NOLSATU

Terraform Administration (TF-ADM)

Keywords

Automation, Infrastructure As Code (IAC), DevOps

References

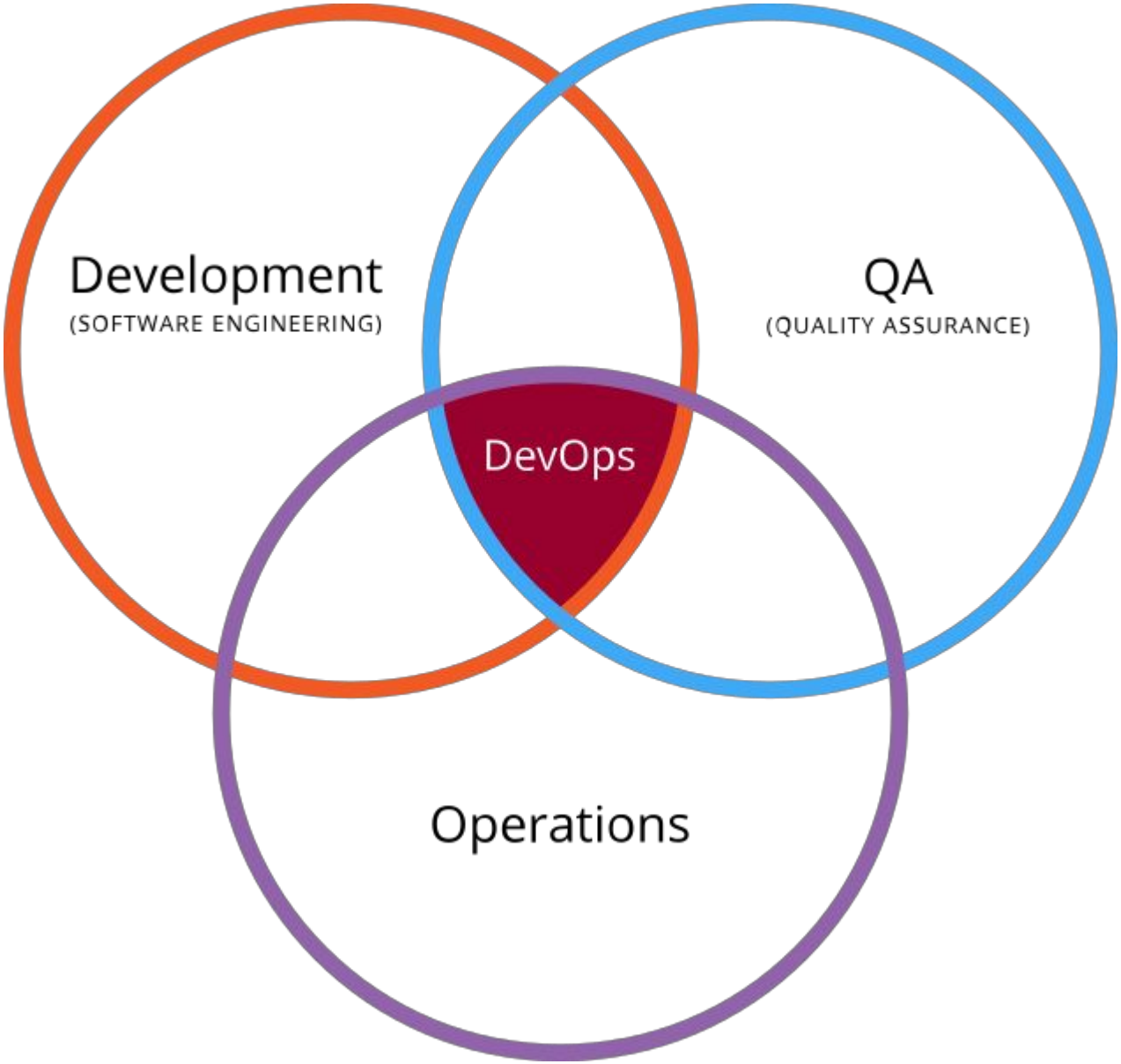
https://www.terraform.io/docs/index.html

