Red Hat

Ready

Business Partner
Training

OSELabs

Docker Container & K8S
- Saravanan

Agenda!

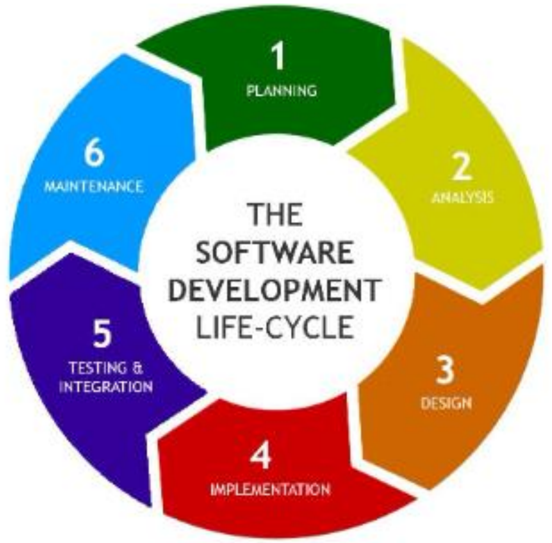
Day-1

- Introduction
- Pre-Assessment
- SDLC Phases
- Monolithic Architecture - Pros & cons
- Virtualization Architecture - Pros & cons
- SOA Architecture – Pros & cons
- Microservice Architecture – Pros & cons
- What is API ?
- What is Stateless & Stateful Applications ?
- Containerizing Stateless & Stateful Applications

SDLC ?

OSELabs

Software Development Life Cycle / Application Development life-cycle



GOALS:

- To create Bug free & high quality software
- To meet client/customer expectation.

PLANNING — Requirement Gathering

- Scope of the entire project
- Anticipated issues
- Opportunities
- 'Software Requirement Specification' docs created

ANALYSIS — Feasibility study

- Define &
- document S/W needs.
- Economic, Legal, operation, Technical, Schedule.

DESIGN — Design

- Software design documents are prepared as per the requirement specification document.
- High level design
- Low level design

IMPLEMENT — Coding

- Developers start build the entire system by writing code using the chosen programming language
- Software is completed, deployed in testing env

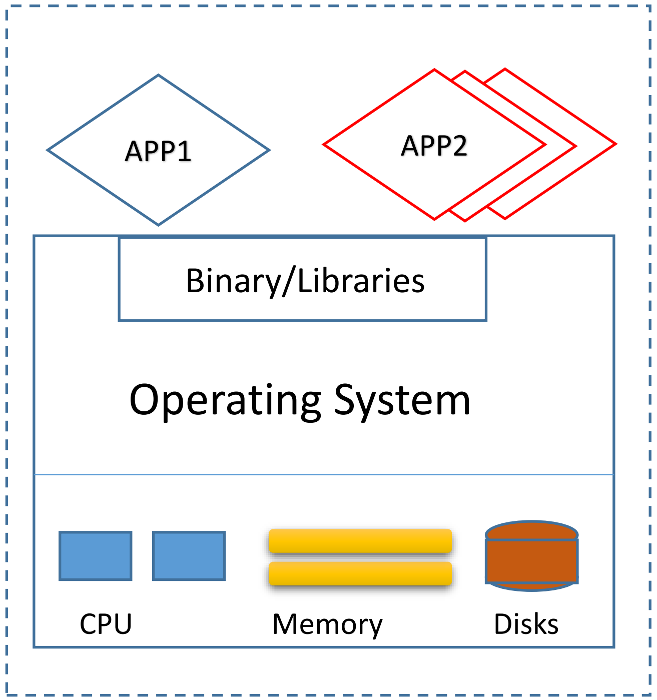
TESTING & INTEGRATION — Testing

- Functionality of the entire system is tested.
- Identified bugs/defects are communicate to developers until the s/w is bug free and final deployment process starts.

INSTALLATION & MAINTENANCE — Maintenance

- Bug fixing
- Upgrade
- Enhancement

Monolithic Architecture

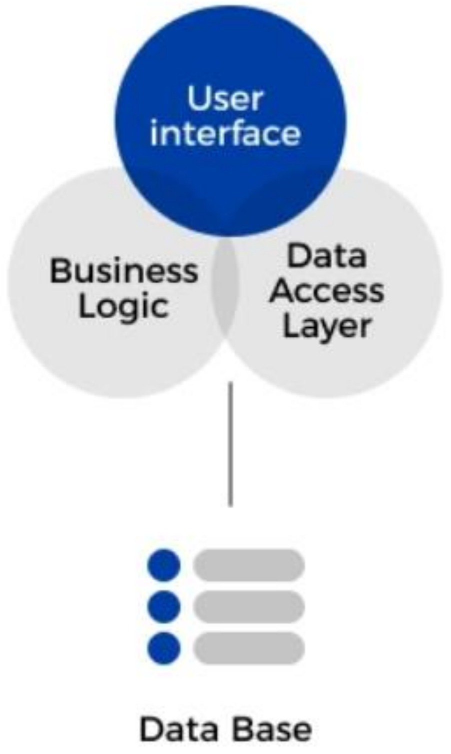


PROS:

- Simpler development and deployment.
- Works Well with Single Application.

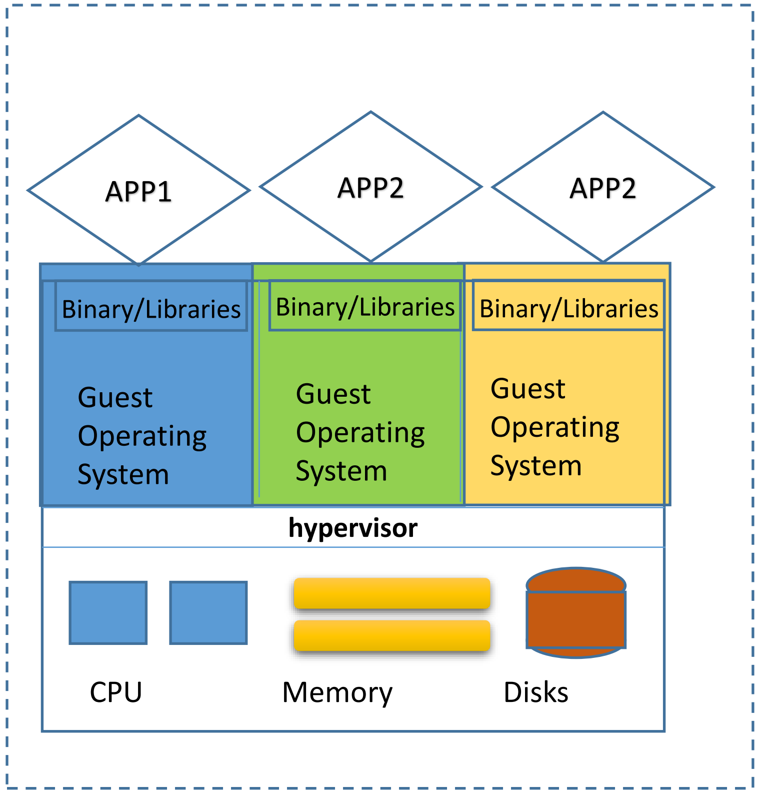
CONS:

- Shared Application shares the binary/libraries.
- Application is tightly coupled to Operating System.
- Compute resources are NOT optimally used.
- Small change will involve complete code change.



