Лабораторна робота №5

Управління командною роботою. Робота з Git, Github

Основні причини, через які виникла потреба в системах контролю версій (version control systems):

1). досить часто над великими проектами працює багато людей, і виникає потреба кооперуватися і працювати над однією і тією ж кодовою базою разом;

2). корисно мати можливість зберігати всі попередні версії коду, так можна до них відкочуватися (скасовувати зміни), бачити, коли і які зміни були внесені, хто їх зробив, де саме вони були зроблені і т. д.

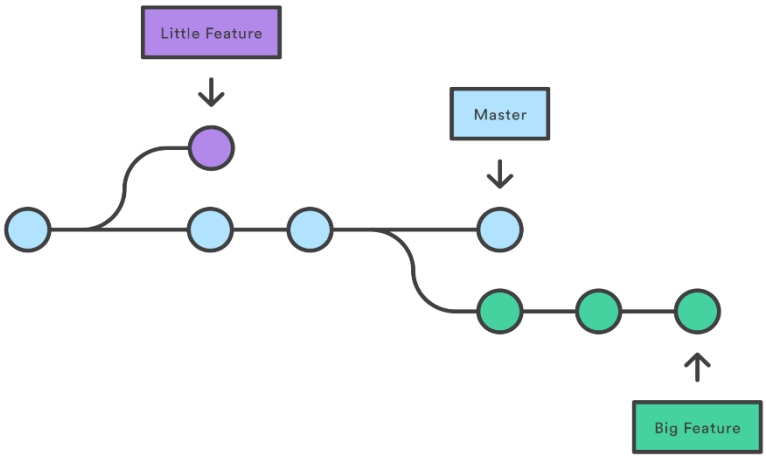
Одна з таких систем - Git. Він не є єдиним у своєму роді (є, наприклад, SVN, Mercurial, CVS), але останнім часом став популярний саме Git, оскільки багато розробників вважають найбільш зручним саме його.

Одне з ключових понять при роботі з системами контролю версій - це репозиторій.

Репозиторій - це віртуальне сховище версій проекту. Фізично в репозиторії "зберігаються" файли. Логічно є ще різні речі, прив'язані вже до Git`у, такі як гілки, теги, версії, коміти.

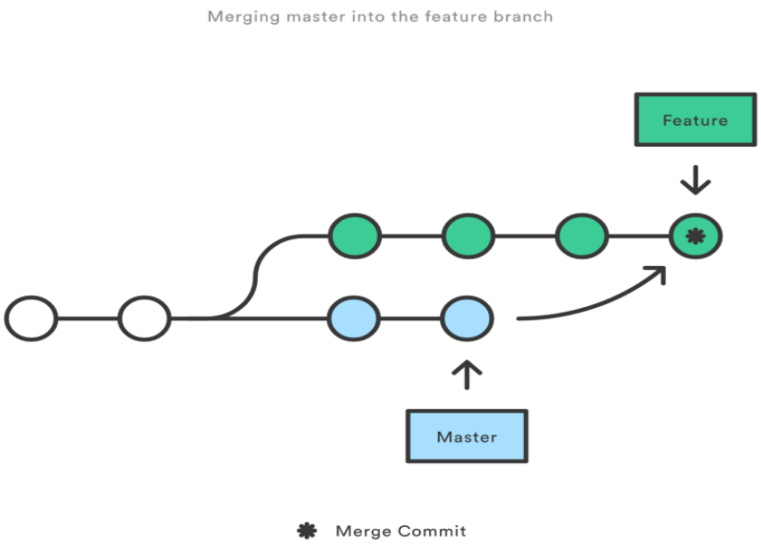
• Коміт - черговий зафіксований стан проекту, "одиниця" версії.

Всі коміти (крім першого) повинні успадковуватися від попереднього коміту, утворюючи таким чином деревоподібну структуру.



Проект може розгалужуватися, наприклад якщо два розробника паралельно працюють і кожен створює свої коміти. Щоб не виникало повного хаосу і можна було орієнтуватися в тому, що відбувається вводять поняття гілки. На зображенні вище крім master є гілки Little Feature і Big Feature.

