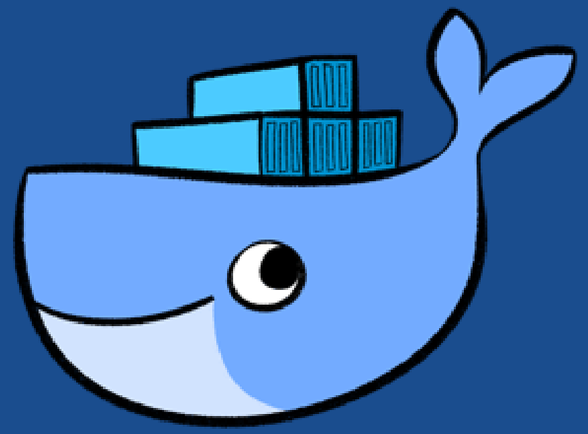
HSSIP 2021

QA: Static Code Analysis

CERN BE-ICS-FT, Philipp Burkhardt and Nik Lehmann



What the ICS group does

Run and maintain the control system for CERN experiments

- Some of the world's largest and most complex infrastructures
- ICS-CS provides power to them and many other "fundamental services"
- Lots of mission critical software, historical and heterogenous, some of it in C++
- Need to do Quality Assurance (QA) on a large scale on this C++ code base

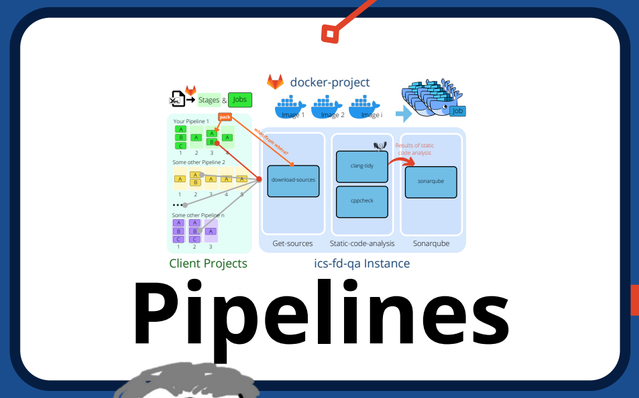
Over

1'000'000

Lines of code

Code powers CERN

Pipelines are the magic behind it



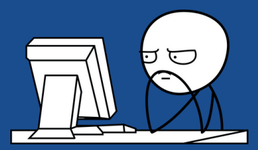
pipelines go brrr

What did we do?

QA pipelines: virtualisation & clang-tidy

Automate Quality Assurance in a pipeline, using advanced tools for static code analysis

- Understand & document what is going on
- Contribute to the QA pipeline

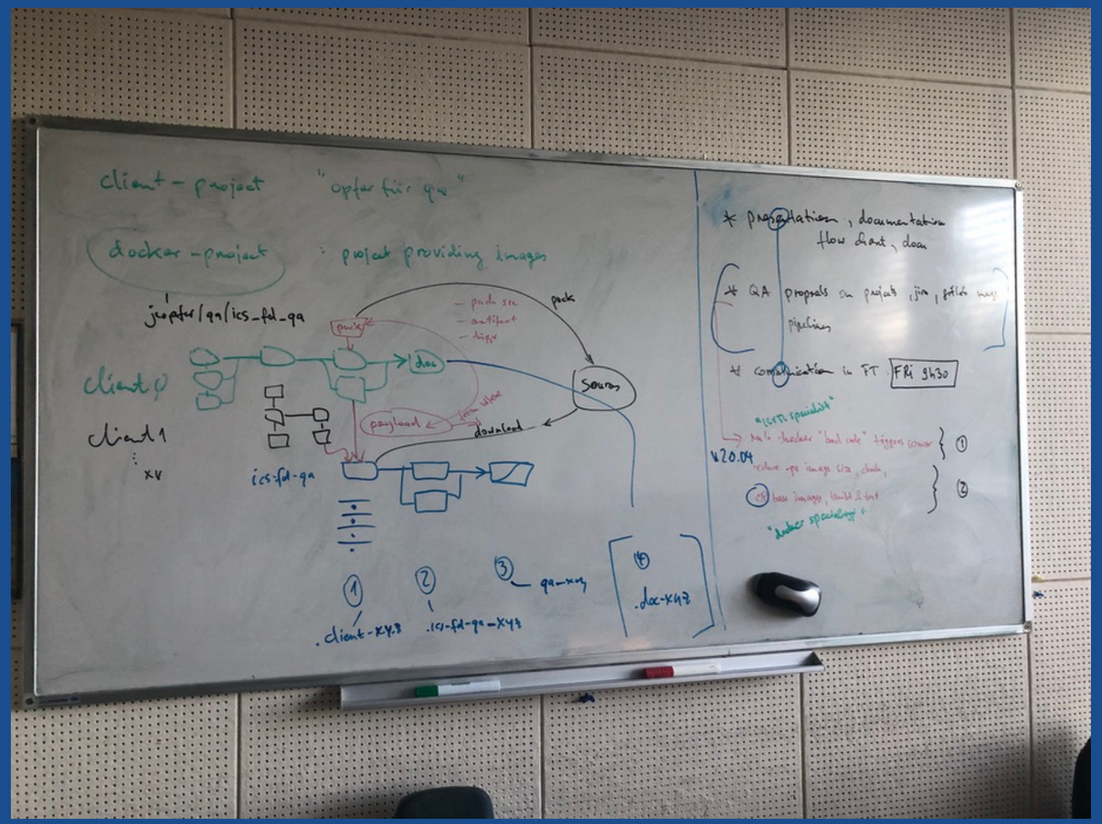


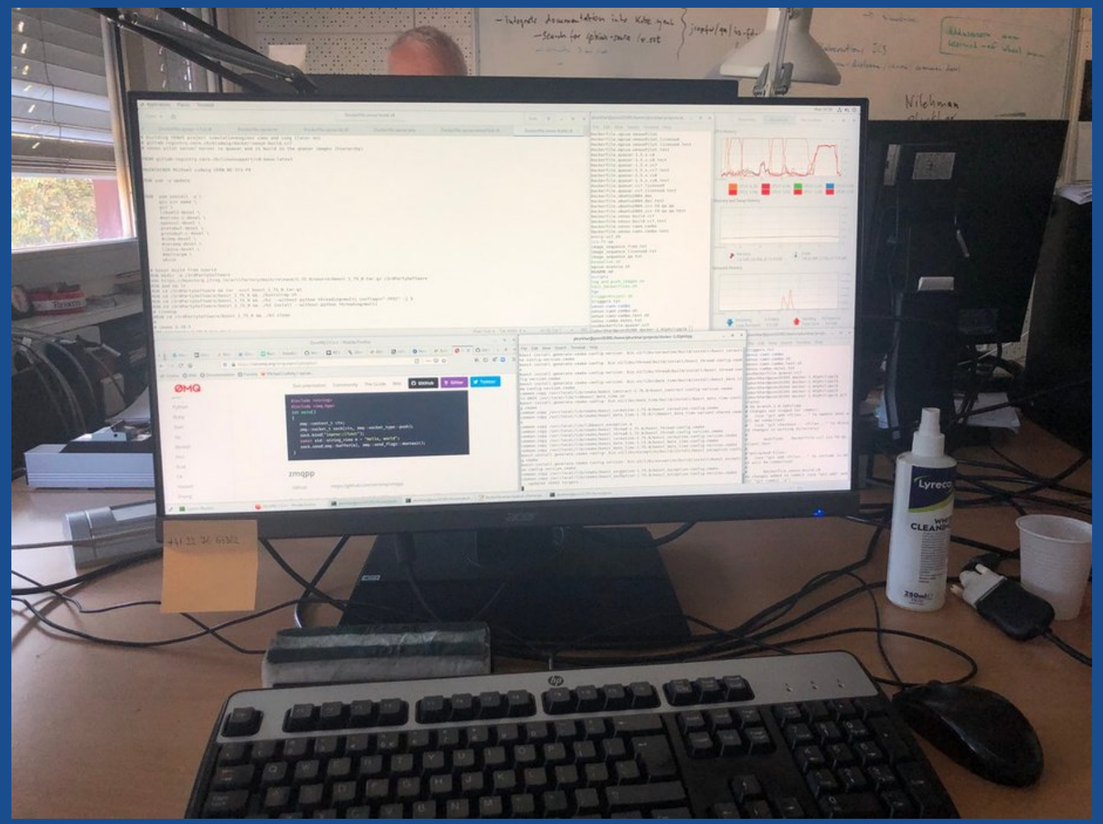
  - ***Nik:*** *Static code analysis project*
    - LLVM clang-tidy 8 ➜ 10

