How to install a developer environment of PowSyBl and iPST

Christian Biasuzzi – Tech Rain
christian.biasuzzi@techrain.eu

PowSyBl/iPST projects

- **Source code available on GitHub, since April 2016:**
  - https://github.com/itesla
  - https://github.com/powsybl
- **License: MPL 2.0**
  - Permissive license: covered code must remain under the MPL, can be mixed with code under different licenses.
- **GitHub is a popular git projects code hosting service**
  - Hosts both private and open-source projects
  - Web based user interface, to access code under version control

PowSyBl/iPST projects

→ 4 GIT repositories:

- powsybl-core: https://github.com/powsybl/powsybl-core
  - IIDM (iTesla Internal Data Model), contingencies, iAL (iTesla Action Language), load-flow&simulation API, computation layer, data converters (CIM, UCT, AMPL), standard security analysis, AFS
- IPST: https://github.com/itesla/ipst
  - Online & offline workflows, sampling, MCLA (Monte Carlo Like Approach), simulators integrations, WCA (Worst Case Approach), Historical database
- IPST-ANSIBLE: https://github.com/itesla/ipst-ansible
  - Scripts to install the platform and its requirement on a target Linux
- CGMES: https://github.com/itesla/CGMES
  - CGMES data converter

PowSyBl/iPST projects

➔ Code statistics

- powsybl-core: ~160k lines of code (75% of code coverage)
- IPST: ~110k lines of code (13% of code coverage)
- CGMES: ~20k lines of code

➔ Project activity overview

- ~20 contributors
- Powsybl-core: 1165 commits; ipst: 983 commits

How to contribute

➔ **Everyone is welcome to contribute in different areas, by:**

- Improving existing code
- Creating new modules, e.g.
  - Computation modules
  - Converters
  - user interfaces
  - …
- Providing suggestions and feedbacks

How to contribute

➔ **Usually, to contribute to an open-source git project:**

- A copy (clone) of the source code repository must be created in the local development environment.
- Here, changes can be made and committed locally, creating a revision history, allowing modifications to be tracked and rolled back, if needed.
- Eventually, changes committed locally can be submitted (pushed) to the remote project, to be officially released.

How to contribute

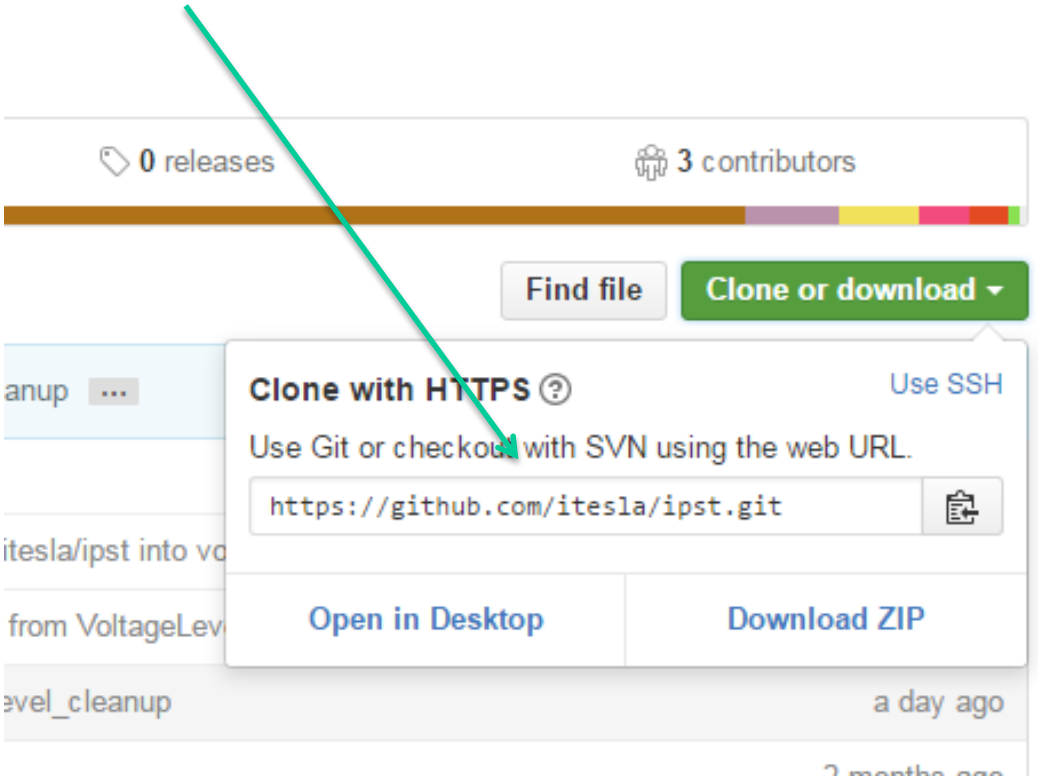
➔ **These tasks can be performed using git enabled tools: e.g.**