Terraform: Up and Running 2nd Edition - Yevgeniy Brikman, 2019

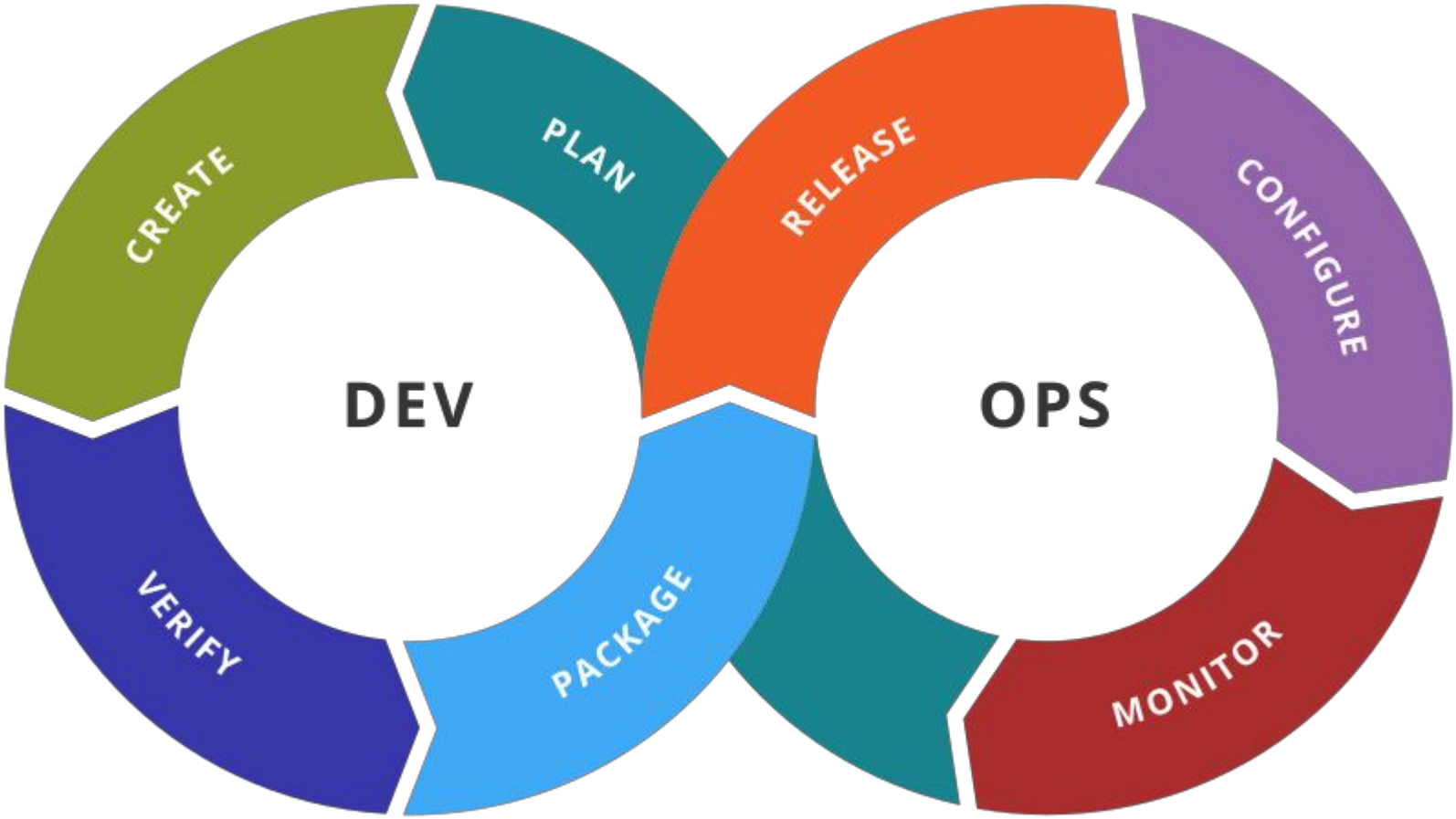
The Rise of DevOps

The goal of DevOps is to make software delivery vastly more efficient.

