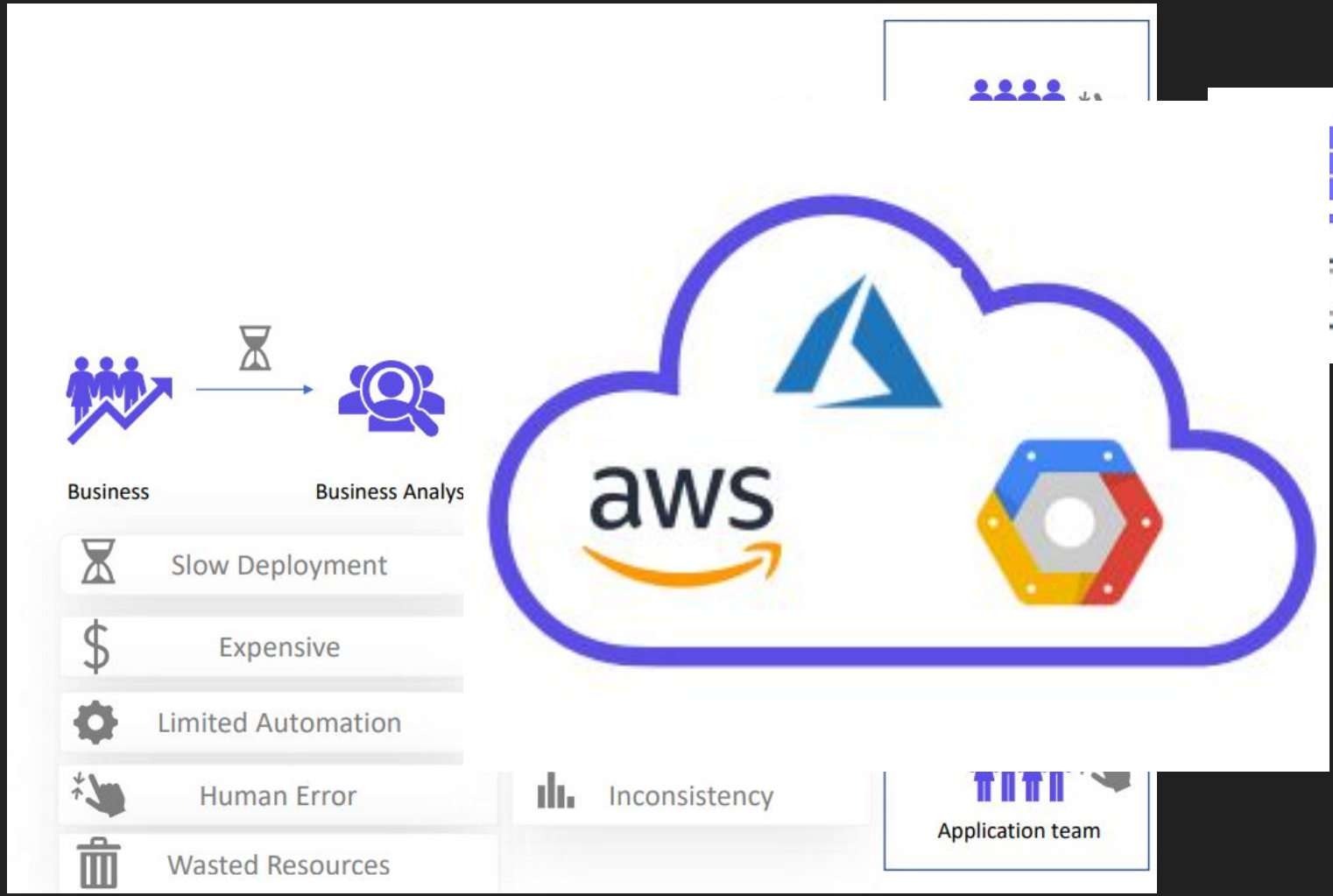
