# Совместная разработка в команде на GitHub.

GitHub стал краеугольным камнем для всего программного обеспечения с открытым исходным кодом. Разработчики любят его, сотрудничают с ним и постоянно создают новые великолепные проекты с помощью него. Помимо хостинга вашего кода, главная привлекательность GitHub заключается в использовании его в качестве инструмента совместной работы. В этом уроке давайте рассмотрим некоторые из наиболее полезных функций GitHub, особенно полезных для работы в командах, что делает его еще более эффективным, продуктивным и, самое главное, забавным!

## Github разработка в команде

В этом руководстве предполагается, что вы уже знакомы с Git, распределенной системой управления версиями с открытым исходным кодом, созданной Линусом Торвальдсом в 2005 году. Если вам нужна ревизия или поиск в Git, посетите наш предыдущий курс скринкастов или даже несколько статей на эту тему. Кроме того, у вас уже должна быть учетная запись Github, а также некоторые базовые функции, такие как создание репозитория и внесение изменений в Github. Если нет, обратитесь к предыдущим учебникам.

В мире разработки при создании своего проекта работа в команде будет неизбежной. В этом руководстве по совместной разработке на Github мы изучим некоторые из наиболее распространенных инструментов, которые нам обычно нужны при работе с командами разработчиков программного обеспечения. Обсуждаемые инструменты:

1. **Добавление членов команды** - организация и соавторы
2. **Pull Requests** - Отправка и слияние

## Инструмент 1: Добавление членов команды

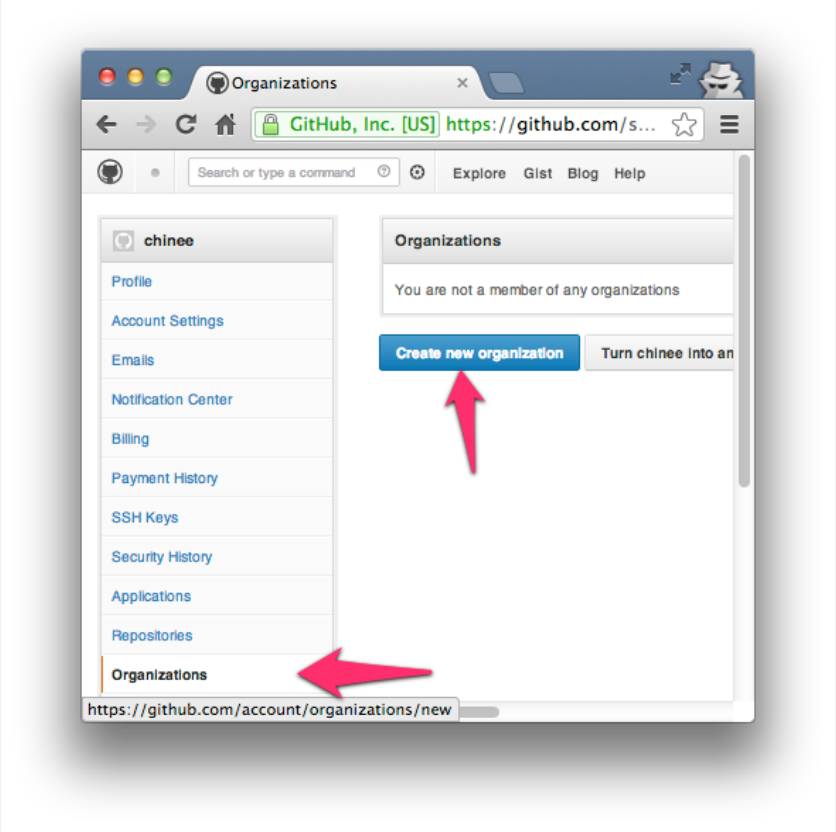
Как правило, существует два способа настройки Github для совместной работы:

1. **Организации**. Владелец организации может создавать множество команд с разными уровнями доступа для различных репозиториев
2. **Сотрудники**. Владелец репозитория может добавлять коллабораторов с доступом Read + Write для одного репозитория

### Organizations

Если вы контролируете несколько команд и хотите установить разные уровни доступа для каждой команды с различными членами и добавить каждого участника в разные репозитории, то организация будет для вас наилучшим вариантом. Любая учетная запись пользователя Github уже может создавать

