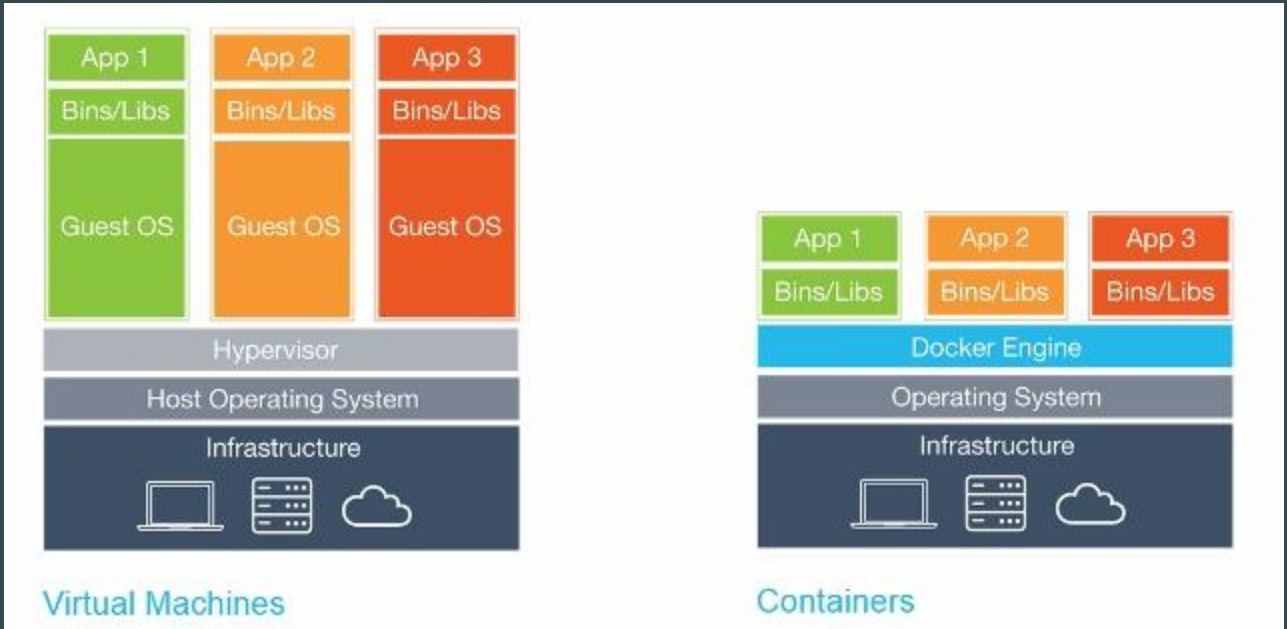
Contenerizando que es gerundio

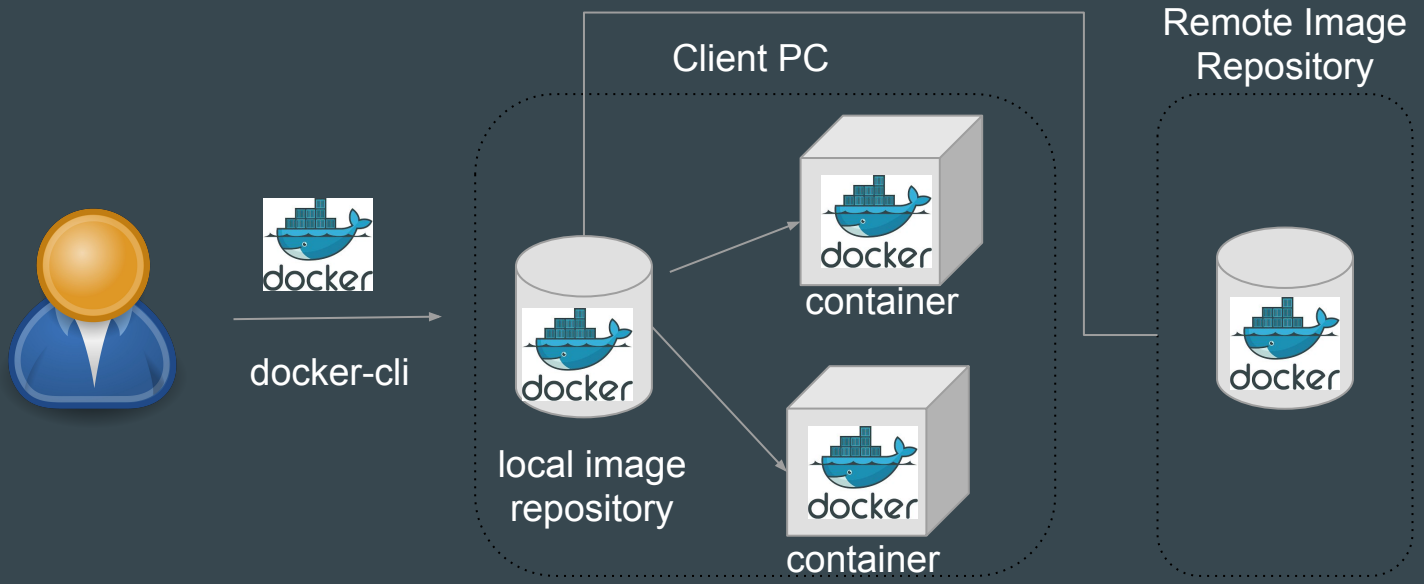
...

101

Containers


App 1
Bins/Libs
Guest OS
App 2
Bins/Libs
Guest OS
App 3
Bins/Libs
Guest OS
Hypervisor
Host Operating System
Infrastructure
Virtual Machines
App 1
Bins/Libs
App 2
Bins/Libs
App 3
Bins/Libs
Docker Engine
Operating System
Infrastructure
Containers

Docker In a Nutshell



Why Docker

- Agilidad
- Portabilidad
- En mi local funciona y en pro tb
- Comunidad

Comandos Docker

docker run -> Permite arrancar imágenes desde un repositorio remoto o local

docker ps -> Permite ver los contenedores en ejecución. Con -a vemos los terminados

docker exec -> Permite ejecutar un comando en un contenedor

