TIANCHI天池

AI开发者的Docker实践

Docker practice for AI developers

第一本面向AI开发者的docker实践指导

玩转天池大赛docker提交

扫码加入天池
龙珠计划 Docker 训练营学习交流群

阿里云开发者“藏经阁”
海量免费电子书下载

目录

前言

随着云原生、AI 等技术的向前推进，容器技术逐渐成为每位算法同学的必备技能之一。本文档主要面向算法开发同学，从 0 基础实现将代码打包 docker 镜像–调试–提交仓库–提交云服务训练模型/天池大赛提交/学校服务器训练 等流程。也同样适用于初次接触 docker 的同学。

区别于开发同学，对于算法同学而言 仅需要掌握一部分基础命令达到自己的使用目的即可。因此此次简明教程面向算法同学和 AI 竞赛参赛者，帮助大家快速上手大赛提交和远程服务器训练。

大家还可以访问下方链接或者扫描下方二维码查看本教程的相关的视频教程。

https://tianchi.aliyun.com/course/351

基础概念

TCC（天池竞赛计算平台-TianChi Computing），天池大赛 docker 提交模型评估的平台简称。

docker 作为虚拟机领域成熟的轻量化容器产品，可以轻松的将代码和所依赖的整个环境（可以理解为包含整个操作系统）都打包在一起，不依赖于软件环境，方便把自己的代码从 windows 电脑分享到 mac 电脑运行、或者服务器上运行等。

docker 三要素：镜像（image）、容器（contarin）、registry(包含多个仓库)

镜像：顾名思义就是咱们将要把代码和环境打包在一起的这个产物，就是镜像。

registry：那么镜像存储在哪里呢 所以就有了 registry，是各云厂商提供的镜像存取服务，类似网盘，将镜像存储在云端仓库，方便我们随时随地在不同的介质上运行自己的代码或分享代码。比如你要把本地开发好的代码放在服务器上做耗时的训练动作，那么只需要在服务器上直接拉取自己云端的镜像运行即可。当然除了存储以外还有诸如版本管理等服务功能，类似 git。

容器：运行起来的镜像我们称之为容器，可以理解为运行环境或者实例。其实质是进程，随着代码运行结束，进程结束容器也就消失了。

环境篇

docker 安装

这里主要介绍 linux 、Windows 10 和 macOS 上的安装。

Linux

```
$ sudo curl -sS https://get.docker.com/ | sh
```

测试

```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
d1725b59e92d: Pull complete
Digest: sha256:0add3ace90ecb4adbf7777e9aacf18357296e799f81cabc9fde470971e499
788
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/
```

*如果机器有支持深度学习的 Gpu，可以继续执行如下命令以支持容器对 gpu 的调用[目前仅支持 linux]:

```
# Add the package repositories
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add -
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo te
e /etc/apt/sources.list.d/nvidia-docker.list



sudo apt-get update && sudo apt-get install -y nvidia-container-toolkit
sudo systemctl restart docker
```

点击下方课程链接即可直达 **Linux 下安装 Docker** 视频教程。

https://tianchi.aliyun.com/course/351/4127

windows

手动下载安装

点击以下链接下载 Stable (opens new window) 或 Edge (opens new windo w)版本的 Docker Desktop for Windows。

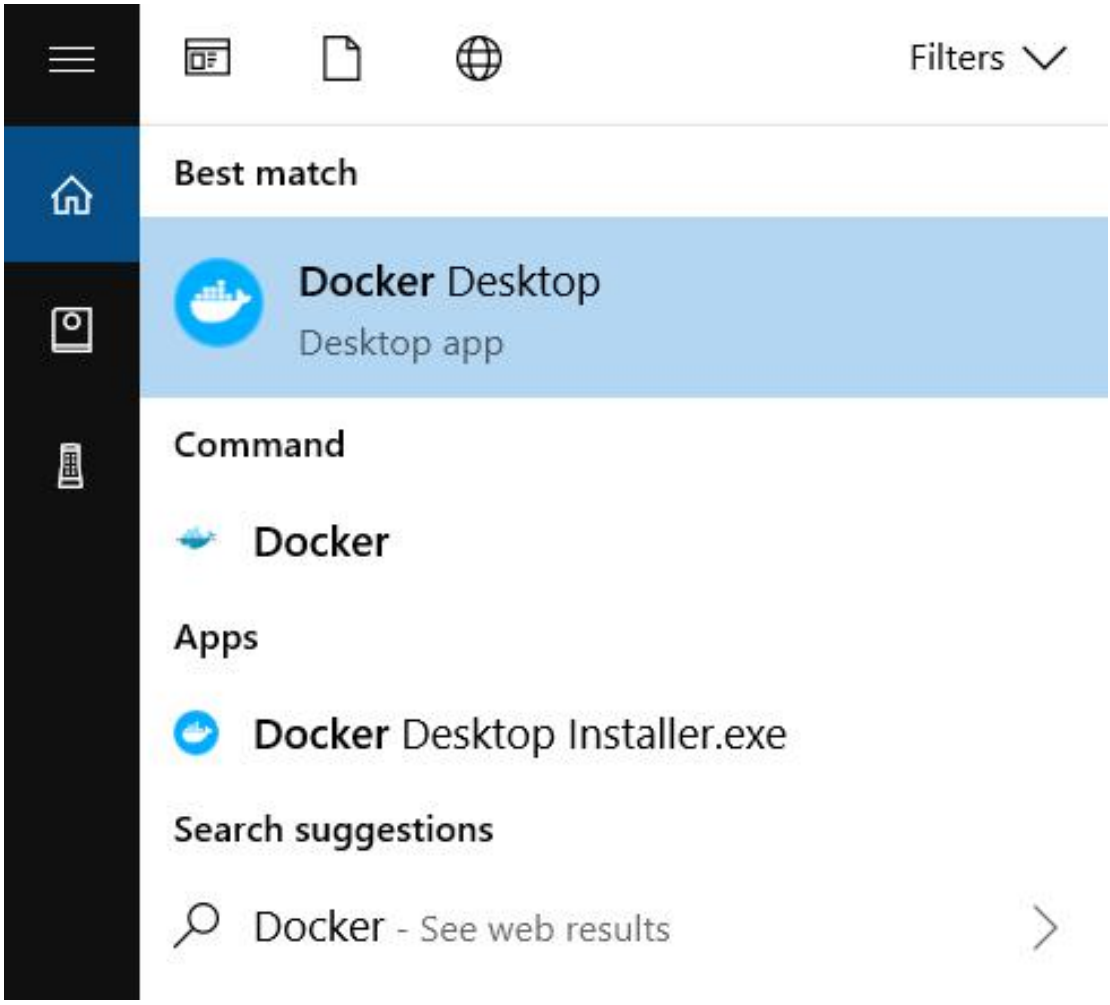
下载好之后双击 `Docker Desktop Installer.exe` 开始安装。

