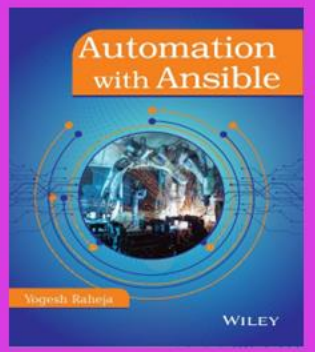
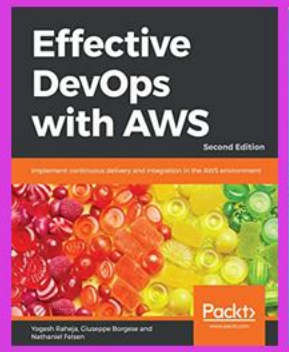
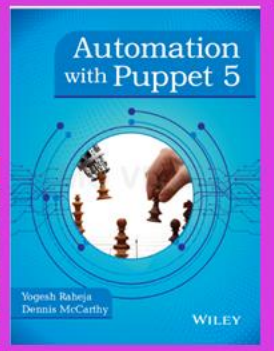
Infrastructure Automation with OpenTofu

By Thinknyx Technologies LLP

Yogesh Raheja



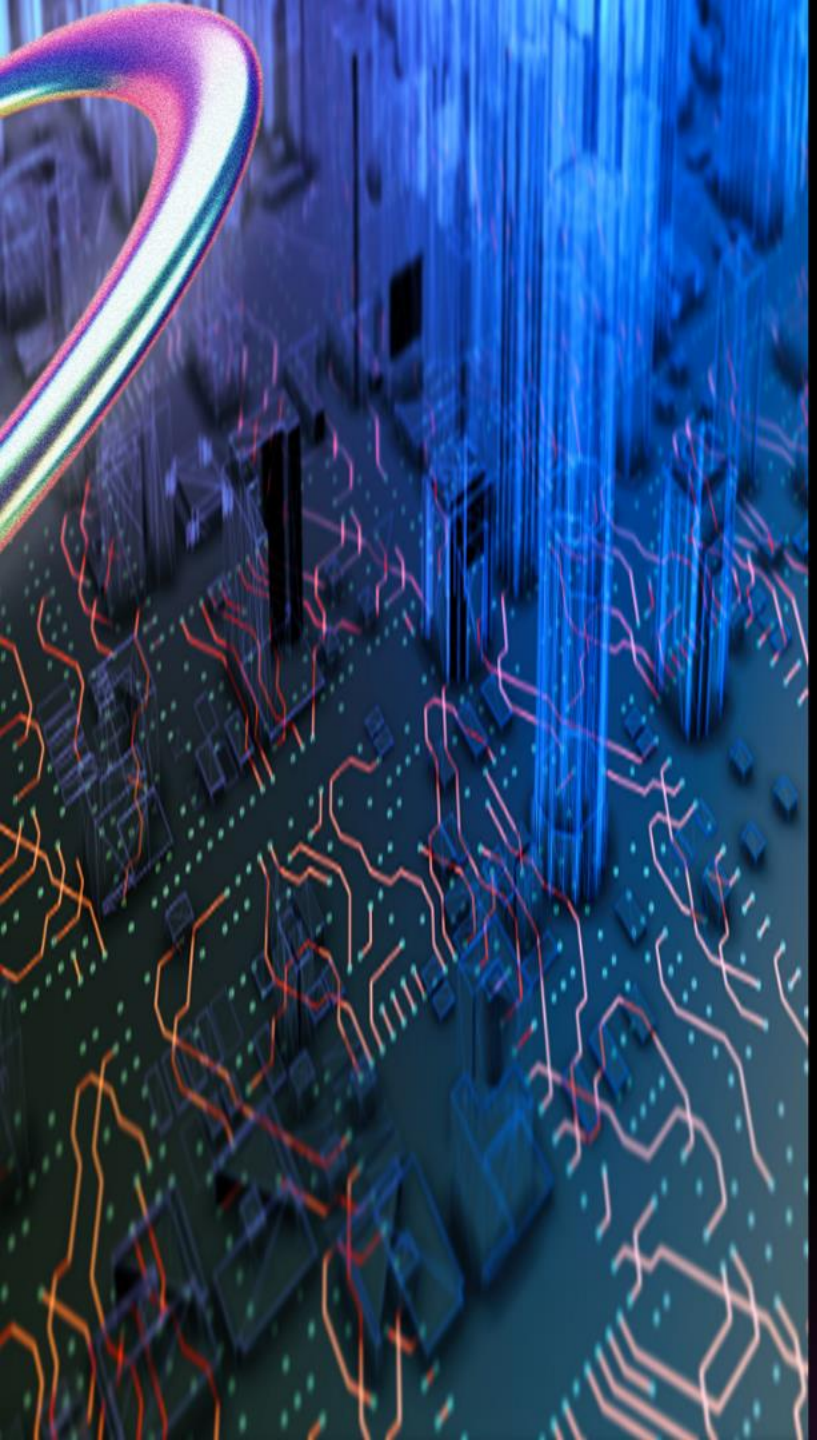




Puppet for the Absolute Beginners - Hands-on - DevOps

SaltStack for the Absolute Beginners - Practical DevOps

Deepthi Narayan



Course Workflow

- Lectures
- Demos
- Coding
- Assignments

Objective

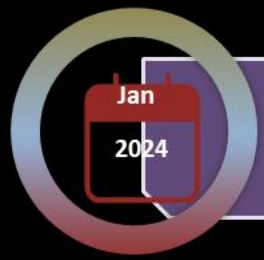
- Introduction
- Environment Setup
- OpenTofu Workflow
- Provisioners and Data Sources
- Other Advance Concepts

- OpenTofu vs Terraform
- Providers and Resources
- Variables and Output Block
- Modules, Backends and State files
- Terraform code migrated to OpenTofu

OpenTofu | Introduction

Introduction to OpenTofu

Infrastructure as code (IaC)



January 2024, Linux Foundation

Open-source

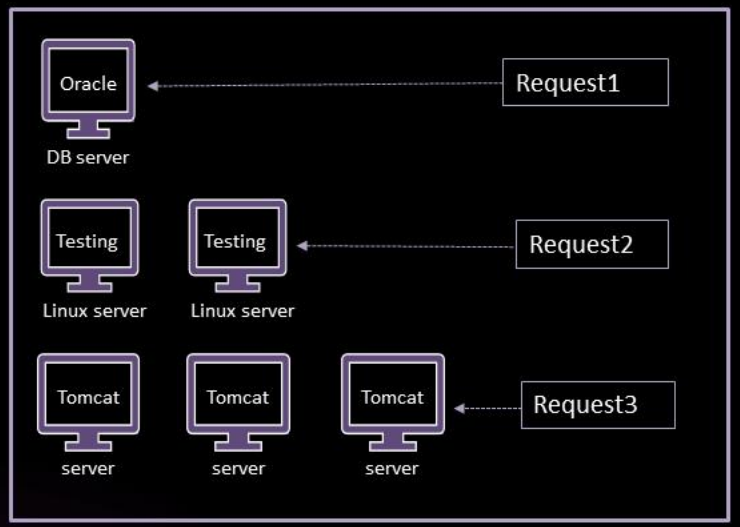
Go (Golang)

Infrastructure as Code

ABC Core

Developers

Operations



Manual Task

✓ Business landscape evolved

✓ No room for delays

Infrastructure as Code


```
resource "aws_instance" "example" {
  ami           = "ami-0c7217cdde317cfec"
  instance_type = "t2.micro"
}
```

IaC

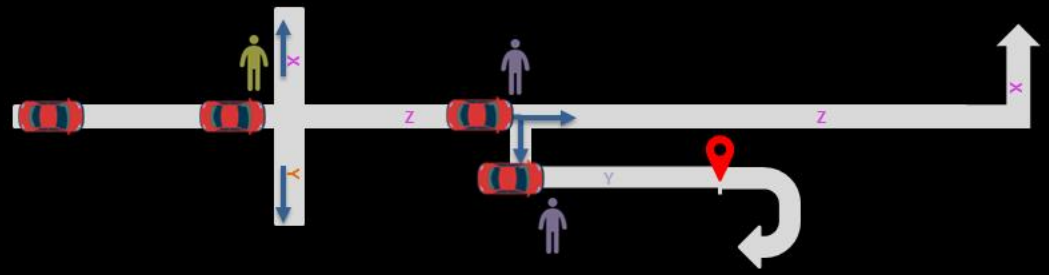
- Simplicity
- Efficiency

✓ Declarative Approach = What ✓

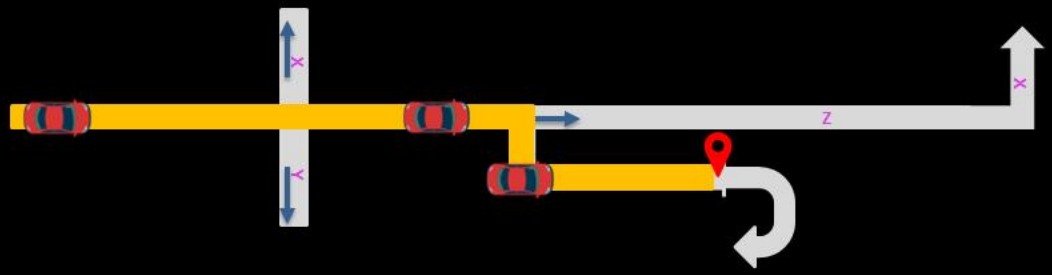
How ✗

Imperative vs Declarative approach

Imperative



Declarative



Destination

Imperative vs Declarative approach

Imperative

```
#!/bin/bash

# Create EC2 instance
INSTANCE_ID=$(aws ec2 run-instances \
    --image-id ami-0c7217cdde317cfec \
    --instance-type t2.micro \
    --output json \
    --query 'Instances[0].InstanceId' \
    | tr -d '"')

echo "Instance $INSTANCE_ID creation started."

# Wait until the instance is running
echo "Waiting for instance $INSTANCE_ID to be in running state..."
aws ec2 wait instance-running --instance-ids $INSTANCE_ID
echo "Instance $INSTANCE_ID is now running."
```

Declarative

```
resource "aws_instance" "example" {
  ami           = "ami-0c7217cdde317cfec"
  instance_type = "t2.micro"
}
```

OpenTofu vs Terraform

- Is OpenTofu different from Terraform?
- Why OpenTofu over Terraform?
- OpenTofu Goals
- Will OpenTofu work with my existing state file?
- Is OpenTofu production grade?

VS

Why OpenTofu?

- ❑ Opensource
- ❑ Community Driven
- ❑ Stability
- ❑ Easy Migration
- ❑ Simple Configuration Language
- ❑ Impartial

