

# Terraform Course

# Module 01

## Introduction

# Course Contents (1)

- 01 - Introduction
- 02 - Getting Started
- 03 – Deploying Infrastructure with Terraform
- 04 - Read, Generate, Modify Configurations
- 05 – Terraform Provisioners
- 06 - Modules

# Course Contents (2)

- 07 – Remote State Management
- 08 – Security Primer
- 09 – Terraform Cloud and Enterprise Capabilities
- 10 - Troubleshooting Terraform

# Terraform

## Overview and Architecture

# Infrastructure as Code (IaC)

- Infrastructure as Code (IaC) is the process of managing and provisioning cloud infrastructure with machine-readable definition files.
- Think of it as executable documentation.
- Benefits of Infrastructure as Code
  - Provide a codified **workflow** to create infrastructure
  - **Change and update** existing infrastructure
  - Safely **test changes** using some form of dry run mode
  - **Integrate** with **application code workflows** (Git, CI/CD tools)
  - Provide **reusable configuration** for easy sharing and collaboration
  - **Enforce** security **policy** and organizational standards
  - Enable **collaboration** between different teams

The benefits of code:

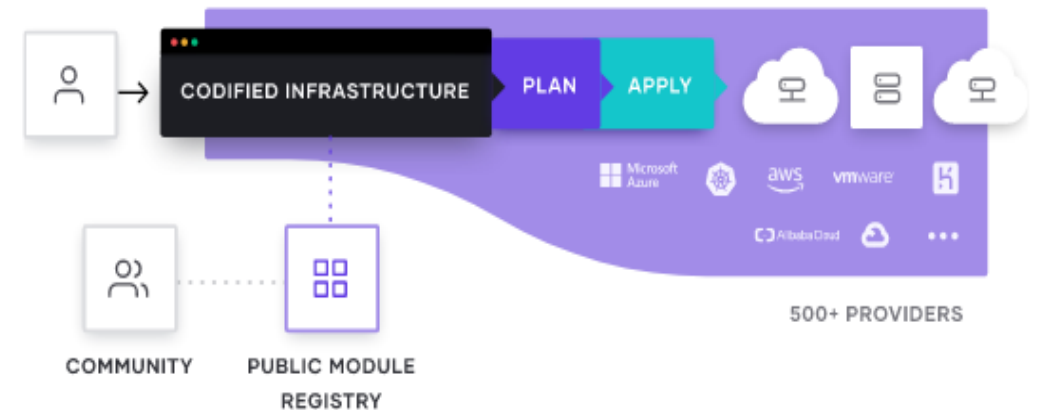
1. Automation
2. Version control
3. Code review
4. Testing
5. Documentation
6. Reuse

# Cloud Provider Native IaC Tools/Services

- Main providers:
  - Azure – Azure Resource Manager ([ARM](#)) or [Bicep](#)
  - AWS – CloudFormation (YAML / JSON templates) and Cloud Development Kit ([CDK](#))
  - Google Cloud Platform (GCP) - [Cloud Deployment Manager](#)
  - OpenStack – [Heat](#) (Template format aligned with CloudFormation)
  - Alibaba ([Resource Orchestration Service](#))
- Typically based on YAML or JSON templates
- Pros:
  - Tighter integration with Cloud Provider (e.g. IAM permissions to create or modify CloudFormation stacks)
- Cons:
  - Platform specific, not suitable for multi-cloud (\*)

# What Is Terraform

- [Terraform](#) is an open-source [infrastructure as code](#) (IaC) software tool that provides a [consistent CLI workflow](#) to manage **hundreds of cloud services**.
- **Key concept:** Terraform [codifies cloud APIs into declarative configuration files](#).
- Terraform allows infrastructure to be expressed as code in a simple, human readable language called HCL (HashiCorp Configuration Language).
  - Terraform reads [configuration files](#) and provides an [execution plan](#) of changes, which can be reviewed for safety and then [applied and provisioned](#).
  - Applies Graph Theory to IaC
  - Extensible [providers](#) allow Terraform to manage a broad range of resources, including IaaS, PaaS, SaaS, and hardware services.
- [Open Source](#). Written in Golang.
  - GA 1.0 June 2021 (see : [v1.0 compatibility promises](#))
  - Current Release (March 2023): 1.3.9 / 1.4.0
  - Course examples: 1.4.0 (March 2023)

