GIT. How to

Розділ 1. Install it!

Linux

```
yum install git
```

```
apt-get install git
```

```
zypper in git
```

Mac

Git - Downloading Package

The entire Pro Git book written
by Scott Chacon and Ben Straub is available to read online for free.
Dead tree versions are available on Amazon.com.

http://git-scm.com/download/mac

Windows

Git for Windows

We bring the awesome Git VCS to Windows

http://git-for-windows.github.io/

Розділ 1: Підготовка

1. Встановлення імені та електронної пошти

Якщо ви ніколи раніше не використовували Git, спершу вам потрібно встановити його. Виконайте наступні команди, щоб Git знав ваше ім'я та електронну пошту. Якщо Git вже встановлено, можете перейти до розділу "2. Фінальна підготовка".

```
ВИКОНАТИ:
git config --global user.name "Ваше ім'я"
git config --global user.email "ваша_електронна_пошта@домен.com"
```

2. Налаштування параметрів завершення рядків

Для користувачів Unix/Mac/Linux:

```
git config --global core.autocrlf input
git config --global core.safecrlf warn
```

Для користувачів Windows:

```
git config --global core.autocrlf true
git config --global core.safecrlf warn
```

3. Встановлення відображення Unicode

За замовчуванням, Git буде виводити не-ASCII символи в назвах файлів у вигляді восьмеричних послідовностей \nnn. Щоб уникнути нерозбірливих рядків, встановіть відповідний прапорець:

```
git config --global core.quotepath off
```

Розділ 2: Робота з проектом

1. Створення «Hello, World»

Почніть роботу в порожньому робочому каталозі. Створіть порожній каталог з іменем "lesson_git", а потім перейдіть у нього та створіть файл з іменем "hello.html" з таким вмістом.

```
mkdir lesson_git
cd lesson_git
touch hello.py
```

ФАЙЛ: hello.py

```
print("hello world")
```

2. Створення репозиторію

Тепер у вас є каталог з одним файлом. Щоб створити Git-репозиторій з цього каталогу, виконайте команду "git init".

```
git init
```

РЕЗУЛЬТАТ:

```
$ git init
Initialized empty Git repository in ВашаАдреса/.../lesson_git/.git/
```

3. Додавання сторінки до репозиторію

Тепер додамо сторінку "Hello, World" до репозиторію.

```
git add hello.py
git commit -m "Перший коміт"
```

РЕЗУЛЬТАТ:

```
$ git add hello.py
$ git commit -m "Перший коміт"
[master (root-commit) 911e8c9] Перший коміт
```

```
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 hello.py
```

Розділ 3: Перевірка стану

1. Перевірка стану репозиторія

Використовуйте команду "git status", щоб перевірити поточний стан репозиторія.

```
git status
```

РЕЗУЛЬТАТ:

```
$ git status
# On branch master nothing to commit (working directory clean)
```

Команда перевірки стану повідомить, що немає чого комітувати. Це означає, що в репозиторії зберігається поточний стан робочого каталогу, і немає жодних змін, які очікують на запис.

Ми будемо використовувати команду "git status" для подальшого відстеження стану репозиторія та робочого каталогу.

Розділ 4: Внесення змін у файл

1. Змініть сторінку «Hello, World»

Додайте деякі рядків до нашого вітання. Змініть вміст файлу на такий:

ФАЙЛ: hello.py

```
name = input("My name is: ")
```

```
print(f"hello,{name}")
```

2. Перевірте стан

Тепер перевірте стан робочого каталогу.

```
git status
```

РЕЗУЛЬТАТ:

```
$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#   modified:   hello.html
#
no changes added to commit (use "git add" and/or "git commit -a")
```

Перше, на що варто звернути увагу, це те, що Git знає про зміни у файлі , але ці зміни ще не внесені до репозиторію.

