구성 가능한 웹서버 배포

보안 그룹과 사용자 데이터 구성을 보면 웹 서버 코드에 8080 포트가 중복되어 있는 것을 알 수 있다.

이는 DRY(Dont's Repeat Yourself)원칙, 즉 반복하지 말라는 원칙을 위반한것이다.

코드 내의 모든 지식은 유일하고, 모호하지 않으며, 믿을 만한 형태로 존재해야 한다. 두 곳에 포트 번호가 있는 경우 한 곳에서 포트 번호를 변경하면 다른 곳도 같이 변경해야 한다는 사실을 잊어버리기 쉽다.

웹 서버 예제의 경우 포트 번호를 지정하는 변수만 있으면 된다.

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

resource "aws_instance" "example" {
  ami           = "ami-07d16c043aa8e5153"
  instance_type = "t2.micro"
  vpc_security_group_ids = [aws_security_group.instance.id]

  user_data = <<-EOF
              #!/bin/bash
              echo "Hello, World" > index.html
              nohup busybox httpd -f -p ${var.server_port} &
              EOF
  # tags 추가
  tags = {
    Name = "example"
  }
}

resource "aws_security_group" "instance" {
  name = "terraform-example-instance"

  ingress {
    from_port   = var.server_port
    to_port     = var.server_port
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

variable "server_port" {
  description = "The port will use for HTTP requests"
  type        = number
}
```

server_port 입력 변수에 default가 없으므로 apply 명령어를 실행하면 테라폼이 server_port에 값을 입력하라는 메시지를 대화식으로 표시하고 변수에 대한 설명을 출력한다.

```
$ terraform plan                                                                [2:23:13]
var.server_port
  The port will use for HTTP requests

  Enter a value: 8080
(...)
```

대화식으로 명령어를 처리하지 않으려면 명령 줄의 -var 옵션으로 변수 값을 제공할 수 있다.

```
$ terraform plan -var "server_port=8080"
```

TF_VAR_<name>이라는 환경 변수를 통해 변수를 설정할 수도 있다. 여기서 <name>은 설정하려는 변수의 이름이다.

```
$ export TF_VAR_server_port=8080
$ terraform plan
```

default 값을 지정해두면 plan이나 apply 명령어를 실행할 때마다 명령 줄 인수를 일일이 기억해서 처리하지 않아도 된다.

```
variable "server_port" {
  description = "The port will use for HTTP requests"
  type        = number
  default     = 8080
}
```

테라폼 코드에서 입력 변수의 값을 사용하려면 변수 참조라는 새로운 유형의 표현식을 사용할 수 있다.

var.<VARIABLE_NAME>

예를 들어, 보안 그룹의 from_port 및 to_port 매개 변수를 server_port 변수의 값으로 설정하는 방법은 아래와 같다.

```
resource "aws_security_group" "instance" {
  name = var.security_group_name

  ingress {
    from_port   = var.server_port
    to_port     = var.server_port
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

또한 사용자 데이터 스크립트에서 포트를 설정할 때도 동일한 변수를 사용하는것이 좋다. 문자열 리터럴 내에서 참조를 사용하려면 보간이라는 새로운 유형의 표현식을 사용해야 한다.

“${..}”

중괄호 안에 참조를 넣을 수 있으며 테라폼은 이를 문자열로 변환한다.

```
user_data = <<-EOF
              #!/bin/bash
              echo "Hello, World" > index.html
              nohup busybox httpd -f -p ${var.server_port} &
              EOF
```

다음 구문에서처럼 테라폼에서는 입력 변수뿐만 아니라 출력 변수도 정의할 수 있다.

```
output "<NAME>" {
  value = <VALUE>
  [CONFIG ...]
}
```

- **NAME :** 출력 변수 이름
- **VALUE :** 출력하려는 테라폼 표현식
- **CONFIG**

  description : 출력 변수에 어떤 유형의 데이터가 포함되어 있는지 알려준다

  sensitive : terraform apply 실행이 끝날 때 출력을 기록하지 않도록 테라폼에 지시하려면 sensitive 매개 변수를 true로 설정한다.

예를 들어 서버의 IP 주소를 찾기 위해 EC2 콘솔을 수동으로 조회하는 대신 IP 주소를 출력 변수로 제공할 수 있다.

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

resource "aws_instance" "example" {
  ami           = "ami-07d16c043aa8e5153"
  instance_type = "t2.micro"
  vpc_security_group_ids = [aws_security_group.instance.id]

  user_data = <<-EOF
              #!/bin/bash
              echo "Hello, World" > index.html
              nohup busybox httpd -f -p ${var.server_port} &
               EOF
  # tags 추가
  tags = {
    Name = "example"
  }
}

resource "aws_security_group" "instance" {
  name = "terraform-example-instance"

  ingress {
    from_port   = var.server_port
    to_port     = var.server_port
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

variable "server_port" {
  description = "The port will use for HTTP requests"
  type        = number
  default     = 8080
}

output "public_ip" {
  value       = aws_instance.example.public_ip
  description = "The public IP of the Instance"
}
```

```
$ terraform output
public_ip = 54.174.13.5
```