

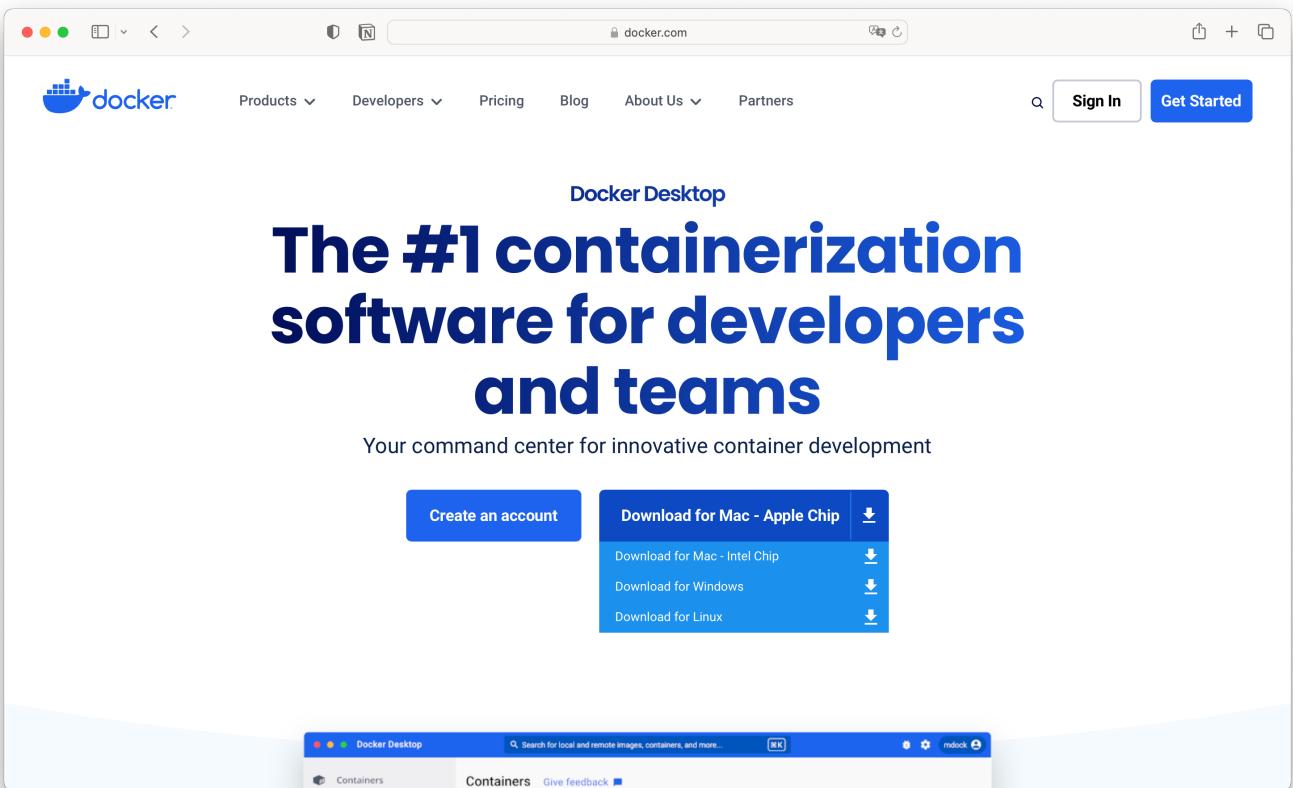
DaSE CSAPP VSCode on Docker

CSAPP Dockerfile for DaSE

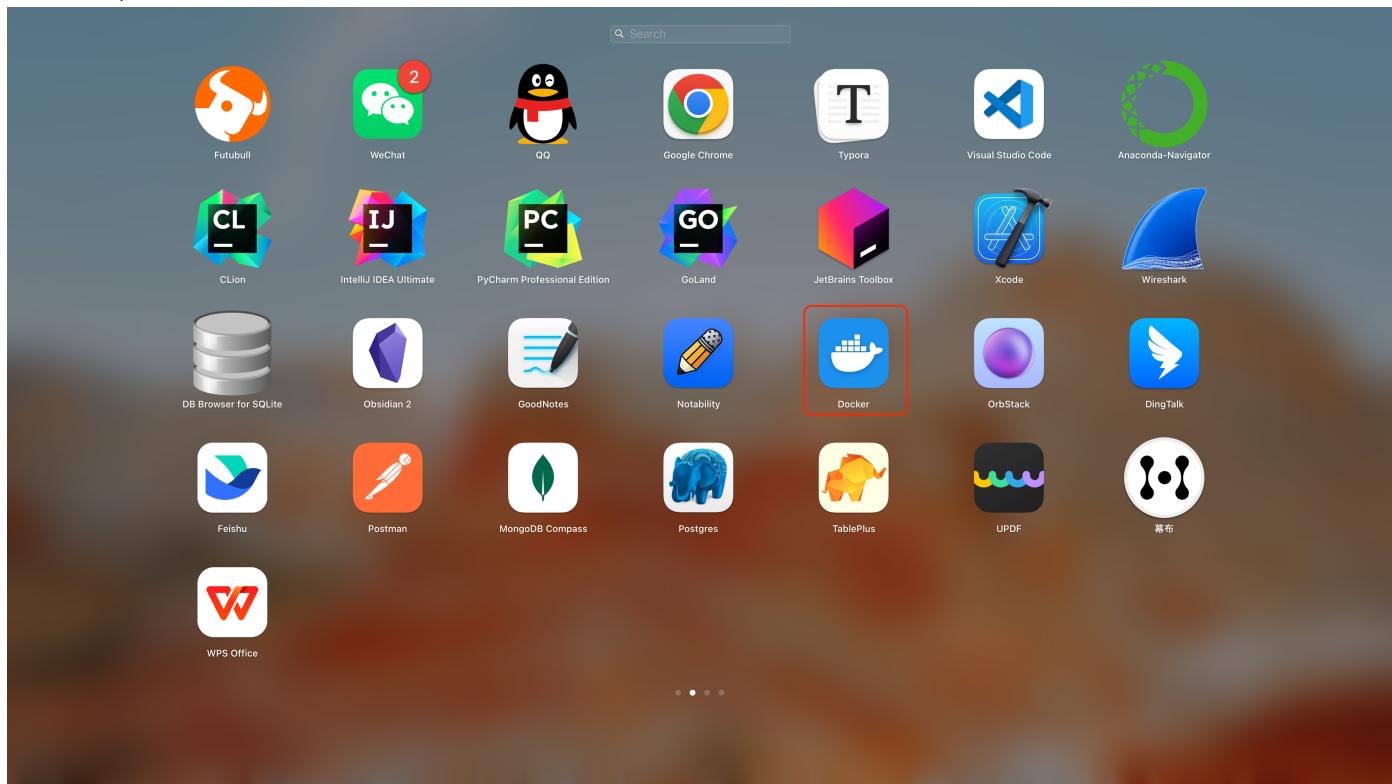
使用方法

安装本地的docker desktop

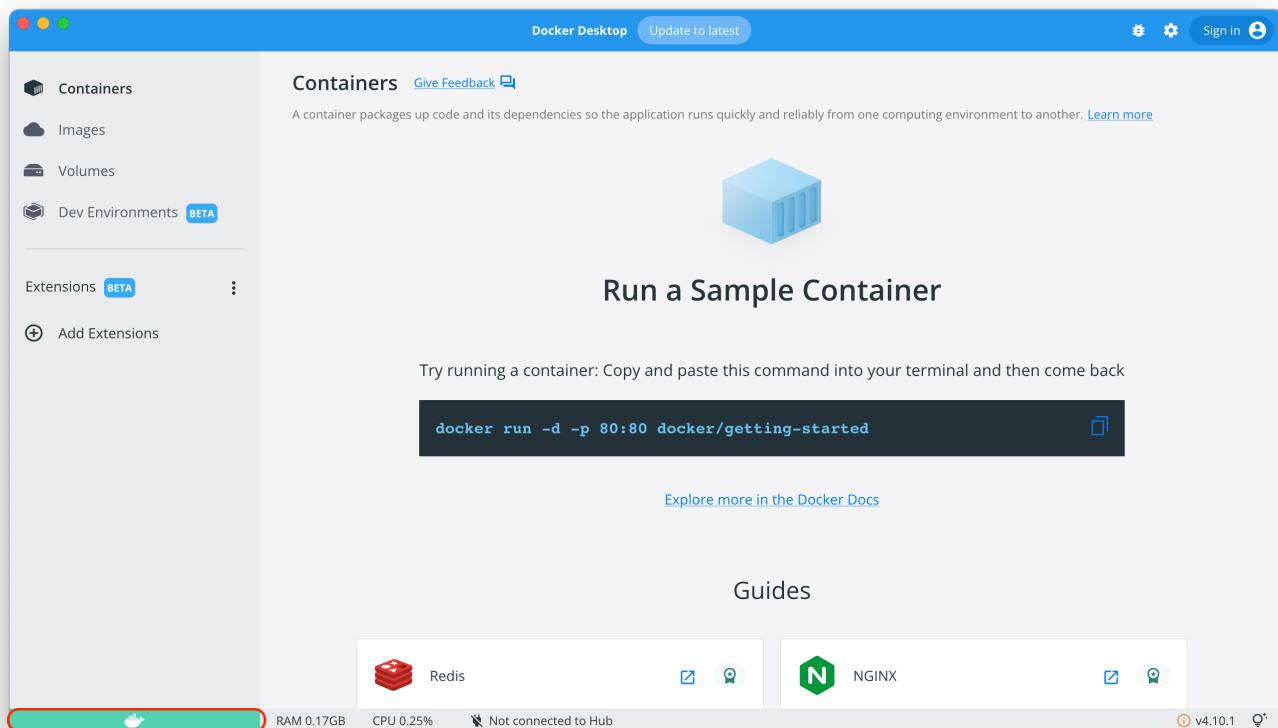
下载链接：<https://www.docker.com/products/docker-desktop/>，根据您使用的操作系统选择对应的版本下载，如下图所示。



下载完后应该可以看到长成这样很可爱的小鲸鱼的图标，然后让我们点击它。点击之后，docker的启动会需要一点时间，所以给点时间给我们可爱的小鲸鱼。



当你看到docker desktop的左下角的小鲸鱼图标变成绿色的时候，就代表它已经启动了，如下图所示。【括号里的内容可以略过：变为绿色代表docker dameon守护进程已经启动，相当于你手动在本机的命令行中输入了 `systemctl start docker`】



好，那到目前为止，我们安装docker小鲸鱼的过程就结束了，接下来我们去把小鲸鱼的食物（dockerfile）给下载下来吧，见后面的小节。

安装Git Large File Storage

下载指导链接: <https://git-lfs.com/>

Github官方教程链接: <https://docs.github.com/en/repositories/working-with-files/managing-large-files/installing-git-large-file-storage>

The screenshot shows the official website for Git Large File Storage (git-lfs.com). At the top, there's a navigation bar with links to Docs, Discussions, Wiki, Installation, Releases, and Source. Below the navigation, a heading reads "An open source Git extension for versioning large files". A text block explains that Git Large File Storage (LFS) replaces large files like audio samples and videos with text pointers in Git, while storing the file contents on a remote server. It includes download links for Mac - Intel Silicon (v3.4.0) and Mac - Apple Silicon (v3.4.0), along with Homebrew and MacPorts installation commands. To the right, a diagram illustrates the LFS architecture: a "Local" machine connects to a "Remote" server, which then connects to a "Large File Storage" database. A file named "file.psd" is shown being stored in the storage. Below the diagram, a note about a security update for Windows users is visible.

因为众所周知的原因，我们下载某些软件总是会面临中断、连不上、网络错误等失败，所以我们事先把要用Docker中要使用到的软件包下载下来并放到了Github仓库中，但是由于这个包有97.3MB的大小，属于大型文件，所以在Github仓库中上传和下载时都会有一些不同的地方，比如原本的`git clone`指令就要变成`git lfs clone`。所以为了能够支持大型文件的下载克隆，我们需要实现安装一个git-lfs如上图的网址所示，如何下载会根据你的电脑操作系统不同而有所区别，上图为我的mac电脑的安装方式。

