https://github.com/kontur-courses/git

H1. Гибкое конфигурирование и качественная документация

Гибкая настройка под любой процесс

- `git help` — список команд
- `git <command> -h` — помощь по команде в терминале
- `git <command> --help` — документация по команде в браузере
- `git config -e --system` — редактировать настройки системы
- `git config -e --global` — редактировать настройки пользователя
- `git config -e` — редактировать настройки репозитория
- `git config --global user.name "<name>"` — задать имя пользователя
- `git config --global user.email "<email>"` — задать почту пользователя

Aliases

- `git config --global alias.it "!git init && git commit -m 'Initial commit' --allow-empty"`
- `git config --global alias.st "status -sb"`
- `git config --global alias.call "!git add . && git commit -m"`
- `git config --global alias.commend "commit --amend --no-edit"`
- `git config --global alias.graph "log --oneline --decorate --graph --all"`
- `git config --global alias.to "checkout"`
- `git config --global alias.pushup "push -u origin HEAD"`
- `git config --global alias.please "push --force-with-lease"`
- `git config --global alias.puff "pull --ff-only"`
- `git config --global alias.pure "pull --rebase --autostash"`
- `git config --global alias.undo "reset --soft HEAD^"`

Принципы Git

S1. Все локально

Все данные хранятся в локальных репозиториях, изменения между ними можно синхронизировать

- `git init` — создать пустой репозиторий
- `git clone <url>` — склонировать репозиторий в новую директорию

S2. Хранятся состояния директории, постепенная сборка коммита

Хранятся файлы, разница вычисляется на лету

Commit index для сборки коммита

- `git add .` — добавить все измененные файлы в индекс
- `git commit -m <msg>` — записать изменения из индекса в репозиторий
- `git status -sb` — вывести состояние директории и индекса кратко с указанием текущей ветки
- `git restore .` или `git checkout .` — отменить изменения в директории по индексу
- `git restore -S .` или `git reset .` — отменить изменения индекса по коммиту (отмена `git add .`)
- `git rm <filename>` — удалить файл из индекса, чтобы перестать хранить его историю в репозитории
- `git show <commit>` — показать содержимое коммита
- `git log --oneline --decorate --graph` — вывести историю коммитов от HEAD в виде дерева
- `git log --oneline --decorate --graph --all` — вывести историю всех коммитов в виде дерева
- `gitk` — открыть графическое представление репозитория
- `git clean` — удалить неотслеживаемые файлы из директории

S3. Манипуляции через ссылки, нет ссылки — в мусор

HEAD — текущая ссылка, tag — фиксированная ссылка, branch — движущаяся ссылка

checkout — перемещение на ветку или коммит, reset — перемещение с веткой на коммит

Видно то, на что есть ссылки, остальное — мусор

- `git tag` — вывести список тегов
- `git tag <tagname>` — создать тег
- `git branch` — вывести список локальных веток
- `git branch -av` — вывести список локальных и удаленных веток
- `git branch <branchname>` — создать ветку
- `git branch -d <branchname>` — удалить ветку
- `git checkout <commit>` — переместить HEAD на коммит (получится detached HEAD)
- `git checkout <branch>` — переместить HEAD на ветку
- `git checkout -b <new_branch>` — создать новую ветку и перейти на нее
- `git reset --hard <commit>` — переместить HEAD и текущую ветку на `<commit>`
- `git reflog show <ref>` — показать лог действий со ссылкой
- `git reflog = git reflog show HEAD` — показать лог действий с HEAD
- `git gc` — удалить ненужные файлы и оптимизировать локальный репозиторий

A1. Трехсторонний merge в три шага

Два состояния можно объединить через merge, mergetool и commit

Участвуют три стороны: current, incoming и base

- `git merge <commit>` — объединить текущую ветку с другой
- `git mergetool` — разрешить имеющиеся конфликты
- `git merge --abort` — отменить слияние

