Terraform - DAY-2

Presented By:
Rashi Rana

Attributes & Output Values

Understanding Attributes

Terraform has the capability to output the attribute of a resource with the output value

Example:

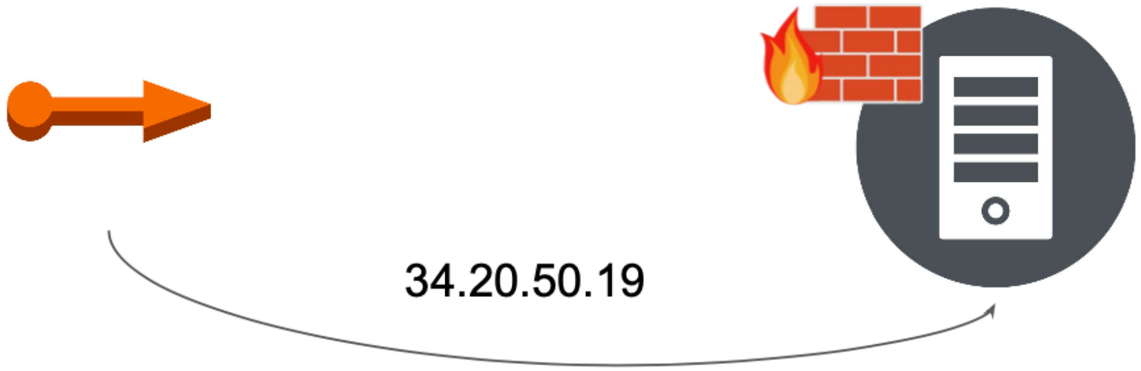
ec2_public_ip = 35.161.21.197
bucket_identifier = terraform-test.s3.amazonaws.com

Attributes are important

An output attribute can not only be used for the user reference but can also act as an input to other resources being created via Terraform

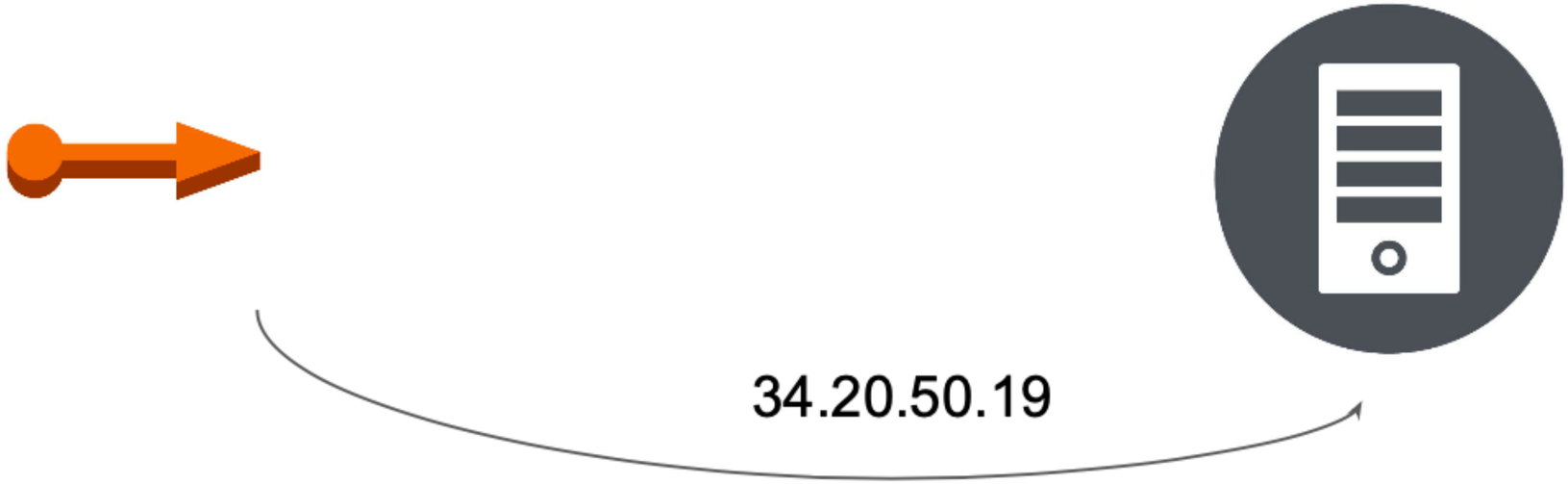
Let’s understand this with an example:

After EIP gets created, its IP address should automatically get whitelisted in the security group.



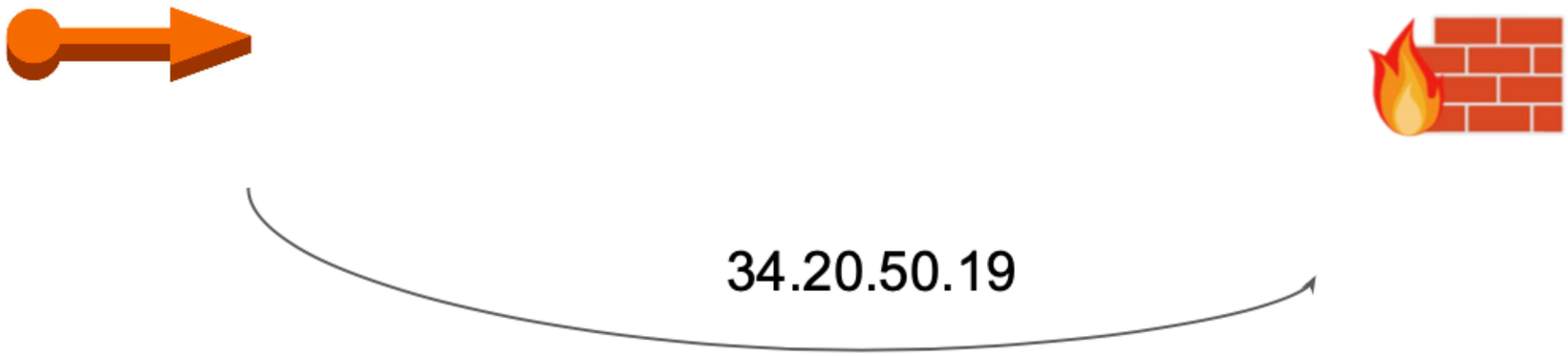
Lab 1: EIP and EC2 Instance

Once the EIP gets created, its IP address should automatically get assigned(referenced) to the ec2 instance



