

Project Documentation

User Manual

BeatDrift-CLI Automation Tool Implementation

COSC595
Trimester 1 2022

Jason O. Aboh
Paldeep Kaur
Sumit Khokhar

Table of Contents

1.0 Project Description

1.1 Client

1.2 Project Team

2.0 Stakeholders

3.0 Installation Manual

3.0.1 Download Terraform on Windows

3.0.2 Integrating Terraform in Visual Studio

3.0.3 Extension used in Visual Studio to run Terraform script

4.0. Terraform as Infrastructure as Code

4.1. Terraform Providers

4.2. Terraform variables

5.0 Terraform Configuration

6.0 Terraform CLI

6.1 Terraform State

6.2 Terraform Refresh

6.3 Terraform Plan

1.0. Project Description

The project is oriented towards creating a CLI tool which will be used to build a pipeline to address drift in an Infrastructure as Code (IaC) environment.

1.1. Client

Client: Mark Wallis, Director of Hunter Orbit Pty Ltd.

Email: mark@hunterorbit.com.au

1.2. Project Team

The team members are:

Name	Email id (s)
1. Jason O.Aboh	jaboh@myune.edu.au , bigboss107@gmail.com
2. Paldeep Kaur	pkaur21@myune.edu.au , paldeepkaur1995@gmail.com
3. Sumit Khokhar	skhokha2@myune.edu.au , sumitkhokhar359@gmail.com

2.0. Stakeholders

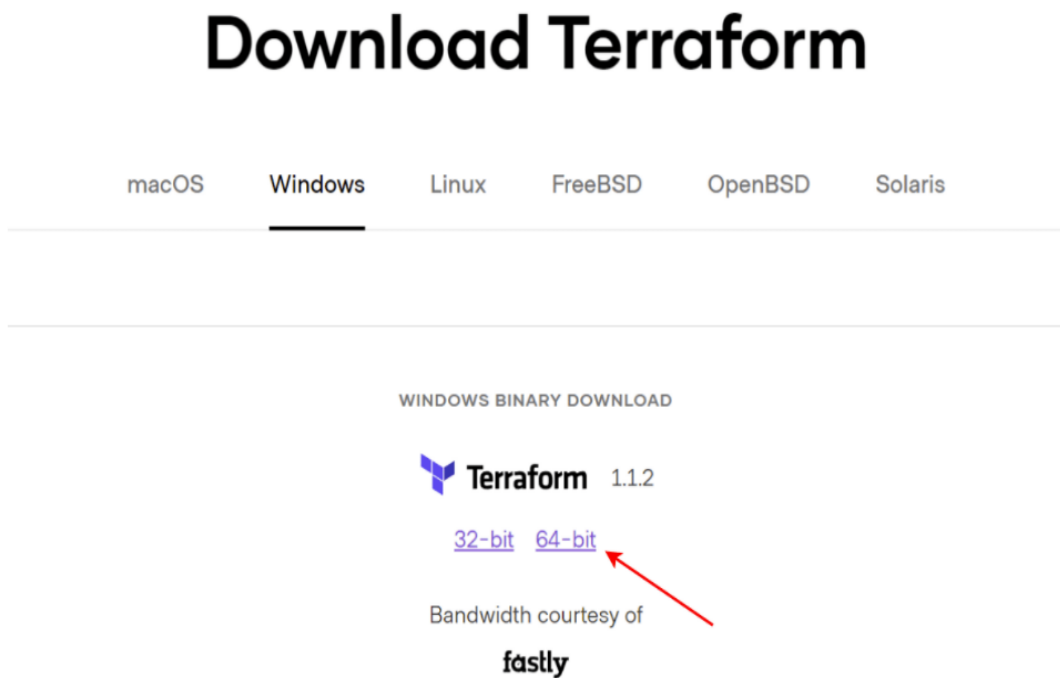
The stakeholders of the project are as follows:

- Client: **Mark Wallis**
- Unit Coordinator (COSC 595): **Dr. Edmund Sadgrove**
- Team Member (Project Lead): **Jason O. Aboh**
- Team Member: **Sumit Khokhar**
- Team Member: **Paldeep Kaur**

