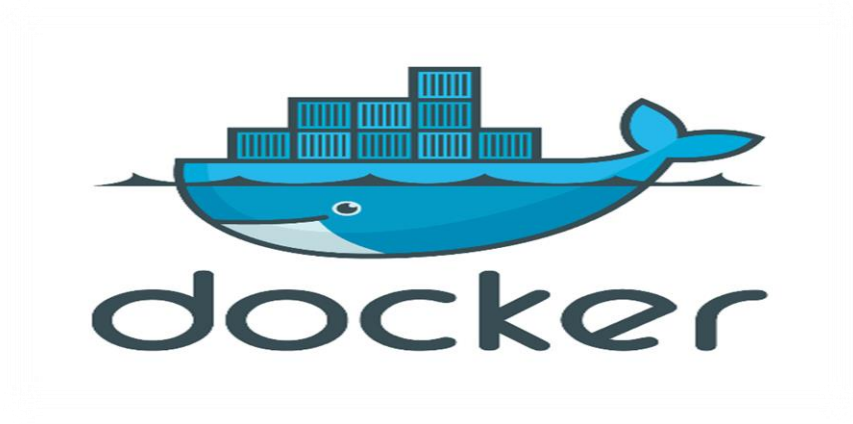

docker

CLOUD ENABLED

ENABLED BY CLOUD | DELIVERING CLOUD

Cloud and Devops Capabilities

USA Office

- New Jersey

India Office

- Bangalore

Web

- https://thecloudenabled.com

Our Capabilities

Cloud Enabled

- 4.5 years of Experience

AN OPENSTACK CONSULTING AND INTEGRATORS

Cloud Enabled » OpenSta × My meetings | GoToMeeti ×

https://www.openstack.org/marketplace/consulting/Cloud-Enabled/cloud-enabled

Explore the value of containers within an OpenStack infrastructure READ NOW

openstack SEARCH SOFTWARE USERS COMMUNITY MARKETPLACE EVENTS BLOG DOCS JOIN LOG IN

OpenStack Marketplace

TRAINING | DISTROS & APPLIANCES | PUBLIC CLOUDS | HOSTED PRIVATE CLOUDS | CONSULTING & INTEGRATORS | DRIVERS

BETA CHECK OUT THE NEW Community App Catalog Find cloud apps to run on OpenStack

< All Providers

CLOUD ENABLED
ENABLED BY CLOUD | DELIVERING CLOUD

ABOUT CLOUD ENABLED

Cloud Enabled announced World's First "OpenStack IaaS Expert " training with certification , this training is backed by our cloud Engineers who

Cloud Enabled

DETAILS & SIGNUP

No Matter you want to Deploy Private or Public powered by OpenStack. Our Exceptionally great Cloud Architects will guide you on POC, Architectural design,Install and also provide vendor neutral sizing or infrastructure resource

Areas of OpenStack Expertise Reviews

Our Flagship Openstack Customers

Makemytrip Inc. is an Indian online travel company founded in 2000.

Openstack Private cloud deployed and Fully managed service by Cloud Enabled

on-demand automation and BPO services model.

Openstack Private cloud deployed and managed by Cloud Enabled

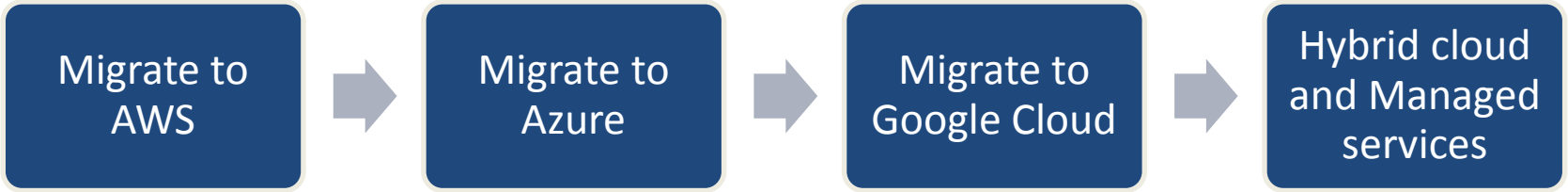
Government client

(confidential in NDA)

A government organization in India

Openstack Private cloud deployed and managed by Cloud Enabled

Multi Cloud –Migration and Managed services



Cloud Enabled - Non certification Trainings

Devops Training Capabilities

Devops
- Devops Technical Essentials (2 Days)
- Devops For Managers (1 Day)

Docker
- Docker Essentials (2 Days)
- Docker Advanced (4 Days)