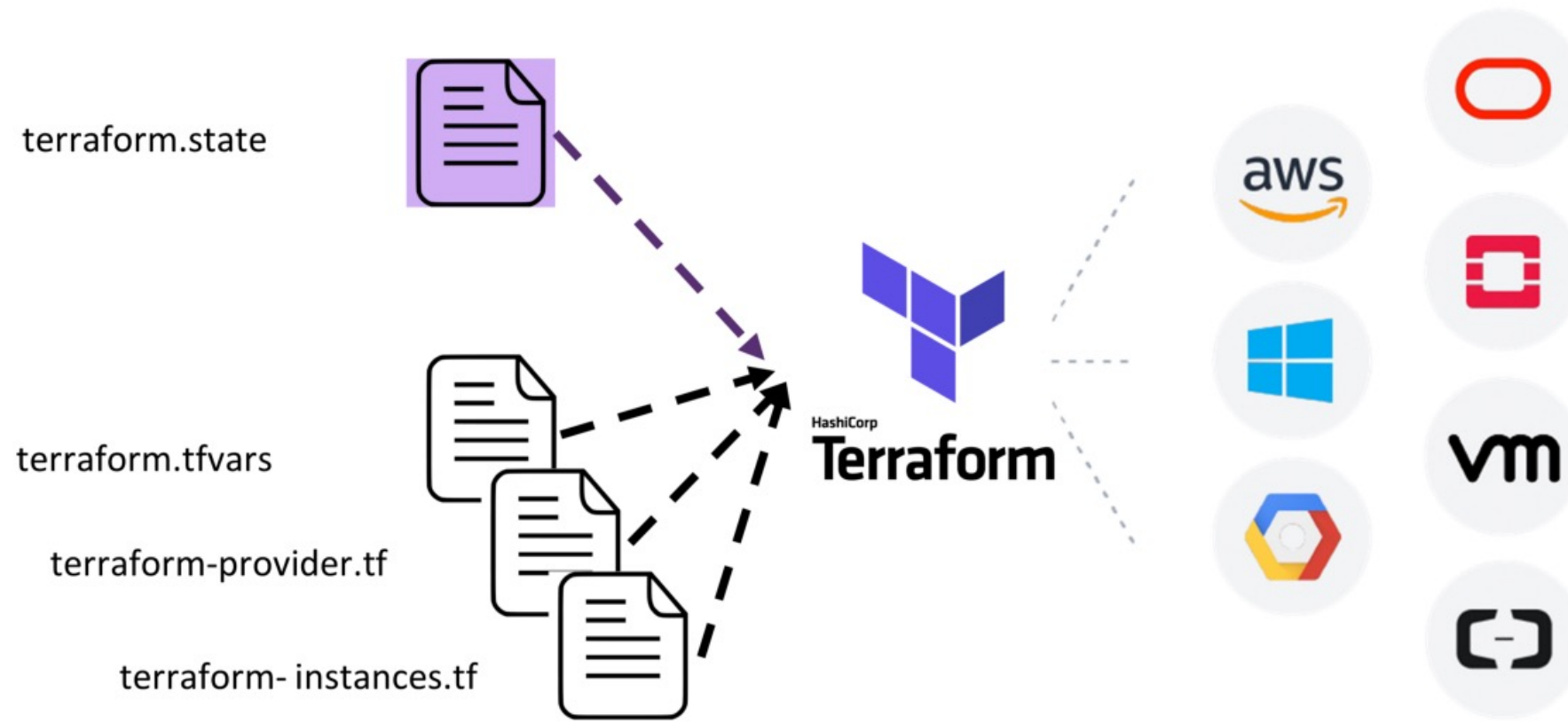




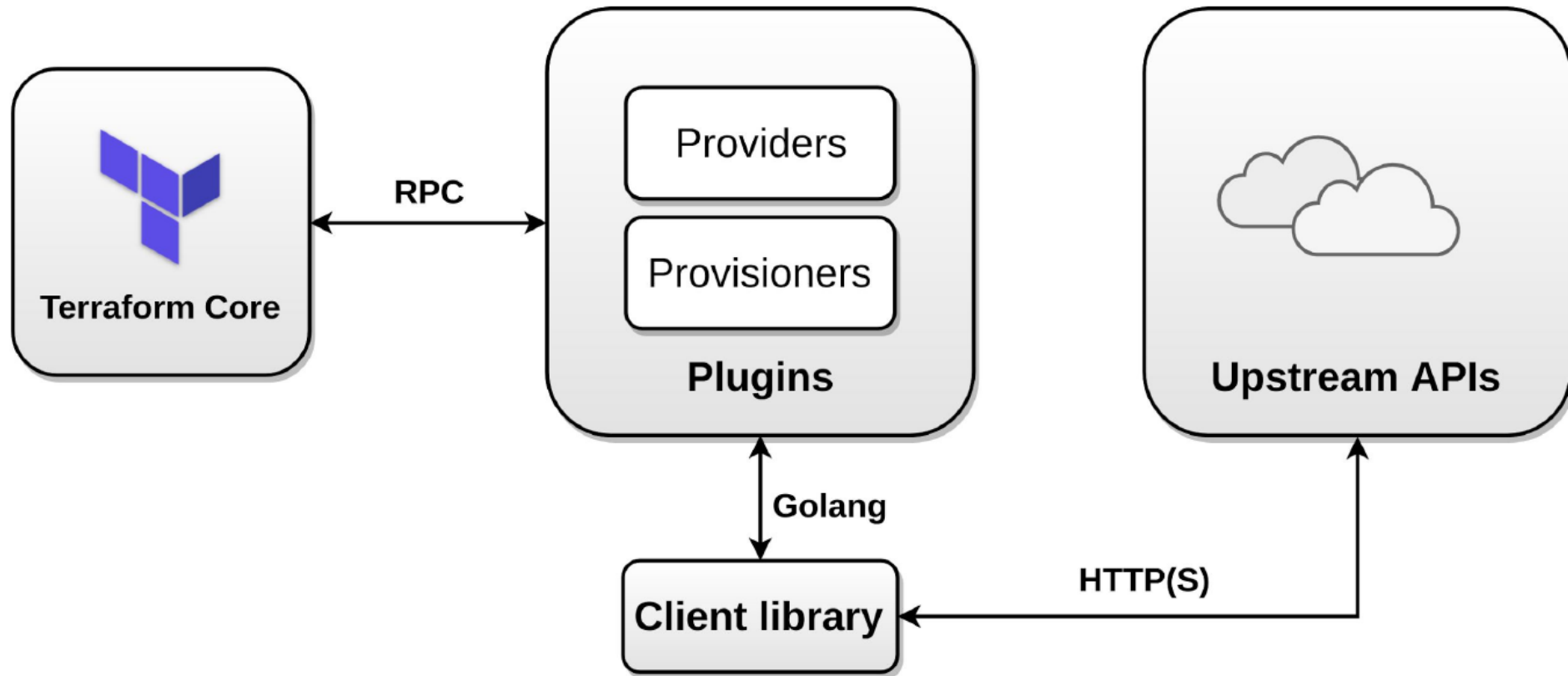
# Why Terraform

- Generic IaC benefits
  - Codify your application infrastructure
    - Reduce human error and increase automation by provisioning infrastructure as code.
  - Create reproducible infrastructure
    - Provision consistent testing, staging, and production environments with the same configuration.
- In addition, Terraform allows
  - Provision infrastructure across 300+ public clouds and services using a single workflow.
  - Unified view of infrastructure as code
  - Managing anything with an API / Technology-agnostic workflow
- **But...** Terraform is no magic bullet – and the benefits of multi-cloud are debatable
  - "Multi-cloud" per-se may not be the right reason to use Terraform

# Terraform Architecture



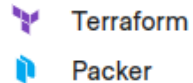
# Terraform Architecture



# HashiCorp – Makers of Terraform

- [Corporate Info](#) (founded 2012)
- [Financials](#) (public, Nasdaq, since Dec. 2021)
- Products

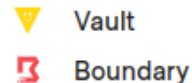
## INFRASTRUCTURE



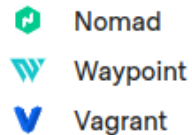
## NETWORKING



## SECURITY

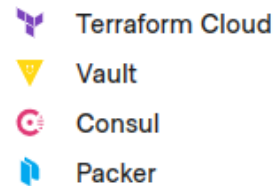


## APPLICATIONS



## HASHICORP CLOUD PLATFORM

A fully managed platform to automate infrastructure on any cloud with HashiCorp products.



