МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Факультет информационных технологий и программирования

Кафедра информационных систем

Лабораторная работа № 3

**Изучение системы управления версиями Git**

Выполнила студентка группы № M3306:  
Жаворонков Сергей

Преподаватель:   
Липкин Евгений Олегович

САНКТ-ПЕТЕРБУРГ  
2018

**git init**

создает в директории пустой репозиторий в виде скрытой директории .git.

--shared репозиторий Git будет совместно использоваться несколькими пользователями.

**git clone**

Клонировать репозиторий

git clone -b branch <url> - клонирование определенной ветки

**git add**

Начать отслеживать (добавить под версионный контроль) новый файл, т.е. внести в индекс изменения, которые затем войдут в коммит.

-A добавить изменения из отслеживаемых и неотслеживаемых файлов; . добавить все новые файлы для отслеживания; можно писать сразу имя файла

-u обновить отслеживаемые файлы (добавляет в индекс измененные и удаленные файлы текущей директории, новые файлы не добавляются).

**git status**

показывает состояния файлов в рабочей директории и индексе: какие файлы изменены, но не добавлены в индекс; какие ожидают коммита в индексе. Вдобавок к этому выводятся подсказки о том, как изменить состояние файлов.

-s/--short показывает статус сокращенно; -u/--untracked-files показывает untracked файлы; --ignored показывает игнорируемые файлы.

**git commit**

Запись изменений в репозиторий

-a все файлы индексируются и сразу выполняется коммит;

-v включить в комментарий к коммиту информацию о том какие именно были сделаны изменения в каких файлах;

-m «I’m a message»;

FILENAME внести в индекс и создать коммит на основе изменений единственного файла;

--amend или --amend -m "Новое описание" изменить описание последнего коммита или добавить файлы в последний коммит;

--dry-run показать, что будет добавлено в коммит.

**git push**

—all push все ветки;

--prune удалить удаленную ветку, не имеющую локального аналога

<remote> <branch> отправка ветки на удаленный сервер;

**git checkout**

использоваться для переключения между ветками или коммитами

<branch> переход на ветку;

commit\_id откат до коммита c commit\_id;

-b <branch> создать ветку с переходом на нее

**git remote**

управление набором отслеживаемых репозиториев. Без аргументов отображает список существующих репозиториев.

add <name> <url> добавить удаленный репозиторий;

rename <old> <new> переименовать удаленный репозиторий;

show просмотр информации об удаленном репозитории;

-v посмотреть URL-адреса

**git branch**

позволяет посмотреть список веток в локальном репозитории

-a просмотр всех удаленных и локальных веток

<branch> создание новой ветки;

-d <branch> удаление ветки;

-v просмотр последнего коммита на каждой из веток;

--merged просмотр веток, для которых выполнено слияние;

--no-merged просмотр веток, для которых не выполнено слияние;

--contains <commit> вывод веток, которые содержат данный коммит.

**git pull**

Выполнение git pull, как правило, извлекает (fetch) данные с сервера, с которого вы изначально склонировали, и автоматически пытается слить (merge) их с кодом, над которым вы в данный момент работаете.

origin master забираем изменения из ветки (master) на удаленном сервере (origin) и проводим слияние с активной веткой;

--no-commit не создает новый коммит слияния

--rebase использовать git rebase вместо git merge для интеграции удаленной ветки с локальной

--verbose подробный вывод во время извлечения (pull), который отображает загружаемое содержимое и сведения о слиянии.

**git merge**

объединить две или более истории разработки.

--continue git остановил слияние из-за конфликтов, можно завершить слияние, запустив continue;

git merge feature --squash объединит изменения всех коммитов ветки feature, перенесет их в ветку master и добавит в индекс. При этом коммит слияния не будет создан, вам нужно будет сделать его вручную.

--no-commit не создавать новый коммит автоматически

--squash branchB берёт всю работу на сливаемой ветке (branchB) и сжимает её в один коммит, не являющийся коммитом-слиянием, и помещает его на верхушку текущей ветки;

**git diff**

Если результат работы команды git status недостаточно информативен для вас — вам хочется знать, что конкретно поменялось, а не только какие файлы были изменены — вы можете использовать команду git diff.

Если вы хотите посмотреть, что вы проиндексировали и что войдёт в следующий коммит, вы можете выполнить git diff --cached.

git diff <commit> <commit> - изменения между двумя коммитами

**git log**

просмотр истории комммитов – код коммита, имя и электронная почта автора, дата создания, сообщение коммита.

**git rm**

удалить файлы из рабочего дерева и индекса

-f удалить проиндексированный измененный файл;

--cached удалить файл из индекса (при этом файл остается в рабочем каталоге)

**git fetch**

получение данных из удалённых проектов. Забирает данные в локальный репозиторий, которых у вас еще нет, но не сливает их с какими-либо наработками и не модифицирует то, над чем вы работаете в данный момент.

**git rebase**

взять все изменения, которые попали в коммиты на одной из веток, и повторить их на другой.

-i перезаписать историю коммитов (git rebase -i HEAD~2 объединить последние два коммита в один);

--abort отменить процесс rebase;

--continue продолжить процесс rebase;

master server команда применит изменения из вашей работы над веткой server на вершину ветки master

**git archive**

упаковать в архив указанные файлы, директории или весь репозиторий;

--format=<fmt> - указать формат архива (tar или zip);

--list показать все доступные форматы;

**git gc**

очистка ненужных файлов и оптимизация локального репозитория.

Запускается сборщик мусора в репозитории, который удаляет ненужные файлы из хранилища объектов и эффективно упаковывает оставшиеся файлы. Эта команда выполняет несколько действий: собирает все «рыхлые» объекты (не в pack-файлах) и упаковывает их в pack-файлы; объединяет несколько упакованных файлов в один большой; удаляет недостижимые объекты, хранящиеся дольше нескольких месяцев, упаковывает ссылки в единый файл.

**git stash**

сохранить локальные незакоммиченные изменения.

list посмотреть стэк спрятанных данных;

pop применить последний набор сохраненных изменений в текущей рабочей директории (будет удален из стэка);

; apply stash@{n} применить n-ый набор изменений (не удалит снимок из стэка);

branch <branch> будет создана новая ветка с началом из того коммита, на котором мы находились во время прятанья, команда восстановит в ней вашу работу и затем удалит спрятанное, если оно применилось успешно.

**git show**

показать изменения, внесенные отдельным коммитом.

для коммитов показывает сообщение журнала и текстовую разницу.