Patterns for research software management using GitHub

Ruairidh MacLeod, EPCC, The University of Edinburgh

r.macleod@epcc.ed.ac.uk

2021-01-13

www.archer2.ac.uk

Reusing this material

Partners

Engineering and Physical Sciences Research Council

Natural Environment Research Council

THE UNIVERSITY of EDINBURGH

CRAY®

a Hewlett Packard Enterprise company

Motivation & Overview

Motivation

- Multitude of tools used to manage software projects
  - Source control management, issue trackers, change management tools, Kanban/Trello boards, automated build/test systems, chat apps (Slack/Mattermost), wikis & forums, web hosting, …

- Why?
  - Significant effort required to effectively manage a software project beyond software development itself
    - Planning and tracking, communication, quality control, ensuring documentation exists and is accurate, release management, …
  - Code itself requires review and management
  - Even solo-run projects need to manage external collaboration
  - -> We rely on tools to simplify this process

- “Indeed, the ratio of time spent reading versus writing is well over 10 to 1. We are constantly reading old code as part of the effort to write new code. ...[Therefore,] making it easy to read makes it easier to write.” -Robert Martin (Clean Code)

Motivation

- Why GitHub?
  - Provides majority of tools listed previously, for free
  - Advantageous to use a single service containing all required tools. Benefit from cohesion. Only have to learn one interface and “ecosystem”
  - Unlimited free public/private repositories. Educational packages available to support students/teachers
  - Free CI (automated testing) pipelines with GitHub Actions
  - Is the de facto standard
- Why not GitLab, or …?
  - More cluttered and complex user interface
  - Lower visibility than GitHub (100k users vs. 40M)[1,2], smaller community
  - Focused on “complete DevOps” platform rather than developer productivity

Overview

- Creating a high-quality, accessible repository
  - What information should be available and easy to find
  - Writing an effective README in markdown
  - Easily searchable & navigable directory layout
  - Automatically publishing documentation and static webpages

- Effectively managing collaboration
  - Within your own team, and with external contributors
  - Issues, pull requests, CI, and code reviews
  - Code linters, formatters
  - Project boards, milestones, wikis

Overview

- Miscellaneous features
  - GitHub CLI & Desktop
  - Dependabot and automatic vulnerability scanning
  - Managing teams and permissions
  - Integrations aplenty
  - …

- Discussion & Conclusions

Overview

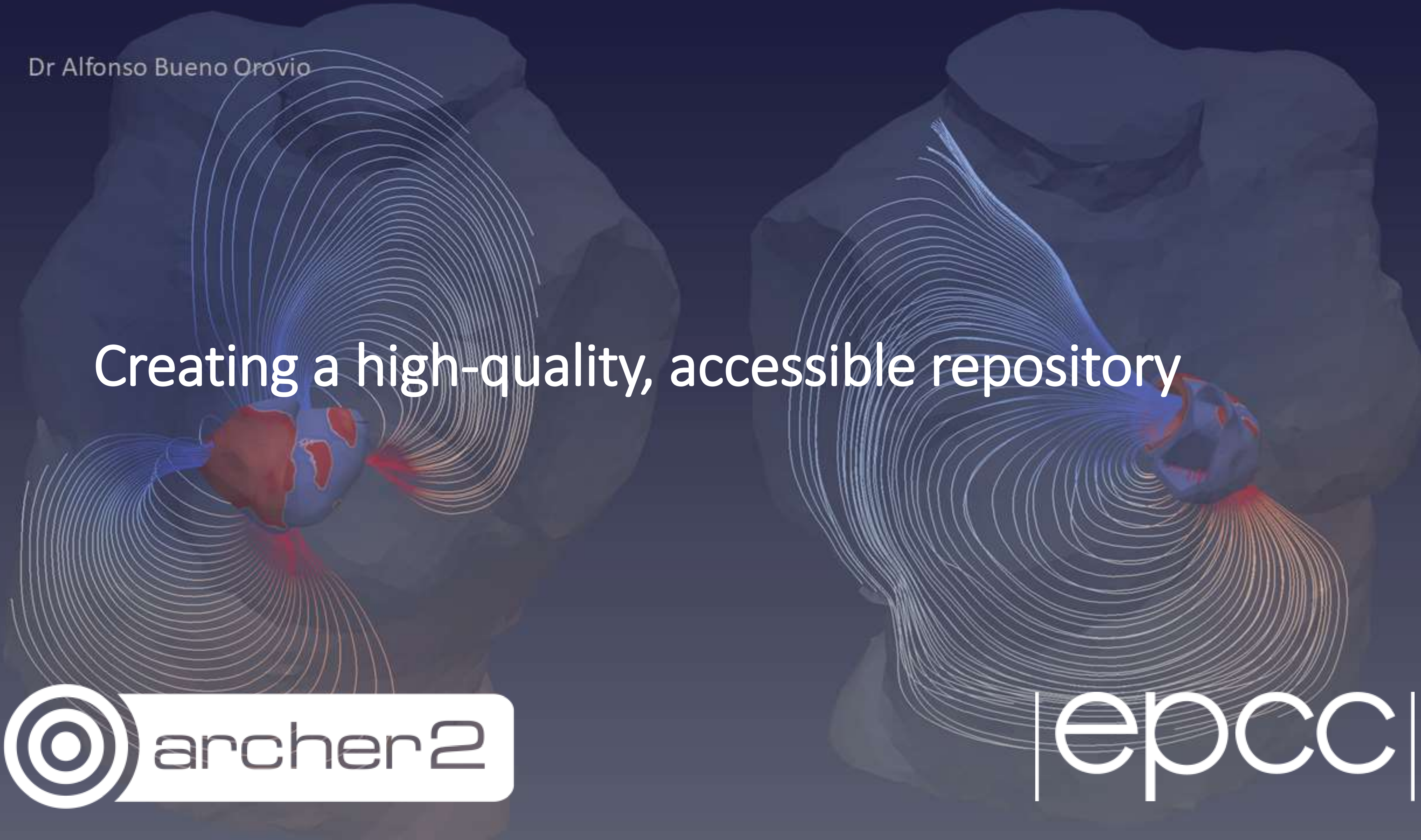
- Won’t cover: Version Control & Git
  - No valid excuse for not using version control in your project!
  - Git is the most widely used today. Plenty of learning resources available, e.g.
    - Software Carpentry Tutorial: https://github.com/swcarpentry/git-novice
    - ARCHER Virtual Tutorial: https://www.youtube.com/watch?v=P6drmyCNEWU
  - Basic understanding of git and git branches is sufficient for this session
  - Many interactive and visual tools available – GitHub Desktop

- Already familiar with GitHub?
  - Great – will also cover some best practices and lesser-known features

Overview

- Disclaimer
  - Contains some opinionated content
  - Not in any way endorsed or sponsored by GitHub!

- Discussion encouraged – please use chat or “raise your hand”



Creating a high-quality, accessible repository

Creating a high-quality, accessible repository

- **Goals**
  - Ensure your project is easy to find, use, and learn. Provide a means for users to interact and provide feedback
  - Ensure your team, the primary users of your repository, have low friction in getting their tasks done

Creating a high-quality, accessible repository

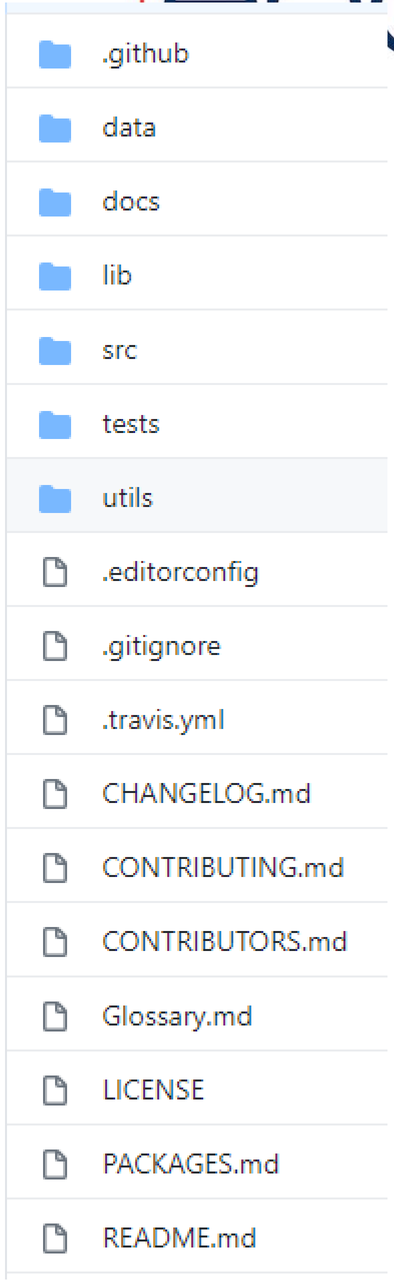
- What does that look like?
- Low-quality, inaccessible
  - No landing page or README
  - Basic README in plain text, no links to relevant content, little detail
  - No visible documentation
  - All documentation contained in a single Word document
  - Every file in the top-level directory
  - No means of contact (i.e. issue tracker disabled)
  - All above are real examples…
- High-quality, accessible
  - Anything other than the above

