МИРЭА
Российский технологический университет

Технологии Разработки программных Приложений

Практические занятия

Жматов Дмитрий Владимирович
кандидат технических наук, доцент
доцент кафедры Математического обеспечения и стандартизации информационных технологий

Git – свободная распределенная система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux.

С точки зрения реализации Git представляет собой набор утилит командной строки, работа которых может управляться параметрами.

Git - это консольная утилита, для отслеживания и ведения истории изменения файлов, в вашем проекте. Чаще всего его используют для кода, но можно и для других файлов.

Windows версия Git используется пакет MSYS - эмулятор POSIX-совместимой командной строки.

Git

В рамках курса "Основы программной инженерии" мы будем использовать Git в командной строке.

- Командная строка единственное место, где доступны *все* команды Git
- Если вы знаете, как выполнить какое-либо действие в командной строке, вы сможете выяснить, как то же самое сделать и в GUI-версии.

Git: установка

http://www.msys2.org

- Скачать программу-инсталлятор оболочки (shell) msys2 и запустить ее.
- Выполнить обновление оболочки:
  - **pacman -Syuu** (обычно несколько раз)
- Установить следующие пакеты
  - **pacman -S git**
  - **pacman -S man**

Git: самая главная команда

Если вам нужна помощь при использовании Git, есть три способа открыть страницу руководства по любой команде Git:

- git help <глагол>
- git <глагол> --help
- man git-<глагол>

Задача

- Реализовать консольную программу для ввода и вывода целочисленного массива.
- Максимальное количество элементов в массиве равно 15.

Создание репозитория

Руководитель проекта:

- создает на удаленном сервере новый репозиторий;
- регистрирует разработчиков;
- выдает разработчикам информацию, необходимую для доступа к репозиторию:
  - URL проекта;
  - имя и пароль пользователя.

Создание репозитория

В примерах, которые разбираются ниже, используются следующие данные:

- URL: http://git.iu7.bmstu.ru/ilomovskoy/demo_X.git
- Разработчик 1: ilomovskoy
- Разработчик 2: tstudent

URL проекта состоит из двух частей:

- http://git.iu7.bmstu.ru - это адрес, по которому расположен сервер;
- ilomovskoy/demo_X.git - это имя репозитория (проекта).

Получение рабочей копии первым разработчиком

Для получения рабочей копии используется команда *clone*.

Команда

```
git clone http://git.iu7.bmstu.ru/ilomovskoy/demo_X.git
```

Результат

```
Клонирование в «demo_0»…
Username for 'http://git.iu7.bmstu.ru':
Password for 'http://ilomovskoy@git.iu7.bmstu.ru':
warning: Похоже, что вы клонировали пустой репозиторий.
```

Получение рабочей копии первым разработчиком

До “checkout” После “checkout”

```
/work          /work
              /demo_X
                /.git  <- скрытый каталог
```

В каталоге work появляется каталог demo_X. Он содержит рабочую копию проекта. Пока в рабочей копии ничего нет.

Каталог .git это локальный репозиторий GIT. Его нельзя удалять!

Базовая версия программы, созданная первым разработчиком

```
arr = list()
n   = int(input("Enter number of elements: "))
print("Enter elements:")
i = 0
while (i < n):
  tmp = int(input(""))
  arr.append(tmp)
  i += 1

print("Array:")
i = 0
while (i < n):
  print(arr[i], end = " ")
  i += 1
print("")
```

Добавление начальной версии проекта первым разработчиком

Поместим каталог example внутрь рабочей копии, предварительно избавившись от лишнего.

```
/work
  /demo_X
    /example
      array.py
```

Git: состояния файлов

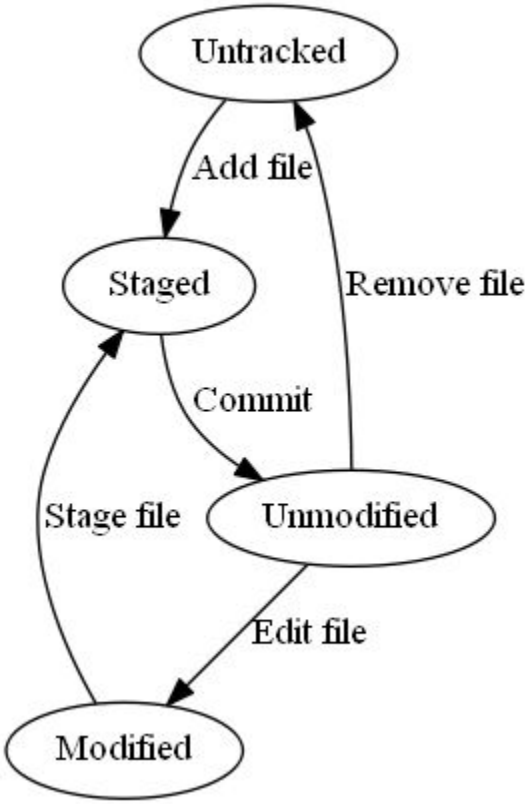
С точки зрения git файлы, находящиеся в рабочей копии, могут находиться в следующих состояниях:

