Лабораторная работа №2

Управление версиями

Камкина Арина Леонидовна

Содержание

Цель работы

- **Изучить идеологию и применение средств контроля версий.**
- **Освоить умения по работе с** git.

Задание

- **Создать базовую конфигурацию для работы с** git.
- **Создать ключ** SSH.
- **Создать ключ** PGP.
- **Настроить подписи** git.
- **Зарегистрироваться на** Github.
- **Создать локальный каталог для выполнения заданий по предмету.**

Теоретическое введение

Основные команды git

Наиболее часто используемые команды git: - **создание основного дерева репозитория:** > git init - **получение обновлений (изменений) текущего дерева из центрального репозитория:** > git pull - **отправка всех произведённых изменений локального дерева в центральный репозиторий:** > git push - **просмотр списка изменённых файлов в текущей директории:** > git status - **просмотр текущих изменения:** > git diff - **сохранение текущих изменений:** - **добавить все изменённые и/или созданные файлы и/или каталоги:** > git add . - **добавить конкретные**

изменённые и/или созданные файлы и/или каталоги: > git add **имена_файлов - удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории):** > git rm **имена_файлов - сохранение добавленных изменений: - сохранить все добавленные изменения и все изменённые файлы:** > git commit -am **'Описание коммита' - сохранить добавленные изменения с внесением комментария через встроенный редактор:** > git commit - **создание новой ветки, базирующейся на текущей:** > git checkout -b **имя_ветки - переключение на некоторую ветку:** > git checkout **имя_ветки (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) - отправка изменений конкретной ветки в центральный репозиторий:** > git push origin **имя_ветки - слияние ветки с текущим деревом:** > git merge –no-ff **имя_ветки - удаление ветки: - удаление локальной уже слитой с основным деревом ветки:** > git branch -d **имя_ветки - принудительное удаление локальной ветки:** > git branch -D **имя_ветки - удаление ветки с центрального репозитория:** > git push origin **:имя_ветки**

Выполнение лабораторной работы

1. **Для начала создала йчетную запичь на** github **и заполнила основные данные.**

2. **Установила** git-flow **в** Fedora Linux **(рис.** [-@fig:001])

```
[alkamkina@fedora ~]$ cd /tmp
[alkamkina@fedora tmp]$ wget --no-check-certificate -q https://raw.github.com/petervanderdoes
/gitflow/develop/contrib/gitflow-installer.sh
[alkamkina@fedora tmp]$ chmod +x gitflow-installer.sh
[alkamkina@fedora tmp]$ sudo ./gitflow-installer.sh install stable
[sudo] пароль для alkamkina:
```

{ #fig:001 width=70% }

3. **Установила** gh **в** Fedora Linux **(рис.** [-@fig:002])

```
[alkamkina@fedora tmp]$ sudo dnf install gh
Последняя проверка окончания срока действия метаданных: 0:00:28 назад, Пн 06 июн 2022 12:18:3
5.
Пакет gh-2.7.0-1.fc35.x86_64 уже установлен.
Зависимости разрешены.
Отсутствуют действия для выполнения.
Выполнено!
```

