# Lesson 11 Demo 03

# Validating Variables and Securing Secrets in Terraform Code

**Objective:** To validate variables and secure sensitive data within the Terraform code for enhanced configuration security and reliability

**Tools required:** Terraform, AWS, and Visual Studio Code

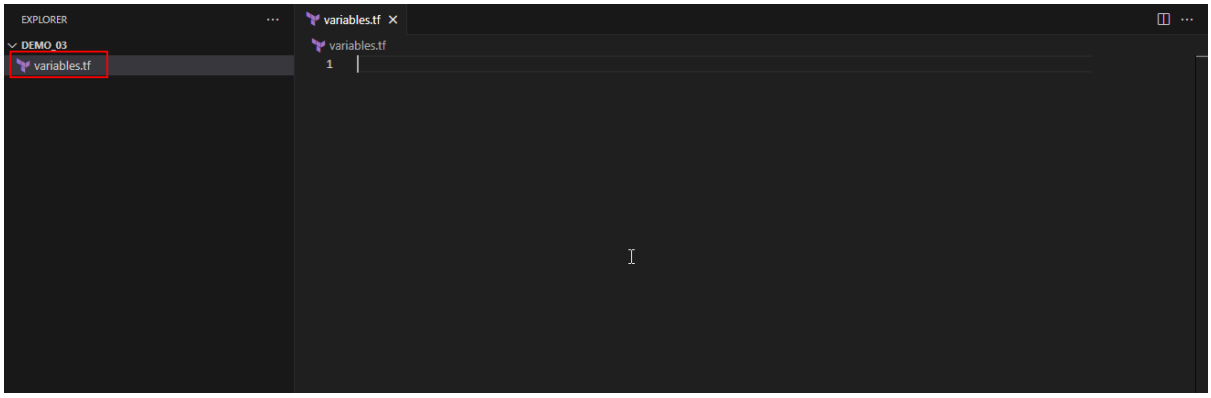
**Prerequisites:** Refer to the **Demo 01** of **Lesson 11** for creating access and secret key

Steps to be followed:

1. Set up and validate variables
2. Secure and manage sensitive data
3. Apply configuration and review outputs

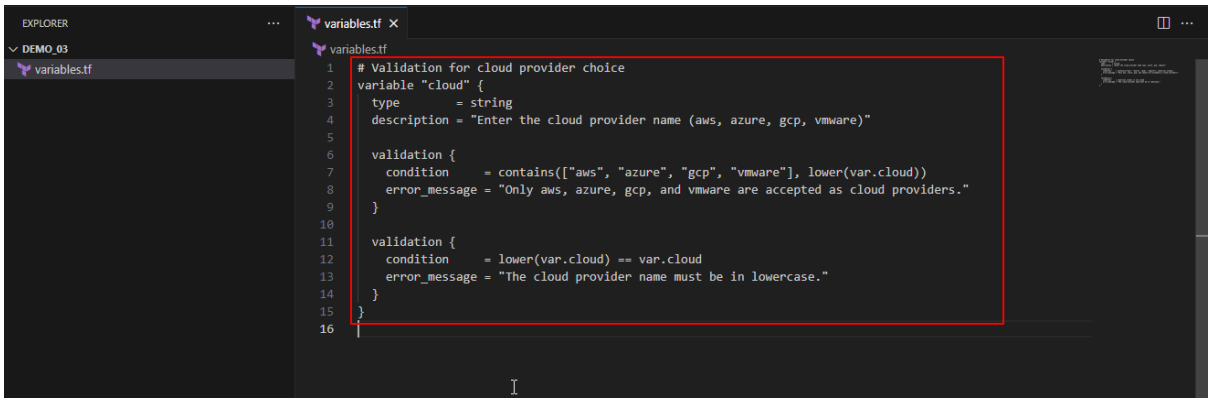
## Step 1: Set up and validate variables

1.1 Open your Terraform configuration environment and create a file named **variables.tf**



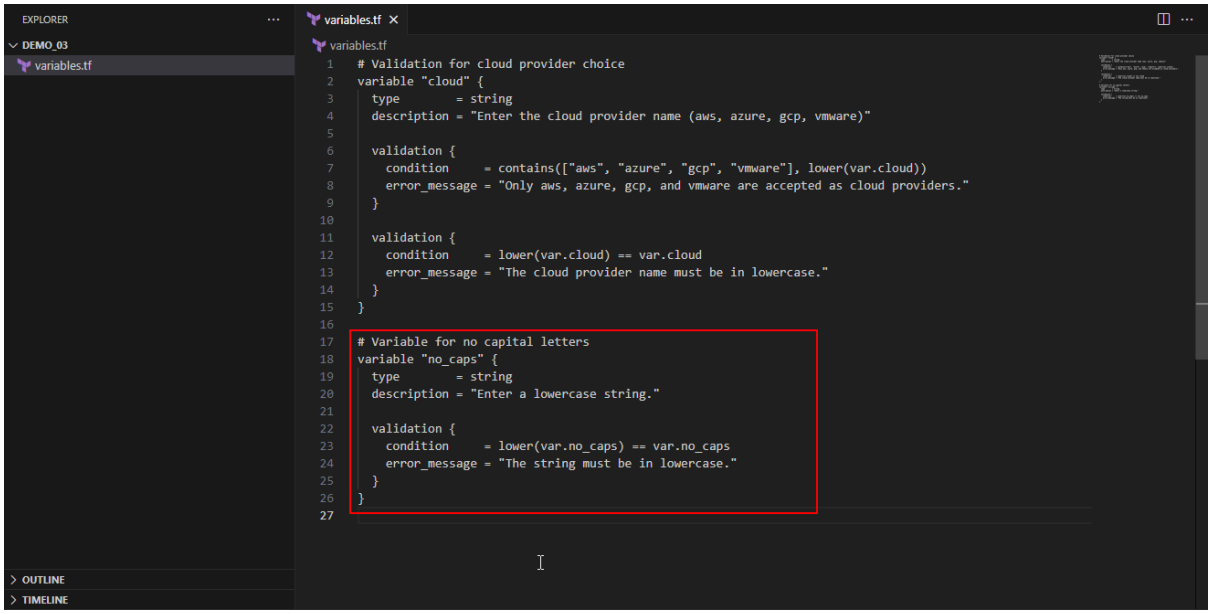
1.2 Open your Terraform configuration environment, create a file named **variables.tf**, and add the variable for cloud provider choice as shown in the screenshot below:

```
# Validation for cloud provider choice
variable "cloud" {
  type        = string
  description = "Enter the cloud provider name (aws, azure, gcp, vmware)"

  validation {
    condition     = contains(["aws", "azure", "gcp", "vmware"], lower(var.cloud))
    error_message = "Only aws, azure, gcp, and vmware are accepted as cloud
providers."
  }

  validation {
    condition     = lower(var.cloud) == var.cloud
    error_message = "The cloud provider name must be in lowercase."
  }
}
```



