# Docker & Kubernetes

from technical aspects

- What it is
- How it is intended to be used
- What advantage it brings to table
- Basic Command
- Dockerfile
- Demo

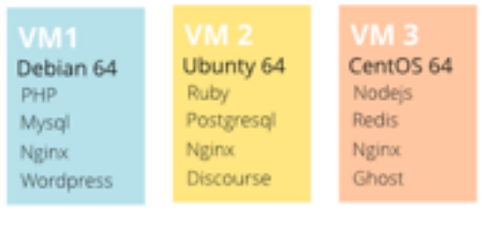
# What Docker is

- Renaissance of Linux Container Technology
- Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in.

flockport

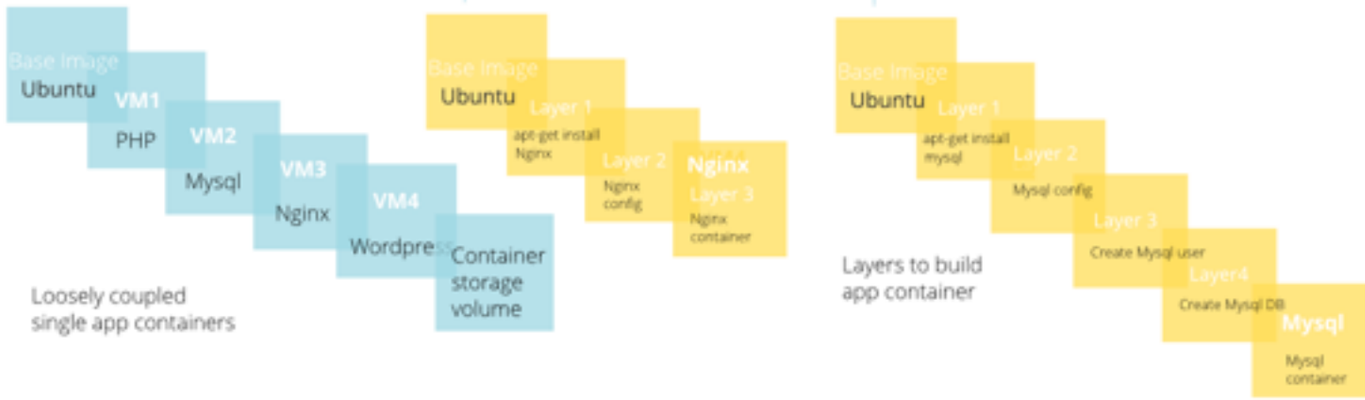
# Key differences between LXC and Docker

## LXC




- Filesystem neutral
- Containers are like VMs with a fully functional OS
- Data can be saved in a container or outside
- Build loosely coupled or composite stacks

