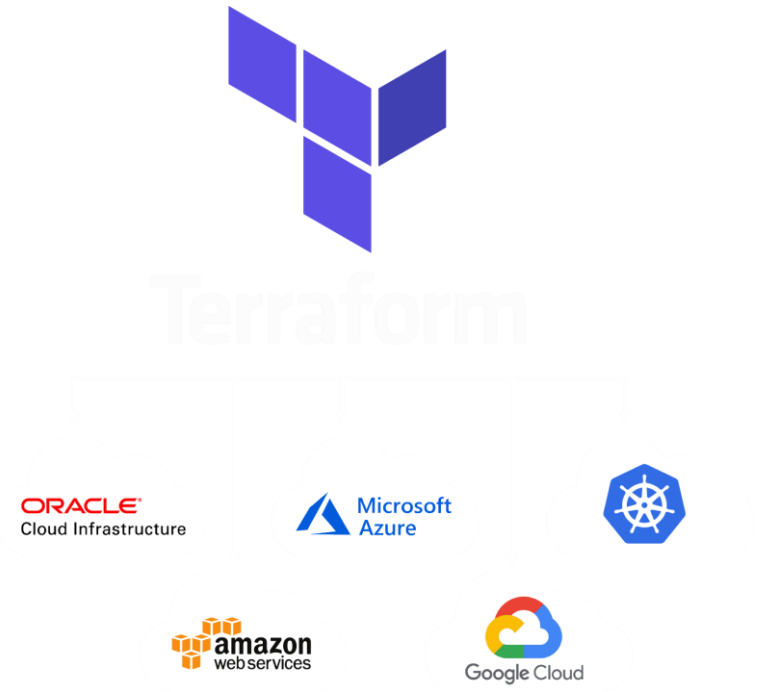
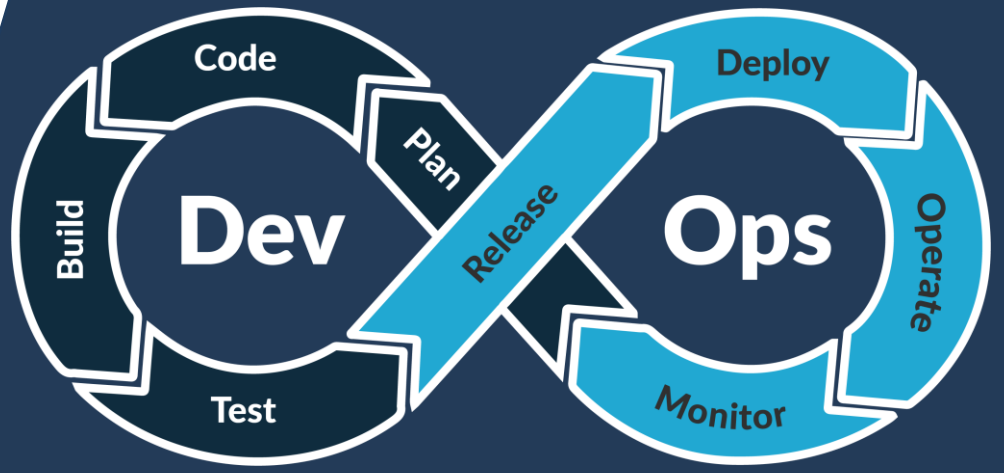
K21Academy
Learn Cloud from Experts



Terraform

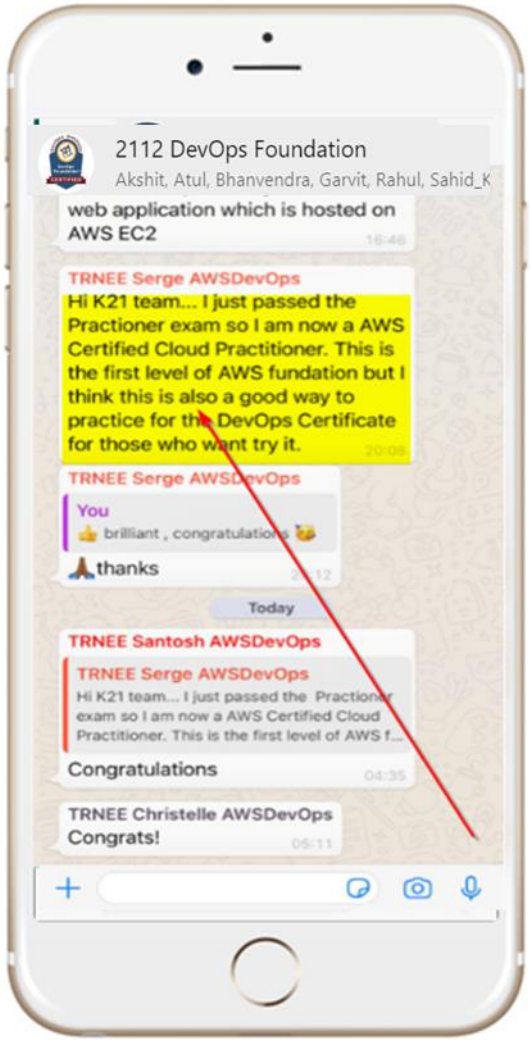


Getting Help

K21Academy
Learn Cloud from Experts

support@k21academy.com

zendesk



K21Academy
Learn Cloud from Experts

Module Agenda

Agenda Module

- Understanding Infrastructure as Code (IaC)
- Advantages of IaC
- Complete Stack of IaC – Devops SDLC Stack
- Terraform & its Benefits
- Multi-Cloud Benefits
- Terraform Architecture
- How Terraform Works
- Wrap-up
- Terraform Installation
- Providers
- Plugin based Architecture
- Terraform Fetches Providers
- Terraform Execution Workflow

K21Academy
Learn Cloud from Experts

Infrastructure as Code (IaC)

Overview

Terraform

Developed By	Hashicorp
Released	2014
Latest Release	1.3.2
Written in	GO
Provider Support	More than 2500+

Providers

- Major Cloud
- Cloud
- Infrastructure Software
- Network
- VCS
- Monitor & System Management
- Database
- Misc.
- Community

Terraform at #4

K21Academy
Learn Cloud from Experts

The Forbes Cloud List -2022

	Rank	Company	Category	HQ	Employee #	Funding	CEO
stripe	#1	Stripe	Payments software	Dublin, Ireland; San Francisco, United States	4,000	$2.2 B	Patrick Collison
databricks	#2	Databricks	AI-focused data platform	San Francisco, California	2,300	$1.9 B	Ali Ghodsi
Canva	#3	Canva	Design software	Sydney, Australia	1,700	$360 M	Melanie Perkins
HashiCorp	#4	HashiCorp	Infrastructure automation	San Francisco, California	1,500	$349 M	David McJannet

