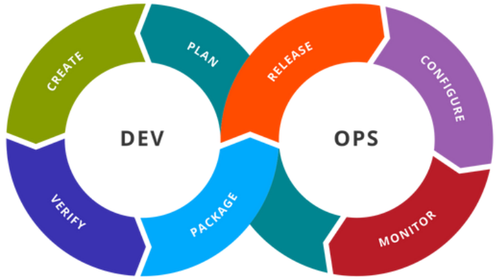
Terraform Associate 2023

Presented By:
Rashi Rana

About Me



Xebia
CREATING DIGITAL LEADERS

ORACLE

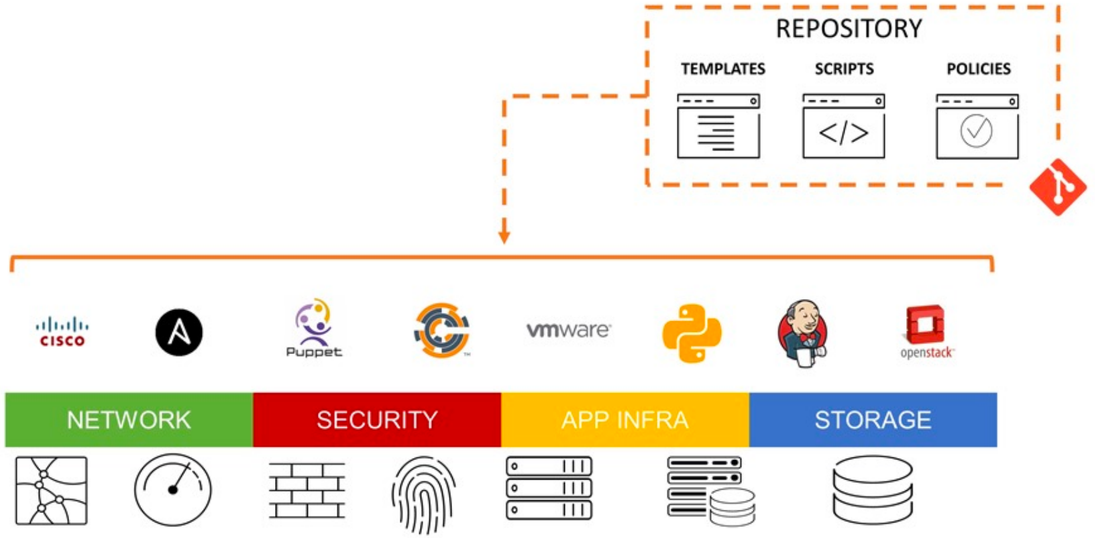
Red Hat

amazon


So. why don't you tell me about yourself?

What is IaC?

Infrastructure as Code (IaC) is the managing and provisioning of infrastructure through code instead of through manual processes.



Exploring Toolsets

There are various types of tools that can allow you to deploy IaC

- Terraform
- CloudFormation
- Heat
- Ansible
- SaltStack
- Chef
- Puppet and others

Why Terraform?

i) Supports multiple platforms, has hundreds of providers.

ii) Simple configuration language and faster learning curve.

iii) Easy integration with configuration management tools like Ansible.

iv) Easily extensible with the help of plugins.

v) Free !!!

Providers Documentation

https://registry.terraform.io/browse/providers

Installating Terraform

https://developer.hashicorp.com/terraform/downloads

Choosing IDE For Terraform

Terraform Code in NotePad!

You can write Terraform code in Notepad and it will not have any impact.

Downsides:

- Slower Development
- Limited Features

*test.tf - Notepad

File Edit Format View Help

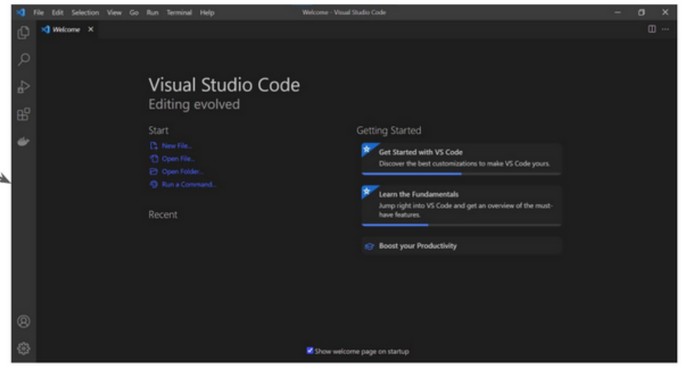
```
variable "elb_names" {
  type = list
  default = ["dev-loadbalancer"]
}

resource "aws_iam_user" "lb" {
  name = var.elb_names[count.index]
  count = 2
  path = "/system/"
}
```

Other Options

There are many popular source code editors available in the market.

