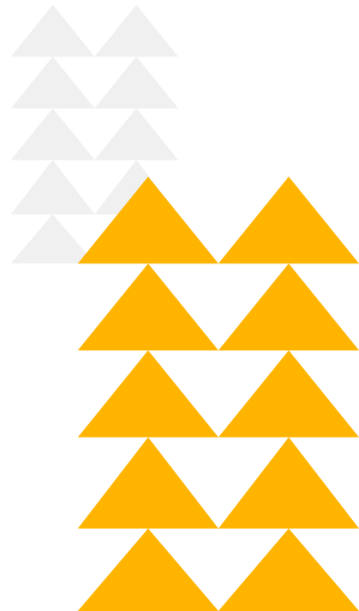
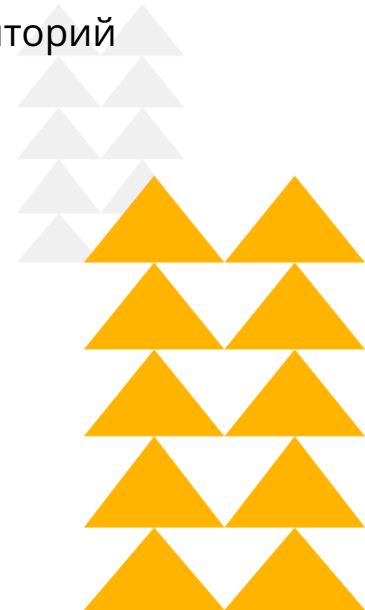
О чём поговорим?

- Зачем нужны системы контроля версий
- Как работает git
- Полезные возможности git
- Подходы к разработке



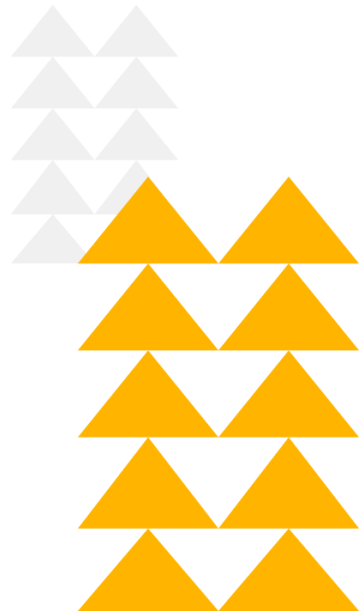
Основные термины

- Репозиторий - виртуальное хранилище для проекта
- Рабочая директория - файлы над которыми сейчас работаете
- Коммит - снимок состояния файлов вместе дополнительной информацией об изменениях (сообщение, автор, время)
- Индекс - снимок состояния рабочей директории, который попадет в репозиторий при коммите



Основные команды

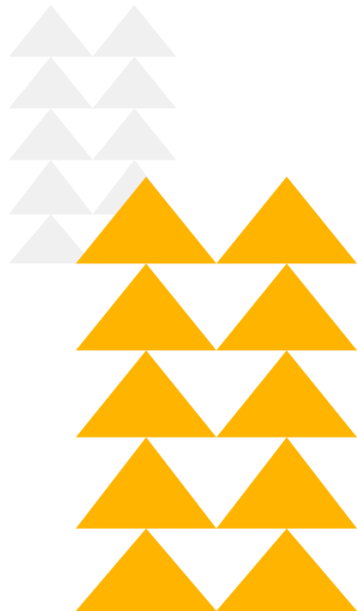
- git config
- git init / clone
- git add / commit / checkout / push / pull
- git status / log / diff
- git blame / stash / rebase
- .gitignore



Настройка пользователя

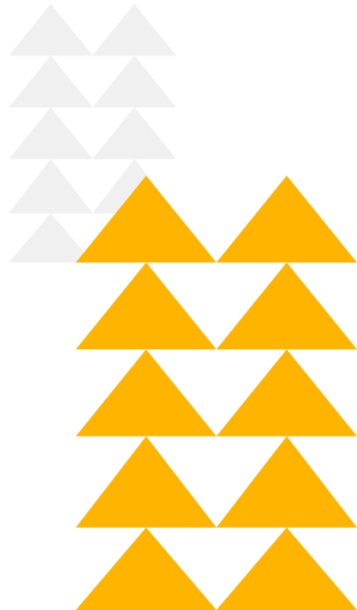
Настройте себе публичный email не аффилированный с вашей компанией. В дальнейшем вы можете менять свой аккаунт для каждого проекта отдельно

```
01. $ git config --global user.email "vital.kudzelka@gmail.com"
02.
03. $ cd leverx_project
04. $ git config user.email "vital.kudzelka@leverx.com"
```