使用 winget (opens new window) 安装

```
# stable
$ winget install Docker.DockerDesktop
# edge
$ winget install Docker.DockerDesktopEdge
```

运行

在 Windows 搜索栏输入 **Docker** 点击 **Docker Desktop** 开始运行。



Docker 启动之后会在 Windows 任务栏出现鲸鱼图标。

等待片刻，点击 Got it 开始使用 Docker。

macOS

如果你已经安装了 Homebrew ，使用 brew 安装非常方便：

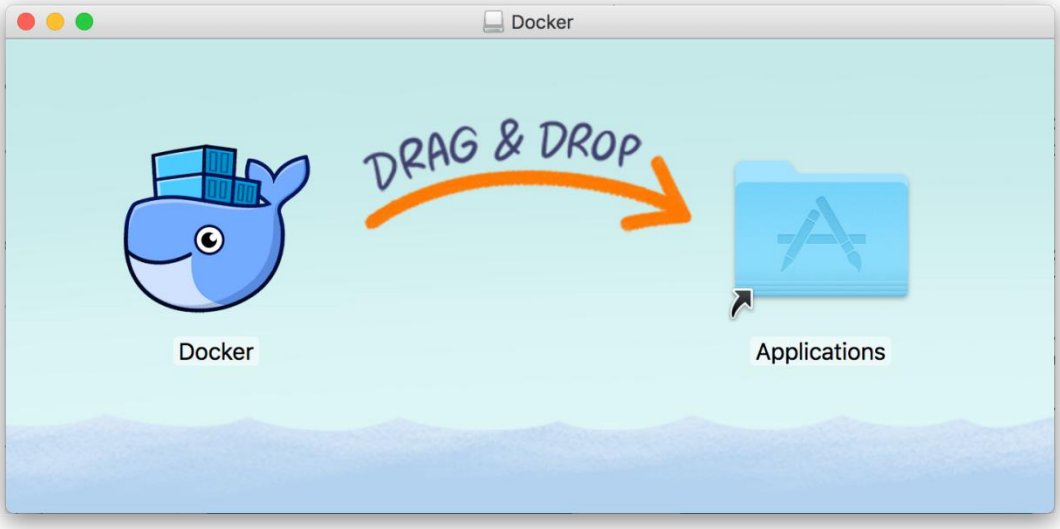
Homebrew (opens new window)的 Cask (opens new window)已经支持 Docker Desktop for Mac，因此可以很方便的使用 Homebrew Cask 来进行安装：

```
$ brew cask install docker
```

没有安装 brew 也可以手动下载安装：

点击下载 Stable (opens new window)或 Edge (opens new window)版本的 Docker Desktop for Mac。

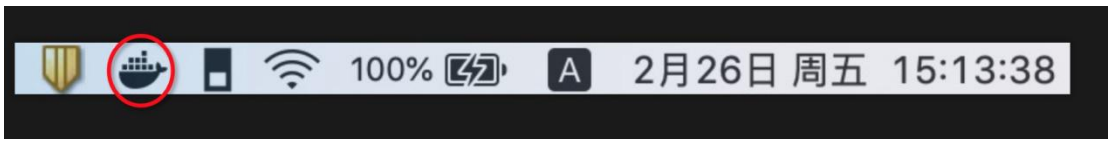
如同 macOS 其它软件一样，安装也非常简单，双击下载的 `.dmg` 文件，然后将那只叫 Moby (opens new window)的鲸鱼图标拖拽到 `Application` 文件夹即可（其间需要输入用户密码）。



