E-CAM Software Development Tools

Deliverable Type: Other
Delivered in Month 8– May2016

E-CAM
The European Centre of Excellence for
Software, Training and Consultancy
in Simulation and Modelling

Funded by the European Union under grant agreement 676531

Deliverable Description

D6.2 - E-CAM software development tools

On-line deployment of centralized tools for software development, documentation and maintenance. These will include tools for automatic extraction of software documentation, bug tracking, version control, low-level software (e.g. memory handling).

This deliverable describes the current status (as of the month of delivery) of the on-line deployment of centralized tools for software development, documentation and maintenance. These include tools for version control, bug tracking, build systems, documentation and other services.

Some useful low-level software is also highlighted, though these cannot be deployed centrally but must be used locally by the developer during the development process.

Contents

May 26, 2016
This deliverable was co-ordinated by Alan Ó Cais[1] (FZJ-JSC) on behalf of the E-CAM consortium.

[1] a.ocais@fz-juelich.de

1 Introduction

One of the main intended outcomes of the E-CAM project is the on-line deployment of centralized tools for software development, documentation and maintenance. This deliverable provides a basic description of the tools and workflow advice that E-CAM has made available to developers.

The services described here are not just for the E-CAM software team but the E-CAM community as a whole and we will support users within that community to effectively utilise the available tools.

The project has purchased a high-end server with 1TB RAID and remote back-ups to host services where required and to support development workflows.

1.1 Outlook

As it stands the initial support infrastructure has been created. With the first Extended Software Development workshops being carried out in one month after the deliverable date (June 2016), the infrastructure currently has no fully functional examples. This will change in the near future, the infrastructure will subsequently mature and adapt to the requirements of the community.

2 On-line Tools

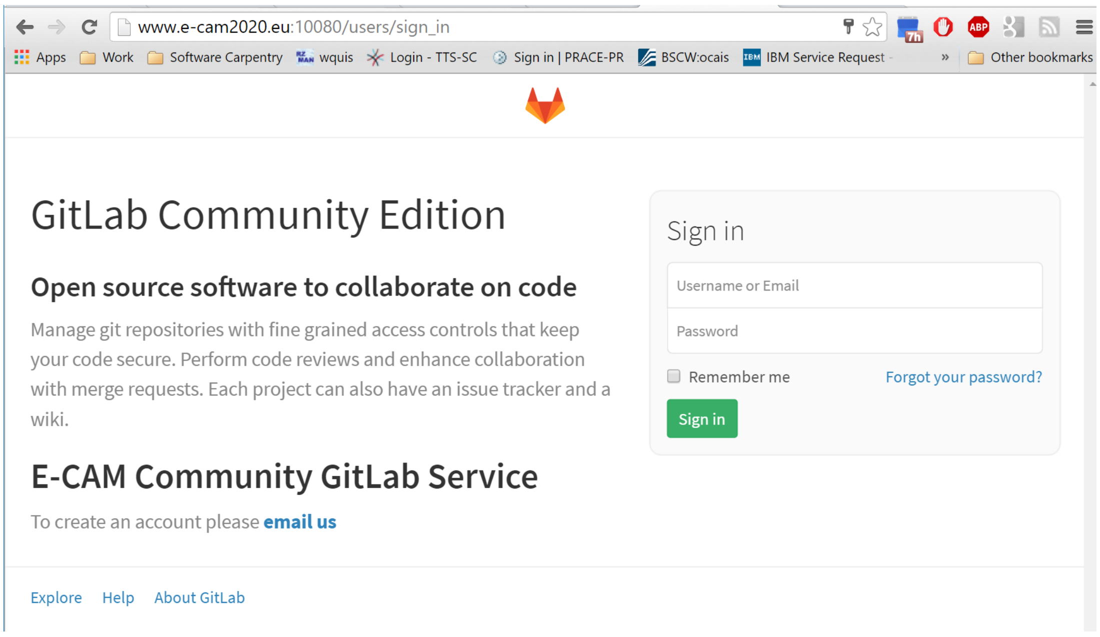
There are a number of core on-line resources that are to be used by the project which are intended to satisfy the requirements of

- Version control,
- Bug Tracking,
- Software documentation.

