DATE : 26.02.2024

DT/NT : DT

LESSON : DEVOPS

SUBJECT : TERRAFORM-1

BATCH : B 224

AWS-DEVOPS

TECHPRO EDUCATION

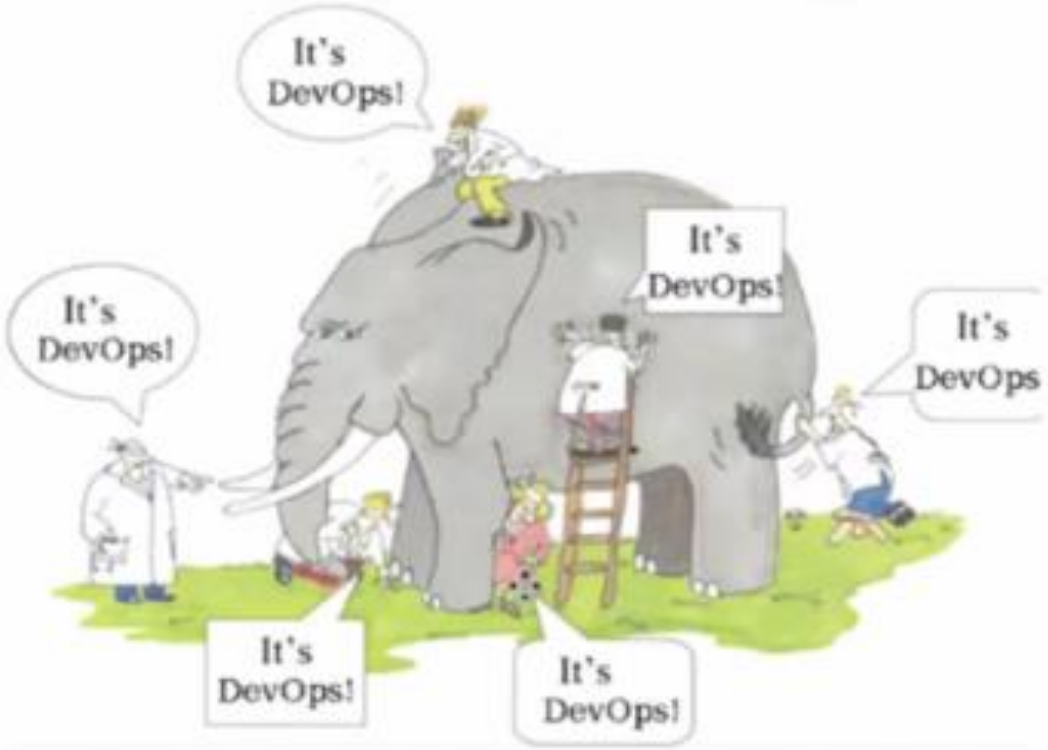
techproeducation.com +1 (585) 304 29 59


HashiCorp
Terraform
code
plan
build
test
release
deploy
operate
monitor
Dev
Ops

What is DevOps

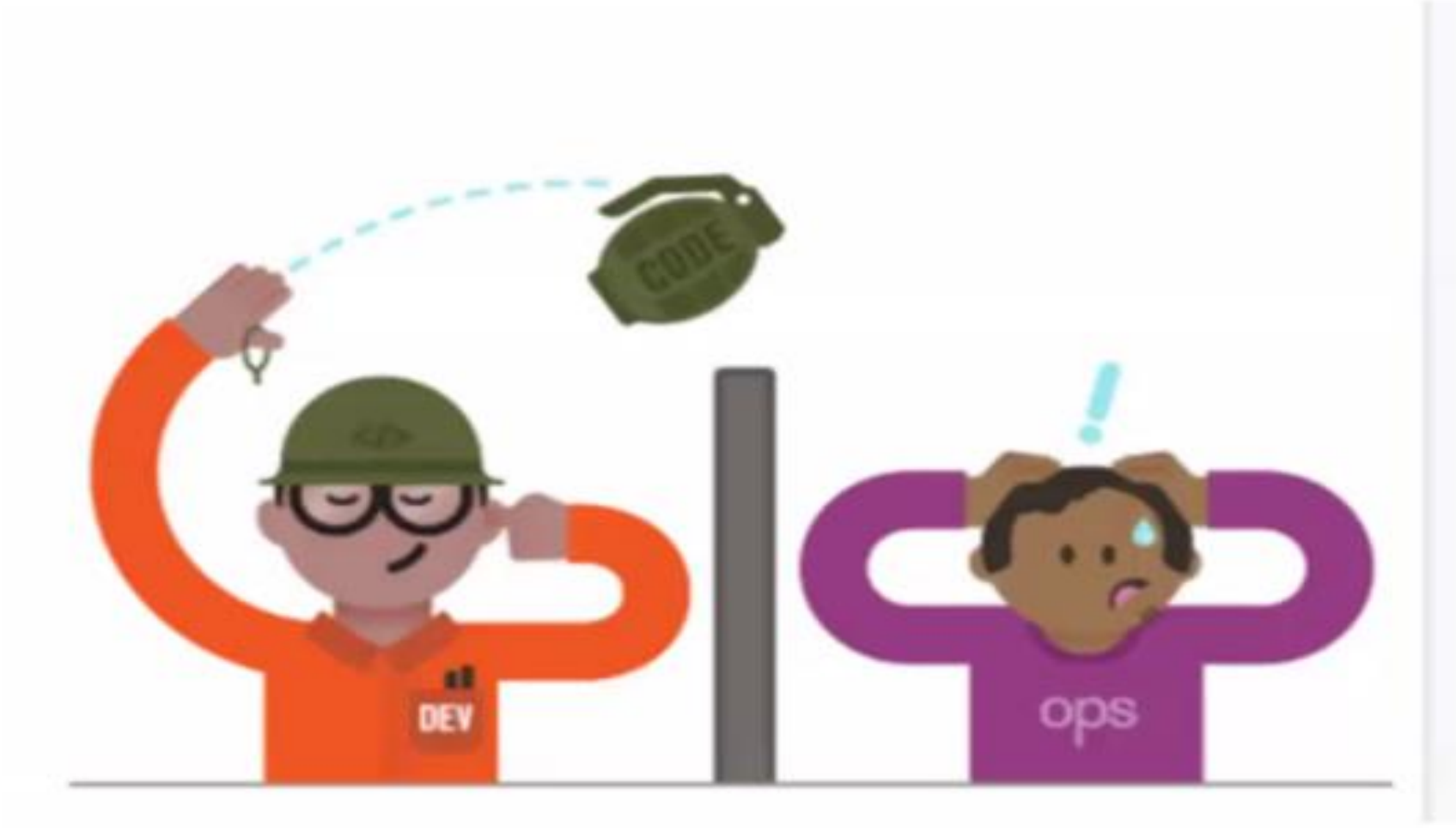
What DevOps is Not...

