

Git и его использование

Grigorii Hait

23 апреля 2019 г.

Содержание

О Git

Основы Git

- ▶ Предположим, вы работаете над проектом и в один момент обнаружили, что удалили что-то важное. Что будете делать?

- ▶ Предположим, вы работаете над проектом и в один момент обнаружили, что удалили что-то важное. Что будете делать?
- ▶ А теперь предположим, что вам нужно работать с кем-то в паре (тройке и т. д.). Как будете организовывать обмен работой?

- ▶ Предположим, вы работаете над проектом и в один момент обнуржили, что удалили что-то важное. Что будете делать?
- ▶ А теперь предположим, что вам нужно работать с кем-то в паре (тройке и т. д.). Как будете организовывать обмен работой?
- ▶ Вывод: нужно опеспечить контроль над версиями

Системы контроля версий

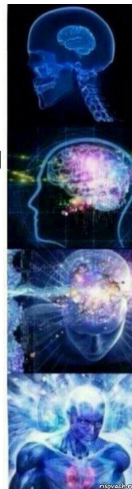
- ▶ Контроль версий – управление изменениями, произведенными при редактировании каких-либо документов
- ▶ Система управления версиями позволяет:
 - ▶ хранить несколько версий одного и того же документа
 - ▶ при необходимости возвращаться к более ранним версиям
 - ▶ определять, кто и когда сделал то или иное изменение

**ИСПОЛЬЗОВАТЬ
GIT**

**ДЕЛАТЬ СКРИНШОТЫ
КОДА И КИДАТЬ ЕГО
ДРУЗЬЯМ В VK**

**ФОТОГРАФИРОВАТЬ
КОД НА ТЕЛЕФОН И
РАСПЕЧАТЫВАТЬ НА
ПРИНТЕРЕ**

**ЗАПОМИНАТЬ
КОД**



Что такое Git

- ▶ Git – распределенная система управления версиями.
- ▶ Git хранит наборы изменений между разными версиями проекта, называемые фиксациями
- ▶ Каждый коммит указывает на предшествующий ему (или предшествующие), таким образом образуется дерево коммитов



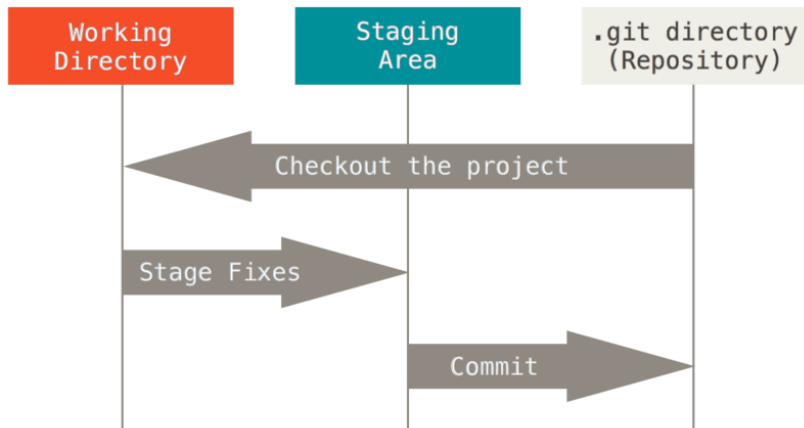
Первоначальная настройка

- ▶ Git можно бесплатно скачать с официального сайта
- ▶ Во время установки будет настроена специальная командная строка для работы с Git
- ▶ Git можно также использовать при помощи различных GUI. Самые известные: SourceTree, GitKraken
- ▶ Перед использованием нужно указать имя и email пользователя Git

Создание репозитория

- ▶ Для использования Git необходимо в первую очередь создать репозиторий или клонировать другой существующий репозиторий
- ▶ Создание происходит командой `git init`
- ▶ Клонирование выполняется при помощи команды `git clone`
- ▶ В результате в текущей папке должна появиться папка `.git`, содержащая все метаданные репозитория
- ▶ Результат можно проверить командой `git status`

Структура Git



Структура Git

Репозиторий Git можно разделить на три зоны:

- ▶ Git-директория (`.git`) — это то место, где Git хранит метаданные и базу объектов вашего проекта (папка `.git/`)
- ▶ Рабочая директория (*Working directory*) – это директория, в которой располагаются все файлы текущей версии проекта. Тут происходит вся основная работа
- ▶ Область подготовленных файлов (*Staging area*) — это набор всех изменений, которые попадут в новый коммит. Эту область ещё называют “индекс”.

Базовая работа с Git

- ▶ Вы изменяете файлы в вашей рабочей директории.
- ▶ Вы выборочно добавляете в индекс только те изменения, которые должны попасть в следующий коммит.
- ▶ Когда вы делаете коммит, используются файлы из индекса как есть, и этот снимок сохраняется в вашу Git-директорию.

