

Infrastructure Automation using Terraform

Implementing Local Provisioner



Table of Contents

Prerequisite:.....	3
Walkthrough:	3
Part 1: Initializing Terraform Directory	3
Part 2: Implement Local Provisioner	6
Part 3: Destroy Resources	6

Infrastructure Automation using Terraform – Lab Guide

This Activity demonstrates the creation of S3 Bucket in AWS using Terraform. The Local Provisioner executes commands on the machine running Terraform.

Prerequisite:

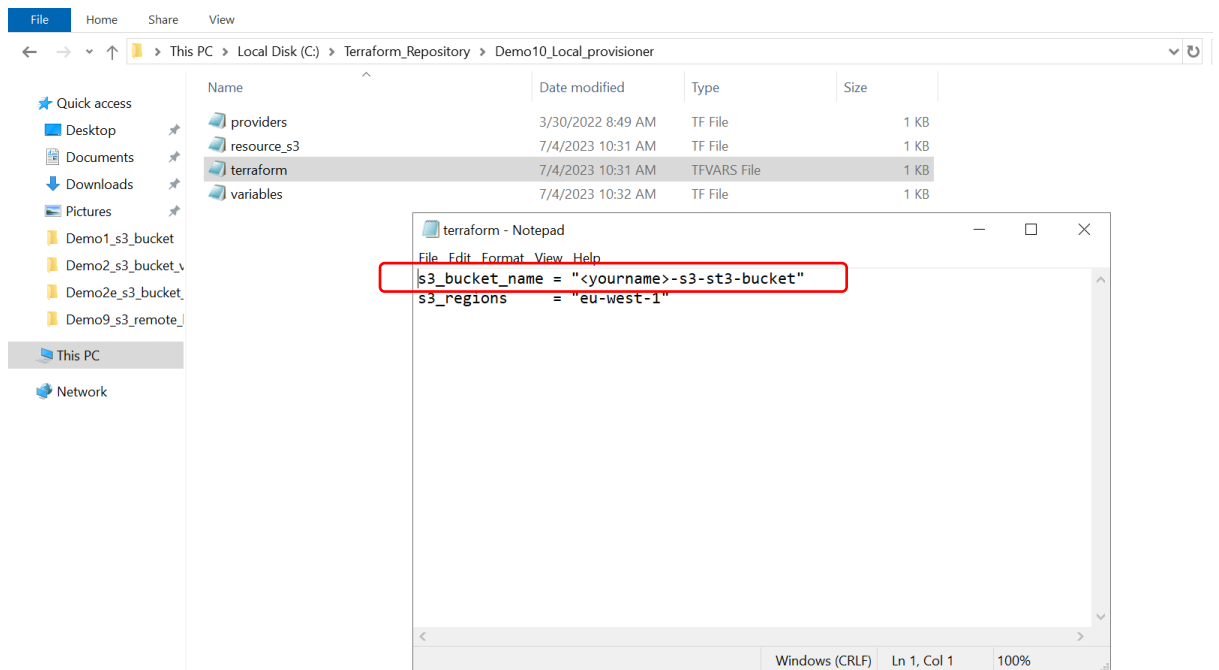
- 1) Download the zip file shared by the trainer and extract it.

Walkthrough:

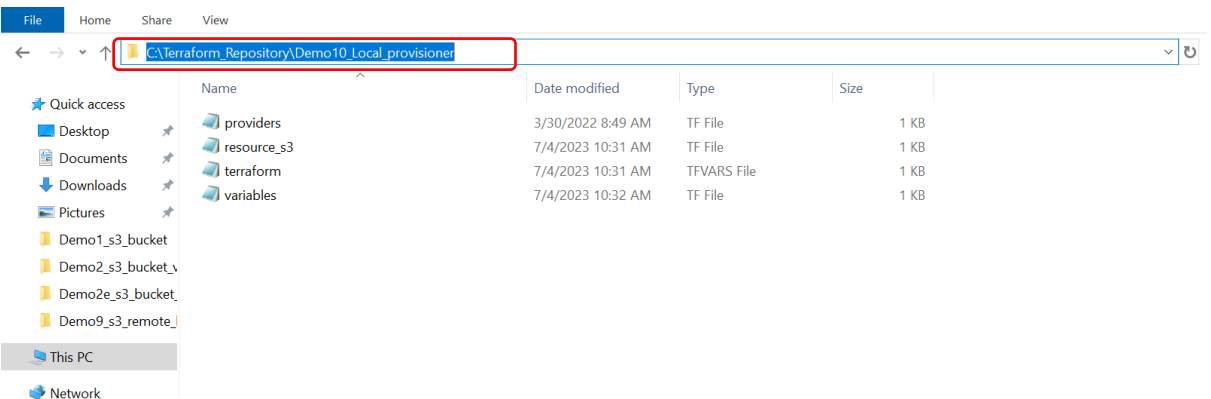
1. Initializing Terraform Directory
2. Implement Local Provisioner
3. Destroy Resources

