链滴

如何在矩池云使用 Poetry 管理项目环境

作者：matpool

原文链接：https://ld246.com/article/1668062429476

来源网站：链滴

官网介绍：Poetry is a tool for dependency management and packaging in Python. It allows y u to declare the libraries your project depends on and it will manage (install/update) them for you.

https://python-poetry.org/docs/

Poetry 是 Python 中用于依赖管理和打包的工具。它允许您声明项目所依赖的库，并且它将为您管理安装/更新)它们。

本文将带大家在矩池云上安装并使用 Poetry 管理项目环境，默认你已经在矩池云上租用了一台机器如果不知道如何在矩池云租用服务器，可以查看矩池云新手入门教程。

安装Poetry

pip install poetry

```
  Downloading https://mirror.sjtu.edu.cn/pypi-packages/ac/a3/8ee4f54d5f12e16eeeda6b7df3dfdbda24e6cc572c86ff959a4ce110391b/distlib-0.3.4-py2.py3-none-any.whl (461 kB)
     |████████████████████████████████| 461 kB 3.1 MB/s
Installing collected packages: urllib3, idna, charset-normalizer, requests, pylev, pastel, msgpack, jeepney, cryptography, crashtest, SecretStorage, platformdirs, lockfile, importlib-metadata, filelock, distlib, clikit, ca
checontrol, virtualenv, tomlkit, shellingham, requests-toolbelt, poetry-core, pkginfo, packaging, keyring, html5lib, cleo, cachy, poetry
  Attempting uninstall: packaging
    Found existing installation: packaging 21.3
    Uninstalling packaging-21.3:
      Successfully uninstalled packaging-21.3
Successfully installed SecretStorage-3.3.1 cachecontrol-0.12.10 cachy-0.3.0 charset-normalizer-2.0.12 cleo-0.8.1 clikit-0.6.2 crashtest-0.3.1 cryptography-36.0.2 distlib-0.3.4 filelock-3.6.0 html5lib-1.1 idna-3.3 importlib
-metadata-4.11.3 jeepney-0.8.0 keyring-23.5.0 lockfile-0.12.2 msgpack-1.0.3 packaging-20.9 pastel-0.2.1 pkginfo-1.8.2 platformdirs-2.5.1 poetry-1.1.13 poetry-core-1.0.8 pylev-1.4.0 requests-2.27.1 requests-toolbelt-0.9.1 s
hellingham-1.4.0 tomlkit-0.10.1 urllib3-1.26.9 virtualenv-20.14.1
```

创建一个poetry项目目录

首先我们进入 /home 目录中，然后执行 poetry new指令，即可新建一个poetry项目，默认包含下几部分。

```
cd /home
poetry new my-project
cd my-project
tree
```

```
(myconda) root@d4ac2b5db392:/# cd /home
(myconda) root@d4ac2b5db392:/home# poetry new my-project
Created package my_project in my-project
(myconda) root@d4ac2b5db392:/home# cd my-project/
(myconda) root@d4ac2b5db392:/home/my-project# tree
.
├── README.rst
├── my_project
│   └── __init__.py
├── pyproject.toml
└── tests
    ├── __init__.py
    └── test_my_project.py

2 directories, 5 files
```

```
my-project
├── README.rst   # 项目说明
├── my_project   # 项目文件目录
│   └── __init__.py
├── pyproject.toml  # poetry配置文件 重要
└── tests   # 测试文件
    ├── __init__.py
    └── test_my_project.py
```

上面目录结构中，最重要的是 pyproject.toml，里面默认包含了下面内容：

```
[tool.poetry]
```

```
name = "my-project"
version = "0.1.0"
description = ""
authors = ["Your Name <you@example.com>"]

[tool.poetry.dependencies]
python = "^3.8"

[tool.poetry.dev-dependencies]
pytest = "^5.2"

[build-system]
requires = ["poetry-core>=1.0.0"]
build-backend = "poetry.core.masonry.api"
```

- tool.poetry 记录项目名称、版本、基本描述和作者
- tool.poetry.dependencies 记录项目依赖工具和版本，比如python
- tool.poetry.dev-dependencies 记录项目依赖的python包
- build-system 记录Poetry环境构建工具

poetry创建、进入虚拟环境

- 创建虚拟环境 poetry env use 本地python解释器路径

```
(myconda) root@c6854bdc088b:/home/my-project# poetry env use /root/miniconda3/envs/
yconda/bin/python
Creating virtualenv my-project-zjY4rh4o-py3.8 in /root/.cache/pypoetry/virtualenvs
Using virtualenv: /root/.cache/pypoetry/virtualenvs/my-project-zjY4rh4o-py3.8
```

- 查看虚拟环境基本信息

```
(myconda) root@c6854bdc088b:/home/my-project# poetry env info

Virtualenv
Python:         3.8.2
Implementation: CPython
Path:           /root/.cache/pypoetry/virtualenvs/my-project-zjY4rh4o-py3.8
Valid:          True

System
Platform: linux
OS:       posix
Python:   /root/miniconda3/envs/myconda
```

- 进入虚拟环境

```
(myconda) root@c6854bdc088b:/home/my-project# poetry shell
Spawning shell within /root/.cache/pypoetry/virtualenvs/my-project-zjY4rh4o-py3.8
sh-4.4# . /root/.cache/pypoetry/virtualenvs/my-project-zjY4rh4o-py3.8/bin/activate
(my-project-zjY4rh4o-py3.8) sh-4.4# pip list
Package    Version
---------- -------
```

```
pip        22.0.4
setuptools 62.1.0
wheel      0.37.1
(my-project-zjY4rh4o-py3.8) sh-4.4#
```

poetry常用指令

- 安装第三方包

进入虚拟环境后，我们可以直接pip innstall 包名 安装自己需要的第三方包，不过这样安装包不会记到pyproject.toml中。

```
# poetry shell 进入虚拟环境后，可以直接pip install 包名安装
(my-project-zjY4rh4o-py3.8) sh-4.4# pip install pandas
Looking in indexes: https://mirrors.aliyun.com/pypi/simple/
Collecting pandas
  Downloading https://mirrors.aliyun.com/pypi/packages/12/07/e82b5de
...
Successfully installed numpy-1.22.3 pandas-1.4.2 python-dateutil-2.8.2 pytz-2022.1 six-1.16.0
(my-project-zjY4rh4o-py3.8) sh-4.4# pip list
Package         Version
--------------- -------
numpy           1.22.3
pandas          1.4.2
pip             22.0.4
python-dateutil 2.8.2
pytz            2022.1
setuptools      62.1.0
six             1.16.0
wheel           0.37.1
```

不进入虚拟环境，我们可以通过poetry add 包名来安装，同时会生成一个 poetry.lock文件，记录安包相关依赖。

```
# 不进入虚拟环境
(myconda) root@cd90f1a3f442:/home/my-project# poetry add pendulum@latest
Using version ^2.1.2 for pendulum

Updating dependencies
Resolving dependencies... (59.4s)

Writing lock file

Package operations: 10 installs, 0 updates, 0 removals

  • Installing pyparsing (3.0.8)
  • Installing attrs (21.4.0)
  • Installing more-itertools (8.12.0)
  • Installing packaging (21.3)
  • Installing pluggy (0.13.1)
  • Installing py (1.11.0)
  • Installing pytzdata (2020.1)
  • Installing wcwidth (0.2.5)
  • Installing pendulum (2.1.2)
```

```
  • Installing pytest (5.4.3)
```

● 查看安装的包依赖关系 poetry show -t

```
(myconda) root@cd90f1a3f442:/home/my-project# poetry show -t
pendulum 2.1.2 Python datetimes made easy
├── python-dateutil >=2.6,<3.0
│   └── six >=1.5
└── pytzdata >=2020.1
pytest 5.4.3 pytest: simple powerful testing with Python
├── atomicwrites >=1.0
├── attrs >=17.4.0
├── colorama *
├── more-itertools >=4.0.0
├── packaging *
│   └── pyparsing >=2.0.2,<3.0.5 || >3.0.5
├── pluggy >=0.12,<1.0
├── py >=1.5.0
└── wcwidth *
```

● 移除安装的第三方包 poetry remove 包名，移除指定第三方包的同时会卸载相关依赖包。

```
(myconda) root@cd90f1a3f442:/home/my-project# poetry remove pendulum
Updating dependencies
Resolving dependencies... (0.1s)

Writing lock file

Package operations: 0 installs, 0 updates, 4 removals

  • Removing pendulum (2.1.2)
  • Removing python-dateutil (2.8.2)
  • Removing pytzdata (2020.1)
  • Removing six (1.16.0)
```

● 导出项目依赖

```
poetry export -f requirements.txt --output requirements.txt
```

常用参数：

```
--format (-f): 导出文件格式，目前仅支持requirements.txt
--output (-o): 导出文件名称
```

● 查看poetry全局配置 poetry config --list

```
(myconda) root@cd90f1a3f442:/home/my-project# poetry config --list
# poetry缓存目录
cache-dir = "/root/.cache/pypoetry"
experimental.new-installer = true
installer.parallel = true
# 默认 true，进行poetry add/install 时如果没有虚拟环境，就创建一个，如果为 false，没有虚拟
境就安装到系统环境中
virtualenvs.create = true
```

```
# 在项目根目录创建虚拟环境
virtualenvs.in-project = null
# 虚拟环境目录
virtualenvs.path = "{cache-dir}/virtualenvs"  # /root/.cache/pypoetry/virtualenvs
```

- 设置poetry全局配置值 poetry config virtualenvs.create false --local

```
(myconda) root@cd90f1a3f442:/home/my-project# poetry config virtualenvs.create false --lo
al
(myconda) root@cd90f1a3f442:/home/my-project# poetry config --list
cache-dir = "/root/.cache/pypoetry"
experimental.new-installer = true
installer.parallel = true
virtualenvs.create = false
virtualenvs.in-project = null
virtualenvs.path = "{cache-dir}/virtualenvs"  # /root/.cache/pypoetry/virtualenvs
```

执行后，会在项目目录下生成一个 poetry.toml文件，记录了修改的配置名和对应的值，--local表示是修改本项目的配置。

更多相关指令

```
poetry install  # 通过项目目录中的pyproject.toml安装相关依赖
poetry check  # 检查依赖关系
poetry search requests  # 查找可用的相关包信息
poetry lock # 更新 pyproject.toml 中依赖版本，加--no-update只更lock新文件，不更新包版本
poetry version  # 查看poetry版本
...
```

更多使用方法，可以阅读学习官方文档：https://python-poetry.org/docs/cli