



Terraform Quickstart Cheatsheet

A Curated Cheatsheet of Terraform CLI Commands
Organized by Topic

By Kirshi Yin

Curious Devs Corner

2025

Table of Contents

Table of Contents	2
Introduction	3
 Setup & Initialization	4
 Code Formatting & Validation	4
 Modules & Providers	5
 Planning Infrastructure	5
 Applying Changes	5
 Destroying Infrastructure	6
 Terraform Cloud Authentication	6
 State Management	7
 Importing Resources	7
 Outputs	8
 Interactive Console	8
 Workspaces	8
 Unlocking State	9
 Dependency Graph	9
 Miscellaneous	9
Final Thoughts	10

Introduction

This cheatsheet gives you quick access to the most useful Terraform CLI commands. Keep it open in a tab or download it as a PDF. It's organized by topic—like init, modules, providers—so you can find what you need fast.

You can use it to look up commands during a project. Or test each one in your own setup if you're learning Terraform.

All commands are up to date with the latest Terraform version (v1.12.1).

Setup & Initialization

- `terraform init` # Initialize working directory and download providers
 - `terraform init -upgrade` # Upgrade modules and providers to latest versions
 - `terraform init -lock=false` # Skip state lock during init
 - `terraform init -input=false` # Disable interactive prompts
 - `terraform init -migrate-state` # Migrate existing state to new backend
 - `terraform -install-autocomplete` # Enable CLI autocompletion
-

Code Formatting & Validation

- `terraform fmt` # Format code to HCL standard
 - `terraform fmt -recursive` # Format code in subdirectories
 - `terraform validate` # Validate configuration syntax
 - `terraform validate -json` # Output validation results in JSON format
 - `terraform validate -backend=false` # Skip backend validation
-



Modules & Providers

- `terraform get` # Download modules
 - `terraform get -update` # Update modules to latest versions
 - `terraform providers` # List providers used in configuration
-



Planning Infrastructure

- `terraform plan` # Show execution plan
 - `terraform plan -out=plan.out` # Save plan to file
 - `terraform plan -destroy` # Show plan to destroy all resources
-



Applying Changes

- `terraform apply` # Apply changes to reach desired state
- `terraform apply plan.out` # Apply changes from saved plan
- `terraform apply -auto-approve` # Apply without interactive approval
- `terraform apply -lock=false` # Skip state lock during apply
- `terraform apply -parallelism=5` # Limit parallel resource operations
- `terraform apply -var="key=value"` # Pass variable via CLI
- `terraform apply -var-file="vars.tfvars"` # Load variables from file

- `terraform apply -replace="resource" # Replace specified resource`
 - `terraform apply -target="resource" # Target specific resource`
 - `terraform apply -refresh=false # Skip state refresh before apply`
-

Destroying Infrastructure

- `terraform destroy # Destroy all managed resources`
 - `terraform destroy -auto-approve # Destroy without interactive approval`
 - `terraform destroy -target="resource" # Destroy specific resource`
-

Terraform Cloud Authentication

- `terraform login # Authenticate with Terraform Cloud`
 - `terraform login <hostname> # Authenticate with specified host`
 - `terraform logout # Remove stored credentials`
 - `terraform logout <hostname> # Remove credentials for specified host`
-



State Management

- `terraform show` # Show current state in human-readable format
 - `terraform refresh` # Update state file with real-world resource data
 - `terraform state list` # List resources in state file
 - `terraform state show resource` # Show details of specific resource
 - `terraform state mv src dst` # Move resource to different address
 - `terraform state rm resource` # Remove resource from state
 - `terraform state pull > state.tfstate` # Download current state to file
 - `terraform state push` # Upload local state file to remote backend
 - `terraform state replace-provider old new` # Replace provider in state
-



Importing Resources

- `terraform import resource_name resource_id` # Import existing resource into state
-



Outputs

- `terraform output` # List all outputs
 - `terraform output -json` # Output in JSON format
 - `terraform output -state=state.tfstate` # Show outputs from specified state file
 - `terraform output output_name` # Show specific output value
-



Interactive Console

- `terraform console` # Open interactive console for expressions
 - `echo 'expression' | terraform console` # Evaluate expression via pipe
-



Workspaces

- `terraform workspace list` # List all workspaces
 - `terraform workspace show` # Show current workspace
 - `terraform workspace new name` # Create new workspace
 - `terraform workspace select name` # Switch to specified workspace
 - `terraform workspace delete name` # Delete specified workspace
-



Unlocking State

- `terraform force-unlock LOCK_ID` # Manually unlock state
-



Dependency Graph

- `terraform graph` # Generate DOT format dependency graph
 - `terraform graph -type=plan` # Graph for plan, refresh-only, or destroy
 - `terraform graph -draw-cycles` # Highlight dependency cycles
-



Miscellaneous

- `terraform version` # Show Terraform version
 - `terraform -help` # Show help for all commands
 - `terraform fmt -help` # Show help for fmt command
-

Final Thoughts

I hope this cheatsheet will save you time and effort. If you're just getting started with Terraform, I recommend you check out my ebook—[Terraform Quickstart](#).