TERRAFORM ADMINISTRATION (TF-ADM)

Keywords

Automation, Infrastructure As Code (IAC), DevOps

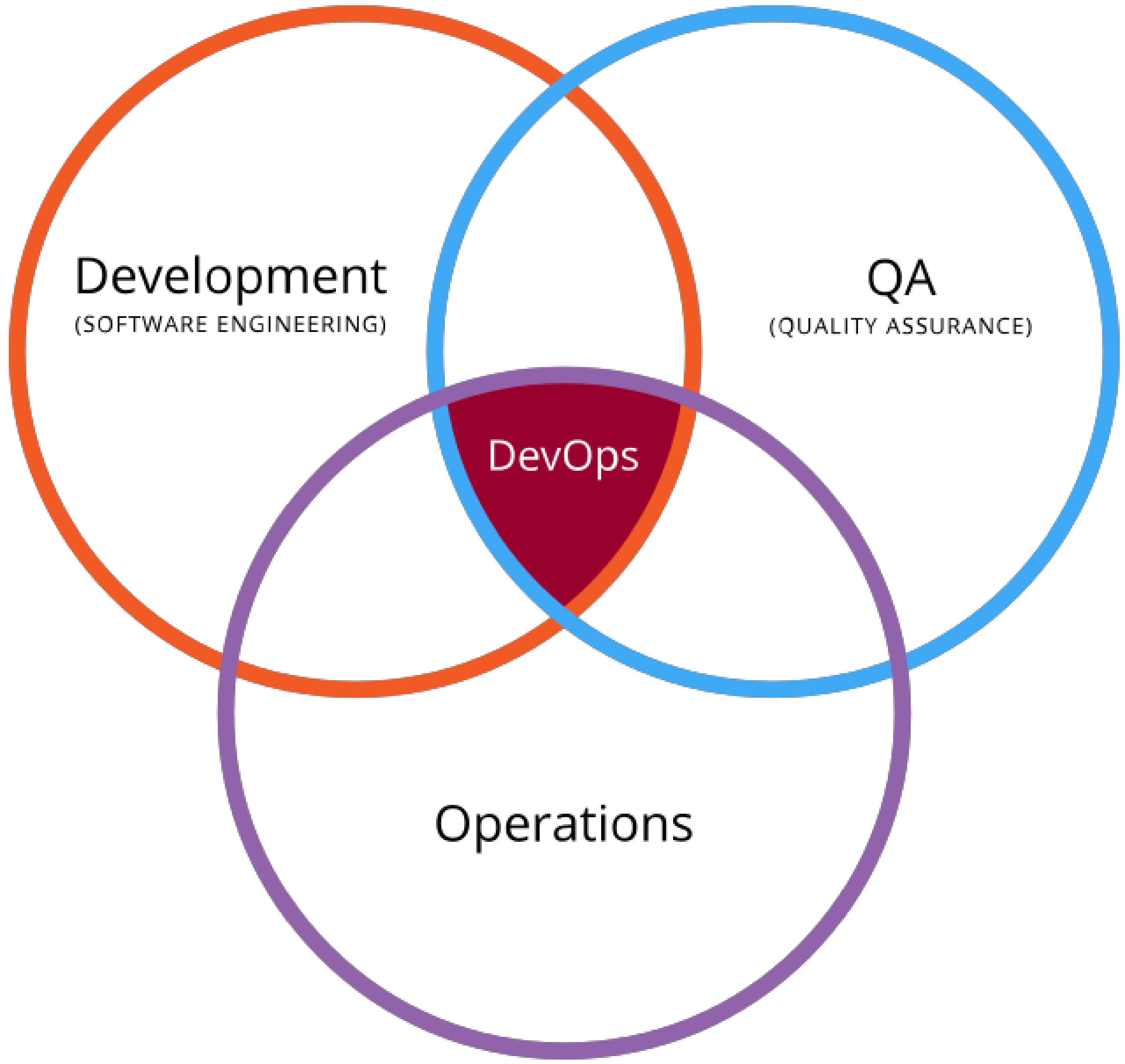
References

- https://www.terraform.io/docs/index.html
- Terraform: Up and Running - Yevgeniy Brikman, 2017