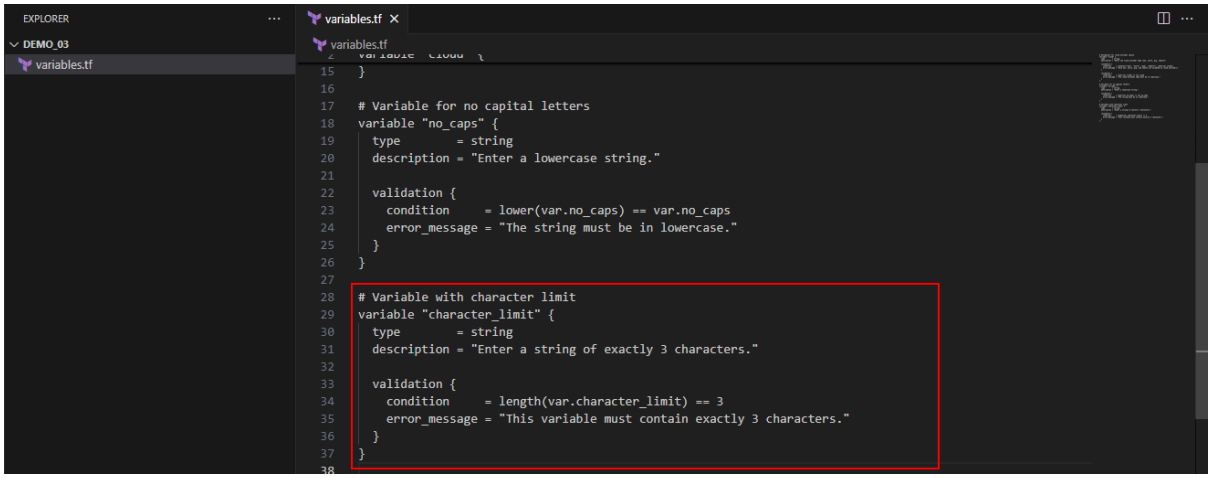
1.3 Add the variable to enforce no capital letters

```
# Variable for no capital letters
variable "no_caps" {
  type        = string
  description = "Enter a lowercase string."

  validation {
    condition     = lower(var.no_caps) == var.no_caps
    error_message = "The string must be in lowercase."
  }
}
```



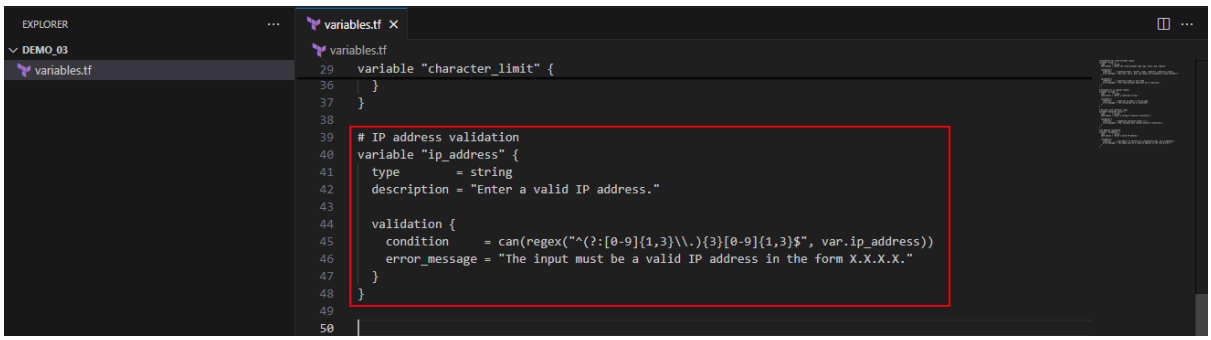
1.4 Add the variable with a character limit as shown in the screenshot below:

```
# Variable with character limit
variable "character_limit" {
  type        = string
  description = "Enter a string of exactly 3 characters."

  validation {
    condition     = length(var.character_limit) == 3
    error_message = "This variable must contain exactly 3 characters."
  }
}
```