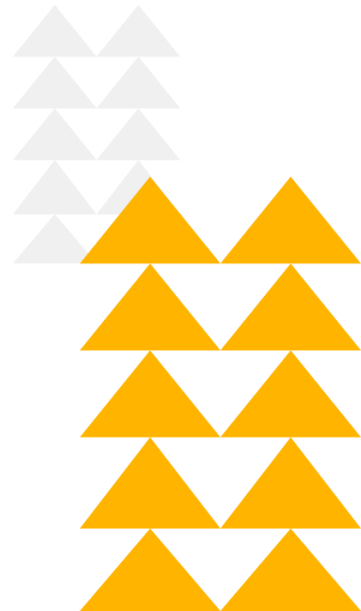
Как писать хорошие сообщения

- Тема не длиннее 50 символов / пишем с большой буквы / без точки в конце
- Тему сообщения отделяем от тела пустой строкой
- Используем императивный тон
- Тело ограничиваем 72 символамии
- Описываем что сделано и почему, а не как.



git show 432081

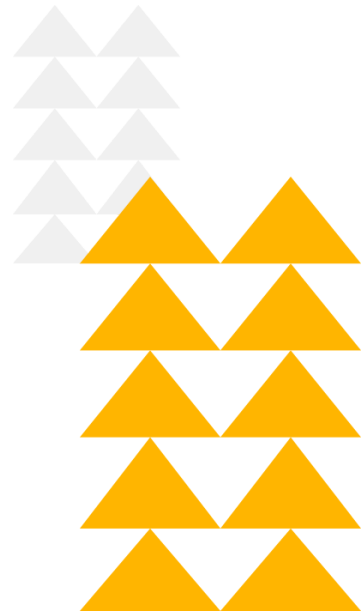
```
01. diff-merges: introduce --diff-merges=on
02.
03. Introduce the notion of default diff format for merges, and the option
04. "on" to select it. The default format is "separate" and can't yet
05. be changed, so effectively "on" is just a synonym for "separate"
06. for now. Add corresponding test to t4013.
07.
08. This is in preparation for introducing log.diffMerges configuration
09. option that will let --diff-merges=on to be configured to any
10. supported format.
```



.gitmessage

```
$ cat .gitmessage
# 50-character subject line
#
# Remember blank line between title and body
#
# 72-character wrapped longer description.
# Explain *what* and *why* (not *how*).

$ git config --global commit.template ~/.gitmessage
```

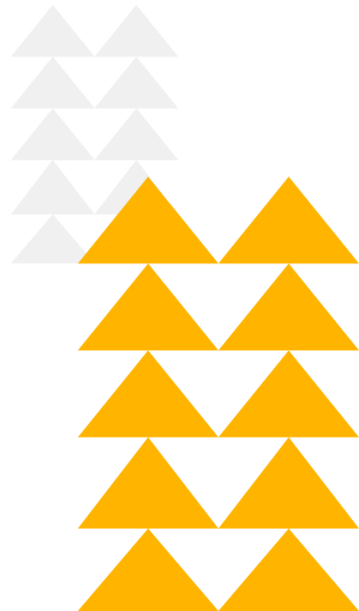


git commit

```
01. $ git log --oneline
02. d382a44b79 (tag: release-5.2.0) Merged in bugfix/OW-24788 (pull request #2150)
03. f59537a2ec OW-24889 Fix wf block name at credential usage
04. 3b1ab97158 OW-24467 Exclude password from email
05. daaaa7bc8a OW-22632 Fix recalculate report hash migration
06. c9e29c3a28 Merged in bugfix/OW-24331-co-customization-system-fields-are-empty (pull r
07. aa73a77080 Merged in bugfix/OW-24473-fix-counter-widget-w-contain-filter-by-co-field
08. 089e373d17 OW-24345 Adjust CO-related table generation to properly specify charset /
09. 0c4cfcd160 OW-24345 Migration to change CO tables charset to utf8
```

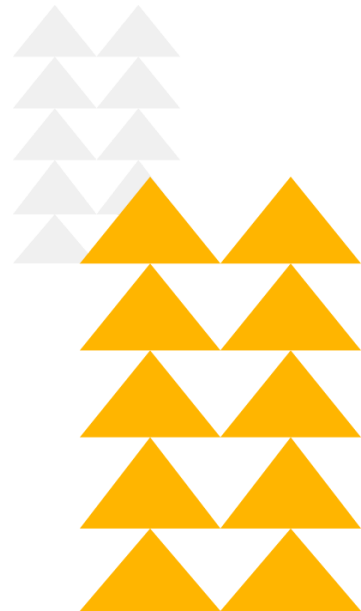
Исправление сообщения

```
01. $ git commit --amend -m "Awesome Feature A is done"
02. [test 5acfc8c] Awesome Feature A is done
03.  Date: Sun May 2 09:55:48 2021 +0300
04.  1 file changed, 2 insertions(+)
```



git stash

```
01. $ git stash
02. $ git stash save -m <message>
03. $ git stash apply stash@{1}
```

