# **Table of Contents**

[Table of Contents 1](#_Toc186794216)

[Understanding Terraform State 2](#_Toc186794217)

[**The Significance of the State file** 2](#_Toc186794218)

[**Local State (Example)** 3](#_Toc186794219)

[**Remote State File (Example)** 5](#_Toc186794220)

# Understanding Terraform State

## **The Significance of the State file**

A **state file** in Terraform is a crucial component that stores the current state of your infrastructure. It is a JSON-formatted file that Terraform uses to map the resources defined in your configuration (.tf files) to real-world objects managed in your cloud provider (e.g., AWS, Azure, GCP). The state file helps Terraform keep track of all the resources it manages and their respective configurations.

**Key Functions of a State File:**

1. **Tracks Infrastructure Changes**:  
   Records the current state of infrastructure to compare with the desired state, guiding terraform apply to apply necessary changes.
2. **Source of Truth**:  
   Serves as the authoritative record of infrastructure, ensuring Terraform can correctly manage resources.
3. **Enables Resource Management**:  
   Helps manage complex infrastructure by tracking resource relationships and dependencies.
4. **Facilitates Collaboration**:  
   With remote backends, allows multiple developers to work on the same infrastructure without conflicts.

**Types of State Files:**

1. **Local State File**  
   Stored on the local machine in the same directory as your Terraform configuration. Best for individual use, but not ideal for team collaboration.
2. **Remote State File**  
   Stored in a remote backend (e.g., AWS S3, Terraform Cloud) to enable collaboration across teams and ensure the state is consistent and shared.

**Example of a State File:**

Here’s a simplified example of a state file:

A screen shot of a computer program

Description automatically generated

## **Local State (Example)**

In this case, the **state file** is stored **locally** on a developer's machine. This is the default behavior in Terraform when no backend is configured.

Developer A working on their own machine (Local State). **Create main.tf**: Developer A is working on local instance. Here’s a simple main.tf file:

A screenshot of a computer program

Description automatically generated

Initializes the local Terraform workspace. No backend configuration is specified, so the state file is stored locally by default.

terraform init

A screen shot of a computer

Description automatically generated

Developer A runs terraform apply to create the EC2 instance.

terraform apply -auto-approve

* This creates the EC2 instance as specified in the main.tf file.
* The state file (terraform.tfstate) is created locally in Developer A’s working directory.
* The state file contains information about the created resources (EC2 instance) such as IDs and tags.

A screenshot of a computer

Description automatically generated

A computer screen shot of a program

Description automatically generated

**Local State File** (terraform.tfstate): After running terraform apply, the terraform.tfstate file in Developer A's directory will look something like this:

A computer screen shot of a program code

Description automatically generated

**Outcome**: The state file (terraform.tfstate) is local and only accessible to Developer A. If **Developer B** works on the project, they will **not** have access to Developer A's state file. As a result, Developer B will initialize a **new Terraform workspace** and create the same resources (e.g., an EC2 instance) with the same configuration, which can lead to **inconsistencies** or **conflicts**.

## **Remote State File (Example)**

In this case, the **state file** is stored **remotely** (e.g., in AWS S3, Terraform Cloud). This allows for better collaboration between multiple developers, ensuring that they all share the same state and that changes are synchronized.

**Collaborative Workflow with Remote State (Using AWS S3)**

Create a new S3 bucket from AWS console

A screenshot of a computer

Description automatically generated

Developer A sets up a backend.tf configuration to store the state file remotely in an S3 bucket.

* **bucket**: The name of the S3 bucket where the state will be stored.
* **key**: The key (path) within the bucket where the state file will be stored (terraform.tfstate).
* **region**: The AWS region where the S3 bucket is located.

A screenshot of a computer

Description automatically generated

Developer A runs terraform init to initialize Terraform with the remote backend.

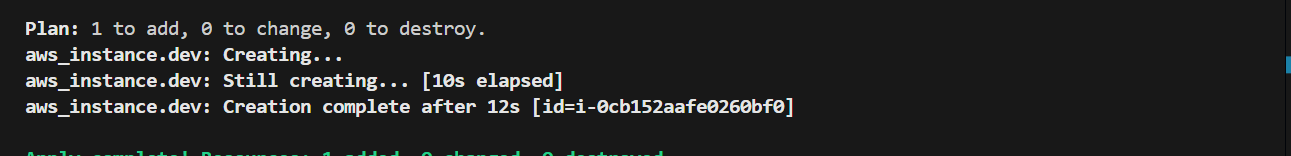
A screen shot of a computer program

Description automatically generated

**Run** terraform apply -auto-approve: Developer A creates the EC2 instance and applies the changes.

A screenshot of a computer

Description automatically generated



State File in Remote S3: The state file is now stored in the S3 bucket under the path terraform.tfstate

A screenshot of a computer

Description automatically generated

Developer B Joins the Project:  
Developer B is also working on the same project and needs access to the latest state. Follow below steps.

* Create a new EC2 instance.
* Assing IAM role which has Admin rights

A close-up of a computer screen

Description automatically generated

* Connect to EC2 instance .
* Install terraform software by following installation instructions from the [link](https://developer.hashicorp.com/terraform/install)
* Install Git if it’s not available
* Generate key using ssh-keygen command on EC2 instance and add the public key in Git repository.

A screenshot of a computer

Description automatically generated

* Clone a repository on EC2 instance

A computer screen shot of a computer program

Description automatically generated

* Change directory to your Day-3 folder : cd terraform/Day-3-State
* Developer B initializes Terraform by running terraform init, which automatically pulls the state from the S3 bucket.

A screenshot of a computer

Description automatically generated

* Developer B now has the latest state from the shared S3 bucket, and any changes made by Developer A are visible.

A screen shot of a computer

Description automatically generated

**Collaborative Workflow**:

* **Developer A** and **Developer B** both modify and apply changes.
* The state is always synchronized because it is stored remotely in the shared S3 bucket.