  - ***Philipp:*** *Virtualisation project*
    - *Docker images cc7 ➜ c8*

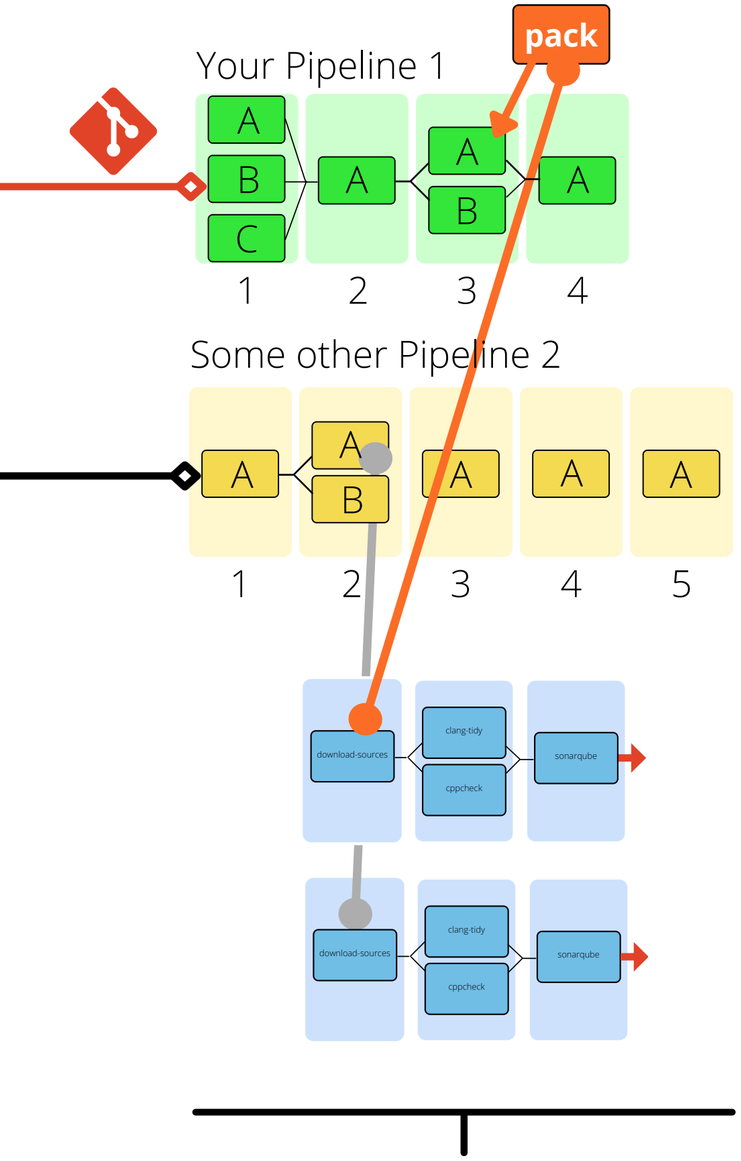
Work as a "Software Engineer"

From big picture to details and fueled by coffee



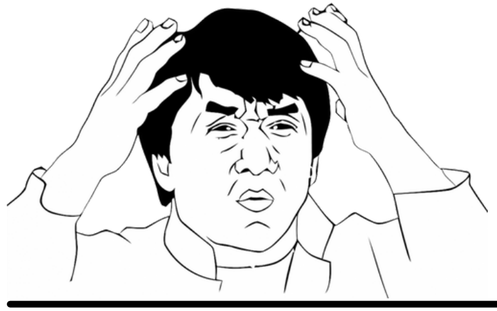


Pipelines



Abstraction

ics-fd-qa?



Various Client Projects

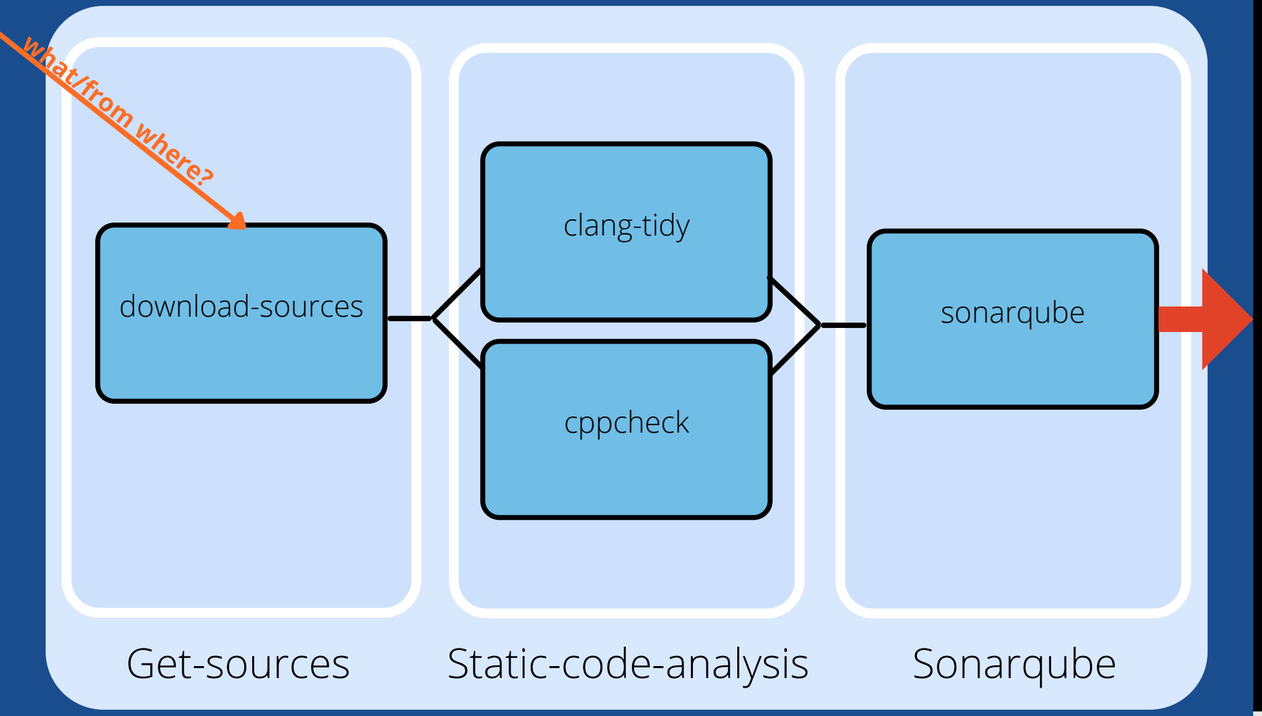
QA Pipeline Instances

Example

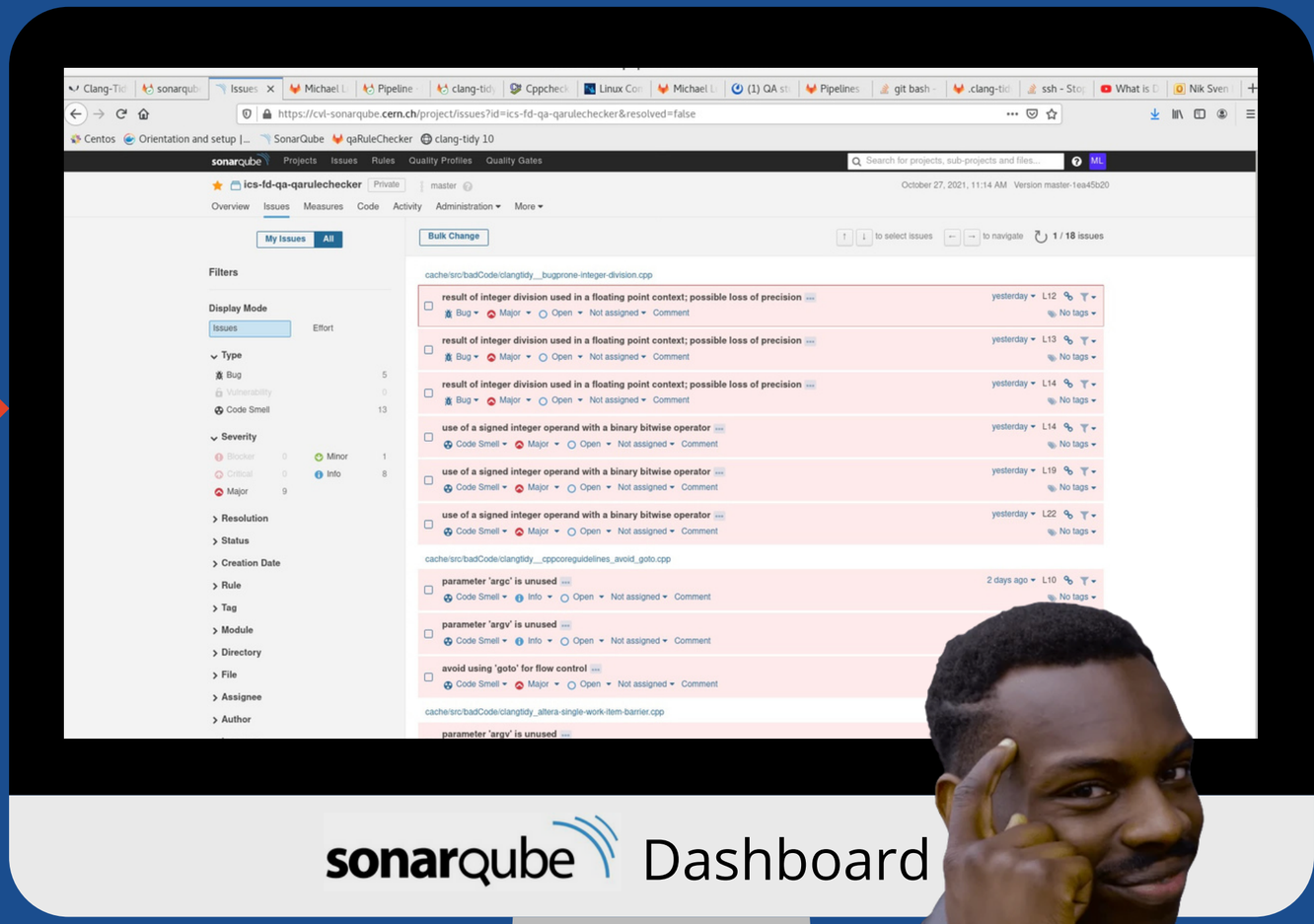
Explanation:

Stages & Jobs

Quality Assurance Pipeline



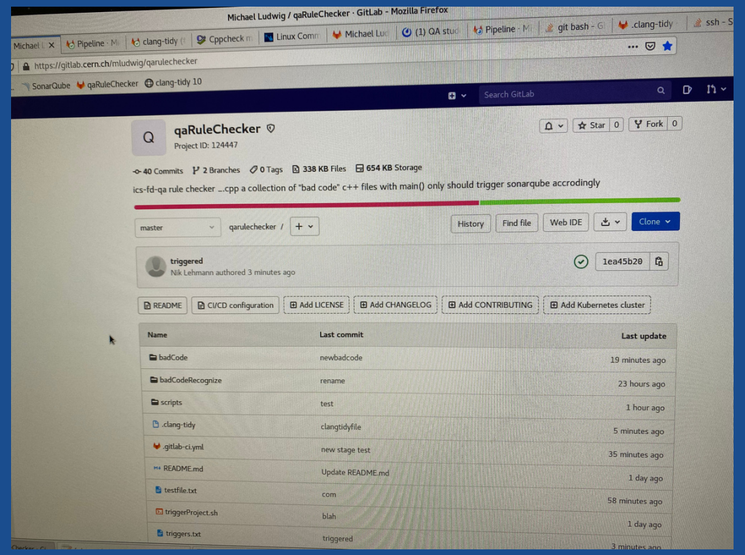
ics-fd-qa Pipeline Instance

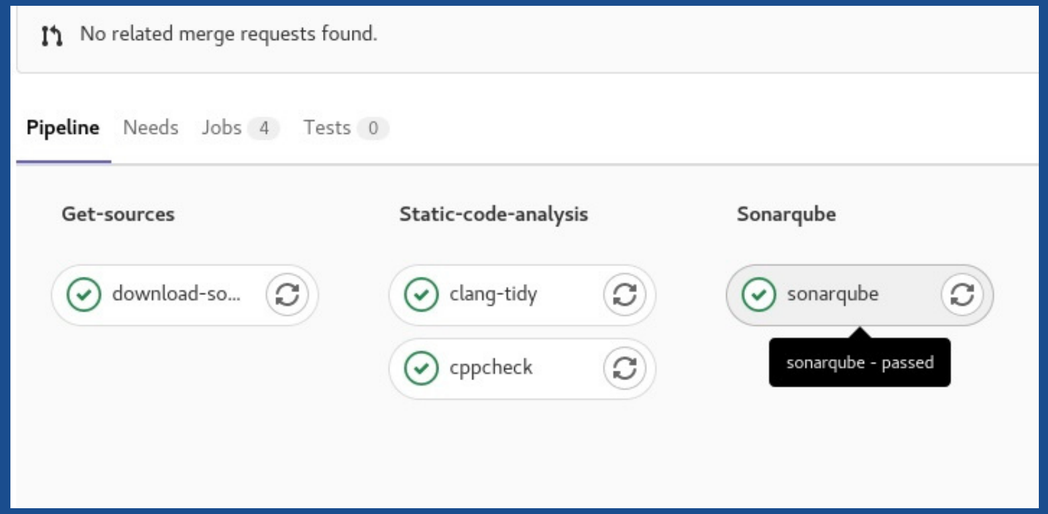


sonarqube Dashboard

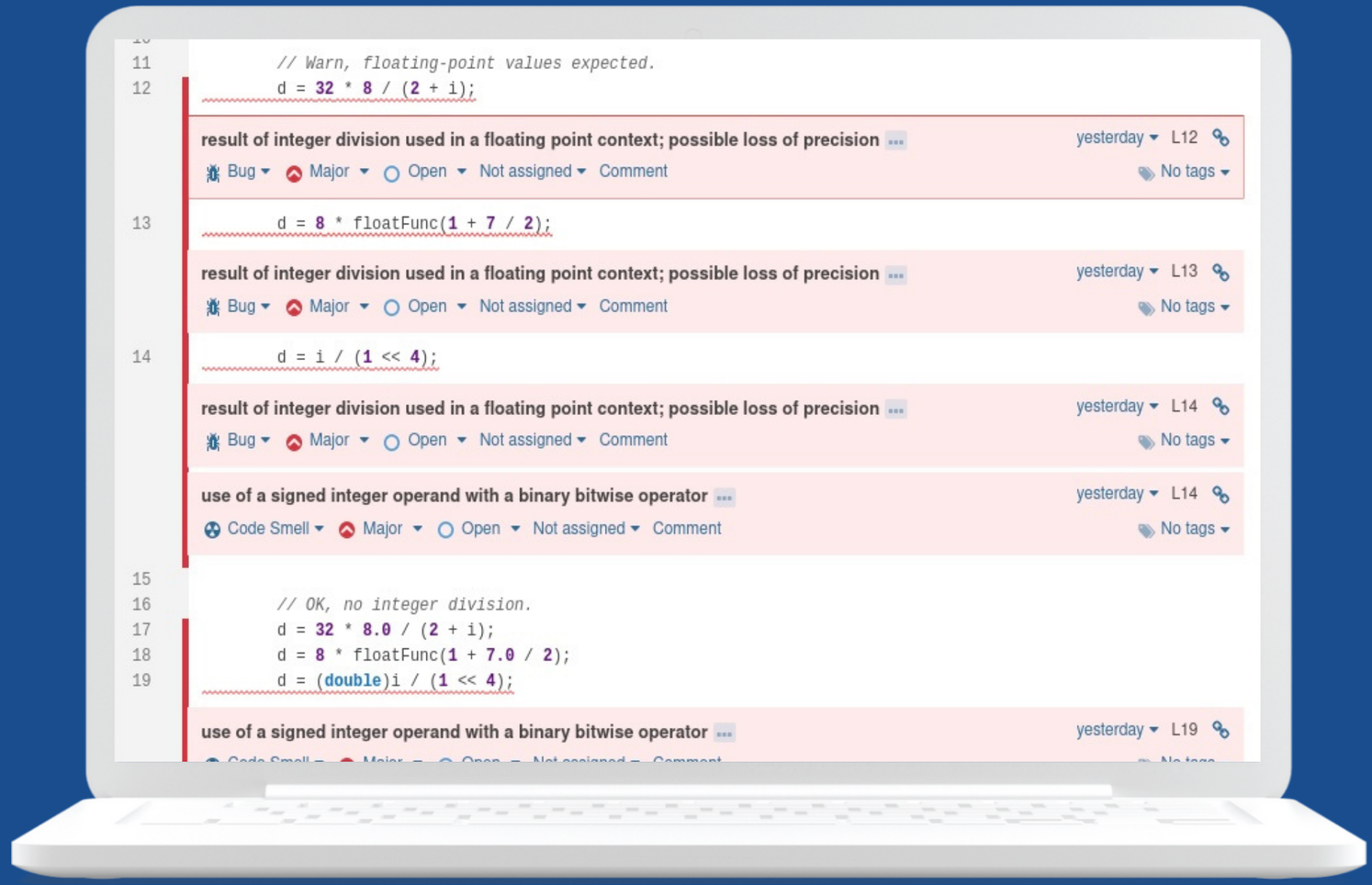
LLVM project – clang-tidy

The purpose of clang-tidy is to provide an extensible framework for diagnosing and fixing typical programming errors, like style violations, interface misuse, or bugs that can be deduced via static analysis.





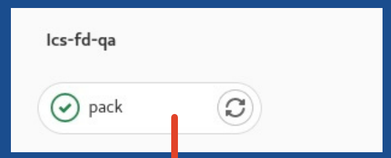
- Update clang-tidy image from llvm.v8 to llvm.v10
- Check rules
- Optimize and configure code checker, update configs
- Test llvm.v10 image inside container



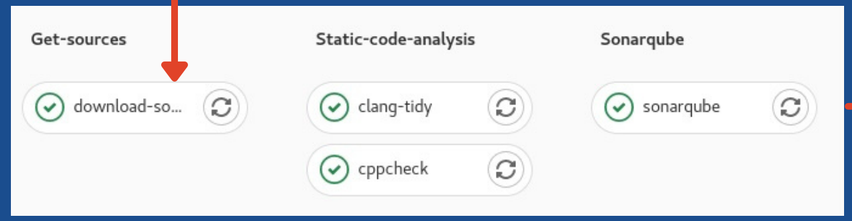
Example for automatically found and reported bugs from injected bad code

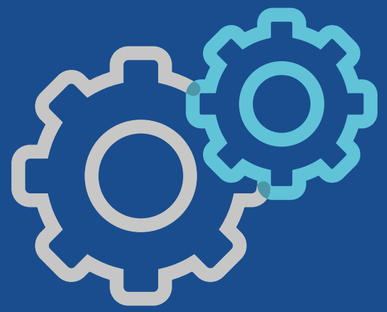
How clang-tidy is used

client pipeline: "qaRuleChecker"



"ics-fd-qa" pipeline





ics-fd-qa-qarulechecker
cache/src/badCode/clangtidy__cppcoreguidelines_avoid_goto.cpp

24 Lines | 3 Issues | 0.0% Coverage

```
1  /**
2   https://releases.llvm.org/12.0.0/tools/clang/tools/extra/docs/clang-tidy/checks/cppcoreguidelines-avoid-goto.html
3
4   The usage of goto for control flow is error prone and should be replaced with looping constructs. Only forward jumps in nested
5   This check implements ES.76 from the CppCoreGuidelines and 6.3.1 from High Integrity C++.
6   For more information on why to avoid programming with goto you can read the famous paper A Case against the GO TO Statement..
7   The check diagnoses goto for backward jumps in every language mode. These should be replaced with C/C++ looping constructs.
8  */
9  using std;
10 int main( int argc, char** argv ){
```

parameter 'argc' is unused — 3 days ago · L10
Code Smell · Info · Open · Not assigned · Comment · No tags

parameter 'argv' is unused — 3 days ago · L10
Code Smell · Info · Open · Not assigned · Comment · No tags

```
11
12        // Bad, handwritten for loop.
13        int i = 0;
14        // Jump label for the loop
15        loop_start:
16                std::cout << "do dummy" << std::endl;
17
18        if (i < 100) {
19          ++i;
20          goto loop_start; // jump backwards
```

avoid using 'goto' for flow control — 3 days ago · L20
Code Smell · Major · Open · Not assigned · Comment · No tags

```
21        }
22        return 0;
23 }
24
```

ics-fd-qa-qarulechecker
cache/src/badCode/clangtidy__cppcoreguidelines_avoid_goto.cpp