OpenTofu Use cases

IaC

Cloud Agnostic

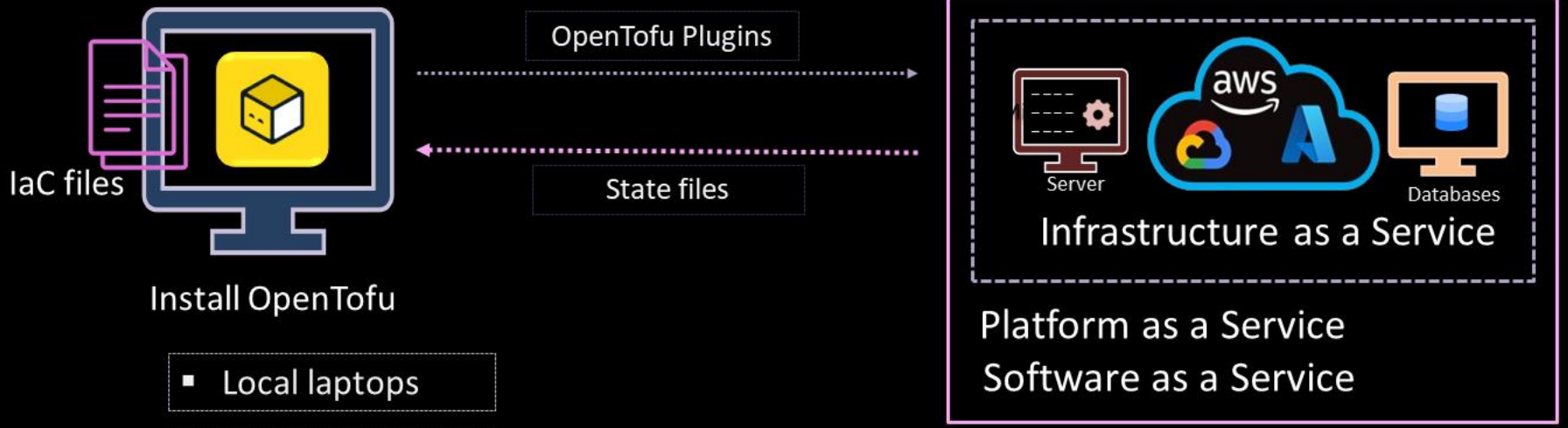
CI/CD

Single bundled Solution

Policy as a code

Kubernetes deployment

OpenTofu Workflow

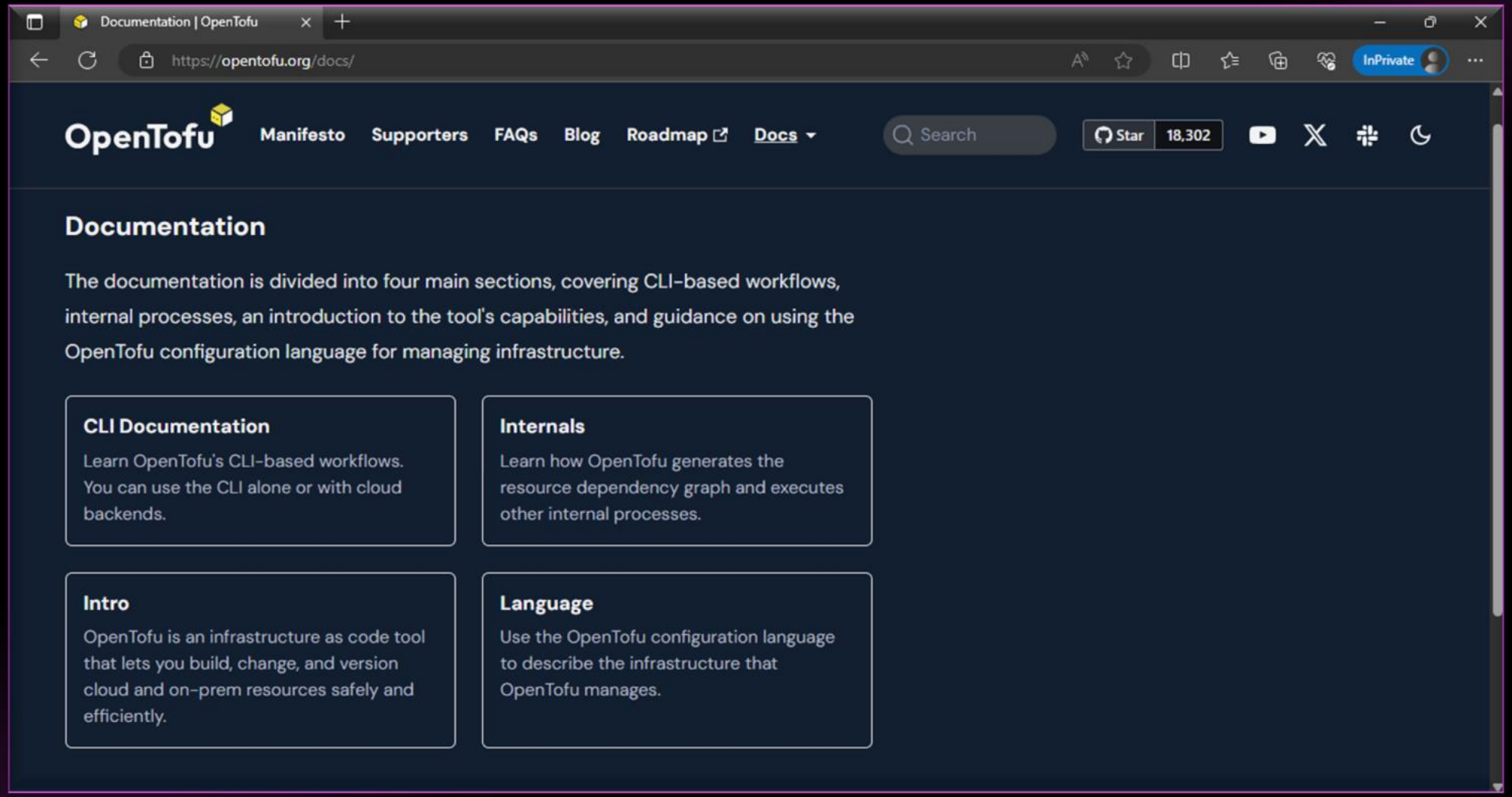


Install OpenTofu

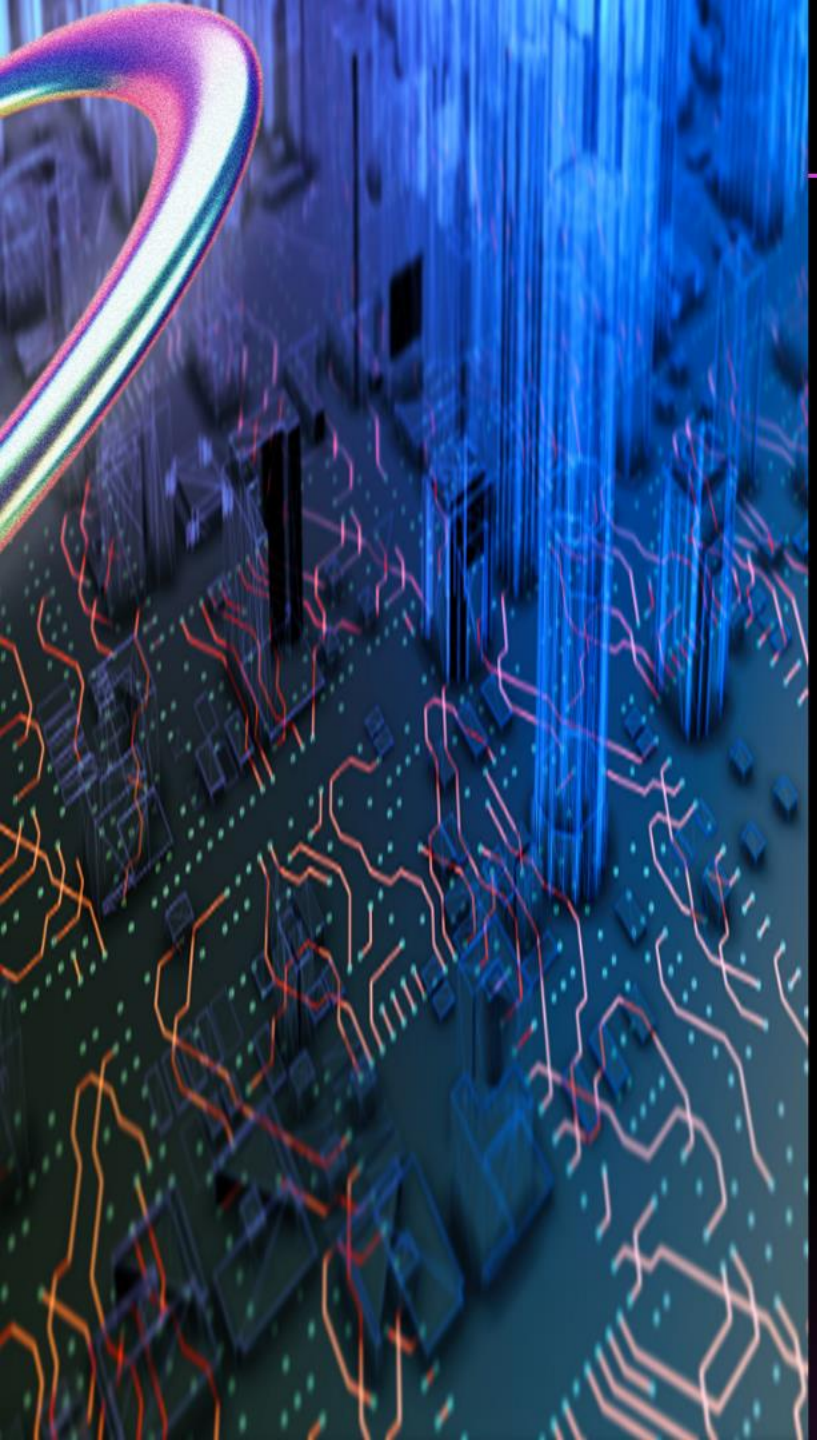
- Local laptops
- Desktops
- Virtual machines

- Write
- Plan
- Apply

Documentation Walkthrough



Demo | OpenTofu Documentation Walkthrough

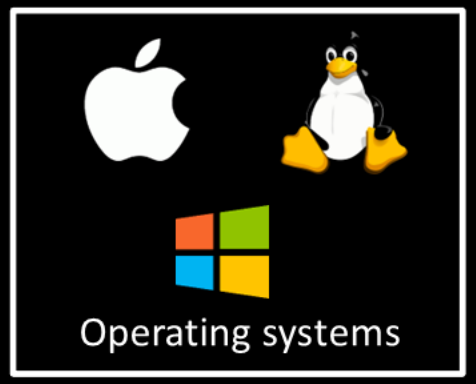


Summary

- ✓ Basics of OpenTofu
- ✓ Use cases
- ✓ Workflow
- ✓ Documentation
- ✓ Install and setup OpenTofu

OpenTofu | Getting Started

Setting up OpenTofu



- ✓ Local laptops
- ✓ Desktops
- ✓ Virtual machines

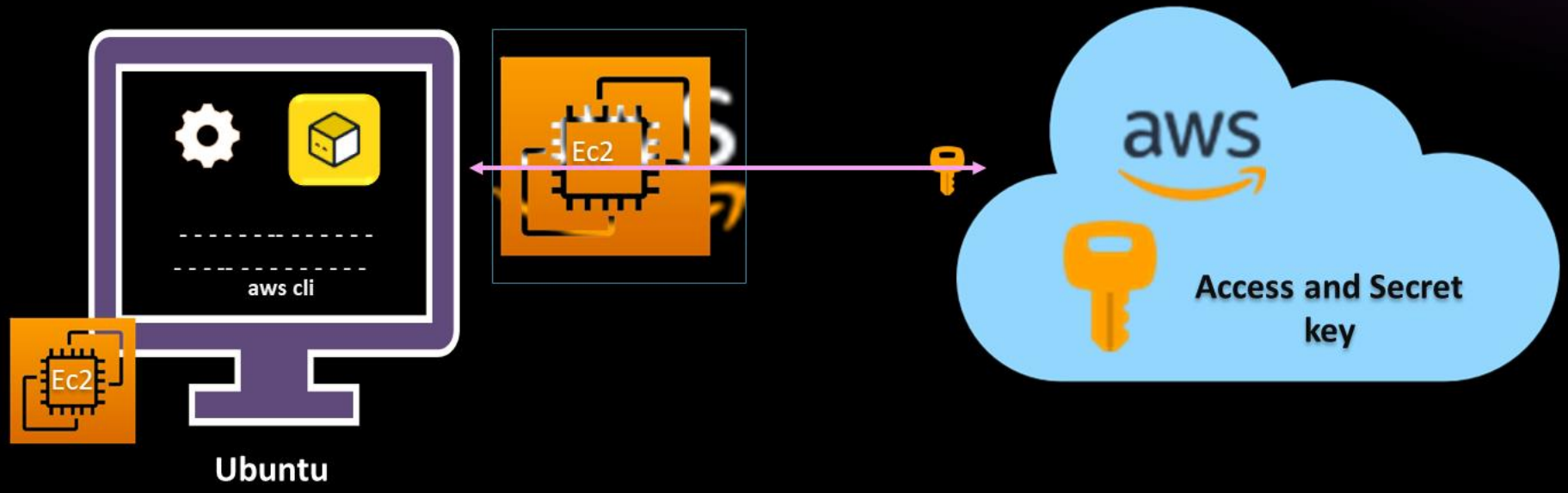
Any Platform

CLI

configure

Other Platforms

OpenTofu with AWS



Demo | AWS Instance Creation

Demo | Complete OpenTofu Setup for AWS

Demo | OpenTofu Setup for macOS

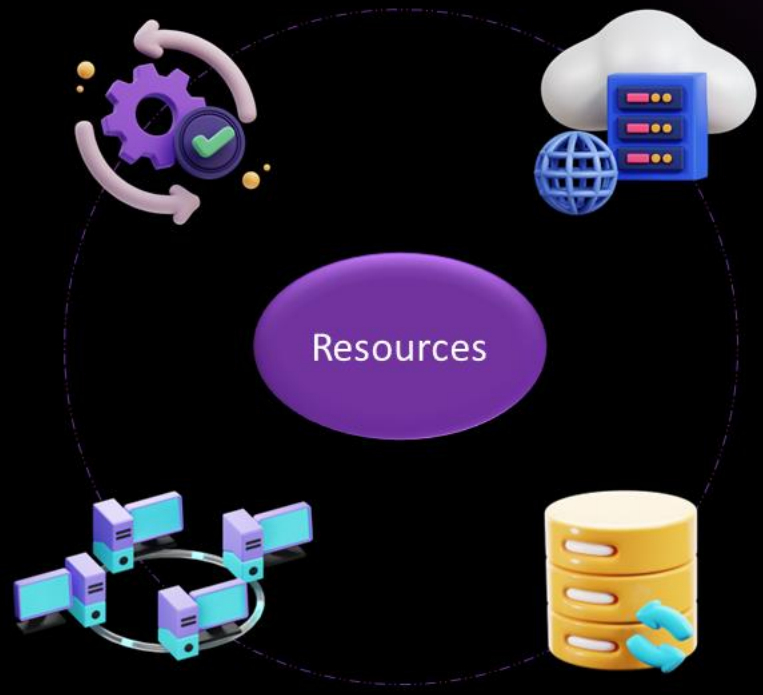
Summary

- ✓ OpenTofu Setup on various platforms
- ✓ OpenTofu concepts and workflow remain same for all platforms

OpenTofu | Configuration Language

OpenTofu Configuration Language

- ✓ Native OpenTofu Language Syntax
- ✓ JSON Syntax



OpenTofu Configuration Language

- ✓ Blocks serve as Foundational Units
- ✓ Providers, Resources, and Outputs

```
<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>" {
  # Block body
  <ARGUMENT NAME1> = <ARGUMENT VALUE1> # Argument1
  <ARGUMENT NAME2> = <ARGUMENT VALUE2> # Argument2
  <ARGUMENT NAME3> = <ARGUMENT VALUE3> # Argument3
}
```

```
resource "aws_instance" "thinknyxserver" {
  ami           = "ami-030ff268bd7b4e8b5"
  instance_type = "t2.micro"
}
```

```
variable "thinknyxserverIP" {

}
```