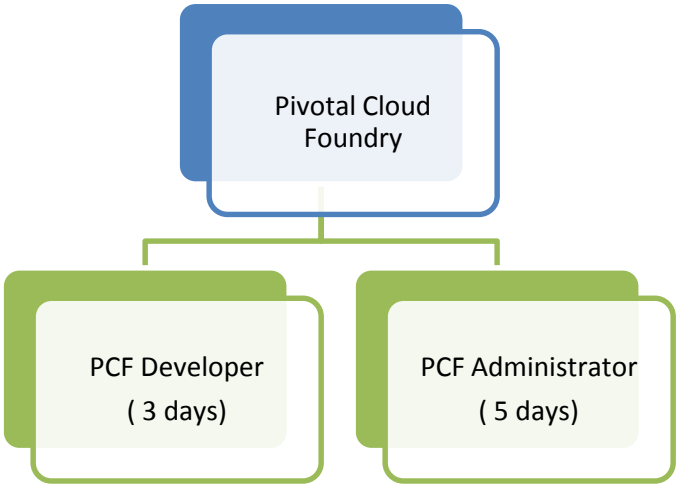
Chef
- Chef Essentials (2 Days)
- Chef Advanced (4 Day)

Puppet
- Puppet Essentials (2 Days)
- Puppet Advanced (4 Days)

Ansible
- Ansible Essentials (2 Days)
- Ansible Advanced (4 Day)

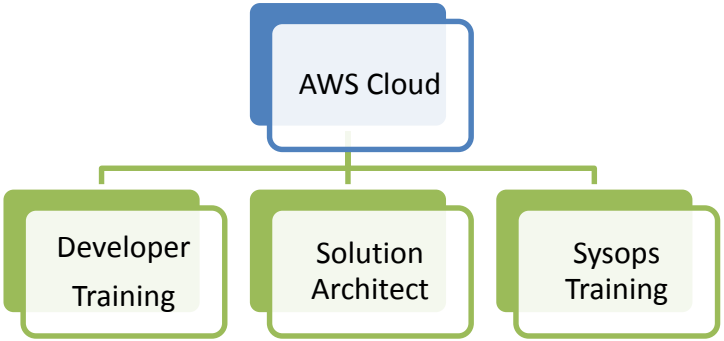
Other tools
- Sonarqube + Nexus (2 Days)
- Github Essentials (1 Days)

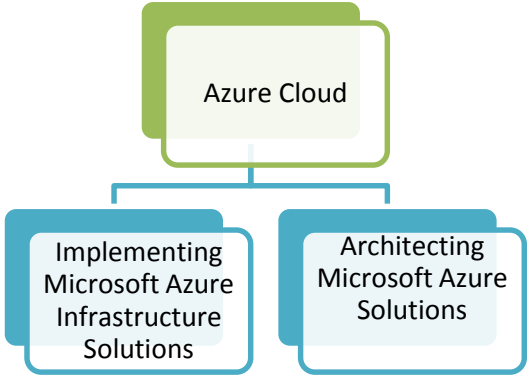
Pivotal Cloud Foundry Training Programs

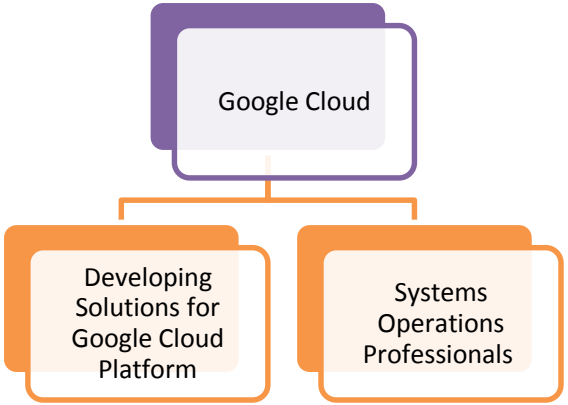


Pre-requisite : Atleast one or more year experience in Linux and any one programming knowledge

