Good (or bad) programming practices

I don’t say you can apply all of this on your current code….

But it would be great

Make readable code !

- You **already know**
- But **in large** code it is **not so easy**
- Some **optimization** make the code **hard to read**.
- Where is the limit ? **KIS** (Keep It Simple)
- Take care of **variable names** : a, b uv, uvv, uvw
  - I know, as **physicist** you **like them**…
  - **Some** of them, **ok**, but **not all**
  - Avoid cryptic names (eg **DaVinci**).
- Your **environment** and **methods** will help you to **make good code or not** .

```
#include   /*recall-the\    /-good--old-\    /IOCCC-days!\    */<unistd.h>
 typedef  unsigned/*int*/  short U;U(main)  [32768],n,r[8];  __attribute__((
# define  R(x)       A(r[  7-(n       >>x&  7)],       (n>>  x>>3       )%8)
#define   C(x)       (U*)  (((*            |IO|             -dpd
*/char*)  main       +(x)  )/*|            |CC|             ll*/
# define  A(v,       i)(i  ?i<2            ?C(v             ):i\
-4?v+=2,  C(i-       6?v-  2:v+       *C(v  -2))       :C(v  -=2)       :&v)
/*lian*/  constructor))U(  x)(){for(;;*r+=  2,*r+=!n?_exit(  write(2,"Illeg"
"al ins"   "truction ;-"    "(\n",24)),0:     n>>8==001?(      signed char



                )n*2   :548==n>>    6&&usleep     /**/(10
                )+n%  64==   4?0*  write  (r[7   /**/],C(
                *C(*  r)),   *C(*  r+2)   )+4:  /**/ n>>9
                ==63   &&--r[7-n/   64%8]?n%+  /**/  64*-
                2:0,         n>>6  ==47   ?*R( 0):n>>12==1?
                *R(0  )=*R   (+6)  :n>>  12==+       14?*
                R(0)   -=*R(2*3)    :0)n=*C(*        r);}
```

IOCC 2016 : endoh3

The right tools & technic

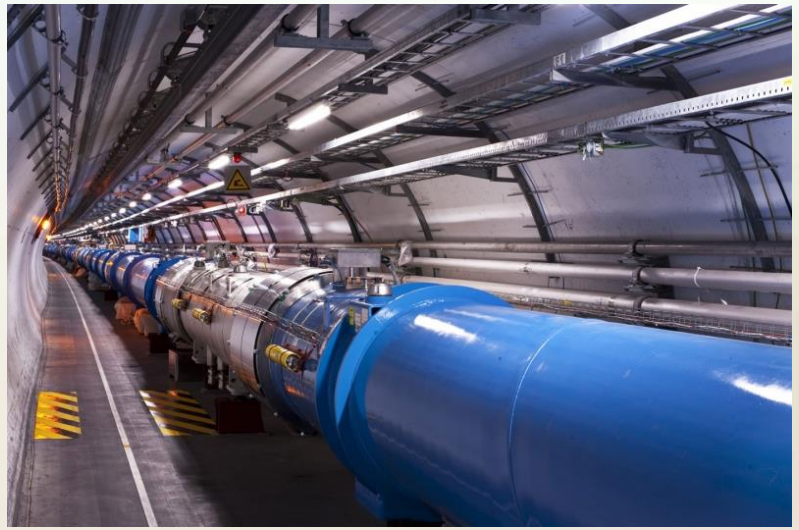
- A **religious & troll** slide..
- Can we build an accelerator with **pre-historic tools & technics** ?

Vi / emacs

Vi / emacs + config
Kdevelop / XCode

?

Eclipse ?



LHCb million lines of code

Version manager

- ~~Make a daily archive with famous researcher names~~
- Always use **a version manager** to track your changes
- Tools : **Git** or SVN
- Today, prefer git
  - **No** need of a **central server** to host the repository
  - Better use of branches
  - **Ease collaboration** with others
- On CERN, use **CERN gitlab** to host your git

```
#create the repo
git init

#Add all the current files
git add .

#Commit
git commit

#look on the history
git log

#look the history (another tool)
tig
```

