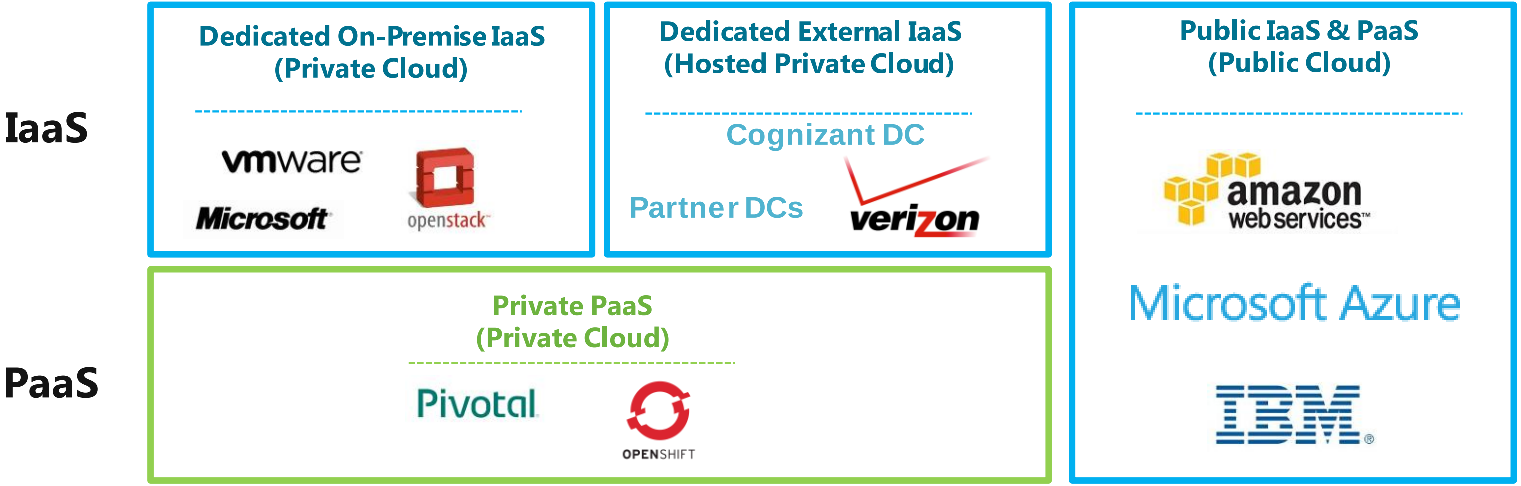
Cognizant

May 2016

Understanding AWS, PCF & Docker

Dileep Hareendran

Introduction to Cloud



IaaS

PaaS

Dedicated On-Premise IaaS (Private Cloud)

vmware

Microsoft

openstack

Dedicated External IaaS (Hosted Private Cloud)

Cognizant DC

Partner DCs

verizon

Public IaaS & PaaS (Public Cloud)

