# Лабораторная работа №1

## Система управления версиями Git

## Сервис Github

### Системы управления версиями

Системы управления версиями (Version control system, VCS) – программное обеспечение, предназначенное для управления быстро меняющейся информацией. Системы управления версиями позволяют сохранять нужное количество предыдущих версий информации (обычно – текстового документа, в том числе с программным кодом, или набора таких документов – проекта), обращаться к этим версиям при необходимости, а также позволяют организовать редактирование одного и того же документа разными пользователями.

Существуют централизованные и распределенные системы управления версиями. В централизованных VCS функции по хранению версий документа и предоставлению доступа к ним берет на себя сервер. Для внесения изменений в какой-либо документ пользователю в первую очередь обычно требуется скачать на свой компьютер «рабочую копию» нужной ему версии документа или обновить уже имеющуюся версию до последней (update) . После внесения изменений пользователь закачивает новую версию документа на сервер (commit), причем предыдущая версия не удаляется, а также сохраняется на сервере.



Централизованная система управления версиями

При работе в многопользовательском режиме сервер также осуществляет отслеживание и управление конфликтами изменений, сделанных разными пользователями. Если несколько пользователей одновременно редактируют один и тот же файл, то изменения, сделанные одним пользователем, будут отменены тем пользователем, который загрузит свою версию того же документа на сервер после первого пользователя. Чтобы избежать конфликтов, может применяться блокировка документа, который один из пользователей скачал на свой компьютер с целью изменения. При этом остальные пользователи не имеют возможности внести в него изменения (то есть загрузить на сервер свою новую версию этого документа), пока первый пользователь не «разблокирует» документ после внесения в него изменений.

Так как механизм блокировки приводит к замедлению совместной работы над документом, часто (например, при слиянии ветвей, см. ниже) вместо недопущения конфликтов применяется механизм их автоматического или ручного разрешения. Автоматически могут быть разрешены конфликты, при которых пользователи внесли изменения в разные части документа. Если же изменения, сделанные разными пользователями, пересекаются, или в случае, если формат документа отличен от текстового (картинка, бинарный файл), система управления версиями предложит пользователям разрешить его конфликт вручную.

Централизованные VCS, как и любые централизованные системы имеют риски потери некоторой части информации при выходе из строя центрального сервера. Для устранения этих рисков могут использоваться децентрализованные системы управления версиями. В децентрализованных VCS пользователи скачивают на свои рабочие компьютеры не только нужные им версии нужных им файлов, а полностью весь репозиторий (хранилище файлов). В этом случае при выходе из строя сервера VCS вся информация может быть восстановлена из рабочих компьютеров пользователей.



Децентрализованная система управления версиями

Помимо этого, системы управления версиями хранят информацию о том, кто, когда и какие изменения выполнил, позволяет сравнивать две версии документа, а также реализует функцию ветвления. Создание «ветви» (branch или fork) позволяет создать две копии одного и того же набора документов, развивать их параллельно и при необходимости снова слить в один проект (merge). Ветвление позволяет, например, поддерживать несколько версий одного и того же продукта и зачастую выполняется перед началом работы над достаточно объемным обновлением для продукта.

### Система управления версиями Git

Git – распределенная кроссплатформенная система управления версиями, созданная в 2005 году для управления разработкой ядра Linux. Благодаря своему удобству и скорости работы является в настоящий момент одной из самых популярных VCS. C Git можно работать как посредством командной строки, так и при помощи любого из множества графических интерфейсов.

Установку Git на Linux-системы можно произвести стандартной командой, например, для Ubuntu:

sudo apt-get install git

Для Windows можно использовать Git for Windows (<https://git-scm.com/download/win>) или любой другой клиент на ваш выбор.

Если вы решили освоить работу с git через командную строку, то ниже приведены некоторые команды для работы с Git из командной строки. Создание репозитория Git в той директории, в которой выполняется команда:

git init

Создание локальной рабочей копии локального или удаленного репозитория:

git clone [url]

После создания рабочей копии можно добавлять и изменять в ней нужные файлы.

Файлы в Git могут находиться в трех различных состояниях:

1. Committed или зафиксированном – файл сохранен в локальном репозитории. Репозиторий Git хранит в сжатом виде все версии файлов и метаданные, это основное хранилище проекта.
2. Modified – файл был изменен в рабочей копии проекта (куда данные выгружаются из репозитория для их изменения), но новая версия еще не была загружена в локальный репозиторий.
3. Staged или подготовленном (другое наименование – добавлен в индекс, индексирован) – новая версия файла еще не добавлена в локальный репозиторий, но проиндексирована, чтобы быть добавленной при следующем выполнении команды commit.



Жизненный цикл состояний файлов в Git

Таким образом, сохранение новой версии файла (а также добавление нового файла) в локальный репозиторий выполняется в два шага – сначала производится индексирование:

git add [имя\_файла] или git add \*

Затем непосредственно добавление в репозиторий:

git commit -m "Commit message"

Желательно при выполнении commit писать комментарий с кратким пояснением выполненных изменений.

Для создания веток используется команда:

git checkout -b [новая\_ветка]

Для переключения на другую ветку:

git checkout [репозиторий]/[ветка]

Для работы с удаленными репозиторием необходимо его подключить:

git remote add origin [url]

Для отправки изменений из ветки master локального репозитория origin на удаленный сервер (в соответствующую ветку):

git push origin master

Для копирования изменений с удаленного сервера в свой локальный репозиторий:

git pull origin master

Для проверки состояния репозитория используется команда:

git status

### Сервис хостинга проектов GitHub

GitHub – это популярный интернет-сервис для хостинга и совместной разработки проектов. GitHub позволяет размещать в открытом (бесплатно) или закрытом (платно) доступе свои проекты, осуществлять совместную разработку, легко создавать новые ветки проектов, а также скачивать код проектов не только через Git, но и просто в виде файлов.

Зарегистрироваться и создать свой открытый репозиторий можно по адресу <https://github.com/>. Короткая инструкция по первым шагам в GitHub приведена ниже в разделе «Дополнительная информация».

### Задание на лабораторную работу

1. Установить на свой рабочий компьютер Git.
2. Попробовать выполнить команды создания локального репозитория, внести изменения в рабочую копию, сохранить их в репозиторий. После каждого из шагов проверять состояние репозитория командой git status
3. Зарегистрироваться на GitHub\*.
4. Создать новый репозиторий через GitHub.
5. Подключиться с рабочего компьютера к репозиторию GitHub и сохранить туда файлы своего локального репозитория или наоборот, создать копию удаленного репозитория у себя на рабочем компьютере.

\*Если вы уже зарегистрированы на GitHub и являетесь его активным пользователем, пункты 3-4 можно пропустить.

### Отчет

В отчете необходимо указать URL вашего репозитория в GitHub, а также скриншоты выполнения основных действий в Git на локальном компьютере (add и commit, status, push…).

Отчет можно предоставить в бумажном виде или прислать на электронную почту [a.vybornova@gmail.com](mailto:a.vybornova@gmail.com). При отправке на электронную почту в теме письма обязательно необходимо указать название предмета – ПО ЦОД, номер лабораторной работы, вашу группу и фамилию, например:

Лаб 1. ПО ЦОД ИКПИ-61 Иванов И.И.

### Дополнительная информация:

1. Книга Pro Git <https://git-scm.com/book/ru/v2>
2. Начало работы с GitHub <https://guides.github.com/activities/hello-world/>
3. Статья о быстром начале работы с Git и Github <https://habrahabr.ru/post/125799/>