
rio VRT

Release 0.3.1

pierrick rambaud

Aug 29, 2026

CONTENTS

- 1 Usage 3**
 - 1.1 Installation 3
 - 1.2 Usage 3
- 2 Contribute 5**
 - 2.1 Workflow for contributing changes 5
 - 2.2 Clone the repository 5
 - 2.3 Contribute to the codebase 6
 - 2.4 Contribute to the docs 6
 - 2.5 Realease new version 6
- 3 Documentation contents 7**

rio vrt has been build to allow users in pure Python to build a vrt without installing GDAL in their environment. This lib is only relying on [rasterio](#).

As a start it supports files: - with same projection - with same number of bands - does not support colortable

Other functionalities from the original [buildvrt GDAL](#) method will be added upon request.

This section contains information about **rio vrt** to get you started.

1.1 Installation

Use pip to install **rio vrt** in your environment:

```
pip install rio-vrt
```

1.2 Usage

As straight forward as it should be: list all the file you want to gather and call the `build_vrt` method.

```
from rio_vrt import build_vrt

raster_files = ["example.tif", "example2.tif", "...", "examplen.tif"]
vrt_file = build_vrt("example.vrt", raster_files)
```


CONTRIBUTE

Thank you for your help improving **rio VRT**!

rio VRT uses **nox** to automate several development-related tasks. Currently, the project uses four automation processes (called sessions) in `noxfile.py`:

- `mypy`: to perform a mypy check on the lib;
- `test`: to run the test with pytest;
- `docs`: to build the documentation in the `build` folder;
- `lint`: to run the pre-commits in an isolated environment

Every nox session is run in its own virtual environment, and the dependencies are installed automatically.

To run a specific nox automation process, use the following command:

```
nox -s <session name>
```

For example: `nox -s test` or `nox -s docs`.

2.1 Workflow for contributing changes

We follow a typical GitHub workflow of:

- Create a personal fork of this repo
- Create a branch
- Open a pull request
- Fix findings of various linters and checks
- Work through code review

See the following sections for more details.

2.2 Clone the repository

First off, you'll need your own copy of **rio VRT** codebase. You can clone it for local development like so:

Fork the repository so you have your own copy on GitHub. See the [GitHub forking guide for more information](#).

Then, clone the repository locally so that you have a local copy to work on:

```
git clone https://github.com/<YOUR USERNAME>/rio-vrt
cd rio-vrt
```

Then install the development version of the extension:

```
pip install -e .[dev]
```

This will install the **rio VRT** library, together with two additional tools: - [pre-commit](#) for automatically enforcing code standards and quality checks before commits. - [nox](#), for automating common development tasks.

Lastly, activate the pre-commit hooks by running:

```
pre-commit install
```

This will install the necessary dependencies to run pre-commit every time you make a commit with Git.

2.3 Contribute to the codebase

Any larger updates to the codebase should include tests and documentation. The tests are located in the `tests` folder, and the documentation is located in the `docs` folder.

To run the tests locally, use the following command:

```
nox -s test
```

See [below](#) for more information on how to update the documentation.

2.4 Contribute to the docs

The documentation is built using [Sphinx](#) and deployed to [Read the Docs](#).

To build the documentation locally, use the following command:

```
nox -s docs
```

For each pull request, the documentation is built and deployed to make it easier to review the changes in the PR. To access the docs build from a PR, click on the “Read the Docs” preview in the CI/CD jobs.

2.5 Realease new version

To release a new version, start by pushing a new bump from the local directory:

```
cz bump
```

The commitizen-tool will detect the semantic version name based on the existing commits messages.

Then push to Github. In Github design a new release using the same tag name and the `release.yaml` job will send it to pipy.

DOCUMENTATION CONTENTS

The documentation contains 3 main sections:

Usage Usage and installation

[usage.html](#) Contribute Help us improve the lib.

[contribute.html](#) API Discover the lib API.

[api/modules.html](#)