Docker 命令

**为什么有Docker?**

Docker 是一种开源平台，用于开发、交付和运行应用程序。它使开发人员能够将应用程序及其所有依赖项打包到一个标准化的单元（称为容器）中，从而确保应用程序在任何环境中都能一致地运行。以下是为什么有 Docker 的一些主要原因：

1. 一致的开发和生产环境：

Docker 容器包含应用程序及其所有依赖项，确保在不同的环境（如开发、测试和生产环境）中运行时行为一致。这有助于减少环境相关的问题，从而提高应用程序的稳定性和可靠性。

2. 轻量级和高效的虚拟化：

Docker 容器与传统的虚拟机相比更加轻量级，不需要整个操作系统的开销。这使得 Docker 容器能够在相同硬件上运行更多的实例，从而提高资源利用率。

3. 简化的持续集成和持续交付 (CI/CD)：

Docker 与 CI/CD 工具集成良好，使得构建、测试和部署过程更加自动化和高效。开发人员可以快速创建和销毁测试环境，从而加速开发和交付周期。

4. 依赖管理：

Docker 容器打包了应用程序及其所有依赖项，从而避免了“在我电脑上可以运行”的问题。开发人员不再需要担心依赖冲突或缺少必要的库。

5. 跨平台兼容性：

Docker 容器可以在任何支持 Docker 的平台上运行，包括本地机器、云服务器和数据中心。开发人员可以轻松地在不同的操作系统和硬件平台之间迁移应用程序。

