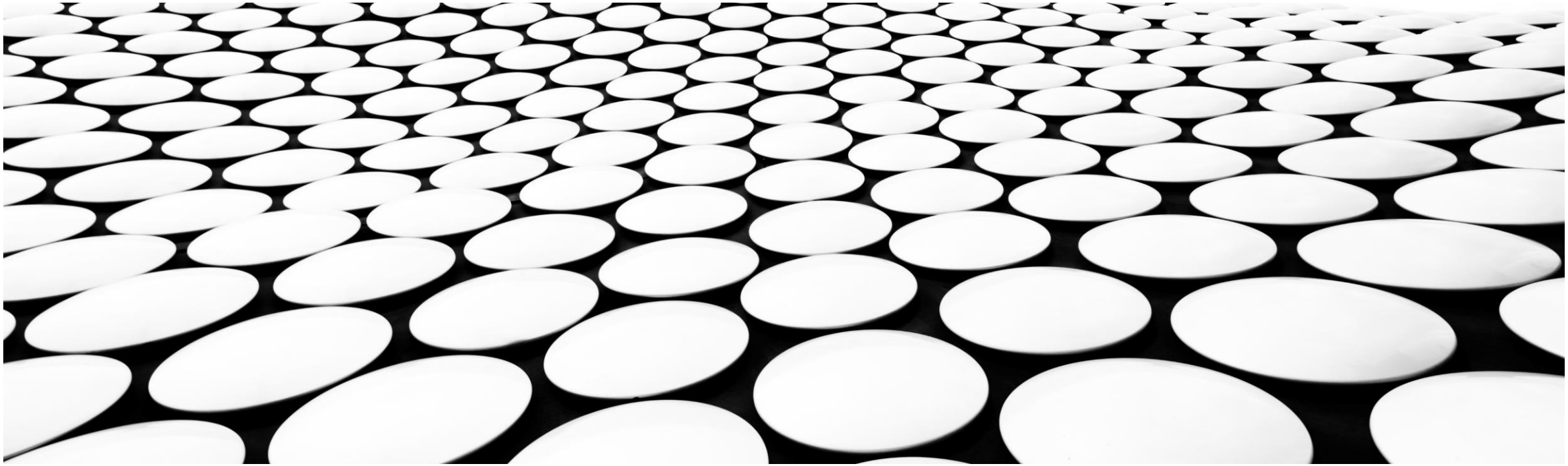
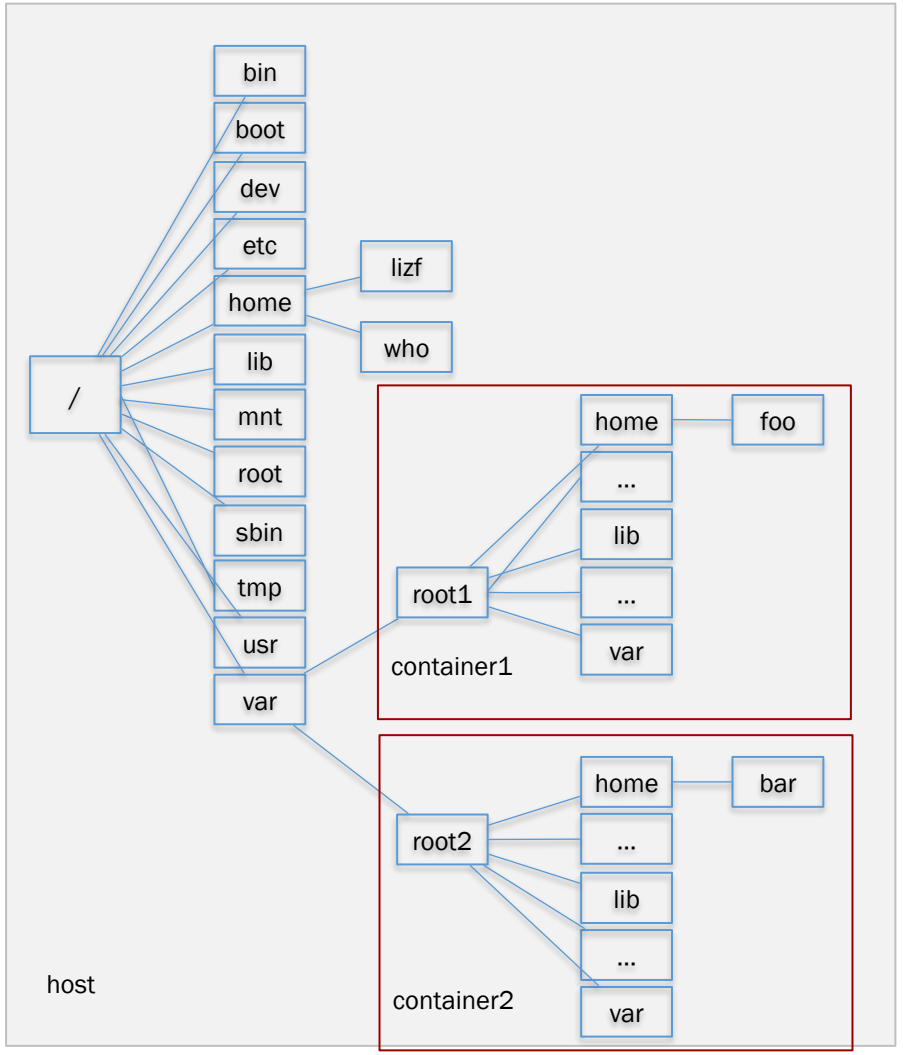


