# Mastering Splat Expressions in Terraform

One powerful feature in Terraform is the use of splat expressions, which simplify handling lists and maps. This blog post will explore splat expressions, their syntax, and how they can be utilized effectively in your Terraform configurations.

## What are Splat Expressions?

Splat expressions in Terraform provide a convenient way to work with lists and maps. They enable you to extract specific attributes from each element of a list or map in a concise and readable manner. Splat expressions can significantly reduce the complexity of your Terraform code.

## Syntax of Splat Expressions

Splat expressions use the `[*]` syntax to iterate over elements of a list or map and extract specific attributes. The general form of a splat expression is:

```
resource "aws_instance" "example" {

  count = 3



  ami           = "ami-0c55b159cbfafe1f0"

  instance_type = "t2.micro"

}



output "instance_ids" {

  value = aws_instance.example[*].id

}
```

In this example, `aws_instance.example[*].id` uses a splat expression to extract the `id` attribute from each instance in the list.

### Extracting Attributes from a Map of Objects

Consider a scenario where you have a map of AWS instances and you want to extract their IDs:

```
resource "aws_instance" "example" {
  for_each = {
    instance1 = "ami-0c55b159cbfafe1f0"
    instance2 = "ami-0c55b159cbfafe1f0"
  }

  ami           = each.value
  instance_type = "t2.micro"
}

output "instance_ids" {
  value = [for key, instance in aws_instance.example : instance.id]
}
```

In this example, the for expression `[for key, instance in aws_instance.example : instance.id]` is used to extract the `id` attribute from each instance in the map.

# Understanding the `zipmap` Function in Terraform

Terraform, an open-source infrastructure as code (IaC) tool by HashiCorp, provides various functions to manipulate data structures within your configurations. One such function is `zipmap`, which is incredibly useful for creating maps from two separate lists. This blog post will explore the `zipmap` function, its syntax, and practical use cases to help you make the most out of it in your Terraform configurations.

## Syntax of `zipmap`

The syntax of the `zipmap` function is straightforward:

**zipmap(keys, values)**

**`keys`: A list of keys to be used in the map.**

**`values`: A list of values to be associated with the keys.**

## Examples of Using `zipmap`

### 1. Basic Example

**Let's start with a simple example where we create a map from two lists of equal length:**

```
variable "keys" {

  type    = list(string)

  default = ["name", "age", "location"] }

variable "values" {

  type    = list(string)

  default = ["Alice", "30", "Wonderland"]}

output "zipped_map" {

  value = zipmap(var.keys, var.values) }
```

In this example, the `zipmap` function combines the `keys` and `values` lists into a map:

```
{
  "name": "Alice",
  "age": "30",
  "location": "Wonderland"
}
```