# General Commands

## #Get the terraform version

### terraform version

## #Download and update root modules

### terraform get -update=true

## #Open up a terraform interactive terminal

### terraform console

## #Format terraform code to HCL standards

### terraform fmt

## #Validate terraform code syntax

### terraform validate

# Intialise terraform

## #Initialise directory/pull providers

### terraform init

## #Initialise directory, do not download plugins

### terraform init -get-plugins=false

## #Initialise directory, do not verify plugins

### terraform init -verify-plugins=false

## #force plugin installation from a directory

### terraform init -plugin-dir=PATH

## #upgrade modules and plugins at initilisation

### terraform init -upgrade

## #Update backend configuration

### terraform init -migrate-state -force-copy

## #Skip backend configuration

### terraform init -backend=false

# Plan Terraform

## #Produce a plan with diff between code and state

### terraform plan

## #output a plan file for reference during apply

### terraform plan -out latest.tfplan

## #Output a plan to show effect of terraform destroy

### terraform plan -destroy

## #Target a specific resource for deployment

### terraform plan -target=ADDRESS

## #Create a destroy plan & output

### terraform plan -destroy

# Apply Terraform

## #Apply the current state of terraform code

### terraform apply

## #Specify a previously generated plan to apply

### terraform apply current.tfplan

## #Enable auto-approval or automation

### terraform apply -auto-approve

## #Specify number of simultaneous apply operations

### terraform apply --parallelism=5

## #lock the state file to stop other Terraform actions

### terraform apply -lock=true

# Destroy Terraform

## #Destroy resources managed by terraform state

### terraform destroy

## #Enable auto-approval or automation

### terraform destroy -auto-approve

# manage terraform state

## #List all resources in terraform state

### terraform state list

## #track an existing resource in state under new name

### terraform state mv <SOURCE> <DESTINATION>

## #Pull state and save to a local file

### terraform state pull > terraform.tfstate

## #Push state to a remote location

### terraform state push <PATH>

## #Replace resource provider

### terraform state replace-provider A B

## #Taint a resource to force redeployment on apply

### terraform taint ADDRESS

## #Untaint a previously tainted resource

### terraform untaint ADDRESS

## #Reconcile the state in the state file

### terraform refresh

# Manage Terraform Workspaces

## #List the available workspaces

### terraform workspace list

## #Create a new workspace

### terraform workspace new <WORKSPACE>

## #Select an existing workspace

### terraform workspace select default

# Inspect Infrastructure Commands

## #Create a dot diagram of terraform dependencies

### terraform graph | dot -Tpng > graph.png

## List the root module outputs

### terraform output

## #List the outputs, particularly in JSON formatting.

### terraform output –json

## #Outputs human readable format

### terraform show

## #Show details about a specific resource

### terraform state show <RESOURCE>

# Miscellaneous

## #Enable tab auto-completion in the terminal

### terraform -install-autocomplete

## #Show information about provider requirements

### terraform providers

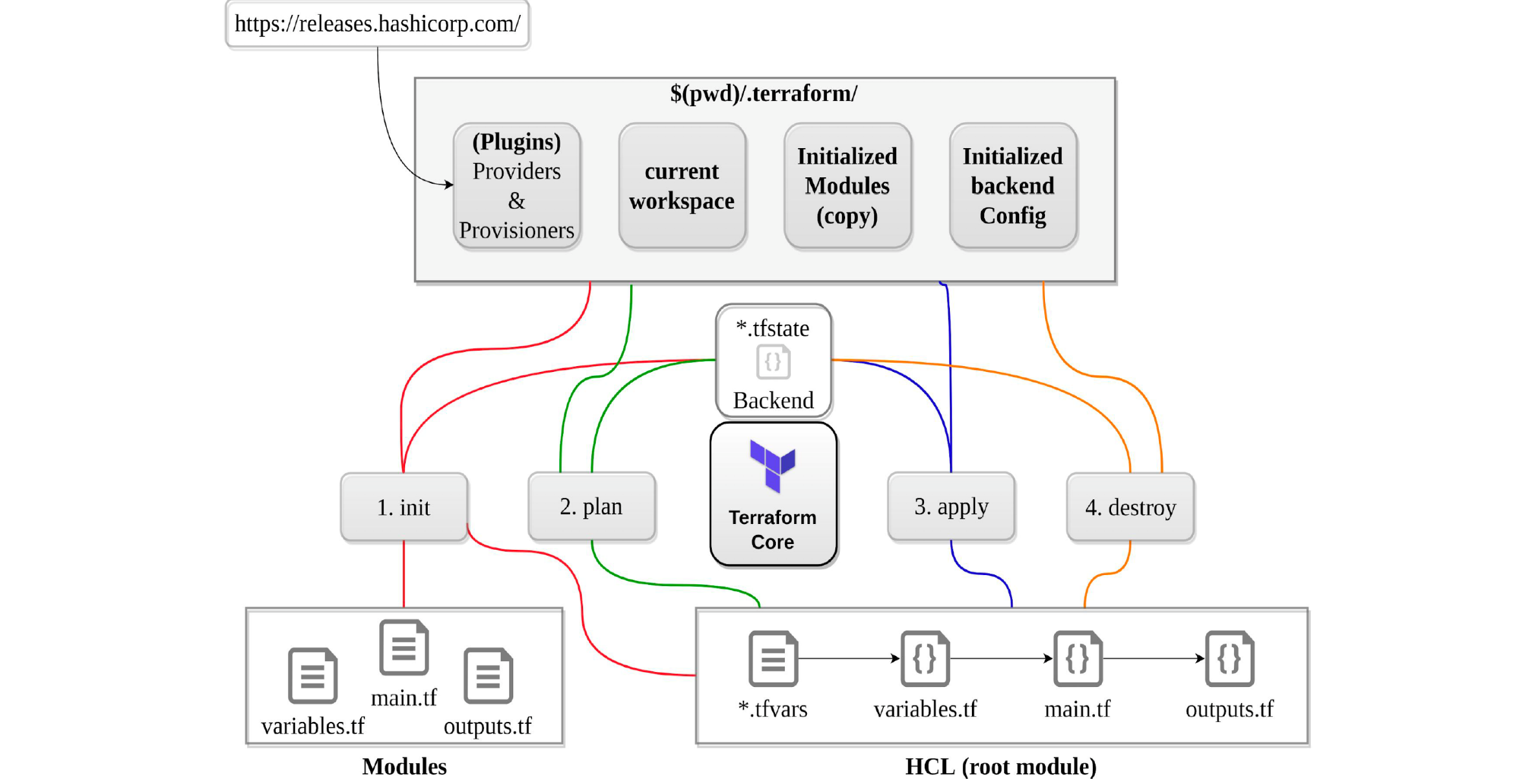
