DOCKER concepts

docker

28/04/2017

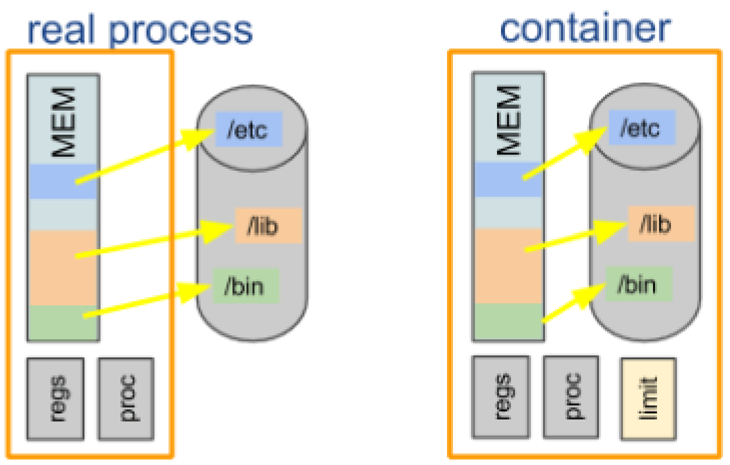
Solocal GROUP

Sommaire

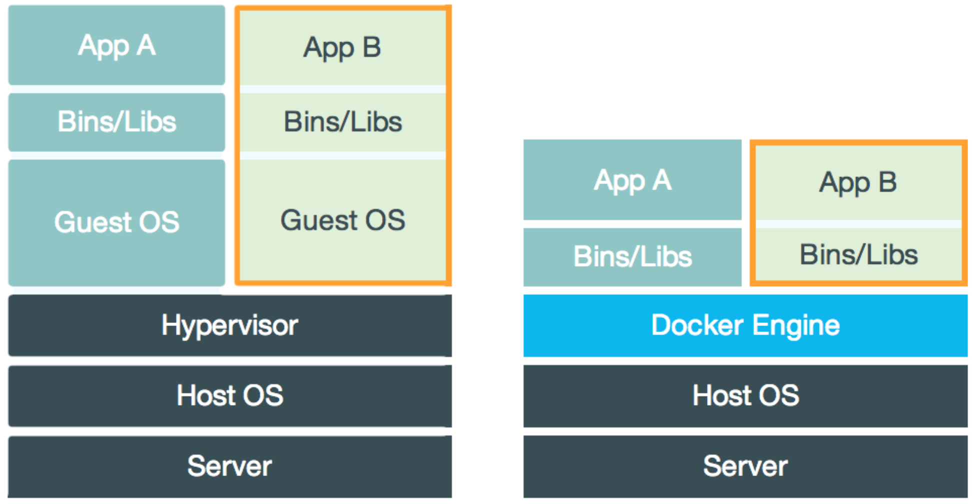
Introduction

Container based virtualization

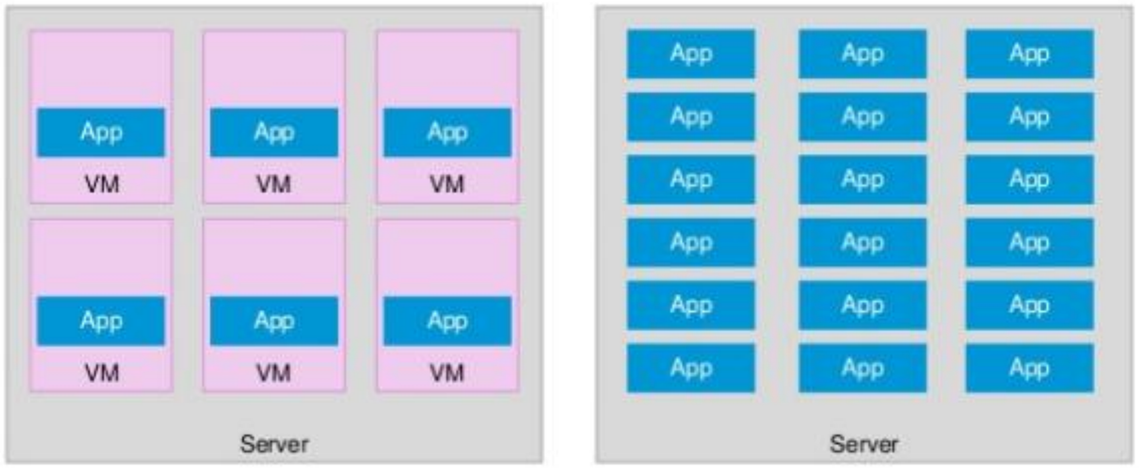
- Isolated systems
- Containers share the same OS kernel
- Containers hold the components necessary to run the desired software



