**Наш удаленный репозиторий находится по адресу:**

<https://github.com/andreyfree2020/SF-project-work-2.git>

**Как выглядит процесс разработки в нашей компании описано в Приложении 1 (ниже в данном файле).**

**Контроль доступа к репозиторию осуществлен на основании следующих принципов:**

1. **Вам предоставлен доступ к репозиторию как Collaborator, вы можете клонировать к себе ветки master и develop.**
2. **Вы не можете вносить изменения в ветки master, release.**
3. **Вы не можете напрямую вносить изменения в ветку develop.**
4. **Вы можете инициировать pull request по мерджу своей ветки JIRA-id в ветку develop.**
5. **Ваша разработка в ветке JIRA-id будет внедрена в ветку develop только после review вашего кода вашим руководителем (team-lead вашей команды).**
6. **Изменения в основную кодовую базы ветки develop вносит ваш team-lead путем подтверждения вашего pull request после процедуры review вашей работы в вашей ветке JIRA-id.**

**Ваши задачи вы будете делать ТОЛЬКО в своей ветке JIRA-id, которую вы будете клонировать от ветки develop.**

**id – это номер вашей задачи в системе JIRA. Задачи вам будет инициировать в системе JIRA ваш руководитель (team-lead команды).**

**Итак, начинаем работу с репозиторием!**

**Если вы никогда ранее не использовали git, для начала работы с удаленным репозиторием установите у себя на ПК Git-клиент (вы находитесь в своем домашнем каталоге ~/):**

~ sudo apt install git

**Выполните следующие команды, чтобы git узнал ваше имя и электронную почту:**

~ git config --global user.name "Your Name"

~ git config --global user.email "your\_email@your\_mail\_host"

**Создайте у себя локально в домашней каталоге (/home/your\_user\_name/) рабочий каталог для вашей задачи** (JIRA-id – вместо id поставьте номер вашей задачи в Jira):

~ mkdir JIRA-id

**Перейдите в свой рабочий каталог:**

~ cd JIRA-id

**Иниализируйте свой локальный репозиторий:**

~ git init

**Свяжите свой локальный репозиторий с удаленным:**

~ git remote add origin <https://github.com/andreyfree2020/SF-project-work-2.git>

**Скачайте себе ветку master с удаленного репозитория содержащую данный файл README, а также файл конфигурации .gitconfig:**

~ git pull origin master

**Измените свой файл конфигурации ~/.gitconfig одним из двух способов:**

* **Добавьте в него блок [alias] из скаченного из репозитория файла .gitconfig.**
* **Или скопируйте скаченный из репозитория файл конфигурации .gitconfig в свой домашний каталог (**~ cp .gitconfig ~/.gitconfig**) и измените в нем блок [user] на свои данные.**

**!!!ВАЖНО!!! Если у вас возникнут сложности при работе с .gitconfig – обратитесь к Приложению 2 (ниже в данном файле), где работа с .gitconfig описана более подробно.**

**Скачайте себе ветку develop с удаленного репозитория:**

~ git pull origin develop

**Переключитесь на ветку develop:**

~ git checkout develop

**Создайте свою рабочую ветку для вашей задачи:**

~ git branch JIRA-id (например, JIRA-102030)

**Перейдите на свою рабочую ветку:**

~ git checkout JIRA-id

**Проверьте, что вы перешли в свою ветку:**

~ git status

**!!ВАЖНО!!!**

**Вся разработка (создание новых или редактирование существующих файлов) делается вами ТОЛЬКО в вашей рабочей ветке JIRA-id.**

**Например:**

**Создаем новый файл my\_file.html:**

~ touch my\_file.html

**Редактируем созданный файл (редакторы vi, nano или ваша IDE):**

~ vi my\_file.html (редактор vi, выход из редактора с сохранением Esc+wq)

~ nano my\_file.html (редактор nano, выход из редактора с сохраненем Ctrl+X, Ctrl+O)

**Добавляем созданный файл в индекс для отслеживания git-ом:**

~ git add my\_file.html

**ОБЯЗАТЕЛЬНО! Фиксируем текущее состояние файла в системе git – пишем конкретный и понятный комментарий по сделанным изменениям:**

~ git commit –m «Ваш комментарий по сделанным изменениям»

**Проверяем текущий статус в git:**

~ git status

**После окончания работы над вашей задачей проверьте:**

* Выполнены ли все пункты ТЗ по вашей задаче в JIRA.
* Соответствует ли id номер вашей задачи в JIRA номеру задачи в имени вашей ветки JIRA-id.
* Наличие всех необходимых файлов в вашем рабочем каталоге ( ~ ls -lah)

**После окончания разработки вашей задачи:**

**НА ВАШЕМ ЛОКАЛЬНОМ ПК:**

**Отправляем вашу созданную ветку в центральный репозиторий:**

~ git push origin JIRA-id

**В УДАЛЕННОМ РЕПОЗИТОРИИ github.com:**

**Формируем pull request на слияние (merge) вашей ветки JIRA-id с основной веткой разработки develop.**

**!!!ВАЖНО!!!**

При формировании pull request для дальнейшего продуктивного ревью вашей работы, сделайте подробный комментарий, что вы сделали в рамках своей задачи и какие файлы включает в себя ваш проект (раздел Comment).

