

# 0. 实验准备

- Docker安装
- VSCode安装与配置
- Linux基本操作

# 什么是docker? <https://www.bilibili.com/video/BV1jT4y1G7M3>

## 环境配置的难题

- 软件开发最大的麻烦事之一，就是环境配置。用户计算机的环境都不相同，你怎么知道自家的软件，能在那些机器跑起来？
- 用户必须保证两件事：操作系统的设置，各种库和组件的安装。只有它们都正确，软件才能运行。举例来说，安装一个 Python 应用，计算机必须有 Python 引擎，还必须有各种依赖，可能还要配置环境变量。
- 如果某些老旧的模块与当前环境不兼容，那就麻烦了。开发者常常会说："它在我的机器可以跑了" (It works on my machine)，言下之意就是，其他机器很可能跑不了。
- 环境配置如此麻烦，换一台机器，就要重来一次，旷日费时。很多人想到，能不能从根本上解决问题，软件可以带环境安装？也就是说，安装的时候，把原始环境一模一样地复制过来。



# 什么是docker? <https://www.bilibili.com/video/BV1jT4y1G7M3>

## 虚拟机方案

虚拟机 (virtual machine) 就是带环境安装的一种解决方案。它可以在一种操作系统里面运行另一种操作系统，比如在 Windows 系统里面运行 Linux 系统。应用程序对此毫无感知，因为虚拟机看上去跟真实系统一模一样，而对于底层系统来说，虚拟机就是一个普通文件，不需要了就删掉，对其他部分毫无影响。

虽然用户可以通过虚拟机还原软件的原始环境。但是，这个方案有几个缺点。

### (1) 资源占用多

虚拟机会独占一部分内存和硬盘空间。它运行的时候，其他程序就不能使用这些资源了。哪怕虚拟机里面的应用程序，真正使用的内存只有 1MB，虚拟机依然需要几百 MB 的内存才能运行。

### (2) 冗余步骤多

虚拟机是完整的操作系统，一些系统级别的操作步骤，往往无法跳过，比如用户登录。

### (3) 启动慢

启动操作系统需要多久，启动虚拟机就需要多久。可能要等几分钟，应用程序才能真正运行。

# 什么是docker? <https://www.bilibili.com/video/BV1jT4y1G7M3>

## Linux 容器方案

由于虚拟机存在这些缺点，Linux 发展出了另一种虚拟化技术：Linux 容器（Linux Containers，缩写为 LXC）。

**Linux 容器不是模拟一个完整的操作系统，而是对进程进行隔离。**或者说，在正常进程的外面套了一个保护层。对于容器里面的进程来说，它接触到的各种资源都是虚拟的，从而实现与底层系统的隔离。

由于容器是进程级别的，相比虚拟机有很多优势。

### （1）启动快

容器里面的应用，直接就是底层系统的一个进程，而不是虚拟机内部的进程。所以，启动容器相当于启动本机的一个进程，而不是启动一个操作系统，速度就快很多。

### （2）资源占用少

容器只占用需要的资源，不占用那些没有用到的资源；虚拟机由于是完整的操作系统，不可避免要占用所有资源。另外，多个容器可以共享资源，虚拟机都是独享资源。

### （3）体积小

容器只要包含用到的组件即可，而虚拟机是整个操作系统的打包，所以容器文件比虚拟机文件要小很多。

总之，容器有点像轻量级的虚拟机，能够提供虚拟化的环境，但是成本开销小得多。

# 什么是docker? <https://www.bilibili.com/video/BV1jT4y1G7M3>

Docker是当前最流行的linux容器技术之一。

Docker 属于 Linux 容器的一种封装，提供简单易用的容器使用接口。它是目前最流行的 Linux 容器解决方案。

Docker 将应用程序与该程序的依赖，打包在一个文件里面。运行这个镜像文件（image），就会生成一个虚拟容器(container)。程序在这个虚拟容器里运行，就好像在真实的物理机上运行一样。有了 Docker，就不用担心环境问题。

总体来说，Docker 的接口相当简单，用户可以方便地创建和使用容器，把自己的应用放入容器。容器还可以进行版本管理、复制、分享、修改，就像管理普通的代码一样。

# 什么是docker? <https://www.bilibili.com/video/BV1jT4y1G7M3>

## Docker的用途

Docker 的主要用途，目前有三大类。

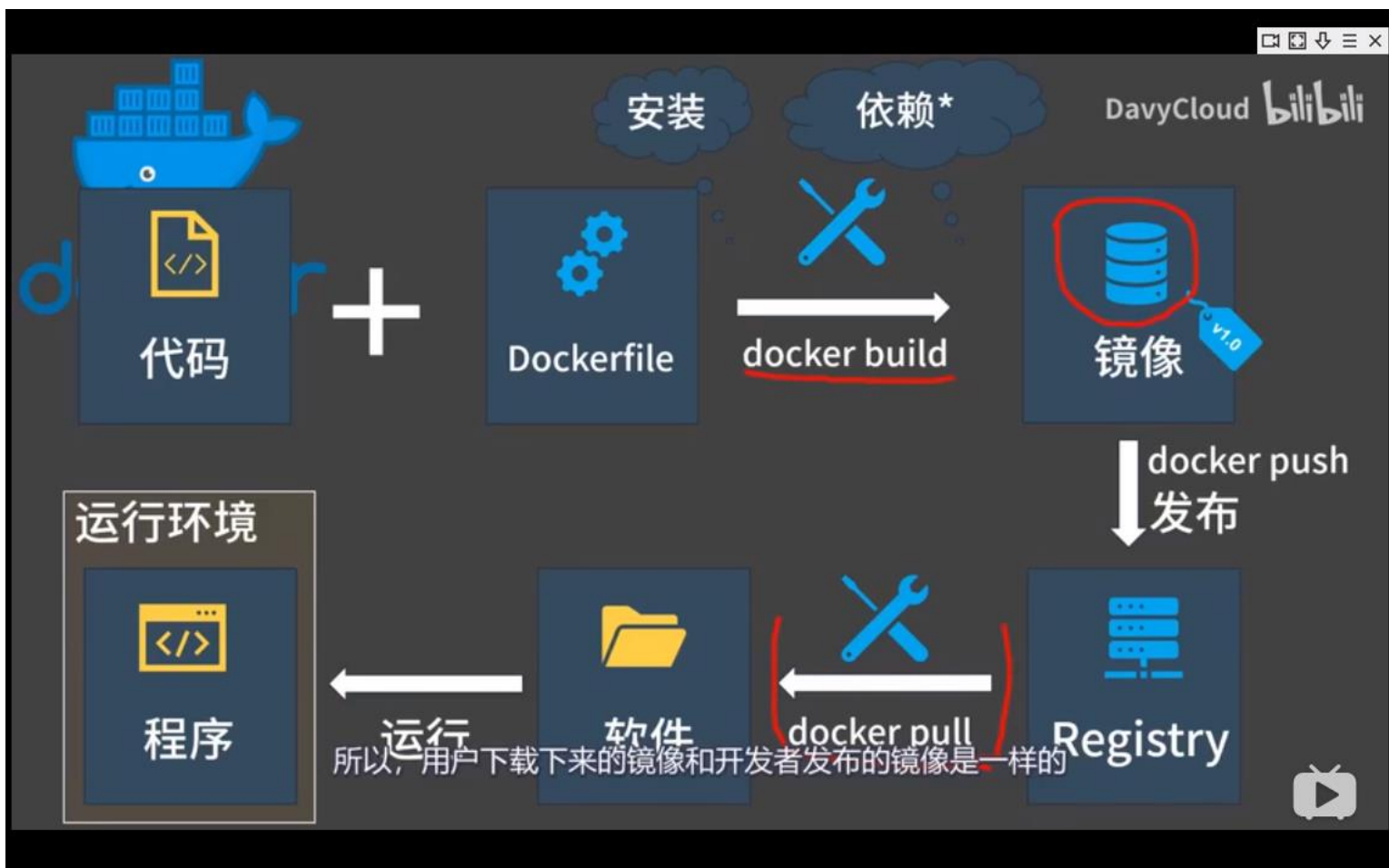
- (1) 提供一次性的环境。比如，本地测试他人的软件、持续集成的时候提供单元测试和构建的环境。
- (2) 提供弹性的云服务。因为 Docker 容器可以随开随关，很适合动态扩容和缩容。
- (3) 组建微服务架构。通过多个容器，一台机器可以跑多个服务，因此在本机就可以模拟出微服务架构。

