

Міністерство освіти і науки, молоді та спорту України

Національний Технічний Університет України

«Київський Політехнічний Інститут»

«Інститут прикладного системного аналізу»

Кафедра системного проектування

Лабораторна робота № 1

з дисципліни «Проектування інформаційних систем»

«Системи контролю версій»

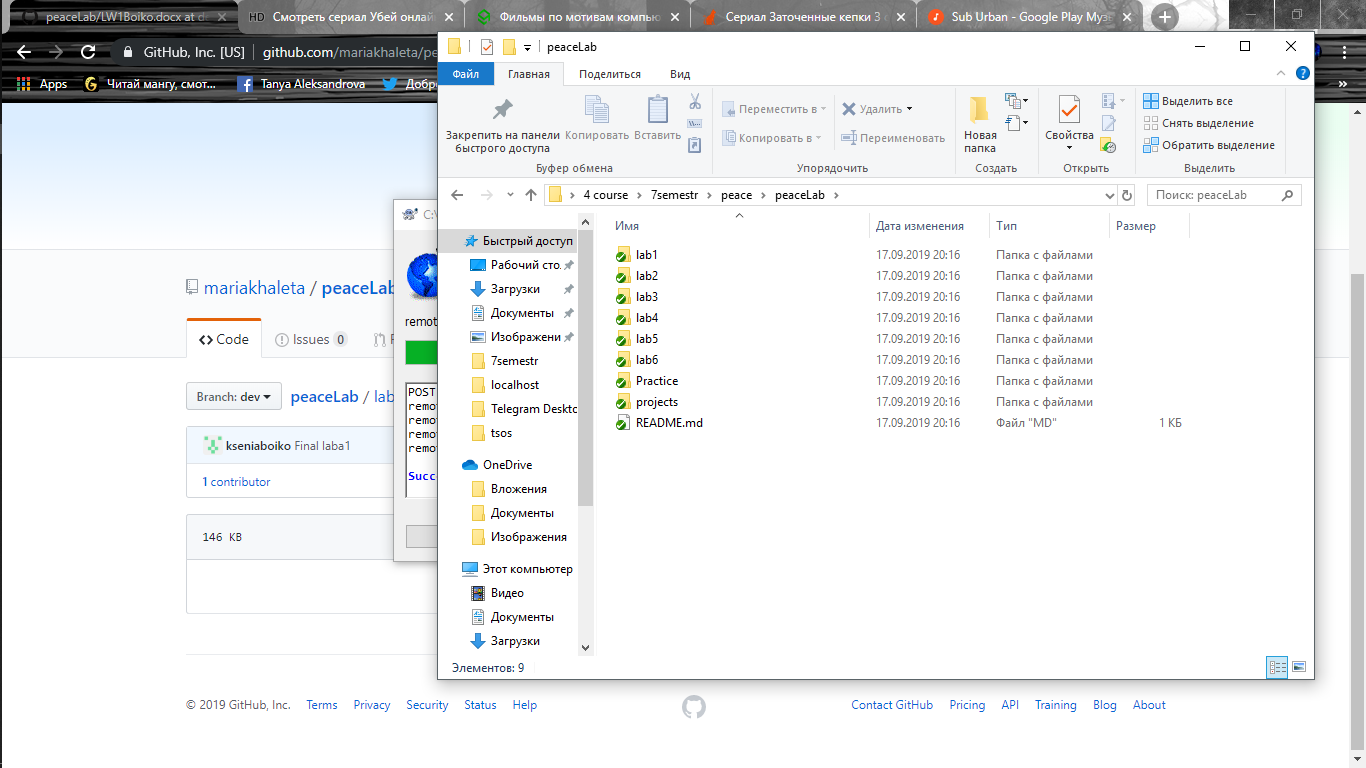
Виконала

студентка ДА-61

Александрова Тетяна

Київ-2019

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



* git clone

Git отримує копію практично всіх даних, що є на сервері.

* git checkout –b

Створюємо та переходимо в вітку.

* git add

Додаємо файли версіонний контроль.

