马哥教育
最专业的Linux培训机构

Docker

主讲：马永亮(马哥)
QQ群:169777636
客服QQ：2813150558, 1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

Container

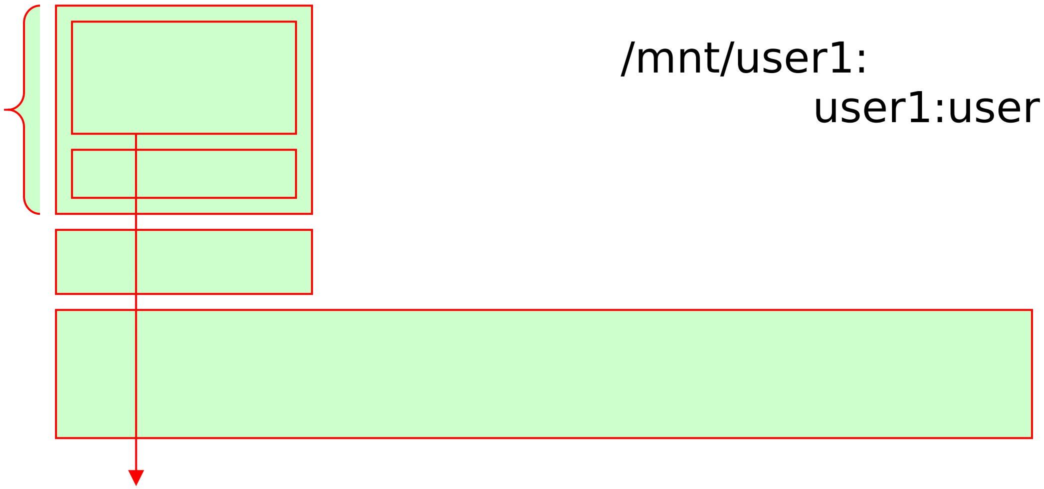
- 容器是一种基础工具；泛指任何可以用于容纳其它物品的工具，可以部分或完全封闭，被用于容纳、储存、运输物品；物体可以被放置在容器中，而容器则可以保护内容物；
- 人类使用容器的历史至少有十万年，甚至可能有数百万年的历史；
- 容器的类型
  - 瓶－指口部比腹部窄小、颈长的容器。
  - 罐－指那些开口较大、一般为近圆筒形的器皿。
  - 箱－通常是立方体或圆柱体。形状固定。
  - 篮－以条状物编织而成。
  - 桶－一种圆柱形的容器。
  - 袋－柔性材料制成的容器，形状会受内容物而变化。
  - 瓮－通常是指陶制，口小肚大的容器。
  - 碗－用来盛载食物的容器。
  - 柜－指一个由盒组成的家俱。
  - 鞘－用于装载刀刃的容器。

马哥教育
www.magedu.com

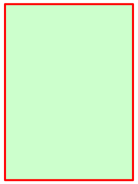
马哥教育
最专业的Linux培训机构

LXC

主讲：马永亮(马哥)
QQ群:169777636
客服QQ：2813150558, 1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

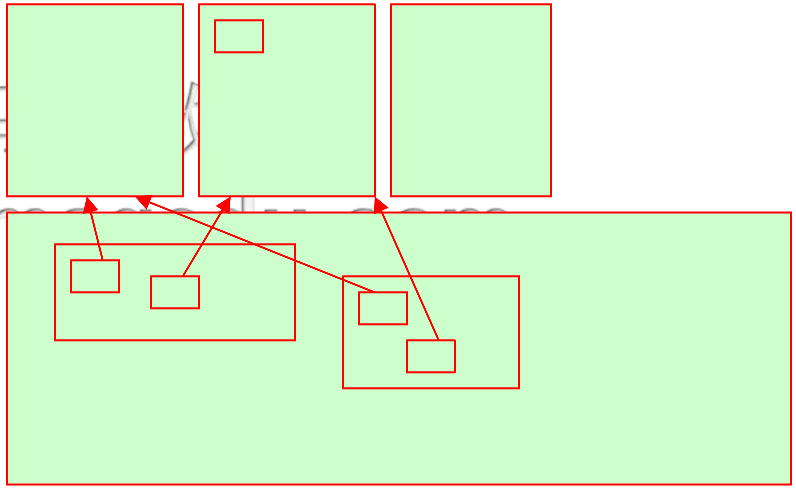


/mnt/user1:
user1:user1

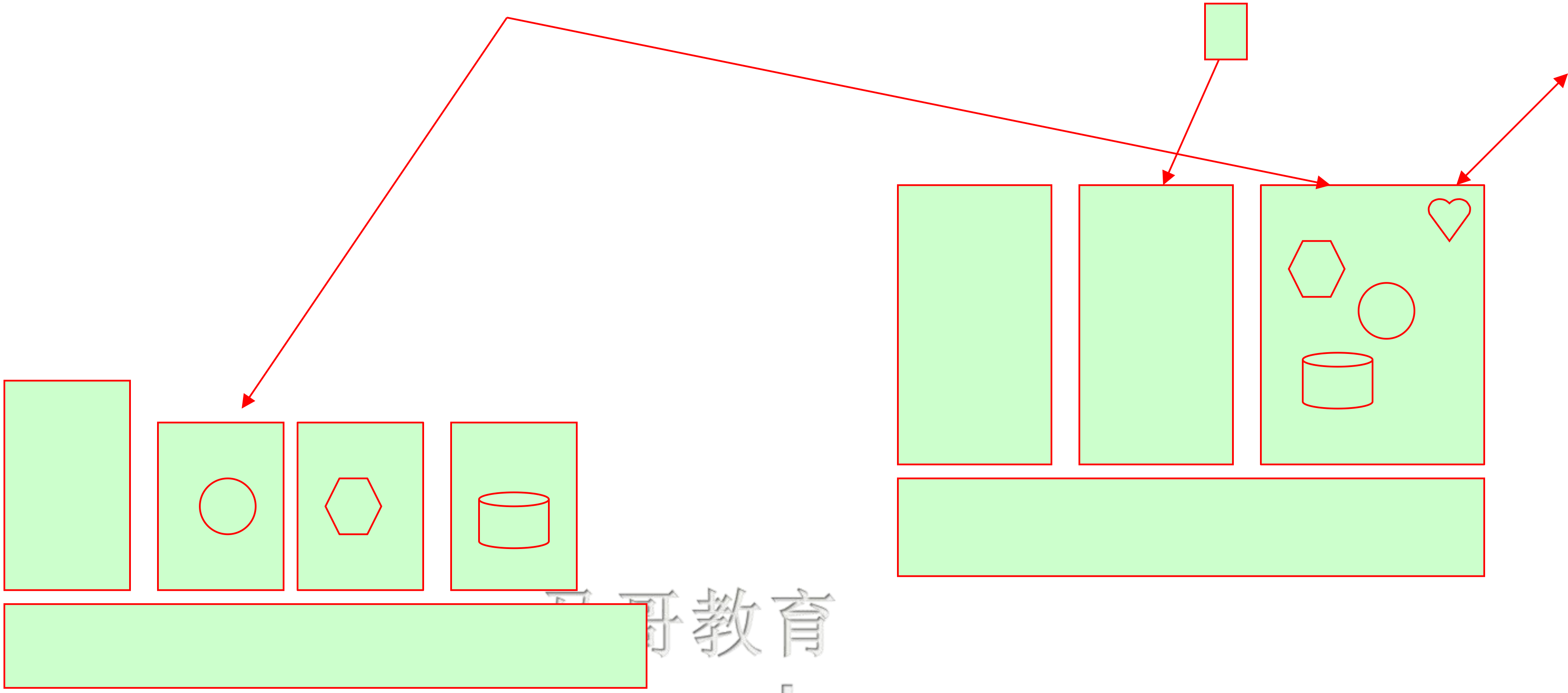


NameSpace
- mount
- PID
- user
- hostname, domain
- network
- ipc

Control Groups: CGroups

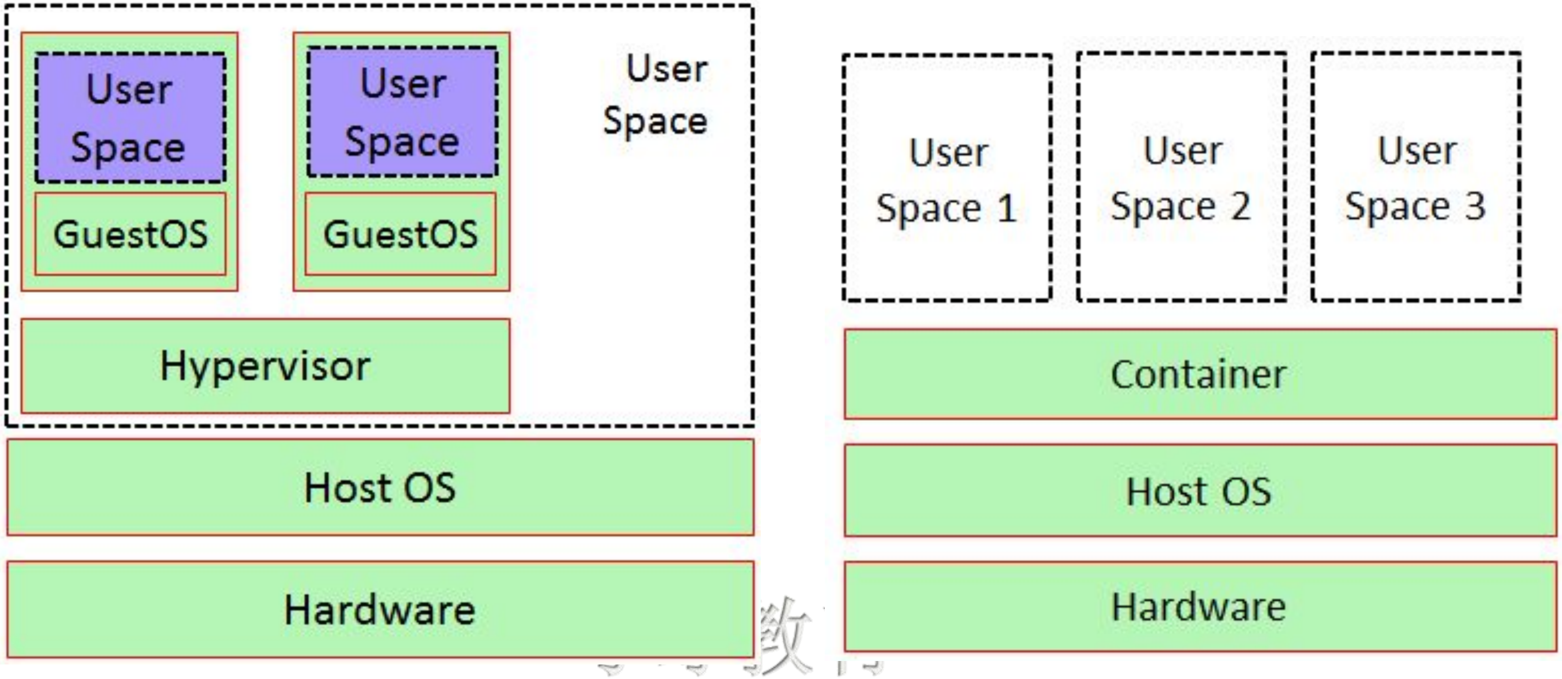


www.magedu.com



马哥教育
www.magedu.com

Virtualization and Container

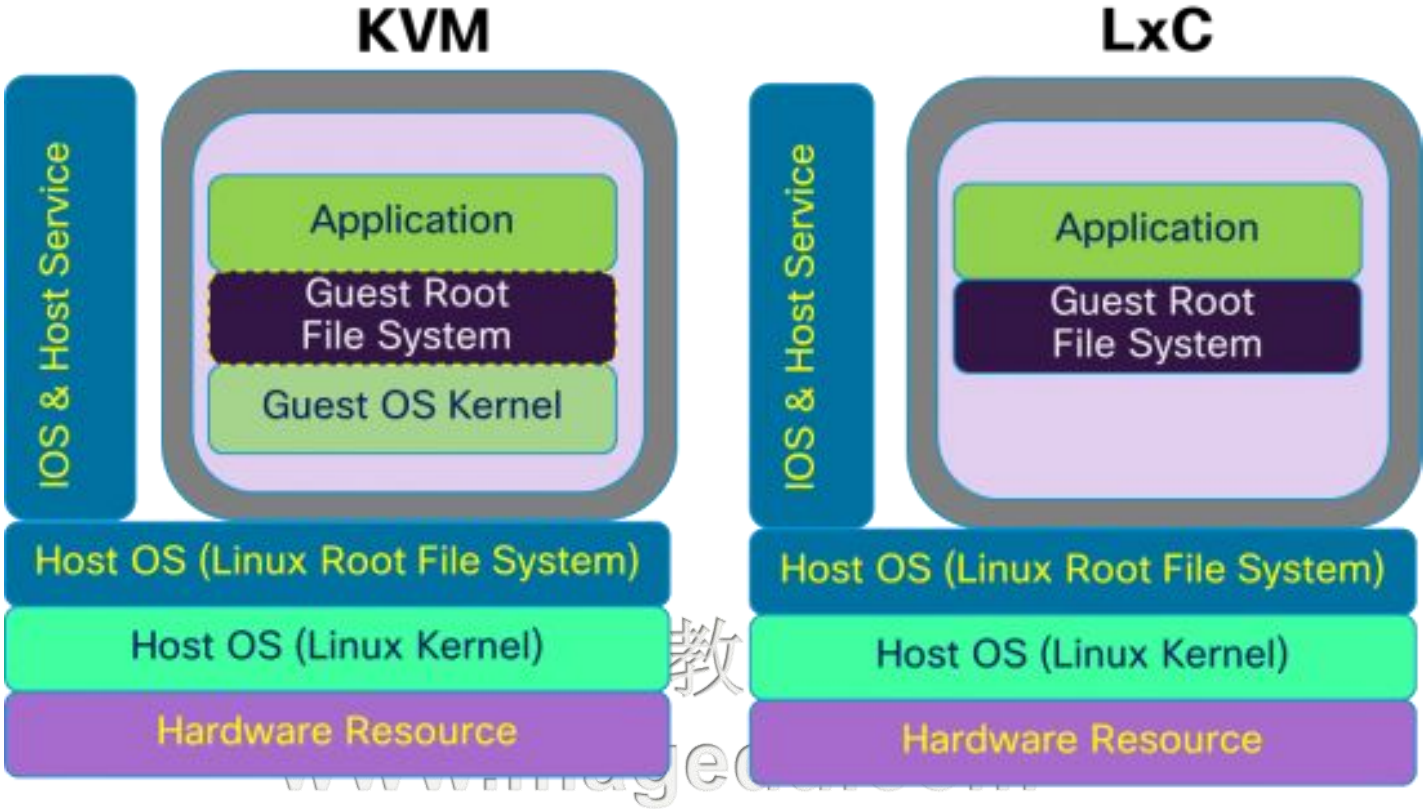


Host Virtualization

Container

www.magedu.com

kvm and lxc



教
www.magedu.com

What's LXC?

- LXC is a userspace interface for the Linux kernel containment features.
- Current LXC uses the following kernel features to contain processes
  - Kernel namespaces (ipc, uts, mount, pid, network and user)
  - Apparmor and SELinux profiles
  - Seccomp policies
  - Chroots (using pivot_root)
  - Kernel capabilities
  - CGroups (control groups)

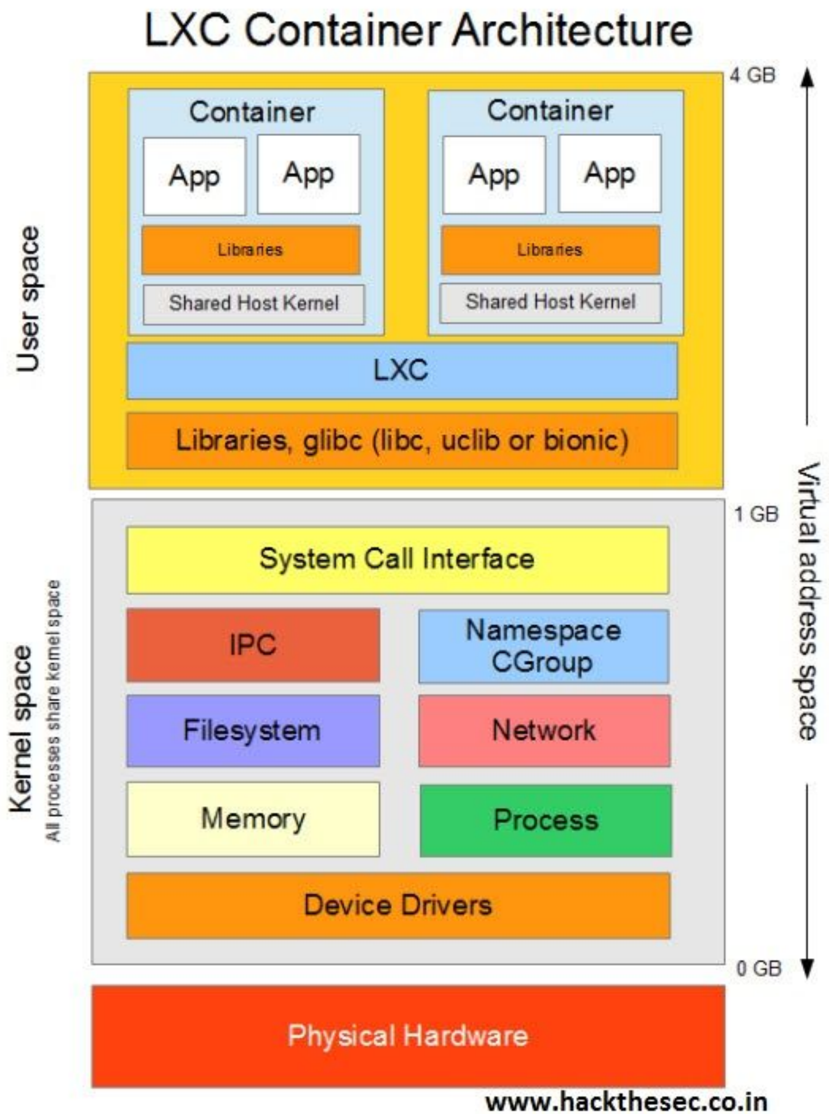
马哥教育
www.magedu.com

What's LXC?

- LXC containers are often considered as something in the middle between a chroot and a full fledged virtual machine.
- The goal of LXC is to create an environment as close as possible to a standard Linux installation but without the need for a separate kernel.

马哥教育
www.magedu.com

LXC Architecture



lxc简单应用

- 安装
  - lxc, lxc–templates
- 命令
  - lxc–checkconfig, lxc–ls, lxc–create, lxc–start, lxc–console, lxc–stop, lxc–info, lxc–clone, lxc–snapshot, lxc–destroy, …
  - WebGUI: lxc–webpanel
    - http://lxc–webpanel.github.io/
    - yum install python–flask
    - git clone https://github.com/lxc–webpanel/LXC–Web–Panel.git
    - python LXC–Web–Panel/lwp.py

马哥教育
www.magedu.com

马哥教育
www.magedu.com

LXC = Linux Containers?

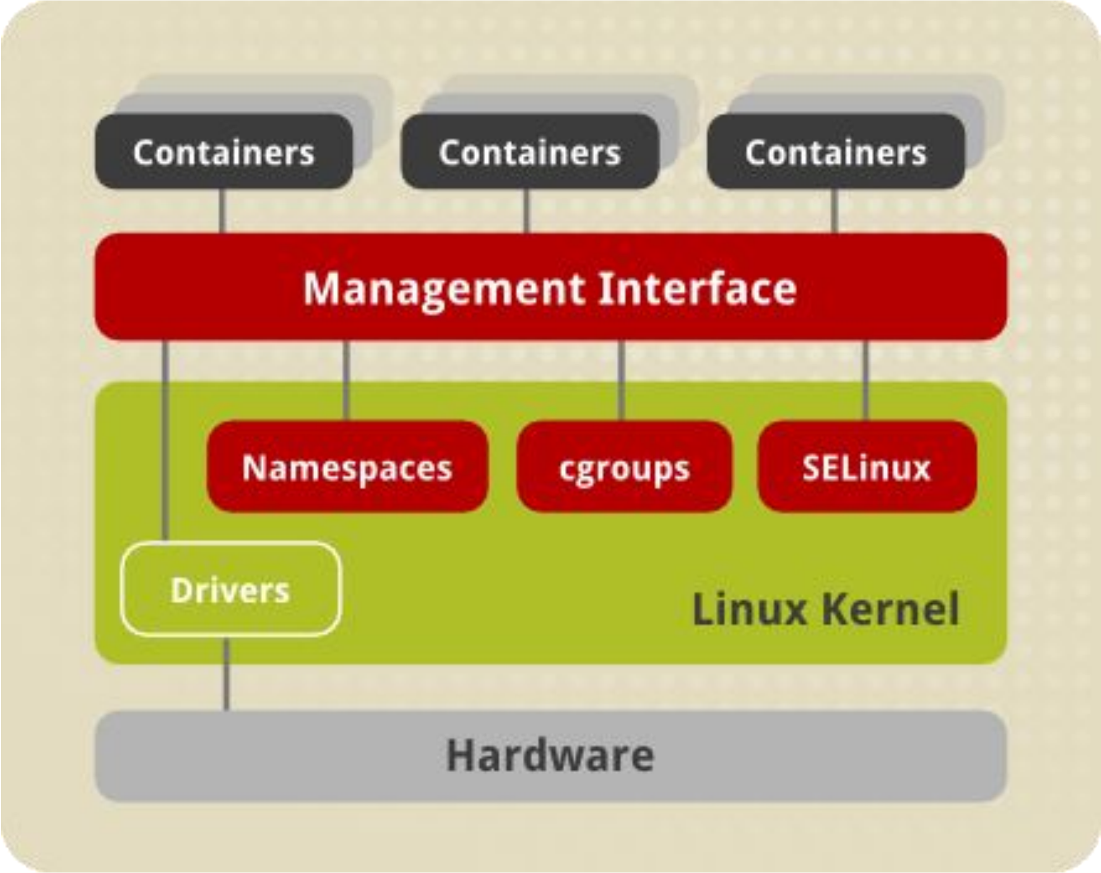
马哥教育
www.magedu.com

再谈容器

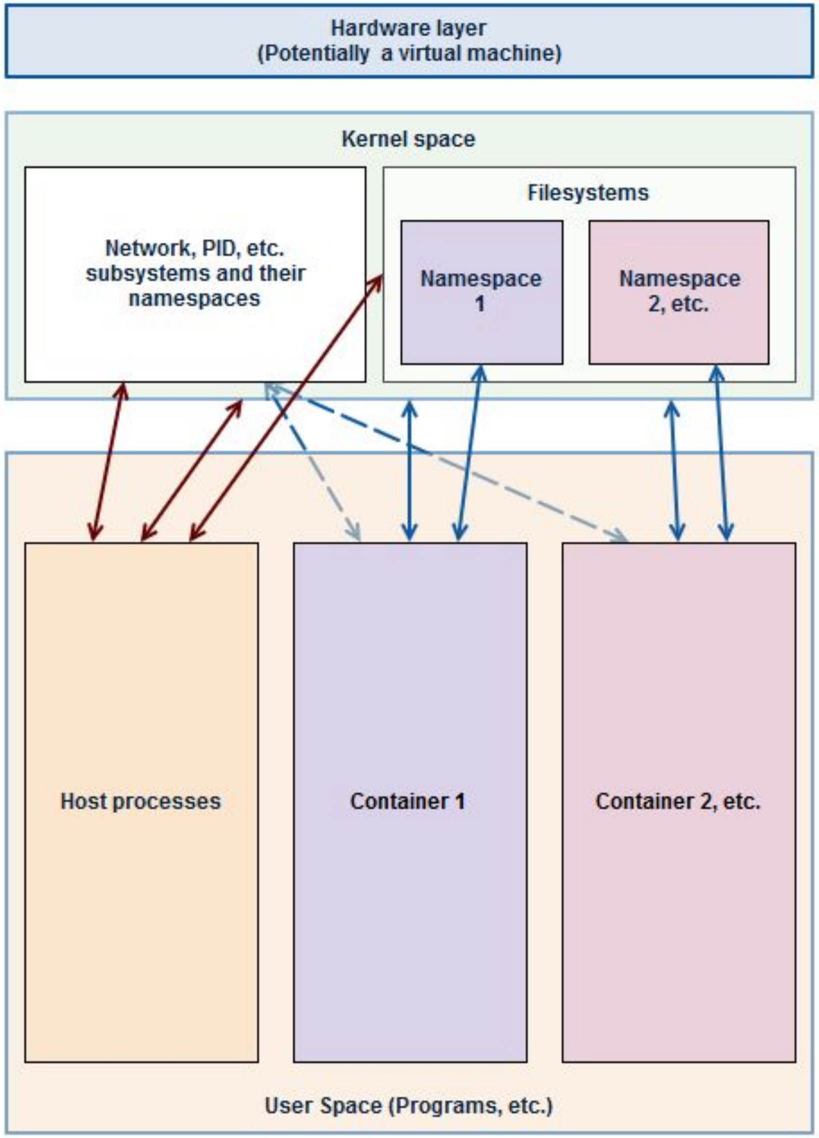
- Linux Containers have emerged as a key open source application packaging and delivery technology, combining lightweight application isolation with the flexibility of image-based deployment methods.
- CentOS 7 implements Linux Containers using core technologies such as Control Groups (Cgroups) for Resource Management, Namespaces for Process Isolation, SELinux for Security, enabling secure multi-tenancy and reducing the potential for security exploits
- lxc, containerd, openvz, systemd-nspawn, runC

马哥教育
www.magedu.com

Linux Container Architecture



Namespaces and Container


Hardware layer
(Potentially a virtual machine)
Kernel space
Network, PID, etc. subsystems and their namespaces
Filesystems
Namespace 1
Namespace 2, etc.
Host processes
Container 1
Container 2, etc.
User Space (Programs, etc.)

Linux Namespaces and CGroups

- Namespaces
  - Mount namespaces：挂载点
  - UTS namespaces：主机名与域名
  - IPC namespaces：信号量、消息队列和共享内存
  - PID namespaces：进程号
  - Network namespaces：网络设备、网络栈、端口等
  - User namespaces：用户和组

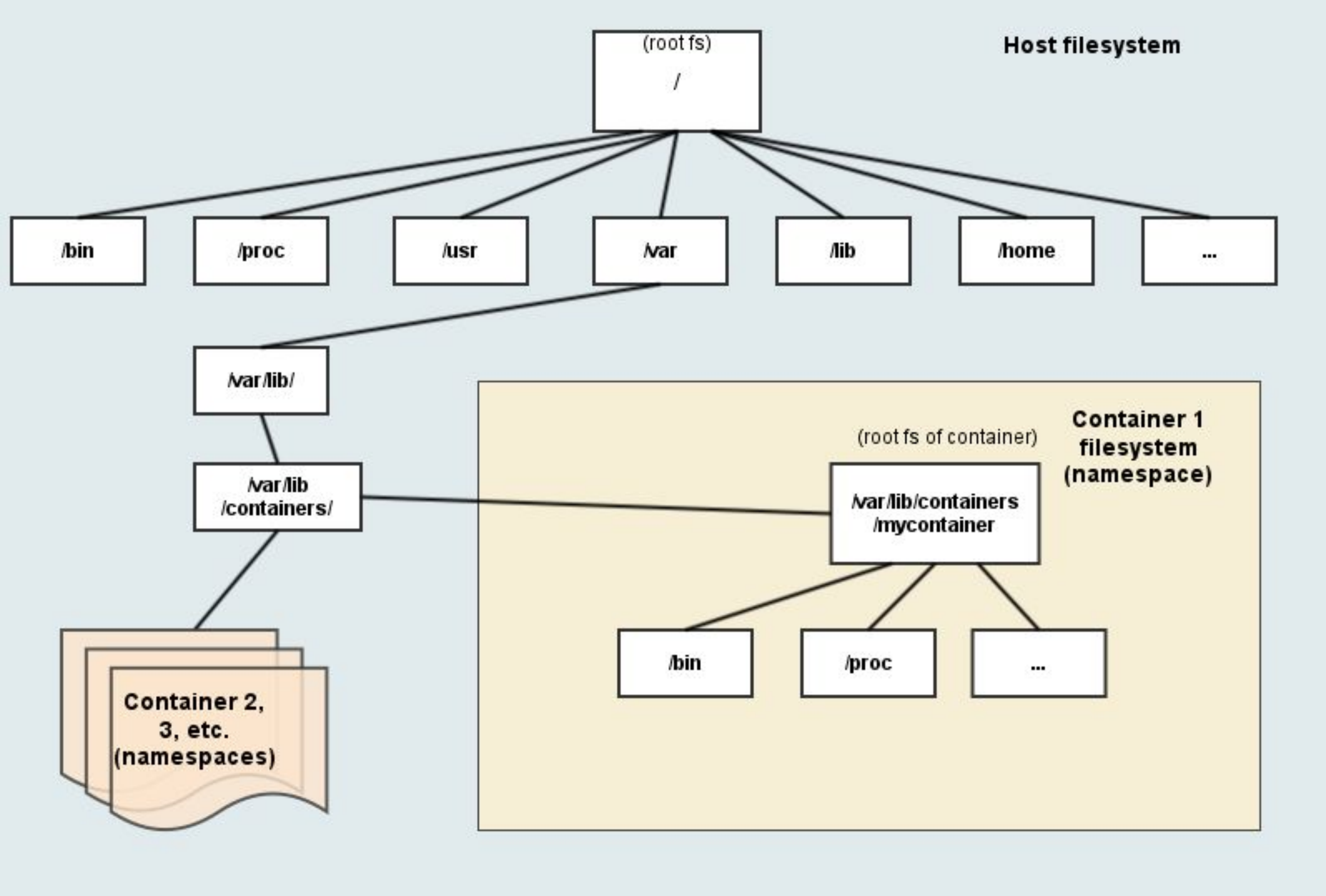
马哥教育
www.magedu.com

Control Groups (cgroups)