Lab 2: EIP and Security Group

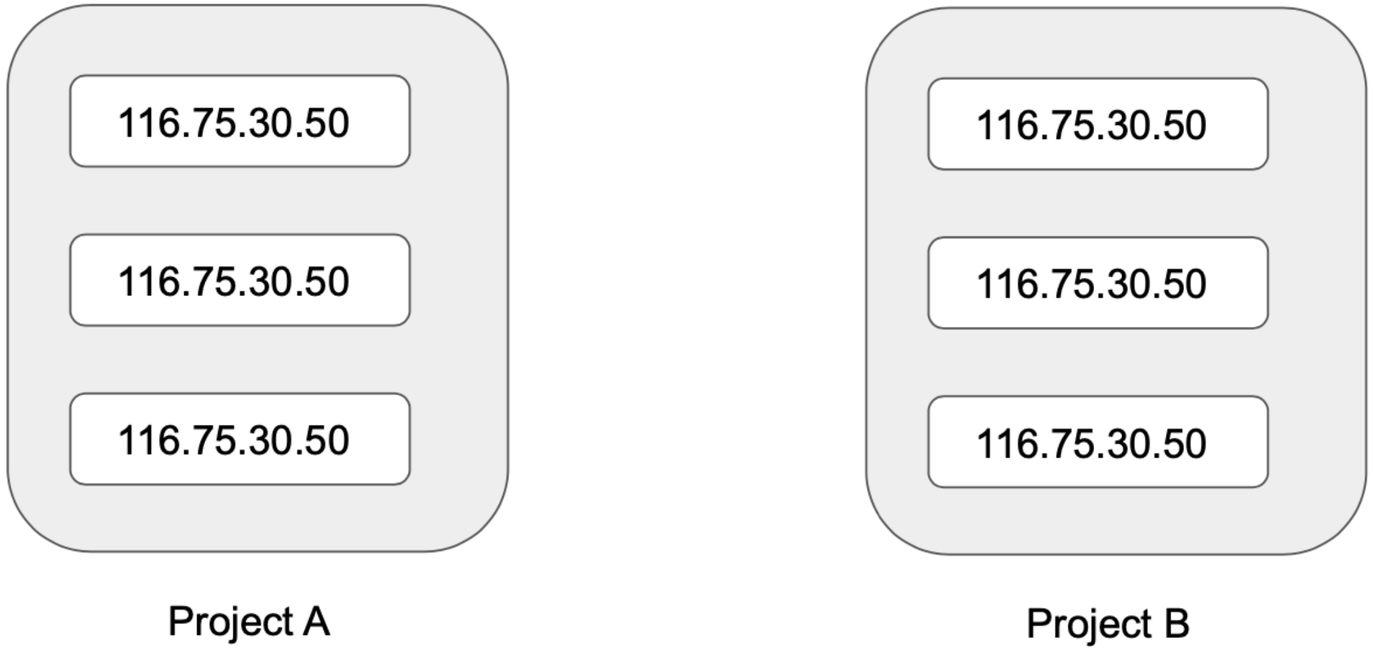
After EIP gets created, its IP address should automatically get whitelisted in the security group.



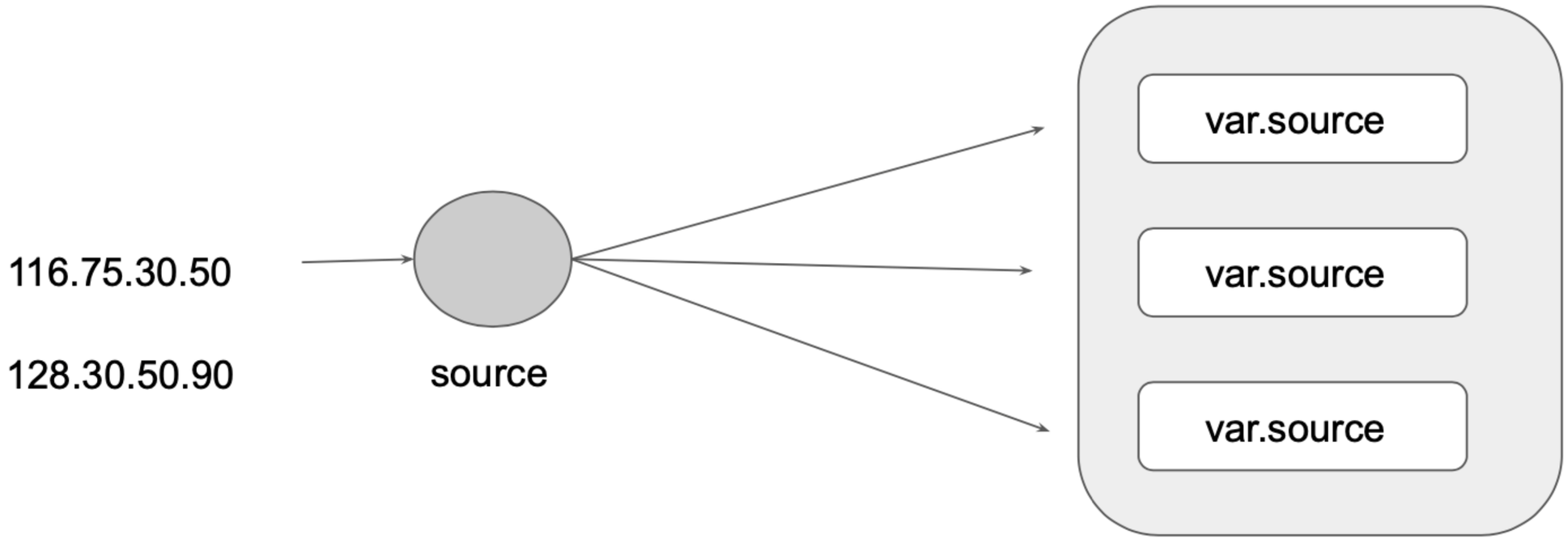
Terraform Variables

Static = Work

Repeated static values can create more work in the future.