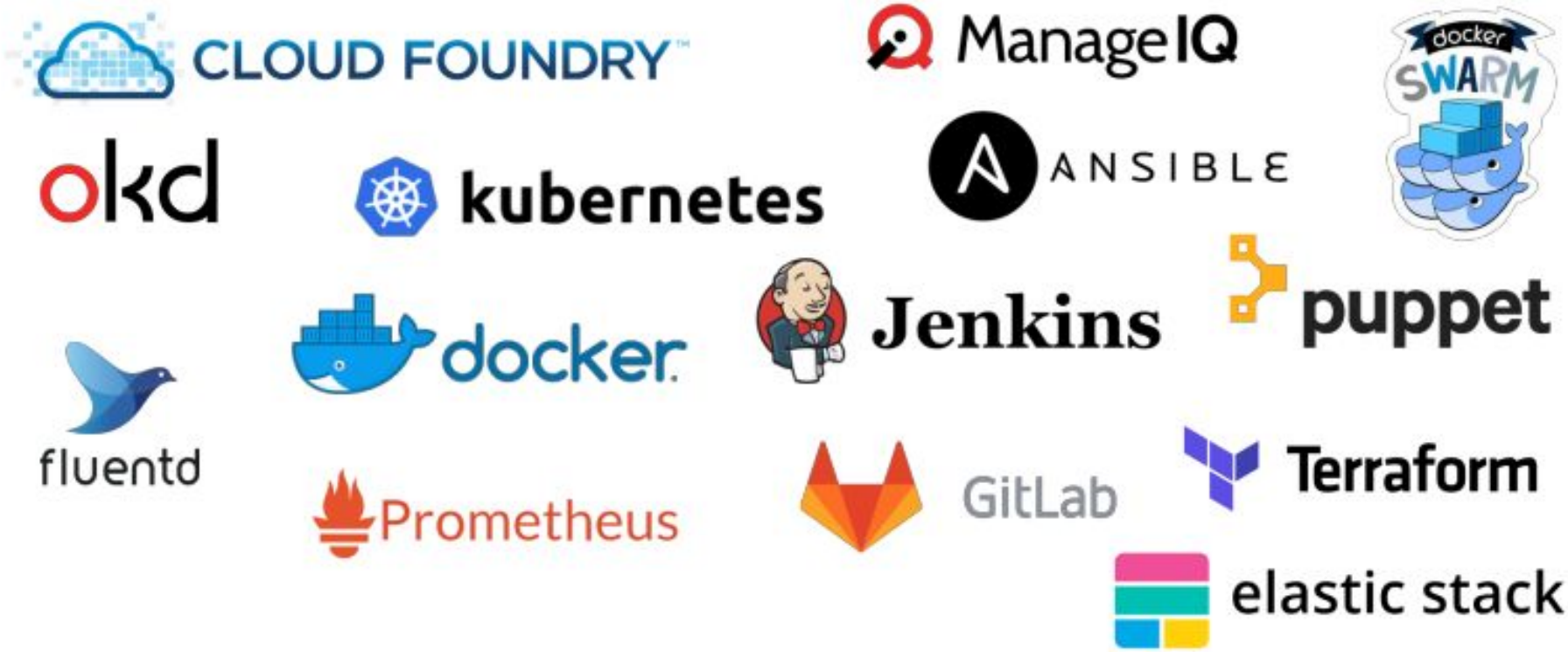
DevOps Intersection



DevOps Stages



DevOps Tools



What Is Infrastructure as Code?

Infrastructure as Code

- Write and execute code to define, deploy, and update your infrastructure.
- Treat all aspects of operations as software
- Manage almost everything in code, including servers, databases, networks, log files, application configuration, documentation, automated tests, deployment processes, and so on.

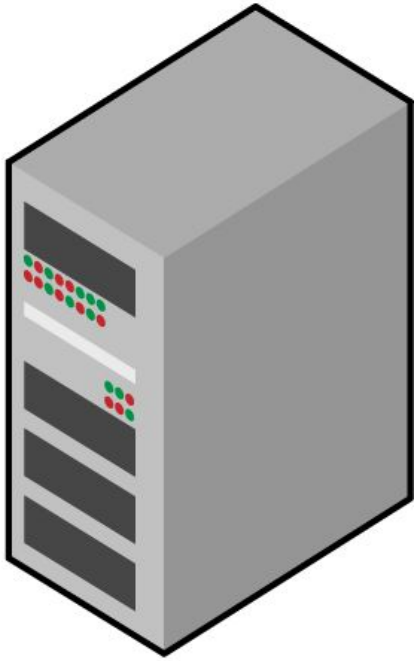
Categories of IAC tools

- Ad hoc scripts
- Configuration management (CM) tools
- Server templating tools
- Server provisioning tools

