**Terraform commands**

**terraform init**

* when you run it required plugin for the provider get automatically downloaded
* Get saved .terraform directory
* Need to run when you insert a new provider into your code

**terraform destroy**

* Allows to destroy all the resource that created within the folder

**terraform plan**

* Creates an execution plan that shows what Terraform will change to your infrastructure

**terraform plan -out=/path**

* Store plan in specific file in binary format if some changes happen in configuration file it and you execute terraform apply it apply the changes from binary file only not from configuration file

**terraform apply**

* Executes the actions proposed in a terraform plan

**terraform apply -auto-approve**

* Executes withought asking permission

**terraform refresh**

* Check the latest state of infra stated in configuration and update state file accordingly ()
* Perform automatically at the time of Plan and apply

**terraform plan -refresh=false**

* It won't refresh the whole state it only refresh where changes will happen(when you have a large infra and u have task to deploy small resource in that case us can use)

**terraform plan -refresh=false -target=<targetresourceName.localName>**

* This will only execute specified changes in infra not whole configuration

**terraform fmt**

* Correct the format of the terraform code

**terraform validate**

* Validates the currant terraform file if unsupported args are there it returns error.

**terraform apply -replace = "resourceName.localName" (eg - aws\_instance.myec2)**

* It will replace the current instance with the same new one

**terraform output iam\_names**

* It will show the output IAM names from state file

**terraform graph > graph.dot**

* Used to generate visual representation will store in graph.dot file. (needs additional tool -graph viz)

**export TF\_LOG\_PATH= /path/terraform-crash.log**

* This will store all the trace related massage in terraform-crash.log file

**terraform plan -generate-config-out=<File\_Name>**

* Command To Autogenerate Code for Imported Resource

**terraform apply -auto-approve**

* Command To Generate the Final State file

**terraform login**

* Login to the terraform cloud, managed terraform via CLI

**terraform Workspace:**

Allow to have multiple workspace and each workspace can have different set of environment variable associated

**cmd:**

|  |  |
| --- | --- |
| **terraform workspace -h** | list cmd available |
| **terraform workspace show** | Show current workspace |
| **terraform workspace new dev** | Will create workspace call dev |
| **terraform workspace new prod** | Will create a workspace prod |
| **terraform workspace list** | List all workspace available |
| **terraform workspace select dev** | Switch into dev workspace |

Terraform commands, their use cases, and examples:

**1. terraform init**

* **Use Case:** Initializes a new or existing Terraform configuration. It downloads provider plugins and sets up the working directory.
* **Example:**

bash

Copy code

terraform init

**2. terraform plan**

* **Use Case:** Creates an execution plan, showing what actions Terraform will take to change the infrastructure based on the current configuration.
* **Example:**

bash

Copy code

terraform plan

**3. terraform apply**

* **Use Case:** Applies the changes required to reach the desired state of the configuration. It provisions or updates resources.
* **Example:**

bash

Copy code

terraform apply

**4. terraform destroy**

* **Use Case:** Destroys all the resources managed by Terraform. This is useful for cleanup or removing an entire environment.
* **Example:**

bash

Copy code

terraform destroy

**5. terraform show**

* **Use Case:** Provides human-readable output of the state or a plan file. It helps to understand the current state of resources.
* **Example:**

bash

Copy code

terraform show

**6. terraform output**

* **Use Case:** Extracts the outputs from the state file. Useful for retrieving information about resources that have been created.
* **Example:**

bash

Copy code

terraform output

**7. terraform validate**

* **Use Case:** Validates the configuration files for syntax and internal consistency. It checks for configuration errors before applying.
* **Example:**

bash

Copy code

terraform validate

**8. terraform import**

* **Use Case:** Imports existing resources into Terraform management. It allows Terraform to manage resources that were created outside of Terraform.
* **Example:**

bash

Copy code

terraform import aws\_instance.example i-12345678

**9. terraform state**

* **Use Case:** Manages the state file. This command provides operations to manipulate and query the state file directly.
* **Example:**

bash

Copy code

terraform state list

terraform state show aws\_instance.example

**10. terraform taint**

* **Use Case:** Marks a resource for recreation during the next apply. This is useful if a resource is in a bad state.
* **Example:**

bash

Copy code

terraform taint aws\_instance.example

**11. terraform untaint**

* **Use Case:** Removes the taint mark from a resource, preventing it from being recreated during the next apply.
* **Example:**

bash

Copy code

terraform untaint aws\_instance.example

**12. terraform workspace**

* **Use Case:** Manages different workspaces, which are separate instances of your Terraform configuration. Useful for managing environments like dev, staging, and prod.
* **Example:**

