

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

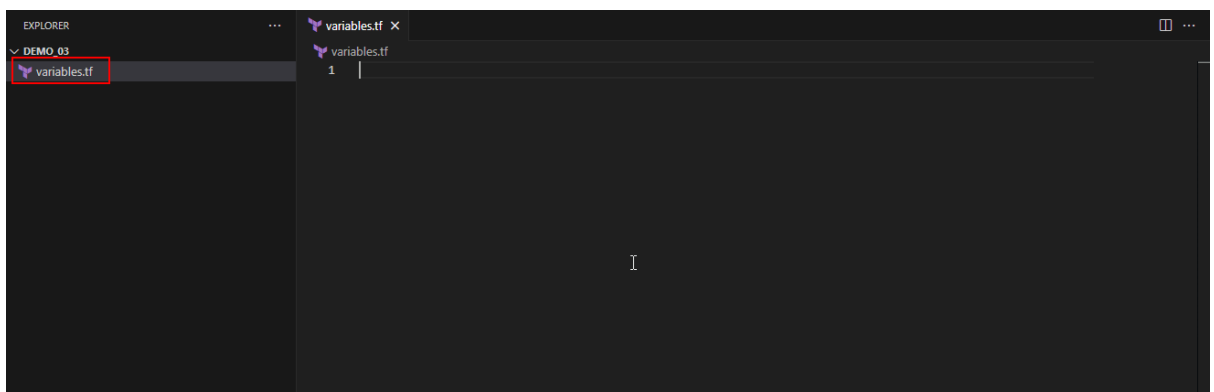
Prerequisites: Refer to the **Demo 01** of **Lesson 11** for creating access and secret key

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

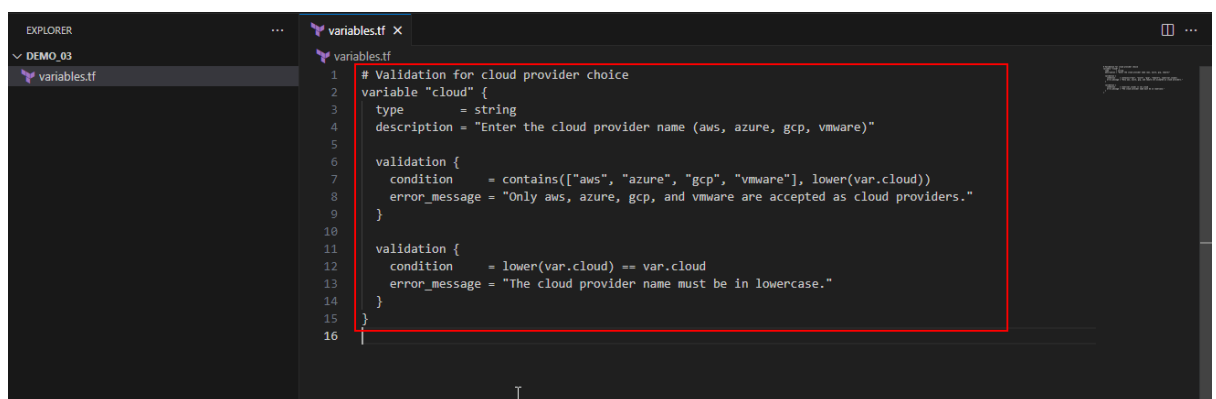
1.1 Open your Terraform configuration environment and create a file named **variables.tf**



- 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."
```

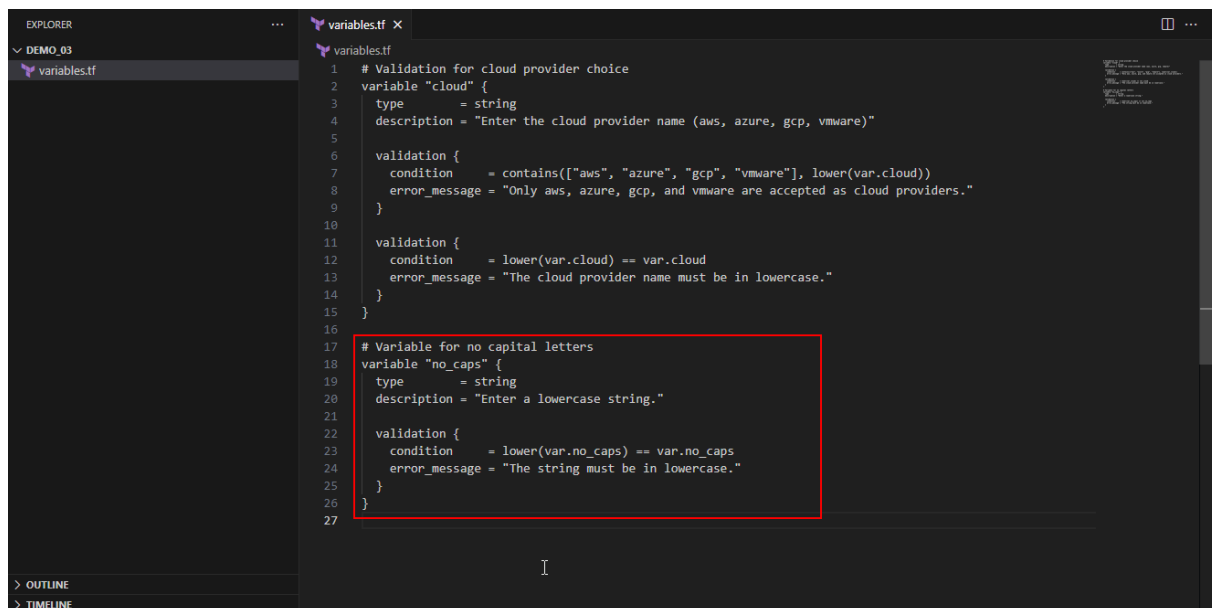
```
  validation {
```