19 Lines | 0 Issues | 0.0% Coverage

```cpp
/**
 https://releases.llvm.org/12.0.0/tools/clang/tools/extra/docs/clang-tidy/checks/cppcoreguidelines-avoid-goto.html

 The usage of goto for control flow is error prone and should be replaced with looping constructs. Only forward jumps in nested loops are accepted.
 This check implements ES.76 from the CppCoreGuidelines and 6.3.1 from High Integrity C++.
 For more information on why to avoid programming with goto you can read the famous paper A Case against the GO TO Statement..
 The check diagnoses goto for backward jumps in every language mode. These should be replaced with C/C++ looping constructs.
*/
using std;
int main(){
        int i = 0;

        if (i < 100) {
          ++i;
          std::cout << "do dummy" << std::endl; //execute here
        }
        return 0;
}
```

ics-fd-qa-qarulechecker
cache/src/badCode/clangtidy__cppcoreguidelines_avoid_goto.cpp

24 Lines | 3 Issues | 0.0% Coverage

```cpp
/**
 https://releases.llvm.org/12.0.0/tools/clang/tools/extra/docs/clang-tidy/checks/cppcoreguidelines-avoid-goto.html

 The usage of goto for control flow is error prone and should be replaced with looping constructs. Only forward jumps in nested
 This check implements ES.76 from the CppCoreGuidelines and 6.3.1 from High Integrity C++.
 For more information on why to avoid programming with goto you can read the famous paper A Case against the GO TO Statement..
 The check diagnoses goto for backward jumps in every language mode. These should be replaced with C/C++ looping constructs.
*/
using std;
int main( int argc, char** argv ){
```

parameter 'argc' is unused — 3 days ago · L10
Code Smell · Info · Open · Not assigned · Comment · No tags

parameter 'argv' is unused — 3 days ago · L10
Code Smell · Info · Open · Not assigned · Comment · No tags

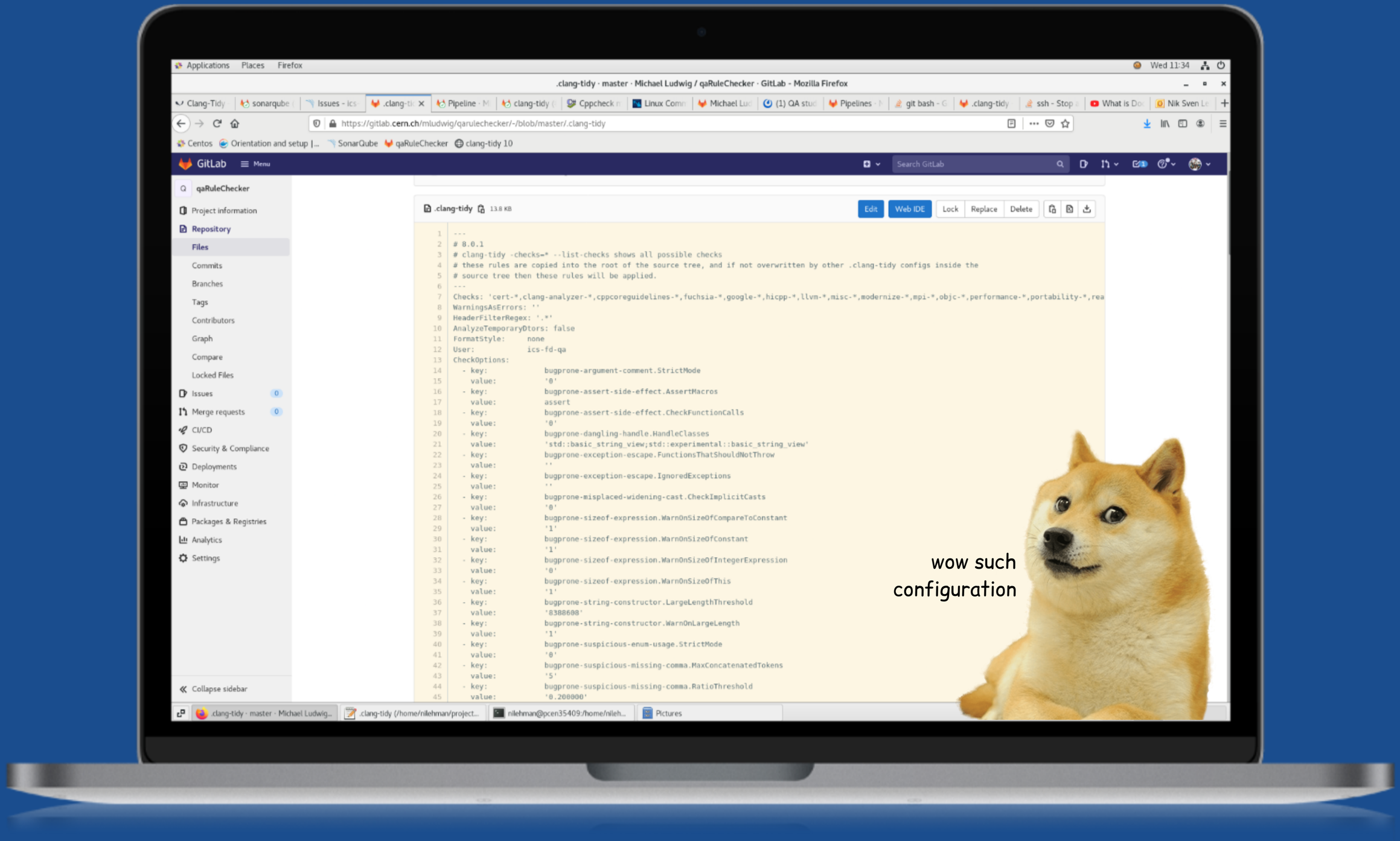
```cpp

        // Bad, handwritten for loop.
        int i = 0;
        // Jump label for the loop
        loop_start:
                std::cout << "do dummy" << std::endl;

        if (i < 100) {
          ++i;
          goto loop_start; // jump backwards
```

avoid using 'goto' for flow control — 3 days ago · L20
Code Smell · Major · Open · Not assigned · Comment · No tags

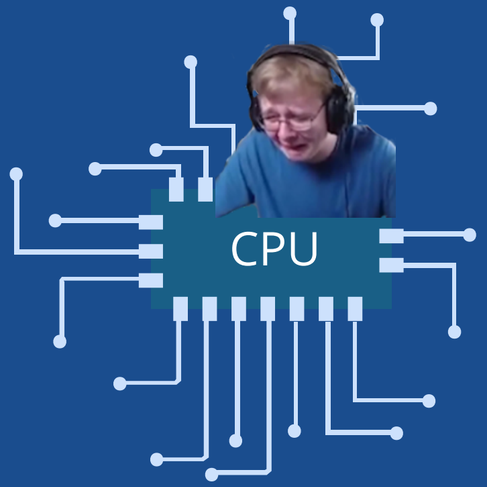
```cpp
        }
        return 0;
}
```


Applications Places Firefox
Wed 11:34
.clang-tidy · master · Michael Ludwig / qaRuleChecker · GitLab - Mozilla Firefox
https://gitlab.cern.ch/mludwig/qarulechecker/-/blob/master/.clang-tidy
GitLab
Menu
Search GitLab
qaRuleChecker
Project information
Repository
Files
Commits
Branches
Tags
Contributors
Graph
Compare
Locked Files
Issues
Merge requests
CI/CD
Security & Compliance
Deployments
Monitor
Infrastructure
Packages & Registries
Analytics
Settings
Collapse sidebar
.clang-tidy 13.8 KB
Edit
Web IDE
Lock
Replace
Delete

8.0.1
clang-tidy -checks=* --list-checks shows all possible checks
these rules are copied into the root of the source tree, and if not overwritten by other .clang-tidy configs inside the
source tree then these rules will be applied.