бесплатные организации для репозиториев с открытым исходным кодом. Чтобы создать организацию, просто перейдите на страницу настроек своей организации:



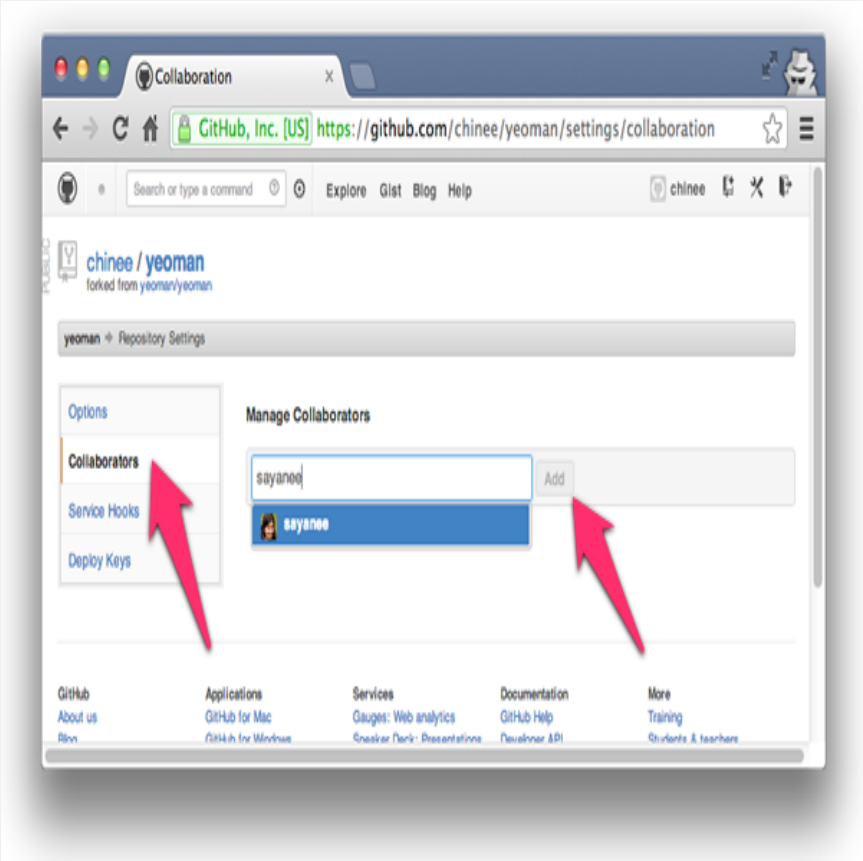

Чтобы получить доступ к странице команд для вашей Организации, вы можете просто перейти на `http://github.com/organizations/[organization-name]/teams`, чтобы просмотреть их или даже посетить `https://github.com/organizations/[organization-name]/teams/new` Для создания новых команд с тремя уровнями доступа, такими как:

1. **Pull Only**: выборка и слияние с другим репозиторием или локальной копией. Доступ только для чтения.
2. **Push and Pull**: (1) наряду с обновлением удаленного репозитория. Читайте + Запись.
3. **Push, Pull & Administrative**: (1), (2) наряду с правами на информацию о выставлении счетов, созданием команд, а также удаление аккаунтов организации. Чтение + запись + доступ администратора


## Соавторы

Коллабораторы (соавторы) используются для предоставления возможности **«читать + писать»** в один репозиторий, принадлежащий личной учетной записи. Чтобы добавить Collaborators (другие личные учетные записи Github), перейдите на страницу

`https://github.com/[username]/[repo-name]/settings/collaboration`:




После этого каждый соавтор увидит изменение статуса доступа на странице репозитория. После того, как у нас есть доступ на запись к репозиторию, мы можем сделать `git clone`, поработать над изменениями, сделать `git pull` для извлечения и слияния любых изменений в удаленном репозитории и, в конечном счете, `git push`, для обновления удаленного репозитория с собственными изменениями:


---

# Инструмент 2: Pull Requests

Pull Requests - отличный способ внести свой вклад в репозиторий, сделав его форк. В конце дня, если мы хотим, мы можем отправить pull request владельцу репозитория, чтобы объединить наши изменения кода. Сам pull request может включать обсуждение качества кода, функций или даже общей стратегии.