```
    condition = lower(var.no_caps) == var.no_caps
```

```
    error_message = "The string must be in lowercase."
```

```
  }
```

```
}
```

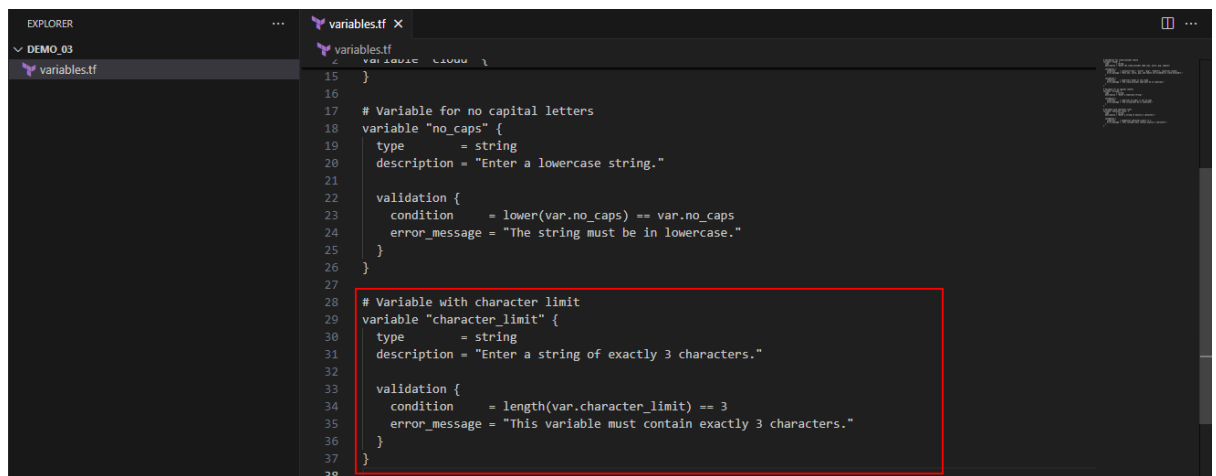


```
1 # Validation for cloud provider choice
2 variable "cloud" {
3   type    = string
4   description = "Enter the cloud provider name (aws, azure, gcp, vmware)"
5
6   validation {
7     condition = contains(["aws", "azure", "gcp", "vmware"], lower(var.cloud))
8     error_message = "Only aws, azure, gcp, and vmware are accepted as cloud providers."
9   }
10
11   validation {
12     condition = lower(var.cloud) == var.cloud
13     error_message = "The cloud provider name must be in lowercase."
14   }
15 }
16
17 # Variable for no capital letters
18 variable "no_caps" {
19   type    = string
20   description = "Enter a lowercase string."
21
22   validation {
23     condition = lower(var.no_caps) == var.no_caps
24     error_message = "The string must be in lowercase."
25   }
26 }
27
```

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."

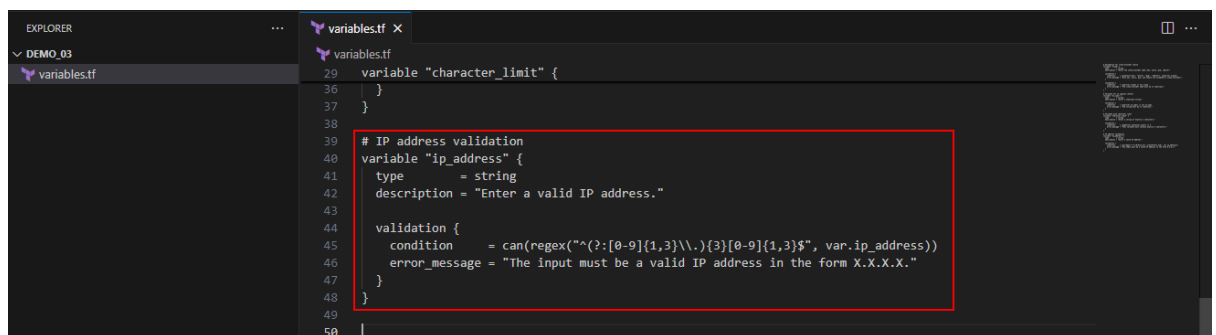
  validation {
    condition  = length(var.character_limit) == 3
    error_message = "This variable must contain exactly 3 characters."
  }
}
```



