Python教程

1. **安装**
2. **Ubuntu**

首先更新包列表并安装构建Python源所需的包：

|  |
| --- |
| sudo apt update  sudo apt install build-essential zlib1g-dev libncurses5-dev libgdbm-dev libnss3-dev libssl-dev libreadline-dev libffi-dev wget |

使用以下wget命令从Python下载页面下载最新版本的源代码：

|  |
| --- |
| wget https://www.python.org/ftp/python/3.7.2/Python-3.7.2.tar.xz |

下载完成后，解压压缩包

tar -xf Python-3.7.2.tar.xz

接下来，导航到Python源目录并运行configure脚本，该脚本将执行大量检查以确保系统上存在所有依赖项：

|  |
| --- |
| cd Python-3.7.2  ./configure --enable-optimizations |

--enable-optimizations选项将通过运行多个测试来优化Python二进制文件，这将使构建过程变慢。

使用make启动Python构建过程：

make -j 8

为了缩短构建时间，请根据处理器修改-j标志。 如果您不知道处理器的核心数，可以通过键入nproc来找到它。 我的系统有8个内核，所以我使用 -j 8 标志。

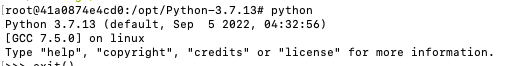
构建完成后，键入以下命令安装Python二进制文件：

sudo make altinstall

不要使用标准的make install，因为它会覆盖默认的系统python3二进制文件。

Python 3.7已安装并可以使用，请输入以下命令进行验证：

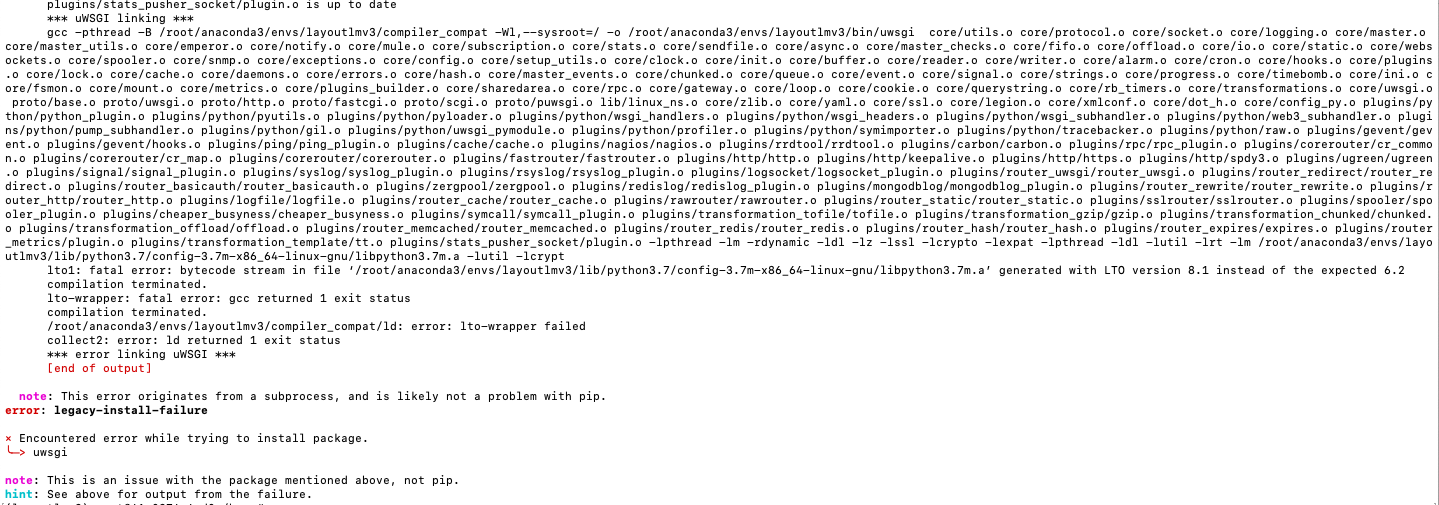
python



1. **Conda安装uwsgi出错**

背景：安装conda后，有个Python服务需要uwsgi，在安装uwsgi的时候始终报错，conda安装其他的时候就很正常。安装命令pip install uwsgi

部分报错信息如下：



环境：系统Ubuntu 18.04 、Python3.7.13、conda4.14.0

原因：gcc版本过高，当时的gcc版本是7.5.0

gcc –v

解决方法：降低gcc版本

1.安装gcc

apt-get install gcc-4.8

3.查看当前版本

ls /usr/bin/gcc\*

4.设置4.8为优先使用版本

sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.8 100

5.重新安装uwsgi

pip install uwsgi

1. **Conda打包转移**

Conda 提供了多种复制项目环境的方法。 创建环境的克隆可以提供定制的基本环境或该环境的快照。spec list 和 conda-pack 可创建特定平台和操作系统的环境副本。 其中 spec list 使用网络来下载环境中特定的软件包，而 conda-pack 可以打包包括软件包二进制文件在内的整个环境，这在带宽不足或没有网络的情况下很有用。 Conda导出 environment.yml 的方式非常适合在不同平台和操作系统之间重新创建环境。spec-list方案生成的环境清单只包含采用conda安装的包，而不包含采用pip安装的包，而environment.yml 则包含环境内的一切包。

conda-pack:本地克隆、复制、转移、复现，无需网络下载

Conda-pack 是一个命令行工具，用于打包 conda 环境，其中包括该环境中安装的软件包的所有二进制文件。 当您想在有限或没有网络访问的系统中重现环境时，此功能很有用。注意，conda-pack 指定平台和操作系统，目标计算机必须具有与源计算机相同的平台和操作系统。

1. **打包**

|  |
| --- |
| # 把 “my\_env”打包，默认以环境名对压缩文件进行命名，保存到此时的运行路径下  conda pack -n my\_env  # 把 “my\_env”打包，以 “out\_name.tar.gz” 进行命名  conda pack -n my\_env -o out\_name.tar.gz  # 把 “my\_env”打包，保存到指定路径下  conda pack -p /explicit/path/to/my\_env |

1. **利用打包好的压缩文件复现环境**

|  |
| --- |
| # Unpack environment into directory `my\_env`  mkdir -p my\_env  tar -xzf my\_env.tar.gz -C my\_env  # Use Python without activating or fixing the prefixes. Most Python  # libraries will work fine, but things that require prefix cleanups  # will fail.  ./my\_env/bin/python  # Activate the environment. This adds `my\_env/bin` to your path  source my\_env/bin/activate  # Run Python from in the environment  (my\_env) $ python  # Cleanup prefixes from in the active environment.  # Note that this command can also be run without activating the environment  # as long as some version of Python is already installed on the machine.  (my\_env) $ conda-unpack |

1. **导出 environment.yml 文件**

生成的文件在conda运行的路径下

conda env export > environment.yml

复现环境

conda env create -f environment.yml

1. **打包docker镜像**

通过上面的导航2可知通过conda打包后的env,然后将env转移到部署服务器,通过解压命令解压到指定目录下

tar -xzf my\_env.tar.gz -C my\_env

就可以通过如下命令使用

source my\_env/bin/activate

如果要打包成docker镜像,则需要在Dockerfile文件里面添加如下内容

|  |
| --- |
| FROM ticket\_keyinfo\_extract\_base:1.0  #如下是解决乱码问题 ENV *LANG* C.UTF-8 ENV *LANGUAGE en*\_US.UTF-8  ADD code/ /opt/code/ ADD venv/layoutlmv3.tar.gz /opt/ #将env加入环境变量,可直接使用env,就不需要再source activate ENV *PYTHON\_PATH*="/opt/layoutlmv3/bin" ENV *LOCAL\_PYTHON*="/opt/layoutlmv3/bin/python3" ENV *PATH*="${*PYTHON\_PATH*}:${*PATH*}"  RUN mkdir /opt/log  # Expose EXPOSE 8098 WORKDIR /opt/code/ ENTRYPOINT /opt/layoutlmv3/bin/uwsgi --ini uwsgi.ini |

1. **Conda**