amazon web services

Microsoft Azure

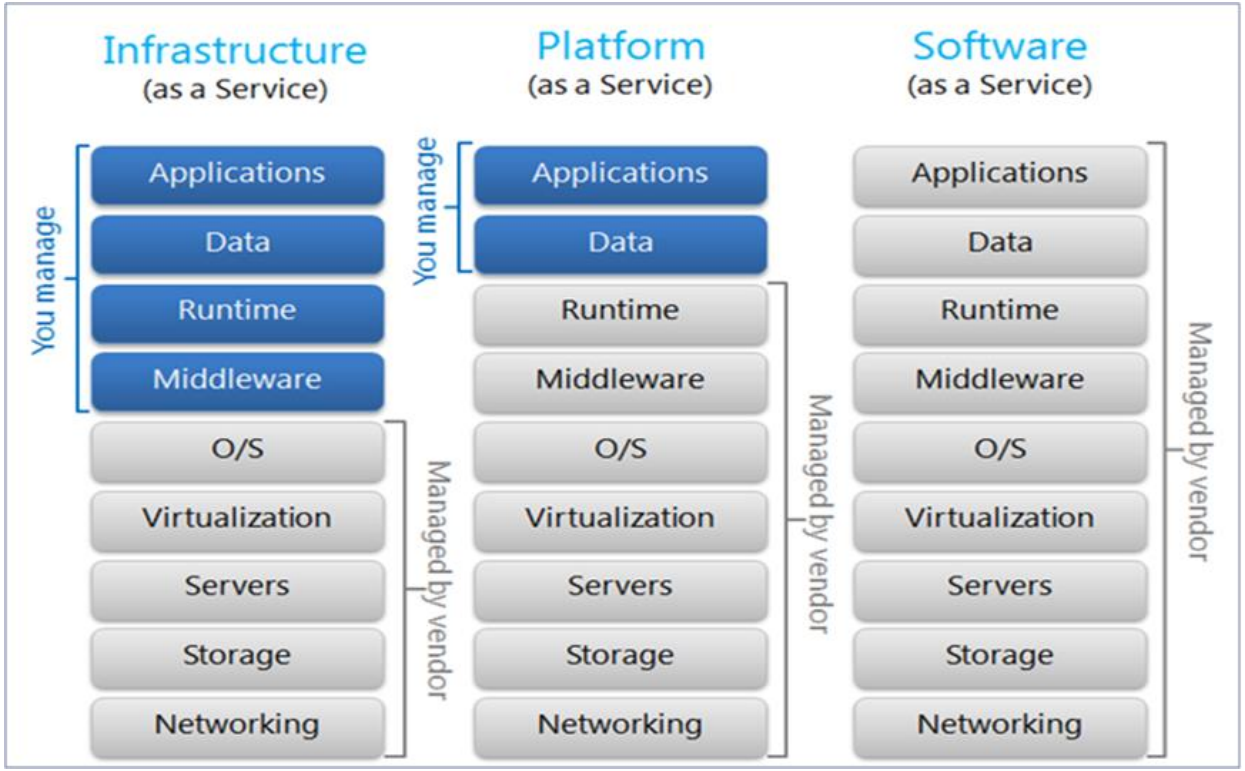
IBM

Private PaaS (Private Cloud)

Pivotal

OPENSHIFT

Cloud Models



Infrastructure
(as a Service)

You manage

Applications

Data

Runtime

Middleware

Managed by vendor

O/S

Virtualization

Servers

Storage

Networking

Platform
(as a Service)

You manage

Applications

Data

Managed by vendor

Runtime

Middleware

O/S

Virtualization

Servers

Storage

Networking

Software
(as a Service)

Managed by vendor

Applications

Data

Runtime

Middleware

O/S

Virtualization

Servers

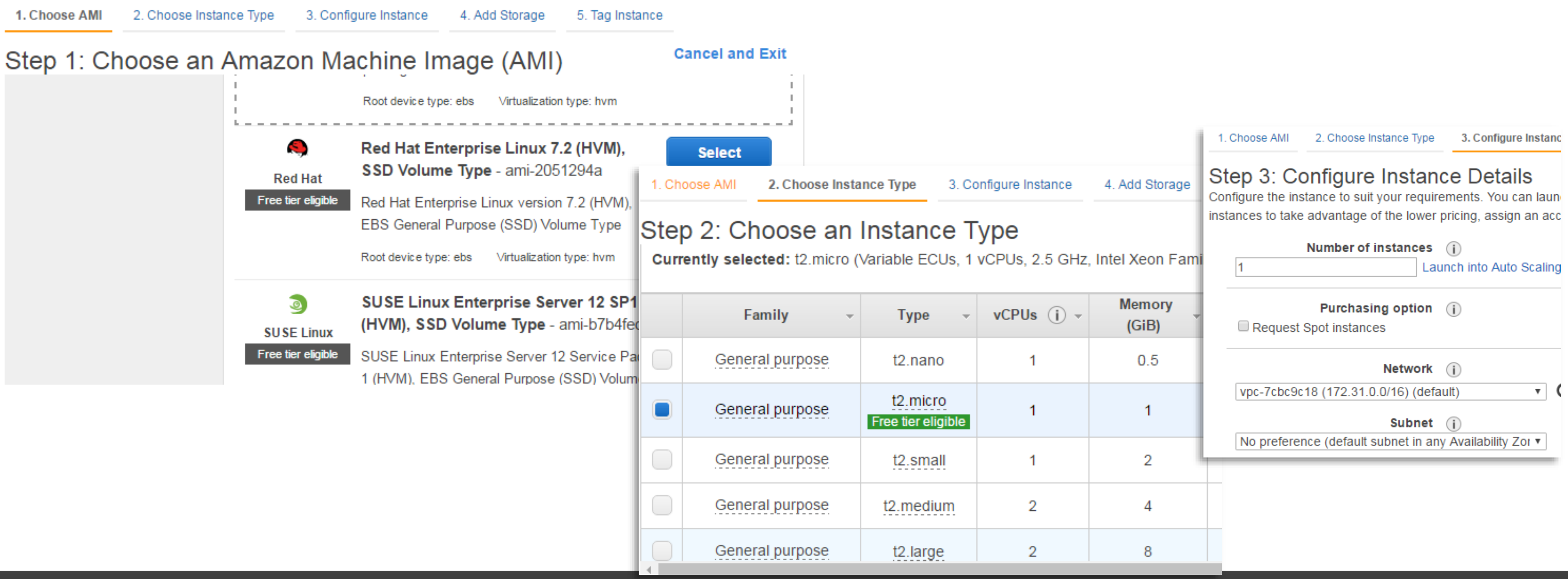
Storage

Networking

AWS IaaS offering

EC2

Creating virtual machine on cloud



1. Choose AMI | 2. Choose Instance Type | 3. Configure Instance | 4. Add Storage | 5. Tag Instance

Step 1: Choose an Amazon Machine Image (AMI)

