Trunk-Based Development в enterprise

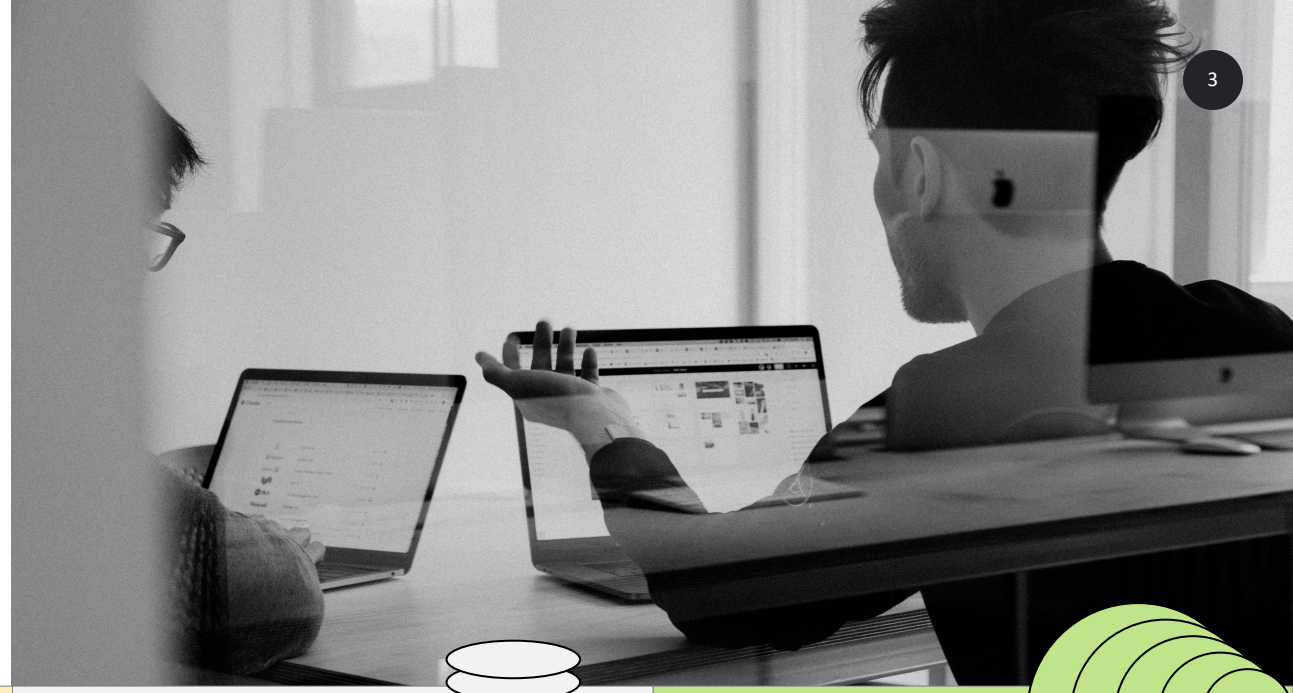
издание 2, дополненное

Артем Шакуров

Артем Шакуров

- Тружусь в X5 Tech руководителем команды разработки
- Отдаю предпочтение async python
- Люблю активный отдых

Как наша команда пришла к TBD?



Дано

Сложный сильно-связный (high coupling) legacy-сервис

Тестов нет

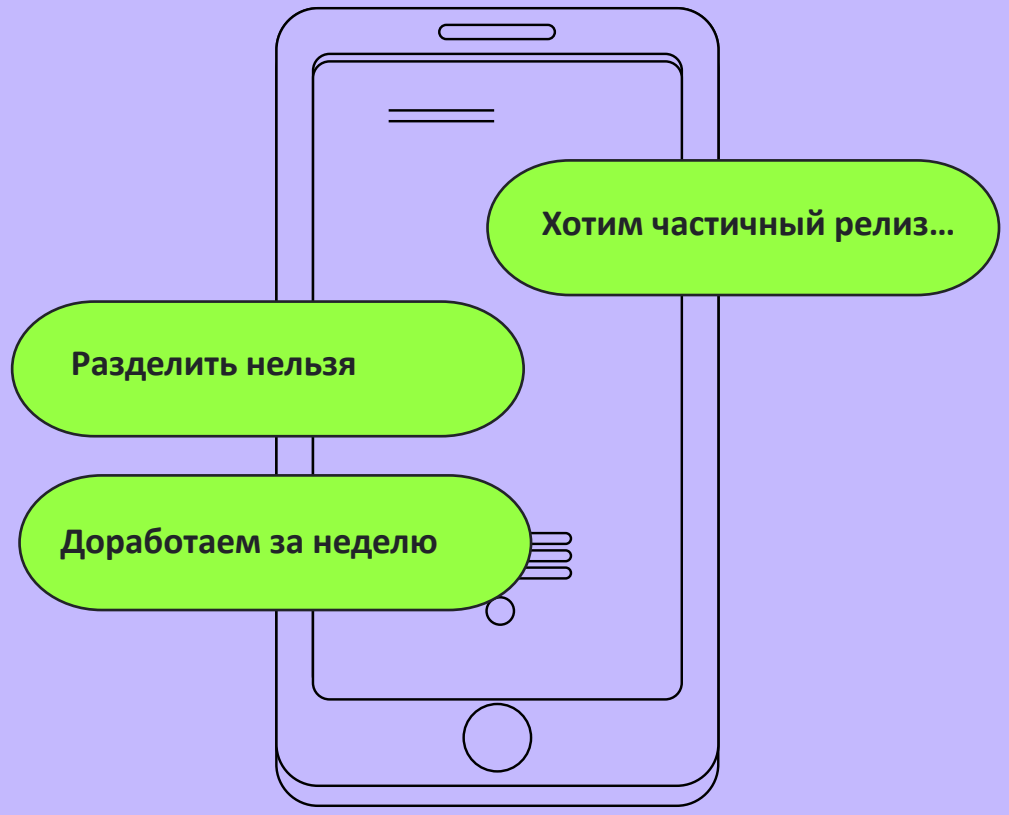
Логистика и финансы – высокая ответственность