OSELabs

Virtualization Architecture



PROS:

- Optimal Compute Resource utilisation through H/w Virtualisation.
- Multiple Application can share same physical foot print.(hardware)

CONS:

- Guest Operating System comes with price.
- Hypervisor needs ample compute resource to function.
- Application is still tightly coupled to Operating System

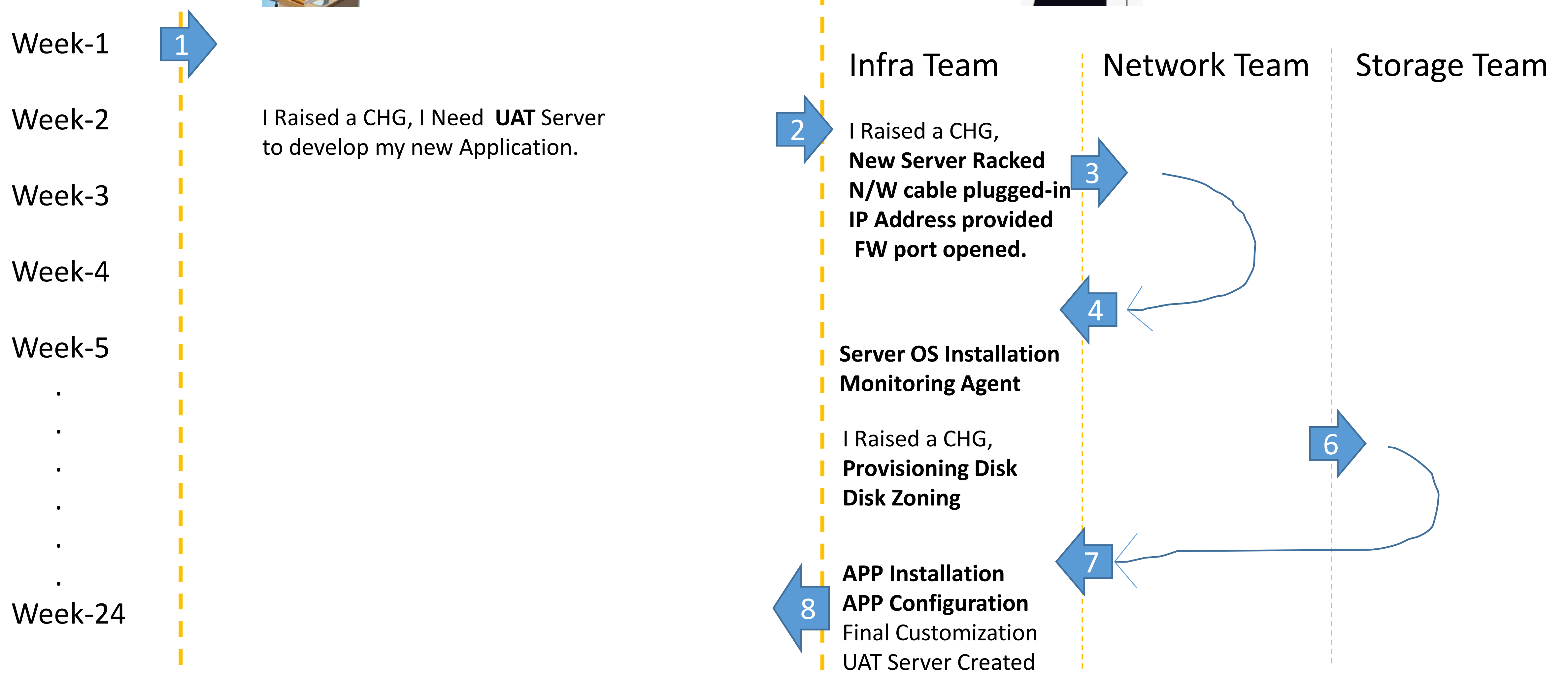
Example:

- VMware -ESX , Microsoft -Hyper-V, XEN

"Development" Vs "Operational" Team

Development Team

Operational Team



I Raised a CHG, I Need **UAT** Server to develop my new Application.

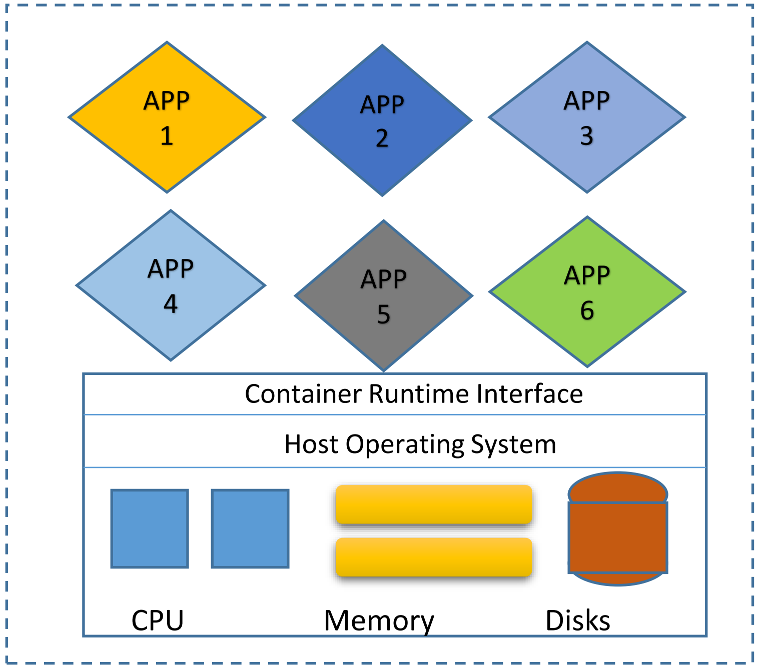
I Raised a CHG,
New Server Racked
N/W cable plugged-in
IP Address provided
FW port opened.

