

Отчет по лабораторной работе №2

Операционные системы

Кузьмин Егор Витальевич

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Установка программного обеспечения	7
3.2	Базовая настройка git	7
3.3	Создание ключа SSH	8
3.4	Создание ключа GPG	9
3.5	Регистрация на Github	10
3.6	Добавление ключа GPG в Github	10
3.7	Настроить подписи Git	10
3.8	Настройка gh	11
3.9	Создание репозитория курса на основе шаблона	12
4	Выводы	14
5	Ответы на контрольные вопросы.	15
	Список литературы	18

Список иллюстраций

3.1	Установка git и gh	7
3.2	Настройка конфига git	8
3.3	Генерация ssh ключа по алгоритму rsa	8
3.4	Генерация ssh ключа по алгоритму ed25519	9
3.5	Генерация ключа	9
3.6	Вывод списка ключей	10
3.7	Добавленный ключ GPG	10
3.8	Настройка подписей Git	11
3.9	Авторизация в gh	11
3.10	Завершение авторизации через браузер	11
3.11	Завершение авторизации	11
3.12	Создание репозитория	12
3.13	Подтверждение пароля	13
3.14	Работа с каталогами	13
3.15	Отправка файлов на сервер	13

Список таблиц

1 Цель работы

Целью данной лабораторной работы является изучение идеологии и получение практических навыков в применении средств контроля версий и работе с git.

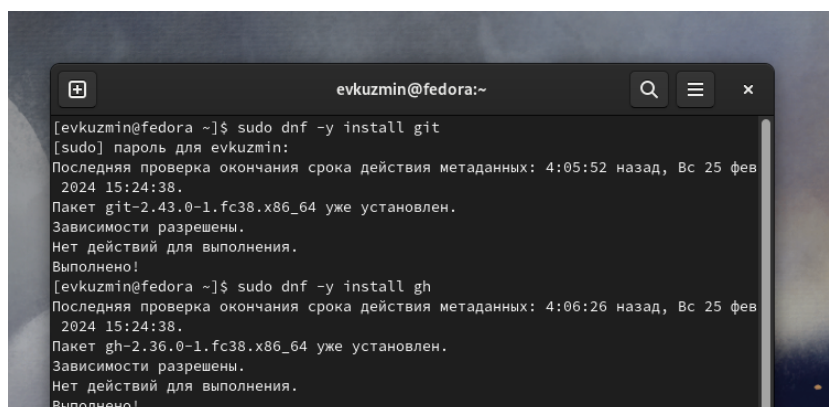
2 Задание

0. Базовое ознакомление с git
1. Создать базовую конфигурацию для работы с git
2. Создать ключ SSH
3. Создать ключ GPG
4. Настроить подписи Git
5. Зарегистрироваться на GitHub
6. Создать локальный каталог для выполнения заданий по предмету.

3 Выполнение лабораторной работы

3.1 Установка программного обеспечения

Устанавливаю необходимое программное обеспечение git и gh через терминал с помощью команд: `dnf install git` и `dnf install gh` (рис. 1).

A screenshot of a terminal window titled 'evkuzmin@fedora:~'. The terminal shows two successful installation commands. The first command is 'sudo dnf -y install git', which prompts for the password 'evkuzmin:' and shows the installation of 'git-2.43.0-1.fc38.x86_64'. The second command is 'sudo dnf -y install gh', which shows the installation of 'gh-2.36.0-1.fc38.x86_64'. Both installations are confirmed as successful.

```
[evkuzmin@fedora ~]$ sudo dnf -y install git
[sudo] пароль для evkuzmin:
Последняя проверка окончания срока действия метаданных: 4:05:52 назад, Вс 25 фев 2024 15:24:38.
Пакет git-2.43.0-1.fc38.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
[evkuzmin@fedora ~]$ sudo dnf -y install gh
Последняя проверка окончания срока действия метаданных: 4:06:26 назад, Вс 25 фев 2024 15:24:38.
Пакет gh-2.36.0-1.fc38.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
```

