

# AWS Security Groups and Terraform Functions- foreach, if else, list, locals/variables

Links-:

[aws\\_security\\_group](#) | [Resources](#) | [hashicorp/aws](#) | [Terraform](#) | [Terraform Registry](#)

1. Defining different names for every instance created through count variable.  
As seen in the previous module, The count function creates the multiple EC2 instances at once, but with the same name. If we want to define every instance with the unique name, we can do it in the following way.

```
admin@DESKTOP-9UJRCUE MINGW64 ~/AkshitTerraformFiles
$ cat Ec2Instance.tf
data "aws_ami" "ami_id" {
  most_recent = true

  filter {
    name     = "name"
    values   = ["amzn2-ami-kernel*"]
  }

  filter {
    name     = "virtualization-type"
    values   = ["hvm"]
  }
}

resource "aws_instance" "myec2Instance" {
  ami           = data.aws_ami.ami_id.id
  instance_type = var.instance_type

  tags = {
    Name = "${var.name}-${count.index}st"
  }

  count = 3
}
```

## CLOUD &amp; DEVOPS

The screenshot displays the AWS Management Console interface for the EC2 Instances page. The left-hand navigation pane includes links to the EC2 Dashboard, EC2 Global View, Events, and a dropdown for Instances, which is currently expanded to show 'Instances', 'Instance Types', 'Launch Templates', and 'Spot Requests'. The main content area is titled 'Instances (3) Info' and features a search bar with the placeholder text 'Find instance by attribute or tag (case-sensitive)'. Below the search bar is a table listing three EC2 instances. Each instance row includes a checkbox, the instance name, the instance ID, the instance state (all are 'Running' with a green checkmark icon), and the instance type (all are 't2.micro'). At the bottom of the table, there is a 'Select an instance' button.

|                          | Name                       | Instance ID         | Instance state | Instance type |
|--------------------------|----------------------------|---------------------|----------------|---------------|
| <input type="checkbox"/> | EC2_Instance_terraform-2st | i-0aeb1ebd42514edfe | Running        | t2.micro      |
| <input type="checkbox"/> | EC2_Instance_terraform-0st | i-0ea60969a27c312f0 | Running        | t2.micro      |
| <input type="checkbox"/> | EC2_Instance_terraform-1st | i-013f7555041ceb526 | Running        | t2.micro      |

## 2. If condition in Terraform:

Scenario- we are giving 2 EC2 resources configurations, one for dev, one for prod. And setting a variable named "is\_test" which is by default set to be false, means it is for production env.

When working on the dev environment EC2 resource, its value is set to be true, stating that the instance environment is test environment and the count is set to be 0. When working on prod EC2 resource, it is set to be false, stating it is production environment instance and its count is set to be 1.

Along with that we gave else condition in test resource stating if the condition is not matched, the count should be 0 and vice versa for prod resource.

As you can see in the AWS output, the prod instance is created cause the condition met for is\_test variable in Prod resource config.

