

# Terraform Variables: A Complete Guide – Interview + Hashicorp Associate Exam

LinkedIn: [Amit Singh](#)

Medium:  Amit Singh – Medium

## 1. What Are Terraform Variables?

Terraform variables are used to make your configurations **dynamic**, **reusable**, and **flexible**. Instead of hardcoding values (like AMI ID, instance type), you define variables and assign values later.

## 2. Declaring Terraform Variables

You declare a variable like this in a variables.tf file:

```
variable "instance_type" {  
  description = "Type of EC2 instance"  
  type        = string  
  default     = "t2.micro"  
}
```

You can also define variables in:

- CLI using -var
- .tfvars files
- Environment variables

## 3. Types of Terraform Variables

Here are the major types you can use:

Type	Description	Example
string	A single text value	"t2.micro"
number	Numeric values	8080, 3.14
bool	Boolean values	true, false
list	Ordered list of values	["us-west-1", "us-east-1"]

map	Key-value pair collection	{ name = "dev", env = "prod" }
object	Complex structured type	{ name = string, age = number }
tuple	Fixed-length, typed list	[string, number, bool]

## 4. Input Variable Example

variables.tf:

```
variable "region" {
  type      = string
  description = "AWS region to deploy resources"
}
```

main.tf:

```
provider "aws" {
  region = var.region
}
```

terraform.tfvars:

```
region = "us-east-1"
```

## 5. Advanced Variable Concepts

- Validation Rules

```
variable "instance_count" {
  type = number

  validation {
    condition     = var.instance_count > 0
    error_message = "Must be greater than 0"
  }
}
```

- Sensitive Variables

```
variable "db_password" {
  type      = string
  sensitive = true
}
```

- Dynamic Default Values (using locals or functions)

## 6. Demo Use Case for All Variable Types

Let's provision an EC2 instance using variables of all types:

**variables.tf:**

```
variable "ami_id" {  
  type = string  
}  
  
variable "instance_count" {  
  type = number  
}  
  
variable "enable_monitoring" {  
  type = bool  
}  
  
variable "availability_zones" {  
  type = list(string)  
}  
  
variable "tags" {  
  type = map(string)  
}  
  
variable "server_config" {  
  type = object({  
    cpu    = number  
    memory = number  
  })  
}
```

**terraform.tfvars:**

```
ami_id          = "ami-0abcd1234efgh5678"  
instance_count  = 2  
enable_monitoring = true  
availability_zones = ["us-east-1a", "us-east-1b"]  
tags = {  
  Name = "MyEC2"  
  Env  = "Production"  
}  
server_config = {  
  cpu    = 2  
  memory = 4096  
}
```

## 7. Useful Terraform CLI Commands

Command	Purpose
terraform init	Initialize working directory
terraform plan	Show execution plan
terraform apply	Apply infrastructure changes
terraform destroy	Destroy managed infrastructure

## 8. Best Practices

- Keep variable files separate: variables.tf, terraform.tfvars
- Use descriptive variable names
- Use validation and sensitive flags
- Avoid hardcoding wherever possible

## HashiCorp Associate Certification Questions (Variables Only)

Q1. What is the correct syntax to declare a variable of type list(string)?

```
variable "names" {  
  type = list(string)  
  default = ["alice", "bob"]  
}
```

✓ *Appears in exam: YES (format validation + type check)*

Q2. What does this variable type mean?

```
variable "settings" {  
  type = map(string)  
}
```

➡ **Answer:** It accepts a key-value pair where all values are strings.

e.g., `{ env = "prod", region = "us-west-1" }`

✓ *Appears in exam: YES (map syntax understanding)*

Q3. What will this variable declaration accept?

```
variable "enable_feature" {  
  type = bool  
  default = true  
}
```

➡ **Answer:** Only true or false values (boolean).

✓ *Appears in exam: YES (default value typing)*

Q4. How can you pass variable values to Terraform?

- Using -var in CLI: `terraform apply -var="region=us-east-1"`
- Through a .tfvars file: `terraform apply -var-file="dev.tfvars"`
- Environment variables: `TF_VAR_region=us-east-1`

✓ *Appears in exam: YES (all 3 passing methods)*

Q5. What happens if you omit a default value and don't provide input?

➡ Terraform will **prompt you** for the variable at runtime.

✓ *Appears in exam: YES (basic behavior test)*

## Interview Questions Based on Terraform Variables

Q1. Can you explain the difference between list(string) and tuple?