docker restart -> Permite el reinicio de un contenedor

docker stop/start -> Permite parar/arrancar un contenedor

docker rm -> Permite eliminar un contenedor

docker inspect -> Permite ver la configuración del contenedor

docker logs -> Permite ver la salida estándar de la ejecución del contenedor

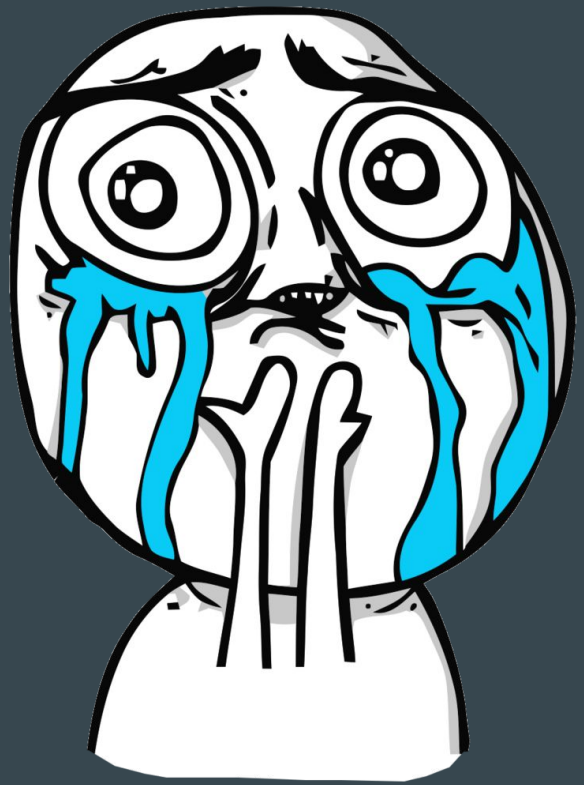
Arrancando contenedores

Docker run :

- -d -> Modo demonio
- -it -> Modo interactivo
- -p puertolocal:puertocontenedor -> Binding de puertos
- -e -> Para introducir variables de entorno en el contenedor
- --name -> Indicamos un nombre al contenedor
- --restart -> Permite indicar una política de cuándo se debe rearrancar el contenedor
- -m -c -> Limitar los recursos hadware a usar por el contenedor
- Muchos más: Seguridad, redes...

