Introduction to DevOps

Sonika Rathi

Assistant Professor
BITS Pilani

Review CS#9

Configuration Management

- Introduction to Configuration [Infrastructure Concept]
- Managing Application Configuration
- Managing Environments
- Infrastructure as code
- Server Provisioning
- Automating and Managing Server Provisioning
- Configuration management tools- Chef, Puppet
- Managing on-demand infrastructure
- Auto Scaling

Agenda

Configuration Management Tools and Virtualization & Containerization

- CM – Tools
    - Chef Configuration Management Tool
    - Puppet & Puppet Enterprise
    - Ansible
- Pre Virtualization World
- Virtualization
- Hypervisors based Vs Containerization
- Docker

Configuration Management Tools

Configuration Management Tool Provides

What they Offer in common

- Quick Provisioning of New Servers
- Quick Recovery from Critical Events
- No More Snowflake Servers [Complex or Unique Servers]
- Version Control for the Server Environment
- Easy Replication of Environments

Offer in common for Servers

- Automation Framework
- Idempotent Behavior
- Templating System
- Extensibility

Configuration Management Tool

Chef

Why

Automate the infrastructure provisioning

Infrastructure as Code

Configuration as Code

Ruby DSL language

Solution is compatible with Physical, Virtual, and Cloud Machines

Capability to get integrated with any of the cloud technology

Open Source

Who

Facebook

Etsy

Cheezburger

Indiegogo

Chef

Chef Components

Chef Server

A hub for configuration data
Chef server stores cookbooks, the policies that are applied to nodes
And metadata that describes each registered node that is managed by Chef

Actual Server (Node)

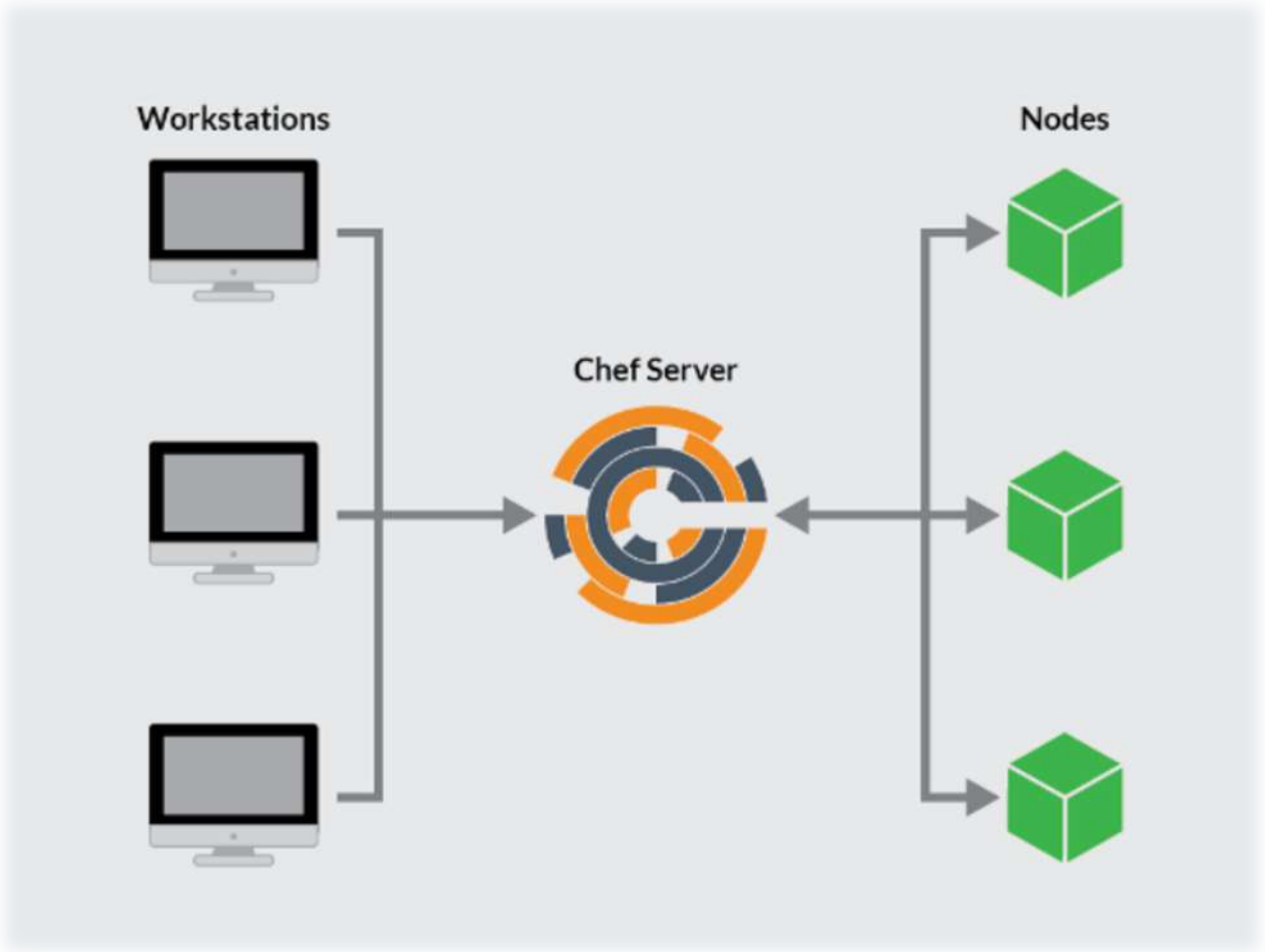
These are the machines that are managed by Chef
The Chef client is installed on each node
It is used to configure the node to its desired state

Chef Workstation

It is the location where users interact with Chef
At Chef Workstation, Users can author and test cookbooks

Chef

Chef Components the Workflow



Chef Terms

Cookbooks

Fundamental unit of configuration and policy distribution

Defines a scenario and contains everything that is required to support that scenario

The chef-client uses Ruby as its reference language for creating cookbooks and defining recipes, with an extended DSL for specific resources

~/chef-repo/cookbooks/lamp_stack/apache.rb

```ruby
package "apache2" do
  action :install
end
```

~/chef-repo/cookbooks/lamp_stack/apache.rb

```ruby
service "apache2" do
  action [:enable, :start]
end
```

Chef Terms

Chef Supermarket

It is the location in which community cookbooks are shared and managed

Cookbooks that are part of the Chef Supermarket may be used by any Chef user

How community cookbooks are used varies from organization to organization

The chef-repo

It is the repository structure in which cookbooks are authored, tested, and maintained and from which policy is uploaded to the Chef server

The chef-repo should be synchronized with a version control system such as Git and then managed

