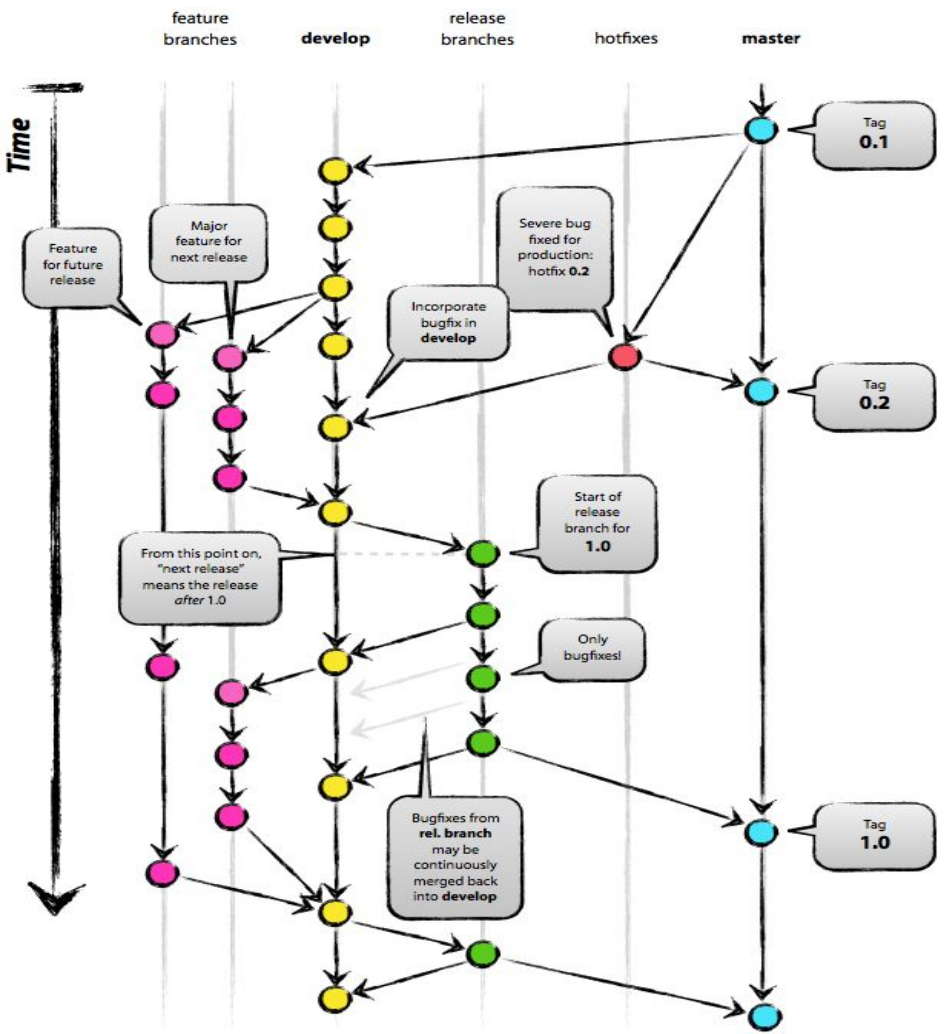
Java

GIT

GIT-FLOW

Git Flow является методологией работы с Git. Это значит, она определяет, какие ветки нужно создать и как производить их слияние



Инициализация GIT FLOW

Первым шагом является создание ветки **develop** от ветки **master**

В этой ветке будет находится вся история проекта, в то время как **master** содержит частичную историю

Остальные разработчики теперь должны клонировать центральный репозиторий и настроить отслеживание для ветки **develop**

Каждая новая функциональность должна разрабатываться в отдельной ветке, которую нужно отправлять (push) в центральный репозиторий (origin) для создания резервной копии/для совместной работы команды

Ветки функций создаются не на основе **master**, а на основе **develop**. Когда работа над новой функциональностью завершена, она вливается назад в **develop**.