- Command line based tools https://git-scm.com (Linux, Windows, OSX)
- Most modern IDEs provide git integration: Eclipse, IntelliJ, etc.
- Dedicated GUI clients, e.g. GitHub Desktop: (https://desktop.github.com Windows 7+ and OSX)

How to contribute

➔ **For example, to create a local copy of the iPST project using the command line :**

```
git clone https://github.com/itesla/ipst.git
```



How to contribute

- **In GitHub only the project owner/maintainer can change directly the project's official "branch" (master)**
- **So, to contribute to iPST/PowSyBl project, a developer:**
  - Must have an active account in GitHub (free)
  - The GitHub account must have write permissions on the project (granted by the project maintainers, currently only RTE employees)
  - Must provide his changes in a separate branch and ask the project maintainer to merge it in the master, by submitting a "Pull Request" (PR)

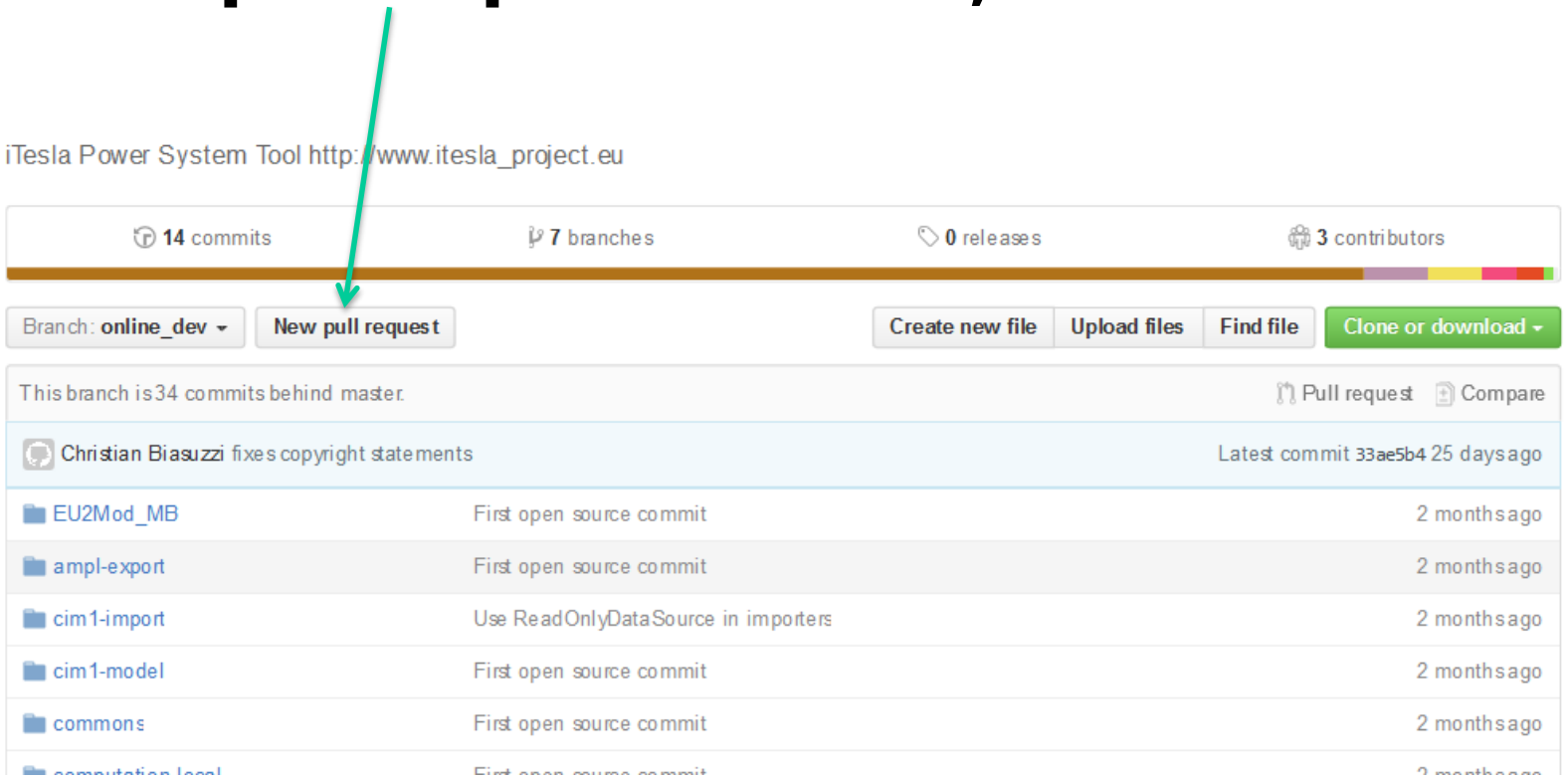
How to contribute

➔ Before submitting the code to GitHub:

- Check that the new code "fits" the projects
  - No compilation errors
  - No errors or failures in the automatic tests
  - Code style ok: checkstyle (http://checkstyle.sourceforge.net/) is used for a static code analysis (e.g. naming convention for attributes and methods, packages imported, etc.)
- Contributors are kindly requested to check also
  - That copyright, license and authorship are included, and that the original license and the original copyright statements are not removed from changed source code (new copyrights can be added to the existing ones)
  - To not include private/sensible data, binary files...
  - GitHub recommends repositories to be kept under 1Gb; in addition, there is a 100Mb strict limit on single files

How to contribute

➔ “New pull request” button, to start a new PR



How to contribute

➔ When a PR is submitted to the iPST project:

- Travis, an automatic Continuous Integration (CI) service (https://travis-ci.org/) checks that there are no build and tests errors; in case of errors changes are rejected
- Code coverage (percentage of lines actually run during tests) is computed and tracked using the Coveralls service (https://coveralls.io). If the coverage decreases below a fixed threshold , changes are rejected
- When Travis and Coveralls are both OK, the project maintainer then reviews the contribution checking that it is OK to be included in the master branch, and finalizes the merging
  - For now, only RTE can merge PRs to the main branch. Others may be granted the right to merge (conditions to be defined)

