



Introduction to git part 4

EPFL - Embedded Systems Laboratory

2025



Introduction to git

introduction and basic use

Working alone

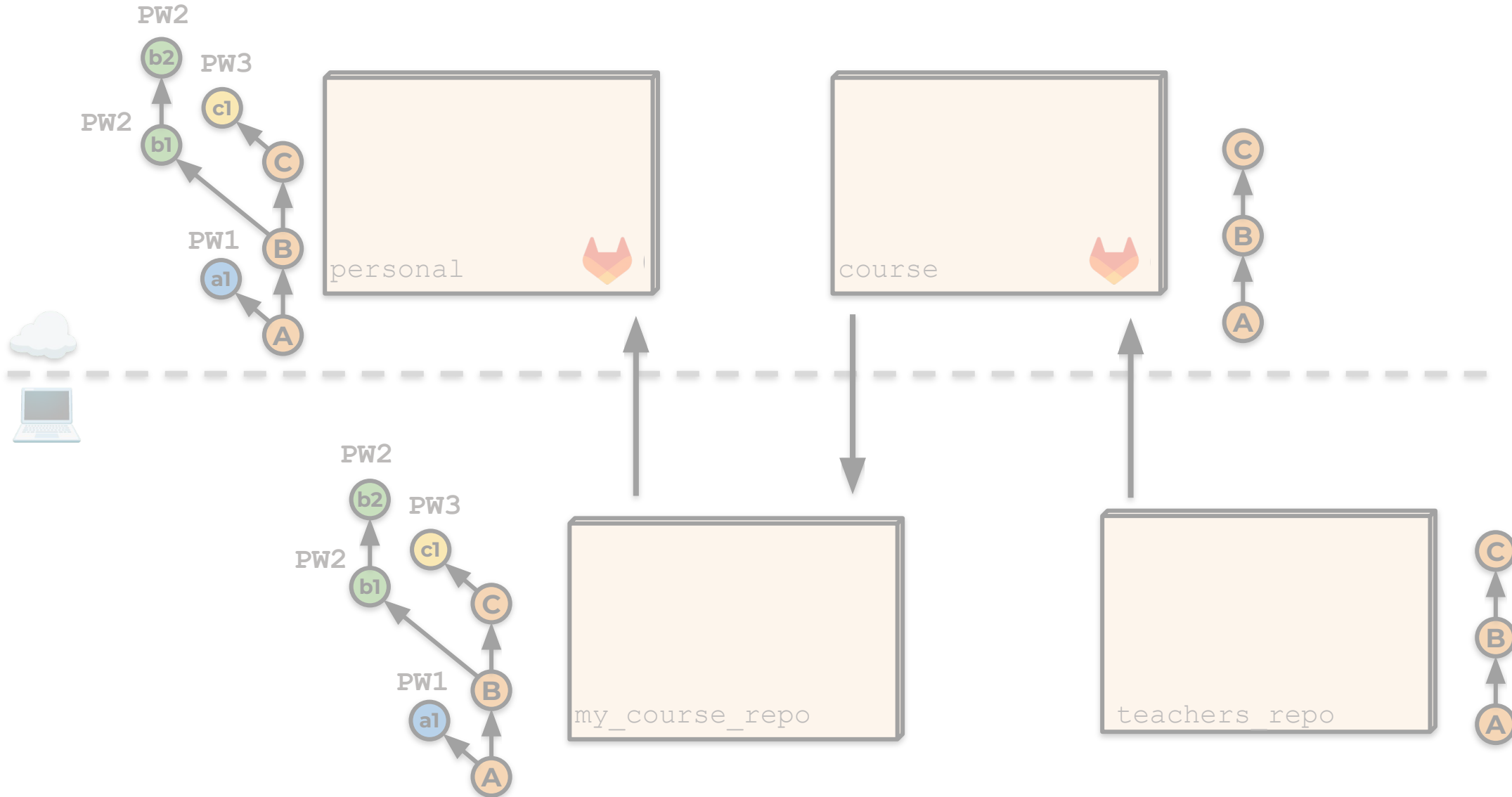
for any project you have

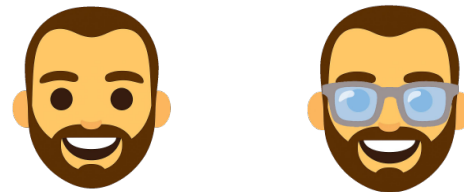
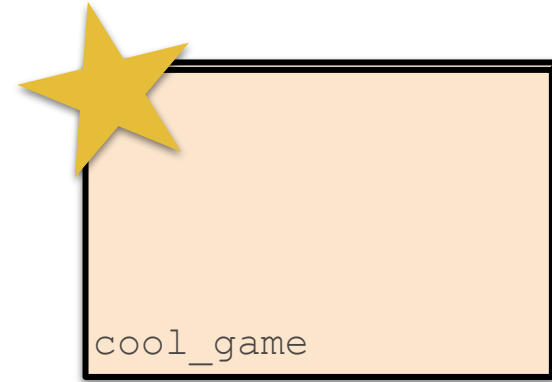
Using other's work