服务端开发-Docker使用

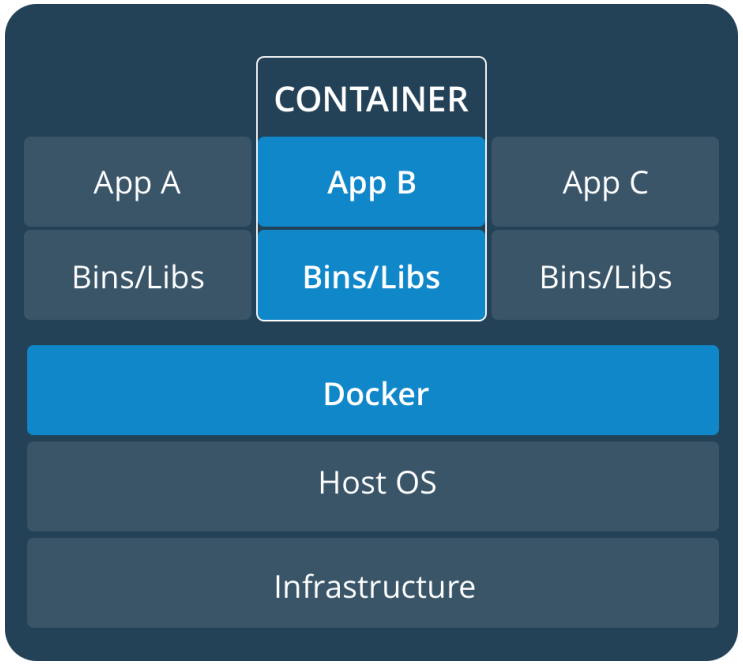
陶召胜

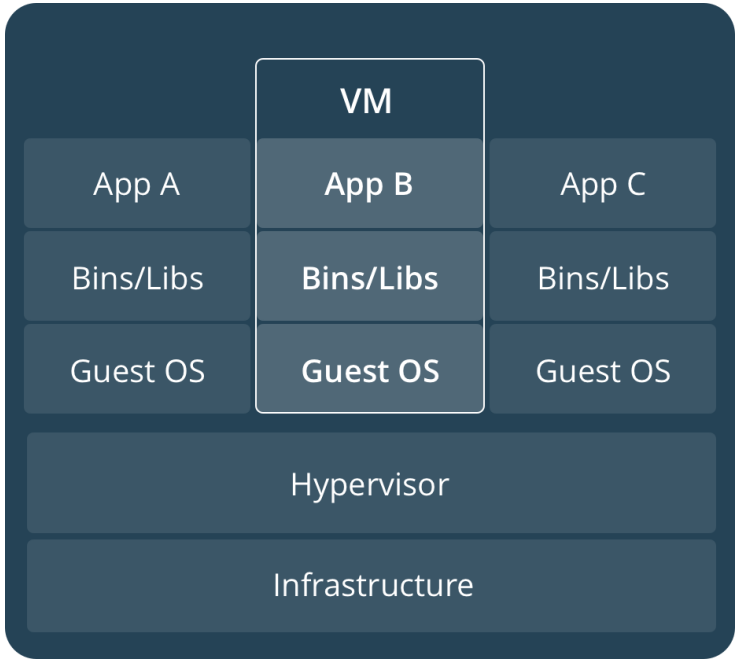
什么是容器?

