

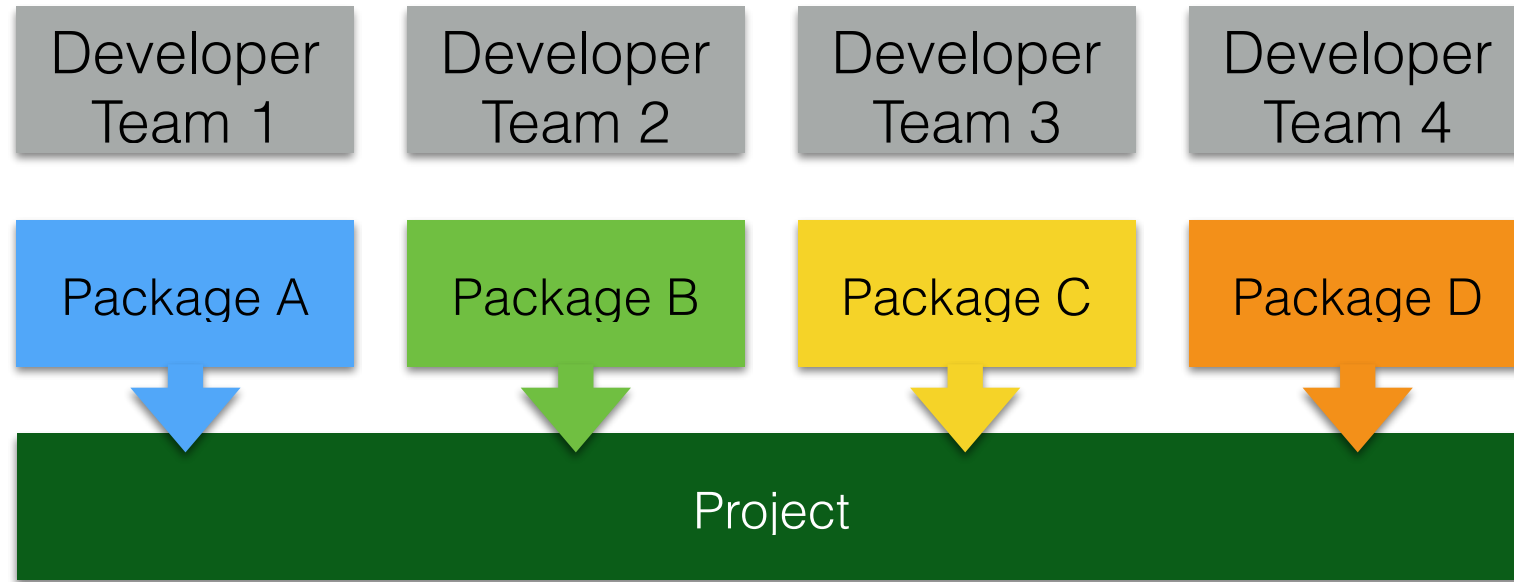
Collaborative Software Design

Continuous Integration & Continuous Deployment

Manuel Giffels (Manuel.Giffels@kit.edu)
KSETA Course — October 2025



Software Projects

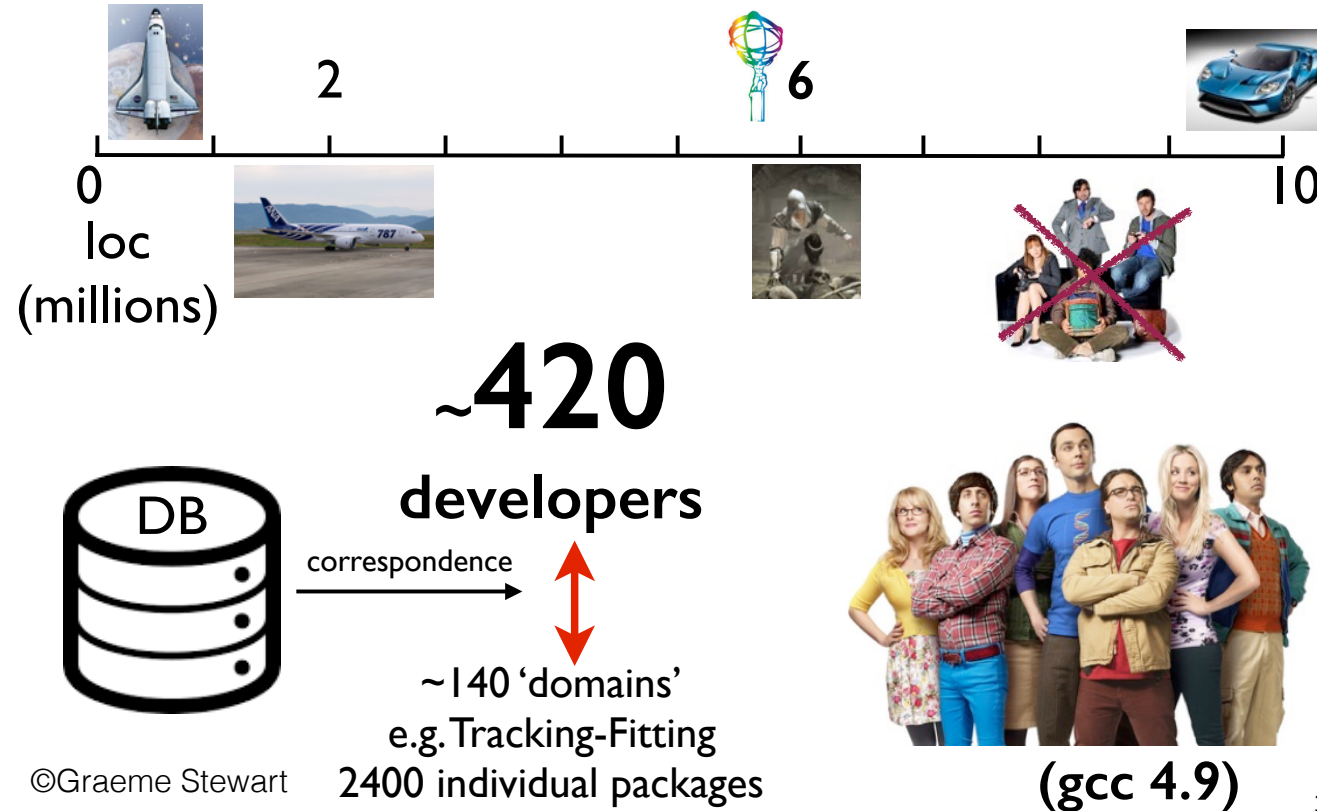


Integration effort increases with:

- Number of packages
- Number of supported platforms
- Number of bugs
- Time since last integration

