**Создание проекта для каждого члена команды разработчиков** (для тех, у кого ещё нету локальной копии проекта)

Для начала работы с центральным репозиторием, следует создать копию оригинального проекта со всей его историей локально.

переходим в нужную директорию (с помощью cd)

Клонируем репозиторий, используя протокол http:

git clone https://github.com/Interbellum/collective\_development.git

Теперь в нашей директории находится полная копия проекта.

Переходим в директорию ~/collective\_development/

Вводим

git branch

(перечисляет существующие ветки и отмечает активную(на которой вы сейчас находитесь)

Теперь у нас есть локальная копия репозитория, теперь можно приступить непосредственно к разработке.

**Создание личной интеграционной ветки для каждого члена команды разработчиков.**

Открывем консоль и Переходим в нашу директорию.

cd ~/collective\_development/

git checkout master

(перешли на ветку master)

git pull

(обновляем все локальные файлы с удаленного репозитория)

Теперь нам надо отрастить собственную ветку (Личную)

git checkout –b <name\_branch>

( например apop\_int что значит apop инициалы человека из таблицы P\_6.4 на стр 9 <https://github.com/Interbellum/collective_development/blob/master/Docs/Project_Plan%20(PPL)/Project_plan%20(%20P%20).doc>, и смысл ветки int (integration))

git push origin apop\_int

(расшариваем если нам нужно нашу ветку)

git merge master

(мерджим нашу ветку с master)

**Стандартная работа с гитом любого члена команды разработчиков.**

Допустим на нас завели тикет #150 с задачей “разработать Quality Requariments”(или выписываем его на себя сами)

Открывем консоль и Переходим в нашу директорию

cd ~/collective\_development/

git checkout apop\_int

(переходим на свою ветку, у каждого само собой собственная ветка, для примера взяли apop\_int)

git pull

(обновляем все локальные файлы с удаленного репозитория)

Теперь отращиваем веточку от нашей собственной ветки.

git checkout –b apop\_int#150

( называем ее по номеру тикета, который будем решать на нашей ветке)

Далее работаем. Изменяем файлы, создаем новые и тд...

После работы нужно применить изменения, для этого командой add добавляем все измененные файлы

git status

git add file.py fil2.pe file3.py

git status

git commit –m “#150”

git status

(вводим в кавычках комментарии с кратким описанием причиной коммита)

git checkout apop\_int

(переходим на нашу «старшую» ветку)

git merge apop\_int#150

(применяем изменения с ветки apop\_int#150 на нашу apop\_int)

git push

(отправляем изменения на удаленный репозиторий)

git branch –d PA\_QR#150

(удаляем нашу временную ветку, где мы решали нашу задачу по данному тикету)

Переводим наш тикет на интегратора с целью проинтегрировать наши изменения. В тикете явно указываем на цель тикета и название вашей ветки на которой хранятся изменения.

Процесс интегрирования (работа интегратора)

(выше описана основная идея работы с гитом, здесь алгоритм будет в основном тот же)

Допустим на нас завели тикет на интеграцию номер #111. Проинтегрировать изменения члена команды akov с ветки akov\_int.

Открывем консоль и Переходим в нашу директорию

cd ~/collective\_development/

git checkout apop\_int

(переходим на свою ветку, у каждого само собой собственная ветка, для примера взяли apop\_int)

git pull

git merge akov\_int

(применяем изменения с ветки akov\_int на нашу apop\_int)

Делаем Sanity тесты.

Проверяем, что наш проект работает, основные функции.

Создаем exel файл Sanity\_test\_#111 в папке SanityTests по адресу <https://github.com/Interbellum/collective_development/tree/master/Docs/SanityTests> где записываем все результаты тестов (PASS or FAIL).

Если тесты не прошли, и наш проект упал, или что то сломалось, значит переназначаем наш тикет на интеграцию обратно тому человеку, который просил интегрировать его изменения.

В тикете четко описываем проблему и условия при которых у нас возникла проблема с ссылкой на Sanity\_test\_#111.

Если Sanity тесты прошли успешно, и наш проект работает как ожидается

git push

(отправляем изменения на удаленный репозиторий)

git checkout master

git merge apop\_int

git push

Переводим наш тикет на тестеровщиков .

Возникновение конфликта.

Конфликт может возникнуть в следущей ситуации, если мы отрастили ветку «ветка1» от Master для решения какого то тикета.

Не закончив работу над этим срочно пришлось отрастить ещё одну ветку «ветку2» для решения второго тикета.

Сделали срочную работу над тикетом2, произвели слияние.

Начали работать над первым тикетом, на ветке1, но теперь мастер указывает на другой коммит, нежели до начала работы над данным тикетом.

 Если вы изменили одну и ту же часть файла по-разному в двух ветках, которые собираетесь слить, Git не сможет сделать это чисто. Если ваше решение проблемы одного тикета изменяет ту же часть файла, что и другой тикет , вы получите конфликт слияния.

Git не создаст новый коммит для слияния и выдаст ошибку.

git status – покажет вам информацию об конфликте.(укажет в каком файле произошел конфликт(ы)

Ваш файл в котором произошел конфликт содержит секцию, которая выглядит примерно так:

<<<<<<<Head:file.py -ниже до знаков ===== находится версия кода на котором вы были, так как именно на неё вы перешли перед выполнением команды merge

«Сдесь текст кода, который был в файле “file.py” на ветке head

==========

«сдесь текст кода, который был на второй ветке»

>>>>>>>>>>>PA\_#150 file.py название файла тот же файл только со второй ветки

Для того чтобы исправить нужно заменить весь текст что после «<» и до последнего знака «> « на какой то из этих двух вариантов кода, или сделать другой вариант кода.

*Но это уже дело не интегратора, так что заводим тикет с полным описанием и указанием проблемы на человека, кто работал над этим файлом.*

После того как решена проблема с конфликтами , отмечаем исправленный файл командой add fily.py и так для каждого конфликтного файла. Индексирование данных файлов будет означать что конфликты исправлены.

Можете выполнить git status ещё раз, чтобы убедиться, что все конфликты были разрешены.

Если вы довольны тем, что получили, и удостоверились, что всё, имевшее конфликты, было проиндексировано, можете выполнить git commit для завершения слияния

Дополнительный список интернет источников которые полезно изучить( описывается тот же процесс более детально и с картинками):

1. <http://git-scm.com/book/ru/%D0%92%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5-%D0%B2-Git-%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-%D0%B2%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F-%D0%B8-%D1%81%D0%BB%D0%B8%D1%8F%D0%BD%D0%B8%D1%8F>
2. <http://www.calculate-linux.org/main/ru/git>
3. <http://habrahabr.ru/post/106912/>