iPMAC
NET YOUR WORK

DEVSECOPS COURSE

INFRASTRUCTURE AS CODE

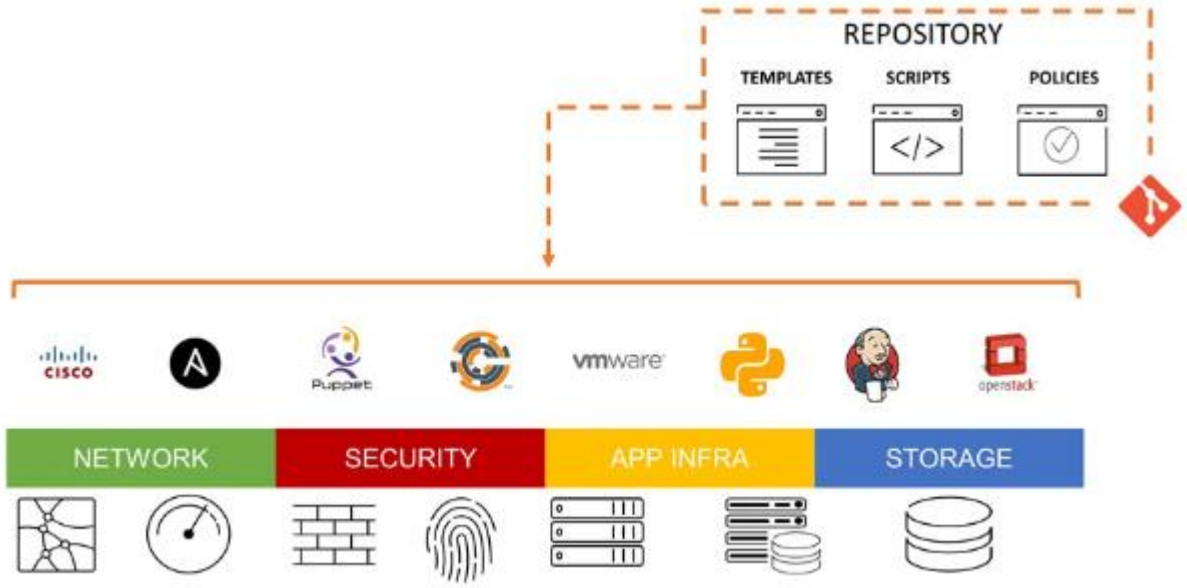
TRAINER: TRAN HUU HOA

AGENDA

- Concepts
- Terraform
- Ansible

CONCEPTS

Infrastructure as Code (IaC) is a practice in which infrastructure is managed and provisioned using code and software development techniques, such as version control and continuous integration. This approach allows for consistent and repeatable infrastructure deployments, reducing the risk of human error and improving collaboration among team



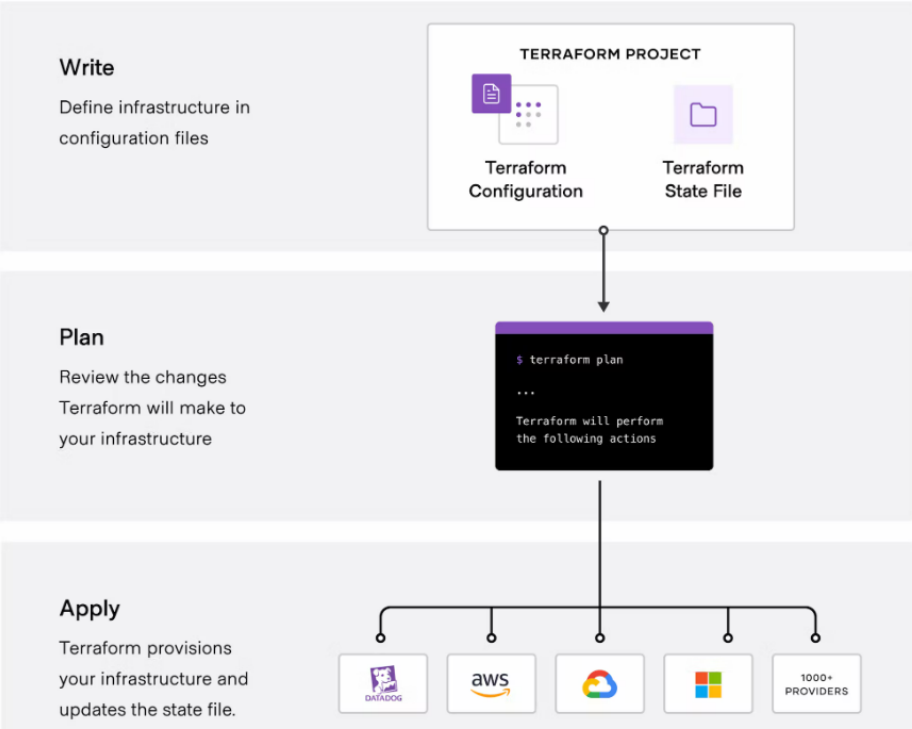
TERRAFORM

Concept

Terraform is an infrastructure as code tool that lets you build, change, and version cloud and on premise resources safely and efficiently. It lets you define both cloud and on premise resources in human-readable configuration files that you can version, reuse, and share. You can then use a consistent workflow to provision and manage all of your infrastructure throughout its lifecycle. Terraform can manage low-level components like compute, storage, and networking resources, as well as high-level components like DNS entries and SaaS features.

TERRAFORM

How it works:



TERRAFORM

How it works:

Core Concepts

- Defined in code
- Stored in source control
- Declarative or imperative
- Idempotent and consistent
- Push or pull

TERRAFORM

Use cases:

- Multi-Cloud Deployment
- Application Infrastructure Deployment, Scaling, and Monitoring Tools
- Self-Service Clusters
- PaaS Application Setup
- Software Defined Networking
- Kubernetes
- Parallel Environments
- Software Demos

