Создание проекта и первые коммиты

Git - это распределенная система контроля версий. И на машине разработчика всегда находится полноценный репозиторий, в котором можно создавать ветки, держать историю и который синхронизируется с другими репозиториями членов команды.

Это означает, что члены команды могут взаимодействовать даже без отдельно выделенного сервера, поддерживающего основной репозиторий. Однако, в таком случае, вам необходимо либо предоставить доступ к вашей машине извне (например, предоставить http-доступ к репозитория или через ssh), или же обмениваться коммитами через email. Это все не всегда удобно и просто.

Поэтому, как правило, команды имеют выделенный сервер, который используется для поддержки основного репозитория и синхронизируются через него. Мы будем использовать для этого GitHub.

Для начала работы, создадим репозиторий на своей локальной машине.

```
mkdir my-project
cd my-project
git init
```

После выполнения последней команды, в вашей директории создастся папка .git, где содержится сам репозиторий - база данных его объектов (blob, деревья, коммиты и т.д.).

Сама ваша директория (my-project в данном случае) является рабочей копией. Для того, чтобы начать отслеживать изменения в своем проекте, нужно добавить нужные файлы в индекс (индекс держит изменения, которые в дальнейшем зафиксируются в виде коммита):

```
echo "My Project"> README.md
git add README.md
```

Теперь можно добавить изменения в истории с помощью команды commit:

```
git commit
```

После того, как вы введете сообщения будет создан комит. Комит-объект создается из данных, которые занесены в ваш индекс.
Теперь нужно отправить изменения в репозиторий на Github. Для этого нужно создать его на сервере.

```
https://github.com/new
```

С помощью данной страницы создайте новый публичный репозиторий. Это, фактически, выполнит git init на сервере. Когда репозиторий создан, вы увидите ссылку, которую нужно использовать для того, чтобы настроить внешний репозиторий, с которым будет проходить синхронизация на вашей машине. Например,

```
git remote add origin git@github.com:roman-mazur/test-repo.git
```

Последняя команда добавит внешний репозиторий с коротким именем "origin". Короткое имя может в дальнейшем использоваться в командах push, fetch и pull. Команда

```
git push -u origin master
```

отправит коммит ветки master в репозиторий origin.
Однако, перед тем, как выполнить эту команду, необходимо настроить ssh-ключи, чтобы сервер мог авторизовать вас и принять изменения. Ключи могут быть настроены на странице

```
https://github.com/settings/ssh
```

Подробнее о настройке ключей можно найти по следующей ссылке:
https://help.github.com/articles/generating-ssh-keys/

Другие члены команды могут клонировать ваш репозиторий:
`git clone <repo_url>`

Для того, чтобы другие члены вашей команды могли выполнять push в ваш репозиторий, в настройках проекта им нужно дать на это права.
https://github.com/<github_name>/<repo_name>/settings/collaboration

Но, поскольку git - это распределенная система контроля версий, последний пункт не обязателен: каждый из членов команды может иметь свой репозиторий на github, и каждый будет забирать изменения, имеющиеся в репозиториях коллег. Например,

```
git remote add friend1 <friend's repo url>
git pull friend1 master
```

Обязательно познакомьтесь с главами 2 и 3 книги Pro Git для следующих задач
https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository
https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

Вам необходимы будут знания о создании ветвей (`git branch, git checkout -b`), слияние ветвей (`git merge`), а также перебазирование Комит (`git rebase`).

Для зачисления практической работы, вам необходимо продемонстрировать github-репозиторий, который будет удовлетворять следующие требования.

1. Ветка master вашего репозитория содержит коммиты всех членов команды (следите за тем, чтобы email, который git записывает, когда сохраняет сведения об авторе, совпадал с email'ом, который вы использовали для регистрации на GitHub).

2. Ветка master содержит коммиты от всех членов команды подряд (т.е. между рядом коммитов, созданных различными членами команды, не должно быть merge-коммитов). Это не обязательно для абсолютно всей истории ветки. Но должно быть по крайней мере один такой фрагмент. Для этого вам нужно познакомиться с тем, что такое fast-forward в git и перебазирование коммитов (git rebase). Об использовании git rebase можете дополнительно посмотреть в данной статье:
https://www.atlassian.com/git/tutorials/merging-vs-rebasing/

3. В репозитории должны присутствовать merge-коммиты, выполненные каждым из членов команды (вам понадобится git merge).
4. Ветка master должна иметь по крайней мере один revert-коммит (который применяет обратные изменения к определенному коммиту). Как ни странно, вам нужно будет воспользоваться командой git revert. Также следите за тем, чтобы commit-сообщение содержало ссылки на то коммит, изменения которого отменяются.

Если возникают вопросы, или вы нашли несоответствия в задачах, пишите в группу обсуждения: https://groups.google.com/forum/#!forum/kpi-integration-2015