Terraform variables

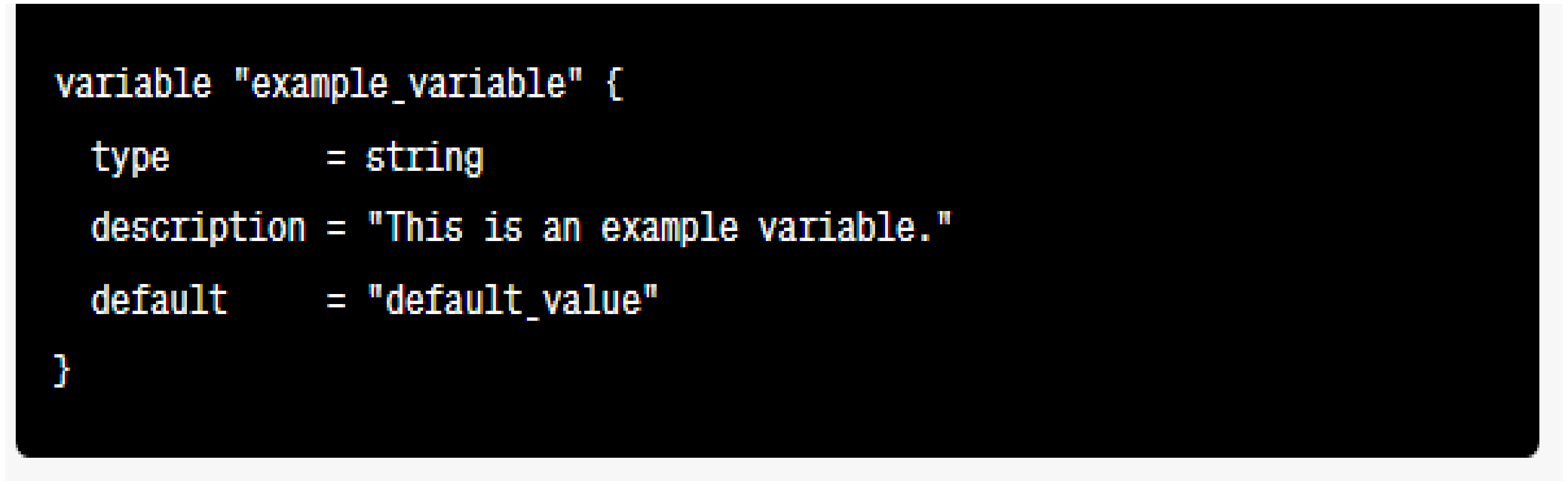
OW

Purpose of Terraform Variables

- Terraform variables are used to parameterize and customize infrastructure configurations. They allow you to define and pass values into your Terraform code, making it more flexible and reusable.
- **Configuration Flexibility**: Variables allow you to easily modify the behavior of your infrastructure deployments without changing the underlying code. Instead of hardcoding values directly into your configuration, you can use variables to make your configurations dynamic and adaptable.
- **Reusability**: Variables enable you to create reusable Terraform modules and configurations. By parameterizing your code with variables, you can create generalized modules that can be used across multiple projects or teams

Variable syntax

- In Terraform, variable declarations follow a specific syntax and structure. Here's an example that demonstrates how to declare a variable with a type, description, and default value:



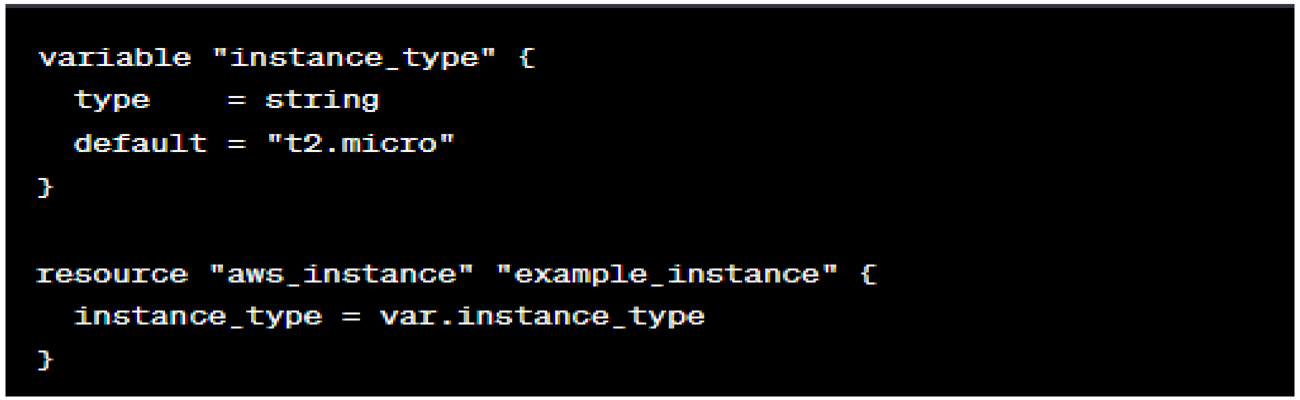
```
variable "example_variable" {
  type        = string
  description = "This is an example variable."
  default     = "default_value"
}
```

explanation

- **variable is the keyword** used to declare a variable in Terraform.
- "**example_variable" is the name** of the variable. You can choose any name that is descriptive and meaningful for your use case.
- **type = string** specifies the type of the variable. In this case, it is a string type. Terraform supports various types, such as string, number, bool, list, map, and object.
- **description =** "This is an example variable." provides a description of the variable. This is **optional .**
- **default = "default_value**" sets a default value for the variable. If no value is explicitly provided when using the Terraform configuration, this default value will be used. The default value should be of the same type as specified in the type attribute.

Interpolation syntax

- Interpolation syntax in Terraform allows you to reference and use the values of variables, resources, and other expressions within your Terraform configurations.
- It enables dynamic configuration by substituting values at runtime.

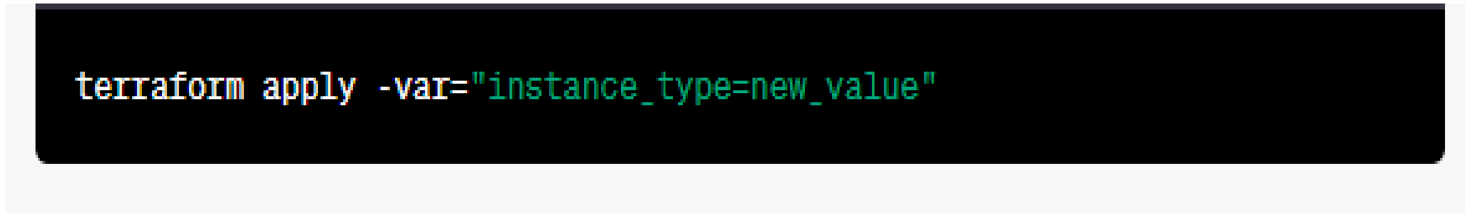


```
variable "instance_type" {
  type    = string
  default = "t2.micro"
}

resource "aws_instance" "example_instance" {
  instance_type = var.instance_type
}
```

Change default value of variable at runtime

- **Command-line Flags:** You can provide variable values using command-line flags when running Terraform commands. For example, to change the default value of a variable named instance_type, you can use the -var flag followed by the variable name and the new value:

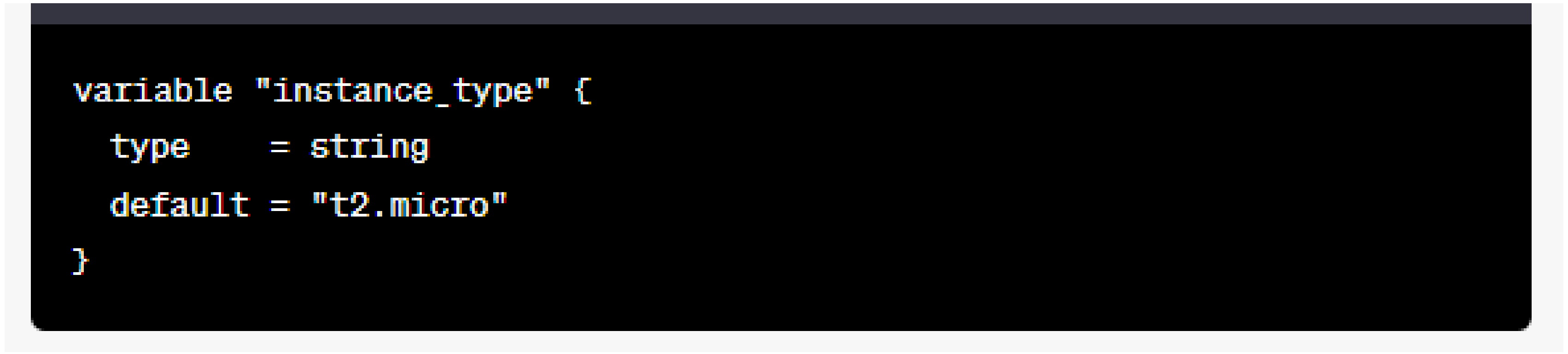


```
terraform apply -var="instance_type=new_value"
```

- This overrides the default value of instance_type with the value new_value during the execution of the Terraform command.