Изменение рабочей папки

- ▶ Разрешается производить любые операции с файлами: добавлять новые, удалять старые, редактировать существующие
- ▶ Текущее состояние рабочей папки можно проверить командой `git status` – она должна показать измененные файлы.
- ▶ Более подробный список изменений можно получить, используя команду `git diff` – она покажет все изменения между рабочей папкой и индексом в каждом файле

```
PS G:\projects\git-presentation> git status
On branch master
Your branch is ahead of 'origin/master' by 3 commits.
(use "git push" to publish your local commits)

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:   git-basics/root.tex
        modified:   presentation.pdf

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        git-basics/basic-workflow-edit.tex

no changes added to commit (use "git add" and/or "git commit -a")
```

Рис. 1: Пример git status

```
diff --git a/git-basics/root.tex b/git-basics/root.tex
index a585679..35d9452 100644
--- a/git-basics/root.tex
+++ b/git-basics/root.tex
@@ -2,4 +2,5 @@
 \input{git-basics/creation.tex}
 \input{git-basics/structure.tex}
 \input{git-basics/structure-text.tex}
-\input{git-basics/basic-workflow.tex}
+ \input{git-basics/basic-workflow.tex}
+ \input{git-basics/basic-workflow-edit.tex}
 \ No newline at end of file
diff --git a/presentation.pdf b/presentation.pdf
index 39b67c8..40c11a7 100644
--- a/presentation.pdf
+++ b/presentation.pdf
@@ -52,4 +52,18 @@
 Git и его использование
```

Рис. 2: Пример git diff

Базовая работа с Git

Добавление файлов в индекс

- ▶ Для добавления файла в индекс используется команда `git add`
- ▶ Файлы можно добавлять выборочно, тогда необходимо указать пути к файлам с командой `git add`
- ▶ Можно также добавить все текущие изменения: для этого используется запись `git add .` или `git add -u`
- ▶ После изменения можно проверить состояние командой `git status` – она должна показать все изменения, добавленные в индекс. Разницу между индексом и последним коммитом можно получить, выполнив `git diff --staged`
- ▶ Если файл, изменения которого уже в индексе, опять изменить, новые изменения надо добавлять отдельно

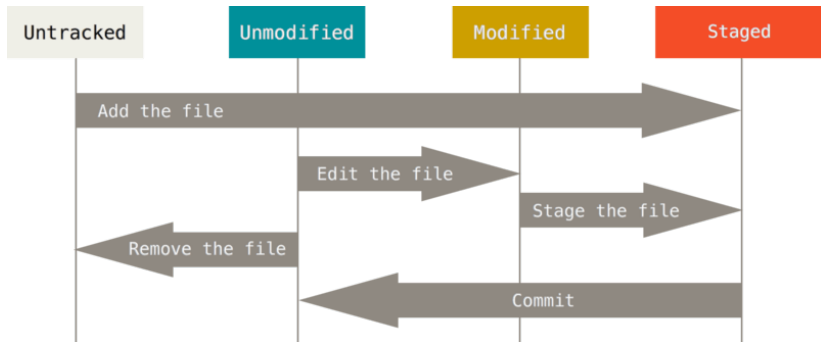
Базовая работа с Git

Создание коммита

- ▶ Коммит создается командой `git commit`. В коммит добавляются все изменения, находящиеся в индексе
- ▶ Коммит невозможно (почти) создать, не указав сообщение – краткое его описание. После вызова команды `git commit` откроется окно редактора, в котором нужно будет указать сообщение
- ▶ Сообщение можно задать при вызове команды `git commit`, указав флаг `-m`, например `git commit -m "Message"`
- ▶ Указав флаг `-a` можно произвести добавление всех изменений, кроме добавления новых файлов, в индекс и затем произвести коммит

Базовая работа с Git

Общая схема



Базовая работа с Git

Журнал коммитов

- ▶ Все созданные коммиты можно просмотреть в журнале коммитов (git log)
- ▶ По умолчанию, будут выведены коммиты, начиная с последнего.
- ▶ Без дополнительных настроек, будет выведена краткая информация о коммите (автор, время создания)
- ▶ При помощи флагов можно так же сортировать и фильтровать коммиты Например, git log --pretty=oneline выведет каждый коммит в отдельной строке

```
commit 18c3eafb33789abfd6898753bae25710bc0721 (HEAD -> master, origin/master)
Author: Grigorii Hail <grffn.94@gmail.com>
Date: Sat Apr 13 02:35:14 2019 +0300

    Add basic workflow scheme

commit af672e7866ac0d70ee9f81d206ca3cf78aba
Author: Grigorii Hail <grffn.94@gmail.com>
Date: Sat Apr 13 02:32:33 2019 +0300

    Add basic workflow commit page

commit 1d787139b08f0893a1d96f084263002bf2558b
Author: Grigorii Hail <grffn.94@gmail.com>
Date: Sat Apr 13 02:21:30 2019 +0300

    Add remark about diff-staged
```

Рис. 3: Пример git log

Историю можно менять

- ▶ Последний коммит можно изменить командой `git commit --amend`. Все содержимое индекса будет добавлено к последнему коммиту