Part 1: Initializing Terraform Directory

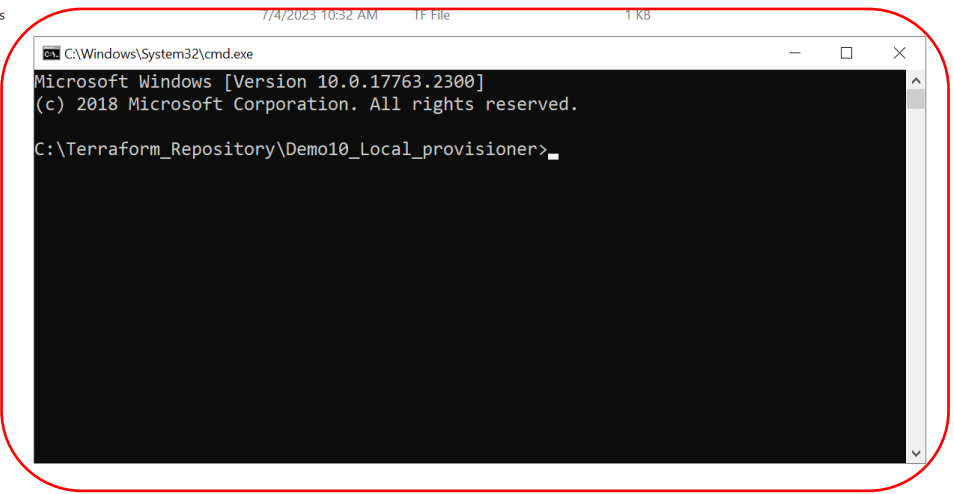
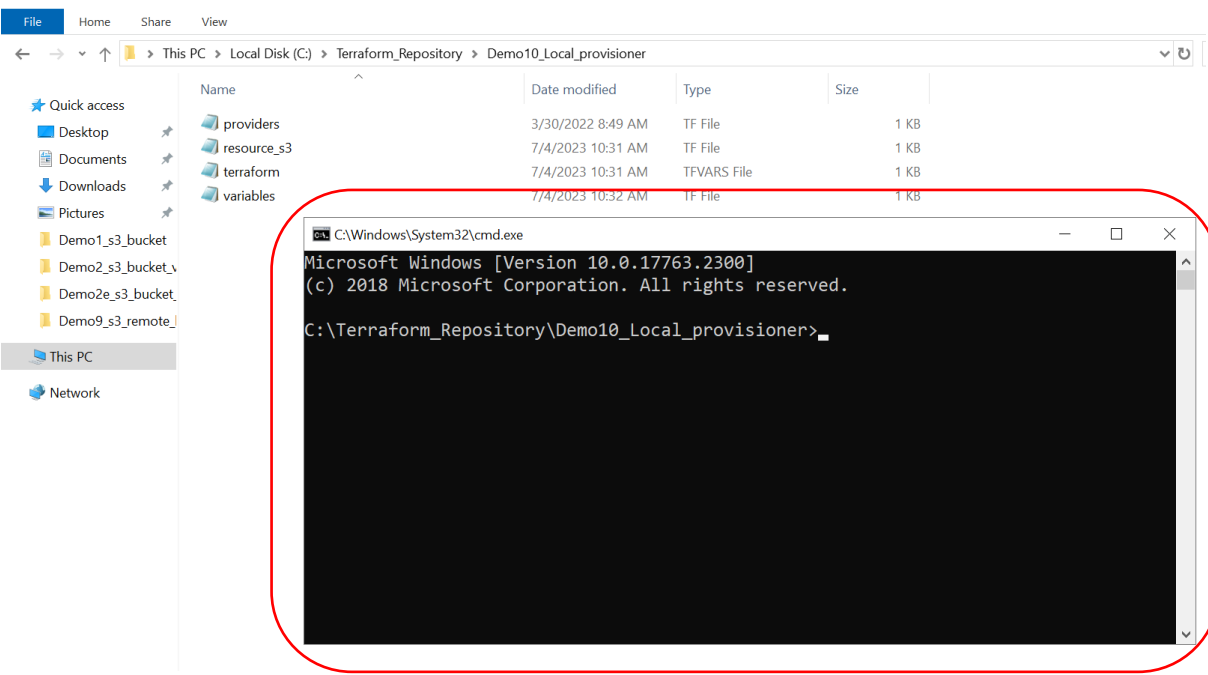
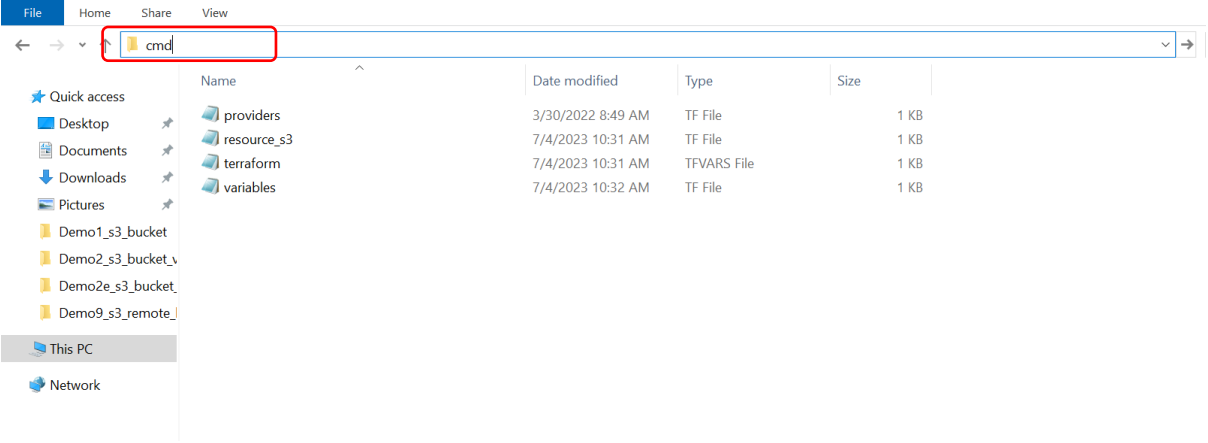
- 1 Open the extracted folder and navigate to “.tf” files. Open “terraform.tfvars” and update the “s3_bucket_name” argument.