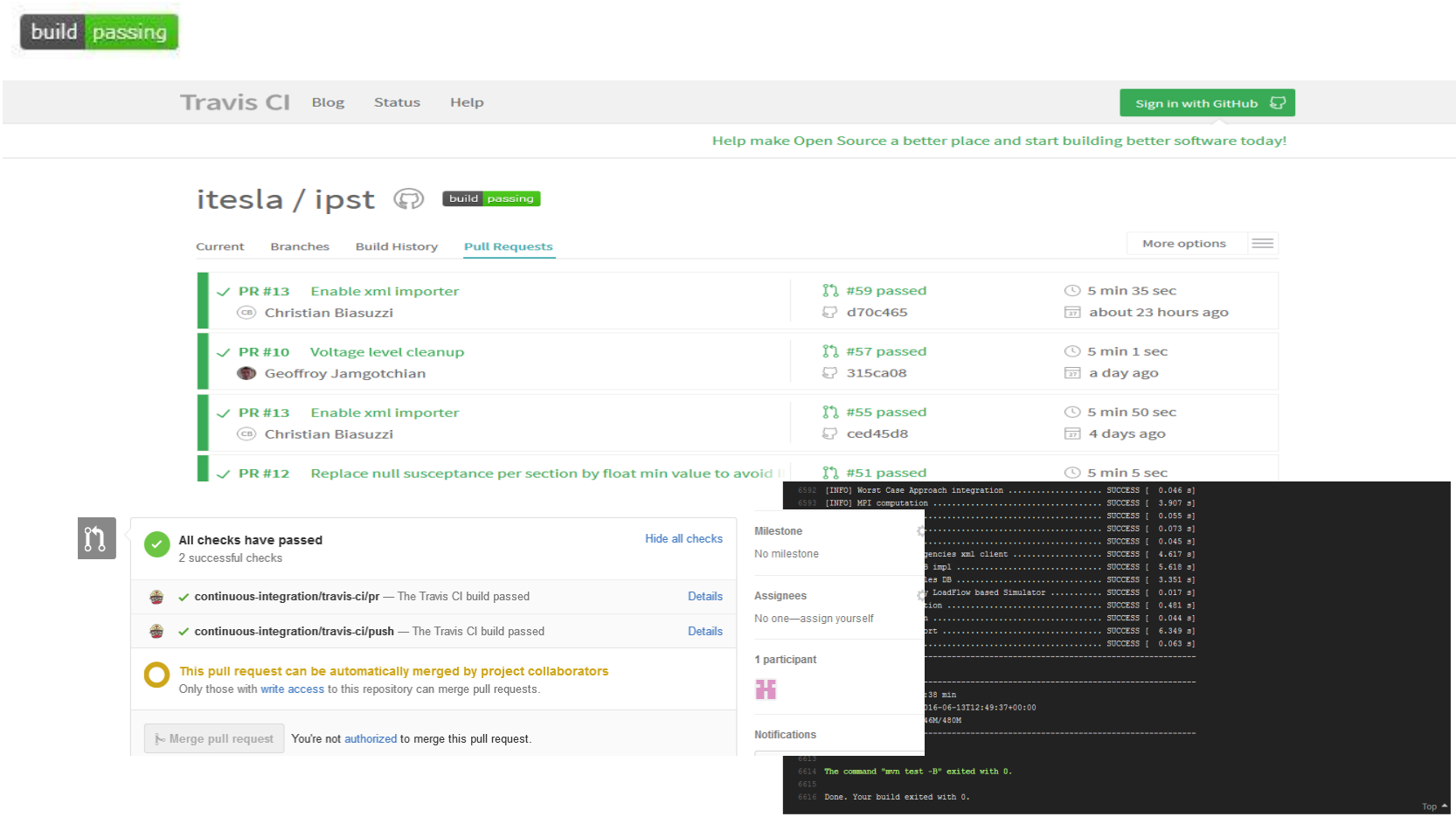
How to contribute

➔ **When a PR is submitted to the PowSybl project:**

- Travis
- AppVeyor: Continuous Integration for Windows (https://www.appveyor.com/)
- Code coverage
- SonarQube: continuous inspection of code quality (https://www.sonarqube.org/), an automatic review of the code, based on rules, to find code smells, bugs and security vulnerabilities. If SonarQube find a blocking or critical issue, changes are rejected
- Project maintainers review
- All the checks needs to pass to finalizes the merging

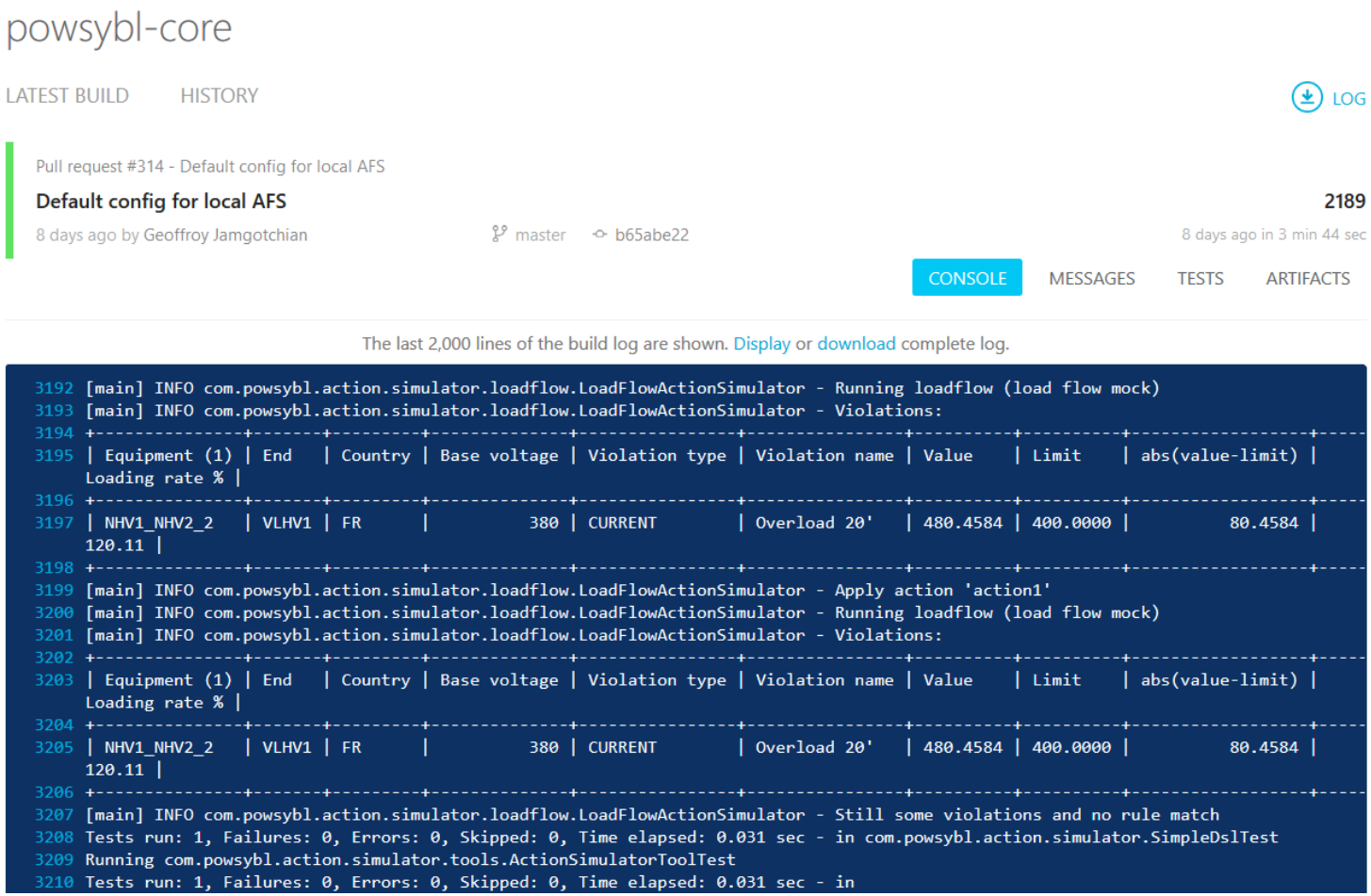
CI tool (TRAVIS)

➔ Travis details page: PR building