* git commit –m “”

Фіксуємо зміни

* git push

Пушим вітку

* git merge dev

Злиття приймає вміст гілки джерела і об'єднує їх з цільової гілкою. У цьому процесі змінюється тільки цільова гілка. Історія вихідних гілок залишається незмінною.

**Різниця між GIT та SVN**

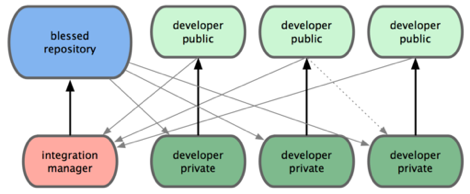
Git це розподілена система керування версіями. Але що насправді означає цей термін? А то, що замість виконання svn checkout (url), щоб отримати останню версію вашого сховища, в Git потрібно виконати git clone (url), що дасть вам повну копію історії проекту. Тому, відразу ж після клонування, ви не маєте історією проекту на сервері. Цікаво, Subversion є настільки неефективною в цьому, що в цілому виконати svn checkout сховища в Subversion майже так само швидко, як клонувати його весь в Git.

Отже, це дає вам дві переваги. По-перше, практично кожна операція виконується з даними на локальному диску, а значить, що це неймовірно швидко і може бути виконано в оффлайн режимі. Ви можете виконувати команди commit, diff, log, branch, merge, створювати анотації до файлів і багато іншого, повністю в оффлайн режимі, моментально. Більшість команд в Git довше набираєш на клавіатурі, ніж вони насправді виконуються. Тепер зупиніться на хвилину і спробуйте згадати, скільки разів ви ходили випити чашечку кави, поки Subversion виконувала будь-яку команду. Або швидко накидайте список випадків, коли ви хотіли виконати Комміт, але не мали підключення до Інтернету або не могли підключитися до корпоративної VPN.

Іншим неявним перевагою такої моделі є те, що робочий процес не має єдиної точки відмови. Оскільки кожна людина, що працює над проектом, має по суті його резервну копію, і втрата зв'язку з серверами загрожує йому, в гіршому випадку, незначними незручностями. Уявіть собі на хвилину, що на вашому Subversion сервері вийшов з ладу жорсткий диск, де зберігалася остання резервна копія, і скільки годин пройде до того моменту, коли можна буде продовжити працювати? У Git, будь-який член команди може скинути історію проекту на будь-який сервер, де кожному дозволено доступ по ssh, і вся команда може просто почати працювати з проектом за лічені хвилини.

Остання перевага розподілених систем, яке я розгляну, полягає в неймовірних схемах робочих процесів, які вони вам надають. Git не залежить від центрального сервера, але має можливість синхронізуватися з іншими репозиторіями Git - виконуючи push і pull змін між ними. Отже, ви можете додати кілька віддалених сховищ для вашого проекту, якісь тільки для читання і якісь з правами на запис, а значить, ви можете мати практично будь-який тип робочого процесу, який тільки можна уявити.

Ви можете продовжувати користуватися централізованою схемою процесу, з одним центральним сервером, куди все можуть викладати зміни і отримувати їх звідти. Однак, ви в змозі робити і цікавіші речі. Наприклад, надати віддалений репозиторій для кожного користувача або підгрупи, де надані права на запис, а потім відповідальний користувач (QA, наприклад) може об'єднати воєдино роботу кожного члена і виконати push в "бойовій" репозиторій.



Також з Git можливо створити будь-яку ієрархічну або "peer-based" модель робочого процесу, про яку можете тільки подумати, на додаток до можливості використовувати її в якості централізованого сервера, як Subversion. Робочий процес може розвиватися і адаптуватися разом з вашої бізнес-моделлю.

Ще один цікавий приклад, це Ruby-хостинг компанії Heroku. Для розгортання своїх систем, просто виконайте push в віддалений репозиторій "heroku". Ви можете працювати в інших віддалених репозиторіях, а потім, коли буде необхідно розмістити свій код на робочому сервері, то просто виконуєте push в репозиторій Heroku. Уявіть собі спробу зробити таке з Subversion.

*Тема проекту*: календарний розподіл нотаток та справ.

**Висновок**

У даній роботі ми познайомились с git, створили власну гілку та помістили свою роботу в папку dev.