- list(string) = a homogeneous list (e.g. all strings)
- tuple = heterogeneous list with specific order and types

➡ Example: `tuple([string, number, bool])`

■ *Asked in advanced interviews*

Q2. What are output variables, and when would you use them?

➡ Output variables are used to expose information from one module or root config to others or to users.

```
output "instance_ip" {  
  value = aws_instance.my_ec2.public_ip  
}
```

■ *Common in both interviews & real projects*

Q3. Can you assign types to variables? What happens if the type doesn't match?

➡ Yes. If you declare a type and pass the wrong type, Terraform will raise a validation error.

```
variable "count" {  
  type = number  
}
```

✓ Interview follow-up: "Can you coerce string "1" to number 1?" → No. It's strict typing.

Q4. What is the purpose of sensitive = true?

```
variable "password" {  
  type      = string  
  sensitive = true  
}
```

➡ Terraform will **hide this variable's value** in CLI outputs and logs.

■ Often asked in secure infrastructure/IAM interviews

Q5. How do you set complex types like object or map(object)?

```
variable "user" {  
  type = object({  
    name = string  
    age  = number  
  })  
}
```

➡ In .tfvars:

```
user = {  
  name = "Alice"  
  age  = 30  
}
```

■ *This is high-level, often asked in mid/senior interviews or exam.*

## Bonus Terraform Exam Practice (Variable Scenarios)

Concept	Sample Question	Answer/Tip
Default value	What happens if no value is passed to a variable without default?	Terraform prompts for input
Type mismatch	What if you assign a number to a string variable?	Terraform throws an error
Complex variable types	How do you define a variable that contains a map of lists of strings?	Use map(list(string))
Required variable file	What's the default filename used if -var-file is not specified?	terraform.tfvars
Valid list assignment	["a", "b", "c"] – What is this?	A list of strings

## Terraform Variables Practice Test (15 Questions)

### 1. What is the correct way to declare a string variable in Terraform?

- A. variable "env" { default = "prod" }
- B. variable "env" { type = str }
- C. variable "env" { type = "string" }
- D. variable env = "prod"

### 2. How do you pass a variable from CLI when running terraform apply?

- A. terraform apply --var region:us-west-1
- B. terraform apply -var="region:us-west-1"
- C. terraform apply -var="region=us-west-1"
- D. terraform apply var=region us-west-1

### 3. Which Terraform type represents a key-value structure where each value is a string?

- A. map
- B. object
- C. map(string)
- D. list(map)

**4. What will Terraform do if a variable is defined without a default and no value is provided?**

- A. Use an empty value
- B. Use the previous value from state
- C. Prompt for input
- D. Throw an error and exit

**5. What is the default variable file Terraform automatically loads?**

- A. main.tfvars
- B. terraform.variables
- C. terraform.auto.tfvars
- D. terraform.tfvars

**6. Which of the following is a valid declaration for a list of numbers?**

- A. type = list(number)
- B. type = number[]
- C. type = array(number)
- D. type = numbers

**7. What does this variable type represent?**

```
variable "my_input" {  
  type = object({  
    name    = string  
    enabled = bool  
  })  
}
```

- A. A map of any types
- B. A structure with fixed keys and defined types
- C. A tuple
- D. A dynamic type map

**8. You need a variable that accepts values like ["t2.micro", "t3.micro"]. Which type would you use?**

- A. map(string)
- B. object



- C. list(string)
- D. set(string)

**9. What is the effect of setting sensitive = true on a variable?**

- A. It disables the variable
- B. It hides the value from output and logs
- C. It encrypts the variable
- D. It forces re-creation of the resource

**10. What type does this represent? tuple([string, number, bool])**

- A. Heterogeneous ordered list
- B. Homogeneous list
- C. Object
- D. Map

**11. What will happen if a string is passed to a variable of type number?**

- A. It is automatically converted
- B. Terraform throws a type mismatch error
- C. It is ignored
- D. It gets set to 0

**12. Which method is NOT valid for assigning variables in Terraform?**

- A. .tfvars file
- B. -var-file CLI flag
- C. Environment variables starting with TF\_VAR\_
- D. Hardcoding values inside terraform.tfstate

**13. How do you make a variable optional in Terraform?**

- A. Add optional = true
- B. Add nullable = false
- C. Set a default value
- D. Define it in a local block