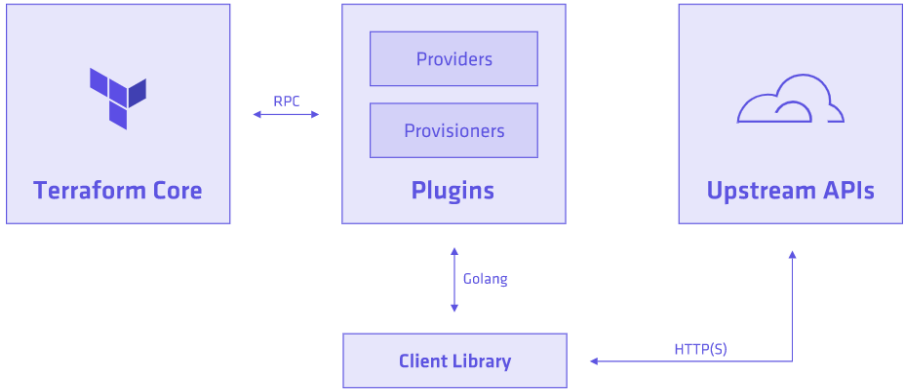
TERRAFORM

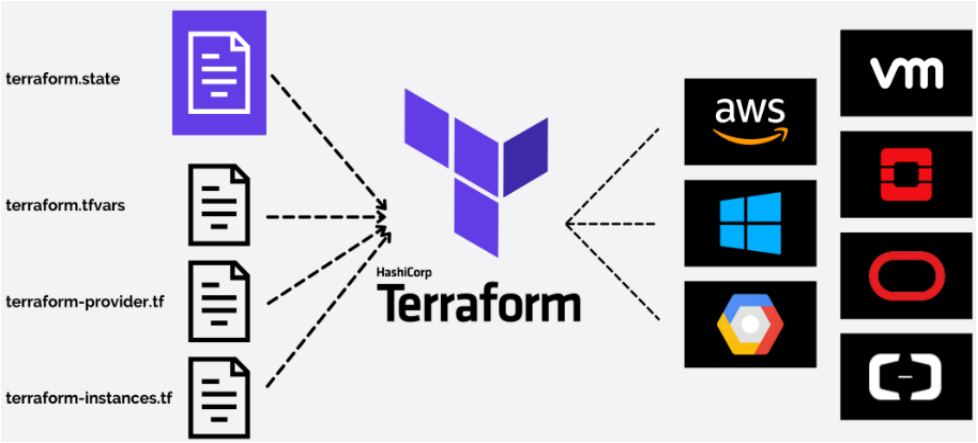
Options:

- Terraform open source
- Terraform cloud
- Terraform Enterprise
- Terraform CDK

TERRAFORM

Architecture





TERRAFORM

source code

HashiCorp configuration language

Why not JSON?

Human readable and editable

Interpolation

Conditional, functions, templates

TERRAFORM

Types

- Variable
- Provider
- Resource
- Output

TERRAFORM

Variable

```
#Create a variable
variable var_name {
 key = value #type, default, description
}
#Use a variable
${var.name} #get string
${var.map["key"]} #get map element
${var.list[idx]} #get list element
```

TERRAFORM

Provider

```
#Create provider
provider provider_name {
 key = value #depends on resource, use alias as needed
}
#Create data object
data data_type data_name {}
#Use data object
${data_type.data_name.attribute(args)}
```

TERRAFORM

Resource

```
#Create resource
resource resource_type resource_name {
 key = value #depends on resource
}
#Reference resource
${resource_type.resource_name.attribute(args)}
```

TERRAFORM

Terraform state

JSON format (Do not touch!)