## Docker



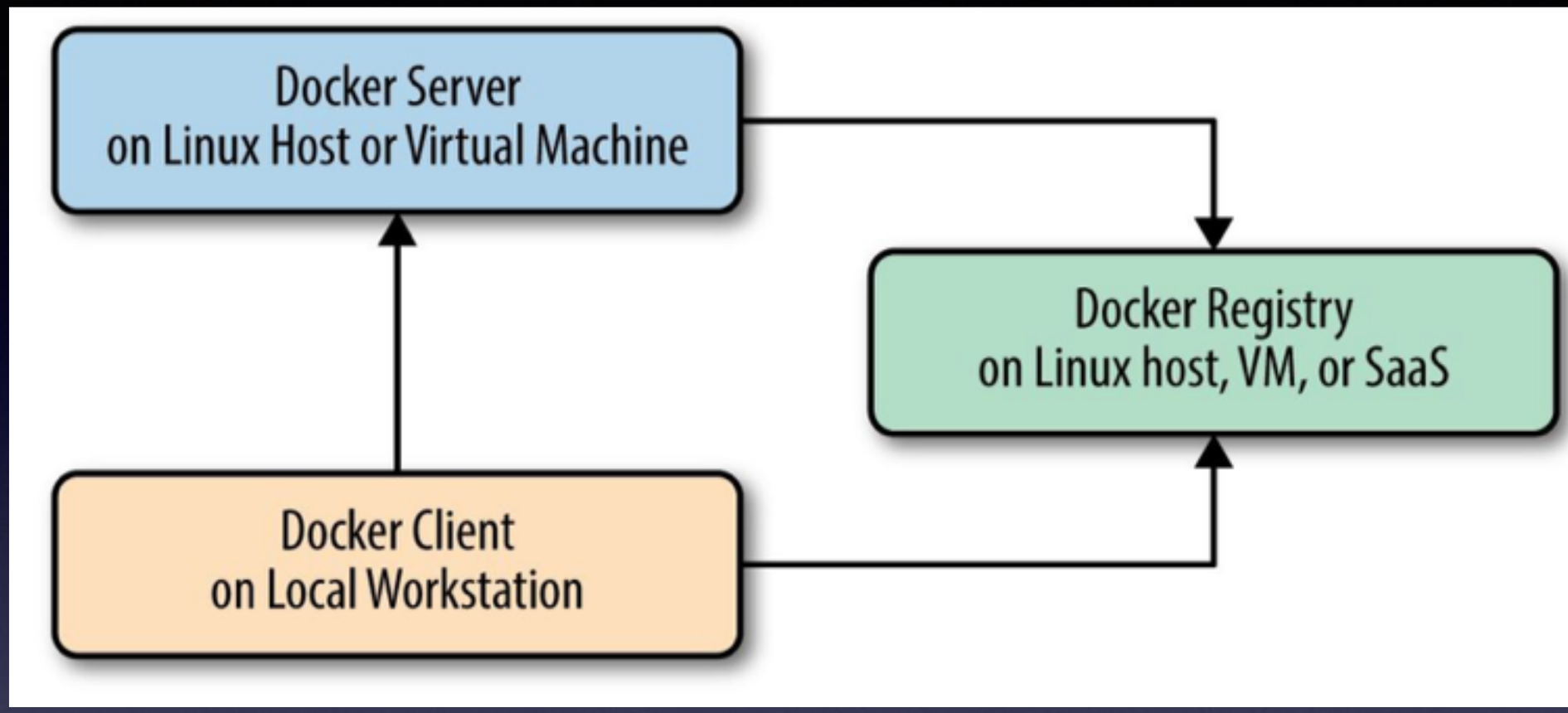

- Containers are made up of read only layers via AUFS/Devicemapper
- Containers are designed to support a single applicaton.
- Instances are ephemeral, persistent data is stored in bind mounts to host or data volume containers

- Lightweight:  Containers running on a single machine all share the same operating system kernel so they start instantly and make more efficient use of RAM. Images are constructed from layered filesystems so they can share common files, making disk usage and image downloads much more efficient.

- e.g: A quick test on Docker 1.4.1 reveals that a newly creatively container image takes 12 kilobytes space, a image usually take hundreds of MBs.

- Secure:  Containers isolate applications from each other and the underlying infrastructure while providing an added layer of protection for the application.

- Open:  Docker containers are based on open standards allowing containers to run on all major Linux distributions and Microsoft operating systems with support for every infrastructure.