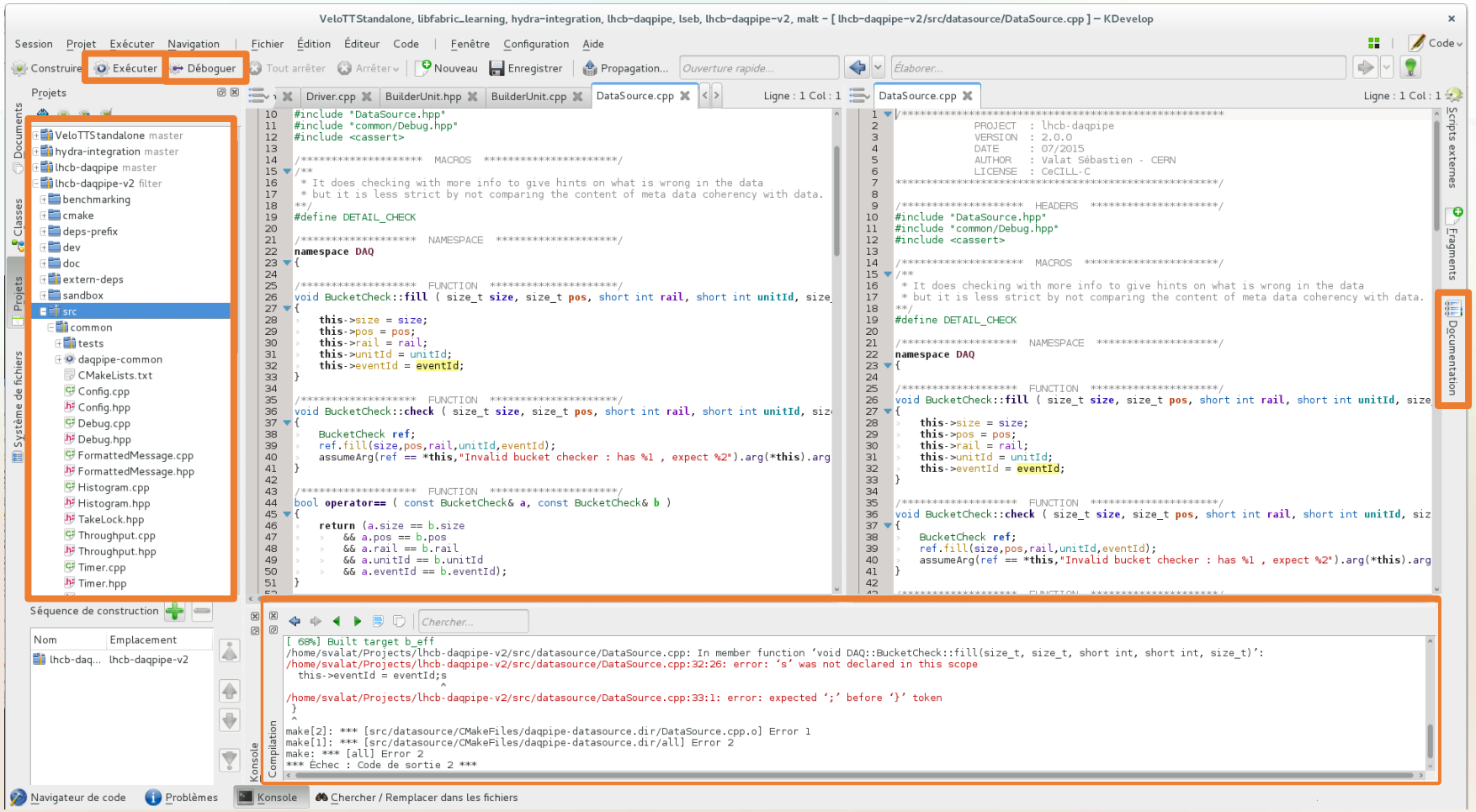
About building tool

- ~~Write your own script ?~~
- ~~*Use Makefile*~~
  - Ok for tiny projects
- Use **cmake** / autotools
- Think of the full **workflow of your code**
- Provide a **tutorial to build** your code
- If it is used by others, **think about packaging**.
- The **more you automate** the easier it will be to use **Continuous integration**.
  - But **KIS** & keep it **clear** & **understandable**
  - If you need 30 (active) minutes to build your project, **you fail**

My way to work

- Limit as most as possible **remote coding** (ssh + vi)
- Prefer to use local **IDE** (**kdevelop**, geany, visual studio, qt creator, atom)
  - It forces my code to **not being linked** to one **particular installation**.
- My **IDE** help me to ease **code refactoring**
  - Usefull to **maintain code quality**
- How to run localy ? (remote data)
- => Use **unit tests**

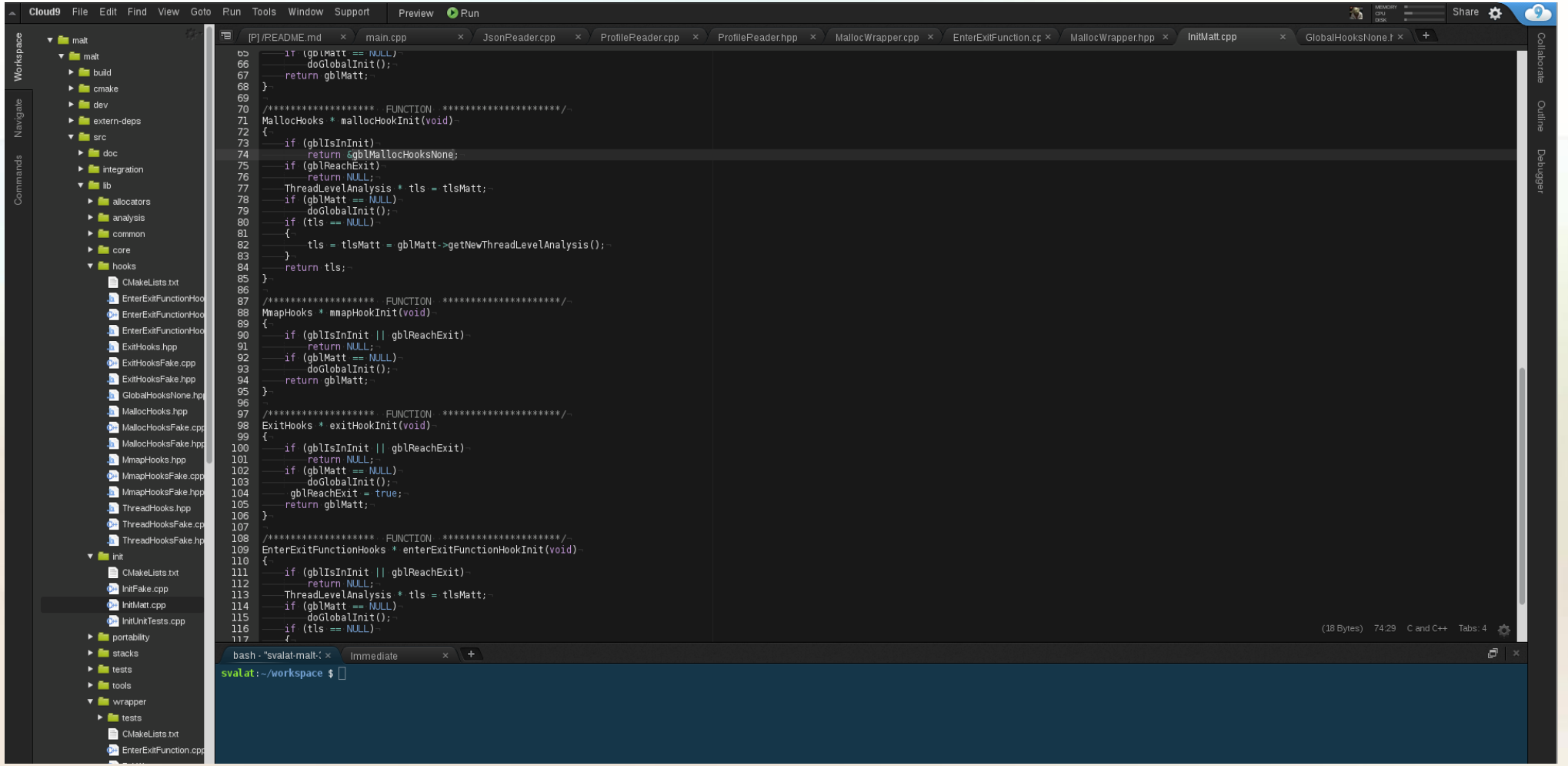
Why an IDE can help ?