- отслеживаемые (под версионным контролем);
- неотслеживаемые.

Отслеживаемые файлы, в свою очередь, могут находиться в следующих состояниях:

- зафиксированное (committed);
- измененное (modified);
- подготовленное (staged/cached).

Git: состояния файлов



Untracked	Неотслеживаемый
Staged	Подготовленный
Unmodified	Зафиксированный
Modified	Измененный
Add file	Планирование для включения в фиксацию
Stage file	
Edit file	Изменение файла
Remove file	Удаление из-под верс. контр.
Commit	Фиксация

Добавление начальной версии проекта первым разработчиком

Узнать в каком состоянии находится файл можно с помощью команды *status*.

Команда

```
git status
```

Результат

```
На ветке master

Начальный коммит

// секция Untracked files
Неотслеживаемые файлы:
(используйте «git add <файл>…», чтобы добавить в то, что будет...

example/
...
```

Добавление начальной версии проекта первым разработчиком

Указать GIT какие каталоги и/или файлы нужно добавить под версионный контроль можно с помощью команды *add*.

Команды

```
git add example // результат никак не отображается
git status
```

Результат

```
На ветке master

Начальный коммит

// секция Changes to be commited
Изменения, которые будут включены в коммит:
  (используйте «git rm --cached <файл>…», чтобы убрать из индекса)

        новый файл:    example/array.py
```

Добавление начальной версии проекта первым разработчиком

Для фиксации изменений в локальном репозитории используется команда *commit*.

Команда

```
git commit -m "Initial version of example."
```

Результат

```
*** Пожалуйста, скажите мне кто вы есть.
Запустите
  git config --global user.email "you@example.com"
  git config --global user.name "Ваше Имя"
для указания идентификационных данных аккаунта по умолчанию.
Пропустите параметр --global для указания данных только для этого репозитория.
fatal: unable to auto-detect email address (got ...
```

Добавление начальной версии проекта первым разработчиком

Команды

```
git config user.name IgorL
// результат никак не отображается
git config user.email ilomovskoy@bmstu.ru
// результат никак не отображается
git commit -m "Initial version of example."
```

Результат

```
[master (корневой коммит) 7e7813f] Initial version of example.
 1 file changed, 19 insertions(+)
 create mode 100644 example/array.py
```

Добавление начальной версии проекта первым разработчиком

- Git для идентификации ревизий использует значение хэша фиксации. Главная причина этого - Git децентрализованная система контроля версий и поэтому монотонной сквозной нумерации фиксации в ней быть просто не может

.

- Важно сопровождать фиксации комментариями, которые кратко раскрывают суть изменений. Эти комментарии помогут вам или вашим коллегам понять, что фиксация сделала для проекта.

«Публикация» изменений первым разработчиком

Для отправки изменений в удаленный репозиторий используется команда *push*.

Команда

```
git push
```

Результат

```
Username for 'http://git.iu7.bmstu.ru':
Password for 'http://ilomovskoy@git.iu7.bmstu.ru':
Подсчет объектов: 4, готово.
Delta compression using up to 4 threads.
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (4/4), 411 bytes | 0 bytes/s, готово.
Total 4 (delta 0), reused 0 (delta 0)
To http://git.iu7.bmstu.ru/ilomovskoy/demo_0
 * [new branch]      master -> master
```

Внесение изменений в проект вторым разработчиком

Работа над проектом начинается с получения рабочей копии.

Команда

```
git clone http://git.iu7.bmstu.ru/ilomovskoy/demo_X.git
```

Результат

```
Клонирование в «demo_0»…
Username for 'http://git.iu7.bmstu.ru':
Password for 'http://tstudent@git.iu7.bmstu.ru':
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 0), reused 0 (delta 0)
Распаковка объектов: 100% (4/4), готово.
```

Внесение изменений в проект вторым разработчиком

Проект оказывается не пустым, для анализа истории изменений проекта используется команда *log*.