本课程使用docker可免去大家配置环境的过程，只要下载我们提供的同一环境镜像（image）即可直接进行代码开发，统一的运行环境也便于作业标准化测试与批改。

# 什么是docker?

感兴趣的同学可参考以下视频，更好地理解docker：

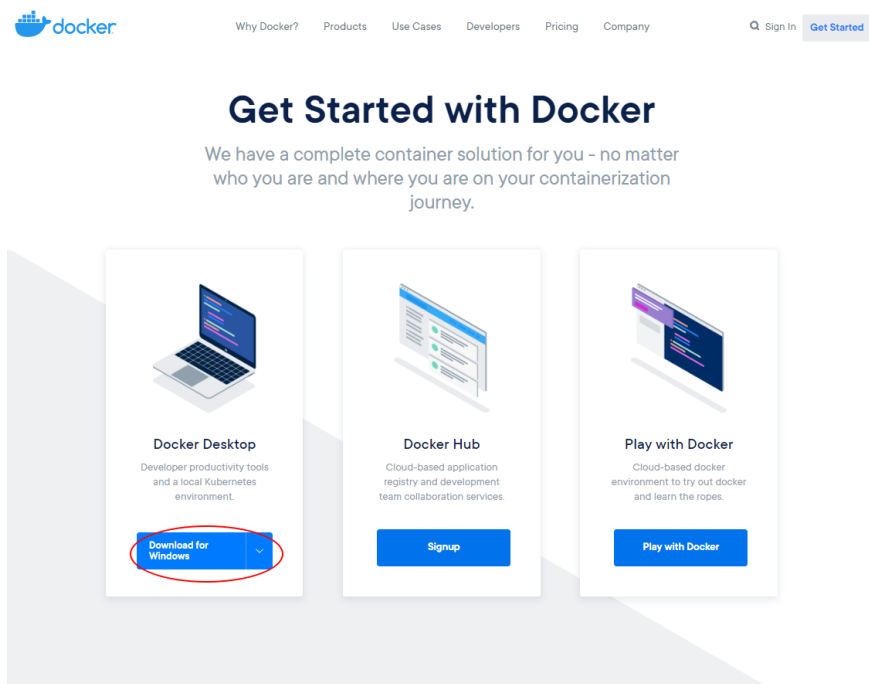
<https://www.bilibili.com/video/BV1jT4y1G7M3>



# 安装docker (Windows 10 Pro, Enterprise, and Education)

详细教程: <https://docs.docker.com/docker-for-windows/install/>

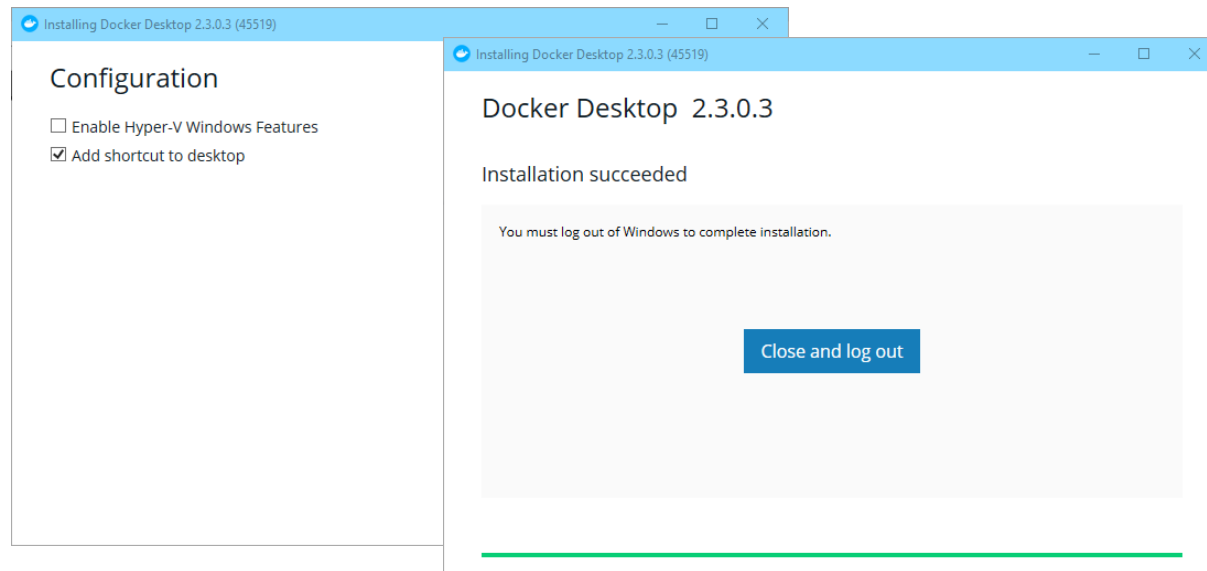
1. 访问 <https://www.docker.com/get-started> , 选择download for windows, 运行安装包。



2. 按照默认选项, 点击下一步安装即可。  
Configuration中如果出现Enable Hyper-V的选项, 可以不勾选 (不影响后续使用) 。

(如勾选, 需要按照<https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/enable-hyper-v> 启用系统的hyper-v功能)

3. 安装完成后按照提示注销或重启。

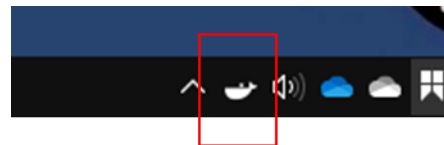




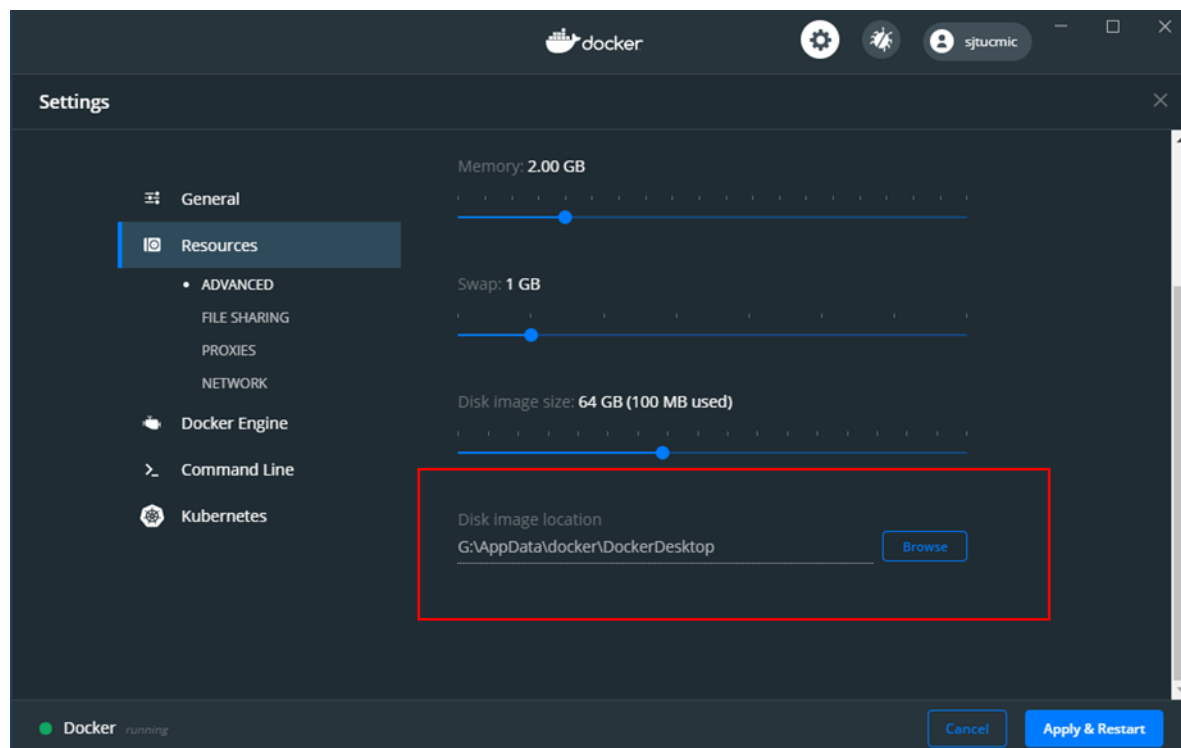
# 安装docker (Windows 10 Pro, Enterprise, and Education)