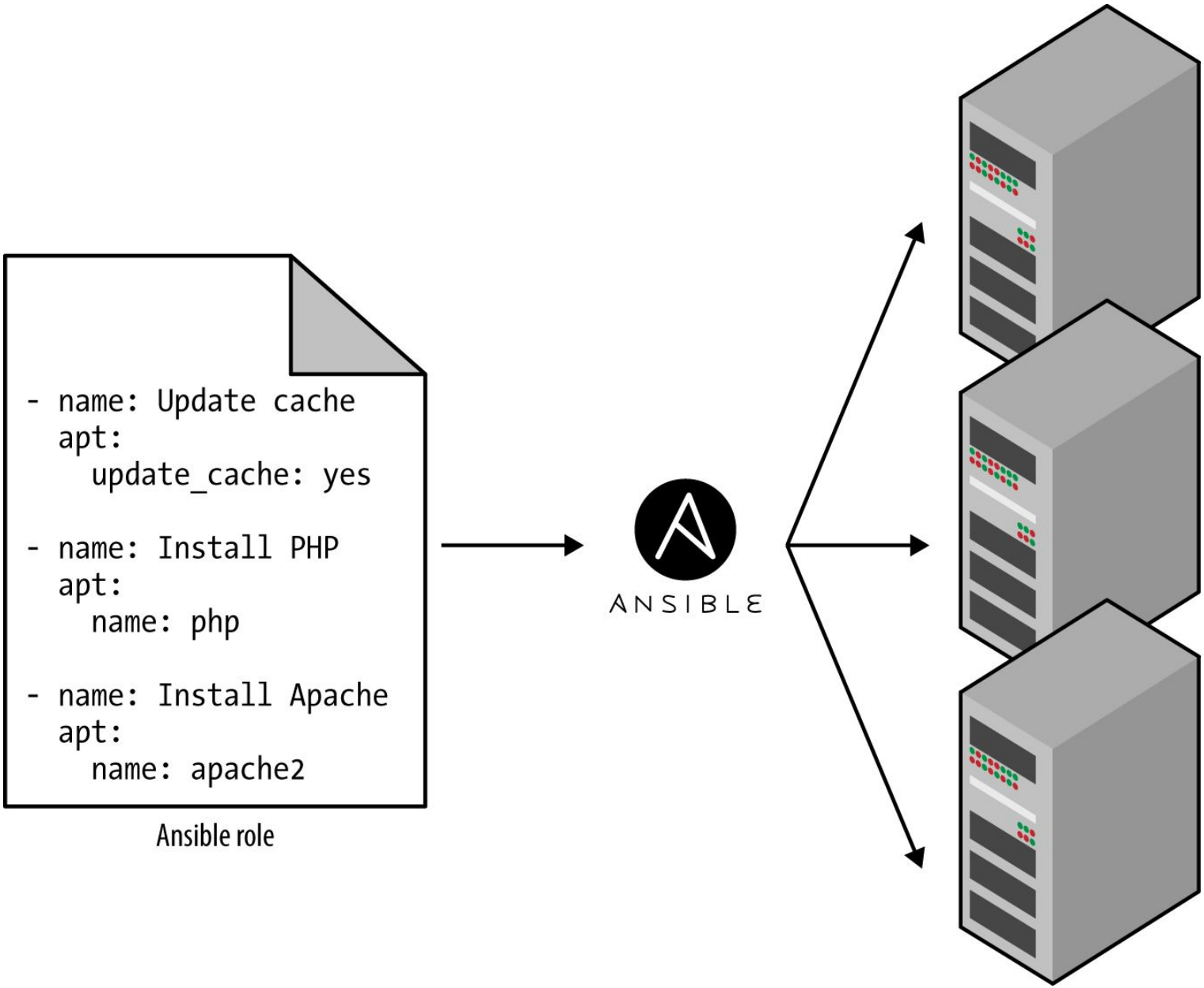
Ad Hoc Scripts

```
apt-get update

apt-get install \
   -y \
   php \
   apache 2

git clone \
   github.com/foo/bar \
   /var/www/html/app

service apache2 start
```

