# **Leveraging GitHub Copilot for Secure Terraform Development on AWS**

## **Introduction**

In this exercise, we focus on using GitHub Copilot Chat to generate Terraform code that provisions AWS resources securely. By interacting solely with the Copilot Chat interface, you’ll learn how to prompt for code generation, review suggested code, and integrate best practices for secure infrastructure-as-code. This session covers:

* Initialization of a Terraform project with secure AWS provider settings.
* Creation of key AWS resources (VPC, EC2, S3, IAM) using secure configurations.
* Defining variables, tfvars, and outputs to maintain modularity and security.
* Best practices for secure development, including least privilege IAM, encryption, and safe variable management.

## **Objective**

After completing this exercise, you will be able to:

* **Utilize GitHub Copilot Chat:** Interact with Copilot Chat to generate Terraform code.
* **Implement Secure Infrastructure:** Create AWS resources with a focus on security best practices.
* **Adopt Secure Development Practices:** Apply principles like least privilege, encryption, and secure variable management.
* **Document and Validate:** Track resource outputs and ensure that your configuration meets production-grade security standards.

| **Note**: Below prompts are provided solely as examples of effective prompt construction. As participants, we must engage in an iterative process to refine these examples and develop the optimal prompt for the task at hand. |
| --- |

## **Step 1: Environment Setup**

**Objective:**Set up a new Terraform project workspace and configure your development environment entirely via GitHub Copilot Chat.

**Initialize the Workspace:**In Copilot Chat, instruct:

| @workspace: Initialize a new Terraform project named "secure\_terraform\_aws" using Terraform. Create the entire project structure automatically, and open it in VS Code with a dedicated .code-workspace file that includes standardized settings for code formatting and linting. |
| --- |

| **Note**: Above prompt would help you to generate the required commands. Once commands are created, hover on the commands and click on **Insert into Terminal** option |
| --- |

This prompt ensures that the project is scaffolded automatically with consistent workspace settings.

**Verify and Refine the Project Structure:**Next, instruct:

| @workspace: Provide a detailed overview of my Terraform project structure, including key files (e.g., main.tf, variables.tf, outputs.tf) and directories. |
| --- |

If you need additional folders, refine with:

| /fix: Update the project structure to include a "modules" folder for reusable components, a "docs" folder for documentation, and a "scripts" folder for automation tasks. |
| --- |

* This iterative refinement guarantees that the workspace follows best practices for scalability and organization.

**Challenges for Step 1:**

* Determine if extra directories (for configuration, environment scripts, etc.) are needed and adjust prompts until the workspace fully meets your organizational requirements.
* Simulate a multi-environment setup (development, staging, production) via additional prompts to refine the structure for larger projects.

**Step 2: Create the Terraform Configuration Files**

**Objective:**Generate and refine the core Terraform configuration that initializes the AWS provider with secure defaults, using GitHub Copilot Chat only.

**Generate the Basic Configuration:**In Copilot Chat, instruct:

| /start: Generate a Terraform configuration file (main.tf) that sets up the AWS provider securely. Ensure the configuration includes a required Terraform version (>= 1.0), a required providers block for "hashicorp/aws" with version "~> 4.0", and an AWS provider configuration that uses a variable (var.aws\_region) for the region. |
| --- |

* This detailed prompt guarantees that the basic configuration is produced with secure defaults and proper variable usage.

**Refine the Configuration:**Then instruct:

| /fix: Update the Terraform configuration to include secure settings, such as ensuring the provider uses HTTPS endpoints and adding documentation that explains the use of variables for sensitive configurations. |
| --- |

* This refinement ensures the configuration adheres to best practices for security and maintainability.

**Step 3: Secure Resource Provisioning**

**Objective:**Generate secure resource blocks for key AWS components—such as VPC, EC2 instance, S3 bucket, and IAM roles—using detailed GitHub Copilot Chat prompts.

**Detailed Process and Prompts:**

**Secure VPC:**Instruct Copilot Chat:

| /start: Generate a Terraform resource block for creating a custom VPC. Use a variable (var.vpc\_cidr) for the CIDR block, enable DNS support and hostnames, and tag the VPC as "SecureCustomVPC". |
| --- |

* **Security Group for EC2:**Then instruct:

| /start: Generate a Terraform resource block for an AWS security group that allows SSH access only from a specified CIDR (var.allowed\_ssh\_cidr) and allows HTTP access globally. Associate this security group with the custom VPC. |
| --- |

* **EC2 Instance:**Next, instruct:

| /start: Generate a Terraform resource block to deploy an EC2 instance in a public subnet. Use variables for the AMI ID (var.ami\_id) and instance type (var.instance\_type), and attach the previously defined security group. |
| --- |

* **Secure S3 Bucket:**Then instruct:

| /start: Generate a Terraform resource block for a secure S3 bucket. The bucket should have a private ACL, versioning enabled, and server-side encryption configured using AES256. Tag the bucket as "SecureAppBucket". |
| --- |

* **IAM Role and Policy:**Finally, instruct:

| /start: Generate a Terraform resource block for an IAM role for EC2 instances that follows the least privilege principle, including an inline policy granting read-only access to the S3 bucket. |
| --- |

**Step 4: Managing Variables and Secure Configurations**

**Objective:**Define all required variables, generate a tfvars file, and set up outputs to ensure modularity and secure configuration management.