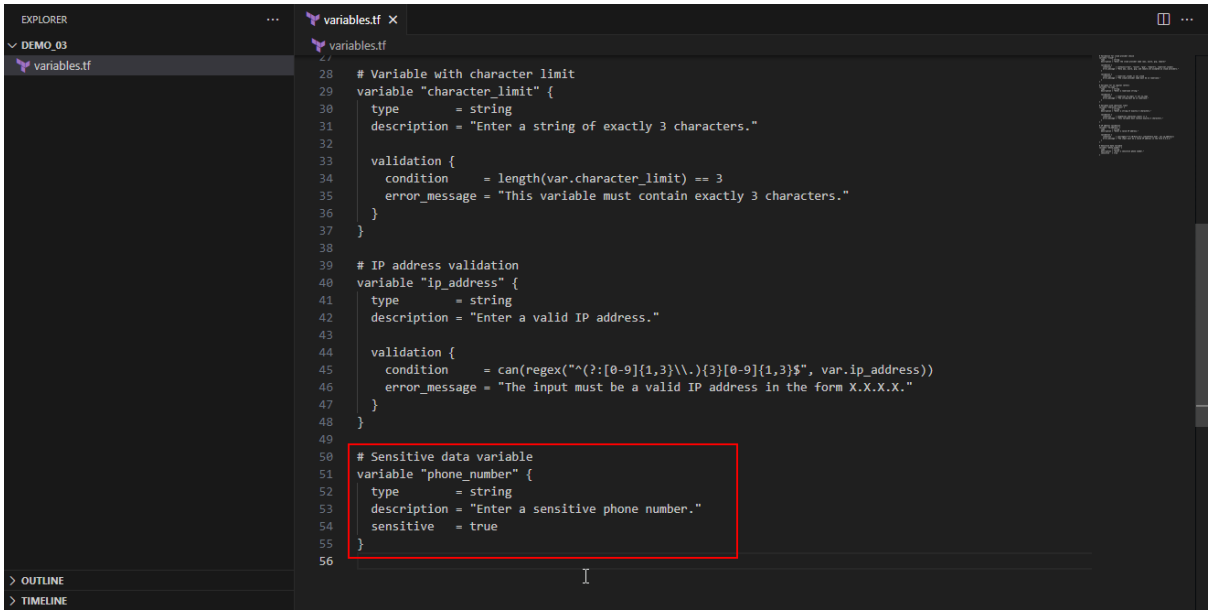

1.5 Add the IP address validation variable as shown in the screenshot below:

```
# IP address validation
variable "ip_address" {
  type        = string
  description = "Enter a valid IP address."

  validation {
    condition     = can(regex("^(?:[0-9]{1,3}\\.){3}[0-9]{1,3}$", var.ip_address))
    error_message = "The input must be a valid IP address in the form X.X.X.X."
  }
}
```



1.6 Add the variable for sensitive data (phone number) as shown in the screenshot below:

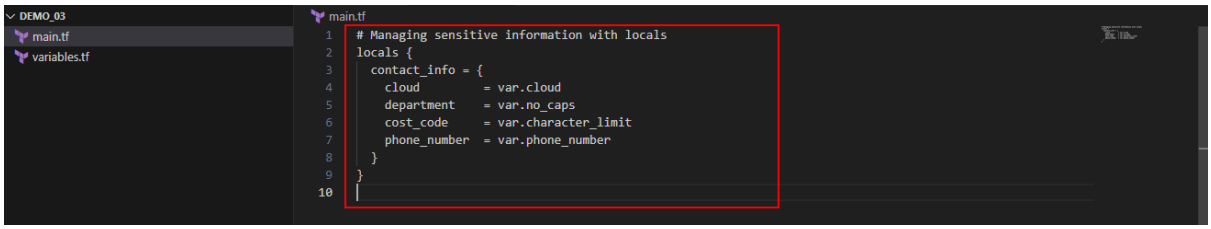
```
# Sensitive data variable
variable "phone_number" {
  type        = string
  description = "Enter a sensitive phone number."
  sensitive   = true
}
```




## Step 2: Secure and manage sensitive data

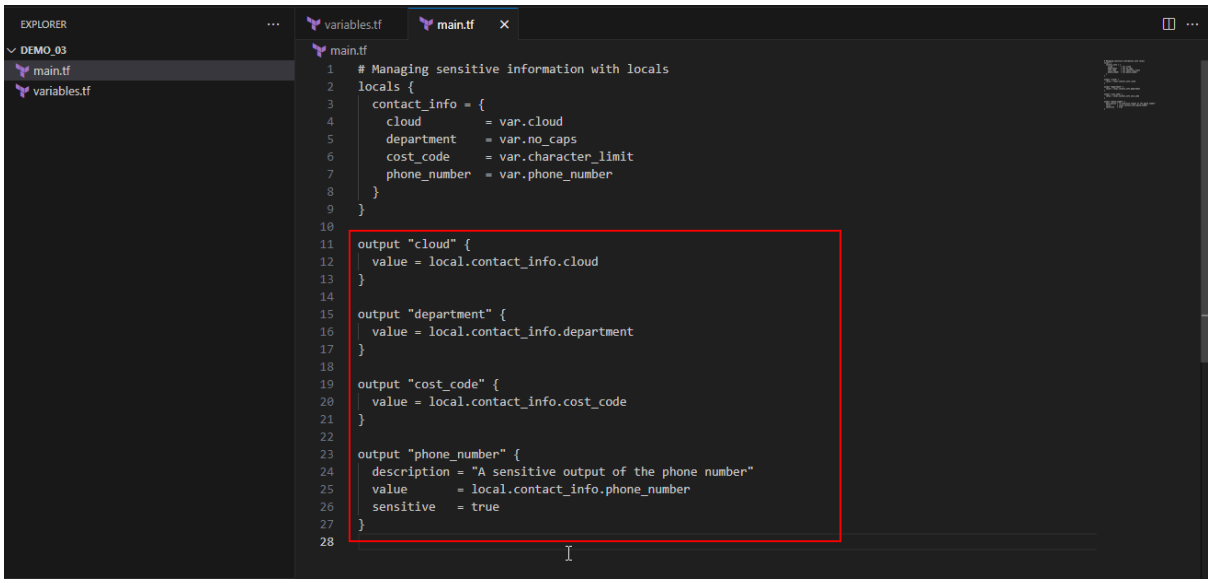
2.1 Create a file named **main.tf**, and add the following block to define local variable for managing sensitive information as shown in the screenshot below:

```
# Managing sensitive information with locals
locals {
  contact_info = {
    cloud        = var.cloud
    department   = var.no_caps
    cost_code    = var.character_limit
    phone_number = var.phone_number
  }
}
```