Новый код не должен отправляться напрямую в **master**.

Инициализация GIT FLOW. Релиз.

Когда в ветку develop уже слито достаточно нового кода для релиза (или подходит установленная дата предрелиза), от ветки develop создается релизная ветка, например, release/0.3.0

Создание данной ветки означает начало следующего цикла релиза, в ходе которой новая функциональность уже не добавляется, а производится только отладка багов, создание документации и решение других задач, связанных с релизом

Когда все готово, ветка release сливается в master, и ей присваивается тег с версией (r0.3.0). Кроме этого, она должна быть также слита обратно в ветку develop, в которой с момента создания ветки релиза могли добавляться изменения с момента создания ветки релиза

Использование отдельной ветки для подготовки релиза позволяет одной команде дорабатывать текущий релиз, пока другая команда уже работает над функциональностью для следующего релиза

Это также позволяет разграничить этапы разработки. Например, можно сказать: «На этой неделе мы готовимся к версии 1.0.0» и фактически увидеть это в структуре репозитория

Инициализация GIT FLOW. Релиз.

Когда релиз готов к отправке, он сливается в **master** и **develop**, а ветка релиза удаляется (может быть сохранена при продуктовой разработке и необходимости поддержки нескольких релизов). Важно влить release обратно в develop, поскольку в ветку релиза могут быть добавлены критические обновления и они должны быть доступны в дальнейшем. Если ваша команда делает акцент на проверку кода, этот момент идеален для мердж-реквеста (MR, PR, Merge/Pull Request)

Релиз помечается тегом равным его имени в ветке master.

Ветки hotfix

Ветки hotfix используются для быстрого внесения исправлений в рабочую версию кода

Ветки hotfix очень похожи на ветки release и feature, за исключением того, что они созданы от master, а не от develop

hotfix - это единственная ветка, которая должна быть создана непосредственно от master. Как только исправление завершено, ветка hotfix должна быть объединена как с master, так и с develop (или с веткой текущего релиза), а master должен быть помечен обновленным номером версии

Наличие специальной ветки для исправления ошибок позволяет команде решать проблемы, не прерывая остальную часть рабочего процесса и не ожидая следующего цикла подготовки к релизу. Можно говорить о ветках hotfix как об особых ветках release, которые работают напрямую с master

Ключевые идеи

Данная модель отлично подходит для организации рабочего процесса на основе релизов

Git-flow предлагает создание отдельной ветки для исправлений ошибок в продуктовой среде

Модель может быть дополнена использованием Project-ID в названии ветки для интеграции таск-трекера: feature/VK-342-ci

Резюме. Последовательность работы.

1. Из **master** создается ветка **develop**
2. Из **develop** создаются ветки **feature**
3. Когда разработка новой функциональности завершена – **feature** ветка объединяется с веткой **develop**
4. Из **develop** создается ветка **release**
5. Когда ветка релиза готова, она объединяется с **develop** и **master**
6. Если в **master** обнаружена проблема, из нее создается ветка **hotfix**
7. Как только исправление на ветке **hotfix** завершено, она объединяется с **develop** и **master**

Д/з

1.Создать новый пустой каталог и добавить в него пустой текстовый файл (либо это может быть проект на Java - Ваше Д/З)

2.Проинициализировать локальный репозиторий командой **git init**

3.Создать новый **удалённый** репозиторий.

4.Связать созданный локальный репозиторий с удалённым.

5.Загрузить файл из локального репозитория в удалённый командами **git commit** и **git push** (или сделать тоже самое через IDEA)

6.Смоделировать реальную разработку с созданием разных веток по методологии GITFLOW.

7.Проверить выполнение запуском команды **gitk** в командной строке, находясь в корне проекта

8.Приложить скриншот из **gitk** в файл README.md

Источники

1. git-scm.com/downloads
2. Команды GIT и инструкция