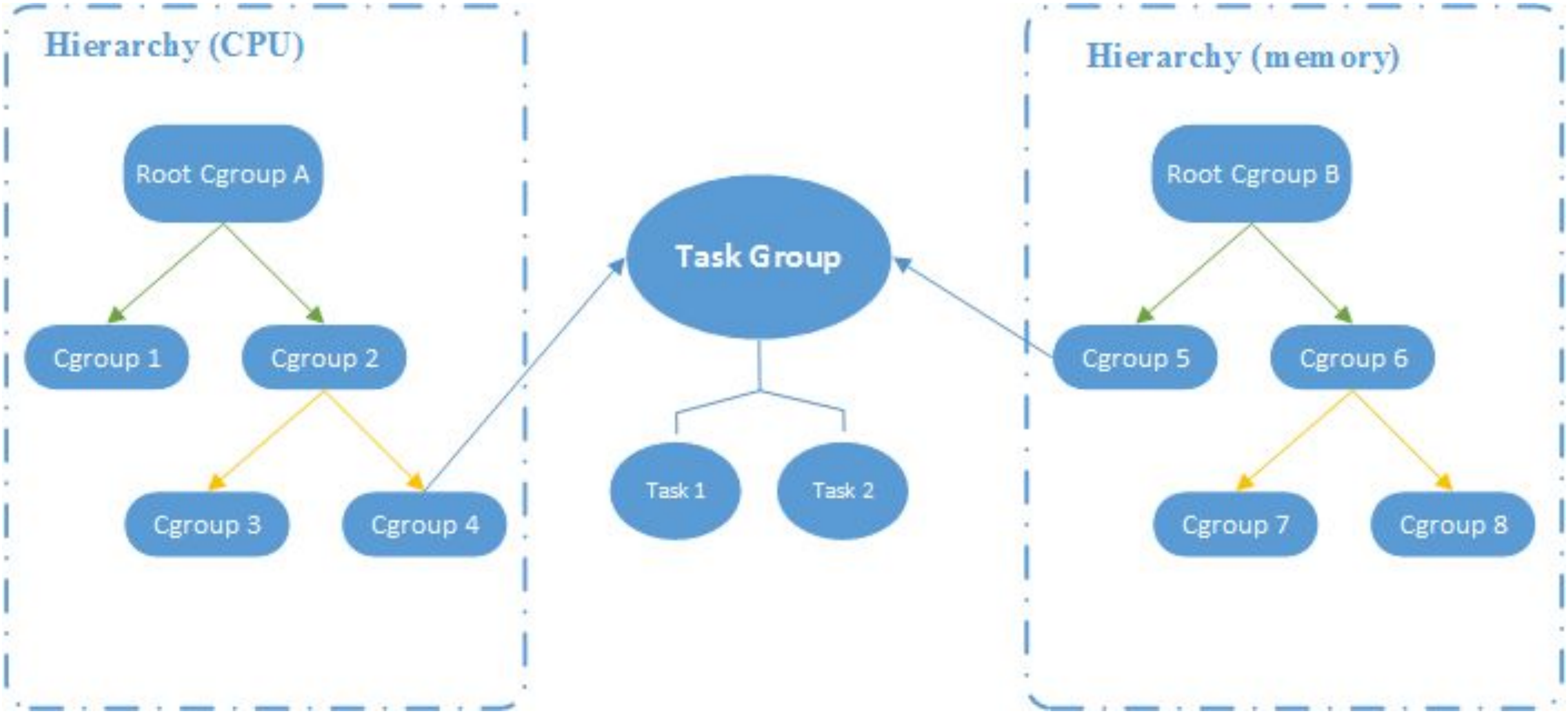
- cgroups
  - blkio：块设备IO
  - cpu：CPU
  - cpuacct：CPU资源使用报告
  - cpuset：多处理器平台上的CPU集合
  - devices：设备访问
  - freezer：挂起或恢复任务
  - memory：内存用量及报告
  - perf_event：对cgroup中的任务进行统一性能测试
  - net_cls：cgroup中的任务创建的数据报文的类别标识符

马哥教育
www.magedu.com

filesystem namespace



CGroups



www.magedu.com

马哥教育
最专业的Linux培训机构

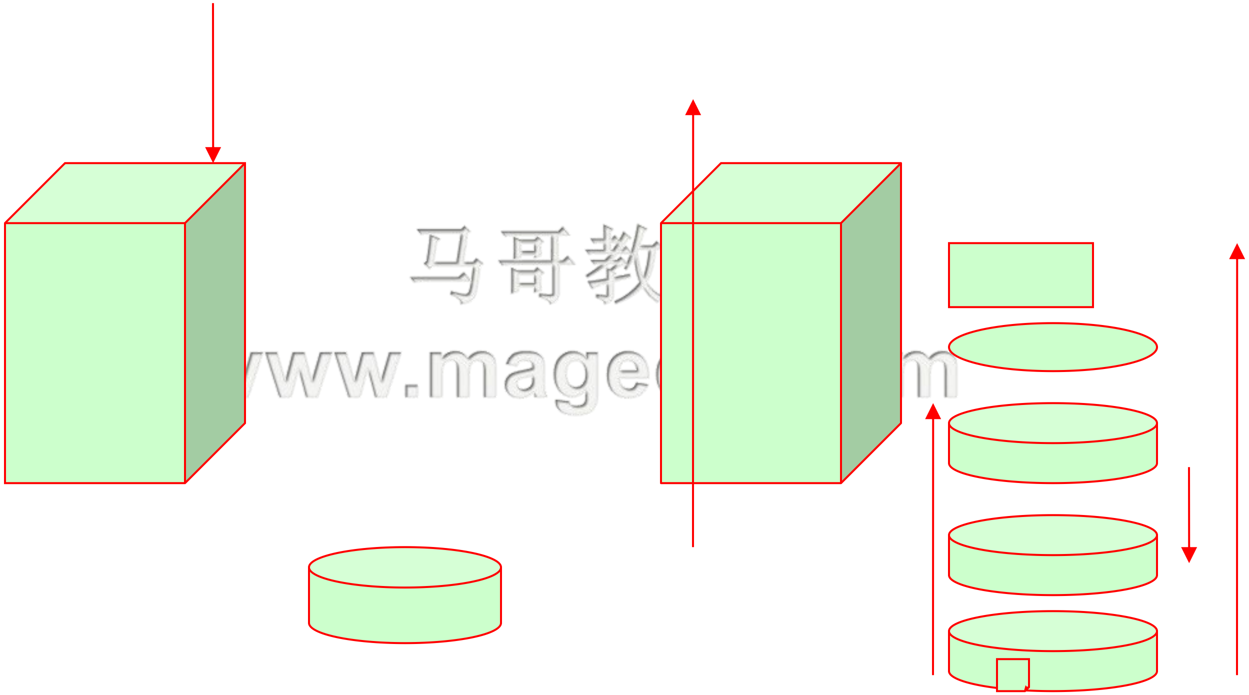
Docker入门

主讲：马永亮(马哥)
QQ群:169777636
客服QQ：2813150558, 1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

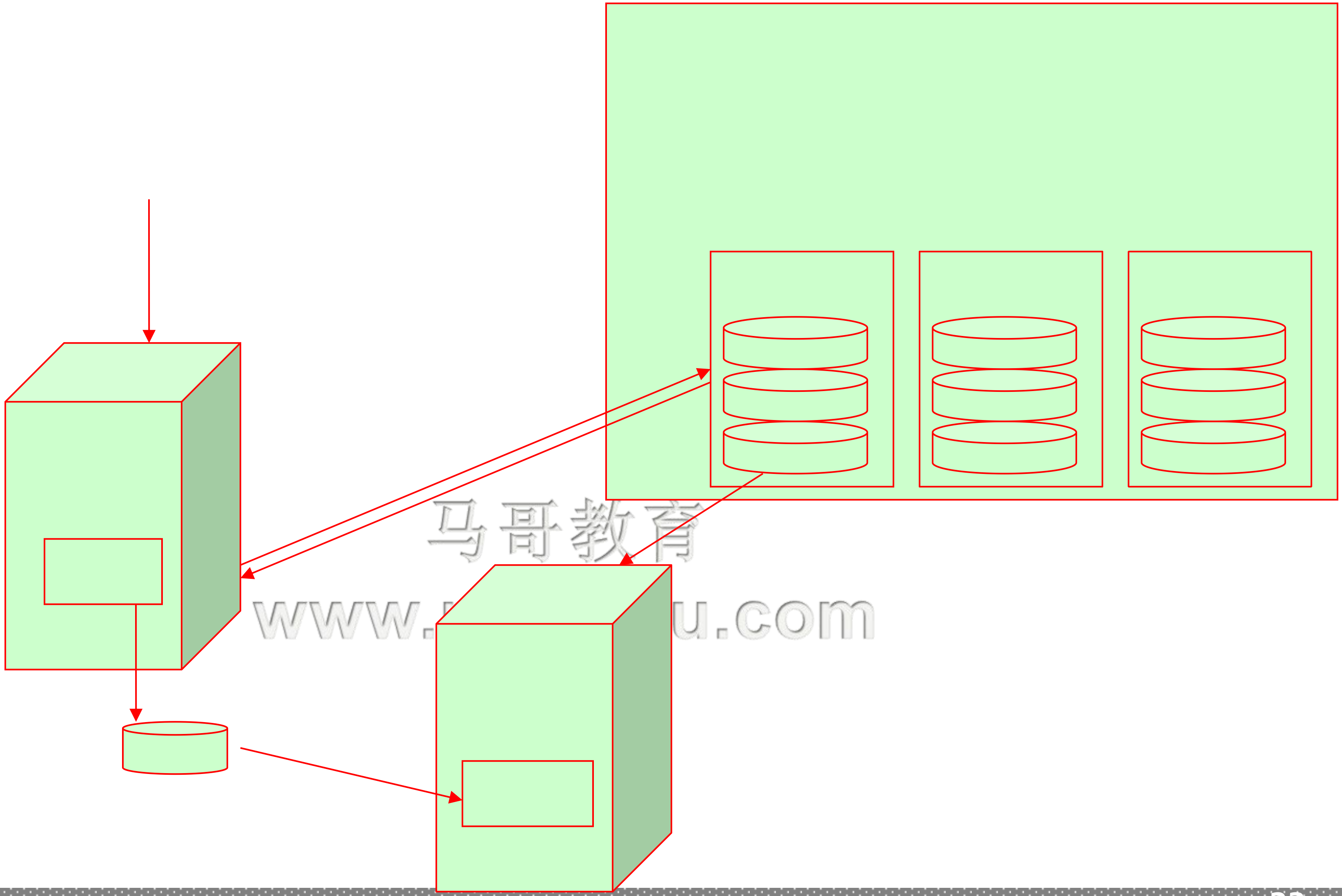
- NameSpaces, CGroups, Chroot
- SELinux, ...

/var/lib/lxc/c1

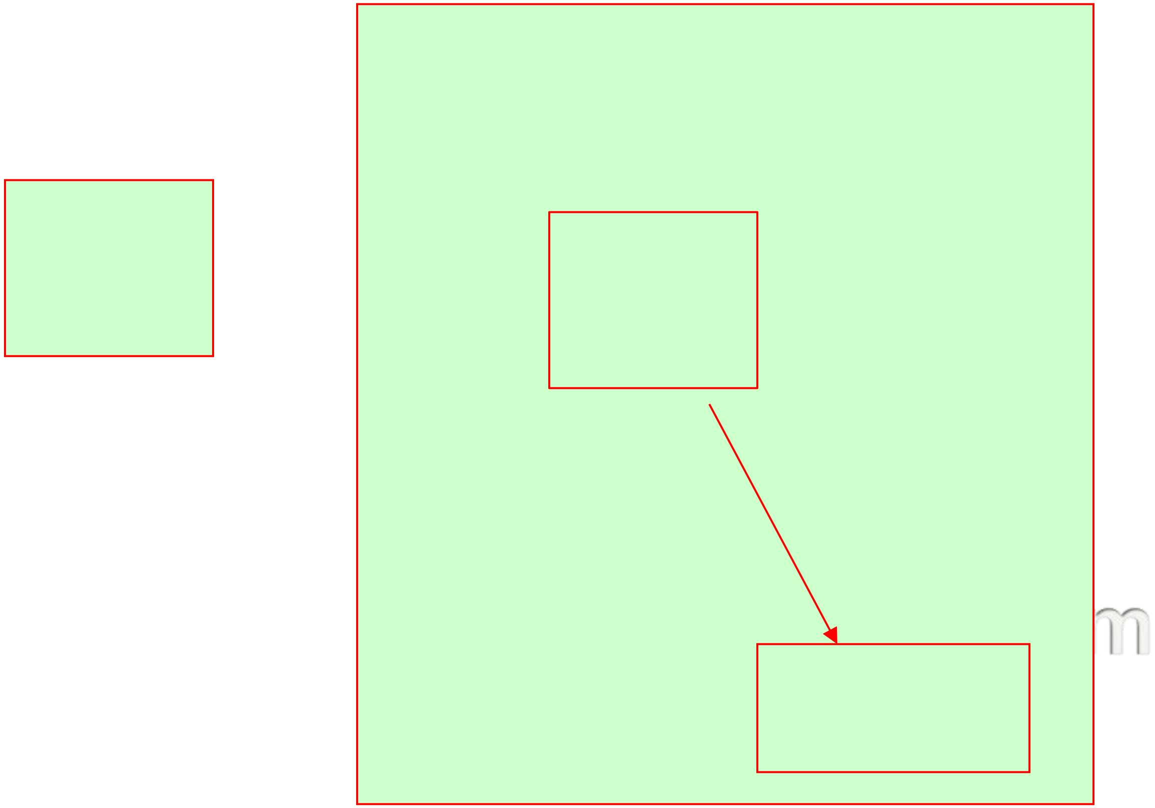
- lxc, openvz, ...



马哥教
www.magedu.com



马哥教育
www.magedu.com



What is docker

- docker中的容器
  - lxc --> libcontainer --> runC

OCI

- Open Container Initiative
  - 由Linux基金会主导于2015年6月创立
  - 旨在围绕容器格式和运行时制定一个开放的工业化标准
  - contains two specifications
    - the Runtime Specification (runtime–spec)
    - the Image Specification (image–spec)

  - The Runtime Specification outlines how to run a “filesystem bundle” that is unpacked on disk
  - At a high–level an OCI implementation would download an OCI Image then unpack that image into an OCI Runtime filesystem bundle

马哥教育
www.magedu.com

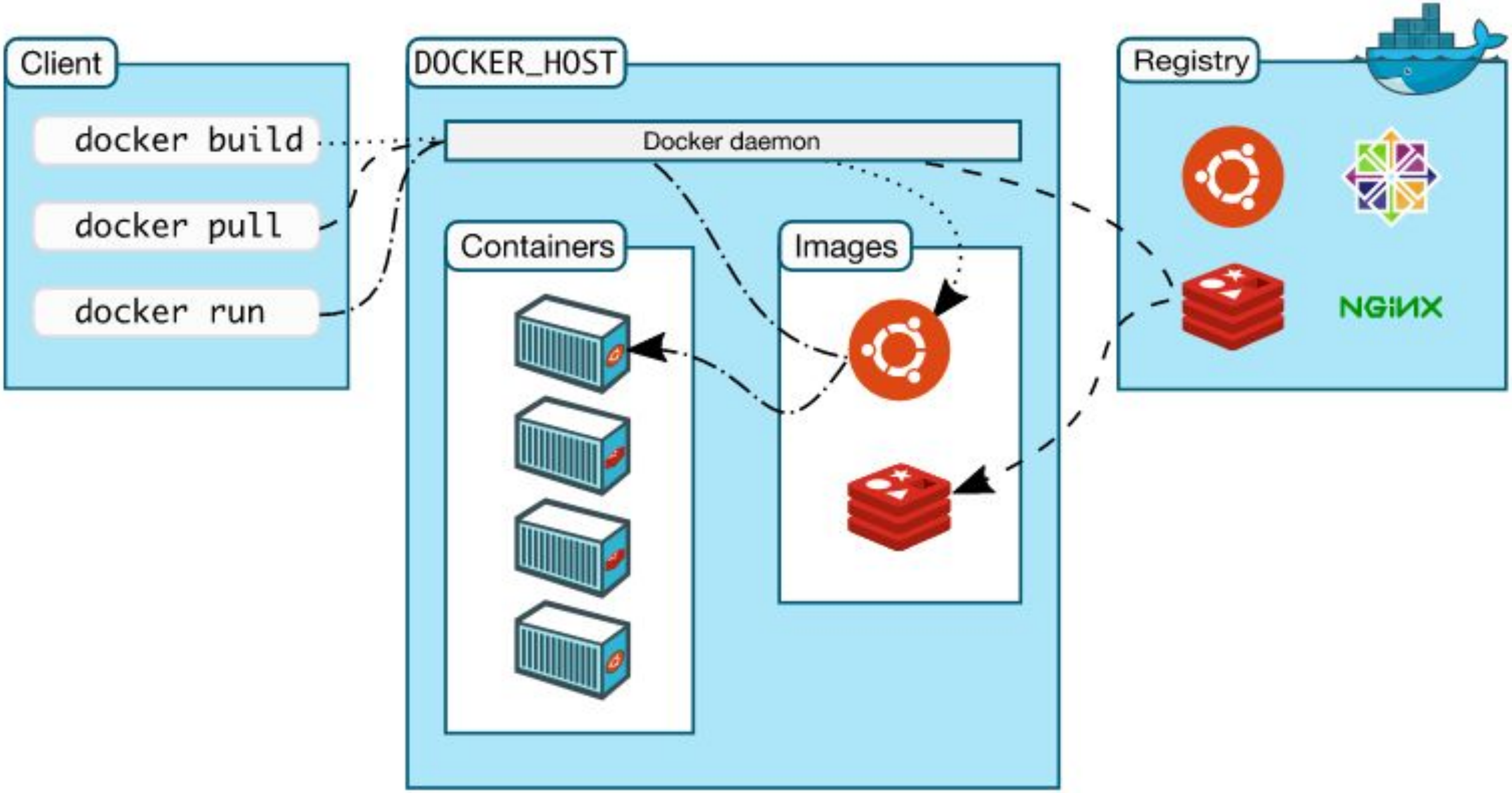
OPEN CONTAINER INITIATIVE

runC

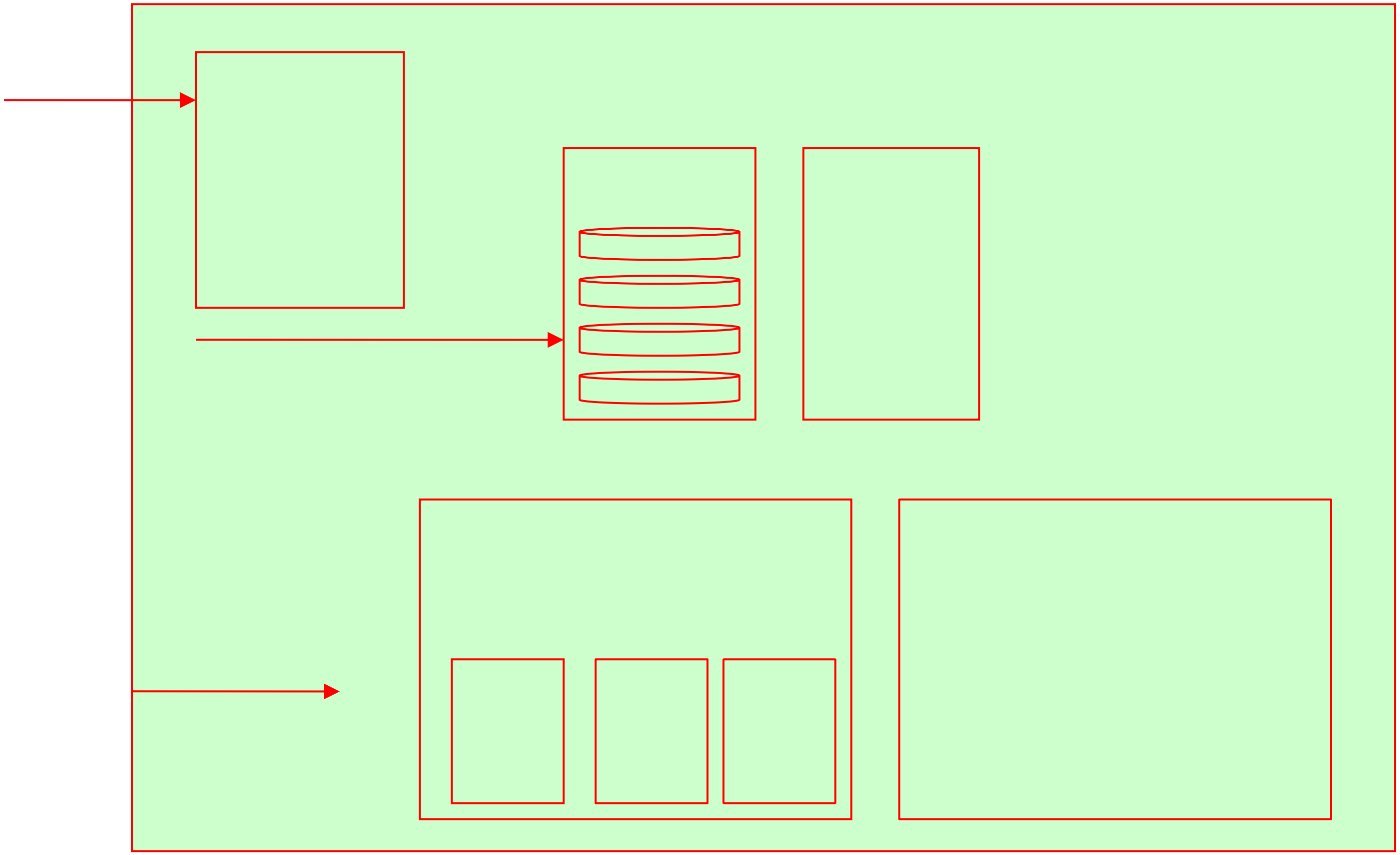
- OCF：Open Container Format
- runC is a CLI tool for spawning and running containers according to the OCI specification
  - Containers are started as a child process of runC and can be embedded into various other systems without having to run a daemon
  - runC is built on libcontainer, the same container technology powering millions of Docker Engine installations

马哥教育
www.magedu.com

docker architecture



Registry



docker architecture

- The Docker daemon
  - The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes.
- The Docker client
  - The Docker client (docker) is the primary way that many Docker users interact with Docker.
  - The docker command uses the Docker API.
- Docker registries
  - A Docker registry stores Docker images.
  - Docker Hub and Docker Cloud are public registries that anyone can use, and Docker is configured to look for images on Docker Hub by default.
  - You can even run your own private registry.

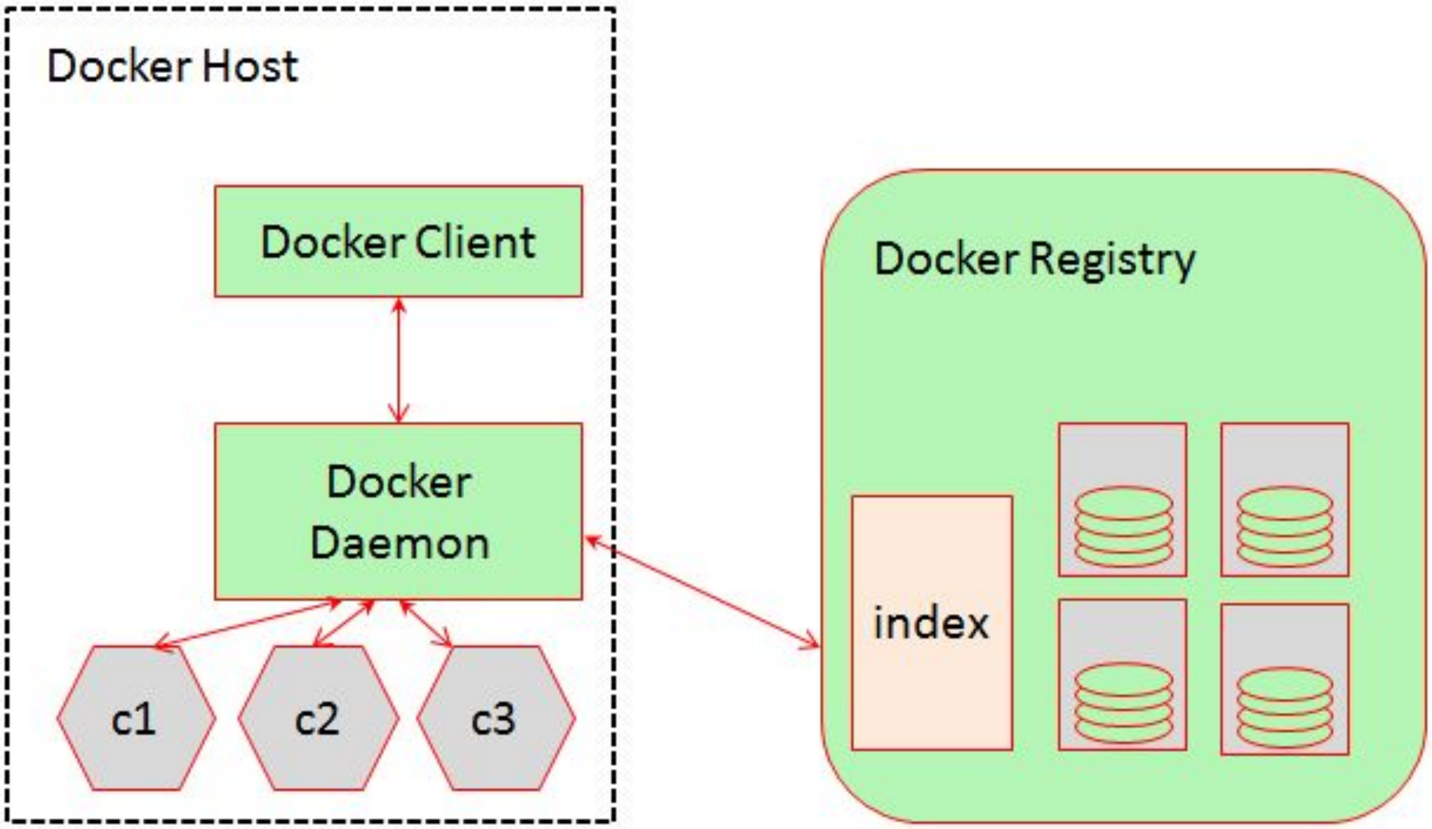
马哥教育
www.magedu.com

Docker objects

- When you use Docker, you are creating and using images, containers, networks, volumes, plugins, and other objects.
  - IMAGES
    - An image is a read–only template with instructions for creating a Docker container.
    - Often, an image is based on another image, with some additional customization.
    - You might create your own images or you might only use those created by others and published in a registry.
  - CONTAINERS
    - A container is a runnable instance of an image.
    - You can create, run, stop, move, or delete a container using the Docker API or CLI.
    - You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.

马哥教育
www.magedu.com

Docker Host and Registry



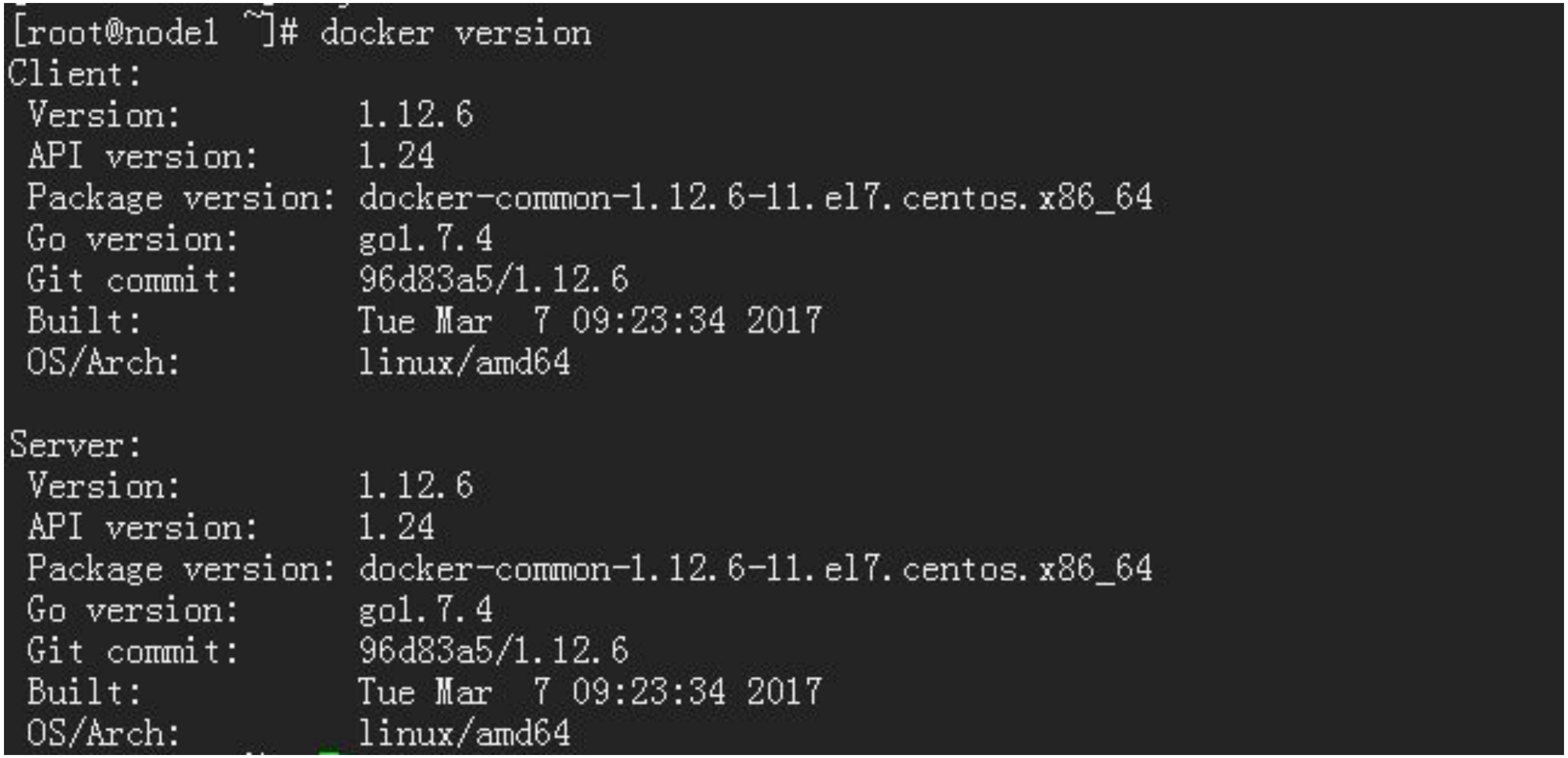
安装及使用docker

- 依赖的基础环境
  - 64 bits CPU
  - Linux Kernel 3.10+
  - Linux Kernel cgroups and namespaces
- CentOS 7
  - “Extras” repository
- Docker Daemon
  - systemctl start docker.service
- Docker Client
  - docker [OPTIONS] COMMAND [arg...]

