Git Etiquette: Best practices

or Mind your Git Manners

Irina Gulina
Senior Software Quality Engineer

Tomas Tomecek
Principal Software Engineer

History, why we are here?

- Red Hat does lots of projects with Universities
- Main feedback on students:

"We often see students don't have the experience of working in Git in team/collaborative projects"

Workflow in a personal GIT repo

not equal to

Workflow in a team GIT repo

Agenda

- Commit
- Push
- PR/MR submitting and review
- Quiz

Happy Birthday, Git!

Source: https://octodex.github.com/

Git Commit

Commit content

- Do: **One commit = One logical change**

1e4faa0	Fix login timeout BZ
2r5asy8	Add foo login step

- Don't: **Two and more changes in one commit**

1e4faa0	Fix login timeout BZ, add foo login step

Commit content

- Separate whitespace changes from code changes, especially unrelated.
    - Mixing those is a great way to introduce a bug and
    - Complicates code review

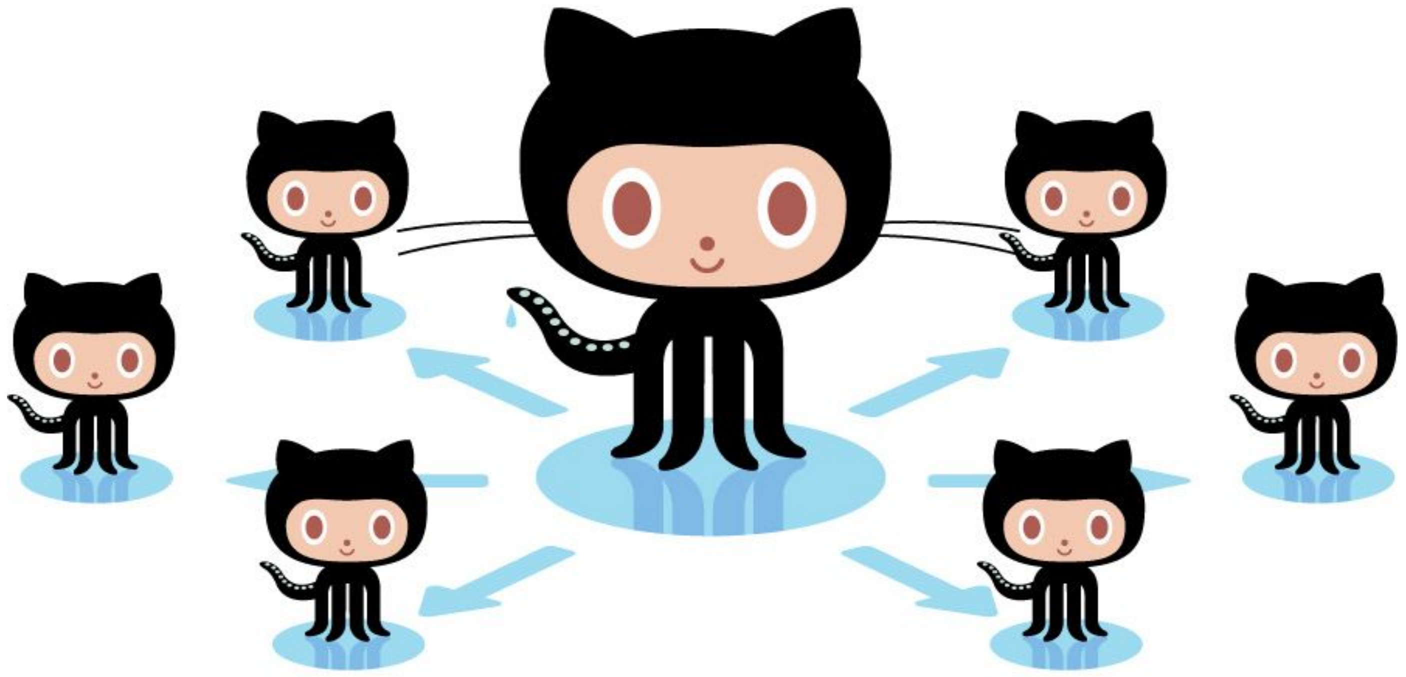
What is a ‘bad’ commit message?

dcc2d35	address comments	<- what comments?
b7aac30	fix issue #123	<- of what project?
0b7a4e4	various docs fixes	<- what docs? why?
1e4faa0	ui bug fix	<- what was the bug?
fc3d081	readme update	<- why?
d21660dc	ToDo	<- 😩😩😩😩
0b7a4e4	Mix fixes and cleanups	<- 🙈🙉🙊
5h3d28g	refactoring	<- 😭

Uninformative, look-elsewhere commit messages (titles)

Poor quality code can be refactored. A terrible commit message lasts **forever**.

For whom do you write commit messages?



Why should I write ‘good’ commit messages?

