Section 7

Understanding and Using terraform Modules

Modules: Terraform reusable infrastructure

The first step in Terraform is to create a working directory on your local machine for a Terraform project to provision an infrastructure to organize all the related terraform code and state file.

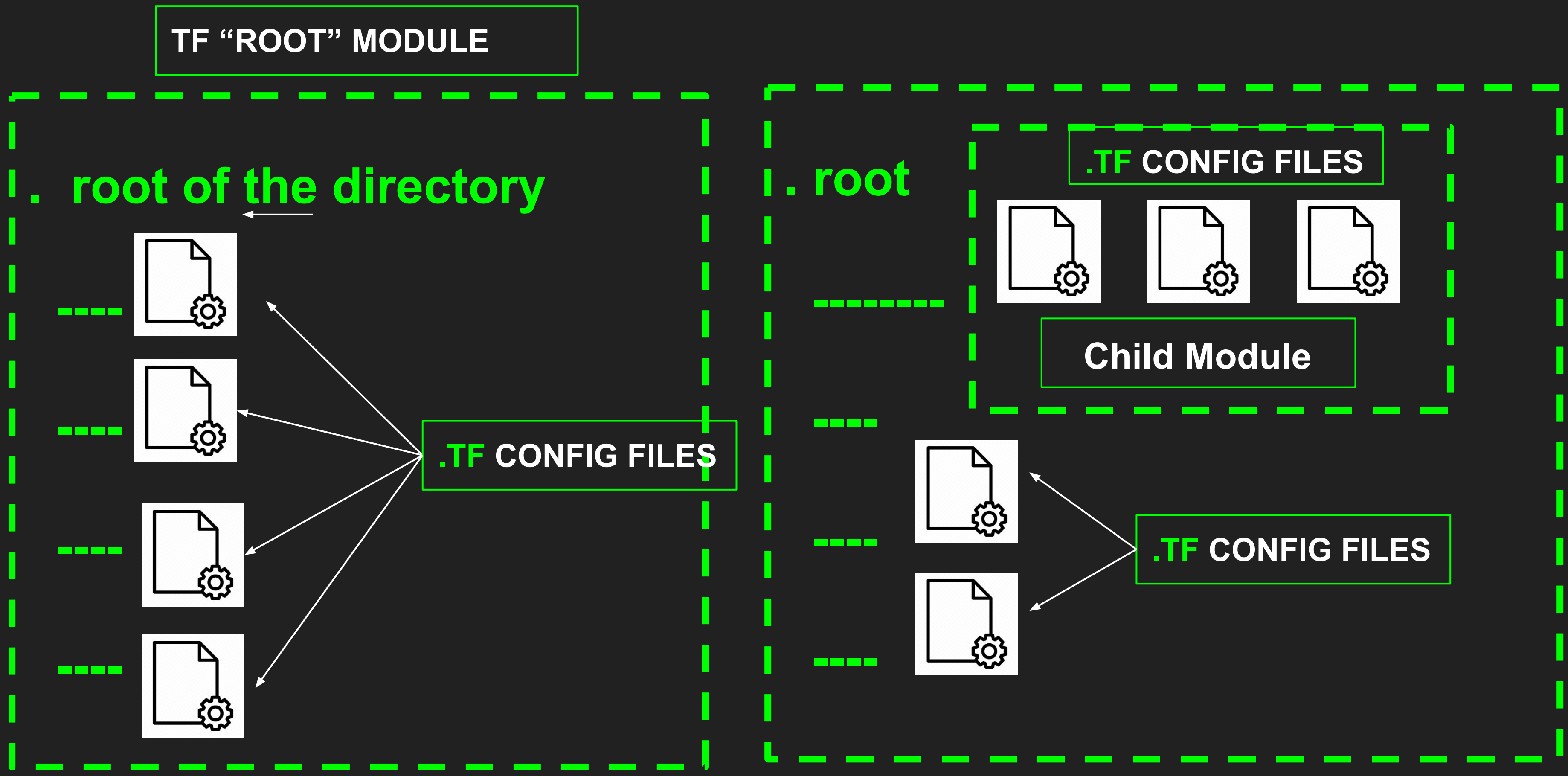
TF module == working directory with .tf files

Root TF module is module with .tf files located at the root of the directory

Additionals Modules can be created within the root module to provide a reusable code in the root module configuration files.

Modules can also be stored in VCS and versioned.

A good use case if for company to use modules to define configuration blueprints for resources to enforce best practices and config consistency for users in a team.


TF "ROOT" MODULE
. root of the directory
.TF CONFIG FILES
. root
.TF CONFIG FILES
Child Module
.TF CONFIG FILES

Modules created in a root module have to be initialized when called upon as child module with the following command:

```
$ terraform init or terraform get
```

A module subdir in the .terraform directory will be automatically created

Calling Child modules in your configuration.

