Git - lista najczęściej wykorzystywanych komend

Git fetch

git fetch -all - ściąga informacje o wszystkich branchach
git fetch remote branchZrodlowy:branchDocelowy merguje (tylko fast forward) bez konieczności przełączania się na te branche jako remote można użyć . (lokalne repozytorium) -x usuwa też ignorowane pliki

Git clean

git clean -n wyświetla co będzie usunięte
-i tryb interaktywny
git clean nieodwracalnie usuwa wszystkie nie śledzone pliki, odwracalnie można to zrobić przez git stash --all ale zapisywane są też śledzone pliki, przeważnie wymaga modyfikatorów -fd

Git checkout

git checkout -b nazwaBrancha tworzy nowego brancha i przechodzi na niego
-B nadpisuje brancha jeśli istniał
git checkout - **przełącza się na poprzedniego brancha** to skrót od git checkout @{-1} inne wartości są dozwolone
git checkout -m lub -- merge robi merga na working area (bez commita)
git checkout --ours -- plik lub -- theirs wybór jednej ze stron przy konflikcie
git checkout A...B przełącza się na ostatni wspólny commit z tych branchy

git checkout plik przywraca plik do postaci zapisanej w commicie, może być trzeba poprzedzić plik przez -- plik
git checkout '*.java' przywraca wszystkie pliki .java do postaci zapisanej w ostatnim commicie

Git reset

reset przesuwa HEADa w obrębie brancha, checkout zmienia brancha, oba obsługują --patch do operowania na fragmentach plików

git reset --soft cofa HEAD (czyli commit)
-- mixed (domyslny) cofa HEAD i index (czyli cofa commit i add)
-- hard cofa heada, index i working area (czyli commit, add i edycje) podając plik jako argument resetu jest on przywracany do zawartości w czasie commitu (podanego jako argument lub HEAD), sam HEAD się nie zmienia

git checkout [branch]--file == git reset [branch] --hard file

Git revert

git revert -m 1 HEAD cofa ostatni commit (i wskazuje który rodzic obowiązuje, przy commitach mergujacych) 1 to branch docelowy 2 mergowany robi problemy w przypadku chęci wprowadzenia cofniętych zmian w innym branchu, rozwiązanie to revert commitu który revertował

Git add

git add -A dodaje do indeksu wszystko
git add . dodaje nowe i zmodyfikowane bez usuniętych
git add -u dodaje zmodyfikowane i usunięte bez nowych
git add -i dodaje interaktywnie
git add -p lub --patch umożliwia dodawanie fragmentów plików Jeśli * przy dodawaniu jest poprzedzone \ wtedy dodawana jest też zawartość **podkatalogów**
git rm oraz git mv robią to samo co Unixowe odpowiedniki plus dodanie do indeksu

Git commit

git commit -a dodaje do indeksu wszystkie śledzone pliki
git commit --amend „dokleja” zawartość indeksu do ostatniego commitu, można zmodyfikować wiadomość przez -m
git commit --interactive tryb interaktywny

Git stash

git stash dodaje do stasha tylko śledzone pliki, jest równoważne z
git stash save
git stash -include-untracked lub -u dodaje też nieśledzone
git stash --all dodaje też ignorowane
git stash show daje mniej informacji niż git show
git stash apply stash@{2} wprowadza zmiany z drugiego

stasha drop usuwa
stasha pop wprowadza zmiany i usuwa
branch wprowadza zmiany i usuwa

Git merge

git merge co doCzego
git merge --no-commit dodaje do indeksu i nie commituje
git merge master @{upstream} lub @{u} zastępuje to nazwę brancha zdalnego (origin/master)
git merge --abort
git merge -s ours lub theirs narzuca strategię mergowania
git merge-base szuka wspólnego przodka dwóch lub więcej commitów, przy trzech argumentach szuka przodka pierwszego oraz commita mergującego drugi i trzeci, jeśli ma nie brać pod uwagę commita mergującego trzeba dodać parametr --octopus

Git rebase

git rebase master server „dokleja" master do serwera
git pull --rebase zmiana domyślnego działania pulla
git rebase -i pozwala modyfikować każdy z commitów
git rebase --skip pomija problematyczny commit
git rebase --abort cofa rebase
git rebase --onto podstawa do Pominiecia docelowy dokleja do podstawy commity z docelowego (który jest odbity od pominiętego doPominięcia)