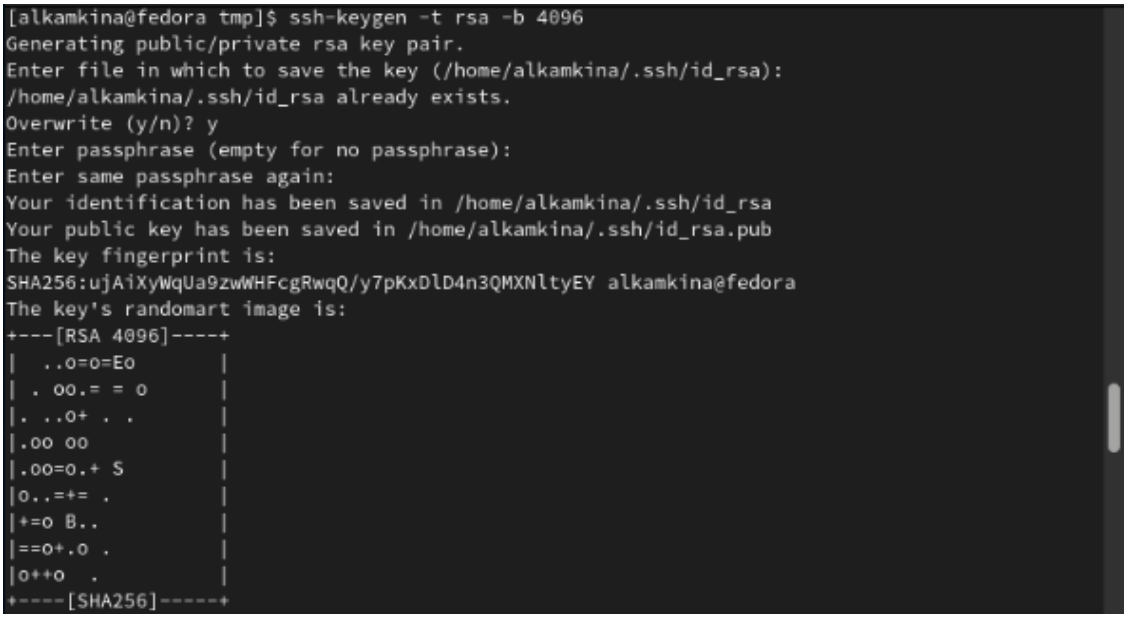
{ #fig:002 width=70% }

4. **Произвела базовую настройку** git **(рис.** [-@fig:003])

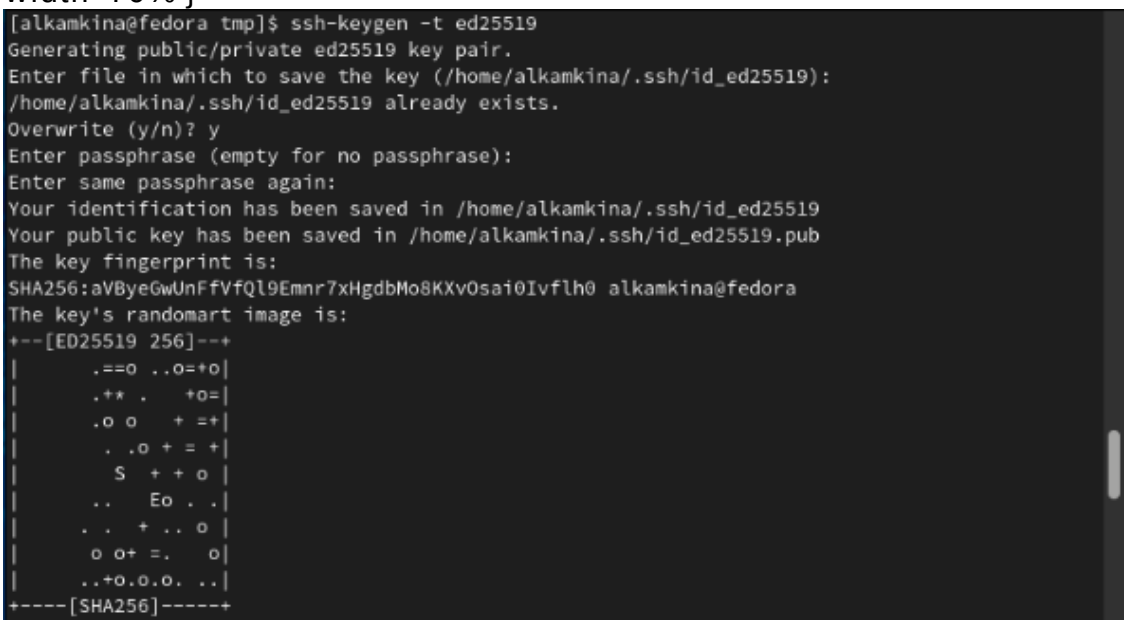
```
[alkamkina@fedora tmp]$ git config --global user.name "Arina Kamkina"
[alkamkina@fedora tmp]$ git config --global user.email "arina.kamkinaleo@mail.ru"
[alkamkina@fedora tmp]$ git config --global init.defaultBranch master
[alkamkina@fedora tmp]$ git config --global core.autocrlf input
[alkamkina@fedora tmp]$ git config --global core.safecrlf warn
```

