Docker 安装教程

官方网站：Docker: Accelerated Container Application Development

中文教程：Docker 教程 | 菜鸟教程

Docker的优点

1. **一致性和可移植性**：Docker 容器可以在任何支持 Docker 的地方运行，无论是开发者的本地机器、 测试环境还是生产环境。
2. **隔离性**：每个容器相互隔离，确保应用程序之间不会互相干扰。
3. **效率**：容器是轻量级的，它们共享操作系统内核，启动速度快，占用资源少。
4. **版本控制和管理**：Docker 允许你为应用程序创建和管理不同的版本。

Docker 的基本概念

1. **镜像（Image）**：Docker 镜像是一个只读的模板，用于创建 Docker 容器。镜像可以包含操作系统、应用程序及其依赖项。
2. **容器（Container）**：容器是由镜像创建的运行实例。它是一个轻量级、独立的执行环境。
3. **Dockerfile**：这是一个文本文件，包含一系列指令，用于自动化地构建 Docker 镜像。
4. **Docker Hub**：这是一个公共的镜像仓库，可以从这里下载官方和社区维护的 Docker 镜像。

Docker Hub Container Image Library | App Containerization

基本命令

1. 安装 Docker

a. 官方教程

- 参考 Docker 官方安装指南。
  - windows：Windows | Docker Docs
  - macOS：Mac | Docker Docs
  - Linux：Linux | Docker Docs

