Pull Requests: Do’s & Don’ts

Arie Bregman
Senior Software Engineer @RedHat

Agenda

- Terminology
- Do’s and Don’ts
- Projects guidelines and templates
- Open Discussion
- Questions

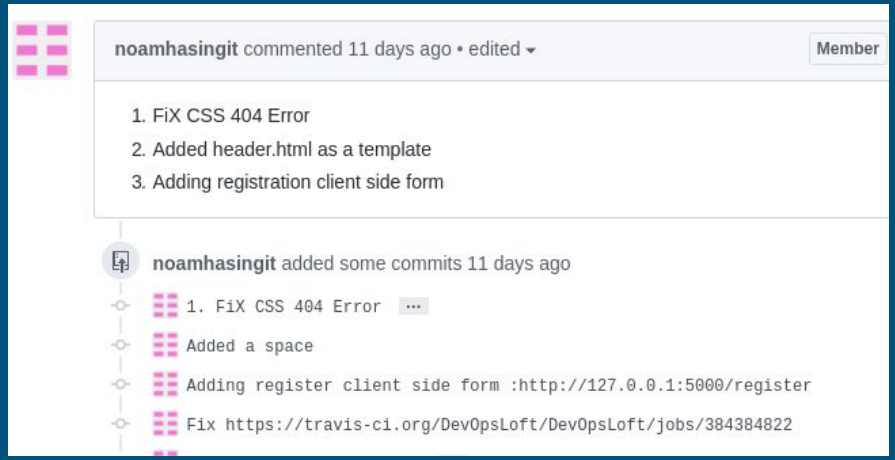
Commit

- A record of your changes
- Adding your changes to the tree of the repository

```
commit a2fc2b4ef9ec5cd27f9e30dfced03acab9bb8ff0
Merge: ad9c0fe 656c594
Author: Liora Milbaum <lioramilbaum@users.noreply.github.com>
Date:   Tue Apr 3 17:50:45 2018 +0300

    Merge pull request #33 from bregman-arie/add_travis

    Add travis configuration file

commit ad9c0fea523679bd50236c8e98ffde3cb5fcd8ca
Merge: 184913e cafa6a4
Author: Liora Milbaum <lioramilbaum@users.noreply.github.com>
Date:   Tue Apr 3 17:39:01 2018 +0300

    Merge pull request #34 from bregman-arie/add_gitignore

    Add gitignore file

commit cafa6a40dfb935a478361e5c85142b36376c53eb (origin/add_gitignore)
Author: Arie Bregman <abregman@redhat.com>
Date:   Wed Mar 28 01:45:25 2018 +0300

    Modify gitignore

    This will make sure we don't add "junk" files
    when doing commits.
```

Pull Request

- A request to submit code
- **Pull request can include several commits**
- Different systems use different approach for code reviews
  - Gerrit
- Why knowing how to submit pull requests is important?