运行

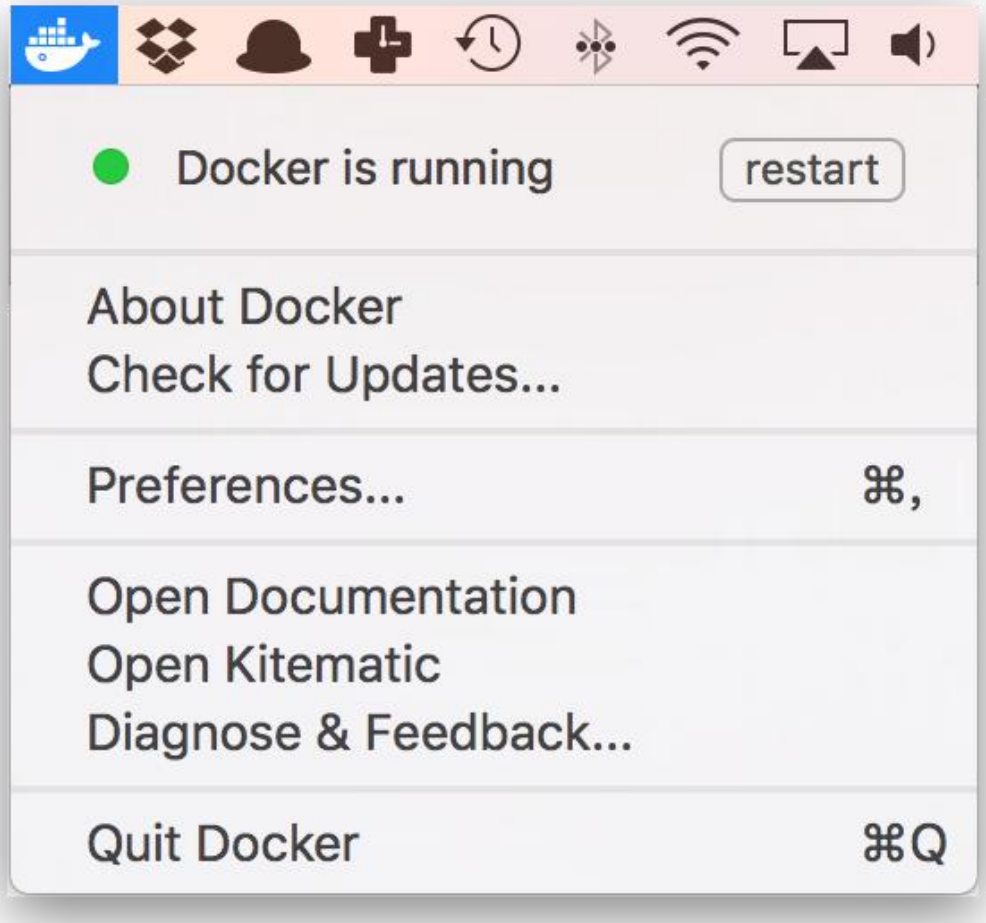
从应用中找到 Docker 并点击运行。

运行之后，会在右上角菜单栏看到多了一个鲸鱼图标，这个图标表明了 Docker 的运行状态。



第一次点击图标，可能会看到这个安装成功的界面，点击 "Got it!" 可以关闭这个窗口。

以后每次点击鲸鱼图标会弹出操作菜单。



启动终端后，通过命令可以检查安装后的 Docker 版本。

```
$ docker --version
Docker version 19.03.8, build afacb8b
$ docker-compose --version
docker-compose version 1.25.5, build 8a1c60f6
```

测试：

```
$ docker run hello-world

Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
d1725b59e92d: Pull complete
Digest: sha256:0add3ace90ecb4adbf7777e9aacf18357296e799f81cabc9fde470971e499
788
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/
```

点击下方课程链接即可直达 **macOS 下安装 Docker** 视频教程。

https://tianchi.aliyun.com/course/351/4128

docker 基础命令学习

拉取镜像

```
docker pull [选项] [docker 镜像地址:标签]
```

如:

```
docker pull hello-world:latest
```

运行镜像

```
$ docker run 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/
```

运行镜像并进入容器

```
$ docker run -it --rm ubuntu:18.04 bash
root@e7009c6ce357:/#  uname -a
Linux bff9f261bab2 4.15.0-106-generic #107-Ubuntu SMP Thu Jun 4 11:27:52 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
root@e7009c6ce357:/# exit
```

`docker run` 就是运行容器的命令,后面如果只跟镜像，那么就执行镜像的默认命令然后退出。

- `-it`：这是两个参数，一个是 `-i`：交互式操作，一个是 `-t` 终端。我们这里打算进入 `bash` 执行一些命令并查看返回结果，因此我们需要交互式终端。
- `--rm`：这个参数是说容器退出后随之将其删除。默认情况下，为了排障需求，退出的容器并不会立即删除，除非手动 `docker rm`。我们这里只是随便执行个命令，看看结果，不需要排障和保留结果，因此使用 `--rm` 可以避免浪费空间。
- `ubuntu:18.04`：这是指用 `ubuntu:18.04` 镜像为基础来启动容器。
- `bash`：放在镜像名后的是 **命令**，这里我们希望有个交互式 Shell，因此用的是 `bash`。

进入容器后，我们可以在 Shell 下操作，执行任何所需的命令。通过 `exit` 退出。

查看本地镜像（list 镜像）

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
  SIZE
  redis               latest              5f515359c7f8        5 days ago
          183 MB
  nginx               latest              05a60462f8ba        5 days ago
          181 MB
```

IMAGE ID 是镜像的唯一标识。

查看运行中的容器

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
9363b1b51118        testlog:1           "bash"              7 weeks ago
Up 7 weeks                              vigilant_bhaskara
```

CONTAINER ID 容器唯一 id 可以通过指定这个 ID 操作 exec shell 进入容器、commit 这个容器的修改、tag 给这个容器打标签等。

docker ps 罗列的是当前活跃的容器。

要查看所有容器执行 docker ps -a：

```
$ docker ps -a
```

进入运行中/后台运行的容器

```
$ docker exec -it [CONTAINER ID] /bin/bash
```

进入运行中的容器后不仅可以调试镜像，还可以对镜像做修改如安装 python 包，如果想对修改做保留，可以执行 2.7 提交。

保存修改

```
docker commit [CONTAINER ID] registry.cn-shanghai.aliyuncs.com/test/pytorch:myversion
```

注意：通过 commint 的形式保存现场为一个新的镜像虽然也能直观的达到构建新镜像的目的，但是实际操作中，并不推荐这种形式，因为 1.commit 操作不仅会把有用的修改保存下来，对一些无关的修改也会保存下来（每一个命令行操作都会生成存储如 ls 操作）就会导致镜像比较臃肿；2.因为 commit 操作属于黑箱操作，后续如果有什么问题维护起来会比较麻烦。

建议 commit 仅作为保留现场的手段，然后通过修改 dockerfile 构建镜像。

打 TAG

有时需要对临时版本，或者节点版本做一个标记保留，打 TAG 标签非常好用，并不会额外占用空间：

```
docker tag registry.cn-shanghai.aliyuncs.com/test/pytorch:myversion my_tmp_version:0.1
```

推送镜像到仓库

```
docker push registry.cn-shanghai.aliyuncs.com/test/pytorch:myversion
```