马哥教育
www.magedu.com

Investigating the Docker environment

- docker version
  - The version option shows which versions of different Docker components are installed

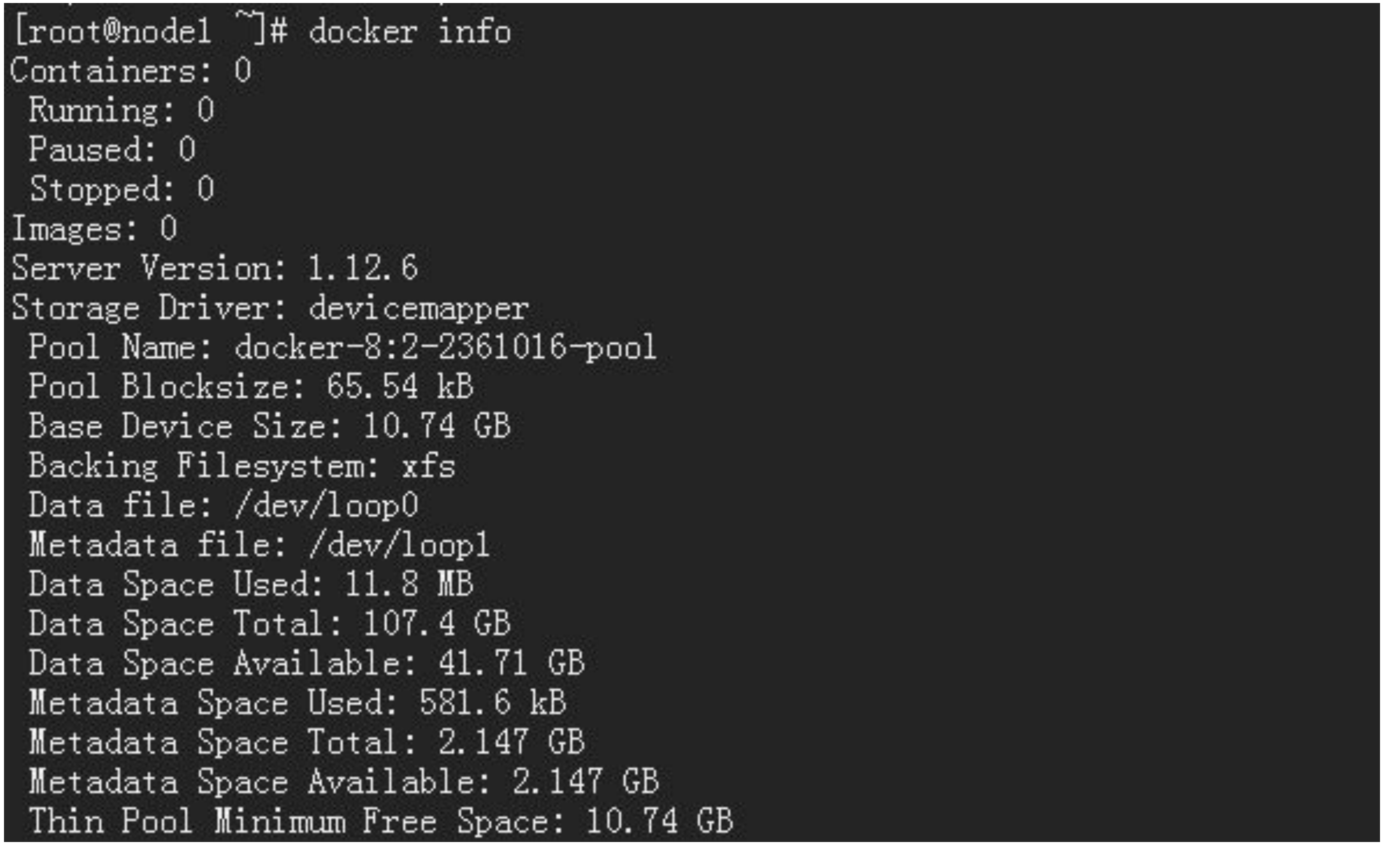


```
[root@node1 ~]# docker version
Client:
 Version:         1.12.6
 API version:     1.24
 Package version: docker-common-1.12.6-11.el7.centos.x86_64
 Go version:      go1.7.4
 Git commit:      96d83a5/1.12.6
 Built:           Tue Mar  7 09:23:34 2017
 OS/Arch:         linux/amd64

Server:
 Version:         1.12.6
 API version:     1.24
 Package version: docker-common-1.12.6-11.el7.centos.x86_64
 Go version:      go1.7.4
 Git commit:      96d83a5/1.12.6
 Built:           Tue Mar  7 09:23:34 2017
 OS/Arch:         linux/amd64
```

Investigating the Docker environment

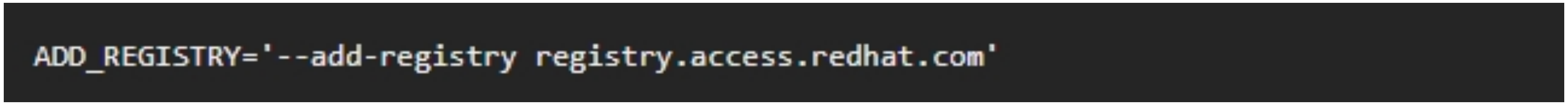
- docker info
  - The info option lets you see how many local containers and images there are, as well as information on the size and location of Docker storage areas



```
[root@node1 ~]# docker info
Containers: 0
 Running: 0
 Paused: 0
 Stopped: 0
Images: 0
Server Version: 1.12.6
Storage Driver: devicemapper
 Pool Name: docker-8:2-2361016-pool
 Pool Blocksize: 65.54 kB
 Base Device Size: 10.74 GB
 Backing Filesystem: xfs
 Data file: /dev/loop0
 Metadata file: /dev/loop1
 Data Space Used: 11.8 MB
 Data Space Total: 107.4 GB
 Data Space Available: 41.71 GB
 Metadata Space Used: 581.6 kB
 Metadata Space Total: 2.147 GB
 Metadata Space Available: 2.147 GB
 Thin Pool Minimum Free Space: 10.74 GB
```

Registry options

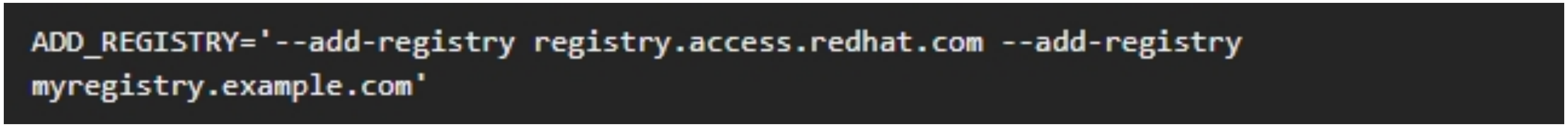
- When asked to search for or pull images, the docker command uses the Docker registry (docker.io) to complete those activities
- In RHEL and RHEL Atomic Host, this entry in the /etc/sysconfig/docker file causes the Red Hat registry (registry.access.redhat.com) to be used first:



```
ADD_REGISTRY='--add-registry registry.access.redhat.com'
```

- If you wanted to add a private registry that you installed yourself, just add another ——add–registry. For example:

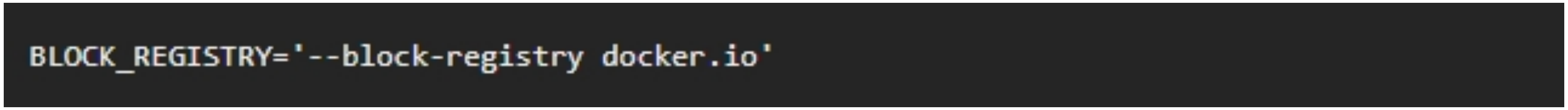
马哥教育



```
ADD_REGISTRY='--add-registry registry.access.redhat.com --add-registry
myregistry.example.com'
```

Registry options(2)

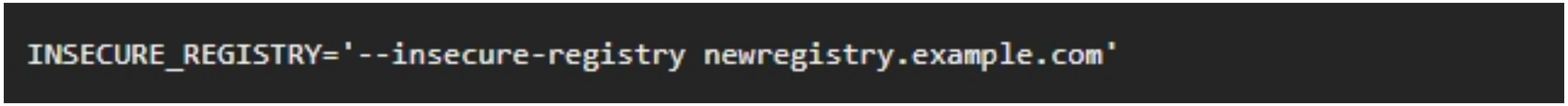
- If you want to prevent users from pulling images from the Docker registry, uncomment the BLOCK_REGISTRY entry so it appears as follows:



```
BLOCK_REGISTRY='--block-registry docker.io'
```

- To access a registry that uses https protocol for security, but is not set up with certificates for authentication, you can still access that registry by defining it as an insecure registry in the /etc/sysconfig/docker file. For example:

马哥教育



```
INSECURE_REGISTRY='--insecure-registry newregistry.example.com'
```

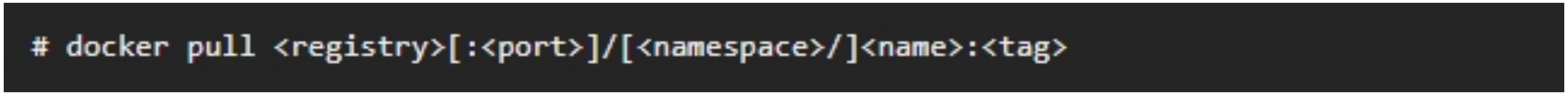
常用操作

- docker search： Search the Docker Hub for images
- docker pull： Pull an image or a repository from a registry
- docker images： List images
- docker create： Create a new container
- docker start： Start one or more stopped containers
- docker run： Run a command in a new container
- docker attach： Attach to a running container
- docker ps： List containers
- docker logs： Fetch the logs of a container
- docker restart： Restart a container
- docker stop： Stop one or more running containers
- docker kill： Kill one or more running containers
- docker rm： Remove one or more containers

马哥教育
www.magedu.com

Getting images from remote Docker registries

- To get Docker images from a remote registry (such as your own Docker registry) and add them to your local system, use the docker pull command:

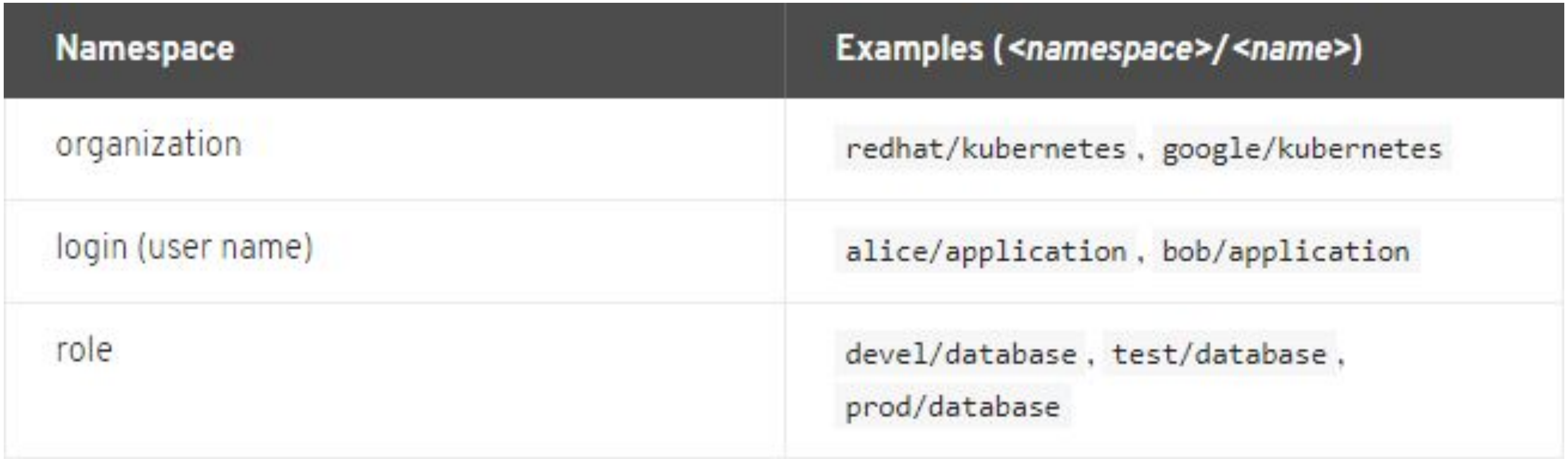


```
# docker pull <registry>[:<port>]/[<namespace>/]<name>:<tag>
```

  - The <registry> is a host that provides the docker–distribution service on TCP <port> (default: 5000)
  - Together, <namespace> and <name> identify a particular image controlled by <namespace> at that registry
    - Some registries also support raw <name>; for those, <namespace> is optional
    - When it is included, however, the additional level of hierarchy that <namespace> provides is useful to distinguish between images with the same <name>

马哥教育
www.magedu.com

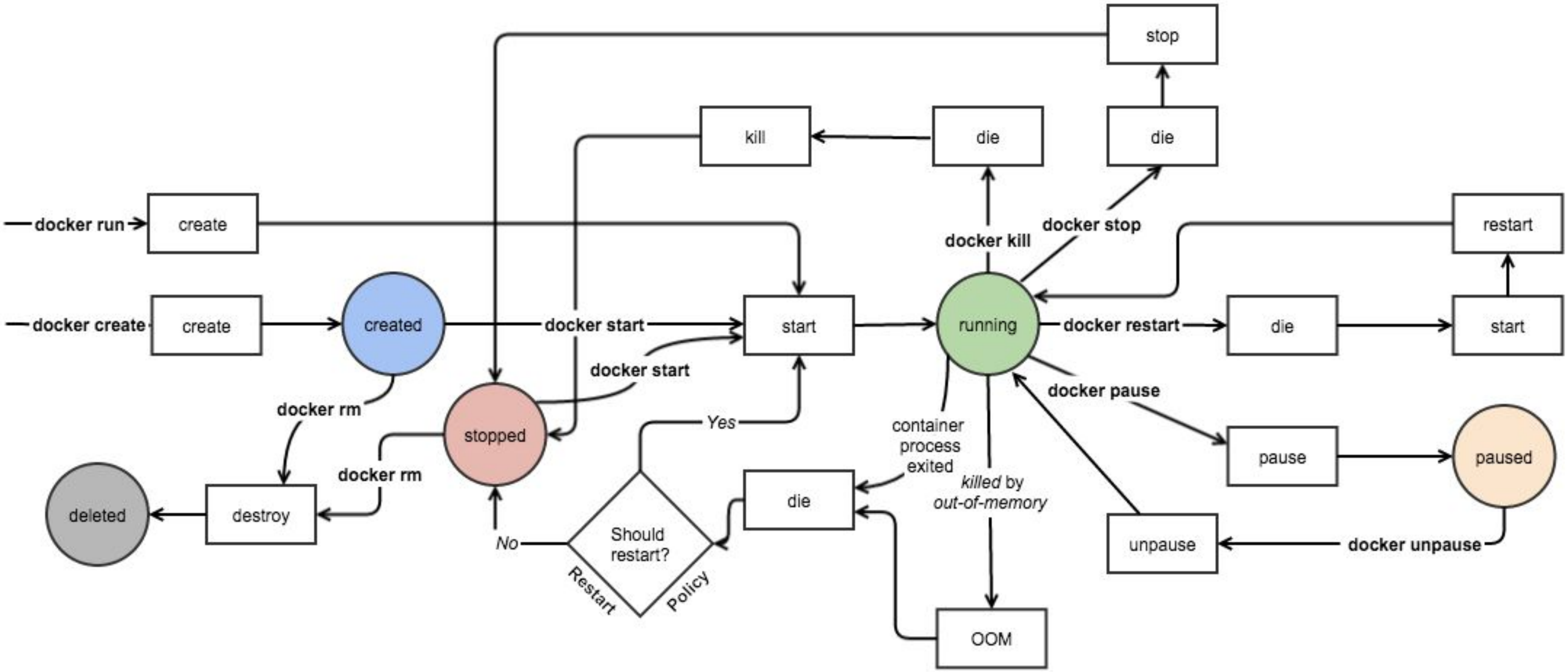
The additional level of hierarchy of <namespace>



Namespace	Examples (*<namespace>/<name>*)
organization	`redhat/kubernetes`, `google/kubernetes`
login (user name)	`alice/application`, `bob/application`
role	`devel/database`, `test/database`, `prod/database`

马哥教育
www.magedu.com

docker event state



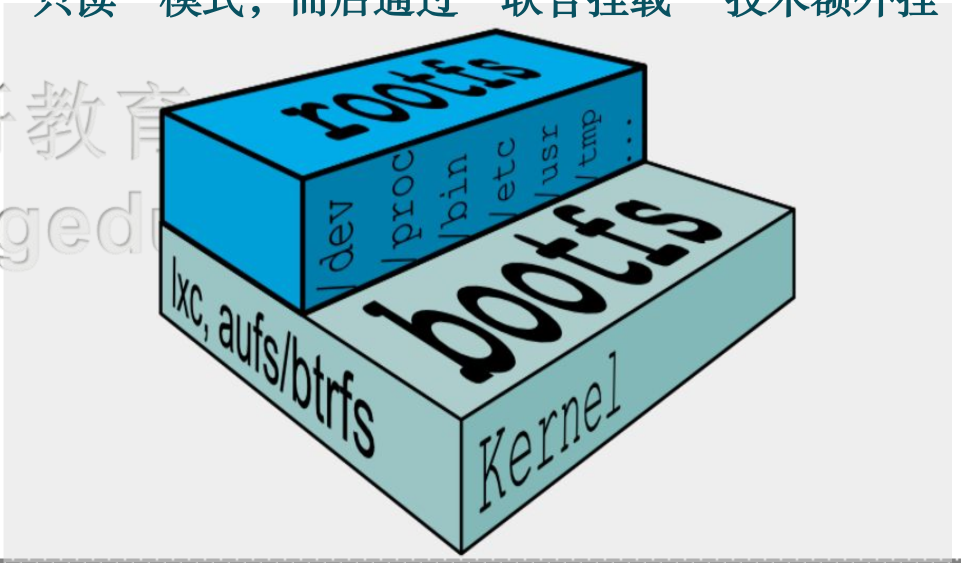
Docker images

主讲：马永亮(马哥)
QQ群:169777636
客服QQ：2813150558, 1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

About Docker Images

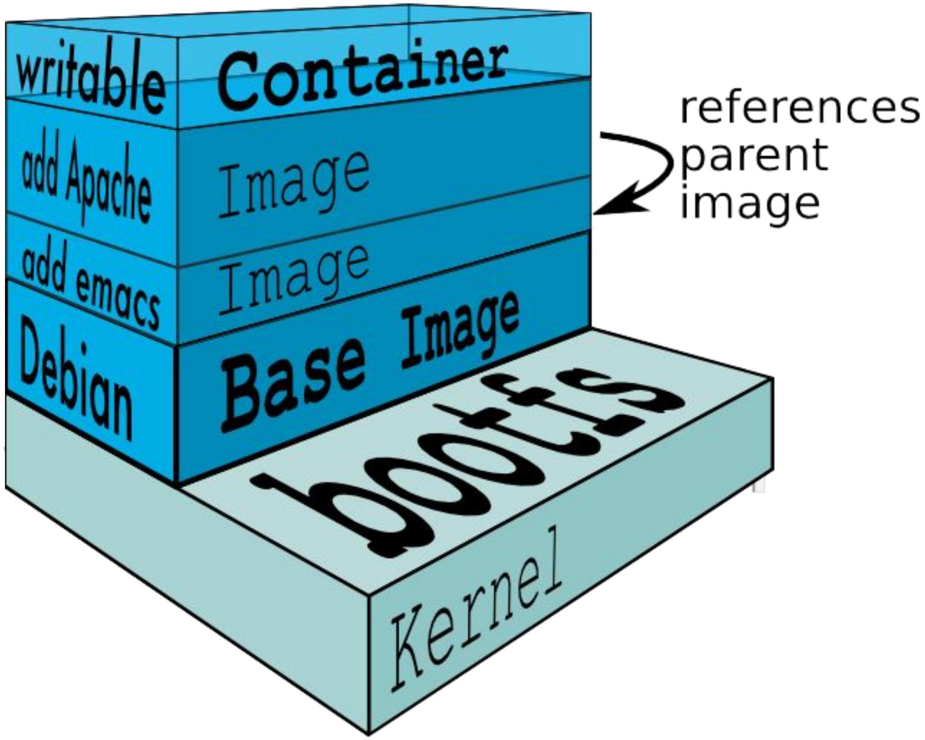
- Docker镜像含有启动容器所需要的文件系统及其内容，因此，其用于创建并启动docker容器
  - 采用分层构建机制，最底层为bootfs，其之为rootfs
    - bootfs：用于系统引导的文件系统，包括bootloader和kernel，容器启动完成后会被卸载以节约内存资源；
    - rootfs：位于bootfs之上，表现为docker容器的根文件系统；
      - 传统模式中，系统启动之时，内核挂载rootfs时会首先将其挂载为“只读”模式，完整性自检完成后将其重新挂载为读写模式；
      - docker中，rootfs由内核挂载为“只读”模式，而后通过“联合挂载 ”技术额外挂载一个“可写”层；

马哥教育
www.magedu.com



Docker Image Layer

- 位于下层的镜像称为父镜像(parent image)，最底层的称为基础镜像(base image)
- 最上层为“可读写”层，其下的均为“只读”层



Aufs

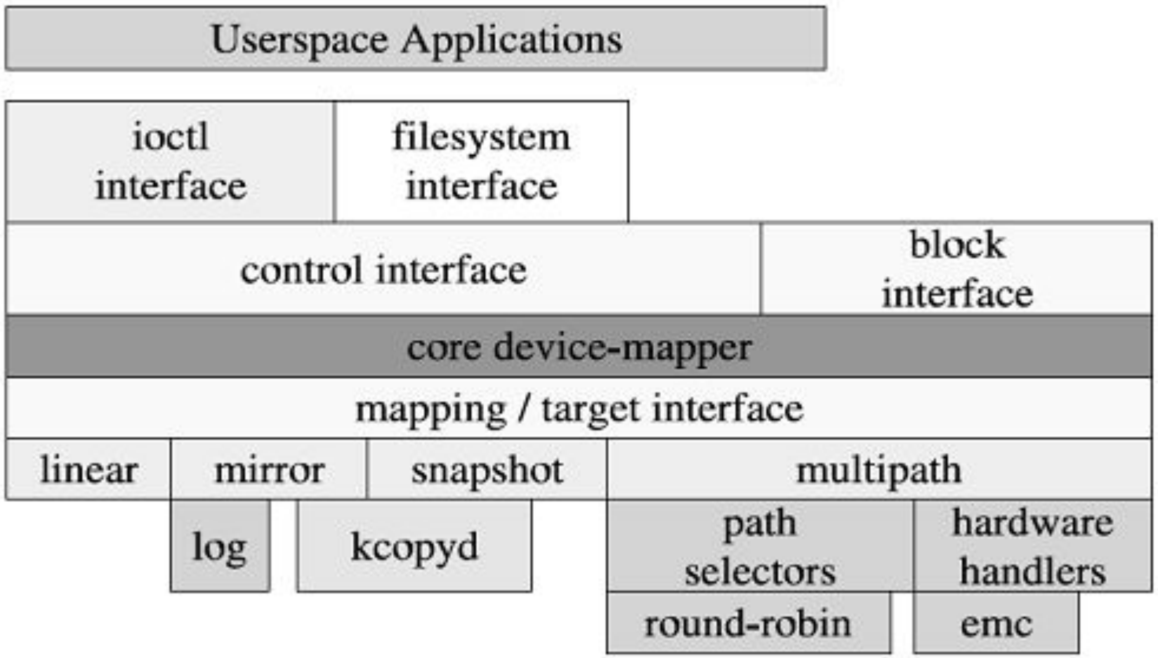
- advanced multi–layered unification filesystem：高级多层统一文件系统
- 用于为Linux文件系统实现“联合挂载”
- aufs是之前的UnionFS的重新实现，2006年由Junjiro Okajima开发；
- Docker最初使用aufs作为容器文件系统层，它目前仍作为存储后端之一来支持；
- aufs的竞争产品是overlayfs，后者自从3.18版本开始被合并到Linux内核；
- docker的分层镜像，除了aufs，docker还支持btrfs, devicemapper和vfs等
  - 在Ubuntu系统下，docker默认Ubuntu的 aufs；而在CentOS7上，用的是devicemapper；

马哥教育
www.magedu.com

Devicemapper

- Device Mapper 是 Linux2.6 内核中支持逻辑卷管理的通用设备映射机制，它为实现用于存储资源管理的块设备驱动提供了一个高度模块化的内核架构

Device Mapper Kernel Architecture



Userspace Applications

ioctl interface

filesystem interface

control interface

block interface

core device-mapper

mapping / target interface

linear

mirror

snapshot

multipath

log

kcopyd

path selectors

hardware handlers

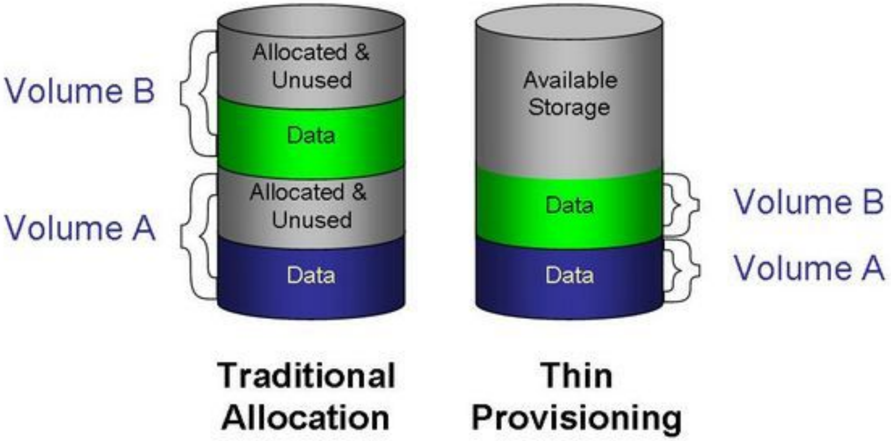
round-robin

emc

Devicemapper

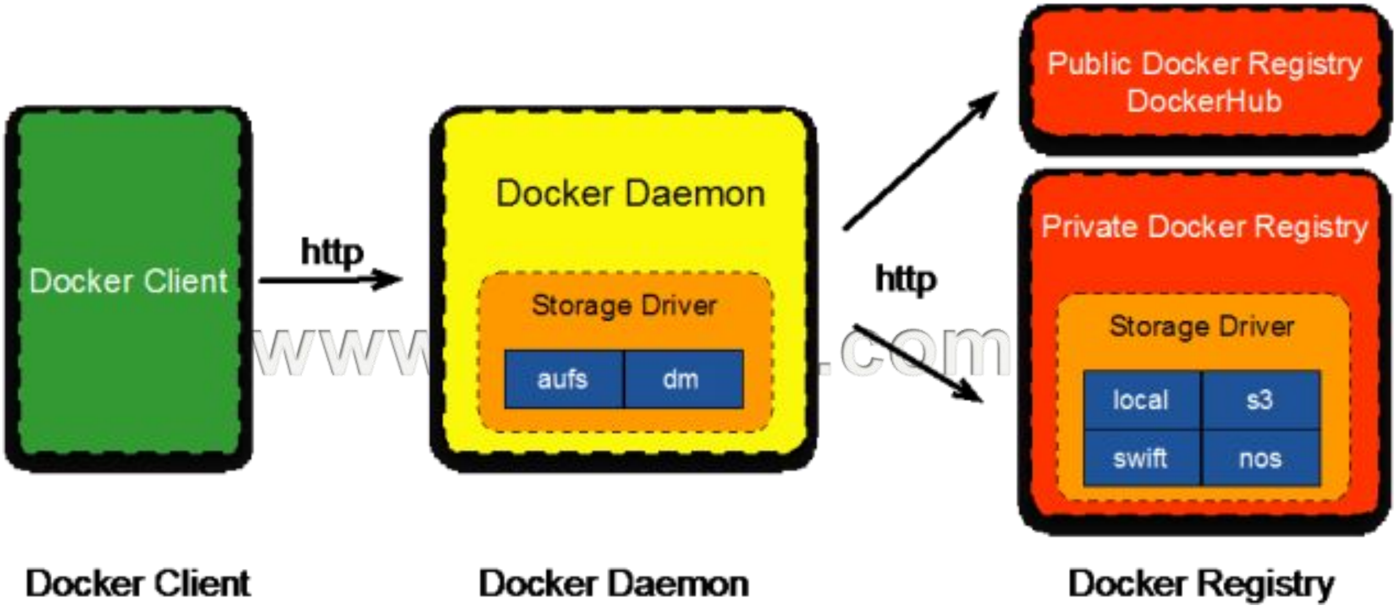
- 在内核中它通过一个一个模块化的 target driver 插件实现对 IO 请求的过滤或者重新定向等工作，当前已经实现的 target driver 插件包括软 raid、软加密、逻辑卷条带、多路径、镜像、快照等
  - 前一页图中 linear、mirror、snapshot、multipath 表示的就是这些 target driver
  - 在这诸多“插件”中，有一种叫Thin Provisioning Snapshot，Docker正是使用了Thin Provisioning的Snapshot的技术实现了类似UnionFS的分层镜像

马哥
www.mag



Docker Registry

- 启动容器时，docker daemon会试图从本地获取相关的镜像；本地镜像不存在时，其将从Registry中下载该镜像并保存到本地；
  - The Registry is a stateless, highly scalable server side application that stores and lets you distribute Docker images.



Docker Registry 分类

- Registry用于保存docker镜像，包括镜像的层次结构和元数据
- 用户可自建Registry，也可使用官方的Docker Hub
- 分类
  - Sponsor Registry：第三方的registry，供客户和Docker社区使用
  - Mirror Registry：第三方的registry，只让客户使用
  - Vendor Registry：由发布Docker镜像的供应商提供的registry
  - Private Registry：通过设有防火墙和额外的安全层的私有实体提供的registry

马哥教育
www.magedu.com

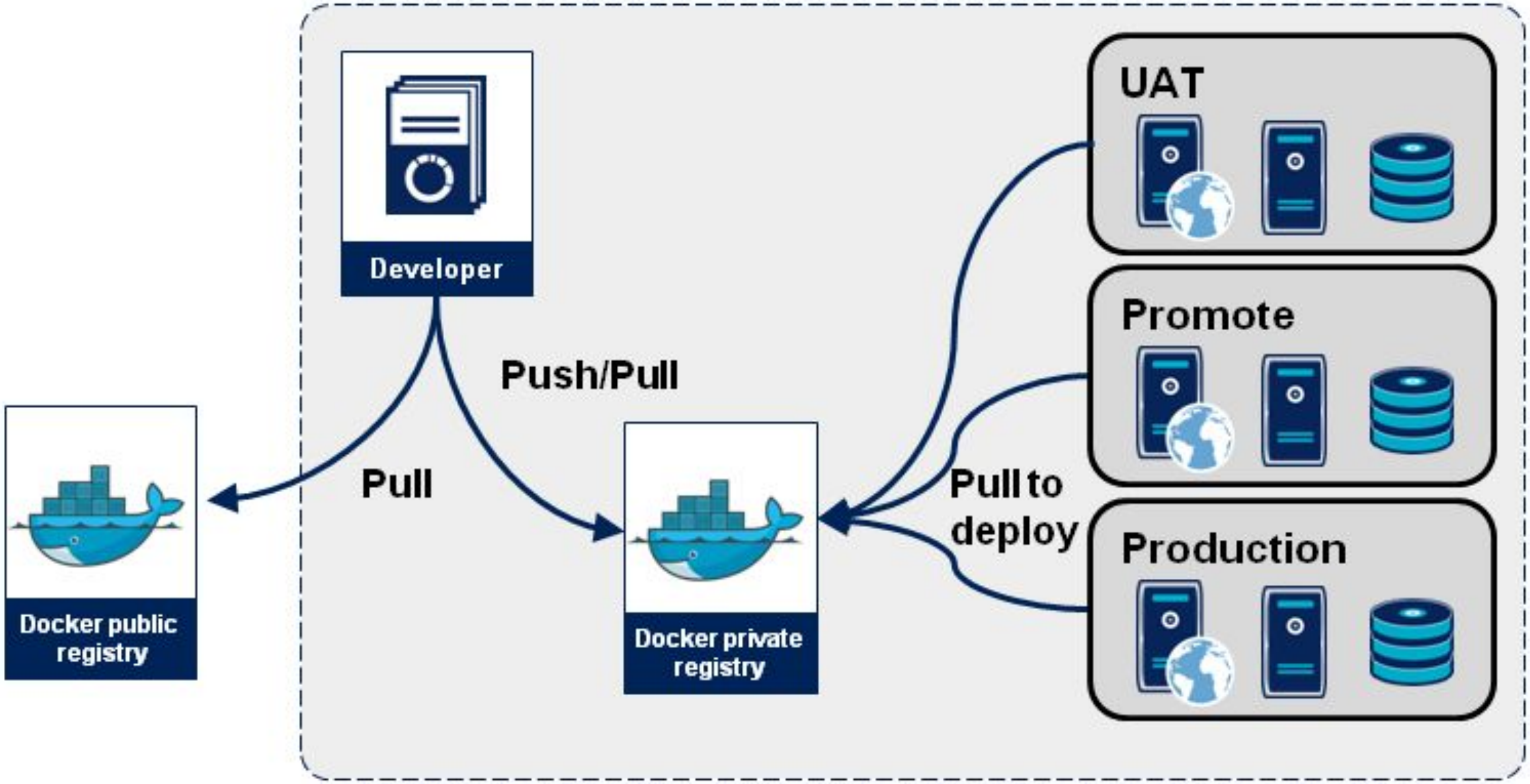
Registry(repository and index)

- Repository
  - 由某特定的docker镜像的所有迭代版本组成的镜像仓库
  - 一个 Registry中可以存在多个Repository
    - Repository可分为“顶层仓库”和“用户仓库”
    - 用户仓库名称格式为“用户名/仓库名”
  - 每个仓库可以包含多个Tag(标签)，每个标签对应一个镜像
- Index
  - 维护用户帐户、镜像的校验以及公共命名空间的信息
  - 相当于为Registry提供了一个完成用户认证等功能的检索接口

马哥教育
www.magedu.com

Docker Registry

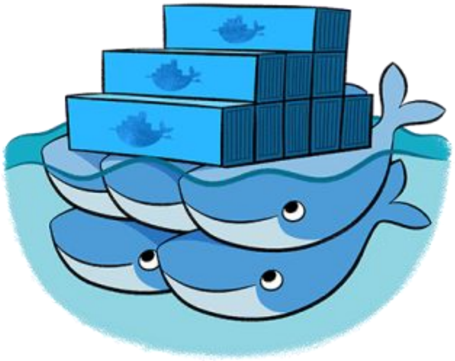
- Docker Registry中的镜像通常由开发人员制作，而后推送至“公共”或“私有”Registry上保存，供其他人员使用，例如“部署”到生产环境；



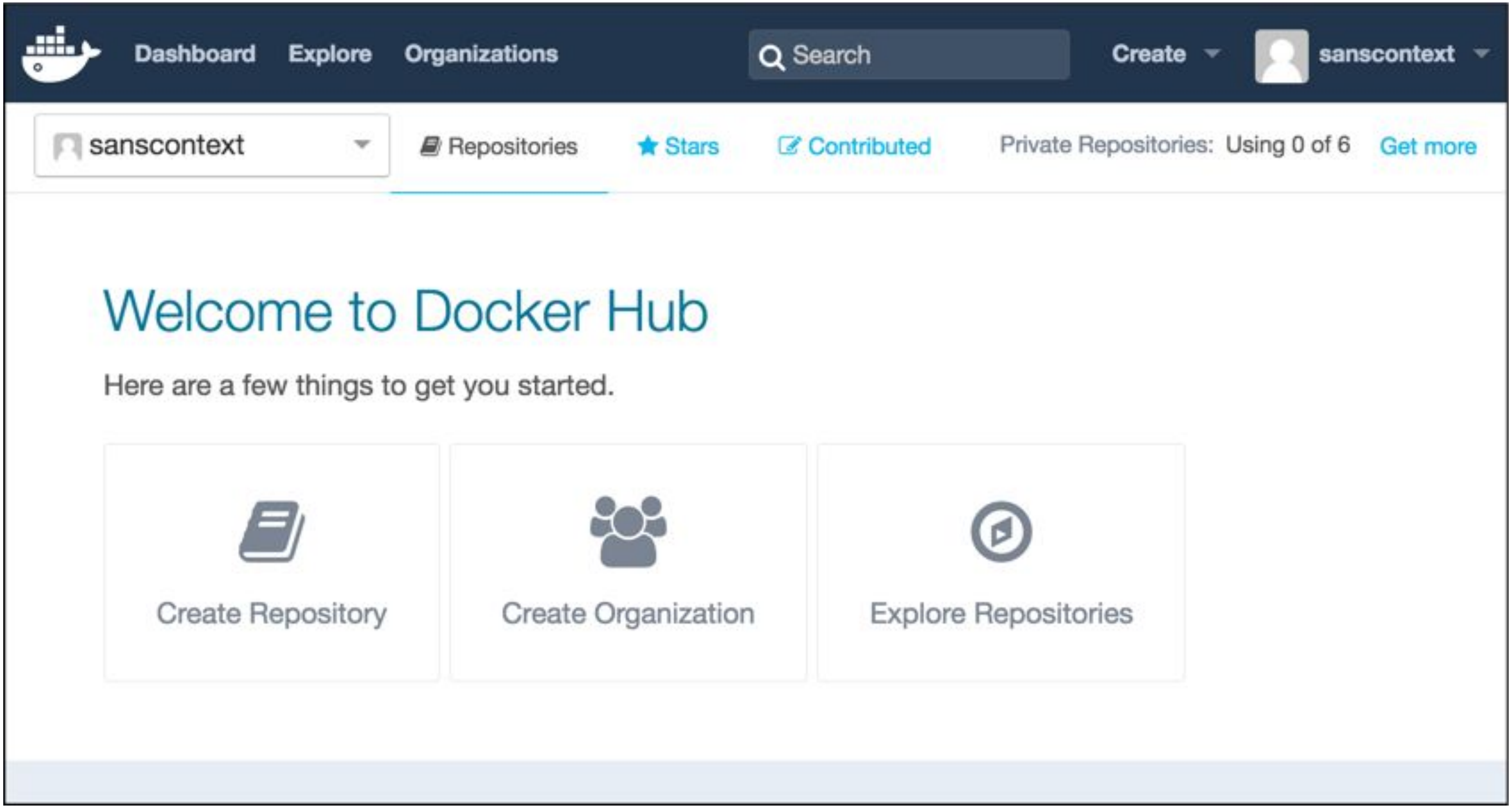
Docker Hub

- Docker Hub is a cloud-based registry service which allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker Cloud so you can deploy images to your hosts.
- It provides a centralized resource for container image discovery, distribution and change management, user and team collaboration, and workflow automation throughout the development pipeline.

