Лабораторна робота №5

Тема: Робота з системами керування версіями

Мета: Оволодіти базовими навичками по роботі з системами керування версіями на прикладі git.

Короткі теоретичні відомості

Необхідне програмне забезпечення.

В першу чергу треба встановити клієнт git: обов'язково потрібно консольний клієнт, доступний за посиланням http://git-scm.com/downloads (підтримуються основні ОС), графічний клієнт можна встановити за бажанням, виходячи зі своїх переваг. На Unix системах можна скористатися менеджером пакетів (yum на fedora і подібних або apt-get на debian, ubuntu і подібних) замість того, щоб завантажувати установник з сайту.

Далі робота з git буде пояснюватися на прикладі роботи з консольним клієнтом з наступних причин:

- Щоб у вас складалося розуміння того, що відбувається і при виникненні проблем ви могли чітко пояснити, що ви робили, і було видно, що пішло не так.

- Всі натискання кнопок в графічних клієнтів в результаті зводяться до виконання певної команди консольного клієнта, в той же час можливості графічних клієнтів обмежені в порівнянні з консольним клієнтом.

- У тих, хто буде працювати в кабінеті на встановлених там комп'ютерах, не буде іншого вибору, окрім як користуватися консольним клієнтом, оскільки він працює без будь-яких проблем.

Для роботи повинна бути встановлена система контролю версій GIT, при цьому для авторизації використовується публічний ключ id-rsa.pub для авторизації і роботи по ssh.

Якщо у Вас ще немає публічного ключа для доступу до ssh, тоді скористайтеся утилітою ssh-keygen.

В консолі bash (встановлюється разом з гітом) вводимо команду:

ssh-keygen -t rsa -C "your@email.com"

При цьому буде згенеровано пара ключів id_rsa і id_rsa.pub в папці C: \ Users \ YourProfile \ .ssh \, де id_rsa.pub нам потрібен буде далі.

При роботі гітхаб запитує логін і пароль для доступу до сховища (це логін і пароль введені при реєстрації в системі), але залити дані не вийде, тому що у Вас немає прав для управління цим репозиторієм. Для отримання прав, потрібно зайти в налаштування свого сховища «Repositories» -> »Settings» -> »Deploy keys» і вставляємо наш публічний id-rsa.pub ключ для роботи по ssh . Дану процедуру потрібно буде проробляти для кожного нового комп'ютера, з якого ви хочете працювати з проектом.

Аккаунт і репозиторії на github.com

Для роботи необхідно зареєструватися на https://github.com/. Після чого можна буде створювати свої репозиторії або приєднатися до роботи над проектами колег, зробивши fork іншого сховища. Fork - це всього лиш копія сховища. Це те ж саме, що branch в Git. Тільки на GitHub такий branch називається fork - є своя термінологія. Саме слово fork в перекладі означає відгалуження. Для того, щоб скористатися fork, потрібно мати свою власну обліковий запис на GitHub; і потрібно увійти під нею на GitHub в момент виконання fork.

Для чого потрібен fork? Для тих же цілей, що і branch в Git. За допомогою нього створюється точна копія оригінального сховища, тільки на сервісі GitHub. У копії сховища можна вносити свої власні зміни, редагувати файли або видаляти директорії.

Як тільки всі зміни будуть внесені, то можна поділитися ними - відправити авторам оригінального сховища запит на злиття вашого зміненого сховища з використанням їх оригінальним репозиторієм. Такий запит називається pull request.

Якщо авторам оригінального сховища ваші зміни сподобаються, то вони можуть внести їх в свій власний оригінальний репозиторій - прийняти запит і виконати злиття.

Вам пропонується почати зі створення fork-а до заведеного сховища

https://github.com/labsinet/labsinetOKS, де будуть викладатися приклади коду до занять і завдання. Для створення форку слід натиснути на кнопку з права вверху в вашому github профілі в браузері.

Робота з кодом зі сховищ на локальному комп'ютері

Створення локального сховища, пов'язаного з віддаленим репозиторієм

Наступним кроком після створення сховища на github, який назвемо віддаленим репозиторієм, є створення локальної копії цього сховища на своєму комп'ютері. Особливістю git явялется наявність на локальному комп'ютері повної копії сховища з усією інформацією про історію змін.

Відкриваємо консольний клієнт.

На Windows після установки клієнта з'являється пункт Git Bash в контекстному меню папки. Досить перейти в бажану папку і скористатися цим пунктом меню - Git Bash, як показано на рисунку 5.1.

