What is Terraform?

- Terraform is a tool for **building, changing, and versioning infrastructure** safely and efficiently. it is developed by Hashicorp & written in GoLang

- Terraform is also known as **infrastructure as code** solution

Infrastructure as code?

- in simple words provisioning / creating infrastructure with code

- **Infrastructure as code**, or **IaC**, has gained a lot of momentum over the years because it helps solve several problems that troubled infrastructure management in the past.

Infrastructure as code (IaC) Enabels

- `Reproducible Environments:`
  - By using code to generate infrastructure, the same environment can be created over and over.
  - Over a perdiod of time an environment can drift away from its desired state and difficult to diagnose issues can creep into your release pipeline.
  - With IaC no environment gets special treatment and fresh new environments are easily created and destroyed.
- `Idempotence & Convergence:`
  - Extending the last point, idempotence is the trait that no matter you apply the configuration described by your IaC, there are no side effects on the environment.
  - Convergence is the trait that actions are only taken if they need to be.
  - In IaC, only the actions needed to bring the environment to the desired state are executed. If the environment is already in the desired state, no actions are taken.

Infrastructure as code (IaC) Enabels

- `Easing collaboration:`
  - Having the code in a version control system like Git allows teams to collaborate on infrastructure.
  - Team members can get specific versions of the code and create their own environments for testing or other scenarios.
- `Self-service infrastructure:`
  - A pain point that often existed for developers before moving to cloud infrastructure was the delays required to have operations teams create the infrastructure they needed to build new features and tools.
  - With the elasticity of the cloud allowing resources to be created on-demand, developers can provision the infrastructure they need when they need it.
  - IaC further improves the situation by allowing developers to use infrastructure modules to create identical environments at any point in the application development lifecycle.
  - The infrastructure modules could be created by operations and shared with developers freeing developers from having to learn another skill.

More IaC Tools

- Every Cloud providers have their own native infrastructure as code solution.
- Each comes with their own unique way of describing infrastructure in code.

Google Cloud
Deployment Manager

Terraform Supports

- one workflow for all types of clouds.
- the hosted infrastructure on all public clouds like
  Amazon Web Services / Azure / Google Cloud Platform
- on-prem in private clouds such as
  OpenStack / VMWare vSphere / Cloud Stack.
- over 100 publicly available infrastructure integrations.
- managing software and services, including databases like MySQL, source control systems like GitHub, configuration management tools like Chef etc...

When should we use Terraform?

- With the trend in industry tending to move toward multi-cloud platform to host, including hybrid clouds, it is more efficient to have one tool and one common workflow to manage infrastructure no matter where it is hosted.

- Even if you are only using one cloud now, it may be worth futureproofing yourself in case you do leverage multiple clouds later on.

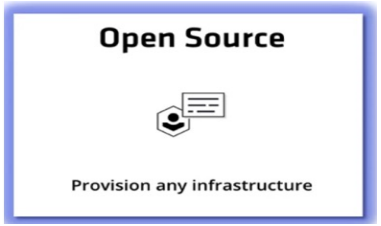
- When you have spread your infrastructure in multi-cloud,

- multi-cloud deployments can be very challenging as many existing tools for infrastructure management are cloud-specific. Terraform is cloud-agnostic and allows a single configuration to be used to manage multiple providers, and to even handle cross-cloud dependencies.

- This simplifies management and orchestration, helping operators build large-scale multi-cloud infrastructures.

- you can find these use cases and more on Terraforms website https://www.terraform.io/intro/use-cases.html

Terraform Product Streams



The open source Terraform product compiles to a tool that you interact with on the command-line, the command-line tool uses configuration files that specify your infrastructure as code.

Pro is software as a service that runs in the cloud and is managed by HashiCorp. The pro tier includes a graphical user interface, version control connections to modify infrastructure as changes are committed, API access to integrate terraform into existing tooling, and a private infrastructure module registry for sharing re-useable modules within your organization. You also receive business hour support from HashiCorp with the Pro tier.

Premium is installed on your own cloud infrastructure, that currently must be in AWS. It includes the features of Pro as well as policy as code, single sign on with SAML, and provides audit logs of every infrastructure change. You also receive 24x7 support gold-level support from HashiCorp.

