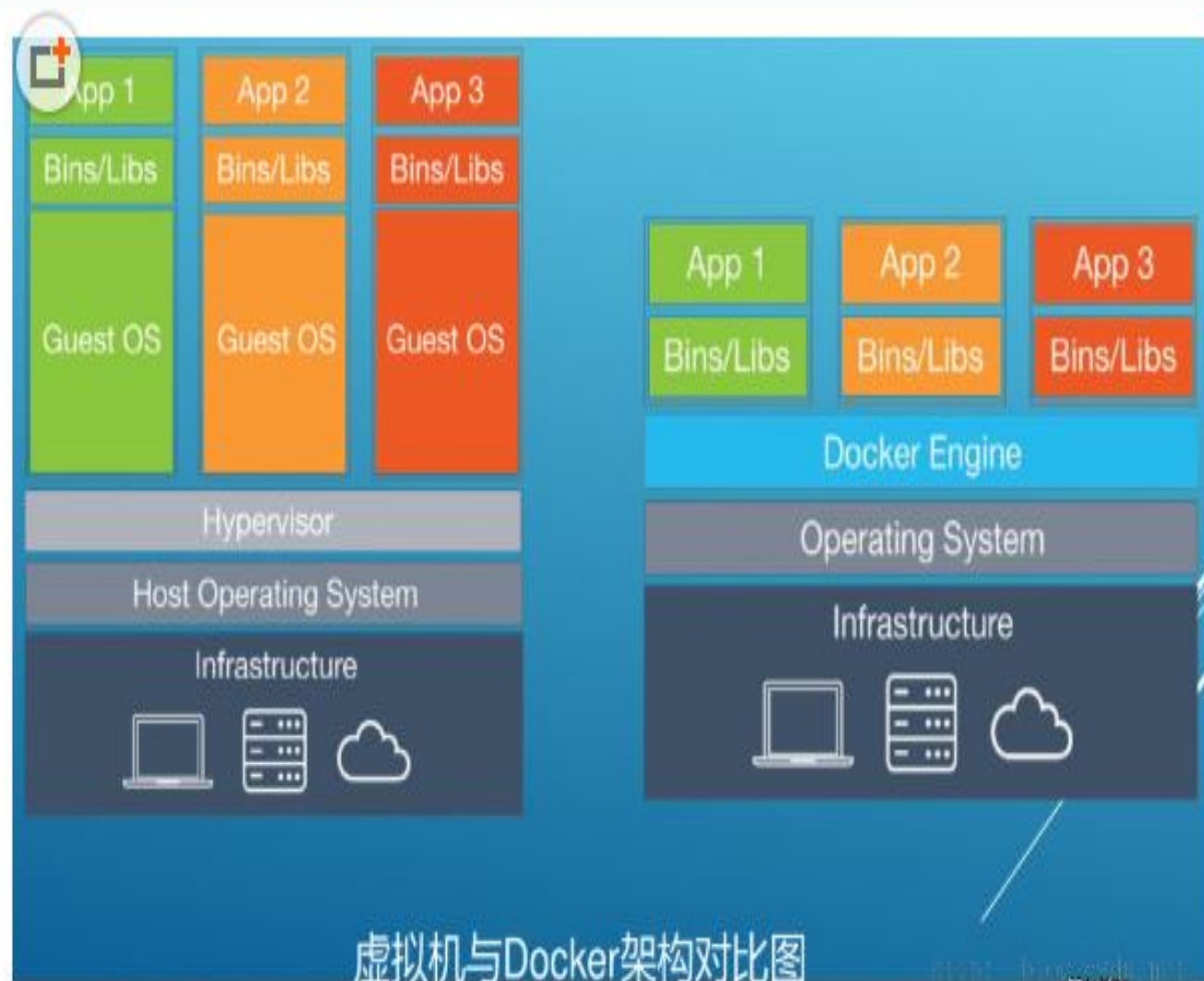




- 基于 Go 语言的云开源项目
- 简单来说, 就是将" 代码+环境" 打包在一起, 使应用达到跨平台无缝接轨使用
- 一次封装, 随处运行

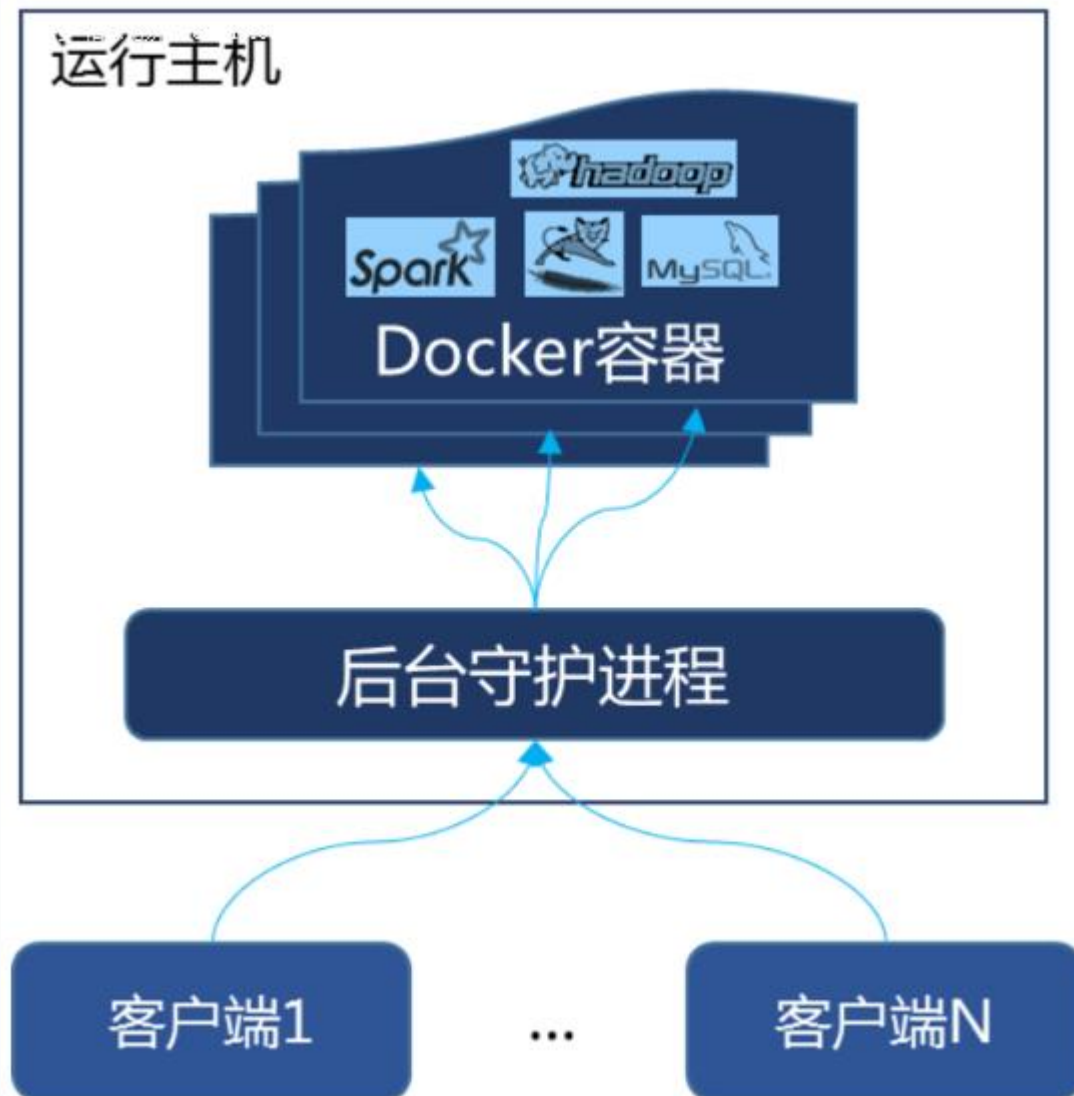
Docker 与传统虚拟化方式的不同之处：

1. 传统虚拟机技术是虚拟出一套硬件, 在其上运行一个完整操作系统, 在该系统上再运行所需应用进程
2. 而Docker容器内的应用进程直接运行于宿主的内核, 容器并没有自己的内核, 而且也没有进行硬件虚拟
3. 每个容器之间互相隔离, 每个容器有自己的文件系统, 容器之间进程不会相互影响, 能区分计算资源