bash

Copy code

terraform workspace new dev

terraform workspace select dev

**13. terraform fmt**

* **Use Case:** Formats Terraform configuration files to a canonical format and style. It helps in maintaining consistent formatting.
* **Example:**

bash

Copy code

terraform fmt

**14. terraform providers**

* **Use Case:** Displays the providers required by the configuration and their versions. It helps in understanding the dependencies in the configuration.
* **Example:**

bash

Copy code

terraform providers

**15. terraform graph**

* **Use Case:** Generates a visual representation of the dependency graph of the resources. Useful for understanding resource relationships.
* **Example:**

bash

Copy code

terraform graph | dot -Tpng > graph.png

**16. terraform workspace list**

* **Use Case:** Lists all available workspaces. It helps in viewing and managing the different workspaces in the configuration.
* **Example:**

bash

Copy code

terraform workspace list

**17. terraform workspace show**

* **Use Case:** Shows the current workspace. Useful for verifying which workspace is currently selected.
* **Example:**

bash

Copy code

terraform workspace show

**18. terraform version**

* **Use Case:** Displays the version of Terraform and the provider plugins used. Useful for troubleshooting and ensuring compatibility.
* **Example:**

bash

Copy code

terraform version

**19. terraform login**

* **Use Case:** Authenticates to the Terraform Cloud or Enterprise service. It is used to obtain a token for accessing Terraform Cloud or Enterprise.
* **Example:**

bash

Copy code

terraform login

**20. terraform logout**

* **Use Case:** Logs out from Terraform Cloud or Enterprise by removing the authentication token. Useful for security and cleanup.
* **Example:**

bash

Copy code

terraform logout

**21. terraform force-unlock**

* **Use Case:** Manually unlocks a state that is locked by another process. Useful if a process was interrupted and the state is left locked.
* **Example:**

bash

Copy code

terraform force-unlock LOCK\_ID

**22. terraform console**

* **Use Case:** Opens an interactive console to evaluate expressions and interact with the Terraform configuration. Useful for testing and debugging.
* **Example:**

bash

Copy code

terraform console

**23. terraform state mv**

* **Use Case:** Moves items in the state file from one address to another. Useful for reorganizing or renaming resources in the state.
* **Example:**

bash

Copy code

terraform state mv aws\_instance.old aws\_instance.new

**24. terraform state rm**

* **Use Case:** Removes items from the state file. Useful for manually deleting resources from state without destroying them.
* **Example:**

bash

Copy code

terraform state rm aws\_instance.example

**25. terraform graph**

* **Use Case:** Generates a visual representation of the dependency graph of resources. This helps in understanding the relationships and dependencies between resources.
* **Example:**

bash

Copy code

terraform graph | dot -Tsvg > graph.svg

**26. terraform validate**

* **Use Case:** Validates the configuration files to ensure they are syntactically valid and internally consistent.
* **Example:**

bash

Copy code

terraform validate

**27. terraform push**

* **Use Case:** Pushes the local state to a remote state storage backend, such as Terraform Cloud. Useful for syncing local and remote state.
* **Example:**

bash

Copy code

terraform push

**28. terraform workspace new**

* **Use Case:** Creates a new workspace. Workspaces allow you to manage multiple versions of your infrastructure.
* **Example:**

bash

Copy code

terraform workspace new prod

**29. terraform workspace delete**

* **Use Case:** Deletes an existing workspace. Useful for cleaning up unused or obsolete workspaces.
* **Example:**

bash

Copy code

terraform workspace delete prod

**30. terraform refresh**

* **Use Case:** Updates the state file with the latest data from the infrastructure. Useful for ensuring the state file reflects the current real-world state.
* **Example:**

bash

Copy code

terraform refresh

**31. terraform debug**

* **Use Case:** Enables debug logging to troubleshoot issues. Provides more verbose output for diagnosing problems.
* **Example:**

bash

Copy code

TF\_LOG=DEBUG terraform apply

**32. terraform validate**

* **Use Case:** Validates the configuration files for syntax and internal consistency. Useful for catching errors before applying changes.
* **Example:**

bash

Copy code

terraform validate

**33. terraform state pull**

* **Use Case:** Downloads and displays the current state file from the remote backend. Useful for examining the remote state.
* **Example:**

bash

Copy code

terraform state pull

**34. terraform state push**

* **Use Case:** Uploads the current state file to the remote backend. Useful for syncing local changes with the remote state.
* **Example:**

bash

Copy code

terraform state push

**35. terraform login**

* **Use Case:** Authenticates to Terraform Cloud or Terraform Enterprise. Useful for managing access to remote backends.
* **Example:**

bash

Copy code

terraform login

**36. terraform logout**

* **Use Case:** Logs out from Terraform Cloud or Enterprise. Useful for removing authentication credentials.
* **Example:**

