



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 (для нетерпеливых)

```
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

The image shows a screenshot of the GitHub homepage in a web browser. The browser's address bar displays "GitHub, Inc. [US] https://github.com". The page features a navigation bar with "Search or type a command", "Explore", "Gist", "Blog", and "Help". A notification banner at the top states: "You don't have any verified emails. We recommend verifying at least one email. Email verification helps our support team help you in case you have any email issues or lose your password." Below this, there are tabs for "dev-blogs", "News Feed", "Pull Requests", and "Issues".

The main content area is dominated by the "GitHub Bootcamp" section, which consists of four numbered steps:

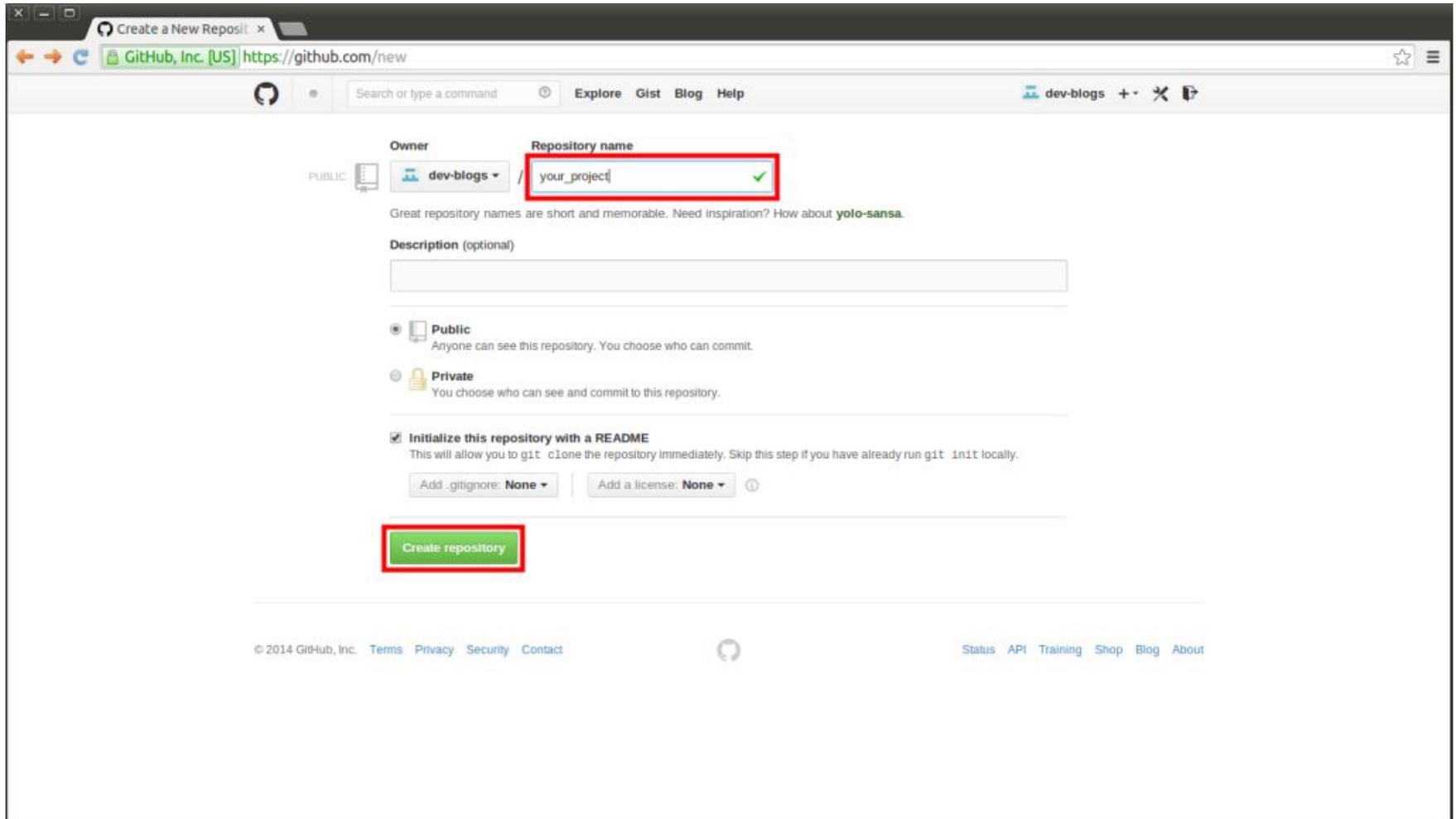
- 1 Set up Git**: A quick guide to help you get started with Git.
- 2 Create repositories**: Repositories are where you'll work and collaborate on projects.
- 3 Fork repositories**: Forking creates a new, unique project from an existing one.
- 4 Work together**: Send pull requests, follow friends, Star and watch projects.

Below the bootcamp section, there is a "Welcome to GitHub! What's next? (18 hours ago)" message with links to "Create a repository", "Tell us about yourself", "Browse interesting repositories", and "Follow @github on Twitter".

On the right side, the "Your repositories" section is visible. It includes a search bar "Find a repository...", tabs for "All repositories", "Public", "Private", "Sources", and "Forks", and a list of repositories: "simple-bad-app", "simple-spring-app", and "simple-spring-annotation-app". A red rectangular box highlights the "+ New repository" button in the top right corner of this section.

Создание репозитория

ШАГ 2



Browser: Create a New Repository | GitHub, Inc. [US] | https://github.com/new

Search or type a command | Explore | Gist | Blog | Help | dev-blogs

Owner: dev-blogs

Repository name: your_project ✓

Great repository names are short and memorable. Need inspiration? How about [yolo-sansa](#).

Description (optional)

Public
Anyone can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with a README
This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None** | Add a license: **None**

Create repository

© 2014 GitHub, Inc. | Terms | Privacy | Security | Contact | Status | API | Training | Shop | Blog | About

Создание репозитория

ШАГ 3

The screenshot shows the GitHub interface for a new repository named 'dev-blogs/your_project'. The repository is public and has 1 commit, 1 branch (master), 0 releases, and 1 contributor. The 'Code' section is expanded, showing the initial commit details. The commit message is 'Initial commit' and it was authored by 'dev-blogs' just now. The commit hash is 'eeb0736c49'. The files included in the commit are .gitignore, LICENSE, and README.md, all created at the time of the initial commit. The README.md content is 'your_project'. The right sidebar shows options for Code, Issues, Pull Requests, Wiki, Pulse, Graphs, Network, and Settings. The footer contains copyright information for GitHub, Inc. and links to Terms, Privacy, Security, and Contact, as well as Status, API, Training, Shop, Blog, and About.

dev-blogs/your_project

1 commit 1 branch 0 releases 1 contributor

branch: master your_project / +

Initial commit

dev-blogs authored just now latest commit eeb0736c49

.gitignore	Initial commit	just now
LICENSE	Initial commit	just now
README.md	Initial commit	just now

README.md

```
your_project
```

© 2014 GitHub, Inc. Terms Privacy Security Contact

Status API Training Shop Blog About

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 иницирует локальный репозиторий:

```
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 клонирован.

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 закоммитить. **Подготовить для фиксации** это означает, что его надо **проиндексировать** командой **git add**:

```
git add *
```

Проиндексированный файл это еще не означает, что он закоммичен, 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**.