Cancel and Exit

Root device type: ebs Virtualization type: hvm

Red Hat
Free tier eligible

Red Hat Enterprise Linux 7.2 (HVM), SSD Volume Type - ami-2051294a

Select

Red Hat Enterprise Linux version 7.2 (HVM), EBS General Purpose (SSD) Volume Type

Root device type: ebs Virtualization type: hvm

SUSE Linux
Free tier eligible

SUSE Linux Enterprise Server 12 SP1 (HVM), SSD Volume Type - ami-b7b4fe

SUSE Linux Enterprise Server 12 Service Pa 1 (HVM), EBS General Purpose (SSD) Volum

1. Choose AMI | 2. Choose Instance Type | 3. Configure Instance | 4. Add Storage

Step 2: Choose an Instance Type

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Fami

	Family	Type	vCPUs	Memory (GiB)
□	General purpose	t2.nano	1	0.5
■	General purpose	t2.micro Free tier eligible	1	1
□	General purpose	t2.small	1	2
□	General purpose	t2.medium	2	4
□	General purpose	t2.large	2	8

1. Choose AMI | 2. Choose Instance Type | 3. Configure Instanc

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can laun instances to take advantage of the lower pricing, assign an acc

Number of instances: 1 Launch into Auto Scaling

Purchasing option: □ Request Spot instances

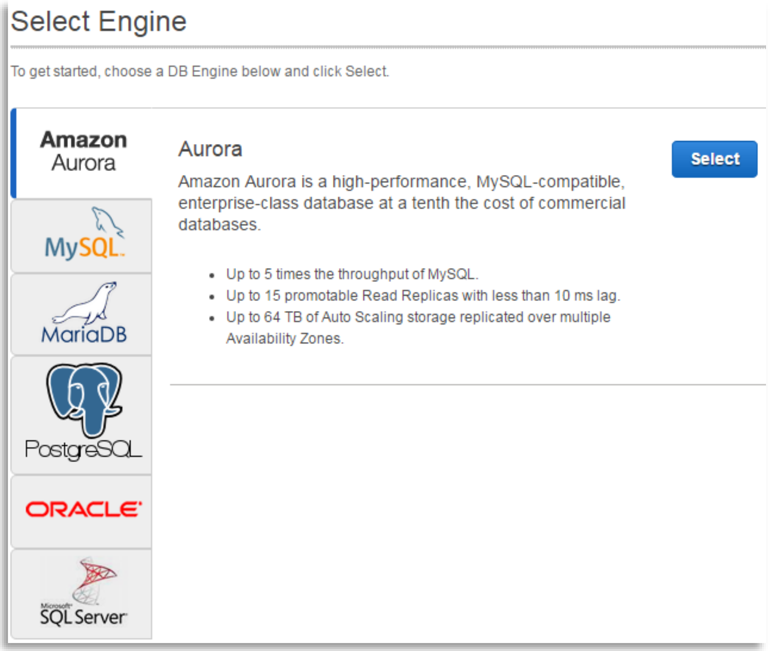
Network: vpc-7cbc9c18 (172.31.0.0/16) (default)

Subnet: No preference (default subnet in any Availability Zo

AWS SaaS offering

RDS

Database as a Service



Select Engine

To get started, choose a DB Engine below and click Select.

Aurora

Amazon Aurora is a high-performance, MySQL-compatible, enterprise-class database at a tenth the cost of commercial databases.

- Up to 5 times the throughput of MySQL.
- Up to 15 promotable Read Replicas with less than 10 ms lag.
- Up to 64 TB of Auto Scaling storage replicated over multiple Availability Zones.

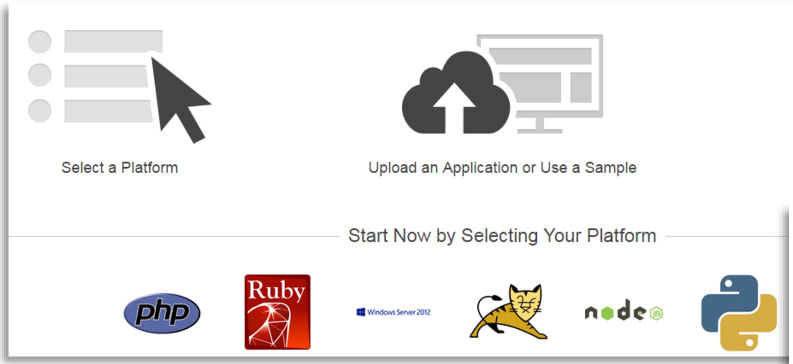
provides **cost-efficient** and **resizable capacity** while managing time-consuming database **administration** tasks

AWS PaaS offering

Elastic Beanstalk

Cloud based application server

deploy, **monitor**, and **scale** an application quickly and easily.

