Лабораторная работа №5

Управление версиями с помощью Git

Для выполнения этого задания рекомендуется использовать ОС GNU/Linux. В случае использования ОС семейства Windows, необходимо установить эмулятор среды UNIX — CyGNUs (скачать дистрибутив и узнать подробности об установке можно по адресу http://git-scm.com/download или https://msysgit.github.io/).

Полное руководство пользователя находится по адресу: git-scm.com/book/ru/v2.

Задание 1. Работа с локальным репозиторием

1. Установите на свой компьютер Git.
2. Каждый коммит в Git связан с определённой персоной, выполнившей то или иное действие. Поэтому, первым делом введите свои данные:

   ```
   git config --global user.name "Имя"
   git config --global user.email "адрес почты"
   ```

3. Проверьте конфигурацию Git, выполнив команду

   ```
   git config --list
   ```

4. Создайте на своём компьютере директорию `PM-projectName`, здесь вместо `projectName` напишите имя вашего командного проекта. Скопируйте туда все свои файлы, созданные в рамках этой дисциплины (можно использовать поддиректории).

   Все дальнейшие команды выполняются в директории `PM-projectName`.

5. Инициализируйте репозиторий:

   ```
   git init
   ```

   Эта ветвь в Git, по умолчанию, называется `master`.

6. Добавьте все файлы из директории `PM-projectName` в индекс Git:

   ```
   git add .
   ```

   В дальнейшем, если надо обновить уже существующие файлы, используйте команду (см. рис. 1)

   ```
   git add -u
   ```

   В именах файлов можно использовать шаблоны.

7. Добавьте все проиндексированные файлы в репозиторий:

   ```
   git commit -m "Новые файлы."
   ```

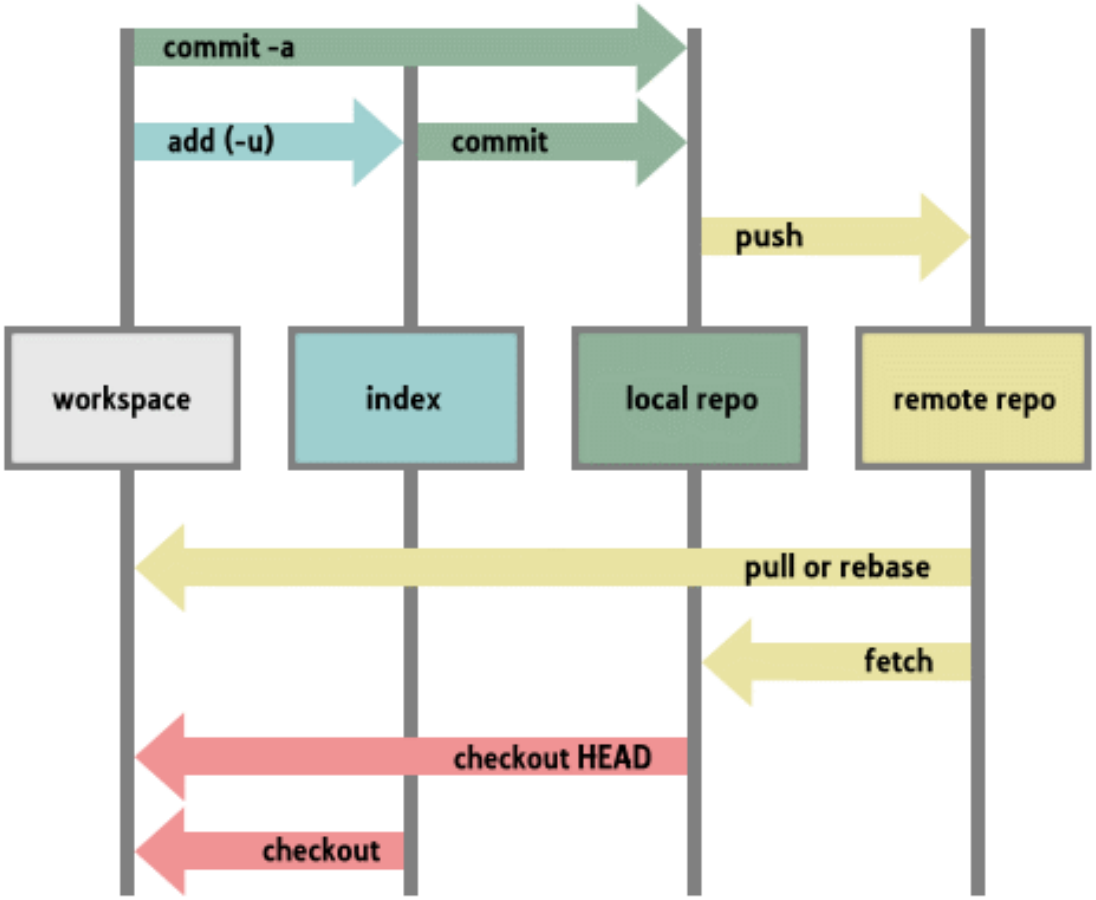

Рис. 1. Структура Git и процессы управления репозиторием

Эта ветвь в Git, по умолчанию, называется `master`.

8. Посмотреть историю изменений можно командой

```
git log
```

Вывод следующей строки осуществляется нажатием Enter, вывод следующей страницы экрана — Space, выход из режима git осуществляется нажатием клавиши q.

Применение этой команды с аргументом `-p` будет показана разница (патч), внесённая в каждый коммит, а `-n` — ограничит вывод только n записей. Например:

```
git log -p -3
```

9. Чтобы посмотреть сокращённую статистику для каждого коммита, используется опция `-stat`:

```
git log --stat
```

10. Воспользуйтесь опцией `-graph`, чтобы увидеть граф в формате ASCII, который показывает текущую ветку и историю слияний:

```
git log --graph
```

Задание 2. Работа с удалённым репозиторием

1. Зарегистрируйтесь на GitHub.

2. Создайте репозиторий `PM-projectName`. Обязательно инициализируйте репозиторий файлом `README`.

3. Создайте персональный токен для доступа к репозиторию. Для этого зайдите в настройки своего профиля, далее — в левой панели ▶ Developer settings ▶ Personal access tokens, Generate new token. На открывшейся странице задайте описание (для чего планируете использовать токен), время действия токена, области применения токена. Чтобы использовать свой токен для доступа к репозиториям из командной строки, обязательно выберите **repo**. Нажмите кнопку Generate token. Скопируйте полученный токен и используйте его вместо пароля (когда GitHub будет запрашивать пароль). Иллюстрированная инструкцию доступна на портале GitHub.

4. Для синхронизации локального и удалённого репозитория выполните команду:

   ```
   git pull https://github.com/ВашеИмя/ВашРепозиторий.git
   ```

5. Скопируйте локальный репозиторий в удалённый:

   ```
   git push https://github.com/ВашеИмя/ВашРепозиторий.git
   ```

6. Скопируйте ветку `https://github.com/zhurenkov/PM` в свой репозиторий, используя инструмент Fork (API GitHub). Для этого перейдите в нужную директорию и нажмите кнопку Fork.

7. Для копирования своего репозитория из GiHub на другой компьютер, выполните пункты 1–5 из первого задания и задайте команду

   ```
   git clone https://github.com/ВашеИмя/ВашРепозиторий.git
   ```

В отчёт (на Moodle) напишите адрес своего удалённого репозитория на GitHub.