steps status, build log



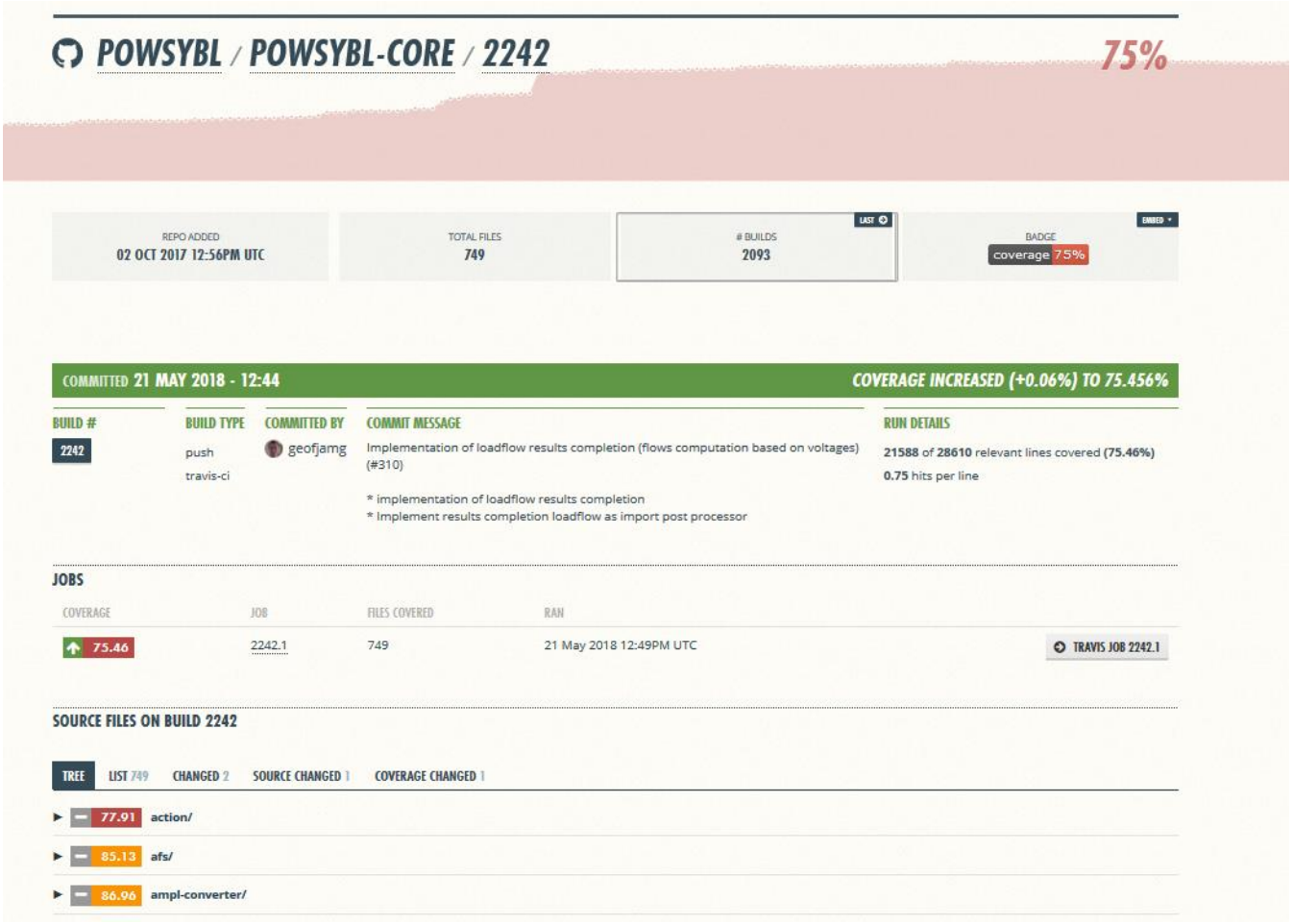
CI tool (AppVeyor)

➔ AppVeyor details page: PR building steps status, build log



Code coverage (Coveralls)

➔ Coveralls details page



Code quality (SonarQube)

➔ Issues summary, code review, rule applied

Java 8's "Files.exists" should not be used

Code Smell Major Main sources java8, performance Available

The `Files.exists` method has noticeably poor performance in JDK 8, and can

The same goes for `Files.notExists`, `Files.isDirectory` and `Files.isRegu`

Note that this rule is automatically disabled when the project's `sonar.java.sou`

Noncompliant Code Example

```
Path myPath;
if(java.nio.Files.exists(myPath)) {  // Noncompliant
 // do something
}
```

Compliant Solution

```
Path myPath;
if(myPath.toFile().exists()) {
 // do something
}
```

powsybl-ci commented 8 days ago Member

SonarQube analysis reported 2 issues

- 2 major

Watch the comments in this conversation to review them.

1 extra issue

Note: The following issues were found on lines that were not modified in the pull request. Because these issues can't be reported as line comments, they are summarized here:

1. ZipFileDataSource.java#L155: Replace this with a call to the "toFile().exists()" method

powsybl-ci reviewed 8 days ago View changes

commons/src/main/java/com/powsybl/commons/datasource/ZipFileDataSource.java

```
...  ...  @@ -72,17 +72,19 @@ public boolean exists(String suffix, String ext) throws I
72   72           return exists(DataSourceUtil.getFileName(baseName, suffix, ext));
73   73       }
74   74
75      -    private static boolean entryExists(Path zipFilePath, String fileName) th
     75   +    private static boolean entryExists(Path zipFilePath, String fileName) {
76   76           if (Files.exists(zipFilePath)) {
```

powsybl-ci 8 days ago Member

Replace this with a call to the "toFile().exists()" method

Reply...

iPST: platform installation

- **LINUX is currently required to run the platform (CentOS v6.x, v7.x, Fedora)**
- **iPST/PowSyBl projects' readme files explain how to build and install the platform.**

➔ Essentially:

1. Retrieve the source code for the platform repositories from GitHub, either by downloading projects .zips from the web UI or using the git command "git clone".
   e.g.
   ```
   git clone https://github.com/powsybl/powsybl-core.git
   git clone https://github.com/itesla/ipst.git
   ```
2. Compile the source code and install the platform: an install.sh script for each project takes care of building and installing the platform's components (powsybl-core must be installed before ipst)

iPST: platform installation

➔ **OS packages / compilers / runtimes required (* = only needed for specific iPST components)**