Команда

```
git log --name-status
```

Результат

```
commit 7e7813f919bcc47a9572cd4d488eaae6ce31aca0
Author: IgorL <ilomovskoy@bmstu.ru>
Date:   Mon Jan 30 18:22:40 2017 +0300
    Initial version of example.
A       example/array.py
```

Внесение изменений в проект вторым разработчиком

tstudent обнаружил ошибку, исправил ее и собирается зафиксировать изменения.

Команда

```
git status
```

Результат

```
На ветке master
Ваша ветка обновлена в соответствии с «origin/master».
Изменения, которые не в индексе для коммита:
  (используйте «git add <файл>…», чтобы добавить файл в индекс)
  (используйте «git checkout -- <файл>…», чтобы отменить изменения
   в рабочем каталоге)
        изменено:      example/array.py
нет изменений добавленных для коммита
(используйте «git add» и/или «git commit -a»)
```

Внесение изменений в проект вторым разработчиком

Для анализа самих изменений служит команда *diff*.

Команда

```
git diff
```

Результат

См. файл diff_1.txt

diff:универсальный формат

- Минусами помечены строки из первого файла, а плюсами - из второго.
- Информация о диапазоне измененных строк (номер, количество) отмечены знаками @@.
- Слова, общие для двух файлов ничем не отмечены.
- Знаком минус помечены строки, которые есть только в первом файле, как бы изъятые из первого файла, если считать его эталонным.
- Знаком плюс помечены строки, которых нет в первом файле, как бы добавленные к нему.

Внесение изменений в проект вторым разработчиком

Зафиксируем изменения и опубликуем их.

```
git add example/array.py
git commit -m "Fix possible array overflow."
[master b96ebd0] Fix possible array overflow.
 1 file changed, 15 insertions(+), 13 deletions(-)

git push
Username for 'http://git.iu7.bmstu.ru':
Password for 'http://tstudent@git.iu7.bmstu.ru':
Подсчет объектов: 4, готово.
Delta compression using up to 4 threads.
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (4/4), 481 bytes | 0 bytes/s, готово.
Total 4 (delta 0), reused 0 (delta 0)
To http://git.iu7.bmstu.ru/ilomovskoy/demo_0.git
   7e7813f..b96ebd0  master -> master
```

Конфликт

ilomovskoy реализовал функции для ввода и вывода массива, протестировал программу и решил зафиксировать свои изменения.

Для анализа изменений разработчика ilomovskoy воспользуемся командой *diff*.

Команда

```
git diff
```

Результат

См. файл diff_2.txt

Конфликт

Фиксация и публикация изменений.

```
git add example/array.py
git commit -m "IO functions were added."
[master 93f689d] IO functions were added.
 1 file changed, 30 insertions(+), 19 deletions(-)
 rewrite example/array.py (97%)

git push
Username for 'http://git.iu7.bmstu.ru':
Password for 'http://ilomovskoy@git.iu7.bmstu.ru':
To http://git.iu7.bmstu.ru/ilomovskoy/demo_0
 ! [rejected]        master -> master (fetch first)
error: не удалось отправить некоторые ссылки в
               «http://git.iu7.bmstu.ru/ilomovskoy/demo_0»
подсказка: Обновления были отклонены, так как внешний репозиторий
подсказка: содержит изменения, которых у вас нет в вашем
подсказка: локальном репозитории ...
```

Конфликт

Для обновления рабочей копии используется команда pull.

Команда

```
git pull
```

Результат

```
Username for 'http://git.iu7.bmstu.ru':
Password for 'http://ilomovskoy@git.iu7.bmstu.ru':
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 0), reused 0 (delta 0)
Распаковка объектов: 100% (4/4), готово.
Из http://git.iu7.bmstu.ru/ilomovskoy/demo_0
   7e7813f..b96ebd0  master     -> origin/master
Автослияние example/array.py
КОНФЛИКТ (содержимое): Конфликт слияния в example/array.py
...
```

Конфликт

Результат объединений изменений разработчиков ilomovskoy и tstudent находится в .файле conflict.txt.