Recap: The ATLAS Code

ATLAS code



Impossible to integrate without automatism!

Continuous Integration (CI)

„Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.“

Martin Fowler

Continuous Integration (CI)

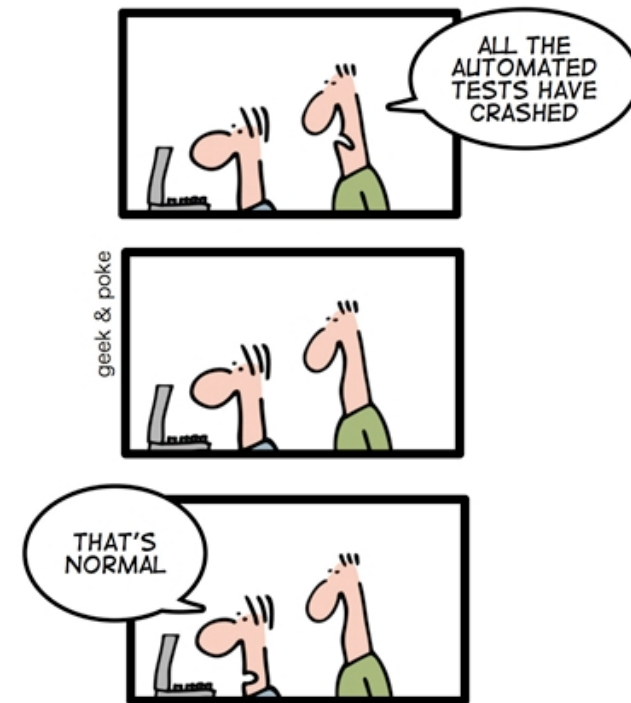
Continuous Integration denotes the possibility to be always ready to release the software to the customer, potentially after every commit

