OPENCAS社团沙龙第一期

# DOCKER入门

## 大纲

- Docker简介
- Aliyun Registry
- Docker使用入门
- Docker实例
- Docker Compose

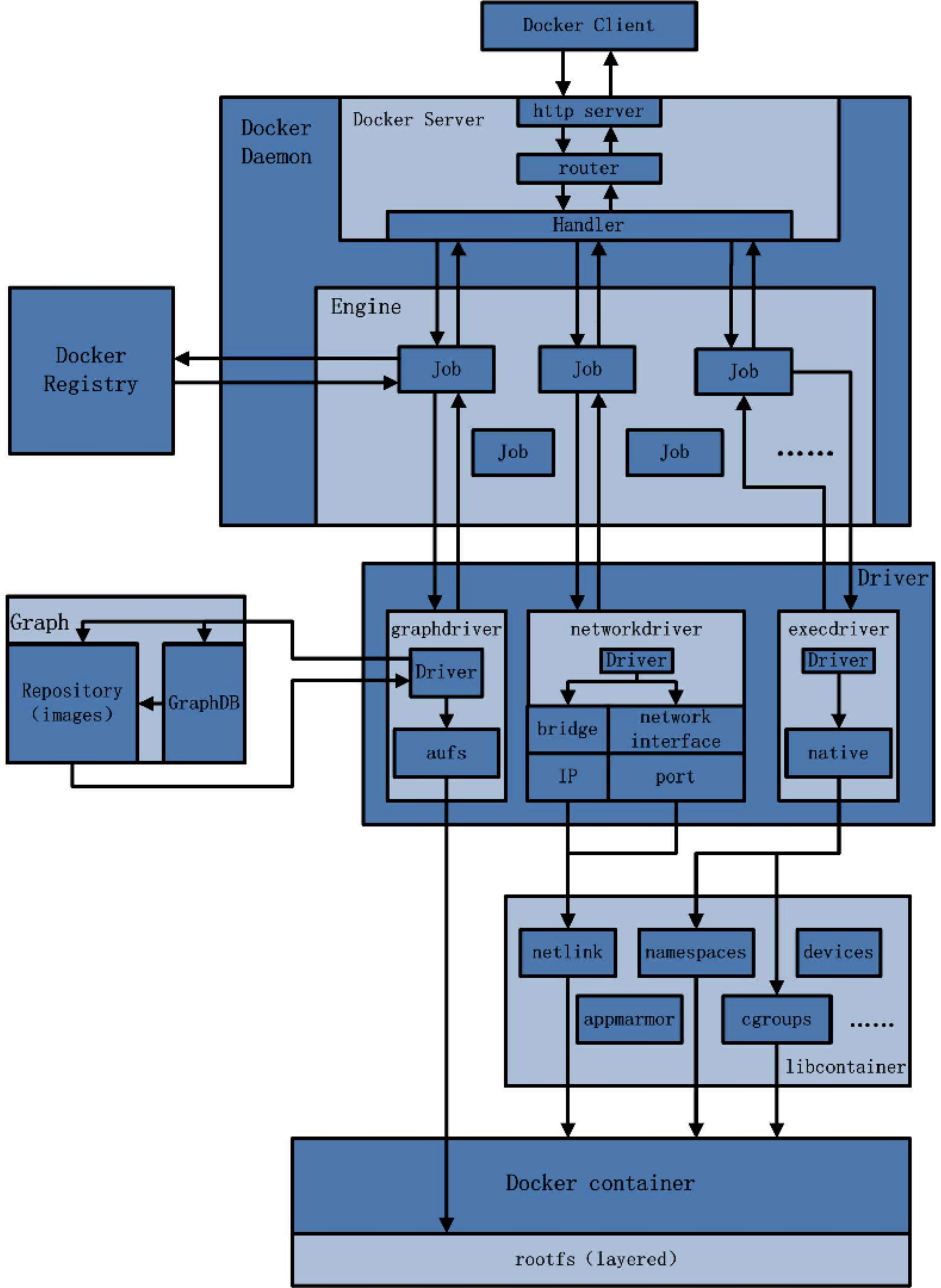
## DOCKER简介

## DOCKER简介

- Docker是一个容器，容器运行于操作系统内核之上的用户空间，是操作系统级的虚拟化技术
- 用途主要是缩短代码从开发、测试到部署、上线运行的周期，使得其具备可移植性，易于构建和协作开发
- 非常适用于微服务和分布式架构
- **个人使用**：开发和部署是一套环境，部署时不需要做重复配置，而且可以随配随删，同时是一个沙箱环境，大大减少工作量

