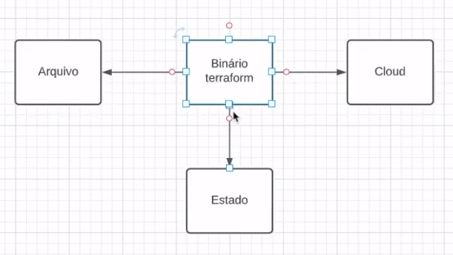
**Terraform – Dia 1 – 25/08/2025**

**1 – Introdução**



**Infraestrutura Mutável**

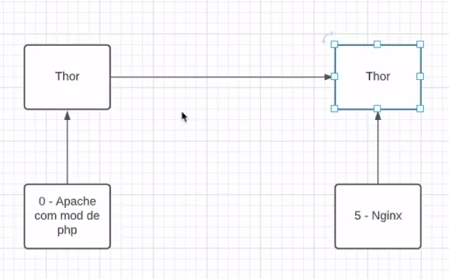
* **Definição:** É a infraestrutura que pode ser alterada após criada.
* Você cria um servidor (ex: EC2 na AWS), entra nele via SSH e instala/atualiza pacotes manualmente ou via scripts.
* Cada alteração vai modificando aquele mesmo servidor ao longo do tempo.

**👉 Características:**

* Atualizações incrementais (patches, upgrades).
* Cada servidor pode acabar ficando diferente dos outros (“**snowflake servers**”).
* Risco maior de inconsistência e erros manuais.

**Exemplo:**

* Criar uma VM e depois rodar apt install nginx nela.
* Usar Ansible para aplicar mudanças em servidores que já estão rodando.



**Infraestrutura Imutável**

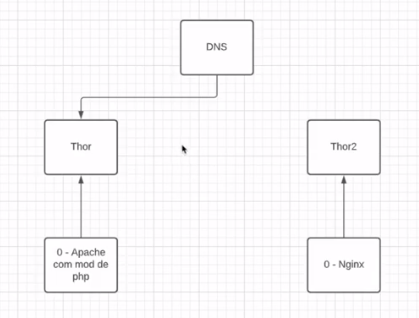
* **Definição:** É a infraestrutura que não sofre mudanças depois de criada.
* Se precisar alterar algo (novo pacote, config), você gera uma nova imagem/servidor e substitui o antigo.
* O servidor é tratado como descartável: “cattle, not pets”.

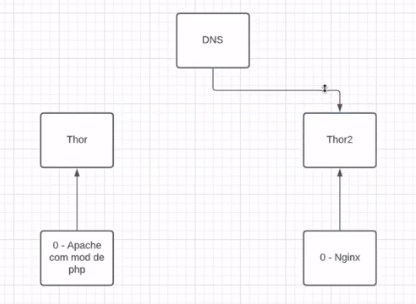
**👉 Características:**

* Cada alteração gera uma nova versão da máquina/imagem/container.
* Fácil de garantir consistência (todas as instâncias são iguais).
* Reduz risco de “**drift**” (diferença entre servidores).

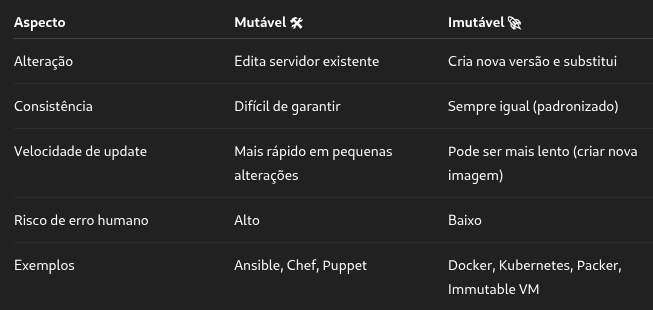
**Exemplo:**

* Criar uma imagem no Packer ou um container Docker com a aplicação já configurada.
* Subir essa imagem em produção. Se precisar mudar, gera uma nova versão da imagem e substitui.





**🔹 Comparação rápida**



**🔹 Resumindo**

* **Mutável:** você altera o servidor existente → mais flexível, mas mais frágil.
* **Imutável:** você substitui por um novo servidor → mais confiável, escalável e alinhado com práticas modernas (cloud, containers).

**🔹 O que é Terraform?**

O **Terraform** é uma ferramenta de Infraestrutura como Código (IaC) criada pela HashiCorp.

Ele serve para **provisionar e gerenciar infraestrutura** (servidores, redes, bancos de dados, balanceadores, storage, etc.) em provedores de nuvem (AWS, Azure, GCP, Oracle, VMware, etc.) de **forma automática e padronizada**.

**🔹 Como funciona?**

* Você escreve **arquivos de configuração** em uma linguagem declarativa chamada **HCL** (HashiCorp Configuration Language).
* Esses arquivos descrevem o que você **quer** (ex: “quero 2 VMs, 1 banco de dados e 1 balanceador”).
* O Terraform conversa com a **API do provedor de nuvem** e cria/atualiza/remove os recursos.
* Ele mantém um state file (terraform.tfstate) para saber como está a infraestrutura atual.

