



*Terraform* in  
the Cloud



Raju Gandhi  
@looselytyped

O'REILLY®

Head First

# Git

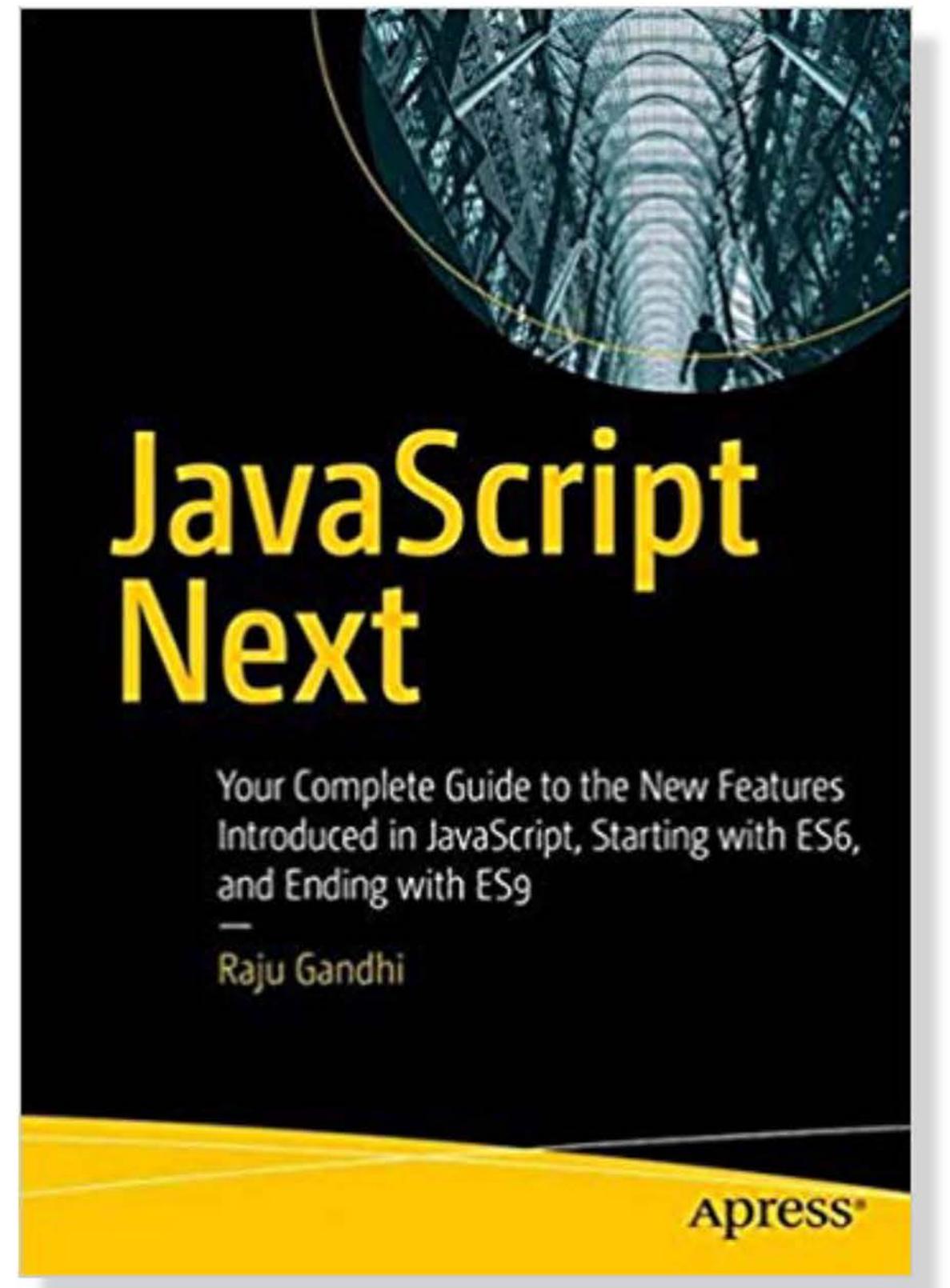
A Learner's Guide  
to Understanding Git  
from the Inside Out

Raju Gandhi



A Brain-Friendly Guide

[HTTPS://LEARNING.OREILLY.COM/LIBRARY/VIEW/HEAD-FIRST-GIT/9781492092506/](https://learning.oreilly.com/library/view/head-first-git/9781492092506/)



Devoops

IT'S  
*about*

better collaboration  
automating everything  
measurements  
extending sdlc

**(POLL - Single Choice)**

## **Embracing DevOps**

- We do it all – Ci/Cd pipelines, Everything-as-Code, Centralized log/event/monitoring etc
- We have a few pieces in the works
- Just getting started

~~FEAR~~ CHANGE

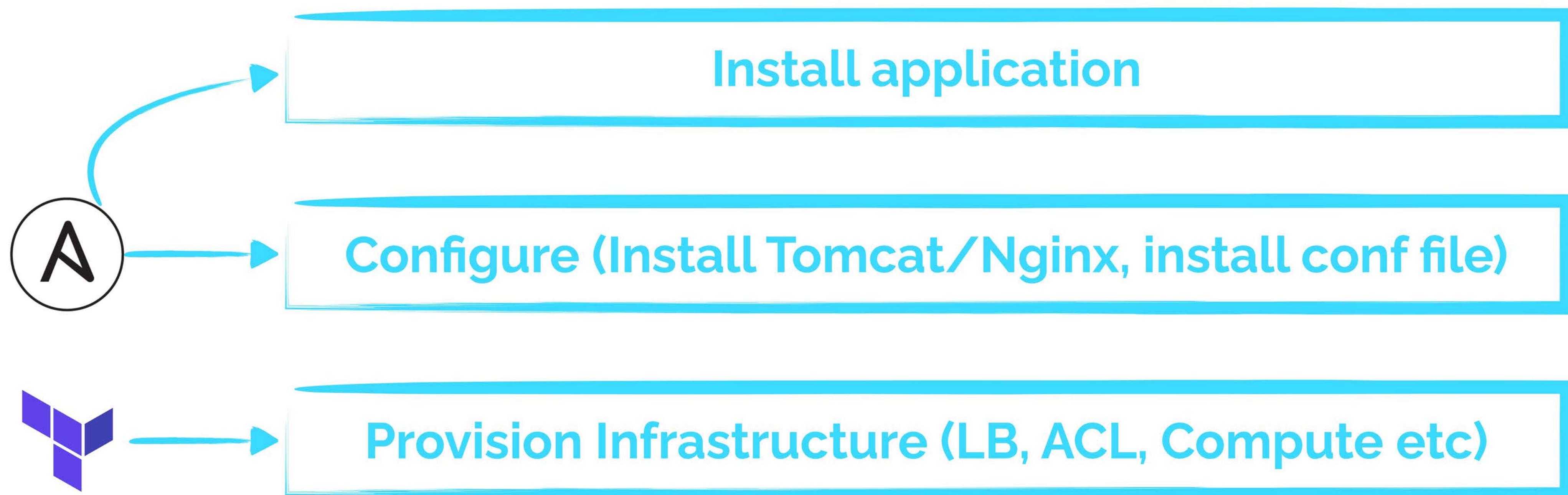
“

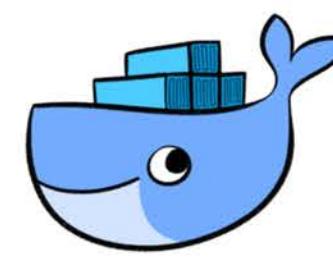
**Infrastructure is all** what is needed to support the **computing** requirements of an **application**.

