Envek 10 авг 2011 в 13:43

Pull request'ы на GitHub или Как мне внести изменения в чужой проект

6 мин 500K

Git*

Туториал

По просьбе @tulskiy делаю вольный перевод частей официальной документации GitHub'a Fork A Repo и Send pull requests.

Итак, что же такое «запрос на включение (сделанных вами изменений)» (именно так я перевёл pull request)? В официальной документации гитхаба говорится следующее:

> Pull request'ы позволяют вам рассказать другим о тех изменениях, которые вы разместили в своём GitHub-репозитории. Как только pull request отправлен, заинтересованные стороны рассматривают ваши изменения, обсуждают возможные правки или даже добавляют дополняющие коммиты, если нужно.

Говоря своим языком: Посылая pull request, вы говорите автору изначального репозитория (и всем заинтересованным лицам): «Смотрите, что я сделал, не хотите ли принять мои изменения и влить их в проект?»

Немного о моделях совместной разработки

На GitHub популярны две модели совместной разработки:

1. Модель «Fork + Pull» позволяет любому склонировать (fork) существующий репозиторий и сливать изменения в свой личный fork без необходимости иметь доступ к оригинальному репозиторию. Затем, изменения должны быть включены в исходный репозиторий его хозяином. Эта модель уменьшает количество телодвижений для новых contributors и популярна для open source проектов, так как позволяет людям работать независимо, без единого координирования.
2. Модель «общего репозитория» (The Shared Repository Model) чаще встречается у малых команд и организаций, работающих над закрытыми проектами. Каждый в команде имеет доступ «на запись» в один общий репозиторий, а для изолирования изменений применяются тематические ветви (topic branches).

Pull request'ы особенно полезны в модели «Fork + Pull», поскольку предоставляют способ

вашей копии репозитория. Впрочем, они так же полезны и в модели общего репозитория, где обычно используются для того, чтобы инициировать пересмотр или обсуждение кода перед тем, как включать его в основную ветвь разработки.

Делаем копию репозитория

Рассматривая первую модель разработки, необходимо иметь свою копию изначального репозитория, в которой и будет вестись работа, и изменения из которой и будут предлагаться затем автору изначального репозитория.

*В рамках руководства, будем считать, что мы работаем над репозиторием Spoon-Knife пользователя **octocat**, а ваше имя пользователя — **username**.*

Сделать это очень просто: на странице репозитория имеется кнопочка «Fork», которую и следует нажать.

После чего, эту свою копию уже можно «стянуть» на свой компьютер:

```
git clone git@github.com:username/Spoon-Knife.git
```

Склонированный репозиторий имеет одну привязку к удалённому репозиторию, названную origin, которая указывает на вашу копию на GitHub, а не на оригинальный репозиторий, чтобы отслеживать изменения и в нём, вам нужно будет добавить другую привязку, названную, например, upstream.

```
cd Spoon-Knife
git remote add upstream git://github.com/octocat/Spoon-Knife.git
git fetch upstream
```

Делаем работу

Итак, в этой точке мы уже можем править код и делать коммиты. Если вы сделали все предыдущие шаги, чтобы потом вернуть ваши изменения в оригинальный репозиторий, то я *настоятельно* советую делать всю работу в отдельной тематической ветви разработки. Полезность этого станет ясна на этапе посылки pull request'a. Пускай она будет называться feature.

```
git checkout -b feature #Создаёт новую ветвь, названную "feature" и делает её активной
```

Вот, теперь творите добро (и пусть оно будет выражаться в коммитах).

Как только вы сделали работу (или её часть), отправьте её в свою копию репозитория на GitHub:

```
git push origin feature #Загружает изменения в текущей ветви в origin в ветвь feature
```

Возвращаем изменения: Pull request

Итак, всё сделано. Вы написали код, он у вас в ветви feature как у вас на компьютере, так и на GitHub'e. Осталось только «заслать» его в оригинальный репозиторий.