马哥教育
www.magedu.com



Docker Hub



Private Repositories: Using 0 of 6 Get more

Welcome to Docker Hub

Here are a few things to get you started.

Docker Hub

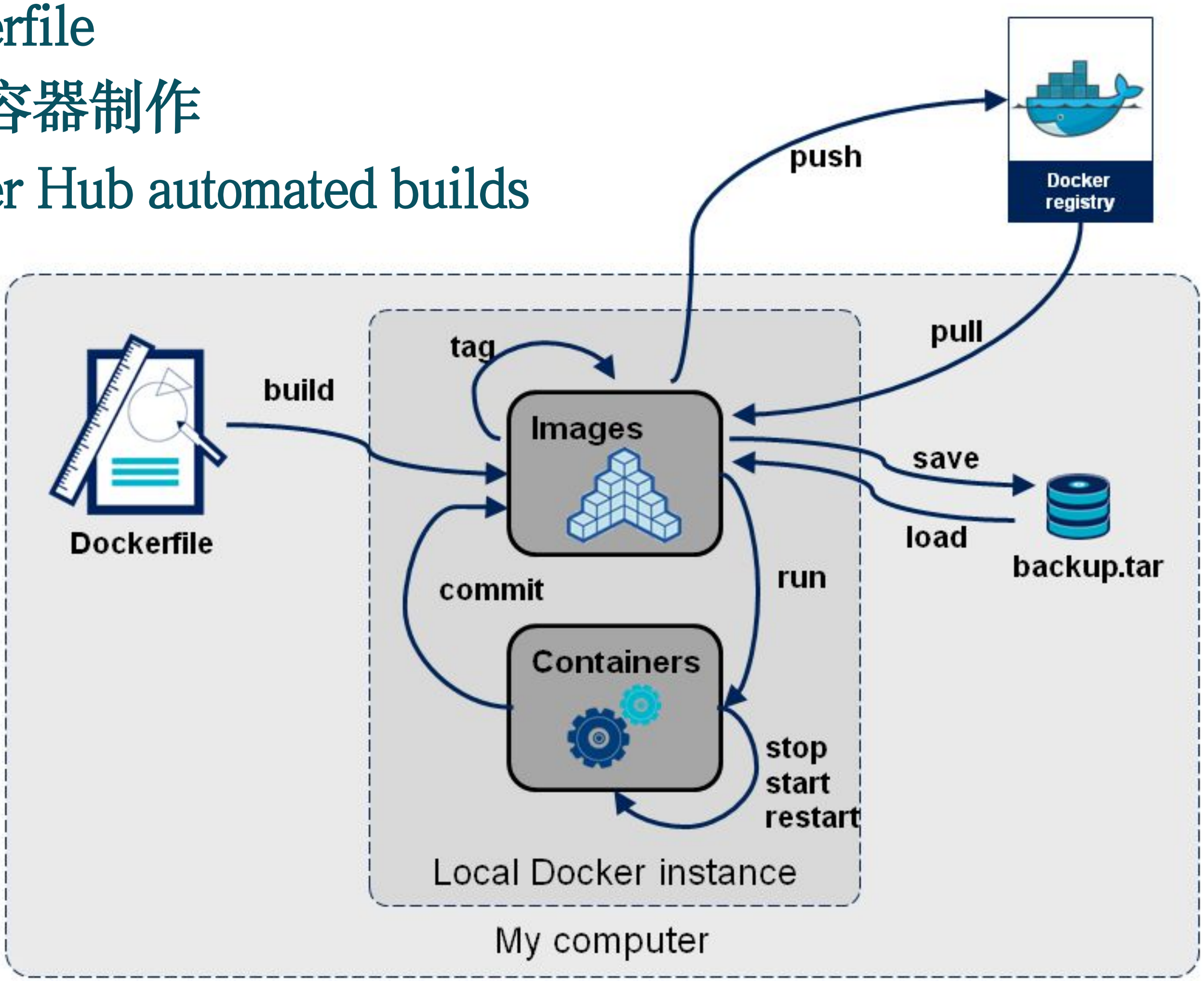
- Docker Hub provides the following major features
  - Image Repositories
    - Find and pull images from community and official libraries, and manage, push to, and pull from private image libraries to which you have access.
  - Automated Builds
    - Automatically create new images when you make changes to a source code repository.
  - Webhooks
    - A feature of Automated Builds, Webhooks let you trigger actions after a successful push to a repository.
  - Organizations
    - Create work groups to manage access to image repositories.
  - GitHub and Bitbucket Integration
    - Add the Hub and your Docker Images to your current workflows.

马哥教育
www.magedu.com

镜像相关的操作

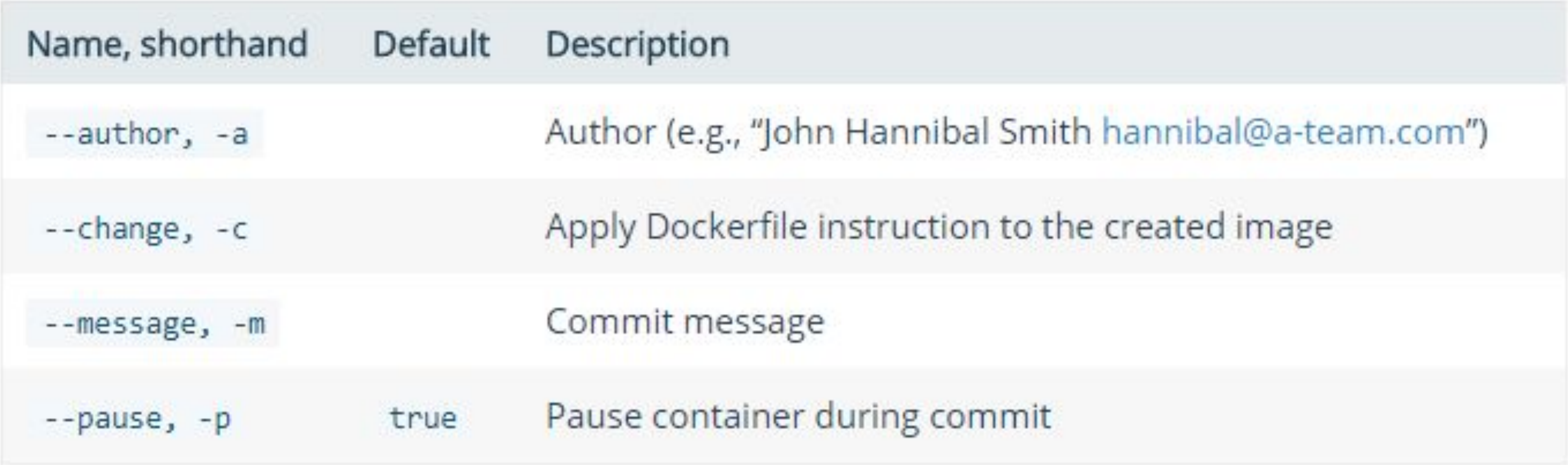
镜像的生成途径

- Dockerfile
- 基于容器制作
- Docker Hub automated builds



基于容器制作镜像

- Create a new image from a container’s changes
- Usage
  - docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]



Name, shorthand	Default	Description
`--author, -a`		Author (e.g., “John Hannibal Smith hannibal@a-team.com”)
`--change, -c`		Apply Dockerfile instruction to the created image
`--message, -m`		Commit message
`--pause, -p`	`true`	Pause container during commit

基于容器制作镜像

- 使用示例
  - 启动容器，执行需要的修改操作
    - ~]# docker run ––rm –it ––name bbox1 busybox
  - 查看修改
    - ~]# docker logs bbox1

```
/ # mkdir /data/httpd/htdocs -pv
created directory: '/data/'
created directory: '/data/httpd/'
created directory: '/data/httpd/htdocs'
/ # echo "Welcome to Busybox." > /data/httpd/htdocs/index.html
```

马哥教育
www.magedu.com

  - 提交镜像
    - ~]# docker commit bbox1 mageedu/busybox/httpd:latest
  - 提交镜像，修改默认运行的命令
    - ~]# docker commit ––change='CMD ["httpd", "–h /data/httpd/htdocs “, “–f”]' –c "EXPOSE 80" bbox1 mageedu/busybox/httpd:v0.1

为镜像设定标签

- docker tag
  - Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  - Syntax
    - docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]
  - 示例
    - 基于ID打标
      - ~]# docker tag 9133dae37bd8 mageedu/busybox/httpd:latest
    - 基于名称和标签打标
      - ~]# docker tag mageedu/busybox/httpd:v0.1 mageedu/busybox/httpd:version0.1
    - 为私有Registry打标
      - ~]# docker tag 9133dae37bd8 myregistry:5000/busybox/httpd:v0.1

马哥教育
www.magedu.com

将镜像push到Docker Hub

- docker push
  - Push images to Docker Cloud or private registry
  - 推送镜像到Docker Hub的前提是于Docker Hub有用户账号，且镜像标签格式为“$DOCKER_USER_ID/IMAGE”
  - 使用docker login命令登录成功后，即可使用docker push命令进行推送
  - 示例
    - ~]# docker push mageedu/busybox/httpd:version0.1

马哥教育
www.magedu.com

镜像导入和导出

- docker save
  - Save one or more images to a tar archive (streamed to STDOUT by default)
  - Usage： docker save [OPTIONS] IMAGE [IMAGE...]
    - ——output, –o： Write to a file, instead of STDOUT
- docker load
  - Load an image from a tar archive or STDIN
  - Usage： docker load [OPTIONS]
    - ——input, –i： Read from tar archive file, instead of STDIN
    - ——quiet, –q： Suppress the load output

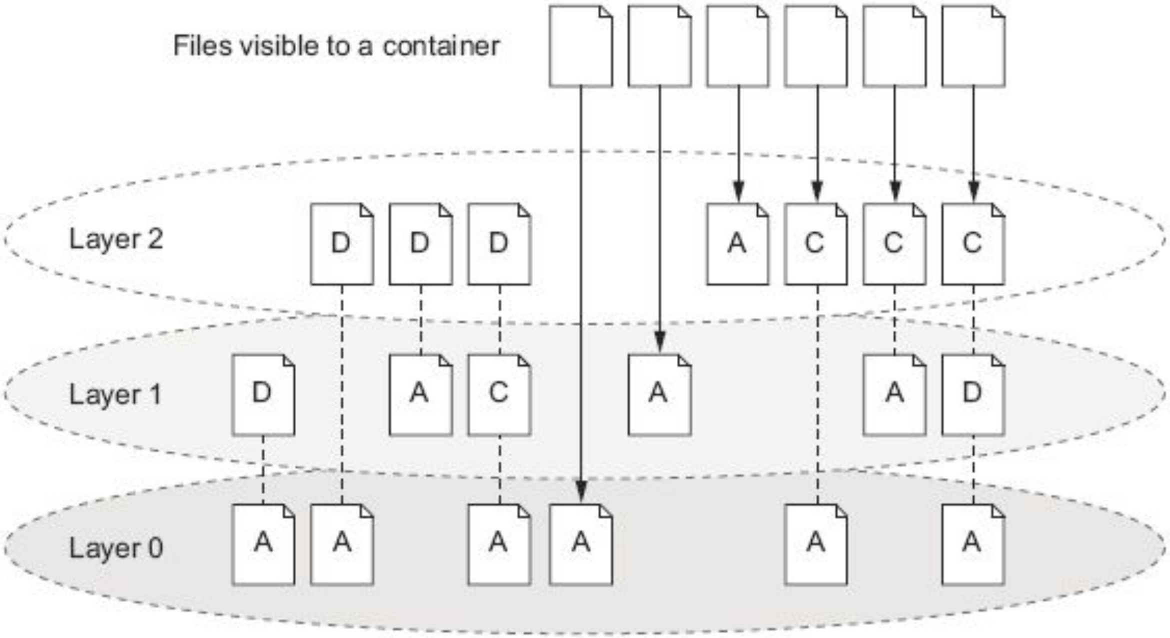
马哥教育
www.magedu.com

Docker Data Volume

主讲：马永亮(马哥)
QQ群:169777636
客服QQ：2813150558, 1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

Why Data Volumes?

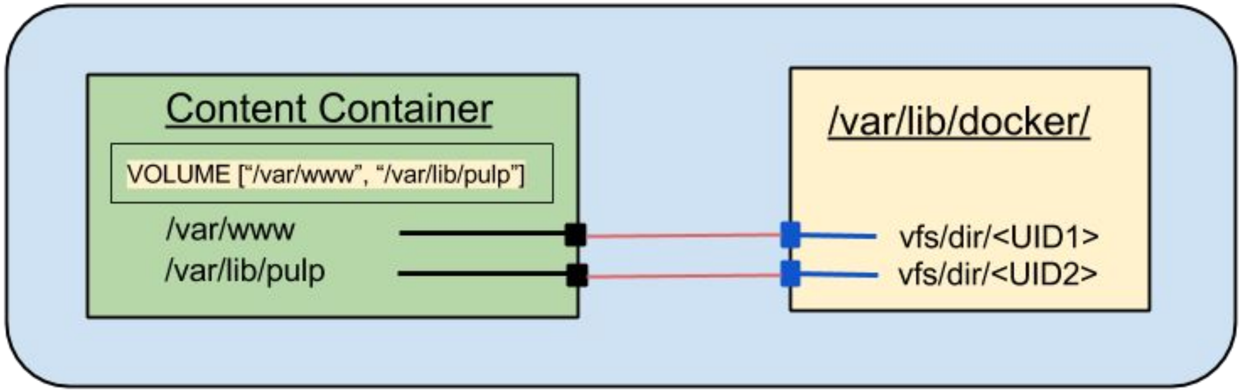
- Docker镜像由多个只读层叠加而成，启动容器时，Docker会加载只读镜像层并在镜像栈顶部添加一个读写层
- 如果运行中的容器修改了现有的一个已经存在的文件，那该文件将会从读写层下面的只读层复制到读写层，该文件的只读版本仍然存在，只是已经被读写层中该文件的副本所隐藏，此即“写时复制(COW)”机制



Why Data Volume?

- 关闭并重启容器，其数据不受影响；但删除Docker容器，则其更改将会全部丢失
- 存在的问题
  - 存储于联合文件系统中，不易于宿主机访问；
  - 容器间数据共享不便
  - 删除容器其数据会丢失
- 解决方案：“卷(volume)”
  - “卷”是容器上的一个或多个“目录”，此类目录可绕过联合文件系统，与宿主机上的某目录“绑定(关联)”

马哥教育



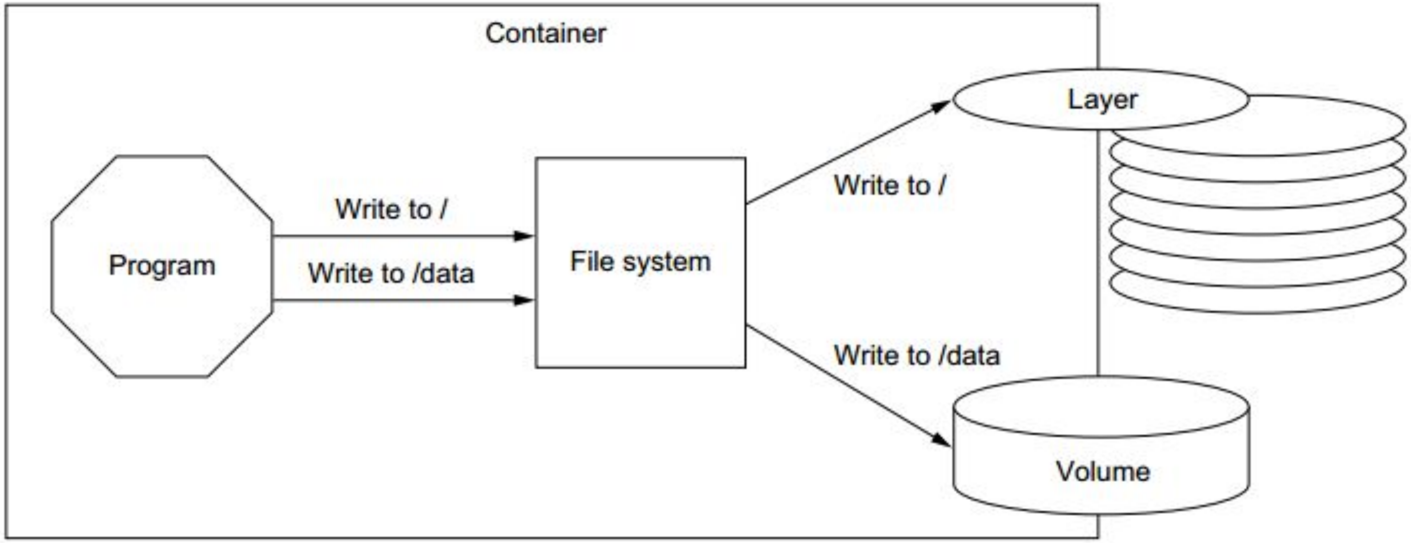
Data volumes

- Data volumes provide several useful features for persistent or shared data
    - Volume于容器初始化之时即会创建，由base image提供的卷中的数据会于此期间完成复制
    - Data volumes can be shared and reused among containers
    - Changes to a data volume are made directly
    - Changes to a data volume will not be included when you update an image
    - Data volumes persist even if the container itself is deleted
- Volume的初衷是独立于容器的生命周期实现数据持久化，因此删除容器之时既不会删除卷，也不会对哪怕未被引用的卷做垃圾回收操作；

马哥教育
www.magedu.com

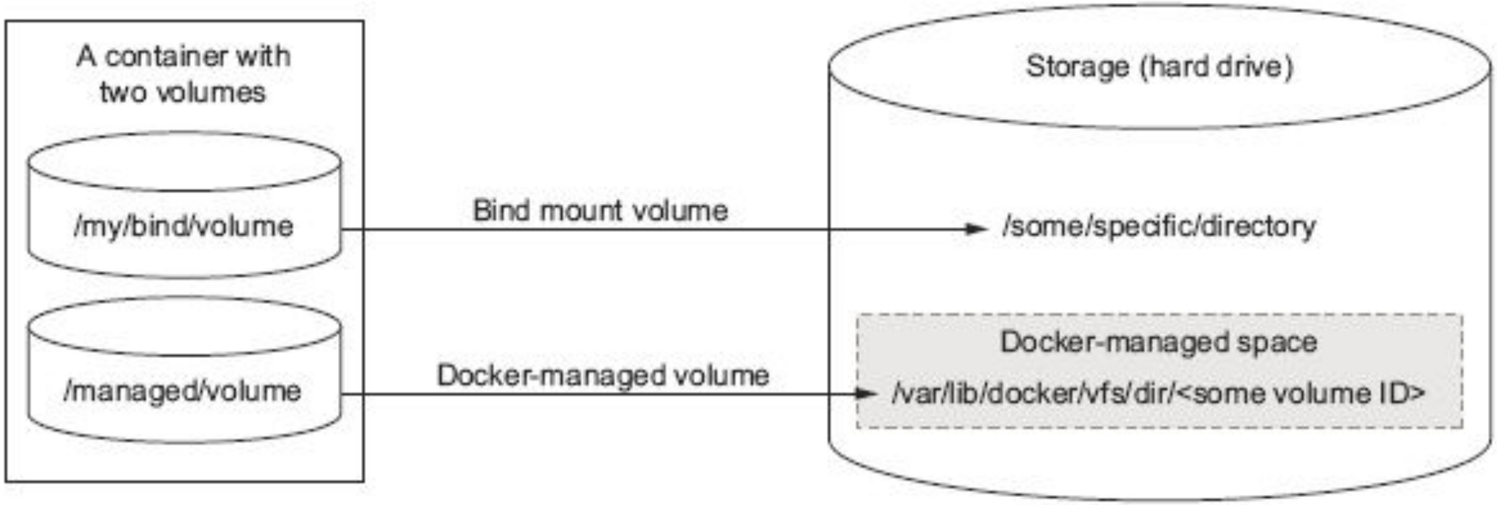
Data volumes

- 卷为docker提供了独立于容器的数据管理机制
  - 可以把"镜像"想像成静态文件，例如"程序"，把卷类比为动态内容，例如"数据"；于是，镜像可以重用，而卷可以共享；
  - 卷实现了"程序(镜像)"和"数据(卷)"分离，以及"程序(镜像)"和"制作镜像的主机"分离，用户制作镜像时无须再考虑镜像运行的容器所在的主机的环境；



Volume types

- Docker有两种类型的卷，每种类型都在容器中存在一个挂载点，但其在宿主机上的位置有所不同；
  - Bind mount volume
    - a volume that points to a user–specified location on the host file system
  - Docker–managed volume
    - the Docker daemon creates managed volumes in a portion of the host’s file system that’s owned by Docker



在容器中使用Volumes

- 为docker run命令使用–v选项即可使用Volume
  - Docker–managed volume
    - ~]# docker run –it –name bbox1 – v /data busybox
    - ~]# docker inspect –f {{.Config.Volumes}}{{.Mounts}} bbox1
      - 查看bbox1容器的卷、卷标识符及挂载的主机目录
  - Bind–mount Volume
    - ~]# docker run –it –v HOSTDIR:VOLUMEDIR ––name bbox2 busybox
    - ~]# docker inspect –f {{.Mounts}} bbox2

马哥教育
www.magedu.com

Sharing volumes

- There are two ways to share volumes between containers
  - 多个容器的卷使用同一个主机目录，例如
    - ~]# docker run – it —name c1 –v /docker/volumes/v1:/data busybox
    - ~]# docker run – it —name c2 –v /docker/volumes/v1:/data busybox
  - 复制使用其它容器的卷，为docker run命令使用—volumes–from选项
    - ~]# docker run –it —name bbox1 –v /docker/volumes/v1:/data busybox
    - ~]# docker run –it —name bbox2 —volumes–from bbox1 busybox

马哥教育
www.magedu.com

删除卷

- 删除容器之时删除相关的卷
  - 为docker rm命令使用-v选项
- 删除指定的卷
  - docker volume rm

马哥教育
www.magedu.com

马哥教育
最专业的Linux培训机构

Docker Networking

主讲：马永亮(马哥)
QQ群:169777636
客服QQ：2813150558, 1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

马哥教育
www.magedu.com

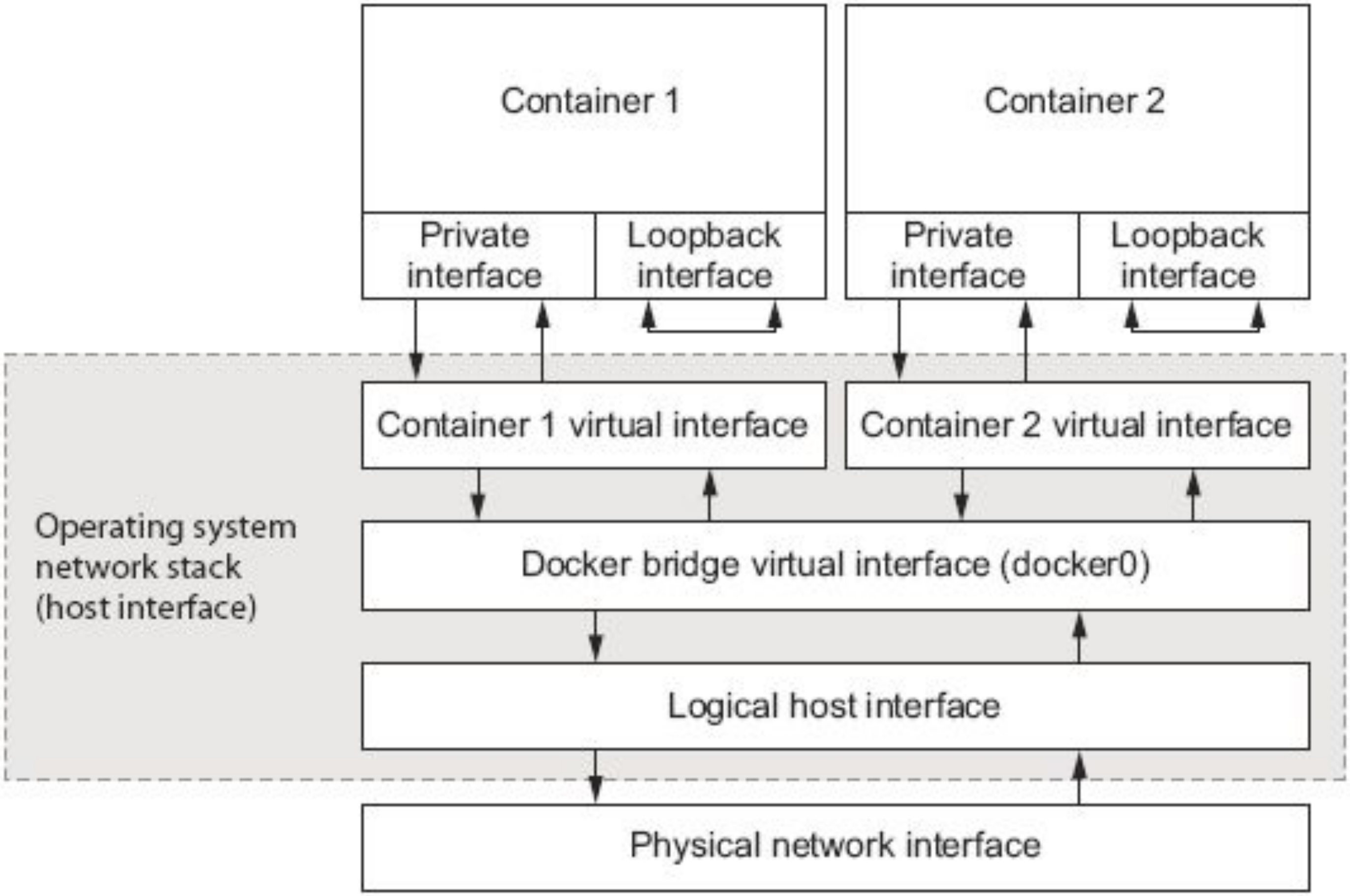
Docker container networking

- Docker is concerned with two types of networking:
  - single–host virtual networks
    - provide container isolation
  - multi–host networks
    - provide an overlay where any container on a participating host can have its own routable IP address from any other container in the network

