Технология командной разработки ПО

Лекция n из N: Git

GIT background

- Разработан *Linus Benedict Torvalds*

- Цель: способствовать развитию Linux ядра

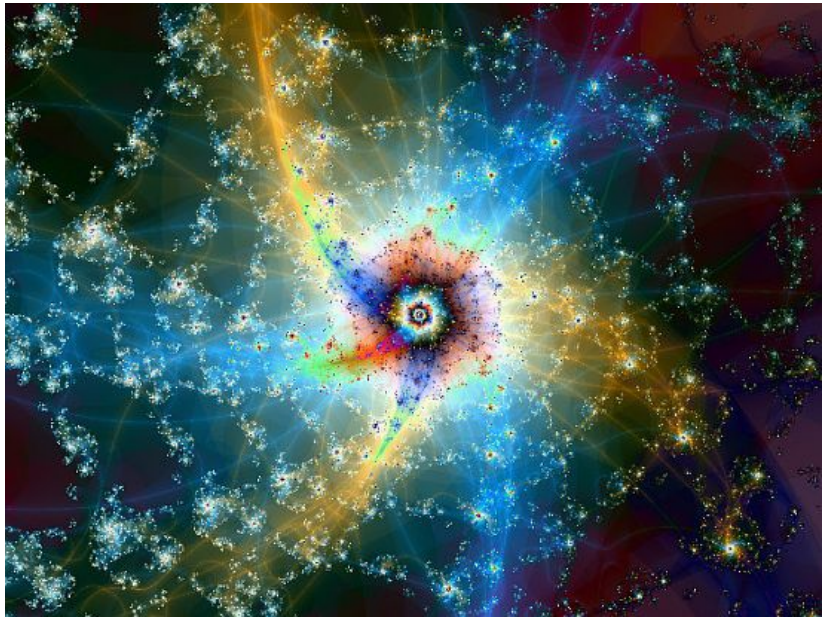
- Уверенность в том, что SVN/CSV - зло

Надежность

- Git следит за всем репозиторием, а не только за отдельными файлам и или их именами
- Для проверки целостности используется SHA1
- Дублирование данных

Распределенная хрень

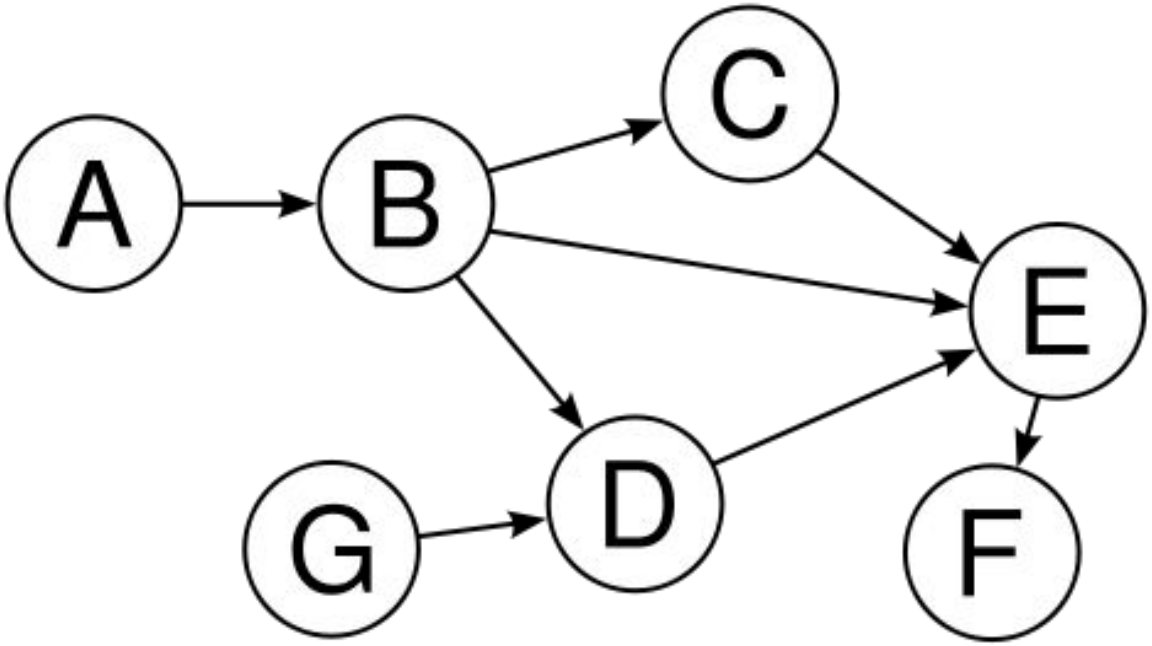
- **Нет** центрального репозитория
- У каждого свой луна-парк
- **Сеть доверие.** Каждый может попробовать положить «варенья» в кашу, но не каждому это будет позволено.



Преимущества, ставшие стандартом

- Branching и merge – за доли секунды.
- Diff ядра Linux меньше чем за секунду.
- Репозиторий KDE меньше 2x Gb, в то время как SVN на это тратил 8 Gb.

