E-CAM Software Development Tools

E-CAM Deliverable 6.2
Deliverable Type: Other
Delivered in Month 8– May 2016

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

Project and Deliverable Information

Project Title	E-CAM: An e-infrastructure for software, training and discussion in simulation and modelling
Project Ref.	Grant Agreement 676531
Project Website	https://www.e-cam2020.eu
EC Project Officer	Juan PELEGRIN
Deliverable ID	D6.2
Deliverable Nature	Other
Dissemination Level	Public
Contractual Date of Delivery	Project Month 8(May 2016)
Resubmission Date	04.09.2017
Description of Deliverable	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).

Document Control Information

Document	Title:	E-CAM Software Development Tools
	ID:	D6.2
	Version:	As of September 4, 2017
	Status:	Accepted by Executive Board
	Available at:	https://www.e-cam2020.eu/deliverables
	Document history:	Internal Project Management Link
Review	Review Status:	Reviewed
Authorship	Written by:	Alan Ó Cais(JSC)
	Contributors:	Dominic Tildesley (EPFL)
	Reviewed by:	Jony Castagna (STFC), Catarina Mendonça (EPFL)
	Approved by:	Catarina Mendonça (EPFL)

Document Keywords

Keywords:	E-CAM, HPC, CECAM, Materials, ...

September 4, 2017

***Disclaimer**:This deliverable has been prepared by the responsible Work Package of the Project in accordance with the Consortium Agreement and the Grant Agreement. It solely reflects the opinion of the parties to such agreements on a collective basis in the context of the Project and to the extent foreseen in such agreements.*

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

Contents

List of Figures

Executive Summary

This deliverable describes the current status (as of the deliverable due date) 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. 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.

This document is a companion to the services that E-CAM provides and focuses primarily on the E-CAM GitLab service since it is this service that supports the specific features of software development, documentation and maintenance. This service allows the project to offer a complete set of additional services:

- Version control
- Code review,
- GitLab Issue Tracker (used for bug tracking and maintenance as well as code review and feature requests),
- GitLab CI for continuous integration and delivery.
- Single sign-on for other E-CAM services via OAuth2
- Application website hosting (via GitLab Pages)
- GitLab repository wikis

Some useful low-level software is also highlighted, in particular the software modules developed in the first ESDW in the context of the Electronic Structure Library. We also highlight EasyBuild to manage software dependencies, system portability and the delivery of software to industrial partners. Further collaborative development projects between the various WPs of E-CAM with respect to low-level libraries are also in the early stages of development (as of the time of delivery, May 2016).

1 Introduction

One of 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.

This document is a companion to the services that E-CAM provides and focuses primarily on the E-CAM GitLab service since it is this service that supports the specific features of software development, documentation and maintenance.

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 Target Groups

The on-line services E-CAM provides are in place to facilitate the implementation of *best practices* as defined in Deliverable 6.1 of WP6 of E-CAM. In practice, this means facilitating version control; code review; test-driven design; source code and application documentation; and cross-platform portability.

While the services are specifically put in place in order for the project to achieve it's internal goals, they are open to all users from the extended E-CAM community. More specifically, the target groups for the tools and work-flow described in this deliverable are:

- post-docs working in our pilot projects;
- post-docs and students participating to our Extended Software Development Workshop (ESDW) events;
- senior scientists involved in E-CAM;
- senior scientists belonging to other communities related to E-CAM (eg. the CECAM community[1]);
- scientists working in industry, which are members of E-CAM.

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.

[1] For example, several software development projects from the Electronic Structure Library community ESL are stored in a GitLab repository hosted by E-CAM.

2 Software Development Tools

The on-line resources provided for the use of the E-CAM community are intended to satisfy the typical requirements of software development using best practices taken from existing open source projects (of similar scale). The services provided are intended to facilitate and simplify (as much as possible) the development, documentation and management of software applications created by the E-CAM community, including the project itself.

2.1 Software Development

Software tools for revision control are essential for the organization of multi-developer projects. Version control within projects connected to E-CAM is recommended to be done via Git, a standard utility available on Linux distributions. Git is not necessarily superior to other options but is chosen because of the eco-system surrounding the tool, mostly due to the use of the tool for the LINUX kernel and the success of hosting platforms such as GitHub and GitLab.