Container based virtualization



Container based virtualization



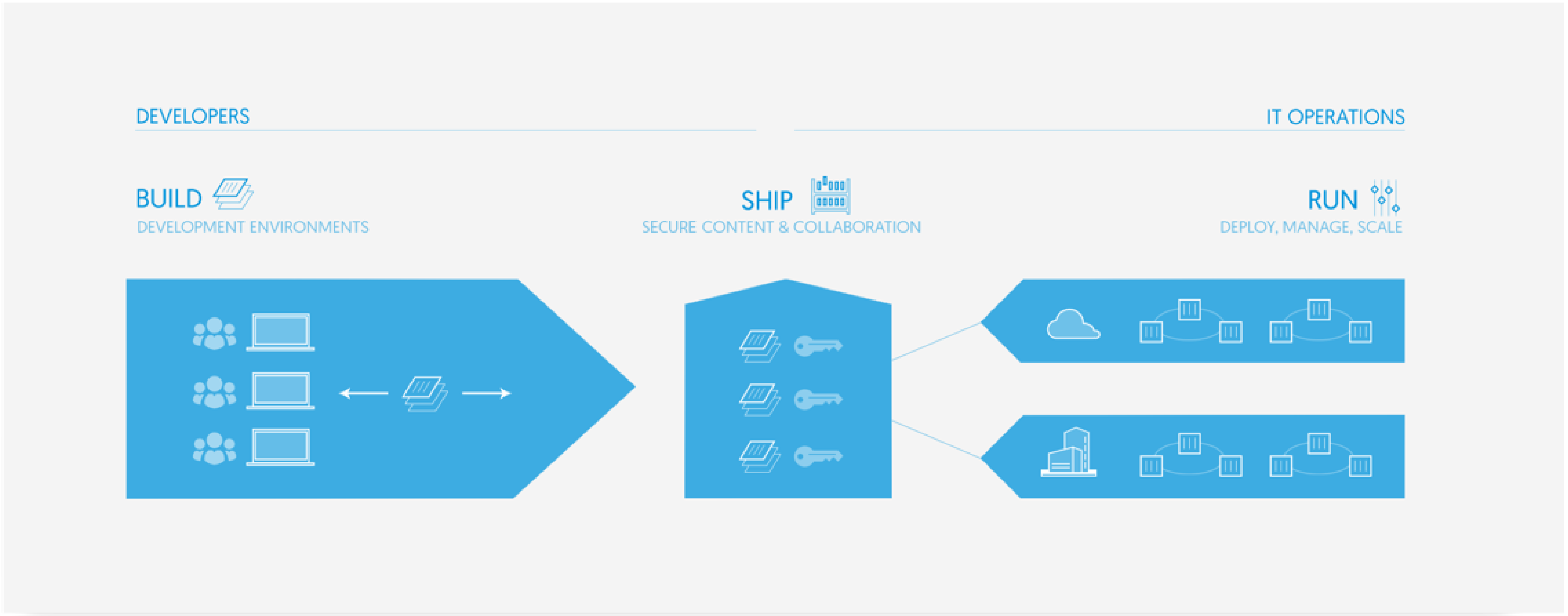
Pets vs Cattle

Pets vs Cattle

Docker Container

- ❑ Docker
  - ❑ API
- ❑ Docker container
  - ❑ cgroups and namespaces
  - ❑ Union file systems

Build Ship Run



Advantages of running docker

- Rapid application deployment (infra as code) + testing
- Portability across machines (hardware agnostic)
- Version control and component reuse (tag , layers)
- Sharing (registry)
- Lightweight footprint and minimal overhead
- Simplified maintenance (immutable infrastructure)

Drawbacks

- Be aware of security
- Don't use it for monolitic application
- You should respect some best practices
  - (https://github.com/docker/labs/tree/master/12factor)

- 1 - Codebase
- 2 - Dependencies
- 3 - Configuration
- 4 - External services
- 5 - Build / Release / Run
- 6 - Processes
- 7 - Port binding
- 8 - Concurrency
- 9 - Disposability
- 10 - Dev / Prod parity
- 11 - Logs
- 12 - Admin processes

Installing Docker

Installing Docker

```
apt-get update
apt-get install -y apt-transport-https ca-certificates linux-image-extra-$(uname -r) linux-image-extra-virtual
apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADBF76221572C52609D
echo "deb https://apt.dockerproject.org/repo ubuntu-xenial main" | sudo tee /etc/apt/sources.list.d/docker.list
apt-get update
apt-get install -y docker-engine
service docker start
usermod -aG docker ubuntu
```

Introduction to images

Differences bw images and containers :

- Image
  - inert, immutable, file
  - snapshot of a container
- Container
  - running instance of that image

Where to find docker images ?

- Docker Hub https://hub.docker.com/
  - Public Registry
  - Share images with co-workers
  - Manage images with tags
  - Use official Images