Fear Driven Development

Как наша команда пришла к TBD?

senior engineer maintaining a legacy application with a bunch of first year devs starter pack

Как наша команда пришла к TBD?



Как наша команда пришла к TBD?

Как наша команда пришла к TBD?

Как наша команда пришла к TBD?

Как наша команда пришла к TBD?

Через 3 месяца случился успешный релиз!

Как наша команда пришла к TBD?

Что не учитывал старый процесс?

- Очень динамичный ритейл
- Много enterprise систем и legacy вокруг
- Встречается мусор в данных
- Много нюансов всплывает в процессе разработки

Как наша команда пришла к TBD?

Что не учитывал старый процесс?

- Очень динамичный ритейл
- Много enterprise систем и legacy вокруг
- Встречается мусор в данных
- Много нюансов всплывает в процессе разработки

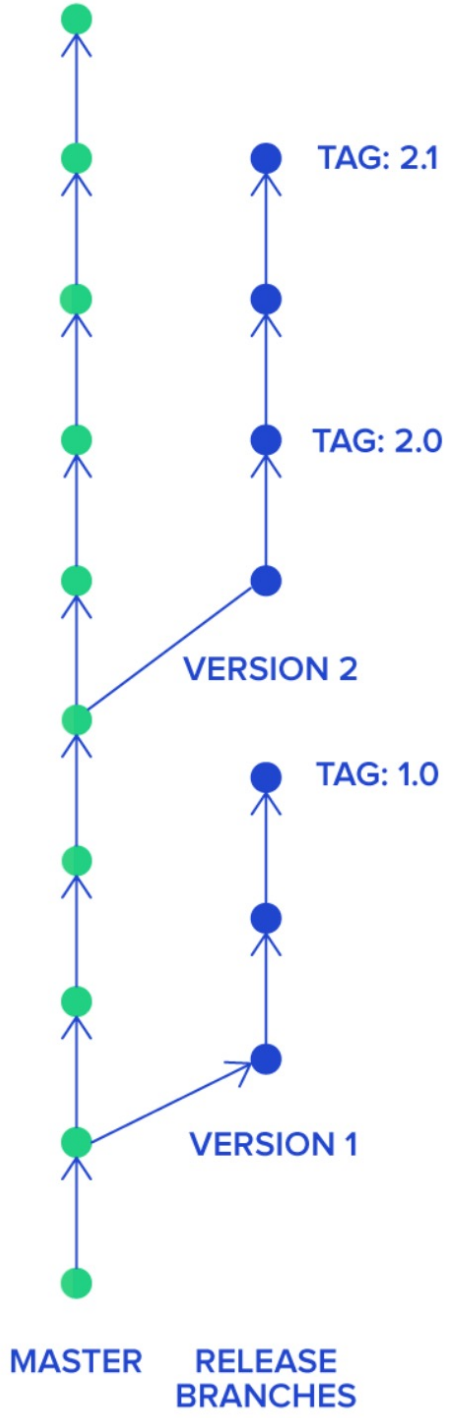
TBD для нас — способ повысить частоту релизов за счет «конструктора» фич.

