spacelift // The Most Flexible IaC Platform

Terraform Cheat Sheet

Get Help

`terraform -help`

Get a list of available commands for execution with descriptions.

`terraform fmt -help`

Display help options for the fmt command.

Format Your Terraform Code

`terraform fmt`

Format your Terraform configuration files using the HCL language standard.

`terraform fmt -recursive`

Also formats files in subdirectories.

Download and Install Modules

`terraform get`

Downloads and installs modules needed for the configuration.

`terraform get -update`

Checks the versions of the already installed modules against the available modules and installs the newer versions if available.

Initialize Your Directory

`terraform init`

Prepare the working directory for use with Terraform by Backend Initialization, Child Module Installation, and Plugin Installation.

`terraform init -get=false`

Disables downloading modules for this configuration.

`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 existing state.

Plan Your Infrastructure

`terraform plan`

Generate an execution plan, showing you what actions will be taken without performing the planned actions.

`terraform plan -out=<path>`

Save the plan file to a given path.

`terraform plan -destroy`

Create a plan to destroy all objects, rather than the usual actions.

Deploy Your Infrastructure

`terraform apply`

Create or update infrastructure depending on the configuration files.

`terraform apply -auto-approve`

Apply changes without having to interactively type 'yes' to the plan.

`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.

`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 -replace="module.appgw[0]"`

Instructs Terraform to replace the given resource. Preferred to taint.

`terraform apply -target="module.appgw[0]"`

Apply changes only to the targeted resource.

Destroy Your Infrastructure

`terraform destroy`

Destroy the infrastructure managed by Terraform.

`terraform destroy -target="module.appgw[0]"`

Destroy only the targeted resource.

`terraform destroy -auto-approve`

Destroys the infrastructure without having to interactively type 'yes' to the plan.

`terraform destroy -target="module.appgw.resource[\"key\"]"`

This will destroy an instance of a resource created with for_each.

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.

Get Provider Information

`terraform providers`

Display a tree of providers used in the configuration files and their requirements.

View Your State File

`terraform show`

Shows the state file in a human-readable format.

Test Your Expressions

`terraform console`

Allow testing and exploration of expressions on the interactive console using the command line.

Validate Your Terraform Code

`terraform validate`

Validates the configuration files in your directory, and does not access any remote state or services.

`terraform validate -json`

By using this option, you can easily see the number of errors and warnings that you have.

Import Existing Infrastructure into Your Terraform State

`terraform import vm1.name -i id123`

Import a VM with id123 into the state and into the configuration files under vm1.name. The configuration has to exist as it is not generated by the command.

View Your Outputs

`terraform output`

List all the outputs currently held in your state file.

`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.

`terraform output -json`

Lists the outputs held in your state file in JSON format to make them machine-readable.

`terraform output vm1_public_ip`

List a specific output held in your state file.

'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`

Use the taint command to mark a resource as not fully functional. It will be deleted and re-created. This command is deprecated, use terraform apply -replace instead.

`terraform untaint vm1.name`

Untaint the already tainted resource instance.

Show Your Terraform Version

`terraform version`

Show the current version of your Terraform and notify you if there is a newer version available for download.

Refresh the State File

`terraform refresh`

Modifies the state file with updated metadata containing information on the resources being managed in Terraform.

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`

List out all the resources that are tracked in the current state file.

`terraform state mv`

Move an item in the state.

`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.

`terraform state rm`

Remove the specified instance from the state file.

Release a Lock on Your Workspace

`terraform force-unlock`

Remove the lock with the specified lock ID from your workspace.

Produce a Dependency Diagram

`terraform graph`

Produce a graph in DOT language showing the dependencies between objects in the state file.

`terraform graph -plan=tfplan`

Produce a dependency graph using a specified plan file.

`terraform graph -type=plan`

Specifies 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.

Manage Your Workspaces

`terraform workspace`

One of the following subcommands must be used with the workspace command.

`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.