CookBook rubrics mapping to Linux Kernel Best Practices

Shreyash Mohatta
shreyashm12@gmail.com

Karan Kajani
karan.kajani98@gmail.com

Priyanka Ambawane
dearpriyankasa@gmail.com

Asrita Kuchibholta
asritakuchibhotla@gmail.com

Vinit Desai
vinit.desai98@gmail.com

ABSTRACT

CookBook is being built as a one stop platform for looking up recipes based on ingredients you have in your fridge. Being college students, we know how much of a pain it is to decide what to cook next. The monotony of same dishes turning up in our heads and subsequently onto our dining tables. We are here trying to help out people going through the same things. CookBook is built on top of a data set of >200,000 recipes and planned to be expanded further through a general purpose web scraper. It shows all the recipe information on a concise page with important macros right in front of you so that you can make an informed decision and enjoy a delicious meal!

ACM Reference Format:
Shreyash Mohatta, Karan Kajani, Priyanka Ambawane, Asrita Kuchibholta, and Vinit Desai. 2022. CookBook rubrics mapping to Linux Kernel Best Practices. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 SHORT RELEASE CYCLES

Shorter release cycles ensure less frustration for the user and the developer. It reduces the chances of merging and pushing inefficient and unstable code as shorter cycles ensure regular testing and update knowledge of the system architecture. This method also helps to bring fast and positive changes which result in the end user being satisfied. For these reasons, we integrate new code in short cycles. The rubrics mentioned in table can be mapped to this practice

Issues reports: there are many
Issues are being closed
Tests that can be run after your software has been built or deployed to show whether the build or deployment has been successful
Automated test suite for your software
Framework to periodically (e.g. nightly) run your tests on the latest version of the source code
Using continuous integration, automatically running tests whenever changes are made to your source code
Test cases are routinely executed

Conference'17, July 2017, Washington, DC, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 DISTRIBUTED DEVELOPMENT MODEL

A distributed Development model is the best way to develop any software. Sharing different functionalities of the software to different individuals, based on their familiarity with the area ensures seamless code review and integration with very minimal chances of blow-up. For this reason, Distributed Development Model has been followed. The following rubrics can be mapped to this practice

Number of commits
Workload is spread over the whole team
Number of commits: by different people
evidence that the whole team is using the same tools
E-mails to our support e-mail ad- dress are received by more than one person
Listing the important partners and collaborators on our website
Do we accept contributions from people who are not part of your project?
Do you have a contributions policy
Is your contributions' policy publicly available?
Evidence that the members of the team are working across multiple places in the code base

2.1 CONSENSUS-ORIENTED MODEL

Integration to the code base need to be agreed upon by all and especially by people who have implemented some functionality and the new code block directly works with that. This ensures not tampering with the stable versions of code. The following rubrics can be mapped to this practice

Chat channel: exists
issues are discussed before they are closed
Project has an e-mail address or forum that is solely for supporting users

2.2 THE NO-REGRESSIONS RULE

The No-regression rule is an important design decision as once the interface with the model gets pushed and is in public use, we should not alter that syntax. This ensures harmony in terms of user calls and less frustrations. We have ensured that we don't take away existing functionality but add to it. The following rubrics can be mapped to this practice

Use of version control tools
Evidence that the members of the team are working across multiple places in the code base
There is a branch of the repository that is always stable

3 ZERO INTERNAL BOUNDARIES

We understand that access to the entire view of the project is important. Even though individuals are working on different functionalities, it does not stop them from making changes in other parts of the code. This results in problems being solved at the source rather than having multiple paths to go through before making actual changes. The following rubrics can be mapped to this practice

whole team is using the same tools
Source code publicly available to download, either as a download- able bundle or via access to a source code repository
E-mails to our support e-mail ad- dress are received by more than one person
Project have a ticketing system to manage bug reports and feature requests