GIT

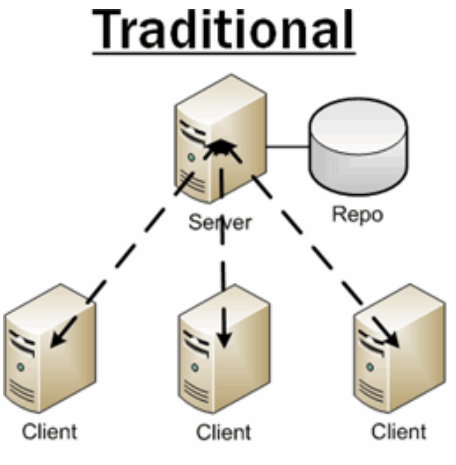
SCM - Source code managment

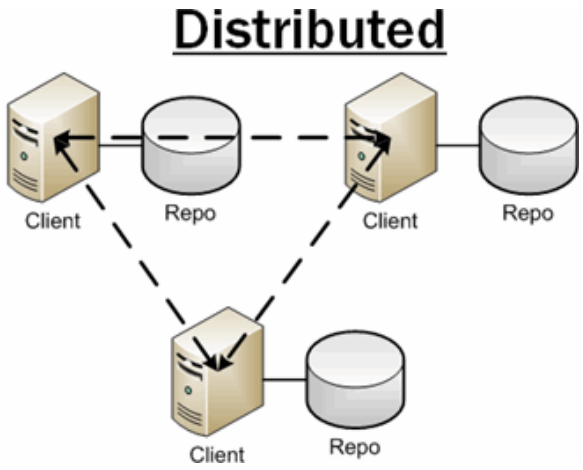
Szkolenie Rails

System kontroli wersji: oprogramowanie służące do śledzenia zmian oraz pomocy programistom w łączeniu zmian dokonanych w plikach przez wiele osób w różnych momentach czasowych.

1. Teoria:

- ❏ Stworzony przez twórcę Linuksa - Linusa Torvaldsa.
- ❏ Szybki
- ❏ Zajmuje mniej miejsca w porównaniu do innych rozwiązań:
  Zamiast zapisywać i podmieniać całe pliki, śledzi i zapisuje tylko różnice.
- ❏ Bezpieczny (rozproszony)





2. Terminologia:

- ❏ **master -** główna domyślna gałąź repozytorium.
- ❏ **clone -** kopiuje istniejące repozytorium git, zwykle ze zdalnej lokacji na twój dysk lokalny.
- ❏ **commit -** tworzenie “rewizji”, zatwierdzanie zmian w repozytorium (lokalnym).

- **fetch or pull -** **“**uaktualnij” lub pobierz najnowszą wersję, różnica pomiędzy fetch a pull to, polega na tym, że pull oprócz tego, że pobiera najnowszy kod z repozytorium to merguje zmiany z naszą rewizją.
- **push -** jest używany do wysyłania kodu na zdalne repozytorium.
- **remote -** remote’y to zdalne lokacje twoich repozytoriów.
- **SHA -** każdy commit w gicie jest identyfikowany unikalnym hashem (kluczem) SHA. Możesz ich używać w różnych poleceniach, aby manipulować wybraną rewizję.
- **head -** jest odwołaniem do obecnego obszaru roboczego w którym wstawiamy zmiany (“węzeł przed commitem”).
- **branch -** gałąź jest lekkim, przesuwalnym wskaźnikiem na któryś z owych zestawów zmian.
- **.gitignore** - Plik w którym możemy ustawić ignorowanie śledzenia zmian dla pewnych plików czy katalogów. Te pliki lub katalogi nie mogą być jednak zapisane w repozytorium. Jeśli są już w repozytorium, to trzeba je usunąć i zatwierdzić zmiany przez git-commit -a.

3. Instalacja i konfiguracja:

```
$ apt-get install git
```

```
$ git config --global user.name "John Bean"an
$ git config --global user.email john@bean.com
```

```
git config --global color.ui true
```

4. Podstawy:

- Zainicjowanie nowego repozytorium:

  ```
  $ git init project_name
  ```

  lub

  ```
  $ cd project_name
  $ git init .
  ```

  Utworzy nam to folder .git w którym zawiera się cała obsługa repozytorium.

