

Terraform - Hands-On - Practice Assessment 1

Part 1 - Multiple Choice Questions: (*Highlight the correct answer in bold*)

1. What language is Terraform using?

b. HCL

2. Terraform can be run on which of the following operating systems?

d. All the above

3. Is Terraform available as a single executable binary?

a. Yes

4. What file extension is used for Terraform configuration file?

a. .tf

5. Which of the following is NOT a text editor for creating Terraform files?

d. Microsoft Word

6. Which of these is NOT a Terraform command?

b. Compile

7. Which command is used to initialize a working directory containing Terraform configuration files?

a. terraforms init

8. Before running `terraform apply`, which command should be executed to see the planned actions?

b. terraforms plan.

9. Terraform's `plan` command is used for what purpose?

c. To preview changes.

10. The command to find Terraform's version is:

b. `terraform version`.

11. What is the purpose of the `terraform show` command?

a. To display the current state or saved plan.

12. Which of the following is a valid Terraform resource type?

d. All the above

13. What is the `terraform destroy` command used for?

a. To remove all previously created infrastructure.

14. What is Terraform mainly used for?

b. Infrastructure as Code

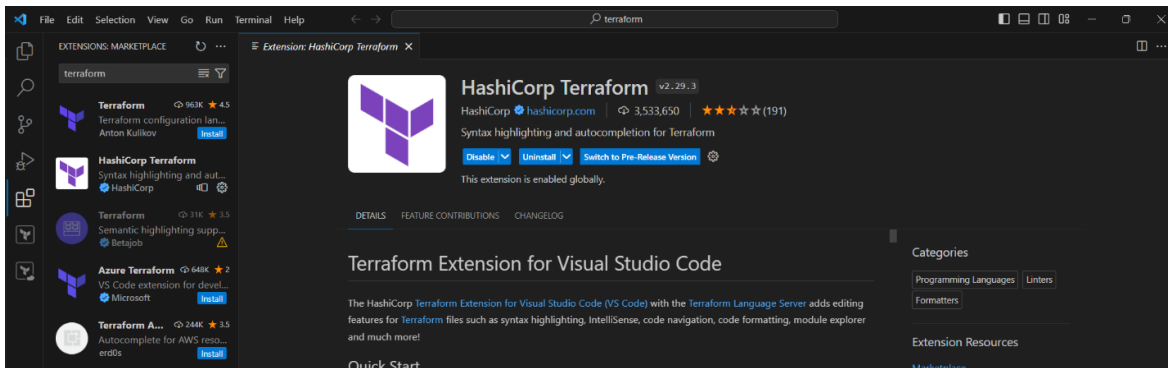
15. Which file is used by Terraform to track the current state of the infrastructure?

a. `terraform.tfstate`

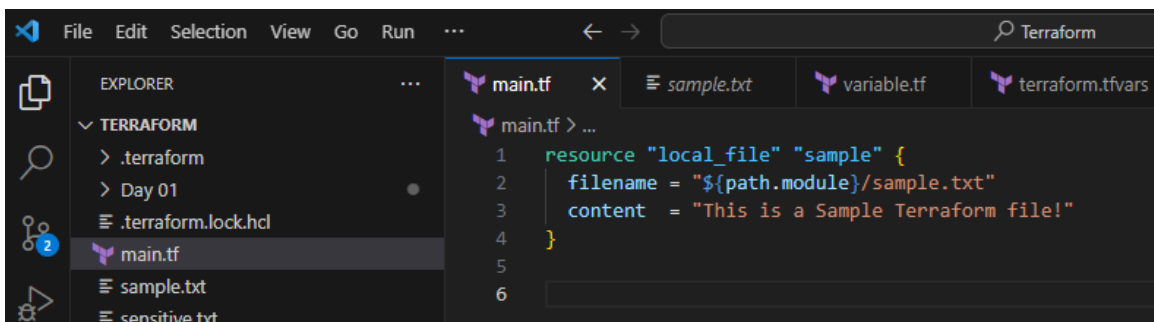
Part 2 – Hands-On Labs

Lab 1: Setting Up a Terraform Project in Visual Studio Code

- **Install Visual Studio Code**
 - If you do not already have Visual Studio Code, download and install it from the official website.
- **Install Terraform Extension in VS Code**
 - Open Visual Studio Code.
 - Go to Extensions
 - Search for "Terraform" and install the extension by HashiCorp.