b. 命令行方式安装

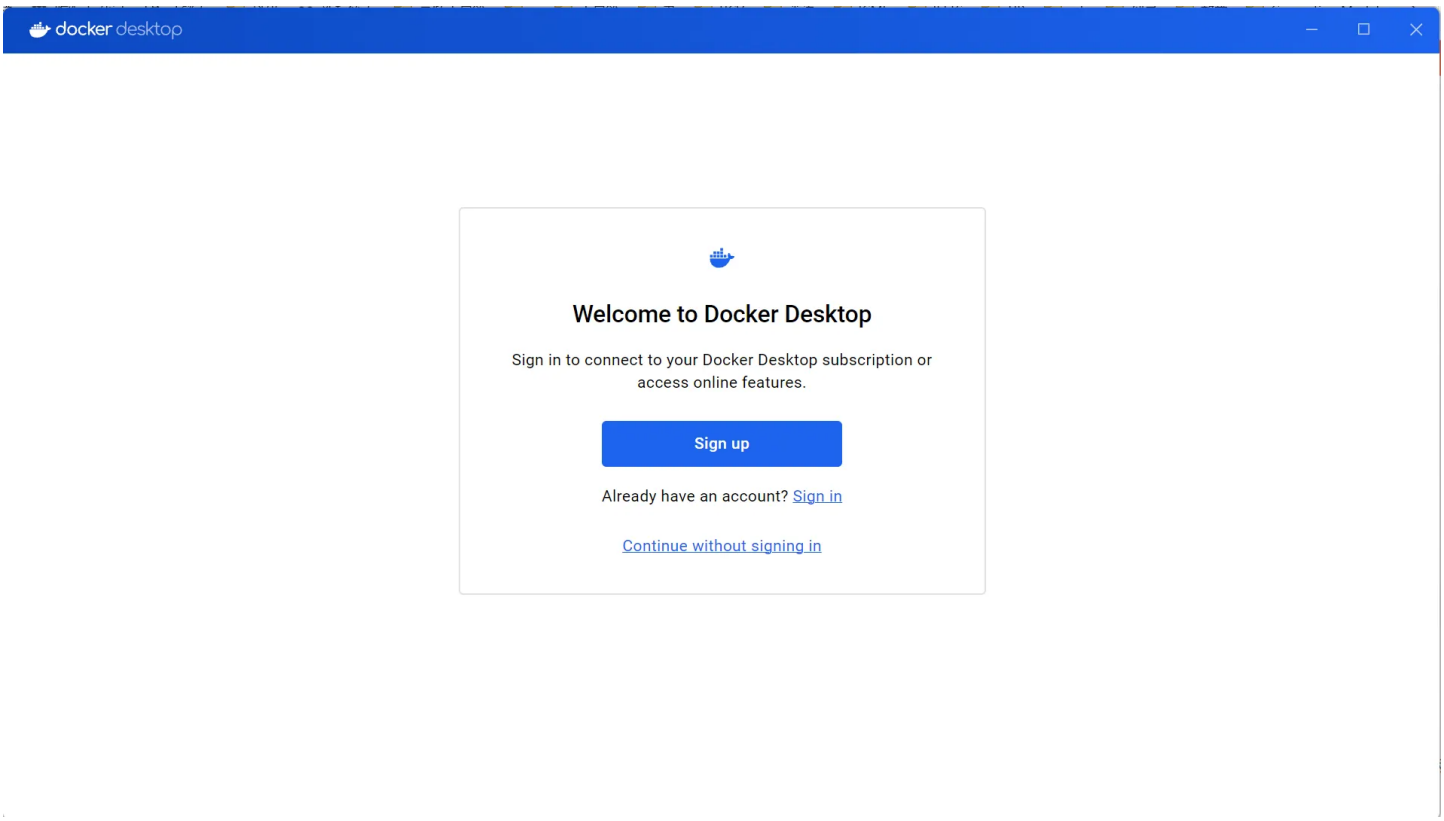
- docker安装手册
- docker-compose安装手册

c. 下载Docker桌面程序（桌面程序一般自带docker-compose）✅

- 下载Docker desktop
