



Git – Part 2

CS255: Programming Lab – Tutorial

Spring Semester 2021

Giorgos Pantelakis – csd4017@csd.uoc.gr

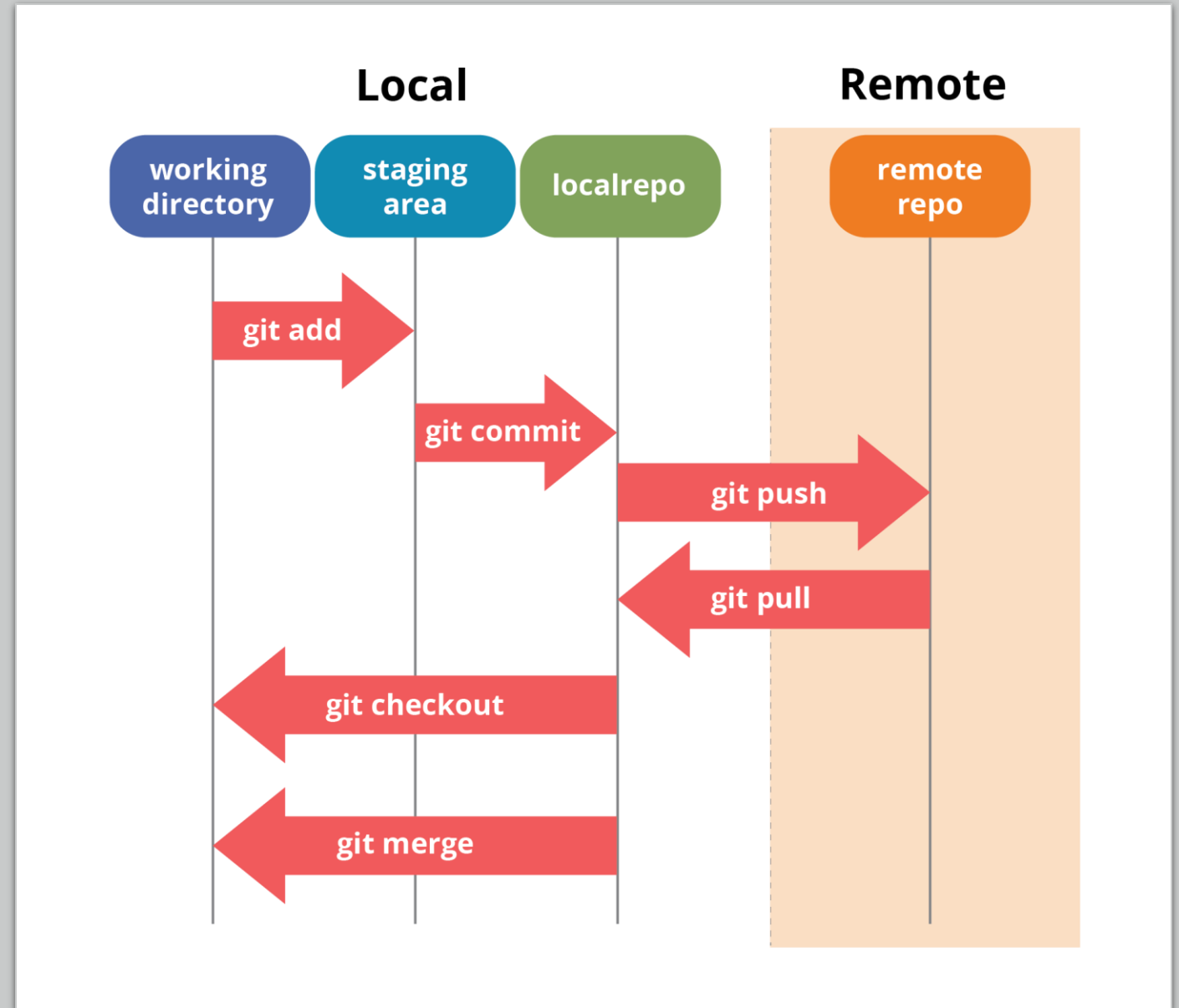
Michalis Vardoulakis – mvard@csd.uoc.gr

The purpose of this tutorial

World Domination!!!

