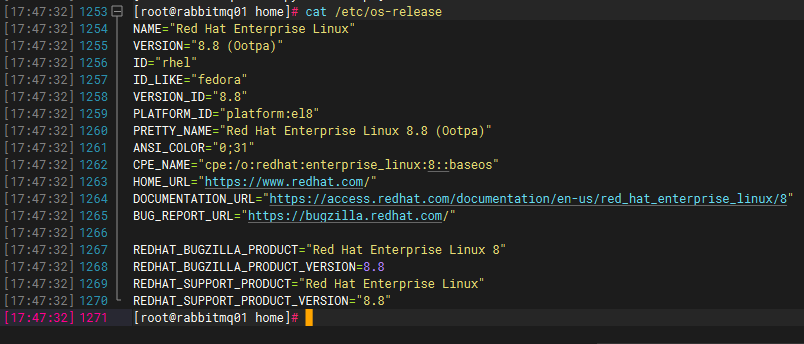
**Linux下Python程序打包方法**

# 系统基础环境检查

# 查看系统版本

cat /etc/os-release



# 查看Miniconda 版本

conda –version

# 如果没有输出版本信息，请先安装

# 创建一个目录

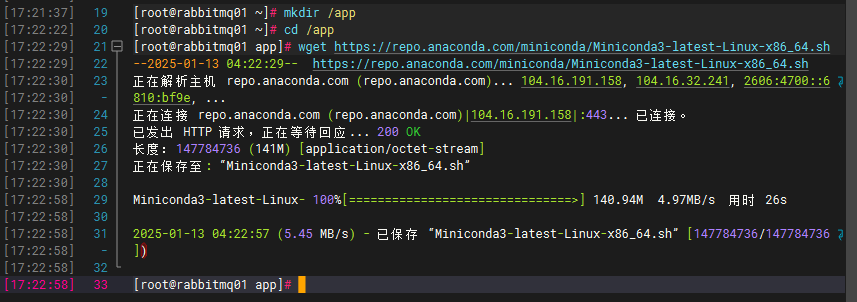
mkdir /app

# 进入创建的目录

cd /app

# 下载Miniconda

wget <https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh>



# 使脚本可执行

chmod +x Miniconda3-latest-Linux-x86\_64.sh

# 运行安装脚本,根据脚本提示安装

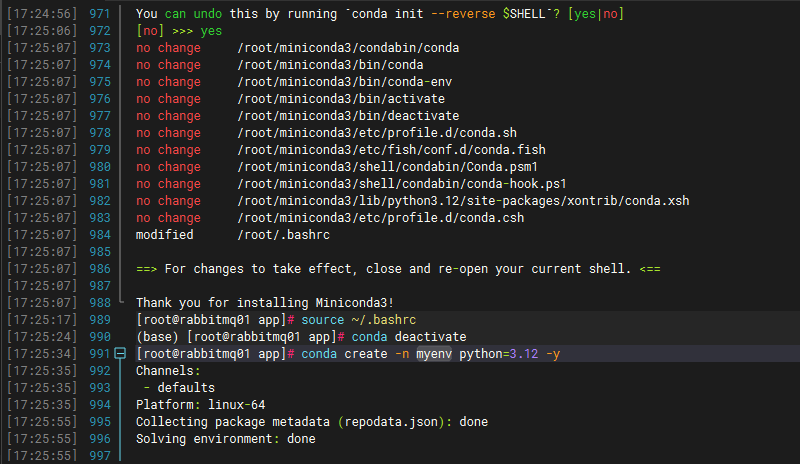
./Miniconda3-latest-Linux-x86\_64.sh

# 生效配置环境变量，默认进入base虚拟环境

source ~/.bashrc

# 退出当前虚拟环境

conda deactivate



# 创建虚拟环境myenv

conda create -n myenv python=3.12 -y

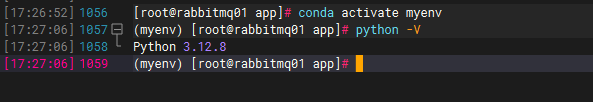
# 激活myenv虚拟环境

conda activate myenv

# 查看Python版本

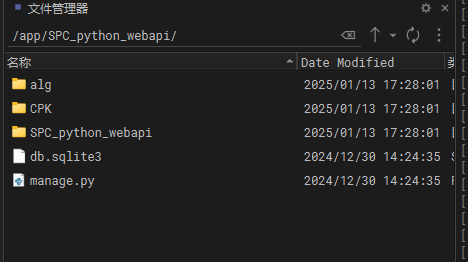
python -V

# 确认Python版本符合你的预期并兼容你的程序



# 上传源码安装依赖

# 上传源码到服务器目录，例如/app



# 进入源码根目录

cd /app/SPC\_python\_webapi/

# 创建依赖文件信息

nano /app/SPC\_python\_webapi/requirements.txt

# 在requirements.txt中填写依赖信息如下

# 注意此版本依赖兼容于RHEL8.8+Python3.12.8，其他版本不保证兼容性，请自行调试

altgraph==0.17.4

asgiref==3.8.1

Django==4.2.13

djangorestframework==3.15.2

numpy==2.2.1

packaging==24.2

pandas==2.2.3

pyinstaller==6.11.1

pyinstaller-hooks-contrib==2024.11

python-dateutil==2.9.0.post0

pytz==2024.2

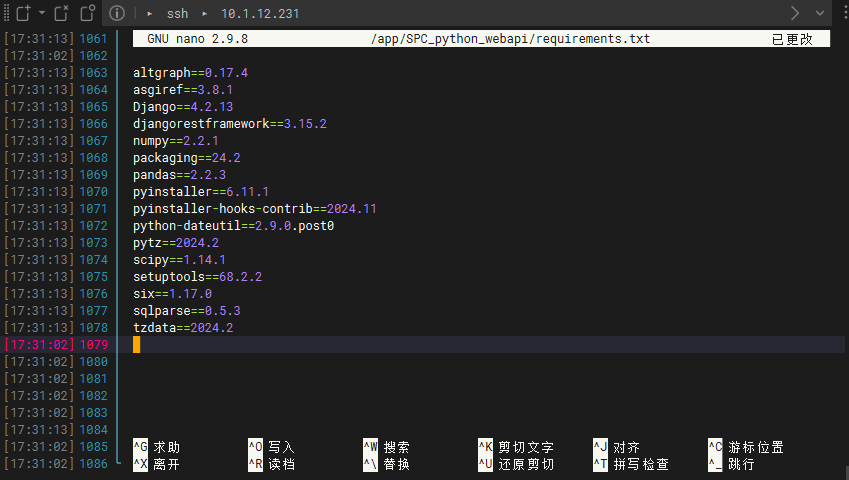
scipy==1.14.1

setuptools==68.2.2

six==1.17.0

sqlparse==0.5.3

tzdata==2024.2



# 在线安装依赖

pip install -r /app/SPC\_python\_webapi/requirements.txt

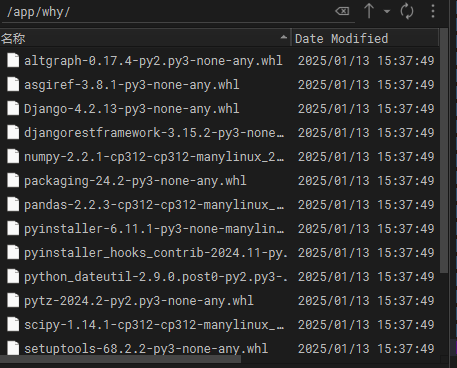
# 查看依赖列表，确认所需依赖都已安装

pip list

# 如果无网络，可以通过离线包安装，我的离线依赖包存放在/app/why

# 离线依赖github下载地址：<https://github.com/lore193/manage>

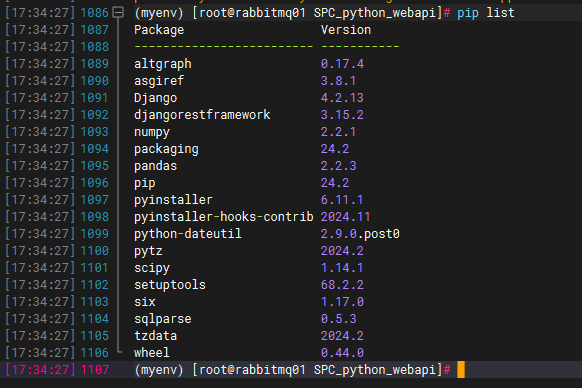
# 上传离线依赖



pip install --no-index --find-links=/app/why -r /app/SPC\_python\_webapi/requirements.txt

# 查看依赖列表，确认所需依赖都已安装

pip list



# 尝试直接运行Python程序

# 删除源码中错误的\_\_init\_\_.py文件

rm /app/SPC\_python\_webapi/CPK/\_\_init\_\_.py

rm /app/SPC\_python\_webapi/CPK/migrations/\_\_init\_\_.py

rm /app/SPC\_python\_webapi/SPC\_python\_webapi/\_\_init\_\_.py

# 重建\_\_init\_\_.py文件

touch /app/SPC\_python\_webapi/CPK/\_\_init\_\_.py

touch /app/SPC\_python\_webapi/CPK/migrations/\_\_init\_\_.py

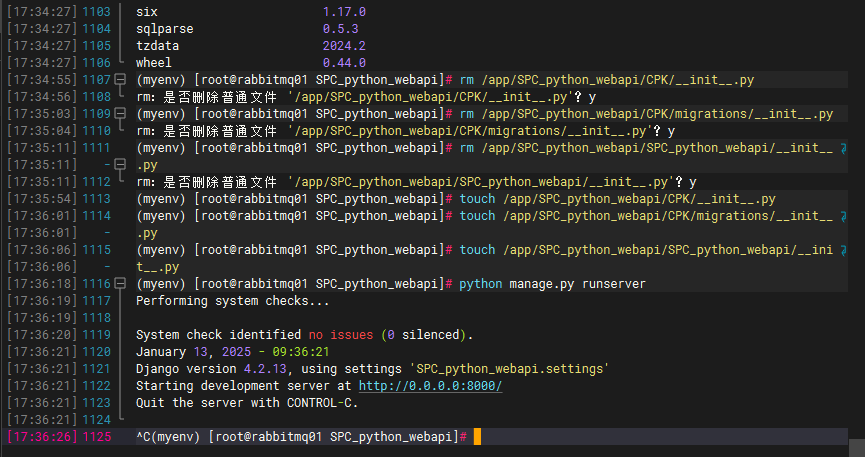
touch /app/SPC\_python\_webapi/SPC\_python\_webapi/\_\_init\_\_.py

# 源码运行Python程序

cd /app/SPC\_python\_webapi/

python manage.py runserver

# 如果所示即程序运行成功



# 打包python程序

# 生成SPEC打包配置文件

pyi-makespec -w manage.py

# 修改manage.spec配置

# 添加配置

a.binaries,

a.datas,

# 添加项目路径

pathex=['/app/SPC\_python\_webapi/'],

# 添加隐含依赖

hiddenimports=[

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'CPK',],

# 添加打包单文件

runtime\_tmpdir=None,

# 删除配置

exclude\_binaries=True,

# 删除coll配置

coll = COLLECT(

exe,

a.binaries,

a.datas,

strip=False,

upx=True,

upx\_exclude=[],

name='manage',

)

# 完整配置如下

# -\*- mode: python ; coding: utf-8 -\*-

a = Analysis(

['manage.py'],

pathex=['/app/SPC\_python\_webapi/'],

binaries=[],

datas=[],

hiddenimports=[

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'CPK',],

hookspath=[],

hooksconfig={},

runtime\_hooks=[],

excludes=[],

noarchive=False,

optimize=0,

)

pyz = PYZ(a.pure)

exe = EXE(

pyz,

a.scripts,

a.binaries,

a.datas,

[],

exclude\_binaries=False,

name='manage',

debug=False,

bootloader\_ignore\_signals=False,

strip=False,

upx=True,

console=False,

disable\_windowed\_traceback=False,

argv\_emulation=False,

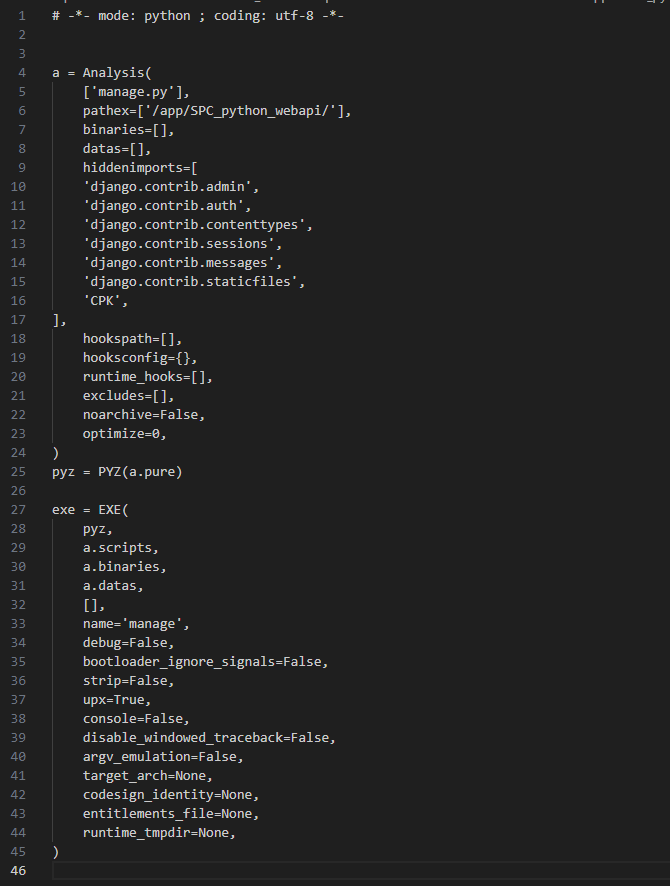
target\_arch=None,

codesign\_identity=None,

entitlements\_file=None,

runtime\_tmpdir=None,

)

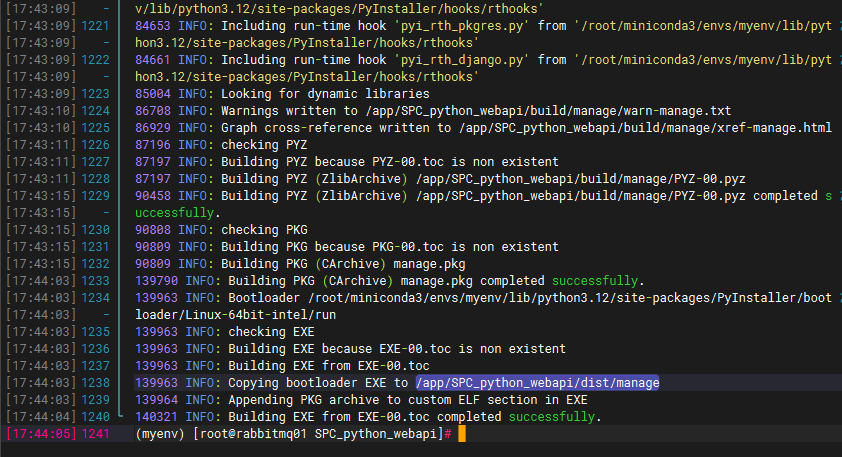


# 打包python程序

pyinstaller manage.spec

# 打包过程中WARNING的not found可以忽略

# 打包成功会在/app/SPC\_python\_webapi/dist/目录下生成一个可执行程序



# 运行打包后的程序

/app/SPC\_python\_webapi/dist/manage runserver 0.0.0.0:8888 --noreload

