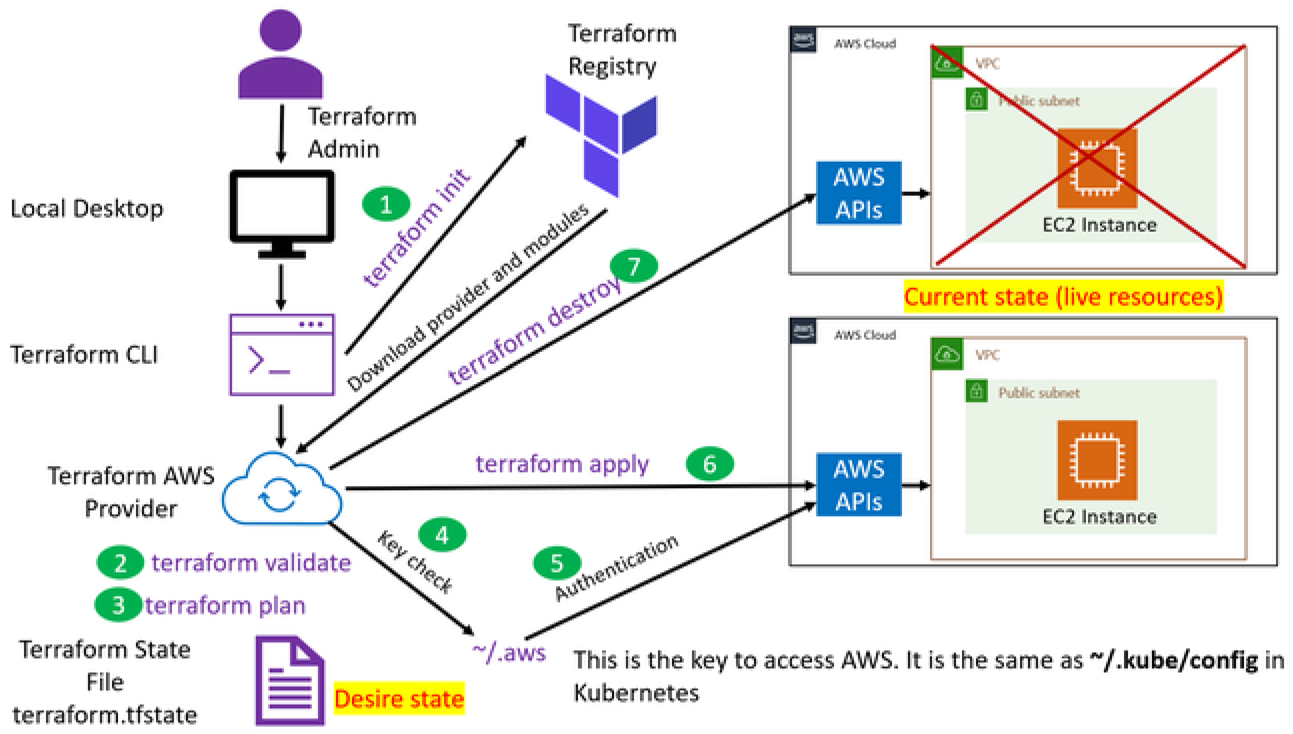
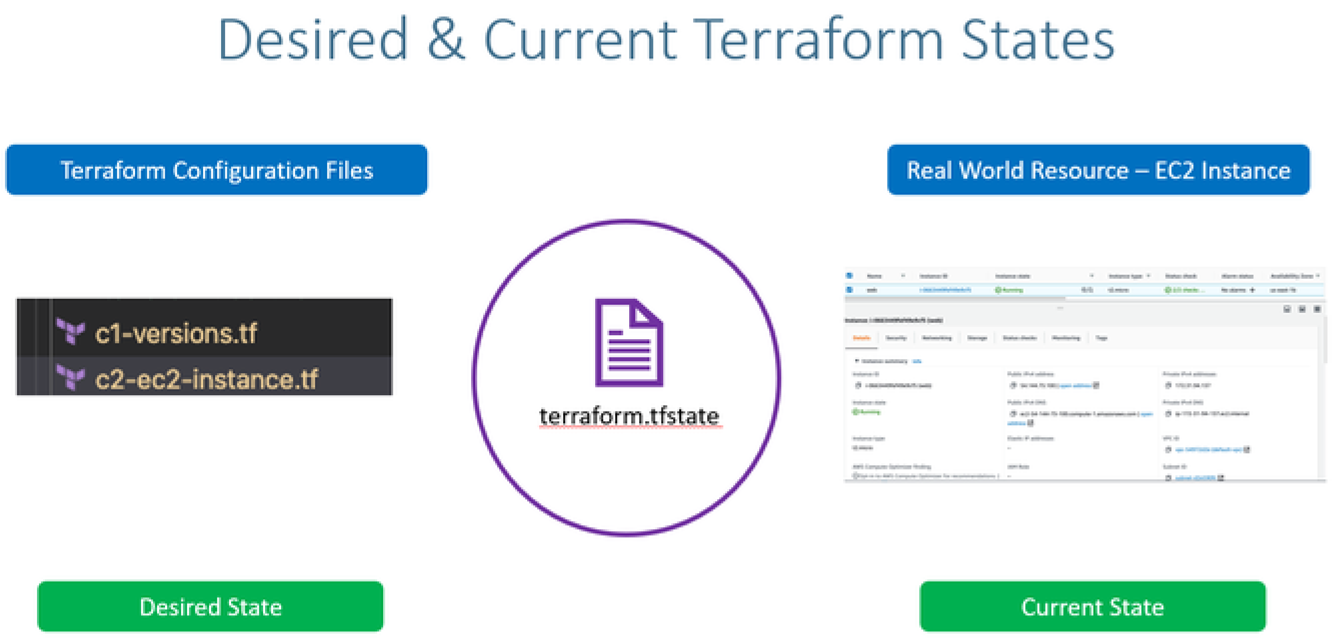
# 05-Terraform Workflow and architecture



