Основи Git 2

№ уроку: 33 **Курс:** Manual QA

Засоби навчання: Браузер, Microsoft Office, Git

Огляд, мета та призначення уроку

Метою даного уроку є дізнатися, які стани може мати файл у репозиторії Git, навчитися додавати, змінювати, видаляти, переміщувати та перейменовувати файли, вимикати відстеження для файлів та директорій, створювати проекти на GitHub, клонувати проекти, надсилати свої зміни на віддалений сервер та завантажувати нові зміни з нього, працювати з гілками, створювати, видаляти та перемикатися між ними, а також виконувати злиття гілок.

Вивчивши матеріал даного заняття, учень зможе:

- Здійснювати будь-які маніпуляції з файлами в репозиторії.
- Переглянути зміни файлів в індексі.
- Додавати файли до .gitignore.
- Самостійно створити проект на GitHub.
- Завантажити свій локальний проект на віддалений сервер.
- Завантажити та надіслати зміни.
- Клонувати існуючий проект.
- Створювати та видаляти гілки.
- Перемикати між гілками.
- Виконувати злиття гілок.

Зміст уроку

- Робота із файлами. Стан файлів
- Віддалений репозиторій
- Розгалуження та злиття
- GitHub Flow
- Практика

Резюме

- **.gitignore** – файл, що дозволяє вказувати що у репозиторії слід відстежувати. Працює з файлами, які знаходяться в стані невідстежуваних спочатку. Але якщо потрібно відключити контроль для файлу, який знаходився під контролем git, то зробити це трохи складніше, потрібно використовувати команду git rm --cached <file>. Або ж можна вручну перемістити файл, видалити його в git, зробивши комміт, додати в .gitignore і повернути на місце.
- Синтаксис .gitignore файлу: кожен рядок – окремий шаблон, коментарі повинні починатися із символу "#". Символ "/" на початку рядка вказує на поточну папку. Символ "*" замінює будь-яку кількість символів. Символ "!" на початку рядка інвертує шаблон. приклад:

  ```
  # Ігнорувати файл foo.txt.
  foo.txt

  # Ігнорувати html-файли
  ```


CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

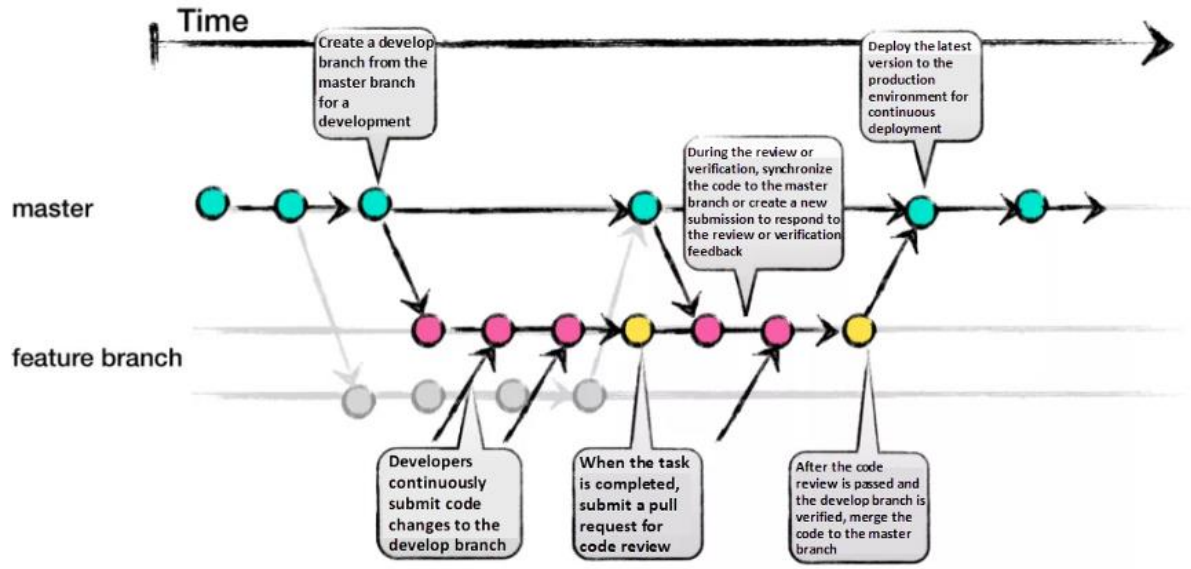
```
*.html

# Але конкретно foo.html не ігнорувати
!foo.html

# Ігнорувати rar-файли у корені проекту
# Припустимо файл /temp/main.rar не буде проігнорований т.як він не у корені
/*.rar

# Ігнорувати css-файли з папки bar не включаючи підпапки

# Припустимо файл /bar/temp/main.css не буде проігнорований т.як він у підпапці temp
/bar/*.css

# Ігнорувати js-файли з папки bar та підпапок, якщо такі будуть
/bar/**.*.js
```

