**Terraform Command Lines**

**Terraform CLI tricks**

* **terraform -install-autocomplete** #Setup tab auto-completion, requires logging back in

**Format and Validate Terraform code**

* **terraform fmt** #format code per HCL canonical standard
* **terraform validate** #validate code for syntax
* **terraform validate -backend=false** #validate code skip backend validation

**Initialize your Terraform working directory**

* **terraform init** #initialize directory, pull down providers
* **terraform init -get-plugins=false** #initialize directory, do not download plugins
* **terraform init -verify-plugins=false** #initialize directory, do not verify plugins for Hashicorp signature

**Plan, Deploy and Cleanup Infrastructure**

* **terraform apply --auto-approve** #apply changes without being prompted to enter "yes"
* **terraform destroy --auto-approve** #destroy/cleanup deployment without being prompted for “yes”
* **terraform plan -out plan.out** #output the deployment plan to plan.out
* **terraform apply plan.out** #use the plan.out plan file to deploy infrastructure
* **terraform plan -destroy** #outputs a destroy plan
* **terraform apply -target=aws\_instance.my\_ec2** #only apply/deploy changes to the targeted resource
* **terraform apply -var my\_region\_variable=us-east-1** #pass a variable via command-line while applying a configuration
* **terraform apply -lock=true** #lock the state file so it can't be modified by any other Terraform apply or modification action(possible only where backend allows locking)
* **terraform apply refresh=false** # do not reconcile state file with real-world resources(helpful with large complex deployments for saving deployment time)
* **terraform apply --parallelism=5** #number of simultaneous resource operations
* **terraform refresh** #reconcile the state in Terraform state file with real-world resources
* **terraform providers** #get information about providers used in current configuration

**Terraform Workspaces**

* **terraform workspace new mynewworkspace** #create a new workspace
* **terraform workspace select default** #change to the selected workspace
* **terraform workspace list**#list out all workspaces

**Terraform State Manipulation**

* **terraform state show aws\_instance.my\_ec2** #show details stored in Terraform state for the resource
* **terraform state pull > terraform.tfstate** #download and output terraform state to a file
* **terraform state mv aws\_iam\_role.my\_ssm\_role module.custom\_module** #move a resource tracked via state to different module
* **terraform state replace-provider hashicorp/aws registry.custom.com/aws** #replace an existing provider with another
* **terraform state list** #list out all the resources tracked via the current state file
* **terraform state rm  aws\_instance.myinstace** #unmanage a resource, delete it from Terraform state file

**Terraform Import And Outputs**

* **terraform import aws\_instance.new\_ec2\_instance i-abcd1234** #import EC2 instance with id i-abcd1234 into the Terraform resource named "new\_ec2\_instance" of type "aws\_instance"
* **terraform import 'aws\_instance.new\_ec2\_instance[0]' i-abcd1234** #same as above, imports a real-world resource into an instance of Terraform resource
* **terraform output** #list all outputs as stated in code
* **terraform output instance\_public\_ip** # list out a specific declared output
* **terraform output -json** #list all outputs in JSON format

**Terraform Miscelleneous commands**

* **terraform version** #display Terraform binary version, also warns if version is old
* **terraform get -update=true** #download and update modules in the "root" module.

**Terraform Console(Test out Terraform interpolations)**

* **echo 'join(",",["foo","bar"])' | terraform console** #echo an expression into terraform console and see its expected result as output
* **echo '1 + 5' | terraform console** #Terraform console also has an interactive CLI just enter "terraform console"
* **echo "aws\_instance.my\_ec2.public\_ip" | terraform console** #display the Public IP against the "my\_ec2" Terraform resource as seen in the Terraform state file

**Terraform Graph(Dependency Graphing)**

* **terraform graph | dot -Tpng > graph.png** #produce a PNG diagrams showing relationship and dependencies between Terraform resource in your configuration/code

**Terraform Taint/Untaint(mark/unmark resource for recreation -> delete and then recreate)**

* **terraform taint aws\_instance.my\_ec2**#taints resource to be recreated on next apply
* **terraform untaint aws\_instance.my\_ec2** #Remove taint from a resource
* **terraform force-unlock LOCK\_ID** #forcefully unlock a locked state file, LOCK\_ID provided when locking the State file beforehand

**Terraform Cloud**

* **terraform login** #obtain and save API token for Terraform cloud
* **terraform logout** #Log out of Terraform Cloud, defaults to hostname app.terraform.io