What is variable.tf and terraform.tfvars?

Apr 11, 2021 · 8 min read · Share on:

Terraform variable.tf is a file where you can define variables for your Terraform configuration. This file can contain the variable definitions as well as the optional default value for the variable. Here is an example of `variable.tf` which has -

1. Two variables with no default value - `instance_type`, `github_repo`
2. One variable with default value - `location`

TERRAFORM

```
# variable.tf

 # No default value
 variable "instance_type" {
   type = string
   description = "EC2 Instance Type"
 }

 # No default value
 variable "tag" {
   type = string
   description = "The tag for the EC2 instance"
 }

 # default value for the variable location
 variable "location" {
   type = string
   description = "The project region"
   default = "eu-central1"
 }
```

Terraform variables.tf

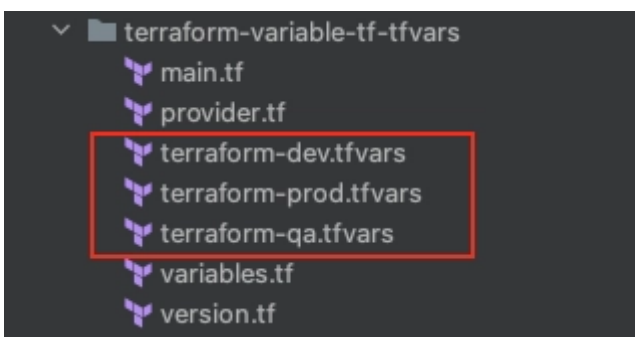
Terraform.tfvars is a file where you actually ***assign a values to the variables***. I am just gonna use the previous `variable.tf` and assign the values to the variables -

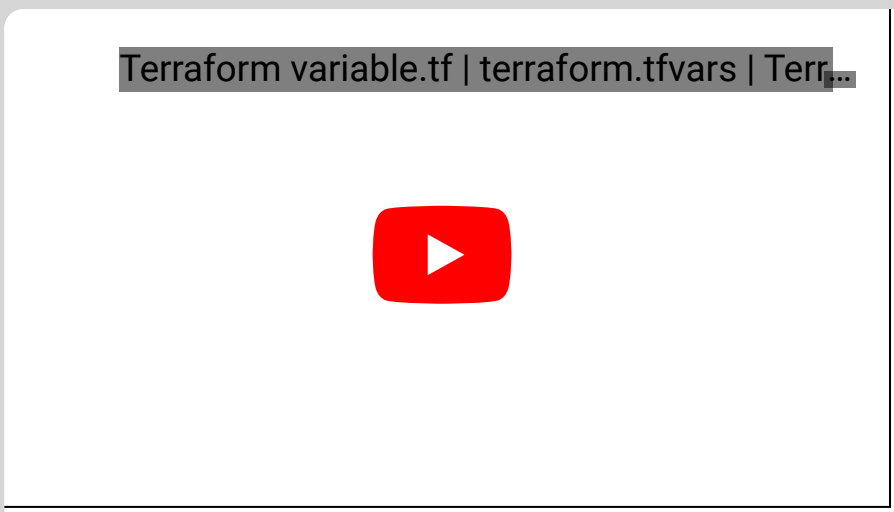
TERRAFORM

```
# terraform-dev.tfvars

instance_type    =   "t2.micro"
tag              =   "EC2 Instnace for DEV"
location         =    "eu-central-1"
```

Here are the benefits of using `terraform.tfvars` -

1. You can have multiple `terraform.tfvars` based on your project setup for example -
   - **DEV** - `terraform-dev.tfvars`
   - **QA** - `terraform-qa.tfvars`
   - **PROD** -`terraform-prod.tfvars`





Pre-requisite

Before we start working with Terraform variables, here are the pre-requisites -

1. How to create variable.tf?

Let's take a very basic example to understand the concept of `variable.tf` in terraform. In this example, we are going to set up an **EC2 Instance** on **AWS.**.