Рис. 3.1: Установка git и gh

3.2 Базовая настройка git

Задаю в качестве имени и email владельца репозитория свои имя, фамилию и электронную почту, настраиваю utf-8 в выводе сообщений git для их корректного отображения, задаю имя начальной ветке, задаю параметры `autocrlf` и `safecrlf` для корректного отображения конца строки (рис. 2).

```
[evkuzmin@fedora ~]$ git config --global user.name "Egor Kuzmin"
[evkuzmin@fedora ~]$ git config --global user.email "1132236046@pfur.ru"
[evkuzmin@fedora ~]$ git config --global core.quotepath false
[evkuzmin@fedora ~]$ git config --global init.default Branch master
[evkuzmin@fedora ~]$ git config --global core.autocrlf input
[evkuzmin@fedora ~]$ git config --global core.safecrlf warn
[evkuzmin@fedora ~]$
```

Рис. 3.2: Настройка конфига git

3.3 Создание ключа SSH

Создаю ключ ssh размером 4096 бит по алгоритму rsa (рис. 3).

```
evkuzmin@fedora:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/evkuzmin/.ssh/id_rsa):
/home/evkuzmin/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/evkuzmin/.ssh/id_rsa
Your public key has been saved in /home/evkuzmin/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:0FpAgCNrW8AV0TMZIdZyLQKoOceHIVDRa505qkqKD2U evkuzmin@fedora
The key's randomart image is:
+----[RSA 4096]-----+
|==+BO|
|B=O@..|
|++=O++|
|B.=.B. |
|o+Eo o+ S|
| + .o .|
|.. ..|
|+..|
|*o.|
+----[SHA256]-----+
[evkuzmin@fedora ~]$
```

Рис. 3.3: Генерация ssh ключа по алгоритму rsa

Создаю ключ ssh по алгоритму ed25519 (рис. 4).

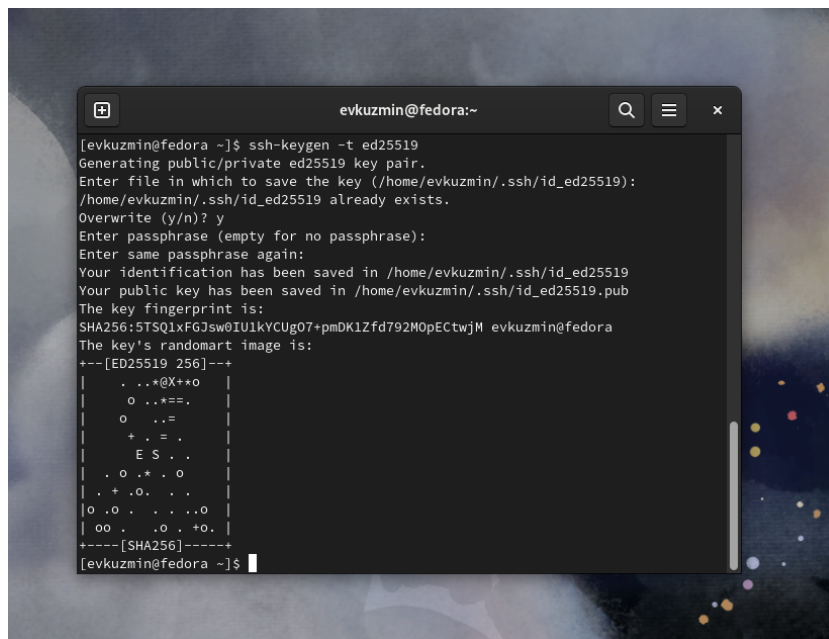


Рис. 3.4: Генерация ssh ключа по алгоритму ed25519

3.4 Создание ключа GPG

Генерирую ключ GPG, затем выбираю тип ключа RSA, задаю максимальную длину ключа: 4096, выбираю неограниченный срок действия ключа. Завершаем настройку (рис. 5).

