Labs

Git training – Lab 1
Basics

Lab 1,1 – basics

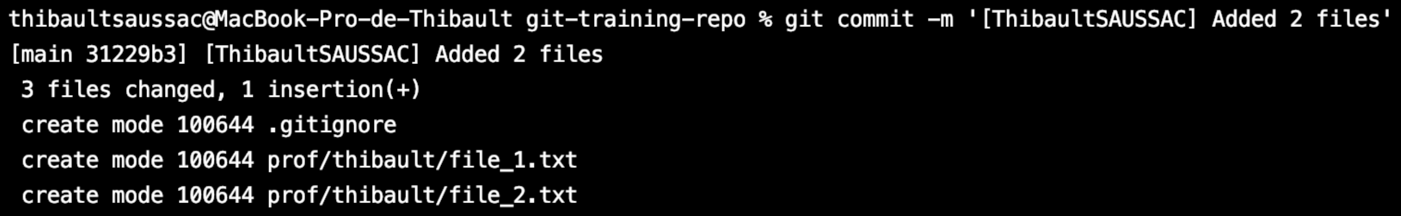
1. *Open Git Bash*
2. *Create “Hello” directory and change into it*
3. *Use init command to create a git repository in that directory :*
   - *Observe that a .git directory is created*
4. *Git configuration*
   - Your identity
     - git config --global user.name “Thibault Saussac”
     - git config --global user.email thibault.saussac@orange.com
   - *List all configuration*
     - *git config --list*

Lab 1,1 – first commit

1. *Create file1.txt*
   - *Observe the output of git status. file1.txt is on the untracked area*
   - *Observe also the help proposed by git status*
2. *Use add command to add the file to the staged area*
   - *Use status command to confirm the staging success*

Lab 1,1 – basics - first commit

3. Use commit command to commit the content of the staged area
   - Observe the commit creation message:



```
thibaultsaussac@MacBook-Pro-de-Thibault git-training-repo % git commit -m '[ThibaultSAUSSAC] Added 2 files'
[main 31229b3] [ThibaultSAUSSAC] Added 2 files
 3 files changed, 1 insertion(+)
 create mode 100644 .gitignore
 create mode 100644 prof/thibault/file_1.txt
 create mode 100644 prof/thibault/file_2.txt
```

- Which branch you committed to
- What SHA-1 checksum the commit has (d9151ed)
- How many files were changed
- Statistics about lines added and remove

Lab 1,2 – basics

1. Make a change to file1.txt
2. Use diff command to view changes details
3. Use status command to see the working repository situation
   - file1.txt is modified and not staged
4. add the file to the staged area, confirm using the status command
5. Commit the modifications
6. Use commit --amend to modify last commit message
7. Modify again file1.txt and add it to the staged are
8. Use commit --amend to append those modification to the last commit

Lab 1,2 – basics

1. Use the log command to see all the commits you made

2. Use the show command to see a commit detail

3. Modify file1.txt
   - Check status ;
     - Observe the help proposed by git
   - use the command checkout -- to undo the modification
   - Check status

4. Modify again file1.txt
   - Check status ; add it to staged area ; check status
     - Observe that in git status command , Git tells you how to unstage a file
   - Use the command reset to unstage the file
   - Check status ; use the checkout-- command to undo the modification

Lab 1,3 – basics – remote

1. Exit the current git directory
2. Use Clone command to clone the remote project :
    1. https://github.com/saussact/git-training-repo.git
3. Move into the new clone project
4. Create a new commit including following activity
    - Into Users folder, create a folder with your name and create two new files into it
        - users/yourName/
    - Create two files (file_1.txt & file_2.txt) into that folder
    - Use add command to add your folder (use the folder name)
        - Use status command to notice that all the folder content is staged
    - Commit with following message “[nom-prenom]: add file 1 & 2”

Lab 1,3 – basics – remote

1. Use Fetch remote command to get the repo modification
   - Use log command with following options to observe the local repo updates
     - git log --graph --oneline --all --decorate
   - Add the previous command as an alias and test it
     - git config --global alias.lg "log --graph --oneline --all --decorate“
     - git lg
2. Use Pull remote to update your local repo
   - Use log command to observe the local repo updates
     - use git lg alias
3. Use Push command to push your commit to remote
   - Confirm using gitk --all command

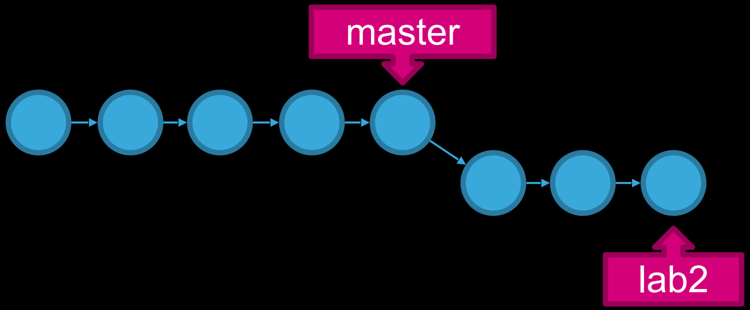
Git training – Lab 2
Branching

Lab 2,1 – Branching (1)

1. Use branch command to create a new branch named *lab2-<yourname>*
   - Use branch -a command to list your branches
     - remotes/origin/main : is the branch master on the remote
2. Use checkout command to switch to the new branch
   - Use status command to confirm the switch
3. Create two new commit including the following activity
   - Into users/youName
     - First commit : Modify file_2.txt ; commit
     - Second commit Create file_3.txt ; commit
4. Use checkout command to switch back to main branch

Lab 2,1 – merging ; fast forward (2)

