반복문 for_each

for_each 표현식을 사용하면 리스트, 집합, 맵을 사용하여 전체 리소스의 여러 복사본 또는 리소스 내 인라인 블록의 여러 복사본을 생성할 수 있다.

```
resource "<PROVIDER>_<TYPE>" "NAME" {
  for_each = <COLLECTION>

  [CONFIG...]
}
```

PROVIDER : aws 같은 공급자의 이름

TYPE : instance와 같이 해당 공급자에서 생성한 리소스의 유형

NAME : my_instance와 같이 테라폼 코드 전체에서 이 리소스를 참조하기 위해 사용할 식별자

COLLECTION : 루프를 처리할 집합 또는 맵, for_each는 리스트를 지원하지 않는다.

CONFIG : 해당 리소스와 관련된 하나 이상의 인수, each.key 또는 each.value를 사용하여 COLLECTION에서 현재 항목의 키와 값에 접근할 수 있다.

```
resource "aws_iam_user" "example" {
  for_each = toset(var.user_names)
  name     = each.value
}
```

var.user_names 리스트를 집합(set)으로 변환하기 위해 toset을 사용한다. for_each는 리소스에서 사용될 때만 집합과 맵을 지원한다. for_each가 이 집합을 반복하면 each.value에서 각 사용자 이름을 사용할 수 있다. 일반적으로 each.key는 키/값 쌍의 맵에만 사용 가능하지만, 사용자 이름은 each.key에서도 사용할 수 있다.

output.tf 수정

```
output "all_user" {
  value = aws_iam_user.example
}
```

실행

```
$ tf apply                                                              [15:06:51]

...

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

Outputs:

all_users = {
  "morpheus" = {
    "arn" = "arn:aws:iam::123456789012:user/morpheus"
    "force_destroy" = false
    "id" = "morpheus"
    "name" = "morpheus"
    "path" = "/"
    "permissions_boundary" = tostring(null)
    "tags" = tomap(null) /* of string */
    "tags_all" = tomap({})
  }
  "neo" = {
    "arn" = "arn:aws:iam::123456789012:user/neo"
    "force_destroy" = false
    "id" = "neo"
    "name" = "neo"
    "path" = "/"
    "permissions_boundary" = tostring(null)
    "tags" = tomap(null) /* of string */
    "tags_all" = tomap({})
  }
  "trinity" = {
    "arn" = "arn:aws:iam::123456789012:user/trinity"
    "force_destroy" = false
    "id" = "trinity"
    "name" = "trinity"
    "path" = "/"
    "permissions_boundary" = tostring(null)
    "tags" = tomap(null) /* of string */
    "tags_all" = tomap({})
  }
}
```

테라폼이 IAM 사용자 3명을 생성하였다. 그리고 all_users 출력 변수가 for_each의 키, 즉 이 경우 사용자 이름을 키로 가지며 해당 리소스이 전체 출력인 맵을 포함한다. all_arns 출

력 변수를 다시 가져오려면 맵에서 값만 반환하는 내장 함수 values를 이용해 ARN을 추출하고 스플랫 표현식을 사용하는 약간의 추가 작업을 수행해야 한다.

```
output "all_users" {
  value       = values(aws_iam_user.example)[*].arn
}
```

```
$ tf apply

...

Outputs:

all_users = [
  "arn:aws:iam::123456789012:user/morpheus",
  "arn:aws:iam::123456789012:user/neo",
  "arn:aws:iam::123456789012:user/trinity",
]
```

for_each를 사용해 리소스를 맵으로 처리하면 컬렉션 중간의 항목도 안전하게 제거 할 수 있어 count를 사용해 리소스를 배열로 처리하는 것보다 이점이 크다. 예를 들어 var.user_names 리스트 중간에서 ‘trinity’를 제거하려고 terraform plan 명령을 실행하면 다음과 같은 내용이 표시된다.

```
$ terraform plan
...

Terraform will perform the following actions:

  # aws_iam_user.example["trinity"] will be destroyed
  # (because key ["trinity"] is not in for_each map)
  - resource "aws_iam_user" "example" {
      - arn           = "arn:aws:iam::123456789012:user/trinity" -> null
      - id            = "trinity" -> null
      - name          = "trinity" -> null
    }
Plan: 0 to add, 0 to change, 1 to destroy.
```

