

Infrastructure Automation using Terraform

Implementing Data Source on S3 Bucket



Table of Contents

Prerequisite:.....	3
Walkthrough:	3
Part 1: Initializing Terraform Directory	3
Part 2: Implementing Data Source on S3 Bucket	6
Part 3: Destroy the Implementation	6

Infrastructure Automation using Terraform – Lab Guide

This Activity demonstrates the Implementation of Data Source in Terraform. Data Sources are used for read-only operation i.e., It can be used to retrieve information regarding infrastructure which is already existing in the cloud. It will not create anything, but it can be used to export information which can be reused within “*.tf” files.

Prerequisite:

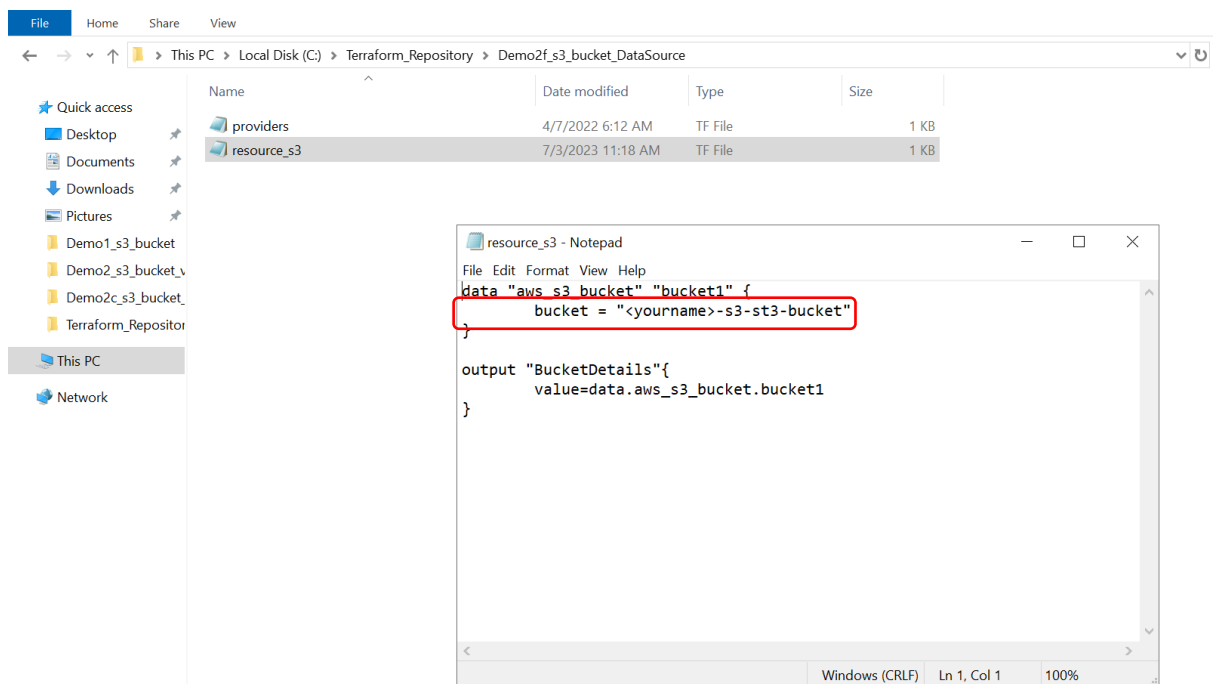
- 1) Download the zip file shared by the trainer and extract it.

Walkthrough:

1. Initializing Terraform Directory
2. Implementing Data Source using S3 Bucket
3. Destroy the Implementation

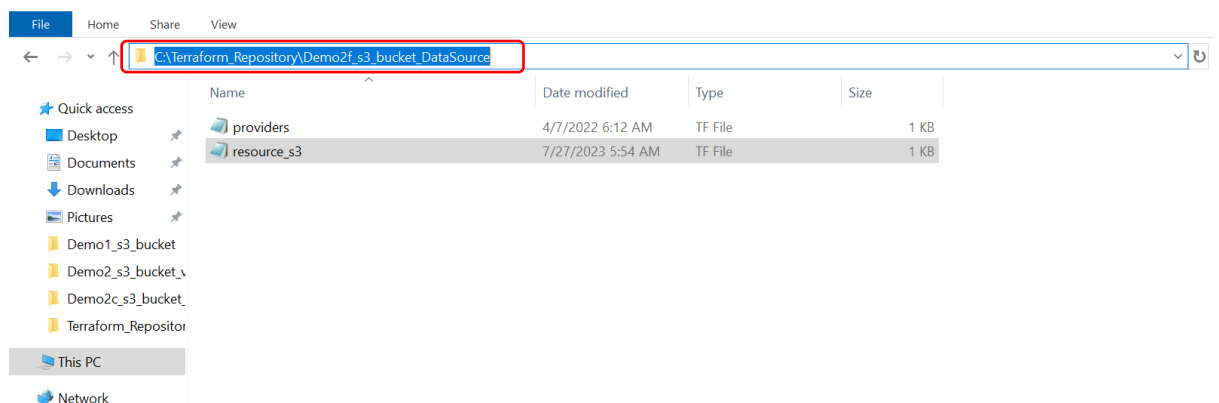
Part 1: Initializing Terraform Directory