Server OS Installation
Monitoring Agent

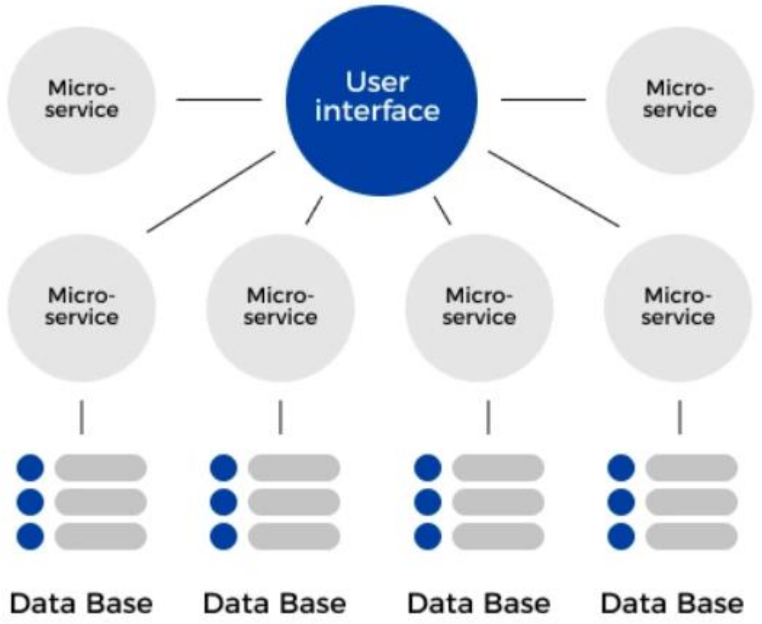
I Raised a CHG,
Provisioning Disk
Disk Zoning

APP Installation
APP Configuration
Final Customization
UAT Server Created

Microservice Architecture



- Container is smallest compute unit runs as processes in host OS.
- Each application hosted inside containers are isolated.
- Containers minimize the impact of any OS update on host OS.
- Except binary/libraries, these environment has dependencies on core operating system.

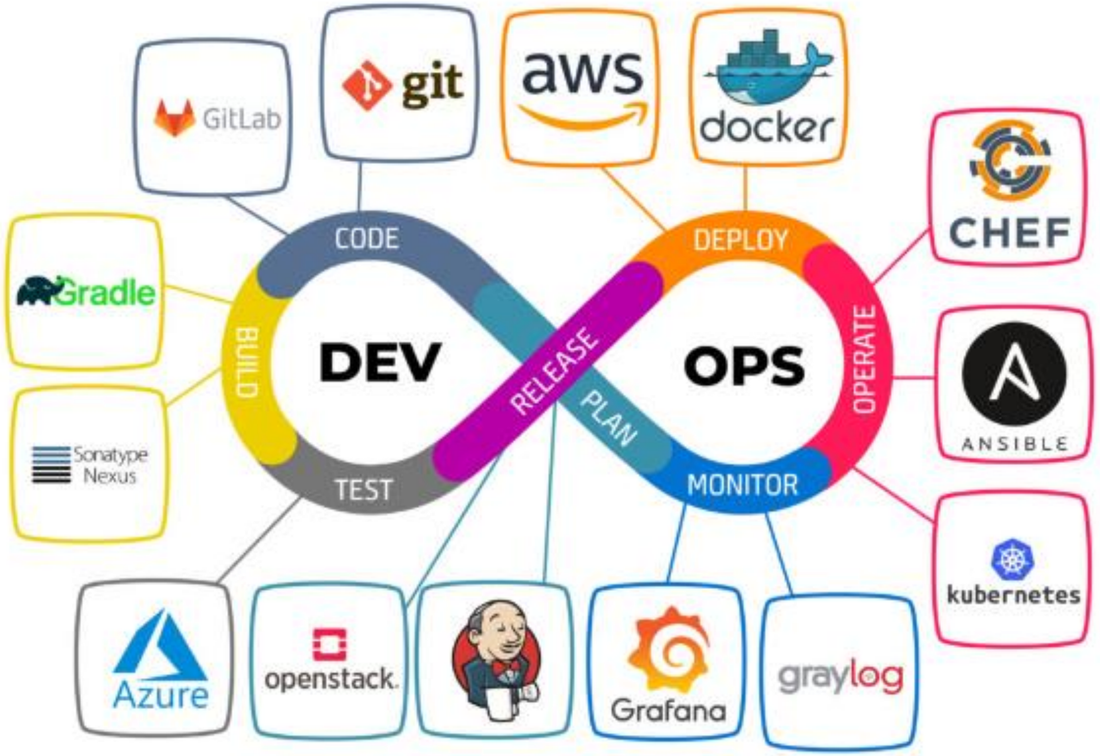


- Containers are Light weight and portable.
- Application running inside containers are called Microservices.
- Application Scaling is easy and Application are decoupled from OS.

