Section 5

The Terraform workflow and Execution plan

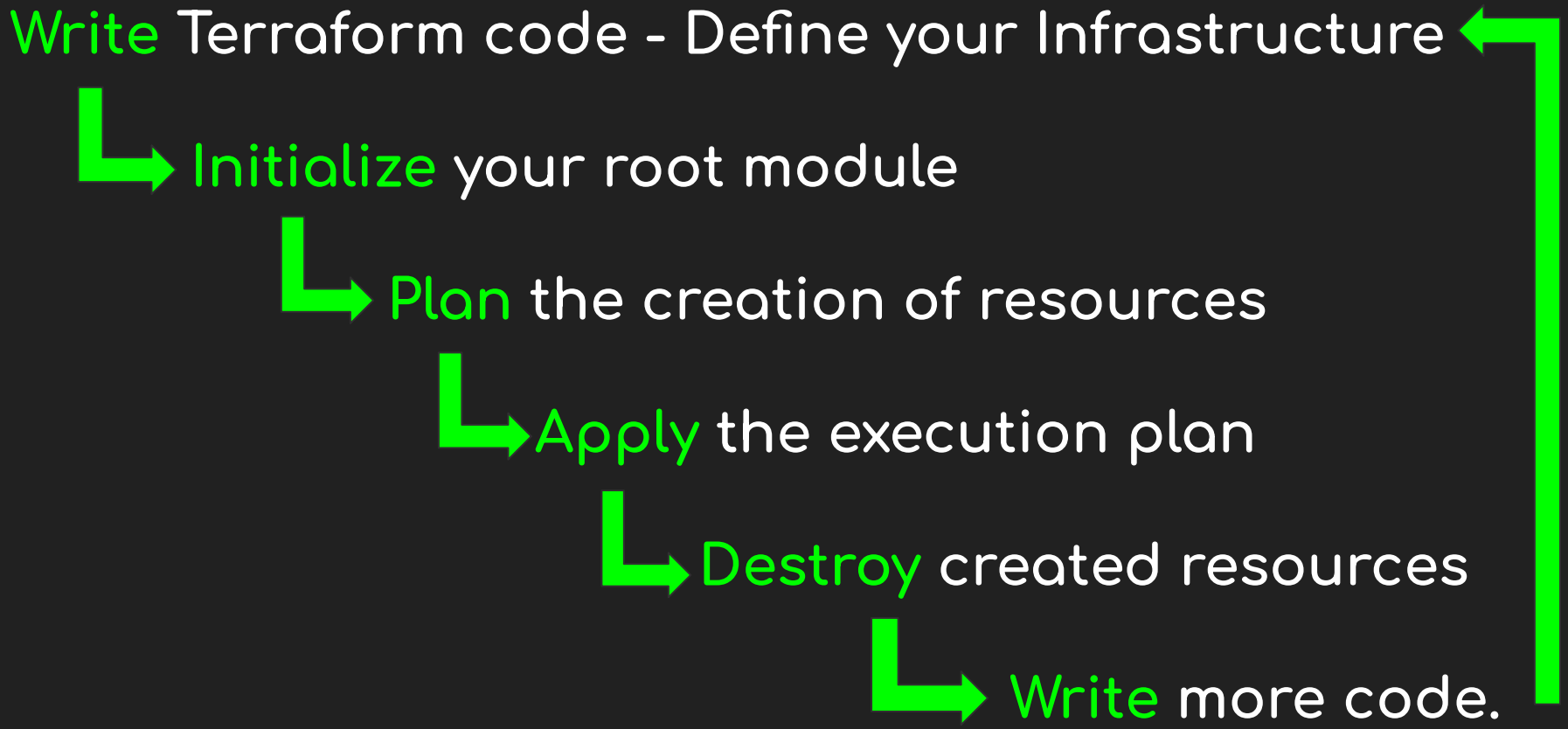
Introduction to the Terraform workflow

The lifecycle of a resource in Terraform is defined by a workflow.