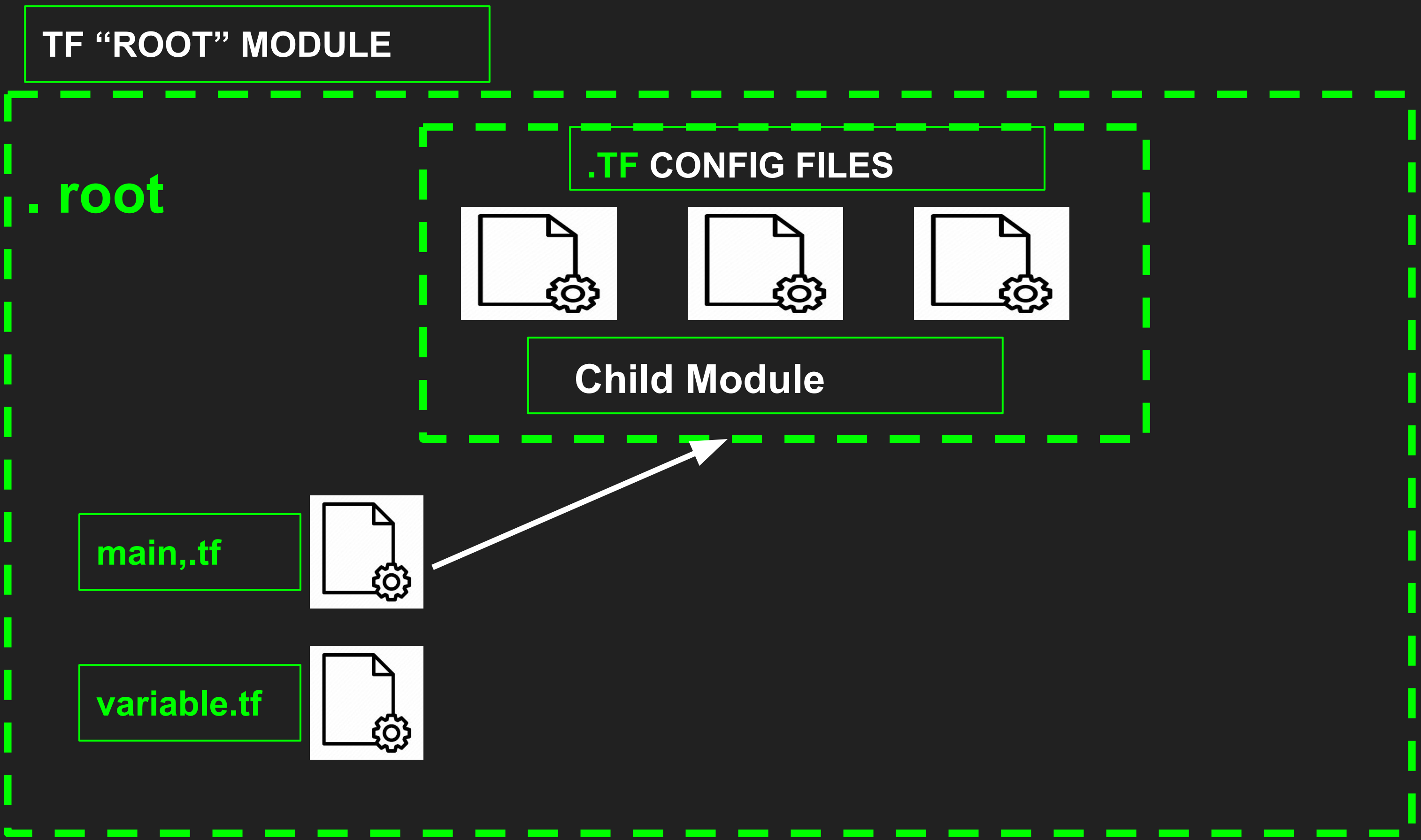
All the terraform code to provision infrastructure resources can be organized in modules within a root module and used as reusable code for the root module configuration .tf files.

They can be called upon by the configuration files in the root module using the module block. Each module block can call upon another module code using the source argument.

Child Module == source module called in a module block in a .tf config file.

```
module "example" {
        source = "path to <child_module>"
}
```

When adding, removing a child module, terraform init has to be re-run.


TF "ROOT" MODULE
. root
.TF CONFIG FILES
Child Module
main,.tf
variable.tf

Using Terraform Modules Input variables and Output Values

Module blocks calling a child modules will use the child module input variable name as arguments

Child module variable name == module block argument name.

IMPORTANT: if value is set literally for a child module resource block argument , it will override any expression value declared in the root module configuration

Module resource attributes can be referenced by another module using module output values with the syntax:

```
module.<module_name>.<module_output_name>
```

When used in a module block the module name is the name of the module block.

The Terraform Public Module Registry

Terraform has created an open source community to promote the collaboration and the sharing of terraform modules for various resources for various infrastructure providers like AWS, Azure, GCP.

Some of these community modules are verified by Terraform and will be flagged as such.

Module in the registry are versioned and versions represent changes are made to the module code.

It is important to select the right version across your team in order not to break the configuration. Please check the Readme under version for dependencies.

The most recent version will be selected if no version number is configured using the version argument

Module versioning with GitHub and git

As part of Terraform immutability, modules in the registry are versioned.

Versions allow the adding of new code to a module which can be tested and validated.

This enable team to safely deploy new code in modules as they can roll back to a known stable version.

Modules can be versioned in VCS like GitHub and Git using git tags:
A module can call upon a child module using the following syntax with the source argument:

"git::git@github.com:repo/address//module_name?ref=<version_number>"

The latest module version number will be downloaded if no version is set.