**PS:** tf files = desire state and AWS cloud = current state


Desired & Current Terraform States
Terraform Configuration Files
c1-versions.tf
c2-ec2-instance.tf
terraform.tfstate
Real World Resource – EC2 Instance
Desired State
Current State

**Current State vs Desired State**

- When running a terraform plan, Terraform must know the current state of resources in order to effectively determine the changes that it needs to make to reach your desired configuration.
- Current State = Current Infrastructure Resource in the cloud
- Desired State = Infrastructure Configuration defined within the Terraform TF Files.
- Terraform will plan to match the desired state to the current state. If there is a difference between both, the desired state will take the preference.

## init

- Used to Initialize a working directory containing terraform config files
- This is the first command that should be run after writing a new Terraform configuration
- Downloads Providers

## validate

- Validates the terraform configurations files in that respective directory to ensure they are syntactically valid and internally consistent.

## plan

- Creates an execution plan
- Terraform performs a refresh and determines what actions are necessary to achieve the desired state specified in configuration files

## apply

- Used to apply the changes required to reach the desired state of the configuration.
- By default, apply scan s the current directory for the configuration and applies the changes appropriately.

## destroy

- Used to destroy the Terraform-managed infrastructure
- This will ask for confirmation before destroying.

## Terraform State

Terraform Local State Storage

Terraform Remote State Storage

## What is Terraform Backend ?

Backends are responsible for storing state and providing an API for state locking.

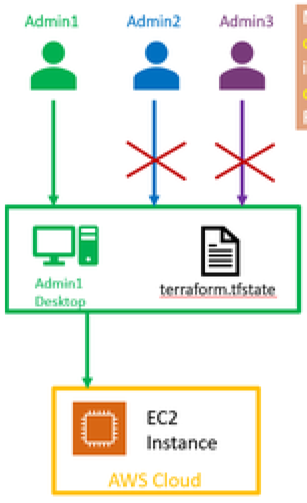
Terraform State Storage

AWS S3 Bucket

Terraform State Locking

AWS DynamoDB

## Local State File



Multiple Team members cannot update the infrastructure as they don't have access to State File

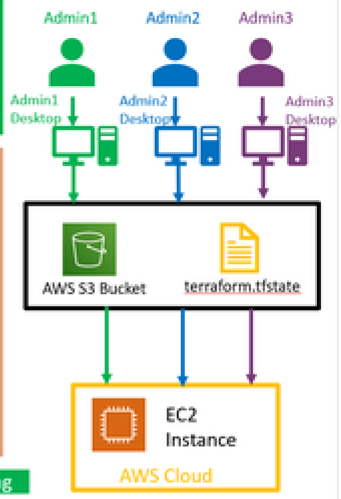
This means we need store the state file in a shared location.