Resources mappings and metadata

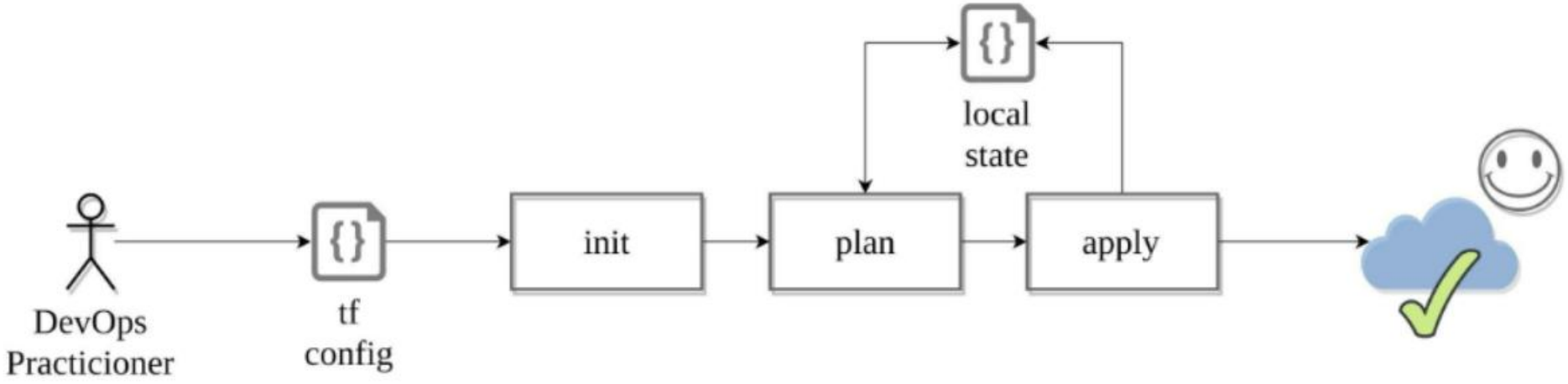
Locking

Local / remote

Environments

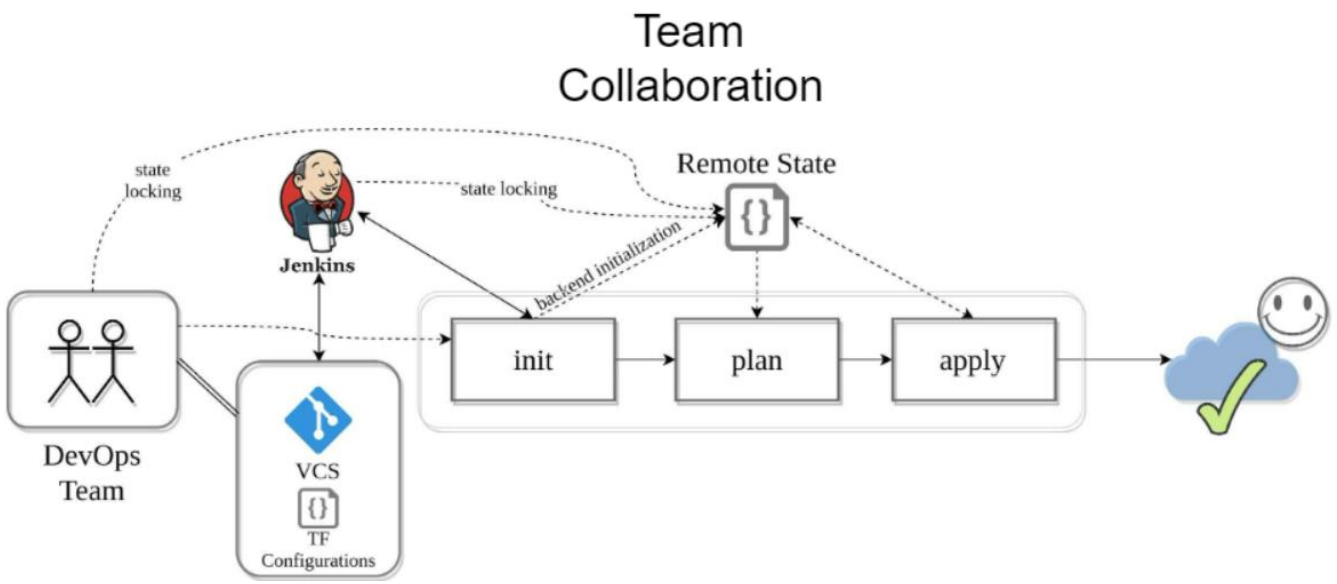
TERRAFORM

Simple Workflow



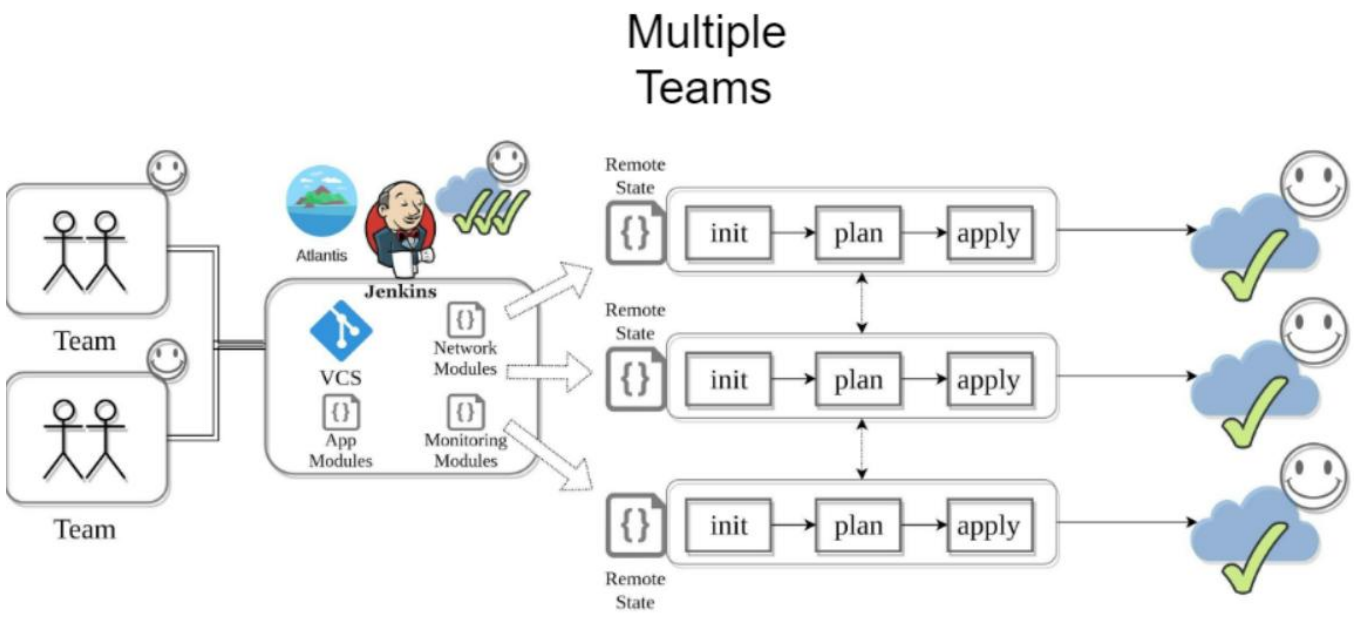
TERRAFORM

Advanced Workflow



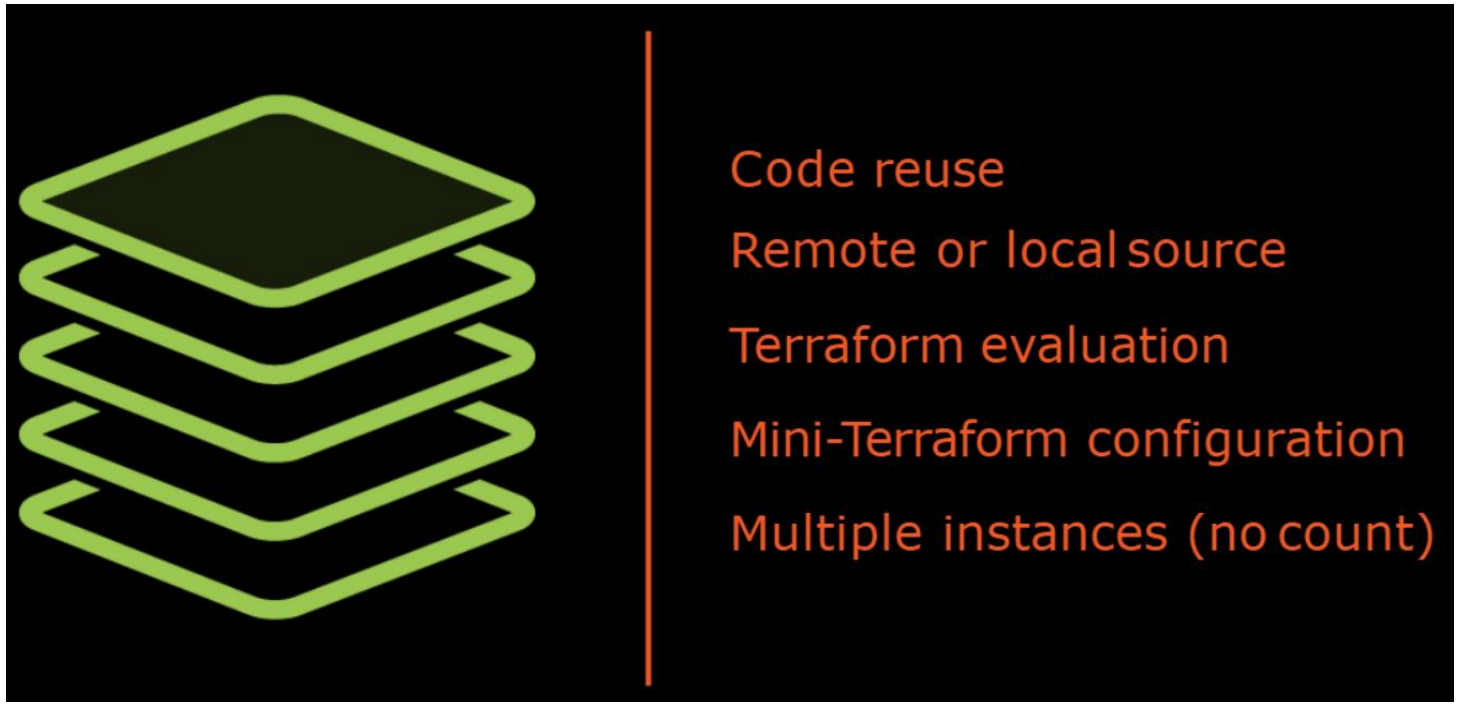
TERRAFORM

Advanced Workflow



TERRAFORM

Terraform Module

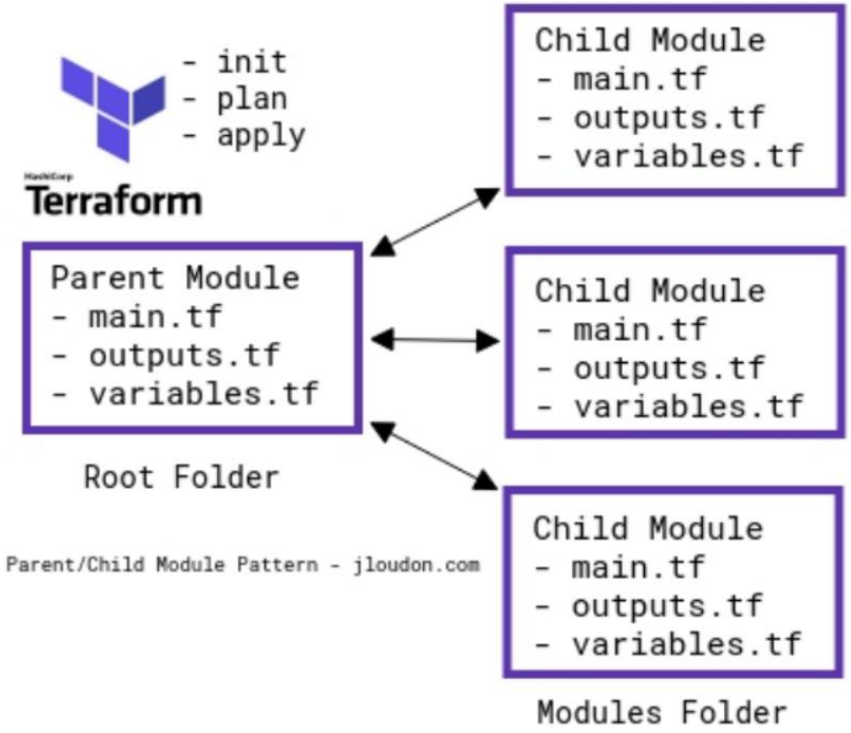


TERRAFORM

Terraform Module - components:

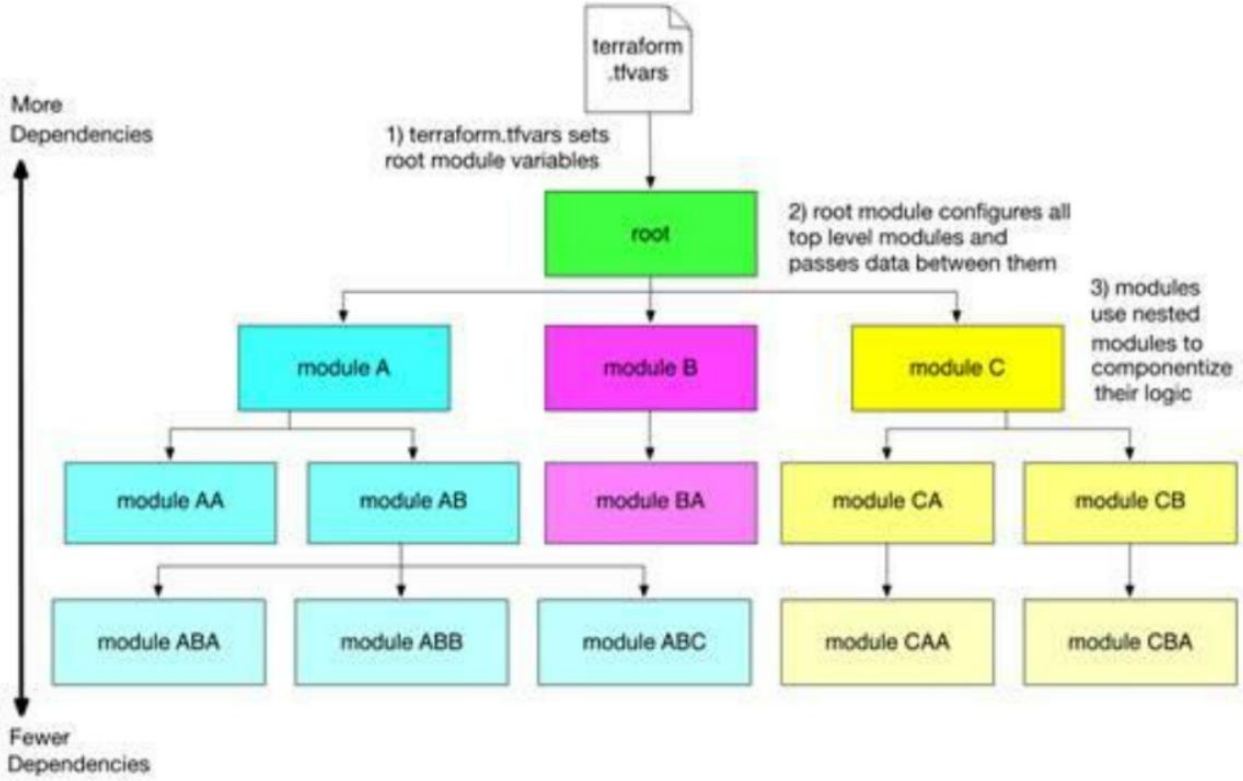
TERRAFORM

Terraform Module – How it works



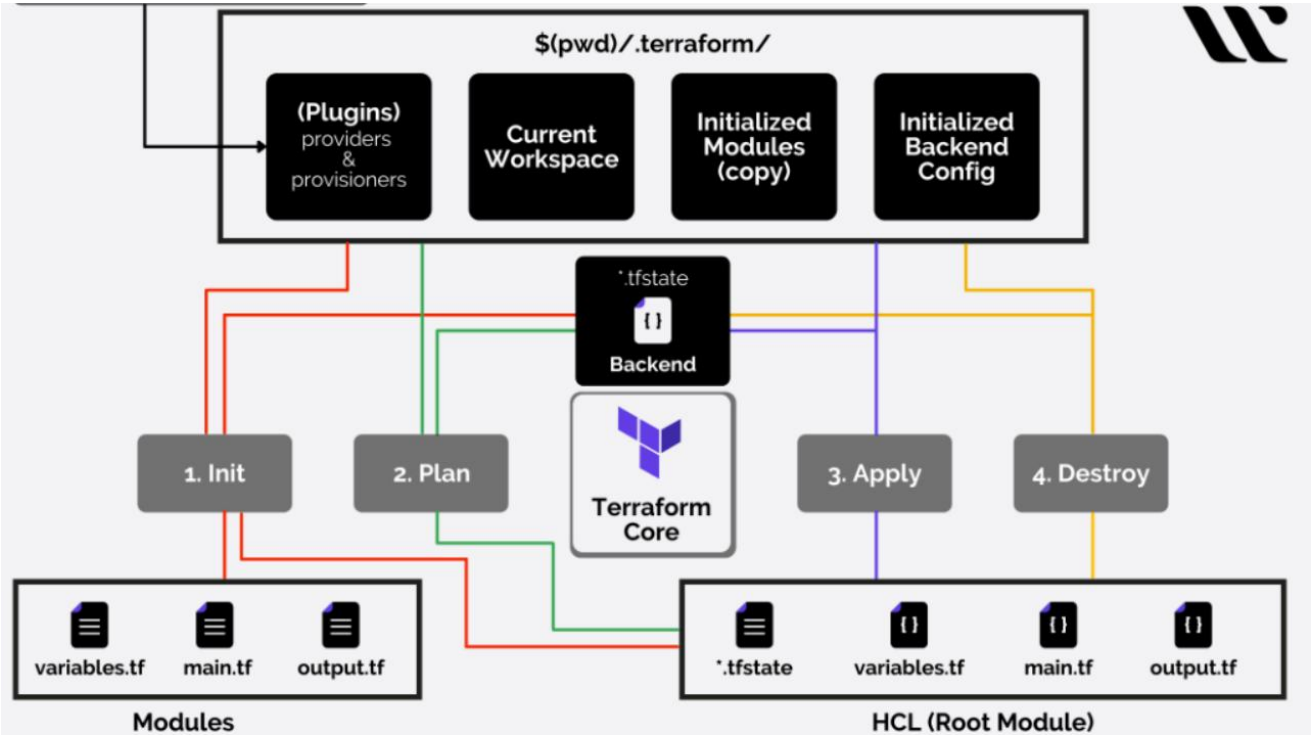
TERRAFORM

Terraform Module – How it works



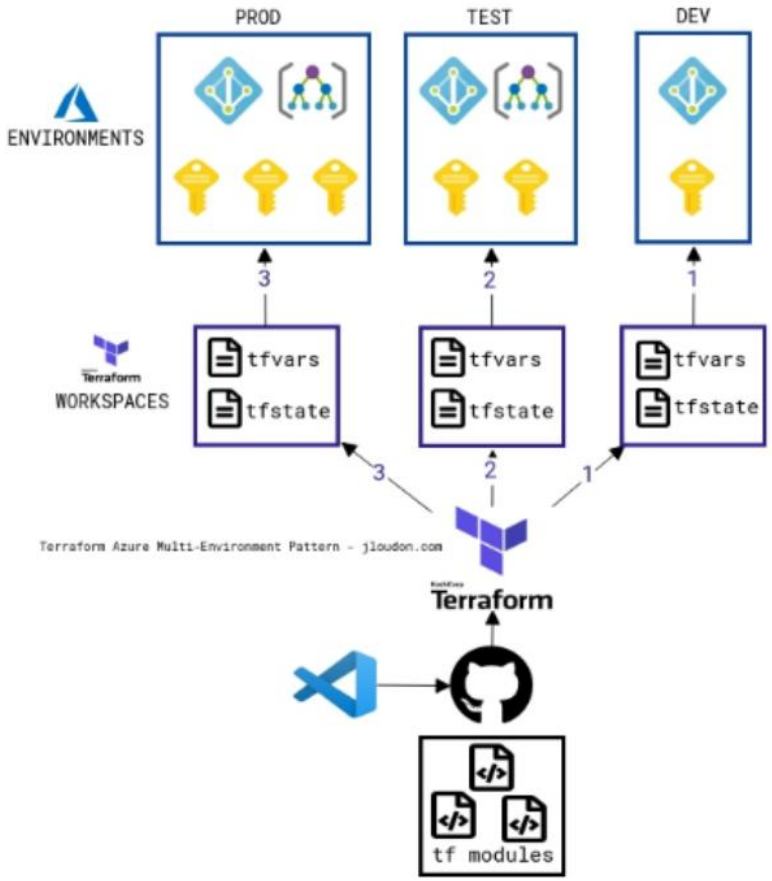
TERRAFORM

Terraform Module – How it works

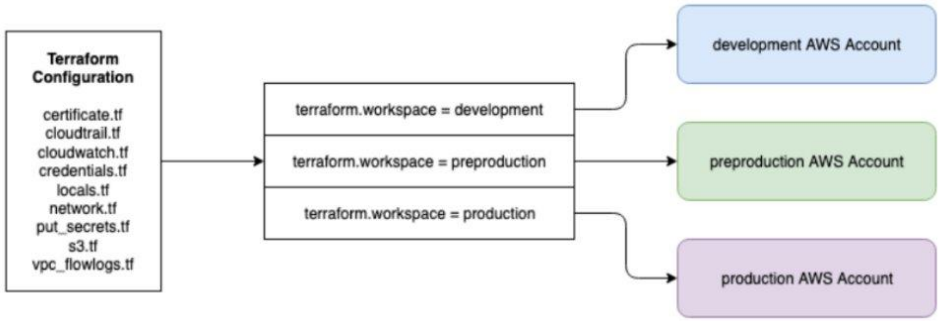


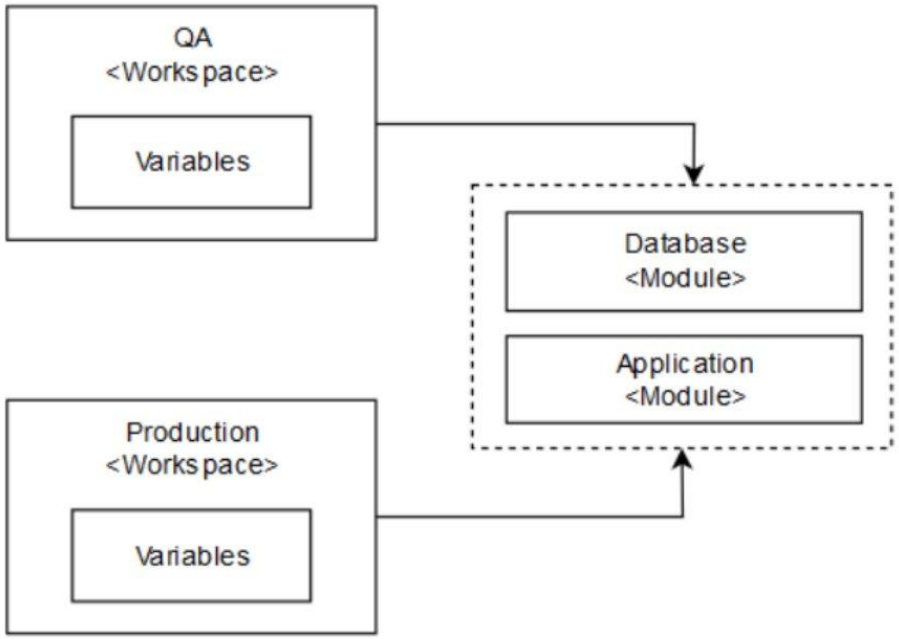
TERRAFORM

Terraform workspace