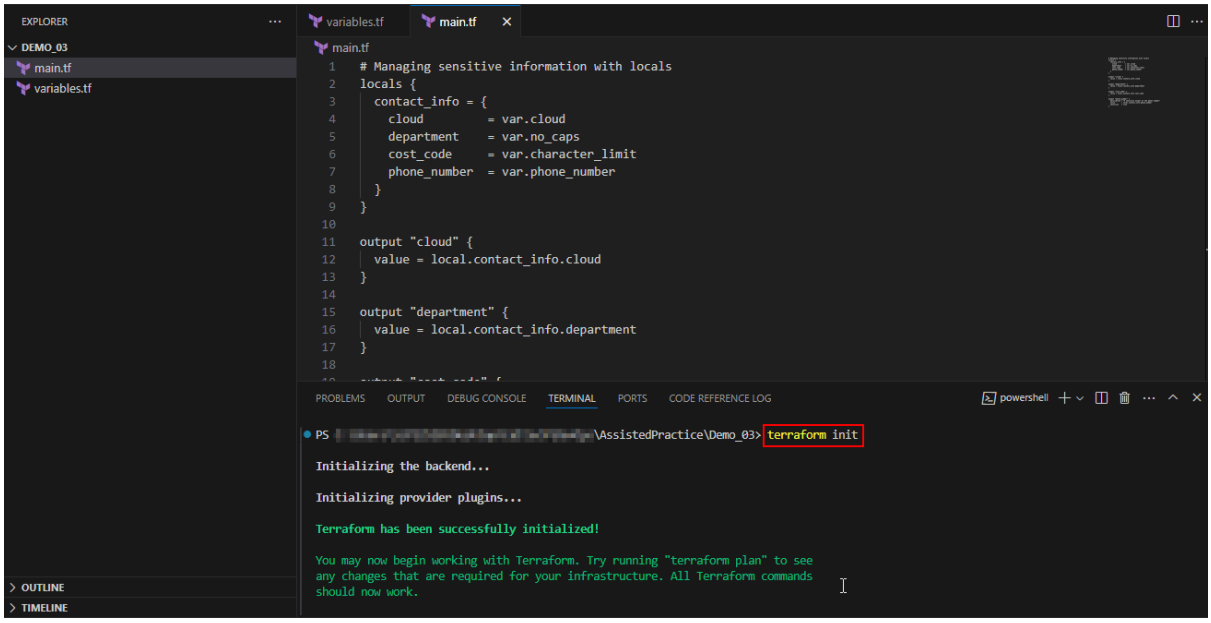


2.2 Add the following block to define output variables for displaying data:

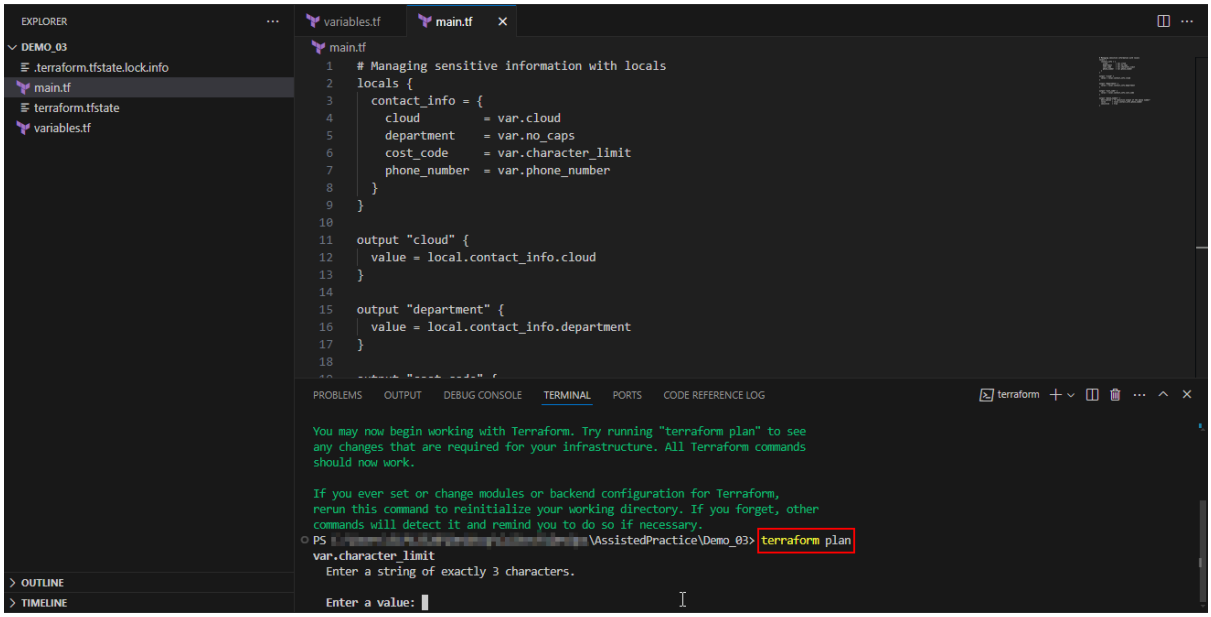
```
output "cloud" {
  value = local.contact_info.cloud
}

output "department" {
  value = local.contact_info.department
}

output "cost_code" {
  value = local.contact_info.cost_code
}

output "phone_number" {
  description = "A sensitive output of the phone number"
  value       = local.contact_info.phone_number
  sensitive   = true
}
```