- @antonbabenko

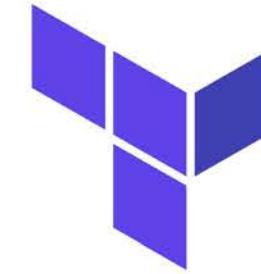
Terraform







**Configuration + Application**

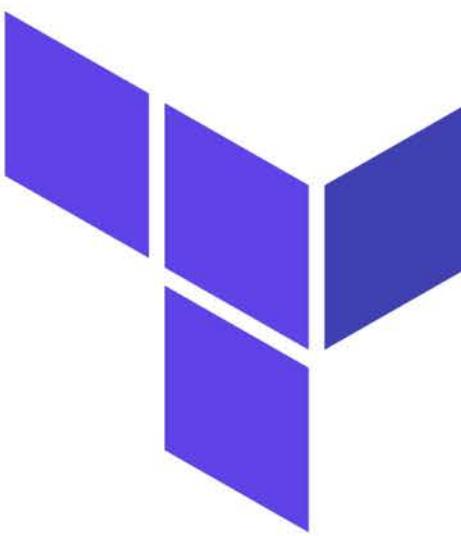


**Provision Infrastructure (LB, ACL, Compute etc)**



+ 130 more

## Providers



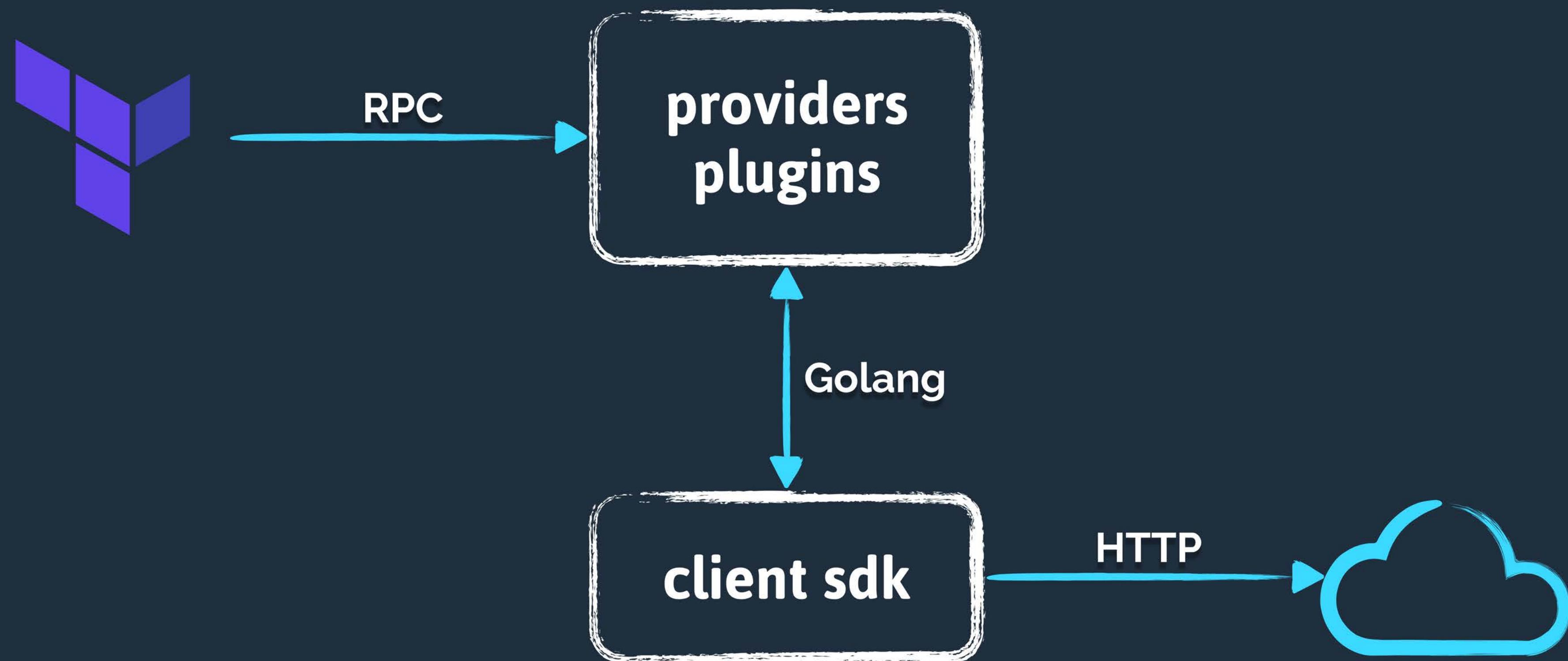


## (POLL - Single Choice)

### Usage of "everything as code" in your organization

- We do it all – IaC (e.g Terraform), CaC (e.g Ansible), DbaC (e.g Liquibase), Builds-as-Code (e.g Jenkinsfile)
- We have a few pieces in the works
- Terraform is our first go at this

*Architecture*



HCL



**Simple**, easy to learn  
In-built **type** system  
Supports **for-loops**, **dynamic blocks**, **conditionals**, string interpolation

## Providers

```
# contains Terraform settings
terraform {
    required_version = "≥ 0.13"
    required_providers {
        aws = {
            # shorthand for registry.terraform.io/hashicorp/aws
            source  = "hashicorp/aws"
            version = "~> 3.0"
        }
    }
}

# block configures the aws provider
provider "aws" {
    region  = "us-east-1"
    profile = "default"
}
```

```
resource "aws_instance" "example" {
    ami          = "ami-07ebfd5b3428b6f4d"
    instance_type = "t2.micro"

    tags = {
        Name = "demo-example"
        Terraform = true
    }
}
```

## Resource



```
resource "aws_instance" "example" {
    ami           = "ami-07ebfd5b3428b6f4d"
    instance_type = "t2.micro"

    tags = {
        Name = "demo-example"
        Terraform = true
    }
}
```

<provider\_type> syntax

## Resource

```
resource "aws_instance" "example" {
    ami           = "ami-07ebfd5b3428b6f4d"
    instance_type = "t2.micro"

    tags = {
        Name = "demo-example"
        Terraform = true
    }
}
```



terraform resource name

## Resource

```
resource "aws_instance" "example" {
    ami          = "ami-07ebfd5b3428b6f4d"
    instance_type = "t2.micro"

    tags = {
        Name = "demo-example"
    }
    Terraform = true
}
```



Name of AWS resource

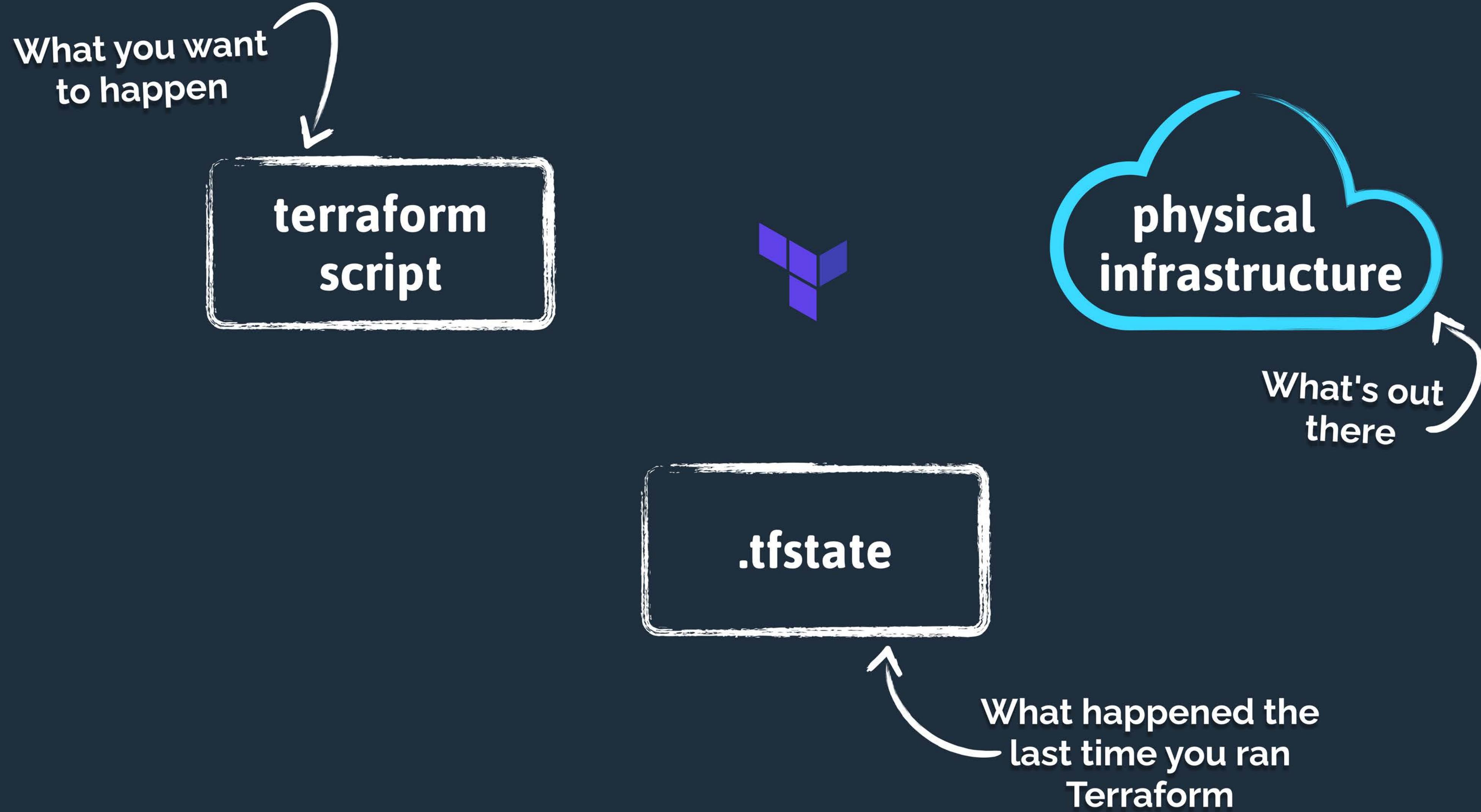
```
resource "aws_instance" "example" {
    ami          = "ami-07ebfd5b3428b6f4d"
    instance_type = "t2.micro"

    tags = {
        Name = "demo-example"
        Terraform = true
    }
}
```



#protip – Tag it appropriately

*Exercise* ++



## References

```
resource "aws_security_group" "sec-8080" {
  name = "terraform-example-instance"

  ingress {
    from_port   = 8080
    to_port     = 8080
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_instance" "example" {
  ami           = "ami-07ebfd5b3428b6f4d"
  instance_type = "t2.micro"
  vpc_security_group_ids = [aws_security_group.sec-8080.id]

  tags = {
    Name = "demo-example"
    Terraform = true
  }
}
```

## References

```
resource "aws_security_group" "sec-8080" {  
    name = "terraform-example-instance"
```

```
    ingress {  
        from_port    = 8080  
        to_port      = 8080  
        protocol     = "tcp"  
        cidr_blocks = ["0.0.0.0/0"]  
    }  
}
```

```
resource "aws_instance" "example" {  
    ami           = "ami-07ebfd5b3428b6f4d"  
    instance_type = "t2.micro"  
    vpc_security_group_ids = [aws_security_group.sec-8080.id]  
  
    tags = {  
        Name = "demo-example"  
        Terraform = true  
    }  
}
```



name the resource

## References

```
resource "aws_security_group" "sec-8080" {
  name = "terraform-example-instance"

  ingress {
    from_port    = 8080
    to_port      = 8080
    protocol     = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_instance" "example" {
  ami           = "ami-07ebfd5b3428b6f4d"
  instance_type = "t2.micro"
  vpc_security_group_ids = [aws_security_group.sec-8080.id]

  tags = {
    Name = "demo-example"
    Terraform = true
  }
}
```