- Build your own and store it in a **private registry**

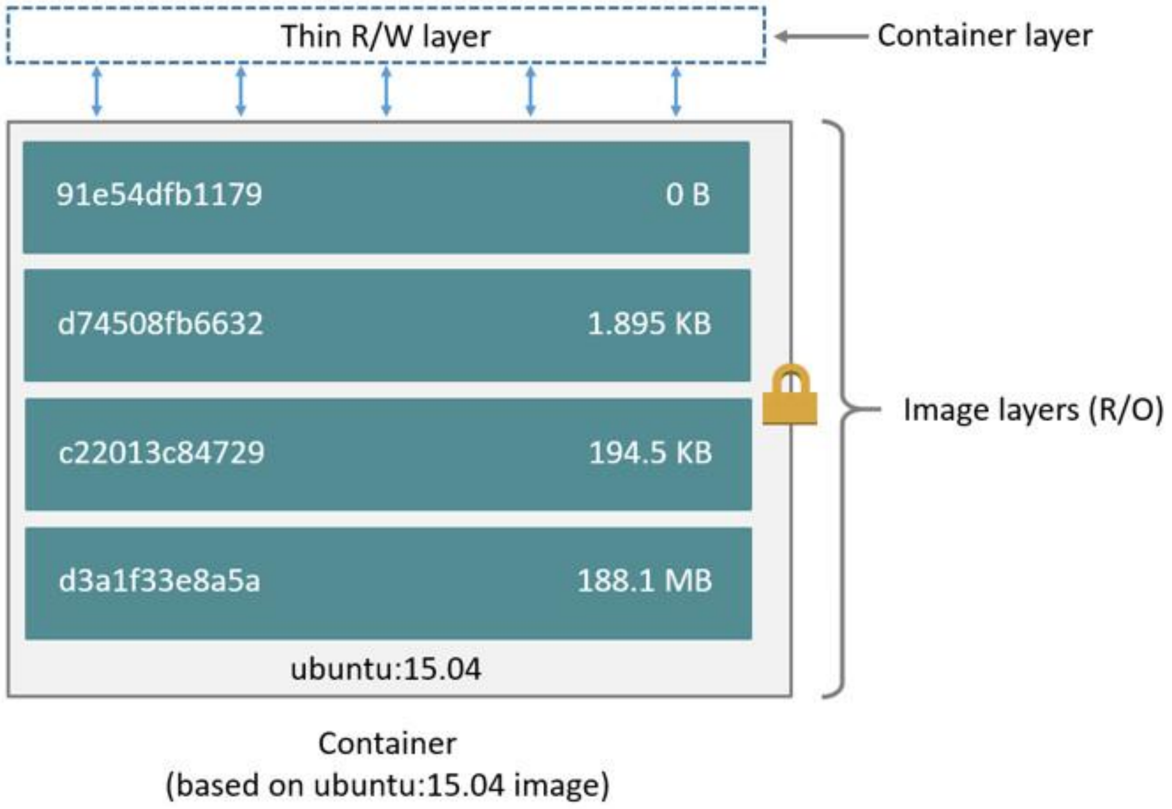
Managing containers

Intro to docker cli

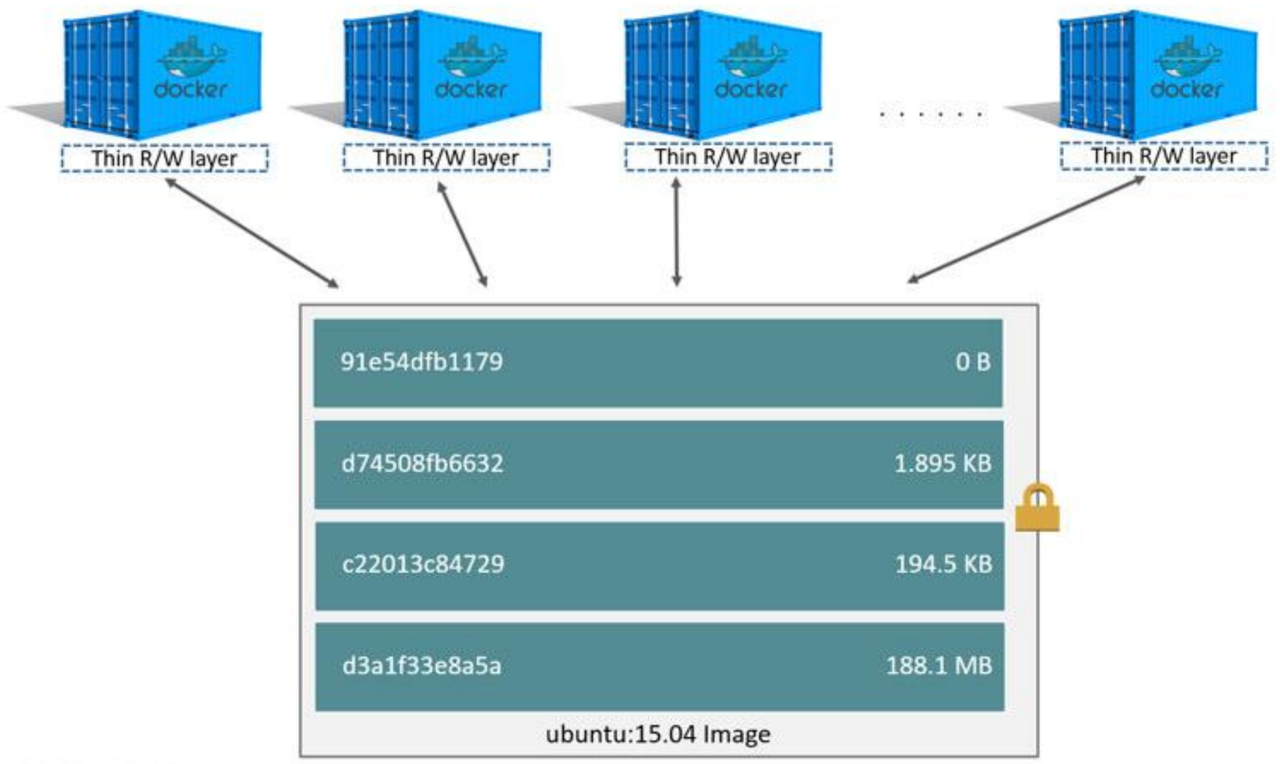
```
docker --help
docker run hello-world
docker --version
docker pull
docker run
docker exec
docker inspect
docker rm
docker ps
docker stats
docker logs
...
```

Building images

Images and layers



Images and layers



Images management

```
docker import / export
docker tag
docker push / pull
docker rmi
```

Dockerfile

- Docker can build images automatically by reading the instructions from a **Dockerfile**

```
docker build -t shykes/myapp  .
```

- **Instructions :**
  - FROM
  - RUN
  - ADD
  - ENV
  - EXPOSE
  - LABEL
  - USER
  - WORKDIR
  - VOLUME
  - …

Volumes

Data volumes

- Data persistence
- Share data between containers
- Plugins (host directory, shared storage,   …)

```
docker run -d -p 80:80 -v /home/ubuntu/localdir:/usr/share/nginx/html/ --name nginxsth nginxsth:latest
```

Networking

Port mapping

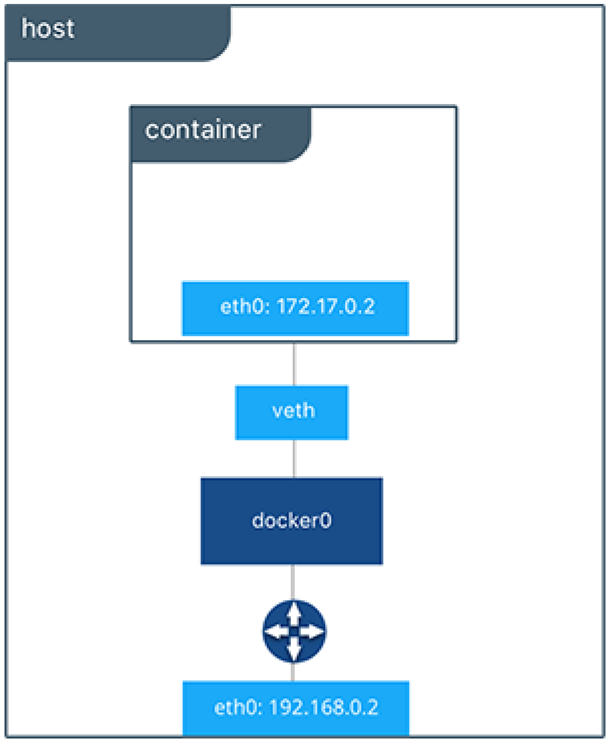
- Default:
  - Containers can make connections to the outside world
  - Outside world cannot connect to containers