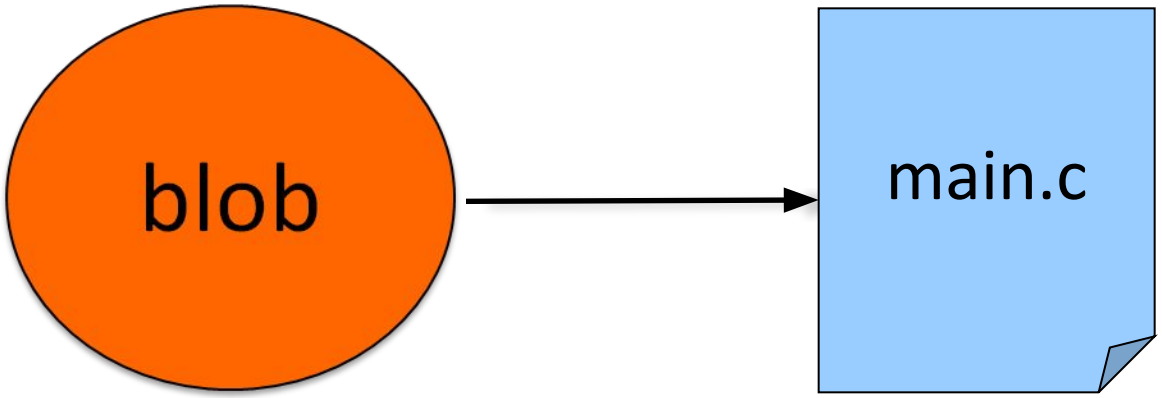
Направленный ациклический граф



Объекты

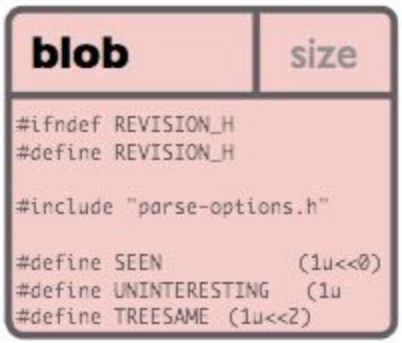
- **blob** используется для хранения содержимого файлов, можно сказать что это и есть файлы
- **tree** можно провести аналогию с директорией, это связанные ссылки на другие объекты типа **«blob»** и/или **«tree»** (по аналогии файлы и вложенные директории)
- **commit** указывает на объект типа **«tree»**, отмечая как проект выглядел в определённый момент времени. Так же содержит мета-данные о времени и авторе изменений с момента последнего «коммита», указатель на предыдущий «коммит»
- tag - это способ добавить особую отметку к «коммиту». Обычно используется для пометок определенных моментов разработки

Blob

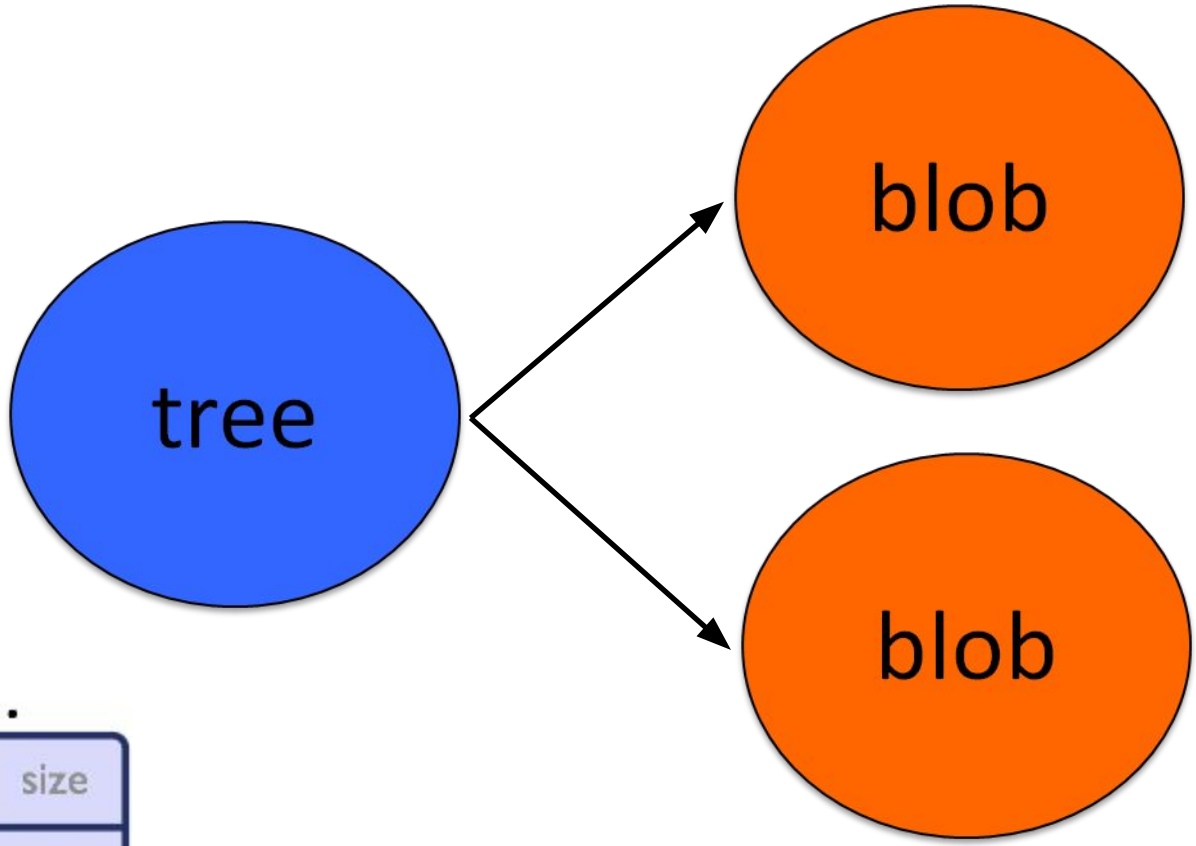


457aef93ff7ffbb289f7e1384f900679eacf044a

5b1d3..



