Terraform State Locking – Documentation

In Terraform, **state locking** is a mechanism that prevents multiple users or processes from simultaneously modifying the same Terraform state file. This ensures data consistency and prevents conflicts during infrastructure changes.

Without state locking, two or more Terraform operations (such as *terraform apply* or *terraform destroy*) could run at the same time against the same state file. This can cause: Corruption of the state file Partial or conflicting infrastructure changes Loss of infrastructure tracking accuracy

In the provided Terraform configuration, state is stored remotely in an Amazon S3 bucket and locked using a DynamoDB table. terraform { backend "s3" { bucket = "8764rtyibf" key = "resources/terraform.tfstate" region = "ap-south-1" dynamodb_table = "terraform-lock" } } Here: **S3 Bucket** stores the Terraform state file remotely so it can be shared between team members. **DynamoDB Table** is used for state locking. Before Terraform updates the state file, it writes a lock entry into the DynamoDB table. If another operation is already running, Terraform detects the lock and waits until it is released before proceeding.

Benefits of State Locking: Prevents simultaneous updates that could corrupt the state file. Ensures only one operation modifies infrastructure at a time. Provides better team collaboration when using remote backends.

Example Scenario:
If you run *terraform apply* on your local machine and a colleague also runs *terraform destroy* from another system at the same time, without state locking the state file could be corrupted. With DynamoDB locking, Terraform will ensure that only one command proceeds while the other waits until the lock is released.

In summary, using an S3 bucket with a DynamoDB table for state locking is a best practice for Terraform in collaborative or production environments, ensuring consistency, stability, and safety of your infrastructure state.