Source Code Editors



VS Code Extensions

Extensions are add-ons that allow you to customize and enhance your experience in Visual Studio by adding new features or integrating existing tools

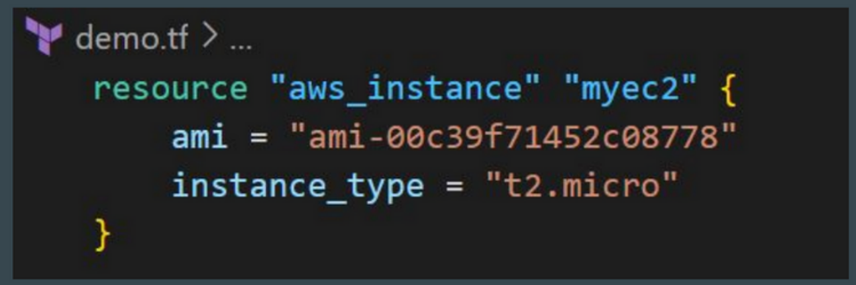
They offer wide range of functionality related to colors, auto-complete, report spelling errors etc.

Terraform Extension

HashiCorp also provides extension for Terraform for Visual Studio Code.

demo.tf

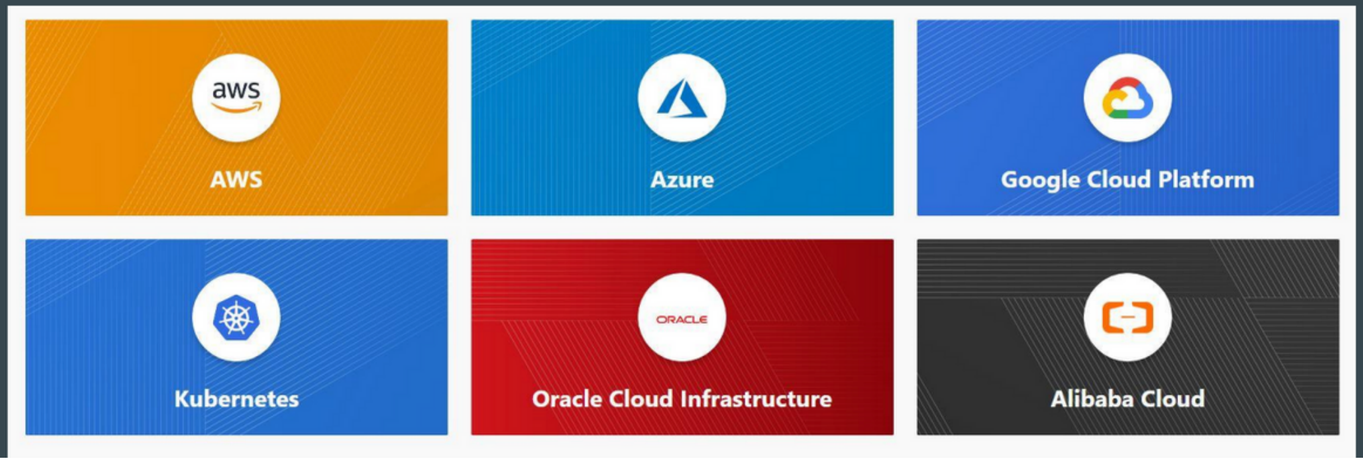
```
resource "aws_instance" "myec2" {
    ami = "ami-00c39f71452c08778"
    instance_type = "t2.micro"
}
```



Basics of Providers

Terraform supports multiple providers.

Depending on what type of infrastructure we want to launch, we have to use appropriate providers accordingly.



Setting up the Lab

Let’s start Rolling !

Deploy first ec2 instance using terraform

Learning 1 - Provider Plugins

A provider is a plugin that lets Terraform manage an external API.

When we run terraform init, plugins required for the provider are automatically downloaded and saved locally to a .terraform directory.

```
C:\Users\zealv\Desktop\kplabs-terraform>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.60.0...
- Installed hashicorp/aws v4.60.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Learning 2 - Resource

Resource block describes one or more infrastructure objects

Example:

- resource aws_instance
- resource aws_alb
- resource iam_user
- resource digitalocean_droplet

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

Learning 3 - Resource Blocks

A resource block declares a resource of a given type ("aws_instance") with a given local name ("myec2"). Resource type and Name together serve as an identifier for a given resource and so must be unique.

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

EC2 Instance Number 1

```
resource "aws_instance" "web" {
  ami           = ami-123
  instance_type = "t2.micro"
}
```

EC2 Instance Number 2

Point to Note

You can only use the resource that are supported by a specific provider. In the below example, provider of Azure is used with resource of aws_instance