Creating a high-quality, accessible repository

- High-quality, accessible
  - Code files - structured in directories and easy to navigate
  - Detailed README markdown file, with links and useful sections
  - Few clicks required to find the most pertinent information
  - Open issue tracker with templates
  - Documentation available
  - Guidelines for contributions
  - Webpage (optional – but GitHub covers this also)
  - View releases and version history in a CHANGELOG
  - Licence file

Source Code Layout

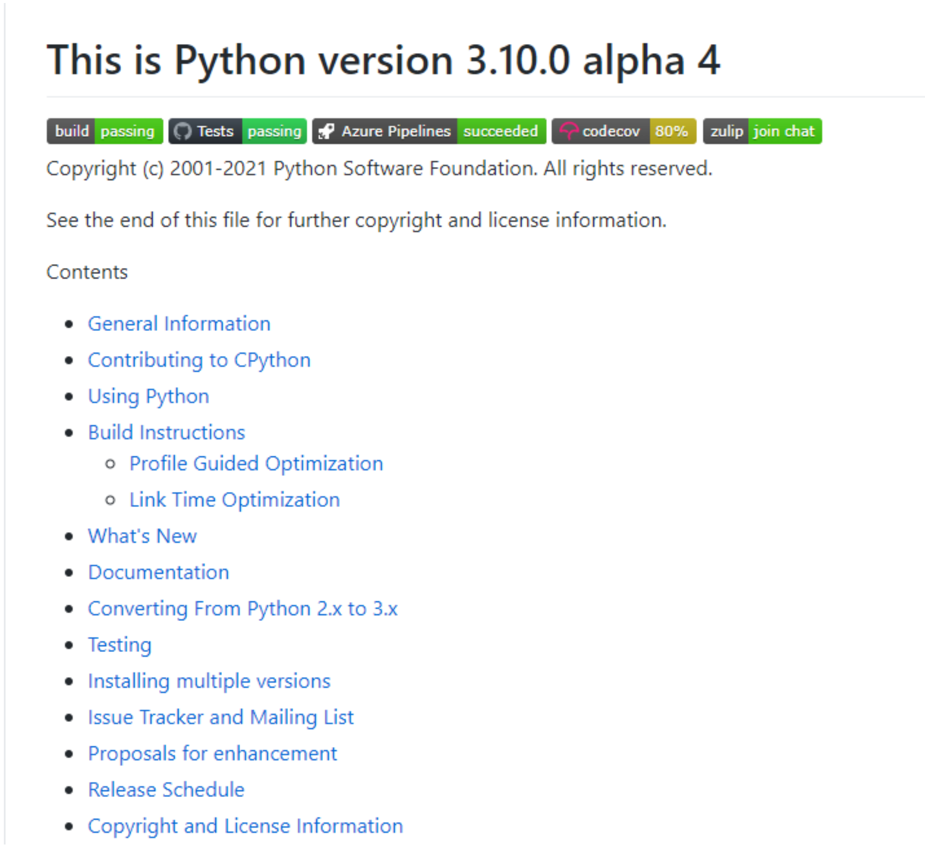
- Use a standard, clear directory layout
  - src/ for source code
  - tests/ data/ docs/ etc.
- Language dependant – some have prescribed layouts
- Look at some popular projects for examples
- Tips
  - Press “t” to perform a fuzzy file search in the GitHub UI
  - “Hide” meta / utility files by using dot prefix



```
.github
data
docs
lib
src
tests
utils
.editorconfig
.gitignore
.travis.yml
CHANGELOG.md
CONTRIBUTING.md
CONTRIBUTORS.md
Glossary.md
LICENSE
PACKAGES.md
README.md
```

README.md

- The landing page – automatically detected and rendered by GitHub
- Should include
  - Project summary and purpose
  - Links to other sections and content
  - Build/usage instructions
  - Usage examples
  - FAQ
  - Contributing guidelines (or as a separate file)


This is Python version 3.10.0 alpha 4

Copyright (c) 2001-2021 Python Software Foundation. All rights reserved.

See the end of this file for further copyright and license information.

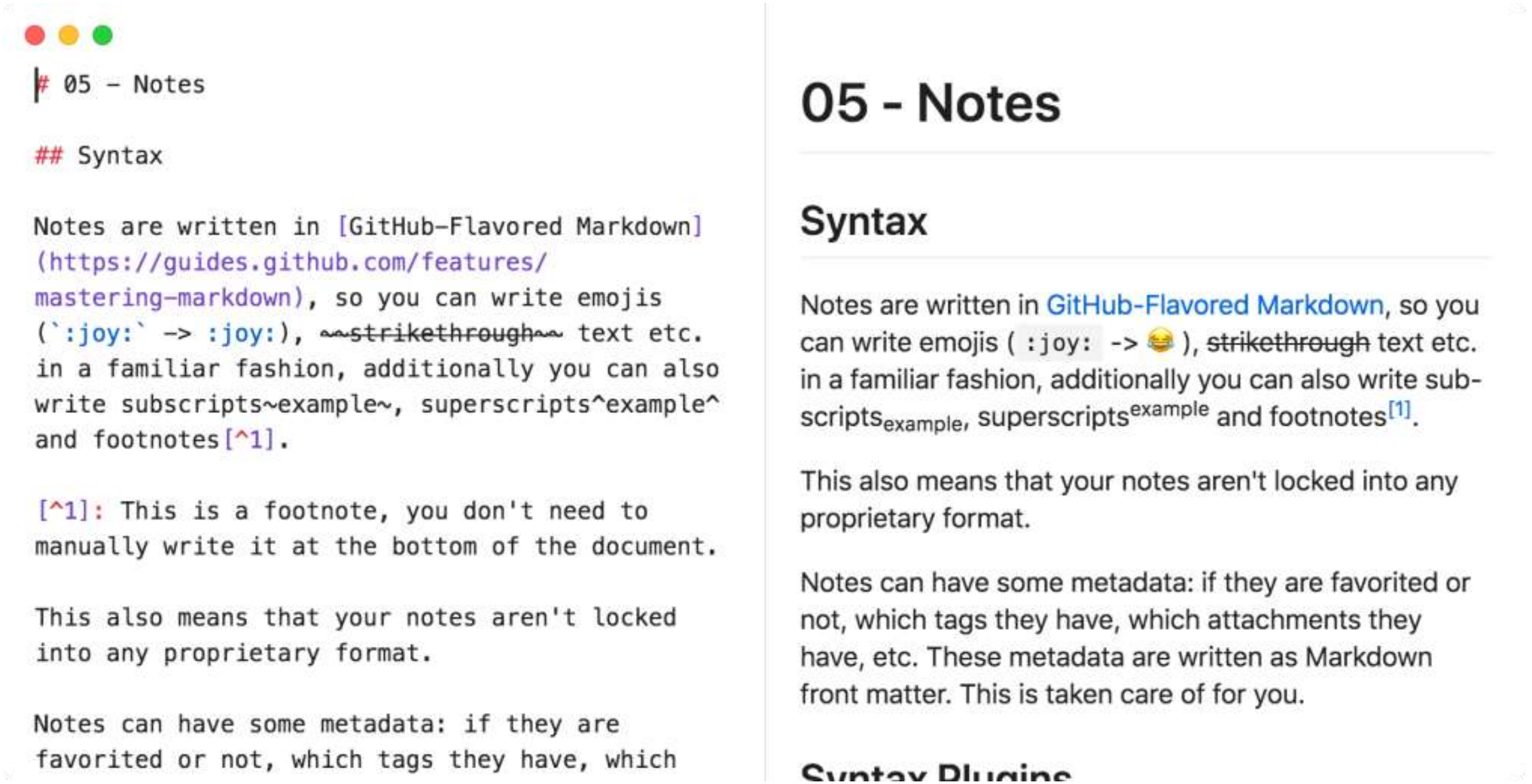
Contents

- General Information
- Contributing to CPython
- Using Python
- Build Instructions
  - Profile Guided Optimization
  - Link Time Optimization
- What's New
- Documentation
- Converting From Python 2.x to 3.x
- Testing
- Installing multiple versions
- Issue Tracker and Mailing List
- Proposals for enhancement
- Release Schedule
- Copyright and License Information

Markdown

- Plain-text format which can be rendered into HTML
- Extremely simple format to write and read, even unrendered in terminal
- Headings, text formatting (bold, italics), lists, links, images, tables, footnotes, formatted code snippets, …
- Searchable
- More forgivable than LaTeX. One mistake won't break your whole doc
- Several markdown "flavours" exist. GitHub markdown is the safe option
- Tips
  - Markdown can be "compiled" to PDF for distribution if needed
  - Can write HTML elements into markdown for more complex things
- reStructuredText (RST) for more technical documentation

Markdown



```
# 05 - Notes

## Syntax

Notes are written in [GitHub-Flavored Markdown]
(https://guides.github.com/features/
mastering-markdown), so you can write emojis
(`:joy:` -> :joy:), ~~strikethrough~~ text etc.
in a familiar fashion, additionally you can also
write subscripts~example~, superscripts^example^
and footnotes[^1].

[^1]: This is a footnote, you don't need to
manually write it at the bottom of the document.

This also means that your notes aren't locked
into any proprietary format.

Notes can have some metadata: if they are
favorited or not, which tags they have, which
```

05 - Notes

Syntax

Notes are written in GitHub-Flavored Markdown, so you can write emojis (`:joy:` -> 😂), ~~strikethrough~~ text etc. in a familiar fashion, additionally you can also write sub-scripts$_{example}$, superscripts^{example} and footnotes[1].

