Terraform CLI Cheat Sheet

Initializing Terraform:

`terraform init` : Initializes a new or existing Terraform configuration by downloading required providers and modules

Managing Providers:

`terraform providers` : Lists the providers used in the configuration

`terraform init -upgrade` : Upgrades provider plugins to the latest versions

Creating and Managing Terraform Resources:

`terraform plan` : Generates an execution plan to show what actions Terraform will take to create, update, or delete resources

`terraform plan --out plan.out` : Generates a plan and saves it to a file named plan.out

`terraform plan --destroy` : Create a plan to destroy all objects, rather than the usual actions

`terraform apply` : Applies the changes defined in the configuration, creating, updating, or deleting resources as necessary

`terraform apply plan.out` : Applies a pre-generated plan from the plan.out file

`terraform apply -auto-approve` : Applies changes automatically without requiring user confirmation

`terraform destroy` : Destroys all resources defined in the configuration

Managing State:

`terraform state list`: Lists all resources in the Terraform state

`terraform state show <resource>` : Displays details about a specific resource in the state

`terraform state rm <resource>` : Removes a resource from the state

Working with Variables:

`terraform apply -var="key=value"` : Set variable values directly in the command line

`terraform apply -var-file="filename.tfvars"` : Load variable values from a variable file

`terraform.tfvars or <name>.auto.tfvars` : Default filenames for variable files that Terraform automatically loads

Managing Workspaces:

`terraform workspace list` : Lists available workspaces `terraform workspace new <name>` : Creates a new workspace

`terraform workspace select <name>` : Switches to a different workspace

`terraform workspace delete <name>` : Deletes a workspace

Outputting Information:

`terraform output` : Displays the values of output variables defined in the configuration

`terraform output <output_name>` : Displays the value of a specific output variable

Importing resources:

`terraform import <resource_type>.<resource_name> <external_id>` : Imports an existing resource into Terraform's state

Config-driven import:

```
import {
    to = <resource_name>
    id = "resource ID"
}
```

`terraform plan -generate-config-out=generated_resources.tf` : Terraform will automatically generate the corresponding HCL configuration for any resource included in an import block that doesn’t have an existing configuration associated with it

Other Useful Commands:

`terraform validate` : Validates the configuration files for syntax and other errors

`terraform fmt` : Automatically formats Terraform configuration files

`terraform fmt --recursive` : Also formats files in subdirectories

`terraform refresh` : Updates the state file with the current real-world resources

`terraform version` : Displays the Terraform version

`terraform graph -type=plan | dot -Tpng -o graph.png` : Visualize your execution plan