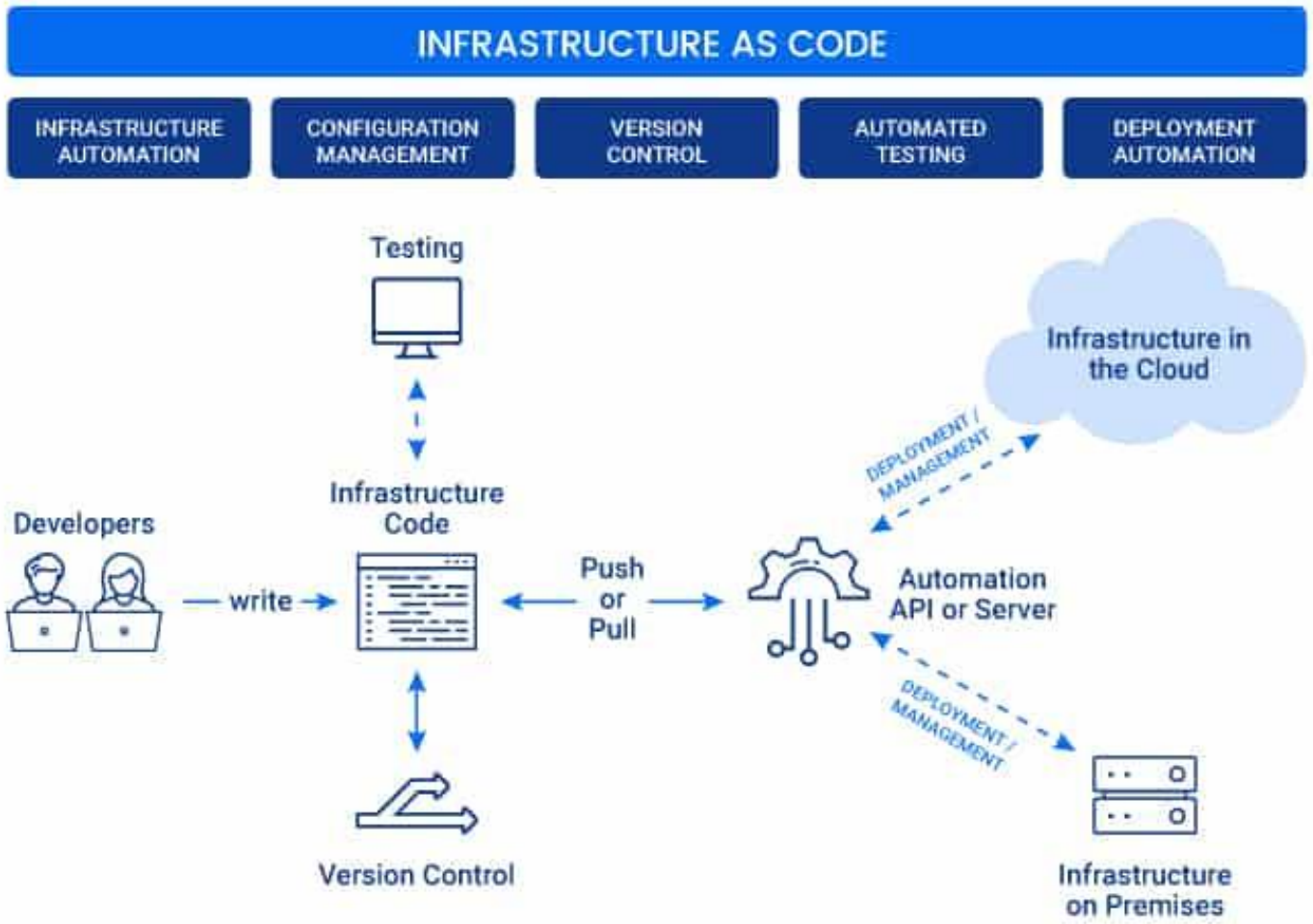
Infrastructure as Code

- Infrastructure is resources on which application runs on.
- Infrastructure as Code
  - Defining & Provisioning
  - Management of Infrastructure
  - Descriptive Model
  - Versioning
- Any (On-prem or Cloud) infrastructure with programmatic interface can participate in IaC
- IaC is a key DevOps practice

When to use IaC

- You use a large amount of IaaS resources.
- Your infrastructure is rented from many different providers or platforms.
- You need to make regular adjustments to your infrastructure.
- You need proper documentation of changes made to your infrastructure.
- You want to optimize collaboration between administrators and developers.

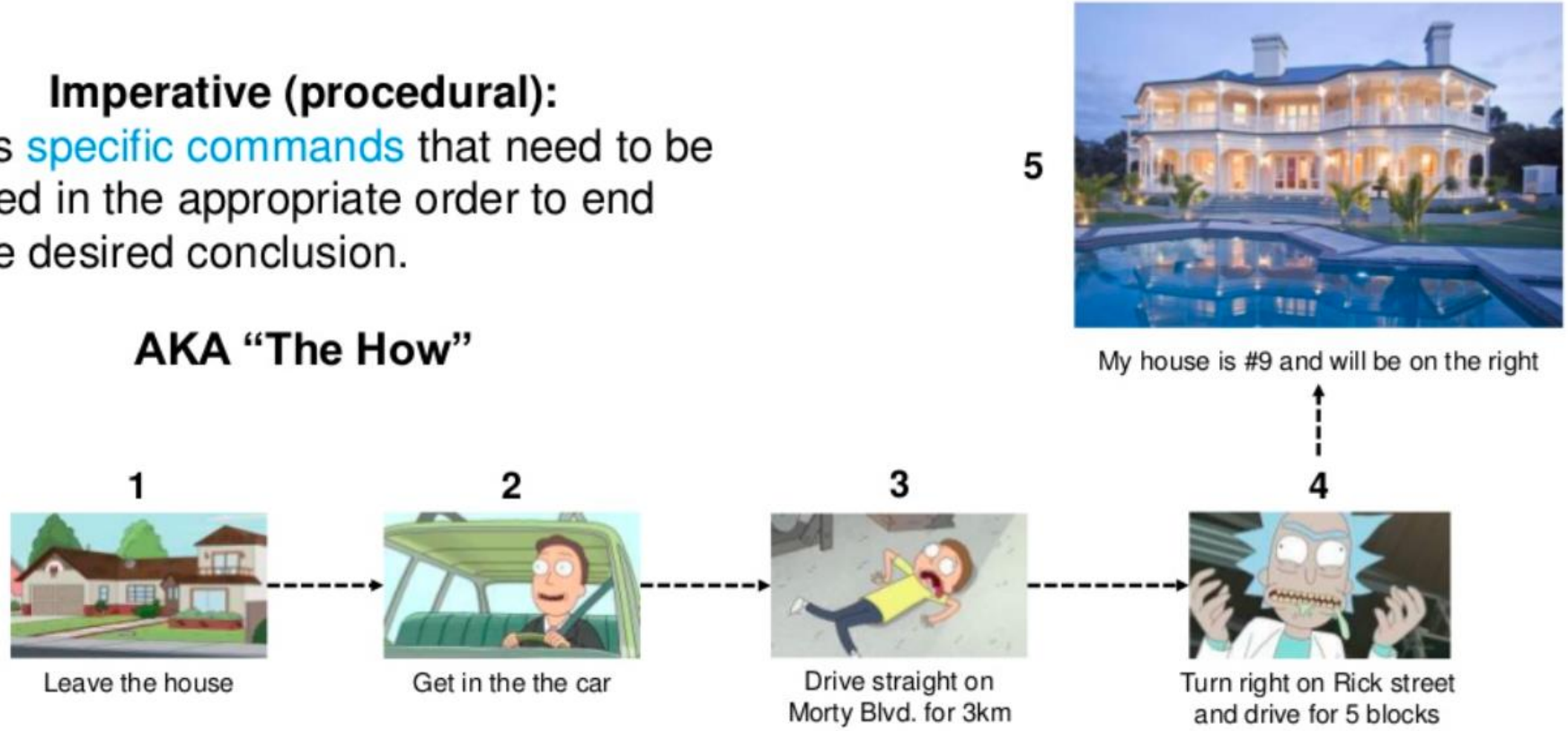
Infrastructure as Code



IaC : Imperative

Imperative (procedural):
Defines specific commands that need to be executed in the appropriate order to end with the desired conclusion.

AKA "The How"



Imperative - focus on the **actual provisioning process** and may reference a file containing a list of settings and configuration values

IaC : Declarative

Declarative (functional):
Defines the desired state and the system executes what needs to happen to achieve that desired state.

AKA "The What"

Declarative - focused on the **desired end state** of a deployment and rely on an interpretation engine to create and configure the actual resources.

IaC : Advantage

Repeatability

Version Control

Speed

Validation

Documentation

Reuse

IaC Tools Categories

IaC Tools

Ad-hoc Scripts

```
# Update the apt-get cache
sudo apt-get update

# Install PHP
sudo apt-get install -y php

# Install Apache
sudo apt-get install -y apache2
```

Configuration Management

- Chef
- Ansible
- Puppet
- Saltack

Server Templating

- Packet
- Docker
- Vagrant

Server Provisioning

- Terraform
- ARM
- Cloud Formation
- Openstack Heat

IaC Tools - Overview

HashiCorp Vagrant

- Builds VMs using a workflow. Specify the base image (called a Box) in a Vagrantfile along with the steps to configure the VM.
- Vagrant does have Provisioners that allow you to deploy on clouds
- Best used for Creating pre-configured developer VMs within VirtualBox.

IaC Tools - Overview

Terraform

- Only tool to focus solely on creating, destroying and managing infrastructure components.
- Use the Hashicorp Configuration Language (HCL) to describe the infrastructure resources you need.
- Provider | Provisioners | Modules | Plan Phase | Apply Phase | Destroy
- Best suited for Managing infrastructure resources

IaC Tools - Overview

ANSIBLE

- Building infrastructure as well as deploying and configuring applications on top of them.
- Ansible is to run in push mode or pull mode.
- Module | Playbook | Role
- Best used for Ad hoc analysis as well as general-purpose, push based, agentless IaC tool

IaC Tools - Overview

CHEF

- Used for configuration management
- Workstation | Cookbook | Recipe | Server Nodes | Knife
- Best used for Deploying and configuring applications using a pull-based approach.

IaC Tools - Overview

- Popular tool for configuration management
- Client Server Model
- needs agents to be deployed on the target machines before puppet can start managing them
- Resource | Class | Manifest | Catalog | Module
- Best used for Deploying and configuring applications using a pull-based approach.