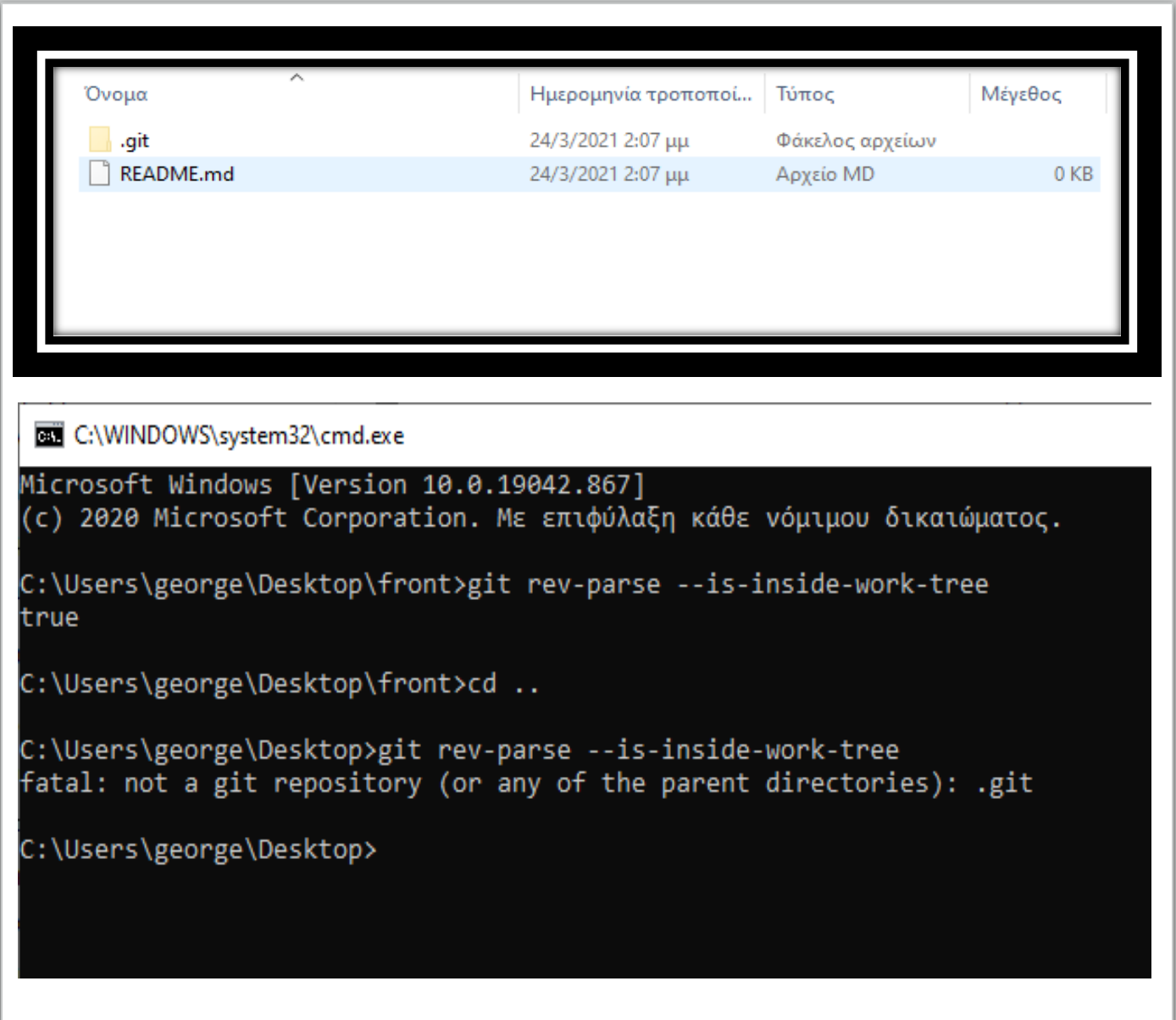
The purpose of this tutorial

- Git pull
- Merging and branching in git
- Git flow
- Answer all your questions!



How can I see if a folder is a git repo?

- Hidden folder .git
- with the command "git rev-parse --is-inside-work-tree"
 - If it is a git repo it will return true
 - Else it will print fatal: not a git repository



How can I see the source of my git repo;

I use the command '**git** config --get **remote.origin.url**'


```
C:\WINDOWS\system32\cmd.exe
```

```
Microsoft Windows [Version 10.0.19042.867]
```

```
(c) 2020 Microsoft Corporation. Με επιφύλαξη κάθε νόμιμου δικαιώματος.
```

```
C:\Users\george\Desktop\assignment0>git config --get remote.origin.url  
https://gitlab-csd.datacenter.uoc.gr/hy-255/assignment0.git
```

```
C:\Users\george\Desktop\assignment0>
```



In this section your username should appear. If it says 'hy-255' you have cloned the wrong repo and it will throw 'access denied' error on push...

