# 21 – Collaborative development with hosted repositories

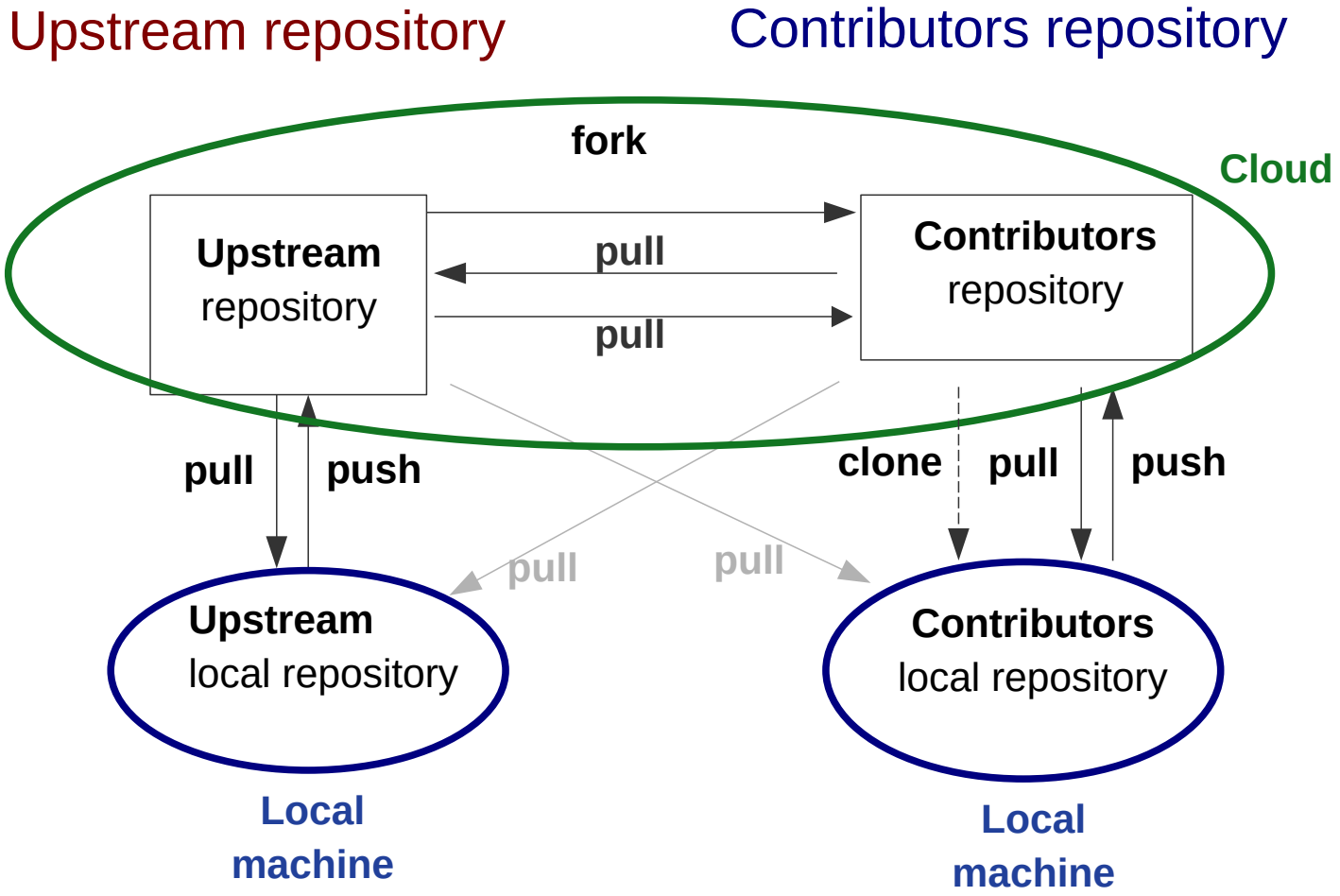
Bálint Aradi

**Scientific Programming in Python (2024)**

https://atticlectures.net/scipro/python-2024/

# Remote git-hosting

- Public git hosting sites use the "**fork-pull-push**" workflow
- Similar to "branch & merge in two repositories"
- Local repository is "published" via **push** to public hosting site
- Changes from other repositories are imported via **pulls** from the public repositories at the hosting site



# Create new repository (upstream)

- **Create** new repository locally (**local repository**) and make first commit

```
mkdir hello
cd hello
git init
```

```
print("Hello!")
```
hello.py

```
git add hello.py
git commit -m "Initial checkin"
```

- Create repository with same name on Git the hosting server (**remote repository**)
- **Connect** local repository with remote repository