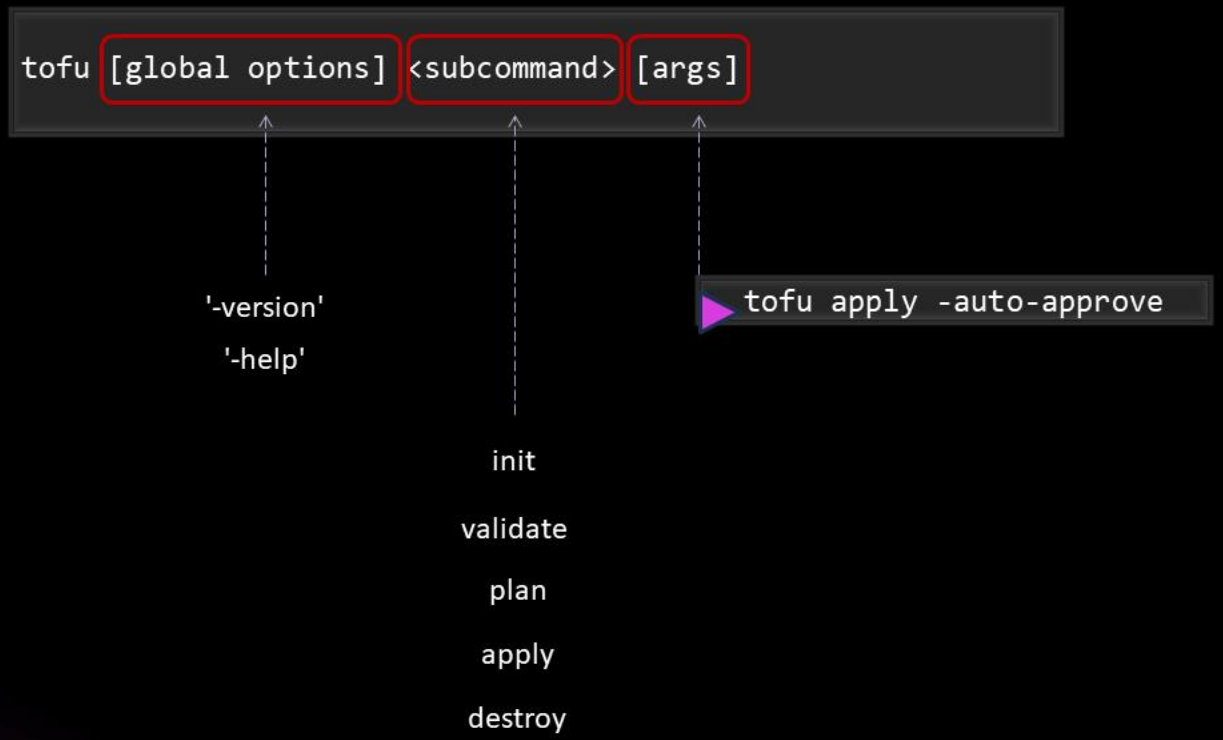
OpenTofu Configuration Language

- ✓ Blocks serve as Foundational Units
- ✓ Providers, Resources, and Outputs

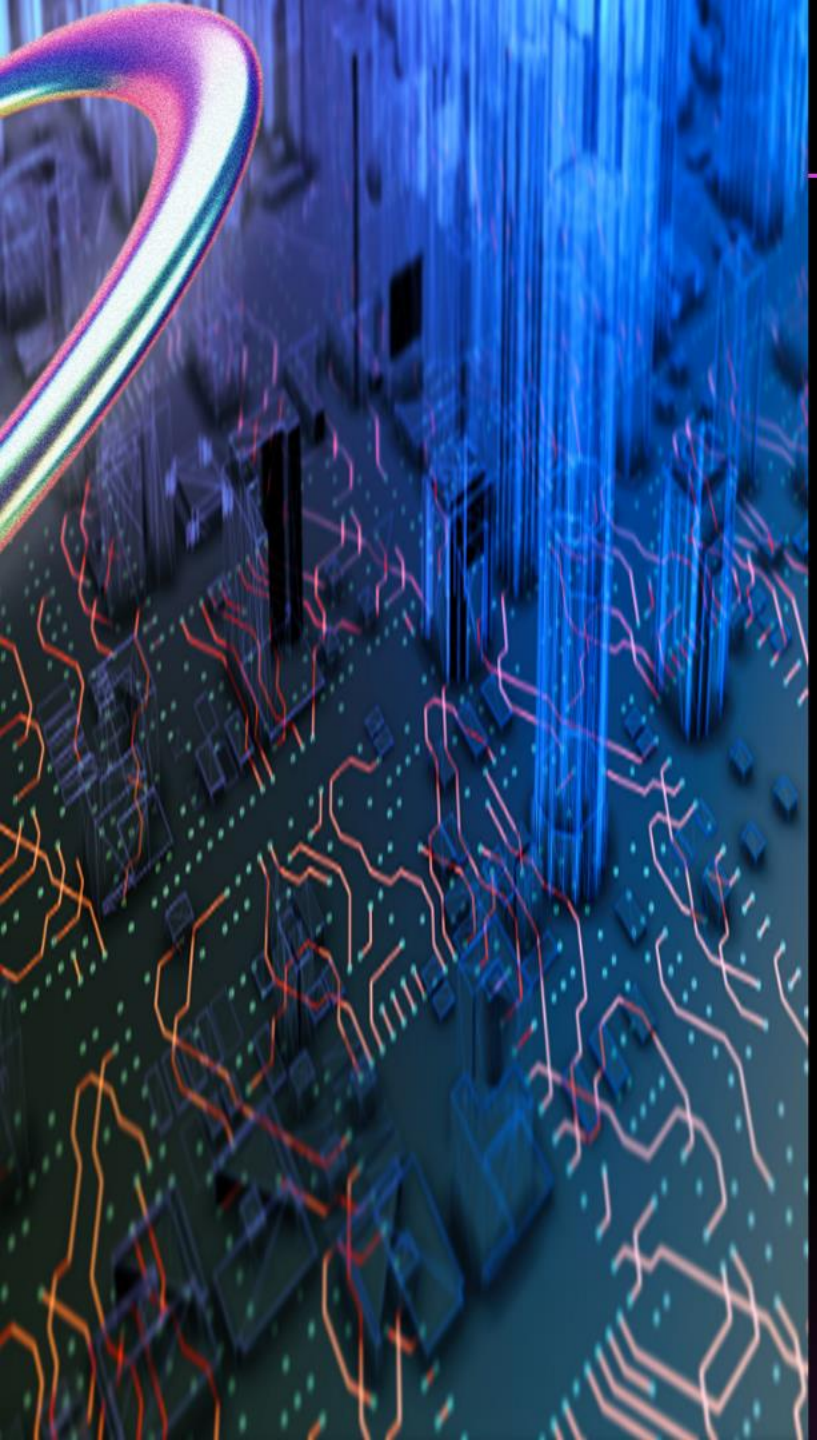
Demo | OpenTofu Configuration Language Documentation

OpenTofu | CLI Overview

OpenTofu CLI



Demo | OpenTofu CLI

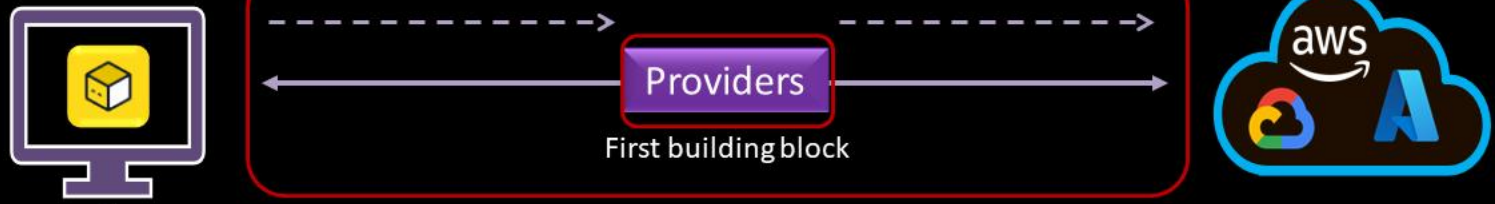


Summary

- ✓ OpenTofu Configuration Language Syntax
- ✓ Sample Files
- ✓ OpenTofu CLI

OpenTofu | Providers

OpenTofu Providers



Block Type

Block Label

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

Argument Name

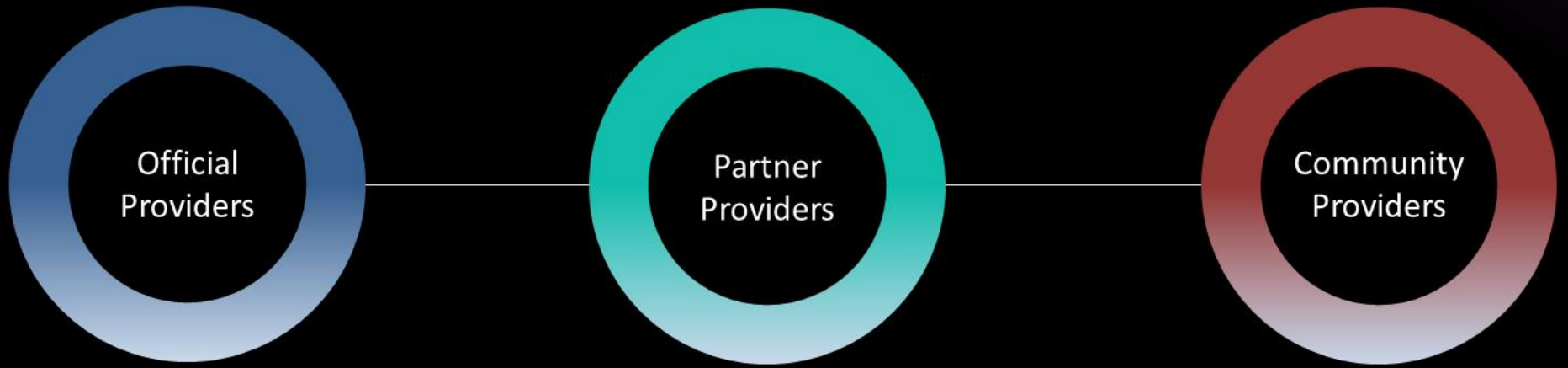
Argument Value

```
tofu init
```

OpenTofu Registry

https://opentofu.org/registry/

Types of Providers



Some providers on the Registry are developed and published by HashiCorp, some are published by platform maintainers, and some are published by users and volunteers.

https://opentofu.org/docs/language/providers/

Multiple Providers

- ✓ Multiple providers can be used simultaneously in OpenTofu

Provider Version

- AWS provider version - 5.37.0 (February 2024)
- Terraform Block
  - ✓ required_version
  - ✓ backend
  - ✓ required_providers

```
terraform {
  required_version = ">= 1.6.0"

  backend "s3" {
    bucket = "my-thinknyx-state-bucket"
    key    = "terraform.tfstate"
    region = "us-east-1"
  }

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 5.33.0"
    }
  }
}
```

Provider Version

- AWS provider version - 5.37.0 (February 2024)
- Terraform Block - First block of the code in OpenTofu

- ✓ required_version
- ✓ backend
- ✓ required_providers

```
terraform {
  required_version = ">= 1.6.0"

  backend "s3" {
    bucket = "my-thinknyx-state-bucket"
    key    = "terraform.tfstate"
    region = "us-east-1"
  }

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 5.33.0"
    }
  }
}
```

Provider Version

required_providers block

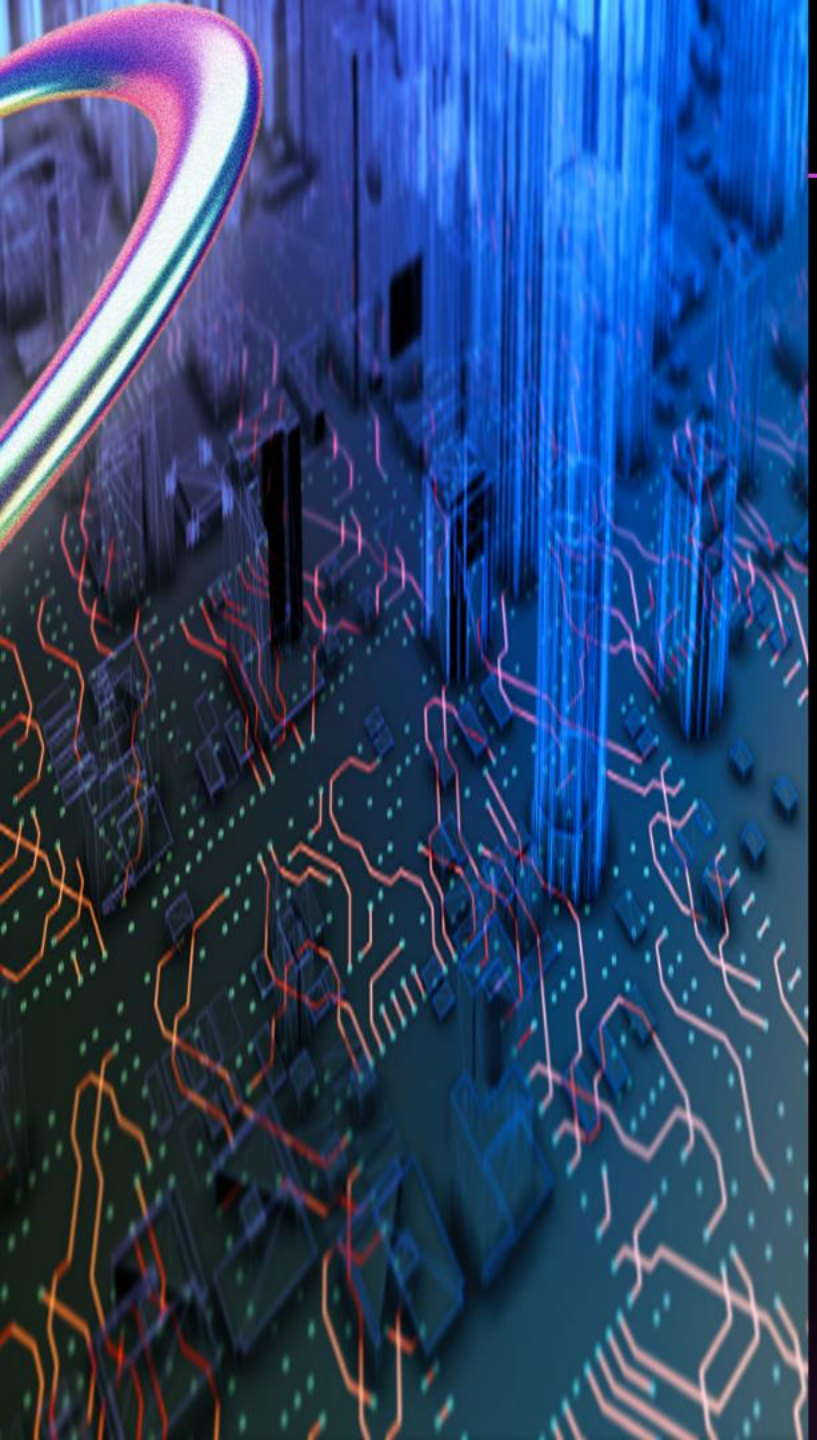
```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = ">= 5.33.0, < 5.36.0"
    }
  }
}

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

Operators	
equal to	=
lesser than	<
greater than	>
less than or equal to	<=
greater than or equal to	>=
minor release incremental	~>
not equal to	!=

Demo | Provider Documentation

Demo | First Provider File in OpenTofu

Demo | Multiple Providers



Summary

- ✓ OpenTofu Providers
- ✓ Basics know-how

OpenTofu | Workflow and Resources

OpenTofu Workflow

Initialize

`tofu init`

Format

`tofu fmt`

Validate

`tofu validate`

Plan

`tofu plan`

-out argument

Apply

`tofu apply`

- ✓ Idempotent
- ✓ Intelligent Dependency Resolver
- ✓ Parallel creation

Destroy

`tofu destroy`

OpenTofu Resources

Resource Type

Resource Name

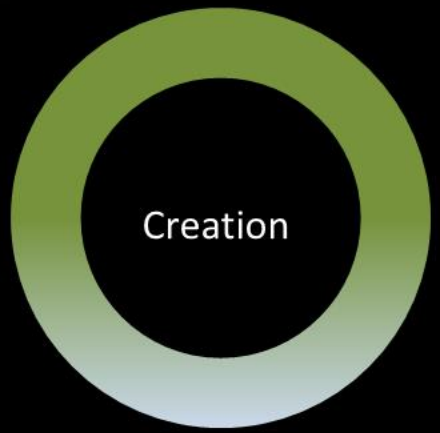
```
resource "aws_instance" "thinknyxserver" {
  ami = "ami-0c7217cdde317cfec"
  instance_type = "t2.micro"
}
```

Arguments

Demo | OpenTofu Resources Documentation

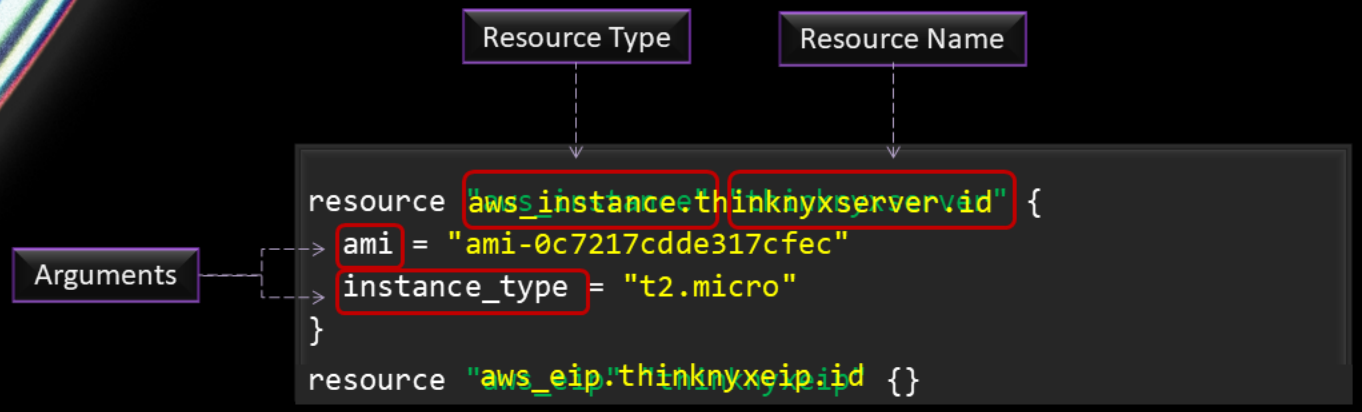
Demo | Workflow for instance creation on AWS

Resource Updates



Demo | Resource Updates In-place and Re-Create

Resource Reference




