

Continuous Integration and Automated Code Review in Open Source Projects

Adrián Tóth

Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2. 602 00 Brno - Královo Pole
xtotha01@fit.vutbr.cz



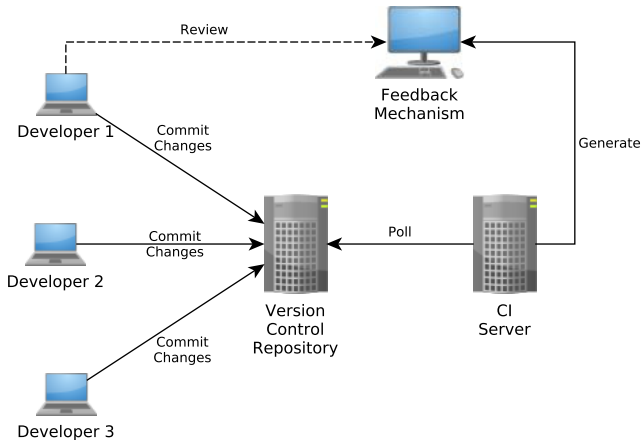
What is *Continuous Integration*?

What is *Automated Code Review*?

Where is it used and why?

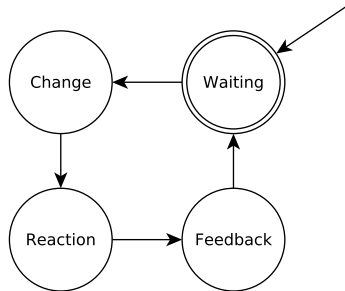
How it works?

- common part of fast software development
- adaptive development technique
- reduce integration problems
- integrations are verified via automated tests and builds
- popular in open source projects which are frequently developed by a group of people
- available CI services: *Travis CI, Jenkins, TeamCity, ...*



Stages of CI:

- 1 Change
- 2 Reaction
- 3 Feedback
- 4 Waiting

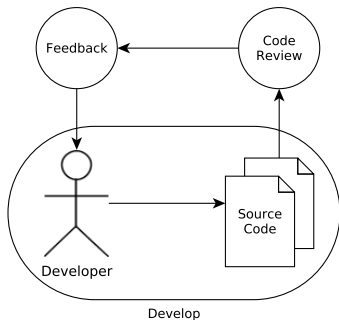


Types of code review:

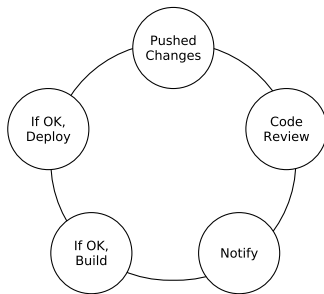
- manual code review
 - collaborative inspection and discussion with project members
 - slow (nearly 100 lines of code per hour)
 - pair programming
- automated code review
 - ...

- inspection of code quality
 - coding standards
 - trailing spaces
 - code duplication
 - not enough / too many comments
 - ...
- detection of basic mistakes and vulnerabilities
- matching set of rules providing static analysis
- Code Climate, RuboCop, SonarQube, RIPS, FlexeLint

Manual



Automated



- *Pronto*¹ integration
- *Webhooks*² integration
- Request for review command implementation
- *Gitter* integration (yell if master is broken)
- *Pull Request Processor*³ integration
- *Github Status API*⁴ integration
- Track dependent pull requests (comment if merged)

¹github.com/prontolabs/pronto

²developer.github.com/webhooks

³github.com/theforeman/prprocessor

⁴developer.github.com/v3/repos/statuses

Thank You For Your Attention !