**14. What does the nullable attribute do in a variable block?**

- A. Makes the variable sensitive
- B. Allows the value to be null
- C. Prevents Terraform from prompting for input
- D. Forces a replan

**15. What is the output of the following if nothing is passed?**

```
variable "tags" {  
  type = map(string)  
}
```

- A. null
- B. {}
- C. Error (no default)
- D. Empty list

## Answers

Q	Answer	Explanation
1	C	Correct type is "string"
2	C	Correct CLI syntax is - var="key=value"
3	C	map(string) = map where each value is a string
4	C	Terraform prompts for input
5	D	Terraform loads terraform.tfvars by default
6	A	Valid syntax for list of numbers
7	B	Object with fixed schema
8	C	A list of strings
9	B	Hides values in output/logs
10	A	Tuple = heterogeneous ordered list
11	B	Type mismatch → error
12	D	You should never hardcode in .tfstate
13	C	Default makes variable optional. Can be overridden
14	B	Allows passing null explicitly
15	C	No default = prompt or error

# Terraform Variables Practice Test – Set 1

## Q1. What is the correct way to declare a string variable in Terraform?

- A. variable "env" { default = "prod" }
- B. variable "env" { type = str }
- C. variable "env" { type = "string" }
- D. variable env = "prod"

✅ **Answer:** C

**Explanation:** type = "string" is the correct syntax for defining a string variable.

## Q2. How do you pass a variable from CLI when running terraform apply?

- A. terraform apply --var region:us-west-1
- B. terraform apply -var="region:us-west-1"
- C. terraform apply -var="region=us-west-1"
- D. terraform apply var=region us-west-1

✅ **Answer:** C

**Explanation:** CLI syntax is -var="key=value"

## Q3. Which Terraform type represents a key-value structure where each value is a string?

- A. map
- B. object
- C. map(string)
- D. list(map)

✅ **Answer:** C

**Explanation:** map(string) ensures all values in the map are strings.

## Q4. What will Terraform do if a variable is defined without a default and no value is provided?

- A. Use an empty value
- B. Use the previous value from state
- C. Prompt for input
- D. Throw an error and exit

✅ **Answer:** C

**Explanation:** Terraform prompts for input for required variables.

## Q5. What is the default variable file Terraform automatically loads?

- A. main.tfvars
- B. terraform.variables

- C. terraform.auto.tfvars
- D. terraform.tfvars

✅ **Answer:** D

**Explanation:** terraform.tfvars and \*.auto.tfvars are automatically loaded.

### Q6. Which of the following is a valid declaration for a list of numbers?

- A. type = list(number)
- B. type = number[]
- C. type = array(number)
- D. type = numbers

✅ **Answer:** A

**Explanation:** list(number) is valid in HCL2.

### Q7. What does this variable type represent?

```
variable "my_input" {  
  type = object({  
    name    = string  
    enabled = bool  
  })  
}
```

- A. A map of any types
- B. A structure with fixed keys and defined types
- C. A tuple
- D. A dynamic type map

✅ **Answer:** B

**Explanation:** Objects define a strict schema of key-value pairs.

### Q8. You need a variable that accepts values like ["t2.micro", "t3.micro"]. Which type would you use?

- A. map(string)
- B. object
- C. list(string)
- D. set(string)

✅ **Answer:** C

**Explanation:** list(string) represents an ordered collection of strings.

### Q9. What is the effect of setting sensitive = true on a variable?

- A. It disables the variable
- B. It hides the value from output and logs

- C. It encrypts the variable
- D. It forces re-creation of the resource

✅ **Answer:** B

**Explanation:** Sensitive values are redacted in CLI/UI output.

### Q10. What type does this represent?

```
tuple([string, number, bool])
```

- A. Heterogeneous ordered list
- B. Homogeneous list
- C. Object
- D. Map

✅ **Answer:** A

**Explanation:** Tuple holds ordered values of mixed types.

### Q11. What will happen if a string is passed to a variable of type number?

- A. It is automatically converted
- B. Terraform throws a type mismatch error
- C. It is ignored
- D. It gets set to 0

✅ **Answer:** B

**Explanation:** You must explicitly convert using tonumber().

### Q12. Which method is NOT valid for assigning variables in Terraform?

- A. .tfvars file
- B. -var-file CLI flag
- C. Environment variables starting with TF\_VAR\_
- D. Hardcoding values inside terraform.tfstate