Checks: 'cert-*,clang-analyzer-*,cppcoreguidelines-*,fuchsia-*,google-*,hicpp-*,llvm-*,misc-*,modernize-*,mpi-*,objc-*,performance-*,portability-*,rea
WarningsAsErrors: ''
HeaderFilterRegex: '.*'
AnalyzeTemporaryDtors: false
FormatStyle: none
User: ics-fd-qa
CheckOptions:
- key: bugprone-argument-comment.StrictMode
value: '0'
- key: bugprone-assert-side-effect.AssertMacros
value: assert
- key: bugprone-assert-side-effect.CheckFunctionCalls
value: '0'
- key: bugprone-dangling-handle.HandleClasses
value: 'std::basic_string_view;std::experimental::basic_string_view'
- key: bugprone-exception-escape.FunctionsThatShouldNotThrow
value: ''
- key: bugprone-exception-escape.IgnoredExceptions
value: ''
- key: bugprone-misplaced-widening-cast.CheckImplicitCasts
value: '0'
- key: bugprone-sizeof-expression.WarnOnSizeOfCompareToConstant
value: '1'
- key: bugprone-sizeof-expression.WarnOnSizeOfConstant
value: '1'
- key: bugprone-sizeof-expression.WarnOnSizeOfIntegerExpression
value: '0'
- key: bugprone-sizeof-expression.WarnOnSizeOfThis
value: '1'
- key: bugprone-string-constructor.LargeLengthThreshold
value: '8388608'
- key: bugprone-string-constructor.WarnOnLargeLength
value: '1'
- key: bugprone-suspicious-enum-usage.StrictMode
value: '0'
- key: bugprone-suspicious-missing-comma.MaxConcatenatedTokens
value: '5'
- key: bugprone-suspicious-missing-comma.RatioThreshold
value: '0.200000'
wow such
configuration
.clang-tidy · master · Michael Ludwig...
.clang-tidy (/home/nilehman/project...
nilehman@pcen35409:/home/nileh...
Pictures

Current situation

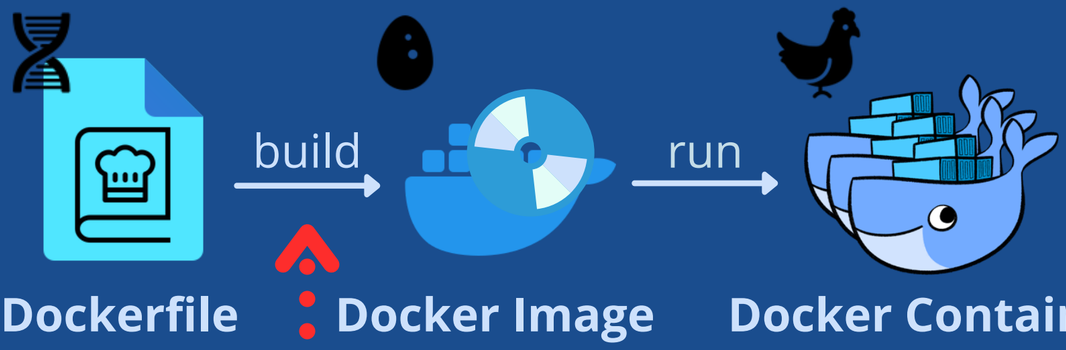


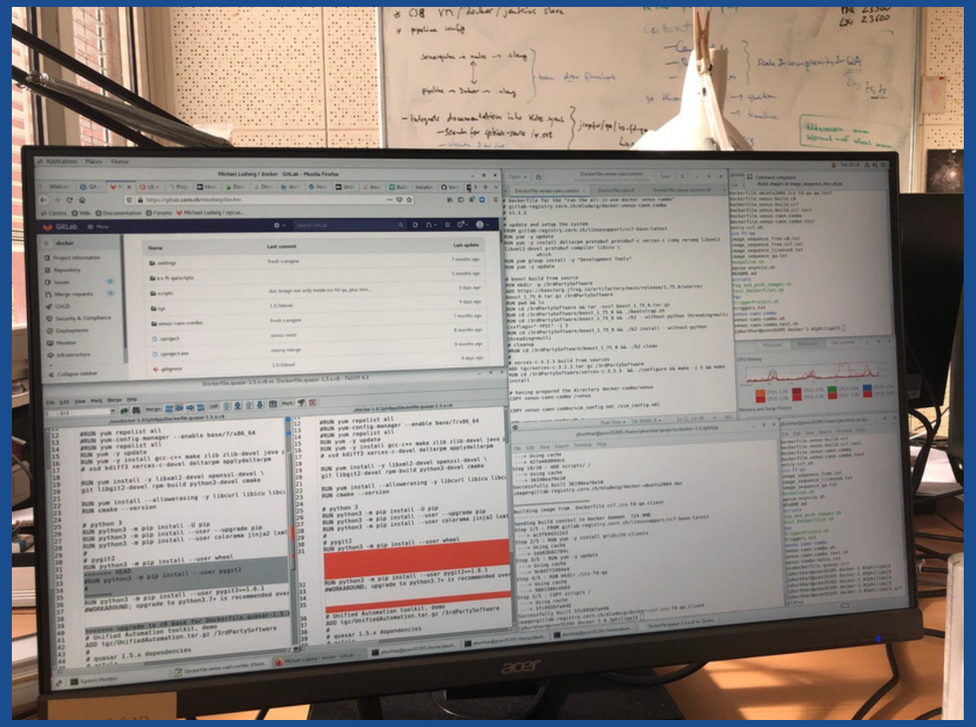
Not enough CPU power → Lack of time

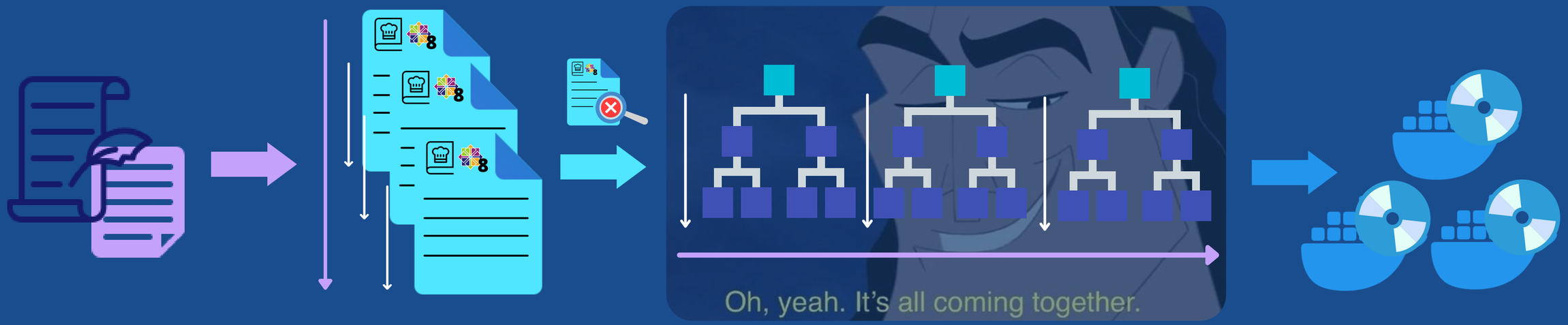
Next steps: Overview of differences between llvm.v8 to llvm.v10 & reasons for upgrade.

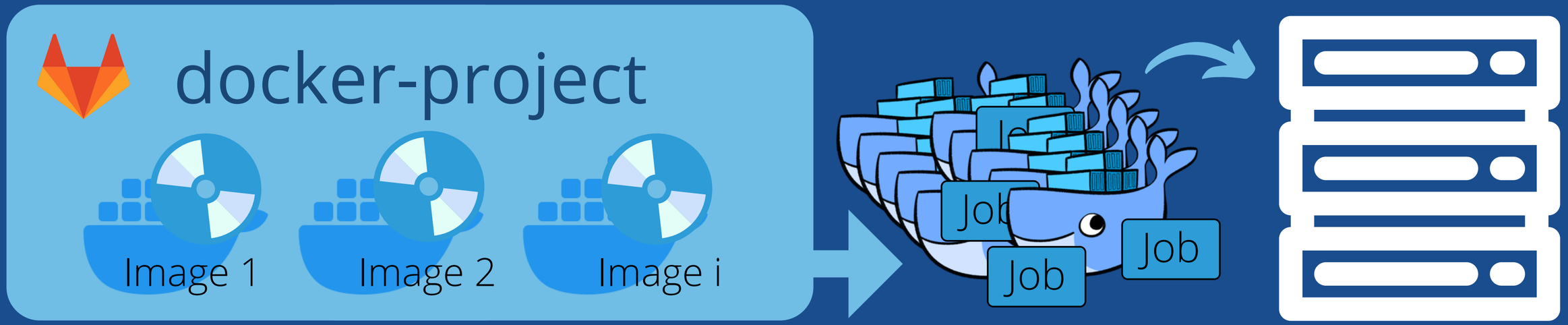
Virtualisation project using Docker

Migrating from CentOS 7 to CentOS 8






Oh, yeah. It's all coming together.

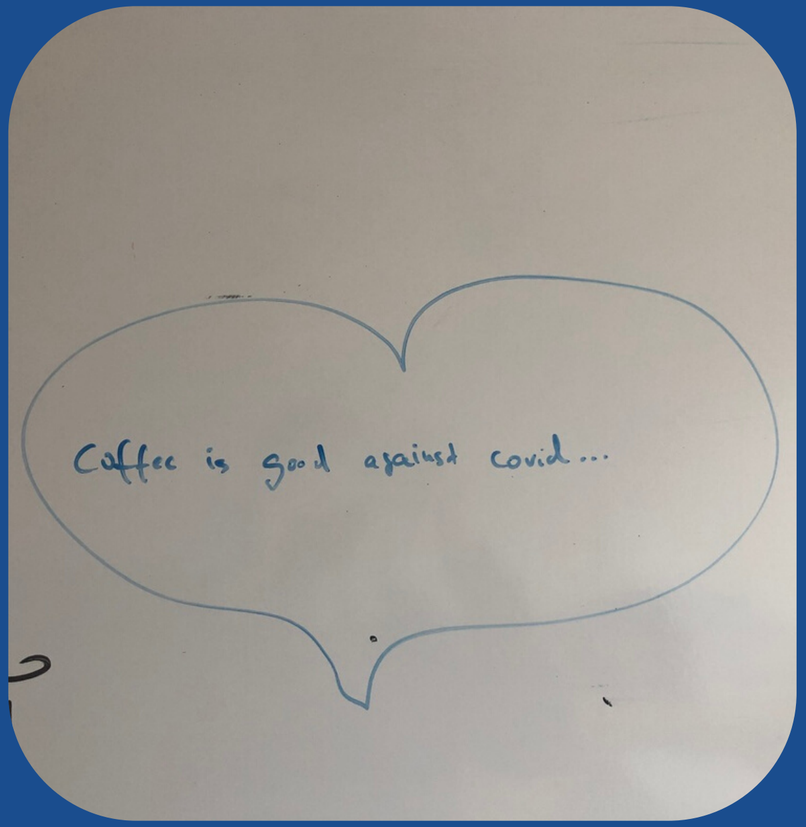

docker-project
Image 1
Image 2
Image i
Job
Job
Job

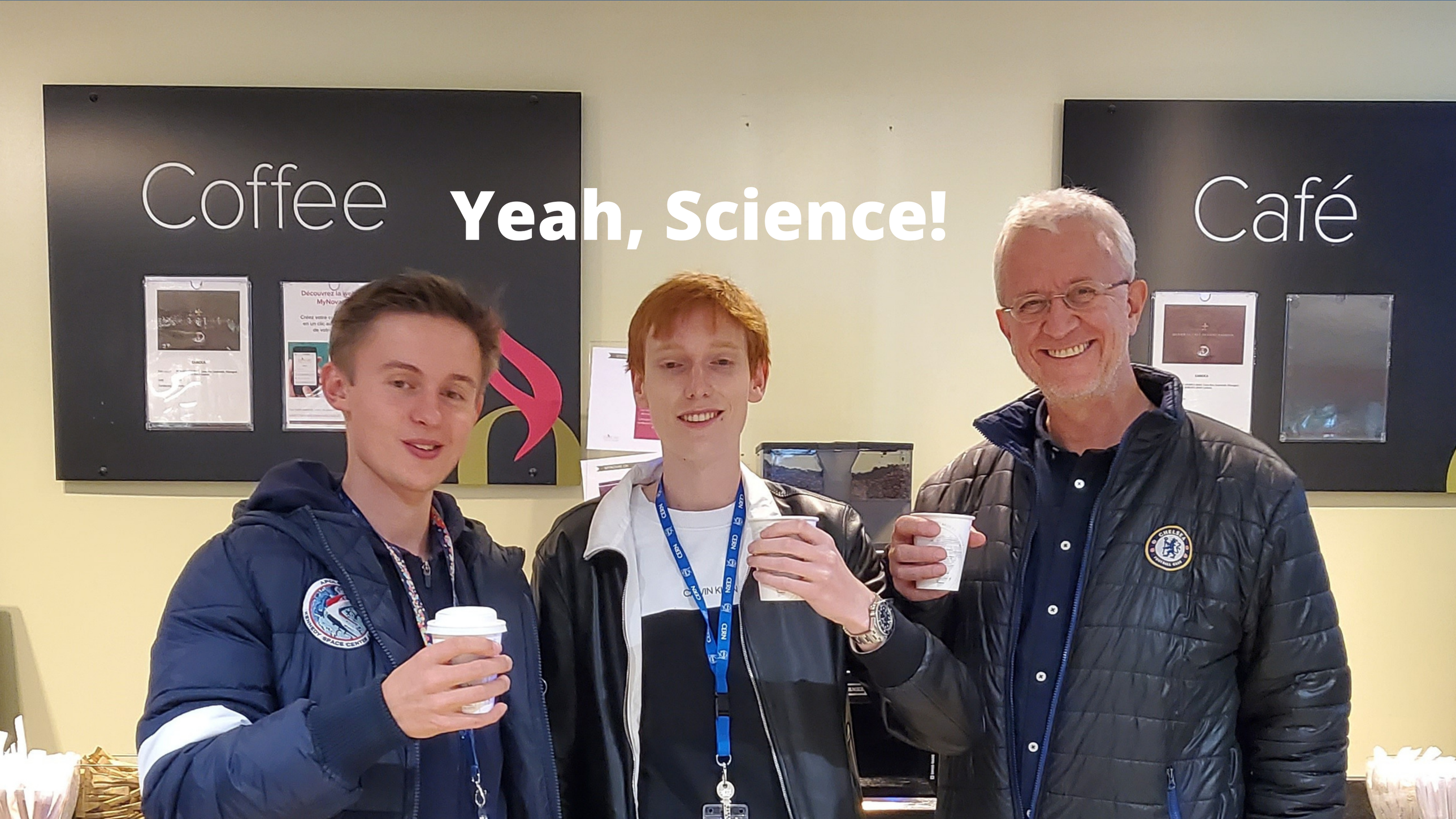
Results

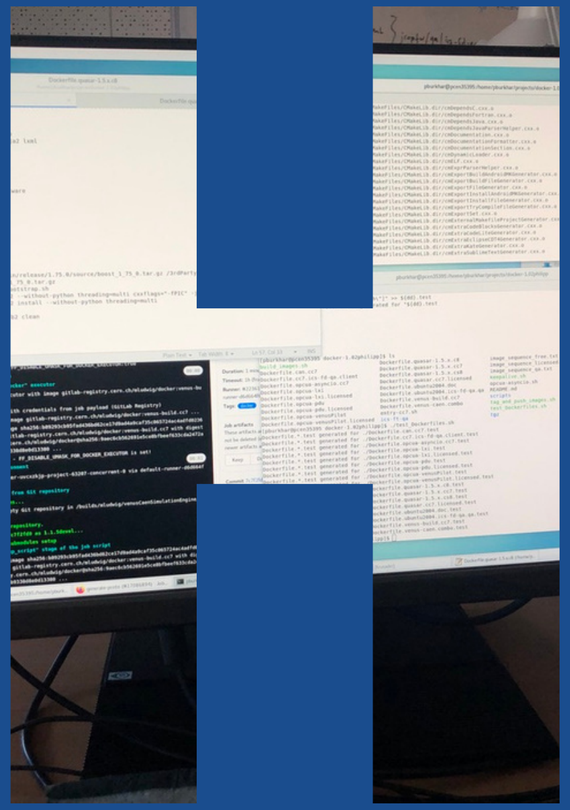
Lessons learned

What we take with us from HSSIP

- Coffee is love, coffee is life
- Software projects are complex, require focus and lots of brain power
- Virtualisation as part of CI/CD is very powerful gitlab, docker, containers
- Linux, Git CLI, Gitlab, Docker and more crazy tools
- Automation becomes absolutely necessary above a certain scale of things
- Communication and keeping the overview is vital