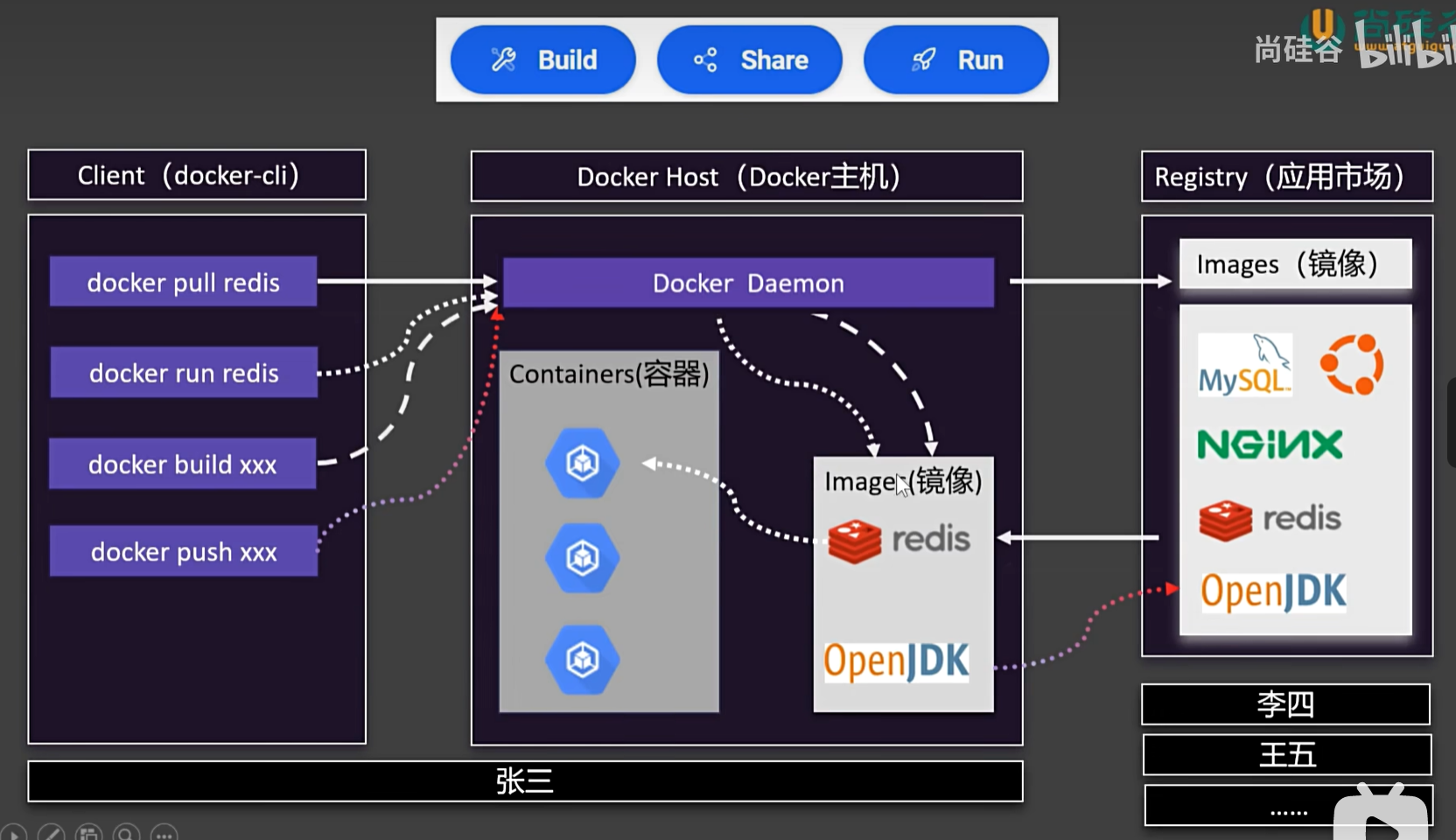
6. 模块化和微服务架构支持：

Docker 容器使得微服务架构的实现更加简单和灵活。每个微服务可以运行在独立的容器中，便于独立开发、测试、部署和扩展。

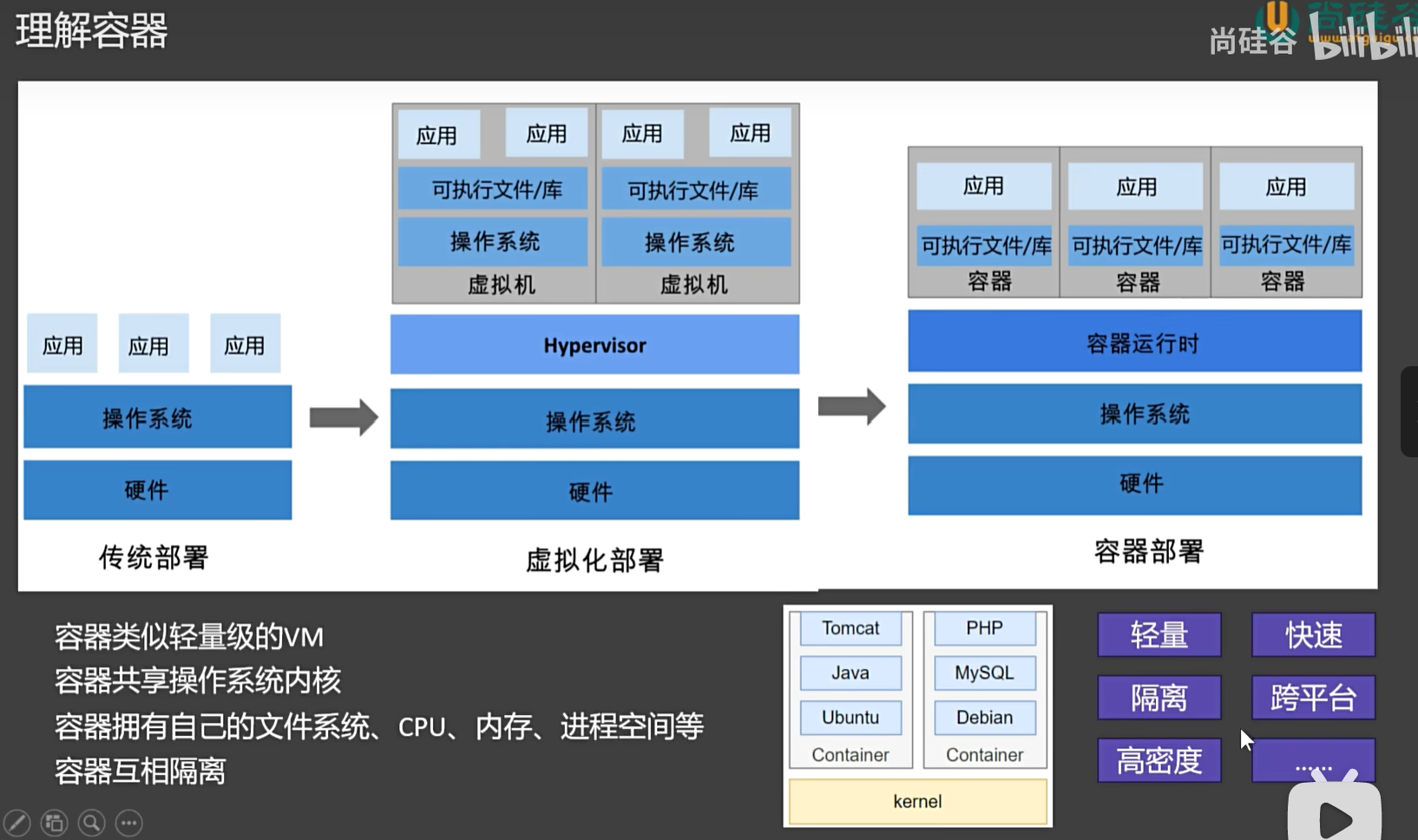
7. 社区支持和生态系统：

Docker 拥有庞大的社区支持和丰富的生态系统，包括 Docker Hub 上的数千个预构建镜像、各种开源工具和企业级解决方案。这些资源极大地简化了容器的使用和管理。