- a tool
- a role
- a team
- something that can be purchased or simply switched on

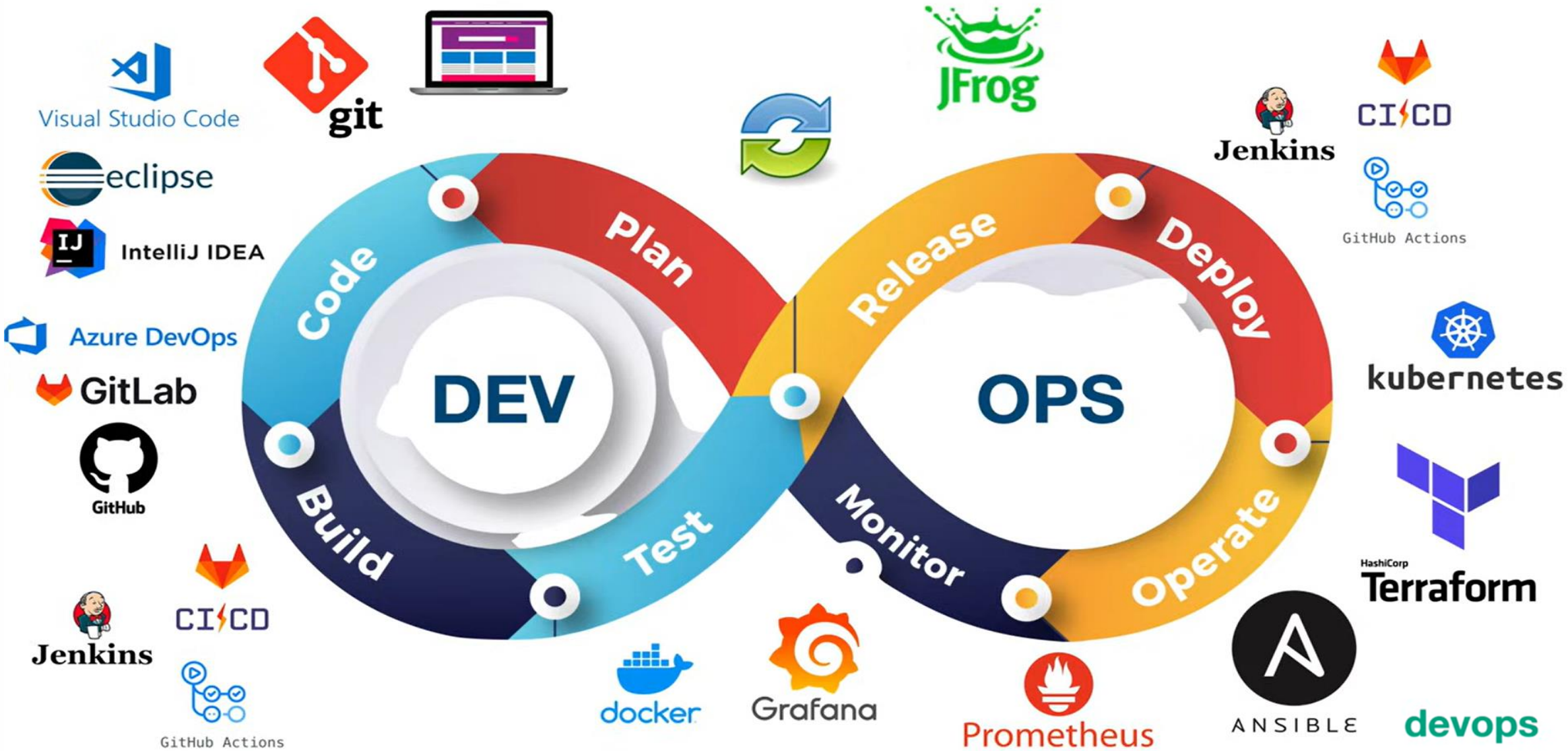


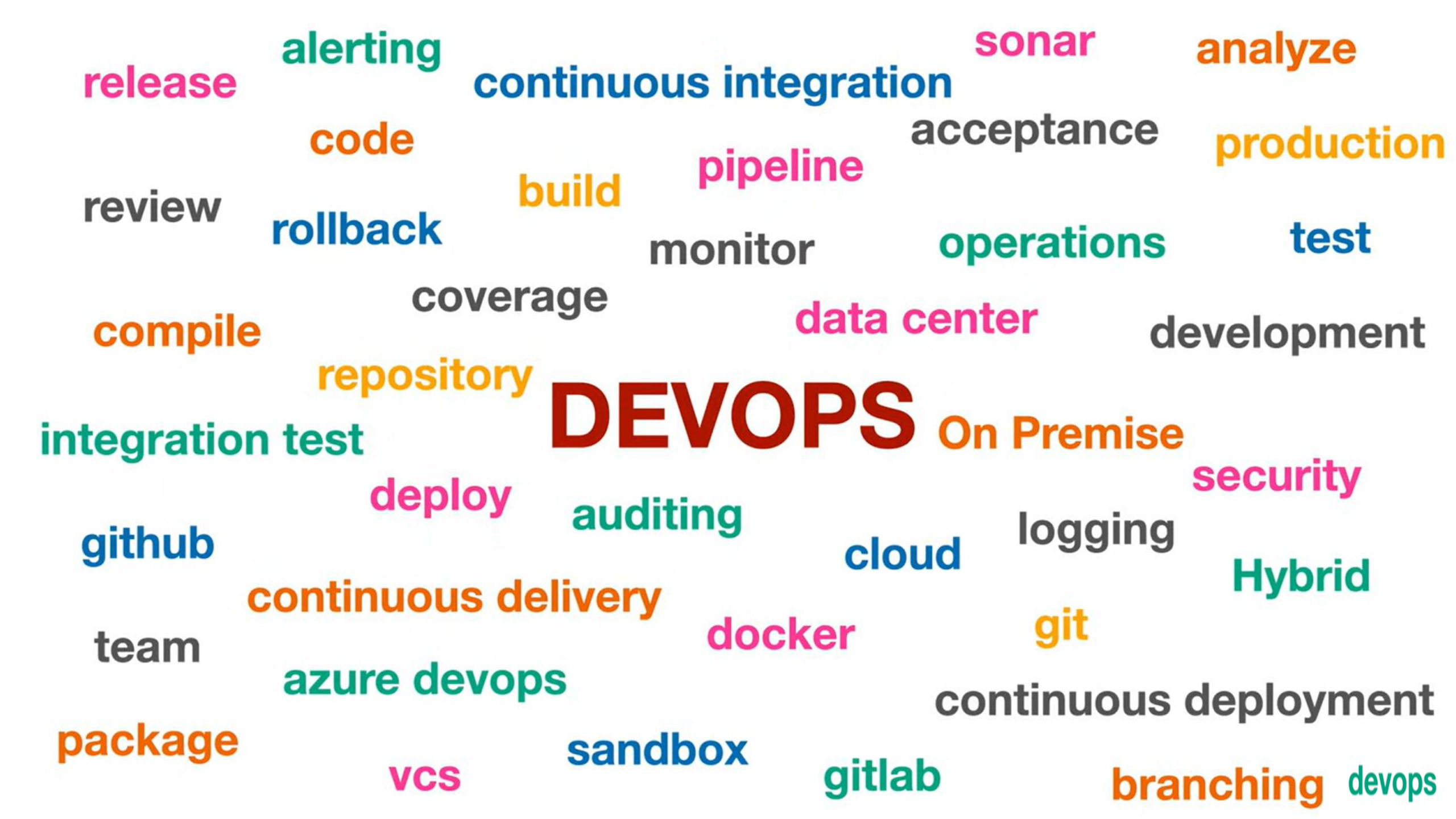
DevOps , Bilgi Teknolojileri departmanı içerisinde bulunan iki temel birimi (Developers and Operations) Geliştiriciler(Yazılım Geliştiriciler, Yazılım Testçileri, vb.), Operasyon (Sistem Mimari ve Altyapı Ekipleri,Güvenlik ve Ağ ekipleri vb.) bir arada etkili bir iletişim içerisinde beraber çalışmalarıdır.DevOps'u aslında bir felsefe, yaklaşım veya