RSE Infrastructure of FEniCSx

Git Workflow and External Contributions

The FEniCSx team employs a forking workflow using feature branches on the developer's forks. Even core developers work on their forks except when finalizing a release. The contribution process starts by creating a fork of the repository. It is recommended to announce new contributions via GitHub issues. After implementing the new feature, a pull request can be made to the main branch of the origin repository. When the test and build pipeline succeeds and the review of a core developer is successful, the pull requests can be merged. Whenever a new version is released, the current code on the main branch is merged into the release branch.

The FEniCSx ecosystem consists of various building blocks. Besides the repository for DOLFINx there are separate repositories for ffcx, ufl and basix. Their RSE infrastructure is similar to the DOLFINx repository. In addition, there are further repositories, such as for the documentation and the website.

Possibilities to contribute are explained in CONTRIBUTING.md. Contributions from new developers are welcome but should follow the code of conduct and the style guides of the project such as the C++ style guide. There are also issues targeting new developers that are labeled as "good first issue".

The long-term goals of the project are formulated in the road map on the website. Questions about the usage of FEniCS can be asked on Discourse. The communication between developers takes place on Slack.

Virtualization and Containers

The Docker platform is used extensively to test and ship the code. On the one hand, Docker containers are used by the testing and CI pipeline to create a standardized environment. On the other hand, Docker containers can be used easily to run the latest stable release of DOLFINx

```
docker run -ti dolfinx/dolfinx:stable
```

to run the latest nightly build of DOLFINx

```
docker run -ti dolfinx/dolfinx:nightly
```

or to run a Jupyter Lab with the latest stable release of DOLFINx

```
docker run --init -ti -p 8888:8888 dolfinx/lab:stable
```

Building and Packaging

The C++ code is built using `cmake` based on `CMakeLists.txt` files. The Python code is packaged using `build` based on the `setup.py` file.

Spack packages are available and can be installed via

```
spack env create fenicsx-env
spack env activate fenicsx-env
spack add py-fenics-dolfinx cflags="-O3" fflags="-O3"
spack install
```

There are also distribution packages, e.g. for Ubuntu and Debian, but they are rather outdated.

Documentation

The documentation of FEniCSx consists of the DOLFINx C++ API reference, the DOLFINx Python API reference, and the API references of the other building blocks of FEniCSx. Besides that, there are some tutorials on the website of a core developer that are treated as official tutorials.

According to the README.md of the C++ API reference

> The C++ API documentation is generated from the source code using Doxygen, and the Doxygen output is curated and rendered using reStructured text with Sphinx and Breathe.

The Python API reference is created using reStructured text with Sphinx.

Testing and CI

Different tools are used to automate the testing and CI tasks. Besides GitHub Actions, the FEniCSx team also uses CircleCI and SonarCloud for automation purposes.

According to the configuration, GitHub Actions are used to build Docker images, build the software in different configurations, and update the documentation on the website. When you make a pull request, GitHub actions are automatically triggered. CircleCI is used to run the tests for different build configurations. Details can be found in the configuration file.

For C++, unit tests using the test framework Catch2 and some demos are executed automatically. For Python, unit tests are run using pytest.