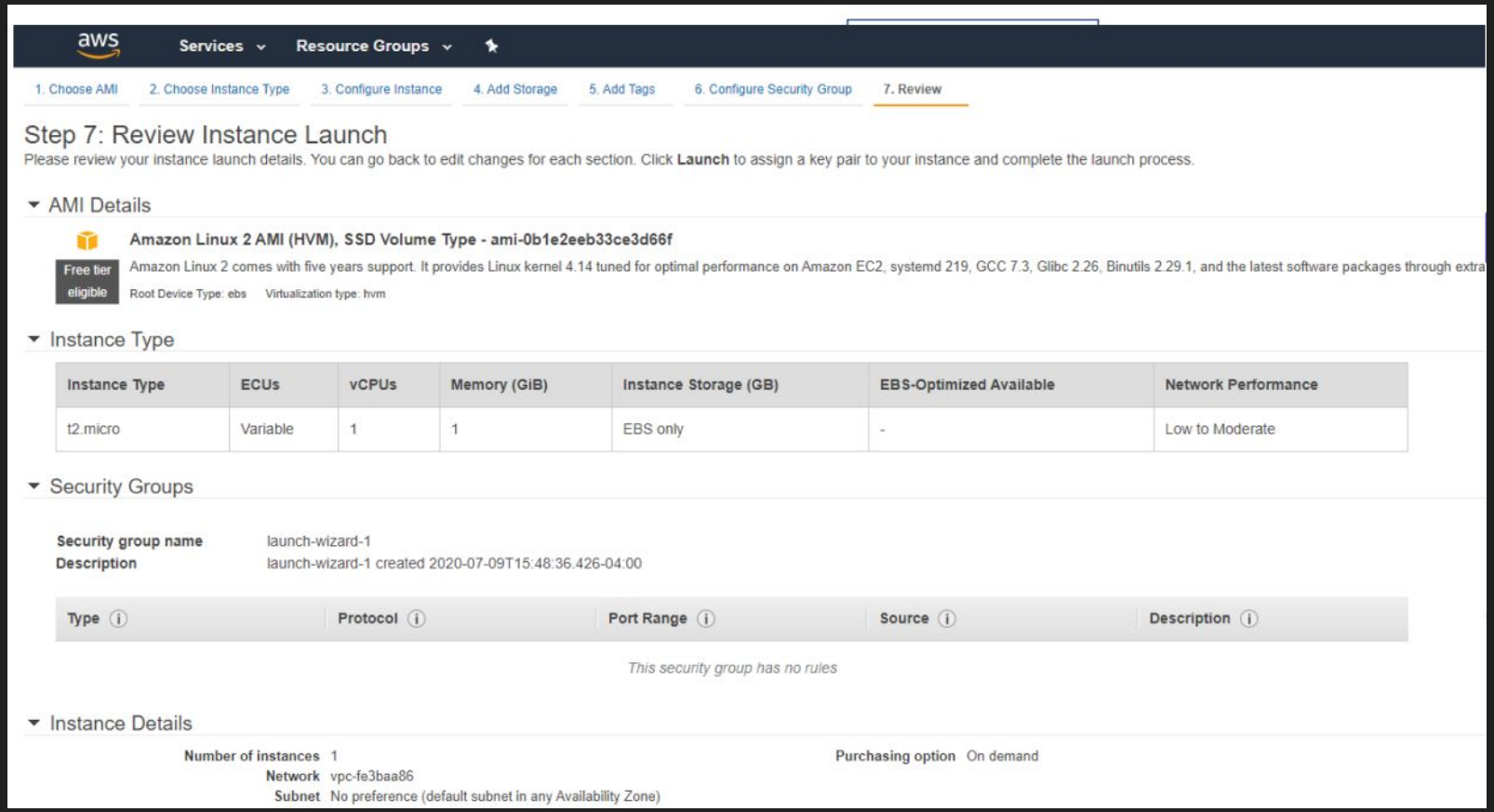
HashiCorp

