

Mastering git

Lesson 3 (working in teams)

Irina Gulina

Tomas Tomecek

No class on October 25th, but on November 1st?

September						
S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

October						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	×	26	27	28
29	30	31				

November						
S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

The first branch is created with the first commit

```
[Irina@localhost test_branch]$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/Irina/test_branch/.git/
[Irina@localhost test_branch]$ git branch
[Irina@localhost test_branch]$ git branch --all
[Irina@localhost test_branch]$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

```
[root@2fa20af94b59 asd]# ls -lha .git/refs
total 0
drwxr-xr-x. 1 root root 18 Sep 30 06:13 .
drwxr-xr-x. 1 root root 98 Sep 30 06:13 ..
drwxr-xr-x. 1 root root  0 Sep 30 06:13 heads
drwxr-xr-x. 1 root root  0 Sep 30 06:13 tags
[root@2fa20af94b59 asd]# ls -lha .git/refs/heads
total 0
drwxr-xr-x. 1 root root  0 Sep 30 06:13 .
drwxr-xr-x. 1 root root 18 Sep 30 06:13 ..
[root@2fa20af94b59 asd]# ls -lha .git/branches
total 0
drwxr-xr-x. 1 root root  0 Sep 30 06:13 .
drwxr-xr-x. 1 root root 98 Sep 30 06:13 ..
[root@2fa20af94b59 asd]# cat .git/HEAD
ref: refs/heads/master
```

Any questions or
suggestions?

We do have questions!

- ▶ What commands do you use to create a new branch?
- ▶ Describe a merge commit: what is it and why do we have such a thing?
- ▶ What is fast-forward?

Git Basic Commands

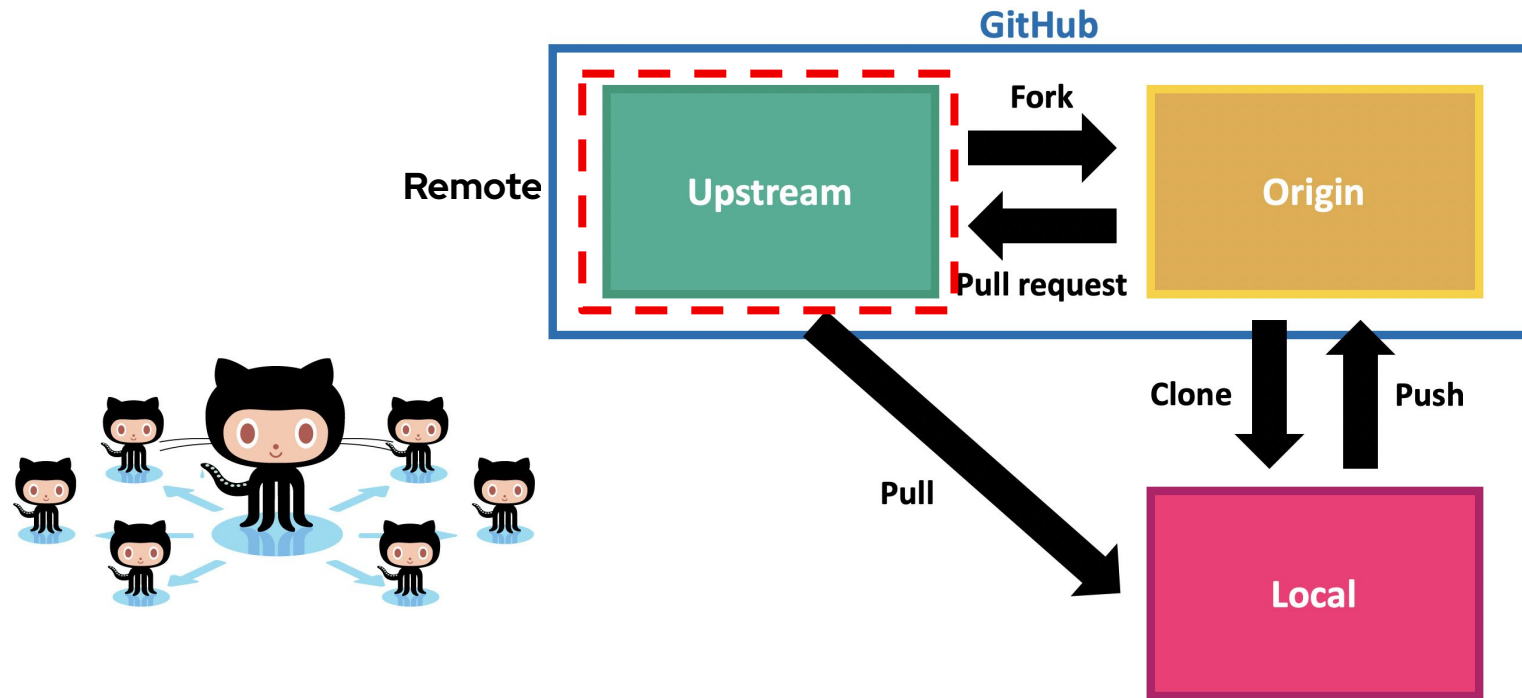
- ▶ **help**
- ▶ **init**
- ▶ **clone**
- ▶ **config**
- ▶ **add**
- ▶ **status**
- ▶ diff
- ▶ **commit**
- ▶ reset
- ▶ mv
- ▶ rm
- ▶ **branch**
- ▶ **switch**
- ▶ checkout