План

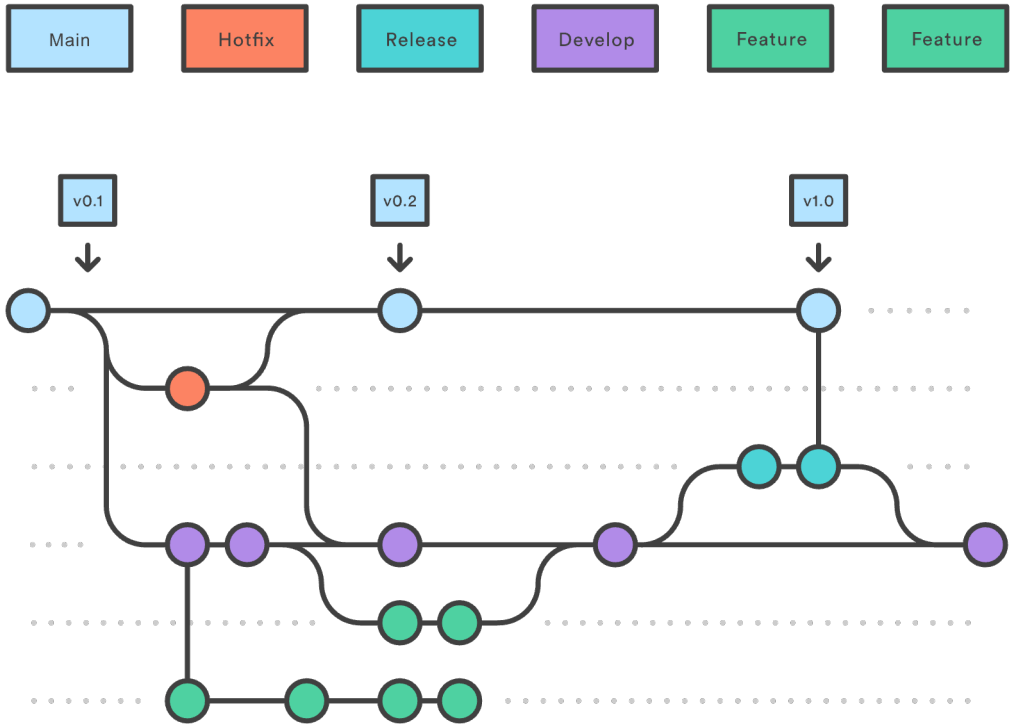
1. Особенности TBD workflow
2. Инструменты в TBD
3. Release-management
4. Ограничения TBD
5. Наш опыт внедрения в сложный продукт

Что такое Trunk?

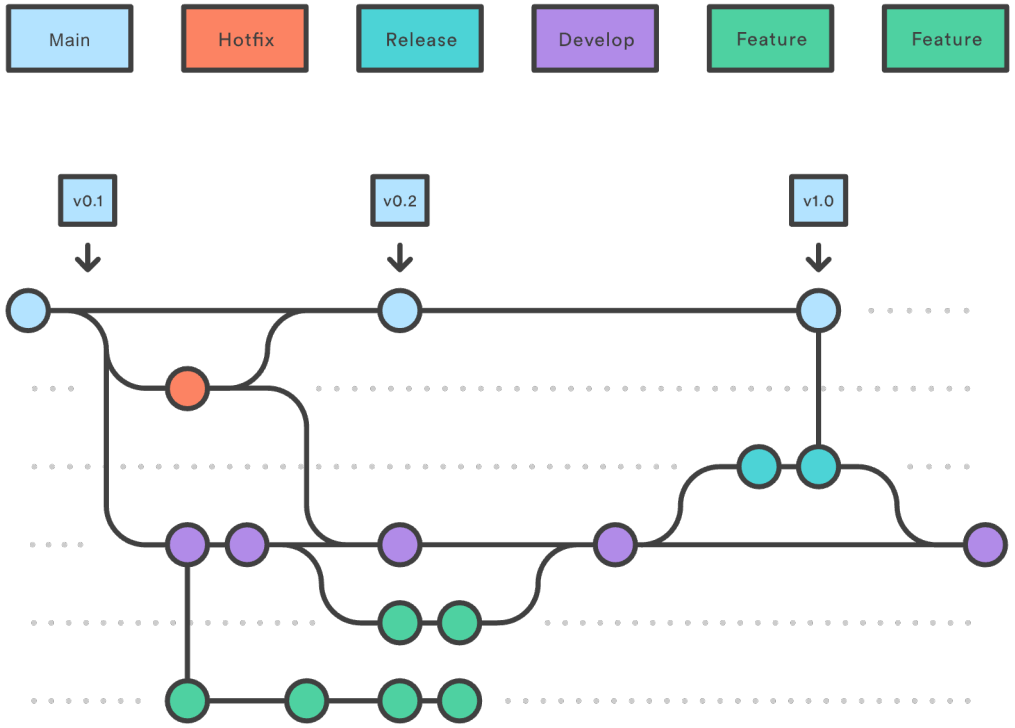
Trunk — ствол или магистраль — легко-читаемая модель ветвления в GIT за счет прозрачной истории коммитов в одной ветке, всегда готовой к релизу.



