1. Initialization Commands

`terraform init`

Initializes Terraform and downloads all required providers/plugins.

`terraform init -upgrade`

Upgrades providers and modules to the latest version.

`terraform init -reconfigure`

Reconfigures backend (used when backend settings change).

`terraform init -backend-config="backend.hcl"`

Loads backend config from an external file.

2. Code Validation & Formatting

`terraform fmt`

Formats Terraform code into standard style.

`terraform fmt -recursive`

Formats code inside all subfolders.

`terraform validate`

Checks syntax and identifies configuration errors.

---

3. Plan Commands

`terraform plan`

Shows what Terraform will create/update/destroy (dry run).

`terraform plan -out=plan.tf`

Saves the plan to a file for later use.

`terraform plan -var-file="dev.tfvars"`

Uses variables from a specific file.

`terraform plan -target=aws_instance.web`

Plans actions only for a specific resource.

4. Apply Commands

`terraform apply`

Applies the changes to infrastructure.

`terraform apply -auto-approve`

Applies changes without asking for confirmation.

`terraform apply plan.tf`

Applies a saved plan file.

5. Destroy Commands

`terraform destroy`

Destroys all created resources.

`terraform destroy -auto-approve`

Destroys resources without confirmation.

`terraform destroy -target=aws_s3_bucket.bucket`

Destroys only a specific resource.

6. State File Commands

`terraform state list`

Lists all resources in the state file.

`terraform state show <resource>`

Shows full details of a resource stored in the state.

`terraform state rm <resource>`

Removes a resource from the state file (Terraform stops managing it).

`terraform state mv <old> <new>`

Moves or renames a resource inside the state file.

7. Import Commands

`terraform import <resource> <id>`

Adds an existing AWS resource into Terraform state.
Example:

```
terraform import aws_instance.myec2 i-0123456
```

---

8. Output Commands

`terraform output`

Shows Terraform output values.

`terraform output public_ip`

Displays a specific output.

---

9. Variable Commands

`terraform apply -var="env=dev"`

Passes variable directly through CLI.

`terraform apply -var-file="prod.tfvars"`

Uses full variable file during apply.

---

10. Workspace Commands

`terraform workspace list`

Shows all workspaces.

`terraform workspace new dev`

Creates a new workspace.

`terraform workspace select dev`

Switches to a workspace.

`terraform workspace delete dev`

Deletes a workspace.

---

11. Providers & Modules

`terraform get`

Downloads Terraform modules.

`terraform providers`

Lists all providers in use.

---

12. Dependency Graph

`terraform graph`

Generates a visual dependency graph (used for debugging infra relationships).

---

13. Debugging & Logging

`export TF_LOG=DEBUG`

Enables detailed Terraform logs.

`unset TF_LOG`

Turn off logging.

---

14. Locking & Force Unlock

`terraform force-unlock <lock-id>`

Unlocks state file if Terraform gets stuck (only when safe).

15. Refresh Commands

`terraform refresh` *(older versions)*

Updates the state file with real-world resource values.

`terraform plan -refresh-only`

Shows only drift (changes outside Terraform).

---

16. Replace Resource Commands

`terraform apply -replace="aws_instance.web"`

Destroys and recreates a specific resource.

---

17. Taint Commands (Deprecated but asked in interviews)

`terraform taint aws_instance.web`

Marks a resource to be destroyed & recreated in next apply.

`terraform untaint aws_instance.web`

Removes taint from resource.

---

18. Locking Commands

`terraform apply -lock=false`

Runs apply without taking a state lock (not recommended in teams).

---

19. Graph Commands

`terraform graph`

Generates a dependency graph in DOT format.

---

20. Format Check

`terraform fmt -check`

Checks if files are properly formatted.

---

21. Validate With Warnings

`terraform validate -no-color`

Validates configuration without colored output (for CI/CD pipelines).

---

22. Show Commands

`terraform show`

Displays the current state or plan.

`terraform show -json`

Shows state/plan output in JSON.

---

23. Login Commands (Terraform Cloud)

`terraform login`

Logs into Terraform Cloud.

`terraform logout`

Logs out from Terraform Cloud.

---

24. Provider Schema

`terraform providers schema`

Displays provider schema and arguments.

