**git config**

$ git config –-system ... относится ко всей системе (file *Git/etc/gitconfig* )

$ git config –-global ... настройки конкретного пользователя (file

*C:/Doc&Settings/$User/,gitconfig* OR *C:/Users/$User/.gitconfig*)

$ git config ... конфигурация отдельного проекта, в котором в данный момент находимся

(file .*git/config*)

$ git config --global user.name "John Doe"

$ git config --global user.email johndoe@example.com

$ git config --global core.editor emacs

$ git config --global merge.tool vimdiff

$ git config user.name - проверка значения ключа

$ git config --list – показывает все имеющиеся настройки

**git clone**

$ git clone [URL]

$ git clone git://github.com/my-tasks/JavaLab.git – клонирует репозиторий с сервера

**git status**

$ git status – состояние всех файлов репозитория

**git add**

$ git add [file or folder or .] – индексация выбранных файлов/катологов/всего контента

$ git add --all – индексация всех файлов

**git reset**

$ git reset HEAD [file or folder or .] – исключение из индекса файла, каталога, всего проиндексированного контента

**git checkout**

$ git checkout [file or folder or .] – откат изменений данных (изменения будут утеряны)

**git diff**

$ git diff - показывает непроиндексированные изменения

$ git diff --cached

$ git diff --staged – показывает проиндексированные изменения

**git rm**

$ git rm [file or folder or \*] – удаление данных из репозитория, можно использовать glob-шаблоны

$ git rm -f [file or folder or \*] – удаление уже проиндексированных данных из репозитория

$ git rm --cached [file or folder or \*] – удаление данных из-под наблюдения git без удаления с винчестера

**git mv**

$ git mv [fromFile] [toFile] – переименование файла

**git commit**

$ git commit – клонирует репозиторий с сервера

$ git commit –m “your comment” – фиксация изменений с комментарием

$ git commit –a –m “your comment” – фиксация изменений с автоматическим add для всех отслеживаемых измененных файлов

$ git commit --amend – замещение последнего коммита

**git tag**

$ git tag – вывод всех меток в алфавитном порядке

$ git tag [name] – создание легковесной метки на последний коммит

$ git tag –a [name] –m ‘comment’ – создание аннотированной метки

$ git tag –s [name] –m ‘comment’ – создание подписанной метки

$ git tag [name] [short-hash]

$ git tag –s [name] –m ‘comment’ [short-hash] – создание метки на конкретный коммит

$ git show [name] – вывод информации о метке (включая коммент/прикрепленную подпись)

$ git tag –l ‘v1.4.\*’ – поиск меток по шаблону

$ git tag –v [name] – верификация подписанной метки (по ssh-ключу)

$ git push origin [name] – отправка метки на удаленный репозиторий

$ git push origin --tags – отправка всех меток на удаленный репозиторий

**git log**

$ git log – история коммитов

$ git log –p – история коммитов с показом изменений

$ git log –-stat – история коммитов с информацией по объему измененных данных

$ git log --pretty=oneline – история коммитов в одну строку

$ git log --pretty=format:"%h %ad | %s%d [%an]" --graph --date=short – история коммитов в формате

$ gitk – вызов графического интерфейса для показа истории коммитов

**Описание выводимых данных**

%H Хеш коммита

%h Сокращённый хеш коммита

%T Хеш дерева

%t Сокращённый хеш дерева

%P Хеши родительских коммитов

%p Сокращённые хеши родительских коммитов

%an Имя автора

%ae Электронная почта автора

%ad Дата автора (формат соответствует параметру --date=)

%ar Дата автора, относительная (пр. "2 мес. назад")

%cn Имя коммитера

%ce Электронная почта коммитера

%cd Дата коммитера

%cr Дата коммитера, относительная

%s Комментарий

**Параметры для git log**

-p Для каждого коммита показывать дельту внесённых им изменений.

--word-diff Показывать изменения на уровне слов.

--stat Для каждого коммита дополнительно выводить статистику по изменённым файлам.

--shortstat Показывать только строку changed/insertions/deletions от вывода с опцией --stat.

--name-only Показывать список изменённых файлов после информации о коммите.

--name-status Выводить список изменённых файлов вместе с информацией о добавлении/изменении/удалении.

--abbrev-commit Выводить только первые несколько символов контрольной суммы SHA-1 вместо всех 40.

--relative-date Выводить дату в относительном формате (например, "2 weeks ago") вместо полной даты.

--graph Показывать ASCII-граф истории ветвлений и слияний рядом с выводом лога.

--pretty Отображать коммиты в альтернативном формате. Возможные параметры: oneline, short, full, fuller и format (где вы можете указать свой собственный формат).

-(n) Показать последние n коммитов

--since, --after Ограничить коммиты теми, которые сделаны после указанной даты.

--until, --before Ограничить коммиты теми, которые сделаны до указанной даты.

--author Показать только те коммиты, автор которых соответствует указанной строке.

--committer Показать только те коммиты, коммитер которых соответствует указанной строке.

**git branch**

$ git branch – список всех веток

$ git branch -v – список всех веток с их последними коммитами

$ git branch --merged – список всех веток, которые слиты с текущей

$ git branch –no-merged – список всех веток, которые не слиты с текущей

$ git branch [name] – новая ветка (указатель но последний коммит)

$ git branch –d [name] – удаление ветки [name]

$ git checkout [name] – перемещение HEAD (переход) на ветку [name]

$ git checkout –b [name] – создание ветки [name] и сразу перемещение HEAD (переход) на нее

$ git push [remote\_rep] [local\_branch] – отправка локальной ветки [local\_branch] в удаленную ветку репозитория [remote\_rep] с таким же именем

$ git push [remote\_rep] [local\_branch]:[remote\_branch] – добавление изменений локальной ветки [local\_branch] в удаленную ветку репозитория [remote\_rep] с именем [remote\_branch]

$ git checkout –b newbranch origin/newbranch – создание ветки на основе удаленной prigin/newbranch

**git merge**

$ git merge [branch\_name] – «вливание» изменений с ветки [branch\_name] в текущую ветку: fast-forward, если последний коммит HEAD является предком последнего коммита ветки [branch\_name], автокоммит (коммит-слияние). Recoursively: если HEAD и [branch\_name] не связаны (слияние по общему на основе общего последнего коммита), автокоммит (коммит-слияние), в случае конфликта (одни и те же данные изменены в обоих ветках) – автокоммит-слияние не сделан, исправляем конфликты вручную, add и commit вручную.

**git remote**

$ git remote – вывод удаленных репозиториев

$ git remote -v – вывод удаленных репозиториев c их url

$ git remote add [name] [url] – добавление удаленного репозитория

$ git remote rm [name] – удаление удаленного репозитория

$ git remote show [name] – вывод информации об удаленном репозитории

$ git remote rename [oldName] [newName] – переименование удаленного репозитория

**git fetch**

$ git fetch [name] – получение изменений удаленного репозитория (без слияния)

**git pull**

$ git pull [name] [branch] – получение изменений удаленного репозитория со слиянием

**git push**

$ git push [name] [branch] – получение изменений удаленного репозитория со слиянием