

# 一：快速上手完成Docker镜像的创建

## 1.确定目标：

打包node环境与一个js文件（实现console.log("Docker镜像加载成功")）。

### 为什么需要打包成docker镜像？

我的本地没有安装Node.js环境。因此javascript文件无法被直接运行

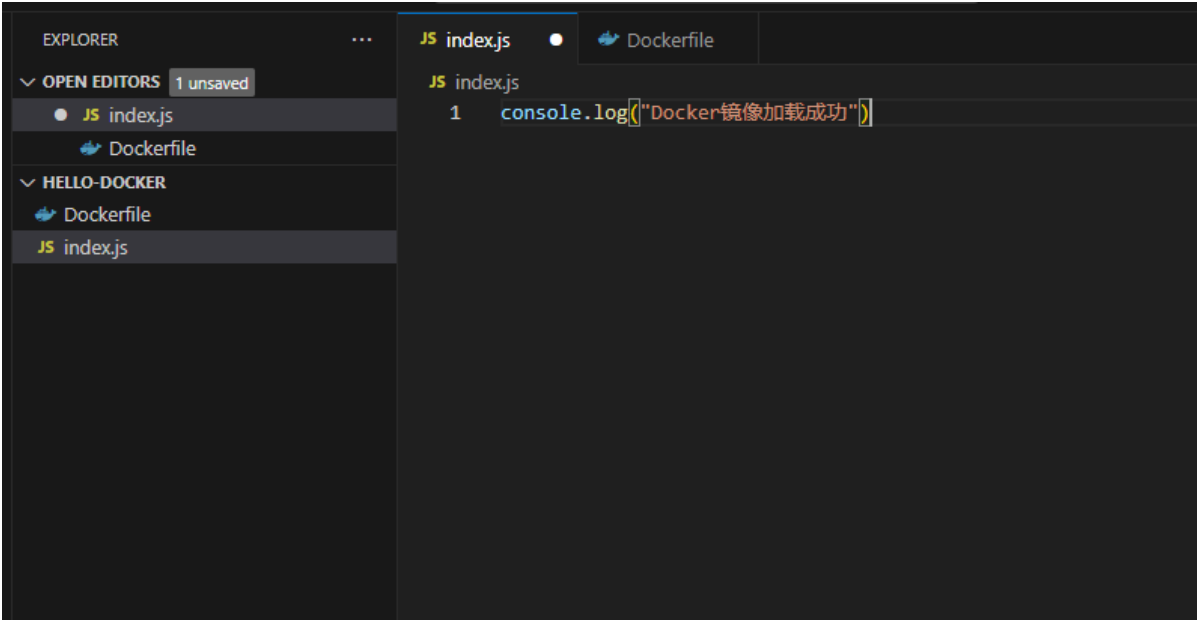
```
node index.js
```

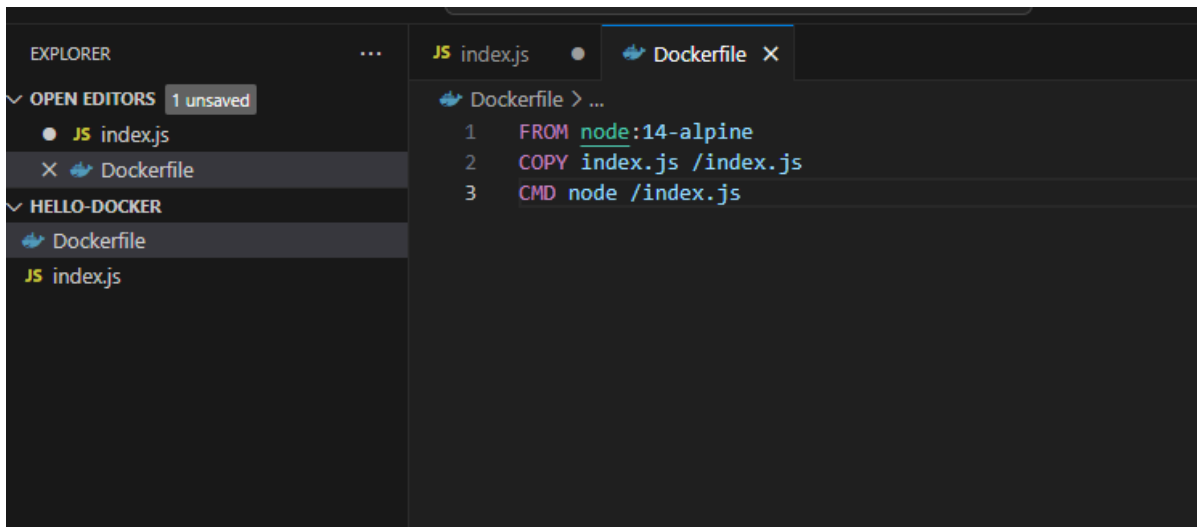
### 解决方案：

将index.js文件与node打包成一个镜像,直接部署镜像即可.