Навіщо це потрібно? Зазвичай, коли розробку веде кілька людей, кожен вносить коміти в окрему гілку, а потім зливає її з головною гілкою, як показано нижче.



Між гілками можна вільно перемикатися.

Гілка - за великим рахунком просто іменоване посилання на якийсь коміт, яка вказує на кінець чергового відгалуження від головної центральної гілки (по дефолту вона називається master).

Як правило, репозиторій працює в прив'язці до якоїсь гілки і при створенні нового коміту, він приєднується до того, на який посилається гілка, і гілка починає вказувати на новий коміт.

Переваги Git

• Безкоштовний, з відкритим вихідним кодом. Це означає, що його можна безкоштовно завантажити і вносити будь-які зміни у вихідний код;

• Невеликий і швидкий. Він виконує всі операції локально, що збільшує його швидкість. Крім того, Git локально зберігає весь репозиторій в невеликий файл без втрати якості даних.

Резервне копіювання.

Git ефективний в зберіганні бекапів, тому відомо мало випадків, коли хтось втрачав дані при використанні Git;

Просте розгалуження. У Git управління гілками реалізовано набагато простіше і ефективніше.

Git - це інструмент, що дозволяє реалізувати розподілену систему контролю версій, а GitHub - це сервіс для проектів, що використовують Git.

GitHub - сервіс онлайн-хостингу репозиторіїв, що володіє всіма функціями розподіленого контролю версій і функціональністю управління вихідним кодом - все, що підтримує Git і навіть більше.

Зазвичай він використовується разом з Git і дає розробникам можливість зберігати їх код онлайн, а потім взаємодіяти з іншими розробниками в різних проектах.

Також у GitHub є контроль доступу, багтрекінг, управління завданнями і гілки для кожного проекту.

Мета GitHub - сприяти взаємодії розробників.

До проекту, завантаженому на GitHub, можна отримати доступ за допомогою інтерфейсу командного рядка Git і Git-команд.

Також є й інші функції, такі як документація, запити на прийняття змін (pull requests), історія комітів, інтеграція з множиною популярних сервісів, email-повідомлення, графіки, вкладені списки завдань і т.д.

Робота з гілками. Додавання гілок.

Гілки Github дозволяють працювати з декількома версіями проекту одночасно. За замовчуванням при створенні сховища створюється гілка master, це основна робоча гілка. Можна створити додаткові гілки, наприклад, для того, щоб тестувати програмне забезпечення перед тим, як воно буде опубліковано в гілці master.

Таким чином, можна одночасно розробляти продукт і надавати користувачам стабільну версію. Також можна створювати окремі гілки для версії програми для різних систем.

Робота з гілками. Додавання гілок.

Гілки Github дозволяють працювати з декількома версіями проекту одночасно. За замовчуванням при створенні сховища створюється гілка master, це основна робоча гілка. Можна створити додаткові гілки, наприклад, для того, щоб тестувати програмне забезпечення перед тим, як воно буде опубліковано в гілці master.

Таким чином, можна одночасно розробляти продукт і надавати користувачам стабільну версію. Також можна створювати окремі гілки для версії програми для різних систем.

Створення запитів злиття (pull request)

Запит злиття або Pull Request - це можливість, завдяки якій будь-який розробник може попросити іншого, наприклад, автора сховища переглянути його код і додати його в основний проект або гілку.

Інструмент роботи з запитами злиття використовує інструмент порівняння diff, тому можливо побачити всі зміни, вони будуть підкреслені іншим кольором. Pull Request можна створити відразу ж після створення комітів.

Звіти про помилки

Зручно використовувати GitHub не тільки для розробки і управління кодом, але і для зворотного зв'язку з користувачами.

На вкладці "Issue" користувачі можуть залишати повідомлення про проблеми, з якими вони зіткнулися при використанні продукту.

Релізи

Коли продукт досяг певної стадії можна випустити реліз.

Завдання:

1. Створити аккаунт GitHub.
2. Створити сховище.
3. Додати гілки.
4. Змінити файл і коміти.
5. Створити запити злиття (Pull Request).
6. Переглянути і схвалити запити на злиття.
7. Створити звіт про помилки.
8. Створити реліз.