Конфликт

Проверив правильность сделанных изменений, сообщим git, что конфликт разрешен с помощью команды commit, предварительно добавив измененный файл в индекс снова.

```
git add example/array.py
git commit -m "Merging with remote branch was done."
[master fa93035] Merging with remote branch was done.

git push
Username for 'http://git.iu7.bmstu.ru':
Password for 'http://ilomovskoy@git.iu7.bmstu.ru':
Подсчет объектов: 8, готово.
Delta compression using up to 4 threads.
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (8/8), 884 bytes | 0 bytes/s, готово.
Total 8 (delta 1), reused 0 (delta 0)
To http://git.iu7.bmstu.ru/ilomovskoy/demo_0
   b96ebd0..fa93035  master -> master
```

Откат локальных изменений

tstudent внес в свою рабочую копию какие-то изменения и программа перестала собираться. Если изменения еще не помещены в индекс, их можно «откатить» с помощью команды checkout.

```
git diff
diff --git a/example/array.py b/example/array.py
index 443ceac..4fdccf0 100644
--- a/example/array.py
+++ b/example/array.py
@@ -1,3 +1,5 @@
+Blah Blah Blah^M
+^M
 N_MAX = 15
 def read_array():

git checkout .
git diff
```

Откат локальных изменений

Если изменения попали в индекс, их можно «откатить» с помощью команды *reset HEAD имя_файла*.

Что такое система управления версиями ?

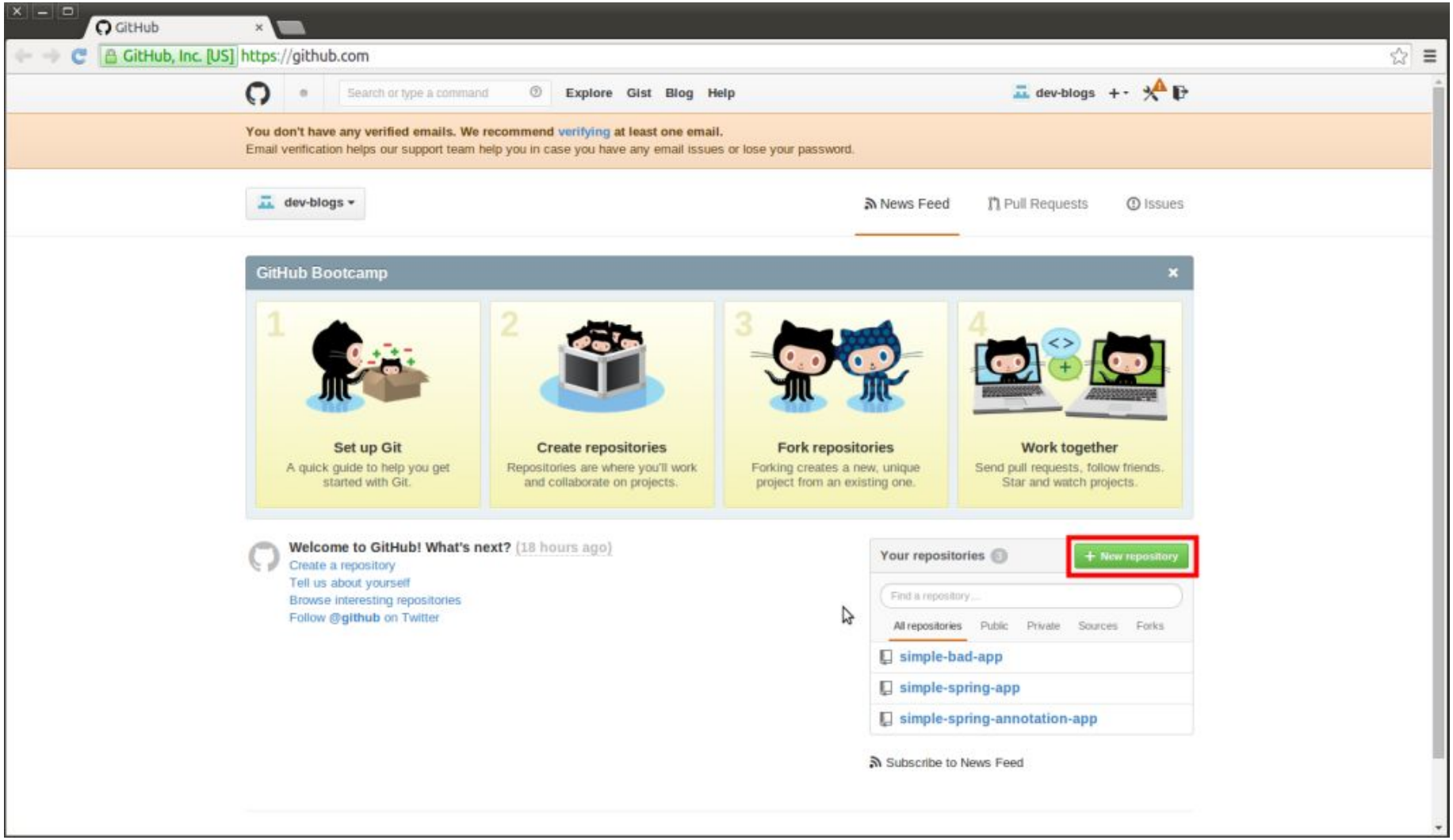
от англ. *Version Control System, VCS* или *Revision Control System*

позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение

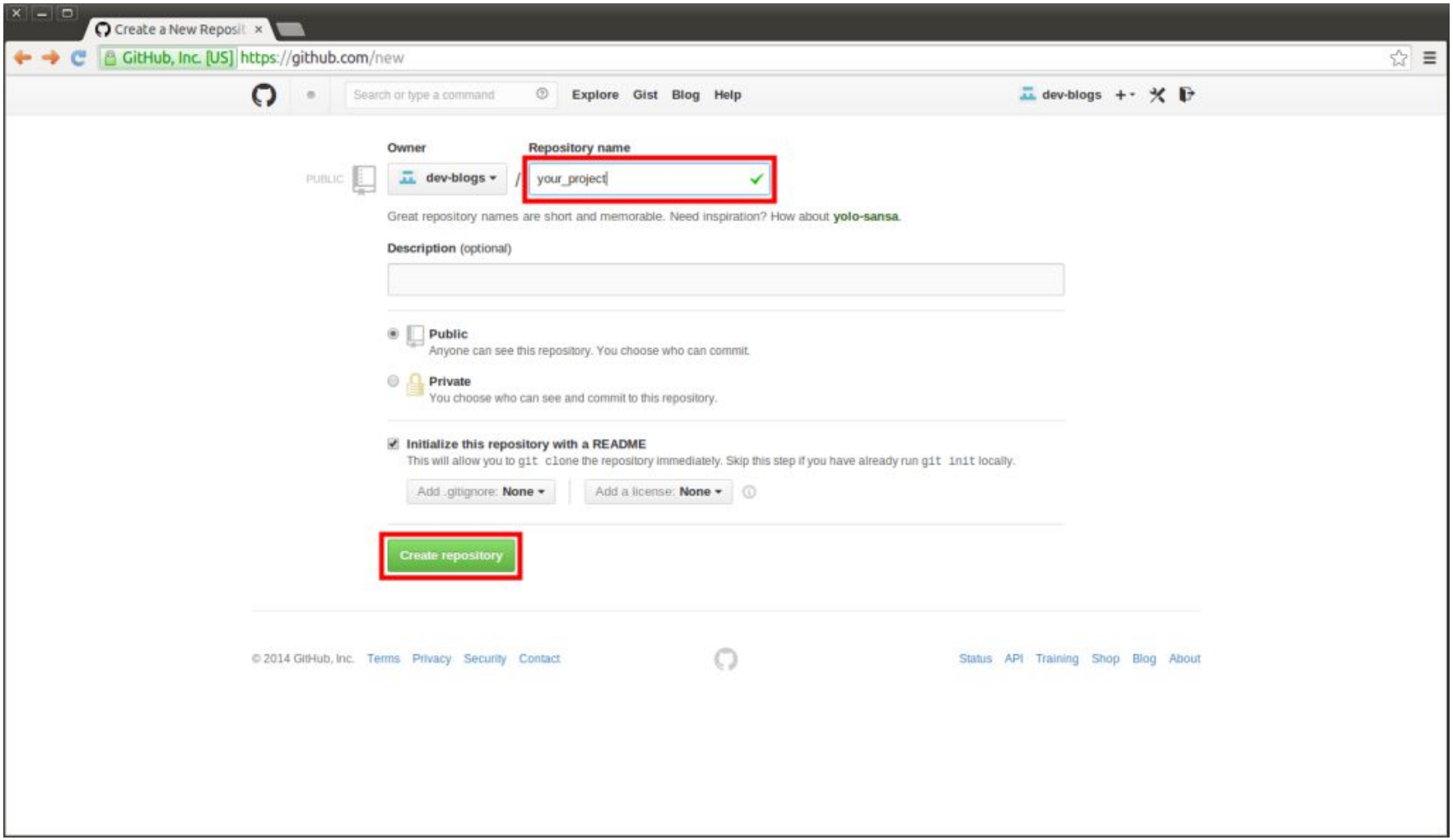
Быстрый старт

```
git init
git add your_file
git commit -m "first commit"
git remote add origin https://github.com/REP/PROJ
git push -u origin master
```