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

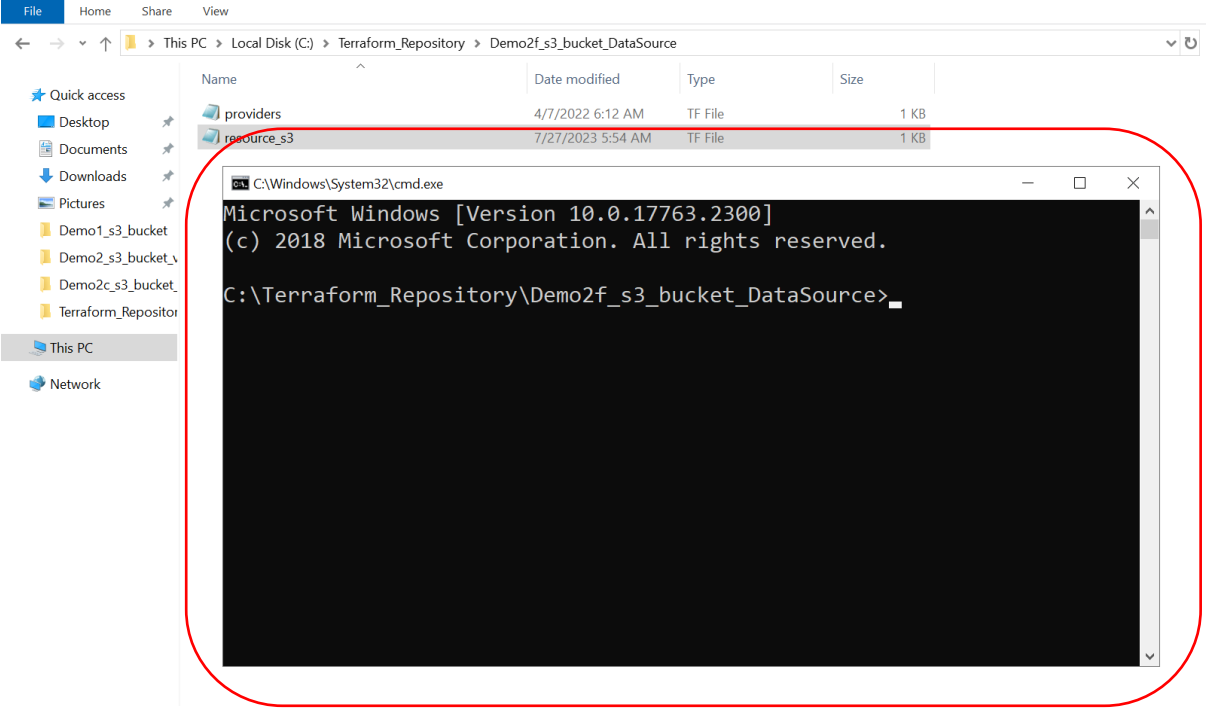
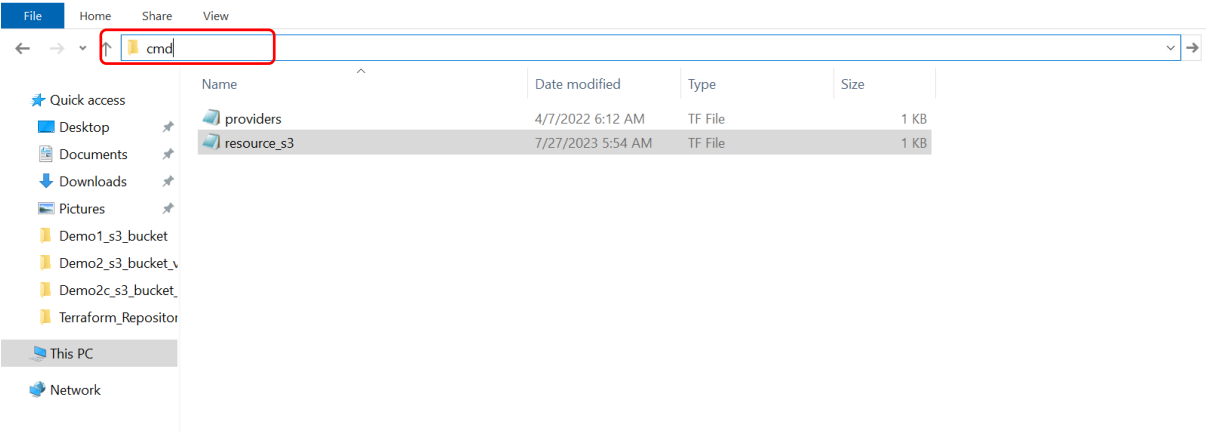


NOTE : The S3 Bucket name which you are updating in above file, make sure that bucket already exists in the AWS cloud. Otherwise you will not be able to export bucket attributes using Data Source block.