- ▶ **merge**
- ▶ log
- ▶ **stash**
- ▶ rebase
- ▶ restore
- ▶ **fetch**
- ▶ **pull**
- ▶ **push**
- ▶ **remote**

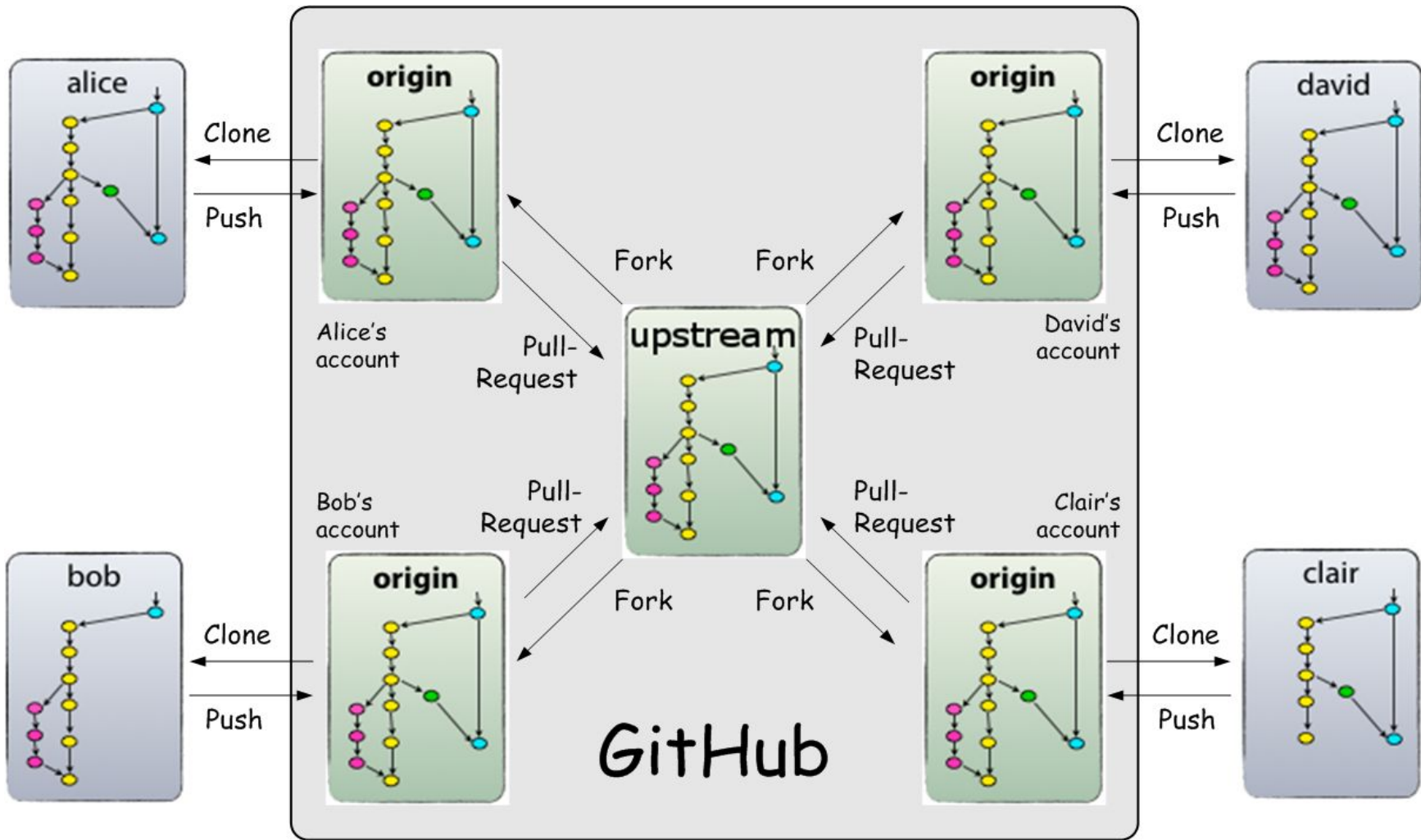
Today's class

- ▶ **Working as a team with a git repository**
- ▶ Deep dive into remotes.
- ▶ What's upstream and a fork?
- ▶ How to work with multiple remote repositories?
- ▶ Moving changes between remote repositories: push, pull and fetch
- ▶ How can I check out someone else's changes locally?
- ▶ The golden rule of push
- ▶ Tracking remote repositories.

- ▶ Labs: be comfortable with creating pull requests

Git workflow

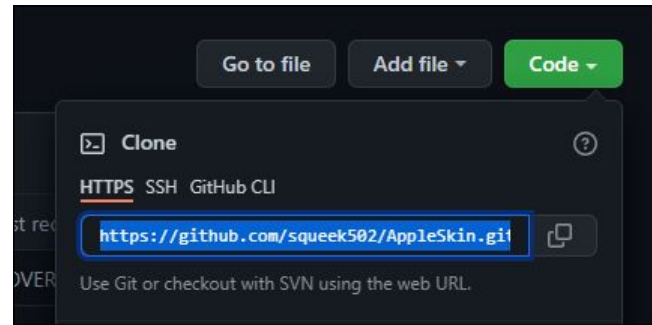
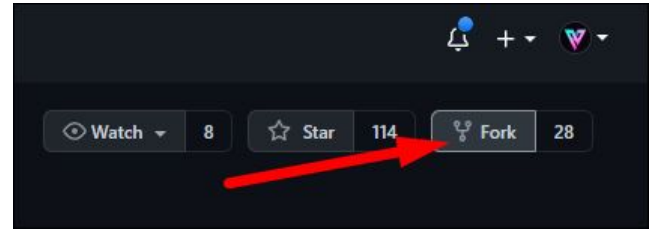




How to contribute to a team/community repo? FORK IT

How to fork

- Click Fork in a team repo
- Go to your personal Git space
- Git clone <ssh> to your local PC
- Add upstream