{ #fig:003 width=70% }

5. **Произвела базовую настройку** git **(рис.** [-@fig:004] [-@fig:005])



{ #fig:004 width=70% }



{ #fig:004 width=70% }

6. **Создала ключ** pgp **и добавила его в** github **(рис.** [-@fig:006])

```
[alkamkina@fedora tmp]$ gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: глубина: 0  достоверных:   3  подписанных:   0  доверие: 0-, 0q, 0n, 0m, 0f, 3u
/home/alkamkina/.gnupg/pubring.kbx
--------------------------------
sec   rsa4096/D9CBF26C842937B3 2022-04-23 [SC]
      A969646C684EB84B3E15B32DD9CBF26C842937B3
uid                 [  абсолютно ] Arina Kamkina <arina.kamkinaleo@mail.ru>
ssb   rsa4096/29859E868515FC99 2022-04-23 [E]

sec   rsa4096/85E83D01527D639B 2022-05-07 [SC]
      924E19183EE6BCF40B62F09685E83D01527D639B
uid                 [  абсолютно ] Arina Kamkina <arina.kamkinaleo@mail.ru>
ssb   rsa4096/48FACB1E829DA0FF 2022-05-07 [E]

sec   rsa4096/62D495A6A6043097 2022-06-06 [SC]
      087307652FA401AC8A943C8B62D495A6A6043097
uid                 [  абсолютно ] Arina Kamkina <arina.kamkinaleo@mail.ru>
ssb   rsa4096/337A731D3334A8A7 2022-06-06 [E]
```

{ #fig:007 width=70% }

7. **Добавила его в** github **и настроила автоматические подписи коммитов** git **и авторизовалась (рис.** [-@fig:007])

```
[alkamkina@fedora tmp]$ gpg --armor --export D9CBF26C842937B3 | xclip -sel clip
[alkamkina@fedora tmp]$ git config --global user.signingkey ^C
[alkamkina@fedora tmp]$ git config --global user.signingkey D9CBF26C842937B3
[alkamkina@fedora tmp]$ git config --global commit.gpgsign true
[alkamkina@fedora tmp]$ git config --global gpg.program $(which gpg2)
[alkamkina@fedora tmp]$ gh auth login
```

{ #fig:007 width=70% }

8. **Создала каталог и затем перешла в него (рис.** [-@fig:008])

```
[alkamkina@fedora tmp]$ mkdir -p ~/work/study/2021-2022/"Операционные системы"
[alkamkina@fedora tmp]$ cd ~/work/study/2021-2022/"Операционные системы"
```

{ #fig:008 width=70% }

9. **Создала репозиторий на основе шаблона (рис.** [-@fig:009])

```
[alkamkina@fedora Операционные системы]$ gh repo create study_2021-2022_os-intro --template=y
amadharma/course-directory-student-template --public
✓ Created repository sqrtlsk/study_2021-2022_os-intro on GitHub
[alkamkina@fedora Операционные системы]$ git clone --recursive git@github.com:sqrtlsk/study_2
021-2022_os-intro.git os-intro
Клонирование в «os-intro»…
```

{ #fig:009 width=70% }

10. **Настроила каалог курса и отправила нужные файлы на сервер (рис.** [-@fig:0010] [-@fig:0011)

```
[alkamkina@fedora os-intro]$ rm package.json
[alkamkina@fedora os-intro]$ make COURSE=os-intro
[alkamkina@fedora os-intro]$ git add .
[alkamkina@fedora os-intro]$ git commit -am 'feat(main): make course structure'
[master 75aea2e] feat(main): make course structure
```