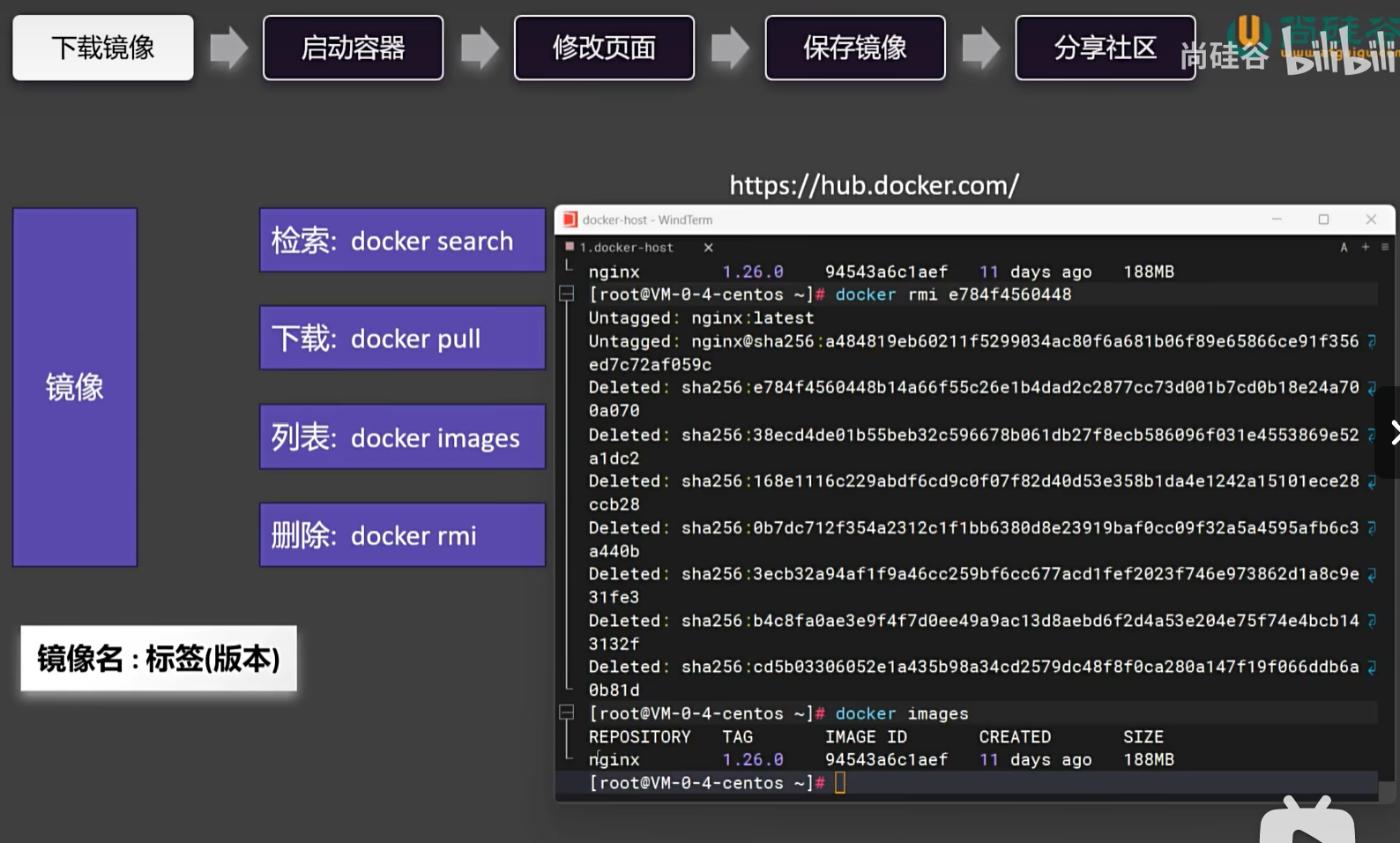
**Docker 如何工作**

****

**理解容器**

****

**镜像操作**

****

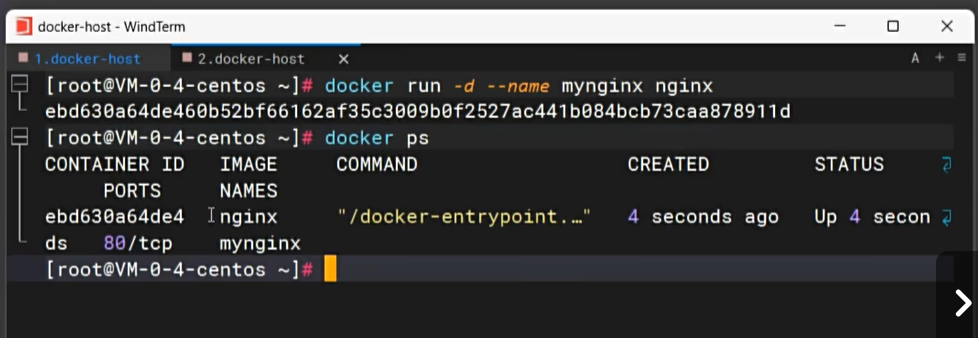
**容器操作**

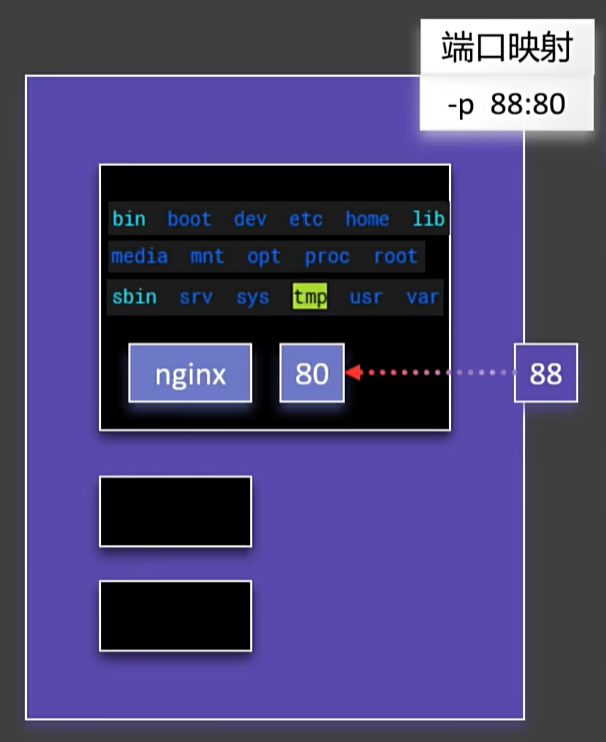