For setting up an **EC2 Instance** we will need the following information -

1. Region - `location`
2. Instance Type - `instance_type`
3. Tags - `tag`

1.1 Let's create variable.tf file for region, instance type, and tags -

Here is the code for `variable.tf` -

TERRAFORM

```
# variable.tf

# No default value
variable "instance_type" {
   type = string
   description = "EC2 Instance Type"
}

# No default value
variable "tag" {
   type = string
   description = "The tag for the EC2 instance"
}

# default value for the variable location
variable "location" {
   type = string
   description = "The project region"
```

1.2 Create main.tf for provisioning EC2 instanc

-

Here is the code for `main.tf` file

```bash
provider "aws" {
   region     = var.location
   access_key = "<INSERT_YOU_ACCESS_KEY>"
   secret_key = "<INSERT_YOU_SECRET_KEY>"
}

resource "aws_instance" "ec2_example" {

   ami           = "ami-0767046d1677be5a0"
   instance_type =  var.instance_type

   tags = {
           Name = var.tag
   }
}
```

2. How to create terraform.tfvars?

After creating the `variables.tf` in Step-1, let's create `.tfvars` and in that file, we are going to assign values to the variable -

1. `location` - "eu-central-1"
2. `instance_type` - "t2.micro"
3. `tag`- "EC2 Instnace for DEV"

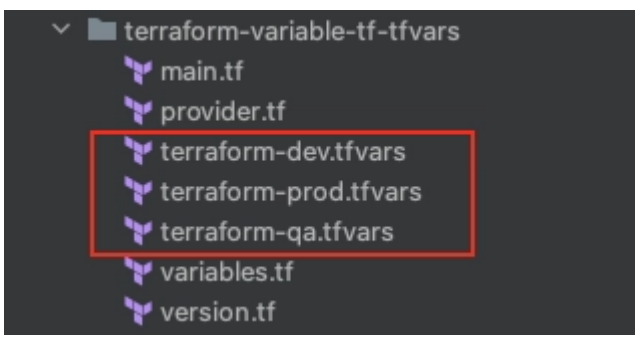
```
instance_type    =    "t2.micro"
tag              =    "EC2 Instnace for DEV"
location         =     "eu-central-1"
```

3. How to create multiple .tfvars files for different environments?

There can be a situation where you need to create multiple tfvars files based on the environment like `DEV, QA, PRODUCTION`.

So in such scenario, you can create one `tfvars` file for each environment -

1. terraform-dev.tfvars
2. terraform-qa.tfvars
3. terraform-prod.tfvars



Terraform tfvars files for DEV, QA, PROD environment

Here is the content of `terraform-dev.tfvars, terraform-qa.tfvars, terraform-prod.tfvars` based on different environment types -

DEV

```
instance_type    =    "t2.micro"
tag              =    "EC2 Instnace for DEV"
location         =     "eu-central-1"
```

QA

TERRAFORM

```
# terraform-qa.tfvars

instance_type    =    "t2.micro"
tag              =    "EC2 Instnace for QA"
location         =     "eu-central-1"
```

PROD

TERRAFORM

```
# terraform-prod.tfvars

instance_type    =    "t2.micro"
tag              =    "EC2 Instnace for PROD"
location         =     "eu-central-1"
```

4. How do you pass a variable(.tfvars) to the command line to Terraform using --var-file?

Referencing the same example from Step-2 we can pass the variables `terraform-dev.tfvars, terraform-qa.tfvars, terraform-prod.tfvars` based on the environment we are working.

Keep in mind that you have to supply the correct `.tfvars` file based on the environment you are working on.

```bash
# 1. terraform init for DEV

terraform init --var-file="terraform-dev.tfvars"

# 2. terraform plan for DEV

terraform plan --var-file="terraform-dev.tfvars"

# 3. terraform apply for DEV

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

QA - Keep in mind that you have to supply the correct `.tfvars` file based on the environment you are working on.

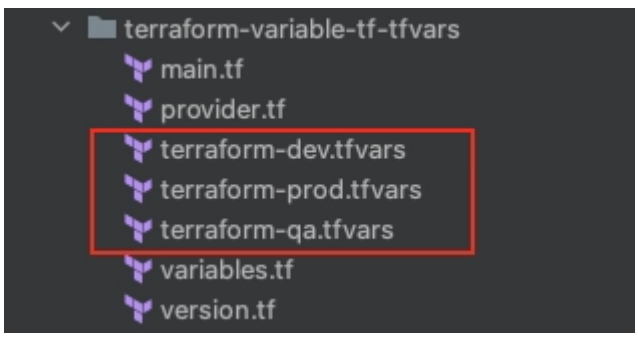
```bash
# 1. terraform init for QA

terraform init --var-file="terraform-qa.tfvars"

