Section 8

Using the Terraform Backend to manage state

The Terraform backends

Terraform uses Backends to manage the state file and the Terraform operations plan(),apply(),fmt(),taint()

The state file is saved by default on local disk and the Terraform ops run on local machine.

The backend config allows to address some challenges by abstracting where the state file is saved or where the TF ops are run.

- Development is a team sport..need to sharing state file across team, use versioning or with CI/CD pipelines.

- Security..state are clear text locally with sensitive data and it is great to have it encrypted. There is also audit logging.

- Remote operation done on a remote system other than your local machine.

There are 2 types of Backend:

1. Enhanced:

Local - default - all on local host

Remote -- TF ops done remotely and state stored remotely.. Only Terraform cloud

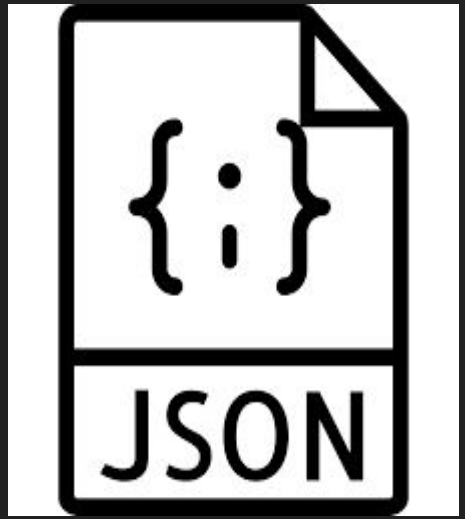
2. Standard:

TF ops locally and state file stored remotely:
S3, artifactory, consul, swift, TF enterprise,..

ENHANCED BACKEND: LOCAL

Terraform Operations

Terraform init()
Terraform plan()
Terraform apply()
Terraform taint()
Terraform fmt()
.....

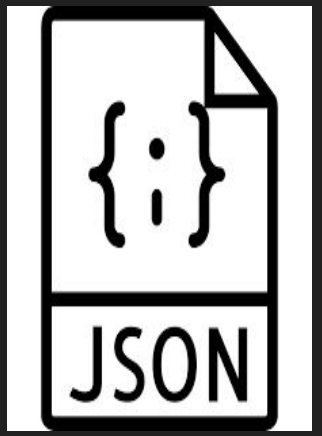


State file

ENHANCED BACKEND: REMOTE

Terraform Operations

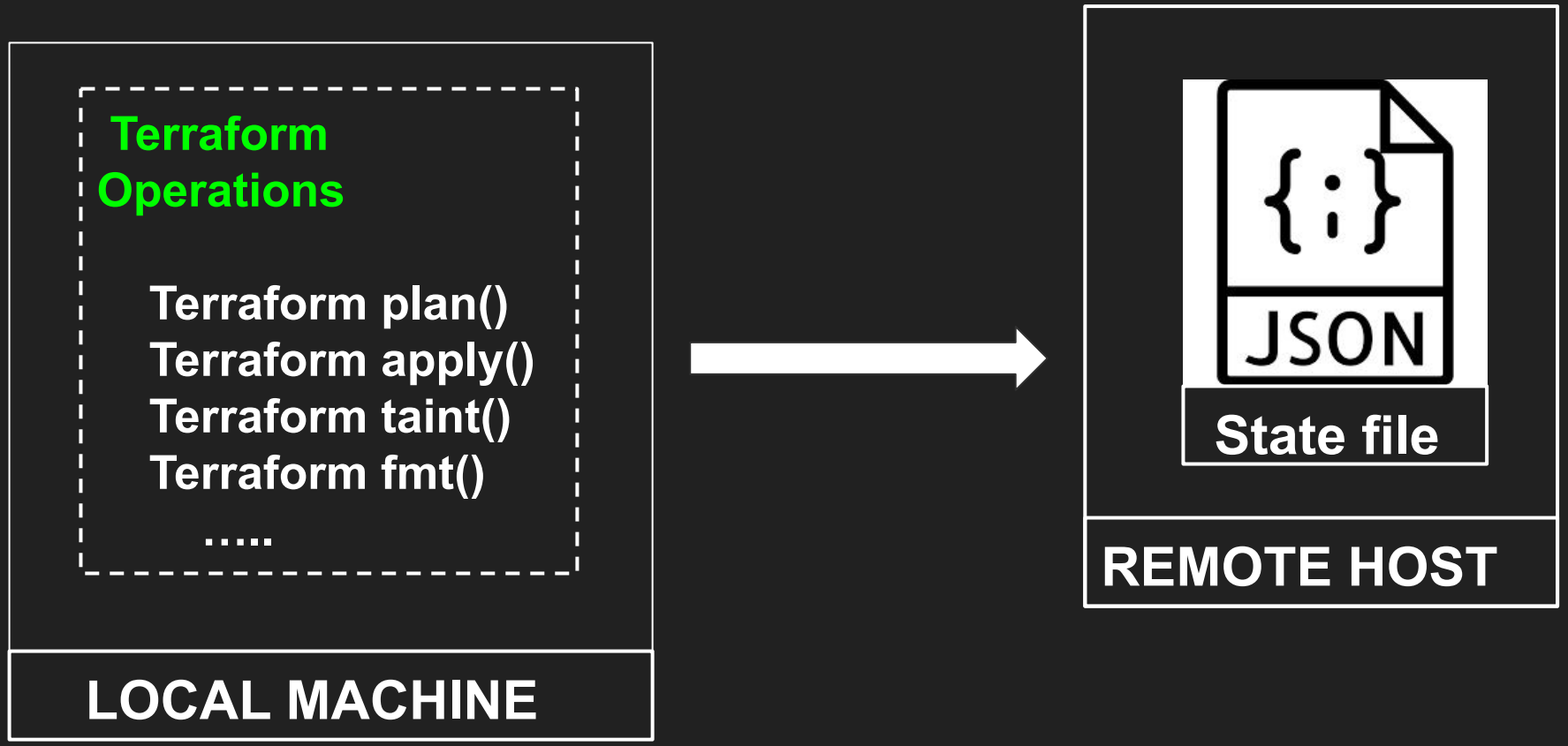
Terraform init()
Terraform plan()
Terraform apply()
Terraform taint()
Terraform fmt()
.....



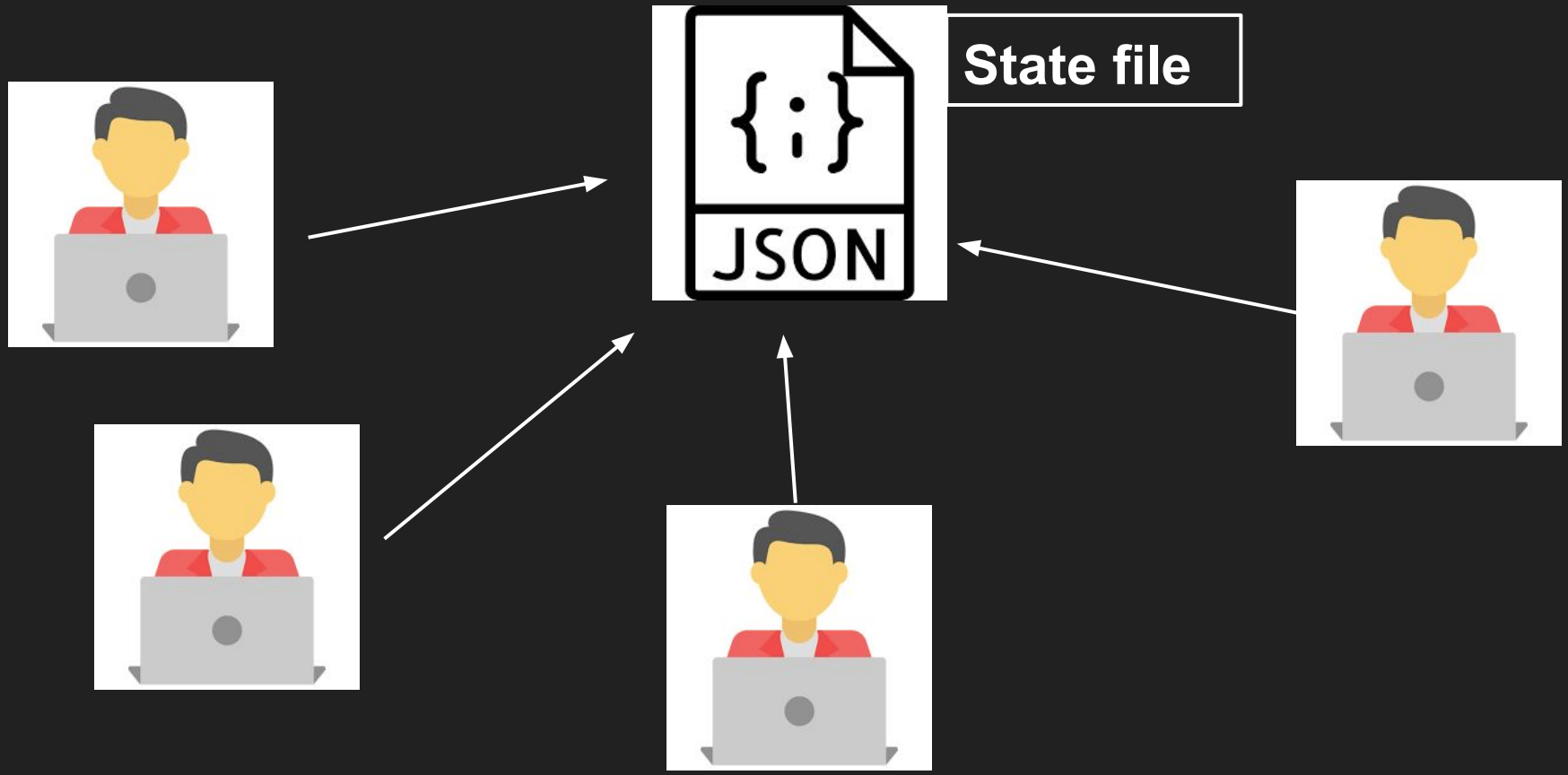
State file

Terraform Cloud

STANDARD BACKEND: remote state



STANDARD BACKEND: remote state



Terraform State Locking

State Locking prevents the corruption of the state file with simultaneous TF operations writing to the state file.

It is optional but recommended.

State locking will lock the state during any TF operations that write to the state file.
Allows changes to be done by multiple teams or individuals without corrupting the state file

Not all backends support state locking
The backend will have to provide an API to support state locking.

A LockID is generated and the lock will only be removed once the user is finished writing to the state.

State-locking can be overridden using:
$ terraform force-unlock LockID

Standard Backend:
remote state with S3 and DynamoDB

S3 does not support state locking by default so you will have to create a DynamoDB database to prevent data corruption.

Benefit of S3 is using SSE encryption to allow encryption of the state file.

The DynamoDB will have to be created with a key named LockID. State lock will be disabled otherwise.

Standard Backend:
remote state with Google Cloud Storage

Refreshing the Terraform state

Terraform refresh will compare the state file with the actual state provisioned in the provider infrastructure and then reconcile the two by updating the terraform state file.

This is can be used to also find out the most up to date state before making any changes. It will change the infrastructure by will trigger any changes for the next Terraform apply

It is good to note that the terraform plan and apply will automatically do a terraform refresh prior to doing anything.

Advanced State Management:
>terraform state list and show

The terraform state has to be managed using the Terraform state commands.

manual editing should be done to the state file at your own risk..

- Terraform state list
- Terraform state show <resource address>
- Terraform state mv
- Terraform state push
- Terraform state pull
- Terraform state rm

