Створення проекту та перші коміти

Git - це розподілена система контролю версій. І на машині розробника завжди знаходиться повноцінний репозиторій, у якому можливо створювати гілки, тримати історію та який синхронізується з іншими репозиторіями членів команди.

Це означає, що члени команди можуть взаємодіяти навіть без окремо виділеного сервера, що підтримує основний репозиторій. Однак, у такому випадку, вам необхідно або надати доступ до вашої машини ззовні (наприклад, надати http-доступ до репозиторія або через ssh), або ж обмінюватися комітами через email. Це все не завжди зручно та просто.

Тому, як правило, команди мають виділений сервер, який використовується для підтримки основного репозиторія та синхронізуються через нього. Ми будемо використовувати для цього GitHub.

Для початку роботи, спочатку створимо репозиторій на своїй локальній машині.

```
mkdir my-project
cd my-project
git init
```

Після виконання останньої команди, у вашій директорії створиться папка .git, де міститься сам репозиторій - база даних його об'єктів (blob, дерева, коміти тощо).

Сама ваша директория (my-project у цьому випадку) є робочою копією. Для того, щоб почати відслідковувати зміни у своєму проекті, потрібно додати потрібні файли до індексу (індекс тримає зміни, які в подальшому зафіксуються у вигляді коміту):

```
echo “My Project” > README.md
git add README.md
```

Тепер можна додати зміни до історії за допомогою команди commit:

```
git commit
```

Піля того, як ви введете повідомлення, буде створено коміт. Коміт-об'єкт створюється з даних, які занесено до вашого індексу.
Тепер потрібно відправити зміни до репозиторія на Github. Для цього потрібно створити його на сервері.

https://github.com/new

За допомогою даної сторінки створіть новий публічний репозиторій. Це, фактично, виконає git init на сервері. Коли репозиторій створено, ви побачите посилання, яке потрібно використати для того, щоб сконфігурувати зовнішній репозиторій, з яким буде проходити синхронізація на вашій машині. Наприклад,

```
git remote add origin git@github.com:roman-mazur/test-repo.git
```

Остання команда додасть зовнішній репозиторій з коротким іменем “origin”. Коротке ім'я може надалі використовуватися у командах push, fetch та pull. Команда

git push -u origin master

відправить коміти гілки master до репозиторія origin.

Однак, перед тим, як виконати дану команду, необхідно нашалашутвати ssh-ключі, щоб сервер міг авторизувати вас та прийняти зміни. Ключі можуть бути налаштовані на сторінці

https://github.com/settings/ssh

Більше інформації про налаштування ключів можна знайти за наступним посиланням:
https://help.github.com/articles/generating-ssh-keys/

Інші члени команди можуть клонувати ваш репозиторій:
`git clone <repo_url>`

Для того, щоб інші члени вашої команди могли виконувати push у ваш репозиторій, у нашатуваннях проекту, їм потрібно дати на це права.
https://github.com/`<github_name>`/`<repo_name>`/settings/collaboration

Але, оскільки git - це розподілена система контролю версій, останній пункт не обов'язковий: кожен з членів команди може мати свій репозиторій на github, і кожен буде забирати зміни, наявні в репозиторіях колег. Наприклад,

```
git remote add friend1 <friend's repo url>
git pull friend1 master
```

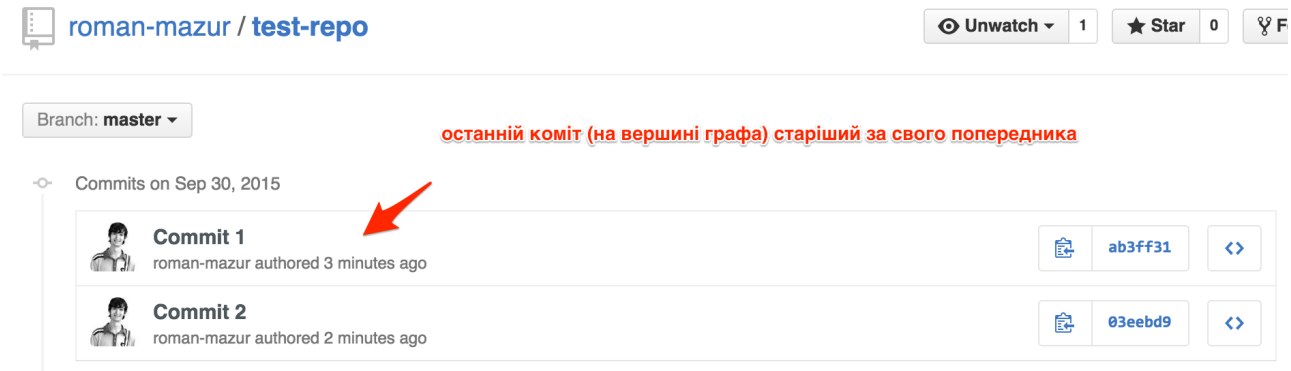
Обов'язково познайомтеся з главами 2 та 3 книги Pro Git для наступних завдань
https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository
https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

Вам необхідні будуть знання про створення гілок (git branch, git checkout -b), злиття гілок (git merge), а також перебазування комітів (git rebase).