4. 重新登录windows后，稍等片刻，直到任务栏出现鲸鱼图标。

(或者在开始菜单里手动启动)



5. 等待docker启动完成之后（鲸鱼图标不再播放动画），右击鲸鱼图标可打开设置，限制内存、CPU资源使用等（C盘空间不够的同学记得在Resources下重新设置disk image location）。



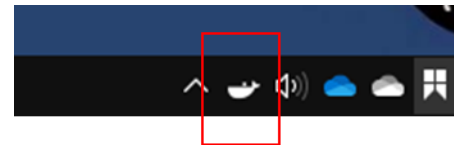
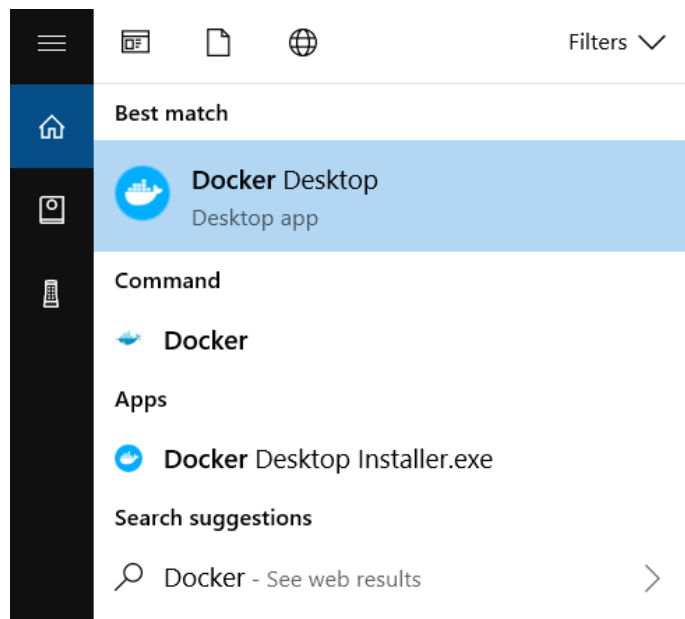
# 安装docker (Windows Home version 2004 or higher)

详细教程: <https://docs.docker.com/docker-for-windows/install-windows-home/>

- 开启WSL2: 以管理员模式打开PowerShell, 输入以下3行:

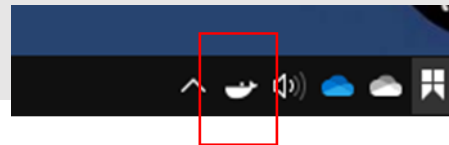
```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart  
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart  
wsl --set-default-version 2
```

- 下载: <https://hub.docker.com/editions/community/docker-ce-desktop-windows/>
- 安装时选中 “Enable WSL 2 Features”
- 安装完成后可以在任务栏看到一个鲸鱼的图标

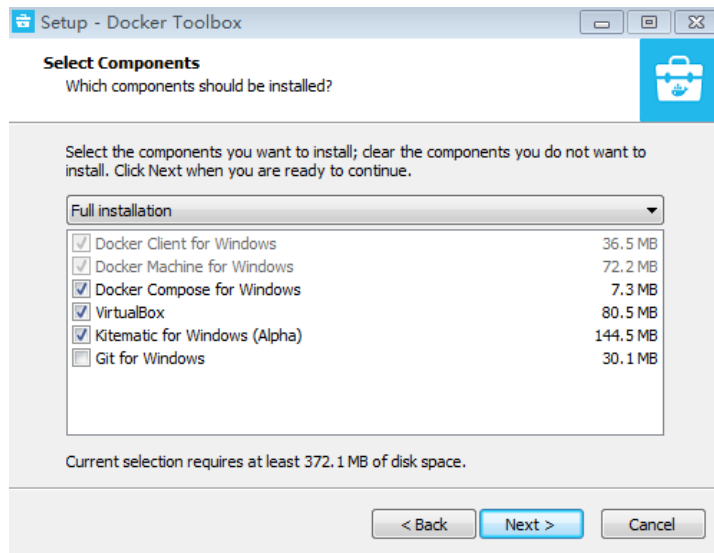


# 安装docker (Windows 旧版本系统)

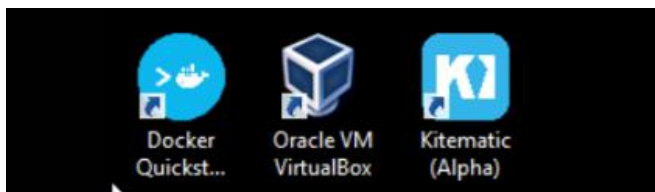
详细教程: <https://www.runoob.com/docker/windows-docker-install.html>



- 下载: <http://mirrors.aliyun.com/docker-toolbox/windows/docker-toolbox/DockerToolbox-18.03.0-ce.exe>
- 双击安装, 勾选以下组件:



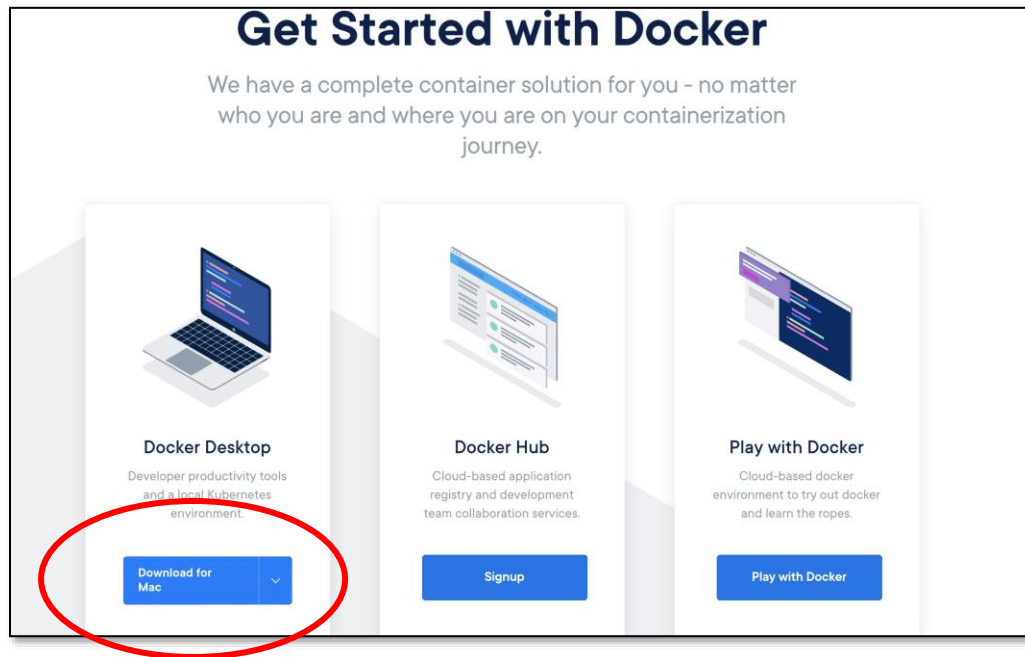
- 下载完成之后直接点击安装, 安装成功后, 桌边会出现三个图标, 如下图所示:



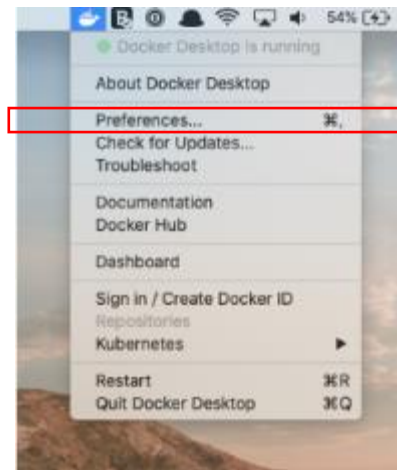
- 点击 Docker QuickStart 图标来启动 Docker Toolbox 终端。
- 如果系统显示 User Account Control 窗口来运行 VirtualBox 修改你的电脑, 选择 Yes。

# 安装docker (MacOS)

1. 访问 <https://www.docker.com/get-started> , 下载Docker Desktop for Mac (<https://download.docker.com/mac/stable/Docker.dmg>) 并双击安装。



2. 安装完成后可在顶部看到一个鲸鱼图标。可点击preferences/settings进行设置。



# 安装docker (Ubuntu/Debian/CentOS)

- 请参考如下教程自行安装:

<https://www.runoob.com/docker/ubuntu-docker-install.html>

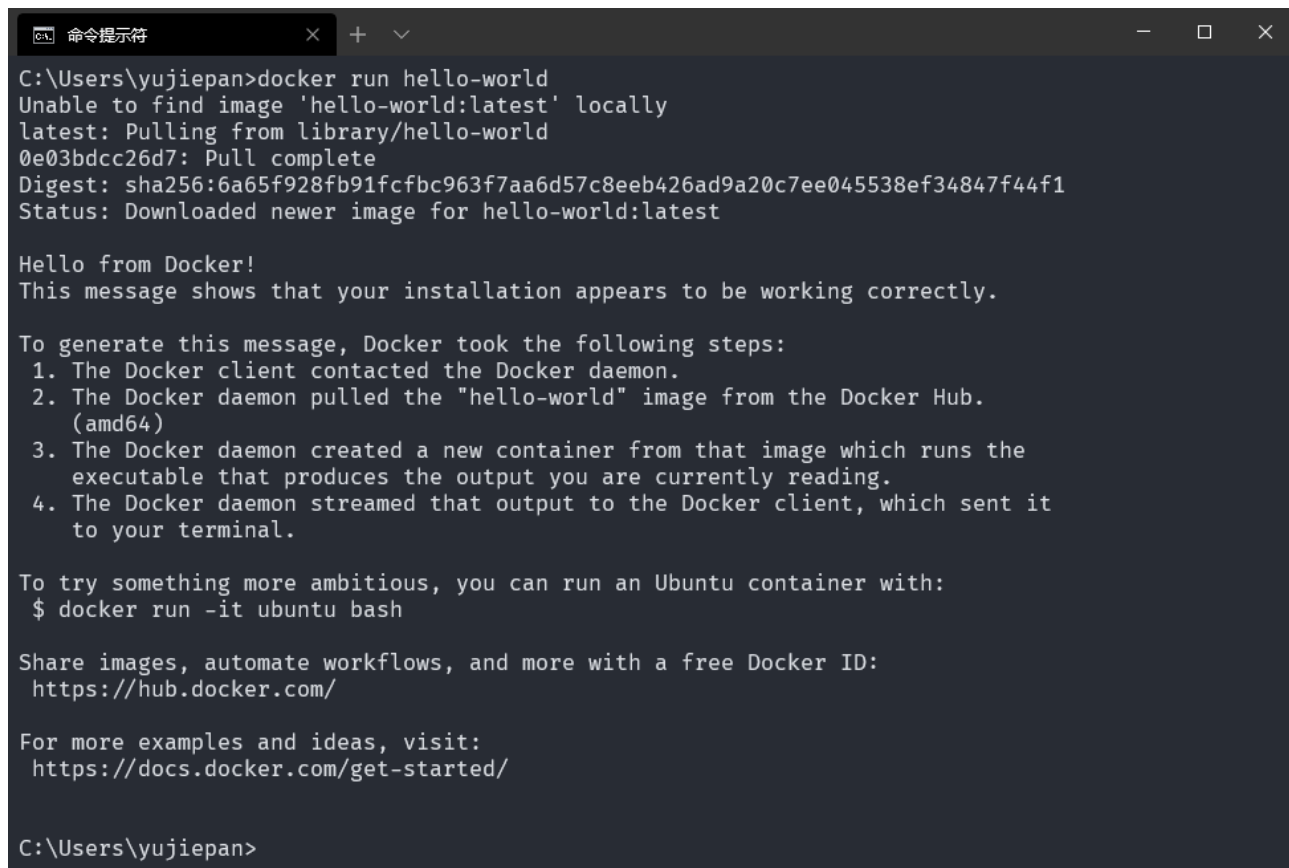
<https://www.runoob.com/docker/debian-docker-install.html>

<https://www.runoob.com/docker/centos-docker-install.html>

<https://docs.docker.com/engine/install/>

# 测试docker安装是否成功

- 打开windows/macOS/ubuntu等的终端（旧版本windows 打开Docker quickstart），输入 `docker run hello-world`
- 出现类似下图的内容则说明安装正确。



```
C:\Users\yujiepan>docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:6a65f928fb91fcfbc963f7aa6d57c8eeb426ad9a20c7ee045538ef34847f44f1
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

C:\Users\yujiepan>
```

# 下载本课程所需的运行环境镜像

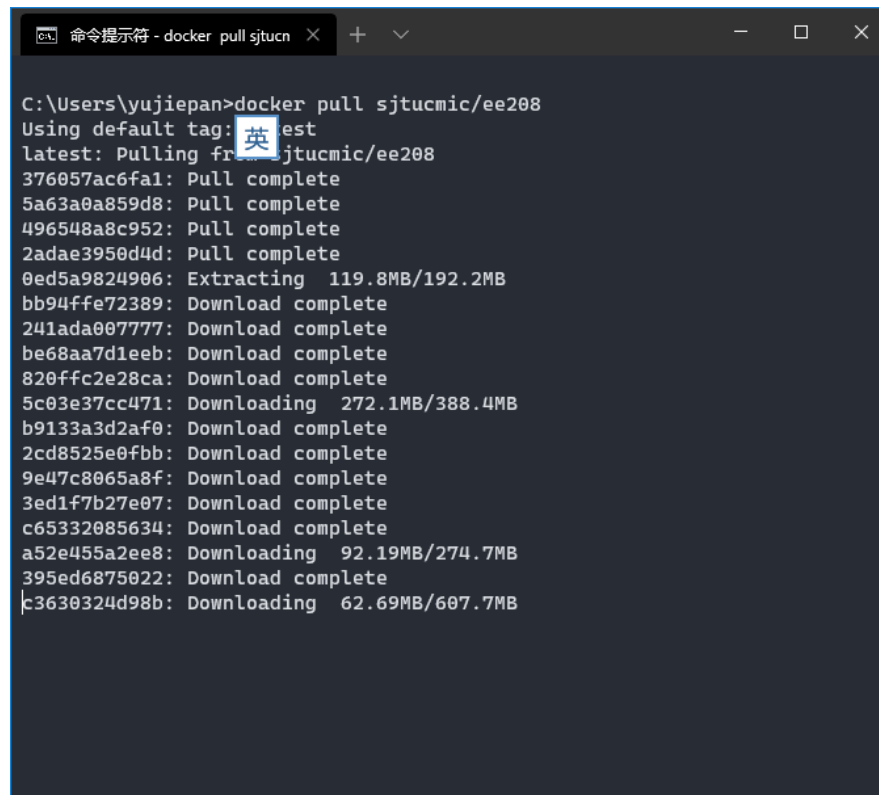
**注意：本课程所有的实验（包括最后的大作业）都需要在指定环境下运行，便于统一测试。**