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

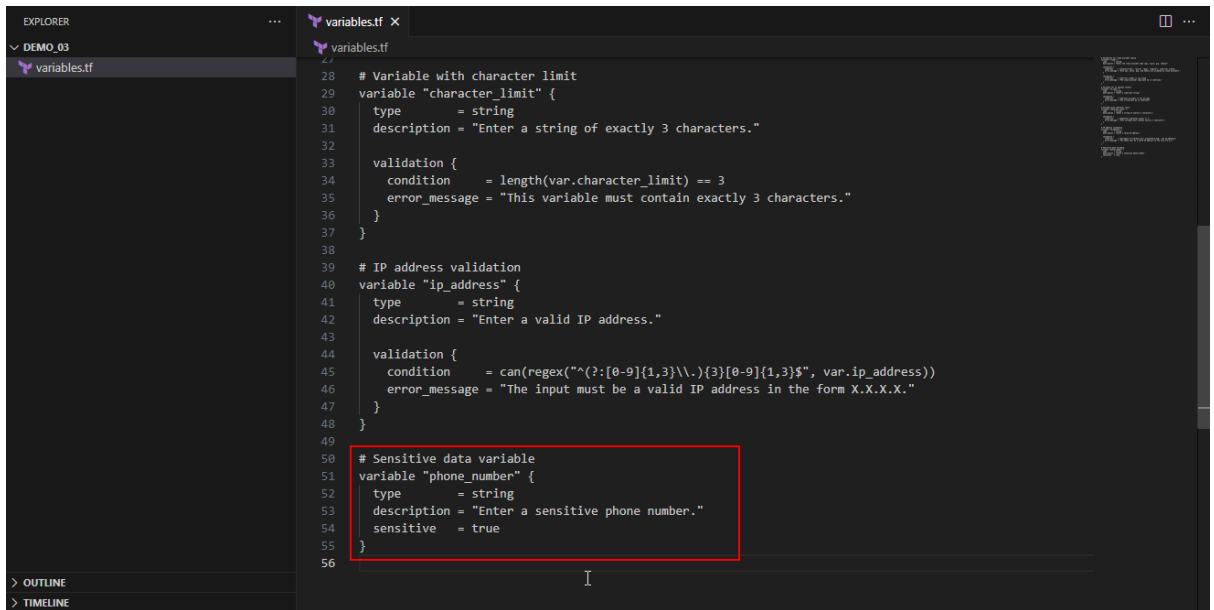
```
# IP address validation
variable "ip_address" {
  type    = string
  description = "Enter a valid IP address."

  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:

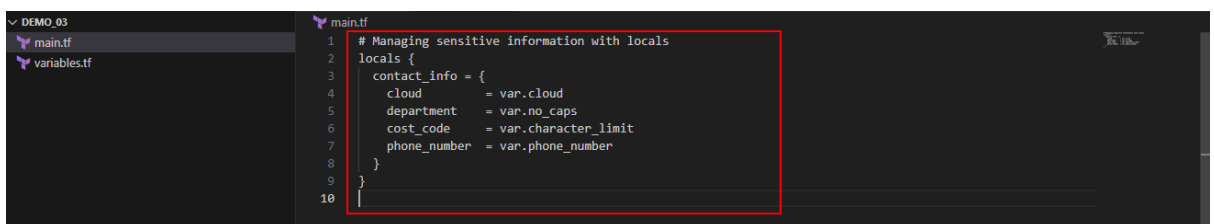
```
# Sensitive data variable
variable "phone_number" {
  type      = string
  description = "Enter a sensitive phone number."
  sensitive = true
}
```



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
  }
}
```



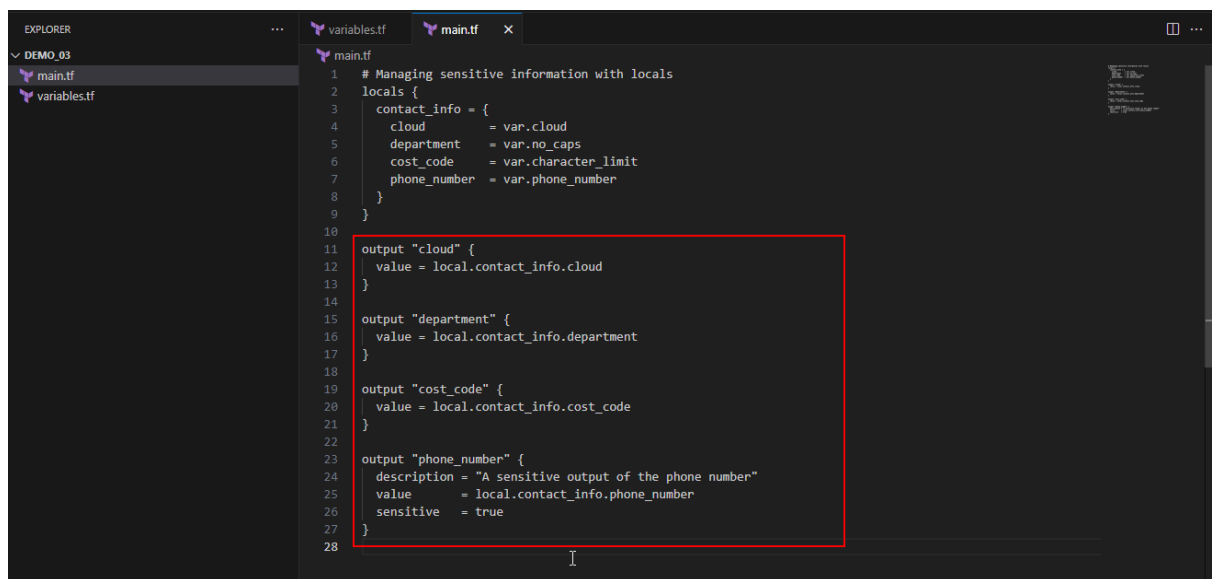
2.2 Add the following block to define output variables for displaying data:

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

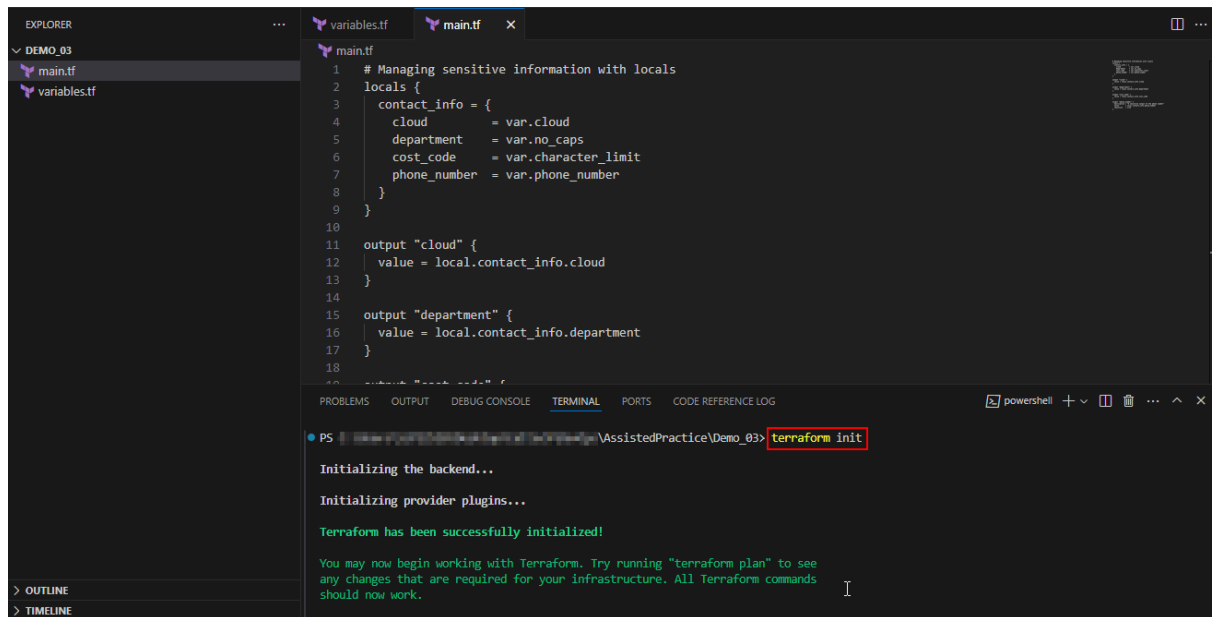
```
output "department" {  
  value = local.contact_info.department  
}
```

```
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**



The screenshot shows the Visual Studio Code interface with a Terraform configuration file open. The Explorer pane on the left shows a project named 'DEMO_03' containing 'main.tf' and 'variables.tf'. The main editor displays the content of 'main.tf', which includes a 'locals' block for managing sensitive information and two 'output' blocks for 'cloud' and 'department'. The bottom panel shows the 'TERMINAL' tab with the command 'terraform init' entered and executed. The terminal output indicates that the backend is initialized, provider plugins are installed, and Terraform has been successfully initialized. A message at the bottom of the terminal suggests running 'terraform plan' to see any changes required for the infrastructure.

```
1 # Managing sensitive information with locals
2 locals {
3   contact_info = {
4     cloud       = var.cloud
5     department  = var.no_caps
6     cost_code   = var.character_limit
7     phone_number = var.phone_number
8   }
9 }
10
11 output "cloud" {
12   value = local.contact_info.cloud
13 }
14
15 output "department" {
16   value = local.contact_info.department
17 }
18
```