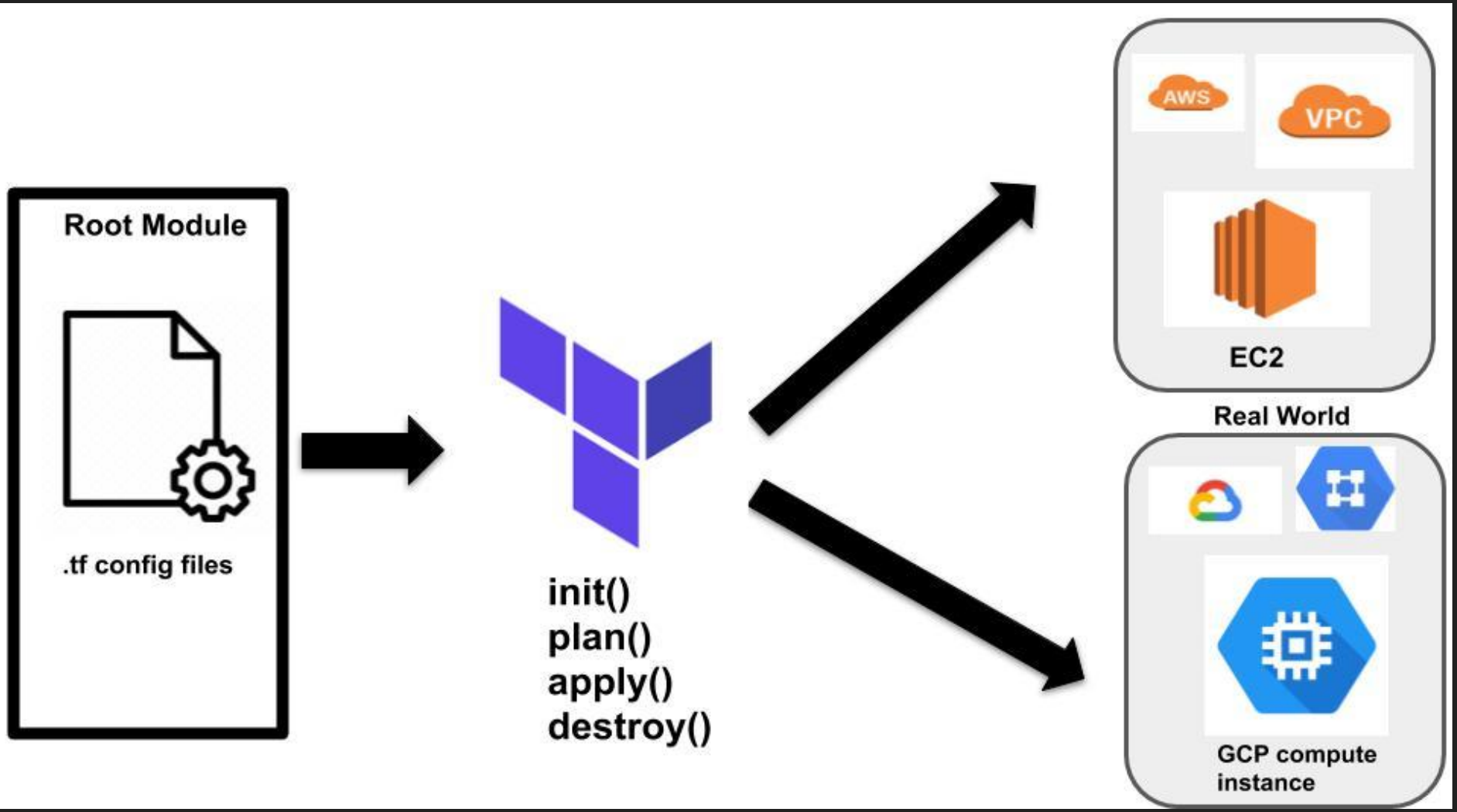
The workflow are all the steps or stages to go from code to created resources in any Infrastructure provider

The workflow for a single Terraform coder:

WRITE ---> PLAN ---> APPLY


Write Terraform code - Define your Infrastructure
Initialize your root module
Plan the creation of resources
Apply the execution plan
Destroy created resources
Write more code.

Writing the code to Terraform compute instances in a multi-cloud infrastructure.