## Remote State File

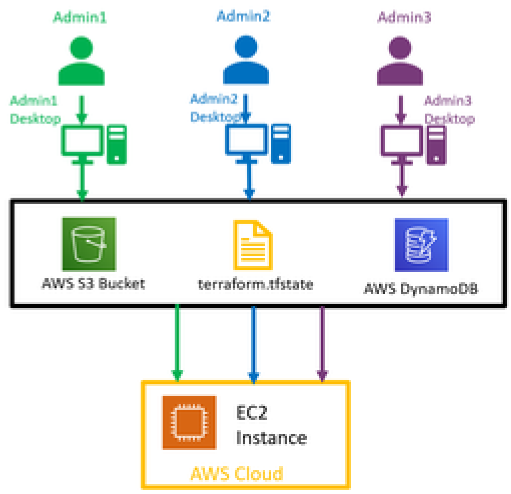
Using Terraform Backend concept we can use AWS S3 as the shared storage for State Files

If two team members are running Terraform at the same time, you may run into race conditions as multiple Terraform processes make concurrent updates to the state files, leading to conflicts, data loss, and state file corruption.

Implement State Locking



## Terraform Remote State File with State Locking



Not all backends support State Locking. AWS S3 supports State Locking with DynamoDB

State locking happens automatically on all operations that could write state (any write operation).

If state locking fails, Terraform will not continue.

You can disable state locking for most commands with the -lock flag but it is not recommended.

If acquiring the lock is taking longer than expected, Terraform will output a status message.

If Terraform doesn't output a message, state locking is still occurring if your backend supports it. (Do not interrupt it)

Terraform has a force-unlock command to manually unlock the state if unlocking failed.

If you interrupt terraform apply, the state file might get lock and you have to force-unlock

## Terraform Backends

**What Backends Do**

There are two things backends will be used for

1. Where state is stored
2. Where operations are performed.

### Store State

Terraform uses persistent state data to keep track of the resources it manages.

Everyone working with a given collection of infrastructure resources must be able to access the same state data (shared state storage).

### State Locking

State Locking is to prevent conflicts and inconsistencies when the operations are being performed

### Operations

"Operations" refers to performing API requests against infrastructure services in order to create, read, update, or destroy resources.

Not every terraform subcommand performs API operations; many of them only operate on state data.

Only two backends actually perform operations: local and remote.

The remote backend can perform API operations remotely, using Terraform Cloud or Terraform Enterprise.

What are Operations ?
terraform apply
terraform destroy

## Terraform Backends

**Backend Types**

### Enhanced Backends

**Enhanced** backends can both store state and perform operations. There are only two enhanced backends: local and remote

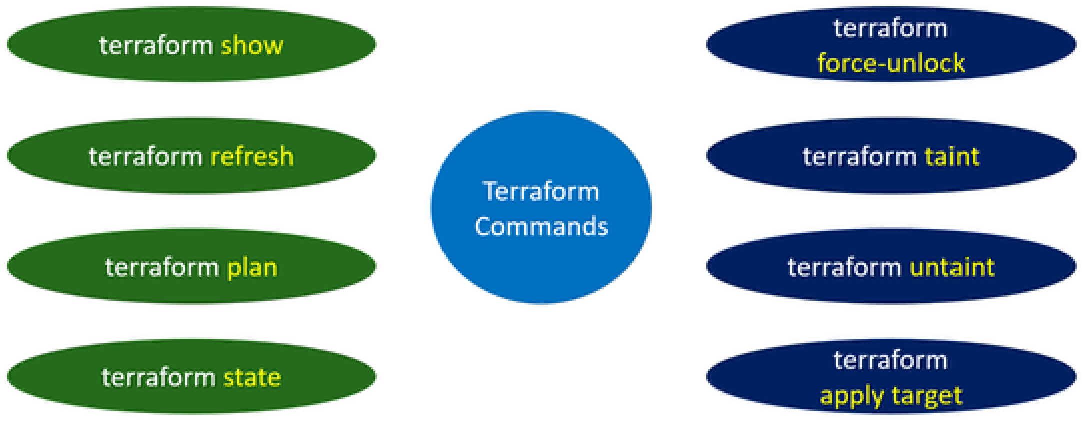
Example for Remote Backend Performing Operations : Terraform Cloud, Terraform Enterprise

### Standard Backends

**Standard** backends only store state, and rely on the local backend for performing operations.

Example: AWS S3, Azure RM, Consul, etcd, gcs http and many more

## Terraform Commands – State Perspective


terraform show
terraform refresh
terraform plan
terraform state
Terraform Commands
terraform force-unlock
terraform taint
terraform untaint
terraform apply target