

# 第12期-容器化Docker应用开发流程

2020年2月22日 18:32

## 常用 Docker 命令行

### 映像常用命令

仓库下载映像: `docker pull <image name>`

查看本地映像: `docker images`

删除指定映像: `docker rmi <image id>`

删无标签映像: `docker rmi $(docker images | grep "^<none>" | awk "{print $3}")`

删除全部映像: `docker rmi $(docker images -q)`

### 容器常用命令

运行容器: `docker run -d -p 5001:80 <image name>`

查看运行容器: `docker ps`

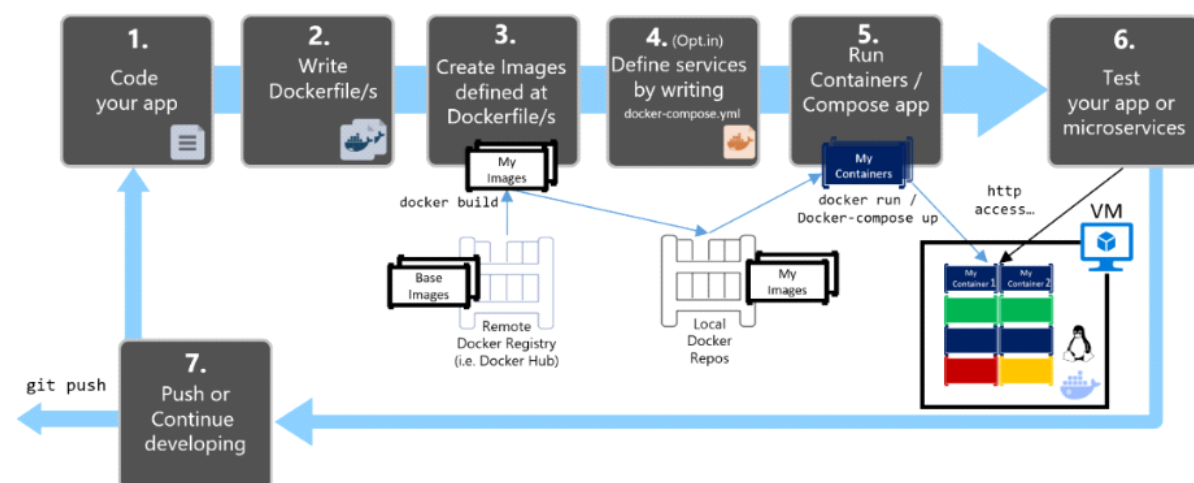
查看所有容器: `docker ps -a`

进入容器内部: `docker exec -it <container id>/bin/bash`

停止全部容器: `docker stop $(docker ps -q)`

删除所有容器: `docker rm $(docker ps -aq)`

## 开发基于 Docker 容器的应用程序的工作流



## 步骤1: 构建开发环境

[Get started with Docker CE for Windows](#)



## .NET Core 跨平台开发

使用 .NET Core、ASP.NET Core、HTML/JavaScript 和包括 Docker 支持的容器生成跨平台应用程序。

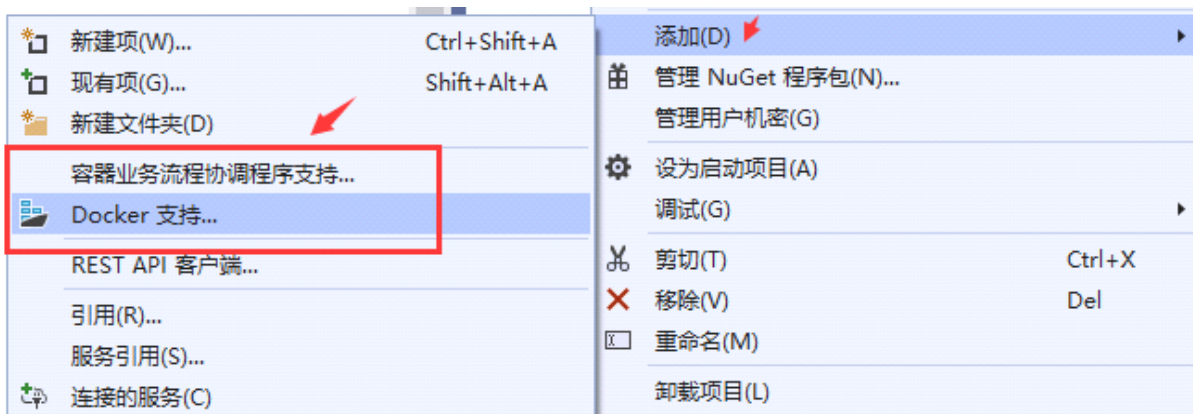


## 步骤2：创建支持 Docker 容器镜像生成的项目

### 新项目直接选择



### 在现有项目上右击添加



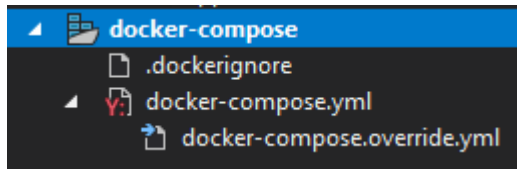
### 微软 .NET Core 映像仓库

[https://hub.docker.com/\\_/microsoft-dotnet-core](https://hub.docker.com/_/microsoft-dotnet-core)