**reference** an existing resource

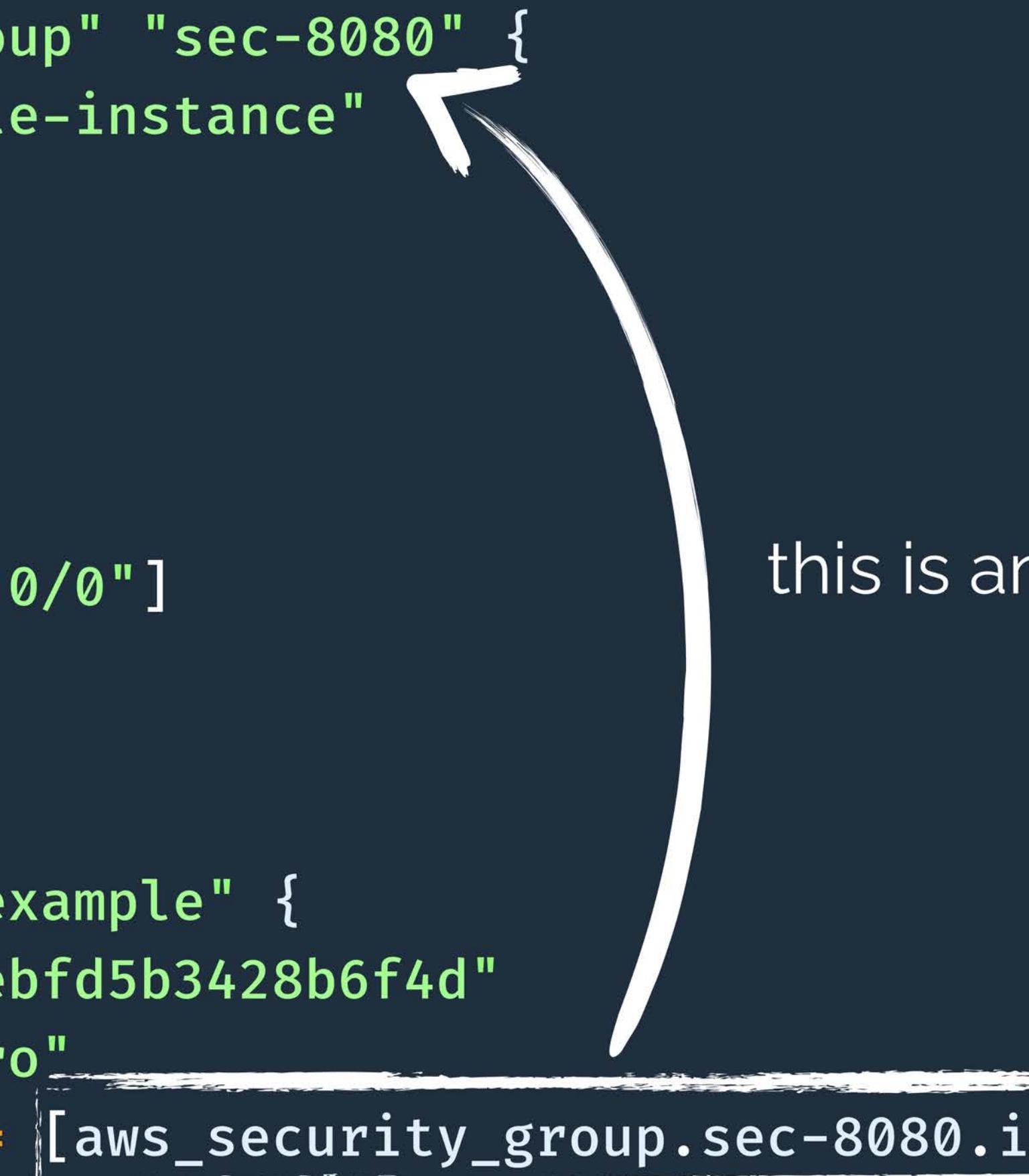
## References

```
resource "aws_security_group" "sec-8080" {
  name = "terraform-example-instance"

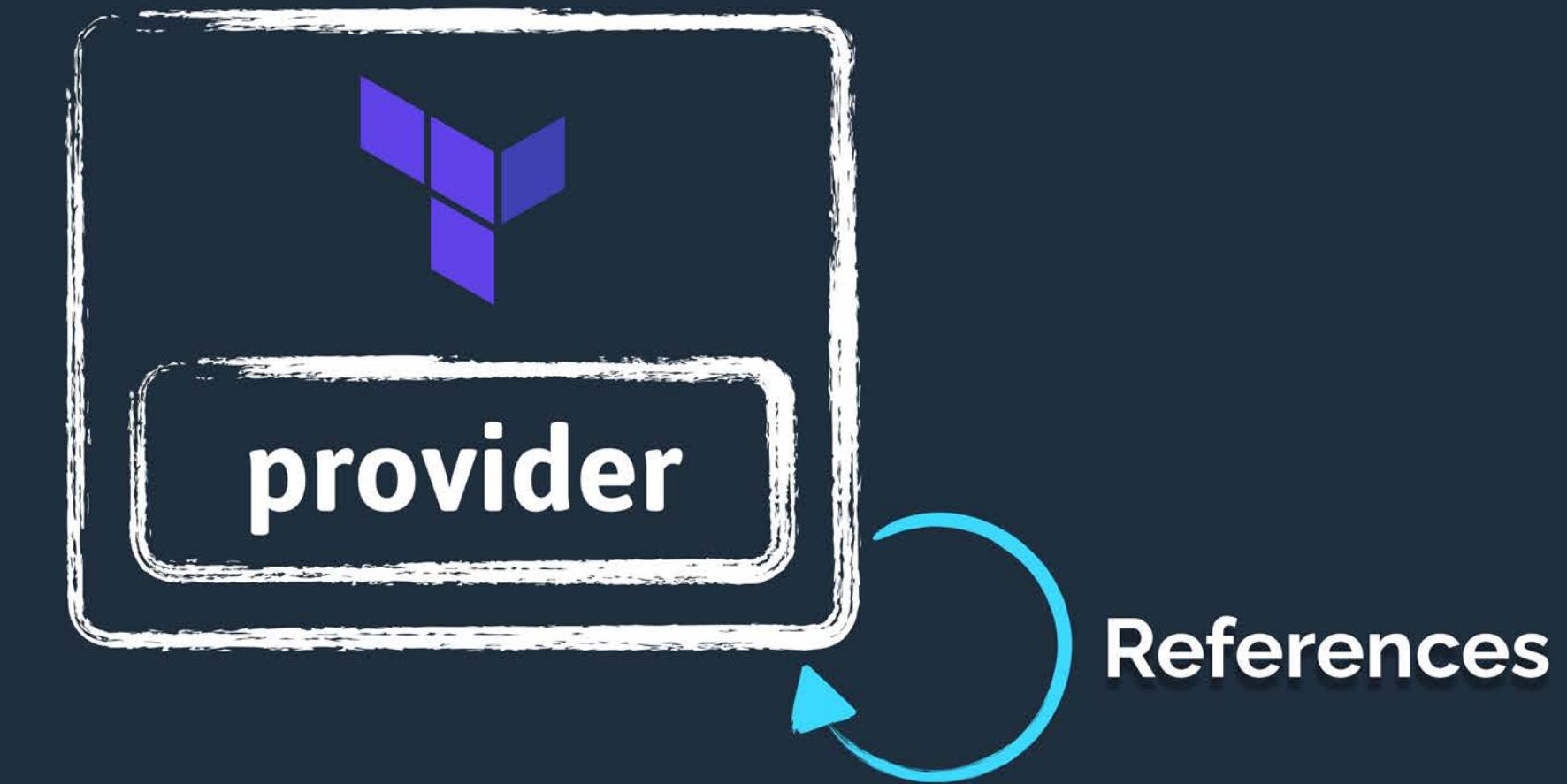
  ingress {
    from_port    = 8080
    to_port      = 8080
    protocol     = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_instance" "example" {
  ami           = "ami-07ebfd5b3428b6f4d"
  instance_type = "t2.micro"
  vpc_security_group_ids = [aws_security_group.sec-8080.id]

  tags = {
    Name = "demo-example"
    Terraform = true
  }
}
```



this is an **implicit** reference



References

*Exercise* ++

## Outputs

```
resource "aws_instance" "example" {
    # truncated for brevity
}

output "example-ip" {
    value      = aws_instance.example.public_ip
    description = "The public IP address of our demo server"
}
```

**output** the value of a variable



## Outputs

```
resource "aws_instance" "example" {
    # truncated for brevity
}

output "example-ip" {
    value      = aws_instance.example.public_ip
    description = "The public IP address of our demo server"
}
```

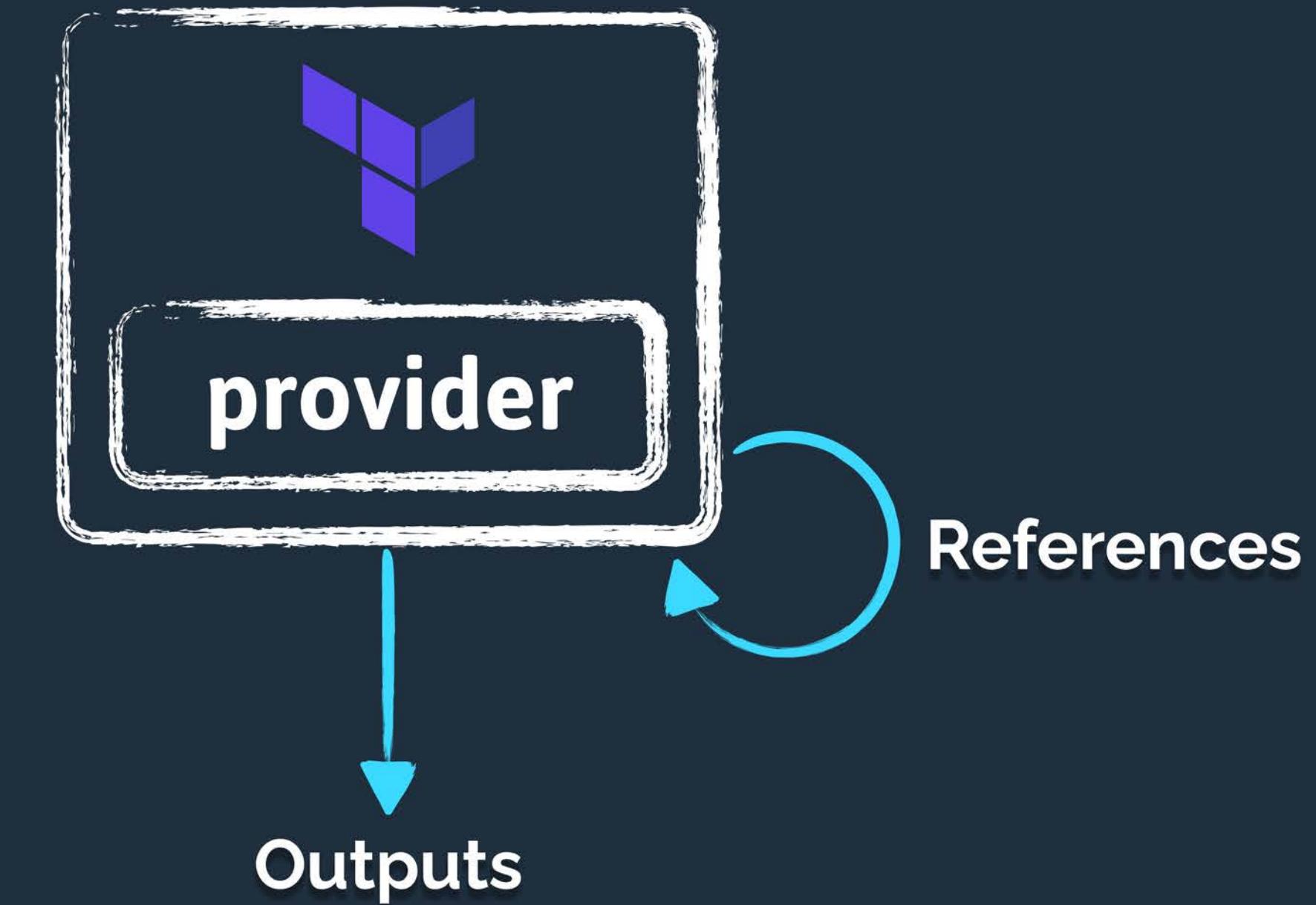
## Outputs

```
resource "aws_instance" "example" {
    # truncated for brevity
}

output "example-ip" {
    value      = aws_instance.example.public_ip
    description = "The public IP address of our demo server"
}
```

```
› terraform apply
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

**Outputs:**  
**example-ip = 3.83.133.229**



*Exercise* ++

## Data

