[cd] - (change directory) попасть в нужную директорию.

Пример: cd /c/data/project/

[pwd] - показать текущую директорию.

[ls] - показывает наличие файлов и папок в текущей директории.

[git clone SSH-key] - клонировать удаленный репозиторий в текущую директорию.

Можно использовать SSH ключ или HTTP веб-страницы.

[git remote -v] - показывает на каком ресурсе находится репозиторий (например GitHub.com).

[git status] - статус файлов.

[git add] - добавить измененные файлы в Index. Пример: git add file.txt

\*\*\* [git add -a] - возможно вносит в Index все модифицированные файлы сразу (надо проверить)

[git commit -m "коментарий к коммиту"] - закоммитить файлы.

[git log] - позволяет увидеть историю коммитов.

[git show] - многогранная команда, призванная показывать какую-то инфу в зависимости от назначенных флагов.

[git push] - отправить готовые файлы в удаленный репозиторий.

[git pull] - подтянуть последнюю версию с удаленного репозитория.

[git gui&] - открывает git gui в новом окне.

[git k&] - открывает git k в новом окне.

[vim file.txt] or [vi file.txt] - открывает указанный файл в графическом редакторе vim.

Чтобы редактировать текст в редакторе vim нужно нажать "Insert" либо ‘A’.

Выйти из режима редактирования - "Esc" и далее:

[:wq] - выйти и сохранить изменения; ('w' - сохранить,'q' - выйти, 'x' - сохранить и выйти)

[:q!] - выйти без сохранения изменений.

[git reset --soft] - переместить файл из Commit в Index.

Если при команде git reset не ставить какой-либо флаг, по умолчанию добавляется флаг --mixed,

что отправляет файл в File System, минуя Index.

[git reset --hard] - удаляет какие-либо изменения. Осторожно! Не имеет обратного действия, нельзя восстановить.

[git reset --soft HEAD~1] - вернуть на один коммит.

[git reset --mixed HEAD^] - вернуть на один коммит в File system (одно ушко = 1 коммит).

[git reset --hard HEAD^^] - откатиться на два коммита безвозвратно.

[git restore file.txt] - откатывает изменения в файле до состояния последнего коммита.

[git diff] - показывает, какие изменения были внесены в файл с момента последнего коммита.

[git diff --staged] - показывает, какие изменения в файлах имеются, если файлы уже в Index.

Т.е. если файлы были изменены и добавлены в Index, команда git diff (без флагов) ничё не покажет.

??? [git commit -am "коментарий"] - позволяет перепрыгнуть файлам Index и сразу с File System попасть в Commit.

[git mv] - позволяет переименовывать и перемещать файл.

Например: git mv file.txt file2.txt - после этой команды файл будет называться file2.txt;

git mv file.txt /data/file2.txt - перенесёт файл в папку data и переименует его в file2.txt.

Причем, команда git mv сразу переносит этот файл в Index, минуя File System.

[clean] - очищает консоль.

[git rm file.txt] - (remove - удалить) - удаляет выбранный файл.

[git restore --staged file.txt] - отменяет удаление файла.

??? [git rm --cached file.txt] - удалить файл в коммите, но при этом оставить его в каталоге.

[touch file.txt] - создать file.txt

[mkdir folder] - создать папку folder

[rmdir folder/] - удалить папку folder

[cat file.txt] - посмотреть содержимое файла

[history] - история всех команд, кот. вводились.

[git checkout -- <file.txt>] – команда для отмены изменений. Например, если мы внесли изменения в один или несколько файлов, в git bash мы видим их как модифицированные, и понимаем, что нужно в одном из файлов отменить изменения, то можем вернуть всё как было с помощью данной команды.

[git checkout .] – отменить изменения во всех изменённых файлах в данной директории и в поддиректориях.

Команда checkout работает в том случае, когда у файлов была какая-то предыдущая версия и мы хотим откатить изменения. Но если файлы в директории новые, checkout не сработает. Для этого нужна команда

[git clean –xdf] – позволяет удалить новые файлы (удаляет ВСЕ новые файлы).

Если мы сделали коммит, а потом поняли, что что-то забыли дописать, то можно дописать в файле нужный контент, а потом воспользоваться командой:

[git commit –amend –m “commit message”] И в таком случае новый контент войдет в уже созданный коммит.

[git revert <sha1>] – отменяет / зеркалит тот контент, что мы добавили / удалили. Т.е. если мы что-то внесли новое, команда revert удалит это. Если удалили что-то, revert вернет. И эта команда создает новый коммит с изменениями.

.gitignore

Синтаксис:

\*.log – скрыть все файлы с расширением log.

!error.log – наоборот, сделать исключение для гита. Т.е. если дать команду скрывать все файлы типа log, и дать вторую команду !error.log, то файл error.log будет читаться гитом.

Можно также игнорировать папки и т.д.

**BRANCH.**

[git branch --all] – посмотреть все имеющиеся ветки проекта.

[git checkout –b <feature>] – (feature – название новой ветки), команда, создающая новую ветку.

[git checkout <master>] – позволяет перейти на ветку master.

Как сделать fast-forward merge. >>> Нужно перейти в ту ветку, в которой будет конечный результат, и пользуемся командой:

[git merge <feature>]

Fast-forward merge – это когда мы имели master ветку, создали feature ветку. Вносили какие-то изменения, пришли к мнению, что все эти изменения на feature ветке хороши и сливаем весь контент на одну ветку (на master).

master 111**1** **2**

feature 222**2**

Где обычные цифры – коммиты. **1** – HEAD master, **2** – HEAD feature, **2** – HEAD master & feature. Sha1 feature **2** = sha1 feature/master **2**, т.к. этот коммит оказался последним верным.

Non-fast-forward merge.

master 111 **1** **X**

feature 222**2**

В данном случае **X** является новым коммитом, объединяющим в себе контент и **1** и **2**.

Как решить конфликт при слиянии веток (merge conflict): Исправить содержимое файлов с конфликтами, сделать коммит изменений.

[git merge --abort] – отменить слияние веток (обычно, в случае конфликта контента двух веток, когда на это сейчас нет времени).

REBASE: Необходима в том случае, когда, допустим, в ветке master от коммита C мы отвели новую ветку, сделали какую-то работу, а в это время в master ветке разработчик создал новые штуки, полезные для нашей ветки, и нам нужно, чтобы в нашей работе они отобразились. Т.е. изначально родителем коммита 1 был коммит С, а нам нужно, чтобы родителем стал коммит E.

Было: Стало:

master A – B – C – D – E master A – B – C – D - E

| |

feature 1-2 feature 1-2

CHERRY-PICK.

Операция копирования коммита в другое место называется cherry-pick. Т.е. эта команда позволяет подхватить коммит из любой ветки и добавить в нужную нам ветку. При слиянии (merge) git сам знает, как совмещать эти штуки (конфликта не будет).

**TAGS**. Тег – это текстовый маркер, которым можно пометить коммит.

A – B – C(tag: ver1) – D – E – F(tag: ver2) – G - … - (n)

[git tag ver1] – mark commit with tag.

[git tag –list] – view tags.

[git push --tags] – push.

[git checkout ver1] – check it out.

**STASHING** – возможность сохранить свою работу во временное хранилище, пока работа ещё не закончена и не «закоммиттена».

*WIP – “work in progress”.*

[git stash save “description”] – save working directory.

[git stash list] – view stashes.

[git stash pop] (*and remove from stash*) &

[git stash apply] (*leave in stash*) – bring them back.

[git stash drop] (*clear*) – remove.

A – B – C – D –(WIP)