```
provider "azurerm" {}

resource "aws_instance" "web" {
  ami           = ami-123
  instance_type = "t2.micro"

}
```

Important Question

The core concepts, standard syntax remains similar across all providers.

If you learn the basics, you should be able to work with all providers easily.

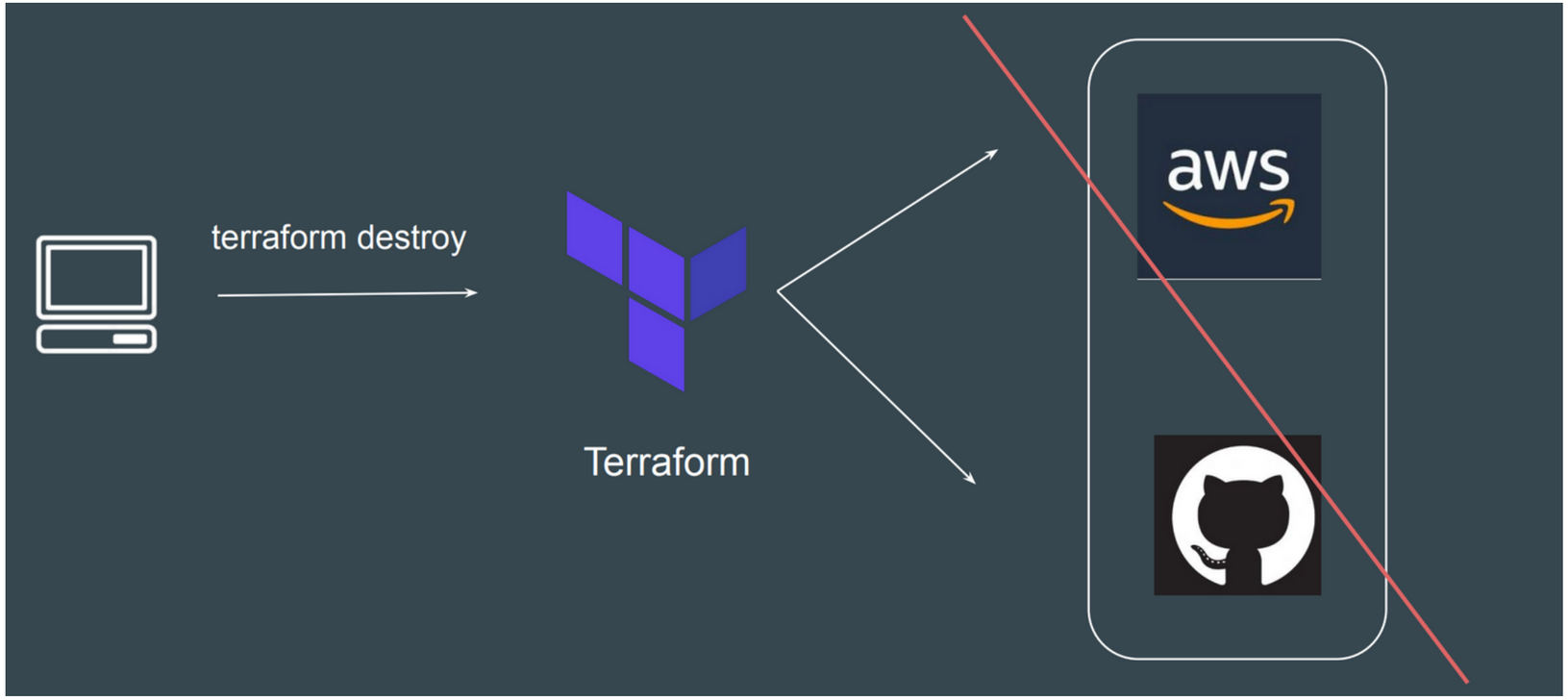
Issues and Bugs with Providers

A provider that is maintained by HashiCorp does not mean it has no bugs. It can happen that there are inconsistencies from your output and things mentioned in documentation. You can raise issue at Provider page.

Terraform Destroy

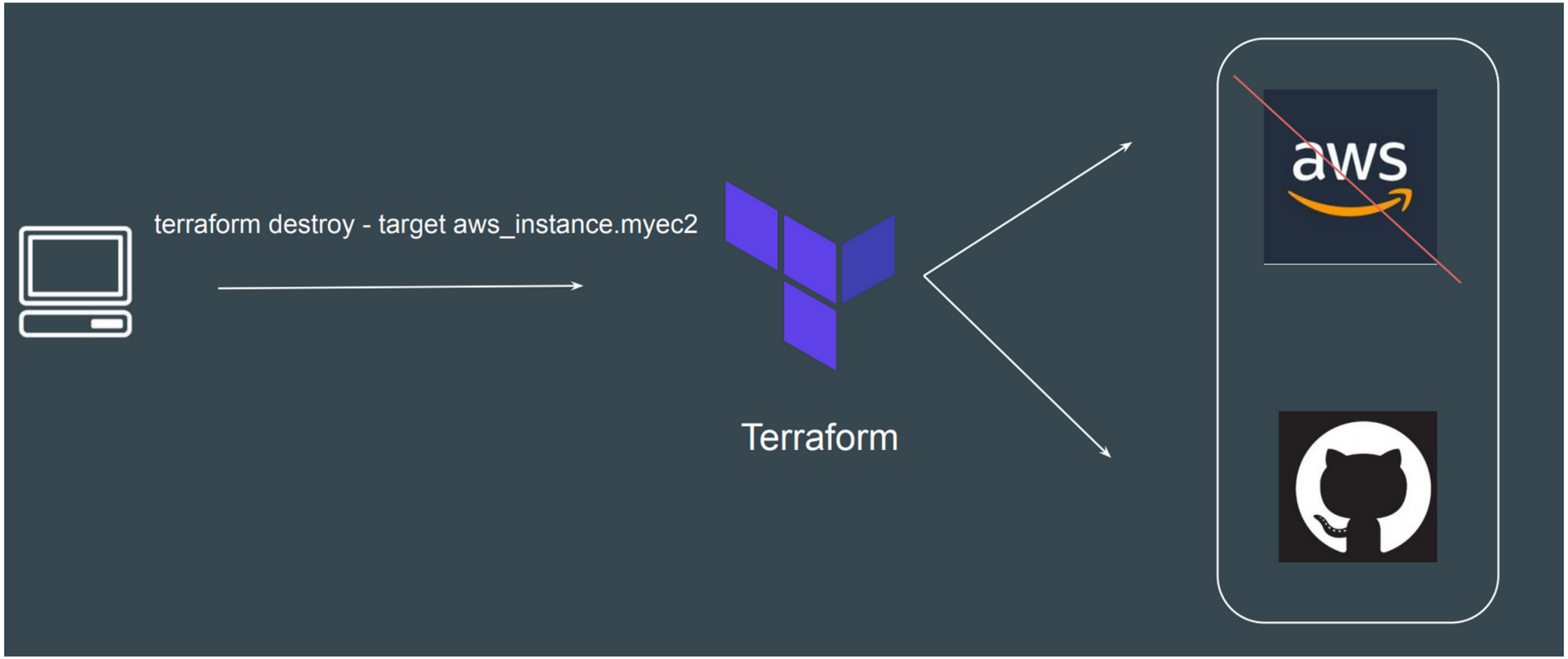
Approach 1 - Destroy ALL

terraform destroy allows us to destroy all the resource that are created within the
folder.



Approach 2 - Destroy Some

terraform destroy with -target flag allows us to destroy specific resource.



Terraform Destroy with Target

The -target option can be used to focus Terraform's attention on only a subset of resources.