```
data "aws_ami" "ubuntu" {
  most_recent = true

  filter {
    name    = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-*"]
  }

  filter {
    name    = "virtualization-type"
    values = ["hvm"]
  }

  owners = ["099720109477"] # Canonical
}
```

## Data

```
data "aws ami" "ubuntu" {  
  most_recent = true
```

→ **data** lookup

```
filter {  
  name    = "name"  
  values  = ["ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-*"]  
}
```

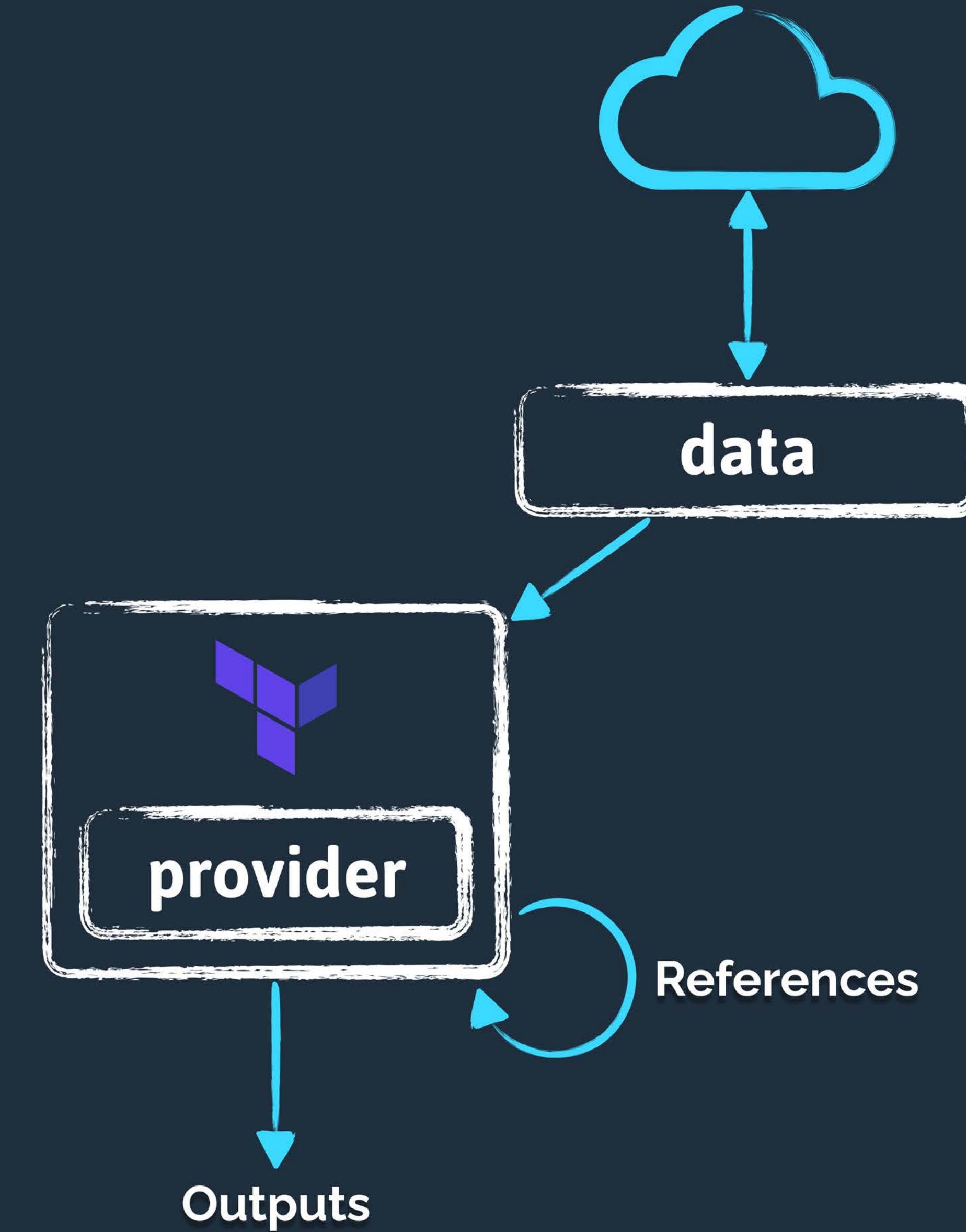
→ **filter** criteria

```
filter {  
  name    = "virtualization-type"  
  values  = ["hvm"]  
}
```

```
owners = ["099720109477"] # Canonical  
}
```

## Data

```
# — Look up a AMI by criteria ——————  
data "aws_ami" "ubuntu" {  
    # ...  
}  
  
# — Defining resources ——————  
resource "aws_instance" "example" {  
    ami                      = data.aws_ami.ubuntu.id  
    instance_type             = "t2.micro"  
    # ...  
}  
  
# — Output useful ids ——————  
output "ami_id" {  
    value      = data.aws_ami.ubuntu.id  
    description = "The image ID we interrogated for Ubuntu AMI"  
}
```



*Exercise* ++

## Variables

```
variable "port" {
  description = "The port to expose on the server"
  type        = number
  default     = 8080
}

resource "aws_security_group" "sec-8080" {
  name = "terraform-example-instance"

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

## Variables

define a variable

```
variable "port" {  
  description = "The port to expose on the server"  
  type        = number  
  default     = 8080  
}  
  
resource "aws_security_group" "sec-8080" {  
  name = "terraform-example-instance"  
  
  ingress {  
    from_port    = var.port  
    to_port      = var.port  
    protocol     = "tcp"  
    cidr_blocks = ["0.0.0.0/0"]  
  }  
}
```

## Variables

```
variable "port" {  
  description = "The port to expose on the server"  
  type = number  
  default = 8080  
}
```



type and default

```
resource "aws_security_group" "sec-8080" {  
  name = "terraform-example-instance"  
  
  ingress {  
    from_port    = var.port  
    to_port      = var.port  
    protocol     = "tcp"  
    cidr_blocks = ["0.0.0.0/0"]  
  }  
}
```

## Variables

```
variable "port" {  
  description = "The port to expose on the server"  
  type        = number  
  default     = 8080  
}
```

```
resource "aws_security_group" "sec-8080" {  
  name = "terraform-example-instance"  
  
  ingress {  
    from_port    = var.port  
    to_port      = var.port  
    protocol     = "tcp"  
    cidr_blocks = ["0.0.0.0/0"]  
  }  
}
```



use the **variable**

## Variables

```
variable "port" {
  description = "The port to expose on the server"
  type        = number
  default     = 8080
}

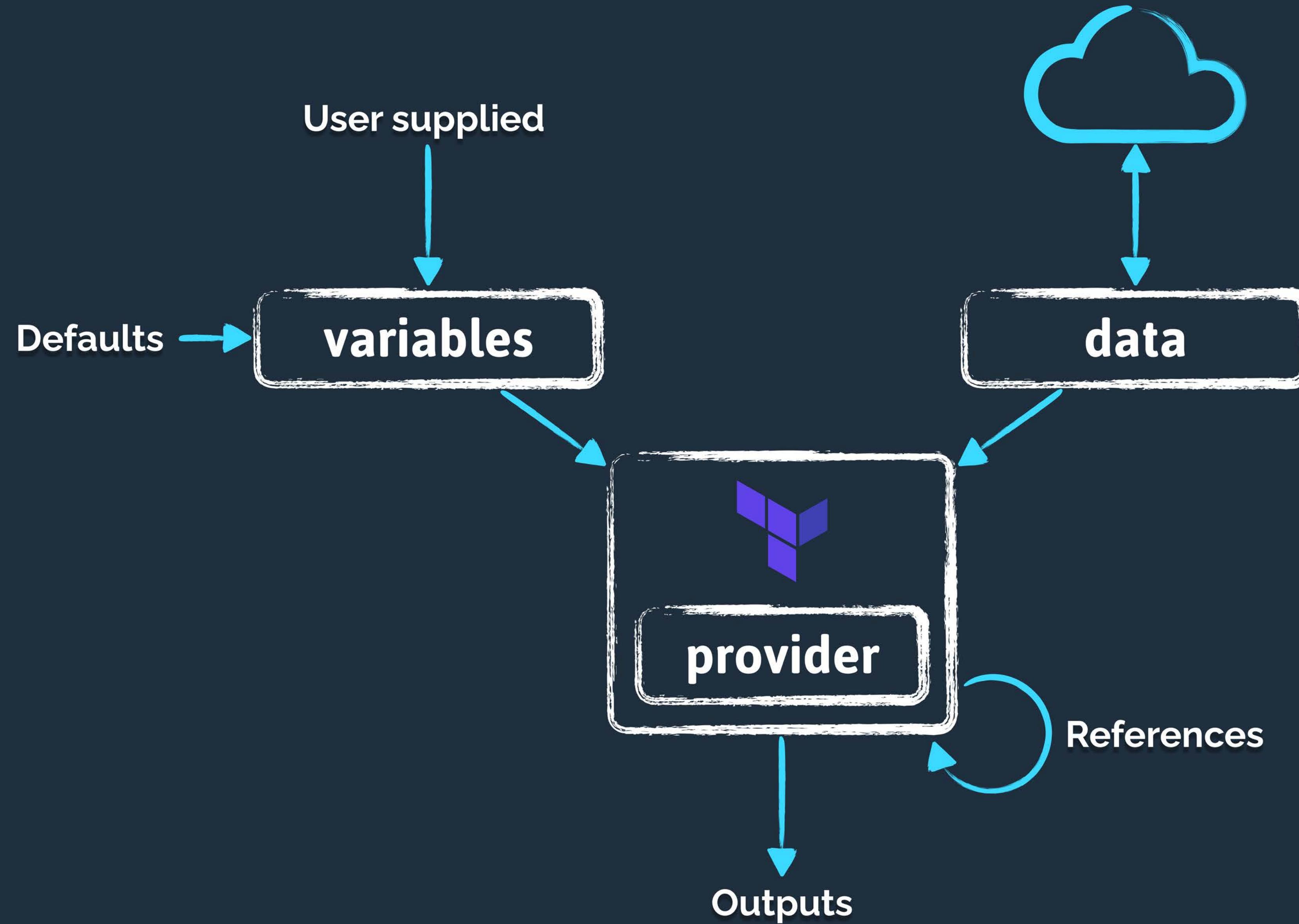
resource "aws_security_group" "sec-8080" {
  name = "terraform-example-instance"

  ingress {
    from_port    = var.port
    to_port      = var.port
    protocol     = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

> terraform apply -var="port=8081"
```



