

Infrastructure Automation using Terraform

Import existing S3 Bucket



Table of Contents

Prerequisite:.....	3
Walkthrough:	3
Part 1: Initializing Terraform Directory	3
Part 2: Import Existing S3 Bucket.....	5
Part 3: Destroying Imported S3 Bucket.....	7

Infrastructure Automation using Terraform – Lab Guide

This Activity demonstrates how to import existing S3 Bucket in AWS using Terraform. By doing this, we can manage existing resources using Terraform commands. For performing a successful import, we need to define a sample resource block manually and then create State file for the imported resource. As part of Re-verifying, we will run “terraform plan” and “terraform apply” to check whether the information stored in “.tf” files , State file is matching the existing object. If not, we need to update “.tf” files to ensure that we have the proper configuration for the imported resource.

Prerequisite:

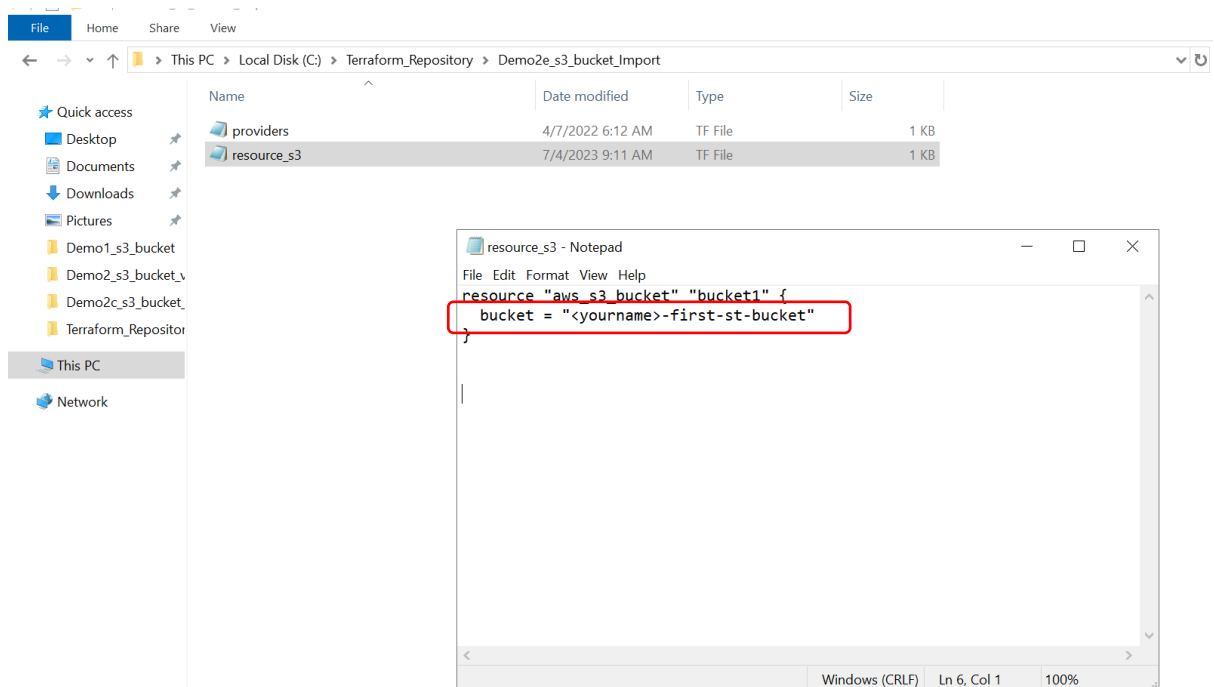
- 1) Download the zip file shared by the trainer and extract it.

Walkthrough:

1. Initializing Terraform Directory
2. Import existing S3 Bucket
3. Destroying Imported S3 Bucket

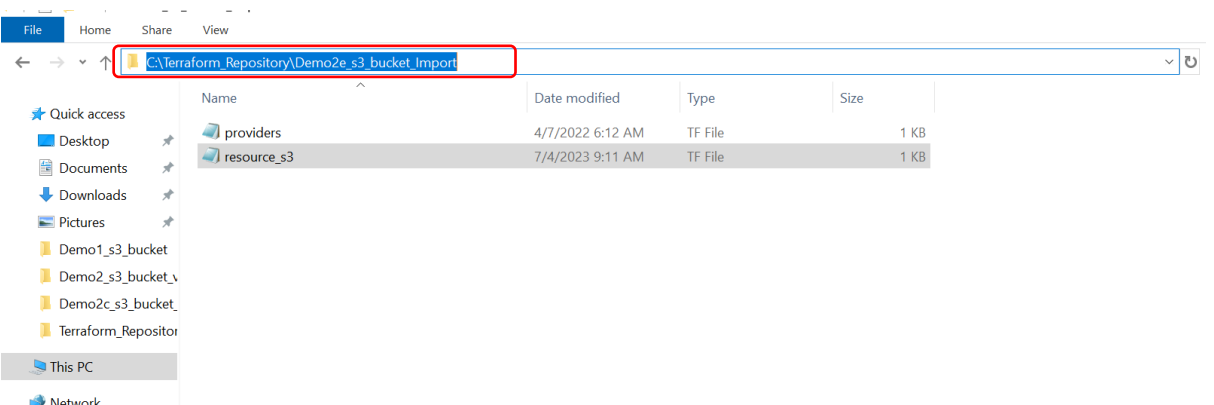
Part 1: Initializing Terraform Directory