- Podstawowe komendy:

```
$ git status
```

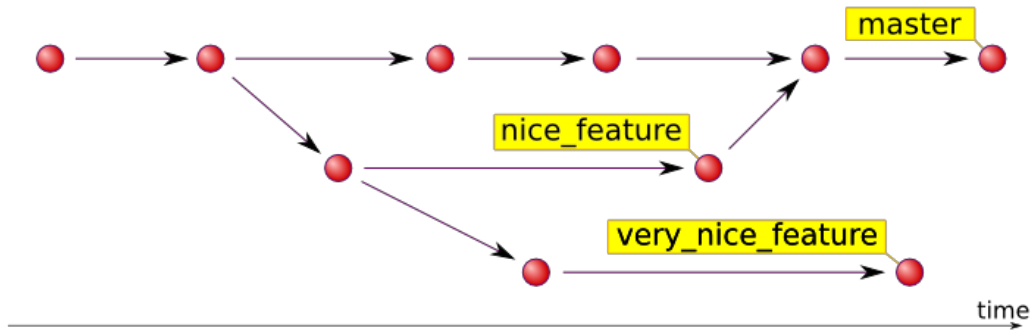
Śledź plik
```
$ git add file.ext
```
Śledź wszystkie pliki w katalogu
```
$ git add .
```
Commit dla plików dodanych do “staged to be commited”
```
$ git commit -m “initial commit“
```
Commit dla wszystkich śledzonych plików
```
$ git commit -am “initial commit“
```

Stwórz kopię repozytorium (lokalnego lub zdalnego)
```
git clone [ścieżka lub url]
```

Wyślij zmiany ze swojego brancha master do zdalnego repozytorium
```
git push origin master
```
Połącz się ze zdalnym repozytorium
```
git remote add origin <server>
```
Lista zdalnych repozytoriów
```
git remote -v
```

5. Branche:

Branching - “Rozgałęzienie” oznacza odbicie od głównego pnia linii rozwoju i kontynuację pracy bez wprowadzania tam bałaganu:



Utworzenie brancha i przejście do niego
```
git checkout -b nazwa branch
```
Przejście z jednego brancha na drugi
```
git checkout nazwa branch
```
Wyświetlenie wszystkich gałęzi

```
git branch
```
Usunięcie wybranej gałęzi
```
git branch -d nazwa gałęzi
```
wysłanie gałęzi na zdalne repozytorium
```
git push origin nazwa branch
```
Przesłanie wszystkich branchy do zdalnego repozytorium
```
git push --all origin
```
Usunięcie brancha na zdalnym repozytorium
```
git push origin :nazwa branch
```
Połącz “zmerguj” inną gałąź z tą na której aktywnie jesteś
```
git merge nazwa brancha
```
wyświetl wszystkie konflity merga
```
git diff
```
Po rozwiązaniu konfliktów należy dodać zmieniony plik
```
git add file name
git commit -m’resolved merge conflict
```

6. Cofanie zmian:

Poprawka do ostatniej rewizji
```
git commit --amend
```
Dołączy pliki “staged to commit” do ostatniego commita, pozwala również na zmianę notki.

Cofanie zmian w zmodyfikowanym pliku, który nie jest dodany do poczekalni - nie jest “staged to commit”

```
git checkout nazwa pliku
```

Wyrzucenie pliku z poczekalni do “unstaged”
```
git reset HEAD nazwapliku
```

Cofanie operacji ostatniego commita
```
git reset HEAD~1 --soft
```

Jeżeli jednak wysłaliśmy już niechcianego commita do zdalnego repo
lub chcemy cofnąć “odwołać” któregoś ze wcześniejszych commitów użyjemy polecenia revert
```
git revert HEAD~1
```

lub
```
git revert [SHA]
```

7. Serwisy / Serwery do przechowywania i zarządzania repozytoriami git:

- ❏ Github.com
- ❏ Bitbucket.org
- ❏ SCM Manager https://www.scm-manager.org/

(Warto wiedzieć że można przechowywać repozytoria również na dropboxie)

Instrukcja generowanie klucza ssh dla githuba:
https://help.github.com/articles/generating-ssh-keys

8. Dokumentacja:

http://git-scm.com/book/pl

HEROKU

Platforma chmurowa stworzona w modelu PaaS (Platform as a Service) obsługująca kilka języków programowania.
Heroku jest jedną z pierwszych tego typu platform. Rozwijana była od czerwca 2007, kiedy udostępniała tylko język Ruby.

1. Instalacja:

Zainstaluj heroku toolbelt:

https://toolbelt.heroku.com/

zaloguj się (automatycznie zostanie wygenerowany i przesłany klucz ssh):

```
$ heroku login
```

2. Przygotowanie serwera bazy postgresql na ubuntu:

```
sudo apt-get install postgresql postgresql-contri
```

```
sudo apt-get install postgresql-clien
```

```
sudo -u postgres psql postgre
```

następnie:

```
\password postgres
```

```
sudo -u postgres createuser -D -P system current user nar
```

#proponuję stworzyć tego użytkownika bez hasła: 2 razy enter i yes.

W Gemfile:

dodaj wersję ruby z której korzystasz:

```
ruby '2.1.1'
```

usuń gem sqlite3
dodaj

```
gem 'pg'
```

```
gem 'rails_12factor', group: :productio
```

konfiguracja pliku database.yml

```
development:
```

```
  adapter: postgresql
```

```
  encoding: unicode
```

```
  database: dbname_dev
```

```
  pool: 5
```

3. Deployment na heroku:

```
$ heroku create --addons heroku-postgresq
```

```
$ git push heroku master
```

Migracja bazy danych na heroku:

```
$ heroku run rake db:migrate
```