Основи Git

№ уроку: 32 **Курс:** Manual QA

Засоби навчання: Браузер, Microsoft Office, Git

Огляд, мета та призначення уроку

Метою даного уроку є здобуття загального розуміння призначення систем контролю версій, навчитися встановлювати git, ініціалізувати репозиторії, створювати комміти, переглядати історію коммітів, дізнатися, які стани може мати файл у репозиторії Git, а також навчитися додавати, змінювати, видаляти, переміщати та перейменовувати файли.

Вивчивши матеріал даного заняття, учень зможе:

- Засвоїти основні засади роботи систем контролю версій.
- Встановити Git.
- Працювати з консоллю Git.
- Створити репозиторій для проекту локально.
- Додавати зміни та фіксувати зміни в репозиторії.

Зміст уроку

- Архітектури систем контролю версій
- Створення репозиторію
- Стан файлів
- Основні команди
- Перший коміт
- Практика

Резюме

- **Репозиторій —** місце, де зберігаються та підтримуються будь-які дані.
- **Git —** розподілена система керування версіями. Проект був створений Лінусом Торвальдсом для управління розробкою ядра Linux, першу версію випущено 7 квітня 2005 року.
- **Система керування версіями (від англ. Version Control System, VCS або Revision Control System)** — програмне забезпечення для полегшення роботи зі змінною інформацією. Система управління версіями дозволяє зберігати кілька версій одного і того ж документа, при необхідності повертатися до більш ранніх версій, визначати, хто і коли зробив ту чи іншу зміну та багато іншого.
- **Локальні системи контролю версій** – зберігають записи про всі зміни файлів локально у базі даних.
- **Централізовані системи контролю версій** - використовують єдиний сервер, що містить усі версії файлів і кілька клієнтів, які отримують файли з цього централізованого сховища. Мінус таких систем - єдина точка відмови, представлена централізованим сервером.
- **Розподілені системи контролю версій** – працюють за допомогою створення повної копії репозиторію. Кожна копія репозиторію є повним бекапом всіх даних. можуть одночасно взаємодіяти з кількома віддаленими репозиторіями.
- **git config** - утиліта для виведення та зміни конфігурації для роботи Git та зовнішнього вигляду. git config --global user.name "Ivan Ivanov" – вказує ім'я користувача
  git config --global user.email ivan@example.com – вказує електронну пошту користувача.

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

- Для створення репозиторію git існує два підходи. Перший – ініціалізувати його локально та відправити на віддалений сервер, другий – ініціалізувати на сервері та клонувати. Коли репозиторій створено, стандартний робочий процес виглядає так:
    1. Ви робите зміни у файлах у своєму робочому каталозі.
    2. Підготовляєте файли, додаючи їх у область файлів готових до комміту (далі називатимемо цю область індексом).
    3. Робите комміт, який поміщає файли з індексу до каталогу Git для зберігання.
- Кожен **коміт** складається зі змін файлів, повідомлення до комміту, де зазвичай вказується коротко які зміни зроблені, дати та часу, автора, хеша та гілки до якої належить коміт. **Хеш** це рядок із 40 шістнадцяткових символів, що обчислюється на основі вмісту файлу або структури каталогу, він унікальний для кожного комміту. Працюючи з Git, ви зустрічатимете такі хеші часто, оскільки у своїй базі даних Git зберігає все не за іменами файлів, а за хешами.
- **git init** – створює репозиторій у поточній директорії (створює нову піддиректорію з ім'ям .git, що містить усі необхідні файли репозиторію).
- **git add** – додає вказані файли під версійний контроль. Іншими словами, додає файли до індексу для наступного комміту.
- **git commit** – використовується для фіксації змін у репозиторії.
- **git diff** – відображення змін, які не були фіксовані виконанням комміту.
- **git status** – використовується для визначення того, які файли в якому стані знаходяться.
- **git log** – використовується для виведення історії комітів. Команда виведе послідовність коммітів із метаданими. Щоб побачити зміни у кожному коміті під час виведення історії використовувати команду **git log -p**.
- **git checkout <commit_hash>** - перемикання на зазначений коміт історія. При такому перемиканні стан репозиторію стає - **detached HEAD**, в цьому стані можна тільки дивитися файли, для того щоб вносити якісь зміни потрібно, щоб HEAD вказував на гілку. (Розгалуження розглядатимуться на наступних заняттях).
- **git help** – виведення довідки, **git help <command>** - введення доступних опцій використання команди.
- Файли в репозиторії поділяються на **відстежувані** (tracked) та **невідстежувані** (untracked).
- Untracked – це новий файл, який був доданий для відстеження командою git add. Всі інші файли мають стан tracked.
- У свою чергу файли, що відстежуються, можуть мати 3 різні стани: **незмінене** (unmodified), **змінене** (modified) и **проіндексоване** (staged).
- Файл знаходиться в незміненому стані, якщо фізично файл у директорії відповідає збереженій копії в сховищі git.
- Файл переходить у змінене, коли він під контролем гіт і він якось модифікувався. Важливий момент у тому, що якщо виконати коміт, то файли в зміненому стані до нього не увійдуть і не будуть зафіксовані..
- Команда git add переводить файли зі станів невідслідковуваного та зміненого у стан проіндексованого і тільки після цієї операції зміни збережуться з виконанням комміту. Цей проміжний стан потрібний, тому що не завжди змінені файли повинні входити в коміт..
- Після коміту гіт статус знову матиме висновок як у першому пункті на слайді.
- **git diff --staged (--cached)** – виведення в консоль змін, які зараз знаходяться в області індексу.
- **git commit -a -m "[message]" –** команда об'єднує дві дії – додавання всіх змін до індексу та виконання комміту (прапор -a показує що всі файли у зміненому стані потрібно проіндексувати перед коммітом).

Закріплення матеріалу

- • Що призвело до необхідності контролю версій?

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

- Наведіть приклади для використання примітивного контролю версій.
- Яка основна перевага розподілених систем контролю версій у порівнянні з централізованими?
- Що потрібно зробити для створення локального репозиторію?
- Де Git зберігає дані про репозиторію?
- Чи можна об'єднати команди додавання до індексу та коміту в одну?

Самостійна діяльність учня

Завдання 1
Створіть локальний репозиторій. Виконайте кілька комітів. Виведіть історію комітів із змінами в кожному коміті.

Завдання 2
Додайте файл до репозиторію, внесіть зміни, потім додайте їх до індексу. Знову змініть файл та виведіть git status. Поясніть результат.

Завдання 3
Пройдіть першу вправу на тренажері Learn Git Branching

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