- 1 Open the extracted folder and navigate to “.tf” files. Open “resource_s3.tf” and update the “bucket” argument.

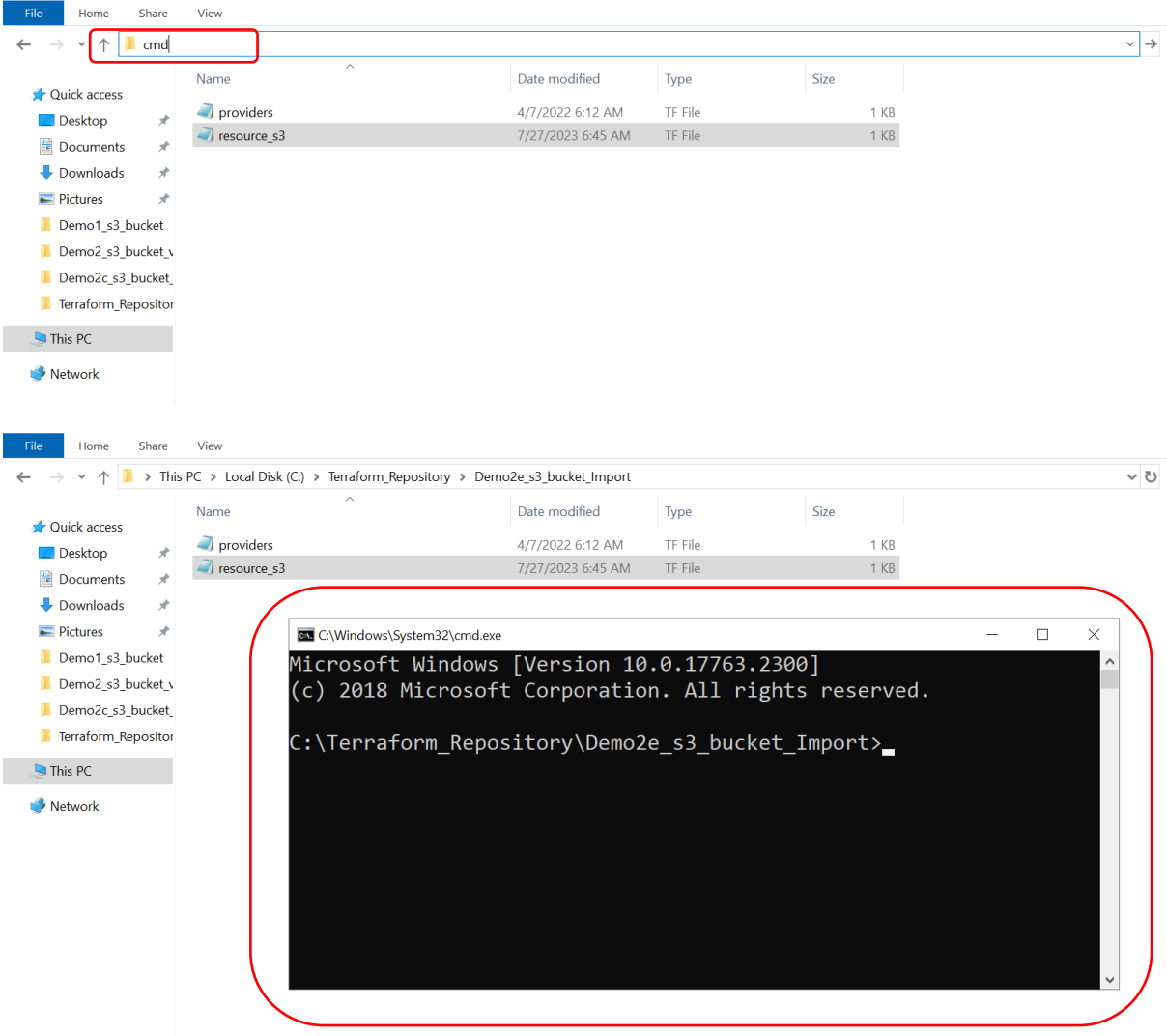


NOTE : The S3 Bucket which you want to import, must already exist in the Cloud.

