Архітектура програмного забезпечення

Лабораторна робота №1

Тема: Git та HTTP сервер
Мета: Отримання навичок роботи з системою контролю версій та HTTP протоколом

Під час виконання даної лабораторної роботи вам необхідно буде продемонструвати навички роботи з git та базові знання мови програмування Go. Наступний розділ надає приклади того, як працювати з Git за допомогою командного рядка. Однак ви також можете виконати завдання користуючись UI-інструментами, такими як Sourcetree або GitHub Desktop.

Завдання виконується в команді. Інструкції для реєстрації команди в системі курсу наведено в кінці документа.

Створення репозиторія та перші коміти

Git - це розподілена система контролю версій. На машині розробника завжди знаходиться повноцінний репозиторій з історією змін, у якому можливо створювати гілки та який синхронізується з репозиторіями інших 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, 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.

Після виконання останньої команди, у вашій директорії створиться папка .git, де міститься сам репозиторій - база даних його об’єктів.

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):

Тепер ми можемо додати зміни до історії за допомогою команди *commit*:

```
git commit
```

Піля того, як ви введете повідомлення-опис ваших змін, буде створено коміт. Коміт-об’єкт створюється з даних, які занесено до вашого індексу.

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

echo “My Project” > README.md # Створюємо новий файл
git add README.md # Додаємо його до індекса
```

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
```

Матеріали, необхідні для виконання завдання

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*).

Хорошим ресурсом для знайомства з мовою програмування Go є "Тур з Go". Окрім базових знань мови вам також знадобиться знання пакету net/http.

Завдання

Для зарахування лабораторної 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. У корені репозиторія можливо запустити команду

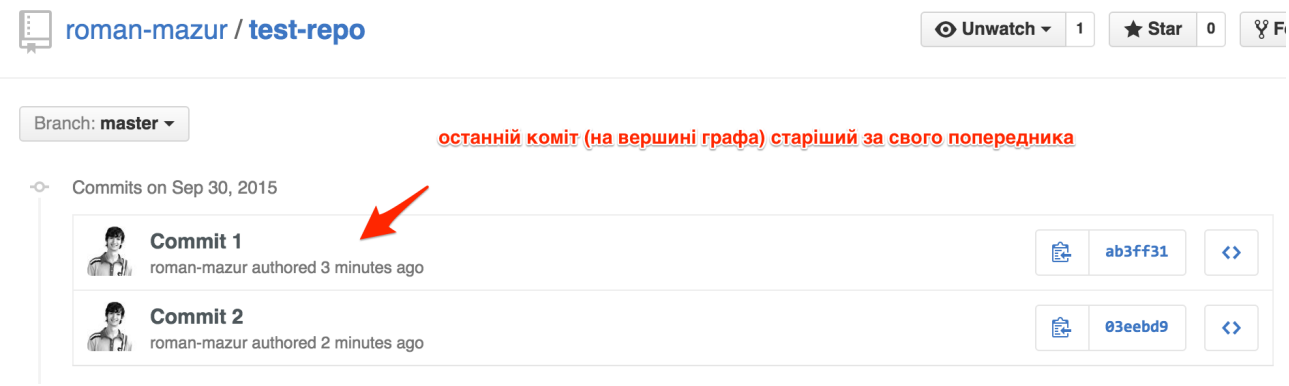
   go run server.go

   після чого за посиланням *http://localhost:8795/time* можливо зробити HTTP GET запит та отримати відповідь у форматі JSON, яка містить інформацію про поточний час сервера у форматі RFC3339 в полі *time*. Наприклад,

   ```
   {
     "time": "<поточний час>"
   }
   ```

3. 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 поняттям 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/

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



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

4. У репозиторії повинні бути присутні merge-коміти, виконані кожним з членів команди (вам знадобиться *git merge*).

5. 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.

Оцінювання

За виконання усіх критеріїв завдання ви можете отримати 5 балів (по 1 за кожне). Ще один бал ви отримаєте, якщо відформатували правильно ваш код (за допомогою go fmt). Всього 6 балів за вашу першу роботу.

У кожної практичної роботи є крайній термін. Якщо робота виконана пізніше за вказаний термін, за кожен тиждень прострочки нараховується штрафний -1 бал.

Крайні терміни для робіт ви побачите на

https://kpi-architecture-course.appspot.com/

Усі члени команди отримують однакову кількість балів. Для того, щоб робота була оцінена, необхідно відправити посилання на github-репозиторій викладачеві. Зробити це можна за допомогою команд у чаті. Вам необхідно буде створити команду, визначивши, з ким ви працюєте над проектом курсу. Усі члени команди мають бути зареєстровані в системі курсу.

Створення команди та відправка завдання

Для створення команди зі студнетами, що мають нікнейми @student1 та @student2, необхідно ввести наступну інстуркцію:

```
/team team-name @student1 @student2
```

Команда створюється **один раз одним із учасників**. Власний нікнейм не потрібно вводити (інструкція вище створює команду з трьох осіб).
Назва команди не може містити пропусків (використовуйте знаки підкреслення, дефіси або CamelCase нотацію для відокремлення слів у назві).
Зміна приналежності студента до певної команди вважається експоненціально складною задачею, а тому не підтуримується через команди в чаті ;)
При великій потребі, говоріть з викладацьким складом - вони допоможуть. Однак такі зміни не бажані.

Після створення команди ви можете віддати свій проект на github на перевірку за допомогою

```
/submit 1 https://github.com/your_name/your_repo
```

Зверніть увагу, що цю команду має виконати власник репозиторія на GitHub (ви можете відправляти на перевірку лише власні репозиторії).
Викладацький склад буде оповіщено, результати оцінювання, ви зможете переглянути на сайті курсу.

Якщо виникають питання, або ви знайшли невідповідності в завданнях, пишіть у Slack (канал #practice).

Успіхів!