- 然后下一步下一步就安装好了，如果拉取镜像比较慢，可以更改为国内镜像地址

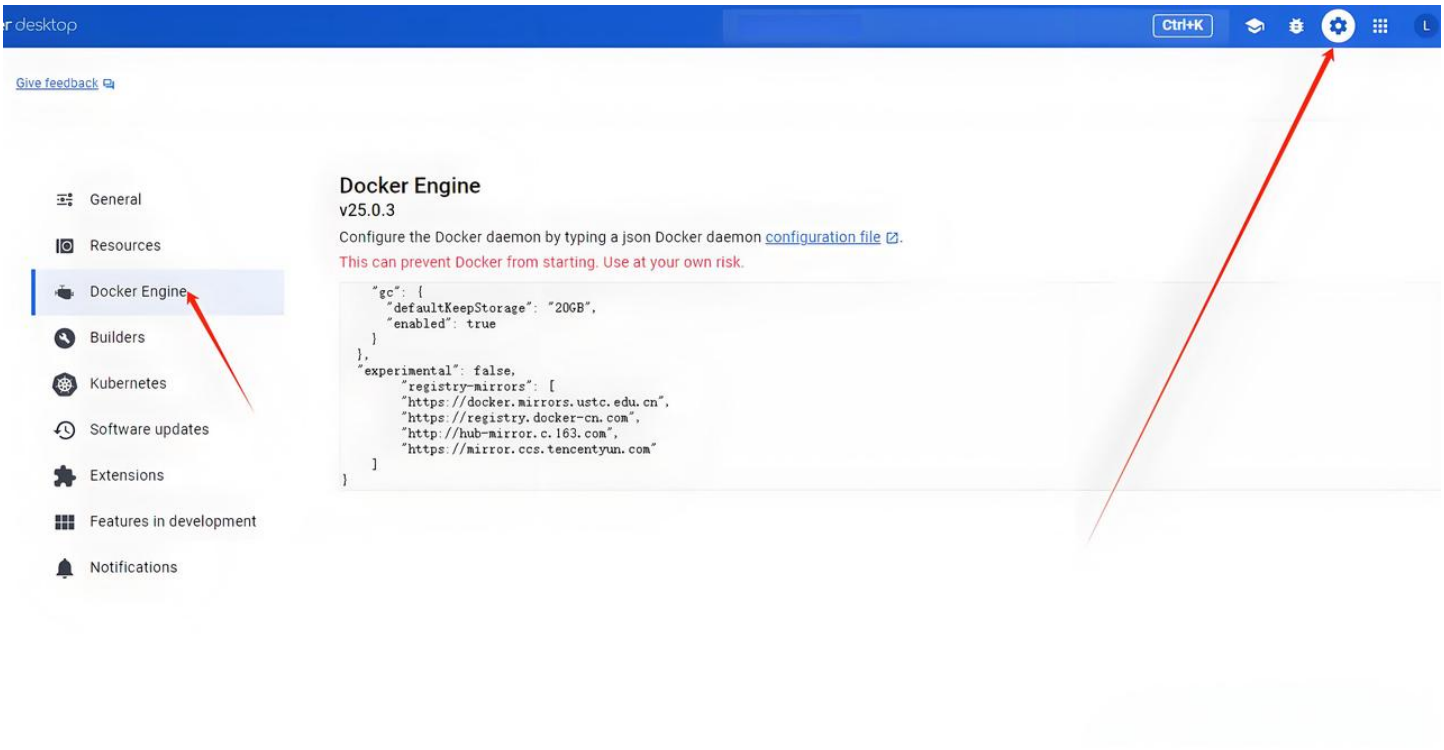
检查是否安装成功，安装正常会打印日志

```
1  docker -v
2  docker-compose -v
```