- 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	<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\Demo2e_s3_bucket_Import>terraform init</pre> <p>Result :</p>

	<pre> C:\Terraform_Repository\Demo2e_s3_bucket_Import>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\Demo2e_s3_bucket_Import>_ </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\Demo2e_s3_bucket_Import>terraform validate </pre> <p>Result :</p> <pre> C:\Terraform_Repository\Demo2e_s3_bucket_Import>terraform validate Success! The configuration is valid. C:\Terraform_Repository\Demo2e_s3_bucket_Import>_ </pre>

Part 2: Import Existing S3 Bucket

1	<p>Now to manage existing resources using Terraform, we need to import it. Execute below command to import existing S3 bucket.</p> <p>Command :</p> <pre> C:\Terraform_Repository\Demo2e_s3_bucket_Import>terraform import "aws_s3_bucket.bucket1" neeha-first-st-bucket </pre> <p>Result :</p> <pre> C:\Terraform_Repository\Demo2e_s3_bucket_Import>terraform import "aws_s3_bucket.bucket1" neeha-first-st-bucket aws_s3_bucket.bucket1: Importing from ID "neeha-first-st-bucket"... aws_s3_bucket.bucket1: Import prepared! Prepared aws_s3_bucket for import aws_s3_bucket.bucket1: Refreshing state... [id=neeha-first-st-bucket] 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. C:\Terraform_Repository\Demo2e_s3_bucket_Import> </pre>
---	--

- 2 After Importing the existing resources, we need to check whether information in “.tf” file , State file and existing resource are same or not. Execute terraform plan and observe if terraform is trying to do any modifications.

Command :

```
C:\Terraform_Repository\Demo2e_s3_bucket_Import>terraform plan
```

Result :

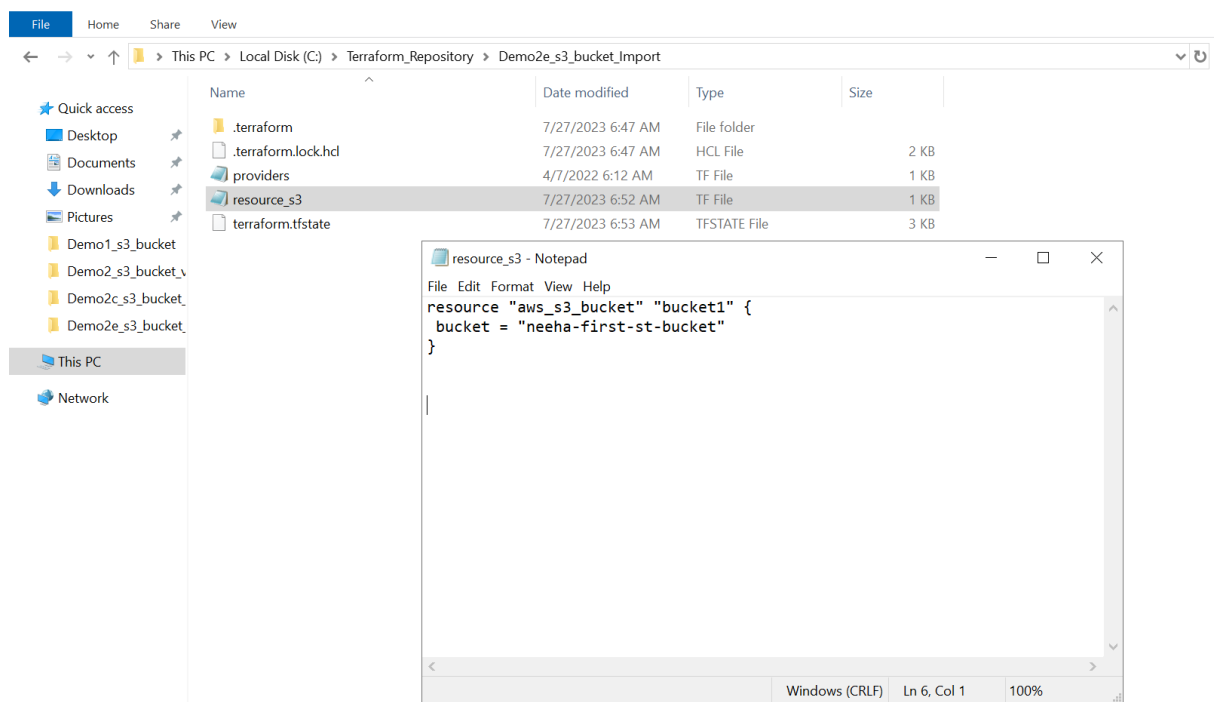
```
C:\Terraform_Repository\Demo2e_s3_bucket_Import>terraform plan
aws_s3_bucket.bucket1: Refreshing state... [id=neeha-first-st-bucket]

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

Terraform will perform the following actions:

# aws_s3_bucket.bucket1 must be replaced
-/+ resource "aws_s3_bucket" "bucket1" {
  + acceleration_status      = (known after apply)
  + acl                      = (known after apply)
  ~ arn                     = "arn:aws:s3:::neeha-first-st-bucket" -> (known after apply)
  ~ bucket                  = "neeha-first-st-bucket" -> "neeha-sample-first-st-bucket" # forces replacement
  ~ bucket_domain_name      = "neeha-first-st-bucket.s3.amazonaws.com" -> (known after apply)
  + bucket_prefix           = (known after apply)
  ~ bucket_regional_domain_name = "neeha-first-st-bucket.s3.eu-west-1.amazonaws.com" -> (known after apply)
  + force_destroy           = false
  ~ hosted_zone_id          = "Z1BKCTXD74EZPE" -> (known after apply)
  ~ id                     = "neeha-first-st-bucket" -> (known after apply)
  ~ object_lock_enabled      = false -> (known after apply)
  + policy                  = (known after apply)
  ~ region                  = "eu-west-1" -> (known after apply)
```