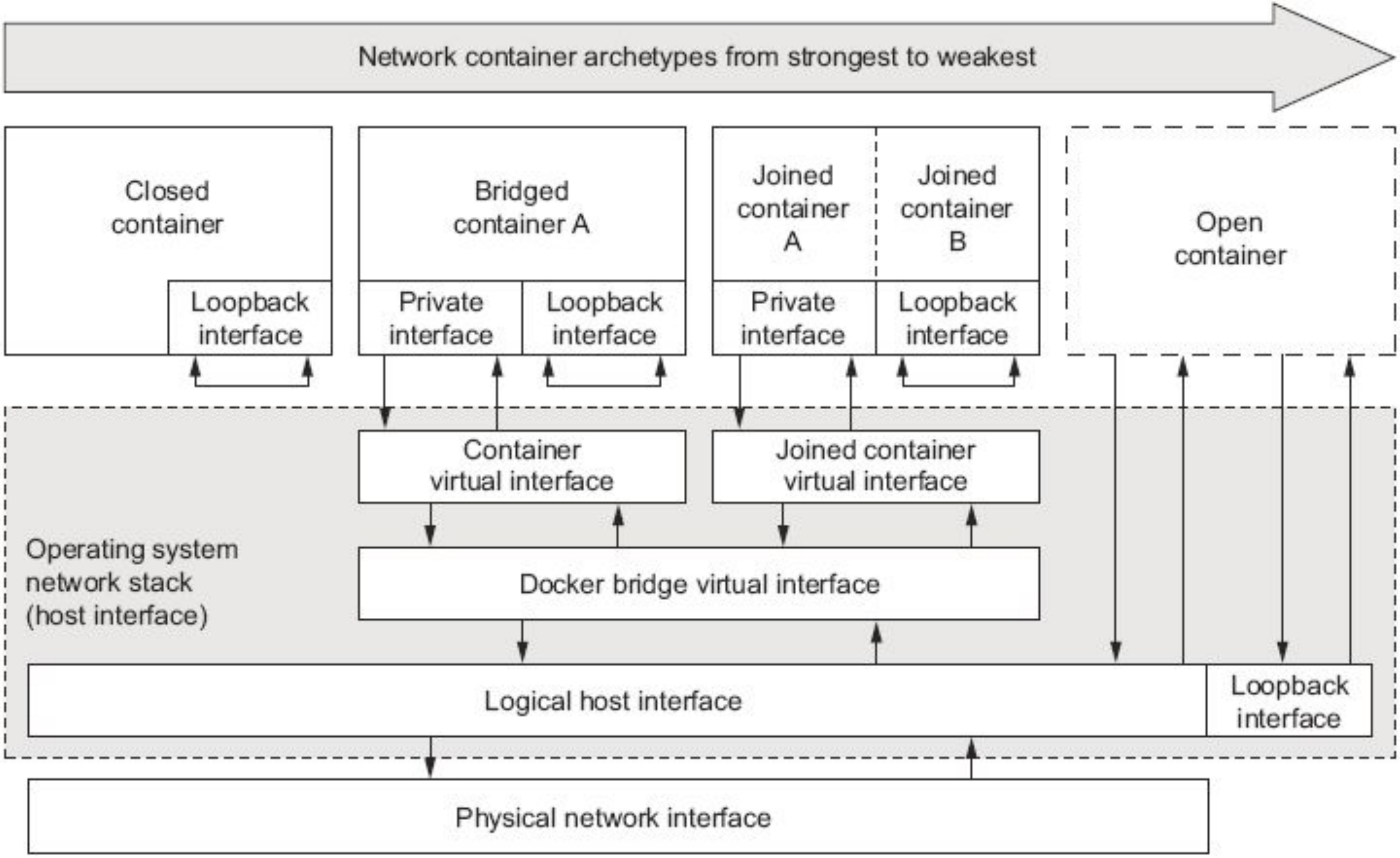
马哥教育
www.magedu.com

The local Docker network topology

- The default local Docker network topology and two attached containers


Container 1
Container 2
Private interface
Loopback interface
Private interface
Loopback interface
Container 1 virtual interface
Container 2 virtual interface
Operating system network stack (host interface)
Docker bridge virtual interface (docker0)
Logical host interface
Physical network interface

Four network container archetypes


Network container archetypes from strongest to weakest
Closed container
Loopback interface
Bridged container A
Private interface
Loopback interface
Joined container A
Joined container B
Private interface
Loopback interface
Open container
Container virtual interface
Joined container virtual interface
Operating system network stack (host interface)
Docker bridge virtual interface
Logical host interface
Loopback interface
Physical network interface

Docker Networks

- Docker安装完成后，会自动创建三个网络，可使用“docker network ls”命令查看
  - bridge
    - represents the docker0 network present in all Docker installations
  - host
  - none
- 创建容器时，可为docker run命令使用—network选项指定要加入的网络

马哥教育
www.magedu.com

Closed containers

- 不参与网络通信，运行于此类容器中的进程仅能访问本地环回接口
- 仅适用于进程无须网络通信的场景中，例如备份、进程诊断及各种离线任务等

```
~]# docker run ––rm ––net none busybox:latest ifconfig –a
```

马哥教育
www.magedu.com

Bridged containers

- 桥接式容器一般拥有两个接口：一个环回接口和一个连接至主机上某桥设备的以太网接口
- docker daemon启动时默认会创建一个名为docker0的网络桥，并且创建的容器为桥接式容器，其以太网接口桥接至docker0
  - --net bridge即为将容器接口添加至docker0桥
- docker0桥为NAT桥，因此，桥接式容器可通过此桥接口访问外部网络，但防火墙规则阻止了一切从外部网络访问桥接式容器的请求
  - ~]# docker run --rm --net bridge busybox:latest ifconfig -a

马哥教育
www.magedu.com

Bridged containers

- 可以为docker run命令使用
  - “––hostname HOSTNAME”选项为容器指定主机名，例如

    ```
    ~]# docker  run  ––rm  ––net bridge  ––hostname bbox.magedu.com
    busybox:latest  nslookup bbox.magedu.com
    ```

  - “––dns DNS_SERVER_IP”选项能够为容器指定所使用的dns服务器地址，例如

    ```
    ~]# docker run  ––rm  ––dns 172.16.0.1  busybox:latest nslookup docker.com
    ```

  - “––add–host HOSTNAME:IP”选项能够为容器指定本地主机名解析项，例如

    ```
    ~]# docker run  ––rm  ––dns 172.16.0.1  ––add–host
    "docker.com:172.16.0.100" busybox:latest nslookup docker.com
    ```

马哥教育
www.magedu.com

Opening inbound communication

- Docker0为NAT桥，因此容器一般获得的是私有网络地址
- 可以把容器想像为宿主机NAT服务背后的主机
- 如果开放容器或其上的服务为外部网络访问，需要在宿主机上为其定义DNAT规则，例如
  - 对宿主机某IP地址的访问全部映射给某容器地址
    - 主机IP　　容器IP
      - –A PREROUTING –d 主机IP –j DNAT ––to–destination 容器IP
  - 对宿主机某IP地址的某端口的访问映射给某容器地址的某端口
    - 主机IP:PORT　　容器IP:PORT
      - –A PREROUTING –d 主机IP –p {tcp|udp} ––dport 主机端口 –j DNAT ––to–destination 容器IP:容器端口
- 为docker run命令使用–p选项即可实现端口映射，无须手动添加规则

马哥教育
www.magedu.com

Opening inbound communication

- –p选项的使用格式
  - –p <containerPort>
    - 将指定的容器端口映射至主机所有地址的一个动态端口
  - –p <hostPort>:<containerPort>
    - 将容器端口<containerPort>映射至指定的主机端口<hostPort>
  - –p <ip>::<containerPort>
    - 将指定的容器端口<containerPort>映射至主机指定<ip>的动态端口
  - –p <ip>:<hostPort>:<containerPort>
    - 将指定的容器端口<containerPort>映射至主机指定<ip>的端口<hostPort>

马哥教育
www.magedu.com

  - "动态端口"指随机端口，具体的映射结果可使用docker port命令查看

Opening inbound communication

- “–P”选项或“––publish–all”将容器的所有计划要暴露端口全部映射至主机端口
- 计划要暴露的端口使用使用––expose选项指定
  - 例如
    - ~]# docker run –d –P ––expose 2222 ––expose 3333 ––name web busybox:latest /bin/httpd –p 2222 –f
  - 查看映射结果
    - ~]# docker port web
- 如果不想使用默认的docker0桥接口，或者需要修改此桥接口的网络属性，可通过为docker daemon命令使用–b、––bip、––fixed–cidr、––default–gateway、––dns以及––mtu等选项进行设定

马哥教育
www.magedu.com

Joined containers

- 联盟式容器是指使用某个已存在容器的网络接口的容器，接口被联盟内的各容器共享使用；因此，联盟式容器彼此间完全无隔离，例如
  - 创建一个监听于2222端口的http服务容器

    -
      ```
      ~]# docker run –d –it ––rm –p 2222 busybox:latest  /bin/httpd –p 2222 –f
      ```
  - 创建一个联盟式容器，并查看其监听的端口

    -
      ```
      ~]# docker run –it ––rm ––net container:web ––name joined busybox:latest netstat –tan
      ```
- 联盟式容器彼此间虽然共享同一个网络名称空间，但其它名称空间如User、Mount等还是隔离的
- 联盟式容器彼此间存在端口冲突的可能性，因此，通常只会在多个容器上的程序需要程序loopback接口互相通信、或对某已存的容器的网络属性进行监控时才使用此种模式的网络模型

马哥教育
www.magedu.com

Open containers

- 开放式容器共享主机网络名称空间的容器，它们对主机的网络名称空间拥有全部的访问权限，包括访问那些关键性服务，这对宿主机安全性有很大潜在威胁
- 为docker run命令使用“—net host”选项即可创建开放式容器，例如：
  - ~]# docker run –it —rm —net host busybox:latest /bin/sh

马哥教育
www.magedu.com

马哥教育
www.magedu.com

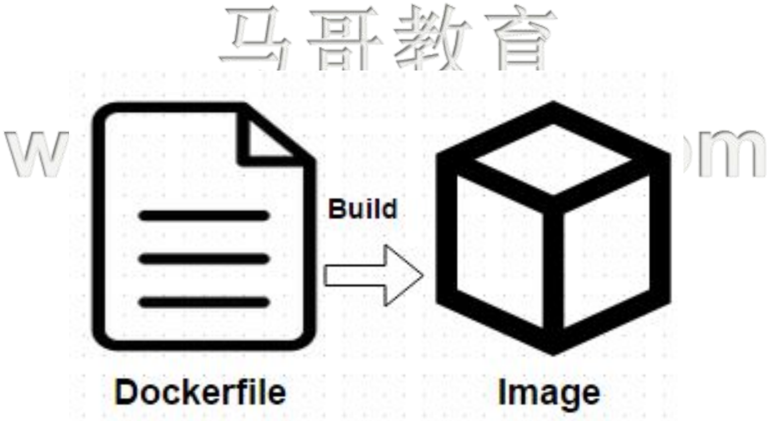
马哥教育
最专业的Linux培训机构

Dockerfile

主讲：马永亮(马哥)
QQ群:169777636
客服QQ：2813150558, 1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

About Dockerfile

- Dockerfile is nothing but the source code for building Docker images
  - Docker can build images automatically by reading the instructions from a Dockerfile
  - A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image
  - Using **docker build** users can create an automated build that executes several command-line instructions in succession



马哥教育
www.magedu.com

Dockerfile Format

- Format
  - # Comment
  - INSTRUCTION arguments
- The instruction is not case-sensitive
  - However, convention is for them to be UPPERCASE to distinguish them from arguments more easily
- Docker runs instructions in a **Dockerfile** in order
- **The first instruction must be \`FROM\`** in order to specify the Base Image from which you are building

马哥教育
www.magedu.com

Environment replacement

- Environment variables (declared with the ENV statement) can also be used in certain instructions as variables to be interpreted by the **Dockerfile**
- Environment variables are notated in the **Dockerfile** either with **$variable_name** or **${variable_name}**
- The **${variable_name}** syntax also supports a few of the standard **bash** modifiers
  - ${variable:–word} indicates that if variable is set then the result will be that value. If variable is not set then word will be the result.
  - ${variable:+word} indicates that if variable is set then word will be the result, otherwise the result is the empty string.

马哥教育
www.magedu.com

.dockerignore file

- Before the docker CLI sends the context to the docker daemon, it looks for a file named **.dockerignore** in the root directory of the context
- If this file exists, the CLI modifies the context to exclude files and directories that match patterns in it
- The CLI interprets the **.dockerignore** file as a newline-separated list of patterns similar to the file globs of Unix shells

马哥教育
www.magedu.com

Dockerfile Instructions

- FROM
  - FROM指令是最重的一个且必须为Dockerfile文件开篇的第一个非注释行，用于为映像文件构建过程指定基准镜像，后续的指令运行于此基准镜像所提供的运行环境
  - 实践中，基准镜像可以是任何可用镜像文件，默认情况下，docker build会在docker主机上查找指定的镜像文件，在其不存在时，则会从Docker Hub Registry上拉取所需的镜像文件
    - 如果找不到指定的镜像文件，docker build会返回一个错误信息
  - Syntax
    - FROM <image>[:<tag>] 或
    - FROM <image>@<digest>
      - <image>：指定作为base image的名称；
      - <tag>：base image的标签，为可选项，省略时默认为latest；

马哥教育
www.magedu.com

Dockerfile Instructions

- MAINTANIER
  - 用于让镜像制作者提供本人的详细信息
  - Dockerfile并不限制MAINTAINER指令可在出现的位置，但推荐将其放置于FROM指令之后
  - Syntax
    - MAINTAINER <authtor's detail>
      - <author's detail>可是任何文本信息，但约定俗成地使用作者名称及邮件地址

马哥教育
www.magedu.com

Dockerfile Instructions

- COPY
  - 用于从Docker主机复制文件至创建的新映像文件
  - Syntax
    - COPY <src> ... <dest> 或
    - COPY ["<src>",... "<dest>"]
      - <src>：要复制的源文件或目录，支持使用通配符
      - <dest>：目标路径，即正在创建的image的文件系统路径；建议为<dest>使用绝对路径，否则，COPY指定则以WORKDIR为其起始路径；
    - 注意：在路径中有空白字符时，通常使用第二种格式
  - 文件复制准则
    - <src>必须是build上下文中的路径，不能是其父目录中的文件
    - 如果<src>是目录，则其内部文件或子目录会被递归复制，但<src>目录自身不会被复制
    - 如果指定了多个<src>，或在<src>中使用了通配符，则<dest>必须是一个目录，且必须以/结尾
    - 如果<dest>事先不存在，它将会被自动创建，这包括其父目录路径

马哥教育
www.magedu.com

Dockerfile Instructions

- ADD
  - ADD指令类似于COPY指令，ADD支持使用TAR文件和URL路径
  - Syntax
    - ADD <src> ... <dest> 或
    - ADD ["<src>",... "<dest>"]
  - 操作准则
    - 同COPY指令
    - 如果<src>为URL且<dest>不以/结尾，则<src>指定的文件将被下载并直接被创建为<dest>；如果<dest>以/结尾，则文件名URL指定的文件将被直接下载并保存为<dest>/<filename>
    - 如果<src>是一个本地系统上的压缩格式的tar文件，它将被展开为一个目录，其行为类似于“tar –x”命令；然而，通过URL获取到的tar文件将不会自动展开；
    - 如果<src>有多个，或其间接或直接使用了通配符，则<dest>必须是一个以/结尾的目录路径；如果<dest>不以/结尾，则其被视作一个普通文件，<src>的内容将被直接写入到<dest>；

马哥教育
www.magedu.com

Dockerfile Instructions

- WORKDIR
  - 用于为Dockerfile中所有的RUN、CMD、ENTRYPOINT、COPY和ADD指定设定工作目录
  - Syntax
    - WORKDIR <dirpath>
      - 在Dockerfile文件中，WORKDIR指令可出现多次，其路径也可以为相对路径，不过，其是相对此前一个WORKDIR指令指定的路径
      - 另外，WORKDIR也可调用由ENV指定定义的变量
    - 例如
      - WORKDIR /var/log
      - WORKDIR $STATEPATH

马哥教育
www.magedu.com

Dockerfile Instructions

- VOLUME
  - 用于在image中创建一个挂载点目录，以挂载Docker host上的卷或其它容器上的卷
  - Syntax
    - VOLUME <mountpoint> 或
    - VOLUME ["<mountpoint>"]
  - 如果挂载点目录路径下此前在文件存在，docker run命令会在卷挂载完成后将此前的所有文件复制到新挂载的卷中

马哥教育
www.magedu.com

Dockerfile Instructions

- EXPOSE
  - 用于为容器打开指定要监听的端口以实现与外部通信
  - Syntax
    - EXPOSE <port>[/<protocol>] [<port>[/<protocol>] ...]
      - <protocol>用于指定传输层协议，可为tcp或udp二者之一，默认为TCP协议
  - EXPOSE指令可一次指定多个端口，例如
    - EXPOSE 11211/udp 11211/tcp

马哥教育
www.magedu.com

Dockerfile Instructions

- ENV
  - 用于为镜像定义所需的环境变量，并可被Dockerfile文件中位于其后的其它指令（如ENV、ADD、COPY等）所调用
  - 调用格式为$variable_name或${variable_name}
  - Syntax
    - ENV <key> <value> 或
    - ENV <key>=<value> ...
  - 第一种格式中，<key>之后的所有内容均会被视作其<value>的组成部分，因此，一次只能设置一个变量；
  - 第二种格式可用一次设置多个变量，每个变量为一个"<key>=<value>"的键值对，如果<value>中包含空格，可以以反斜线(\)进行转义，也可通过对<value>加引号进行标识；另外，反斜线也可用于续行；
  - 定义多个变量时，建议使用第二种方式，以便在同一层中完成所有功能

马哥教育
www.magedu.com

Dockerfile Instructions

- RUN
  - 用于指定docker build过程中运行的程序，其可以是任何命令
  - Syntax
    - RUN <command> 或
    - RUN ["<executable>", "<param1>", "<param2>"]
  - 第一种格式中，<command>通常是一个shell命令，且以“/bin/sh –c”来运行它，这意味着此进程在容器中的PID不为1，不能接收Unix信号，因此，当使用docker stop <container>命令停止容器时，此进程接收不到SIGTERM信号；
  - 第二种语法格式中的参数是一个JSON格式的数组，其中<executable>为要运行的命令，后面的<paramN>为传递给命令的选项或参数；然而，此种格式指定的命令不会以“/bin/sh –c”来发起，因此常见的shell操作如变量替换以及通配符(?,*等)替换将不会进行；不过，如果要运行的命令依赖于此shell特性的话，可以将其替换为类似下面的格式。
    - RUN ["/bin/bash", "–c", "<executable>", "<param1>"]

马哥教育
www.magedu.com

Dockerfile Instructions

- CMD
  - 类似于RUN指令，CMD指令也可用于运行任何命令或应用程序，不过，二者的运行时间点不同
    - RUN指令运行于映像文件构建过程中，而CMD指令运行于基于Dockerfile构建出的新映像文件启动一个容器时
    - CMD指令的首要目的在于为启动的容器指定默认要运行的程序，且其运行结束后，容器也将终止；不过，CMD指定的命令其可以被docker run的命令行选项所覆盖
    - 在Dockerfile中可以存在多个CMD指令，但仅最后一个会生效
  - Syntax
    - CMD <command> 或
    - CMD [“<executable>”, “<param1>”, “<param2>”] 或
    - CMD ["<param1>","<param2>"]
  - 前两种语法格式的意义同RUN
  - 第三种则用于为ENTRYPOINT指令提供默认参数

马哥教育
www.magedu.com

Dockerfile Instructions

- ENTRYPOINT
  - 类似CMD指令的功能，用于为容器指定默认运行程序，从而使得容器像是一个单独的可执行程序
  - 与CMD不同的是，由ENTRYPOINT启动的程序不会被docker run命令行指定的参数所覆盖，而且，这些命令行参数会被当作参数传递给ENTRYPOINT指定指定的程序
    - 不过，docker run命令的—entrypoint选项的参数可覆盖ENTRYPOINT指令指定的程序
  - Syntax
    - ENTRYPOINT <command>
    - ENTRYPOINT ["<executable>", "<param1>", "<param2>"]
  - docker run命令传入的命令参数会覆盖CMD指令的内容并且附加到ENTRYPOINT命令最后做为其参数使用
  - Dockerfile文件中也可以存在多个ENTRYPOINT指令，但仅有最后一个会生效

马哥教育
www.magedu.com

Dockerfile Instructions

- USER
  - 用于指定运行image时的或运行Dockerfile中任何RUN、CMD或ENTRYPOINT指令指定的程序时的用户名或UID
  - 默认情况下，container的运行身份为root用户
  - Syntax
    - USER <UID>|<UserName>
    - 需要注意的是，<UID>可以为任意数字，但实践中其必须为/etc/passwd中某用户的有效UID，否则，docker run命令将运行失败

马哥教育
www.magedu.com

Dockerfile Instructions

- ONBUILD
  - 用于在Dockerfile中定义一个触发器
  - Dockerfile用于build映像文件，此映像文件亦可作为base image被另一个Dockerfile用作FROM指令的参数，并以之构建新的映像文件
  - 在后面的这个Dockerfile中的FROM指令在build过程中被执行时，将会“触发”创建其base image的Dockerfile文件中的ONBUILD指令定义的触发器
  - Syntax
    - ONBUILD <INSTRUCTION>
  - 尽管任何指令都可注册成为触发器指令，但ONBUILD不能自我嵌套，且不会触发FROM和MAINTAINER指令
  - 使用包含ONBUILD指令的Dockerfile构建的镜像应该使用特殊的标签，例如ruby:2.0-onbuild
  - 在ONBUILD指令中使用ADD或COPY指令应该格外小心，因为新构建过程的上下文在缺少指定的源文件时会失败

马哥教育
www.magedu.com

马哥教育
www.magedu.com

马哥教育
最专业的Linux培训机构

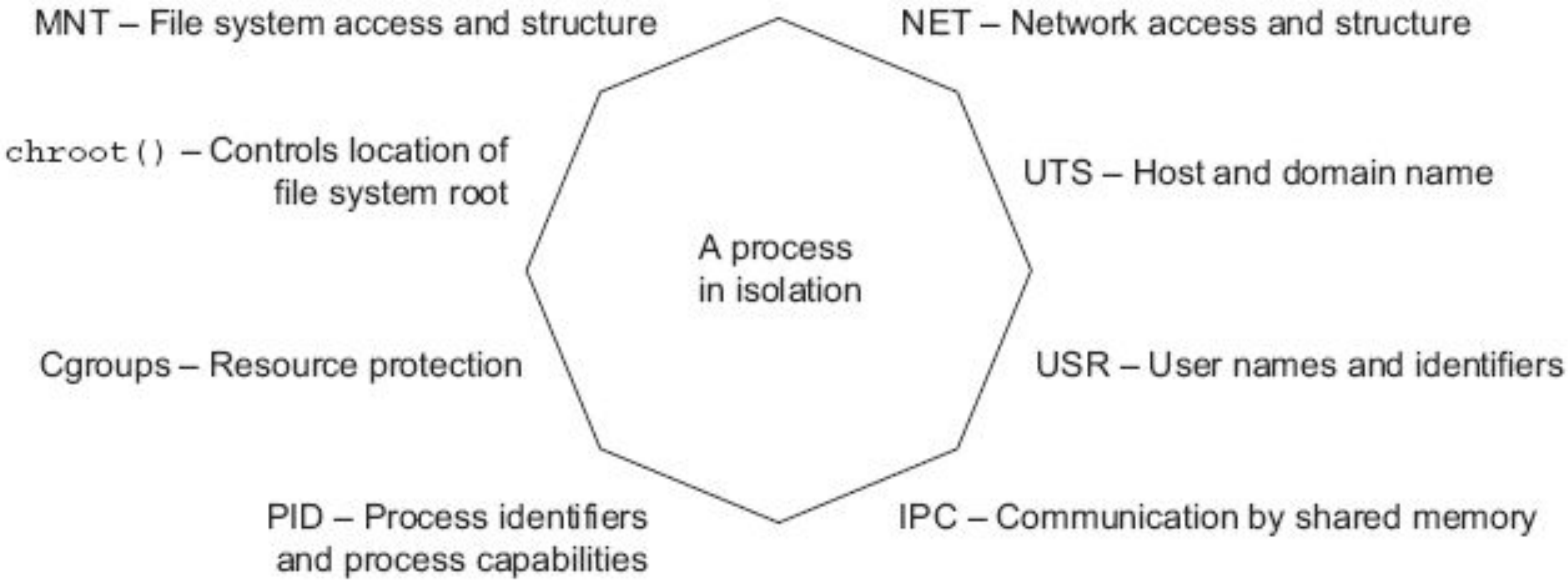
Docker资源配置

主讲：马永亮(马哥)
QQ群:169777636
客服QQ：2813150558, 1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

magedu.com
专注于Linux培训

Resource allowances

Eight–sided containers



MNT – File system access and structure

NET – Network access and structure

`chroot()` – Controls location of file system root

UTS – Host and domain name

A process in isolation

Cgroups – Resource protection

USR – User names and identifiers

PID – Process identifiers and process capabilities

IPC – Communication by shared memory

www.magedu.com

Resource allowances

- Docker provides three flags on the docker run and docker create commands for managing three different types of resource allowances that you can set on a container
  - memory
    - –m or ––memory
  - CPU
    - ––cpu–shares
    - ––cpuset–cpus
  - devices
    - ––device

马哥教育
www.magedu.com

Shared memory

- Docker creates a unique IPC namespace for each container by default
  - IPC
    - —-ipc

马哥教育
www.magedu.com

Running a container with full privileges

- In those cases when you need to run a system administration task inside a container, you can grant that container privileged access to your computer
- Privileged containers maintain their file system and network isolation but have full access to shared memory and devices and possess full system capabilities
  - −−privileged

马哥教育
www.magedu.com

马哥教育
www.magedu.com

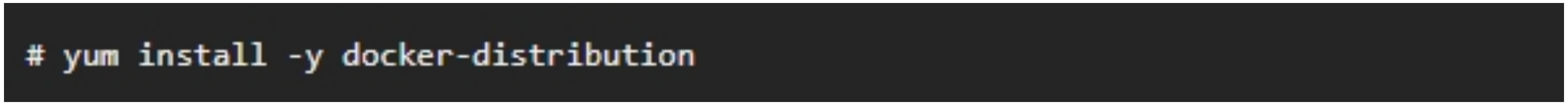
马哥教育
最专业的Linux培训机构

Private Registry

主讲：马永亮(马哥)
QQ群:169777636
客服QQ：2813150558, 1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

Creating a private Docker registry

- To create a private Docker registry you can use the **docker–distribution** service
  - You can install the docker–distribution package in CentOS 7 and enable and start the service as follows:



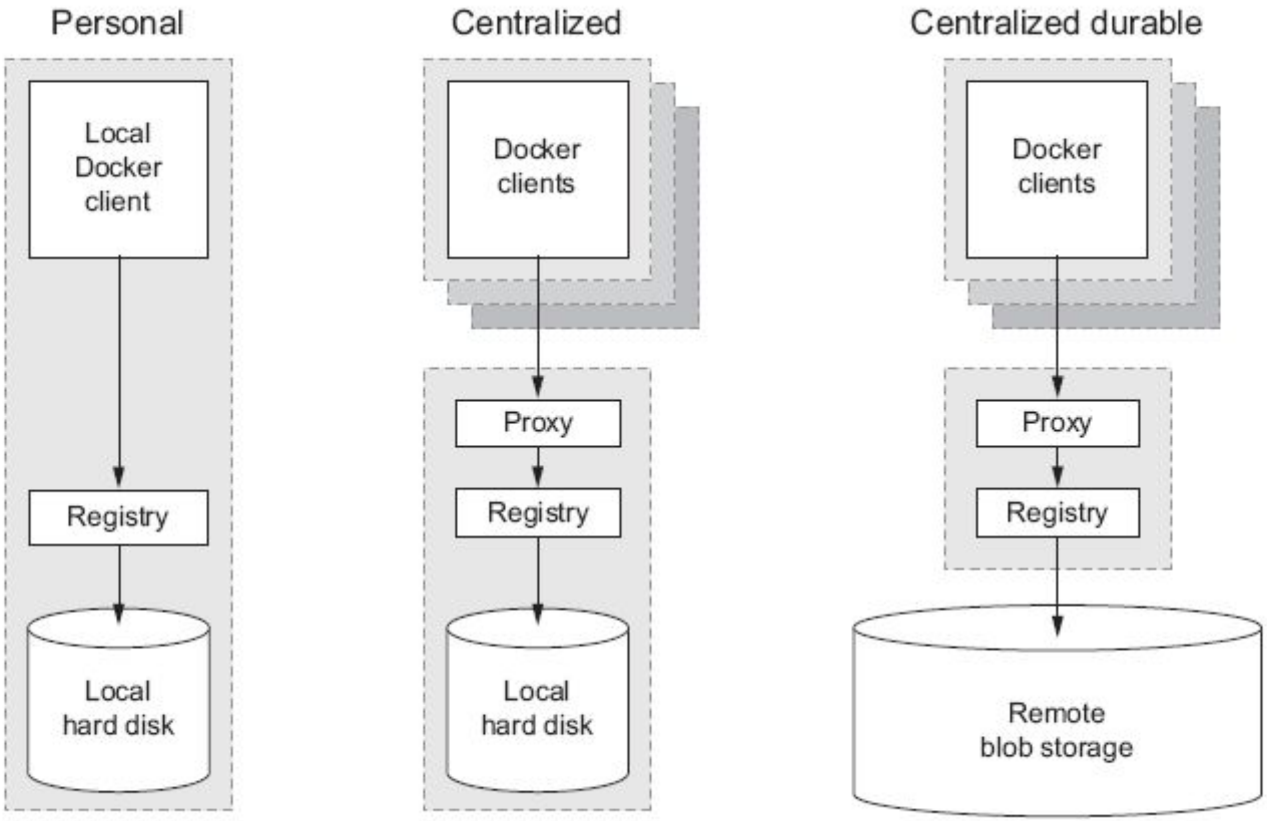
```
# yum install -y docker-distribution
```