- 容器是另外一种轻量级的虚拟化，容器是共用主机内核，利用内核的虚拟化技术隔离出一个独立的运行环境，拥有独立的一个文件系统，网络空间，进程空间视图等



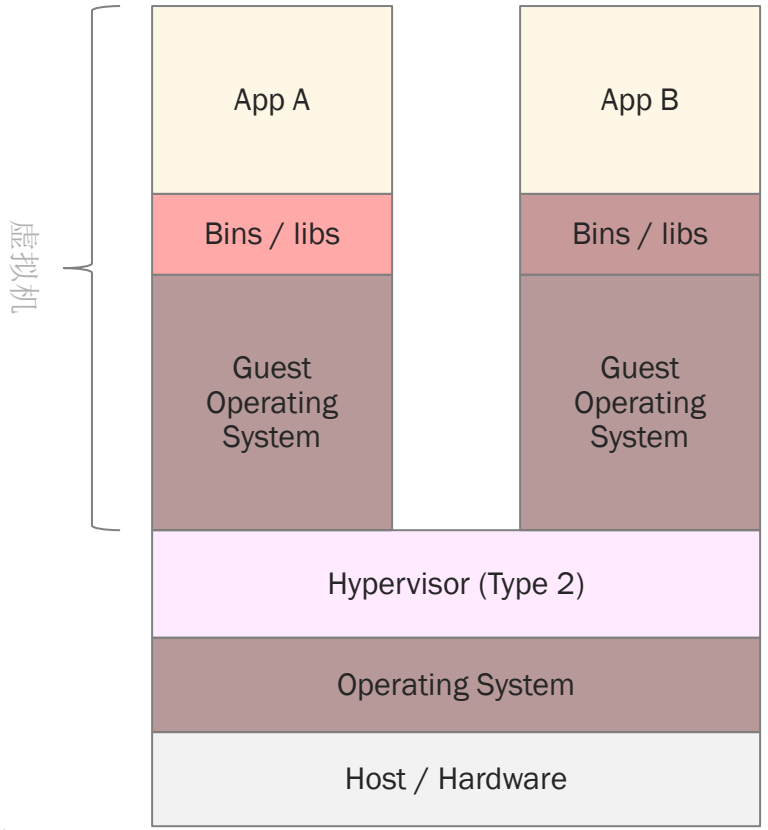
容器与虚机



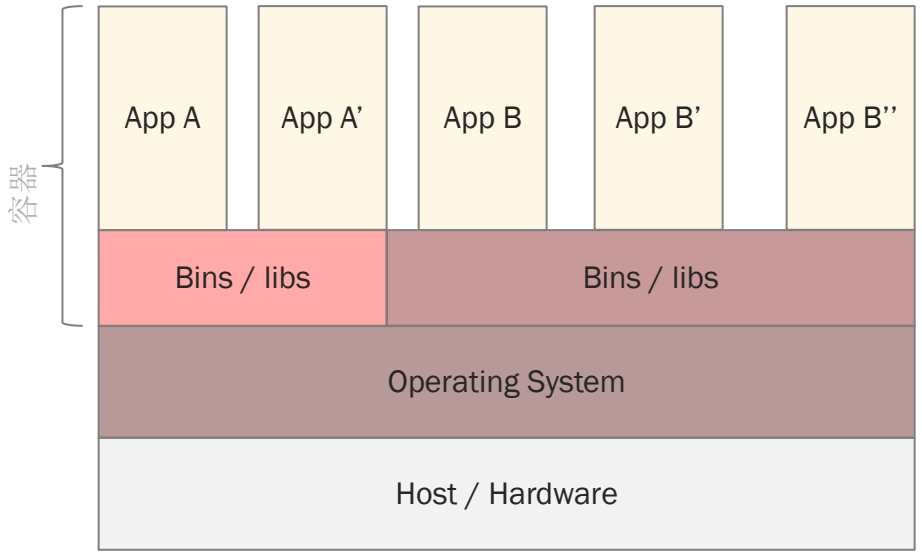


从虚拟化层看容器，轻量级、高性能是核心价值

- 容器是在Linux内核实现的轻量级资源隔离机制
- 虚拟机是操作系统级别的资源隔离，容器本质上是进程级的资源隔离



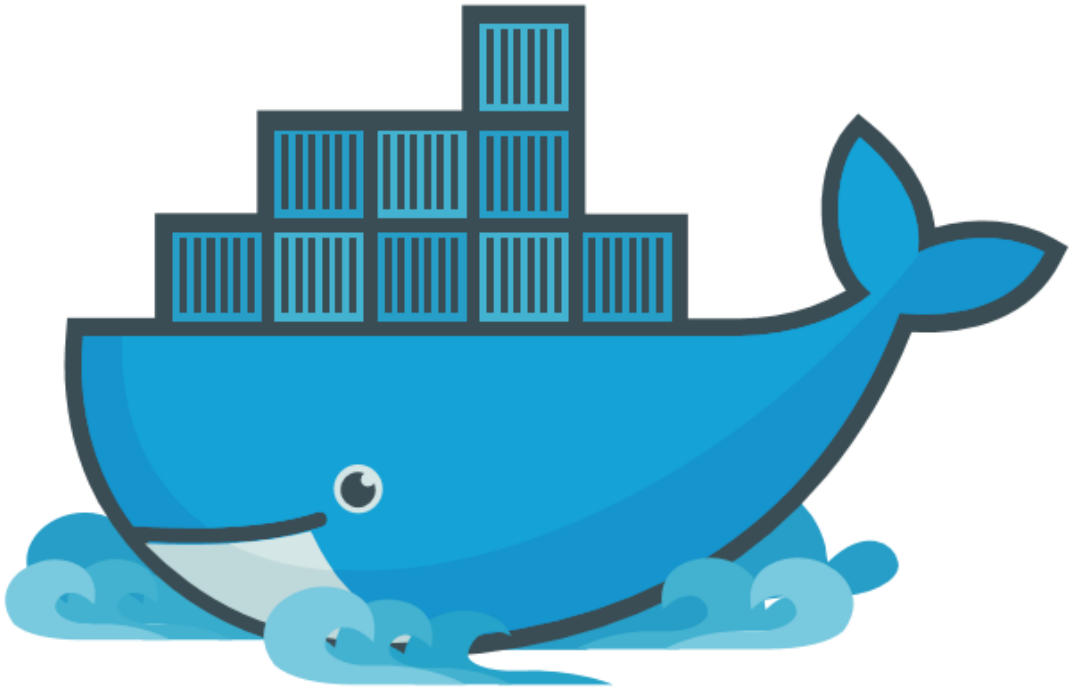
Containers are isolated, but share OS and, where appropriate, bins/libraries