```
resource "aws_eip_association" "thinknyxeip_assoc" {
  instance_id   =
  allocation_id =
}
```

<RESOURCE TYPE>.<NAME>	"aws_instance.thinknyxserver"
<RESOURCE TYPE>.<NAME>.<ARGUMENT>	"aws_instance.thinknyxserver.instance_type"

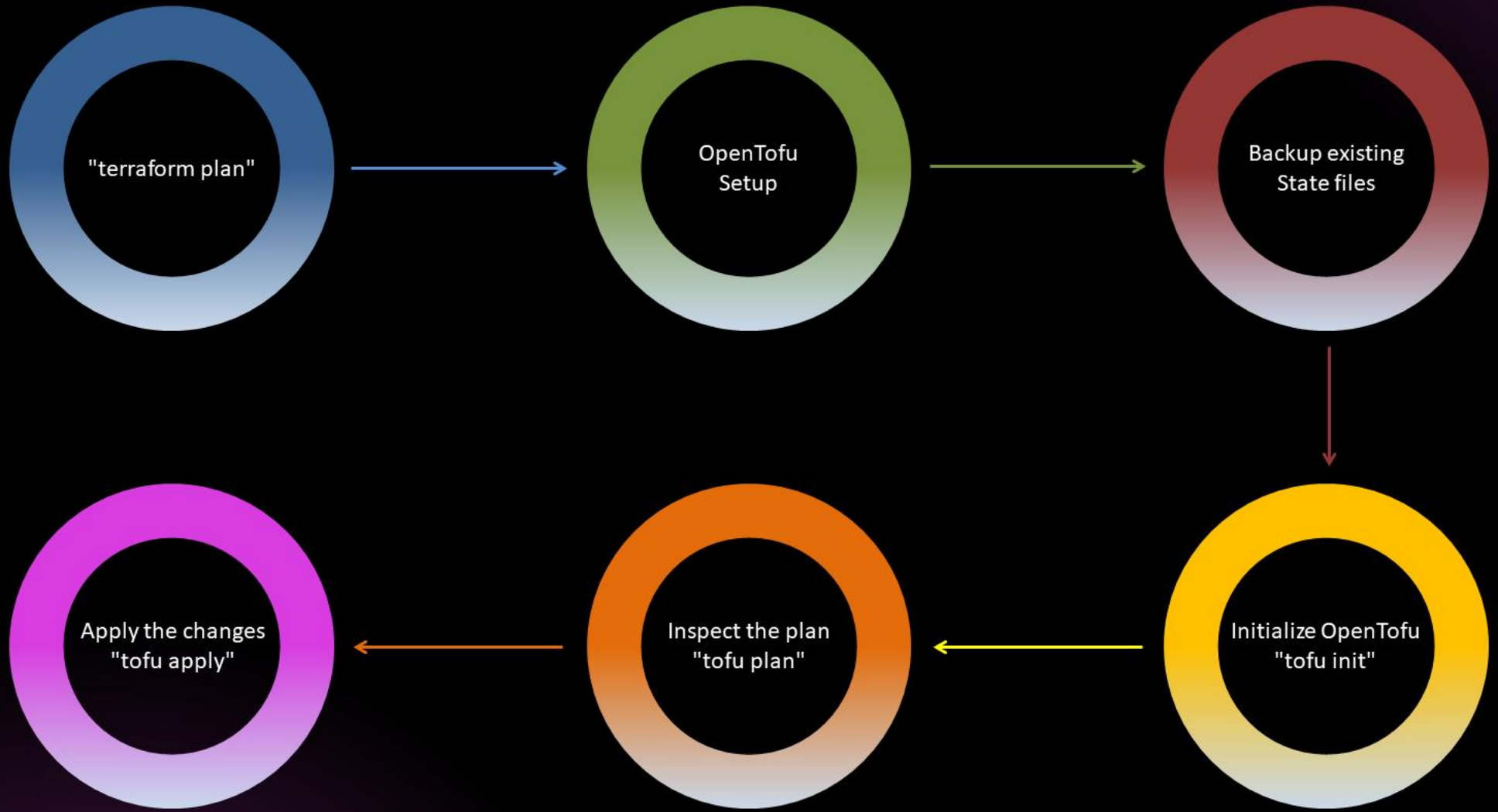
Demo | Resource Reference

Summary

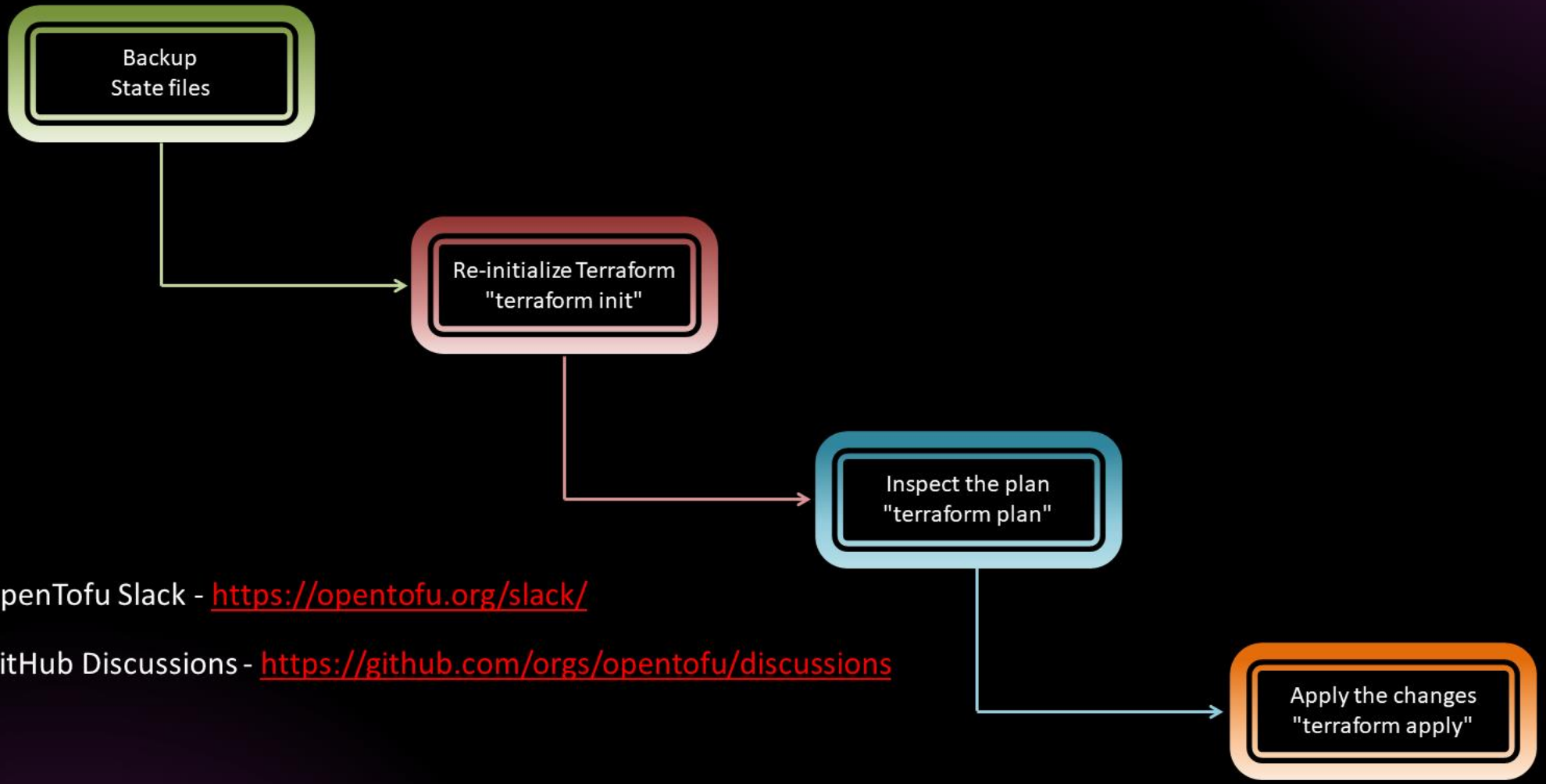
- ✓ OpenTofu Resources
- ✓ Detailed OpenTofu Workflow

OpenTofu | Migrating from Terraform to OpenTofu

Migration Process

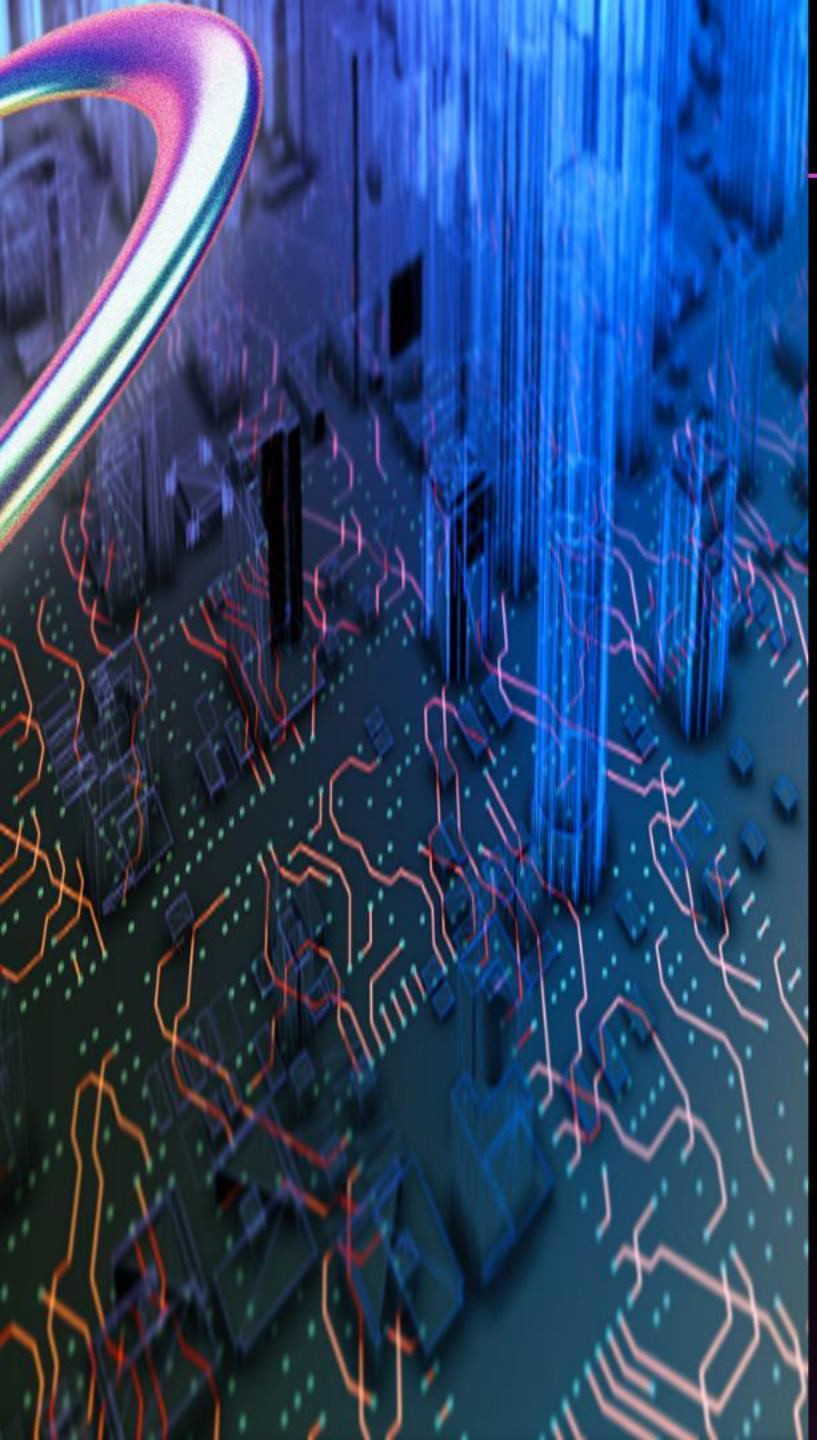


RollBack Process



- OpenTofu Slack - https://opentofu.org/slack/
- GitHub Discussions - https://github.com/orgs/opentofu/discussions

Demo | Terraform to OpenTofu Migration



Summary

- ✓ Migration from Terraform to OpenTofu
- ✓ Next - OpenTofu Variables

OpenTofu | Input Variables

OpenTofu files

Demo | OpenTofu files

Input Variables

main.tf