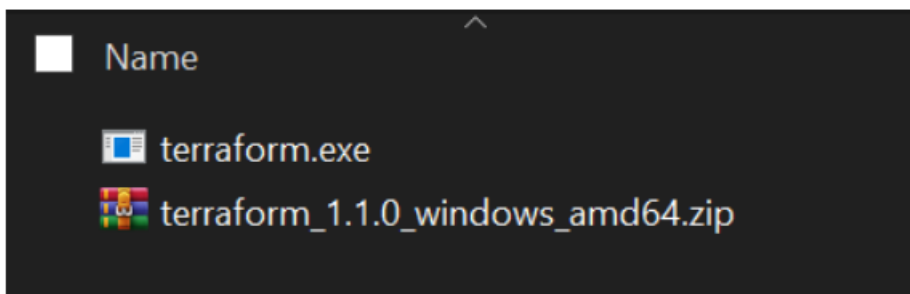
3.0. Installation Manual

3.0.1. Download Terraform on Windows

Step 1: Downloading Terraform for Windows from the link <https://www.terraform.io/downloads>



Step 2: The configuration file will get downloaded in .zip and it is required to extract the file in the system.



Step 3: Updation of Path environment variables

In the start, menu need to search for the “Environment variables” option and in the “Environment variable page need to click on the edit option as shown below.

User variables for amlan

Variable	Value

New...

Edit...

Delete

System variables

Variable	Value
Path	C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Wi...

New...

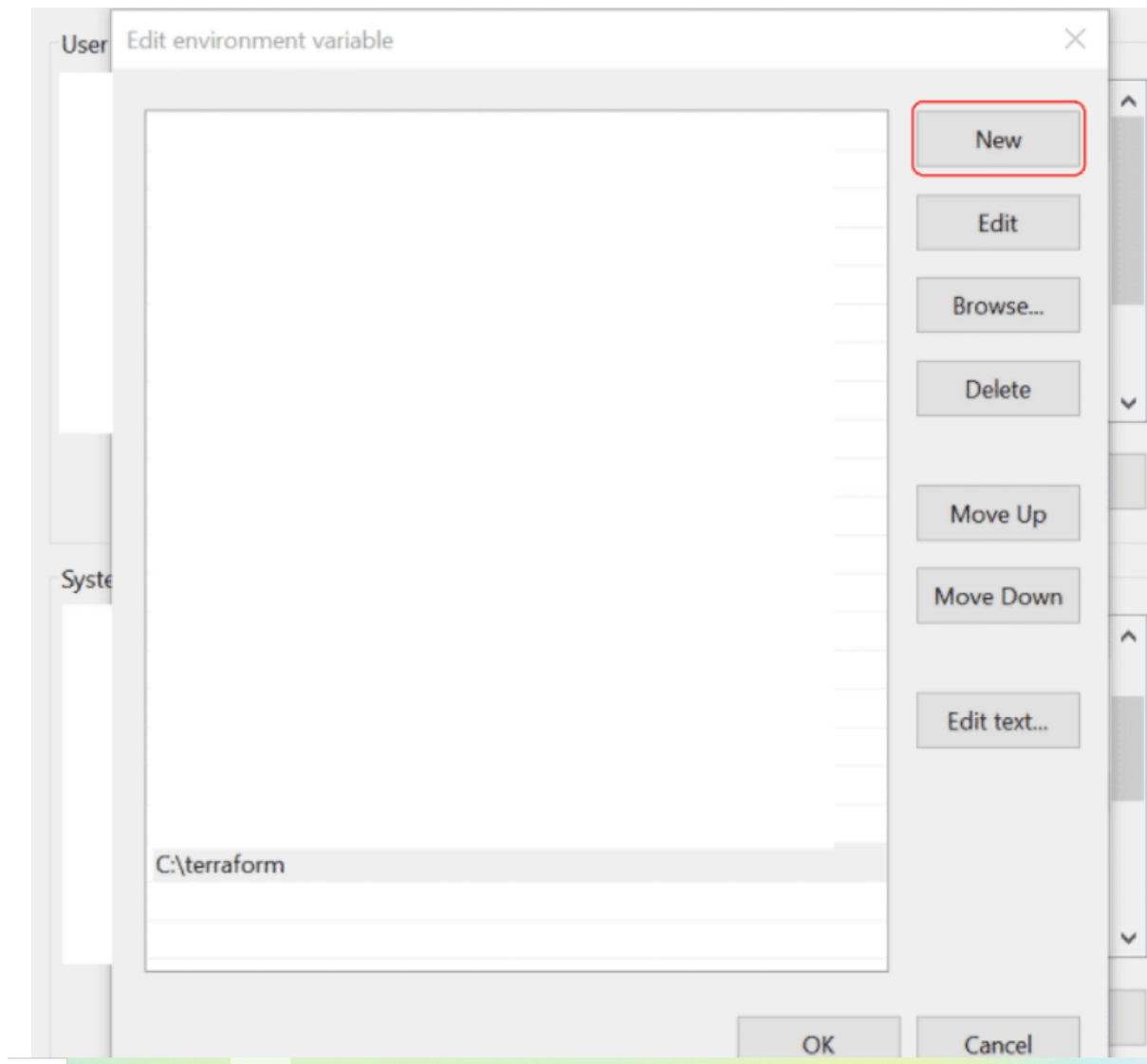
Edit...