使用 dockerfile 构建镜像

Dockerfile 示例(注意一般文件名命名为 Dockerfile 无后缀名，如果命名为其他名字，构建时需要额外指定文件名)：

```
# Base Images
## 从天池基础镜像构建(from 的 base img 根据自己的需要更换，建议使用天池 open list 镜像链接：https://tianchi.aliyun.com/forum/postDetail?postId=67720)
FROM registry.cn-shanghai.aliyuncs.com/tcc-public/python:3
##安装依赖包
RUN pip install numpy -i https://pypi.tuna.tsinghua.edu.cn/simple
##或者从 requirements.txt 安装
##RUN pip install --no-cache-dir -r requirements.txt -i https://pypi.tuna.tsinghua.edu.cn/simple
## 把当前文件夹里的文件构建到镜像的根目录下,并设置为默认工作目录
ADD . /
WORKDIR /
## 镜像启动后统一执行 sh run.sh
CMD ["sh", "run.sh"]
```

点击下方课程链接即可直达 dockerdfile 入门视频教程。

https://tianchi.aliyun.com/course/351/4129

构建镜像

```
docker build -t registry.cn-shanghai.aliyuncs.com/target:test .
```

如要指定 dockerfile：

```
docker build -f ./dockerfile -t
registry.cn-shanghai.aliyuncs.com/target:test .
```

删除镜像/容器

删除镜像：

```
docker rmi registry.cn-shanghai.aliyuncs.com/target:test
```

删除容器：

```
docker rm [CONTAINER ID]
```

如果容器还在运行，则会删除失败，应先结束掉容器：

```
docker kill [CONTAINER ID]
```

查看运行中的容器：

```
docker ps
```

查看所有容器：

```
docker ps -a
```

常规技巧

- 检查基础镜像软件源和 pip 源是否替换为国内源，如果非国内源后续每次构建镜像会比较浪费时间。
- 必备软件包可直接安装于基础镜像内，以减少每次构建镜像时都要安装一遍的等待时间。
- 镜像面临调试问题时，可交互式进入容器后直接调试修改，直到成功后退出再在 dockerfile 中修改。
- 养成使用 Dockerfile 的习惯，不要依赖于 commit。
- 每次镜像修改都给定新的版本号或标签，方便区分版本管理，有意义的版本最好使用有含义的字符作为版本号，如：frist_submit。

深度学习常用镜像集合（包含国内源和海外源）

https://tianchi.aliyun.com/forum/postDetail?postId=67720

海外选手网速受限提交方案

- 使用 github 等代码托管（推荐 code.aliyun.com）。
- 在 cr（阿里云容器服务）产品做代码源绑定，并开启自动构建。

- 构建完成设置 hook 自动提交（警惕次数消耗），或者谨慎起见构建完成后手动提交。

触发器设置方法：

https://help.aliyun.com/document_detail/60949.html?spm=5176.8351553.0.dexternal.7f231991sUvASL

点击下方课程链接即可直达**如何创建自己的基础镜像**视频教程。

https://tianchi.aliyun.com/course/351/4131

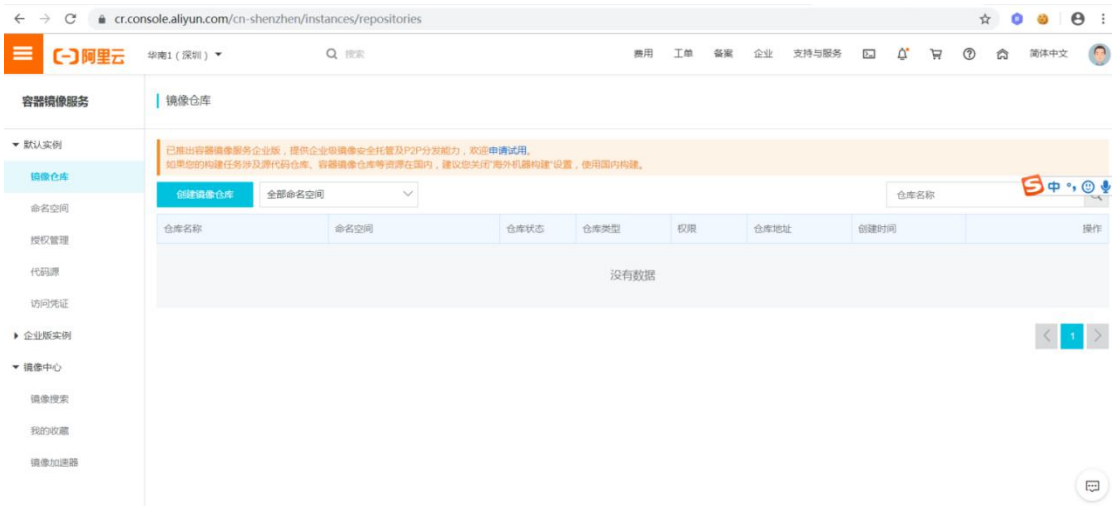
实践篇

创建镜像仓库账号

这里以申请阿里云容器镜像服务（免费），并创建仓库为例，其他仓库如 dockerhub、谷歌、亚马逊、腾讯等详见对应产品说明书。

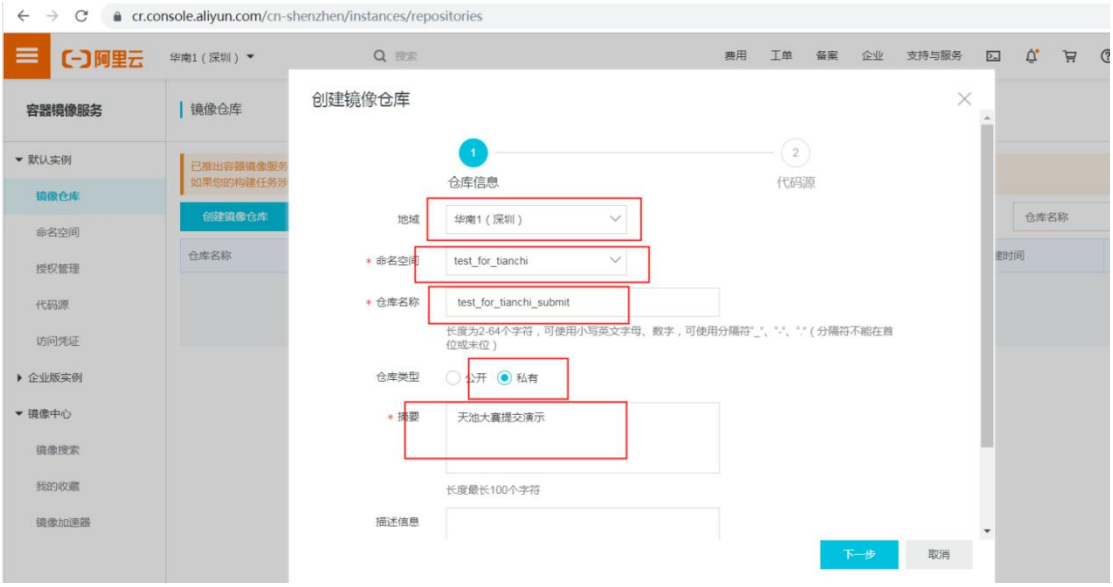
打开阿里云容器服务地址为（https://cr.console.aliyun.com）