```
git remote add upstream https://github.com/upstream/project.git
```

Remote URLs

```
$ git config --list | grep remote
```

```
remote.origin.url=git@github.com:TomasTomecek/ansible-language-server.git
```

```
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
```

```
branch.main.remote=origin
```

```
remote.upstream.url=https://github.com/ansible/ansible-language-server.git
```

```
remote.upstream.fetch=+refs/heads/*:refs/remotes/upstream/*
```

How to work with a remote, syncing changes

```
git remote + (git fetch + git merge) or git pull + git push
```

git remote

Create, view and delete remote connections in `./.git/config`

`git remote`

`git remote -v`

`git remote show <name>`

`git remote add <name> <url>`

`git remote rm <name>`

`git remote rename <old-name> <new-name>`

`git remote set-url <name> <url>`

git remote -v

```
$ cd grit
$ git remote -v
bakkdoor https://github.com/bakkdoor/grit (fetch)
bakkdoor https://github.com/bakkdoor/grit (push)
cho45 https://github.com/cho45/grit (fetch)
cho45 https://github.com/cho45/grit (push)
defunkt https://github.com/defunkt/grit (fetch)
defunkt https://github.com/defunkt/grit (push)
koke git://github.com/koke/grit.git (fetch)
koke git://github.com/koke/grit.git (push)
origin git@github.com:mojombo/grit.git (fetch)
origin git@github.com:mojombo/grit.git (push)
```

git remote show origin

```
* remote origin
Fetch: origin http://someurl_git/application
Push: origin http://someurl_git/application
HEAD branch: main
Remote branches:
  Branch1 tracked
  develop tracked
  Branch2 tracked
Local branches configured for 'git pull':
  develop merges with remote develop
  main merges with remote main
Local refs configured for 'git push':
  develop pushes to develop (local out of date)
  main pushes to main (up to date)
```


git push

Write to remote connections defined in `./.git/config`

```
git push <remote_name> <branch_name>
```

git fetch

Download commits, files and refs from remote connections defined in `./.git/config`. **No merge!**

```
git fetch <remote>
```

```
git fetch <remote> <branch>
```

```
git fetch --all
```

```
git fetch --dry-run
```

git fetch

Download commits, files and refs from remote connections defined in `./.git/config`

`git remote + git fetch + git branch + git switch + git reset`

git push

Upload local repository content to a remote repository

```
git push <remote> <refspec>
```

```
refspec = <src>[:<dst>]
```

git push --force

You can force push on only to

- ▶ Local branch which is not upstream yet
- ▶ Unmerged PR

You can't force push to upstream public branches.

git pull

fetch and download content from a remote repository and immediately update the local repository to match that content, i.e. git pull = git fetch + git merge

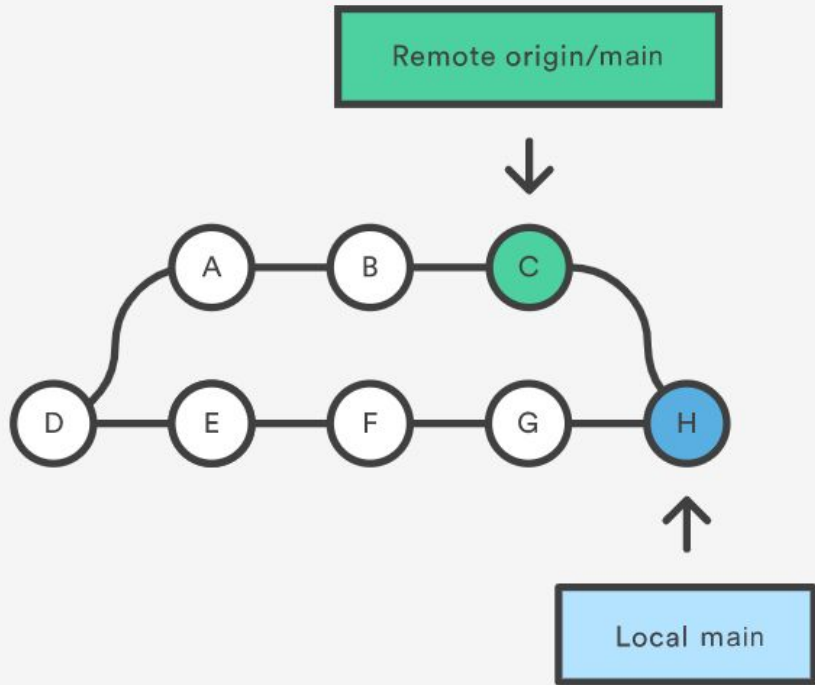
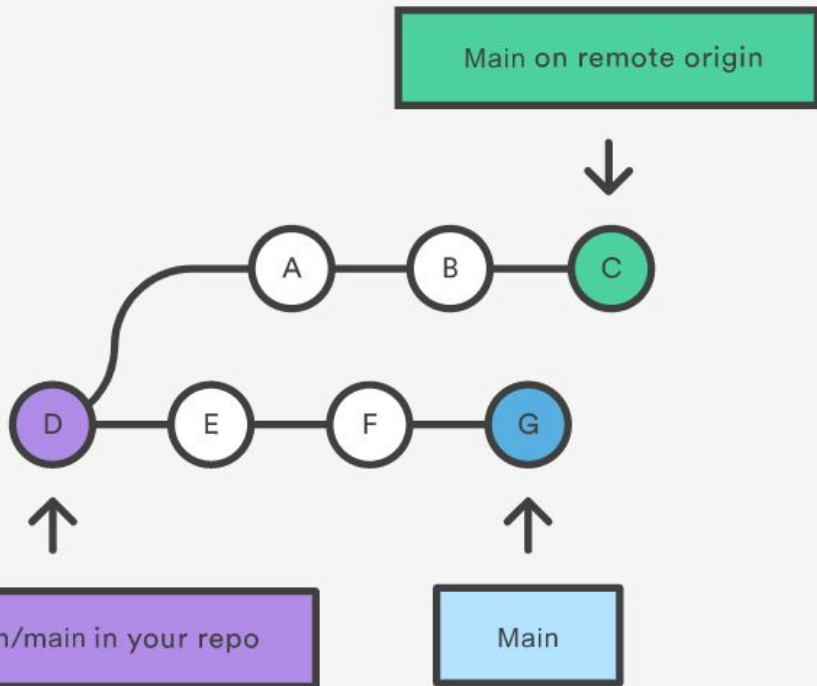
```
git pull <remote>      = git fetch <remote> && git merge origin/<current_branch>
```