Delete

OK

Cancel

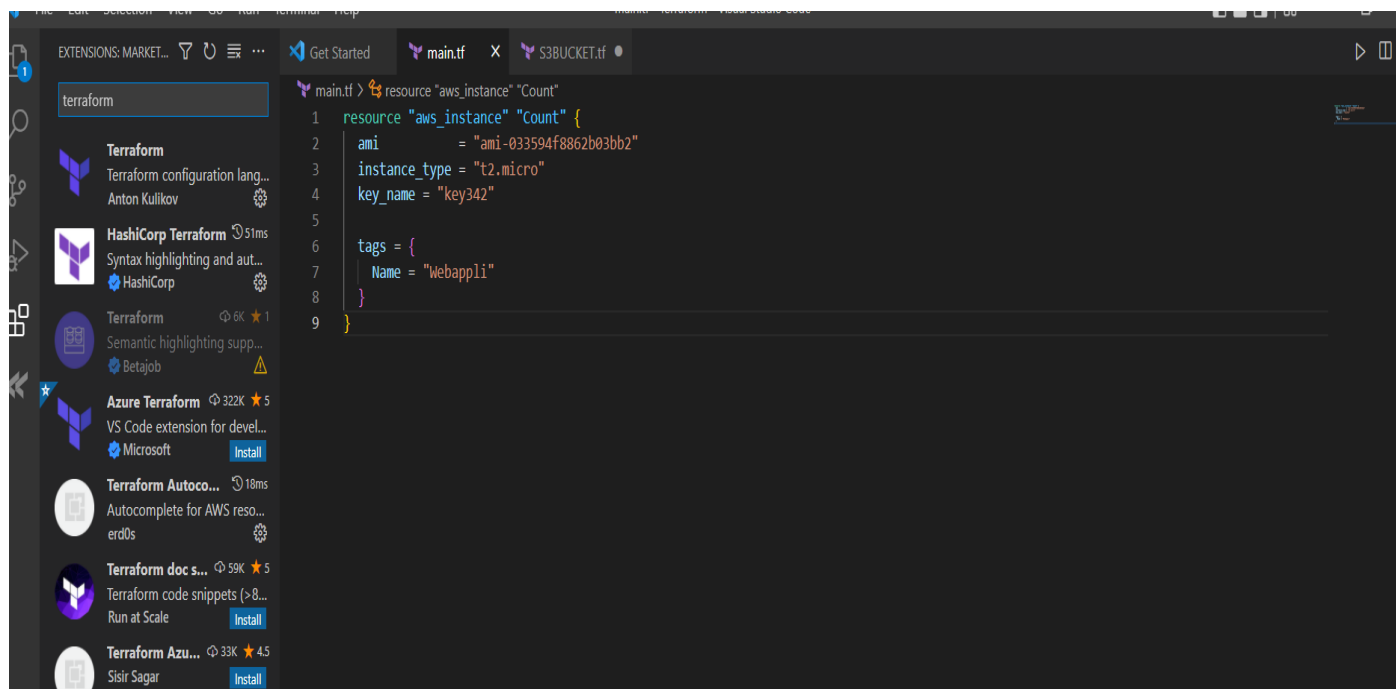
After clicking on Edit need to click on New to add the path where the configuration file of Terraform is installed.



Step 4: Open the command prompt to see whether the correct version of Terraform is installed or not.