My dream, a web IDE

- Launch a **server** on the **remote machine** (via ssh)
- **Forward** the SSH port locally (through the ssh connection)
- Connect with your **browser**
- You get a **nice GUI** + **remote compilation / completion**
- Can do **share working** on the **same session** for debugging.

The technic is there… but not open Cloud9



Making safer code : assert

- Use **assertions**
- An assertion is a **test** which is executed **only in debug mode** (suppressed by **–DNDEBUG**)

```
#include <assert.h> //or <cassert> in C++

UnitType Transport::getUnitType ( int rank, int unitId )
{
            //errors
            assert(unitId >=0 && unitId < DAQ_MAX_NODE_PER_PROCESS);
            assert(rank >= 0 && rank < this->getWorldSize());

            //action
            return unitTypes[rank * DAQ_MAX_NODE_PER_PROCESS + unitId];
}
```

- I also define **assume** in my codes :

```
#include “Debug.hpp”

UnitType Transport::getUnitType ( int rank, int unitId )
{
            //errors
            assume(rank >= 0, “Rank must be positive !”);
            …
}
```

To be safer => unit test

- Take a **small part (function, class)** of the code to be tested
- It **must not** depend on **100 objects**
- A **good code** is **testable**
- It force you to make **better code**
- It force you to **think about your API**
- It **help** you to **refactor** (you are sure to see what you break)

Example using GTest

- It also gives **example of usage**
- Ease debugging
- I'm using **GTest** for C++
- Similar pattern in all languages
- Do **not** try to **implement all cases**
- Add tests **when you encounter bugs**

```
TEST(DummyDataSource,isEventAvailable)
{
            Config config;
            config.dataSizeRandomRatio = 0;
            DummyDataSource datasource(&config);
            datasource.setId(0);

            EXPECT_TRUE(datasource.IsEventAvailable(0));
            EXPECT_FALSE(datasource.IsEventAvailable(100000));
}
```

```
Running main() from gmock_main.cc
[==========] Running 8 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 8 tests from DummyDataSource
[ RUN      ] DummyDataSource.getTotalSize
[       OK ] DummyDataSource.getTotalSize (142 ms)
[ RUN      ] DummyDataSource.isEventAvailale
/home/svalat/Projects/lhcb-daqpipe-
v2/src/datasource/tests/TestDummyDataSource.cpp:97: Failure
Value of: datasource.IsEventAvailable(100000)
  Actual: false
Expected: true
[  FAILED  ] DummyDataSource.isEventAvailale (138 ms)
[ RUN      ] DummyDataSource.operatorStream
[       OK ] DummyDataSource.operatorStream (0 ms)
[----------] 8 tests from DummyDataSource (1050 ms total)

[----------] Global test environment tear-down
```

Unit test / integration test

- If you need to instantiate **all your classes** => **integration test**
- You need some
- But **hard to maintain** if you do **only this**
- You can **mock** to **break dependencies**

Unit test & mocking

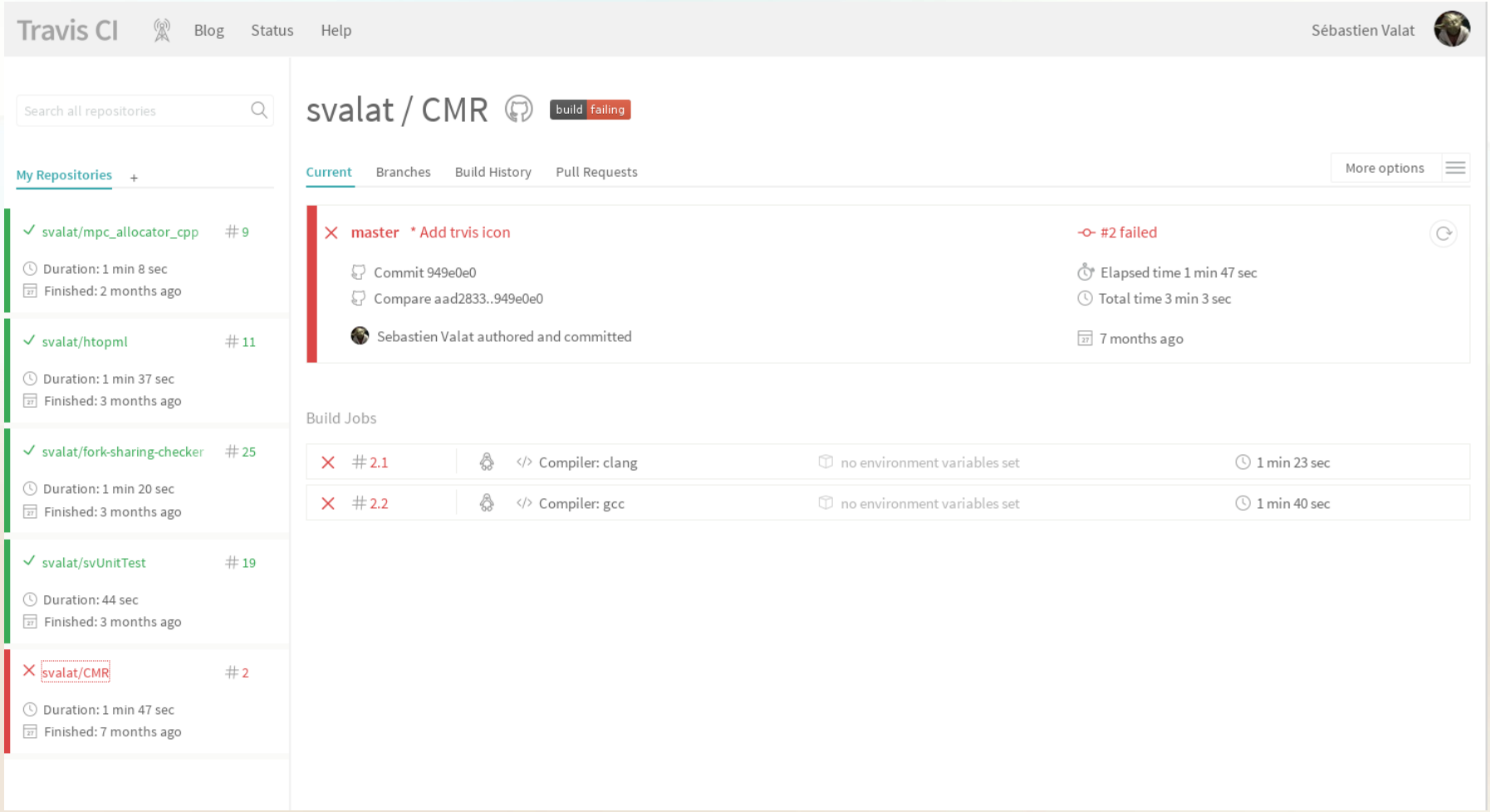
```
#include "gmock/gmock.h"  // Brings in Google Mock.

class MockTurtle : public Turtle {
    public:
          ...
          MOCK_METHOD0(PenUp, void());
          MOCK_METHOD0(PenDown, void());
          MOCK_METHOD1(Forward, void(int distance));
          MOCK_METHOD1(Turn, void(int degrees));
          MOCK_METHOD2(GoTo, void(int x, int y));
          MOCK_CONST_METHOD0(GetX, int());
          MOCK_CONST_METHOD0(GetY, int());
};
```

```
TEST(PainterTest, CanDrawSomething) {
    MockTurtle turtle;
    EXPECT_CALL(turtle, PenDown())
          .Times(AtLeast(1));

    Painter painter(&turtle);

    EXPECT_TRUE(painter.DrawCircle(0, 0, 10));
}
```