```
git pull --no-commit <remote>
```

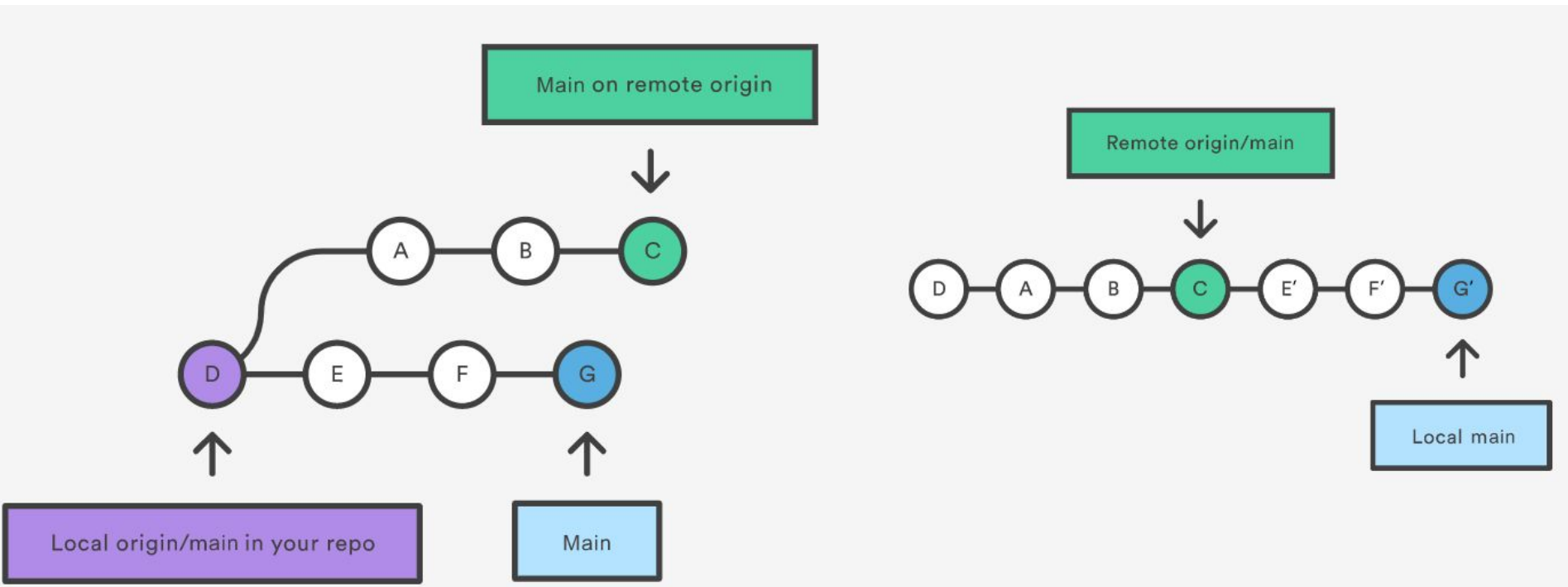
```
git pull --verbose
```