```
PS C:\AssistedPractice\Demo_03> terraform init

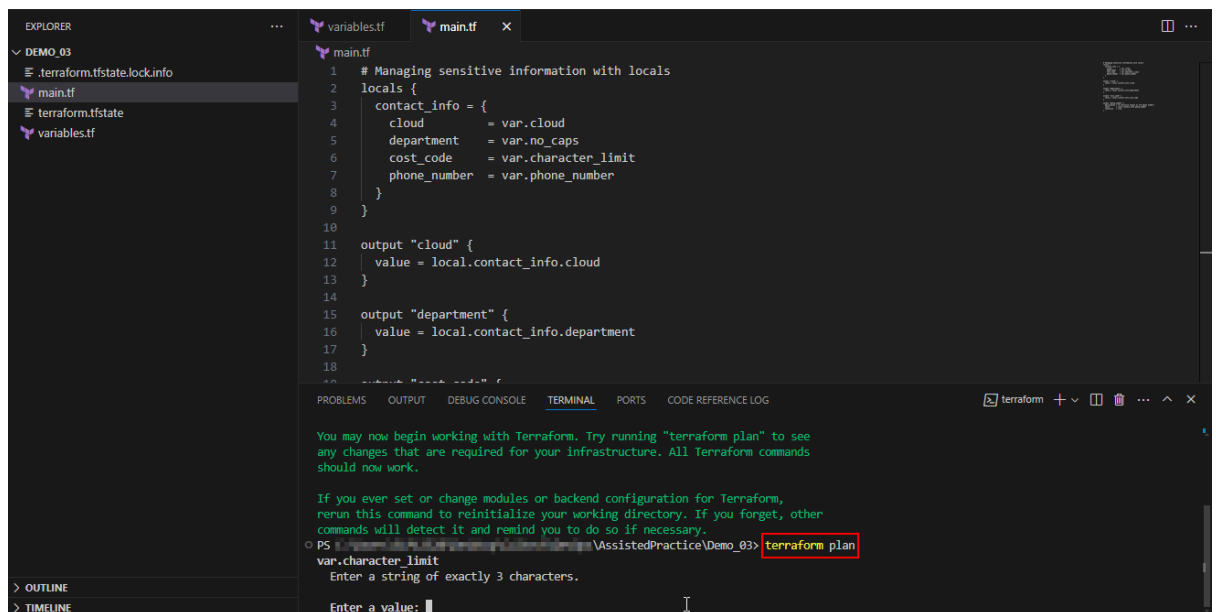
Initializing the backend...

Initializing provider plugins...

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.
```

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



The screenshot shows the Visual Studio Code interface with the same Terraform configuration file open. The Explorer pane on the left now includes '.terraform.tfstate.lock.info' and 'terraform.tfstate' alongside 'main.tf' and 'variables.tf'. The main editor still displays the content of 'main.tf'. The bottom panel shows the 'TERMINAL' tab with the command 'terraform plan' entered and executed. The terminal output indicates that the user can now begin working with Terraform and suggests running 'terraform plan' to see any changes required for the infrastructure. A message at the bottom of the terminal suggests running 'terraform plan' to see any changes required for the infrastructure.

```
1 # Managing sensitive information with locals
2 locals {
3   contact_info = {
4     cloud       = var.cloud
5     department  = var.no_caps
6     cost_code   = var.character_limit
7     phone_number = var.phone_number
8   }
9 }
10
11 output "cloud" {
12   value = local.contact_info.cloud
13 }
14
15 output "department" {
16   value = local.contact_info.department
17 }
18
```

```
PS C:\AssistedPractice\Demo_03> terraform plan

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

var.character_limit
  Enter a string of exactly 3 characters.
  Enter a value: 
```

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

```
1 # Managing sensitive information with locals
2 locals {
3   contact_info = {
4     cloud       = var.cloud
5     department  = var.no_caps
6     cost_code   = var.character_limit
7     phone_number = var.phone_number
8   }
9 }
10
11 output "cloud" {
12   value = local.contact_info.cloud
13 }
14
15 output "department" {
16   value = local.contact_info.department
17 }
18
19 output "cost_code" {
20   value = local.contact_info.cost_code
21 }
```

PS C:\AssistedPractice\Demo_03> terraform plan

var.character_limit
Enter a string of exactly 3 characters.
Enter a value:

var.cloud
Enter the cloud provider name (aws, azure, gcp, vmware)
Enter a value:

var.ip_address