bakış açısı olarak değerlendirebiliriz.Yazılım geliştiricilerin alışık olduğu Scrum, Agile, Kanban ve diğer yöntemler gibidir.

Yazılımcı güncelleme getiriyor test etmem gerekiyor diyor, Operations (eski zamanda) server ları yürütmeye göre ayarladık yeni server lazım olacak iş çıkarıyorsunuz diyor.



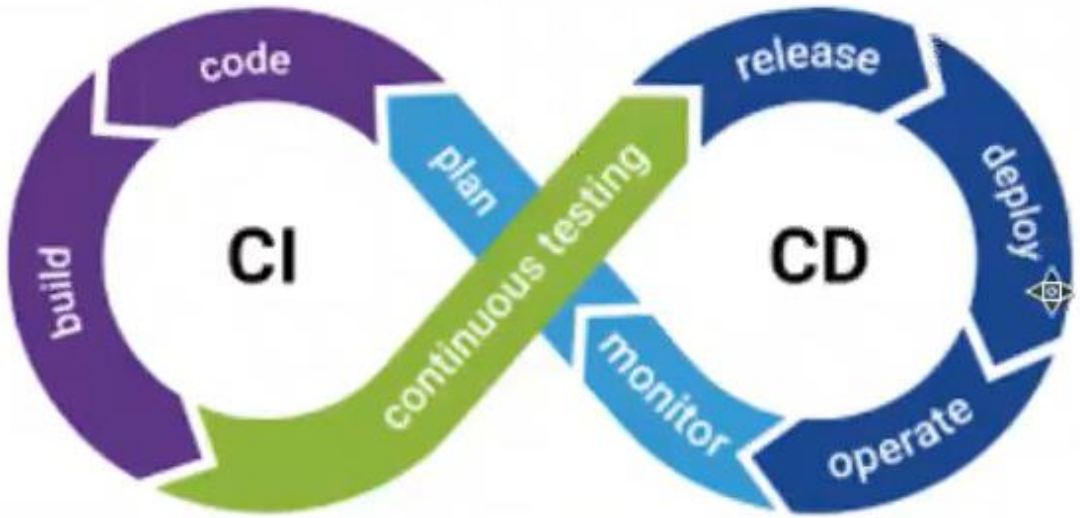
Ya da developer kodu yazar, çok güzel bir güncellemedir Operations a gönderir, ancak güncelleme müthiş şekilde RAM ve CPU yiyor. Bu da Operations içinde ciddi bir sorundur.