**override** the variable at runtime



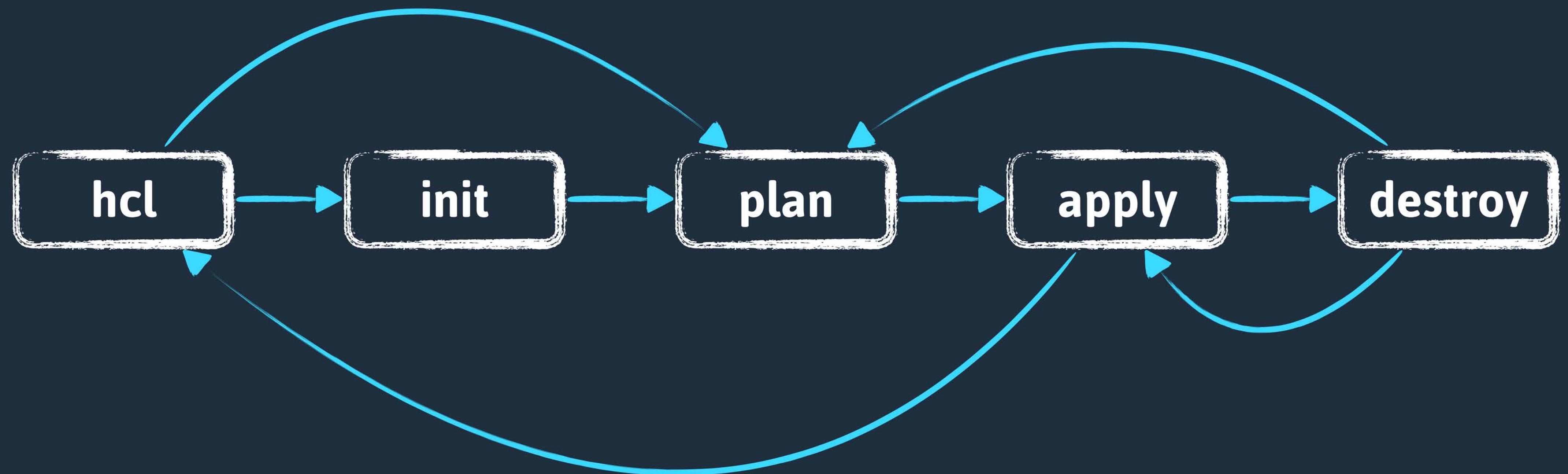
*Exercise* ++

## (POLL - Single Choice)

### Who owns what?

- Development own everything — Ops is focused on SRE
- OPS manages core infra — Development is only responsible for the application layer
- OPS manages everything
- Other (Post in chat)

