

Текстовая расшифровка видео:

НЕМНОГО О ПОЭЗИИ (POETRY)

План:

- Poetry pyproject.toml.

Poetry pyproject.toml

Основная идея Poetry: не нужно писать Setup.py, потому что Poetry сам будет все обрабатывать, вместо этого необходимо использовать файл pyproject.toml.

Пример:

```
1. [tool.poetry]
2. name = "slurm-example"
3. version = "0.1.0"
4. description = ""
5. authors = ["Nikolay Markov <markov@alignedresearch.com>"]
6. readme = "README.md"
7. packages = [{include = "slurm_example"}]
8.
9. [tool.poetry.dependencies]
10. python = "^3.10"
11. requests = "^2.29.0"
12. pandas = ">=2.0"
13.
14. [tool.poetry.group.test.dependencies]
15. pytest = "^7.3.1"
16.
17. [build-system]
18. requires = ["poetry-core"]
19. build-backend = "poetry.core.masonry.api"
```



У нас есть декларативное описание нашего проекта, также здесь указаны различные зависимости, которые мы можем использовать.

Можно создать каталог проекта, например, «my_project» и зайти в него. Начать проект стоит с создания poetry init.

Poetry задаст ряд вопросов:

- Название проекта;
- Версия;
- Описание;
- Автор;
- Лицензия;
- Совместимая версия Python и т.д:

```
meow-nofer@pikachu ~/Experiments/slurm mkdir my_project
meow-nofer@pikachu ~/Experiments/slurm cd my_project
meow-nofer@pikachu ~/Experiments/slurm/my_project poetry init

This command will guide you through creating your pyproject.toml config.

Package name [my_project]:
Version [0.1.0]:
Description []:
Author [Nikolay Markov <markov@alignedresearch.com>, n to skip]:
License []:
Compatible Python versions [^3.11]:

Would you like to define your main dependencies interactively? (yes/no) [yes] no
Would you like to define your development dependencies interactively? (yes/no) [yes] no
```

Пропустим описание некоторых зависимостей и сгенерируем файл:

```
pyproject.toml
" Press ? for help
.. (up a dir)
</slurm/my_project/
  o pyproject.toml

1 [[tool.poetry]]
2 name = "my-project"
3 version = "0.1.0"
4 description = ""
5 authors = ["Nikolay Markov <markov@alignedresearch.com>"]
6 readme = "README.md"
7 packages = [{"include = "my_project"}]
8
9 [tool.poetry.dependencies]
10 python = "^3.11"
11
12
13 [build-system]
14 requires = ["poetry-core"]
15 build-backend = "poetry.core.masonry.api"
```

Далее, вместо «pip install» говорим «poetry add requests». Poetry создает виртуальное окружение для этого проекта и ставит версию «requests»:

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

Do you confirm generation? (yes/no) [yes]
meow-nofer@pikachu ~/Experiments/slurm/my_project vim
meow-nofer@pikachu ~/Experiments/slurm/my_project poetry add requests
Creating virtualenv my-project-euUU9nZM-py3.11 in /home/meow-nofer/.cache/pypoetry/virtualenvs
Using version ^2.30.0 for requests

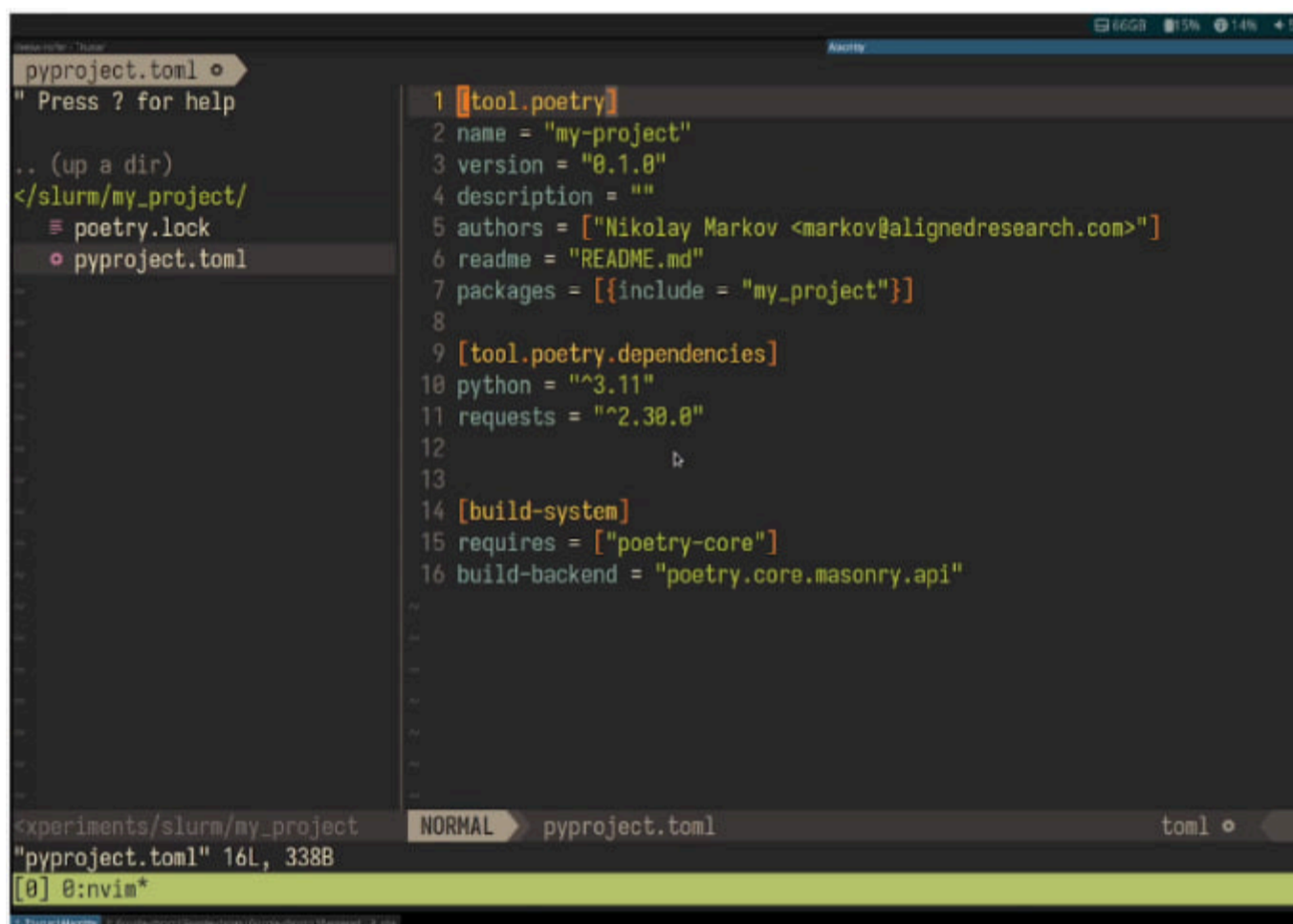
Updating dependencies
Resolving dependencies... (1.0s)

Writing lock file

Package operations: 5 installs, 0 updates, 0 removals

  • Installing certifi (2023.5.7)
  • Installing charset-normalizer (3.1.0)
  • Installing idna (3.4)
  • Installing urllib3 (2.0.2)
  • Installing requests (2.30.0)
meow-nofer@pikachu ~/Experiments/slurm/my_project
[0] 0:zsh*
```

Помимо этого, Poetry автоматически обновляет pyproject.toml:

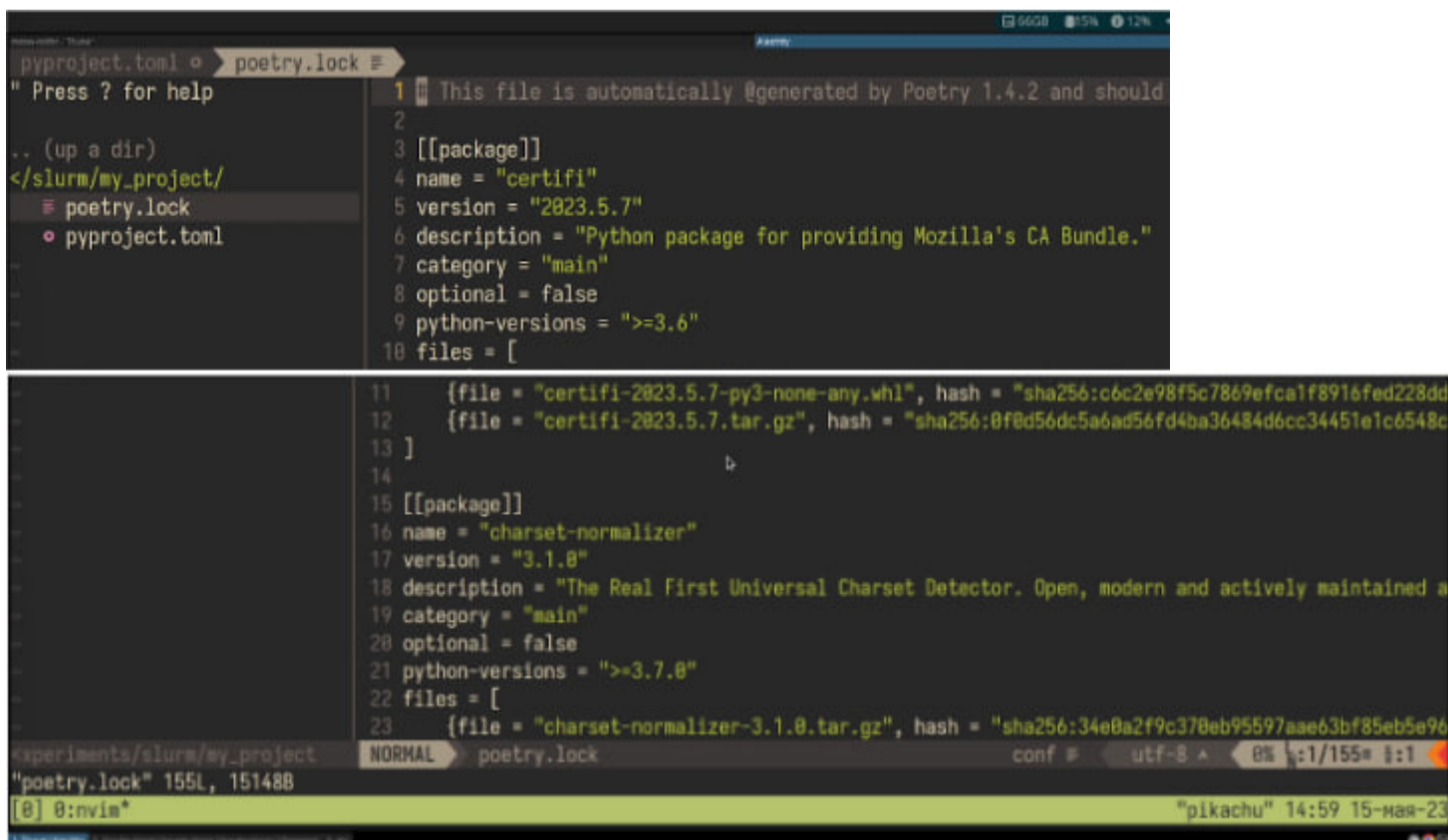


```
pyproject.toml
" Press ? for help
.. (up a dir)
</slurm/my_project/
  poetry.lock
  pyproject.toml

1 [[tool.poetry]]
2 name = "my-project"
3 version = "0.1.0"
4 description = ""
5 authors = ["Nikolay Markov <markov@alignedresearch.com>"]
6 readme = "README.md"
7 packages = [{include = "my_project"}]
8
9 [[tool.poetry.dependencies]]
10 python = "^3.11"
11 requests = "^2.30.0"
12
13
14 [[build-system]]
15 requires = ["poetry-core"]
16 build-backend = "poetry.core.masonry.api"

experiments/slurm/my_project NORMAL pyproject.toml toml
"pyproject.toml" 16L, 338B
[0] 0:nvim*
```

и генерирует еще один poetry.lock – файл, замораживающий зависимости:



```
pyproject.toml poetry.lock
" Press ? for help
.. (up a dir)
</slurm/my_project/
  poetry.lock
  pyproject.toml

1 This file is automatically @generated by Poetry 1.4.2 and should
2
3 [[package]]
4 name = "certifi"
5 version = "2023.5.7"
6 description = "Python package for providing Mozilla's CA Bundle."
7 category = "main"
8 optional = false
9 python-versions = ">=3.6"
10 files = [
11   {file = "certifi-2023.5.7-py3-none-any.whl", hash = "sha256:c6c2e98f5c7869efca1f8916fed228dd
12   {file = "certifi-2023.5.7.tar.gz", hash = "sha256:8f8d56dc5a6ad56fd4ba36484d6cc34451e1c6548c
13 ]
14
15 [[package]]
16 name = "charset-normalizer"
17 version = "3.1.0"
18 description = "The Real First Universal Charset Detector. Open, modern and actively maintained a
19 category = "main"
20 optional = false
21 python-versions = ">=3.7.0"
22 files = [
23   {file = "charset-normalizer-3.1.0.tar.gz", hash = "sha256:34e0a2f9c378eb95597aae63bf85eb5e96

experiments/slurm/my_project NORMAL poetry.lock conf utf-8 8% 1/155 1:1
"poetry.lock" 155L, 15148B
[0] 0:nvim* "pikachu" 14:59 15-мая-23
```

Как работать?

Говорим «poetry shell», так Poetry автоматически запустит shell внутри virtualenvs. После активации мы можем посмотреть результат pip freeze. Также мы можем увидеть все зависимости и продолжить с ними

работать:

```
Using version ^2.30.0 for requests
Updating dependencies
Resolving dependencies... (1.0s)
Writing lock file
Package operations: 5 installs, 0 updates, 0 removals
  • Installing certifi (2023.5.7)
  • Installing charset-normalizer (3.1.0)
  • Installing idna (3.4)
  • Installing urllib3 (2.0.2)
  • Installing requests (2.30.0)

meow-nofer@pikachu ~/Experiments/slurm/my_project
meow-nofer@pikachu ~/Experiments/slurm/my_project poetry shell
Spawning shell within /home/meow-nofer/.cache/pypoetry/virtualenvs/my-project-euU9nZM-py3.11
meow-nofer@pikachu ~/Experiments/slurm/my_project emulate bash -c './home/meow-nofer/.cache/pypoetry/virtualenvs/my-project-euU9nZM-py3.11/bin/activate'
(my-project-euU9nZM-py3.11) meow-nofer@pikachu ~/Experiments/slurm/my_project pip freeze
certifi==2023.5.7
charset-normalizer==3.1.0
idna==3.4
requests==2.30.0
urllib3==2.0.2
(my-project-euU9nZM-py3.11) meow-nofer@pikachu ~/Experiments/slurm/my_project
[0] 8:python* "pikachu" 15:00 15-мая-23
```

Мы можем управлять зависимостями в группах.

Когда мы работаем с реальным проектом, мы знаем, что у него не одна группа зависимостей, а еще и отдельные зависимости для запуска тестов, сборки документации, линтеры для локальной разработки и т.д. Poetry позволяет создавать разные группы этих зависимостей.

Недостаток Poetry – принципиальное отсутствие возможности иметь конфликтующие зависимости даже в разных группах.

Часто интегрируется с ruenv и позволяет создавать динамические virtualenv с конкретной версией Python.

Poetry позволяет собирать Python-пакеты:

```
meow-nofer@pikachu ~/Experiments/slurm/my_project poetry build
Building my-project (0.1.0)
/home/meow-nofer/Experiments/slurm/my_project/my_project does not contain any element
meow-nofer@pikachu ~/Experiments/slurm/my_project
[0] 8:python* "pikachu" 15:04 15-мая-23
```

Пометка: наш пример пока не пакет, поэтому появилась следующая строка, сообщающая о том, что собирать пока нечего.

Вставим команду «poetry build». Он собирает нам пакет. Если сказать «poetry publish» и он зальет в репозиторий, какой нам будет необходим.

Как вам урок?



Далее >

