** INSTALLATION **

```
# Enable shell autocompletion
terraform -install-autocomplete
```

** INITIALIZING TERRAFORM **

```
# Prepare your working directory for other commands
terraform init

# Upgrade all modules to their latest versions
terraform init -upgrade

# Only download and install modules
terraform get
```

** MAKING INFRASTRUCTURE CHANGES **

```
# Show changes required by the current configuration
terraform plan

# Write the plan to a file to apply it later
terraform plan -out=<file>

# Create a plan for a specific module or resource
terraform plan -target <resource>

# Force the plan to replace a specific resource
terraform plan -replace <resource>

# Set a value for one of the input variables
terraform plan -var '<key>=<value>'

# Inspect resource drift without updating the state file
terraform plan -refresh-only

# Create or update infrastructure
terraform apply

# Create or update infrastructure using a plan file
terraform apply <file>

# Create or update a specific resource
terraform apply -target <resource>

# Force the replacement of a specific resource
terraform apply -replace <resource>

# Skip interactive approval of plan before applying
terraform apply -auto-approve
```

** INSPECTING OUTPUT VALUES **

```
# Show all output values
terraform output

# Show all output values in JSON format
terraform output -json

# Show a specific output value
terraform output <name>
```

```
# Show a specific output value without quotes
terraform output -raw <name>


** MANAGING STATE **

# Update the state to match remote systems
terraform refresh

# Show the current state
terraform show

# Show a saved plan
terraform show <file>

# Show the plan in JSON format
terraform show -json <file>

# List all resources in the state file
terraform state list

# Show details about a resource
terraform state show <resource>

# Rename a resource in the state file
terraform state mv <source> <dest>

# Remove a resource from the state file
terraform state rm <resource>

# Replace provider for resources in the state
terraform state replace-provider <from> <to>

# Import existing infrastructure into Terraform
terraform import <resource> <remote_id>

# Pull current state and output to stdout
terraform state pull


** TAINTING RESOURCES **

# (Deprecated) Mark a resource to be replaced
terraform taint <resource>

# Mark a resource as no longer tainted
terraform untaint <resource>


** DESTROYING INFRASTRUCTURE **

# Destroy infrastructure managed by Terraform
terraform destroy

# Destroy a specific resource
terraform destroy -target <resource>


** FORMATTING AND VALIDATION **

# Check whether the configuration is valid
terraform validate
```

```
# Reformat your configuration in the standard style
terraform fmt

# Check whether the configuration is formatted correctly, return non-zero exit
code if not
terraform fmt -check


** TERRAFORM CLOUD / REMOTE AUTHENTICATION **

# Log in to Terraform Cloud
terraform login

# Log in to a different host
terraform login <hostname>

# Log out of Terraform Cloud
terraform logout

# Log out of a different host
terraform logout <hostname>


** MANAGING WORKSPACES **

# List all existing workspaces
terraform workspace list

# Show the name of the current workspace
terraform workspace show

# Select a different workspace
terraform workspace select <name>

# Create a new workspace
terraform workspace new <name>

# Delete an existing workspace
terraform workspace delete <name>


** OTHER COMMANDS **

# Show the providers required for this configuration
terraform providers

# Release a stuck lock
terraform force-unlock <lock-id>

# Try Terraform expressions at an interactive prompt
terraform console

# Generate a visual graph of Terraform resources
terraform graph | dot -Tpng > graph.png

# Show the current Terraform version
terraform version

# Show help output for Terraform
terraform -help

# Show help output for a specific Terraform command
terraform -help <command>
```

REFERENCING NAMED VALUES

```
# Reference to a managed resource
<RESOURCE_TYPE>.<NAME>

# Reference to an input variable
var.<NAME>

# Reference to a local value
local.<NAME>

# Reference to a child module
module.<MODULE_NAME>

# Reference to a data source
data.<DATA_TYPE>.<NAME>
```

** CONDITIONAL EXPRESSIONS **

```
# If condition is true, return true, otherwise return false
condition ? true : false
```

** SPLAT EXPRESSIONS **

```
# Return a list of values for the given attribute of all instances of a resource
<RESOURCE_TYPE>.<NAME>[*].<ATTRIBUTE>
```

** RESOURCE META-ARGUMENTS **

```
# Explicitly specify resource dependencies
depends_on

# Create multiple instances of a resource
count

# Create an instance of a resource for each element in a map or set
for_each

# Specify a provider configuration block to use for this resource
provider

# Configure the behavior of a resource over its lifetime
lifecycle
```

** LIFECYCLE META-ARGUMENT ATTRIBUTES **

```
# Create the new resource before destroying the old one
create_before_destroy

# Prevent Terraform from destroying the resource
prevent_destroy

# Ignore changes to specific resource attributes
ignore_changes
```