Automate your life with Gitlab-CI

Szymon Tomasz Datko

Summer Student

at

Computer Security Team

9th of August, 2016

Presentation in a nutshell

1) Motivation – why it may be useful?

2) Overview – how it works in general?

3) Usage – what do I need to do?

4) Summary – where to look for more?

Presentation in a nutshell

1) Motivation – why it may be useful?

2) Overview – how it works in general?

3) Usage – what do I need to do?

4) Summary – where to look for more?

1) Motivation

Stages of programming:

- building → compilation
- testing → unit tests
- testing → code analyzers
- deploying → packaging

1) Motivation

Stages of programming:

- building → compilation
- testing → unit tests
- testing → code analyzers
- deploying → packaging

1) Motivation

Stages of programming:

- building → compilation
- testing → unit tests
- testing → code analyzers
- deploying → packaging

Continuous Integration is about automating this and performing regularly

Presentation in a nutshell

1) Motivation – why it may be useful?

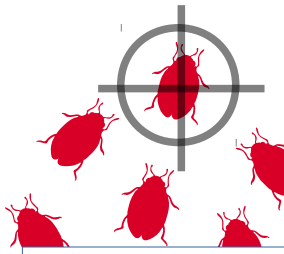
2) Overview – how it works in general?

3) Usage – what do I need to do?

4) Summary – where to look for more?

2) Overview – Code Analyzers

- **static** code analysis (without executing a program)
- sometimes integrated into advanced code editors
- simply **scans** source code for unsafe calls
- complex **structure** analysis of syntax tree
- a lot of tools available:
  - **R**ough **A**uditing **T**ool for **S**ecurity
  - *Lint (PyLint, JSLint, CppLint, ...)
  - **PMD** (Project Mess Detector?)
  - …



Icon made by Freepik from www.flaticon.com

Apple's `goto fail`

```
...
if ((err = SSLFreeBuffer(&ctx)) != 0)
    goto fail;
if ((err = Ready(&SSL, &ctx)) != 0)
    goto fail;
if ((err = SSL.update(&ctx, &crnd)) != 0)
    goto fail;
if ((err = SSL.update(&ctx, &srnd)) != 0)
    goto fail;
if ((err = SSL.update(&ctx, &prms)) != 0)
    goto fail;
    goto fail;
if ((err = SSL.final(&ctx, &out)) != 0)
    goto fail;
...
```

unreachable code

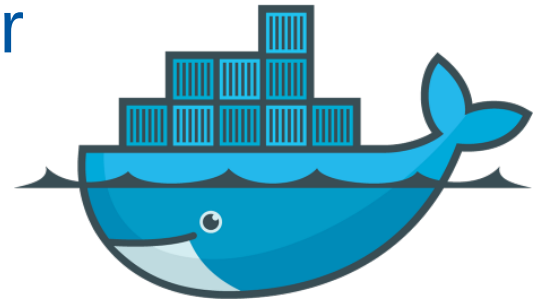
2) Overview – Gitlab-CI

- **part** of Gitlab (since version 8.0)

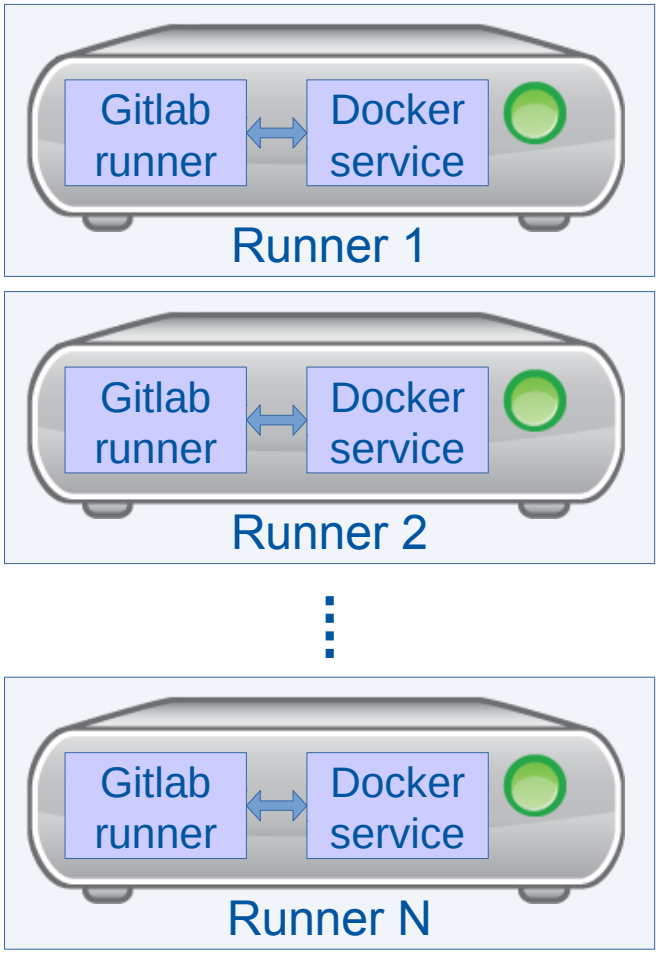
- allows one to introduce **Continuous Integration** easy way – automate tests, build and deploy

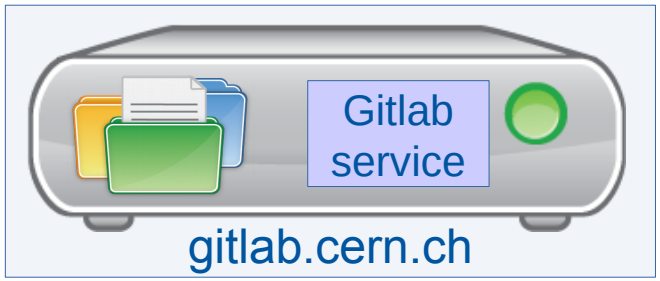
- **jobs** configured per repository – simply in a file

- multiple **executors** available – run your jobs:
  – directly on a local or remote server
  – inside a container
  – inside a virtual machine

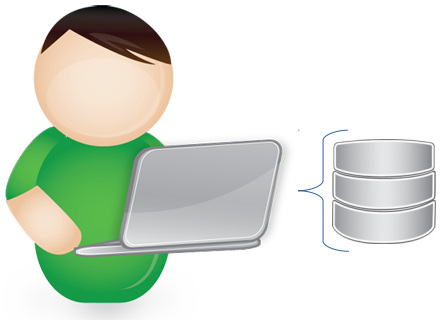


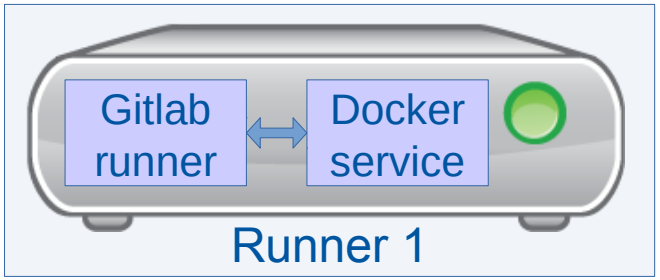
2) Overview - architecture

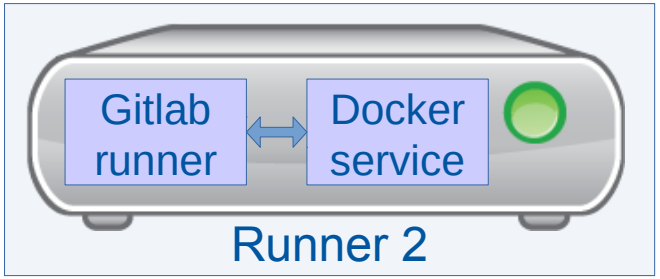


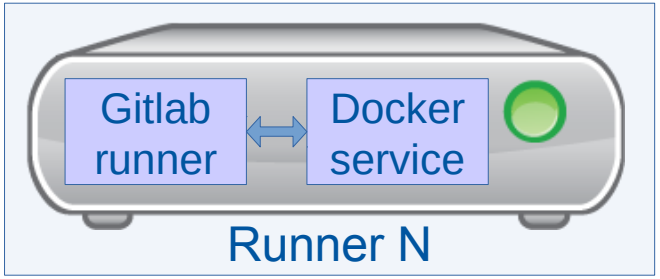


2) Overview - architecture

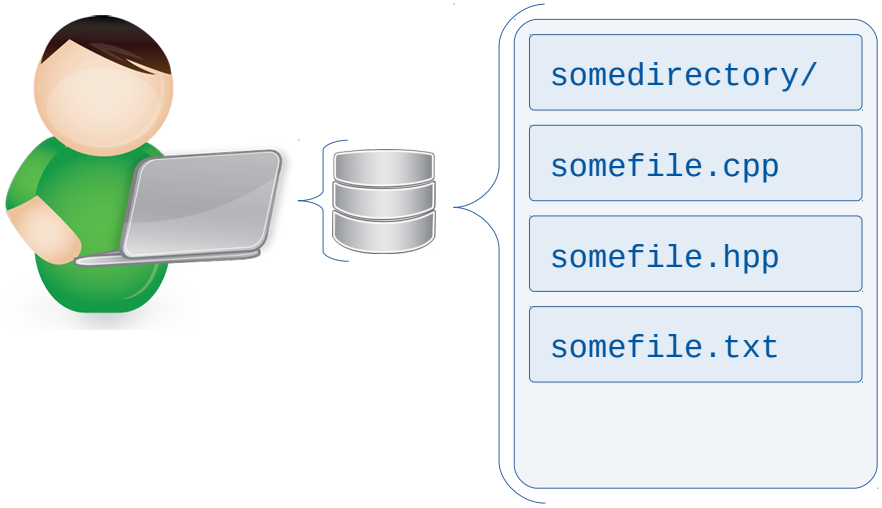


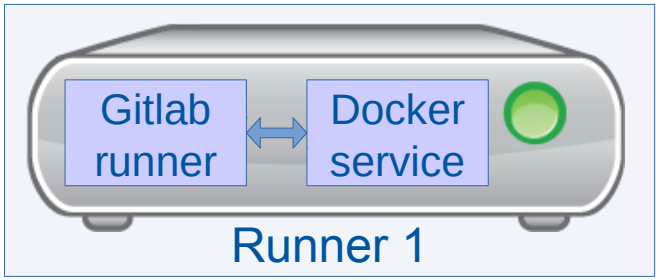


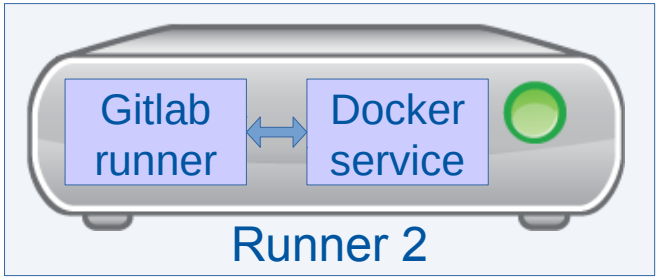


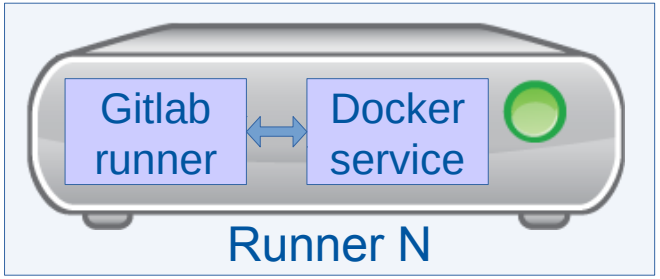


2) Overview - architecture

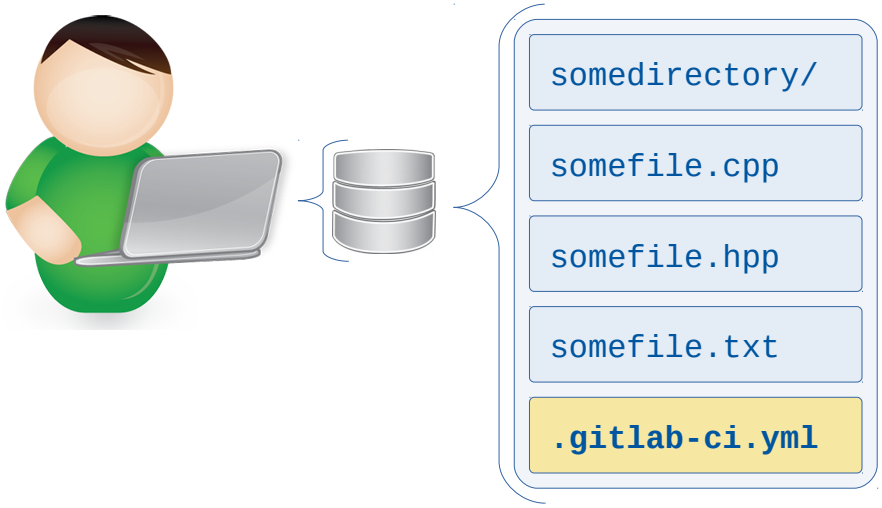


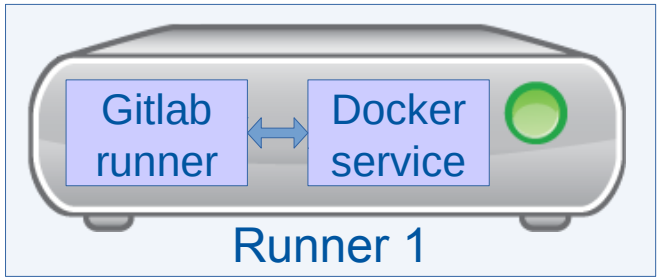


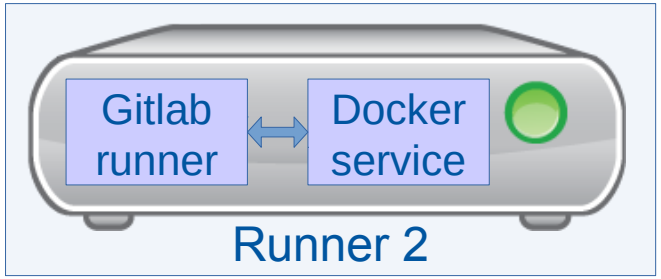


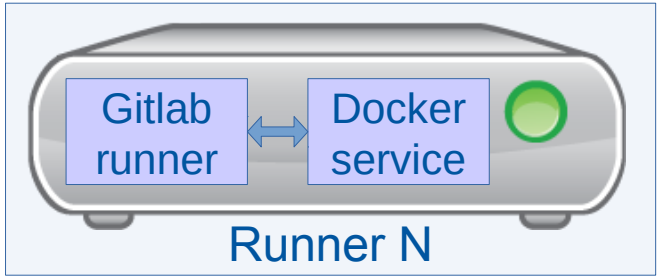


2) Overview - architecture

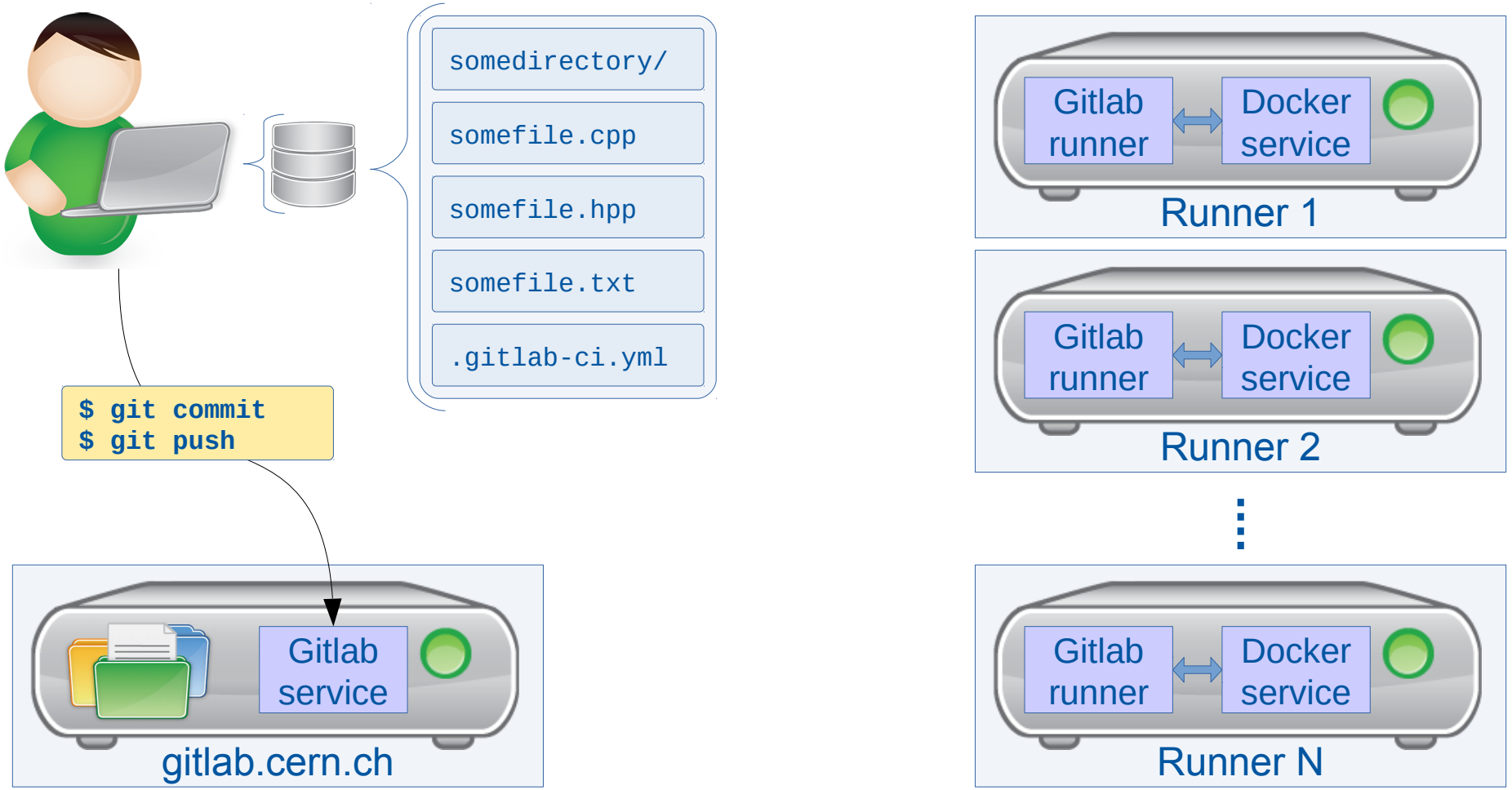




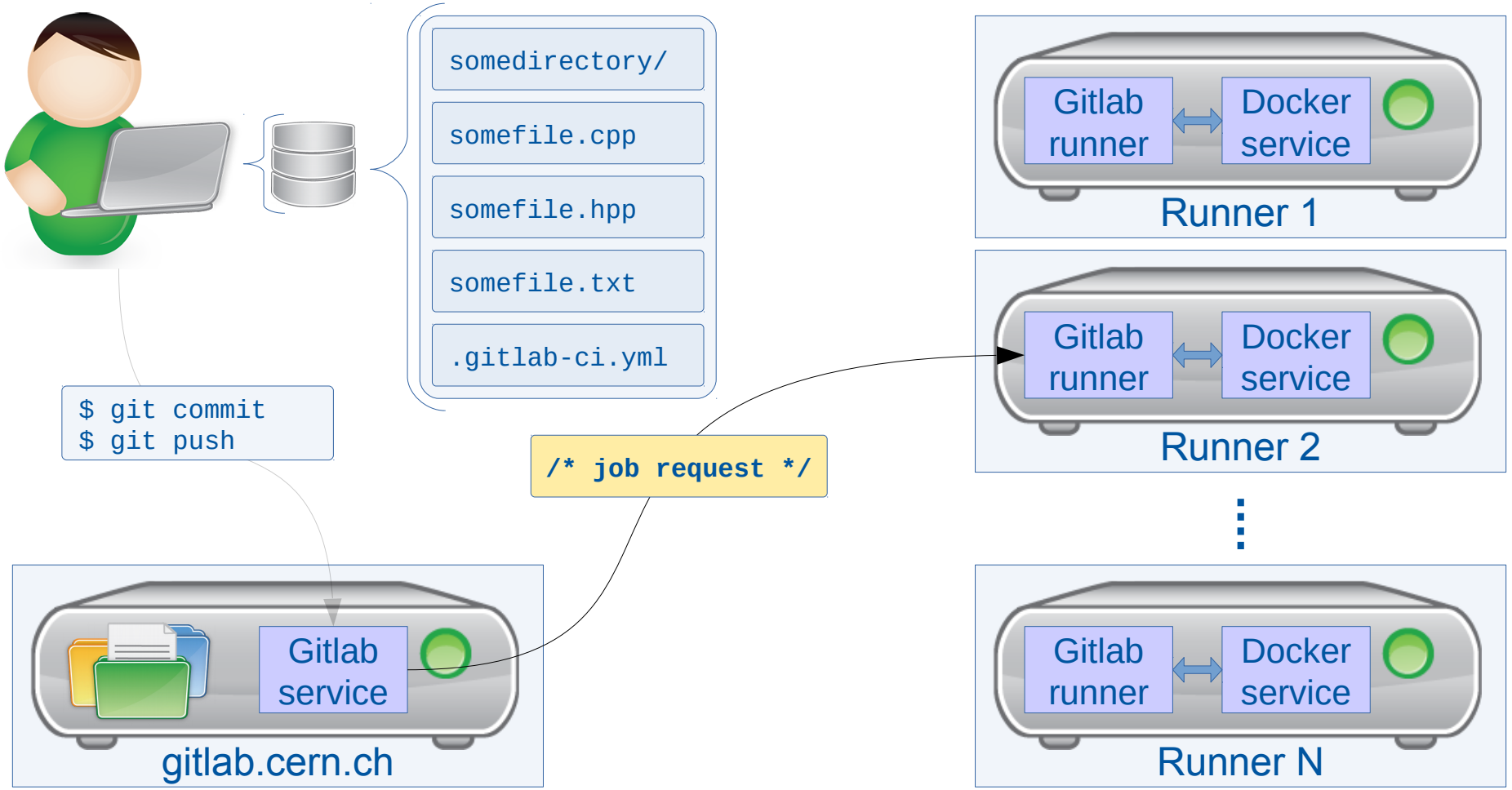




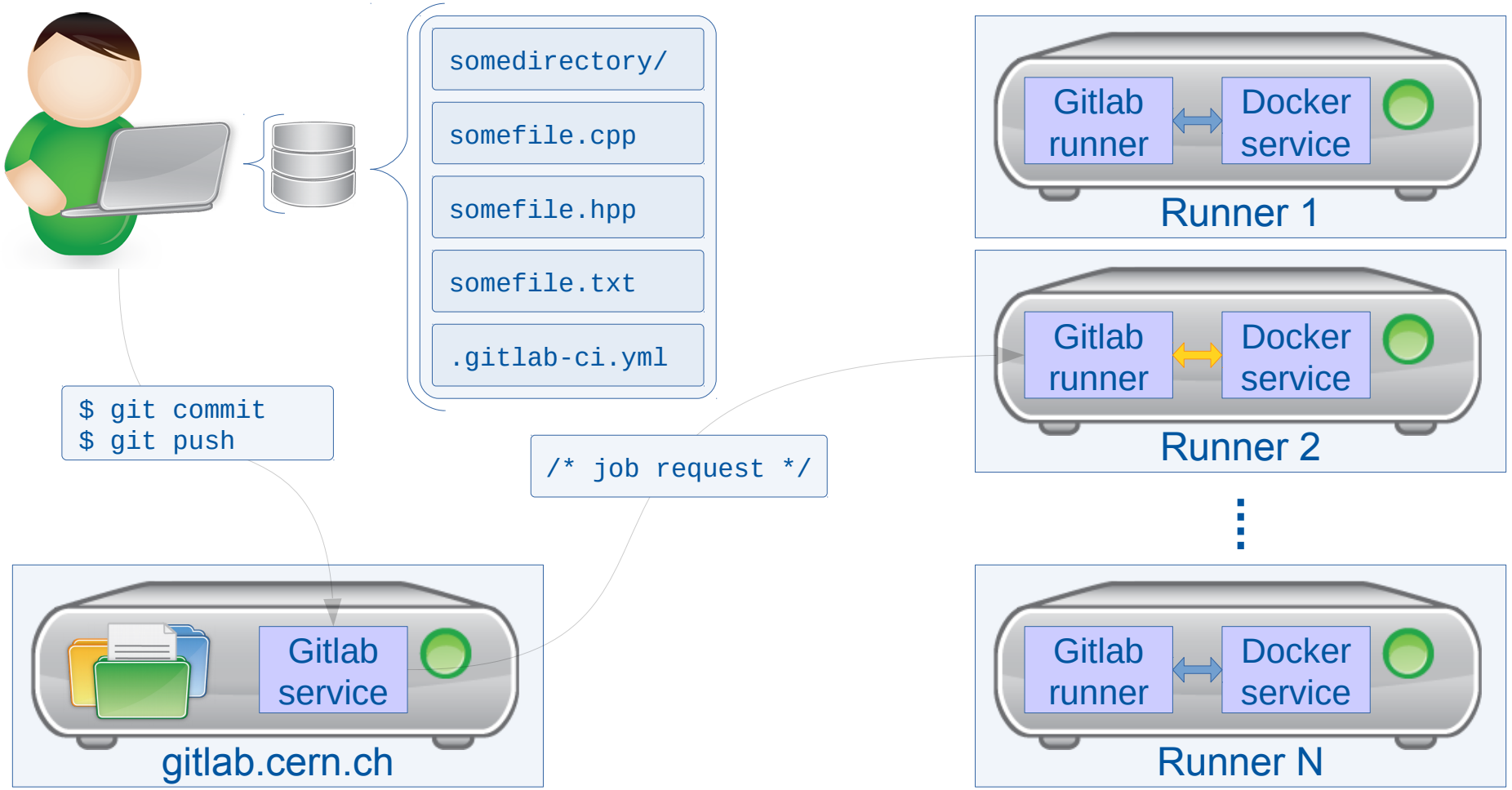
2) Overview - architecture



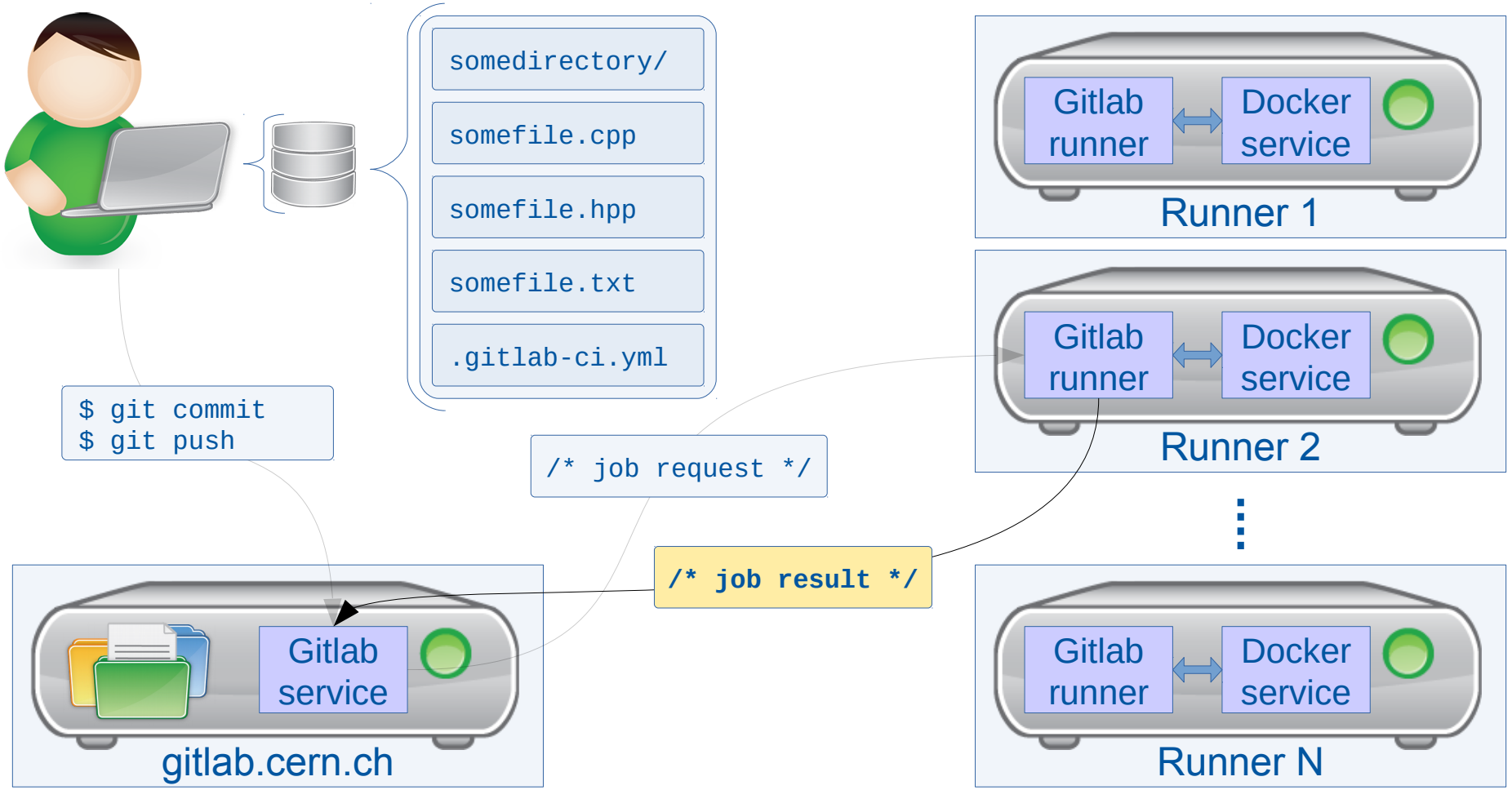
2) Overview - architecture



2) Overview - architecture



2) Overview - architecture



Presentation in a nutshell

1) Motivation – why it may be useful?

2) Overview – how it works in general?

3) Usage – what do I need to do?

4) Summary – where to look for more?

3) Usage - jobs configuration


```
# Example CI workflow definition for Gitlab-CI
image: docker.cern.ch/security-services/code-checking:latest
job1:
  type: test
  script:
    - rats ./* > result.txt
    - check.sh 'result.txt'
  artifacts:
    untracked: true
job2:
  type: test
  script:
    - pylint ./*
```

Comment

Docker image to be used

Job definition

- Job type
- Shell commands to execute
- Definition of job artifacts

3) Usage - jobs results

- badges – out of box, always there

http:// **<gitlab-server>** / **<user>** / **<project>** / badges / **<branch>** / build.svg

Example: *http://gitlab.cern.ch/sdatko/Vulnerable-Codes/badges/master/build.svg*

Get it through user interface: **Gitlab** > **Project** > **Project settings** > **Badges**

- artifacts – some configuration required

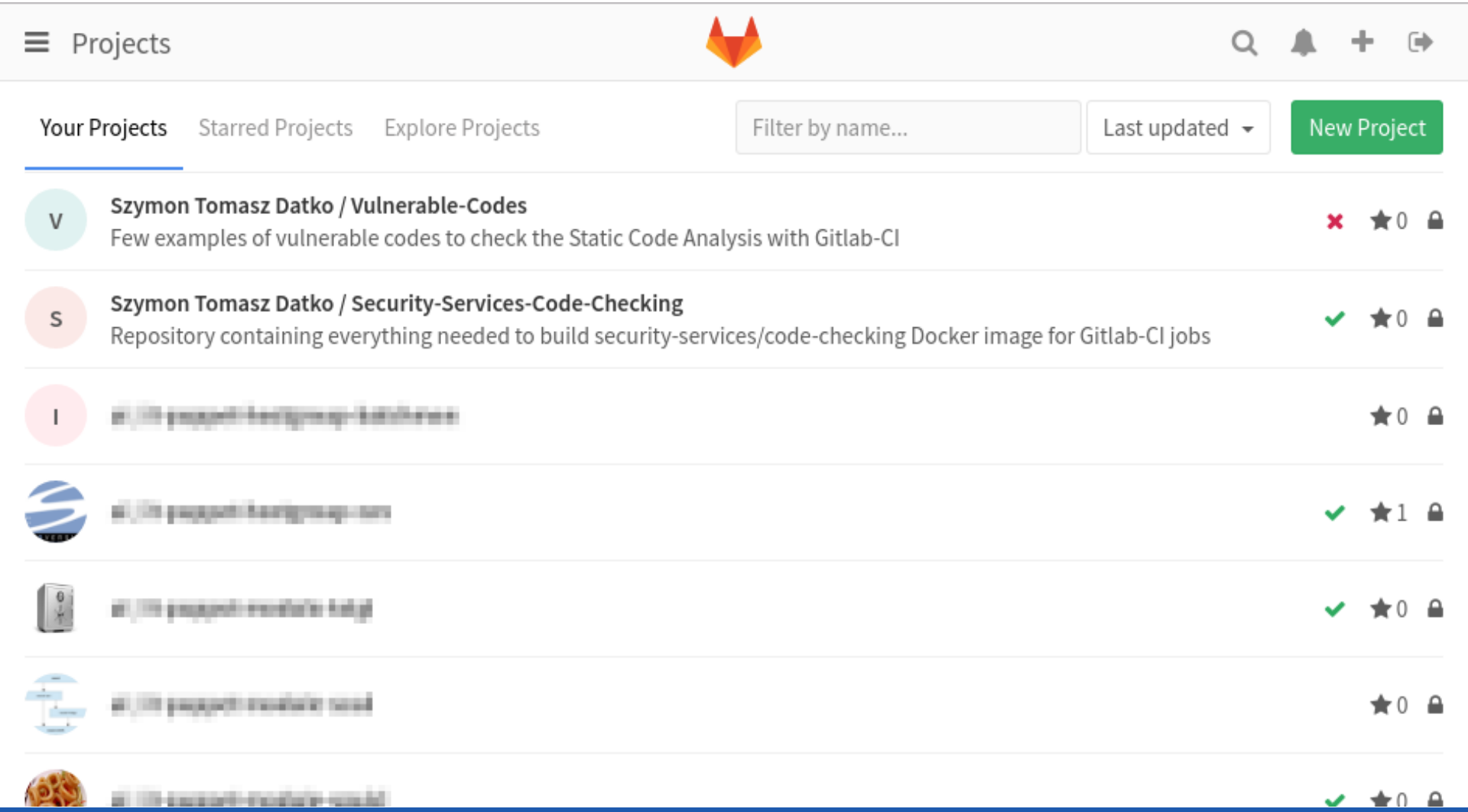
„… specify a list of files and directories which should be attached to the build after success.”

1) Declare them for each job in .gitlab-ci.yml

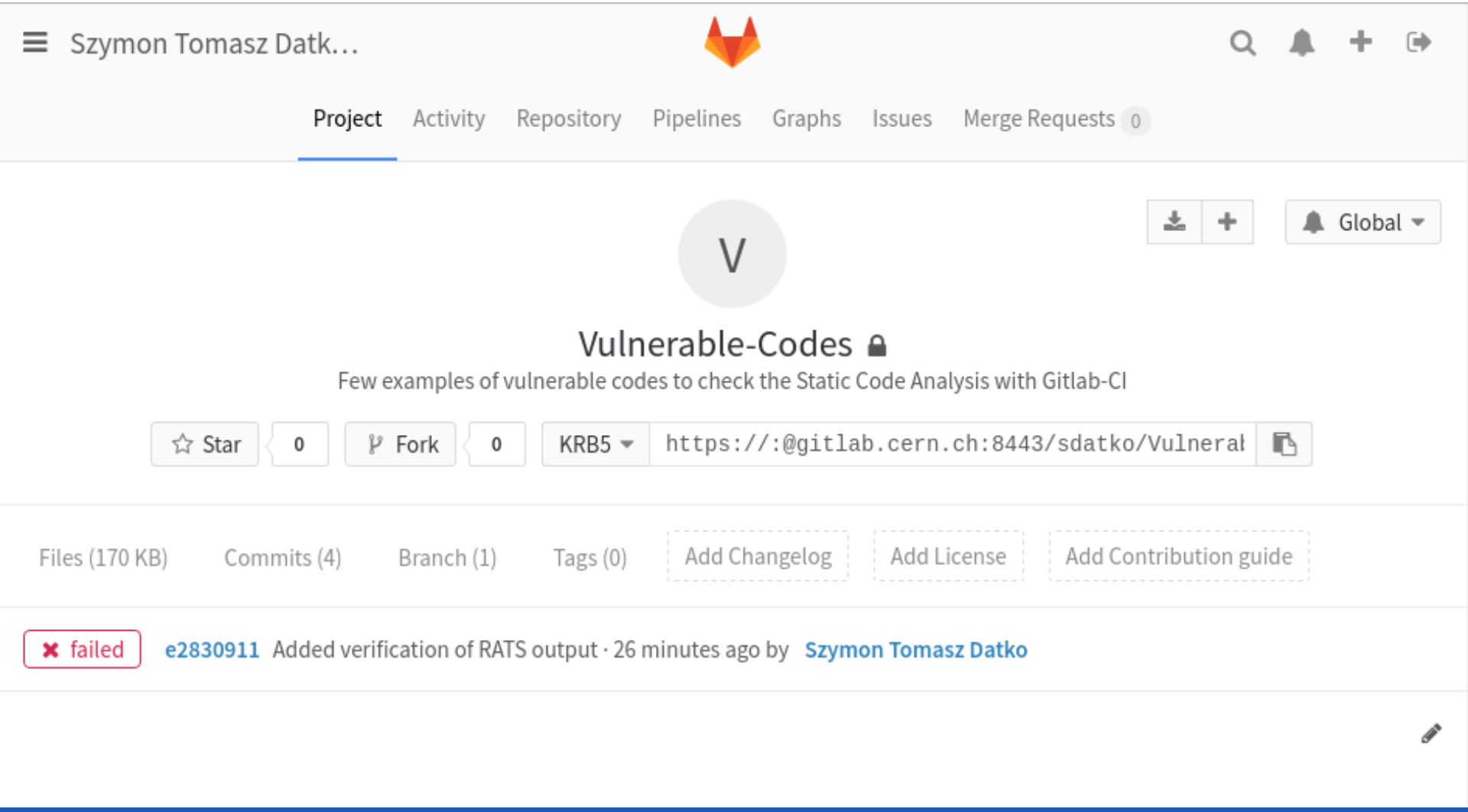
2) Download or browse directly in web interface after job execution

In the future: nice summary dashboard with table/preview of tests result? (~JUnit XML parser)

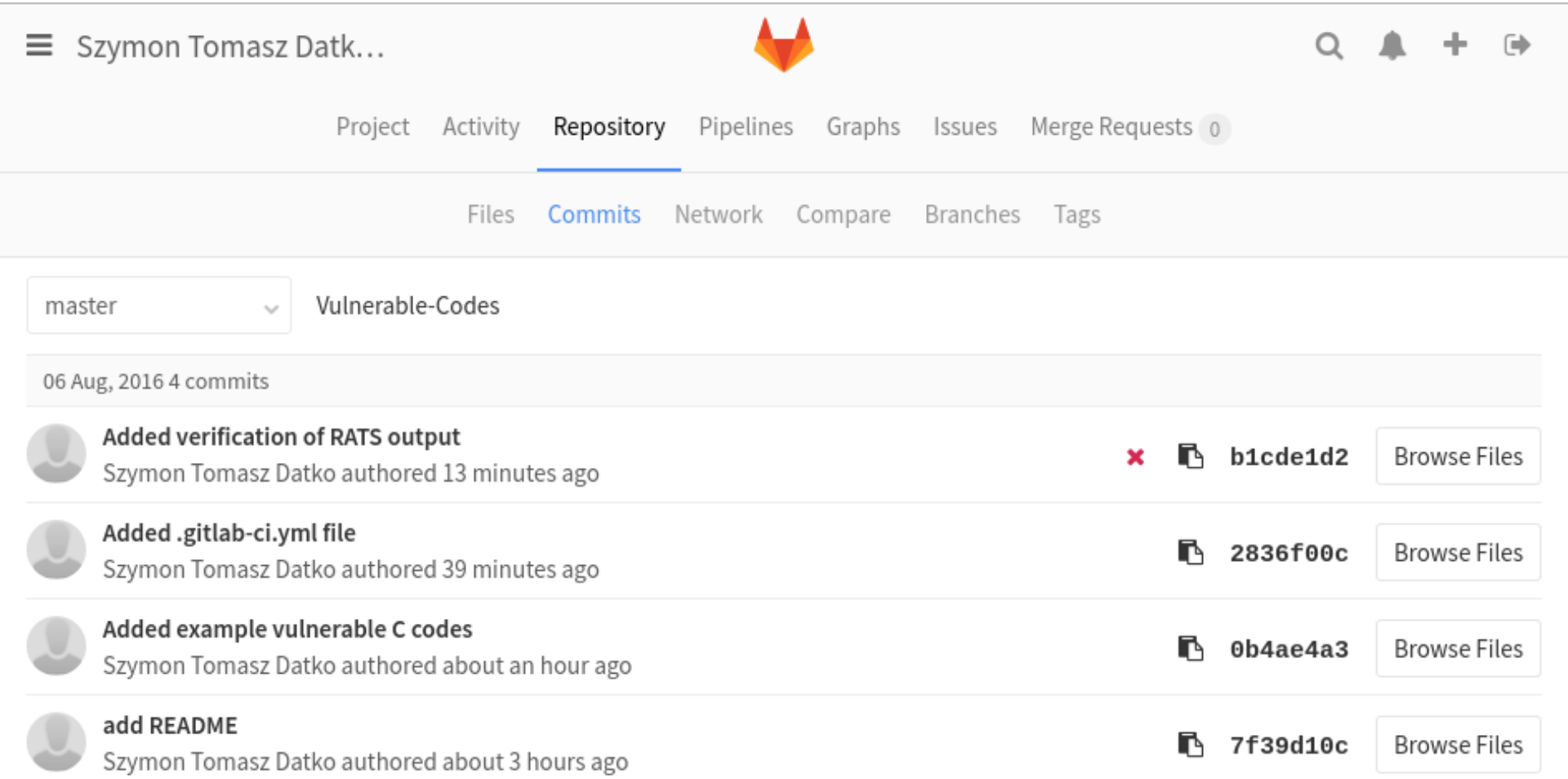
3) Usage - demonstration



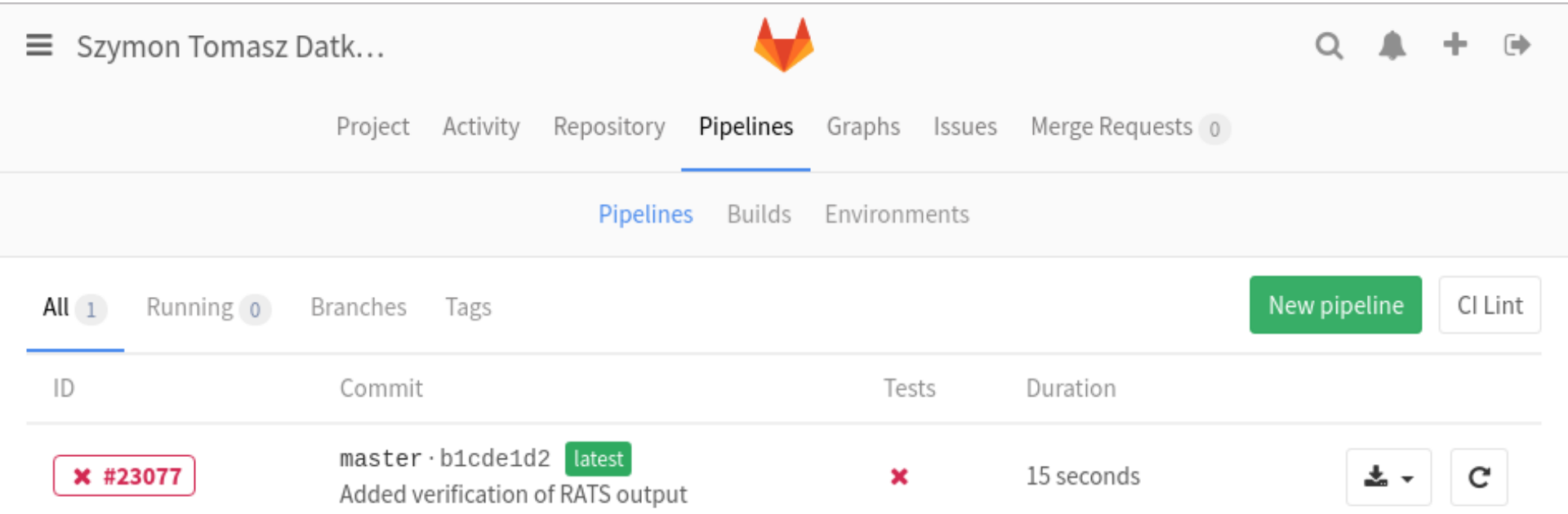
3) Usage - demonstration



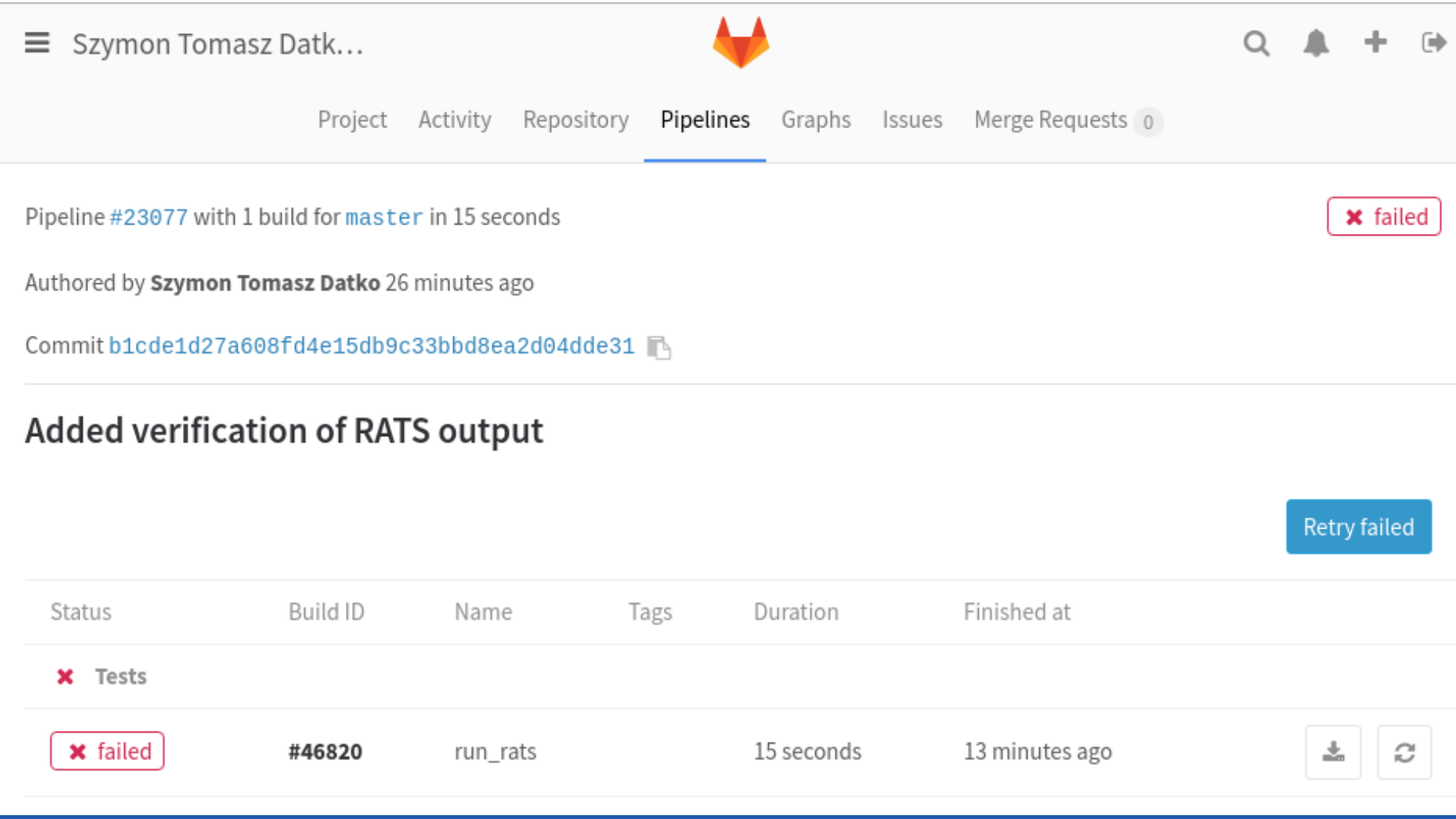
3) Usage - demonstration



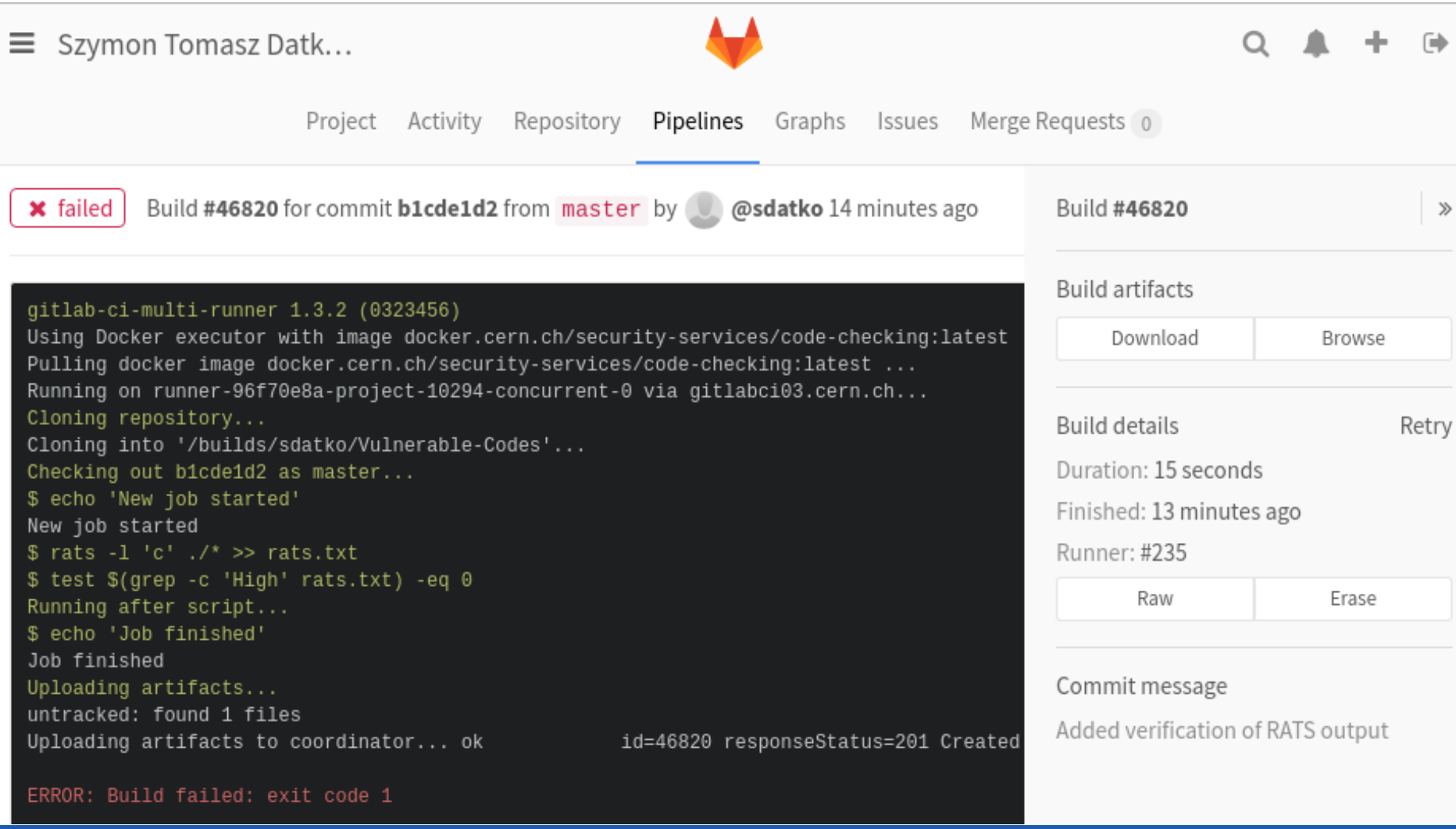
3) Usage - demonstration



3) Usage - demonstration



3) Usage - demonstration



Presentation in a nutshell

1) Motivation – why it may be useful?

2) Overview – how it works in general?

3) Usage – what do I need to do?

4) Summary – where to look for more?

4) Summary

- Gitlab-CI can check your code automatically

- container image with necessary tools is provided

- only one very simple configuration file required…

- ...yet it is powerful!

For details:

`gitlab.cern.ch/gitlabci-examples/`**`Static_Code_Analysis`**

Automate your life with Gitlab-CI

Szymon Tomasz Datko

Summer Student

at

Computer Security Team

9th of August, 2016