## #Replace resource provider

### terraform state replace-provider A B

## #Download and update modules in the “root” module

### terraform get -update=true

To target a particular resource and its dependencies use the **-target**flag. Works with PLAN, APPLY & DESTROY



#### init

* initialises a working directory containing Terraform configuration files.
* performs
* backend initialization , storage for terraform state file.
* modules installation, downloaded from terraform registry to local path
* provider(s) plugins installation, the plugins are downloaded in the sub-directory of the present working directory at the path of .terraform/plugins
* can be run multiple times, to bring the working directory up to date with changes in the configuration
* **does not delete the existing configuration or state**

#### validate

* is used to validate/check the syntax of the Terraform files.
* verifies whether a configuration is syntactically valid and consistent, regardless of any provided variables or existing state.
* syntax check is done on all the terraform files in the directory and will display an error if any of the files doesn’t validate.

#### plan

* creates a execution plan
* calculates the difference between the last-known state and the current state, then presents this difference as the output of the terraform plan operation to the user in their terminal
* **does not modify the infrastructure or state.**
* allows a user to see which actions Terraform will perform prior to making any changes to reach the desired state
* will scan all \*.tf  files in the directory and create the plan
* supports -out to save the plan

#### apply

* apply changes to infrastructure to reach the desired state.
* scans the current directory for the configuration and applies the changes appropriately.
* can be provided with an explicit plan, saved as output from running terraform plan
* If no explicit plan file is given on the command line, terraform apply will create a new plan automatically and prompt for approval to apply it
* **will modify the infrastructure and the state.**
* if a resource successfully creates but fails during provisioning:
* *Terraform will error and mark the resource as “tainted”.*
* *A resource that is tainted has been physically created but can’t be considered safe to use since provisioning failed.*
* *Terraform does not automatically roll back and destroy the resource during the apply when the failure happens, because that would go against the execution plan: the execution plan will said a resource will be created, but does not say it will ever be deleted.*
* does not import any resource.
* supports -auto-approve to apply the changes without asking for a confirmation
* supports -target to apply a specific module

#### destroy

* **destroy** the infrastructure and all resources
* **modifies both state and infrastructure**
* terraform destroy -target can be used to destroy targeted resources
* terraform plan -destroy allows creation of destroy plan that can be run later