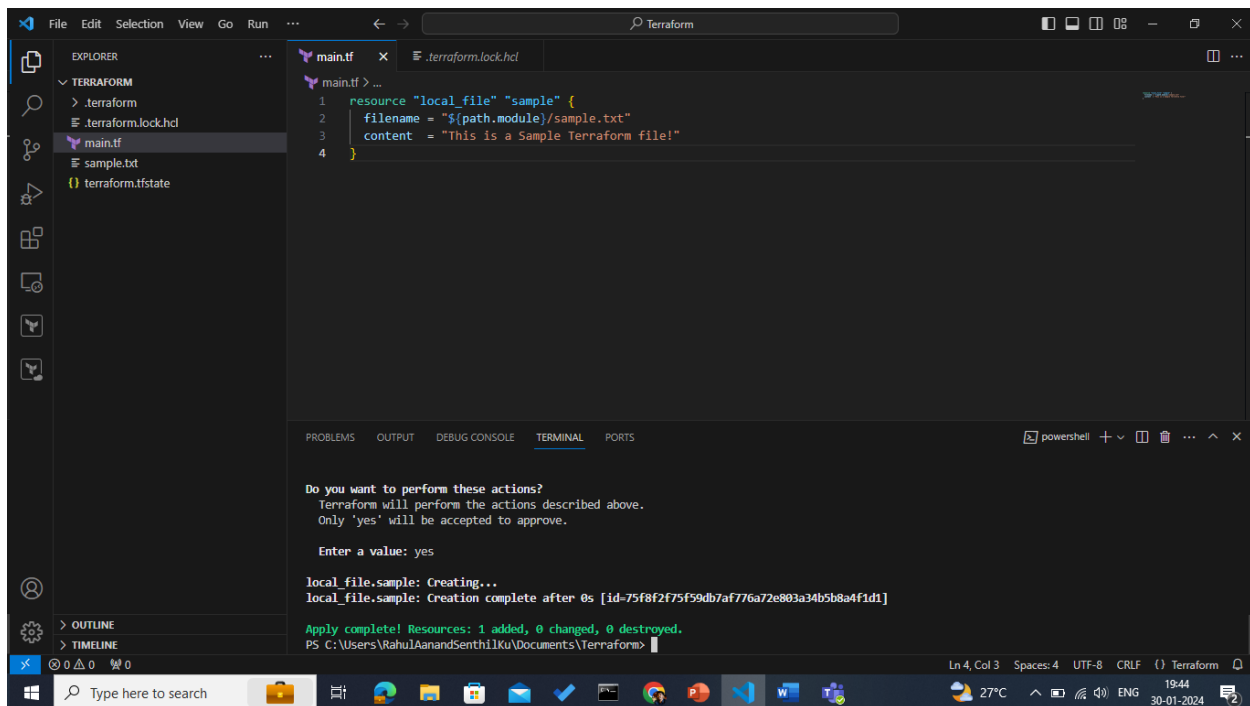


- **Create a New Project Folder**
 - Create a new folder on your computer where you will store your Terraform files.
 - Open this folder in Visual Studio Code (File > Open Folder).
- **Initialize a New Terraform Configuration File**
 - Create a new file in the folder with the `.tf` extension, for example, `main.tf`.
 - Write a simple Terraform configuration or leave it blank for now.



Lab 2: Basic Local File Operation

- **Define a Local File Resource**
 - In **main.tf**, start by defining a resource to create a local file. For example:
- **Initialize Terraform**
 - Open the terminal in VS Code (Terminal > New Terminal).
 - Run **terraform init** to initialize the Terraform project. This command sets up Terraform to run your configuration.
- **Apply Configuration**
 - Run **terraform apply** to apply your configuration.
 - Confirm the action in the terminal when prompted.
 - This step will create a file named **sample.txt** with the content "Hello, Terraform!" in your project directory.



The screenshot shows the Visual Studio Code interface with a Terraform project. The Explorer sidebar on the left shows the project structure: **TERRAFORM** (expanded), **.terraform**, **.terraform.lock.hcl**, **main.tf** (selected), **sample.txt**, and **terraform.tfstate**. The main editor displays the **main.tf** file with the following content:

```
1 resource "local_file" "sample" {
2   filename = "${path.module}/sample.txt"
3   content  = "This is a Sample Terraform file!"
4 }
```