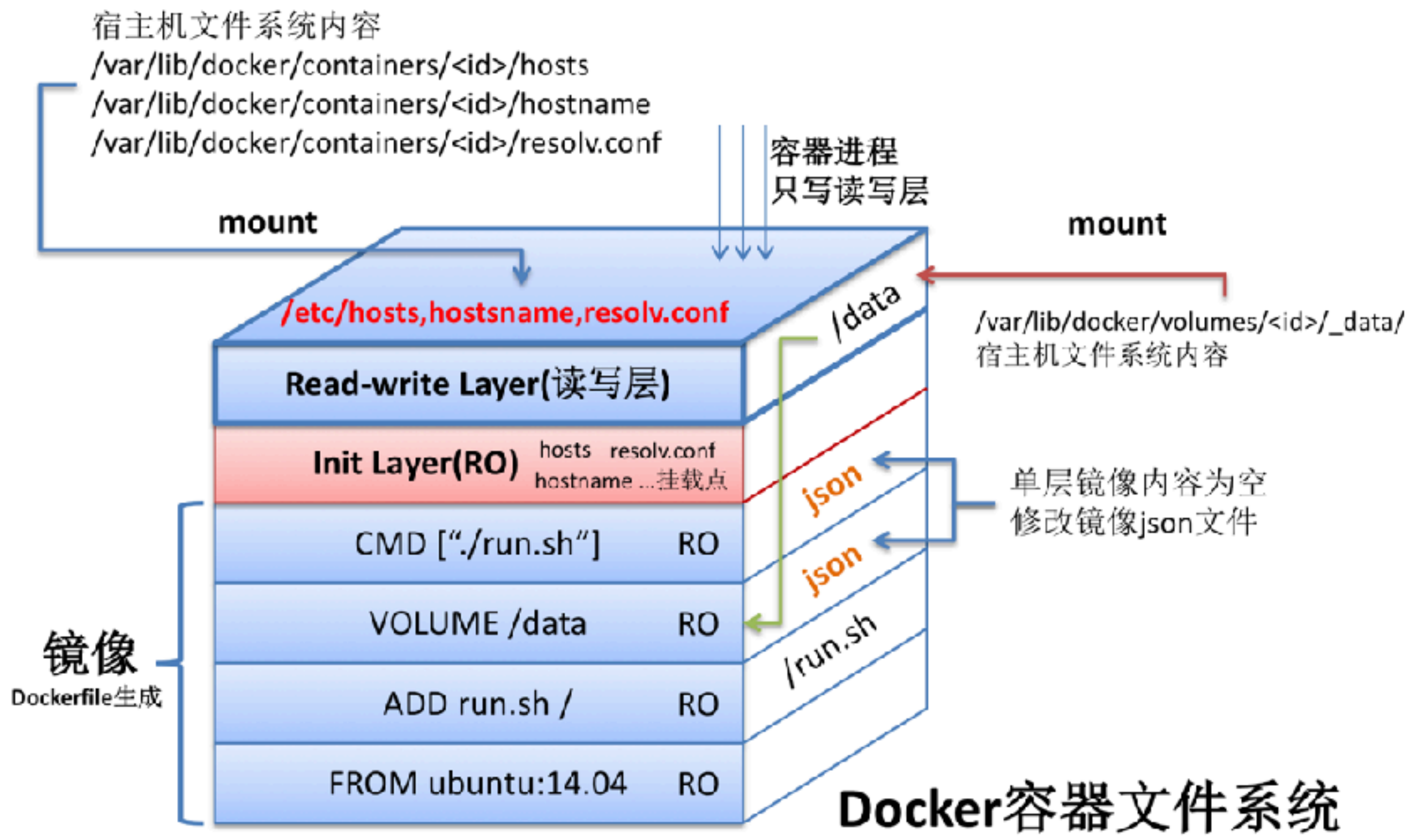
# DOCKER组成

- Docker由客户端和服务器组成，C/S架构，也称作Docker引擎
- Docker镜像，class
- Registry，类似GitHub的存在
- Docker容器，instance