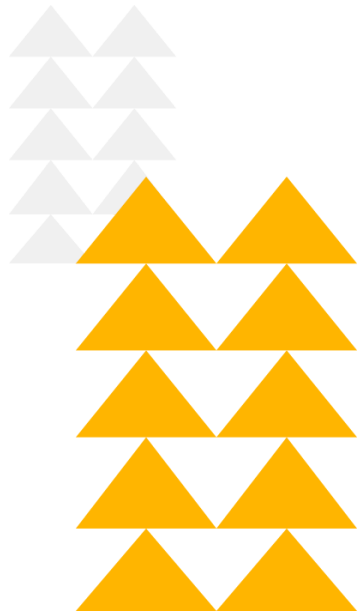


git rebase -i

Помогает причёсывать историю и аккуратно оформить коммиты перед тем как их опубликовать

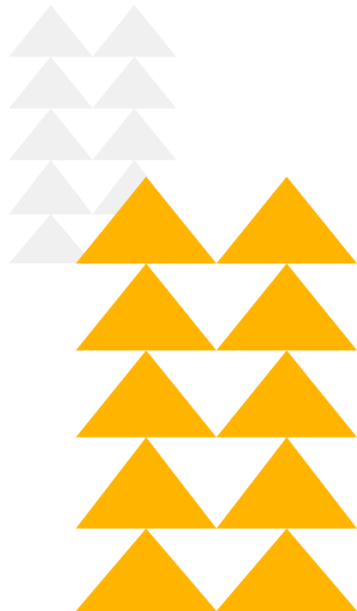
```
$ git log --oneline
4010d5c Feature A fix one more time
6cdd9e7 Feature A fix
19b3129 Feature A is done
1815138 Feature A is almost done
037192d Feature A work in progress
b735fbb Previous commit

$ git rebase -i b735fbb
```



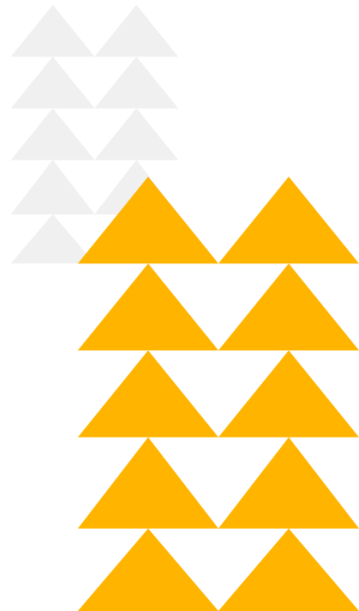
git rebase -i

```
01. > vim
02. fixup Feature A fix one more time
03. fixup Feature A fix
04. fixup Feature A is done
05. fixup Feature A is almost done
06. reword Feature A work in progress
07. b735fbb Previous commit
```



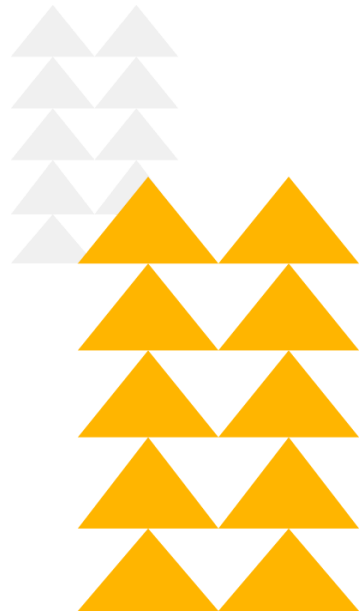
git rebase -i

```
01. $ git log --oneline
02. a37067f Feature A is done
03. b735fbb Previous commit
```



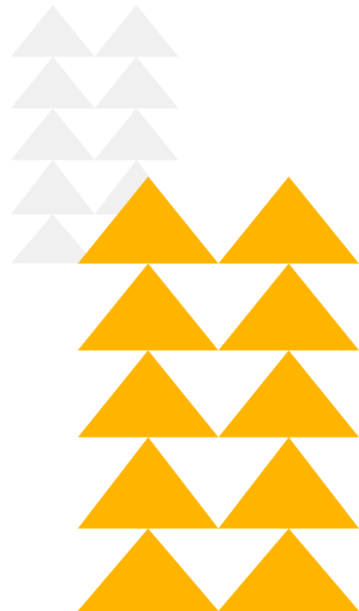
Переключение на предыдущую ветку

```
01. $ git checkout -
02. $ git checkout @{-n}
```