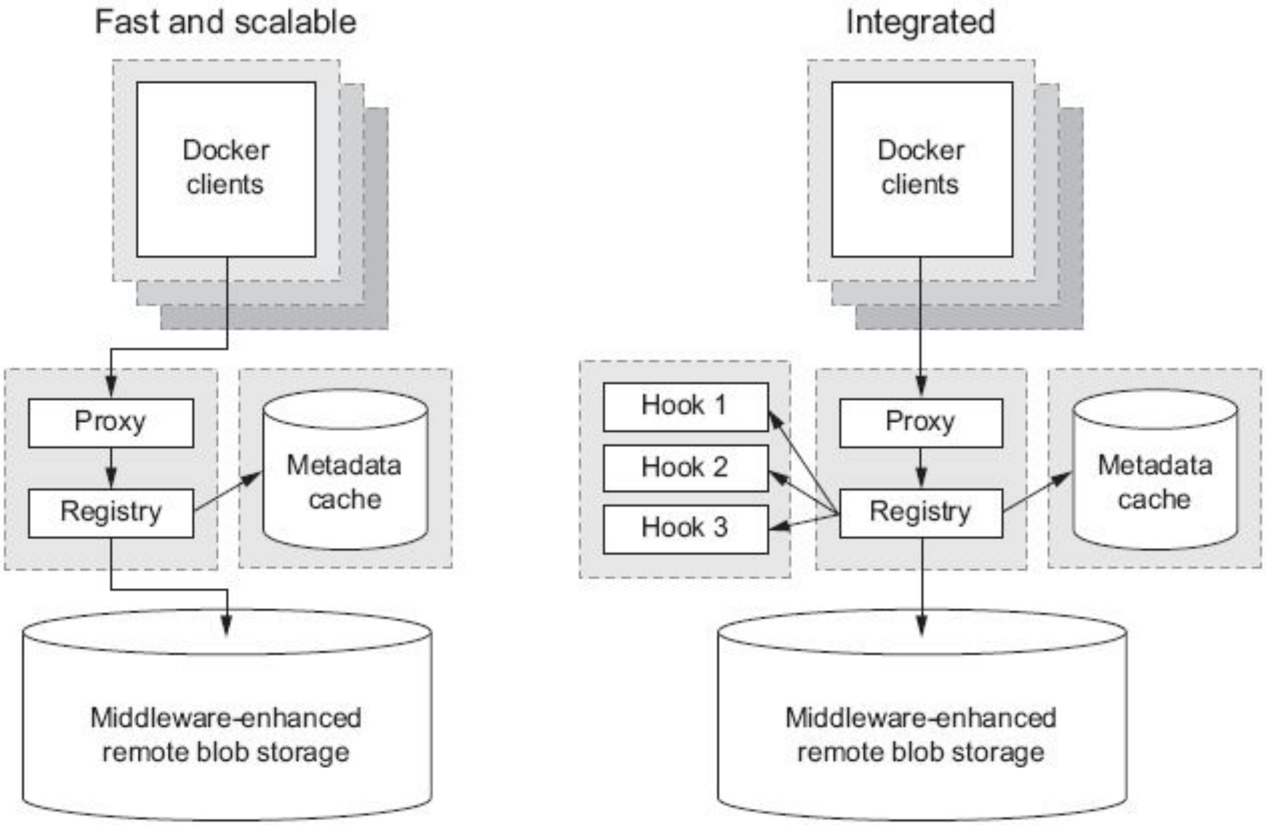
马哥教育
www.magedu.com

马哥教育
www.magedu.com

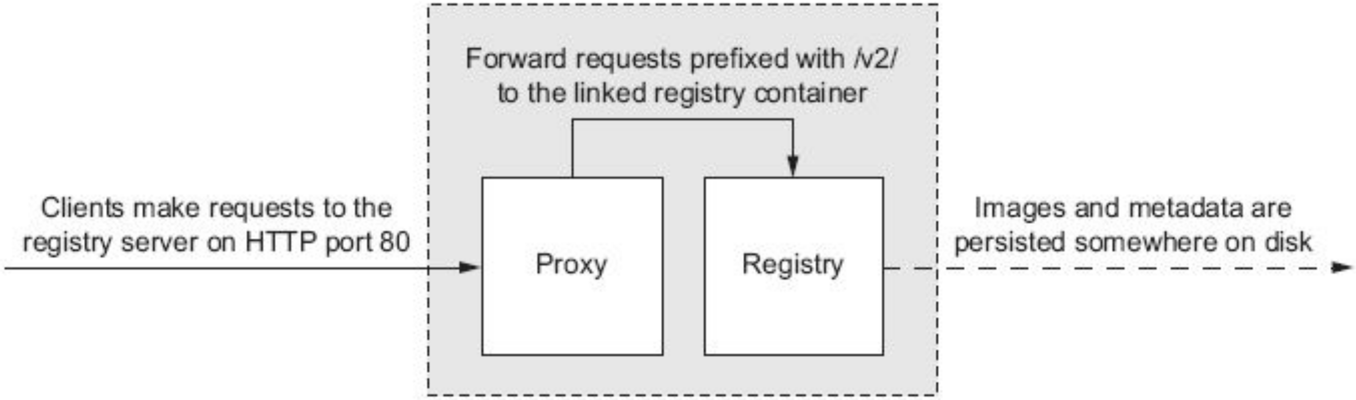
马哥教育
www.magedu.com

Registry configurations

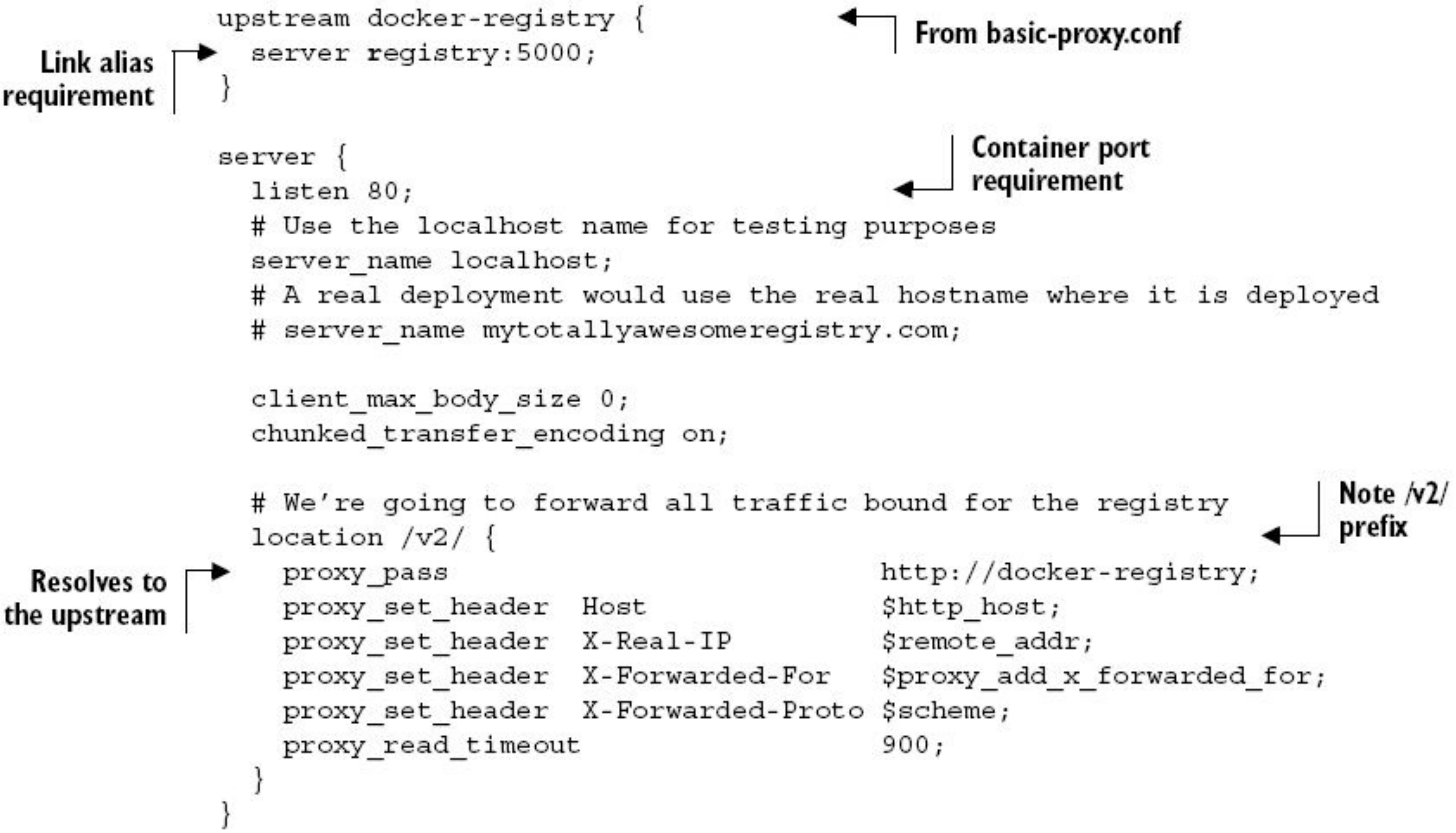



Fast and scalable
Docker clients
Proxy
Registry
Metadata cache
Middleware-enhanced remote blob storage
Integrated
Docker clients
Hook 1
Hook 2
Hook 3
Proxy
Registry
Metadata cache
Middleware-enhanced remote blob storage

Creating a reverse proxy


Forward requests prefixed with /v2/
to the linked registry container
Clients make requests to the
registry server on HTTP port 80
Proxy
Registry
Images and metadata are
persisted somewhere on disk

马哥教育
www.magedu.com



```
upstream docker-registry {
  server registry:5000;
}

server {
  listen 80;
  # Use the localhost name for testing purposes
  server_name localhost;
  # A real deployment would use the real hostname where it is deployed
  # server_name mytotallyawesomeregistry.com;

  client_max_body_size 0;
  chunked_transfer_encoding on;

  # We're going to forward all traffic bound for the registry
  location /v2/ {
    proxy_pass                                http://docker-registry;
    proxy_set_header  Host                    $http_host;
    proxy_set_header  X-Real-IP               $remote_addr;
    proxy_set_header  X-Forwarded-For         $proxy_add_x_forwarded_for;
    proxy_set_header  X-Forwarded-Proto       $scheme;
    proxy_read_timeout                        900;
  }
}
```

From basic-proxy.conf

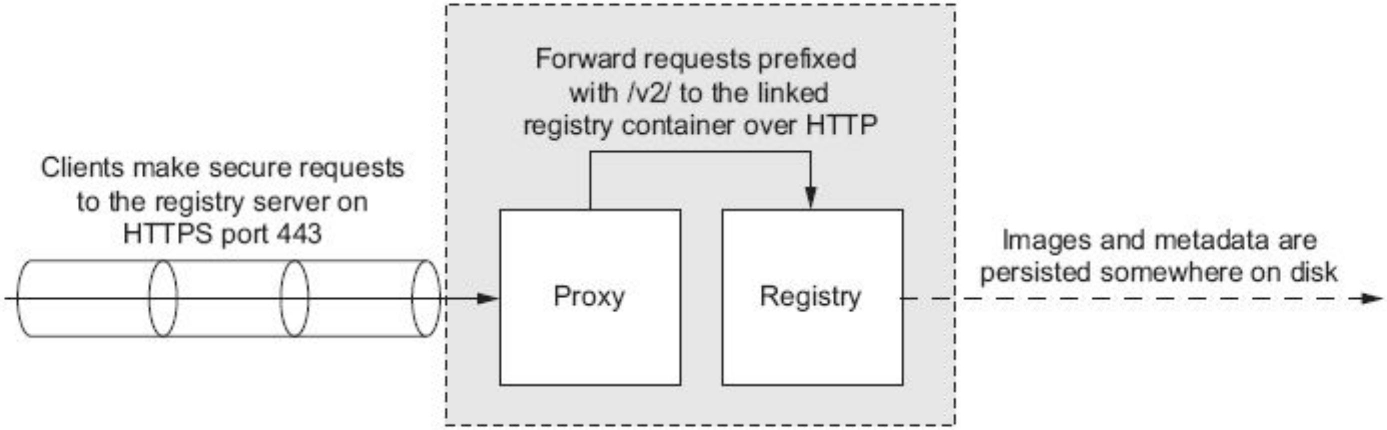
Link alias requirement

Container port requirement

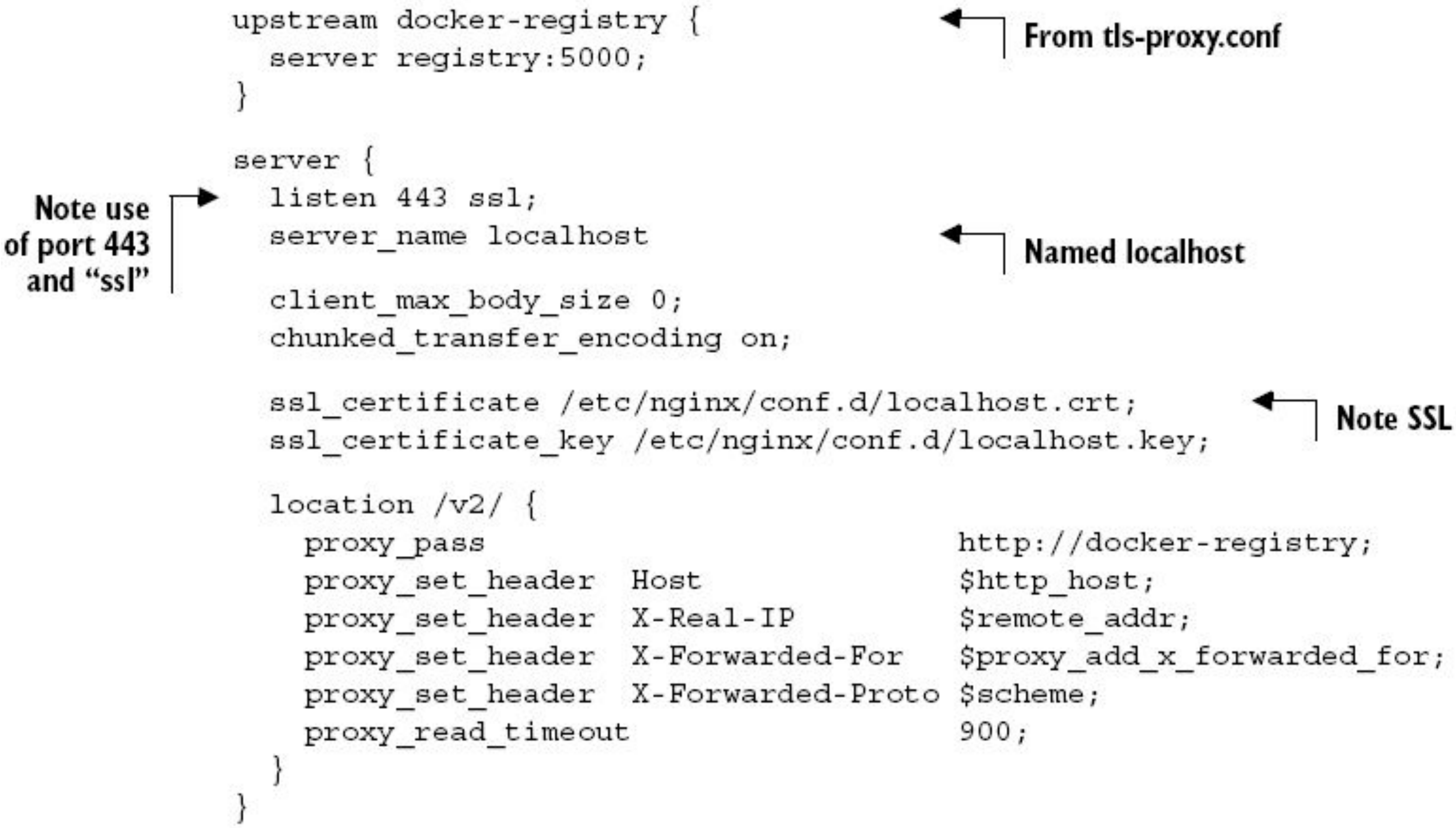
Note /v2/ prefix

Resolves to the upstream

Configuring HTTPS (TLS) on the reverse proxy


Forward requests prefixed with /v2/ to the linked registry container over HTTP
Clients make secure requests to the registry server on HTTPS port 443
Proxy
Registry
Images and metadata are persisted somewhere on disk

www.magedu.com



```
upstream docker-registry {
  server registry:5000;
}

server {
  listen 443 ssl;
  server_name localhost

  client_max_body_size 0;
  chunked_transfer_encoding on;

  ssl_certificate /etc/nginx/conf.d/localhost.crt;
  ssl_certificate_key /etc/nginx/conf.d/localhost.key;

  location /v2/ {
    proxy_pass                            http://docker-registry;
    proxy_set_header  Host                $http_host;
    proxy_set_header  X-Real-IP           $remote_addr;
    proxy_set_header  X-Forwarded-For     $proxy_add_x_forwarded_for;
    proxy_set_header  X-Forwarded-Proto   $scheme;
    proxy_read_timeout                    900;
  }
}
```

From tls-proxy.conf

Note use of port 443 and “ssl”

Named localhost

Note SSL configuration

马哥教育
www.magedu.com

马哥教育

最专业的Linux培训机构

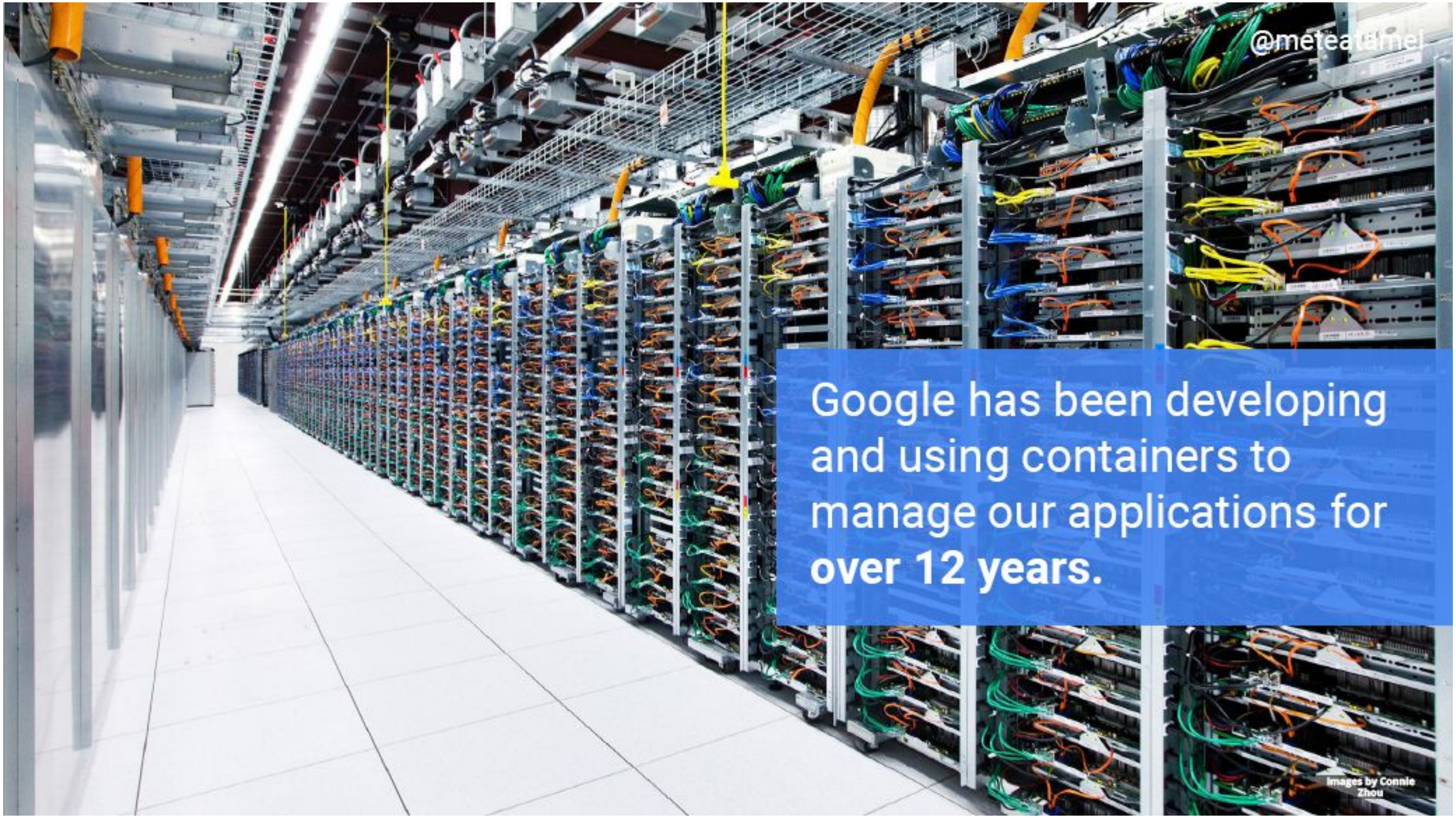
Kubernetes

主讲：马永亮(马哥)
QQ群:169777636
客服QQ：2813150558, 1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

Kubernetes

- Kubernetes is an open source system for managing containerized applications across multiple hosts, providing basic mechanisms for deployment, maintenance, and scaling of applications
- Kubernetes builds upon a decade and a half of experience at Google running production workloads at scale using a system called Borg, combined with best–of–breed ideas and practices from the community

马哥教育
www.magedu.com



Google has been developing and using containers to manage our applications for **over 12 years.**

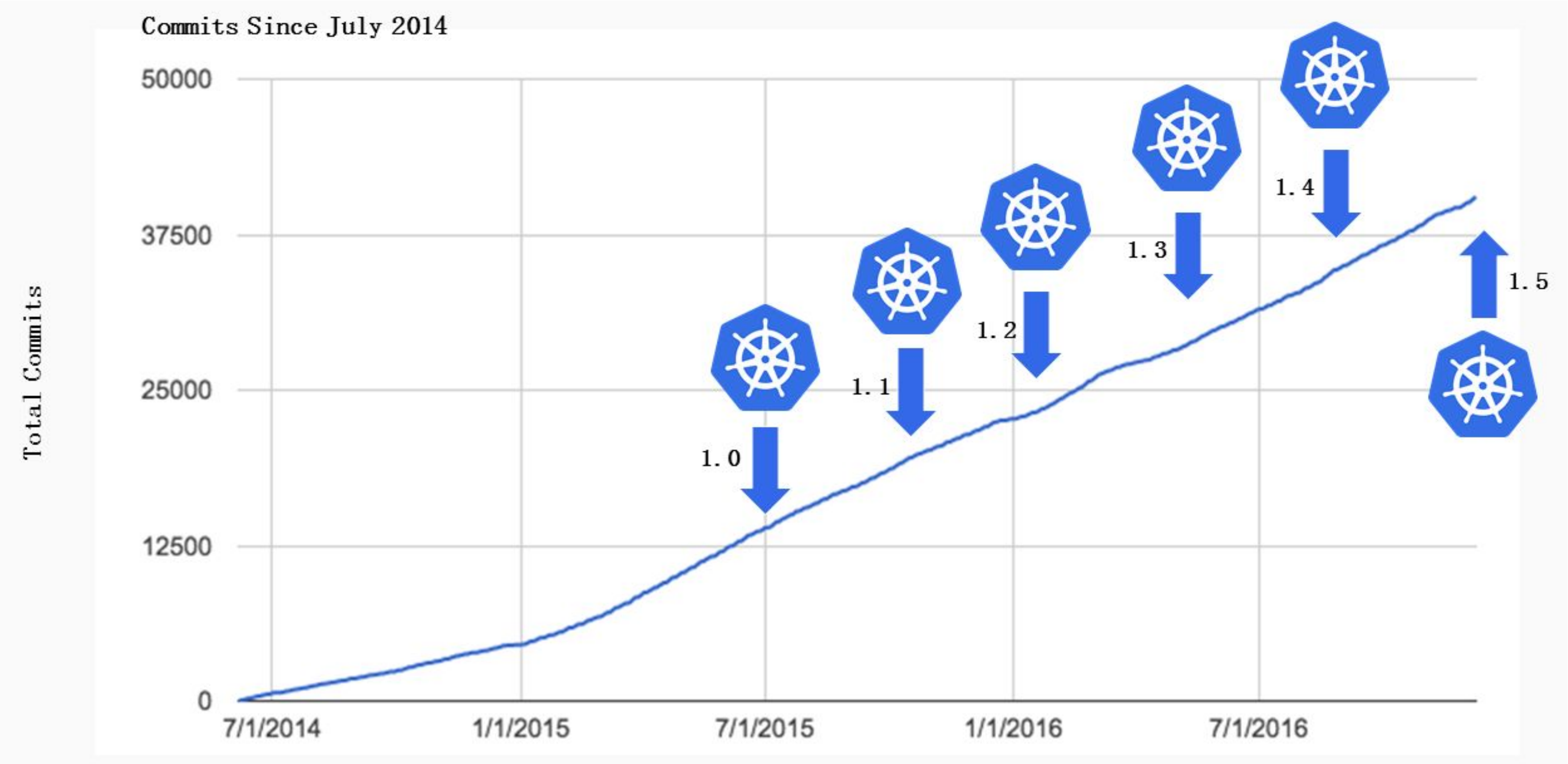
Everything at Google runs in containers

Gmail, Web Search, Maps, ...
MapReduce, batch, ...
GFS, Colossus, ...
Even **Google's Cloud Platform:** our VMs run in containers!

We launch over **2 billion** containers **per week**

Shipping Containers At Clyde, by Steve Gibson

Velocity


Commits Since July 2014
Total Commits
50000
37500
25000
12500
0
7/1/2014
1/1/2015
7/1/2015
1/1/2016
7/1/2016
1. 0
1. 1
1. 2
1. 3
1. 4
1. 5

Adoption

~4k Commits in 1.5

+25% Unique Contributors

Top 0.01% of all Github Projects

3500+ External Projects Based on K8s

Companies Contributing

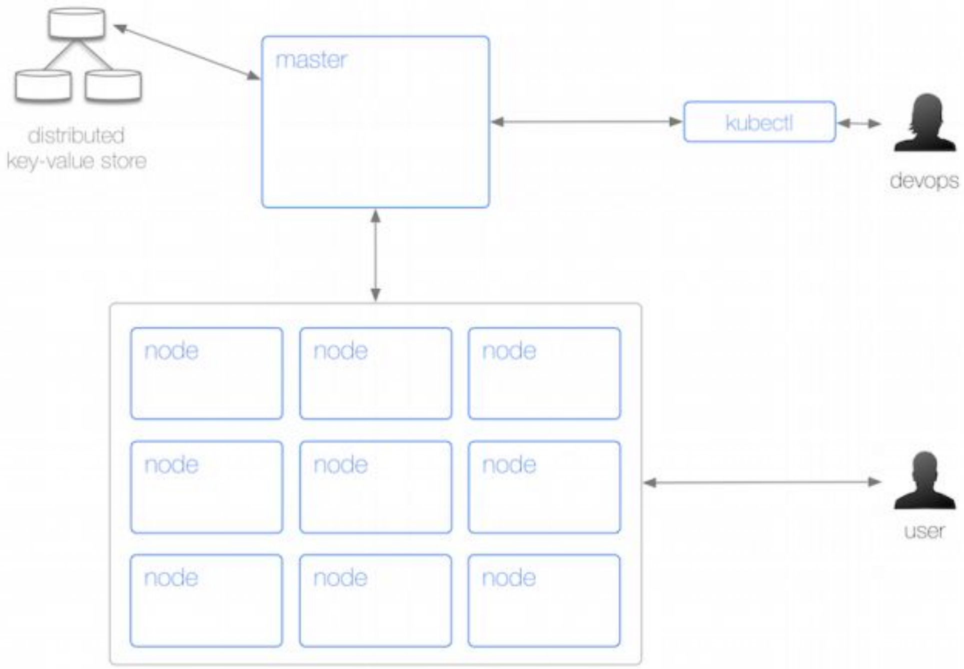
CoreOS weaveworks Microsoft HUAWEI TIGERA MIRANTIS UNIVA
redhat FUJITSU vmware HITACHI intel Engine Yard CITRIX REDAPT IBM

Companies Using

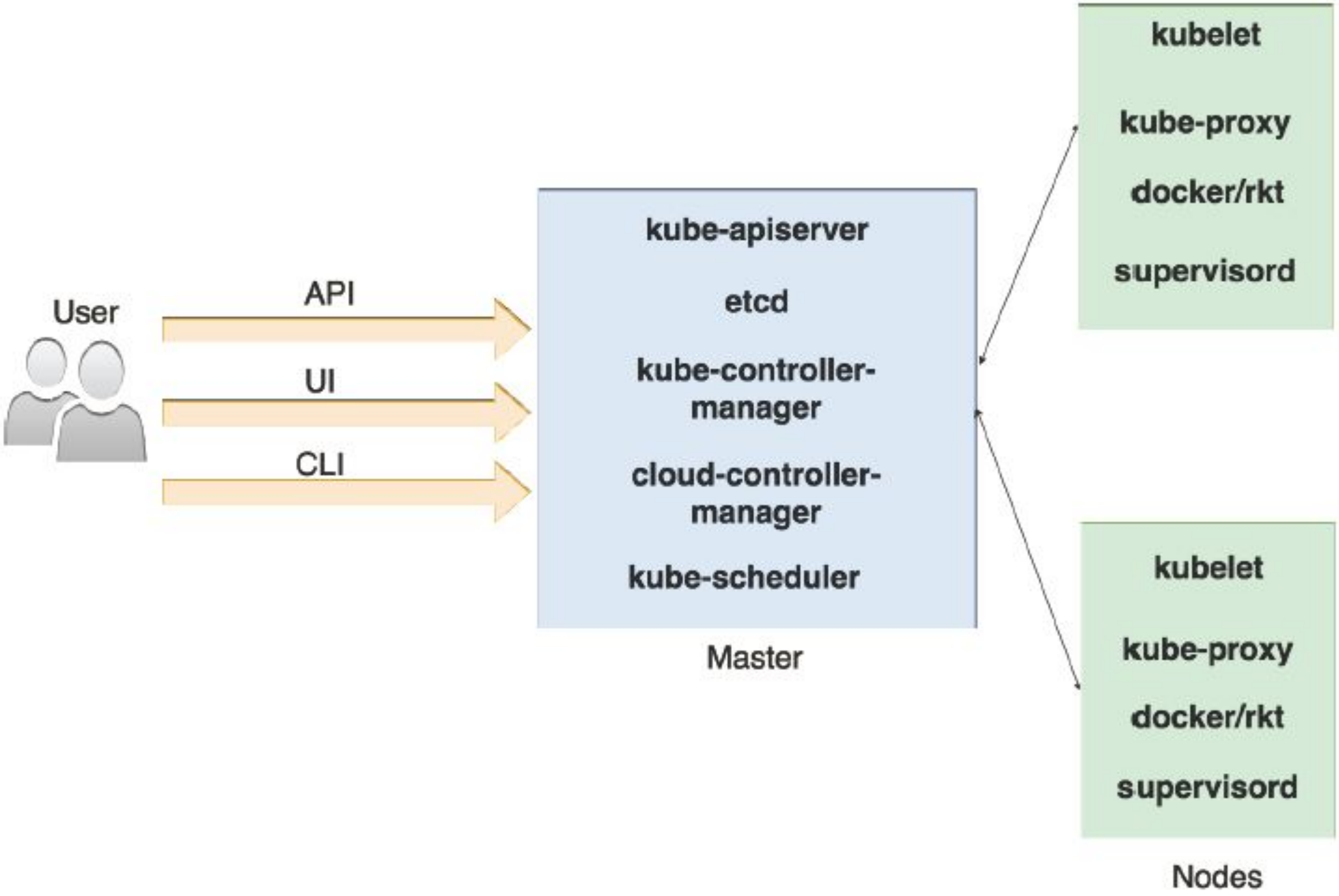
YAHOO! JAPAN box CONCUR ebay Goldman Sachs SAMSUNG SAMSUNG SDS
Pearson The New York Times SOUNDCLOUD DigitalOcean WIKIMEDIA FOUNDATION OpenAI

Kubernetes Cluster

- A running Kubernetes cluster contains node agents (kubelet) and a cluster control plane (AKA *master*), with cluster state backed by a distributed storage system (etcd)