CLI > \_



State

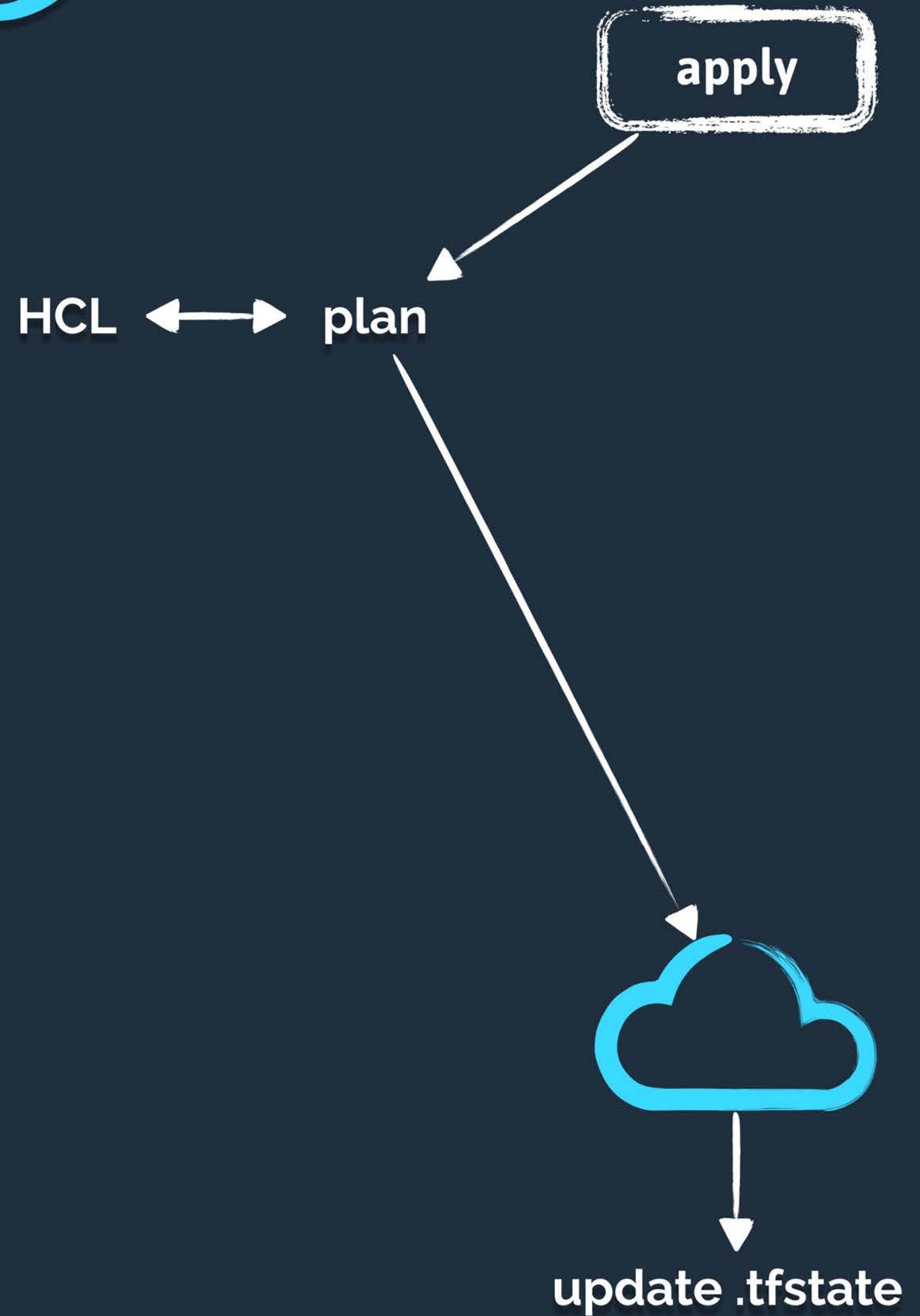


*State*

The **secret sauce** of Terraform

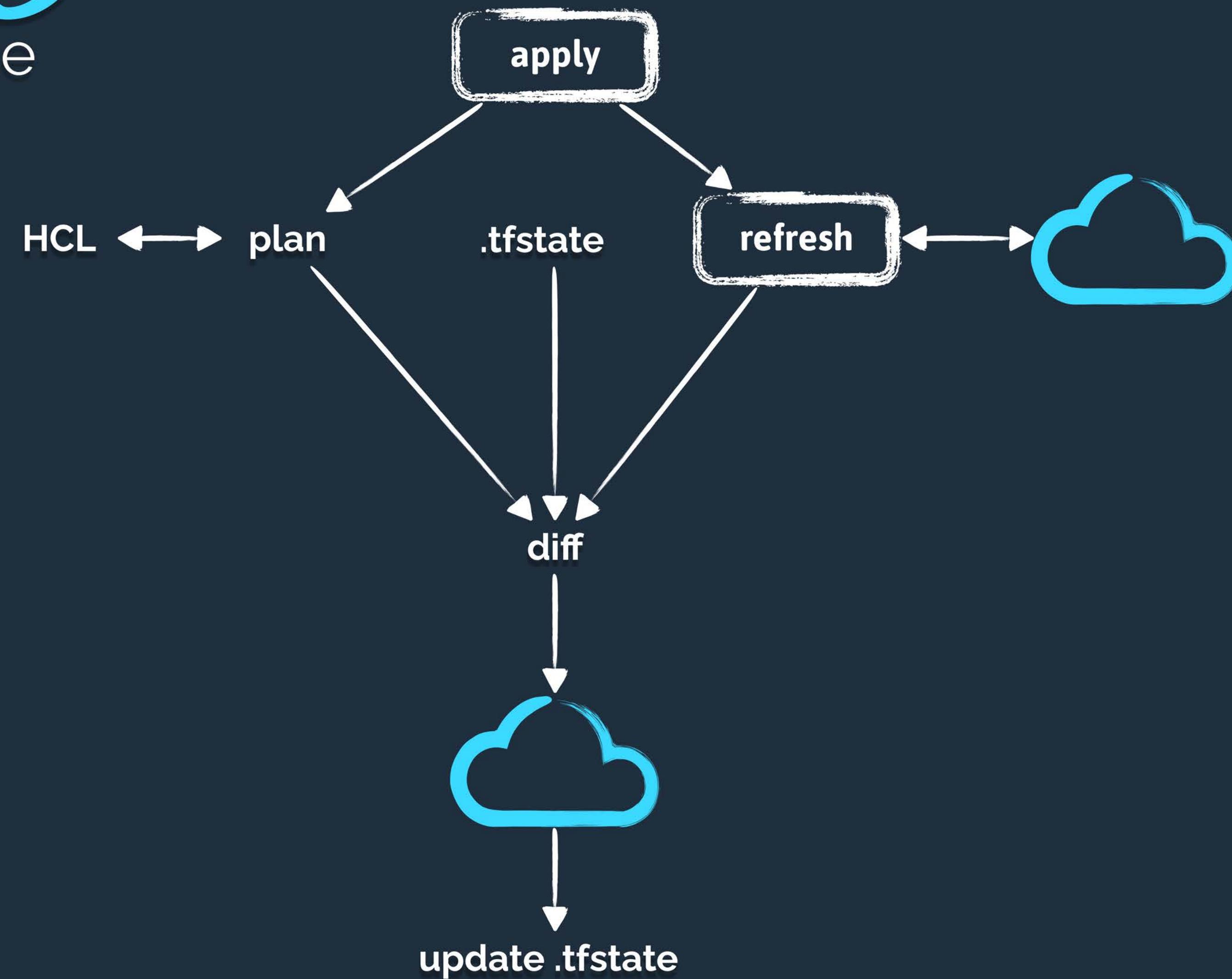
# State

First time



# State

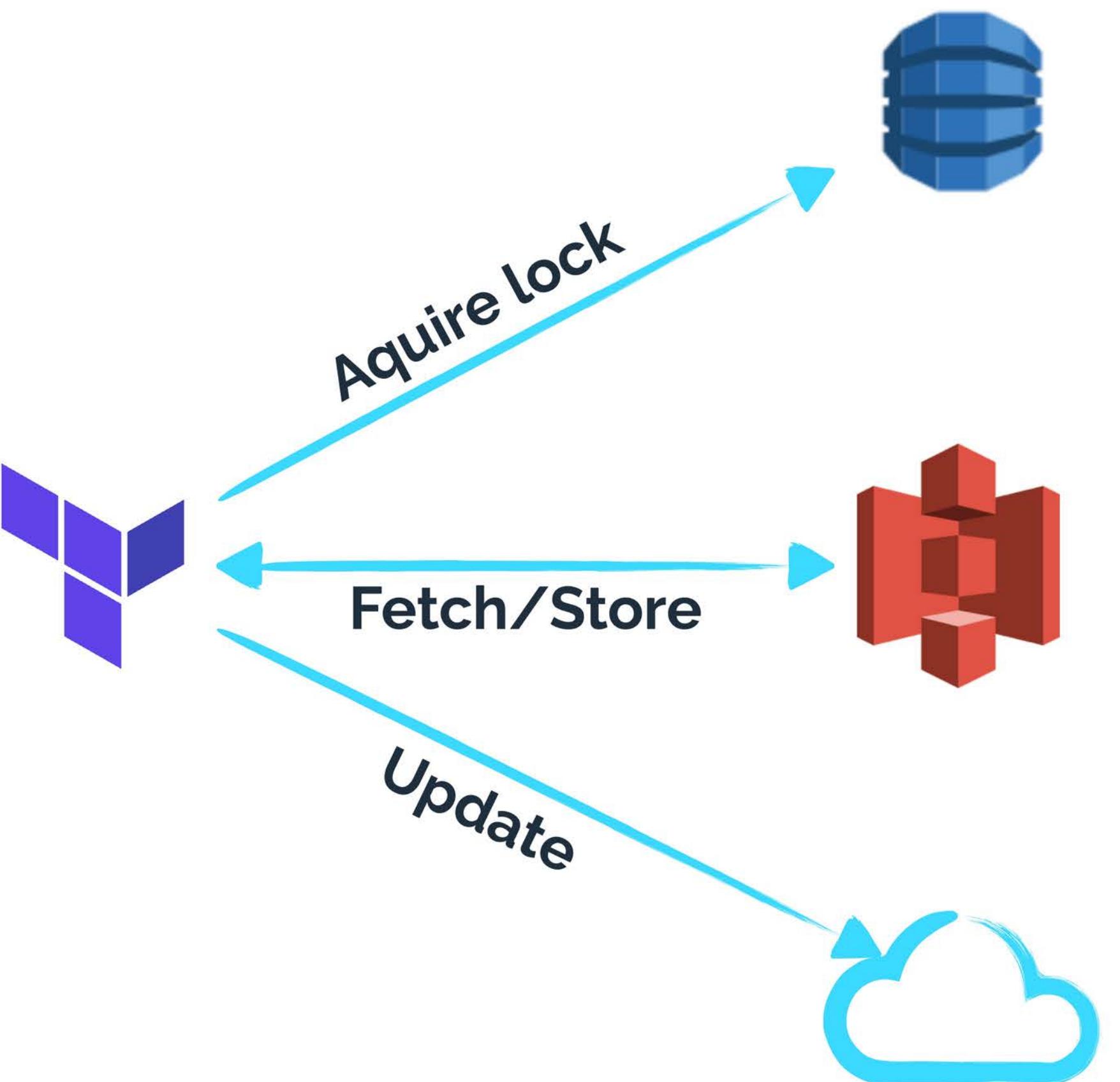
Every other time



```
{
  "version": 4,
  "terraform_version": "0.15.1",
  "serial": 4,
  "lineage": "672708eb-8c19-0463-9eb0-81e20b008a9d",
  "outputs": {},
  "resources": [
    {
      "mode": "managed",
      "type": "aws_instance",
      "name": "exercise_0020",
      "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
      "instances": [
        {
          "schema_version": 1,
          "attributes": {
            "ami": "ami-085925f297f89fce1",
            "arn": "arn:aws:ec2:us-east-1:641995674308:instance/i-0f5f630af308f2516",
            // truncated for brevity
            "vpc_security_group_ids": [
              "sg-02447ae77b0fda694",
              "sg-02a4ff78bfae29ce2"
            ]
          },
          "dependencies": [
            "aws_security_group.sec_http",
            "aws_security_group.sec_ssh"
          ]
        }
      ]
    }
  ]
}
```

```
{
  "version": 4,
  "terraform_version": "0.15.1",
  "serial": 4,
  "lineage": "672708eb-8c19-0463-9eb0-81e20b008a9d",
  "outputs": {},
  "resources": [
    {
      "mode": "managed",
      "type": "aws_instance",
      "name": "exercise_0020",
      "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
      "instances": [
        {
          "schema_version": 1,
          "attributes": {
            "ami": "ami-085925f297f89fce1",
            "arn": "arn:aws:ec2:us-east-1:641995674308:instance/i-0f5f630af308f2516",
            // truncated for brevity
            "vpc_security_group_ids": [
              "sg-02447ae77b0fda694",
              "sg-02a4ff78bfae29ce2"
            ]
          },
          "dependencies": [
            "aws_security_group.sec_http",
            "aws_security_group.sec_ssh"
          ]
        }
      ]
    }
  ]
}
```

```
{  
  "version": 4,  
  "terraform_version": "0.15.1",  
  "serial": 4,  
  "lineage": "672708eb-8c19-0463-9eb0-81e20b008a9d",  
  "outputs": {},  
  "resources": [  
    {  
      "mode": "managed",  
      "type": "aws_instance",  
      "name": "exercise_0020",  
      "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",  
      "instances": [  
        {  
          "schema_version": 1,  
          "attributes": {  
            "ami": "ami-085925f297f89fce1",  
            "arn": "arn:aws:ec2:us-east-1:641995674308:instance/i-0f5f630af308f2516",  
            // truncated for brevity  
            "vpc_security_group_ids": [  
              "sg-02447ae77b0fda694",  
              "sg-02a4ff78bfae29ce2"  
            ]  
          }  
        }  
        {  
          "dependencies": [  
            "aws_security_group.sec_http",  
            "aws_security_group.sec_ssh"  
          ]  
        }  
      ]  
    }  
  ]  
}
```



```
resource "aws_s3_bucket" "my_terraform_state" {
  bucket = "looselytyped.com.nfjs.terraforming-your-cloud-state-demo"

  # Prevent accidental deletion of this S3 bucket
  # If you decide to destroy this be sure to destroy my_terraform_locks as well
  lifecycle {
    prevent_destroy = true
  }

  # Enable versioning so we can see the full revision history of our
  # state files
  versioning {
    enabled = true
  }

  # Enable server-side encryption by default
  server_side_encryption_configuration {
    rule {
      apply_server_side_encryption_by_default {
        sse_algorithm = "AES256"
      }
    }
  }
}
```

```
resource "aws_dynamodb_table" "my_terraform_locks" {
    name          = "terraforming-your-cloud-state-demo"
    billing_mode = "PAY_PER_REQUEST"
    hash_key      = "LockID"

    # Prevent accidental deletion of this table
    # If you decide to destroy this be sure to destroy my_terraform_state as well
    lifecycle {
        prevent_destroy = true
    }

    attribute {
        name = "LockID"
        type = "S"
    }
}
```



