**Terraform**

### **What is Terraform?**

**Terraform is a tool for infrastructure provisioning to build out infrastructure through code, often called Infrastructure as Code.**

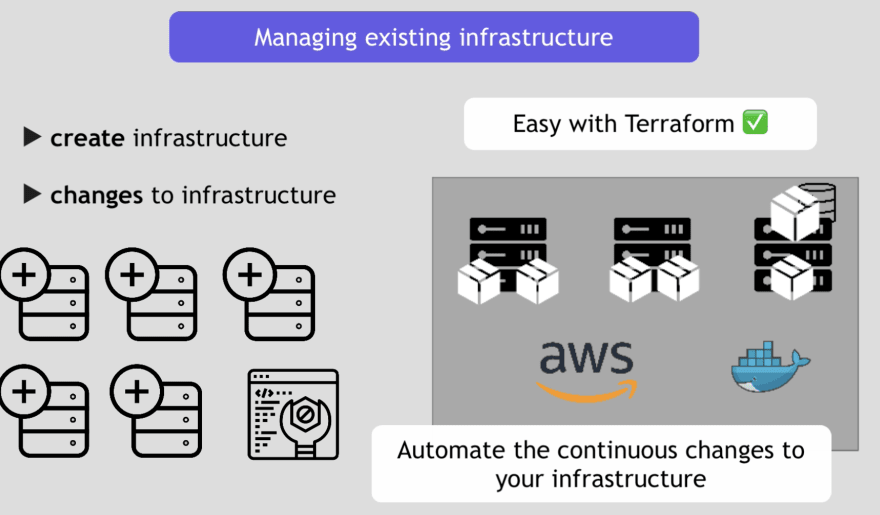
**So, Terraform allows you to automate and manage your infrastructure, your platform, and the services that run on that platform.**

**It's open source and declarative, which means you define WHAT you want (the desired "end state") rather than describing exactly each step or HOW to do it.**

**Terraform is an open-source Infrastructure as Code (IaC) tool developed by HashiCorp. It allows users to define and provision infrastructure using a high-level configuration language called HashiCorp Configuration Language (HCL) or JSON. With Terraform, you can automate the management of infrastructure across multiple cloud providers and on-premises environments.**

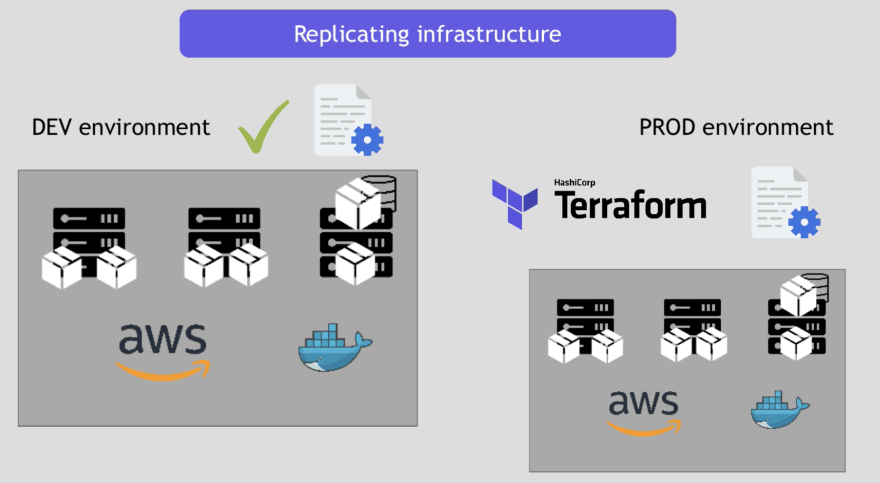
### **What is Terraform used for? Use Cases**

**On the one hand, Terraform is used for creating or provisioning new infrastructure and for managing existing infrastructure:**

****

**What is Terraform used for?**

**On the other hand, it can be used to replicate infrastructure. E.g. when you want to replicate the development setup also for staging or production environment:**

****

### **How Terraform works? Terraforms Architecture**

**Terraform has 2 main components:**

* **CORE**

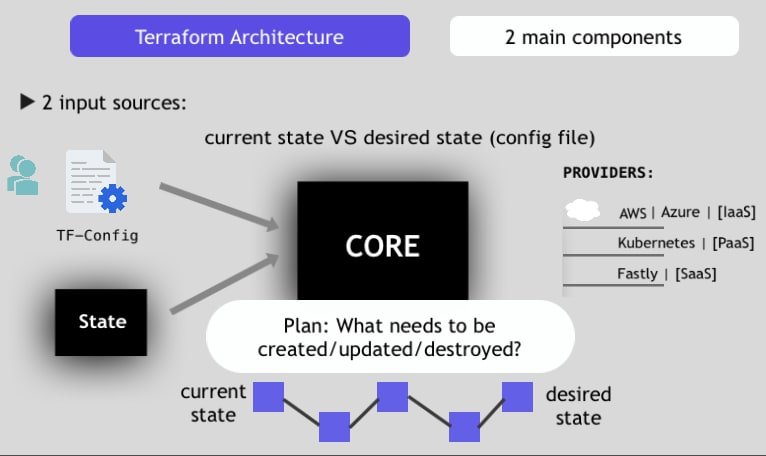
**Terraform's Core takes two input sources, which are your configuration files (your desired state) and second the current state (which is managed by Terraform).**