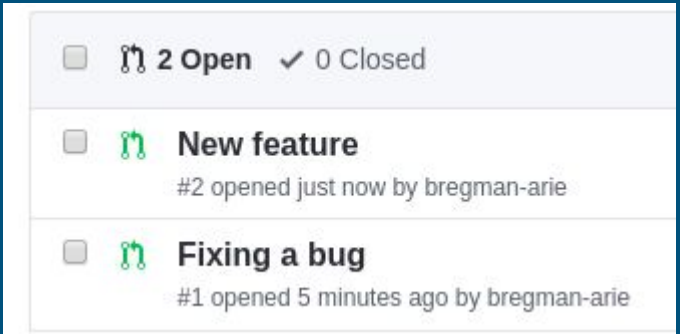
Reviewer

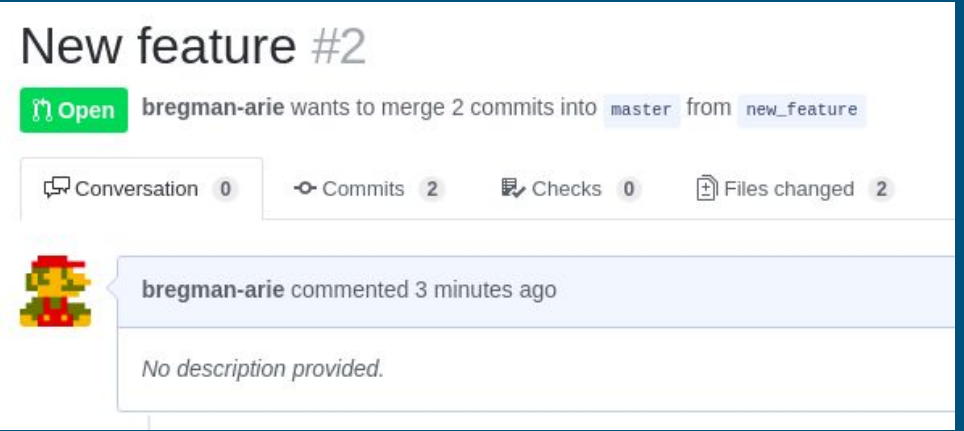
- A person who goes over your changes, comment about them and vote accordingly
- Reviewer can be anyone
  - Technical or Non-technical
  - Someone from your team or external
  - Junior or Senior

CI

- What is CI? We need a separate session for this
- Trying to describe the most common workflow would be:
  - An automated process for easily and safely testing your pull request with latest revision of your project
- CI can includes stages like compiling, testing, installing and configuring the project
- Popular CI systems
  - Jenkins
  - Travis

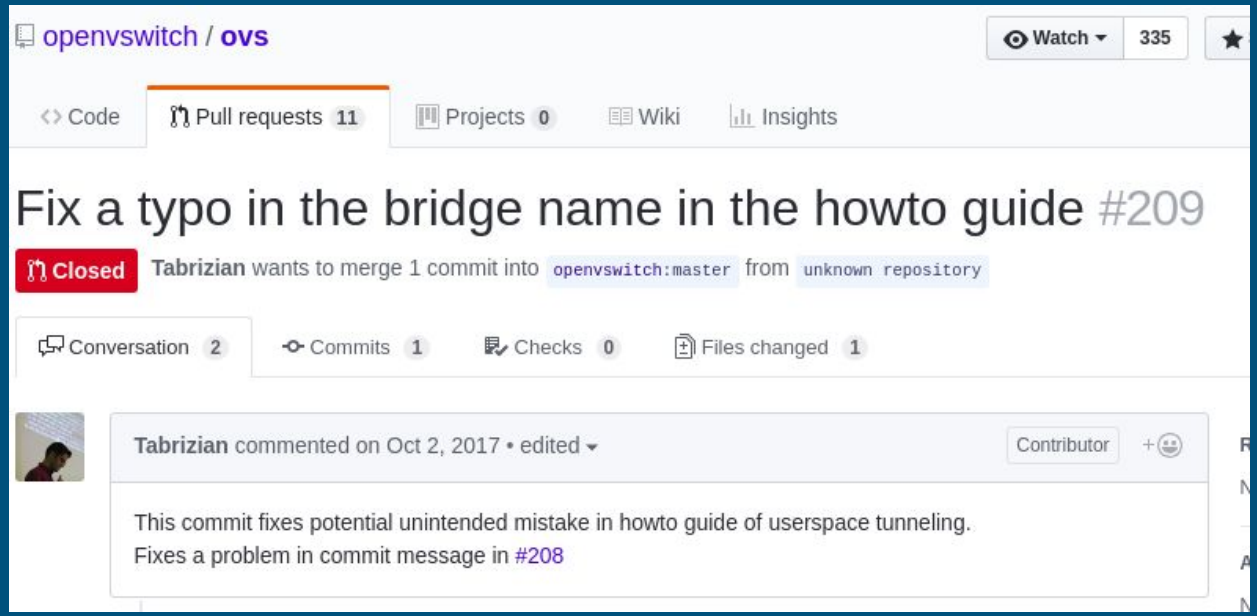
Don’t: Meaningless commit messages





- Why this is bad?
  - Reviewer doesn’t know what your code suppose to do
  - In the future, people will not be able to understand why a certain change was done when running for example ‘git log’

Do: Meaningful commit messages



ansible / **ansible**

Watch 1,914 Sta

Code | Issues 3,597 | Pull requests 1,485 | Projects 22 | Insights

openstack: Correct fail_json call in module_utils #41064

Merged **mkrizek** merged 1 commit into `ansible:devel` from `Akasurde:openstack_fail_json` a day ago

Conversation 3 | Commits 1 | Checks 0 | Files changed 1

Akasurde commented 2 days ago — Member

SUMMARY

Fixes: #40925

Signed-off-by: Abhijeet Kasurde akasurde@redhat.com

ISSUE TYPE

- Bugfix Pull Request

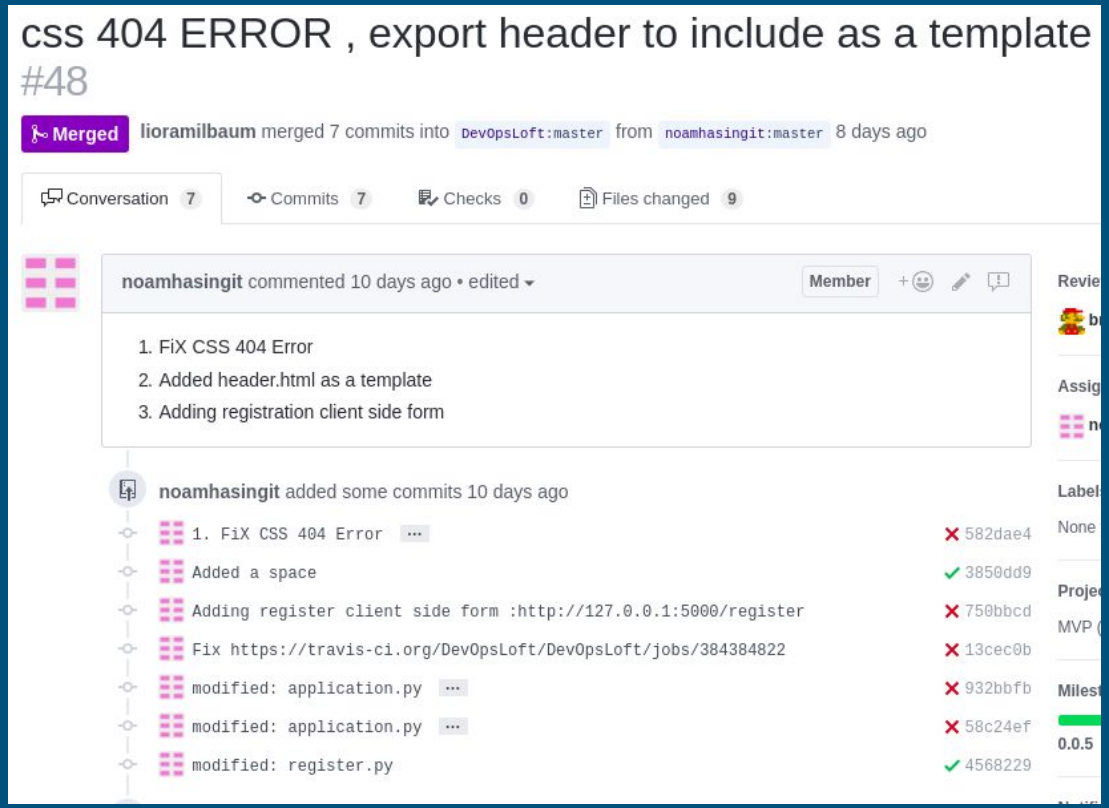
COMPONENT NAME

lib/ansible/module_utils/openstack.py

ANSIBLE VERSION

```
2.7devel
```

Don’t: include multiple commits in a single PR



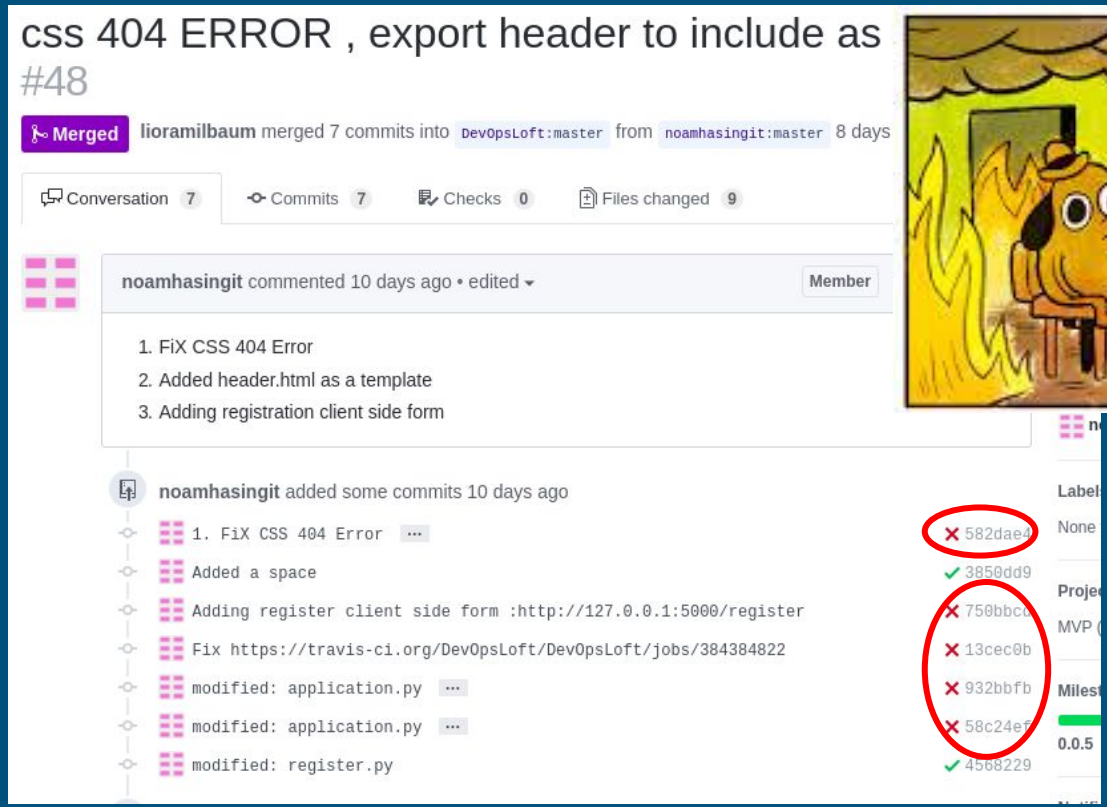
Don’t: include multiple commits in a single PR

- Why this is bad?
  - You are enforcing one reviewer to review several changes
    - Can take a lot of time, instead of splitting the review effort
    - Changes can be unrelated and then there is really no reason to send them together

Do: Limit number of commits per PR

- One is usually enough
- There is sometimes a reason for including two
- More than two? Ask yourself if it’s necessary

Don’t: Ignore CI



Don’t: Ignore CI

- Why this is bad?
  - You don’t want to merge something that breaks the project
  - Some reviewers will not review your PR until you fix the issues

Do: Act based on CI results

- CI usually has purpose :)
- Your PR doesn’t pass CI? FIX IT
- Don’t expect the reviewer to provide feedback/vote before you fixed everything
- There are some good lessons in understand how CI works and why it fails
  - Code styling and conventions
  - Learning more on the tests side (= the dark side for many folks ;])

Don’t: Commit messages in the code itself

```
@contextlib.contextmanager
def defer_apply(self):
    """Defer apply context."""
    self.filter_defer_apply_on()
    try:
        yield
    finally:
        # Yesterday, while eating my breakfast, the application
        # failed for me since it was missing the armageddon fix function.