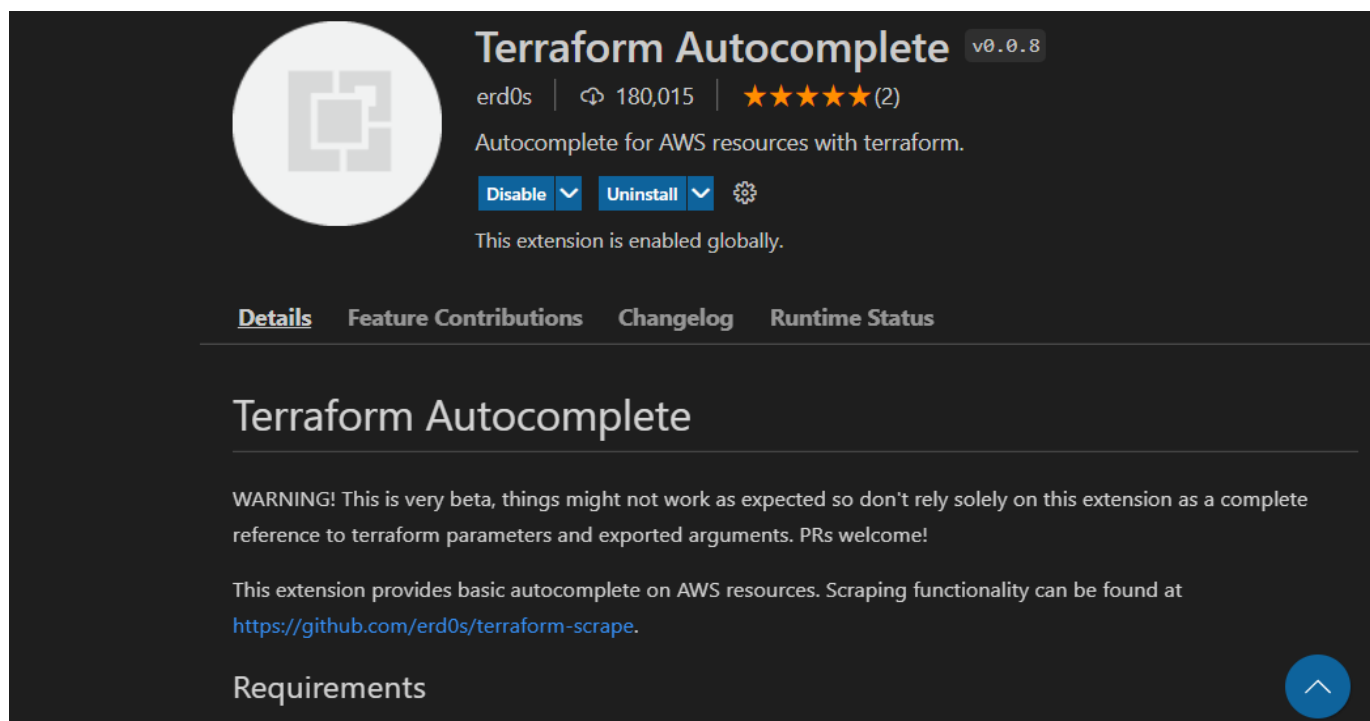
```
C:\Users\amlan>terraform --version
Terraform v1.1.0
on windows_amd64
```

3.0.2. Integrating Terraform in Visual Studio



3.0.3 Extension used in Visual Studio to run Terraform script

Terraform Autocomplete



HashiCorp Terraform



HashiCorp Terraform v2.23.0

HashiCorp | 1,804,163 | ★★★★★ (146)

Syntax highlighting and autocompletion for Terraform

[Disable](#) [Uninstall](#) ⚙️

This extension is enabled globally.

[Details](#) [Feature Contributions](#) [Changelog](#) [Runtime Status](#)

Terraform Extension for Visual Studio Code


The HashiCorp [Terraform Extension for Visual Studio Code \(VS Code\)](#) with the [Terraform Language Server](#) adds editing features for [Terraform](#) files such as syntax highlighting, IntelliSense, code navigation, code formatting, module explorer and much more!

Quick Start

Get started writing Terraform configurations with VS Code in three steps:

- **Step 1:** If you haven't done so already, install [Terraform](#)

Python Extension to run Terraform CLI



Python v2022.8.0

Microsoft | 58,188,552 | ★★★★★ (479)

IntelliSense (Pylance), Linting, Debugging (multi-threaded, remote), Jupyter Notebooks, code formatting.

[Disable](#) [Uninstall](#) [Switch to Pre-Release Version](#) ⚙️

This extension is enabled globally.

[Details](#) [Feature Contributions](#) [Changelog](#) [Extension Pack](#) [Runtime Status](#)

Python extension for Visual Studio Code

A Visual Studio Code extension with rich support for the [Python language](#) (for all [actively supported versions](#) of the language: ≥ 3.7), including features such as IntelliSense (Pylance), linting, debugging, code navigation, code formatting, refactoring, variable explorer, test explorer, and more!