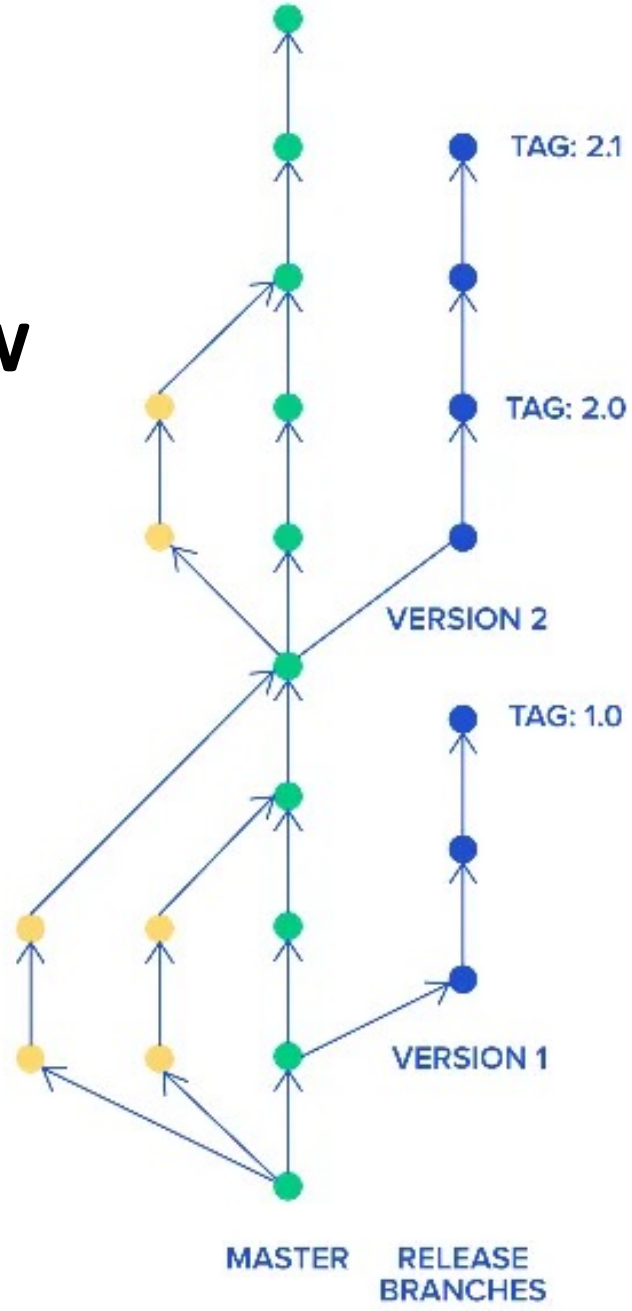
Gitflow



Gitflow



TBD flow



За счет чего такая простота?

1. Максимальная декомпозиция на микротаски с короткоживущими ветками

2. Feature-flags

3. Squash-rebase

4. Release-candidate-ветки

5. Практики тестирования

6. Подходящий процесс Continuous Delivery

Как обычно декомпозируют?

Создать страницу с тегами (append-only) на БД-Бэк-Фронт

1. Создать сущность tag в БД
2. Реализовать API на добавление тега
3. Реализовать API на чтение тегов
4. Создать экран тегов на клиенте
5. Написать тесты

Как обычно декомпозируют?

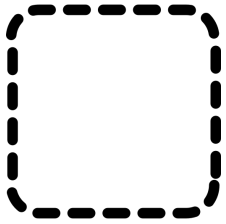
Создать страницу с тегами (append-only) на БД-Бэк-Фронт

1. Создать сущность tag в БД
2. Реализовать API на добавление тега
3. Реализовать API на чтение тегов
4. Создать экран тегов на клиенте
5. Написать тесты

Минимальный набор микротасков

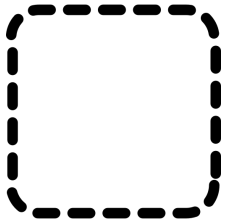
1. Создать миграцию на сущность в БД
2. Реализовать DTO / repository
3. Реализовать API на добавление
4. Реализовать API на чтение одного тега
5. Реализовать API на чтение списка с пагинацией
6. Создать экран на клиенте
7. Создать форму добавления
8. Создать форму отображения
9. Написать тесты на добавление уникального объекта
10. Написать тесты на добавление с ошибкой уникальности
11. Написать тесты на чтение одного тега
12. Написать тесты на чтение списка с пагинацией

Цели создания микротасков

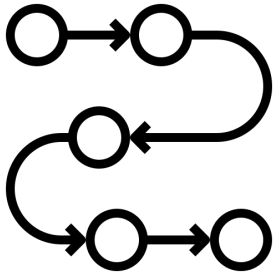


Очертить границы **исходной задачи**

Цели создания микротасков



Очертить границы исходной задачи



Задать task-chains изменений

Task-chains

- Последовательность разработки
- Последовательность доставки
- Не дерево, а DAG
- Неактивный код на PROD лучше, чем code-freeze и merge-hell!

Минимальный набор микротасков

1. Создать миграцию на сущность в БД
2. Реализовать DTO / repository
3. Реализовать API на добавление
4. Реализовать API на чтение одного тега
5. Реализовать API на чтение списка с пагинацией
6. Создать экран на клиенте
7. Создать форму добавления
8. Создать форму отображения
9. Написать тесты на добавление уникального объекта
10. Написать тесты на добавление с ошибкой уникальности
11. Написать тесты на чтение одного тега
12. Написать тесты на чтение списка с пагинацией

Свойства микротаска

- 15 мин – 1 час работы
- Не будет конфликтовать с другими
- Очень скромен по ресурсам
- Прозрачен: минимальный контроль
- Необязательно карточка в task tracker
- Но обязательно хотя бы предложение!
- Необязательно завершается кодом
  комментарии к задаче, скрины,
  ветки переписок из корпоративной почты
- Помогает в написании unit-тестов