注册开通后产品页面如下:



第一步切换标签页到命名空间，创建地址唯一的命名空间。

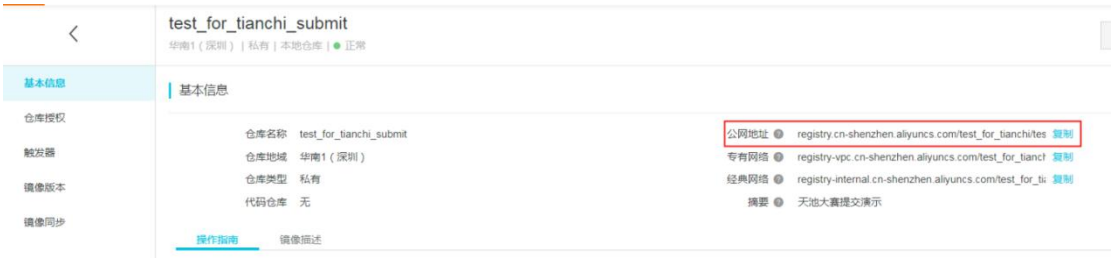
根据大赛要求选择对应的地域，其他的按照自己需求选择或填写。



下一步，选择本地仓库，不建议其他选项，完成创建。

点击管理，可查看详情。

详情页如下，有基本的操作命令，仓库地址一般使用公网地址即可。



按照页面的指令在本地完成登陆:

```
export DOCKER_REGISTRY= your_registry_url<docker registry url>
(注意这里 your_registry_url 最后字段结尾，不能多不能少 E.g
registry.cn-shanghai.aliyuncs.com/xxxx/xxxx)
docker login $DOCKER_REGISTRY \
    --username your_username \
    --password your_password
```

实践 docker 练习赛

练习赛链接&资料

- gpu 版本练习赛链接 【推荐】
- cpu 版本练习赛链接
- 从 0 开始大赛 docker 提交视频演示

GPU 版 docker 练习赛实践

这里以 GPU 版 docker 练习赛中的练习一为例来构建镜像并提交:

首先我们写一个 main.py，实现读取/tcdata 下的数据，计算 a*b 生成 result.npy 文件:

```
#main.py
import os
import numpy as np
import torch
```

```
device = torch.device("cuda")

data_dir = '/tcdata'
a = np.load(os.path(data_dir,a.npy))
b = np.load(os.path(data_dir,b.npy))

a = torch.from_numpy(a).to(device)
b = torch.from_numpy(b).to(device)
c = torch.matmul(a,b).cpu()

print(c)
np.save("result.npy", c)
```

编写入口文件 run.sh：

```
#bin/bash
#打印 GPU 信息
nvidia-smi
#执行 math.py
python3 math.py
```

然后编写 Dockerfile 用于打包 main.py 和运行环境为镜像：

```
# Base Images
## 从天池基础镜像构建(from 的 base img 根据自己的需要更换，建议使用天池 open list 镜像链接：https://tianchi.aliyun.com/forum/postDetail?postId=67720)
FROM registry.cn-shanghai.aliyuncs.com/tcc-public/pytorch:1.1.0-cuda10.0-py3
##安装 python 依赖包
RUN pip install numpy -i https://pypi.tuna.tsinghua.edu.cn/simple
## 把当前文件夹里的文件构建到镜像的根目录下,并设置为默认工作目录
ADD . /
WORKDIR /
## 镜像启动后统一执行 sh run.sh
CMD ["sh", "run.sh"]
```

命令行执行，构建镜像：

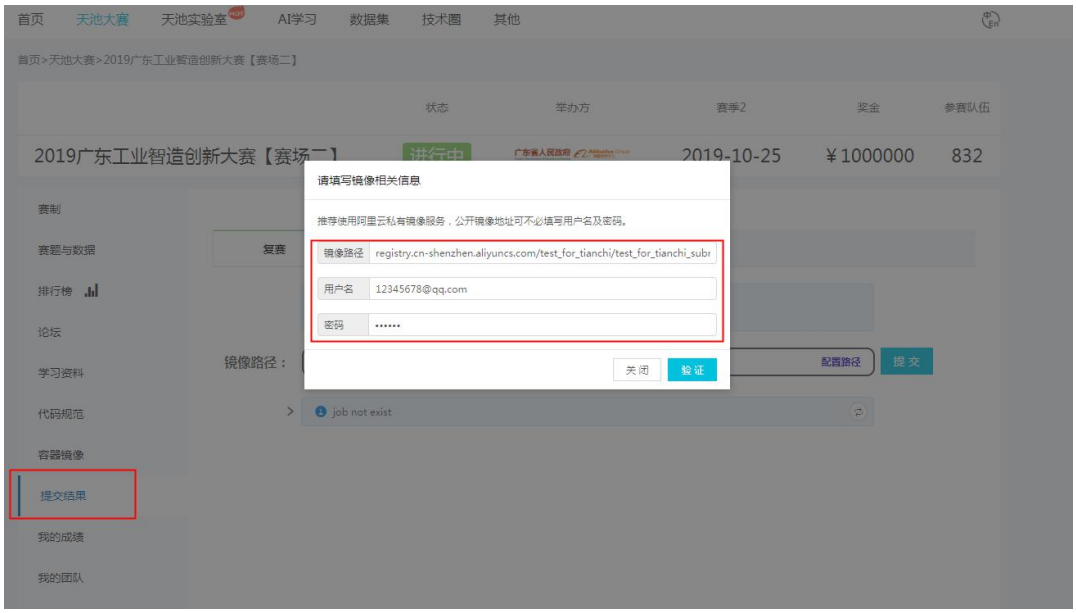
tips：镜像命名根据自己申请的仓库 registry 来，可以省去 tag 步骤直接上传，保持本地镜像清洁。

```
$ docker build -t registry.cn-shanghai.aliyuncs.com/xxxx/test:0.1 .
```

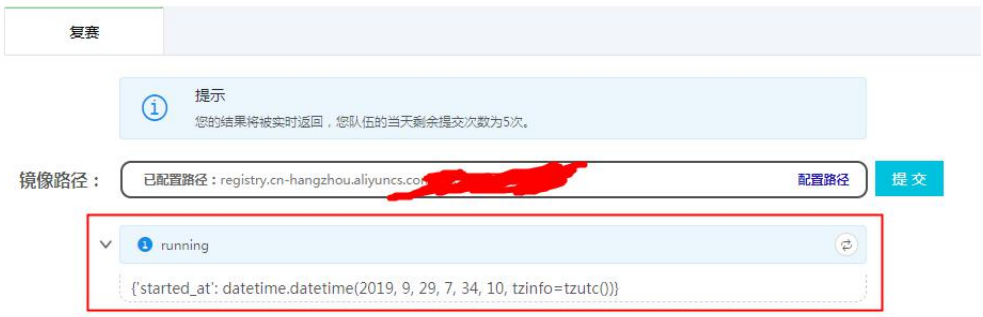
上传镜像仓库：

```
$ docker push registry.cn-shanghai.aliyuncs.com/xxxx/test:0.1
```

天池页面提交：



正在运行的状态 running,其他状态对应展开也可以看到详情，如错误状态展开可看到大致 reason。



运行结束或者运行失败都会有邮件通知到提交人留在天池的邮箱里，收到提示即可回到大赛页面查看成绩及日志。

docker 使用 gpu

docker 在新的版本中均已支持直接调用 gpu，通过--gpu 指定使用哪个 gpu，--gpu all 则是使用所有 gpu。

前提：请确保已按照前文环境篇 linux 末尾安装了 Nvidia 对 docker 的软件支持。

```
docker run --gpu all registry.cn-shanghai.aliyuncs.com/xxxx/test:0.1
```

调试:

```
docker run -it --gpu all
registry.cn-shanghai.aliyuncs.com/xxxx/test:0.1 /bin/bash
#  nvidia-smi
Thu Jan  7 19:04:55 2021
+-----------------------------------------------------------------------------
------------+
| NVIDIA-SMI 418.87.01    Driver Version: 418.87.01    CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------
----------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile
Uncorr. ECC |
```

```
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util
Compute M.
 |
|===============================+======================+============
==========|
|   0  Tesla P100-PCIE...  On   | 00000000:00:08.0 Off |                    0 |
| N/A   29C    P0    24W / 250W |      0MiB / 16280MiB |      0%      Default |
+-------------------------------+----------------------+----------
------------+
|   1  Tesla P100-PCIE...  On   | 00000000:00:09.0 Off |                    0 |
| N/A   31C    P0    26W / 250W |      0MiB / 16280MiB |      0%      Default |
+-------------------------------+----------------------+----------
------------+

+-----------------------------------------------------------------------------
------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================
=========|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------
------------+
```

天池竞赛规范

天池竞赛 TCC 规范

数据集挂载

天池大赛数据集默认以只读形式挂载在/tcdata 下（训练集+测试集）。

因此本地调试时可通过-v 模拟挂载与线上一致：

```
docker run -it -v /your_data:/tcdata
registry.cn-shanghai.aliyuncs.com/xxxx/test:0.1 /bin/bash
```

预训练权重

如果大赛允许使用预训练权重等外部数据，则默认以只读形式挂载在/external_data 下，一般情况下如果你需要的权重大赛未提供，可向赛题方提出，需要注意的是所有外部数据必须使用官方在线上提供的，否则如自己放在镜像里提交则在代码审核时直接取消名次。

持久化存储

如大赛出现超大数据集，常见视频类竞赛，官方会提供持久化存储以方便选手对数据处理、模型训练等耗时操作的中间结果提供持久化存储空间直到大赛结束。

持久化空间默认以可读写的形式挂载在/myspace 下，但是需要注意此空间仅为日常调试时使用，最终排行榜最优成绩对应提交需是端到端完整代码运行得出，不能使用存储空间的中间结果。

执行入口

大赛一律以镜像的工作空间目录下 run.sh 为运行入口，且仅以此进程为程序运行状态如结束的标志。

调试时可在本地执行 docker run registry.cn-shanghai.aliyuncs.com/xxxx/test:0.1 sh run.sh 模拟测试。

网络环境

天池大赛一律无网络链接，切记镜像内不要有网络操作否则会导致线上运行时由于无网络触达而导致程序卡住。

超时时间

一般大赛“容器镜像”页面会注明一个超时时间，此超时时间为你的代码“从开始运行到完整结束”的整个时间限制，如果你的代码仅仅超出这个时间一点（约 30min 内）则依然会返回成绩但是此成绩不会计入排行，仅能在日志页面内看到，如果超出这个时间许多还没结束，则会被强制清理掉。

另外 如果涉及流评测（后续大赛会逐渐被流评测代替）。

日常调试提交

如果你本地有满足赛题的计算资源，那么你日常提交时可以提交打包本地训练好的权重仅在线上做预测过程，这样可以快速出分，提高大赛效率，但是这样的提交在最终是无效的 所以一定注意在大赛要求的最后几天（一般是一周）重新提交最优模型的端到端代码完成数据处理到训练预测全流程，否则最后如果最优成绩无完整代码会被直接淘汰，所以切记！

自定义镜像

如需自定义镜像，一定要确保镜像内安装了 curl，且 Dockerfile 中不能使用 ENTRYPOINT。

线上存储空间大小

物理机存储默认为 500G 但是有系统、软件、docker 镜像等占用空间，实际可用空间建议通过打印获取，一般大赛足够使用，可不必考虑。

线上共享内存大小

共享内存大小无特殊说明情况下默认为内存的一半。

docker 本地调试时可指定如--shm_size 64 设置与线上环境大小一致。

流评测工具包 AI-HUB

天池大赛已逐渐全面升级为流评测形式，已规避针对测试集做训练、或者以未来测试条件预测当前数据、或工业流水线对单次预测效率严控等情况。

安装：pip install ai-hub

ai-hub 的 inferServer 模块针对天池大赛流评测定制，详情可参考：https://github.com/gaoxiaos/AI_HUB

如果自己构建大赛的 infer server 则需要注意 API 接口默认为 tccapi 端口为 8080 并实现当收到 exit 请求时服务自动退出能力。

常用基础镜像

天池为大家构建了常用的竞赛镜像仓库，并更新维护。点击直达

tips：

- 如有 list 中缺失的可在大赛群中提出诉求。
- list 中镜像仓库存放在阿里云上海地域，同地域上传下载速度更快。

如何停止正在运行的镜像

点击下方课程链接即可直达**如何停止正在运行的镜像**视频教程。

https://tianchi.aliyun.com/course/351/4132

一直处于 waiting 状态怎么办

点击下方课程链接即可直达一直处于 waiting 状态怎么办视频教程。

https://tianchi.aliyun.com/course/351/4136

docker&大赛

docker 大赛技巧

本地调试可以切一部分训练集做测试集通过 docker -v /drta_dir:/tcdata 模拟线上环境进行测试，一般本地成功运行线上都能通过。线上运行与本地运行的区别点：

- 本地有网络，线上无网络。
- 本地内存大小、存储大小、显存大小如果大于线上配置，则需要注意资源使用不要超过线上配置。

如 docker 本地测试运行时可通过-m 参数限制内存使用大小。

```
docker run -m 1024 registry.cn-shanghai.aliyuncs.com/xxxx/test:0.1 sh run.sh
```

- 线下驱动版本高于线上 nvidia 驱动版本，可能对应的 cuda 在线上不能支持。

可参见 Nvidia 官网的驱动版本与 cuda 的对照表。

- 遇到异常情况自己要完整跟一遍状态变化，关注是在那个环节出现问题，向官方求助时需说明状态变化及错误信息，并附带该次提交的日志链接，如果还在运行的，可提供历史日志链接。日志链接非常重要，用于快速查询你的队伍信息。

天池大赛下限制条件/注意事项/常见问题集

运行时间/效率：

概念解释：costtime：（单位 ms）与选手运行时间无关，无需关注，这个时间是系统读取选手预测结果并计算成绩耗费的时间。

运行时间计算方式为（选手镜像运行结束变为 terminal 时刻）-（选手镜像变为 running 时刻）-（系统评测占用时间 costtime）。

超时时间：

超时时间分为三个维度，一般大赛仅涉及两个维度（严重超时：从镜像提交到选手代码运行结束过程中时间超过官方规定的超时时间+30min 到 60min，视为程序严重超时/程序死机，此时会启动资源回收机制强制 kill 选手进程； 超时：从选手代码开始运行到结束运行时间超过官方规定的超时时间，则视为选手程序效率不符合赛题要求，成绩会被置为无效，运行时间超过“超时时间”但未超过“严重超时时间”会正常运行结束并可在日志页看到成绩，以供调整参考）；第三个超时维度为细粒度的 infer 效率控制，对于工业等流水线业务对实时性要求高的赛题可能会设置“单次 infer”超时时间，测试数据以流的形式依此喂给选手模型，模型“从收到单例数据到 infer 返回结果”时间需满足官方规定，超时则成绩无效。

日志异常

a.cant connect tianchi.aliyun.com（已知偶现 bug 跟随某台 ecs 存在，此时由于网络不可达且未代码运行，并不会扣除提交次数，可直接再次提交，并告知官方出现的问题及对应日志链接)。

长时间 init

init 一般为初始化状态，如果刚提交后出现 init 为正常现象，有时 init 很快变为 watiing 有时则需很久（不超过 30min），这是因为机器库存在根据大家提交量动态变化，提交量突增时机器库存不足则会触发自动购买添加动作，该动作可能会耗费较长时间 但不会超过 30 分钟，如果超过 30 分钟则不正常。

长时间 creating

如果状态一直停留在 creating，且展开状态栏的 reason 显示 imgbackoff，此时证明你的镜像在拉取时出现错误，可能镜像链接有误或者账号密码不对。

这里需要注意，每次修改镜像链接，包括仅修改版本号，都需要重新填写账号密码。

代码规范

为了方便代码审核，除特殊说明的大赛外，养成如下格式习惯：

/workspace #存放选手代码的目录，需要在 dockerfile 中设置此目录为 WORKDIR（官方镜像会逐渐整改，全部默认设置为/workspace ）。此目录下需包含且仅包含大赛的完整代码。

代码审核

代码审核没有特殊说明的大赛默认取的是（规定时间内）排行榜最优成绩那次提交对应的备份镜像进行审核。如果最优成绩对应镜像不符合审核要求（如不包含官方要求的完整代码）则会直接淘汰。如果出现前期仅提交预测代码得到的最优成绩在最后几天提交完整代码时无法复现出来，可以在官方群联系赛题组取消无法复现的最优成绩。

遇见问题如何提问?

