Infrastructure As Code (IaC)

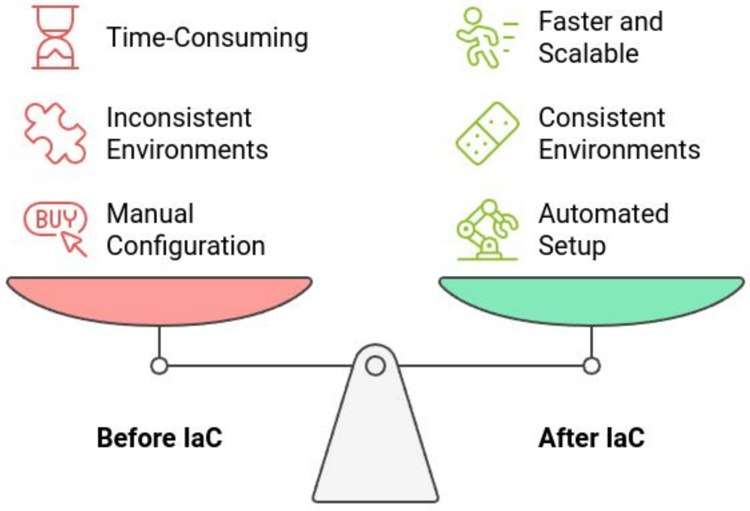
Boris TEABE
boris.teabe@inp-toulouse.fr

Introducing Infrastructure as Code (IaC)

- Infrastructure is resources on which application runs on.
- Infrastructure as Code
  - Defining & Provisioning
  - Management of Infrastructure
  - Descriptive Model & Versioning
- Any **(On-prem or Cloud)** infrastructure with programmatic interface can participate in IaC
- IaC is a key DevOps practice

Introducing Infrastructure as Code (IaC)

- You use a large amount of IaaS resources.
- Your infrastructure is rented from many different providers or platforms.
- You need to make regular adjustments to your infrastructure.
- You need proper documentation of changes made to your infrastructure.
- You want to optimize collaboration between administrators and developers.



IaC transforms infrastructure management from manual to automated.

Introducing Infrastructure as Code (IaC)

Two main approaches

- **Declarative:** focused on the desired end state of a deployment and rely on an interpretation engine to create and configure the actual resources e.g Terraform
- **Imperative:** focus on the actual provisioning process and may reference a file containing a list of settings and configuration values. e.g Ansible

