Docker是什么？

Docker 是一种基于容器化技术的开源平台，它的核心概念是容器。容器是一种轻量级、独立、可移植的软件打包方式，其中包含了应用程序及其所有依赖项，如代码、运行时、系统工具、系统库等。每个容器都是相互隔离的，拥有自己的文件系统，可以在同一台主机上并行运行多个容器。

Docker 提供了一种方便的方式来创建、分发和运行容器。它通过使用 Docker 镜像来实现这一点。Docker 镜像是一个只读模板，其中包含了运行应用程序所需的所有内容。用户可以基于现有的 Docker 镜像创建自定义的容器，并通过 Docker Hub 或私有的 Docker 仓库共享这些镜像。

Docker像是一个集装箱这样将应用程序和运行所需的打包在一起，无论在什么环境下运行都不会因为环境不同而出现问题

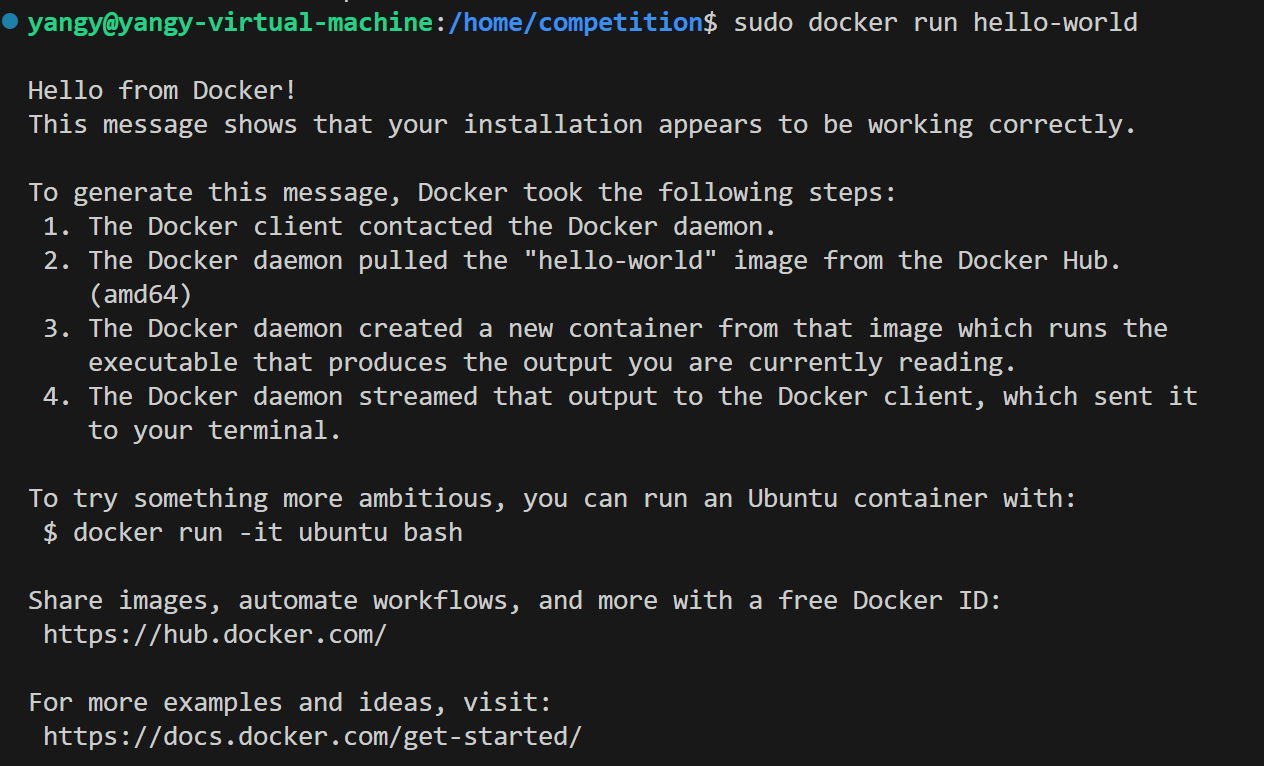
Docker客户端：管理命令