HAS to be this name

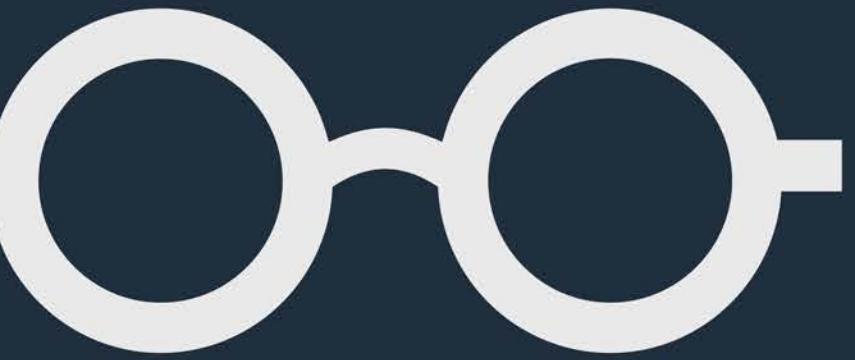
```
terraform {
  backend "s3" {
    bucket          = "looselytyped.com.nfjs.terraforming-your-cloud-state-demo"
    key             = "looselytyped/dev/terraform.tfstate"
    region          = "us-east-1"

    dynamodb_table = "terraforming-your-cloud-state-demo"
    encrypt         = true
  }
}

output "s3_bucket_arn" {
  value      = aws_s3_bucket.my_terraform_state.arn
  description = "The ARN of the S3 bucket"
}

output "dynamodb_table_name" {
  value      = aws_dynamodb_table.my_terraform_locks.name
  description = "The name of the DynamoDB table"
}
```

Demo

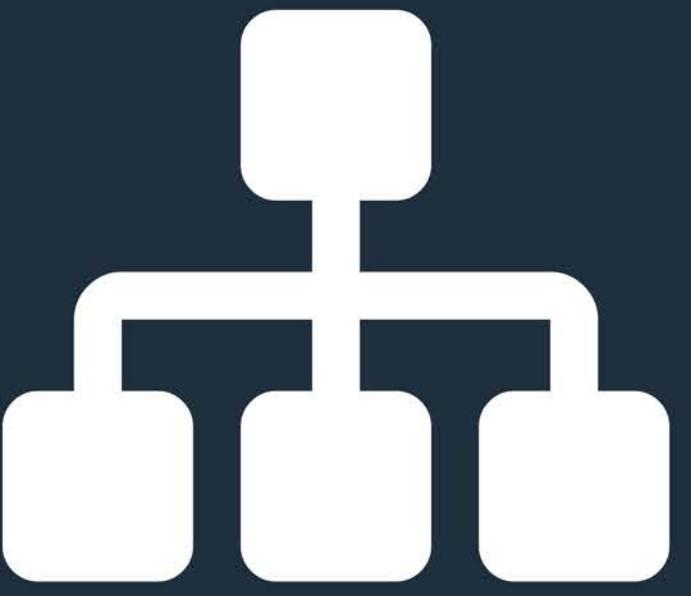


## (POLL - Single Choice)

### CI/CD strategy

- We use one set of tooling (Gitlab/AWS/Azure DevOps/Jenkins) for both CI/CD
- We separate responsibilities (e.g Jenkins for CI, Octopus for CD/ promotions)
- Other (Post in chat)

# Organization



One  
FOR  
One

```
base/
└── base.tfvars
    └── outputs.tf

dev/
├── locals.tf
├── main.tf
├── providers.tf
└── terraform.tfvars

prod/
├── locals.tf
├── main.tf
├── providers.tf
└── terraform.tfvars

modules/
├── main.tf
└── outputs.tf
    └── variables.tf

env.example
initialize.tf
variables.tf
```

*Modules*

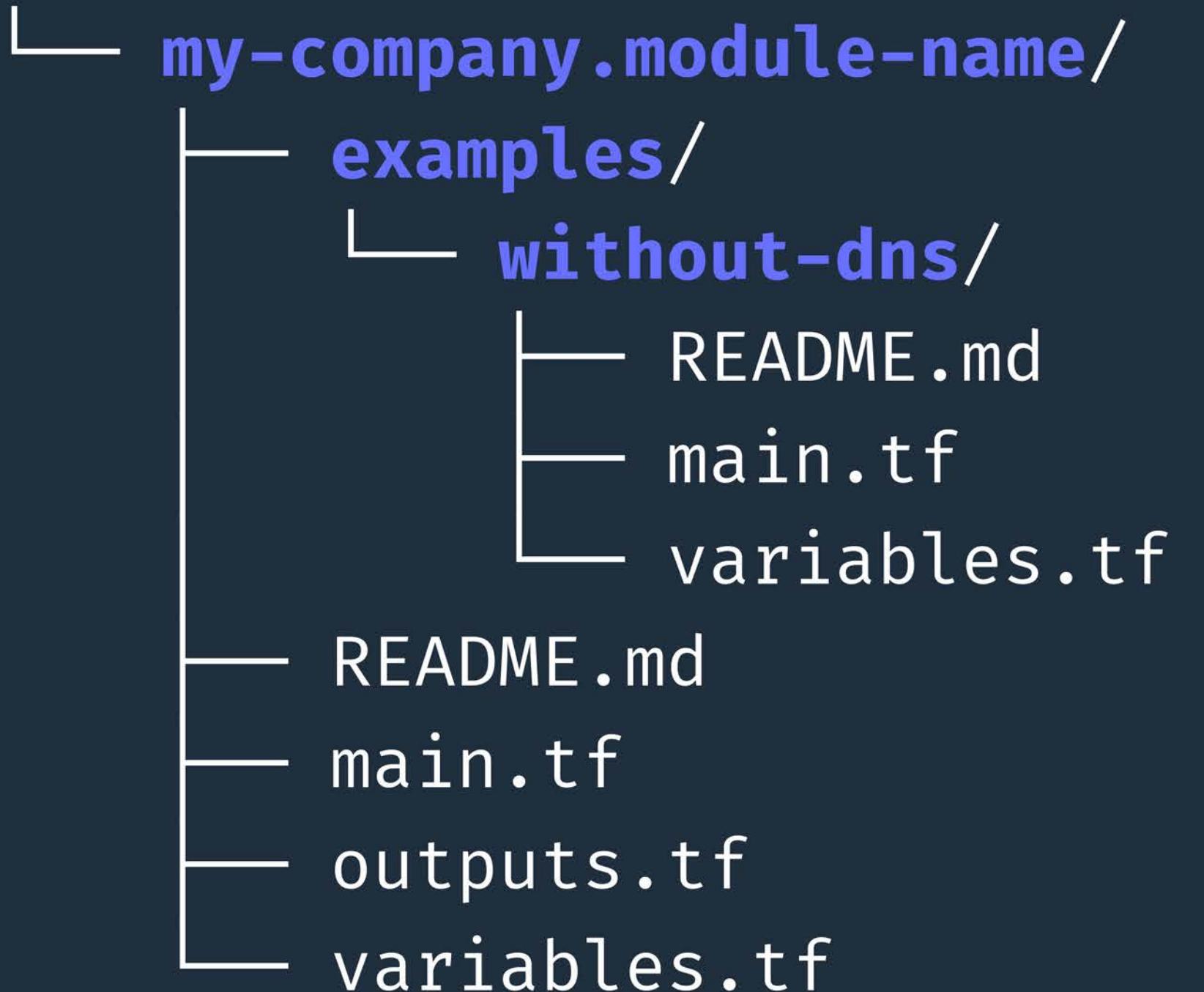


# Module

code reuse

› tree modules

## modules

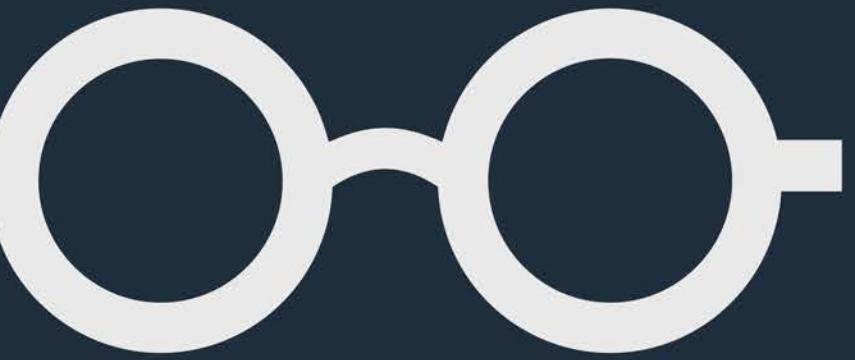


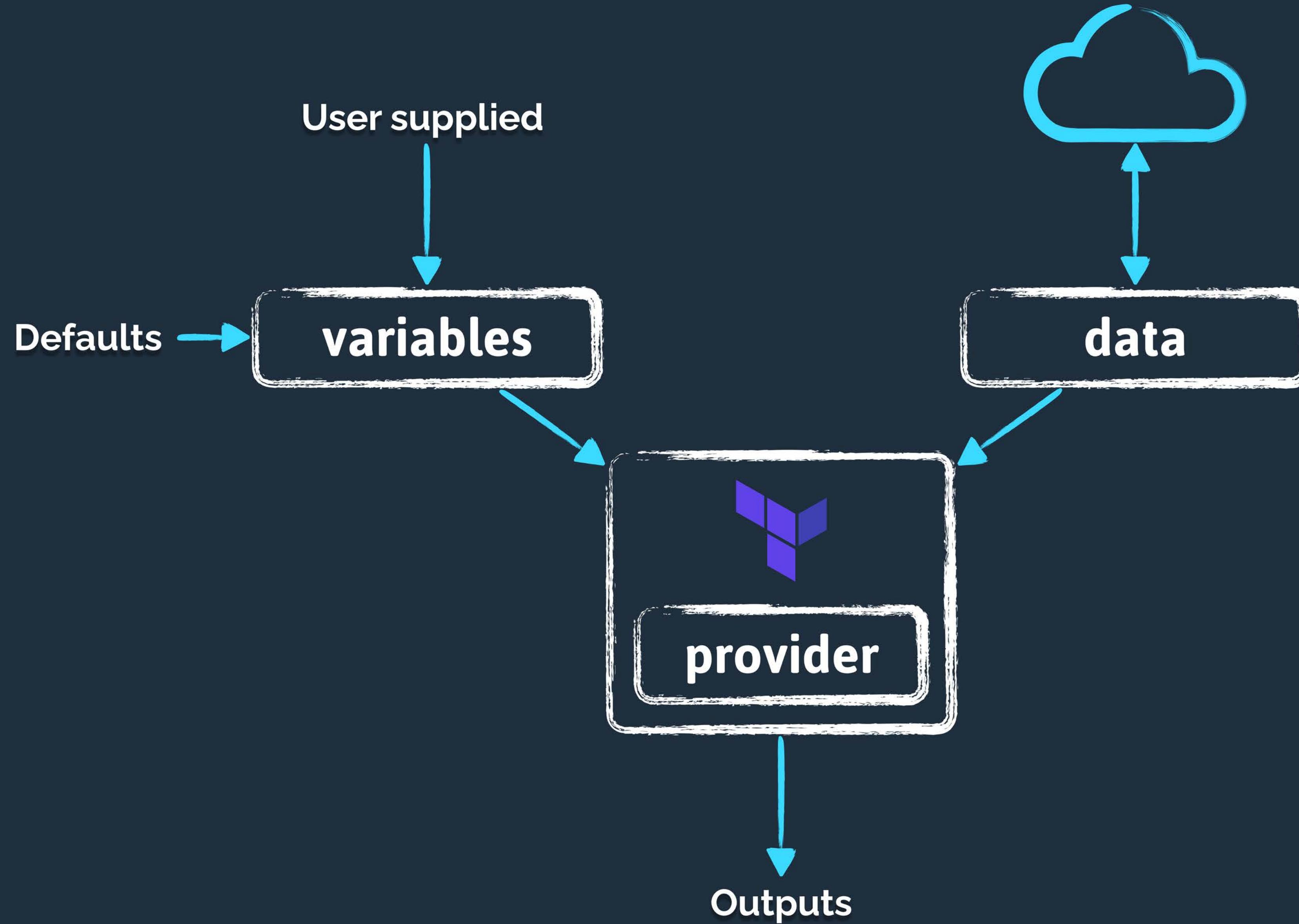
```
module "static_media_cdn" {
  source = ".. /my-company.module-name"

  name = "example"
  ip = var.ip
}
```