1. 启动docker，保证鲸鱼图标在任务栏（否则会出现error during connect的错误）

2. 在命令行里输入docker pull sjtucmic/ee208，等待下载完成。

备注：以上指令默认下载latest最新版，已满足课程要求。

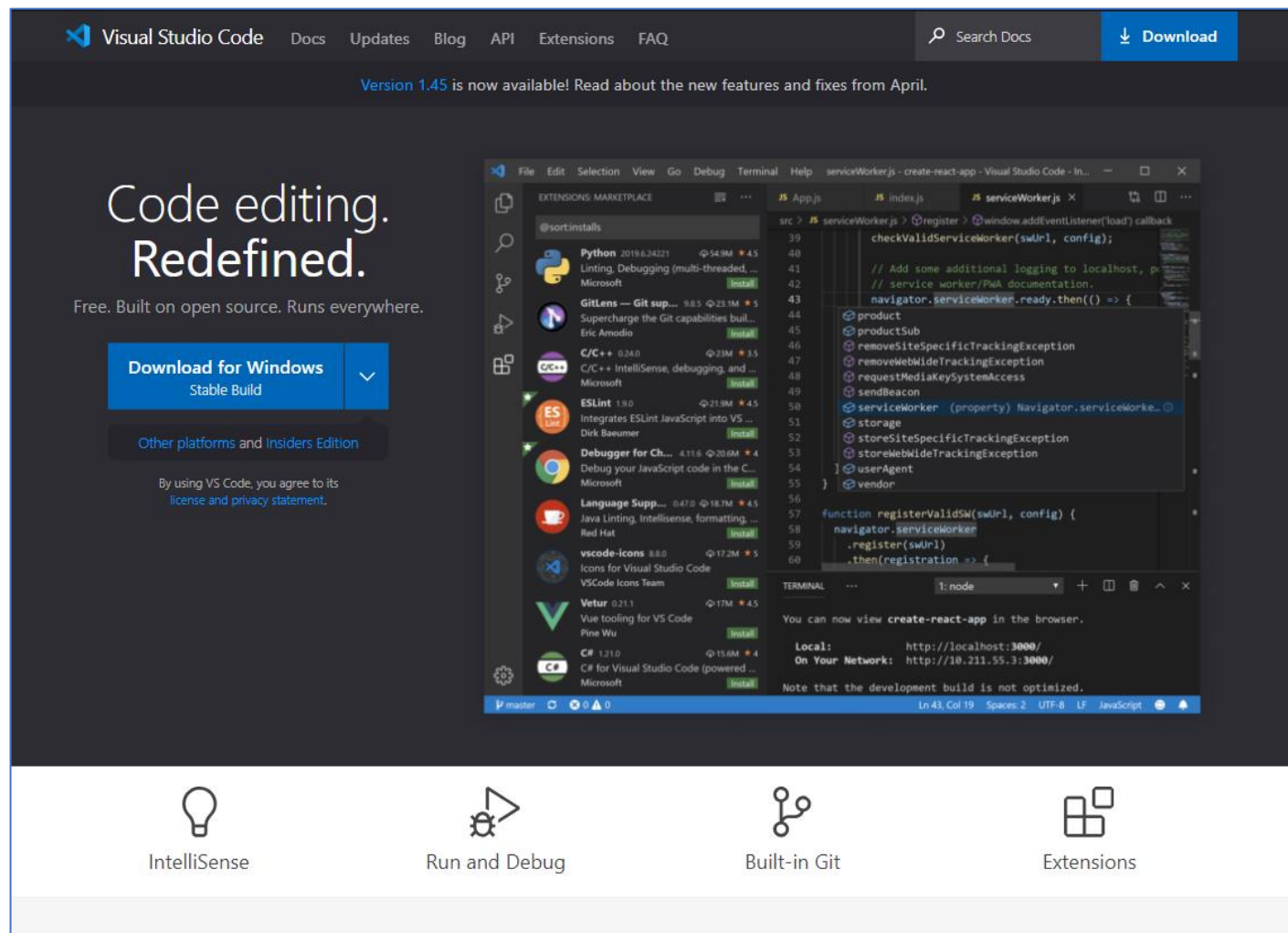
其他版本可在<https://hub.docker.com/r/sjtucmic/ee208/tags> 上查看。



```
C:\Users\yujiepan>docker pull sjtucmic/ee208
Using default tag: latest
latest: Pulling from sjtucmic/ee208
376057ac6fa1: Pull complete
5a63a0a859d8: Pull complete
496548a8c952: Pull complete
2adae3950d4d: Pull complete
0ed5a9824906: Extracting 119.8MB/192.2MB
bb94ffe72389: Download complete
241ada007777: Download complete
be68aa7d1eeb: Download complete
820ffc2e28ca: Download complete
5c03e37cc471: Downloading 272.1MB/388.4MB
b9133a3d2af0: Download complete
2cd8525e0fbb: Download complete
9e47c8065a8f: Download complete
3ed1f7b27e07: Download complete
c65332085634: Download complete
a52e455a2ee8: Downloading 92.19MB/274.7MB
395ed6875022: Download complete
c3630324d98b: Downloading 62.69MB/607.7MB
```

# 安装VSCode

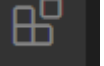
- 访问 <https://code.visualstudio.com/>, 下载并安装软件。