****

**Docker run 细节：**

后台启动 + 自定义命名

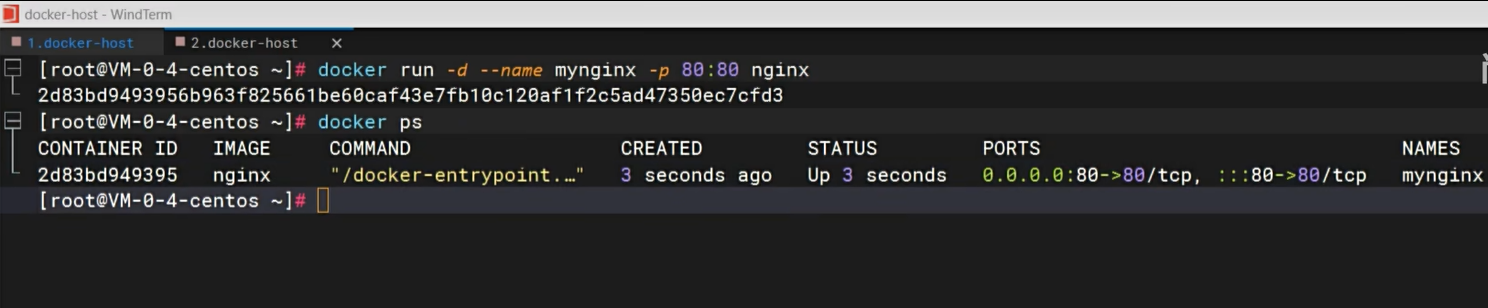
此时访问nginx会发现无法访问，因为这个只运行在当前容器之中，从外部主机无法访问，需要端口映射，假设使用 -p 88:80，就是外部的88端口直接监听该容器内部的 80 端口

