Гілки GIT

HTTPS://GIT-SCM.COM/

ПІДРУЧНИК

HANDBOOK

Створення нової гілки

Щоб створити нову гілку на поточному коміті та переключитися на неї треба виконати команди

```
$ git branch new_branch
$ git checkout new_branch
```

Тут `new_branch` назва нової гілки, яку задає програміст. Це саме виконає команда `checkout` з параметром `-b` одразу:

```
$ git checkout -b new_branch
```

Детальніше...

Видалення непотрібної гілки

Для видалення гілки використовується команда

```
$ git branch -d new_branch
```

Для видалення гілки, необхідно переключитися на іншу гілку. Треба звернути увагу, що ця команда спрацює, якщо гілка `new_branch` повністю замерджена в поточну гілку. Інакше Git відмовить у видаленні.
Якщо треба видалити гілку незалежно від того чи замерджена в поточку гілку, використовують цю ж команду, проте з параметром `-D` (велика літера D)

```
$ git branch -D new_branch
```

Звернемо увагу, що в такому разі є ризик втратити дані, адже ланцюг комітів, на якій не вказує жодна гілка буде невдовзі видалений системою контролю версій

Детальніше...

Перегляд гілок

Команда `branch` без параметрів виводить список гілок репозиторію:

```
$ git branch
```

Якщо додати до команди `branch` параметр -v

```
$ git branch -v
```

гілки будуть виведені з останніми комітами на кожній з них

Детальніше...

Перейменування гілок

Для перейменування поточної гілки використовується команда branch з параметром -m:

```
$ git branch -m <new name>
```

де <new name> нове ім’я поточної гілки. Наприклад

```
$ git branch -m develop
```

Зливання гілок

Ідея об’єднання двох гілок, полягає в тому, що ми маємо в першу гілку злити зміни з другої гілки, після чого другу гілку можна видалити (при необхідності). Припустимо, ми хочемо об’єднати гілки `main` та develop, тобто у гілку `main` влити зміни з гілки develop. Для цього треба переключитися на гілку `main`

```
$ git checkout main
```

та виконати команду

```
$ git merge develop
```

У результаті такої операції Git автоматично створює новий коміт, який називають комітом злиття (merge commit) та його особливістю є те, що він має більше одного батьківського коміту.

Детальніше...

Основи конфліктів зливання

Трапляється, що цей процес не проходить гладко. Якщо ви маєте зміни в одному й тому самому місці в двох різних гілках, Git не зможе їх просто злити. В результаті злиття двох таких гілок буде отримано конфлікт злиття (merge conflict).

У цьому випадку Git не створив автоматичний коміт зливання. Він призупинив процес допоки ви не вирішите конфлікт.

```
$ git merge develop
Auto-merging main.py
CONFLICT (content): Merge conflict in main.py
Automatic merge failed; fix conflicts and then commit the result.
```

Детальніше...

Основи конфліктів зливання

Щоб переглянути які саме файли спричинили конфлікт треба виконати команду `git status`
Все, що має конфлікти, які не були вирішені є в списку незлитих (unmerged) файлів.

```
$ git status
On branch main
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:   main.py

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

Детальніше...

Основи конфліктів зливання

У кожен такий файл, що містить конфлікт, Git додає стандартні позначки-вирішенння для конфліктів. Якщо відкрити такий файл за допомогою текстового редактора, то побачимо

```
<<<<<<< HEAD
q = input("Hello")
=======
print("Hello, world!")
>>>>>>> develop
```

Область між позначками
`<<<<<<< HEAD` та `>>>>>>> develop` це зона конфлікту злиття поточної гілки main (позначається HEAD) з гілкою у develop нашому випадку.
Все, що знаходиться між `<<<<<<< HEAD` та `=======` це те, що міститься на гілці main.
Те, що знаходиться між `=======` та `>>>>>>> develop` – на гілці develop.

Для вирішення конфлікту, задача розробника обрати правильні фрагменти коду, видалити створені Git позначки та створити коміт вирішення конфлікту.

Детальніше...

Основи конфліктів зливання

Для вирішення конфлікту, задача розробника обрати правильні фрагменти коду, видалити створені Git позначки (нижче ми обрати код з обох секцій конфлікту зображеного на попередньому слайді)

```
q = input("Hello")
print("Hello, world!")
```

додати всі зміни в індекс (щоб показати для Git, що конфлікт вирішено.

```
$ git add .
```

та створити коміт вирішення конфлікту.

```
$ git commit
```

Детальніше...