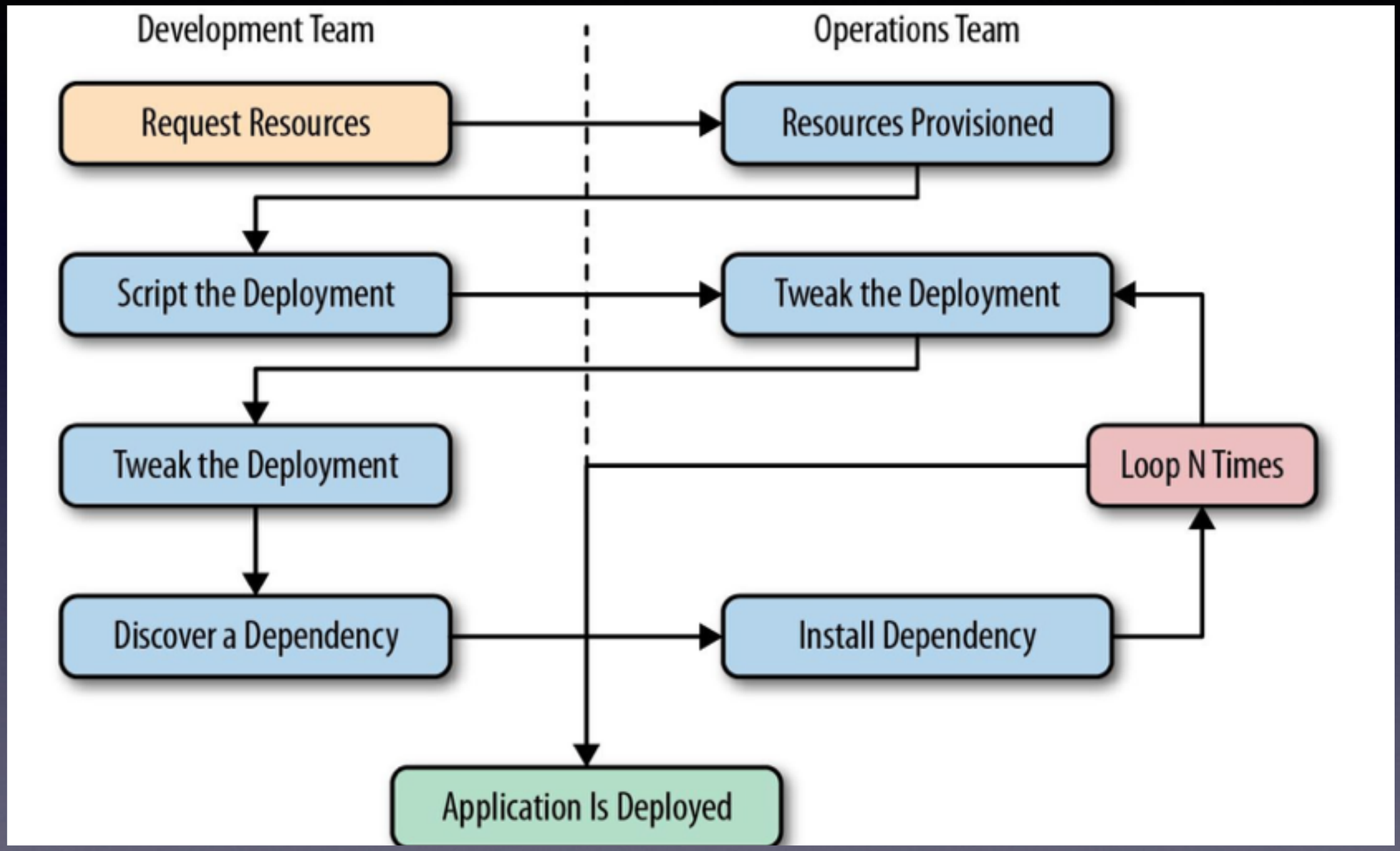



# Architecture

1.C/S 2.Single Binary 3. The original TCP port that docker was configured to use was 4243/2375/2376

