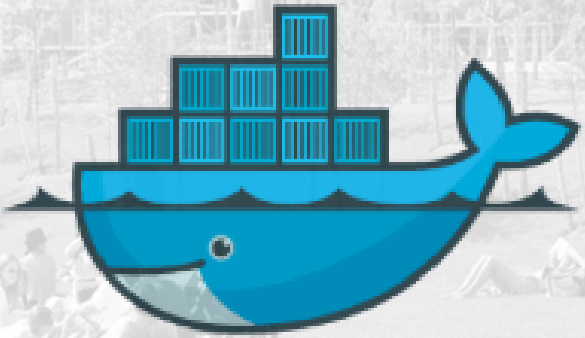
Master Class

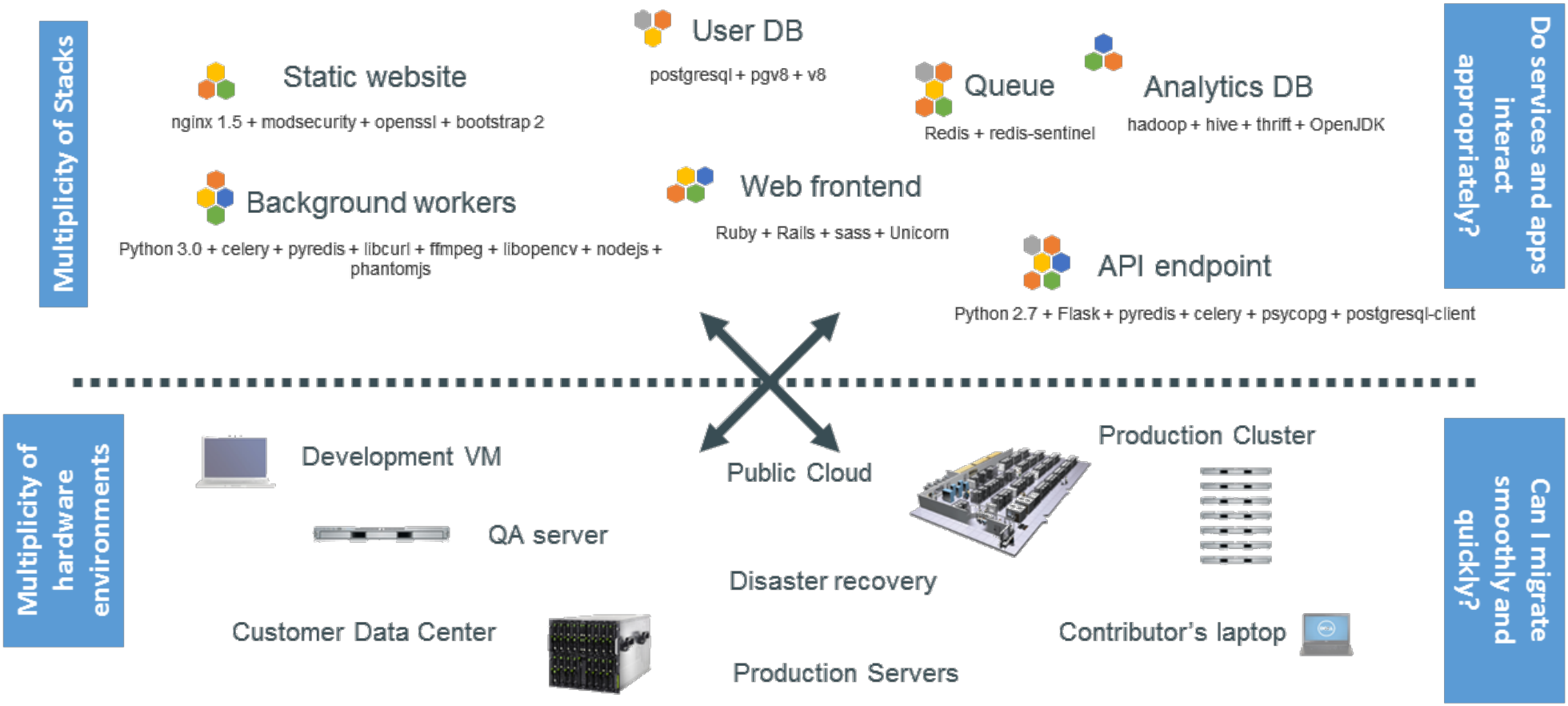
Introduction to Docker



Hopla! Software

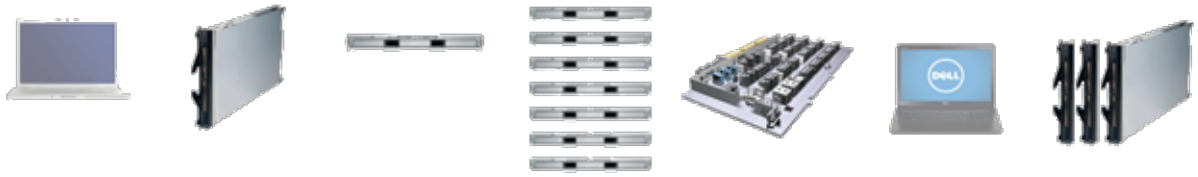
The Enterprise Software & Support Company

Beginning with a Common Problem

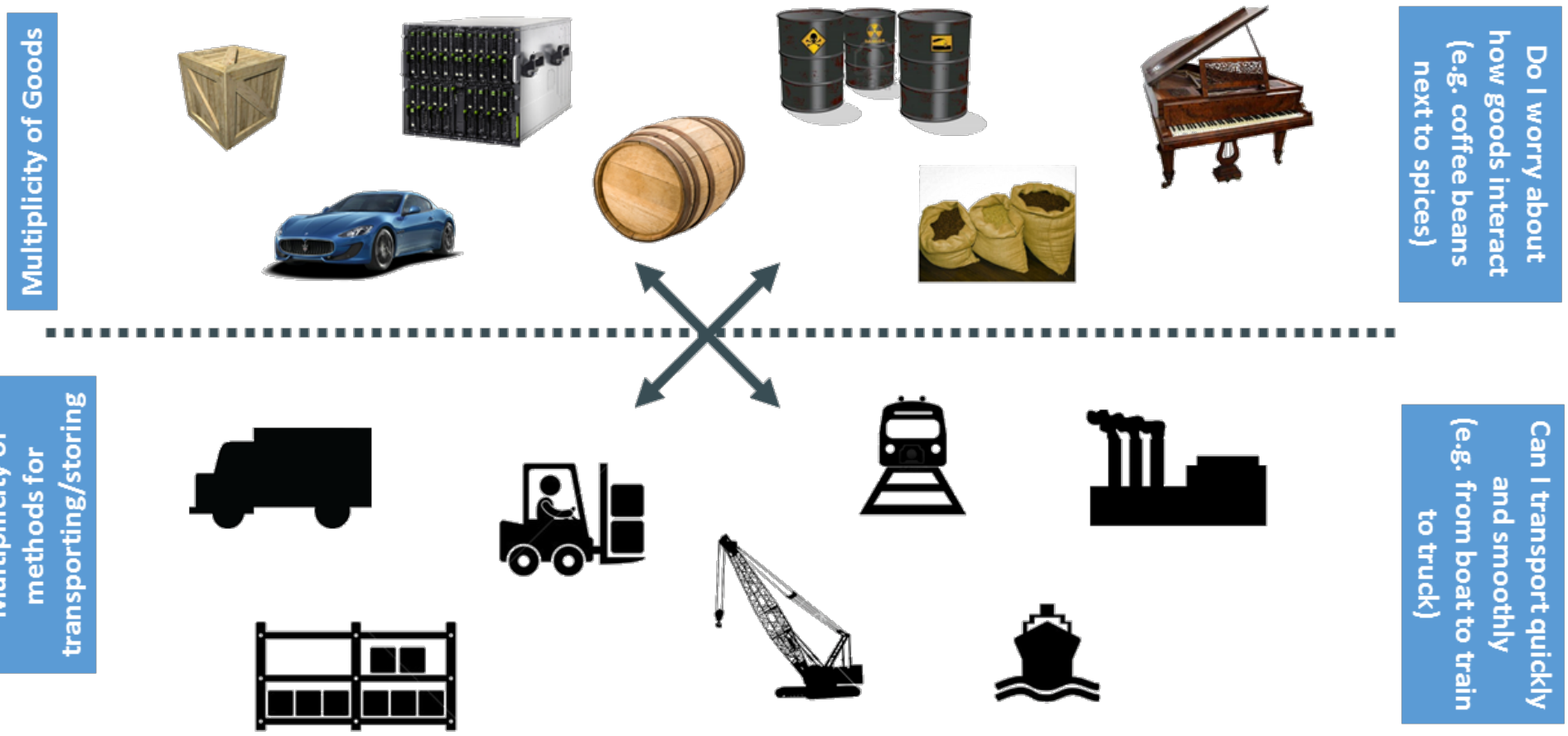


Try to resolve this Problem Matrix

	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers
Static website	?	?	?	?	?	?	?
Web frontend	?	?	?	?	?	?	?
Background workers	?	?	?	?	?	?	?
User DB	?	?	?	?	?	?	?
Analytics DB	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?

