

VM: can run on windows,unix,linux. Containers can run on windows,unix based on technology used.

VM requires hypervisor (monitor,create,destroy VMs).

Hypervisor-type1(directly run on host m/c hardware-ESX,XEN).Type-2: Use s/w to run on top of host m/c-VMware, virtualbox.

-Can run any guest OS regardless of host OS. Install full OS. Incur startup, shutdown times.

Have variety file system config,process, n/w config.

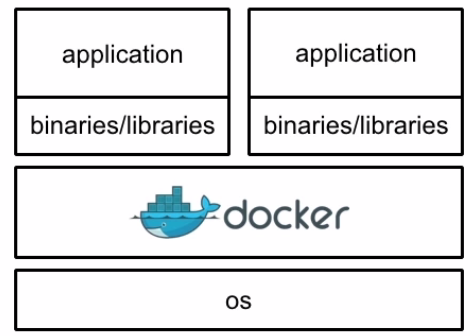
Containers significantly low overhead than VMs, as they have share kernel with host one. So, easy to start-stop extremely fast**. Generally time taken to start is = Containerized process time to start.**

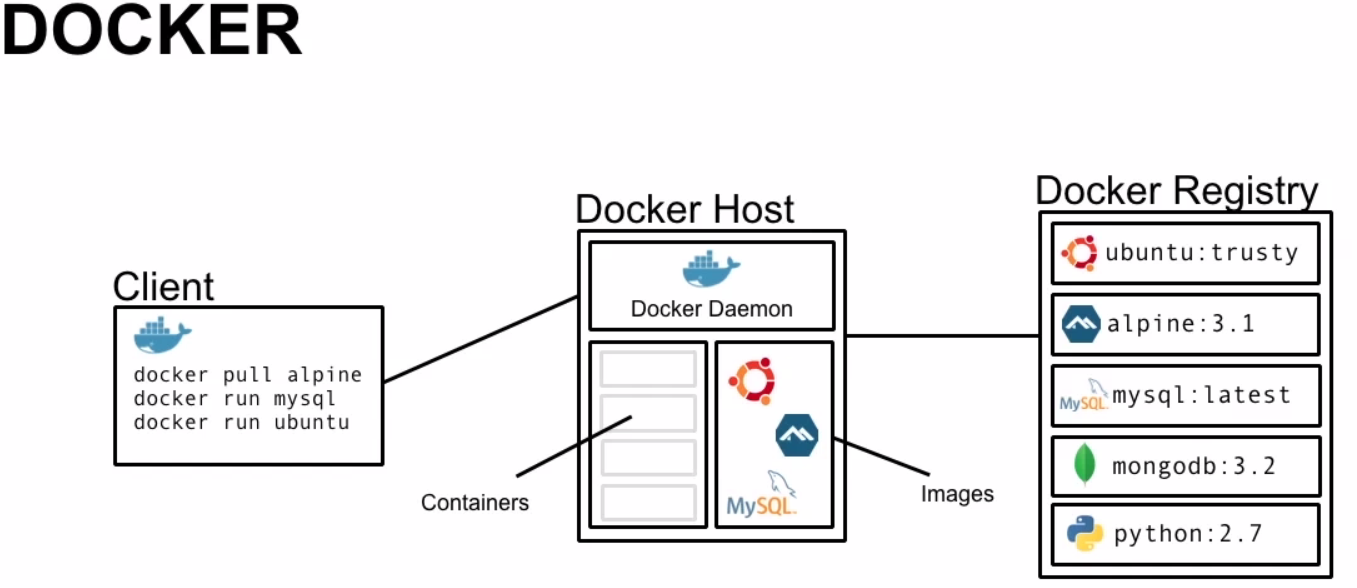
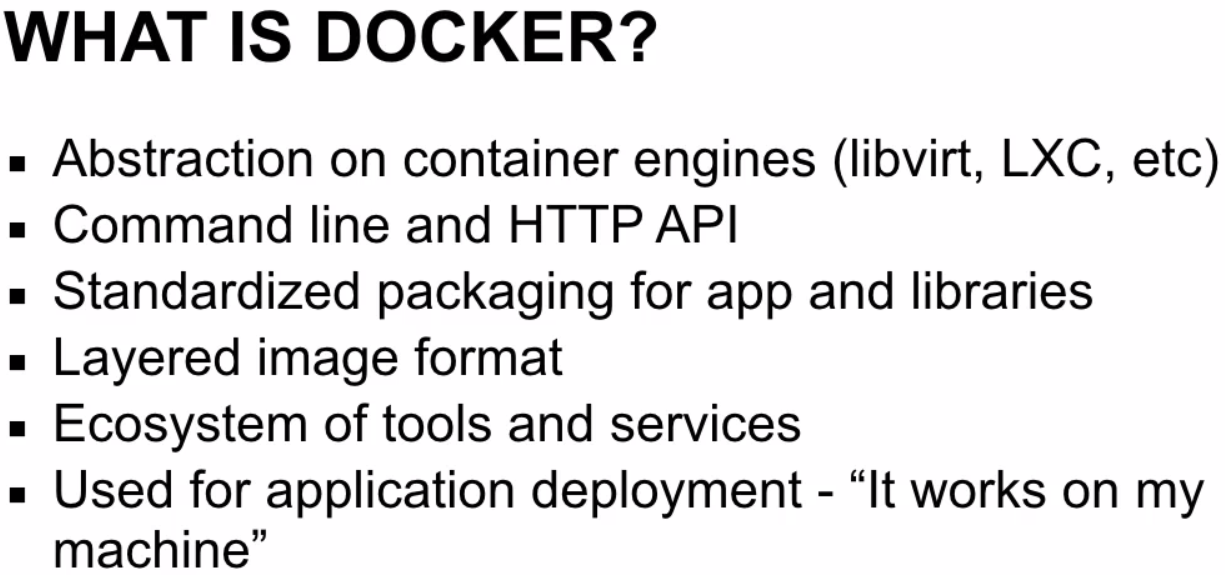
Disadv: They share same OS kernel of host.So they must use same OS.