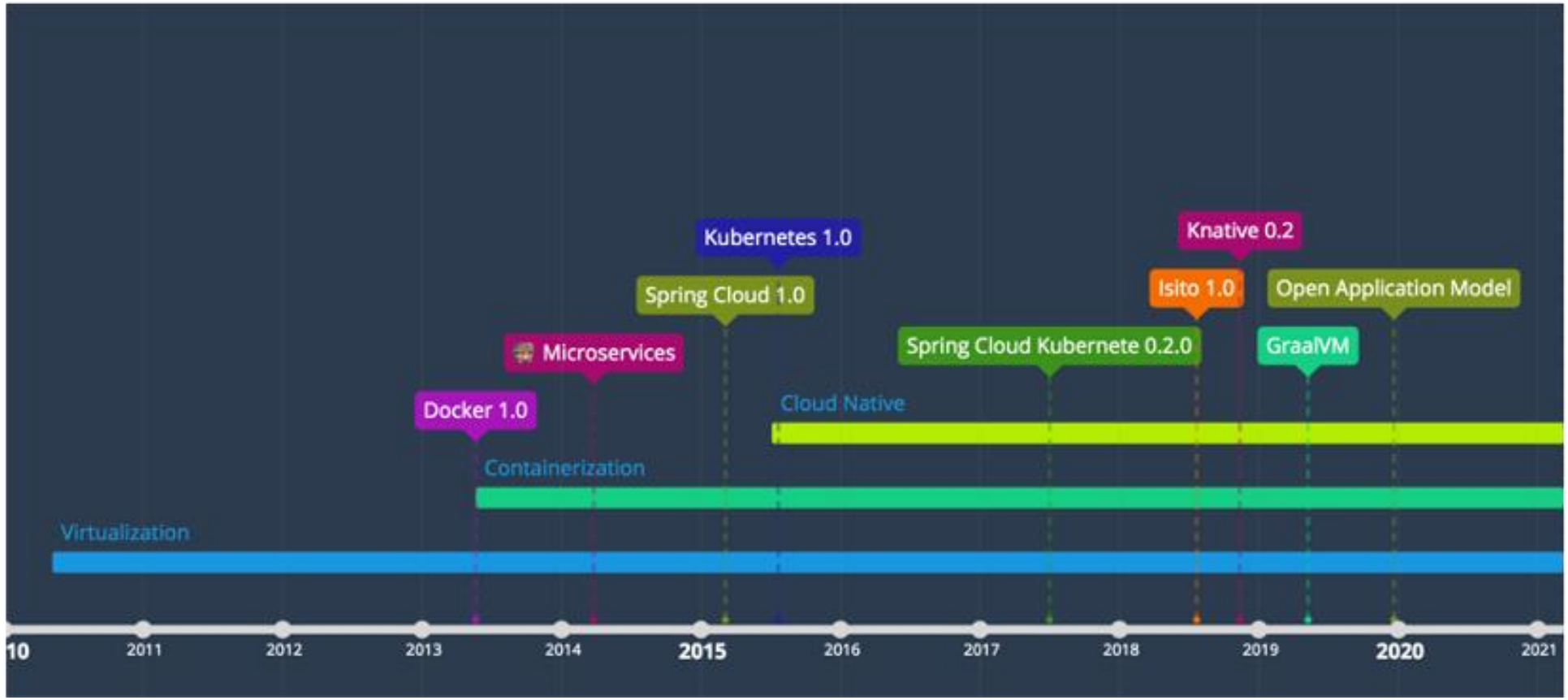
Docker

Build, Ship, Run

Docker is the world's leading software containerization platform



docker带来的技术改变

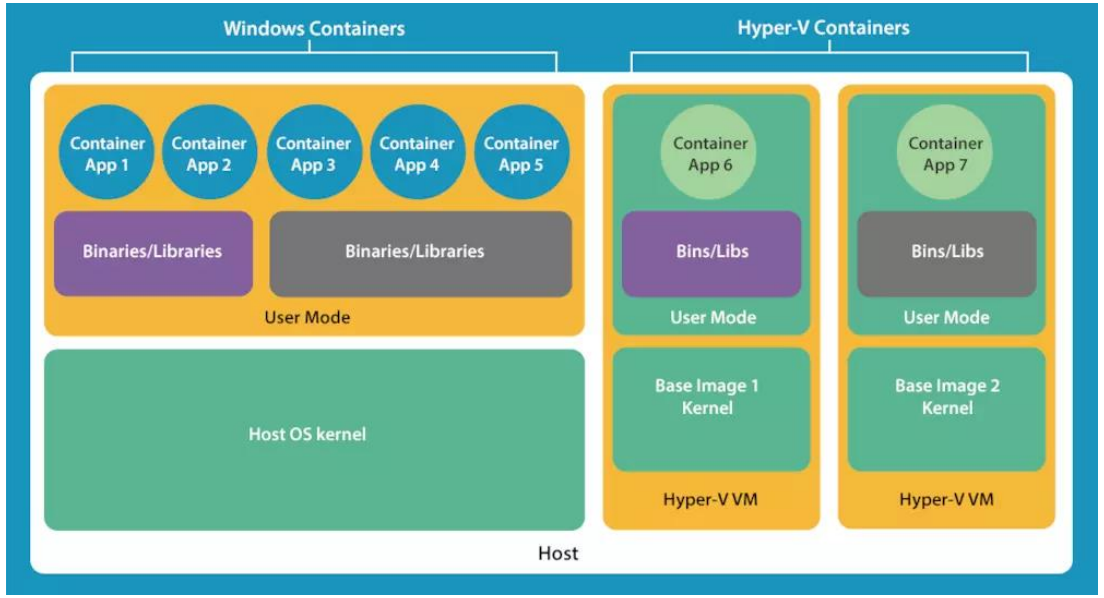


Docker可运行在以下操作系统

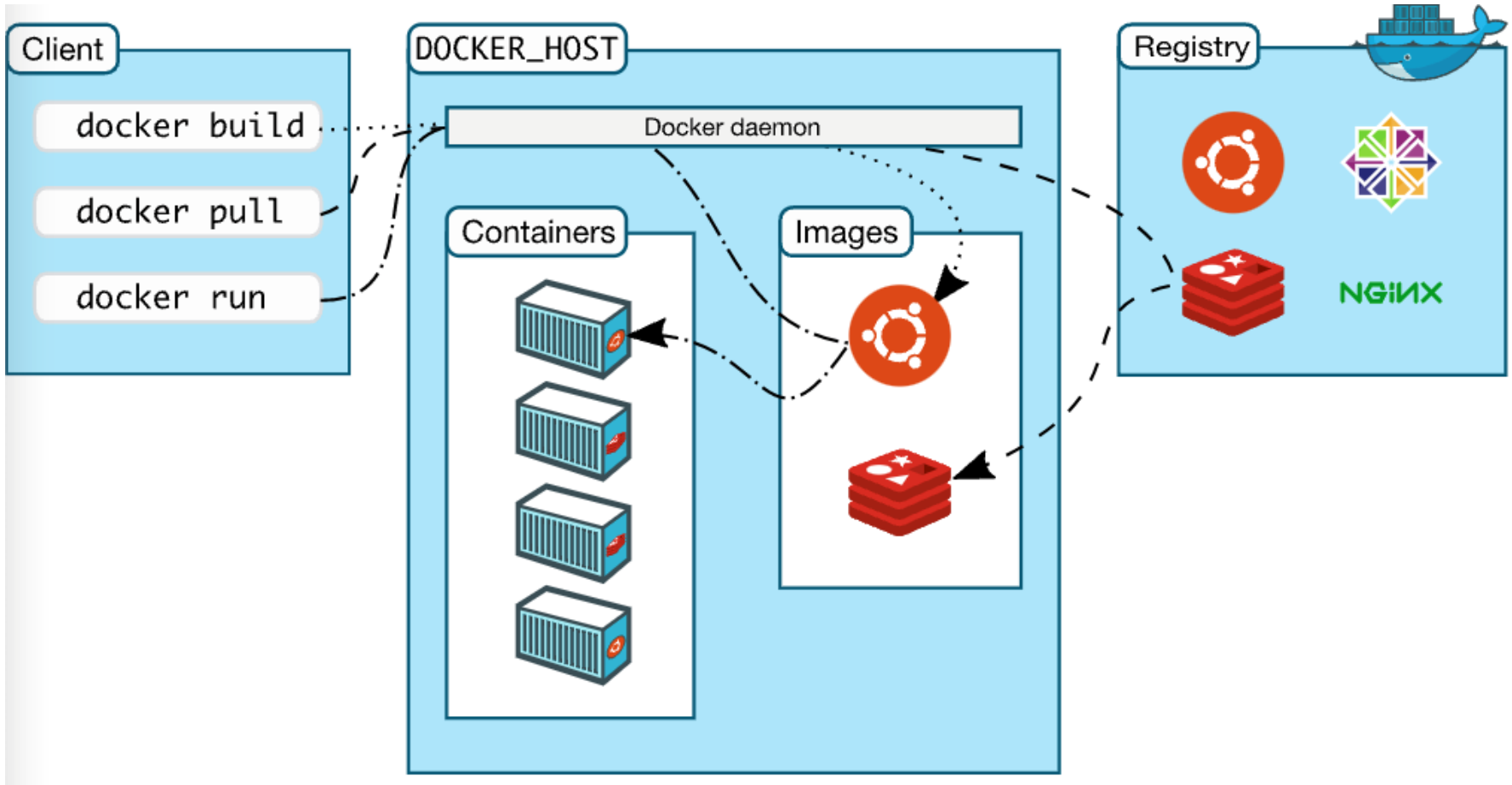
- Windows
- OS X
- Linux

Windows下的两类容器

- Windows Container
- linux container



Docker的三部分



docker 基本命令

- docker
- docker container --help
- docker --version
- docker version
- docker info
- docker image ls
- docker pull

搜索镜像

- docker search mongo

```
C:\Windows\system32>docker search mongo
NAME                                DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
mongo                               MongoDB document databases provide high avai…   9205      [OK]
mongo-express                       Web-based MongoDB admin interface, written w…   1239      [OK]
bitnami/mongodb                     Bitnami MongoDB Docker Image                    191                  [OK]
rapidfort/mongodb                   RapidFort optimized, hardened image for Mong…   14
circleci/mongo                      CircleCI images for MongoDB                     12                   [OK]
bitnami/mongodb-exporter                                                            8
bitnami/mongodb-sharded                                                             8
percona/percona-server-mongodb-operator   mongod image for PSMDB operator           4
percona/mongodb_exporter            A Prometheus exporter for MongoDB including …   3
rancher/mongodb-conf                                                                2
ibmcom/mongodb                                                                      1
ibmcom/mongodb-ppc64le                                                              1
litmuschaos/mongo                                                                   1
rancher/mongodb-config                                                              0
corpusops/mongo                     https://github.com/corpusops/docker-images/     0
ibmcom/mongo-c-driver-ppc64le       Docker image for mongo-c-driver-ppc64leDocke…   0
ibmcom/mongo-java-driver-ppc64le    Docker image for mongo-java-driver-ppc64le       0
litmuschaos/mongo-utils                                                             0
ibmcom/mongodb-s390x                                                                0
drud/mongodb                        Mongodb                                         0                    [OK]
ibmcom/mongodb-amd64                                                                0
ibmcom/mongodb-exporter-ppc64le                                                     0
kope/mongodb                                                                        0
rapidfort/mongodb-perfomance-test                                                   0
formio/mongotest                                                                    0
```