DevopS

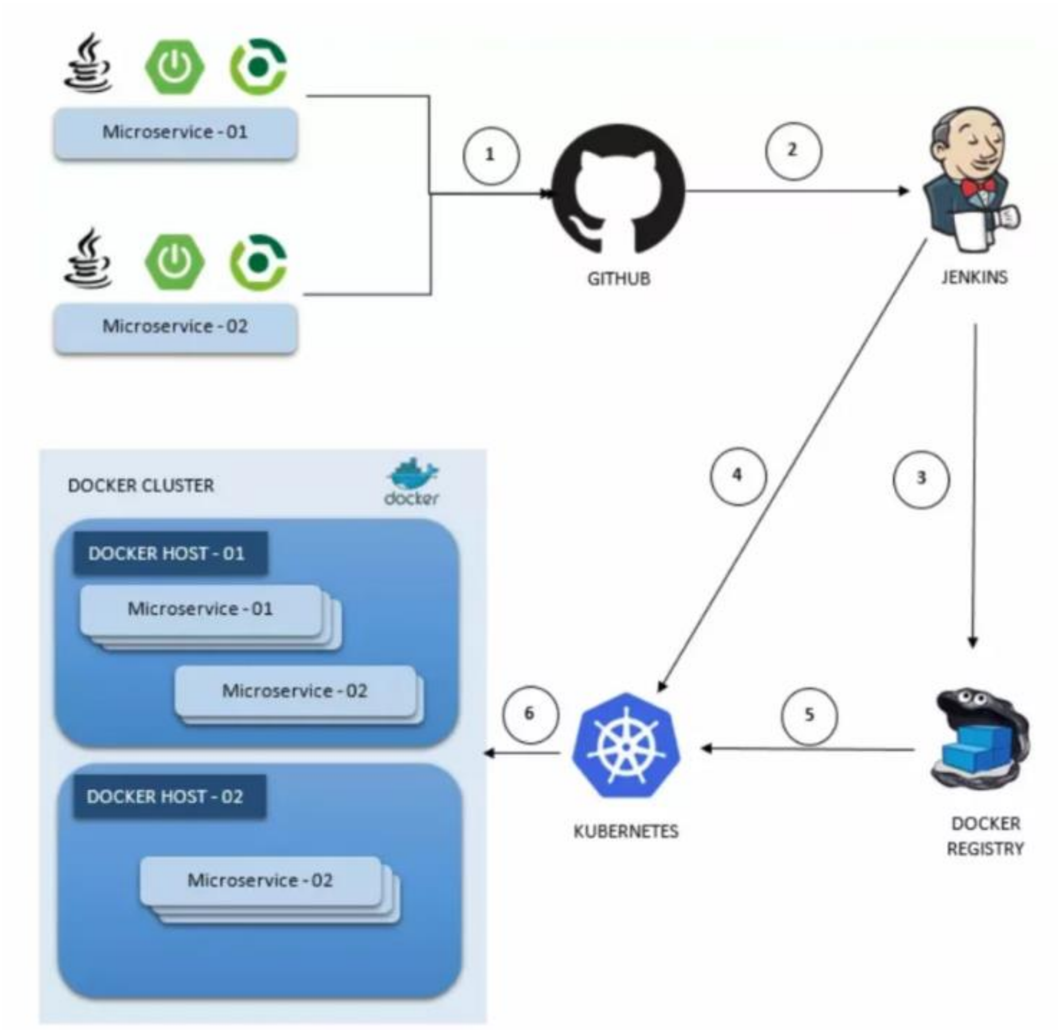
OSELabs

- Address all the limitation by new Culture i.e. Dev+Ops.
- Faster development and deployment of applications
- Decrease in software delivery time
- Improves customer experience and satisfaction.
- Leads to better team engagement and productivity
- Automation is an key aspect of Devops
- CI/CD Pipeline helps in achieving DEVOPS goals.



PIC Credit: https://shalb.com/blog/what-is-devops-and-where-is-it-applied/

CI-CD Pipeline

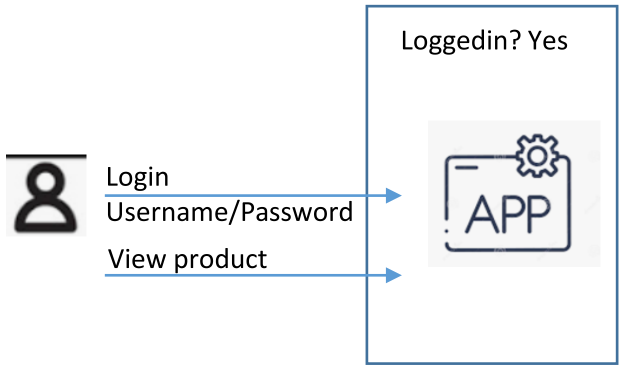


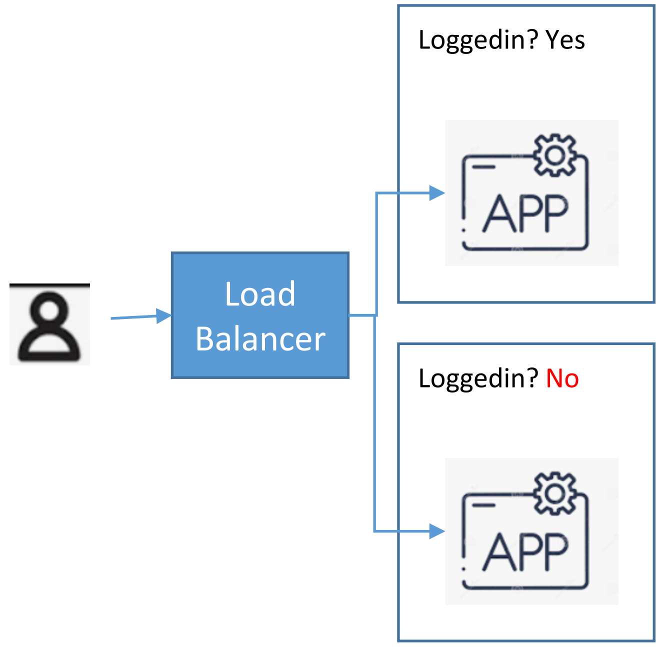
Continuous Integration and Deployment Pipeline

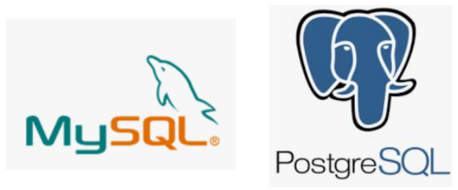
- **Continuous Integration:**
- **Continuous delivery**
- **Continuous Deployment:**