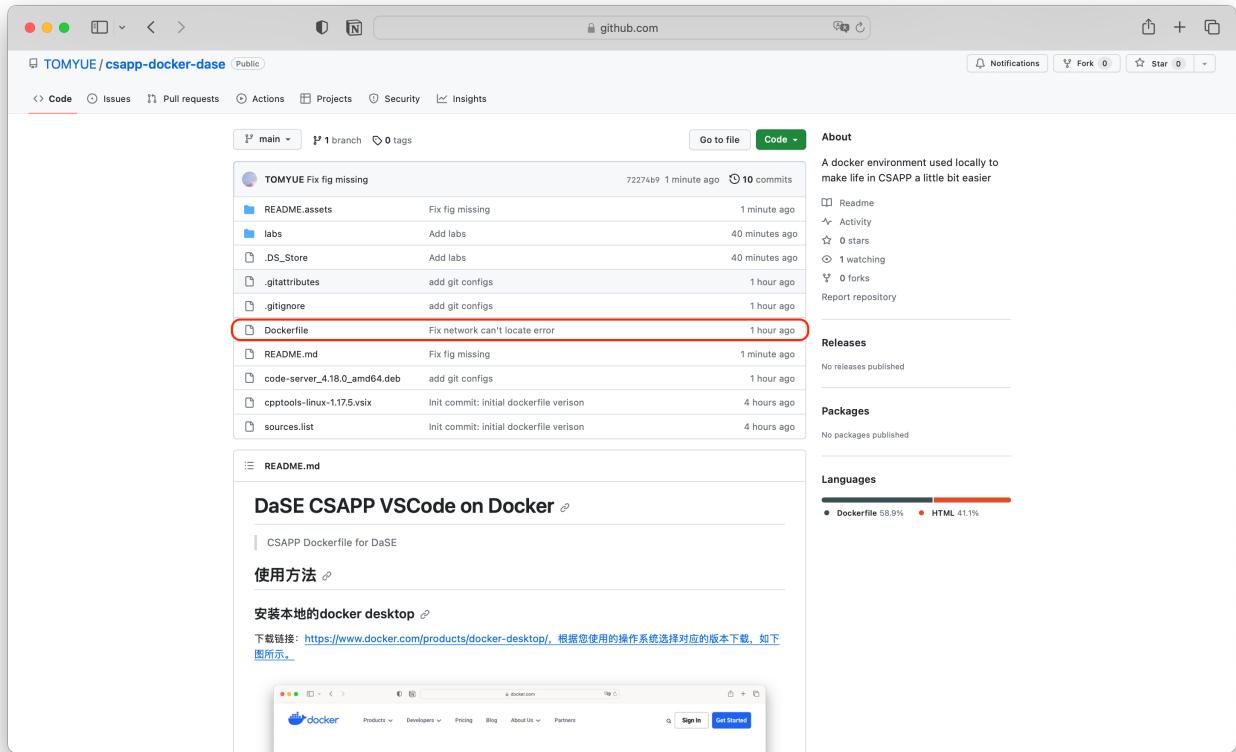
构建Docker镜像依据我们编写的Dockerfile

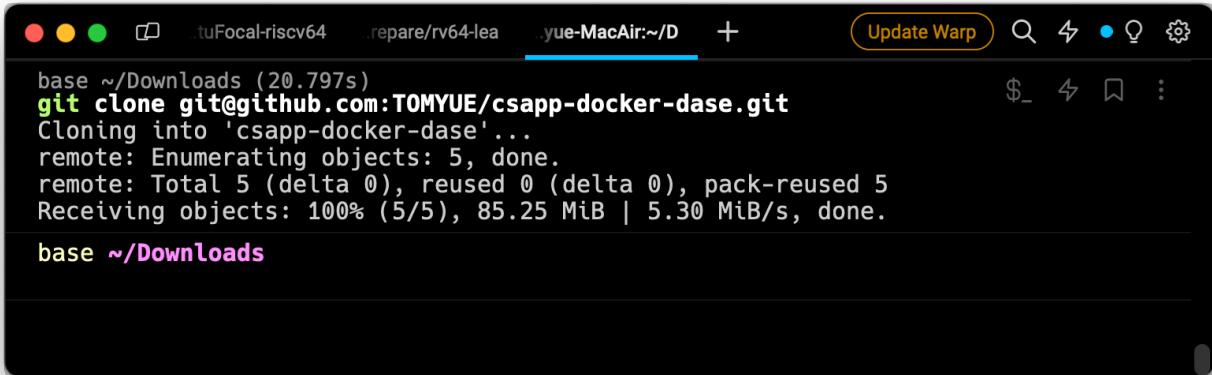
命令：

```
> git lfs install # 使用此指令确认你已经安装了git lfs  
> git lfs clone https://github.com/TOMYUE/csapp-docker-dase.git csapp-docker  
> cd csapp-docker  
> docker build --platform linux/amd64 -t csapp .  
#这条命令会让docker默认按照当前目录下的Dockerfile启动Docker容器，并且指定在amd64架构下
```

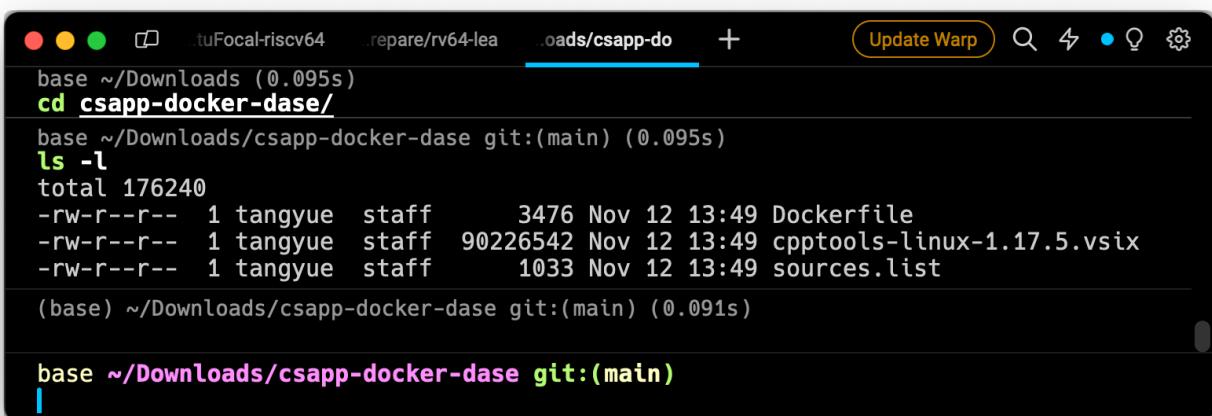
命令解释：

1. 克隆仓库到本地：`git lfs clone https://github.com/TOMYUE/csapp-docker-dase.git` 或者如果你配置过Github的ssh key的话使用`git lfs clone git@github.com:TOMYUE/csapp-docker-dase.git`，在不能科学上网的情况下，ssh也是可以很快速的使用的，没配置过的话推荐配置一下，因为仅依靠https下载是个玄学事件。同时这份clone任务不会很快能执行完，因为其中包含了一份我事先加入的`cpptools-linux-1.17.5.vsix`的给vscode code_server的插件，这份插件比较大所以麻烦各位耐心了。





```
base ~/Downloads (20.797s)
git clone git@github.com:TOMYUE/csapp-docker-dase.git
Cloning into 'csapp-docker-dase'...
remote: Enumerating objects: 5, done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 5
Receiving objects: 100% (5/5), 85.25 MiB | 5.30 MiB/s, done.
base ~/Downloads
```



```
base ~/Downloads (0.095s)
cd csapp-docker-dase/
base ~/Downloads/csapp-docker-dase git:(main) (0.095s)
ls -l
total 176240
-rw-r--r-- 1 tangyue staff      3476 Nov 12 13:49 Dockerfile
-rw-r--r-- 1 tangyue staff  90226542 Nov 12 13:49 cpptools-linux-1.17.5.vsix
-rw-r--r-- 1 tangyue staff     1033 Nov 12 13:49 sources.list
(base) ~/Downloads/csapp-docker-dase git:(main) (0.091s)

base ~/Downloads/csapp-docker-dase git:(main)
```