修改镜像源

在docker设置->docker引擎->配置文件中增加镜像源



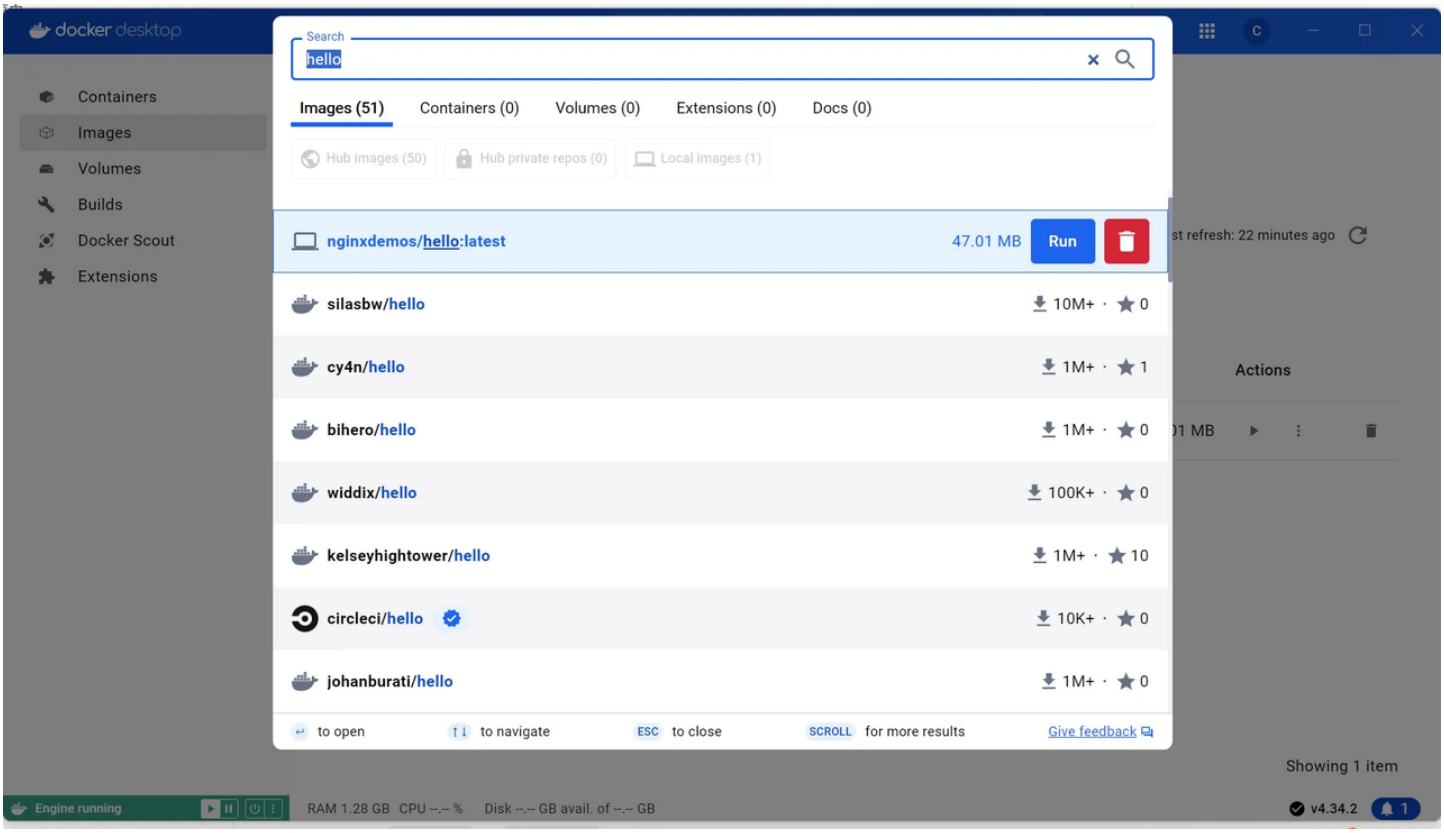

```
{
  "builder": {
    "gc": {
      "defaultKeepStorage": "20GB",
      "enabled": true
    }
  },
  "experimental": false,
  "registry-mirrors": [
      "https://hub.rat.dev",
      "https://dockerpull.org/",
      "https://mirror.azure.cn/"
  ]
}
```