Install Terraform CLI

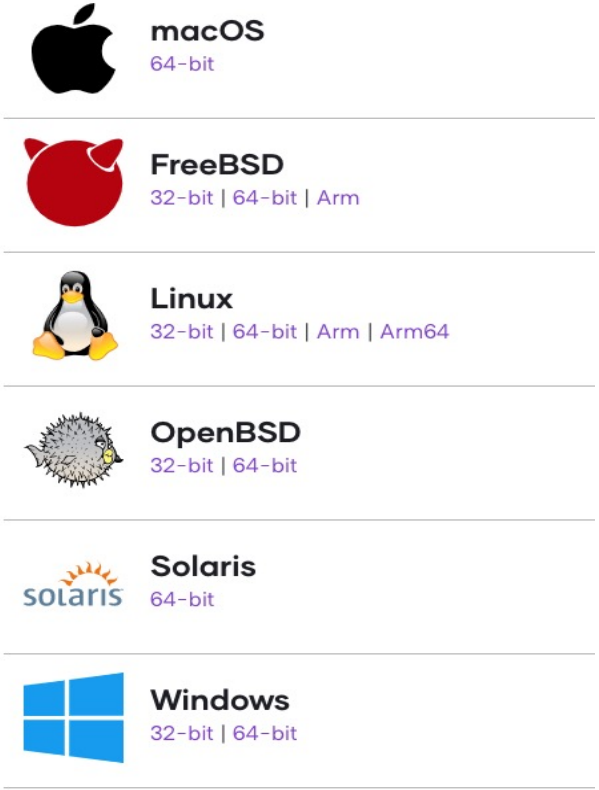
https://www.terraform.io/downloads.html

Terraform is distributed as a single binary. Install Terraform by unzipping it and moving it to a directory included in your system's PATH

You can also install Terraform CLI with system package managers such as

On Debian/Uubuntu APT Packages
https://www.terraform.io/docs/cli/install/apt.html

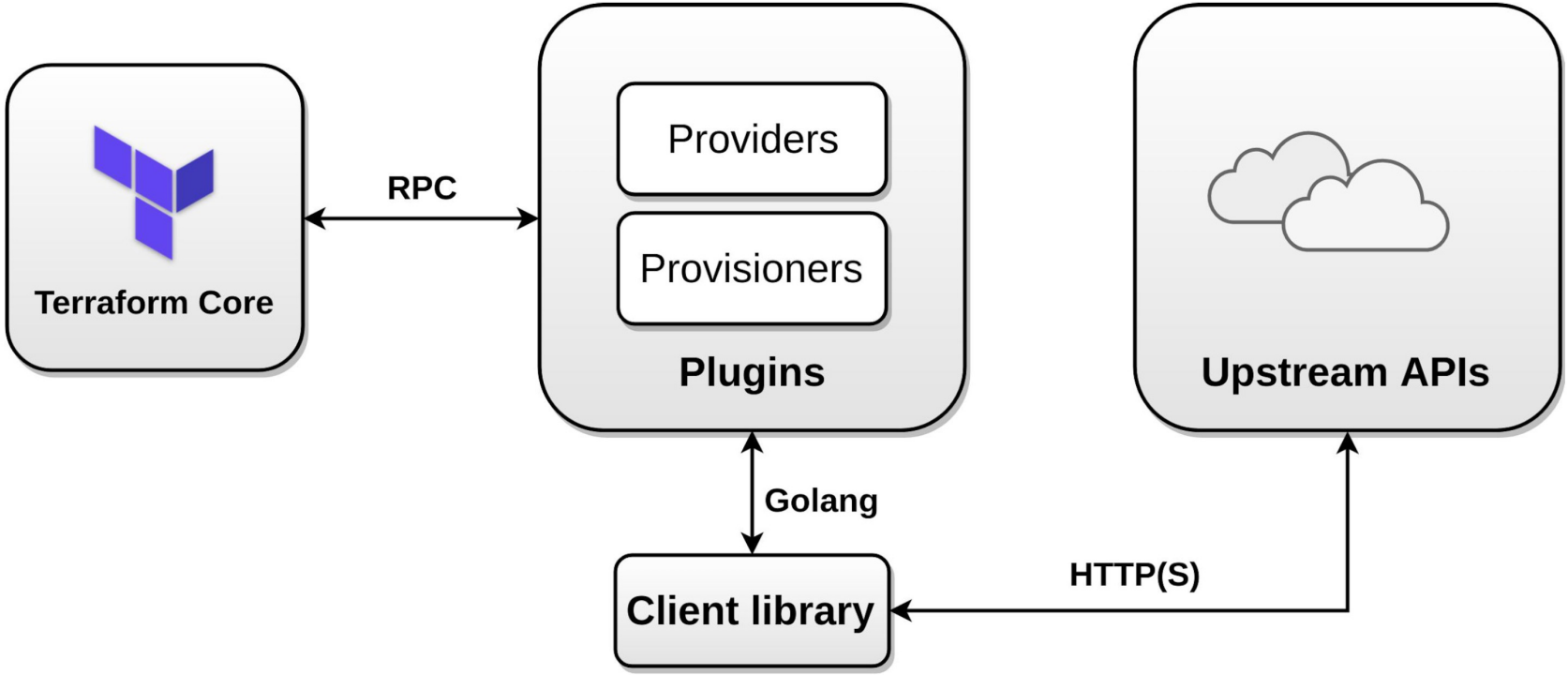
On RHEL/Fedora Yum Packages
https://www.terraform.io/docs/cli/install/yum.html



Install Terraform CLI on Linux

- Below would install the terraform Version 1.0.1
- cd /tmp ; wget https://releases.hashicorp.com/terraform/1.0.1/terraform_1.0.1_linux_amd64.zip
- apt update ; apt install –y unzip ➔ on Ubuntu / Debian Systems
- yum update ; yum install –y unzip ➔ on RHEL/Centos/Fedora
- unzip /tmp/terraform_1.0.1_linux_amd64.zip
- mv /tmp/terraform /usr/local/bin/
- chmod 755 /usr/local/bin/terraform
- terraform –version ➔ validate the version

Terraform Architecture



Terraform Workflow