Terraform


Business
Business Analys
Slow Deployment
Expensive
Limited Automation
Human Error
Wasted Resources
aws
Inconsistency
Application team


aws
Services
Resource Groups
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 7: Review Instance Launch
Please review your instance launch details. You can go back to edit changes for each section. Click Launch to assign a key pair to your instance and complete the launch process.
AMI Details
Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0b1e2eeb33ce3d66f
Free tier eligible
Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extra
Root Device Type: ebs
Virtualization type: hvm
Instance Type
Instance Type
ECUs
vCPUs
Memory (GiB)
Instance Storage (GB)
EBS-Optimized Available
Network Performance
t2.micro
Variable
1
1
EBS only
-
Low to Moderate
Security Groups
Security group name
launch-wizard-1
Description
launch-wizard-1 created 2020-07-09T15:48:36.426-04:00
Type
Protocol
Port Range
Source
Description
This security group has no rules
Instance Details
Number of instances
1
Network
vpc-fe3baa86
Subnet
No preference (default subnet in any Availability Zone)
Purchasing option
On demand

Shell

Python

Ruby

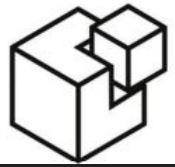
Powershell

ec2.sh

```
#!/bin/bash

IP_ADDRESS="10.2.2.1"

EC2_INSTANCE=$(ec2-run-instances --instance-type
t2.micro ami-0edab43b6fa892279)

INSTANCE=$(echo ${EC2_INSTANCE} | sed 's/*INSTANCE //'
| sed 's/ .*//')

# Wait for instance to be ready
while ! ec2-describe-instances $INSTANCE | grep -q
"running"
do
  echo Waiting for $INSTANCE is to be ready...
done

# Check if instance is not provisioned and exit
if [ ! $(ec2-describe-instances $INSTANCE | grep -q
"running") ]; then
  echo Instance $INSTANCE is stopped.
  exit
fi

ec2-associate-address $IP_ADDRESS -i $INSTANCE

echo Instance $INSTANCE was created successfully!!!
```

Infrastructure as Code

Infrastructure as Code



Types of IAC Tools

Configuration Management

Server Templating

Provisioning Tools

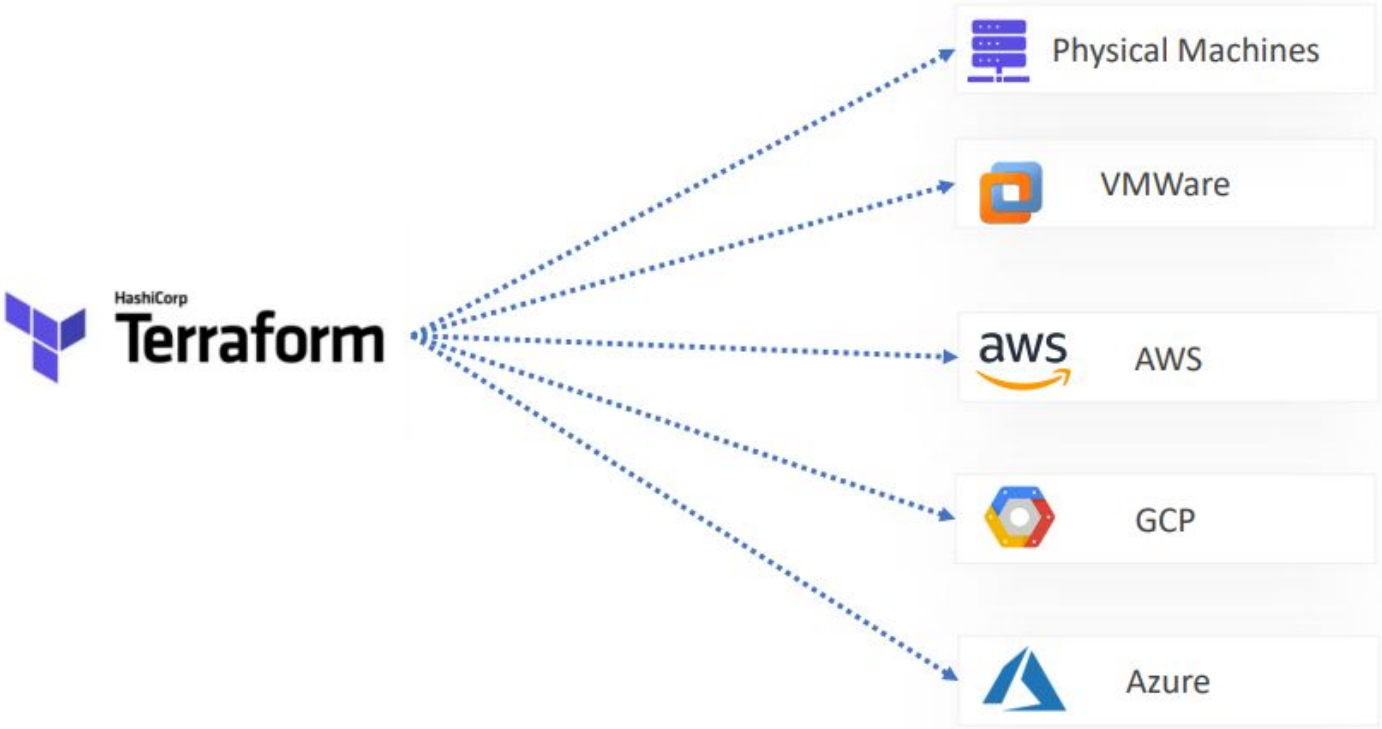
ec2.sh

```
#!/bin/bash

IP_ADDRESS="10.2.2.1"

EC2_INSTANCE=$(ec2-run-instances --instance-type
t2.micro ami-0edab43b6fa892279)

INSTANCE=$(echo ${EC2_INSTANCE} | sed 's/*INSTANCE //'
| sed 's/ .*//')

# Wait for instance to be ready
while ! ec2-describe-instances $INSTANCE | grep -q
"running"
do
  echo Waiting for $INSTANCE is to be ready...
done

# Check if instance is not provisioned and exit
if [ ! $(ec2-describe-instances $INSTANCE | grep -q
"running") ]; then
  echo Instance $INSTANCE is stopped.
  exit
fi

ec2-associate-address $IP_ADDRESS -i $INSTANCE

echo Instance $INSTANCE was created successfully!!!
```

