Что можно делать на GitHub

1. О парадигме

При работе с GitHub-ом важно помнить идеологию, которая в него заложена:

1. *Версия* любого документа представляет из себя последовательность *коммитов*. Каждый коммит — это изменение и перезапись **всего файла целиком** (а точнее, некоторого непустого множества файлов — то есть, всего репозитория целиком).
2. Версии называются *ветвями*. Каждый репозиторий может содержать в себе несколько ветвей, но среди них есть выделенная — `master` — которая видна всем и является, в некотором роде, «парадной».
3. Каждый человек работает в репозитории, который ему принадлежит — либо он создал этот репозиторий с нуля, либо его пригласили в репозиторий в качестве соавтора, либо он скопировал чужой репозиторий.
4. Основная операция, посредством которой осуществляется работа GitHub — слияние двух ветвей. Если кто-то отредактировал у себя репозиторий, скопированный у другого человека (например, `/fbakharev/gazpromneft-report`), он может запросить слияние своей ветви с общей, чтобы сделать изменения в оригинальном репозитории. Даже открытие встроенного в GitHub текстового редактора и правка кода онлайн — это создание новой ветви, а сохранение этого кода в `master` — неявное слияние двух ветвей.

2. Как стать соавтором

Для этого нужно написать создателю репозитория, он добавит нового пользователя в соавторы. Соавторство позволяет изменять файлы в `master` без необходимости одобрять изменения у кого-либо ещё. Рекомендуется всем, принимающим активное участие в написании отчёта, стать соавторами.

Модераторами будем называть создателя и соавторов.

3. Что может делать соавтор

Главное — **изменять файлы в** `master`. Это растяжимое понятие включает в себя вообще всё, что только можно себе вообразить при работе с файлами: правка, создание, удаление, переименование, создание папок (главное — чтобы они сразу были непустыми).

Большие возможности влекут большую ответственность: правки в `master` невозможно отменить, и при внесении правок напрямую в `master` система сайта не проверяет и не показывает, где конкретно были сделаны изменения. Поэтому рекомендуется вносить такие правки, только если они мизерны, либо над файлом работаете только вы, либо вы очень уверены в том, что нигде не совершили ошибок и не удалили важные изменения.

Не менее главное — соавтор может создавать, обрабатывать и принимать запросы на слияние (т.н. *pull requests*). Создавать запросы на слияние могут вообще все, а вот их обработка и приём — дело исключительно соавторов.

4. Если вы модератор: про запросы на слияние

Запрос на слияние создаётся тогда, когда автор изменений хочет убедиться (или уточнить у модераторов), что вносимые им изменения не нарушат работу проекта в целом. В частности, поэтому **все** изменения не-соавторов проходят через запросы на слияние. Большинство pull request-ов система GitHub способна слить автоматически.

Однако,

(1) иногда GitHub неспособен выполнить слияние и *просит* модератора

(2) модератор всегда, даже если GitHub готов сделать всё сам, *может*

внести изменения в ветвь, которую предлагают слить с `master`.

В первом случае GitHub предлагает сделать `Resolve conflicts`, где представляет документ целиком, указывая на конкретные конфликтные места, и модератор может сам написать, что конкретно с ними сделать. Возможно, удалить всё к чёрту и написать что-то совершенно другое, но сейчас слово модератора — закон; и то, что выйдет из-под его пера, станет будущей версией вне зависимости от конфликтов с `master`, которые могли там остаться).

Во втором случае GitHub покажет, что он намерен удалить из ветви `master`, а что добавить в неё из предлагаемой ветви при слиянии. Модератор может попросить открыть текстовый редактор, где перед ним предстанет будущая «слитая» версия в представлении GitHub. Он может внести в неё какие-то правки (опять же, сейчас его слово — закон):

например, скопировать что-то, что GitHub хочет удалить, и впихнуть его в будущую версию.

Практика показывает, что:

1. Если в предлагаемой ветви одна строка текста заменена на другую, можно сделать только (2). При автоматическом слиянии в коде останется только вторая строка.
2. Если в предлагаемой ветви большой кусок текста заменён на что-то, случается (1).
3. Если предлагаемая ветвь отличается от текущего состояния `master` тем, что в одном месте убран кусок текста, а в другое место добавлен кусок текста, то GitHub предлагает автоматическое слияние (то есть (2)), и в «слитой» версии появятся **оба** куска текста.

В частности, третий пункт удобен тем, что два человека могут независимо впечатывать свои куски текста в один файл **оффлайн**, то есть, гарантируя успешную компиляцию файла, который потом появится в репозитории — и затем двумя последовательными запросами на слияние поместить в репозиторий плоды работы обоих.

5. Что может делать произвольный пользователь GitHub

Вкратце: работать с запросами на слияние.

0. Первым делом `fork`: эта опция создаёт в подчинении данного пользователя копию текущей версии репозитория (как тот же самый `/fbakharev/gazpromneft-report`).
1. Можно сделать pull request из оригинального репозитория в свой: это, разумеется, никак не изменяет общий репозиторий, а просто копирует его текущую версию (чтобы не отставать, если хочется вносить правки у себя).
2. Наконец, pull request из своего репозитория в общий: для помещения в текущую «отчётную» версию правок, внесённых в своей копии репозитория.

Для опций „1.“ и „2.“ на страницах репозиториев есть кнопка „New pull request“, позволяющая выбрать, что и с чем хочется слить.

3. Делать в своём репозитории всё то же, что модератор может делать в общем: приглашать своих соавторов, обрабатывать запросы на слияние от каких-то других пользователей GitHub (например, своих прямых подчинённых).