$terraform state list command outputs the list of terraform managed resources using their resource address.

$terraform state show <resource_address> command outputs a single resource attributes in a json format after it is provisioned in an infrastructure provider

Advanced State Management:
>terraform state push and pull

$terraform state pull outputs the entire content of the state file in json format locally or remotely

The terraform show command would only the modules and resource details of the state file but not the lineage, serial number, TF version used etc..

$terraform state push command pushes the local state file to a remote state file. (Rarely used)

If a state file already exist in the standard backend, the local state file will have to have the same lineage and a serial number higher than the remote state file for the push to be successful

Advanced State Management:
>terraform state mv and rm

$terraform state mv command can be used for renaming resources within the state file. Here we are referring the name of the resource in the resource address.

It can also be used for moving resources or modules to another module within the same state file or to another state file

$terraform state rm command will delete a resource or module from the state file so that it will no longer be managed by Terraform.

For example, goal is not to destroy the resource in the real world but just remove it from the config and the state file or just the state file if the config is changed.

Terraform sensitive info encryption.

Terraform data in local or remote state files are stored in plain-text JSON files are not encrypted at rest by default

This may contain sensitive information like username and password for DB login or other credentials related requirements.

Remote state backend can encrypt state file at rest and communication to local system using TLS. This depends on remote backend types such S3 and terraform cloud

Using the encrypt parameter is needed in the backend block or by pre provisioning your S3 with encryption on the AWS side.