1.docker底层依托于linux怎么实现资源隔离的?

基于Namespaces的视图隔离

Docker使用Linux内核的命名空间功能来提供不同进程间的隔离

每个命名空间提供了一个独立的视图，使得进程认为自己拥有独立的系统资源

Docker使用了多种类型的命名空间来实现容器的隔离，包括PID（Process ID）命名空间、网络命名空间、挂载命名空间、UTS（UNIX Time-Sharing System）命名空间、IPC（Inter-Process Communication）命名空间、用户命名空间

基于cgroups的资源隔离

cgroups是Linux内核的一个功能，用于限制、记录和隔离进程组使用的物理资源（如CPU、内存、磁盘I/O等）

Docker使用cgroups来确保每个容器只能使用分配给它的资源，并且不会影响其他容器或宿主机的性能

2.docker的工作原理是什么，讲一下

docker是一个Client-Server结构的系统，docker守护进程运行在宿主机上，守护进程从客户端接受命令并管理运行在主机上的容器，容器是完全使用沙箱机制，相互之间不会有任何接口

- **Docker使用容器来运行和管理应用程序，而Docker本身是一个提供容器化服务的平台**
- **沙箱（Sandbox）机制是一种安全技术，用于隔离运行中的程序，以限制其对系统资源的访问和操作，从而保护系统免受恶意软件、病毒或未授权的访问的影响**

3.docker的组成包含哪几大部分

Docker引擎

Server

一个常驻进程，用来管理整个Docker的交互，实现客户端和服务端的通信

Client

Docker客户端提供了一系列命令，用于与Docker服务端进行交互。用户可以通过命令行或其他Docker客户端工具发送请求给Docker服务端

基础组件

Docker包含三个重要的基础组件，这些组件共同构成了Docker的核心功能

镜像（Image）

Docker镜像是一个轻量级、可执行的独立软件包，它包含了运行应用所需的所有内容，如代码、运行时、库、环境变量和配置文件等

镜像可以被看作是一个只读的模板，通过它可以创建出容器实例

容器（Container）

容器是Docker的运行实例，它是基于镜像创建的。容器包含了应用程序及其运行时环境，并且具有独立的文件系统、网络和进程空间

容器之间是相互隔离的，每个容器都拥有自己的资源限制，这样可以确保容器内的应用不会影响到其他容器或宿主机

仓库（Repository）

Docker仓库是用于集中存放镜像文件的场所。它类似于代码仓库，可以根据所存储的镜像是否公开分享，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，其中存放着大量成熟的镜像供用户下载使用。此外，用户也可以在本地网络创建自己的私有仓库

附加组件与工具

除了上述核心组件外，Docker还提供了一些附加组件和工具，以扩展其功能和使用范围

Docker Compose

一个用于定义和运行多容器Docker应用程序的工具。通过YAML文件来配置应用程序的服务、网络和卷等

Docker Swarm

Docker的内置容器编排工具，可以将多个Docker节点组成一个集群，实现容器的自动化管理和负载均衡等功能

Docker Machine

一个可以在虚拟主机上安装Docker的工具，它可以简化在远程主机上部署Docker的步骤

Docker Registry

除了作为镜像的集中存放地外，Docker Registry还支持镜像的搜索、版本控制等功能

Dockerfile

一个文本文件，其中包含了创建镜像所需的一系列命令和参数。通过Dockerfile，用户可以自定义构建自己的镜像

- **Docker守护进程（Docker daemon），也称为Docker服务端或Docker引擎**

4.docker与传统虚拟机的区别什么?

特性/技术	Docker	传统虚拟机
架构	共享宿主机的内核	需要模拟整个硬件环境
资源使用	资源占用少，启动迅速	资源占用多，启动较慢
启动速度	迅速，通常几秒	较慢，可能需要数分钟
部署与可移植性	简化部署，一致的运行环境，高可移植性	部署复杂，迁移成本较高
隔离性	基于进程和用户空间，中等隔离	硬件辅助虚拟化，强隔离

特性/技术	Docker	传统虚拟机
安全性	容器内root权限可能影响宿主机，需安全配置	较低安全风险，但效率较低
系统开销	低开销，高服务器利用率	高开销，可能导致资源浪费

- **“容器内root权限可能影响宿主机，需安全配置”这句话的意思是，在Docker等容器技术中，如果容器内的应用程序以root权限运行，那么这些应用程序可能会利用这一权限对宿主机（即运行容器的物理机或虚拟机）的安全构成威胁**

5.docker技术的三大核心概念是什么?

基础组件

Docker包含三个重要的基础组件，这些组件共同构成了Docker的核心功能

镜像（Image）

Docker镜像是一个轻量级、可执行的独立软件包，它包含了运行应用所需的所有内容，如代码、运行时、库、环境变量和配置文件等

镜像可以被看作是一个只读的模板，通过它可以创建出容器实例

容器（Container）

容器是Docker的运行实例，它是基于镜像创建的。容器包含了应用程序及其运行时环境，并且具有独立的文件系统、网络和进程空间

容器之间是相互隔离的，每个容器都拥有自己的资源限制，这样可以确保容器内的应用不会影响到其他容器或宿主机

仓库（Repository）

Docker仓库是用于集中存放镜像文件的场所。它类似于代码仓库，可以根据所存储的镜像是否公开分享，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，其中存放着大量成熟的镜像供用户下载使用。此外，用户也可以在本地网络创建自己的私有仓库

6.centos镜像几个G，但是docker centos镜像才几百兆，这是为什么?