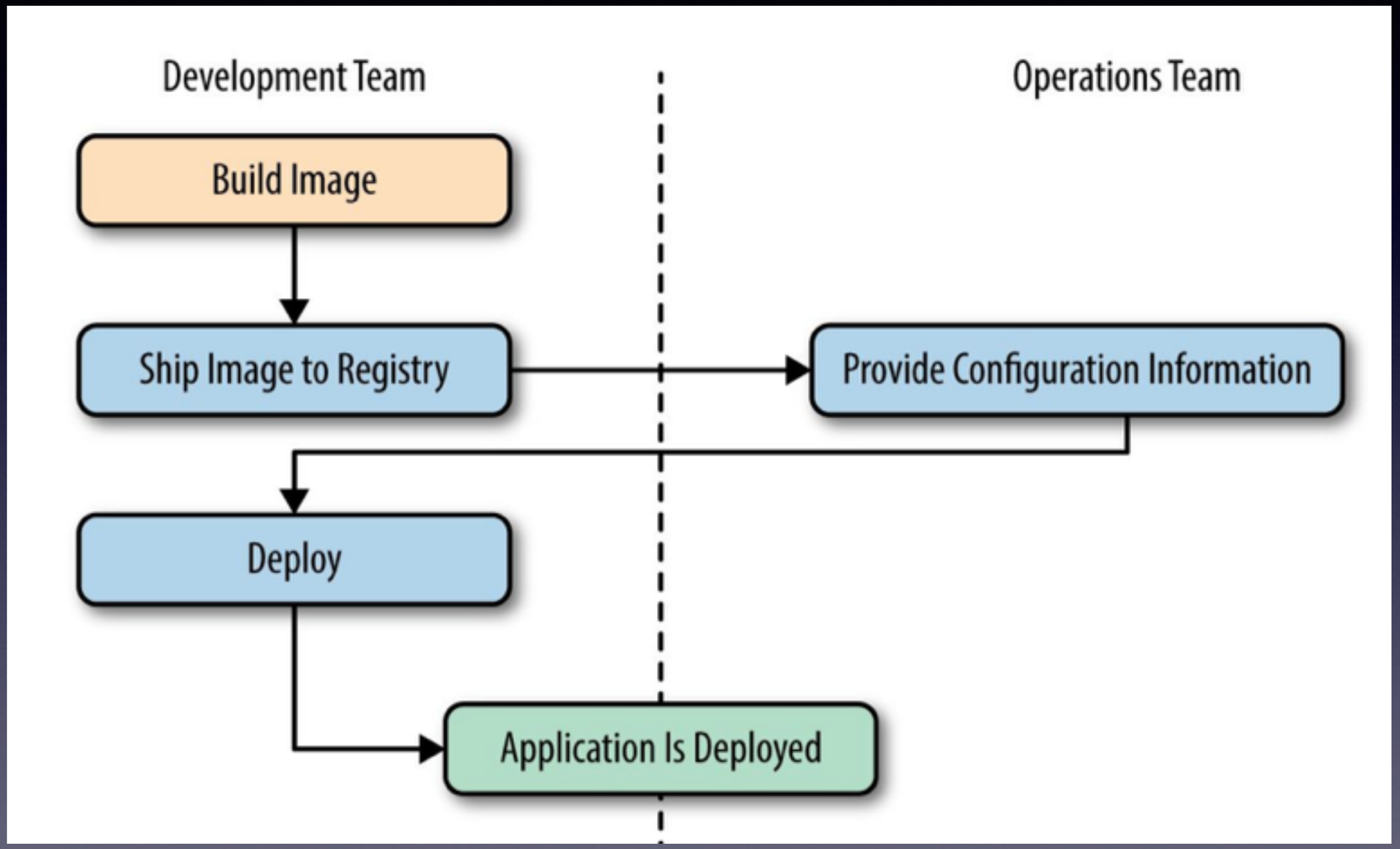
- The Docker command-line tool and docker -d daemon talk to each other over network sockets
- You can have the Docker daemon listen on one or more TCP on Unix Sockets.
- Command-line tool : Build a container image, Pull Images from registry, start a container, Retrieve docker logs

# How docker is used




History without docker

# How docker is used




With help of docker

# Basic Command

Container, Info, Image, Registry

# Containers

- docker create: creates a container but does not start it
- docker run: creates and starts a container
- docker stop: stops it
- docker rm: deletes a container
- docker attach: will connect to a running container

# Info

- docker ps: shows running containers
- docker logs: gets logs from container
- docker inspect looks at all the info on a container
- docker events: gets events from container
- docker port: shows public facing port of container
- docker top: shows running processes in container

# Executing Commands

- docker exec: to execute a command in container

- e.g: docker exec -it foo /bin/bash

# Images

- docker images: show all images
- docker import: create an image from a tarsal
- docker build: creates image from Dockerfile
- docker commit: creates image from a container
- docker rmi: remove an image

# Registry & Repository

- docker login: to login to a registry
- docker search: searchers registry for image
- docker pull: pulls an image from registry to local
- docker push: pushes an image to the registry from local machine

# Dockerfile

- A Dockerfile is a configuration file that contains instruction for building a Docker image.
- Provides a more effective way to build images compared to using docker commit
- Easily fits into your continuous integration and deployment process

# Dockerfile Instructions

- Instructions specify what to do when building the image
- FROM instruction specifies what the base image should be
- RUN instruction specifies a command to execute

```
FROM    centos:centos6

# Enable EPEL for Node.js
RUN     rpm -Uvh http://download.fedoraproject.org/pub/epel/6/i
# Install Node.js and npm
RUN     yum install -y npm

# Bundle app source
COPY . /src
# Install app dependencies
RUN cd /src; npm install

EXPOSE  8080
CMD ["node", "/src/index.js"]
```