From unit test to continuous integration

- If you have unit test you can move to **continuous integration**
- Your **integration platform** will perform **test for each commit**
- You can be **notified by mail** in case of failure
- **Useful** for example if you support **multiple OS / use cases**
- Tools : **Gitlab-CI, Travis-CI** or **Jenkins**

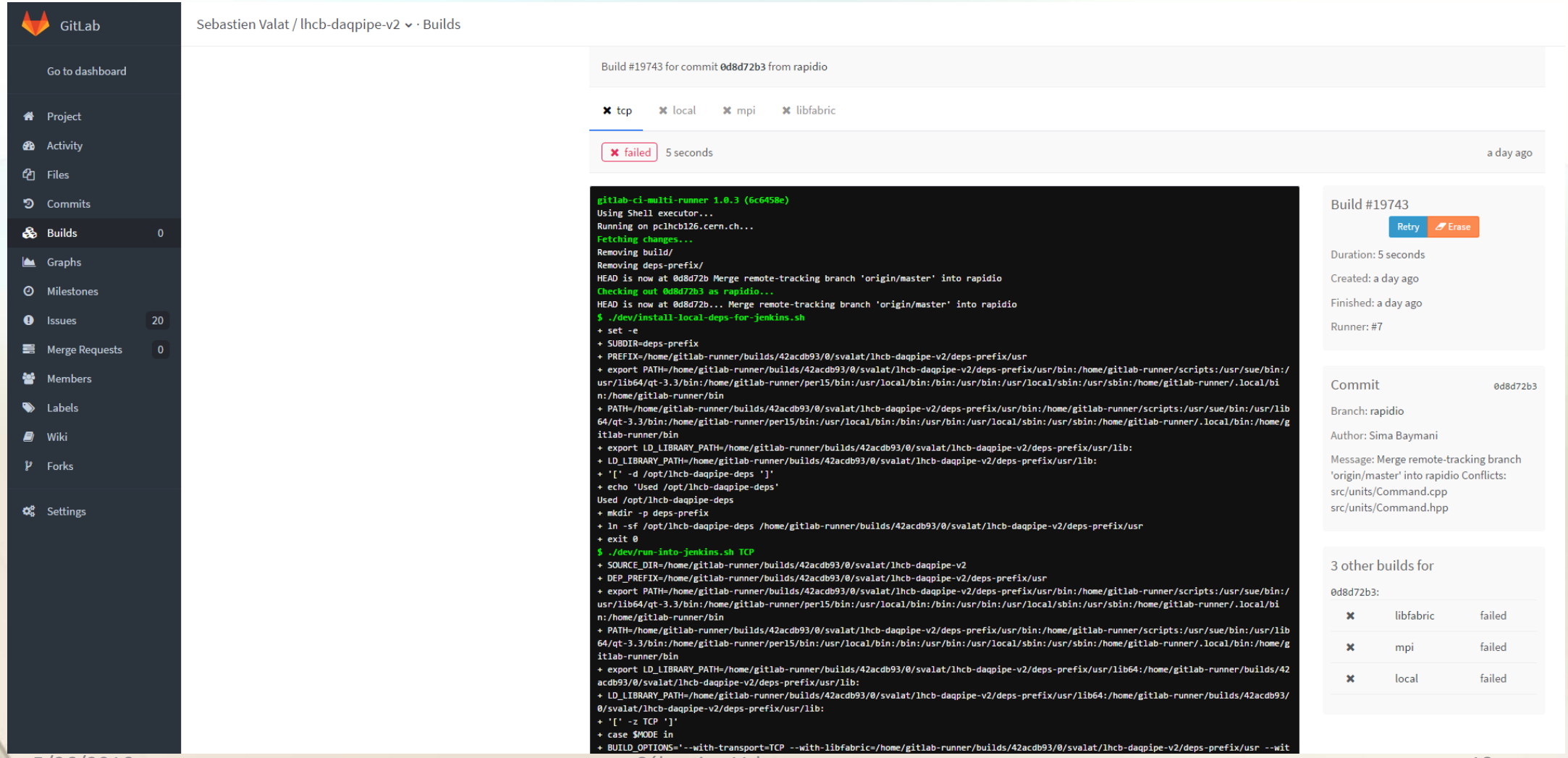
Example : Travis-CI



Quick error report

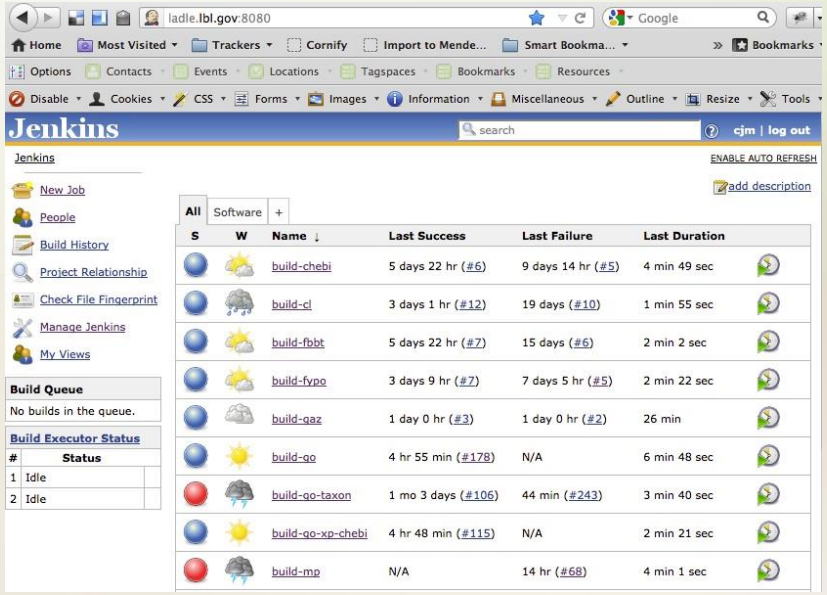
```
552       Start 29: TestMarkNoTransf
553 29/34 Test #29: TestMarkNoTransf .................   Passed    0.00 sec
554       Start 30: TestReplace
555 30/34 Test #30: TestReplace ......................   Passed    0.00 sec
556       Start 31: TestImplicitMul
557 31/34 Test #31: TestImplicitMul ..................   Passed    0.00 sec
558       Start 32: TestExtractLoops
559 32/34 Test #32: TestExtractLoops .................   Passed    0.00 sec
560       Start 33: TestReplaceAlias
561 33/34 Test #33: TestReplaceAlias .................***Failed    0.01 sec
562       Start 34: TestCodeTemplate
563 34/34 Test #34: TestCodeTemplate .................   Passed    0.00 sec
564
565 97% tests passed, 1 tests failed out of 34
566
567 Total Test time (real) =   0.15 sec
568
569 The following tests FAILED:
570         33 - TestReplaceAlias (Failed)