Для зарахування практичної роботи, вам необхідно продемонструвати github-репозиторій, який задовольнятиме наступні вимоги.

1. Гілка master вашого репозиторію містить коміти усіх членів команди (стежте за тим, щоб email, який git записує, коли зберігає відомості про автора, співпадав з email'ом, який ви використали для реєстрації на GitHub).
2. Гілка master містить коміти від усіх членів команди підряд (тобто між рядом комітів, створених різними членами команди, не повинно бути merge-комітів). Це не обов'язково для абсолютно всієї історії гілки. Але має бути принаймні один такий фрагмент. Для цього вам потрібно познайомитися з тим, що таке fast-forward у git та перебазування комітів (git rebase). Про використання git rebase можете додатково подивитися у даній статті: https://www.atlassian.com/git/tutorials/merging-vs-rebasing/

   Увага, доповнення! Коміти від різних членів команди також повинні опинитися в графі не в хронологічному порядку (одного прикладу достатньо).

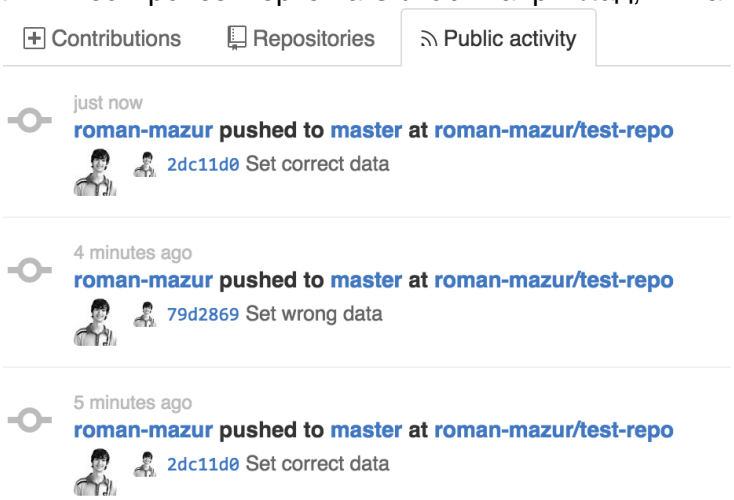


Ви можете цього досягнути, якщо ви зробите коміт, пізніше коміт зробить ваш колега, і ви виконаєте rebase на коміт колеги.

3. У репозиторії повинні бути присутні merge-коміти, виконані кожним з членів команди (вам знадобиться git merge).
4. Гілка master повинна мати принаймні один revert-коміт (який застосовує зворотні зміни до певного коміта). Як не дивно, вам потрібно буде скористатися командою git revert. Також слідкуйте за тим, щоб commit-повідомлення містило посилання на той коміт, зміни якого скасовуються.

Додаткові завдання (не обов'язкові):

5. Одна з гілок проекту має бути "перезатерта" за домогою команди git push -f, а потім відновлена (тою ж таки командою). Це можна продемонстувати за допомогою акитвності репозиторію на GitHub. Наприклад, як на наступному знімку:



6. Продемонструйте revert merge-коміта, яким скасуєте злиття якоїсь гілки. А потім злийте цю ж гілку знову. Вам допоможе ця стаття: https://git-scm.com/blog/2010/03/02/undoing-merges.html

Оцінювання

За кожу практичну роботу (всього їх 7) ви можете отримати по 8 балів. Ще 44 бали ви можете отримати за розрахунково-графічну роботу: демонстрація робочого проекту (мінімальної версії), на якому буде висвітлено всі аспекти нашого курсу.

У кожній з практичних робіт є ряд завдань, бали між якими розподіляються рівномірно. Так, за кожне з обов'язкових завдань у цій роботі ви можете отримати 2 бали. Також у роботах можуть бути присутні додаткові завдання. Вони приносять по 1-ому балу, але і без них можливо отримати максимум. Також можуть бути додаткові завдання в аудиторії, про які ви дізнаєтеся, прийшовши на заняття.

У кожної роботи є 2 крайні терміни. Якщо робота здана після першого терміну, ви отримаєте 80% від зароблених балів. Якщо після другого - не отримуєте балів узагалі. Крайні терміни для робіт ви побачите на kpi-integration.appspot.com

Усі члени команди отримують однакову кількість балів. Для того, щоб робота була оцінена, необхідно відправити посилання на github-репозиторій викладачеві. При оцінці будуть враховуватися лише ті коміти репозиторія, які були зроблені до часу відправлення листа.

Для допуску до заліку, необхідно здати всі роботи (навіть без балів).
Для отримання заліку автоматом, необхідно набрати 65 балів. Інакше доведеться пройти іспит та відповісти на ряд питань, які розглядалися на лекції.

Якщо виникають питання, або ви знайшли невідповідності в завданнях, пишіть у групу обговорення:
https://groups.google.com/forum/#!forum/kpi-integration-2015

Успіхів!