

Введение в git

Артем Оганджян

ЛОЛ-2021

10 августа 2021 г.

Оглавление

- 1 Мотивация
 - История изменений
 - Резервное копирование
 - Работа в команде
- 2 Принцип работы git
- 3 Основные команды
- 4 Советы
- 5 GitHub

Потеря кода

Потеря кода

Напишу-ка я тетрис.

```
$ vim tetris.cpp  
$ g++ tetris.cpp  
$ ./a.out
```

Потеря кода

Напишу-ка я тетрис.

```
$ vim tetris.cpp  
$ g++ tetris.cpp  
$ ./a.out
```

Добавлю меню...

```
$ vim tetris.cpp  
$ g++ tetris.cpp  
$ ./a.out
```

Потеря кода

Напишу-ка я тетрис.

```
$ vim tetris.cpp  
$ g++ tetris.cpp  
$ ./a.out
```

Добавлю меню...

```
$ vim tetris.cpp  
$ g++ tetris.cpp  
$ ./a.out
```

Работает!

Потеря кода

Напишу-ка я тетрис.

```
$ vim tetris.cpp  
$ g++ tetris.cpp  
$ ./a.out
```

Добавлю меню...

```
$ vim tetris.cpp  
$ g++ tetris.cpp  
$ ./a.out
```

Ой, что-то сломалось. Зря я код не сохранил...

Архивы

```
$ ls versions  
v0.1.tar  v0.2.tar
```


Архивы

```
$ ls versions
```

```
v0.1.tar  v0.2.tar  v0.3.tar  v0.4.tar  v0.5.tar  
v0.6.tar  v0.7.tar  v0.8.tar  v0.9.tar
```

Архивы

```
$ ls versions
```

```
v0.1.tar  v0.2.tar  v0.3.tar  v0.4.tar  v0.5.tar  
v0.6.tar  v0.7.tar  v0.8.tar  v0.9.tar
```

+ Просто

Архивы

```
$ ls versions
```

```
v0.1.tar  v0.2.tar  v0.3.tar  v0.4.tar  v0.5.tar  
v0.6.tar  v0.7.tar  v0.8.tar  v0.9.tar
```

- + Просто
- Неудобно

Архивы

```
$ ls versions
```

```
v0.1.tar  v0.2.tar  v0.3.tar  v0.4.tar  v0.5.tar  
v0.6.tar  v0.7.tar  v0.8.tar  v0.9.tar
```

- + Просто
- Неудобно
- Дублирование файлов

Несчастные случаи

Несчастные случаи

- Украли ноутбук

Несчастные случаи

- Украли ноутбук
- Утопили ноутбук

Несчастные случаи

- Украли ноутбук
- Утопили ноутбук
- Умер жёсткий диск

Несчастные случаи

- Украли ноутбук
- Утопили ноутбук
- Умер жёсткий диск
- Бекапы?

Несчастные случаи

- Украли ноутбук
- Утопили ноутбук
- Умер жёсткий диск
- Бекапы?
 - На флешке
 - В облаке

Несчастные случаи

- Украли ноутбук
- Утопили ноутбук
- Умер жёсткий диск
- Бекапы?
 - На флешке
 - В облаке
 - Неудобно, ещё больше архивов

Работа в команде

- Алиса:

Работа в команде

- Алиса:

v0.1.tar v0.2.tar

Работа в команде

- Алиса:
v0.1.tar v0.2.tar
- Приходит Боб:

Работа в команде

- Алиса:
v0.1.tar v0.2.tar
- Приходит Боб:
v0.3.tar v0.4.tar

Работа в команде

- Алиса:
v0.1.tar v0.2.tar
- Приходит Боб:
v0.3.tar v0.4.tar
- Тем временем Алиса:
v0.1.tar v0.2.tar v0.3.tar v0.4.tar v0.5.tar

Работа в команде

- Алиса:
v0.1.tar v0.2.tar
- Приходит Боб:
v0.3.tar v0.4.tar
- Тем временем Алиса:
v0.1.tar v0.2.tar v0.3.tar v0.4.tar v0.5.tar
- Приходит Ева.