571 Errors while running CTest
572 make: *** [test] Error 8
573
574 The command "mkdir -p build && cd build && ../configure --enable-debug && make && make test"
    exited with 2.
575
576 Done. Your build exited with 1.
                                                                                        Top
```

Similar with Gitlab-CI



More details with Jenkins

- **Jenkins** is a **more advanced integration platform**
- It provide detailed **per test error report**
- Build **reports** from static / dynamic **code analysis tools** :
  - Valgrind
  - CppCheck
  - Rats
- You can use the **CERNFORGE** service to create an instance
- Can produce your **RPM**
- Manage **inter-project dependencies**



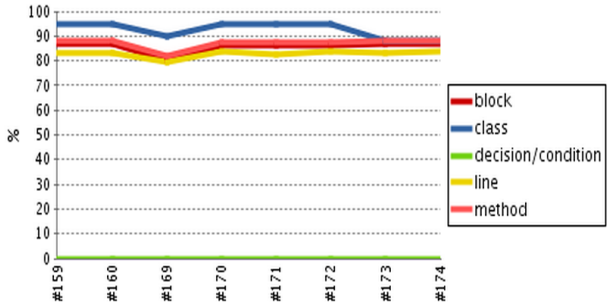
Jenkins – code coverage

Jenkins search tomhoward | log out

Jenkins › BWUnit › List › BWUnit-02-Test Ant 1.7.0 › #174 › Emma

- Back to Project
- Status
- Changes
- Console Output [raw]
- Edit Build Information
- Delete Build
- Parameters
- Environment Variables
- Mercurial Build Data
- Test Result
- **Coverage Report**
- Rebuild
- Downstream build view
- JS Widgets
- Previous Build

Emma Coverage Report



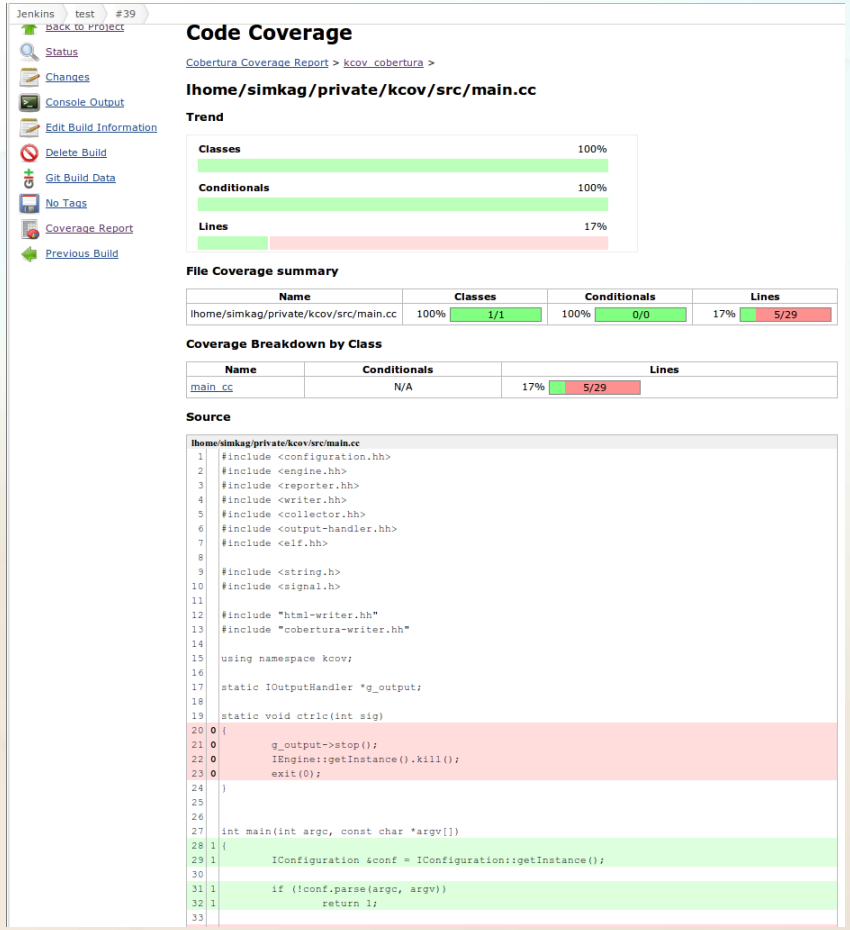
Overall Coverage Summary

name	class	method	block	line
all classes	87.9% 189/215	87.9% 189/215	86.8% 197/227	83.7% 923/1103