Visual Studio Code
git
JFrog
Jenkins
CI/CD
eclipse
GitHub Actions
IntelliJ IDEA
Azure DevOps
kubernetes
GitLab
GitHub
HashiCorp
Terraform
Jenkins
CI/CD
GitHub Actions
docker
Grafana
Prometheus
ANSIBLE
devops
Code
Plan
Build
Test
Release
Deploy
Monitor
Operate
DEV
OPS


alerting
sonar
analyze
release
continuous integration
acceptance
code
production
pipeline
review
build
rollback
monitor
operations
test
coverage
compile
data center
development
repository
DEVOPS
integration test
On Premise
security
deploy
auditing
logging
github
cloud
Hybrid
continuous delivery
docker
git
team
azure devops
continuous deployment
package
sandbox
vcs
gitlab
branching
devops

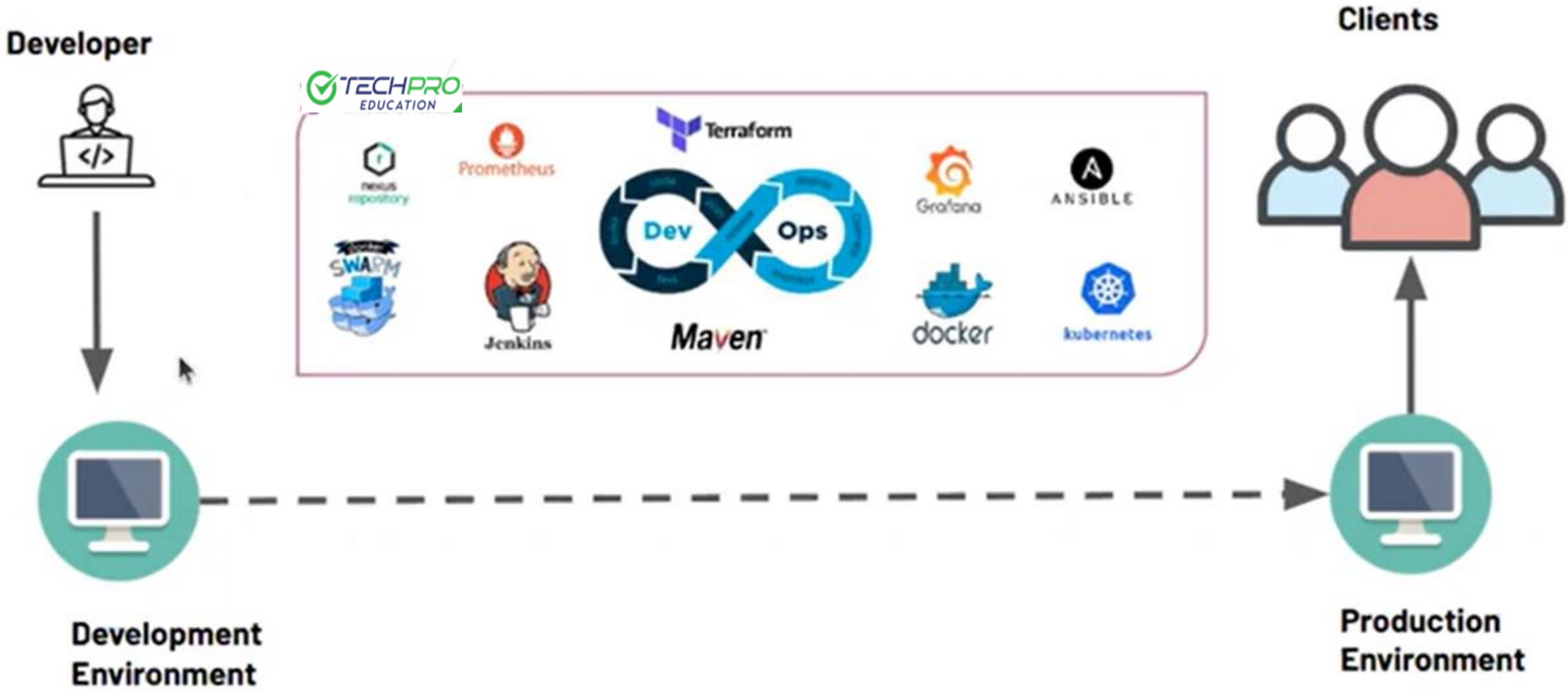
Introduction to DevOps

What is CI/CD?



In software engineering, **CI/CD** is the combined practices of continuous integration **(CI)** and (more often) continuous delivery or (less often) continuous deployment (**CD**).