Support for [vscode.dev](#)

The Python extension does offer [some support](#) when running on [vscode.dev](#) (which includes [github.dev](#)). This includes partial IntelliSense for open files in the editor.

Installed extensions

The Python extension will automatically install the [Pylance](#) and [Jupyter](#) extensions to give you the best

Categories

- Programming Languages
- Debuggers
- Linters
- Formatters
- Other
- Data Science
- Machine Learning
- Notebooks

Extension Resources

- [Marketplace](#)
- [Repository](#)
- [License](#)

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** JUPYTER

PS C:\Users\Paldeep Kaur\python_terraform>

4.0 Terraform as Infrastructure as a Code

Terraform is an open source infrastructure as a code platform created by Hashicorp. Terraform is a coding tool that helps developers to use a high definition language known as HCL (Hashicorp Configuration Language) to describe the infrastructure for an application. Terraform application tool is used in those cases where the developers use a multi-cloud or hybrid cloud environment.

As the main aim of the project is to detect the drift in the CLI environment when comparing the resources of AWS and Terraform and while comparing the resources of AWS and Terraform it is very important to understand the Infrastructure as a Code environment. Generally the Infrastructure as a code environment supports drifting and so for this the Infrastructure as a Code environment such as AWS and Terraform applications are being used to detect the drift in the CLI environment. Apart from the benefit of drifting that the Infrastructure as a Code environment possess, there are other benefits also that the Infrastructure as a Code environment possess and these are Improving in Speed, Improve reliability, preventing configuration drift and also supporting experimentation, testing and optimisation.

Thus, the benefits of Infrastructure as a Code in detail are explained as below:

- **Speed:** In Infrastructure as a Code environment the automation is faster and hence there is always the increase in Speed. Speed is an important factor in detecting drift in Infrastructure as Code environment.
- **Reliability:** There is always reliability while using Infrastructure as Code because with Infrastructure as Code the resources are configured exactly.
- **Prevent Configuration drift:** Configuration drift happens when the configuration provisioned for the environment does not match with the actual environment. Infrastructure as a Code environment prevents the configuration drift.
- **Supports testing optimisation and other experimentation features:** Infrastructure as a Code environment supports testing of different environments, validation of different resources and other features.

4.1 Terraform Providers

There are different Terraform Providers that are plug-ins which help in implementing resource types. Apart from the implementation of different resources Terraform providers contain different codes that are used to connect different services. The resources of the terraform providers are used to provision infrastructure.

4.2 Terraform Variables

There are different types of variables that are used in Terraform provider and these variables are as follows:

- **Local variables :** In Terraform the local variables are declared within the local block and specifically the local variables are used to provide values to the attributes. The local variables are used with the help of “local” keyword. Local variables are used to create any instance in aws with the help of Terraform provider. In the case of implementing Local variables it can be said that the local variables did not depend on any cloud provider (Ninawe,2021).
- **Input variables:** Input variables have similarity with that of local variables and it can also be said that Input variables are used as resource attributes. Regarding Input variables it can be said that the main function of the input variables is associated with the modules. Multiple data types are supported by Input variables (Ninawe, 2021).
- **Environment variables:** Environment variables are used in crawling different resources in Terraform. Terraform uses different environment variables such as TF_log, TF_CLI_ARGS, TF_DATA_DIR (Ninawe, 2021)

5.0 Terraform Configuration:

A Terraform configuration is a complete document in the Terraform language that tells Terraform how to manage a given collection of infrastructure. A configuration can consist of multiple files and directories. The syntax of the Terraform language consists of only a few basic elements:

```
resource "aws_vpc" "main" {  
  cidr_block = var.base_cidr_block  
}  
  
<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>" {  
  # Block body  
  <IDENTIFIER> = <EXPRESSION> # Argument  
}
```

- Blocks are containers for other content and usually represent the configuration of some kind of object, like a resource. Blocks have a block type, can have zero or more labels, and have a body that contains any number of arguments and nested blocks. Most of Terraform's features are controlled by top-level blocks in a configuration file.
- Arguments assign a value to a name. They appear within blocks.

- Expressions represent a value, either literally or by referencing and combining other values. They appear as values for arguments, or within other expressions.

