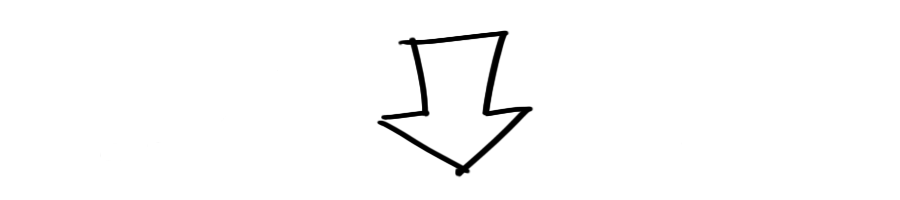
git - the simple guide

простое руководство по работе с git. Ничего сложного ;)



установка

[Скачать git для OSX](http://git-scm.com/download/mac)

[Скачать git для Windows](http://msysgit.github.io/)

[Скачать git для Linux](http://book.git-scm.com/2_installing_git.html)

создание нового репозитория

Для того чтобы создать новый репозиторий git, необходимо открыть папку где вы хотите его разместить и выполнить команду  
git init

получение репозитория

Создать локальную рабочую копию репозитория можно командой  
git clone /путь/к/репозиторию  
когда используется удаленный сервер, команда будет  
git clone юзер@хост:/путь/к/репозиторию

рабочий процесс

Ваш локальный git репозиторий состоит из трех "сущностей". Рабочий каталог (Working Directory) содержит файлы. Индекс (Index) или область подготовленных файлов (Staging Area), содержит информацию о том, что должно войти в следующий коммит и HEAD указывает на последний коммит что вы сделали.



подготовка и коммит

Чтобы подготовить изменения (добавить их в **Индекс**), используйте  
git add <имя\_файла>  
git add \*  
Это первый шаг в основном рабочем процессе. Сделать коммит подготовленных изменений можно командой  
git commit -m "Описание коммита"  
Теперь изменения закреплены в локальном репозитории, и на них указывает **HEAD**, но еще не в удаленном репозитории.

отправка изменений

Чтобы отправить эти изменения в ваш удаленный репозиторий, выполните  
git push origin master  
Можно изменить *master* на любую другую ветвь, чтобы отправить изменения на неё.  
  
Если вы еще не клонировали существующий репозиторий и хотите подключить ваш к удаленному, вам нужно добавить его, выполнив  
git remote add origin <адрес\_сервера>  
Теперь вы можете отправлять изменения на удаленный репозиторий

ветвление

Ветки используются для разработки функциональности, изолированной от остальной. Ветка *master* используется по умолчанию, когда вы создаете репозиторий. Используйте другие ветки для разработки и слияние в *master*, когда разработка завершена.



Создать новую ветку с названием "feature\_x" и переключиться на неё можно командой  
git checkout -b feature\_x  
переключиться обратно на *master*  
git checkout master  
удалить ветку  
git branch -d feature\_x  
ветка не будет доступна тем, кто пользуется с вами удаленным репозиторием, пока вы не отправите её туда  
git push origin <имя\_ветки>

обновление и слияние

Обновить ваш локальный репозиторий можно командой  
git pull  
которая заберет изменения из удаленного репозитория и проведет слияние с активной веткой.  
Для того чтобы слить другую ветку с активной (например master), используйте команду  
git merge <имя\_ветки>  
В обоих случаях git пытается автоматически слить изменения. К сожалению, это не всегда возможно и результатом станет *конфликт*. Вы ответственны за разрешение возникших *конфликтов*, путем ручного редактирования файлов, указанных git. После изменений вам надо пометить их как слитые  
git add <имя\_файла>  
перед слиянием вы можете предварительно посмотреть на изменения  
git diff <имя\_ветки> <имя\_другой\_ветки>

метки

Рекомендуется использовать метки для закрепления момента выпуска версий. Это популярная практика, которая также используется в SVN. Создать новую метку с именем *1.0.0* можно, выполнив  
git tag 1.0.0 1b2e1d63ff  
*1b2e1d63ff* это первые десять цифр уникального идентификатора (id), с которым будет связана метка. Чтобы посмотреть идентификаторы коммитов, выполните  
git log  
Можно использовать меньшее количество символов в качестве идентификатора, с учетом того, что он является уникальным.

замена локальных изменений

В случае, если вы сделали что-то не то, вы можете заменить локальные изменения, используя команду  
git checkout -- <имя\_файла>  
произойдет замена изменений в вашем рабочем каталоге, на то, что сейчас находится в HEAD. Изменения, уже внесенные в индекс, также как и новые файлы, будут сохранены.

Если же вы хотите удалить все ваши локальные изменения и коммиты, получите (fetch) последние изменения с сервера и укажите локальной ветке master на них вот так  
git fetch origin  
git reset --hard origin/master

твики и удобные команды

встроенный в git графический интерфейс  
gitk  
использовать цветной вывод в терминале  
git config color.ui true  
выводить в логе коммит на одной строке  
git config format.pretty oneline  
интерактивный способ добавления в индекс  
git add -i

полезные ссылки

графические интерфейсы

* [GitX (L) (OSX, открытый исходный код)](http://gitx.laullon.com/)
* [Tower (OSX)](http://www.git-tower.com/)
* [Source Tree (OSX, бесплатный)](http://www.sourcetreeapp.com/)
* [GitHub for Mac (OSX, бесплатный)](http://mac.github.com/)
* [GitBox (OSX)](https://itunes.apple.com/gb/app/gitbox/id403388357?mt=12)

руководства

* [Git Community Book](http://book.git-scm.com/)
* [Pro Git (перевод)](http://progit.org/book/ru/)
* [Think like a git](http://think-like-a-git.net/)
* [GitHub Help](http://help.github.com/)
* [A Visual Git Guide](http://marklodato.github.com/visual-git-guide/index-en.html)