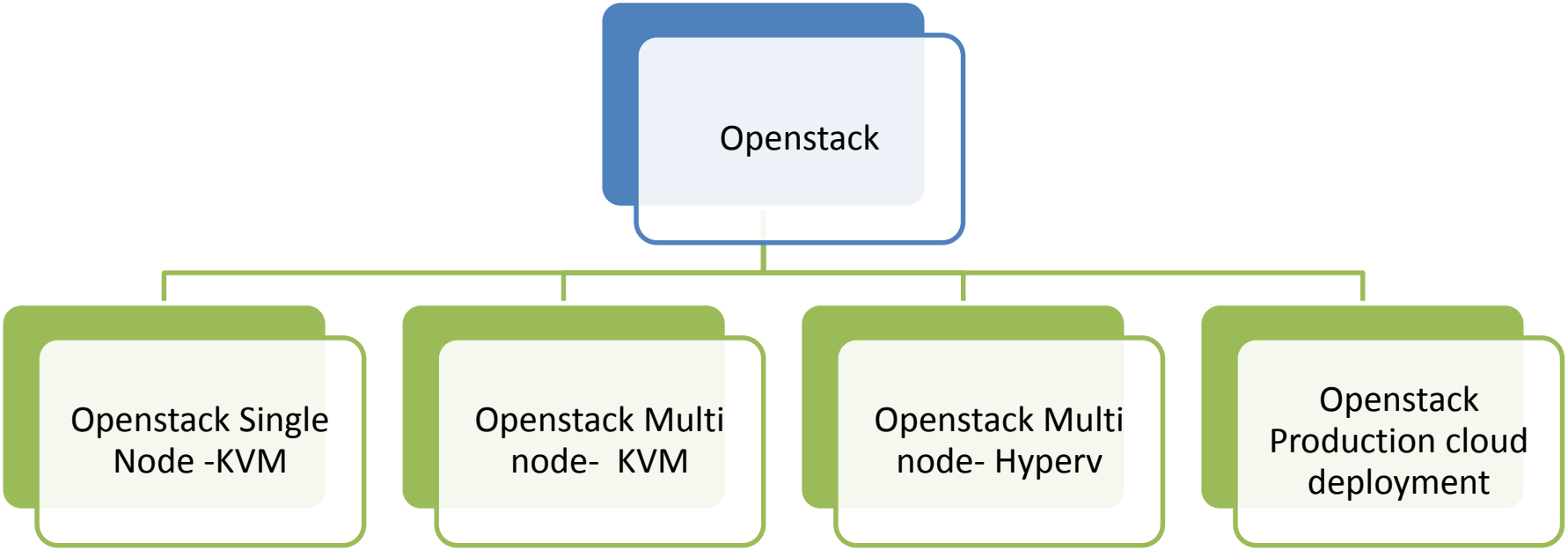
Public Cloud Certification Training Programs





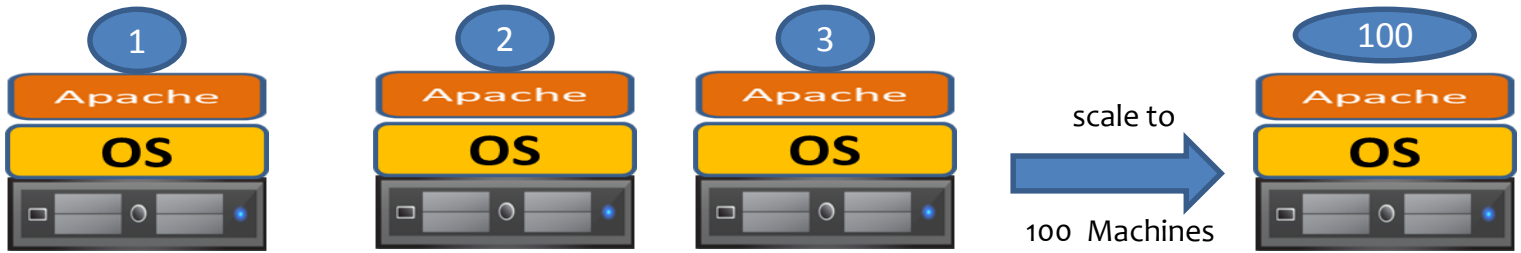


Openstack Certification Training Programs



Case Study : A company with 100 developers

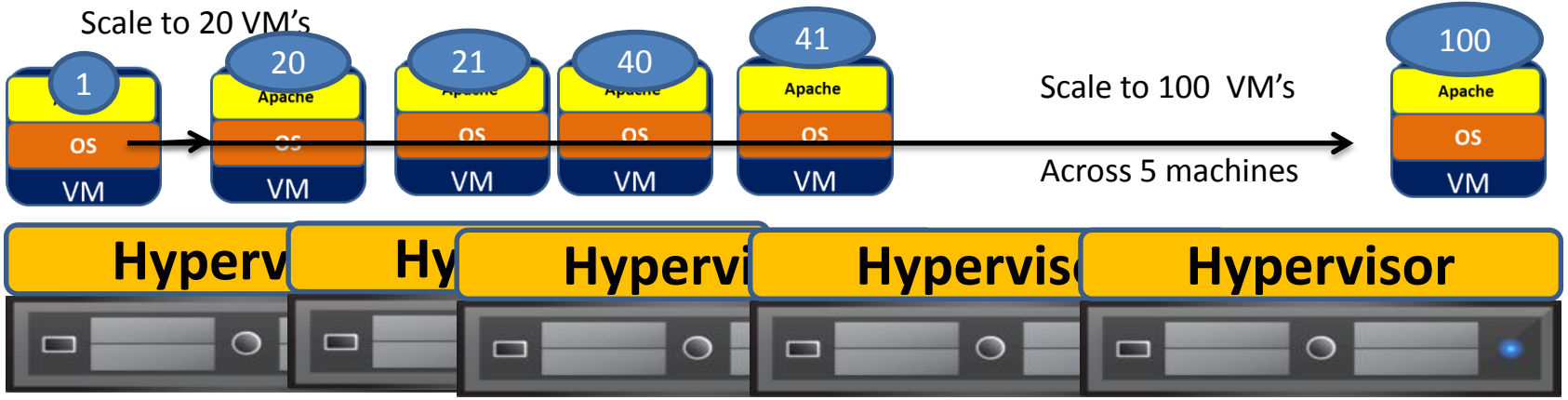
Physical world : 100 developer web environment