****

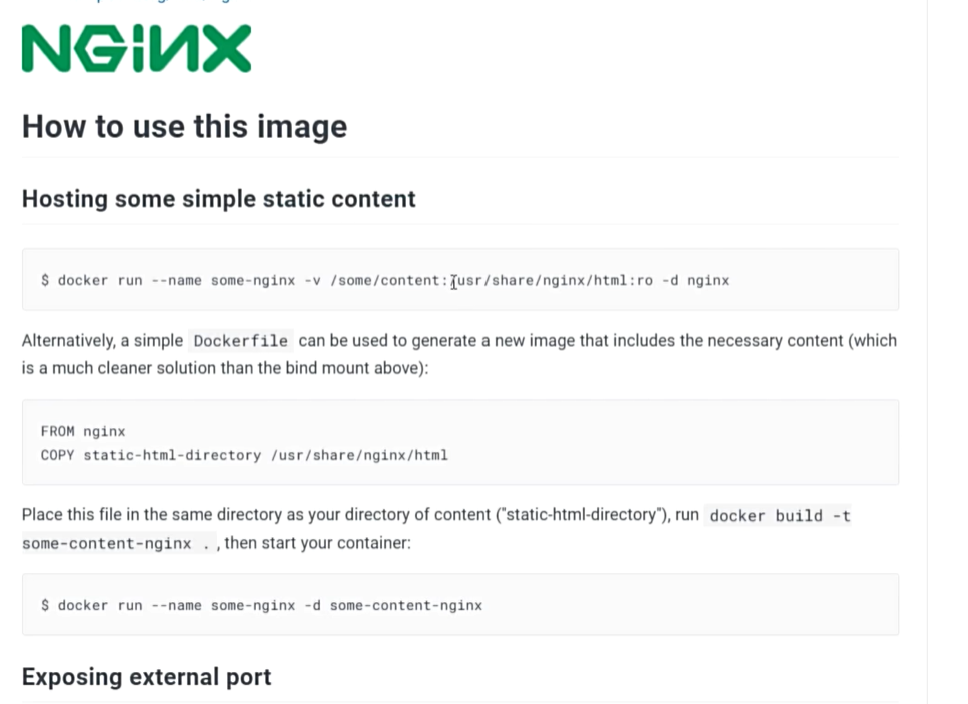


注意：容器端口可重复，但是主机端口不可重复使用

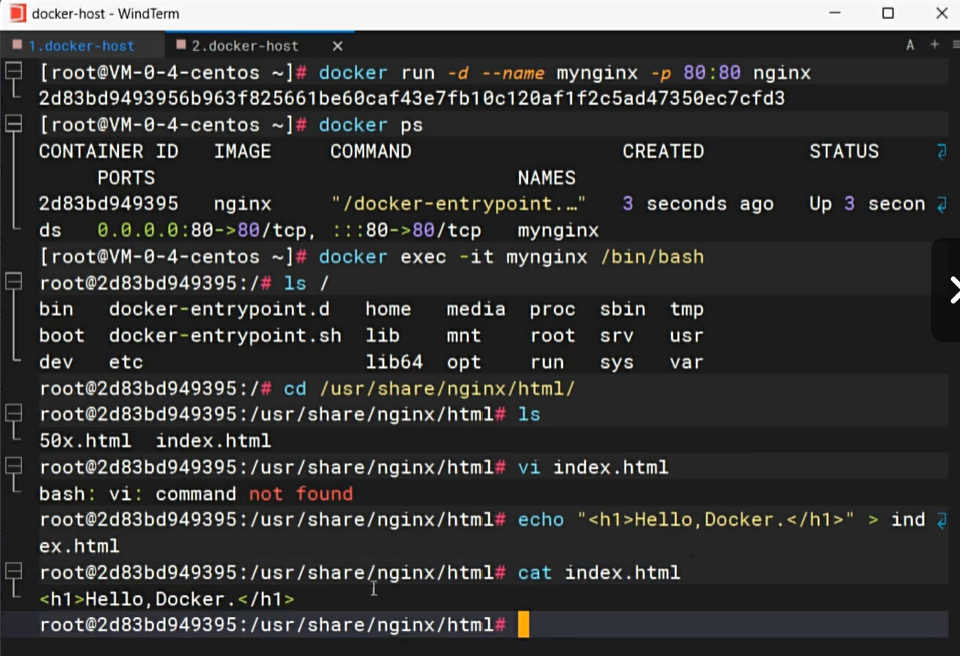
每个Docker 容器都可以被视为一个独立的 Linux 系统



阅读官方 Docker hub文档可以获取更多关于这个官方容器的信息，比如默认网页文件的位置



使用docker exec 和单个容器进行交互，进入这个容器后使用 ls可以检查当前文件目录，会发现和 Linux 十分类似。这里对默认网页文件内容进行了修改。可以使用 exit 命令推出当前容器



使用docker exec 和单个容器进行交互，进入这个容器后使用 ls可以检查当前文件目录，会发现和 Linux 十分类似。这里对默认网页文件内容进行了修改。可以使用 exit 命令推出当前容器

