Лабораторная работа 1
Знакомство с GIT

Общие сведения

Система управления версиями (от англ. Version Control System, VCS или Revision Control System) – программное обеспечение для облегчения работы с изменяющейся информацией. 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, и многое другое.

Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы. Однако они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов.

Git – распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года.

Ядро Git представляет собой набор утилит командной строки с параметрами. Все настройки хранятся в текстовых файлах конфигурации. Такая реализация делает Git легко портируемым на любую платформу и даёт возможность легко интегрировать Git в другие системы (в частности, создавать графические git – клиенты с любым желаемым интерфейсом).

Репозиторий Git представляет собой каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранящие операции, выполняемые над репозиторием, индекс, описывающий расположение файлов, и хранилище, содержащее собственно файлы. Структура хранилища файлов не отражает реальную структуру хранящегося в репозитории файлового дерева, она ориентирована на повышение скорости выполнения операций с репозиторием. Когда ядро обрабатывает команду изменения (неважно, при локальных изменениях или при получении патча от другого узла), оно создаёт в хранилище новые файлы, соответствующие новым состояниям изменённых файлов. Существенно, что никакие операции не изменяют содержимого уже существующих в хранилище файлов.

По умолчанию репозиторий хранится в подкаталоге с названием `.git` в корневом каталоге рабочей копии дерева файлов, хранящегося в репозитории. Любое файловое дерево в системе можно превратить в репозиторий git, отдав команду создания репозитория из корневого каталога этого дерева (или указав корневой каталог в параметрах программы). Репозиторий может быть импортирован с другого узла, доступного по сети. При импорте нового репозитория автоматически создаётся рабочая копия, соответствующая последнему зафиксированному состоянию импортируемого репозитория (то есть не копируются изменения в рабочей копии исходного узла, для которых на том узле не была выполнена команда `commit`).

Установка git

Установка в Linux

Если вы хотите установить Git под Linux как бинарный пакет, это можно сделать, используя обычный менеджер пакетов вашего дистрибутива. Если у вас Fedora (или другой похожий дистрибутив, такой как RHEL, CentOS, РедОС), можно воспользоваться `dnf`:

```
$ sudo dnf install git-all
```

Если же у вас дистрибутив, основанный на Debian, например, Ubuntu или Астра Линукс, попробуйте `apt`:

```
$ sudo apt install git
```

Чтобы воспользоваться дополнительными возможностями, посмотрите инструкцию по установке для нескольких различных разновидностей Unix на сайте Git http://git-scm.com/download/linux.

Установка на Mac

Существует несколько способов установки Git на Mac. Самый простой — установить Xcode Command Line Tools. В версии Mavericks (10.9) и выше вы можете добиться этого просто первый раз выполнив git в терминале:

```
$ git --version
```

Если Git не установлен, вам будет предложено его установить.

Установка в Windows

Для установки Git в Windows также имеется несколько способов. Официальная сборка доступна для скачивания на официальном сайте Git. Просто перейдите на страницу http://git-scm.com/download/win, и загрузка запустится автоматически. Обратите внимание, что это отдельный проект, называемый Git для Windows; для получения дополнительной информации о нём перейдите на https://gitforwindows.org.

Другой простой способ установки Git — установить GitHub для Windows. Его установщик включает в себя утилиты командной строки и GUI Git. Он также корректно работает с Powershell, обеспечивает надёжное сохранение учётных данных и правильные настройки CRLF. Вы познакомитесь с этими вещами подробнее несколько позже, здесь же отметим, что они будут вам необходимы. Вы можете загрузить GitHub для Windows с вебсайта GitHub Desktop.

Первоначальная настройка Git

Теперь, когда Git установлен в вашей системе, самое время настроить среду для работы с Git под себя. Это нужно сделать только один раз — при обновлении версии Git настройки сохранятся. Но, при необходимости, вы можете поменять их в любой момент, выполнив те же команды снова.

В состав Git входит утилита `git config`, которая позволяет просматривать и настраивать параметры, контролирующие все аспекты работы Git, а также его внешний вид. Эти параметры могут быть сохранены в трёх местах:

Файл `/etc/gitconfig` содержит значения, общие для всех пользователей системы и для всех их репозиториев. Если при запуске `git config` указать параметр `--system`, то параметры будут читаться и сохраняться именно в этот файл.

Файл `~/.gitconfig` или `~/.config/git/config` хранит настройки конкретного пользователя. Этот файл используется при указании параметра `--global`.

Файл `config` в каталоге Git (т.е. `.git/config`) репозитория, который вы используете в данный момент, хранит настройки конкретного репозитория.

Настройки на каждом следующем уровне подменяют настройки из предыдущих уровней, то есть значения в `.git/config` перекрывают соответствующие значения в `/etc/gitconfig`.

Чтобы посмотреть все установленные настройки и узнать где именно они заданы, используйте команду:

```
$ git config --list --show-origin
```

Имя пользователя

Первое, что вам следует сделать после установки Git — указать ваше имя и адрес электронной почты. Это важно, потому что каждый коммит в Git содержит эту информацию, и она включена в коммиты, передаваемые вами, и не может быть далее изменена:

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

Опять же, если указана опция `--global`, то эти настройки достаточно сделать только один раз, поскольку в этом случае Git будет использовать эти данные для всего, что вы делаете в этой системе. Если для каких-то отдельных проектов вы хотите указать другое имя или электронную почту, можно выполнить эту же команду без параметра `--global` в каталоге с нужным проектом.

Многие GUI-инструменты предлагают сделать это при первом запуске.

Выбор редактора

Теперь, когда вы указали своё имя, самое время выбрать текстовый редактор, который будет использоваться, если будет нужно набрать сообщение в Git. По умолчанию Git использует стандартный редактор вашей системы, которым обычно является Vim. Если вы хотите использовать другой текстовый редактор, например, `nano`, можно проделать следующее:

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

В системе Windows следует указывать полный путь к исполняемому файлу при установке другого текстового редактора по умолчанию. Пути могут отличаться в зависимости от того, как работает инсталлятор.

Настройка ветки по умолчанию

Когда вы инициализируете репозиторий командой `git init`, Git создаёт ветку с именем master по умолчанию. Начиная с версии 2.28, вы можете задать другое имя для создания ветки по умолчанию.

Например, чтобы установить имя `main` для вашей ветки по умолчанию, выполните следующую команду:

```
$ git config --global init.defaultBranch main
```

Проверка настроек

Если вы хотите проверить используемую конфигурацию, можете использовать команду `git config --list`, чтобы показать все настройки, которые Git найдёт:

```
$ git config --list
user.name=John Doe
user.email=johndoe@example.com
color.status=auto
color.branch=auto
color.interactive=auto
color.diff=auto
...
```

Некоторые ключи (названия) настроек могут отображаться несколько раз, потому что Git читает настройки из разных файлов (например, из `/etc/gitconfig` и `~/.gitconfig`). В таком случае Git использует последнее значение для каждого ключа.

Также вы можете проверить значение конкретного ключа, выполнив `git config <key>`:

```
$ git config user.name
```

Как получить помощь?

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

```
$ git help <команда>
$ git <команда> --help
$ man git-<команда>
```

Например, так можно открыть руководство по команде `git config`:

```
$ git help config
```

Эти команды хороши тем, что ими можно пользоваться всегда, даже без подключения к сети.

Так же, если вам нужно посмотреть только список опций и вы не хотите читать полную документацию по команде, вы можете использовать опцию `-h` для вывода краткой инструкции по использованию:

```
$ git add -h
```

Создание Git-репозитория

Для создания Git-репозитория вы можете использовать два основных подхода. Во-первых, импорт в Git уже существующего проекта или директории. Во-вторых, клонирование существующего репозитория с другого сервера.

В обоих случаях вы получите готовый к работе Git репозиторий на вашем компьютере.

Создание репозитория в директории

Создайте директорию с бушующим проектом, который будет находится под версионным контролем, например:

```
$ mkdir my_project
```

Перейдите в эту директорию:

```
$ cd my_project
```

а затем выполните команду:

```
$ git init
```

Эта команда создаёт в текущей директории новую поддиректорию с именем `.git`, содержащую все необходимые файлы репозитория — структуру Git-репозитория. На этом этапе ваш проект ещё не находится под версионным контролем.

Проверить статус репозитория можно при помощи команды:

```
$ git status
```

Также можно использовать сокращенный вывод статуса при помощи команды:

```
$ git status -s
```

Добавьте в данную папку какой-нибудь файл будущего проекта, например `readme.md`:

```
$ touch readme.md
```

Теперь добавим под версионный контроль созданный файл, для это нужно добавить его в индекс и осуществить первый коммит изменений. Добиться этого вы сможете запустив команду `git add`, указав индексируемые файлы, а затем выполнив `git commit`, с указанием сопроводительного сообщения:

```
$ git add readme.md
$ git commit -m 'initial commit'
```

Теперь у вас есть Git-репозиторий с отслеживаемыми файлами и начальным коммитом.

Попробуйте изменить содержимое файла `readme.md` любым текстовым редактором, например `nano`, и проверьте вывод команды `git status`:

```
$ nano readme.md
$ git status
```

Чтобы новые изменения попали в репозиторий потребуется снова выполнить команду `git commit`.

Игнорирование файлов

Зачастую, у вас имеется группа файлов, которые вы не только не хотите автоматически добавлять в репозиторий, но и видеть в списках неотслеживаемых. К таким файлам обычно относятся автоматически генерируемые файлы (различные логи, результаты сборки программ и т.п.) и файлы с чувствительными данными (например файл с переменными окружения `.env`, где могут быть логины/пароли и ключи доступа). В таком случае, вы можете создать файл `.gitignore`. с перечислением шаблонов соответствующих таким файлам. Вот пример файла `.gitignore`:

```
# Исключить все файлы с расширением .a
*.a

# Но отслеживать файл lib.a даже если он подпадает под исключение выше
!lib.a

# Исключить файл TODO в корневой директории, но не файл в subdir/TODO
/TODO

# Игнорировать все файлы в директории build/
build/

# Игнорировать файл doc/notes.txt, но не файл doc/server/arch.txt
doc/*.txt

# Игнорировать все .txt файлы в директории doc/
doc/**/*.txt
```

GitHub поддерживает довольно полный список примеров `.gitignore` файлов для множества проектов и языков https://github.com/github/gitignore это может стать отправной точкой для `.gitignore` в вашем проекте.

В простейшем случае репозиторий будет иметь один файл `.gitignore` в корневой директории, правила из которого будут рекурсивно применяться ко всем поддиректориям. Так же возможно использовать `.gitignore` файлы в поддиректориях. Правила из этих файлов будут применяться только к директории, в которой находятся. Например, репозиторий исходного кода ядра Linux содержит 206 файлов `.gitignore`.

Детали использования файлов `.gitignore` доступны в справке `man gitignore`.

Задание

1. Установить git (проверить можно командой `git --version`).
2. Произвести первоначальную настройку (имя пользователя и email).
3. Создать каталог и сделать в нём git-репозиторий.
4. Создать в нём любой файл и добавить его в индекс репозитория.
5. Сделать первый commit с любым сообщением.
6. Изменить содержимое файла и проверить статус.
7. Сделать второй commit с указанием проделанных изменений.
8. Создать файл `.gitignore`, указать в нём имя файла `.env`.
9. Создать файл `.env` с любым содержимым. Убедится что он не отслеживается системой контроля версий.