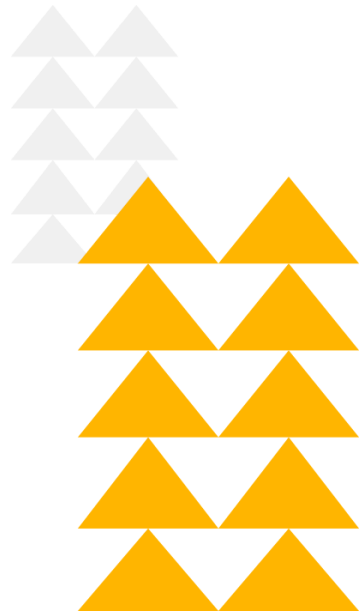
Указание версии

```
01. HEAD   <-- текущая ветка / коммит
02. HEAD^  <-- родитель
03. HEAD^^ <-- родитель родителя
04. HEAD~1 <-- 1ой родитель
05. HEAD~2 <-- 2ой родитель
06.
07. $ git help rev-parse
```



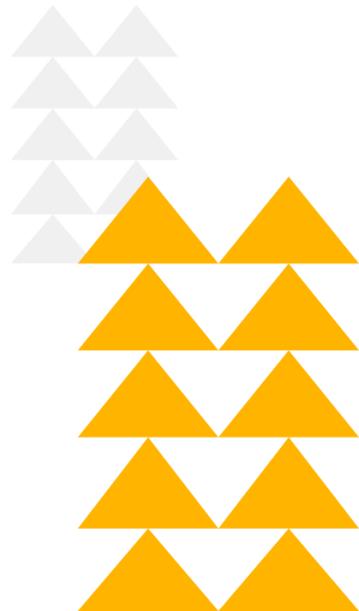
Указание версии

```
01.        G     H  I    J     J = F^2   = B^3^2    = A^^3^2
02.         \   /   \   /       I = F^    = B^3^     = A^^3^
03.          \ /     \ /        G = A^^^ = A^1^1^1 = A~3
04.           D   E   F
05.            \  |  / \        E = B^2   = A^^2
06.             \ | /   \       D = A^^   = A^1^1    = A~2
07.              \|/    |       F = B^3   = A^^3
08.               B     C
09.                \   /        B = A^    = A^1      = A~1
10.                 \ /         C = A^2   = A^2
11.                  A           A =       = A^Ø
```



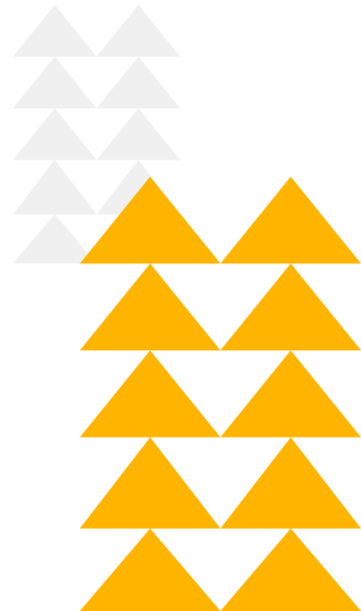
Последние ветки в которых были изменения

```
01. [alias]
02.     workon = !git for-each-ref \
03.         --count=10 \
04.         --sort=-committerdate \
05.         --format='%(refname:short)' \
06.         refs/heads/
```



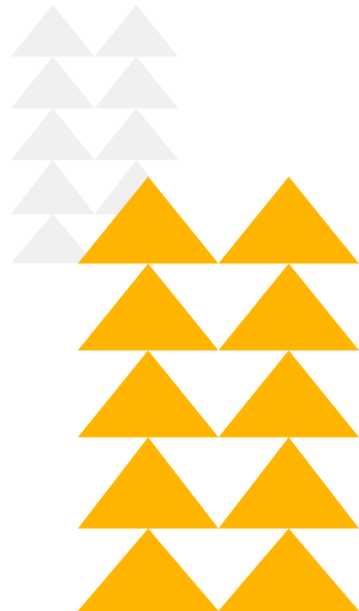
Отмена последнего коммита

```
[alias]
    undo = git reset --soft HEAD~1
```