```
resource "aws_instance" "thinknyxserver" {
  ami = "ami-0c7217cdde317cfec"
  instance_type = "t2.micro"
  tags = {
    Name= "thinknyxserver"
  }
}
```

t2.smallb

Block Type | Block Label

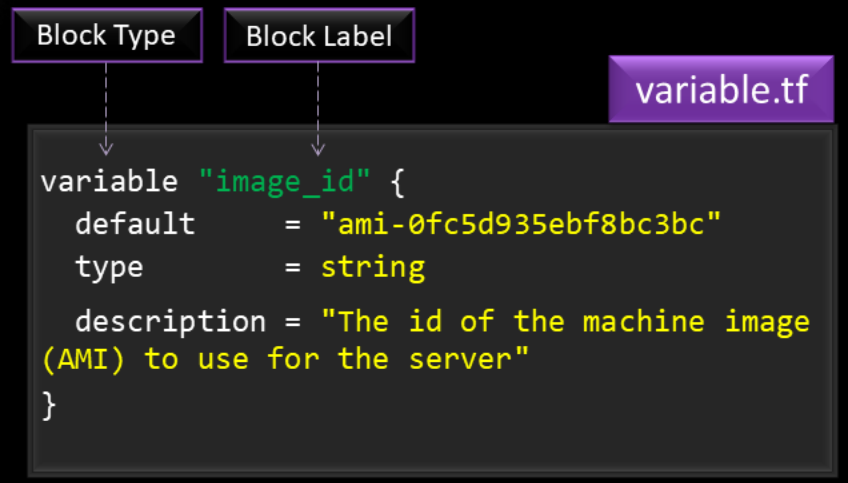
variable.tf

```
variable "image_id" {
  default     = "ami-0fc5d935ebf8bc3bc"
  type        = string
  description = "The id of the machine image
(AMI) to use for the server"
}
```

Example

```
variable "image_id" {
  validation {
    condition     = length(var.image_id) > 4 && substr(var.image_id, 0, 4) == "ami-"
    error_message = "The image_id value must be a valid AMI id, starting with \"ami-\"."
  }
}
```

Input Variables




```
variable "user_information" {
  type = object({
    name    = string
    address = string
})
  sensitive = true
}
```

```
variable "image_id" {
  nullable = false
}
```

Input Variables

```
variable "image_id" {
  default = "ami-0c7217cdde317cfec"
  type    = string
}
variable "instance_type" {
  default = "t2.micro"
  type    = string
}
variable "instance_name" {
  default = "thinknyxserver"
  type    = string
}
```

```
resource "aws_instance" "thinknyxserver" {
  ami           =
  instance_type =
  tags = {
    Name =
  }
}
```

```
variable "image_id" {}
```

```
tofu apply
```

```
var.image_id
   Enter a value: []
```

Input Variables

```
variable "image_id" {


}
variable "instance_type" {
  default = "t2.micro"
  type    = string
}
variable "instance_name" {
  default = "thinknyxserver"
  type    = string
}
```

```
resource "aws_instance" "thinknyxserver" {
  ami           =
  instance_type =
  tags = {
    Name =
  }
}
```

```
tofu apply
```

```
var.image_id
   Enter a value: []
```

Input Variables

```
tofu apply -var instance_type=t2.small
```

```
export TF_VAR_instance_type="t2.nano"
```

terraform.tfvars

```
image_id = "ami-0c7217cdde317cfec"
instance_type = "t2.large"
instance_name = "thinknyxserver"
```

```
tofu apply
```

✓ "terraform.tfvars"

✓ ".auto.tfvars"

Demo | Input Variables

Demo | Input Variables with -var option

Demo | Input Variables with Environment variables

Demo | Input Variables with .tfvars files

Demo | Input Variables Precedence

Variable - Type constraints

Keywords

- ✓ String
- ✓ Number
- ✓ Bool

Constructors

- ✓ List
- ✓ Set
- ✓ Map
- ✓ Object
- ✓ Tuple

any

Variable - Type constraints

```
variable "example_string" {
  default = "Hello, OpenTofu!"
  type    = string
}
```

```
variable "example_number" {
  default = 42
  type    = number
}
```

```
variable "enable_feature" {
  default = true
  type    = bool
}
```

```
variable "example_list" {
  default = ["value1","value2","value3"]
  type    = list(string)
}
```

```
variable "example_map" {
  default = {
    key1 = "value1"
    key2 = "value2"
  }
  type    = map(string)
}
```

```
variable "example_any" {
  type = any
}
```

Variable - Type constraints

```
variable "example_string" {
  default = "Hello, OpenTofu!"
  type    = string
}
```

```
variable "example_number" {
  default = 42
  type    = number
}
```

```
variable "enable_feature" {
  default = true
  type    = bool
}
```

```
variable "example_list" {
  default = ["value1","value2","value3"]
  type    = list(string)
}
```

['a', 15, true] – **list(any)**

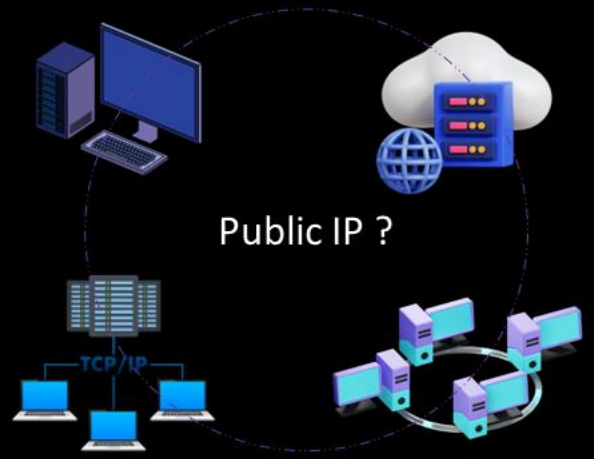
Demo | Variable - Type constraints

Demo | Advance use of Variables

OpenTofu | Output Values

OpenTofu Output Values

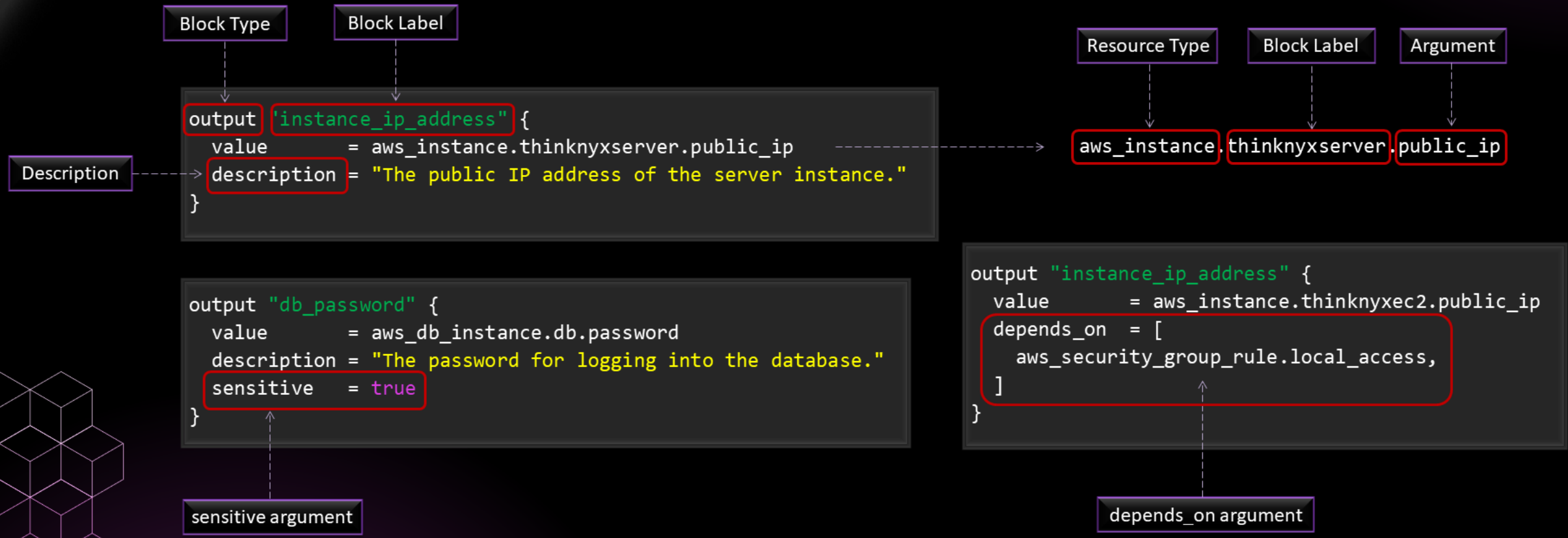
- Output block allows to retrieve and display critical information



- ✓ Output Block – Extract the values from provisioned resources

OpenTofu Output Values

- ✓ Value Argument - Takes an expression. Value is a mandatory argument.
- ✓ Description - Optional argument



Demo | Tofu Console Command and Output Block

OpenTofu | Provisioners

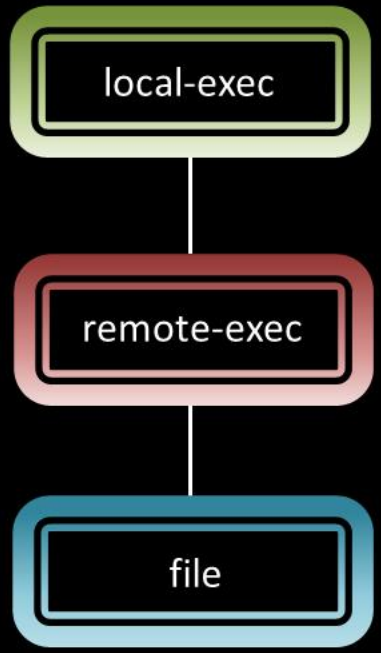
Introduction to Provisioners

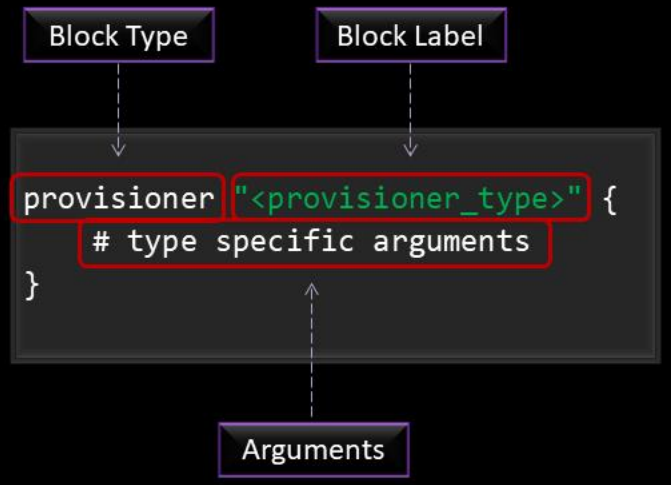
- Help in executing Scripts or Commands
- Vital for automating post-deployment tasks

AWS EC2 Instance

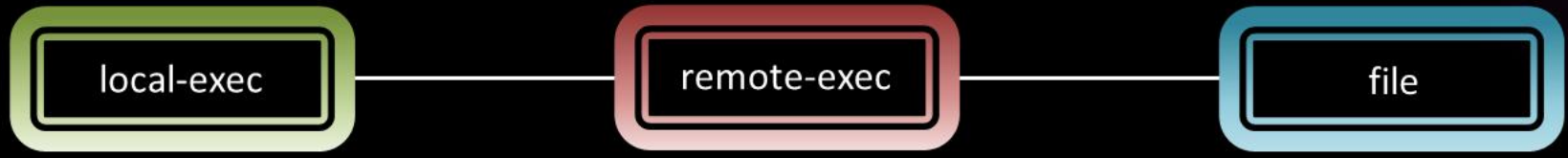
- Provisioners: remote-exec
  - ✓ Automatic install and configure components
  - ✓ Perform post-provisioning steps on the provisioned resources

Types of Provisioners





Types of Provisioners



Provisioner Type

```
provisioner "local-exec" {
  command = "echo ${self.public_ip} >> public_ips.txt"
}
```

Arguments

```
provisioner "local-exec" {
    when    = destroy
    command = "echo 'Instance deleted'"
  }
```

Arguments

- ✓ create
- ✓ destroy
- ✓ change

Types of Provisioners

Provisioner Type

```
provisioner "remote-exec" {
   script/scripts = "path of script/scripts"
 }
```

Arguments

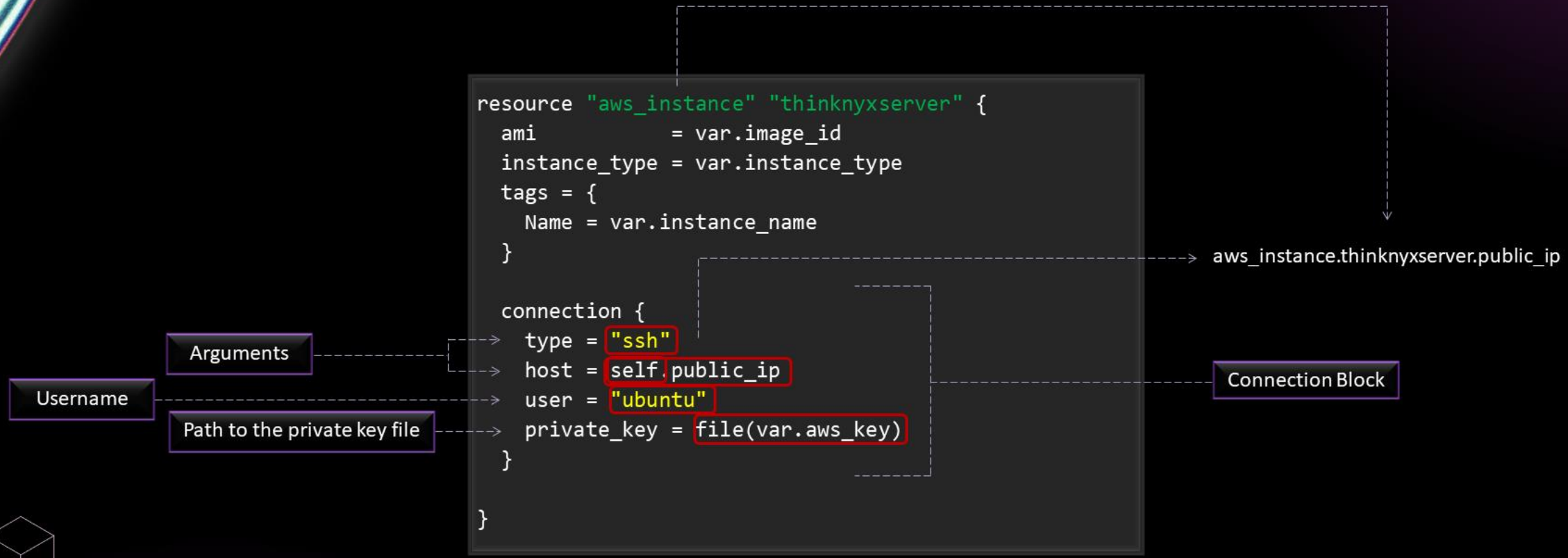
Provisioner Type