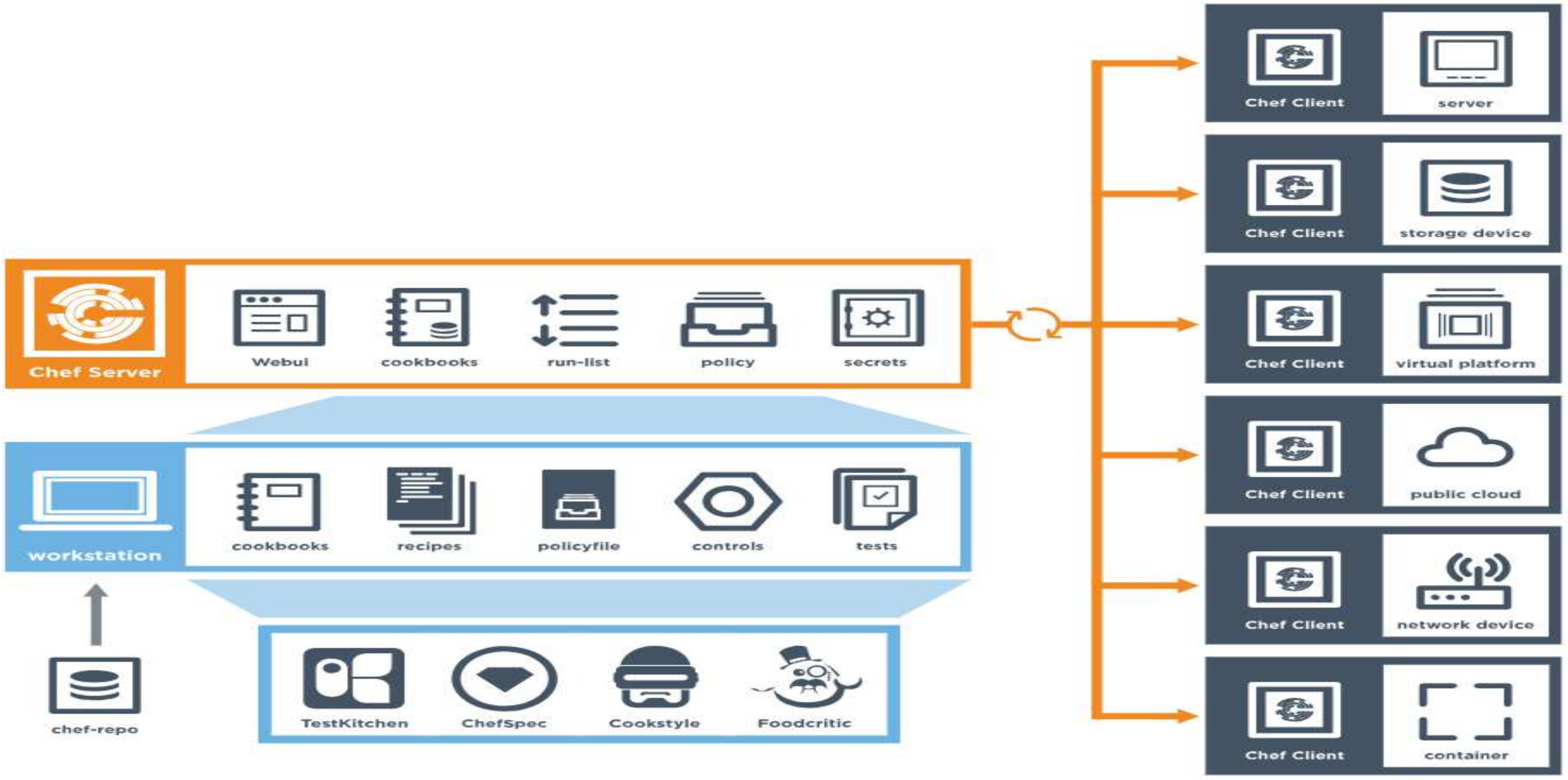
Chef

Workstation Components and Tools

The Chef Development Kit	• It is a package that contains everything that is needed to start using Chef Chef-client, chef and knife command line tools
Chef	• Chef is the command-line tool • Chef is used to work with items in a chef-repo
Knife	• knife is also the command-line tool • It interact with nodes or work with objects on the Chef server
Test Kitchen	• It is a testing harness for rapid validation of Chef code
InSpec	• It is a Chef’s open source security & compliance automation framework
chef-run	• It is a tool for running ad-hoc tasks

Chef

Lets Summarize



Configuration Management Tool

Puppet

Why

Automate the infrastructure provisioning

Infrastructure as Code

Configuration as Code

Puppet gives you an automatic way to inspect, deliver, operate and future-proof all of your infrastructure and deliver all of your applications faster

Open Source

Puppet

Puppet Components

Puppet Master

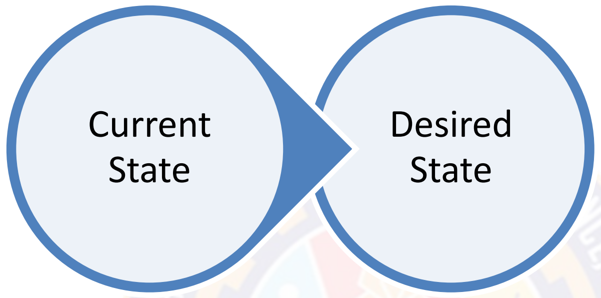
- Puppet master is a Ruby application
- It compiles configurations for any number of Puppet agent nodes
- Puppet Server is an application that runs on the Java Virtual Machine (JVM)
- Supported platforms:
  - Red Hat Enterprise Linux
  - Fedora
  - Debian
  - and Ubuntu

Puppet Agent

- Managed nodes run the Puppet agent application, as a background service
- Periodically, the agent sends facts to a master and requests a catalog
- The master compiles the catalog using several sources of information, and returns the catalog to the agent

Puppet Terms

Catalog

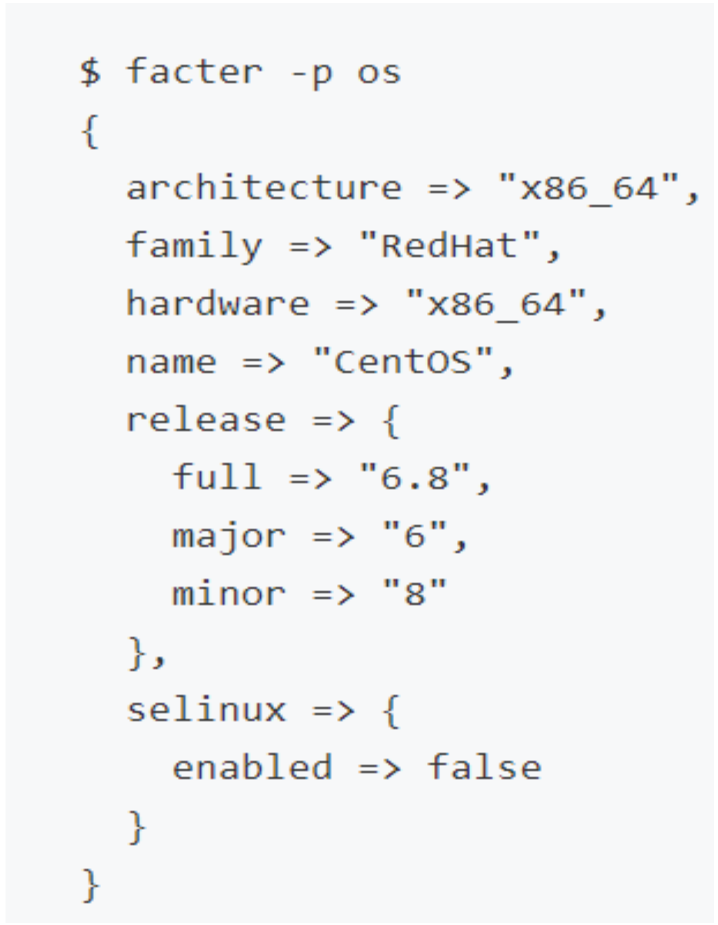


- The catalog describes the desired state for each resource that should be managed
- It also specify dependency information for resources that should be managed in a certain order
- The agent uses a document called a catalog to configure; which it downloads from master
- For each resource under management, the catalog describes its desired state
- The ”puppet apply command” compiles its own catalog

Puppet Terms

Facter

- Facter is the cross-platform system profiling library in Puppet
- It discovers and reports per-node facts, which are available in your Puppet manifests as variables
- Before requesting a catalog, the agent uses Facter to collect system information



```
$ facter -p os
{
  architecture => "x86_64",
  family => "RedHat",
  hardware => "x86_64",
  name => "CentOS",
  release => {
    full => "6.8",
    major => "6",
    minor => "8"
  },
  selinux => {
    enabled => false
  }
}
```

Puppet Terms

Resources

- Puppet code is composed primarily of resource declarations
- A resource describes something about the state of the system
- Such as a certain user or file should exist
- Here is an example of a user resource declaration

Resource Example:

```
user { sonika':
 ensure     => present,
 uid        => '1000',
 gid        => '1000',
 shell      => '/bin/bash',
 home       => '/home/sonika’
}
```

Puppet Terms

Manifests

- Manifests are Puppet programs
- Manifests are composed of puppet code
- And their filenames use the .pp extension
- The default main manifest in Puppet installed via apt is /etc/puppet/manifests/site.pp

Manifest: Example

```
file { "/var/bits/devops":
        ensure => "present",
        owner => "sonika",
        group => "bits",
        mode => "664",
        content => "This is a test file created
using puppet.
                Puppet is really cool",
}
```