# 配置VSCode访问docker container

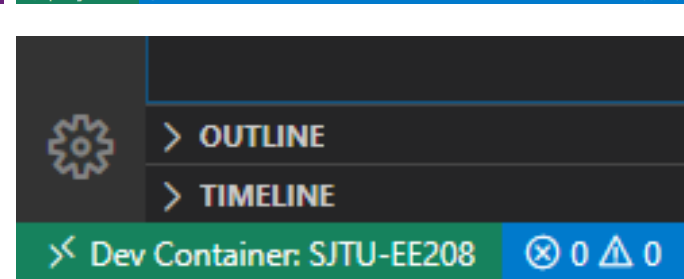
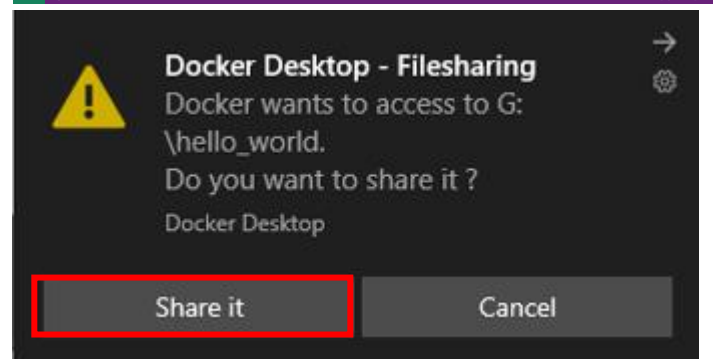
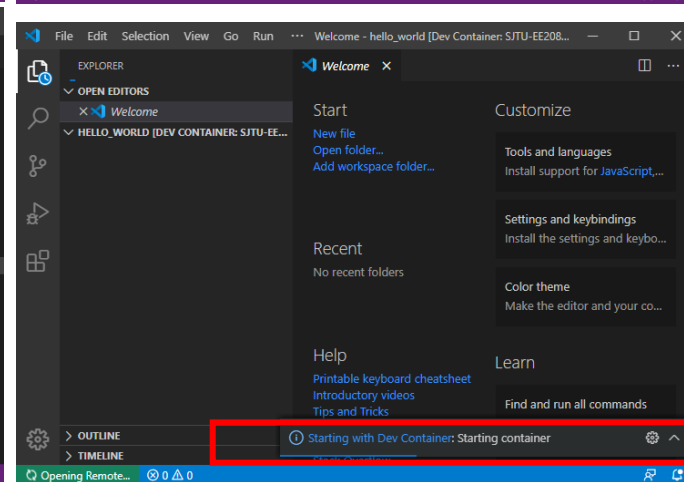
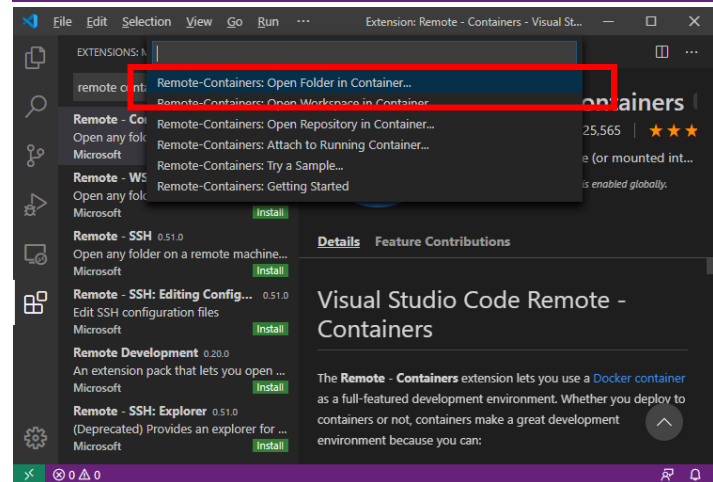
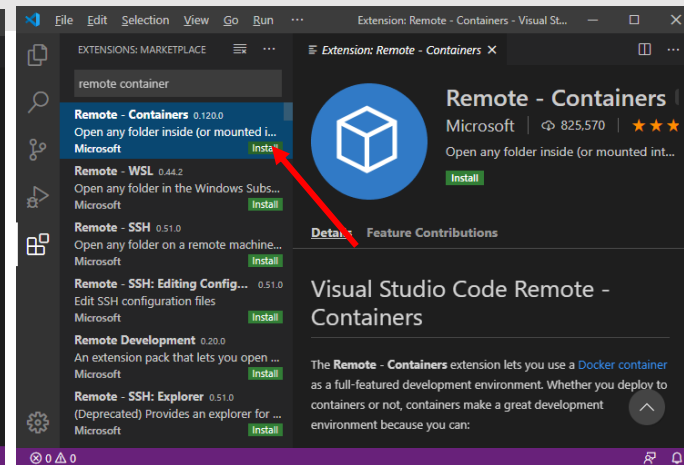
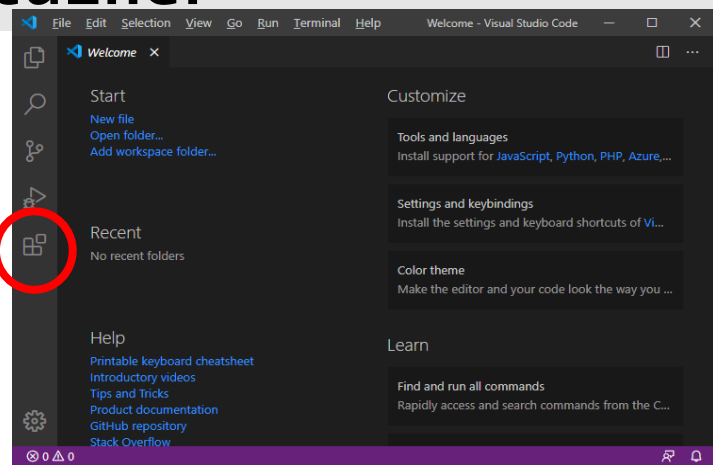
1. 启动docker

2. 打开VSCode, 点击  图标, 搜索 remote containers, 安装该插件。

3. 安装插件之后会在左下角看到一个  图标, 点击后选择 “Open Folder in Container”, 选择实验课件提供的 hello\_world文件夹。

4. 当出现file sharing的询问时, 选择 “共享”(share it)。

5. 当左下角出现Dev Container:xxx 则表示配置成功。



# 测试VSCode + Docker

- 使用“**Ctrl+`**”调出terminal，在默认路径下输入`python hello_world.py`显示如下。在文件夹下出现一个test文件夹和output.txt文件说明成功。
- 注意：docker容器中的文件系统和你的主机磁盘是隔离的，

**除了已与主机共享的文件夹，其他所有在docker系统下的改动都可能在容器关闭后丢失！**

**除了已与主机共享的文件夹，其他所有在docker系统下的改动都可能在容器关闭后丢失！**

**除了已与主机共享的文件夹，其他所有在docker系统下的改动都可能在容器关闭后丢失！**

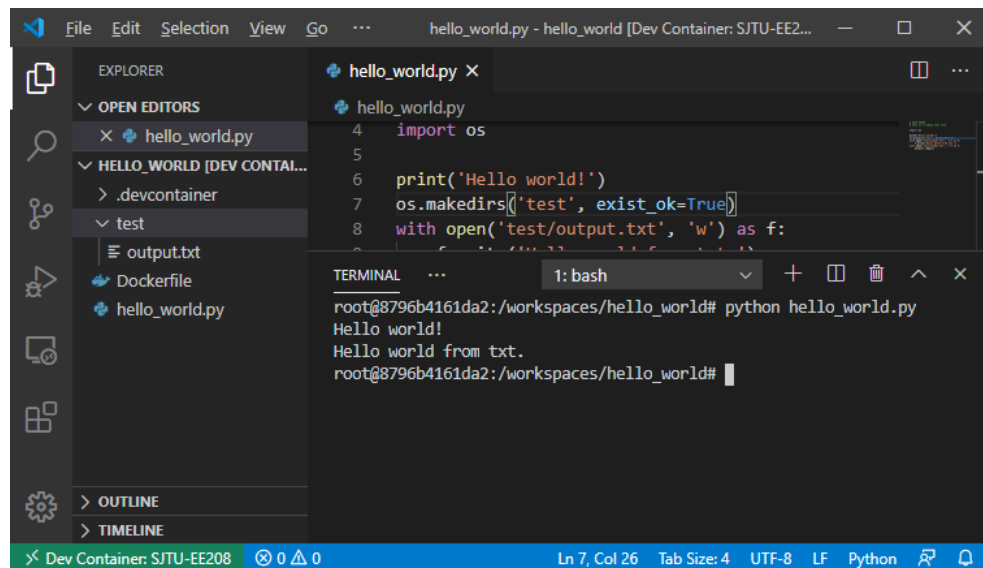
你可以把docker理解成网吧里的电脑，重启后所有改动都会还原。好在我们之前设置了file share文件夹共享（即vscode打开的文件夹），它就好像一个U盘，**存储在该文件夹下的数据不会因为容器关闭而丢失。**