# demo:dockerfile

# Demo 1

build and deploy one web server so that everyone can access

# Demo 2

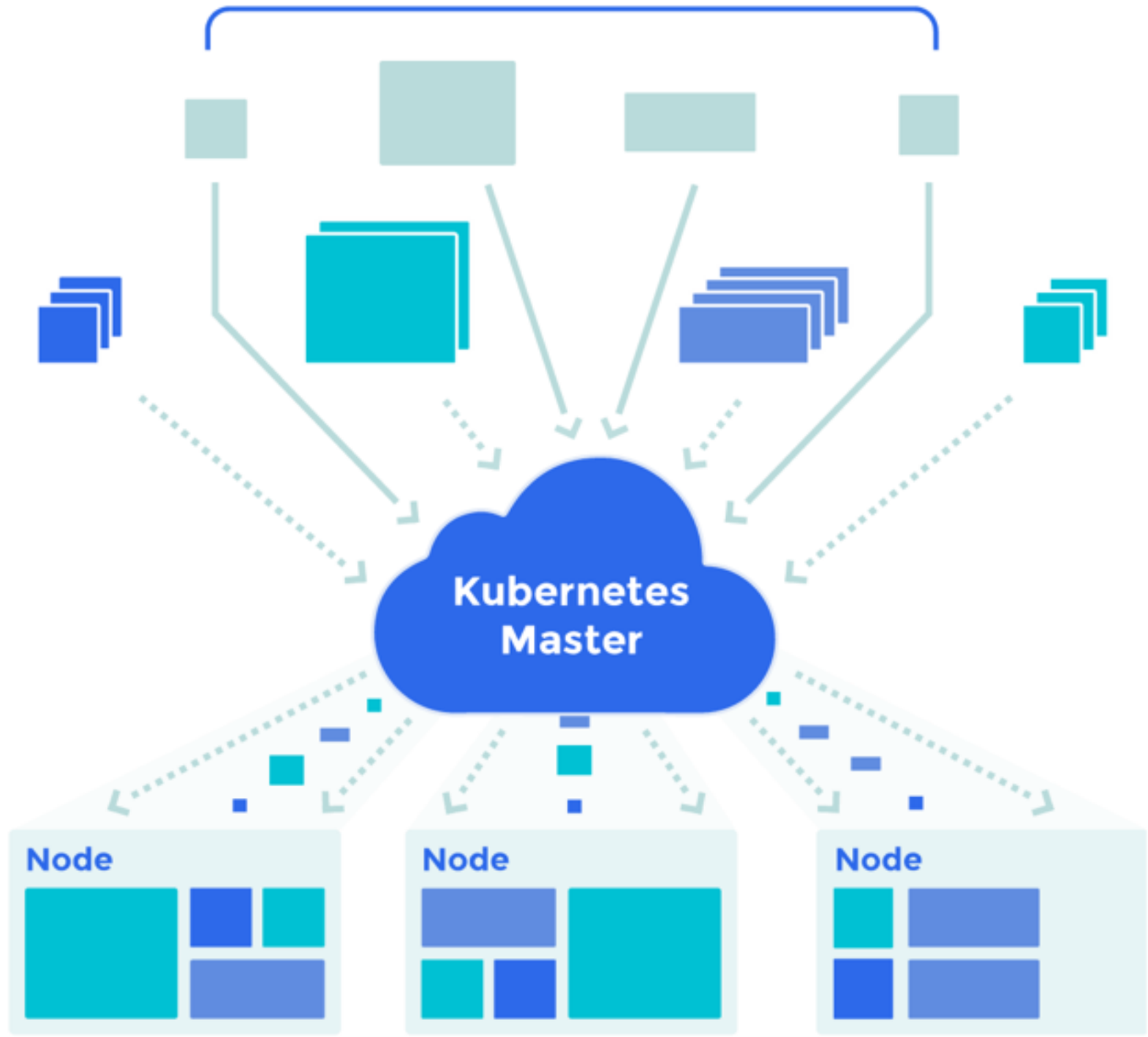
Get Spark run immediately and play around with it

# Kubernetes

# What is Kubenetes

- Service for Container Cluster Management
- Open Sourced By Google
- Used to manage Docker container as default implementation

