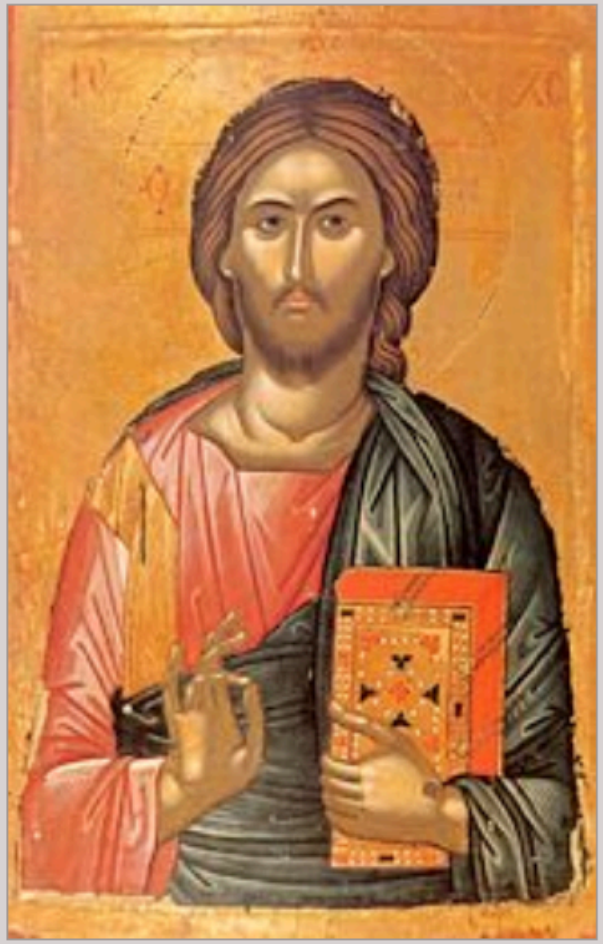
Build-Benedictions

```
$ buildben init_proj
```

Managing Multiple (Python) Projects & Dependencies

Dr. rer. nat. Martin Kuric

Academy of Sciences Göttingen · Germania Sacra / HisQu



Icon of *Jesus Christ Pantokrator* by Theophanes the Cretan. His right hand is raised in benediction.

From Wikipedia:

*"A **benediction** (Latin: bene, 'well' + dicere, 'to speak') is a short **invocation** for divine help, blessing and guidance [...]."*

*"**Invocation** is the act of calling upon a deity, spirit, or supernatural force, typically through prayer, ritual, or **spoken formula**, to seek guidance, assistance, or presence."*

Build-Benedictions: Main Features

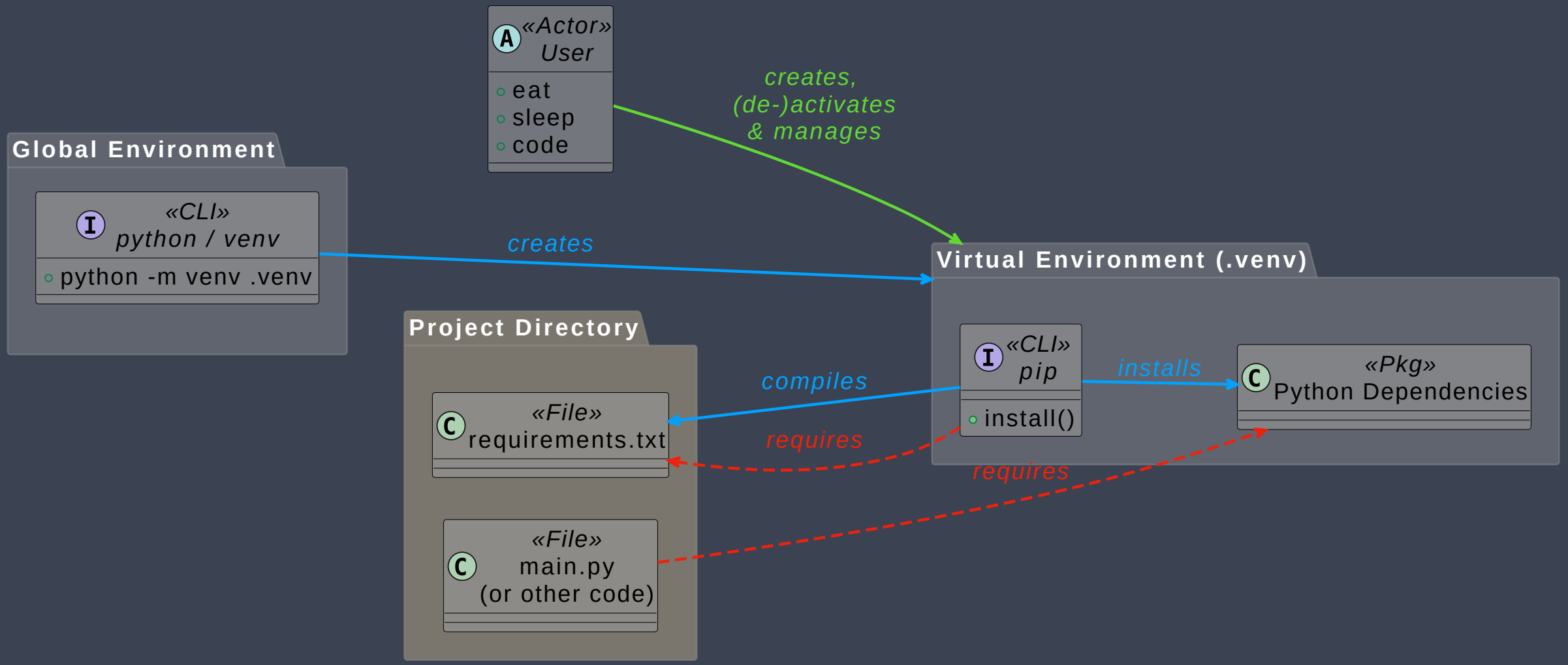
Standardize setups with template scaffolds:

- `buildben init_proj`: Create a new **project** in `/src`-layout.
- `buildben add_experiment`: Add a new **experiment** to a project.
- `buildben init_database`: Create a new central **database**. *(WIP)*

Integrate popular CLI-tools:

- `direnv`: Automate virtual **environments** & variables.
- `pip-tools`: Automate **dependency** management.
- `just`: Summarizing tasks into **one-liners** (upgrade, test, etc.).
- `docker`: **Snapshot** current state of project.

Minimal Python Project: `requirements.txt` & `.venv`



Minimal Python Project: Dependency Management

```
pip freeze > requirements.txt  # Write all dependencies installed in current .venv
```

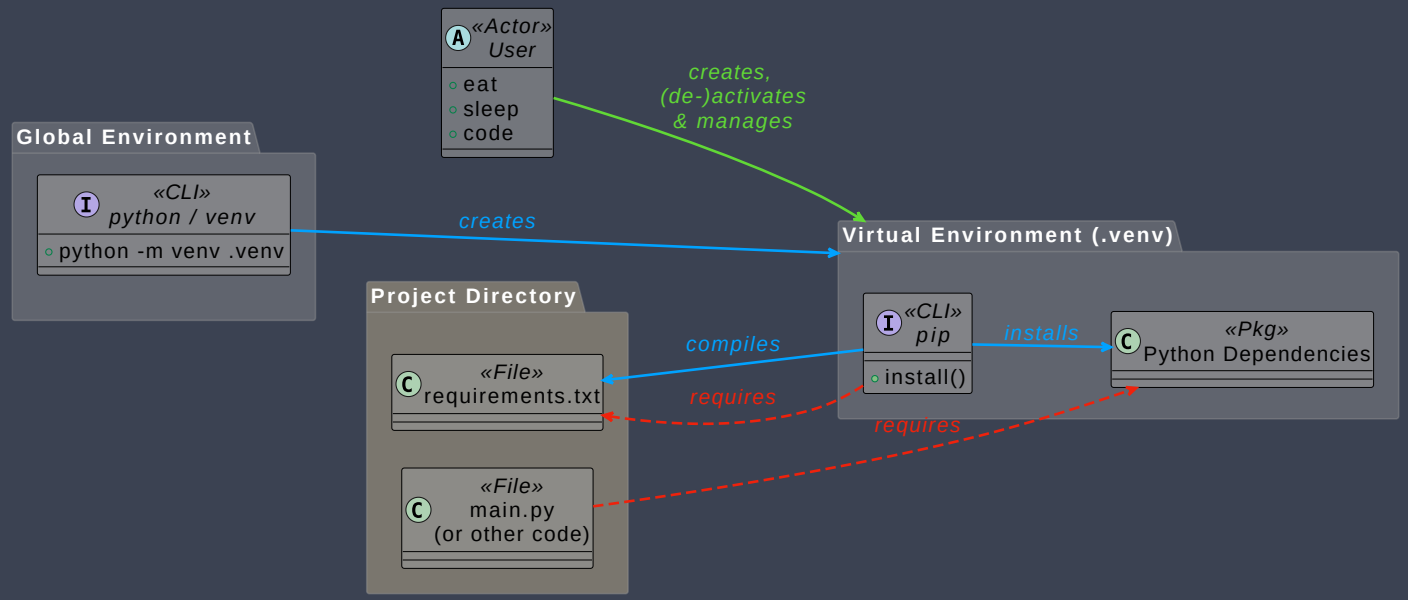
```
# Inside requirements.txt:
asttokens==3.0.0
build==1.2.2.post1
click==8.2.1
comm==0.2.2
debugpy==1.8.14
decorator==5.2.1
executing==2.2.0
ipykernel==6.29.5
ipython==9.4.0
ipython_pygments_lexers==1.1.1
jedi==0.19.2
...
```

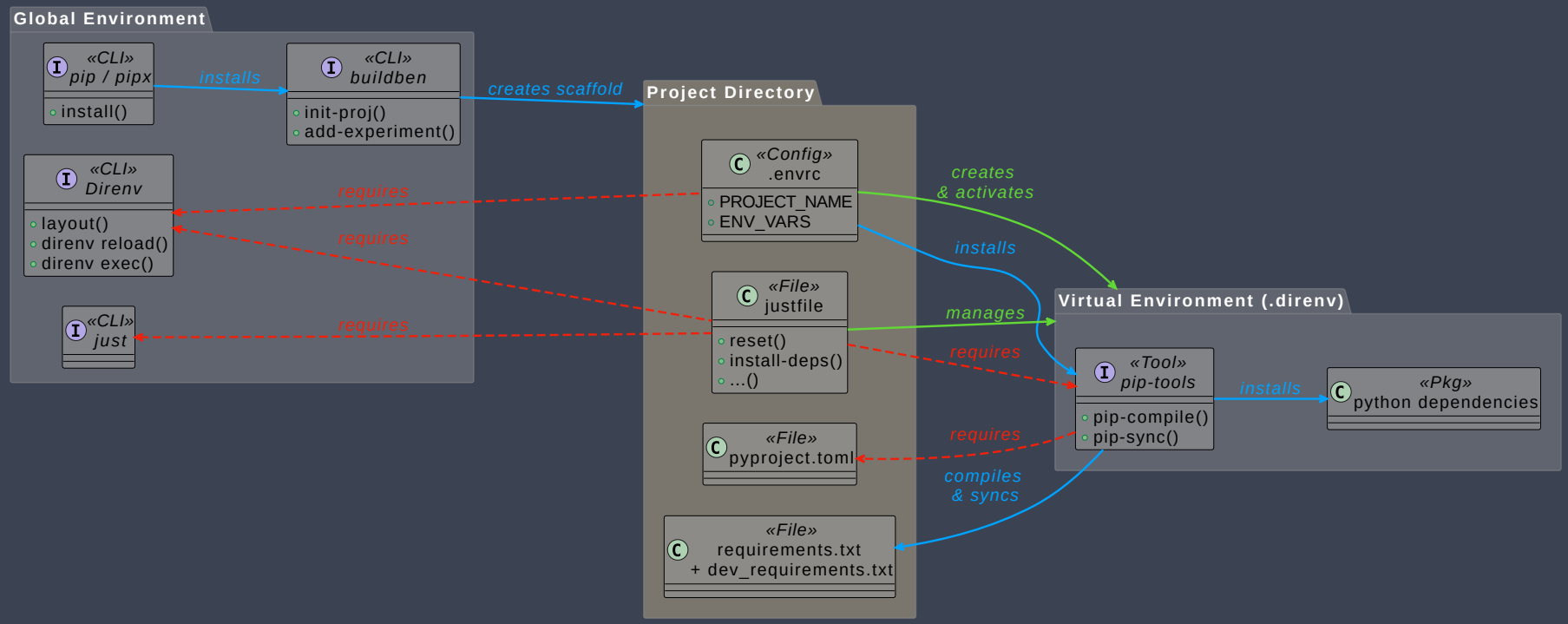
Minimal Python Project: Setup

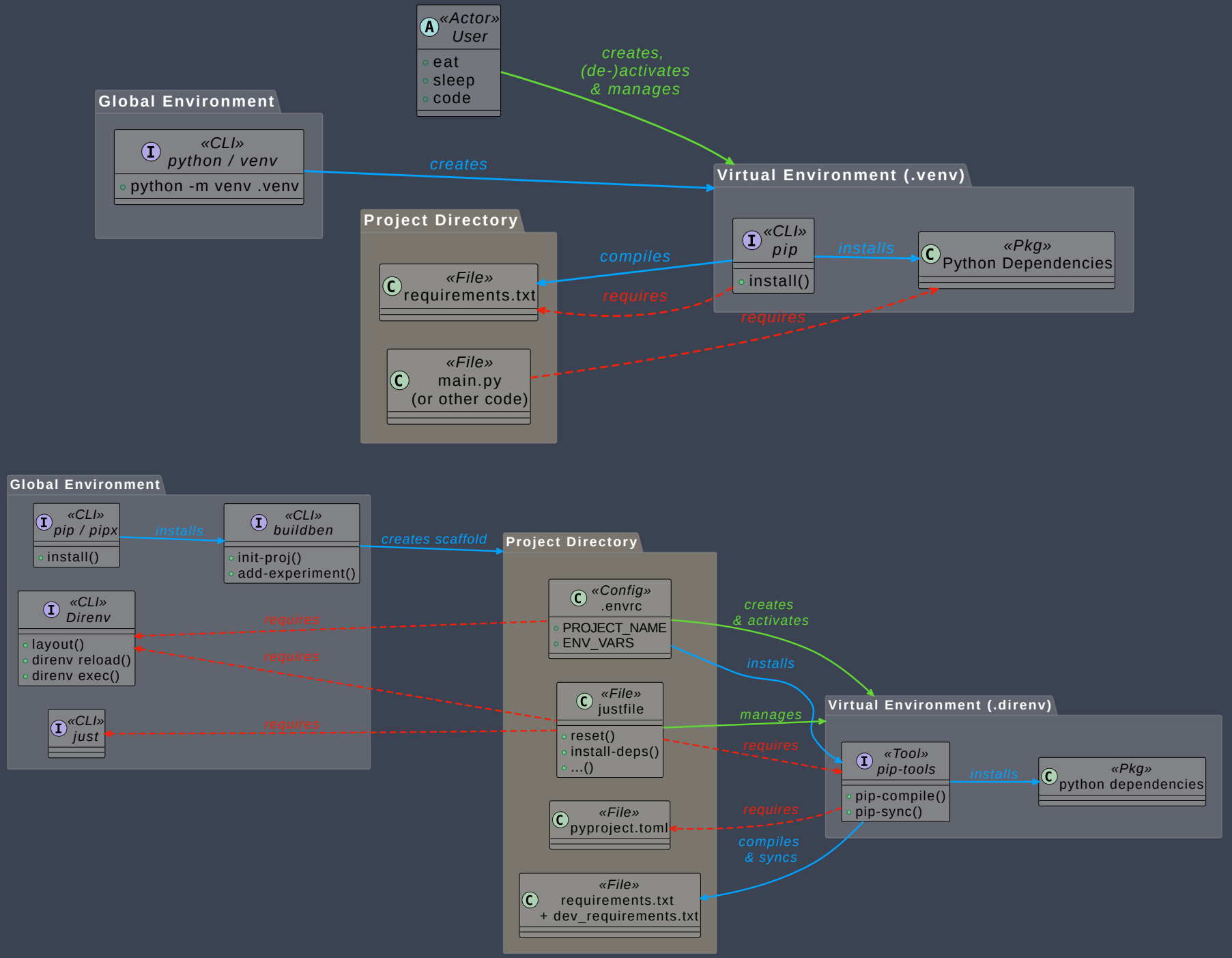
```
git clone "<repo-url>"                # Download
python -m venv ".venv"                # Protect system packages
source .venv/bin/activate             # Activate virtual environment
pip install -r requirements.txt       # Install dependencies
```

Limits:

- `requirements.txt` only holds dependencies, not the **project structure**.
  - Python **can't import** modules one directory up.
  - VS Code (sometimes) struggles with **refactoring** & **typing** across packages.
- `requirements.txt` must be manually updated.
- `requirements.txt` mixes runtime and development dependencies.
- Activating `.venv` can be forgotten or annoying.


«Actor»
User
eat
sleep
code
creates,
(de-)activates
& manages
Global Environment
«CLI»
python / venv
python -m venv .venv
creates
Virtual Environment (.venv)
Project Directory
«CLI»
pip
install()
installs
«Pkg»
Python Dependencies
compiles
«File»
requirements.txt
requires
requires
«File»
main.py
(or other code)


Global Environment
«CLI»
pip / pipx
install()
installs
«CLI»
buildben
init-proj()
add-experiment()
creates scaffold
«CLI»
Direnv
layout()
direnv reload()
direnv exec()
requires
requires
«CLI»
just
requires
Project Directory
«Config»
.envrc
PROJECT_NAME
ENV_VARS
creates
& activates
installs
«File»
justfile
reset()
install-deps()
...()
manages
requires
Virtual Environment (.direnv)
«Tool»
pip-tools
pip-compile()
pip-sync()
installs
«Pkg»
python dependencies
«File»
pyproject.toml
requires
compiles
& syncs
«File»
requirements.txt
+ dev_requirements.txt


«Actor»
User
eat
sleep
code
creates,
(de-)activates
& manages
Global Environment
«CLI»
python / venv
python -m venv .venv
creates
Virtual Environment (.venv)
Project Directory
«CLI»
pip
install()
installs
«Pkg»
Python Dependencies
compiles
«File»
requirements.txt
requires
requires
«File»
main.py
(or other code)
Global Environment
«CLI»
pip / pipx
install()
installs
«CLI»
buildben
init-proj()
add-experiment()
creates scaffold
«CLI»
Direnv
layout()
direnv reload()
direnv exec()
requires
requires
«CLI»
just
requires
Project Directory
«Config»
.envrc
PROJECT_NAME
ENV_VARS
creates
& activates
installs
«File»
justfile
reset()
install-deps()
...()
manages
requires
Virtual Environment (.direnv)
«Tool»
pip-tools
pip-compile()
pip-sync()
installs
«Pkg»
python dependencies
«File»
pyproject.toml
requires
compiles
& syncs
«File»
requirements.txt
+ dev_requirements.txt

Project Directory: `src`-Layout

```
# src layout (good)
myproject/
├── src/
│   └── myproject/
│       ├── main.py
│       └── package/module.py
├── tests/
│   └── test_module.py
├── README.md
```

```
# flat layout (risky)
myproject/
│
│
├── main.py
├── package/module.py
├── tests/
│   └── test_module.py
├── README.md
```

Benefits:

- Avoids imports from working directory via `PYTHONPATH`
  → Forces tests to run on installed code: `pip install -e .` → Catches `import` bugs
- Builds **clean wheels**: Stray files never ship to PyPI
- Recommended by Python Packaging Authority (PyPA)

Project Directory: Inside `src`

```
myproject/
├── src/
│   └── myproject/              # Single directory, same name as project root (Recommended)
│       ├── __init__.py         # Marks directory as package; runs on first import!
│       ├── main.py             # Optional CLI entry-point (wired in via pyproject.toml)
│       ├── sheesh.py           # >>> import myproject.sheesh
│       ├── clients/            # >>> import myproject.clients
│       │   ├── __init__.py    # Sub-package "clients"
│       │   ├── llm.py         # >>> import myproject.clients.llm
│       │   └── embedding.py   # >>> import myproject.clients.embedding
│       └── utils/              # >>> import myproject.utils
│           ├── __init__.py    # Sub-package "utils"
│           ├── cooltool.py    # >>> import myproject.utils.cooltool
│           └── module6.py     # >>> import myproject.utils.module6
│
```

Project Directory: Auxiliary Files in Project Root

```
myproject/
├── .venv/                    # Virtual environment (or .direnv!)
├── .env                      # Environment variables (& secrets)
├── .gitignore
├── .git/                     # Repository metadata
├── src/
│   └── myproject/          # Separate source code from tests!
├── tests/
│   └── test_module1.py     # Tests for module1
├── justfile                  # Development tasks
├── pyproject.toml            # Project metadata, Setup!
├── requirements.txt          # Dependencies
├── requirements-dev.txt      # Development dependencies
├── README.md
├── LICENSE
```