因此，**请保证所有生成的数据都保存在vscode打开的文件夹下。**

在代码中涉及到路径操作时（比如`os.makedirs()`），

**请使用相对路径，不要使用绝对路径！**

可参考：<http://c.biancheng.net/view/5693.html>



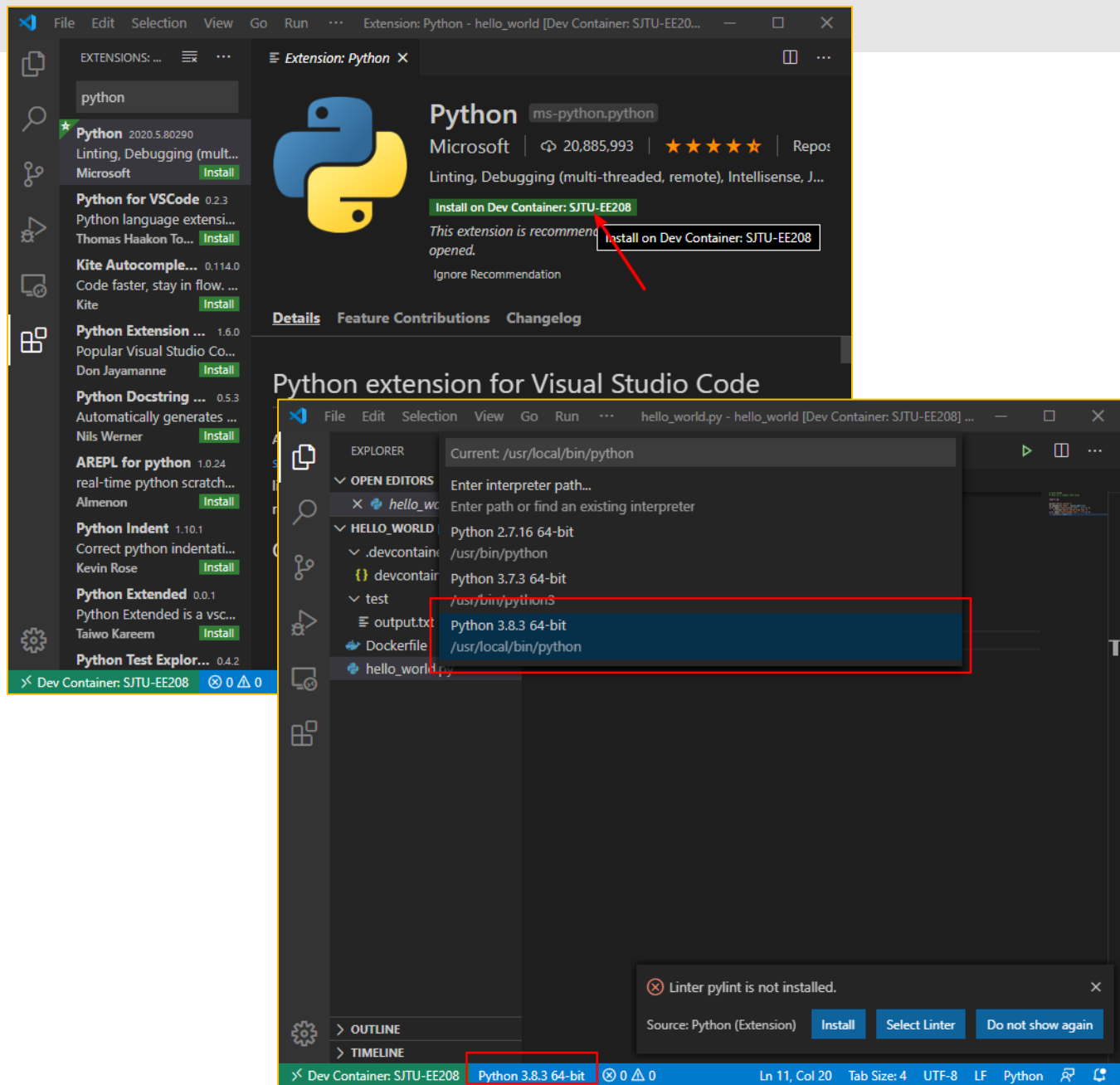
The screenshot shows the VS Code interface with a Docker container named 'SJTU-EE208'. The Explorer sidebar on the left shows the file structure: 'hello\_world.py', '.devcontainer', 'test' (containing 'output.txt'), 'Dockerfile', and 'hello\_world.py'. The main editor displays the code in 'hello\_world.py':

```
1: #!/usr/bin/env python
2: # -*- coding: utf-8 -*-
3:
4: import os
5:
6: print('Hello world!')
7: os.makedirs('test', exist_ok=True)
8: with open('test/output.txt', 'w') as f:
```

The TERMINAL panel at the bottom shows the command prompt '1: bash' and the execution of 'python hello\_world.py', resulting in the output: 'Hello world!' and 'Hello world from txt.'.

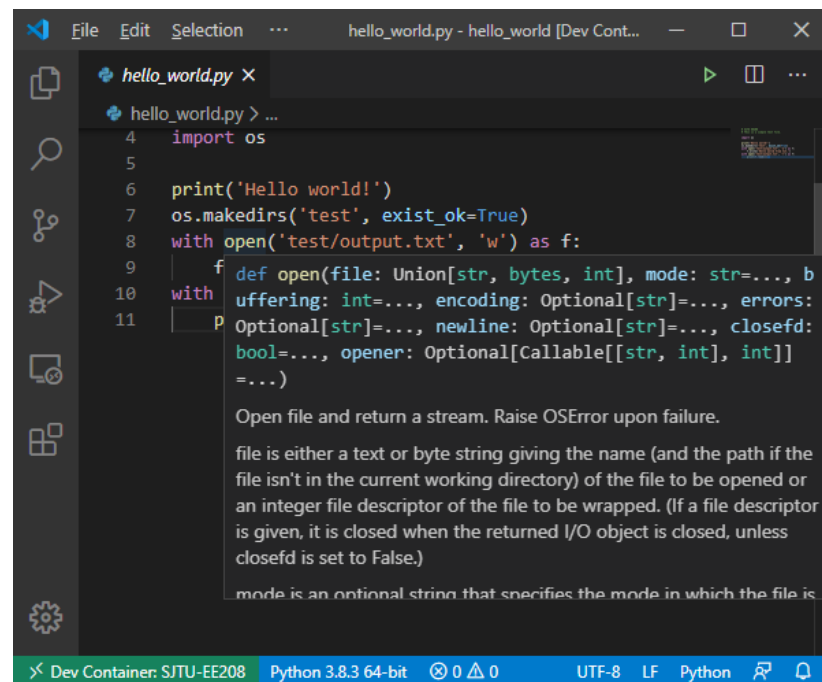
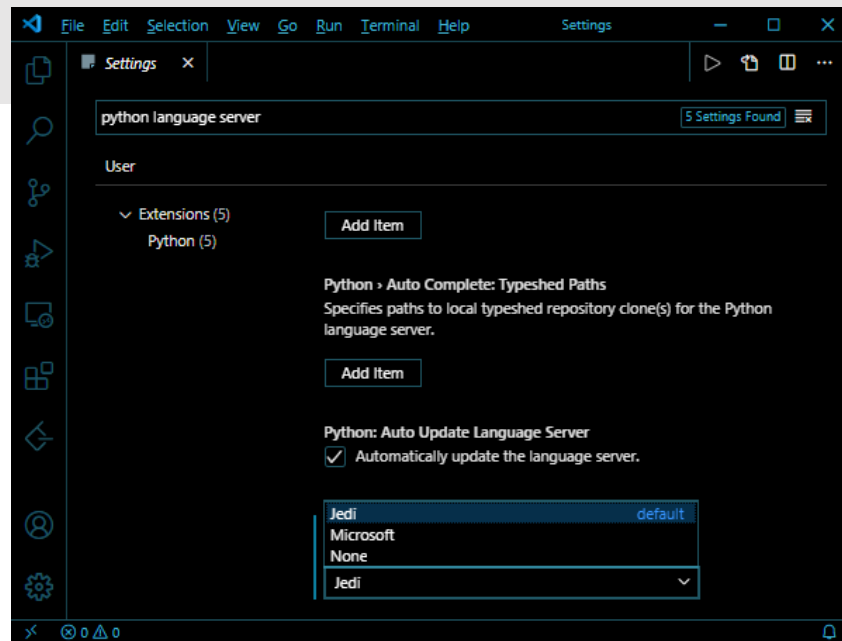
# 代码自动提示等插件安装