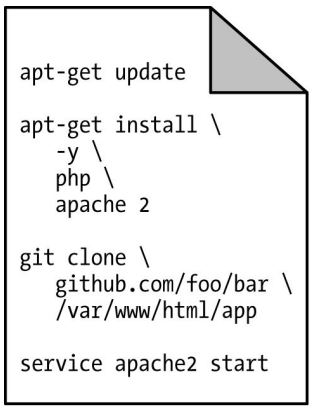
Ad hoc script

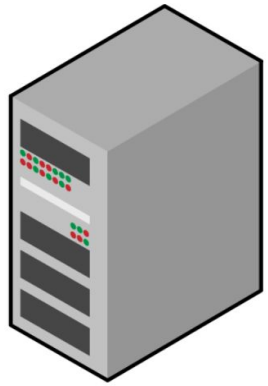


Configuration Management Tools



Server Templating Tools

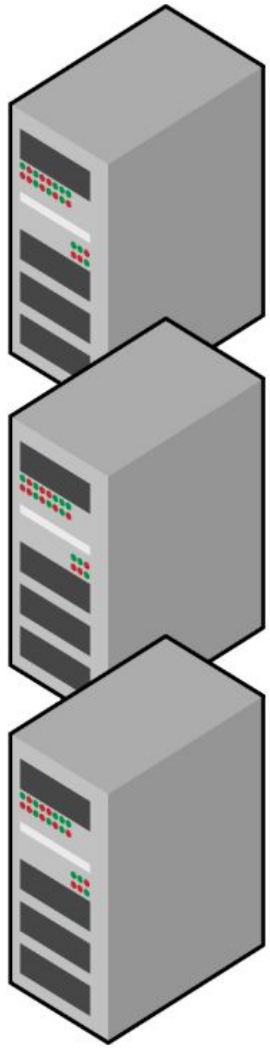




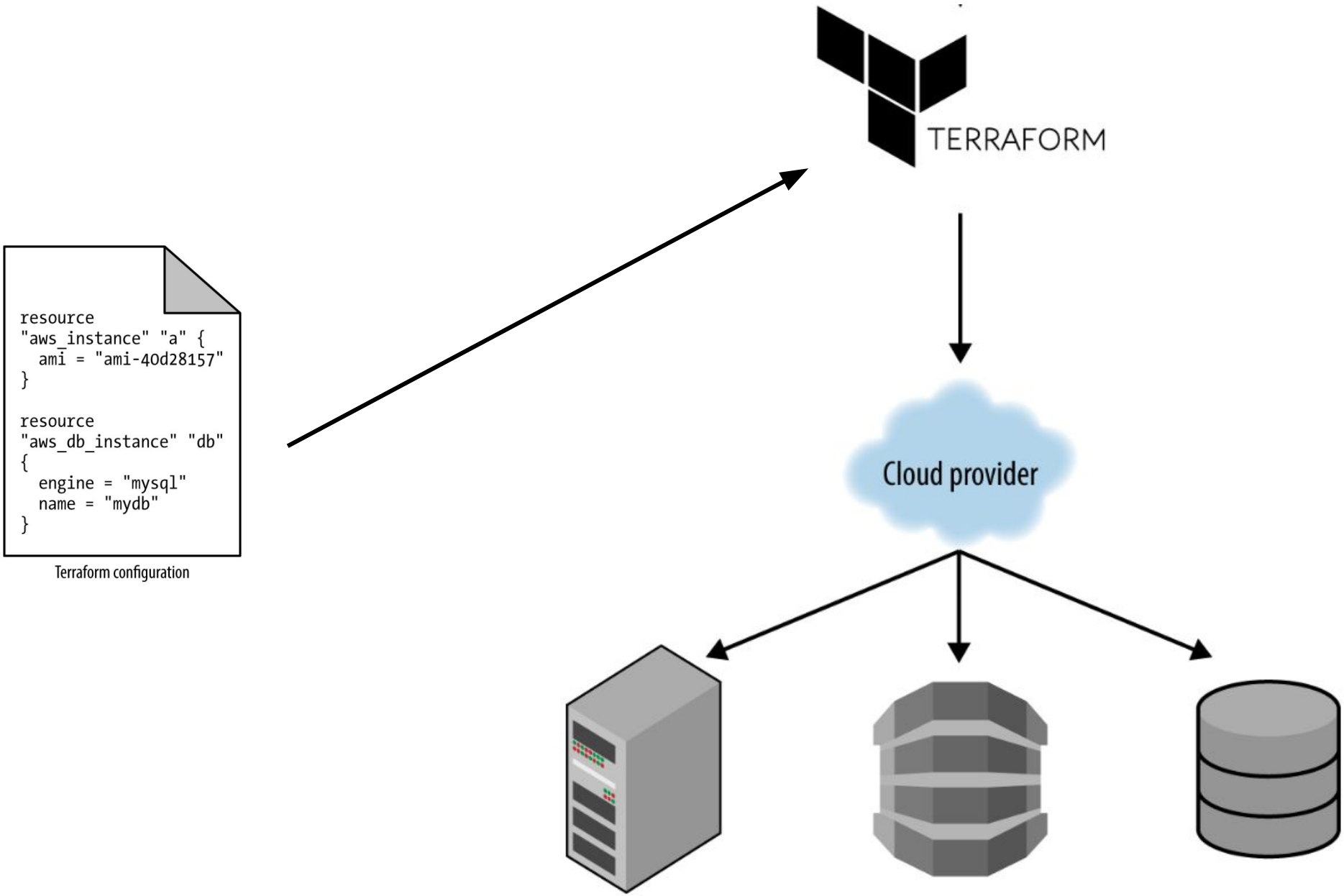
Server image

HashiCorp
Packer

ANSIBLE



Server Provisioning Tools



Benefits of IAC

- Self-service
- Speed and safety
- Documentation
- Version control
- Validation