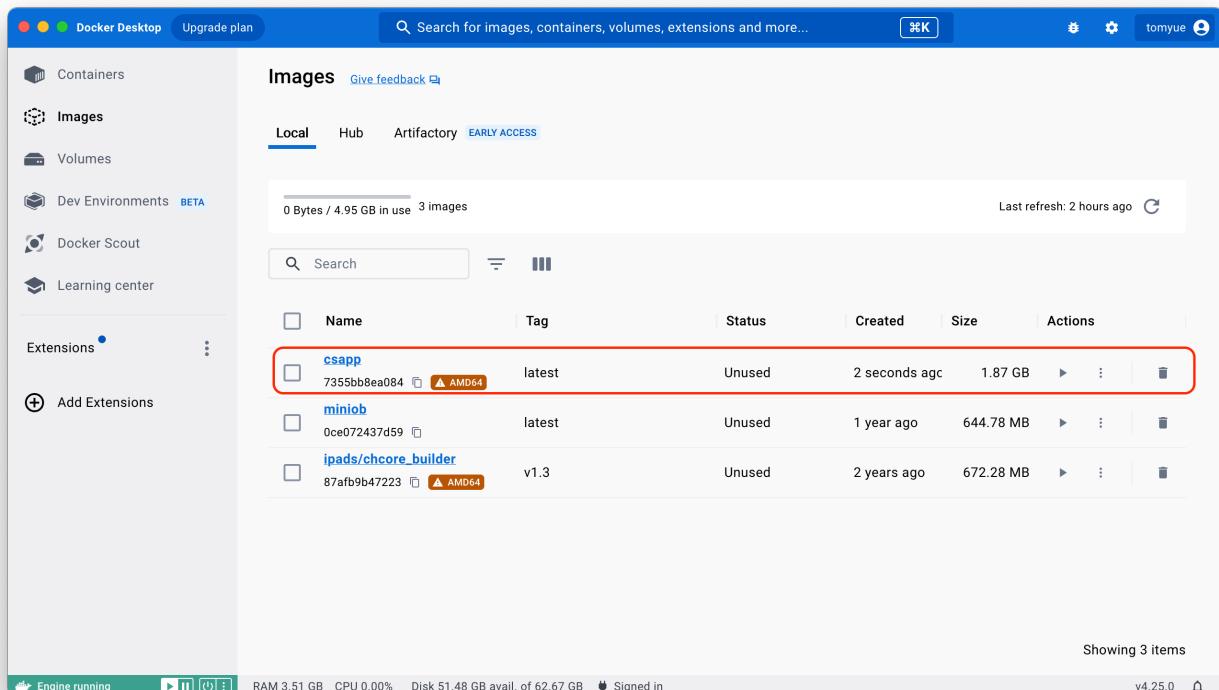
2. 构建并运行docker: `docker build --platform linux/amd64 -t csapp .`

命令本身上，在这里我们使用 `--platform` 标志：在 `docker build` 命令中使用 `--platform linux/amd64` 标志来明确指定要构建的是 `amd64` 平台的镜像。`-t csapp`：是一个选项，用于为构建的镜像指定一个标签（tag）。在这里，`csapp` 是镜像的名称，我们可以根据需要自定义，比如你取名为 `-t ILoveYou` 或者 `-t LOL` 都是可以的，随你啦。总之，标签是用来标识镜像的可读性标识符，最好取名为有意义方便你快速识别的名称。

对于这条命令具体做了什么，简单来说就是按照我在 `dockerfile` 里写的一行行的指令去执行对应的操作，仅此而已，没有任何神秘的地方。

这里放心大胆的敲这行命令就行了，不要恐惧，需要注意⚠的是这个运行可能会有点长（我自己运行了10min左右，627.4s），所以耐心一点。下图是整个运行过程的记录截图，你可以观察一下，其中还有每一步构建所花费的时间，可以看一下：

构建好后，在你的Docker Desktop中你还能看到多出来一个镜像(Docker Image)，如下图所示：



那么接下来就是我们的最后一步了，把它给跑起来，看下一节😊。

运行我们构建的镜像

我们接下来输入这行命令来运行我们的镜像，不够先别当急急国王，我们先看看这行命令的意思。

```
docker run -p 8765:8765 -v "./labs:/home/csapp/project" csapp
```

我们可以将Docker当作一个能够在你的计算机上创建小型、隔离的环境的魔法工具。这个命令，`docker run -p 8765:8765 -v "./labs:/home/csapp/project" csapp`，就是在使用这个魔法工具来做两件事情：

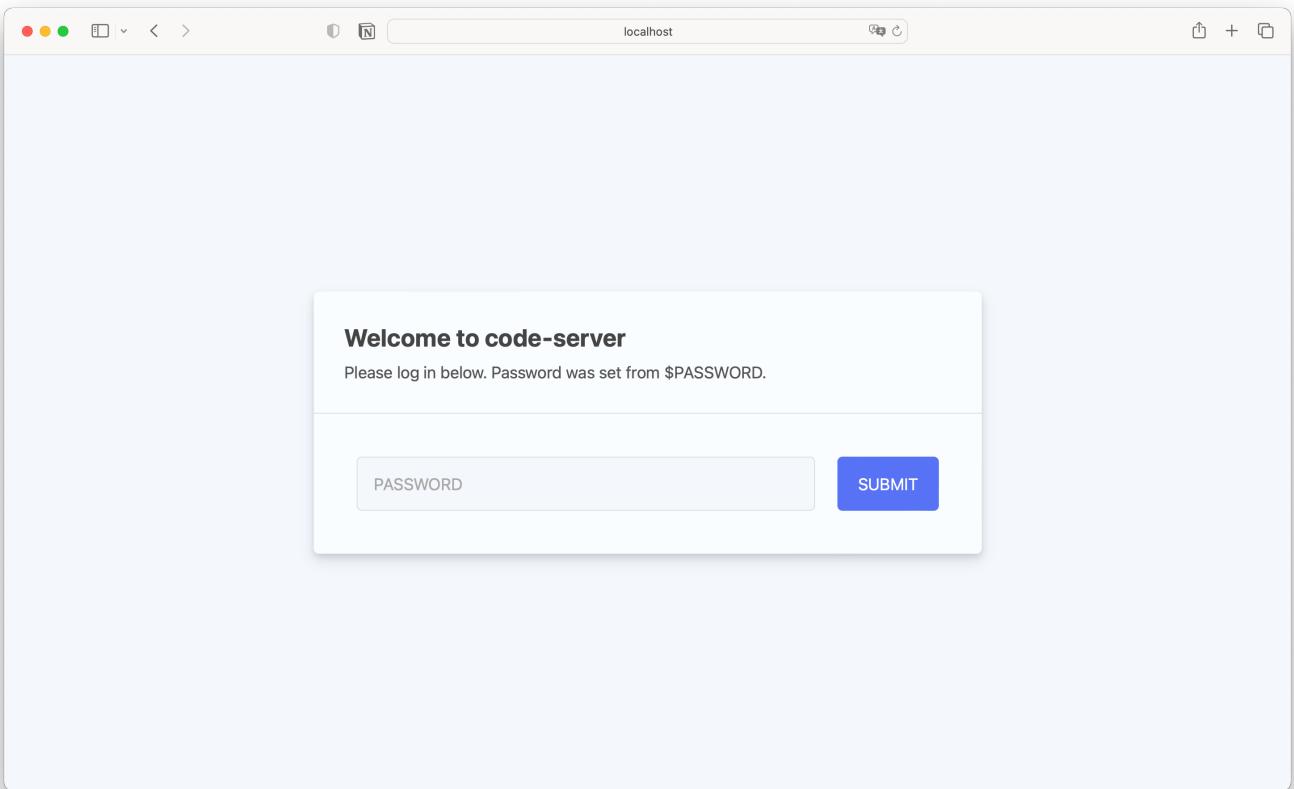
1. `-p 8765:8765`：这部分就像是在为这个小环境（我们称之为容器）设置一个小窗口。这个窗口连接着外面的世界（你的电脑）和容器的内部世界。通过这个窗口，任何访问你电脑上 8765 端口的请求都会被直接传递到容器内的 8765 端口，反之亦然。
2. `-v "./labs:/home/csapp/project"`：这一部分是在说，“嘿 Docker，我有一个文件夹叫做 `labs`，我想在容器里也能访问到它。”这样，Docker 就会确保你的 `labs` 文件夹（位于当前目录，即 `./labs`）在容器内部以 `/home/csapp/project` 的形式出现。这就像是在两个世界之间建立了一个小桥，让它们能够共享同一组文件。

