Terraform State

- How does Terraform knows which resources it created previously?

What is Terraform State

- Every time you run Terraform, it records information about what infrastructure it created in a *Terraform state file*
- By default, it gets created in the directory where you execute the `terraform` command.
- The file format is a private API that changes with every release and is meant for only internal use. You should never edit files or write code that depends on it.
- What happens if someone removes/updates the created resource using AWS console?

Hands-on:

> Let's look at the tfstate file that was generated as part of previous command Now run terraform destroy, and look up tfstate file again

- Key takeaways:
  - Terraform stores the state locally
  - terraform refresh command is used to refresh the state file
  - By default, Terraform always refreshes the state when you run terraform apply
  - terraform apply -refresh=false can be triggered to disable refresh for very large infrastructure

Hands-on:

> Let's learn how to get information from tfstate file

- terraform show command

```
terraform show
```

---

unplug

Problems

- Can you think of any problems with the default location of tfstate file?
  - For personal project, it's fine, but how will the team use terraform?

Shared storage for state files

- Where should we store the tfstate file so that all team members can use it?
  - Github?

    - What if I forgot to push?
    - What if I forgot to pull?

        - Passwords will be exposed
    - A shared drive accessible to all? - Locking state files
        - Without locking, 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.

Remote Backends

- The best way to manage shared storage for state files is to use Terraform’s built-in support for remote backends
- A Terraform backend determines how Terraform loads and stores state.
- The default backend, which you’ve been using this whole time, is the local backend, which stores the state file on your local disk.
- Remote backends allow you to store the state file in a remote, shared store. A number of remote backends are supported, including Amazon S3, Azure Storage, Google Cloud Storage, and more.
- It solves all the problems we talked earlier.
- If you’re using Terraform with AWS, Amazon S3 (Simple Storage Service), which is Amazon’s managed file store, is typically your best bet as a remote backend for the following reasons:
    - It’s a managed service, so you don’t have to deploy and manage extra infrastructure to use it.
    - It’s designed for 99.999999999% durability and 99.99% availability, which means you don’t have to worry too much about data loss or outages.
    - It supports encryption
    - It supports locking via DynamoDB
    - It supports versioning

HandsOn:

> Let's setup Remote Backends Refer 04-State/01-Setup-Remote-Backend

- Key takeaways:
    - You need to have a DynamoDB table and S3 pre-created.
    - The DynamoDB table must have a primary key named LockID. If not present, locking will be disabled.
    - DynamoDB table is optional
    - tfstate file gets created in S3 bucket. Let's verify.

> Let's destroy the ec2 instance only. Keep s3 bucket and DynamoDB for next HandsOn.