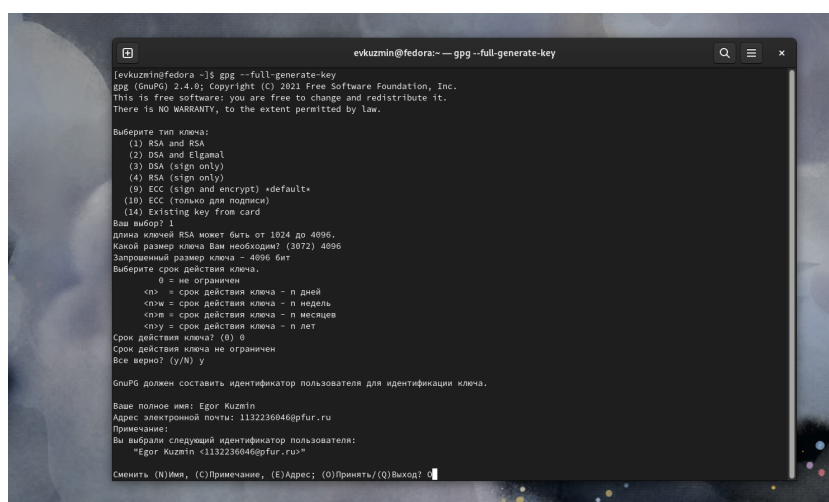


Рис. 3.5: Генерация ключа

3.5 Регистрация на Github

У меня уже был создан аккаунт на Github, соответственно просто вхожу в свой аккаунт.

3.6 Добавление ключа GPG в Github

Вывожу список созданных ключей в терминал, ищу в результате запроса отпечаток ключа, потом копирую его в буфер обмена. Ввожу в терминале команду, с помощью которой копирую сам ключ GPG в буфер обмена, за это отвечает утилита xclip (рис. 6).

```
[evkuzmin@fedora ~]$ gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: gpgmail: 0 достоверных: 2 подписанных: 0 доверие: 0-, 0q, 0n, 0f, 2u
/home/evkuzmin/.gnupg/pubring.kbx
-----
sec rsa4096/DCB850EAD44A6468 2024-02-25 [SC]
uid [ абсолютно ] Egor <1132236046@pfur.ru>
ssb rsa4096/8E627040A01FD91 2024-02-25 [E]
-----
sec rsa4096/2783037481DCFEF7 2024-02-25 [SC]
uid [ абсолютно ] Egor Kuzmin <1132236046@pfur.ru>
ssb rsa4096/A11A0974351F18CD 2024-02-25 [E]
-----
[evkuzmin@fedora ~]$ gpg --armor --export 2783037481DCFEF7 | xclip -sel clip
```

Рис. 3.6: Вывод списка ключей

Мы видим добавленный ключ GPG на GitHub (рис. 7).

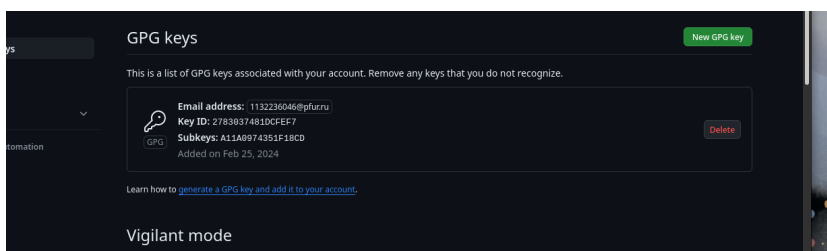


Рис. 3.7: Добавленный ключ GPG

3.7 Настроить подписи Git

Настраиваю автоматические подписи коммитов git (рис. 8).

```
[evkuzmin@fedora ~]$ git config --global user.signingkey 27830274810CFE77
[evkuzmin@fedora ~]$ git config --global commit.gpgsign true
[evkuzmin@fedora ~]$ git config --global gpg.program $(which gpg2)
[evkuzmin@fedora ~]$
```

Рис. 3.8: Настройка подписей Git

3.8 Настройка gh

Начинаю авторизацию в `gh`, отвечаю на вопросы, в конце выбираю авторизоваться через браузер (рис. 9).

```
[evkuzmin@fedora ~]$ gh auth login
? What account do you want to log into? GitHub.com
✓ You're already logged into github.com. Do you want to re-authenticate? Yes
? What is your preferred protocol for git operations? https
? How would you like to authenticate GitHub CLI? Login with a web browser
First copy your one-time code: C929-6803
Press Enter to open github.com in your browser...
```

Рис. 3.9: Авторизация в `gh`

Завершаю авторизацию на сайте (рис. 10).