Advantages of IaC

Repeatability

Version Control

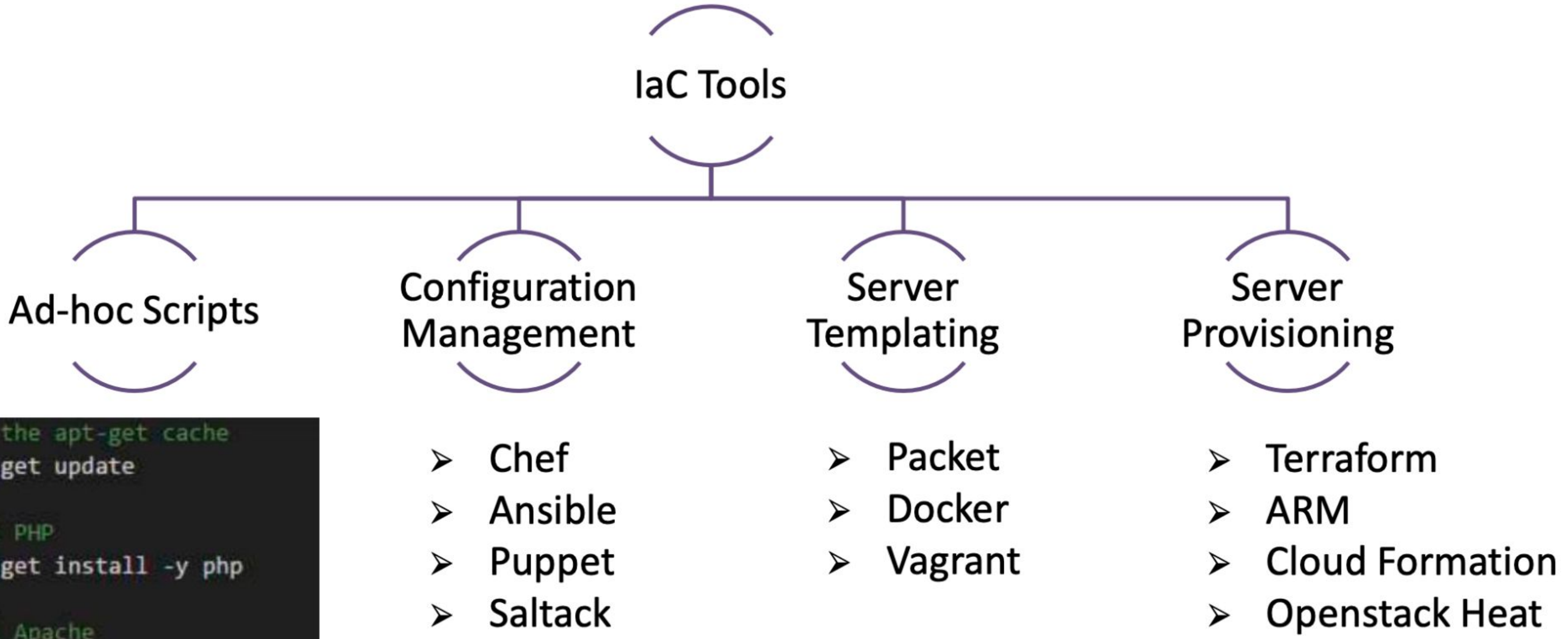
Speed

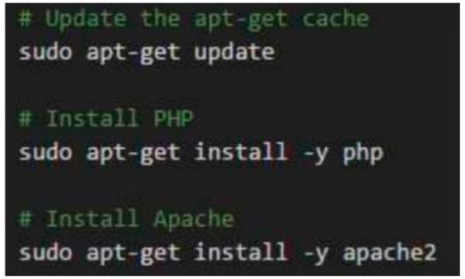
Validation

Documentation

Reuse

Advantages of IaC





```
# Update the apt-get cache
sudo apt-get update

# Install PHP
sudo apt-get install -y php

# Install Apache
sudo apt-get install -y apache2
```

IaC Tools

AWS CloudFormation

puppet

Puppet

Ansible

Terraform

Chef

Azure Resource Manager

SALTSTACK

Saltstack

Vagrant

Pulumi

Pulumi

IaC Tools

Puppet: Popular tool for configuration management

- Client Server Model
- needs agents to be deployed on the target machines before puppet can start managing them
- Resource | Class | Manifest | Catalog | Module
- Best used for Deploying and configuring applications using a pull-based approach.

IaC Tools

Chef: Used for configuration management

- Workstation | Cookbook | Recipe | Server Nodes | Knife
- Best used for Deploying and configuring applications using a pull-based

approach.

IaC Tools

Vagrant: Builds VMs using a workflow.
Specify the base image (called a Box) in a
Vagrantfile along with the steps to configure the VM.

- Vagrant does have Provisioners that allow you to deploy on clouds
- Best used for Creating pre-configured developer VMs within VirtualBox.

IaC Tools

Terraform: Only tool to focus solely on creating, destroying and managing infrastructure components.

- Use the Hashicorp Configuration Language (HCL) to describe the infrastructure resources you need.
- Provider | Provisioners | Modules | Plan Phase | Apply Phase | Destroy
- Best suited for Managing infrastructure resources

IaC Tools

Ansible: Building infrastructure as well as
deploying and configuring applications on top of them.