Микротаски на изменение кода

Заменить поле в ETL-процессе

1. Добавить новое поле в целевую БД
2. Добавить обработку нового поля из источника данных
3. Адаптировать тесты под новое поле
4. Удалить старые код, тесты, поле в БД и пр.
   (после стабилизации пп.1-3)

Домыслы и слухи про микротаски

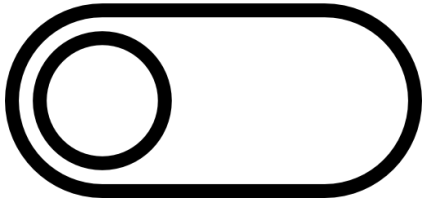
Работа с микротасками требует суперкрутых менеджеров

Микротаски == микро-менеджмент

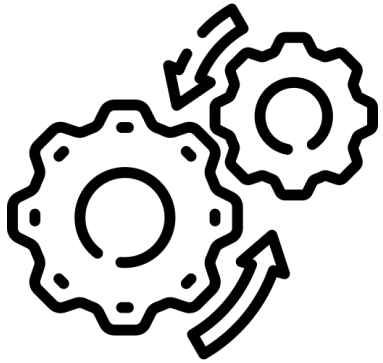
Время на декомпозицию тратится неоправданно

Можно просидеть над задачей много дней из-за неверной оценки

feature-flag



Feature-flag – возможность включать и выключать фичи в проекте



Желательно делать FF динамическими,
т.е. с минимальной зависимостью от CI/CD-процессов

Feature-flags не так просты

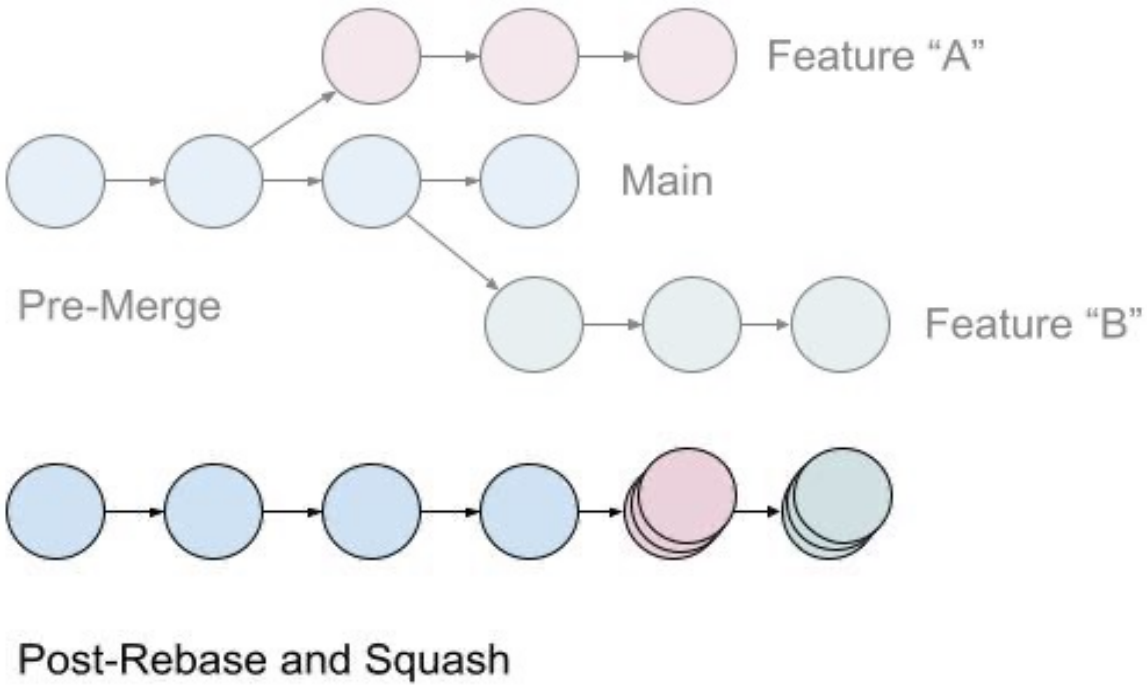
Что такое feature-flag?

- Заглушка на сырой функционал?
- Переключатель разных алгоритмов?
- Включатель для новых фич?