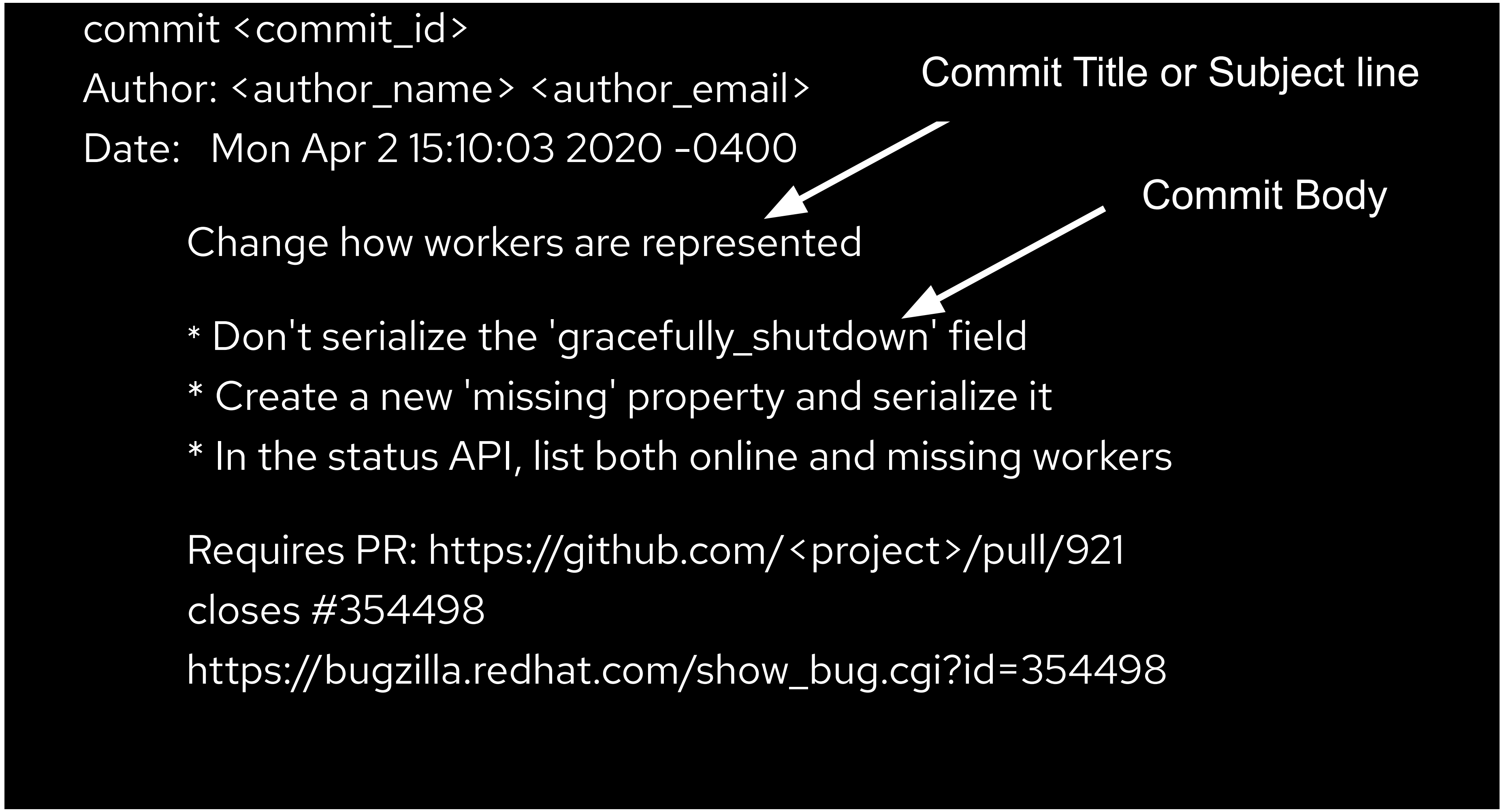
- To help to understand the code change
  - What has been changed?
  - Why is that change necessary?
- To speed up the reviewing process
- To help to locate a bug
- To write a good release note or script it

Red Hat

What is a commit message?

- Title/subject line
- Body

Commit message example


Commit Title or Subject line
Commit Body

```
commit <commit_id>
Author: <author_name> <author_email>
Date:   Mon Apr 2 15:10:03 2020 -0400

    Change how workers are represented

    * Don't serialize the 'gracefully_shutdown' field
    * Create a new 'missing' property and serialize it
    * In the status API, list both online and missing workers

    Requires PR: https://github.com/<project>/pull/921
    closes #354498
    https://bugzilla.redhat.com/show_bug.cgi?id=354498
```

Usage of a commit title

- git log --pretty=oneline
- git rebase --interactive
- merge.summary
- git shortlog
- git format-patch, git send-email, ...
- reflogs
- GUI tools for committing and browsing
- GitHub, SourceForge, Bitbucket, GitLab, ... service

What constitutes a good commit message?

