**34. Организация совместной разработки в СКВ. Методология GitFlow**

Наиболее популярной СКВ для организации совместной разработки является Git. Ветвление и слияние обычно считаются опасными из-за конфликтов слияния и потому проводятся как можно реже.  
Но с Git эти действия становятся исключительно простыми и дешёвыми, и потому на деле они становятся центральными элементами обычного *ежедневного* рабочего процесса.   
Благодаря своей простоте и предсказуемости, ветвление и слияние больше не являются действиями, которых стоит опасаться. Теперь инструменты управления версиями способны помочь в ветвлении и слиянии больше, чем какие-либо другие.  
Gitflow— это модель которая, по сути, является просто набором процедур, которые исполняет каждый член команды, чтобы все вместе могли достичь высокой управляемости процесса разработки.

Она предполагает выстраивание строгой модели ветвления вокруг релиза проекта, которая дает надежную схему управления крупными проектами.

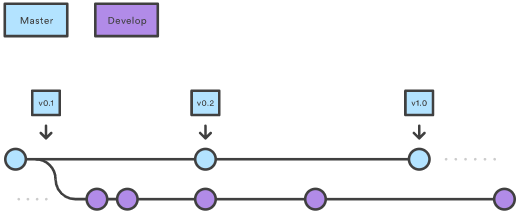
Кроме feature-веток в ней используются отдельные ветки для подготовки, поддержки и записи релиза.

Gitflow использует собственный набор инструментов git-flow, который легко интегрируется с Git, добавляя новые команды Git.

**Начало работы**

Набор инструментов git-flow нужно установить отдельно. После установки git-flow необходимо выполнить команду git flow init. Git-flow является оберткой для Git. Команда git flow init является расширением стандартной команды git init и ничего не меняет в вашем репозитории, кроме того, что создает ветки.

**Как это работает**



**Ветки master и develop**

Вместо использования одной ветки master, в этой модели используется две ветки для записи истории проекта. В ветке master хранится официальная история релиза, а ветка develop служит в качестве интеграционной ветки для новых функций. Также, удобно тегировать все коммиты в ветке master номером версии.

Первым шагом является создание ветки develop от ветки master. Проще всего это сделать одному разработчику, локально создав пустую ветку и отправив ее в центральный репозиторий:

git branch develop

git push -u origin develop

В этой ветке будет находиться вся история проекта, в то время как master содержит частичную историю. Остальные разработчики теперь должны клонировать центральный репозиторий и создать отслеживающую ветку для ветки develop.

При использовании библиотеки расширений git-flow, для создания ветки develop можно выполнить git flow init в существующем репозитории:

$ git flow init

Initialized empty Git repository **in** ~/project/.git/

No branches exist yet. Base branches must be created now.

Branch name **for** production releases: [master]

Branch name **for** "next release" development: [develop]

How to name your supporting branch prefixes?

Feature branches? [feature/]

Release branches? [release/]

Hotfix branches? [hotfix/]

Support branches? [support/]

Version tag prefix? []

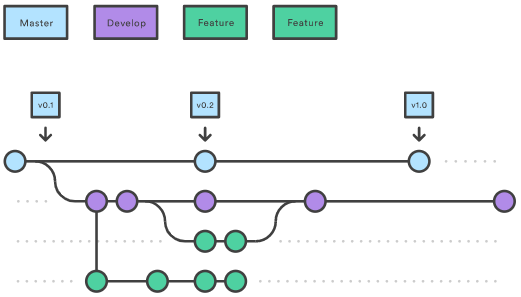
$ git branch

**\*** develop

master

**Ветки для функций (feature branches)**

Каждая новая функциональность должна разрабатываться в отдельной ветке, которую можно отправлять в центральный репозиторий для создания резервной копии/для совместной работы команды. Ветки функций создаются не на основе master, a на основе develop. Когда работа над новой функциональностью завершена, она вливается назад в develop. Новый код не должен отправляться напрямую в master.



Ветки функций объединяются с веткой develop как в модели Feature Branch Workflow. Но на этом работа по схеме Gitflow не заканчивается.

**Создание ветки функции**

Без использования расширений git-flow:

git checkout develop

git checkout -b feature\_branch

При использовании git-flow:

git flow feature start feature\_branch

Далее, продолжайте работу c Git как обычно.