{ #fig:0010 width=70% }

```
[alkamkina@fedora os-intro]$ git push
Перечисление объектов: 20, готово.
Подсчет объектов: 100% (20/20), готово.
Сжатие объектов: 100% (16/16), готово.
Запись объектов: 100% (19/19), 266.52 КиБ | 1.61 МиБ/с, готово.
Всего 19 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пакетов
0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:sqrtlsk/study_2021-2022_os-intro.git
   a416fa0..75aea2e  master -> master
```
{ #fig:0011 width=70% }

Выводы

Освоила работу с git **и получила знания о некоторых его функциях.**

Контрольные вопросы

1. **Системы контроля версий** (Version Control System, VCS) **применяются при работе нескольких человек над одним проектом. Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.**

- **Хранилище - централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов**
- **Commit - команда для записи индексированных изменений в репозиторий.**
- **История - место, где сохраняются все коммиты, по которым можно посмотреть данные о коммитах.**
- **Рабочая копия - текущее состояние файлов пректа основанное на версии, загруженной из хранилища.**
- **Централизованные системы – это системы, в которых одно основное хранилище всего проекта, и каждый пользователь копирует необходимые ему файлы, изменяет и вставляет обратно. Пример** – Subversion.
- **Децентрализованные системы – система, в которой каждый пользователь имеет свой вариант репозитория и есть возможность добавлять и забирать изменения из репозиториев. Пример** – Git.

4. **В рабочей копии, которую исправляет человек, появляются правки, которые отправляются в хранилище на каждом из этапов. То есть в правки в рабочей копии появляются, только если человек делает их.**

5. **Если хранилище общее, то в рабочую копию каждого, кто работает над проектом, приходят изменения, отправленные на сервер одним из команды. Рабочая правка каждого может изменяться вне зависимости от того, делает ли конкретный человек правки или нет.**
6. **У** Git **две основных задачи: первая — хранить информацию обо всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.**

- **создание основного дерева репозитория:** > git init
- **получение обновлений (изменений) текущего дерева из центрального репозитория:** > git pull
- **отправка всех произведённых изменений локального дерева в центральный репозиторий:** > git push
- **просмотр списка изменённых файлов в текущей директории:** > git status
- **просмотр текущих изменения:** > git diff
- **сохранение текущих изменений:**
- **добавить все изменённые и/или созданные файлы и/или каталоги:** > git add .
- **добавить конкретные изменённые и/или созданные файлы и/или каталоги:** > git add **имена_файлов**
- **удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории):** > git rm **имена_файлов**
- **сохранение добавленных изменений:**
- **сохранить все добавленные изменения и все изменённые файлы:** > git commit -am **‘Описание коммита’**
- **сохранить добавленные изменения с внесением комментария через встроенный редактор:** > git commit
- **создание новой ветки, базирующейся на текущей:** > git checkout -b **имя_ветки**
- **переключение на некоторую ветку:** > git checkout **имя_ветки (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)**
- **отправка изменений конкретной ветки в центральный репозиторий:** > git push origin **имя_ветки**
- **слияние ветки с текущим деревом:** > git merge –no-ff **имя_ветки**
- **удаление ветки:**
- **удаление локальной уже слитой с основным деревом ветки:** > git branch -d **имя_ветки**
- **принудительное удаление локальной ветки:** > git branch -D **имя_ветки**
- **удаление ветки с центрального репозитория:** > git push origin **:имя_ветки**
- **Работа с удаленным репозиторием:** git remote – **просмотр списка настроенных удаленных репозиториев.**
- **Работа с локальным репозиторием:** git status - **выводит информацию обо всех изменениях, внесенных в дерево директорий проекта по сравнению с последним коммитом рабочей ветки.**

9. **Ветка — это последовательность коммитов, в которой ведётся параллельная разработка какого-либо функционала. Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.**
10. **Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты. В** Git **нет специальной команды для указания игнорируемых файлов: вместо этого необходимо вручную отредактировать файл.**

- **Временно игнорировать изменения в файле можно командой** git > update-index-assumeunchanged