Git is our chosen version control system and our bug tracking and task management system is GitLab/GitHub. Software source documentation is done through Doxygen (see Section 3), with application documentation done using ReStructured Text and online integration done via hooks into readthedocs.org.

2.1 GitLab/GitHub

Version control is done via Git, a standard utility available on Linux distributions. The hosting of project repositories is done via GitLab or GitHub. GitHub is a well known repository hosting service which is free to use for public open-source projects.



GitLab Community Edition

Open source software to collaborate on code

Manage git repositories with fine grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

E-CAM Community GitLab Service

To create an account please **email us**

Sign in

Username or Email

Password

Remember me Forgot your password?

Sign in

Explore Help About GitLab

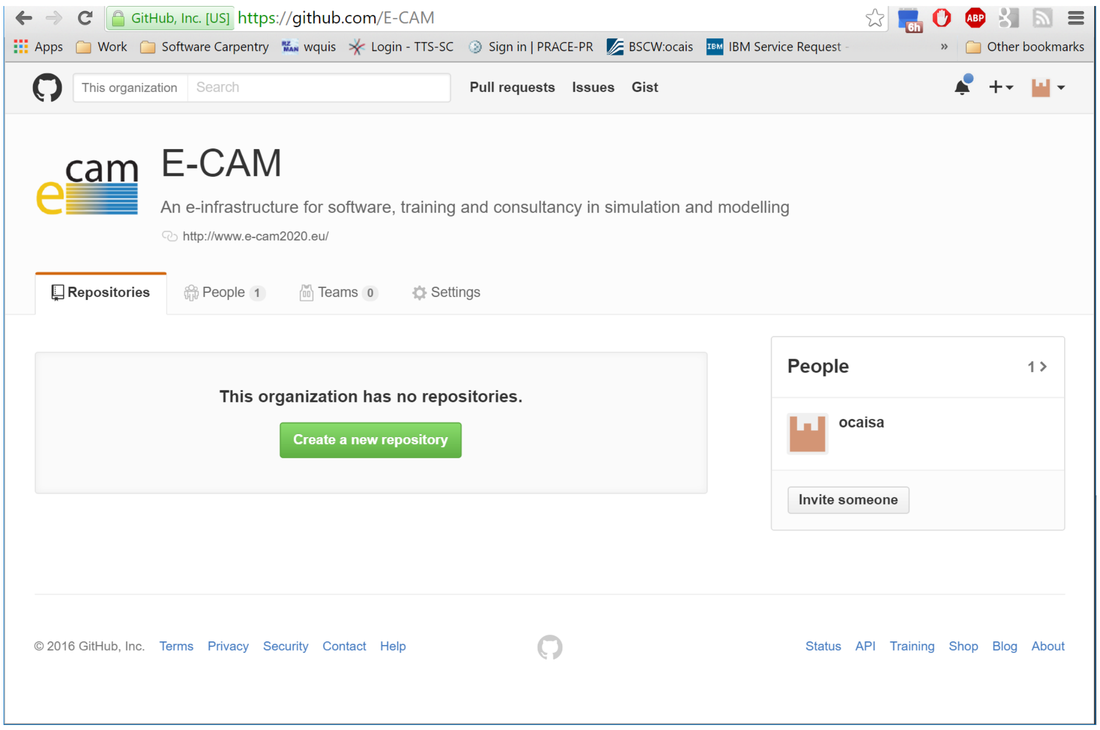
Figure 1: The GitLab service provided by E-CAM

The GitLab service, see Fig.1 is hosted by the project and provides the capability of having unlimited private repositories for the E-CAM community. This is a necessary addition to the project as some potential use cases of the project will require this protection due to licence limitations on the source code or similar reasons. Both services offer similar bug and issue tracking capabilities, as well as task allocation, for GitLab the feature list includes:

- Git repository management,
- code reviews,
- issue tracking,
- activity feeds
- wikis
- GitLab CI for continuous integration and delivery.