IaC Tools - Overview

Tool	Tool Type	Infrastructure	Architecture	Approach	Manifest Written Language
puppet	Configuration Management	Mutable	Pull	Declarative	Domain Specific Language (DSL) & Embedded Ruby (ERB)
CHEF	Configuration Management	Mutable	Pull	Declarative & Imperative	Ruby
ANSIBLE	Configuration Management	Mutable	Push	Declarative & Imperative	YAML
SALTSTACK	Configuration Management	Mutable	Push & Pull	Declarative & Imperative	YAML
Terraform	Provisioning	Immutable	Push	Declarative	HashiCorp Configuration Language (HCL)

K21Academy
Learn Cloud from Experts

Infrastructure Provisioning Tool Terraform & its Benefits

Cloud Vendor: Automation Tools

K21Academy
Learn Cloud from Experts

Azure | aws | ORACLE CLOUD INFRASTRUCTURE | Google Cloud

Each Cloud has its own Automation Tool

- Cloud Formation for AWS
- Azure Resource Manager for Azure
- Google Cloud Resource Manager
- Resource Manager for Oracle (OCI)

Terraform: Problem it Solves

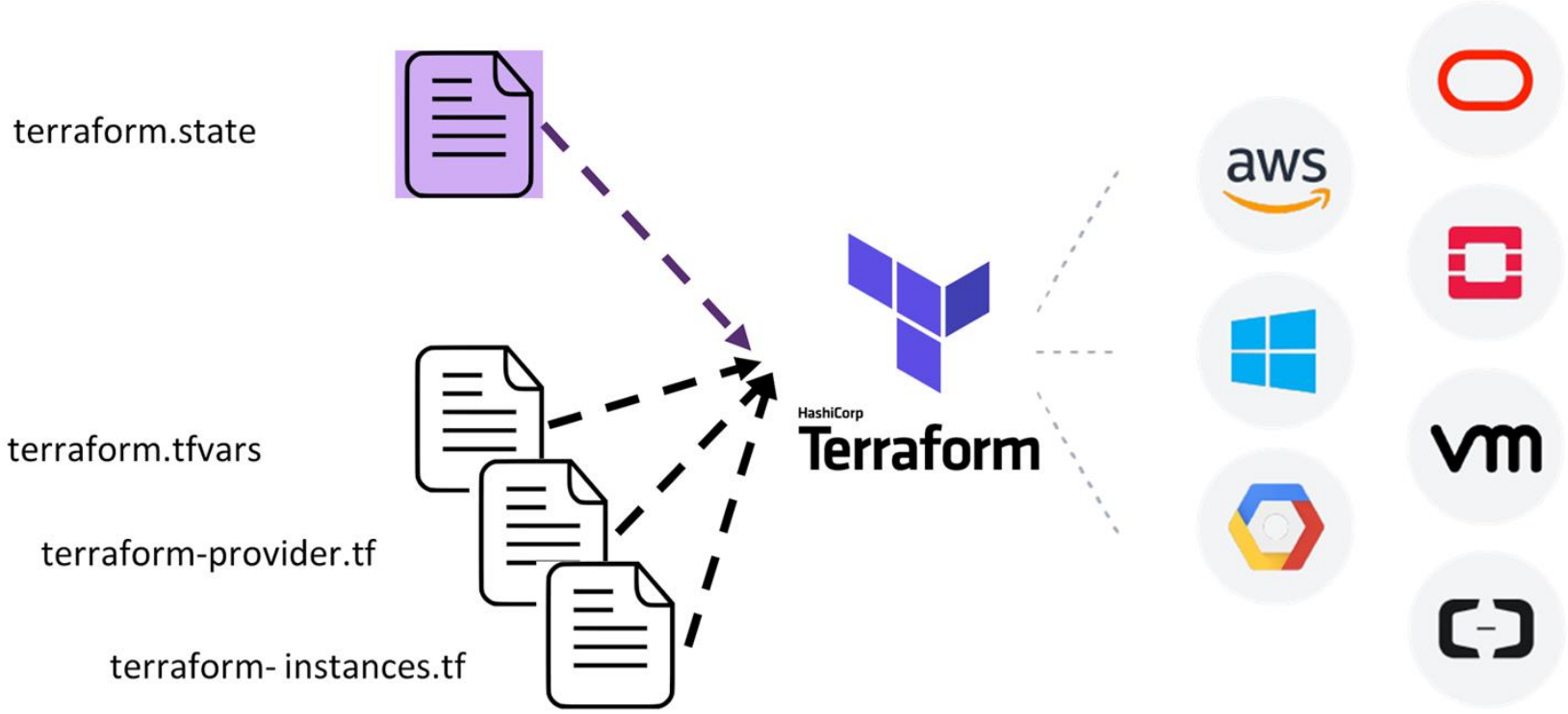
- Time To Build
- Human Error
- Tracking Changes
- Replicating Environment
- Rebuilding Environment
- Managing Multiple Resources

Terraform : Overview

K21Academy
Learn Cloud from Experts

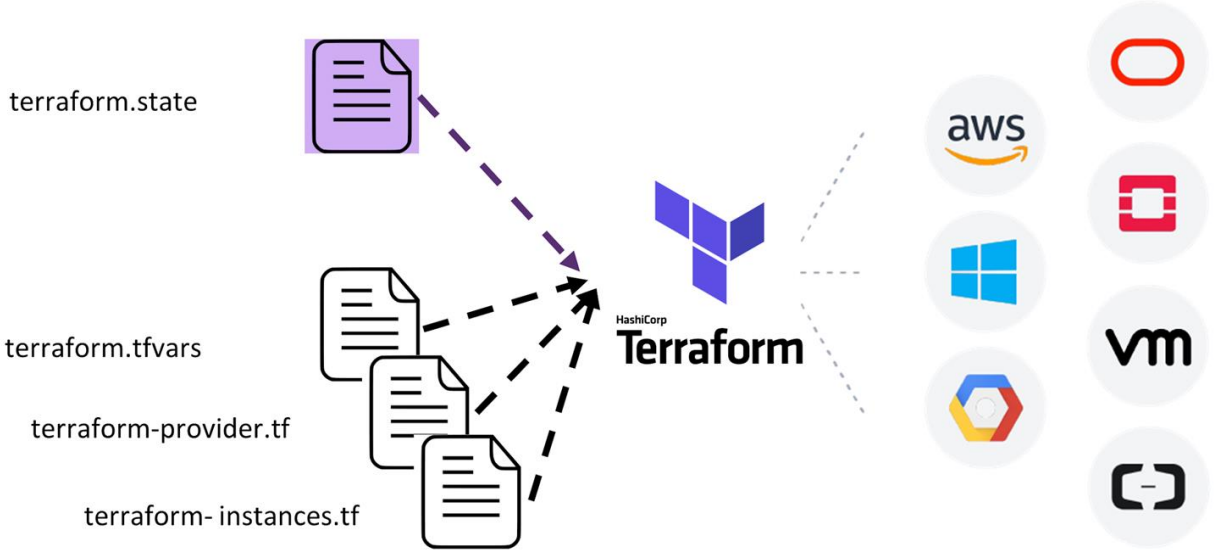
- Cloud Provider
- Resource
- Configuration Files
- Variables
- State Files
- Data Source
- Modules
- Workspaces
- Provisioners
- Security
- Terraform Enterprise
- Best Practices

Terraform



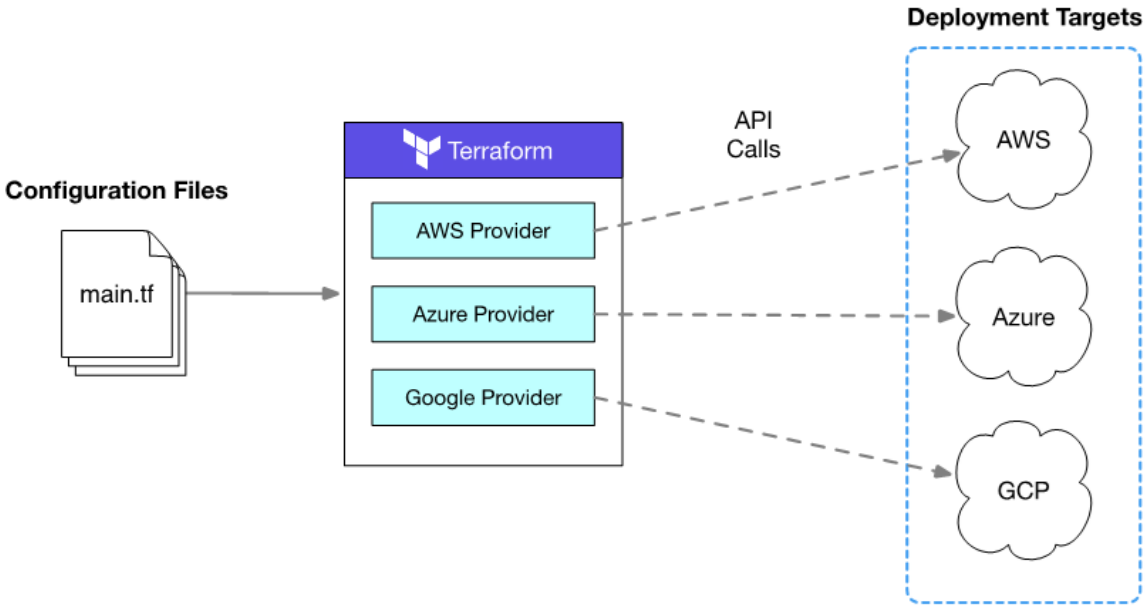
Terraform at a Glance

- Open Source tool written in GO
- For Building, Changing & Versioning Infrastructure
- Configuration files HCL or JSON format
- Ability to manage standard Cloud vendor or custom in-house solutions
- Configuration file (.tf) used to define required resources.
- Terraform generated execution plan to reach th desired state.
- Responsible for the creation of server and associated services
- Manages Low-Level & High Level Components.