For ex: Docker containers must run Linux distribution, and cannot run windows or unix distribution.

Docker built as abstraction top of linux containers allows easy programmatic creation, distribution of container images.., and as well as launching and deploying containers themselves.

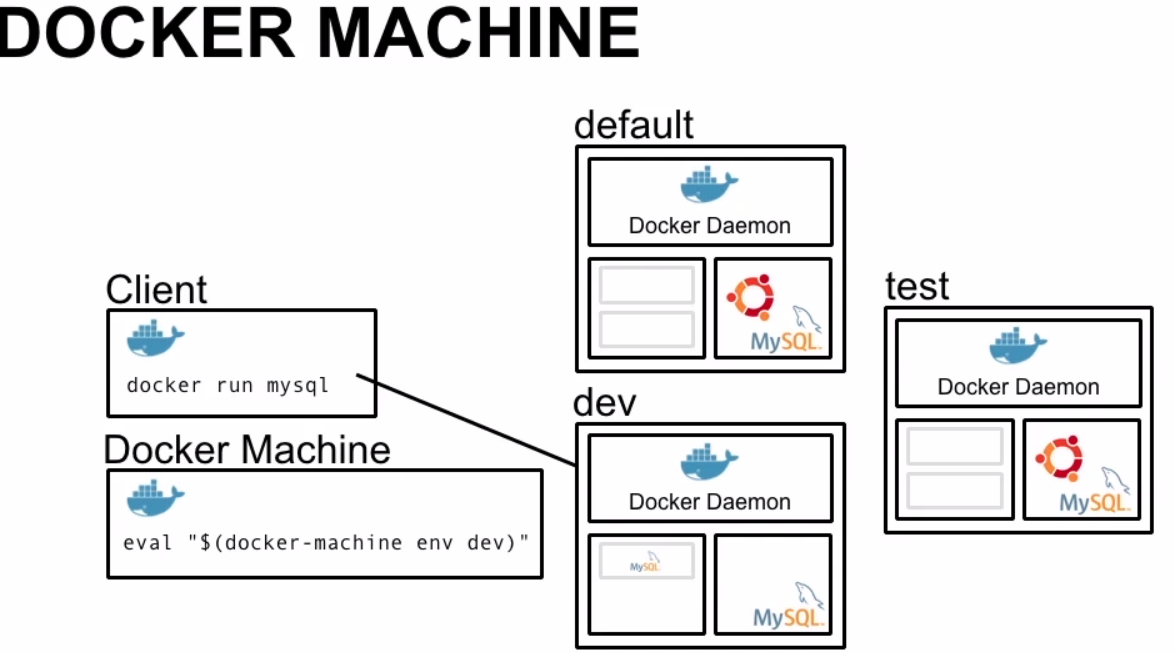
Docker containers ship with containerized applications. So deployment remains same regardless of running on technology or linux distribution running in container.