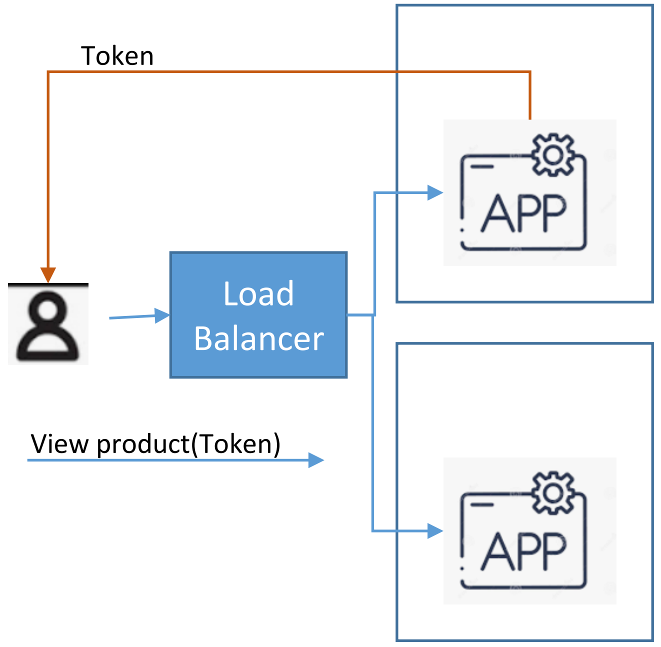
Stateful Application

Stateless Application

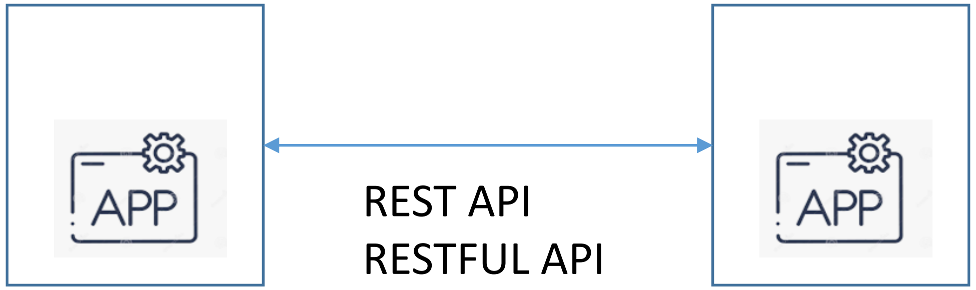






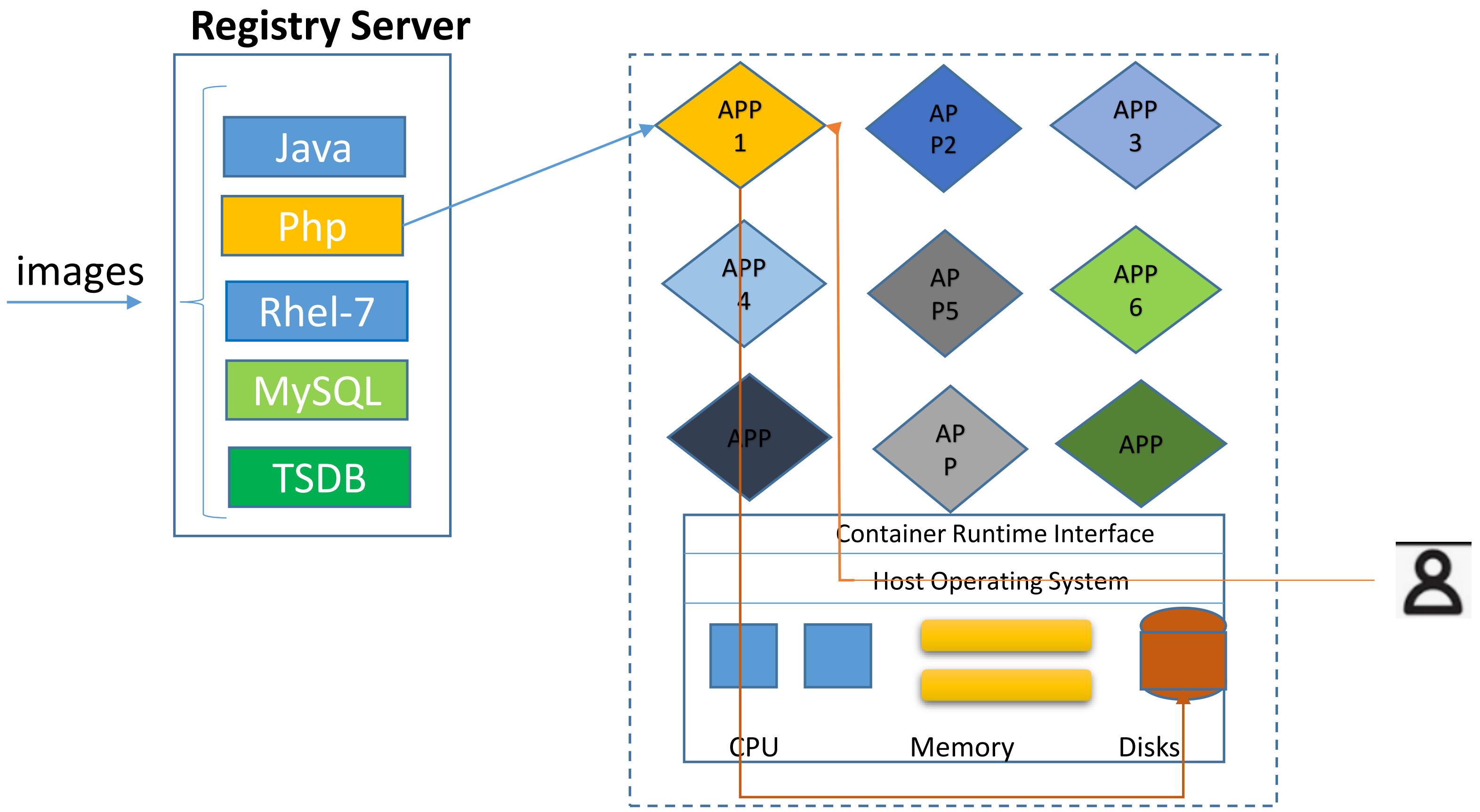


Application Program Interface (API)



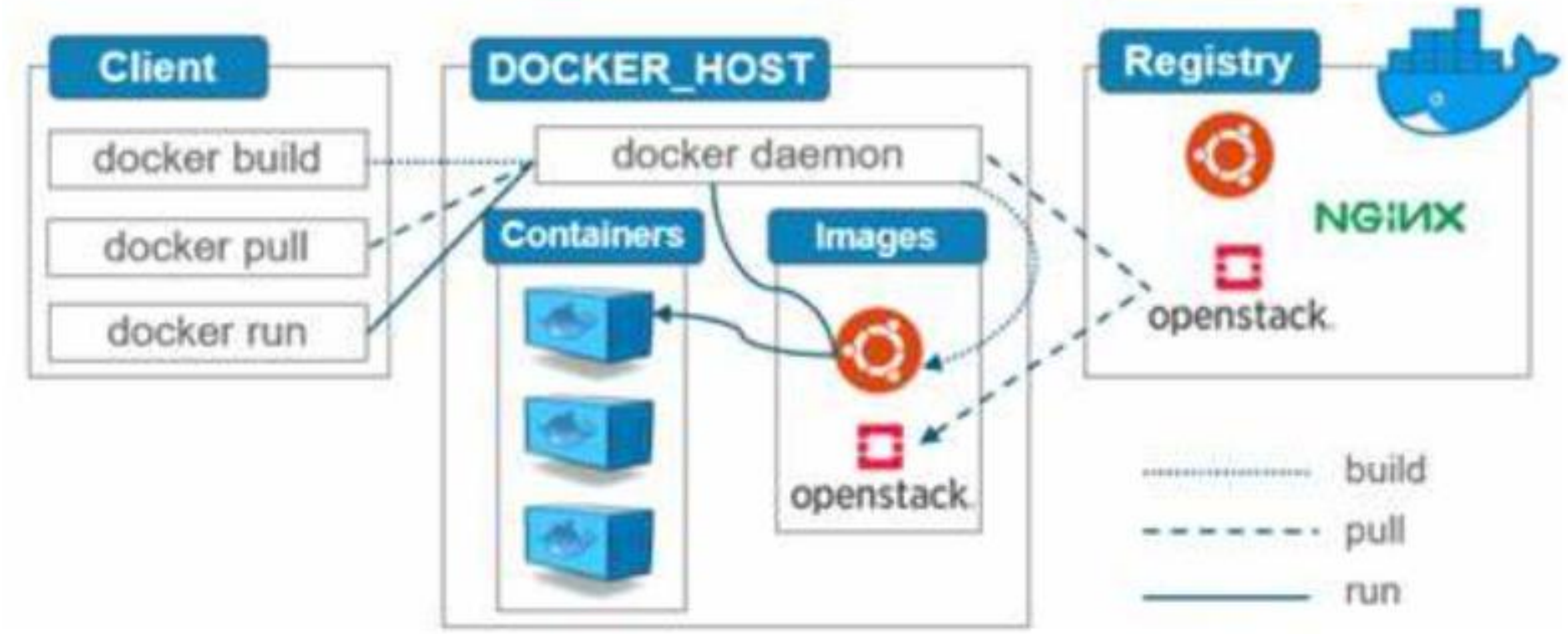
- API is intermediatory software between two different applications
- Microservice architecture applications are built with API