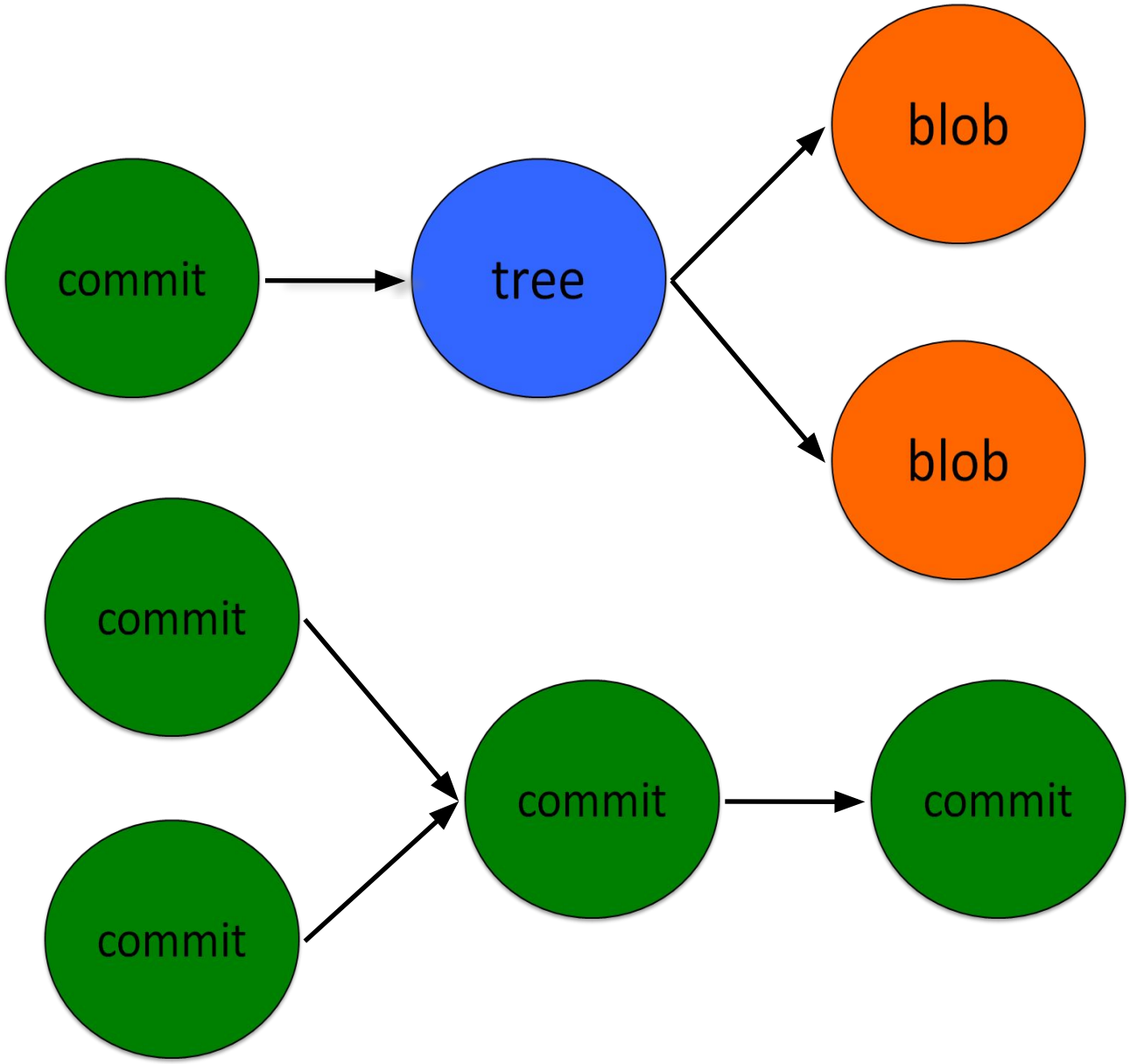
tree



c36d4..

tree		size
blob	5b1d3	README
tree	03e78	lib
tree	cdc8b	test
blob	cba0a	test.rb
blob	911e7	xdiff

