단일 서버 배포

단일 서버 배포

테라폼 코드는 확장자가 .tf인 해시코프 구성 언어(HasiCorp Configuration Language, HCL)로 작성된다.

해시코프 언어는 선언적 언어이므로 원하는 인프라를 설명하기 위해 코드를 작성하고, 테라폼이 인프라를 만드는 방법을 알아낸다.

```
provider "aws" {
  region = "ap-northeast-2"
}
```

각 유형의 공급자마다 서버, 데이터베이스 및 로드 밸런서와 같이 우리가 작성할 수 있는 다양한 종류의 리소스가 있고 테라폼에서 리소스를 생성할 때는 다음과 같은 구문을 사용한다.

```
resource "<PROVIDER>_<TYPE>" "<NAME>" {
  [CONFIG ...]
}
```

- PROVIDER : 아마존 웹 서비스처럼 공급자의 이름
- TYPE : instance와 같이 생성하고자 하는 리소스의 종류
- NAME : 테라폼 코드에서 해당 리소스를 지칭하는 식별자
- CONFIG : 해당 리소스에 선언할 수 있는 하나 이상의 설정 변숫값들로 구성

예를 들어 EC2 인스턴스라고 하는 AWS에 단일 가상 서버를 배포하려면 다음과 같이 main.tf의 aws_instance 리소스를 사용한다.

```
terraform {
  # Terraform 버전 지정
  required_version = ">= 1.0.0, < 2.0.0"
```

```
  # 공급자 지정
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.0"
    }
  }
}

provider "aws" {
  region = "ap-northeast-2"
}

resource "aws_instance" "example" {
  ami           = "ami-06eea3cd85e2db8ce" # Ubuntu 20.04
  instance_type = "t2.micro"
}
```

- **AMI(Amazon Machine Image)**

EC2 인스턴스를 구동시키는 골드 이미지, AWS 인스턴스 생성에서 instance_type에 맞는 id를 확인할 수 있다.

- **instance_type**

EC2 인스턴스를 구동시키기 위한 필수 정보. 각 EC2 인스턴스 타입은 각기 다른 CPU, 메모리, 디스크 용량, 네트워크 수용량을 갖고 있으며, 아마존 웹 서비스 페이지에서 목록을 확인 할 수 있다.

Terraform 실행

```
$ terraform init
Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Using hashicorp/aws v4.3.0 from the shared cache directory

....

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

...
```

테라폼 바이너리에는 테라폼의 기본 기능이 포함되어 있지만 모든 공급자(예를 들어 AWS, Azure, Google 등)에 대한 코드가 포함되어 있지 않다.

테라폼을 처음 사용한다면 terraform init 명령어를 실행하여 테라폼에 코드를 스캔하도록 지시하고, 어느 공급자인지 확인하고, 필요한 코드를 다운로드하도록 해야 한다.

기본적으로 공급자 코드는 테람폼의 .terraform 폴더에 다운로드 된다.

terraform plan을 사용하면 무언가를 실제로 변경하기 전에 테라폼이 수행할 작업을 확인할 수 있다.

이것은 실제 운영 환경에 적용하기 전에 코드의 온전성을 검사 할 수 있는 방법이다.

plan 명령어의 결괏값은 “+”가 있는 항목은 추가되고, “-”가 있는 항목은 삭제 된다는 뜻이다.

“~”가 있는 항목은 수정된다.

```
$ terraform plan
Terraform used the selected providers to generate the following
execution plan. Resource actions are indicated with the following
symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.example will be created
  + resource "aws_instance" "example" {
      + ami                                  = "ami-06eea3cd85e2db8ce"
      + arn                                  = (known after apply)
      ....
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      ....

Plan: 1 to add, 0 to change, 0 to destroy.
....
```

terraform apply 명령어는 실제로 코드를 수행한다.

```
$ terraform apply
Terraform used the selected providers to generate the following
execution plan. Resource actions are indicated with the following
symbols:
  + create

Terraform will perform the following actions:
```

```
  # aws_instance.example will be created
  + resource "aws_instance" "example" {
      + ami                                  = "ami-06eea3cd85e2db8ce"
....

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

aws_instance 리소스의 tags에 Name이라는 태그로 이름을 추가한다.

```
provider "aws" {
  region = "ap-northeast-2"
}

resource "aws_instance" "example" {
  ami           = "ami-06eea3cd85e2db8ce"
  instance_type = "t2.micro"

  # tags 추가
  tags = {
    Name = "example"
  }
}
```