FEniCS Project Documentation

FEniCS Project Team

Mar 25, 2020

Contents

This is experimental documentation for the FEniCS Project. This version of the documentation on Read the Docs is under development.

CHAPTER 1

Installation

This guide summarises how to install FEniCS. The most reliable way to get started is using the Docker containers (Linux, macOS and Windows).

For installation in high performance computing clusters we recommend always building from source.

1.1 From source

FEniCS consists of Python components FIAT, dijitso, UFL, FFC, and C++/Python components DOLFIN and optional mshr. DOLFIN is the main user interface of FEniCS, both for C++ and Python.

FEniCS needs Python 3. For building CMake is needed and pip is recommended.

For building optional Python interface of DOLFIN and mshr, pybind11 is needed since version `2018.1.0`. To install it:

```
PYBIND11_VERSION=2.2.3
wget -nc --quiet https://github.com/pybind/pybind11/archive/v${PYBIND11_VERSION}.tar.
↪gz
tar -xf v${PYBIND11_VERSION}.tar.gz && cd pybind11-${PYBIND11_VERSION}
mkdir build && cd build && cmake -DPYBIND11_TEST=off .. && make install
```

It may be useful to add `cmake` flag `-DCMAKE_INSTALL_PREFIX=<prefix>` to install to a user location.

1.1.1 Stable version

To install the Python components of FEniCS:

```
pip3 install fenics-ffc --upgrade
```

This will install FFC and its dependencies. It may be useful to add flag `--user` or `--prefix=<prefix>` to install to a user location. To install DOLFIN, and optionally mshr and/or Python interface of DOLFIN/mshr:

```
FENICS_VERSION=$(python3 -c"import ffc; print(ffc.__version__)")
git clone --branch=$FENICS_VERSION https://bitbucket.org/fenics-project/dolfin
git clone --branch=$FENICS_VERSION https://bitbucket.org/fenics-project/mshr
mkdir dolfin/build && cd dolfin/build && cmake .. && make install && cd ../..
mkdir mshr/build   && cd mshr/build   && cmake .. && make install && cd ../..
cd dolfin/python && pip3 install . && cd ../..
cd mshr/python   && pip3 install . && cd ../..
```

It may be useful to add `cmake` flag `-DCMAKE_INSTALL_PREFIX=<prefix>` and `pip3` flag `--user` or `--prefix=<prefix>` to install to a user location.

See detailed instructions for DOLFIN and mshr.

Todo: Update DOLFIN documentation for latest release, including updates for building Python interface with pybind11, and provide a link to documentation of stable version here.

Consolidate multiple documentation locations: RTD, webserver.

1.1.2 Development version

The FEniCS source code can be found on the FEniCS Bitbucket pages and the FEniCS Github pages. To download and build current development version run the following commands:

```
git clone https://github.com/FEniCS/fiat.git
git clone https://bitbucket.org/fenics-project/dijitso
git clone https://github.com/FEniCS/ufl.git
git clone https://bitbucket.org/fenics-project/ffc
git clone https://bitbucket.org/fenics-project/dolfin
git clone https://bitbucket.org/fenics-project/mshr
cd fiat    && pip3 install . && cd ..
cd dijitso && pip3 install . && cd ..
cd ufl     && pip3 install . && cd ..
cd ffc     && pip3 install . && cd ..
mkdir dolfin/build && cd dolfin/build && cmake .. && make install && cd ../..
mkdir mshr/build   && cd mshr/build   && cmake .. && make install && cd ../..
cd dolfin/python && pip3 install . && cd ../..
cd mshr/python   && pip3 install . && cd ../..
```

See detailed instructions for DOLFIN and mshr.

1.2 Debian/Ubuntu packages

FEniCS is available as a package for Debian and Ubuntu[1] in the official repositories. If you are using Ubuntu, we recommend the Ubuntu PPA.

1.2.1 Ubuntu PPA

The Ubuntu Personal Package Archives (PPA) version is the latest release of FEniCS. To install FEniCS from the Ubuntu PPA:

[1] mshr is not available from official Debian and Ubuntu repositories.

```
sudo apt-get install --no-install-recommends software-properties-common
sudo add-apt-repository ppa:fenics-packages/fenics
sudo apt-get update
sudo apt-get install --no-install-recommends fenics
```

1.2.2 Official Debian/Ubuntu repositories

The version of FEniCS in the Debian/Ubuntu repositories[1] is not always the most recent FEniCS release. To install FEniCS from the official Debian/Ubuntu repositories:

```
sudo apt-get update
sudo apt-get install --no-install-recommends fenics
```

1.3 Containers/Docker (Linux, macOS and Windows - 64 bit)

A collection of Docker containers for FEniCS are available. To get started, install Docker, and then run

```
docker run -ti -v $(pwd):/home/fenics/shared -w /home/fenics/shared quay.io/
↪fenicsproject/stable:current
```

A helper script is also available. To install it automatically to `$HOME/.local/bin`, run the command:

```
curl -s https://get.fenicsproject.org | bash
```

To run the FEniCS Docker image, use the command `fenicsproject run`. For more options and features, see `fenicsproject help`.

For detailed instruction on the Docker containers and background, a see http://fenics-containers.readthedocs.org/en/latest/ for how to run FEniCS inside a container.

1.4 Conda (Linux and macOS - 64 bit)

To install the latest FEniCS release from using conda:

```
conda install -c conda-forge fenics
```

To install a development snapshot:

```
conda install -c conda-forge/label/prerelease -c conda-forge fenics
```

The packages are part of conda forge (see https://anaconda.org/conda-forge/fenics), and the recipes are maintained at https://github.com/conda-forge/fenics-feedstock/.

Note: Conda support is experimental and subject to changes.

CHAPTER 2

Getting started

Todo: Getting started guide.

CHAPTER 3

Developers

This notes are for FEniCS developers.

3.1 Development workflows

Todo: Describe work flow that is common across projects.

3.2 Making releases

These instructions are for developer making release of FEniCS packages.

3.2.1 Releasing packages

Check that all CI systems are green before making a release.

Python packages

These instruction cover:

- FFC
- FIAT
- UFL
- dijitso

Making the release

1. Checkout branch and make sure it is clean:

```
git checkout master
git clean -fdx
```

2. Update release notes.
3. Update version number in `<src>/__init__.py`
4. Commit changes.
5. Tag repository and push tag:

```
git tag -a $VERSION -m "Version $VERSION"
git push origin $BRANCH
```

6. Update version number and add `dev` in `<src>/__init__.py`. Commit and push.

Uploading to PyPI

This applies to packages FFC, dijitso, FIAT, UFL and the FEniCS Project Python Metapackage.

This should be done soon after a release is made.

1. Checkout release tag and make sure it is clean:

```
git checkout tags/$VERSION
git clean -fdx
```

2. Build source distribution:

```
python3 setup.py sdist
```

3. Sign the package:

```
gpg --detach-sign -a dist/package-1.0.1.tar.gz
```

   The gpg key is in the Steering Council LastPass account.

4. Upload to PyPI:

```
twine upload dist/package-1.0.1.tar.gz dist/package-1.0.1.tar.gz.asc
```

   The username on PyPI is `fenicsproject`.

Note: To use the PyPI test repository:

```
twine upload --repository-url=https://test.pypi.org/legacy/ dist/*
```

Todo:

- Update on Read-the-Docs

DOLFIN

Todo: Fill in

To create tarball from Git tag:

```
VERSION=2018.1.0.post0
git archive -9 --prefix=dolfin-${VERSION}/ -o dolfin-${VERSION}.tar.gz ${VERSION}
```

Don't forget to make sure that Git LFS data files are packaged. Then sign the tarball using FEniCS PGP keypair by procedure above and upload both files to Bitbucket.

3.2.2 Docker containers

Todo: Fill in

3.2.3 Anaconda packages

Todo: Fill in

3.2.4 Ubuntu PPA

Todo: Fill in

3.2.5 Versioning scheme

FEniCS uses a hybrid date-based/serial version numbering scheme. Releases have a version number:

```
year.number.fix
```

In the case of packaging errors (rather than bugs in the release) you can also introduce .postx releases, e.g.:

```
year.number.fix.post0
```

3.2.6 FFC and the FEniCS Python Metapackage

Assume we want to release the `2017.2.0` release. `master` would currently specify:

```
VERSION = "2017.2.0.dev0"
RESTRICT_REQUIREMENT = ">=2017.2.0.dev0,<2017.3"
```

1. On the release branch `2017.2.0`:

```
VERSION = "2017.2.0"
RESTRICT_REQUIREMENTS = ">=2017.2,<2017.3"
```

The upper bound should be *tight* against the current release, i.e. don't do:

```
VERSION = "2017.2.0"
RESTRICT_REQUIREMENTS = ">=2017.2,<2018.1"
```

2. The update to `master` post-release would be:

```
VERSION = "2018.1.0.dev0"
RESTRICT_REQUIREMENTS = ">=2018.1.0.dev0,<2018.2"
```

3.2.7 Other FEniCS Python packages

1. For the release branch, remove the `.dev0` suffix:

```
VERSION = "year.number.0"
```

2. On release of a new version, the `VERSION` string in `setup.py` on `master` branches of the FEniCS Python packages should be incremented:

```
VERSION = "year.number+1.0.dev0"
```

In the case of a release late in the year, it may be more appropriate to increment the year:

```
VERSION = "year+1.number.0.dev0"
```

Todo: Document and explain

CHAPTER 4

Documentation for components

FEniCS is a collection of inter-operating modules. Links to the documentation for each module are listed below. For end-users, the DOLFIN, mshr and UFL documentation is most relevant.

- DOLFIN
- mshr
- UFL
- FFC
- FIAT
- dijitso

4.1 Documentation build status

Main	
DOLFIN	
mshr	
UFL	
FFC	
FIAT	
dijitso	