Работа в команде

- Алиса:
v0.1.tar v0.2.tar
- Приходит Боб:
v0.3.tar v0.4.tar
- Тем временем Алиса:
v0.1.tar v0.2.tar v0.3.tar v0.4.tar v0.5.tar
- Приходит Ева. Где сейчас последняя версия?

Работа в команде

- Алиса:
v0.1.tar v0.2.tar
- Приходит Боб:
v0.3.tar v0.4.tar
- Тем временем Алиса:
v0.1.tar v0.2.tar v0.3.tar v0.4.tar v0.5.tar
- Приходит Ева. Где сейчас последняя версия?
- Надо совместить наработки.

Работа в команде

- Алиса:
v0.1.tar v0.2.tar
- Приходит Боб:
v0.3.tar v0.4.tar
- Тем временем Алиса:
v0.1.tar v0.2.tar v0.3.tar v0.4.tar v0.5.tar
- Приходит Ева. Где сейчас последняя версия?
- Надо совместить наработки. Много пересекающихся изменений.

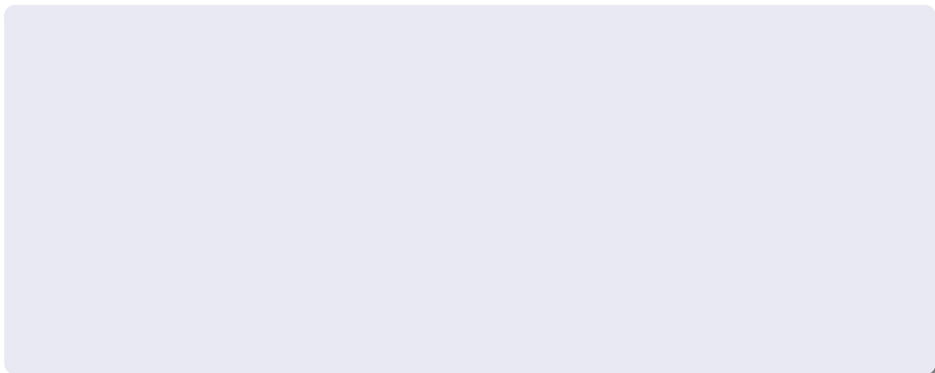
Работа в команде

- Алиса:
v0.1.tar v0.2.tar
- Приходит Боб:
v0.3.tar v0.4.tar
- Тем временем Алиса:
v0.1.tar v0.2.tar v0.3.tar v0.4.tar v0.5.tar
- Приходит Ева. Где сейчас последняя версия?
- Надо совместить наработки. Много пересекающихся изменений.
- Нет линейной истории версий.

Оглавление

- 1 Мотивация
- 2 Принцип работы git
 - Структура
 - Децентрализация
- 3 Основные команды
- 4 Советы
- 5 GitHub

Структура хранилища



Структура хранилища

```
* b054512 Add line  
|  
* 0d0461e Add square  
* f4d5af4 Initial commit
```


Структура хранилища

Список версий?

```
* b054512 Add line  
|  
* 0d0461e Add square  
* f4d5af4 Initial commit
```

Структура хранилища

Список версий?

```
* f2af708          Add menu
* efbd2a5 Add score
* | 6ef17c4 Add Z-shapes
* | 6347d69 Add L-shapes
* | b054512 Add line
|/
* 0d0461e Add square
* f4d5af4 Initial commit
```

Структура хранилища

Дерево версий?

```
* f2af708          Add menu
* efbd2a5 Add score
* | 6ef17c4 Add Z-shapes
* | 6347d69 Add L-shapes
* | b054512 Add line
|/
* 0d0461e Add square
* f4d5af4 Initial commit
```

Структура хранилища

Дерево версий?

```
* 4d21842 Merge branch 'bob' into alice
| \
| * f2af708 Add menu
| * efbd2a5 Add score
* | 6ef17c4 Add Z-shapes
* | 6347d69 Add L-shapes
* | b054512 Add line
| /
* 0d0461e Add square
* f4d5af4 Initial commit
```