Docker containers ship with containerized application, So deployment remains same regardless of technology or linux distribution running on container.

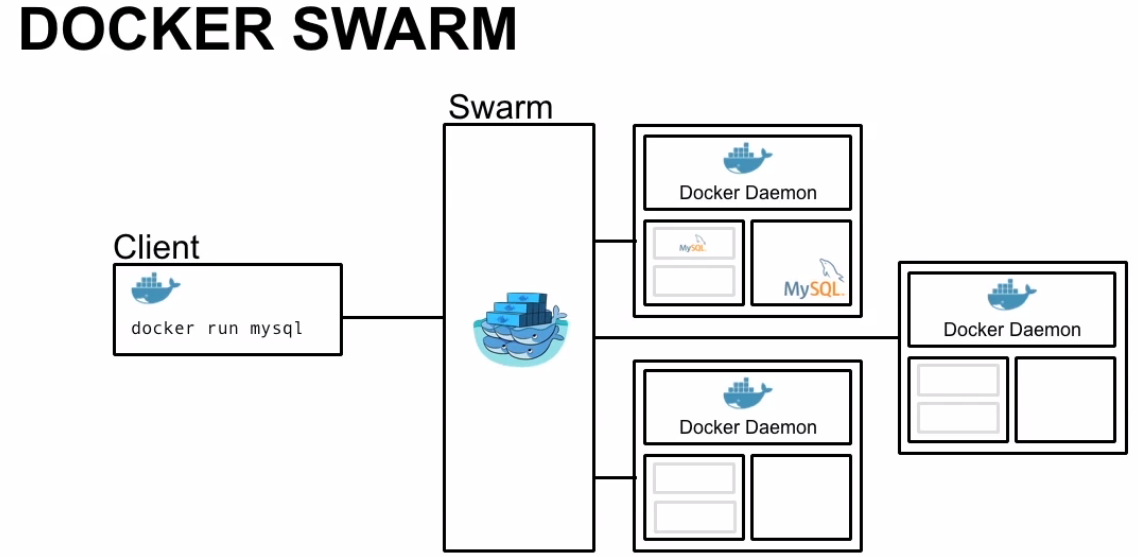
Docker is client-server application.Docker binary doesn’t interact with containers or images directly.Docker binary simply accepts command line calls and sends to Docker daemon via HTTP or TCP socket.If both components are on same host machine, this communication is also used.

Docker daemon is server host,which actually interact with images,containers. Docker daemon also host images,containers.Docker host exposes HTTP API, that can be used by other programs to manage containers and images.



Docker-machine is command line tool used to create, manage Docker hosts.