**🔹 Conceitos principais**

* **Providers →** Plugins que permitem ao Terraform interagir com um serviço (AWS, Azure, Kubernetes, Docker).

**Exemplo:**

provider "aws" {

region = "us-east-1"

}

* **Resources →** São os objetos que você cria (VM, rede, bucket S3, banco).

resource "aws\_s3\_bucket" "meu\_bucket" {

bucket = "meu-bucket-terraform"

acl = "private"

}

* **Variables →** Parametrizam valores (ex: tamanho da VM, região).

variable "region" {

default = "us-east-1"

}

* **State →** Arquivo que guarda o estado atual da infraestrutura (terraform.tfstate).
* **Modules →** Forma de organizar código reutilizável (ex: módulo pronto para criar VPC, outro para criar cluster Kubernetes).

**🔹 Ciclo de uso do Terraform**

* Escrever o código da infraestrutura (.tf).
* Rodar terraform init → baixar os plugins necessários.
* Rodar terraform plan → mostra o que vai ser criado/modificado.
* Rodar terraform apply → aplica as mudanças.
* Rodar terraform destroy → remove tudo (quando quiser apagar a infra).

**🔹 Exemplo simples (AWS EC2)**

Um código mínimo para criar uma máquina na AWS:

provider "aws" {

region = "us-east-1"

}

resource "aws\_instance" "exemplo" {

ami = "ami-08c40ec9ead489470" # Ubuntu na região us-east-1

instance\_type = "t2.micro"

tags = {

Name = "MeuServidor"

}

}

**Rodando:**

terraform init

terraform plan

terraform apply

👉 O Terraform cria a instância automaticamente na AWS.

**🔹 Diferença Terraform x Ansible**

* **Terraform:** cria a infraestrutura (máquinas, redes, storage, etc.).
* **Ansible:** configura o que está dentro das máquinas (pacotes, usuários, serviços).

➡️ Muitas empresas usam Terraform + Ansible juntos:

Terraform cria os servidores → Ansible configura o sistema.

**⚡ Resumindo:**

Terraform é como um "engenheiro de obras digitais": você descreve a planta (código), e ele constrói e mantém toda a infraestrutura em qualquer nuvem.

**2 – Cloud e S3**

**☁️ O que é Cloud (Computação em Nuvem)?**

**Cloud Computing** (ou computação em nuvem) é o modelo de usar **recursos de TI sob demanda** (servidores, bancos de dados, redes, armazenamento, etc.) através da internet, em vez de manter tudo em um datacenter próprio.

👉 Em vez de comprar e manter hardware, você aluga recursos de provedores como AWS, Azure, Google Cloud, Oracle Cloud etc.

**Principais modelos:**

* **IaaS (Infrastructure as a Service) →** infraestrutura pronta (máquinas virtuais, redes, storage).
* **PaaS (Platform as a Service) →** plataforma para rodar apps sem gerenciar servidor (ex: Heroku, AWS Elastic Beanstalk).
* **SaaS (Software as a Service) →** software pronto via internet (ex: Gmail, Office 365, Dropbox).

**Vantagens:**

✅ Escalabilidade (cresce ou diminui recursos conforme necessidade).

✅ Paga só pelo uso (pay-as-you-go).

✅ Alta disponibilidade (datacenters distribuídos).

✅ Menos custo com hardware e manutenção.

**📦 O que é S3 (Simple Storage Service)?**

O Amazon S3 é um serviço da AWS (Amazon Web Services) para armazenamento de objetos na nuvem.

👉 Pense nele como um HD infinito na internet, onde você pode guardar e recuperar arquivos de qualquer tamanho, a qualquer hora, de qualquer lugar.

**Conceitos principais:**

* **Buckets →** pastas principais (ex: meu-bucket-arquivos).
* **Objects →** os arquivos que você coloca dentro dos buckets (ex: foto.jpg, backup.zip).
* **Keys →** o caminho/identificador do objeto dentro do bucket.
* **Region →** local físico onde os dados ficam armazenados (ex: us-east-1, sa-east-1 em SP).

**Casos de uso:**

* Armazenar backups e logs.
* Hospedar sites estáticos.
* Guardar imagens, vídeos, PDFs, dados para Big Data/IA.
* Base de dados para Data Lakes.

**Exemplo (com AWS CLI):**

# Criar um bucket

aws s3 mb s3://meu-bucket-exemplo

# Enviar um arquivo

aws s3 cp arquivo.txt s3://meu-bucket-exemplo/

# Listar arquivos

aws s3 ls s3://meu-bucket-exemplo/

**🔹 Resumindo**

* **Cloud =** uso de recursos de TI pela internet (em vez de datacenter próprio).
* **S3 =** serviço da AWS para armazenar objetos (arquivos) de forma simples, barata e escalável.

**Dia 2 – 25/08/2025**