Структура хранилища

Ациклический граф версий

```
* 4d21842      Merge branch 'bob' into alice
| \
| * f2af708      Add menu
| * efbd2a5 Add score
* | 6ef17c4 Add Z-shapes
* | 6347d69 Add L-shapes
* | b054512 Add line
| /
* 0d0461e Add square
* f4d5af4 Initial commit
```

Структура хранилища

Ациклический граф версий, указатели

```
* 4d21842 (alice) Merge branch 'bob' into alice
|\
| * f2af708 (HEAD -> bob) Add menu
| * efbd2a5 Add score
* | 6ef17c4 Add Z-shapes
* | 6347d69 Add L-shapes
* | b054512 Add line
|/
* 0d0461e Add square
* f4d5af4 Initial commit
```

Термины

Термины

- Система контроля версий, version control system, VSC.

Термины

- Система контроля версий, version control system, VSC.
- Репозиторий.

Термины

- Система контроля версий, version control system, VSC.
- Репозиторий.
- Версия — commit.

Термины

- Система контроля версий, version control system, VSC.
- Репозиторий.
- Версия — commit.
- Указатель — branch.

Термины

- Система контроля версий, version control system, VSC.
- Репозиторий.
- Версия — commit.
- Указатель — branch. Не обязательно выглядит как ветка.

Копии файлов

Копии файлов

- На каждый commit создаётся копия всех изменённых файлов.

Копии файлов

- На каждый commit создаётся копия всех изменённых файлов.
- Файлы, которые не менялись, не копируются.

Копии файлов

- На каждый commit создаётся копия всех изменённых файлов.
- Файлы, которые не менялись, не копируются.
- Можно было бы хранить только изменения (diff) файлов.

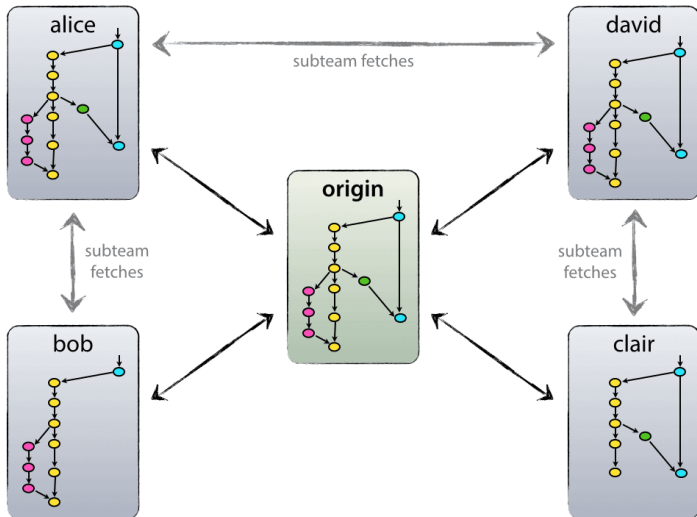
Копии файлов

- На каждый commit создаётся копия всех изменённых файлов.
- Файлы, которые не менялись, не копируются.
- Можно было бы хранить только изменения (diff) файлов. Но git этого не делает.

Копии файлов

- На каждый commit создаётся копия всех изменённых файлов.
- Файлы, которые не менялись, не копируются.
- Можно было бы хранить только изменения (diff) файлов. Но git этого не делает.
- Зато git (по умолчанию) сжимает файлы.

git — децентрализованная VCS



Преимущества

Преимущества

- Локальные изменения

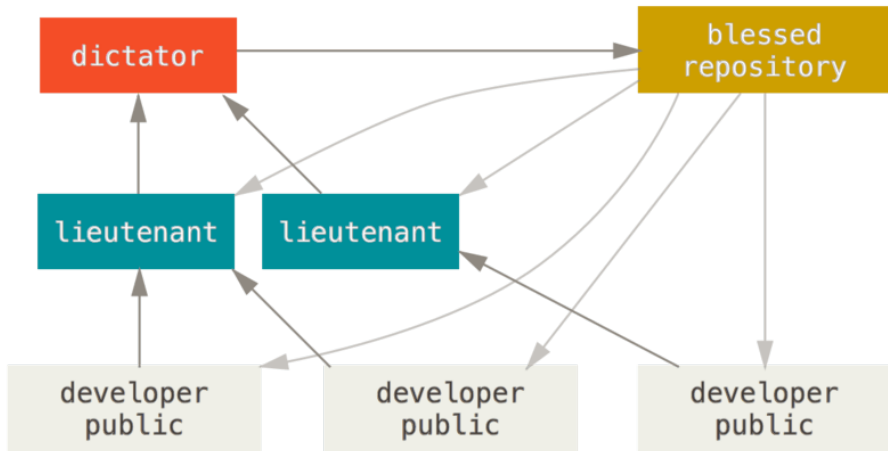
Преимущества

- Локальные изменения
- Отказоустойчивость

Преимущества

- Локальные изменения
- Отказоустойчивость
- Довольный Линус

Довольный Линус

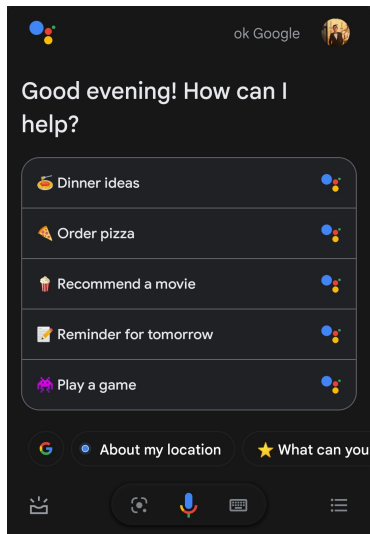


Оглавление

- 1 Мотивация
- 2 Принцип работы git
- 3 Основные команды**
 - Начало работы
 - Учёт изменений
 - Проверка
 - Внесение изменений
 - Удалённые репозитории
- 4 Советы
- 5 GitHub

Самый бесполезный раздел

Самый бесполезный раздел



Самый бесполезный раздел

```
$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
        [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        <command> [<args>]
```

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)

clone	Clone a repository into a new directory
init	Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)

add	Add file contents to the index
mv	Move or rename a file, a directory, or a symlink
restore	Restore working tree files
rm	Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)

bisect	Use binary search to find the commit that introduced a bug
diff	Show changes between commits, commit and working tree, etc
grep	Print lines matching a pattern
log	Show commit logs
show	Show various types of objects
status	Show the working tree status

Установка

```
$ sudo apt install git  
$ git config
```

Установка

```
$ sudo apt install git  
$ git config
```

Пример

```
$ git config --global user.name "Artem Ohanjanyan"  
$ git config --global user.email artemohanjanyan@gmail.com  
$ git config --global core.editor vim
```

Установка

```
$ sudo apt install git  
$ git config --help
```

Пример

```
$ git config --global user.name "Artem Ohanjanyan"  
$ git config --global user.email artemohanjanyan@gmail.com  
$ git config --global core.editor vim
```

Установка

```
$ sudo apt install git  
$ git config --help  
$ man git-config
```

Пример

```
$ git config --global user.name "Artem Ohanjanyan"  
$ git config --global user.email artemohanjanyan@gmail.com  
$ git config --global core.editor vim
```


Получение репозитория

Получение репозитория

- `$ git init`

Получение репозитория

- `$ git init`
- `$ git clone`

Получение репозитория

- `$ git init`
- `$ git clone`

Пример

```
git clone https://github.com/artemohanjanyan/git-tutorial.git
```

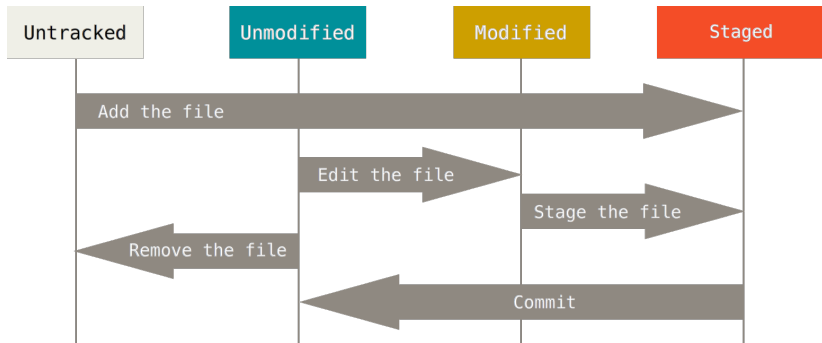
Получение репозитория

- `$ git init --help`
- `$ git clone --help`

Пример

```
git clone https://github.com/artemohanjanyan/git-tutorial.git
```

Жизненный цикл файла



Команды

- `$ git add`
- `$ git mv`
- `$ git reset`
- `$ git rm`

Команды

- `$ git add`
- `$ git mv`
- `$ git reset`
- `$ git rm`

Пример

```
$ git add slides.tex  
$ git mv dictator.png img/
```


Команды

- `$ git add --help`
- `$ git mv --help`
- `$ git reset --help`
- `$ git rm --help`

Пример

```
$ git add slides.tex  
$ git mv dictator.png img/
```

Команды

- `$ git status`
- `$ git log`

Команды

- `$ git status`
- `$ git log`

Пример

```
$ git status
On branch bob
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   main.cpp

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   main.cpp

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        scoreboard
```

Команды

- `$ git status --help`
- `$ git log --help`

Пример

```
$ git status
On branch bob
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   main.cpp

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   main.cpp

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        scoreboard
```

Команды

Пример

```
$ git log --graph --decorate --oneline
* 4d21842 (alice) Merge branch 'bob' into alice
| \
| * f2af708 (HEAD -> bob) Add menu
| * efbd2a5 Add score
* | 6ef17c4 Add Z-shapes
* | 6347d69 Add L-shapes
* | b054512 Add line
| /
* 0d0461e Add square
* f4d5af4 Initial commit
```

Команды

- `$ git commit`

Команды

- `$ git commit`
- `$ git checkout`

Команды

- `$ git commit`
- `$ git checkout`
- `$ git diff`

Команды

- `$ git commit`
- `$ git checkout`
- `$ git diff`
- `$ git merge`

Команды

- \$ git commit
- \$ git checkout
- \$ git diff
- \$ git merge
- \$ git branch

Команды

- `$ git commit`
- `$ git checkout`
- `$ git diff`
- `$ git merge`
- `$ git branch`
- `$ git tag`

Команды

- `$ git commit --help`
- `$ git checkout --help`
- `$ git diff --help`
- `$ git merge --help`
- `$ git branch --help`
- `$ git tag --help`

git commit

Commit message

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch bob
#
# Changes to be committed:
#       modified:   main.cpp
#
# Changes not staged for commit:
#       modified:   main.cpp
#
# Untracked files:
#       scoreboard
#
```

git checkout

git checkout

- `$ git checkout master`

git checkout

- `$ git checkout master`
- `$ git checkout 6be2821`

git checkout

- `$ git checkout master`
- `$ git checkout 6be2821`
- `$ git checkout -b feature`

git merge

git merge

- `$ git merge feature`

git merge

- `$ git merge feature`
- merge-конфликты

git merge

- `$ git merge feature`
- merge-конфликты

Пример

```
<<<<<< HEAD
```

```
Changes on master
```

```
=====
```

```
Changes on feature
```

```
>>>>>> feature
```

git merge

- `$ git merge feature`
- merge-конфликты
- fast-forward

Пример