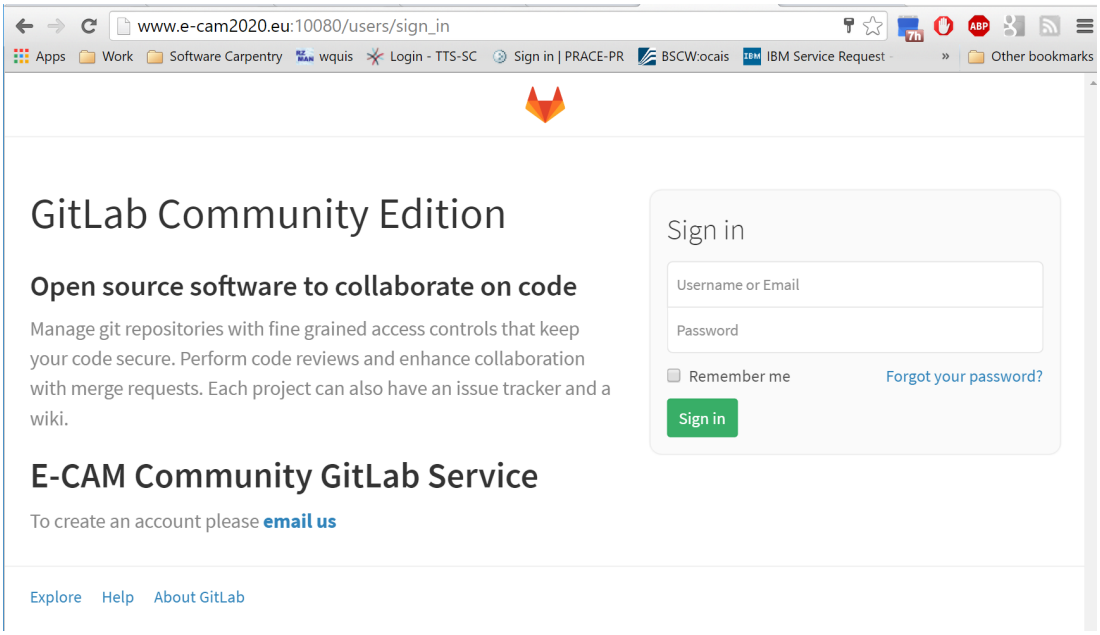

Figure 1: The GitLab service provided by E-CAM

In the case of the E-CAM community development projects, the hosting of project repositories is primarily done via the E-CAM GitLab service, see Fig.1. Hosting via GitHub is also possible (GitHub is a well known repository hosting service which is free to use for public open-source projects) via the E-CAM organisation there, though E-CAM provides no additional services through GitHub. The GitHub E-CAM organisation site is shown in Fig.2.

The E-CAM GitLab service is the preferred hosting method as this allows the project to offer a complete set of additional services:

- Code review,
- GitLab Issue Tracker (used for bug tracking and maintenance as well as code review and feature requests),
- GitLab CI for continuous integration and delivery.
- Single sign-on for other E-CAM services via OAuth2
- Application website hosting (via GitLab Pages)
- GitLab repository wikis

In addition, hosting our own GitLab service provides E-CAM with the capability of having unlimited private repositories for the E-CAM community. This is a necessary addition as repositories of some potential use cases may require this protection due to licence limitations on the source code (or similar restrictions).

Public repositories can be mirrored between GitLab and GitHub, providing an implicit back-up facility and leaving the repository owners free to chose which platform to prefer for management, distribution and issue tracking. A list of existing hosted repositories of E-CAM are available at https://gitlab.e-cam2020.eu/explore/projects.