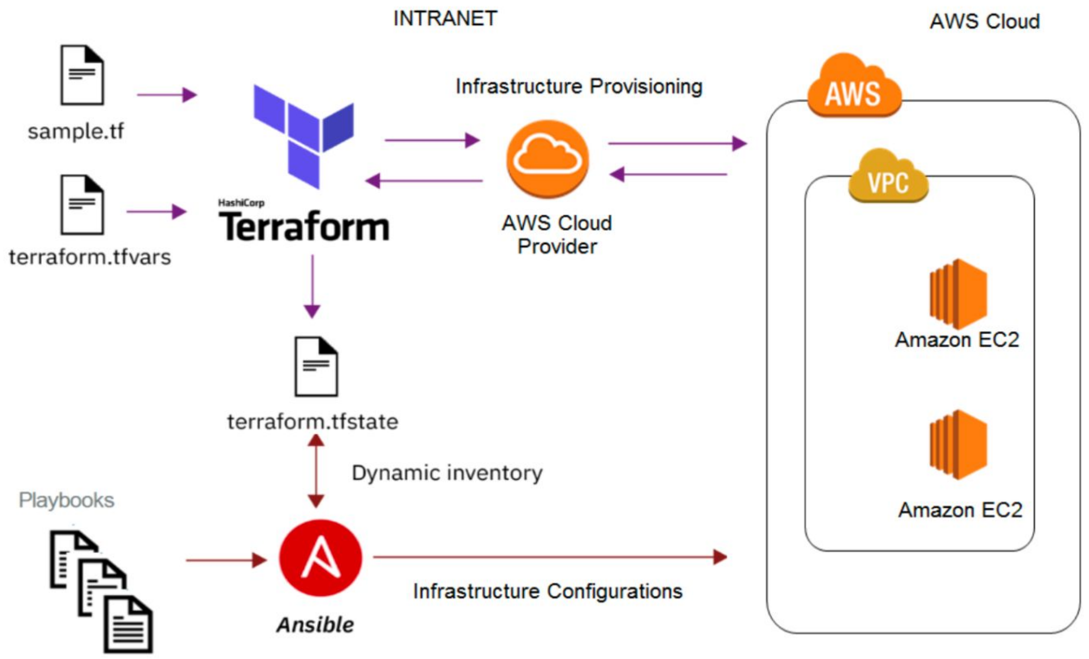
- Ansible is to run in push mode or pull mode.
- Module | Playbook | Role
- Best used for Ad hoc analysis as well as general-purpose, push based, agentless IaC tool

IaC: Provisioning and Configuring

Terraform

&

Ansible

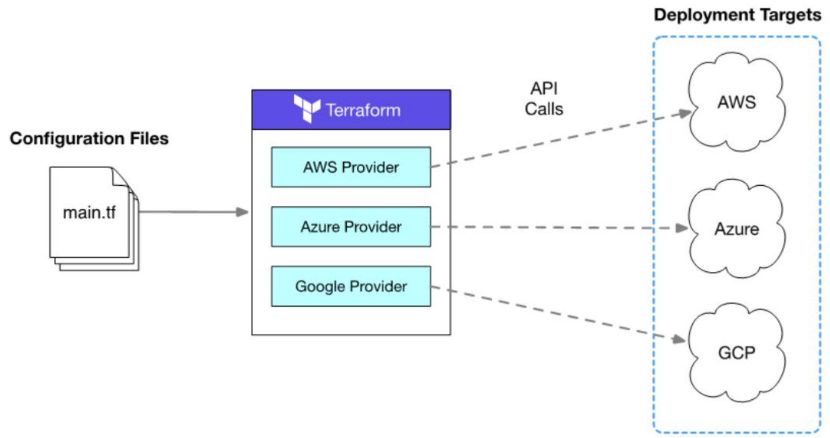


Terraform

- Open Source tool written in GO For Building, Changing & Versioning Infrastructure
- Configuration files HCL or JSON format
- Ability to manage standard Cloud vendor or custom in-house solutions
- Configuration file (.tf) used to define required resources.
- Terraform generated execution plan to reach the desired state.
- Responsible for the creation of server and associated services
- Manages Low-Level & High Level Components.