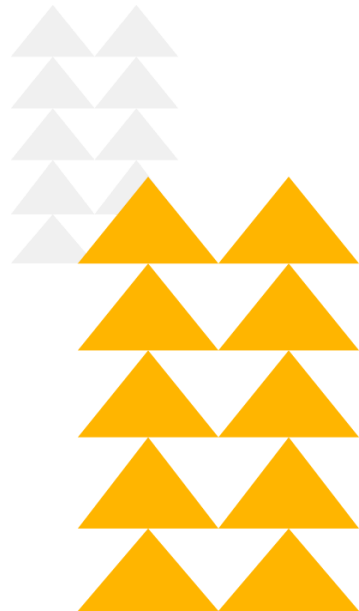
Смотрим за удаленной веткой

```
01. [alias]
02.     track = "!f() { git branch --set-upstream-to=origin/$1 $1; }; f"
```



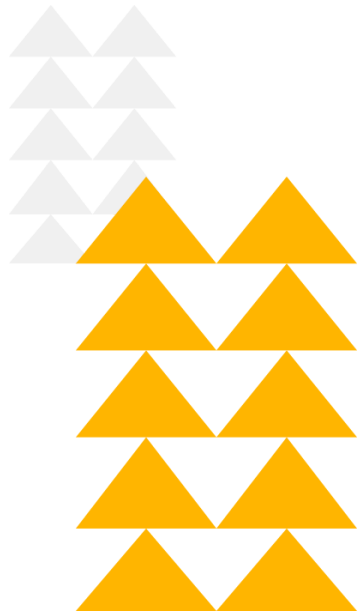
Коммиты сделанные мной

```
01. [alias]
02.     recap = !git log --all --oneline --no-merges \
03.         --author=${1-$(git config user.email)}
04.
05.     today = !git log --all --since=00:00:00 --oneline --no-merges \
06.         --author=${1-$(git config user.email)}
```



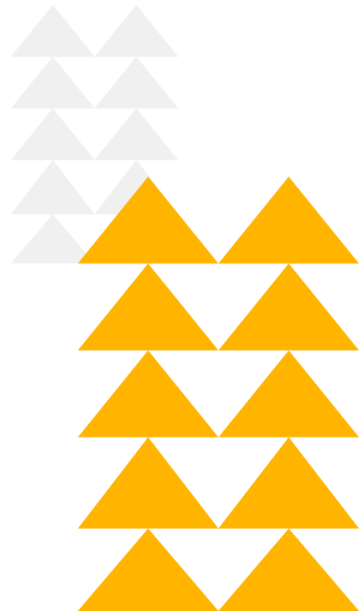
Удаление веток которые были смержены

```
$ git fetch --prune
```



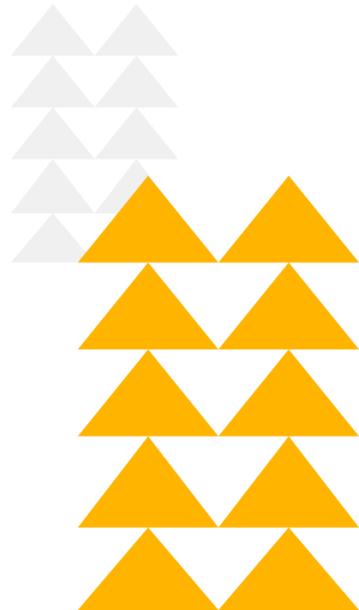
Обновляем файл из другой ветки

```
$ git checkout master ./path/to/file
```