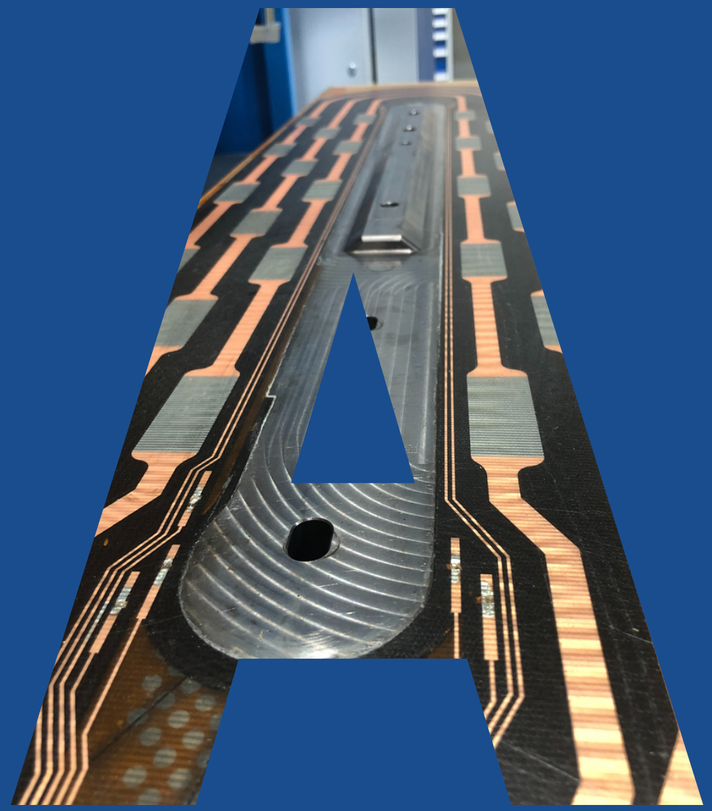


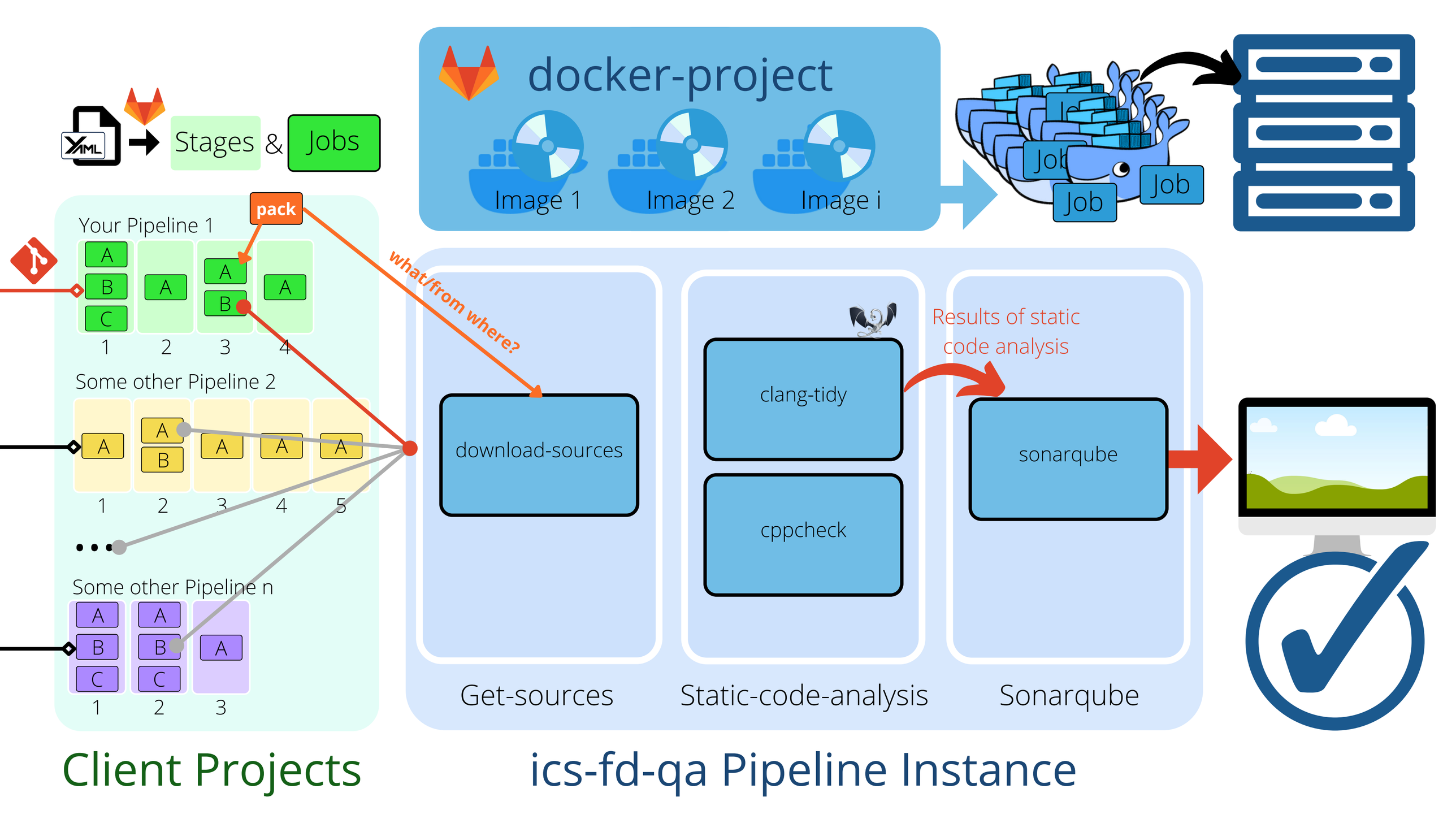

Coffee
Café
Yeah, Science!


for your attention

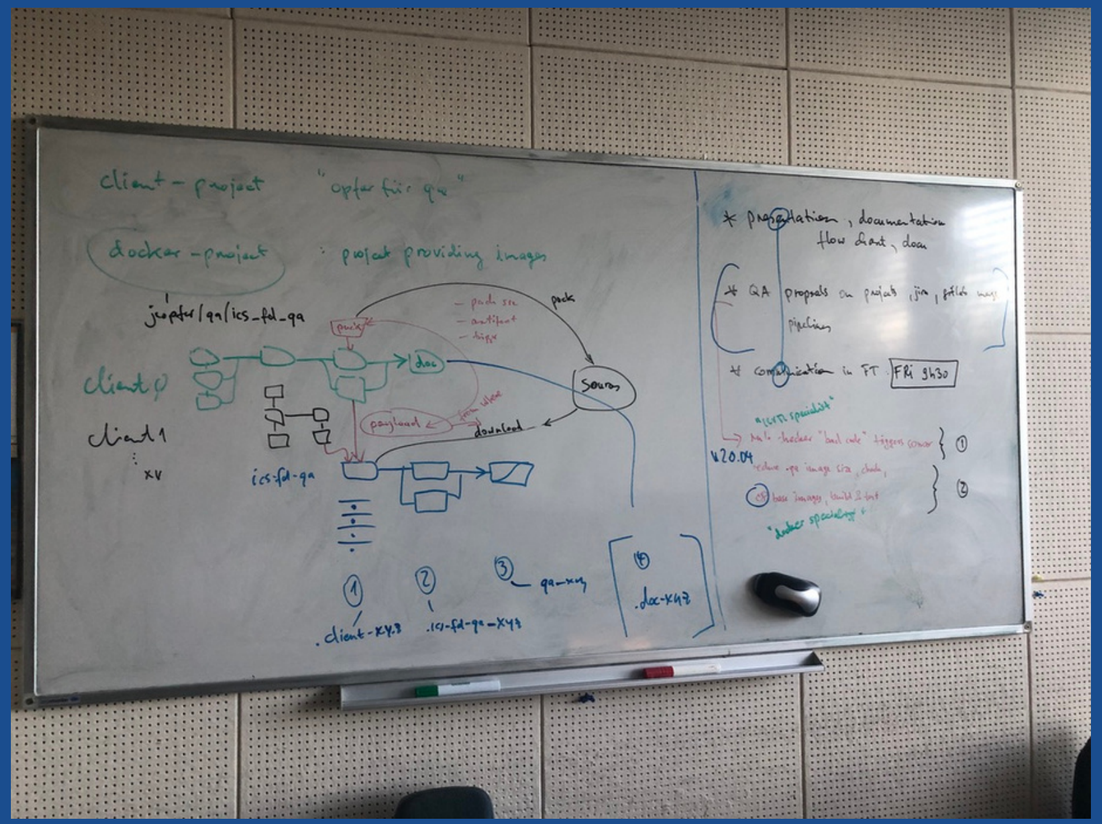

ANTIMATTER
FACTORY
ELENA

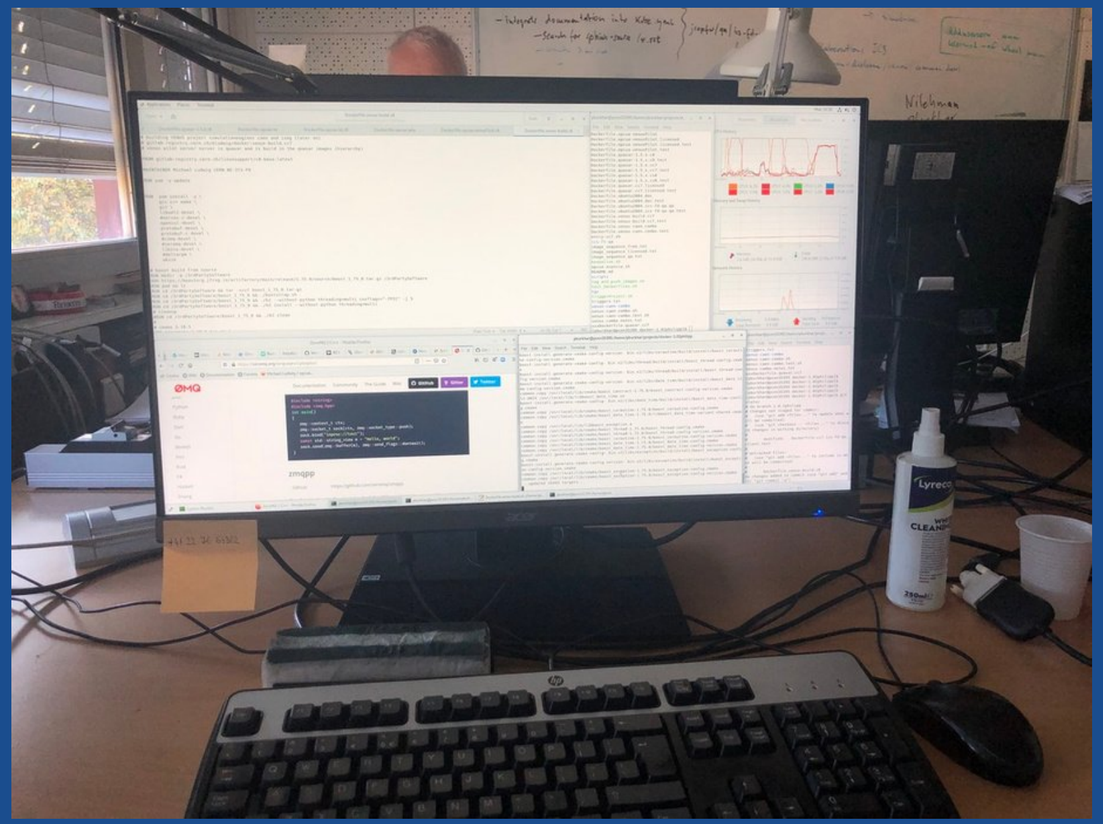



Stages & Jobs
docker-project
Image 1
Image 2
Image i
Job
Job
Job
Your Pipeline 1
pack
A
B
C
A
A
A
B
A
1
2
3
4
Some other Pipeline 2
A
A
B
A
A
A
1
2
3
4
5
Some other Pipeline n
A
B
C
A
B
C
A
1
2
3
what/from where?
download-sources
clang-tidy
cppcheck
Results of static code analysis
sonarqube
Get-sources
Static-code-analysis
Sonarqube
Client Projects
ics-fd-qa Pipeline Instance

Work as a "Software Engineer"