**With this information, the Core then creates a plan of what resources need to be created/changed/removed.**

* **Provider**

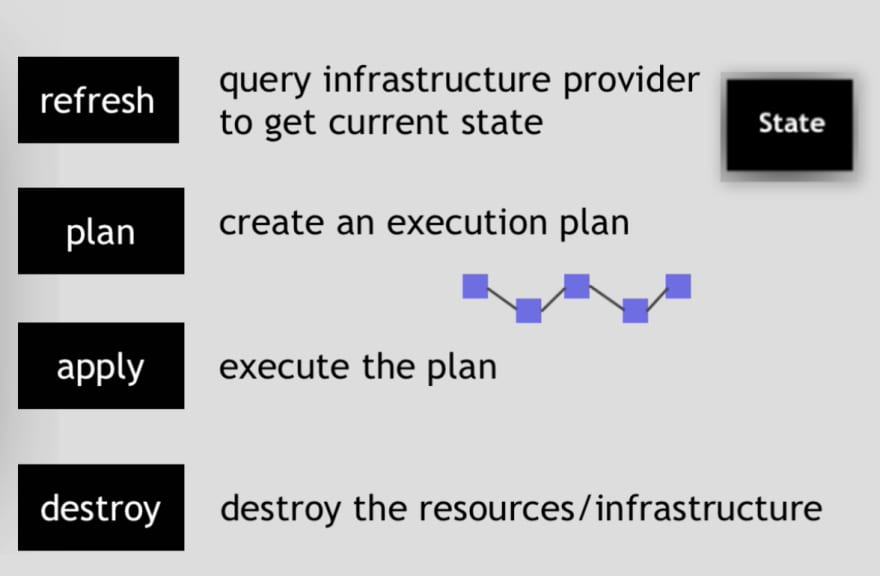
**The second part of the Architecture is providers. Providers can be IaaS (like AWS, GCP, Azure), PaaS (like Heroku, Kubernetes), or SaaS services (like Cloudflare).**

**Providers expose resources, which makes it possible to create infrastructure across all these platforms.**

****

### **Terraform Basic Commands**

**Terraform has commands to go through different stages:**

****

### **Key Features of Terraform:**

1. **Multi-Cloud Support:**
   * **Terraform supports various cloud platforms like AWS, Azure, Google Cloud, and even on-premises infrastructure like VMware or OpenStack.**
   * **This allows you to manage infrastructure across different providers using a consistent approach.**
2. **Declarative Language:**
   * **Terraform uses a declarative approach, meaning you define the desired state of your infrastructure, and Terraform automatically figures out the steps to create, update, or delete resources to match that state.**

**Example:  
hcl  
Copy code  
resource "aws\_instance" "example" {**

**ami = "ami-12345678"**

**instance\_type = "t2.micro"**

**}**

* + **In this example, Terraform provisions an AWS EC2 instance with the specified attributes.**

1. **Execution Plan (Plan & Apply):**
   * **Terraform generates an execution plan that shows what actions will be taken to achieve the desired state. This helps you review changes before applying them.**
   * **terraform plan generates the execution plan, and terraform apply executes the changes.**
2. **State Management:**
   * **Terraform keeps track of your infrastructure using a state file. This file records the current state of your resources, allowing Terraform to manage changes effectively.**
   * **The state can be stored locally or in a remote backend (like S3) for collaboration.**
3. **Modules:**
   * **Terraform supports reusable components called modules, which enable you to define and reuse pieces of infrastructure across projects. This promotes modular and scalable architecture.**
4. **Immutability:**
   * **Terraform promotes the idea of immutable infrastructure, where resources are replaced rather than updated in place. This helps prevent configuration drift and ensures a consistent environment.**
5. **Providers:**
   * **Terraform uses providers to interact with APIs of cloud platforms and services (e.g., AWS, Azure, GCP). Each provider offers resources that Terraform can manage.**

### **Advantages of Terraform:**

* **Infrastructure automation: Simplifies infrastructure provisioning and management.**
* **Version control: Infrastructure code can be versioned, allowing for better collaboration and change tracking.**
* **Scalability: Easily manage and scale infrastructure across multiple environments.**
* **Consistency: Ensures the same configuration is applied in different environments (e.g., Dev, Test, Prod).**