一个完整的Linux操作系统包含Linux内核和rootfs（/usr、/bin、/etc等目录）

平时看到的CentOS除了rootfs，还会选装很多软件，服务，图形桌面等

对于容器镜像而言，所有容器都是共享宿主机的Linux 内核的，只需要包含最基本的命令，工具，程序库即可

- **rootfs（root file system，根文件系统）**

7.讲一下镜像的分层结构以及为什么要使用镜像的分层结构?

节省空间

相同基础层的镜像可以共享，避免重复存储

构建效率

可以利用缓存，避免重复构建未改变的层

安全性

镜像是只读的，确保了内容的一致性

灵活性

可以轻松定制和扩展镜像

8.讲一下容器的copy-on-write特性，修改容器里面的内容会修改镜像吗?

镜像是分层的，镜像的每一层都可以被共享，同时，镜像是只读的

当一个容器启动时，一个新的可写层被加载到镜像的顶部，这一层被称作"容器层"，“容器层”之下的都叫“镜像层”

添加文件

在容器中创建文件时，新文件被添加到容器层中

读取文件

在容器中读取某个文件时，Docker 会从上往下依次在各镜像层中查找此文件

一旦找到，立即将其复制到容器层，然后打开并读入内存

修改文件

在容器中修改已存在的文件时，Docker 会从上往下依次在各镜像层中查找此文件

一旦找到，立即将其复制到容器层，然后修改之

只有当需要修改时才复制一份数据（copy-on-write），容器层保存的是镜像变化的部分，不会对镜像本身进行任何修改

删除文件

在容器中删除文件时，Docker 也是从上往下依次在镜像层中查找此文件

找到后，会在容器层中记录下此删除操作

- **COW（Copy-on-Write，写时复制；当有多个调用者或用户/进程同时请求相同资源时，他们会共同获取相同的指针指向相同的资源，直到某个调用者试图修改资源的内容时，系统才会真正复制一份专用副本给该调用者，而其他调用者所见到的最初的资源仍然保持不变）**

9.简单描述一下Dockerfile的整个构建镜像过程

创建一个目录用于存放应用程序以及构建过程中使用到的各个文件等

在这个目录下创建一个Dockerfile文件，一般建议Dockerfile的文件名就是Dockerfile

编写Dockerfile文件，编写指令

使用`docker build -t 镜像名:tag .`命令来构建镜像（最后一个点是表示当前目录），docker会默认寻找当前目录下的Dockerfile文件来构建镜像，如果不使用默认，可以使用-f参数来指定dockerfile文件，如`docker build -t 镜像名:tag -f /xx/xxx/Dockerfile`

使用docker build命令构建之后，docker会将当前目录下所有文件发送给docker daemon，顺序执行Dockerfile文件里的指令（在构建过程中会检查缓存，如果之前的层没有变化，会重用该层以加快构建速度），构建完成后，Docker会生成一个新的镜像，包含了`Dockerfile`中所有指令的结果

10.Dockerfile构建镜像出现异常，如何排查?

Dockerfile是一层一层的构建镜像，期间会产生一个或多个临时容器，构建过程中其实就是在临时容器里面安装应用，如果因为临时容器安装应用出现异常导致镜像构建失败，这时容器虽然被清理掉了，但是期间构建的中间镜像还在，那么我们可以根据异常时上一层已经构建好的临时镜像，将临时镜像运行为容器，然后在容器里面运行安装命令来定位具体的异常

11.Dockerfile的基本指令有哪些?

FROM

指定基础镜像，作为构建新镜像的起点

```
FROM node:14
```

RUN

执行命令，可以是shell形式或exec形式，用于安装软件包、编译代码等

```
RUN apt-get update && apt-get install -y nginx
```

CMD

提供容器启动时默认执行的命令

```
CMD ["nginx", "-g", "daemon off;"]
```

ENTRYPOINT

设置容器启动时的入口点，可以是一个命令或脚本

```
ENTRYPOINT ["java", "-jar", "app.jar"]
```

COPY

从构建上下文复制新文件或目录到容器的文件系统

```
COPY . /app
```

ADD

与`COPY`类似，但具有额外的功能，比如自动解压缩

```
ADD file.tar.gz /app
```

ENV

设置环境变量

```
ENV NODE_ENV production
```

ARG

定义构建时的变量

```
ARG VERSION=1.0
```

VOLUME

创建挂载点，用于数据持久化

```
VOLUME /var/www/html
```

EXPOSE

声明容器运行时监听的端口

```
EXPOSE 80
```

WORKDIR

设置工作目录

```
WORKDIR /app
```

USER

设置运行容器时的用户和用户组

```
USER 1000
```

HEALTHCHECK

定义检查容器健康状况的命令

```
HEALTHCHECK --interval=30s --timeout=30s CMD curl -f http://localhost/ || exit 1
```

STOPSIGNAL

设置停止容器时发送的系统调用信号

```
STOPSIGNAL SIGTERM
```

LABEL

为镜像添加元数据

```
LABEL maintainer="name@example.com"
```

12.如何进入容器？使用哪个命令

```
docker exec -it <container_name_or_id> /bin/bash
-i 或 --interactive
保持容器的标准输入（STDIN）开放，即使没有附加任何东西也是如此。这允许你与在容器中运行的命令进行交互
-t 或 --tty
分配一个伪终端（TTY），这通常是一个命令行界面，允许你像在本地机器上一样与容器进行交互
```