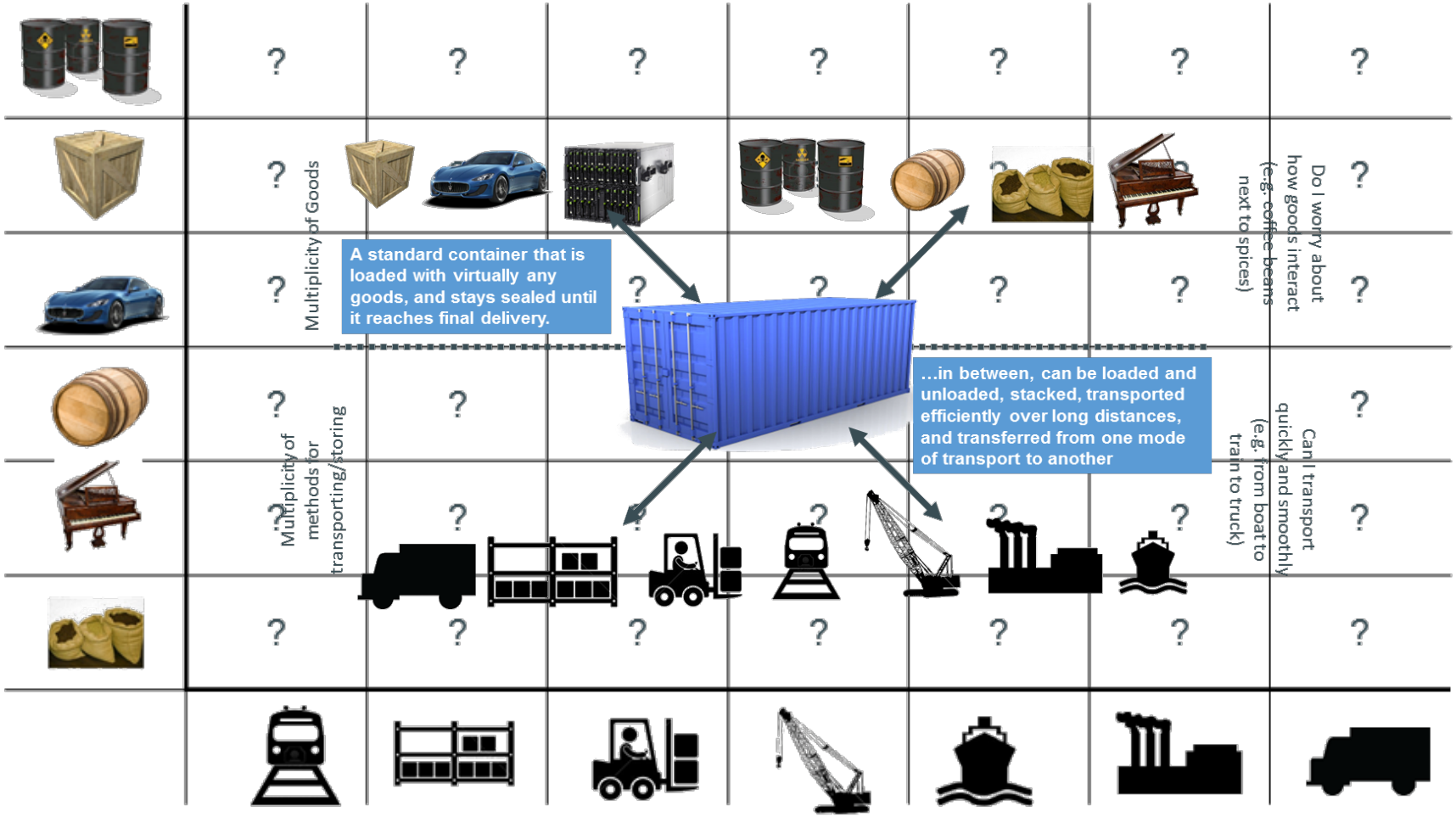


But it Looks Like an Old Problem...



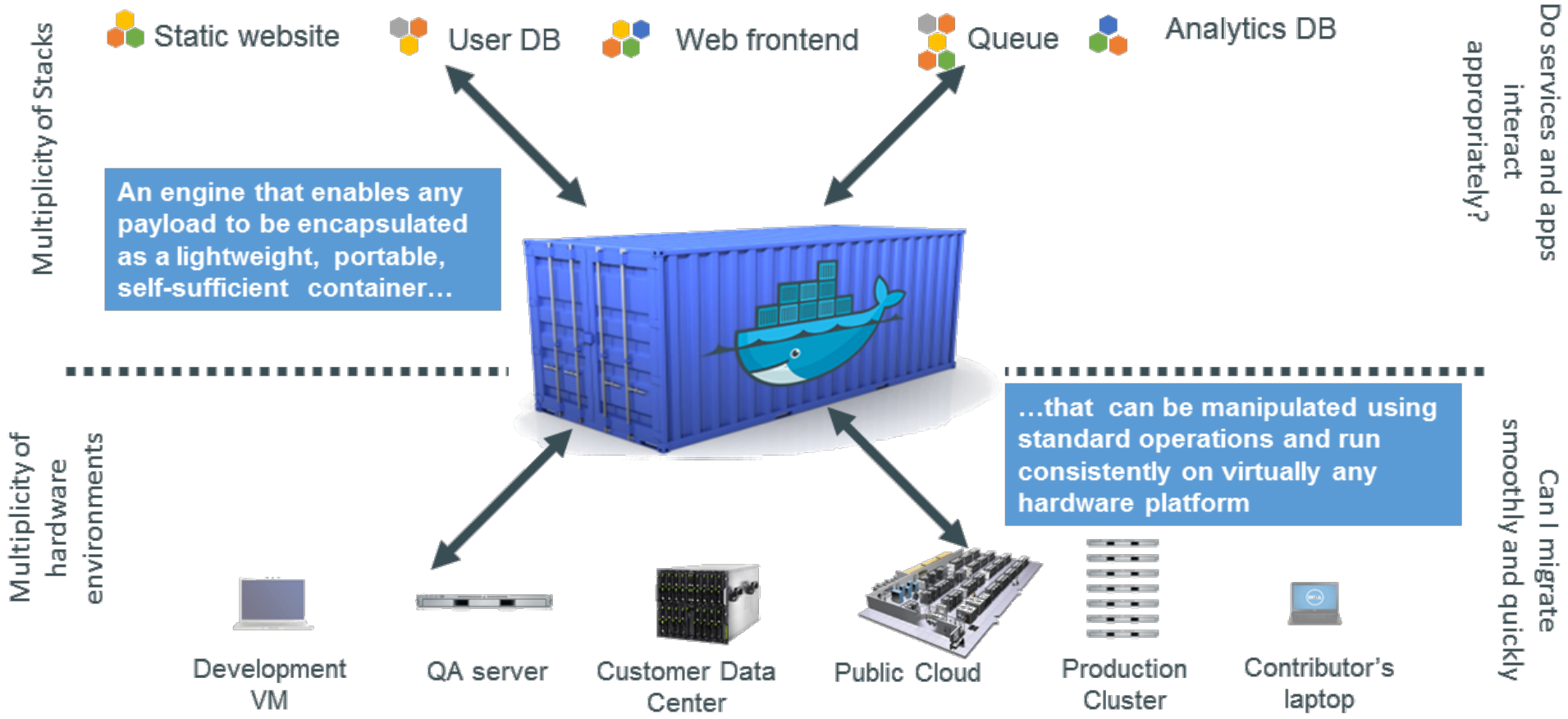
Old Problem Solved!!