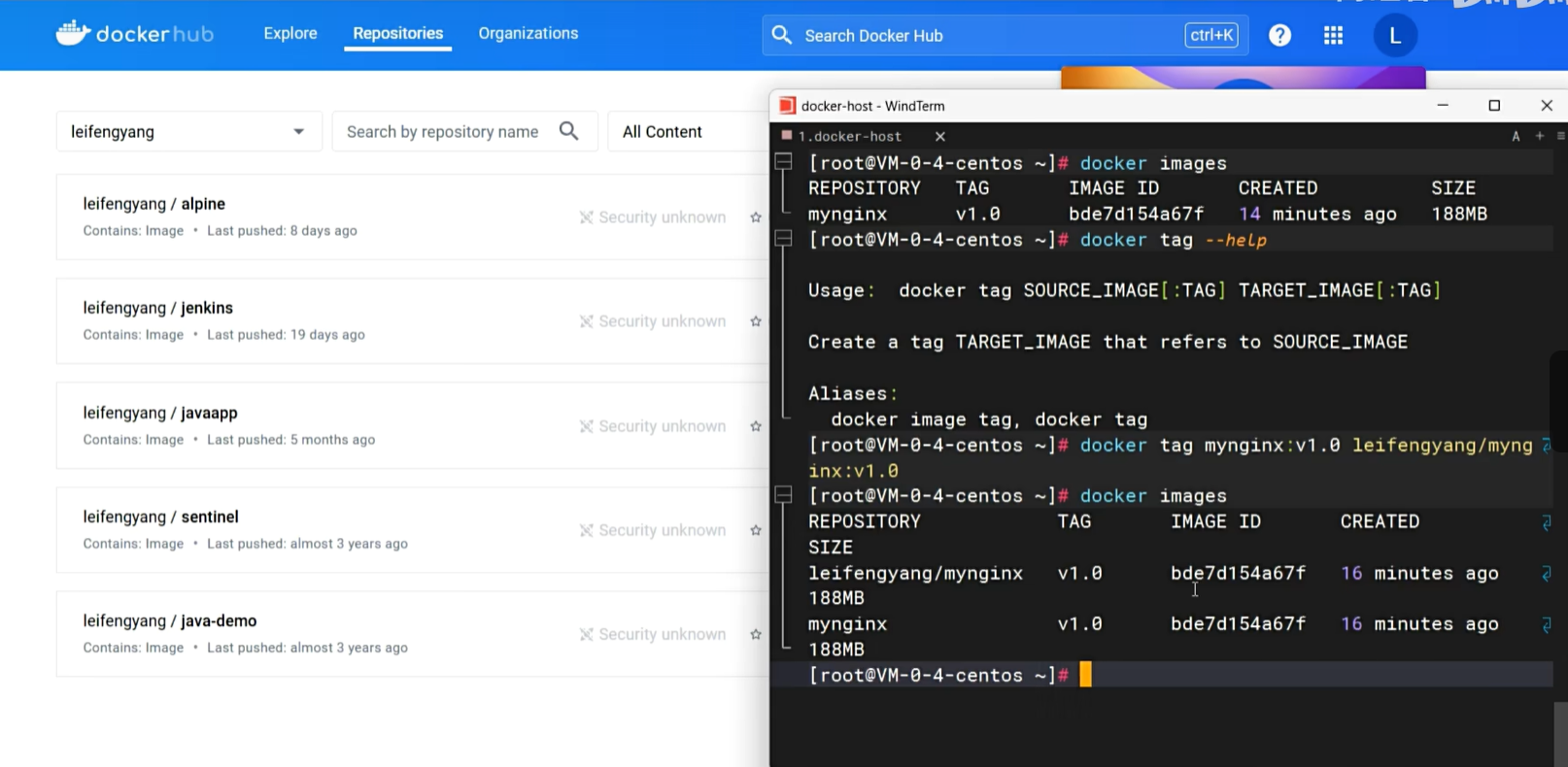
**保存镜像**

****

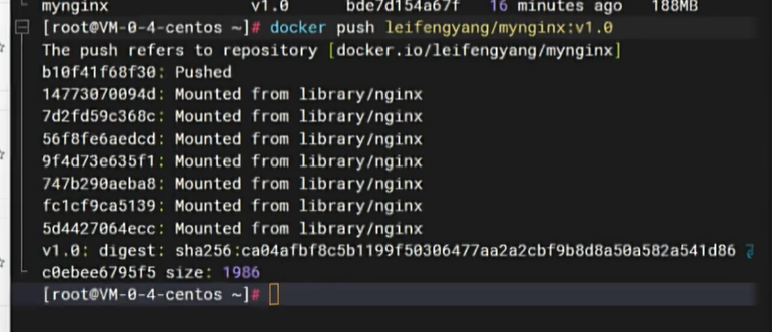
**分享社区**

**登陆后可以分享镜像，每个镜像都有自己独特的名字，一般是用户名/镜像名，需要用docker tag 来改名，不管怎么改名，镜像 id 是不变的**