Також зверніть увагу на повідомлення про стан, яке дає вам підказку щодо подальших дій. Якщо ви хочете додати ці зміни до репозиторію, використовуйте команду "git add". В іншому випадку, використовуйте команду "git checkout" для скасування змін. Але про це далі

Розділ 5: Індексація змін

1. Додайте зміни

Тепер дайте команду Git індексувати зміни. Перевірте стан.

```
ВИКОНАТИ:
git add hello.py
```

```
git status
```

РЕЗУЛЬТАТ:

```
$ git add hello.html
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
# modified:   hello.py
#
```

Зміни в файлі "hello.py" були індексовані. Це означає, що Git тепер знає про зміни, але ці зміни ще не постійно (назавжди) включені до репозиторію. Наступний коміт включить індексовані зміни.

Якщо ви вирішили, що не хочете комітувати ці зміни, команда стану нагадає вам, що за допомогою команди "git reset" можна скасувати індексацію цих змін.

Розділ 6: Індексація та коміт

Відокремлений етап індексації в Git дозволяє вам продовжувати внесення змін до робочого каталогу, а потім, коли ви вирішите взаємодіяти з системою контролю версій, Git дозволить записати ці зміни у невеликих комітах, які фіксують те, що ви зробили.

Допустимо, ви відредагували три файли (a.html, b.html і c.html).Так, ми можемо зберігати та оперувати будь-якими файлами. Тепер ви хочете закомітити всі зміни, причому зміни в a.html і b.html мають бути одним комітом, в той час як зміни в c.html логічно не пов'язані з першими двома файлами і повинні йти окремим комітом.

У теорії ви можете зробити так:

```
git add a.html
git add b.html
git commit -m "Зміни для a і b"
```

```
git add c.html
git commit -m "Неспоріднена зміна для c"
```

Розділяючи індексацію та коміт, ви маєте можливість легко налаштувати, що йде в який коміт.

Розділ 7: Дісклеймер

Ми можемо зберігати на гід абсолютно всі файли, які ми використовуємо нашому проекті. Однак не слід забувати, що серверний простір виділений під проект може бути обмежене, тому беріть за практику використання не самих файлів, а посилання на них.

Розділ 8: Комітування змін

Мета: Навчитися комітувати зміни в репозиторій.

1. Закомітьте зміни

Після індексації ми можемо зробити комміт того, що ми проіндексували, в репозиторій.

Коли ви раніше використовували git commit для коміту початкової версії файлу "hello.html" в репозиторій, ви включили прапорець -m, який дозволяє вам ввести коментар у командному рядку. Команда commit також дозволяє вам інтерактивно редагувати коментарі для коміту. Давайте перевіримо це зараз.

Якщо ви пропустите прапорець -m з командного рядку, Git перенесе вас до редактора на ваш вибір. Редактор вибирається з наступного списку (в порядку пріоритету):

1. Змінна середовища GIT_EDITOR.
2. Параметр конфігурації core.editor.
3. Змінна середовища VISUAL.
4. Змінна середовища EDITOR.

У мене змінна середовища EDITOR налаштована на "emacsclient" (доступний для Linux і Mac).

Зробіть коміт зараз і перевірте стан.

```
ВИКОНАТИ:
git commit
```

Ви побачите у своєму редакторі:

```
РЕЗУЛЬТАТ:
#
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
# modified:   hello.html
#
```

У першому рядку введіть коментар: "Додано тег h1". Збережіть файл і вийдіть з редактора (для цього в редакторі за замовчуванням (Vim) вам потрібно натиснути клавішу ESC, ввести :wq і натиснути Enter).

```
РЕЗУЛЬТАТ:
git commit
Waiting for Emacs...
[master 569aa96] Додано тег h1
 1 files changed, 1 insertions(+), 1 deletions(-)
```

Рядок "Waiting for Emacs..." отримується з програми emacsclient, яка надсилає файл у запущену програму Emacs і очікує його закриття. Інші виходи - стандартні коментарі для коміту.

2. Перевірте стан

Завершивши, давайте ще раз перевіримо стан.

```
ВИКОНАТИ:
git status
```

Ви побачите:

```
РЕЗУЛЬТАТ:
$ git status
# On branch master
nothing to commit (working directory clean)
```

Робочий каталог чистий, і ви можете продовжити роботу.