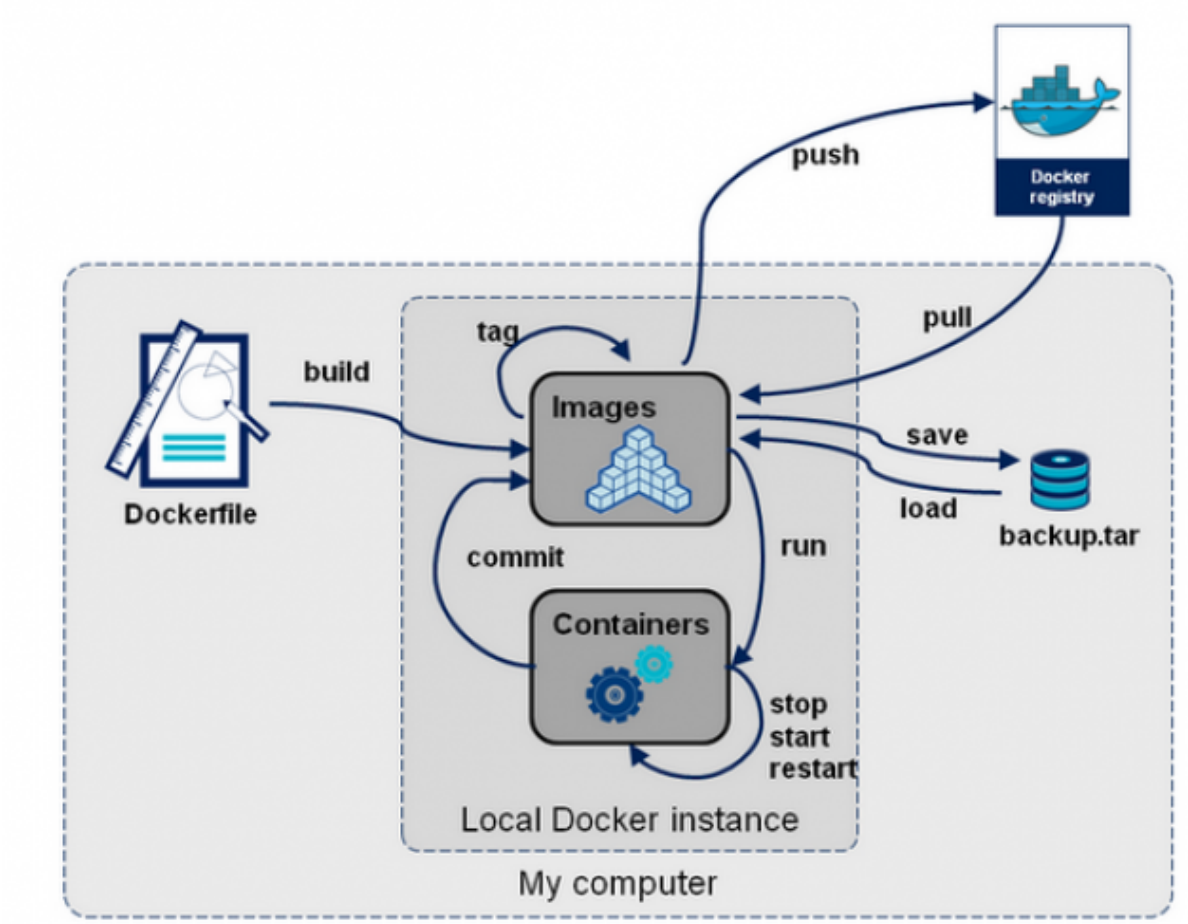
安装docker [Top]

docker阿里安装教程(推荐)

docker hub docker docs docker安装 非root账户使用docker 视频教程推荐



docker Linux通用安装 [Top]

```
sudo curl -sS https://get.docker.com/ | sh
```

测试安装是否成功。

```
$ docker run hello-world
```

如果机器有支持深度学习的GPU，新版docker可安装 Nvidia 对 docker 的软件支持[目前仅支持linux]:

```
# <a name="2">Add the package repositories</a><a style="float:right;text-decoration:none;" href="#index">[Top]</a>
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add -
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo tee /etc/apt/sources.list.d/nvidia-docker.list


sudo apt-get update && sudo apt-get install -y nvidia-container-toolkit
sudo systemctl restart docker
```