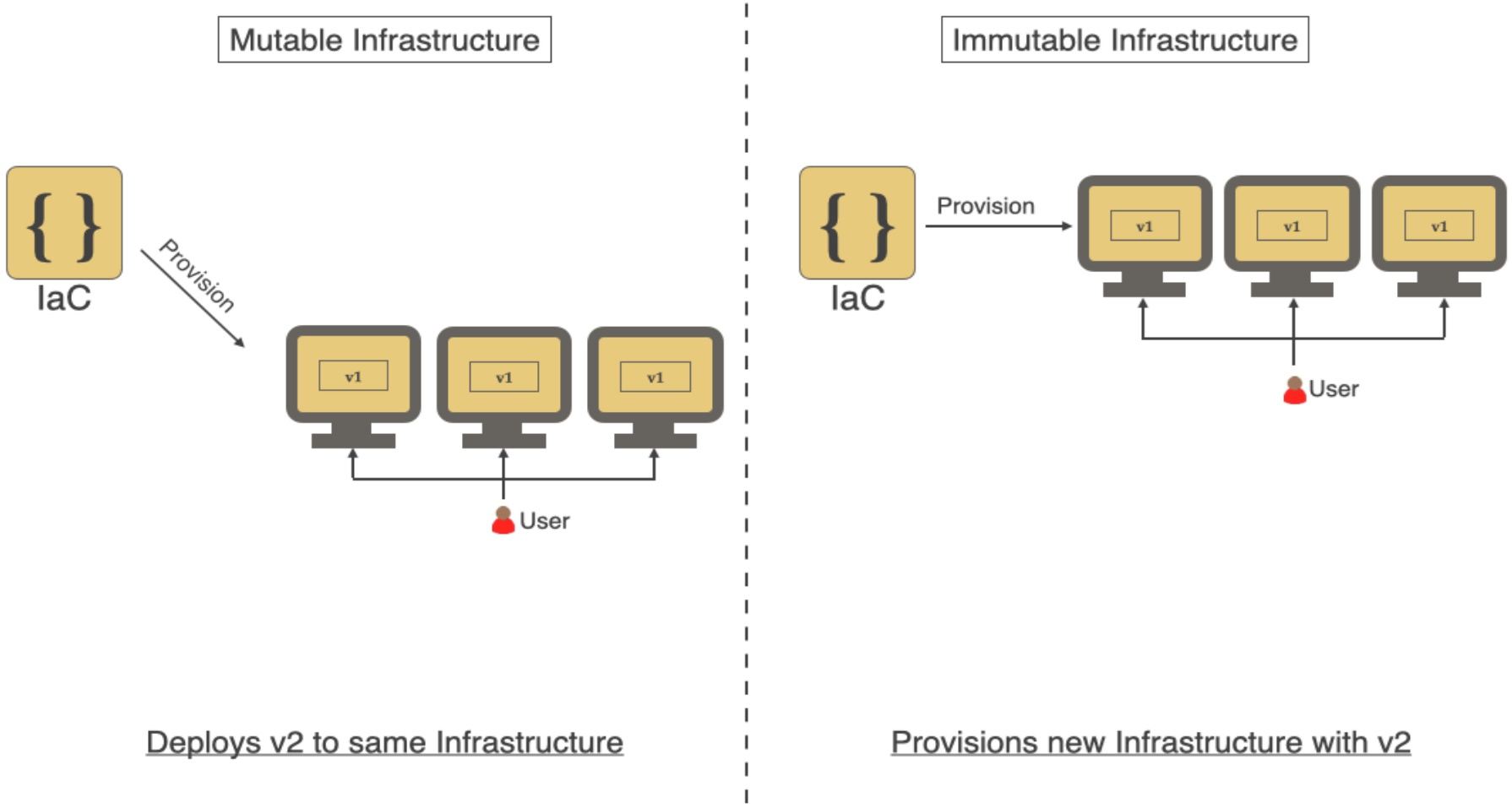


Terraform for Multi-Cloud

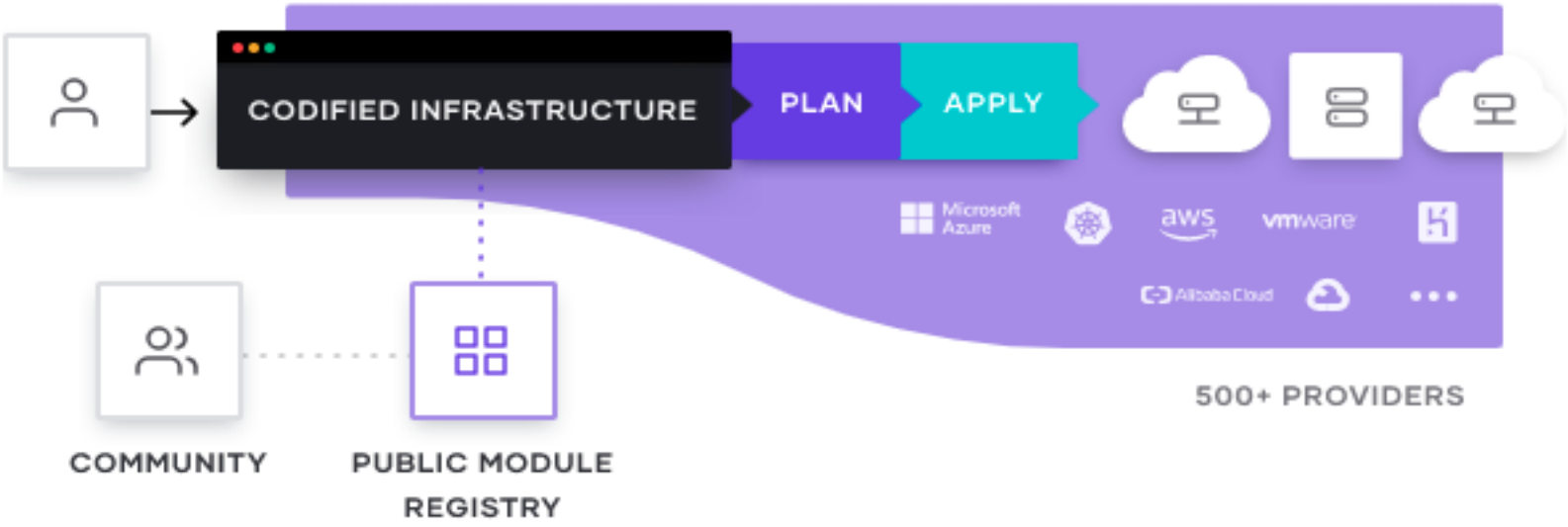
- Cloud agonist.
- Multi-cloud Support – Almost all cloud providers
- Other solutions including VMware, Kubernetes, and MySQL
- Provide a common tool, process, and language (HashiCorp Configuration Language) to be used across multiple clouds and services



Terraform: Mutable & Unmutable



Terraform: How does it works?