A2. rebase и cherry-pick, чтобы пересоздать неизменяемую историю

Нельзя переписать историю — можно создать новую

- `git commit --amend --no-edit` — заменить последний коммит ветки на отредактированный с дополнительными изменениями без изменения сообщения
- `git rebase <upstream>` — применить все коммиты от общего родителя до текущего к `<upstream>`
- `git rebase -i <upstream>` — применить заново все коммиты, указав действие с каждым коммитом
- `git rebase --continue` — продолжить rebase после разрешения конфликтов
- `git rebase --abort` — отменить rabase
- `git cherry-pick <commit>` — применить указанный коммит к HEAD

A3. stash, reset, revert для управления изменениями

Изменения можно временно припрятать

Можно получить разницу между любыми коммитами

Коммит можно отменить другим коммитом

- `git stash` — сохранить все модифицированные файлы в виде набора изменений
- `git stash pop` — восстановить последний сохраненный набор изменений и удалить его из списка
- `git stash list` — показать список сохраненных наборов изменений
- `git reset --hard <commit>` — переместить текущую ветку на `<commit>`, задать индекс и директорию согласно коммиту, устранив всю разницу
- `git reset --mixed <commit>` — переместить текущую ветку на `<commit>`, задать индекс согласно коммиту, оставить разницу между исходным и новым состоянием в директории
- `git reset --soft <commit>` — переместить текущую ветку на `<commit>`, не задавать индекс и директорию согласно коммиту, а оставить разницу между исходным и новым состоянием в индексе и директории
- `git reset --hard HEAD~1` — отменить последний коммит
- `git revert <commit>` — создать коммит, отменяющий изменения из коммита
- `git diff <from_commit> [<to_commit>]` — вывести разницу между двумя коммитами
- `git diff --name-status <from_commit> [<to_commit>]` — список измененных файлов
- `git difftool <from_commit> [<to_commit>]` - вывести разницу с помощью difftool из настроек

R1. Доступен fetch коммитов любого репозитория в любой момент

- `git remote -v` — вывести список удаленных репозиториев с их адресами
- `git remote add <name> <url>` — добавить удаленный репозиторий с URL и дать ему указанное имя
- `git fetch = git fetch origin` — получить содержимое основного удаленного репозитория
- `git fetch --all` — получить содержимое всех удаленных репозиториев из списка

R2. Удаленное изменение — это push

- `git push <remote> <local_branch>:<remote_branch>` — добавить изменения из локальной ветки `<local_branch>` и переместить ветку `<remote_branch>` удаленного репозитория
- `git push = git push origin HEAD` — добавить изменения из текущей локальной ветки и переместить соответствующую ветку удаленного репозитория
- `git push -f` — выполнить `push`, даже если удаленная ветка уже не является предком
- `git push --force-with-lease` — выполнить `push`, если является предком или удаленная ветка не сдвигалась (использовать вместо предыдущей команды)
- `git push <remote> -d <branch>` — удалить ветку в удаленном репозитории
- `git push <remote> tag <tag>` — отправить тег в удаленный репозиторий
- `git push --mirror` — выполнить агрессивный `push` для всех тегов, веток и HEAD, подходит для создания удаленной копии локального репозитория

R3. Явное сопоставление локальных веток с upstream

- `git branch -vv` — вывести список локальных веток с указанием привязанных к ним upstream-веток
- `git branch -u <upstream> [<branchname>]` — задать upstream-ветку для указанной или текущей ветки
- `git push -u origin HEAD` — создать удаленную ветку, соответствующую локальной и установить между ними upstream-связь, затем добавить изменения из локальной ветки в удаленный репозиторий
- `git checkout <remote_branchname>` — создать локальную ветку, соответствующую удаленной и установить между ними upstream-связь, затем переместить HEAD на нее
- `git pull = git pull origin` — получить содержимое основного удаленного репозитория и влить изменения из удаленной ветки в соответствующую локальную ветку
- `git pull --ff-only` — получить содержимое, а затем влить, если возможен fast-forward merge
- `git pull --rebase` — получить содержимое и выполнить rebase локальной ветки на удаленную ветку
- `git pull --rebase --autostash` — сохранить локальные изменения, получить содержимое, выполнить rebase локальной ветки на удаленную ветку, применить сохраненные изменения
- `git config --global push.default simple` — задать simple-режим действий с upstream-связями при push. Это режим по умолчанию в Git 2.0 и выше