Introduction to DevOps



DevOps

DevOps uygulamaları genellikle çeşitli araçları içerir. Bu araçlar, kod depolama, sürüm kontrolü, yapı (build) otomatizasyonu, konfigürasyon yönetimi, izleme ve loglama gibi farklı amaçlar için kullanılır. Örneğin:

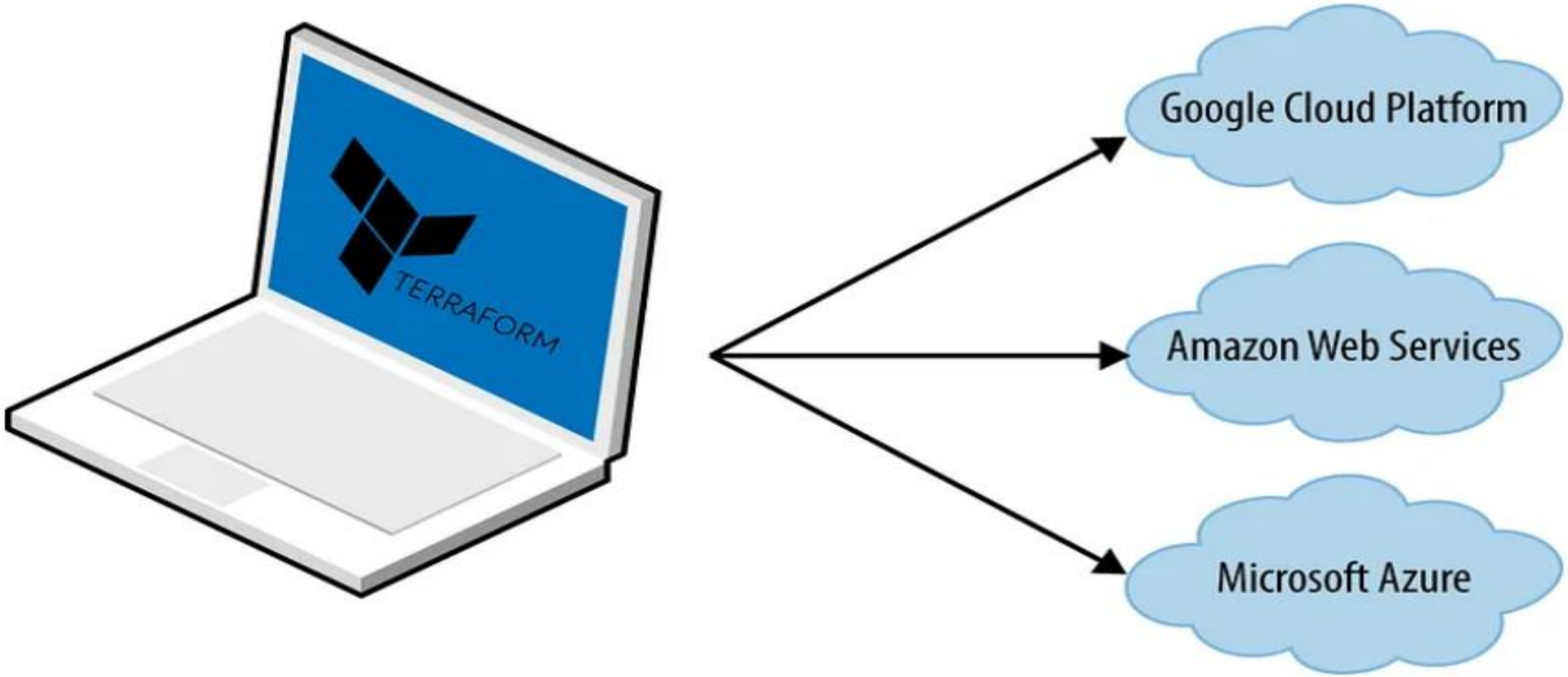
- **Jenkins** - Sürekli entegrasyon ve sürekli dağıtım (CI/CD).
- **Docker** - Konteynerizasyon.
- **Kubernetes** - Konteyner orkestrasyonu.
- **Ansible/Chef/Puppet** - Konfigürasyon yönetimi.
- **Git** - Sürüm kontrolü.
- **Prometheus/Grafana** - İzleme ve metrik.
- **ELK Stack/Elastic Stack** - Log yönetimi.

Terraform nedir ?

HashiCorp
Terraform

Terraform HashiCorp firması tarafından, güvenli ve verimli bir şekilde altyapı oluşturmak, değiştirmek ve iyileştirmek için geliştirilen **Open Source** bir tooldur.

Terraform


TERRAFORM
Google Cloud Platform
Amazon Web Services
Microsoft Azure

Terraform

- **AWS :** *CloudFormation*
- **Azure :** *Resource Manager*
- **Google Cloud :** *Deployment Manager vb.*