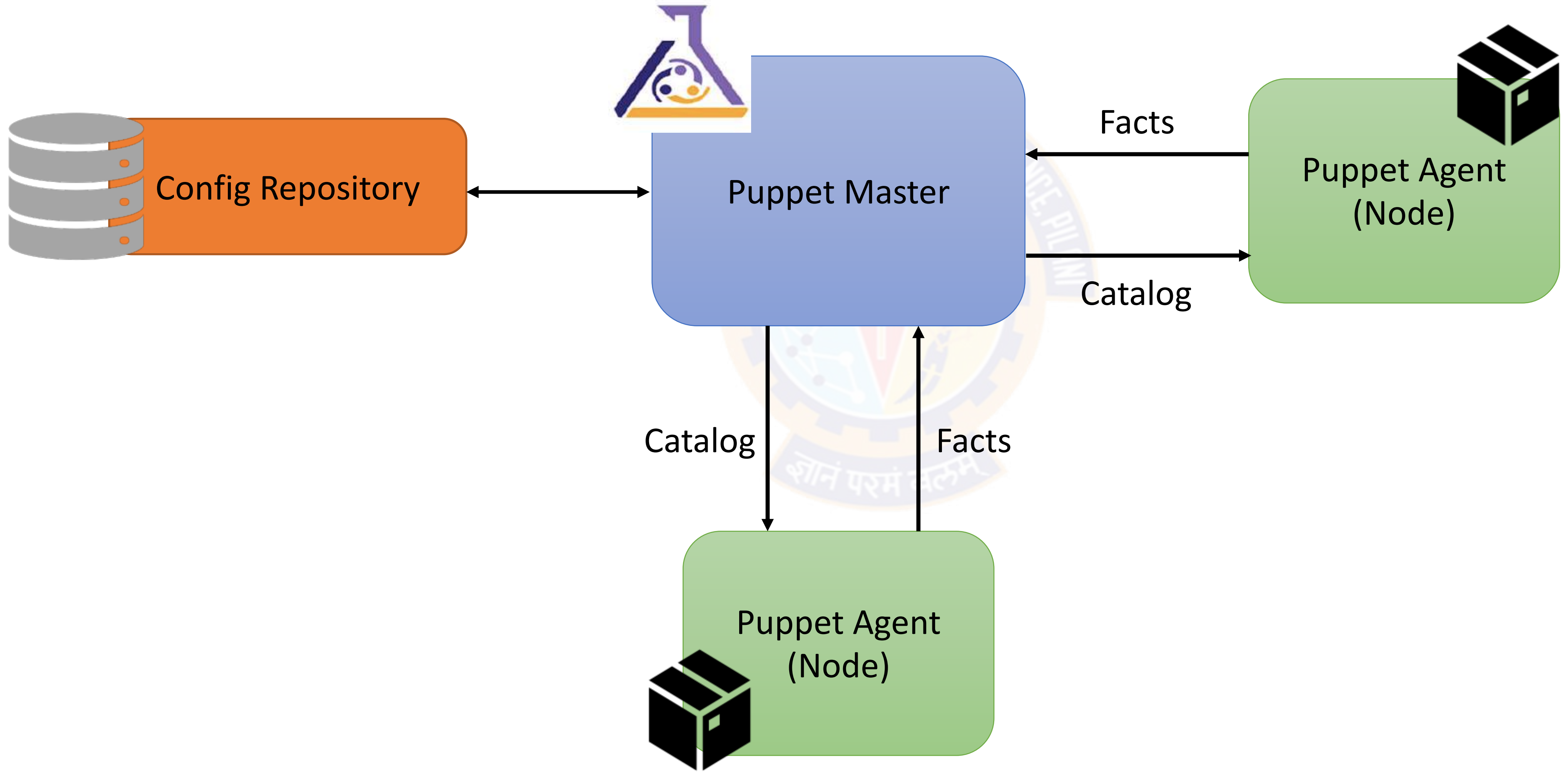
Puppet Terms

The Puppet Forge

- Puppet Forge was launched in 2012
- The Puppet Forge is one of the most utilized resources in the Puppet world
- It is a repository of modules written both by Puppet and by the Puppet user community
- These modules solve a wide variety of problems, and using them can save you time and effort
- Puppet Forge has over 5,500 modules published
- The puppet module tool provides a command line interface for managing modules from the Forge

Puppet Architecture

Client-Server Architecture & Working



Configuration Management Tool

Ansible

Why

Agentless

- It uses no agents and no additional custom security infrastructure, so it's easy to deploy

High level interaction

- It uses a very simple language (YAML, in the form of Ansible Playbooks)

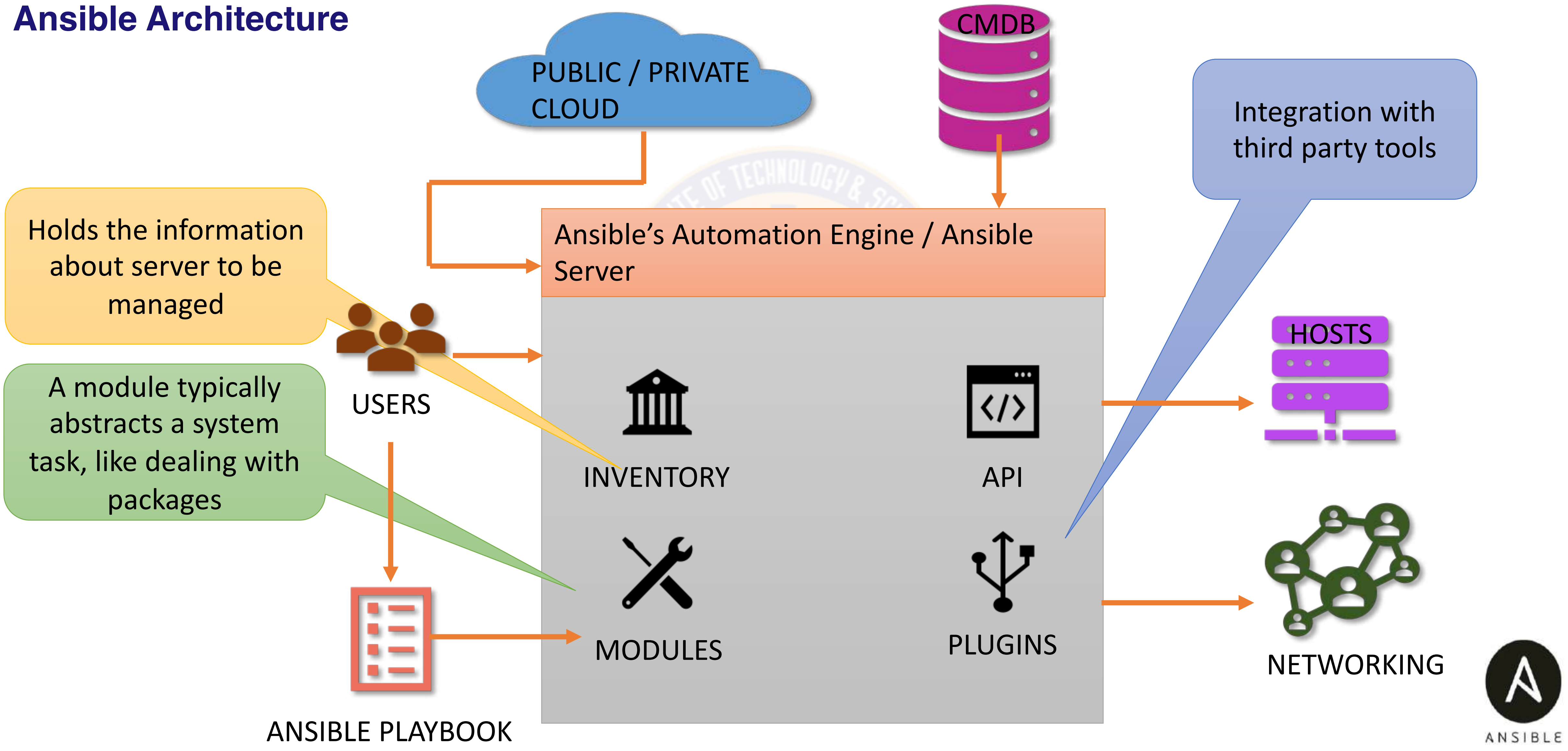
Multi-tier deployment

- Ansible models your IT infrastructure by describing how all of your systems inter-relate, rather than just managing one system at a time