```
provisioner "file" {
    source      = "thinknyx.txt"
    destination = "/tmp/thinknyx.txt"
  }
```

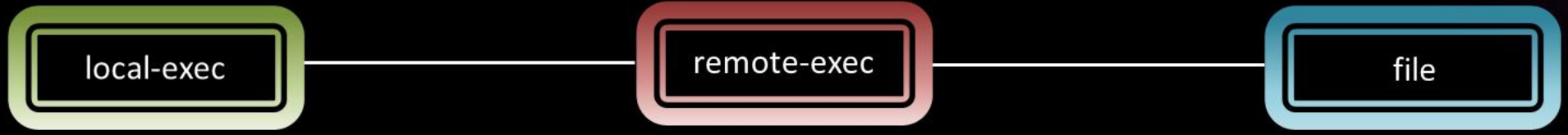
Arguments

Arguments

Connection Block



Connection Block



- ✓ Runs on local machine
- X No Connection Block

- ✓ Requires Connection Block

- ✓ Requires Connection Block

Why Provisioners should be a Last Resort

- Due to limitation and potential challenges
- Better to use built-in tools
- Introduce complexity and reliance on external scripts
- Can also pose challenges
- Recommendation: Explore other OpenTofu features and best practices

Pragmatically

Document Thoroughly

Chef

Puppet

Ansible

Demo | Local Provisioner | When and On-failure options

Demo | Remote Provisioner

Demo | File Provisioner

OpenTofu | Data Sources

Case Study

- Hardcoded all values, such as AMI ID, Instance type, etc.
- Two significant concerns:
  1. Outdated details
  2. Dependency on specific regions
- Equally applicable to private images
- Crucial to address these challenges for longevity and adaptability

Data Sources

Data Sources

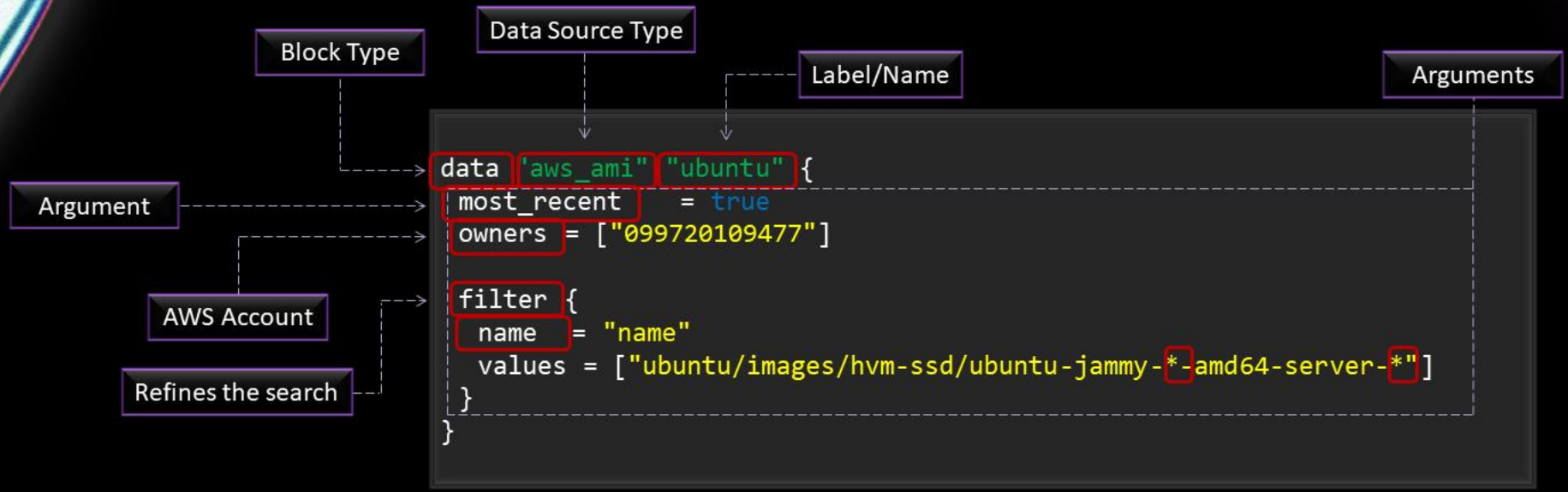
Providers

- OpenTofu Resources
- OpenTofu Data Sources

Data Sources

- Allow to use information defined outside of OpenTofu

Data Sources



data.[data_source_type].[data_source_name].[argument_name]

```
resource "aws_instance" "thinknyxserver" {
 ami           = data.aws_ami.ubuntu.id
 instance_type = var.instance_type
 tags = {
  Name = var.instance_name
 }
}
```

Resource vs Data Source

Resources

- Blueprint for creating, managing, or deleting infrastructure components
- Articulate the desired state and directly interface with cloud provider's API

Data Sources

- Extracting Read-only information
- Fetch specific attributes needed in OpenTofu configuration

Demo | Challenges without Data Sources

Demo | Resolution using Data Sources

OpenTofu | Modules and Registries

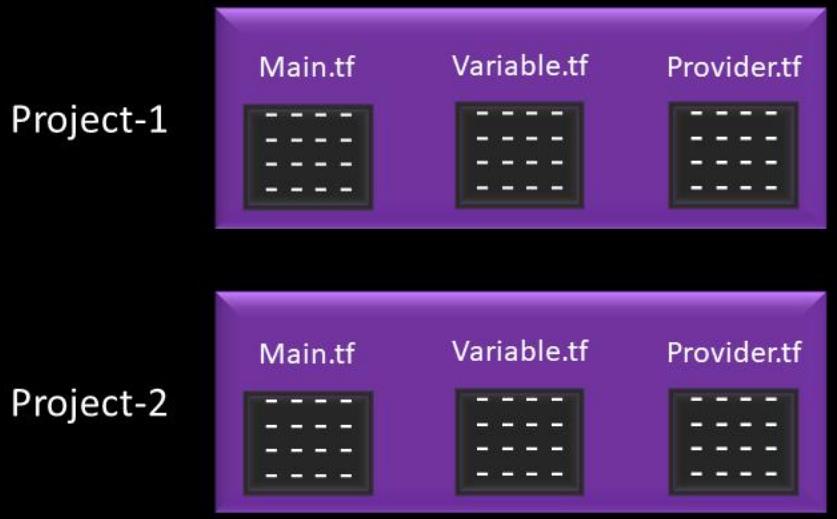
Why Modules?

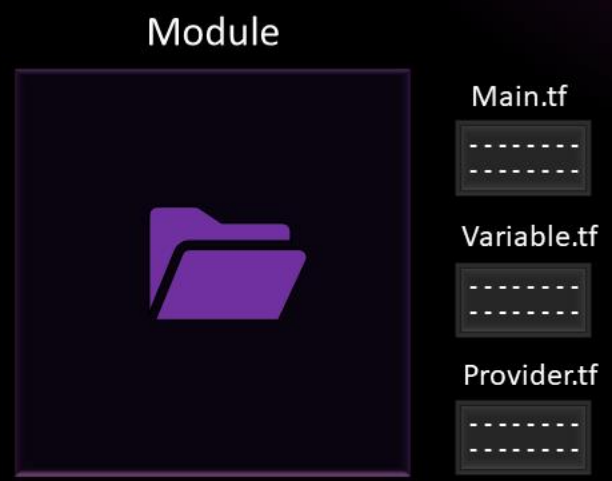
Project-1

Project-2

Module

Why Modules?





- ✓ Reusable, scalable, customizable
- ✓ Package and reuse configurations
- ✓ Sharable, plug-and-play solution

What are Modules?

Module

- Root Modules
  - Create, update, and destroy services
  - Every configuration – At least one root module
  - Can also reference and use child modules
- Child Modules
  - Encapsulate resources or functionalities
  - Reusable pieces of infrastructure code
  - Promotes code organization, reusability, and maintainability

What are Modules?


```
mkdir -p /var/tmp/modules/web_servers
```

```
cd /var/tmp/modules/web_servers
```

main.tf

```
variable "instance_type" {
 description = "The type of EC2 instance"
 default     = "t2.micro"
}

resource "aws_instance" "thinknyxserver" {
 ami           = "ami-12345678"
 instance_type = var.instance_type
}
```

```
cd /var/tmp
```

main.tf

```
module "web_servers" {
  source        = "./modules/web_servers"
  instance_type = "t2.small"
}
```

❑ Version argument - Optional

```
tofu init
```

✓ Local paths

✓ Module Registry

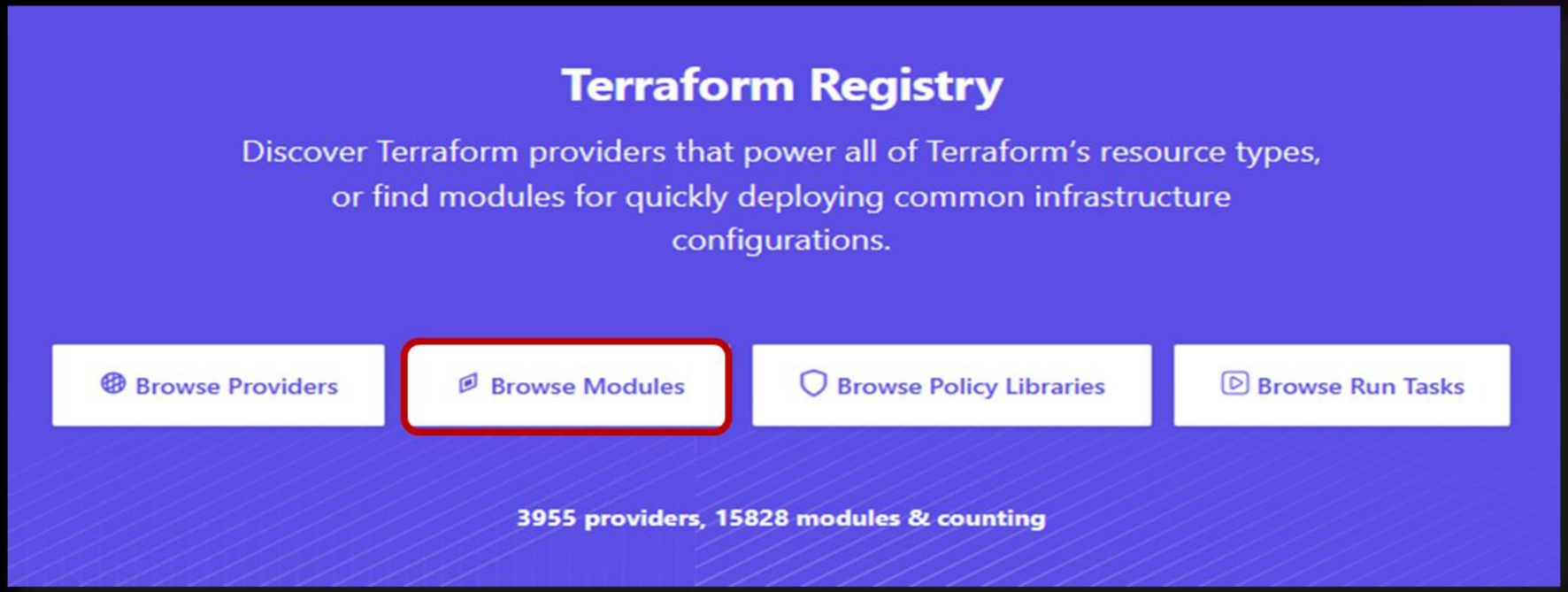
✓ GitHub

✓ S3 Buckets

for_each

depends_on

Module Registry



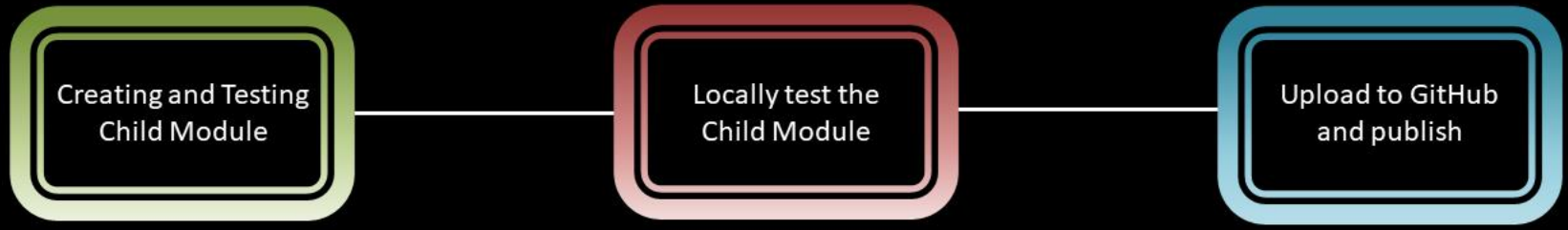