配置完成后点击左下角的重启

2. 运行第一个容器

```
docker run hello-world
```

这条命令会下载一个 hello-world 镜像并运行它。



3. 查看运行的容器

```
docker ps
```

这条命令会列出所有正在运行的容器。

4. 停止容器

```
docker stop [容器ID]
```

替换 `[容器ID]` 为 `docker ps` 命令中显示的实际容器 ID。

5. 删除容器

```
docker rm [容器ID]
```

删除指定的容器。

6. 构建镜像

创建一个名为 `Dockerfile` 的文件，然后运行：

```
docker build -t myimage .
```

这条命令会根据 `Dockerfile` 构建一个名为 `myimage` 的镜像。

Docker Desktop

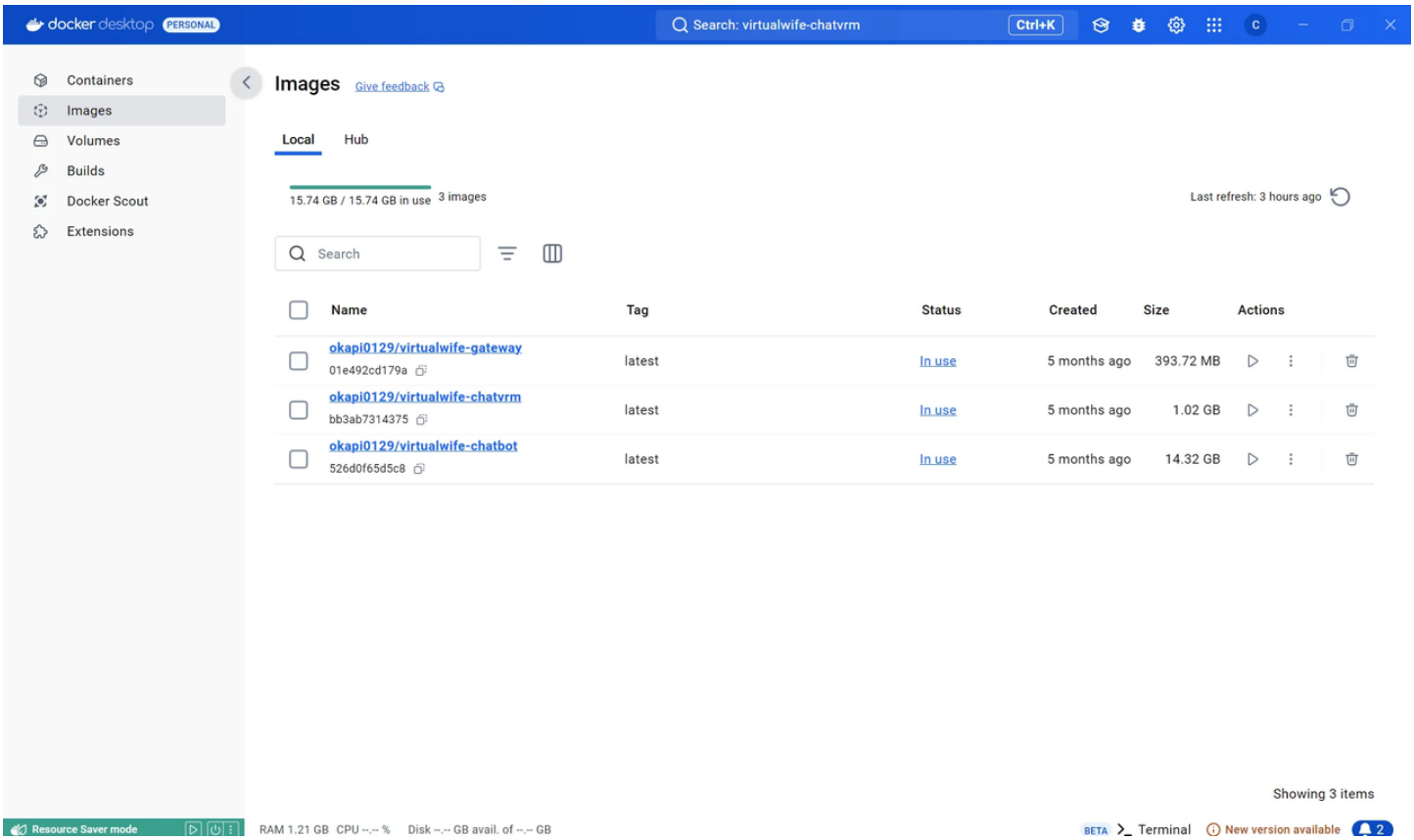
介绍

Docker Desktop 是 Docker 官方提供的桌面版应用程序，它允许用户在 Windows 和 Mac 上轻松地使用 Docker。以下是 Docker Desktop 的一些主要特点和功能：

1. 简单易用：Docker Desktop 提供了一个图形用户界面，使得管理容器变得非常简单。用户可以通过点击几个按钮来创建、启动和停止容器。
2. 集成开发环境：Docker Desktop 可以与 IDE（如 Visual Studio Code）集成，可以直接在 IDE 中构建和运行容器。
3. 支持多种操作系统：Docker Desktop 支持 Mac 和 Windows 两种操作系统。
4. 支持多种应用：Docker Desktop 支持运行各种应用，包括 Web 应用、数据库、后台服务等。
5. 自动更新：Docker Desktop 会自动检查并安装更新。
6. Docker Engine：Docker Desktop 包括 Docker Engine，这是 Docker 的核心组件，负责容器的创建、运行和网络管理。
7. Docker Compose：Docker Desktop 提供了 Docker Compose，这是一个用于定义和运行多容器 Docker 应用程序的工具。通过一个 YAML 文件，可以配置应用程序的服务，然后使用一个命令来启动和停止它们。
8. Docker Build：Docker Desktop 提供了一个简化的容器构建工具，允许将代码打包并构建成可以在任何地方运行的容器镜像。
9. Docker Debug：提供了高级的故障排除工具，用于诊断和解决容器和镜像中的问题。
10. Docker Private Extensions Marketplace：提供了一个定制扩展的市场，允许你根据特定需求定制和增强你的 Docker 环境。