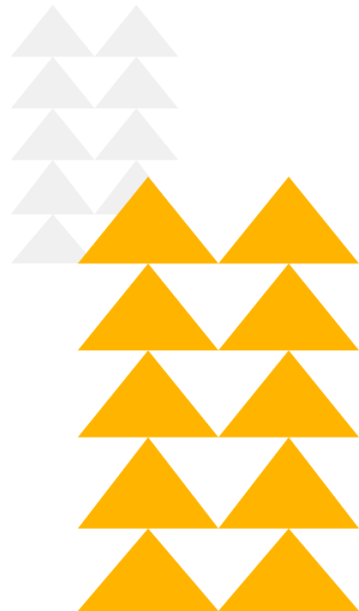
Take away: git

- git help



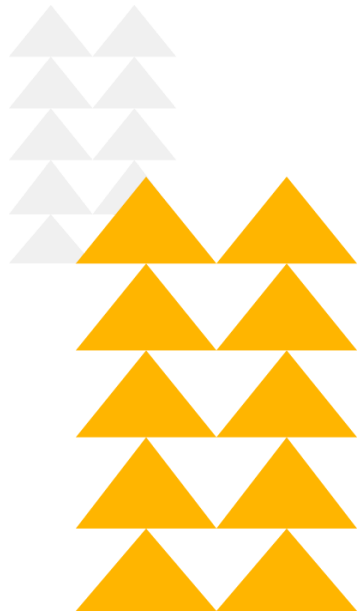
Git Workflows

- Centralized
- Feature branching*
- Gitflow*
- Forking

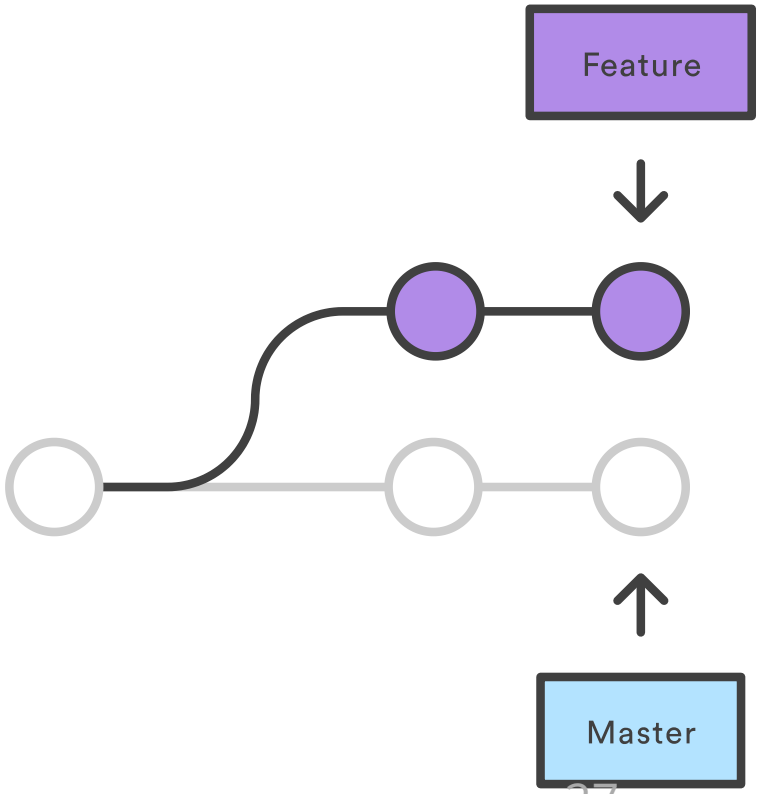


Feature Branch Workflow

- Вся разработка должна проходить в отдельных независимых ветках
- Такое разделение позволяет нескольким разработчикам работать над разными фичами
- Готовый код проходит код ревью перед мержем в master
- Основная ветка содержит рабочий код