✅ **Answer:** D

**Explanation:** .tfstate is managed by Terraform and should not be edited.

### Q13. How do you make a variable optional in Terraform?

- A. Add optional = true
- B. Add nullable = false
- C. Set a default value
- D. Define it in a local block

✅ **Answer:** C

**Explanation:** A variable is optional if a default value is provided.

### Q14. What does the nullable attribute do in a variable block?

- A. Makes the variable sensitive
- B. Allows the value to be null
- C. Prevents Terraform from prompting for input
- D. Forces a replan

✅ **Answer:** B

**Explanation:** nullable = true lets users explicitly pass null.

### Q15. What is the output of the following if nothing is passed?

```
variable "tags" {  
  type = map(string)  
}
```

- A. null
- B. {}
- C. Error (no default)
- D. Empty list

✅ **Answer:** C

**Explanation:** No default value = required input. Terraform will error if not provided.

## Terraform Variables – Practice Test (Set 2: Advanced)

### Q1. Which keyword is used to restrict acceptable values for a variable in Terraform 0.13+?

- A. type\_check
- B. validation
- C. constraint
- D. rule

✅ **Answer:** B

**Explanation:** The validation block allows you to define custom rules for variable values.

### Q2. What does the following block do?

```
variable "env" {  
  type = string  
  
  validation {  
    condition      = contains(["dev", "stage", "prod"], var.env)  
    error_message = "env must be one of dev, stage, or prod."  
  }  
}
```

- A. Converts env to lowercase
- B. Restricts input to one of 3 values
- C. Converts list to string
- D. Creates 3 separate variables

✅ **Answer:** B

**Explanation:** The validation block ensures env can only be "dev", "stage", or "prod".

### Q3. What is the use of can() function in variable validation or conditionals?

- A. Checks if a variable is empty
- B. Checks if a block exists
- C. Evaluates if an expression can run without error
- D. Executes a command

✅ **Answer:** C

**Explanation:** can() safely tests expressions that might fail (e.g., can(regex(...))).

### Q4. What is the difference between object and map in Terraform types?

- A. Objects are always strings; maps are not
- B. Maps have strict keys; objects don't
- C. Objects are fixed-key structures; maps are dynamic
- D. No difference

✅ **Answer:** C

**Explanation:** object has a defined schema; map is dynamic with uniform value types.

### Q5. How would you access a specific object field in a module input variable?

```
variable "user_info" {  
  type = object({  
    name = string  
    age  = number  
  })  
}
```

What's the correct way to access the age?



- A. user\_info["age"]
- B. var.user\_info.age
- C. var.age
- D. user\_info.age

✓ **Answer:** B

**Explanation:** You use var.<variable>.<key> for object fields.

### Q6. Which command creates a .tfplan file based on the variables passed?

- A. terraform plan --out=tfplan
- B. terraform apply tfplan
- C. terraform init tfplan
- D. terraform validate --save=tfplan

✓ **Answer:** A

**Explanation:** --out=tfplan saves the execution plan to a binary file.

### Q7. What is the result of assigning a null value to a variable with no default and marked as required?

- A. It uses system default
- B. It throws an error
- C. It converts null to 0
- D. It sets the value as empty string

✓ **Answer:** B

**Explanation:** Required variables cannot be null unless explicitly allowed (nullable = true).

### Q8. How do you define a complex nested object variable?

```
variable "config" {  
  type = object({  
    db = object({  
      host = string  
      port = number  
    })  
    tags = map(string)  
  })  
}
```

What is this useful for?

- A. Dynamically importing cloud resources
- B. Managing structured data inputs
- C. Disabling default values
- D. Creating local variables

✓ **Answer:** B

**Explanation:** Nested objects are used for grouped, structured inputs.

**Q9. How can you enforce a variable to be not null, even if a default is provided?**

- A. Use nullable = false
- B. Use sensitive = false
- C. Set required = true
- D. It's not possible

✓ **Answer:** A

**Explanation:** nullable = false ensures the variable cannot accept null.

**Q10. What will this block output if nothing is passed and no default is set?**

```
variable "team" {  
  type = list(string)  
}
```

- A. An error – required variable
- B. Empty list
- C. null
- D. Terraform skips it

✓ **Answer:** A

**Explanation:** A variable with no default is required and will throw an error.

**Q11. What type will this return?**

```
type = tuple([string, list(number), bool])
```

- A. A list of same types
- B. A fixed-structure mixed list
- C. A map
- D. An object