Давайте теперь рассмотрим основные шаги для pull request.

### Инициирование pull request

В Github есть две модели для pull request:

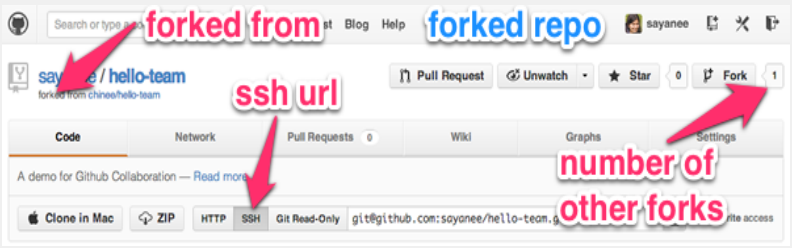
> **Модель Fork & Pull** - используется в общедоступном репозитории, на который у вас нет push-доступа

**Share Repository Model** - используется в частном репозитории, на который у нас есть push-доступ. В этом случае форк не требуется.

Здесь мы видим рабочий процесс между двумя пользователями (`repo-owner` и `forked-repo-owner`) для модели Fork and Pull:

Определите репозиторий Github, в который вы хотите внести свой вклад, и нажмите кнопку «Fork», чтобы создать клон репозитория в вашей собственной учетной записи Github:


Это создаст точную копию репозитория в вашем собственном аккаунте




Выберите URL-адрес SSH, чтобы он запрашивал вашу парольную кодовую фразу SSH вместо имени пользователя и пароля каждый раз, когда вы делаете `git push` или `git pull`. Затем мы будем клонировать этот репозиторий на наш локальный компьютер:

```
$ git clone [ssh-url] [folder-name]

$ cd [folder-name]
```

Как правило, мы создадим новую ветку git для каждой новой задачи. Это хорошая практика, потому что в будущем, если мы продолжим обновление ветки после некоторых обсуждений, запрос на перенос будет автоматически обновляться. Давайте создадим новую ветку, чтобы внести очень простое изменение в файл `readme.md`:

```
$ git checkout -b [new-feature]
```

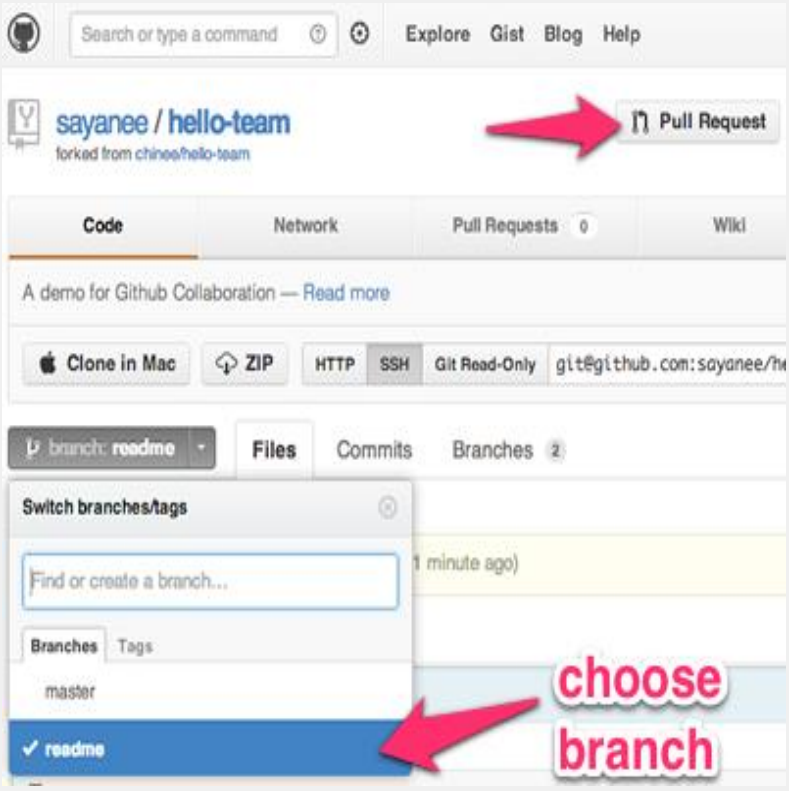
После внесения соответствующих дополнений для создания новых функций мы просто передадим новые изменения и проверку в ветку git master:

```
$ git add .

$ git commit -m "information added in readme"

$ git checkout master
```