```
9 }
10
11 output "cloud" {
12   value = local.contact_info.cloud
13 }
14
15 output "department" {
16   value = local.contact_info.department
17 }
18
19 output "cost_code" {
20   value = local.contact_info.cost_code
21 }
22
```

Enter a value: aws

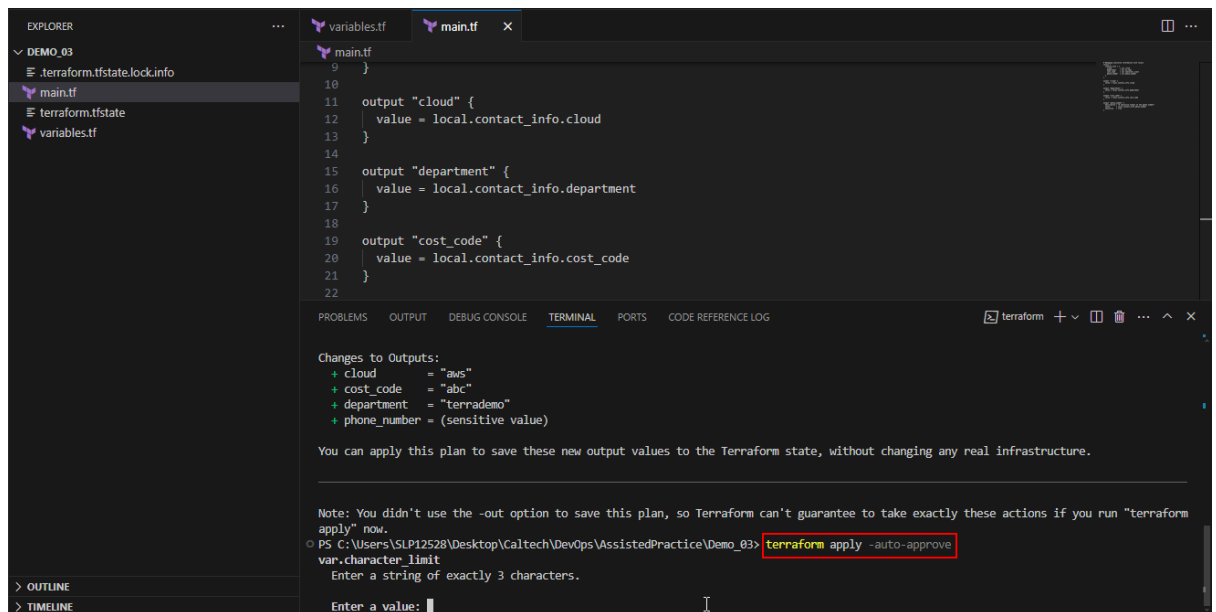
var.ip_address
Enter a valid IP address.
Enter a value:

var.no_caps
Enter a lowercase string.
Enter a value:

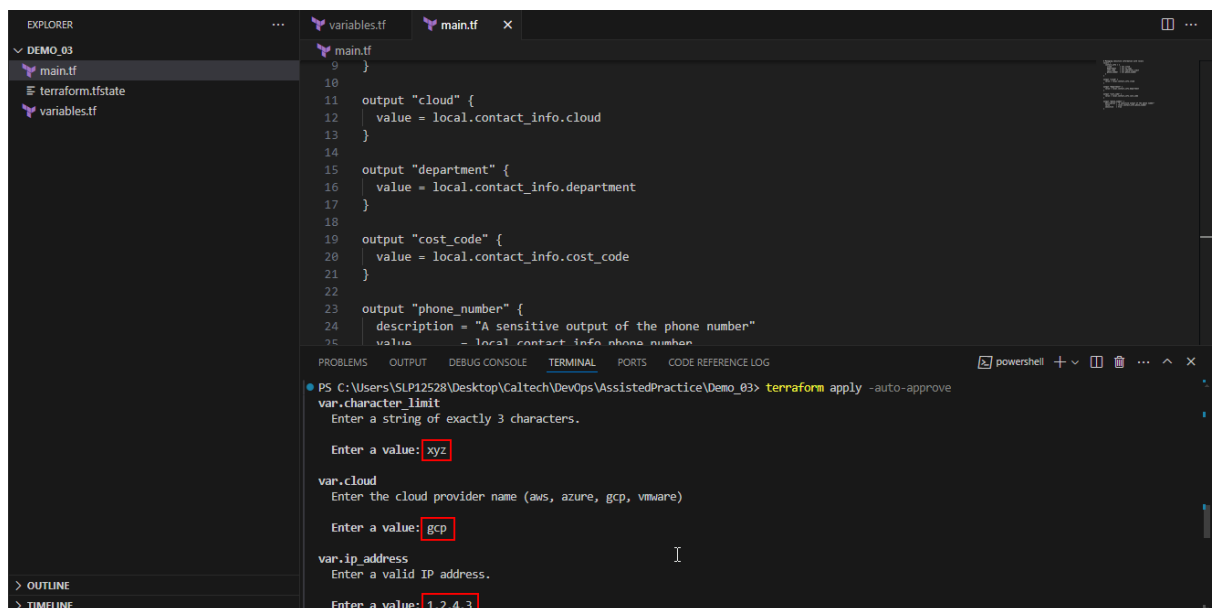
var.phone_number
Enter a sensitive phone number.
Enter a value:

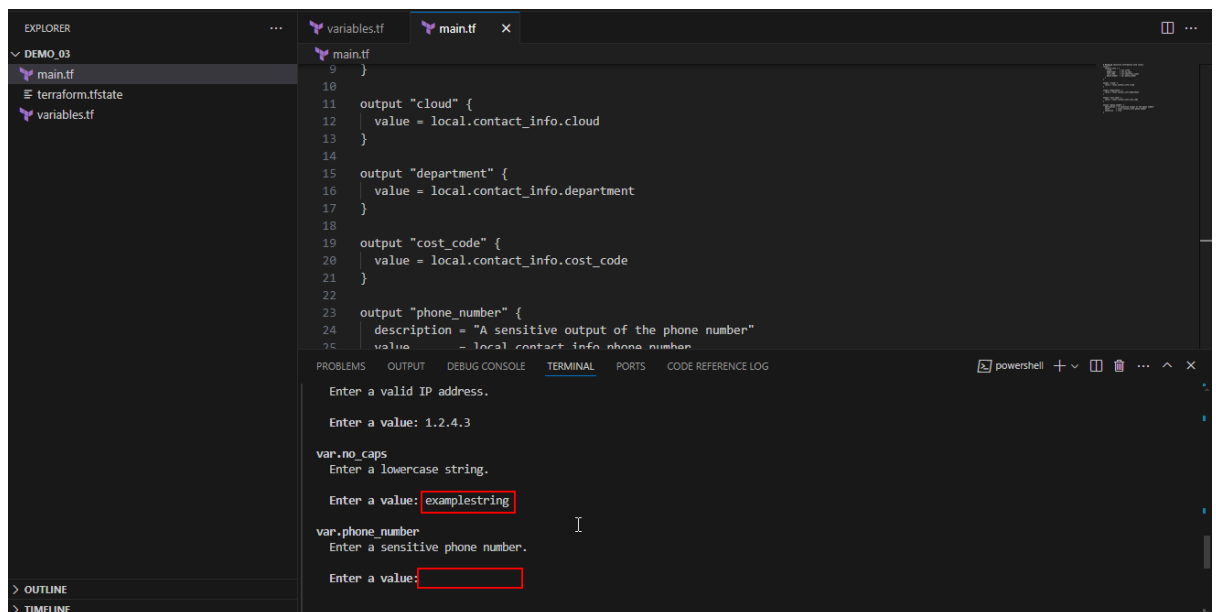
Note: Try entering different values for the inputs, such as **1.23.12.3** for the **IP address**, and similarly for the other input fields, and observe any errors that occur.

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:

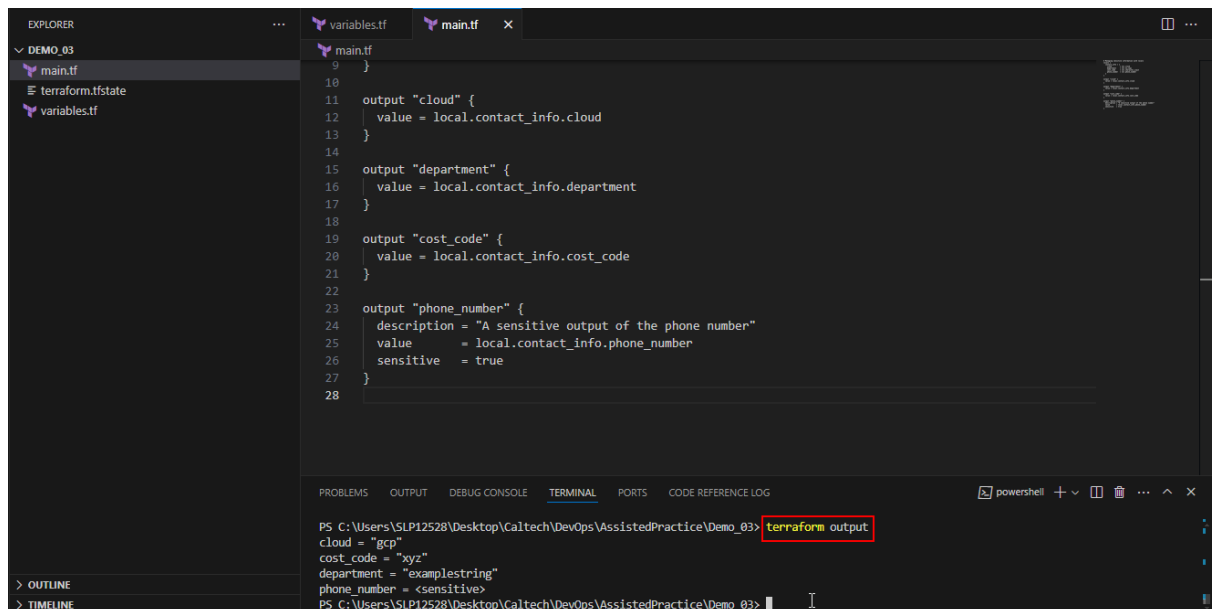