✓ **Answer:** B

**Explanation:** Tuple allows heterogeneous types in specific order.

**Q12. How do you securely hide a sensitive input variable in Terraform Cloud UI?**

- A. type = sensitive
- B. sensitive = true
- C. hidden = true
- D. secure = yes

✓ **Answer:** B

**Explanation:** sensitive = true redacts the value in CLI and UI.

### Q13. What will happen if two variables have the same name in two modules?

- A. Terraform will crash
- B. Variables will overwrite each other
- C. They will remain isolated to their modules
- D. Only one will be used globally

✓ **Answer:** C

**Explanation:** Module variables are scoped – they don't conflict across modules.

### Q14. Which of the following is true about terraform.tfvars.json?

- A. It cannot contain nested structures
- B. It supports only strings
- C. It's automatically loaded if present
- D. It must be manually sourced every time

✓ **Answer:** C

**Explanation:** Terraform auto-loads terraform.tfvars.json just like .tfvars.

### Q15. Which expression evaluates to a default if the variable is empty?

- A. var.instance\_type ?? "t2.micro"
- B. coalesce(var.instance\_type, "t2.micro")
- C. var.instance\_type || "t2.micro"
- D. ifnull(var.instance\_type, "t2.micro")

✓ **Answer:** B

**Explanation:** coalesce() returns the first non-null argument.

## Terraform variable - PART 2

### 1. Declaring Variables in a Separate File (variable.tf)

#### Scenario:

You want to keep your Terraform code clean by separating variables from the main logic.

#### How it works:

- main.tf: Contains provider and resource.
- variable.tf: Contains variable declaration (no values).

#### Demo:

- Define instance\_type in variable.tf like:

```
variable "instance_type" {  
  type = string  
}
```

- Then assign the value directly in main.tf or use default, or other methods.

## 2. Using .tfvars Files

### Scenario:

You want to **provide values separately**, keeping your code flexible.

### Files involved:

- main.tf: Terraform configuration.
- variable.tf: Declare variables.
- terraform.tfvars: Assign values like:

```
instance_type = "t2.micro"
```

### Why it's useful:

You can now reuse the same .tf files and change values without editing them directly.

## 3. Multiple .tfvars for Different Environments

### Scenario:

You want to deploy the same infrastructure to different environments: staging, production, etc.

### How it's done:

- main.tf: Uses variables instance\_type, environment\_name.
- variable.tf: Declares those variables.
- staging.tfvars: Sets values for staging:

```
instance_type  = "t2.micro"  
environment_name = "stage"
```

- production.tfvars: Sets values for production:

```
instance_type  = "t3.medium"  
environment_name = "prod"
```

### Command Example:

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

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

✅ This is **real-world best practice**—you don't hardcode environment-specific settings.

## 4. Passing Variables from the Command Line

### Scenario:

You want to override variable values quickly from the CLI without using a `.tfvars` file.

### Command Example:

```
terraform apply -var="instance_type=t2.micro"
```

### Why it's useful:

You don't need extra files for quick tests or automation via scripts.

## Conclusion + Cleanup

Rahul also teaches:

- How to destroy resources to avoid AWS charges:

terraform destroy

- How to verify everything on the **AWS EC2 dashboard**.
- All the code used is available on his **guide/website/Youtube channel** (Jhook).

## Summary Table

Method	File Name	Purpose
Basic Variable (Inline)	main.tf	Declare and use variables in one file
Clean Setup	main.tf + variable.tf	Separate logic and variable declaration
Environment Management	main.tf, variable.tf, *.tfvars	Use different .tfvars for staging, prod
Quick Overrides	Command Line (-var)	Pass variables directly from CLI

## Using variable.tf to Separate Declaration

Variables declared separately; values provided by default or in .tfvars.

main.tf

```
provider "aws" {  
    region = "us-east-1"  
}  
  
resource "aws_instance" "example" {  
    ami          = "ami-0c55b159cbfafa1f0"  
    instance_type = var.instance_type  
    tags = {  
        Name = "FromVariableTF"  
    }  
}
```

variable.tf

```
variable "instance_type" {  
    type = string  
}
```

terraform.tfvars

```
instance_type = "t2.micro"
```

---

## Multiple .tfvars Files for Different Environments

Reusability for staging, production, etc.

