucket_Statefile>_
```



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