仓库：有远程仓库，运行redis，如果本地没有镜像从远程拉镜像再制作成容器

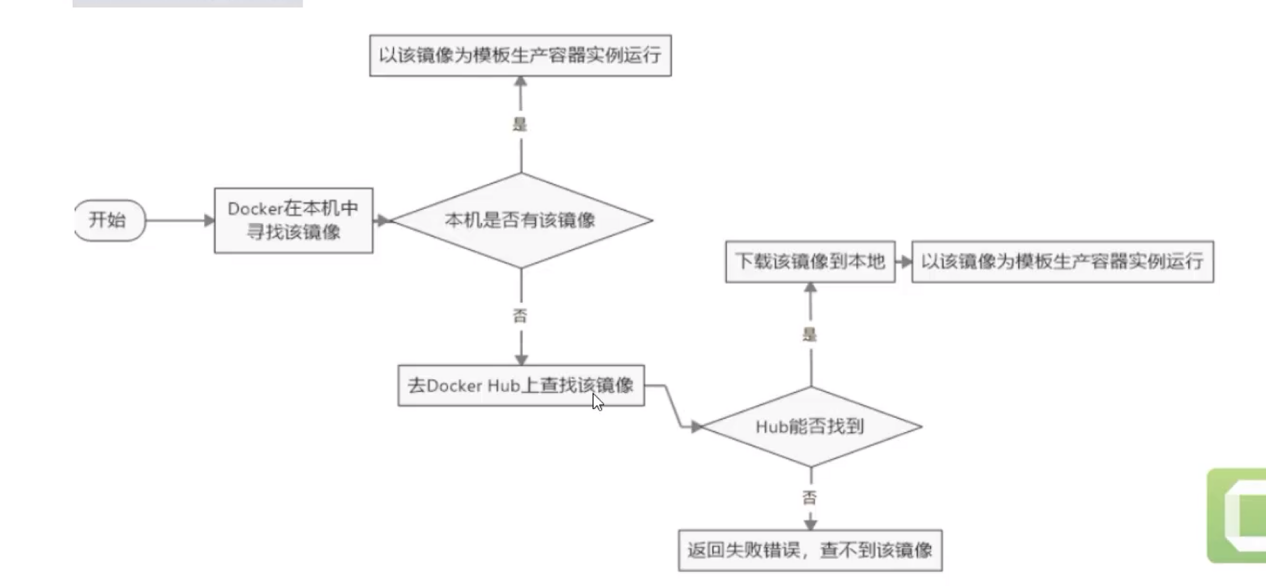
主体

镜像：一个文件含了用于运行特定应用程序的所有信息，可以被 Docker 引擎加载并用于创建容器。想象一下镜像就像是一个盒子，里面包含了一个完整的小世界。这个小世界里有一套完整的规则和资源：有人、房子、道路、食物等等。而这个盒子可以在任何地方打开，里面的世界都会完整展现出来。每当你需要运行一个应用程序时，你只需找到合适的盒子（镜像），打开它（创建容器），你就能够在里面运行你的应用程序了。

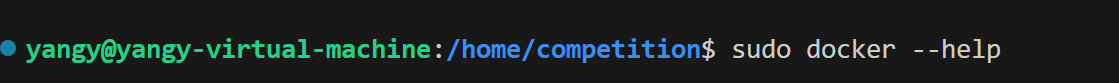
与docker的关系：docker利用镜像创建容器，每个容器都是一个运行中的镜像实例，比如运行hello world的镜像，这个镜像就是用来显示这个字符串，本地没有镜像，从远程仓库拉入然后建立容器，底下为helloworld镜像实现：