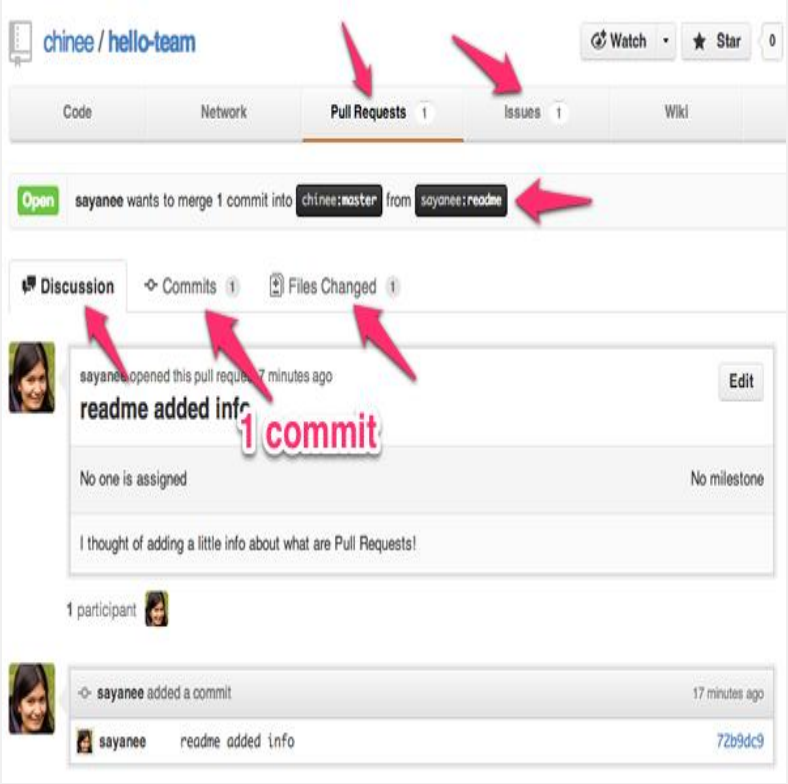
На этом этапе мы запушим ветку в удаленный репозиторий. Для этого мы сначала перейдем на ветку с новой задачей, а так же на псевдоним для удаленного репозитория. Затем мы будем пушить изменения с помощью `git push [git-remote-alias] [branch-name]`:

```
$ git branch

* master

readme

$ git remote -v

origin

git@github.com:[forked-repo-owner-username]/[repo-name].git (fetch)

origin

git@github.com:[forked-repo-owner-username]/[repo-name].git (push)

$ git push origin readme
```

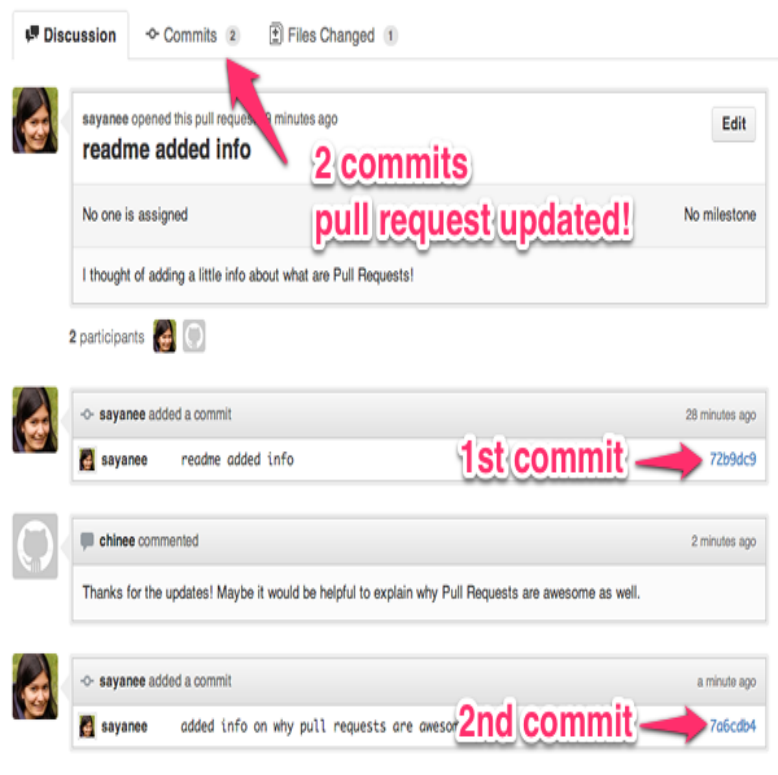
На нашей развязанной странице Github репозитория мы перейдем к ветке с новой функцией, а затем нажмите кнопку «Pull Request».




