

init

* initializes 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
* supports -upgrade to update all previously installed plugins to the newest version that complies with the configuration’s version constraints
* is safe to 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

* validates syntactically for format and correctness.
* is used to validate/check the syntax of the Terraform files.
* verifies whether a configuration is syntactically valid and internally consistent, regardless of any provided variables or existing state.
* A 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

* create a execution plan
* traverses each vertex and requests each provider using parallelism
* calculates the difference between the last-known state and  
  the current state and presents this difference as the output of the terraform plan operation to 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
* will perform refresh for each resource and might hit rate limiting issues as it calls provider APIs
* all resources refresh can be disabled or avoided using
  + -refresh=false or
  + target=xxxx or
  + break resources into different directories.
* supports -out to save the plan

apply

* apply changes to reach the desired state.
* scans the current directory for the configuration and applies the changes appropriately.
* can be provided with a explicit plan, saved as out from 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 also 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 would’ve 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

refresh

* used to reconcile the state Terraform knows about (via its state file) with the real-world infrastructure
* does not modify infrastructure, but does modify the state file

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

https://www.youtube.com/[Terraform theory.docx](Terraform%20theory.docx)watch?v=l5k1ai\_GBDE