TERRAFORM SHORT NOTES

- **>>What is Terraform ?**
- **- Terraform is an open-source infrastructure as code software tool that enables you to safely and predictably create, change, and improve infrastructure.**

- **>>What is the use of Terraform ?**
- **- Terraform is an IAC tool, used primarily by DevOps teams to automate various infrastructure tasks. The provisioning of cloud resources, for instance, is one of the main use cases of Terraform. It's a cloud-agnostic, open-source provisioning tool written in the Go language and created by HashiCorp**

>>Terraform basic commands

$terraform

$terraform -h (Help command)

$terraform --help (Help command)

$terraform fmt (This command reformats your configuration in the

standard style)

$terraform init (The init command looks at your configuration files and determines which providers and modules it needs to pu ll down from the registry to allow your configuration to work properly)

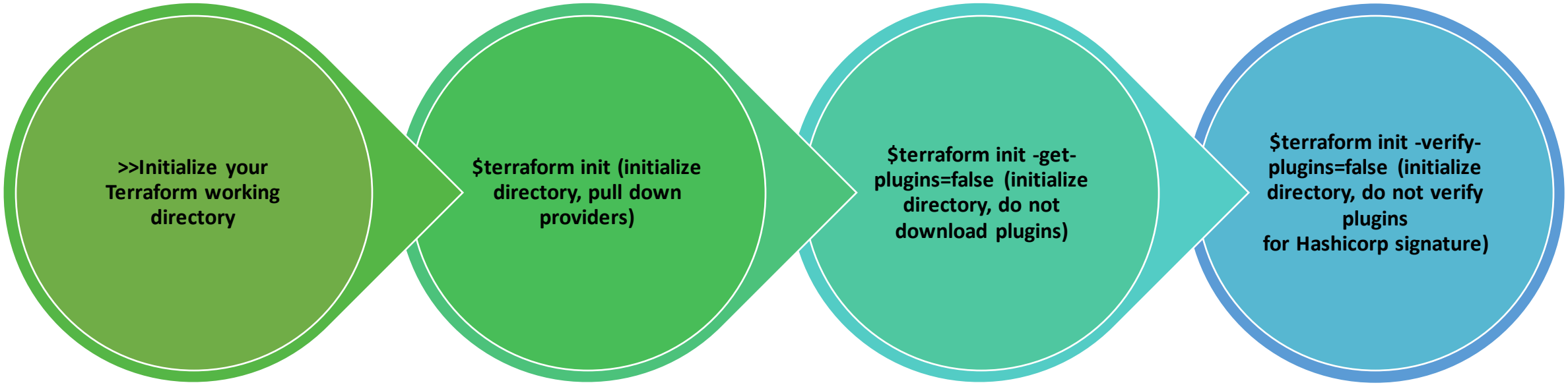
$terraform validate (Validation will catch syntax errors, version errors, and other issues. One thing to note here is that yo u can’t run validate before you run the init command)

$terraform plan (You can even use one of the subcommands with terraform plan to output your plan to apply it later.)

- $terraform apply ((This is the command that deploys or applies your configuration to a provider.))
- $terraform destroy ((The destroy command, obviously, will destroy your infrastructure))
- $terraform output (you can use the output command to make those defined outputs to display certain information)
- $terraform show (The show command shows the current state of a saved plan, providing good information about the infrastructure you've deployed)
- $terraform state (If you use state and then the subcommand list, it'll give you a consolidated list of the resources that are being managed by your configuration.)
- $terraform version (the version command quite a bit to check your Terraform version)

- **>>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 Workspace

$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-prq344s (import EC2 instance with id i-prq344s into the Terraform resource named “new_ec2_instance” of type “aws_instance”)

$terraform import 'aws_instance.new_ec2_instance[0]' i-prq344s (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)