Terraform

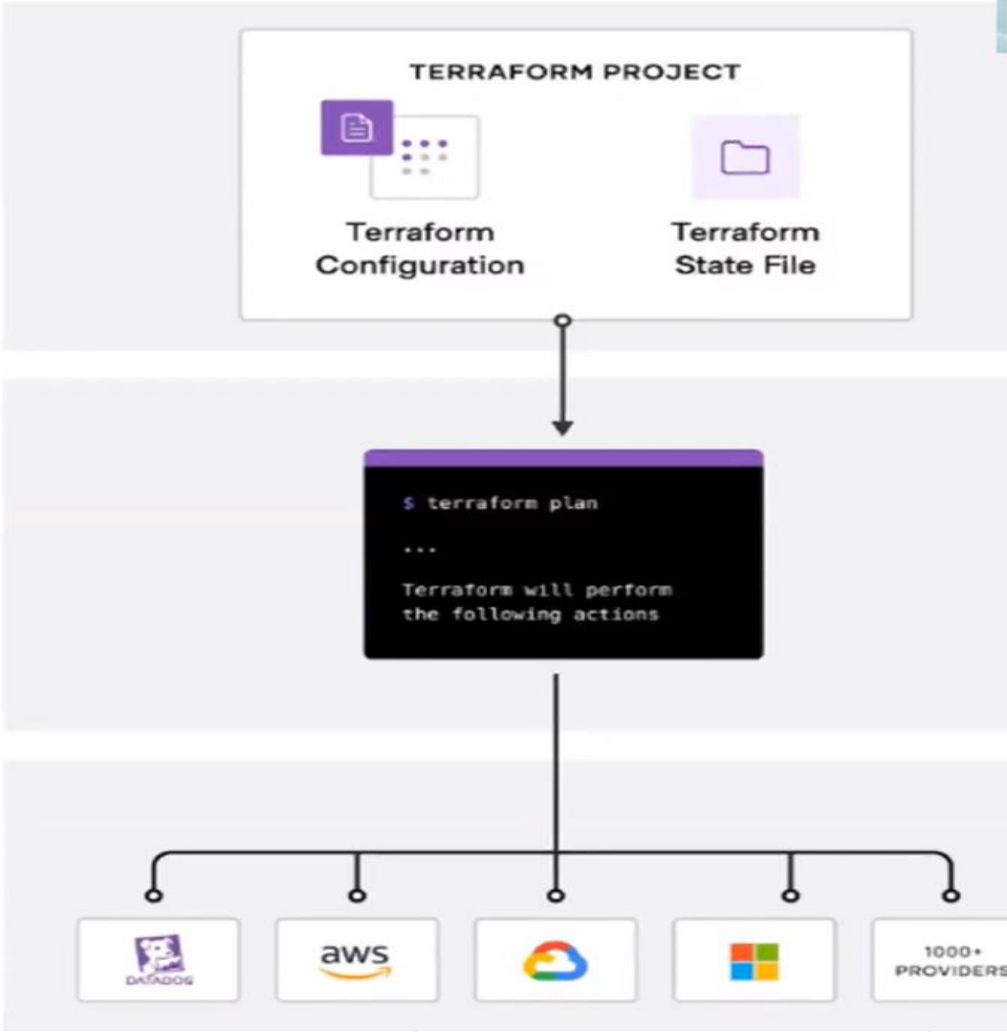
Workflows

Write: Define infrastructure in configuration files

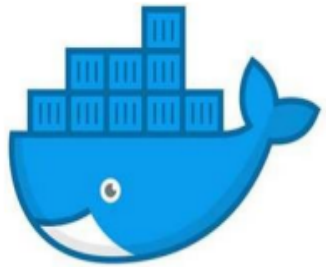
Initialize: Download the correct provider plug-ins for the project.

Plan: Review the changes Terraform will make to your infrastructure

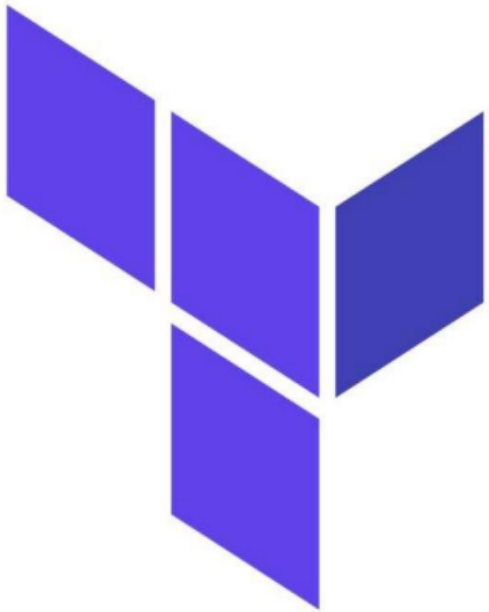
Apply: Terraform provisions your infrastructure and updates the state file



- Terraform'u
  - VPC(Amazon Virtual Private Cloud) oluştururken,
  - AWS kullanıcısı oluştururken ve kullanıcıyla bağlantılı izinleri yönetirken
  - Sunucuları spin ederken(Spinning Up the Servers)
  - ve Docker I yüklerken kullanırız.
  - Terraform la ayrıca yukarıdaki görevlerin doğru bir sırada yapıldığına emin oluruz.



Terraform



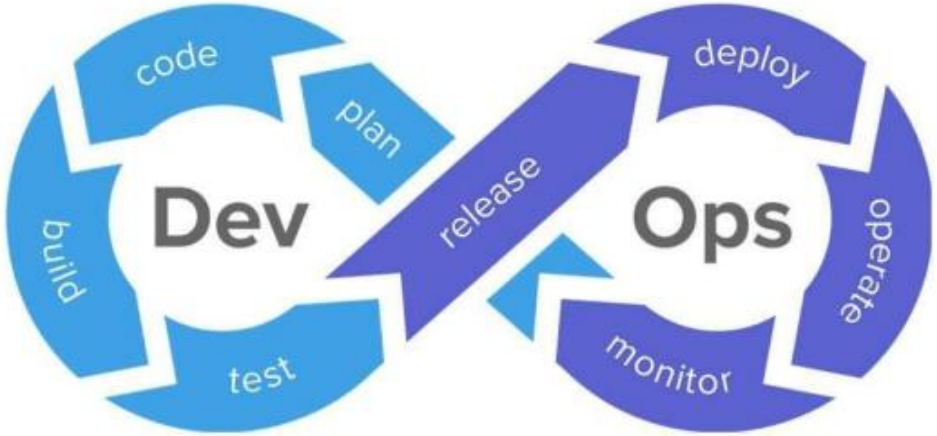
* Terraform'a giriş
* Providers
* Resources & Data Sources
* Terraform State
* Output
* Variables
* Enviroment Variables
* Modules

Terraform

- - Infastructure ını
- - Platform unu
- - Platformunda çalıştırdığın servisleri(AWS, Jenkins gibi) yönetmene izin verir.
- Açık kaynak kodludur.
- “Declaretive” dir. Yani her bir adımı teker teker manuel olarak kodlamazsın da ideal durumu verirsin. Terraform varolan kodu ideal duruma uyarlar.

