

自动驾驶中的SLAM技术Docker环境部署

本文档提供**自动驾驶中的SLAM技术**的标准开发环境, 支持arm64和amd64架构的cpu, 旨在减少同学们环境配置的工作量。

自动驾驶中的SLAM技术代码地址

```
1 | https://github.com/gaoxiang12/slam\_in\_autonomous\_driving
```

配套课程

```
1 | https://www.shenlanxueyuan.com/my/course/615
```

Dockerfile开源在

```
1 | https://github.com/zijiechenrobotics/slam\_in\_autonomous\_driving\_docker
```

1 安装Docker

1.1 Linux

Linux系统只需安装docker engine, 参考官网: <https://docs.docker.com/engine/install/ubuntu/>

具体步骤为

```
1 | sudo apt-get update
2 |
3 | sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-
  | plugin
4 |
5 | sudo docker run hello-world
```

为了能在非**sudo**模式下使用Docker, 需要将当前用户加入Docker Group

```
1 | # 加入group
2 | sudo usermod -aG docker $USER
3 |
4 | # 重启一下电脑, 使上述变更生效
5 | reboot
```

运行hello-world出现以下画面表示docker安装成功

```
czj2020@haique-Tower:~/workspace/learn_docker/sad_workspace$ docker run hello-world

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

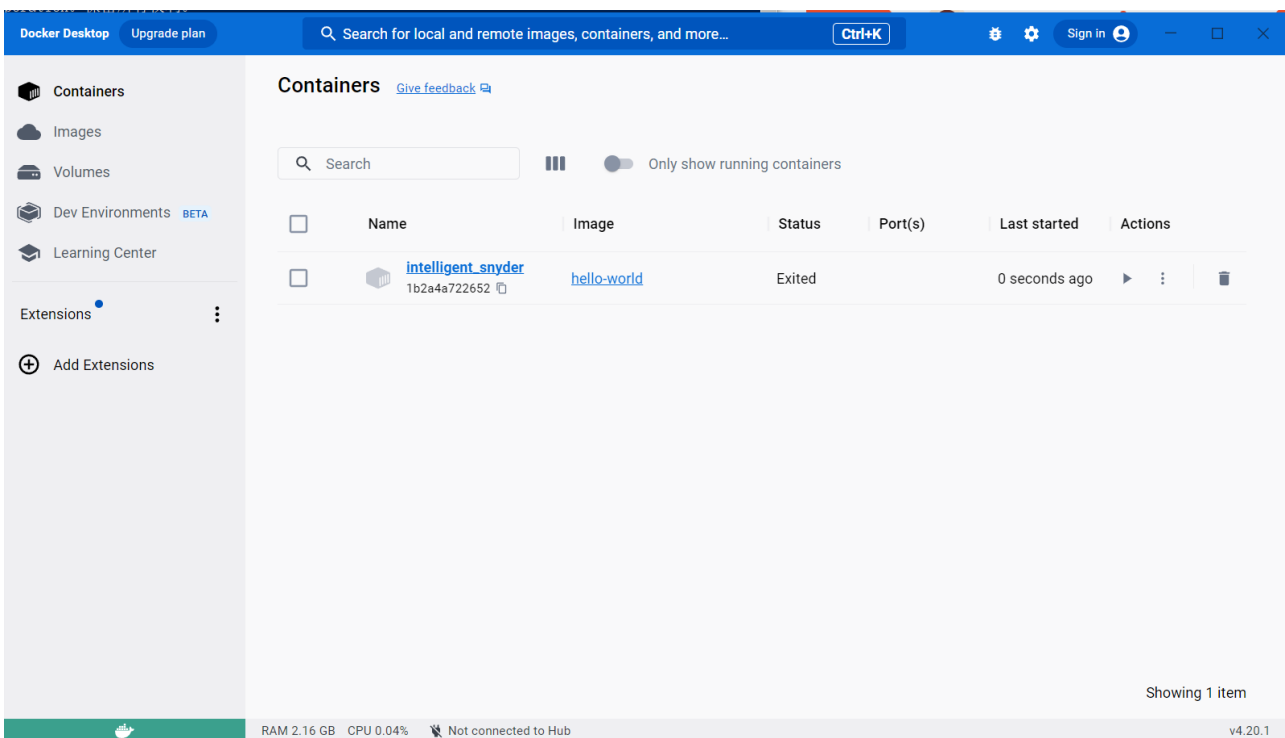
Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

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

1.2 Windows

进入<https://www.docker.com/>下载Docker Desktop for Windows并安装

启动Docker Desktop，您将看到如下画面（刚下载好是没有containers的）



启动PowerShell，输入docker run hello-world，如果看到以下画面，则证明安装成功