Old Problem Solved!!

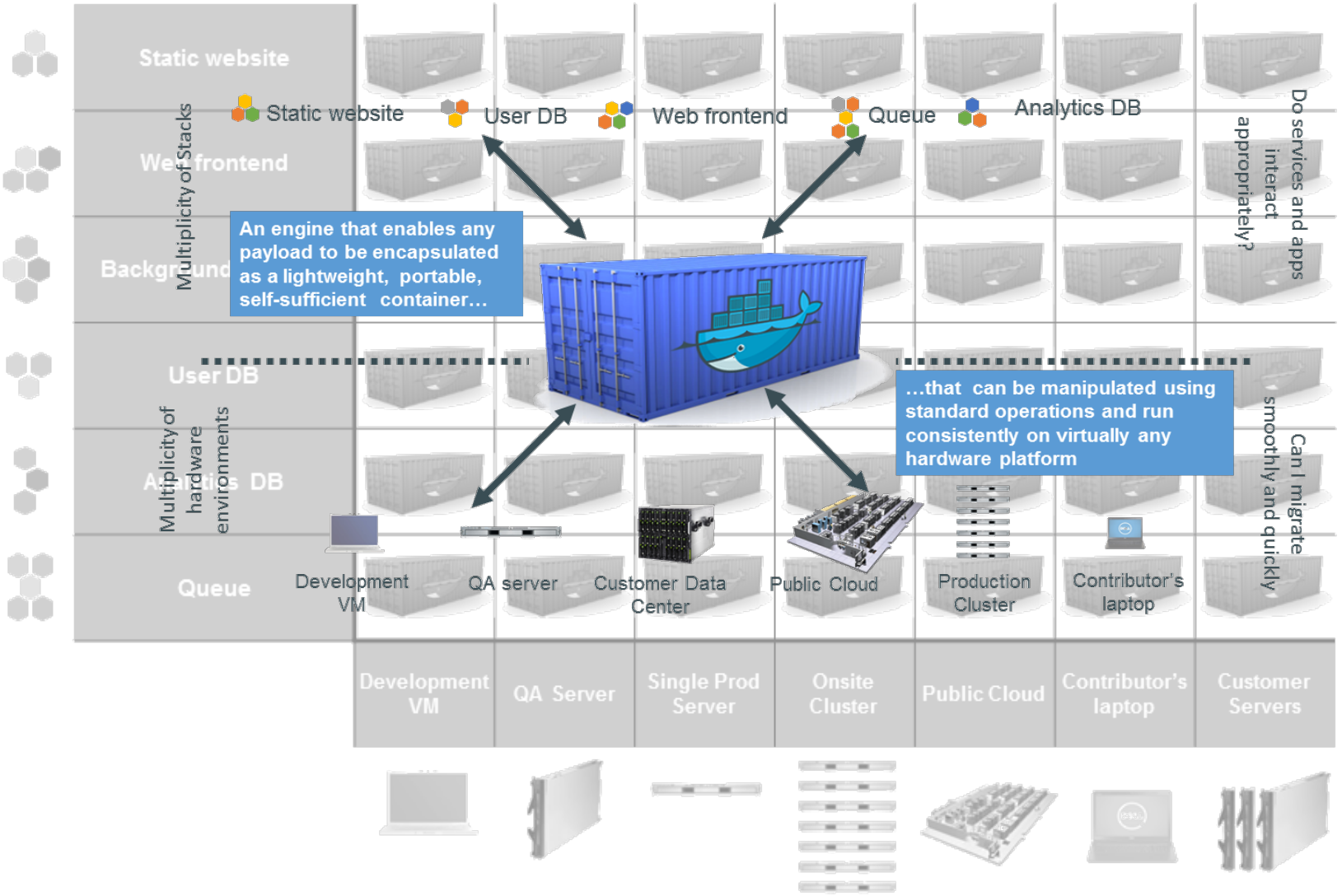


Hopla! Software

Apply What We Have Learned..

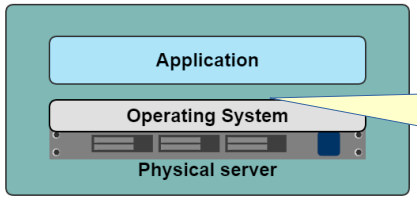


Problem Solved Again!!!

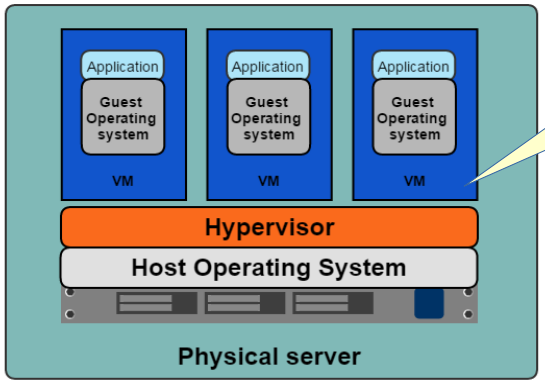


Hopla! Software

Baremetal vs Virtual vs Container



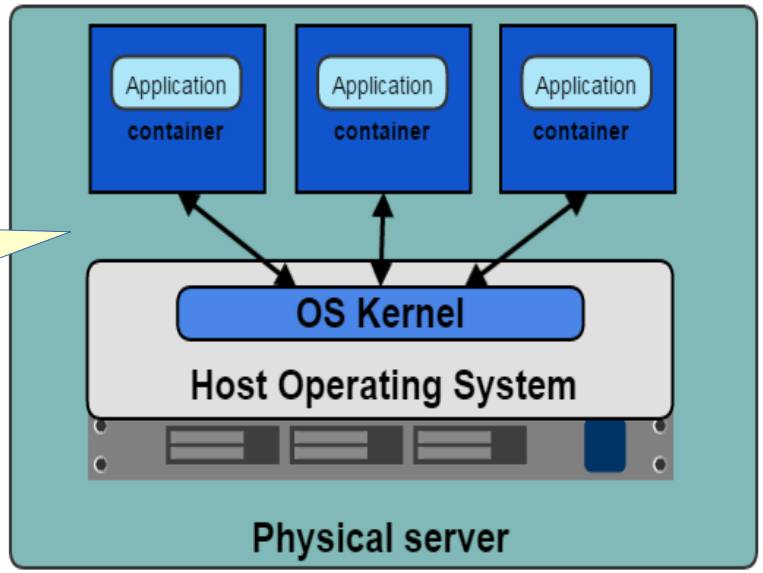
Manual Deployment: slow, complicated...
High costs
Wasted resources
Difficult to scale
Difficult to migrate
Vendor lock-in



Benefits:
- Central resource pooling
- Easy to scale
- Cloud: Elasticity & Pay-per-use

Limitations
- Full OS , besides CPU, RAM, storage
- Inefficiency due to OS assignment
- Application portability not guaranteed

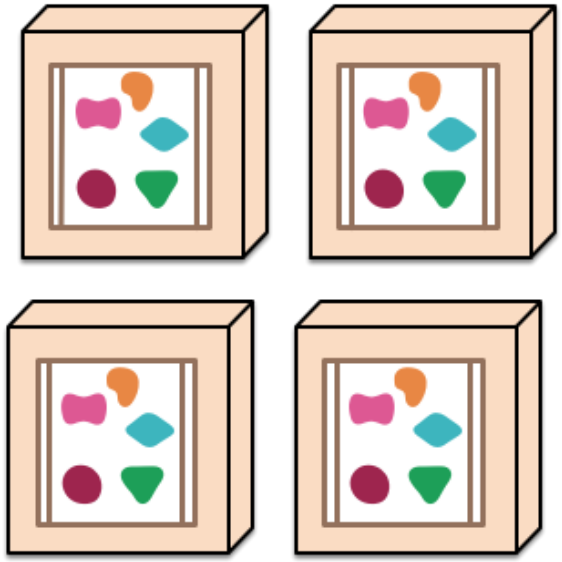
OS not needed
Less resources (RAM, CPU, storage)
Higher density (10's vs 100's)
Elasticity, flexibility, less provisioning time (0,5s)
Full portability



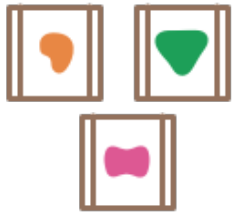
Microservices and SOA

A monolithic application puts all its functionality into a single process...