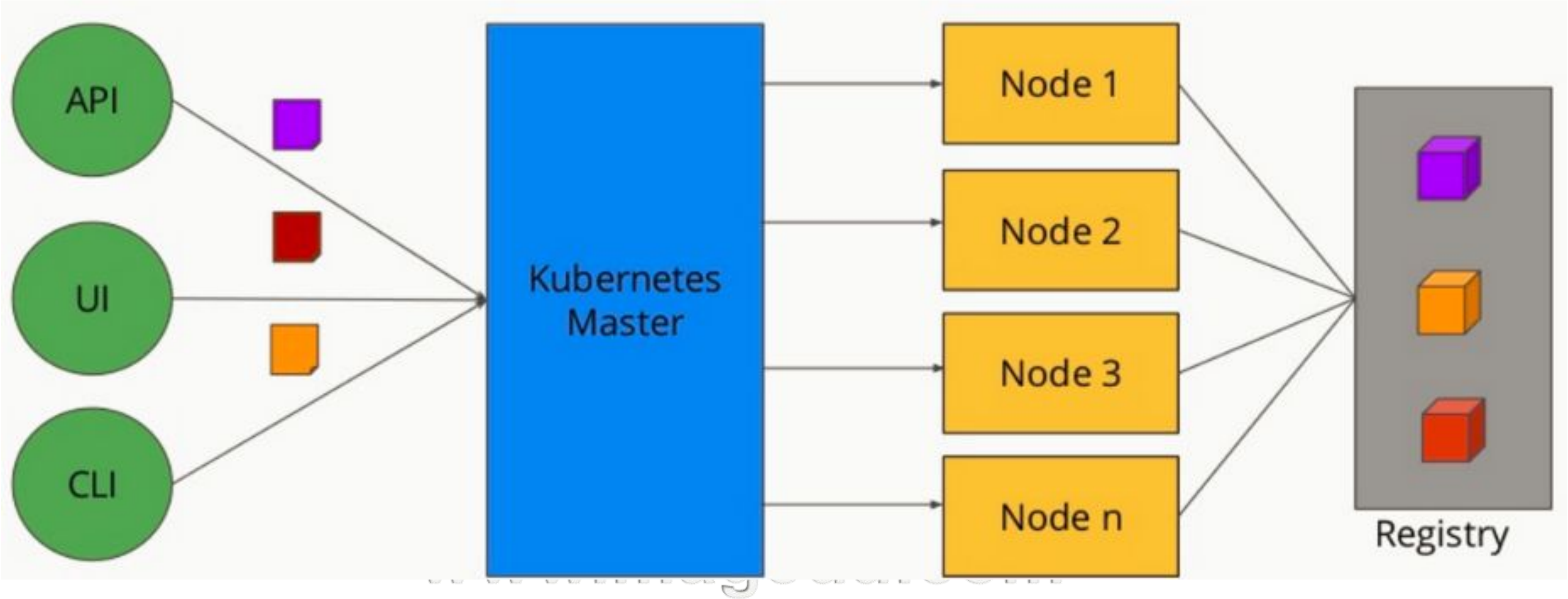


Kubernetes Architecture View



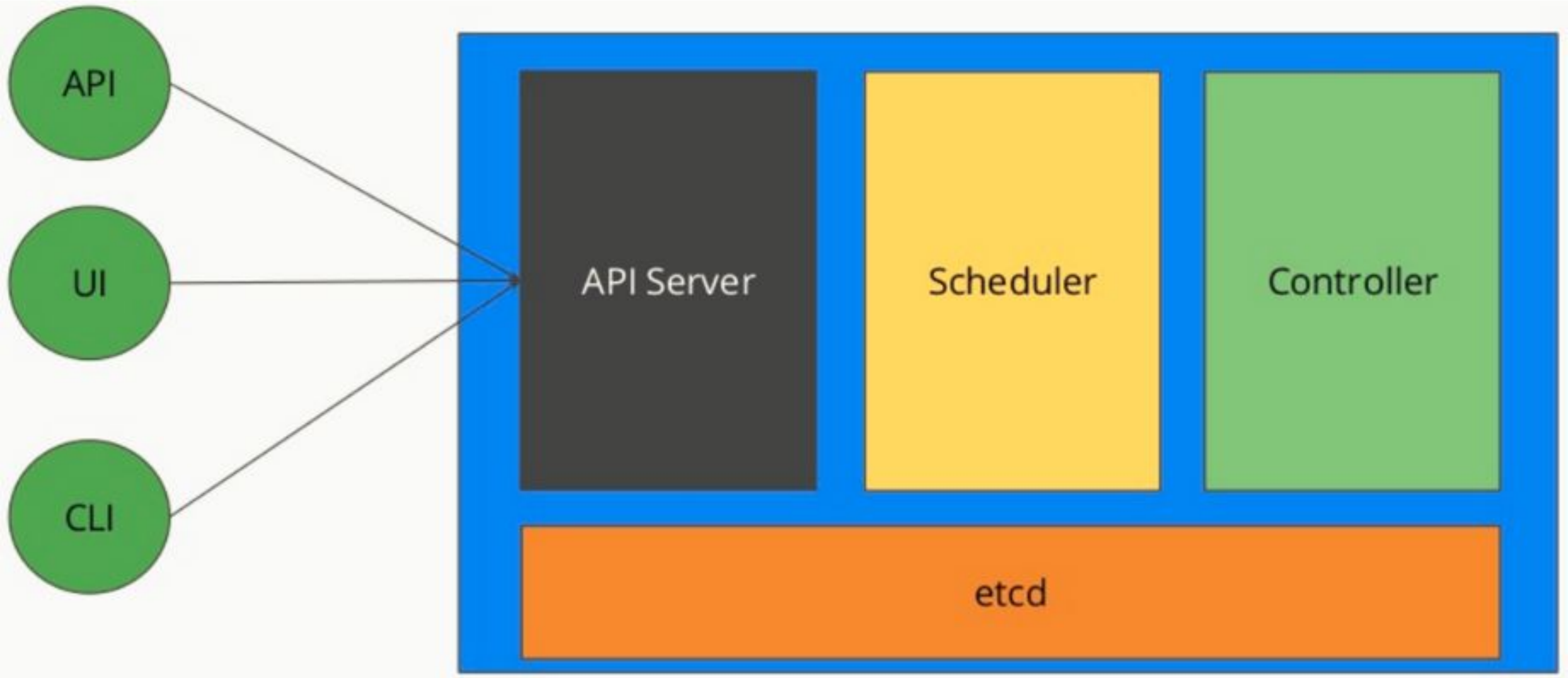
Kubernetes Architecture

- Clients, Master, Node, Registry



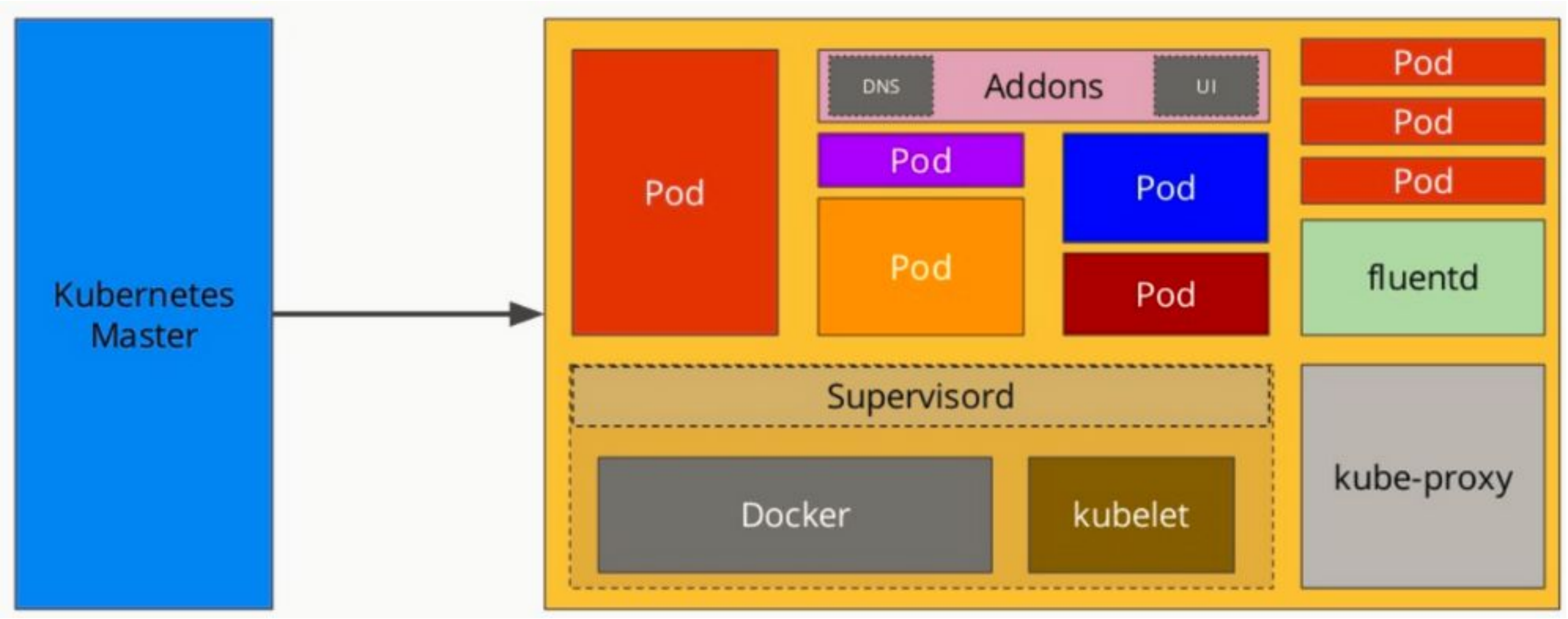
Kubernetes Master

- API Server, Scheduler, Controller–Manager Server

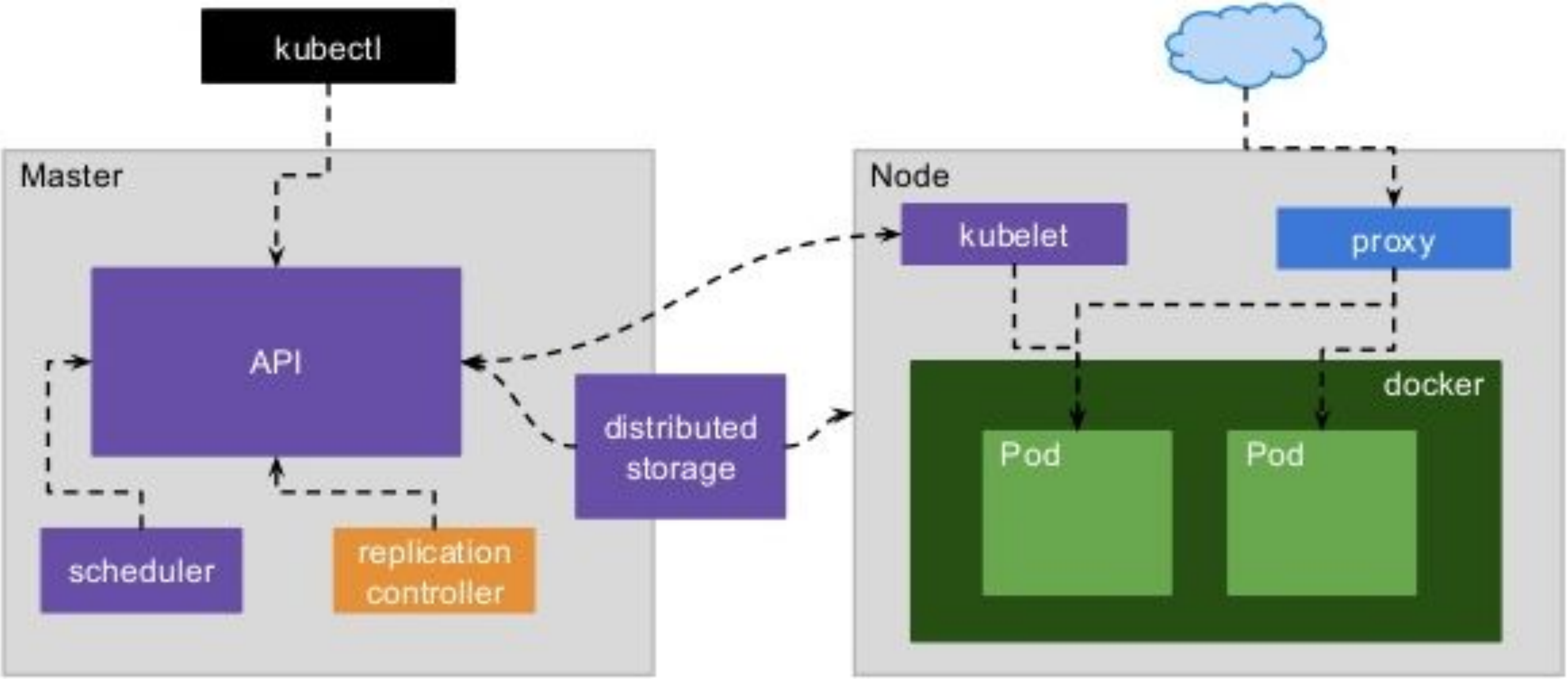


Kubernetes Node

- Kube–proxy, kubelet, Container runtime(docker, rkt, cri–o, frakti) , Pod



Kubernetes Architecture



www.magedu.com

Key Concepts of Kubernetes

- Pod – A group of Containers
- Labels – Labels for identifying pods
- Kubelet – Container Agent
- kube–proxy – A load balancer for Pods
- etcd – A metadata service
- cAdvisor – Container Advisor providers resource usage/performance statistics
- Replication Controller – Manages replication of pods
- Scheduler – Schedules pods in worker nodes
- API Server – Kubernetes API server

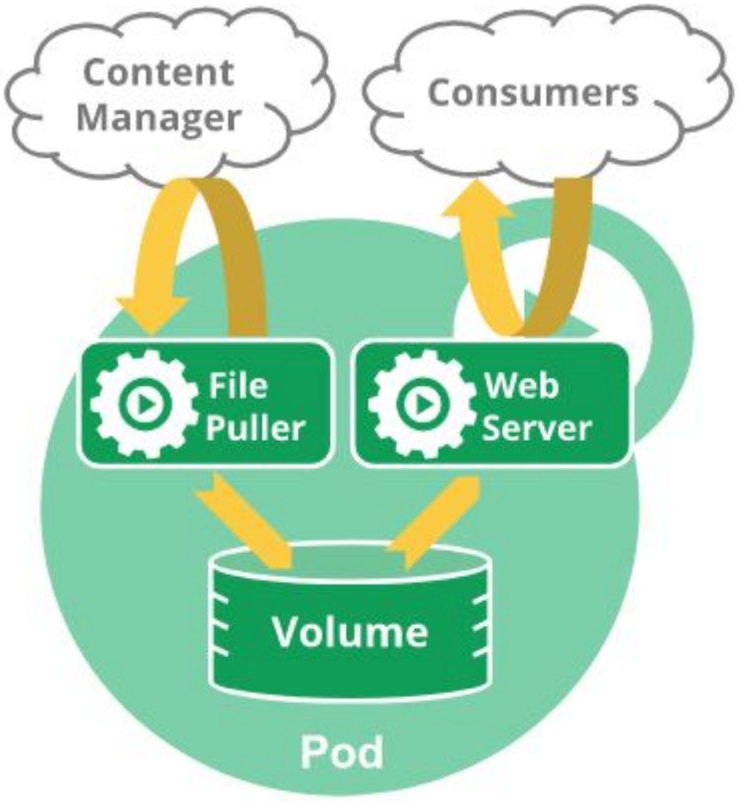
马哥教育
www.magedu.com

What is Pod?

- A group of one or more containers that are always co–located and co–scheduled that share the context
- Containers in a pod share the same IP address, ports, hostname and storage
- Modeled like a virtual machine
  - Each container represents one process
  - Tightly coupled with other containers in the same pod
- Pods are scheduled in Nodes
- Fundamental unit of deployment in Kubernetes

马哥教育
www.magedu.com

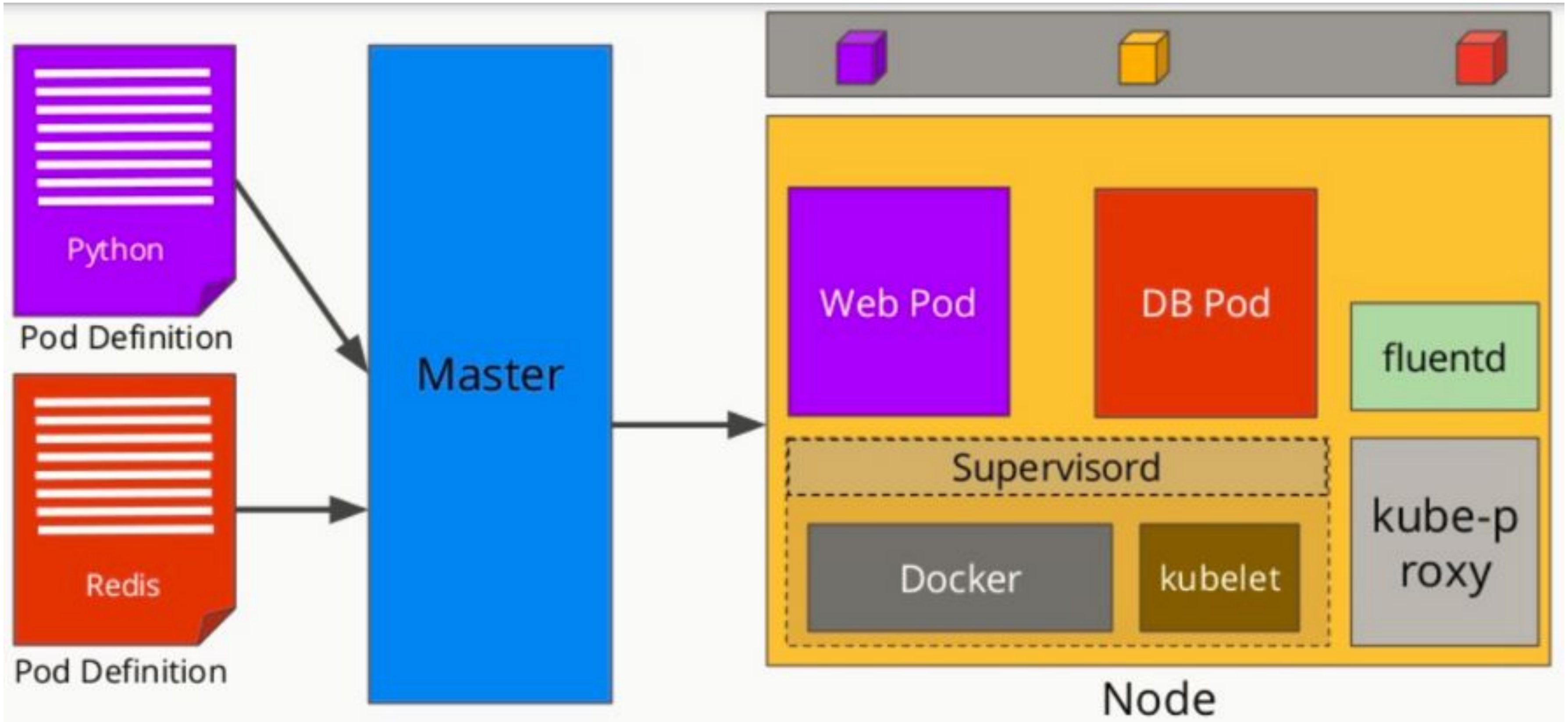
What is a Pod?



- Containers within the same pod communicate with each other using IPC;
- Containers can find each other via localhost;
- Each container inherits the name of the pod
- Each pod has an IP address in a flat shared networking space
- Volumes are shared by containers in a pod

哥教育
magedu.com

Deploying a Pod



Use cases for Pod

- Content management systems, file and data loaders, local cache managers, etc.
- Log and checkpoint backup, compression, rotation, snapshotting, etc.
- Data change watchers, log tailers, logging and monitoring adapters, event publishers, etc.
- Proxies, bridges, and adapters
- Controllers, managers, configurators, and updaters

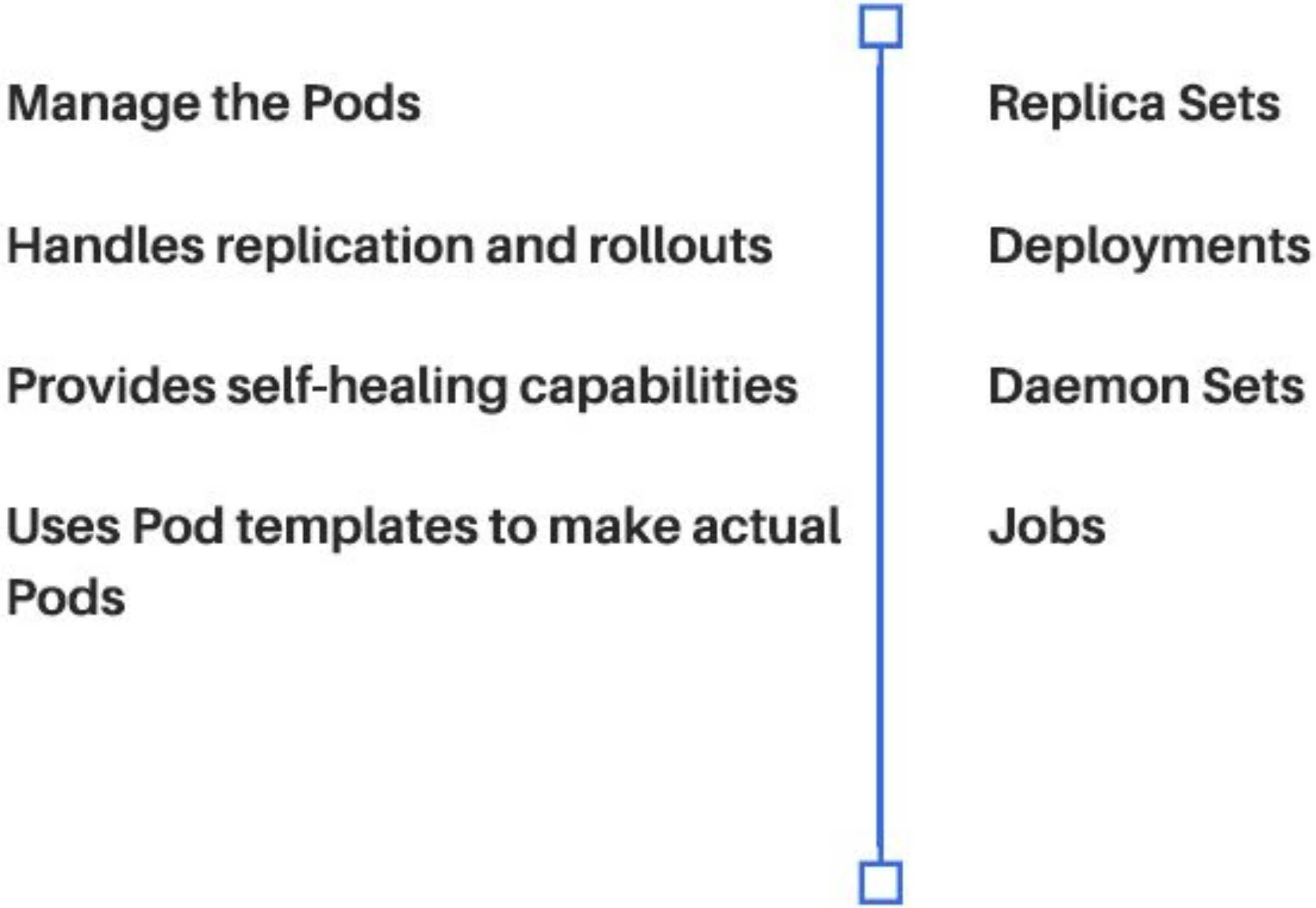
马哥教育
www.magedu.com

Replication Controller

- Ensures that a Pod or homogeneous set of Pods are always up and available
- Always maintains desired number of Pods
  - If there are excess Pods, they get killed
  - New pods are launched when they fail, get deleted, or terminated
- Creating a replication controller with a count of 1 ensures that a Pod is always available
- Replication Controller and Pods are associated through Labels

马哥教育
www.magedu.com

Controllers



Manage the Pods

Handles replication and rollouts

Provides self-healing capabilities

Uses Pod templates to make actual Pods

Replica Sets

Deployments

Daemon Sets

Jobs

Services

- Kubernetes Pods are mortal, they are born and when they die, they are not resurrected
- ReplicationControllers in particular create and destroy **Pods** dynamically (e.g. when scaling up or down or when doing rolling_updates)
- While each **Pod** gets its own IP address, even those IP addresses cannot be relied upon to be stable over time
- This leads to a problem: if some set of **Pods** (let's call them backends) provides functionality to other **Pods** (let's call them frontends) inside the Kubernetes cluster, how do those frontends find out and keep track of which backends are in that set?

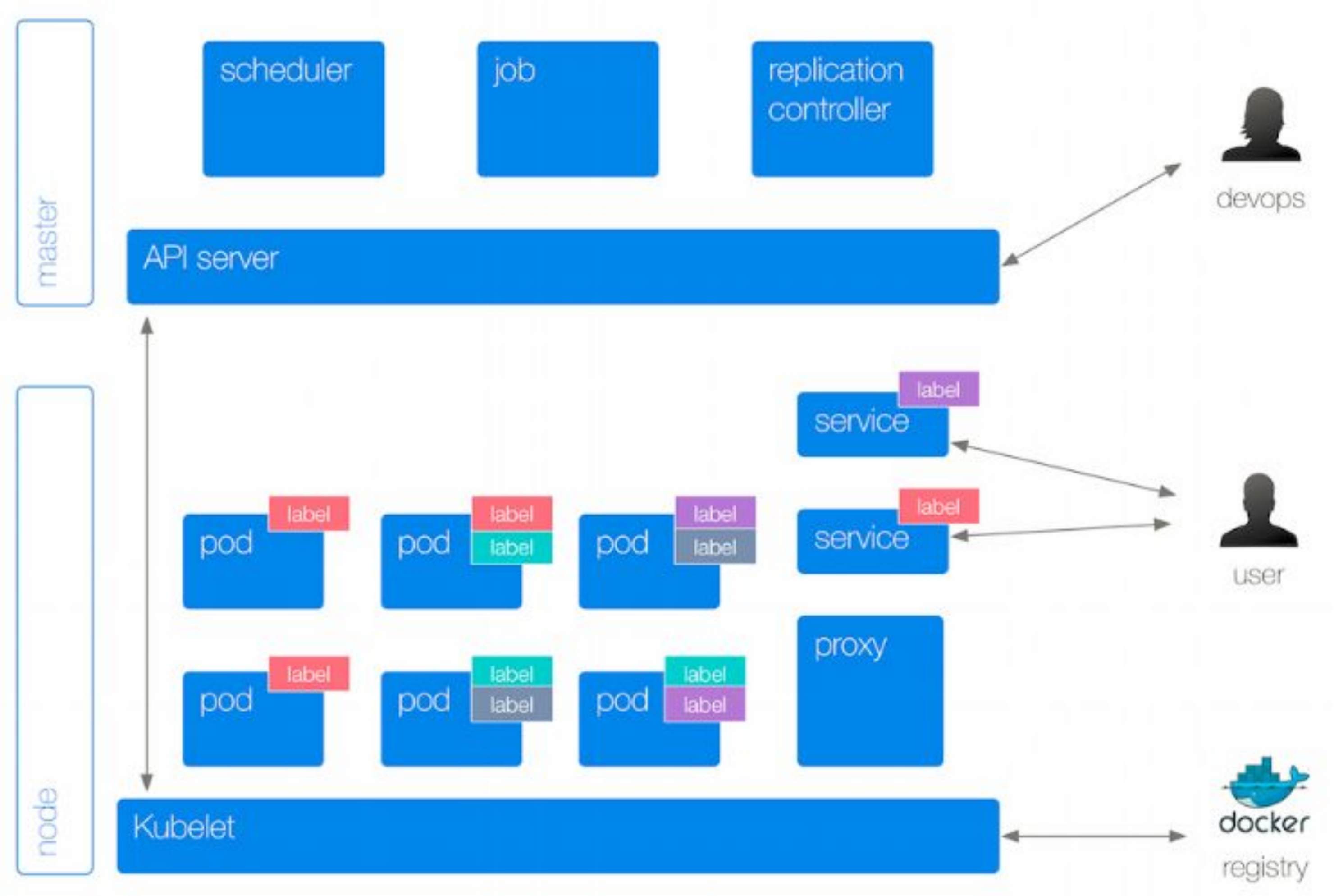
马哥教育
www.magedu.com

Services

- A Kubernetes **Service** is an abstraction which defines a logical set of **Pods** and a policy by which to access them – sometimes called a micro-service
- The set of **Pods** targeted by a **Service** is (usually) determined by a Label Selector
- For Kubernetes-native applications, Kubernetes offers a simple **Endpoints** API that is updated whenever the set of **Pods** in a **Service** changes
- For non-native applications, Kubernetes offers a virtual-IP-based bridge to Services which redirects to the backend **Pods**

马哥教育
www.magedu.com

Kubernetes Object: Service



magedu.com
专注于Linux培训

Labels & Selectors

- Key/Value pairs associated with Kubernetes objects
- Used to organize and select subsets of objects
- Attached to objects at creation time but modified at any time
- Labels are the essential glue to associated one API object with other
  - Replication Controller   Pods
  - Service   Pods
  - Pods   Nodes

马哥教育
www.magedu.com

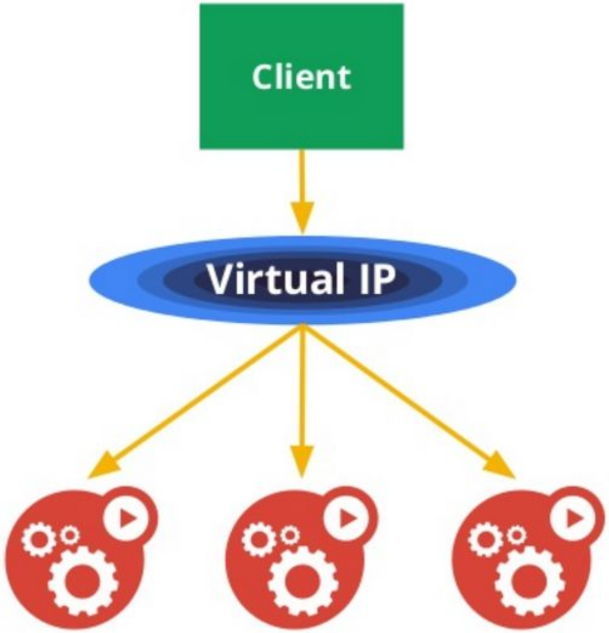
Services

- An abstraction to define a logical set of Pods bound by a policy by to access them
- Services are exposed through internal and external endpoints
- Services can also point to non–Kubernetes endpoints through a Virtual–IP–Bridge
- Supports TCP and UDP
- Interfaces with kube–proxy to manipulate iptables
- Service can be exposed internal or external to the cluster
  - A Service as a static API object
  - virtual, but static IP
  - no service discovery necessary

马哥教育
www.magedu.com

Services

- A group of pods that work together
  - grouped by a selector
- Defines access policy
  - "load balanced" or "headless"
- Gets a stable virtual IP and port
  - sometimes called the service portal
  - also a DNS name
- VIP is managed by kube–proxy
  - watches all services
  - updates iptables when backends change
- Hides complexity–ideal for non–native apps



马哥教育
www.magedu.com

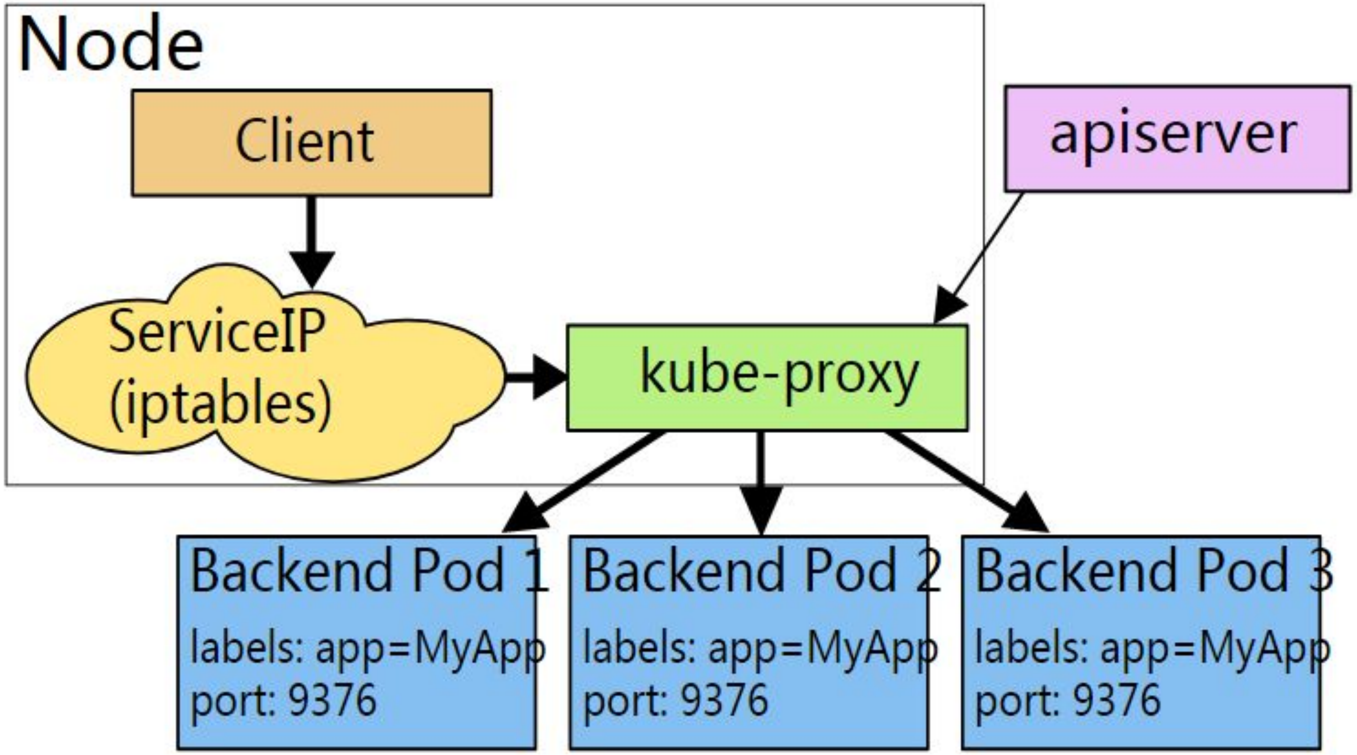
kube–proxy

- Runs on each node – proxies UDP and TCP – does not understand HTTP – provides load balancing – is just used to reach services
  - The Kubernetes network proxy runs on each node
  - This reflects services as defined in the Kubernetes API on each node and can do simple TCP,UDP stream forwarding or round robin TCP,UDP forwarding across a set of backends
  - Service cluster ips and ports are currently found through Docker–links–compatible environment variables specifying ports opened by the service proxy
  - There is an optional addon that provides cluster DNS for these cluster Ips
  - The user must create a service with the apiserver API to configure the proxy