- git commit -m “Fix login timeout bug”
- git commit or git commit --verbose

```
Redirect user to the requested page after login

https://link/to/issue/tracker
```

What constitutes a good commit message?

- Capital letter, 50/72, no punctuation in the end

```
$ git commit
A brief summary of the commit

A paragraph describing what changed and its impact."
```

What constitutes a good commit message?

- Present Tense and Imperative Mood

```
cf31d12 Adds unit tests for login
7a9kj4f Fixed unit tests
101q2wd Update unit tests
1b7hn61 Removing unit test
```

"If accepted, this commit will *<your commit message goes here>*."

Red Hat

Ticket number in commit messages

- Ticketing system **!=** git log
  - “TICKET-123456 add missing params to class”
  - “Add missing meta fields to response”

❏ Takes space in 50 chars limit title

❏ Look-elsewhere for details message, I’m lazy

❏ May be not available for interested user or reviewer (permissions, outage)

Signing off your commits

- Kernel requires you to “sign-off” your code changes:
    - kernel.org/.../submitting-patches.html#sign-your-work-the-developer-s-certificate-of-origin
- In general: please make sure your proposal conforms to contribution guidelines

```
commit a6b88effc8b24d7216a762a42f365adeb31c903c
    Build SRPMs in Copr

    Signed-off-by: Tomas Tomecek <ttomecek@redhat.com>
```

What constitutes a good commit message?

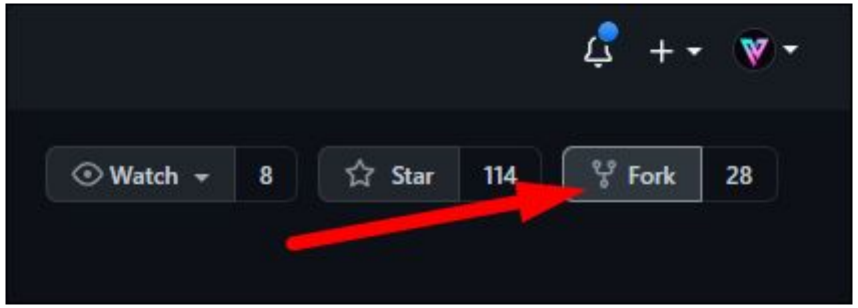
- Clear Title - What is commit about?
- Present Tense and Imperative Mood
- No punctuation in a title
- Clear Body - What and why is it needed/changed vs how?
- 50/72
- Reference to an issue in a body message
- Follow the commit convention defined by the team

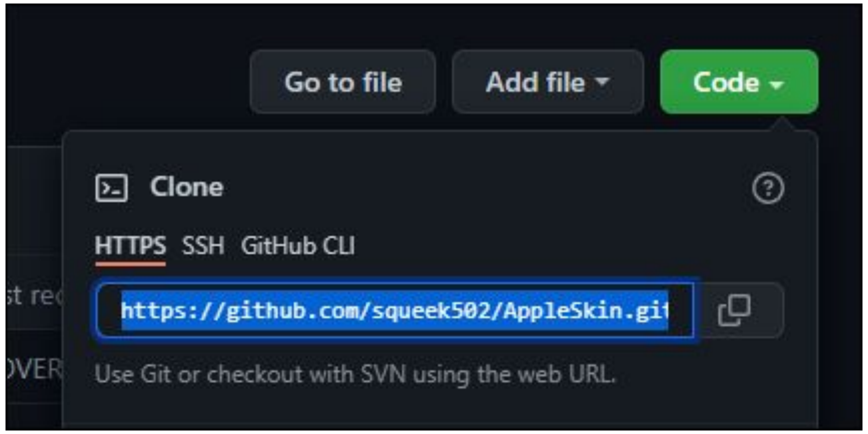
How to contribute to a team/community repo?

FORK IT

How to fork

- Click Fork in a team repo
- Target your personal namespace





https://github.com/squeek502/AppleSkin.git

Use Git or checkout with SVN using the web URL.