# 2. terraform plan for QA

terraform plan --var-file="terraform-qa.tfvars"

# 3. terraform apply for QA

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

PROD - Keep in mind that you have to supply the correct `.tfvars` file based on the environment you are working on.

```
terraform init --var-file="terraform-prod.tfvars"

# 2. terraform plan for PROD

terraform plan --var-file="terraform-prod.tfvars"

# 3. terraform apply for PROD

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

5. Best practices for using variable.tf and terraform.tfvars

1. **Separate reusable variables into a separate tfvars file:** Having a separate `.tfvars` file for all reusable variables provides clarity, readability, and maintainability. Below you will find a screenshot of my project where i have 3 different `tfvars` for `DEV, QA, PROD` environment -



Terraform tfvars files for DEV, QA, PROD environment

2. **Lockdown read/write access:** Ensure that only the appropriate people can modify the variable values, whether through IAM policies or other security controls.

s risk. Here is an example of how to set default values -

TERRAFORM

```
# default value assigned for variable `location`

variable "location" {
  type = string
  description = "The project region"
  default = "eu-central1"
}
```

4. **Reuse variable names:** Reusing variable names between variable.tf and terraform.tfvars helps reduce confusion and ensures that all variable values referenced in variable.tf can be overridden in terraform.tfvars.

5. **Organize variable values:** Organize, group, and document the variables to provide context and clarity. This will reduce the amount of time needed to understand what values are being used for