Terraform

- Provisioning infostructure(DevOps)



- Application Deploying(Developer Takımı)

Terraform Elements

Providers

A provider is responsible for understanding API interactions and exposing resources. Every Terraform provider has its own documentation, describing its resource types and their arguments.

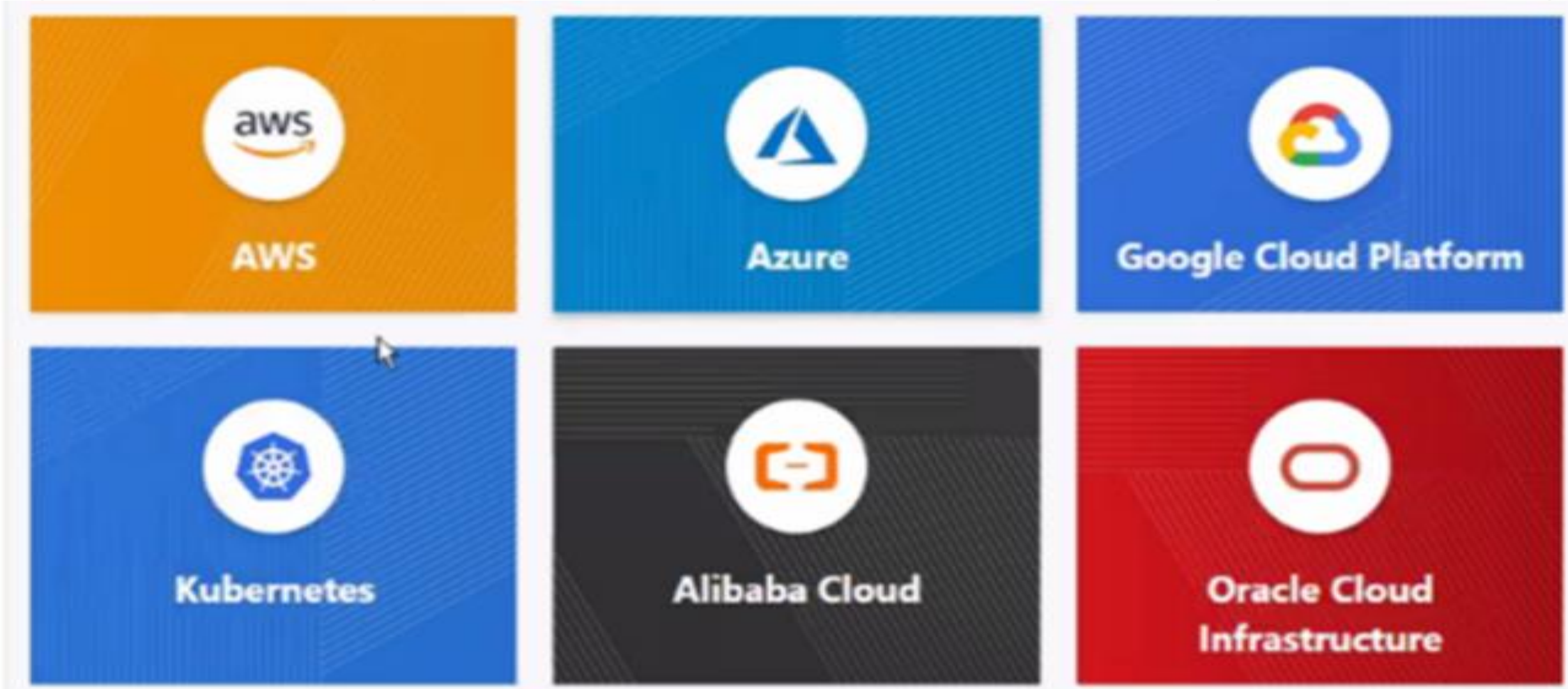
Terraform

Bu kod ile kullanacağımız platform (burada aws) un versionunu belirtmiş oluyoruz. Provider ne demiştik, Terraform un platformlar ın API ile iletişime geçen bileşenidir.

terraform-lesson > terraform-aws > main.tf > ...

```
1   terraform {
2     required_providers {
3       aws = {
4         source = "hashicorp/aws"
5         version = "4.57.1"
6       }
7     }
8   }
9
10  provider "aws" {
11    # Configuration options
12  }
13
14
```

Terraform



Terraform

Terraform Elements

Modules

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. So in this sense, every Terraform configuration is part of a module.

```
$ tree minimal-module/
.
├── LICENSE
├── README.md
├── main.tf
├── variables.tf
├── outputs.tf
```

Terraform

Terraform Elements

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. You can successfully use Terraform without ever having to learn or use backends. Backends are used for keeping sensitive information off disk.

Terraform

Advantages of Terraform

Platform Agnostic

- In a modern datacenter, you may have several different clouds and platforms to support your various applications.
- With Terraform, you can manage a **heterogeneous environment** with the same workflow by creating a configuration file to fit the needs of your project or organization.

Documentation

Learn More Terraform

- Terraform Documentation
- Hashicorp/terraform (Github Page)
- Shuaibiyy/awesome-terraform
- tfutils/tfenv
- gruntwork-io/terragrunt
- 28mm/blast-Radius
- Terraform Registry

Terraform & Ansible

- **FARKLAR**

Terraform

- - Genellikle Infrastructure provisioning tool(Altyapı sağlama aracı) olarak kullanılır.
- - Görece daha yenidir.
- - Orchestration yeteneği daha gelişmiştir. Orchestration, bilgisayar sistemlerinin, uygulamaların ve hizmetlerin otomatik yapılandırması, yönetimi ve koordinasyonudur. IT departmanının karmaşık görevleri ve iş akışlarını daha kolay yönetmesine yardımcı olur.
-

Ansible

- - Genellikle configuration tool olarak kullanılır. Yani önce infrastructure u oluşturursun. Sonra onu configure etmek için Ansible I kullanırsın.
- - Terraforma göre daha eskidir.