ANSIBLE

Ansible

Ansible Architecture


PUBLIC / PRIVATE CLOUD
CMDB
Integration with third party tools
Holds the information about server to be managed
Ansible's Automation Engine / Ansible Server
USERS
HOSTS
A module typically abstracts a system task, like dealing with packages
INVENTORY
API
MODULES
PLUGINS
NETWORKING
ANSIBLE PLAYBOOK
ANSIBLE

Ansible

Playbook, Facts, Handlers

- The entry point for Ansible provisioning, where the automation is defined through tasks using YAML format
  - Play: A provisioning executed from start to finish is called a play. In simple words, execution of a playbook is called a play
  - Task: A block that defines a single procedure to be executed, e.g. Install a package
- Facts: Global variables containing information about the system, like network interfaces or operating system
- Handlers: Used to trigger service status changes, like restarting or stopping a service

Ansible

Examples

```
---
[webservers]
www1.example.com
www2.example.com

[dbservers]
db0.example.com
db1.example.com
```

Example of Inventory

```
---
- hosts: webservers
serial: 5 # update 5 machines at a time
roles:
- common
- webapp

- hosts: content_servers
roles:
- common
- content
```

Example of Playbook

```
---
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum:
      name: httpd
      state: latest
  - name: write the apache config file
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running
    service:
      name: httpd
      state: started
  handlers:
    - name: restart apache
      service:
        name: httpd
        state: restarted
```

Example: Playbook of webapp

Puppet Vs Ansible

Lets Compare

Puppet

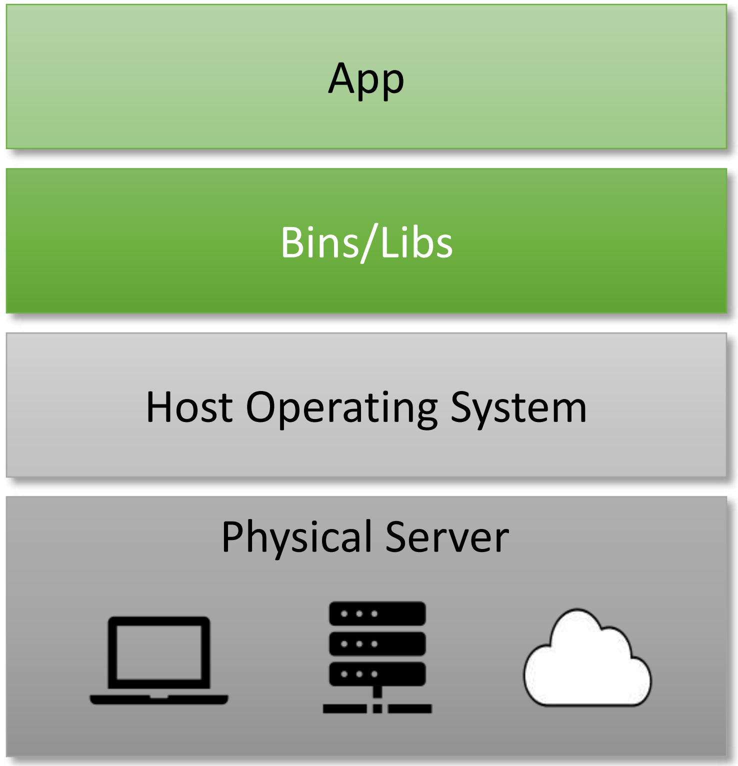
- Puppet is introduced in 2005
- Agent based Solution
- Puppet is model-driven and was built with systems administrators in mind; Moderate complex to setup and use
- Pull Base Methodology; end nodes contact back to Master
- Matured knowledge base
- Puppet Forge: Almost 6000 Modules available

Ansible

- Ansible is introduced in 2012
- Agent less Solution
- Ansible is easy to setup and understand than Puppet; More end user friendly
- Push Base Methodology; Master push tasks to end nodes
- Growing knowledge base
- Ansible Galaxy is still growing

Pre virtualization World

How it was



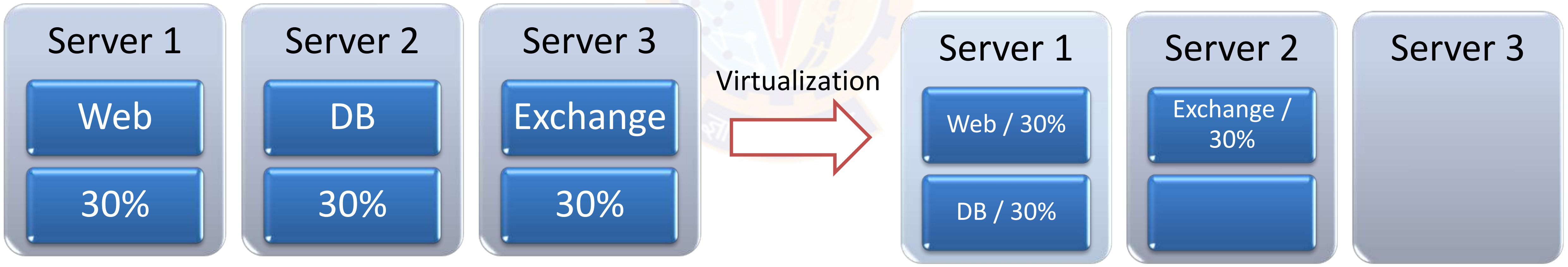
Problem

- Huge cost
- Slow Deployment
- Hard to migrate

Virtualization!!!

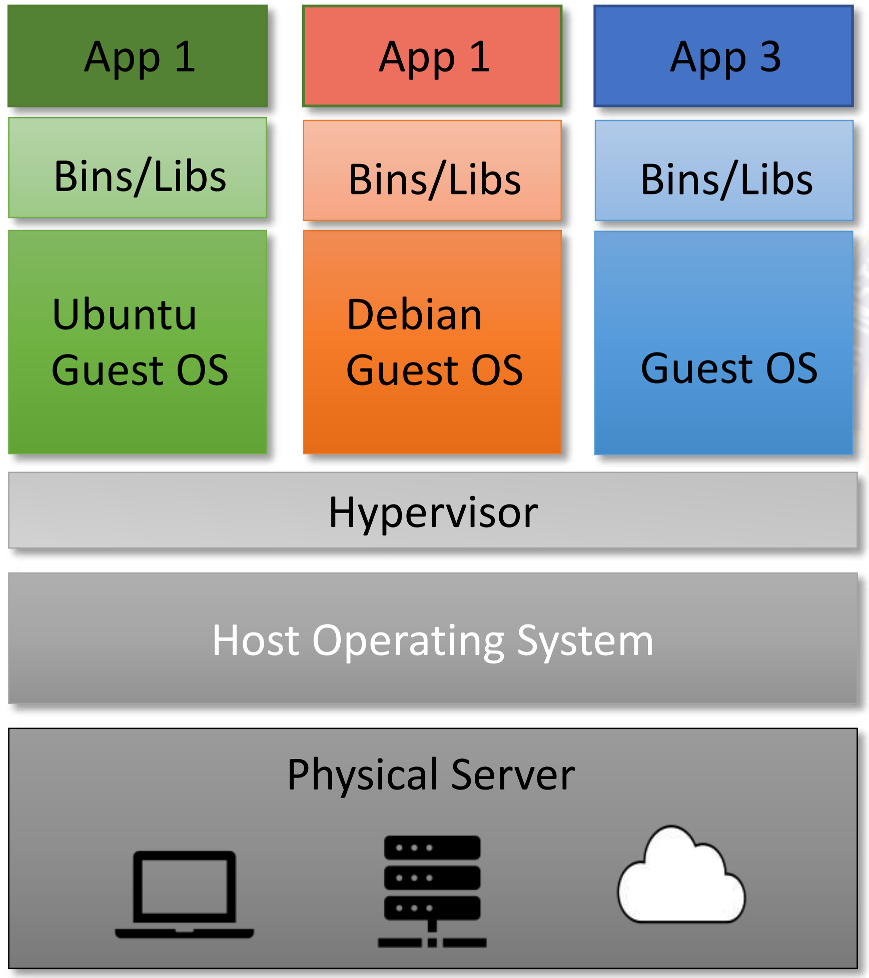
How it is

- Virtualization is technology that lets you create useful IT services using resources that are traditionally bound to hardware
- It allows you to use a physical machine's full capacity by distributing its capabilities among many users or environments



Virtualization!!!