Docker是如何工作的?

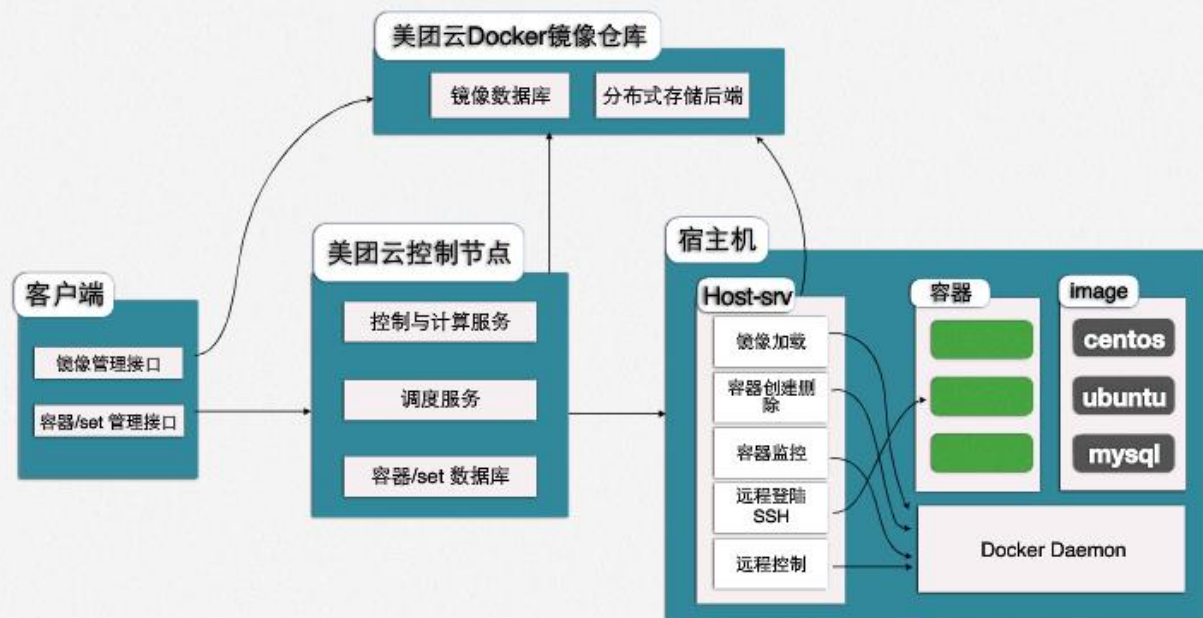
>> Docker是一个Client-Server结构的系统，Docker守护进程运行在主机上，然后通过Socket连接从客户端访问，守护进程从客户端接受命令并管理运行在主机上的容器。容器，是一个运行时环境，就是我们所说的鲸鱼背上的集装箱。



Why Docker

- 更轻量：基于容器的虚拟化，仅包含业务运行所需的runtime环境，CentOS/Ubuntu基础镜像仅170M；宿主机可部署100~1000个容器
- 更高效：无操作系统虚拟化开销
 - ✦ 计算：轻量，无额外开销
 - ✦ 存储：系统盘aufs/dm/overlayfs；数据盘volume
 - ✦ 网络：宿主机网络，NS隔离
- 更敏捷、更灵活：
 - ✦ 分层的存储和包管理，devops理念
 - ✦ 支持多种网络配置