**注意问题：** node无法直接存在,需要依托操作系统,因此我们需要先构建linux的发行版,再于这个基础上构建node.最后打包我们的js文件.

## 2.js文件与Dockerfile文件的





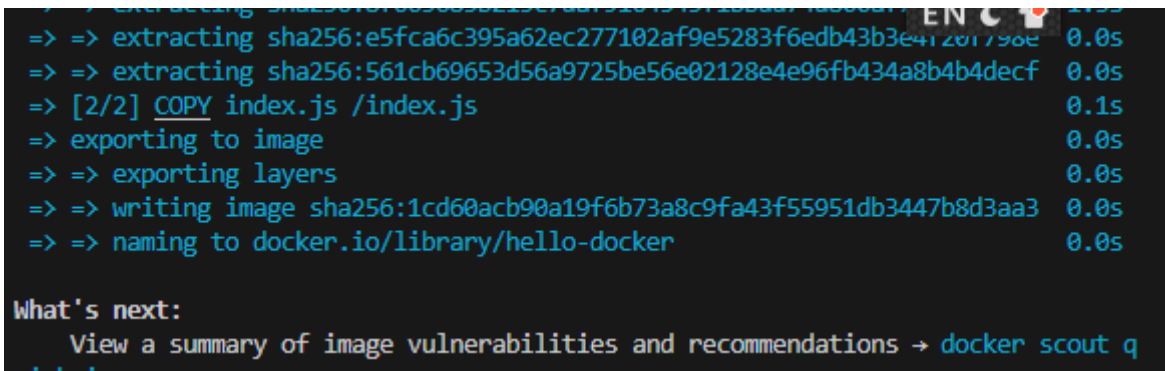
我们推荐使用Vscode打开并且进行编辑，将代码实现到以上形式

这里FROM的是基于alpine构建的Node.alpine是一个Linux发行版，相较于Ubuntu等镜像，体积非常小，适合快速部署。其体积只有几十MB左右

### 3.打包镜像

打开控制台输入

```
docker build -t hello-docker .
```

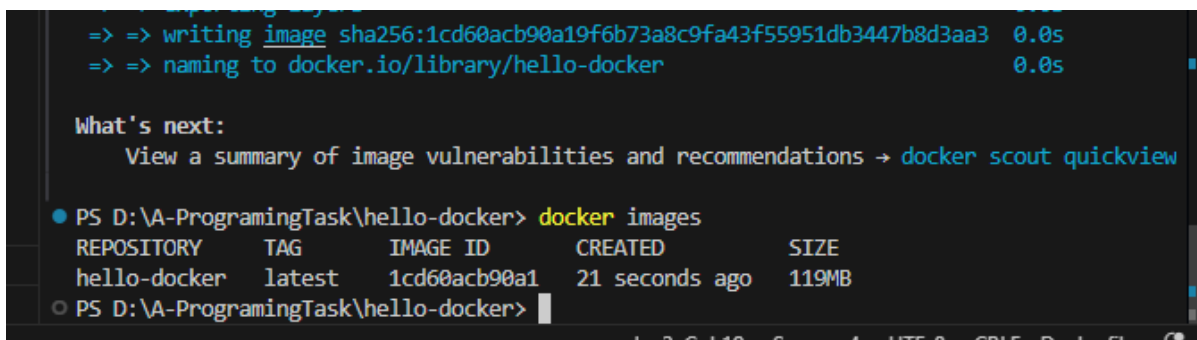


出现上图所示则打包成功

### 4.查询所有镜像

```
docker images
```

我们可以发现：刚刚被打包的镜像已经被出现在下方了



```
~/Desktop/HelloDocker docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-docker	latest	52e3ddcec2db	2 minutes ago	119MB
<none>	<none>	785487be3453	25 hours ago	119MB
xrsec/awvs	latest	1b9a70057987	7 weeks ago	1.77GB
yrzr/gitlab-ce-arm64v8	latest	83ca55befda4	8 weeks ago	2.86GB
oceanbase/oceanbase-ce	latest	961226c97a93	2 months ago	548MB