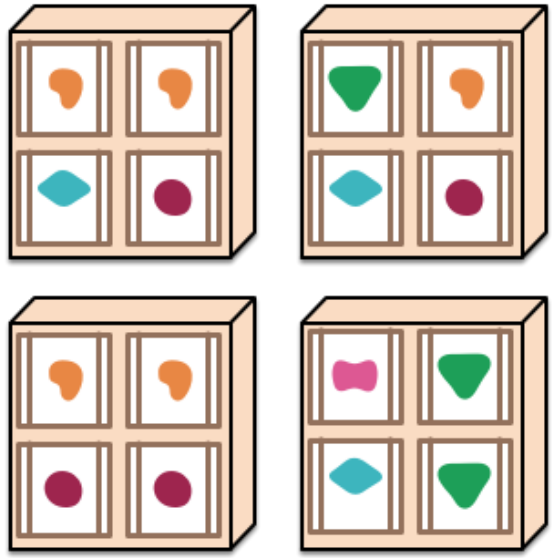
... and scales by replicating the monolith on multiple servers



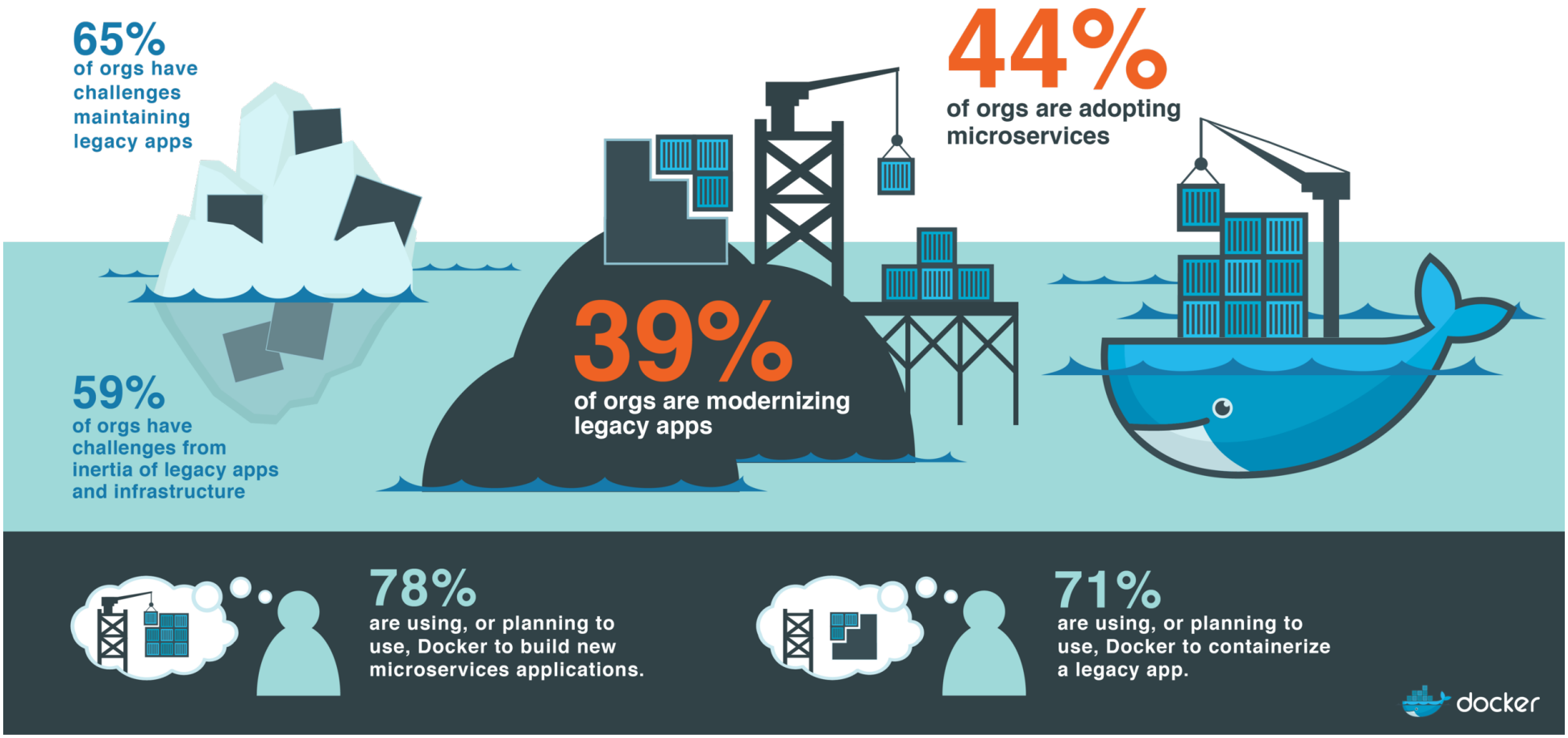
A microservices architecture puts each element of functionality into a separate service...



... and scales by distributing these services across servers, replicating as needed.



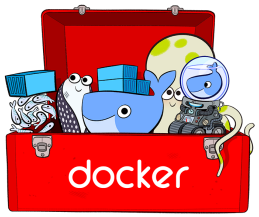
Moving Forward

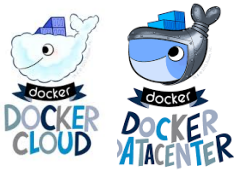


Why Docker?