Registers remote repository as “origin”

```
git remote add origin git@github.com:USERNAME/hello.git
git remote -v
```

Lists registered remote repositories

- **Push** local copy to remote repository “origin”

```
git push -u origin main
```

-u: connect main permanently with origin/main

# Fork & clone existing repository (contributor)

- **Fork** repository of upstream user on the Git hosting server
  (creates a copy on the hosting server, copy remains associated with upstream repository)
- **Clone** repository from your namespace

```
git clone git@github.com:YOUR_USERNAME/hello.git
```

- **Register upstream** repository (e.g. for keeping track of updates on upstream/main)

```
cd hello
git remote add upstream git@github.com:UPSTREAM_USER/hello.git
git remote -v
```

# Developing a feature (upstream)

**Upstream:** develop feature in feature branch, merge into main, push

```
git switch -c cleanup
```

hello.py

```
def main():
    print("Hello!")

if __name__ == "__main__":
    main()
```

```
git add -u
git commit -m "Wrap script in main()"
```

# Developing a feature (upstream)

README.rst

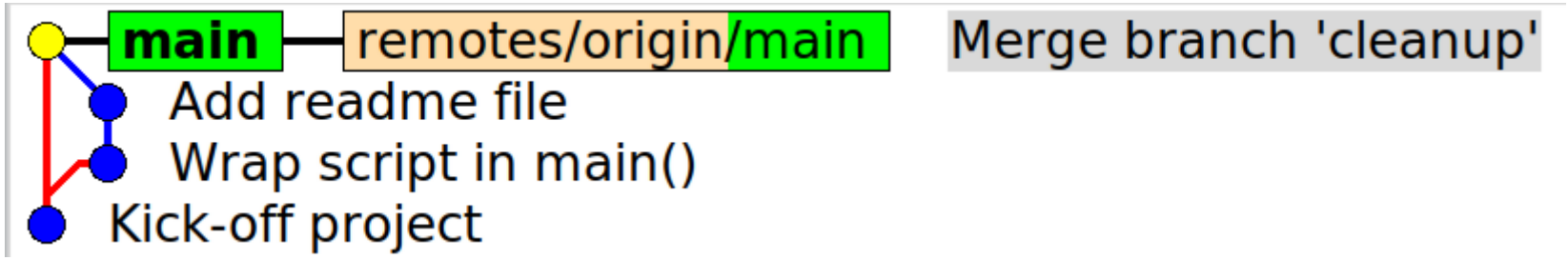
```
*****
Hello
*****


Trivial demo project.
```

```
git add README.rst
git commit -m "Add readme file"
```

```
git switch main
git merge --no-ff cleanup
git branch -d cleanup
```

## Publish your project changes on the remote git-host

```
git push origin main
```



# Developing a feature (contributor)

**Contributor:** develop feature in feature branch, synchronize changes, push, make PR

```
git switch main
git switch -c extend
```



```
print("Hello, World!")
```
hello.py

```
git add -u
git commit -m "Extend greeting"
```



```
print("Hello, World!")
print("How are you doing?")
```
hello.py

```
git add -u
git commit -m "Make greeting more polite"
```

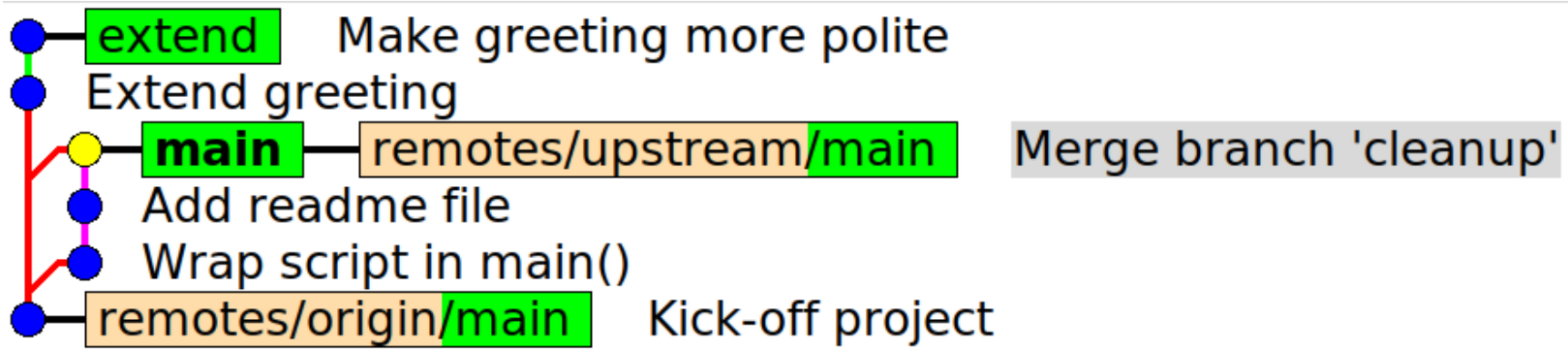
# Developing a feature (contributor)

## **Developer 2**: synchronize main branch with devel1/main

```
git switch main
git pull --ff-only upstream main
```

**Fast-forward only**: Would fail if main had been modified apart of being pulled from devel1/main

→ Main branch of contributor identical to upstream/main



# Developing a feature (contributor)

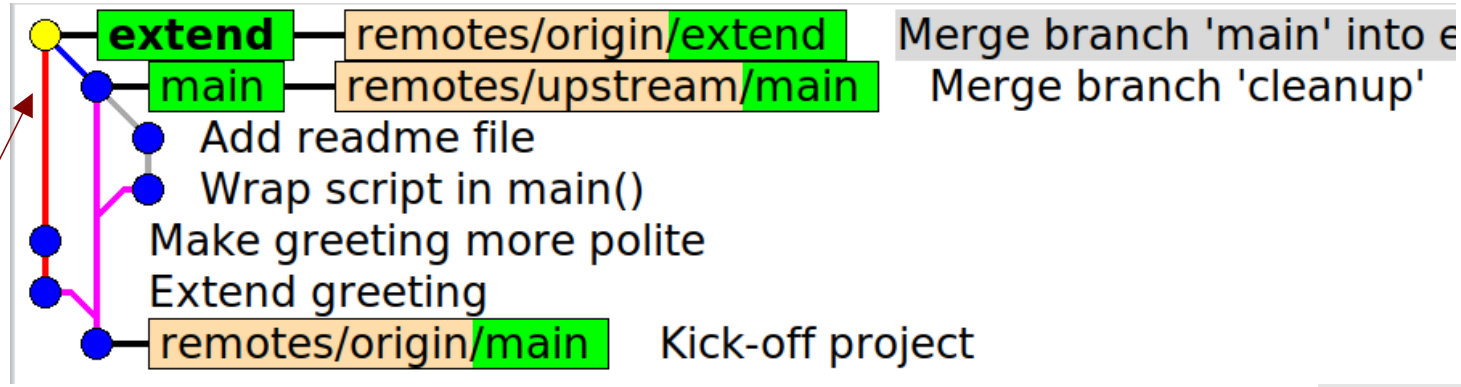
**Developer 2**: merge updated main into feature branch, fix eventual conflicts

```
git switch extend
git merge main
```

hello.py (resolved)

```
def main():
    print("Hello, World!")
    print("How are you doing?")

if __name__ == "__main__":
    main()
```

