**Основні команди Git**

[Комментарии ()](http://webit.in.ua/article/Osnovni-komandi-Git/#comment-form)2015-04-03 18:51:06 By admin 



Файл конфігурації Git'а для Вас, може зберігатися в трьох різних місцях

1. На рівні ОС  
   **/etc/gitconfig**
2. В домашньому каталозі  
   **~/.gitconfig**
3. В каталозі проекту  
   **my\_project/.git/config**

git config --system - задавання системної конфігурації

git config --global - задавання конфігурації користувача

git config - задавання конфігурації для конкретного проекту

git config --global user.name "<my\_name>"

git config --global user.email "<my\_email>"

git config --global core.editor "nano" - задаємо редактор за змовчанням

git config --global color.ui true - задаємо використання кольорів при виводі в консоль

git config --global core.excludesfile ~/.gitignore\_global - вказуємо де знаходиться файл із правилами глобального ігнорування

export PS1=\W'$(\_\_git\_ps1 "(%s)") > ' - змінює командний рядрк на **<текучий каталог>(<name\_branch>)>**

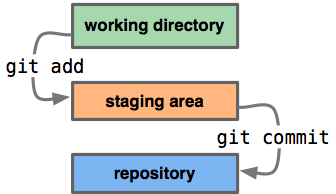
git config branch.<branch\_name>.remote <alias> - задаємо відслідковування бранча на віддалений репозиторій

git config --list - перегляд конфігурації

git config --global alias.st status - приклад створення аліаса (st) для команди status

Деякі аліаси, які стали в багатьох випадках стандартом

git config --global alias.co checkout  
git config --global alias.ci commit  
git config --global alias.br branch  
git config --global alias.df diff  
git config --global alias.dfs "diff --stage"  
git config --global alias.logg "log --graph --decorate --oneline --abbrev-commit --all"



git init - створення git-репозитарія

git add <filename> - добавити файл в буфер

git add . - добавити усі файли в буфер

git commit -m "commit text" - коміт. відправляємо вміст буфера в репозиторій

git -am "commit text" - добавлення в буфер та виконуємл коміт однією командою (не працює для нових та видалених файлів, лише для файлів які були змінені)

git staus

git log - найчастіша команда, показує логи (коміти)

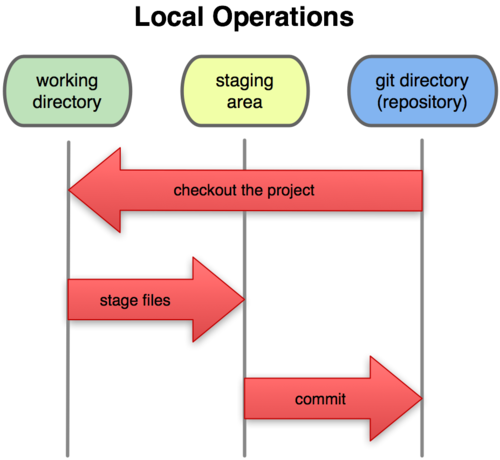
git log --oneline - стиснутий вигляд перегляду комітів

git log --since="2015-01-01" - показати коміти від 2015-01-01  
git log --until="2015-01-01" - показати коміти до 2015-01-01  
git log --author=<name> - фільтр по автору коміта  
git log --grep="temp" - пошук усіх логів, в яких є слово temp  
git log 2907d12..acf7750 - показати коміти від 2907d12 до acf7750 (sha1)

git log -p - покаже логи із прееглядом змін (що було зроблено)  
git log --graph - покаже логи у вигляді дерева

git log --oneline --graph --all --decore - симпатичний список логу

git show <sha-1> - перегляд коміта



git diff <filename> - показати відмінність у  файлу в робочій директорії та репозиторієм  
git diff --storage <filename> - показати відмінність у редагуванні файлу із буферної зони та репозиторієм

git diff --color-words <filename> показує зміни одну біля одної (в одному рядку)

git diff <sha-1> <filename> - покаже зміни у файлі в коміті <sha-1>

git diff <sha-1>..<sha-1> - покаже зміни між двома сліпками

опції для git diff:

* -b ігнорує зміни з пробілами (ігнорує якщо було змінено 2 пробіла на один і так далі)
* -w ігнурує усі пробіли

git diff <name\_branch1>..<name\_branch2> - показує зміни між бранчами

git mv <filename> <new\_filename>

git rm --cached <filename> - видалення файлу із репозиторія (це приведе до того, що файл більше не буде відслідковуватися)



git checkout -- <filename> - взяти файл із репозиторія в робочу директорію (відміна внесених змін)

git checkout <hash> <filename> - чекаут файлу із попередньої рівізії <hash> (файл попадає в буферну зону!)