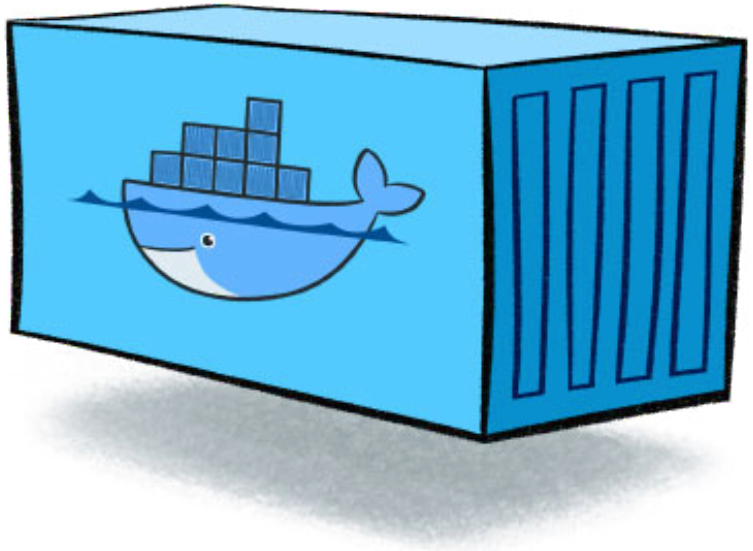
- OCI Standard
  - Format
  - Runtime

- ToolBox
  - Engine
  - Client
  - Enterprise Solutions



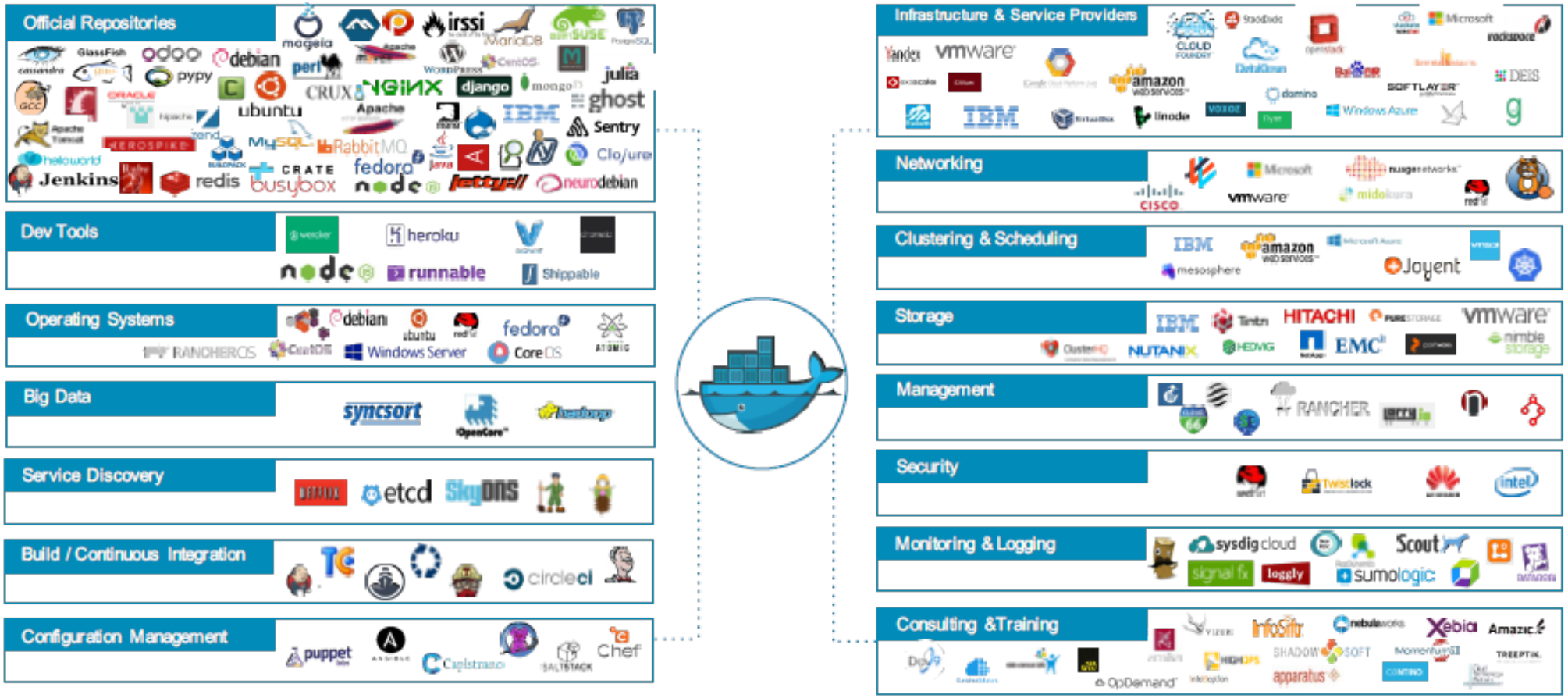


- Different Platforms



Build, Ship and Run

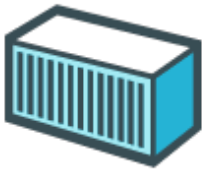
Why Docker?


Official Repositories
Dev Tools
heroku
runnable
Shippable
Operating Systems
Windows Server
CoreOS
Big Data
syncsort
Service Discovery
etcd
SkyDNS
Build / Continuous Integration
circleci
Configuration Management
puppet
Chef
Infrastructure & Service Providers
vmware
amazon web services
Windows Azure
linode
Networking
CISCO
vmware
Clustering & Scheduling
IBM
amazon web services
mesosphere
Joyent
Storage
IBM
HITACHI
vmware
NUTANIX
EMC
Management
RANCHER
Security
Monitoring & Logging
sysdig cloud
Scout
signal fx
loggly
sumologic
Consulting &Training
InfoSiftr
Xebia
Amazic
apparatus
OpDemand

Docker Define A Process

Build

Develop an app using Docker containers with any language and any toolchain.



Ship

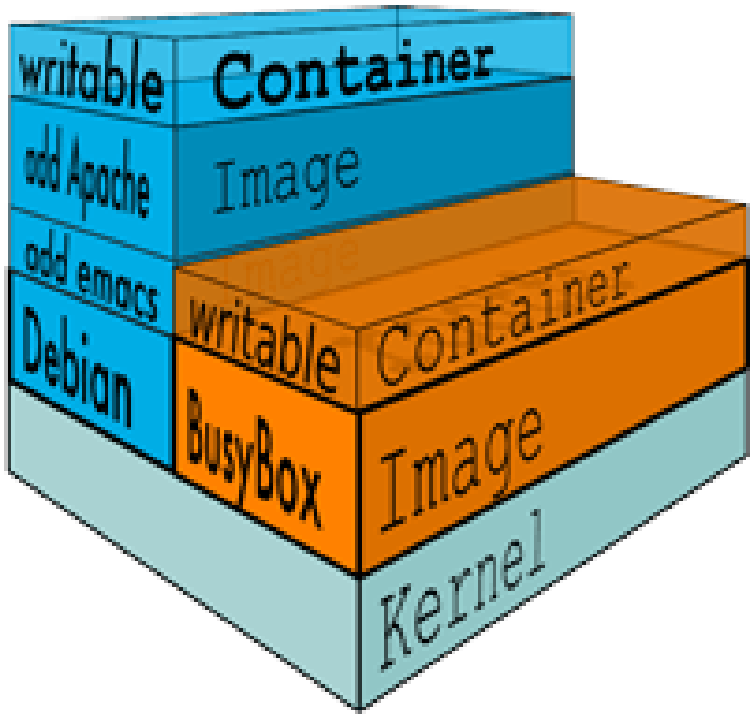
Ship the “Dockerized” app and dependencies anywhere - to QA, teammates, or the cloud - without breaking anything.

Run

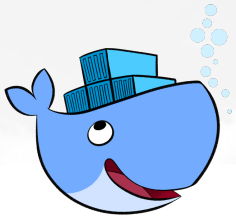
Scale to 1000s of nodes, move between data centers and clouds, update with zero downtime and more.

Concepts

- Image
  - Local
  - Remote → Registry
- Container
- Engine / Client
- Security
- Persistence vs Ephemeral