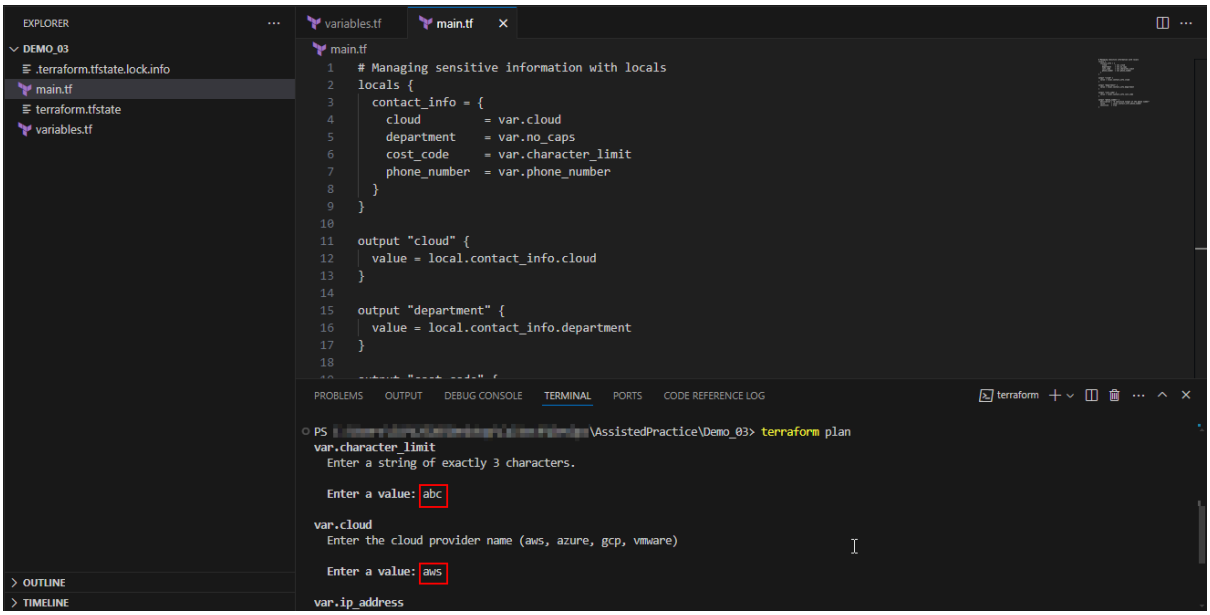
2.3 Initialize the Terraform configuration using the following command:
**terraform init**

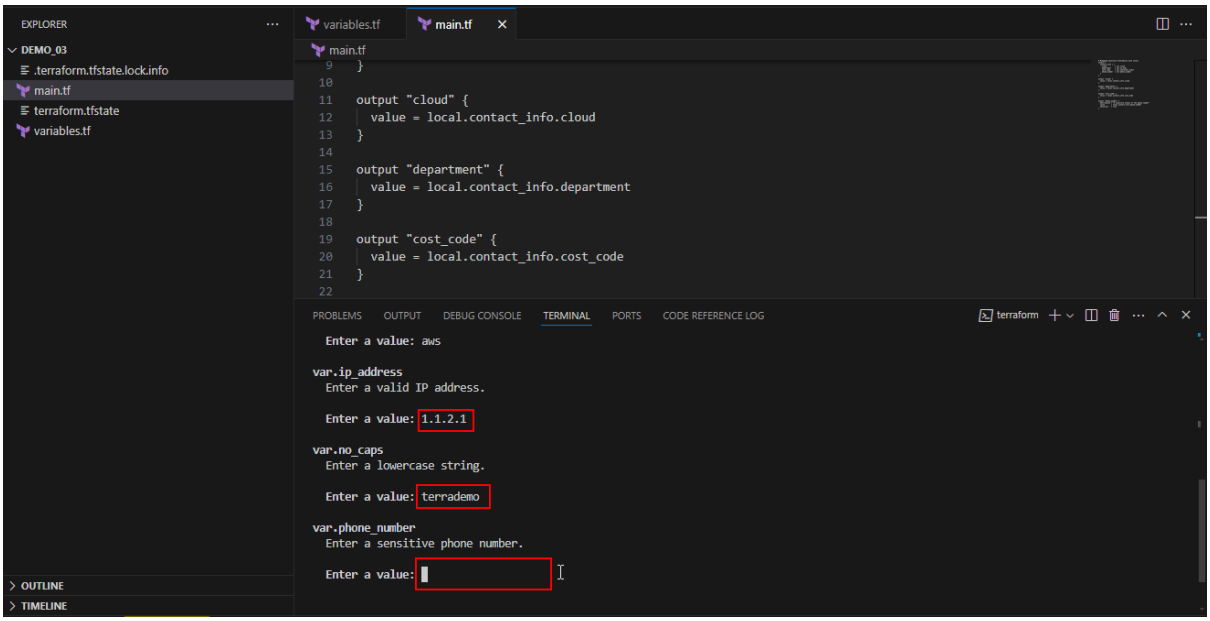



2.4 Use the following command to plan and review the setup for ensuring all validations and sensitive data settings are correct:
**terraform plan**