What does "pulling" mean?

- The pull command changes from the remote git repo to our local copy
- Good practice: in a group assignment always pull before you start working
- Attention points:
 - Before a pull, you must already have cloned the repo
 - You need to commit or stash your local changes before you pull
 - You need to be a project member to pull
 - For more info: <https://git-scm.com/docs/git-pull> or `man git pull`

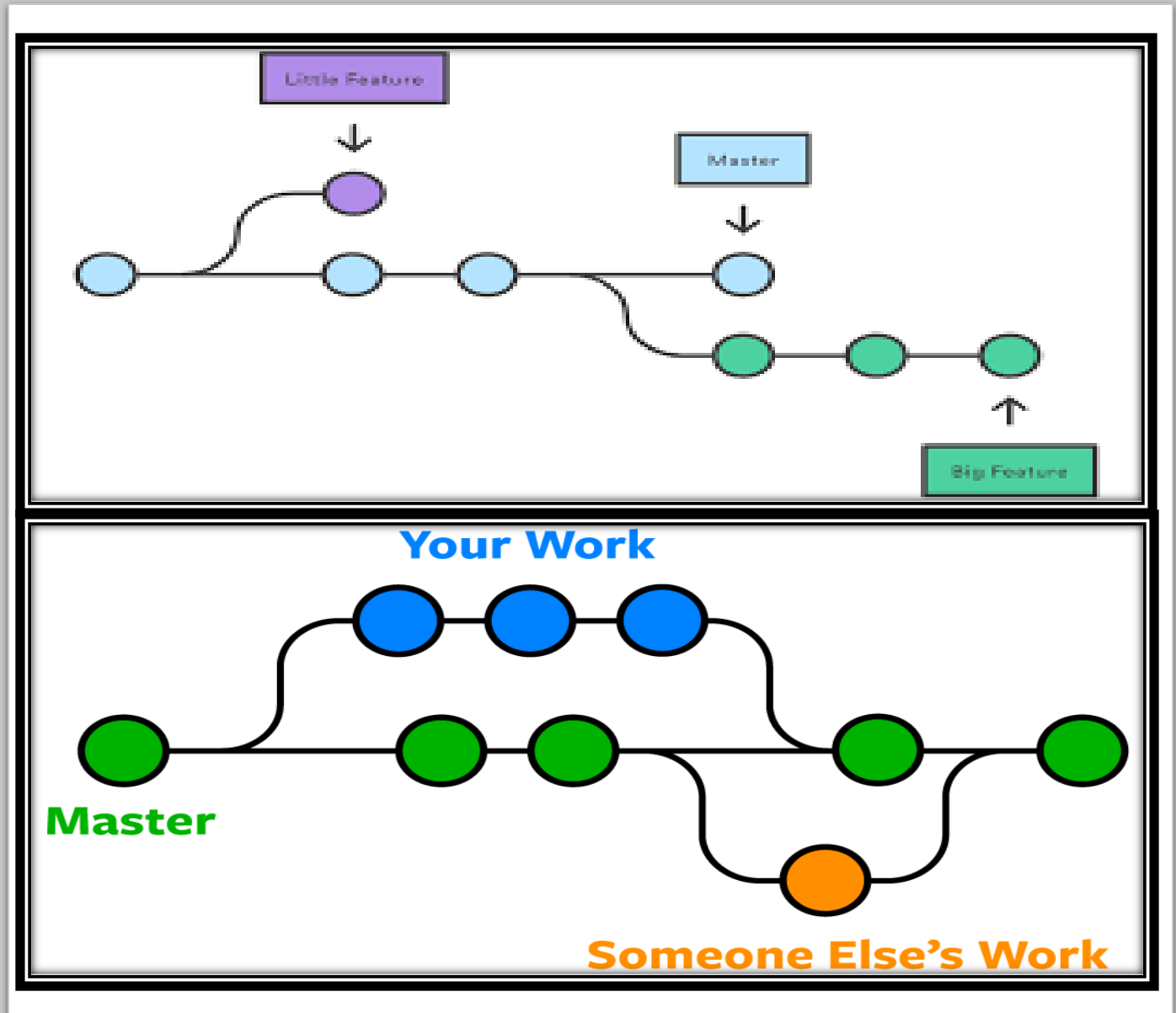


The time of truth!!!