- 3 After inspecting above command's output, you will notice that terraform is trying to do some changes. But it's not what we want to achieve. So, we need to update the “.tf” file to match with the State file. Open “resource_s3.tf” file and update the name. Change bucket attribute value from “neeha-sample-first-st-bucket” to “neeha-first-st-bucket”.



- 4 Now again run “terraform plan” and observe the output. We must repeat above step until we get this message as output after “terraform plan” command is executed - “No changes. Your infrastructure matches the configuration”.

Command:

	<pre>C:\Terraform_Repository\Demo2e_s3_bucket_Import>terraform plan</pre> <p>Result:</p> <pre>No changes. Your infrastructure matches the configuration. Your configuration already matches the changes detected above. If you'd like to update the Terraform state to match, create and apply a refresh-only plan: terraform apply -refresh-only C:\Terraform_Repository\Demo2e_s3_bucket_Import>_</pre>
5	<p>Execute below command to ensure that you can manage imported resource using terraform commands.</p> <p>Command:</p> <pre>C:\Terraform_Repository\Demo2e_s3_bucket_Import>terraform apply</pre> <p>Result:</p> <pre>C:\Terraform_Repository\Demo2e_s3_bucket_Import>terraform apply aws_s3_bucket.bucket1: Refreshing state... [id=neeha-first-st-bucket] 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. Apply complete! Resources: 0 added, 0 changed, 0 destroyed. C:\Terraform_Repository\Demo2e_s3_bucket_Import>_</pre>

Part 3: Destroying Imported S3 Bucket

1	<p>Execute below command to destroy the S3 Bucket which we have imported previously. 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\Demo2e_s3_bucket_Import>terraform destroy</pre> <p>Result:</p>
---	--

```
C:\Terraform_Repository\Demo2e_s3_bucket_Import>terraform destroy
aws_s3_bucket.bucket1: Refreshing state... [id=neeha-first-st-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.bucket1 will be destroyed
- resource "aws_s3_bucket" "bucket1" {
  - arn                = "arn:aws:s3:::neeha-first-st-bucket" -> null
  - bucket             = "neeha-first-st-bucket" -> null
  - bucket_domain_name = "neeha-first-st-bucket.s3.amazonaws.com" -> null
  - bucket_regional_domain_name = "neeha-first-st-bucket.s3.eu-west-1.amazonaws.com" -> null
  - hosted_zone_id     = "Z1BKCTXD74EZPE" -> null
  - id                 = "neeha-first-st-bucket" -> null
  - object_lock_enabled = false -> null
  - policy              = jsonencode(
    {
      - Statement = [
        - {
          - Action      = "s3:*"
          - Condition = {
            - Bool = {
              - aws:SecureTransport = "false"

```

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:

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.

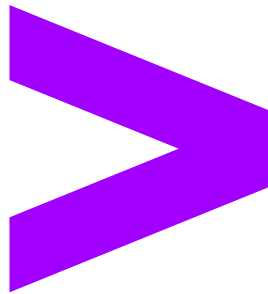
Enter a value: yes

aws_s3_bucket.bucket1: Destroying... [id=neeha-first-st-bucket]

aws_s3_bucket.bucket1: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.

C:\Terraform_Repository\Demo2e_s3_bucket_Import>_



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