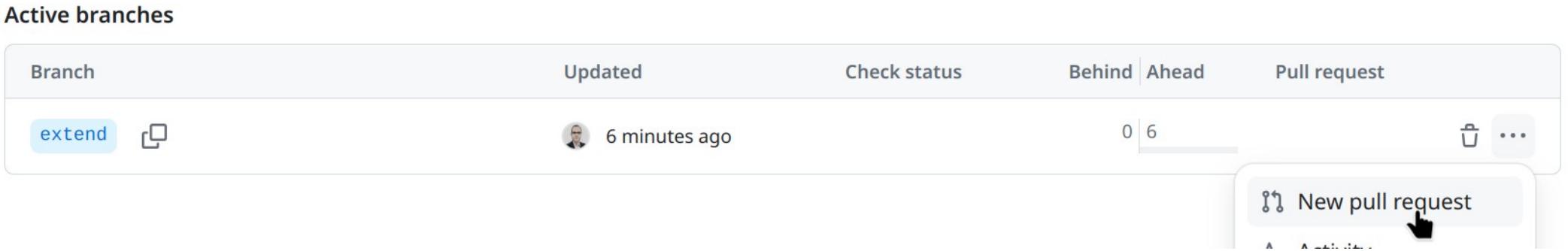
```
git add hello.py
git commit
```

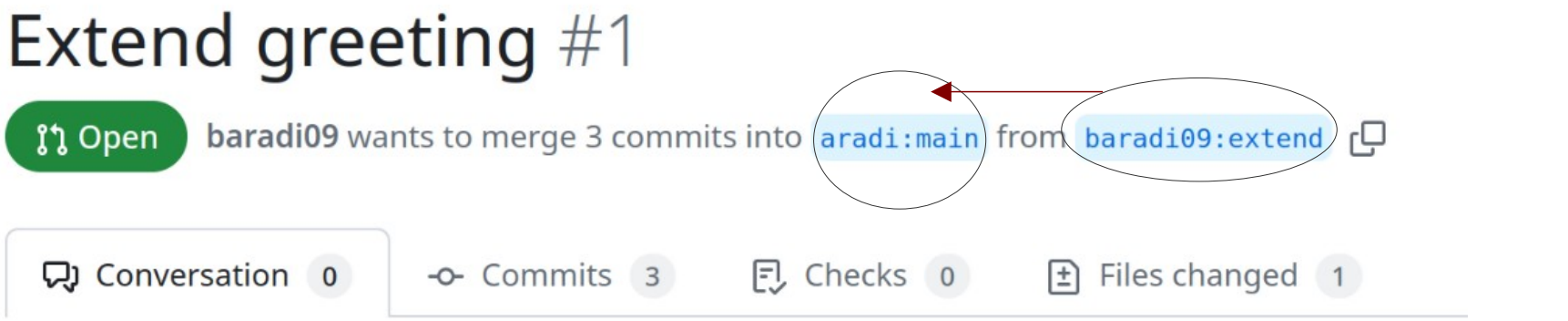



**Publish your changes**

```
git push origin extend
```

# Make pull request / merge request (contributor)

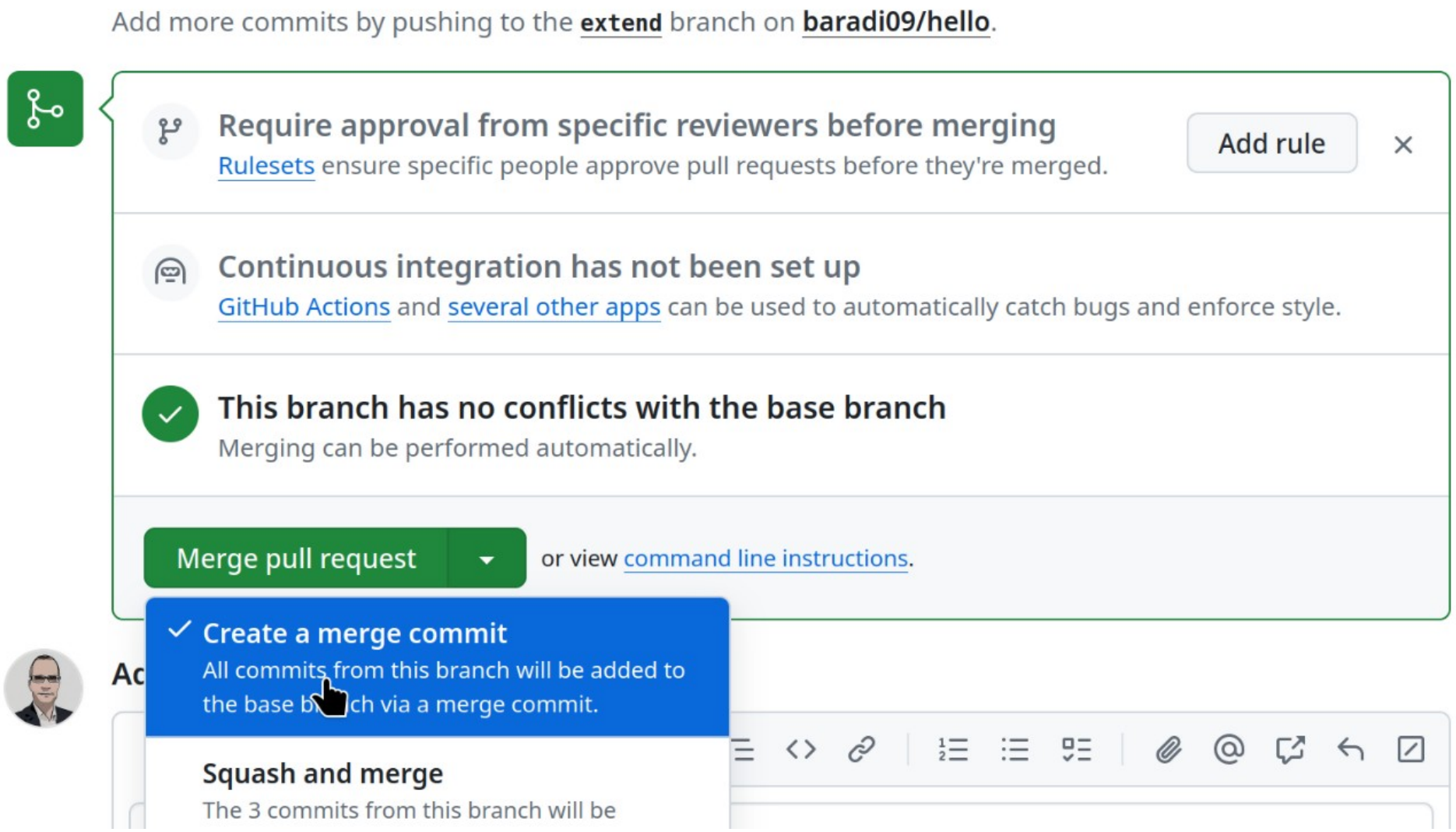



- You can add further commits to this branch during review and push it as before

# Accept pull request (upstream)

- When PR gets accepted, changes are merged to upstream/main



# Sync main with upstream/main (contributor)

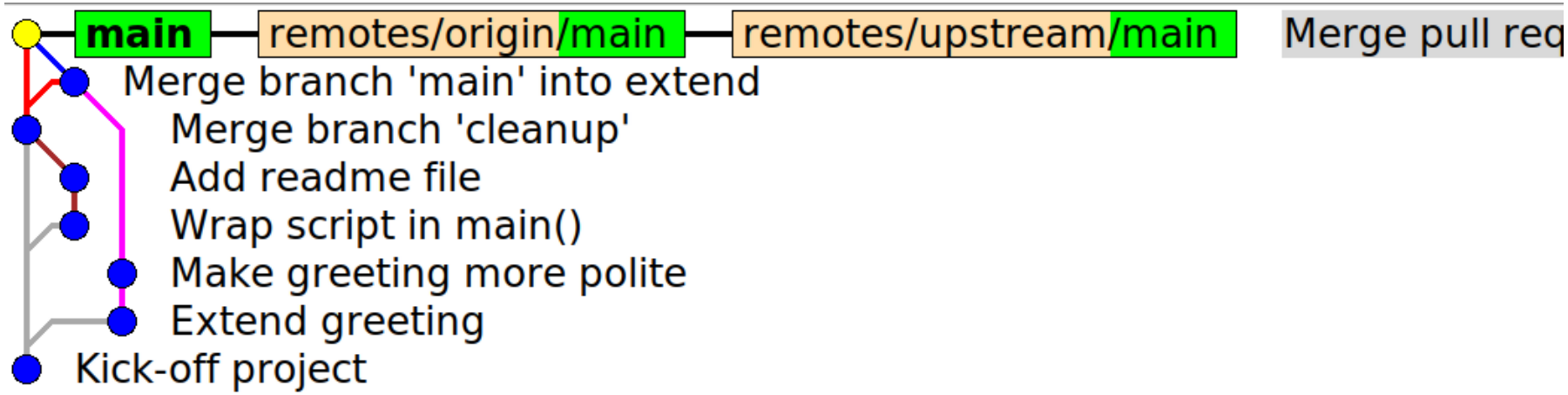
```
git pull --ff-only upstream main
git push origin main
```

```
git branch -d extend
```

Deletes local branch

```
git push --delete origin extend
```

Deletes remote branch



# Contributors workflow (summary)

Fork / Pull / Push model

- Keep up to date with current changes on main
- Create feature branch

```
git switch main
git pull --ff-only upstream main
```

```
git switch -c BRANCH_NAME
```

- Develop feature
- (Synchronize with upstream changes, if needed)

Develop locally

Synchronize with upstream if needed

- Push branch

```
git push origin BRANCH_NAME
```

Make pull request on host

- Make pull / merge request

Wait for upstream to merge changes

- When feature had been merged:
  synchronize,
  delete local and remote branch

```
git switch main
git pull --ff-only upstream main
git branch -d BRANCH_NAME
git push --delete origin BRANCH_NAME
```

# Few random notes

- **Repository** can be made **private**, contributors must be then invited to the repository (they usually gain write access to the repository then)
- If all developers have write access to the repository (small projects), the same repository might contain all temporary feature branches (no forking necessary)
- Git hosting services also offer automatic testing / code checking, etc. (CI – **continuous integration**)
- It is a good idea to configure git globally to **only allow fast-forward pulls**

```
[pull]                          ~/.gitconfig
    ff = only
```

Makes --ff-only the default for pulls

- Two strategies for **incorporating project changes from upstream**:
  - **Merging** (as explained here)
  - **Rebasing** (might be more complicated, but allows linear history)

**Further reading:**

- GitHub quickstart guide
- Any other GitHub / GitLab / Bitbucket tutorial
- Several projects have their own detailed Git-workflow guides (e.g. DFTB+ Git-workflow)