Red Hat

Git remotes

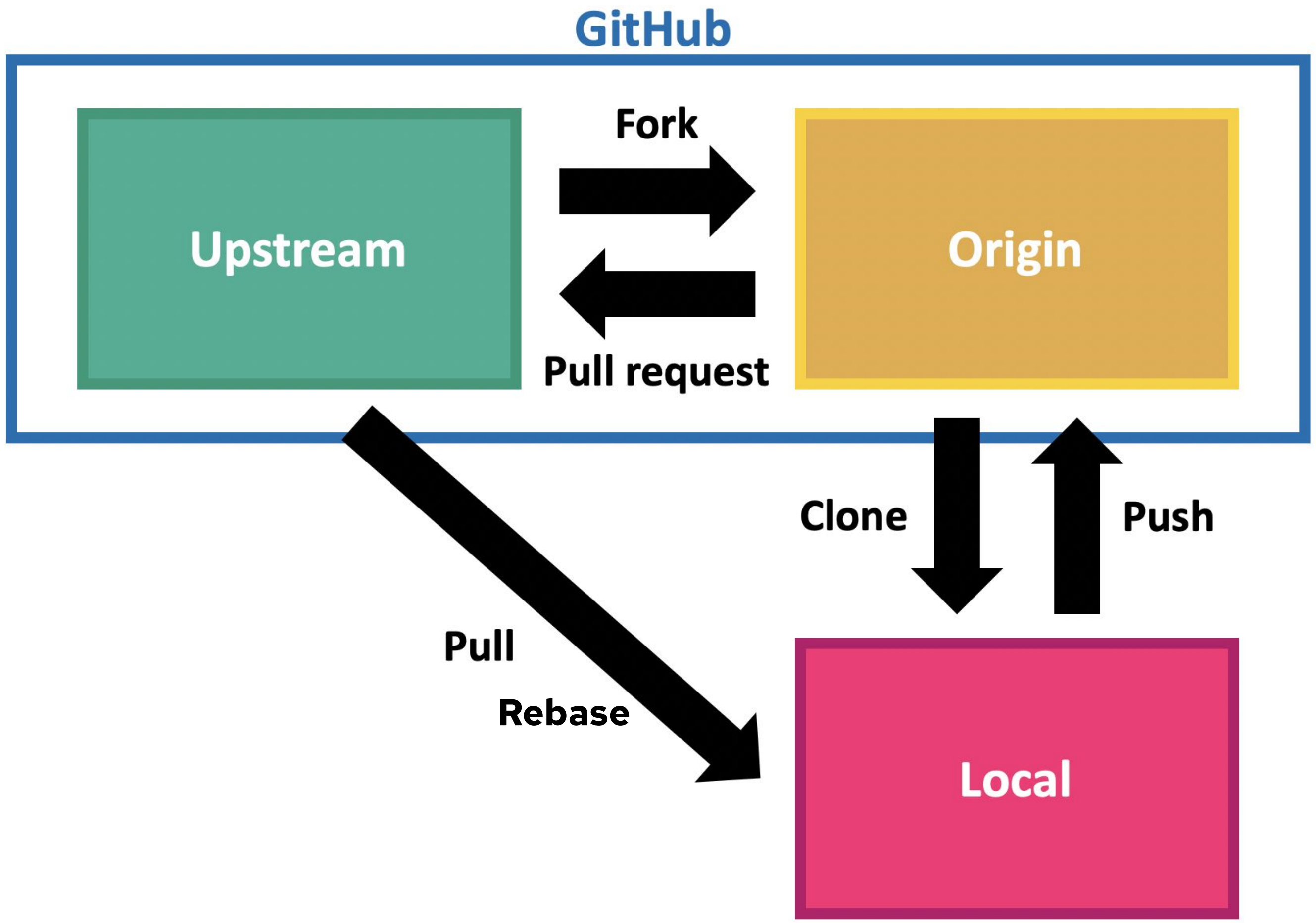
Clone your fork via SSH

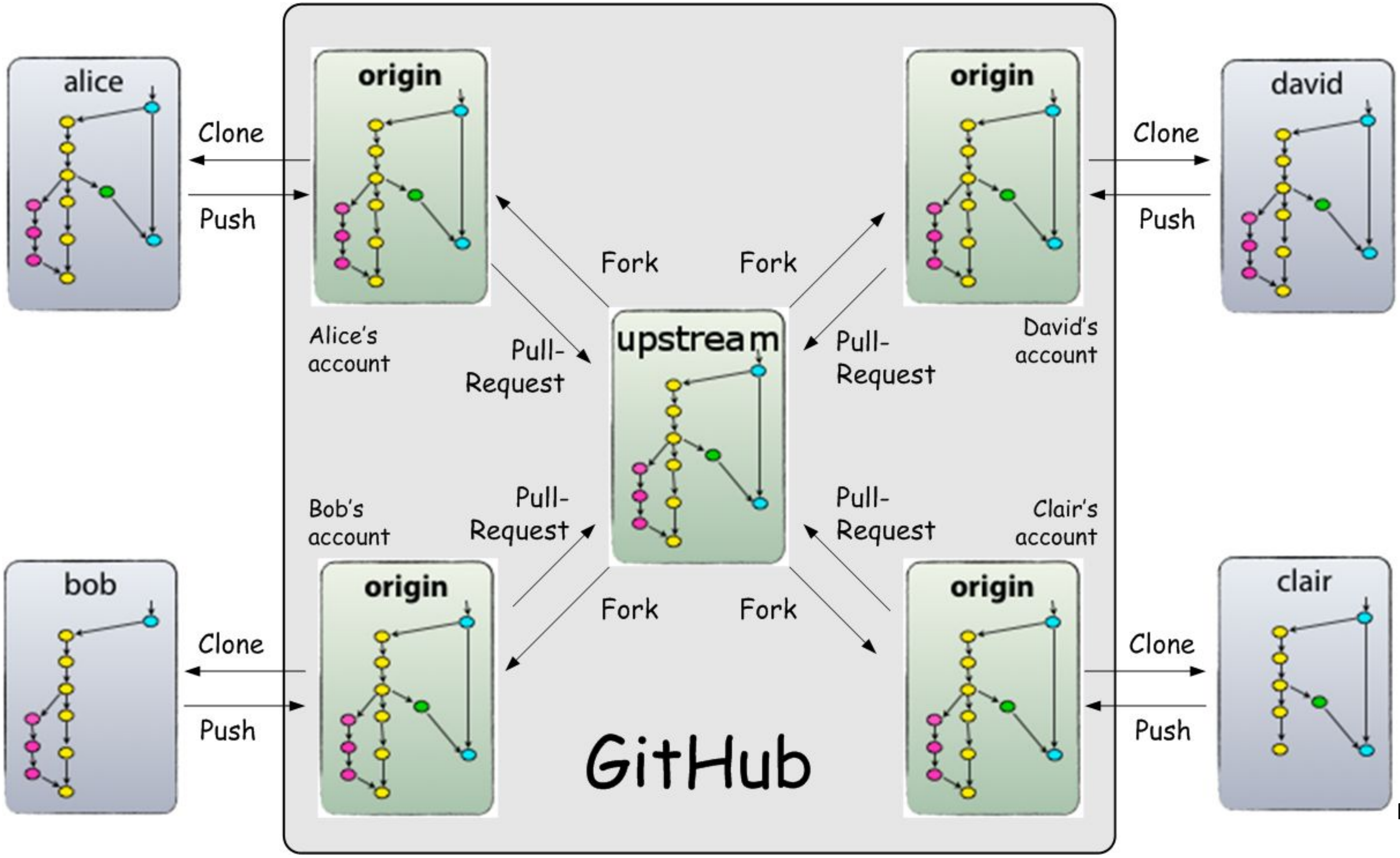
```
$ git clone
git@github.com:TomasTomecek/packit.git
Cloning into 'packit'...
remote: Enumerating objects: 13351, done.
remote: Counting objects: 100% (436/436), done.
remote: Compressing objects: 100% (321/321),
done.
remote: Total 13351 (delta 238), reused 255
(delta 115), pack-reused 12915
Receiving objects: 100% (13351/13351), 22.10
MiB | 5.31 MiB/s, done.
Resolving deltas: 100% (9359/9359), done.

$ cd packit
```

Add HTTPS remote upstream

```
$ git remote add \
upstream https://github.com/packit/packit.git

$ git remote -v
origin   git@github.com:TomasTomecek/packit.git
upstream https://github.com/packit/packit.git
```

How to fork


GitHub
Upstream
Fork
Origin
Pull request
Clone
Push
Pull
Rebase
Local



alice
Clone
Push
origin
Alice's account
Fork
Pull-Request
origin
David's account
Fork
Pull-Request
david
Clone
Push
upstream
bob
Clone
Push
origin
Bob's account
Pull-Request
Fork
origin
Clair's account
Pull-Request
Fork
clair
Clone
Push
GitHub

Fork a community/team repo

- In your local Fork do whatever you want, BUT
- Mind branch naming (if they are for PR/MR)
  - Bad branch name: **Irina**, 🍌🦷🏔️, **main**
  - Good branch name: **docs_on_upgrade_feature**, **fix1337**
- Mind commit titles

Git Push

IF YOU DO FORCE PUSH...

May the force stay with you.

You have a great freedom…

to change your history **locally.**

Git push --force trap

- It’s ok to force push to your local branch
- It’s ok to force push to your (unmerged*) PR
- **It’s not ok to force push to a public branch**

Git push --force consequences

- Lost data
- Altered history
- Not happy colleagues
- Lost karma points

How to avoid unwanted force push

- Protect important branches
- Backup
- Use git checkout -b
- Use --force-with-lease, carefully
- Use PR revert

Submitting a change

Pull request (PR)

==

Merge request (MR)

