Modules

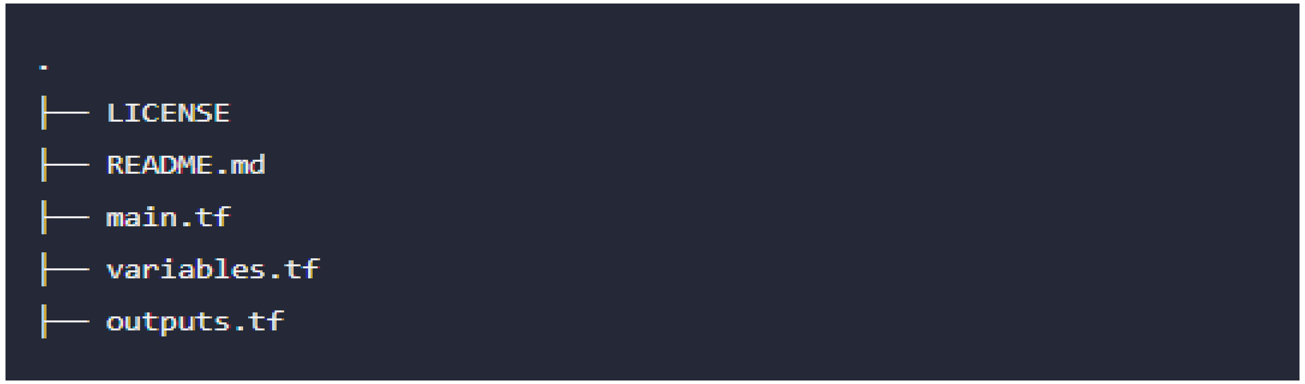
Ow

Definition

- A module is a container for multiple resources that are used together.
- Every Terraform configuration has at least one module, known as its root module, which consists of the resources defined in the .tf files in the main working directory.
- A module can call other modules, which lets you include the child module's resources into the configuration in a concise way. Modules can also be called multiple times, either within the same configuration or in separate configurations, allowing resource configurations to be packaged and re-used.

Root Module

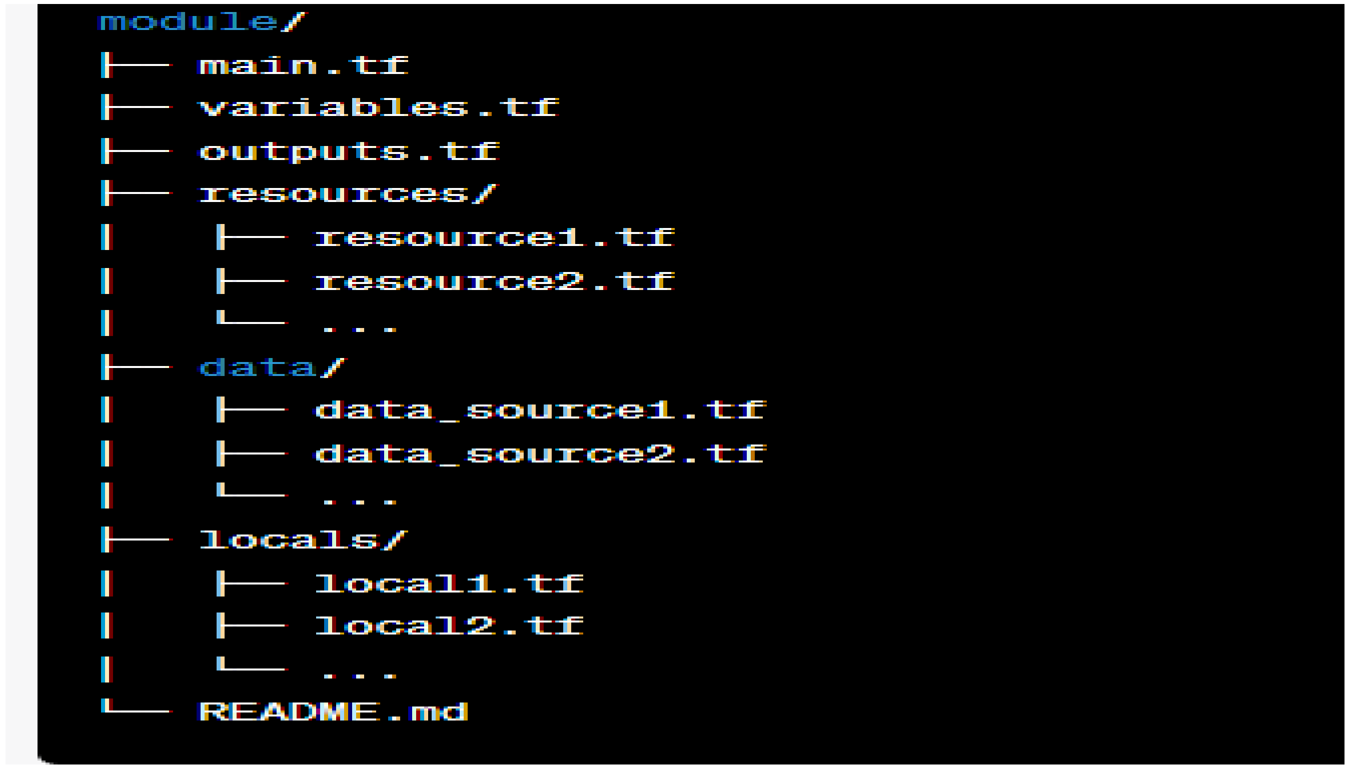
- A Terraform module is a set of Terraform configuration files in a single directory. Even a simple configuration consisting of a single directory with one or more .tf files is a module. When you run Terraform commands directly from such a directory, it is considered the root module



```
.
├── LICENSE
├── README.md
├── main.tf
├── variables.tf
├── outputs.tf
```