```
module "ec2-instance" {
  source  = "terraform-aws-modules/ec2-instance/aws"
  version = "5.6.0"
}
```

Demo | Navigating Modules Documentation

Demo | Module Registry

Creating Custom Modules

- Our own Module - Custom Modules



Demo | Create Child Module

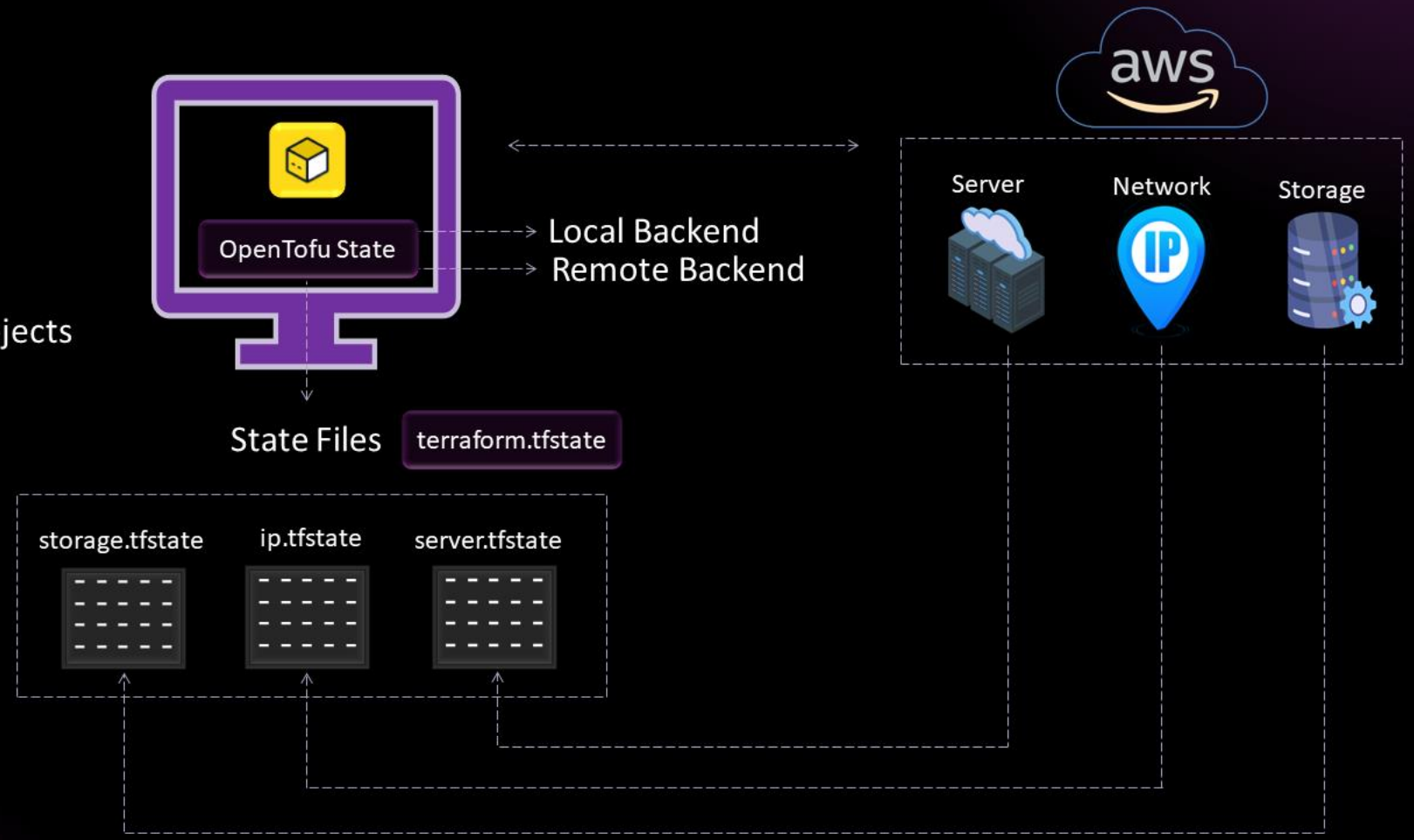
Demo | Create Root Module

Demo | Publish Code to Registry

OpenTofu | State and State Files

OpenTofu State and State Files

- To understand the current state
- Determine the necessary action
- Performs a refresh
- Maintain association between objects
- One-to-one mapping



State Files Essence

Purpose of OpenTofu State

Mapping to the real world

terraform.tfstate

Centralizing

Metadata

terraform.tfstate.backup

Speed up
Operations

Tofu State Command

`tofu state list`	List the resources and data sources
`tofu state show`	Show the resource state
`tofu state rm`	Remove the resource state
`tofu state mv`	Rename a resource
`tofu state replace-provider`	Replace provider
`tofu state pull`	Pull current state and update the local state copy
`tofu state push`	Update remote state from a local state file

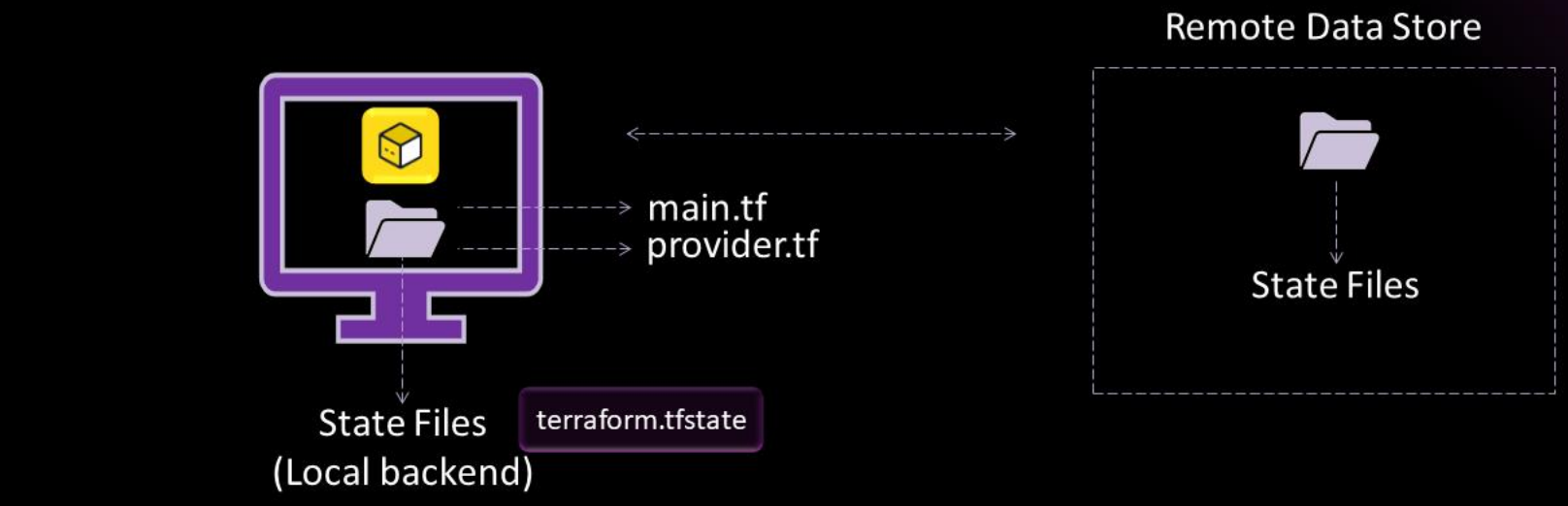
Demo | State File - terraform.tfstate and terraform.tfstate.backup

Demo | State Files in Action

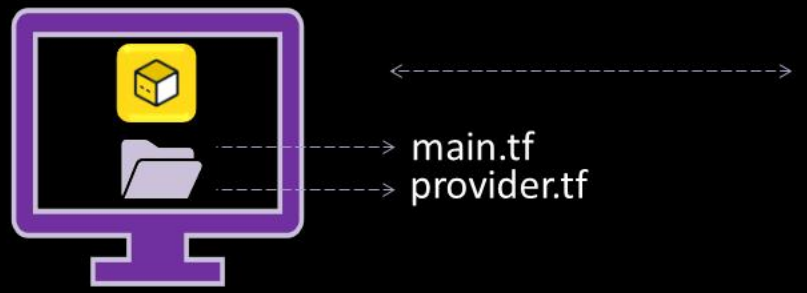
Demo | "tofu state" CLI

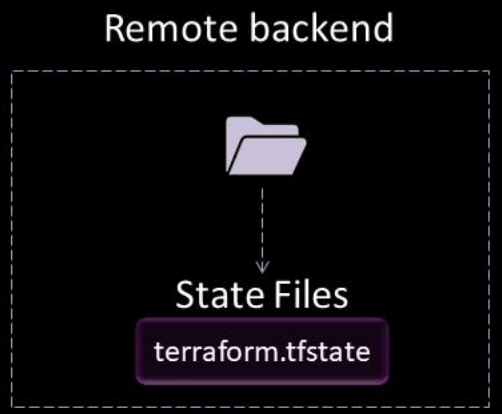
OpenTofu | Remote State and Backends

Remote State



OpenTofu Backend





- ✓ Offers enhanced locking mechanism

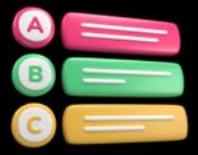
- ❑ HashiCorp Consul
- ❑ Amazon S3
- ❑ Microsoft Azure Blob Storage
- ❑ Google Cloud Storage
- ❑ Alibaba Cloud OSS

Why Backend?

Collaborate

Scalability

Consistency



History

Prevent Conflicts

Durable

Versioning

Security

OpenTofu AWS Remote Backend

S3 and DynamoDB

OpenTofu AWS Remote Backend

Step-1

```
resource "aws_s3_bucket" "thinknyx_bucket" {
  bucket        = "thinknyx-bucket"
  force_destroy = false
  tags = {
    Name        = "Thinknyx bucket"
    Environment = "Dev"
  }
}
```

Step-2

```
resource "aws_dynamodb_table" "thinknyx_lock_dynamodb" {
  name         = "thinknyx-dynamodb-lock-table"
  billing_mode = "PAY_PER_REQUEST"
  hash_key     = "LockID"
  attribute {
    name = "LockID"
    type = "S"
  }
  tags = {
    "Name" = "Thinknyx DynamoDB Terraform State Lock Table"
  }
}
```

Step-3

```
terraform {
 backend "s3" {
   bucket       = "thinknyx-bucket"
   key          = "path/to/your/key"
   region       = "us-east-1"
   dynamodb_table = "thinknyx-dynamodb-lock-table"
 }
}
```

Demo | Preparing Remote Backend

Demo | Implementing Remote Backend

OpenTofu | Working with OpenTofu

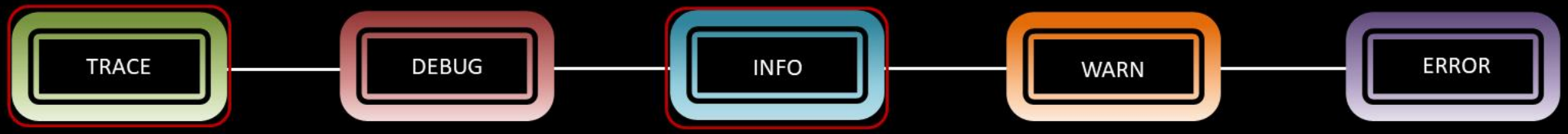
Timeouts

- Special configuration option
- Default time interval
- Configurable Block

```
resource "aws_instance" "thinknyxserver" {
  ami           = "ami-0fbc7ac82ff7690bc"
  instance_type = "t2.micro"

  tags = {
    Name       = "udemycourse"
    coursename = "OpenTofuCourse"
    release    = "March24"
  }
  timeouts {
    create = "20m"
    update = "25m"
  }
}
```

Debugging

- Prebuilt functionality
- Inbuilt environment variable - TF_LOG



- Setting log levels to TRACE, INFO
- Default logging is not enabled by default
- OpenTofu Environment Variables :

TF_LOG	To set use "export TF_LOG=INFO"
TF_LOG_PATH	To set use "export TF_LOG_PATH=/var/tmp/tofu.logs"

Demo | Operation Timeouts and Debugging

OpenTofu Taint

- ✓ Changes highlighted in Plan

instance_type = t2.small

OpenTofu Taint

main.tf

```
---------------
---------------
----t2.micro ---
---------------
---------------
```

```
tofu taint aws_instance.thinknyxserver
```

```
tofu plan
```

"aws_instance.thinknyxserver" is tainted, so must be replaced.

```
tofu untaint aws_instance.thinknyxserver
```

```
tofu plan
```

```
resource "aws_instance" "thinknyxserver" {
  ami           = "ami-0fbc7ac82ff7690bc"
  instance_type = "t2.micro"

  tags = {
    Name       = "udemycourse"
    coursename = "OpenTofuCourse"
    release    = "March24"
  }
  timeouts {
    create = "20m"
    update = "25m"
  }
}
```

Demo | Taint and Untaint

OpenTofu Graph Command

```
tofu graph
```

- Dot output format : Text based graph description language

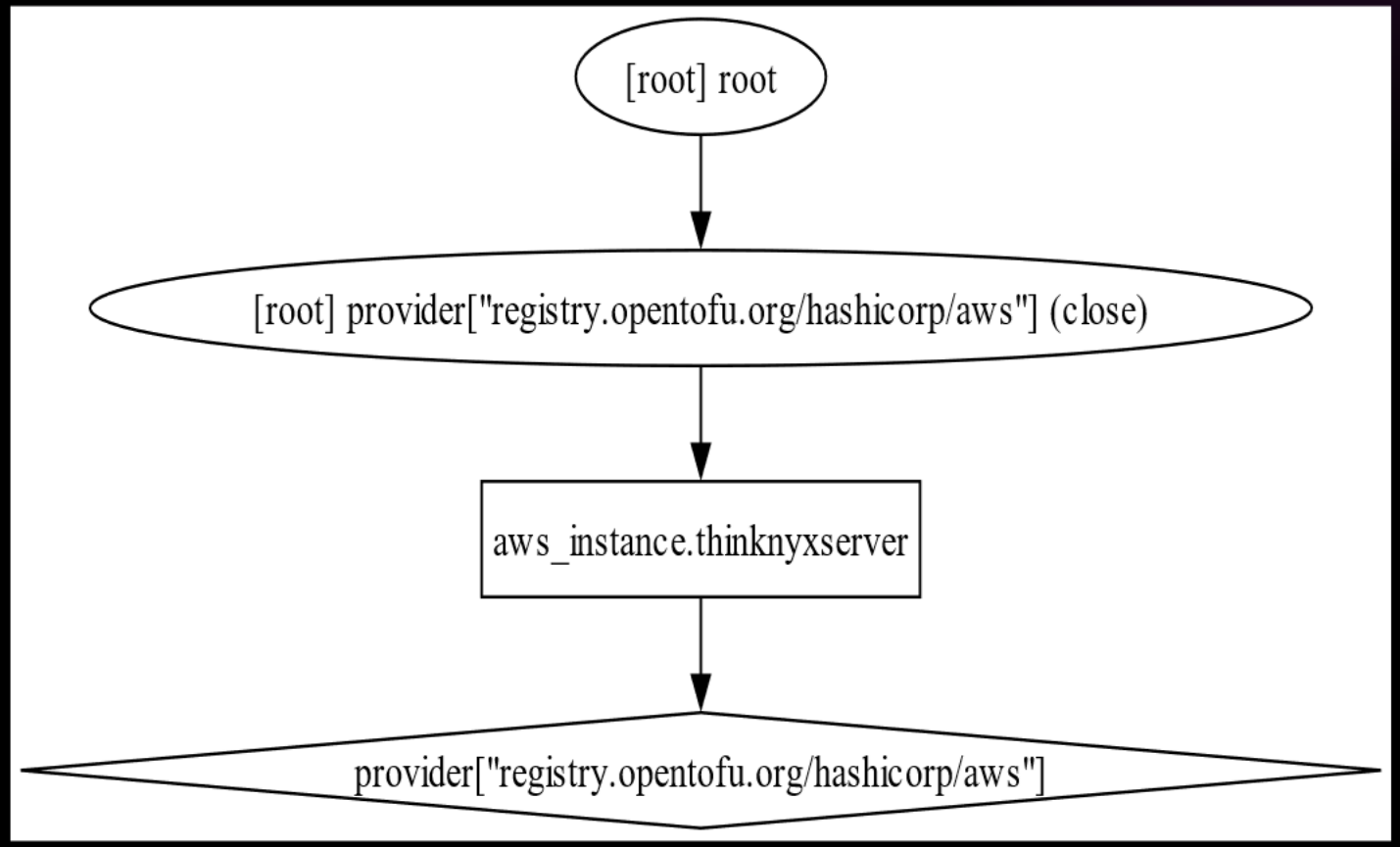
```
user1@Thinknyx-MacBook tofutest % tofu graph
digraph {
	compound = "true"
	newrank = "true"
	subgraph "root" {
		"[root] aws_instance.thinknyxserver (expand)" [label =
"aws_instance.thinknyxserver", shape = "box"]
		"[root] provider[\"registry.opentofu.org/hashicorp/aws\"]" [label =
"provider[\"registry.opentofu.org/hashicorp/aws\"]", shape = "diamond"]
		"[root] aws_instance.thinknyxserver (expand)" -> "[root]
provider[\"registry.opentofu.org/hashicorp/aws\"]"
		"[root] provider[\"registry.opentofu.org/hashicorp/aws\"] (close)"
-> "[root] aws_instance.thinknyxserver (expand)"
		"[root] root" -> "[root]
provider[\"registry.opentofu.org/hashicorp/aws\"] (close)"
	}