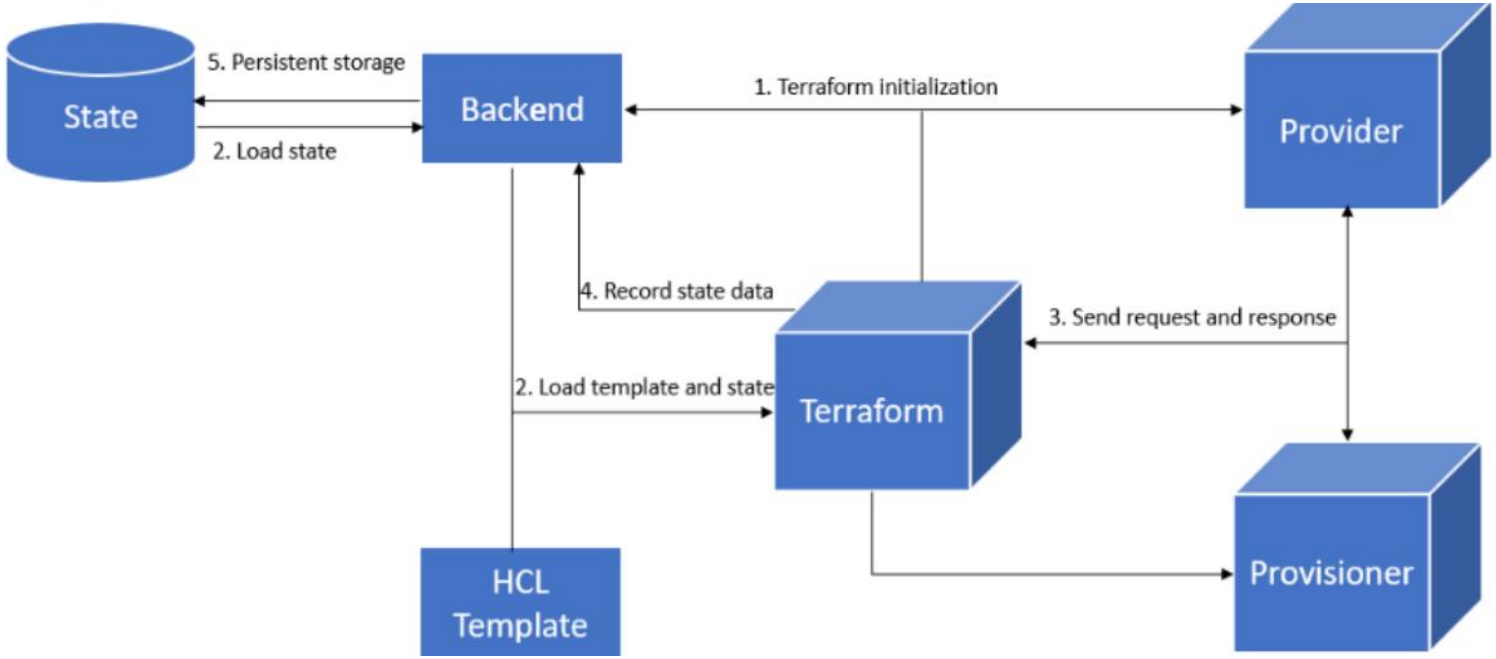
TERRAFORM


Terraform Configuration
certificate.tf
cloudtrail.tf
cloudwatch.tf
credentials.tf
locals.tf
network.tf
put_secrets.tf
s3.tf
vpc_flowlogs.tf
terraform.workspace = development
terraform.workspace = preproduction
terraform.workspace = production
development AWS Account
preproduction AWS Account
production AWS Account


QA
<Workspace>