最后的 `csapp` 是告诉 Docker，“我想运行一个名叫 `csapp` 的环境。”，这个 `csapp` 就是我们在上一节中构建 (build) 的 Docker 镜像 Image。

OK，那么现在我们来运行这个命令吧。输入这行命令后，我们能看到如下的输出，仔细阅读其中的意思，我们可以看到，在容器内部的 <http://0.0.0:8765/> 和我们电脑本机的 `localhost:8765` 之间建立了映射关系。

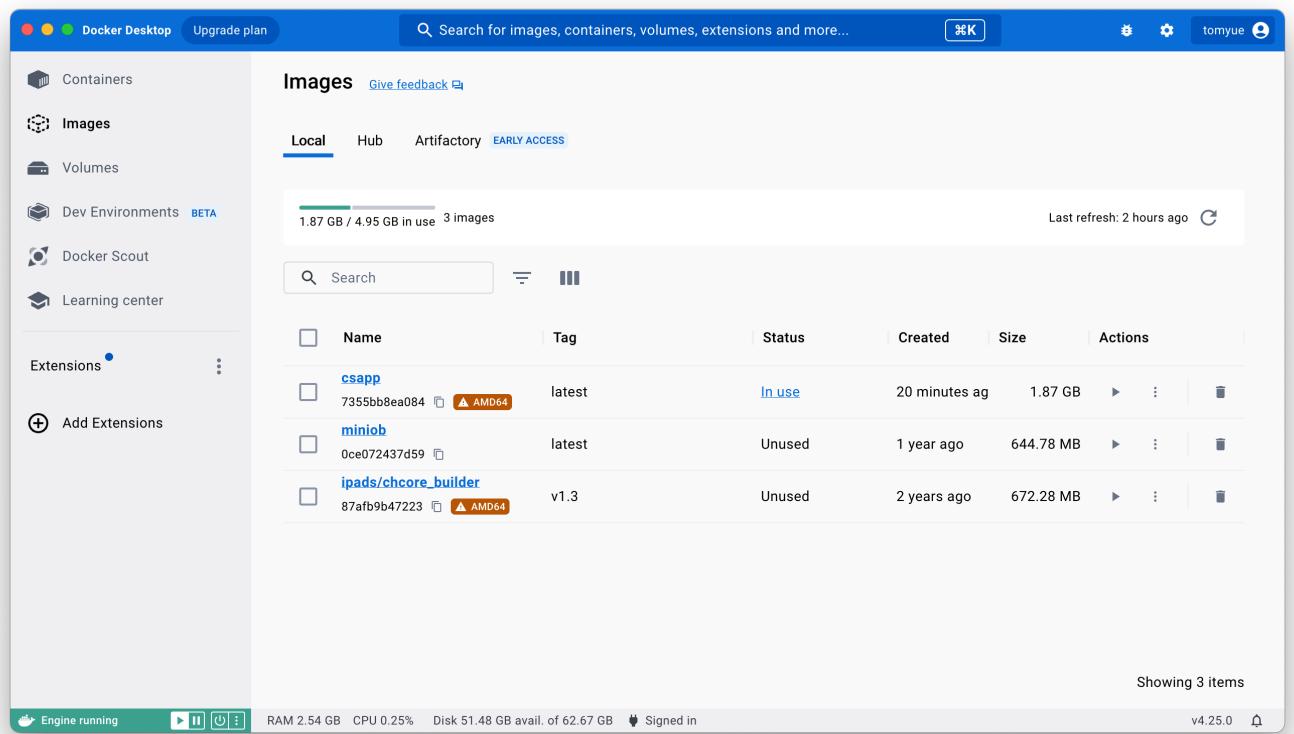
```
base ~/Downloads/csapp-docker-dase git:(main)
docker run -p 8765:8765 -v "./labs:/home/csapp/project" csapp
WARNING: The requested Image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested
[2023-11-12T09:01:50.536Z] info code-server 4.18.0 d7a2b4936af1bf80cb96f2567af68badc2325e3
[2023-11-12T09:01:50.540Z] info Using user-data-dir /home/csapp/.local/share/code-server
[2023-11-12T09:01:50.647Z] info Using config file /home/csapp/.config/code-server/config.yaml
[2023-11-12T09:01:50.648Z] info HTTP server listening on http://0.0.0.0:8765/
[2023-11-12T09:01:50.648Z] info - Authentication is enabled
[2023-11-12T09:01:50.649Z] info - Using password from $PASSWORD
[2023-11-12T09:01:50.649Z] info - Not serving HTTPS
[2023-11-12T09:01:50.649Z] info Session server listening on /home/csapp/.local/share/code-server/code-server-ipc.sock
```

然后我们在一个浏览器中输入 `localhost:8765`，然后我们可以看到浏览器中弹出来如下的界面，而在我们的命令行中也出现了 `[09:04:54] Extension host agent started.` 的信息。同时这里启动后，我们在 Docker Desktop 中也可以看到 In use 的显示。