- 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



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\Demo10_Local_provisioner>terraform init
```

Result :

	<pre> C:\Terraform_Repository\Demo10_Local_provisioner>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\Demo10_Local_provisioner>_ </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\Demo10_Local_provisioner>terraform validate </pre> <p>Result :</p> <pre> C:\Terraform_Repository\Demo10_Local_provisioner>terraform validate Success! The configuration is valid. C:\Terraform_Repository\Demo10_Local_provisioner>_ </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\Demo10_Local_provisioner>terraform plan -out "local_prov.tfplan" </pre> <p>Result :</p>

Infrastructure Automation using Terraform – Lab Guide

	<pre>C:\Terraform_Repository\Demo10_Local_provisioner>terraform plan -out "local_prov.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_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" } + tags_all = { + "env" = "dev" } + website_domain = (known after apply) }</pre>
--	--

Part 2: Implement Local Provisioner

1	<p>For creating resources , execute below command and observe the actions performed by the command.</p> <p>Command :</p> <pre>C:\Terraform_Repository\Demo10_Local_provisioner>terraform apply "local_prov.tfplan"</pre> <p>Result :</p> <pre>C:\Terraform_Repository\Demo10_Local_provisioner>terraform apply "local_prov.tfplan" aws_s3_bucket.mybucket: Creating... aws_s3_bucket.mybucket: Provisioning with 'local-exec'... aws_s3_bucket.mybucket (local-exec): Executing: ["cmd" "/C" "echo neeha-s3-st3-bucket >> private_ips.txt"] aws_s3_bucket.mybucket: Provisioning with 'local-exec'... aws_s3_bucket.mybucket (local-exec): Executing: ["cmd" "/C" "echo neeha-s3-st3-bucket.s3.amazonaws.com >> private_ips.txt"] aws_s3_bucket.mybucket: Creation complete after 4s [id=neeha-s3-st3-bucket] Apply complete! Resources: 1 added, 0 changed, 0 destroyed. Outputs: bucketname = "neeha-s3-st3-bucket" C:\Terraform_Repository\Demo10_Local_provisioner></pre>
---	--

Part 3: 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\Demo10_Local_provisioner>terraform destroy</pre> <p>Result:</p>
---	--

```
C:\Terraform_Repository\Demo10_Local_provisioner>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"
  } -> null

  - grant {
    - id           = "e1fe41990fbdc91adb1ec737a1727e2747f9af00f7975646b349aa05f802964e" -> null
    - permissions = [
```

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

Changes to Outputs:

```
- 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:

```
- 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.

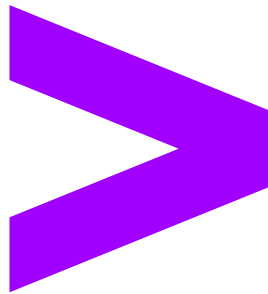
Enter a value: yes

```
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\Demo10_Local_provisioner>
```



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