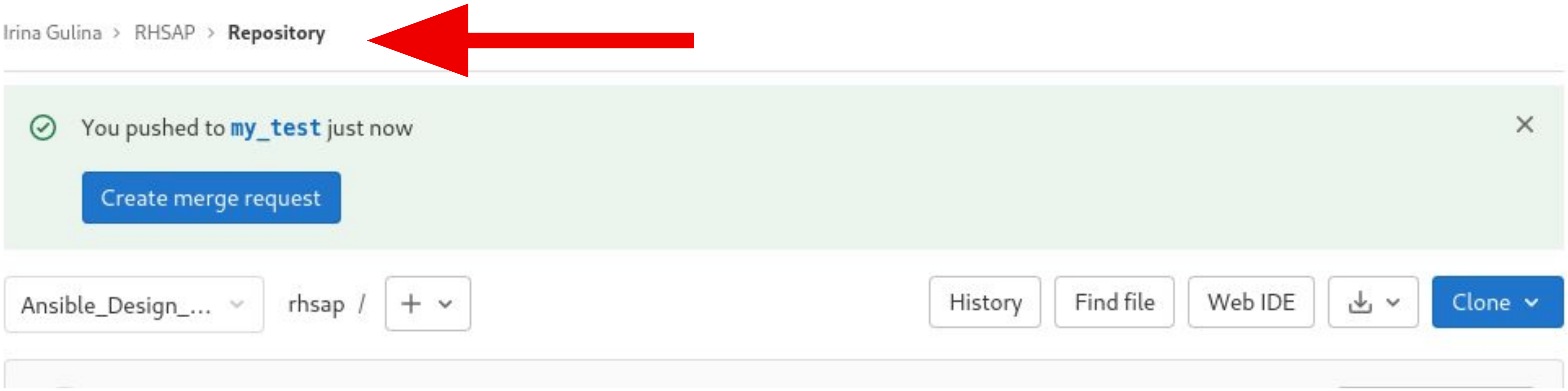
Red Hat

Why do we use PR/MR workflow?

- Share changes
- Get review and feedback
- Encourage quality
- Test and collaborate consistently

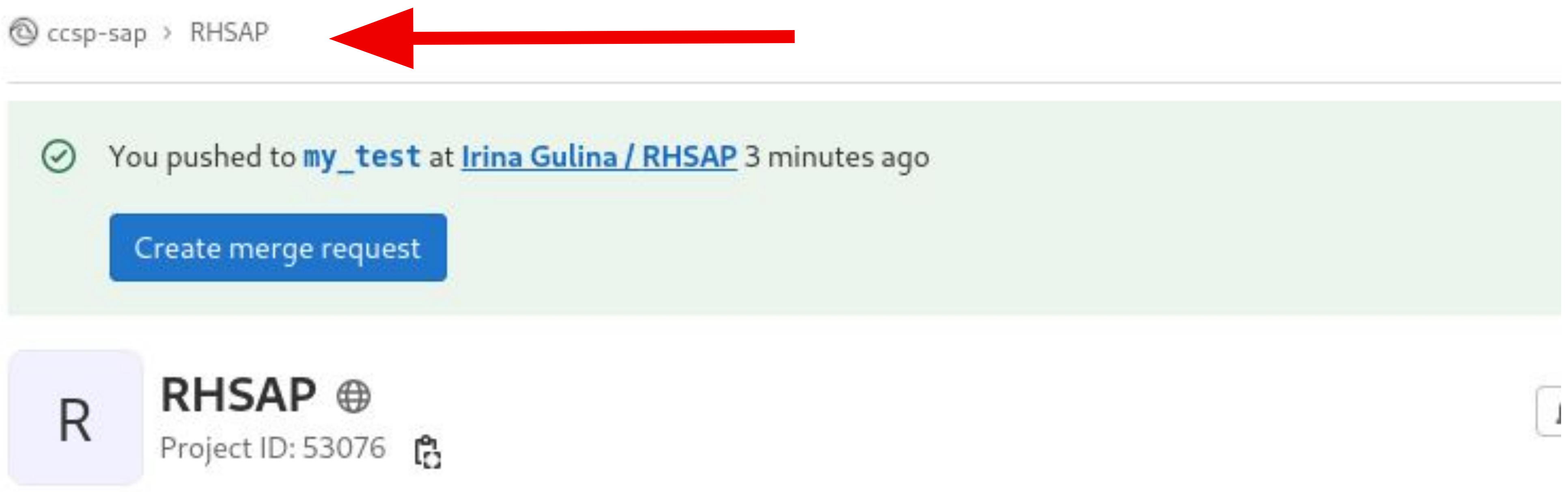
Creating a PR

- When you push from your local to your origin, every time it will ask you to create a MR/PR