---

25. Move Commands (Between Modules)

```
terraform state mv module.old.aws_instance.web module.new.aws_instance.web
```

Moves a resource between modules.

---

26. Replace Provider

```
terraform state replace-provider <old> <new>
```

Replaces a provider in the state file.

---

27. Pull & Push State

```
terraform state pull
```

Downloads the latest state file.

```
terraform state push
```

Uploads a state file manually.

---

28. Remote Backend Commands

```
terraform init -migrate-state
```

Migrates local state to a remote backend.

```
terraform init -force-copy
```

Copies existing state even if backend differs.

---

29. Sensitive Output

```
terraform output -raw password
```

Prints sensitive output without quotes.

---

30. Environment Variables

```
export TF_VAR_region=us-east-1
```

Passes a Terraform variable through environment.

```
unset TF_VAR_region
```

Removes an environment variable.

---

31. Destroy Specific Module

```
terraform destroy -target=module.vpc
```

Destroys only the selected module.

---

32. Parallelism Control

```
terraform apply -parallelism=20
```

Controls number of parallel resource operations.

---

33. Disable Input Prompt

```
terraform plan -input=false
```

Stops Terraform from asking for variable input.

---

34. Clean Local Cache

```
rm -rf .terraform/
```

Deletes local Terraform directory and plugin cache.

---

35. Re-download Providers

```
terraform init -upgrade=true
```

Downloads the latest version of providers.

36. Generate Execution Plan in JSON

```
terraform plan -json > plan.json
```

Exports the full plan output in JSON format (used in automation & policy checks).

---

37. Show State in JSON

```
terraform show -json > state.json
```

Converts the Terraform state into JSON format.

---

38. Check Provider Version Constraints

```
terraform providers lock
```

Creates a dependency lock file to ensure consistent provider versions.

---

39. Initialize Provider Mirror

```
terraform providers mirror ./providers
```

Downloads provider binaries locally for offline use or air-gapped environments.

---

40. Format Only Changed Files

```
terraform fmt -diff
```

Shows formatting differences before applying them.

---

41. Debug Provider Plugin Issues

```
export TF_LOG_PROVIDER=TRACE
```

Enables detailed provider-level debug logs.

---

42. Generate Input Variable Documentation

```
terraform providers schema -json > schema.json
```

Outputs provider documentation schema in JSON.

---

43. Convert Modules to Latest Structure

```
terraform mod tidy
```

Cleans unused modules and updates module references.

---

44. Check Mounted Providers in State

```
terraform state replace-provider
```

Used when the provider namespace changes (e.g., from legacy to official).

---

45. Replace All Instances of a Resource

```
terraform apply -replace="aws_security_group.sg"
```

Forces recreation of the entire resource.

---

46. Get All Modules

```
terraform get
```

Downloads all modules referenced in the configuration.

```
terraform get -update
```

Refreshes modules to the latest version.

---

47. Validate Connectivity to Remote Backend

```
terraform init -verify-plugins=false
```

Skips plugin verification (rarely used).

---

48. Show Current Backend Config

`terraform providers`

Prints which providers and backends are being used.

---

49. Check Terraform Version

`terraform version`

Shows Terraform version and installed provider versions.

---

50. Install Specific Terraform Version (tfenv)

(tfenv must be installed)

`tfenv install 1.6.0`

Installs a specific Terraform version.

`tfenv use 1.6.0`

Switches to that version.

---

51. Create Dependency Lock File

`terraform providers lock -platform=linux_amd64`

Locks providers for specific platforms (CI/CD).

---

52. Sync Remote State

`terraform state push`

Pushes your local state to remote backend.

`terraform state pull`

Downloads the latest remote state.

---

53. Remove Orphan Resources

```
terraform state rm <resource>
```

Removes unmanaged or drifted resources from state.

---

54. Export Plan Summary

```
terraform show -json plan.tfplan > plan.json
```

Converts saved plan to JSON.

---

55. Create & Manage Outputs

```
terraform output -state=state.tfstate
```

Reads output values from a specific state file.

---

56. Convert Configuration to JSON

```
terraform show -json > config.json
```

Useful for automation and compliance scanners.

---

57. Apply Without Refreshing State

```
terraform apply -refresh=false
```

Applies changes without checking real cloud state.