Coverage Breakdown by Package

name	class	method	block	line
Currency/main/WSXServiceBroker	100.0% 2/2	100.0% 2/2	100.0% 2/2	62.5% 5/8
Currency/main/WSXServiceBroker/DataFormatTransformer	100.0% 3/3	100.0% 3/3	100.0% 3/3	90.9% 10/11
Currency/test/IntegrationContractTestSuite/CurrencyRateTestCase	100.0% 3/3	100.0% 3/3	100.0% 3/3	92.3% 12/13
Currency/test/UnitTestSuite/ConvertTestCase	100.0% 4/4	100.0% 4/4	100.0% 4/4	90.5% 19/21
Currency/test/UnitTestSuite/GetRateTestCase	100.0% 4/4	100.0% 4/4	100.0% 4/4	90.0% 18/20
Currency/test/stubs	100.0% 1/1	100.0% 1/1	100.0% 1/1	100.0% 2/2
HelloWorld	100.0% 1/1	100.0% 1/1	100.0% 1/1	100.0% 2/2
HelloWorld/HelloTestSuite/HelloTestCase	100.0% 2/2	100.0% 2/2	100.0% 2/2	100.0% 8/8

Jenkins – code coverage & valgrind



Jenkins › test › #39

Back to Project
Status
Changes
Console Output
Edit Build Information
Delete Build
Git Build Data
No Tags
Coverage Report
Previous Build

Code Coverage

Cobertura Coverage Report > kcov_cobertura >

lhome/simkag/private/kcov/src/main.cc

Trend

Classes	100%
Conditionals	100%
Lines	17%

File Coverage summary

Name	Classes	Conditionals	Lines
lhome/simkag/private/kcov/src/main.cc	100% 1/1	100% 0/0	17% 5/29

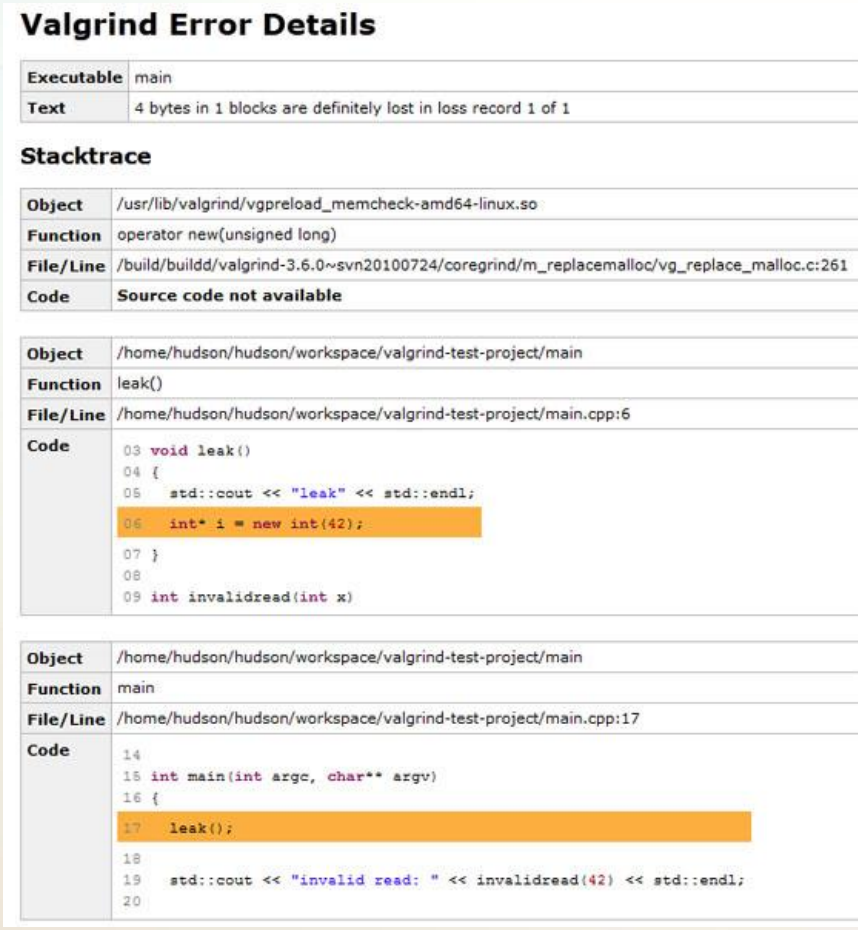
Coverage Breakdown by Class

Name	Conditionals	Lines
main_cc	N/A	17% 5/29

Source

lhome/simkag/private/kcov/src/main.cc

```
1   #include <configuration.hh>
2   #include <engine.hh>
3   #include <reporter.hh>
4   #include <writer.hh>
5   #include <collector.hh>
6   #include <output-handler.hh>
7   #include <elf.hh>
8
9   #include <string.h>
10  #include <signal.h>
11
12  #include "html-writer.hh"
13  #include "cobertura-writer.hh"
14
15  using namespace kcov;
16
17  static IOutputHandler *g_output;
18
19  static void ctrlc(int sig)
20 0{
21 0       g_output->stop();
22 0       IEngine::getInstance().kill();
23 0       exit(0);
24  }
25
26
27  int main(int argc, const char *argv[])
28 1{
29 1       IConfiguration &conf = IConfiguration::getInstance();
30
31 1       if (!conf.parse(argc, argv))
32 1               return 1;
33
```



Valgrind Error Details

Executable	main
Text	4 bytes in 1 blocks are definitely lost in loss record 1 of 1

Stacktrace

Object	/usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so
Function	operator new(unsigned long)
File/Line	/build/buildd/valgrind-3.6.0~svn20100724/coregrind/m_replacemalloc/vg_replace_malloc.c:261
Code	**Source code not available**

Object	/home/hudson/hudson/workspace/valgrind-test-project/main
Function	leak()
File/Line	/home/hudson/hudson/workspace/valgrind-test-project/main.cpp:6

Code

```
03 void leak()
04 {
05   std::cout << "leak" << std::endl;
06   int* i = new int(42);
07 }
08
09 int invalidread(int x)
```

Object	/home/hudson/hudson/workspace/valgrind-test-project/main
Function	main
File/Line	/home/hudson/hudson/workspace/valgrind-test-project/main.cpp:17