Root Module
.tf config files
init()
plan()
apply()
destroy()
AWS
VPC
EC2
Real World
GCP compute instance

Notes:

With AWS default VPC should be created. Please create one with the following aws cli.
$ aws ec2 create-default-vpc

we will use the default vpc and we will explore creating a custom vpc and subnet at the end of this lecture.

credentials should be stored as env variables or using the cloud provider cli

Initialize your Terraform root module

Now that we have created our code and organized in terraform configuration files with provider.tf and main.tf, we initialize the root module to download the provider plugins using:

$ terraform init

In this case terraform will download the aws plugin and create a hidden directory to store all the plugin details under .terraform/plugins

$ terraform init

Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "aws" (hashicorp/aws) 2.70.0...

Terraform has been successfully initialized!

Validating or Linting your Terraform config

It is recommended to lint your Terraform configuration in order to check for any syntax or mistakes done whole writing the Terraform code before moving to further stages like plan and apply

The Terraform validate will check your configuration for mistakes by parsing your code. It is not verifying with any remote provider or using the API.

This can save a lot of time and trouble...

The Terraform Execution plan

Desired end state -> what should the infra look like as defined in the TF configuration.

Real world or current state -> how the infra “actually” looks like!

Terraform always diff() the 2 using the state file in order to create execution plan!

The $Terraform plan command generates the execution plan

Can be used for review, approval or audit purpose.

Terraform refresh is done automatically when using the terraform plan command when a state file exists to ensure that the state file capture the latest real world state.

Execution plan actions to be taken to achieve desired end state are marked with these symbols:

```
 +  create
 -  destroy
-/+ destroy and recreate  (taint)
 ~   update in place (without destroying)
<=   read (rare)
```

An -out option allows to save the plan output as a file. This can be used with CICD pipelines or for auditing purposes.

A warning will be displayed but this is not a problem

$ Terraform plan -out=”path”

Applying the Terraform execution plan

The Terraform apply command.

It is apply and not create or provision because it applies the actions or required changes defined in the execution plan to meet the configuration desired end state..could be create or update or destroy or recreate

A input will be asked to verify but can be overridden using -input=false or -auto-approve flags. These are required with automation and CICD pipelines.

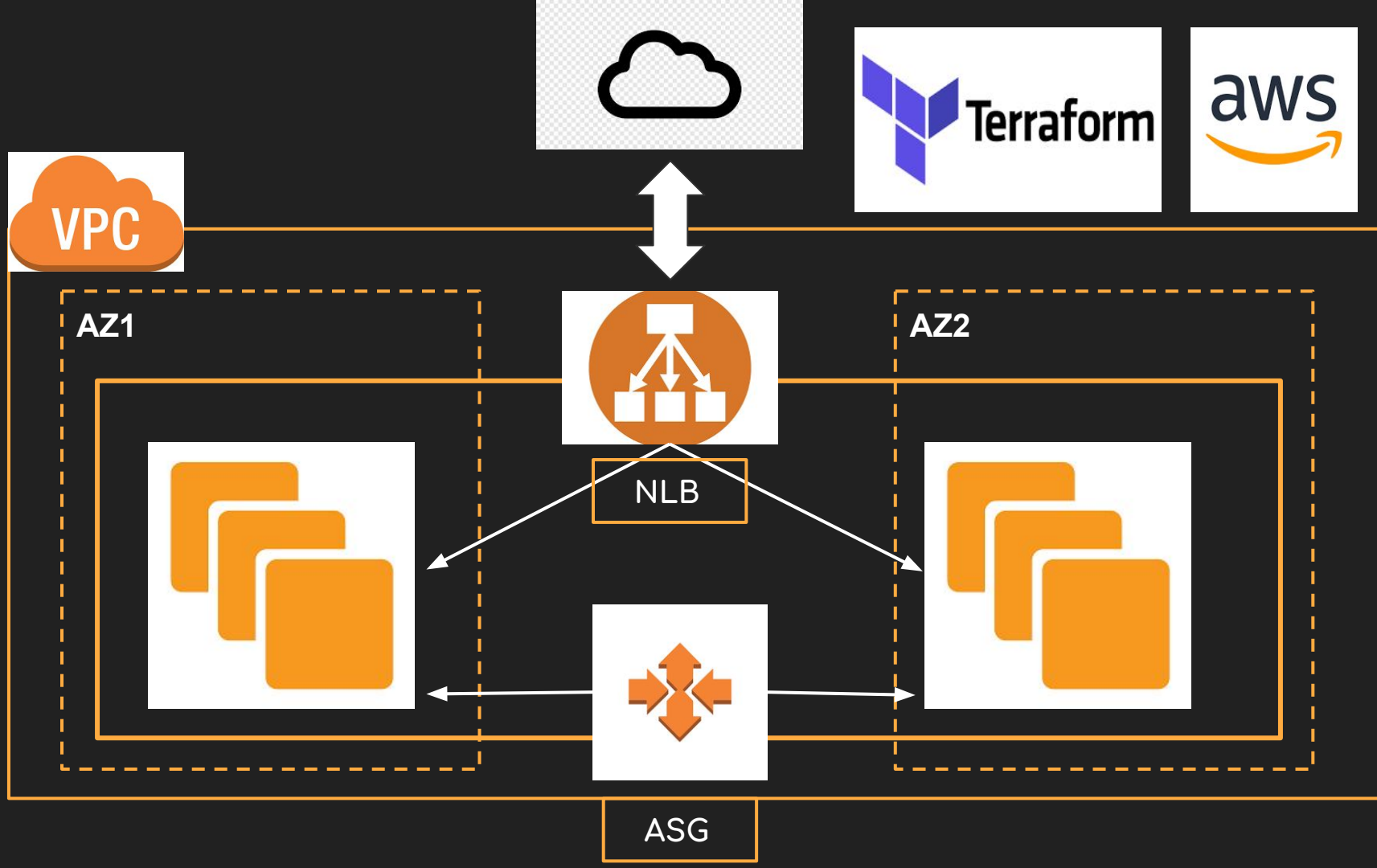
Destroying your Terraform managed real world infrastructure