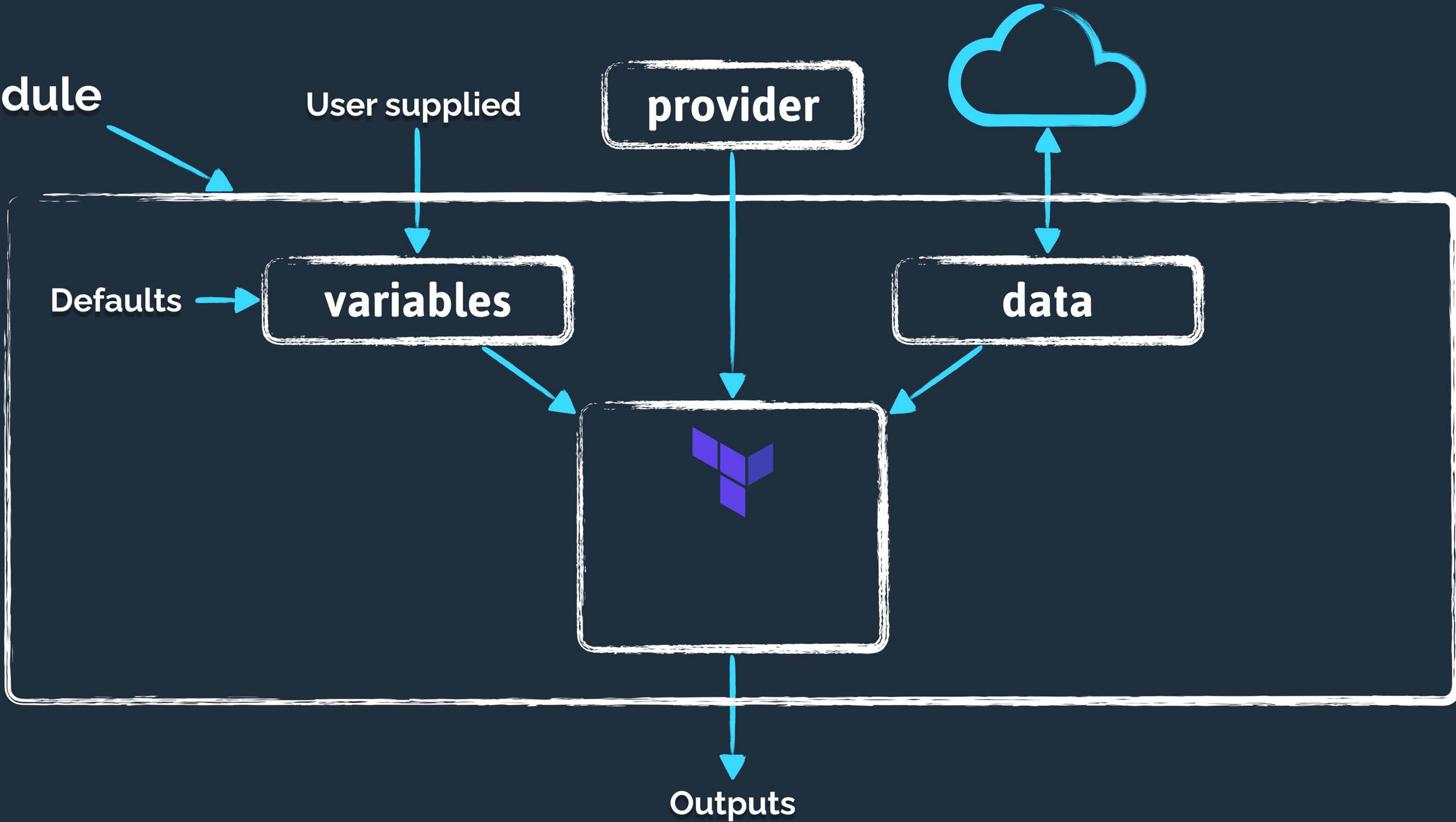
## Module Sources

Demo





**Module**



*Checking* 

terraform fmt  
terraform validate  
terraform plan  
tflint

} *static* ANALYSIS

terratest  
deploy  
validate  
destroy

} *integration*  
**TESTS**

use a sandbox env

} *integration*  
TESTS

#protip

Sentinel

# Orchestration



# Terragrunt

**DRY**

**LIFECYCLE**

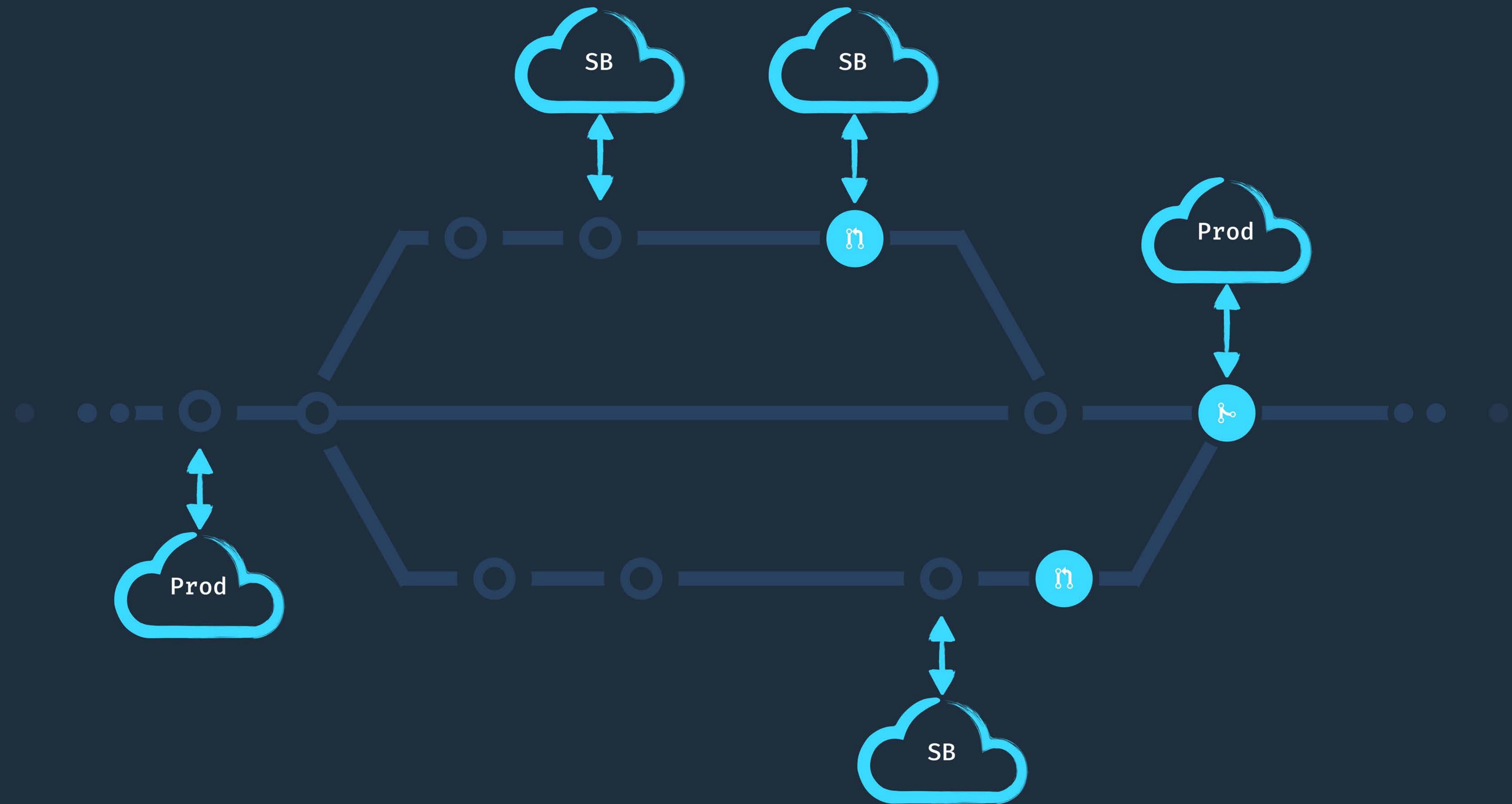
**EASY CLI**

**MULTI-AWS**

**RETRY**

# Ansible

# *Terraform* IN PRODUCTION



in a  
*nutshell*

Terraform is a **simple** tool

in a  
*nutshell*

Terraform makes provisioning **repeatable**

in a  
*nutshell*

Leverage the **ecosystem** around Terraform

*Thanks*

# Resources

[Terraform Registry](#)

# *Credits*

Theme

Cover Slide Background