```
git pull --rebase <remote>
```

Before -> git pull -> After

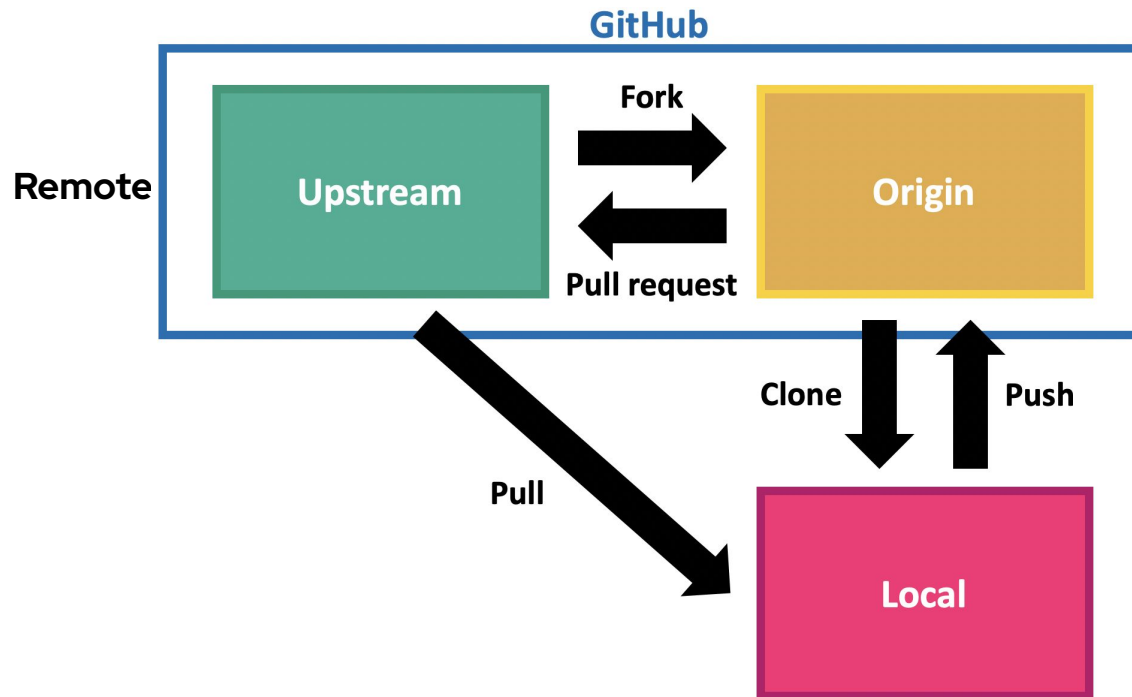


Before -> git pull --rebase -> After



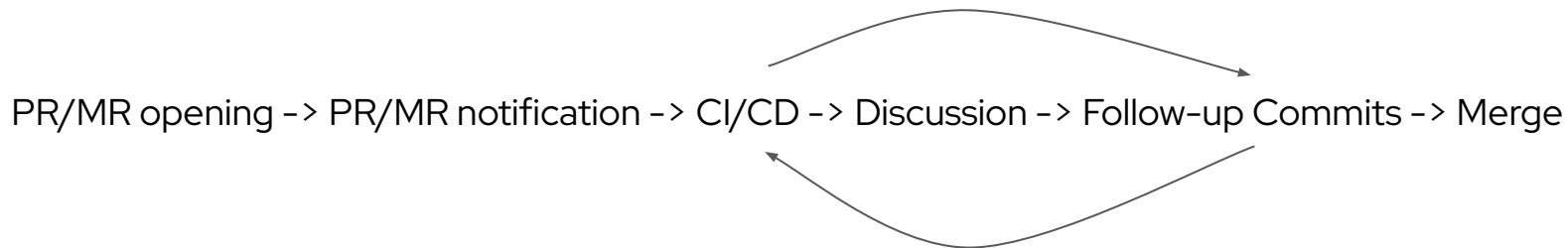
git fetch vs git pull

Git workflow



Pull/Merge request

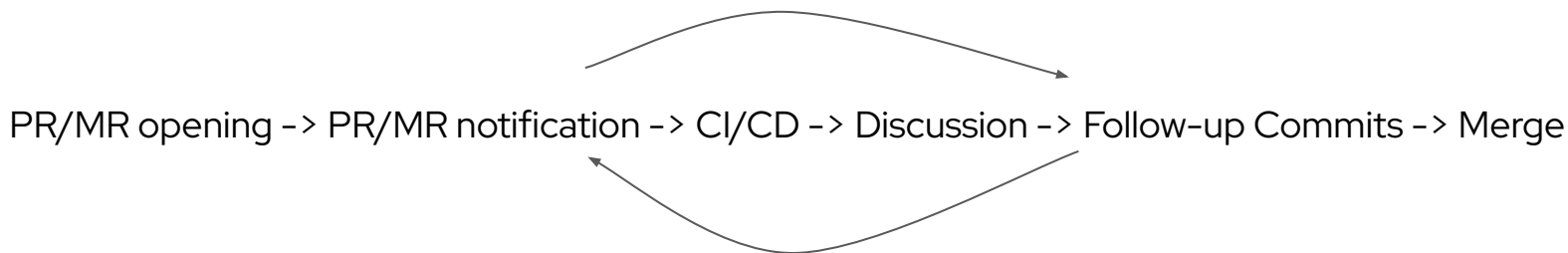
A user-friendly way for discussing changes before integrating them into the official project.



Why do we use PR/MR workflow?

- Share changes
- Get review and feedback
- Encourage quality
- Get better at coding

PR/MR workflow (simplified version)



- Create a dedicated branch in a local repo with the code change
- Push that branch to the remote
- File a PR
- Review a PR
- Merge a PR

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 > Repository



✔ You pushed to `my_test` just now

Create merge request

Ansible_Design_...

rhsap /

+ ▾

History

Find file


Web IDE

↓ ▾

Clone ▾




Creating a 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

[Create merge request](#)

 **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



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
<input type="text" value="igulina/rhsap"/> <input type="text" value="my_test"/>	<input type="text" value="ccsp-sap/rhsap"/> <input type="text" value="main"/>
 add empty_file Irina authored 5 minutes ago	<input type="text" value="095ae6aa"/> 
<input type="button" value="Compare branches and continue"/>	

What constitutes a good PR?

- Complete piece of work about **ONE** logical change
- Adds value in some way
- Solid title and body
- Clear commit history
- Small
- Meets project's contribution guidelines

Creating a PR from main vs. a feature branch

- Clarity - “main” is not descriptive
 - “fix-1338” tells a better picture what’s inside
- You cannot have 2 pull requests created from the same branch at the same time
- You can work in parallel on 2 branches at the same time
- `git switch -c my-feature-branch`
 - is just *one* extra command for all 3 benefits

Test your knowledge now!

Class 3 lab

- ▶ <https://gitlab.com/redhat/research/mastering-git/-/blob/main/labs/lab3.md>

THE END

Questions?

Class 3 homework

- ▶ <https://gitlab.com/redhat/research/mastering-git#class-3-homework>

THE END

Questions?

THE END

THANK YOU!