Note: Try entering different values for the inputs, such as **1.23.12.3** for the **IP address**, and similarly for the other input fields, and observe any errors that occur.

Step 3: Verify and utilize output values

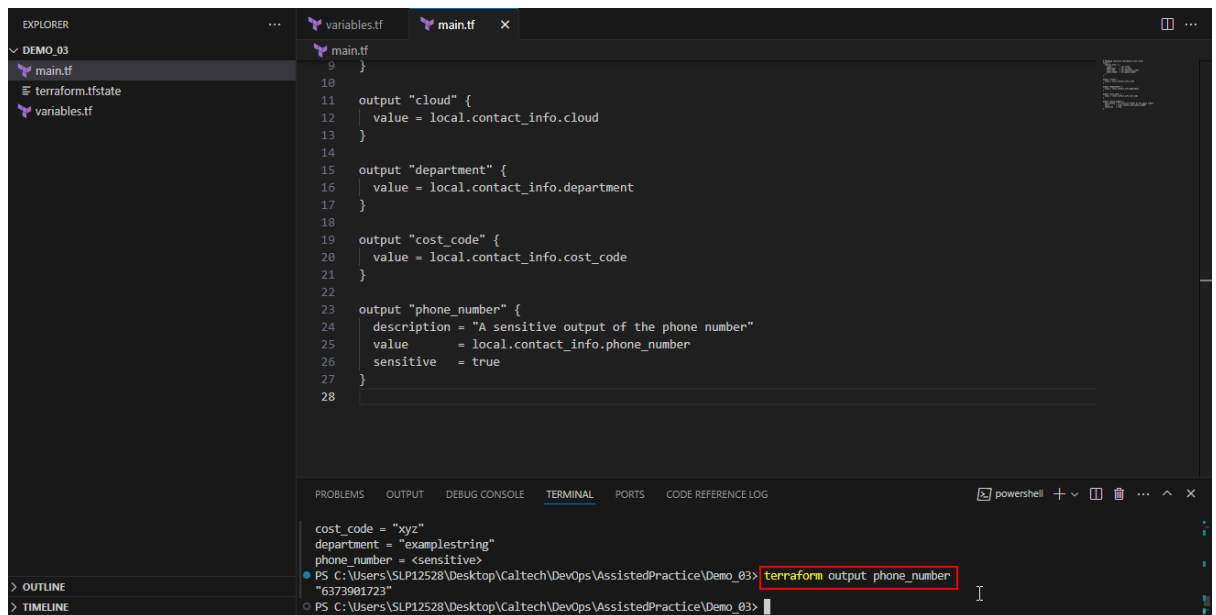
- 3.1 Check the output values using the following command as shown in the screenshot below:
terraform output



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

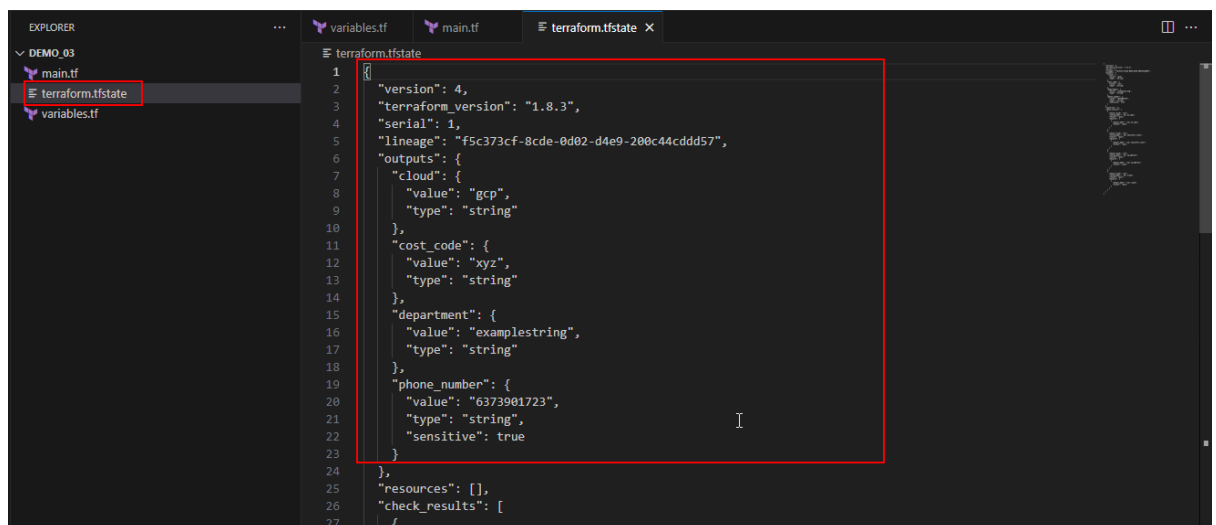


The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a project named 'DEMO_03' with files 'main.tf', 'terraform.tfstate', and 'variables.tf'. The main editor displays the 'main.tf' file, which contains Terraform output definitions for 'cloud', 'department', 'cost_code', and 'phone_number'. The 'phone_number' output is marked as sensitive. The terminal at the bottom shows the command 'terraform output phone_number' being executed, and the output is '6373901723', which is highlighted with a red box.