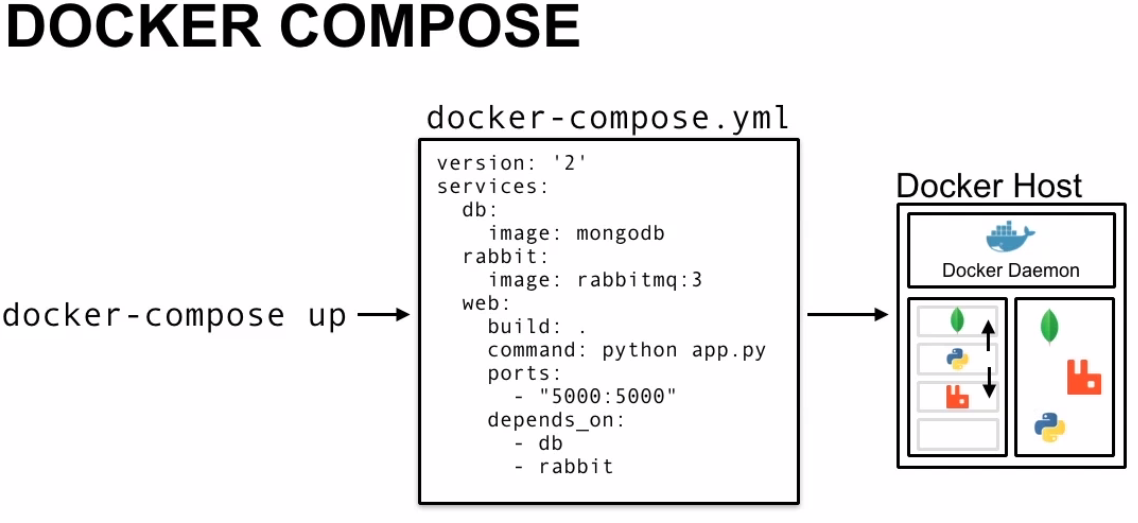
Docker-machine allows non-linux users to run docker-client locally on their localhosts and issue docker commands on remote VM using hypervisor or cloud provider.

EX: Docker-machine create dev creates fresh Docker host dev, that can be controlled using local Docker client. We can set proper environmental variable using docker-machine to point local client to deamon and we can issue commands. 

Clustering tool allows manage multiple docker hosts as single docker host.

Docker swarm publishes API almost similar to normal docker API which means most tool is compatible.

We can use Docker client API to run containers with docker swarm, and easily scale up multiple hosts.



Sometimes we need to containerize applications with multiple hosts.

Using Docker-compose, we can specify group of containers and wire containers together.

EX: Runs different container mangoDB,rabbitmq,python web app.

Running Docker-compose, will produce 3 applications network together..

Docker-compose is great tool for deploying multi container application

Since Docker requires, Linux based kernel to run- MacOS, Unix, Windows cannot run Docker natively.

In Early days of docker,we have tool called boot2docker which is stripped down linux image with docker installed.Boot2docker allows users to issue Docker commands on mac to remote docker Daemon running virtual machine.

More recently docker packed doecker engine, Kitemachine, docker-machine, docker-compose as a product called docker Toolbox, which can be easily installed on Mac m/c.Boot2docker is still used-which is managed by docker-machine.

Docker needs hypervisor to run boot2doker VM.Virtualbox is installed.

Docker Quickstart Terminal: It will start boot2docvker VM on virtualbox using docker-machine.It will create default VM for you.

There is VM machine, default running and docker is configured to control it running.Docker-machines will run with IP:192.168.99.100. When we refer to dockerhost, this is ip of machine.For docker m/c it will refer to localhost.

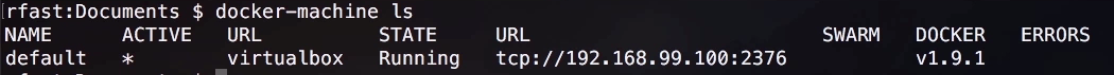
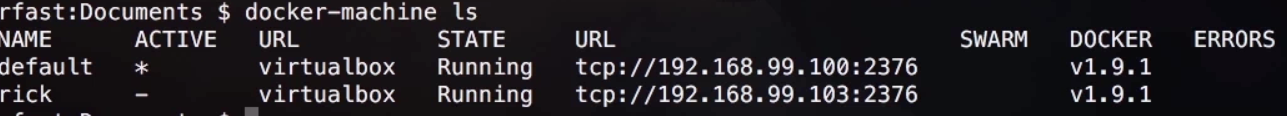
**Docker-machine:**

-A tool that ships along with docker toolbox

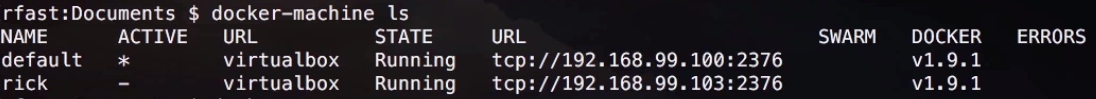
-Allows users on non-linux based users of mac, windows to use launch multiple Docker instances from host machine.