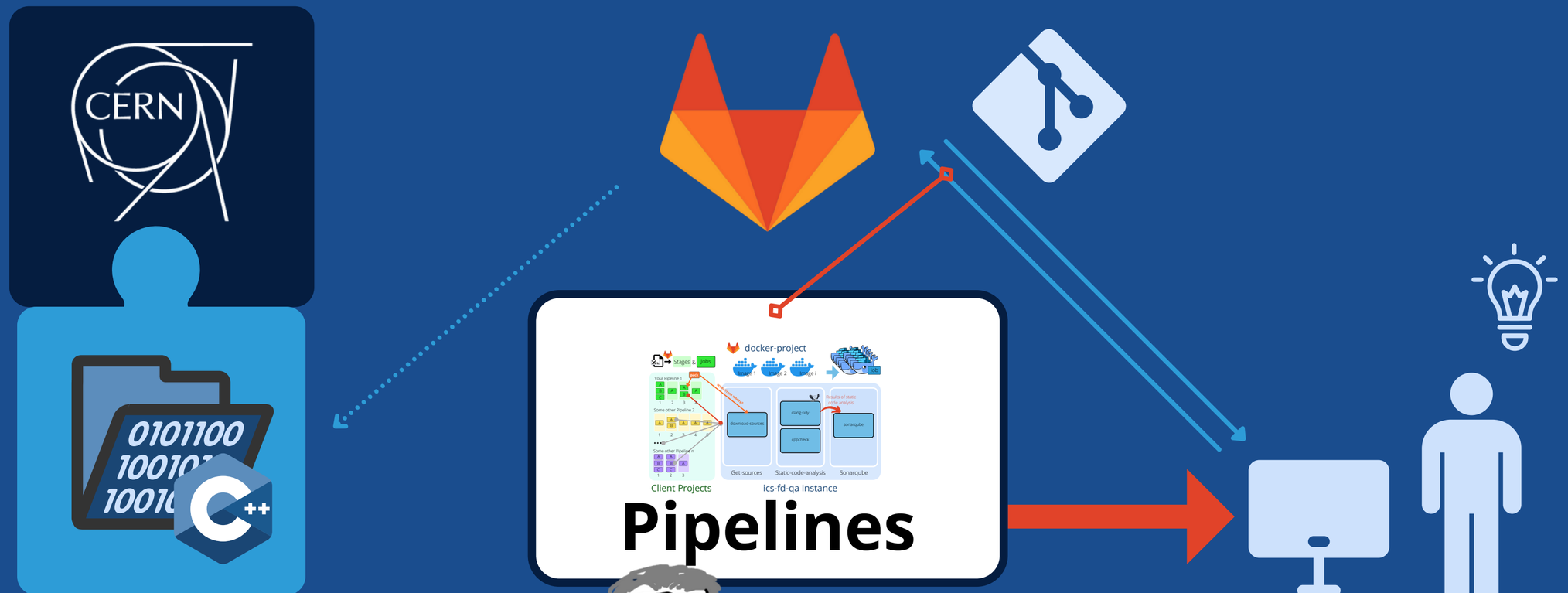
From big picture to details and fueled by coffee





Code powers CERN

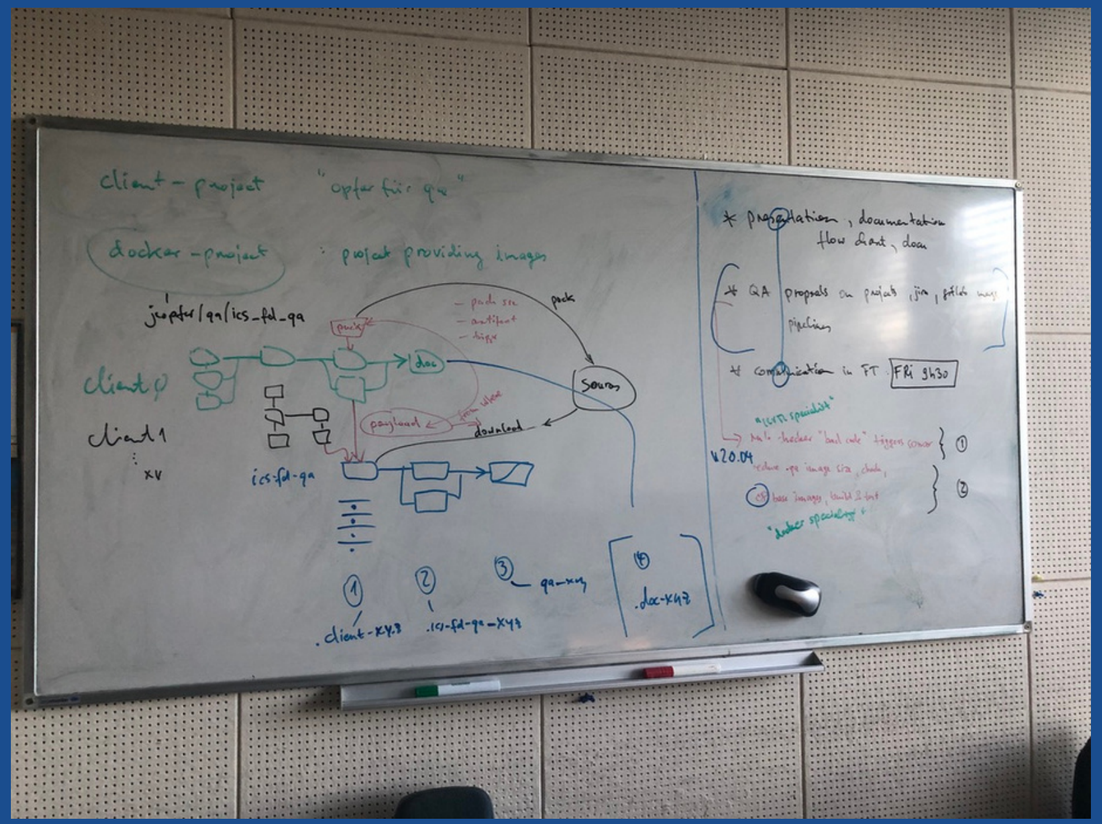
Pipelines are the magic behind it

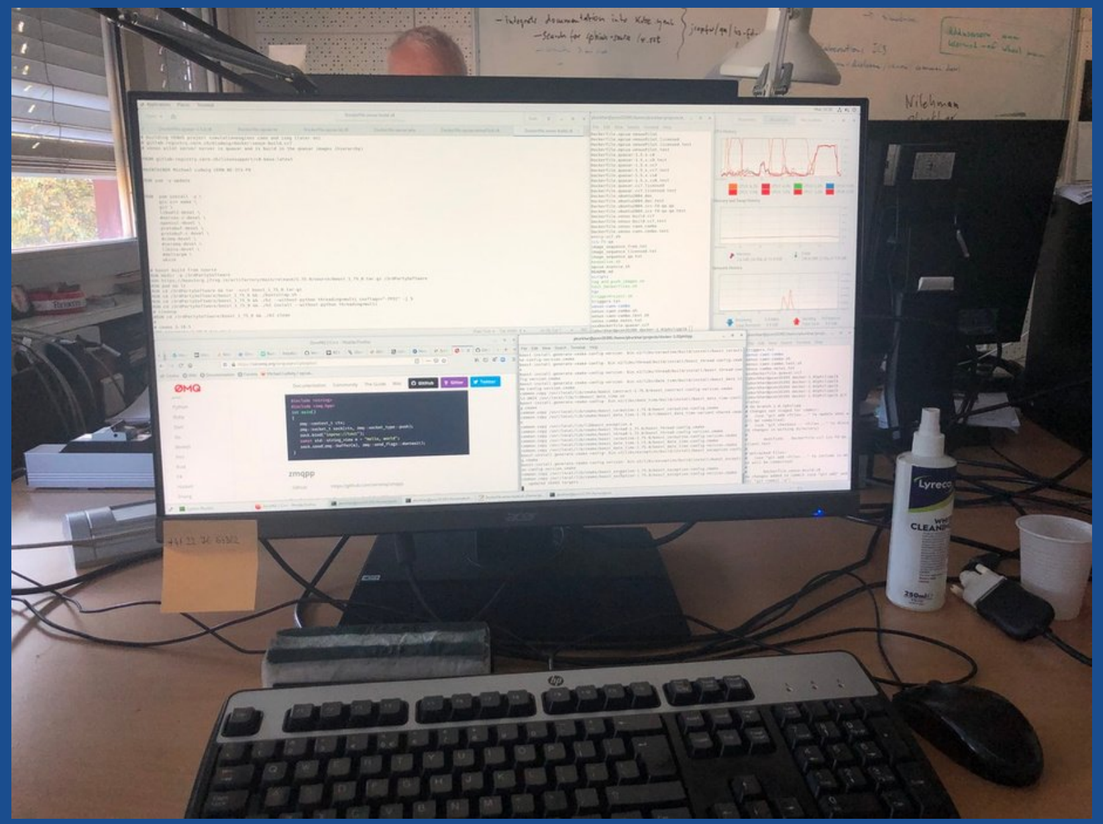


pipelines go brrr

Big picture to details

Working as a "Software Engineer"





cplusplus0.cpp [Read-Only]

```cpp
// clang.llvm.org/docs/analyzer/checkers.html#cplusplus-newdeleteleaks-c


// 1.1.2.1. cplusplus.InnerPointer (C++)
void deref_after_assignment() {
  std::string s = "llvm";
  const char *c = s.data(); // note: pointer to inner buffer of 'std::string' obtained here
  s = "clang"; // note: inner buffer of 'std::string' reallocated by call to 'operator='
  consume(c); // warn: inner pointer of container used after re/deallocation
}

const char *return_temp(int x) {
  return std::to_string(x).c_str(); // warn: inner pointer of container used after re/deallocation
  // note: pointer to inner buffer of 'std::string' obtained here
  // note: inner buffer of 'std::string' deallocated by call to destructor
}


// 1.1.2.3. cplusplus.NewDeleteLeaks (C++)
void test() {
  int *p = new int;
} // warn


// 1.1.2.4. cplusplus.PlacementNewChecker (C++)
#include <new>

void f() {
  short s;
  long *lp = ::new (&s) long; // warn
}


// 1.1.2.2. cplusplus.NewDelete (C++)
void f(int *p);

void testUseMiddleArgAfterDelete(int *p) {
  delete p;
  f(p); // warn: use after free
}

class SomeClass {
public:
  void f();
};

void test() {
  SomeClass *c = new SomeClass;
  delete c;
  c->f(); // warn: use after free
}

void test() {
  int *p = (int *)__builtin_alloca(sizeof(int));
```