Combination of: Resource Type + Local Resource Name

EX: **terraform destroy -target aws_instance.myec2**

Task 1

1. **Use GitHub provider and create a repo.**
2. **Paste the screenshot of the plan in the chat.**

Task 2

1. **Create an ec2 instance**
2. **Paste the Screenshot from the terraform plan in the chat**

Task 3

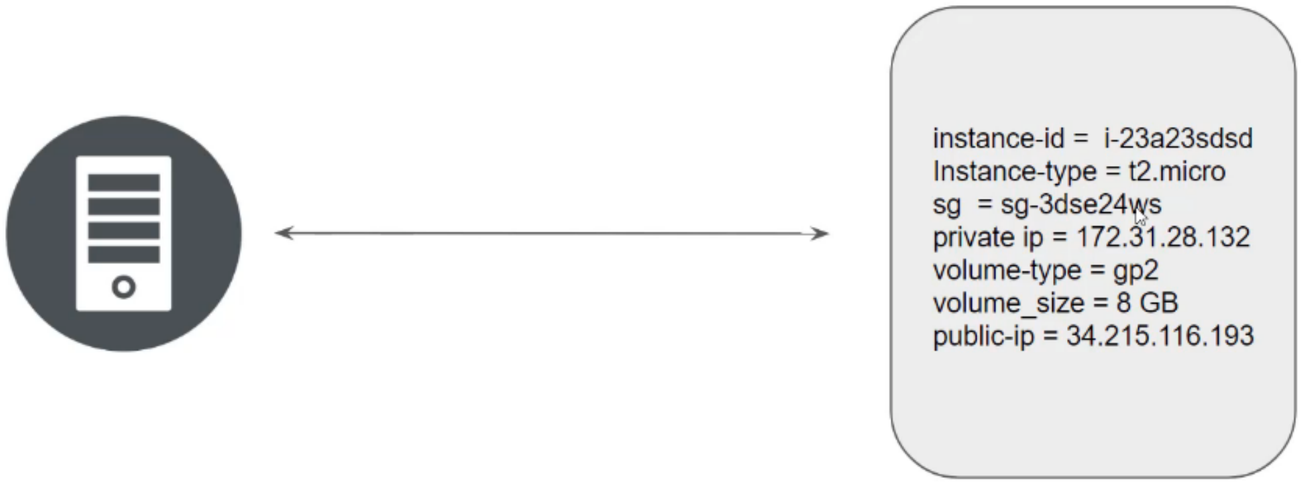
1. **Target destroy GitHub repo**
2. **Paste the Screenshot from the terraform plan and the command used in the chat**

Understanding the state file

State File

Terraform stores the state of the infrastructure that is being created from the TF files.

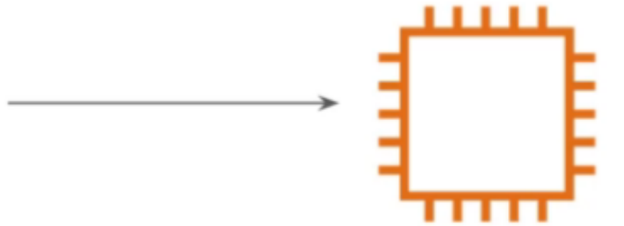
This state allows Terraform to map real-world resources to your existing configuration.



Desired & Current state

Terraform's primary function is to create, modify and destroy infrastructure resources to match the desired state described in a terraform configuration.

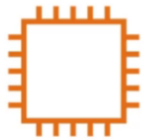
```
resource "aws_instance" "myec2" {