docker run命令

- docker run hello-world
- -d: 后台运行容器，并返回容器ID
- -i: 以交互模式运行容器，通常与 -t 同时使用
- -t: 为容器重新分配一个伪输入终端，通常与 -i 同时使用
- -p: 指定（发布）端口映射，格式为：主机(宿主)端口:容器端口
- -P: 随机端口映射，容器内部端口随机映射到主机的高端口
- --name="nginx-lb": 为容器指定一个名称
- -e username="ritchie": 设置环境变量
- --env-file=c:/temp1/t1.txt: 从指定文件读入环境变量
- --expose=2000-2002: 开放（暴露）一个端口或一组端口，用于指出容器内可能对外暴露的端口，如果加上-P则会建立外部端口映射
- --link my-mysql:server : 添加链接到另一个容器
- -v c:/temp1:/data: 绑定一个卷(volume)
- --rm 退出时自动删除容器

busybox镜像

- docker run --rm -it busybox sh

- cat /etc/hosts
- cat /proc/version
- uname -a

练习：gcc

- 拉取镜像命令：docker pull gcc
- docker run --rm -v C:\codec\gcc-hw:/hw -w /hw –it --name=server gcc
- docker run --rm -v C:\codec\gcc-hw:/hw -w /hw –it --name=client gcc

- 查询容器的IP:
- cat /etc/hosts

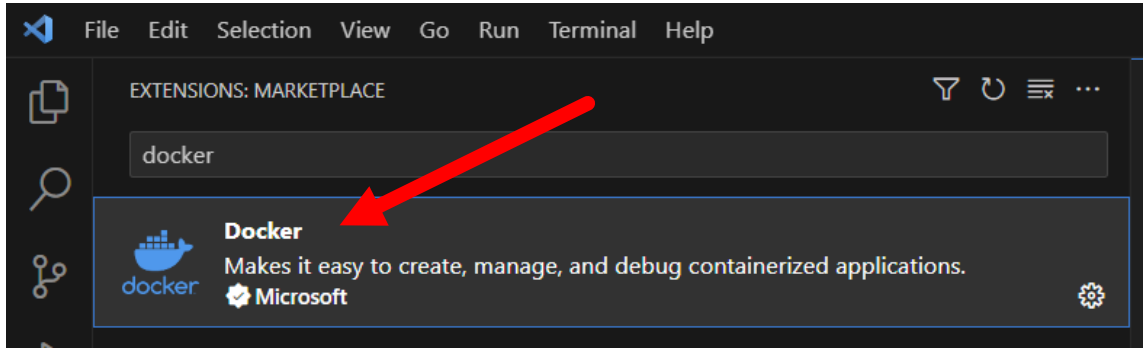
- cat /proc/version
- uname -a
- cat /etc/issue

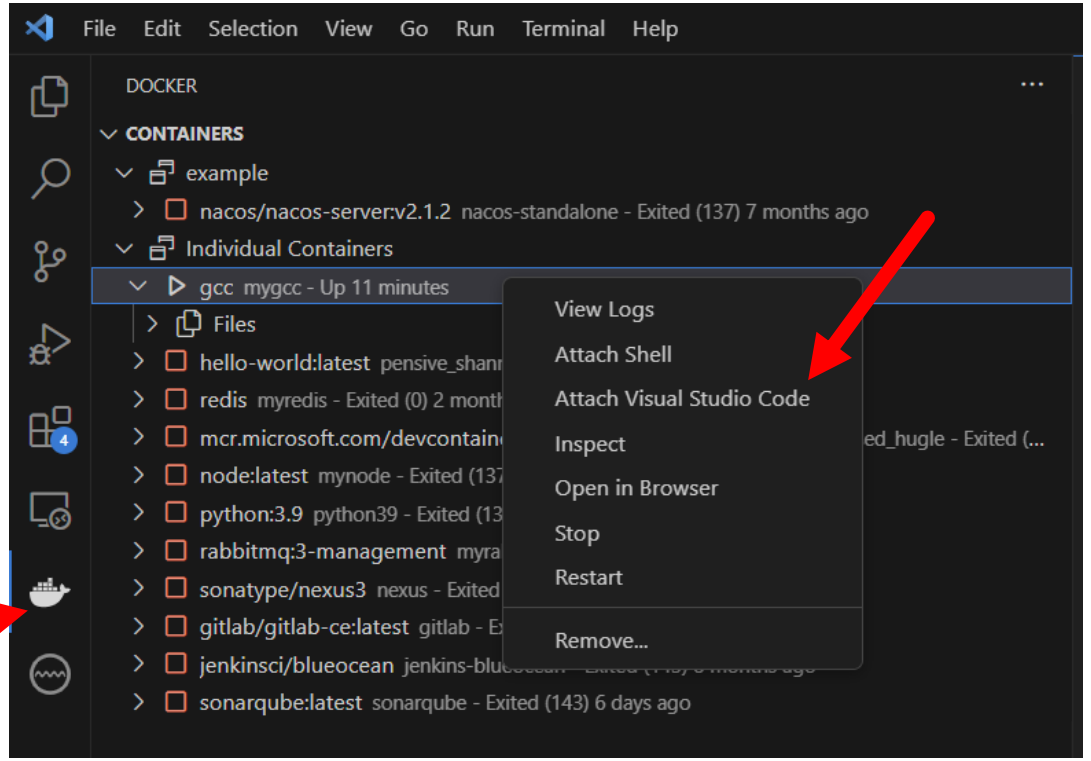
VSCode开发

- 通过安装插件，VSCode可用于C/C++、Java、Python、JS等语言的开发
- VSCode安装
  - ✓ https://code.visualstudio.com/