- Mapping:
  - To accept incoming connections :
    - specify option -P or -p IP:host_port:container_port in 'run' command
  - iptables -t nat -L -n

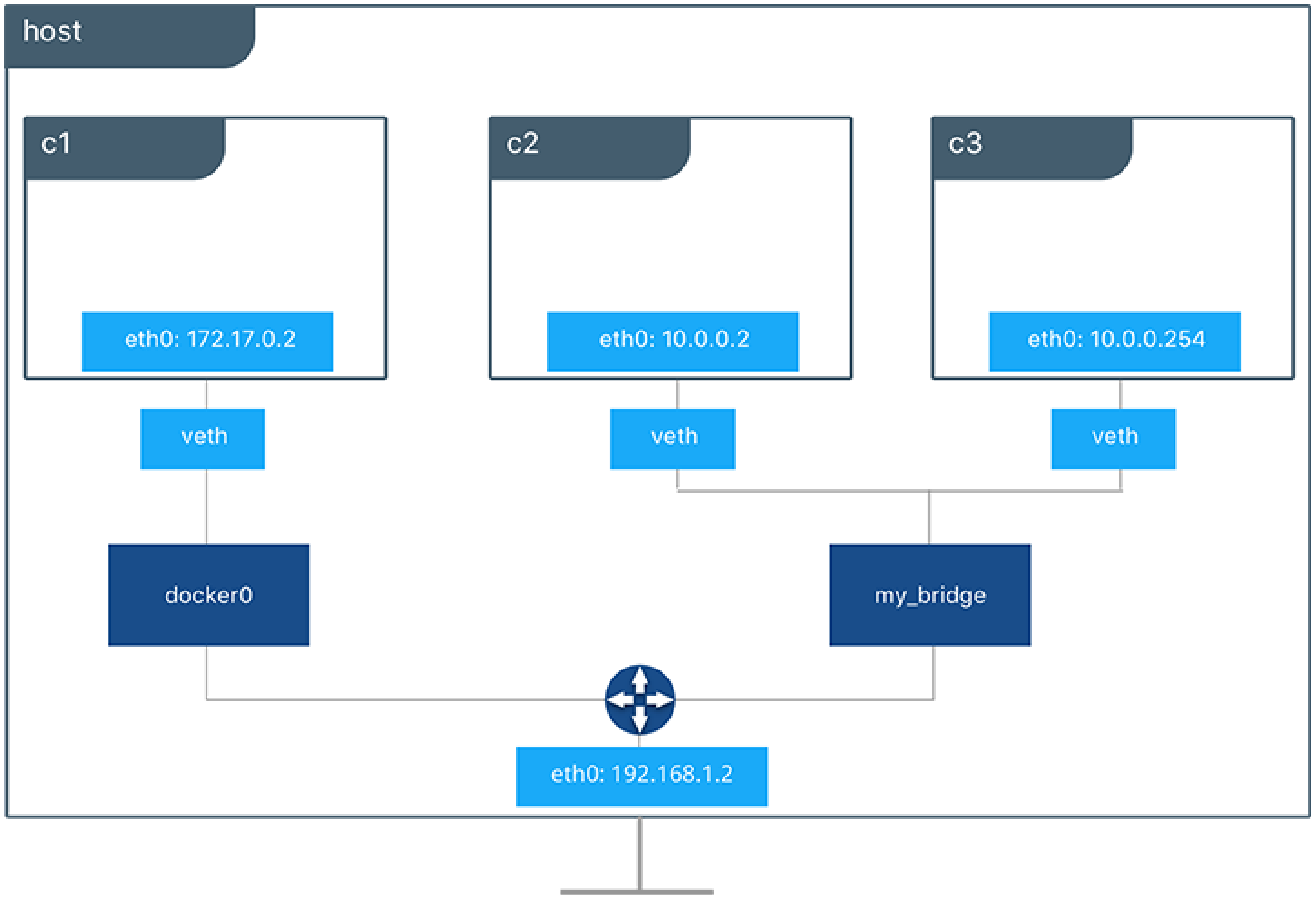
DNS

- ❑ Embedded DNS
  - ❑ Embedded DNS to provide service discovery for containers (127.0.0.11:53)
  - ❑ Key/value store in Docker Engine
  - ❑ Network-scoped (Containers not on the same network cannot resolve each other's addresses)