### 步骤3：创建项目的 Docker 映像

- 配置文件所在目录执行命令：docker build 或 docker-compose up --build
- 在 Visual Studio 中按下 F5（或 Ctrl-F5）创建映像并在容器中运行。

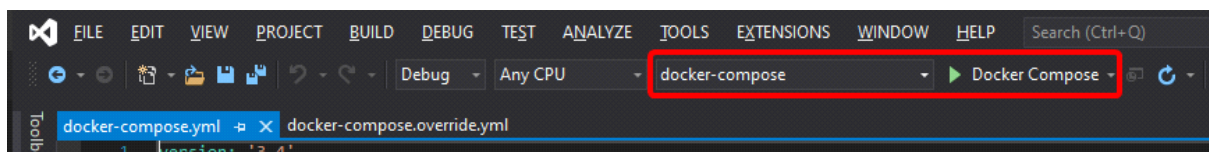
### 步骤4：使用 docker-compose 管理多个容器



### 步骤5：启动容器运行应用程序

运行单个容器：docker run -t -d -p 80:5000 <image name>

运行多个容器：docker-compose up



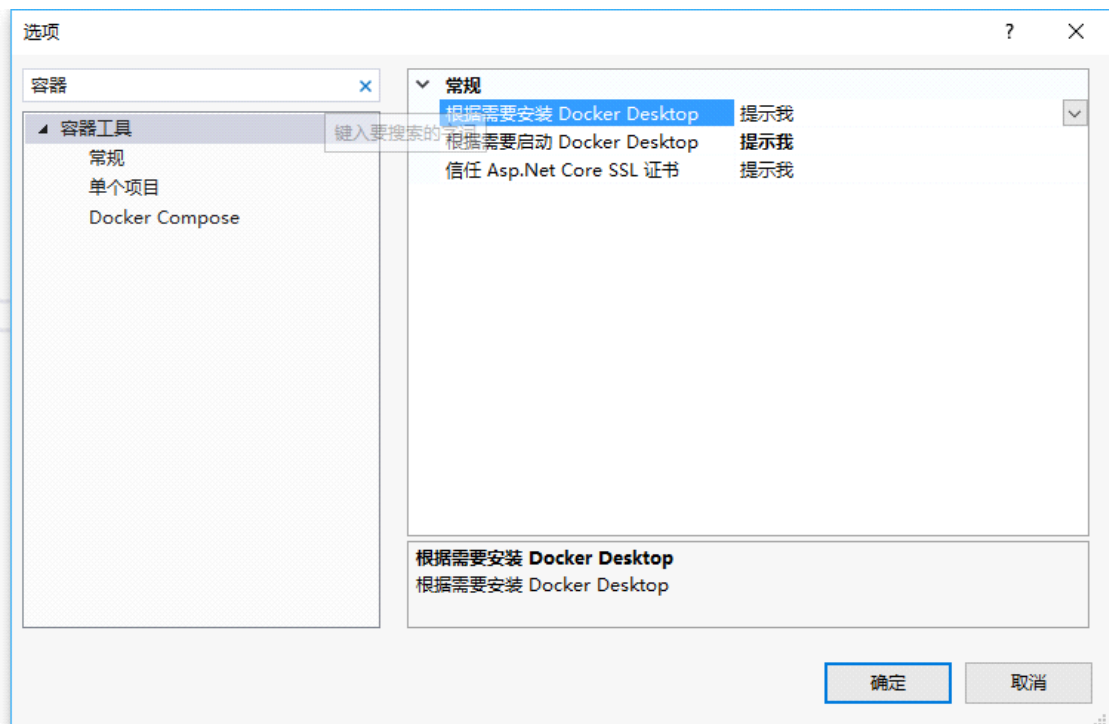
### 环境变量与配置文件

配置环境变量：ASPNETCORE\_ENVIRONMENT= Development or Production

启动多个容器：docker-compose -f docker-compose.yml -f docker-compose.prod.yml  
up

环境配置文件：docker-compose.vs.release.yml 或 docker-compose.vs.debug.yml

### Visual Studio 中的相关设置



## Visual Studio 中的容器工具

来自 <<https://docs.microsoft.com/zh-cn/visualstudio/containers>>