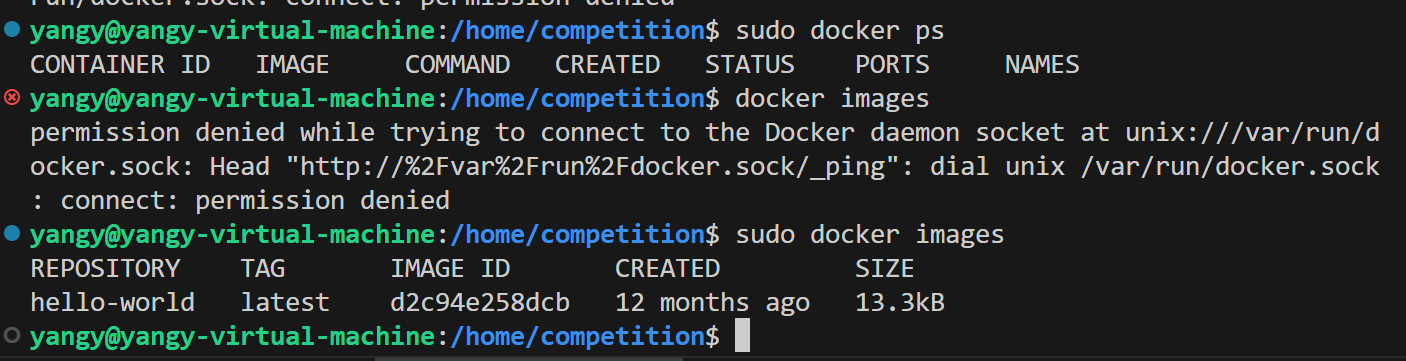
Docker run



Docker操作镜像命令



一个查容器一个镜像



容器镜像是一个轻量独立可执行软件包

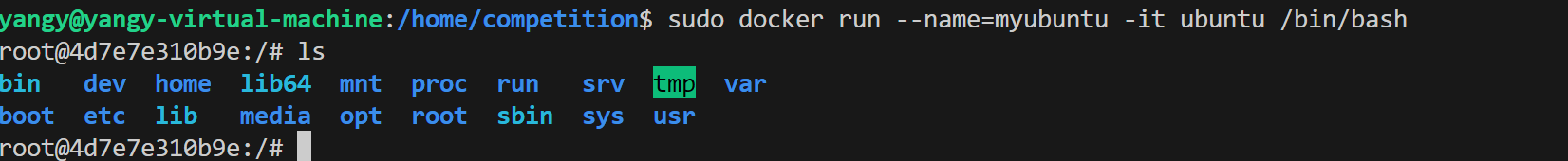
创建并启动容器：

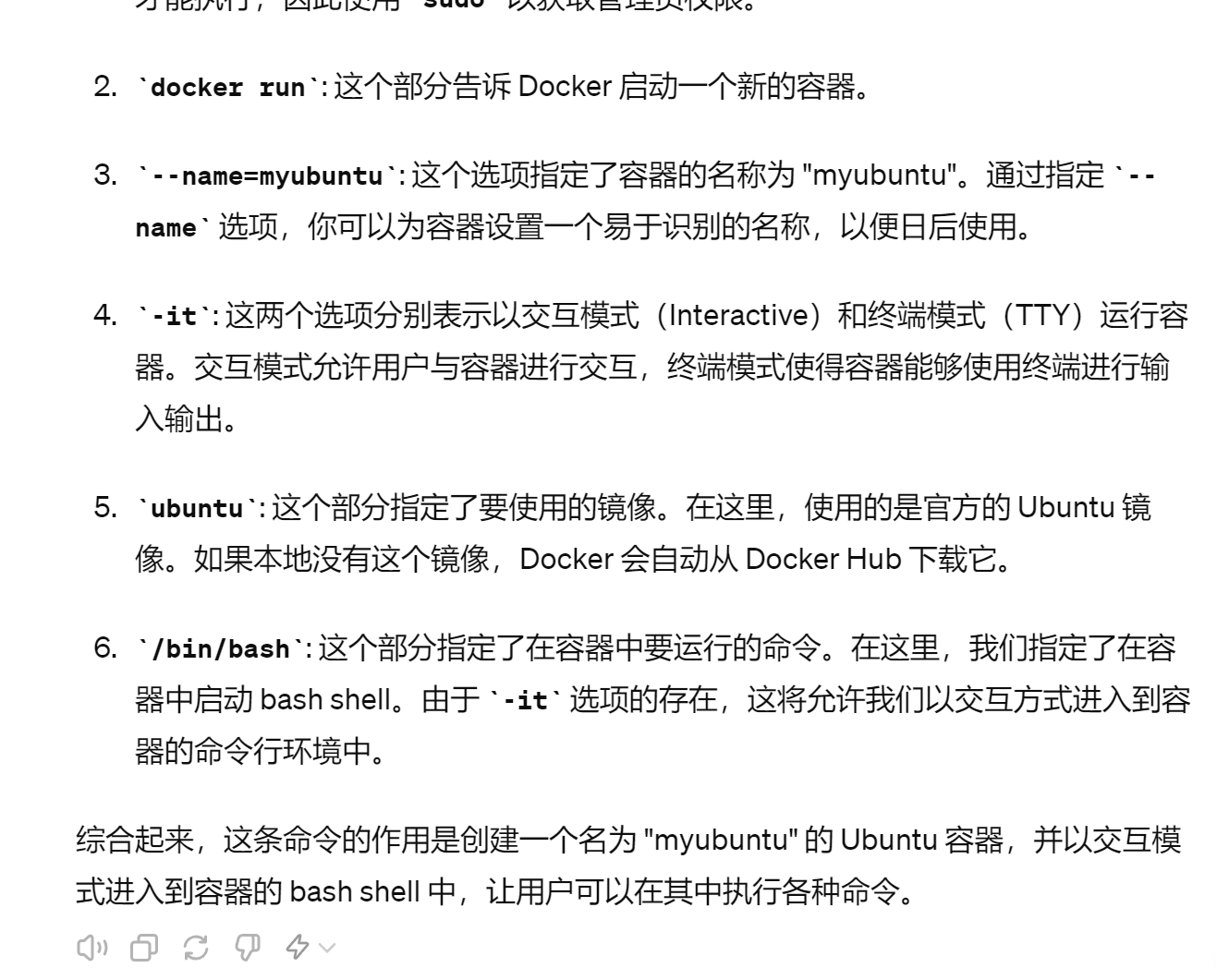




容器名 指定镜像名 进入到容器里面