bash

Copy code

terraform logout

**37. terraform workspace select**

* **Use Case:** Switches to a different workspace. Useful for managing multiple environments within the same configuration.
* **Example:**

bash

Copy code

terraform workspace select dev

**38. terraform workspace show**

* **Use Case:** Displays the current workspace. Useful for confirming the workspace you are working in.
* **Example:**

bash

Copy code

terraform workspace show

**39. terraform import**

* **Use Case:** Imports existing resources into Terraform management. Useful for incorporating existing infrastructure into Terraform.
* **Example:**

bash

Copy code

terraform import aws\_instance.example i-12345678

**40. terraform graph**

* **Use Case:** Generates a graph of the resources and their dependencies. Useful for visualizing the infrastructure.
* **Example:**

bash

Copy code

terraform graph | dot -Tpng > graph.png

**41. terraform output -json**

* **Use Case:** Outputs the values of outputs in JSON format. Useful for integration with other tools and scripts.
* **Example:**

bash

Copy code

terraform output -json

**42. terraform state list**

* **Use Case:** Lists all the resources in the state file. Useful for inspecting the contents of the state file.
* **Example:**

bash

Copy code

terraform state list

**43. terraform state show**

* **Use Case:** Displays detailed information about a resource in the state file. Useful for examining the current state of a specific resource.
* **Example:**

bash

Copy code

terraform state show aws\_instance.example

**44. terraform workspace list**

* **Use Case:** Lists all workspaces associated with your Terraform configuration. Useful for managing and viewing available workspaces.
* **Example:**

bash

Copy code

terraform workspace list

**45. terraform workspace new**

* **Use Case:** Creates a new workspace. Useful for separating different environments or versions of your infrastructure.
* **Example:**

bash

Copy code

terraform workspace new staging

**46. terraform workspace delete**

* **Use Case:** Deletes a workspace. Useful for cleaning up unused or obsolete workspaces.
* **Example:**

bash

Copy code

terraform workspace delete staging

**47. terraform workspace select**

* **Use Case:** Switches to a different workspace. Useful for working in different environments or configurations.
* **Example:**

bash

Copy code

terraform workspace select staging

**48. terraform graph**

* **Use Case:** Generates a dependency graph of the resources defined in your configuration. Helps visualize relationships and dependencies.
* **Example:**

bash

Copy code

terraform graph | dot -Tpng > graph.png

**49. terraform validate**

* **Use Case:** Checks the configuration files for syntax errors and internal consistency. Ensures that configurations are correct before applying.
* **Example:**

bash

Copy code

terraform validate

**50. terraform console**

* **Use Case:** Opens an interactive console for evaluating expressions and querying resources. Useful for debugging and exploration.
* **Example:**

bash

Copy code

terraform console

**51. terraform import**

* **Use Case:** Imports existing infrastructure into Terraform management. Allows Terraform to manage resources created outside of Terraform.
* **Example:**

bash

Copy code

terraform import aws\_instance.example i-12345678

**52. terraform state mv**

* **Use Case:** Moves resources within the state file. Useful for renaming resources or changing resource addresses.
* **Example:**

bash

Copy code

terraform state mv aws\_instance.old aws\_instance.new

**53. terraform state rm**

* **Use Case:** Removes resources from the state file without destroying them. Useful for managing state file content.
* **Example:**

bash

Copy code

terraform state rm aws\_instance.example

**54. terraform state pull**

* **Use Case:** Downloads the latest state from the remote backend. Useful for inspecting the current state file.
* **Example:**

bash

Copy code

terraform state pull

**55. terraform state push**

* **Use Case:** Uploads the local state to the remote backend. Useful for syncing changes made locally.
* **Example:**

bash

Copy code

terraform state push

**56. terraform state list**

* **Use Case:** Lists all the resources tracked in the state file. Useful for inspecting resources managed by Terraform.
* **Example:**

bash

Copy code

terraform state list

**57. terraform state show**

* **Use Case:** Displays detailed information about a specific resource in the state file. Useful for examining resource attributes.
* **Example:**

bash

Copy code

terraform state show aws\_instance.example

**58. terraform providers**

* **Use Case:** Lists all providers used in the configuration along with their versions. Useful for managing and verifying provider dependencies.
* **Example:**

bash

Copy code

terraform providers

**59. terraform version**

* **Use Case:** Displays the current version of Terraform. Useful for verifying the installed version.
* **Example:**

bash

Copy code

terraform version

**60. terraform workspace show**

* **Use Case:** Shows the name of the current workspace. Useful for confirming the workspace context.
* **Example:**

bash

Copy code

terraform workspace show