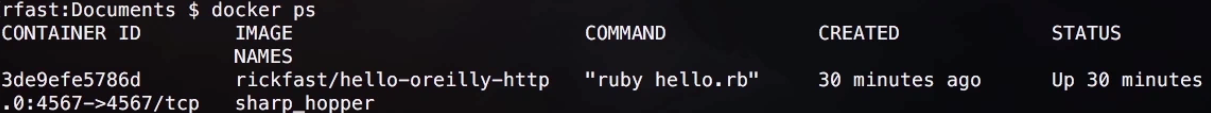
-docker-machine allows users to use docker regardeless of their OS.

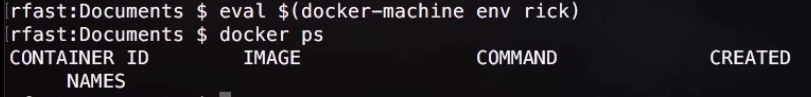
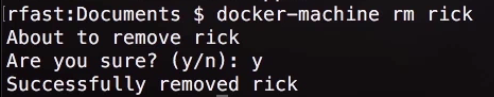
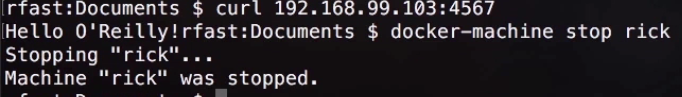
-Nice tool for launching VMs on hypervisors,cloud



When we install docker toolbox,Virtual box will be installed as well.Virtualbox will be used to launch VMs. Docker supports hypervisors, cloud providers-digital ocean,AWS,azure.



Since we have docker client on local machine, we can issue commands on remote docker-daemon as if we are on local machine.

Oops!! We have containers running on default machine.We wont be able to issue commands to new machine rick, because we didn’t tell local Docker client that we want control over new remote docker daemon running on new docker VM rickNow we are connected to new docker-machine.

Once we are done, we can issue command: docker-machine stop rick.



Here: we created Ubuntu container that run PWD command (/)

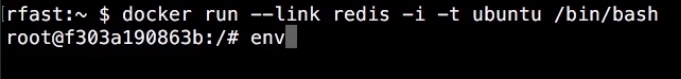


But here pwd output differs.

It has its own userID space, networking, process IDs.

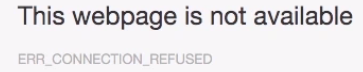
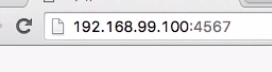
Now, our DB is running.

Start web application now: Web application needs to connect to redis to store and retrieve data.

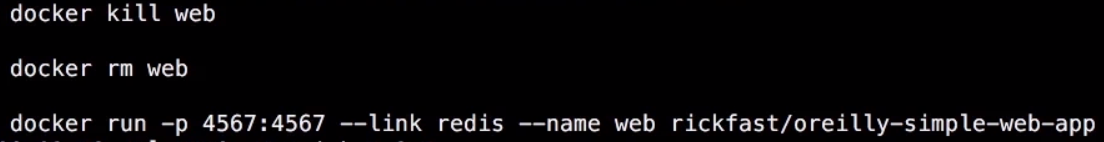
Start container and link to DB. Link allows containers to find each other by IP and Port.

NAME\_PORT\_PORTEXPOSED\_TCPADDR.So we have to know name of container,and port we want to connect to..

Our simple appl. Is used to connect to redis..Connect and store data.



Issue: map port

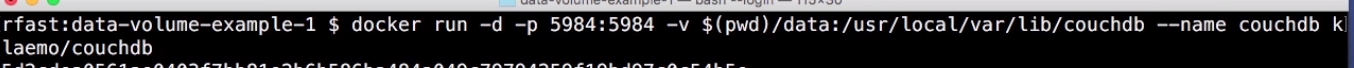


The dats in redis database is ---(blank).