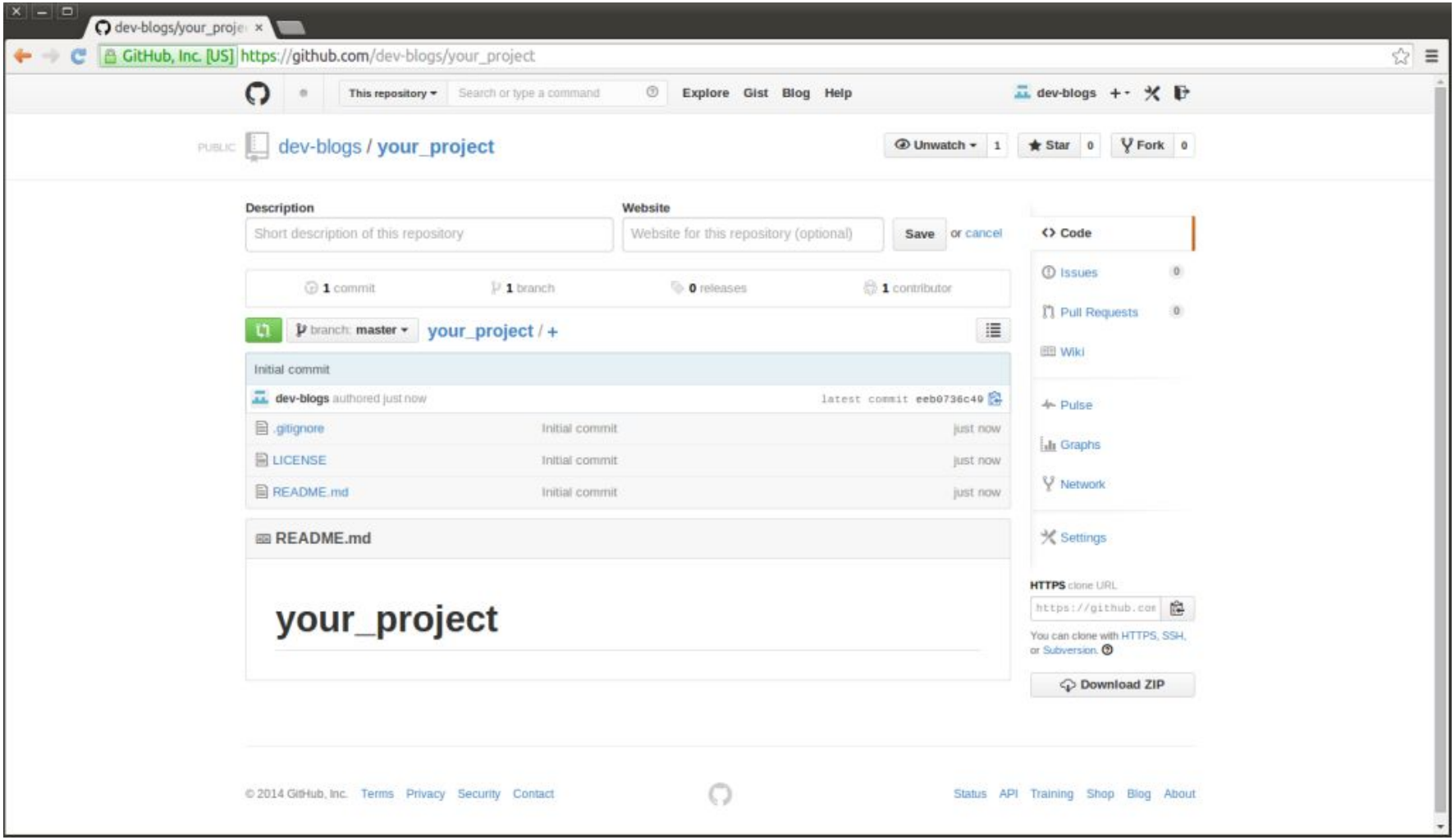
Создание репозитория
ШАГ 1



Создание репозитория
ШАГ 2



Создание репозитория
ШАГ 3



JAVA проект

Нужно за комментировать ***.jar** в файле **.gitignore**. Открываем на редактирование файл **.gitignore** и за комментируем, в нашем случае, седьмую строчку:

```
1  *.class
2
3  # Mobile Tools for Java (J2ME)
4  .mtj.tmp/
5
6  # Package Files #
7  #*.jar
8  *.war
9  *.ear
10 # virtual machine crash logs, see http://www.java.com/en/download/help/error_hotspot.xml
11 hs_err_pid*
```

Подготовка локального git репозитория

Два способа:
1) создать репозиторий с нуля с последующим переносом изменений в удаленный репозиторий;
2) сделать клон удаленного репозитория.

Создание локального репозитория с нуля командой **git init**

Создадим проект на локальной машине с таким именем:

```
mkdir your_project
```

перейдем в этот

```
cd your_project
```

Выполним команду **git init** которая инициирует локальный

```
git init
```

Дальше можно добавлять файлы в локальный репозиторий.

Сделать на локальной машине клон удалённого репозитория командой **git clone**:

```
git clone https://github.com/you_account/your_project
```

После этой команды у нас появится новый каталог в котором находится копия удаленного репозитория, а все файлы которые в нем находятся будут отслеживаться гитом. Тут очень важный момент именно **копия всего репозитория**, а не снимок текущего состояния удаленного репозитория. В отличие от обычного снимка удаленного репозитория, например как в **SVN** мы, будучи скопировав удаленный репозиторий, можем покопаться в его истории, посмотреть все его правки, кто и когда вносил изменения, какие у него ветки, то есть у нас на машине полноценный репозиторий который теперь не зависит от удаленного репозитория с которого был склонирован.

Подготовка локального файла

После того как появился локальный репозиторий, добавим в него джава класс. Перейдём в каталог, который отслеживается репозиторием и создадим какой-нибудь файл

TestGitHub.java

```
1 public class TestGitHub {
2     public static void main(String [] args) {
3         System.out.println("Test gitHub");
4     }
5 }
```

Помещение файла в репозиторий