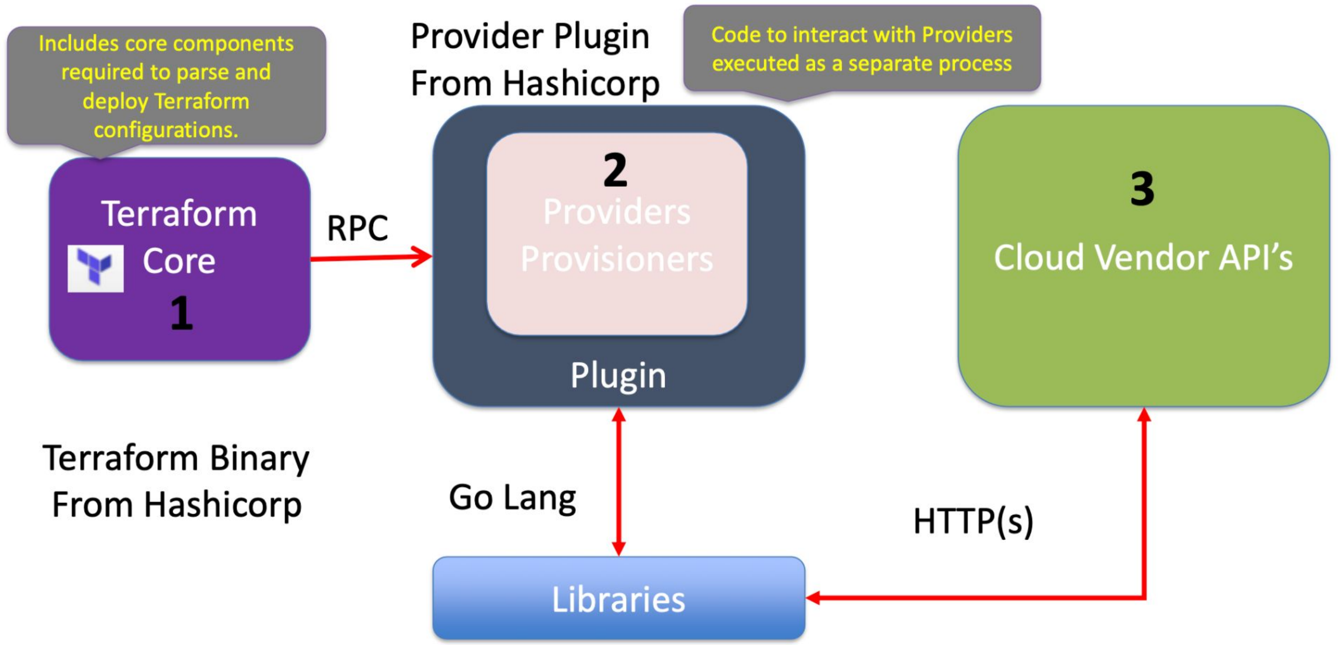
Terraform

- Multi-cloud Support – Almost all cloud providers
- Provide a common tool, process, and language (HashiCorp Configuration Language) to be used across multiple clouds and services



Terraform

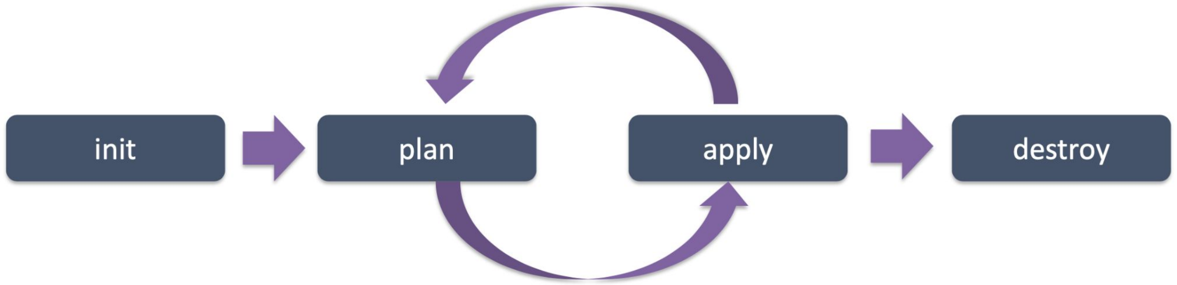
Plugin Based Architecture


Includes core components required to parse and deploy Terraform configurations.
Provider Plugin From Hashicorp
Code to interact with Providers executed as a separate process
Terraform Core
1
RPC
2
Providers
Provisioners
Plugin
3
Cloud Vendor API's
Terraform Binary From Hashicorp
Go Lang
HTTP(s)
Libraries

Terraform

- Provider plugins by Hashicorp
- Automatically installed by providers and are provided via plugins
- Each plugin provides an implementation for a specific service executed as a separate process
- Communicate with the main Terraform binary over an RPC interface. Plugins are built using dynamic libraries
- Each plugin is an independent program
- Main process communicates with the plugin process over HTTP.