**Окончание работы с веткой**

По окончании разработки новой функциональности следующим шагом следует объединить ветку feature\_branch c develop. Используйте команды:

Без использования расширений git-flow:

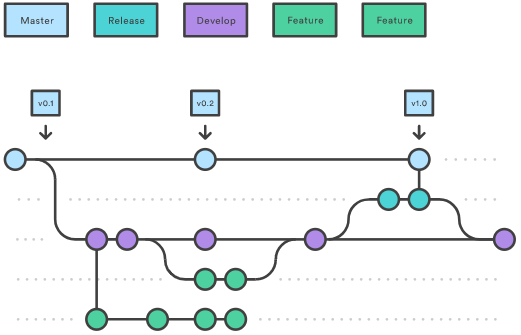
git checkout develop

git merge feature\_branch

При использовании git-flow:

git flow feature finish feature\_branch

**Ветки релиза**



Когда в ветку develop уже слито достаточно нового кода для релиза (или подходит установленная дата предрелиза), от ветки develop создается ветка release. 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. 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, легко сказать: «На этой неделе мы готовимся к версии 4.0» и фактически увидеть это в структуре репозитория).

Создание веток релиза – это еще одна простая операция ветвления. Как и ветки функций, ветки релизов основаны на ветке develop. Новая ветка release может быть создана с использованием следующих команд:

Без использования расширений git-flow:

git checkout develop

git checkout -b release/0.1.0

При использовании git-flow:

$ git flow release start 0.1.0

Switched to a new branch 'release/0.1.0'

Когда релиз готов к отправке, он сливается в master и develop, а ветка релиза удаляется. Важно влить ее обратно в develop, поскольку в ветку release могут быть добавлены критические обновления, и они должны быть доступны для новых функций. Если ваша команда делает акцент на проверку кода, этот момент идеален для пул-реквеста.

Для завершения работы на ветке релиза, используйте следующие команды:

Без использования расширений git-flow:

git checkout develop

git merge release/0.1.0

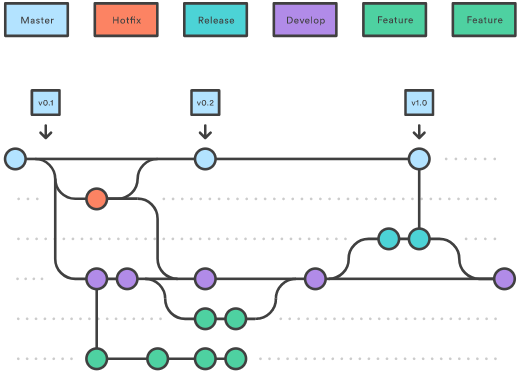
Или при использовании git-flow:

git checkout master

git checkout merge release/0.1.0

git flow release finish '0.1.0'

**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. 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 relese, которые работают напрямую с master. Ветка hotfix может быть создана с помощью следующих методов:

Без использования расширений git-flow:

git checkout master

git checkout -b hotfix\_branch

Или при использовании git-flow:

$ git flow hotfix start hotfix\_branch

Как и в работе с веткой release, ветка hotfix объединяется как с master, так и с develop.

git checkout master

git merge hotfix\_branch

git checkout develop

git merge hotfix\_branch

git branch -D hotfix\_branch

$ git flow hotfix finish hotfix\_branch

**Пример**

Пример команд, демонстрирующий полный цикл работы с веткой функции, выглядит следующим образом. Предположим, что у нас есть репозиторий с веткой master.

git checkout master

git checkout -b develop

git checkout -b feature\_branch

*# работа ведется на ветке feature*

git checkout develop

git merge feature\_branch

git checkout master

git merge develop

git branch -d feature\_branch

Помимо ветки функции и release, приведем пример создания ветки hotfix:

git checkout master

git checkout -b hotfix\_branch

*# работа сделана, коммиты добавлены в hotfix\_branch*

git checkout develop

git merge hotfix\_branch

git checkout master

git merge hotfix\_branch

**Ключевые идеи, которые нужно запомнить о Gitflow:**

* Данная модель отлично подходит для организации рабочего процесса на основе релизов.
* Gitflow предлагает создание отдельной ветки для исправлений ошибок в продуктовой среде.

**Последовательность работы при использовании модели Gitflow:**

1. Из master создается ветка develop.
2. Из develop создаются ветки feature.
3. Когда разработка новой функциональности завершена, она 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.