Основи роботи з Git

1. Теоретичні відомості

2. Основні команди

3. Приклад роботи

1. Теоретичні відомості

<https://git-scm.com/book/en/v2/Git-Tools-Reset-Demystified>

Git – система контролю версій (VCS, version control system) яка використовується для розробки програмного забезпечення і інших задач.

Git дозволяє постійно слідкувати за змінами в проекті.

Git-проект можна уявляти з трьох частин:

1. Робоча директорія (working directory) – де виконується уся робота: створення, редагування, видалення і організація файлів.

2. Індексування (staging area) – де перераховуються зміни, які зроблені в робочій директорії.

3. Репозиторій (Git repository)– де Git зберігає зміни як різні версії проекту.

Процес роботи з Git можна описати, як редагування файлів в робочій директорії, додавання їх до області індексування і збереження змін у Git-репозиторії.

2. Основні команди

Джерело - <https://services.github.com/kit/downloads/github-git-cheat-sheet.pdf>

**КОНФІГУРАЦІЯ**

Конфігурація інформації про користувача для всіх локальних репозиторіїв

$ git config --global user.name “[name]”

- задає ім’я користувача, яке буде приєднана до усіх транзакцій комітів

$ git config --global user.email "[email address]"

- задає адресу електронної пошти

$ git config --global color.ui auto

- дозволяє корисну кольоризацію виводів команд

**СТВОРЕННЯ РЕПОЗИТОРІЮ**

$ git init [project-name]

- створення нового репозиторію з заданим іменем

$ git clone [url]

- завантажує проект і історію його версій

**ВНЕСЕННЯ ЗМІН**

$ git status

- перераховує нові або зміненні файли для коміту

$ git add [file]

- індексує файли (додає на staging area)

$ git reset [file]

- прибирає файли з індексації, але зберігає їх контент, (скидає індекс і робочий каталог на стан вашої останньої фіксації)

$ git diff

- показує зміни, які ще не проіндексовані

$ git diff --staged

- показує зміни в файлу між індексованою і останньою версією файлу

$ git commit -m "[descriptive message]"

- записує знімок файлу в історію версій

**ГРУПУВАННЯ ЗМІН**

Іменування серії комітів і об’єднання завершених зусиль

$ git branch

- перераховує усі локальні гілки в поточному репозиторії

$ git branch [branch-name]

- створіє нову гілку

$ git checkout [branch-name]

- перемикає до визначеної гілки і оновлює робочу директорію

$ git merge [branch]

- об’єднує визначену історію гілки до поточної гілки

$ git branch -d [branch-name]

- видаляє визначену гілку

**РЕФАКТОРИНГ ІМЕН ФАЙЛІВ**

Релокація і видалення версійованих файлів

$ git rm [file]

видаляє файл з робочої директорії і індексує видалення

$ git rm --cached [file]

Видаляє файл з контролювання версії, але зберігає його локально

$ git mv [file-original] [file-renamed]

- змінює ім’я файлу і готує його до коміту.

Changes the file name and prepares it for commit

ПРИПИНЕННЯ ВІДСТЕЖУВАННЯ

Виключення тимчасових файлів і шляхів

\*.log

build/

temp-\*

- текстовий файл, який називається .gitignore пригнічує випадкового управління версіями файлів і шляхів, відповідних зазначеним шаблони

$ git ls-files --other --ignored –exclude-standard

- перелічує всі ігноровані файли і цьому проекті

ЗБЕРЕЖЕННЯ ФРАГМЕНТІВ

Відкладання і відновлення незавершених змін

$ git stash

- тимчасово зберагіє (припасає) усі змінені відслідковувані файли

$ git stash list

- перераховує усі припасені (stashed) набори змін

$ git stash pop

- відновлює останні припасені файли

$ git stash drop

- відкидає останні припасені набори змін

**ПЕРЕГЛЯД ІСТОРІЇ**

Перевірка і інспекція еволюції файлів проекту

$ git log

- перераховує історію версій для поточної гілки

$ git log --follow [file]

- перераховує історію верій для файлу, включаючи переіменування

$ git diff [first-branch]...[second-branch]

- показує зміни контенту між двома гілками

$ git show [commit]

- виводить зміни метаданих і контенту для визнаеного коміту.

ПЕРЕРОБЛЕНЯ КОМІТІВ

Стирання помилок і виправлення історії змін

$ git reset [commit]

- відміняє усі коміти послі визначеного [commit], зберігаючи зміни локально

$ git reset --hard [commit]

- відкидає назад всю історію і зміни до визначеного коміту.

СИНХРОНІЗАЦІЯ ЗМІН

Реєстрація закладки репозиторію і обмін історією версій

$ git fetch [bookmark]

- завантажує всю історію з репозиторію закладки

$ git merge [bookmark]/[branch]

- зливає гілку закладки в поточну локальну гілку

$ git push [alias] [branch]

- вивантажує усі локальні коміти гілки на GitHub

$ git pull

- завантажує історію закладки і включає зміни