美团云Docker框架



Docker应用实例

Docker基本组件

•镜像 (Image)

Docker 镜像 (Image) 就是一个只读的模板。镜像可以用来创建 Docker 容器，一个镜像可以创建很多容器。

•容器 (Container)


Docker 利用容器 (Container) 独立运行的一个或一组应用。容器是用镜像创建的运行实例。它可以被启动、开始、停止、删除。每个容器都是相互隔离的、保证安全的平台。可以把容器看做是一个简易版的 Linux 环境（包括root用户权限、进程空间、用户空间和网络空间等）和运行在其中的应用程序。

容器和镜像的关系可以类比 OOP 中的类和对象

- 容器 <-> 对象
- 镜像 <-> 类

•仓库 (Repository)

仓库 (Repository) 是集中存放镜像文件的场所。仓库分为公开仓库 (Public) 和私有仓库 (Private) 两种形式。最大的公开仓库是 Docker Hub(<https://hub.docker.com/>)。



Docker基本使用（centOS环境 为例）

- `yum search docker`
- `yum -y install docker`
- `systemctl start docker.service`



Docker常用命令：

1. 帮助命令

查看 Docker 版本号

`docker version`

查看当前 Docker 有关信息

`docker info`

帮助

`docker --help`

2. 镜像命令

- 查看本地所有镜像

docker images [OPTION]

- REPOSITORY: 镜像仓库源
- TAG: 镜像的标签,默认是latest
- IMAGE ID: 镜像 id
- CREATED: 镜像创建时间
- SIZE: 镜像大小

options:

- a: 列出本地所有的镜像 (含中间映像层)
- q: 只显示镜像 id (IMAGE ID)
- digests: 显示镜像的摘要信息
- no-trunc: 显示完整的镜像信息

```
ssf@H5QJJH2:mypipe$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mypipe	1.0	dc7b16e40855	7 seconds ago	1.48GB
centos	latest	470671670cac	4 weeks ago	237MB
ubuntu	16.04	96da9143fb18	4 weeks ago	124MB
ensemblorg/ensembl-vep	latest	cb22f80fced9	2 months ago	765MB
griffithlab/pvactools	latest	9266cd07ce5d	2 months ago	4.82GB
agentejo/cockpit	latest	a1184050dfce	9 months ago	482MB
topgenpipetest	latest	3d19d37eaabe	9 months ago	322MB
pipev001	latest	d17e949c65f6	11 months ago	4.35GB
ubuntu	latest	94e814e2efa8	11 months ago	88.9MB
trusight-oncology-500	1.3.1.3	2f0996374f88	14 months ago	1.95GB

• 搜索镜像

docker search image [OPTION]

options:

- s: 列出 stars 数不小于指定值
- no-trunc: 显示完整镜像信息
- automated: 只列出 automated build 类型的镜像

```
ssf@H5QJJH2:mypipe$ docker search centos
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
centos	The official build of CentOS.	5824	[OK]	
ansible/centos7-ansible	Ansible on Centos7	128		[OK]
jdeathe/centos-ssh	OpenSSH / Supervisor / EPEL/IUS/SCL Repos - ...	114		[OK]
consol/centos-xfce-vnc	Centos container with "headless" VNC session...	109		[OK]
centos/mysql-57-centos7	MySQL 5.7 SQL database server	68		
imagine10255/centos6-lamp-php56	centos6-lamp-php56	58		[OK]
tutum/centos	Simple CentOS docker image with SSH access	45		
centos/postgresql-96-centos7	PostgreSQL is an advanced Object-Relational ...	40		
kinogmt/centos-ssh	CentOS with SSH	29		[OK]
pivotaldata/centos-gpdb-dev	CentOS image for GPDB development. Tag names...	10		
guyton/centos6	From official centos6 container with full up...	9		[OK]
drecom/centos-ruby	centos ruby	6		[OK]
centos/tools	Docker image that has systems administration...	5		[OK]
darksheer/centos	Base Centos Image -- Updated hourly	3		[OK]
mamohr/centos-java	Oracle Java 8 Docker image based on Centos 7	3		[OK]

- 下拉镜像

`docker pull 镜像[:标签]`

- 删除镜像

`docker rmi [-f] image[:TAG]`

如果不写标签, 默认删除的是 latest

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

3. 容器命令

- 启动容器

`docker run [OPTIONS] IMAGE [COMMAND] [ARGS]`

- 本地有新建运行
- 本地没有，先去dockerhub下载，再运行

options:

- name: 为容器指定一个
- d: 后台运行容器, 并返回容器 id, 即启动守护式容器
- 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: 随机端口映射
- p: 指定端口映射

```
ssf@H5QJJH2:mypipe$ docker run -it --name pipe01 mypipe:1.0
root@0dc8369213f4:/data/ngs# ll
total 12
drwxr-xr-x 1 root root 4096 Feb 18 07:05 ./
drwxr-xr-x 1 root root 4096 Feb 18 06:54 ../
drwxr-xr-x 1 root root 4096 Feb 18 07:08 soft/
```

- 列出当前正在运行的容器

`docker ps [OPTIONS]`

options:

- a: 列出当前所有正在运行的容器 + 历史上运行过的容器
- l: 显示最近创建的容器
- n: 显示最近 n 个创建的容器
- q: 静默模式, 只显示容器编号

- 退出容器

`exit` 离开同时关闭 container

`CTRL + P + Q` 离开, 但不关闭容器

- 删除容器

`docker rm <container_id>`

- 容器必须是非运行状态下才可以删除

- 启动守护式容器

`docker run -d image[:TAG]`

- Docker容器后台运行,就必须有一个前台进程.

- 查看容器日志

`docker logs [OPTION] <container_id>`

```
ssf@H5QJJH2:mypipe$ docker run -d centos
576e196464e78bb21b7b0e718d50e95299f4643bf4ffb91c2b4da62a084b3426
ssf@H5QJJH2:mypipe$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
ssf@H5QJJH2:mypipe$ docker run -d centos /bin/sh -c "while true;do echo hello world;sleep 2;done"
77e6d950a5cbb8e3af7033bfed341ab3b1152404f6c18fde9defbf4f529dff9e
ssf@H5QJJH2:mypipe$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
77e6d950a5cb        centos              "/bin/sh -c 'while t..." 2 seconds ago       Up 1 second         dazdling_payne
ssf@H5QJJH2:mypipe$ docker logs -t --tail 3 77e6d950a5cb
2020-02-18T08:29:19.037717315Z hello world
2020-02-18T08:29:21.039992637Z hello world
2020-02-18T08:29:23.042263855Z hello world
```

- 进入正在运行的容器

`docker exec -it <container_id> ls -l`

`docker attach <container_id>`

exec 和 attach 的区别:

- exec: 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

```
ssf@H5QJJH2:mypipe$ docker exec -it 77e6d950a5cb ls -l
total 48
lrwxrwxrwx   1 root root    7 May 11  2019 bin -> usr/bin
drwxr-xr-x   5 root root  340 Feb 18  08:28 dev
drwxr-xr-x   1 root root 4096 Feb 18  08:28 etc
drwxr-xr-x   2 root root 4096 May 11  2019 home
lrwxrwxrwx   1 root root    7 May 11  2019 lib -> usr/lib
lrwxrwxrwx   1 root root    9 May 11  2019 lib64 -> usr/lib64
drwx-----  2 root root 4096 Jan 13 21:48 lost+found
drwxr-xr-x   2 root root 4096 May 11  2019 media
drwxr-xr-x   2 root root 4096 May 11  2019 mnt
drwxr-xr-x   2 root root 4096 May 11  2019 opt
dr-xr-xr-x 366 root root    0 Feb 18  08:28 proc
dr-xr-xr-x   2 root root 4096 Jan 13 21:49 root
drwxr-xr-x  11 root root 4096 Jan 13 21:49 run
lrwxrwxrwx   1 root root    8 May 11  2019/sbin -> usr/sbin
drwxr-xr-x   2 root root 4096 May 11  2019 srv
dr-xr-xr-x  13 root root    0 Mar 13  2019 sys
drwxrwxrwt   7 root root 4096 Jan 13 21:49 tmp
drwxr-xr-x  12 root root 4096 Jan 13 21:49 usr
drwxr-xr-x  20 root root 4096 Jan 13 21:49 var
ssf@H5QJJH2:mypipe$ docker attach 77e6d950a5cb
hello world
hello world
hello world
hello world
```

• 容器数据卷

- Docker 容器产生的数据, 如果不 docker commit 那么容器删除后, 数据也就丢失了, 而容器数据卷就是为了容器数据的持久性。

直接命令行创建:


`docker run -it -v /宿主机绝对路径:/容器内目录:权限 <image_name>`

- 在一次 run 中多次使用可以挂载多个数据卷
- `docker inspect`可查看数据卷是否挂载成功 (返回json格式数据)

使用 Dockerfile 添加:

- 在 Dockerfile 中使用 VOLUME 指令来给镜像添加一个或多个数据卷
- 生成容器后, 容器内的卷目录地址会生成主机对应的默认地址, 可用 `docker inspect` 查看

```
1 # Dockerfile example
2 FROM centos
3 VOLUME ["/dataVolumeContainer1", "/dataVolumeContainer1"]
4 CMD /bin/bash
```



DockerFile解析

- Dockerfile 是用来构件 Docker 镜像的构建文件, 是由一系列命令和参数构成的脚本

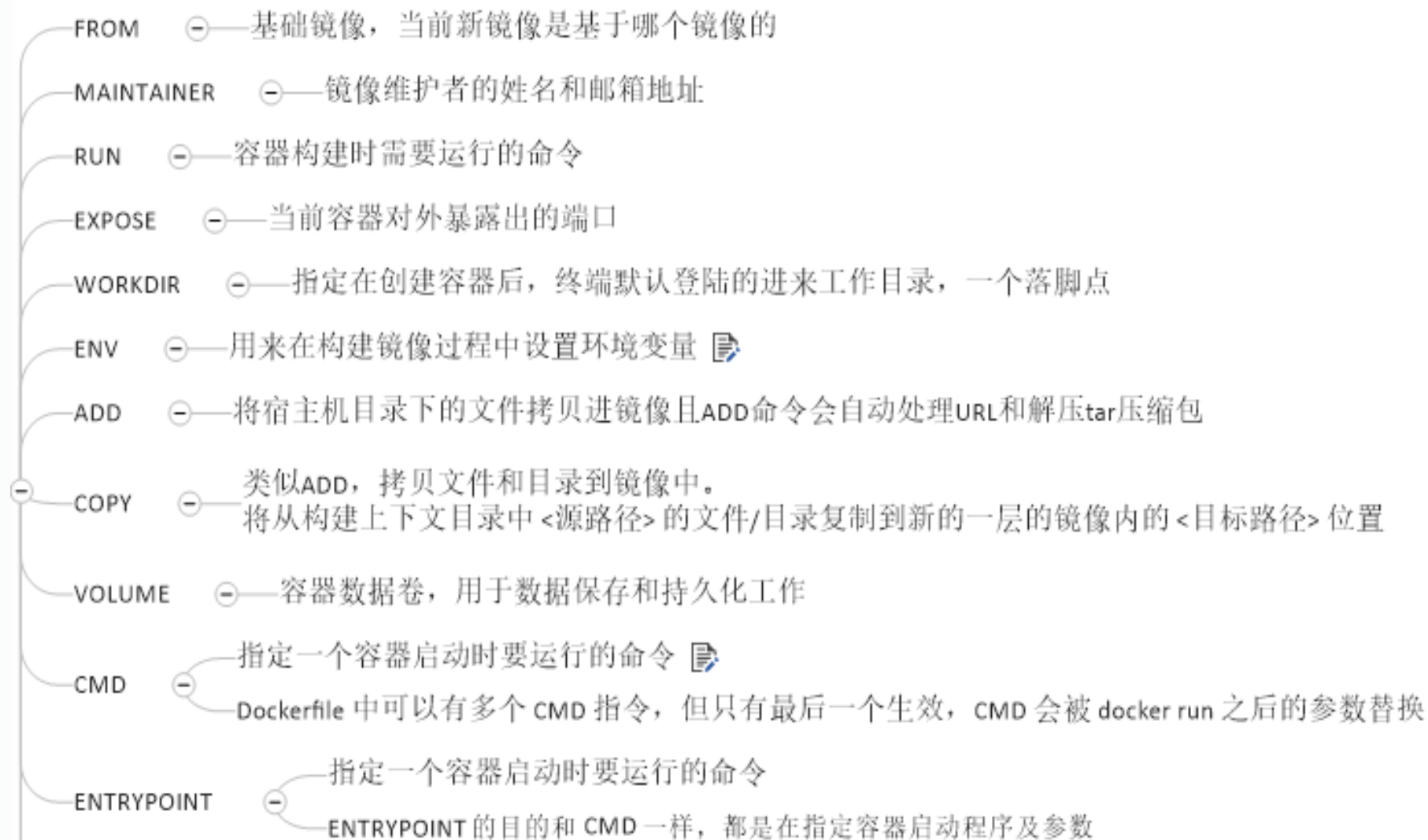
• Dockerfile内容基础知识:

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

• Docker执行dockerfile的流程:

1. Docker 从基础镜像运行一个容器
2. 执行一条指令并对容器作出修改
3. 执行类似 docker commit 的操作提交一个新的镜像层
4. Docker 再基于刚提交的镜像运行一个新容器
5. 执行 Dockerfile 中的下一条指令直到完成

• Dockerfile保留字指令：



- Dockerfile生成镜像:

docker build \
-f path/of/dockerfile \
-t image:tag \
.

- build 构建镜像
- f dockerfile文件所在位置绝对路径
- t 构建镜像的名称和标签


```
ssf@H5QJJH2:myubuntu$ cat Dockerfile
FROM ubuntu:16.04

ENV MYPATH /usr/bin/
WORKDIR $MYPATH

RUN mkdir my_test; \
    cd my_test; \
    touch docker_test.txt

CMD /bin/bash/

ssf@H5QJJH2:myubuntu$ docker build -f /home/ssf/soft/myubuntu/Dockerfile -t myubuntu:1.0 .
Sending build context to Docker daemon 2.048kB
Step 1/5 : FROM ubuntu:16.04
----> 96da9143fb18
Step 2/5 : ENV MYPATH /usr/bin/
----> Using cache
----> dd8b75272103
Step 3/5 : WORKDIR $MYPATH
----> Using cache
----> 4c68ca59939a
Step 4/5 : RUN mkdir my_test;      cd my_test;      touch docker_test.txt
----> Running in c89508568301
Removing intermediate container c89508568301
----> 6718f3be2a3a
Step 5/5 : CMD /bin/bash/
----> Running in 3918336895e8
Removing intermediate container 3918336895e8
----> 57736a28a6a0
Successfully built 57736a28a6a0
Successfully tagged myubuntu:1.0
```



- 基于容器创建新镜像

`docker commit [OPTIONS] 容器ID [REPOSITORY[:TAG]]`

options:

- a 提交的镜像作者
- m 提交时的说明文档

```
ssf@H5QJJH2:myubuntu$ docker run -it --name myubu myubuntu:1.0 /bin/bash
root@f9bca9e5ddfa:/usr/bin# echo "hello docker" > my_test/docker_test.txt
root@f9bca9e5ddfa:/usr/bin# cat my_test/docker_test.txt
hello docker
root@f9bca9e5ddfa:/usr/bin# ssf@H5QJJH2:myubuntu$
ssf@H5QJJH2:myubuntu$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
f9bca9e5ddfa        myubuntu:1.0       "/bin/bash"        5 minutes ago       Up 5 minutes                myubu
ssf@H5QJJH2:myubuntu$ docker commit -a "shishanfu" -m "update docker_test.txt" f9bca9e5ddfa myubuntu:2.0
sha256:45e34f480439faa84b467e8188e9d5449662836e73d81c8ce2f9b408b028dba2
ssf@H5QJJH2:myubuntu$ docker images myubuntu
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
myubuntu            2.0                45e34f480439       13 seconds ago     124MB
myubuntu            1.0                57736a28a6a0       20 minutes ago     124MB
```

镜像的保存和加载

一、保存镜像

`docker save centos_cutadapt -o centos_cutadapt.tar`



A screenshot of a file manager interface. The top toolbar contains icons for a dropdown menu, settings, share, folder creation, information, a list view, a toggle visibility, and a search bar labeled '搜索'. Below the toolbar is a table with three columns: '名称' (Name), '大小' (Size), and '种类' (Type). The table contains one entry: 'centos_cutadapt.tar' with a size of '1.18 GB' and a type of 'tar 归档'.

名称	大小	种类
centos_cutadapt.tar	1.18 GB	tar 归档

二、加载镜像

在任何安装有docker的机器上可以加载

`docker load -i centos_cutadapt.tar`

• 流程Docker封装

1. 环境配置

2. 软件编译安装

3. 脚本和数据库共享

```
ssf@H5QJJH2:mypipe$ docker images mypipe
REPOSITORY          TAG             IMAGE ID         CREATED          SIZE
mypipe              1.0            dc7b16e40855    6 hours ago     1.48GB
ssf@H5QJJH2:mypipe$ docker run -it --name mypipe01 -v /data/ngs:/data/ngs/ mypipe:1.0
root@ed52d733d6ee:/data/ngs# cd files/
root@ed52d733d6ee:/data/ngs/files# ll
total 212
drwxrwxrwx 19 1001 1001  4096 Feb 18 13:03 ./
drwxrwxr-x 28 1001 1001  4096 Jan 13 08:27 ../
drwxr-xr-x  4 1001 1001  4096 Dec 25 06:04 HBV---????????????????????/
drwxr-xr-x  4 1001 1001  4096 Dec 25 05:58 HPV---????????????????????/
drwxrwxr-x  2 1001 1001  4096 Jul 15 2019 QCReport/
drwxrwxr-x  8 1001 1001  4096 Nov 26 01:53 SJZP_data/
drwxrwxr-x  2 1001 1001  4096 Feb 15 13:29 SampleSheet/
drwxrwxr-x  2 1001 1001  4096 Nov 15 02:09 Sentieon_test/
drwxrwxr-x  4 1004 1004    65 Aug 28 08:58 SeqFiles/
drwxrwxr-x  4 1001 1001    71 Dec  9 08:59 bclFiles/
drwxrwxr-x 52 1001 1001 8192 Feb 17 04:26 clean/
drwxrwxr-x  2 1001 1001  4096 Feb 16 04:20 figures/
drwxr-xr-x 20 1001 1001  4096 Dec 23 17:58 igenetechdemo/
-rw-----  1 1001 1001 122231 Nov 22 21:16 nohup.out
drwxrwxr-x 54 1001 1001 12288 Feb 17 04:24 rawdata/
drwxrwxr-x  2 1001 1001  4096 Dec 24 06:42 result/
drwxrwxr-x  3 1001 1001  4096 Dec  5 2018 shoudu_data/
drwxrwxr-x  2 1001 1001    60 Dec  5 2018 shoudu_pon/
drwxrwxr-x  5 1001 1001 12288 Aug 26 09:05 zhengzhou_rawdata/
drwxr-xr-x  9 1001 1001  4096 Dec 25 06:53 ??????????????????????---????????????????????/
```



感谢聆听