git reset HEAD <filename> - витягнути файл із буфера назад в робочу директорію

git commit --amend -m "commit text" - зміна останнього коміта

git revert <hash> - відміняє коміт (виконує повну протилежність, те що було знищено вертає, те що було добавлено знищує)

git reset --soft <hash>- переміщує HEAD

git reset --mixed <hash>- переміщує HEAD на конкретний коміт, та змінює буфер, щоб він відповідав репозиторію, при цьому робоча директорія не змінюється

git reset --hard <hash> - переміщує HEAD на конкретний коміт, усі зміни які були за цим комітом будуть знищені

gir clean -n тестово показує, які невідслідковані файли будуть знищені командою git clean -f

git clean -f знищує невідслідковані файли із робочої директорії (яких немає в буфері та репозиторії)

git ls-tree HEAD - покаже файли для HEAD

.gitignore - файл,який містить перелік нових файлів (один файл на рядок), які потрібно ігнорувати (можливе використання регулярних виразів)  
Приклад:  
\*.php - ігнорувати усі файли із розширенням .php  
!index.php - не ігнурувати index.php ( ! - вказує, що не портрібно ігнорувати)

git branch [-r,-a]- покаже список бранчів (\* - активний бранч) (-r покажи віддалені бранчі, -a покаже локільні і віддалені бранчі)

git branch <name\_branch> - створення нового бранча

git branch --merged - покаже перелік бранчів в текучому бранчі

git branch -m <name\_branch> <new\_name\_branch> - переіменування бранча

git branch -u <branch\_name> <alias>/<branch\_name> - задаємо відслідковування бранча на віддаленому репозиторію

git checkout <name\_branch> - переключення на бранч

git checkout -b <name\_branch> - створення нового бранча та переключення на нього

git checkout -d <name\_branch> - знищення бранча

git checkout -D <name\_branch> - знищення бранча (в незалежності, злитий він з іншим бранчем чи ні)

git merge <name\_branch> - зливання бранча <name\_branch>із тикучим бранчом (в якому ми зараз знаходимя)

git merge <alias>/<name\_branch> - зливання бранча <name\_branch> із репозиторія <alias> із тикучим бранчом (в якому ми зараз знаходимя)

git merge --no-ff branch\_name - не робити швидке зливання, зробити  новий коміт із повідомленням (докуиентація процесу зливання)

git merge --ff-only branch\_name - зробити зливання, лише, якщо воно буде швидким

git merge --abort -    перервати(вийти з режиму) зливання

git stash list - перегляд записів у копілці

git stash save "comment" - збереження змін в "копілку"

git stash show stash@{0} -  перегляд елемента stash@{0} в копілці

git stash show -p stash@{0} -  перегляд елемента stash@{0} в копілці із виводом інформації про зміни

git stash pop <element\_name> - переносить зміни із копілки в текучу робочу директорію (якщо не вказати елемент, то по змовчанню бере перший елемент)

git stash apply - копіює зміни із копілки в текучу робочу директорію

git stash drop <element\_name> - знищення елементу із копілки

**Віддалені репозиторії**

git remote [-v] - список усіх віддалених репозиторіїв, які відомі git'у

git remote add <alias> <url> - добавлення віддаленого серверу під назвою <alias>, який ми можемо знайти за адресою <url>

git remove rm <alias> - знищення запису про віддалений репозиторій

git push [-u] <alias> <branch\_name> - відправити код на віддалений репозиторій ( -u відслідковувати віддалений репозиторій)

git push - відправляє внесені зміни на віддалений репозиторій, який раніше був заданий як відслідковуваний

git push <alias> :<branch\_name> - знищення бранчу на віддаленому репозиторію

git push <alias> <local\_branch\_name>:<branch\_name> - відправлення локального бранчу <local\_branch\_name> у віддалений репозиторій <alias> у бранч <branch\_name>

git push <alias> --delete <branch\_name> - ще одна (більш зрозуміла команда) знищення бранчу

git clone <url> [<folder>] - клонування віддаленого репозиторія

git pull = git fetch +  git merge

* Git не відслідковує пусті директорії, оскільки він відслідуковує файли. Потрібно створити пустий файл, наприклад **.gitkeep**
* Про commit:  
   - Перший рядок commit'а не більше 50 символів, а наступні рядки commit'а не більше 72. Текст коміта в теперішньому часі.  
   - коміт фіксить баг, а не я фіксю баг (правильно: "fix bug","fixes bug", не правильно "fixed bug")  
   - коміти мають бути не загальні, а більш точні, щоб було зрозуміло про що конкретно йдеться:  
        bad: "update login code"  
        good: "change user authentication to use Blowfish"
* копілка - четверта область git'а, куда можна зберігати зміни для тимчасового збереженні, які ми не бажаємо комітити
* Популярні git-хостингові компанії  
  [GitHub](http://github.com/)  
  [BitBucket](http://bitbucket.org/)  
  [GitOriuos](http://gitoriuos.org/)
* Пакет для створення git-хостинга на своєму сервері -  
  [Gitosis](http://gtihub.com/tv42/gitosis) (не розвивається)  
  [Gitolite](http://github.com/sitaramc/gitolite)

**Git auto-complation**

|  |  |
| --- | --- |
| 1 | cd ~ |
| 2 | curl -OL <https://github.com/git/git/raw/master/contrib/completion/git-complation.>bash | |

|  |  |
| --- | --- |
| 3 | mv ~/git-complation.bash ~/.git-complation.bash |

Редагуємо ~/.bash\_profile або ~/.bashrc

|  |  |  |
| --- | --- | --- |
| 1 | if [ -f ~/.git-completion.bash ]; then | |
| 2 | source ~/.git-competion.bash |

|  |  |
| --- | --- |
| 3 | fi |

**Приклад початку роботи над новою фічою із віддаленим бранчем**

git checkout master

git fetch

git merge origin/master

git checkout -b feedback\_form

git add feedback.html

git commit -m "Add customer feedback form"

git fetch

git push -u origin feedback\_form