Branching...

What is and where do I use branching?

- A branch represents an independent line of development
- Commits are performed separately in each branch; you can develop a feature without affecting your master branch
- Branch benefits:
 - "context switch" for concurrent tasks on the same project, for example fixing a bug and developing a new feature
 - experiment with our project and commit changes without impacting our users or other contributors



git branch <branch_name>

- Creating a new branch locally that is called branch_name.

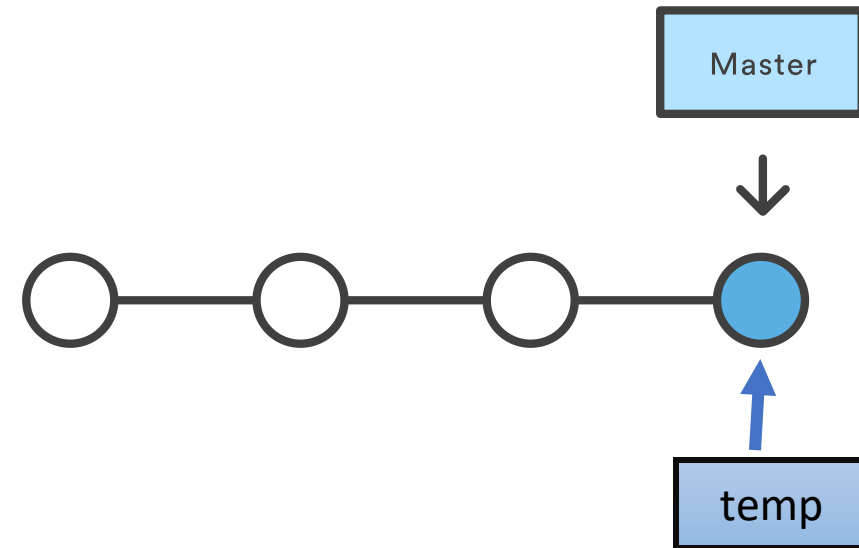
git push <remote-name> <branch-name>

- It pushes the branch called branch_name to the remote repo.

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. Με επιφύλαξη κάθε νόμιμου δικαιώματος.

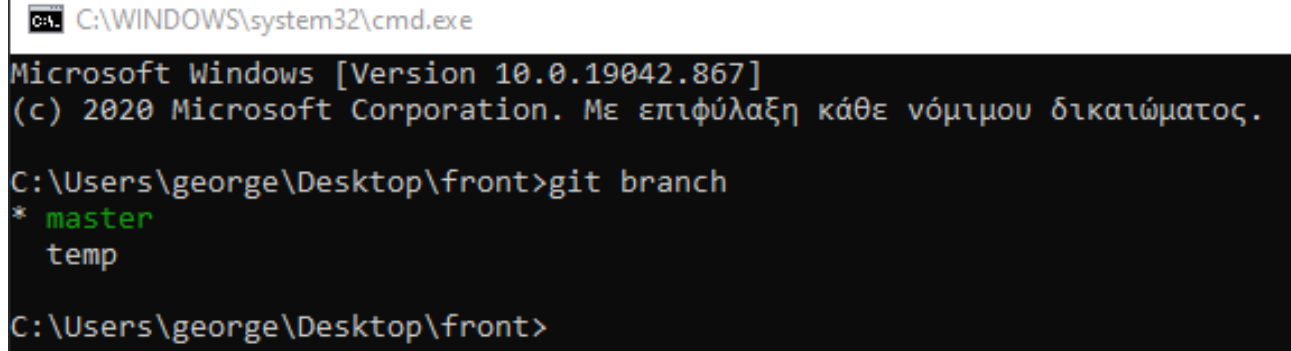
C:\Users\george\Desktop\front>git branch temp

C:\Users\george\Desktop\front>
```



git branch

- Returns a list with all the branches



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. Με επιφύλαξη κάθε νόμιμου δικαιώματος.

C:\Users\george\Desktop\front>git branch
* master
  temp

C:\Users\george\Desktop\front>
```

git checkout <branch_name>

- Changing our working branch to the <branch_name> branch

C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. Με επιφύλαξη κάθε νόμιμου δικαιώματος.