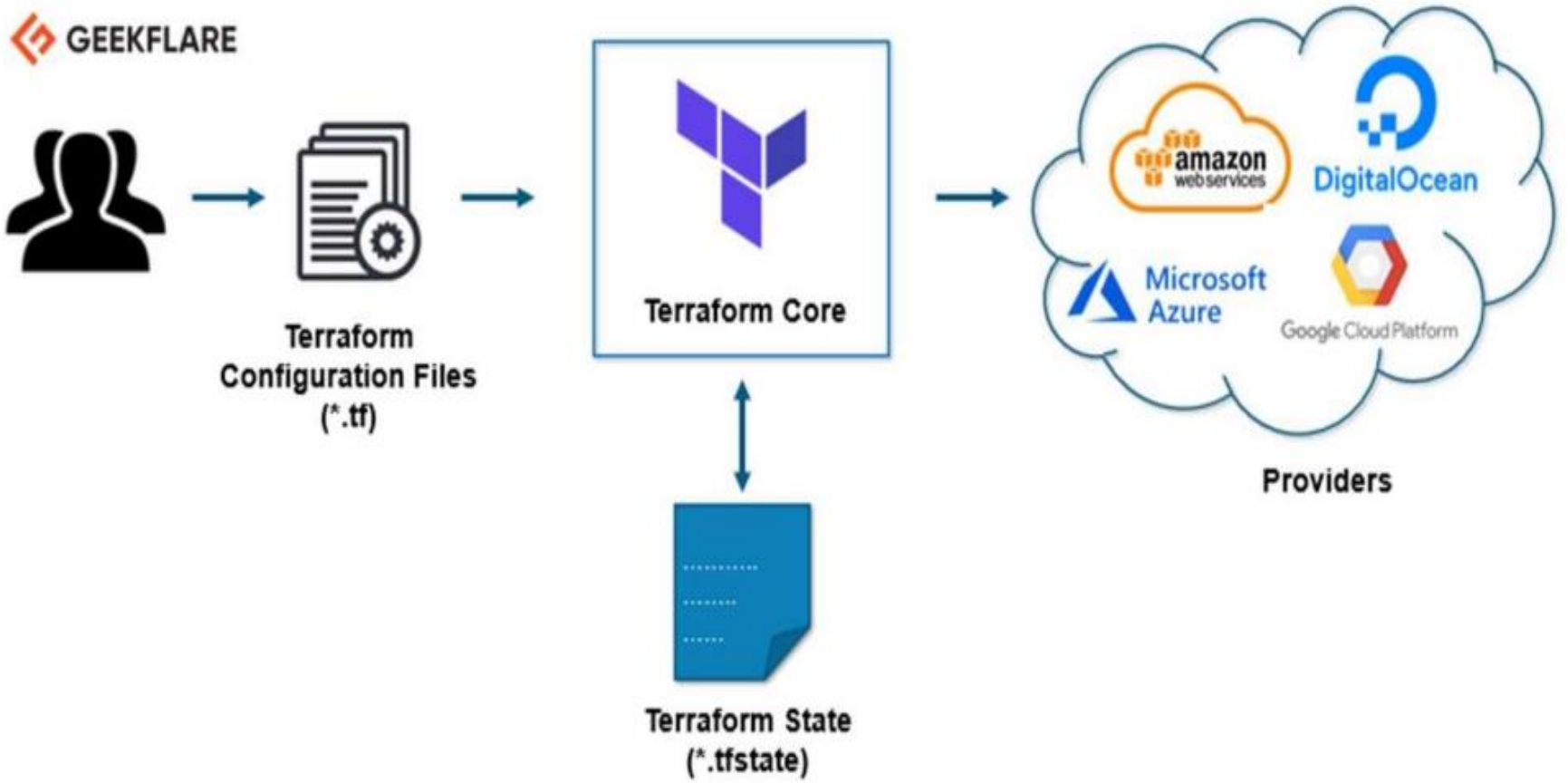
Terraform & Ansible

BENZERLİKLER

- İkisi de IAC tool'u olarak kullanılır. Yani ikisiyle de altyapıyı sağlarız, yapılandırırırz ve yönetiriz.

Terraform

- 1) Core
- 2) State



main.tf

```hcl
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "4.40.0"
    }

    github = {
      source = "integrations/github"
      version = "5.9.1"
    }
  }
}

provider "aws" {
  # Configuration options
  region = "us-east-1"
  # profile = "profile_name"
}

provider "github" {
    token = 'github_repository.myrepo.name
```

main.tf

```hcl
resource "github_repository" "myrepo" {
    name = "bookstore-repo"
    auto_init = true
    visibility = "private"
}

resource "github_branch_default" "main" {
    branch = "main"
    repository = github_repository.myrepo.name
}

variable "files" {
    default = ["bookstore-api.py", "Dockerfile", "docker-compose.yml", "requirements.txt"]

}

resource "github_repository_file" "myfiles" {
    for_each = toset(var.files)
    content = file(each.value)
    file = each.value
    repository = github_repository.myrepo.name
    branch = "main"
    commit_message = "managed by terraform"
    overwrite_on_create = true
}
```

HashiCorp Terraform

amazon web services
Google Cloud Platform
Azure
kubernetes
Nomad
openstack
PostgreSQL
Bitbucket
docker
CONSUL
GitHub
New Relic.

TERRAFORM