**State Locking in Terraform**

**State Locking in Terraform (Easy Explanation)**

Terraform **state locking** prevents multiple users from making conflicting changes to the infrastructure at the same time.

**Example Scenario:**

Imagine two engineers, **Alice** and **Bob**, are working on the same Terraform project whose state file stored in Backends (like AWS S3, Azure storage).

* Alice runs terraform apply to create an **EC2 instance**.
* At the same time, Bob also runs terraform apply to modify the same infrastructure.
* Without state locking, both changes could overlap, causing **conflicts, resource duplication, or failures**.

**How State Locking Works:**

* When one person runs “**terraform apply”**, Terraform **locks** the state file (terraform.tfstate).
* The lock prevents others from modifying the infrastructure **until the operation is complete**.
* Once the operation finishes, the lock is **released**.

**Where is the Lock Stored?**

* If using **local state** (terraform.tfstate file), locking **does not happen**.
* If using **remote backends** (e.g., AWS S3 + DynamoDB, Terraform Cloud, Azure Storage), the state is locked **automatically**.

**Example: AWS S3 with DynamoDB Locking**

terraform {

backend "s3" {

bucket = "my-terraform-state-bucket"

key = "terraform.tfstate"

region = "us-east-1"

dynamodb\_table = "terraform-lock-table" # Enables state locking

}

}

**What Happens in the Above Setup?**

* The **S3 bucket** stores the Terraform state.
* The **DynamoDB table** locks the state file to prevent simultaneous modifications.

**What If Someone Tries to Apply While the State is locked?**

Terraform shows an error like:

Error: Error acquiring the state lock

This means another process is using the state, and you need to wait.

**Summary:**

✅ **Prevents conflicts when multiple people work on Terraform.**  
✅ **Locks the state until the current operation finishes.**  
✅ **Commonly used with remote backends like AWS S3 + DynamoDB.**