Variables
Production
<Workspace>
Variables
Database
<Module>
Application
<Module>

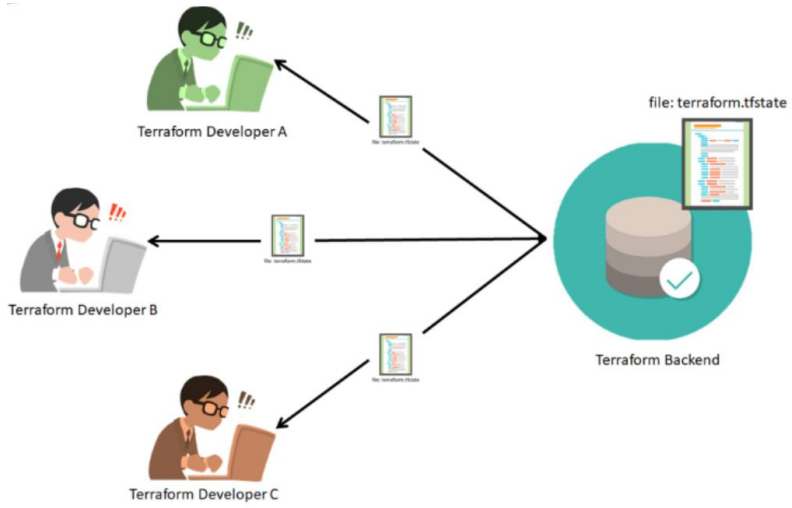
TERRAFORM

Terraform backend



TERRAFORM

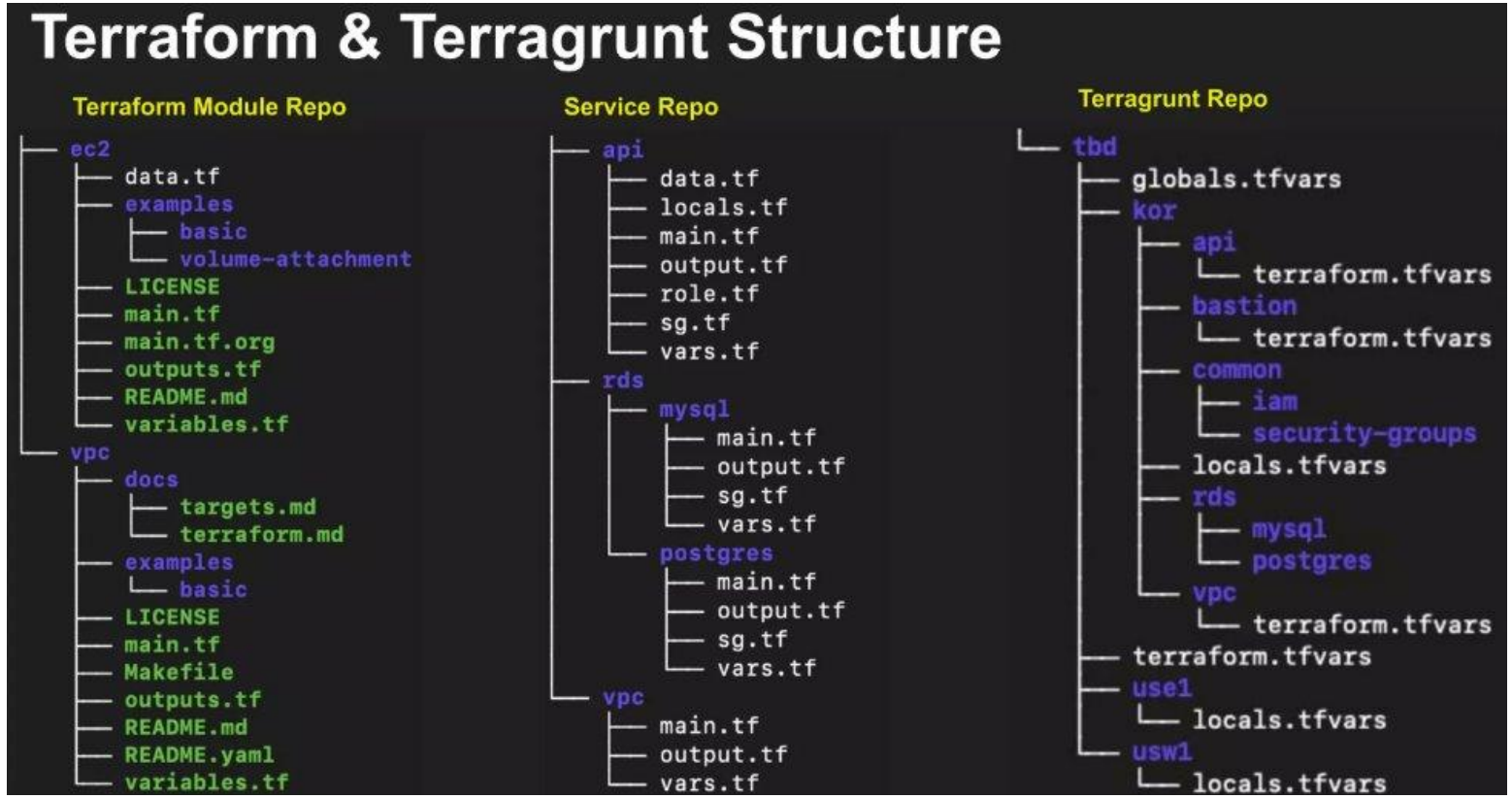
Terraform backend



- Locking
- Workspaces (former known as environments)
- Encryption at rest
- Versioning
- Note: Backend configuration doesn't support interpolations.

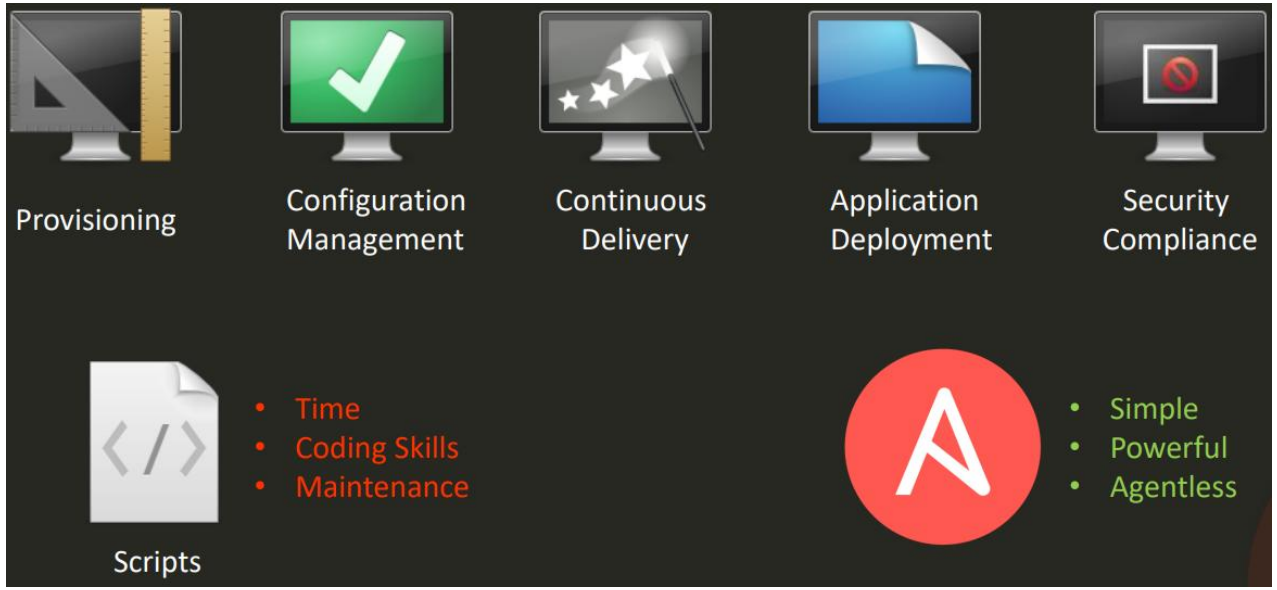
HashiCorp Consul
etcd
Azure Blob
AWS S3
Terraform ENTERPRISE
Google Cloud Storage

TERRAFORM


Terraform & Terragrunt Structure
Terraform Module Repo
ec2
data.tf
examples
basic
volume-attachment
LICENSE
main.tf
main.tf.org
outputs.tf
README.md
variables.tf
vpc
docs
targets.md
terraform.md
examples
basic
LICENSE
main.tf
Makefile
outputs.tf
README.md
README.yaml
variables.tf
Service Repo
api
data.tf
locals.tf
main.tf
output.tf
role.tf
sg.tf
vars.tf
rds
mysql
main.tf
output.tf
sg.tf
vars.tf
postgres
main.tf
output.tf
sg.tf
vars.tf
vpc
main.tf
output.tf
vars.tf
Terragrunt Repo
tbd
globals.tfvars
kor
api
terraform.tfvars
bastion
terraform.tfvars
common
iam
security-groups
locals.tfvars
rds
mysql
postgres
vpc
terraform.tfvars
terraform.tfvars
use1
locals.tfvars
usw1
locals.tfvars

ANSIBLE

Why Ansible



ANSIBLE

Why Ansible

Scripts vs Ansible Playbook

```
#!/bin/bash
# Script to add a user to Linux system
if [ $(id -u) -eq 0 ]; then
   $username=johndoe
   read -s -p "Enter password : " password
   egrep "^$username" /etc/passwd >/dev/null
   if [ $? -eq 0 ]; then
      echo "$username exists!"
      exit 1
   else
      useradd -m -p $password $username
      [ $? -eq 0 ] && echo "User has been added
to system!" || echo "Failed to add a user!"
   fi
fi
```

```
- hosts: all_my_web_servers_in_DR
  tasks:
    - user:
        name: johndoe
```

ANSIBLE

How to install

Control Node

Redhat or CentOS –
```
$ sudo yum install ansible
```

Fedora –
```
$ sudo dnf install ansible
```

Ubuntu –
```
$ sudo apt-get install ansible
```

PIP –
```
$ sudo pip install ansible
```

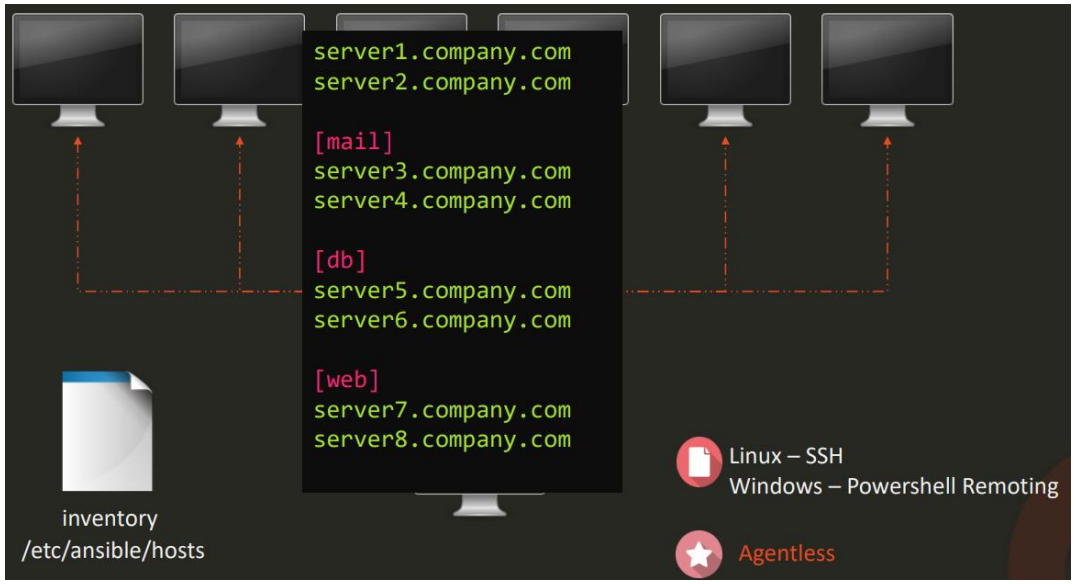
Additional Options:
- Install from source on GIT
- Build RPM yourself

Ansible Control Machine
- Playbooks
- Inventory
- Modules

Control Machine - Linux Only

ANSIBLE

Component - Inventory




```
#Sample Inventory File

server1.company.com                ansible_connection=ssh    ansible_user=root
server2.company.com                ansible_connection=winrm  ansible_user=admin
server3.company.com                ansible_connection=ssh    ansible_ssh_pass=P@#
server4.company.com                ansible_connection=winrm

localhost ansible_connection=localhost
```

Inventory Parameters:

- ansible_connection – ssh/winrm/localhost
- ansible_port – 22/5986
- ansible_user – root/administrator
- ansible_ssh_pass - Password

Security: Ansible Vault

ANSIBLE

Component - Playbook

- Execute a command
- Run a script
- Manage a service (start/stop/restart)
- Line in file

- Playbook – A single YAML file
  - Play – Defines a set of activities (tasks) to be run on hosts
    - Task – An action to be performed on the host
      - Execute a command
      - Run a script
      - Install a package
      - Shutdown/Restart

YAML format

playbook.yml

```
-
  name: Play 1
  hosts: localhost
  tasks:
   - name: Execute command 'date'
     command: date

   - name: Execute script on server
     script: test_script.sh

   - name: Install httpd service
     yum:
      name: httpd
      state: present

   - name: Start web server
     service:
      name: httpd
      state: started
```

- Execute Ansible Playbook
- Syntax: ansible-playbook <playbook file name>

```
ansible-playbook playbook.yml
ansible-playbook --help
```

ANSIBLE

Component - Variable

- Stores information that varies with each host

```
-
  name: Set Firewall Configurations
  hosts: web
  tasks:
  - firewalld:
      service: https
      permanent: true
      state: enabled

  - firewalld:
      port: '{{ http_port }}'/tcp
      permanent: true
      state: disabled

  - firewalld:
      port: '{{ snmp_port }}'/udp
      permanent: true
      state: disabled

  - firewalld:
      source: '{{ inter_ip_range }}'/24
      Zone: internal
      state: enabled
```

```
#Sample Inventory File

Web http_port=     snmp_port=        inter_ip_range=
```

```
#Sample variable File – web.yml

http_port: 8081
snmp_port: 161-162
inter_ip_range: 192.0.2.0
```

{{ }}

Jinja2 Templating

```
✗ source: {{ inter_ip_range }}
✓ source: '{{ inter_ip_range }}'
✓ source: SomeThing{{ inter_ip_range }}SomeThing
```

ANSIBLE

loop:

```
-
  name: Create users
  hosts: localhost
  tasks:
   - user: name='{{ item }}'   state=present
     loop:
       - joe
       - george
       - ravi
       - mani
```

With_*:

```
-
  name: Create users
  hosts: localhost
  tasks:
   - user: name='{{ item }}'   state=present
     with_items:
       - joe
       - george
       - ravi
       - mani
```

ANSIBLE

With_*:

```
with_items
with_file
with_url
with_mongodb
with_dict
with_etcd
with_env
with_filetree
With_ini
With_inventory_hostnames
With_k8s
With_manifold
With_nested
With_nios
With_openshift
With_password
With_pipe
With_rabbitmq
With_redis
With_sequence
With_skydive
With_subelements
With_template
With_together
With_varnames
```

ANSIBLE

Conditional

When

```yaml
---
- name: Install NGINX
  hosts: all
  tasks:
  - name: Install NGINX on Debian
    apt:
      name: nginx
      state: present
    when:  ansible_os_family == “Debian”

  - name: Install NGINX on Redhat
    yum:
      name: nginx
      state: present
    when:  ansible_os_family == “RedHat”
```

Or

```yaml
---
- name: Install NGINX
  hosts: all
  tasks:
  - name: Install NGINX on Debian
    apt:
      name: nginx
      state: present
    when:  ansible_os_family == “Debian”

  - name: Install NGINX on Redhat
    yum:
      name: nginx
      state: present
    when:  ansible_os_family == “RedHat”  or
           ansible_os_family == “SUSE”
```

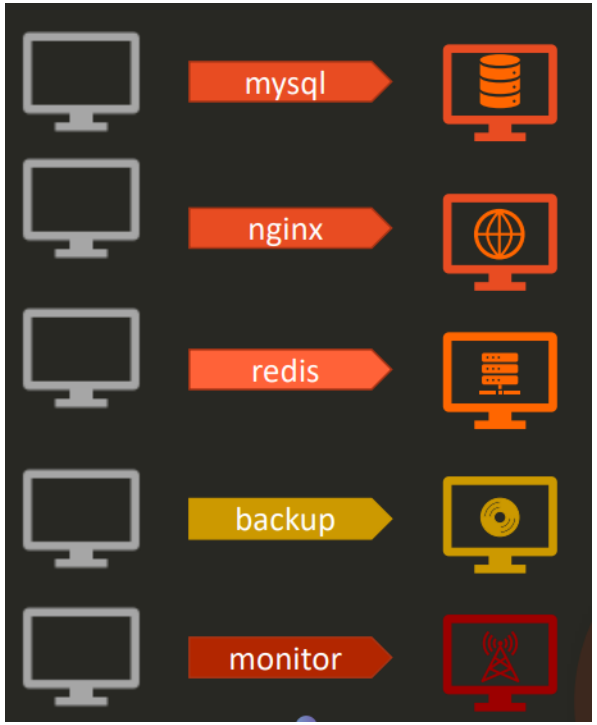
and

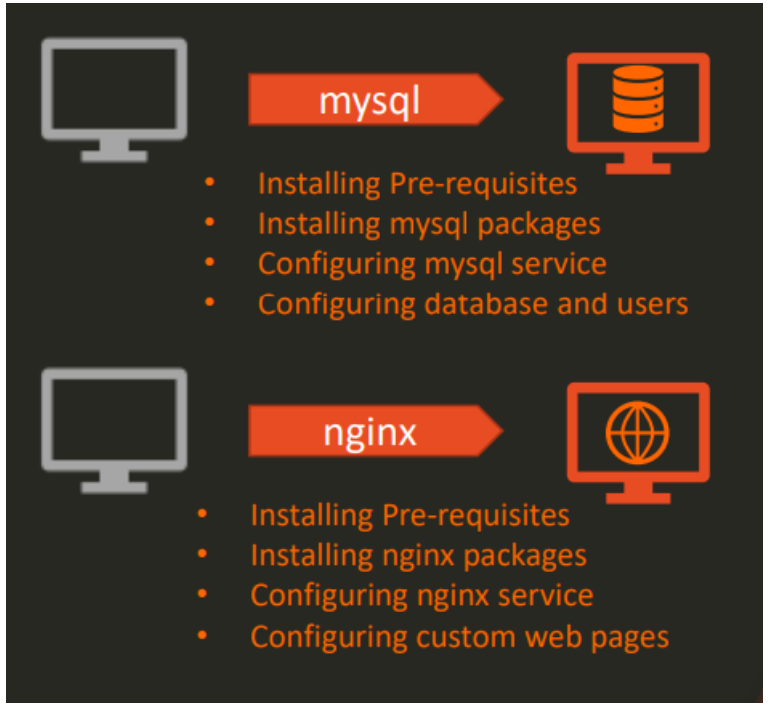
```yaml
---
- name: Install NGINX
  hosts: all
  tasks:
  - name: Install NGINX on Debian
    apt:
      name: nginx
      state: present
    when:  ansible_os_family == “Debian”    and
           ansible_distribution_version == “16.04”

  - name: Install NGINX on Redhat
    yum:
      name: nginx
      state: present
    when:  ansible_os_family == “RedHat”  or
           ansible_os_family == “SUSE”
```

ANSIBLE

Roles





ANSIBLE

Roles