My First Container

1. docker pull busybox
2. docker run busybox -it



Un paso más

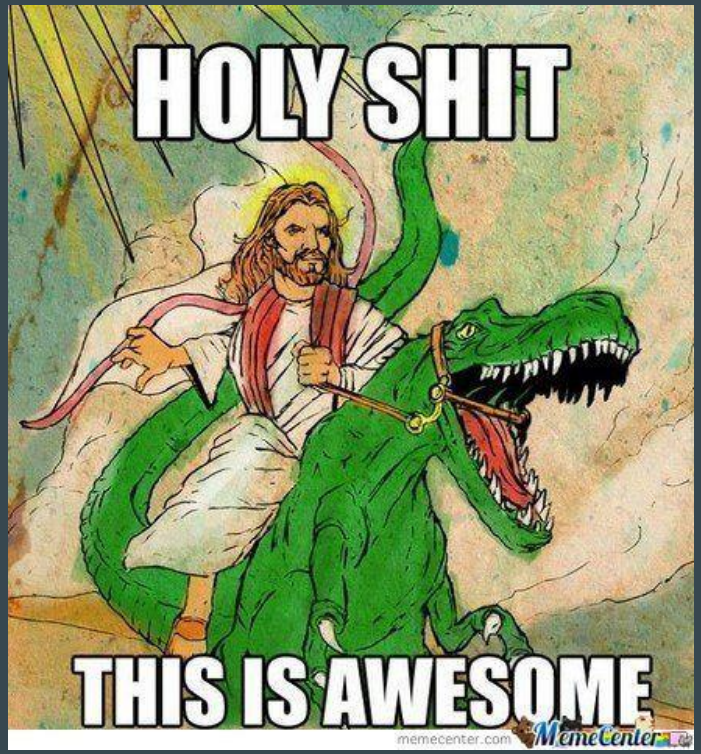
1. docker run -d -p 8080:80 --name mi-apache httpd:alpine
2. docker inspect mi-apache
3. docker exec -it mi-apache /bin/sh

Persistencia

- docker run -d -p 8080:80 -v mi_directorio:/usr/local/apache2/htdocs --name mi-apache-volumen httpd:alpine
- vi mi_directorio/index.html
- curl localhost:8080