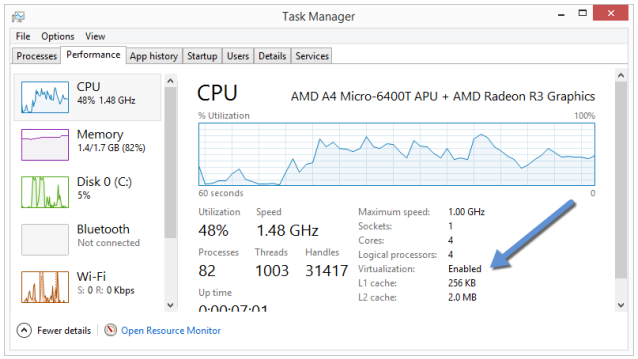


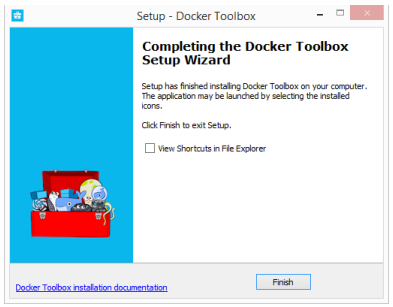
Docker Installation

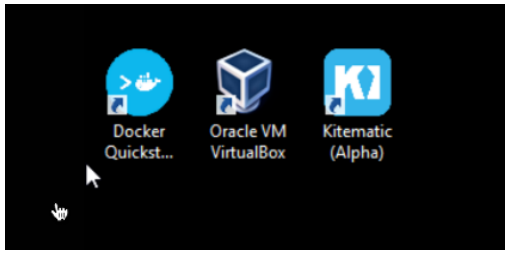


- Docker ToolBox on MacOS and Windows

https://docs.docker.com/windows/step_one/







- Docker on Linux

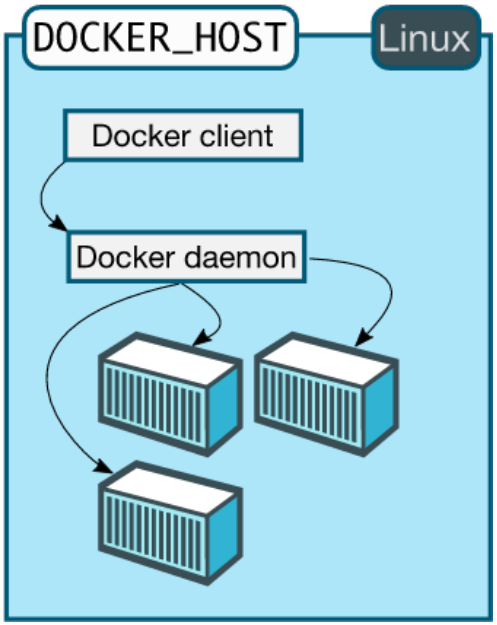
https://docs.docker.com/linux/step_one/

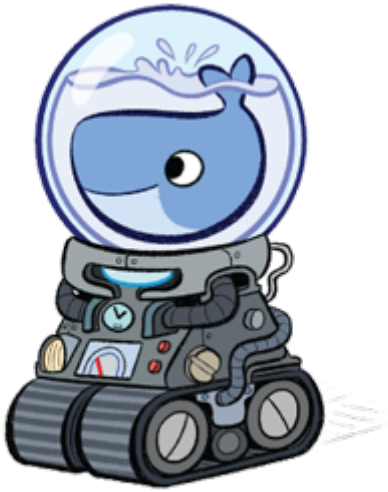
 # curl -fsSL https://get.docker.com/ | sh

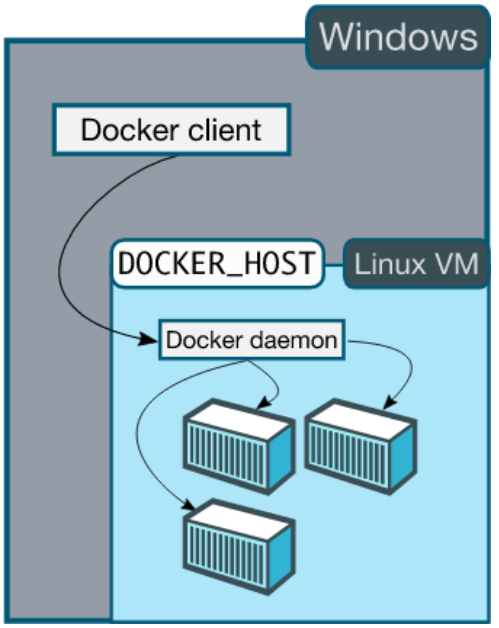
If computer is behind proxy/firewall dowload and add pgp key
 # curl -fsSL https://get.docker.com/gpg | sudo apt-key add -

Use root or add your user to docker group
usermod -aG docker <USER>

Docker on Native Linux vs Docker on Windows/MacOS








```
$ docker-machine ip default
$ docker-machine env default
$ eval $(docker-machine env default)
```

https://docs.docker.com/machine/install-machine/

Let’s Run Some Containers

$ docker run hello-world

$ docker run -ti busybox

$ docker run -tid busybox

$ docker ps

Container's Properties and Naming

Docker container name's could be specified for simpler access.

$ docker inspect <DOCKERID or DOCKERNAME>

$ docker ps -ql

$ docker run -tid --name "mybusybox" busybox

$ docker stop "mybusybox"

Managing Container Status

$ docker start "mybusybox"

$ docker stats "mybusybox"

Running vs Starting/Stopping

$ docker run -tid --name "mybusybox" busybox

$ docker stop "mybusybox"

$ docker rm "mybusybox"

Let’s create a simple service

$ docker run -d --name “mynginx” nginx

$ docker ps -ql

NGINX ports are ready, but not service ...

$ curl http://<localhost or virtualhost>:80

$ docker rm -f “mynginx”

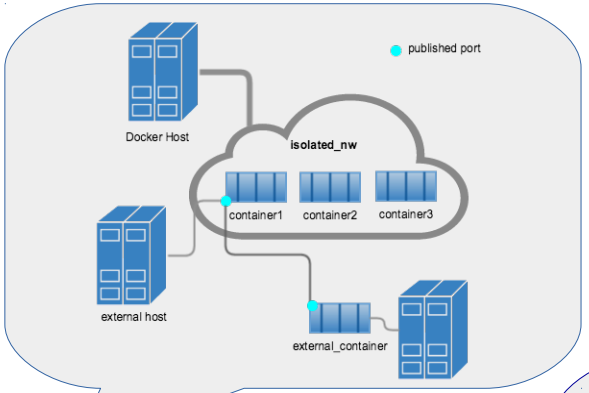
We can use -d HOST_PORT:CONTAINER_PORT or -P and docker will use a random port

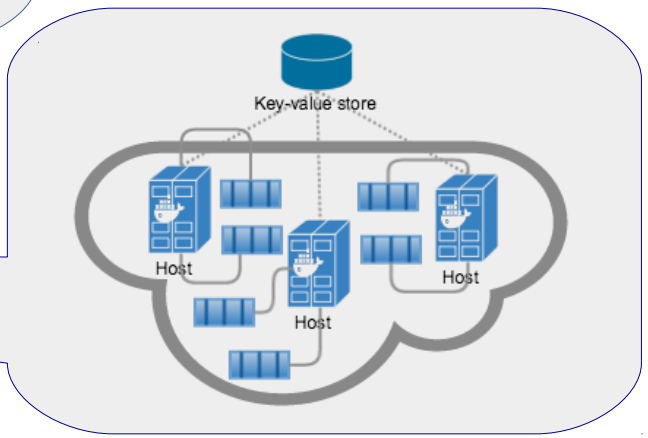
$ docker run -d -p 8080:80 --name “mynginx” nginx

$ curl http://<localhost or virtualhost>:8080

Networking in Docker

- Docker Standard Drivers:
  - None
  - Host
  - Container
  - Bridge (default bridge0)
  - Overlay



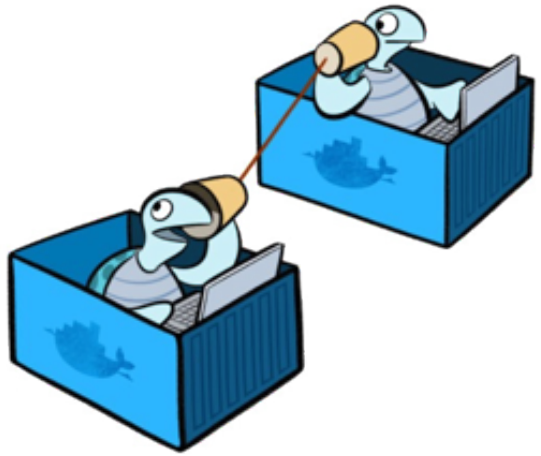


- Linking between containers
- Non Standard Drivers (IPVLAN/MACVLAN)

Linking Containers

- Simple default bridge linking
  - “mynginx” container is already running
  - Run a simple busybox linked to “mynginx” and test website

```
$ docker run --link=mynginx -ti --rm busybox

/ #  wget -q http://mynginx:80 -O -
```



Persistent Data on Docker Containers

- Docker containers data
  - Runtime Data
  - Configuration Data
  - Application Data
- Docker Volumes
  - Simple docker volume
  - Host’s filesystem directory
  - Special volumes provided by docker engine drivers

VOLUMES

A data volume is a specially-designated directory within one or more containers that bypasses the Union File System.