- C/C++development tools (gcc, g++, make, cmake, git, etc)
- Oracle JDK8
- Apache Maven
- OpenMPI (from sources, with flag --enable-mpi-thread-multiple)
- MysqlDB / MariaDB (*)
- WildFly application server (*)
- MATLAB Runtime (MCR) (*)
- Integrated components (Hades, Eurostag and Intel Fortran Compiler, Dymola, …) (*)
- test data (*)
- …

iPST: platform installation

- Some of the requirements in the list could be already available, on a developer machine, the missing ones have to be installed before iPST.
- To help automate installation of (most of) the available requirements and the platform the **IPST-ANSIBLE** repository provides a set of scripts for Ansible, an open-source software provisioning / automation engine tool https://www.ansible.com/

- … Ansible has to be installed, though:
- http://docs.ansible.com/ansible/intro_installation.html#latest-release-via-yum
- See http://github.com/itesla/ipst-ansible page, for more details.

iPST: simple installation scenario

➔ Starting from the sources (powsybl-core v1.0.0 and ipst), install the platform on a Linux CentOS v7:

1. Retrieve and configure the iPST Ansible scripts from GitHub
2. Execute the installation scripts
3. Download the sample dataset (Nordic44 - 2015)
4. Test the installation running a load flow (using Hades) on the sample data

➔ **Nordic44-Nordpool: An Open Data Repository and a Data Processing Software Toolset for an Equivalent Nordic Grid Model Matched to Historical Electricity Market Data**

The raw and processed data files corresponding to the model are available as an open data set and documented in:

L. Vanfretti, S.H. Olsen, V.S. Narasimham Arava, G. Laera, A. Bidadfar, T. Rabuzin, Sigurd H. Jakobsen, J. Lavenius, M. Baudette, F.J. Gómez-López, An open data repository and a data processing software toolset of an equivalent Nordic grid model matched to historical electricity market data, Data in Brief, Available online 13 February 2017, ISSN 2352-3409, [http://dx.doi.org/10.1016/j.dib.2017.02.021] (http://www.sciencedirect.com/science/article/pii/S2352340917300409). The final published article can be downloaded directly from the journal website, it is an open access journal.

The model is first presented in:

L. Vanfretti, T. Rabuzin, M. Baudette, M. Murad, iTesla Power Systems Library (iPSL): A Modelica library for phasor time-domain simulations, SoftwareX, Available online 18 May 2016, ISSN 2352-7110, http://dx.doi.org/10.1016/j.softx.2016.05.001.

Generated case files for every hour of every day of 2015, available at:: https://zenodo.org/record/162907

➔ **RTE HADES** is a load flow tool provided by RTE for load flow calculations. A 64 bit Linux version of Hades is freely downloadable here: http://www.rte.itesla-pst.org . Commercial use is forbidden. The license is restricted to academic, R&D and trial uses as precised in the end user license agreement provided in the archive.

iPST: simple installation scenario

➔ Retrieve the Ansible scripts:

`e.g. git clone https://github.com/itesla/ipst-ansible.git`

➔ Configure the scripts, in the ipst-ansible directory

1. create file ipst-hosts

```
[ipst_hosts]
127.0.0.1  ansible_user=USER ansible_ssh_pass=PWD
ansible_become_user=ADMIN ansible_become_pass=ADMIN_PWD
ansible_become_method=sudo
```

2. Copy file `group_vars_ipst_hosts.example` to `group_vars/ipst_host`

Ref. ipst-ansible's readme, for additional configurations / optional modules.

➔ Install HADES

1. get HADES here: http://www.rte.itesla-pst.org
   (license agreement form required)
2. drop the HADES tar.gz file in: `~/Downloads`

iPST: simple installation scenario

➔ Execute the installation script

```
ansible-playbook -i ipst-hosts ./ipst.yml -e "hades_process=True"
```

Default target directory is : `~/ipst`

➔ retrieve the Nordic44 data set

```
ansible-playbook -i ipst-hosts ./ipst.yml -e "installNordic44=true“
-t Nordic44
```

Default target directory is : `~/samples/Nordic44/caserepo`

➔ run a LF on a Nordic44 file

```
~/ipst/bin/itools run-loadflow --case-file
~/samples/Nordic44/caserepo/CIM/SN/2015/01/01/20150101_0000_SN4_UX0.zi
p
```