        # Now I'm adding the fix so others won't fail on this as well
        # The function itself running several tools like "fix-all" and
        # "this-will-never fail".
        # In the future, we should apply this also in files like "a.py"
        # and "b.py". Let's also discuss if we want to have it in "c.py"
        # as well.
        self.armageddon_fix()
        self.filter_defer_apply_off()
```

Don’t: Commit messages in the code itself

- Why this is bad?
  - In my opinion comments in code should explain the code itself and commit messages should explain why a PR was submitted and what its goal
  - Affects readability of the code

- Note: this message is not even a valid commit message :)

Do: Distinguish between commit message and code comments

- Find some logic that works for you to make code readable and yet, easy for the reviewer to understand your code

Summary

- Commit messages
  - Structure is important
  - To the point
- Submitter-Reviewer relationship
  - Put yourself in the reviewer shoes
  - Listen to what he/she have to say. No matter how experienced she/he is.
  - It includes some of the dynamics you have when having a technical discussion with someone face-to-face
- CI
  - Git hooks is a great way to save some time
  - Act based on the results
- Limit number of commits per PR

Project’s guidelines & templates

- Some projects provide their own insights, guidelines and/or templates for pull requests & commits
  - OpenStack Git commit best practices
  - Ansible template
- GitHub blog post
- When joining a big project or community, check if they have some predefined guidelines

Open Discussion

- Do you have your own unwritten rules you follow?
- Do you agree with everything I mentioned?

Questions?