Docker Volumes and Host data

- Docker Volumes

  $ docker run -ti --rm --name one -v myvol:/myimportantdata busybox

  / # touch /myimportantdata/persistent-file

  / # touch /non-persistent-file

  $ docker run -ti --rm --name two -v myvol:/myimportantdata busybox

  / # ls /myimportantdata/persistent-file

  / # ls /non-persistent-file

- Host Filesystem

  $ docker run -ti --rm --name one -v /tmp:/myimportantdata busybox

  / # touch /myimportantdata/persistent-file

  / # touch /non-persistent-file

  $ docker run -ti --rm --name two -v /tmp:/myimportantdata busybox

  / # ls /myimportantdata/persistent-file

  / # ls /non-persistent-file

Building Images

- Creating Dockerfiles
  - Base image

    FROM BASEIMAGE

  - Install application binaries and libraries

    RUN yum install package

  - Add required files and code

    ENV APPLICATION_OR_RESOURCE_ENVIRONMENT
    COPY SOURCE DESTINE

  - Prepare Docker container start commands

    ENTRYPOINT [“/container_starting_script.sh”]
    CMD [“start_container”]

  - Expose service ports

    EXPOSE PORTS/PROTOCOL

Keep Images Small and Simple

Dockerfile

```
FROM debian:jessie

MAINTAINER NGINX Docker Maintainers "docker-maint@nginx.com"

ENV NGINX_VERSION 1.11.1-1~jessie

RUN apt-key adv --keyserver hkp://pgp.mit.edu:80 --recv-keys 573BFD6B3D8FBC641079A6ABABF5BD827BD9BF62 \
&& echo "deb http://nginx.org/packages/mainline/debian/ jessie nginx" >> /etc/apt/sources.list \
&& apt-get update \
&& apt-get install --no-install-recommends --no-install-suggests -y \
ca-certificates \
nginx=${NGINX_VERSION} \
nginx-module-xslt \
nginx-module-geoip \
nginx-module-image-filter \
nginx-module-perl \
nginx-module-njs \
gettext-base \
&& rm -rf /var/lib/apt/lists/*

# forward request and error logs to docker log collector
RUN ln -sf /dev/stdout /var/log/nginx/access.log \
&& ln -sf /dev/stderr /var/log/nginx/error.log

EXPOSE 80 443

CMD ["nginx", "-g", "daemon off;"]
```

Build a Simple Nodejs Application

$ git clone https://github.com/hopla-training/docker-simplest-nodeapp.git

$ docker build -t simpleapp:1.0 -f docker-simplest-nodeapp/Dockerfile docker-simplest-nodeapp

$ docker run -d -p 4000:3000 --name simpleapp simpleapp:1.0

$ curl http://0.0.0.0:4000

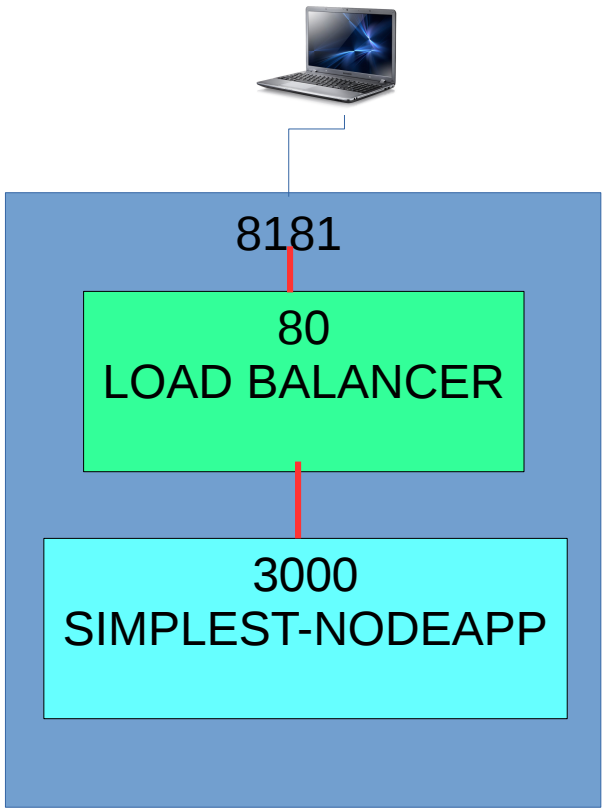
$ docker rm -f simpleapp

$ docker login

```
FROM alpine
RUN apk --update --no-progress add nodejs
ENV APPDIR /APP

WORKDIR ${APP}
COPY simpleapp.js simpleapp.js
COPY package.json package.json
RUN npm install
CMD ["node","simpleapp.js","3000"]
EXPOSE 3000
```

$ docker tag simpleapp:1.0 registry/username/repository:tag

Docker Compose



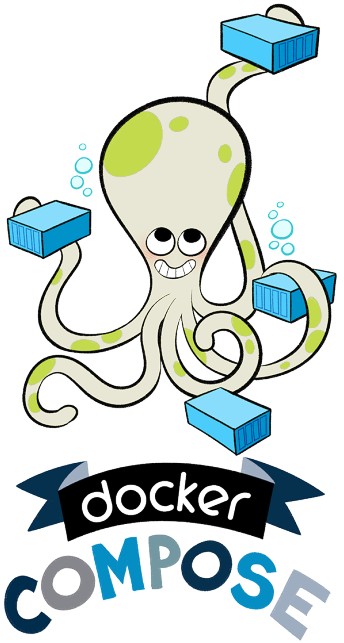

```
version: "2"
# A Docker Compose file for configuration of the development environment

services:
  # load balancer
  nginx-lb:
    image: hopla/alpine-docker-nginx-lb:demo
    container_name: nginx-lb
    restart: unless-stopped
    networks:
      - simplest-nodeapp-lb-demo-net
    ports:
      - "8181:80"

  simplest-nodeapp:
    image: hopla/alpine-docker-simplest-nodeapp:demo
    restart: unless-stopped
    networks:
       simplest-nodeapp-lb-demo-net:
        aliases:
          - simplest-nodeapp
    expose:
      - 3000
    depends_on:
      - nginx-lb

networks:
  simplest-nodeapp-lb-demo-net: {}
```

Docker Compose

- Docker Compose install

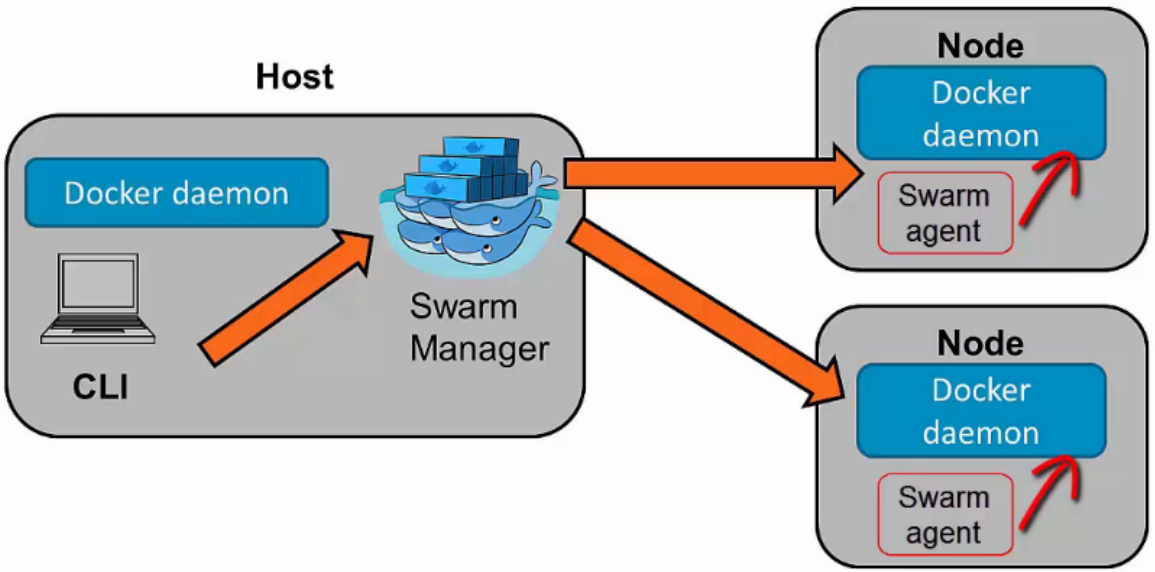
https://docs.docker.com/compose/install/

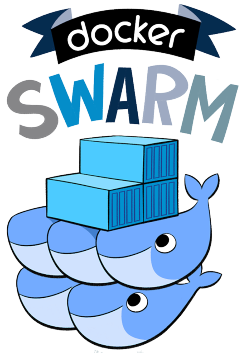
```
$ curl -L https://github.com/docker/compose/releases/download/1.7.1/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose

$ chmod 755  /usr/local/bin/docker-compose

$ docker-compose -f docker-compose up -d

$ docker-compose -f docker-compose ps

$ docker-compose -f docker-compose scale simplest-nodeapp=3
```



Docker Swarm

- Nodes in pool act as a single engine and can use standard client/API.
- Key-Value manager allow creation of multi-host-engine objects.