H/w level virtualization



Benefits:

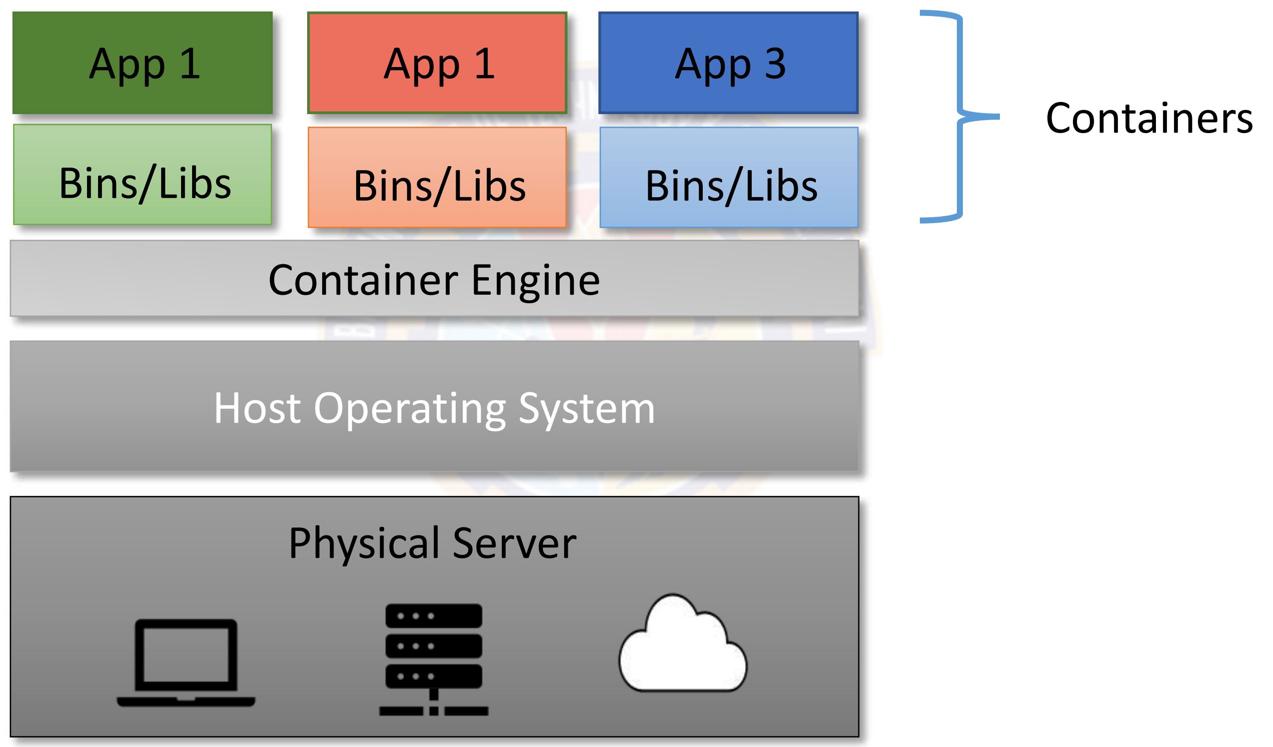
- More cost effective
- Easy to scale

Limitations:

- Kernel Resource Duplication
- Application portability issue

Virtualization!!!