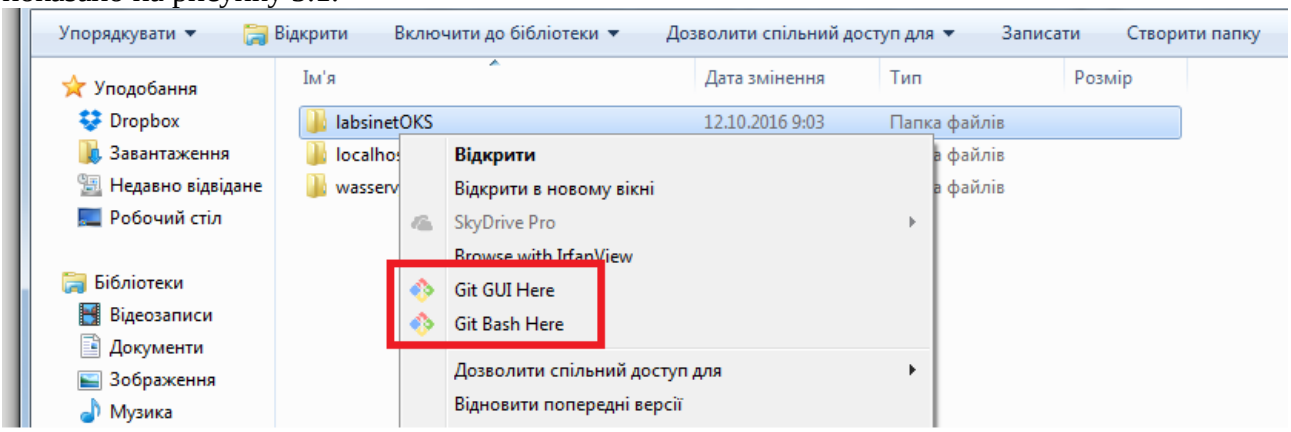

Рисунок 5.1 — Відкрите контекстне меню папки з виділеними пунктами для git

На Unix системах досить відкрити термінал і перейти в потрібну директорію. При стандартній установці консольного клієнта буде доступна команда git без додаткових зусиль.

Виконуємо команду `git clone https://github.com/%user_login%/%repo_name%.git`. Наприклад `$ git clone https://github.com/labsinet/labsinetOKS.git`. Повне https посилання на репозиторій для його викачування можна також знайти на сторінці самого сховища на github. Після цього в цій папці з'явиться нова папка з ім'ям% repo_name%, що містить копію віддаленого (remote) сховища.

Переходимо в новостворене папку сховища та налаштовуємо його:

git config user.name ivan.ivanov

git config user.email ivanov@example.com

Внесення і оформлення змін в локальному сховищі

Скориставшись командою git status можна дізнатися, на який гілці (branch) сховища ви зараз перебуваєте, які зміни присутні у вашій робочої копії і іншу інформацію.

обочою копією називається сукупність файлів в локальній папці сховища за винятком службових файлів.

Після внесення будь-яких змін в робочу копію їх можна закоммітити в локальний репозиторій:

- спочатку потрібна частина змін готується до коммітів з використанням команди
  git add% file_path%
- після чого проводиться Комміт командою git commit.

Використання команди без аргументів відкриє текстовий редактор, де треба буде написати коментар для комміту, Комміт обов'язково повинен мати коментар. Іншим варіантом задання коментаря до комміту є використання команди

git commit -m "% commit_message%"

Історію змін можна подивитися командою git log або git log --name-only. Якщо вся історія змін не вміщується на екрані, то можна користуватися клавішами прокрутки на клавіатурі ("стрілочки", PgUp, PgDown), вихід з режиму перегляду змін здійснюється натисканням клавіші "q".

Завантаження локальних змін в віддалений репозиторій

Після того, як були виконані необхідні локальні комміти, зміни можна завантажити в віддалений репозиторій за допомогою команди

git push origin master.

GIT клієнт при цьому запросить ім'я користувача і пароль для доступу до github. Виконання цієї команди може закінчитися з помилкою, якщо в локально репозиторії відсутні останні зміни, наявні в віддаленому репозиторії. Для вирішення цієї проблеми треба виконати команду git pull, яка завантажить останні зміни з віддаленого сховища та змержить їх з вашими локальними правками, після чого можна повторити команду git push.

Для добавлення комітів в репозиторій слід виконати наступні команди:

git init

git status – виведе поточний статус

git add . - добавить усі змінені файли в коміт, або якщо вказати замість крапки ім'я файлу, то добавить тільки вказані файли

git commit -m “first commit” - виконає добавлення коміта на сервер.

git pull

Дана команда отримує оновлену версію з віддаленого сховища, при цьому вона перевіряє на наявність різних проблем при об'єднанні репозиторіїв і повідомляє про це.