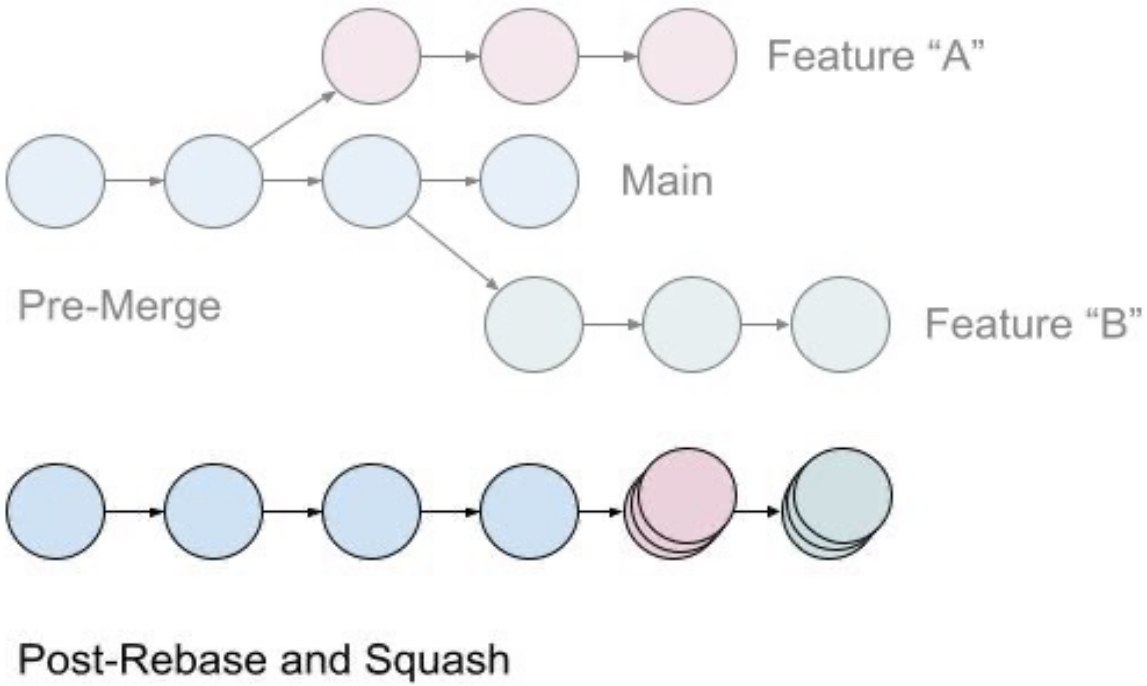
Как реализовать?

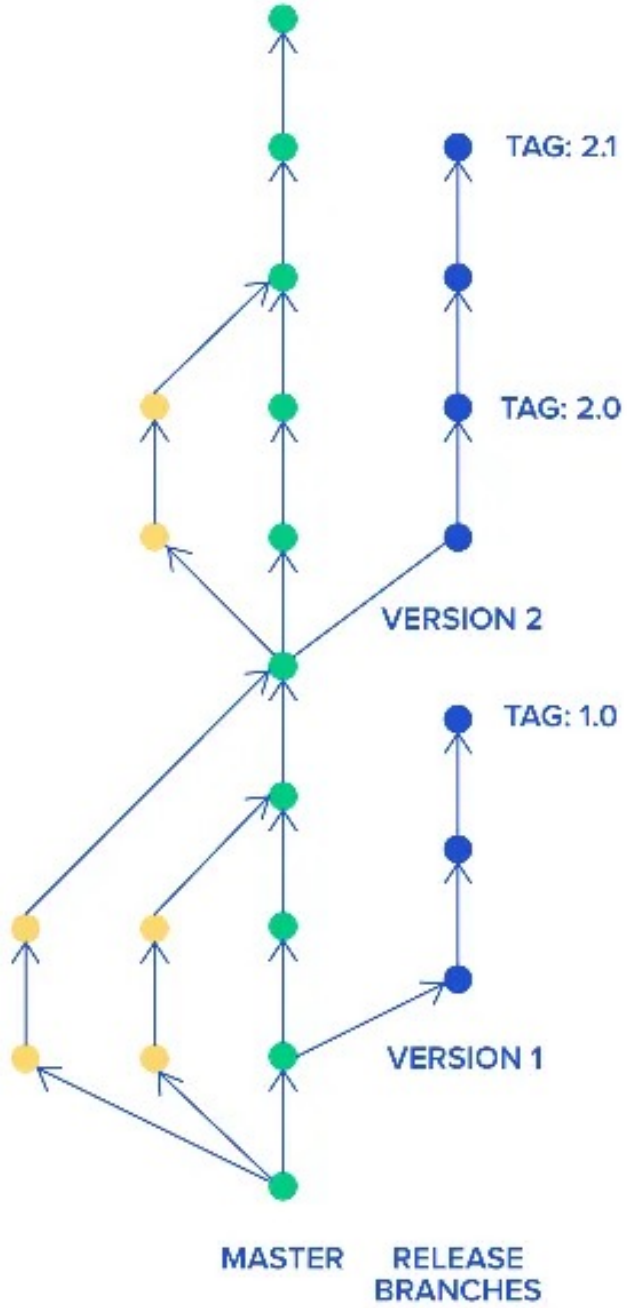
- if ...
- переменные окружения
- ЛК пользователя
- админка
- Consul, Redis и т.д.