Containers



- Light weight
- container runs for an purpose , there must be a task for it.
- CRI manages the containers
- Port mapping
- Have dependencies on host operating system except binary/library.
- Read-only (data-lost)
- Persistent Volume

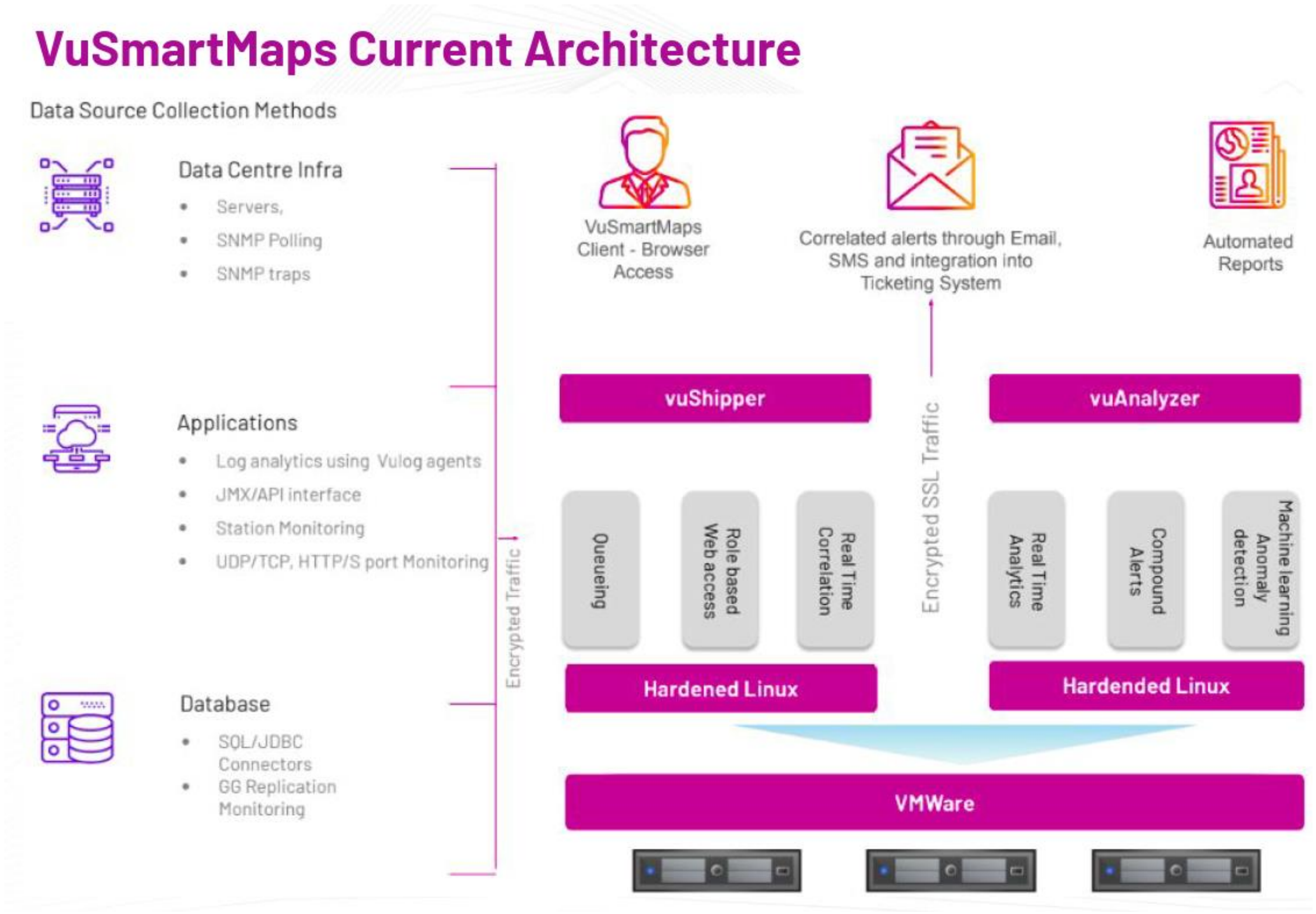
Docker


Client
docker build
docker pull
docker run
DOCKER_HOST
docker daemon
Containers
Images
openstack
Registry
NGINX
openstack
build
pull
run

Docker Containers (DEMO)

- Docker Pull to pull docker images from registry server.
- Create Container using the pulled images.
- Removing the unused images.
- Executing remote command on docker container
- Logging in to container and execute commands
- Stop Container
- Delete and recreate container
- Mapping ports between host operating system and containers
- Setting up persistent storage.

