**Terraform**

* **Terraform** is the infrastructure as code, offering from HashiCorp.
* It is a tool for building, changing, and managing infrastructure in a safe,repeatable way.
* **IaC** is the process of managing infrastructure.
* Simple and human readable language.
* A **simple workflow** for deployment will follow closely to the steps below.
* **Scope:** Confirm what resources need to be created for a given Project.
* This state is used by Terraform to map real world resources to your configuration, keep track of metadata, and to improve performance for large infrastructures.
* A provider is responsible for understanding. Aws, Google cloud, Azure,docker, kubernetes..
* **Modules** a Terraform module is a set of Terraform configuration files in a single directory.
* **Backends a** "backend" in Terraform determines how state is loaded and how an operation such as <apply> is executed. By default, Terraform uses the "local" backend, which is the normal behavior of Terraform you're used to. Backends are completely optional.
* **Platform Agnostic;** you may have several different clouds and platforms to support your various applications.
* **State Management** Terraform creates **a state file** when a project is first initialized. Terraform uses this local state to create plans and make changes to your infrastructure.
* Aynı folder altındaki Xxxxx**.tf** uzantılı tüm dosyaları çalıştırır.

sudo yum update -y

sudo yum install -y yum-utils

sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo

sudo yum -y install terraform

sudo terraform version

sudo usermod -a -G docker ec2-user

newgrp docker

export PS1='\[\033[01;31m\]\u\[\033[00m\]\[\033[01;33m\]@\h\[\033[00m\]\[\033[01;32m\]\w\[\033[01;36m\]:\$\[\033[00m\]'

export PS1="\[\e[1;34m\]\u\[\e[33m\]@\h# \W:\[\e[32m\]\\$\[\e[m\] "

terraform init

terraform plan (Plan: 1 to add, 0 to change, 0 to destroy)

terraform fmt

terraform apply

terraform graph

terraform destroy

terraform import aws\_security\_group.tf-sg sg-01b92e29e828a2177

terraform import "aws\_instance.tf-instances[1]" i-092fe70d1cef163c1

terraform-modules

   ├── dev

   │   └── dev-vpc.tf

   ├── modules

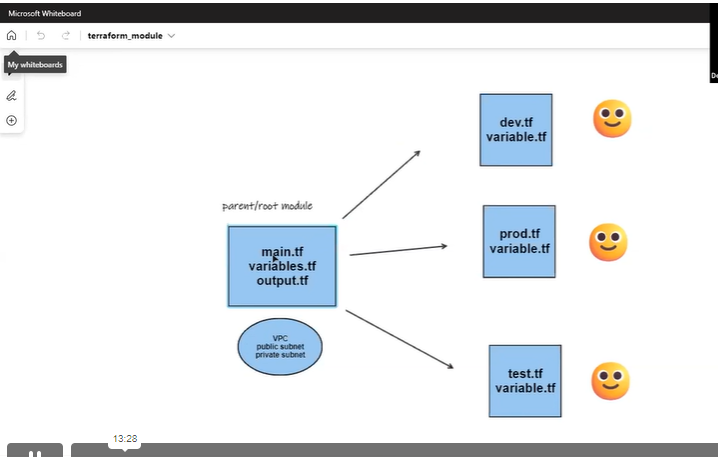
   │   ├── main.tf

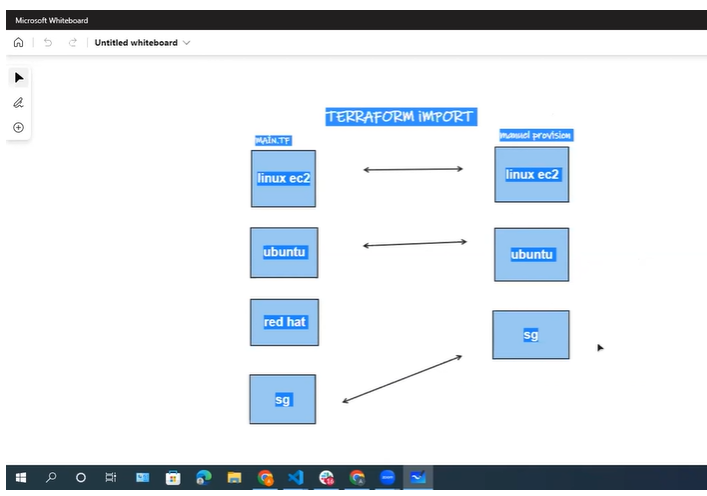
   │   ├── outputs.tf

   │   └── variables.tf

   └── prod

       └── prod-vpc.tf





**Some Terraform Commands**

**terraform init**

The terraform init command is used to initialize a working directory containing Terraform configuration files. Initialization downloads and installs the provider's plugin so that it can later be executed.

**terraform plan**

The terraform plan command is used to create an execution plan. It will not modify things in infrastructure. This command is a convenient way to check whether the execution plan for a set of changes matches your expectations without making any changes to real resources or to the state.

**terraform apply**

The terraform apply command is used to apply the changes required to reach the desired state of the configuration. Terraform apply will also write data to the terraform.tfstate file.

**terraform refresh**

The terraform refresh command is used to reconcile the state Terraform knows about (via its state file) with the real-world infrastructure. This does not modify infrastructure but does modify the state file.

**terraform destroy**

The terraform destroy command is used to destroy the Terraform-managed infrastructure.

**terraform fmt**

The terraform fmt command is used to rewrite Terraform configuration files to a canonical format and style. For use-case, where the all configuration written by team members needs to have a proper style of code, terraform fmt can be used.

**terraform validate**

The terraform validate command validates the configuration files in a directory. Validate runs checks that verify whether a configuration is syntactically valid and thus primarily useful for general verification of reusable modules, including the correctness of attribute names and value types.

**terraform import**

Terraform is able to import existing infrastructure. This allows you to take resources that you've created by some other means and bring them under Terraform management.

**terraform output**

The terraform output command is used to extract the value of an output variable from the state file.

**terraform graph**

The terraform graph command is used to generate a visual representation of either a configuration or execution plan. The output is in the DOT format, which can be used by GraphViz to generate charts.

**terraform show**

The terraform show command is used to provide human-readable output from a state or plan file. This can be used to inspect a plan to ensure that the planned operations are expected, or to inspect the current state as Terraform sees it.

**terraform state list**

The terraform state list command is used to list resources within a Terraform state.

**terraform taint**

The terraform taint command manually marks a Terraform-managed resource as tainted, forcing it to be destroyed and recreated on the next apply.

**terraform console**

The terraform console command provides an interactive console for evaluating expressions.