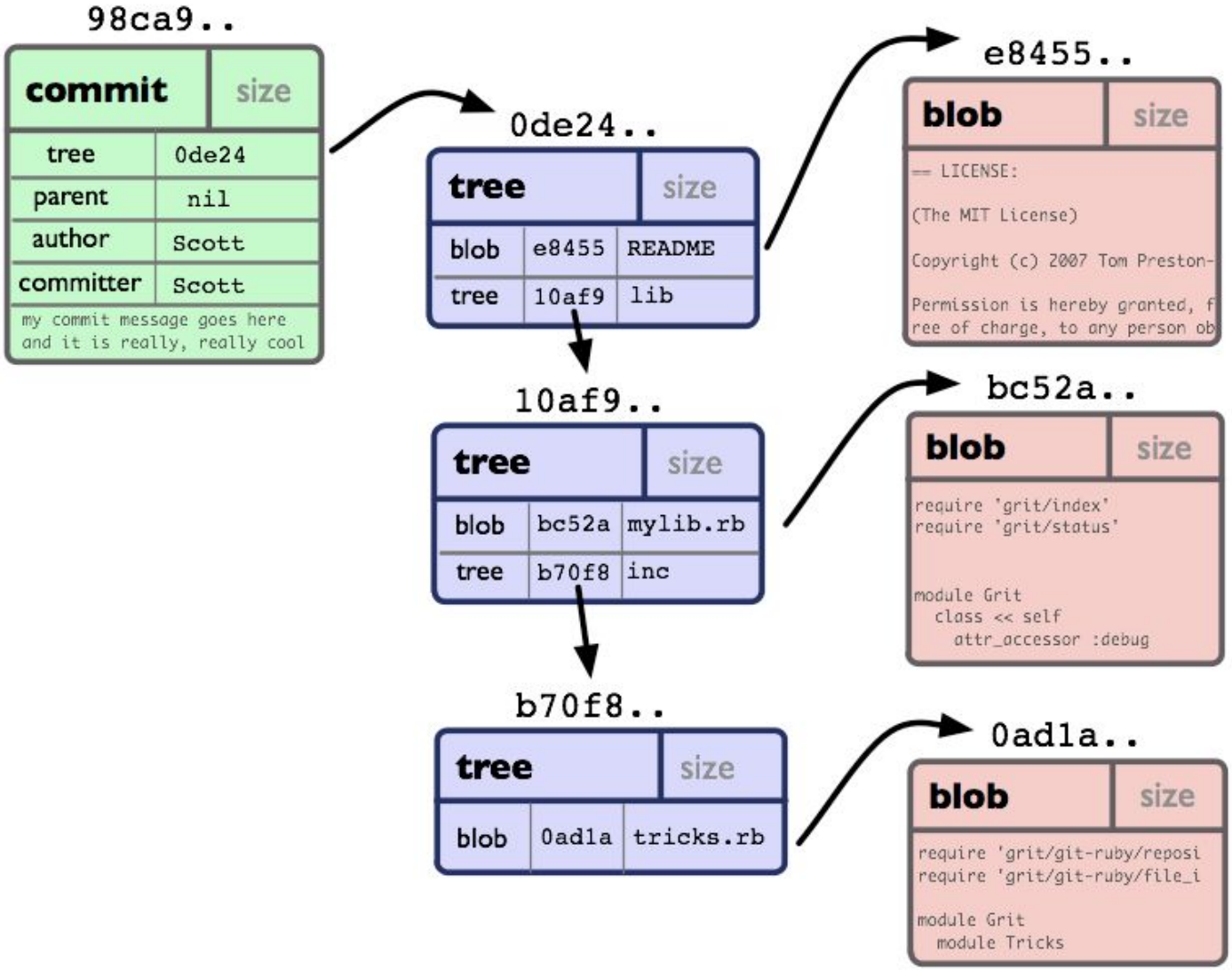
Commit



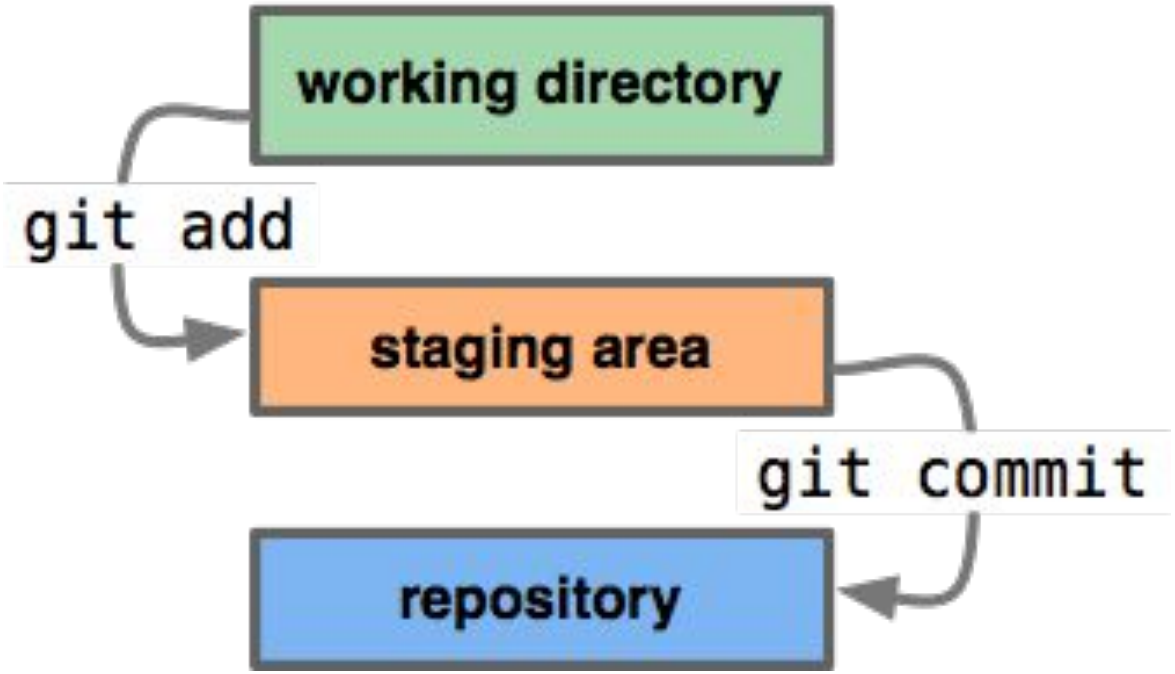
ae668..

commit	size
tree	c4ec5
parent	a149e
author	Scott
committer	Scott
my commit message goes here and it is really, really cool	

И все вместе


98ca9..
commit
size
tree
0de24
parent
nil
author
Scott
committer
Scott
my commit message goes here
and it is really, really cool
0de24..
tree
size
blob
e8455
README
tree
10af9
lib
e8455..
blob
size
== LICENSE:
(The MIT License)
Copyright (c) 2007 Tom Preston-
Permission is hereby granted, f
ree of charge, to any person ob
10af9..
tree
size
blob
bc52a
mylib.rb
tree
b70f8
inc
bc52a..
blob
size
require 'grit/index'
require 'grit/status'
module Grit
class << self
attr_accessor :debug
b70f8..
tree
size
blob
0ad1a
tricks.rb
0ad1a..
blob
size
require 'grit/git-ruby/reposi
require 'grit/git-ruby/file_i
module Grit
module Tricks

Промежуточная область



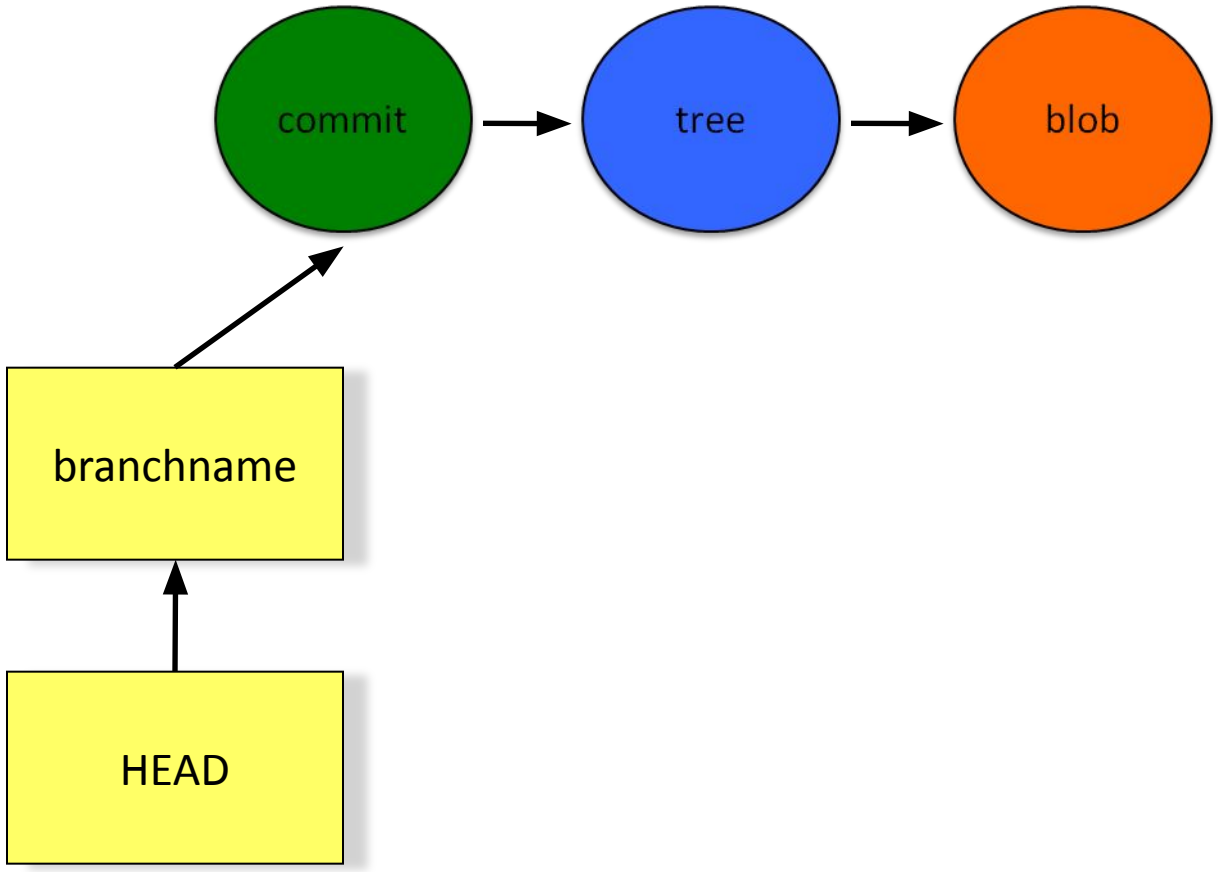
git add fileName
git add -a
git status

Сохранения состояния рабочей копии

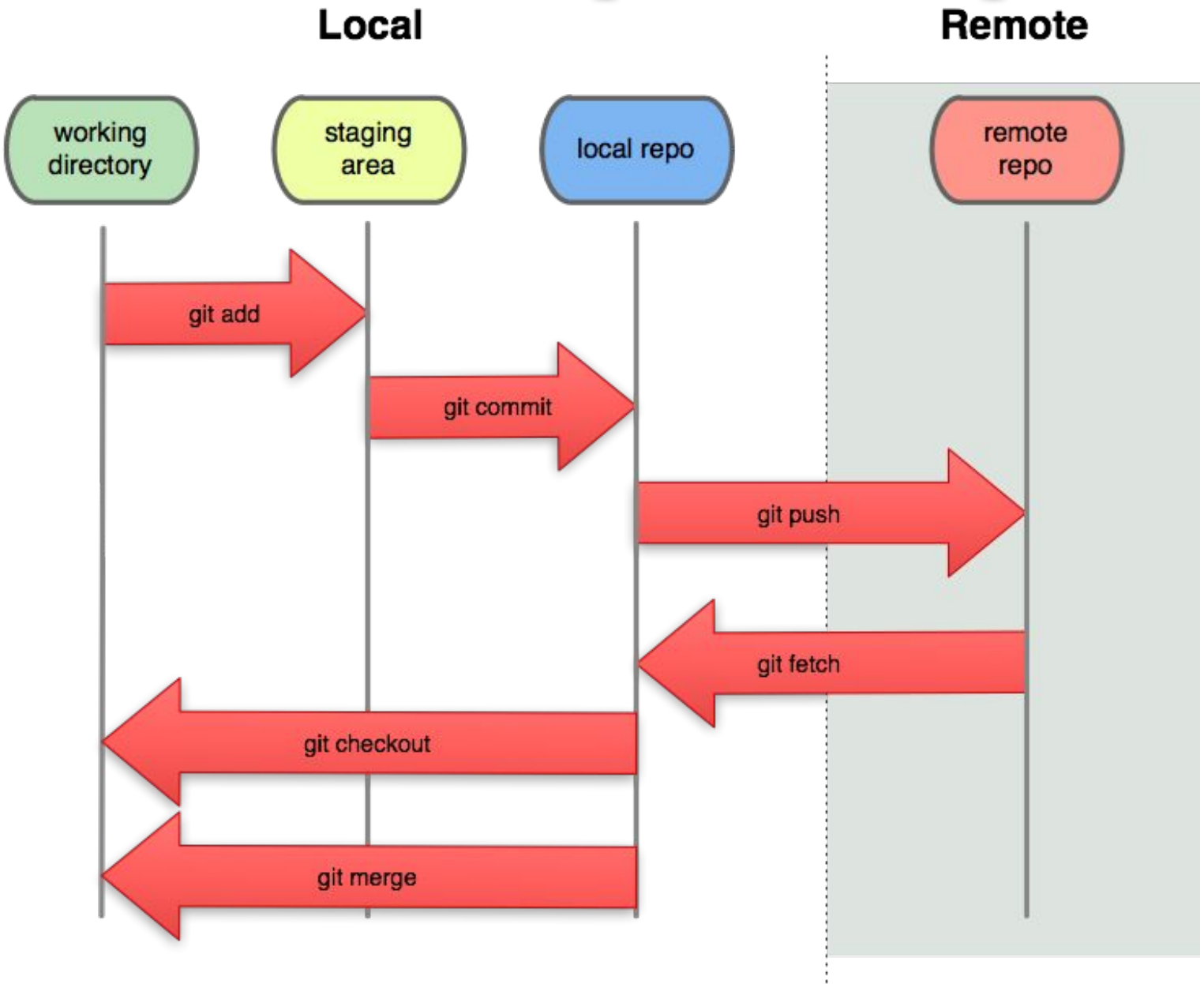
- git stash
- git stash pop
- git stash list
- git stash save "Описание”