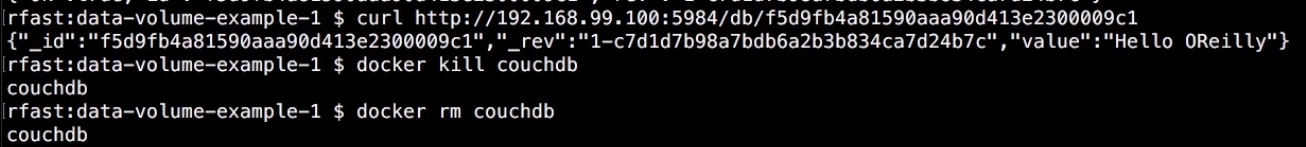
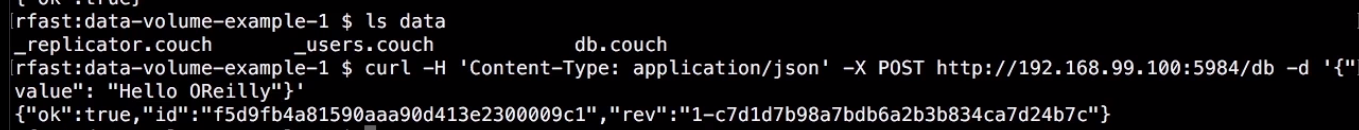
Our web application has other end point…Data to load (hello).Now data is stored in redis key value store..

Map all available ports



**Data volumes:**

Since Cache DB is HTTP service, we can use CURL to create cache DB using PUT



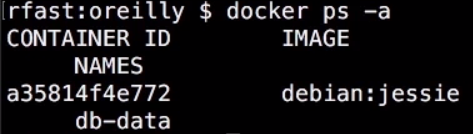
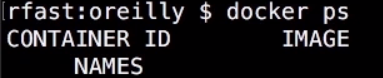
Now we remove container, and create new one using the volume.

We can now use same data on volume

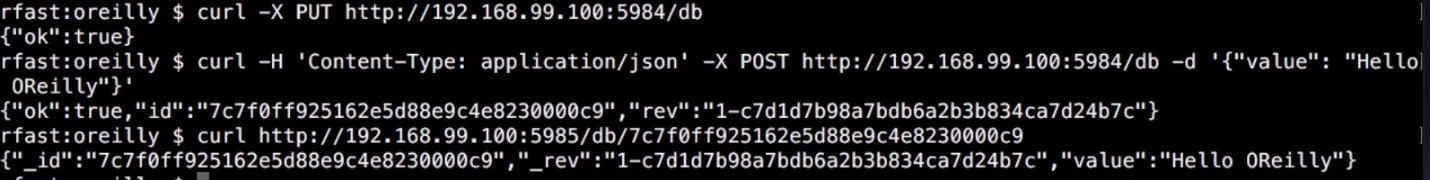
**Data Volume:** Share data between containers.(not on host)

Data Containers doesn’t run applicATIONS..But have only volume shared.

Syntax similar to docker run. We don’t need to specify path on our host machine because shared data reside on data volume container. We can simply reference directory between containers.



**Now, we can start couple on couchdb containers to utilize shared data volume.. Remember running 2 containers on shared directory is not recommended for any prod. Scenario.., especially if both are riding to data volume..**

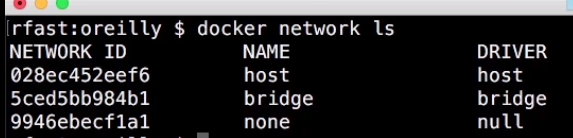
****

**For storage mgmt.: Flocker tool by Clusterized queue..**

**Networking:**

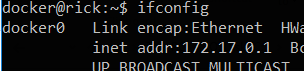
Our application may require multiple containers to function properly.