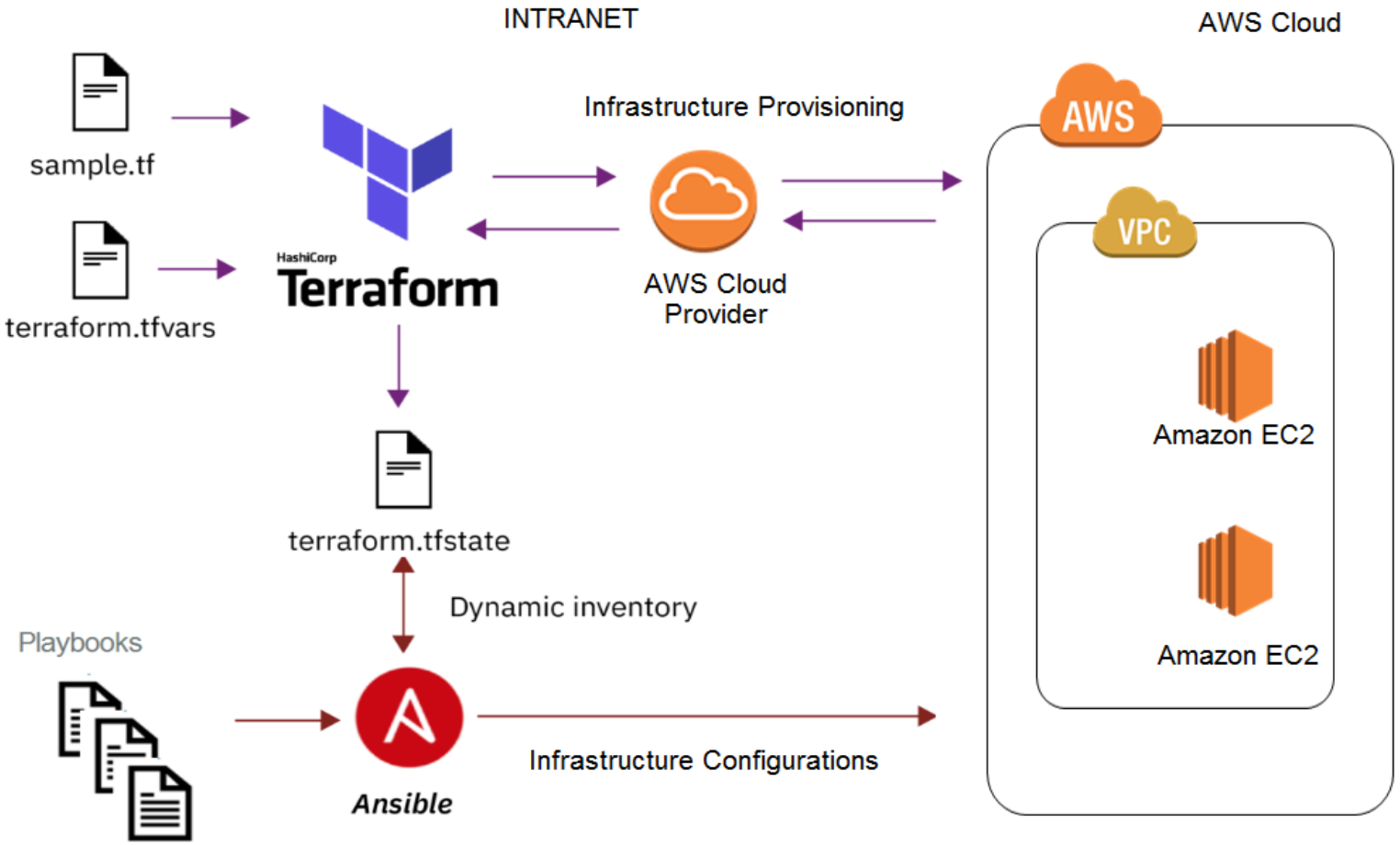
Define resources -> Execute Plan -> Apply to provision infrastructure

Provision + Configuration

Terraform

&

Ansible



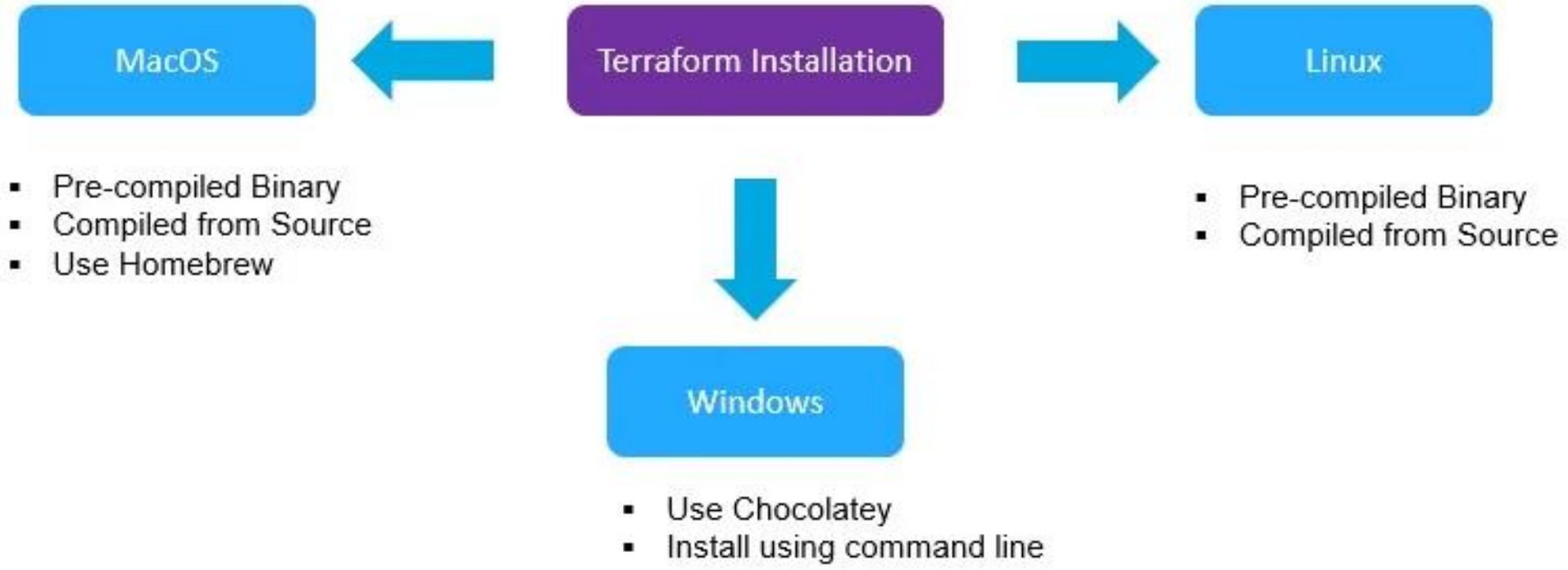
K21Academy
Learn Cloud from Experts

Terraform Installation

Linux | MacOS | Windows

Terraform Installation

- Written in Golang
- Compiled as a single binary for multiple operating systems
- Supported on Windows, macOS, and Linux, FreeBSD, OpenBSD, and Solaris.



Terraform Installation : Linux

K21Academy
Learn Cloud from Experts

Terraform Binary

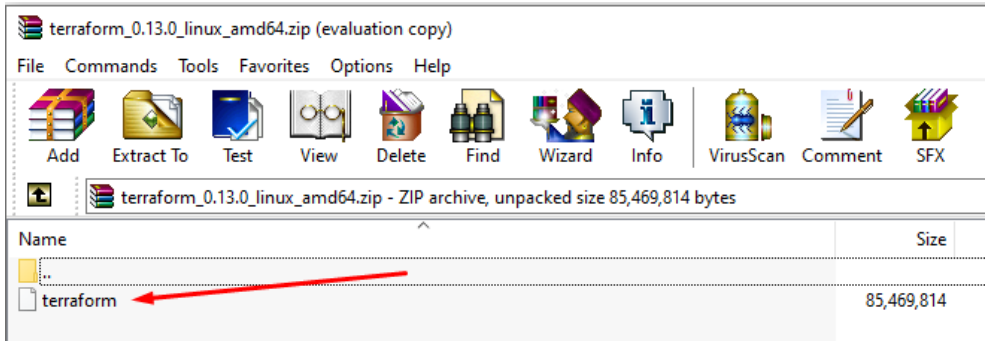
Pre-compiled Binary

Compiled at Source

https://www.terraform.io/downloads.html

Linux

32-bit | 64-bit | Arm




```
$ unzip terraform_0.12.3_linux_amd64.zip
$ sudo mv terraform /usr/local/bin/
$ sudo chown -R root:root /usr/local/bin/terraform
```

Terraform Installation : Mac-OS

K21Academy
Learn Cloud from Experts

OPTIONAL

Homebrew - open-source package management system

Install Homebrew

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

Paste that in a macOS Terminal or Linux shell prompt.

The script explains what it will do and then pauses before it does it. Read about other installation options.

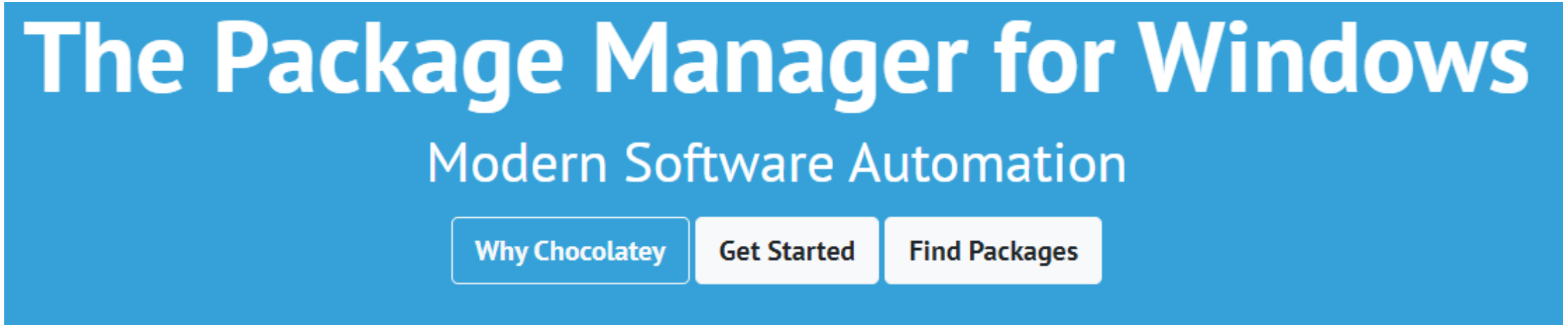
```
$ brew install terraform
```

Terraform Installation : Windows

K21Academy
Learn Cloud from Experts

OPTIONAL

Chocolatey - open-source package management system




```
PS C:\WINDOWS\system32> choco install terraform
Chocolatey v0.10.15
Installing the following packages:
terraform
By installing you accept licenses for the packages.
Progress: Downloading terraform 0.12.28... 100%

terraform v0.12.28 [Approved]
```

