Шпаргалка по Git

http://brainwashing.pro/rails

Репо

Создать репо в текущей папке
```
git init
```

Или
```
git clone
  [(git|ssh|http{s})://]
  host.org/project[.git]
```

Добавить внешний репо
```
git remote add
  <remote_repo> <branch>
```

Обновить текущую ветку
от ветки с внешнего репо
через fast forward или merge
```
git pull
  <remote_repo> <branch>
```

Обновить текущую ветку от ветки
с внешнего репо через ребейс
```
git pull --rebase
  <remote_repo> <branch>
```

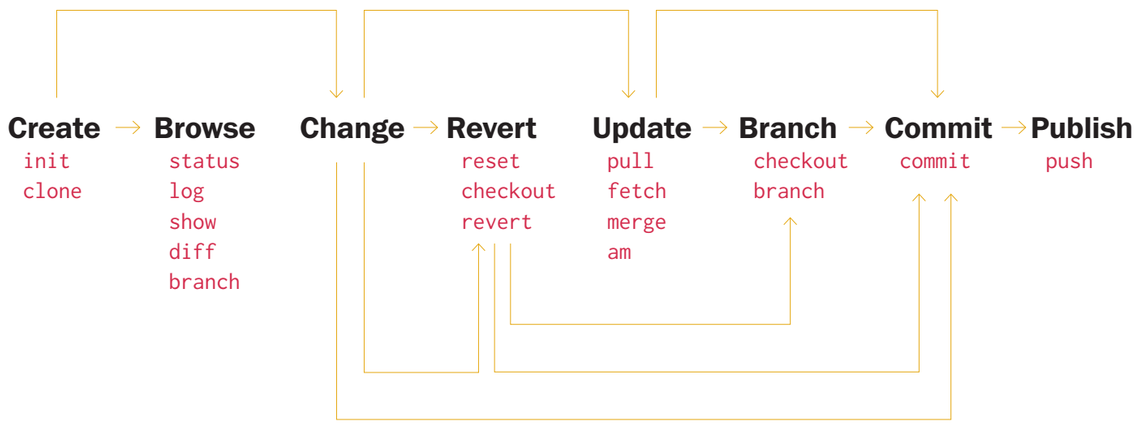
Получить изменения
из внешнего репо
```
git fetch [remote_repo]
```

Отправить изменения локальной
ветки в ветку внешнего репо
```
git push
  <remote_repo> <branch>
```

Удалить ветку из внешнего репо
```
git push
  <remote_repo> :<branch>
```

⚠ Переписать ветку
на внешнем репо
```
git push -f
  <remote_repo> <branch>
```

Workflow



Переписать

Отменить последний коммит
и изменения из него в индексе
```
git reset --soft HEAD~1
```

Новый коммит, отменяющий
изменения из коммита SHA1
```
git revert SHA1
```

Переписать последний коммит
с учетом текущего индекса
и нового сообщения
```
git commit --amend
```

Изменить историю, порядок, коли-
чество коммитов в выбранном
промежутке от текущего коммита
до SHA1
```
git rebase -i SHA1
```

Применить коммиты new_head
к new_base без old_head
```
git rebase --onto <new_base>
  <old_base> <new_head>
```

Ветки

Создать новую ветку, указывающую
на текущий коммит
```
git branch <new_branch>
```

Список всех веток
```
git branch -a
```

Создать и переключиться на новую
ветку (если она не существовала)
```
git checkout -b <new_branch>
```

Создать и переключиться
на новую ветку (переписать,
если существовала)
```
git checkout -B <new_branch>
```

Перейти на другую ветку
```
git checkout <branch>
```

Смержить текущую ветку с другой
```
git merge <branch>
```

Удалить ветку
```
git branch -d <branch>
```

Сохранить

Добавить файл в индекс
```
git add <file>
```

Интерактивный режим
добавления файлов в индекс
```
git add -i
```

Создать патч из кусков изменений
```
git add -p
```

Закоммитить все изменения
```
git commit -am
  "All my changes"
```

Сохранить в другую ветку
```
git stash
git checkout <other_branch>
git stash apply
git commit -am "My changes
  for other branch"
```

Сохранить в stash
```
git stash save
  "My temp changes"
```

Состояния

Убрать все изменения
из индекса в unstaged
```
git reset HEAD
```

Отменить последний коммит,
сохранив изменения в индексе
```
git reset --soft HEAD~1
```

Отменить последний коммит
```
git reset --hard HEAD~1
```

Отложить все изменения в stash
```
git stash
```

Вытащить состояние файла
из определенной ветки
```
git checkout <branch> <file>
```

Поиск пропаж

Показать локальную работу
за последние пару месяцев
```
git reflog
```

Аналогично
```
git log -g
```

Поиск «подвисших объектов»
```
git fsck
```

Просмотр всех сохраненных
изменений в stash
```
git stash list
```

Поиск something по всем объектам
```
git log -S<something> --all
```

Поиск regexp по всем объектам
```
git log -G<regexp> --all
```

Показать

Измененные файлы
```
git status
```

Изменения в файлах
относительно HEAD
```
git diff
```

Разницу двух состояний
```
git diff <SHA1> [SHA2]
```

Историю текущей ветки
```
git log
```

Историю файла/директории
в текущей ветке
```
git log -p [file/dir]
```

Историю, начиная с определенного
SHA1 коммита
```
git log --stat <SHA1>
```

Кто последний менял каждую
строку в файле
```
git blame [file]
```

Коммит SHA1
```
git show <SHA1>
```

Изменения файла из коммита SHA1
```
git show <SHA1>:<file>
```

Список всех веток
```
git branch -a
```

Список всех веток
[не] вмерженных в текущую
```
git branch --[no-]merged
```