Docker Desktop 是一个强大的工具，它通过提供用户友好的界面和集成开发工具，简化了容器的管理和运行，从而提高了开发人员的生产力和效率。

Images



Docker 镜像是一个轻量级、可执行的软件包，包含了运行一个容器所需的所有内容：代码、运行时、库、环境变量和配置文件。镜像具有以下特点：

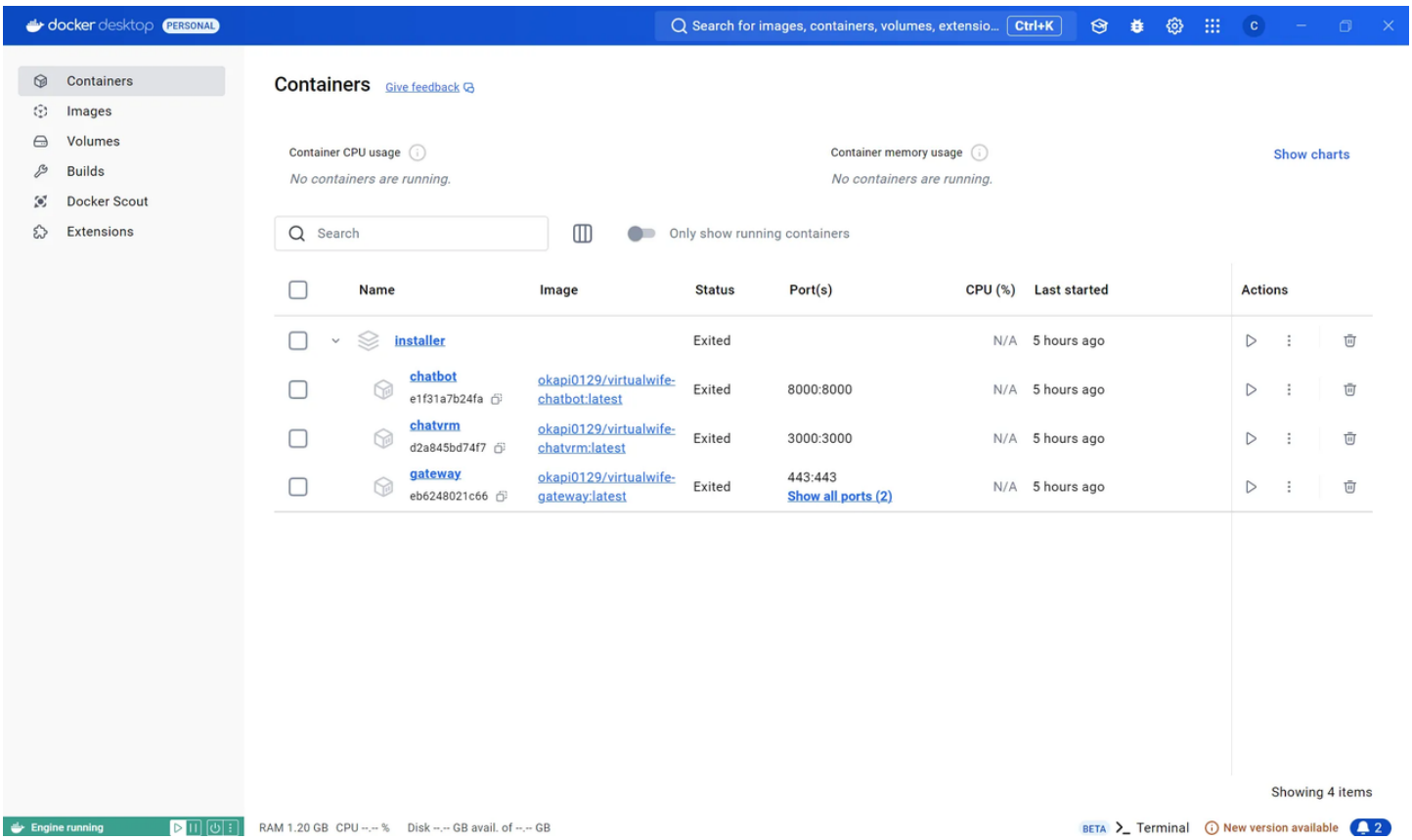
1. 只读：镜像一旦创建，就是只读的。当容器基于镜像启动时，Docker 会在镜像的顶层添加一个可写层，以便对文件系统进行更改。
2. 分层：Docker 镜像由多个层组成，每一层代表 Dockerfile 中的一个指令。这种分层结构有助于优化存储和加快构建过程，因为层可以被重用和共享。
3. 可移植：Docker 镜像可以在任何安装了 Docker Engine 的机器上运行，无论是物理机还是虚拟机，无论是本地还是云环境。
4. 版本控制：镜像可以标记为不同的版本，并且可以推送到 Docker Hub 或其他 Docker 仓库进行存储和分发。
5. 可堆叠：多个镜像可以堆叠在一起，例如，你可以创建一个基础镜像，然后在上面添加其他层来构建特定的应用程序镜像。
6. 快速启动：由于镜像是只读的，Docker 可以快速启动容器，因为不需要对镜像本身进行修改。
7. 安全：Docker 镜像可以被扫描以查找安全漏洞，确保运行的容器是安全的。

