# Terraform Modules

Terraform

# Agenda

- What is a Module
- How to use

# What is a Module

Terraform

# Terraform Modules

- A module is a container for multiple resources that are used together
- Bring DRY (Don’t Repeat Yourself) approach to Terraform
- Use usual programming languages strategy for code reuse, separation of concerns and modularity
- Source for modules can be local or remote
- Using modules, your main Terraform file is like an orchestrator

# Benefits

- **Reusability**: Write once, use everywhere.
- **Organization**: Keep configurations manageable.
- **Collaboration**: Share infrastructure patterns.

# Module Structure

- **main.tf**: Defines resources
- **variables.tf**: Declares input variables
- **outputs.tf**: Exposes output values

- Folder Structure (recommendation)

```
/modules
  /network
    main.tf
    variables.tf
    outputs.tf
```

# Demo – How to create and use a module

Terraform

# Share modules

Terraform

# Terraform Registry

- Prebuilt, reusable configurations published by Terraform and the community
- Public Terraform Registry: registry.terraform.io
- You may use it directly with source parameter

```
module "eks_cluster" {
  source  = "terraform-aws-modules/eks/aws"
  version = "18.0.0"
  cluster_name = "example-cluster"
}
```

# Terraform Registry

- Most of the modules are defined and maintained by the vendor
- AWS, Azure and GCP have the concept of verified modules
- Using them can simplify a lot when you need to start building something with Terraform
- Another benefit is the version updates that can be done more easily
- On the other side, you may lose the control of the code used

# Private sharing

- You can add share your module using a git module
- Using this approach, you may use private repos
- You need to make sure that you have authentication and authorization on the machine you use the module

# Versioning

- On both strategy you should use versioning on your modules
- Using a registry, you define the version you want to include
- Using a git repo, you can define a branch name, tag or commit SHA

# Best Practices

- Keep modules focused and single-purpose
- Use meaningful variable names and defaults
- Version-control your modules
- Document inputs and outputs thoroughly

# Demo – How to use shared module

Terraform

# Lab 06 – Use Modules

Terraform