- Count

```
resource "aws_default_security_group" "this" {
  count = local.create_vpc && var.manage_default_security_group ? 1 : 0
```

'Count' is an argument allow us to repeat n times a block instead of write n times that block.
In this case we can see that 'count' is used to evaluate a condition to conditionate the creation of a block

- For each

```
dynamic "ingress" {
  for_each = var.default_security_group_ingress
  content {
    self             = lookup(ingress.value, "self", null)
    cidr_blocks      = compact(split(",", lookup(ingress.value, "cidr_blocks", "")))
    ipv6_cidr_blocks = compact(split(",", lookup(ingress.value, "ipv6_cidr_blocks", "")))
    prefix_list_ids  = compact(split(",", lookup(ingress.value, "prefix_list_ids", "")))
    security_groups  = compact(split(",", lookup(ingress.value, "security_groups", "")))
    description      = lookup(ingress.value, "description", null)
    from_port        = lookup(ingress.value, "from_port", 0)
    to_port          = lookup(ingress.value, "to_port", 0)
    protocol         = lookup(ingress.value, "protocol", "-1")
  }
```

For_each is an argument allow us to repeat n times a block but with the difference that this iterate over a map, this is useful if we need to create n block with different values. In the example we can see how 'for_each' is used to create a list of ingress rules within a security group

- Element()

```
"Name" = var.single_nat_gateway ? "${var.name}-${var.private_subnet_suffix}" : format(
  "${var.name}-${var.private_subnet_suffix}-%s",
  element(var.azs, count.index),
)
```

element() is a function that allow us to retrieve the data at the indicated index of a list. In the example we can see how is used within the format function to create dynamically tags using count to specify the index of the variable azs