   ami = "ami-082b5a644766e0e6f"
   instance_type = "t2.micro"
}
```



EC2 - t2.micro

Desired vs Current

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



EC2 - t2.micro

Desired

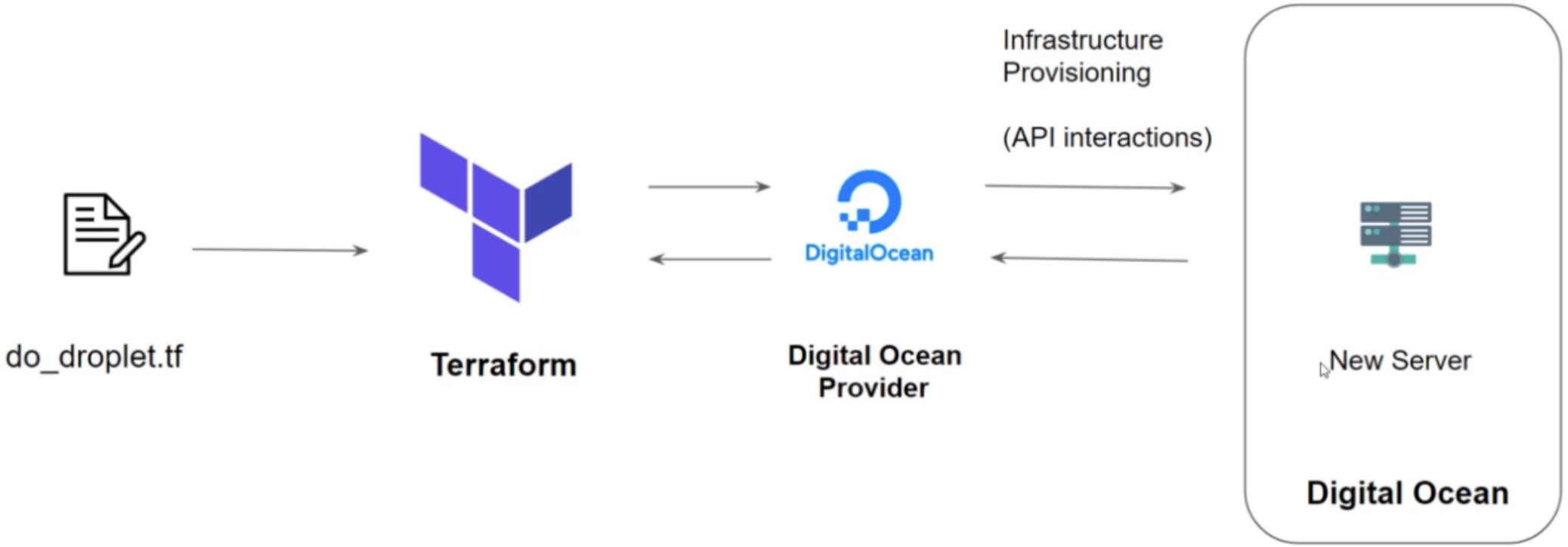
```
resource "aws_instance" "myec2" {
   ami = "ami-082b5a644766e0e6f"
   instance_type = "t2.micro"
}
```

t2.medium

Current

Provider Versioning

Provider Architecture



Overview of Provider Versioning

Provider plugins are released separately from Terraform itself.

They have a different set of version numbers.

Explicitely Setting Provider Version

During terraform init, if the version argument is not specified, the most recent provider will be downloaded during initialization.

For production use, you should constrain the acceptable provider versions via configuration, to ensure that new versions with breaking changes will not be automatically installed.

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
}

provider "aws" {
  region = "us-east-1"
}
```