Bridging



Bridging

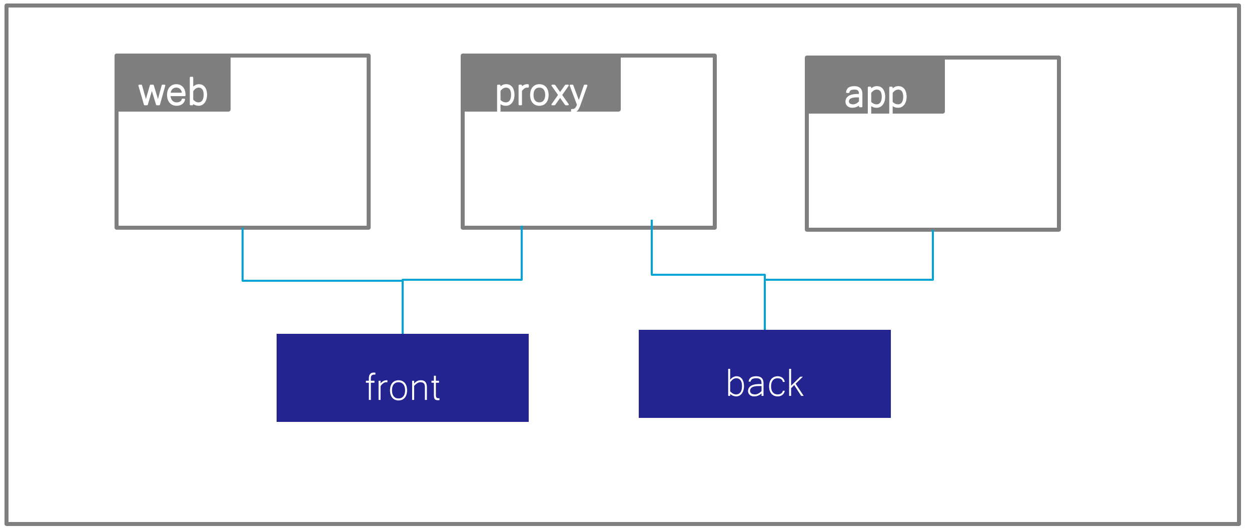

host
c1
eth0: 172.17.0.2
veth
docker0
c2
eth0: 10.0.0.2
veth
c3
eth0: 10.0.0.254
veth
my_bridge
eth0: 192.168.1.2

Network demo

```
docker network create --driver bridge isolated_nw
docker run -d -P --network=isolated_nw nginxsth:latest
docker network inspect isolated_nw
docker exec -ti fb82aad80f58 bash
ping 172.18.0.2 => ok
ping reverent_ramanujan => KO
docker run -d -P --network=isolated_nw --name c1 nginxsth:latest
docker run -d -P --network=isolated_nw --name c2 nginxsth:latest
docker exec -ti c1 bash
ping c2 => ok
```

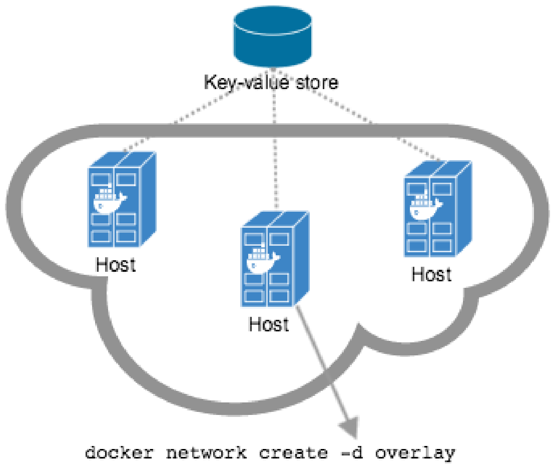
Network demo

```
docker network create --driver bridge front
docker network create --driver bridge back
docker run -d --net=front --name web nginxsth
docker run -d --net=front --name proxy nginxsth
docker network connect back proxy
docker run -d --net=back --name app nginxsth
```



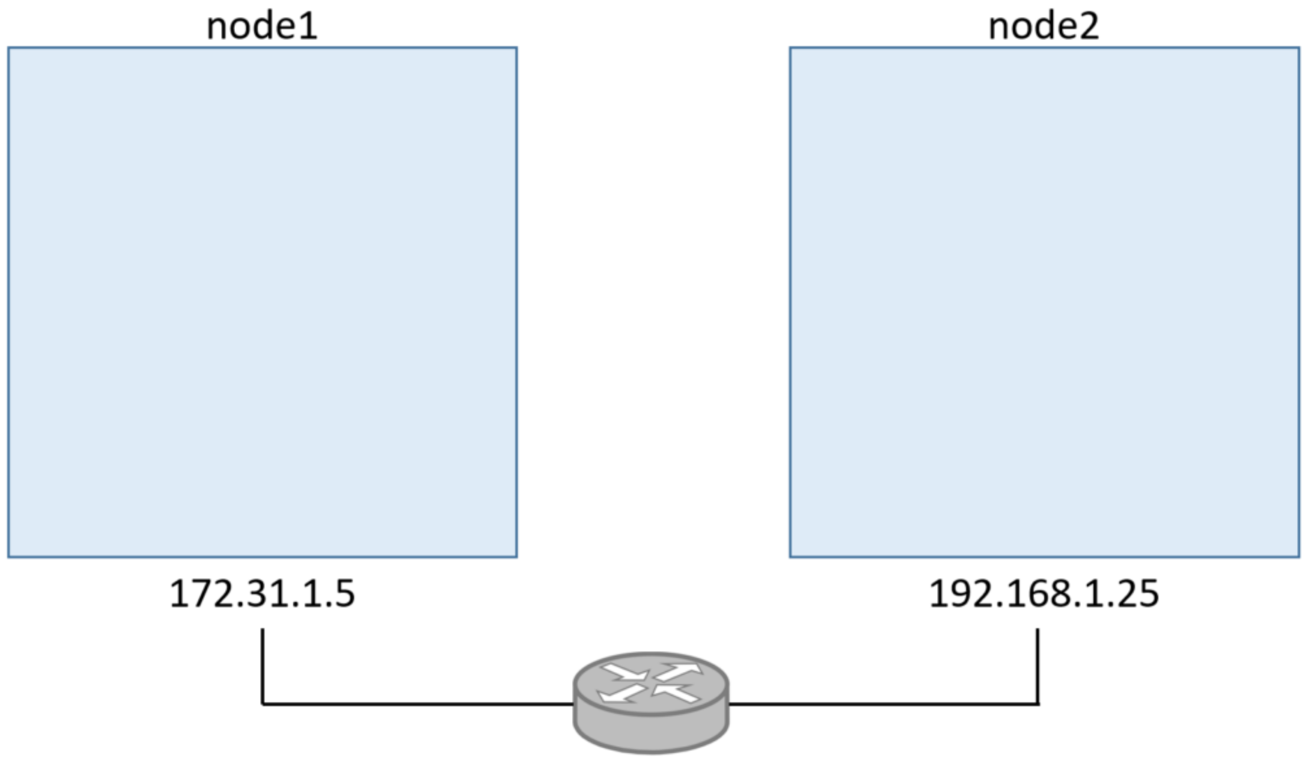
Multihost networking

- Overlay outside swarm mode
  - Requires a valid key-value store service
  - Consul, Etcd, and ZooKeeper
  - Configure docker engines to use the key-value store