Imágenes

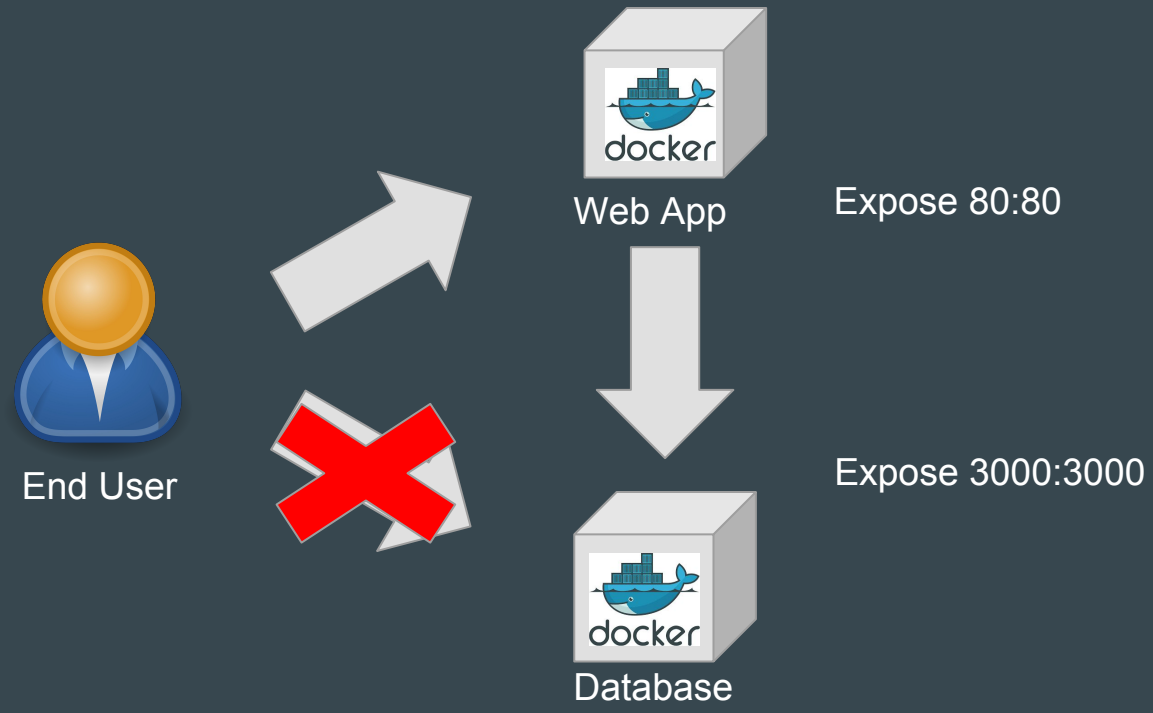
```
FROM alpine:latest

RUN mkdir /tmp/webApps \

&& apk add --update python

ADD ./index.html /tmp/webApps

VOLUME /tmp/webApps

WORKDIR /tmp/webApps

EXPOSE 8000

CMD [“python”,”-m”,”SimpleHTTPServer”,”8000”]
```



Registry


docker
Docker File
docker push
docker
REGISTRY
docker pull
docker
Local image repository

Comunicación entre contenedores



Comunicación entre contenedores

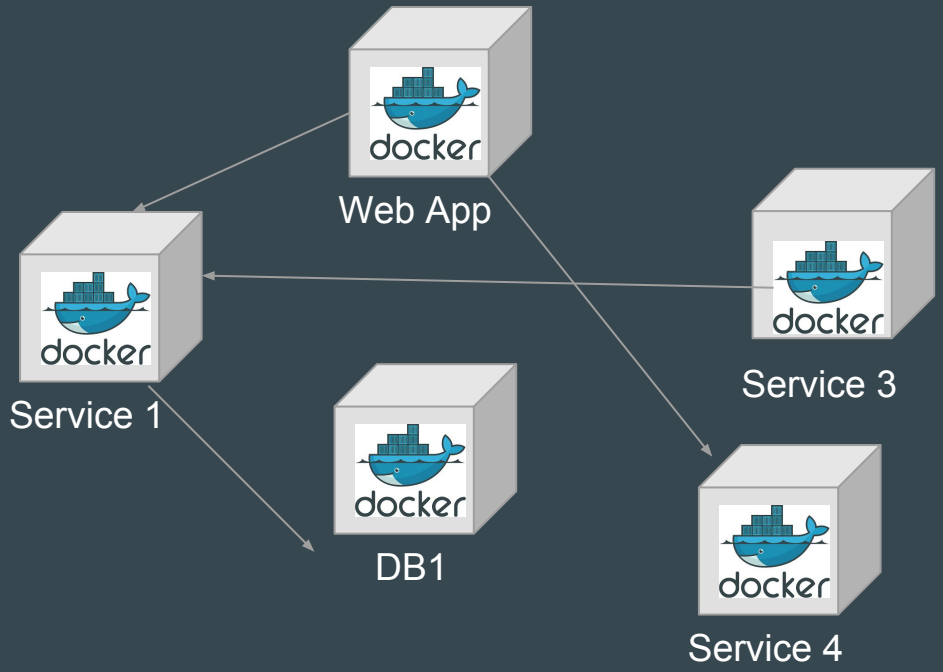
- docker run --name some-mysql -d -e MYSQL_ROOT_PASSWORD=123456789 mysql
- docker run --name some-wordpress --link some-mysql:mysql -p 8080:80 -d wordpress