```
<<<<<< HEAD
```

```
Changes on master
```

```
=====
```

```
Changes on feature
```

```
>>>>>> feature
```

Удалённые ветки

```
* 3d6c452 (upstream/master) build version update 3.0.1
* 9d35a7a Changelog updated for 3.0.1
* 425bd86 SolidMap equals improvement
*   f96d7e1 Merge pull request #17 from artemohanjanyan/equals
| \
| * 670d71e (origin/equals) Add more tests
| * ba53c22 Improve equals implementation
| /
*   b7bd7f3 Merge pull request #15 from artemohanjanyan/zipWith
| \
| * 7c55777 Generalize zipWith argument
| * 238c358 Add zipWith operator
* |   d8c1109 Merge pull request #14 from artemohanjanyan/patch-1
| \ \
| | /
| / |
| * 72b251f Update flatMap javadoc
| /
* 652acad (HEAD -> master, origin/master, origin/HEAD) 3.0.0
```

Удалённые ветки

- `$ git checkout -b equals origin/equals`

Удалённые ветки

- `$ git checkout -b equals origin/equals`
- `$ git checkout --track origin/equals`

Удалённые ветки

- `$ git checkout -b equals origin/equals`
- `$ git checkout --track origin/equals`
- `$ git checkout equals`

Удалённые ветки

```
* 3d6c452 (upstream/master) build version update 3.0.1
* 9d35a7a Changelog updated for 3.0.1
* 425bd86 SolidMap equals improvement
*   f96d7e1 Merge pull request #17 from artemohanjanyan/equals
|\
| * 670d71e (HEAD -> equals, origin/equals) Add more tests
| * ba53c22 Improve equals implementation
|/
*   b7bd7f3 Merge pull request #15 from artemohanjanyan/zipWith
|\
| * 7c55777 Generalize zipWith argument
| * 238c358 Add zipWith operator
* |   d8c1109 Merge pull request #14 from artemohanjanyan/patch-1
|\ \
| | /
| /|
| * 72b251f Update flatMap javadoc
|/
* 652acad (origin/master, origin/HEAD, master) 3.0.0
```

Удалённые репозитории

- `$ git remote`

Пример

```
$ git remote add origin https://github.com/artemohanjanyan/git-tutorial.git
```

Удалённые репозитории

- `$ git remote`
- `$ git fetch`

Пример

```
$ git remote add origin https://github.com/artemohanjanyan/git-tutorial.git
```

Удалённые репозитории

- `$ git remote`
- `$ git fetch`
- `$ git push`

Пример

```
$ git remote add origin https://github.com/artemohanjanyan/git-tutorial.git
```

Удалённые репозитории

- `$ git remote`
- `$ git fetch`
- `$ git push`
- `$ git pull`

Пример

```
$ git remote add origin https://github.com/artemohanjanyan/git-tutorial.git
```

Удалённые репозитории

- `$ git remote --help`
- `$ git fetch --help`
- `$ git push --help`
- `$ git pull --help`

Пример

```
$ git remote add origin https://github.com/artemohanjanyan/git-tutorial.git
```


Оглавление

- 1 Мотивация
- 2 Принцип работы git
- 3 Основные команды
- 4 Советы**
 - Коммиты
 - Всякое
- 5 GitHub

Коммиты

КОММИТЫ

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Частота коммитов

- Маленькие коммиты с конкретными изменениями

Частота коммитов

- Маленькие коммиты с конкретными изменениями
 - a lot of code

Частота коммитов

- Маленькие коммиты с конкретными изменениями
 - a lot of code
 - something

Частота коммитов

- Маленькие коммиты с конкретными изменениями
 - a lot of code
 - something
 - ± some progress on issue #5

Частота коммитов

- Маленькие коммиты с конкретными изменениями
 - a lot of code
 - something
 - ± some progress on issue #5
 - + Fix issue #5

Сообщения коммитов

Сообщения коммитов

- 1 Отделяем заголовок от тела пустой строкой.
- 2 Заголовок не длиннее 50 символов,
- 3 с большой буквы,
- 4 без точки в конце,
- 5 в повелительном наклонении.
- 6 Тело укладываем в 72 символа по ширине,
- 7 описываем в нём «что», а не «как».

Пример

Пример

- * [276da50](#) Compile pdf
- * [611092f](#) Add `\mathrm` to 'ord' keyword
- * [0ba96ea](#) Use `\coloneqq` instead of `:=`
- * [6be881a](#) Use `\mathit` for keywords such as 'Consis', 'Proof' etc
- * [8cc97bb](#) Use function commands instead of plain names for better typesetting
- * [85a128e](#) Use `\mid` instead of `|` for better spacing
- * [1d85a0c](#) Use `\with` instead of `\&` for better spacing

Пример

