**ЛАБОРАТОРНАЯ РАБОТА № 1**

**Git**

*Цель*: изучение основ пользования системой Git, создание собственного репозитория и размещения в нем своего проекта.

**1.1 Основные теоретические положения**

Git — это набор консольных утилит, которые отслеживают и фиксируют изменения в файлах (чаще всего речь идет об исходном коде программ, но можно использовать его для любых файлов). С его помощью можно откатиться на более старую версию проекта, сравнивать, анализировать, сливать изменения и многое другое. Этот процесс называется контролем версий. Существуют различные системы для контроля версий: SVN, Mercurial, Perforce, CVS, Bitkeeper и другие.

Git является распределенным, то есть не зависит от одного центрального сервера, на котором хранятся файлы. Вместо этого он работает полностью локально, сохраняя данные в папках на жестком диске, которые называются репозиторием. Тем не менее, вы можете хранить копию репозитория онлайн, это сильно облегчает работу над одним проектом для нескольких людей. Для этого используются сайты вроде github, gitlab, bitbucket и т.д..

**1.1.1 Установка**

Linux — нужно просто открыть терминал и установить приложение при помощи пакетного менеджера вашего дистрибутива. Для Ubuntu команда будет выглядеть следующим образом:

sudo apt-get install git

Windows — рекомендуется git for windows, так как он содержит и клиент с графическим интерфейсом, и эмулятор bash.

OS X — проще всего воспользоваться homebrew. После его установки запустите в терминале:

brew install git

Если вы новичок, клиент с графическим интерфейсом(например GitHub Desktop и Sourcetree) будет полезен, но, тем не менее, знать команды очень важно.

Так можно использовать портативную версию git, таким образом он всегда будет с вами на флешке, куда бы вы не пошил.

**1.1.2 Настройка**

Итак, мы установили git, теперь нужно добавить немного настроек. Есть довольно много опций, с которыми можно играть, но мы настроим самые важные: наше имя пользователя и адрес электронной почты. Откройте терминал git и запустите команды:

git config --global user.name "My Name"

git config --global user.email myEmail@example.com

Теперь каждое наше действие будет отмечено именем и почтой. Таким образом, пользователи всегда будут в курсе, кто отвечает за какие изменения — это вносит порядок.

**1.1.3 Создание нового репозитория**

Git хранит свои файлы и историю прямо в папке проекта. Чтобы создать новый репозиторий, нам нужно открыть терминал, зайти в папку нашего проекта и выполнить команду init. Это включит приложение в этой конкретной папке и создаст скрытую директорию .git, где будет храниться история репозитория и настройки.

Создайте на диске D папку под названием git\_exercise. Для этого в окне терминала введите:

mkdir D:/git\_exerciese/

cd D:/git\_exerciese/

git init

Командная строка должна вернуть что-то вроде:

Initialized empty Git repository in D:/git\_exerciese/.git/

Это значит, что наш репозиторий был успешно создан, но пока что пуст. Теперь создайте текстовый файл под названием hello.txt и сохраните его в директории git\_exercise.

**1.1.4 Определение состояния**

Status — это еще одна важнейшая команда, которая показывает информацию о текущем состоянии репозитория: актуальна ли информация на нём, нет ли чего-то нового, что поменялось, и так далее. Запуск git status на нашем свежесозданном репозитории должен выдать:

$ git status

On branch master

No commits yet

Untracked files:

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

hello.txt

nothing added to commit but untracked files present (use "git add" to track)

Сообщение говорит о том, что файл hello.txt не отслеживаемый. Это значит, что файл новый и система еще не знает, нужно ли следить за изменениями в файле или его можно просто игнорировать. Для того чтобы начать отслеживать новый файл, нужно его специальным образом объявить.

**1.1.5 Подготовка файлов проекта**

В git есть концепция области подготовленных файлов. Можно представить ее как холст, на который наносят изменения, которые нужны в коммите. Сначала он пустой, но затем мы добавляем на него файлы (или части файлов, или даже одиночные строчки) командой add и, наконец, коммитим все нужное в репозиторий (создаем слепок нужного нам состояния) командой commit.

В нашем случае у нас только один файл, так что добавим его:

$ git add hello.txt

Если нам нужно добавить все, что находится в директории, мы можем использовать

$ git add -A

Проверим статус снова, на этот раз мы должны получить другой ответ:

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: hello.txt

Файл готов к коммиту. Сообщение о состоянии также говорит нам о том, какие изменения относительно файла были проведены в области подготовки — в данном случае это новый файл, но файлы могут быть модифицированы или удалены.

**1.1.6 Коммит (фиксация изменений)**

Коммит представляет собой состояние репозитория в определенный момент времени. Это похоже на снапшот, к которому мы можем вернуться и увидеть состояние объектов на определенный момент времени.

Чтобы зафиксировать изменения, нам нужно хотя бы одно изменение в области подготовки (мы только что создали его при помощи git add), после которого мы может коммитить:

$ git commit -m "Commit text"

Эта команда создаст новый коммит со всеми изменениями из области подготовки (добавление файла hello.txt). Ключ -m и сообщение «Commit text» — это созданное пользователем описание всех изменений, включенных в коммит. Считается хорошей практикой делать коммиты часто и всегда писать содержательные комментарии. Commit text может быть любым текстом, он служит для того, чтобы проще было ориентироваться в проекте.

**1.1.7 Удаленные репозитории**

Сейчас наш коммит является локальным — существует только в директории .git на нашей файловой системе. Несмотря на то, что сам по себе локальный репозиторий полезен, в большинстве случаев мы хотим поделиться нашей работой или доставить код на сервер, где он будет выполняться.

**1.1.7.1 Создание удаленного репозиторию**

Для подключения к удаленному репозиторию и отправке на него проектов для начала необходимо создать свой аккаунт. Вы можете воспользоваться любым доступным для вас репозиторием. В данной работе будет использован репозиторий Gitlab так как он имеет возможность создавать приватные репозитории бесплатно.

Для регистрации на Gitlab перейдите по ссылке <https://gitlab.com/users/sign_in#register-pane> и следуйте инструкции.

После регистрации создайте проект с именем **lab4.**

**1.1.7.2 Подключение к удаленному репозиторию**

Чтобы загрузить что-нибудь в удаленный репозиторий, сначала нужно к нему подключиться.

Чтобы связать наш локальный репозиторий с репозиторием на Gitlab, выполним следующую команду в терминале. Обратите внимание, что нужно обязательно изменить URI репозитория на свой.

$ git remote add origin https://gitlab.com/yourlogin/lab4.git

Проект может иметь несколько удаленных репозиториев одновременно. Чтобы их различать, мы дадим им разные имена. Обычно главный репозиторий называется origin.

**1.1.7.3 Отправка изменений на сервер**

Сейчас самое время переслать наш локальный коммит на сервер. Этот процесс происходит каждый раз, когда мы хотим обновить данные в удаленном репозитории.

Команда, предназначенная для этого — **push**. Она принимает два параметра: имя удаленного репозитория (мы назвали наш origin) и ветку, в которую необходимо внести изменения (master — это ветка по умолчанию для всех репозиториев).

$ git push origin master

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Writing objects: 100% (3/3), 210 bytes | 210.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://gitlab.com/yourlogin/lab4.git

\* [new branch] master -> master

В зависимости от сервиса, который вы используете, вам может потребоваться аутентифицироваться, чтобы изменения отправились. Если все сделано правильно, то когда вы посмотрите в удаленный репозиторий при помощи браузера, вы увидете файл hello.txt

**1.1.7.4 Клонирование репозитория**

Сейчас другие пользователи GitHub могут просматривать ваш репозиторий. Они могут скачать из него данные и получить полностью работоспособную копию вашего проекта при помощи команды clone.

$ git clone https://gitlab.com/yourlogin/lab4.git

Новый локальный репозиторий создается автоматически с GitHub в качестве удаленного репозитория.

**1.2 Задание**

Список заданий для выполнения лабораторной работы:

1. создать свой аккаунт на любом git репозитории;
2. установить или запустить портативную версию git и произвести стартовые настройки;
3. проинициализировать директорию для git у себя на компьютере;
4. поместить свой проект проинициализированной папке;
5. закоммитить все данные;
6. запулить свой проект на репозиторий, на котором вы зарегистрировались;
7. сформировать новую ветку;
8. внести изменения в новой ветке;
9. произвести новый коммит;
10. создать файл readme.md с описанием проекта - <http://webdesign.ru.net/article/pravila-oformleniya-fayla-readmemd-na-github.html>;
11. клонировать свой проект в другую папку.

**1.3 Отчет**

Отчет должен содержать в себе выполнение всех пунктов заданий и быть оформлен в следующем порядке:

* титульная страница, оформленная по нормам;
* 1-я страница должна состоять из:
  + тема;
  + цель;
  + задание лабораторной работы;
  + описание предметной области (если необходимо);
  + листинг;
* скриншоты работы программы. Скриншоты должны быть оформлены в соответствии с гостами, на любое изображение необходимо сформировать ссылку в тексте, так же изображение должно иметь подпись снизу по центру;
* вывод.