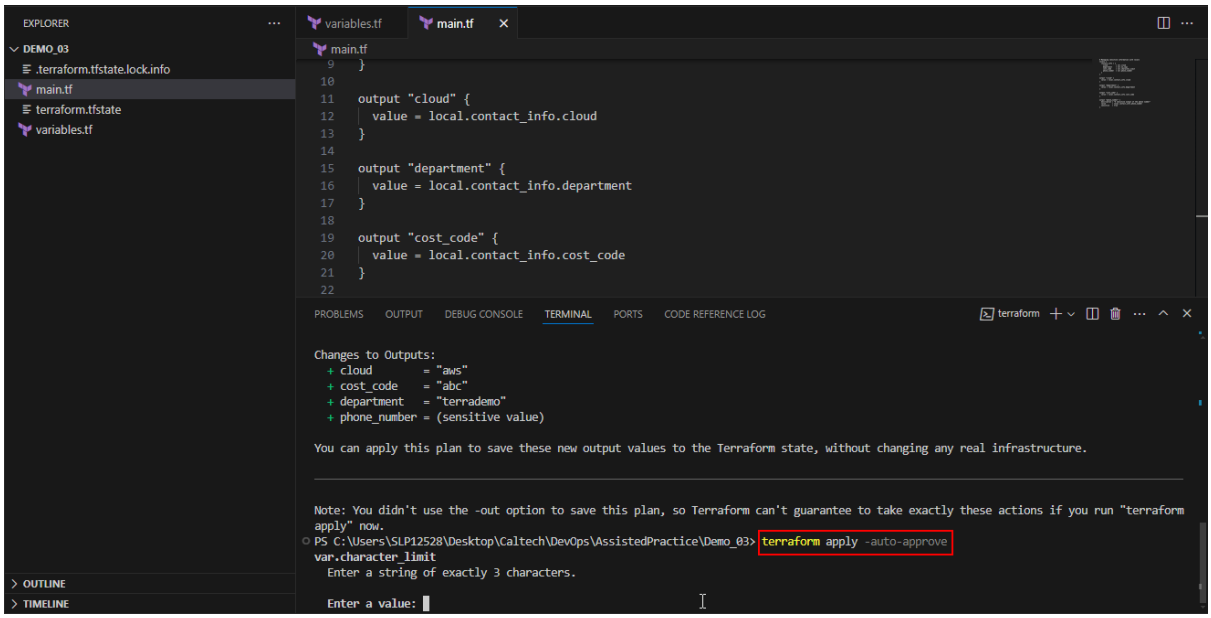
2.5 Add the required input data for the variables as shown in the screenshots below:



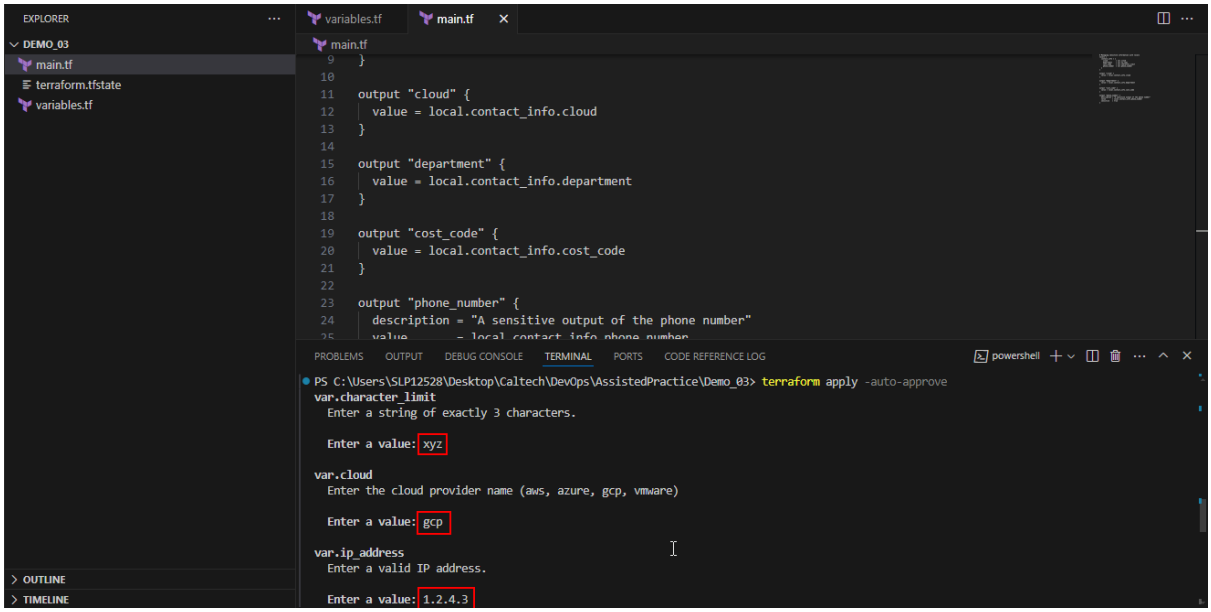



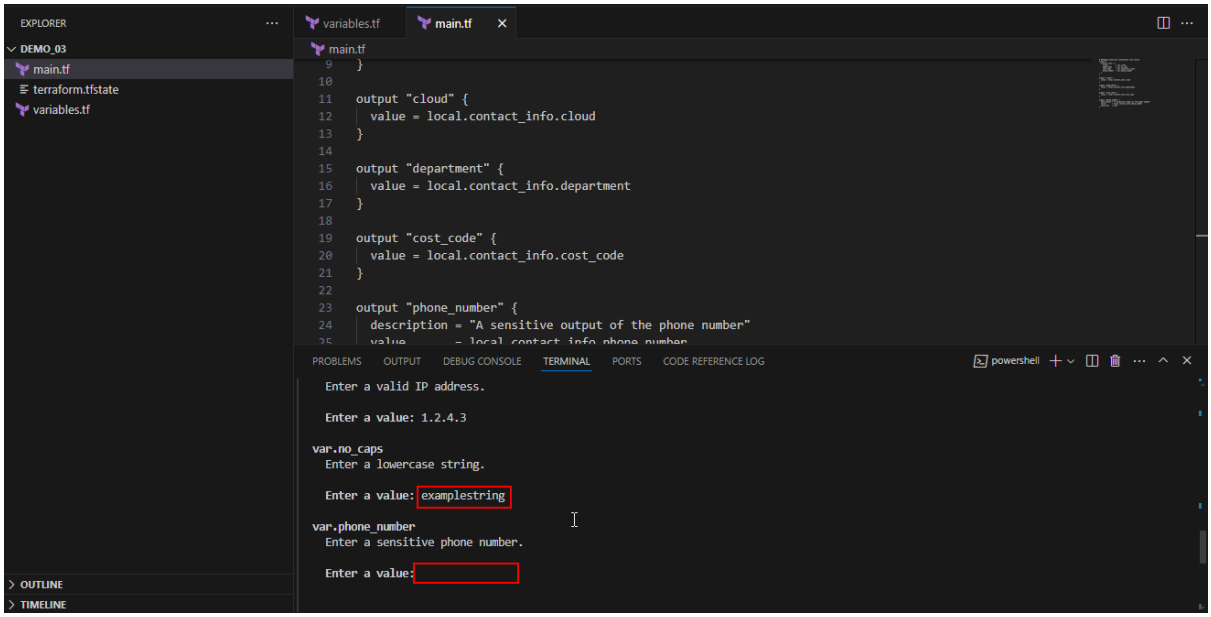

**Note:** Try entering different values for the inputs, such as **1.23.12.3** for the **IP address**, and similarly for the other input fields, and observe any errors that occur.

2.6 Apply the configuration using the following command to deploy the changes as shown in the screenshot below:
**terraform apply -auto-approve**




2.7 Add the required input data for the variables as shown in the screenshots below:

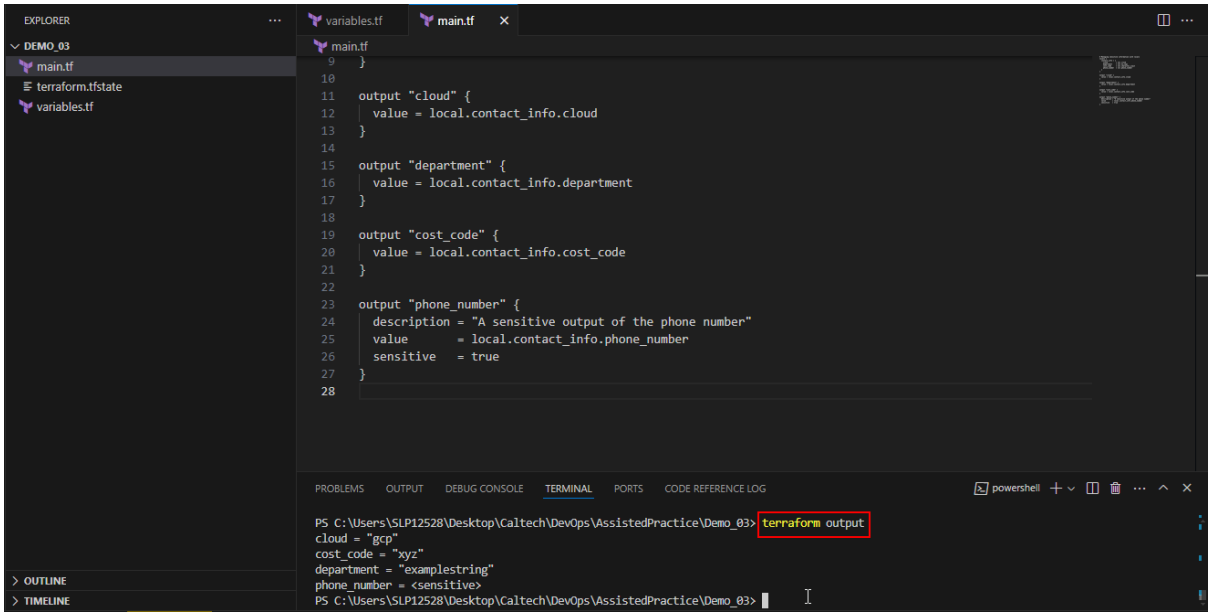





**Note:** Try entering different values for the inputs, such as **1.23.12.3** for the **IP address**, and similarly for the other input fields, and observe any errors that occur.