- Очень помогает при работе с внешним репозиторием git или svn

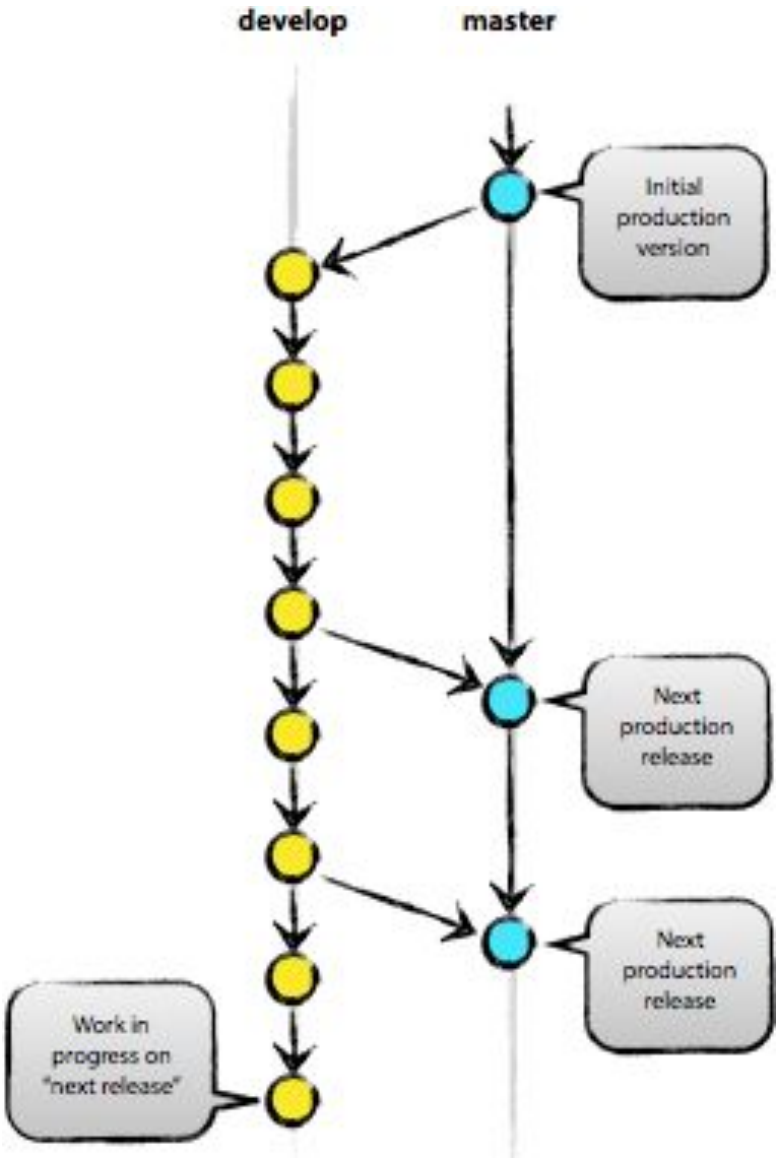
Ветки



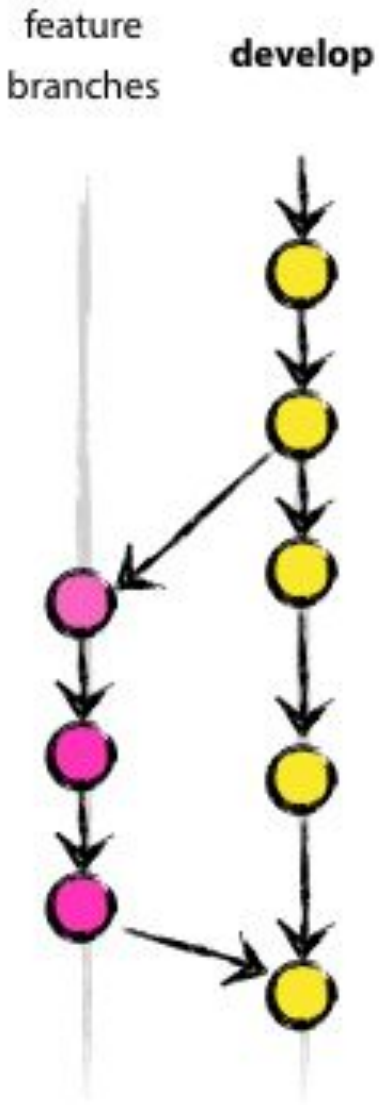
Удаленные репозитории

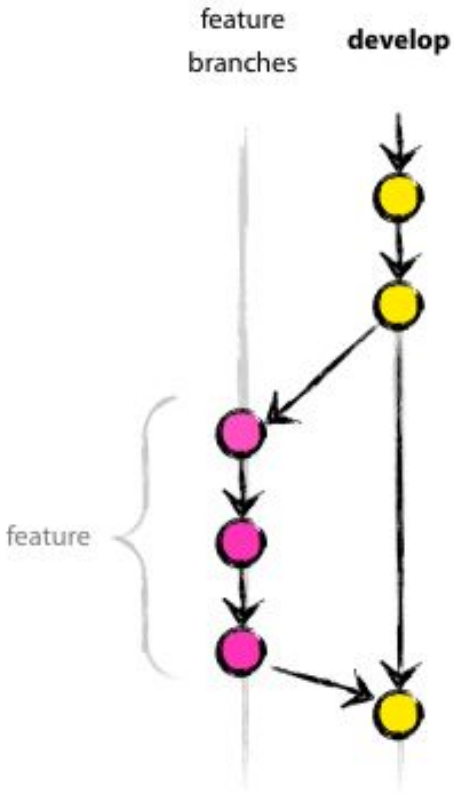


git flow

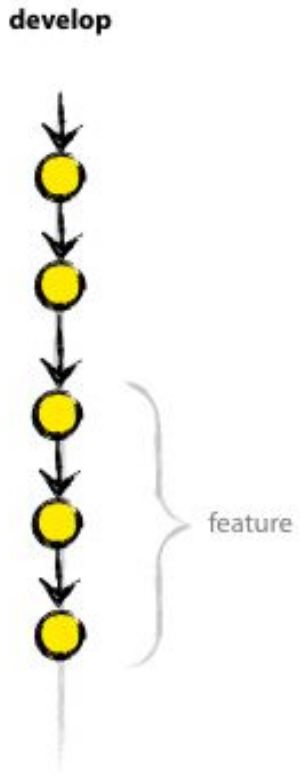

develop
master
Initial production version
Next production release
Next production release
Work in progress on "next release"