Code

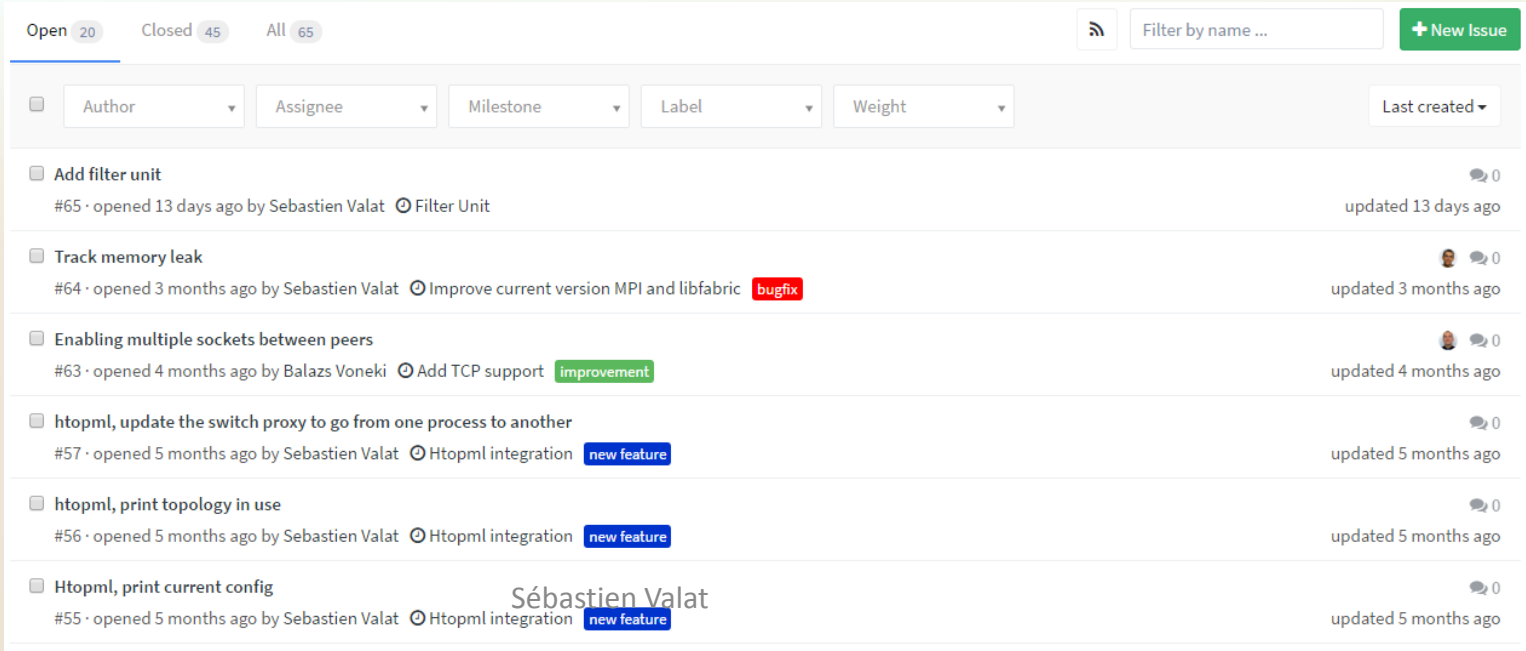
```
14
15 int main(int argc, char** argv)
16 {
17   leak();
18
19   std::cout << "invalid read: " << invalidread(42) << std::endl;
20
```

Refactoring

- Your needs **evolve** (especially in research field)
- If your **code become inadequate** => **refactor**
- It is easier to do **using tools**
- Having **unit tests is certainly safer**
- **Without** unit **test**, this is a **risky challenge**