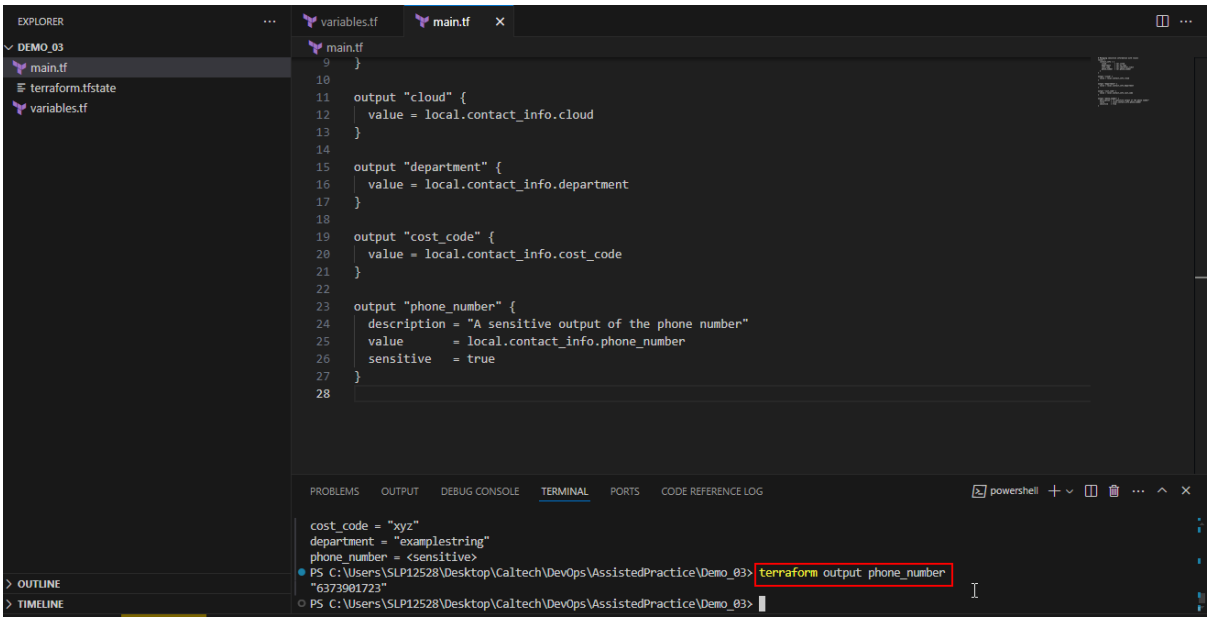
## Step 3: Verify and utilize output values

3.1 Check the output values using the following command as shown in the screenshot below:
**terraform output**

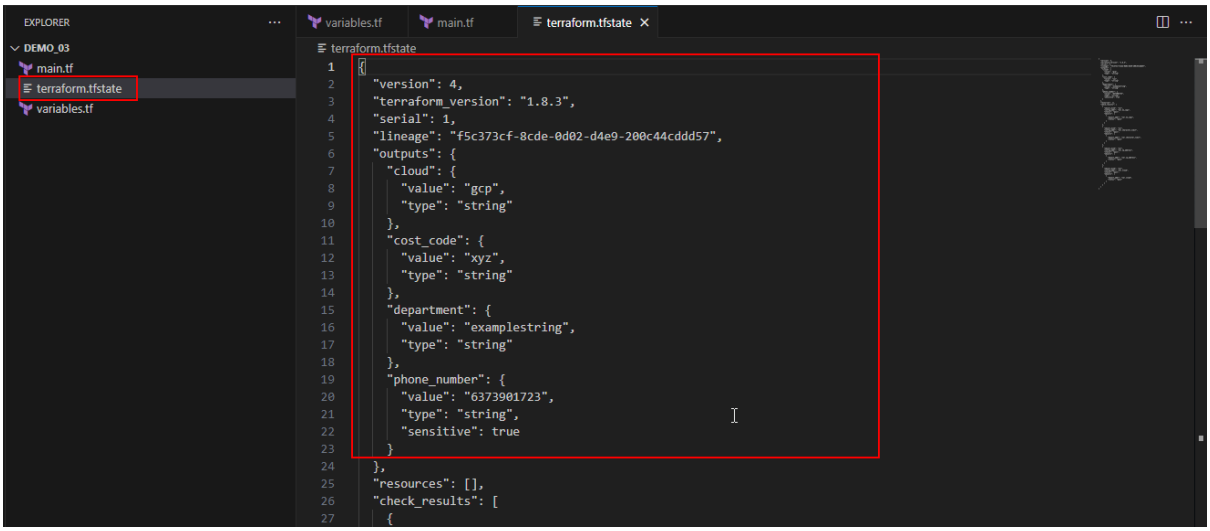



**Note**: The **phone_number** is set as sensitive and will not be displayed.

3.2 Execute the following command to display the value of the sensitive **phone_number** output:
**terraform output phone_number**




3.3 Review the **terraform.tfstate** file to ensure access is securely restricted due to the sensitive information it contains




**Note:** Even though items are marked as sensitive within the Terraform configuration, they are stored within the Terraform state file. It is therefore critical to limit access to the Terraform state file.

By following these steps, you have successfully validated variables and secured sensitive data within your Terraform configuration for improved security and reliability.