C:\Users\george\Desktop\front>git checkout temp
Switched to branch 'temp'

C:\Users\george\Desktop\front>git branch
  master
* temp

C:\Users\george\Desktop\front>
```

How do I delete a branch?

`git branch -d <branch-name>`

- Delete a branch that is called `branch_name`
- If you want to force delete it use `git branch -D <branch-name>`

`git push origin --delete <branch-name>`

- It deletes the branch called `branch_name` from the remote repo

Merge Conflicts [1/5]

When collaborating on a project, sometimes git might not be able to merge our changes with those of our collaborators. This is called a merge conflict

```
$ git add README.md
$ git commit -m "update my readme again"
[master d2c862d] update my readme again
1 file changed, 1 insertion(+)
$ git push
To gitlab.com:mvard/my-first-git-project.git
! [rejected]          master -> master (fetch
first)
error: failed to push some refs to
'git@gitlab.com:mvard/my-first-git-project.git'
hint: Updates were rejected because the remote
contains work that you do
hint: not have locally. This is usually caused by
another repository pushing
hint: to the same ref. You may want to first
integrate the remote changes
hint: (e.g., 'git pull ...') before pushing
again.
hint: See the 'Note about fast-forwards' in 'git
push --help' for details.
```

Merge Conflicts [2/5]

Let's run a git status:

```
$ git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each,
respectively.
    (use "git pull" to merge the remote branch into
    yours)

You have unmerged paths.
    (fix conflicts and run "git commit")
    (use "git merge --abort" to abort the merge)

Unmerged paths:
    (use "git add <file>..." to mark resolution)
    both modified:   README.md

no changes added to commit (use "git add" and/or
"git commit -a")
```

Merge Conflicts [3/5]

- There are remote changes that are not incorporated in our local clone
- First, we need to pull those changes:

```
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0),
pack-reused 0
Unpacking objects: 100% (3/3), 291 bytes | 145.00
KiB/s, done.
From gitlab.com:mvard/my-first-git-project
    d115c2f..336d42c  master    -> origin/master
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then
commit the result.
```

Merge Conflicts [4/5]

- In cases where two commits edit the same line of a file, we must manually merge these changes
- Let's open the file to see what's wrong
- We can edit the file to remove the marks added by git in lines 2, 4 and 6 and to decide what change we want to keep

```
1. This is my first git project!
2. <<<<<< HEAD
3. This is the second line of the
   readme          This is our change
4. =====
5. NO! This is the second line of the
   readme          This is the change we pulled
6. >>>>>>
   336d42c199933c431b7b3605c16f6e85ddbc9720
```

Merge Conflicts [5/5]

- Let's keep our own change. The new contents of the file are:

```
1. This is my first git project!  
2. This is the second line of the readme
```

- We now need to add this new change and create a merge commit to finalize this merge

```
$ git add README.md  
$ git commit
```

- We can edit the commit message in the open editor and then save and close to commit
- This commit resolves our merge conflict

How to best put our new-found knowledge to practice?

Git Feature Branch workflow

Git Feature Branch Workflow

- All feature development happens in dedicated feature branches
- Eases collaboration between team members
- Master branch is always in a stable state
- Works well with pull/merge requests

Pull or merge request:

It's called pull request in GitHub or merge request in GitLab. It's a tool that allows you to share a changeset (i.e commits) with your collaborators and receive feedback on them before merging them to another branch (usually the master branch)

How it works

- Leverages a central repository (GitLab and GitHub are exactly that)
- Start a new branch to work on something specific – be it a new feature or an issue
- You can push your code to the central repository as often as you like since it won't affect any of your collaborators
- You merge your branch into master once you are done – usually through a pull/merge request but can be done using the `git merge` command locally

How to use this workflow in practise? [1/2]

Step 0: Make sure you are in the latest commit of master branch

```
$ git checkout master  
$ git pull
```

Step 1: Create a new branch

```
$ git checkout -b symbol-table-list
```

Step 2: Write some code

Step 3: Commit said code

```
$ git status  
$ git add list.c list.h  
$ git commit -m "Implemented linked list"
```

How to use this workflow in practise? [2/2]

Step 5: Publish changes to central repository

```
$ git push -u origin symbol-table-list
```

Step 6: Merge changes to master branch

- You can either initiate a pull/merge request from the web UI of GitHub/GitLab (we won't cover this in this tutorial)
- Or perform the merge from your terminal

```
$ git checkout master
```

```
$ git pull
```

```
$ git merge symbol-table-list
```

```
$ git push
```

Further reading material

- Git Feature Branch Workflow:
<https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow>
- Gitflow Workflow:
<https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>

Thanks for your attention!
Any questions?