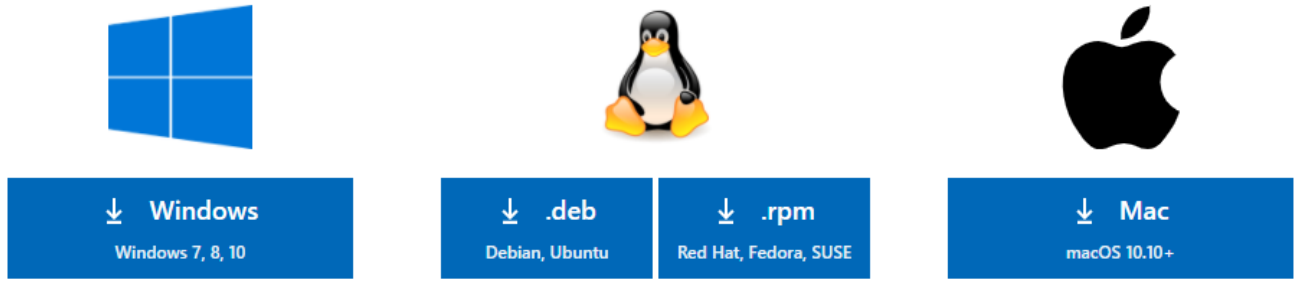
Terraform Coding : VS Code

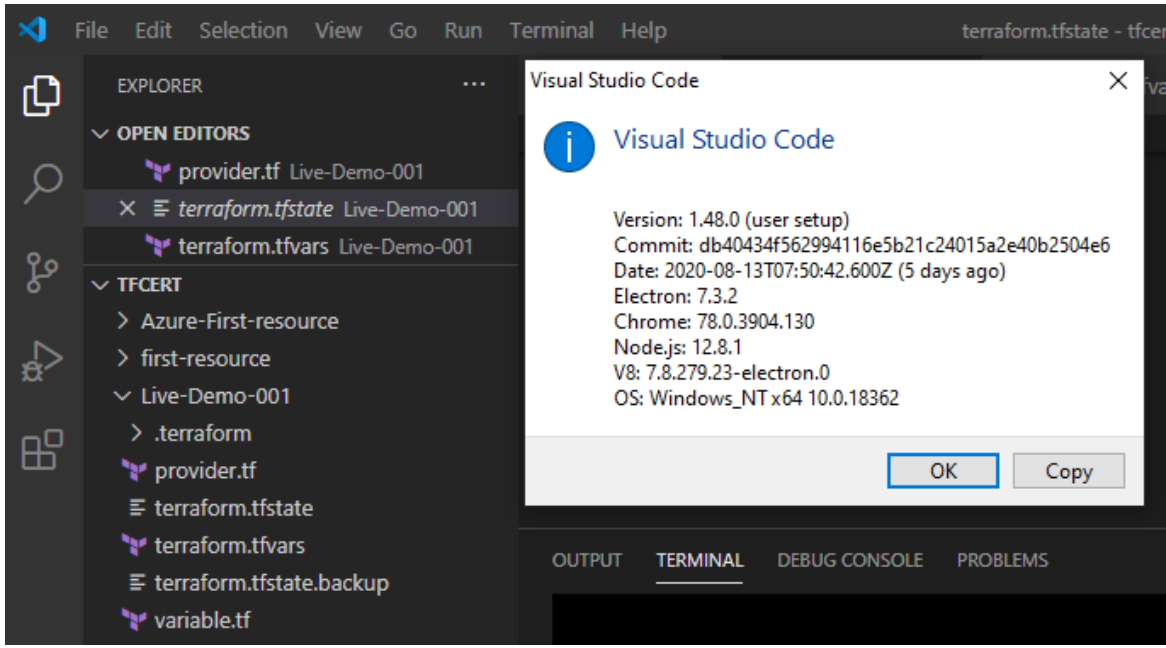
K21Academy
Learn Cloud from Experts

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

https://code.visualstudio.com/download

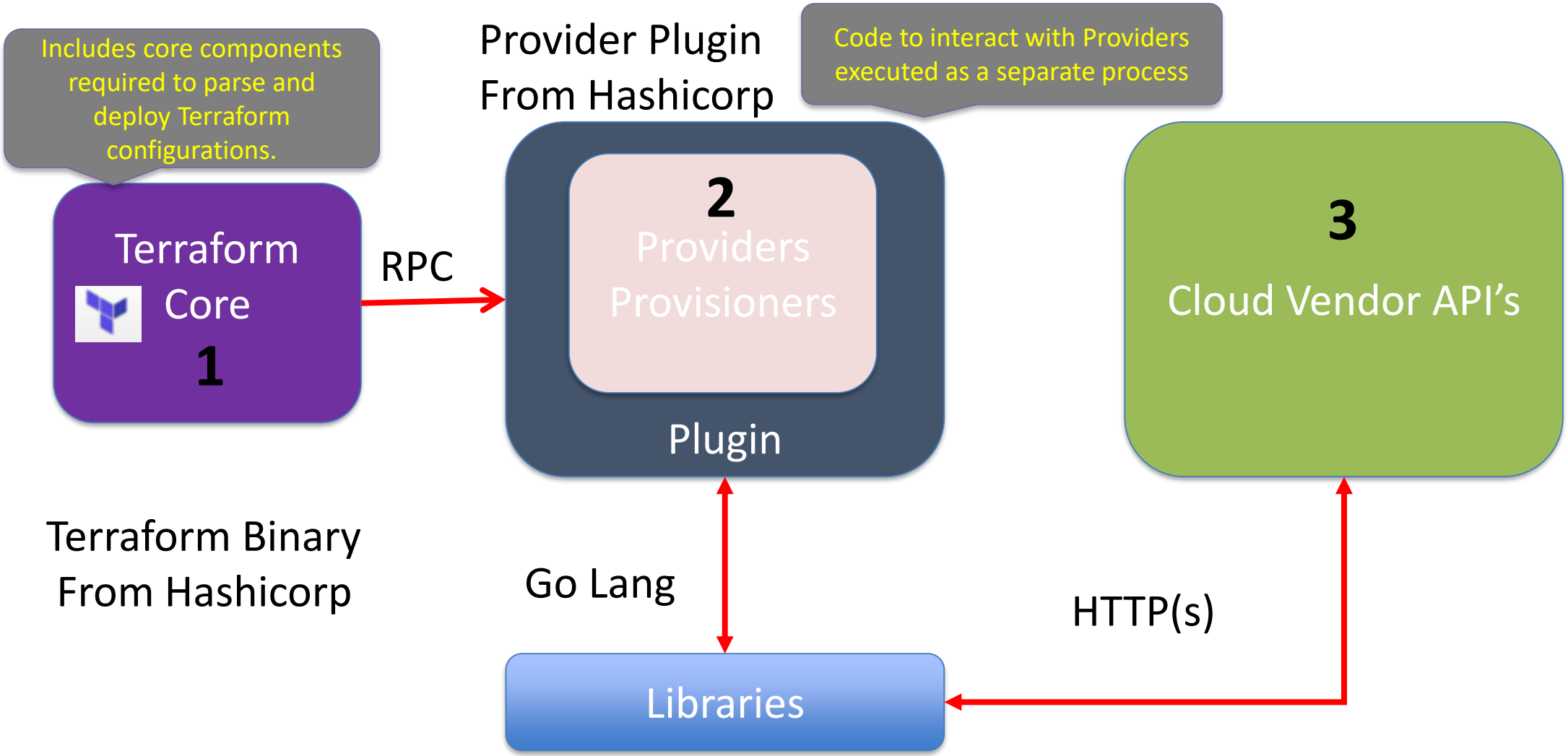




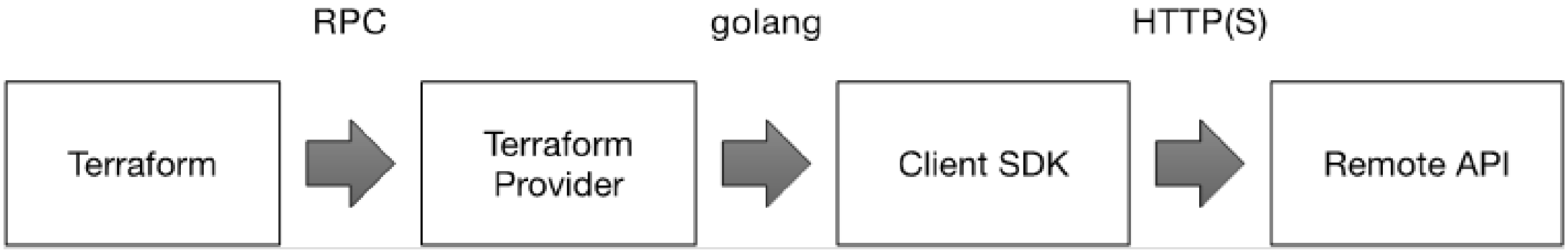
K21Academy
Learn Cloud from Experts

Terraform
Plugin Based Architecture

Plugin Based Architecture



Communication Flow



- Providers are plugins for Terraform which communicate with Terraform over RPC
- Providers are almost always written in Go
- Terraform plugin SDK library is written exclusively in Go.

