# Подготовка к работе

## Тимлид

1. Создаёт **пустой публичный** репозиторий для проекта **на своём** GitHub-аккаунте.
2. Клонирует репозиторий [**сборки проекта**](https://github.com/goitacademy/parcel-project-template) себе на машину с GitHub Desktop.
3. В GitHub Desktop переходит в настройки клонированного репозитория и изменяет значение **remote** на url ранее созданного репозитория своего проекта выбрав HTTPS.
4. Открывает проект в редакторе кода и изменяет значения в файле **package.json** согласно инструкции. Делает коммит и пушит изменения на GitHub.
5. Проверяет работу авто деплоя.
6. Заходит в **«Settings»** репозитория и вносит следующие изменения.
   1. На вкладке **«Options»** в секции **«Merge button»** ставит чек на **Automatically delete head branches**.
   2. На вкладке **«Options»** в секции **«GitHub Pages»** настраивает хостинг проекта из ветки **gh-pages**.
   3. На вкладке **«Branches»** в секции **«Branch protection rules»** устанавливает правила для ветки **main**.
   4. На вкладке **«Manage access»** добавляет членов команды в репозиторий. Каждому будет выслан инвайт на почту.

## Разработчики

1. Устанавливают **LTS-версию** Node.js.
2. Клонируют репозиторий созданный тимлидом через GitHub Desktop.
3. В терминале выполняют команду **npm ci**.

# Разработка фичи или багфикс

1. Каждая фича или багфикс выполняется в отдельной **ветке**.
2. Перед началом работы над новой фичей или багфиксом обновляем локальный **main** чтобы он был идентичен **main** на GitHub.
3. Делаем новую ветку из **main**. Рекомендуется называть ветки по шаблону **feature/название-фичи** и **bugfix/название-бага**.
4. Работаем в этой ветке и вносим необходимые изменения. Коммитим изменения по необходимости.
5. После того как фича готова, отправляем историю (коммиты) локальной ветки на GitHub и открываем **Pull request**.
6. Назначаем минимум одного **Reviewer** чтобы сделал ревью кода.
7. Если необходимы правки, вносим их в той же ветке и отправляем изменения на GitHub. Текущий пул-реквест обновится автоматически, открывать **новый не надо**.
8. После того как пул-реквест прошел ревью кода, **тимлид** принимает его (мержит в ветку **main**).
9. После того как пул-реквест был принят, разработчик, в GitHub Desktop, переходит на ветку **main** и делает **fetch+pull** чтобы синхронизировать локальную ветку с той что на GitHub.
10. После синхронизации ветви **main**, разработчик удаляет свою локальную ветку фичи/бага в GitHub Desktop.

# Разрешение git-конфликтов

Происходят когда два разработчика одновременно изменяют одну и туже строку в каком-то файле. Возникают во время пул-реквеста, из-за чего он не может быть успешно смержен в базовую ветку. Лучше разрешать **локально**, а не на GitHub.

1. В GitHub Desktop переходим в ветку **main** и делаем **fetch+pull** чтобы синхронизировать её с веткой на GitHub в которой есть все изменения.
2. Переходим в ветку фичи/багфикса и сливаем (merge) в неё ветку **main**.
3. Разрешаем конфликт и делаем коммит.
4. Отправляем изменения на GitHub чтобы обновить пул-реквест.