Getting Started with AWS VPC, Subnets, and EC2 Instances Using Terraform

Introduction:

Amazon Web Services (AWS) provides a robust platform for hosting cloud-based applications and services. Understanding key AWS concepts, such as Virtual Private Cloud (VPC), subnets, and EC2 instances, is essential for building scalable and secure cloud infrastructure. In this tutorial, we'll explore how to leverage Terraform, an Infrastructure as Code tool, to provision AWS resources, configure networking components, and deploy EC2 instances within a VPC.

Prerequisites:

* An AWS account
* Terraform installed on your local machine
* Basic understanding of AWS services and Terraform concepts

1. Overview of AWS Concepts:

* Virtual Private Cloud (VPC): A virtual network dedicated to your AWS account. It allows you to define your virtual network topology, including IP address ranges, subnets, route tables, and gateways.
* Subnet: A range of IP addresses within a VPC. Subnets can be public (accessible from the internet) or private (accessible only from within the VPC).
* Availability Zone (AZ): Distinct locations within a region. Each availability zone is isolated but interconnected through low-latency links, providing high availability and fault tolerance.
* Amazon Machine Image (AMI): A pre-configured template for launching EC2 instances. AMIs contain the operating system, software, and configurations required for the instance.
* CIDR Block: Classless Inter-Domain Routing block is a range of IP addresses specified in CIDR notation. It defines the address range for your VPC and subnets.
* Region: A geographical area containing multiple availability zones. AWS resources are region-specific, and selecting the right region can optimize latency and compliance requirements.

2. Terraform Configuration:

a. Provider Configuration:

provider "aws" {

region = "us-east-2"

}

This section configures the AWS provider with the specified region. You can replace "us-east-1" with the AWS region of your choice.

b. VPC Resource:

resource "aws\_vpc" "example\_vpc" {

cidr\_block = "10.0.0.0/16"

}

Defines a VPC resource with the CIDR block 10.0.0.0/16.

c. Subnet Resources:

resource "aws\_subnet" "example\_subnet\_1" {

vpc\_id = aws\_vpc.example\_vpc.id

cidr\_block = "10.0.1.0/24"

availability\_zone = "us-east-2a"

}

resource "aws\_subnet" "example\_subnet\_2" {

vpc\_id = aws\_vpc.example\_vpc.id

cidr\_block = "10.0.2.0/24"

availability\_zone = "us-east-2b"

}

Creates two subnets within the VPC, each in a different availability zone (us-east-2a and us-east-2b).

d. EC2 Instance Resources:

resource "aws\_instance" "example\_instance\_1" {

ami = "ami-xxxxxxxx"

instance\_type = "t2.micro"

subnet\_id = aws\_subnet.example\_subnet\_1.id

}

resource "aws\_instance" "example\_instance\_2" {

ami = "ami-xxxxxxxx"

instance\_type = "t2.micro"

subnet\_id = aws\_subnet.example\_subnet\_2.id

}

Launches two EC2 instances within the created subnets, specifying the AMI ID and instance type.

3. Running Terraform Commands:

Before running Terraform commands, ensure you have configured your AWS access key ID and secret access key using the aws configure command.

aws configure

Then, initialize the Terraform configuration, view the execution plan, and apply the changes.

terraform init

terraform plan

terraform apply

Conclusion:

In this tutorial, you've learned how to use Terraform to provision AWS resources, including VPC, subnets, and EC2 instances. By understanding the concepts of AWS networking and Terraform basics, you can efficiently manage and deploy infrastructure in the cloud. Experiment with different configurations to customize your AWS environment according to your application requirements. Below is the complete Terraform configuration:

provider "aws" {

region = "us-east-2"

}

resource "aws\_vpc" "example\_vpc" {

cidr\_block = "10.0.0.0/16"

}

resource "aws\_subnet" "example\_subnet\_1" {

vpc\_id = aws\_vpc.example\_vpc.id

cidr\_block = "10.0.1.0/24"

availability\_zone = "us-east-2a"

}

resource "aws\_subnet" "example\_subnet\_2" {

vpc\_id = aws\_vpc.example\_vpc.id

cidr\_block = "10.0.2.0/24"

availability\_zone = "us-east-2b"

}

resource "aws\_instance" "example\_instance\_1" {

ami = "ami-xxxxxxxx"

instance\_type = "t2.micro"

subnet\_id = aws\_subnet.example\_subnet\_1.id

}

resource "aws\_instance" "example\_instance\_2" {

ami = "ami-xxxxxxxx"

instance\_type = "t2.micro"

subnet\_id = aws\_subnet.example\_subnet\_2.id

}