Ревью вашей работы в процессе pull request проводит team-lead вашей команды. Вы не сможете завершить pull request самостоятельно! Для сдачи вашей работы необходимо учесть все замечания и внести необходимые корректировки от вашего руководителя.

**ПРИЛОЖЕНИЯ:**

**ПРИЛОЖЕНИЕ 1. Как выглядит процесс разработки в нашей компании.**

**ПРИЛОЖЕНИЕ 2. Работа с файлом .gitconfig**

**ПРИЛОЖЕНИЕ 1. Как выглядит процесс разработки в нашей компании.**

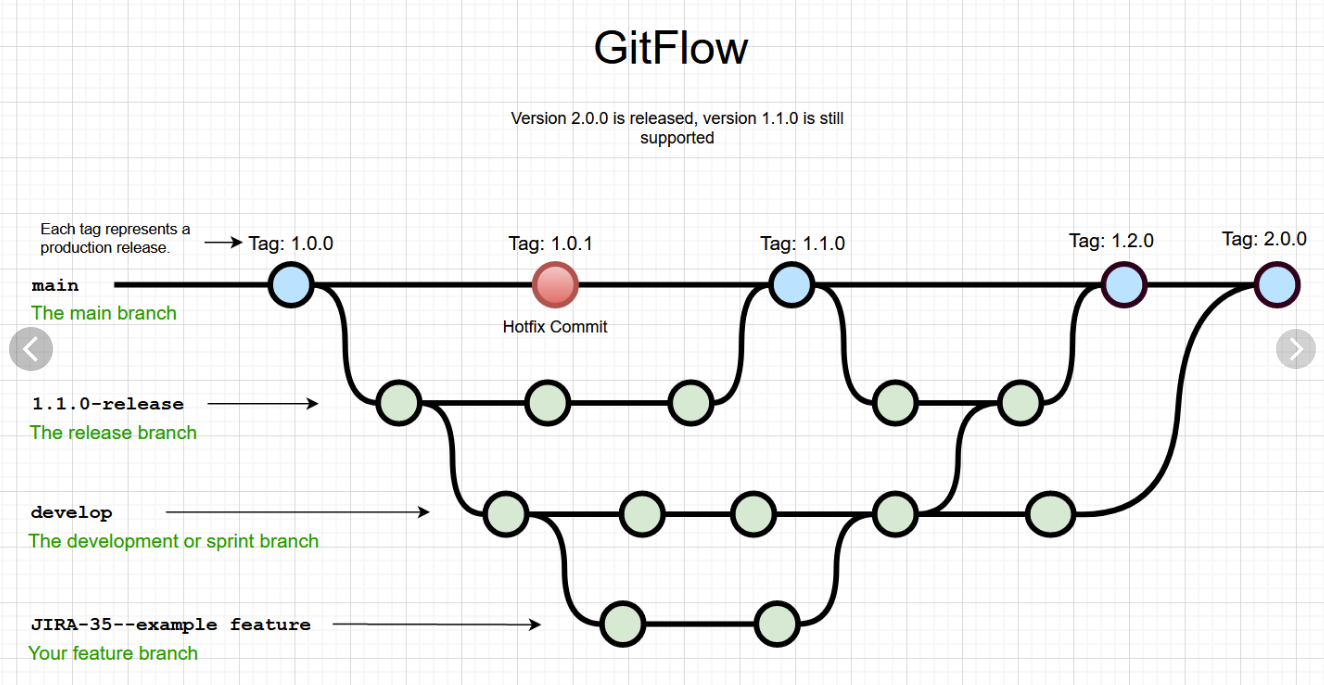
**Главная ветка - main/master.**

**Основная ветка разработки - develop.**

**Когда мы начинаем новую задачу – мы отводим фича ветку JIRA-id от develop ветки.**

**После прохождения код-ревью вливаем фича ветку JIRA-id обратно в develop.**

**Собираем релизы из develop ветки в ветку release и далее релизимся в main/master.**



**ПРИЛОЖЕНИЕ 2. Работа с файлом .gitconfig**

**Чтобы посмотреть в каком файле, какие настройки сейчас установлены используйте:**

~ git config --list --show-origin

**Файл конфигурации для локальной работы с git (~/.gitconfig) включает в себя 2 блока - [user] и [alias]:**

[user]

email = your\_email@your\_mail\_host

name = your\_name

[alias]

history = log --pretty=format:\"%h %ad | %s%d [%an]\" --graph --date=short

!**!!ВАЖНО!!!**

**Измените в блоке [user] файла .gitconfig данные в строках email и name на свои данные при регистрации на github.com.**

**Например:**

[user]

email = andreyfree2020@example.com

name = andreyfree2020

**Дополнительно, в раздел [alias] вы можете добавить сокращения (псевдонимы или алиасы) наиболее часто используемых вами команд.**

**Например:**

[alias]

history = log --pretty=format:\"%h %ad | %s%d [%an]\" --graph --date=short

co = checkout

ci = commit

st = status

br = branch