Visit [cloud.hashicorp.com](https://cloud.hashicorp.com) →

# Terraform Workflow Overview

# Terraform Workflow Overview

- Core workflow
  - **Write** - Author infrastructure as code.
    - Create templates in HCL (HashiCorp Configuration Language) (xxx.tf files)
    - Templates incorporated into version control systems (VCS)
  - **Initialize** Terraform in directory where terraform templates live
    - Installs providers and modules
  - **Validate** configuration for syntax
  - **Plan** - Preview changes before applying.
    - Validate the templates (terraform validate) and other tools (e.g. tf-lint)
    - terraform plan -> verify what changes will be performed
  - **Apply** - Provision reproducible infrastructure.
    - terraform apply
    - Verify state changes
  - **Destroy**
- This course will constantly revolve around these steps and the underlying concept of "state"

# Terraform Workflow - Demo

- Simple AWS EC2 instance
- This is a simple overview – all these concepts will be covered in detail during the course

```
terraform {  
  required_providers {  
    aws = {  
      source  = "hashicorp/aws"  
      version = "~> 4.16"  
    }  
  }  
  required_version = ">= 1.3.0"  
}  
  
provider "aws" {  
  profile = "tf"  
  region  = "eu-west-1"  
}  
  
variable "my_ami" {  
  description = "ami for EC2 instance"  
  type        = string  
  default     = "ami-0b752bf1df193a6c4"  
}  
  
resource "aws_instance" "server1" {  
  ami           = var.my_ami  
  instance_type = "t2.micro"  
  
  tags = {  
    Name = "vm1"  
    project = "acme"  
  }  
}
```

# Terraform vs Other Tools



# Overview


- Terraform is generally considered an infrastructure "provisioning" tool (as are some cloud-specific services such as AWS CloudFormation, Azure Resource Manager / Bicep, OpenStack Heat ...)
- Tools like Chef, Ansible, Puppet, etc. are generally classified as "configuration" tools
- These two categories can and do overlap for specific products
  - Ansible, Chef, Puppet can be used to provision infrastructure
  - Terraform can be used to configure servers, deploy software, etc.
    - See for instance [Terraform "provisioners"](#) as an example of using Terraform to configure infrastructure after provisioning it.
    - **IMPORTANT:** Terraform's warns that "Provisioners are a last resort" : "The following sections describe some situations which can be solved with provisioners in principle, but where better solutions are also available. We do not recommend using provisioners for any of the use-cases described in the following sections."

# Tool Comparison

- Discussion in [Terragrunt](#) web page
  - Configuration Management vs Provisioning
  - Mutable Infrastructure vs Immutable Infrastructure
  - Procedural vs Declarative
  - Master vs Masterless
  - Agent vs Agentless
  - Large Community vs Small Community
  - Mature vs Cutting Edge
  - Using Multiple Tools Together
- See also *Terraform: Up and Running 3rd Edition* (Sept. 2022) - Preview available at [O'Reilly](#)

	Source	Cloud	Type	Infrastructure	Language	Agent	Master	Community	Maturity
Chef	Open	All	Config Mgmt	Mutable	Procedural	Yes	Yes	Large	High
Puppet	Open	All	Config Mgmt	Mutable	Declarative	Yes	Yes	Large	High
Ansible	Open	All	Config Mgmt	Mutable	Procedural	No	No	Huge	Medium
SaltStack	Open	All	Config Mgmt	Mutable	Declarative	Yes	Yes	Large	Medium
CloudFormation	Closed	AWS	Provisioning	Immutable	Declarative	No	No	Small	Medium
Heat	Open	All	Provisioning	Immutable	Declarative	No	No	Small	Low
Terraform	Open	All	Provisioning	Immutable	Declarative	No	No	Huge	Low

# Terraform vs AWS Native IaC Tool (CloudFormation)

	 <b>Terraform</b>	 <b>AWS CloudFormation</b>
Code Syntax	HCL	YAML/JSON
Dynamic Features	✓	Limited
Usable Functions	Extensive	Limited
State Management	Required	No State Files to Maintain
Cloud Providers	Multi-Provider Support	Only Supports AWS Resources
Enterprise Support	Opensource	Native AWS Service
Automating IaC	Complex to Automate without Tooling	Complex to Automate without Tooling

spacelift.io

- Comparison table from SpaceLight (IaC automation vendor - opinion may be biased :-) )