1. Use the merge command to merge the lab2 branch work
   - Observe the **Fast-forward** merge message
   - Confirm the merge success using the log command
   - Observe the merge success using gitk command

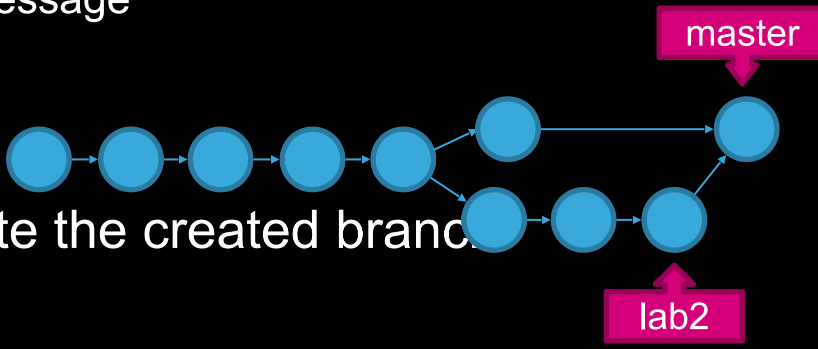


2. Use status command to ensure that your working directory is clean
   - Observe the message “your branch is ahead of 'origin/main' by 2 commits”

```
thibaultsaussac@MacBook-Pro-de-Thibault git-training-repo % git status
Sur la branche main
Votre branche est en avance sur 'origin/main' de 2 commits.
  (utilisez "git push" pour publier vos commits locaux)
```

Lab 2,2 – merging ; merge commit

1. Use checkout command to switch to lab2 branch
   - Use status command to confirm the switch
2. Create one new commit including the following activity
   - Into users/yourName: : Modify file_2.txt ; commit
3. Use checkout command to switch back to main branch
4. On branch main one commit including the following activity
   - Into users/youName: First commit : modify file1.txt
5. Use the merge command to merge the lab2 branch work
   - The editor is opened: enter merge commit message
   - Observe the merge commit creation
6. Use the branch -D command to delete the created branch
   - Confirm with the command branch –a



Git training – Lab 3
Merge conflicts

Lab 3 – Merge Conflicts

1. Use checkout -b command to create a branch named *lab3-<yourName>*
   - Use status command to confirm branch creation and switch
2. Create two new commit including the following activity
   - Into users/yourName
     - First commit : modify file1.txt ; commit
     - Second commit modify again file1.txt ; commit
3. Use checkout command to switch back to main branch
4. Create one new commit including the following activity
   - Into users/yourName
     - First commit : modify file1.txt ; commit

Lab 3 – Merge Conflicts

5. Use the merge command to merge the work done in lab3 branch
   - Notice the git merge conflict message
   - Use git status command to see the files concerned by the merge conflict

6. Let's resolve the conflict
   - Open file1.txt
     - Local changes between **<<<<< HEAD** and **=======**
     - Remote changes **=======** and **>>>>> branchName**
   - Edit the file and fix the resolution then save

7. Use add command to confirm the merge conflict resolution

8. Use status command to view the merge status
   - Notice the message: all conflict fixed but you are still merging

Lab 3 – Merge Conflicts

9. Use commit to conclude merge
   - Notice that conflict resolution is done through a new commit : the merge commit
10. Use log commands to confirm the merge

Git training – Lab 4
Rebasing

Lab 4 – Rebase

1. Use checkout -b command to create a branch named *lab4-<yourName>*
   - Use status command to confirm branch creation and the switch to
2. Create two new commits including the following activity
   - Into users/yourName
     - First commit : modify file1.txt ; commit
     - Second commit modify again file1.txt ; commit
3. Checkout to main branch and create one commit:
   - Into users/yourName
     - First commit : Create a new file
4. Checkout lab4 branch
5. Use rebase command to rebase lab4 branch content with main
   - Use log command to view rebase operation effect

Git training – Lab 5
Workflow

Lab 5 – workflow (1)

- Clone the workflow repository
    - git clone https://gitlab.forge.orange-labs.fr/hpnt9572/workflow.git
- Let's say your task name is "*issue #1 : implement feature 1*"
    1. Checkout a new task branch name with the task id and a short descriptive title
        - git checkout -b issue1-implement-feature-<yourName>
            - The ID to easily associate the track with its tracker
            - The description if for a human little hint on what's in it
    2. Do you work on this branch
        - Into users/<yourName> : Perform **four** Commits of your choose (change 1, change 2..)
    3. use interactive rebase to squash all commits together
        - git rebase -i main

Lab 5 – workflow (2)

6. git will display an editor window with lists of your commits
        - pick 05dd574 file1
        - pick 2950a15 file3
        - pick 88a13e7 file4
        - pick 26ec678 file2
    - Now we tell git what we want to do (squash)
        - pick 05dd574 file1
        - squash 2950a15 file3
        - squash 88a13e7 file4
        - squash 26ec678 file2
    - Save and close the file
        - This will squash all commit together into one commit

- Git displays a new editor where we can give the new commit a clear message
    - Message must be written on the first line (lines after are commit message details)
    - We will use the task ID and tile : **issue #01 : implement feature 1**
    - Save and close the editor

Lab 5 – workflow (3)

7. Merge your changes back into master
   - git checkout main
   - git merge issue1-implement-feature-<yourname>
     - It must be a fast-forward merge
8. Finally push your change to upstream
   - If, meanwhile origin is updated do:
     - git fetch origin
     - git rebase origin/main
9. Use gitk --all to observe the result

(BONUS) Lab 6 – gitflow (3)

- Play with gitflow
  - Perform a release
  - Make a feature branch
  - Perform an hotfix
  - Create branch