redis	latest	edf4b3932692	2 months ago	111MB
nginx	latest	114aa6a9f203	2 months ago	135MB
ubuntu	latest	730eeb702b69	2 months ago	69.2MB
kalilinux/kali-rolling	latest	b298a7e01984	3 months ago	139MB
mysql	latest	ad405c51acf6	3 months ago	544MB
zhusaidong/nacos-server-m1	2.0.3	f1d91cde7e60	22 months ago	1.12GB

TAG:可以自定义的版本。默认是最新，也可以自定义为 x.x.x的形式

## 5.运行镜像

```
docker run 镜像名
```

```
PS D:\A-ProgramingTask\hello-docker> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
hello-docker latest 1cd60acb90a1 8 minutes ago 119MB
PS D:\A-ProgramingTask\hello-docker> docker run hello-docker
```

## 6.拉取镜像

```
docker pull geekhour/hello-docker
```

在拉取之后 我们查看是否拉取成功

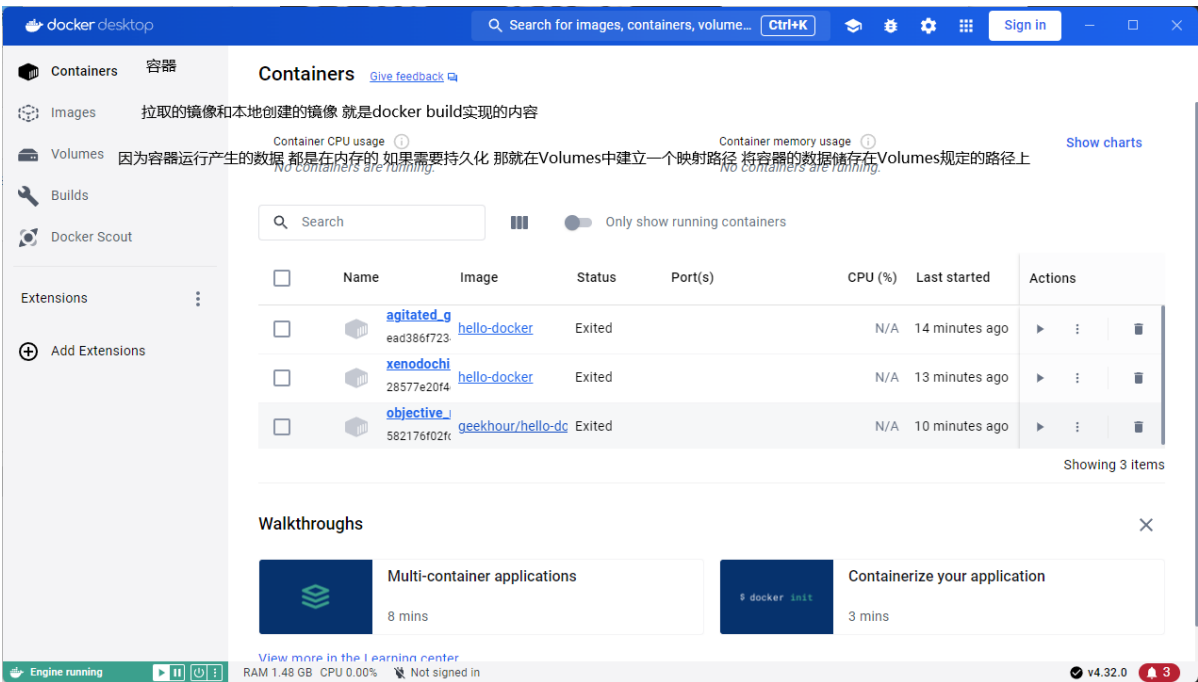
```
docker images //查看所有的镜像
```