马哥教育
www.magedu.com

Proxy–mode: userspace

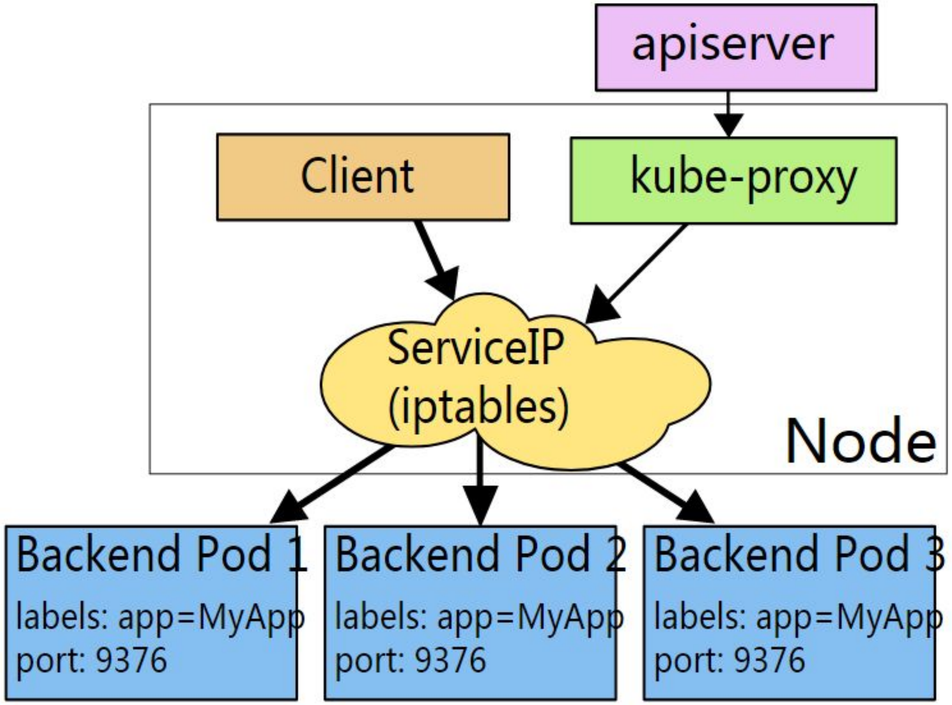
- For each Service it opens a port (randomly chosen) on the local node
- Any connections to this "proxy port" will be proxied to one of the **Service**'s backend **Pods**



Proxy–mode: iptables

- For each **Service** it installs iptables rules which capture traffic to the **Service**'s **clusterIP** (which is virtual) and **Port** and redirects that traffic to one of the **Service**'s backend sets
- For each **Endpoints** object it installs iptables rules which select a backend **Pod**

马
www.m



Discovering services

- Kubernetes supports 2 primary modes of finding a Service – environment variables and DNS
- Environment variables
  - When a **Pod** is run on a **Node**, the kubelet adds a set of environment variables for each active **Service**. It supports both Docker links compatible variables and simpler {SVCNAME}_SERVICE_HOST and {SVCNAME}_SERVICE_PORT variables , where the Service name is upper–cased and dashes are converted to underscores
- DNS
  - An optional (though strongly recommended) cluster add–on is a DNS server
  - The DNS server watches the Kubernetes API for new **Services** and creates a set of DNS records for each
  - If DNS has been enabled throughout the cluster then all **Pods** should be able to do name resolution of **Services** automatically

马哥教育
www.magedu.com

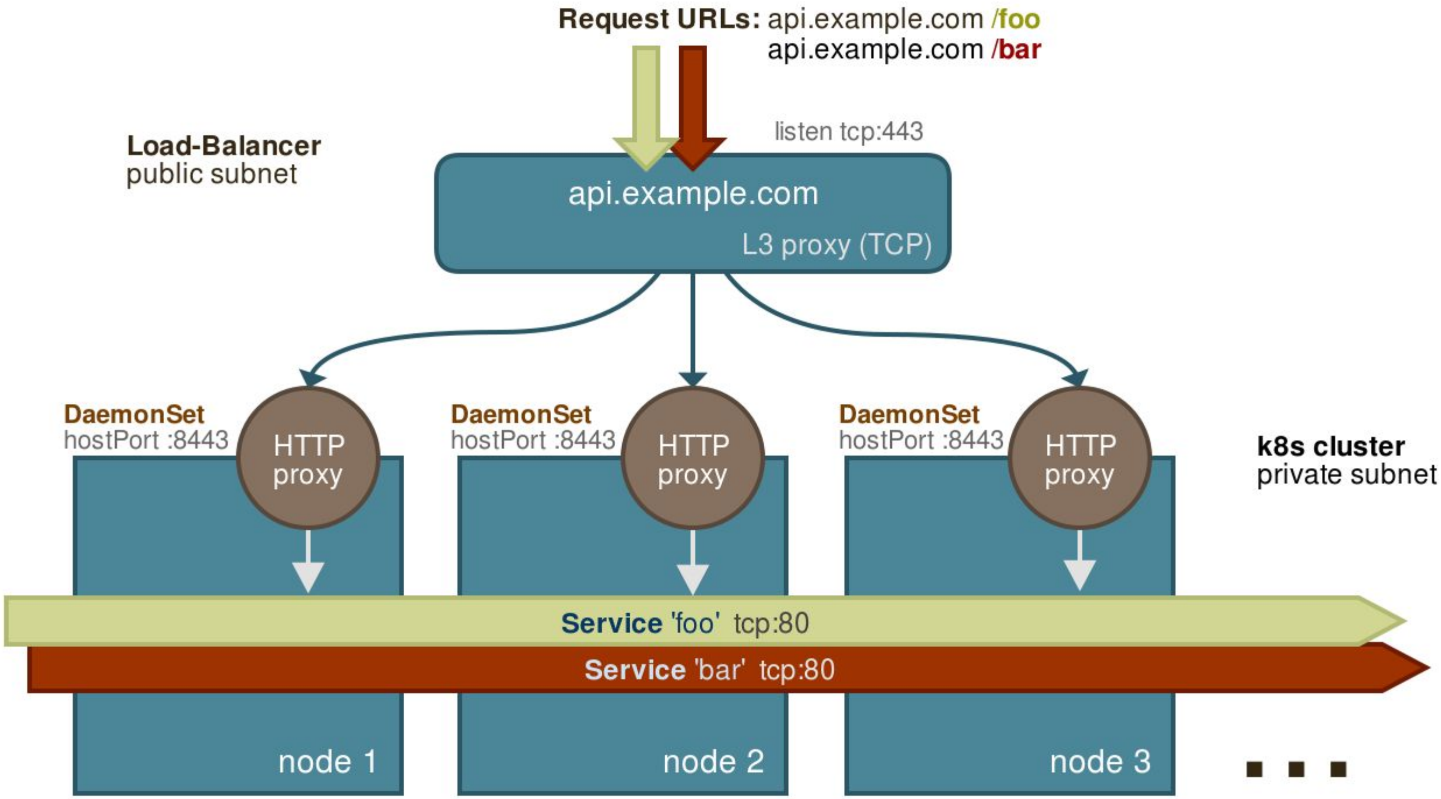
What is Ingress?

- Typically, services and pods have IPs only routable by the cluster network. All traffic that ends up at an edge router is either dropped or forwarded elsewhere
- An Ingress is a collection of rules that allow inbound connections to reach the cluster services

马哥教育
www.magedu.com

Ingress

- Ingress –– url based routing
- An Ingress is a collection of rules that allow inbound connections to reach the cluster services.



马哥教育
www.magedu.com

马哥教育
www.magedu.com

马哥教育
www.magedu.com

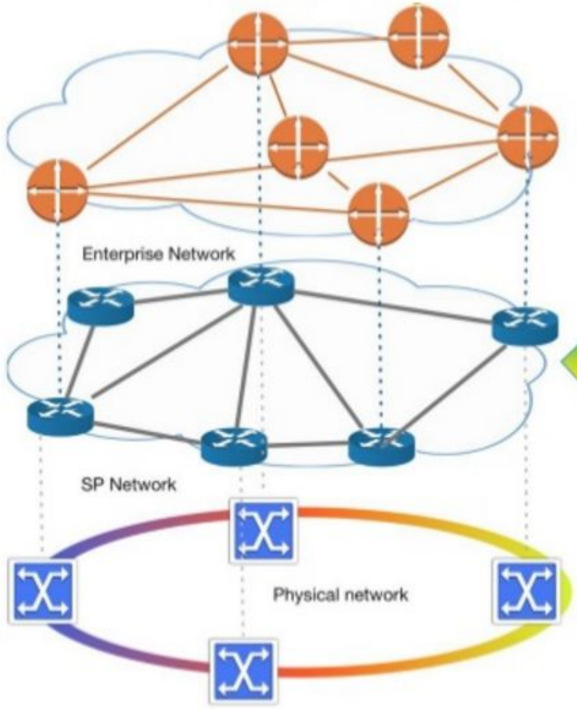
Kubernetes networking

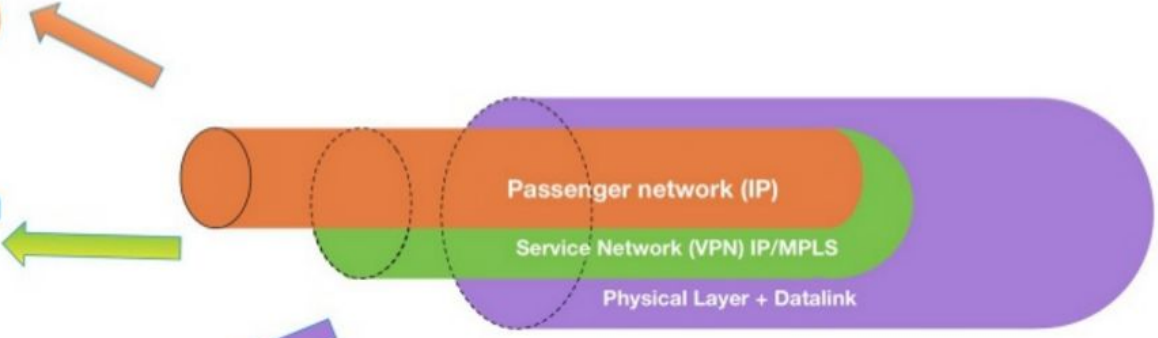
- On GCE(Google Compute Engine)/GKE
  - GCE Advanced Routes (program the fabric)
  - "Everything to 10.1.1.0/24, send to this VM"
- Plenty of other ways
  - Cilium
  - OVN (Open Virtual Networking)
  - Calico
  - Romana
  - Flannel
  - OVS
  - Open Contrail
  - Cisco Contiv
  - Others...

马哥教育
www.magedu.com

Overlay network concept

Network layers concept

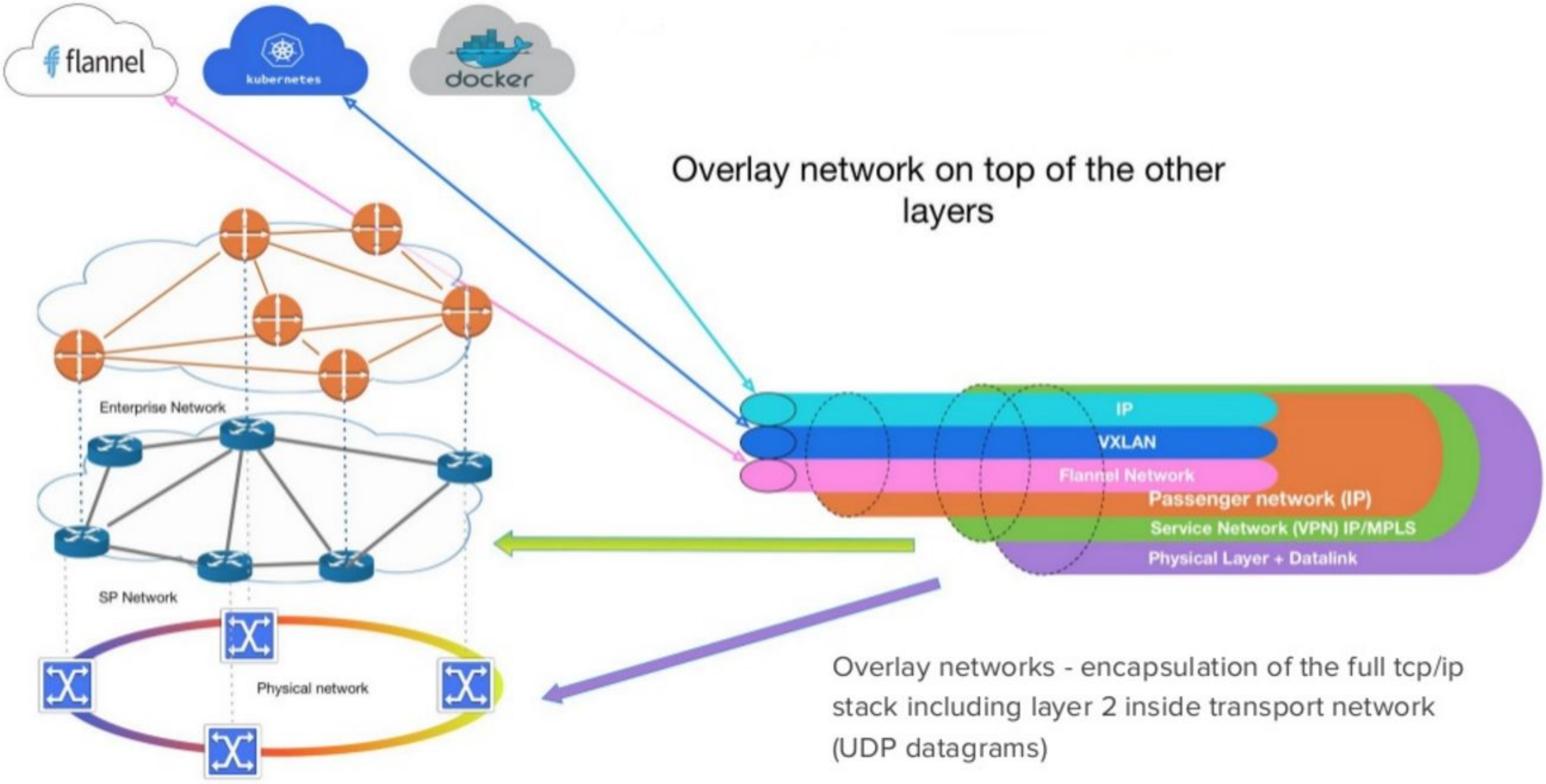




Stacking nature of networking allows encapsulation of different protocol stack at each network layer:

- Datalink could be encapsulated in datalink, IP, transport (e.g. UDP, IPSEC)
- IP could be encapsulated in IP, MPLS, Datalink
- MPLS allows simple stacking for complex service-oriented topologies

Overlay network concept

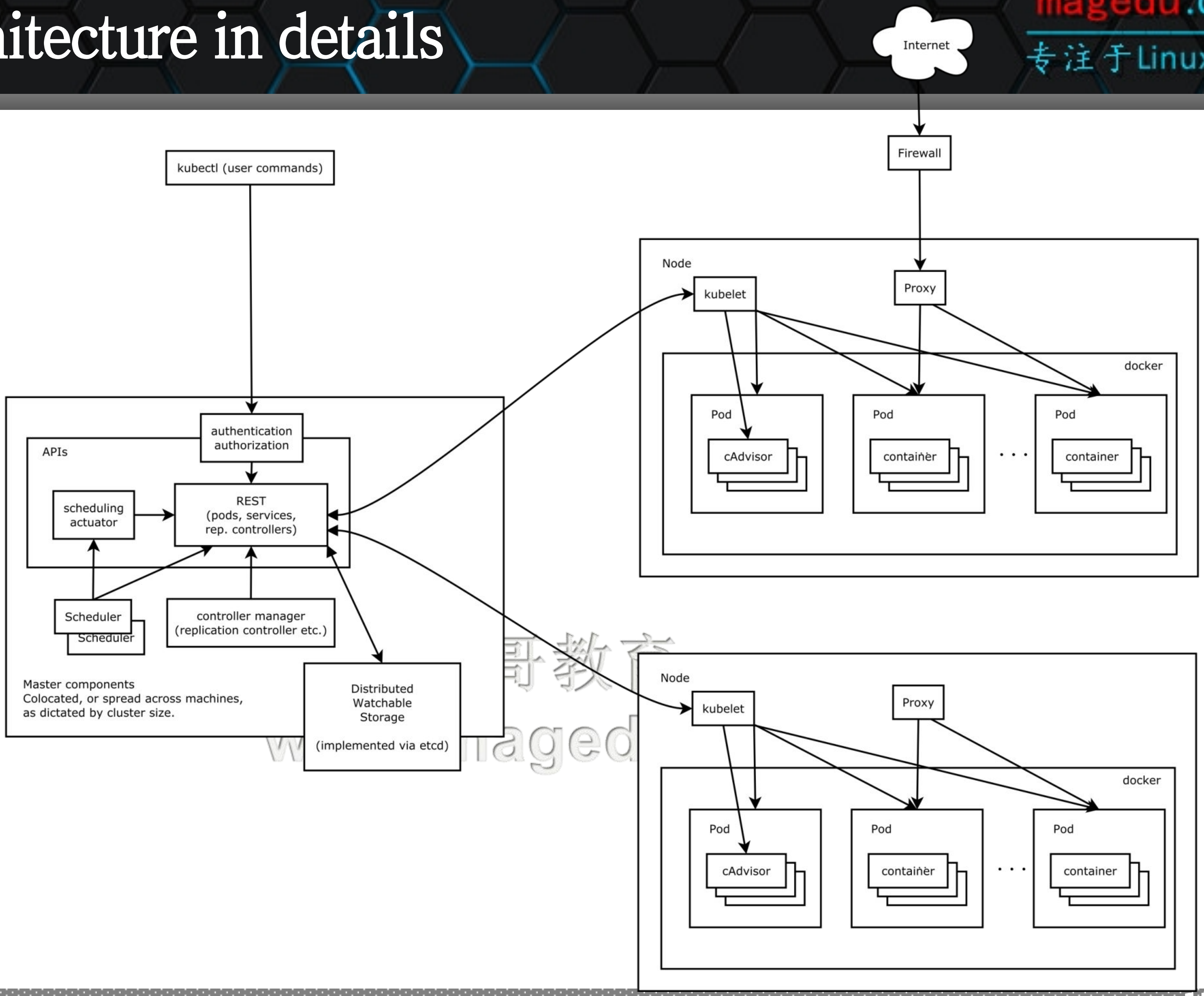


Overlay network on top of the other layers

Overlay networks - encapsulation of the full tcp/ip stack including layer 2 inside transport network (UDP datagrams)

4

Architecture in details



马哥教育
www.magedu.com

Kubernetes Objects

- Kubernetes Objects are persistent entities in the Kubernetes system
- Kubernetes uses these entities to represent the state of your cluster
- Specifically, they can describe:
  - What containerized applications are running (and on which nodes)
  - The resources available to those applications
  - The policies around how those applications behave, such as restart policies, upgrades, and fault–tolerance

马哥教育
www.magedu.com

Kubernetes Objects

- A Kubernetes object is a "record of intent"–once you create the object, the Kubernetes system will constantly work to ensure that object exists
- By creating an object, you're effectively telling the Kubernetes system what you want your cluster's workload to look like; this is your cluster's desired state
- To work with Kubernetes objects–whether to create, modify, or delete them–you'll need to use the Kubernetes API
- When you use the **kubectl** command-line interface, for example, the CLI makes the necessary Kubernetes API calls for you; you can also use the Kubernetes API directly in your own programs

马哥教育
www.magedu.com

Object Spec and Status

- Every Kubernetes object includes two nested object fields that govern the object's configuration: the object *spec* and the object *status*
  - The *spec*, which you must provide, describes your *desired state* for the object–the characteristics that you want the object to have
  - The *status* describes the *actual state* for the object, and is supplied and updated by the Kubernetes system
- **At any given time, the Kubernetes Control Plane actively manages an object's actual state to match the desired state you supplied**

马哥教育
www.magedu.com

Describing a Kubernetes Object

- When you create an object in Kubernetes, you must provide the object spec that describes its desired state, as well as some basic information about the object (such as a name)
- When you use the Kubernetes API to create the object (either directly or via kubectl), that API request must include that information as JSON in the request body
- **Most often, you provide the information to kubectl in a .yaml file. kubectl** converts the information to JSON when making the API request

马哥教育
www.magedu.com

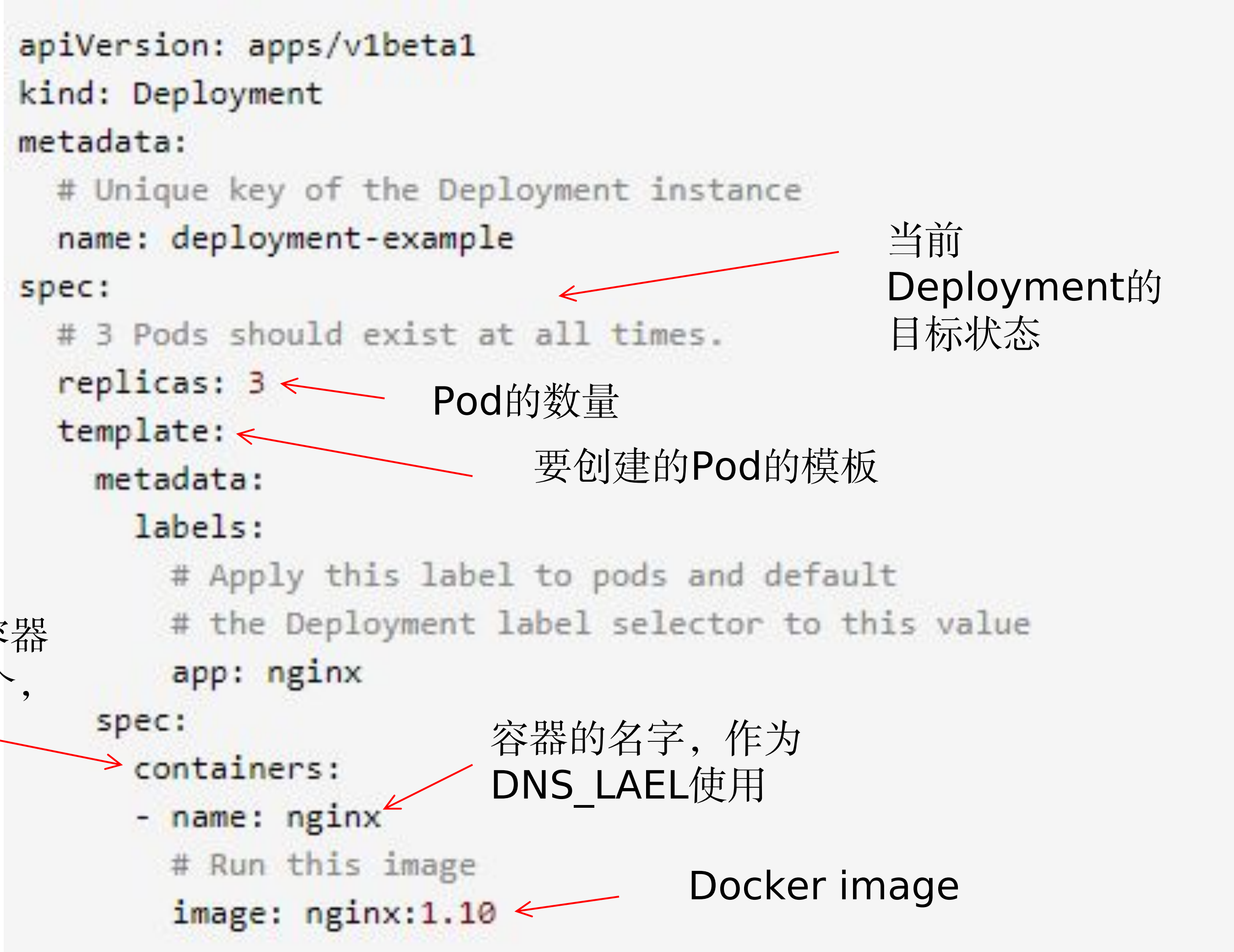
Deployment

- Deployment enables declarative updates for Pods and ReplicaSets

马哥教育
www.magedu.com

一个示例

- kubectl create –f ./deployment–example.yaml ––record



```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  # Unique key of the Deployment instance
  name: deployment-example
spec:
  # 3 Pods should exist at all times.
  replicas: 3
  template:
    metadata:
      labels:
        # Apply this label to pods and default
        # the Deployment label selector to this value
        app: nginx
    spec:
      containers:
      - name: nginx
        # Run this image
        image: nginx:1.10
```

当前Deployment的目标状态

Pod的数量

要创建的Pod的模板

列出Pod中的容器，至少应该一个，且不可被更新

容器的名字，作为DNS_LAEL使用

Docker image

Required Fields

- In the .yaml file for the Kubernetes object you want to create, you'll need to set values for the following fields
  - apiVersion – Which version of the Kubernetes API you're using to create this object
  - kind – What kind of object you want to create
  - metadata – Data that helps uniquely identify the object, including a name string, UID, and optional namespace

马哥教育
www.magedu.com

- 各种类型对象及其相关属性文档：
  https://kubernetes.io/docs/resources-reference/v1.5

马哥教育
www.magedu.com

马哥教育
www.magedu.com

马哥教育
www.magedu.com

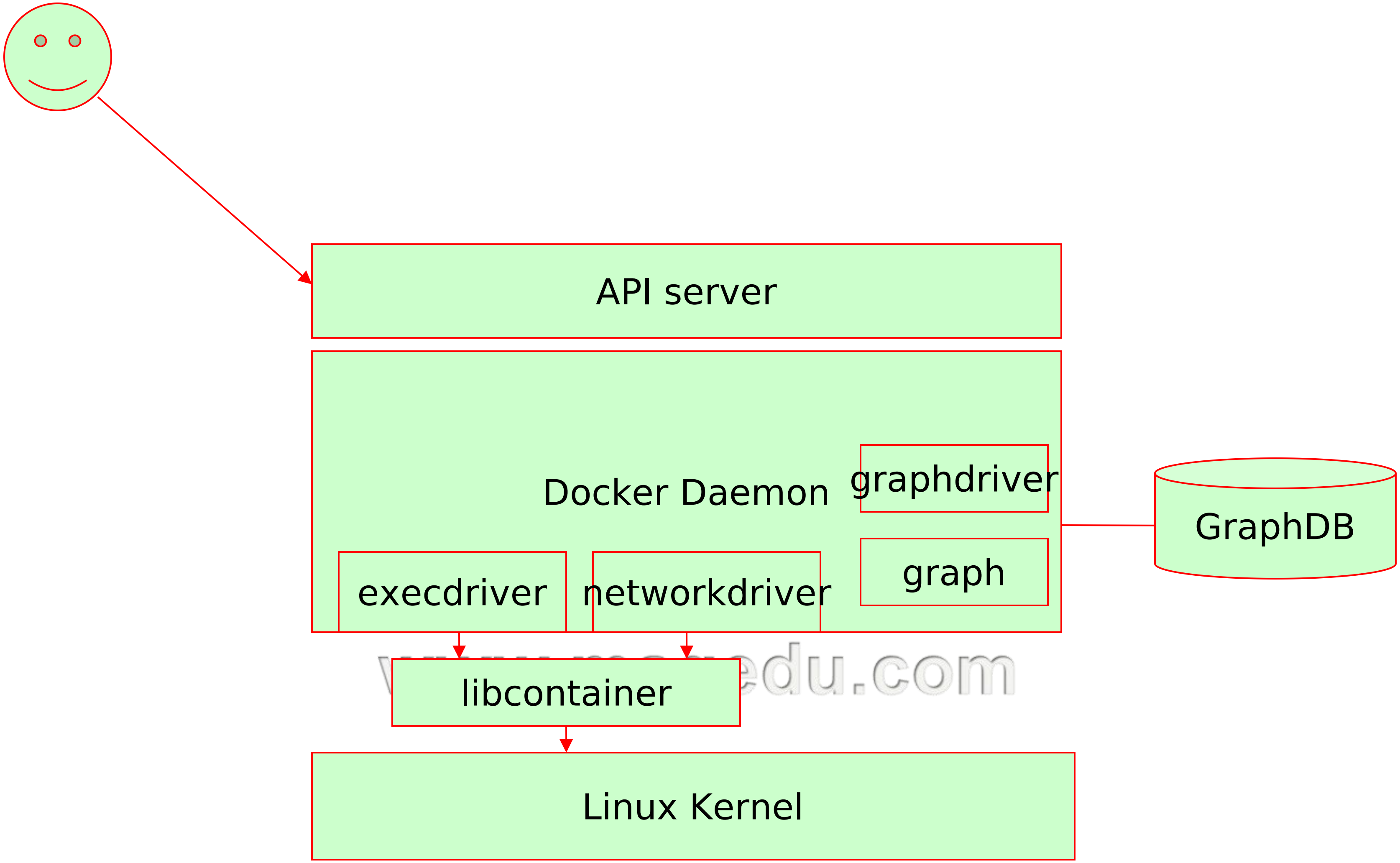
马哥教育
www.magedu.com

Kubernetes Workloads资源类型

- WORKLOADS
  - Container
  - CronJob
  - DaemonSet
  - Deployment
  - Job
  - Pod
  - ReplicaSet
  - ReplicationController
  - StatefulSet

马哥教育
www.magedu.com

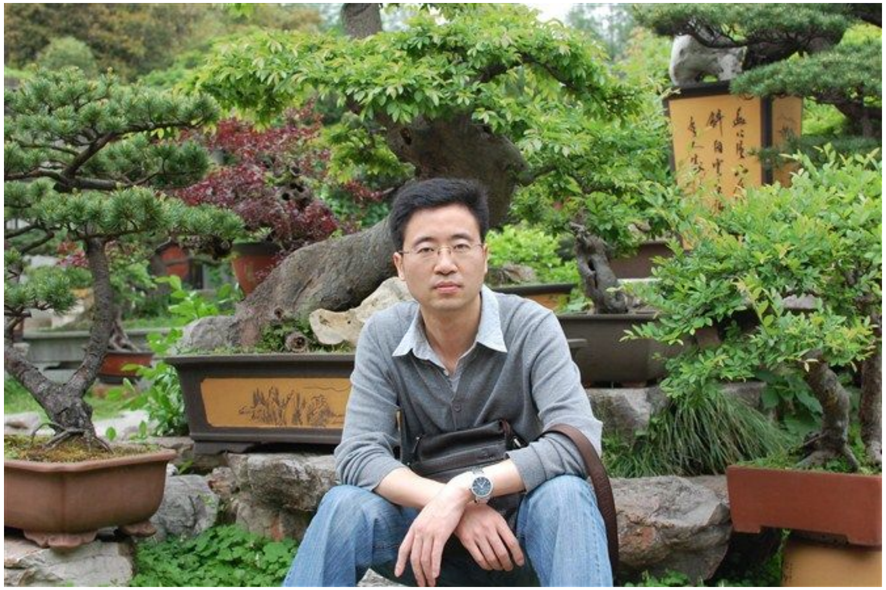
架构



www.magedu.com

关于马哥教育

- 博客：http://mageedu.blog.51cto.com
- 主页：http://www.magedu.com
- QQ：2813150558, 1661815153, 113228115
- QQ群：169777636, 279599283



马哥教育

最专业的Linux培训机构

Thank You!