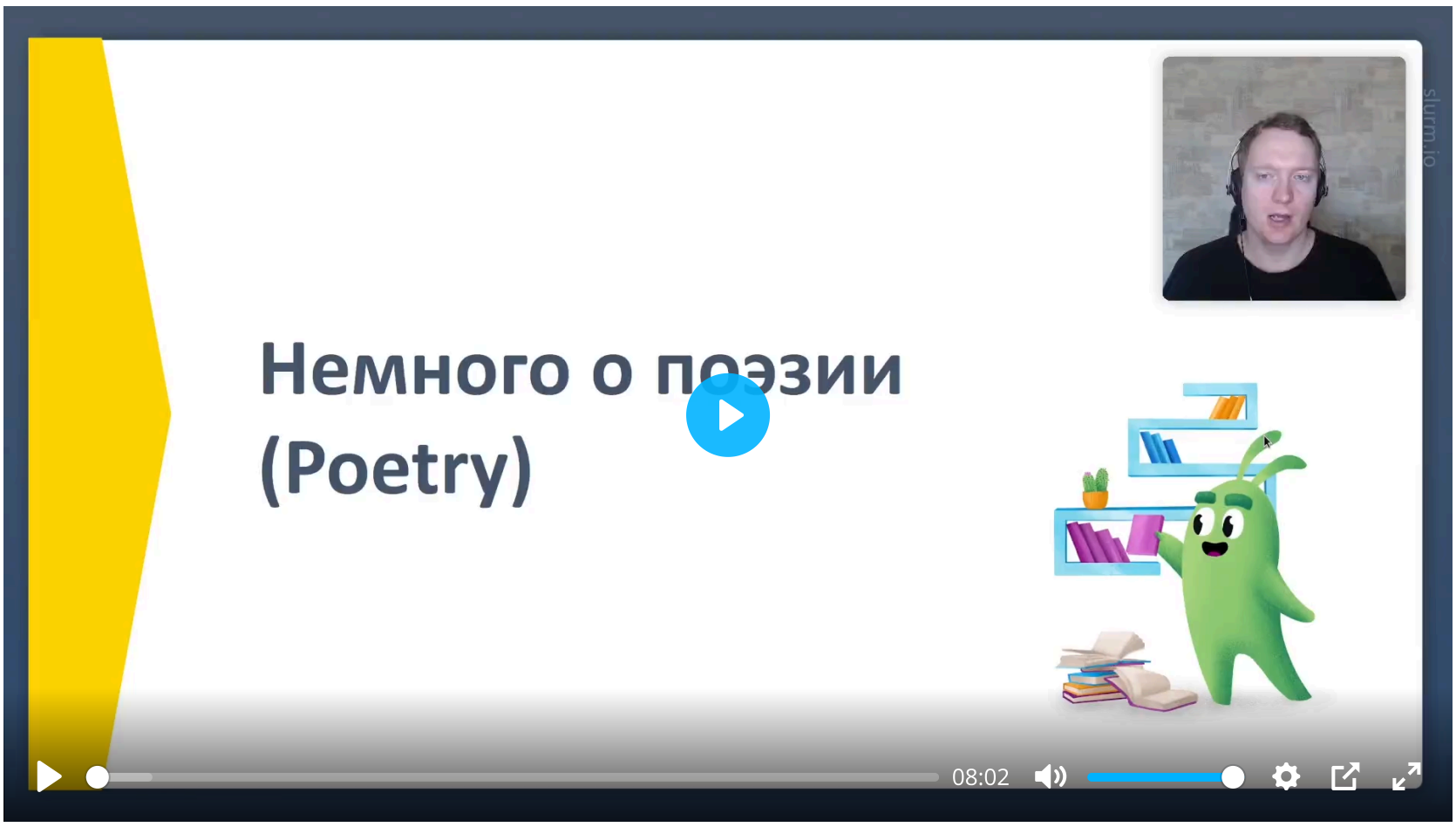


Текстовая расшифровка видео:

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:

```
[tool.poetry]
name = "slurm-example"
version = "0.1.0"
description = ""
authors = ["Nikolay Markov <markov@alignedresearch.com>"]
readme = "README.md"
packages = [{include = "slurm_example"}]

[tool.poetry.dependencies]
python = "^3.10"
requests = "^2.29.0"
pandas = ">=2.0"

[tool.poetry.group.test.dependencies]
pytest = "^7.3.1"

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

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

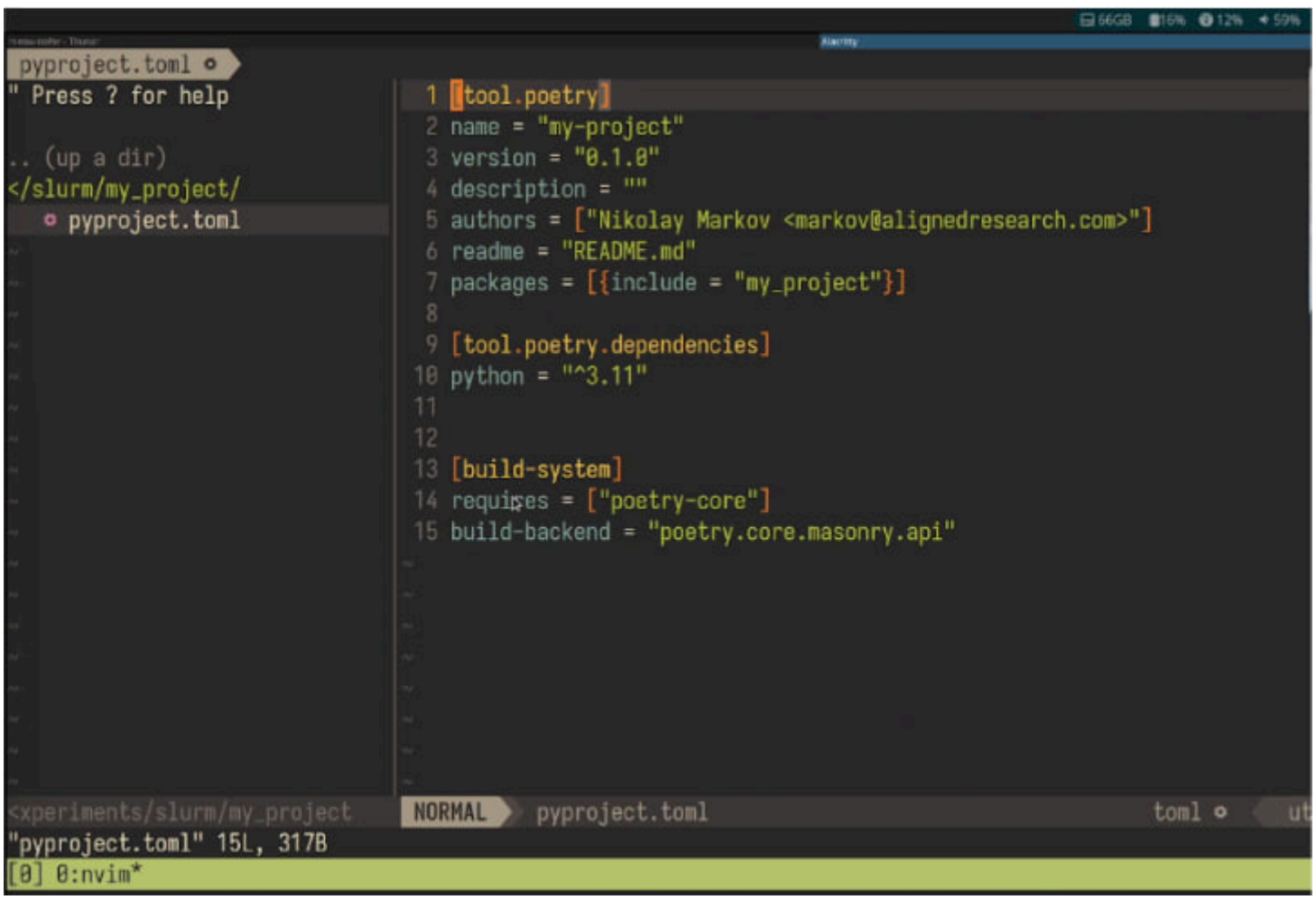
Можно создать каталог проекта, например, «my_project» и зайти в него. Начать проект стоит с создания poetry init.

Poetry задаст ряд вопросов:

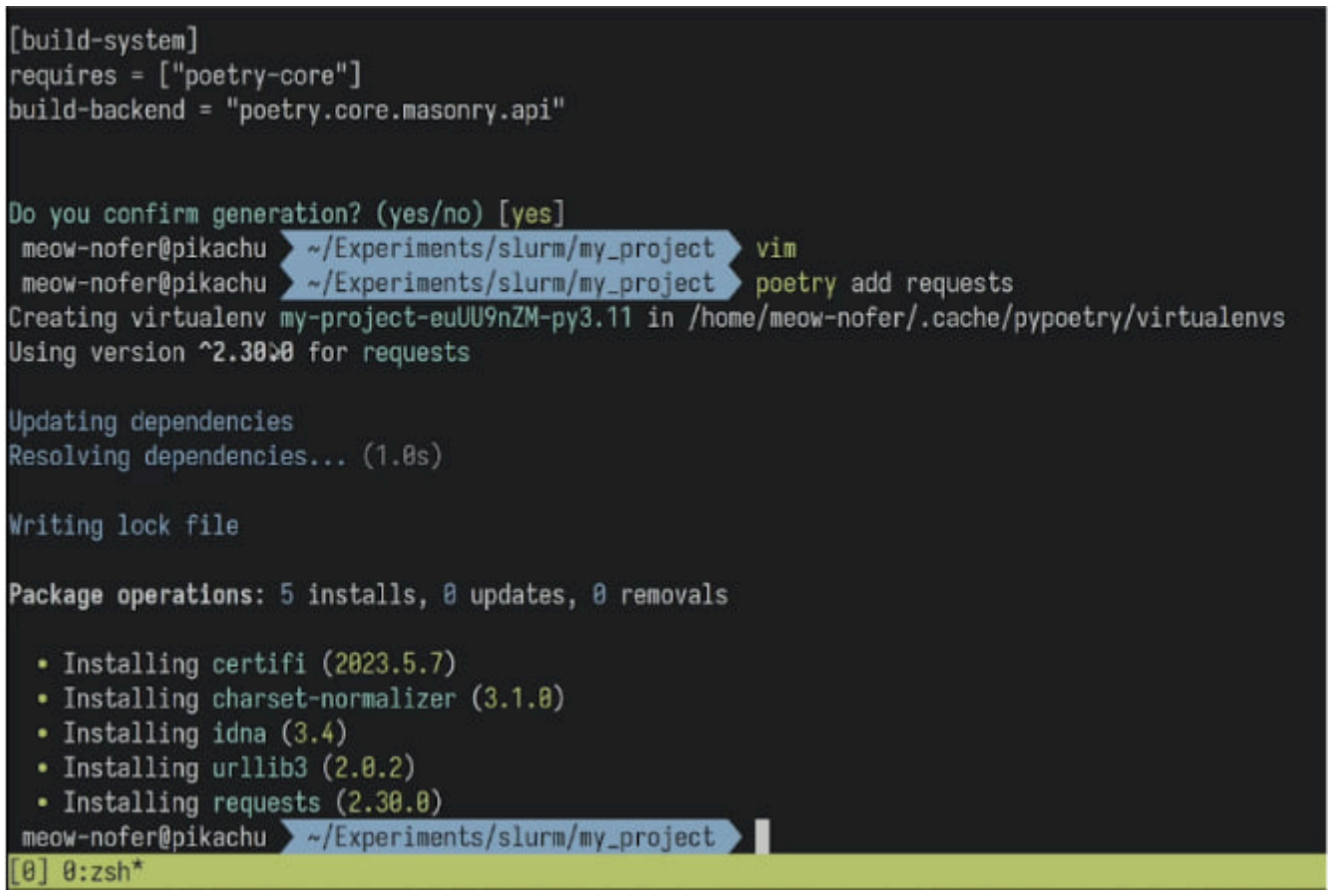
- Название проекта;
- Версия;
- Описание;
- Автор;
- Лицензия;
- Совместимая версия Python и т.д:

```
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
```

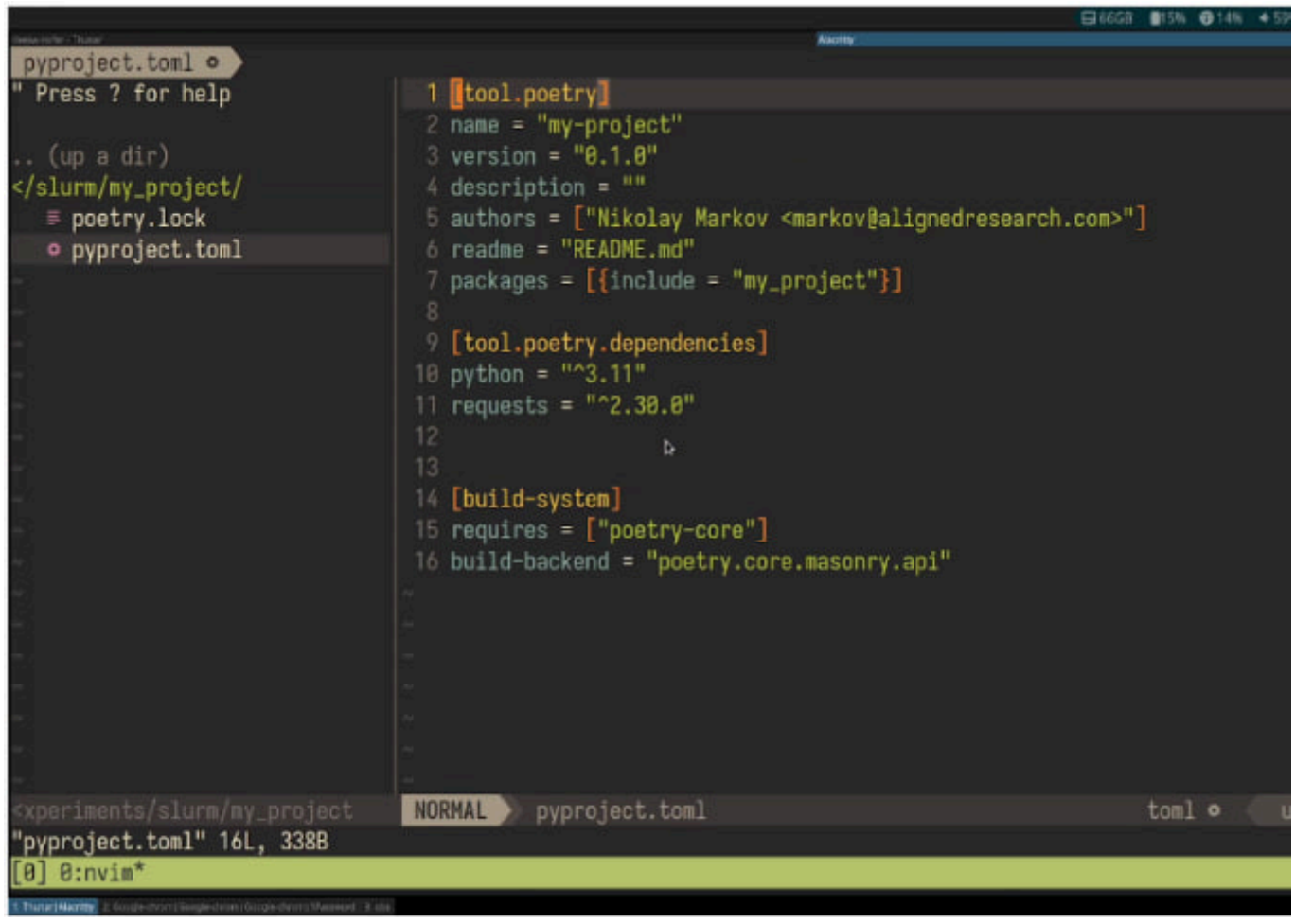
Пропустим описание некоторых зависимостей и сгенерируем файл:



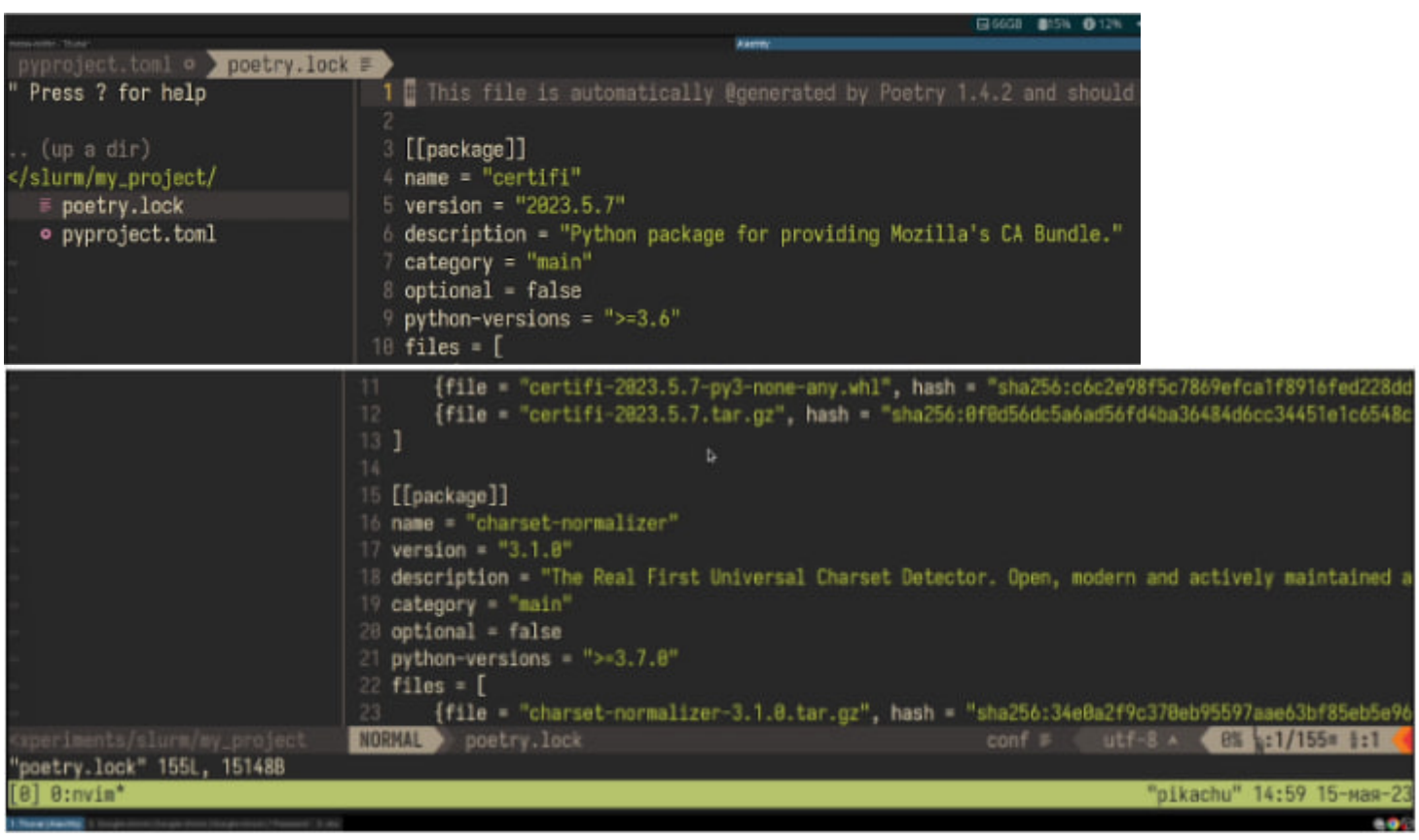
Далее, вместо «pip install» говорим «poetry add requests». Poetry создает виртуальное окружение для этого проекта и ставит версию «requests»:



Помимо этого, Poetry автоматически обновляет pyproject.toml:

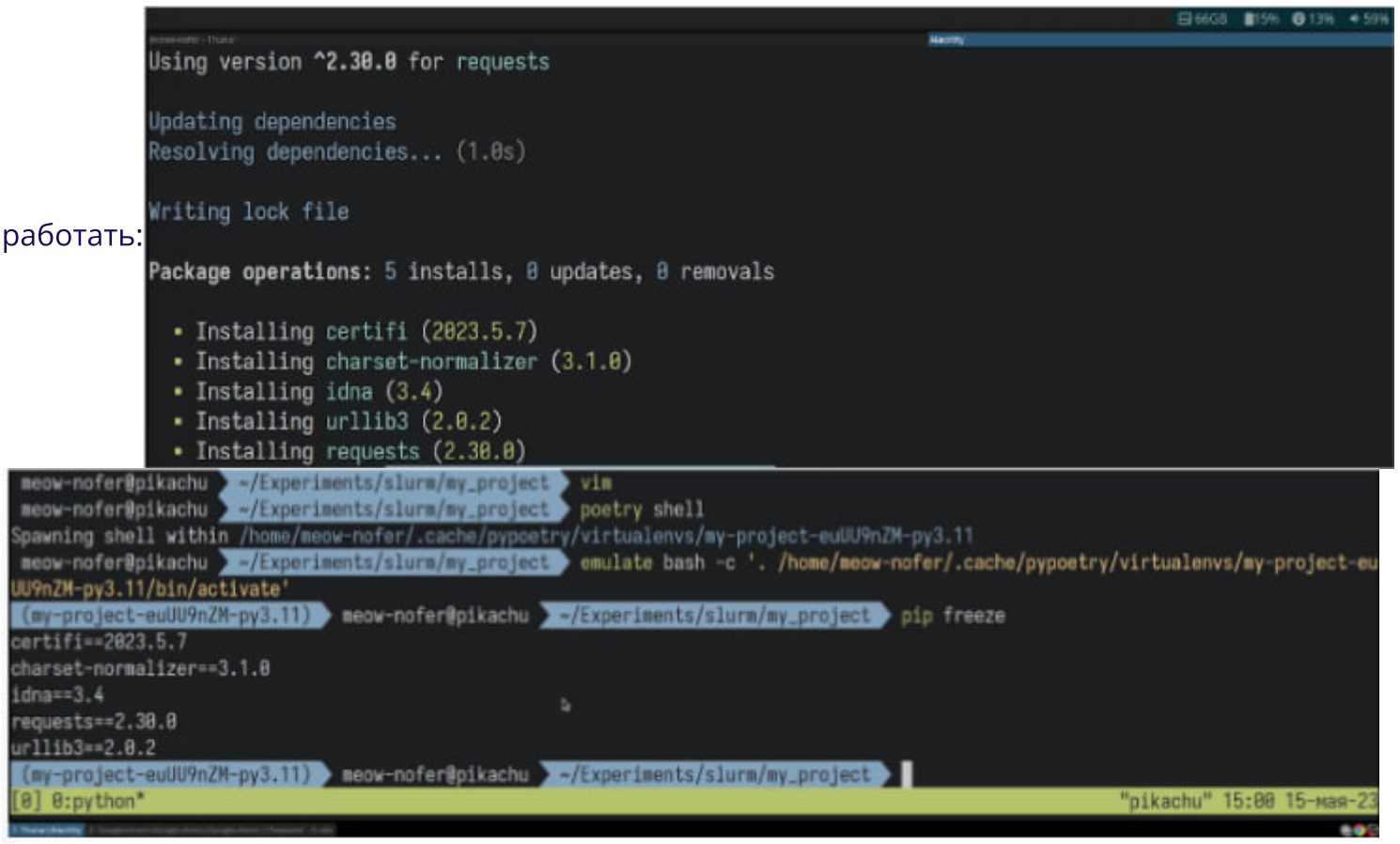


и генерирует еще один poetry.lock – файл, замораживающий зависимости:



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:



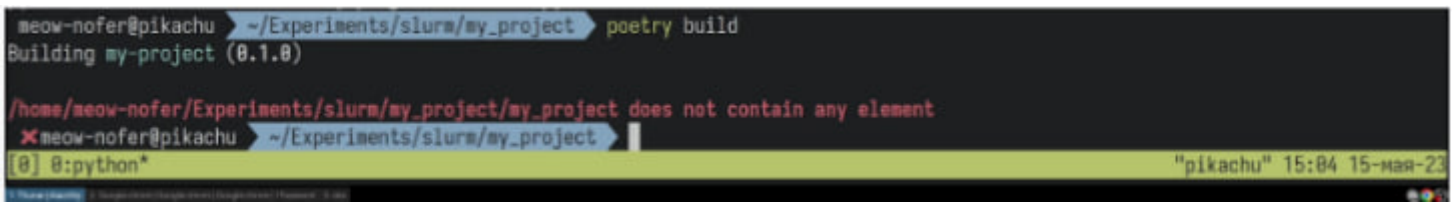
Мы можем управлять зависимостями в группах.

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 pyenv и позволяет создавать динамические virtualenv с конкретной версией Python.

Poetry позволяет собирать Python-пакеты:



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