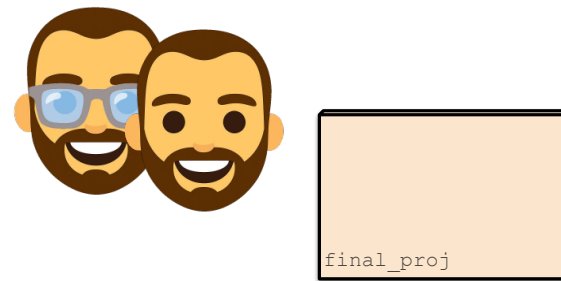
e.g. the practical works

Working with others

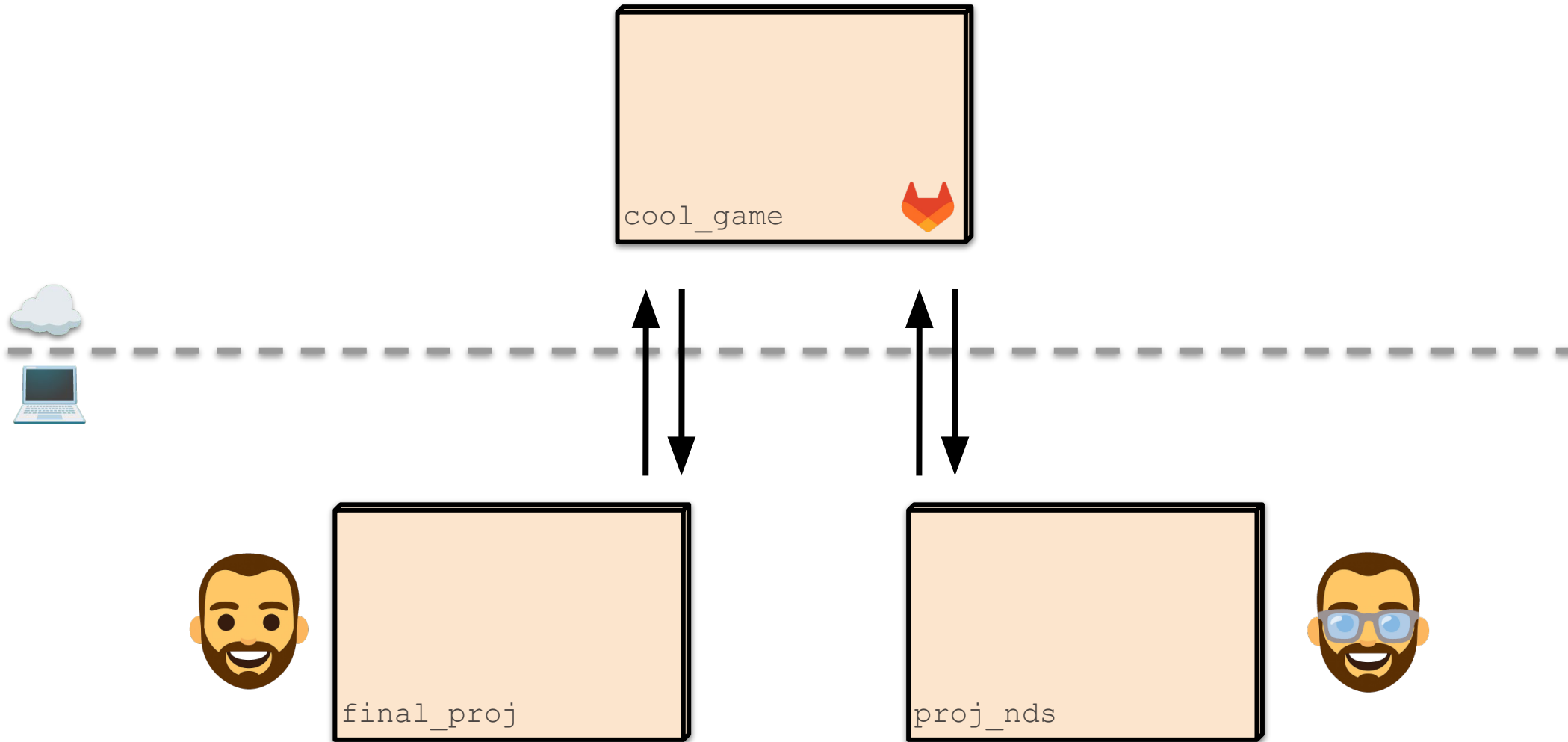
e.g. your final project



