Lab: Terraform State Locking

Terraform uses persistent state data to keep track of the resources it manages. Since it needs the state in order to know which real-world infrastructure objects correspond to the resources in a configuration, everyone working with a given collection of infrastructure resources must be able to access the same state data.

Terraform's local state is stored on disk as JSON, and that file must always be up to date before a person or process runs Terraform. If the state is out of sync, the wrong operation might occur, causing unexpected results. If supported, the state backend will "lock" to prevent concurrent modifications which could cause corruption.

This lab demonstrates Terraform State locking.

- Task 1: Issue a Terraform Apply
- Task 2: Specify a Terraform Lock Timeout
- Task 3: Explore State Backends that Support Locking

Task 1: Generate a Terraform State Lock

This task requires that you open two seperate terminal windows to your working directory.

In Terminal #1 generate a lock on your state file by issuing a `terraform apply`

`Terminal #1`

```
terraform apply

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value:
```

Do not provide any answer at this time.

In Terminal #2 within the same working directory, issue a `terraform apply`

`Terminal #2`

```
terraform apply

| Error: Error acquiring the state lock
```


```
|
| Error message: resource temporarily unavailable
| Lock Info:
|   ID:        7aa0f3c3-51dc-f0d8-76cf-64f4953cbeee
|   Path:      terraform.tfstate
|   Operation: OperationTypeApply
|   Who:       gabe@MacBook-Pro.local
|   Version:   1.0.10
|   Created:   2021-11-08 04:56:27.40246 +0000 UTC
|   Info:
|
|
| Terraform acquires a state lock to protect the state from being written
| by multiple users at the same time. Please resolve the issue above and try
| again. For most commands, you can disable locking with the "-lock=false"
| flag, but this is not recommended.
```

Task 2: Specify a Terraform Lock Timeout

In Terminal #2, re-run a `terraform apply` this time with `-lock-timeout` value

`Terminal #2`

```
terraform apply -lock-timeout=60s
Acquiring state lock. This may take a few moments...
```

`Terminal #1` Answer `no` to the `terraform apply` command in the first terminal to free the state lock and observe be behavior in the second terminal.

Task 3: Explore State Backends that Support Locking

Not all Terraform backends support locking - Terraform's documentation identifies which backends support this functionality. Some common Terraform backends that support locking include:

- Remote Backend (Terraform Enterprise, Terraform Cloud)
- AWS S3 Backend (with DynamoDB)
- Google Cloud Storage Backend
- Azure Storage Backend

Obviously locking is an important feature of a Terraform backend in which there are multiple people collaborating on a single state file.