Як працювати з репозиторієм (GitHub)

1) створення репозиторія для командного проєкту

2) налаштування репозиторію  
 3) розробка фічі або багфікс  
 4) видалення гілок

5) вирішення git-конфліктів

6) список команд для роботи з терміналом

Як Ви пам’ятаєте, **Git** – це програмне забезпечення для контролю версій, яке зробить співпрацю з колегами по команді дуже простою.

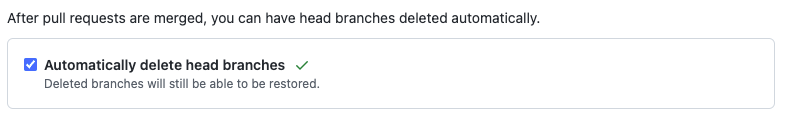
Тому розпочнемо саме з

1. **створення репозиторія для командного проєкту**

**Тімлід** створює **пустий публічний** репозиторій для проєкту **на своєму** GitHub-акаунті

1. **налаштування репозиторію**  
   На сторінці вже створеного репозиторію **Тімлід** заходить до **Settings**  та вносить наступні зміни

* на вкладці **General** у секції **Pull Requests** ставить чек поряд з **Automatically delete head branches**

****

* У вкладці **Branches** у розділі **Branch protection rules** встановлює правила для гілки **main**
* **Тімлід** у вкладці **Collaborators** додає членів команди в репозиторій. Кожному буде надіслано інвайт на пошту.
* після чого усі розробники з команди клонують репозиторій, створений **Тімлідом*.***

1. **розробка фічі або багфікс**

* кожна фіча або багфікс виконується окремою гілкою.
* перед початком роботи над новою фічею або багфіксом оновлюємо локальний **main**, щоб він був ідентичний **main на GitHub**. Для цього переходимо на гілку **main** і стягуємо останні зміни, щоб синхронізувати локальну гілку з тією, що на **GitHub.**
* робимо нову гілку з **main.** Рекомендується давати назву гілки за шаблоном **feature/назва-фічі** та **bugfix/назва-бага**.
* працюємо у цій гілці та вносимо необхідні зміни. Комітимо зміни за потребою.
* після того, як фіча готова, відправляємо історію (комміти) локальної гілки на GitHub і відкриваємо **Pull request.**
* призначаємо мінімум одного **Reviewer,** щоб зробив ревью коду.
* якщо необхідне редагування, робимо його у тій самій гілці і відправляємо зміни на **GitHub.** Поточний **Pull request** оновиться автоматично, відкривати новий не треба.
* після того як **Pull request** пройшов ревю коду, **тімлід** приймає його (мерже у гілку **main**).
* після того, як **Pull request** був прийнятий, розробник переходить на гілку **main** і стягує останні зміни,щоб синхронізувати локальну гілку з тією, що на **GitHub.**
* після синхронізації гілки **main** розробник видаляє свою локальну гілку фічі/бага в GitHub Desktop.
* після злиття коду в **main,** проєкт повинен успішно компілюватися та збиратися. Ситуація, коли **Pull request** прийнятий, злитий, а проєкт не збирається - epic fail
* забороняється безпосередньо заливати код у гілку **main**

**4) видалення гілок**

щоб видалити гілку через **термінал**

*git branch -d <назва\_гілки>* - видалити гілку

*git push origin --delete <назва\_гілки>* - видалити гілку на віддаленому репозиторії

**5) вирішення git-конфліктів**

Конфлікти відбуваються коли два розробники одночасно змінюють один і той же рядок в одному тому самому файлі. Виникають під час **Pull Request**, через що він не може бути успішно cmergeний у базову гілку. Конфлікти бажано вирішувати локально, а не на GitHub. Для цього

у **терміналі** *git checkout main*

*git pull*

*git checkout ‘назва-вашої-гілки’*

*git merge main*

Вирішуємо конфлікт і робимо коміт. Після чого відправляємо зміни на GitHub, щоб оновити **Pull Request**.

**6) список команд для роботи з терміналом**

**git clone https://посилання\_на\_репозиторій** клонуєте репозиторій

**git checkout ‘назва-гілки’** перехід на гілку ‘назва-гілки’

**git pull origin ‘назва-гілки’** стягує останні змінигілки ‘назва-гілки’

**git checkout -b ‘назва-нової-гілки’** створює нову гілку

**git add .** та **git commit -m “назва-коміта”** записує зміни до локальної гілки

**git push origin ‘назва-вашої-гілки’** публікує ваші зміни на гілці віддаленого репозиторію

**git merge ‘назва-гілки’** залиття коду з гілки ‘назва-гілки’ на поточну гілку

**git branch -d ‘назва-гілки’** видалення гілки ‘назва-гілки’

**git fetch** стягує всі зміни з віддаленого репозиторію

**git push origin –delete –** видалення гілки на GitHub

## Список корисних команд для роботи з git

1. git clone - копіює репозиторій(папку з файлами) на компютер
2. git branch "branchName" - створює гілку з назвою "branchName"
3. git checkout "branchName" - переходить на гілку з назвою "branchName"
4. git checkout -b "branchName" - створюємо гілку з назвою "branchName" і переходимо на неї
5. git add . - зберігаємо зміни в файлах
6. git commit -m "commit message" - підписуємо збереженні зміни в файлах
7. git push - відправляємо зміни на сайт github
8. git pull - отримуємо останні зміни з сайту github
9. git status - показуємо статус проекту
10. git branch - показуємо список гілок в проекті
11. git branch -r - показуємо список гілок на сайті github
12. git branch -a - показуємо список гілок на компютері та на сайті github
13. git fetch - отримуємо зміни з сайту github
14. git stash - зберігаємо не збережені зміни в файлах і кладемо їх в буфер обміну
15. git stash apply - вставляємо збережені зміни з буфера обміну
16. git merge "banchName" - зливаємо гілку з назвою "branchName" в поточну гілку
17. git merge --abort - відміняємо зливання гілок
18. git branch -d branchName - видаляє гілку локально з проекту
19. git push origin --delete name - видаляє гілку з сайту github
20. git diff - показує відрізки рядків між двома версіями файлу (між двома комітами)
21. git log - показує історію комітів
22. cd gitTutorial - переходимо в папку gitTutorial

Для того щоб вийти з режиму перегляду комітів використовуйте клавішу q (стосується команди №20 і №21)