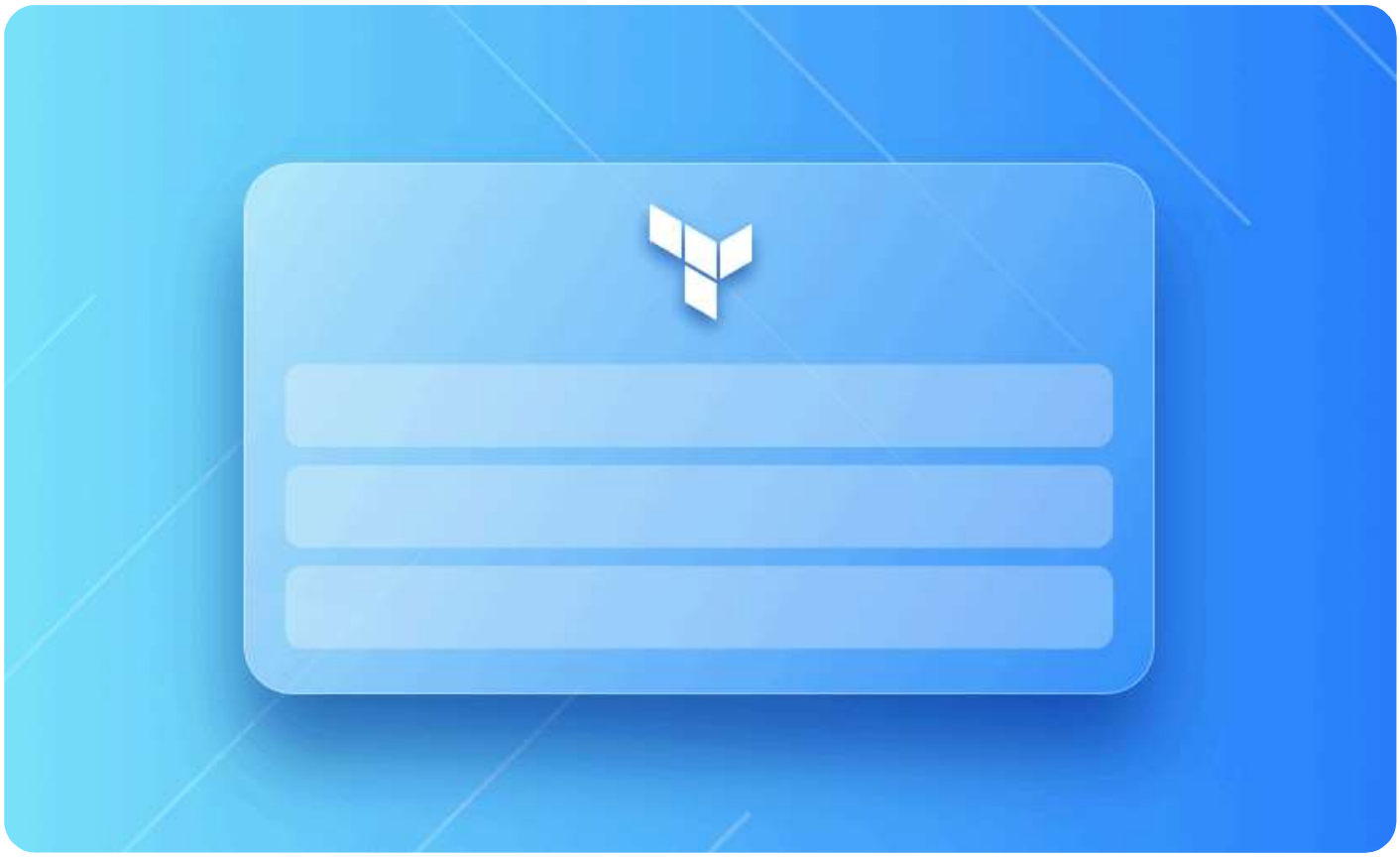
TERRAFORM

Terraform Cheat Sheet – 23 Terraform CLI Commands & Examples

Flavius Dinu, Jack Roper

Updated 09 Nov 2023 · 26 min read



Sometimes you just want to get straight to the commands you need to use with a particular tool, without having to trawl through all the documentation. In this post, I'll highlight the commonly used commands used on the Terraform CLI so you can get straight into the action without the pain! You will also find here a Terraform Cheat Sheet PDF version to download.

This Terraform command reference guide was written using the latest version of Terraform v.1.3.7. New commands and subcommands can be added and depreciated over time with

Note: New versions of Terraform will be placed under the BUSL license, but everything created before version 1.5.x stays open-source. OpenTofu is an open-source version of Terraform that will expand on Terraform's existing concepts and offerings. It is a viable alternative to HashiCorp's Terraform, being forked from Terraform version 1.5.6. OpenTofu retained all the features and functionalities that had made Terraform popular among developers while also introducing improvements and enhancements. OpenTofu works with your existing Terraform state file, so you won't have any issues when you are migrating to it.

Terraform Commands Cheat Sheet:

1. Get Help
2. Show your Terraform version
3. Format your Terraform code
4. Initialize your directory
5. Download and install modules
6. Validate your Terraform code
7. Plan your Infrastructure
8. Deploy your Infrastructure
9. Destroy your infrastructure
10. 'Taint' or 'Untaint' your resources

20. Produce a dependency diagram
21. Test your expressions
22. Switch Working Directory
23. Shell Tab-completion

Get Help

`terraform -help` — Get a list of available commands for execution with descriptions. Can be used with any other subcommand to get more information.

`terraform fmt -help` — Display help options for the `fmt` command.

Show Your Terraform Version

`terraform version` — Show the current version of your Terraform and notifies you if there is a newer version available for download.

Format Your Terraform Code

`terraform fmt --diff` — Display differences between original configuration files and formatting changes.

`terraform fmt --check` — Useful in automation CI/CD pipelines, the check flag can be used to ensure the configuration files are formatted correctly, if not the exit status will be non-zero. If files are formatted correctly, the exit status will be zero.

Initialize Your Directory

`terraform init` — In order to prepare the working directory for use with Terraform, the `terraform init` command performs Backend Initialization, Child Module Installation, and Plugin Installation.

`terraform init -get-plugins=false` — Initialize the working directory, do not download plugins.

`terraform init -lock=false` — Initialize the working directory, don't hold a state lock during backend migration.

`terraform init -input=false` — Initialize the working directory, and disable interactive prompts.

`terraform init -migrate-state` — Reconfigure a backend, and attempt to migrate any

ctory, do not verify

Download and Install Modules

Note this is usually not required as this is part of the `terraform init` command.

`terraform get` — Download and installs modules needed for the configuration.

`terraform get -update` — Check the versions of the already installed modules against the available modules and installs the newer versions if available.

Validate Your Terraform Code

`terraform validate` — Validate the configuration files in your directory and does not access any remote state or services. `terraform init` should be run before this command.

`terraform validate -json` — To see easier the number of errors and warnings that you have.

Learn how to validate your configuration locally.

Plan Your Infrastructure

what actions will be

n then be passed to

Deploy Your Infrastructure

`terraform apply` — Create or update infrastructure depending on the configuration files. By default, a plan will be generated first and will need to be approved before it is applied.

`terraform apply -auto-approve` — Apply changes without having to interactively type 'yes' to the plan. Useful in automation CI/CD pipelines.

`terraform apply <planfilename>` — Provide the file generated using the `terraform plan -out` command. If provided, Terraform will take the actions in the plan without any confirmation prompts.

`terraform apply -lock=false` — Do not hold a state lock during the Terraform apply operation. Use with caution if other engineers might run concurrent commands against the same workspace.

`terraform apply -parallelism=<n>` — Specify the number of operations run in parallel.

`terraform apply -var="environment=dev"` — Pass in a variable value.

`terraform apply -var-file="varfile.tfvars"` — Pass in variables contained in a file.

`terraform apply -target "module.app[0]"` — Apply changes only to the targeted

`terraform destroy --auto-approve` — Destroy the infrastructure without having to interactively type 'yes' to the plan. Useful in automation CI/CD pipelines.

`terraform destroy -target="module.appgw.resource[\"key\"]"` — Destroy an instance of a resource created with for_each.

'Taint' or 'Untaint' Your Resources

Use the `taint` command to mark a resource as not fully functional. It will be deleted and re-created.

`terraform taint vm1.name` — Taint a specified resource instance.

`terraform untaint vm1.name` — Untaint the already tainted resource instance.

Refresh the State File

`terraform refresh` — Modify the state file with updated metadata containing information on the resources being managed in Terraform. Will not modify your infrastructure.

View Your State File

c state file, you can

vn.

Manipulate Your State File

`terraform state` — One of the following subcommands must be used with this command in order to manipulate the state file.

`terraform state list` — Lists out all the resources that are tracked in the current state file.

`terraform state mv` — Move an item in the state, for example, this is useful when you need to tell Terraform that an item has been renamed, e.g. `terraform state mv vm1.oldname vm1.newname`

`terraform state pull > state.tfstate` — Get the current state and outputs it to a local file.

`terraform state push` — Update remote state from the local state file.

`terraform state replace-provider hashicorp/azurerm customproviderregistry/azurerm` — Replace a provider, useful when switching to using a custom provider registry.

`terraform state rm` — Remove the specified instance from the state file. Useful when a resource has been manually deleted outside of Terraform.

ce in the state file.

n State

defined in the configuration files under vm1.name.

See our terraform import tutorial for more details.

Btw. We created a comprehensive pdf version of Terraform Cheatsheet dedicated to those who want to learn and remember the key Terraform commands and have a quick reference guide in pdf form. You can get it below.

Get Provider Information

uration files and

workspace command. Workspaces can be useful when an engineer wants to test a slightly different version of the code. It is not recommended to use Workspaces to isolate or separate the same infrastructure between different development stages, e.g. Dev / UAT / Production, or different internal teams.

`terraform workspace show` — Show the name of the current workspace.

`terraform workspace list` — List your workspaces.

`terraform workspace select <workspace name>` — Select a specified workspace.

`terraform workspace new <workspace name>` — Create a new workspace with a specified name.

`terraform workspace delete <workspace name>` — Delete a specified workspace.

View Your Outputs

`terraform output` — List all the outputs currently held in your state file. These are displayed by default at the end of a `terraform apply`, this command can be useful if you want to view them independently.

`terraform output -state=<path to state file>` — List the outputs held in the specified state file. **-state** option is ignored when the remote state is used.

Release a Lock on Your Workspace

`terraform force-unlock <lock_id>` — Remove the lock with the specified lock ID from your workspace. Useful when a lock has become 'stuck', usually after an incomplete Terraform run.

Log In and Out to a Remote Host (Terraform Cloud)

`terraform login` — Grab an API token for Terraform cloud (app.terraform.io) using your browser.

`terraform login <hostname>` — Log in to a specified host.

`terraform logout` — Remove the credentials that are stored locally after logging in, by default for Terraform Cloud (app.terraform.io).

`terraform logout <hostname>` — Remove the credentials that are stored locally after logging in for the specified hostname.

Produce a Dependency Diagram

pendencies

m called Graphwiz

`terraform graph -type=plan` — Specify the type of graph to output, either `plan`, `plan-refresh-only`, `plan-destroy`, or `apply`.

`terraform graph -draw-cycles` — You can see if there are any dependency cycles between the resources.

Test Your Expressions

`terraform console` — Allow testing and exploration of expressions on the interactive console using the command line. e.g. 1+2 🙄

With the `terraform console` command, you have the ability to test different pieces of code. All you have to do is write `terraform console`, and then you can write HCL code.

```
terraform console

# The below command will merge list elements into a string, separating them with comma
> join(",",["foo","bar"])
"foo,bar"

# The below command will do math operations
> 1 + 5
6
```

Switch Working Directory

You also have the ability to run Terraform from another directory if the need arises. This is particularly useful when you are using different automations and you don't want to change directory. This is done by:

```
terraform -chdir=”../dev” apply
```

Shell Tab-completion

Terraform also comes with an optional Shell Tab-completion. It can be useful if you are just starting out with Terraform. However, Terraform CLI is pretty lightweight, and you won't usually reach very long commands.

To install the Shell Tab-completion you will need to first run:

```
terraform -install-autocomplete
```

After that you will need to resource your profile. This is done by either closing and opening the terminal, or by running source *path_to_your_profile*.

, which was initially
ge called
ces. Terraform

others. It is stateful, keeping track of the deployed infrastructure using a state file.

What is Terraform CLI?

The Terraform Command Line Interface (CLI), is a command-line tool that provides a simple way for users to interact with the infrastructure components defined in the Terraform configuration. It offers multiple commands, from initializing your terraform directory to planning, applying, and destroying infrastructure resources. With the Terraform CLI, you also have the ability to check outputs, do state-related operations, and even test different expressions.

Why use Spacelift with Terraform?

Working in a team of Terraform developers can be challenging. Spacelift is built to provide a great CI/CD experience concerning IaC. It can be used for version control of the code, state management, and much more. Spacelift lets you customize the entire infrastructure life-cycle management by providing the ability to run pre and post commands at every stage, which can be very useful in keeping track of things. If you are interested in trying it, create your free trial account.

Key Points

using the Terraform

wing blog posts:

ices.

Written by

Flavius Dinu

Flavius is a passionate Developer Advocate with an Infrastructure as Code mindset and expertise in DevOps & Cloud Engineering. He holds ITIL Foundation Certificate in IT Service Management and Hashicorp Terraform Associate Certification. He currently works at Spacelift, and in his free time, he blogs at techblog.flaviusdinu.com, where he provides tutorials, tips, and

on cloud and DevOps technologies. He specializes in Terraform, Azure, Azure DevOps, and Kubernetes and holds multiple certifications from Microsoft, Amazon, and Hashicorp. Jack enjoys writing technical articles for well-regarded websites.

Read also

Terraform CLI Commands Cheatsheet

Initialize/ plan/ apply your IaC, manage modules, state, and more.

9 min read

TERRAFORM

28 min read

Terraform Functions, Expressions, Loops (Examples)

Product

Documentation

How it works

Spacelift Tutorial

Pricing

Customer Case Studies

Integrations

Security

System Status

Product Updates

Company

About Us

Careers

Contact Sales

Partners

Learn

Blog

Atlantis Alternative

Terraform Cloud Alternative

Terraform Enterprise Alternative

Spacelift for AWS

Terraform Automation

Get our newsletter

Subscribe

Terraform CLI Commands Cheatsheet

Initialize/ plan/ apply your IaC, manage modules, state, and more.