## CLOUD &amp; DEVOPS

```
admin@DESKTOP-9UJRCUE MINGW64 ~/AkshiTerraformFiles
$ cat Ec2Instance.tf
data "aws_ami" "ami_id" {
  most_recent = true

  filter {
    name     = "name"
    values   = ["amzn2-ami-kerne*"]
  }

  filter {
    name     = "virtualization-type"
    values   = ["hvm"]
  }
}

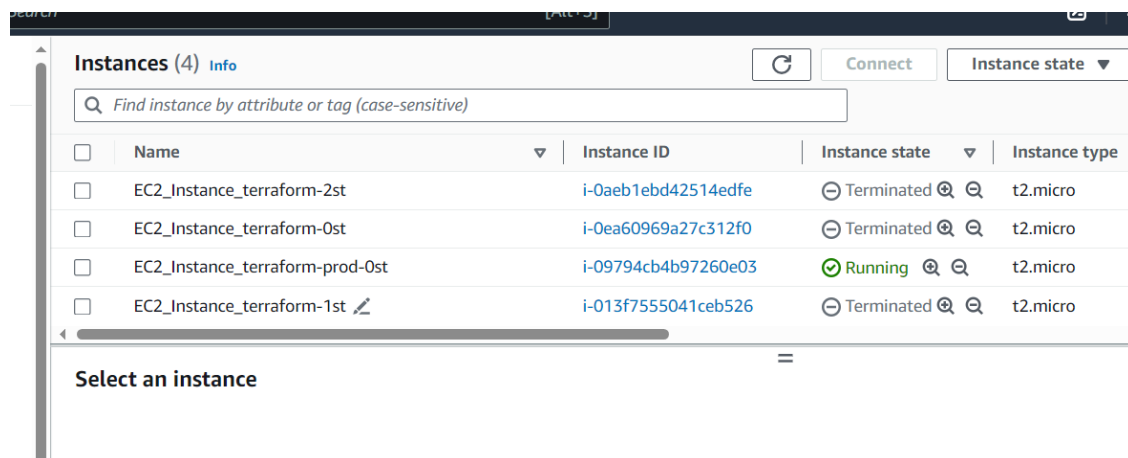
variable "is_test"{
  default = false
}

resource "aws_instance" "dev" {
  ami           = data.aws_ami.ami_id.id
  instance_type = var.instance_type

  tags = {
    Name = "${var.name}-dev-${count.index}st"
  }
  count = var.is_test == true ? 1 : 0
}

resource "aws_instance" "prod" {
  ami           = data.aws_ami.ami_id.id
  instance_type = var.instance_type

  tags = {
    Name = "${var.name}-prod-${count.index}st"
  }
  count = var.is_test == false ? 1 : 0
}
```



| <input type="checkbox"/> | Name                            | Instance ID         | Instance state | Instance type |
|--------------------------|---------------------------------|---------------------|----------------|---------------|
| <input type="checkbox"/> | EC2_Instance_terraform-2st      | i-0aeb1ebd42514edfe | Terminated     | t2.micro      |
| <input type="checkbox"/> | EC2_Instance_terraform-0st      | i-0ea60969a27c312f0 | Terminated     | t2.micro      |
| <input type="checkbox"/> | EC2_Instance_terraform-prod-0st | i-09794cb4b97260e03 | Running        | t2.micro      |
| <input type="checkbox"/> | EC2_Instance_terraform-1st      | i-013f7555041ceb526 | Terminated     | t2.micro      |

3. You can also give type bool in place of default value in the is\_test variable. This will allow the user to set the value of true/false at run time, so that accordingly the AWS resource block can be called out and accordingly the dev or prod environment instance will be generated. This is

## CLOUD &amp; DEVOPS

the real time used case.

```
admin@DESKTOP-9UJRCUE MINGW64 ~/AkshitTerraformFiles
$ cat Ec2Instance.tf
data "aws_ami" "ami_id" {
  most_recent = true

  filter {
    name   = "name"
    values = ["amzn2-ami-kernel*"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
}

variable "is_test"{
  type = bool
}

resource "aws_instance" "dev" {
  ami          = data.aws_ami.ami_id.id
  instance_type = var.instance_type

  tags = {
    Name = "${var.name}-dev-${count.index}st"
  }
  count = var.is_test == true ? 1 : 0
}

resource "aws_instance" "prod" {
  ami          = data.aws_ami.ami_id.id
  instance_type = var.instance_type

  tags = {
    Name = "${var.name}-prod-${count.index}st"
  }
  count = var.is_test == false ? 1 : 0
}
```

```
admin@DESKTOP-9UJRCUE MINGW64 ~/AkshitTerraformFiles
$ terraform apply
var.is_test
Enter a value: true
```