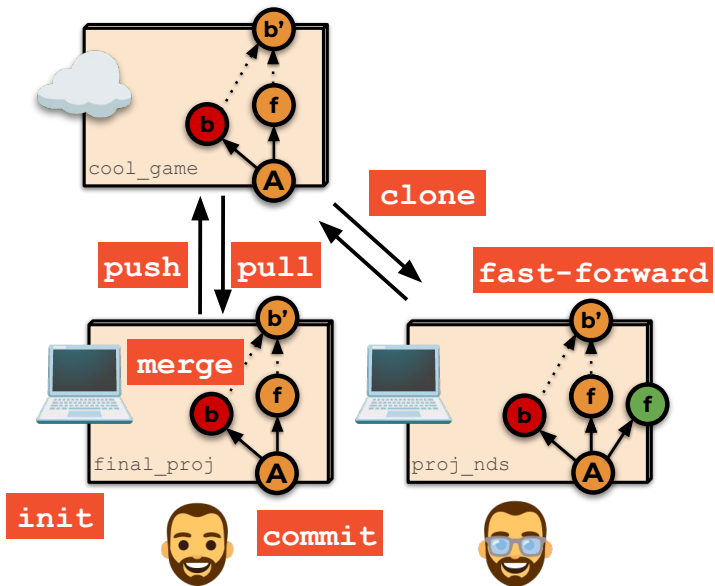








Sharing a single
remote repo,
merging locally

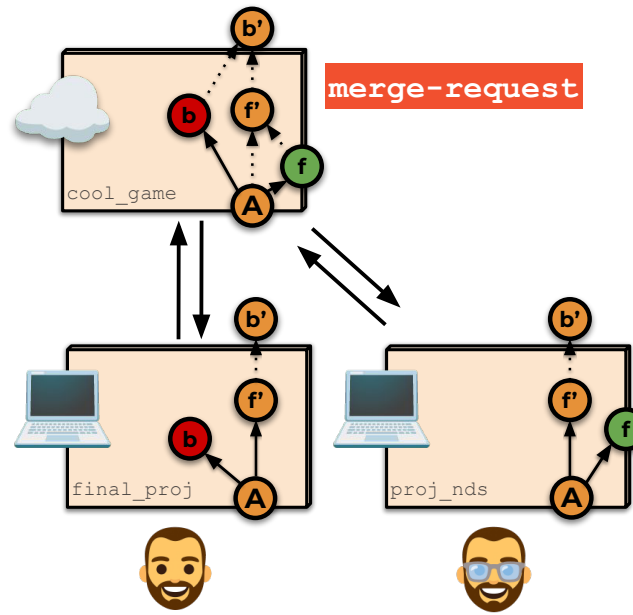
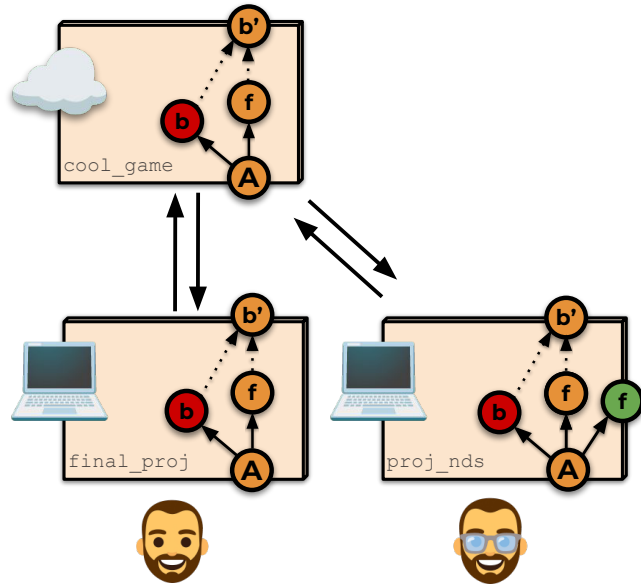