VSCode里安装Docker插件

- Docker插件用于管理和连接容器





安装VSCode插件

- C/C++
- C/C++ Extension Pack
- CMake
- CMake Tools
- CodeLLDB，用于调试用，如果无法在线安装，可以下载codelldb-x86_64-linux.vsix（ codelldb-x86_64-windows.vsix）文件后手工安装
- Gcov Viewer，用于覆盖率分析
- SonarLint，静态检查工具
- GoogleTest Adapter
- 可选插件： Makefile Tools

DEMO：基于VSCode的开发与测试

- C:\codec\demo_tdd_cxx-module

练习： MongoDB的启动和访问

- docker pull mongo
- docker run --name mymongo -d mongo
- docker run -it --link mymongo:mymongo2 --rm mongo mongo --host mymongo2
- docker run -it --link mymongo --rm mongo mongo --host mymongo

练习： Redis的启动和访问

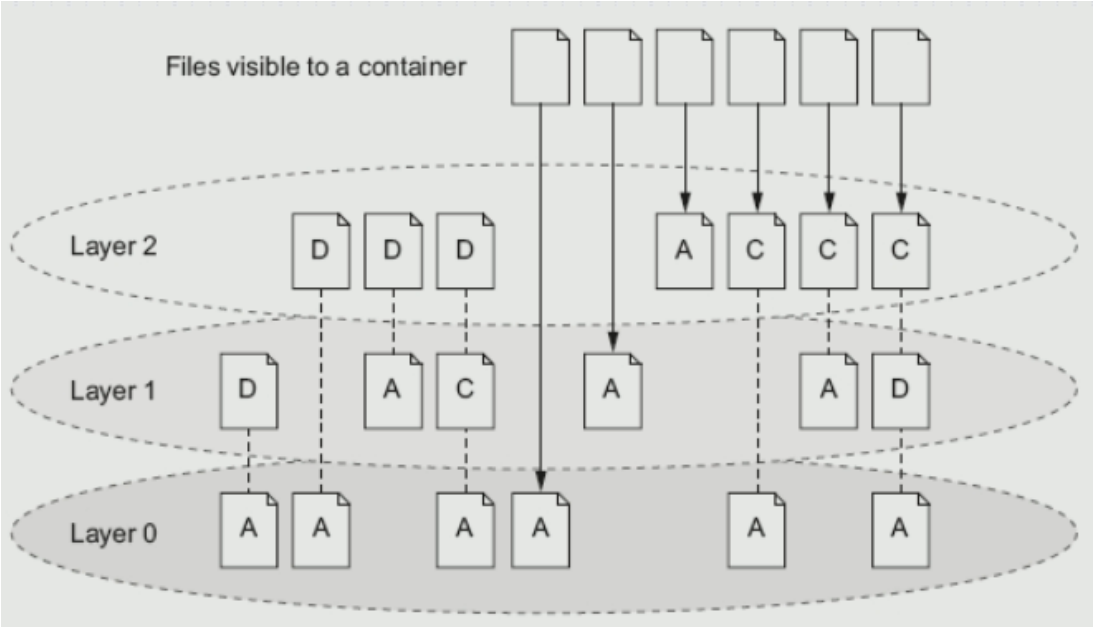
- docker pull redis
- docker run -d --name myredis redis
- docker run -it --rm --link myredis:myredis2 redis redis-cli -h myredis2
- docker run -it --rm --link myredis redis redis-cli -h myredis

练习： mysql的启动和访问

- docker pull mysql:5.7
- docker run --name my-mysql -e MYSQL_ROOT_PASSWORD=exampledb20 -d mysql:5.7
- docker run -it --rm --link my-mysql:server mysql:5.7 mysql –hserver -uroot -pexampledb20
- docker run -it --rm --link my-mysql mysql:5.7 mysql –hmy-mysql -uroot -pexampledb20

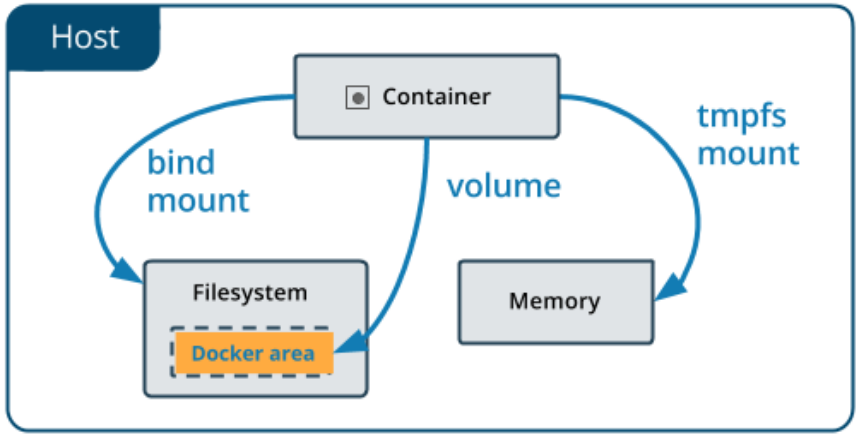
镜像分层

- 写时复制(COW, Copy-On- Write)
- docker history <image name> 查看镜像的层



三种应用的数据存储方式，数据卷(Volume)

