변수 지정

테라폼은 입력 변수를 정의하게 하므로 코드가 종복되지 않고 구성을 관리하기도 쉽다. 변수를 선언하는 구문은 다음과 같다.

```
variable "NAME" {
  [CONFIG ...]
}
```

변수 선언에 필요한 매개 변수

- **description**

  변수 사용방법을 문서화하려면 이 매개 변수를 사용한다. 팀원은 코드를 읽을 때뿐만 아니라 plan 또는 apply 명령어를 실행할 때 이 설명을 볼 수 있다.

- **default**

  변수에 값을 전달하는 여러 가지 방법이 있다. 명령 줄(-var 옵션 사용)로 보내거나 파일(-var-file) 또는 환경 변수(테라폼은 이름이 'TF_VAR_<variable_name>'인 환경 변수를 찾는다)를 통해 변수에 값을 전달할 수 있다. 만약 값이 전달되지 않으면 기본값이 전달된다. 기본값이 없으면 테라폼은 대화식으로 사용자에게 변수에 관한 정보를 묻는다.

- **type**

  유형 제약 조건으로 사용자가 전달하는 변수의 유형을 지정할 수 있다. string, number, bool, list, map, set, object, tuple 등의 제약조건이 있다.

main.tf

```
provider "local" {
}

output "number_example" {
  value = var.number_example
}
```

variable.tf

```
# 전달할 값이 number인지 확인
variable "number_example" {
  description = "An example of a number variable in Terraform"
  type        = number
  default     = 42
}
```

실행 결과

```
$ terraform apply
...
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

number_example = 42
```

```
# 전달할 값이 string인지 확인
variable "security_group_name" {
  description = "The name of the security group"
  type        = string
  default     = "terraform-example-instance"
}
```

```
# 전달할 값이 list인지 확인
variable "list_example" {
  description = "An example of a list in Terraform"
  type        = list
  default     = ["a", "b", "c"]
}
```

```
# 제약조건의 결합 사용, 리스트의 모든 항목이 number인 list
variable "list_numeric_example" {
  description = "An example of a numeric list in Terraform"
  type        = list(number)
  default     = [1, 2, 3]
}
```

```
# 전달할 값이 string인 map의 예
variable "map_example" {
  description = "An example of a map in Terraform"
  type        = map(string)

  default = {
    key1 = "value1"
    key2 = "value2"
    key3 = "value3"
  }
}
```

```
# object 또는 tuple 제약 조건을 사용
variable "object_example" {
  description = "An example of a structural type in Terraform"
  type        = object({
    name    = string
    age     = number
    tags    = list(string)
    enabled = bool
  })

  default = {
    name    = "value1"
    age     = 42
    tags    = ["a", "b", "c"]
    enabled = true
  }
}
```

만약 변수에 유형이 일치하지 않는 값을 설정하려고 하면 테라폼은 즉시 유형 오류를 표시한다. 다음은 enabled에 불리언이 아닌 문자열을 설정하려는 경우이다.

```
variable "object_example_with_error" {
  description = "An example of a structural type in Terraform with error"
  type        = object({
    name    = string
    age     = number
    tags    = list(string)
    enabled = bool
  })

  default = {
    name    = "value1"
    age     = 42
    tags    = ["a", "b", "c"]
    enabled = "invalid"
  }
}
```

웹 서버 예제의 경우 포트 번호를 지정하는 변수만 있으면 된다.

```
variable "server_port" {
  description = "The port the server 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 apply                                                                [14:02:44]
var.server_port
  The port the will use for HTTP requests

  Enter a value:
```

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

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

TF_VAR_<name>이라는 환경변수를 통해 변수를 설정할 수 도 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 the server will use for HTTP requests"
  type        = number
  default     = 8080
}
```

테라폼 코드에서 입력 변수의 값을 사용하려면 변수 참조(variable reference)라는 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 다음과 같다.

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

  ingress {
    from_port   = var.server_port
    to_port     = var.server_port
    protocol    = "tcp"
```

```
    cide_blocks = ["0.0.0.0/0"]
  }
}
```

또한 사용자 데이터 스크립터에서 포트를 설정할 때도 동일한 변수를 사용하는 것이 좋다. 문자열 리터럴 내에서 참조를 사용하려면 보간이라는 유형의 표현식을 사용해야 한다.

```
"${...}"
```

중괄호 안에 참조를 넣을 수 있으며 테라폼은 이를 문자열로 변환한다. 예를 들어, 사용자 데이터 문자열 내에서 var.server_port를 사용하는 방법은 다음과 같다.

```
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 변수의 이름

VALUE : 출력하려는 테라폼 표현식

CONFIG : 다음의 2가지 선택적 매개 변수를 추가로 포함할 수 있다.

- **description**

출력 변수에 어떤 유형의 데이터가 포함되어 있는지를 알려준다.

- **sensitive**

terraform apply 실행이 끝날 때 출력을 기록하지 않도록 테라폼에 지시하려면 sensitive 매개 변수를 true로 설정한다. 이는 출력 변수에 패스워드나 개인 키와 민감한 자료 또는 시크릿이 포함되어 있는 경우에 유용하다.

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

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

이 코드는 속성 참조를 다시 사용하는데, 이번에는 aws_instance 리소스의 public_ip 속성을 참조한다.

terraform output 명령어를 사용하여 변경 사항을 적용하지 않고도 모든 결괏값을 나열할 수 도 있다.

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

terraform output <OUTPUT_NAME>을 실행하여 <OUTPUT_NAME>이라는 특정 변수의 값을 확인할 수 있다.

```
$ terraform output public_ip
54.174.13.5
```