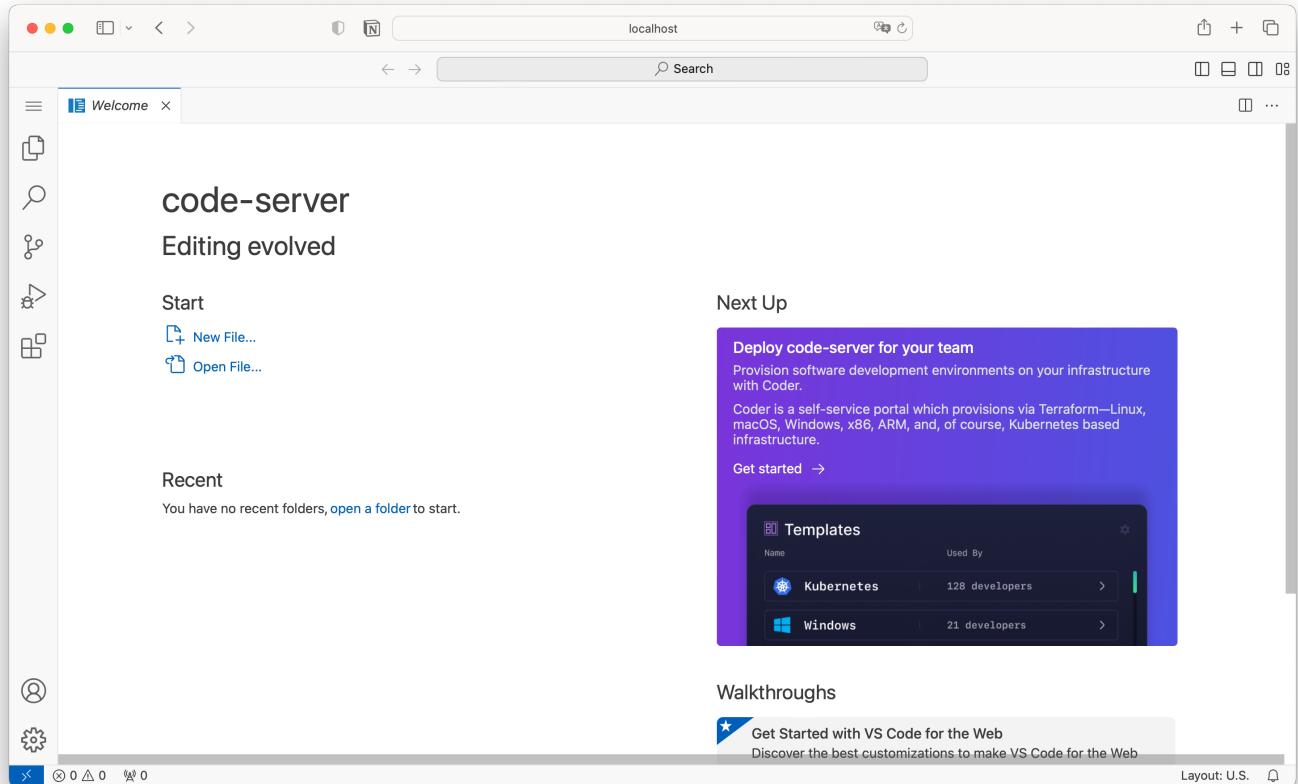


```
base ~/Downloads/csapp-docker-dase git:(main)
docker run -p 8765:8765 -v "./Labs:/home/csapp/project" csapp
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested
[2023-11-12T09:01:50.536Z] info  code-server 4.18.0 d7a2b4936af1bfd80cb96f2567af68badc2325e3
[2023-11-12T09:01:50.540Z] info  Using user-data-dir /home/csapp/.local/share/code-server
[2023-11-12T09:01:50.647Z] info  Using config file /home/csapp/.config/code-server/config.yaml
[2023-11-12T09:01:50.648Z] info  HTTP server listening on http://0.0.0.0:8765/
[2023-11-12T09:01:50.648Z] info    - Authentication is enabled
[2023-11-12T09:01:50.649Z] info    - Using password from $PASSWORD
[2023-11-12T09:01:50.649Z] info    - Not serving HTTPS
[2023-11-12T09:01:50.649Z] info  Session server listening on /home/csapp/.local/share/code-server/code-server-ipc.sock
[09:04:54]
```

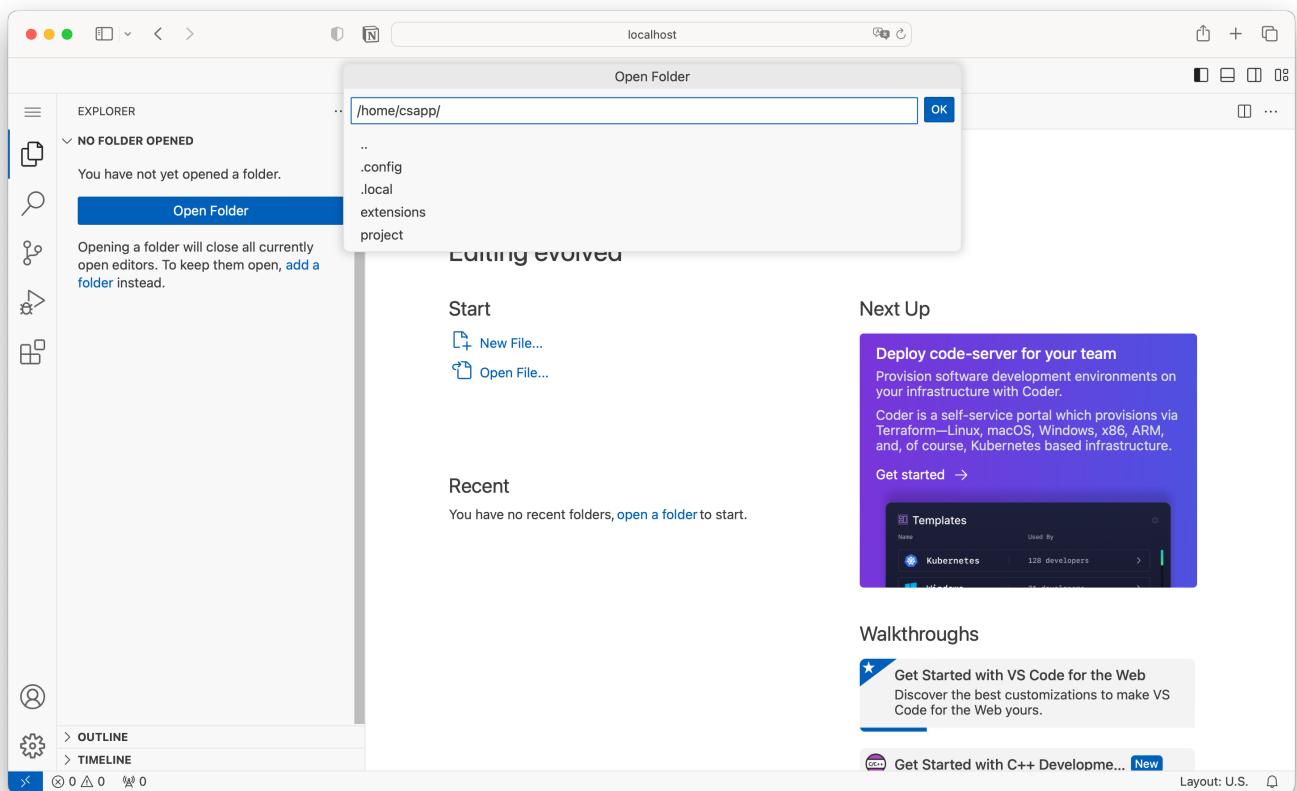
[09:04:54] Extension host agent started.