/bin/bash 直接进到容器里面 (这里可以体现出容器隔离思想





Exit退出容器并且直接把容器关掉，前提是容器以交互模式启动

所以尽量使用-d保证它可以一直在后台运行，守护式进程

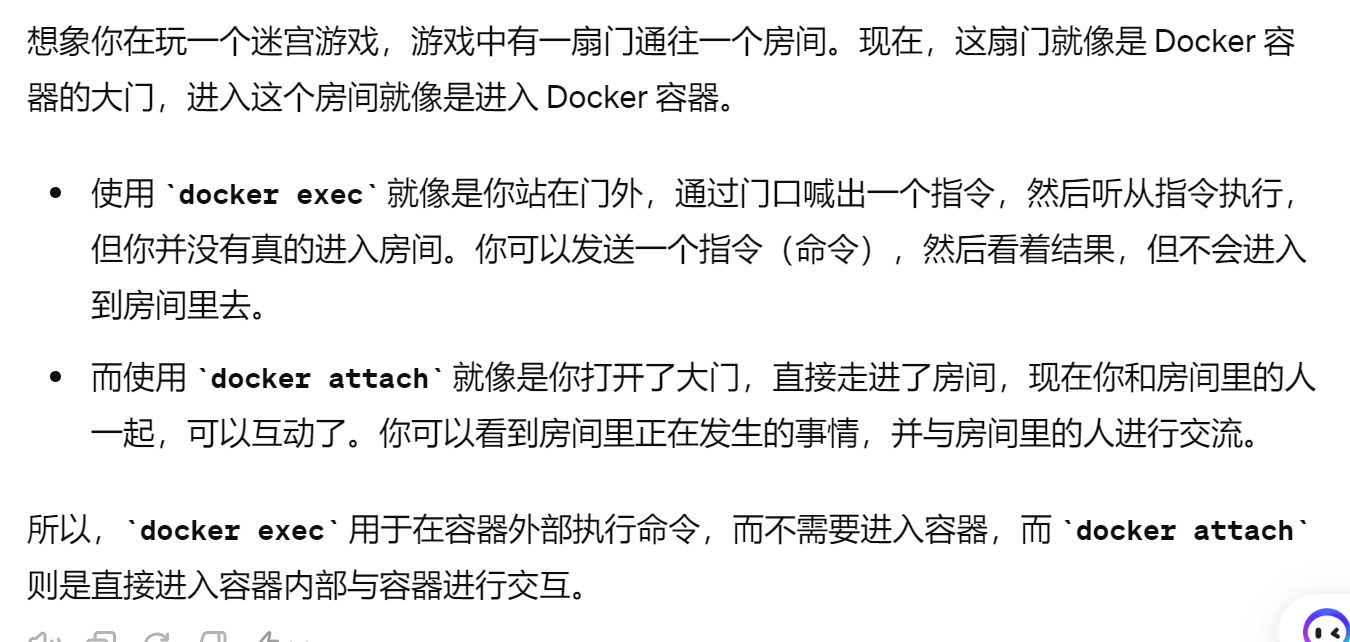
Docker就像是一个很小的操作系统



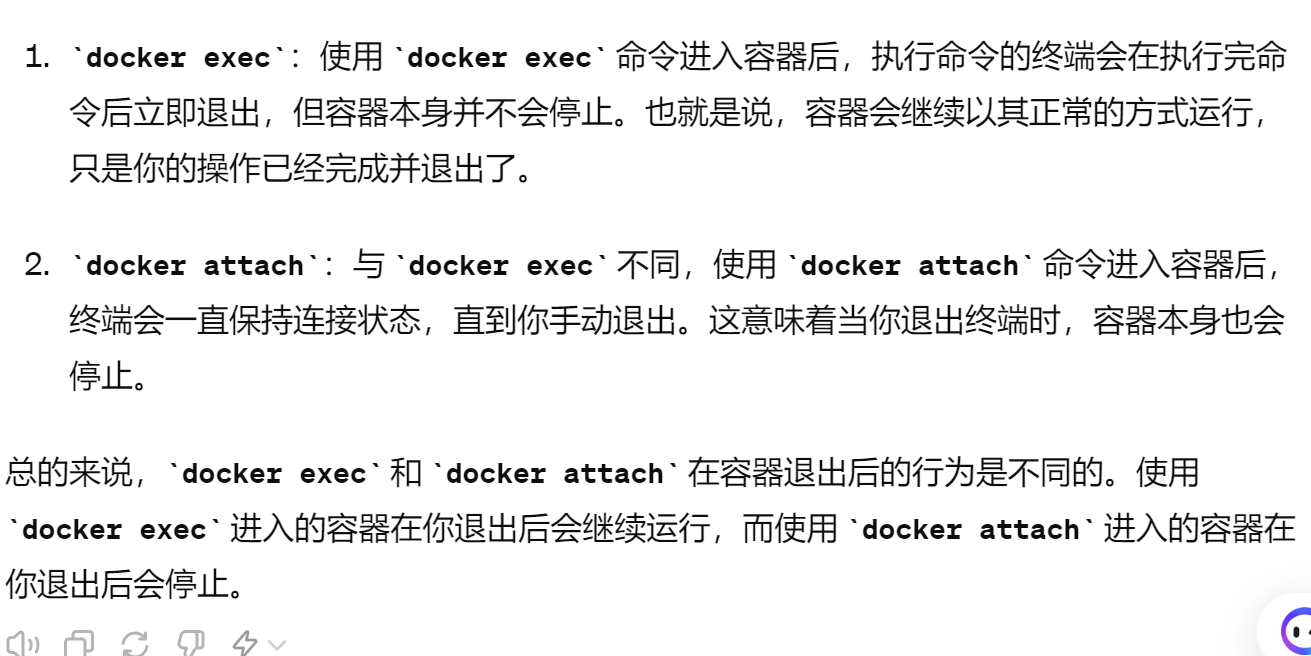
进入到容器内部



还有一个attach命令他们的区别



而且exit退出后使用attach会直接关闭掉正在进行的容器但是exec并不会





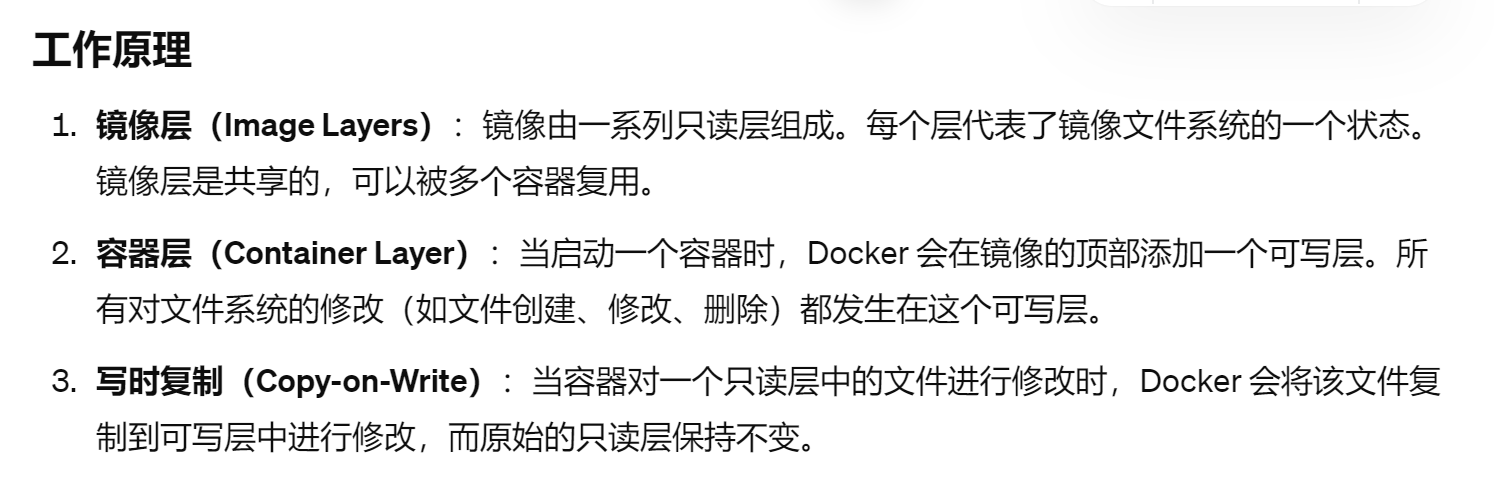
拷贝容器中的文件到主机

可以利用cp

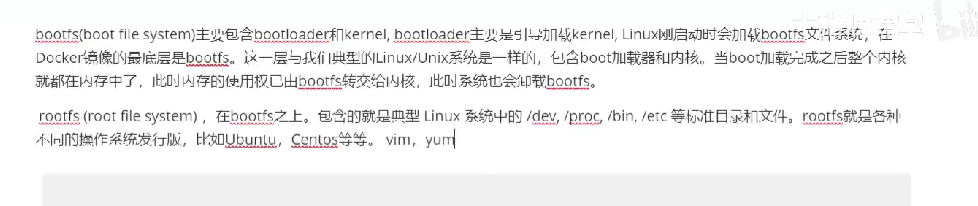


Docker镜像

由一层一层文件系统构成，unionfs，docker使用它来构成镜像和容器的，Docker 镜像由多个只读层组成，每个层代表了文件系统的一个快照。当创建一个新的容器时，Docker 会在镜像的顶部添加一个可写层，从而形成容器的文件系统。这种层次结构使得 Docker 能够高效地管理存储，并且可以轻松地创建、共享和分发镜像。



容器文件系统构成：



容器本身也是一种镜像，只是它可以写

创建镜像commit操作

就是把当前的容器状态保存为一个新的镜像，你可能在这个容器修改了一些部分，保存为新的镜像方便下载，因为比如你要保存容器里的东西，但是还有对容器进行一些挂载操作，你就要把这个容器保存为新的镜像



