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

1. **`terraform apply --auto-approve`** #apply changes without being prompted to enter “yes”
2. **`terraform destroy --auto-approve`** #destroy/cleanup deployment without being prompted for “yes”
3. **`terraform plan -out plan.out`** #output the deployment plan to plan.out
4. **`terraform apply plan.out`** #use the plan.out plan file to deploy infrastructure
5. **`terraform plan -destroy`** #outputs a destroy plan
6. **`terraform apply -target=aws_instance.my_ec2`** #only apply/deploy changes to the targeted resource
7. **`terraform apply -var my_region_variable=us-east-1`** #pass a variable via command-line while applying a configuration
8. **`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)
9. **`terraform apply refresh=false`** # do not reconcile state file with real-world resources(helpful with large complex deployments for saving deployment time)

10. **`terraform apply --parallelism=5`** #number of simultaneous resource operations
11. **`terraform refresh`** #reconcile the state in Terraform state file with real-world resources
12. **`terraform providers`** #get information about providers used in current configuration

Terraform Workspaces

1. **`terraform workspace new mynewworkspace`** #create a new workspace
2. **`terraform workspace select default`** #change to the selected workspace
3. **`terraform workspace list`** #list out all workspaces

Terraform State Manipulation

1. **`terraform state show aws_instance.my_ec2`** #show details stored in Terraform state for the resource
2. **`terraform state pull > terraform.tfstate`** #download and output terraform state to a file
3. **`terraform state mv aws_iam_role.my_ssm_role module.custom_module`** #move a resource tracked via state to different module

4. **`terraform state replace-provider hashicorp/aws registry.custom.com/aws`** #replace an existing provider with another
5. **`terraform state list`** #list out all the resources tracked via the current state file
6. **`terraform state rm aws_instance.myinstace`** #unmanage a resource, delete it from Terraform state file

Terraform Import And Outputs

1. **`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”
2. **`terraform import 'aws_instance.new_ec2_instance[0]' i-abcd1234`** #same as above, imports a real-world resource into an instance of Terraform resource
3. **`terraform output`** #list all outputs as stated in code
4. **`terraform output instance_public_ip`** # list out a specific declared output
5. **`terraform output -json`** #list all outputs in JSON format

Terraform Miscelleneous commands

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

Terraform Console(Test out Terraform interpolations)

1. **`echo 'join(",",["foo","bar"])' | terraform console`** #echo an expression into terraform console and see its expected result as output
2. **`echo '1 + 5' | terraform console`** #Terraform console also has an interactive CLI just enter "terraform console"
3. **`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)

1. **`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)

1. `terraform taint aws_instance.my_ec2` #taints resource to be recreated on next apply
2. `terraform untaint aws_instance.my_ec2` #Remove taint from a resource
3. `terraform force-unlock LOCK_ID` #forcefully unlock a locked state file, LOCK_ID provided when locking the State file beforehand

Terraform Cloud

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