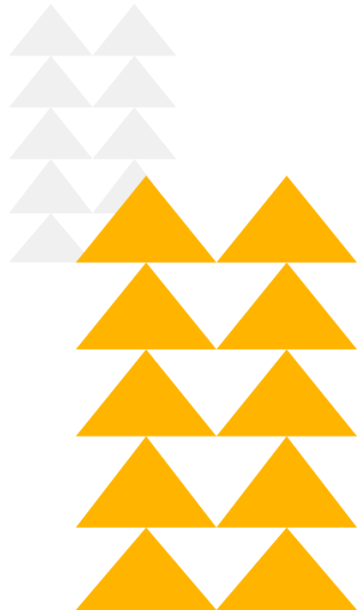


Feature Branch Workflow

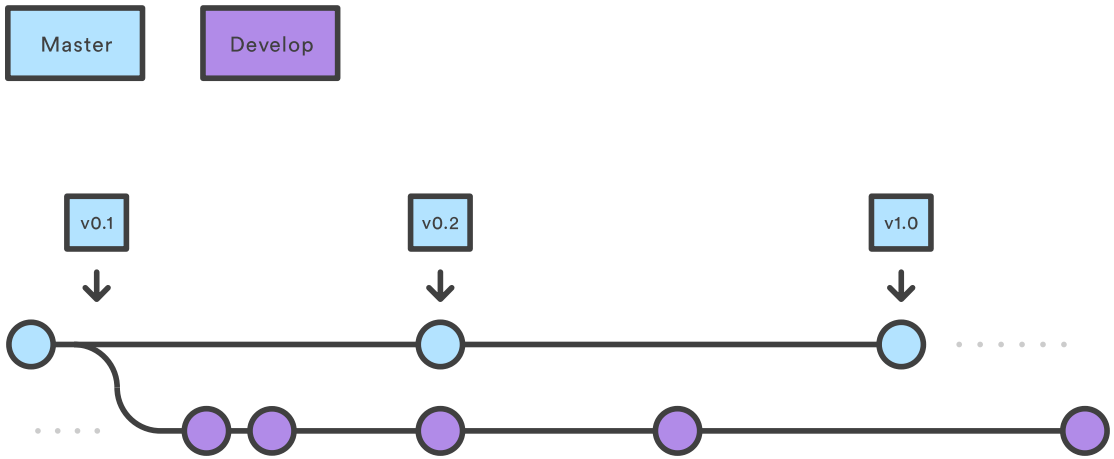


Git Flow

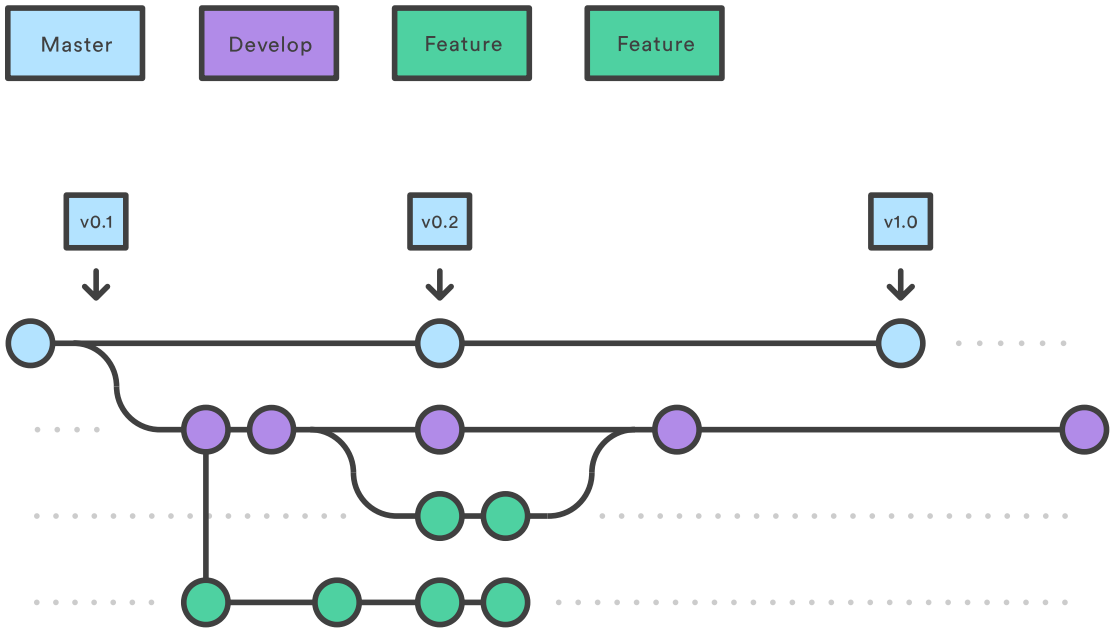
- Включает в себя строгие правила именования и работы с ветками.
- Отлично подходит для проектов с запланированным релизным циклом.