Arguments for Specifying provider

There are multiple ways for specifying the version of a provider.

Version Number Arguments	Description
>=1.0	Greater than equal to the version
<=1.0	Less than equal to the version
~>2.0	Any version in the 2.X range.
>=2.10,<=2.30	Any version between 2.10 and 2.30

Dependency Lock File

Terraform dependency lock file allows us to lock to a specific version of the provider.

If a particular provider already has a selection recorded in the lock file, Terraform will always re-select that version for installation, even if a newer version has become available.

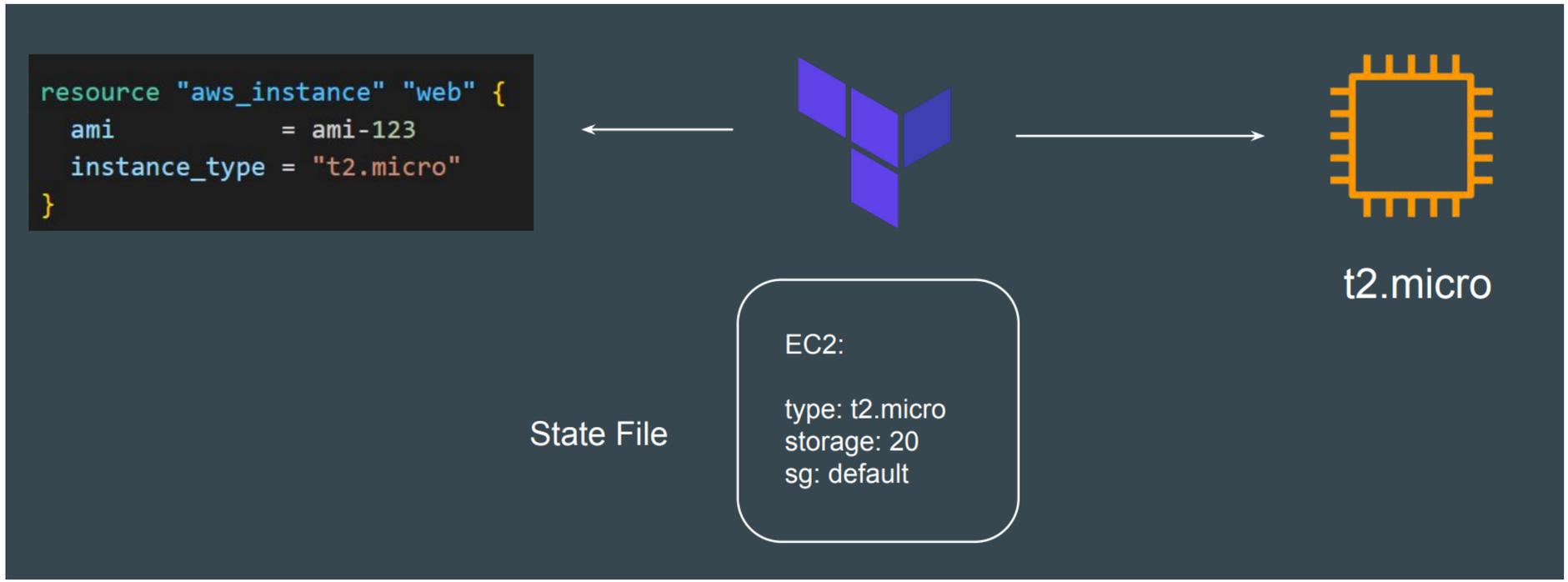
You can override that behavior by adding the -upgrade option when you run terraform init,

```
provider "registry.terraform.io/hashicorp/aws" {
  version     = "2.70.0"
  constraints = ">= 2.31.0, <= 2.70.0"
  hashes = [
    "h1:fx8tbGVwK1YIDI6UdHLnorC9PA1ZPSWEeW3V3aDCdWY=",
    "zh:01a5f351146434b418f9ff8d8cc956ddc801110f1cc8b139e01be2ff8c544605",
    "zh:1ec08abbaf09e3e0547511d48f77a1e2c89face2d55886b23f643011c76cb247",
    "zh:606d134fef7c1357c9d155aadbee6826bc22bc0115b6291d483bc1444291c3e1",
    "zh:67e31a71a5ecbbc96a1a6708c9cc300bbfe921c322320cdbb95b9002026387e1",
```