Variables are good



Approaches to Variable Assignment

Multiple Approaches to Variable Assignment

variables in Terraform can be assigned values in multiple ways.

Some of these include:

1. Environment variables
2. Command Line Flags
3. From a File
4. Variable Defaults

Data Types for Variables

Overview of Type Constraints

The type argument in a variable block allows you to restrict the type of value that will be accepted as the value for a variable

```
variable "image_id" {
  type = string
}
```

If no type constraint is set then a value of any type is accepted.

Example Use-Case

Every employee in Medium Corp is assigned an Identification Number.

Any resource that an employee creates should be created with the name of the identification number only.

variables.tf	terraform.tfvars
variable “instance_name” {}	instance_name=”john-123”

Example Use-Case

Every employee in Medium Corp is assigned an Identification Number.

Any resource that an employee creates should be created with the name of the identification number only.

variables.tf	terraform.tfvars

variable “instance_name” {
type=number
}
	instance_name=”john-123”

Overview of Data Types

Type Keywords	Description
string	Sequence of Unicode characters representing some text, like "hello".
list	Sequential list of values identified by their position. Starts with 0 [“mumbai” ,”singapore”, ”usa”]
map	a group of values identified by named labels, like {name = "Mabel", age = 52}.
number	Example: 200

Count Parameter

Overview of Count Parameter

The count parameter on resources can simplify configurations and let you scale resources by simply incrementing a number.

Let's assume, you need to create two EC2 instances. One of the common approach is to define two separate resource blocks for aws_instance.

```
resource "aws_instance" "instance-1" {
   ami = "ami-082b5a644766e0e6f"
   instance_type = "t2.micro"
}
```

```
resource "aws_instance" "instance-2" {
   ami = "ami-082b5a644766e0e6f"
   instance_type = "t2.micro"
}
```

Overview of Count Parameter

With count parameter, we can simply specify the count value and the resource can be scaled accordingly.

```
resource "aws_instance" "instance-1" {
    ami = "ami-082b5a644766e0e6f"
    instance_type = "t2.micro"
    count = 5
}
```

Count Index

In resource blocks where the count is set, an additional count object is available in expressions, so you can modify the configuration of each instance.

This object has one attribute:

count.index — The distinct index number (starting with 0) corresponding to this instance.

Understanding Challenge with Count

With the below code, terraform will create 5 IAM users. But the problem is that all will have the same name.

```
resource "aws_iam_user" "lb" {
  name = "loadbalancer"
  count = 5
  path = "/system/"
}
```

Understanding Challenge with Count

count.index allows us to fetch the index of each iteration in the loop.

```
resource "aws_iam_user" "lb" {
  name = "loadbalancer.${count.index}"
  count = 5
  path = "/system/"
}
```

Understanding Challenge with Default Count Index

Having a username like loadbalancer0, loadbalancer1 might not always be suitable.

Better names like dev-loadbalancer, stage-loadbalancer, prod-loadbalancer is better.

count.index can help in such scenario.

```
variable "elb_names" {
 type = list
 default = ["dev-loadbalancer", "stage-loadbalanacer","prod-loadbalancer"]
}
```

Conditional Expression

Overview of Conditional Expression

A conditional expression uses the value of a bool expression to select one of two values.

Syntax of Conditional expression:

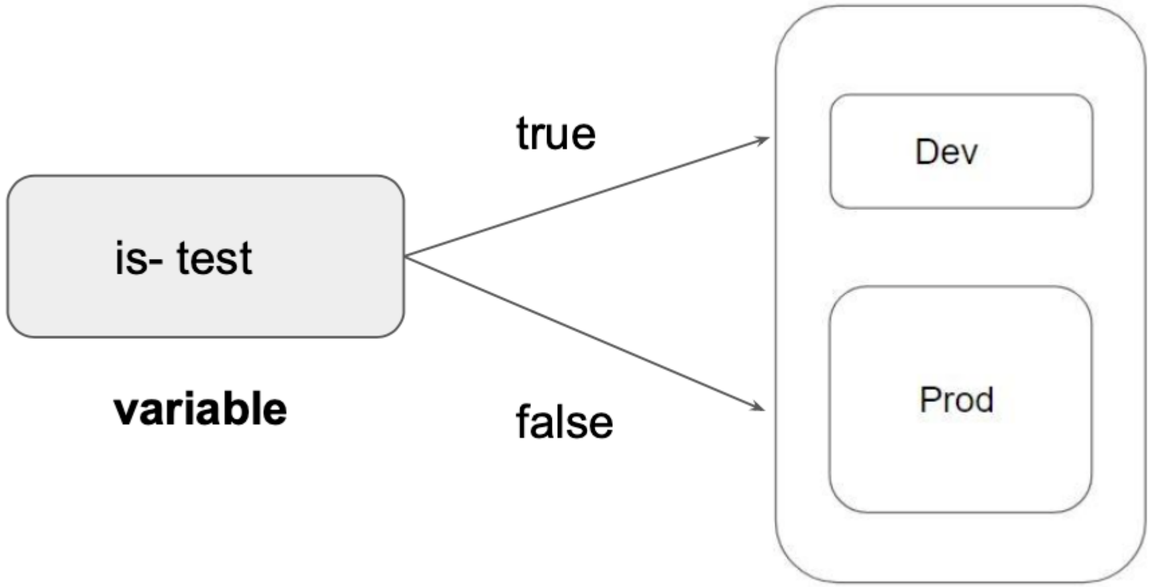
condition ? true_val : false_val

If condition is true then the result is true_val. If condition is false then the result is false_val.

Example of Conditional Expression

Let’s assume that there are two resource blocks as part of terraform configuration.