6. Difference between terraform.tfvars vs variables.tf

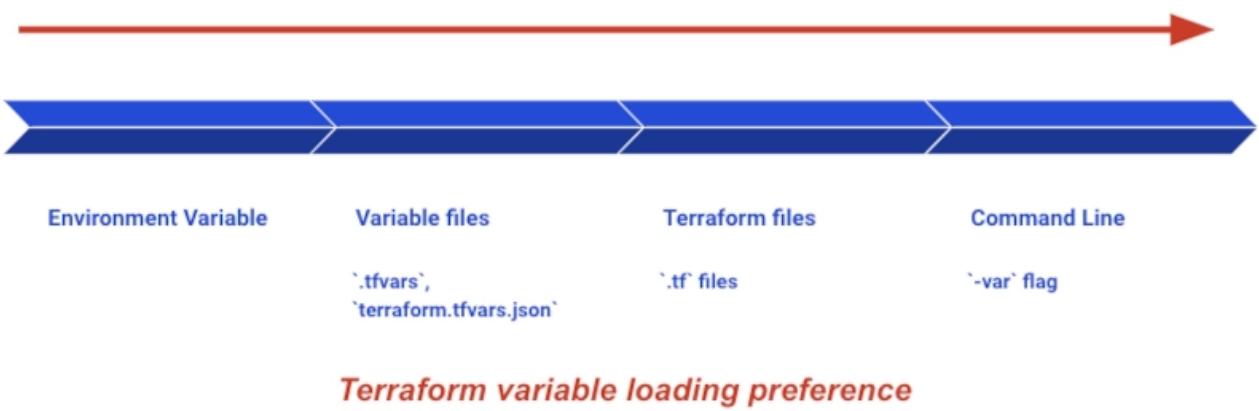
Here are my 5 key differences between `terraform.tfvars` and `variables.tf` -

Terraform.tfvars	variables.tf
It stores variable values	It stores variable definitions such as data type and possible values
Terraform.tfvars are local configuration files	variables.tf files are used to define variables in multiple environments

Variables in Terraform.tfvars are only valid for that particular environment	variables in variables.tf can be used across environments when defined properly
Terraform.tfvars provides default values for the variables declared in variables.tf	variables.tf do not provide any default values

7. Terraform variable loading preference - How do terraform loads variables?

Terraform variable loading preference refers to the order in which Terraform on loads variables when multiple sources specify the same variable. By default, Terraform looks in the following order to find variables with the same name.



Terraform variable loading preference

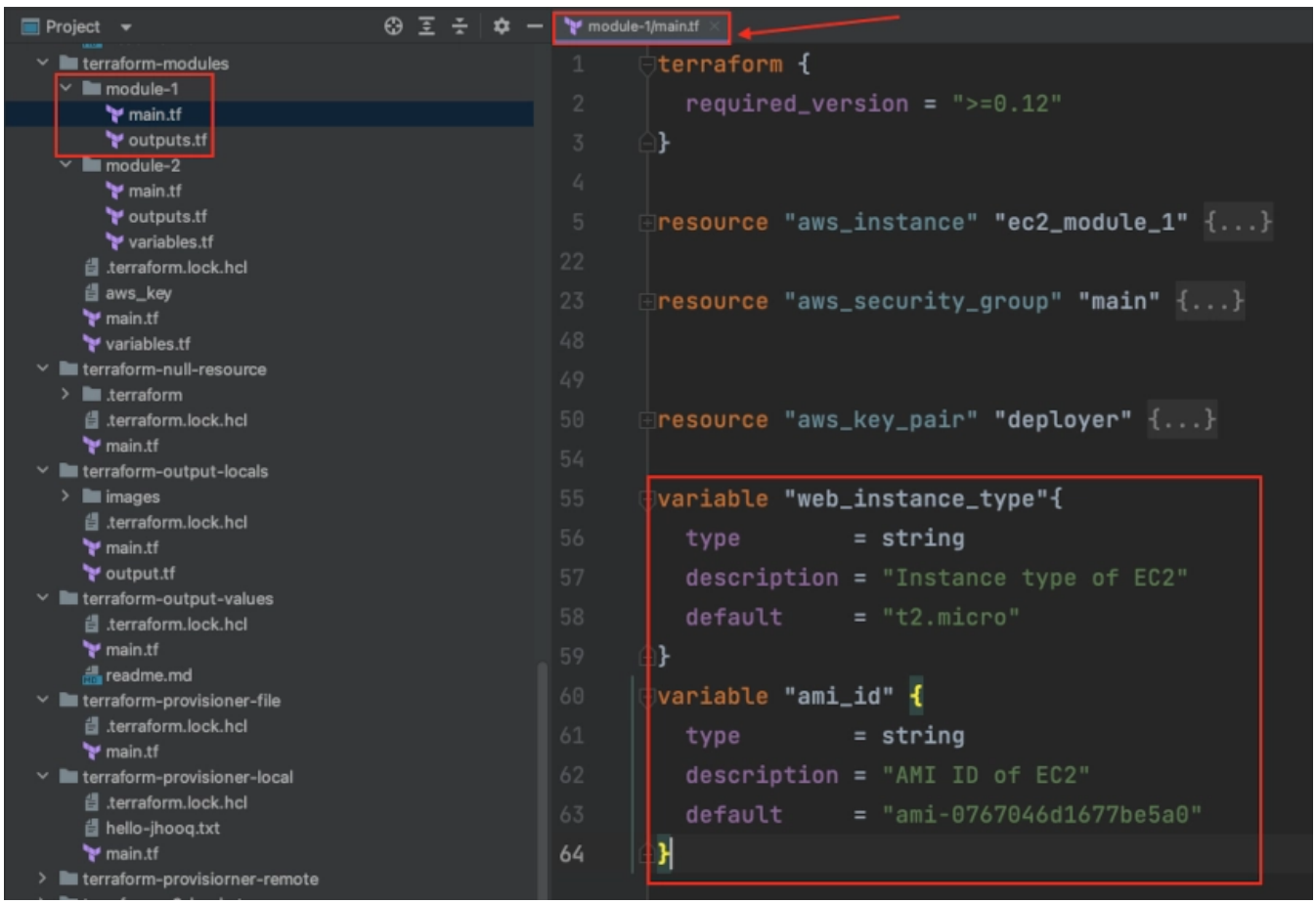
1. Environment variables
2. Variable files (files with a `.tfvars` or `terraform.tfvars.json` extension)
3. From Terraform files (using `.tf` files)

or in a variable file. If a variable is set in multiple sources, the ordering described above is followed.

8. How to pass variables into a module in Terraform?

There are three ways to define the ***variables*** for module -

1. **Define variables inside the module's `main.tf`:** The first and the easiest way to define the variables inside the `main.tf` of terraform module. Here is the screenshot of my project in which I have created `main.tf` and in the same file I have declared the variables -