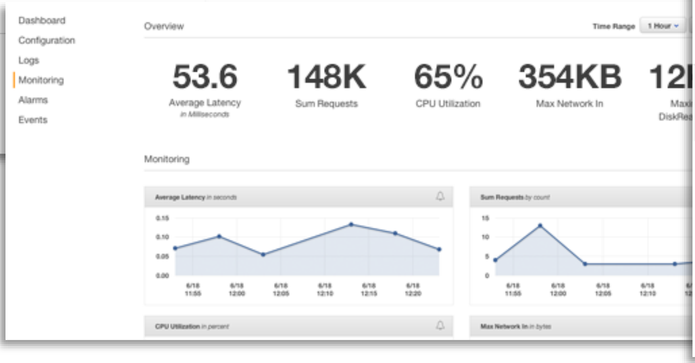


Select a Platform

Upload an Application or Use a Sample

Run it!

Start Now by Selecting Your Platform



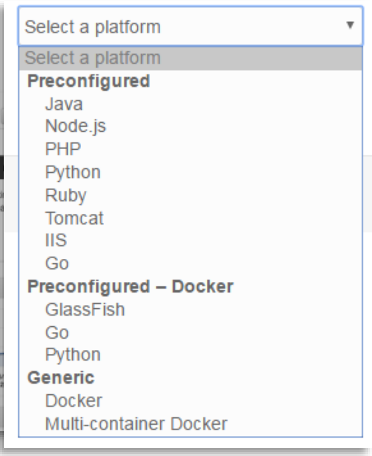
53.6 Average Latency *in Milliseconds*

148K Sum Requests

65% CPU Utilization

354KB Max Network In

12 Maxi DiskRea



Select a platform ▾

Select a platform

Preconfigured
- Java
- Node.js
- PHP
- Python
- Ruby
- Tomcat
- IIS
- Go

Preconfigured – Docker
- GlassFish
- Go
- Python

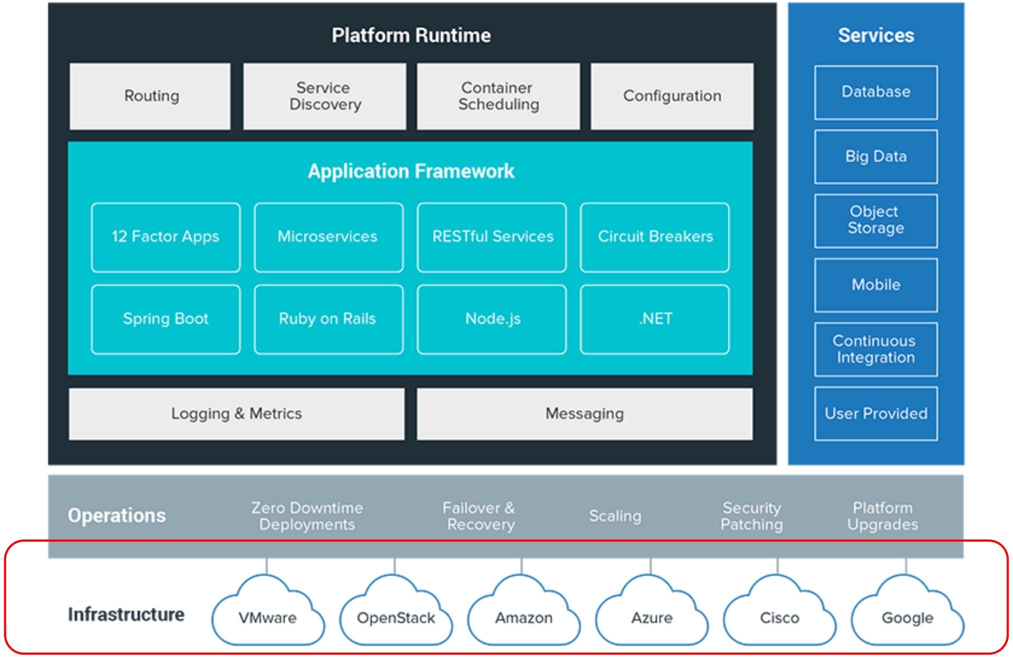
Generic
- Docker
- Multi-container Docker