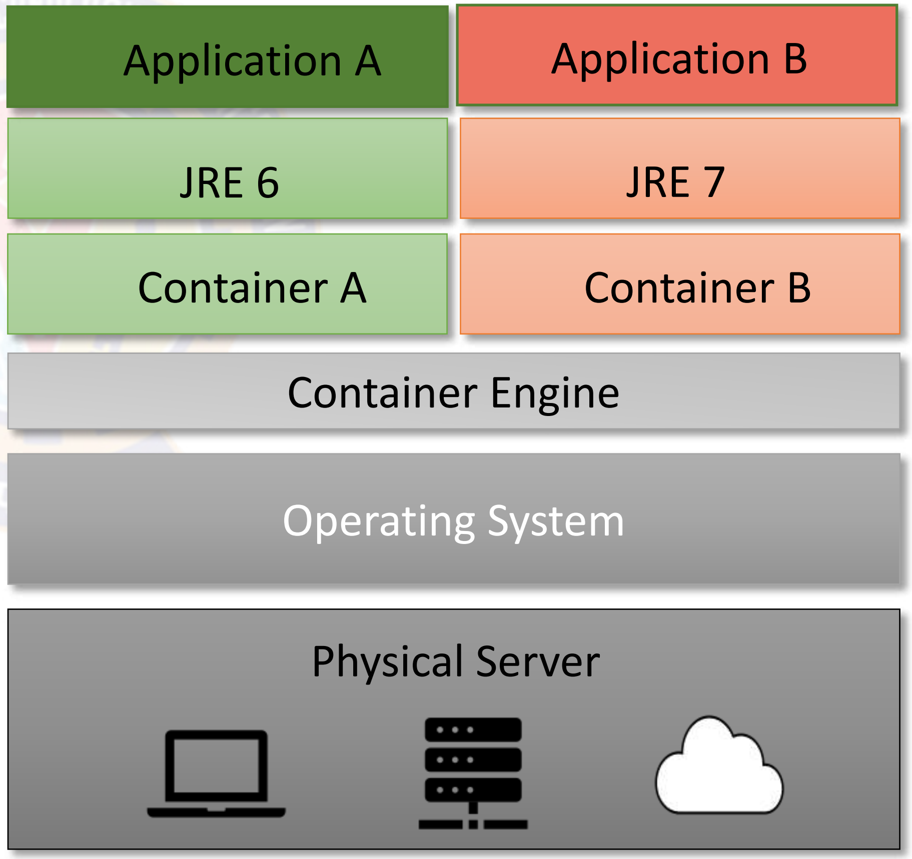
OS level virtualization



Container based Virtualization

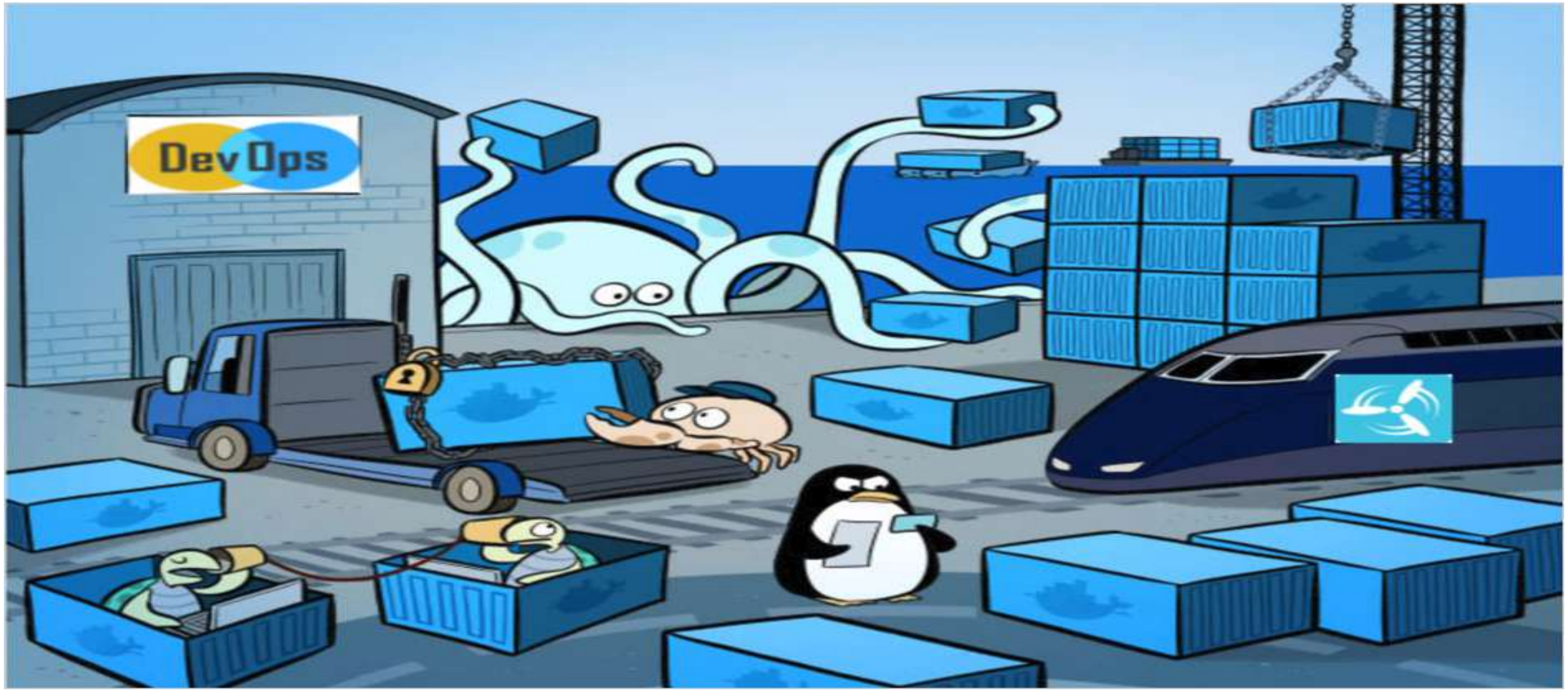
Benefits

- More cost effective
- Faster deployment speed
- Great portability



Container based Virtualization

Docker



Docker

Why

Open platform for developing, shipping, and running applications

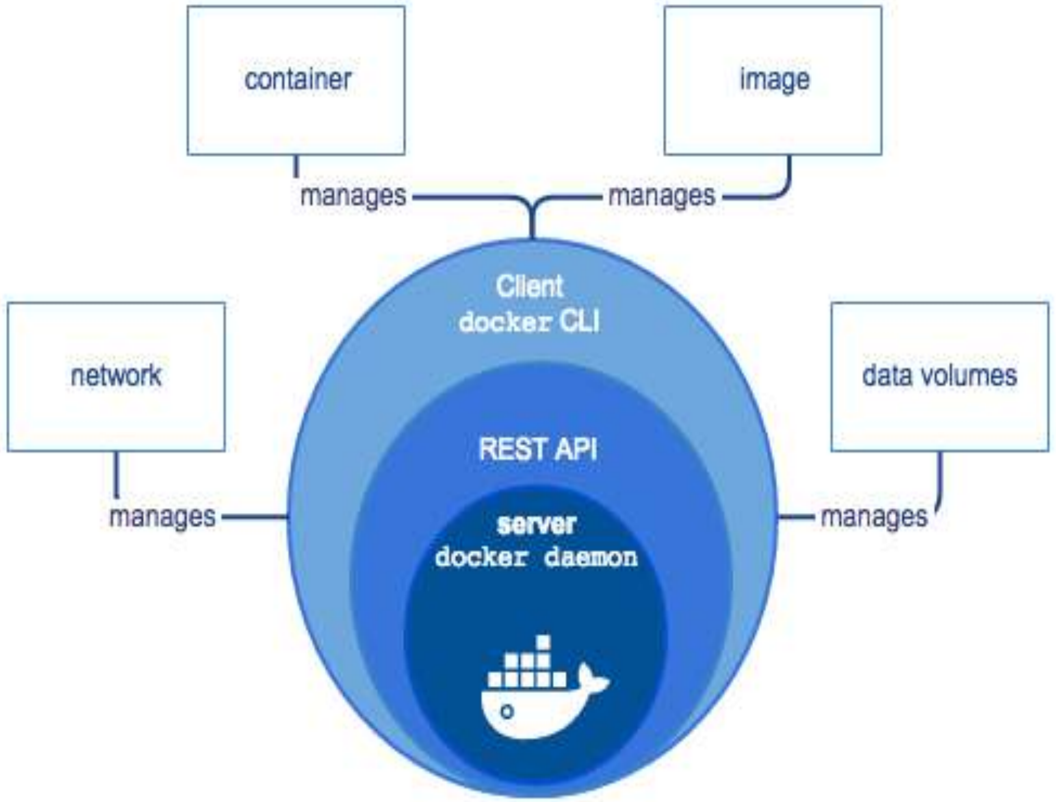
Enables you to separate your applications from your infrastructure so you can deliver software quickly

By taking advantage of Docker’s methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production

Docker

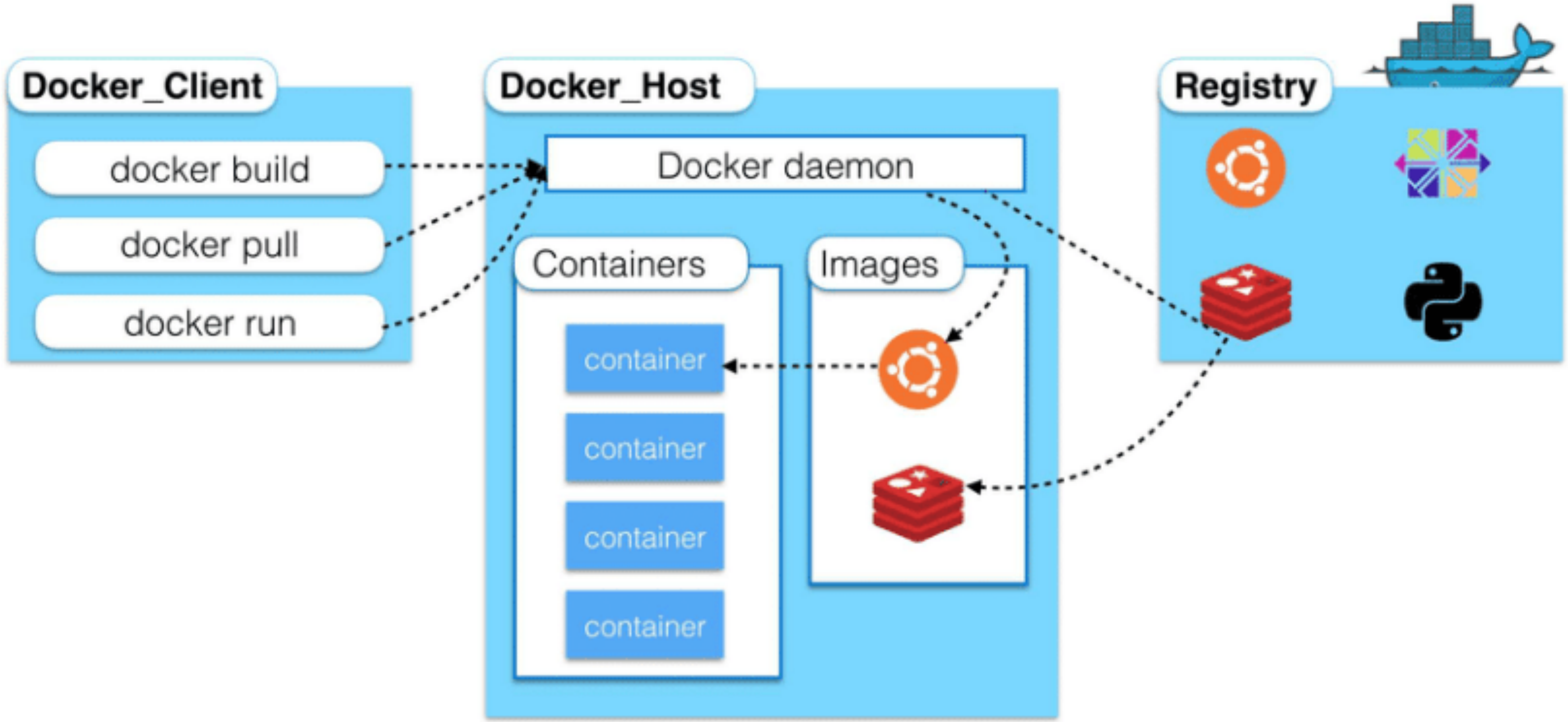
Docker Engine

- Client-server application
- A server which is a type of long-running program called a daemon process (the dockerd command)
- A REST API which specifies interfaces that programs can use to talk to the daemon and instruct it what to do
- A command line interface (CLI) client (the docker command)



Docker architecture

How it Works!!!