Creating a MR/PR

- When you push from your local to your origin, every time it will ask you to create a MR/PR


ccsp-sap › RHSAP

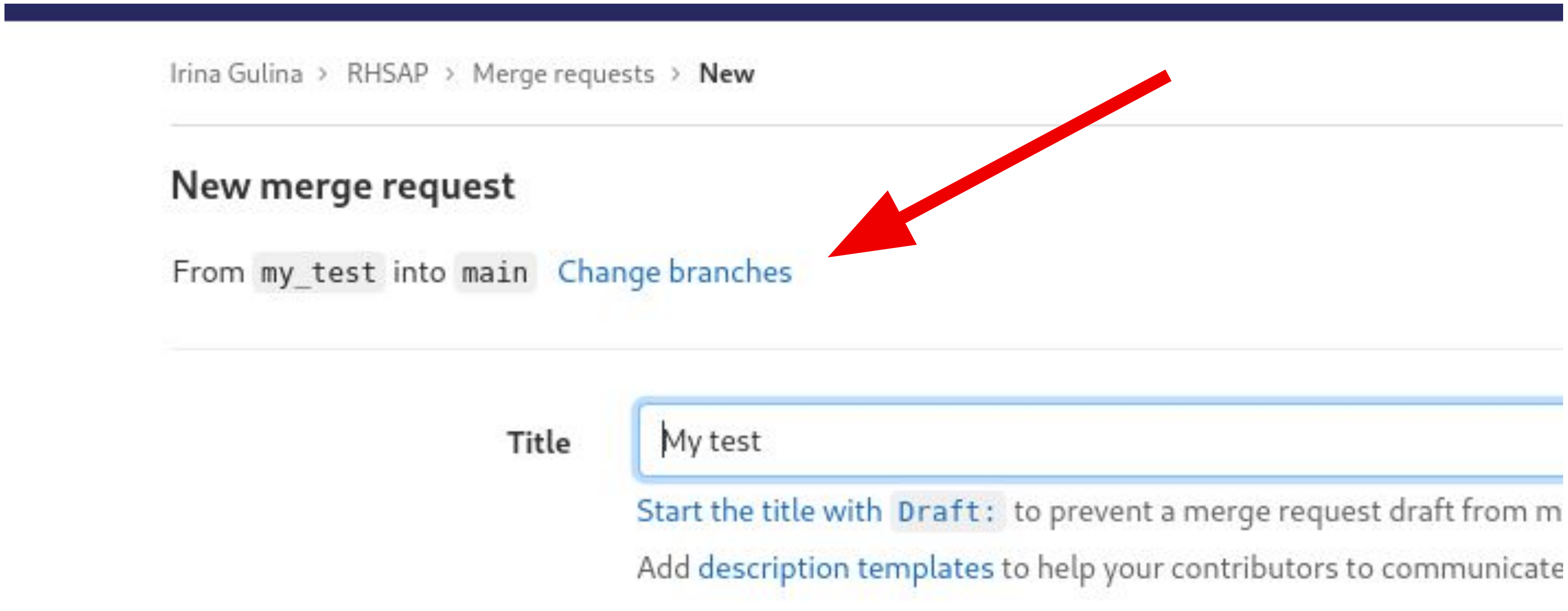
You pushed to my_test at Irina Gulina / RHSAP 3 minutes ago

R RHSAP

Project ID: 53076

Creating a PR

- When you push from your local to your origin, every time it will ask you to create a MR/PR


Irina Gulina › RHSAP › Merge requests › New

New merge request

From `my_test` into `main` Change branches

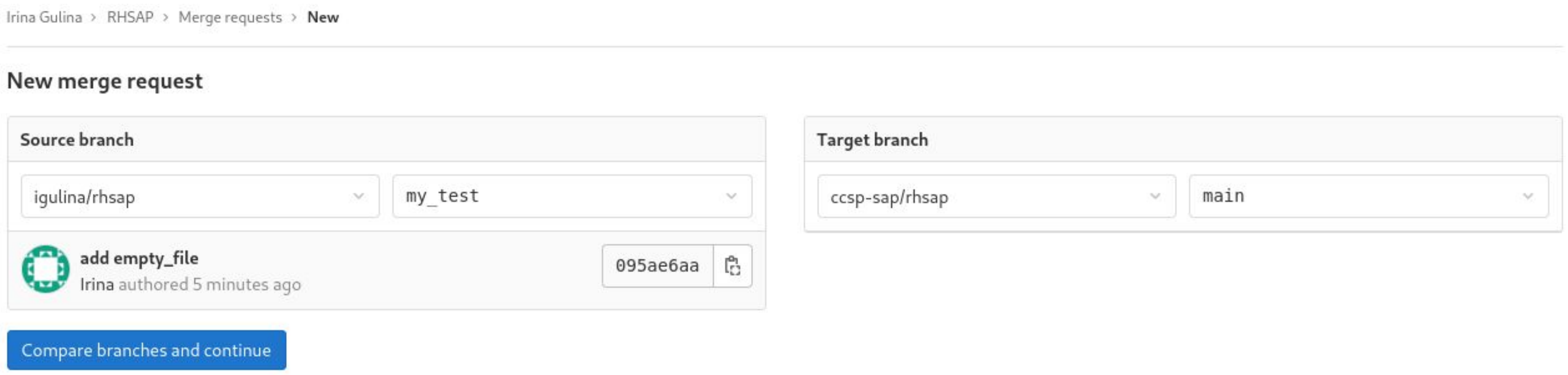
Title: My test

Start the title with `Draft:` to prevent a merge request draft from m

Add description templates to help your contributors to communicate

Creating a PR

- When you push from your local to your origin, every time it will ask you to create a MR/PR



Irina Gulina › RHSAP › Merge requests › New

New merge request

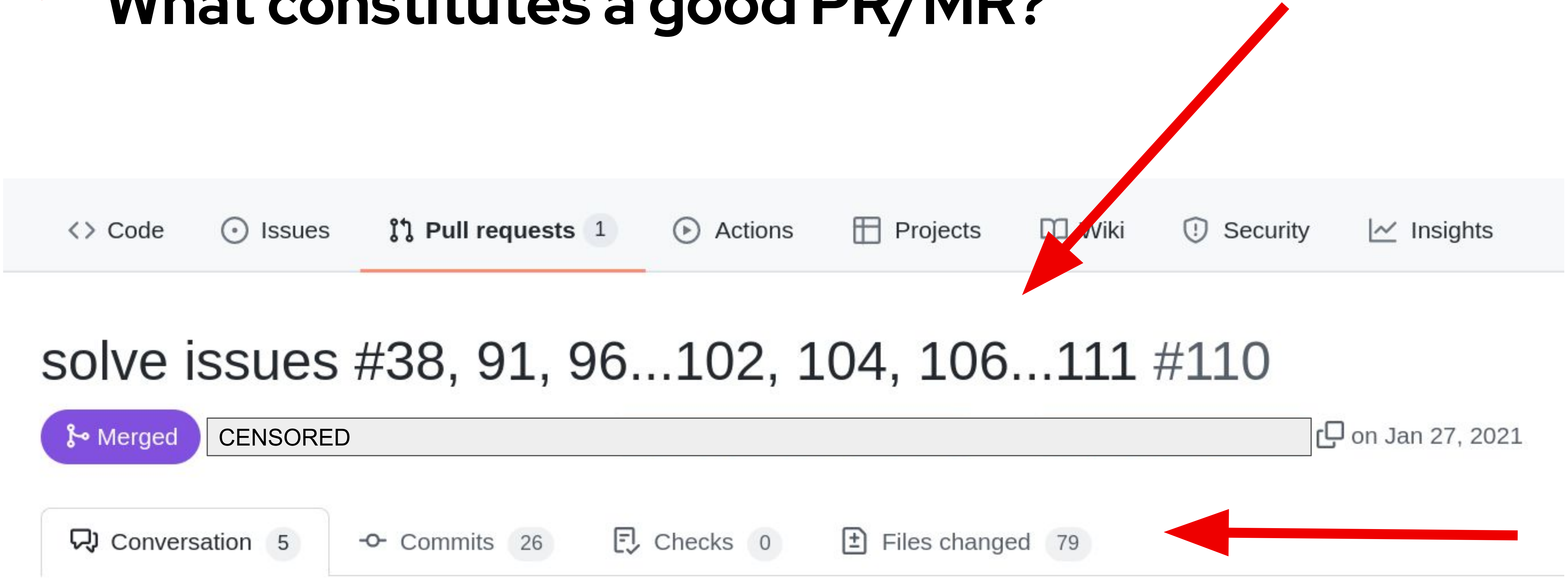
Source branch		Target branch	
igulina/rhsap	my_test	ccsp-sap/rhsap	main

add empty_file
Irina authored 5 minutes ago
095ae6aa

What constitutes a good PR/MR?

- Complete piece of work
- Adds value in some way
- Solid title
- Body explains the change
- Clear commit history
- Small
- Meets project's contribution guidelines

What constitutes a good PR/MR?



solve issues #38, 91, 96...102, 104, 106...111 #110

Merged CENSORED on Jan 27, 2021

Contributors (before submitting a PR/MR)

- Follow the repo’s conventions (especially templates)
- Double check your code (and TODOs)
- Your change is documented
- Keep changes small
- Separate branch (don’t create from main)
- Be clear and specific
- And kind, please

Contributors (after submitting a PR/MR)

- Check your ego and be polite
    - **@username ping!**
    - **@username review please**
- Ensure your branch can be merged and tests pass
- Use --amend, --fixup or rebase -i
- Don't merge your own PR

WIP PR/MR

- WIP = Work in progress
- Don’t overuse WIP label
- Remove WIP label when ready
- “This is ready for review, please.”

Reviewing a PR

PR Reviewers

- Be kind and polite
  - **@username ping, error here!**
  - **@username s/foo/bar/, because bar can...**
- Check commit history
- Don’t fix issues
- Collaborate, don’t command
- Ensure the branch can be merged
- CI Tests pass
- Don’t merge WIPs
- Follow project’s merge process

Source: https://octodex.github.com/

Red Hat

QUESTIONS?

Irina Gulina, QE
igulina@redhat.com

Red Hat

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat