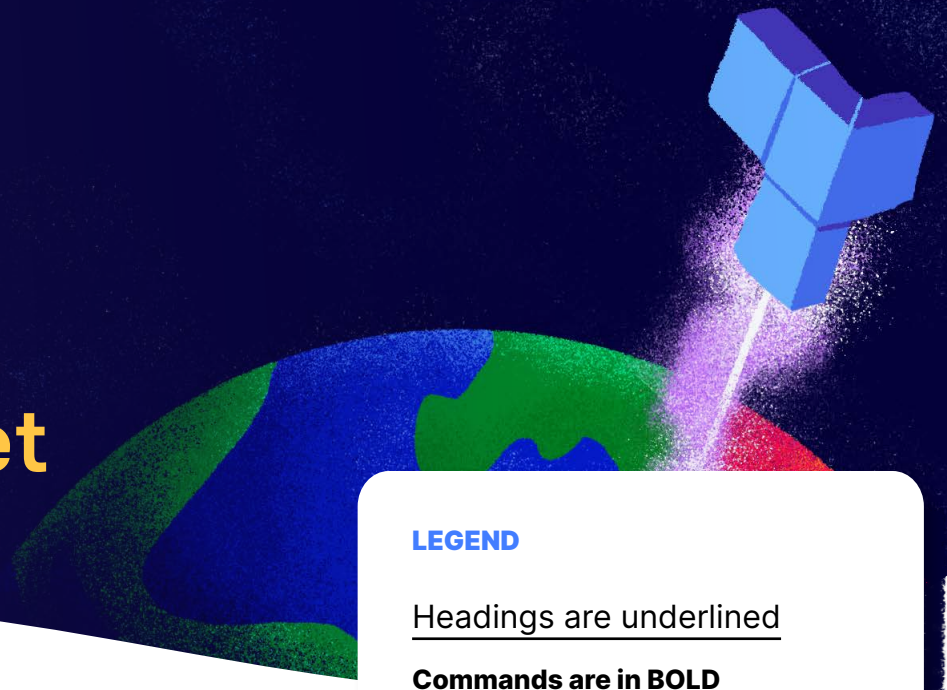
A CLOUD GURU

Terraform Cheatsheet



LEGEND

Headings are underlined

Commands are in BOLD

Helpful command descriptions are green

Plan, deploy and cleanup infrastructure

1. **terraform apply --auto-approve** Apply changes without being prompted to enter “yes”
2. **terraform destroy --auto-approve** Destroy/cleanup deployment without being prompted for “yes”
3. **terraform plan -out plan.out** Output the deployment plan to plan.out
4. **terraform apply plan.out** Use the plan.out plan file to deploy infrastructure
5. **terraform plan -destroy** Outputs a destroy plan
6. **terraform apply -target=aws_instance.my_ec2** Only apply/deploy changes to the targeted resource
7. **terraform apply -var my_region_variable=us-east-1** Pass a variable via command-line while applying a configuration
8. **terraform apply -lock=true** Lock the state file so it can’t be modified by any other Terraform apply or modification action (possible only where backend allows locking)
9. **terraform apply refresh=false** Do not reconcile state file with real-world resources(helpful with large complex deployments for saving deployment time)
10. **terraform apply --parallelism=5** Number of simultaneous resource operations
11. **terraform refresh** Reconcile the state in Terraform state file with real-world resources
12. **terraform providers** Get information about providers used in current configuration

Terraform CLI tricks

1. **terraform -install-autocomplete** Setup tab auto-completion, requires logging back in

Format and validate Terraform code

1. **terraform fmt** Format code per HCL canonical standard
2. **terraform validate** Validate code for syntax
3. **terraform validate -backend=false** Validate code skip backend validation

Initialize your Terraform working directory

1. **terraform init** Initialize directory, pull down providers
2. **terraform init -get-plugins=false** Initialize directory, do not download plugins
3. **terraform init -verify-plugins=false** Initialize directory, do not verify plugins for Hashicorp signature

Terraform Workspaces

1. **terraform workspace new mynewworkspace** Create a new workspace
2. **terraform workspace select default** Change to the selected workspace
3. **terraform workspace list** List out all workspaces

Terraform state manipulation

1. **terraform state show aws_instance.my_ec2** Show details stored in Terraform state for the resource
2. **terraform state pull > terraform.tfstate** Download and output terraform state to a file
3. **terraform state mv aws_iam_role.my_ssm_role module.custom_module** Move a resource tracked via state to different module
4. **terraform state replace-provider hashicorp/aws registry.custom.com/aws** Replace existing provider with another
5. **terraform state list** List all the resources tracked in the current state file
6. **terraform state rm aws_instance.myinstace** Unmanage a resource, delete it from Terraform state file

Terraform Import and Outputs

1. **terraform import aws_instance.new_ec2_instance i-abcd1234** Import EC2 instance with id i-abcd1234 into the Terraform resource named "new_ec2_instance" of type "aws_instance"
2. **terraform import 'aws_instance.new_ec2_instance[0]' i-abcd1234** Same as above, imports a real-world resource into an instance of Terraform resource
3. **terraform output** List all outputs as stated in code
4. **terraform output instance_public_ip** List a specific declared output
5. **terraform output -json** List all outputs in JSON format

Terraform miscellaneous commands

1. **terraform version** Display Terraform binary version, also warns if version is old
2. **terraform get -update=true** Download and update modules in the "root" module

Terraform Console (Test out Terraform interpolations)

1. **echo 'join(",",["foo","bar"])' | terraform console** Echo an expression into terraform console and see its expected result as output
2. **echo '1 + 5' | terraform console** Terraform console also has an interactive CLI just enter "terraform console"
3. **echo "aws_instance.my_ec2.public_ip" | terraform console** Display the Public IP against the "my_ec2" Terraform resource as seen in the Terraform state file

Terraform Graph (dependency graphing)

1. **terraform graph | dot -Tpng > graph.png** Produce a PNG diagram showing relationship and dependencies between Terraform resources in your configuration/code

Terraform Taint/Untaint

1. **terraform taint aws_instance.my_ec2** Taint resource to be recreated on next apply
2. **terraform untaint aws_instance.my_ec2** Remove taint from a resource
3. **terraform force-unlock LOCK_ID** Force-unlock a locked state file, LOCK_ID provided when locking the State file beforehand

Terraform Cloud

1. **terraform login** Obtain and save API token for Terraform cloud
2. **terraform logout** Log out of Terraform Cloud, defaults to hostname app.terraform.io