for_each의 장점은 리소스 내에서 열 개의 인라인 블록을 만들 수 있다는 것이다. 예를 들어 for_each를 사용하면 webserver_cluster 모듈에서 ASG에 대한 tag 인라인 블록을 동적으로 생성할 수 있다. 먼저 사용자가 사용자 정의 태그를 지정할 수 있게 하려면 modules/services/webserver-cluster/variables.tf에 custom_tags라는 새로운 맵 입력 변수를 추가한다.

```
variable "custom_tags" {
  description = "Custom tags to set on the Instances in the ASG"
  type        = map(string)
  default     = {}
}
```

다음으로 프로덕션 환경의 live/prod/services/webserver-cluster/main.tf 에서 다음과 같이 일부 사용자 정의 태그를 설정한다.

```
module "webserver_cluster" {
  source = "../../../../modules/services/webserver-cluster"

  cluster_name           = webservers-prod
  db_remote_state_bucket = "(Your Bucket Name)"
  db_remote_state_key    = "prod/data-stores/mysql/terraform.tfstate"
  instance_type        = "m4.large"
  min_size             = 1
  max_size             = 2

  custom_tags = {
    Owner     = "team-foo"
    DeployedBy = "terraform"
  }

}
```

Owner : 이 ASG를 소유한 팀을 명시

DeployedBy : 이 인프라가 테라폼을 사용하여 배포되었음을 명시, 이 인프라를 수동으로 수정해서는 안된다는 의미이다.

이제 태그를 지정했으니 이것을 aws_autoscaling_group 리소스에 설정한다.

```
resource "aws_autoscaling_group" "example" {
  launch_configuration = aws_launch_configuration.example.name
  vpc_zone_identifier  = data.aws_subnet_ids.default.ids
```

```
  target_group_arns    = [aws_lb_target_group.asg.arn]
  health_check_type    = "ELB"

  min_size = var.min_size
  max_size = var.max_size

  tag {
    key                 = "Name"
    value               = var.cluster_name
    propagate_at_launch = true
  }

  for (tag in var.custom_tags) {
    tag {
      key                 = tag.key
      value               = tag.value
      propagate_at_launch = true
    }
  }
}
```

이 의사 코드는 작동하지 않지만 for_each 표현식은 작동한다. for_each를 사용해 인라인 블록을 동적으로 생성하는 구문은 다음과 같다.

```
dymaic "<VAR_NAME>" {
  for_each = <COLLECTION>

  content {
    [CONFIG...]
  }
}
```

VAR_NAME : 각 반복의 값을 저장할 변수

COLLECTION : 반복되는 리스트 또는 맵

content : 각 반복에서 생성되는 항목

content 블록 내에서 <VAR_NAME>.key 및 <VAR_NAME>.value를 사용해 COLLECTION에 있는 현재 항목의 키와 값에 각각 액세스할 수 있다.

for_each를 리스트와 함께 사용하는 경우 key는 인덱스가 되고 value는 해당 인덱스 목록에 있는 항목이 된다. 그리고 for_each를 맵과 함께 사용하는 경우 key와 value는 맵의 키-값 쌍 중 하나가 된다.

결과

```
resource "aws_autoscaling_group" "example" {
  launch_configuration = aws_launch_configuration.example.name
  vpc_zone_identifier  = data.aws_subnet_ids.default.ids
  target_group_arns    = [aws_lb_target_group.asg.arn]
  health_check_type    = "ELB"

  min_size = var.min_size
  max_size = var.max_size

  tag {
    key                 = "Name"
    value               = var.cluster_name
    propagate_at_launch = true
  }

  for (tag in var.custom_tags) {
    tag {
      key                 = tag.key
      value               = tag.value
      propagate_at_launch = true
    }
  }
}

dynamic "tag" {
    for_each = var.custom_tage
    content {
      key                 = tag.key
      value               = tag.value
      propagate_at_launch = true
    }
  }
}
```