Docker Concepts

Images and Containers

Image

- An image is an executable package
- An image is a read-only template with instructions
- To build your own image, you create a Dockerfile

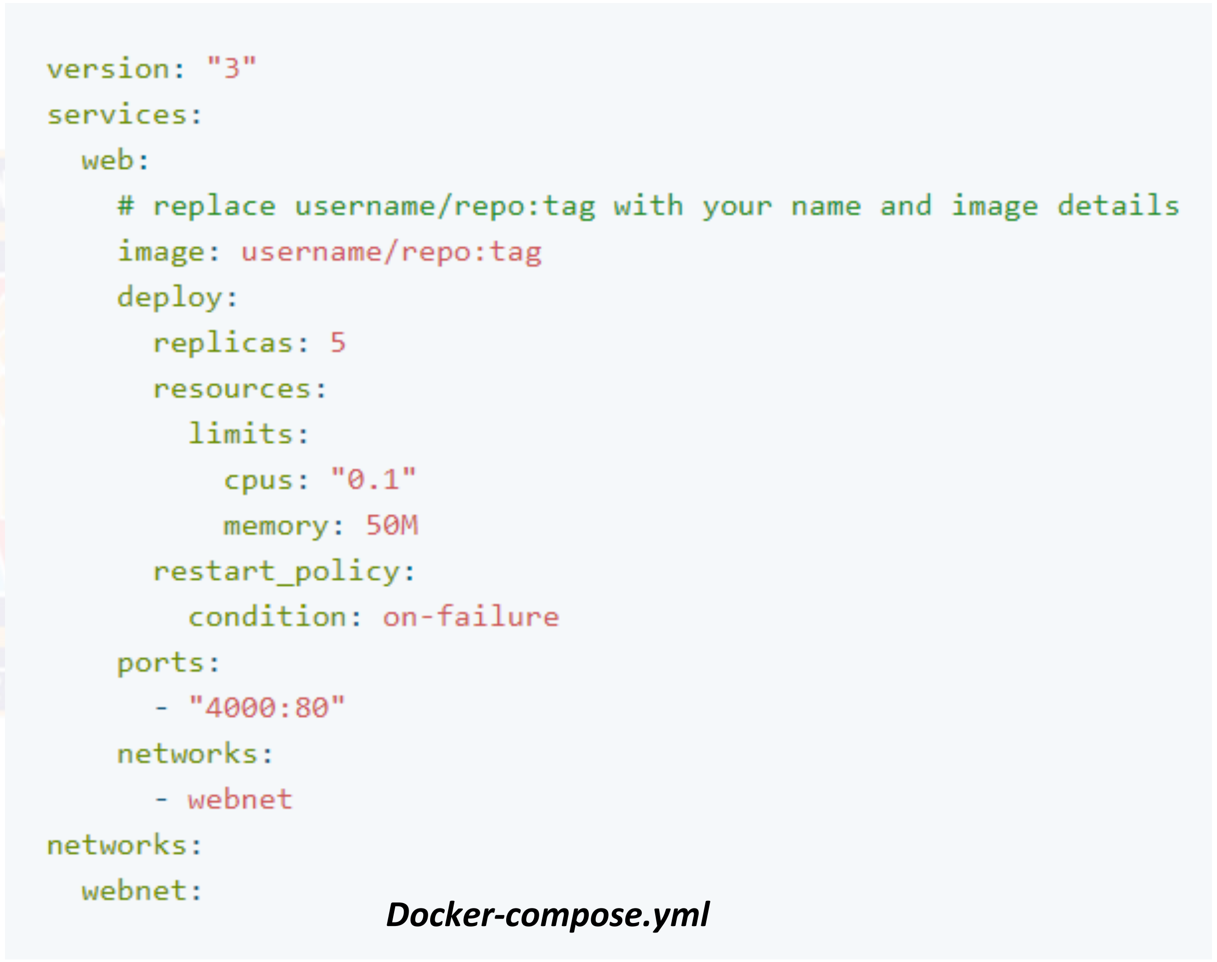
Containers

- A container is launched by running an image
- A container is a runtime instance of an image
- One can create, start, stop, move, or delete a container using the Docker API or CLI

Docker Concepts

Service

- Services are really just containers in production
- A service only runs one image, but it codifies
  - Way the Image Runs
  - What Ports it should use
  - How many replicas and etc.
- Scaling a service
  - No of container instances
  - Assign more compute resources



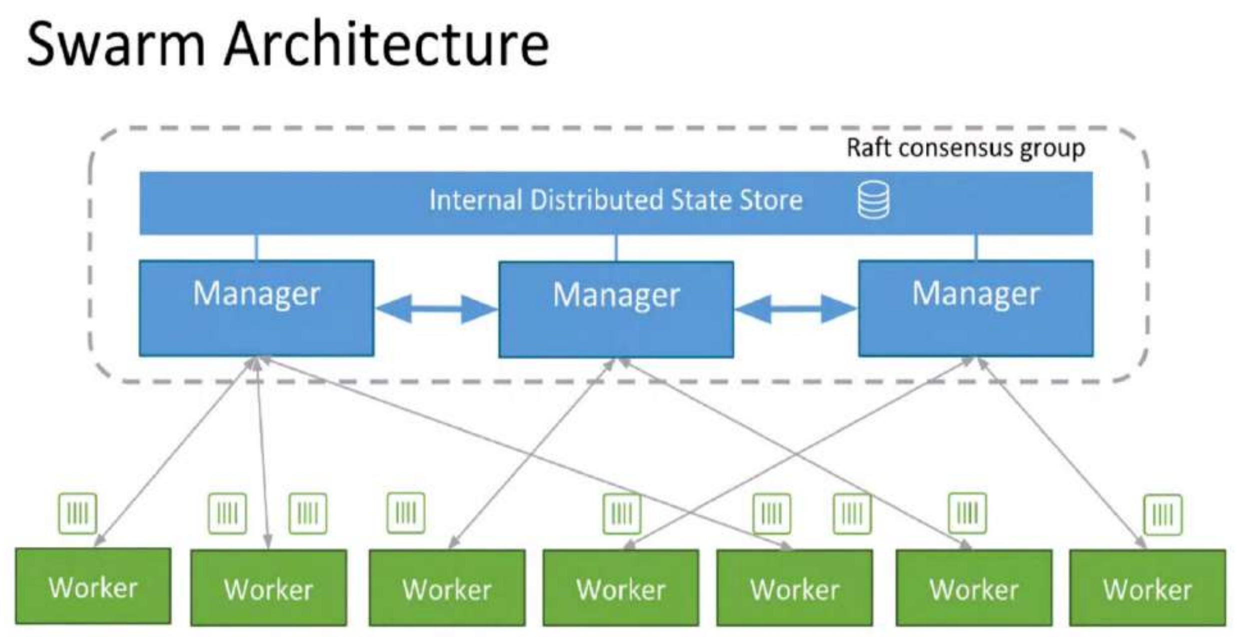
```
version: "3"
services:
  web:
    # replace username/repo:tag with your name and image details
    image: username/repo:tag
    deploy:
      replicas: 5
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
      restart_policy:
        condition: on-failure
    ports:
      - "4000:80"
    networks:
      - webnet
networks:
  webnet:
```

