# **Table of Contents**

[Table of Contents 1](#_Toc186878898)

[Understanding Terraform State File Locking 2](#_Toc186878899)

[**The Significance of the State File Locking** 2](#_Toc186878900)

[**Challenges Without Locking the State File** 2](#_Toc186878901)

[**Lock the State File in Terraform** 4](#_Toc186878902)

# Understanding Terraform State File Locking

## **The Significance of the State File Locking**

**Terraform state file locking** is a mechanism to ensure that only one process or user can modify the Terraform state file at a time. This prevents issues such as:

1. **State File Corruption**: Avoids simultaneous writes that could make the state file unusable.
2. **Race Conditions**: Ensures predictable behavior by sequentially processing changes.
3. **Infrastructure Drift**: Prevents conflicting updates that could lead to discrepancies between the state file and actual infrastructure.

**How State Locking Works**

1. **Acquire Lock**: Before Terraform performs an operation (plan, apply, etc.), it attempts to acquire a lock.
2. **Modify State**: Once the lock is acquired, Terraform updates the state file.
3. **Release Lock**: After the operation is complete, the lock is released, allowing other processes to proceed.

**Supported Backends for Locking**

* **AWS S3 with DynamoDB**: Uses DynamoDB to maintain lock records.
* **Terraform Cloud**: Built-in locking mechanism.
* **Azure Blob Storage**: Automatically locks the state file during operations.

## **Challenges Without Locking the State File**

Terraform state files store information about your infrastructure. Without state locking, concurrent modifications can result in issues such as corruption, drift, and deployment failures.

**Example Scenario Without Locking**

**Initial Setup**:

Assume that Developer A is working on a local system, while Developer B is working on an EC2 instance.

Developers A and B both have access to the same main.tf file and Terraform state file.

A screenshot of a computer program

Description automatically generated

**Both Developer-A** and **Developer-B** run terraform init to Initialize terraform

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

**Both Developer-A** and **Developer-B** simultaneously run terraform apply to create an EC2 instance.

In this scenario, the EC2 instance creation by Developer B (submitted from the EC2 instance) was completed before Developer A (submitted from the local system).

Instance details created from Developer-A

A black screen with green text

Description automatically generated

Instance details created from Developer-B

A screen shot of a computer

Description automatically generated

Observe that two instances have been created: one by Developer A and the other by Developer B

A screenshot of a computer

Description automatically generated

Check the information in the state file. Observer it contains information from Developer-A.

A white rectangular object with black text

Description automatically generated

Both processes read the same state file, and when they write back their changes, Developer-B's updates overwrite Developer-A's changes, leaving the state file corrupted and inconsistent with the actual infrastructure.

## **Lock the State File in Terraform**

Locking the Terraform state file ensures that only one operation (e.g., plan, apply) can modify the state at a time. This prevents conflicts, corruption, and inconsistencies when multiple users or automated processes interact with the same state file.

**Steps to Lock the State File**

Create an S3 bucket named <your-initials> -state-bucket (replace <your-initials> with your initials) for state storage using the AWS Management Console, AWS CLI, or Terraform

A screenshot of a chat

Description automatically generated

Create a DynamoDB Table named *TerraformLockTable* for Locking using the AWS Management Console, CLI, or Terraform

A screenshot of a computer

Description automatically generated

Click on Create table

A close-up of a white background

Description automatically generated

Configure Terraform backend.tf

terraform {

backend "s3" {

bucket = "prt-state-bucket"

key = "terraform.tfstate"

region = "us-west-2"

dynamodb\_table = "TerraformLockTable"

encrypt = true

}

}

A screenshot of a computer program

Description automatically generated

**Both Developer-A** and **Developer-B** simultaneously run terraform apply

Observe that you will get below message if there is a lock already

A screen shot of a computer program

Description automatically generated