main.tf

```
provider "aws" {  
    region = "us-east-1"  
}  
  
resource "aws_instance" "example" {  
    ami           = "ami-0c55b159cbfafa1f0"  
    instance_type = var.instance_type  
    tags = {  
        Name = var.environment_name  
    }  
}
```

variable.tf

```
variable "instance_type" {  
    type = string  
}  
  
variable "environment_name" {  
    type = string  
}
```

staging.tfvars

```
instance_type = "t2.micro"  
environment_name = "staging"
```

production.tfvars

```
instance_type = "t3.medium"  
environment_name = "production"
```

Command to Apply

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

# Pass Variables via Command Line

Great for automation or scripting.

main.tf

```
provider "aws" {  
    region = "us-east-1"  
}  
  
resource "aws_instance" "example" {  
    ami          = "ami-0c55b159cbfafa1f0"  
    instance_type = var.instance_type  
    tags = {  
        Name = "FromCLI"  
    }  
}
```

variable.tf

```
variable "instance_type" {  
    type = string  
}
```

Command to Apply

`terraform apply -var="instance_type=t2.micro"`

Clean Up After Demo

`terraform destroy`

## Topic: Terraform Variables and .tfvars

### 1. What is the purpose of a terraform.tfvars file in Terraform?

- A) To define provider configurations
- B) To define backend configurations
- C) To store variable **values** used in a configuration
- D) To override main.tf settings

**Correct Answer: C**



## 2. Which command correctly passes a variable from the command line?

- A) terraform apply var=instance\_type=t2.micro
- B) terraform apply --instance\_type=t2.micro
- C) terraform apply -var="instance\_type=t2.micro"
- D) terraform apply -tfvars="instance\_type=t2.micro"

**Correct Answer: C**

## 3. Which file should you use to declare a variable without assigning a value?

- A) main.tf
- B) terraform.tfvars
- C) provider.tf
- D) variables.tf

**Correct Answer: D**

## 4. Which of the following is true about multiple .tfvars files like dev.tfvars, prod.tfvars?

- A) Only one .tfvars file is supported by Terraform
- B) Terraform automatically loads all .tfvars files in the directory
- C) You must manually specify which .tfvars to use
- D) They are deprecated in the latest version

**Correct Answer: C**

## 5. How do you instruct Terraform to use a specific .tfvars file during terraform apply?

- A) terraform apply --load="dev.tfvars"
- B) terraform apply --tfvars dev.tfvars
- C) terraform apply -var-file="dev.tfvars"
- D) terraform apply dev.tfvars

**Correct Answer: C**

## 6. Which variable type is best for storing AWS tags like Name, Environment, Owner in key-value format?

- A) list
- B) map
- C) string
- D) set

**Correct Answer: B**

## 7. What will happen if you declare a variable in variable.tf but do not assign any value using default, .tfvars, or CLI?

- A) Terraform will use a system default value
- B) Terraform will skip that variable
- C) Terraform will throw an error and prompt you
- D) Terraform will assume it as null silently

**Correct Answer: C**

## 8. What is the precedence order when assigning variable values in Terraform? (Highest to lowest)

- A) Environment variables > Default in variables.tf > CLI
- B) CLI > Environment variables > terraform.tfvars > default
- C) terraform.tfvars > CLI > default > env
- D) CLI > terraform.tfvars > env > default

**Correct Answer: B**

## Terraform Variables & .tfvars – Interview Key Points

### 1. Purpose of terraform.tfvars

- Used to **assign values to input variables**.
- Helps separate config from environment-specific values (e.g., dev, prod).

### 2. Declaring Variables

- Declared in variables.tf using the variable block.
- No default = prompts user at terraform apply.

```
variable "instance_type" {  
  description = "EC2 instance type"  
  type        = string  
}
```

### 3. Assigning Variable Values – 4 Ways (Precedence Order)

*Highest to Lowest:*

1. **CLI**: -var="key=value" or -var-file="dev.tfvars"
2. **Environment variables**: TF\_VAR\_<var\_name>=<value>
3. **terraform.tfvars** or \*.auto.tfvars
4. **Default** in variable block

### 4. Passing Variables via CLI

```
terraform apply -var="instance_type=t2.micro"
terraform apply -var-file="dev.tfvars"
```

## 5. .tfvars Usage

- .tfvars files store **key-value pairs**.
- Used to **override defaults** for different environments.

```
instance_type = "t3.micro"
region        = "us-west-2"
```

- Must be manually specified with -var-file, unless it's named terraform.tfvars.