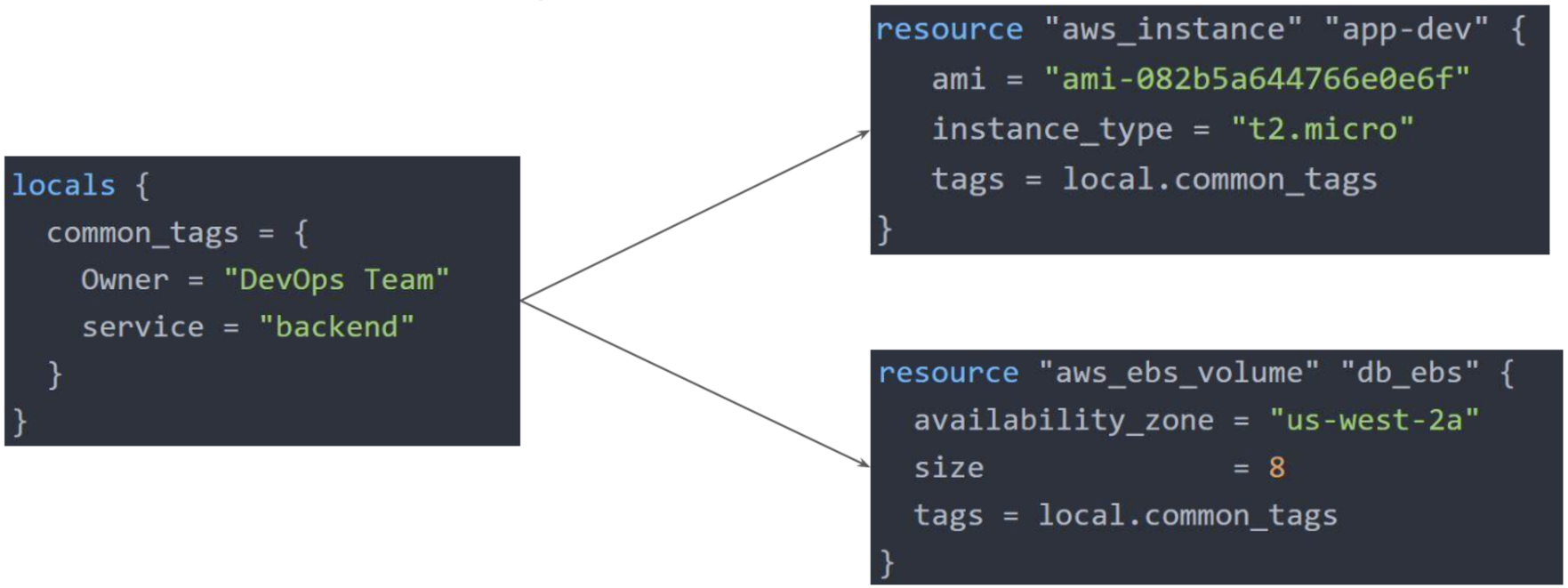
Depending on the variable value, one of the resource blocks will run.



Local Values

Overview of Local Values

A local value assigns a name to an expression, allowing it to be used multiple times within a module without repeating it.



Terraform Functions

Overview of tf functions

The Terraform language includes a number of built-in functions that you can use to transform and combine values.

The general syntax for function calls is a function name followed by comma-separated arguments in parentheses:

function (argument1, argument2)

Example:

> max(5, 12, 9)

12

List of Available functions

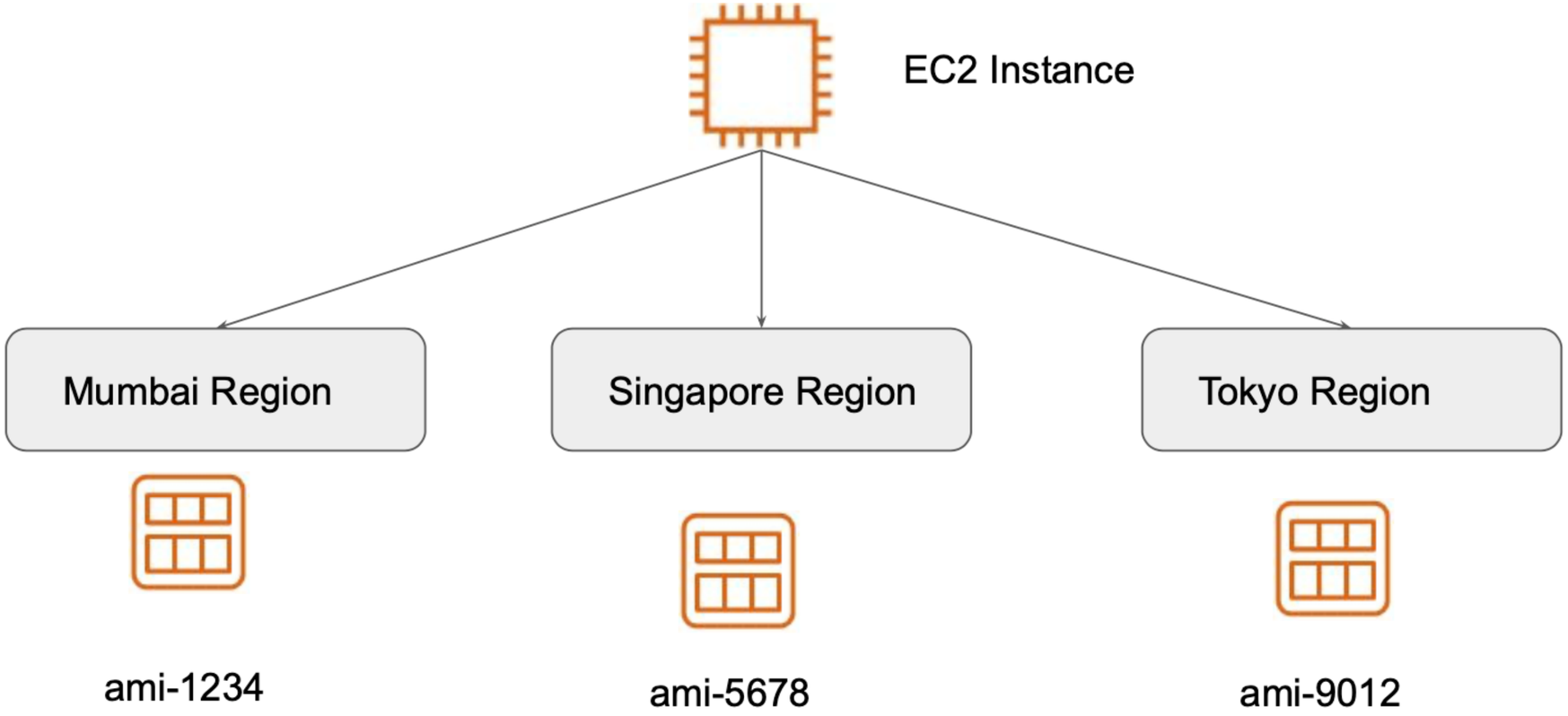
The Terraform language does not support user-defined functions, and so only the functions built in to the language are available for use

- Numeric
- String
- Collection
- Encoding
- Filesystem
- Date and Time
- Hash and Crypto
- IP Network
- Type Conversion

Data Sources

Overview

Data sources allow data to be fetched or computed for use elsewhere in Terraform configuration.



Data Source Code

- Defined under the data block.
- Reads from a specific data source (aws_ami) and exports results under “app_ami”

```
data "aws_ami" "app_ami" {
  most_recent = true
  owners = ["amazon"]


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

```
resource "aws_instance" "instance-1" {
    ami = data.aws_ami.app_ami.id
   instance_type = "t2.micro"
}
```

More about filters

https://docs.aws.amazon.com/cli/latest/reference/ec2/describe-instances.html

Terraform Debugging

Terraform has detailed logs which can be enabled by setting the TF_LOG environment variable to any value.

You can set TF_LOG to one of the log levels TRACE, DEBUG, INFO, WARN or ERROR to change the verbosity of the logs

Terraform Format

Anyone who is into programming knows the importance of formatting the code for readability.

The terraform fmt command is used to rewrite Terraform configuration files to take care of the overall formatting.

Before fmt

```
provider "aws" {
      region     = "us-west-2"
  access_key = "AKIAQIW66DN2W7WOYRGY"
  secret_key    = "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"
  version    = ">=2.10,<=2.30"
}
```

After fmt

```
provider "aws" {
  region     = "us-west-2"
  access_key = "AKIAQIW66DN2W7WOYRGY"
  secret_key = "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"
  version    = ">=2.10,<=2.30"
}
```

Terraform Validate

Terraform Validate primarily checks whether a configuration is syntactically valid.

It can check various aspects including unsupported arguments, undeclared variables and others.

```
resource "aws_instance" "myec2" {
  ami           = "ami-082b5a644766e0e6f"
  instance_type = "t2.micro"
  sky = "blue"
}
```

```
bash-4.2# terraform validate

Error: Unsupported argument

  on validate.tf line 10, in resource "aws_instance" "myec2":
  10:   sky = "blue"

An argument named "sky" is not expected here.
```

Understanding Semantics

Terraform generally loads all the configuration files within the directory specified in alphabetical order.

The files loaded must end in either .tf or .tf.json to specify the format that is in use.

Dynamic Block

Understanding the Challenge

In many of the use cases, there are repeatable nested blocks that need to be defined.

This can lead to a long code and it can be difficult to manage in a longer time.

```
ingress {
  from_port   = 9200
  to_port     = 9200
  protocol    = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}
```

```
ingress {
  from_port   = 8300
  to_port     = 8300
  protocol    = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}
```

Dynamic Blocks

Dynamic Block allows us to dynamically construct repeatable nested blocks which is supported inside resource, data, provider, and provisioner blocks:

```
dynamic "ingress" {
  for_each = var.ingress_ports
  content {
    from_port   = ingress.value
    to_port     = ingress.value
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

Iterators

The iterator argument (optional) sets the name of a temporary variable that represents the current element of the complex value

If omitted, the name of the variable defaults to the label of the dynamic block ("ingress" in the example above).

```
dynamic "ingress" {
  for_each = var.ingress_ports
  content {
    from_port   = ingress.value
    to_port     = ingress.value
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

```
dynamic "ingress" {
  for_each = var.ingress_ports
  iterator = port
  content {
    from_port   = port.value
    to_port     = port.value
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

Terraform Taint/Replace

Use Case

You have created a new resource via Terraform.

Users have made a lot of manual changes (both infrastructure and inside the server)

Two ways to deal with this: Import Changes to Terraform / Delete & Recreate the resource

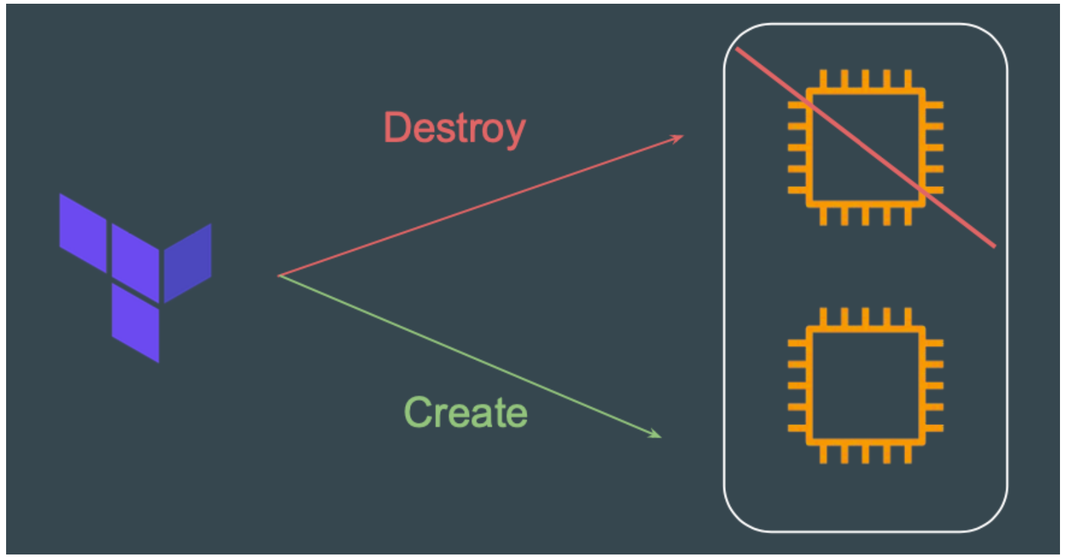
Recreating the Resource

The -replace option with *terraform apply* to force Terraform to replace an object even though there are no configuration changes that would require it.

```
terraform apply -replace="aws_instance.web"
```

Similar kind of functionality was achieved using terraform taint command in older versions of Terraform.

For Terraform v0.15.2 and later, HashiCorp recommend using the -replace option with terraform apply



Spalat Expression

Overview

Splat Expression allows us to get a list of all the attributes.

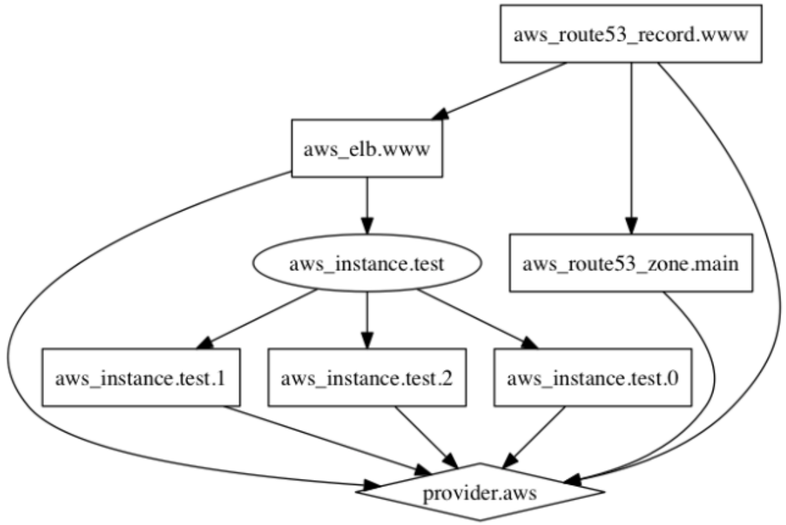
```
resource "aws_iam_user" "lb" {
  name = "iamuser.${count.index}"
  count = 3
  path = "/system/"
}

output "arns" {
  value = aws_iam_user.lb[*].arn
}
```

Terraform Graph

The terraform graph command is used to generate a visual representation of either a configuration or execution plan

The output of terraform graph is in the DOT format, which can easily be converted to an image.



Saving Terraform Plan to a File

Terraform Plan File

The generated terraform plan can be saved to a specific path.

This plan can then be used with terraform apply to be certain that only the changes shown in this plan are applied.

Example:

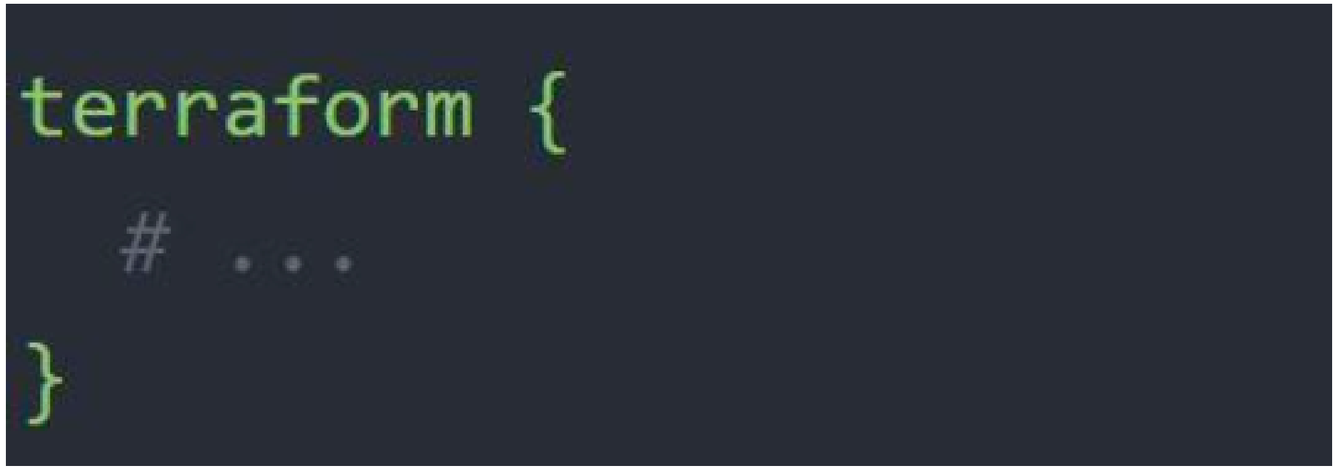
terraform plan -out=path

Terraform Settings

Overview of Terraform Settings

The special terraform configuration block type is used to configure some behaviors of Terraform itself, such as requiring a minimum Terraform version to apply your configuration.

Terraform settings are gathered together into terraform blocks:



Setting 1 - Terraform Version

The required_version setting accepts a version constraint string, which specifies which versions of Terraform can be used with your configuration.

If the running version of Terraform doesn't match the constraints specified, Terraform will produce an error and exit without taking any further actions.

```
terraform {
  required_version = "> 0.12.0"
}
```