- 代码提示等插件需要手动安装。点击  图标，搜索python，选择“Install on Dev Container: xxx”，按照提示点击reload。
- 类似方法可自行搜索其他插件进行安装（可选）  
（插件推荐：如 <https://www.jianshu.com/p/5ba8586c7819> ）
- 重新打开hello\_world.py之后，在左下角Dev Container右边点击Select python interpreter选择python版本，请统一选择 /usr/local/bin/python  
（具体版本号可能与图例中不同）
- 若提示pylint（语法检查器），formatter（代码格式整理）未安装时，点击安装即可。



# 代码自动提示等插件安装

- 此时，鼠标悬停到代码某函数上，应该会出现代码提示。
- 由于国内网络问题，若无法出现代码提示，可以点击左下角齿轮，选择settings，分别把User和Remote两个标签页的python language server选项都改为“Jedi”。按照提示reload窗口。

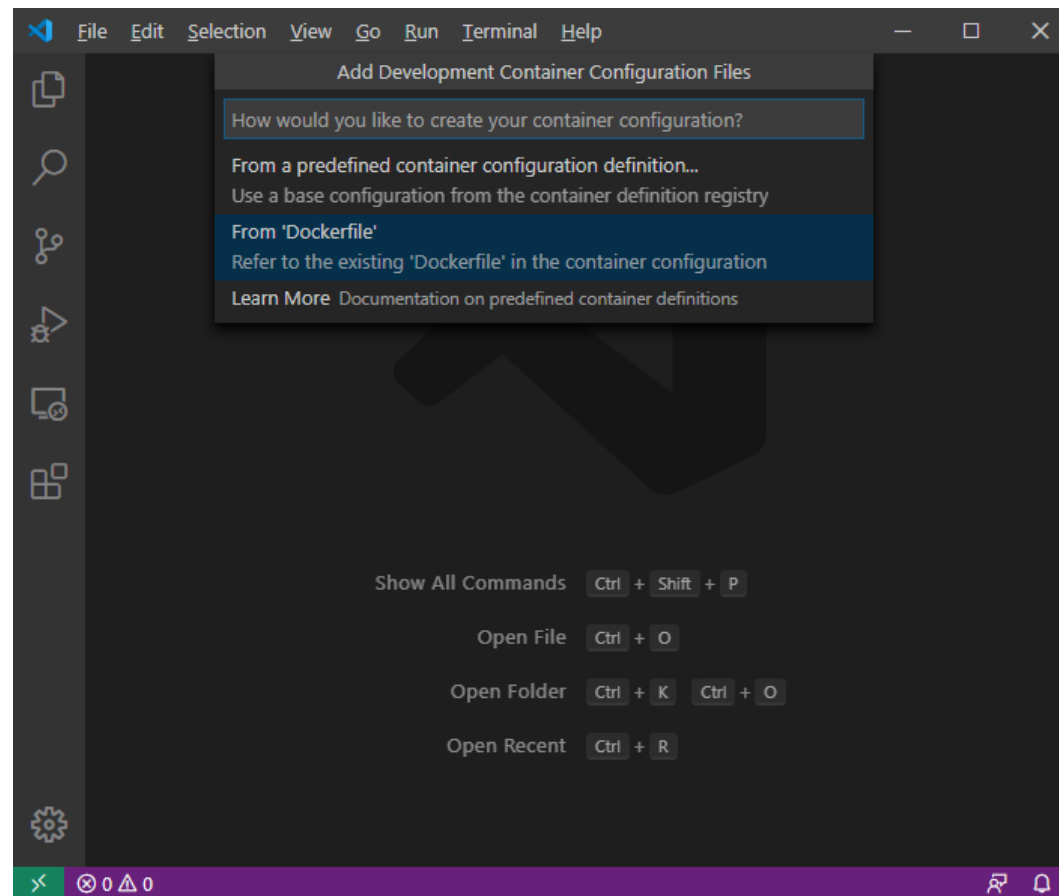


# 如何在自己的代码文件夹中使用docker？（重要）

- 在做后续作业时，请把hello\_world中提供的Dockerfile文件复制到你新建的**代码文件夹根目录**，然后使用前面的方法，点击左下角，选择“Open Folder in Container”，选择“From Dockerfile”。
- 在提交作业时，也请将Dockerfile文件附在根目录。（可能需要重新安装插件）

- 注意：

1. 如无特别需求，请不要修改Dockerfile的文件名和内容
2. 写作业时不要删除vscode自动生成的.devcontainer和.vscode文件夹。但最终提交作业时，无需提交这两项，只需要附上Dockerfile。



# VSCode快捷键

- 调出终端: `ctrl+``
- 查找文件: `ctrl+p`
- 代码格式整理: `shift+alt+f`
- 单行注释: `[ctrl+k,ctrl+c]` 或 `ctrl+/*`

其他功能请大家自行探索~

# Linux基本操作

- `pwd`: 查看当前路径
- `cd xxx`: 打开xxx文件夹
- `cd ..` : 返回上一级文件夹
- `ls`: 显示当前文件夹下的所有文件 (`ls -a`: 显示所有文件, 包括隐藏文件)
- `mkdir`: 新建文件夹
- `touch xx.txt`: 新建xx.txt文件
- `ps -ux`: 显示正在运行的进程
- `mv source target`: 移动文件、文件夹(也可以用作文件改名)
- `cp source target`: 复制
- `rm xxx`: 删除 (`rm -r xxx`: 删除文件夹)
  
- `cd ~`: 打开用户主目录。“~”可以看作一个缩写。实际路径可以通过`pwd`查看。
- 注意 `cd /lib` 和 `cd lib` 的区别: 在linux中, /是最顶层的目录(而windows中是C:/,D:/等)。 `cd /lib` 是打开最顶层目录下的lib文件夹, 而`cd lib`是打开当前目录(通过`pwd`查看)下的lib文件夹。
- 当文件或者文件夹名字很长时, 可以按下tab键自动补全。

# 参考资料

- 如何通俗解释Docker是什么? ★ <https://www.bilibili.com/video/BV1jT4y1G7M3>
- <https://www.zhihu.com/question/28300645>, <https://zhuanlan.zhihu.com/p/38533234>
- Docker 安装视频: <https://www.bilibili.com/video/BV137411F7ny>
- ★ Docker 入门教程: <https://www.ruanyifeng.com/blog/2018/02/docker-tutorial.html>
- Docker 教程: <https://www.runoob.com/docker/docker-tutorial.html>
- Dockerfile 的写法: <https://www.runoob.com/docker/docker-dockerfile.html>
- VSCode 必装的 10 个高效开发插件: <https://zhuanlan.zhihu.com/p/46781330>
- VSCode remote-containers的原理: <https://code.visualstudio.com/docs/remote/containers>
- ★ Linux基本操作看这篇就够了: <https://zhuanlan.zhihu.com/p/36801617>
- Linux命令大全: <https://www.runoob.com/linux/linux-command-manual.html>
  
- 延伸阅读:
- Docker是虚拟机吗? <https://www.zhihu.com/question/48174633>