}
```

OpenTofu Graph Command

```
tofu graph
```

```
user1@Thinknyx-MacBook tofutest % tofu graph
digraph {
	compound = "true"
	newrank = "true"
	subgraph "root" {
		"[root] aws_instance.thinknyxserver (expand)" [label =
"aws_instance.thinknyxserver", shape = "box"]
		"[root] provider[\"registry.opentofu.org/hashicorp/aws\"]" [label =
"provider[\"registry.opentofu.org/hashicorp/aws\"]", shape = "diamond"]
		"[root] aws_instance.thinknyxserver (expand)" -> "[root]
provider[\"registry.opentofu.org/hashicorp/aws\"]"
		"[root] provider[\"registry.opentofu.org/hashicorp/aws\"] (close)"
-> "[root] aws_instance.thinknyxserver (expand)"
		"[root] root" -> "[root]
provider[\"registry.opentofu.org/hashicorp/aws\"] (close)"
	}
}
```

```
tofu graph | dot -Tsvg > graph.svg
```



https://www.graphviz.org/

Demo | "tofu graph" CLI Usage

Import

```
tofu import
```

- ✓ State files
- ✓ No .tf files will be created
- ✓ Manual efforts

Import

Step - 1

```
resource "aws_instance" "testserver" {

}
```

Step - 2

```
tofu import aws_instance.testserver i-0749ed3551175a815
```

```
user1@Thinknyx-MacBook importtest % tofu import aws_instance.testserver i-0749ed3551175a815

╷
│ Error: Inconsistent dependency lock file
│
│ The following dependency selections recorded in the lock file are inconsistent with the current configuration:
│   - provider registry.opentofu.org/hashicorp/aws: required by this configuration but no version is selected
│
│ To make the initial dependency selections that will initialize the dependency lock file, run:
│   tofu init
╵
```

```
tofu init
```

Import

Step - 3

```
tofu import aws_instance.testserver i-0749ed3551175a815
```

```
user1@Thinknyx-MacBook importtest % tofu import aws_instance.testserver i-0749ed3551175a815
aws_instance.testserver: Importing from ID "i-0749ed3551175a815"...
aws_instance.testserver: Import prepared!
  Prepared aws_instance for import
aws_instance.testserver: Refreshing state... [id=i-0749ed3551175a815]

Import successful!

The resources that were imported are shown above. These resources are now in
your OpenTofu state and will henceforth be managed by OpenTofu.

user1@Thinknyx-MacBook importtest %
```

Import

Step - 4

```
tofu plan
```

```
resource "aws_instance" "testserver" {
  ami = "ami-0c7217cdde317cfec"
  instance_type = "t2.micro"
}
```

```
user1@Thinknyx-MacBook importtest % tofu plan

╷
│ Error: Missing required argument
│
│   with aws_instance.testserver,
│   on main.tf line 1, in resource "aws_instance" "testserver":
│    1: resource "aws_instance" "testserver" {
│
│ "ami": one of `ami,launch_template` must be specified
╵

╷
│ Error: Missing required argument
│
│   with aws_instance.testserver,
│   on main.tf line 1, in resource "aws_instance" "testserver":
│    1: resource "aws_instance" "testserver" {
│
│ "instance_type": one of `instance_type,launch_template` must be specified
╵

╷
│ Error: Missing required argument
│
│   with aws_instance.testserver,
│   on main.tf line 1, in resource "aws_instance" "testserver":
│    1: resource "aws_instance" "testserver" {
│
│ "launch_template": one of `ami,instance_type,launch_template` must be specified
```

Import

Step - 5

```
tofu plan
```

```
user1@Thinknyx-MacBook importtest % tofu plan
aws_instance.testserver: Refreshing state... [id=i-0749ed3551175a815]

No changes. Your infrastructure matches the configuration.

OpenTofu has compared your real infrastructure against your configuration and found
no differences, so no changes are needed.
```

Demo | Importing Existing Infrastructure

Local Values Block

```
resource "aws_instance" "thinknyxserver" {
  ami           = "ami-0fbc7ac82ff7690bc"
  instance_type = "t2.micro"
  availability_zone = "us-east-1a"

}

resource "aws_ebs_volume" "thinknyxvolume" {
  availability_zone = "us-east-1a"
  size = 1
  type = "standard"

}
```

Tags

Use same tags across various services for same project

```
resource "aws_instance" "thinknyxserver" {
  ami           = "ami-0fbc7ac82ff7690bc"
  instance_type = "t2.micro"
  availability_zone = "us-east-1a"

  tags = {
    Name       = "udemycourse"
    coursename = "OpenTofuCourse"
    release    = "March24"
  }
}

resource "aws_ebs_volume" "thinknyxvolume" {
  availability_zone = "us-east-1a"
  size = 1
  type = "standard"

  tags = {
    Name       = "udemycourse"
    coursename = "OpenTofuCourse"
    release    = "March24"
  }
}
```

Local Values Block

```
resource "aws_instance" "thinknyxserver" {
  ami           = "ami-0fbc7ac82ff7690bc"
  instance_type = "t2.micro"
  availability_zone = "us-east-1a"

}

resource "aws_ebs_volume" "thinknyxvolume" {
  availability_zone = "us-east-1a"
  size = 1
  type = "standard"

}

locals {
  project_tags = {

    Name       = "udemycourse"
    coursename = "OpenTofuCourse"
    release    = "March24"

  }
}
```

Local Values Block

```
resource "aws_instance" "thinknyxserver" {
  ami           = "ami-0fbc7ac82ff7690bc"
  instance_type = "t2.micro"
  availability_zone = "us-east-1a"

  tags = local.project_tags

}

resource "aws_ebs_volume" "thinknyxvolume" {
  availability_zone = "us-east-1a"
  size = 1
  type = "standard"

  tags = local.project_tags

}

locals {
  project_tags = {

    Name       = "udemycourse"
    coursename = "OpenTofuCourse"
    release    = "March24"

  }
}
```

Demo | Local Values Block

Alias

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "5.37.0"
    }
  }
}

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

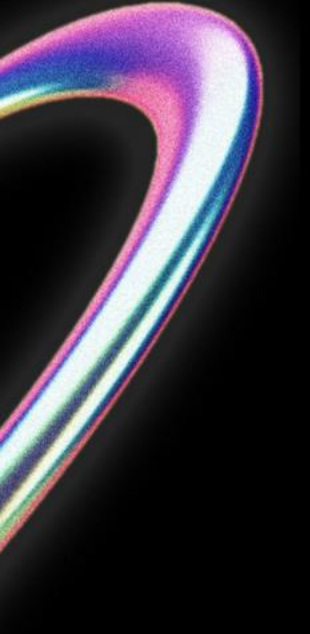
```
resource "aws_eip" "thinknyxeipone" {

}

resource "aws_eip" "thinknyxeiptwo" {

}
```

Alias

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "5.37.0"
    }
  }
}

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

provider "aws" {
  region = "us-west-1"
  alias = “uswest"
}
```

```
resource "aws_eip" "thinknyxeipone" {

}

resource "aws_eip" "thinknyxeiptwo" {
  provider = aws.uswest

}
```

Demo | Provider Aliases



OpenTofu | Meta-Arguments, Functions and Workspaces

Lifecycle Meta-Argument

- ✓ Changes highlighted in Plan

instance_type = t2.small

```
OpenTofu used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
-/+ destroy and then create replacement

OpenTofu will perform the following actions:

  # aws_instance.thinknyxserver must be replaced
-/+ resource "aws_instance" "thinknyxserver" {
      ~ ami                                  = "ami-0440d3b780d96b29d" -> "ami-030ff268bd7b4e8b5" # fo
rces replacement
      ~ arn                                  = "arn:aws:ec2:us-east-1:554660509057:instance/i-0b83a622
5434e7187" -> (known after apply)
      ~ associate_public_ip_address          = true -> (known after apply)
      ~ availability_zone                    = "us-east-1c" -> (known after apply)
```

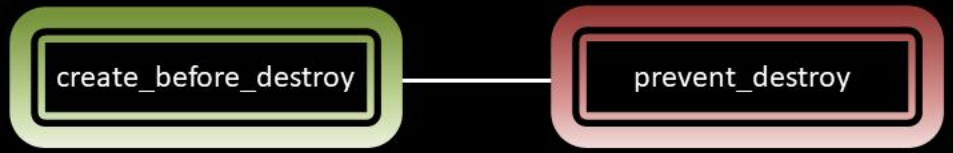
create_before_destroy — prevent_destroy — ignore_changes — replace_triggered_by

Lifecycle Meta-Argument

create_before_destroy

```
resource "aws_instance" "thinknyxserver" {
  ami           = "ami-0fbc7ac82ff7690bc"
  instance_type = "t2.micro"
  tags = {
    Name        = "TofuDemo"
  }
  lifecycle {
    create_before_destroy = true
  }
}
```

Lifecycle Meta-Argument




```
resource "aws_instance" "thinknyxserver" {
  ami           = "ami-0fbc7ac82ff7690bc"
  instance_type = "t2.micro"
  tags = {
    Name        = "TofuDemo"
  }
  lifecycle {
    prevent_destroy = true
  }
}
```

Lifecycle Meta-Argument


```
resource "aws_instance" "thinknyxserver" {
  ami           = "ami-0fbc7ac82ff7690bc"
  instance_type = "t2.micro"
  tags = {
    Name        = "TofuDemo"
  }
  lifecycle {
    ignore_changes = [
      tags,instance_type
    ]
  }
}
```

Lifecycle Meta-Argument


```
resource "aws_instance" "thinknyxserver" {
  ami           = "ami-0fbc7ac82ff7690bc"
  instance_type = "t2.micro"
  tags = {
    Name        = "TofuDemo"
  }
  lifecycle {
    replace_triggered_by = [
      aws_vpc.thinknyxvpc.id
    ]
  }
}
```

Demo | Lifecycle Rules

depends_on Meta-Argument

```
resource "aws_instance" "thinknyxserver" {
  ami           = "ami-0fbc7ac82ff7690bc"
  instance_type = "t2.micro"
  availability_zone = "us-east-1a"
  tags = local.project_tags
  depends_on = [
    aws_ebs_volume.thinknyxvolume
  ]
}
resource "aws_ebs_volume" "thinknyxvolume" {
  availability_zone = "us-east-1a"
  size = 1
  type = "standard"
  tags = local.project_tags
}
locals {
  project_tags = {
     Name       = "udemycourse"
     coursename = "OpenTofuCourse"
     release    = "March24"
  }
}
```

```
OpenTofu will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_ebs_volume.thinknyxvolume: Creating...
aws_ebs_volume.thinknyxvolume: Still creating... [10s elapsed]
aws_ebs_volume.thinknyxvolume: Creation complete after 12s
[id=vol-0d05bfc072a236999]
aws_instance.thinknyxserver: Creating...
aws_instance.thinknyxserver: Still creating... [10s elapsed]
aws_instance.thinknyxserver: Still creating... [20s elapsed]
aws_instance.thinknyxserver: Still creating... [30s elapsed]
aws_instance.thinknyxserver: Creation complete after 36s [id=i-
036c20abafdeedb56]

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

Demo | depends_on Meta-Argument

count Meta-Argument

```
resource "aws_instance" "thinknyxserver" {
  ami           = "ami-0fbc7ac82ff7690bc"
  instance_type = "t2.micro"
  tags = {
    Name = thinknyxdemo
  }
  count = 2
}
```

```
resource "aws_instance" "thinknyxserver" {
  ami           = "ami-0fbc7ac82ff7690bc"
  instance_type = "t2.micro"
  tags = {
    Name = var.servername[count.index]
  }
  count = 2
}
variable "servername" {
  default = [
    "demoone",
    "demotwo"
  ]
}
```

Demo | count Meta-Argument

for_each Meta-Argument

```
provider "aws" {
  region = "us-east-1"
}
resource "aws_instance" "thinknyxserver" {
  ami           = "ami-0fbc7ac82ff7690bc"
  instance_type = "t2.micro"
  tags = {
    Name = each.value
  }
  for_each = var.servername
}
variable "servername" {
  type = map(string)
  default = {
    one = "demoone",
    two = "demotwo",
  }
}
```

Demo | for_each Meta-Argument

Functions

```
<FUNCTION NAME>(<ARGUMENT 1>, <ARGUMENT 2>)
```

```
user1@Thinknyx-MacBook % tofu console
> min(100,50,900)
50
```

file	Use to read and display the content
length	Shows the number of elements of a list
toset	Converts the list into a set
sensitive	To mark a value as sensitive
lookup	To extract the value of a single element from a map

Demo | Built-in Functions

OpenTofu Workspaces

```
resource "aws_instance" "thinknyxserver" {
  ami           = "ami-0fbc7ac82ff7690bc"
  instance_type = "t2.micro"
  tags = {
    Name  = "projectone"
  }
}
```

- Maintain multiple copies of deployments

```
tofu workspace list
```

```
user1@Thinknyx-MacBook % tofu workspace list
* default
```

OpenTofu Workspaces	OpenTofu Modules
A concept in OpenTofu to create and manage different set of resources using same configuration by isolating state files.	Logical bundle for multiple resources to work together for better structuring and reusability.

OpenTofu Workspaces

Step - 1

```
resource "aws_instance" "thinknyxserver" {
  ami           = "ami-0fbc7ac82ff7690bc"
  instance_type = "t2.micro"
  tags = {
    Name  = "projectone"
  }
}
```

Step - 2

```
user1@Thinknyx-MacBook % tofu workspace list
* default
```

```
user1@Thinknyx-MacBook % tofu workspace new projectone
Created and switched to workspace "projectone"!

You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "tofu plan" OpenTofu will not see any existing state
for this configuration.
```

Step - 3

```
user1@Thinknyx-MacBook % tofu workspace list
  default
* projectone
```

```
tofu apply
```

Step - 4

```
user1@Thinknyx-MacBook % tree
.
├── main.tf
└── terraform.tfstate.d
    └── projectone
        └── terraform.tfstate
3 directories, 2 files
```

Demo | Workspaces in OpenTofu

www.thinknyx.com

https://github.com/thinknyx-technologies-llp/InfrastructureAutomationWithOpenTofu