Система контроля версий

Система управления версиями (от англ. Version Control System, VCS или Revision Control System) — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

Для чего нужно: история изменений, откат нежелательных изменений, совместная работа, код не теряется, нерабочие фичи не ломают основной билд

Для чего применять VCS:

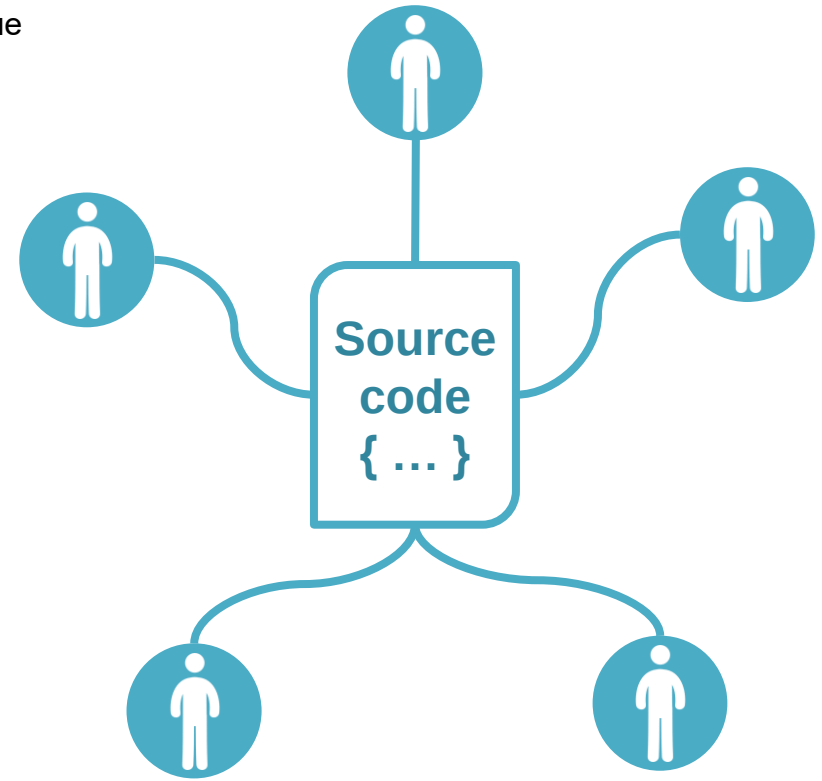
- для любых текстовых файлов: исходники ПО, README (в простом текстовом формате), html и различные скрипты

Для чего не применять VCS:

- для любых двоичных файлов (дистрибутивы, документы в формате doc, xls и прочее, архивы, скомпилированные сущности) - для их версионирования часто имеются специальные инструменты,
- автоматически сгенерированные файлы - сгенерированные классы веб-сервисов, файлы проекта (за некоторым исключением) - данные файлы не требуют контроля версий

https://sbtatlas.sigma.sbrf.ru/wiki/pages/viewpage.action?pageId=44502656

https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud

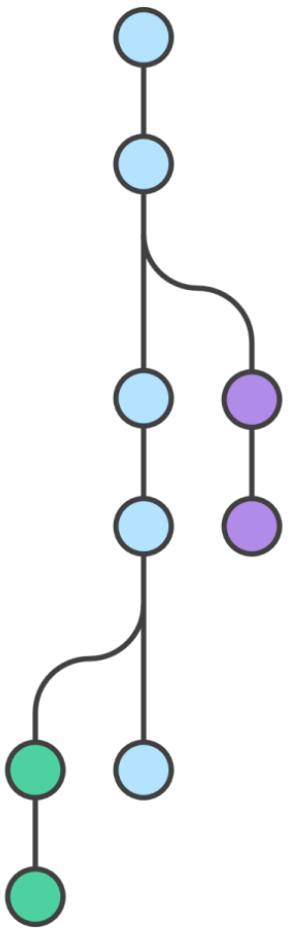


Модели ветвления

Существуют различные модели ветвления, каждая из которых преследует свои цели.

На данный момент наиболее актуальны и популярные модели ветвления – это:

- **Git flow**
- **Github flow**
- **Gitlab flow**



! ВАЖНО: Модели ветвления работают только при условии тщательного соблюдении схемы.

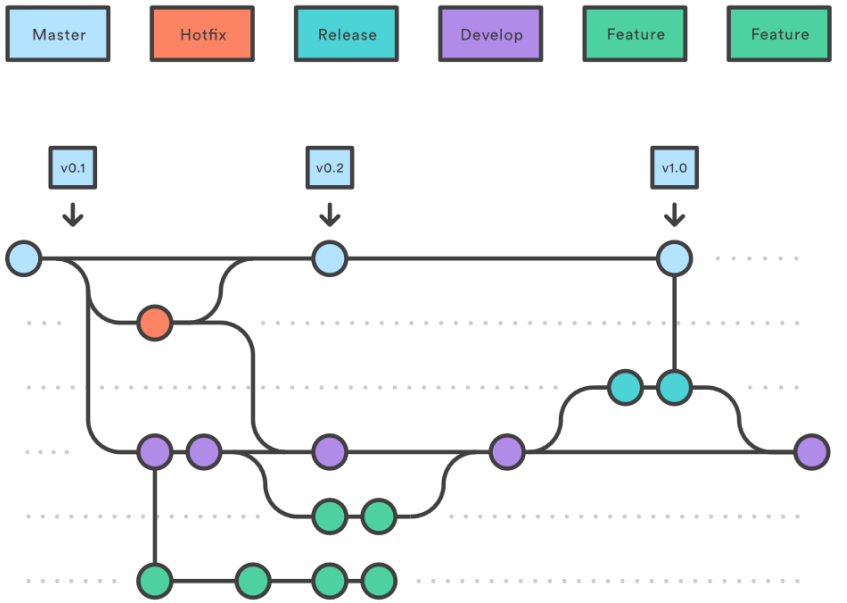
Базовые принципы популярных моделей ветвления

- Любое значимое изменение должно оформляться как отдельная ветвь
- Текущая версия главной ветви всегда корректна. В любой момент сборка проекта, проведённая из текущей версии, должна быть успешной
- Версии проекта помечаются тегами. Выделенная и помеченная тегом версия более никогда не изменяется
- Любые рабочие, тестовые или демонстрационные версии проекта собираются только из репозитория системы

https://www.atlassian.com/git/tutorials/comparing-workflows

Модели ветвления: Git Flow

Данная модель является практической реализацией основных принципов разработки ПО в VCS с учетом релизных циклов ПО. Подходит как для Scrum, так и для Waterfall.



Принципы

- Две главные ветки:
  - **master** – текущая стабильная версия, работающая на пром-среде
  - **develop** – основная ветка разработки
- Вспомогательные ветки:
  - **feature** – новый функционал или исправление некритичных багов; сливаются в develop
  - **release** – фиксация и стабилизация релиза (feature-freeze); сливаются в master и develop
  - **hotfix** – исправление критичных багов; сливаются в master и develop
- Все слияния происходят только через pull-request'ы

Плюсы

- Фиксация и стабилизация функционала релиза до попадания на пром
- Возможность выпускать ночные сборки (nightly build)

Минусы

- Не самая простая схема работы; требует досконального понимания от разработчика, какие ветки откуда создаются и куда сливаются
- Готовый функционал попадает на пром с некоторой задержкой

https://sbtatlas.sigma.sbrf.ru/wiki/pages/viewpage.action?pageId=65257821

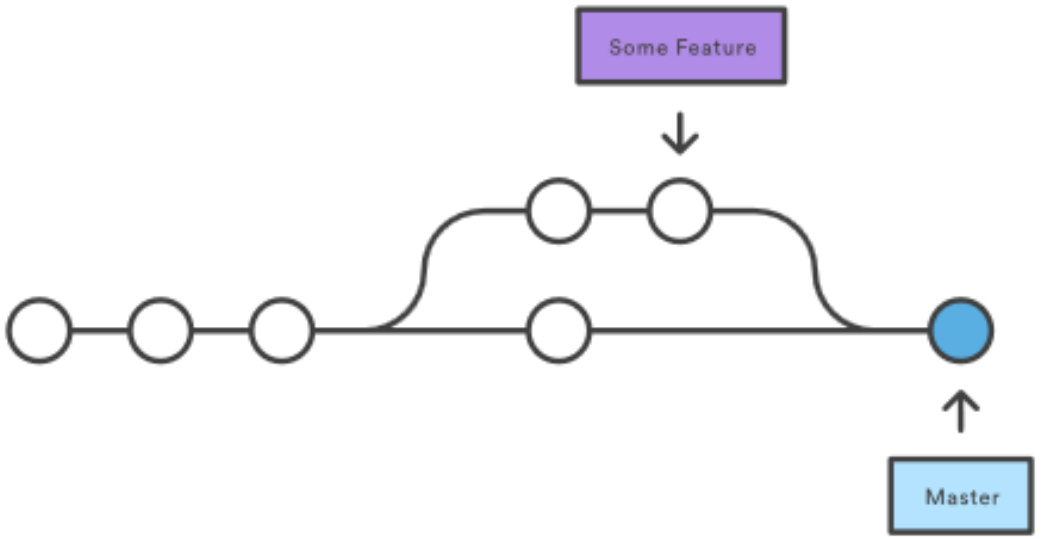
https://habrahabr.ru/post/106912/

Модели ветвления: Github Flow

Github Flow – модель ветвления, являющаяся практической реализацией основных принципов разработки ПО в VCS без учета релизных циклов ПО. Подходит для методологии Kanban.

Принципы

- Одна главная ветка **master** – текущая стабильная версия
- Вспомогательные **feature** или **hotfix** ветки ответвляются от master и сливаются обратно в неё через pull-request
- Все слияния происходят только через pull-request'ы



Плюсы

- Готовый функционал не «отлёживается» в релизах, а сразу же отправляется на пром
- Простая и прозрачная схема работы для разработчика

Минусы

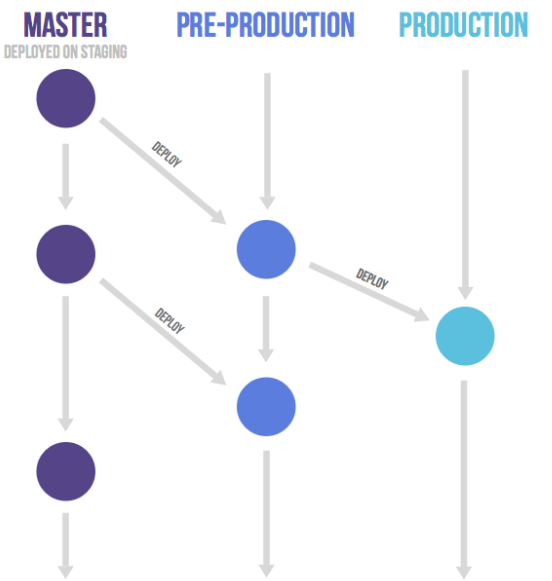
- Требует крайне высокой культуры разработки и сопровождения
- Требует максимальной автоматизации процессов тестирования и доставки

https://guides.github.com/introduction/flow/

https://habrahabr.ru/post/189046/

Модели ветвления: Gitlab Flow

Gitlab Flow – модель, являющаяся некоторым симбиозом Git Flow и Github Flow. Она также проста как и Github Flow, но в то же время позволяет контролировать релиз как Git Flow.



Принципы

- Основная ветка **master** – текущая стабильная версия
- Вспомогательные **feature** или **hotfix** ветки ответвляются от master и сливаются обратно в неё через pull-request
- Стабильная ветка **production** для автоматического деплоя на пром, в которую переносится код из master в тот момент, когда нужно выложиться
- Возможно использование дополнительных веток для различных сред
- Все слияния происходят только через pull-request'ы

Плюсы

- Простая и прозрачная схема работы для разработчика
- Готовый функционал сразу же готов к отправке на пром

Минусы

- Требует крайне высокой культуры разработки и сопровождения
- Требует максимальной автоматизации процессов тестирования и доставки

https://habrahabr.ru/company/softmart/blog/316686/

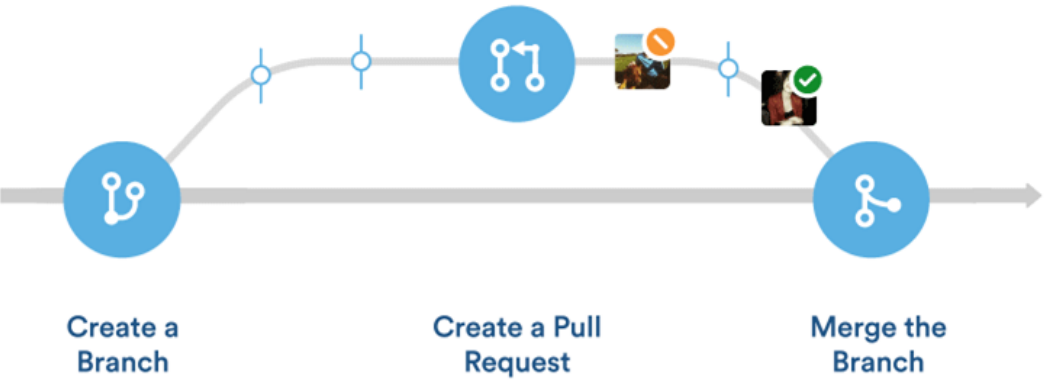
https://about.gitlab.com/2014/09/29/gitlab-flow/

Запросы на слияние (Pull-Request)

Запрос на слияние (Pull-Request) – механизм системы контроля версий, позволяющий оформить изменения из ветки в виде предложения к слиянию в основную (или какую-то иную) ветку репозитория.

Что даёт

- Описание предлагаемого изменения видно в интерфейсе системы контроля версий всем заинтересованным участникам
- Возможность провести code review и оставить комментарии ещё до включения изменений в целевую ветку
- Возможность не допустить слияния, пока не будут выполнены все необходимые условия. Например:
  - Минимальное количество подтверждений от участников, проводящих ревью
  - Успешно прошедшая сборка в системе CI
  - Отсутствие критичных замечаний по результатам автоматического статического анализа



https://confluence.atlassian.com/bitbucketserver/using-pull-requests-in-bitbucket-server-776639997.html