## Objective 7: Implement and maintain state

▼ Describe default local backend

- **Backends** - by default Terraform uses 'local' backend
  - This is an abstraction that determines how state is loaded and how an operation is executed. It allows such actions as non-local file state storage and remote execution
  - Benefits:
    - Working in a team - can store state remotely and use locks to prevent corruption in state
    - Keeping sensitive information off disk - state in backends are only stored in memory
    - Remote operations - `terraform apply` can take time for larger infrastructures, some backends can use remote operations instead to execute commands remotely
- Local example config:

```
terraform {
    backend "local" {
    path = "relative/path/to/terraform.tfstate"
  }
}
```

▼ Outline state locking

State Locking

- if supported by your backend state can be locked so others cannot change it while another change is being made.
- this is automatic for all operations that can write state
- Backends types supporting locking: (standard)artifactory,azurerm,consul,cos,etcd,etcdv3,gcs,http,manta,oss,pg,s3,swift,terraform enterprise, and in enhanced backends there are remote operations as well (plan, apply, etc.)
- A lock can be forced open with `force-unlock` which requires a unique nonce lock ID

▼ Handle backend authentication methods

- Different backends have different configuration for authentication, authentication can be done different ways within a backend.
- Example with azurerm:

```
#authenticating using the Azure CLI or a Service Principal:
```

```
terraform {
backend "azurerm" {
    resource_group_name  = "StorageAccount-ResourceGroup"
    storage_account_name = "abcd1234"
    container_name       = "tfstate"
     key                 = "prod.terraform.tfstate"
    }
}

#---------------------------------
#authenticating using Managed Service Identity (MSI):
 terraform {
    backend "azurerm" {
        storage_account_name = "abcd1234"
        container_name       = "tfstate"
        key                  = "prod.terraform.tfstate"
        use_msi              = true
        subscription_id  = "00000000-0000-0000-0000-000000000000"
        tenant_id        = "00000000-0000-0000-0000-000000000000"
        }
}
```

▼ Describe remote state storage mechanisms and supported standard backends

**Remote State Storage**

- Uses Terraform Cloud as a backend, allows free remote state management
- Tutorial for Remote State Storage

**Standard backends**

- artifactory,azurerm,consul,cos,etcd,etcdv3,gcs,http,manta,oss,pg,s3,swift,terraform enterprise

▼ Describe effect of Terraform refresh on state

- `terraform refresh`
- reconciles the state Terraform knows about via the state file.
- refresh does not modify the infrastructure, it does modify the state file.

▼ Describe backend block in configuration and best practices for partial configurations

Backend Config

- Backends are configured in the Terraform files.

- there can only be one backend
- This is an example of a config for "consul":

```
terraform {
  backend "consul" {
    address = "demo.consul.io"
    scheme  = "https"
    path    = "example_app/terraform_state"
  }
}
```

Partial Configuration

- You can omit certain arguments from the backend configuration.
- This is done to avoid storing access keys or private data in the main configuration
- adding the omitted arguments must be done during the initialization process by doing the following:
  - Interactively - If interact input is enabled it will ask you for the required values
  - File - `terraform init -backend-config=PATH` that contains the variables
  - Command-link key/value pairs - `terraform init -backend-config="KEY=VALUE"` **This isn't recommended for secret keys since CL flags can be stored in a history file.

▼ Understand secret management in state files

- state contains resource IDs and attributes, db data that may have passwords.
- with remote state, state is only in memory when in use. This is more secure
- also some backends can encrypt the state data at rest
- Terraform Cloud encrypts state at rest and protects it with TLS in transit.
- Terraform Cloud keeps track of user identity, and state changes.

⏮️Objective 6 || Objective 8 ⏭️

⬅️BACK README