Setting 2 - Provider Version

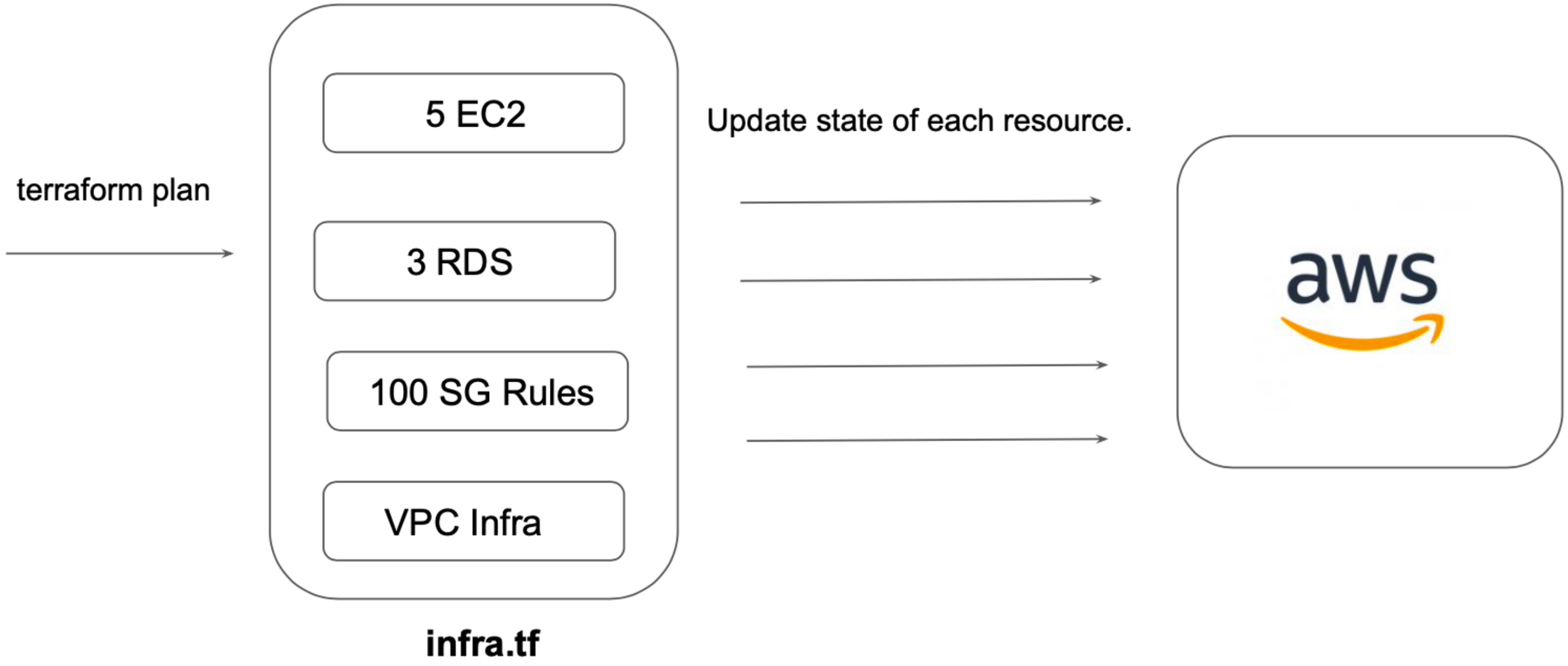
The required_providers block specifies all of the providers required by the current module, mapping each local provider name to a source address and a version constraint.

```
terraform {
  required_providers {
    mycloud = {
      source  = "mycorp/mycloud"
      version = "~> 1.0"
    }
  }
}
```

Dealing with Larger Infrastructure

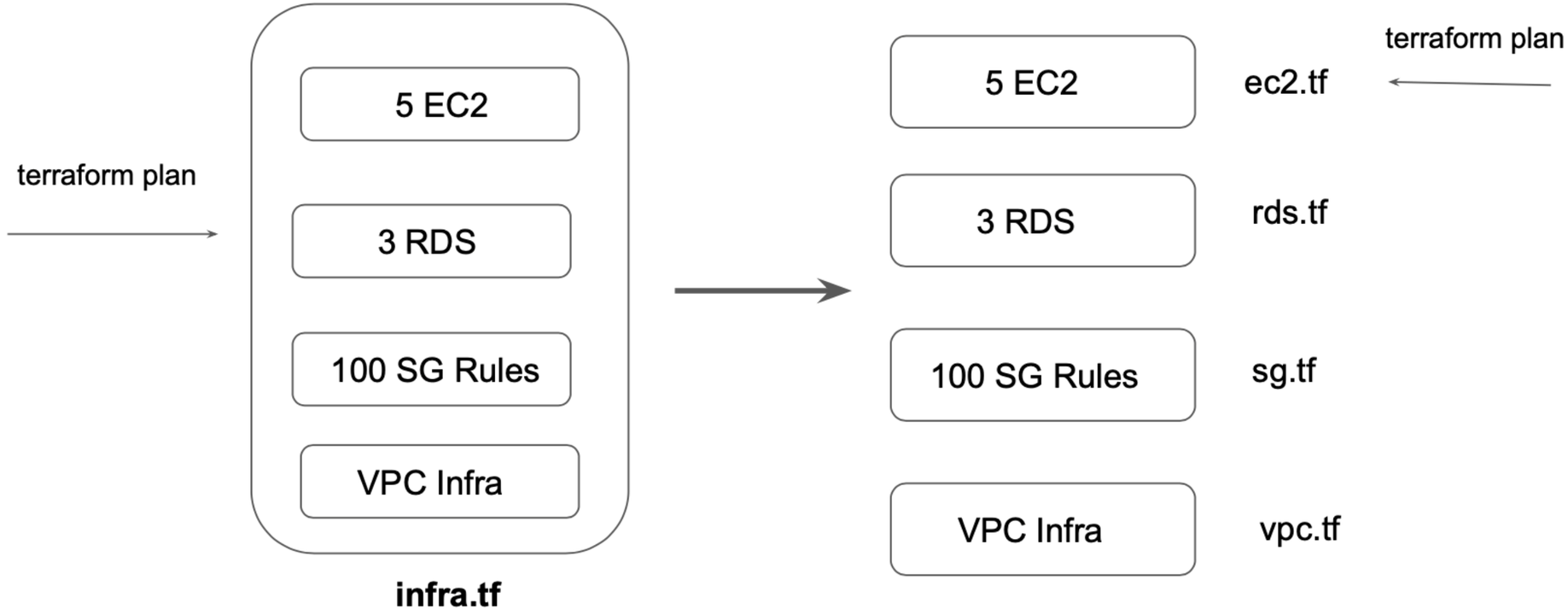
Challenges with Larger Infrastructure

When you have a larger infrastructure, you will face issue related to API limits for a provider.