---

58. Force Re-download of Backends

```
terraform init -reconfigure
```

Reloads backend and state settings (used after switching S3 bucket/dynamodb).

---

59. Force Store Sensitive Values

```
terraform login
```

Creates a credentials file for Terraform Cloud.

---

60. Dry Run Destroy

```
terraform plan -destroy
```

Shows what will be destroyed (without destroying).

61. Override Parallelism for Planning

```
terraform plan -parallelism=1
```

Runs plan with only 1 parallel operation (useful for debugging race issues).

---

62. Override Parallelism for Apply

```
terraform apply -parallelism=1
```

Slows down apply to debug dependency issues.

---

63. Initialize with Plugin Cleanup

```
terraform init -upgrade -lockfile=readonly
```

Uses existing lockfile only; avoids modifying provider lock file.

---

64. Generate Sensitive Plan

```
terraform plan -input=false -lock=true
```

Prevents prompting for variables during automation (accurate CI runs).

---

65. Force Unlock State

```
terraform force-unlock <LOCK_ID>
```

Releases a locked state manually (when previous apply is stuck).

---

66. Show Lock ID for Debugging

```
terraform show | grep "lock"
```

Helps identify active state locks.

---

67. Compare Two State Files

```
diff <(terraform show -json state1.tfstate) <(terraform
show -json state2.tfstate)
```

Compares two states side-by-side (Linux only).

---

68. Validate Module Structure

```
terraform init -backend=false
```

Initializes only local modules, ignores backend.

---

69. Module Registry Debug

```
terraform providers mirror .
```

Creates a local mirror of all providers (for air-gapped or restricted networks).

---

70. Backup State File Manually

```
cp terraform.tfstate terraform.tfstate.backup
```

Creates a local backup before risky operations.

---

71. Extract Only Resource Addresses

```
terraform state list | sort
```

Lists resources in sorted order.

---

72. Display Terraform Dependency Tree

```
terraform graph | dot -Tsvg > graph.svg
```

Creates a visual dependency map (great for documentation).

---

73. Check Remote Backend Health

```
terraform init -backend=true
```

Forces backend recheck after S3/DynamoDB issues.

---

74. Remove Local Cache of Providers

```
rm -rf .terraform/plugins
```

Deletes cached plugins to force clean provider download.

---

75. Remove All Local Terraform Metadata

```
rm -rf .terraform/ .terraform.lock.hcl
```

Fully wipes local Terraform environment.

---

76. Create a Specified Workspace with Variables

```
terraform workspace new prod --
```

Creates workspace and allows variable overrides.

---

77. Show Resources Not Yet Created

```
terraform plan | grep "+"
```

Shows only resources Terraform will create.

---

78. Show Resources That Will Be Destroyed

```
terraform plan | grep "-"
```

Shows only resources Terraform will delete.

---

79. Show Resources That Will Be Modified

```
terraform plan | grep "~"
```

Shows only resources Terraform will modify.

---

80. Reset Entire Terraform Environment

```
rm -rf .terraform*
```

Resets directory to fresh Terraform environment.

---

81. Generate Policy Check for Terraform Cloud

```
terraform plan -json > plan.json
```

Used for Sentinel & OPA policy checks.

---

82. Disable Remote Backend for Testing

```
terraform init -backend=false
```

Runs Terraform locally without S3/DynamoDB backend.

---

83. Check Missing Required Providers

```
terraform providers schema -json | jq
```

Parses provider schema (advanced automation use).

84. Trigger Only Specific Module Apply

```
terraform apply -target=module.eks
```

Applies only the EKS module.

85. Move State Between Backends Manually

```
terraform state pull > state.json
```

Export state.

```
terraform state push state.json
```

Import state into new backend.

86. Run Terraform With Env File

```
set -a; source dev.env; set +a
```

Loads env variables as Terraform vars.

87. Debug Resource Refresh Behaviour

```
TF_LOG=TRACE terraform refresh
```

Shows detailed refresh logic.

88. Validate Before Apply Script

```
terraform validate && terraform plan
```

Common CI workflow.

---

89. Create Custom Backend Config Inline

```
terraform init -backend-config="bucket=mybucket"
```

Pass backend values from CLI.

---

90. Output Only Specific Value

```
terraform output -raw vpc_id
```

Prints output without formatting.