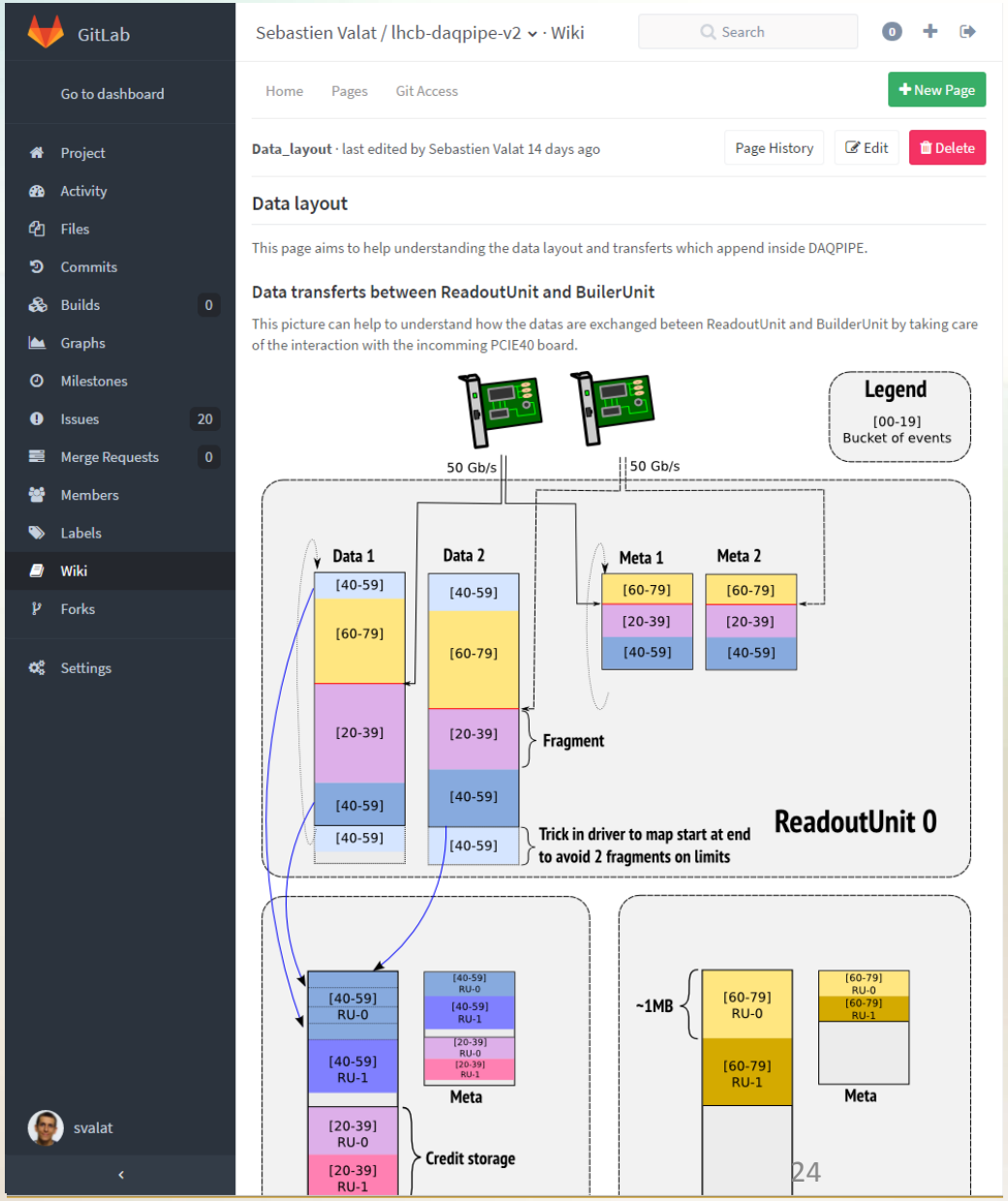
Ticket system

- If you are **many on the project**
- But if you are rigorous : **also alone**.
- It **save you** when you come back after a **couple of month**
- Make you **think in advance** on what **need to be done**.
- Use a ticket system (**Gitlab** or Jira)



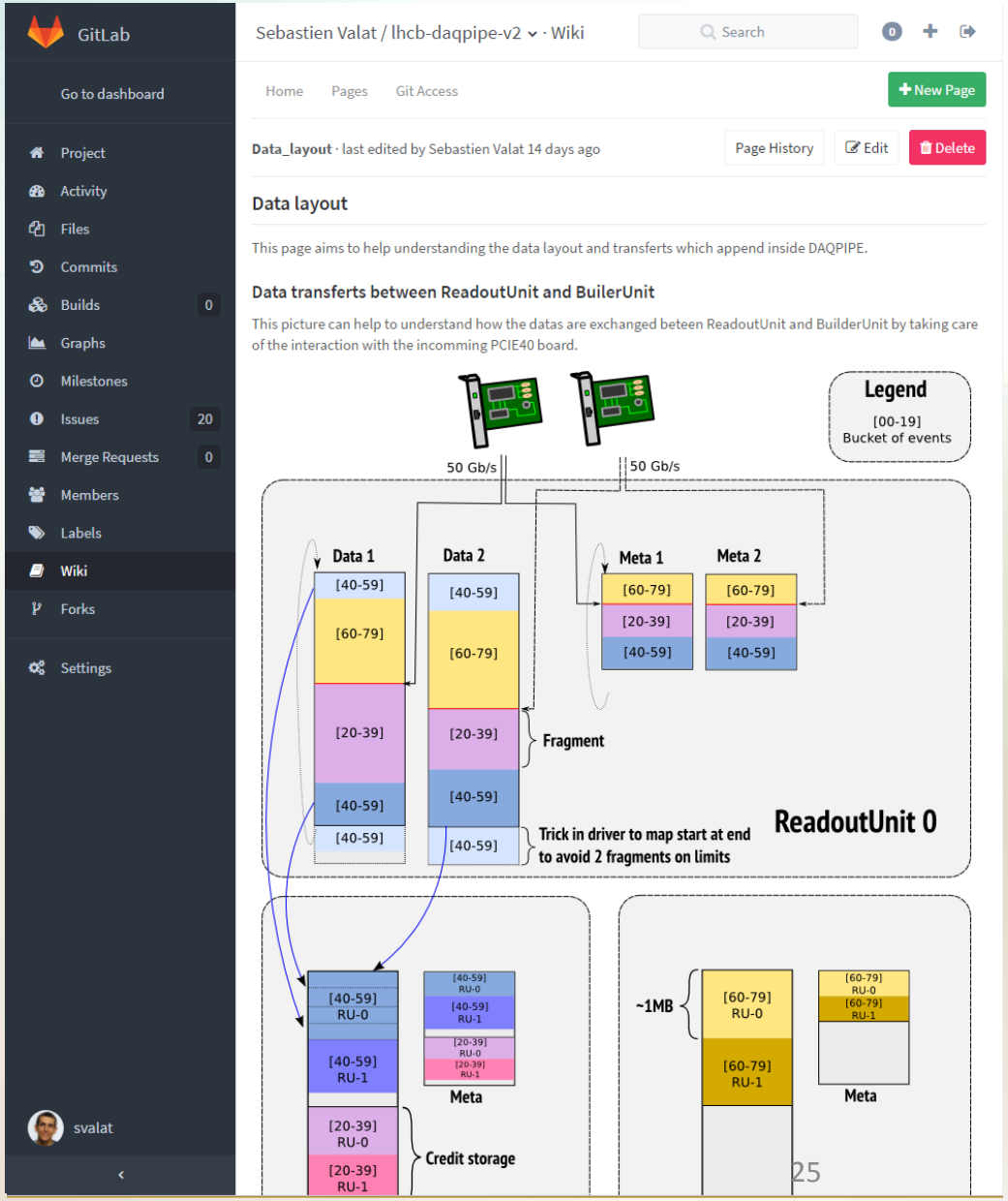
Documentation

- **Doxygen** : **yes** but it is **not enough**
- I like to also have a **wiki** with a **global view** on **how the code is working**
  - On **release** I make a **snapshot** of the wiki **in the source code**
- Provide some usage **examples** (unit tests) ?
- The **knowledge** must **not be** only in your **code**



Documentation

- **For who** ? Me ? You ?
- I use documentation for **me**.
- If you **cannot explain** what you have done…. **rework it**
- It let you the **time to think** what you are doing
- It is **not up-to-date** ? Ok, still better than having just the code.



Final message

Think about your **development strategy, always improve** and **use the right tools**….

If you want an example search **lhcb-daqpipe-v2** in cern gitlab

Project management

- **Out of the scope** of this presentation
- But if you are **a team**, look on **agile methods** :
  - **Scrum**
- Using **agile methods** requires all the **previous stuff** !
- Agile method **is not having no method**….