Disadvantages

- 100 U – Rack Space in data centre
- 100 x - Power cost
- 100 x - cooling cost
- **Under utilization :** Since it is dev environment to check code execution ,may not use entire CPU of each server

Virtualization world : 100 developer web environment



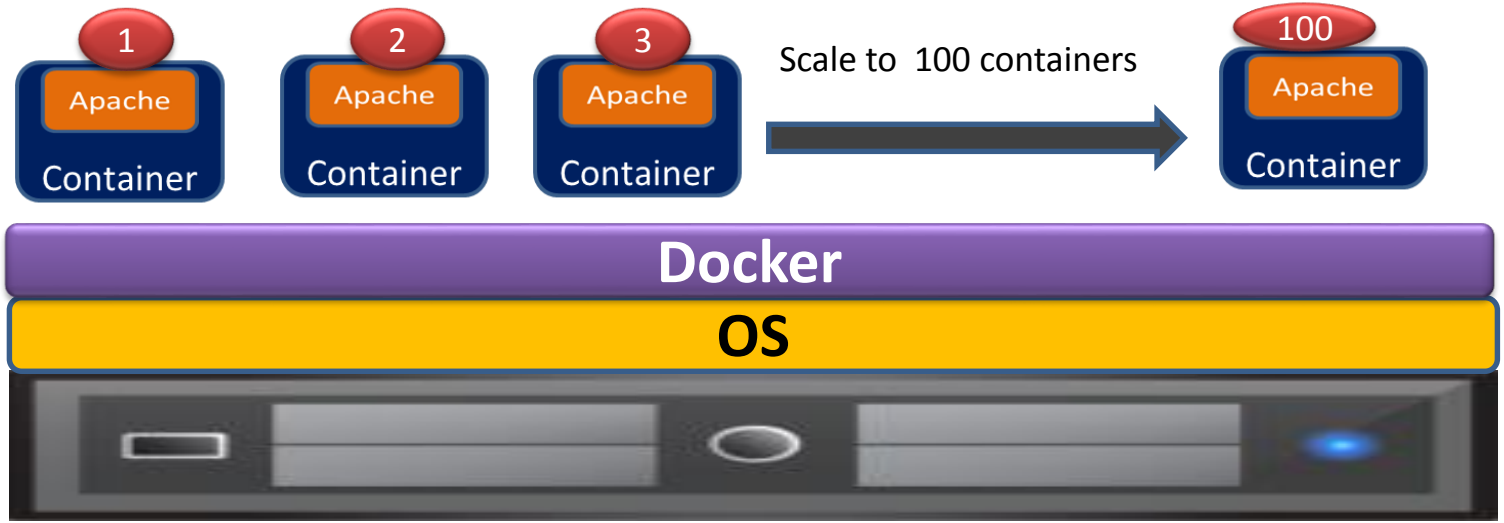
Advantages

- 5 U – Rack Space in data centre
- 5 x - Power cost
- 5 x - cooling cost

Disadvantage

- Each OS instance on VM running consumes CPU cycles and adds overhead just to run OS in order to have isolated app environments

Container world : 100 developer web environment



Advantages

- 1 U – Rack Space in data centre
- 1 x - Power cost
- 1 x - cooling cost

Docker- Networking

Docker Networking

When you install docker engine a virtual switch gets created on your docker host by name dockero