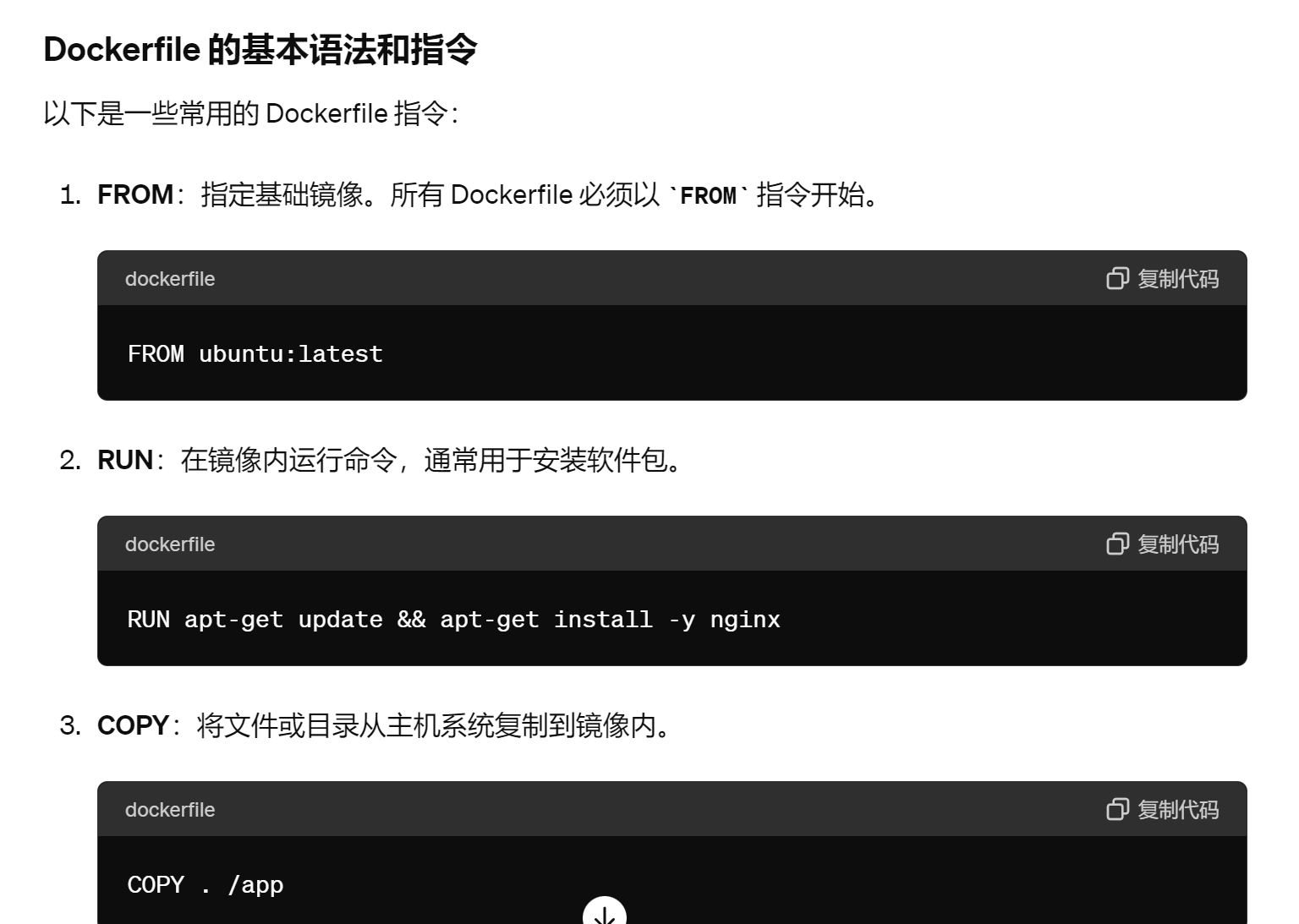
添加数据卷：volume

容器内部一些东西你不想用，但是你一删，项目也删了，所以可以采用数据卷，把主机上的目录挂载到容器内的目录上，前面cp只能修改一小段代码，比如想workapps你就不想要，用cp修改显得太low了

- v --mount用于创建数据卷

Dockerfile一个文本文件，包含了用于构建docker镜像的指令

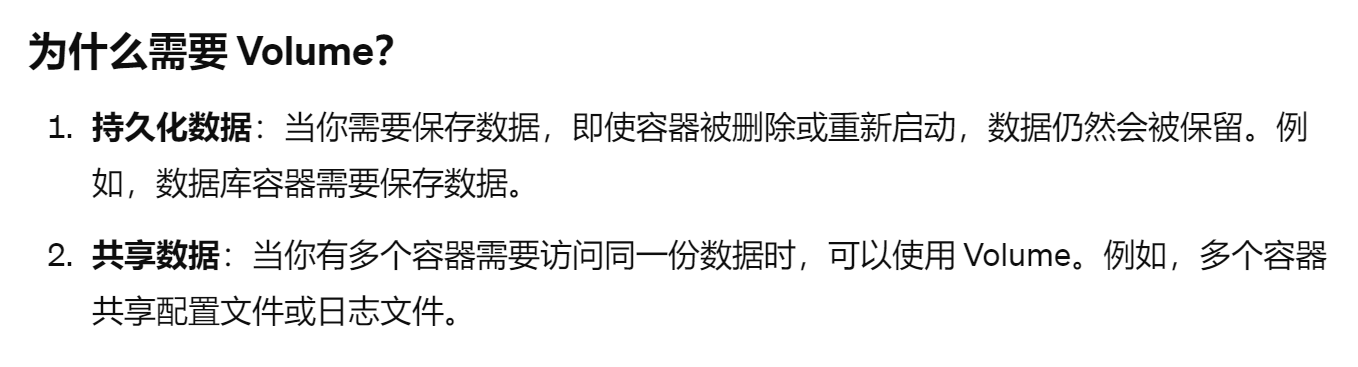
From基础镜像





重点介绍volume：类似于容器外部的硬盘，存储数据，容器删除后仍然存在，

为何出现：



创建volume和使用你可以把它连接到一个或多个容器上面



类型：匿名卷（就是直接进行目录挂载）

命名卷：





volume使用