创建 Docker 镜像的常见步骤包括：

- 使用 `docker build` 命令根据 Dockerfile 创建镜像。
- 使用 `docker pull` 从 Docker Hub 或其他仓库拉取现成的镜像。
- 使用 `docker tag` 给镜像打上标签，以便识别和分发。
- 使用 `docker push` 将镜像推送到远程仓库。

Docker 镜像是 Docker 容器化平台的基石，它们使得应用程序的部署、分发和运行变得更加容易和可靠。

Containers



Docker 容器是 Docker 镜像的运行实例。当启动一个 Docker 镜像时，Docker 创建一个容器，并在隔离的环境中运行应用程序或服务。以下是 Docker 容器的一些基本特点：

1. 隔离性：每个容器都与其他容器隔离，拥有自己的文件系统、网络设置和进程空间。这种隔离确保了容器之间的操作不会相互干扰。
2. 轻量级：容器共享宿主机的内核，不需要像虚拟机那样运行完整的操作系统，因此启动迅速，资源利用率高。
3. 可移植性：容器可以在任何安装了 Docker 的机器上运行，无论是在开发环境、测试环境还是生产环境。
4. 可堆叠：多个容器可以在同一宿主机上运行，并且可以相互协作，形成一个复杂的应用程序。
5. 版本控制：容器的创建和运行可以基于不同版本的镜像，这有助于版本控制和回滚。
6. 易于管理：Docker 提供了命令行工具来创建、启动、停止、删除和管理容器。
7. 网络和存储：容器可以配置网络连接和存储卷，以实现数据持久化和网络通信。
8. 可配置：容器可以通过环境变量、配置文件或命令行参数进行配置。
9. 可连接性：容器可以设置端口映射，使得外部可以访问容器内部运行的服务。

创建和运行 Docker 容器的基本命令包括：

- 使用 `docker run` 命令从镜像创建并启动一个新容器。
- 使用 `docker start` 命令启动一个已经停止的容器。
- 使用 `docker stop` 命令停止一个运行中的容器。
- 使用 `docker restart` 命令重启一个容器。
- 使用 `docker rm` 命令删除一个容器。
- 使用 `docker ps` 命令列出当前运行的容器。
- 使用 `docker logs` 命令查看容器的输出日志。

Docker 容器是 Docker 核心功能的一部分，它们使得应用程序的部署和扩展变得简单、灵活和可靠。