```
View a summary of image vulnerabilities and recommendations → docker scout quickv
geekhour/hello-docker
PS D:\A-ProgramingTask\hello-docker>
PS D:\A-ProgramingTask\hello-docker> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
hello-docker latest 1cd60acb90a1 10 minutes ago 119MB
geekhour/hello-docker latest a952aa943afc 14 months ago 119MB
PS D:\A-ProgramingTask\hello-docker> docker run geekhour/hello-docker
欢迎来到一小时Docker教程，拜托一键三连了！
PS D:\A-ProgramingTask\hello-docker>
```

```
docker run geekhour/hello-docker
```

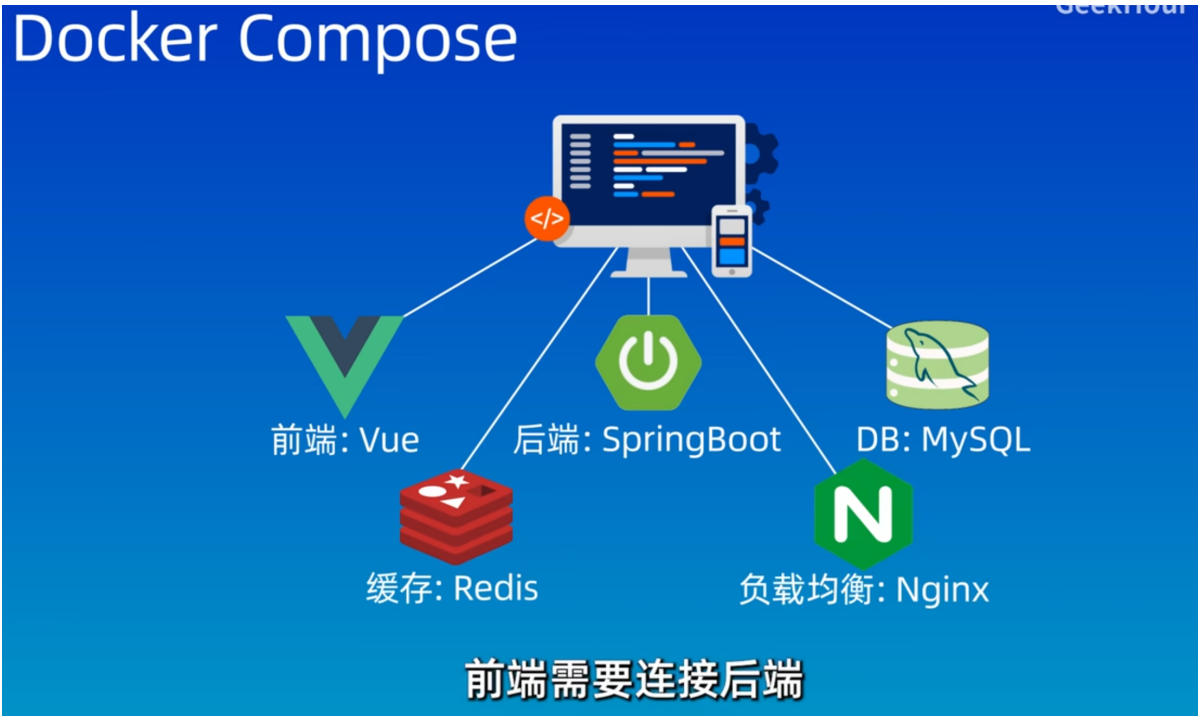
我们发现，镜像已经被成功的拉取并且执行了。这个操作是不是很像github的拉取操作呢？

## 二：关于DockerDesktop



## 三： DockerCompose

一个网站会用到前端，后端，数据库。问题是：用到大量的中间件：例如Redis mysql MongoDB。在微服务项目中，会涉及到更多的外部组件。那如何同一对它们进行管理呢？



# Docker Compose

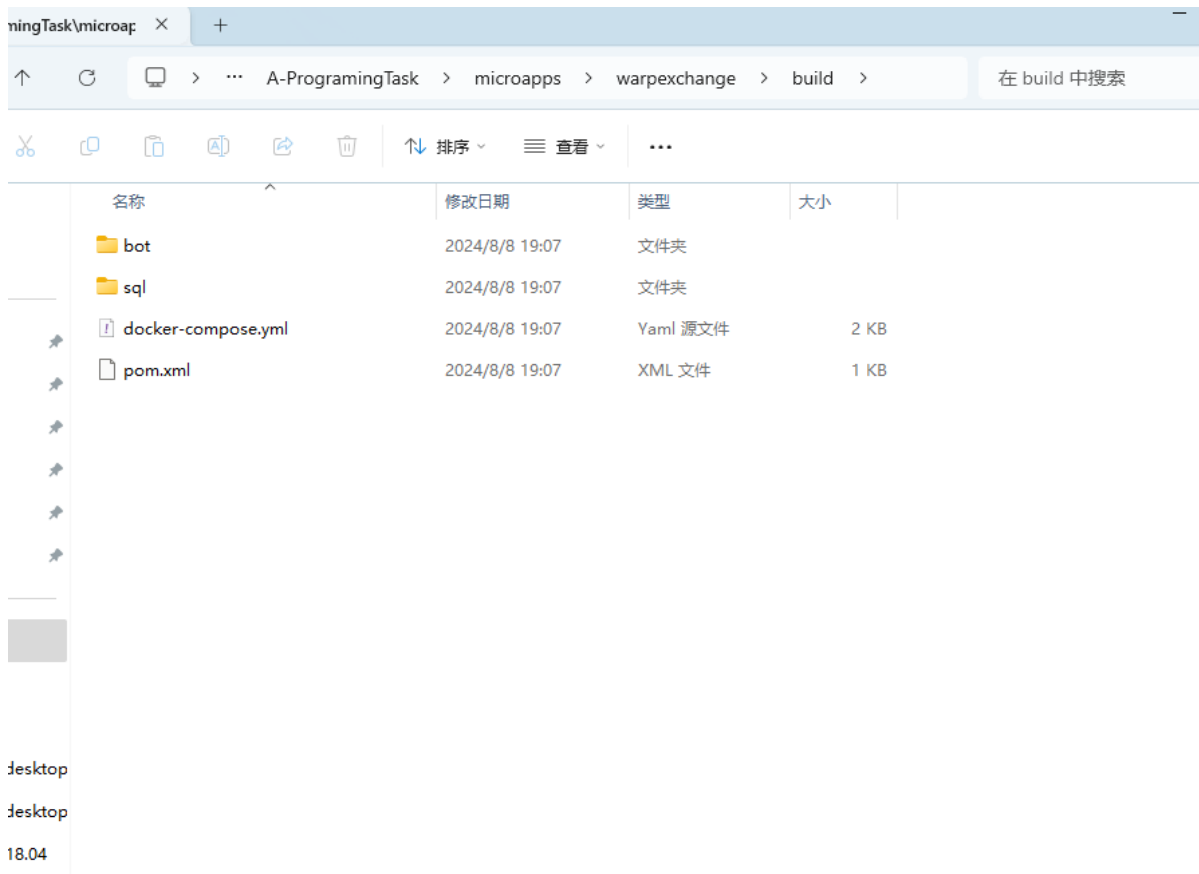


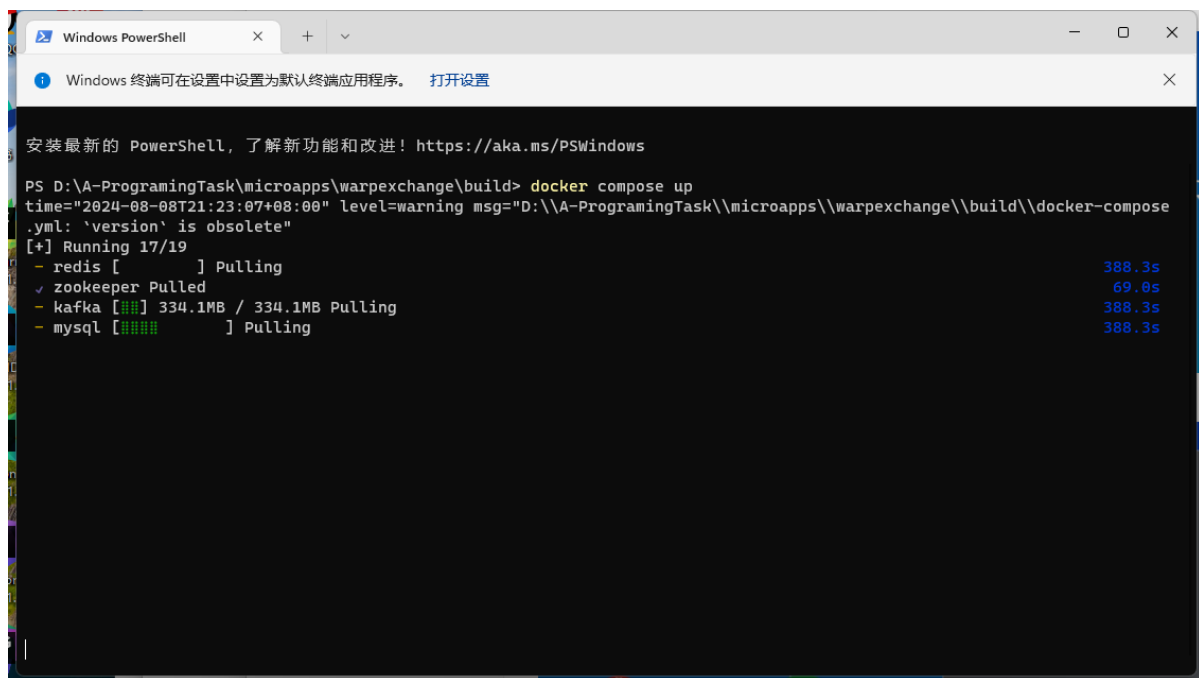
docker-compose.yml

