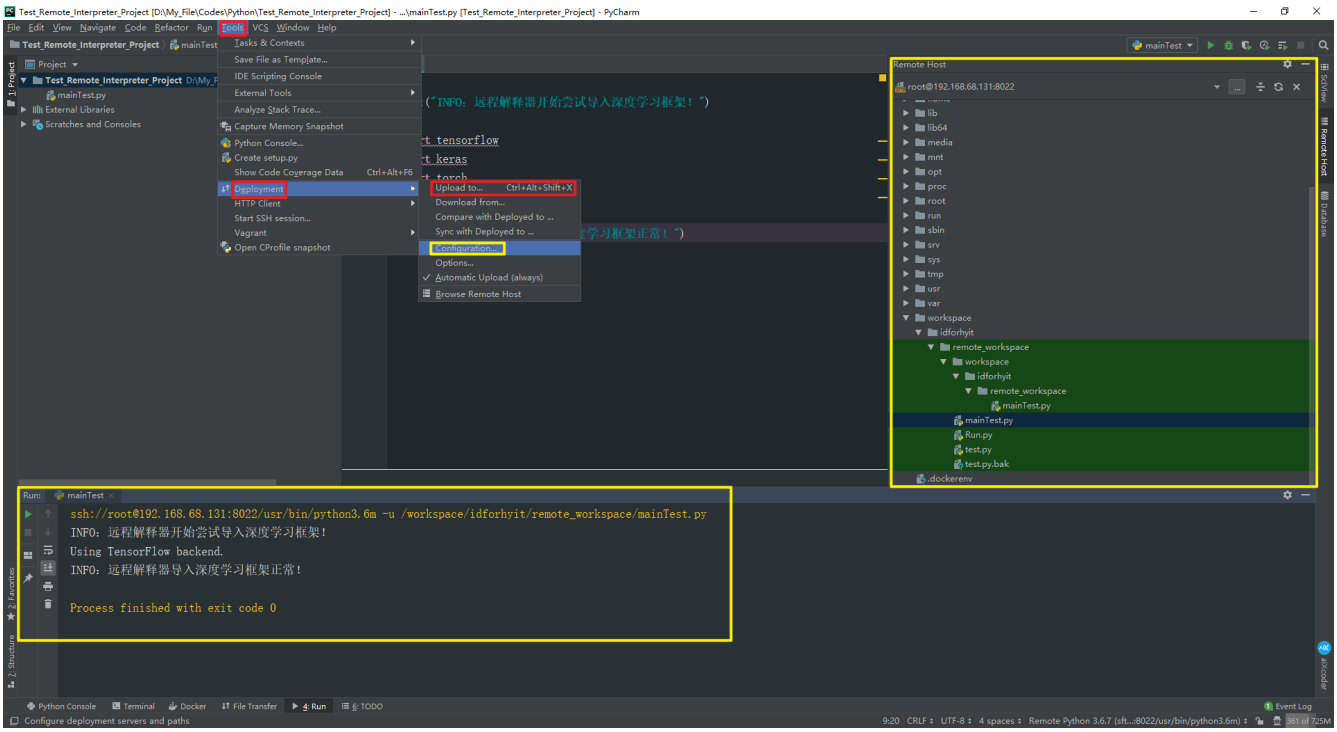
本文在线地址：https://idforhyit.github.io/2019/06/01/Docker-GPU-RemoteInterpreter/

Pycharm+Docker配置备忘录

> PyCharm+Docker：打造最舒适的深度学习炼丹炉

当你配置完成之后，你将能在本地PyCharm IDE上使用远程服务器的Python解释器和环境进行调试和运行，同时能够很方便的将代码在本地和服务器上进行同步。

> 本篇文章纯手码，请勿转载。文章内容来自本菜鸟的亲自操作，查阅了很多资料，但作为菜鸟一枚，肯定还存在许多错误，大佬轻喷，如见错误或操作不一致的地方，烦请在本篇博文地址的文章底部评论指出，戳我直达。



PyCharm+Docker的远程解释器效果图

目录

如有错误或疑问，烦请评论指出

流程命令总览

```
# 以管理员权限运行
su root
[password]

# 从DockerHub仓库下载一个深度学习作业系统的映像档
docker pull hub-mirror.c.163.com/ufoym/deepo
### 这里采用deepo，她几乎支持所有的深度学习框架，可以帮助俺们快速搭建一个包含TensorFlow、PyTorch、DarkNet等深度学习的研究环境。这里用的是CPU镜像，也可以采用GPU版本的 ###

# 显示本机已有的映像档
sudo docker images

# 查看目标端口占用 为下一步配置容器内的SSH作准备
lsof -i

# 新建docker container
# sudo nvidia-docker run -it -p 5592:5592 -p 5593:5593 -p 8022:22 --name="RemoteInterpreter_GPU" -v /host/data:/data -v /host/config:/config -v ~/workspace/idforhyit/remote_workspace:/workspace/idforhyit/remote_workspace ufoym/deepo:cpu /bin/bash
### PS. 一定要留一个端口映射到容器22端口，因为SFTP默认使用22端口） ###

# 守护态执行/后台运行
sudo docker run -d -it -p 5592:5592 -p 5593:5593 -p 8022:22 --name="RemoteInterpreter_GPU_d" -v /host/data:/data -v /host/config:/config -v ~/workspace/idforhyit/remote_workspace:/workspace/idforhyit/remote_workspace ufoym/deepo:cpu /bin/bash
# 获取容器的输出信息
docker container logs RemoteInterpreter_GPU_d
# 进入容器 推荐用docker exec方式
docker exec -it RemoteInterpreter_GPU_d bash

# 查看已经安装包的命令以及版本
```

```
pip freeze
# 显示本机已有的容器（包括停止运行的）
docker ps -a
# 停用并删除全部容器:
docker stop $(docker ps -q) & docker rm $(docker ps -aq)

# 配置SSH服务
apt update
apt install -y openssh-server
mkdir /var/run/sshd
echo "user:password" | chpasswd # [用户名:密码] 按需设置并牢记
sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config
sed 's@session\s*required\s*pam_loginuid.so@session optional pam_loginuid.so@g' -i
/etc/pam.d/sshd
echo "export VISIBLE=now" >> /etc/profile
# 修改/etc/ssh/sshd-config以使可远程密码登录docker的root账户
### 属性值
### PermitRootLogin yes
### PasswordAuthentication yes
service ssh restart # 重启SSH激活配置
exit    #退出容器
sudo docker port RemoteInterpreter_GPU_d 22 # 0.0.0.0:8022 正确配置后的输出
ssh IDforHYIT@192.168.68.131 -p 8022    # 过程中确认的密码是你前面配置SSH设置的

```

```
# Docker换源

# 复制源文件备份，以防万一
cp /etc/apt/sources.list /etc/apt/sources.list.bak
# 查看新版本信息
lsb_release -c
### 如果不是bionic 就将下列地址中的版本换为对应的 ###
```
Ubuntu 12.04 (LTS)代号为precise, Ubuntu 14.04 (LTS)代号为trusty,
Ubuntu 15.04 代号为vivid, Ubuntu 15.10 代号为wily, Ubuntu 16.04 (LTS)代号为xenial
```
# 编辑源列表文件
vim /etc/apt/sources.list
# 将原有的内容注释掉，添加以下内容（或者你把里面内容修改成下面的就可以，但是不能有除了以下内容的有效内
容）
deb http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse

deb-src http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe
multiverse

deb-src http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe
multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe
multiverse

```

```
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse

deb-src http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse

deb-src http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse
```

详细步骤记录

> 全文纯手码

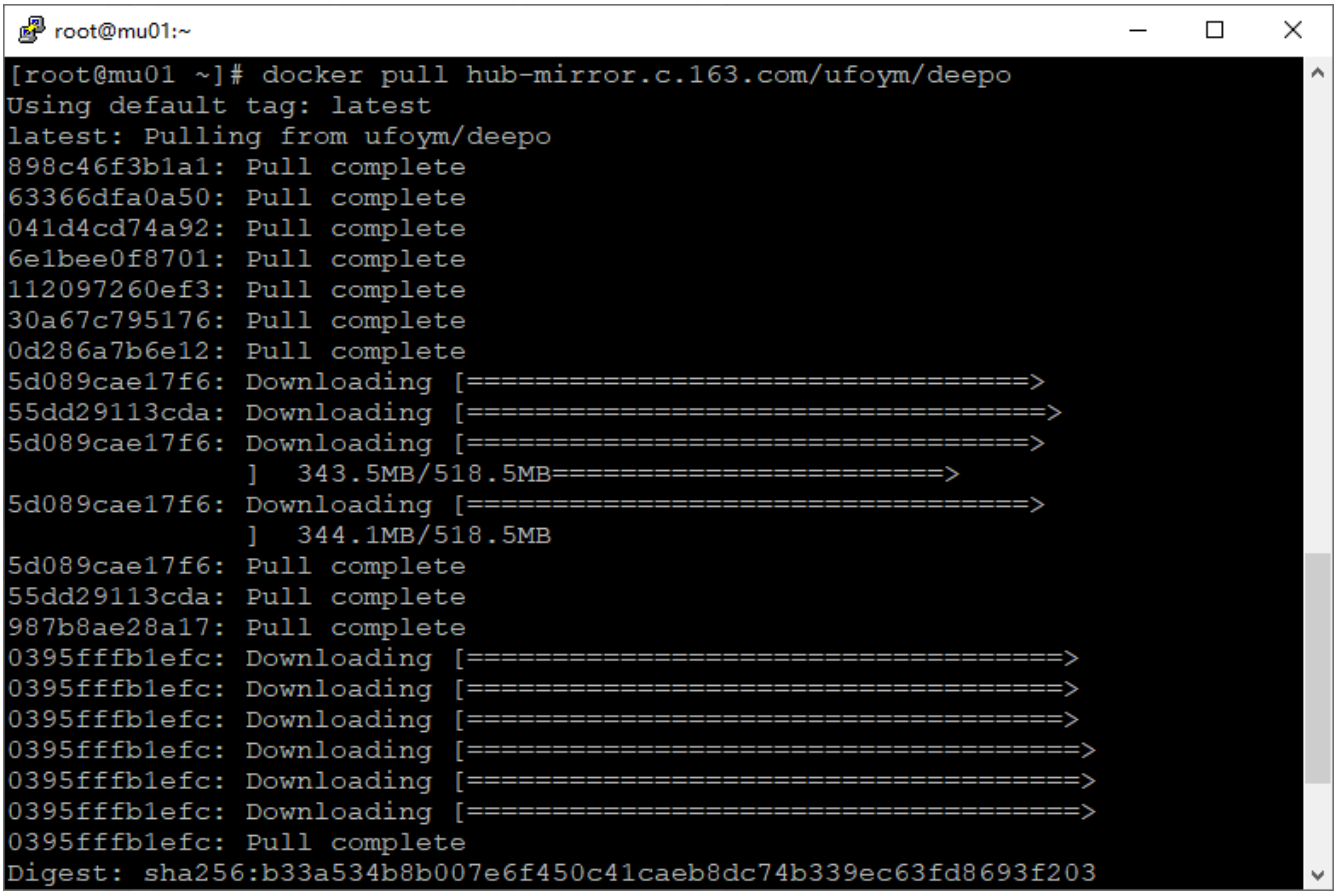
前期准备

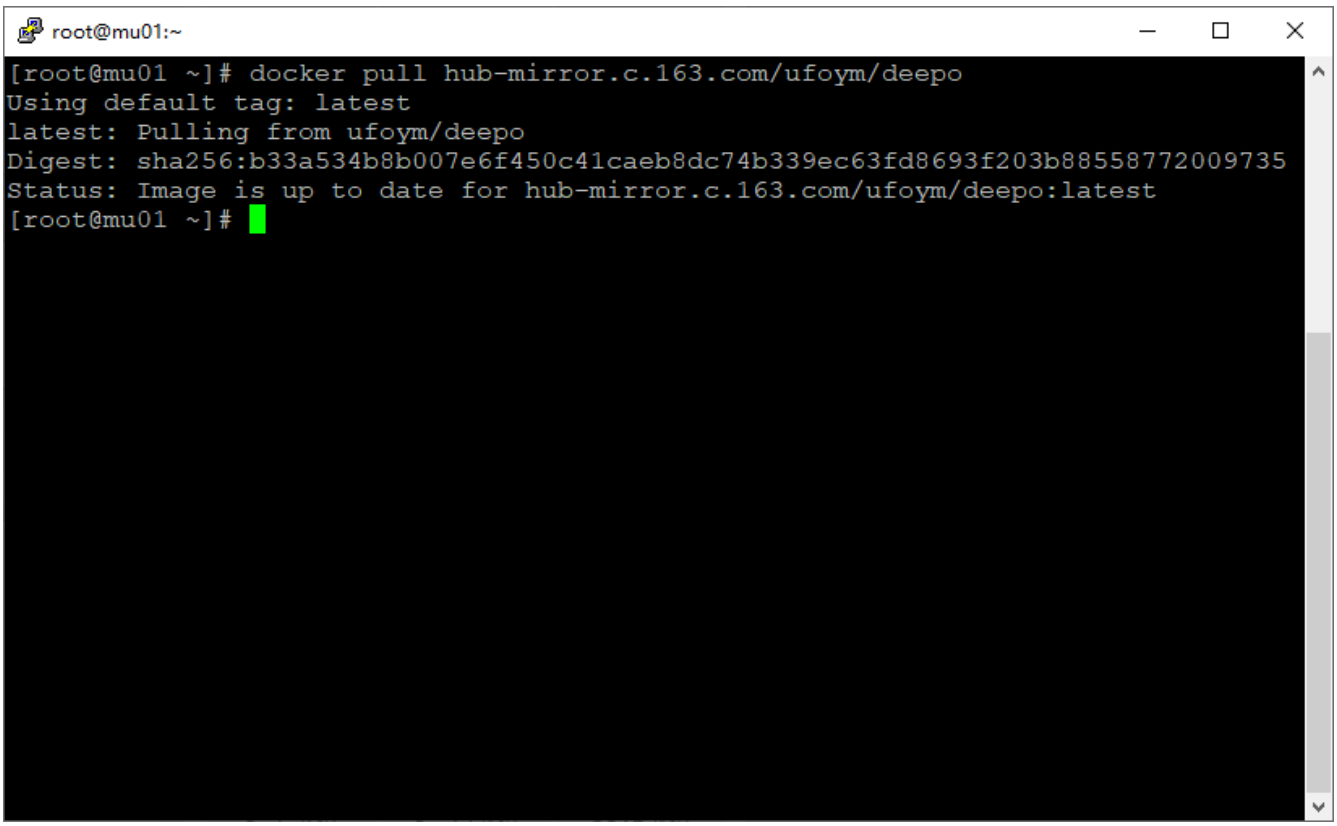
以管理员权限运行

```
su root
[password]
```

从DockerHub仓库下载一个深度学习作业系统的映像档

```
docker pull hub-mirror.c.163.com/ufoym/deepo
# 采用国内网易163镜像源 加快Pull速度
```

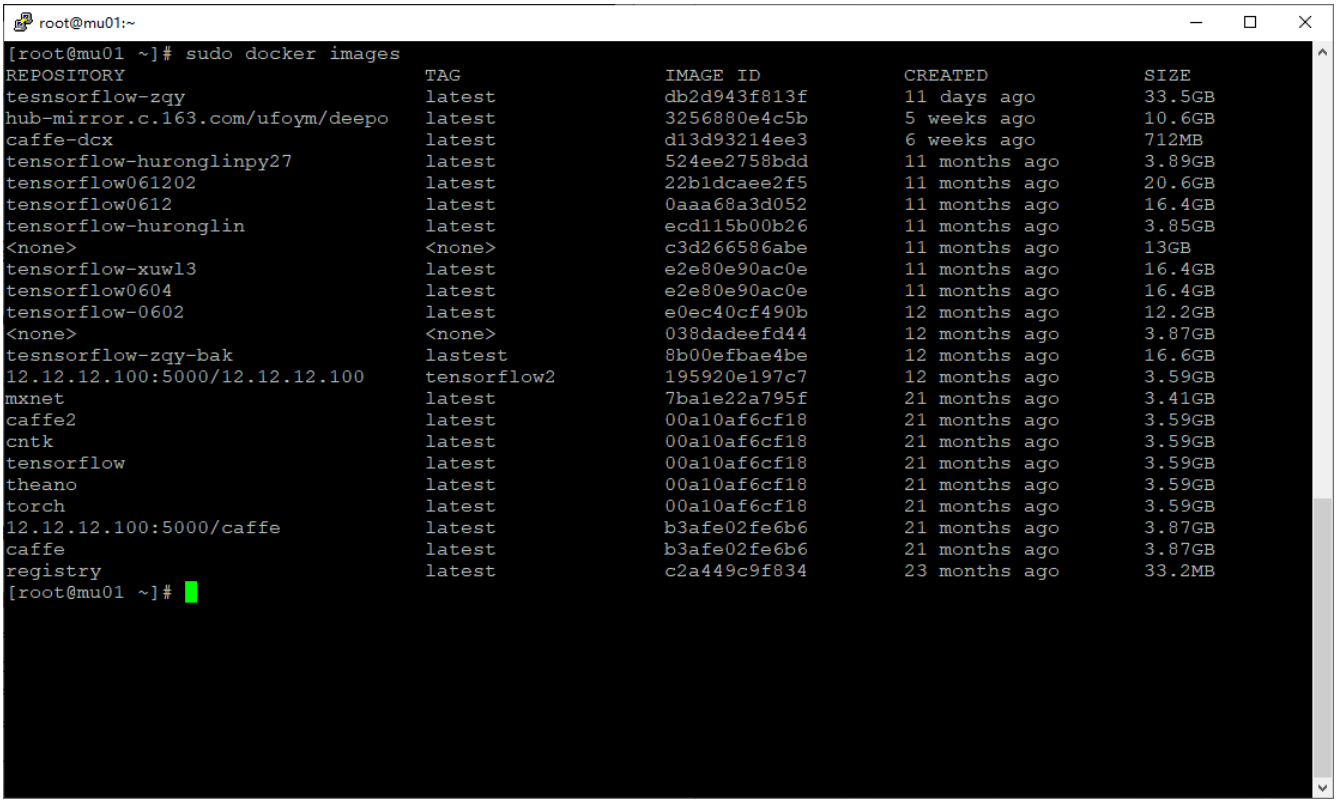




Note: 这里采用`deepo`，它几乎支持所有的深度学习框架，可以帮助俺们快速搭建一个包含`TensorFlow`、`PyTorch`、`DarkNet`等深度学习的研究环境。这里用的是GPU版本的Docker Image

显示本机已有的映像档

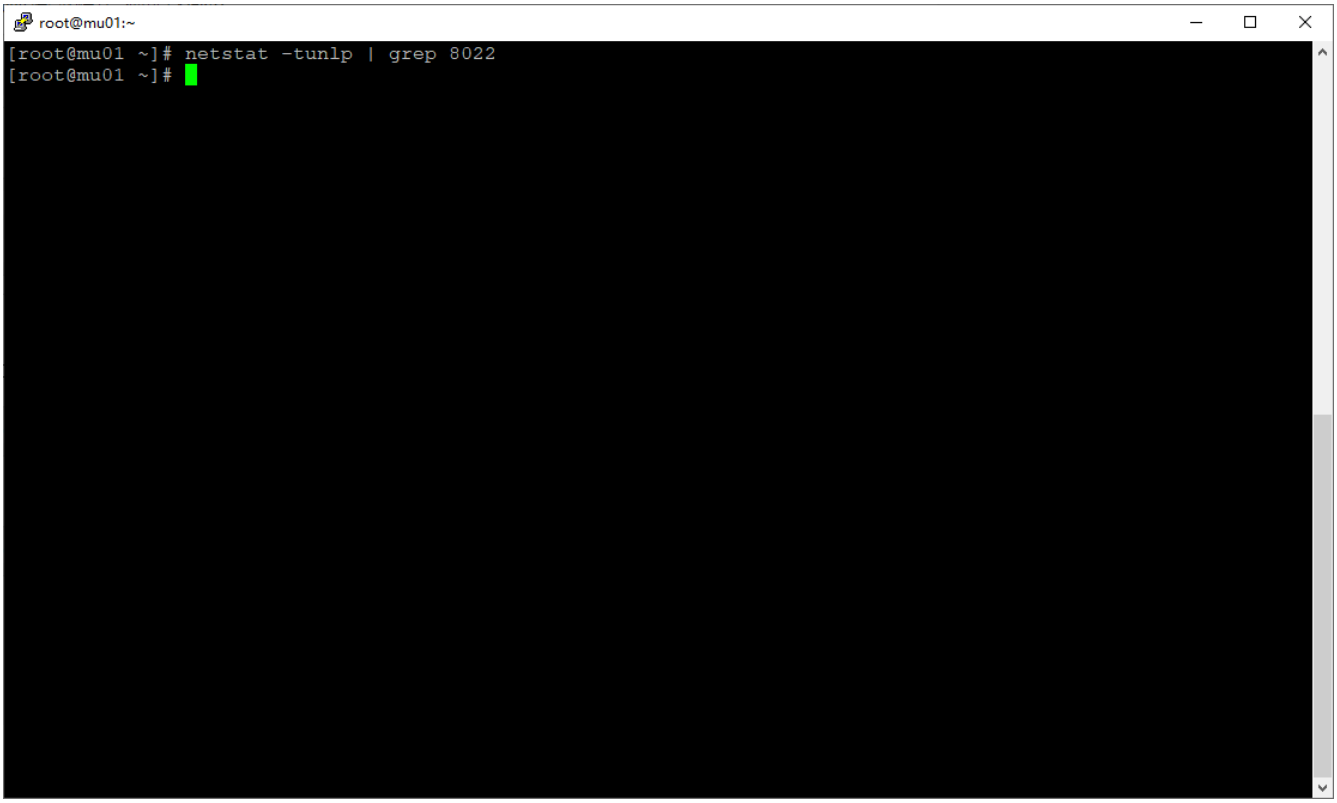
```
sudo docker images
```



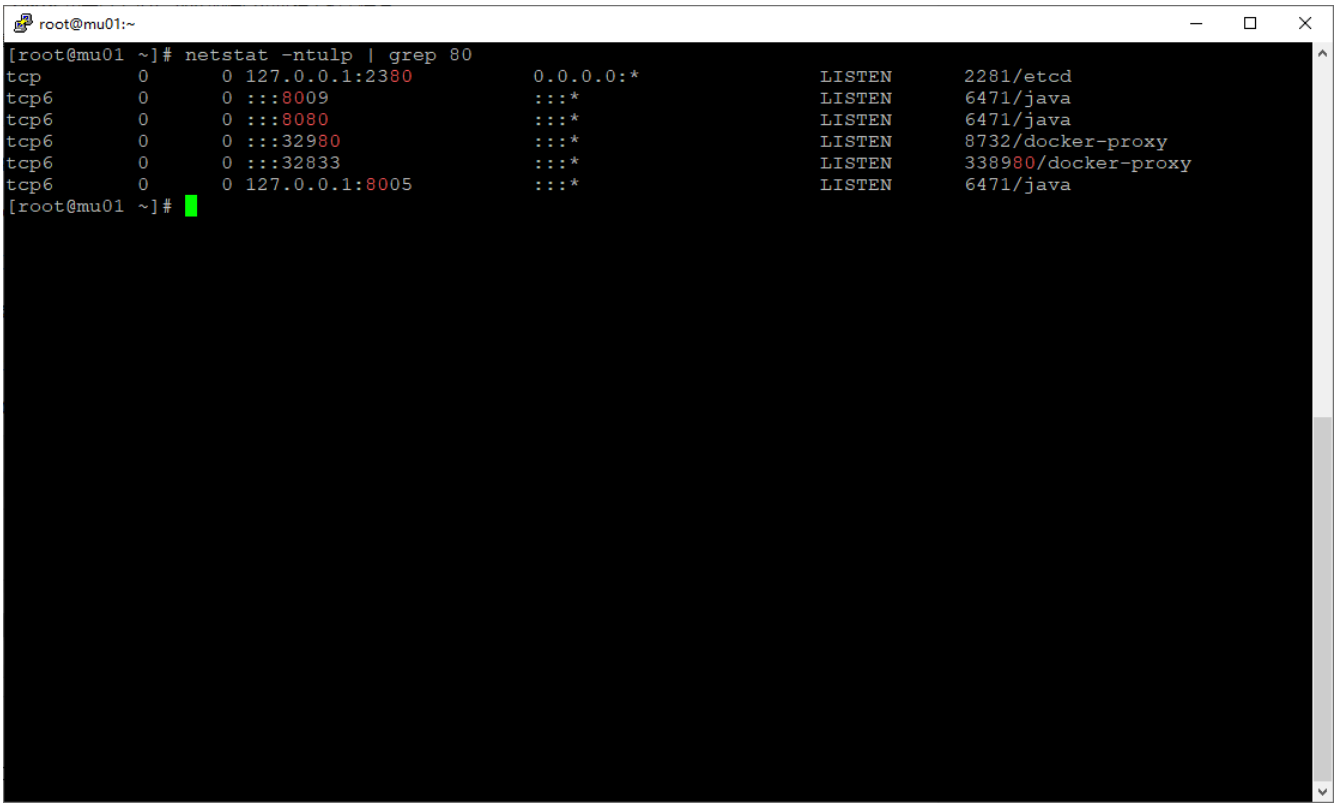
查看目标端口占用 为下一步配置容器内的SSH作准备

```
netstat -tunlp | grep 8022  # 8022是拟启用端口，用来转发docker容器SFTP的22端口
```

- Linux系统中，没返回信息就是最好的信息，2333



- 端口占用返回界面，以查看80端口使用情况为例，返回建立连接的程序及监听状态



开始配置

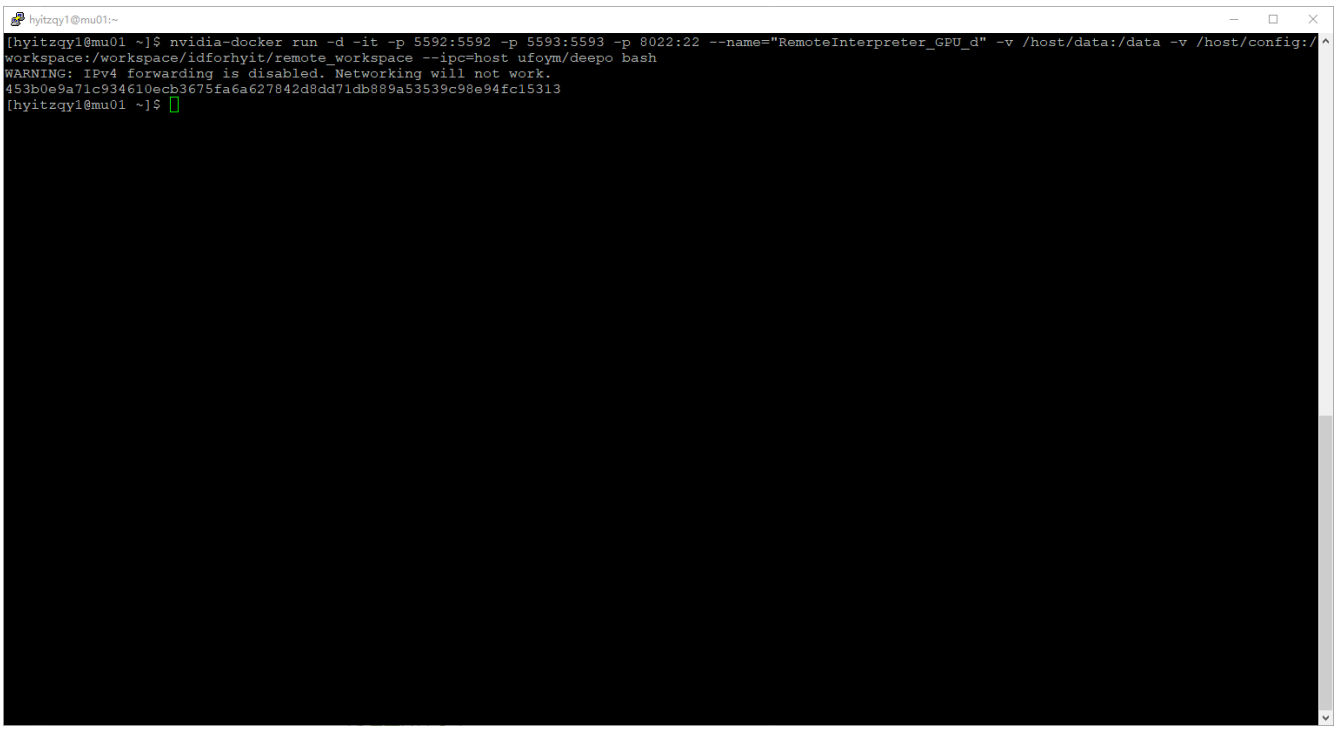
> 命令参考：https://hub.docker.com/r/ufoym/deepo#GPU
>
> https://www.runoob.com/docker/docker-command-manual.html

新建docker container

```
### PS. 一定要留一个端口映射到容器22端口，因为SFTP默认使用22端口） ###
```

守护态执行/后台运行方式新建容器

```
sudo nvidia-docker run -d -it -p 5592:5592 -p 5593:5593 -p 8022:22 --name="RemoteInterpreter_GPU_d" -v /host/data:/data -v /host/config:/config -v ~/MaSiwei/workspace/idforhyit/remote_workspace:/workspace/idforhyit/remote_workspace --ipc=host ufoym/deepo bash
# or
nvidia-docker run -d -it -p 5592:5592 -p 5593:5593 -p 8022:22 --name="RemoteInterpreter_GPU_d" -v /host/data:/data -v /host/config:/config -v ~/workspace/idforhyit/remote_workspace:/workspace/idforhyit/remote_workspace ufoym/deepo bash
```



> 新建的docker容器，端口映射，容器名，镜像和路径映射这些换成你自己的就行，但是一定要留一个端口映射到容器22端口，因为SFTP默认使用22端口。

Docker Tips

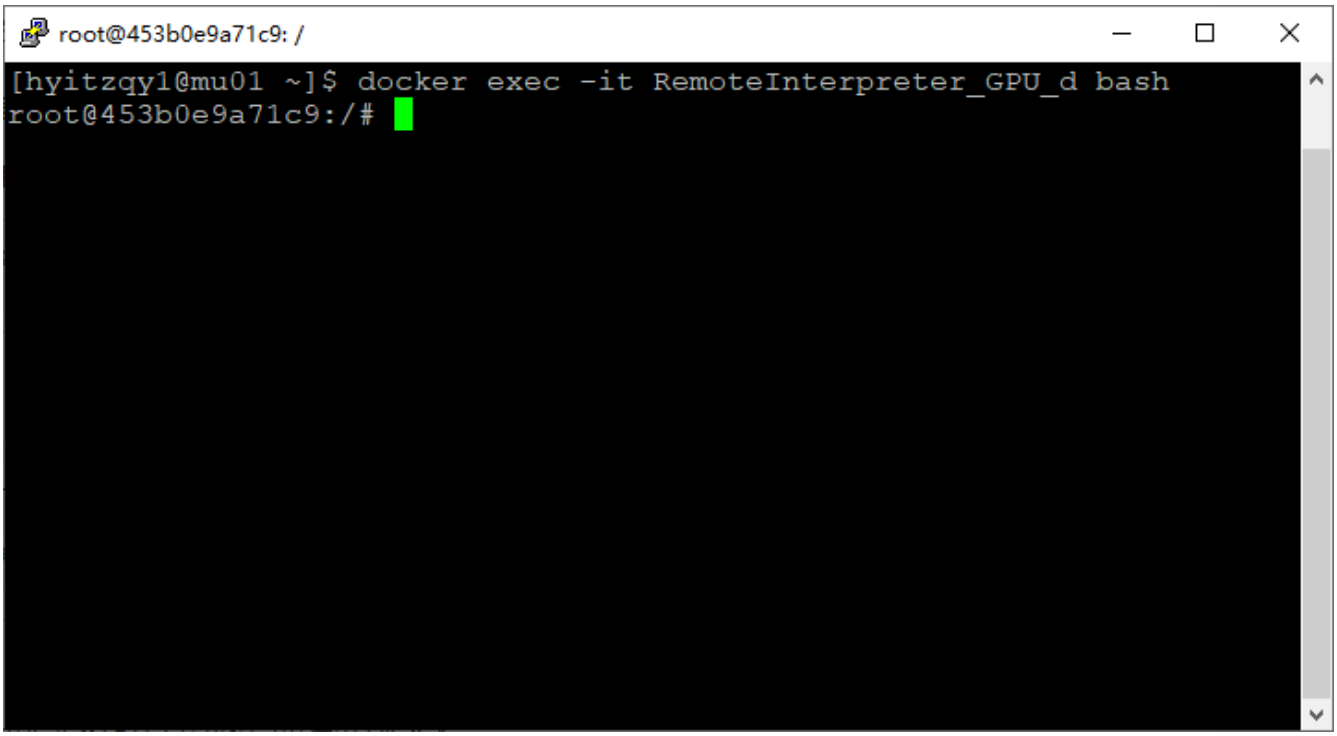
- 守护态执行新建后获取容器的输出信息

```
docker container logs RemoteInterpreter_GPU_d
```

- **进入容器** （非启动）

  > 推荐用docker exec方式

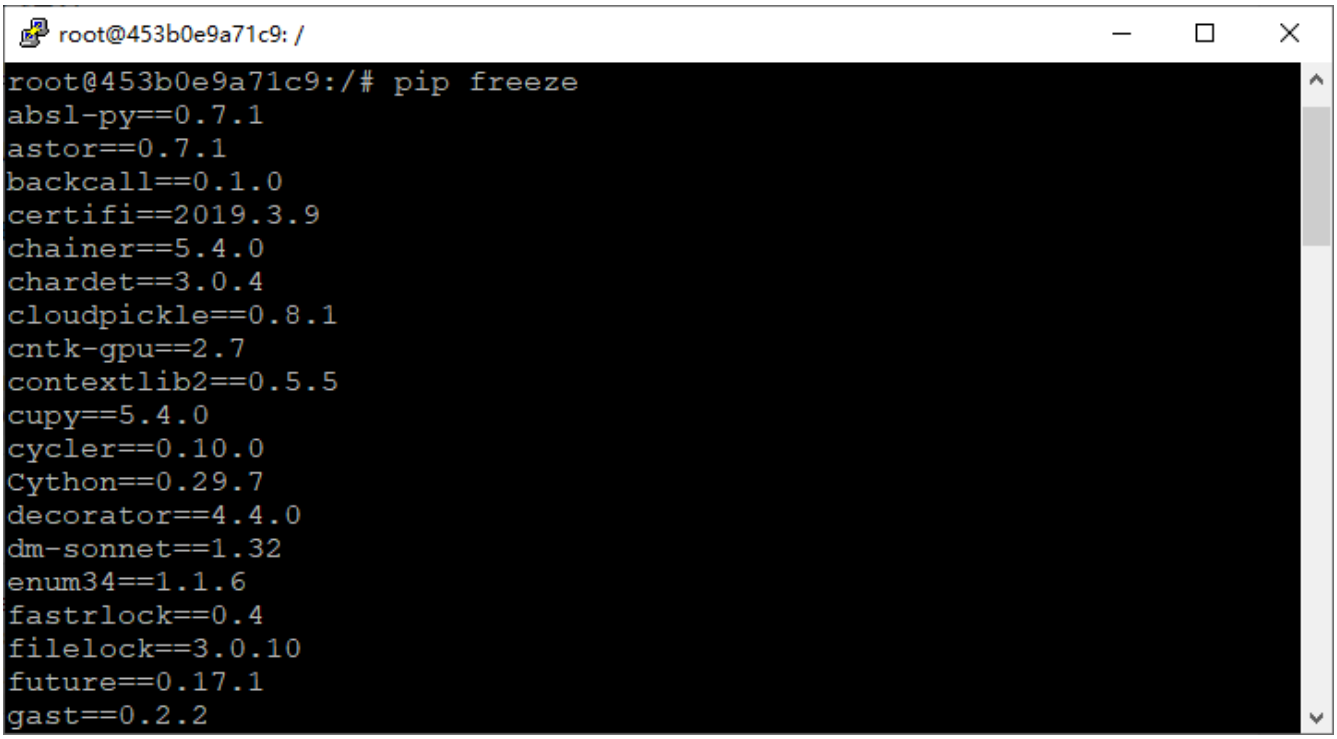
```
docker exec -it RemoteInterpreter_GPU_d bash
```



- 查看已经安装包的命令以及版本

  > 进入容器后使用，查看一下`deepo`的已有环境

```
pip freeze
```



> 下面为`DockerHub`中`ufoym/deepo`的已有环境内容，可以看到默认已经带了比较全面的机器学习的开发框架（`TensorFlow`，`PyTorch`，`Keras`，`scikit-learn`），这个容器对于深度学习科研实验来说，已经达到了开箱即用的程度，不然自己配置会很麻烦。另外，因为`DockerHub`开源，镜像安全性也是有一定保障的。
>
> Docker，懒癌患者的福音。

```
root@453b0e9a71c9:/# pip freeze
absl-py==0.7.1
astor==0.7.1
backcall==0.1.0
certifi==2019.3.9
chainer==5.4.0
chardet==3.0.4
cloudpickle==0.8.1
cntk-gpu==2.7
contextlib2==0.5.5
cupy==5.4.0
cycler==0.10.0
Cython==0.29.7
decorator==4.4.0
dm-sonnet==1.32
enum34==1.1.6
fastrlock==0.4
filelock==3.0.10
future==0.17.1
gast==0.2.2
google-pasta==0.1.5
graphviz==0.10.1
grpcio==1.20.0
h5py==2.9.0
idna==2.8
imageio==2.5.0
ipython==7.4.0
ipython-genutils==0.2.0
jedi==0.13.3
Keras==2.2.4
Keras-Applications==1.0.7
Keras-Preprocessing==1.0.9
kiwisolver==1.0.1
Lasagne==0.2.dev1
leveldb==0.194
Mako==1.0.9
Markdown==3.1
MarkupSafe==1.1.1
matplotlib==3.0.3
mxnet-cu100==1.4.0.post0
networkx==2.3
nose==1.3.7
numpy==1.16.3
onnx==1.5.0
pandas==0.24.2
parso==0.4.0
pexpect==4.7.0
```

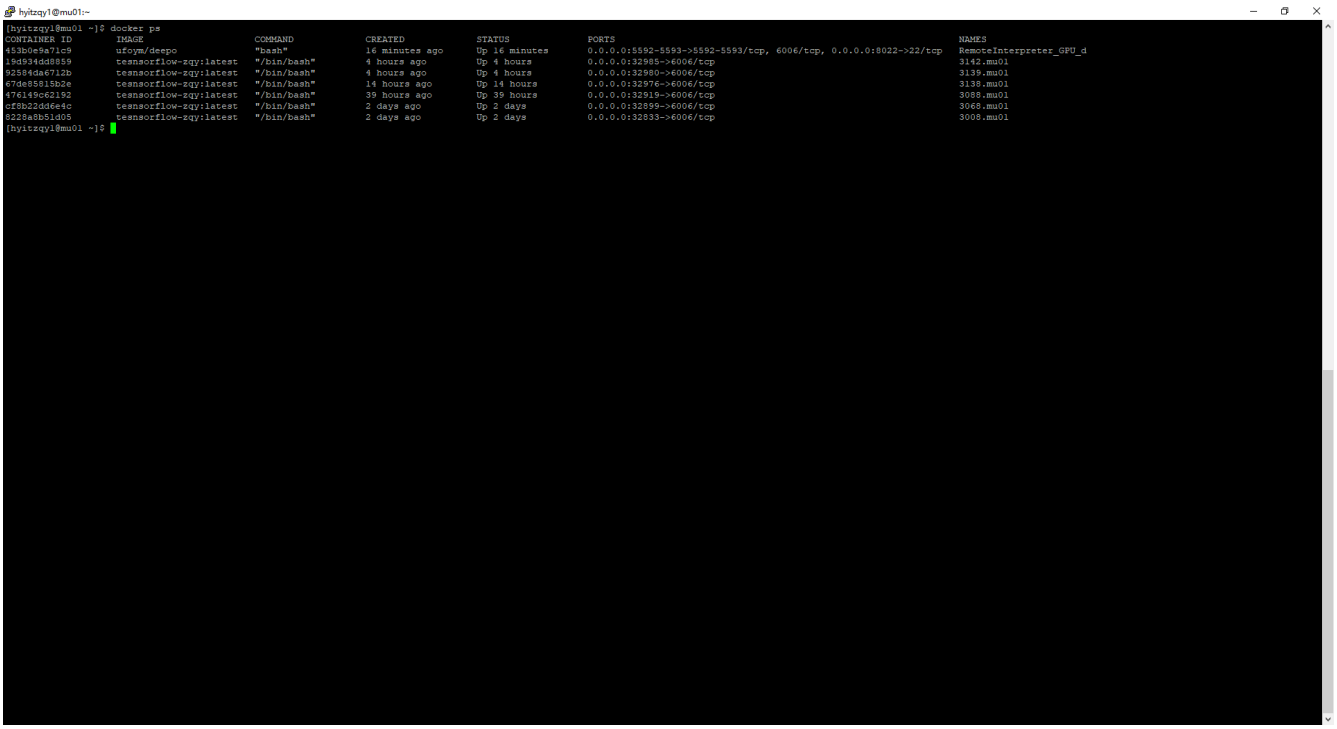
```
pickleshare==0.7.5
Pillow==6.0.0
prompt-toolkit==2.0.9
protobuf==3.7.1
ptyprocess==0.6.0
Pygments==2.3.1
pygobject==3.26.1
pygpu==0.7.6
pyparsing==2.4.0
python-apt==1.6.3+ubuntu1
python-dateutil==2.8.0
python-distutils-extra==2.39
python-gflags==3.1.2
pytz==2019.1
PyWavelets==1.0.3
PyYAML==5.1
requests==2.21.0
scikit-image==0.15.0
scikit-learn==0.20.3
scipy==1.2.1
semantic-version==2.6.0
six==1.12.0
tb-nightly==1.14.0a20190424
tensorflow-estimator-2.0-preview==1.14.0.dev2019042300
tensorflow-probability==0.6.0
termcolor==1.1.0
tf-nightly-gpu-2.0-preview==2.0.0.dev20190424
Theano==1.0.4+unknown
torch-nightly==1.1.0.dev20190424
torchvision-nightly==0.2.3
tqdm==4.31.1
traitlets==4.3.2
typing==3.6.6
typing-extensions==3.7.2
urllib3==1.24.2
wcwidth==0.1.7
Werkzeug==0.15.2
wrapt==1.11.1
```

- **退出容器**（非终止）

```
exit
# or
Ctrl+D
```

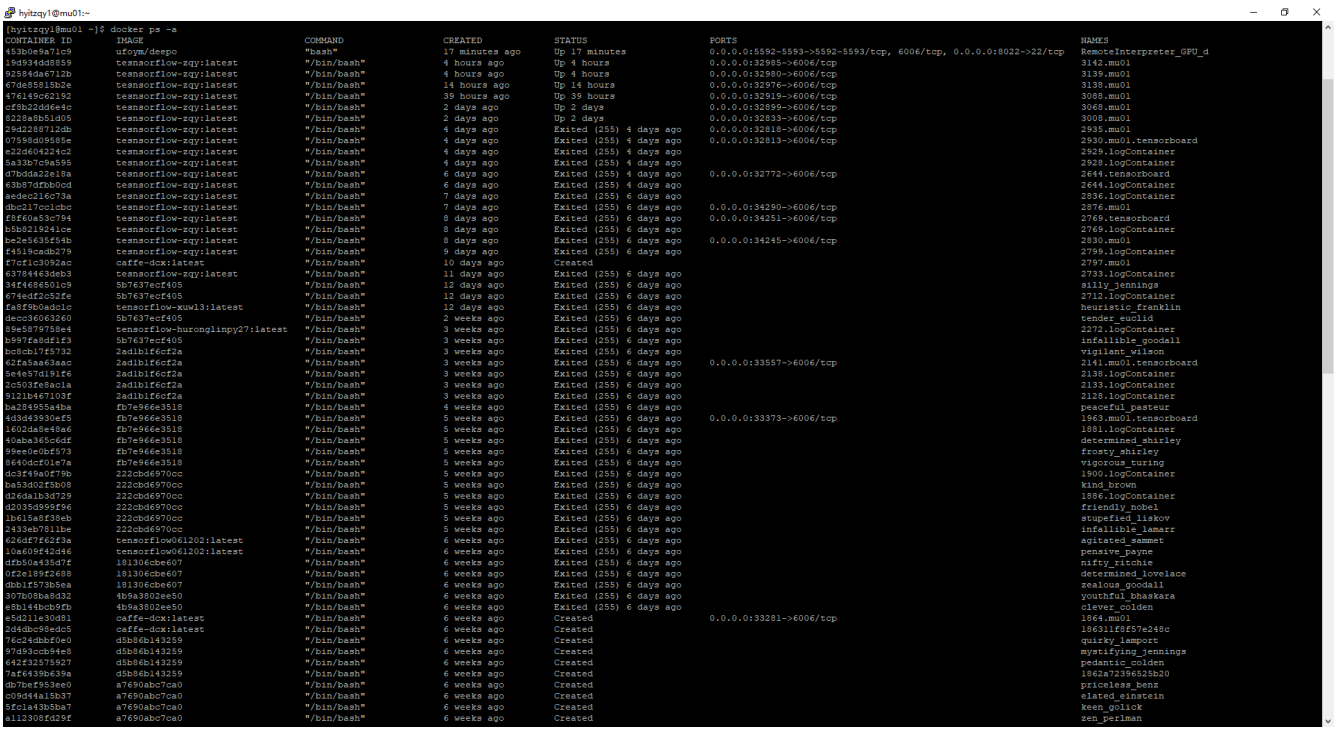
- **显示本机已有的容器（正在运行的）**

```
docker ps
```



- 显示本机已有的容器（包括停止运行的）

```
docker ps -a
```



- **终止容器**

> 使用`docker stop $CONTAINER_ID`来终止一个运行中的容器。并且可以使用`docker ps -a`来看终止状态的容器。
>
> 终止状态的容器，可以使用`docker start`来重新启动。

```
docker stop [Name或者ID]
```

- 删除容器（Container）

```
docker rm [Name或者ID]

#
docker stop RemoteInterpreter_GPU_d
docker rm RemoteInterpreter_GPU_d
```

- 删除镜像（Image）

  > 一般不删除镜像，毕竟十几个G，重下也麻烦。
  >
  > **删除前务必保证该镜像的所有实例容器（Container）全部`stop`停止和删除**，否者会很麻烦。
  >
  > 需要注意删除镜像和容器的命令不一样。`docker rmi ID`,其中 容器(`rm`) 和 镜像(`rmi`)。

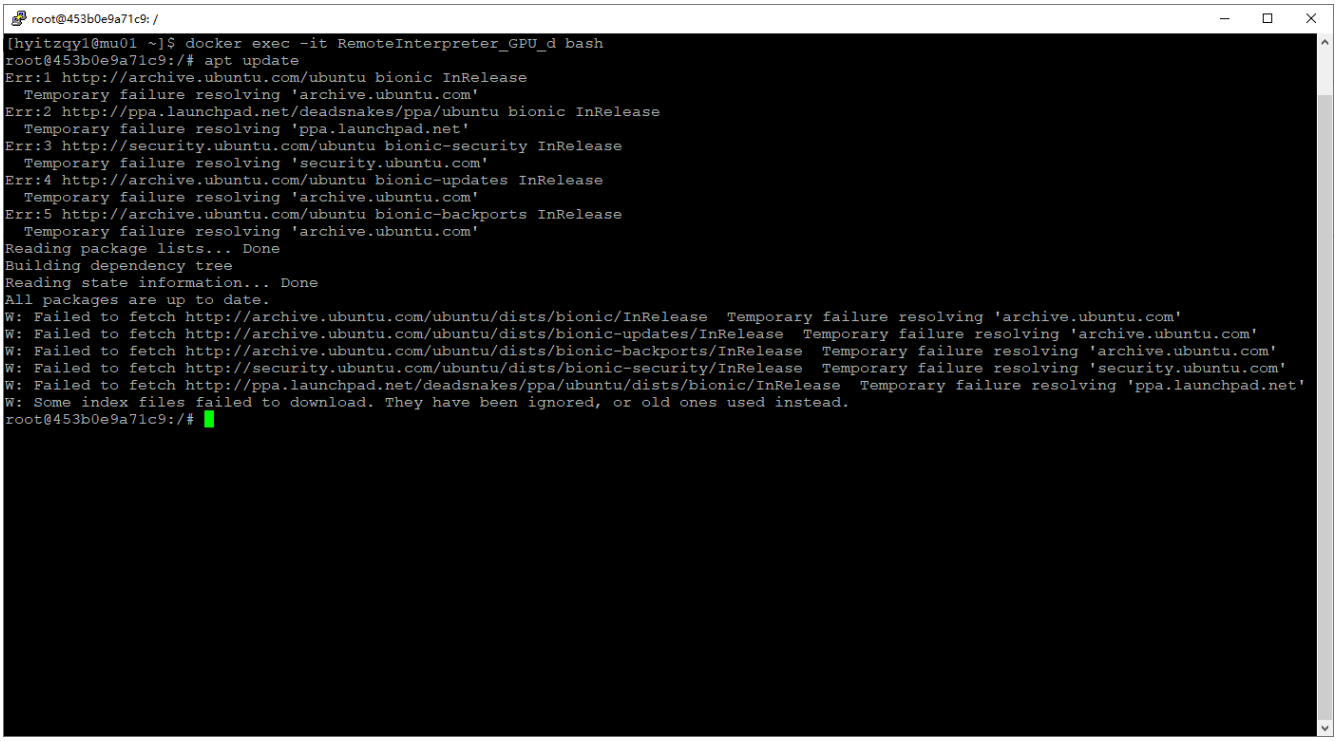
  ```
  # 查询镜像，拿到欲删除的`IMAGE ID`
  docker images
  # 查询容器，拿到镜像实例容器的`CONTATNER ID`
  docker ps -a
  # 先删除容器，如果容器正在运行，先用`docker stop`停止
  docker rm [CONTATNER ID]
  docker ps -a    # 查看容器是否删除成功
  # 再删除镜像
  docker rmi [IMAGE ID]
  ```

配置SSH服务

> **配置部分命令在目标容器内进行**

先对目标容器环境进行更新

```
docker exec -it RemoteInterpreter_GPU_d bash    # 如已在容器内请忽略本条
apt update
```



> 获更新失败解决参考

Docker换源三部曲

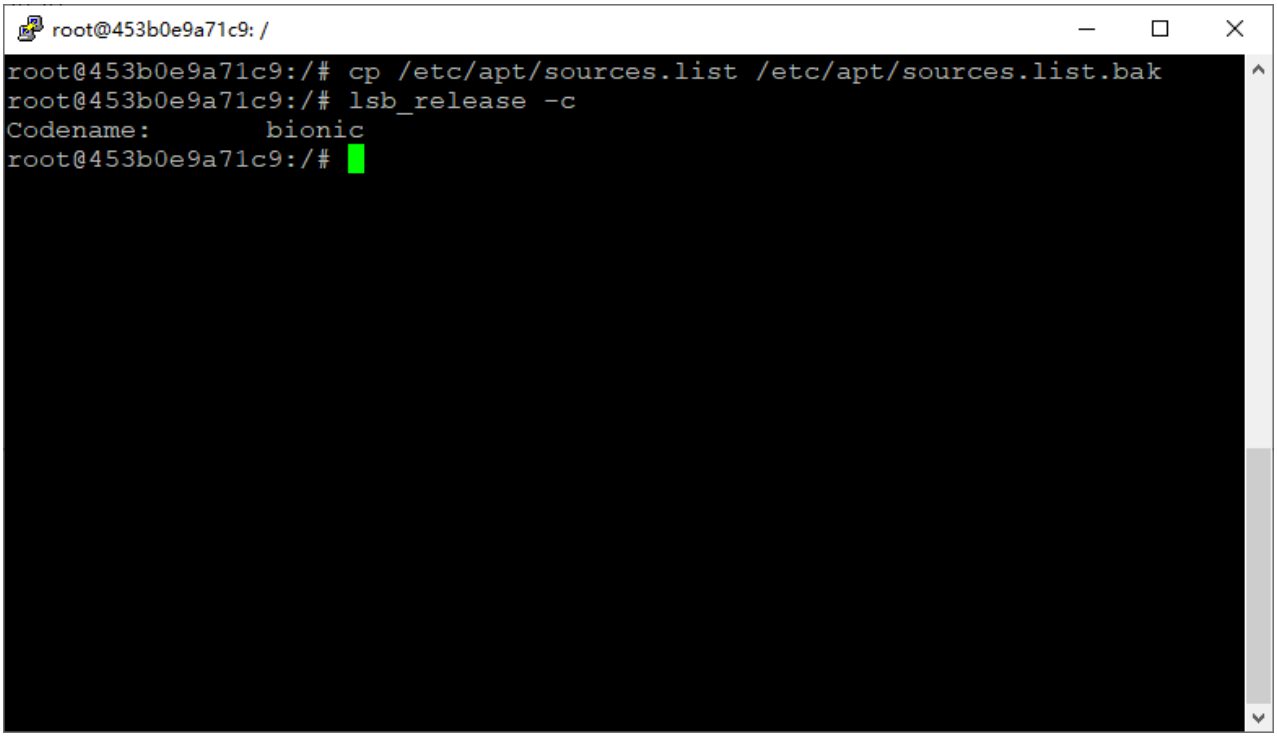
- 官方源国内不好使，那咱就换源，盘他~

1. 复制源文件备份，以防万一

```
cp /etc/apt/sources.list /etc/apt/sources.list.bak
```

2. 查看新版本信息

```
lsb_release -c
```



- **如果不是`bionic`就将之后镜像源地址中的版本代号部分换为对应的版本代号**
  - Ubuntu 12.04 (LTS)代号为`precise`
  - Ubuntu 14.04 (LTS)代号为`trusty`
  - Ubuntu 15.04 代号为`vivid`
  - Ubuntu 15.10 代号为`wily`
  - Ubuntu 16.04 (LTS)代号为`xenial`
  - Ubuntu 18.04 (LTS)代号为`bionic`

3. 编辑源列表文件

```
vim /etc/apt/sources.list
```

> 将原有的有效行前面添加`#`注释掉，添加以下内容（或者你把里面内容修改成下面的就可以，但是不能有除了以下内容的有效内容）

- Vim使用方法，防对命令行操作不熟悉的童鞋。笔记本键盘童鞋，如果操作失败，想想`Fn`键的作用。
  - `Insert`，进行`插入`方式的编辑文本文件
  - `Shift`+`Insert`，进行`粘贴`粘贴板中的文本内容
    - 如果和博主使用的Dock镜像不一样，注意将文本中`bionic`替换为对应版本的代号
  - `ESC` ==> `Shift`+`;`（即`:`） ==> `wq`，`保存修改并退出`编辑

```
deb http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse
```

> *开启行号*：`:` + `set number`

更多vi/vim命令详见：https://www.runoob.com/linux/linux-vim.html

```
# 国内源
deb http://cn.archive.ubuntu.com/ubuntu bionic main multiverse restricted universe
deb http://cn.archive.ubuntu.com/ubuntu bionic-updates main multiverse restricted universe
deb http://cn.archive.ubuntu.com/ubuntu bionic-security main multiverse restricted universe
deb http://cn.archive.ubuntu.com/ubuntu bionic-proposed main multiverse restricted universe

```

```
#添加阿里源
deb http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse

##中科大源
deb https://mirrors.ustc.edu.cn/ubuntu/ bionic main restricted universe multiverse
deb-src https://mirrors.ustc.edu.cn/ubuntu/ bionic main restricted universe multiverse
deb https://mirrors.ustc.edu.cn/ubuntu/ bionic-updates main restricted universe multiverse
deb-src https://mirrors.ustc.edu.cn/ubuntu/ bionic-updates main restricted universe multiverse
deb https://mirrors.ustc.edu.cn/ubuntu/ bionic-backports main restricted universe multiverse
deb-src https://mirrors.ustc.edu.cn/ubuntu/ bionic-backports main restricted universe multiverse
deb https://mirrors.ustc.edu.cn/ubuntu/ bionic-security main restricted universe multiverse
deb-src https://mirrors.ustc.edu.cn/ubuntu/ bionic-security main restricted universe multiverse
deb https://mirrors.ustc.edu.cn/ubuntu/ bionic-proposed main restricted universe multiverse
deb-src https://mirrors.ustc.edu.cn/ubuntu/ bionic-proposed main restricted universe multiverse

##163源
deb http://mirrors.163.com/ubuntu/ bionic main restricted universe multiverse
deb http://mirrors.163.com/ubuntu/ bionic-security main restricted universe multiverse
deb http://mirrors.163.com/ubuntu/ bionic-updates main restricted universe multiverse
deb http://mirrors.163.com/ubuntu/ bionic-proposed main restricted universe multiverse
deb http://mirrors.163.com/ubuntu/ bionic-backports main restricted universe multiverse
deb-src http://mirrors.163.com/ubuntu/ bionic main restricted universe multiverse
deb-src http://mirrors.163.com/ubuntu/ bionic-security main restricted universe multiverse
deb-src http://mirrors.163.com/ubuntu/ bionic-updates main restricted universe multiverse
```

```
deb-src http://mirrors.163.com/ubuntu/ bionic-proposed main restricted universe multiverse
deb-src http://mirrors.163.com/ubuntu/ bionic-backports main restricted universe multiverse

##清华源
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic main restricted universe multiverse
deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-updates main restricted universe multiverse
deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-updates main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-backports main restricted universe multiverse
deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-backports main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-security main restricted universe multiverse
deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-security main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-proposed main restricted universe multiverse
deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-proposed main restricted universe multiverse
```

在容器内安装`openssh-server`并配置用户名和密码

```
apt update
apt install -y openssh-server
mkdir /var/run/sshd
echo "user:password" | chpasswd # [用户名:密码] 按需设置并牢记
sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config
sed 's@session\s*required\s*pam_loginuid.so@session optional pam_loginuid.so@g' -i /etc/pam.d/sshd
echo "export VISIBLE=now" >> /etc/profile
```

- 修改`sshd-config`属性值

  > 修改`/etc/ssh/sshd-config`以使可远程密码登录docker的root账户

  ```
  vi /etc/ssh/sshd-config
  ```

NAME	KEY
`PermitRootLogin`	`yes`
`PasswordAuthentication`	`yes`

验证配置

```
service ssh restart # 重启SSH激活配置
exit    #退出容器
sudo docker port RemoteInterpreter_GPU_d 22 # 0.0.0.0:8022 正确配置后的输出
ssh IDforHYIT@172.20.1.100 -p 8022  # 过程中确认的密码是你前面配置SSH设置的
```

> 宿主机的IP地址可用`ifconfig`获得，按照博主几个虚拟机来瞎测试来总结的经验，一般是第一个以`en`开头的网卡就是本机对外IP地址。

本地Pycharm配置

> 社区版无远程解释器功能，**请用专业版。**激活教程请在本博内搜索`Jetbrain`。

因上一步服务器Docker内的SSH配置中因服务器安全设置等原因，禁用了Docker的端口转发，Docker内无法连接外网，故本步骤用之前虚拟机搭建的IP地址进行配置，除了IP地址、账号密码等因配置原因不一样外，其他无异。

配置过程

大致配置过程如下：

- 从启动页进入设置（**任何项目可用**）

`Pycharm启动页` ---> `Configure` ---> `Setting` ---> `Project Interpreter` ---> `小齿轮按钮` ---> `Add...` ---> `SSH Interpreter` ---> `New server configuration` ---> `Host` ---> `Port` ---> `Username` ---> `Next` ---> `Password` ---> `Next` ---> `Interpreter` ---> `Finish`

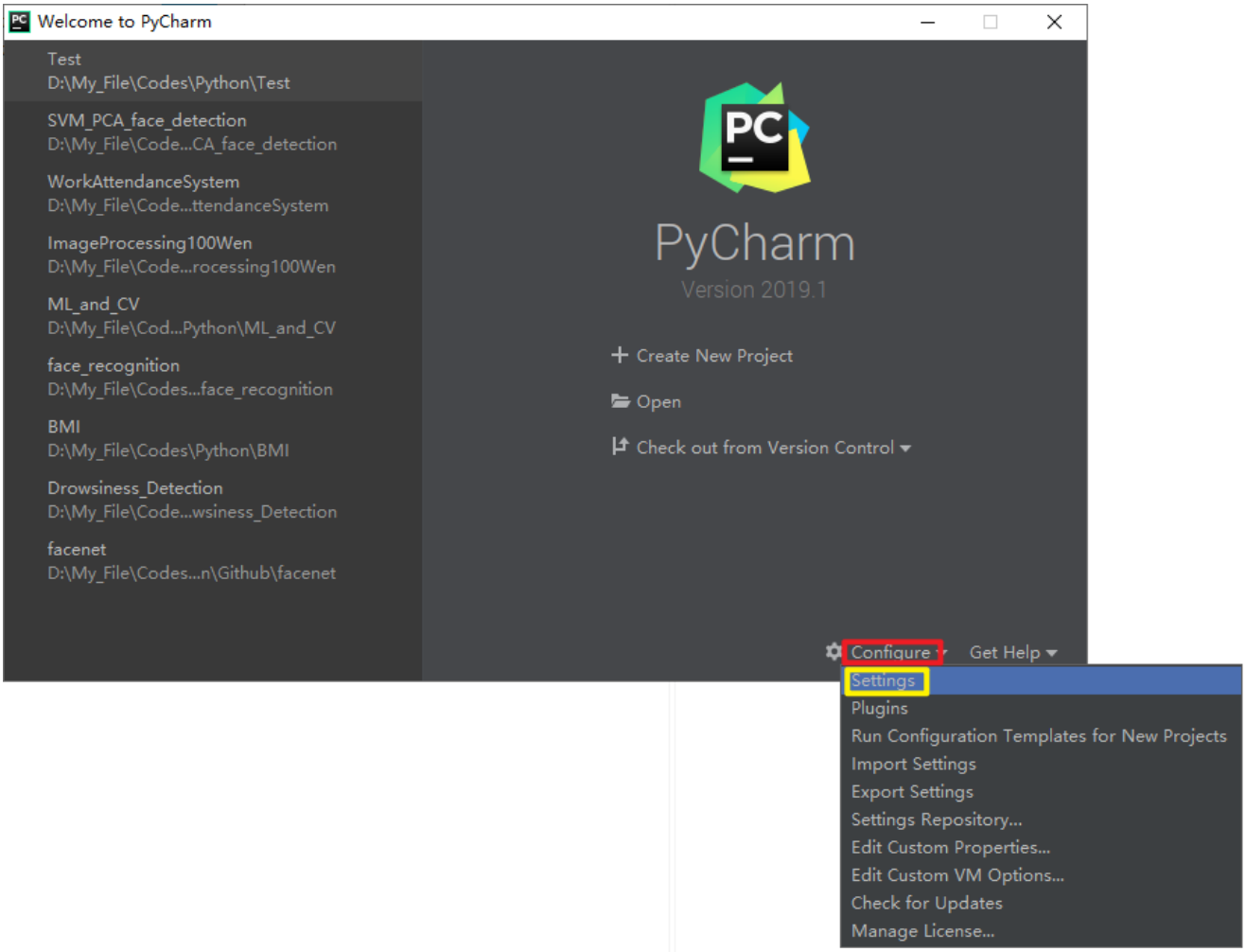
- 从项目页进入设置（**默认仅配置项目可见**）

`菜单栏` ---> `File` ---> `Settings...` ---> `Project: (项目名)` ---> `Project Interpreter` ---> `小齿轮按钮` ---> `Add...` ---> `SSH Interpreter` ---> `New server configuration` ---> `Host` ---> `Port` ---> `Username` ---> `Next` ---> `Password` ---> `Next` ---> `Interpreter` ---> `Finish`

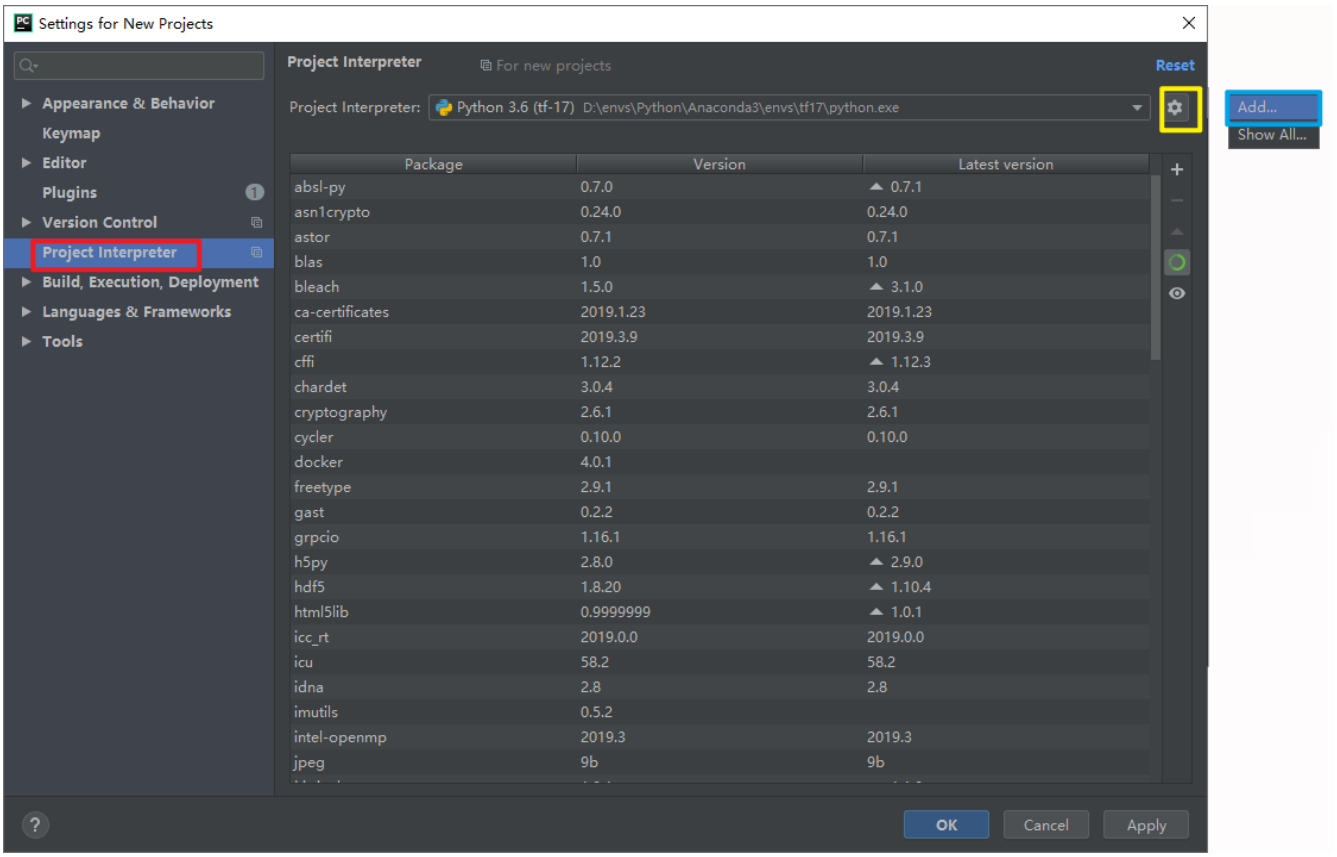
具体过程截图说明：

从启动页进入设置（任何项目可用）

1. `Pycharm启动页` ---> `Configure` ---> `Setting`

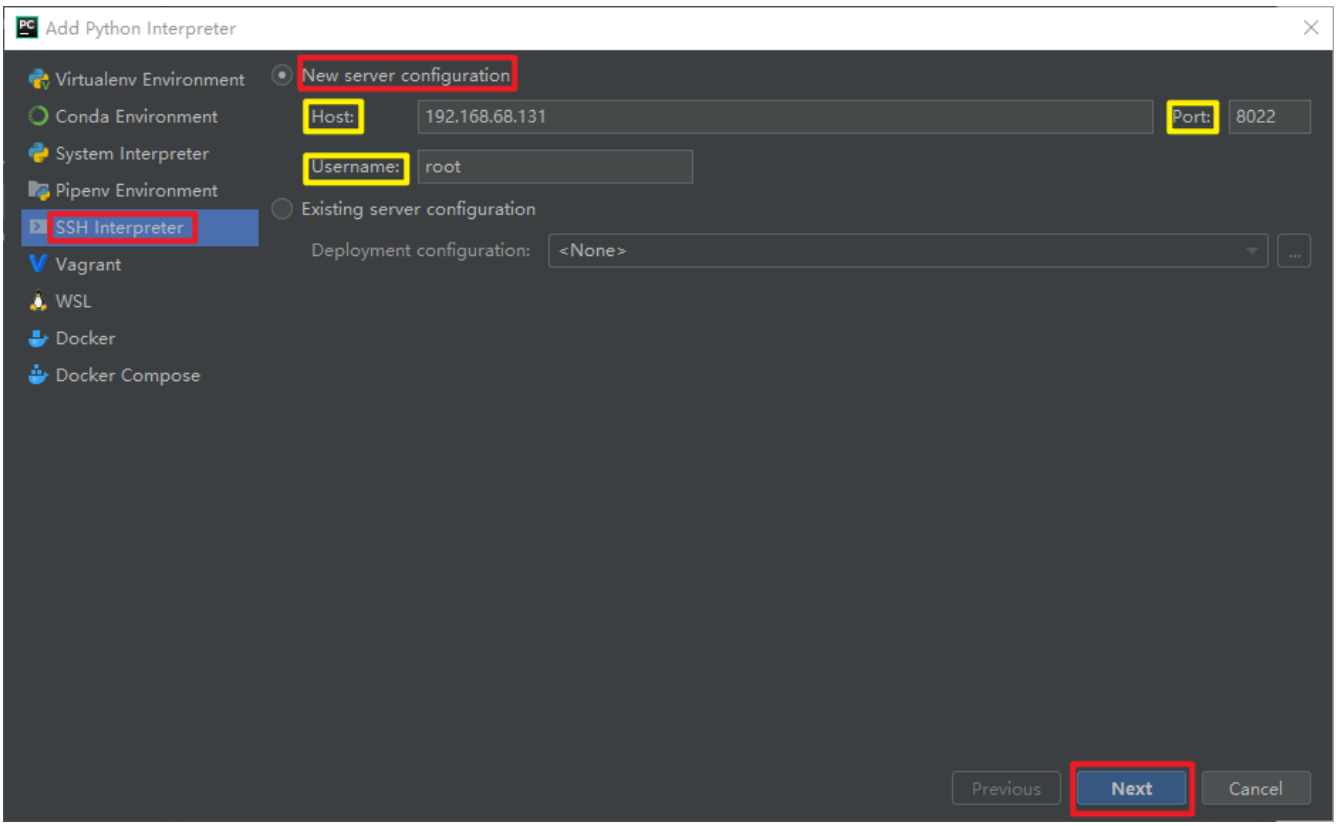


2. `Project Interpreter` ---> 小齿轮按钮 ---> `Add...`



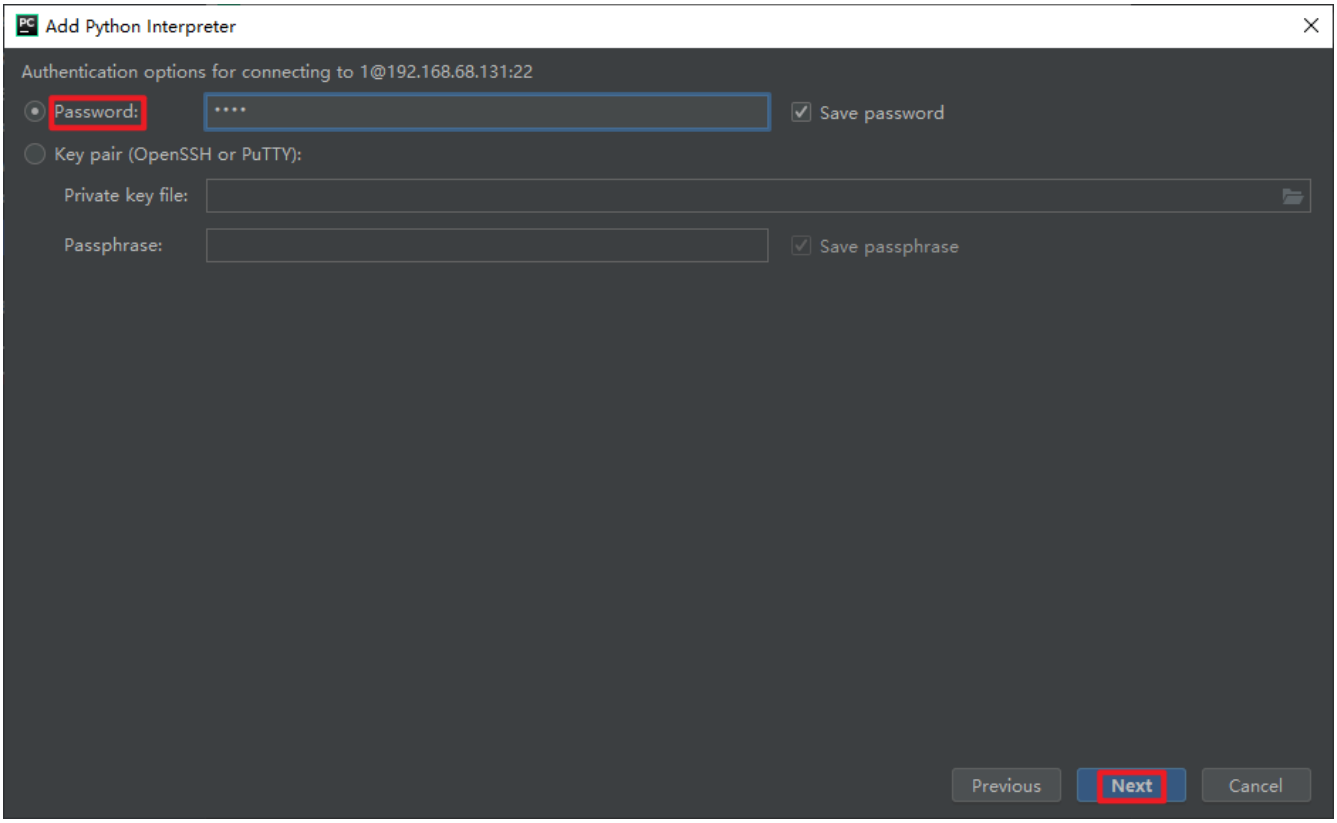
3. `SSH Interpreter` ---> `New server configuration` ---> 填入上一步SSH配置中获得的Host地址 ---> `Port`输入之前新建Docker时设置的22端口的转发端口 ---> `Username` ---> `Next`

> `Host`为宿主机的IP地址，可用`ifconfig`命令获得，之前说过了
>
> `Port`转发端口为之前`新建Docker`时设置的宿主机端口22的转发端口
>
> > `docker run … -p xxx:22 …`命令中的`xxx`即是配置的转发端口
>
> 本步及下一步的账号密码请填入你在SSH配置中用`echo "user:password" | chpasswd`命令配置的`[用户名:密码]`

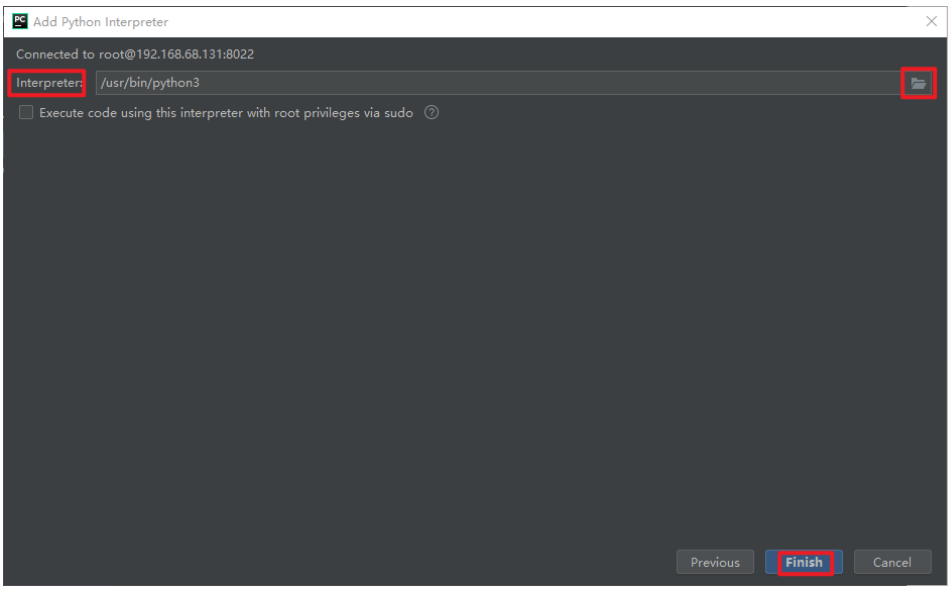


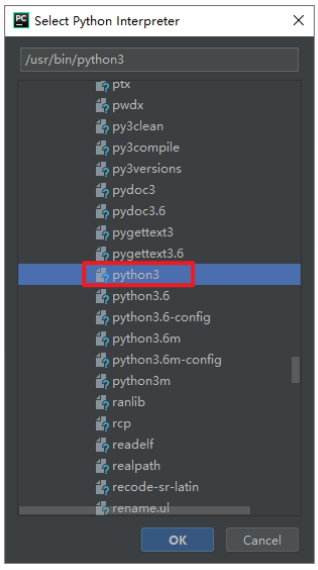
4. `Password`---> `Next`

   建议`Save password`复选框选中



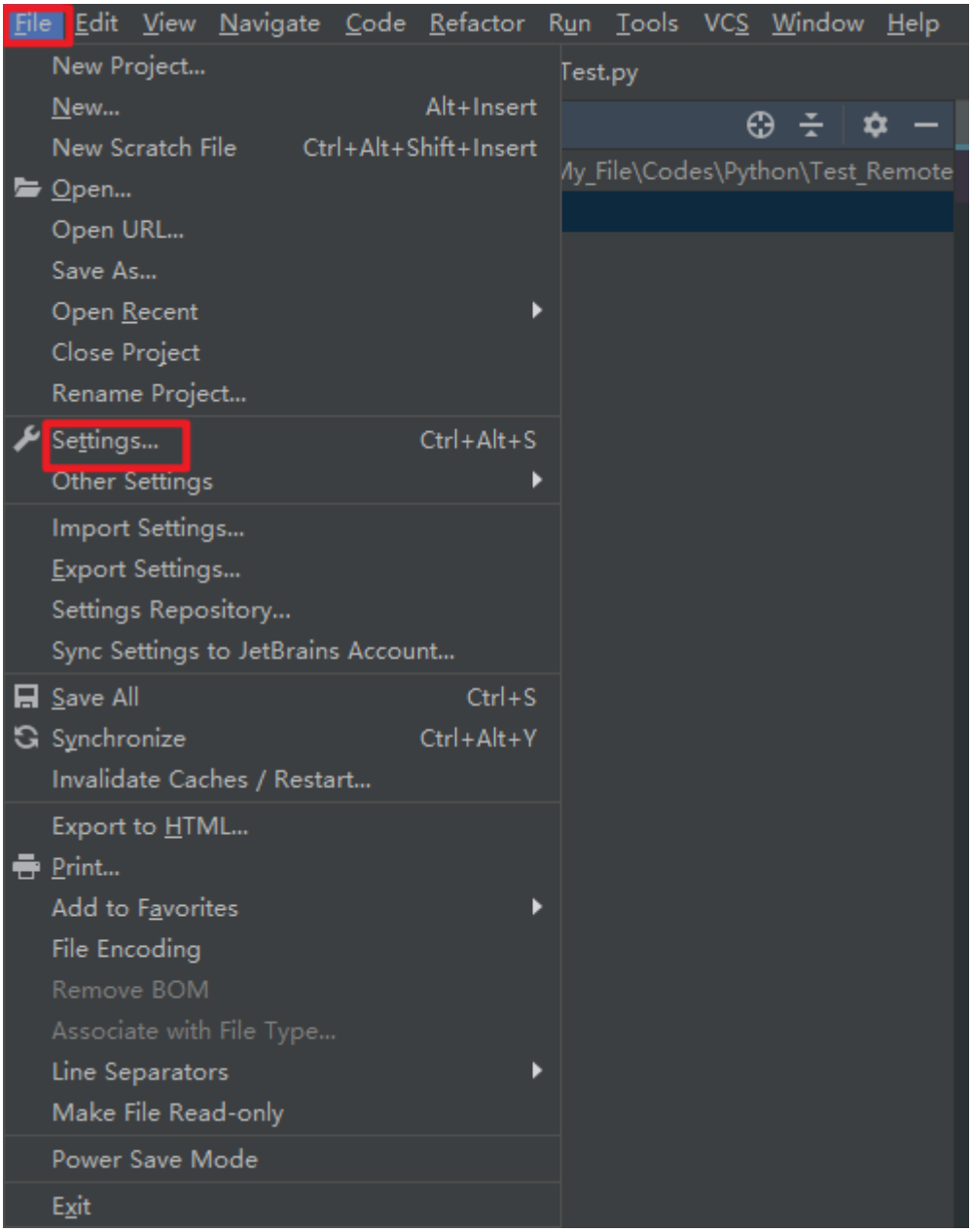
5. `Interpreter` 浏览你目标使用的远程解释器的位置，一般是 `/usr/bin/python` 或 `/usr/bin/python3` ---> `Finish`



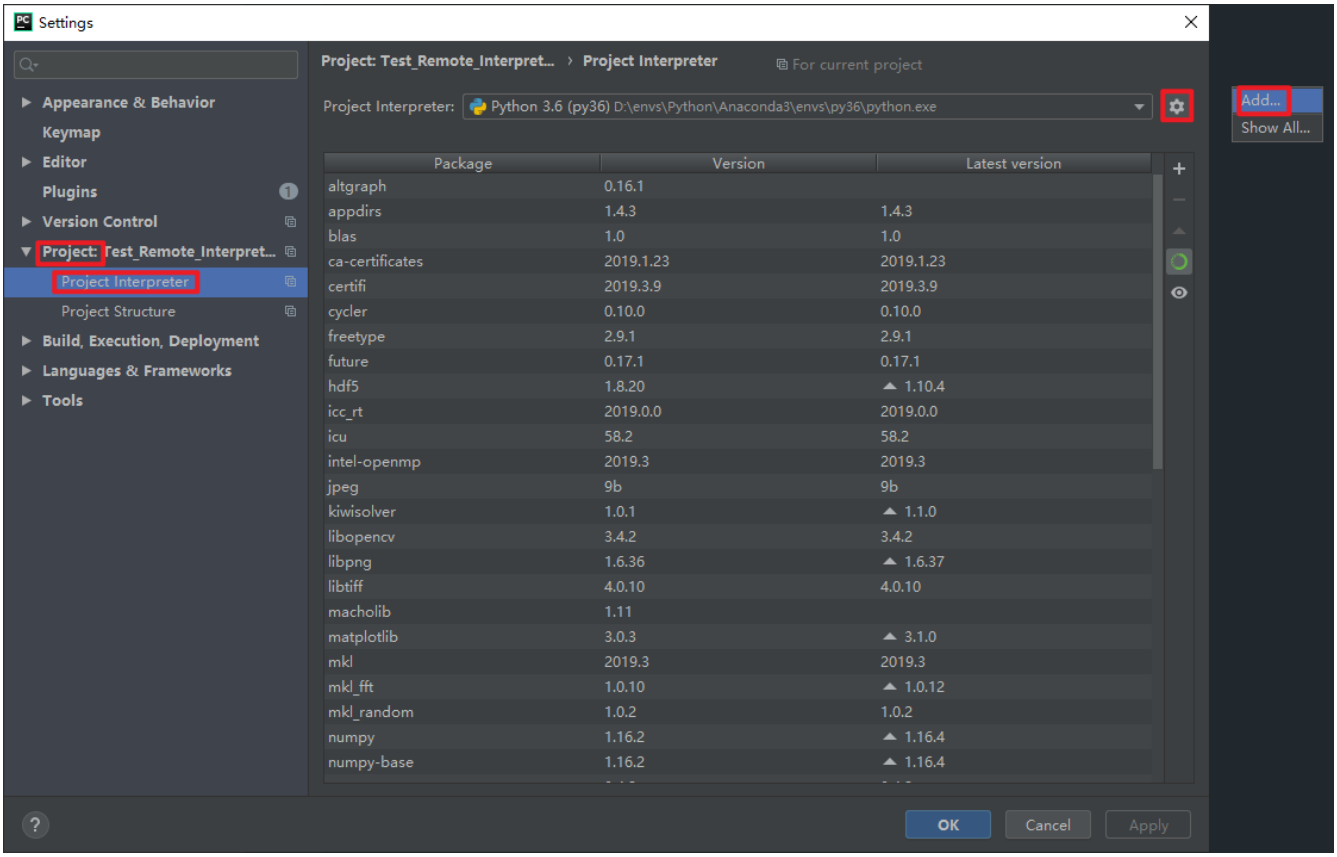


从项目页进入设置（默认仅配置项目可见）

1. 菜单栏 ---> `File` ---> `Settings...`

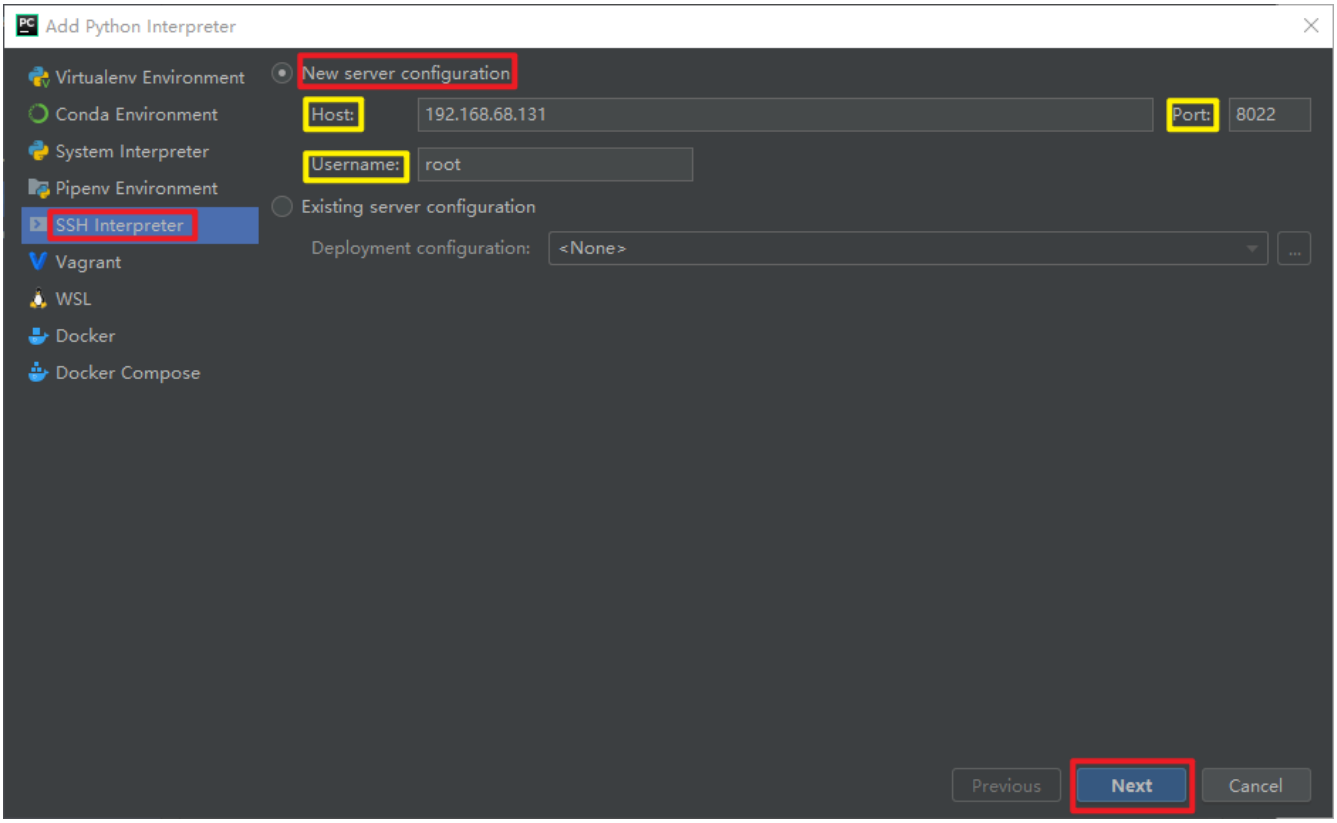


2. `Project: (项目名)` ---> `Project Interpreter` ---> 小齿轮按钮 ---> `Add...`



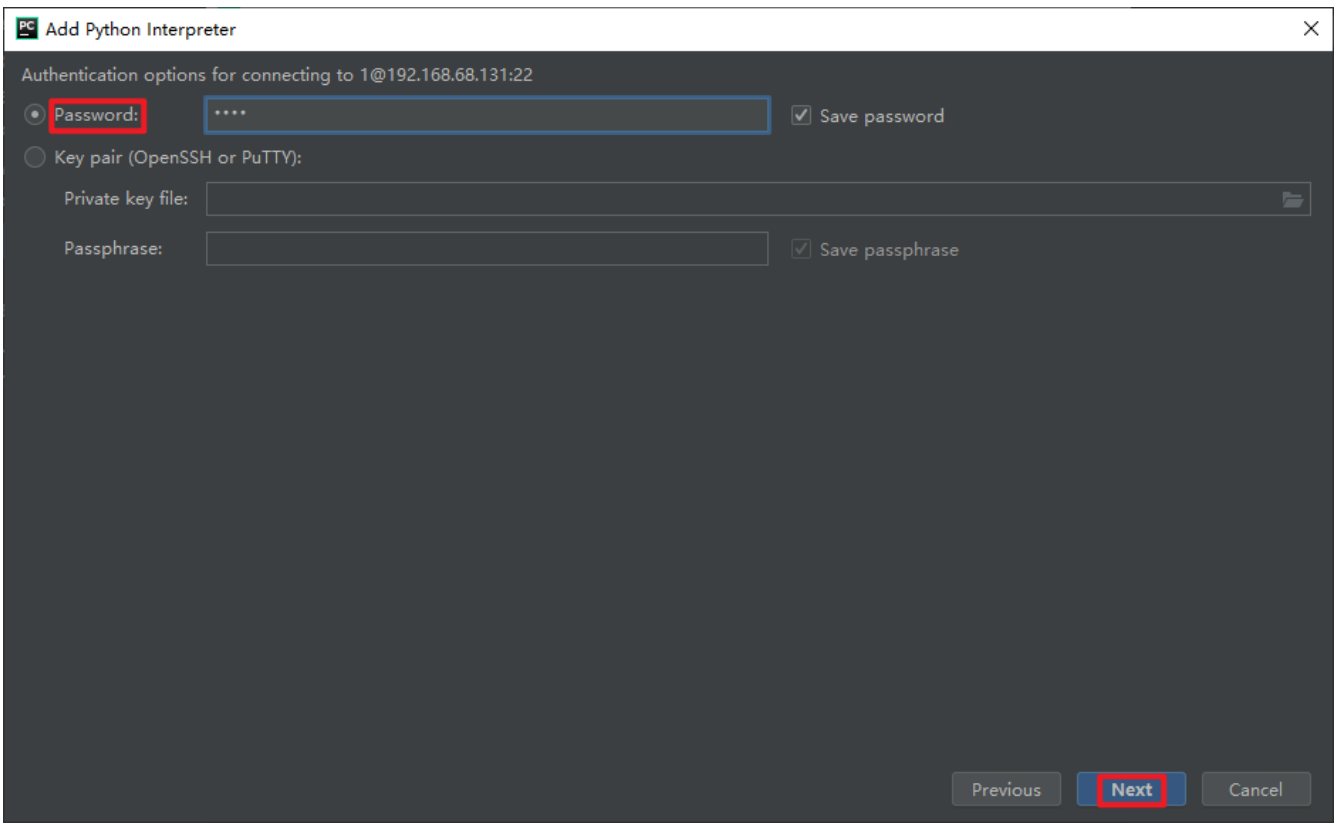
3. `SSH Interpreter` ---> `New server configuration` ---> 填入上一步SSH配置中获得的Host地址 ---> `Port`输入之前新建Docker时设置的22端口的转发端口 ---> `Username` ---> `Next`

> `Host`为宿主机的IP地址，可用`ifconfig`命令获得，之前说过了
>
> `Port`转发端口为之前`新建Docker`时设置的宿主机端口22的转发端口
>
> > `docker run … -p xxx:22 …`命令中的`xxx`即是配置的转发端口
>
> 本步及下一步的账号密码请填入你在SSH配置中用`echo "user:password" | chpasswd`命令配置的`[用户名:密码]`



4. `Password` ---> `Next`

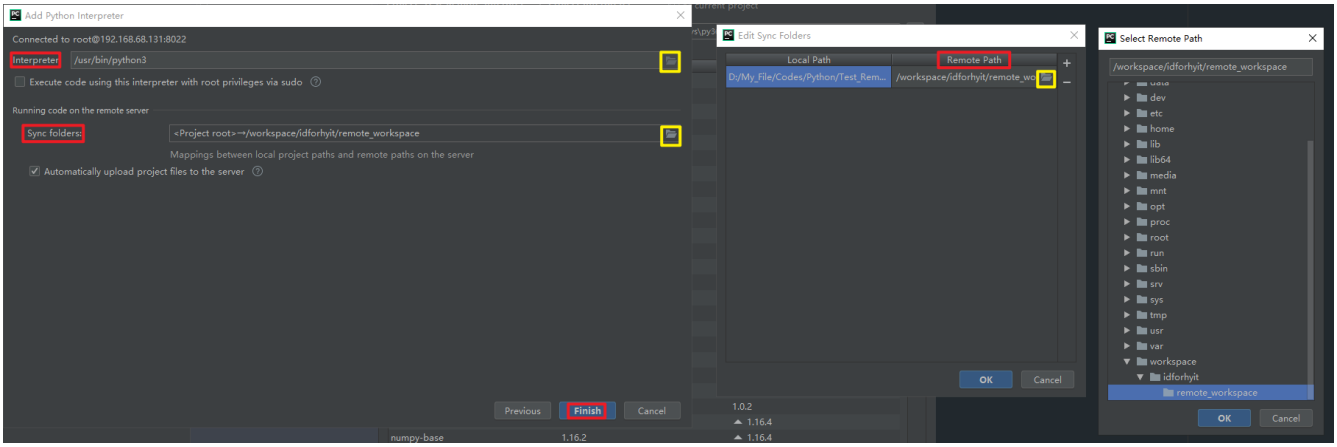
   建议 `Save password` 复选框选中



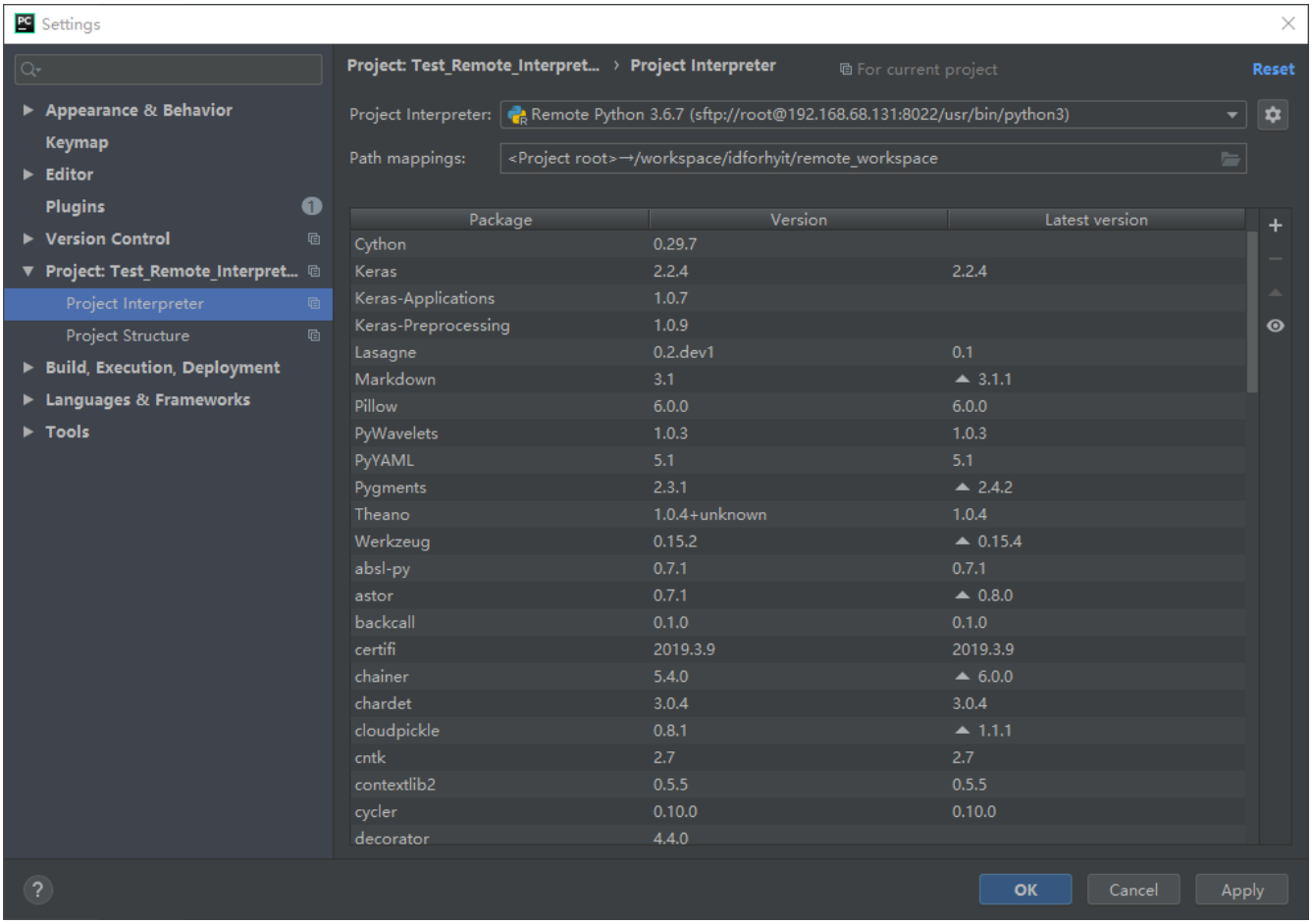
5. `配置Interpreter` ---> `配置Sync folders` ---> `Finish`

> `Interpreter` 是远程解释器的地址，一般是 `/usr/bin/python` 或 `/usr/bin/python3`

> `Sync folders` 是远程解释器的项目存放目录，一般与之前创建Docker时用到的命令地址一样就可，也可点击文件夹图标，选定远程服务器的目录地址。这个地址是你代码的远程同步地址，项目运行时会同步你的代码到此目录，然后用远程解释器的算力进行运算。



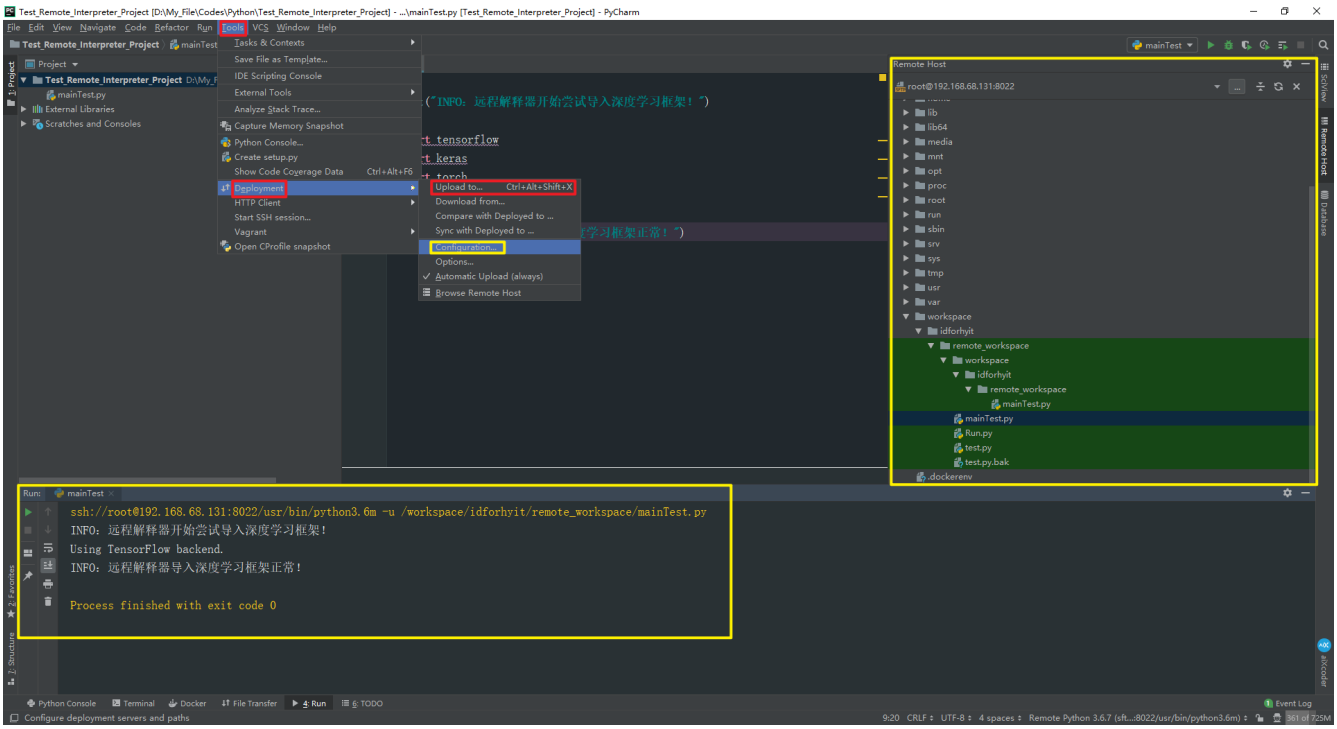
6. 配置完的效果及代码执行界面



点击 菜单栏 ---> `Tools` ---> `Deployment` ---> `Configuration..` 可进一步配置。

运行代码前，点击 菜单栏 ---> `Tools` ---> `Deployment` ---> `Upload to...` 或 `Ctrl+Alt+Shift+X` 将代码同步到远程服务器的配置目录。

最后，整个远程解释器执行代码时的界面如下图所示。代码执行不消耗本地资源，与本地孱弱的性能相比，远程服务器跑实验模型的时间大大缩短。如果学校或是有其他云平台的算力，能够拿到云平台的算力分配给俺们，还是比较美滋滋的，会节省很多时间。



如有错误或疑问，烦请评论指出

> 本篇文章纯手码，请勿转载。文章内容来自博主的亲自操作，查阅了很多资料，但作为菜鸟一枚，肯定还存在许多错误，大佬轻喷，如见错误或操作不一致的地方，烦请在本篇博文地址的文章底部评论指出，会及时于本博文地址进行更正。