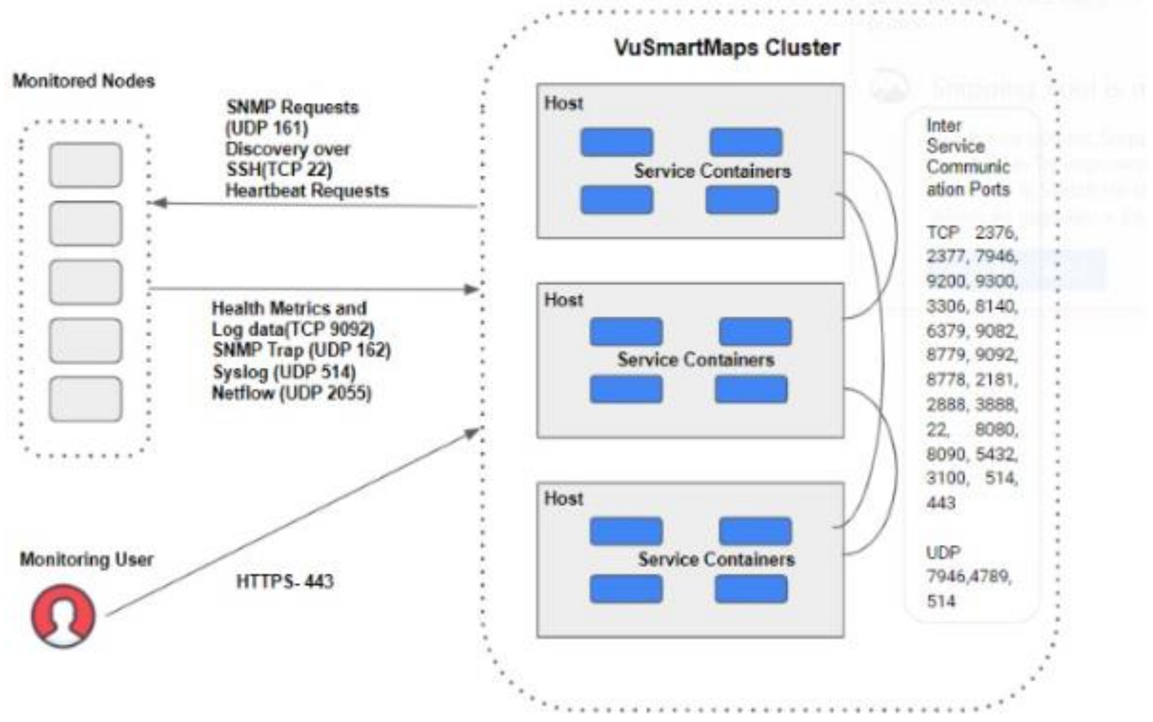
VuSmartMaps



Introduction:

This document explains the steps to install vuSmartMaps using docker containers.

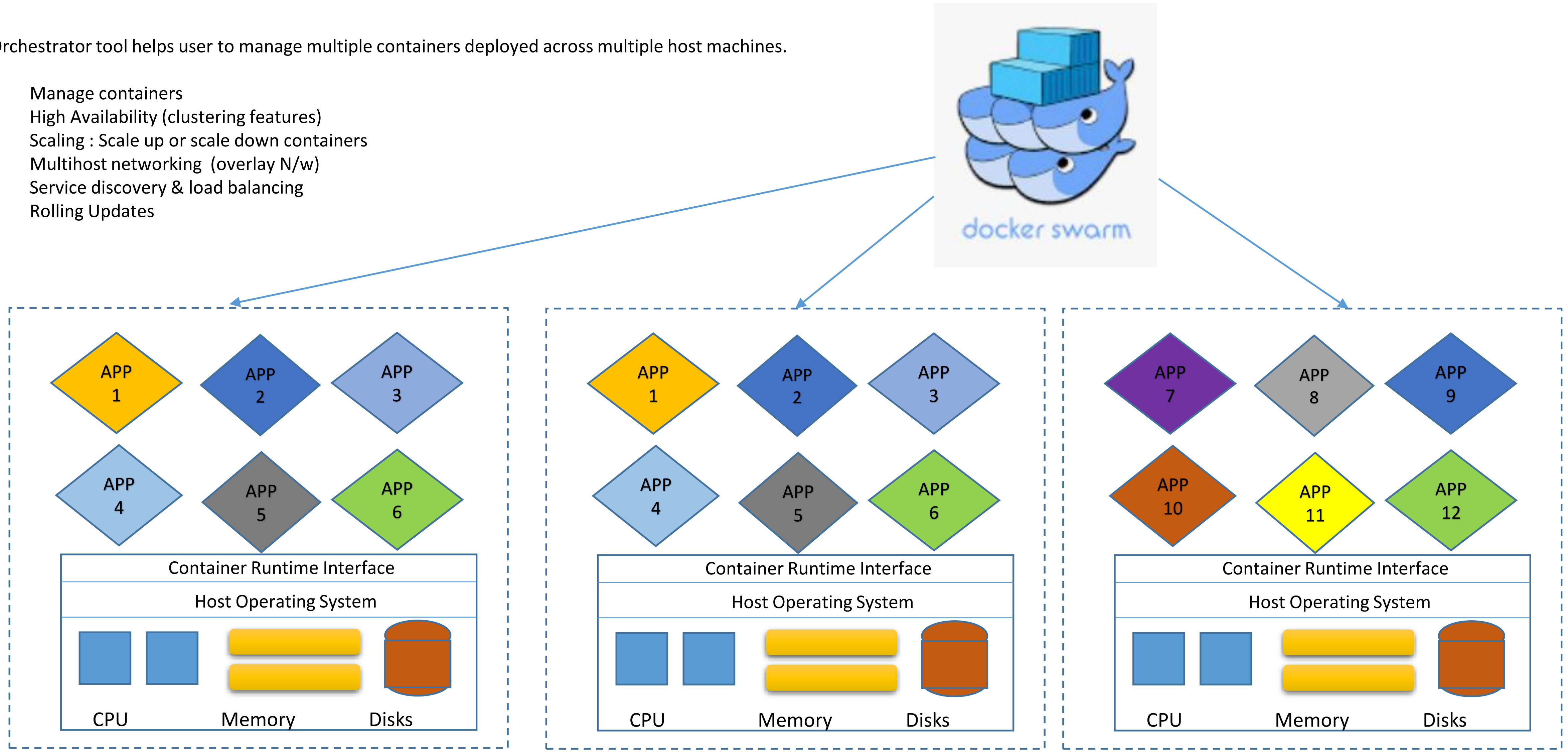
The VuSmartMaps container infrastructure uses Docker services on multiple nodes orchestrated using Docker swarm.


VuSmartMaps Cluster
Monitored Nodes
SNMP Requests (UDP 161) Discovery over SSH(TCP 22) Heartbeat Requests
Health Metrics and Log data(TCP 9092) SNMP Trap (UDP 162) Syslog (UDP 514) Netflow (UDP 2055)
Monitoring User
HTTPS- 443
Host
Service Containers
Host
Service Containers
Host
Service Containers
Inter Service Communication Ports
TCP 2376, 2377, 7946, 9200, 9300, 3306, 8140, 6379, 9082, 8779, 9092, 8778, 2181, 2888, 3888, 22, 8080, 8090, 5432, 3100, 514, 443
UDP 7946,4789, 514