This also means that your notes aren't locked into any proprietary format.

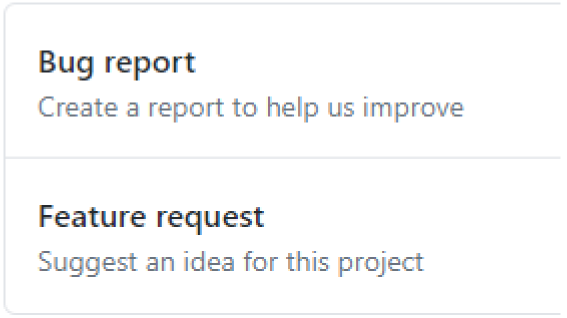
Notes can have some metadata: if they are favorited or not, which tags they have, which attachments they have, etc. These metadata are written as Markdown front matter. This is taken care of for you.

Syntax Plugins

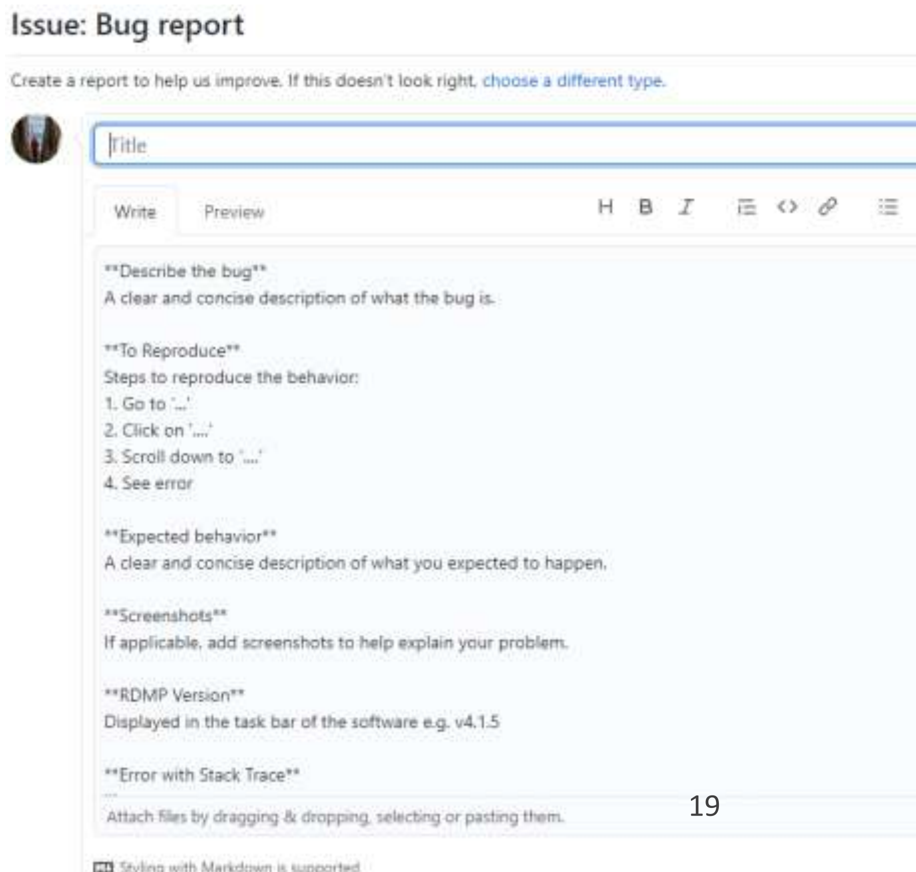
https://github.com/notable/notable

Issue tracker

- Main contact point for your project
  - Bug reports, feature requests, Q&A
- Problem: Users don’t provide detailed bug reports. Solution: Provide issue templates. (also for Pull Requests)
- Prefer text over screenshots (searchability)
- Separate discussions (forums) coming soon


Bug report
Create a report to help us improve

Feature request
Suggest an idea for this project


Issue: Bug report

Create a report to help us improve. If this doesn't look right, choose a different type.

```
**Describe the bug**
A clear and concise description of what the bug is.

**To Reproduce**
Steps to reproduce the behavior:
1. Go to '...'
2. Click on '....'
3. Scroll down to '....'
4. See error

**Expected behavior**
A clear and concise description of what you expected to happen.

**Screenshots**
If applicable, add screenshots to help explain your problem.

**RDMP Version**
Displayed in the task bar of the software e.g. v4.1.5

**Error with Stack Trace**
```

Attach files by dragging & dropping, selecting or pasting them.

Styling with Markdown is supported

Documentation

- Write some.
- Basic build & usage instructions sufficient. Bonus points for FAQ, common issues etc.
- Many different levels
  - Keep in plain text (i.e. markdown) and include in repo
  - Wiki to arrange larger collections of documents
  - Documentation generation & hosting, i.e. GH Pages, or Sphinx + ReadTheDocs
    - Pages easy to configure with pre-built templates. Published to “yourproj.github.io”
    - Generate simple static website from markdown or RST
    - “Continuous Documentation” – docs are built, tested, and published with every commit
    - See https://cirrus.readthedocs.io/en/master

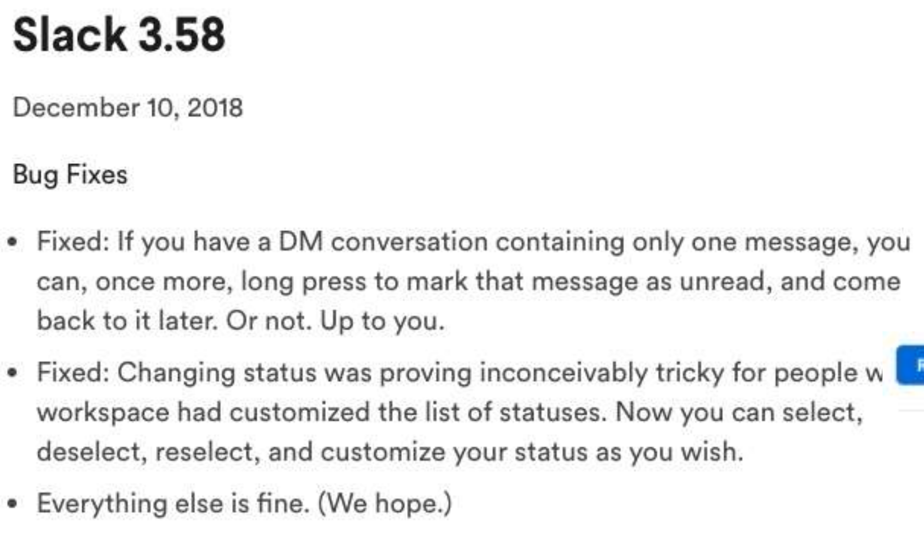
Versioning & Releasing

- Pick a versioning scheme. SemVer most common
  - Major.Minor.Patch
  - https://semver.org/

- Keep a CHANGELOG https://keepachangelog.com
  - Reduces need for “git archaeology”

- Use Git tags and GitHub releases
  - Upload distributable copies of your code in many formats
  - Don’t require that the user has root access to install

Versioning & Releasing

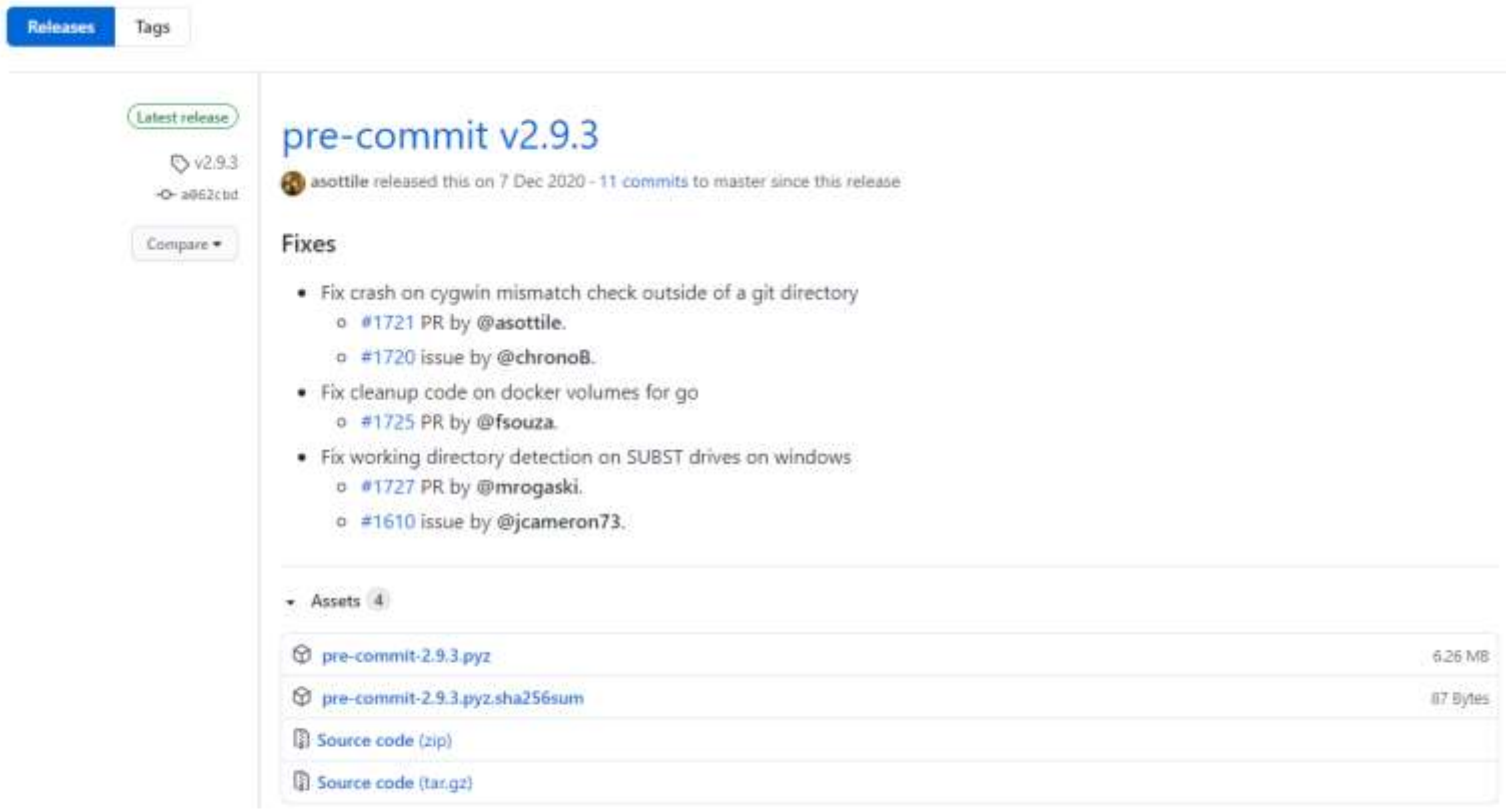


Slack 3.58

December 10, 2018

Bug Fixes

- Fixed: If you have a DM conversation containing only one message, you can, once more, long press to mark that message as unread, and come back to it later. Or not. Up to you.
- Fixed: Changing status was proving inconceivably tricky for people w
workspace had customized the list of statuses. Now you can select, deselect, reselect, and customize your status as you wish.
- Everything else is fine. (We hope.)



pre-commit v2.9.3

asottile released this on 7 Dec 2020 - 11 commits to master since this release

Fixes

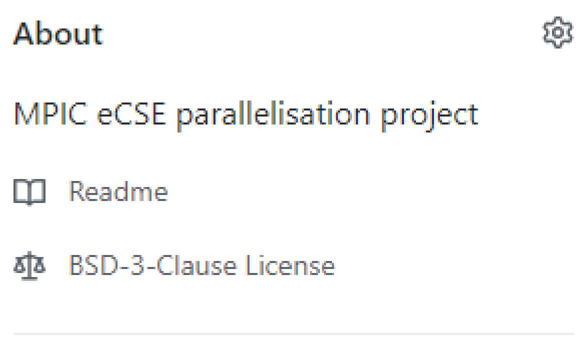
- Fix crash on cygwin mismatch check outside of a git directory
  - #1721 PR by @asottile.
  - #1720 issue by @chronoB.
- Fix cleanup code on docker volumes for go
  - #1725 PR by @fsouza.
- Fix working directory detection on SUBST drives on windows
  - #1727 PR by @mrogaski.
  - #1610 issue by @jcameron73.

Assets 4

pre-commit-2.9.3.pyz	6.26 MB
pre-commit-2.9.3.pyz.sha256sum	87 Bytes
Source code (zip)	
Source code (tar.gz)	

Licences

- Add one
- Often not considered but important
  - States how others can safely use, share, modify your work
  - Potential users may not be permitted to use software that doesn't display a licence
- https://choosealicense.com/ can help you pick one depending on situation
- GitHub automatically detects and displays LICENSE files
- Make sure you are permitted to share
  - Ask if unsure



About

MPIC eCSE parallelisation project

Readme

BSD-3-Clause License

Discussion

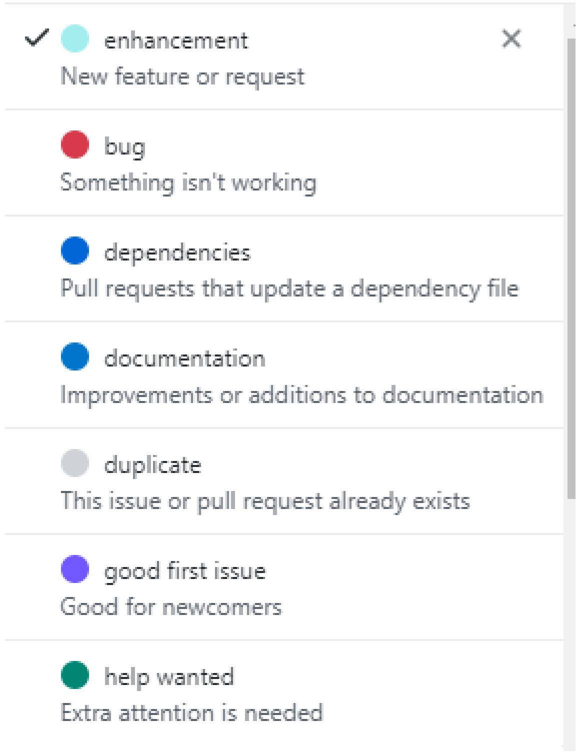
Effectively managing collaboration

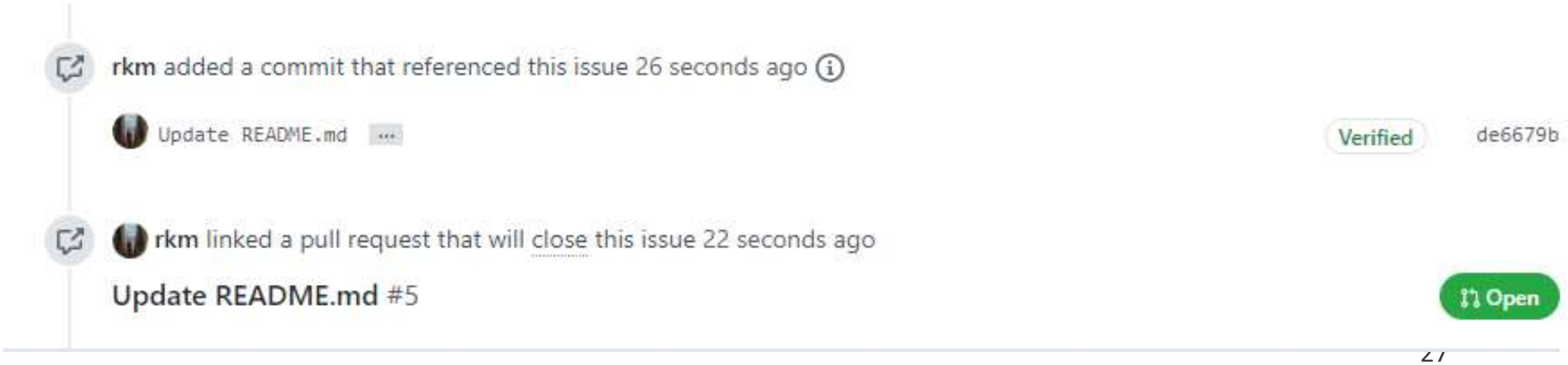
Effectively managing collaboration

- **Goals**
  - Use built-in GitHub tools to ease effort of managing your project
  - Ensure contributions can be made with little friction

Managing issues

- Main contact point for your project
  - Bug reports, feature requests, Q&A
  - Closed/resolved issues are useful documentation to refer to
- Issue assignment
- Labels to classify and help future searches
- Reference issues in commits / PRs, and automatically close

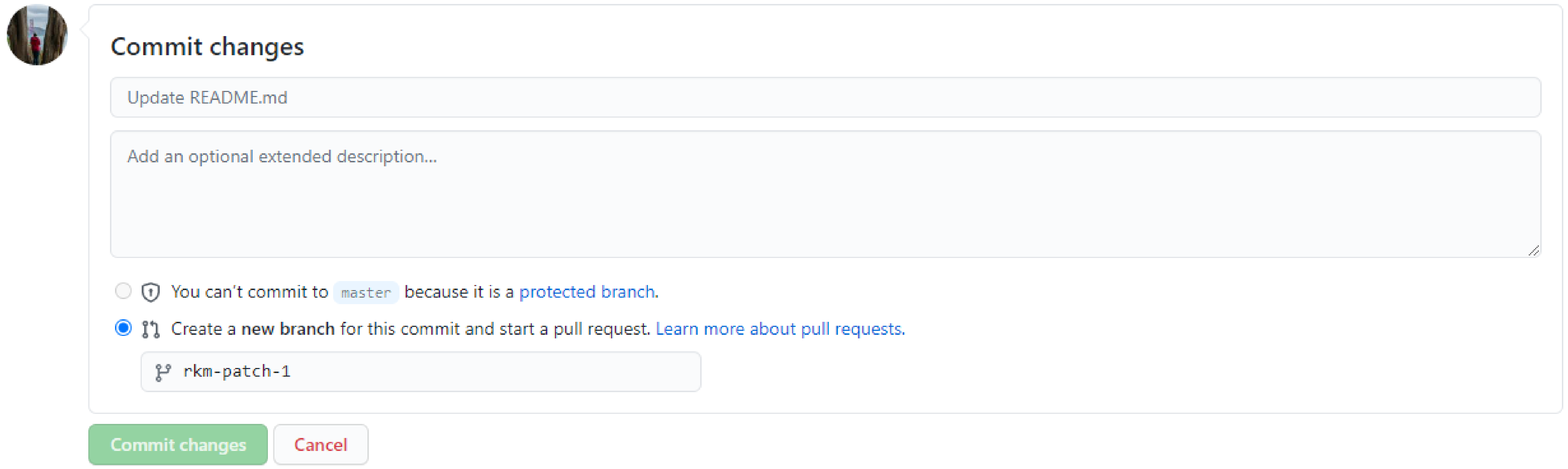

✓ enhancement
New feature or request
bug
Something isn't working
dependencies
Pull requests that update a dependency file
documentation
Improvements or additions to documentation
duplicate
This issue or pull request already exists
good first issue
Good for newcomers
help wanted
Extra attention is needed


rkm added a commit that referenced this issue 26 seconds ago
Update README.md ... Verified de6679b
rkm linked a pull request that will close this issue 22 seconds ago
Update README.md #5

Git workflow

- How do users contribute to your project?
- Git branching model. Single main branch, with “feature” branches which are merged back to main via a Pull Request (PR)
- For external contributors -> fork and merge
- Many workflows possible – discuss with your team, and document!
- Tips
  - Can commit directly in GitHub UI, and automatically branch and create PR if needed. Great for small documentation fixes
  - Can configure branch rules to force all changes to be via a reviewed PR

- More on PRs soon, but first…



Commit changes

Update README.md

Add an optional extended description...

You can't commit to `master` because it is a protected branch.

Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

[✓ Create a new branch for this commit and start a pull request.]

`rkm-patch-1`

Tests

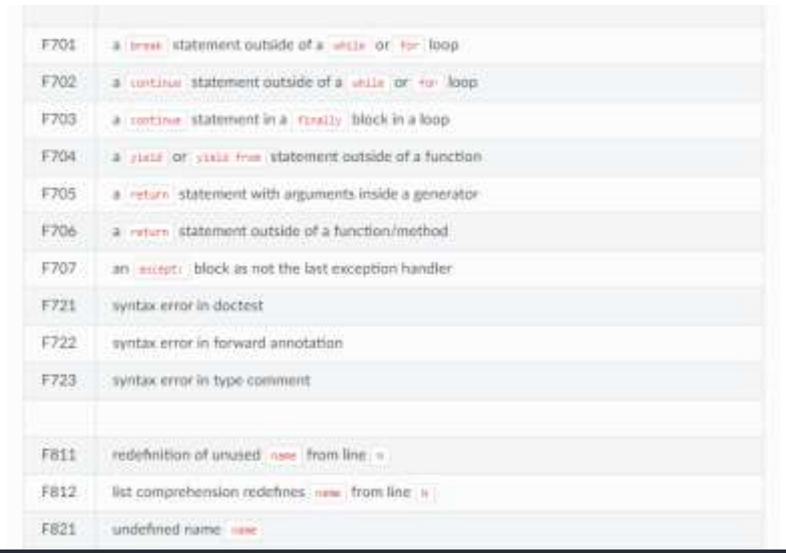
- Write some.

- Document how users should run tests when working with the code
- Will assume you already have tests for following sections

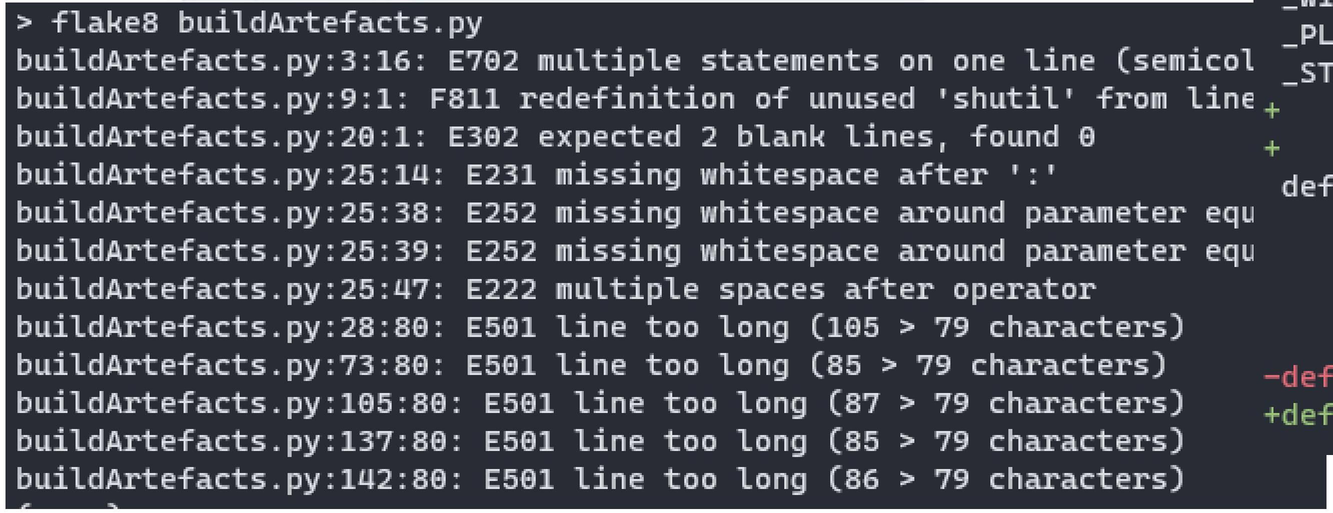
Linters & Formatters

- Linters
  - Tools to scan code (statically & dynamically) and report on common mistakes, warnings
  - Many to choose per language (python: mypy, flake8, pylint)
- Formatters
  - Tools to warn or re-format code to ensure a particular agreed on code “style”
  - Some languages have recommended styles and even built-in formatters (gofmt, rustfmt)
- Recommendation: use both
  - Act as “gates” to ensure quality
  - Many benefits e.g. ease of code review & common style
  - Drawbacks: misconfigured or excessive tools like this may actually *increase* friction for contributors
- Tips
  - Use pre-commit to help users manage and run your linters & formatters. Ensures they are run (and pass!) before commit is made
  - Run them in your CI as well (pre-commit-ci)

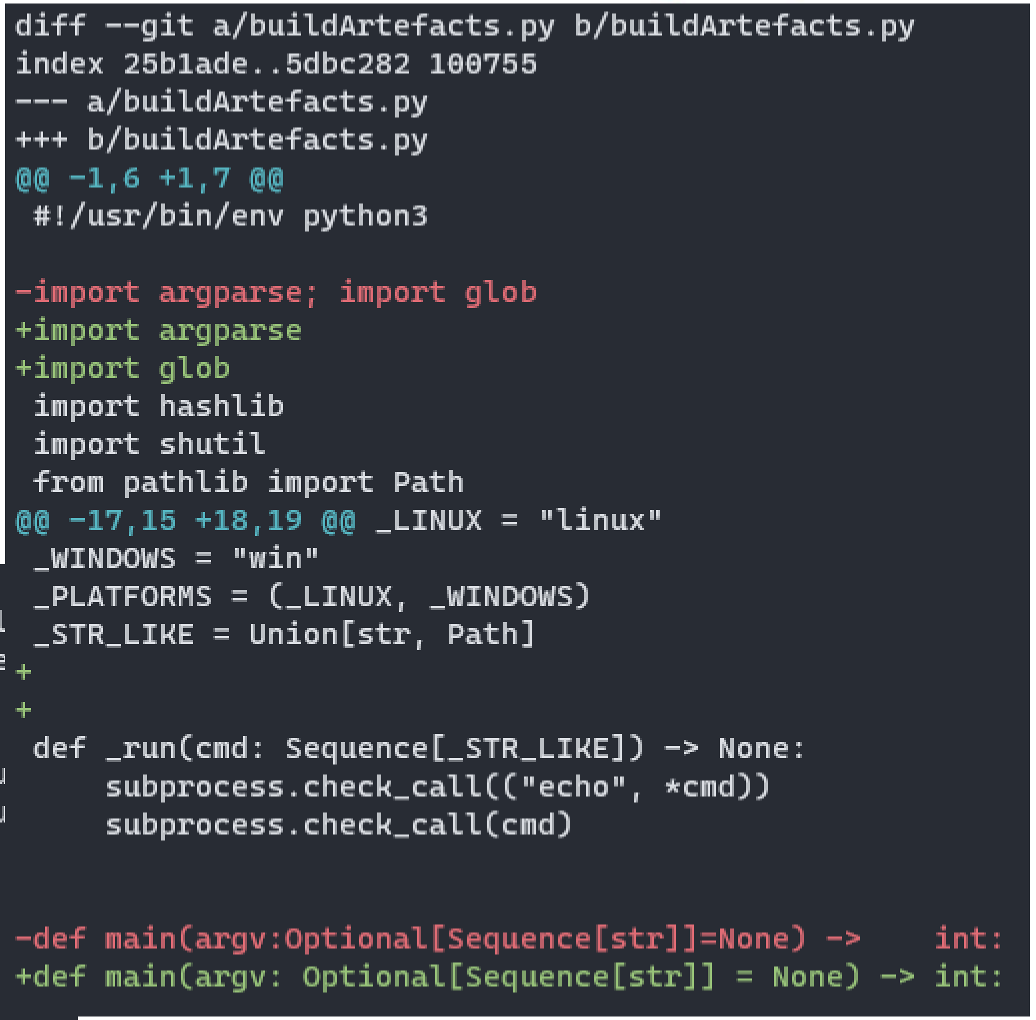
Linters & Formatters



F701	a `break` statement outside of a `while` or `for` loop
F702	a `continue` statement outside of a `while` or `for` loop
F703	a `continue` statement in a `finally` block in a loop
F704	a `yield` or `yield from` statement outside of a function
F705	a `return` statement with arguments inside a generator
F706	a `return` statement outside of a function/method
F707	an `except:` block as not the last exception handler
F721	syntax error in doctest
F722	syntax error in forward annotation
F723	syntax error in type comment
F811	redefinition of unused `name` from line `N`
F812	list comprehension redefines `name` from line `N`
F821	undefined name `name`



```
> flake8 buildArtefacts.py
buildArtefacts.py:3:16: E702 multiple statements on one line (semicol
buildArtefacts.py:9:1: F811 redefinition of unused 'shutil' from line
buildArtefacts.py:20:1: E302 expected 2 blank lines, found 0
buildArtefacts.py:25:14: E231 missing whitespace after ':'
buildArtefacts.py:25:38: E252 missing whitespace around parameter equ
buildArtefacts.py:25:39: E252 missing whitespace around parameter equ
buildArtefacts.py:25:47: E222 multiple spaces after operator
buildArtefacts.py:28:80: E501 line too long (105 > 79 characters)
buildArtefacts.py:73:80: E501 line too long (85 > 79 characters)
buildArtefacts.py:105:80: E501 line too long (87 > 79 characters)
buildArtefacts.py:137:80: E501 line too long (85 > 79 characters)
buildArtefacts.py:142:80: E501 line too long (86 > 79 characters)
```



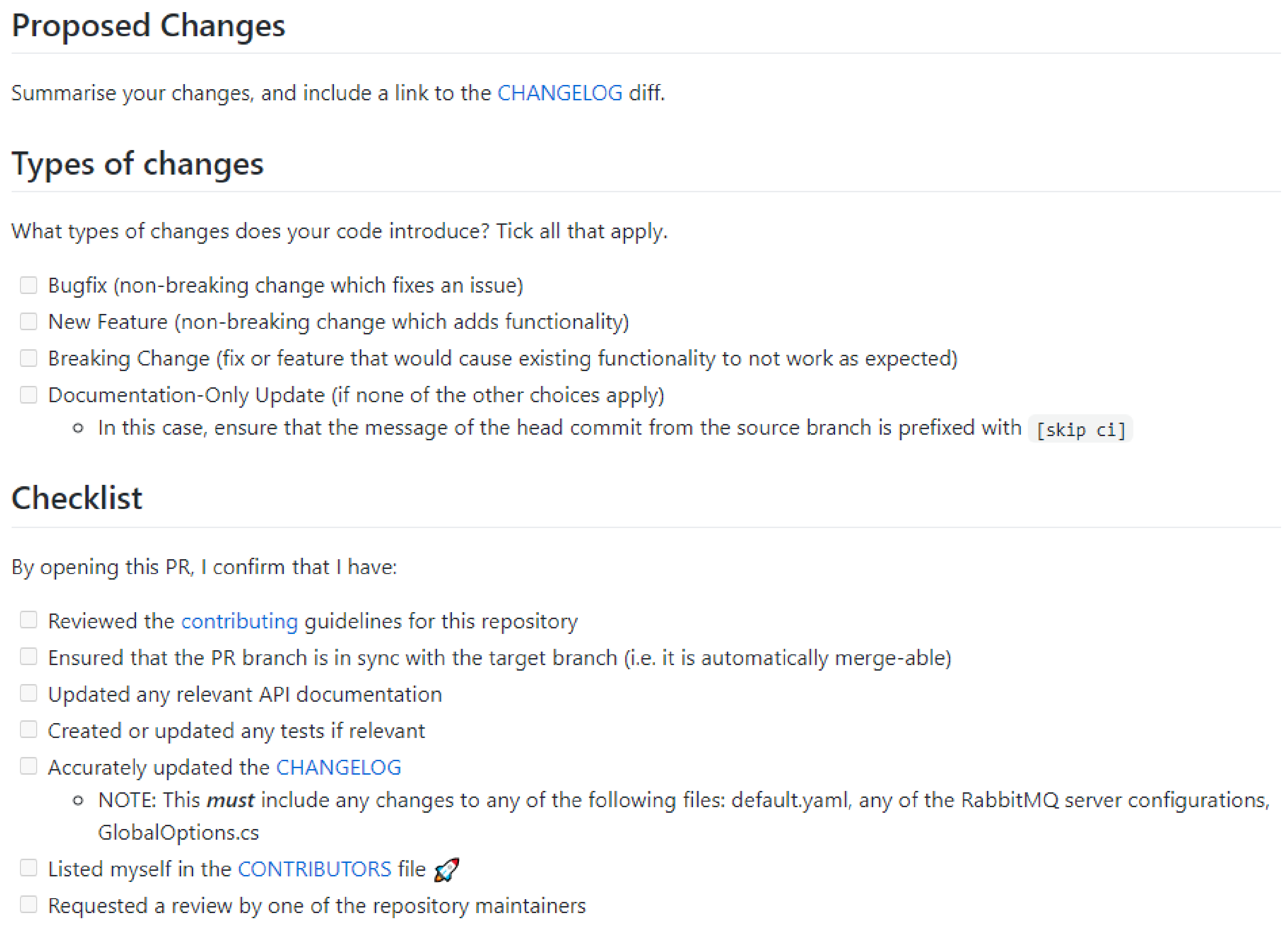
```
diff --git a/buildArtefacts.py b/buildArtefacts.py
index 25b1ade..5dbc282 100755
--- a/buildArtefacts.py
+++ b/buildArtefacts.py
@@ -1,6 +1,7 @@
 #!/usr/bin/env python3

-import argparse; import glob
+import argparse
+import glob
 import hashlib
 import shutil
 from pathlib import Path
@@ -17,15 +18,19 @@ _LINUX = "linux"
 _WINDOWS = "win"
 _PLATFORMS = (_LINUX, _WINDOWS)
 _STR_LIKE = Union[str, Path]
+
+
 def _run(cmd: Sequence[_STR_LIKE]) -> None:
     subprocess.check_call(("echo", *cmd))
     subprocess.check_call(cmd)


-def main(argv:Optional[Sequence[str]]=None) ->    int:
+def main(argv: Optional[Sequence[str]] = None) -> int:
```

Continuous Integration (CI)

- In short: automatically runs your tests on every commit to ensure success
- Aim is to uncover bugs at any point before release / production – “push left”
- Free and built-into GitHub these days – GitHub Actions
- Acts as another “gate” to ensure high-quality changes
- Not just tests. CI can run linters, formatters, code analysis tools, workflow steps, automatic publishing of releases…

Pull Requests

- How do changes make it into your project?
  - Contributors make changes, then open Pull Request
  - CI runs your tests, linters, formatters etc.
  - Code review!
  - Accept and merge, or feedback and request changes
- Also supports templates (have they created tests, updated documentation, …)
- Code review
  - Dependant on project type. Useful to build knowledge in team
  - Do: be fair and balanced
  - Don’t: Nit-pick minor issues (linters/formatters should remove these)s
- Tips
  - Use CI and automated checks to ease the PR process and reduce manual effort involved



Proposed Changes

Summarise your changes, and include a link to the CHANGELOG diff.

Types of changes

What types of changes does your code introduce? Tick all that apply.

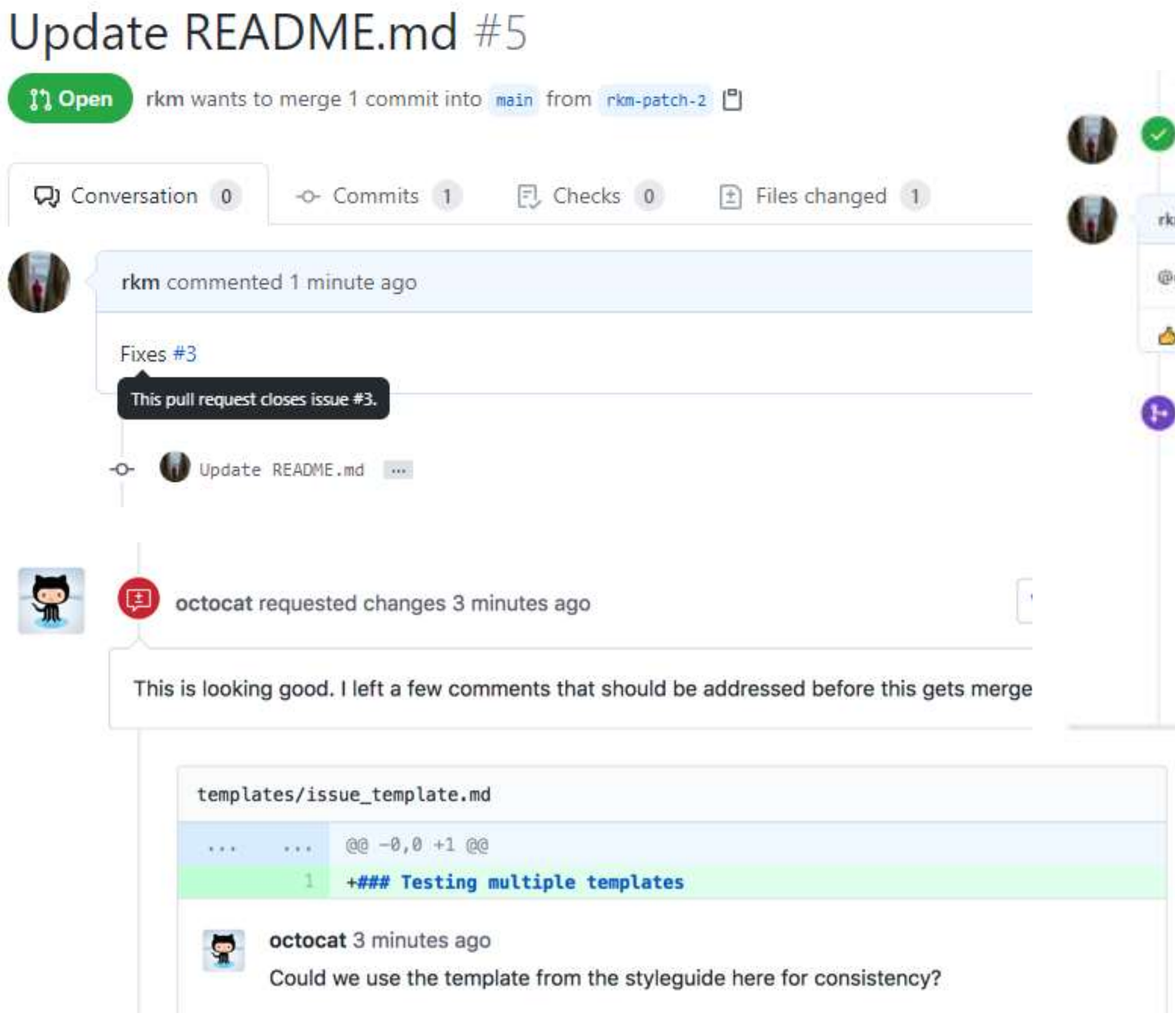
- [ ] Bugfix (non-breaking change which fixes an issue)
- [ ] New Feature (non-breaking change which adds functionality)
- [ ] Breaking Change (fix or feature that would cause existing functionality to not work as expected)
- [ ] Documentation-Only Update (if none of the other choices apply)
  - In this case, ensure that the message of the head commit from the source branch is prefixed with `[skip ci]`

Checklist

By opening this PR, I confirm that I have:

- [ ] Reviewed the contributing guidelines for this repository
- [ ] Ensured that the PR branch is in sync with the target branch (i.e. it is automatically merge-able)
- [ ] Updated any relevant API documentation
- [ ] Created or updated any tests if relevant
- [ ] Accurately updated the CHANGELOG
  - NOTE: This ***must*** include any changes to any of the following files: default.yaml, any of the RabbitMQ server configurations, GlobalOptions.cs
- [ ] Listed myself in the CONTRIBUTORS file 🚀
- [ ] Requested a review by one of the repository maintainers

Pull Requests



Update README.md #5

Open rkm wants to merge 1 commit into `main` from `rkm-patch-2`

Conversation 0 · Commits 1 · Checks 0 · Files changed 1

rkm commented 1 minute ago

Fixes #3

This pull request closes issue #3.

`Update README.md` ...

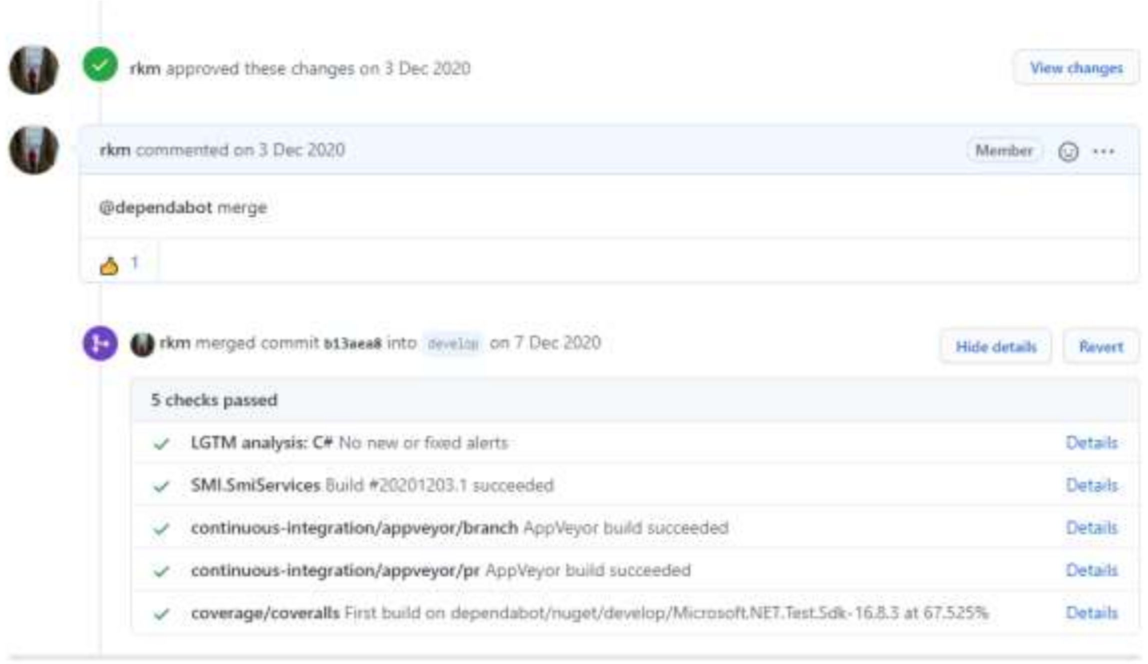
octocat requested changes 3 minutes ago

This is looking good. I left a few comments that should be addressed before this gets merge

```
templates/issue_template.md
...     ...   @@ -0,0 +1 @@
          1   +### Testing multiple templates
```

octocat 3 minutes ago

Could we use the template from the styleguide here for consistency?



rkm approved these changes on 3 Dec 2020 — View changes

rkm commented on 3 Dec 2020 — Member

@dependabot merge

👍 1

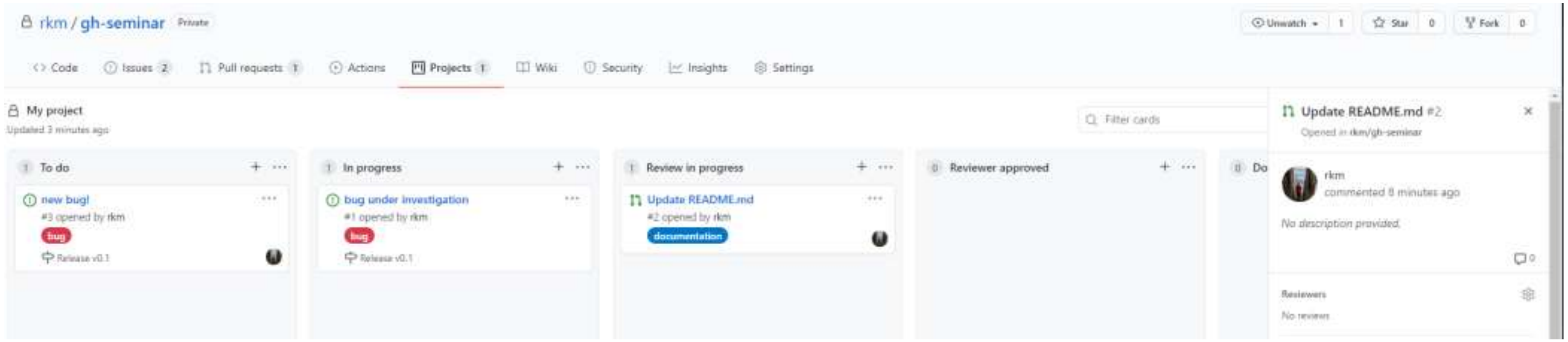
rkm merged commit **b13aea8** into `develop` on 7 Dec 2020 — Hide details · Revert

5 checks passed

✓	**LGTM analysis: C#** No new or fixed alerts	Details
✓	**SMI.SmiServices** Build #20201203.1 succeeded	Details
✓	**continuous-integration/appveyor/branch** AppVeyor build succeeded	Details
✓	**continuous-integration/appveyor/pr** AppVeyor build succeeded	Details
✓	**coverage/coveralls** First build on dependabot/nuget/develop/Microsoft.NET.Test.Sdk-16.8.3 at 67.525%	Details

Project Boards

- Kanban/Trello style boards to visualise current project tasks and workflow
- Can be automated to automatically update when issues/PRs resolved
- May encourage contributions



Interacting with Contributors

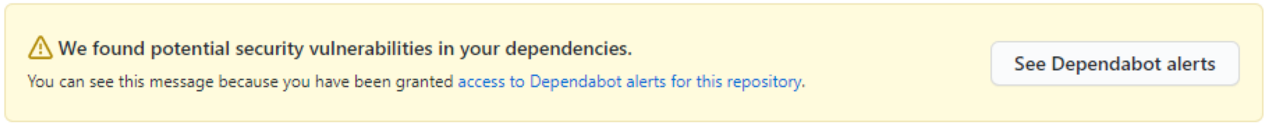
- Tricky topic, but worth mentioning
- Tips
  - Picture that you're actually speaking to whoever you're typing at (might be difficult to recall what that's like…). Comments are public
  - Treat bug reports and pull requests as opportunities to improve your project
  - Bug reports are not personal attacks!
- Conversely
  - You have no obligation to add a particular feature in your project (which you will then have the burden to support forever)
  - Easy to ignore issues and PRs that don't follow your templates!
- Consider that many open source contributors are volunteering their own time

Discussion

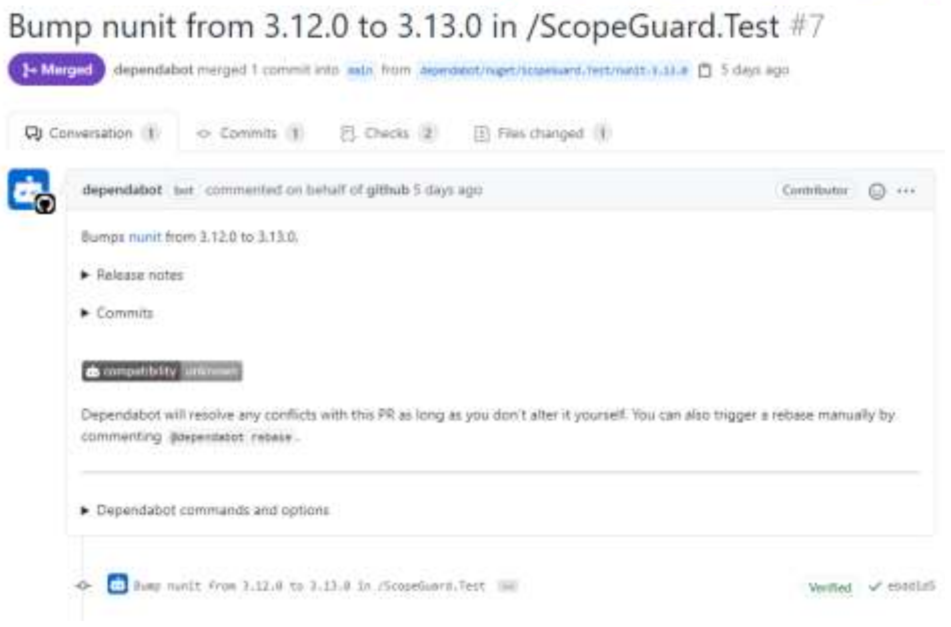
Misc. Features

- Wikis are backed by a separate git repo – can clone and update locally
- GitHub CLI and Desktop
- Dependabot for automated dependency updates
  - Now built-into GitHub
- Marketplace of pre-built tools, integrations
- Security / automated code scanning tools
  - LGTM (looks good to me!)
- Manage permissions across your repositories / teams
- Dark mode…

Misc. Features


We found potential security vulnerabilities in your dependencies.

You can see this message because you have been granted access to Dependabot alerts for this repository.


Bump nunit from 3.12.0 to 3.13.0 in /ScopeGuard.Test #7

Merged dependabot merged 1 commit into main from dependabot/nuget/ScopeGuard.Test/nunit-3.13.0 5 days ago

dependabot bot commented on behalf of github 5 days ago

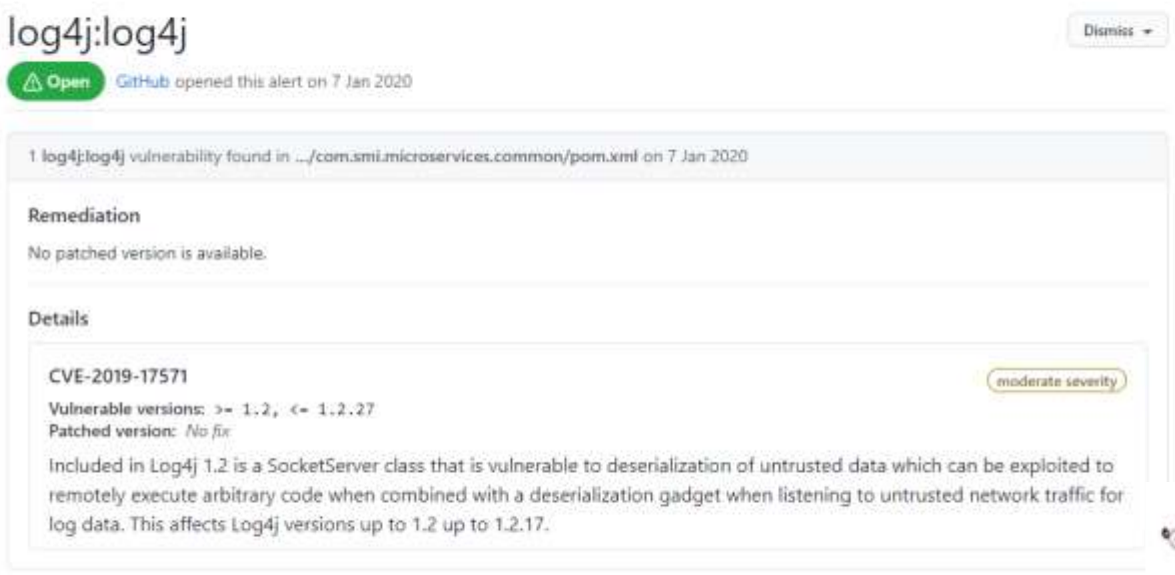
Bumps nunit from 3.12.0 to 3.13.0.

- Release notes
- Commits

Dependabot will resolve any conflicts with this PR as long as you don't alter it yourself. You can also trigger a rebase manually by commenting `@dependabot rebase`.

- Dependabot commands and options

Bump nunit from 3.12.0 to 3.13.0 in /ScopeGuard.Test


log4j:log4j

Open GitHub opened this alert on 7 Jan 2020

1 log4j:log4j vulnerability found in .../com.smi.microservices.common/pom.xml on 7 Jan 2020

Remediation

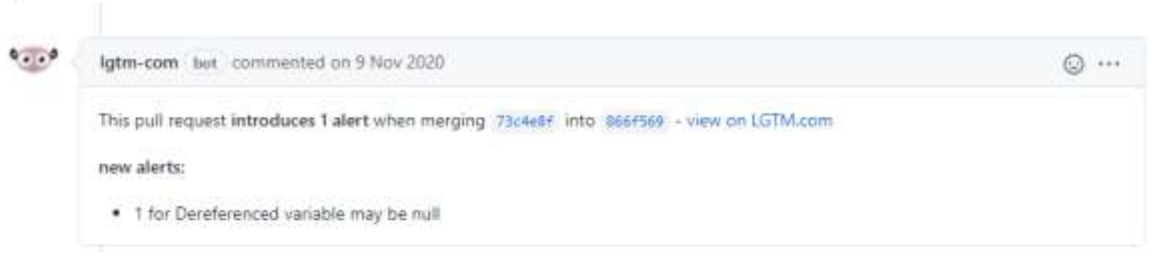
No patched version is available.

Details

CVE-2019-17571

Vulnerable versions: `>= 1.2, <= 1.2.27`
Patched version: *No fix*

Included in Log4j 1.2 is a SocketServer class that is vulnerable to deserialization of untrusted data which can be exploited to remotely execute arbitrary code when combined with a deserialization gadget when listening to untrusted network traffic for log data. This affects Log4j versions up to 1.2 up to 1.2.17.


lgtm-com bot commented on 9 Nov 2020

This pull request **introduces 1 alert** when merging `73c4e8f` into `866f569` - view on LGTM.com

new alerts:

- 1 for Dereferenced variable may be null

Summary

- Lots of “extra” effort involved in managing software projects. Use GitHub workflows and automated tools to ease this
  - Ensure your project is easy to find, use, and learn. Provide a means for users to interact and provide feedback
  - Ensure your team, the primary users of your repository, have low friction in getting their tasks done
  - Ensure contributions can be made with little friction

- Key takeaways
  - Learn to write and use Markdown
  - Use Issue and PR templates to reduce review effort
  - Use linters, formatters, and CI to ensure quality control

References

1. https://expandedramblings.com/index.php/gitlab-statistics-and-facts/
2. https://expandedramblings.com/index.php/github-statistics/