git flow



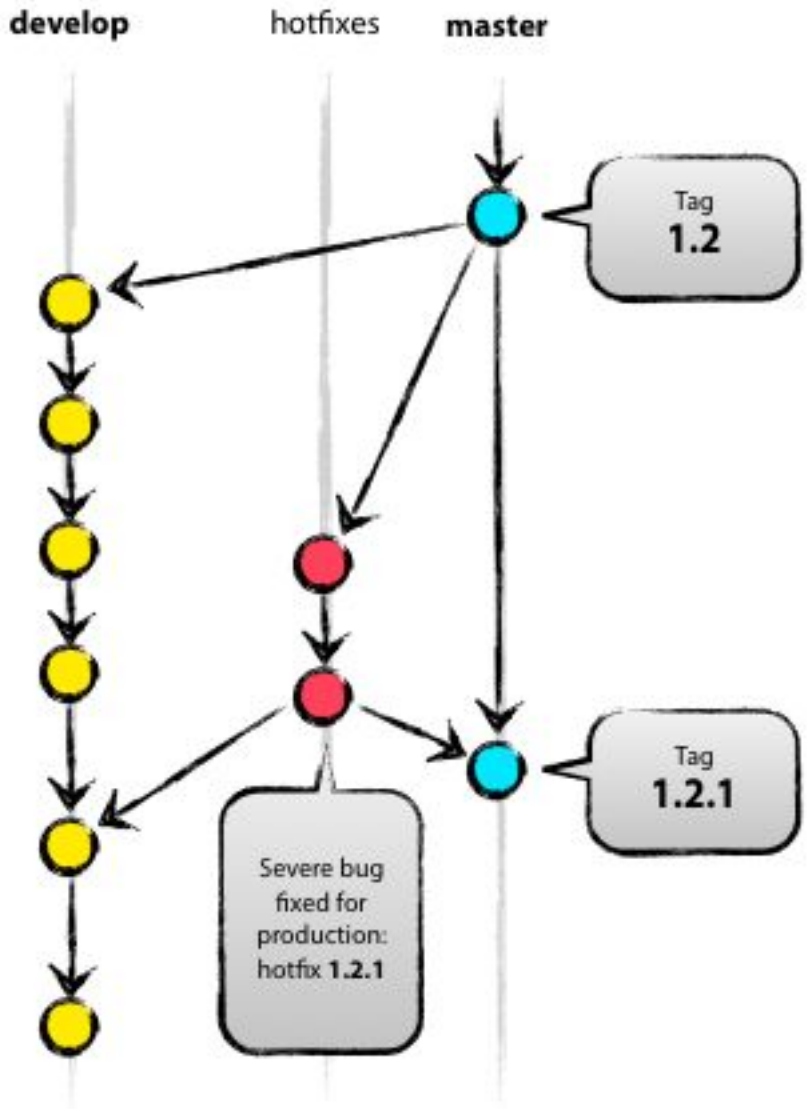


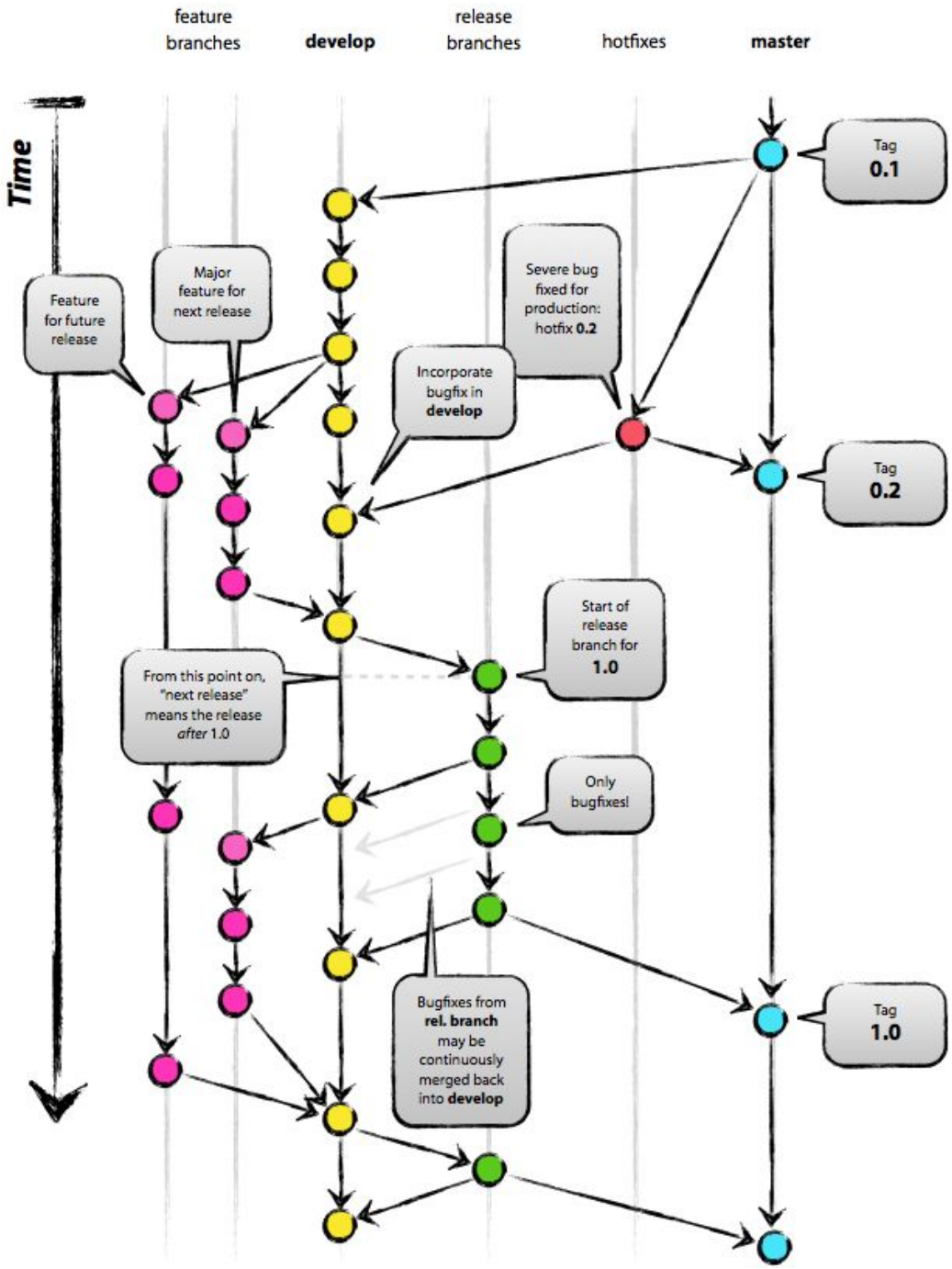
git merge --no-ff



git merge
(plain)

git flow


develop
hotfixes
master
Tag
1.2
Tag
1.2.1
Severe bug
fixed for
production:
hotfix 1.2.1


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Показать примеры onboarding'a и регламента работы с git

- QUICK START
- Дорожная карта нового разработчика
- Инструкция по работе с gitlab
- Git коммиты в Jira

Каким должен быть commit

- Коммит должен быть маленьким.
  Маленькие коммиты соответствуют парадигме юнит-тестирования и позволяют остановиться здесь и сейчас и продолжить работу потом другому человеку или в другом месте.
- Коммит должен быть частым.
- Коммит должен быть целостным.
  То есть коммитятся сразу все связанные измененные файлы.
- Семантически разные изменения (правка чужих опечаток, переносов строк и непосредственно работа по тикету) должны идти разными коммитами.
  Да, это демотивирует делать исправления, но так диффы становятся в разы читабельнее.
- **Каждый коммит должен сопровождаться ссылкой на тикет, который выполняется**
  Это позволяет связать коммиты с тикетом.
- Не должно быть коммитов типа "Коммит — просто коммит" или "Правки после ревью".
  Должно быть понятно, какого рода изменения были применены.
- Коммиты должны отражать последовательность действий при работе над тикетом.
- Доработки после ревью желательно делать в виде "fixup!" коммитов.
  Отдавать в следующий раз в ревью надо ветку с именно этой порцией коммитов.
- Обязательно перед мёржем feature-ветки надо fixup-ить все " fixup!" коммиты.
- Логические изменения должны быть сделаны в отдельных атомарных коммитах, код в которых желательно должен оставаться рабочим на любом участке ветки.
  - Тесты, фикстуры, документация, правки NEWS атомарно сопровождают тестируемый код/функционал в рамках одного коммита
  - Поправки тестов, кода, документации, линтеров, чего-либо ещё стороннего функционала (не связанных напрямую с тикетом) должны выполняться в рамках отдельных коммитов
  - Исправления баг и улучшения производительности – это отдельные коммиты
  - Однотипное изменение во многих файлах/модулях можно поместить в один коммит

Сообщение к commit'у

Сообщение каждого коммита должно иметь следующий формат:

Резюме изменений с заглавной буквы (строго меньше 100 символов) XXXXX

Текст более детального описания, если необходим. Старайтесь не превышать длину строки в 100 символов. Пустая строка, разделяющая сообщение, критически важна (если существует детальное описание); различные инструменты типа `log`, `shortlog` и `rebase` могут не понять вас, если заголовок и тело не будут отделены.

Последующие параграфы должны отделяться пустыми строками.

- Списки тоже подходят
- Обычно, элементы списка обозначаются с помощью тире или звёздочки, предваряются одиночным пробелом

где, XXXX – идентификатор задачи в jira, например HYDRA-413 или B2B-13

- Резюме обязательно.
- Резюме показывается в заголовках почтовых сообщений патчах, показывается в однострочном git log.
- Детальное описание может отсутствовать.
- Разделитель резюме от описания обязателен (если описание вообще присутствует).
- Ограничение в 100 взято из-за того, что gitlab обрезает сообщения длинной >= 100 символов:

Сообщение к commit'у

- Помните, что 100 символов - это строгое ограничение, продиктованное используемым у нас инструментом.
- В open source сообществе приветствуется правило "50/72 Formatting", при котором резюме должно быть не длиннее 50 символов, а детальное описание не превышает длину строки в 72 символа. И стремиться к такому никогда не помешает (smile)
- Текст сообщения к коммиту должен быть на русском языке, исключением может являться случай коммита в код, который компания открывает в open source.
- Если есть подпроекты (например в репозитории da), то в резюме надо указать подпроект.
- Подпроект перед резюме позволяет легко определить, к чему относятся изменения.
- Пишите в инфинитиве в совершенном виде:
- "применить", "исправить", а не "применил", "сделано", "добавлено", "применяю", "изменения"
- Детализируйте ЧТО изменилось и ПОЧЕМУ изменилось, а не КАК.
- КАК что-то было проделано – есть в diff-е коммита.
- Важна МОТИВАЦИЯ (отраженная в сообщении к коммиту) изменений в реализации (diff).
- Следите за орфографией в сообщениях.
- Если нужно, используйте редакторы с проверкой орфографии (например, Vim)