Terraform

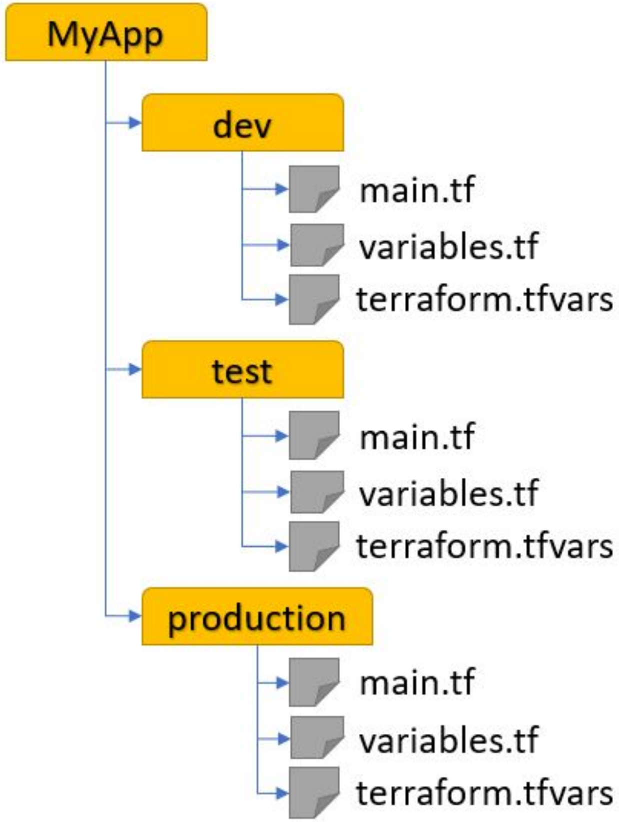


Execution flow

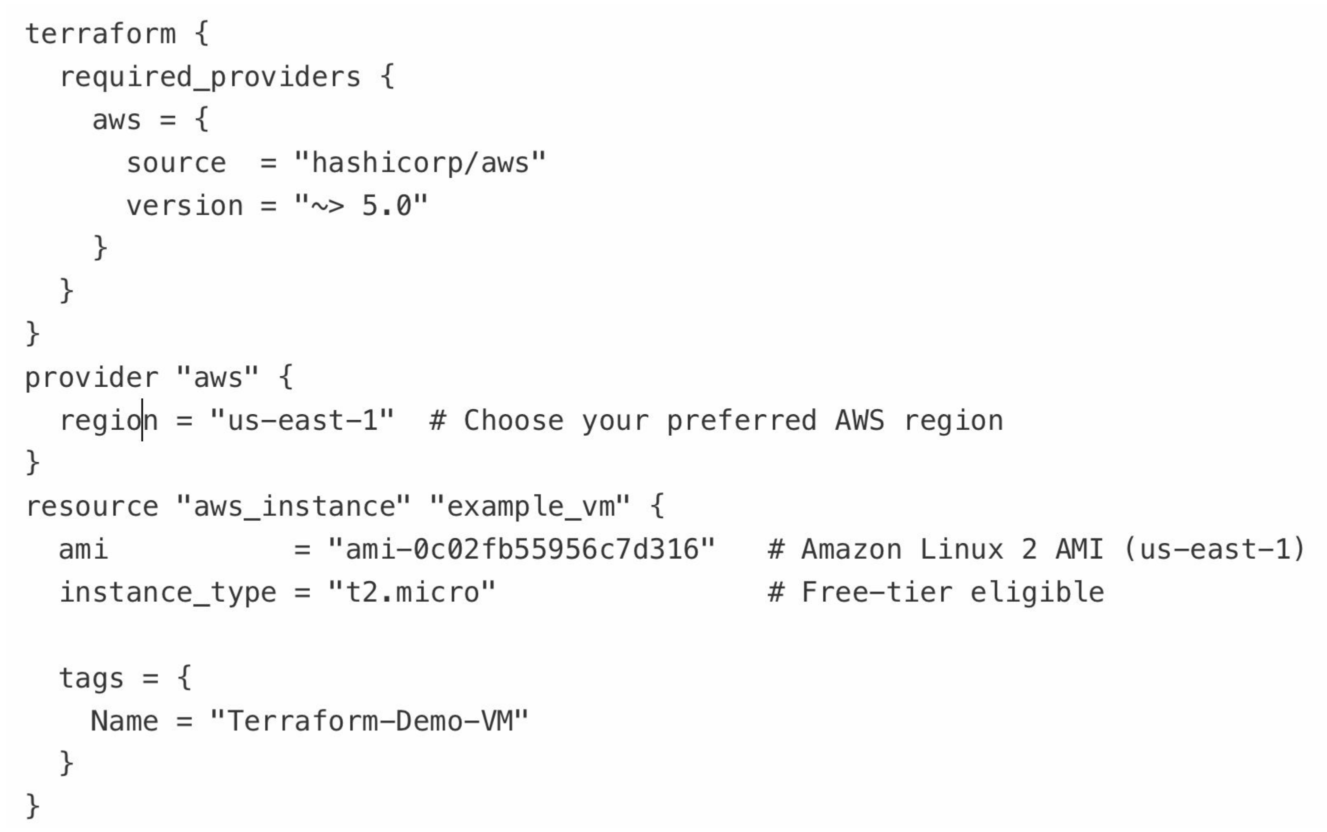
- **Init:** Initialize the (local) Terraform environment.
  Usually executed only once per session.
- **Plan**: Compare the Terraform state with the as-is state in the cloud, build and display an execution plan. This does not change the deployment (read-only).
- **Apply.** The plan from the plan phase. This potentially changes the deployment (read and write Destroy all resources that are governed by this specific terraform environment.

Terraform

Project organisations



Terraform file e.g

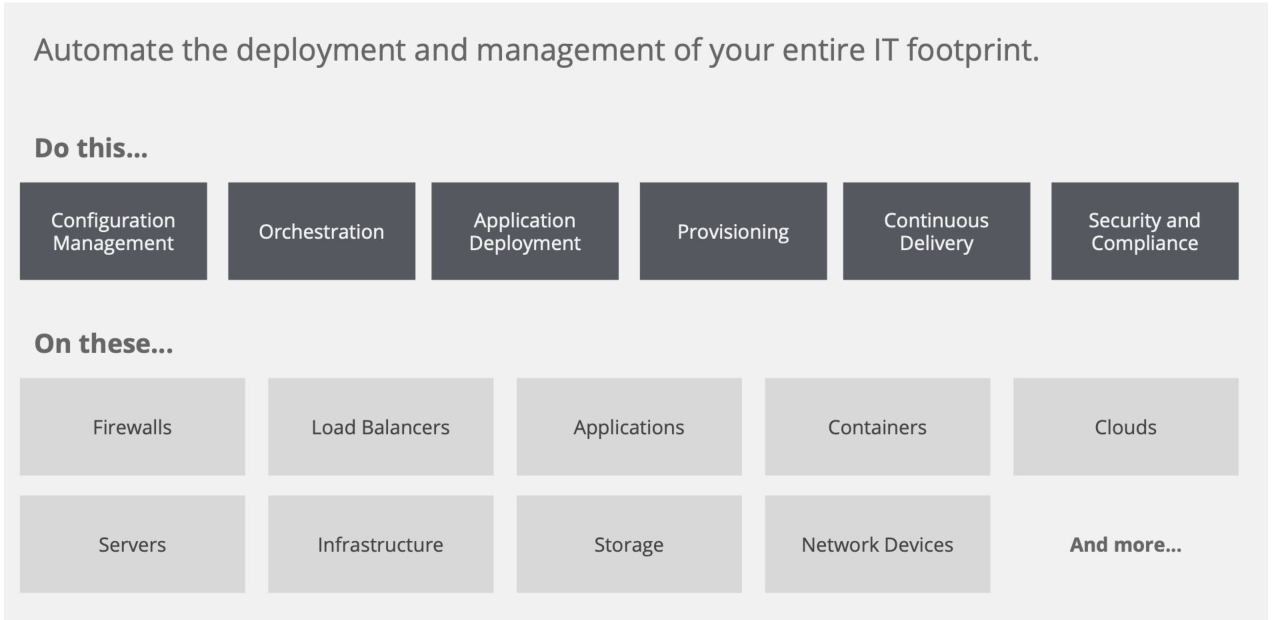


```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}
provider "aws" {
  region = "us-east-1"  # Choose your preferred AWS region
}
resource "aws_instance" "example_vm" {
  ami           = "ami-0c02fb55956c7d316"   # Amazon Linux 2 AMI (us-east-1)
  instance_type = "t2.micro"                # Free-tier eligible

  tags = {
    Name = "Terraform-Demo-VM"
  }
}
```

Ansible

- The Ansible project is an open source community sponsored by Red Hat.
- It’s also a simple automation language that perfectly describes IT application environments in Ansible Playbooks.


Automate the deployment and management of your entire IT footprint.

Do this...

Configuration Management | Orchestration | Application Deployment | Provisioning | Continuous Delivery | Security and Compliance

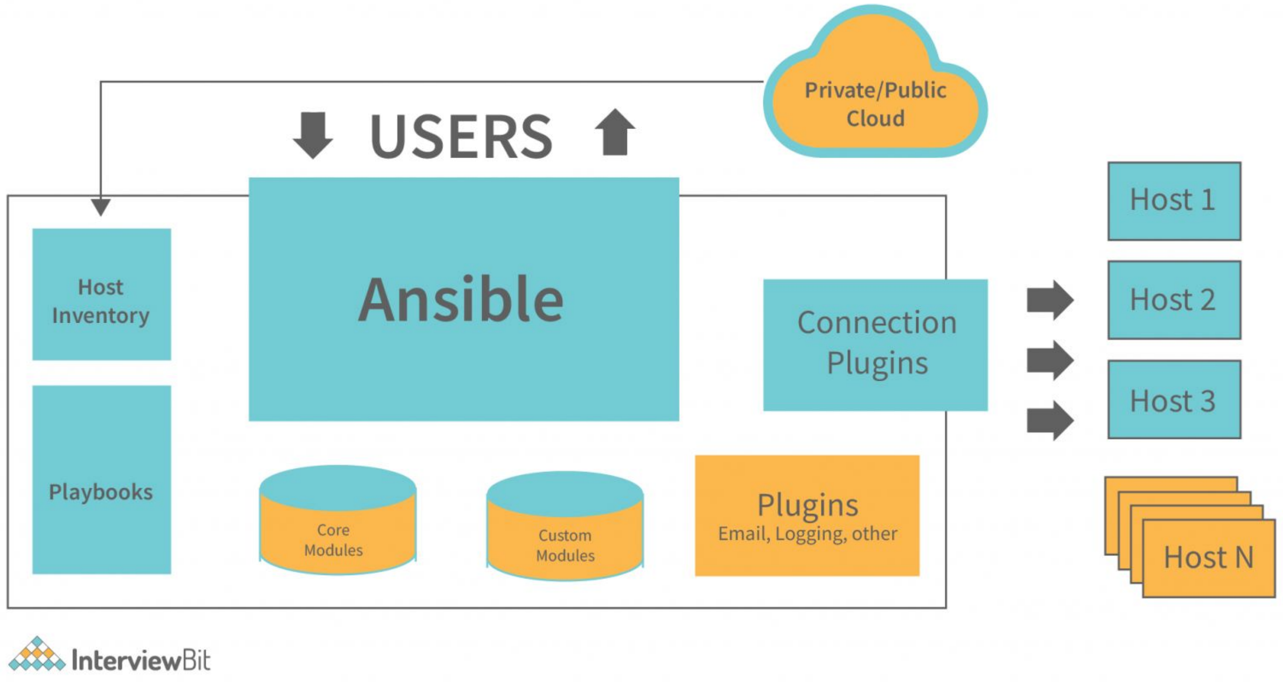
On these...

Firewalls | Load Balancers | Applications | Containers | Clouds

Servers | Infrastructure | Storage | Network Devices | **And more...**

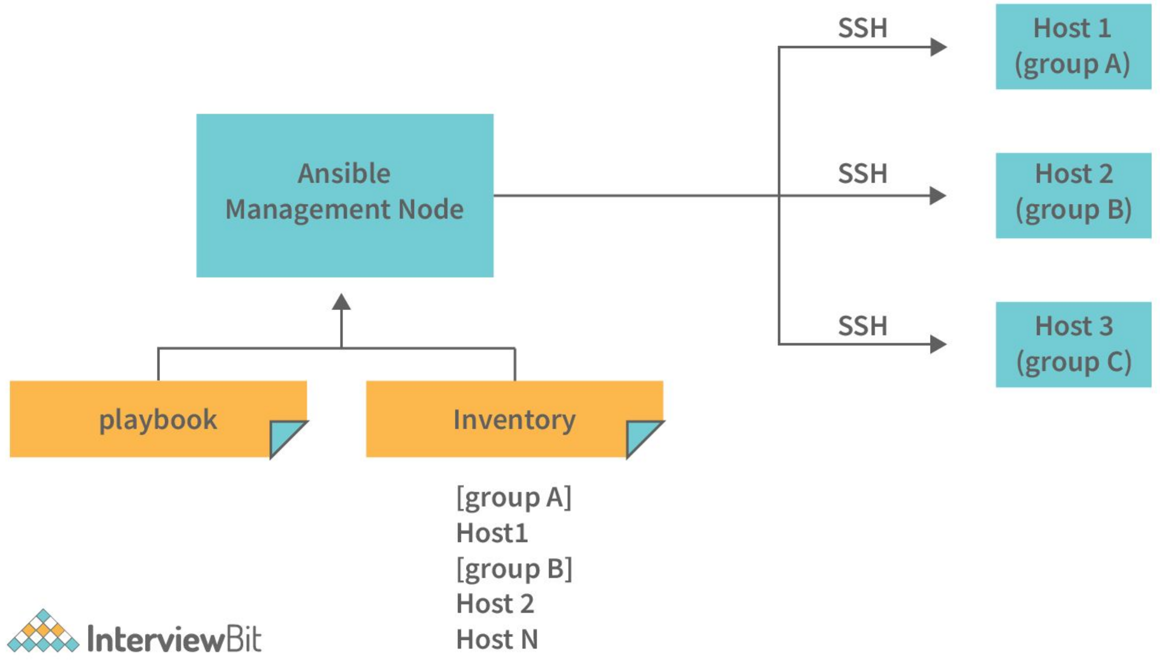
Ansible

Architecture



Ansible

Architecture



Ansible

Exemple

```yaml
---
- name: Configure web server
  hosts: webservers
  become: yes   # run as sudo
  tasks:
    - name: Update package list
      apt:
        update_cache: yes

    - name: Install Nginx
      apt:
        name: nginx
        state: present

    - name: Start Nginx service
      service:
        name: nginx
        state: started
        enabled: yes
```

Ansible

Exemple

```
yum install @base xfsprogs libaio net-tools bind-utils gtk2 libicu xulrunner tcsh
sudo libssh2 expect cairo graphviz iptraf-ng krb5-workstation krb5-libs libpng12
ntp ntpdate nfs-utils lm_sensors rsyslog openssl098e openssl
PackageKit-gtk3-module libcanberra-gtk2 libtool-ltdl xorg-x11-xauth numactl
```

```
- name: install required packages
    yum: state=latest name={{ item }}
    with_items:
      - chrony
      - xfsprogs
      - libaio
      - net-tools
      - bind-utils
         ...
      - numactl
      - tuned-profiles-sap-hana
```