Below the editor, the **TERMINAL** panel shows the output of the **terraform apply** command:

```
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

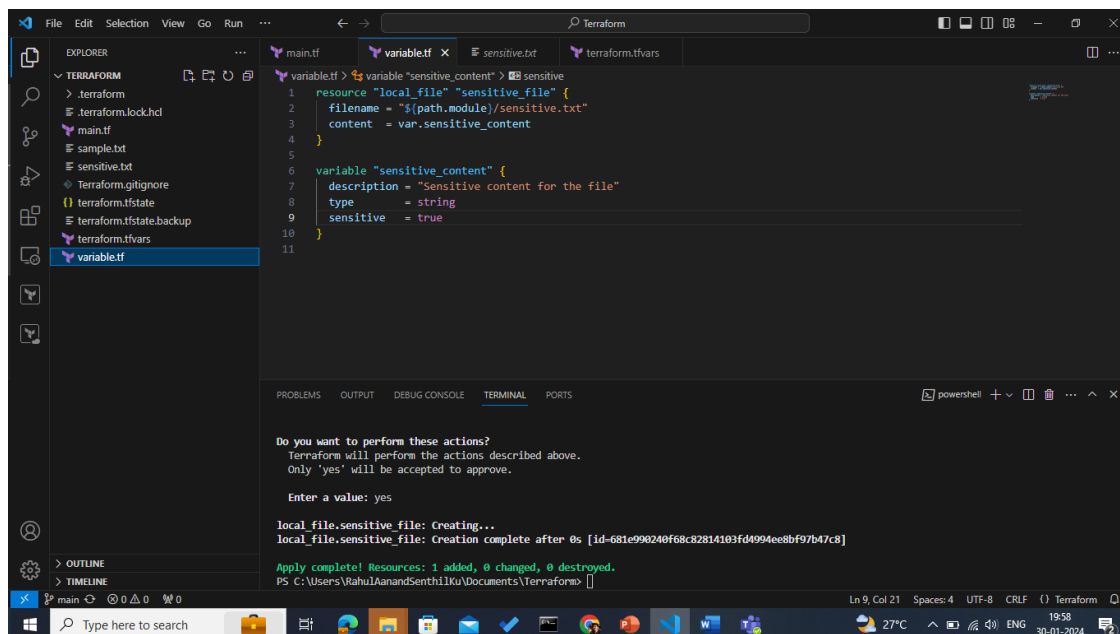
local_file.sample: Creating...
local_file.sample: Creation complete after 0s [id=75f8f2f75f59db7af776a72e803a34b5b8a4f1d1]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Rahu\AanandSenthil\Documents\Terraform>
```

The status bar at the bottom indicates the current cursor position (Ln 4, Col 3), encoding (UTF-8), line ending (CRLF), and the active file (Terraform).

Lab 3: Handling Sensitive File Operations

- **Create a Sensitive File Resource**
 - Now, let us handle a sensitive file operation. For example, you might want to create a file that contains sensitive information.
 - In `main.tf`, add a new resource block:
- **Define Variables**
 - Create a new file named `variables.tf` and define a variable for the sensitive content:
- **Add Sensitive Content**
 - Create a `terraform.tfvars` file to store the value of the sensitive content.
 - Add your sensitive content in `terraform.tfvars`, like:
- **Re-run Terraform Apply**
 - Run `terraform apply` again in your terminal.
 - Confirm the action when prompted.
 - Terraform will now create another file named `sensitive.txt` with the sensitive content, and it will treat the content as sensitive in its output.
- **Verify the Files**
 - Check your project directory. You should see two new files: `sample.txt` and `sensitive.txt`, each with the specified content.



The screenshot shows a Visual Studio Code editor with a Terraform project. The Explorer pane on the left shows the file structure: `TERRAFORM` directory containing `.terraform`, `.terraform.lock.hcl`, `main.tf`, `sample.txt`, `sensitive.txt`, `Terraform.gitignore`, `terraform.tfstate`, `terraform.tfstate.backup`, `terraform.tfvars`, and `variable.tf`. The `variable.tf` file is selected and open in the editor. It contains the following code:

```
1 resource "local_file" "sensitive_file" {
2   filename = "${path.module}/sensitive.txt"
3   content  = var.sensitive_content
4 }
5
6 variable "sensitive_content" {
7   description = "Sensitive content for the file"
8   type        = string
9   sensitive   = true
10 }
11
```

The terminal at the bottom shows the output of a Terraform apply command. It prompts for confirmation to perform actions, and the user enters 'yes'. The output shows that the `local_file.sensitive_file` resource is being created and is complete after 0s. The final output is:

```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Rahul\Documents\Terraform>
```