- Reuse
- Happiness

HashiCorp

Terraform

Terraform

- An open source tool created by HashiCorp.
- Written in the Go programming language.
- A tool for building, changing, and versioning infrastructure safely and efficiently.

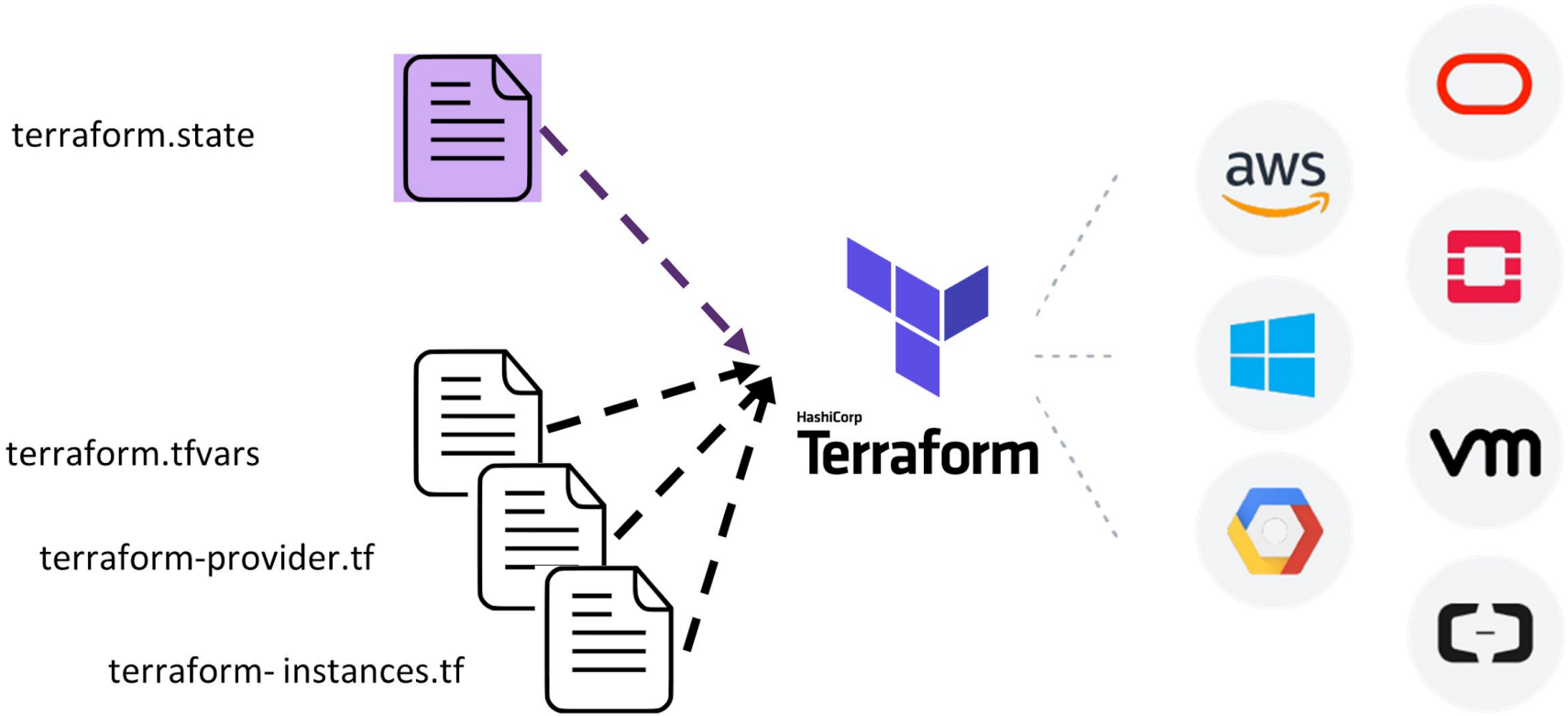
How Terraform Works

Configuration files describe to Terraform the components needed to run a single application or your entire datacenter.