Plugins

K21Academy
Learn Cloud from Experts

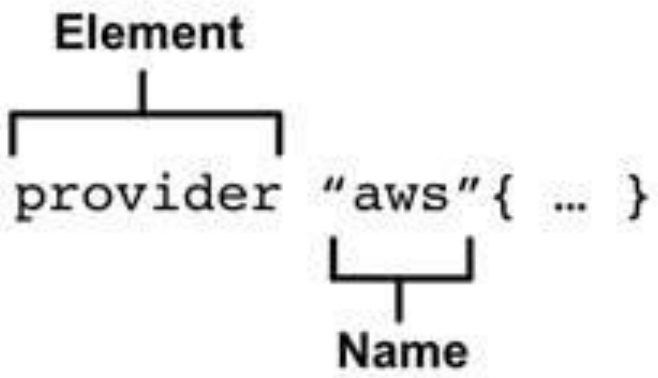
- Terraform plugins are written in the Go programming language and are executable binaries that get invoked by Terraform Core via RPCs

Plugins

- Provider plugins Distributed by Hashicorp
- Automatically installed by terraform init
  Providers and provisioners are provided via plugins
- Each plugin provides an implementation for a specific service
- Executed as a separate process
- Communicate with the main Terraform binary over an RPC interface.
  Plugins are built using dynamic libraries
- Each plugin is an independent program
- Main process communicates with the plugin process over HTTP.

Providers

- Enables resources and data sources used by Terraform
- Executable plug-in that contains code necessary to interact with the API of the service
- Includes
  - way to authenticate to a service
  - manage resources
  - access data sources
- Providers only have one label: name
- Consume an external client library
- Each Provider requirement is different in terms of authentication.
- Belong in the root module of a Terraform configuration

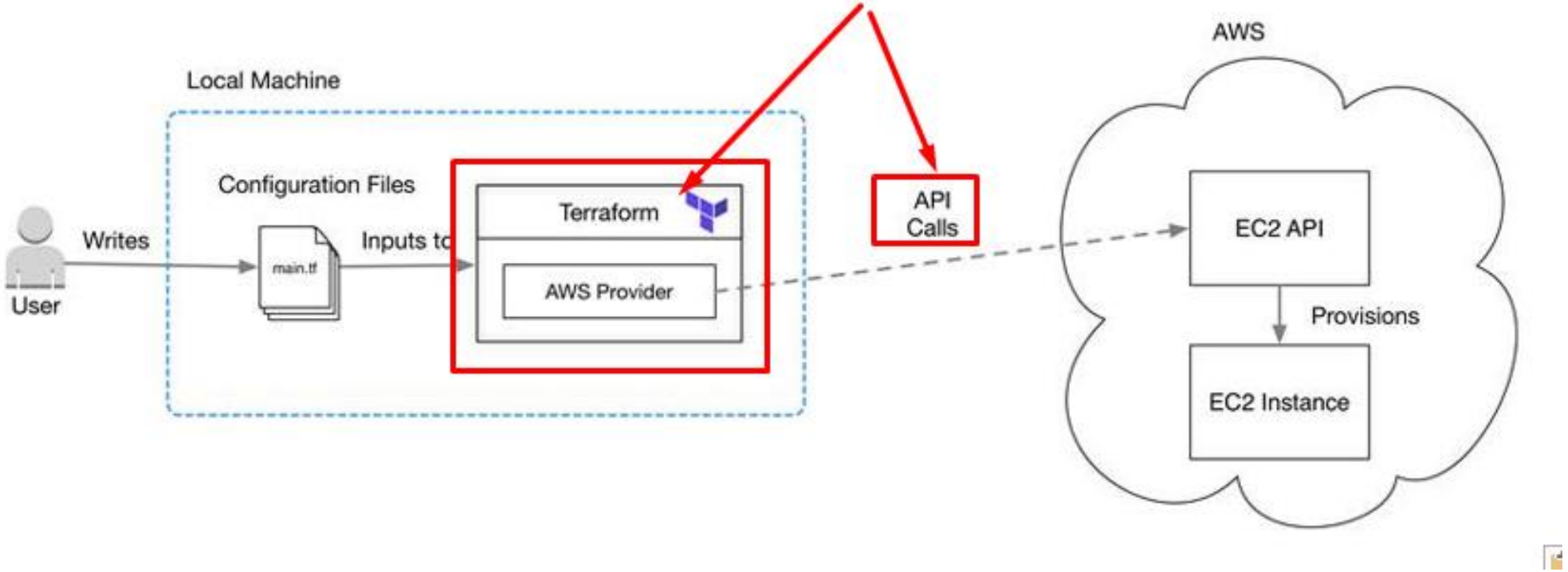


Providers

K21Academy
Learn Cloud from Experts

2000 + Cloud Providers
Some Major You can see Below

Example



Providers

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

Providers Version (Optional)

- Example Azure provider, the authentication information could be supplied through an environment variable or cached credentials from the Azure CLI
- **version** argument is used to constrain the provider to a specific version or a range of versions
- >= 1.41.0 is greater than or equal to the version.
- <= 1.41.0 is less than or equal to the version.
- ~> 1.41.0 this one is funky. It means any version in the 1.41.X range.
- >= 1.20, <= 1.41 is any version between 1.20 and 1.41 inclusive.

```
terraform {
  required_version = ">= 1.0"
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 1.0"
    }
  }
}

provider "aws" {
}
```

Authentication For Azure

Using the Azure CLI

Azure CLI current version

Download and install the current release of the Azure CLI.

Current release of the Azure CLI 1

Download CLI

Azure-Example01 > main.tf

```
1  provider "azurerm" {
2      features{}
3
4  }
```

Initiate Login

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
D:\tfcert\Module-demo>az login
```

2