# DOCKER技术组件

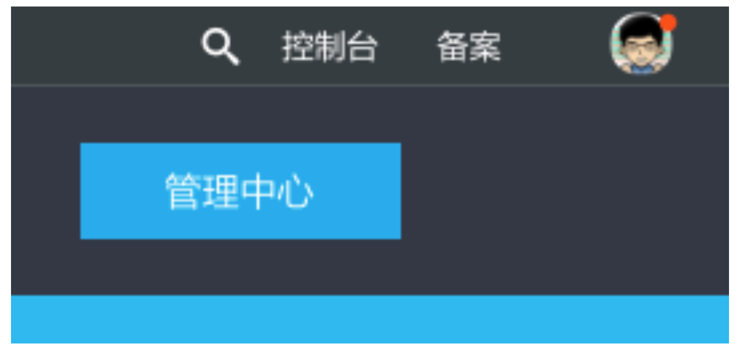
- 原生的Linux容器格式，称为libcontainer
- Linux内核的namespace，用于隔离文件系统、进程和网络
- 文件系统隔离、进程隔离、网络隔离
- 资源隔离和分组，使用cgroups
- 写时复制
- 日志与交互式shell

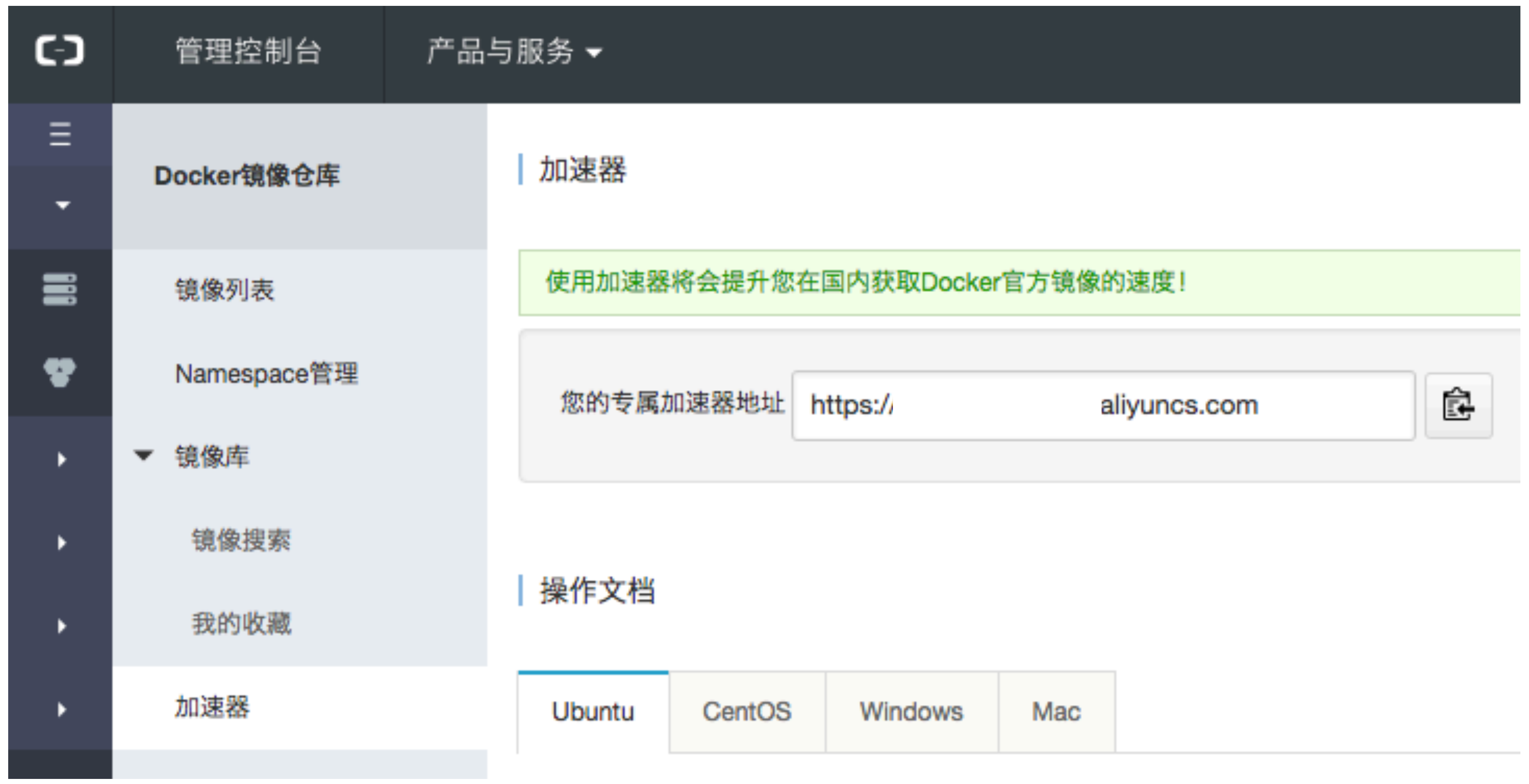


## ALIYUN REGISTRY

## ALIYUN REGISTRY

- https://dev.aliyun.com/



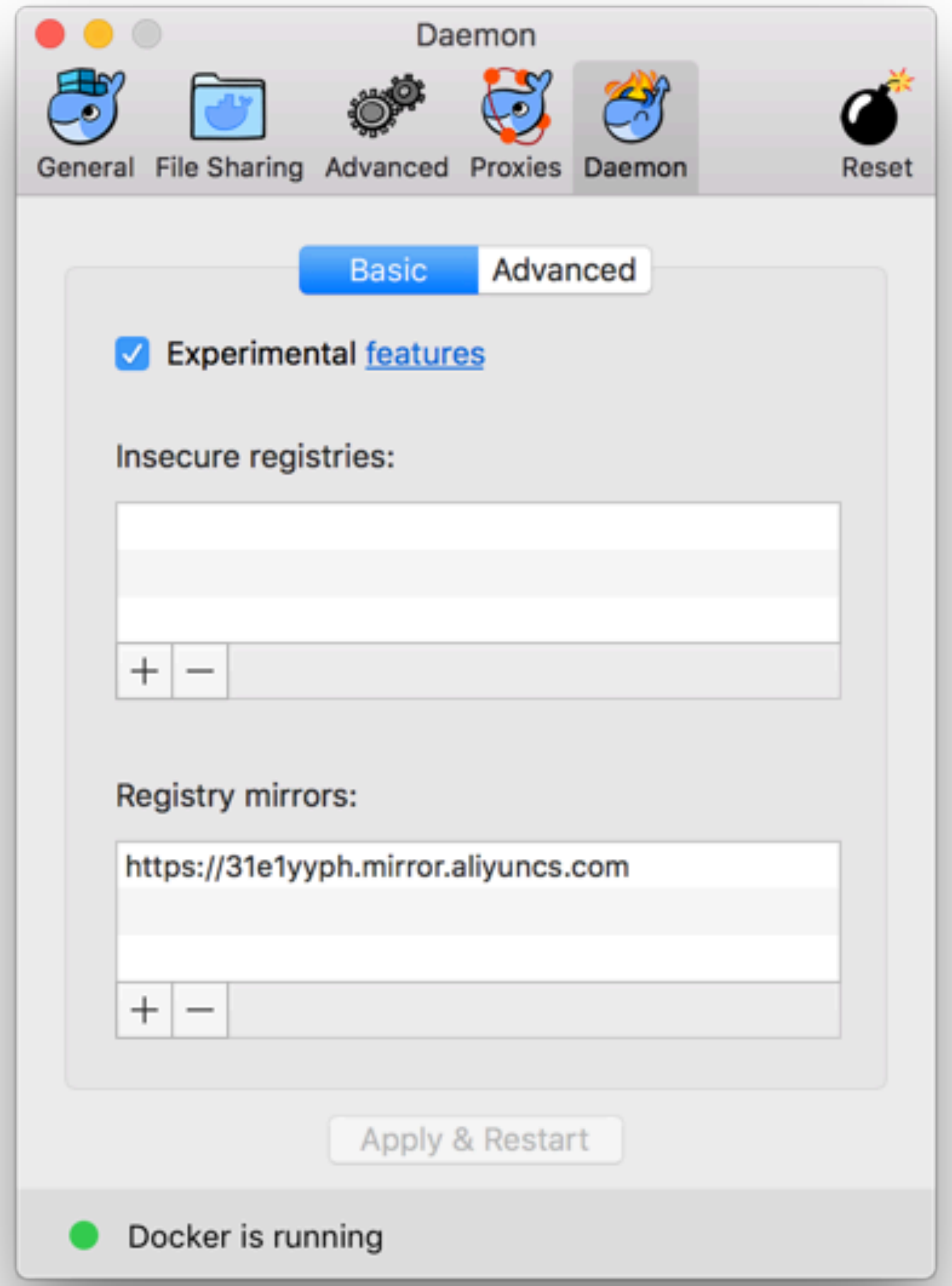


加速器

使用加速器将会提升您在国内获取Docker官方镜像的速度！

您的专属加速器地址 https:/ aliyuncs.com

操作文档



☑ Experimental features

Insecure registries:

Registry mirrors:
https://31e1yyph.mirror.aliyuncs.com

Docker is running

## DOCKER使用入门

## DOCKER基本命令

- `docker info`
- `docker [run/create] —name hello -i -t ubuntu /bin/bash`
- `docker ps -a`
- `docker stop [hello/id]`
- `docker start [hello/id]`
- `docker attach [hello/id]`
- `docker rm [hello/id]`

## DOCKER基本命令

- `docker run —name wow -d ubuntu /bin/sh -c “while true; do echo wow; sleep 1; done”`
- `docker ps -a`
- `docker logs [-f -t —tail 10] wow`
- `docker top wow`
- `docker stats wow`
- `docker exec -d wow touch /etc/config_file`
- `docker exec -t -i wow /bin/bash`
- `docker rm wow`

## DOCKER基本命令

- docker run —restart=[on-failure:5/always] —name overwatch -d ubuntu /bin/sh -c “while true; do echo hey; sleep 1; done”
- docker **inspect** overwatch
- docker rm overwatch

## DOCKER镜像使用

- `docker images`
- `docker pull ubuntu:12.04`
- `docker images`
- `docker run -it –name new ubuntu:12.04 /bin/bash`

  从Registry处拉取镜像，有两种类型的仓库：顶层仓库和用户仓库
  用户仓库命名方式 **用户名/仓库名**，顶层就直接 **仓库名**
- `docker search django`
- `docker rmi ubuntu:12.04`

## DOCKER构建镜像

- Dockerfile方式构建，docker commit方式不推荐使用
- mkdir static_web
- cd static_web
- touch Dockerfile

```
FROM ubuntu:latest
MAINTAINER imcmy <chenmingyi@iie.ac.cn>
RUN apt-get update
RUN apt-get install -y nginx
RUN echo ‘Hello World' \
    > /var/www/html/index.html
EXPOSE 80
```

- docker **build** -t <uname>/<repo> .
- docker run -it <uname>/<repo> /bin/bash

## DOCKER构建镜像

- docker images
- docker **history** <uname>/<repo>
- docker run -d [-p [ip:8080:]80/-P] –name static_web <uname>/<repo> nginx -g “daemon off;”
- docker ps -a
- docker **port** static_web
- curl localhost:8080

## DOCKER推送镜像

- `docker login –username=xxx registry.aliyuncs.com`
- `docker push registry.aliyuncs.com/xxx`

# DOCKERFILE 指令

- CMD [ “/bin/bash”, “-l” ]
- ENTRYPOINT [ “/usr/bin/nginx” ]
- ENV ANDORID_HOME=/opt/android/sdk
- WORKDIR $ANDROID_HOME
- USER nginx
- VOLUME [ “/opt/project” ]

DOCKER RUN的指令会覆盖CMD，然后传给ENTRYPOINT

# DOCKERFILE 指令

- `ADD src.cpp /opt/project`
- `COPY conf.d/ /etc/apahce2`
- `LABEL version=“1.0”`
- `ONBUILD ADD . /var/www/`

ADD的对象如果是一个TAR.GZ等压缩文件会自动解压
COPY则不会

## DOCKER NETWORKING

- `docker network create vnet1`
- `docker network inspect vnet1`
- `docker network ls`
- `docker run -d –net=vnet1 –name v1 -h v1 ubuntu`
- `docker run -d –net=vnet1 –name v2 -h v2 ubuntu`
- `docker network [connect/disconnect] vnet1 v2`

ping v1.vnet1

## DOCKER VOLUME

- Volume用于持久化数据及在容器间共享数据
- 两种方式，Dockerfile中的VOLUME和docker run的-v
- `docker run -it –name test -v /data ubuntu /bin/bash`
- `VOLUME /data`
- `docker inspect -f {{.Volumes}} test`
- `docker run -v /home/user/data:/data ubuntu ls /data`

## DOCKER VOLUME

- `docker run -it –volumes-from test ubuntu /bin/bash`
- 备份：`docker run --rm --volumes-from test -v $(pwd):/backup ubuntu tar cvf /backup/backup.tar /data`
- `docker rm -v test`

## DOCKER实例

## FETCHER

- `mkdir fetcher`
- `cd fetcher`
- `touch Dockerfile`

```
FROM ubuntu:latest
MAINTAINER imcmy <chenmingyi@iie.ac.cn>
ENV REFRESHED_AT 2017-04-16

RUN apt-get -qq update
RUN apt-get -qq install wget

VOLUME [ "/var/lib/tomcat7/webapps/" ]
WORKDIR /var/lib/tomcat7/webapps/

ENTRYPOINT [ "wget" ]
CMD [ "--help" ]
```

## FETCHER

- `docker build -t imcmy/fetcher .`
- `docker run -it –name fetcher imcmy/fetcher https://tomcat.apache.org/tomcat-7.0-doc/appdev/sample/sample.war`
- `docker inspect -f “{{ range .Mounts }}{{.}}{{end}}” fetcher`
- `ls -l`

## TOMCAT

- `mkdir tomcat`
- `cd tomcat`
- `touch Dockerfile`

```
FROM ubuntu:latest
MAINTAINER imcmy <chenmingyi@iie.ac.cn>
ENV REFRESHED_AT 2017-04-16

RUN apt-get -qq update
RUN apt-get -qq install tomcat7 default-jdk

ENV CATALINA_HOME /usr/share/tomcat7
ENV CATALINA_BASE /var/lib/tomcat7
ENV CATALINA_PID /var/run/tomcat7.pid
ENV CATALINA_SH /usr/share/tomcat7/bin/catalina.sh
ENV CATALINA_TMPDIR /tmp/tomcat7-tomcat7-tmp

RUN mkdir -p $CATALINA_TMPDIR

VOLUME [ "/var/lib/tomcat7/webapps/" ]

EXPOSE 8080

ENTRYPOINT [ "/usr/share/tomcat7/bin/catalina.sh", "run" ]
```

## TOMCAT

- `docker build -t imcmy/tomcat .`
- `docker run -d -P –name tomcat –volumes-from fetcher`
- `docker port tomcat`

## DOCKER COMPOSE

## DOCKER COMPOSE

- Docker Compose是一个用来定义和运行复杂应用的Docker工具
- 通过YAML文件定义一组要启动的容器，然后使用一条命令来启动应用，完成一切准备工作

## DOCKER COMPOSE

- `mkdir composeapp`
- `cd composeapp`
- `touch Dockerfile`

```
FROM python:3.5
MAINTAINER imcmy <chenmingyi@iie.ac.cn>
ENV REFRESHED_AT 2017-04-16

ADD . /composeapp

WORKDIR /composeapp

RUN pip install -r requirements.txt
```

## DOCKER COMPOSE

- touch app.py

```
from flask import Flask
from redis import Redis
import os

app = Flask(__name__)
redis = Redis(host="redis", port=6379)

@app.route('/')
def hello():
    redis.incr('hits')
    return 'Seen {0} times’.format(redis.get('hits'))

if __name__ == "__main__":
    app.run(host="0.0.0.0", debug=True)
```

# DOCKER COMPOSE

- `touch requirements.txt`

```
flask
redis
```

- ~~`docker build -t imcmy/composeapp .`~~
- `touch docker-compose.yml`
- `docker-compose up [-d]`
- `docker-compose [ps/logs/start/stop/rm]`

```
web:
  build: imcmy/composeapp
  image: imcmy/composeapp
  command: python app.py
  ports:
   - "5000:5000"
  volumes:
   - .:/composeapp
  links:
   - redis
redis:
  image: redis
```

OPENCAS社团沙龙第一期

## 谢谢