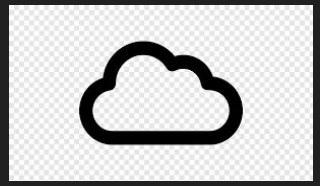
The terraform destroy command will destroy all the infrastructure managed by Terraform in a provider.

An approval step is there but can be bypassed when using CICD and automation with -auto-approve

The destroy execution plan can be reviewed with:
$terraform plan -destroy

LAB - How to Terraform a load balanced autoscaled web application in a multi-cloud environment with AWS and GCP - Part1


Terraform
aws
VPC
AZ1
AZ2
NLB
ASG



internet

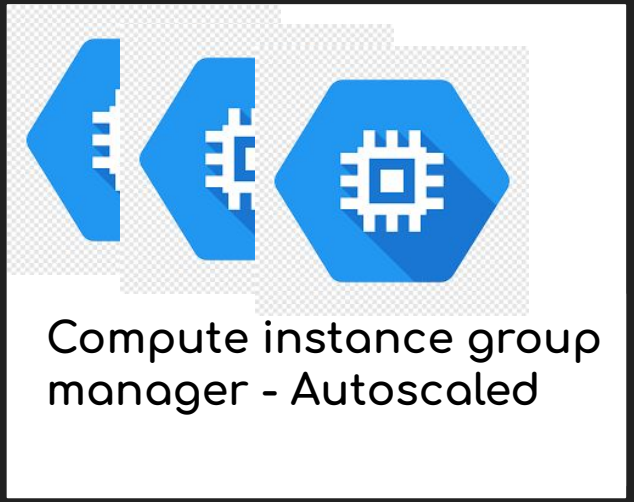
Terraform

API

Google Cloud Platform

VPC Network

Forwarding rule
Cloud Load balancer



Compute instance group manager - Autoscaled

you can reference another resource's attributes in an expression (represents the syntax of a value) using:

<resource_type>.<name>.<attribute>

Attributes are infrastructure provider object data like id or public ip or dns name....

Useful in the case that an attribute is known only after creation of the resource for example..

LAB - How to Terraform a load balanced autoscaled web application in a multi-cloud environment with AWS and GCP - Part2

LAB - How to Terraform a load balanced autoscaled web application in a multi-cloud environment with AWS and GCP - Part3