Objective 6: Navigate Terraform workflow

▼ Describe Terraform workflow (Write -> Plan -> Create)

- **Write**
  - Author infrastructure as code
- **Plan**
  - Preview changes before applying
- **Create (Apply)**
  - Provision reproducible infrastructure
- Configuration is written like any program, use version control to keep track of changes

```
# Create repository
$ git init my-infra && cd my-infra
Initialized empty Git repository in /.../my-infra/.git/
# Write initial config
$ vim main.tf
# Initialize Terraform
$ terraform init
Initializing provider plugins...
# ...
Terraform has been successfully initialized!
```

- running `Terraform plan` repeatedly is useful to make sure there are no syntax errors and the correct code is being written per the desired outcome.
- First run `Terraform apply` before pushing to git to make sure the provisions are correct
- While working in teams it is best to use branches to avoid code collision.

```
$ git checkout -b <branch-name>
 Switched to a new branch <branch-name>
```

- **Teams** can review changes via Terraform plans and pull requests
- **Terraform cloud** helps streamline this process in a team setting
  - Write - secure location for storing variables and state with the "remote" backend, then a Terraform Cloud API key is used to edit the configuration and run plans against the state file.

```
terraform {
backend "remote" {
    organization = "my-org"
    workspaces {
```

```
      prefix = "my-app-"
    }
  }
}
#----------------------------------------
$ terraform workspace select my-app-dev
Switched to workspace "my-app-dev".
$ terraform plan
Running plan remotely in Terraform Enterprise.
Output will stream here. To view this plan in a browser, visit:
https://app.terraform.io/my-org/my-app-dev/.../
Refreshing Terraform state in-memory prior to plan...
# ...
Plan: 1 to add, 0 to change, 0 to destroy.
```

  - Plan - plans are automatically run when a pull request is created. Status updates are shown in the pull request view.
  - Apply - A confirm and apply is needed after merging to run an `apply`.

The next section will go over Terraform Commands

For a reference of all commands checkout out this file on Terraform CLI

▼ Initialize a Terraform working directory (terraform init)

```
terraform init
```

- prepares working directory for use
- safe to run multiple times to bring the working directory up to date
- it will never delete a configuration or state

▼ Validate a Terraform configuration (terraform validate)

```
terraform validate
```

- validates the configuration files in the dir, this does not apply to things like remote state or provider APIs
- validate checks for syntax, internal consistency, such as attribute names and value types
- safe to run automatically or as a test step for CI
- requires initialized working directory

▼ Generate and review an execution plan for Terraform (terraform plan)

```
Terraform plan
```

- Creates an execution plan, automatically performs a refresh

▼ Execute changes to infrastructure with Terraform (terraform apply)

```
terraform apply
```

- applies changes needed for the desired state of the configuration
- runs set of actions defined by a `terraform plan` command

▼ Destroy Terraform managed infrastructure (terraform destroy)

```
terraform destroy
```

- completely destroys and terraform created infrastructure

⏮️Objective 5 || Objective 7 ⏩

🔙README