- 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\Demo2f_s3_bucket_DataSource>terraform init
```

Result :

	<pre> C:\Terraform_Repository\Demo2f_s3_bucket_DataSource>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\Demo2f_s3_bucket_DataSource>_ </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\Demo2f_s3_bucket_DataSource>terraform validate </pre> <p>Result :</p> <pre> C:\Terraform_Repository\Demo2f_s3_bucket_DataSource>terraform validate Success! The configuration is valid. C:\Terraform_Repository\Demo2f_s3_bucket_DataSource>_ </pre>
5	<p>Next run below command and observe the output. The output contains exported information of the bucket. You can make use of this according to your requirements.</p> <p>Command :</p> <pre> C:\Terraform_Repository\Demo2f_s3_bucket_DataSource>terraform plan -out "s3_data.tfplan" </pre> <p>Result :</p>

	<pre>C:\Terraform_Repository\Demo2f_s3_bucket_DataSource>terraform plan -out "s3_data.tfplan"</pre> <p>Changes to Outputs:</p> <pre>+ BucketDetails = { + arn = "arn:aws:s3:::neeha-s3-st3-bucket" + bucket = "neeha-s3-st3-bucket" + bucket_domain_name = "neeha-s3-st3-bucket.s3.amazonaws.com" + bucket_regional_domain_name = "neeha-s3-st3-bucket.s3.eu-west-1.amazonaws.com" + hosted_zone_id = "Z1BKCTXD74EZPE" + id = "neeha-s3-st3-bucket" + region = "eu-west-1" + website_domain = null + website_endpoint = null }</pre> <p>You can apply this plan to save these new output values to the Terraform state, without changing any real infrastructure.</p> <hr/> <p>Saved the plan to: s3_data.tfplan</p> <p>To perform exactly these actions, run the following command to apply:</p> <pre>terraform apply "s3_data.tfplan"</pre> <pre>C:\Terraform_Repository\Demo2f_s3_bucket_DataSource>_</pre>
--	---

Part 2: Implementing Data Source on S3 Bucket

1	<p>Execute below command and observe the output. It is showing all the attribute values of the bucket which we have successfully exported using Data Source.</p> <p>Command :</p> <pre>C:\Terraform_Repository\Demo2f_s3_bucket_DataSource>terraform apply "s3_data.tfplan"</pre> <p>Result :</p> <pre>C:\Terraform_Repository\Demo2f_s3_bucket_DataSource>terraform apply "s3_data.tfplan"</pre> <p>Apply complete! Resources: 0 added, 0 changed, 0 destroyed.</p> <p>Outputs:</p> <pre>BucketDetails = { "arn" = "arn:aws:s3:::neeha-s3-st3-bucket" "bucket" = "neeha-s3-st3-bucket" "bucket_domain_name" = "neeha-s3-st3-bucket.s3.amazonaws.com" "bucket_regional_domain_name" = "neeha-s3-st3-bucket.s3.eu-west-1.amazonaws.com" "hosted_zone_id" = "Z1BKCTXD74EZPE" "id" = "neeha-s3-st3-bucket" "region" = "eu-west-1" "website_domain" = toString(null) "website_endpoint" = toString(null) }</pre> <pre>C:\Terraform_Repository\Demo2f_s3_bucket_DataSource>_</pre> <p>NOTE: We didn't create anything new in the cloud. So, it's showing "Resources: 0 added, 0 changed, 0 destroyed." Data Source will just pull the bucket attributes (read-only operation).</p>
---	--

Part 3: Destroy the Implementation

1	<p>Execute below command to destroy the above implementation. 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 , provide "yes".</p> <p>Command:</p>
---	--

```
C:\Terraform_Repository\Demo2f_s3_bucket_DataSource>terraform destroy
```

Result:

```
C:\Terraform_Repository\Demo2f_s3_bucket_DataSource>terraform destroy
```

Changes to Outputs:

```
- BucketDetails = {
  - arn                = "arn:aws:s3:::neeha-s3-st3-bucket"
  - bucket             = "neeha-s3-st3-bucket"
  - bucket_domain_name = "neeha-s3-st3-bucket.s3.amazonaws.com"
  - bucket_regional_domain_name = "neeha-s3-st3-bucket.s3.eu-west-1.amazonaws.com"
  - hosted_zone_id     = "Z1BKCTXD74EZPE"
  - id                 = "neeha-s3-st3-bucket"
  - region             = "eu-west-1"
  - website_domain     = null
  - website_endpoint   = null
} -> null
```

You can apply this plan to save these new output values to the Terraform state, without changing any real infrastructure.

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:

Do you really want to destroy all resources?

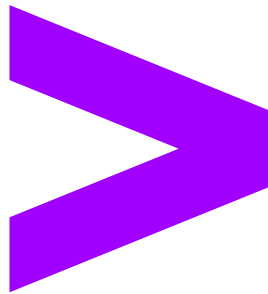
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

Destroy complete! Resources: 0 destroyed.

```
C:\Terraform_Repository\Demo2f_s3_bucket_DataSource>
```

NOTE: After executing “terraform destroy”, it will erase all the data which we have exported using Data Source block.



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