Demo: Using AI Tools like GitHub Copilot to Automatically Generate Terraform Configuration Scripts Without Access to EC2

Objective:

In this demo, we will use GitHub Copilot to automatically generate a Terraform configuration script that sets up infrastructure. Instead of deploying to real instances (like EC2), we will simulate the configuration locally using the Terraform tool. This is a basic setup to demonstrate how AI tools can assist in writing Infrastructure as Code (IaC) without requiring access to actual cloud infrastructure.

Steps:

Step 1: Set Up Your Environment

Objective: Prepare the environment by installing Terraform.

Instructions:

1. Install Terraform:

   - Download Terraform from the official website: [Terraform Download](https://www.terraform.io/downloads).

   - Follow the installation instructions for your operating system.

2. Create a Working Directory for Your Terraform Project:

   ```bash

   mkdir demo_terraform_project

   cd demo_terraform_project

   ```

Step 2: Generate a Basic Terraform Configuration Using GitHub Copilot

Objective: Use GitHub Copilot to automatically generate a basic Terraform configuration script.

Instructions:

1. Create a New File for Your Terraform Configuration:

  ```bash
  touch main.tf
  ```

2. Write a Prompt for GitHub Copilot:

  - Open `main.tf` in your text editor (e.g., Visual Studio Code).

  - At the top of the file, write the following comment to instruct GitHub Copilot to generate a Terraform configuration:

    ```hcl
     Generate a simple Terraform configuration for a local setup
    ```

3. Let GitHub Copilot Generate the Terraform Configuration:

  - GitHub Copilot will generate something like this:

    ```hcl
    provider "local" {}

    resource "local_file" "example" {
      content  = "Hello, Terraform!"
    ```

```
  filename = "${path.module}/hello_terraform.txt"
}
```

4. Explanation of Generated Code:

- Provider "local": This configures Terraform to interact with your local system, without needing access to cloud infrastructure (like EC2).

- local_file Resource: This creates a local file called `hello_terraform.txt` containing the text "Hello, Terraform!".

Step 3: Apply the Terraform Configuration

Objective: Use Terraform to apply the configuration and simulate infrastructure provisioning locally.

Instructions:

1. Initialize the Terraform Project:

```bash
terraform init
```

2. Apply the Terraform Configuration:

- This will create the local file as specified in the Terraform script.

```bash
terraform apply
```

- You will be prompted to confirm the action. Type `yes` to proceed.

3. Verify the Output:

- Once the configuration is applied, Terraform will create a local file `hello_terraform.txt` in your project folder with the following content:

```
Hello, Terraform!
```

4. Check the State File:

- Terraform maintains a state file (`terraform.tfstate`) to track the resources it has created. You can open this file to view details about the resources managed by Terraform.

Step 4: Modify the Configuration Using GitHub Copilot

Objective: Use GitHub Copilot to add more functionality to the Terraform configuration.

Instructions:

1. Write a Comment to Add More Local Resources:

- In `main.tf`, write a comment instructing Copilot to create another local file:

```hcl
Add another local file to demonstrate multiple resources
```

2. GitHub Copilot Suggestion:

- Copilot will generate something like:

```hcl
resource "local_file" "second_example" {
```

```
  content  = "This is the second file created by Terraform."
  filename = "${path.module}/second_terraform_file.txt"
}
```

3. Apply the Changes:

```bash
terraform apply
```

- Confirm the action by typing `yes`.

4. Verify the Output:

- Terraform will now create another file named `second_terraform_file.txt` in your project folder with the content specified.

Step 5: Clean Up the Resources

Objective: Clean up the local resources created by Terraform.

Instructions:

1. Destroy the Resources:

- To remove all resources created by Terraform, use the `destroy` command:

```bash
terraform destroy
```

- Confirm the action by typing `yes`.

2. Verify that the Files are Removed:

   - The files `hello_terraform.txt` and `second_terraform_file.txt` should now be deleted from your project folder.