## 6. Variable Types

- **string**: single text value
- **list**: ordered values [ "a", "b" ]
- **map**: key-value (best for AWS tags)
- **bool, number, set**: advanced use cases

## 7. Prompting for Missing Values

- If a variable has no default and no value provided via .tfvars, CLI, or env, **Terraform will prompt the user** interactively.

## 8. Best Practices

- Use terraform.tfvars for shared defaults.
- Use dev.tfvars, prod.tfvars for environment-specific overrides.
- Do **not hardcoded** values inside main.tf.

## Why .tfvars is useful:

**"You can now reuse the same .tf files and change values without editing them directly."**

Let me break it down simply:

### Without .tfvars:

Imagine you have a main.tf like this:

```
resource "aws_instance" "web" {
  ami        = "ami-0abcd1234"
  instance_type = "t2.micro"
```

}

If you want to deploy to **production** with a different instance type or AMI, you'd have to **edit this file manually every time**. That's not reusable or scalable.

## With .tfvars:

You declare variables in your .tf file:

```
variable "instance_type" {}
variable "ami" {}

resource "aws_instance" "web" {
  ami          = var.ami
  instance_type = var.instance_type
}
```

Now you create **different .tfvars files**:

### dev.tfvars

```
ami          = "ami-dev123"
instance_type = "t2.micro"
```

### prod.tfvars

```
ami          = "ami-prod456"
instance_type = "t3.large"
```

Then run:

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


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

💡 Result:

- ✓ You **don't touch your main .tf files**.
- ✓ You **reuse the same Terraform code** across **multiple environments** (dev, staging, prod).
- ✓ You avoid accidental changes to infrastructure code.

In Interview Terms:

Using .tfvars files **decouples configuration from code**, allowing safe, repeatable, and environment-specific deployments.



## Terraform Variables Interview Q&A

#	Question	One-Liner Answer
1	How do you declare a string variable in Terraform?	variable "name" { type = string }
2	Which file is automatically loaded by Terraform for variable values?	terraform.tfvars and *.auto.tfvars
3	What happens if a required variable has no default and no input?	Terraform prompts the user for input
4	How do you pass a variable from the CLI?	terraform apply -var="key=value"
5	What is the correct syntax for a list of strings in Terraform?	type = list(string)
6	Which type defines strict key-value pairs with defined types?	object({ key = type, ... })
7	What does sensitive = true do for a variable?	Hides value from logs and outputs
8	How do you declare a variable without assigning a value?	Define in variables.tf without default
9	What's the default behavior if no value is passed and no default is set?	Terraform throws an error or prompts
10	What type is represented by tuple([string, number])?	Heterogeneous ordered list
11	How do you restrict acceptable values for a variable?	Use the validation block
12	What is the purpose of the validation block?	Enforce custom rules for inputs
13	What does the can() function do?	Tests if expression can run without error
14	How do object and map types differ?	object has fixed keys, map is dynamic

15	How do you access an object's property in a variable?	var.variable_name.key
16	Which command saves the execution plan to a file?	terraform plan --out=tfplan
17	What happens if null is passed to a non-nullable variable?	Terraform throws an error
18	Why use a nested object in a variable?	For structured or grouped data
19	How do you make a variable accept null values?	Use nullable = true
20	What type does list(map(string)) represent?	List of maps with string values
21	What is the use of a .tfvars file?	Stores variable values separately
22	How do you specify a .tfvars file during apply?	terraform apply -var-file="file.tfvars"
23	Can you use multiple .tfvars files for environments?	Yes, like dev.tfvars, prod.tfvars
24	Which file should contain variable declarations only?	variables.tf
25	What's the variable value precedence order in Terraform?	CLI > Env vars > .tfvars > Defaults
26	How do you override a variable quickly without a file?	Use -var="key=value" on the CLI
27	Why is terraform.tfvars useful in real-world use?	Separates environment values from logic
28	How does Terraform handle environment variable inputs?	Reads TF_VAR_<name> automatically
29	What file should be used to separate variable values for dev/prod?	Separate .tfvars files
30	Which format does terraform.tfvars.json use and is it auto-loaded?	JSON format; Yes, it's auto-loaded



**Thanks Everyone!**

Connect with me: [Amit Singh](#)