All of these features are available for use by all hosted repositories.

Public repositories will be mirrored between GitLab and GitHub, providing an implicit back-up facility and leaving the repository owners free to chose which platform they prefer for management, distribution and issue tracking (in this case). The GitHub E-CAM organisation site is shown in Fig.2.



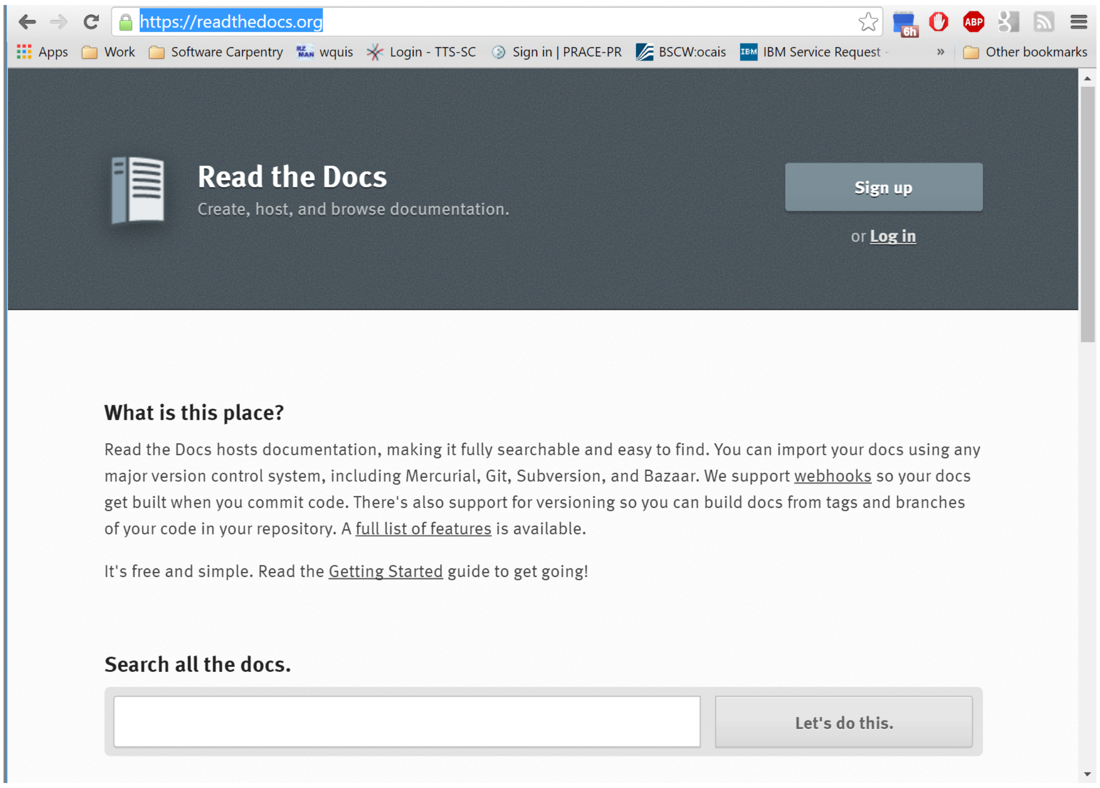
An e-infrastructure for software, training and consultancy in simulation and modelling

This organization has no repositories.

Figure 2: The GitHub organistion provided by E-CAM

2.2 Software Documentation

Source-code documentation is done via Doxygen, easily available through any Linux Distribution. We encourage the overarching application documentation to be done using ReStructured Text which is automatically understood by GitLab/GitHub.