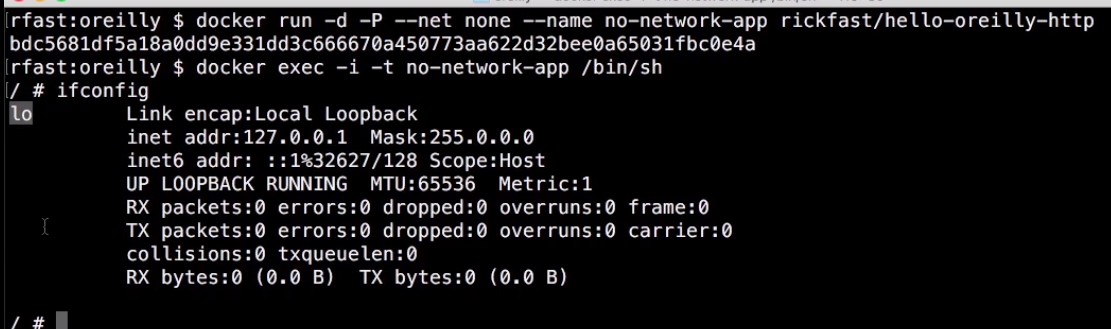
Docker by default runs 3 n/ws on docker host.



When we launch containers.., we can use one of these ..Default:**bridge.**

One container cannot talk to other unless port is mapped.



**None:**no network ..loop back n/w is only there.

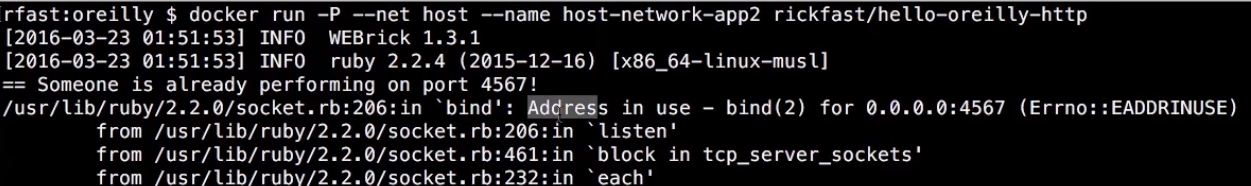
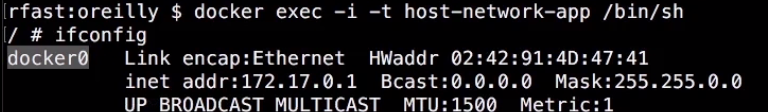
Host:Same n/w as host is using..

Running containeras on host=>No need to map ports, as they are bound to docker network directly.., because they are bound to docker host directly..

Using host n/w allows to elimate isolation bridge n/w enforses

This means..,if we run 2 applications on containers..exposing port 80..There will be confilt.

Since they are bound to same host.



Failed because another container running on host n/w is using port:4567

User defined bridge or own driver:

Start/stop Docker





Docker provides a number of networking options that allows containers to communicate with each other using std. like DNS. But, how can containers communicate with each other on separate environments??

Also, how can legacy applications interact with containers on dynamic environment??

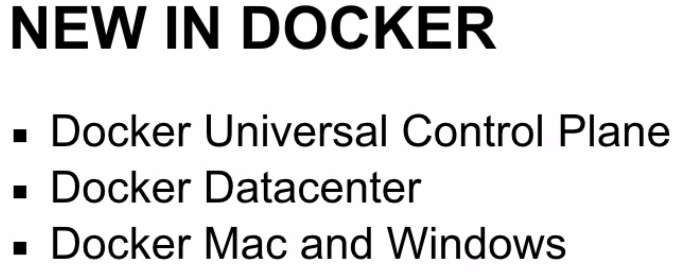
One way to manage is to use service discovery component.





Docker relies on mounts, env.varibale to configure application.Both provides security concerns

Hashy Corvault, Amazon KMS, Keywis-🡪 to store distributed secrets.



Universal Control Plane: Easy to manage workloads on cloud using **dashboard.**

Docker Datacenter: Universal dockerplane and docker registry. Provides ldap access control.

Running Docker container on production environment is easy.

Modern Cloud env. Provides elasticity, disposability. Docker is built for these kind of environments.