Terraform generates an execution plan describing what it will do to reach the desired state, and then executes it to build the described infrastructure.

As the configuration changes, Terraform is able to determine what changed and create incremental execution plans which can be applied.

How Terraform Works



Use Cases

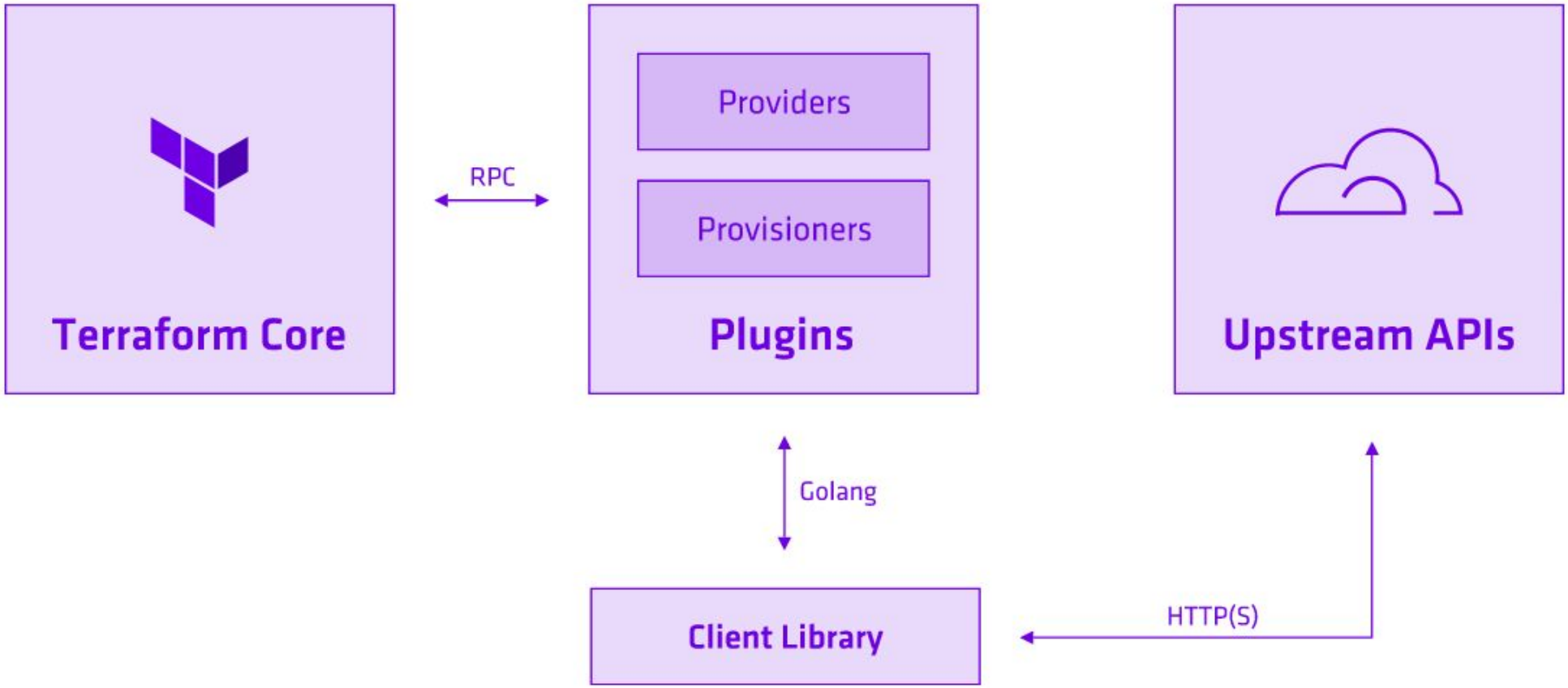
- Heroku App Setup
- Multi-Tier Applications
- Self-Service Clusters
- Software Demos
- Disposable Environments
- Resource Schedulers
- Multi-Cloud Deployment

Terraform Main Part

Terraform Core

Terraform Plugins

Terraform Main Part



Terraform Core

Infrastructure as code: reading and interpolating configuration files and modules

Resource state management

Plan execution

Communication with plugins over RPC

Terraform Plugins

Terraform Plugins are written in Go and are executable binaries invoked by Terraform Core over RPC.