```
* 5eab8ed Merge pull request #6 from artemohanjanyan/master
|\
| * 276da50 Compile pdf
| * 611092f Add \mathrm to 'ord' keyword
| * 0ba96ea Use \coloneqq instead of :=
| * 6be881a Use \mathit for keywords such as 'Consis', 'Proof' etc
| * 8cc97bb Use function commands instead of plain names for better typesetting
| * 85a128e Use \mid instead of | for better spacing
| * 1d85a0c Use \with instead of \& for better spacing
```

.gitignore

Перечисляем файлы, которые git должен игнорировать.

Ссылки на коммиты

Ссылки на коммиты

- HEAD, master

Ссылки на коммиты

- HEAD, master
- HEAD~

Ссылки на коммиты

- HEAD, master
- HEAD~
- HEAD~~~

Ссылки на коммиты

- HEAD, master
- HEAD~
- HEAD~~~
- HEAD~10

Ссылки на коммиты

- HEAD, master
- HEAD~
- HEAD~~~
- HEAD~10
- HEAD^^

Ссылки на коммиты

- HEAD, master
- HEAD~
- HEAD~~~
- HEAD~10
- HEAD^^
- `$ man git-rev-parse`

SSH ключи

- Чтобы не писать пароль каждый раз.

SSH ключи

- Чтобы не писать пароль каждый раз.
- Создаём.

SSH ключи

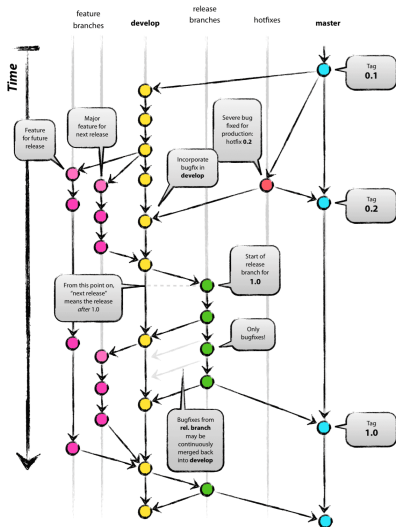
- Чтобы не писать пароль каждый раз.
- Создаём.
- Открытый ключ отправляем на сервер.

SSH ключи

- Чтобы не писать пароль каждый раз.
- Создаём.
- Открытый ключ отправляем на сервер.
- Закрытым ключом доказываем, что мы это мы.

Git flow

Git flow



Danger zone

- `$ git rebase`
- `$ git cherry-pick`

Danger zone

- `$ git rebase`
- `$ git cherry-pick`
- Опасно

Danger zone

- `$ git rebase`
- `$ git cherry-pick`
- Опасно
- `$ git reflog`
- `$ git reset`

Альтернативы

- Subversion — централизованная, старая.
- Mercurial — как git, но написана на Python.

Оглавление

- 1 Мотивация
- 2 Принцип работы git
- 3 Основные команды
- 4 Советы
- 5 GitHub**

Что такое GitHub?

Что такое GitHub?

- Бесплатные открытые репозитории

Что такое GitHub?

- Бесплатные открытые репозитории
- Pull request-ы

Что такое GitHub?

- Бесплатные открытые репозитории
- Pull request-ы
- fork-и

Что такое GitHub?

- Бесплатные открытые репозитории
- Pull request-ы
- fork-и
- <https://github.com/artemohanjanyan/git-tutorial>

Что такое GitHub?

- Бесплатные открытые репозитории
- Pull request-ы
- fork-и
- <https://github.com/artemohanjanyan/git-tutorial#lol-2021>

Альтернативы

Альтернативы

- Bitbucket

Альтернативы

- Bitbucket
- GitLab

Альтернативы

- Bitbucket
- GitLab
- Gitea

Ссылки

<https://www.google.com/>

<https://git-scm.com/book>

<https://github.com/artemohanjanyan/git-tutorial>

<https://www.wix.engineering/post/virtual-monorepo-for-bazel>

Лекцию слушали

Вопросы?

Вопросы?

Спасибо за внимание!