

Apollo 本地调试系统安装指南

1、本地调试系统安装总流程

欢迎您参加由百度举办的百度 Apollo 城市道路自动驾驶虚拟仿真赛，本赛题深度还原自动驾驶车辆在城市道路行驶场景，赛所用的地图场景由百度 Apollo 提供，参赛选手需按照赛题要求，基于本地竞赛环境 Dreamview 进行软件算法开发与测试，测试完成后提交代码到 ApolloStudio 线上评测系统进行评测。为确保赛事顺利进行，特编写《本地调试系统安装指南》，以便参赛选手下载本地调试系统，顺利完成竞赛任务的调试开发。本地调试系统安装流程如图 1 所示：



图 1 本地调试系统安装流程示意图

2、本地调试系统安装要求

电脑系统	版本要求	硬件配置要求	备注
Ubuntu	Ubuntu18.04 +	CPU ≥ 4 核 内存 $\geq 8G$ (推荐 16G)	可以采用单机或者双系统方式安装 Ubuntu 18.04.6 版。
Windows	WSL2	CPU ≥ 4 核 内存 $\geq 16G$	可以在 Windows 的 WSL2 中安装 Ubuntu 18.04.6 版本。

3、本地调试系统安装步骤

3.1 安装 ubuntu

原则上，Apollo 可运行于多数的 Linux 发行版，但经过测试，我们推荐您使用 Ubuntu 18.04.6 作为主机的操作系统

- (1) 安装 Ubuntu 18.04 的步骤，请参见 Ubuntu 官方或互联网上安装指南。
- (2) 安装完成后，请执行更新软件的操作，操作指令如下所示。

```
sudo apt-get update
```

3.2 配置 gitee 公钥

本次竞赛所用 Apollo 代码需要参赛选手从 Gitee 拉取。Gitee 提供了基于 SSH 协议的 Git 服务，在使用 SSH 协议访问仓库之前，需要先配置好账户/仓库的 SSH 公钥。

- (1) 注册 Gitee 账号，注册地址：<https://gitee.com/>
- (2) 按照 Gitee 帮助中心 > 生成/添加 SSH 公钥进行操作，操作链接：

<https://gitee.com/help/articles/4181#article-header0>

3.3 安装 Apollo

为了便于大家快速稳定的安装 apollo，我们提供了自动化安装脚本，该脚本中实现了 git 的安装、docker 安装及 docker 配置、拉取 Gitee 上 edu_sim_contest 分支代码以及拉取 docke_img 镜像。

请下载大赛资料页 docker.sh.zip 压缩包，解压缩并将其放置于当前用户的主目录下，然后打开命令行窗口，进入主目录，执行 apollo 自动化安装脚本。

```
bash docker.sh
```

此过程视网速快慢大概需要 1.5h 以上（约拉取 20G 资源，请确保网速与流量适中），请安装过程耐心等待，切勿随意中断操作。在安装过程中有以下几个关键步骤示意图如下所示：

```
=====
To run Docker as a non-privileged user, consider setting up the
Docker daemon in rootless mode for your user:

    dockerd-rootless-setuptool.sh install

Visit https://docs.docker.com/go/rootless/ to learn about rootless mode.

To run the Docker daemon as a fully privileged service, but granting non-root
users access, refer to https://docs.docker.com/go/daemon-access/

WARNING: Access to the remote API on a privileged Docker daemon is equivalent
to root access on the host. Refer to the 'Docker daemon attack surface'
documentation for details: https://docs.docker.com/go/attack-surface/

=====
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
Adding user apollo to group docker
Please reboot the computer and run the scripts again!
Initialized empty Git repository in /home/apollo/.git/
Cloning into 'apollo'...
remote: Enumerating objects: 332764, done.
remote: Counting objects: 100% (208/208), done.
remote: Compressing objects: 100% (171/171), done.
```

图 2 docker 安装及配置完成示意图

```
[INFO] Create volume apollo_faster_rcnn_volume_apollo from image: apolloauto/apollo:faster_rcnn_volume-traffic_light_detection_model-x86_64-latest
[INFO] Start pulling docker image apolloauto/apollo:faster_rcnn_volume-traffic_light_detection_model-x86_64-latest ...
faster_rcnn_volume-traffic_light_detection_model-x86_64-latest: Pulling from apolloauto/apollo
5d20c808ce19: Pull complete
893a5d077c3f4: Pull complete
Digest: sha256:c57f57aba718ad948d27d98e50115c6955fbeb054e99c269aa4d1830384846
Status: Downloaded newer image for apolloauto/apollo:faster_rcnn_volume-traffic_light_detection_model-x86_64-latest
docker.io/apolloauto/apollo:faster_rcnn_volume-traffic_light_detection_model-x86_64-latest
[INFO] Create volume apollo_smoke_volume_apollo from image: apolloauto/apollo:smoke_volume-yolo_obstacle_detection_model-x86_64-latest
[INFO] Start pulling docker image apolloauto/apollo:smoke_volume-yolo_obstacle_detection_model-x86_64-latest ...
smoke_volume-yolo_obstacle_detection_model-x86_64-latest: Pulling from apolloauto/apollo
5d20c808ce19: Already exists
4b0b6cdc565b: Pull complete
```

图 3 Apollo 代码拉取完成示意图

```
Digest: sha256:848c6b03e7398155a07975768d64e4adcc6b2ec65d2ac2f1c46838f5c3f9
Status: Downloaded newer image for apolloauto/apollo:smoke_volume-yolo-obstacle_detection_model-x86_64-latest
docker.io/apolloauto/apollo:smoke_volume-yolo-obstacle_detection_model-x86_64-latest
[warn] Starting docker container "apollo_dev_apollo" ...
+ docker run -itd --privileged --name apollo_dev_apollo --e DISPLAY=:0 --e DOCKER_USER=apollo --e DOCKER_GRP=apollo --e DOCKER_GRP_ID=1000 --e DOCKER_IMG=apolloauto/apollo
dev-x86_64-18.04-20210914_1336 --e USE_GPU_HOST=0 --e NVIDIA_VISIBLE_DEVICES=all --e NVIDIA_DRIVER_CAPABILITIES=compute,video,graphics,utility --volume apollo_map_volume-sunnyvale_big_loop_apollo:/apollo/mo
dules/map/data/sunnyvale_big_loop --volume apollo_map_volume-sunnyvale_loop_apollo:/apollo/modules/map/data/sunnyvale_loop --volume apollo_map_volume-sunnyvale_with_two_offices_apollo:/apollo/modules/map/
data/sunnyvale_with_two_offices --volume apollo_map_volume-san_nateo_apollo:/apollo/modules/map/data/san_nateo --volume apollo_audio_volume_apollo:/apollo/modules/audio/data/ --volume apollo_yolov4_volume
_apollo:/apollo/modules/perception/camera/lib/obstacle_detector/yolov4/model/ --volume apollo_faster_rcnn_volume_apollo:/apollo/modules/perception/production/data/perception/camera/models/traffic_light_de
tection_faster_rcnn_model --volume apollo_smoke_volume_apollo:/apollo/modules/perception/production/data/perception/camera/models/yolo_obstacle_detector/smoke_libtorch_model --v /home/apollo/apollo/apollo
0.0.1 --add-host baidu:127.0.0.1 --hostname in-dev-docker --shm-size 2G --pid=host --v /dev/null:/dev/raw1394 apolloauto/apollo:dev-x86_64-18.04-20210914_1336 /bin/bash
ba7222b4de7a2945550f2fcc03c607d85d5c8b94762b31218599639a11f7f
+ '[' 0 -ne 0 ']'
+ set +x
Adding group 'apollo' (GID 1000) ...
Done.
Adding user 'apollo' ...
Adding new user 'apollo' (1000) with group 'apollo' ...
Creating home directory '/home/apollo' ...
Copying files from '/etc/skel' ...
[ OK ] Congratulations! You have successfully finished setting up Apollo Dev Environment.
[ OK ] To login into the newly created apollo_dev_apollo container, please run the following command:
[ OK ] bash docker/scripts/dev_into.sh
[ OK ] Enjoy!
```

图 4 docker 环境启动示意图

3.4 编译 Apollo

- (1) 在图 4 所示的 docker 环境下，执行进入 docker 环境指令。

```
bash docker/scripts/dev_into.sh
```

进入成功后，用户名会变为 in-dev-docker，操作示意如图 5 所示：

```
[ OK ] To login into the newly created apollo_dev_apollo container, please run the following command:
[ OK ] bash docker/scripts/dev_into.sh
[ OK ] Enjoy!
apollo@apollo:~/apollo$ bash docker/scripts/dev_into.sh
apollo@in-dev-docker:/apollo$
```

图 5 成功进入 docker 环境示意图

- (2) 启动 docker 环境成功后，在当前命令窗口下，执行 apollo 编译指令。

```
bash apollo.sh build
```

首次编译需要从网上拉取部分依赖包，因此受网速与电脑配置影响大概需要 40 分钟左右，请耐心等待编译完成，编译完成样式如图所示：

```
Starting local Bazel server and connecting to it...
(05:33:38) INFO: Invocation ID: 20691677-8cde-453c-994a-81053cfee279
(05:33:38) INFO: Current date is 2022-04-06
(05:33:44) INFO: Analyzed 4353 targets (679 packages loaded, 15887 targets configured).
(05:33:44) INFO: Found 4353 targets...
(05:33:52) INFO: Elapsed time: 15.019s, Critical Path: 1.34s
(05:33:52) INFO: 10 processes: 10 internal.
(05:33:52) INFO: Build completed successfully, 10 total actions
=====
[ OK ] Done building apollo. Enjoy!
=====
[apollo@in-dev-docker:/apollo]$
```

图 6 Apollo 代码编译成功示意图

3.5 调试 Apollo

当完成 2.4 节编译 Apollo 代码后，参赛选手就可以根据《竞赛规则任务书》进行竞赛任务的开发与调试。为确保参赛者快速掌握使用 sim_control 组件来调试 Apollo 代码的方法，请各位参赛者根据《SampleCode 使用指南》进行学习。

4、参考资料

- 1、生成/添加 SSH 公钥: <https://gitee.com/help/articles/4181#article-header0>
- 2、Gitee 代码管理: <https://gitee.com/help/articles/4122>
- 3、Apollo 赛事代码分支: https://gitee.com/ApolloAuto/apollo/tree/edu_sim_context/

5、Q&A

- 1、在虚拟机中可以运行 apollo 吗?

经测试, 使用本自动化安装脚本可正常安装 Apollo 系统, 但在虚拟机中, 编译 Apollo 会出现较多问题, 建议采用单机 ubuntu 系统或者电脑安装双系统的方法运行 Apollo, 使用该方法要求电脑内存不低于 8G, 为了更好的体验, 建议内存 16G。

- 2、Windows 的子系统 ubuntu 可以运行 apollo 吗?

经过测试, 在 windows 的子系统 WSL2 中可以使用本自动化安装脚本可正常安装 Apollo 系统并稳定运行 Apollo, 但内存最低配置要求 16G, 低于 16G, 编译 Apollo 会出现较多问题。