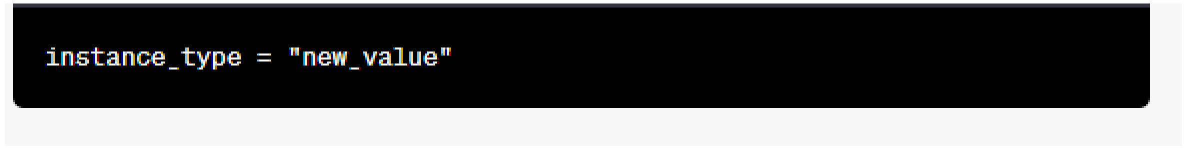
..

- **Variable Files:** Another approach is to use variable files to specify new values. Variable files are separate files containing variable values in key-value format.



```
variable "instance_type" {
  type    = string
  default = "t2.micro"
}
```

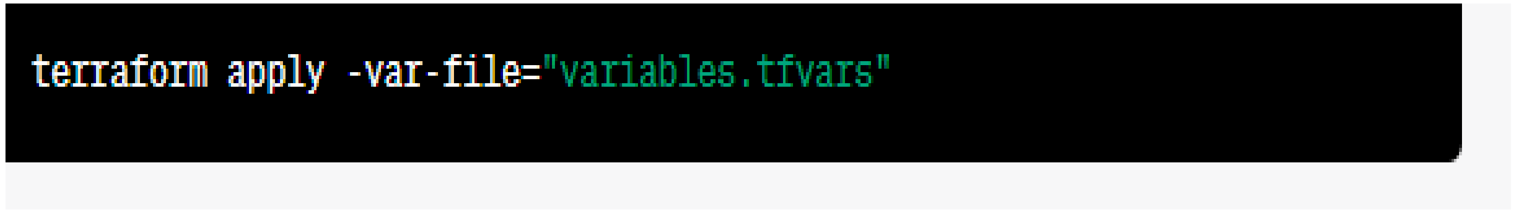
- Example variable file (variables.tfvars):



```
instance_type = "new_value"
```

..

- Then, when running Terraform commands, you can use the -var-file flag to specify the path to the variable file:

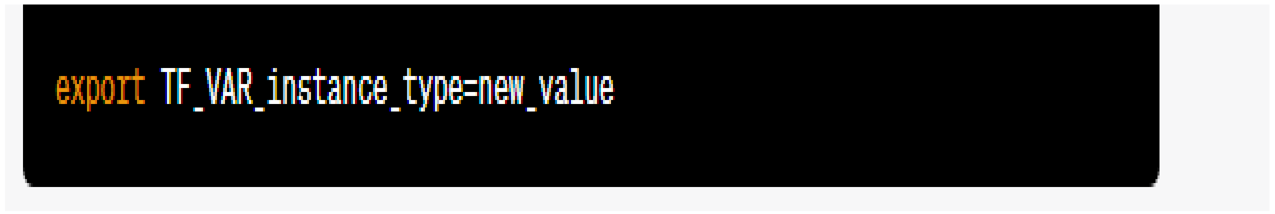


```
terraform apply -var-file="variables.tfvars"
```

- This will override the default value of instance_type with the value specified in the variable file

..

- **Environment Variables:** Terraform also supports setting variables through environment variables. You can set an environment variable with the name TF_VAR_variable_name and the desired value. For example, to change the default value of instance_type, you can set the environment variable TF_VAR_instance_type to the new value:



```
export TF_VAR_instance_type=new_value
```

Variable Precedence

- **Command-line Flags**: Values provided via command-line flags take the **highest precedence**. You can pass variable values directly through the command-line using the -var flag.
- **Environment Variables**: If no value is provided via command-line flags, **Terraform checks for environment variables**. Terraform looks for environment variables prefixed with **TF_VAR_ followed by the variable name**. For example, an environment variable named **TF_VAR_instance_type** would set the value for the instance_type variable.

..

- **Variable Files**: If no value is provided through command-line flags or environment variables, Terraform looks for variable files. Variable files are specified using the **-var-file** flag followed by the path to the file containing the variable values.
- **Default Values:** If no value is provided through the above methods, Terraform uses the default value specified in the variable declaration within the configuration. The default value is defined using the default attribute within the variable block.

Labs

- Question??