**Detailed Process and Prompts:**

**Generate Variables File:**In Copilot Chat, instruct:

| /start: Generate a variables.tf file that defines variables for aws\_region, vpc\_cidr, ami\_id, instance\_type, s3\_bucket\_name, and allowed\_ssh\_cidr with detailed descriptions and appropriate default values. |
| --- |

* **Generate tfvars File:**Then instruct:

| /start: Generate a terraform.tfvars file that provides concrete values for the variables defined in variables.tf, such as "us-west-2" for aws\_region and realistic defaults for the other variables. |
| --- |

* **Generate Outputs File:**Finally, instruct:

| /start: Generate an outputs.tf file that defines outputs for key resource information, such as the VPC ID, EC2 instance public IP, and S3 bucket ARN. |
| --- |

**Step 5: Secure Development Best Practices**

**Objective:**Ensure that your Terraform code follows security best practices for secure AWS deployments.

**Detailed Process and Prompts:**

**Sensitive Variable Management:**In Copilot Chat, instruct:

| /start: Generate best practices for managing sensitive variables in Terraform, emphasizing the use of environment variables or secret management tools instead of hardcoding values in variables.tf. |
| --- |

* **Enforce Least Privilege:**Then instruct:

| /start: Generate recommendations for configuring AWS IAM roles in Terraform using the principle of least privilege. Provide guidelines for structuring inline policies. |
| --- |

* **Emphasize Encryption and Versioning:**Finally, instruct:

| /start: Generate a summary of why enabling server-side encryption and versioning on S3 buckets is essential for data protection and recovery. |
| --- |

**Step 6: Finalize and Validate the Terraform Configuration**

**Objective:**Review, refine, and validate your complete Terraform configuration to ensure it meets production-grade security standards.

**Detailed Process and Prompts:**

**Review the Configuration:**In Copilot Chat, instruct:

| @workspace: Provide a comprehensive summary of all generated Terraform files (main.tf, variables.tf, terraform.tfvars, outputs.tf) for the secure AWS deployment. |
| --- |

* **Validate Secure Settings:**Then instruct:

| /explain: Outline the steps to validate that the configuration adheres to security best practices, including least privilege IAM, encryption settings, and proper variable management. |
| --- |

**Step 7: Deploy and Validate the Terraform Configuration Using GitHub Copilot Chat**

**Objective:**Deploy your Terraform configuration from within VS Code, with all execution commands generated and run via GitHub Copilot Chat.

**Detailed Process and Prompts:**

**Initialize Terraform:**In Copilot Chat, instruct:

| @terminal: Generate and execute the command for initializing the Terraform configuration and downloading the required provider plugins. |
| --- |

| **Note**: Above prompt would help you to generate the required commands. Once commands are created, hover on the commands and click on **Insert into Terminal** option |
| --- |

**Generate a Terraform Plan:**Next, instruct:

| @terminal: Generate and execute the command for producing a detailed Terraform plan to show the planned changes for deploying AWS resources. |
| --- |

| **Note**: Above prompt would help you to generate the required commands. Once commands are created, hover on the commands and click on **Insert into Terminal** option |
| --- |

**Apply the Configuration:**Then, instruct:

| @terminal: Generate and execute the command for applying the Terraform configuration automatically, deploying all defined AWS resources without manual intervention. |
| --- |

| **Note**: Above prompt would help you to generate the required commands. Once commands are created, hover on the commands and click on **Insert into Terminal** option |
| --- |

**Validate the Deployment:**Finally, instruct:

| @terminal: Generate and execute the command for displaying Terraform outputs to verify key resource information such as the VPC ID, EC2 instance public IP, and S3 bucket ARN. |
| --- |

| **Note**: Above prompt would help you to generate the required commands. Once commands are created, hover on the commands and click on **Insert into Terminal** option |
| --- |

**Conclusion**

This exercise has illustrated how to generate secure, production-ready Terraform scripts using **GitHub Copilot Chat only**. By sending clear, descriptive prompts to Copilot Chat, we obtained code that adheres to security best practices—such as using least privilege IAM roles, encrypting S3 buckets, and restricting network access. Integrating these secure development practices ensures that your AWS deployments are robust, maintainable, and aligned with industry standards.

Happy coding and secure infrastructure provisioning with GitHub Copilot Chat!