Effective CI using Automated Quality Checker

Ruchir Garg
October 8th, 2018
Flour  Sugar  Eggs  Chocolate  Baking Powder

Baker (Quality Checker)  Mixer (CI System)  The Cake (Final Product)
Qualities of a great baker

- Quick decisions (uses automation)
- Eye for bad eggs (avoids regression)
- Measures precisely (avoids technical debt)
- Follows the recipe (adheres to process)
- Rejects unwanted ingredients (ignores non-production code)
- Records observations (provides metrics for improvements)
Quality Checker (The Baker)

- Fully automated system
- Part of the CI system
- Improves the effectiveness of the CI system

Let us explore how....
Avoid technical debt

● New code accepted only with new tests
● Builds a habit
● No technical backlog

HOW

● Define tests directories
● Checker looks for tests committed along with production code
● Merge Requests (MRs) / Pull Requests (PRs) are blocked if no tests are found
...Avoid technical debt

Checker blocking a merge request due to missing tests
...Avoid technical debt

Checker marking the request when the author adds some tests

Checker lauding the author when the code is submitted along with the tests
Don’t break the working code

- New code must pass the regression tests
- Unit tests, integration tests, functional tests
- Safety net adds confidence

**HOW**

- The checker validates the result of regression tests
- Build is marked failed on a test failure
- MRs / PRs are blocked
Maintain code coverage

- Add **sufficient** new tests
- Ensures code coverage is maintained

**HOW**

- Code coverage is measured using tools like *Codecov*
- If coverage lowered, MRs / PRs are blocked
Ignore non-production code

- Rules meant only for production code

**HOW**

- Define directories that contain non-production code
- Checker skips all such directories
- Manual override is available for exceptions

*Checker skipping a merge request containing only non-production code*
Master the process

*Reviewer’s accountability*

- Manual review by reviewer is necessary
- Review feedback must be provided in time
- The checker can ensure this through a timer/notification
Master the process

Work-in-progress = Ignore

- MRs / PRs are often marked WIP/work-in-progress
- Ensures efforts are not wasted on incomplete code

**HOW**

- The checker looks up such strings and marks the MRs
- The checker then skips scanning such MRs and blocks the merge

![Checker adding a “work-in-progress” label to the merge request](image)
...Master the process

Checker blocking the merge request unless the “work-in-progress” label is removed

Checker removing the “work-in-progress” label from the merge request

Checker unblocking the merge request
Provide reason for the merge request

- MRs / PRs are usually submitted in context of a task or an issue
- References help reviewers with a context

**HOW**

- The checker looks up MR’s description for a reference
- The checker then marks such MRs
- Optionally, minimum description text could be mandated
...Master the process

Checker marking a merge request, as a link to the issue it fixes, is missing
Deep insights

- Checker not only provides quality software, but also quality metrics
- Such data can provide deep insights into problem areas

HOW

- All quality check failures are recorded

*Let us explore how...*
Deep insights

**Symptom:** A significant number of MRs blocked due to missing or insufficient tests

**Insight:** Skill gap

**Symptom:** Significant MRs that failed regression tests

**Insight:** Carelessness or lack of design knowledge

**Additional Insight:** This also indicates the number of defects that could have entered production, but didn’t
Symptom: A significant number of defects were found during MR’s review process

Insight: The review process is working and is very effective

Additional Insight: All defects found during review are early defects
Deep insights

- Changes requested via manual review could be tracked as individual defects
- Special tags like `/defect-high Missing null check` could be used for tracking
- Author can mark a defect resolved `/defect-high-fixed Missing null check`
- The checker keeps a count of such defects
- For a MR/PR to merge, all defects reported against it must be zero
- Option to manually override
…Deep insights

**Insight:** Code-to-test matrix is possible if both reside in a common system

**Benefit:** This could be used for smart regressions

**Insight:** A defect could be traced back to its origin if code and defects are managed in a common system

**Benefit:** This could be used to learn about the common causes of defect injection
Process recap

1. Merge Request / Pull Request Submitted
2. Quality Checker
   - Test Check
   - Regression Check
   - Coverage Check
   - WIP Check
   - Production Code Check
   - Rejected/Blocked
3. Merge Request / Pull Request Approved for Review
4. Merge Request / Pull Request Reviewed and Approved
5. Merge Request / Pull Request Merged to Master Branch
Final thoughts

- Checker not only enables quality software, but also process compliance and quality metrics
- Self-sustaining model of quality that keeps pace with development
- A system like this need to be invested in and developed as readymade solutions aren’t available
- Folks using GitHub have a head start. You can look at the following open source tool developed by engineers at Red Hat:

  Arquillian – Ike-prow-plugin:  
  http://arquillian.org/ike-prow-plugins/
References

Codecov
https://codecov.io/

Arquillian – Ike-prow-plugin
http://arquillian.org/ike-prow-plugins/

GitHub
https://github.com/
Reach me @

Ruchir Garg

@ ruchirg@gmail.com / rgarg@redhat.com

https://www.linkedin.com/in/ruchirgarg

https://twitter.com/ruchirg