Harnessing the Power of `for_each` in Terraform

One of the most versatile features in Terraform is the `for_each` expression, which enables dynamic resource creation and management based on the elements of a collection. This blog post will explore the `for_each` expression, its syntax, and practical use cases to help you make the most out of it in your Terraform configurations.

1. Creating Resources from a List

Consider a scenario where you need to create multiple AWS S3 buckets with different names:

```
variable "bucket_names" {
  type    = list(string)
  default = ["bucket1", "bucket2", "bucket3"]
}
resource "aws_s3_bucket" "example" {
  for_each = toset(var.bucket_names)
  bucket = each.value
tags = {
    Name = each.value
  }
}
```

In this example, the `for_each` expression iterates over the list of bucket names, creating an S3 bucket for each name.

Nested `for_each`

You can also use `for_each` in nested constructs. Consider a scenario where you need to create AWS EC2 instances within different subnets:

```
variable "subnets" {
 type = list(string)
  default = ["subnet-abc123", "subnet-def456"]
}
variable "instances" {
  type = map(string)
  default = {
    "web"  = "t2.micro"
    "db"   = "t2.small"
  }
}
resource "aws_instance" "example" {
  for_each = { for subnet in var.subnets : subnet => var.instances }
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = each.value[each.key]
  subnet_id     = each.key
  tags = {
    Name = each.key
  }
}
```

In this example, `for_each` is used to create instances within each subnet, with different instance types for web and database servers.