6.0 Terraform CLI:

Terraform helps in managing drift with the help of the following features and the features present in Terraform involves the following:

- **Terraform State:** Terraform helps in tracking the state of the resources in Terraform (koehler, 2018)
- **Terraform Refresh :** The refresh feature in Terraform helps in restoring drift with the help of a state file which is generated when any instance is created in aws with the help of terraform script(koehler, 2018)
- **Terraform Plan :** Terraform Plan is used for execution purposes of instances in Terraform and restoring configuration matching with the real world state(koehler, 2018).
- **Terraform Config:** The configuration feature in Terraform helps in managing the drift in Terraform(koehler, 2018).

6.1 Terraform State

The main functions of the Terraform state file are as follows:

- Mapping the resources present in the configuration with the real world resources.
- Tracking metadata present in the resources.
- Managing large infrastructure in Terraform.

Terraform show command is used to visualise the current state of the configuration(koehler, 2018).

After successful creation of an instance in AWS, Terraform show command is used in displaying the internal configuration of the instance that got created in AWS with the help of Terraform provider.

```
ami                = "ami-033594f8862b03bb2"
arn                = "arn:aws:ec2:us-east-1:238541900632:instance/i-04112fb501a83927d"
associate_public_ip_address = true
availability_zone   = "us-east-1a"
cpu_core_count      = 1
cpu_threads_per_core = 1
disable_api_termination = false
ebs_optimized       = false
get_password_data    = false
hibernation          = false
id                  = "i-04112fb501a83927d"
instance_initiated_shutdown_behavior = "stop"
instance_state       = "running"
```

Image 17: Internal Configuration (EC2 instance)

The above Image3 specifies the internal configuration of EC2 instance which was generated by executing terraform show command in Visual studio while running Terraform script. With the help of the configuration file attached in the above image the detailed structure of the configuration is visualised while creating EC2 instance in AWS using Terraform provider.

6.2. Terraform Refresh

The refresh feature is used in Terraform to update the contents of the state file which matches with the real world status. The refresh operation is executed in Terraform by running **terraform refresh** command. By running refresh will not modify the infrastructure but it only modifies and configures the state file. Refresh helps in detecting the drift in Terraform when there is a change in the state file compared to that of the last time (koehler, 2018).

By doing a refresh, if in any case the state file is corrupted or damaged then it can be retrieved by running **terraform.tfstate.backup** (koehler, 2018).

6.3. Terraform Plan

Terraform Plan helps in creating all of the resources in the configuration. Running Terraform Plan helps in creating the plan which helps in predicting what changes will be there in the infrastructure. The main goal of the Terraform Plan is to compare the configuration file with the current state file and to read the output which matches with the current configuration. When every time the Terraform plan gets executed, the resource life cycle configuration runs for every resource (2021). The Resource life cycle consists of different flags and the flags are as follows:

prevent_destroy: The prevent_destro flag provides an extra protection to the destruction of any resource (2021).

ignore_changes: The ignoe_changes flag help to identify which attributes need to be ignored when applying the changes (2021).

create_before_destroy: The create_before_destroy flag is used to enable the replacement of the resource which is created before the deployment of the original instance(2021).

Note: When running Terraform plan, if it predicts that there are no changes required to be made in the infrastructure then it will be displayed as “ Your infrastructure matches the configuration”.

```
PS C:\Users\saiika\OneDrive\Desktop\Terraform> terraform plan
aws_s3_bucket.demo-bucket: Refreshing state... [id=terraform-20220609225651146800000001]
aws_instance.Count: Refreshing state... [id=i-04112fb501a83927d]

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.
PS C:\Users\saiika\OneDrive\Desktop\Terraform>
```

Image 18: Output of Terraform Plan

From the above Image 4, it is visualised that already S3 bucket and EC2 instance are created inside Terraform environment and hence the message is getting displayed that the infrastructure matches the configuration and no changes are required.

References:

- <https://www.ibm.com/cloud/learn/terraform#toc-what-is-terraform>
- Christie Koehler - Jun 07, 2018 - Detecting and Managing drift with Terraform Retrieved from <https://www.hashicorp.com/blog/detecting-and-managing-drift-with-terraform>
- (February 26, 2021) - Terraform Drift Detection and Performance Retrieved from <https://blog.anupamyaadav.in/2021/02/terraform-drift-detection-and-management.html>
- Sumeet Ninawe - (Aug 10 , 2021) - How to use Terraform variables (Locals, Input, Output, Environment) <https://spacelift.io/blog/how-to-use-terraform-variables>