```
redis:
  image: redis
db:
  image: mysql
frontend:
  - 3000:3000
links:
  - redis
backend:
  image: springbot-app
.....
```

\$ docker compose up

下面看一个具体例子 在有docker-compose.yml的目录下启动cmd





The screenshot shows a Windows PowerShell terminal window with a dark background. At the top, there's a title bar for 'Windows PowerShell' and a notification bar that says 'Windows 终端可在设置中设置为默认终端应用程序。 打开设置'. Below the notification, there's a message about installing the latest PowerShell. The main content is the output of the command 'docker compose up'. It shows a warning about an obsolete version in the compose.yml file, followed by progress bars for pulling Redis, Zookeeper, Kafka, and MySQL images. The progress bars are represented by green and blue blocks, and the time taken for each pull is shown on the right.

```
Windows PowerShell
Windows 终端可在设置中设置为默认终端应用程序。 打开设置

安装最新的 PowerShell, 了解新功能和改进! https://aka.ms/PSWindows

PS D:\A-ProgramingTask\microapps\warpxexchange\build> docker compose up
time="2024-08-08T21:23:07+08:00" level=warning msg="D:\\A-ProgramingTask\\microapps\\warpxexchange\\build\\docker-compose
.yml: 'version' is obsolete"
[+] Running 17/19
- redis [ ] Pulling 388.3s
✓ zookeeper Pulled 69.0s
- kafka [ ] 334.1MB / 334.1MB Pulling 388.3s
- mysql [ ] Pulling 388.3s
```

即可在本地部署镜像了