

一、服务器连接

- 1 - 推荐一个云服务器平台：恒源云

The screenshot shows a grid of GPU models and their configurations:

M40-24G	4090-24G	4080-16G	4070 Ti-12G	3090 Ti-24G	4070-12G	4060 Ti-16G
3090-24G	3080 Ti-12G	3080-10G	3080-12G	4060-8G	3070-8G	3060 Ti-8G
3060Ti-8G	3060-12G	2080 Ti-11G	2070 SUPER-8G	2060 SUPER-8G	1080 Ti-12G	V100-32G
P40-24G	T4-16G	P4-8G	TITAN V-12G	TITAN X-12G	TITAN XP-12G	

Below the grid, there is a "GPU数量" (GPU Quantity) dropdown with options 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, and a note: "多机多卡需求请联系客服开通".

The main configuration table includes columns: GPU类型 (GPU Type), 地区 (Region), 数量 (Quantity), 显存 (Memory), 显卡驱动版本 (GPU Driver Version), 最高CUDA版本 (Max CUDA Version), CPU型号 (CPU Model), CPU配置 (CPU Configuration), 内存 (Memory), 实例硬盘 (Instance Disk), 网络 (Network), 价格 (Price), 到期时间 (Expiration Date), and 可用券 (Coupon). A specific row is highlighted for "3090-24G" in "华北" (North China) with 1 unit, 24GB memory, and other details.

At the bottom, it says "数据盘: 免费 50GB".

可以看到 24G 显存的 3090 只需要 0.9 元/小时
不需要大显存的组也可以换成 3060 or 3080，也都很便宜

- 2 - 镜像的选择

The screenshot shows a list of frameworks and their versions:

框架名称	框架版本	Cuda 名称	Python 名称
PyTorch	2.2.1	11.8.0	3.8
TensorFlow	2.0.0	11.7.0	
PaddlePaddle	1.13.1		
OneFlow	1.13.0		
OpenCV	4.5.5		
MXNet	1.12.0		
KataGo	1.11.0		

实例镜像 section shows "没有需要的镜像?" (No required images?).

Configuration details on the right include: CPU配置 (2 32-Core), 内存 (16核), 实例硬盘 (31.5G), 网络 (系统盘: 20G, 数据盘: 50GB SSD 不可扩容), 价格 (¥0.9/小时), 到期时间 (2024-12-21), and 可用券.

At the bottom, it says "费用金额: ¥0.90/小时 计费说明" (Billing amount: ¥0.90/hour Billing details) and "账户可用余额: ¥1.58". A blue "创建实例" (Create Instance) button is at the bottom right.

BERT 是基于 PyTorch 构建的，所以我们在创建服务器时就可直接选择，推荐 PyTorch==2.0.0
Cuda==11.7.0 Python==3.8，比较 stable
然后点击 创建实例 就可以了

- 3 - 服务器的连接

The screenshot shows the instance status: "创建中" (Creating) with ID "1dd1c298100cd1ab5". Other details include: 系统磁盘 (System Disk) 正常 (Normal), 按量付费 (Pay-as-you-go), and 创建时间 (Creation time) 2024-10-18 14:00:36.

等待创建完成，=>

The screenshot shows the instance status: "运行中" (Running) with ID "1dd1c298100cd1ab5". Other details include: 系统磁盘 (System Disk) 正常 (Normal), 按量付费 (Pay-as-you-go), 余额不足/2小时 (Insufficient balance/2 hours), and 创建时间 (Creation time) 2024-10-18 14:00:36.

打开连接工具，首推VSCode，再推荐一个MobaXterm



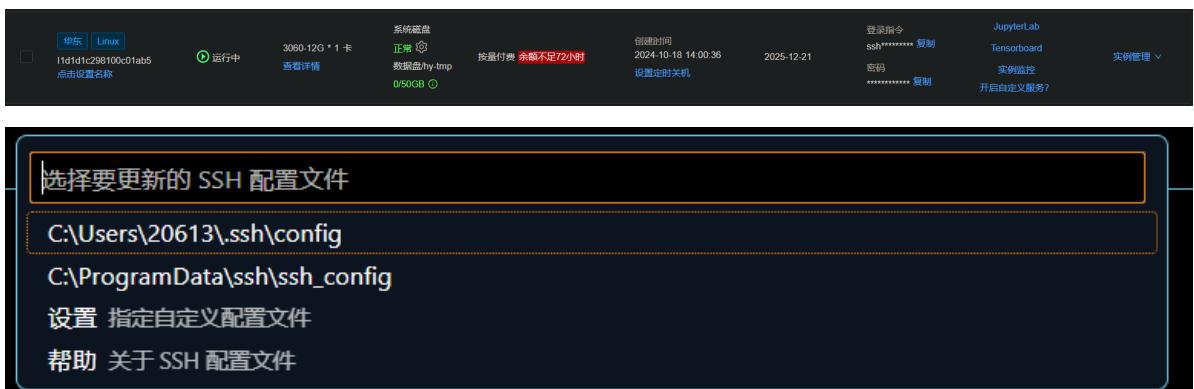
找到左侧的扩展，搜索 `remote ssh`，安装完成之后左侧会有一个新的图标，点击 `SSH` 右侧的 `+`



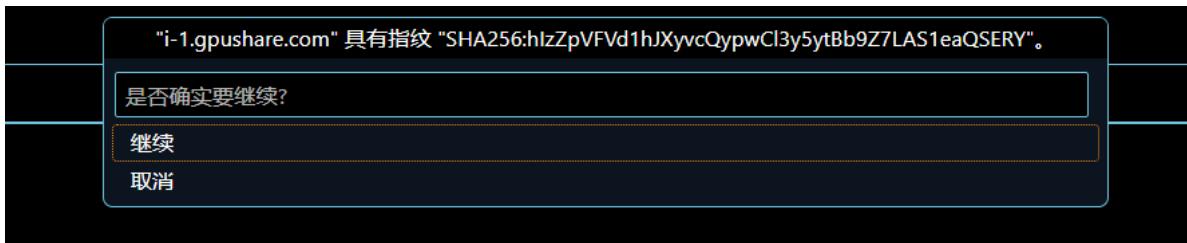
=>



然后点击 登录指令 的第一行 `ssh*****` 右侧的 复制 按钮，输入到VSCode的输入框中，按回车



配置文件选第一个，右下角会出现 `已添加主机`，点击连接



有些可能会出现指纹提示，点 `继续`，有的可能会出现选平台，选 `Linux`

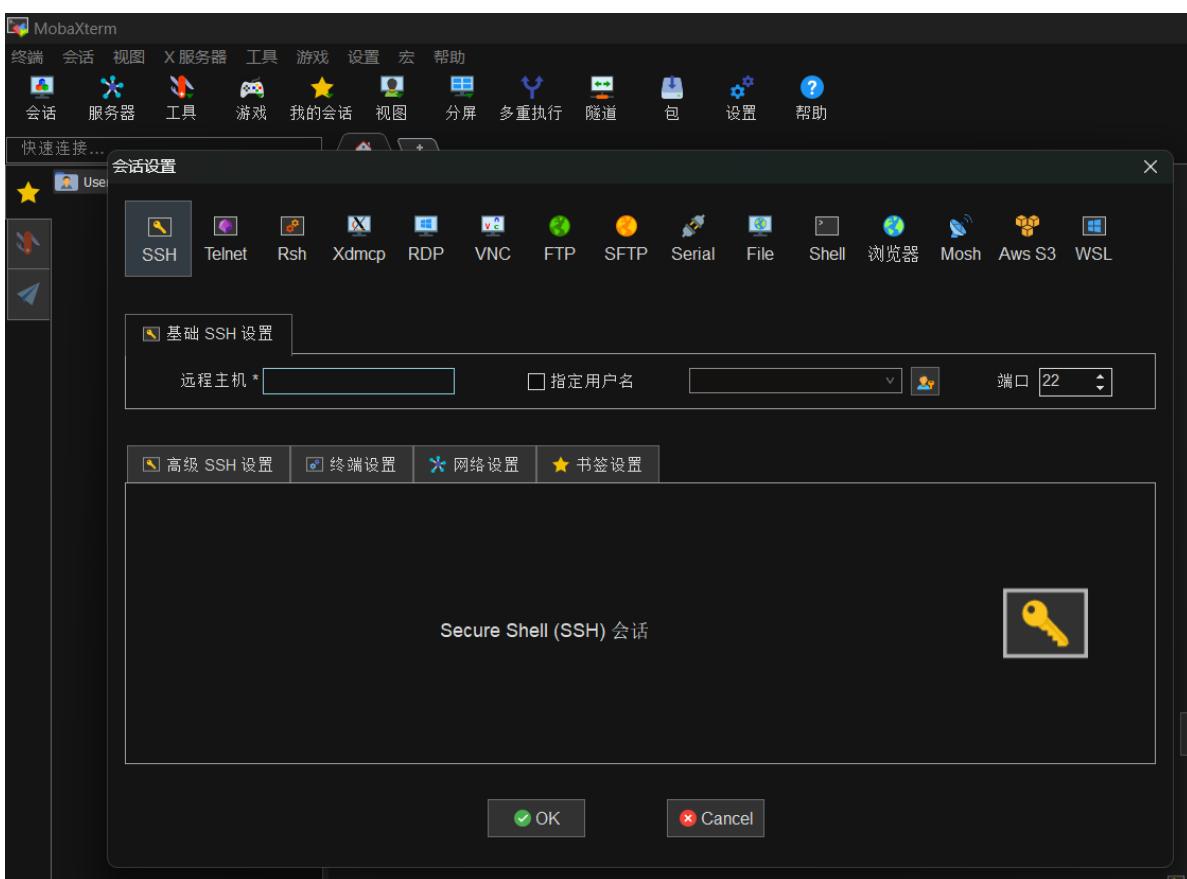


然后再将密码复制过来输入

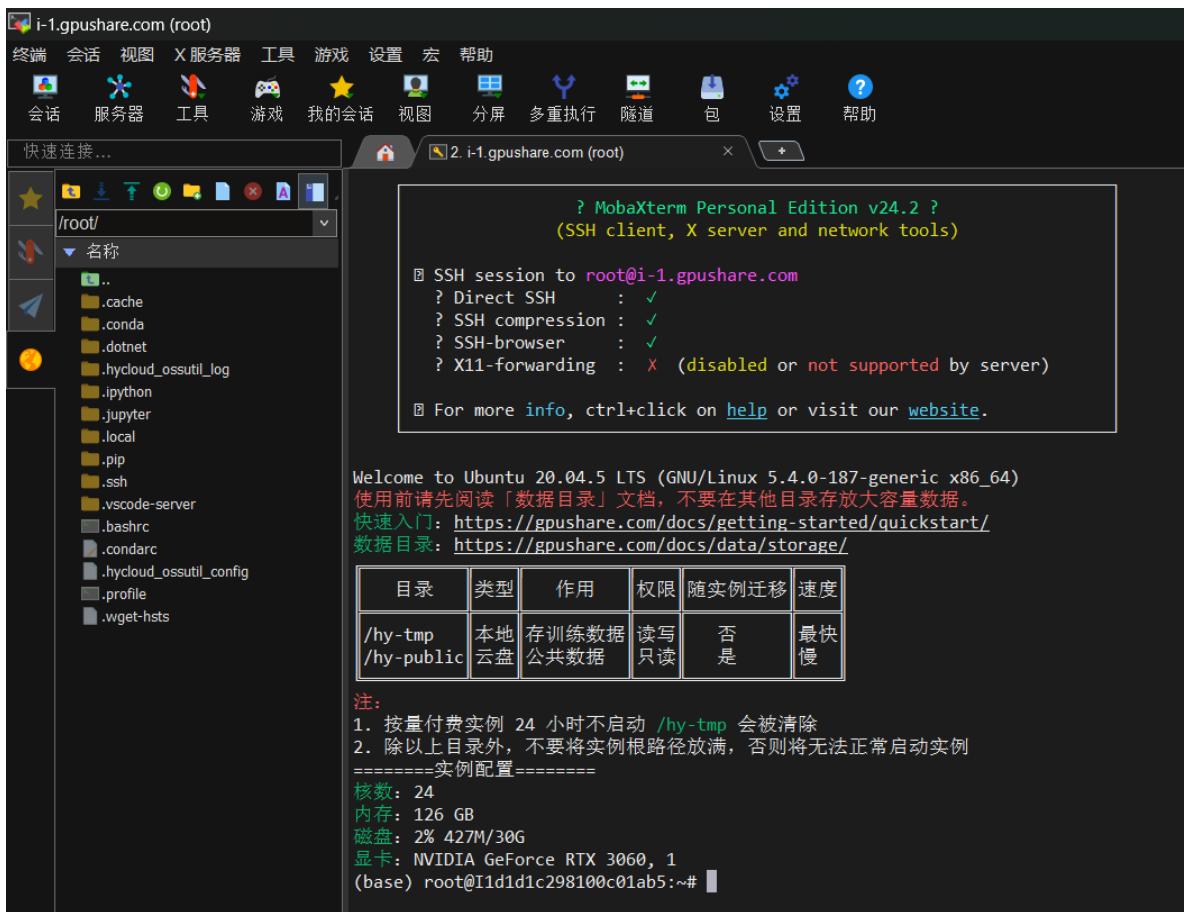


这样就是连接成功！

MobaXterm连接（自行下载）



点击 `会话` 的 `SSH`，将登录指令复制下来，然后分别输入 `远程主机`、`指定用户名` 和 `端口号`，例如我的命令是 `ssh -p 34552 root@i-1.gpushare.com`，那我就应该分别填 `i-1.gpushare.com`、`root`、`34552`，然后点击 `OK`，然后按照提示输入密码。注意，MobaXterm的默认粘贴命令是 `Shift+Insert`！！！（可改）



这样就是连接成功了!

PS: VSCode那么好用, 为什么要MobaXterm?

因为远程服务器不带显示器, VSCode默认没有X-server (一种远程显示技术), 但是MobaXterm自带, 在一些需要显示图片或者其他任务时, VSCode只能保存下来看, 但是MobaXterm可以实时观看。

二、BERT环境配置

因为在创建服务器时就已经装好了 PyTorch，那么我们配环境中最困难的一步已经让别人帮我们做好了，可以运行以下测试代码：test_cuda.py

```
import torch

# 检查CUDA是否可用
cuda_available = torch.cuda.is_available()
print(f"CUDA 是否可用: {cuda_available}")

if cuda_available:
    # 检查当前可用的GPU数量
    gpu_count = torch.cuda.device_count()
    print(f"可用的 GPU 数量: {gpu_count}")
    # 获取当前GPU名称
    for i in range(gpu_count):
        print(f"GPU {i} 名称: {torch.cuda.get_device_name(i)}")
    # 检查当前设备
    current_device = torch.cuda.current_device()
    print(f"当前使用的设备索引: {current_device}")
else:
    print("未检测到可用的 CUDA 设备")
```

```
root@SSH-1:1GPUshare.com:~/.../test_cuda.py
1 import torch
2
3 # 检查CUDA是否可用
4 cuda_available = torch.cuda.is_available()
5 print(f"CUDA 是否可用: {cuda_available}")
6
7 if cuda_available:
8     # 检查当前可用的GPU数量
9     gpu_count = torch.cuda.device_count()
10    print(f"可用的 GPU 数量: {gpu_count}")
11
12    # 获取当前GPU名称
13    for i in range(gpu_count):
14        print(f"GPU {i} 名称: {torch.cuda.get_device_name(i)}")
15
16    # 检查当前设备
17    current_device = torch.cuda.current_device()
18    print(f"当前使用的设备索引: {current_device}")
19 else:
20    print("未检测到可用的 CUDA 设备")

(base) root@SSH-1:1GPUshare.com:~/.../test_cuda.py
CUDA 是否可用: True
可用的 GPU 数量: 1
GPU 0 名称: NVIDIA GeForce RTX 3060
当使用设备索引: 0
(base) root@SSH-1:1GPUshare.com:~/.../test_cuda.py
```

得到如图输出，说明 torch 包的 CUDA 和 GPU 都是可用的

但是想要运行 BERT，还要 pip install transformers，因为他是基于 Transformer 架构的
安装好后，可以运行以下测试代码：test_bert.py

```
from transformers import BertTokenizer, BertForSequenceClassification
import torch

model_name = "roberta-base-finetuned-jd-binary-chinese" # 加载预训练的 tokenizer
和 微调好的情感分类模型(bert-base-chinese)
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertForSequenceClassification.from_pretrained(model_name)
```

```

# 移动模型到 GPU
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)

# 准备输入的文本（中文情感分析）
def predict_sentiment(text):
    # 编码输入
    inputs = tokenizer(text, return_tensors='pt', padding=True, truncation=True,
max_length=512)
    # 将输入移动到 GPU
    inputs = {key: value.to(device) for key, value in inputs.items()}
    # 模型推理（不计算梯度）
    with torch.no_grad():
        outputs = model(**inputs)
    # 获取模型的输出 logits 并将其转换为概率
    logits = outputs.logits
    probs = torch.softmax(logits, dim=-1)
    # 获取最大概率的类别（0：消极，1：积极）
    sentiment = torch.argmax(probs, dim=-1).item()
    # 返回情感类别和相应的概率
    return "积极" if sentiment == 1 else "消极", probs

# 测试文本
text = "今天的天气真好，我感到非常开心！"
# text = input("请输入测试文本：")

# 预测情感
sentiment, probability = predict_sentiment(text)
print(f"文本: {text}")
print(f"预测的情感: {sentiment}")
print(f"概率: {probability}")

```

```

root@192.168.1.1:~# python test_bert.py
  14 def predict_sentiment(text):
  15     inputs = tokenizer(text,
  16         max_length=512)
  17     inputs = {key: value.to(device) for key, value in inputs.items()}
  18     with torch.no_grad():
  19         outputs = model(**inputs)
  20     logits = outputs.logits
  21     probs = torch.softmax(logits, dim=-1)
  22     # 我们需要最大概率的类别 (0: 消极, 1: 积极)
  23     sentiment = torch.argmax(probs, dim=-1).item()
  24     # 返回情感类别和相应的概率
  25     return "积极" if sentiment == 1 else "消极", probs
  26
  27 # 测试文本
  28 text = "今天的天气真好，我感到非常开心！"
  29 # text = input("请输入测试文本：")
  30
  31 g 评估结果
  32 sentiment, probability = predict_sentiment(text)
  33 print(f"文本: {text}")
  34 print(f"预测的情感: {sentiment}")
  35 print(f"概率: {probability}")
  36
  37 # text = input("请输入测试文本：")
  38
  39 g 评估结果
  40 sentiment, probability = predict_sentiment(text)
  41 print(f"文本: {text}")
  42 print(f"预测的情感: {sentiment}")
  43 print(f"概率: {probability}")

root@192.168.1.1:~# 

```

得到如图输出，就说明 BERT 的基础环境也已经配好了！

着手开始构建属于自己的模型吧！

附录

1. 如果服务器连接不上，请核对要填的各个信息是否正确，也可以在 `windows Terminal` 中直接输入连接命令以及密码，测试命令以及密码是否正确，若否，请重启服务器。
2. 运行 `test_bert.py` 时若报错显示没有或者无法下载 `tokenizer` 以及模型，可以前往 [`uer/roberta-base-finetuned-jd-binary-chinese-Hugging Face`](#) 手动下载，其他模型(大量)也可在 [`Hugging Face`](#) 找到并手动下载。

`Mail` : jiuri.z.0796@gmail.com

`Date` : 2024/10/18