# Scrapy Release Procedure

This page outlines the procedure used to release a new version of Scrapy.

You will need bump2version installed.

## Release notes

First the release notes page of the documentation must be udpated to include an entry for the new version. Do this in the target branch, which will be `master` if you are releasing a major or minor version, or a version-specific branch in case of a patch version.

The changes must include looking for the following references in the code base and replacing them with the corresponding values where appropriate:

- `VERSION` : new major or minor version being released, without the patch version number ( `x.y` , not `x.y.0` )
- `PREVIOUS_VERSION` : the major or minor version before the new version being released, without the patch version number ( `x.y` , not `x.y.0` )

A pull request for this can be created at any point, to allow for discussions, and updated as new changes are accepted into the target branch. Once the code is frozen and the release date set, the pull request can be updated accordingly and merged into the target branch.

## Version and git branch

**Note:** The code below assumes that the Scrapy repository is configured as the `upstream` remote in your local clone, which you can do with: `git remote add upstream git@github.com:scrapy/scrapy.git`

If you are releasing major version `x.0.0` :

```
git checkout master
git pull --ff-only upstream master
git log -1  # Check that the last commit is the release notes update
bump2version major
git checkout -b x.0
git push upstream master x.0 x.0.0
```

If you are releasing minor version `x.y.0` :

```
git checkout master
git pull --ff-only upstream master
git log -1  # Check that the last commit is the release notes update
bump2version minor
git checkout -b x.y
git push upstream master x.y x.y.0
```

If you are releasing patch version `x.y.z` :

```
git checkout x.y
git pull --ff-only upstream x.y
git log -1  # Check that the last commit is the release notes update
bump2version patch
git push upstream x.y x.y.z
```

## Github release

Create a Github release entry, with a description including the highlights from the release notes and a link to the release notes of the new version in Read the Docs.

This step can be skipped in the case of new tags created just to trigger re-packagings in the Python Package Index.

## Python Package Index

New Scrapy versions are automatically deployed on the Python Package Index through Travis CI, at the end of the tag build job where `PYPI_RELEASE_JOB=true` is defined.

## Read the Docs

For new major and minor versions:

- Enable the new version:

  1. Go to https://readthedocs.org/projects/scrapy/versions/
  2. Find the entry of the new *branch* (not *tag*) under **Inactive Versions**, at the bottom of the page, and select **Edit**.
  3. Mark the *Active* checkbox, and select **Save**.

- Set the new version as default:

  1. Go to https://readthedocs.org/dashboard/scrapy/advanced/
  2. Change the **Default branch** to the new *branch* (not *tag*), and select **Save** at the bottom of the page.

## scrapy.org website

Update scrapy versions

## Send announcements

In the case or major and minor versions, or patch versions that users should install with urgency (e.g. patch versions fixing security issues):

- Leave a message in the `#scrapy` IRC channel with a link to the release notes
- Submit a link entry to the Scrapy subreddit that links to the release notes.
- Tweet from @ScrapyProject with a link to the release notes. (only for major and minor versions)

## Update Conda packages

Create a pull request in https://github.com/conda-forge/scrapy-feedstock against `master` which updates recipe/meta.yaml as follows:

1. Update the `version` string.
2. Update the `sha256sum` from the PyPI's source release tarball.
3. Reset the `build/number` to `0`. This number is only increased when we re-build a package with no version change.
4. Update the requirements if needed. This can be done by checking the requirements declared in `setup.py`.

The pull request will trigger builds in Travis CI, Circle CI and Appveyor. This may take a while depending on the conda-forge build queue. Once builds are OK, the pull request is ready to merge and packages will be uploaded to the `conda-forge` channel automatically.

Alternatively, you can wait for https://github.com/regro-cf-autotick-bot to create such a pull request, and just edit it as needed (you should be able to commit directly to it, even though the GitHub online editor).

## Cherry-pick into master

If you’ve released a patch version, you should cherry-pick the required commits into master (exclude version-bumping changes).

## Upgrade CI packages

To avoid the risk of CI suddenly failing and confusing pull request authors, we freeze some CI packages:

- bandit
- flake8
- pylint
- pytest-cov

Right after a major or minor release would be a good time to update those packages, solving or allow-listing any issue raised by their new versions.