Docker Machine

- docker-machine create --driver virtualbox maquina1
- docker-machine ls
- docker-machine ssh maquina1
- docker-machine rm maquina1

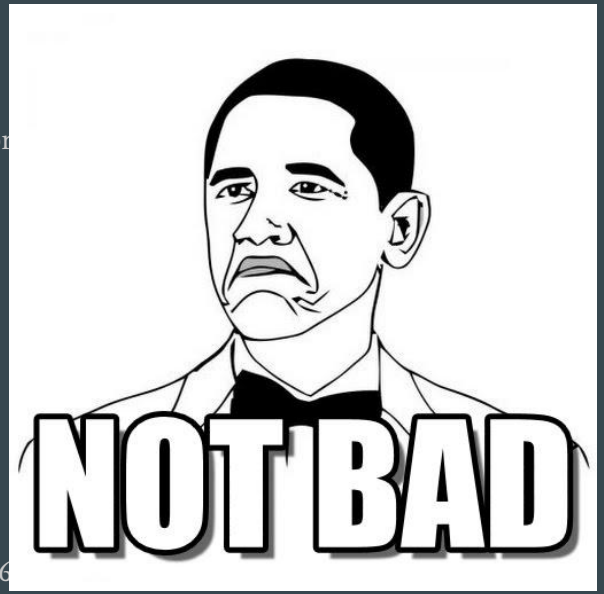
Docker En Producción

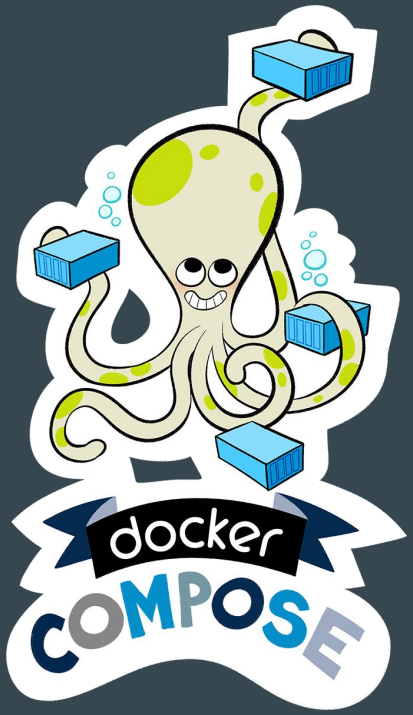
Orquestación



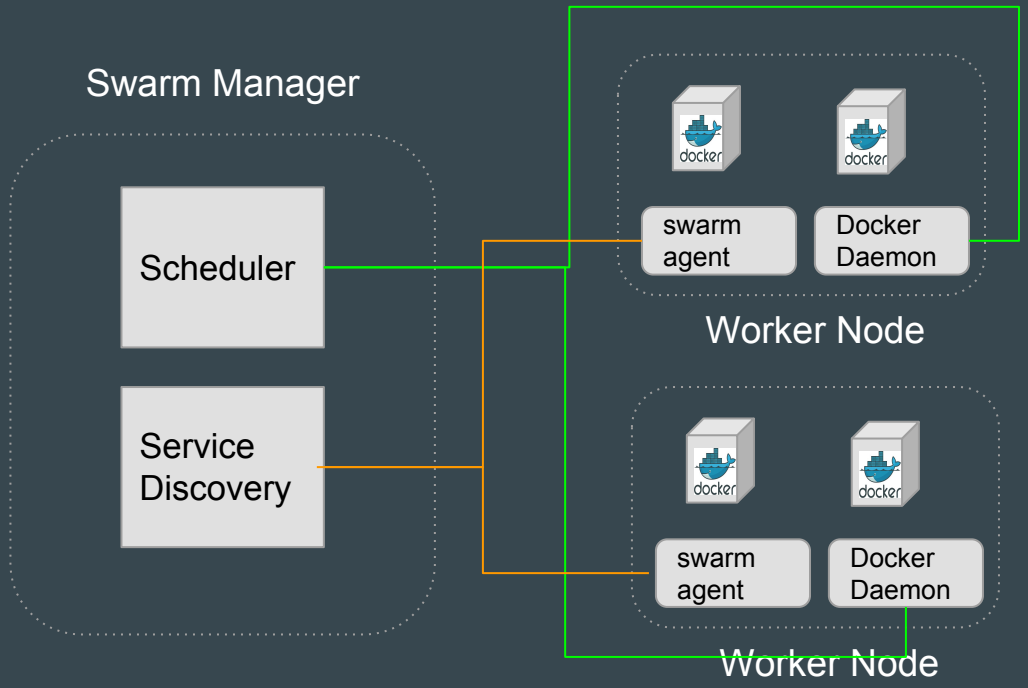
Docker Compose

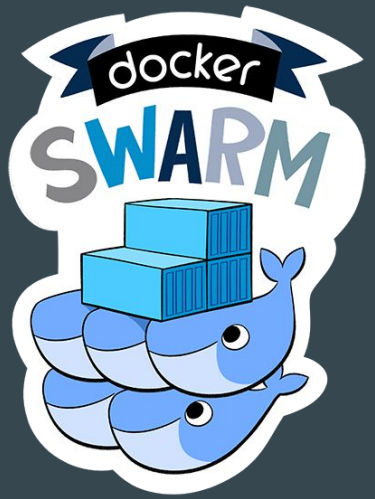
```
version: '2'
services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: wor
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress
  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_PASSWORD: wordpress
volumes:
    db_data:
```





Docker Swarm





Docker Swarm Concepts

Manager -> Nodo que maneja el envío de tareas a los Nodos Workers

Worker -> Nodo que ejecuta las tareas enviadas desde el manager

Scheduler -> Elemento que elige en qué nodo se ejecutarán las tareas

Service Discovery -> Elemento que maneja los cambios en los servicios

Service -> Definición de la imagen a ejecutar en contenedor con propiedades de réplica

Task -> Tarea enviada por el manager para la ejecución del contenedor en el nodo

Load Balancing -> Balanceo de carga entre las réplicas de los servicios

Docker Swarm

docker-machine create --driver virtualbox manager

docker-machine create --driver virtualbox node

docker-machine ssh manager

docker swarm init --advertise-addr <ip creada en el docker-machine>

docker-machine ssh node

docker join ...

