

scverse cookiecutter template

Follow along: [scverse.org/
cookiecutter-scverse-
presentation](https://scverse.org/cookiecutter-scverse-presentation)



Having impact with software

- Aim of many academics: Publish papers and get citations
- High quality software helps a lot with both
- Being part of the software ecosystem ensures that you meet a set of minimum requirements for maximal impact
- Goal of this workshop is to provide you a path to developing high quality software

scverse ecosystem

- Set of packages built around the scverse core packages
- Featured on the [scverse ecosystem](#) website

Ecosystem packages maintained by scverse community

Many popular packages rely on scverse functionality. For instance, they take advantage of established data format standards such as AnnData and MuData, or are designed to be integrated into the workflow of analysis frameworks. Here, we list ecosystem packages following development best practices (continuous testing, documented, available through standard distribution tools).

This listing is a work in progress. See [scverse/ecosystem-packages](#) for inclusion criteria, and to submit more packages.

Search through 59 packages

Package	Description
CellOracle	A computational tool that integrates single-cell transcriptome and epigenome profiles to infer gene regulatory networks (GRNs), critical regulators of cell identity.
CellRank	CellRank is a toolkit to uncover cellular dynamics based on Markov state modeling of single-cell data. It contains two main modules - kernels compute cell-cell transition probabilities and estimators generate hypothesis based on these.
Cell_BLAST	Cell BLAST is a cell querying tool for single-cell transcriptomics data.

scverse ecosystem requirements

- Name, description, ...
- OSI-approved license
- Versioned releases that can be installed from a standard registry (PyPI, Conda, ...)
- Automated tests (executed via continuous integration)
- API and use-case documentation
- Uses scverse datastructures where appropriate

How to get into the scverse ecosystem

- Instructions to add your package are on the [scverse ecosystem repository](#)
- Either submit existing package
- Easier alternative: [scverse cookiecutter template](#)

scverse cookiecutter template

- Project template for scverse packages
- Provides a best practice structure
- Many features such as continuous integration, documentation setup, ...
- Ensures that you automatically tick all scverse ecosystem requirements

Steps

1. Set up global environment to create project
2. Create project using [scverse cookiecutter template](#)
3. Set up project specific development environment
4. Develop your package
5. Submit to [scverse ecosystem](#)

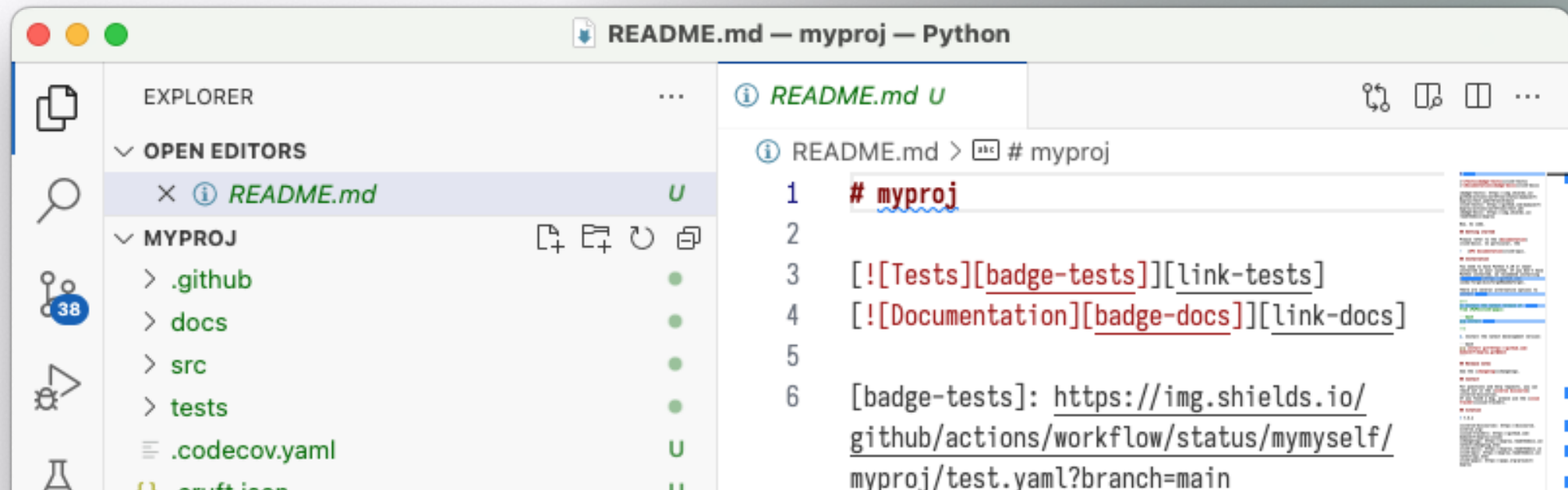
Step 1: Global environment

- Install using a package manager or installer:
 - `code` , `hatch` , & `git`
 - `uv` or `pipx`
- Install using a package manager, `pipx` , or `uv` :
 - `cruft` & `pre-commit`

```
$ pipx install cruft pre-commit # or  
$ uv tool install cruft pre-commit
```


Step 2: Creating the project

```
$ cruft create https://github.com/scverse/cookiecutter-scverse  
project_name (project-name): myproj  
[...]  
$ code myproj
```




Step 3: Environment management

Hatch envs **basic usage**:

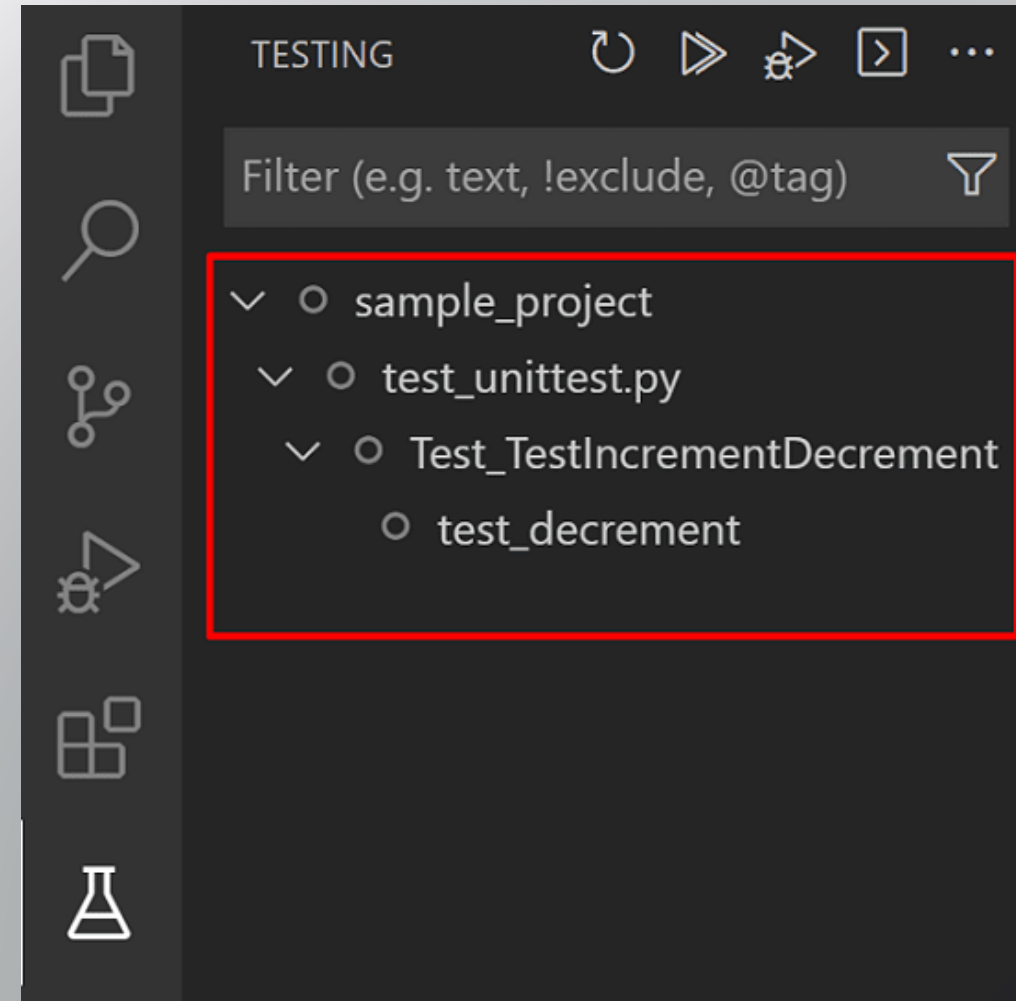
```
$ hatch run [env:]command [...args] # e.g. `... docs:build -T`  
$ hatch test [...args]  
$ hatch env remove <name> # or `hatch env prune` for all  
$ hatch find hatch-test  
~/.local/share/hatch/env/virtual/myproj/FsejNibV/hatch-test.py3.12  
[...]
```

Tell VS Code:

 | $\wedge + \square + P \rightarrow$ Python: Select Interpreter

running tests

```
$ hatch test --help
[...]  
Options:  
  -r, --randomize  
  -p, --parallel  
  -c, --cover  
  -a, --all  
  -py, --python=X.Y  
  -i, --include=VAR=VAL  
  -x, --exclude=VAR=VAL  
  -s, --show  
...otherwise same as `pytest`
```

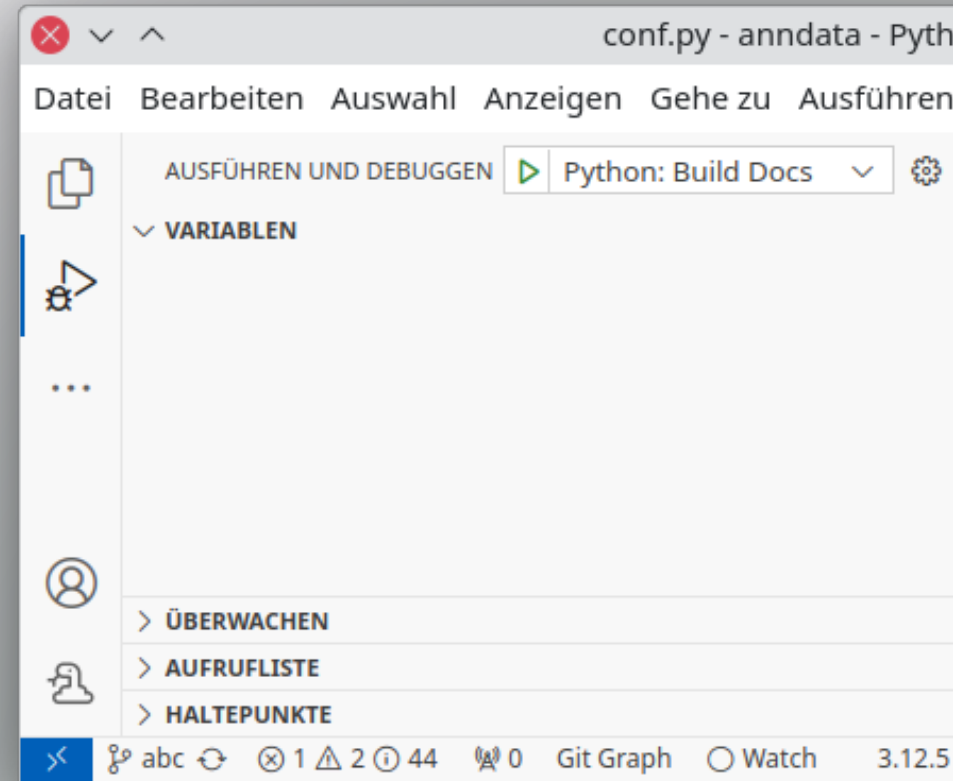


building docs

```
$ hatch run docs:build  
$ hatch run docs:open  
$ hatch run docs:clean
```

See `pyproject.toml`:

```
[tools.hatch.envs.docs]  
scripts.build = "..."  
...
```



formatting and linting

VS Code:

```
{  
  "[python]": {  
    "editor.formatOnSave": true,  
    "editor.defaultFormatter": "charliermarsh.ruff",  
    "editor.codeActionsOnSave": { ... },  
  }, ...  
}
```

CLI: `pre-commit` (or `hatch run pre-commit`)

```
$ pre-commit install # `git commit` hook  
$ pre-commit run --all-files
```

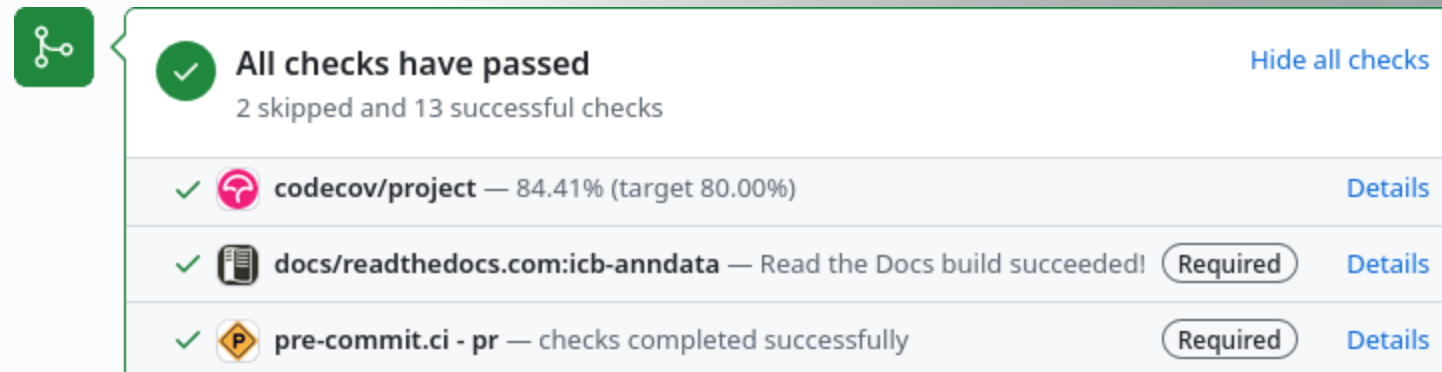
existing project

- Simple project
 1. Instantiate template
 2. Replace `src/*` directory with your package
 3. Edit `[project]` table in `pyproject.toml`
- Complex project
 1. Step by step PRs: formatter, ...
 2. We can help!

follow along

Step 4: committing code

- Use PRs, don't push to `main`
- Set up pre-commit.ci, codecov.io on github.com/<you>/<yourpackage>/settings/installation



A screenshot of a GitHub Actions workflow status card. The card has a green header with a checkmark icon and the text "All checks have passed" and "2 skipped and 13 successful checks". Below the header is a table of checks:

Check Name	Status	Details
codecov/project — 84.41% (target 80.00%)	✓	Details
docs/readthedocs.com:icb-anndata — Read the Docs build succeeded!	✓	Details
pre-commit.ci - pr — checks completed successfully	✓	Details

ReadTheDocs

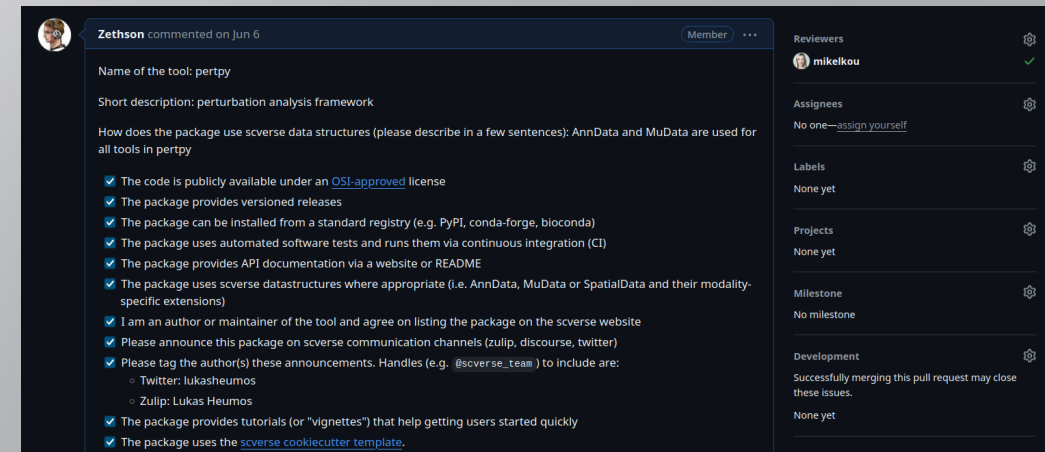
- Set up readthedocs.org and its [PR previews](#):

Warning

This page [was created](#) from a pull request ([#1616](#)).

Step 5: Submit to scverse ecosystem

- Follow the [instructions](#)
- Submit a pull request



Thank you