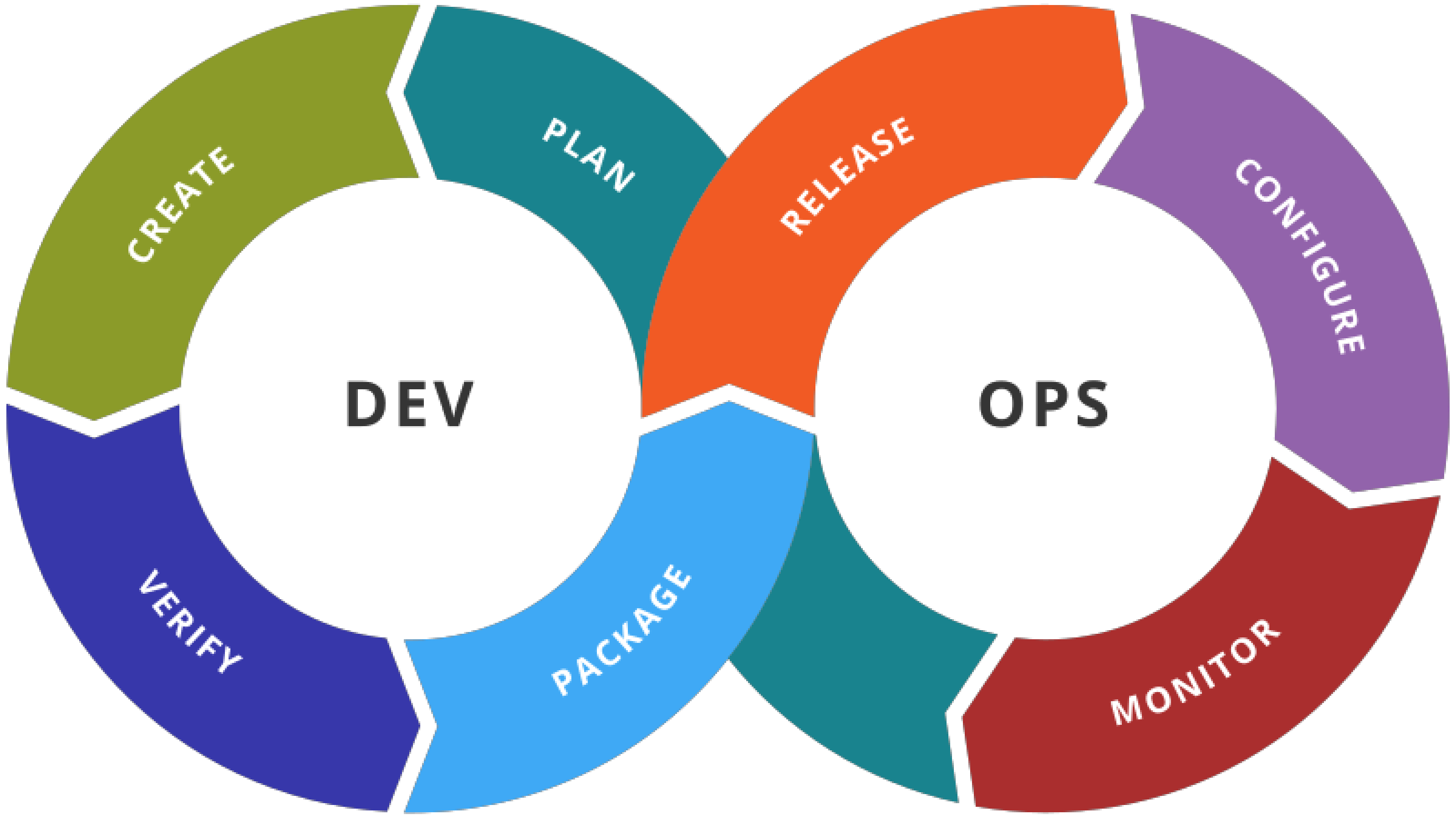
The Rise of DevOps

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

DevOps Intersection



DevOps Stages



DevOps Tools

CLOUD FOUNDRY™

ManageIQ

docker SWARM

okd

kubernetes

ANSIBLE

Jenkins

puppet

fluentd

docker

Prometheus

GitLab

Terraform

elastic stack

What Is Infrastructure as Code?