Terraform Refresh

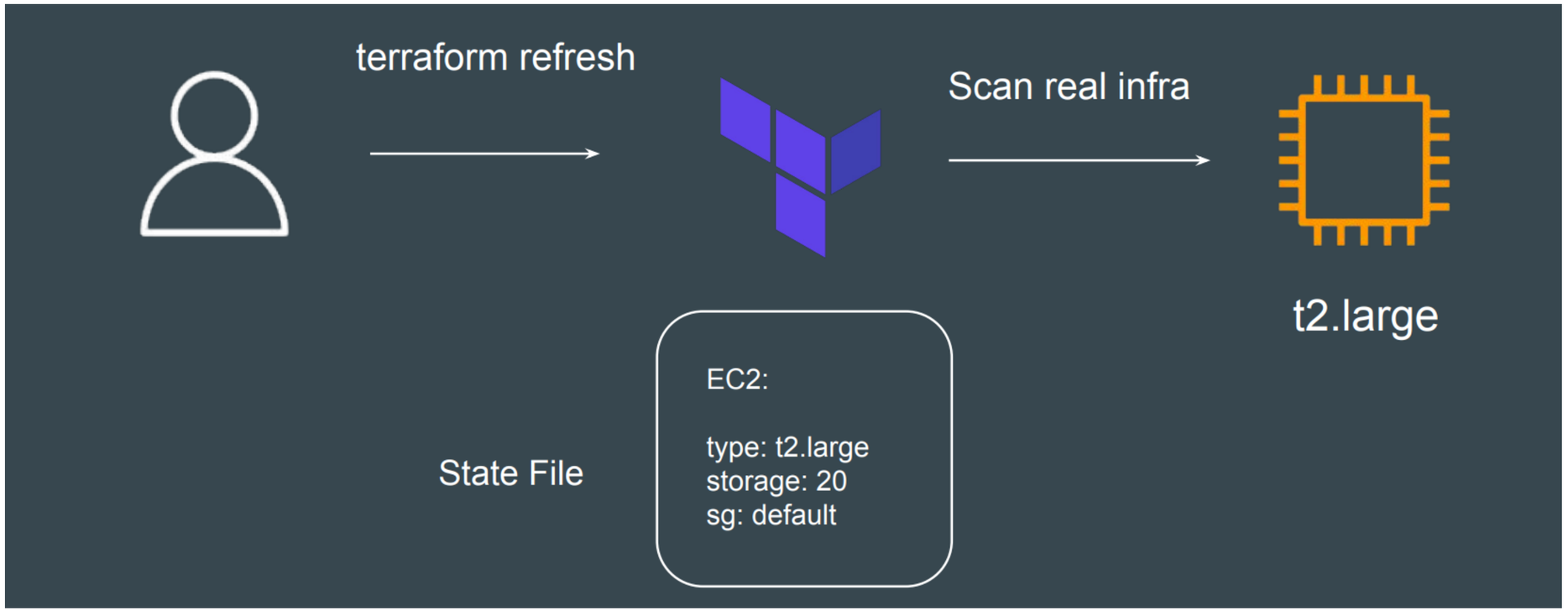
Understanding the Challenge

Terraform can create an infrastructure based on configuration you specified. It can happen that the infrastructure gets modified manually.



Understanding the Challenge

The terraform refresh command will check the latest state of your infrastructure and update the state file accordingly.



Important

You shouldn't typically need to use this command, because Terraform automatically performs the same refreshing actions as a part of creating a plan in both the Terraform plan and Terraform apply commands.

Lets see this practically!!

Quiz

Question 1:

Based on the following terraform configuration, what is the local name associated with the resource that is defined?

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

1. **aws_instance**
2. **t2.mico**
3. **myec2**
4. **resource**

Quiz

Question 1:

Based on the following terraform configuration, what is the local name associated with the resource that is defined?