- **GitHub** – найбільший веб-сервіс для хостингу IT-проектів та їхньої спільної розробки. Веб-сервіс заснований на системі контролю версій Git та розроблений на Ruby on Rails, та Erlang компанією GitHub, Inc.
- Для створення нового репозиторію є два способи:
  - Перший – ініціалізувати репозиторій локально та відправити його на GitHub.
  - Другий – створити та ініціалізувати репозиторій на GitHub і потім клонувати, для отримання локальної копії.
- Альтернативами GitHub є сервіси хостингу проектів як Bitbucket, GitLab. У тому числі й Azure. Ми розглянемо всі операції саме на прикладі GitHub. Розібравшись з ним - переключитися на якийсь інший не складе особливих труднощів.
- **git remote add <name> <url>** – підключає віддалений репозиторій з ім'ям <name>, доступний за посиланням <url>.
- **git push <remote> <local_brunch>** – відправка всіх змін до <local_brunch> на віддалений репозиторій.
- **git fetch <remote>** – скачування об'єктів та посилань з віддаленого репозиторію.
- **git pull <remote>** – включає зміни з віддаленого репозиторію в поточну гілку.
- Обидві команди git pull і git fetch завантажують вміст віддаленого репозиторію. Команда git fetch є «безпечним» варіантом із цих двох команд. Вона завантажує вміст, але не оновлює робочий стан локального репозиторію, залишаючи вашу поточну роботу незайманою. Команда git pull завантажує вміст для поточної локальної гілки (командою git fetch) і виконує команду git merge, створюючи комміт злиття для нового вмісту. Є ймовірність, як і за звичайного злиття, що з'являться конфлікти, які треба буде вирішувати вручну.
- **git clone <url>** – клонування віддаленого репозиторію.
- **Гілка** — це покажчик на коміт. За замовчуванням, ім'я основної гілки Git - це master. Як тільки ви почнете створювати комміти, гілка master завжди вказуватиме на останній комміт. Щоразу, при створенні комміту, покажчик гілки master пересуватиметься на наступний коміт автоматично.
- Команда **git branch <name>** – створює нову гілку, тобто - новий покажчик на поточний коміт.
- Покажчик HEAD майже завжди вказує на одну з гілок, при виконанні комміту – він буде доданий до тієї гілки, де зараз знаходиться покажчик HEAD.
- • Для перемикання вказівника HEAD на іншу гілку використовується команда **git checkout**
  <branch_name>. Git дає можливість створити гілку та переключитися на неї однією командою - **git checkout -b <branch_name>**.

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Переглянути список гілок, які існують у репозиторії, можна також командою: **git branch**.
- Для злиття використовується команда **git merge <branch_name>** де <branch_name> – ім'я гілки зміни, яку ви хочете злити в поточну (на яку вказує HEAD під час виконання команди).
- Якщо коміт зливається з тим, який буде доступний рухаючись по історії прямо, то використовується спрощена процедура - просто переносячи покажчик гілки вперед, оскільки немає розбіжностей у змінах. Така операція називається **"fast-forward".**
- Альтернативою операції merge є операція **rebase** (Перебазування). Вона бере зміни поточної гілки та застосовує їх поверх всього, що є у зазначеній.
- **Конфлікт злиття** відбувається, коли ту саму частину коду змінювали в обох гілках, які зливаються в одну. Якщо git самостійно не може вирішити, які зміни будуть результатом злиття – необхідно робити це вручну.
- Стратегія розгалуження GitHub. Набір правил, яких слід дотримуватися:
    - Код у master гілці повинен бути не поламаним і готовим до розгортання у будь-який час (тобто не можна туди покласти код, який завадить зібрати проект і розгорнути його на сервері).
    - Коли планується робота над новою функціональністю, необхідно створити нову гілку (feature гілку) на основі master гілки і дати їй зрозуміле ім'я. Комітити свій код локально та регулярно пушити свої зміни на цю ж гілку у віддалений репозиторій.
    - Відкрити Pull-Request, коли є чітке відчуття, що робота готова і може бути смерджена в master гілку.
    - Після того, як нову фічу в пул-реквесті запрувили, її можна смердити в master гілку.
    - Коли зміни смерджені в master гілку, їх потрібно розгорнути на сервері негайно.



Закріплення матеріалу

- У який стан перейде файл після того, як було виконано його коміт?
- Якою командою можна завантажити собі віддалений репозиторій, маючи лише посилання на нього?
- Якою командою можна підключити дистанційний репозиторій до локального?
- Яку команду потрібно використовувати для надсилання змін на віддалений сервер?
- Яку команду потрібно використовувати, щоб завантажити зміни з віддаленого сервера?

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Яка команда дозволяє створити гілку і відразу на неї перейти?
- Яка операція є альтернативою команди злиття?

Самостійна діяльність учня

Завдання 1
Створіть репозиторій на GitHub. Склонуйте репозиторій. Поміняйте файл через GitHub портал і виконайте там коміт. Потім, використовуючи команду git pull, скачайте собі цей коміт. Перевірте, чи зміни застосовано.

Завдання 2
Пройдіть другу, третю та четверту вправи на тренажері Learn Git Branching.
Пройдіть вправи 1-8 блоку «Push & Pull -- віддалені репозиторії в Git!» у вкладці «Віддалені репозиторії»

Завдання 3
Створіть нову гілку (feature) і зробіть кілька коммітів у ній, потім перейдіть на гілку master. Виконайте кілька коммітів у гілці master. Виконайте злиття гілки feature у гілку master. Видаліть гілку feature.

Рекомендовані ресурси

Офіційна документація Git
https://git-scm.com/docs

Книга Pro Git, Scott Chacon, Ben Straub
https://git-scm.com/book/uk/v2

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com