An ocean of
user containers



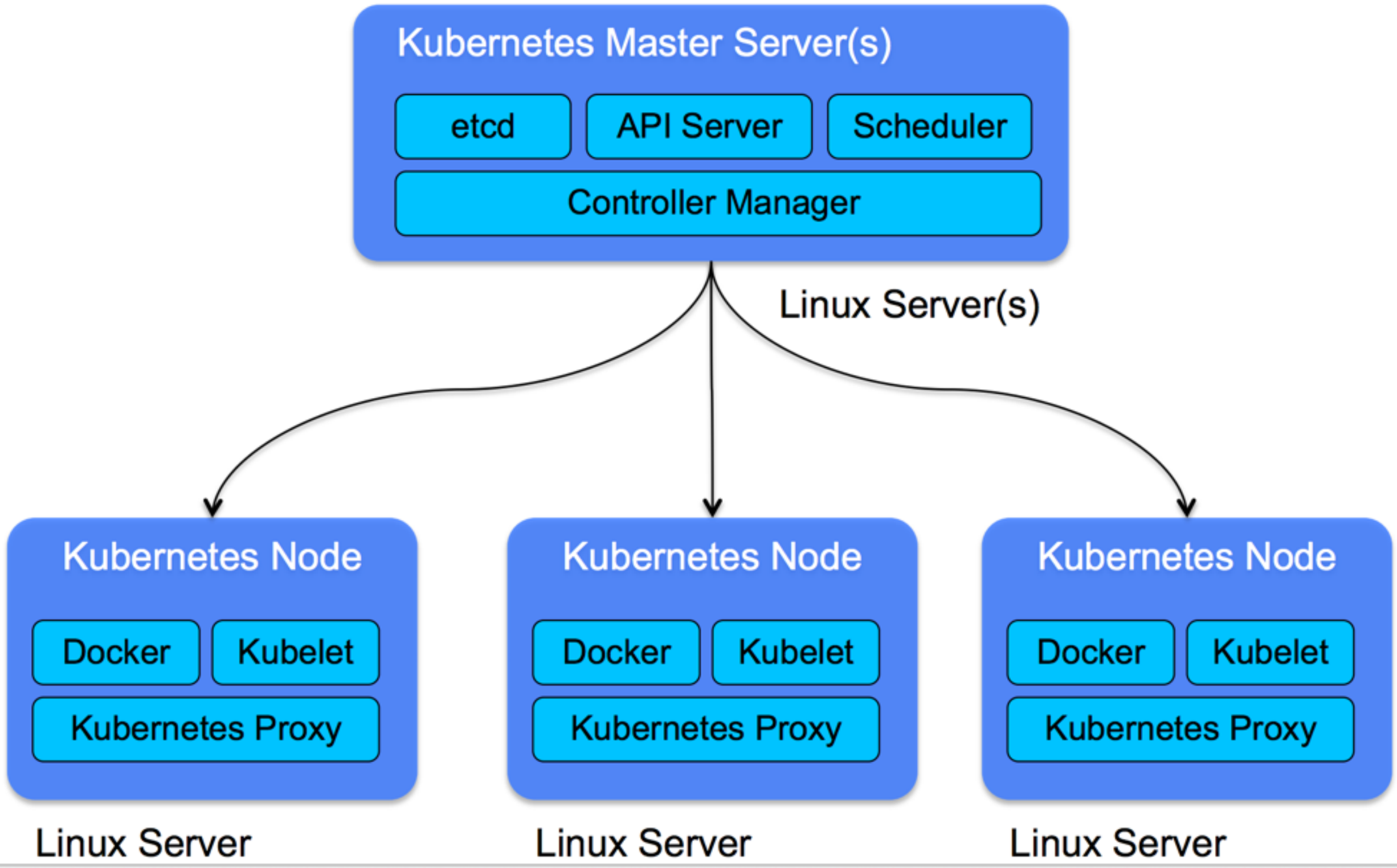

Scheduled and packed
dynamically onto nodes

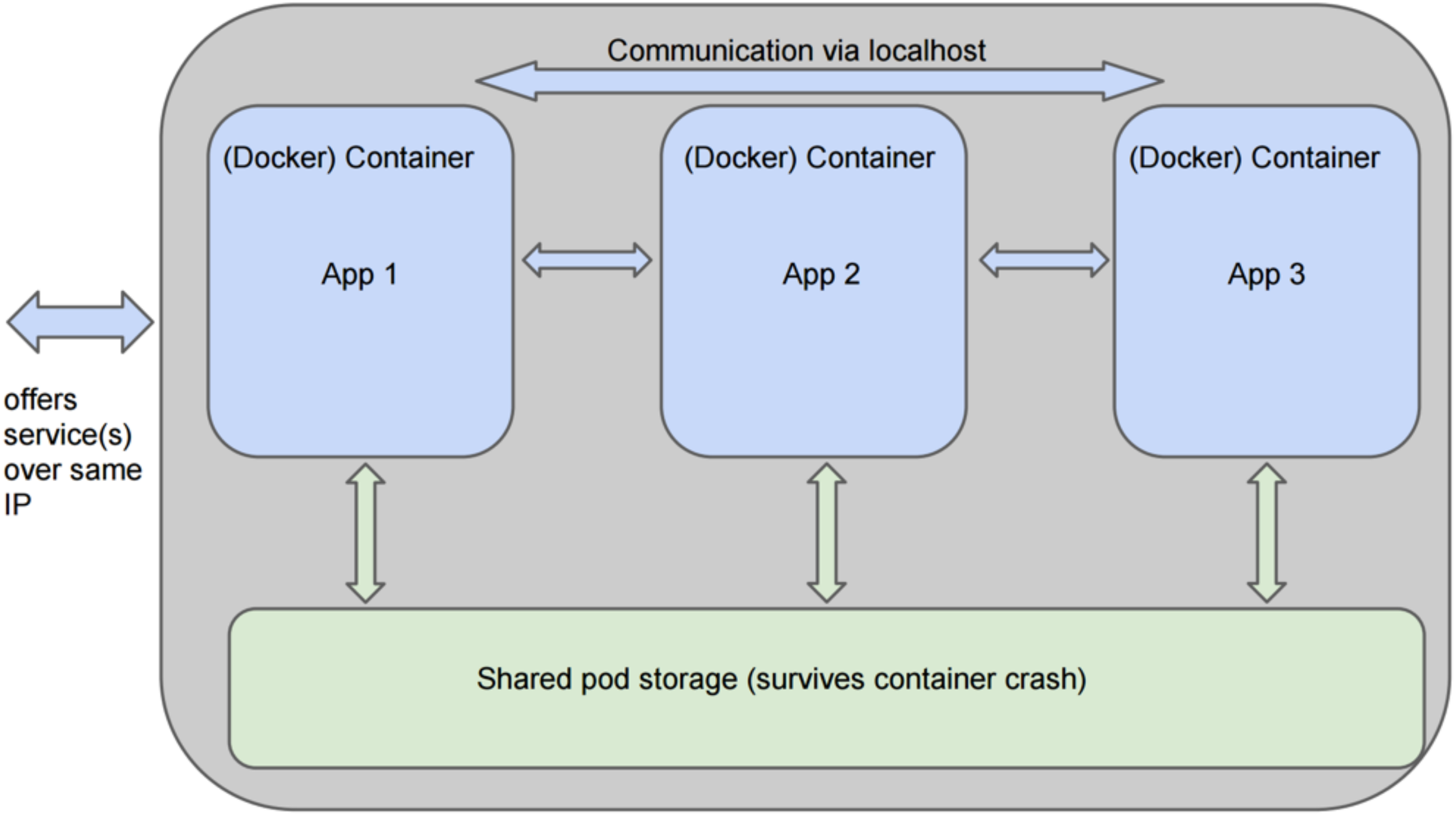
# Key Concept

- Master
- Minion
- Pod: Grouping for Containers
- Service and Labels
- Kubernetes Node

# Master

- Master maintains the State of the Kubernetes Server runtime
- It is the point of entry for all the client call configure and manage Kubernetes components like Minions, Pods, Containers
- Made up of: API Server, Scheduler, Registries, Storage


Kubernetes Master Server(s)
etcd
API Server
Scheduler
Controller Manager
Linux Server(s)
Kubernetes Node
Docker
Kubelet
Kubernetes Proxy
Linux Server
Kubernetes Node
Docker
Kubelet
Kubernetes Proxy
Linux Server
Kubernetes Node
Docker
Kubelet
Kubernetes Proxy
Linux Server


Communication via localhost
(Docker) Container
App 1
(Docker) Container
App 2
(Docker) Container
App 3
offers service(s) over same IP
Shared pod storage (survives container crash)

# Minion

- Represents the Host/VM where containers are created
- Minion is identified with a name and a Host name
- Key Components of a Minion: Group of PODs