git push

Дана команда цілком передає всі зроблені вами зміни, вже закоміченние в локальний репозиторій, в віддалений репозиторій. Для передачі тегів необхідно використовувати аргумент -tags

git push --tags

Базові операції

Для базової роботи з будь-якою системою контролю версій потрібно не особливо великий набір операцій: додавання файлу в репозиторій, видалити файл зі сховища, комміт змін в репозиторій, скасування незакоміченних змін і отримання списку змін. Додавання списку файлів в коміт:

git add file1 file2 ... fileN

Додавання всіх недобавленних файлів в комміт:

git add -a

Видалення файлу з комміта:

git rm file1 file2 ... fileN

Видалення файлу з комміту і з жорсткого диска:

git rm -f file1 file2 ... fileN

Комміт в локальний репозиторій (треба відзначити, що в такому випадку закоммітятся тільки файли, які були оброблені за допомогою git add / rm):

git commit

Комміт всіх змін в локальний репозиторій:

git commit -a

Скасування всіх змін, зроблених в дереві, до стану, який був при останньому комітети в локальний репозиторій (дуже небезпечна команда, подумайте перш ніж користуватися нею):

git reset —hard

Створення Діфа щодо останнього коміта:

git diff

Використання гілок

Рано чи пізно в будь-якому проекті виникає ситуація, коли потрібно заморозити зміни,

але продовжувати працювати, а на заморожені зміни накладати тільки баг-фікси. Для цієї мети служать гілки (branch)

У гіті можна створити гілку від будь-якого місця. Для створення гілки від основного дерева треба виконати наступну команду:

git checkout --track -b name_of_newbranch origin / master

В результаті цієї команди ви побачите приблизно таке повідомлення

Branch name_of_newbranch set up to track remote branch refs / remotes / origin / master.

Switched to a new branch "name_of_newbranch"

Це означає, що в локальному сховищі у вас створилася нова гілка. Якщо в цій команді замінити origin / master на origin / remote_branch_name то ви створите гілку від іншої гілки.

Щоб ваша гілка була видна всім, її потрібно пропхати в віддалений репозиторій. Робиться це так:

git push origin local_branch_name: remote_branch_name

Зрозуміло, треба також уміти і отримувати гілки в своє розпорядження

git branch local_branch_name origin / remote_branch_name

git checkout local_branch_name

В результаті ви отримаєте шукану гілку після наступного повідомлення

Switched to branch "local_branch_name"

Робота з тегами

Як правило, крім гілок розробники використовують теги - щоб запам'ятати стан коду в якийсь момент. Тег - це своєрідний зліпок, точно ідентифікує стан коду. Гіт вміє працювати з підписаними GPG тегами і з підписаними. Тут розглянемо тільки непідписані теги. Для створення такого тега необхідно виконати команду

git tag

Щоб прибрати тег необхідно виконати

git tag -d

Для того, щоб тег стало видно всім, необхідно відправити його в віддалений репозиторій

git push —tags

Щоб отримати версію з конкретного тега необхідно створити від нього локальну гілку і розчекаутити цю гілку:

git fetch origin tag

git branch

git checkout

Зрозуміло, в майбутньому цю гілку можна буде зробити глобальною і вислати в віддаленій репозиторій.

Налаштування git

Для прискорення деяких операцій і збільшення зручності роботи можна провести пару налаштувань:

Налаштування кольорового виводу:

git config --global color.ui "auto"

Налаштування імені користувача та поштової адреси (між іншим, це хороший тон):

git config user.name "FirstName LastName"

git config user.email "user@example.com"

Прискорюємо діфи і скасовуємо обмеження на кількість потоків упаковки при push:

git config --global diff.renamelimit "0"

git config --global pack.threads "0"

Git - дуже потужна і зручна система контролю версій. Для неї існує кілька GUI утиліт, які можуть полегшити роботу, кілька веб інтерфейсів для моніторингу поточного стану, а

також є включення даної системи в IDE, наприклад від JetBrains та Microsoft. Останнім часом все більшу кількість проектів переходять на використання git, і це показник того, що git успішно розвивається і відповідає останнім вимогам в області систем контролю версій.

Завдання до виконання роботи

1. Зареєструватися на github.
2. Знайти проект https://github.com/labsinet/labsinetOKS
3. Налаштувати git під себе, ввівши імя та прізвище.
4. Зробити копію проекту (fork).
5. Добавити розроблену html сторінку в репозиторій.
6. Добавити файл readmy.md до проекту і закоммітити його.
7. Вивчити можливості одиночного добавлення файлів, добавлення всіх файлів проекту, перегляду стану, видалення файлів з репозитарію, добавлення тегів, відміну змін, отримання змінених файлів.
8. Оформити звіт.

Контрольні запитання

1. Перерахуйте сучасні системи керування версіями.
2. Як сформувати і добавити власний ключ для github.
3. Як сконфігурувати git?
4. Як добавити файл в комміт?
5. Як передати файл в репозиторій?
6. Як повідомити про внесені зміни?
7. Як прийняти файли з репозиторію?
8. Як поглянути статус ?
9. Як проглянути історію змін?
10. Як об'єднати дані в репозиторії?
11. Як видалити файли з репозиторію?