Handling a Project

Sharing a single remote repo, merging locally

Sharing a single remote repo, merge requests



Prune the compiled code to its minimum expression #639

Conversation 53 | Commits 38 | Checks 4 | Files changed 18 | +124 -199

Objective

The objective of this PR is to minimize the code as much as possible. Some new improvements will not affect compilation at all, such as adding `--gc-sections`, which simply removes unused sections and can free several kB.

Other changes might be more drastic and I will set them as optional and heavily disadvised configurations.

Background

Just read **Document**

The initial size was 24 kB. By removing the unused sections it goes down to 12 kB.

The remaining 12 kB are mostly due to:

Tasks

- Remove unused sections
- Not compile unused interrupt handlers
- Prune functions in appcalls to make it lighter
- Test on Quesastm
- Test on FPGA

Important

This PR builds on top of #636. It is blocked by the merging of that one first (or I will have to cherry pick changes)

Review changes

```
sw/device/lib/drivers/fast_intr_ctrl/fast_intr_ctrl.c (Quoted)
65 + #define INTERRUPT_HANDLER_API_ATTRIBUTE __attribute__((aligned(4), naked))
66 +
67 +
68 + /**
```

davideschiavone on Feb 21

I don't think this is dangerous, I think this is wrong. When you jump to an async event (and return from it) you need to push to the stack all the registers you are gonna modify within the ISR and then restore them. How do you know that you are gonna use only those 5 registers?

Especially if they are weak, so could be redefined by users?

JuanSapirza on Feb 21

In the handler the registers used are well known, the problem is indeed in the weak implementation.

If I managed to make all the registers be backed-up upon entering the weak implementation, that would be perfect. However, I could not manage to make it work. Can you think of anything that would prevent that?

Reply...

Resolve conversation

JuanSapirza commented on Mar 10

Also working on FPGA. Tested some random apps: hello_world, example_minimal, fit

Auto-merge

davideschiavone approved these changes on Mar 10

davideschiavone merged commit be330e into esp1.main on Mar 10

JuanSapirza deleted the pr_bin_sw branch 8 months ago

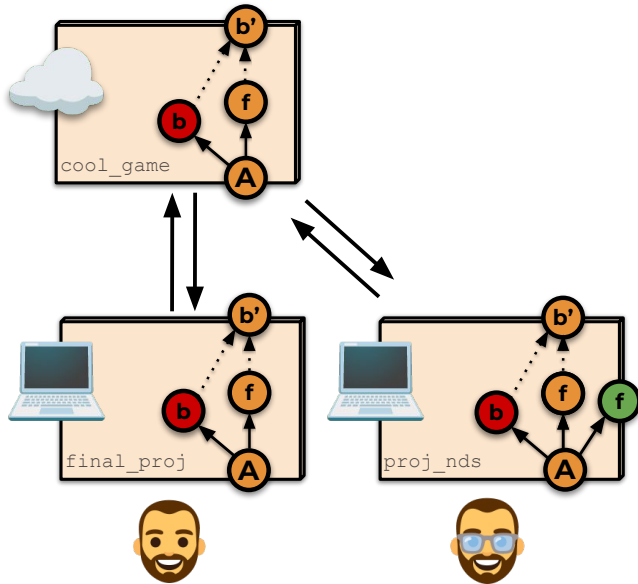
davideschiavone added a commit that referenced this pull request on Mar 14

pr_rabasa_spi (#679)