Pivotal Cloud Foundry

Pivotal

PCF

Cloud independent PaaS



Platform Runtime

Routing

Service Discovery

Container Scheduling

Configuration

Application Framework

12 Factor Apps

Microservices

RESTful Services

Circuit Breakers

Spring Boot

Ruby on Rails

Node.js

.NET

Logging & Metrics

Messaging

Services

Database

Big Data

Object Storage

Mobile

Continuous Integration

User Provided

Operations

Zero Downtime Deployments

Failover & Recovery

Scaling

Security Patching

Platform Upgrades

Infrastructure

VMware

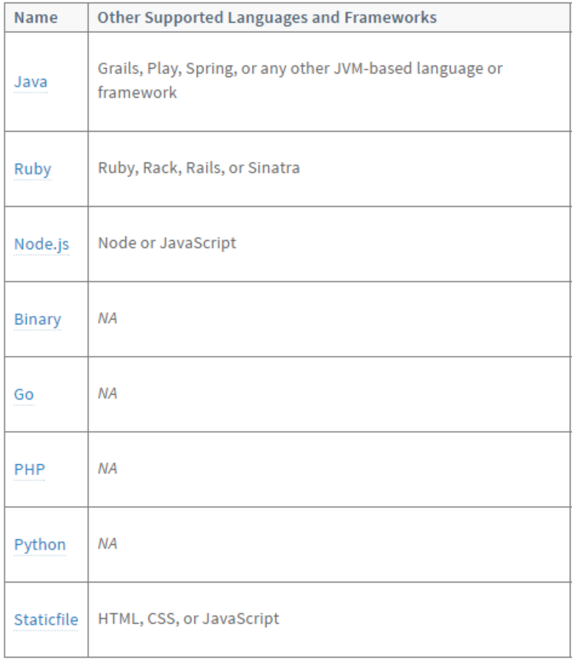
OpenStack

Amazon

Azure

Cisco

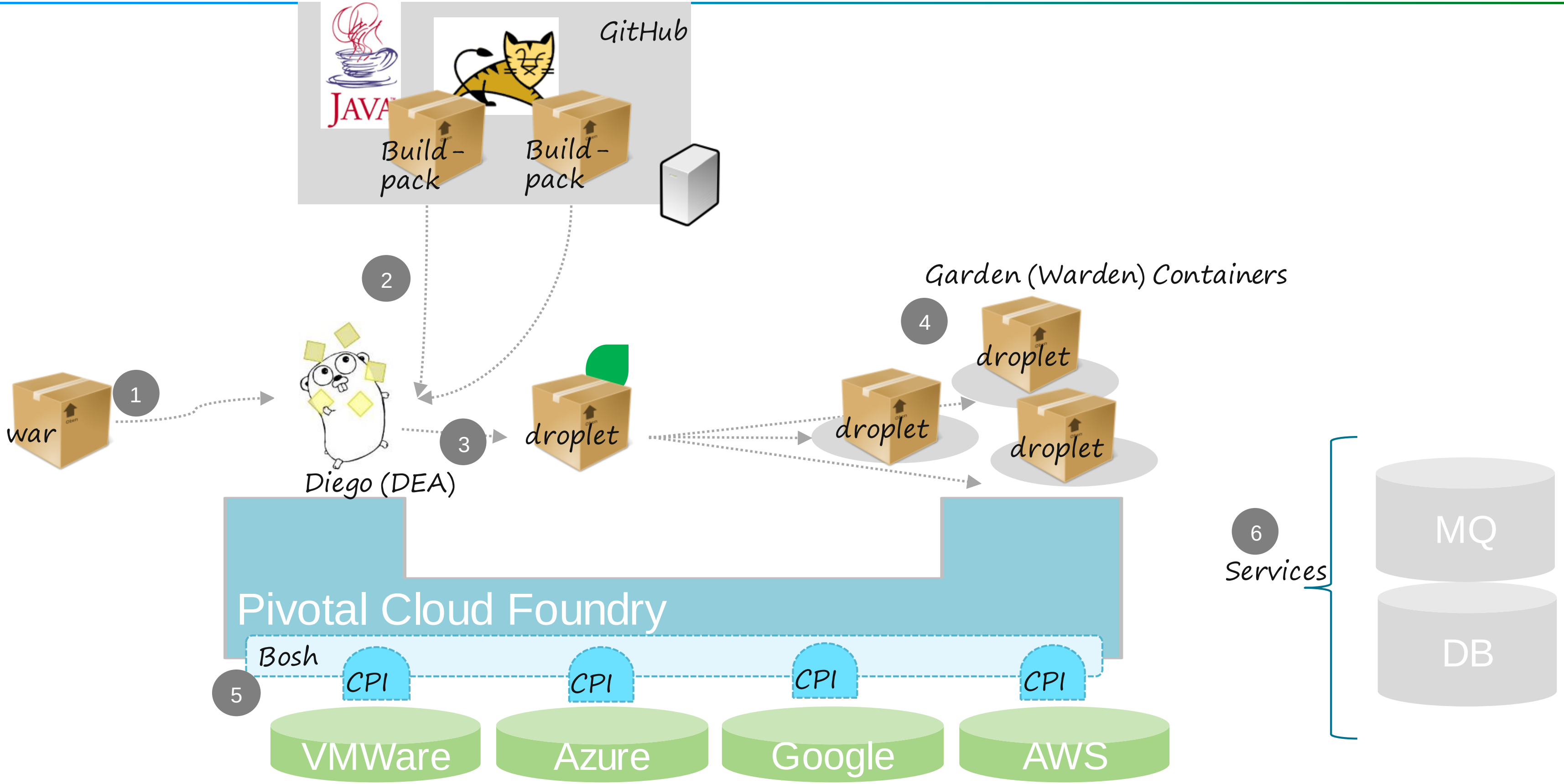
Google



Name	Other Supported Languages and Frameworks
Java	Grails, Play, Spring, or any other JVM-based language or framework
Ruby	Ruby, Rack, Rails, or Sinatra
Node.js	Node or JavaScript
Binary	*NA*
Go	*NA*
PHP	*NA*
Python	*NA*
Staticfile	HTML, CSS, or JavaScript

Other CloudFoundry vendors are IBM BlueMix and GE Predix

How PCF works



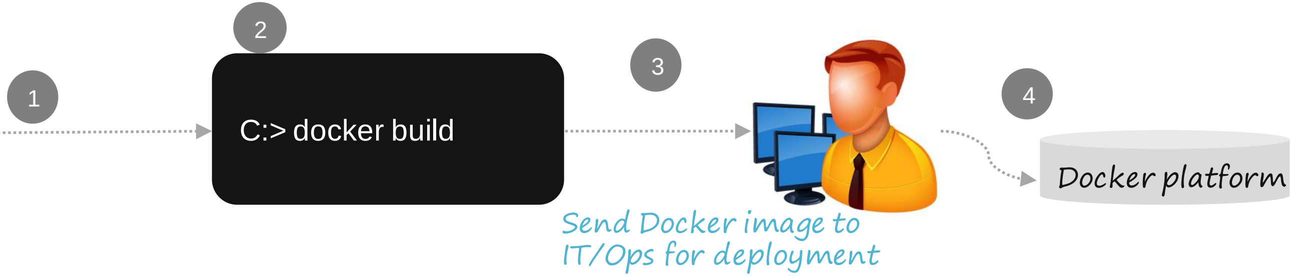
Containers

Docker

Infrastructure as Code

Example dockerfile

1. Linux command to download tomcat.
2. Linux command to copy java deployment file to tomcat
3. Linux command to run tomcat
4. Command to expose port



Docker Platforms

Docker on IaaS/ On Premise

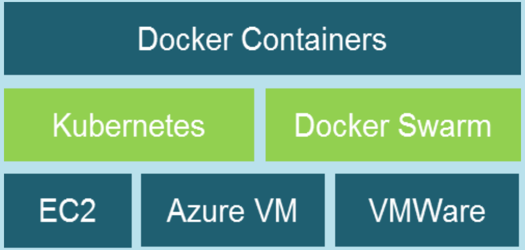
- Docker Containers
- Kubernetes | Docker Swarm
- EC2 | Azure VM | VMWare

Docker on PaaS

- Docker Containers
- Elastic Beanstalk | Google Container Engine | Azure Container Service

Docker Platforms

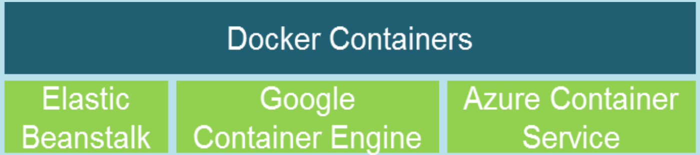
Docker on IaaS/ On Premise


Docker Containers

Kubernetes | Docker Swarm

EC2 | Azure VM | VMWare

Docker on PaaS

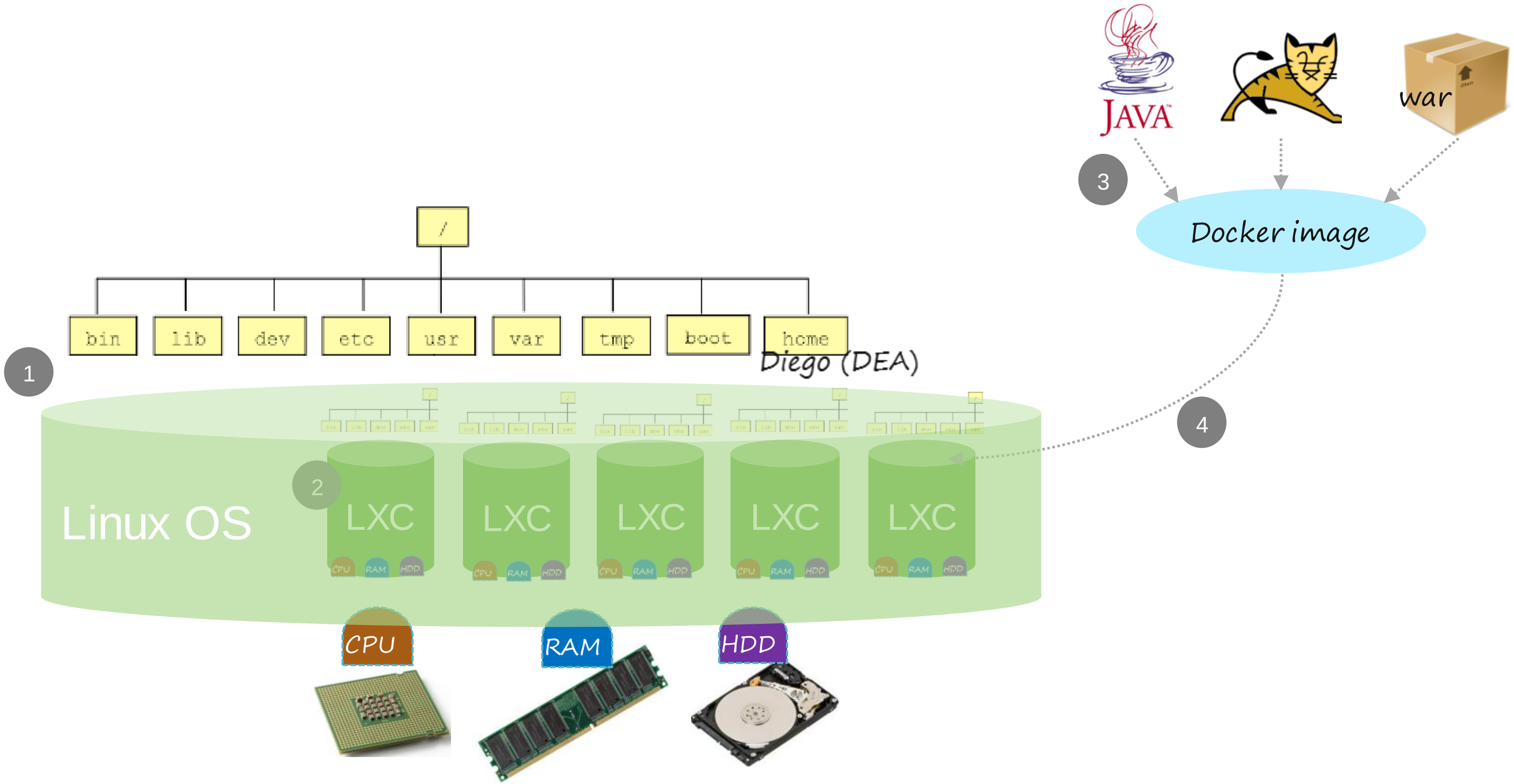

Docker Containers

Elastic Beanstalk | Google Container Engine | Azure Container Service

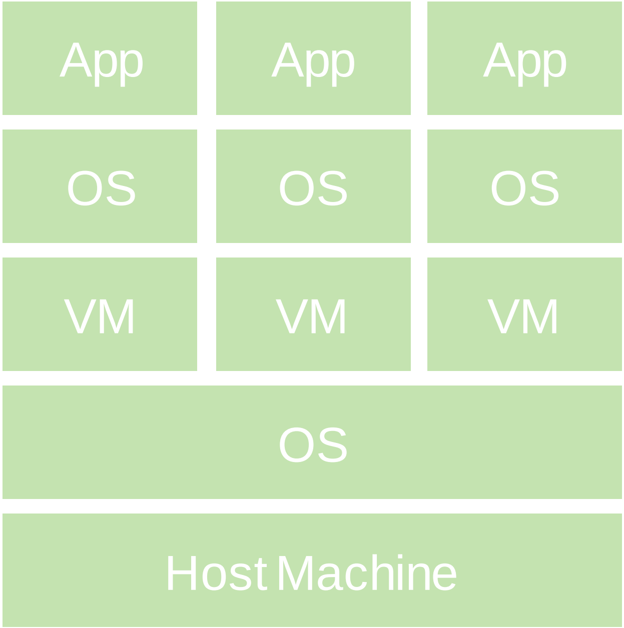
Manages Docker container deployment, communication with underlying layers, cluster management, resource allocation, file system access, scaling etc.

Treats Docker containers just like Java or .Net deployment files. Provides PaaS features.

How Docker Works



Docker vs VMWare



App App App

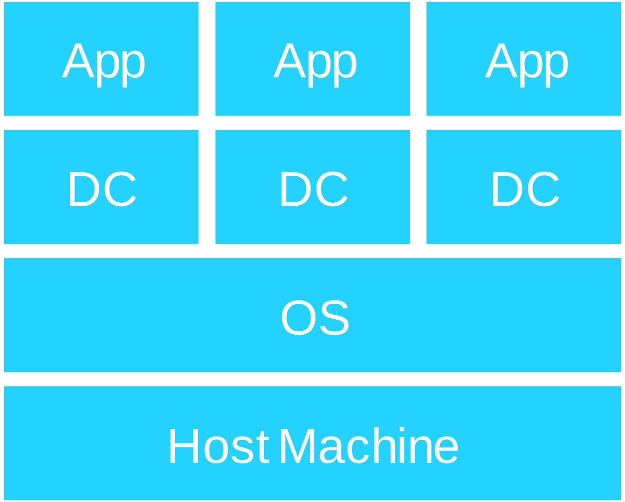
OS OS OS

VM VM VM

OS

Host Machine

VMWare Stack



App App App

DC DC DC

OS

Host Machine

Docker Stack

PCF vs Docker

Pivotal Cloud Foundry

Pros

- Scaling of application is automated
- One centralized API for monitoring application and infrastructure
- Centralized logging
- Going from private cloud/on premise to any cloud doesn’t need learning a new technology
- No lock-in to any cloud provider
- No lock-in to any technology
- No lock-in to any versions

Cons

- Technologies supported is limited to build-packs available. For ex. No support for Perl or Websphere
- Applications have to be *cloud-native* to fully utilize features
- Cannot run complex applications – using ESB or Big Data
- Environment decisions taken by platform. Limited control to developer

Docker

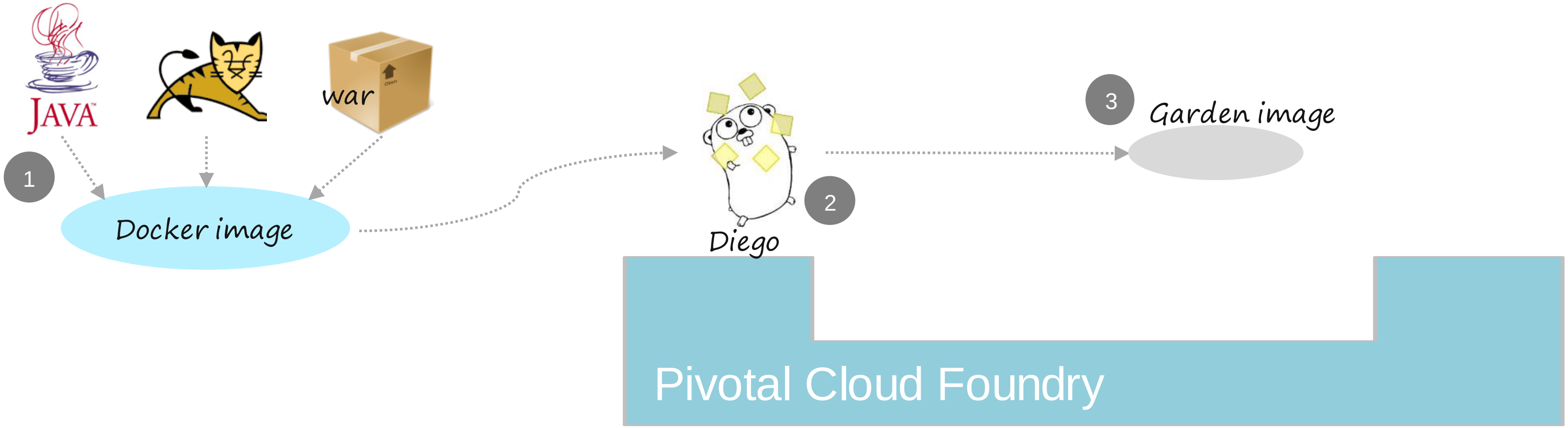
Pros

- Can run any technology as long as it is on Linux
- Any application topology is supported
- Can run legacy or cloud-native
- Doesn’t need a cloud concept.
- Developers can exactly configure the environment in which their application needs to run.

Cons

- Scaling capability depends on the Docker platform used and its cloud support (Kubernetes or Swarm)
- Monitoring support depends on Docker platform used.
- No centralized logging concept.

Docker + PCF



JAVA
war
1
Docker image
Diego
2
3
Garden image
Pivotal Cloud Foundry