МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Інститут Прикладного системного аналізу

Кафедра Системного проектування

**Лабораторна робота № 1.**

З дисципліни

«Проектування інформаційних систем»

На тему

***«***Системи контролю версій SVN, GIT***»***

Виконала

студентка групи ДА-61

Материнська Софія

КИЇВ

2019

***1. Мета роботи:*** за допомогою системи контролю версій завантажити коди програми у репозіторій. Відтворити типовий цикл розробки програмного забезпечення з використанням системи контролю версій.

***2. Завдання роботи:***

1. Обрати безкоштовну систему репозиторія для системи контролю версіями, наприклад projectlocker, або інш.

2. Встановити клієнтське безкоштовне програмне забезпечення для роботи с системою контролю версій (GIT, SVN clients).

3. Протягом роботи над лабораторними роботами 2-6 використовувати систему контролю версіями.

4. Описати цикл розробки програмного забезпечення з використанням системи контрою версій.

***3. Оформлення результатів роботи:***

*3.1. Опис команд, які використовувалися протягом виконання роботи з системою контролю версіями.*

Робота з репозиторієм зводиться до основного набору команд (push - вливання локальних змін в віддалений репозиторій; pull - вливання змін з віддаленого сховища в локальний):

git fetch origin - скачати всі гілки з сервера, але не мержити їх

git checkout master - переключитися на гілку master

git pull origin master - отримати всі зміни гілки master з віддаленого сервера

git checkout -b feature /( умовна назва гілки ) - створення нової гілки. "Feature /" прийнято позначати гілки, пов'язані з розробкою нового функціоналу

git checkout -b fix /( умовна назва гілки ) - аналогічно попередньому. "Fix /" зазвичай використовується для гілок, що реалізують виправлення за кодом, виконаному раніше

git add # файл (и) # - файл або файли (через пробіл), зміни за якими повинні бути поміщені в дану гілку

git commit -m "Короткий опис змін" - Комміт проведених змін в гілку

git push origin feature /( умовна назва гілки ) - відправити гілку на сервер

git merge feature /( умовна назва гілки ) - смержити гілку "feature /( умовна назва гілки ) " в поточну гілку

Ось, в принципі, і весь основний набір команд для роботи з репозиторієм (без урахування можливості наявності кофлікт, звичайно :)). Далі розглянемо більш докладно повний набір команд, які можуть стати в нагоді при роботі з GIT:

***Базові операції з репозиторієм***

Ініціалізувати репозиторій

**git init**

Додати віддалений репозиторій

**git remote add origin git@bitbucket.org: username / project.git**

Клонувати репозиторій з сервера

**git clone git@bitbucket.org: username / project.git**

***статус***

Для того, щоб побачити статус файлів, існує команда, яка покаже, які файли ще не були додані репозиторій, а які були змінені:

**git status**

Додаємо і виключаємо файли

додаємо file.js

**git add file.js**

Додаємо всі файли

**git add.**

Скасувати зміни файлу file.js

**git reset file.js**

Ресет для всіх файлів:

**git reset.**

***зміни***

дивимося зміни

**git diff**

Дивимося зміни в file.js

**git diff file.js**

***видалення файлів***

Видалити з віддаленого сховища непотрібні файли

**git ls-files --deleted -z | xargs -o git rm**

**git commit -m "Deleted ..."**

**git push origin branchname**

Видалення файлів з кеша:

**git rm -r --cached / dir**

***робимо комміти***

робимо комміт

**git commit -m 'commit message'**

Робимо Комміт, відразу додавши всі файли

**git commit -a -m 'commit message'**

Перезаписуємо останній Комміт

**git commit --amend -m 'new commit message'**

Ресет останнього комміту

**git reset --soft HEAD ^**

Відкотитися до конкретного комміту (хеш дивимося в «git log»):

**git reset --hard d8578edf8458ce06fbc5bb76a58c5ca4a58c5ca4**

***управляємо гілками***

Дивимося локальні гілки

**git branch**

Дивимося локальні і віддалені гілки

**git branch -a**

Перемикаємося на майстер

**git checkout master**

Створюємо нову гілку і перемикається на неї

**git checkout -b new\_branch**

Створюємо локальну гілку як копію з віддаленого сервера

**git checkout -b new\_branch origin / new\_branch**

Видаляємо локальну гілку (порожню або влитися в майстер)

**git branch -d new\_branch**

Просто видаляємо локальну гілку

**git branch -D new branch**

Видаляємо гілку з сервера

**git push origin: new\_branch**

Підтягуємо зміни гілки майстер

**git pull --rebase origin master**

***Підтягуємо зміни з сервера***

Підтягуємо всі зміни з сервера, що не вливаючи їх

**git fetch**

Пушим гілку

**git push origin new\_branch**

Пушим гілку з перезаписом історії коммітов (тільки коли добре розумієте, що робите)

**git push -f origin new\_branch**

***злиття гілок***

Процес злиття гілок при командній роботі - досить трудомісткий процес - потрібно правильно враховувати зміни, зроблені усіма учасниками команди. Процес об'єднання гілок може бути в форматі merge і rebase. Даний процес добре описаний в статті.