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

1. **aws_instance**
2. **t2.mico**
3. **myec2**
4. **resource**

Quiz

Question 1:

Based on the following terraform configuration, what might be the provider that would had been used?

```
resource "aws_instance" "myec2" {
  ami           = "ami-090fa75af13c156b4"
  instance_type = "t2.micro"
}
```

1. **myec2**
2. **aws_instance**
3. **aws**
4. **azure**

Quiz

Question 1:

Based on the following terraform configuration, what might be the provider that would had been used?

```
resource "aws_instance" "myec2" {
  ami           = "ami-090fa75af13c156b4"
  instance_type = "t2.micro"
}
```

1. **myec2**
2. **aws_instance**
3. **AWS**
4. **azure**

Quiz

Question 1:

A running EC2 instance in AWS environment represents which state in terraform?

1. **Desired**
2. **Current**

Quiz

Question 1:

A running EC2 instance in AWS environment represents which state in terraform?

1. **Desired**
2. **Current**

Quiz

Question 1:

What is the name of the file were terraform stores state information?

1. **myec2.tf**
2. **terraform-tfstate**
3. **terraform.tfstate**
4. **none of the above**

Quiz

Question 1:

What is the name of the file were terraform stores state information?

1. **myec2.tf**
2. **terraform-tfstate**
3. **terraform.tfstate**
4. **none of the above**

Quiz

Question 5:

Aashish has created an EC2 instance via Terraform. Aashish has defined the following rules within the security group:

1. **Port 443 allowed from 0.0.0.0/0**
2. **Port 22 allowed from 125.36.50.23/32**

Sneha has added the following rule manually:
Port 80 allowed from 0.0.0.0/0
Next time when Aashish runs a terraform plan, what will happen?

1. **Terraform Plan will show everything is up to date.**
2. **Terraform Plan will show output to remove the manually created rule.**
3. **Terraform Plan will fail since some of the resources were created manually.**
4. **None of the Above.**

Quiz

Question 5:
Aashish has created an EC2 instance via Terraform. Aashish has defined the following rules within the security group:

1. **Port 443 allowed from 0.0.0.0/0**
2. **Port 22 allowed from 125.36.50.23/32**

Sneha has added the following rule manually:
Port 80 allowed from 0.0.0.0/0
Next time when Aashish runs a terraform plan, what will happen?

1. **Terraform Plan will show everything is up to date.**
2. **Terraform Plan will show output to remove the manually created rule.**
3. **Terraform Plan will fail since some of the resources were created manually.**
4. **None of the Above.**

Quiz

Question 6:

There are 3 resources that have been created using Terraform.
1 resource among them has been deleted manually from the AWS console.
What will happen in the next terraform apply operation?

1. **The Deleted resource will be created again.**
2. **Terraform will throw an error.**
3. **No Modification will happen.**
4. **None of the Above.**

Quiz

Question 6:

There are 3 resources that have been created using Terraform.
1 resource among them has been deleted manually from the AWS console.
What will happen in the next terraform apply operation?

1. **The Deleted resource will be created again.**
2. **Terraform will throw an error.**
3. **No Modification will happen.**
4. **None of the Above.**

Quiz

Question 7:

Sneha has created a new EC2 instance with the instance type of t2.micro
It was realized that the instance type of t2.micro is too slow for the application to run and hence it needs to be changed from t2.micro to m5.large
While Sneha is modifying the Terraform configuration with the newer instance type, some one from the Team manually modified the instance type to m5.large from the AWS Console.
When Sneha runs terraform apply operation, what will happen?

1. **The instance will be terminated and new instance will be created.**
2. **Terraform will give an error since the instance is manually modified.**
3. **No resource will be updated as desired state will match the current state.**
4. **None of the Above**

Quiz

Question 7:

Sneha has created a new EC2 instance with the instance type of t2.micro
It was realized that the instance type of t2.micro is too slow for the application to run and hence it needs to be changed from t2.micro to m5.large
While Sneha is modifying the Terraform configuration with the newer instance type, someone from the Team manually modified the instance type to m5.large from the AWS Console.
When Sneha runs terraform apply operation, what will happen?

1. **The instance will be terminated and a new instance will be created.**
2. **Terraform will give an error since the instance is manually modified.**
3. **No resource will be updated as the desired state will match the current state.**
4. **None of the Above**

Quiz

Question 8:

The following resource configuration represents what state in Terraform?

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

1. **Current State**
2. **Desired State**

Quiz

Question 8:

The following resource configuration represents what state in Terraform?

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

1. **Current State**
2. **Desired State**

Task 4

1. **Configure terraform provider to use version >=2.10,<=2.30, and send the screenshot in the chat.**
2. **Configure terraform provider to use the exact version v3.76.1 and send the screenshot in the chat.**
3. **Configure terraform provider to use any version in the range of v4.x and send the screenshot in the chat.**
4. **Configure terraform provider to use the latest version and send the screenshot in the chat.**