После того как мы создали файл его надо подготовить для фиксации и зафиксировать в репозитории, то есть закомитить. **Подготовить для фиксации** это означает, что его надо **проиндексировать** командой **git add**:

```
git add *
```

Проиндексированный файл это еще не означает, что он закомичен, это означает, что он готов для коммита в репозиторий, а сам коммит выполняется командой **git commit**:

Помещение файла в репозиторий

```
git commit –m  "create project"

[master 412c945] create project
Committer: NAME <NAME@DOMAIN.(none)>
Your name and email address were configured automatically
based on your username and hostname.
Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"
git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit
with:

git commit --amend --reset-author
1 file changed, 5 insertions(+)
create mode 100644 src/TestGitHub.java
```

Перенос изменений на удаленный репозиторий

Локальный репозиторий готов, теперь осталось перенести его на удаленный. Переносится репозиторий командой **git push**, но прежде чем переносить мы должны выяснить со сколькими репозиториями мы работаем и выбрать из списка тот, в который мы хотим перенести наши изменения. Для того, чтобы увидеть все удаленные репозитории нужно выполнить команду **git remote -v**:

```
git remote -v

origin https://github.com/your_account/your_project (fetch)
origin https://github.com/your_account/your_project (push)
```

Перенос изменений на удаленный репозиторий

В данном случае мы работаем с одним удаленным репозиторием, которому присвоено короткое имя по умолчанию **origin**, который находится по адресу **https://github.com/your_account/your_project** как для фетча, так и для пуша.
Теперь можем переносить все изменения для репозитория **origin** командой
git push

```
git push origin master

Username for 'https://github.com': LOGIN
Password for 'https://LOGIN@github.com':
To https://github.com/LOGIN/your_project

eeb0736..412c945 master -> master
```

После этого github запросит имя юзера и пароль.
То что мы сейчас сделали мы запушили (выложили) наши локальные изменения на удаленный репозиторий у которого айдишник **origin** в ветку **master**.

Литература

- Pro Git https://git-scm.com/book/ru/v2
- Различные учебные пособия (tutorials), например, https://githowto.com/ru