| Instances (5) <a href="#">Info</a>  |                                 |                     |                |               | <a href="#">Refresh</a> | <a href="#">Connect</a> | Instance state <a href="#">▼</a> |
|---|---------------------------------|---------------------|----------------|---------------|-------------------------|-------------------------|----------------------------------|
| <input type="text" value="Find instance by attribute or tag (case-sensitive)"/> |                                 |                     |                |               |                         |                         |                                  |
| <input type="checkbox"/>  | Name                            | Instance ID         | Instance state | Instance type |                         |                         |                                  |
| <input type="checkbox"/>  | EC2_Instance_terraform-0st      | i-0ea60969a27c312f0 | Terminated     | t2.micro      | <a href="#">🔍</a>       | <a href="#">🔍</a>       |                                  |
| <input type="checkbox"/>  | EC2_Instance_terraform-prod-0st | i-09794cb4b97260e03 | Terminated     | t2.micro      | <a href="#">🔍</a>       | <a href="#">🔍</a>       |                                  |
| <input type="checkbox"/>  | EC2_Instance_terraform-1st      | i-013f7555041ceb526 | Terminated     | t2.micro      | <a href="#">🔍</a>       | <a href="#">🔍</a>       |                                  |
| <input type="checkbox"/>  | EC2_Instance_terraform-dev-0st  | i-08042059971e47cd4 | Running        | t2.micro      | <a href="#">🔍</a>       | <a href="#">🔍</a>       |                                  |

Select an instance

- Now let us learn about creating the multiple security groups in AWS through Terraform using ForEach function.

We will also see how to define the List and values in the terraform file

Refer to the AWS documentation-

[aws\\_security\\_group](#) | [Resources](#) | [hashicorp/aws](#) | [Terraform](#) | [Terraform Registry](#)

KeyTerms-:

ingress refers to the incoming network traffic

egress refers to the Outgoing network traffic

AWS DOCUMENTATION

Filter

aws provider

- > Guides
- > ACM (Certificate Manager)
- > ACM PCA (Certificate Manager Private Certificate Authority)
- > AMP (Managed Prometheus)
- > API Gateway
- > API Gateway V2
- > Account Management
- > Amplify
- > App Mesh
- > App Runner
- > AppConfig
- > AppFlow

Basic Usage

```
resource "aws_security_group" "allow_tls" {
  name           = "allow_tls"
  description    = "Allow TLS inbound traffic"
  vpc_id        = aws_vpc.main.id

  ingress {
    description = "TLS from VPC"
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = [aws_vpc.main.cidr_block]
    ipv6_cidr_blocks = [aws_vpc.main.ipv6_cidr_block]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
    ipv6_cidr_blocks = [":::/0"]
  }
}
```

First, we will perform the ingress networks creation

You may delete the aws\_instance resources config files for this scenario because we are just setting up the inbound networking ports and rules right now.

We have set up the variable named portList in which we have defined the list of different port numbers which we need for inbound network.

\*please note, these are just demo port numbers for the practice

In the ingress function, we have opened the for each loop, in which we have defined the list variable of ports which is called through the iterator in the content section, which defines the port numbers

You will be needing the cidr\_blocks section to be filled to see the all 4 inbound networks in AWS.

After executing the code, you will be able to see the new network added in the security networks section, with all 4 port sections in the inbound rules.

## CLOUD &amp; DEVOPS

```
admin@DESKTOP-9UJRCUE MINGW64 ~/AkshitTerraformFiles
$ cat AWSProvider_Configurations.tf
provider "aws" {
  region = "ap-south-1"
  shared_credentials_files = ["~/.aws/credentials"]
}

variable "portList"{
  type = list(number)
  default = [8080,8282,8686,8989]
}

resource "aws_security_group" "allow_tls" {

  dynamic "ingress" {
    for_each = var.portList
    iterator = port
    content{
      description      = "Inbound Traffic port"
      from_port        = port.value
      to_port          = port.value
      protocol         = "tcp"
      cidr_blocks      = ["0.0.0.0/0"]
    }
  }
  tags = {
    Name = "my_terraform_network"
  }
}
```

