首先是安装谷歌浏览器，这里分两种情况，一种你是ubuntu 内核的，你安ubuntu去安装，centos就按照centos的去安装，这次正好我两个服务器一个ubuntu一个centos7的，两边的雷我都趟过了

输入这个命令区分版本

```
uname -a
```

centos

```
[root@VM-8-4-centos ~]#  uname -a
Linux VM-8-4-centos 3.10.0-1160.99.1.el7.x86_64 #1 SMP Wed Sep 13 14:19:20 UTC 2023
 x86_64 x86_64 x86_64 GNU/Linux
[root@VM-8-4-centos ~]#
```

ubuntu

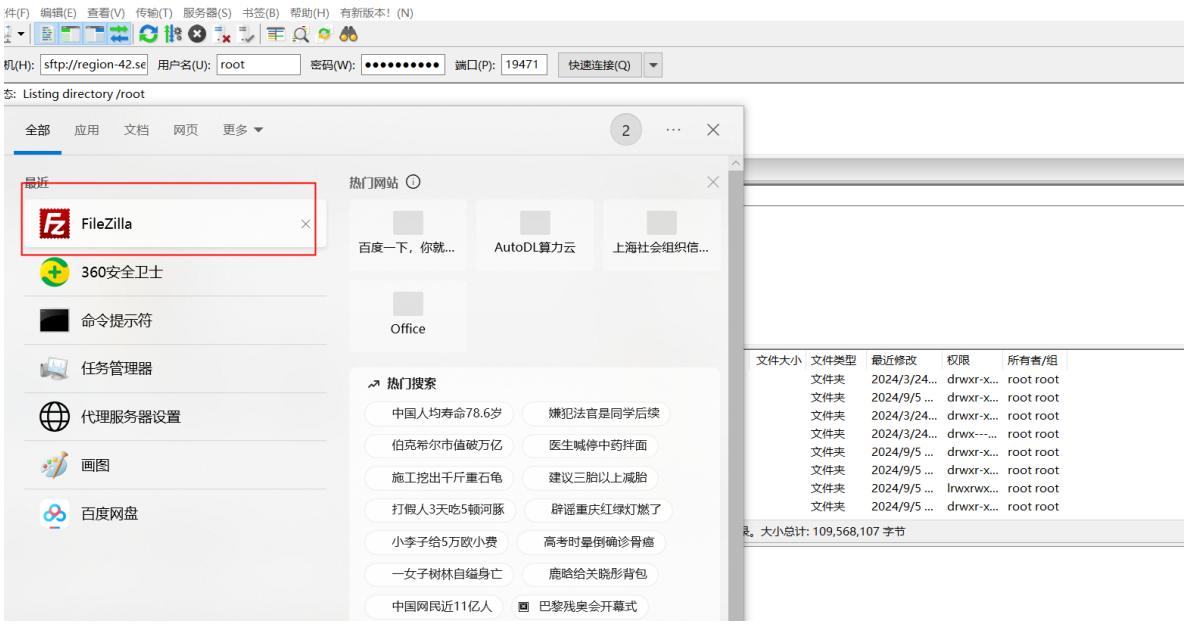
```
root@autodl-container-8e31479808-3ac54cdf:~/ultralytics# uname -a
Linux autodl-container-8e31479808-3ac54cdf 5.4.0-182-generic #202-Ubuntu SMP Fri Apr 26 12:29:36 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux
```

先说centos

参考

https://blog.csdn.net/qq_49349528/article/details/138154019

因为我有fz,所以我就用这个软件去传文件下载文件了，比较好用



百度网盘地址

https://pan.baidu.com/s/1n1ZN-aZwsdsSc7RTo-omhQ?pwd=8d49

把百度网盘下载好的东西传上去，然后

```
yum localinstall google-chrome-beta-125.0.6422.4-1.x86_64.rpm
```

基本就完事了

需要确定的输个y回车就行

ubuntu

参考

https://blog.csdn.net/howard2005/article/details/124906494

```
wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
sudo dpkg -i google-chrome-stable_current_amd64.deb
sudo apt-get -f install    这步是自动安装依赖包，然而为用了不好使，提示缺少依赖包把我装的移除了
```

但是按照这个装我还是会报错，提示缺少依赖包，自动修复不了，当场去世

解决方案：

```
更新包列表：
sudo apt-get update
安装缺少的依赖包：
sudo apt-get install fonts-liberation libgbm1 xdg-utils
重新安装 google-chrome-stable
sudo dpkg -i google-chrome-stable_current_amd64.deb
自动解决依赖关系（如果需要）：
sudo apt-get install -f
按照上述方法，问题解决
```

装好之后，使用以下命令来验证是否装上了，也可以直接执行启动文件，如果没装好会报错

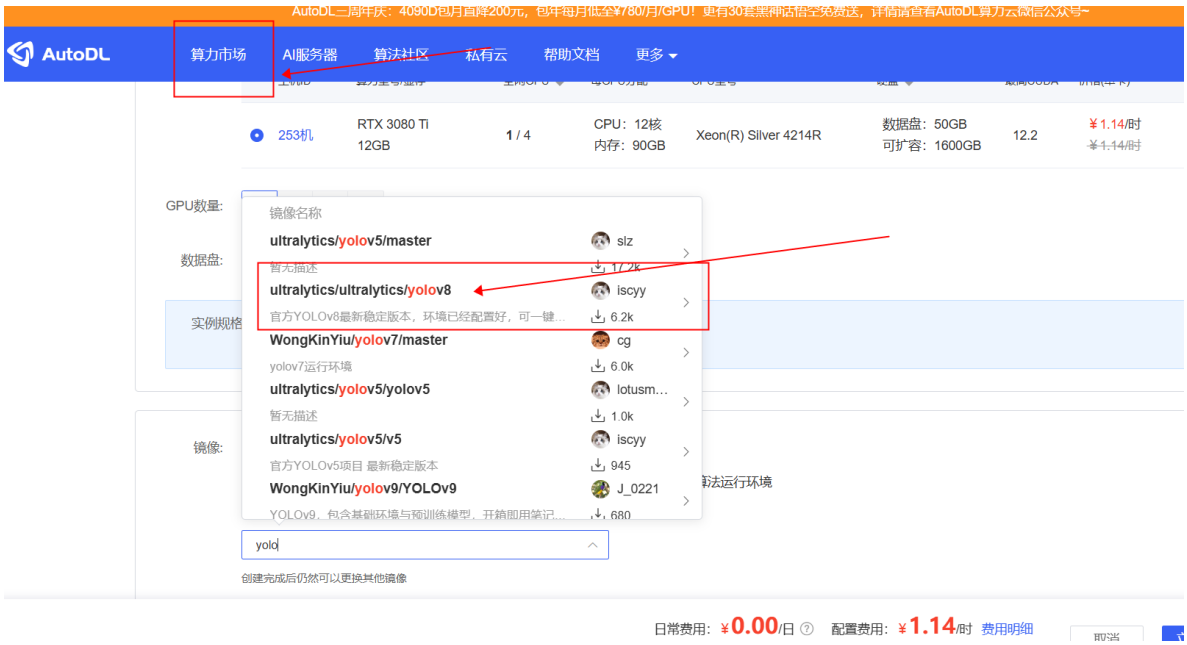
```
google-chrome
```

```
[root@VM-8-4-centos ~]# google-chrome
[11089:11089:0905/214259.962157:ERROR:zygote_host_impl_linux.cc(99)] Running as roo
t without --no-sandbox is not supported. See https://crbug.com/638180.
[root@VM-8-4-centos ~]#
```

然后我们吃现成的

打开炼丹平台

AutoDL算力云 | 弹性、好用、省钱。租GPU就上AutoDL



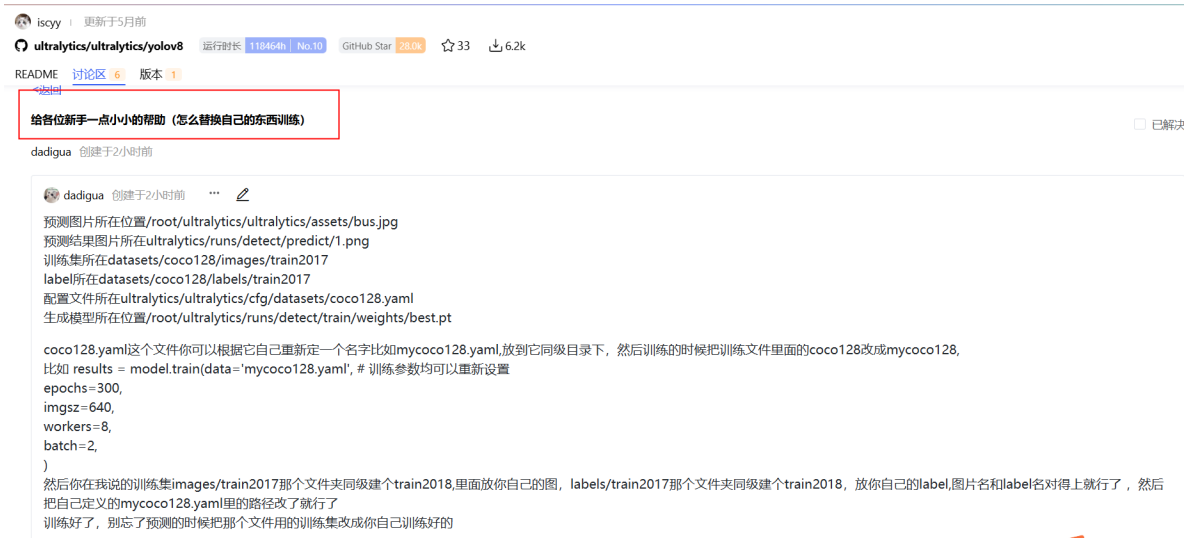
整个服务器，用这个现成的镜像，省心了

这块 的显卡都行都能用，选个便宜的就行其实，2080ti就不错，没了就3080ti ,3090有点贵，按量计费就可以了

这个项目有点乱，这里给个位置

预测图片所在位置/root/ultralytics/ultralytics/assets/bus.jpg
预测结果图片所在ultralytics/runs/detect/predict/1.png
训练集所在datasets/coco128/images/train2017
训练集所在datasets/coco128/images/train2018/1.png
label所在datasets/coco128/labels/train2017
label所在datasets/coco128/labels/train2018/1.txt
配置文件所在ultralytics/ultralytics/cfg/datasets/coco128.yaml
生成模型所在位置/root/ultralytics/runs/detect/train/weights/best.pt

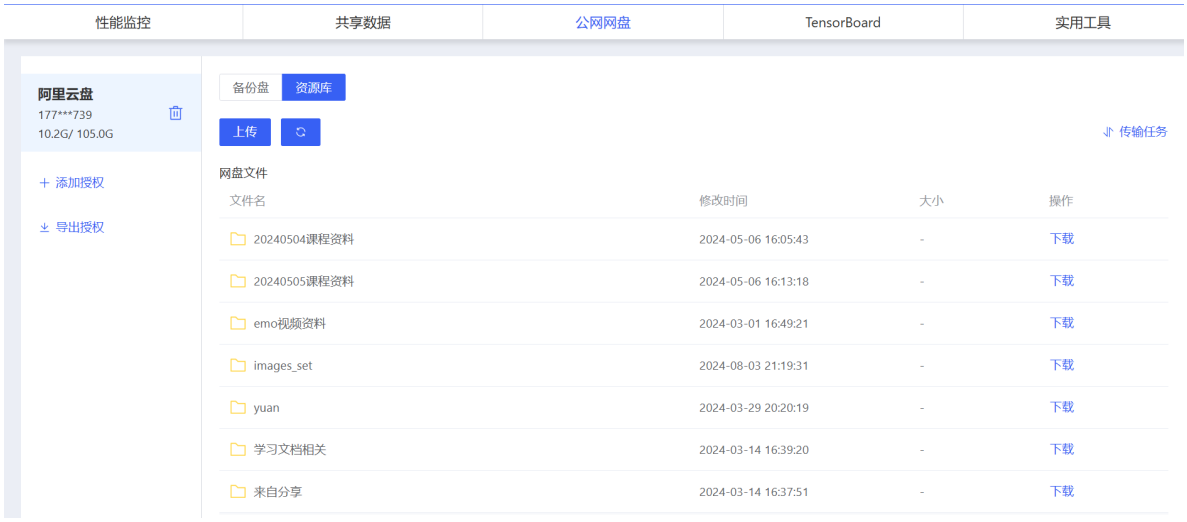
另外这个是我留下的



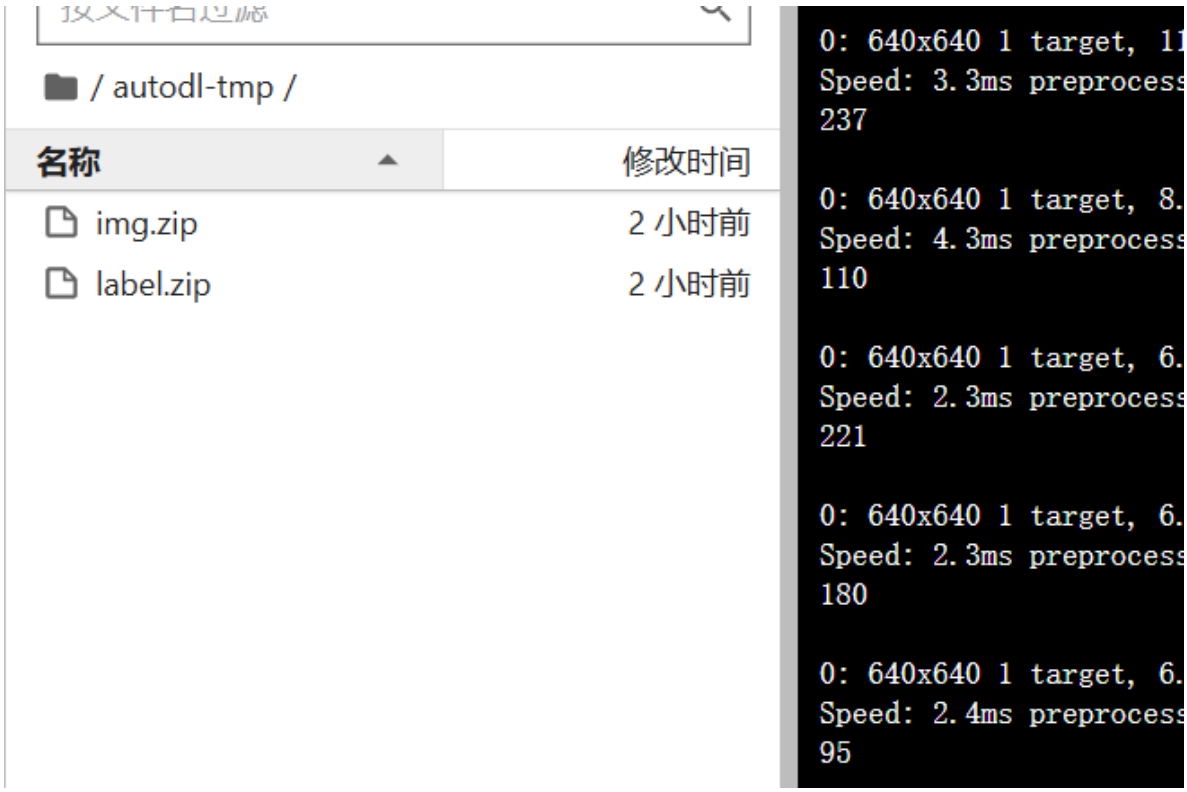
买完服务器大概这样

1是你的控制台，可以用来输入命令，2是你的网盘用来把本地文件通过网盘上传到服务器，也可以把服务器的东西下载到网盘上

网盘要用阿里网盘，打开这样

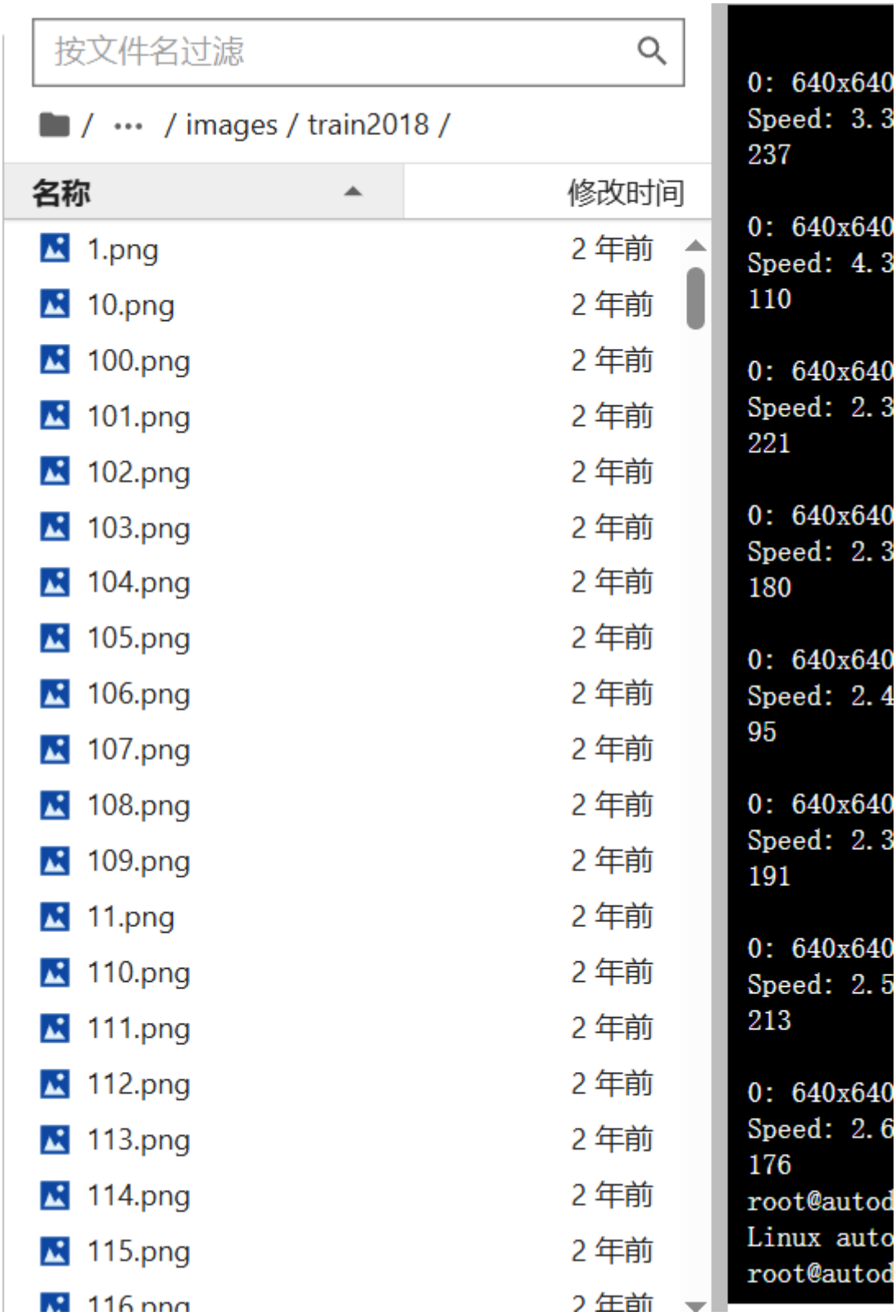


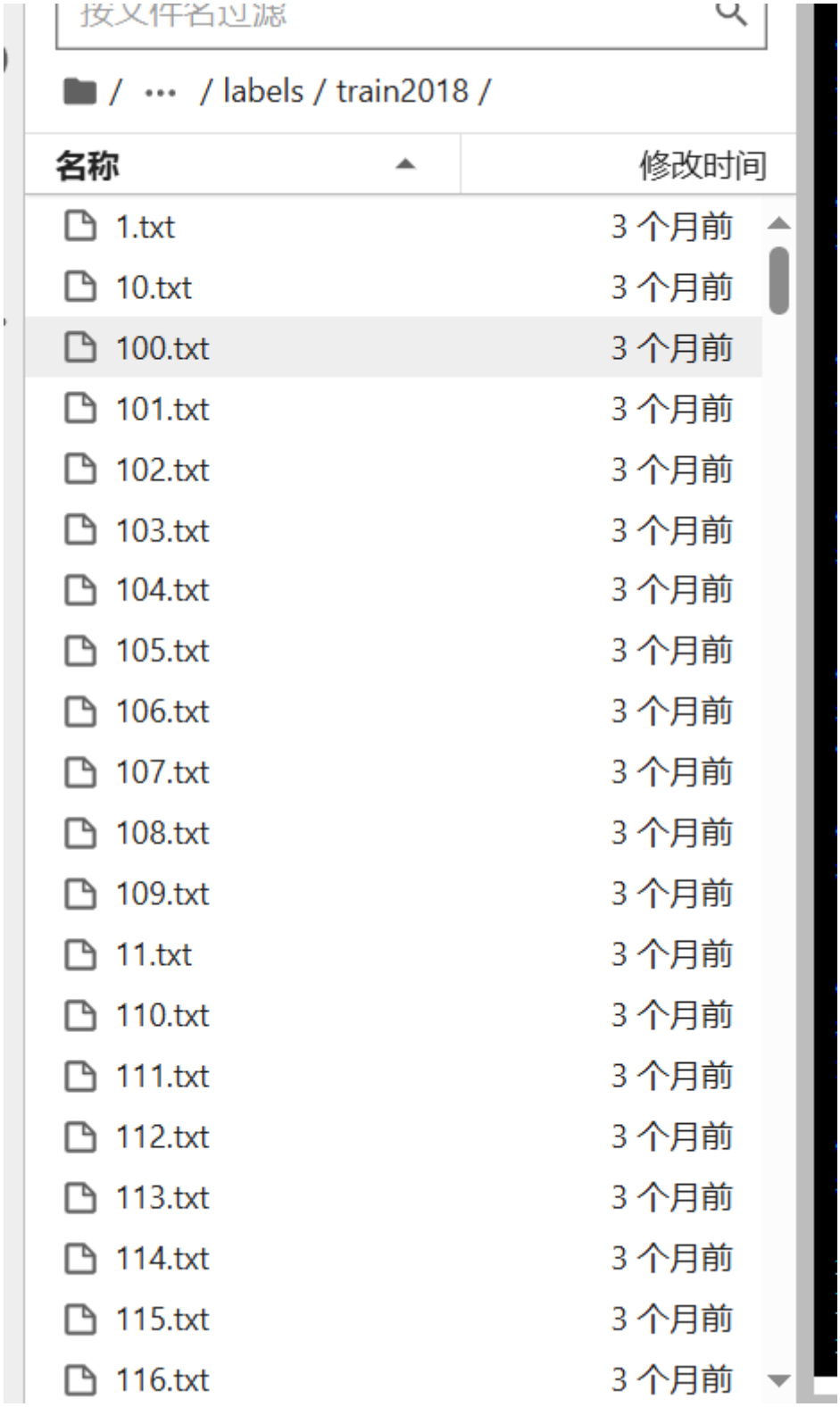
通过网盘上传的东西都在



你下载的东西当然都到网盘里了