```

选择 Windows PowerShell
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

尝试新的跨平台 PowerShell https://aka.ms/pscore6

PS C:\Users\kris> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:a13ec89cdf897b3e551bd9f89d499db6ff3a7f44c5b9eb8bca40da20eb4ealfa
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/

PS C:\Users\kris>

```

1.3 MacOS

步骤与windows下类似, 安装docker desktop, 然后测试hello-world判断是否安装成功

2 拉取镜像


终端或powershell输入, 约4.88GB

```

1 # 国内用户
2 docker pull registry.cn-guangzhou.aliyuncs.com/zijiechenrobotics/sad-workspace:latest
3
4 # 国外用户
5 docker pull zijiechenrobotic/sad-workspace:latest

```

image使用buildx进行跨平台编译, 可以支持amd64和arm64架构的cpu, 即可以在windows, ubuntu以及macos中运行。

TAG				
latest				
Last pushed 3 hours ago by zijiechenrobotic				
DIGEST	OS/ARCH	SCANNED	LAST PULL	COMPRESSED SIZE 
4b9418226811	linux/amd64	---	3 hours ago	1.51 GB
66a72d5d025c	linux/arm64	---	3 hours ago	1.42 GB



若阿里云的image不能在arm64中运行, 可以尝试拉取docker hub中的image

3 启动镜像

新建一个 `docker-compose.yml` 文件，内容如下(推荐在github上复制源码, 直接复制pdf的可能启动不成功, pdf编码的原因)

```
1 version: '3.4'
2 services:
3   sad-workspace:
4     image: registry.cn-guangzhou.aliyuncs.com/zijiechenrobotics/sad-workspace:latest
5     # or
6     # image: zijiechenrobotic/sad-workspace:latest
7     environment:
8       - VNC_PW=abc123 # vnc密码, 可以改成自己想要的密码
9       - VNC_GEOMETRY=1920x1080 # vnc分辨率
10      - VNC_DEPTH=24 # 颜色位数16/24/32可用, 越高画面越细腻, 但网络不好的也会更卡
11     volumes:
12       - ./workspace:/root/workspace # 存储空间映射, 映射到放代码的空间
13     ports:
14       # noVNC port:
15       - 46080:6080
16       # standard VNC port:
17       - 45901:5901
```

以及一个workspace文件夹，用于存放git clone代码。具体摆放如下

 workspace	2023/6/24 19:20	文件夹	
 docker-compose.yml	2023/6/24 19:22	YML 文件	1 KB

终端cd到yml文件的目录下，输入以下命令，启动docker

```
1 docker compose up -d
```

关闭docker

```
1 docker compose down
```

3.1 通过浏览器访问ubuntu桌面

任意浏览器，地址栏输入

```
1 | localhost:46080
```

点击其中的 `vnc.html`，再点击 `连接`，输入 `yml` 文件中的 `vnc` 密码即可进入桌面

同时，可以点击左边 `设置`，使用 `缩放模式：本地缩放` 得到更好的操作体验

3.2 通过VNC访问ubuntu桌面

需要下载任意支持vnc协议的远程桌面助手，如：vncviewer。

vnc的ip，请输入

```
1 | localhost:45901
```

4 编译课程代码

终端cd到workspace目录下，clone课程代码

```
1 | git clone https://github.com/gaoxiang12/slam_in_autonomous_driving.git
```

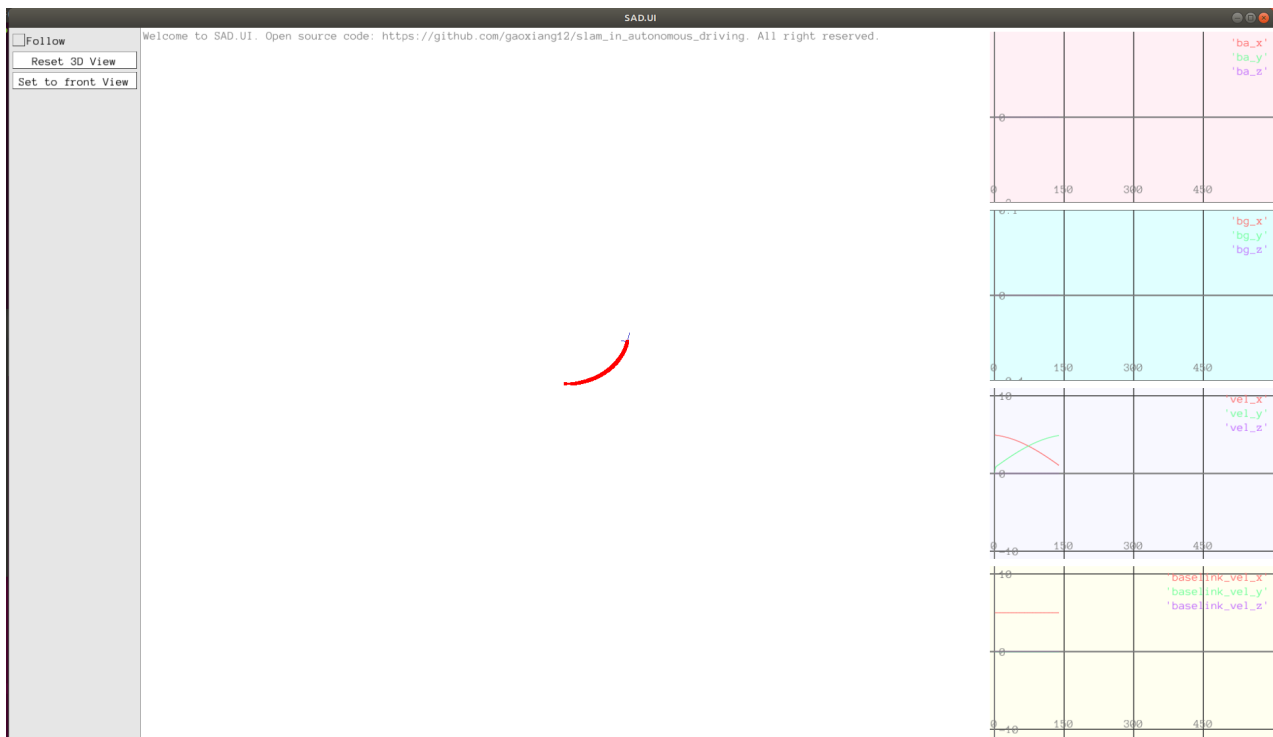
通过浏览器或vnc件进入docker容器的workspace中

```
1 | # 首先编译g2o
2 | cd slam_in_autonomous_driving/thirdparty/g2o/
3 | cmake .
4 | make -j8 # 按照自己电脑性能设置使用的核心数，这里使用了8个
5 |
6 | # 然后正式编译课程代码
7 | cd ../../../
8 | mkdir build
9 | cd build
10 | make -j8 # 按照自己电脑性能设置使用的核心数
```

编译完成就可以尝试运行一下代码：

```
1 | ./bin/motion
```

可以看到如下界面则证明编译成功，可以开始愉快地学习了😊



在之后的学习代码中，同学们可以使用**vscode**中的**Dev Containers**插件进入docker容器，从而进行代码编写，然后使用浏览器或vnc进行效果的可视化。由于使用了存储空间映射，workspace中的内容独立于容器，容器关闭后不影响workspace中的文件，即编译一次后，下一次进入docker不需要重新编译程序。

5 已知问题

5.1 终端使用不了g2o_viewer

sad代码依赖thirdparty中的g2o，为了减少image的大小，image中并未安装g2o。因此在运行第九章代码时使用不了g2o_viewer查看闭环检测的位姿图。解决思路为：

- 1 # 在源码中启动g2o
- 2 ./thirdparty/g2o/bin/g2o_viewer
- 3
- 4 # 通过g2o_viewer中的“load”加载第九章的位姿图