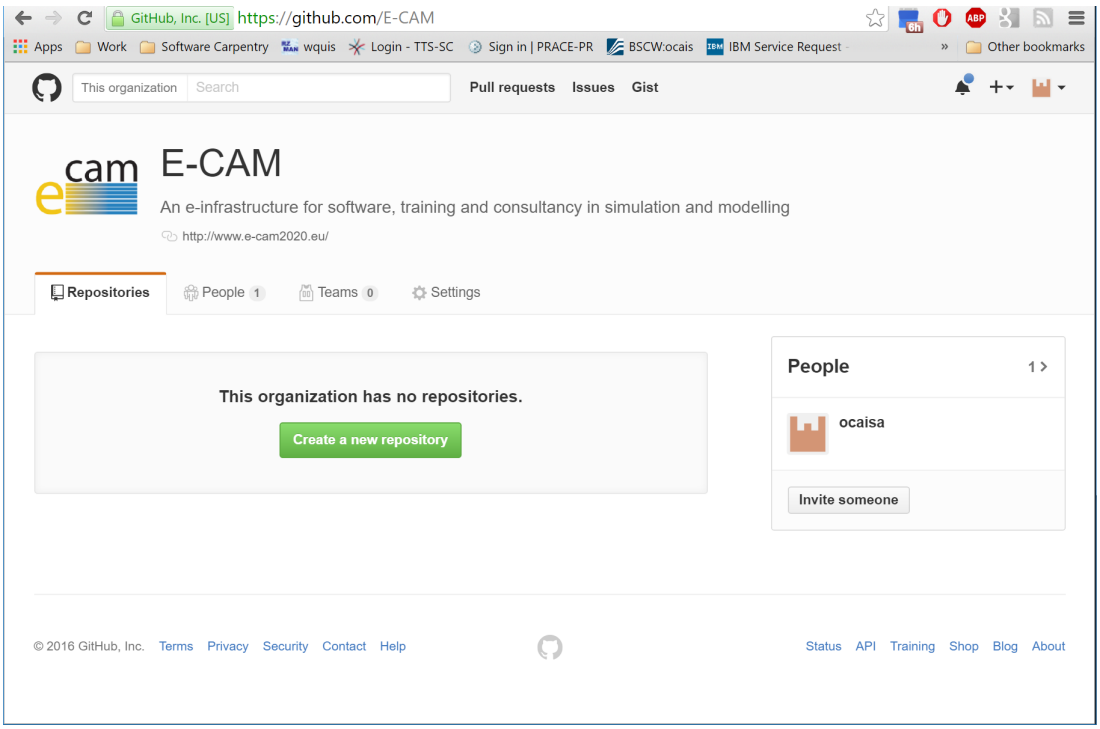

Figure 2: The GitHub organistion provided by E-CAM

2.1.1 Low-level Software

We outline here the contributions of E-CAM to providing *scalable* software libraries that can be leveraged by a large fraction of the E-CAM target communities. The libraries mentioned are intended to be leveraged during the development process of new applications that span most of the electronic structure community, but the main focus are codes that implement Density Functional Theory, which can be used to determine a wide range of properties of atoms, molecules, and materials.

In addition we highlight applications (supported by E-CAM) that facilitate the portability and optimisation of applications developed in this context.

For developers considering accelerators, we suggest that they be exposed, in an ESDW context, to other developers who have already gone down this path. One of our programmers is a GPU expert and for the KNL architecture vectorisation is key (combined with OpenMP and MPI for inter-node communication). These architectures are currently the most commonly found accelerators.

Electronic Structure Library

From a software perspective, the Electronic Structure community is the most well developed in the E-CAM target communities with a significant number of mature community codes. The Electronic Structure Library is a project to build a community-maintained library of software of use for electronic structure simulations (touching on the target communities of three centres of excellence in E-CAM, MaX and NOMAD). The goal is to create an extended library that can be employed by everyone for building their own packages and projects. It consists of entries documenting functionalities, algorithms, interfaces, standards and pieces of code ranging from small routines for performing simple tasks, all the way up to complete libraries. Although wiki entries may document software maintained elsewhere, the ESL also provides a software development infrastructure for internally developed projects. This infrastructure also permits the regular building of automatic documentation, for both internal and external projects.

The ambition of the ESL is to segregate layers of functionality within modules which are general, standardised and efficient. In this way, new ideas, and new science, can be coded by scientists without needing to rewrite functionalities that are already well-established, and without needing to know more software engineering than science. In other words, we want to separate the coding effort for cutting-edge research from the software infrastructure it rests on top of, which needs maintaining and rewriting at every step of the hardware race.

ESL development is being carried out in a series of coding-based workshops which are being supported by E-CAM. The first of such workshops was held in Spain in June 2016 centred on the development of new libraries revolving

around the broad theme of solvers. Work was carried out on three specific library bundles, all relating to crucial parts of any density-functional theory code:

- Kohn-Sham eigensolvers: Development for the ESL is already underway in this area. New steps forward will be taken through a new "Electronic Structure Infrastructure" project, in direct coordination with ESL and also connecting U.S. based researchers to the wider ESL effort. ELSI brings together several different approaches to solving or circumventing the Kohn-Sham eigenvalue problem, including a density matrix solver (libOMM), a resolvent-based method (PEXSI), and a scalable diagonalization method (ELPA), and will be open to integrating other open source solvers that are compatible and/or complementary.
- Poisson solvers: similarly to the eigensolvers, the aim is to implement, in a single package, several different algorithms of use in different situations, providing a unified and clean interface for the user. Special attention will go to allowing different FFT back ends to be connected to the library.
- Atomic solvers: this is one of the most notable examples of replicated development in different code bases, with no modern independent library available. The aim is to create a new library with a modern, coherent list of features, ultimately targeting spherical Hartree-Fock, semilocal and hybrid DFT, and different optional relativistic levels of theory. It will also be connected to LibXC for immediate access to many different exchange-correlation functionals (and, in doing so, will avoid replicated development).

Modules related to these developments were the subject of Deliverable 2.1 of WP2. Further development of these libraries are targets of future ESDWs. It is expected that these modules will also be of use to the other WPs.

Portability and Optimisation

Development with extreme-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.

Standards for interoperability of software written in different languages have been recommended in Deliverable 6.1 of WP6 of E-CAM with C recommended as an effective lingua franca. The online version of these Technical Guidelines (still under development at the time of submission) will include considered recommendations on optimised mathematical libraries, interface design, I/O, performance analysis, etc. All of these greatly impact the achievable performance and scalability of application codes.

EasyBuild is leveraged by E-CAM to ease portability issues between compilers, MPI implementations and math library functionalities. It can also encode build information for accelerators such as GPUs and many-core coprocessors where the target platform supports this functionality. EasyBuild provides a dependency resolution layer for build procedures that can ensure reproducibility of software installations. Finally, it also provides an effective software delivery mechanism to our industrial partners.

2.1.2 Integrated Development Environment

The usage of an Integrated Development Environment (IDE) is strongly advised when developing large codes. In particular, Eclipse is free software developed originally for JAVA applications, but since extended to C, C++, Python and Fortran languages. Moreover, Eclipse is well integrated in a large ecosystem, with version control tools, editors and tools for Parallel Performance Analysis and Debugger available on the Eclipse Marketplace.

In particular, the Eclipse Parallel Tools Platform (PTP) project provides an integrated development environment to support the development of parallel applications written in C, C++, and Fortran. PTP provides support for the MPI, OpenMP and UPC programming models, as well as OpenSHMEM and OpenACC. A wide range of batch systems and runtime systems, including PBS/Torque, LoadLeveler, GridEngine, Parallel Environment, Open MPI, and MPICH2 can be used to launch and remotely debug your applications.

Finally, an NVidia variation of Eclipse, the NSight Eclipse Edition is particularly useful for NVidia GPU programming (both in CUDA and OpenACC languages) allowing a very flexible and powerful environment, especially when debugging applications with thousand of threads.

2.2 Software Documentation

A number of software source-code documentation tools are recommended (Doxygen for it's cross-language availability, Ford for Fortran, Sphinx for Python) however, given the diversity of software and development teams, no unique solution is likely. With the provision of the online version of this documentation delivered via the GitLab Pages service, E-CAM does need to commit to any single solution, instead providing a flexible documentation hosting environment.

2.2.1 E-CAM Module Documentation

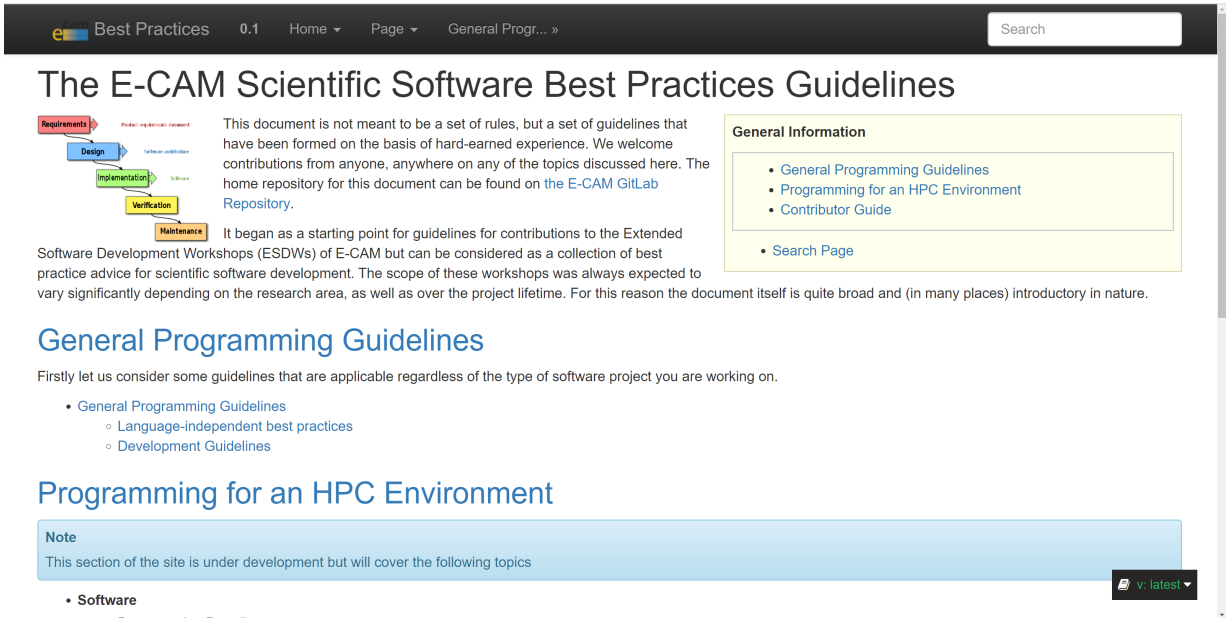

Figure 3: The Best Practices site on readthedocs.org

E-CAM module documentation is done using ReStructured Text and online integration done via hooks into readthedocs.org, a method that we also recommend to our target groups. 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.

Extensive guidelines for how to contribute to E-CAM module documentation are provided (for example, the contribution guidelines to WP4)

2.2.2 Automated Extraction

GitLab Pages can be built automatically with each commit to the git repository. Examples of how to do this via the continuous integration feature of GitLab for many of these tools can be found on the GitLab repository https://gitlab.com/pages.

We encourage the overarching application documentation to be done using ReStructured Text where possible since this can be automatically rendered by GitLab/GitHub.

2.3 Software Maintenance

The GitLab Issue Tracker is a tool that allows for the tracking of the evolution of a new idea or the process of solving a problem. It allows the E-CAM developers to share and discuss proposals before and while implementing them. It is extensively used during the certification of E-CAM software modules and recommended to all other E-CAM projects.

It is also possible to keep *Issues* private, even for public projects. For instance, let's assume you have a public project but want to start a discussion on something you don't want to be public. With *Confidential Issues*, you can discuss private matters among the project members, and still keep your project public, open to collaboration.

2.3.1 Bug Tracking

Our bug tracking and general task management system revolves around the GitLab Issue Tracker service provided by E-CAM. *Issues* can, however, have endless applications such as:

- Reporting bugs and malfunction
- Asking questions
- Obtaining support
- Submitting feature proposals
- Discussing the implementation of a new idea
- Elaborating new code implementations

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 is provided via GitLabs own continuous integration infrastructure and is hooked directly into the GitLab service (and is already being used for the GitLab Pages feature). 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.

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 where appropriate, but other solutions may be more appropriate in some scenarios and all are supported by the GitLab Pages feature
- 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
- Documentation building

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

References

Acronyms Used

CECAM Centre Européen de Calcul Atomique et Moléculaire

HPC High Performance Computing

ESDW Extended Software Development Workshop

WP Work Package

URLs referenced

Page ii

https://www.e-cam2020.eu ... https://www.e-cam2020.eu
https://www.e-cam2020.eu/deliverables ... https://www.e-cam2020.eu/deliverables
Internal Project Management Link ... https://redmine.e-cam2020.eu/issues/1
a.ocais@fz-juelich.de ... mailto:a.ocais@fz-juelich.de
http://creativecommons.org/licenses/by/4.0 ... http://creativecommons.org/licenses/by/4.0

Page iii

Best Practices ... http://scientific-software-best-practices.readthedocs.io/en/latest/
readthedocs.org ... https://readthedocs.org/

Page 1

E-CAM GitLab service ... https://gitlab.e-cam2020.eu/
Code review ... https://about.gitlab.com/2017/03/17/demo-mastering-code-review-with-gitlab/
GitLab Issue Tracker ... https://docs.gitlab.com/ee/user/project/issues/
GitLab CI ... https://about.gitlab.com/features/gitlab-ci-cd/
OAuth2 ... https://docs.gitlab.com/ce/api/oauth2.html
GitLab Pages ... https://docs.gitlab.com/ee/user/project/pages/index.html
GitLab repository wikis ... https://docs.gitlab.com/ce/user/project/wiki/index.html
Electronic Structure Library ... http://esl.cecam.org/Main_Page
EasyBuild ... http://easybuild.readthedocs.io/en/latest/

