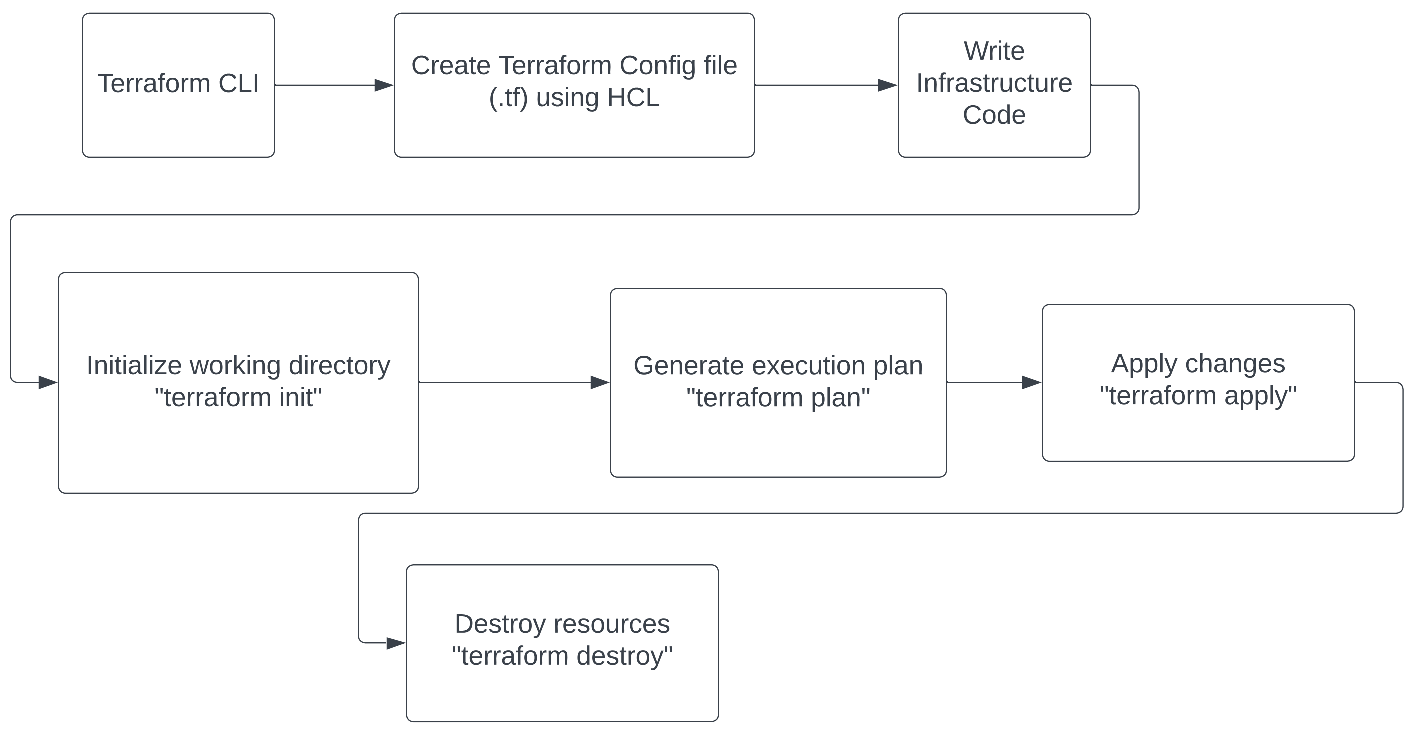
Terraform - Sean Corzo 8/12/2023

Terraform is an open-source infrastructure as code (IaC) tool that allows you to define and manage your infrastructure in a declarative way.


Terraform CLI
Create Terraform Config file (.tf) using HCL
Write Infrastructure Code
Initialize working directory "terraform init"
Generate execution plan "terraform plan"
Apply changes "terraform apply"
Destroy resources "terraform destroy"

EXAMPLE INSTRUCTION CODE

```
provider "aws" {
  region = "us-west-1"
}

resource "aws_instance" "example_server" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"
}
```

FOUNDATIONAL CONCEPTS

Providers:
At the beginning of your configuration file, you specify the provider you'll be working with (e.g., AWS, Azure, Google Cloud). Providers define the cloud platform where your resources will be created. You configure provider-specific settings such as access credentials and regions.
Example:

```
provider "aws" {   region = "us-west-1" }
```

Resources:
Resources are the fundamental building blocks of your infrastructure. They represent the various components you want to create, such as virtual machines, networks, databases, security groups, etc. Each resource has a resource type and a unique name within your configuration.

Example:

```
resource "aws_instance" "example_server" {   ami
= "ami-0c55b159cbfafe1f0"   instance_type = "t2.micro" }
```

In this example, an EC2 instance named "example_server" will be created using the specified Amazon Machine Image (AMI) and instance type.

Attributes and Arguments:
Resources have attributes and arguments that define their properties. Attributes provide information about a resource that you can reference in other parts of your configuration. Arguments are used to configure the properties of the resource.

Example:

```
resource "aws_instance" "example_server" {   ami
= "ami-0c55b159cbfafe1f0"   instance_type = "t2.micro"
tags = {     Name = "ExampleServer"   } }
```

In this example, the "tags" attribute is set using the map to assign a name to the created instance.

Variables and Input:
You can define variables to parameterize your configuration and make it more reusable. Variables allow you to input values from external sources or pass values between different parts of your configuration.
Example:

```
variable "instance_ami" {   description = "The ID of the
AMI for the instance" }  resource "aws_instance"
"example_server" {   ami           = var.instance_ami
instance_type = "t2.micro" }
```

Modules:
Modules are a way to organize and encapsulate parts of your configuration for reuse. You can create custom modules or use existing ones from the Terraform Module Registry to abstract and modularize your infrastructure code.
Example:

```
module "example_instance" {   source      =
"./modules/example_instance"   instance_ami =
"ami-0c55b159cbfafe1f0" }
```

Output Values:
Output blocks allow you to define values that will be displayed to the user after applying the configuration. These values can be useful to communicate information about the created resources or data that other parts of your infrastructure might depend on.
Example:

```
output "instance_ip" {   value =
aws_instance.example_server.private_ip }
```

COMMON RESOURCE TYPE EXAMPLES (GCP)

Virtual Machine (Compute Engine Instance on GCP):

```
resource "google_compute_instance" "example_instance" {
  name         = "example-instance"
  machine_type = "n1-standard-1"
  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-9"
    }
  }
  network_interface {
    network = "default"
  }
}
```

Virtual Network (VPC on GCP):

```
resource "google_compute_network" "example_network" {
  name = "example-network"
}
```

Subnet (GCP Subnet):

```
resource "google_compute_subnetwork" "example_subnet" {
  name          = "example-subnet"
  ip_cidr_range = "10.0.1.0/24"
  network       = google_compute_network.example_network.id
}
```

Firewall Rule (GCP Firewall):

```
resource "google_compute_firewall" "example_firewall" {
  name    = "example-firewall"
  network = google_compute_network.example_network.name
  allow {
    protocol = "tcp"
    ports    = ["80"]
  }
  source_ranges = ["0.0.0.0/0"]
}
```

Database Instance (Cloud SQL - MySQL on GCP):

```
resource "google_sql_database_instance" "example_db" {
  name             = "example-db"
  database_version = "MYSQL_5_7"
  settings {
    tier = "db-n1-standard-1"
    backup_configuration {
      enabled = true
    }
    database_flags {
      name  = "slow_query_log"
      value = "off"
    }
    ip_configuration {
      ipv4_enabled = true
    }
  }
}
```