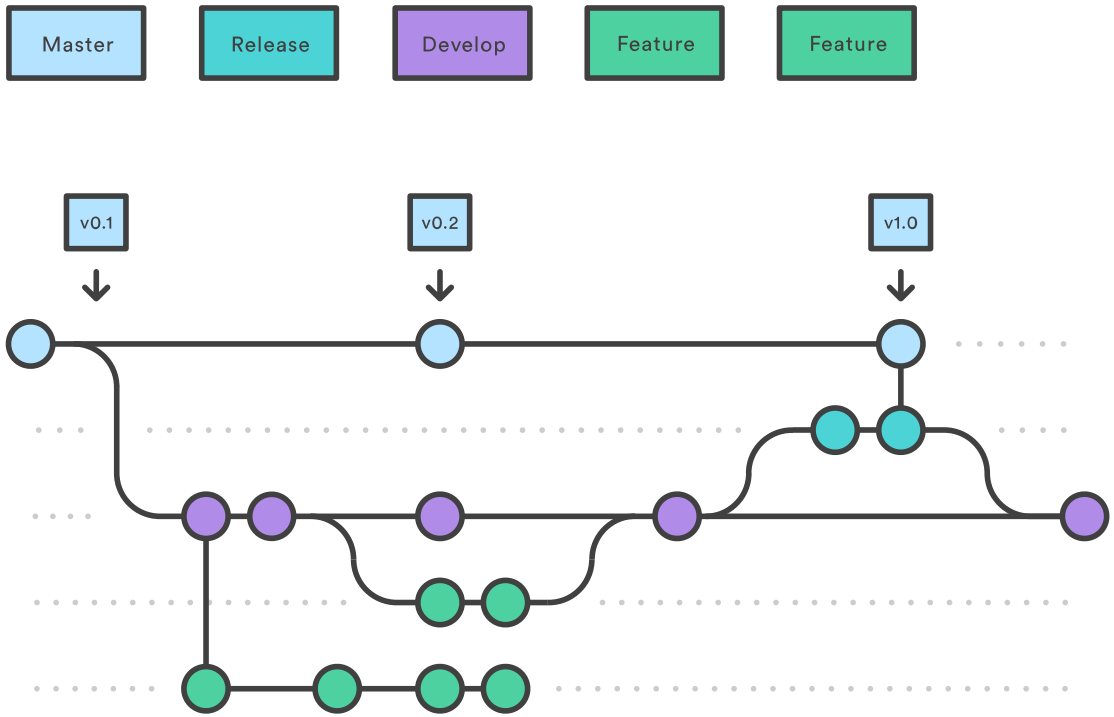
Develop and Master Branches



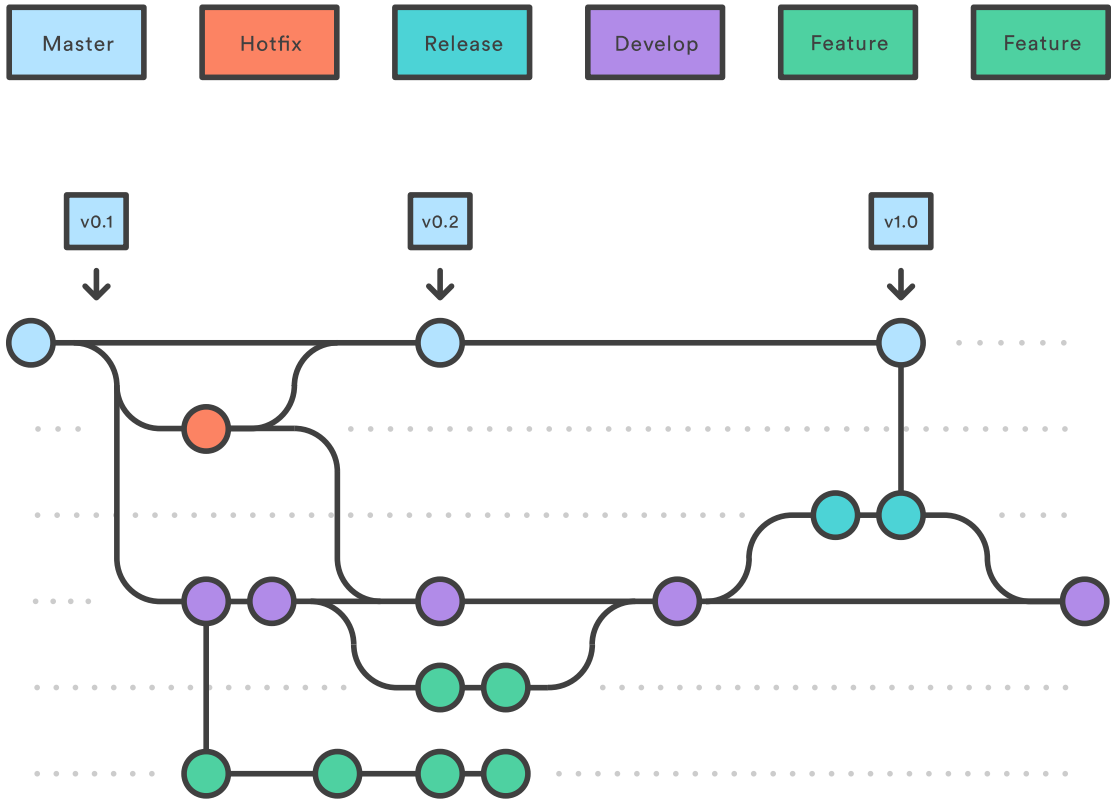
Feature Branches



Release Branches

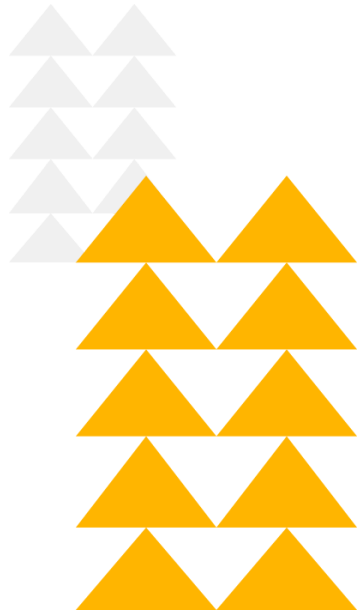


Hotfix Branches



Take away: Git Flow

1. Ветку develop создаем из master
2. Ветку release* создаем из develop
3. Ветку feature* создаем из develop
4. Когда feature* готова мержим ее в develop
5. Когда release* готов мержим его в develop и master
6. Если нашли дефект, создаем ветку hotfix* из master
7. Когда hotfix* готов мержим его в develop и master



Спасибо за внимание

Виталий Кудёлка

Разработчик программного обеспечения

vital.kudzelka@leverx.com

github.com/vitalk