Page 2

E-CAM GitLab service ... https://gitlab.e-cam2020.eu/
Deliverable 6.1 of WP6 of E-CAM ... https://www.e-cam2020.eu/wp-content/uploads/2016/05/D6.1-Guidelines-fo
pdf
ESL ... http://esl.cecam.org/Main_Page

Page 3

E-CAM GitLab service ... https://gitlab.e-cam2020.eu/
E-CAM GitLab service ... https://gitlab.e-cam2020.eu/
Code review ... https://about.gitlab.com/2017/03/17/demo-mastering-code-review-with-gitlab/
GitLab Issue Tracker ... https://docs.gitlab.com/ee/user/project/issues/
GitLab CI ... https://about.gitlab.com/features/gitlab-ci-cd/
OAuth2 ... https://docs.gitlab.com/ce/api/oauth2.html
GitLab Pages ... https://docs.gitlab.com/ee/user/project/pages/index.html
GitLab repository wikis ... https://docs.gitlab.com/ce/user/project/wiki/index.html
https://gitlab.e-cam2020.eu/explore/projects ... https://gitlab.e-cam2020.eu/explore/projects

Page 4

Electronic Structure Library ... http://esl.cecam.org/Main_Page
E-CAM ... https://www.e-cam2020.eu/
MaX ... http://www.max-centre.eu/
NOMAD ... https://nomad-coe.eu/

Page 5

"Electronic Structure Infrastructure" ... https://wordpress.elsi-interchange.org/
Deliverable 2.1 of WP2 ... https://www.e-cam2020.eu/wp-content/uploads/2017/01/D2.1_311016.pdf
Virtual Institute for High-Productivity Supercomputing ... http://www.vi-hps.org/
Scalasca ... http://www.scalasca.org/

Deliverable 6.1 of WP6 of E-CAM . . . https://www.e-cam2020.eu/wp-content/uploads/2016/05/D6.1-Guidelines-fo
pdf
Technical Guidelines . . . https://www.e-cam2020.eu/wp-content/uploads/2016/05/D6.1-Guidelines-for-web-1.
pdf
EasyBuild . . . http://easybuild.readthedocs.io/en/latest/
Eclipse . . . https://eclipse.org/ide/
Eclipse Marketplace . . . https://marketplace.eclipse.org/
Eclipse Parallel Tools Platform . . . http://www.eclipse.org/ptp/
NSight Eclipse Edition . . . https://developer.nvidia.com/nsight-eclipse-edition

Page 6

Doxygen . . . http://www.stack.nl/~dimitri/doxygen/
Ford . . . http://fortranwiki.org/fortran/show/FORD
Sphinx . . . http://www.sphinx-doc.org/en/stable/
GitLab Pages . . . https://docs.gitlab.com/ee/user/project/pages/index.html
Best Practices . . . http://scientific-software-best-practices.readthedocs.io/en/latest/
readthedocs.org . . . https://readthedocs.org/
readthedocs.org . . . https://readthedocs.org/
Read the Docs . . . https://readthedocs.org/
contribution guidelines to WP4 . . . http://e-cam-meso-and-multi-scale-modelling-modules.readthedocs.
io/en/latest/contributing.html
GitLab Pages . . . https://docs.gitlab.com/ee/user/project/pages/index.html
https://gitlab.com/pages . . . https://gitlab.com/pages
ReStructured Text . . . http://docutils.sourceforge.net/rst.html

Page 7

GitLab Issue Tracker . . . https://docs.gitlab.com/ee/user/project/issues/
GitLab Issue Tracker . . . https://docs.gitlab.com/ee/user/project/issues/

Page 8

GitLab Pages . . . https://docs.gitlab.com/ee/user/project/pages/index.html
GitLab-CI . . . https://about.gitlab.com/gitlab-ci/
Travis CI . . . https://travis-ci.org/
NetCDF . . . http://www.unidata.ucar.edu/software/netcdf/
Doxygen . . . http://www.stack.nl/~dimitri/doxygen/
GitLab Pages . . . https://docs.gitlab.com/ee/user/project/pages/index.html
Google style guides . . . https://github.com/google/styleguide
cpplint . . . https://github.com/google/styleguide/tree/gh-pages/cpplint
Pylint . . . https://www.pylint.org/
SWIG . . . http://www.swig.org/
doxy2swig . . . https://github.com/m7thon/doxy2swig
pytest . . . http://pytest.org/latest/
CMake . . . https://cmake.org/

Citations