Infrastructure as Code

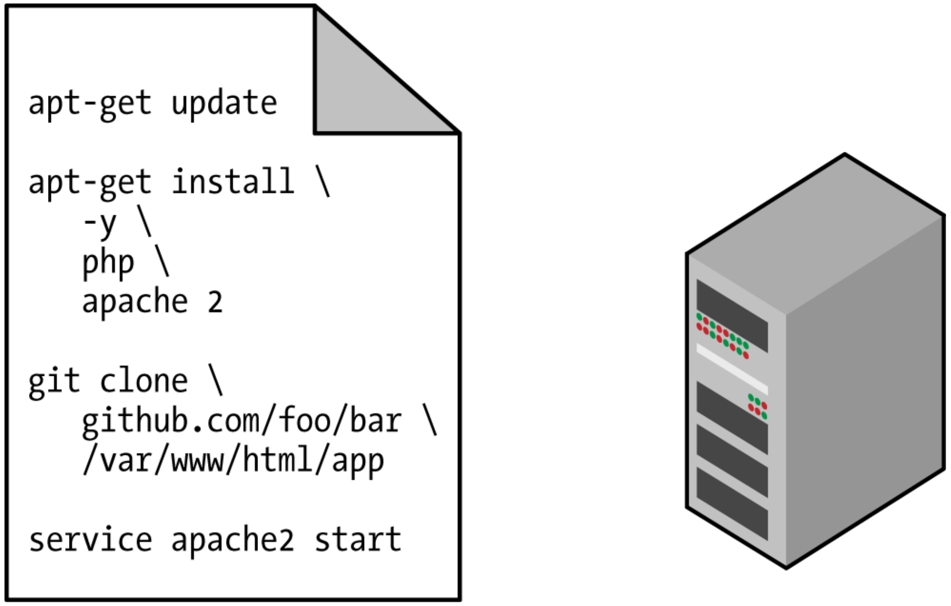
- The idea behind infrastructure as code (IAC) is that you write and execute code to define, deploy, and update your infrastructure.
- Treat all aspects of operations as software
- A key insight of DevOps is that you can manage almost everything in code, including servers, databases, networks, log files, 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

You take whatever task you were doing manually, break it down into discrete steps, use your favorite scripting language (e.g., Bash, Ruby, Python) to define each of those steps in code, and execute that script on your server

