## 应用开发概述

### 1、在Docker上开发新应用

如果刚刚开始在Docker上开发全新的应用程序，请查看这些资源以了解一些最常见的模式，以从Docker中获得最大收益。

* 学习 [build an image from a Dockerfile](https://docs.docker.com/get-started/part2/)（从Dockerfile构建镜像）。
* 使用[multistage builds](https://docs.docker.com/engine/userguide/eng-image/multistage-build/)（多阶段构建）来保持镜像精简。
* 使用 [volumes](https://docs.docker.com/engine/admin/volumes/volumes/)（卷）管理应用程序数据并 [bind mounts](https://docs.docker.com/engine/admin/volumes/bind-mounts/)（绑定安装）。
* 使用Kubernetes [Scale your app](https://docs.docker.com/get-started/part3/)（扩展你的应用）。
* [Scale your app](https://docs.docker.com/get-started/part4/) （扩展你的应用）为大量服务。
* 通用应用程序开发最佳实践。

### 2、了解如何使用Docker开发特定语言的应用程序

* [Docker for Java developers](https://github.com/docker/labs/tree/master/developer-tools/java/) lab
* [Port a node.js app to Docker](https://github.com/docker/labs/tree/master/developer-tools/nodejs/porting)
* [Ruby on Rails app on Docker](https://github.com/docker/labs/tree/master/developer-tools/ruby) lab
* [Dockerize a .Net Core application](https://docs.docker.com/engine/examples/dotnetcore/)
* [Dockerize an ASP.NET Core application with SQL Server on Linux](https://docs.docker.com/compose/aspnet-mssql-compose/) using Docker Compose

### 3、使用SDK或API进行高级开发

在你编写Dockerfiles或Compose文件并使用Docker CLI之后，通过使用适用于Go/Python的Docker Engine SDK或直接使用HTTP API将其提升到一个新的水平。

## 应用开发最佳实践

### 1、如何缩小镜像

在启动容器服务时，小镜像可以更快地通过网络并更快地加载到内存中。以下是一些保持镜像小尺寸的经验法则：

* 从一个合适的基础镜像开始。例如，如果你需要一个JDK，考虑使用官方的openjdk镜像，而不是从一个通用的ubuntu镜像开始，然后安装openjdk作为Dockerfile的一部分。
* [使用多级构建](https://docs.docker.com/develop/develop-images/multistage-build/)。例如，可以使用maven镜像构建Java应用程序，然后重置为tomcat镜像并将Java构建复制到正确的位置以部署应用程序，所有这些都是在同一个Dockerfile中。这意味着你的最终镜像不包括由构建引入的所有库和依赖项，只包括运行它们所需的工作和环境。
  + 如果需要使用一个不包含多级构建的Docker版本，那么尝试通过最小化Dockerfile中独立运行命令的数量来减少镜像中的层数。可以将多个命令合并到一个运行中，并使用shell的机制将它们组合在一起。考虑以下两个片段。第一种方法在镜像中创建两个层，而第二种方法只创建一个层。

RUN apt-get -y update

RUN apt-get install -y python

RUN apt-get -y update && apt-get install -y python

* 如果你有多个具有许多共同点的镜像，请考虑使用共享组件创建你自己的基本镜像，并在此基础上创建独特的镜像。Docker只需要加载一次公共层，然后缓存它们。这意味着你的派生镜像可以更有效地使用Docker主机上的内存，加载速度也更快。
* 要保持生产镜像精简，但允许进行调试，可以考虑使用生产镜像之上添加其他测试或调试工具。
* 在构建镜像时，一定要用有用的标记标记它们，这些标记将版本信息、目标（例如，prod或test）、稳定性或其他在不同环境中部署应用程序时有用的信息编入编码。不要依赖自动创建的最新标记。

### 2、在何处以及如何持久化应用数据

* 避免使用 [storage drivers](https://docs.docker.com/engine/userguide/storagedriver/)将应用程序数据存储在容器的可写层中。这增加了容器的大小，并且从I/O的角度来看，它的效率比使用卷或绑定装载要低。
* 使用[volume](https://docs.docker.com/engine/admin/volumes/volumes/)存储数据。
* 在开发期间使用 [bind mounts](https://docs.docker.com/engine/admin/volumes/bind-mounts/)是合适的一种情况，此时您可能希望挂载源目录或刚构建到容器中的二进制文件。对于生产环境，应该使用卷，将其安装到开发期间安装绑定安装相同的位置。
* 对于产品，使用[secrets](https://docs.docker.com/engine/swarm/secrets/)来存储服务使用的敏感应用程序数据，并使用配置文件等非敏感数据来配置配置。如果目前你使用的是独立容器，那么可以考虑迁移到使用单副本服务，这样你就可以利用这些值提供服务的特性。

### 3、使用CL/CD进行测试和部署

* 当你将更改签入源代码控制或创建拉请求时，使用Docker Hub或另一个CI/CD管道自动构建和标记Docker镜像并对其进行测试。
* 在Docker Engine中更进一步，要求你的开发、测试和安全团队在部署到生产环境之前对镜像进行签名。通过这种方式，你可以确保在将镜像部署到生产环境之前，它已经经过了测试，并得到了开发、质量和安全团队的批准。

### 4、开发和生产环境的差异

|  |  |
| --- | --- |
| **开发** | **生产** |
| Use bind mounts to give your container access to your source code. | Use volumes to store container data. |
| Use Docker Desktop for Mac or Docker Desktop for Windows. | Use Docker Engine - Enterprise if possible, with [userns mapping](https://docs.docker.com/engine/security/userns-remap/) for greater isolation of Docker processes from host processes. |
| Don’t worry about time drift. | Always run an NTP client on the Docker host and within each container process and sync them all to the same NTP server. If you use swarm services, also ensure that each Docker node syncs its clocks to the same time source as the containers. |