Every time new code is added to the project:

- Project is built and run on all target platforms
- Automated testing and codes style checking
- Ensure other components of the application (like database schema) work properly
- The documentation is up-to-date and building
- Automated deployment or automated release if desired

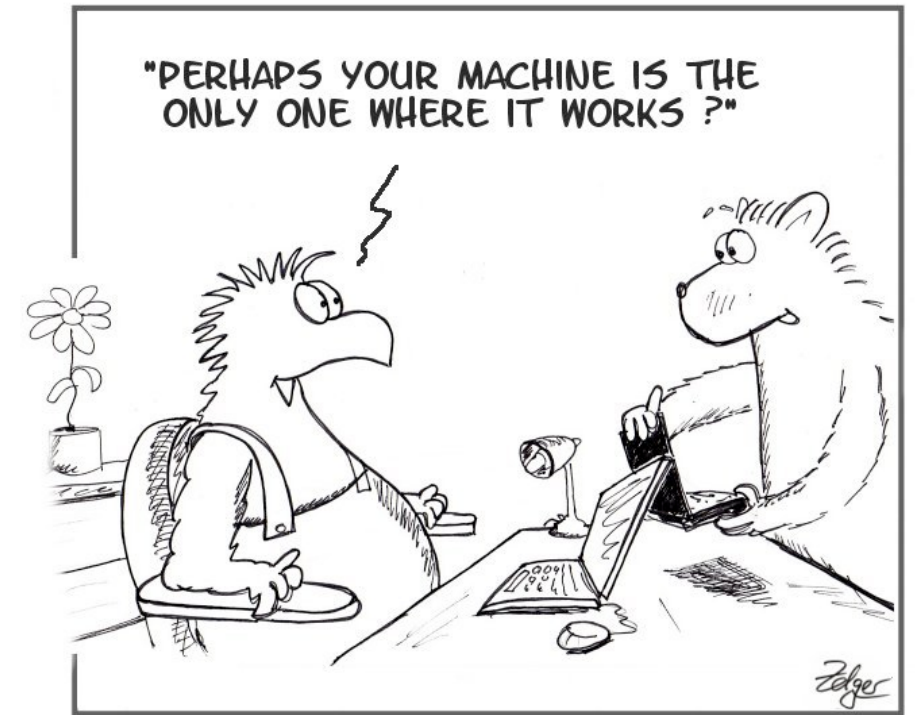
*GEEK & POKE'S LIST OF
BEST PRACTICES*

*TODAY: CONTINUOUS INTEGRATION
GIVES YOU THE COMFORTING
FEELING TO KNOW THAT
EVERYTHING IS NORMAL*



Advantages

- Immediate detection of bugs and integration issues
- Immediate check of code style
- Simplifies code review process
- Ensures a working code base for developments
- Automated deployment and release (if desired)
- Public name and shame towards anyone who breaks the project



CI does not mean that you do not need to test your code before submitting!

Tools



CODESHIP



circleci



Bamboo



Solano Labs



snap



semaphore



AppVeyor

drone.io

The KSETA CI/CD Setup

Luckily the KIT GitLab has already a standard set of GitLab runners enable to perform Continuous Integration

More about it can be found in the documentation

https://docs.gitlab.kit.edu/en/gitlab_runner/

Enable Continuous Integration

To enable continuous integration in a repository, please do the following:

- Add a .gitlab-ci.yml file to your repository
- Here is a simple example for the Scientific Calculator repository

```
simple_example:
  # using tagged runner with tag "kgr1-instance-standard"
  tags:
  | - "kgr1-instance-standard"

  # specifying stage
  # (most common stage and order is build->test->deploy)
  stage: build

  image: "gitlab/gitlab-runner:ubuntu"

  variables:
  | DISPLAY: ":99"

  before_script:
  | - apt-get update -qq
  | - apt-get install -y python3 python3-pip python3-tk xvfb
  | - Xvfb :99 -screen 0 1024x768x24 2>/dev/null &

  # specifying the commands to be run by the runner
  script:
  | - ./run_test.sh
```

Enable Continuous Integration

S

Scientific Calculator 

 main ▾

scientific-calculator /

 ▾

Find file

Edit ▾

Code ▾





Forked from [KIT / KSETA Workshops / KSETA CSD 2025 / Scientific Calculator](#)
1 commit ahead of the upstream repository.

Create merge request



Add CI/CD workflow

Manuel Giffels authored 12 minutes ago



7cb557af 

History

Enable Continuous Integration

Add CI/CD workflow

✓ Passed **Manuel Giffels** created pipeline for commit `7cb557af` 5 minutes ago, finished 4 minutes ago

For `main`

latest branch 1 job 41 seconds, queued for 3 seconds

Pipeline Jobs 1 Tests 0

build

✓ simple_example



Enable Continuous Integration

```
1116 Setting up libheif-plugin-aomdec:amd64 (1.17.6-1ubuntu4.1) ...
1117 Setting up libheif-plugin-libde265:amd64 (1.17.6-1ubuntu4.1) ...
1118 Setting up libheif1:amd64 (1.17.6-1ubuntu4.1) ...
1119 Setting up libgd3:amd64 (2.3.3-9ubuntu5) ...
1120 Setting up libc-devtools (2.39-0ubuntu8.6) ...
1121 Setting up libheif-plugin-aomenc:amd64 (1.17.6-1ubuntu4.1) ...
1122 Processing triggers for libc-bin (2.39-0ubuntu8.6) ...
1123 $ Xvfb :99 -screen 0 1024x768x24 2>/dev/null &
1124 $ ./run_test.sh
1125 .....
1126 -----
1127 Ran 27 tests in 0.954s
1128 OK
✓ 1129 Cleaning up project directory and file based variables
1130 Job succeeded
```

00:00

Enable Continuous Integration

More advanced example using different stages (style, test):

```
stages:
- lint
- test

Style Checking:
stage: lint
tags:
- "kgr1-instance-standard"
image: "gitlab/gitlab-runner:ubuntu"
cache:
  untracked: true
before_script:
- apt-get update -qq
- apt-get install -y python3 python3-flake8
script:
- python3 -m flake8 calculator tests

Unittests:
stage: test
tags:
- "kgr1-instance-standard"
image: "gitlab/gitlab-runner:ubuntu"
variables:
  DISPLAY: ":99"
before_script:
- apt-get update -qq
- apt-get install -y python3 python3-tk xvfb
- Xvfb :99 -screen 0 1024x768x24 2>/dev/null &
script:
- ./run_test.sh
```

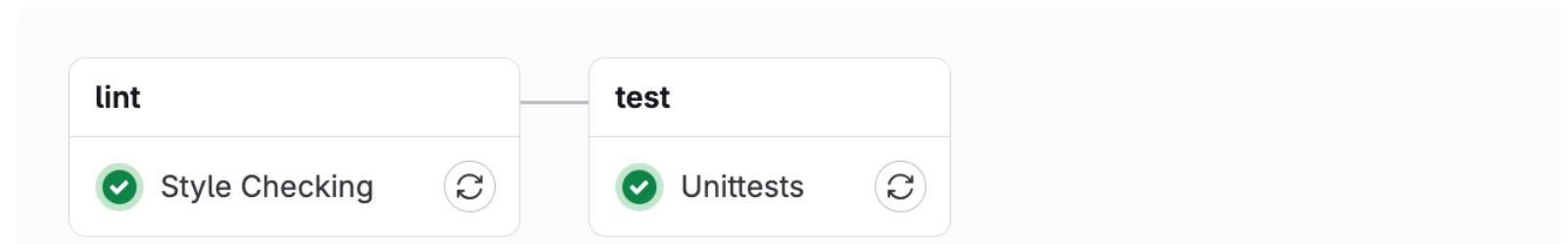
Update gitlab CI/CD

✓ Passed Manuel Giffels created pipeline for commit d983d335 2 minutes ago, finished 1 minute ago

For main

latest branch 2 jobs 52 seconds, queued for 3 seconds

Pipeline Jobs 2 Tests 0



More information can be found in <https://docs.gitlab.com/ci/>



Questions?



Hands-on

Hands-on

- Again work together in teams of two people
- Activate a simple GitLab CI on your private fork of the Scientific Calculator project, running unit tests, check code coverage and perform a flake 8 code style check