Each plugin exposes an implementation for a specific service, such as AWS, or provisioner, such as bash.

All Providers and Provisioners used in Terraform configurations are plugins.

Terraform Plugin Types

Providers

Provisioners

Providers

Providers are the most common type of Plugin, which expose the features that a specific service offers via its application programming interface (API).

Providers define Resources and are responsible for managing their life cycles.

Examples of providers are OpenStack Provider and Docker Provider.

Providers

Initialization of any included libraries used to make API calls

Authentication with the Infrastructure Provider

Define Resources that map to specific Services

Providers


libvirt
VIRTUALIZATION API
vmware®
TM
aws
ORACLE
CLOUD
HUAWEI

Provisioners

Execute scripts on a local or remote machine as part of resource creation or destruction.

Bootstrap a resource, cleanup before destroy, run configuration management, etc.

Provisioners

chef

connection

file

local-exec

null_resource (Without a Resource)

remote_exec

salt_masterless

Provisioner Connection

```
provisioner "file" {
  source      = "file/myapp.conf"
  destination = "/etc/myapp.conf"

  connection {
    type     = "ssh"
    user     = "root"
    password = "rahasia"
    host     = "10.10.10.1"
  }
}
```

Provisioner Connection

```
provisioner "remote-exec" {
  connection {
    type     = "ssh"
    user     = "root"
    password = "rahasia"
    host     = "10.10.10.1"
  }

  inline = [
    "chmod +x /tmp/script.sh",
    "/tmp/script.sh args",
  ]
}
```

Provisioner file

```
provisioner "file" {
  source      = "file/myapp.conf"
  destination = "/etc/myapp.conf"

  connection {
    type     = "ssh"
    user     = "root"
    password = "rahasia"
    host     = "10.10.10.1"
  }
}
```

Provisioner Local Exec

```
resource "openstack_compute_instance_v2" "instance" {
  provisioner "local-exec" {
    command = "echo {var.address} >> private_ips.txt"
  }
}
```

```
resource "null_resource" "testing" {
  provisioner "local-exec" {
    command = "echo testing"
  }
}
```

Provisioner Remote Exec

```
provisioner "remote-exec" {
  connection {
    type     = "ssh"
    user     = "root"
    password = "rahasia"
    host     = "10.10.10.1"
  }

  inline = [
    "chmod +x /tmp/script.sh",
    "/tmp/script.sh args",
  ]
}
```

Provisioner Null Resource

```
resource "null_resource" "testing_remote"  {
  provisioner "remote-exec" {
    connection {
      type     = "ssh"
      user     = "root"
      password = "rahasia"
      host     = "10.10.10.1"
    }
  inline = [ "apt -y update" ]
  }
}
```

```
resource "null_resource" "testing" {
  provisioner "local-exec" {
    command = "echo testing"
  }
}
```

Provisioner Chef

```
provisioner "chef"  {
  connection {
    type     = "ssh"
    user     = "root"
    password = "rahasia"
    host     = "10.10.10.1"
  }
  environment = "_default"
  run_list = ["nginx::default"]
  node_name = var.instance_name
  server_url = var.chef_server
  recreate_client = true
  user_name = var.chef_server_user
  user_key = var.chef_server_key
  fetch_chef_certificates = true
}
```

Provisioner Puppet

```
provisioner "puppet" {
  server              = var.puppet_server
  server_user         = "ubuntu"
  extension_requests = { pp_role = "webserver" }
}
```

Provisioner Salt Masterless

```
provisioner "salt-masterless" {
  "local_state_tree" = "/srv/wbbserver.sls"
}
```

Lab 1

Terraform Administration

Lab 1 – Manage OpenStack

Install Terraform

Create OpenStack Instance

Create OpenStack Instance - Using Variable

Create OpenStack Instance - Bootstrap Web

Create OpenStack Instance - LB With HAProxy

Lab 2

Terraform Administration

Lab 2 – Manage Docker

Install Docker

Run a container

Run a container - Expose Port

Run a container - File Upload

Run a container - Volume

Deploy app to Heroku

NolSatu.id