官方安装教程

注意：以前是安装nvidia-container-toolkit，现在官方又回退到sudo apt-get install -y nvidia-docker2

检验GPU是否可以使用 [Top]

```
docker run --gpus all --it nvidia/cuda:10.2-cudnn8-runtime-ubuntu18.04
```

--gpus all 则是使用所有 gpu。

docker 架构 [Top]

镜像（image）：

> 镜像是文件与meta data的集合
>
> 分层的，并且每一层都可以添加删除文件，从而形成新的镜像
>
> 不同的镜像可以共享相同的层（layout）
>
> 只读的

容器（container）：

> 通过image创建
>
> 在image 的最后一层上面再添加一层，这一层比较特殊，可读写。
>
> image负责存储和分发，container负责运行

仓库（repository）：

> 仓库（Repository）是集中存放镜像文件的场所。
>
> 仓库(Repository)和仓库注册服务器（Registry）是有区别的。仓库注册服务器上往往存放着多个仓库，每个仓库中又包含了多个镜像，每个镜像有不同的标签（tag）。

基本操作 [Top]

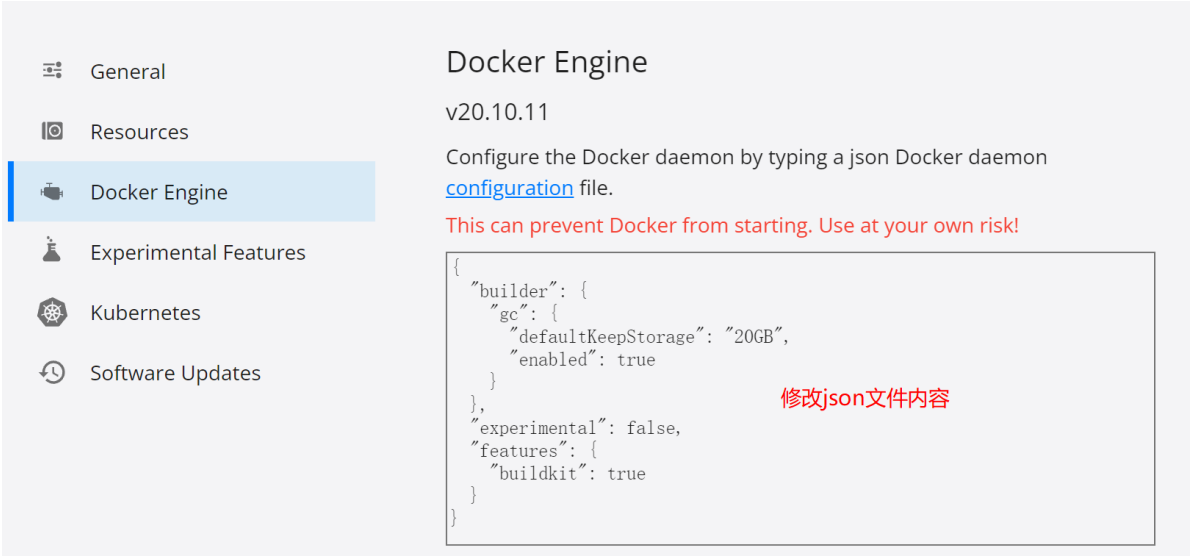
配置阿里云镜像加速 [Top]

1、介绍：https://www.aliyun.com/product/acr

2、进入管理控制台设置密码，开通

3、进入阿里云容器镜像服务，找到镜像加速器。可以看到配置镜像加速的方法。

windows系统配置镜像加速 [Top]




```
{
  "registry-mirrors": [
    "https://docker.mirrors.ustc.edu.cn",
    "http://hub-mirror.c.163.com",
  ],
  "insecure-registries": [],
  "debug": false,
  "experimental": false,
  "features": {
    "buildkit": true
  },
  "builder": {
    "gc": {
      "enabled": true,
      "defaultKeepStorage": "20GB"
    }
  }
}

```

"registry-mirrors" 加速镜像源；"experimental" 是否开启试验性功能

基础命令 [Top]

```
docker version # 显示 Docker 版本信息。
docker info # 显示 Docker 系统信息，包括镜像和容器数。
docker COMMAND --help # 帮助
```

镜像命令 [Top]

`docker search 镜像的名称` 搜索镜像，对应DockerHub仓库中的镜像

`docker images` 列出本地所有镜像

`docker pull [选项] [docker 镜像地址:标签]` 拉取镜像 docker pull mysql:5.7

`docker rmi -f 镜像id` 删除单个镜像

`docker rmi -f $(docker images -qa)` 删除全部镜像

阿里基础镜像

容器命令 [Top]

新建容器并启动 [Top]

`docker run --name hello -it ubuntu:18.04 bash`

> docker run 就是运行容器的命令,后面如果**只跟镜像**，那么就**执行镜像的默认命令然后退出。**Docker容器后台运行，就必须有一个前台进程，容器运行的命令如果不是那些一直挂起的命令，就会自动退出。
>
> --name：给容器指定一个名字
>
> -it：这是两个参数，一个是 -i：交互式操作，一个是 -t 终端。我们这里打算进入bash 执行一些命令并查看返回结果，因此我们需要交互式终端。
>
> -d：后台方式运行容器，并返回容器的id！
>
> --rm：这个参数是说容器退出后随之将其删除。
>
> -P：随机端口映射（大写）
>
> -p：指定端口映射（小写），一般可以有四种写法 hostPort:containerPort (常用)
>
> ubuntu:18.04：这是指用 ubuntu:18.04 镜像为基础来启动容器。 **如果只写ubuntu则是用ubuntu:latest作为基础镜像。**
>
> bash：放在镜像名后的是 **命令**，这里我们希望有个交互式 Shell，因此用的是 bash。
>
> 养成使用--name来指定容器名称和镜像写完整带tag的习惯。

`docker run --name myubuntu -itd ubuntu:18.04 bash` 创建myubuntu容器，并且后台运行（单纯使用-d不可以，必须使用-itd）。

退出容器 [Top]

`exit` 停止当前终端退出，可能会造成容器停止

`ctrl+P+Q` 仅退出，终端不关闭，容器后台运行。先摁ctrl P再摁Q

启动停止容器 [Top]

`docker start (容器id or 容器名)` 启动容器

`docker restart (容器id or 容器名)` 重启容器

`docker stop (容器id or 容器名)` 停止容器

`docker kill (容器id or 容器名)` 强制停止容器

查看容器 [Top]

`docker ps` 列出运行中的容器

`docker ps -a` 列出所有容器

`docker stats 容器id` 查看容器的cpu内存和网络状态

查看容器终端输出 [Top]

`docker logs -tf --tail 10 容器id` 查看最后10条的终端输出

查看容器/镜像的元数据 [Top]

`docker inspect 容器id /镜像id`

进入正在运行的容器 [Top]

`docker exec -it 容器id /bin/bash`

`docker attach 容器id`

> exec 是在容器中打开新的终端，并且可以启动新的进程
>
> attach 直接进入容器启动命令的终端，不会启动新的进程

从容器内拷贝文件到主机上 [Top]

`docker cp 容器id:容器内路径 目的主机路径`

从容器创建镜像 [Top]

`docker commit -m="提交的描述信息" -a="作者" 容器id 要创建的目标镜像名:[标签名]`

> 注意：commit的时候，repository的名字不能有大写，否则报错：invalid reference format
>
> 建议 commit 仅作为保留现场的手段，然后通过修改 dockerfile 构建镜像。

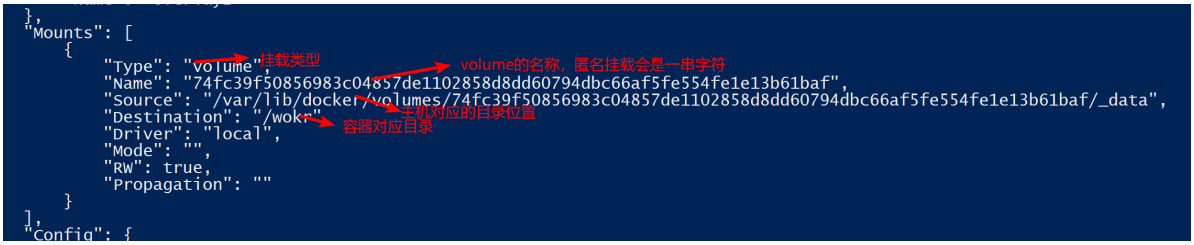
删除容器 [Top]

`docker rm 容器id` 删除指定容器，使用`-f`参数强制删除

`docker ps -a -q|xargs docker rm` 删除全部不在运行的容器，末尾使用`-f`参数强制删除全部容器

容器数据卷 [Top]

查看数据卷是否挂载成功 `docker inspect 容器id` ，查看Mounts的配置数据。



几种挂载方式 [Top]

- `bind mount` 直接把宿主机目录映射到容器内，适合挂代码目录和配置文件。可挂到多个容器上

`docker run -it -v 宿主机绝对路径目录:容器内目录 镜像名`

- `volume` 由容器创建和管理，创建在宿主机，所以删除容器不会丢失，官方推荐，更高效，Linux 文件系统，适合存储数据库数据。可挂到多个容器上。可以通过以下两种方式进行volume挂载。

`docker run -it -v 容器内目录 镜像名`

`docker run -it -v 数据卷名字:容器内目录 镜像名`

- `tmpfs mount` 适合存储临时文件，存宿主机内存中。不可多容器共享。

容器之间同步数据 [Top]

`docker run -it --name docker02 --volumes-from docker01 centos` 通过centos:latest镜像创建docker02容器，容器docker02使用和容器docker01相同的数据卷。

```
# 改变文件的读写权限

# ro: readonly

# rw: readwrite

# 指定容器对我们挂载出来的内容的读写权限

docker run -d -P --name nginx02 -v nginxconfig:/etc/nginx:ro nginx

docker run -d -P --name nginx02 -v nginxconfig:/etc/nginx:rw nginx
```

容器之间配置信息的传递，数据卷的生命周期一直持续到没有容器使用它为止。

存储在本机的文件则会一直保留！

DockerFile [Top]

基础知识：

1、每条保留字指令都必须为大写字母且后面要跟随至少一个参数

2、指令按照从上到下，顺序执行

3、# 表示注释

4、每条指令都会创建一个新的镜像层，并对镜像进行提交

```
1  FROM # 基础镜像，当前新镜像是基于哪个镜像的
2  MAINTAINER # 镜像维护者的姓名混合邮箱地址
3  RUN # 容器构建时需要运行的命令
4  EXPOSE # 当前容器对外保留出的端口
5  WORKDIR # 指定在创建容器后，终端默认登录的进来工作目录，一个落脚点
6  ENV # 用来在构建镜像过程中设置环境变量
7  ADD # 将宿主机目录下的文件拷贝进镜像且ADD命令会自动处理URL和解压tar压缩包
8  COPY # 类似ADD，拷贝文件和目录到镜像中！
9  VOLUME # 容器数据卷，用于数据保存和持久化工作
10 CMD # 指定一个容器启动时要运行的命令，dockerFile中可以有多个CMD指令，但只有最 后一个生效！
11 ENTRYPOINT # 指定一个容器启动时要运行的命令！和CMD一样
12 ONBUILD # 当构建一个被继承的DockerFile时运行命令，父镜像在被子镜像继承后，父镜像的ONBUILD被触发
```

> ADD 相比COPY除了复制功能，还提供两种附加功能**解压**和**下载**。

构建镜像 [Top]

```
docker build -f dockerfile地址 -t 新镜像名字:TAG .
```

最后的`.`是构建镜像的路径，不可以省掉。

如果dockerfile文件名为Dockerfile且在当前目录，则可以省略`-f dockerfile地址`

查看镜像变更历史 [Top]

```
docker history 镜像名
```

CMD 和 ENTRYPOINT 的区别 [Top]

CMD：Dockerfifile 中可以有多个CMD 指令，但只有最后一个生效，CMD 会被 docker run 之后的参数

替换!

ENTRYPOINT： docker run 之后的参数会被当做参数传递给 ENTRYPOINT，之后合并形成新的命令组合!

多容器通信 [Top]

镜像托管发布 [Top]

Docker hub [Top]

- 首先你要先 注册一个账号
- 命令行登录账号：
  ```
  docker login -u username
  ```
- "docker tag"命令重命名镜像，名字必须跟你注册账号一样，可以先通过`docker images`查看所有镜像
  ```
  docker tag image-id username/image_name:tag
  ```

对于阿里云需重命名为`仓库地址/命名空间/镜像仓库名:tag`，如registry.cn-beijing.aliyuncs.com/liansendocker/test_ai:1.0

- 推上去
`docker push username/image_name:tag`

阿里云托管 [Top]

- 阿里云镜像服务
- 创建个人实例，并且进入，创建命名空间，创建镜像仓库

备份和迁移数据 [Top]

如果你是用`bind mount`直接把宿主机的目录挂进去容器，那迁移数据很方便，直接复制目录就好了。如果你是用`volume`方式挂载的，由于数据是由容器创建和管理的，需要用特殊的方式把数据弄出来。

备份和导入 Volume 的流程 [Top]

备份：

- 运行一个 ubuntu 的容器，挂载需要备份的 volume 到容器，并且挂载宿主机目录到容器里的备份目录。
- 运行 tar 命令把数据压缩为一个文件
- 把备份文件复制到需要导入的机器

导入：

- 运行 ubuntu 容器，挂载容器的 volume，并且挂载宿主机备份文件所在目录到容器里
- 运行 tar 命令解压备份文件到指定目录

备份 MongoDB 数据演示 [Top]

- 运行一个 mongodb，创建一个名叫`mongo-data`的 volume 指向容器的 /data 目录
`docker run -p 27018:27017 --name mongo -v mongo-data:/data -d mongo:4.4`
- 运行一个 Ubuntu 的容器，挂载`mongo`容器的所有 volume，映射宿主机的 backup 目录到容器里面的 /backup 目录，然后运行 tar 命令把数据压缩打包
`docker run --rm --volumes-from mongo -v d:/backup:/backup ubuntu tar cvf /backup/backup.tar /data/`

最后你就可以拿着这个 backup.tar 文件去其他地方导入了。

恢复 Volume 数据演示 [Top]

- 运行一个 ubuntu 容器，挂载 mongo 容器的所有 volumes，然后读取 /backup 目录中的备份文件，解压到 /data/ 目录
`docker run --rm --volumes-from mongo -v d:/backup:/backup ubuntu bash -c "cd /data/ && tar xvf /backup/backup.tar --strip 1"`

> 注意，volumes-from 指定的是容器名字
> strip 1 表示解压时去掉前面1层目录，因为压缩时包含了绝对路径