Working with Swarm

- Communication with swarm cluster should be encrypted using certificates (TLS environment variables should be applied).

  $ docker -H tcp://docker-swarm-manager:PORT info

We will use a Docker Swarm Environment created with Virtualbox/Vagrant. Swarm manager port is published on local port 8501 and communication isn't encrypted for these examples. Consul is configured as keyvalue storage.

$ docker -H tcp://*192.168.1.10*:8501 info

$ docker -H tcp://*192.168.1.10*:8501 ps

$ docker -H tcp://*192.168.1.10*:8501 run --name swarm-busybox -tid busybox

$ docker -H tcp://*192.168.1.10*:8501 ps

Compose Application on Swarm

We will use a Docker Swarm Environment created with Virtualbox/Vagrant. Swarm manager port is published on local port 8501 and communication isn't encrypted for these examples. Consul is configured as keyvalue storage.

$ docker-compose -f docker-simple-DEMO.yml -H tcp://*192.168.1.10*:8501 up -d

Creating network "practice_simplest-nodeapp-lb-demo-net" with the default driver
Pulling nginx-lb (hopla/alpine-docker-nginx-lb:demo)...
node3: Pulling hopla/alpine-docker-nginx-lb:demo... : downloaded
manager: Pulling hopla/alpine-docker-nginx-lb:demo... : downloaded
node2: Pulling hopla/alpine-docker-nginx-lb:demo... : downloaded
Pulling simplest-nodeapp (hopla/alpine-docker-simplest-nodeapp:demo)...
node3: Pulling hopla/alpine-docker-simplest-nodeapp:demo... : downloaded
manager: Pulling hopla/alpine-docker-simplest-nodeapp:demo... : downloaded
node2: Pulling hopla/alpine-docker-simplest-nodeapp:demo... : downloaded
Creating nginx-lb
Creating practice_simplest-nodeapp_1

Compose Application on Swarm

We will use a Docker Swarm Environment created with Virtualbox/Vagrant. Swarm manager port is published on local port 8501 and communication isn't encrypted for these examples. Consul is configured as keyvalue storage.

```
$ docker-compose -f docker-simple-DEMO.yml -H tcp://192.168.1.10:8501 ps
          Name                        Command           State           Ports
-------------------------------------------------------------------------------------
nginx-lb                    nginx -g daemon off;     Up      10.0.200.12:8181->80/tcp
practice_simplest-nodeapp_1   node simpleapp.js 3000   Up      3000/tcp
```

Compose Application on Swarm

We will use a Docker Swarm Environment created with Virtualbox/Vagrant. Swarm manager port is published on local port 8501 and communication isn't encrypted for these examples. Consul is configured as keyvalue storage.

$ docker-compose -f docker-simple-DEMO.yml -H tcp://*192.168.1.10*:8501 scale simplest-nodeapp=5

Creating and starting practice_simplest-nodeapp_2 ... done
Creating and starting practice_simplest-nodeapp_3 ... done
Creating and starting practice_simplest-nodeapp_4 ... done
Creating and starting practice_simplest-nodeapp_5 ... done

$ docker-compose -f docker-simple-DEMO.yml -H tcp://192.168.1.10:8501 ps

```
Name                    Command          State          Ports
-------------------------------------------------------------------------------
nginx-lb                nginx -g daemon off;     Up      10.0.200.12:8181->80/tcp
practice_simplest-nodeapp_1   node simpleapp.js 3000   Up      3000/tcp
practice_simplest-nodeapp_2   node simpleapp.js 3000   Up      3000/tcp
practice_simplest-nodeapp_3   node simpleapp.js 3000   Up      3000/tcp
practice_simplest-nodeapp_4   node simpleapp.js 3000   Up      3000/tcp
practice_simplest-nodeapp_5   node simpleapp.js 3000   Up      3000/tcp
```

Compose Application on Swarm

We will use a Docker Swarm Environment created with Virtualbox/Vagrant. Swarm manager port is published on local port 8501 and communication isn't encrypted for these examples. Consul is configured as keyvalue storage.

$ docker-compose -f docker-simple-DEMO.yml -H tcp://*192.168.1.10*:8501 scale simplest-nodeapp=5

Creating and starting practice_simplest-nodeapp_2 ... done
Creating and starting practice_simplest-nodeapp_3 ... done
Creating and starting practice_simplest-nodeapp_4 ... done
Creating and starting practice_simplest-nodeapp_5 ... done

$ docker-compose -f docker-simple-DEMO.yml -H tcp://192.168.1.10:8501 ps

```
Name                    Command          State          Ports
-------------------------------------------------------------------------------
nginx-lb                nginx -g daemon off;     Up      10.0.200.12:8181->80/tcp
practice_simplest-nodeapp_1   node simpleapp.js 3000   Up      3000/tcp
practice_simplest-nodeapp_2   node simpleapp.js 3000   Up      3000/tcp
practice_simplest-nodeapp_3   node simpleapp.js 3000   Up      3000/tcp
practice_simplest-nodeapp_4   node simpleapp.js 3000   Up      3000/tcp
practice_simplest-nodeapp_5   node simpleapp.js 3000   Up      3000/tcp
```

Compose Application on Swarm

We will use a Docker Swarm Environment created with Virtualbox/Vagrant. Swarm manager port is published on local port 8501 and communication isn't encrypted for these examples. Consul is configured as keyvalue storage.

```
$ docker -H tcp://192.168.1.10:8501 ps
CONTAINER ID        IMAGE                                  COMMAND                  CREATED             STATUS              PORTS               NAMES
057c2f73fcf6        hopla/alpine-docker-simplest-nodeapp:demo   "node simpleapp.js 30"   3 minutes ago       Up 3 minutes        3000/tcp            manager/practice_simplest-nodeapp_2
a006138910f7        hopla/alpine-docker-simplest-nodeapp:demo   "node simpleapp.js 30"   3 minutes ago       Up 3 minutes        3000/tcp            node2/practice_simplest-nodeapp_5
008db2c28d06        hopla/alpine-docker-simplest-nodeapp:demo   "node simpleapp.js 30"   3 minutes ago       Up 3 minutes        3000/tcp            node3/practice_simplest-nodeapp_3
7cefd3ea3f92        hopla/alpine-docker-simplest-nodeapp:demo   "node simpleapp.js 30"   3 minutes ago       Up 3 minutes        3000/tcp            node3/practice_simplest-nodeapp_4
4e865de93f68        hopla/alpine-docker-simplest-nodeapp:demo   "node simpleapp.js 30"   11 minutes ago      Up 11 minutes       3000/tcp            node3/practice_simplest-nodeapp_1
d7d95aad2a53        hopla/alpine-docker-nginx-lb:demo           "nginx -g 'daemon off'"  11 minutes ago      Up 11 minutes       10.0.200.12:8181->80/tcp   node2/nginx-lb
```

Compose Application on Swarm

We will use a Docker Swarm Environment created with Virtualbox/Vagrant. Swarm manager port is published on local port 8501 and communication isn't encrypted for these examples. Consul is configured as keyvalue storage.

$ docker -H tcp://192.168.1.10:8501 network ls

NETWORK ID	NAME	DRIVER
f0d064543170	manager/bridge	bridge
1387948ad38d	manager/docker_gwbridge	bridge
e0ca47aabc12	manager/host	host
438e45797455	manager/none	null
8a9a2ab3b900	node2/bridge	bridge
21920f38392c	node2/docker_gwbridge	bridge
af8749ae262c	node2/host	host
d0d754a8a074	node2/none	null
775c0a3e45b6	node3/bridge	bridge
737a57b33bfd	node3/docker_gwbridge	bridge
7b8eb51e5a71	node3/host	host
2e10abf83c8a	node3/none	null
809b7f1c7a4e	**practice_simplest-nodeapp-lb-demo-net**	**overlay**

Thanks....