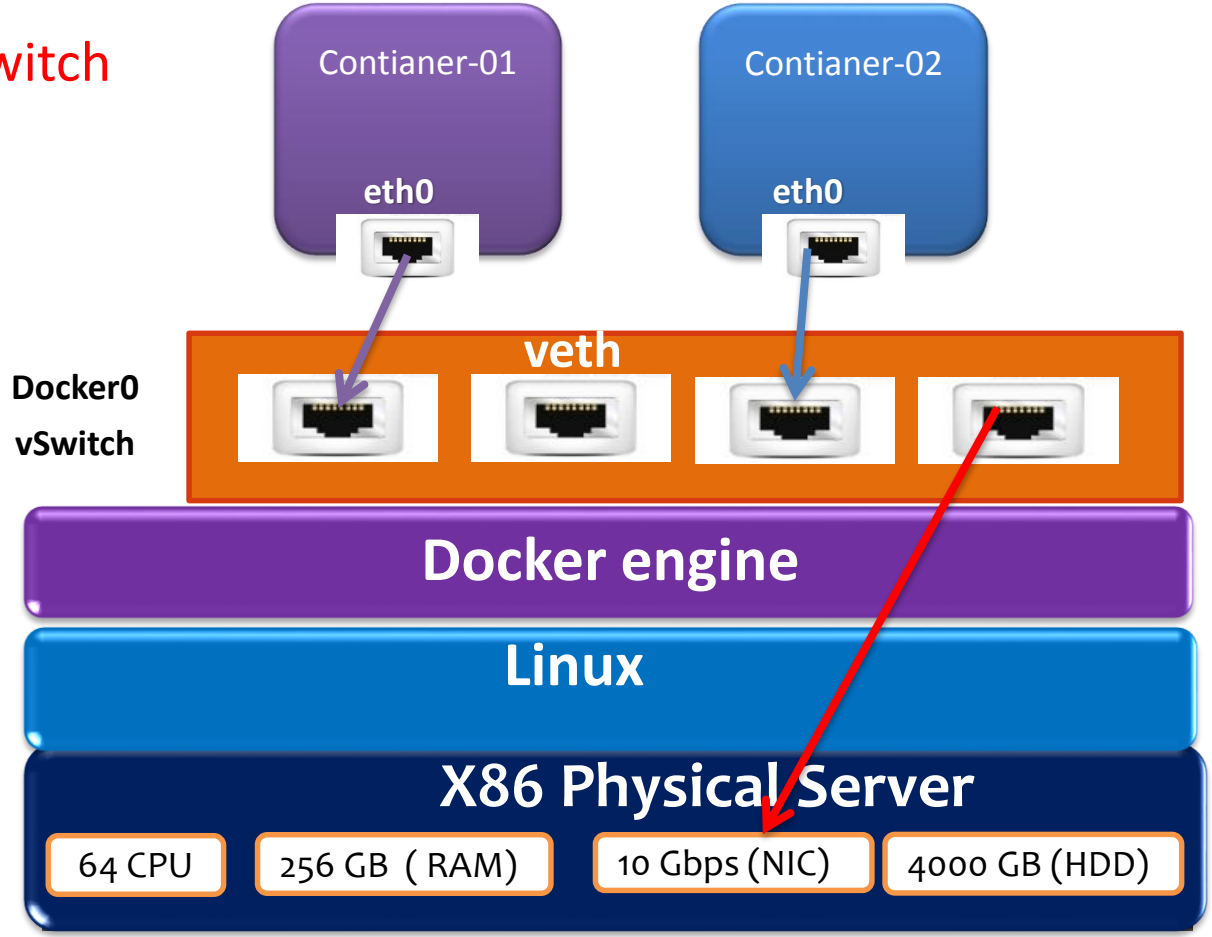
Command >>>> ip a

```
root@ip-192-168-1-49:/home/ubuntu# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 0e:34:07:cf:c0:88 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.49/24 brd 192.168.1.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::c34:7ff:fecf:c088/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:33:ed:08:ad brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:33ff:feed:8ad/64 scope link
       valid_lft forever preferred_lft forever
```

Docker0 bridge gets created

Docker Networking – vSwitch

- Vswitch gets created (A vSwitch is Layer2 software based Ethernet switch)
- Each container would have virtual Ethernet interface and get attached to vSwitch
- VSwitch has uplink to physical adapter on your host to access external world



Docker Networking

Once you a launch a **first** container , a virtual ethernet interface gets created on dokcero vswitch