遇到问题，不要仅描述问题，同时需附带日志链接和大赛名称。组委会优先处理信息完整的提问。

设置账号密码时提示验证通过，并非账号密码正确。

镜像仓库服务和提交运行服务（tcc）是独立的两套系统，此时验证的仅仅是镜像链接格式无误，不会验证账号密码。

运行过程中主动结束

遇到想要主动结束运行的情况时可通过直接覆盖提交实现运行新的提交，或提交一个错误链接终止掉正在运行的程序。

提交次数计数逻辑

提交次数的扣除仅以“状态进入 running”为唯一标准，如果镜像提交错误或还未变成 running 就覆盖提交都不会扣除提交次数。

实时日志

有些训练/预测时间特别久的比赛，如淘宝直播大赛视频类型竞赛，查看实时日志对选手而言非常重要，所以运行时间较长的比赛会开放实时日志，用于及早发现问题。通过点击实时日志框右上角刷新按钮更新最新日志（实时日志有大小限制，如果打印过多内容，历史日志会被实时截断，优先显示最新日志）。

其他技巧

docker 共享主机屏幕，实现可视化

算法同学不同于应用开发，经常在本地调试过程中需要可视化一些内容，如 plot 或者查看数据集里的一张图片等，而 docker 自身并不支持这些能力，但是你可以把本机的屏幕挂给容器(/tmp/.X11-unix)，这样在容器内执行 plot 时，画面会在本机桌面上直接显示，sample:（需要注意，如果你不是在本地运行，宿主机并没有屏幕可以共享给容器，此时想要可视化会比较麻烦，可通过 ai-hub 包绑定微信进行显示，详见 https://github.com/gaoxiaos/AI_HUB）。

```
docker run --gpus all -it -v /tmp/.X11-unix:/tmp/.X11-unix
registry.cn-shanghai.aliyuncs.com/tcc-public/super-mario-ppo:localdisplay /bin/bash
```

docker 容器后台运行

一些提供服务类的镜像，一般需要放在后台一直运行。或者在本地运行长时间任务（训练模型）的镜像，希望运行结束容器不销毁。

docker 提供了-d 选项 保持容器在后台：

```
docker run -d xserver:latest
```

两个 docker 容器绑定通信（一般用与分离的服务端和客户端）

当你有时需要一个固定的服务时，不妨把服务端单独拎出来作为后台容器一直存在，专注开发客户端代码，每次运行时通过(--link)绑定服务端容器，sample:

```
docker run -itd --link display:xserver client:latest /bin/bash
```

容器中文件读写权限问题

由于 docker 镜像一般都精简掉了非必要模块，所以不支持 sudo 权限，那么遇到目录写权限报错时该怎么做呢?

- 一般工作目录下都是读写全开的，最简单就是写在有权限的目录。
- 一定要写在该目录，可在 dockerfile 中 chmod 对应权限 777 是最高权限，你可以相应调整。

```
RUN chmod 777 /your_dir
```

- 同理，不想在 dockerfile 中执行的话 ，直接修改基础镜像在基础镜像中对指定目录执行 chmod 后作为新的基础镜像也可。

海外同学上传镜像慢的建议方案

由于大赛的计算节点一般都在中国的上海或深圳,所以当镜像较大时海外选手上传镜像到国内会非常慢，尽管有些大赛开通了海外专线，但是速度还是比较受限。

因此建议海外同学可以使用阿里云镜像服务（ACR）的自动构建（镜像）功能，步骤如下:

- 使用 github 等托管自己的代码（推荐 code.aliyun.com）。
- 在 acr（阿里云容器服务）产品页做代码源绑定，并开启自动构建。

- 构建完成设置 hook 自动提交（警惕次数消耗），或者谨慎起见构建完成后手动提交。

触发器设置方法：

https://help.aliyun.com/document_detail/60949.html?spm=5176.8351553.0.dexternal.7f231991sUvASL

容器中使用 plot 可视化图表，或者可视化数据集中的图片等数据

由于容器无显示设备，因此需要绑定宿主机的显示设备用来可视化，Linux 下可通过 3.1 节其他技巧的共享宿主机屏幕实现：

```
docker run -it -v /tmp/.X11-unix:/tmp/.X11-unix
registry.cn-shanghai.aliyuncs.com/tcc-public/super-mario-ppo:localdisplay /bin/bash
```

如果宿主机也没有显示设备，则需考虑把图片等保存下来传送到本地查看，这个流程比较复杂，这里推荐大家使用 ai-hub 的 plot 函数或命令行，直接会把图像发送至绑定的微信账号查看，详情查看（公共号能力尚未开通，可先关注，后续在公众号通知）https://github.com/gaoxiaos/AI_HUB

容器中做耗时任务（如训练模型）消息触达（通知）

消息触达模块可使用 AI-HUB 中的 Notice 模块，目前公众号尚未开通消息能力，可先关注公众号，后续更新会在公众号中通知。

https://github.com/gaoxiaos/AI_HUB

常见问题

天池大赛出现长时间 waiting,展开状态栏 reason 显示 imagebackoff

imgbackoff 一般是因为账号名密码或链接错误导致拉去镜像失败出现的错误。请重新填写链接、账号密码提交，此时不会扣提交次数。

tips:填写账号密码时，点确认按钮时提交验证通过，仅仅是验证 url 的合法性，并未进行账号密码验证，所以验证通过不代表账号密码正确。

本地出现一些<none>镜像

<none>镜像出现的原因是旧的镜像名被新的占用，导致旧的镜像变成了<none>，比如你 build 新的镜像时指定了就镜像相同的镜像名：版本号

删除<none>镜像可使用：

```
docker image prune
```

天池大赛日志为空，仅提示网络错误

此时可能遇到了 bad 节点，虽然机率较小，但是 k8s 社区仍然会有类似 bug，遇到时可重新提交即可，此时不会扣除提交次数。

镜像没有读写权限怎么办

点击下方课程链接即可直达**镜像没有读写权限怎么办**视频教程。

https://tianchi.aliyun.com/course/351/4130

附录

《AI 开发者的 Docker 实践》课程

大家直接点击访问下方链接或者扫描下方二维码即可进入《AI 开发者的 Docker 实践》课程页面进行学习。

https://tianchi.aliyun.com/course/351

天池龙珠计划 Docker 训练营

大家直接点击访问下方链接即可进入《天池龙珠计划 Docker 训练营》页面进行学习。

https://tianchi.aliyun.com/specials/activity/promotion/aicampdocker

扫码加入天池
龙珠计划 Docker 训练营学习交流群

阿里云开发者“藏经阁”
海量免费电子书下载