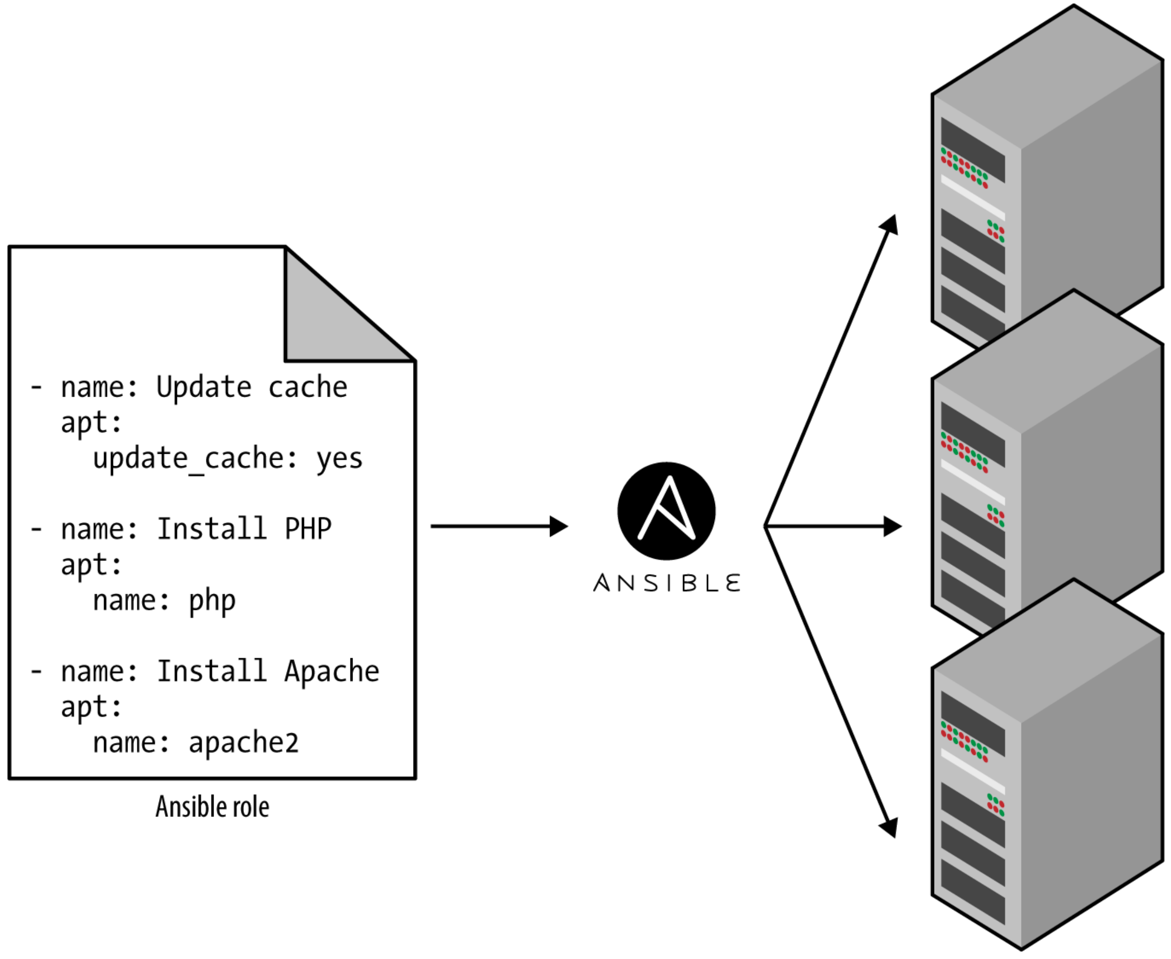


```
apt-get update

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

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

service apache2 start
```

Ad hoc script

CM Tools

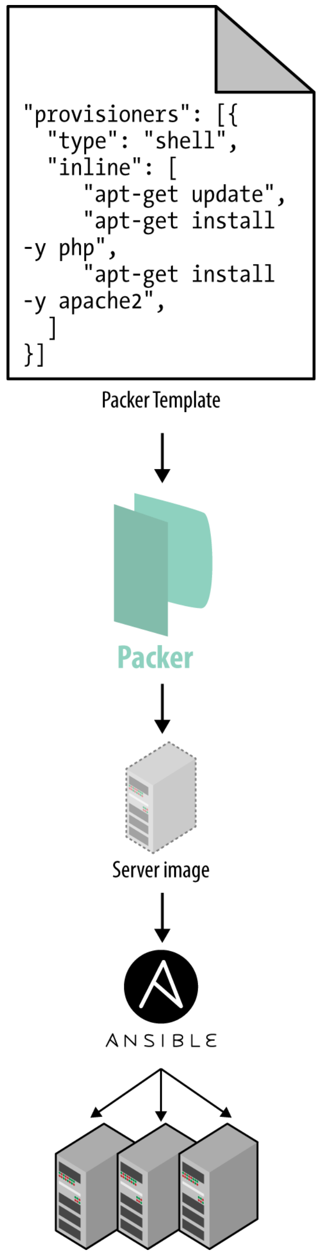
- Tools: Chef, Puppet, Ansible, and SaltStack
- They are designed to install and manage software on existing servers.



```
- name: Update cache
  apt:
    update_cache: yes

- name: Install PHP
  apt:
    name: php

- name: Install Apache
  apt:
    name: apache2
```

Server Templating Tools

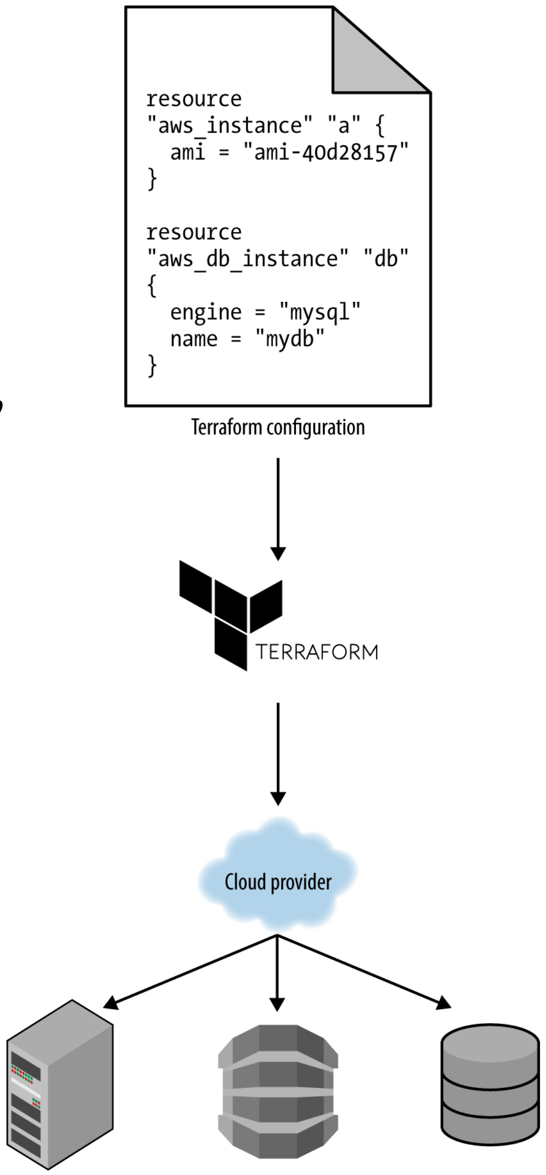
- An alternative to configuration management that are server templating tools such as Docker, Packer, and Vagrant.
- The idea behind server templating tools is to create an image of a server that captures a fully self-contained “snapshot” of the the operating system, the software, the files, and all other relevant details.



```
"provisioners": [{
  "type": "shell",
  "inline": [
     "apt-get update",
     "apt-get install
-y php",
     "apt-get install
-y apache2",
  ]
}]
```

Server Provisioning Tools

- Server provisioning tools such as Terraform, CloudFormation, and OpenStack Heat are responsible for creating the servers themselves.
- You can use provisioning tools to not only create servers, but also databases, caches, load balancers, queues, monitoring, subnet configurations, firewall settings, routing rules, SSL certificates, and almost every other aspect of your infrastructure



```
resource
"aws_instance" "a" {
  ami = "ami-40d28157"
}

resource
"aws_db_instance" "db"
{
  engine = "mysql"
  name = "mydb"
}
```

Benefits of IAC

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

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.

The key features

- Infrastructure as Code
- Execution Plans
- Resource Graph
- Change Automation

Use Cases

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

Terraform’s Main Part

- Terraform Core
- Terraform Plugins

Terraform Core (1)

- Terraform Core is a statically-compiled binary written in the Go programming language.
- The compiled binary is the command line tool (CLI) terraform, the entrypoint for anyone using Terraform.
- Terraform Core uses remote procedure calls (RPC) to communicate with Terraform Plugins, and offers multiple ways to discover and load plugins to use.
- The code is open source and hosted at github.com/hashicorp/terraform.

Terraform Core (2)

The primary responsibilities of Terraform Core are:

- Infrastructure as code: reading and interpolating configuration files and modules
- Resource state management
- Construction of the Resource Graph
- Plan execution
- Communication with plugins over RPC

Terraform Plugins (1)

- 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 Plugins (2)

There are two types of plugins supported by Terraform:

- Providers
- Provisioners

Terraform Plugins (3)

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.

Terraform Plugins (4)

Providers

The primary responsibilities of Provider Plugins are:

- Initialization of any included libraries used to make API calls
- Authentication with the Infrastructure Provider
- Define Resources that map to specific Services

Terraform Plugins (5)

Provisioners

- Provisioners are used to execute scripts on a local or remote machine as part of resource creation or destruction.
- Provisioners can be used to bootstrap a resource, cleanup before destroy, run configuration management, etc.

Terraform Plugins (6)

Provisioners

The primary responsibilities of Provisioner Plugins are:

- Executing commands or scripts on the designated Resource after creation, or on destruction.
-

Providers (1)

- Major Cloud Provider
  1. AliCloud
  2. AWS
  3. Azure
  4. Google Cloud
  5. Oracle Cloud Infrastructure
  6. vCloud Director
  7. VMware vSphere

https://www.terraform.io/docs/providers/type/major-index.html

Providers (2)

- Cloud
  1. Arukas
  2. Brightbox
  3. CenturyLinkCloud
  4. CloudScale.ch
  5. CloudStack
  6. DigitalOcean
  7. Fastly
  8. OpenStack
  9. Scaleway
  10. Heroku
  11. Hetzner Cloud
  12. HuaweiCloud
  13. Linode
  14. Nutanix

https://www.terraform.io/docs/providers/type/cloud-index.html

Providers (3)

- Infrastructure Software

1. Chef
2. Consul
3. Docker
4. Helm
5. Kubernetes
6. Mailgun
7. Nomad
8. RabbitMQ
9. Rancher
10. RightScale
11. Rundeck
12. Spotinst
13. Terraform Enterprise
14. Vault

https://www.terraform.io/docs/providers/type/infra-index.html

Providers (4)

- Network
  1. Cloudflare
  2. Cisco ASA
  3. DNS
  4. DNSimple
  5. DNSMadeEasy
  6. F5 BIG-IP
  7. HTTP
  8. NS1
  9. Palo Alto Networks
  10. PowerDNS
  11. UltraDNS

https://www.terraform.io/docs/providers/type/network-index.html

Providers (5)

- Version Control
  1. Bitbucket
  2. GitHub
  3. GitLab

https://www.terraform.io/docs/providers/type/vcs-index.html

Providers (6)

- Monitoring & System Management
  1. Circonus
  2. Datadog
  3. Dyn
  4. Grafana
  5. Icinga2
  6. Librato
  7. Logentries
  8. LogicMonitor
  9. New Relic
  10. OpsGenie
  11. PagerDuty
  12. Runscope
  13. StatusCake

https://www.terraform.io/docs/providers/type/monitor-index.html

Providers (7)

- Database
  1. InfluxDB
  2. MySQL
  3. PostgreSQL

https://www.terraform.io/docs/providers/type/database-index.html

Providers (8)

- Miscellaneous
  1. ACME
  2. Archive
  3. Cobbler
  4. External
  5. Ignition
  6. Local
  7. Netlify
  8. Null
  9. Random
  10. Template
  11. TLS

https://www.terraform.io/docs/providers/type/misc-index.html

Providers (9)

- Community
  1. Active Directory
  2. Docker Machine
  3. Dropbox
  4. Elasticsearch
  5. Kibana
  6. Kafka
  7. Logentries
  8. LXD
  9. libvirt
  10. oVirt
  11. Proxmox
  12. Spinnaker

https://www.terraform.io/docs/providers/type/community-index.html

Provisioners (9)

1. chef
2. connection
3. hile
4. habitat
5. local-exec
6. null_resource
7. remote_exec
8. salt_masterless

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

Lab 1

Terraform Administration

Lab 1 – Manage OpenStack

- Install Terraform
- Create OpenStack Instance
- Create OpenStack Instance - Using Variable
- Create OpenStack Instance - Bootstrapweb

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

NolSatu.id