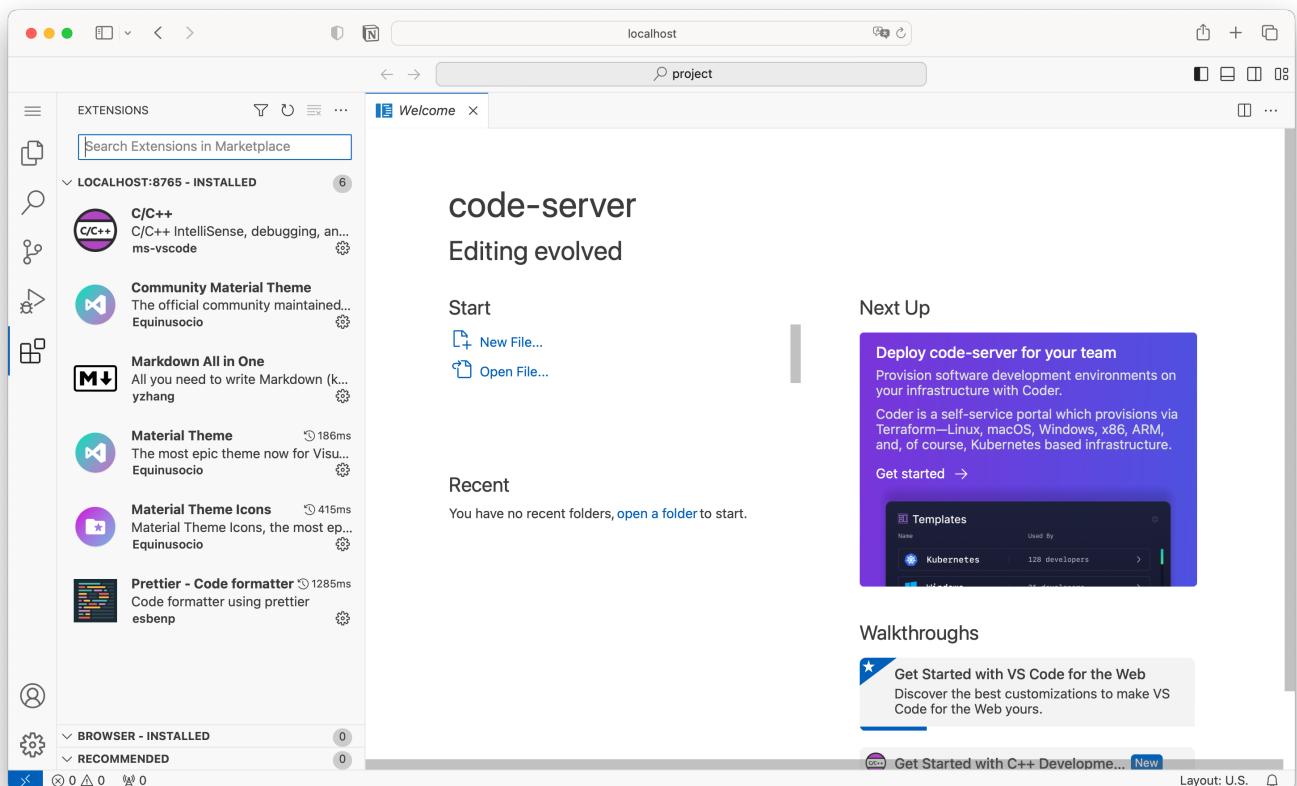
然后我们在这个密码中输入：dase 就可以进入了，这里要稍微等一下，加载不会特别快，等加载好后我们就可以看到一个完整的在linux系统上的可以直接使用的vscode了



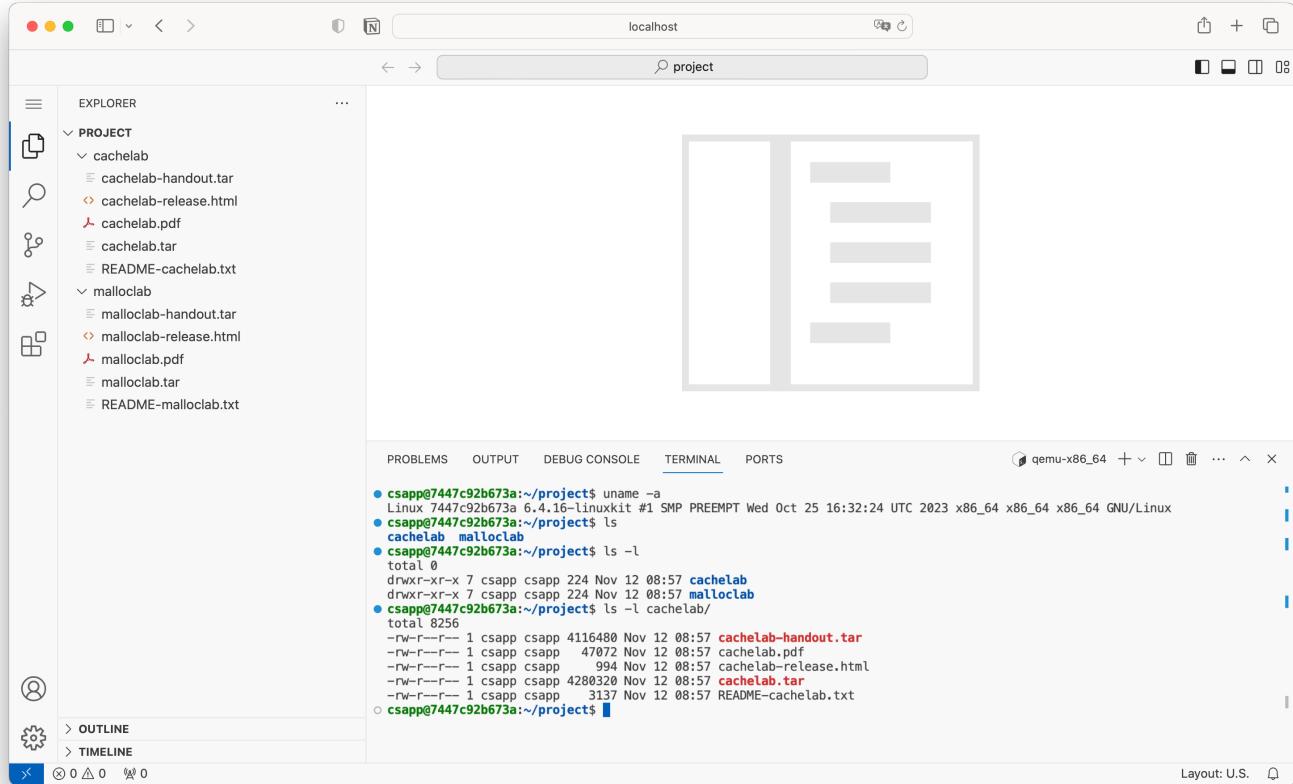
我们点击左侧的Folder（文件）选项后可以看到几个可选的文件夹，我们选择project，在其中我们就可以看到已经为各位准备好的cachelab和malloclab的源码了。