- volumes：Docker管理宿主机文件系统的一部分，默认位于 /var/lib/docker/volumes 目录中
- bind mounts：意味着可以存储在宿主机系统的任意位置
- tmpfs (temporary file system)：挂载存储在宿主机系统的内存中，而不会写入宿主机的文件系统



bind mounts练习

- docker run -d --name=mynginx -p 8080:80 -v C:\nginx-1.23.1\dist:/usr/share/nginx/html nginx
- 访问：http://localhost:8080/#/

导出和导入容器镜像

- 针对容器导出，然后导入到镜像
  - ✓ docker export 1e560fca3906 > ubuntu.tar ，导出容器 1e560fca3906 快照到本地文件 ubuntu.tar
  - ✓ docker import ubuntu.tar test/ubuntu:v1 ， 将快照文件 ubuntu.tar 导入到镜像 test/ubuntu:v1
  - ✓ docker import http://example.com/exampleimage.tgz example/imagerepo
- 针对镜像备份
  - ✓ 备份：docker save -o 文件名.tar 镜像
  - ✓ 恢复：docker load -i 文件名.tar

dangling镜像

- dangling是一种特殊的，不会再被使用到的镜像，即无tag的镜像
- docker image ls -f dangling=true
- docker image prune -f

```
C:\Windows\system32>docker image ls -f dangling=true
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
redis        <none>    08502081bff6   16 months ago    105MB
```

Docker System命令

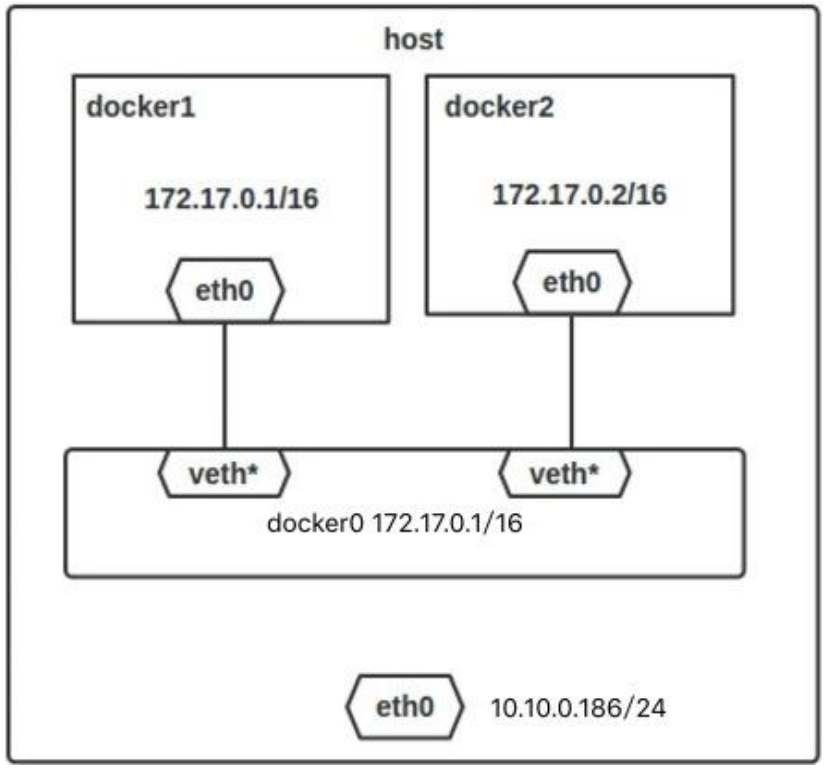
- docker system df命令，类似于Linux上的df命令，用于查看Docker的磁盘使用情况
- RECLAIMABLE指可回收的，对image，指的是没有被容器使用的镜像
- docker system events

```
C:\Windows\system32>docker system df
TYPE            TOTAL     ACTIVE    SIZE      RECLAIMABLE
Images          43        24        15.32GB   7.638GB (49%)
Containers      60        26        1.223GB   1.211GB (99%)
Local Volumes   155       8         19.65GB   14.8GB (75%)
Build Cache     249       0         1.156GB   1.156GB
```

```
C:\Windows\system32>docker system events
2022-10-31T18:01:23.149718100+08:00 volume destroy 318cd4b8106f851ed020a54b
2022-10-31T18:01:23.149848800+08:00 container destroy 2a54f4fbb70e446dfecdb
redis)
2022-10-31T18:01:46.306538100+08:00 image untag sha256:08502081bff61084d64f
1bff61084d64fc76f0f90ea39b89935cd071d9e12c5374ae191ff53c0)
2022-10-31T18:01:46.540533900+08:00 image delete sha256:08502081bff61084d64
81bff61084d64fc76f0f90ea39b89935cd071d9e12c5374ae191ff53c0)
```

容器网络

- none网络，--net=none
- host网络，--net=host
- bridge网络，--net=bridge ， docker0 的 linux bridge
- container模式，--net=container:NAME_or_ID



谢谢观看！