Dealing with Larger Infrastructure

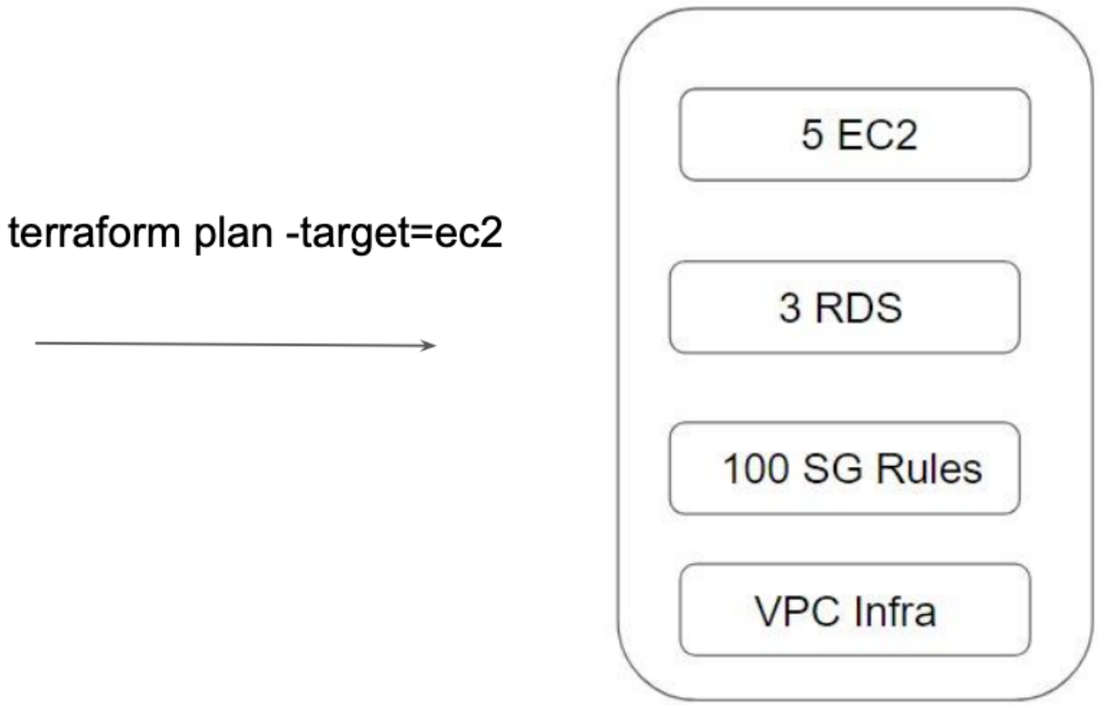
Switch to smaller configuration were each can be applied independently.



Specify the target

The -target=resource flag can be used to target a specific resource.

Generally used as a means to operate on isolated portions of very large configurations



Zipmap - Sample Use case

You are creating multiple IAM users.

You need output which contains direct mapping of IAM names and ARNs

```
zipmap = {
  "demo-user.0" = "arn:aws:iam::018721151861:user/system/demo-user.0"
  "demo-user.1" = "arn:aws:iam::018721151861:user/system/demo-user.1"
  "demo-user.2" = "arn:aws:iam::018721151861:user/system/demo-user.2"
}
```

Comments in Terrafrom Code

The Terraform language supports three different syntaxes for comments:

Type	Description
#	begins a single-line comment, ending at the end of the line.
//	also begins a single-line comment, as an alternative to #.
/* and */	are start and end delimiters for a comment that might span over multiple lines.

Data Type - SET

for_each

QUIZ

Question:

Suresh has copied a sample terraform code from a GitHub repository to his local terraform configuration file. There are certain formatting issues in the code.

```
resource "aws_instance" "kplabs-ec2" {
 ami =         "ami-090fa75af13c156b4"
 instance_type =         "m5.large"
 count =               5
}
```

Which command can Suresh run to ensure that the code is automatically formatted?

Question:

Suresh has copied a sample terraform code from a GitHub repository to his local terraform configuration file. There are certain formatting issues in the code.

```
resource "aws_instance" "kplabs-ec2" {
 ami =         "ami-090fa75af13c156b4"
 instance_type =         "m5.large"
 count =               5
}
```

Which command can Suresh run to ensure that the code is automatically formatted?

Answer: **terraform fmt**

Question:

James is having an issue with his Terraform code. As part of the troubleshooting process, he intends to enable debugging.

What is the way to do that?

1. Create an environment variable of TF_VAR_LOG=TRACE
2. Add debug=true parameter in the Terraform code.
3. Terraform does not support Debugging functionality yet.
4. Create an environment variable of TF_LOG=TRACE

Question:
James is having an issue with his Terraform code. As part of the troubleshooting process, he intends to enable debugging.
What is the way to do that?

1. Create an environment variable of TF_VAR_LOG=TRACE
2. Add debug=true parameter in the Terraform code.
3. Terraform does not support Debugging functionality yet.
4. **Create an environment variable of TF_LOG=TRACE**

Question:

Alice has written a following Terraform code that includes a count parameter along with conditional expression.
If the condition is met and true, how many EC2 instances will be created?

```
resource "aws_instance" "kplabs-ec2" {
 ami = "ami-090fa75af13c156b4"
 instance_type = "m5.large"
 count = var.kplabs == true ? 3 : 0
}
```

1. 0
2. 3

Question:

Alice has written a following Terraform code that includes a count parameter along with conditional expression.
If the condition is met and true, how many EC2 instances will be created?

```
resource "aws_instance" "kplabs-ec2" {
 ami = "ami-090fa75af13c156b4"
 instance_type = "m5.large"
 count = var.kplabs == true ? 3 : 0
}
```

1. 0
2. **3**

Question:

Matthew is new to terraform. He is creating an EC2 instance.
To fetch the Public IP of the EC2, Matthew needs to manually login to the EC2 console and search for EC2 or search through Terraform state file.
Matthew wants to IP address of the instance to show in output automatically once the EC2 instance is created. Which terraform feature can be used?

1. Terraform Variables
2. Terraform Modules
3. Terraform Outputs
4. Terraform Functions

Question:

Matthew is new to terraform. He is creating an EC2 instance.
To fetch the Public IP of the EC2, Matthew needs to manually login to the EC2 console and search for EC2 or search through Terraform state file.
Matthew wants to IP address of the instance to show in output automatically once the EC2 instance is created. Which terraform feature can be used?

1. Terraform Variables
2. Terraform Modules
3. **Terraform Outputs**
4. Terraform Functions

Question:

Alice is writing a VPC module in AWS and a variable needs to be defined in the following format:

1. az=["ap-south-1a","ap-south-1b"]

What is the data type that alice can use to match this type of data?

1. Bool
2. String
3. Number
4. List
5. Null

Question:

Alice is writing a VPC module in AWS and a variable needs to be defined in the following format:

1. az=["ap-south-1a","ap-south-1b"]

What is the data type that alice can use to match this type of data?

1. Bool
2. String
3. Number
4. **List**
5. Null

Closure