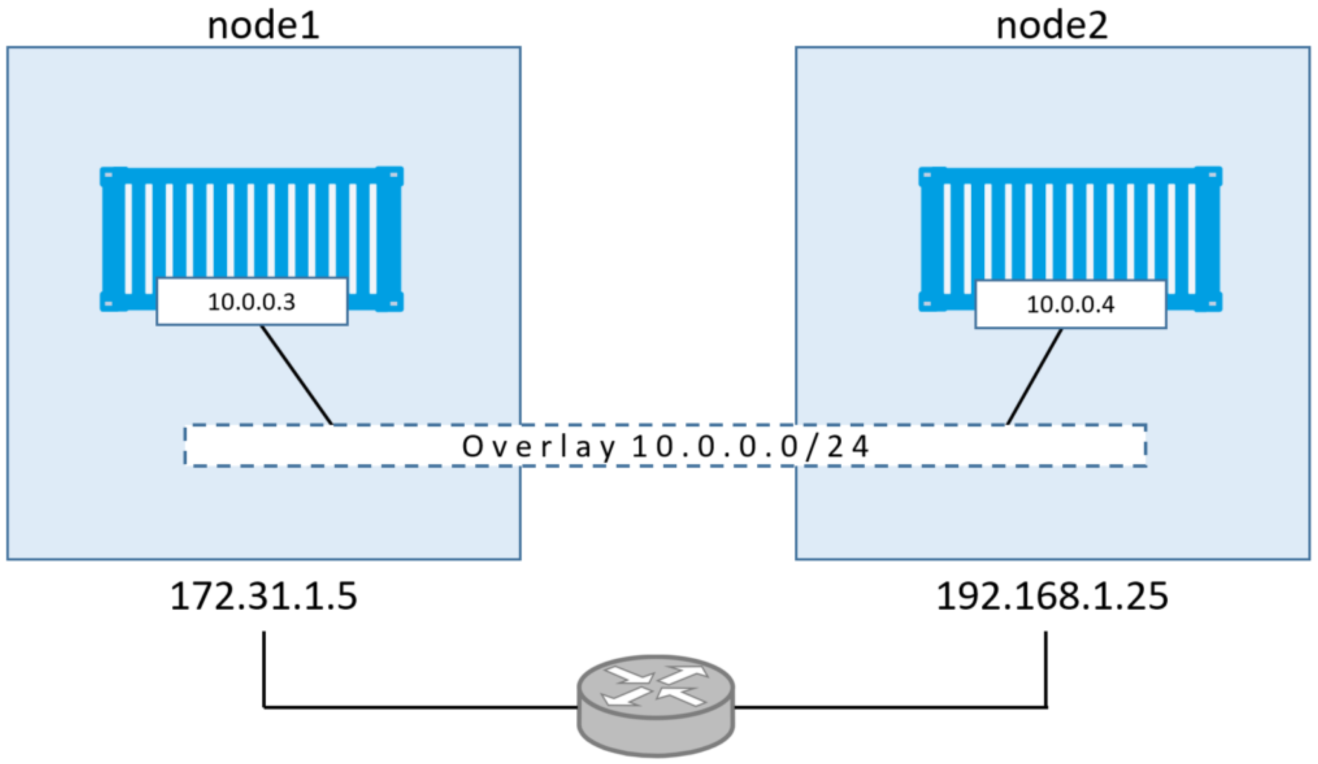


- Overlay in swarm mode
  - Embedded key-value store

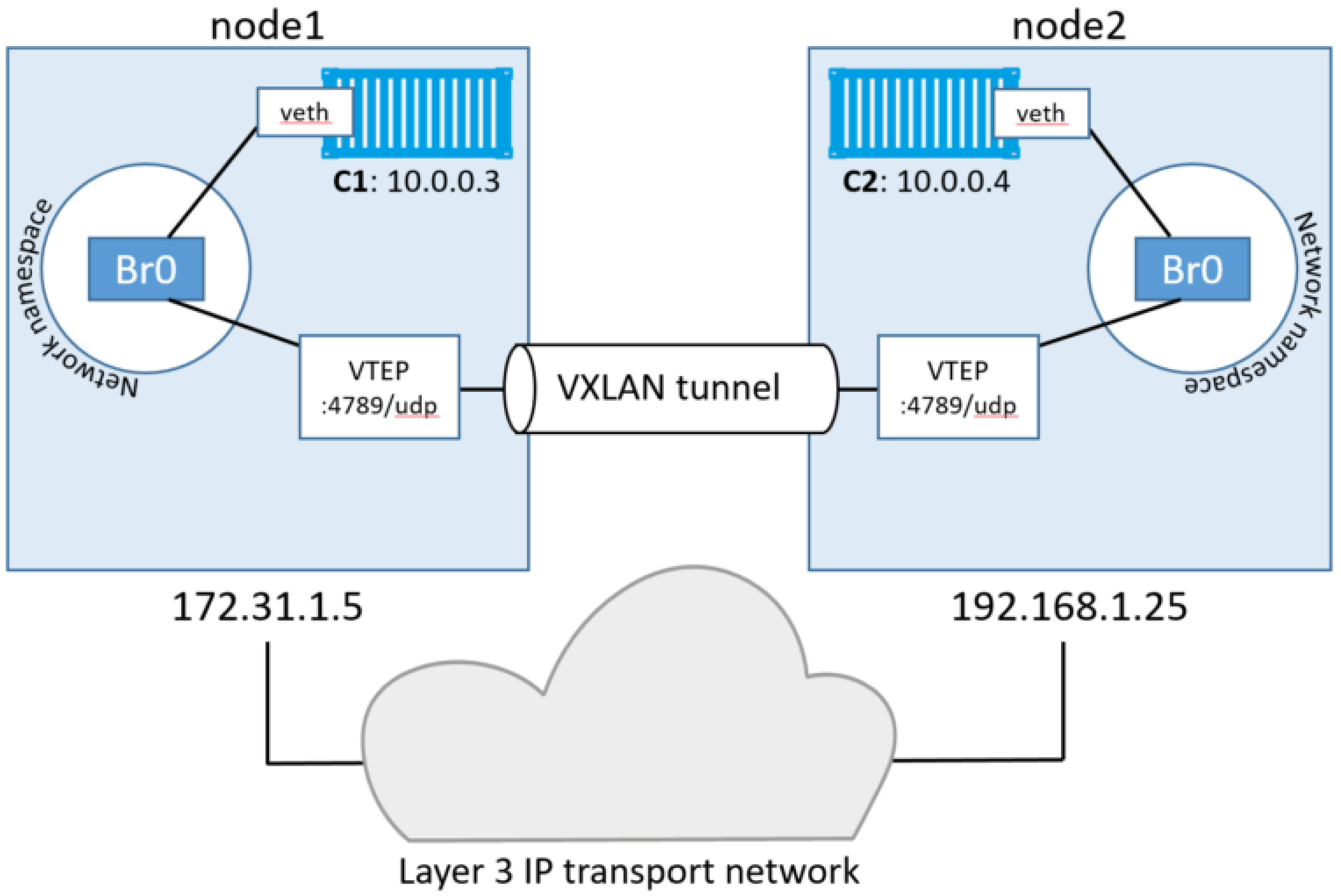
Overlay VXLAN



Overlay VXLAN



Overlay VXLAN



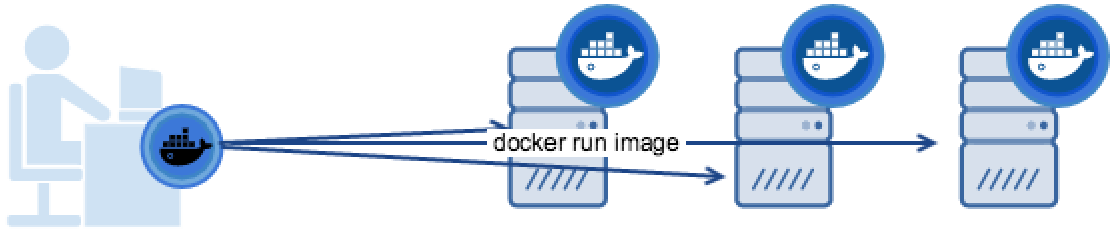
Docker machine

Docker machine

- Docker machine enables you to
  - Provision and manage multiple remote Docker hosts
  - Provision Swarm clusters
- Docker machine
  - Automatically creates hosts
  - Installs Docker Engine on them
  - Configures the docker clients (~/.docker)

- Drivers
  - AWS (ok)
  - Openstack (ok)
  - Virtualbox (ok)
  - Azure
  - Google Compute Engine
  - VMware

…



Docker compose

Docker compose