MINGW64/c/Users/admin/AkshitTerraformFiles

```
admin@DESKTOP-9UJRCUE MINGW64 ~/AkshitTerraformFiles
$ terraform apply
aws_security_group.allow_tls: Refreshing state... [id=sg-0817b2d0a15a80599]

Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
~ update in-place

Terraform will perform the following actions:

# aws_security_group.allow_tls will be updated in-place
~ resource "aws_security_group" "allow_tls" {
  id = "sg-0817b2d0a15a80599"
  ~ ingress = [
    + {
      + cidr_blocks = [
        + "0.0.0.0/0",
      ]
      + description = "Inbound Traffic port"
      + from_port   = 8080
      + ipv6_cidr_blocks = []
      + prefix_list_ids = []
      + protocol    = "tcp"
      + security_groups = []
      + self        = false
      + to_port     = 8080
    },
    + {
      + cidr_blocks = [
        + "0.0.0.0/0",
      ]
      + description = "Inbound Traffic port"
      + from_port   = 8282
      + ipv6_cidr_blocks = []
      + prefix_list_ids = []
      + protocol    = "tcp"
      + security_groups = []
      + self        = false
      + to_port     = 8282
    },
    + {
      + cidr_blocks = [
        + "0.0.0.0/0",
      ]
    },
  ]
}
```

## CLOUD &amp; DEVOPS

The screenshot displays the AWS IAM console interface. At the top, there's a header for 'Security Groups (1/5)' with an 'Info' link, a refresh button, an 'Actions' dropdown, an 'Export security groups to CSV' button, and a 'Create security group' button. Below this is a search bar labeled 'Filter security groups'. A table lists security groups with columns: Name, Security group ID, Security group name, VPC ID, Description, and Owner. The first row is selected, showing 'my\_terraform\_net...' with ID 'sg-0817b2d0a15a80599', name 'terraform-202310170...', VPC ID 'vpc-0af797f95f0553827', description 'Managed by Terraform', and owner '3463579...'. Below the table, there's a section for 'Inbound rules (4)' with a refresh button, 'Manage tags' button, and 'Edit inbound rules' button. A search bar labeled 'Filter security group rules' is present. A table lists inbound rules with columns: Security group rule..., IP version, Type, Protocol, Port range, and Source. The first rule is 'sgr-0ff0ae9828db8d1ea' with IP version 'IPv4', Type 'Custom TCP', Protocol 'TCP', Port range '8686', and Source '0.0.0.0/0'.

| Name                | Security group ID    | Security group name    | VPC ID                | Description               | Owner      |
|---------------------|----------------------|------------------------|-----------------------|---------------------------|------------|
| my_terraform_net... | sg-0817b2d0a15a80599 | terraform-202310170... | vpc-0af797f95f0553827 | Managed by Terraform      | 3463579... |
| -                   | sg-027f8e814657881a6 | launch-wizard-3        | vpc-0af797f95f0553827 | launch-wizard-3 create... | 3463579... |
| -                   | sg-0ddaba4b03b9c86dd | launch-wizard-2        | vpc-0af797f95f0553827 | launch-wizard-2 create... | 3463579... |

| Security group rule... | IP version | Type       | Protocol | Port range | Source    |
|------------------------|------------|------------|----------|------------|-----------|
| sgr-0ff0ae9828db8d1ea  | IPv4       | Custom TCP | TCP      | 8686       | 0.0.0.0/0 |
| sgr-056aed342bbffe334  | IPv4       | Custom TCP | TCP      | 8080       | 0.0.0.0/0 |
| sgr-0c48c1ada6c10c303  | IPv4       | Custom TCP | TCP      | 8282       | 0.0.0.0/0 |
| sgr-091e166f8144f632e  | IPv4       | Custom TCP | TCP      | 8989       | 0.0.0.0/0 |

5. You may also use “locals” function with variable to fix the value of a particular variable which cannot be changed at run time by the user or by any other config file. In this scenario, local function will set the port number constantly to 80.

```
resource "aws_security_group" "mysg1" {
  name           = "allow_tls"
  description    = "Allow TLS inbound traffic"

  ingress {
    description    = "TLS from VPC"
    from_port      = local.port
    to_port        = local.port
    protocol       = "tcp"
  }
}

// locals is a block in terraform where the variable value get fixed/constant
// lling the module.

locals{
  port = 80
}

variable "port" {
}

~
```