在其中的Extensions中你可以看到我已经为你们准备好的插件了，剩下需要的插件你可以自己安装，和你自己的vscode没有任何区别。



同时，在这其中如果你想直接使用Terminal终端，使用`ctrl-j`（Mac上是`command-j`）可以直接唤出终端，输入命令，我们可以看到这是一个完全的linux环境，到这里差不多一切都结束了，剩下的路靠各位自己了🙏😊😊😊。



那么当我们写了一段时间以后，不想写了呢就可以把这个容器关掉，当我们想要使用的时候再打开，具体我们可以直接使用Docker Desktop中的启动和停止的按钮来控制，在下面中间那张图中你可以看到，我们关闭这个容器后就网页里的vscode就不能在链接了。之后我们开启和终止容器都适用这个Docker Desktop里的Actions按钮就行了，不需要再在命令行中输入`docker run`的命令了。

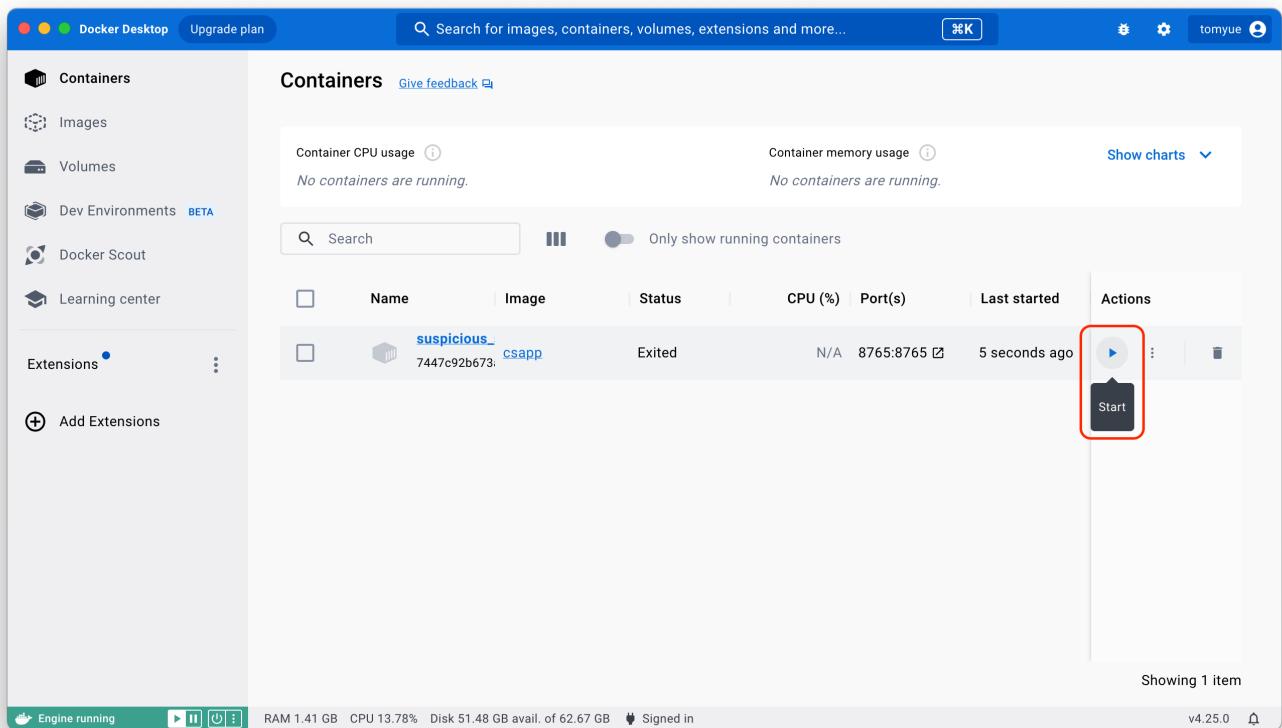
Docker Desktop interface showing a single running container named "suspicious_". The container details are as follows:

Name	Image	Status	CPU (%)	Port(s)	Last started
suspicious_	7447c92b673a: csapp	Running	1.61%	8765:8765	24 minutes ago

The "Actions" column for the "suspicious_" container includes a "Stop" button, which is highlighted with a red box.

VS Code interface showing a terminal window connected to a Docker container. The terminal output is as follows:

```
Linux 7447c92b673a 6.4.16-linuxkit #1 SMP PREEMPT Wed Oct 25 16:32:24 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
● csapp@7447c92b673a:~/project$ ls
cachelab malloclab
● csapp@7447c92b673a:~/project$ ls -l
total 0
drwxr-xr-x 7 csapp csapp 224 Nov 12 08:57 cachelab
drwxr-xr-x 7 csapp csapp 224 Nov 12 08:57 malloclab
● csapp@7447c92b673a:~/project$ ls -l cachelab/
total 8256
-rw-r--r-- 1 csapp csapp 4116480 Nov 12 08:57 cachelab-handout.tar
-rw-r--r-- 1 csapp csapp 47072 Nov 12 08:57 cachelab.pdf
-rw-r--r-- 1 csapp csapp 994 Nov 12 08:57 cachelab-release.html
-rw-r--r-- 1 csapp csapp 4280320 Nov 12 08:57 cachelab.tar
-rw-r--r-- 1 csapp csapp 3137 Nov 12 08:57 README-cachelab.txt
● csapp@7447c92b673a:~/project$
```



BTW, 如果在这个环境中对于debug, 以及launch.json配置不太熟悉的话, 和我说一下, 我们之后再写一些文档给到大家。辛苦大家阅读了!

TOMYUE Nov 12th