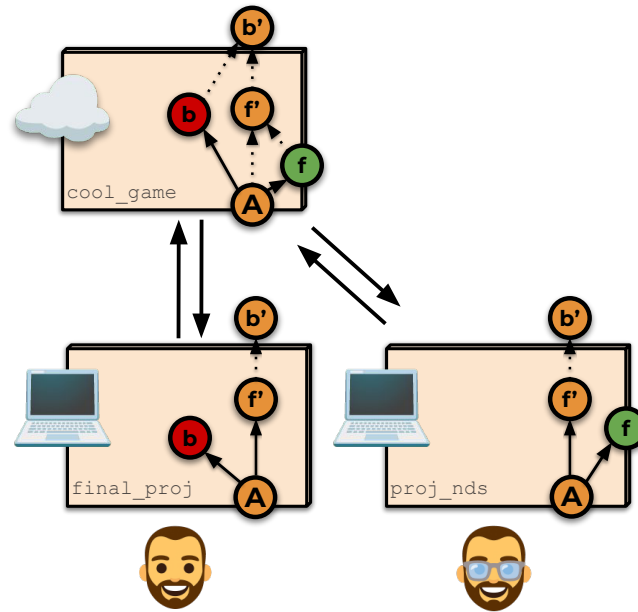
Pull request successfully merged and closed

You're all set — the branch has been merged.

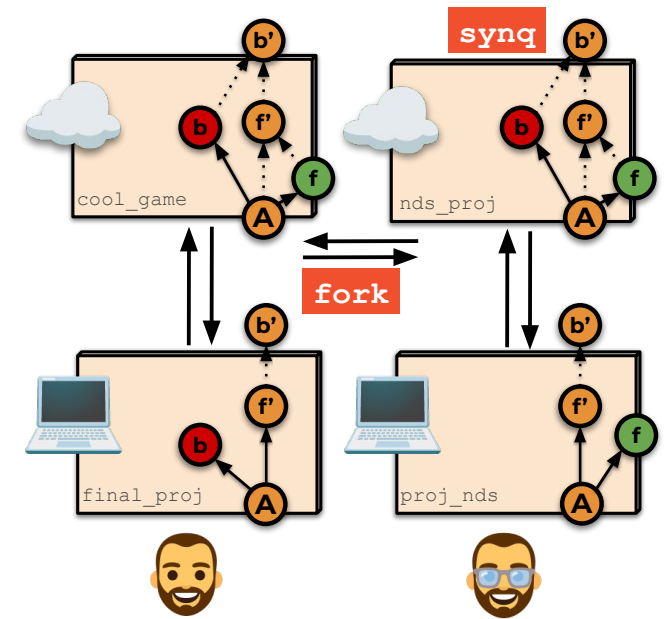
Sharing a single remote repo, merging locally



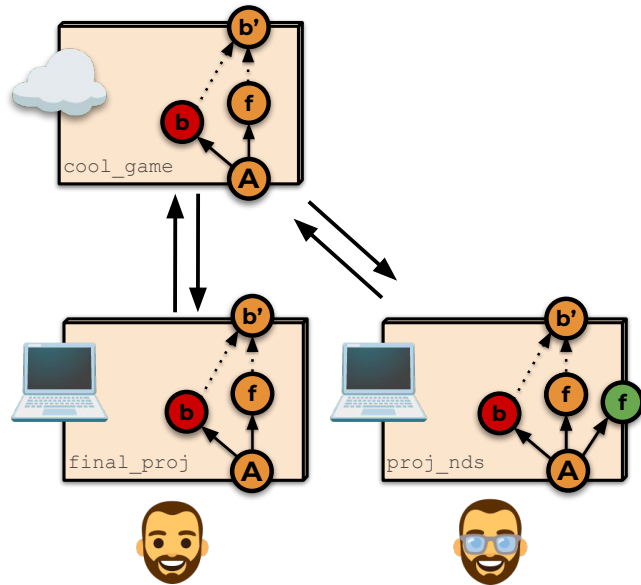
Sharing a single remote repo, merge requests