Просто мерджім гілку в поточну

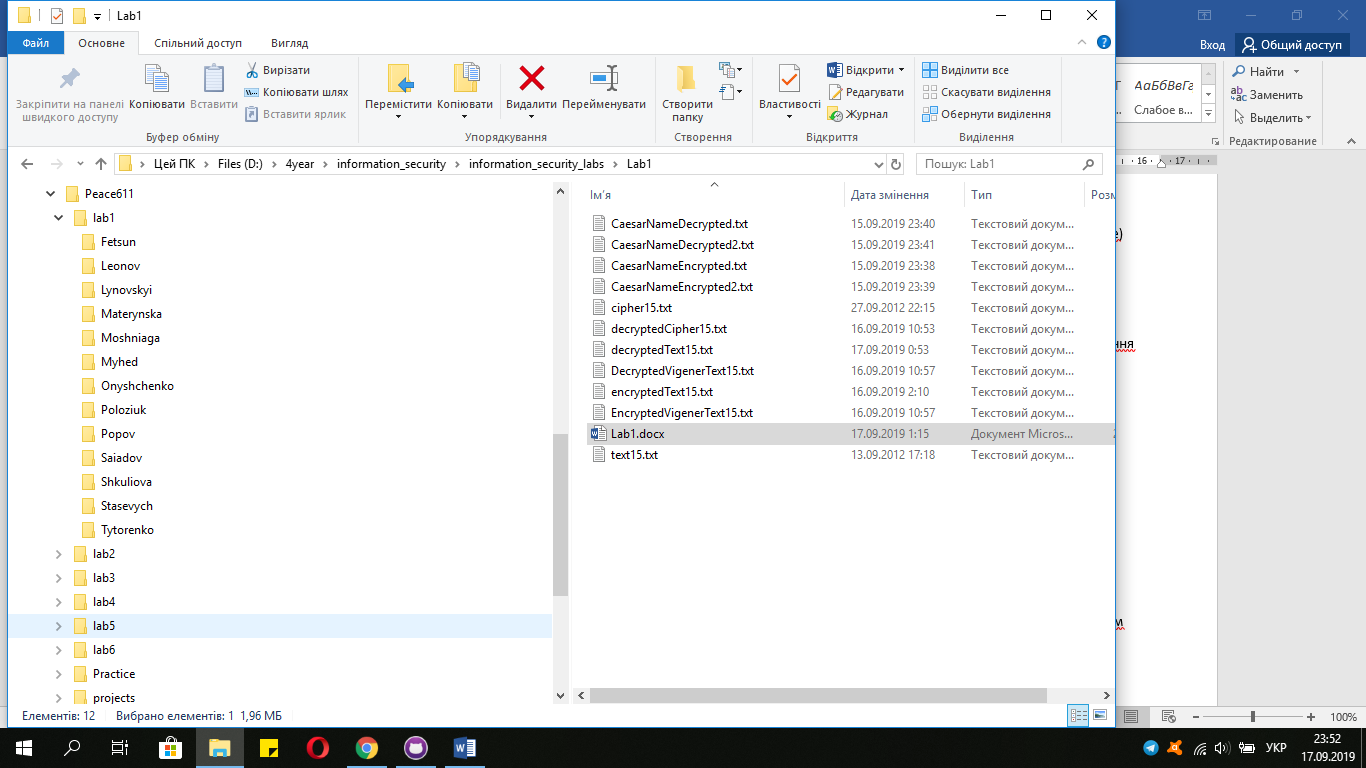
**git merge new\_branch**

Мерджим гілку зі створенням мердж-коммітів

**git merge --no-ff new\_branch**

Ребейзимося від майстра

**git rebase master**

3.2. Лістинг каталогів у репозиторію для програмних кодів для л.р. 2-6.

***4. Висновки:*** за допомогою системи контролю версій було завантажено коди програми у репозіторій. Відтворено типовий цикл розробки програмного забезпечення з використанням системи контролю версій.