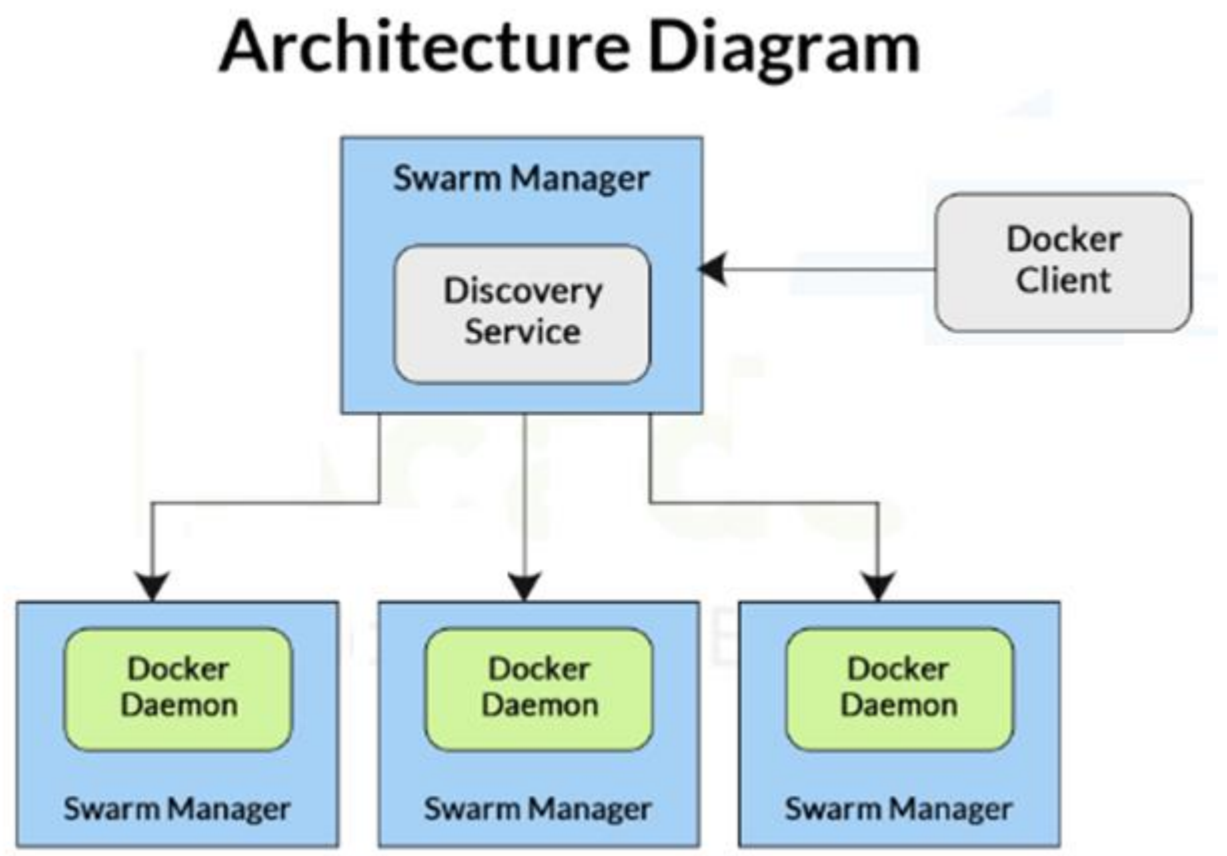
Orchestration

Orchestrator tool helps user to manage multiple containers deployed across multiple host machines.

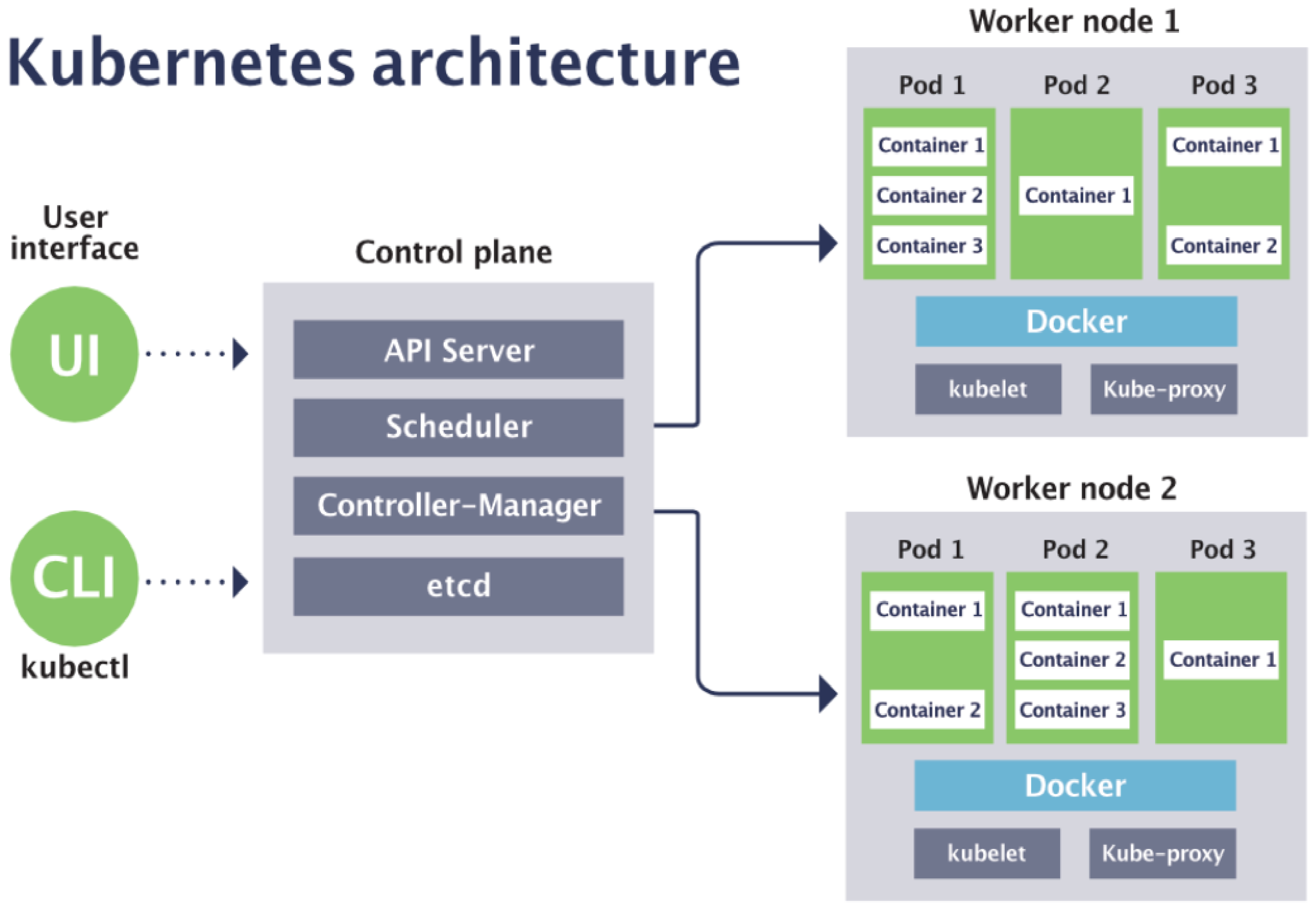
- Manage containers
- High Availability (clustering features)
- Scaling : Scale up or scale down containers
- Multihost networking (overlay N/w)
- Service discovery & load balancing
- Rolling Updates



Orchestration using Docker Swarm



Orchestration using Kubernetes



Orchestrator tool helps user to manage multiple containers deployed across multiple host machines.

- Manage containers
- High Availability (clustering features)
- Scaling : Scale up or down containers
- Multihost networking (overlay N/w)
- Service discovery & load balancing
- Rolling Updates

Questions?