Fork of main repo, merge requests

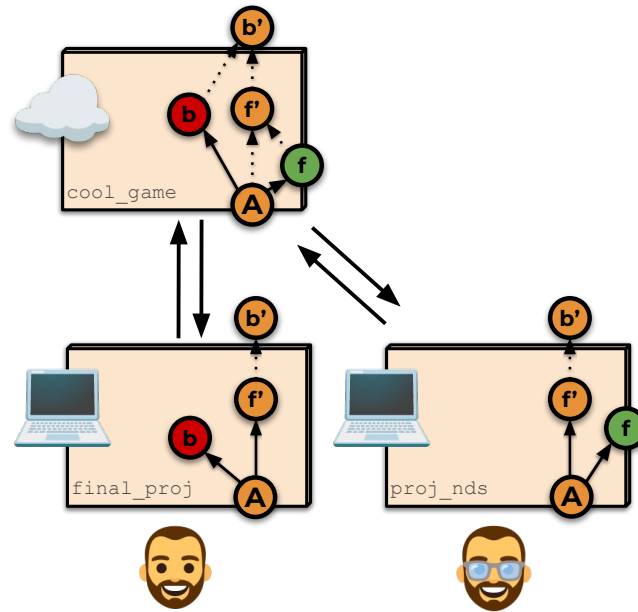


Sharing a single remote repo, merging locally



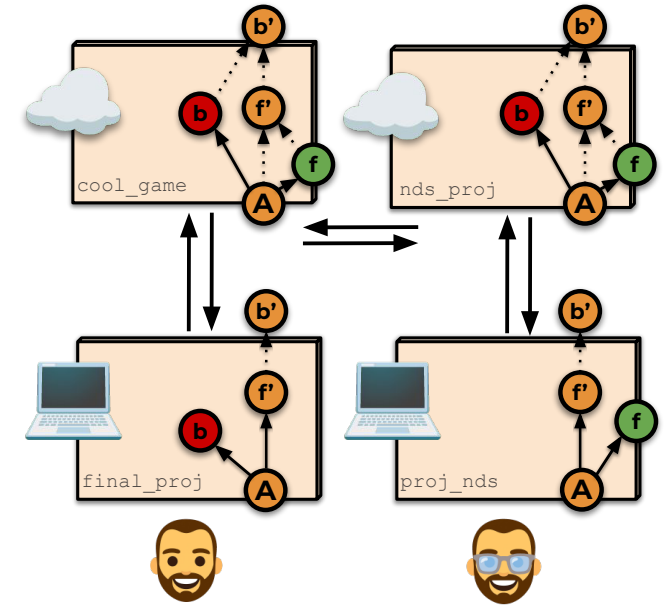
Lots of manual merging

Sharing a single remote repo, merge requests



Manual merging only for conflicts

Fork of main repo, merge requests

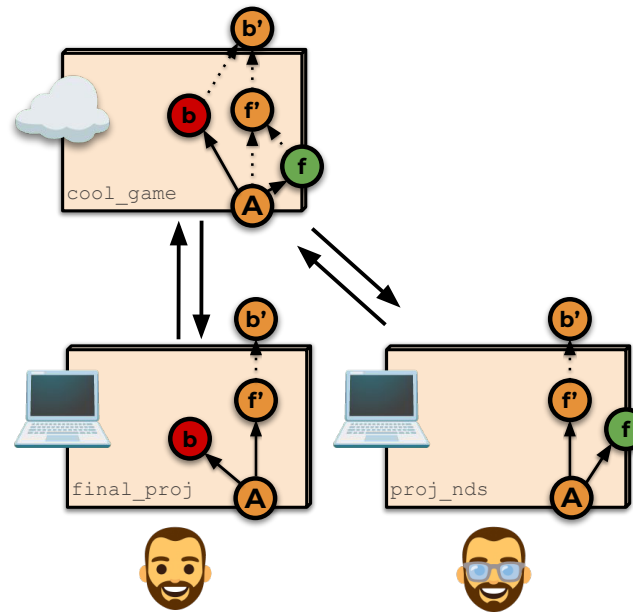


For larger projects

Ignoring files

- ◆ Share a repo
- ◆ Open an MR
- ◆ Reviewing an MR
- ◆ Merging an MR
- ◆ Solving conflicts

