Geophysical Research Abstracts
Vol. 21, EGU2019-16869, 2019
EGU General Assembly 2019

Development workflows of the open-source porous media simulator OpenGeoSys

Lars Bilke (1), Dmitri Naumov (1), Olaf Kolditz (1,2)

(1) Helmholtz Centre for Environmental Research - UFZ, Environmental Informatics, Leipzig, Germany, (2) Technische Universität Dresden, Applied Environmental Systems Analysis, Dresden, Germany

1 OpenGeoSys introduction

OpenGeoSys[1] (OGS) is a scientific open source project for the development of numerical methods for the simulation of thermo-hydro-mechanical-chemical (THMC) processes in porous and fractured media. OGS is implemented in C++, it is object-oriented with an focus on the numerical solution of coupled multi-field problems (multi-physics). Parallel versions of OGS are available relying on both MPI and OpenMP concepts. Application areas of OGS are currently CO_2 sequestration, geothermal energy, water resources management, hydrology and waste deposition. OGS is developed by the OpenGeoSys Community consisting of the core developer team at UFZs Environmental Informatics department and developer and user from national and international research facilities and universities.

We employ a sophisticated approach to open-source software development enabling developers a rapid iteration on new implementations which is described in the following.

1.1 OGS software engineering & continuous integration

OGS-6 is developed as an open-source project by using the GitHub platform for code hosting, code review, and bug and issue tracking[2]. A Jenkins continuous integration (CI) server[3] builds and tests the software on every proposed change (pull request) on supported platforms (Windows, Linux, Mac Desktop systems and Linux HPC cluster systems) for quality assurance (QA).

1.1.1 Approach

The QA process is automated and defined in Jenkins Pipeline Domain Specific Language (DSL)[4] scripts which are part of the source code to allow the concurrent testing of multiple development branches.

Submitting a proposed change (pull request) triggers a CI-job which consist of several sub-jobs with several stages each. A sub-job is created for every supported platform. We test both on bare-metal hardware as well as on containerized environments using Docker. Docker container providing Linux environments with different compiler setups are defined via the Dockerfile DSL in code as well [5]. Sub-job stages group complex steps into logical parts such as configuring a build, running static code analyzers, building binaries and testing build executables. After a CI-job finishes the developer gets immediate feedback which sub-job or even stage failed with its corresponding log output. Binaries for all supported platforms are generated and can be obtained for further manual testing if required.

[1] https://www.opengeosys.org
[2] https://github.com/ufz/ogs
[3] https://jenkins.opengeosys.org
[4] https://jenkins.io/doc/book/pipeline/syntax
[5] https://github.com/ufz/dockerfiles

1.1.2 Benefits

A developer can get started quickly by following step-by-step guides[6] on setting up a development environment. Upon creating a pull request on GitHub he gets immediate feedback (about 20 minutes after creation) from the CI system and can discuss questions with the core developer team to improve the implementation. Once the pull request is merged the software and corresponding documentation is packaged and deployed automatically.

With this setup large parts of the whole software engineering infrastructure and processes are formalized and defined via DSLs in version controllable code-repositories allowing for easy contributing and peer-review on the code, infrastructure and process levels breaking up the black-box behavior of traditional testing setups.

The user can obtain up-to-date quality-tested binaries, test data files and documentation inducing a rapid iteration between user feedback and developer implementation.

A clear versioning scheme of both software and data as well as archived software binaries allow reproducibility of scientific results. Software versions can be cited with a digital object identifier (DOI).

[6] https://opengeosys.org/docs/devguide