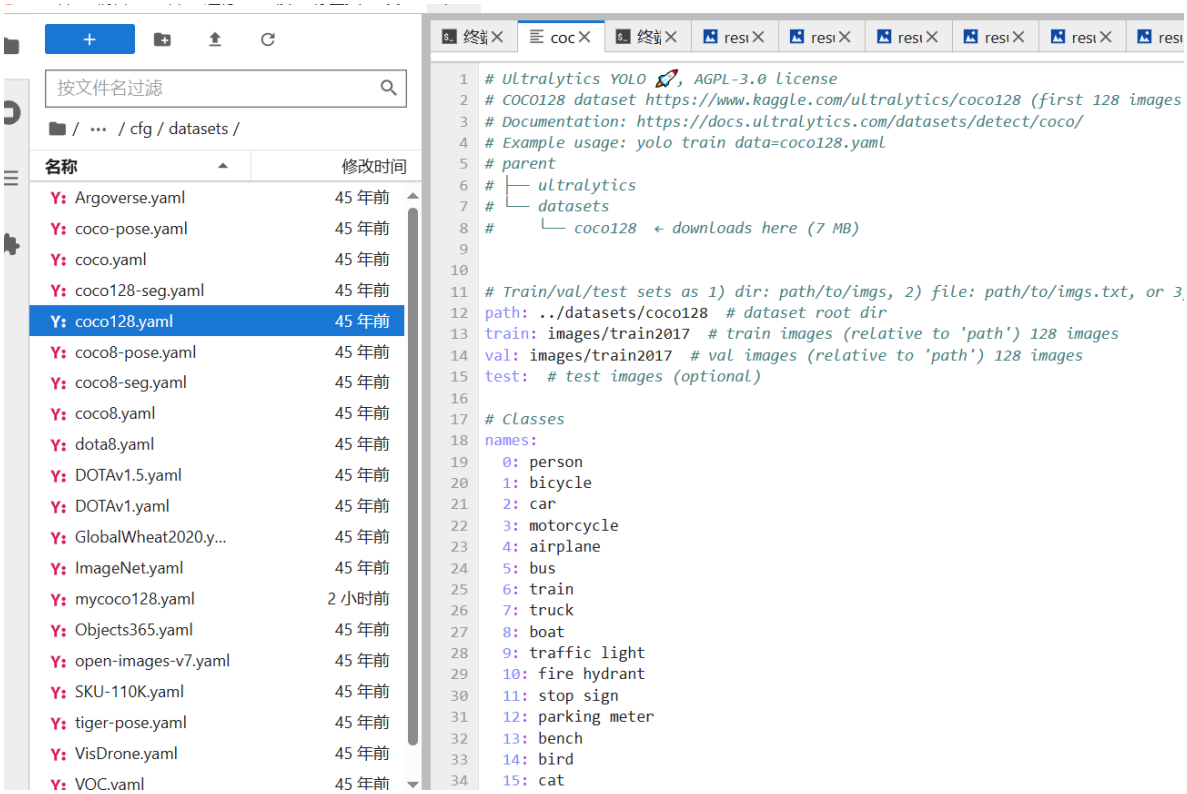
这里给大家看下，首先是替换自己的训练集


按文件名过滤
/ ··· / images / train2018 /
名称
修改时间
1.png 2 年前
10.png 2 年前
100.png 2 年前
101.png 2 年前
102.png 2 年前
103.png 2 年前
104.png 2 年前
105.png 2 年前
106.png 2 年前
107.png 2 年前
108.png 2 年前
109.png 2 年前
11.png 2 年前
110.png 2 年前
111.png 2 年前
112.png 2 年前
113.png 2 年前
114.png 2 年前
115.png 2 年前
0: 640x640
Speed: 3.3
237
0: 640x640
Speed: 4.3
110
0: 640x640
Speed: 2.3
221
0: 640x640
Speed: 2.3
180
0: 640x640
Speed: 2.4
95
0: 640x640
Speed: 2.3
191
0: 640x640
Speed: 2.5
213
0: 640x640
Speed: 2.6
176
root@autod
Linux auto
root@autod



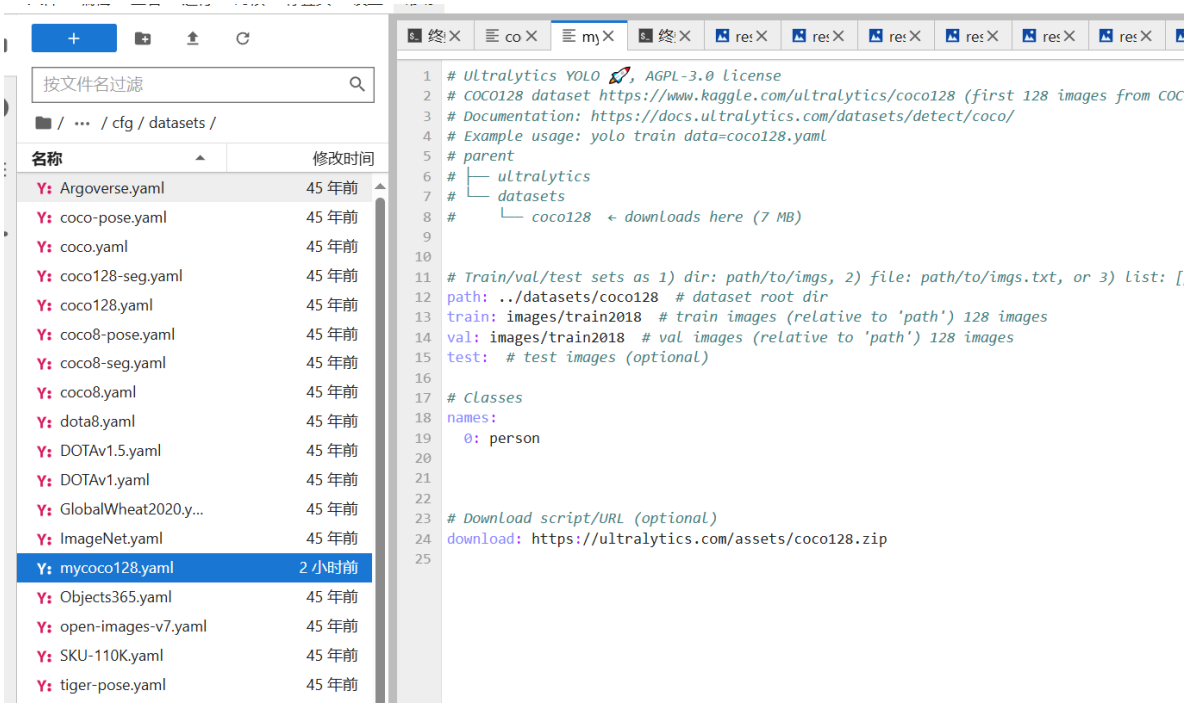
可以看到是这样的

然后改coco128.yaml 这样



我们找到它在它同级再建一个，名字随便起，可以叫mycoco128.yaml

内容



这里我们其实只识别一个图，就是滑块，所以可以把名字那删除的就剩一个就行了，名字也可以改改，我比较懒就没改

然后改train_v8.py这个文件

```
import sys
import argparse
import os

# sys.path.append('/root/ultralyticsPro/') # Path 以Autodl为例
```

```
from ultralytics import YOLO

def main(opt):
    yaml = opt.cfg
    model = YOLO(yaml)

    model.info()

    results = model.train(data='mycoco128.yaml',  # 训练参数均可以重新设置
                          epochs=300,
                          imgsz=640,
                          workers=8,
                          batch=2,
                          )

def parse_opt(known=False):
    parser = argparse.ArgumentParser()
    parser.add_argument('--cfg', type=str,
default='ultralytics/cfg/models/v8/yolov8.yaml', help='initial weights path')
    parser.add_argument('--weights', type=str, default='last.pt', help='')


    opt = parser.parse_known_args()[0] if known else parser.parse_args()
    return opt

if __name__ == "__main__":
    opt = parse_opt()
    main(opt)
```

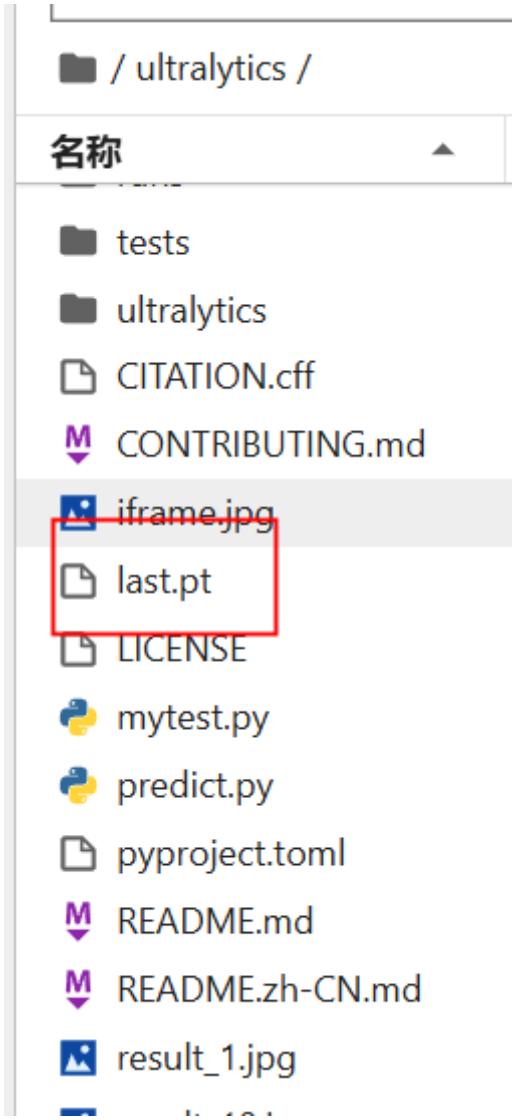
主要把results = model.train(data='mycoco128.yaml', # 训练参数均可以重新设置
epochs=300,
imgsz=640,
workers=8,
batch=2,
)

这个名字改成我们自己定义那个

另外注意——weights这个参数，后面可以接模型名字，也就是如果我们已经有训练好的模型想接着训练就可以把路径加到它后面，这里我把训练好的last.pt拉到和它同目录了，所以默认值给的也是last.pt,这样就直接

```
python train_v8.py
```

这样去执行就可以了



这个last是我从训练好的模型位置拉过来的，那个保存训练好的模型的位置我上面给了

然后我们安装

```
pip install DrissionPage
```

这个环境里的python是3.8的所以没有问题

然后我们搞一个文件测试一下

这里我选豆瓣

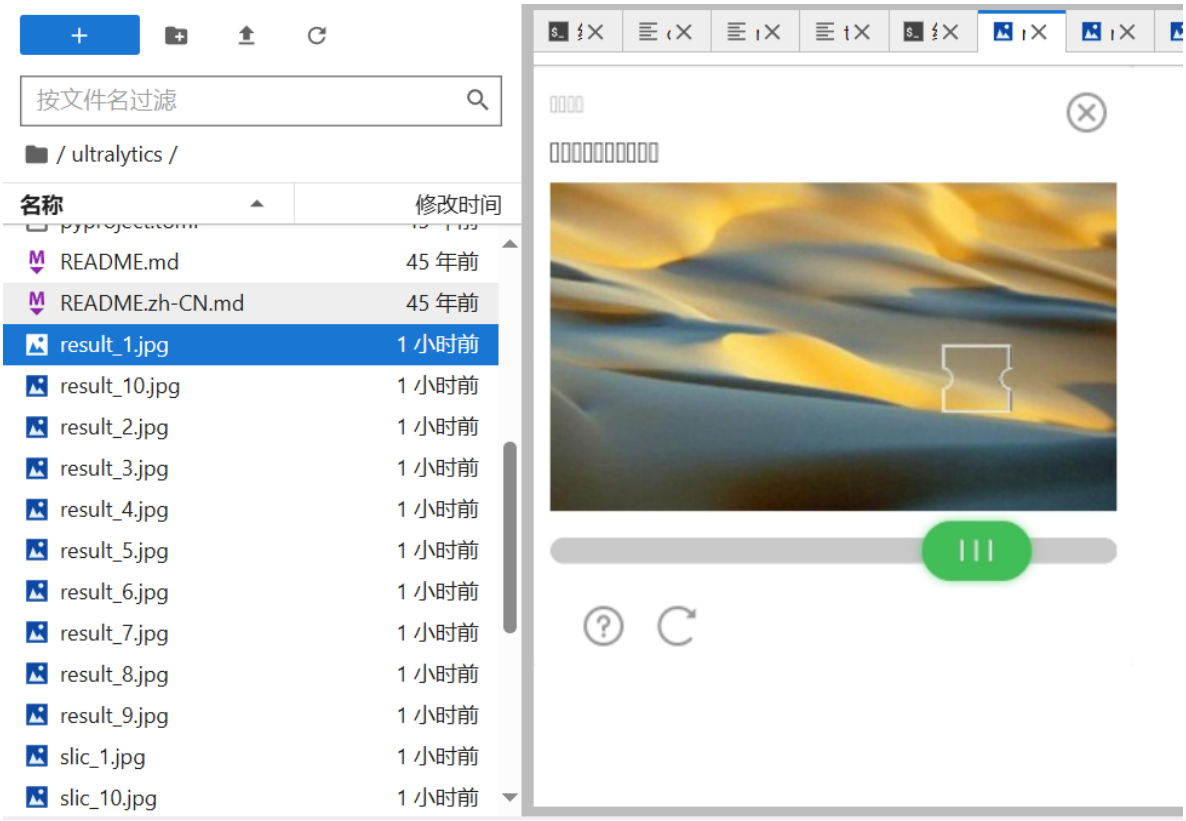
https://accounts.douban.com/passport/login?source=main

```
from DrissionPage import ChromiumPage,ChromiumOptions
from ultralytics import YOLO
import cv2

# 加载模型
model = YOLO('last.pt')  # 替换为你的模型权重文件路径
co = ChromiumOptions().set_paths(browser_path=r"/opt/google/chrome/google-chrome")
co.headless(True)  # 设置无头加载  无头模式是一种在浏览器没有界面的情况下运行的模式，它可以提高浏览器的性能和加载速
```

```
co.set_argument('--no-sandbox')  # 禁用沙箱 禁用沙箱可以避免浏览器在加载页面时进行安全检查,从而提高加载速度 默认情况下，所有Chrome 用户都启用了隐私沙盒选项 https://zhuanlan.zhihu.com/p/475639754
co.set_argument("--disable-gpu")  # 禁用GPU加速可以避免浏览器在加载页面时使用过多的计算资源，从而提高加载速度
co.set_user_agent( user_agent='Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36')  # 设置ua
page = ChromiumPage(co)
page.get('https://accounts.douban.com/passport/login?source=main')
page.ele('密码登录').click()
page.ele('#username').clear()
page.ele('#username').input('13204080367')
page.ele('#password').clear()
page.ele('#password').input('a123456')
page.ele('.:btn btn-account btn-active').click()
num = 1
for ii in range(10):
    page.wait(1)
    iframe = page.get_frame('#tcaptcha_iframe_dy')
    iframe.get_screenshot(path='slic_{}.jpg'.format(num))
    # 读取图像
    image_path = 'slic_{}.jpg'.format(num)  # 替换为你的图像路径
    image = cv2.imread(image_path)

    # 进行预测
    results = model(image)

    # 提取左边距
    # left_margins = []
    for result in results:
        boxes = result.boxes.xyxy  # 边界框坐标
        for box in boxes:
            x1, y1, x2, y2 = map(int, box)
            # left_margins.append(x1)
            print(x1-46)
    iframe.actions.move_to('x://*[@id="tcOperation"]/div[6]').hold().move(offset_x=x1-46,duration=1.5)
    iframe.get_screenshot(path='result_{}.jpg'.format(num))
    #点击刷新按钮
    iframe.actions.move_to('x://*[@id="reload"]/img').click()
    num+=1
```

上面代码简单来说就是识别缺口位置，然后拖动滑块过去截个图，但是不松手，之后直接就点击刷新开始下一次，一共重复10次，分别保存拖动前和拖动之后的，来看准不准。这里注意以下，我这里减去一个48是因为那个滑块按钮的位置一开始就不是在最左边的，它有一段距离，而我们打印的x1是检测到滑块的左边距

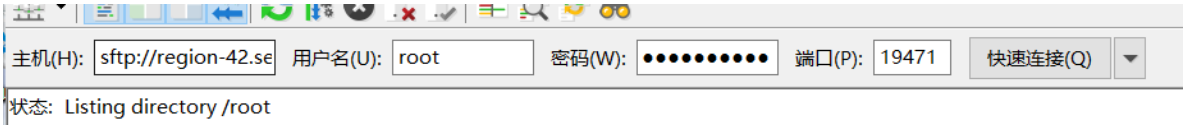
大概这样



我觉得还是有不准的，可能样本量还不够，目前是300图

另外说下fz怎么连接远程服务器

链接买的算力服务器是这样的



根据

这个图的3可以得出

正常的服务器，端口是22，这个服务器的端口你复制粘贴出来就可以看到，大概这样：

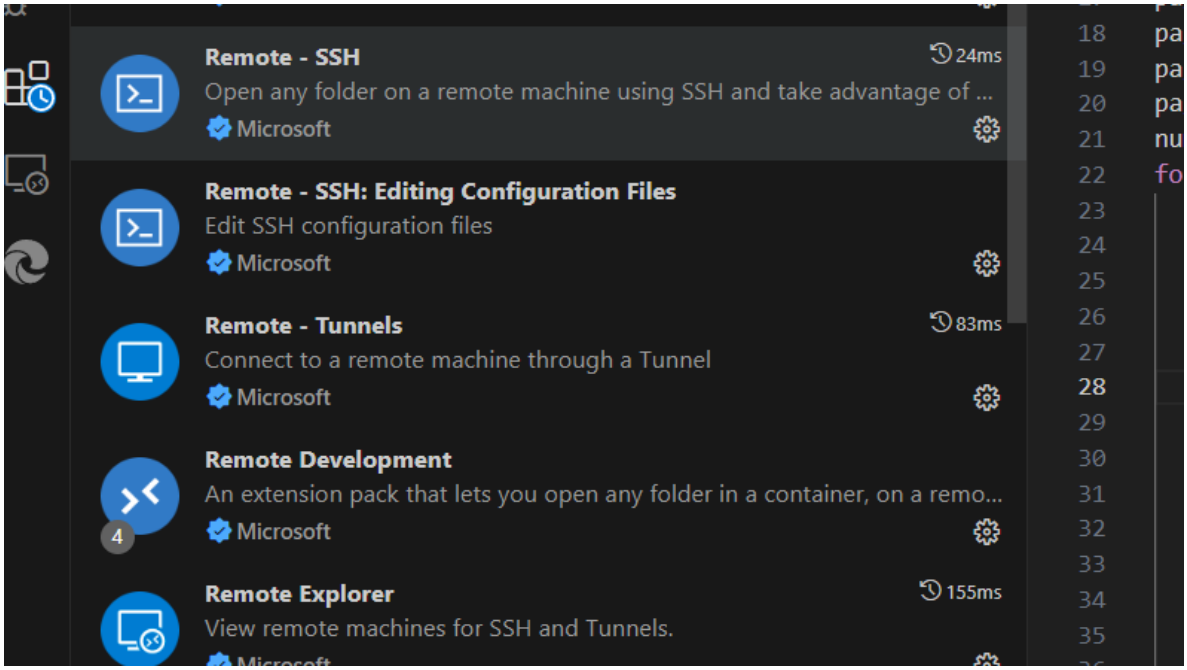
```
ssh -p 19471 root@region-42.seetacloud.com

ft2tgItBSxSL
```

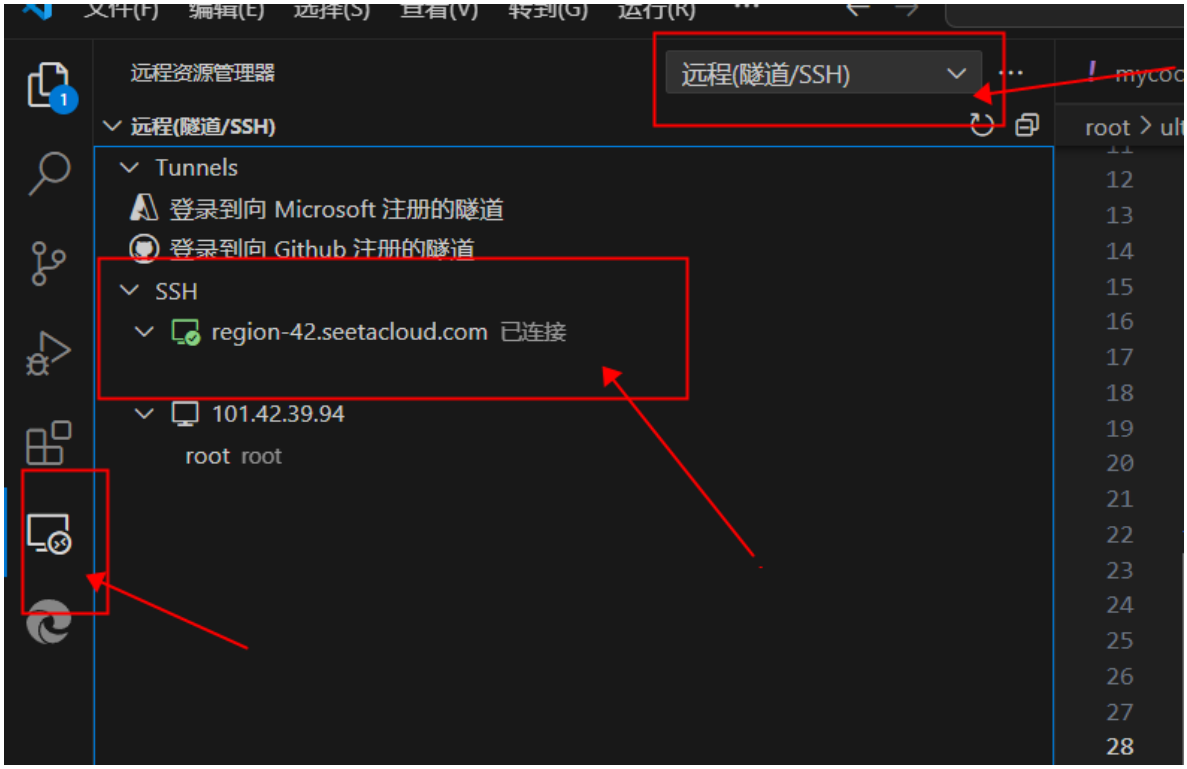
这个玩意一会用vscode链接远程服务器来编辑代码还会用到，因为linux自带的编辑器实在是蛋疼，所以我们用vscode链接远程服务器，来编辑代码

用vscode链接远程服务器

参考

[vscode通过ssh连接远程服务器+免密登录（图文） vscode ssh-CSDN博客]





这几个位置，注中间卡住要你输入密码

ssh -p 19471 root@region-42.seetacloud.com

这个格式要变成

ssh root@region-42.seetacloud.com -p 19471

输入进去

@后面是你服务器的地址，有域名就用域名，没有就用IP地址也行

改完配置记得点击一下刷新，要不出不来