GeekBrains

Git. Базовый курс

Урок 3

Локальная работа с Git-репозиторием

Консольная команда git

Чтобы убедиться, что git установлен на вашем компьютере, в консоли следует набрать команду **git** и передать ей параметр **--version**.

```
$ git --version
```

В ответ команда сообщает текущую версию Git. Эта команда очень характерна, именно так будут выглядеть почти все команды при работе с системой контроля версий. Мы вводим **git**, далее следует команда или параметр, нажимаем клавишу <Enter>. В ответ утилита выдает отчет или изменяет внутреннее состояние проекта.

Настройка git

Перед началом работы необходимо сообщить **git**, кто вы и как вас представлять другим участникам распределенной системы контроля версий. Пока вы не подпишетесь, система не даст вам регистрировать снимки проекта, коммиты. Все ваши изменения должны быть подписаны вашим именем и электронным адресом, чтобы другие участники проекта знали, чьи это правки и как с вами связаться. Сделать это нужно один раз, как правило, сразу после установки git, если вы переустановите систему, процедуру потребуется повторить.

Чтобы задать ваши имя и электронный адрес, следует воспользоваться командой **git config**:

```
$ git config --global user.name "Igor Simdyanov"
$ git config --global user.email igorsimdyanov@gmail.com
```

Убедиться в том, что настройки успешно установлены, можно, запросив их список при помощи команды **git config --list**.

```
$ git config --list
user.name=Igor Simdyanov
user.email=igorsimdyanov@gmail.com
```

Она выдает множество настроек, в том числе и только что установленные значения, которые сохраняются в одном из многочисленных конфигурационных файлов git. Определить местоположение конфигурационного файла можно, если добавить к команде параметр **--show-origin**:

```
$ git config --list --show-origin
file:/Users/igorsimdyanov/.gitconfig    user.name=Igor Simdyanov
file:/Users/igorsimdyanov/.gitconfig    user.email=igorsimdyanov@gmail.com
...
```

Конфигурационный файл .**gitconfig** — это обычный текстовый файл. При желании его можно открыть в любом текстовом редакторе и отредактировать вручную.

Создание git-проекта

Git следит за состоянием не всей файловой системы, а выбранной папки на компьютере. Такую папку необходимо пометить как Git-проект.

Существует два способа получить в свое распоряжение git-проект: воспользоваться командой **git init**, чтобы пометить текущую папку как новый git-проект, или загрузить уже существующий проект с git-репозитория при помощи команды git clone.

Давайте создадим папку test и отправимся в консоль, чтобы инициализировать git-проект. Сделать это можно в редакторе, проводнике или UNIX-подобной операционной системе при помощи команды **mkdir**:

```
$ mkdir test
```

Для того, чтобы перейти внутрь папки, можно воспользоваться командой **cd**:

```
$ cd test
```

Чтобы пометить текущую папку как git-проект, выполняем команду **git init**:

```
$ git init
```

Команда создает в текущей папке подпапку **.git**, именно в ней будет содержаться вся необходимая информация и сохраняться текущее состояние git-проекта.

```
.git
  hooks
  info
  objects
  refs
  config
  description
  HEAD
```

Создание первого коммита

На этом этапе файлы проекта пока не находятся под версионированием системы контроля версий. Давайте добавим, какой-либо файл, например создадим новый файл **hello.txt**. Сделать этого можно либо в текстовом редакторе, либо, если мы находимся в UNIX-подобной операционной системе, при помощи команды **touch**:

```
$ touch hello.txt
```

Убедиться, что файл успешно создан, можно при помощи команды **ls**:

```
$ ls -la
total 0
drwxr-xr-x  3 igorsimdyanov  staff  96  8 с е н 21:55 .
drwxrwxrwx  3 igorsimdyanov  staff  96  8 с е н 21:55 ..
-rw-r--r--  1 igorsimdyanov  staff   0  8 с е н 21:55 hello.txt
```

Чтобы добавить файл под контроль git, необходимо проиндексировать его при помощи команды **git add**:

```
$ git add .
```

Здесь точка означает все новые файлы в папке, вместо этого можно указать избранные файлы или каталоги:

```
$ git add hello.txt
```

Такая команда добавит в индекс только файл hello.txt.

После того, как мы сообщили Git, какие файлы проекта мы хотим отслеживать, их можно зафиксировать. Для этого создается снимок состояния, который называется коммитом.

Сделать это можно при помощи команды git commit:

```
$ git commit -m 'П е р в ы й  к о м м и т'
[master (root-commit) c24ec38] П е р в ы й  к о м м и т
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 hello.txt
```

В параметре -m мы задаем комментарий к коммиту.

Пока порядок действий может показаться непонятным, не очень ясно, как наши действия отражаются на состоянии системы. Однако мы обязательно научимся контролировать все состояния в течение курса.

Редактор по умолчанию для комментариев к коммиту

Давайте что-нибудь запишем в файл, например, фразу "Hello, world!".

```
echo 'Hello, world!' > hello.txt
```

В тексте методического пособия мы это делаем при помощи перенаправления стандартного вывода в файл. На самом деле не важно, как эта строка появится в файле, для этого можно воспользоваться текстовым редактором или любым привычным для вас способом.

Снова добавляем файл в индекс:

```
$ git add hello.txt
```

Далее фиксируем состояние командой **git commit**:

```
$ git commit
```

Здесь намеренно не указывается комментарий. После нажатия клавиши <Enter> команда, вместо того, чтобы завершиться, открывает редактор по умолчанию, как правило, консольный vim.

```

# 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:
# modified: hello.txt
#
```

Чтобы оставить комментарий, необходимо перейти в режим редактирования. Для этого на клавиатуре последовательно нажимаются клавиши <Esc> и <i>. Клавиши нажимаются в английской раскладке, если вы находитесь в русской раскладке, без специальной настройки vim комбинация не сработает.

В режиме редактирования можно набирать комментарий:

```
Comment
```

```
# 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:
# modified: hello.txt
#
```

Чтобы прекратить набор текста и иметь возможность управлять редактором, необходимо перевести редактор в командный режим. Для этого необходимо на клавиатуре нажать клавишу <Esc>, а затем двоеточие <:>.

Внизу экрана появится приглашение для ввода управляющей последовательности. Чтобы выйти из редактора, следует набрать последовательность **wq** (сокращение для write quit — записать и выйти).

После нажатия на клавишу <Enter> редактор закрывается, коммит добавляется с введенным ранее комментарием 'Comment'.

Консольный редактор vim сложен, если он вам не нравится, работать в нем не обязательно и на уровне настроек git можно задать какой-то другой редактор.

Взаимодействие с редактором vim может быть довольно утомительно, так как для комфортной работы потребуется изучить его режимы и многочисленные клавиатурные сокращения. Если работа с редактором vim вас пугает, можете либо каждый раз добавлять параметр **-m** и задавать комментарий в команде **git commit**, либо настроить другой редактор по умолчанию. Например, в качестве такого редактора можно задать nano, который гораздо проще, чем vim.

Чтобы была возможность добавить еще один коммит, потребуется в третий раз изменить файл hello.txt. Далее при помощи команды **git config** изменяем редактор по умолчанию:

```
$ git config --global core.editor nano
$ git add .
$ git commit
```

В открывшемся редакторе nano можно сразу вводить комментарий. Чтобы записать файл, придется воспользоваться комбинацией <Ctrl>+<O>. После этого можно выйти при помощи комбинации клавиш <Ctrl>+<X>.

Можно использовать любой другой текстовый редактор, даже графический, главное, чтобы он запускался из консоли и мог принимать название файла в качестве параметра.

Например, в операционной системе Windows можно воспользоваться блокнотом. Далее команды выполняются в предположении, что в Windows используется Git for Windows.

Чтобы посмотреть текущие настройки, можно воспользоваться командой **git config**:

```
$ git config --list
```

По умолчанию, при создании коммита, если не указать параметр **-m**, в Windows тоже откроется консольный редактор vim. Давайте заменим его на блокнот, запустить который из командной строки можно при помощи команды **notepad**:

```
notepad
```

Чтобы задействовать его в качестве редактора по умолчанию, зададим его командой **git config**:

```
$ git config core.editor notepad
$ git config --list
```

Любым удобным способом создаем файл hello.txt и добавляем его в индекс при помощи команды git add:

```
git add hello.txt
```

Выполняем команду git commit:

```
git commit
```

В результате запускается блокнот, в котором можно набрать комментарий, например, «Новый файл». Сохранить результат можно комбинацией клавиш <Ctrl> + <S>. После закрытия блокнота коммит добавляется.

Как и в UNIX-операционных системах, добавления комментария во внешнем редакторе можно избежать, если в команде **git commit** использовать параметр **-m**:

```
git commit -m "Новый файл"
```

Клонирование проекта

Если проект под управлением Git уже существует в каком-то сетевом репозитории, его можно загрузить при помощи команды git clone. Такой процесс называется клонированием.

Давайте воспользуемся таким готовым git-репозиторием на git-хостинге GitHub: `https://github.com/web-standards-ru/dictionary.git`. Проект представляет собой словарик по переводу терминов.

С GitHub мы еще подробно познакомимся, когда будем говорить об удаленных репозиториях, сейчас он нас интересует только как источник какого-нибудь готового проекта.

Получить ссылку для клонирования можно, воспользовавшись кнопкой Clone or Download. Копируем ссылку для клонирования в буфер обмена, она нам потребуется в качестве источника в команде **git clone**.

В консоли выполняем команду **git clone**, передавая ей в качестве аргумента ссылку на клонируемый репозиторий:

```
$ git clone https://github.com/web-standards-ru/dictionary.git
```

Команда выполняется гораздо дольше, чем все предыдущие, так как содержимое репозитория, всю его историю необходимо загрузить по сети. В результате в каталоге, где была выполнена команда, будет создан git-репозиторий словаря.

Если необходимо переименовать папку, новое название можно указать сразу после адреса репозитория в команде git clone:

```
$ git clone https://github.com/web-standards-ru/dictionary.git dict
```

В результате создается копия репозитория в папке dict.

Графические клиенты

Для работы с системой контроля версий Git предусмотрено множество графических клиентов. Часть из них доступна прямо из коробки, например, gitk и git-gui. Другие, такие как GitHub, требуют отдельной установки.

Многие интеграционные среды — Эклипс, линейка редакторов JetBrains — также поддерживают интеграцию с git.

Графический клиент gitk позволяет просматривать историю. Запустить его можно командой gitk в git-проекте:

```
$ gitk
```

В окне gitk можно видеть историю коммитов: время их создания, автора, а также изменения, которые были добавлены в этих коммитах.

Клиент git-gui предназначен для формирования коммитов:

```
$ echo 'Hello, Git!' > hello.txt
```

После запуска клиента командой **git gui** можно видеть измененные файлы, понимать, что в них поменялось, добавлять комментарии.

Нажимаем кнопку Stage Changed, чтобы проиндексировать файл. Это аналогично вызову команды **git add**. Далее для создания коммита можем нажать кнопку commit.

Используемые источники

Для подготовки методического пособия мы использовали эти ресурсы:

- Официальная документация Git: **https://git-scm.com/book/ru/v2.**