Command >>>> docker run -i -t ubuntu:14.04 /bin/bash

Command >>>> ip a

```
root@ip-192-168-1-49:/home/ubuntu# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 0e:34:07:cf:c0:88 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.49/24 brd 192.168.1.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::c34:7ff:fecf:c088/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:33:ed:08:ad brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:33ff:feed:8ad/64 scope link
       valid_lft forever preferred_lft forever
21: veth7ff9c54@if20: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group default
    link/ether 9a:a3:15:8d:9a:c0 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::98a3:15ff:fe8d:9ac0/64 scope link
       valid_lft forever preferred_lft forever
root@ip-192-168-1-49:/home/ubuntu#
```

virtual ethernet interface gets created

Docker Networking - important commands

- Once you a launch a **second** container , a virtual ethernet interface gets created on dokcer0 vswitch
- Command >>>> docker run -i -t ubuntu:14.04 /bin/bash
- Command >>>> brctl show

```
root@ip-192-168-1-49:/home/ubuntu# brctl show
bridge name     bridge id               STP enabled     interfaces
docker0         8000.024233ed08ad       no              veth0c0276f
                                                        veth7ff9c54

root@ip-192-168-1-49:/home/ubuntu#
```

virtual ethernet interface for second contianer

Docker Networking - Default bridge gateway

Ip 172.17.0.1 acts as gateway

```
root@ip-192-168-1-49:/home/ubuntu# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN g
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast
    link/ether 0e:34:07:cf:c0:88 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.49/24 brd 192.168.1.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::c34:7ff:fecf:c088/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    link/ether 02:42:33:ed:08:ad brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:33ff:feed:8ad/64 scope link
       valid_lft forever preferred_lft forever
21: veth7ff9c54@if20: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdis
    link/ether 9a:a3:15:8d:9a:c0 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::98a3:15ff:fe8d:9ac0/64 scope link
       valid_lft forever preferred_lft forever
```

Docker Networking –Default bridge /network

- The bridge network represents the **docker0** network present in all Docker installations.
- Unless you specify otherwise with the docker run **--network=<NETWORK>** option, the Docker daemon connects containers to this network by default.
- You can see this bridge as part of a host's network stack by using the ifconfig command on the host.
- The default **docker0** bridge network supports the use of port mapping
- docker run --link to allow communications between containers in the docker0 network

Docker Networking – user created network

- You can create your own user-defined networks that better isolate containers
- docker provides some default **network drivers** for creating these networks
- You can create multiple networks.
- You can add containers to more than one network.
- Containers can only communicate within networks but not across networks.
- A container attached to two networks can communicate with member containers in either network.

Docker Networking – overlay network

- Create an overlay network on one of the machines in the swarm.
- **$ docker network create --driver overlay my-multi-host-network**
- This results in a single network spanning multiple hosts. An overlay network provides complete isolation for the containers
- Then, on each host, launch containers making sure to specify the network name.
- **$ docker run -itd --network=my-multi-host-network ubuntu**
- Once connected, each container has access to all the containers in the network regardless of which Docker host the container was launched on.

Docker Networking – Mapping Ports

- Auto mapping ports if you don't specify (random port is assigned by Docker automatically)
- **$ docker run –d –p nginx**
- Specify port mapping (in below example 9090 : is host port and port 80 port on container)
- **$ docker run –d –p 9090:80 tomcat**

Docker: Compose

Docker compose - Overview

- Compose is a tool for defining and running multi-container Docker applications.
- With Compose, you use a Compose file to configure your application's services.
- Then, using a single command, you create and start all the services from your configuration
- Lately (early february 2016) Docker officially announced the availability of the new version (V2) of the docker-compose.yml file format,
  - with support for the latest features in docker engine: volumes and networks.

Docker compose – 3 step process

- Using Compose is basically a three-step process
  - Define your app's environment with a Dockerfile so it can be reproduced anywhere.
  - Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment.
  - Lastly, run docker-compose up and Compose will start and run your entire app.

Docker compose- Sample file

A docker-compose.yml looks like this:

```
version: '2'

services:
  db:
    image: mysql:5.7
    volumes:
     - db_data:/var/lib/mysql
    restart: always
    environment:
     MYSQL_ROOT_PASSWORD: wordpress
     MYSQL_DATABASE: wordpress
     MYSQL_USER: wordpress
     MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
     - db
    image: wordpress:latest
    ports:
     - "8000:80"
    restart: always
    environment:
     WORDPRESS_DB_HOST: db:3306
     WORDPRESS_DB_PASSWORD: wordpress
volumes:
    db_data:
```