Sharing a single remote repo, merge requests



Manual merging
only for conflicts

```
# Create the local repo
mkdir Juan
git init

# Create a new folder
# Turn it into a repo

# Create the remote repo
# New project/repository
# Create Blank repository
# Uncheck add README
# ADD MEMBERS! Manage > Members
# Code > copy SSH URL

# Add a commit and push
git remote add origin <url>
echo hello > a.txt
git add a.txt
git commit -m "First commit"
git push origin main

# Clone the repo as someone else ("Juano")
cd ../
git clone <url> Juano
git checkout -b feature
echo bye > b.txt
git add b.txt
git commit -m "Added a feature in b"
git push origin feature # Check how a new branch appears in the remote

# Create a Merge Request
# Ctrl+click on the link to create MR
# Mark as draft and add a description
# Create it

cd ../Juan/
# commit, restore or stash changes
git checkout -b feature origin/feature
# You can commit and push, but be careful! Juano will need to pull these changes to continue working.
# Add comments or and a review

# Add another commit and push
echo Good byeee > b.txt
git commit -am "Emphasized"
git push origin feature

# Check changes, mark as ready, and merge!
```

```
# See the effects of merging remotely
git fetch
git log --all --oneline --graph # Note that main is behind, and that a merge was performed remotely
git checkout main
git pull # Update main branch

# Do another MR, but using squash commit
git checkout -b other_feature
# Do 2 commits on this branch
git push origin other_feature

# Create MR
# Accept and squash commits

git log --all --oneline --graph # Note that changes are missing
git fetch
git log --all --oneline --graph # Note that changes in "Other" are not the ones merged!
# Also note that the feature branch was not deleted??

git fetch --prune
git log --all --oneline --graph

# Make changes as the other colleague
cd ../Juan
git fetch
git log --all --oneline --graph # Notice how behind we are
git checkout -b someother_fefature
echo something > b.txt # Add a conflicting change
echo something else > c.txt
git add b.txt c.txt
git commit -m "some cool feature"
git push origin someother_feature

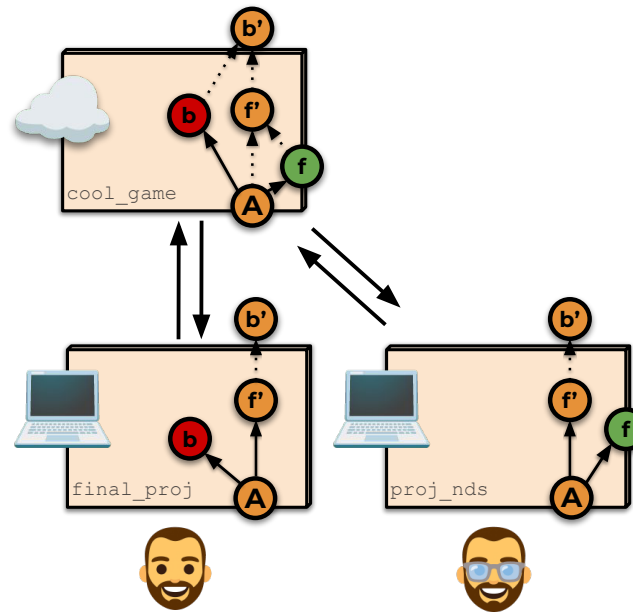
# Create a MR
# Conflicts will prevent the merging: accept "theirs"

git fetch --prune
git log --all --oneline --graph # Notice how the remote merged main on someother_feature, and viceversa
```

Ignoring files

- ◆ Share a repo
- ◆ Open an MR
- ◆ Reviewing an MR
- ◆ Merging an MR
- ◆ Solving conflicts

Sharing a single
remote repo,
merge requests



Manual merging
only for conflicts



Thank you!

EPFL - Embedded Systems Laboratory