Docker-compose.yml

Docker Concepts

Swarms

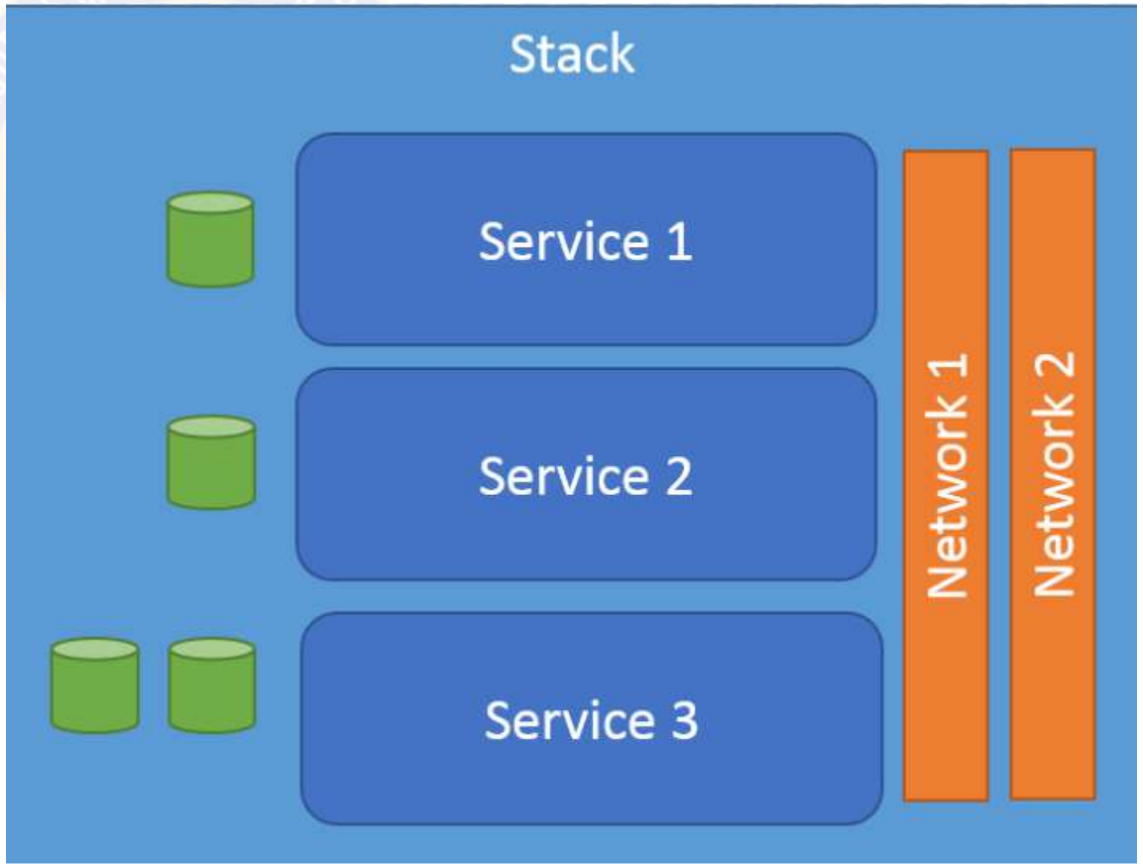
- A swarm is a group of machines that are running Docker and joined into a cluster
- Docker Command gets executed on a cluster by a swarm manager
- The machines in a swarm can be physical or virtual
- After joining a swarm, machines are referred to as nodes
- Swarm Strategies:
    - emptiest node
    - global



Docker Concepts

Stacks

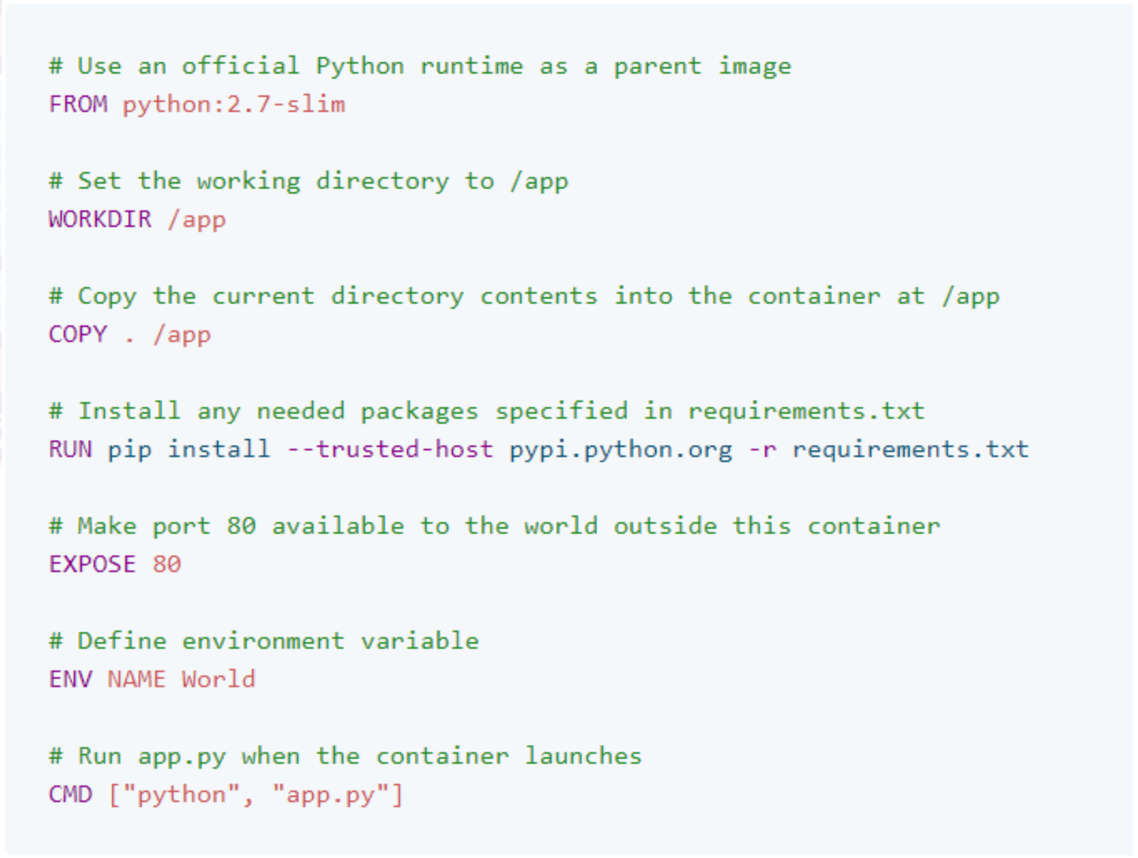
- A stack is a group of interrelated services that share dependencies, and can be orchestrated and scaled together
- A single stack is capable of defining and coordinating the functionality of an entire application



Docker Concepts

Docker file

- Dockerfile defines what goes on in the environment inside a container
- This file has information about all resources, that is network interface and disk drives used for the container



```
# Use an official Python runtime as a parent image
FROM python:2.7-slim

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
```

Docker Concepts

Docker Cloud

- Docker Cloud provides a hosted registry service with build and testing facilities for Dockerized application images
- Access to Docker Cloud is managed by Docker ID
- Manage Builds and Images
- Manage Swarms
- Manage Infrastructure
- Manage Nodes and Apps
- Integration with AWS and Microsoft Azure Services

Docker Concepts

Docker Store

- For developers and operators, Docker Store is the best way to discover high-quality Docker content
- Independent Software Vendors (ISVs) can utilize Docker Store to distribute and sell their Dockerized content
- Store Experience:
  - Access to Docker's large and growing customer-base
  - Customers can try or buy your software
  - Use of Docker licensing support
  - Docker handle checkout
  - Seamless updates and upgrades for your customers
  - Become Docker Certified

Docker Concepts

Docker Hub

- Docker Hub is a cloud-based registry service
- Allows link to code repositories
- Build images and test them
- Stores Images
- Links to Docker Cloud to deploy images to hosts
- It is a Centralize Solution for:
  - container image discovery
  - distribution and change management
  - user and team collaboration
  - workflow automation throughout the development pipeline

Docker Concepts

How is Docker Store different from Docker Hub

Docker Hub	Docker Store
Contents by Docker community	Contents submitted for approved by qualified Store Vendor Partners
Anyone can push new images to the community	Contents are published and maintained by Entity
No guarantees around the quality or compatibility	Docker Certified quality assurance
Common Thing: The Docker official Images are Available in both	

References

External

- https://docs.chef.io/
- https://docs.puppet.com/
- https://forge.puppet.com/
- https://docs.ansible.com/
- https://www.redhat.com/en/topics/virtualization/what-is-virtualization
- https://docs.docker.com/engine/docker-overview/#docker-engine
- https://docs.docker.com/

Q&A

Thank You!

In our next session: