Git

# Lection 1

Установка имени

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

установка имейла

git config --global user.email [your\_email@whatever.com](mailto:your_email@whatever.com)

git init – создание репозитория

Установление открытия по умолчанию не в Vim а VSCOde

git config --global core.editor "code --wait"

<https://gist.github.com/isqua/fcafd18019b5ce61e3de31c567d7202b> – сайт популярных редакторовю

# Lection 2

Добавление страницы в репозиторий

Создадим файлик test.txt. Внесем туда изменения

Командой **git status** проверим состояние файлов

Увидим что гит знает что test.txt изменет, но изменения не зафиксированы в репозиторий.

git add . – индексация всех изменений и вызовем git status

файл проиндексирован, Это означает, что git теперь знает об изменении, но изменение пока не перманентно (читай, навсегда) записано в репозиторий. Следующий коммит будет включать в себя проиндексированные изменения.

Если вы решили, что не хотите коммитить изменения, команда состояния напомнит вам о том, что с помощью команды git reset можно снять индексацию этих изменений.

git commit -m "first commit" – фиксация всех изменений, и они будут записаны в репозиторий.

git log просмотр всех коммитов

# Lection 3

Получение истории, возвращение к комитам.

Команда checkout скопирует любой снимок из репозитория в рабочий каталог.

git checkout <hash>

отмена изменений до индексации:

Внесите нежелательные изменение в файл index.html

git status проверим , что файл index.html был изменен, но еще не проиндексирован.

git checkout index.html вернет к предыдущему состоянию(до изменений)

Отмена проиндексированных изменений (перед коммитом)

Внесите нежелательные изменение в файл index.html

git reset HEAD index.html

Команда reset сбрасывает буферную зону к HEAD. Это очищает буферную зону от изменений, которые мы только что проиндексировали.

Команда reset (по умолчанию) не изменяет рабочий каталог. Поэтому рабочий каталог все еще содержит нежелательный комментарий. Мы можем использовать команду checkout

git checkout index.html

git status

Наш рабочий каталог опять чист.

Удаление коммита

git reset --hard HEAD~1 гле 1 количество комитов вниз. Затрет все изменения в этих коментариях.

# Lection 4

Внесение изменений в коммиты

git commit --amend -m "Add an author/email comment"

изменение предыдущего коммита

git rebase -i HEAD~2 – переименовать / склеить коммиты (2 последних).

git push –force принудительная отправка в ветку

# Lection 5

Ветки

git checkout -b <имяветки> создание ветки.  Является шорткатом для git branch <имяветки> за которым идет git checkout <имяветки>.

git branch –просмотр текущей ветки.

git checkout master – переключение между ветками

git log -- all – просмотр всех комитов в ветках

# Lection 6

Слияние

git merge master Слияние переносит изменения из двух веток в одну. Давайте вернемся к ветке style и сольем master с style. (нужно находится в ветке style)

Изменения из мастер подтянуться в нашу ветку (style)

Конфликты слияния

git rebase master делает то же самое что и merge?

Дерево коммитов ветки style было переписано таким образом, что ветка master является частью истории коммитов. Это делает цепь коммитов линейной и гораздо более читабельной.

git merge newBranch - Слияние style в master

# Lection 7

Удаленные ветки

* git remote - Узнать об именах удаленных репозиториев.

git fetch - извлечь изменения из удаленного репозитория.

git merge origin/master - Слейте извлеченные изменения в локальную ветку master

git pull - Извлечение и слияние изменений (эквивалентно двум следующим шагам: git fetch git merge origin/master )

git branch –a - просмотр всех веток

# Добавление ветки наблюдения

git branch --track style origin/style - Добавьте локальную ветку, которая отслеживает удаленную ветку.

Отправка данных в удаленный репозиторий

git push - добавление изменений с локальный репозиторий

git push origin myRepository – добавить изменеия в удаленный репозиторий.

git push –force принудительная отправка в ветку

git rebase -i HEAD~2 – переименовать / склеить коммиты (2 последних).