Squash-rebase



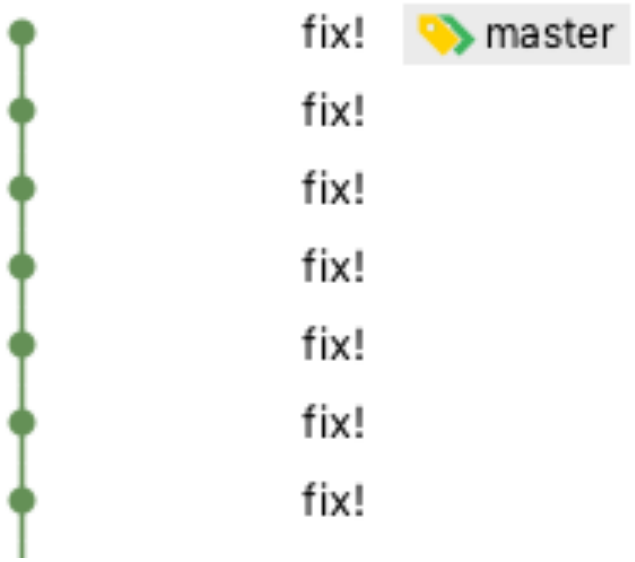
Squash-rebase





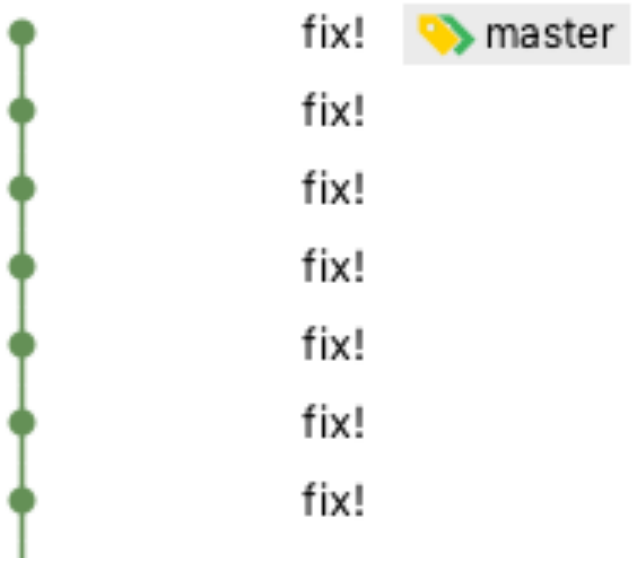
Squash-rebase обязательно!

Обязательный squash **не допускает лишнего в trunk:**

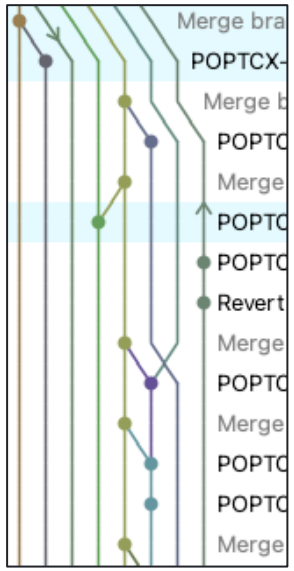


Squash-rebase обязательно!

Обязательный squash **не допускает лишнего в trunk:**

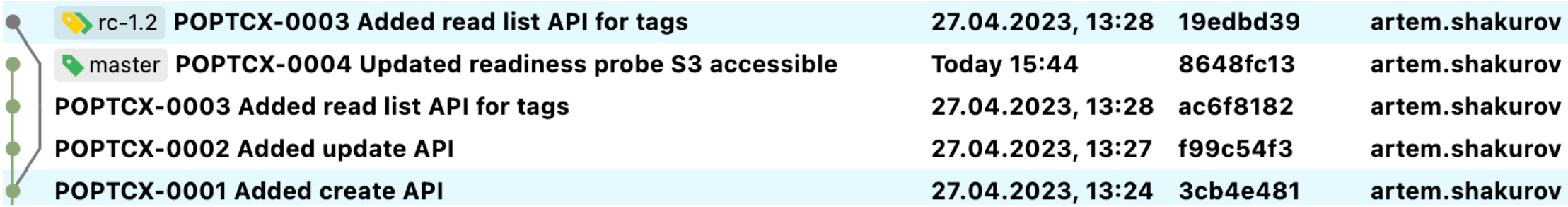


Обязательный post-rebase (fast-forward в gitlab) **убережет от «макарон»**:



release-candidate-ветки

- Имеет смысл хранить не более двух release-веток
- Название как договоритесь: rc-1.2, 4.5.x, Rel 8.12.x
- Cherry-pick для расширение релиза или для патча



rc-* можно «пересобирать»

- Нежелательные изменения можно убрать с помощью revert

master	POPTCX-0004 Updated readiness probe S3 accessible	Today 15:44	8648fc13	artem.shakurov
rc-1.1	Revert "POPTCX-0002 Added update API"	Today 15:41	e4d0fbb9	artem.shakurov
	POPTCX-0003 Added read list API for tags	27.04.2023, 13:28	ac6f8182	artem.shakurov
	POPTCX-0002 Added update API	27.04.2023, 13:27	f99c54f3	artem.shakurov
	POPTCX-0001 Added create API	27.04.2023, 13:24	3cb4e481	artem.shakurov

rc-* можно «пересобирать»

Черепикнул лишнее?

rc-1.2	POPTCX-0003 Added read list API for tags	27.04.2023, 13:28	3a3ca08e	artem.shakurov
	POPTCX-0002 Added update API	27.04.2023, 13:27	9bcc0f19	artem.shakurov
master	POPTCX-0004 Updated readiness probe S3 accessible	Yesterday 15:44	8648fc13	artem.shakurov
	POPTCX-0003 Added read list API for tags	27.04.2023, 13:28	ac6f8182	artem.shakurov
	POPTCX-0002 Added update API	27.04.2023, 13:27	f99c54f3	artem.shakurov
	POPTCX-0001 Added create API	27.04.2023, 13:24	3cb4e481	artem.shakurov

reset --hard в помощь

rc-1.2	POPTCX-0003 Added read list API for tags	27.04.2023, 13:28	43732d3c	artem.shakurov
master	POPTCX-0004 Updated readiness probe S3 accessible	Yesterday 15:44	8648fc13	artem.shakurov
	POPTCX-0003 Added read list API for tags	27.04.2023, 13:28	ac6f8182	artem.shakurov
	POPTCX-0002 Added update API	27.04.2023, 13:27	f99c54f3	artem.shakurov
	POPTCX-0001 Added create API	27.04.2023, 13:24	3cb4e481	artem.shakurov

Continuous Delivery

rolling

blue-green

cannary

При внедрении TBD стоит учитывать

Процессы QA
и практики тестирования

Время на перестройку CI/CD

Зрелость команды разработки

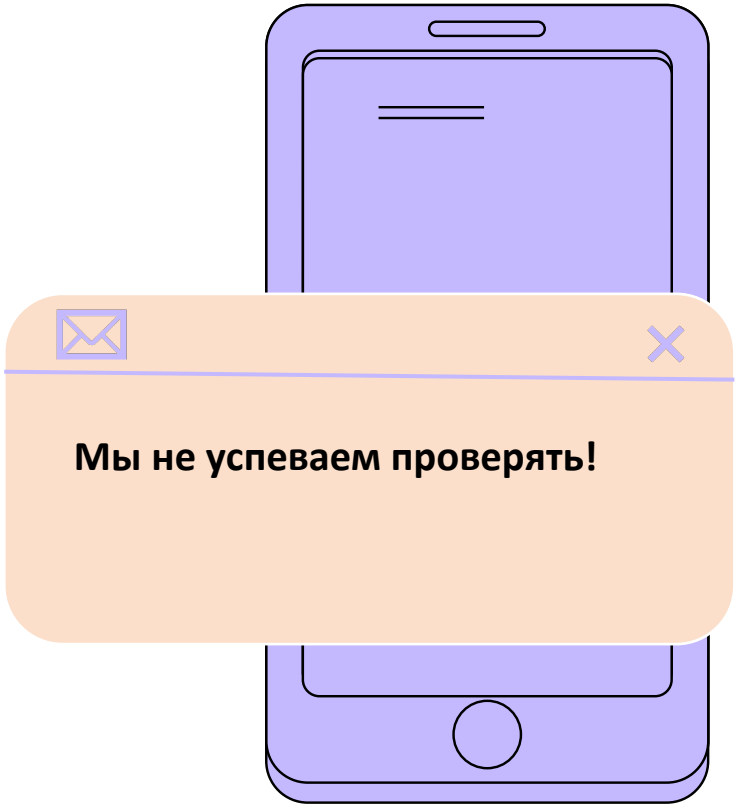
Готова ли команда к TBD?

- Соблюдаете формат commit-message,
  и работает связка с task tracker’ом
- Пи́шите тесты
- Знаете GIT: reset --hard, revert, rebase и пр.
- Ведете рефакторинг отдельно от бизнес-изменений
- Общаетесь в команде с помощью артефактов

Наш результат

1 релиз раз в
3 месяца

1 релиз раз в
неделю



Всегда можно замедлиться =)

Наш результат

Необязательно менять все

У нас остались процессы:

- UAT при завершении разработки функциональности
- Технические окна — отказ в обслуживании при релизе
- Ручные code-review

- Таски-**активаторы** – это наши feature-flags =)

Советы по внедрению TBD

- [x] Внедряйте TBD постепенно
- [x] Ведите чек-листы
  - [ ] DoR и DoD
  - [ ] Шаблон релиза
  - [ ] Шаблон постмортема
- [x] Автоматизируйте
  - [ ] Боты
  - [ ] Merge Trains
  - [ ] Свои webhooks

Если не хотите сразу в TBD

Важно понимать: особые преимущества дают практики и улучшения процессов, не сама методология.

Попробуйте:

- декомпозировать задачи на микротаски
- поэкспериментировать с историей в GIT
- контролировать фичи с помощью feature-flag’ов

https://trunkbaseddevelopment.com/

Практические примеры разбиения больших задач на микротаски
https://teamleadconf.ru/moscow/2020/abstracts/6240

Why Google Stores Billions of Lines of Code in a Single
https://csci572.com/papers/2016GoogleRepository.pdf

Kira's family bots
https://github.com/wemake-services/kira

X5 Tech