What is this place?

Read the Docs hosts documentation, making it fully searchable and easy to find. You can import your docs using any major version control system, including Mercurial, Git, Subversion, and Bazaar. We support webhooks so your docs get built when you commit code. There's also support for versioning so you can build docs from tags and branches of your code in your repository. A full list of features is available.

It's free and simple. Read the Getting Started guide to get going!

Search all the docs.

Figure 3: The main *Read the Docs* site

For additional features and a more user-friendly end product, we will support the use of Read the Docs (see Fig.3) via hooks into GitLab/GitHub. This will also allow for published documentation to be version controlled and explicitly linked to published releases. The full feature list of Read the Docs includes:

- Auto-updating
- Support of canonical URLs (gives better search performance, by pointing all URLs to one version)
- Versions
- PDF Generation
- Search

Use of these features can make documentation more relevant, appealing, accessible and ensure similar quality control measures as code contributions.

3 Workflow Support

In addition to the publicly visible online services of Section 2, E-CAM will support and guide the workflow of the E-CAM community through the provision of a continuous integration infrastructure.

The continuous integration (CI) infrastructure will be provided via GitLabs own continuous integration infrastructure and will be hooked directly into the GitLab service. The GitLab-CI service is built on top of the open source components of Travis CI and provides similar capabilities in a tunable environment:

- Multi-language: build scripts are command line driven and work with Java, PHP, Ruby, C, and any other language
- Stable: your builds run on a different machine than GitLab.
- Parallel builds: GitLab CI splits builds over multiple machines, for fast execution.
- Realtime logging: a link in the merge request takes you to the current build log that updates dynamically.
- Versioned tests: a .gitlab-ci.yml file that contains your tests, allowing everyone to contribute changes and ensuring every branch gets the tests it needs.
- Pipeline: you can define multiple jobs per stage and you can trigger other builds.
- Autoscaling: you can automatically spin up and down VM's to make sure your builds get processed immediately and minimize costs.
- Continuous Delivery (CD): Continuous delivery and deployment are easy with multiple types of jobs, and secure environmental variables.

A major advantage of this approach is that it will also support private repositories without an additional cost to the community.

3.1 The Recommended Workflow

For new repositories we would suggest the following:

- The recommended interface language to be used for libraries should be C. Creating bindings for other languages from C is well documented and well supported, including for Fortran. Language bindings apart from Python should probably be implemented as a separate repository, to allow for independent library development without unnecessary overhead (as is now done for NetCDF, for example)
- Source code should be documented using Doxygen
- Source code should conform to the accepted style guides. For C/C++, the Google style guides and the cpplint C/C++ style checker help automate conformance checking. For Python, Pylint is a comprehensive tool.
- Python bindings can be created by the use of SWIG. Automatic creation of the python binding documentation is possible via doxy2swig
- Use the python bindings to create the testing infrastructure via pytest.
- Use CMake as the build tool

All of the necessary tools for the above will be supported by the CI infrastructure and templates will be provided for how to implement the flow.

3.2 Automated Testing

The CI infrastructure will allow automated testing of pull requests to repositories. The potential scope of testing is up to the end user but support for the following will be available:

- Style checks
- Build tests
- Unit tests
- Regression tests

These tests (or a subset) can be performed on every pull request, and merging of pull requests could be contingent on the tests passing.

3.3 Low-level Development Tools

Development with exa-scale targets is complex with debugging and performance analysis an essential part of the workflow.

There are many tools available to aid the code development process for HPC architectures. Their use cannot be automated and must be used directly by the developer during the development process. For performance analysis we direct you to the Virtual Institute for High-Productivity Supercomputing which covers many of these tools.

Specifically within the project, we can support Scalasca for performance optimisation. Other tools, such as debuggers, we can advise on their use and support them as necessary.

Index of URLs referenced