- Tool for defining and running multi-container Docker applications

  => compose file
- Create and start all the services

  => docker compose up

Docker compose

```
version: '2'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: wordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "80:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_PASSWORD: wordpress
volumes:
    db_data:
```

Docker Swarm

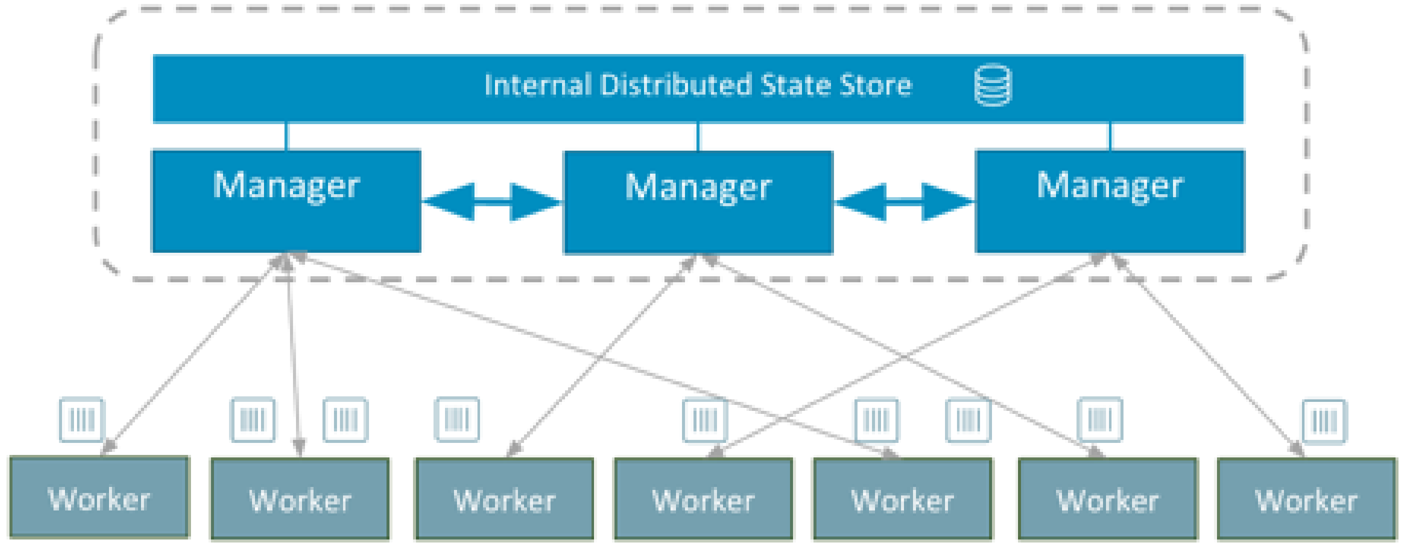
Docker Swarm

- Manage a cluster of Docker Engines:
    - Scaling
    - State reconciliation
    - Multi-host networking
    - Service discovery
    - Load balancing
    - Rolling updates
    - …