```
D:\tfcert\Module-demo>az login
You have logged in. Now let us find all the subscriptions to which you have access...
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "335d1fbc-2124-4173-9863-[redacted]a0e",
    "id": "6ac97dc6-d184-[redacted]61-209c5e5289a3",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Free Trial",
    "state": "Enabled",
    "tenantId": "335d1fbc-[redacted]9863-17e7051a2a0e",
    "user": {
      "name": "AP[redacted]",
      "type": "user"
    }
  }
]
```

4

All Set to Go

Microsoft Azure

Microsoft

Pick an account

Amit Pancholia (CW)
Signed in

Use another account

3

Back

Authenticate

K21Academy
Learn Cloud from Experts

Terraform

How it fetch Providers ?

Terraform fetching Providers

K21Academy
Learn Cloud from Experts

Terraform Providers

HashiCorp distributed

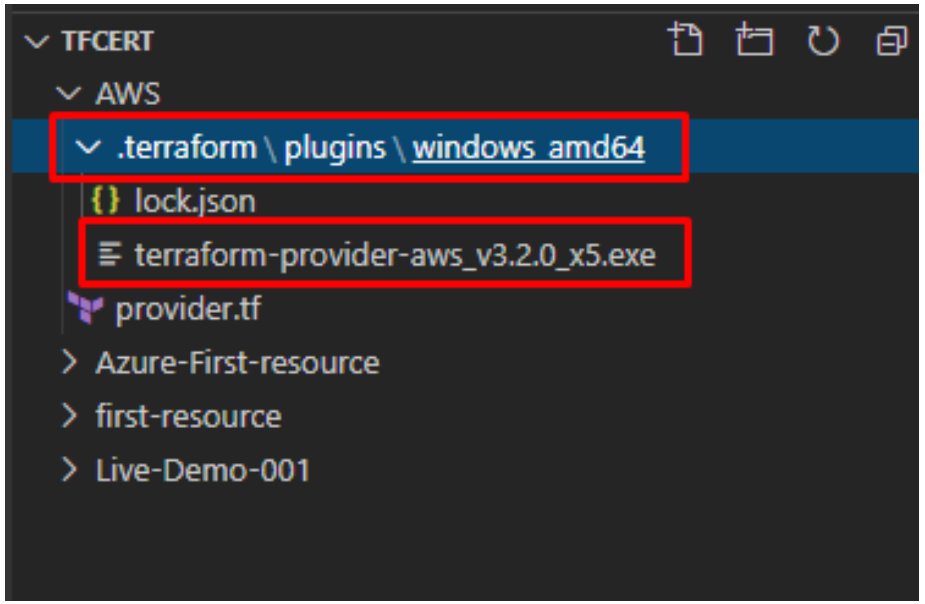
Providers are available for download automatically during Terraform initialization

Third-Party

Third party providers must be placed in a local plug-ins directory located
Windows : %APPDATA%\terraform.d\plugins
Other: ~/.terraform.d/plugins

Terraform fetching Providers

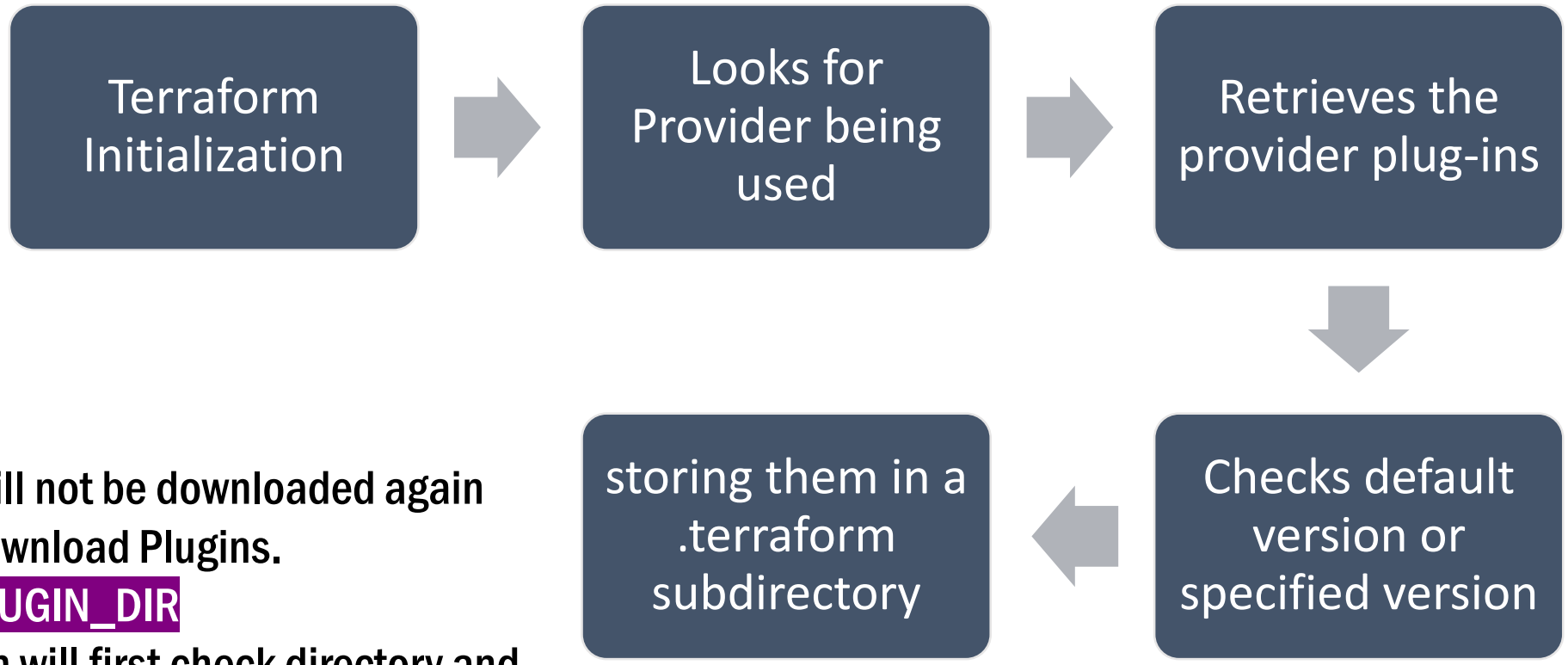
```
OUTPUT    TERMINAL    DEBUG CONSOLE    PROBLEMS

(c) 2019 Microsoft Corporation. All rights reserved.

D:\tfcert>cd AWS

D:\tfcert\AWS>terraform init

Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "aws" (hashicorp/aws) 3.2.0...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.aws: version = "~> 3.2"
```



Terraform fetching Providers

.terraform/plugins/<OS>_<ARCH>	Automatically downloaded providers
~/.terraform.d/plugins or %APPDATA% \terraform.d\plugins	The user plugins directory
~/.terraform.d/plugins/<OS>_<ARCH> or %APPDATA% \terraform.d\plugins\<OS>_<ARCH>	The user plugins directory, with explicit OS and architecture

Terraform fetching Providers



- Enable Caching so Plugin will not be downloaded again
- Internet not necessary to download Plugins.
- Environment variable TF_PLUGIN_DIR
- Plugin Dir set then Terraform will first check directory and not download.

K21Academy
Learn Cloud from Experts

Terraform
Multiple Providers

Multiple Providers

Scenario could be Using multiple different providers in a configuration
Scenario could be using multiple instances of the same provider

- Optionally define multiple configurations for the same provider
- Select which one to use on a per-resource or per-module basis.
- To support multiple regions for a cloud platform

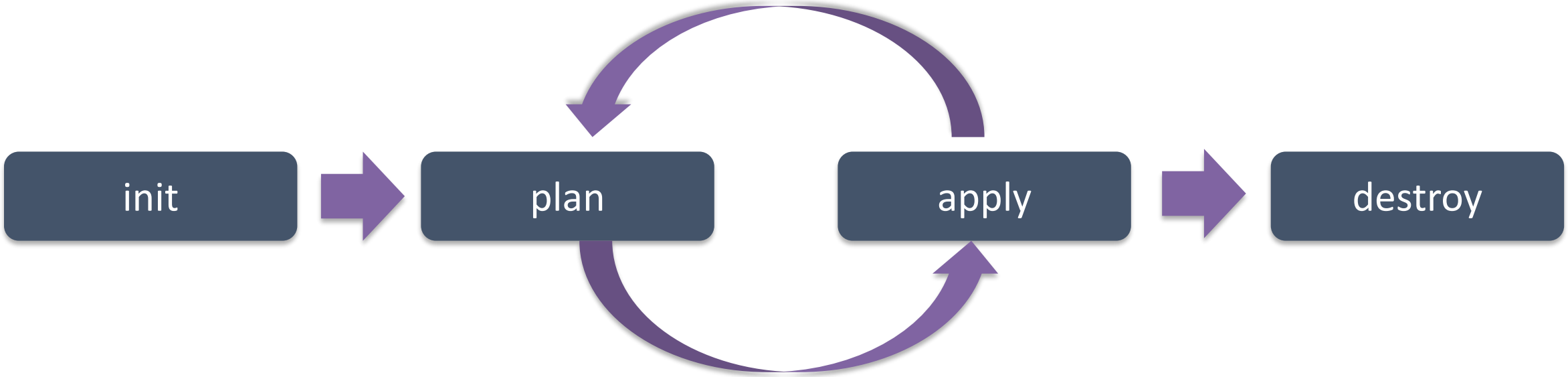
Each additional non-default configuration use alias meta-argument

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

# Additional provider configuration for west coast region; resources can
# reference this as `aws.west`.
provider "aws" {
  alias  = "west"
  region = "us-west-2"
}
```

K21Academy
Learn Cloud from Experts

Terraform
Execution Workflow

Terraform Flow



Init Initialize the (local) Terraform environment. Usually executed only once per session.
Plan. Compare the Terraform state with the as-is state in the cloud, build and display an execution plan. This does not change the deployment (read-only).
Apply the plan from the plan phase. This potentially changes the deployment (read and write
Destroy all resources that are governed by this specific terraform environment.

Lets get started !!!

K21Academy
Learn Cloud from Experts

- Install terraform
- Decide a providers name which you want to work – AWS, Azure, OCI, GCP
- Decide a provider's resources and Argument Reference
- Initialize terraform providers
- Write terraform providers resources and Arguments in .tf file
- PLAN (DRY RUN) using terraform plan
- APPLY (Create a Resources) using terraform apply
- APPLY (Update a Resources) using terraform apply
- DESTROY (DELETE a Resources) using terraform destroy

What you Need to Start

K21Academy
Learn Cloud from Experts

- Cloud Account - AWS or Azure or OCI or provider you are working with
- Visual Studio Code
- Terraform Executable
- Authentication setup via Terraform

You are ready to Terraform Now !!!!!!!

Find Us

K21Academy
Learn Cloud from Experts

https://www.facebook.com/K21Academy

http://twitter.com/k21Academy

https://www.linkedin.com/company/k21academy

https://www.youtube.com/k21academy

https://www.instagram.com/k21academy/