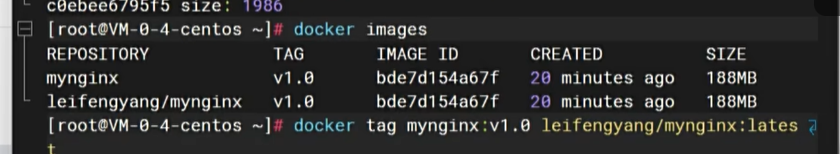
****

****

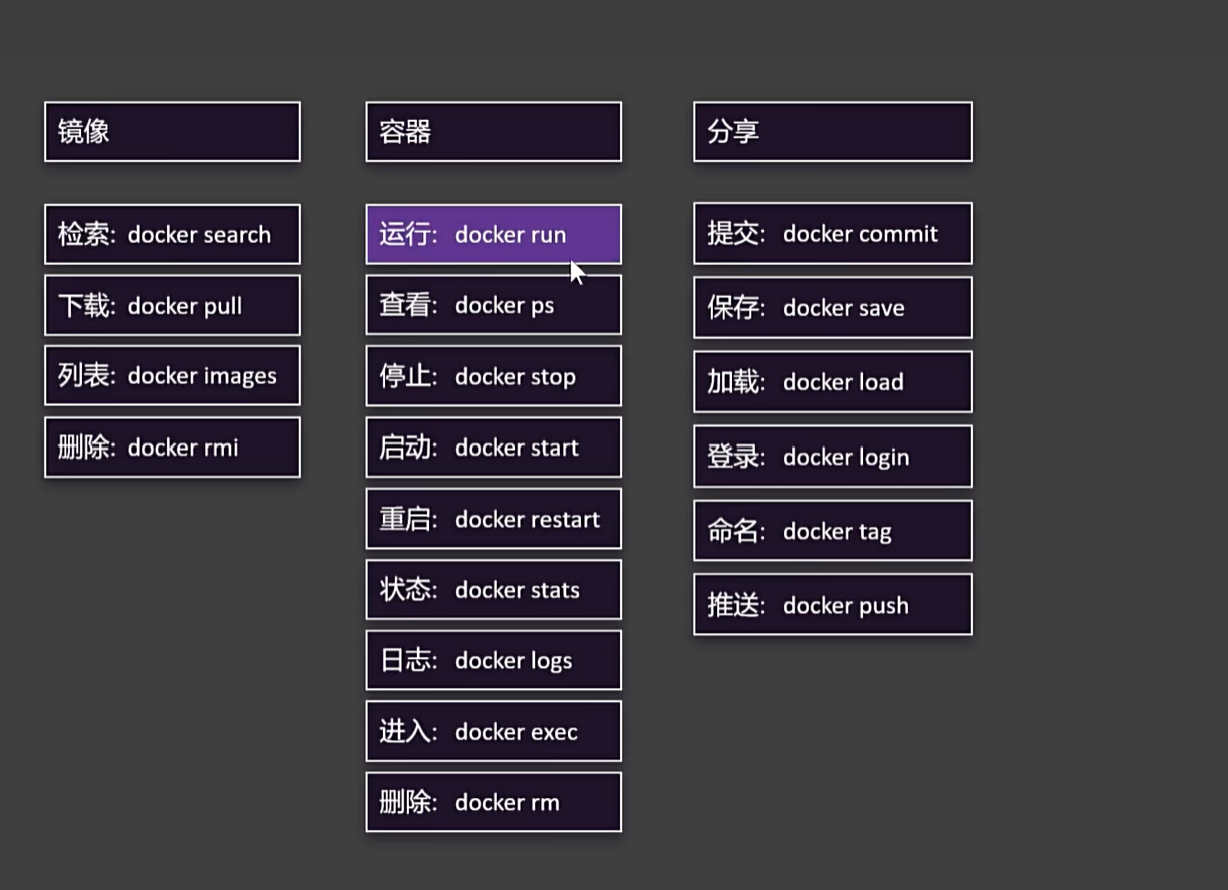
**docker push加镜像tag进行推送**

****

**可以设置自己最新的镜像为lastest，这样别人下载时即便不加版本号也不报错，就会默认下载最新版本镜像**

****

**总结**

****