Terraform Core Commands

You'll use these constantly

These commands form the standard workflow for provisioning and managing infrastructure. Get comfortable with them — you'll run them every day.

Initialise your workspace

```
terraform init
```

Downloads provider plugins and sets up your backend.

Run this **once** when you start a new project, and again whenever you add new providers or change your backend configuration.

Check your syntax

```
terraform validate
```

Validates your configuration files without connecting to Azure.

> *Use this to catch syntax errors and misconfigurations before running `plan`. It's fast and costs nothing.*

Preview changes

```
terraform plan
```

Shows exactly what Terraform will create, update, or delete — without touching anything.

Always run this before applying. The output uses symbols:

- `+` creates a new resource
- `~` modifies an existing resource
- `-` deletes a resource
- `-/+` replaces a resource (deletes then recreates)

Apply changes

```
terraform apply
```

Applies your configuration to Azure. Terraform shows you the plan and asks for confirmation — type `yes` to proceed.

Skipping the prompt

```
terraform apply -auto-approve
```

Skips the confirmation step entirely.

> *Use `auto-approve` only in CI/CD pipelines or when you're certain about the changes. Never use it if you haven't reviewed the plan output.*

Destroy resources

```
terraform destroy
```

Deletes **all** resources Terraform manages in the current state.

> *Use this to clean up when you're done testing. Treat it with the same caution as `apply` — always review the plan output first.*

Format your code

```
terraform fmt
```

Formats all `.tf` files to match Terraform's style conventions.

Run this before committing code. In our pipeline, `terraform fmt -check` will fail the build if files aren't formatted correctly — so run it locally first.

Check your version

```
terraform version
```

Shows your installed Terraform version. Useful when troubleshooting or confirming compatibility with a provider.

The standard workflow

Here's the sequence you'll follow every time:

1. Write your configuration in `.tf` files
2. `terraform init` — once per project setup
3. `terraform validate` — catch errors early
4. `terraform plan` — review what will change
5. `terraform apply` — deploy to Azure
6. `terraform destroy` — clean up when done

> *Always **validate before plan**, and always **plan before apply**.*

A note on state

Terraform tracks everything it manages in a **state file**. This is how it knows what exists in Azure and what needs to change.

In our pipeline, state is stored remotely in Azure Blob Storage — never commit a local `terraform.tfstate` file to your repository.