Docker- Datacentre

Docker Data centre - An overview

Docker Data centre

Docker Trusted registry

docker tutum

Docker Data centre – Docker cloud differences

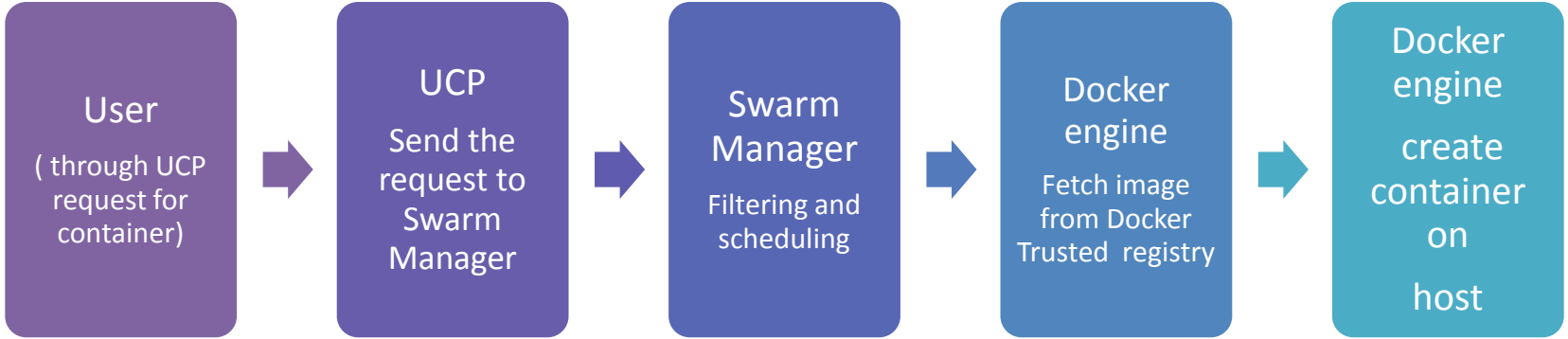
Image Management

- Docker Trusted registry
- Store and retrieve images by your users

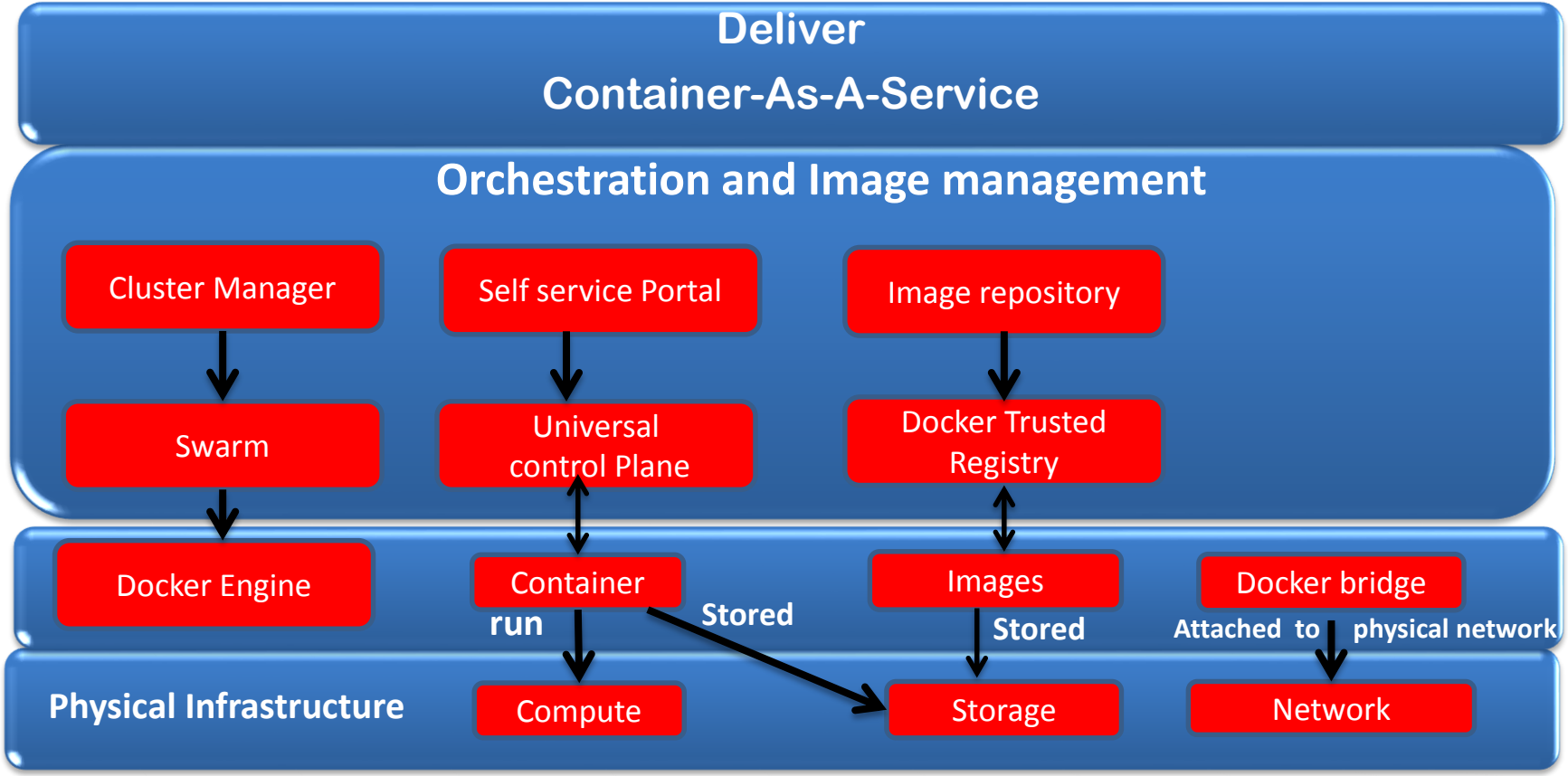
User Interface

- Universal control Plane
- User and Admin Portal to Create, Stop, and Terminate containers

Docker data centre behind the scenes



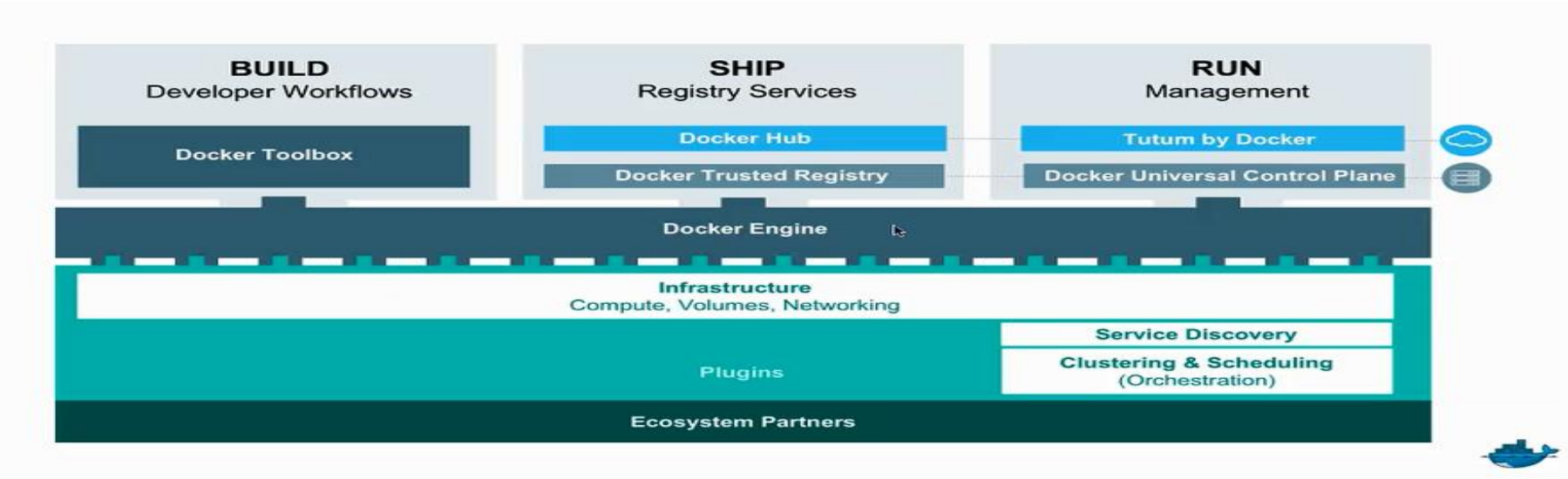
Docker Data Centre – Infrastructure Framework



Difference between docker data cnetre and docker cloud

Docker cloud = Docker hub (to store images) + tutum control plane (for UI)

Docker DC = Docker Trusted registery (to store images behind the firewall) + UCP (UI)



Pricing

FREE 30-DAY TRIAL

This trial lets you take the Docker Containers-as-a-Service (CaaS) Platform for a spin.

30 Day Free Trial

Docker Universal Control Plane

Docker Trusted Registry

Supported Docker Engine

BUSINESS DAY

Build, ship and run your applications behind the firewall, with business day support from Docker.

$150 monthly per instance

$1500 yearly per instance

Purchase Online

Contact Sales

Docker Universal Control Plane

Docker Trusted Registry

Supported Docker Engine

Mon-Fri 9am to 6pm Local Time

BUSINESS CRITICAL

Build, ship and run your applications behind the firewall, with business critical 24 x 7 support from Docker.

$300 monthly per instance

$3000 yearly per instance

Purchase Online

Contact Sales

Docker Universal Control Plane

Docker Trusted Registry

Supported Docker Engine

24/7/365

Thank You