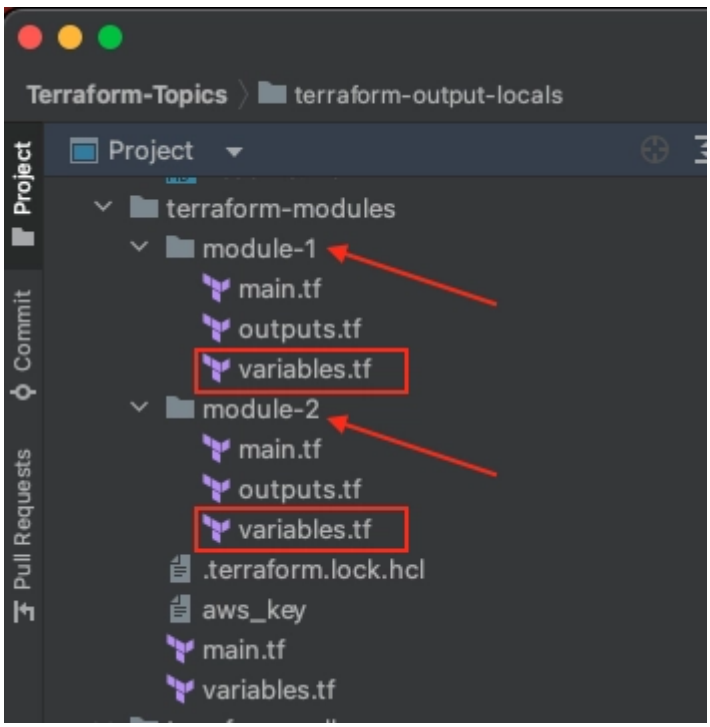


Declare variable inside the main.tf of module

so that your terraform code is more optimized.

Please have a look at the screenshot of my project structure. In this screenshot, you will find two modules -

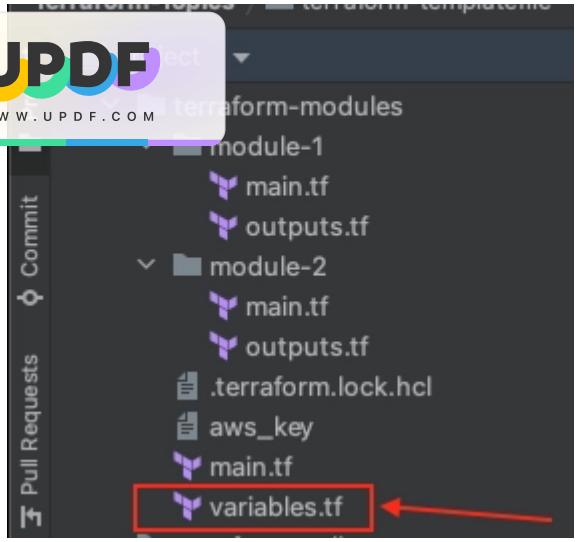
- **module-1 :** There is `variables.tf` for module-1
- **module-2 :** As well as there is separate `variables.tf` for module-2



Separate variable files for module-1 and module-2

3. **Common project level `variables.tf` for all modules:** The third option would be to create common `variables.tf` at the project level so that all the modules within that project can access the same `variables.tf`.

Here is the screenshot of the project in which I have created common `variables.tf` for all the modules -



Common variable.tf at project level for all the modules withing project

3. **Pass variables to modules from the command line:** Just like we pass `variable` from the command line in terraform similarly you can also the variables from the command line to all the modules within that project.

But the variables passed via the command line will override all the local variables.

Here are some example commands for passing the variables to modules -

BASH

```
terraform init --var-file="terraform.tfvars"

terraform plan--var-file="terraform.tfvars"

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

9. Conclusion

I hope this article will help you to understand the `variables.tf` and `terraform.tfvars` in more detailed way. You can clone my GitHub Repo for Terraform where I have created all the sample codes