main.tf

```
resource "aws_instance" "webserver" {
  ami           = "ami-0edab43b6fa892279"
  instance_type = "t2.micro"
}
```

Why Terraform


HashiCorp
Terraform
Physical Machines
VMWare
aws
AWS
GCP
Azure

Physical Machines	BigIP	DataDog	InfluxDB
VMWare	CloudFlare	Grafana	MongoDB
AWS	DNS	Auth0	MySQL
GCP	Palo Alto	Wavefront	PostgreSQL
Azure	Infoblox	Sumo Logic	VCS

🌟 What Terraform Does

✅ Create
Build new infrastructure — servers, networks, databases, and more.

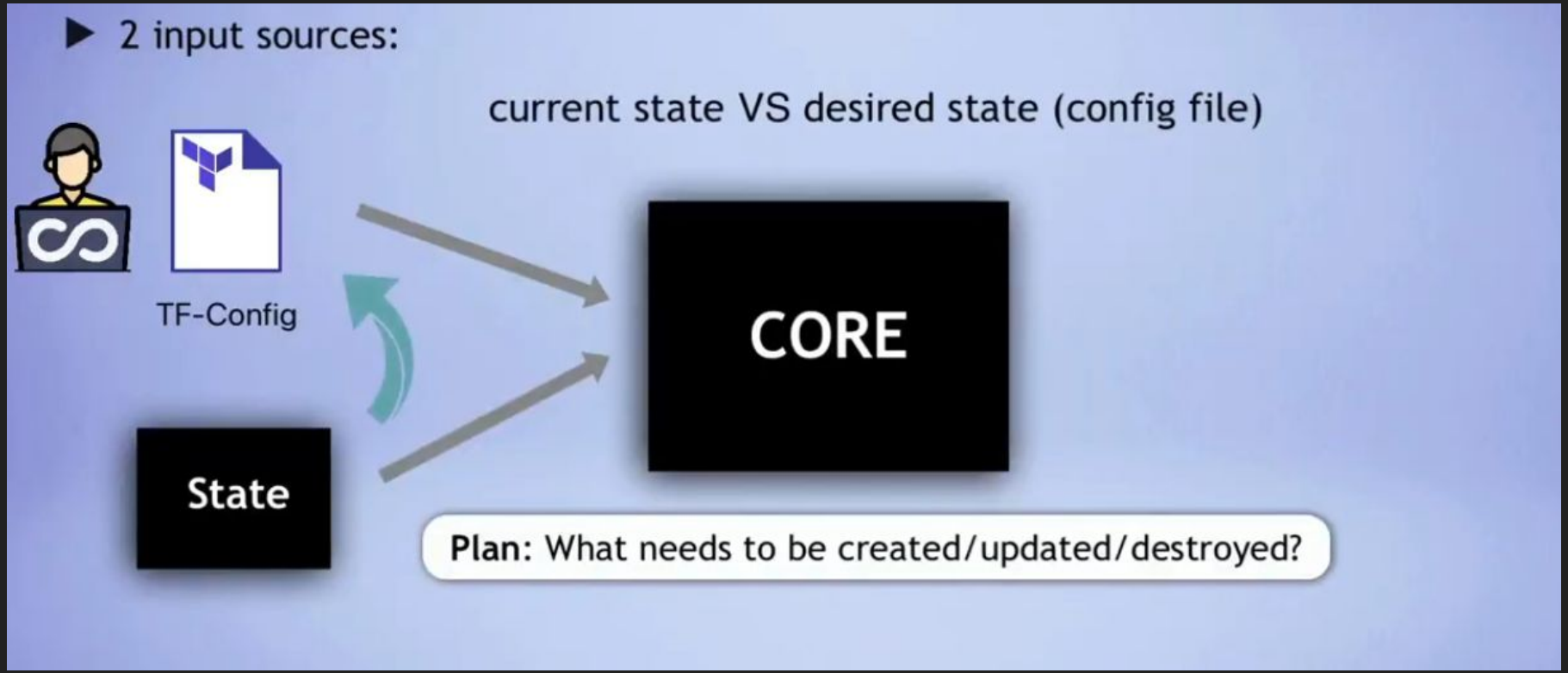
✅ Update
Modify existing infrastructure safely with automated change management.