docker-machine ssh manager

docker nodes ls

Docker Swarm

docker-machine ssh manager

docker service create --replicas 1 --name test apline ping google.es

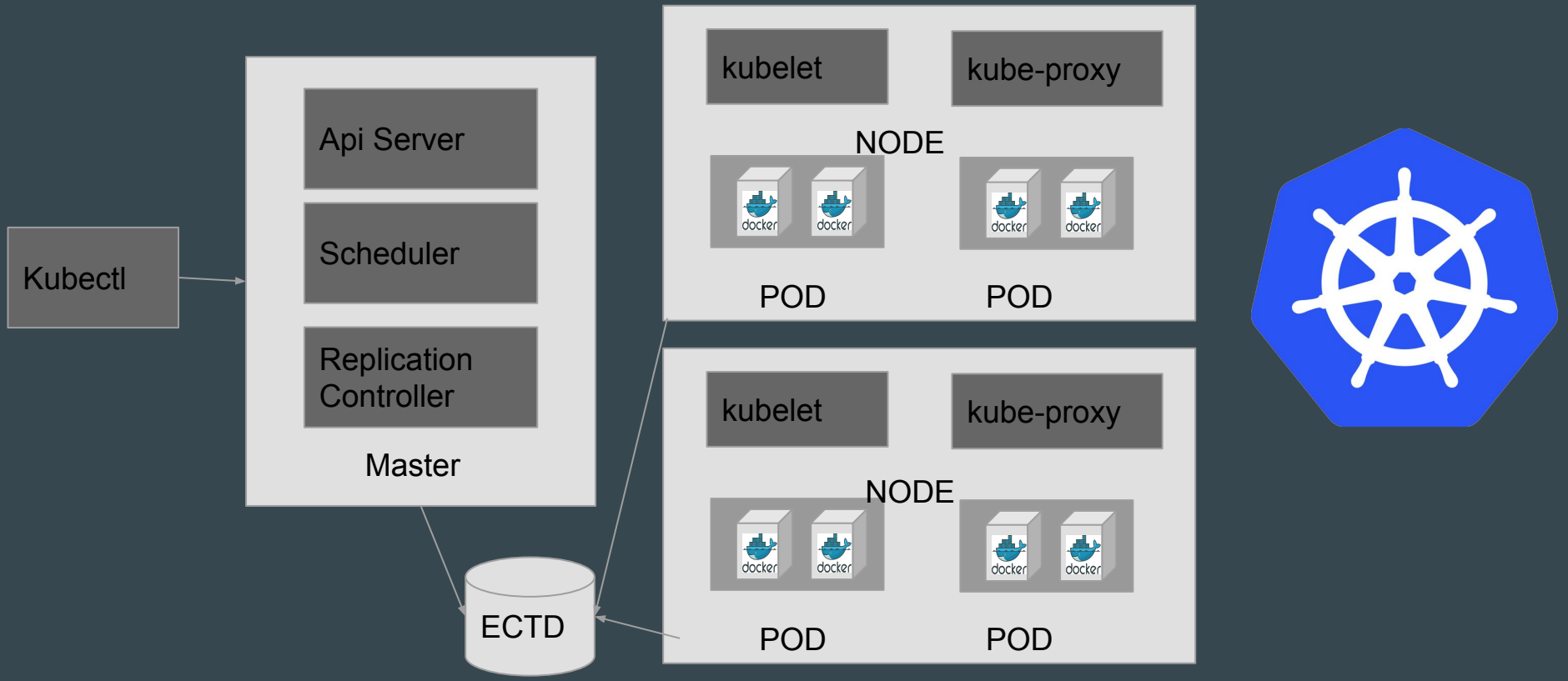
docker service ls

docker service scale test=2

docker service ls

docker service update --image alpine2 -> Roll Out

Kubernetes


Kubectl
Api Server
Scheduler
Replication Controller
Master
ECTD
kubelet
kube-proxy
NODE
docker
docker
docker
docker
POD
POD
kubelet
kube-proxy
NODE
docker
docker
docker
docker
POD
POD

Kubernetes Concepts

- kubectl -> cliente línea de comandos para manejar el cluster de kubernetes
- Node -> Máquina física o virtual donde se despliegan los pods
- Pods -> Conjunto de contenedores y volúmenes
- kubelet -> Proceso que se comunica con el nodo master
- kube-proxy -> Balanceador para los servicios de expuestos para los pods
- services-> Conjunto de pods expuestos con un endpoint
- Replication Controller -> Controla el número de réplicas de un pod

Kubernetes

minikube start

kubectl create -f
https://raw.githubusercontent.com/kubernetes/kubernetes/master/examples/mysql-wordpress-pd/local-volumes.yaml
kubectl create secret generic mysql-pass --from-file=password.txt
kubectl create -f
https://raw.githubusercontent.com/kubernetes/kubernetes/master/examples/mysql-wordpress-pd/mysql-deployment.yaml
kubectl create -f
https://raw.githubusercontent.com/kubernetes/kubernetes/master/examples/mysql-wordpress-pd/wordpress-deployment.yaml

Kubernetes

Pros:

- Madurez
- Multicontenedor
- Comunidad
- Declartivo

Contras:

- Multicontenedor

Swarm

Pros:

- Docker Nativo
- Comunidad

Contra:

- Solo uso de la api de docker
- Imperativo