Module Structure

- When structuring a Terraform module, it is common to follow a recommended directory structure for better organization and maintainability. Here is an example of a typical Terraform module structure:



```
module/
├── main.tf
├── variables.tf
├── outputs.tf
├── resources/
│   ├── resource1.tf
│   ├── resource2.tf
│   └── ...
├── data/
│   ├── data_source1.tf
│   ├── data_source2.tf
│   └── ...
├── locals/
│   ├── local1.tf
│   ├── local2.tf
│   └── ...
└── README.md
```

..

main.tf: The main configuration file for the module. It contains the definition of resources, data sources, and other Terraform constructs specific to the module.

variables.tf: This file defines input variables that allow customization of the module. Declare variables, specify their types, and define any defaults or validation rules.

outputs.tf: Here, you define the outputs of the module. These outputs represent values that can be accessed or used by the calling module or the Terraform CLI.

resources/: This directory contains individual resource configuration files (resource1.tf, resource2.tf, etc.). Each file defines the resources created by the module, such as AWS instances, security groups, or S3 buckets.

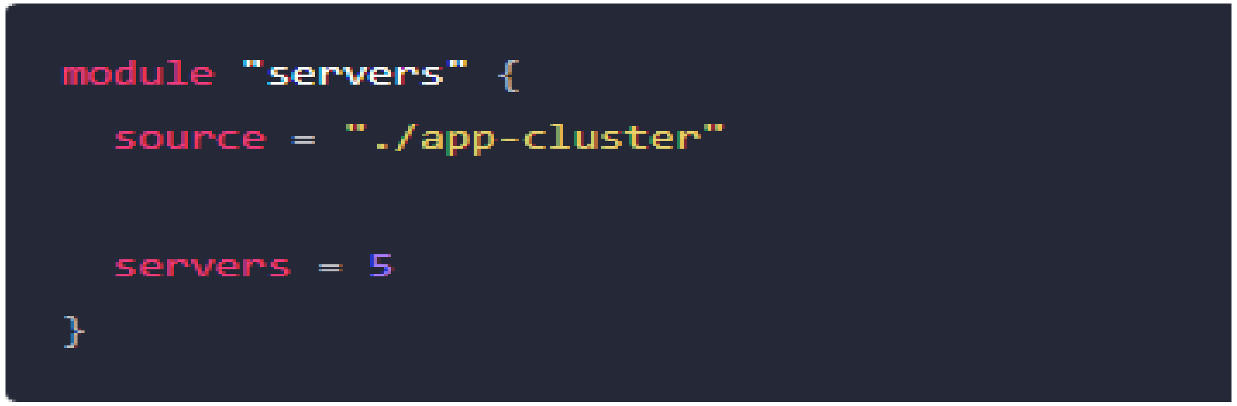
data/: This directory contains individual data source configuration files (data_source1.tf, data_source2.tf, etc.). Each file defines data sources used by the module to fetch information from external systems.

locals/: This directory contains individual local value configuration files (local1.tf, local2.tf, etc.). Each file defines local values or computations that can be used within the module.

README.md: An optional file that provides documentation for the module. You can include usage examples, important notes, and any other relevant information.

Calling module

- To call a module means to include the contents of that module into the configuration with specific values for its input variables. Modules are called from within other modules using module blocks:



```
module "servers" {
  source = "./app-cluster"

  servers = 5
}
```

Within the block body (between { and }) are the arguments for the module. Module calls use the following kinds of arguments:

The source argument is mandatory for all modules.

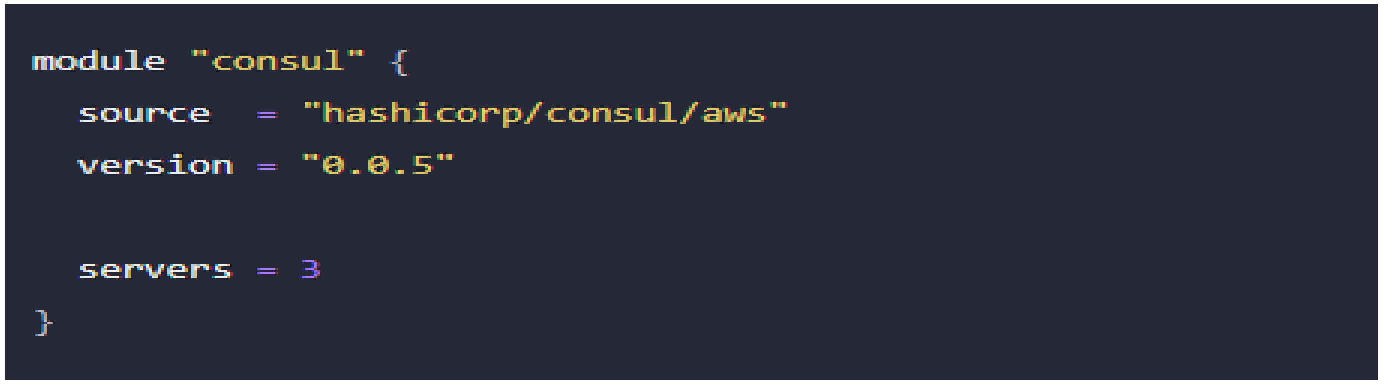
The version argument is recommended for modules from a registry.

Source

- All modules require a source argument, which is a meta-argument defined by Terraform.
- Its value is either the path to a local directory containing the module's configuration files, or a remote module source that Terraform should download and use
- The same source address can be specified in multiple module blocks to create multiple copies of the resources defined within, possibly with different variable values.
- After adding, removing, or modifying module blocks, you must re-run terraform init to allow Terraform the opportunity to adjust the installed modules

Version

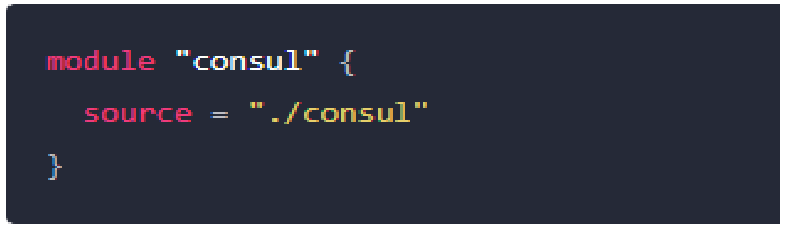
- When using modules installed from a module registry, we recommend explicitly constraining the acceptable version numbers to avoid unexpected or unwanted changes.
- Use the version argument in the module block to specify versions:



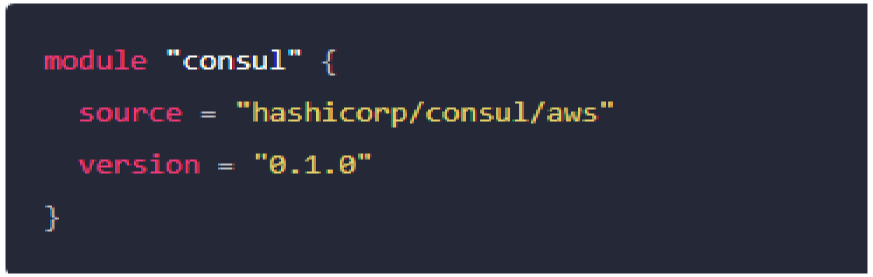
```
module "consul" {
  source  = "hashicorp/consul/aws"
  version = "0.0.5"

  servers = 3
}
```

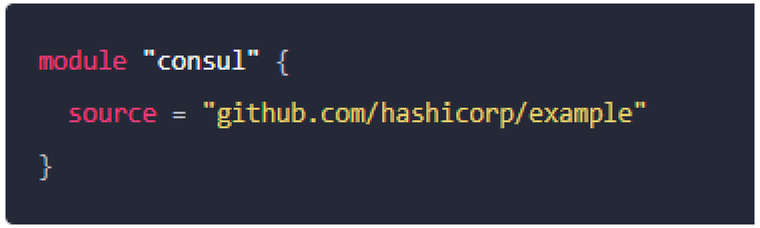
Module Sources



```
module "consul" {
  source = "./consul"
}
```



```
module "consul" {
  source = "hashicorp/consul/aws"
  version = "0.1.0"
}
```



```
module "consul" {
  source = "github.com/hashicorp/example"
}
```