Schedulling

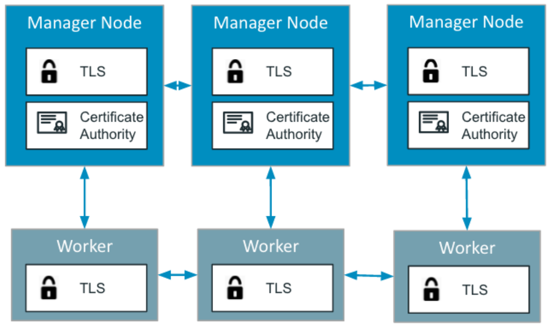
- Schedulling Strategies
  - Spread
  - Binpack
  - Random
- Filters
  - Node constraint filter
  - Node name
  - Label
  - Storage driver
  - etc
- Container filters
  - Affinity
  - Label
  - etc

Architecture



Encryption

- Control plane
  - A CA is created with swarm init on the manager nodes
  - All communication is encrypted over TLS.
  - The node keys and certificates are automatically renewed (default 90 days)



- Data plane
  - IPSEC tunnels with AES algorithm between nodes. Keys are automatically rotated by managers every 12 hours.

```
docker network create --opt encrypted --driver overlay my-multi-host-network
```

Internal Load balancing (from container to container)

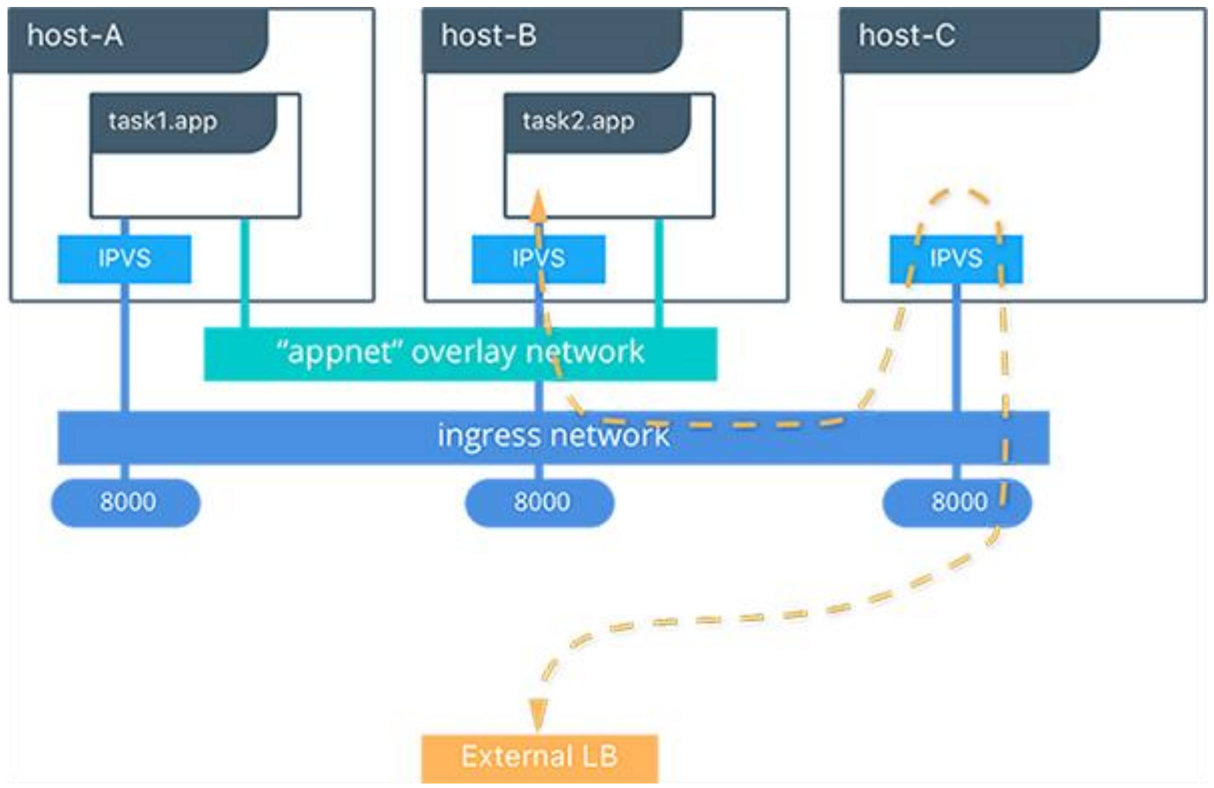
- 2 options (--endpoint-mode)
  - DNS RR
  - IPVS Virtual IP (VIP)

```
docker service inspect myservice
[. . .]

"VirtualIPs": [
            {
                "NetworkID": "a59umzkdj2r0ua7x8jxd84dhr",
                "Addr": "10.0.0.3/24"
            },
]
```

External Load balancing (to internet)

- Docker Routing Mesh



Stack

- Task = atomic unit ~ container
- Service = group of *n* tasks
- Stack = collection of services ~ application

```
services:
        lb:
          image: dockercloud/haproxy
          links:
            - web
          ports:
            - "80:80"
          roles:
            - global
        web:
          image: dockercloud/quickstart-python
          links:
            - redis
          target_num_containers: 4
        redis:
          image: redis
```

Secret management

- Use it for data that should not be
  - transmitted over a network
  - stored unencrypted in a Dockerfile or in your application's source code.

- such as a
  - Password
  - SSH private key
  - SSL certificate
  - etc

```
docker secret create
docker secret inspect
docker secret ls
docker secret rm
--secret flag for docker service create
```

Solocal GROUP

Security

Security

- SANS Institute checklist:
  - ❑ Ensure good host security
  - ❑ Check Image Provenance
  - ❑ Monitor Containers
  - ❑ Do Not Run Container Processes as Root
  - ❑ Do Not Store Secrets in Containers
  - ❑ Base Image Security
  - ❑ Limit container resources

https://www.sans.org/reading-room/whitepapers/auditing/checklist-audit-docker-containers-37437

News

New in 1.13

- Compose to deploy stack in Swarm
- New cli commands : docker container and docker image
- Docker system command
- Secret management

- Experimental :
    - Docker service logs
    - Docker Metrics in Prometheus format