```
main.tf
9 }
10
11 output "cloud" {
12   value = local.contact_info.cloud
13 }
14
15 output "department" {
16   value = local.contact_info.department
17 }
18
19 output "cost_code" {
20   value = local.contact_info.cost_code
21 }
22
23 output "phone_number" {
24   description = "A sensitive output of the phone number"
25   value       = local.contact_info.phone_number
26   sensitive   = true
27 }
28
```

```
cost_code = "xyz"
department = "examplestring"
phone_number = <sensitive>
PS C:\Users\SLP12528\Desktop\Caltech\DevOps\AssistedPractice\Demo_03> terraform output phone_number
6373901723
```

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



The screenshot shows the Visual Studio Code interface with the 'terraform.tfstate' file open in the main editor. The Explorer pane on the left shows the file 'terraform.tfstate' selected. The main editor displays the JSON content of the state file, which includes the 'phone_number' output value '6373901723' and its sensitive status. A red box highlights the 'phone_number' entry in the 'outputs' section.

```
terraform.tfstate
1 {
2   "version": 4,
3   "terraform_version": "1.8.3",
4   "serial": 1,
5   "lineage": "f5c373cf-8cde-0d02-d4e9-200c44cddd57",
6   "outputs": {
7     "cloud": {
8       "value": "gcp",
9       "type": "string"
10    },
11    "cost_code": {
12      "value": "xyz",
13      "type": "string"
14    },
15    "department": {
16      "value": "examplestring",
17      "type": "string"
18    },
19    "phone_number": {
20      "value": "6373901723",
21      "type": "string",
22      "sensitive": true
23    }
24  },
25  "resources": [],
26  "check_results": [
27    {
```

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.