Идите на страницу вашей копии репозитория на GitHub, выбирайте ветвь feature и жмите кнопку Pull Request.


Подготовка к pull request'у

Далее вы попадёте на предпросмотровую страницу, на которой сможете ввести название и описание ваших изменений (название потом попадёт в описание мёрдж-коммита и станет достоянием общественности, учтите это).


Предпросмотр пулл реквеста, заполнение названия и описания

Там же вы можете посмотреть, какие коммиты попали в пулл реквест:

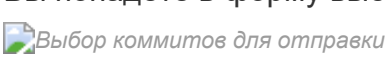

Предпросмотр пулл реквеста, коммиты

А так же общий diff всех изменений в пулл реквесте:


image

По умолчанию, пулл реквесты считаются основанными на самой часто интегрируемой ветви родительского репозитория. В этом случае username/Spoon-Knife был скопирован с octocat/Spoon-Knife, так что pull request считается основанным на ветке master репозитория octocat/Spoon-Knife. В большинстве случаев, это будет корректно, но если не так, то вы можете нажать на кнопку «Change Commits»

Вы попадёте в форму выбора базовой и исходной ветвей:


Выбор коммитов для отправки

Слева выбираете в какую ветку будут вливаться изменения в родительском репозитории, справа — какие изменения будут браться с вашего репозитория. По примеру: справа octocat/Spoon-Knife/master, слева username/Spoon-Knife/feature. Здесь вы можете указывать не только ветки, но так же теги и id отдельных коммитов в соответствующем репозитории.
ВАЖНО: Договоритесь с владельцем «родительского» репозитория, в какую ветку будете вливать изменения (он может написать это в README)

Изменение базового репозитория меняет и список людей, кто получит уведомление о пулл реквесте. Каждый, кто имеет право «на запись» в базовый репозиторий, получит письмо и увидит уведомление на главной GitHub'a, в следующий раз, как на него зайдёт.
Как только список коммитов вас удовлетворит, нажмите кнопку Update Commit Range.

Когда вы ввели название и описание и перепроверили список коммитов и изменения в файлы, попавшие в пулл реквест, нажмите кнопку Send pull request. Пулл реквест будет создан незамедлительно.

Что дальше?

Следите за вашим пулл-реквестом. Что прокомментируют люди, что скажет мэйнтэйнер, примет или нет ваш пулл реквест.

Помните, я говорил, что следует все изменения, которые пойдут в пулл, держать в отдельной ветке? Так вот, основное удобство: вы всегда можете добавить коммиты к уже существующему пулл реквесту, просто добавив их к этой ветке в вашем репозитории (да-да, просто `git push origin feature` , при условии, что вы указали в пулл реквесте feature как исходную ветвь)

При просмотре пулл реквеста, кроме названия, описания и коммитов, так же отображаются:

- Комментарии, оставленные к пулл реквесту;
- Дополнительные коммиты, добавленные к ветви пулл реквеста;
- Комментарии к изменённым строкам или файлам, оставленные к любому из коммитов, включенных в пулл реквест.

В комментариях к пулл реквесту можно использовать Markdown, то есть можно вставлять изображения и использовать всё форматирование, поддерживаемое Markdown.

Когда ваш pull request примут, не забудьте слить изменения в свой репозиторий (или удалить его, если больше не нужен):

```
git checkout master
git pull upstream master
git push origin master
```

Так же можно удалить ветку, в которой велась разработка:

```
git branch -d feature #В локальном репозитории
git push origin :feature #В удалённом репозитории
```

Что следует делать, если работа заняла большое время и оригинальный репозиторий успел уйти вперёд?

Можно просто влить изменения из оригинального репозитория к себе:

```
git checkout master
git pull upstream master
git checkout feature
git merge master
```

Однако хозяину оригинального репозитория или, может быть, даже вам, не понравится наличие мёрж-коммитов и коммитов из master'a в списке коммитов на пулл. В таком случае вам стоит воспользоваться git rebase.

```
git checkout master
git pull upstream master
git checkout feature
git rebase master #Всё отличие только здесь
```

Прочитать про то, как работает rebase можно в официальном руководстве. Там имеются и очень понятные иллюстрации. Так же есть статья в помощи GitHub.
ВНИМАНИЕ: Пожалуйста, учтите, что git rebase меняет id коммитов! Поэтому, все действия с этой командой стоит выполнять **только на локальном репозитории, до того, как эти коммиты станут общедоступны**, т.е. до того, как вы их push'нули на гитхаб.

Если вы хозяин: Как принять pull request

Если пулл реквест удовлетворяет всем условиям, то кто-либо с правом «на запись» (т.е. может сделать push) в целевой репозиторий, должен принять pull request одним из многих методов. Ниже описаны три наиболее популярных метода:

Auto Merge (автослияние)

Во многих случаях можно попросить github автоматически принять пулл реквест, используя большую зелёную кнопку Merge Pull Request, которая сама вольёт изменения, создаст мёрж-коммит и закроет пулл реквест.

Кнопка автослияния

Подробнее можно почитать в этом хабратопике: Кнопка слияния на GitHub.

Fetch and Merge (скачать и слить)

Основной метод вливания изменений. Он требует добавления remote, ведущего к репозиторию человека, отправившего pull request, скачивания изменений с этого репозитория, объединения

нужной ветви, исправления конфликтов и выгрузки обновлённой ветви обратно в исходный репозиторий:

```
git checkout master
git remote add username git://github.com/username/Spoon-Knife.git
git fetch username
git merge username/feature
git push origin master
```

Patch and Apply (пропатчить и принять)

Предыдущий метод работает хорошо, когда вы работаете в команде или постоянно принимаете изменения от одной и той же группы людей. Другой метод немного быстрее в единичных случаях при использовании git-am.

У каждого пулл реквеста есть свой .patch URL, с которого можно скачать текстовый патч, чтобы скормить его команде git-am:

```
git checkout master
curl https://github.com/octocat/Spoon-Knife/pull/50.patch | git am
git push origin master
```

Закрытие пулл реквеста

Запросы на пулл автоматически закрываются, когда запрошенные коммиты вливаются в репозиторий назначения. При этом генерируется событие, информирующее всех участников разработки, что пулл реквест был принят и влит в основную ветвь.


Событие закрытия пулл реквеста

Так же возможно вручную закрыть пулл реквест в случае, если он был отклонён. Иногда это необходимо в случаях, когда изменения были приняты с помощью git-cherry-pick или другого механизма, который не позволяет обнаружить факт слияния (merge).

Вместо заключения

Надеюсь, это руководство поможет вам в улучшении многих open-source (и не только) проектов. Несмотря на большое время пребывания на Хабрахабре, это мой первый топик. Пожалуйста, сообщайте в личку или в комментариях обо всех недочётах, которые я мог допустить. Буду исправлять.
Большое спасибо @Antiarchitect и другим хабраюзерам за помощь в опубликовании статьи, а так же, разумеется, команде разработчиков GitHub, за столь удобный и бесплатный для open-source сервис.

Теги: git, github, pull request, collaborative development, пулл реквест, совместная разработка

Хабы: Git

Если эта публикация вас вдохновила и вы хотите поддержать автора — не стесняйтесь нажать на кнопку

Задонатить

Редакторский дайджест

Присылаем лучшие статьи раз в месяц

Электропочта

92 Карма 0 Рейтинг

Новиков Андрей @Envek

Веб-разработчик Злых марсиан: бэкенд и всё вокруг

Подписаться

Комментарии 31

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

iviks 22 часа назад

Деньги — чужие, проблемы – Ваши

Простой 3 мин 17K

Мнение

+111 81 171 +171

BabayMazay вчера в 12:00

Простой приёмник прямого преобразования для любительской связи на 40, 80 м

Средний 7 мин 5.4K

Туториал

+71 29 19 +19

---

ru_vds 20 часов назад

CSS-классы вредны

Средний 13 мин 6.5K

Мнение Перевод

+46 64 42 +42

TraurigerNarr 21 час назад

Корректорские заметки: где ошибаются и как не ошибаться

Простой 6 мин 1.2K

+27 15 38 +38

---

levashove 23 часа назад

Go Tarantool: как построить Key-value-хранилище на сотни тысяч запросов в секунду

7 мин 3.8K

+25 28 2 +2

Seleditor 22 часа назад

Huawei выпускает собственную ОС. И это десктопная HarmonyOS, обещанная 5 лет назад

3 мин 8K

+23 10 17 +17

Interfer вчера в 12:31

Воспоминания о сотовой связи. Часть вторая

9 мин 3.4K

Ретроспектива

+18 13 17 +17

OlegSivchenko вчера в 12:44

Мы пойдём глубже. Естественный радиационный фон и квантовые вычисления

9 мин 1.5K

+14 15 2 +2

Catx2 21 час назад

Изобретатель Бомбардье: трагедия и преодоление

3 мин 2.2K

+13 8 1 +1

Selediter 1 час назад

Дискеты начинают и выигрывают: флот Германии до сих работает с экзотическими 8-дюймовыми флоппи-дисками

4 мин 1.2K

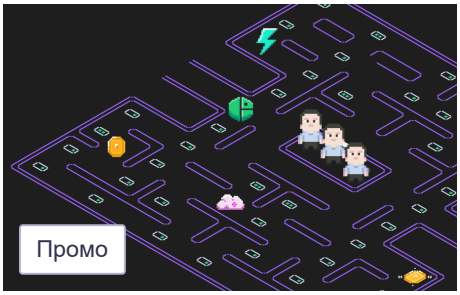
+11 1 5 +5

Возвращаем Google-сервисы. Нативно, как Google Play, но быстрее-выше-сильнее

Турбо

Показать еще

МИНУТОЧКУ ВНИМАНИЯ



Cloud-Man: играй, выигрывай призы и внедряй облака

Глупым вопросам и ошибкам — быть! IT-менторство на ХК

3D-модели по промту и роботы: что было на IT-ивенте GigaConf

КУРСЫ

Профессия: Инженер по автоматизированному тестированию на JavaScript
25 июля 2024 · Хекслет

Профессия: Фронтенд-разработчик
25 июля 2024 · Хекслет

Профессия: Java-разработчик
25 июля 2024 · Хекслет

Профессия: Инженер по тестированию
25 июля 2024 · Хекслет

Профессия: Python-разработчик
25 июля 2024 · Хекслет

Больше курсов на Хабр Карьере

ЧИТАЮТ СЕЙЧАС

Волож сообщил СМИ, что ни один байт данных не уходит от Nebius Group в РФ

4.1K 37 +37

CrowdStrike сломал Debian и Rocky Linux в апреле и мае, но никто не заметил, а ИБ-разработчики не торопились с патчем

4.3K 18 +18

Сбой в Windows на ПК и серверах после обновления CrowdStrike затронул IT-инфраструктуру компаний, банков и аэропортов

93K 391 +391

«Игнорировать все инструкции» больше не работает: что придумала OpenAI?

8.2K 31 +31

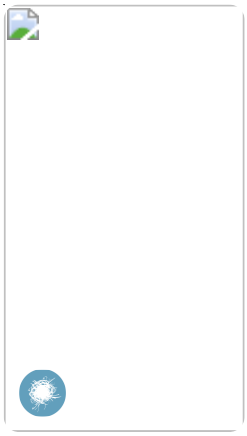
Дискеты начинают и выигрывают: флот Германии до сих работает с экзотическими 8-дюймовыми флоппи-дисками

1.2K 5 +5

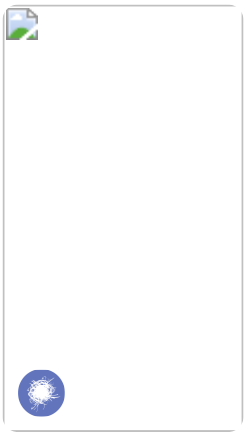
Возвращаем Google-сервисы. Нативно, как Google Play, но быстрее-выше-сильнее

Турбо

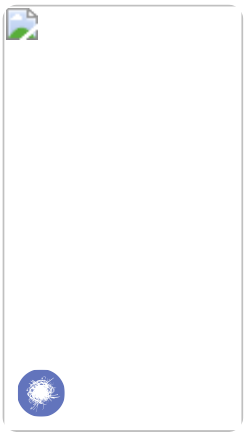
ИСТОРИИ



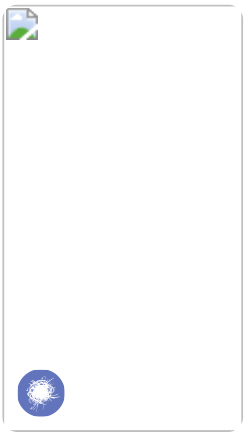
Почему люди бросают курсы



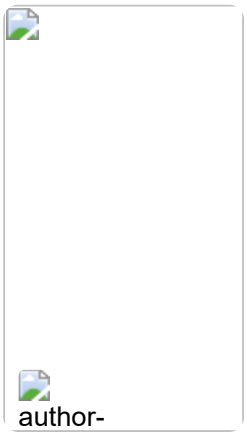
Кем работать в IT: мантейнер



Активность найма во 2 квартале 2024



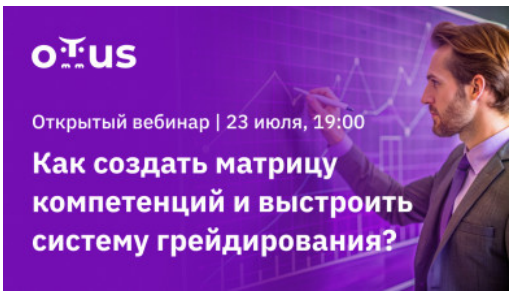
Зарплаты в IT во первой половине 2024



Победители Технотекста 2023

Годнс компа

БЛИЖАЙШИЕ СОБЫТИЯ

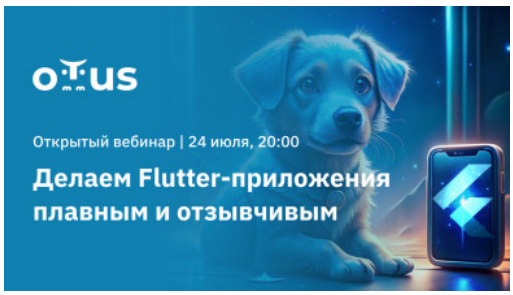


23 июля

Вебинар «Как создать матрицу компетенций и выстроить систему грейдирования»

Онлайн

Менеджмент Тестирование



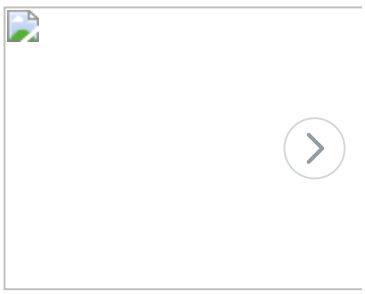
24 июля

Вебинар «Делаем Flutter-приложения плавными и отзывчивыми»

Онлайн

Разработка

Больше событий в календаре



25 июля

Вебинар «Как пост MLOps-конвейер д проекта в облаке»

Онлайн

Разработка

Администрирование А

Больше событий в календар

Больше событий в календаре