После отправки пулреквеста он напрямую приведет нас к странице запроса в исходном репозитории. Мы увидим наш запрос на pull.




После обсуждения возможно, что владелец форкнутого репозитория может захотеть добавить изменения в новую функцию. В этом случае мы выберем одну и ту же ветку на нашей локальной машине, зафиксируем ее и запушим ее обратно на Github. Когда мы заходим на страницу запроса в оригинальном репозитории, он будет автоматически обновляться!

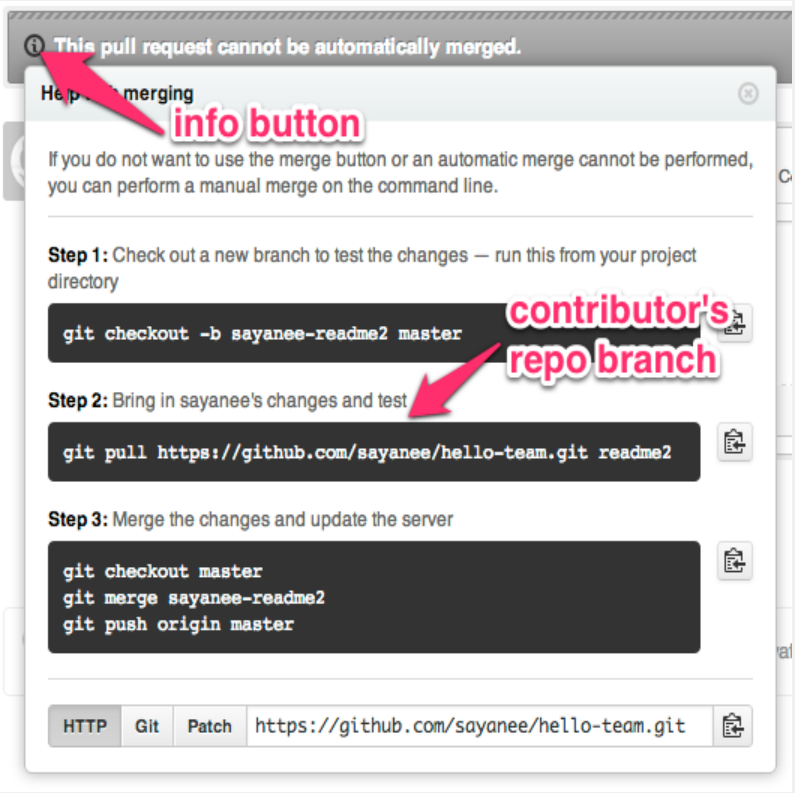



## Слияние пул реквеста

Если вы являетесь владельцем оригинального репозитория, существует два способа слияния входящего пул реквеста:

**Слияние непосредственно на Github**: если мы делаем слияние непосредственно на Github, то убедитесь, что нет конфликтов, и реквест готов к объединению в ведущую ветку. Владелец оригинального хранилища может просто щелкнуть зеленую кнопку «Слить запрос», чтобы сделать это:


**Слияние на наших локальных машинах**: в других случаях могут возникнуть конфликты слияния, и после нажатия кнопки «Информация» у Github будут четкие инструкции о том, как мы можем объединить ветвь на нашей локальной машине, потянув за изменения из ветви контрибьютера:




Существуют различные модели создания веток, используемые для управления версиями в командах разработки программного обеспечения. Вот две популярные модели рабочего процесса git: (1) рабочий процесс Github, который имеет простую ветвящуюся модель и использует запросы на pull, и (2) Gitflow, который имеет более обширное разветвление. Модель, которая в конечном итоге будет выбрана, определенно будет меняться в зависимости от команды, проекта и ситуации.