✅ Replicate
Reuse code to deploy identical infrastructure across different environments (dev, test, prod).

✅ Delete
Cleanly destroy resources when they're no longer needed.

✅ Manage
Continuously control and track your infrastructure across cloud and on-prem systems.

How does Terraform work ?



desired state

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
}

# Create a VPC
resource "aws_vpc" "example" {
  cidr_block = "10.0.0.0/16"
}
```

refresh query infrastructure provider to get current state

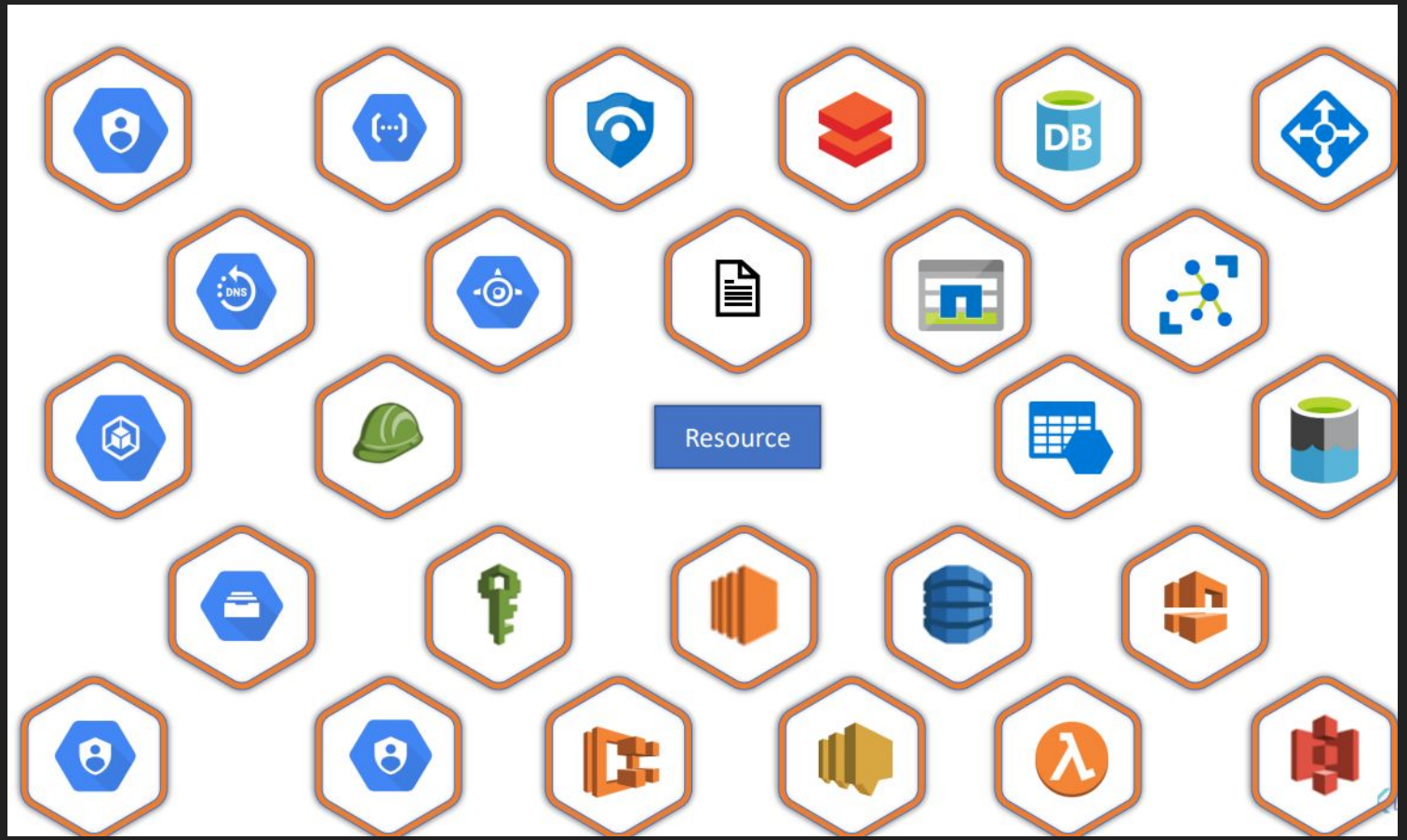
State

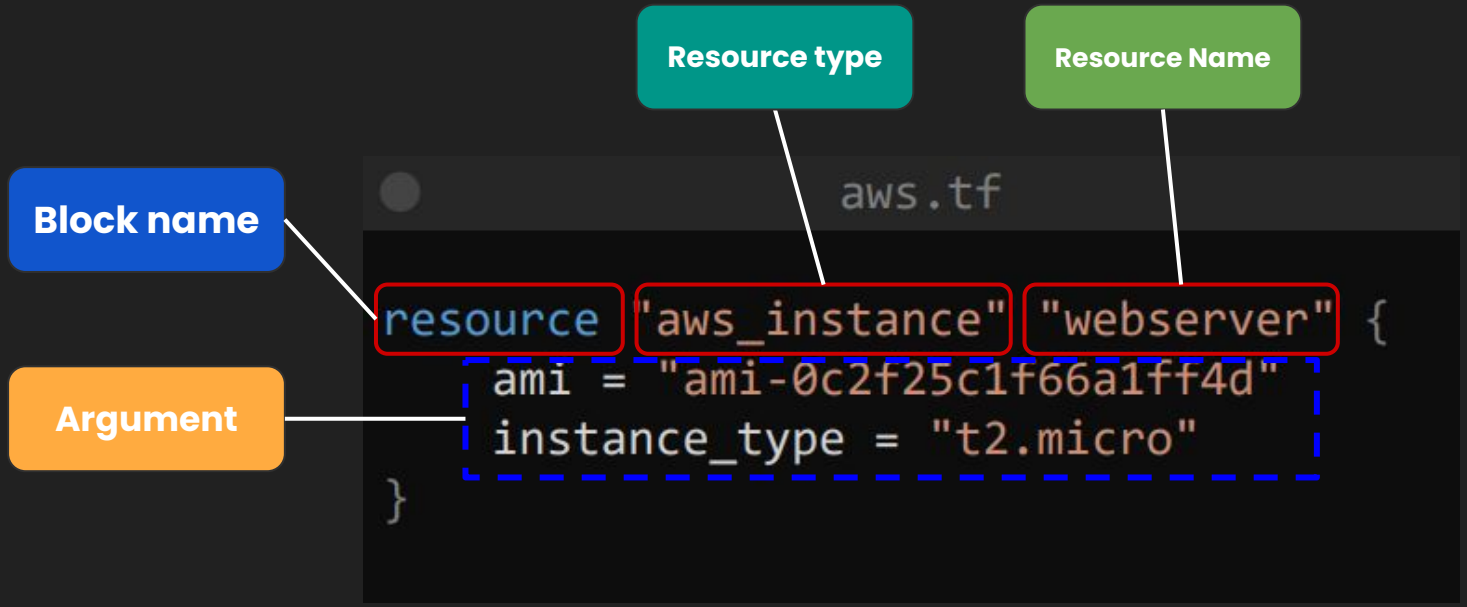
plan create an execution plan

apply execute the plan

destroy destroy the resources/infrastructure

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
}

# Create a VPC
resource "aws_vpc" "example" {
  cidr_block = "10.0.0.0/16"
}
```


DB
DNS
Resource


Resource type
Resource Name
Block name
Argument
aws.tf
resource "aws_instance" "webserver" {
ami = "ami-0c2f25c1f66a1ff4d"
instance_type = "t2.micro"
}

Terraform Installation

Working with Provider

Provision AWS infrastructure:

Demo Overview

1) Create custom **VPC**

2) Create custom **Subnet**

3) Create **Route Table & Internet Gateway**

4) Provision **EC2 Instance**

5) Deploy nginx Docker container

6) Create **Security Group** (Firewall)