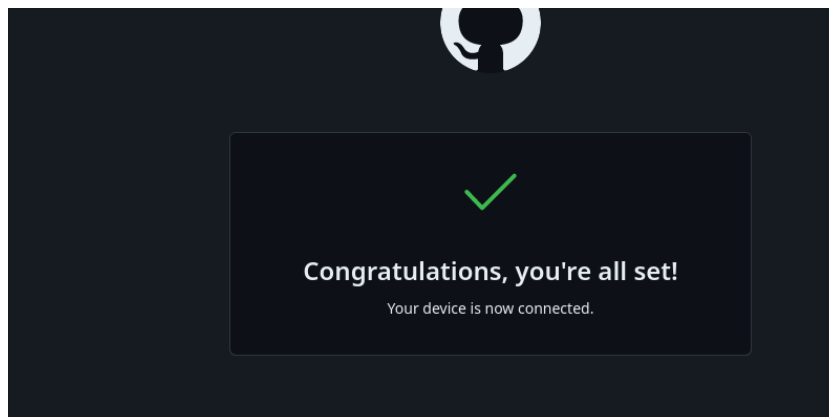


Рис. 3.10: Завершение авторизации через браузер

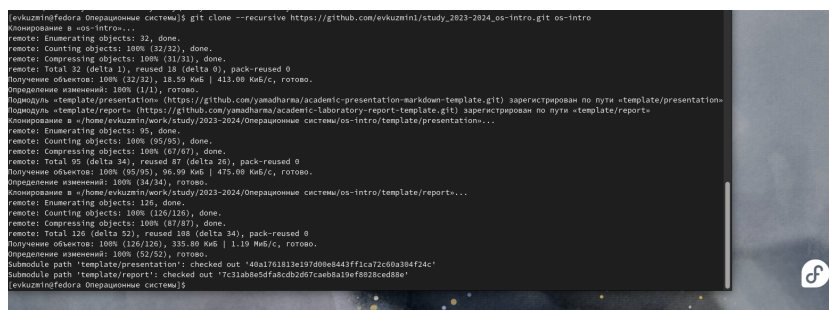
Вижу сообщение о завершении авторизации (рис. 11).

```
Press Enter to open github.com in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as evkuzmin1
[evkuzmin@fedora ~]$
```

Рис. 3.11: Завершение авторизации

3.9 Создание репозитория курса на основе шаблона

Сначала создаю директорию с помощью утилиты `mkdir` и флага `-p`. После этого с помощью утилиты `cd` перехожу в только что созданную директорию “Операционные системы”. Далее в терминале ввожу команду `gh repo create study_2022-2023_os-intro --template yamadharma/course-directory-student-trmplate --public`, чтобы создать репозиторий на основе шаблона репозитория. После этого клонирую репозиторий к себе в директорию, я указываю ссылку с протоколом `https`, а не `ssh`, потому что при авторизации в `gh` был выбран протокол `https` (рис. 12).



```
evkuzn1@fedora: Операционные системы$ git clone --recursive https://github.com/evkuzn1/study_2023-2024_os-intro.git os-intro
Клонирование в os-intro...
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 32 (delta 1), reused 18 (delta 0), pack-reused 0
Получение объектов: 100% (32/32), 18.59 KiB | 413.00 KiB/c, готово.
Определение изменений: 100% (1/1), готово.
Подсудный «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по нуте «template/presentation»
Подсудный «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по нуте «template/report»
Клонирование в /home/evkuzn1/work/study/2023-2024/Операционные системы/os-intro/template/presentation...
remote: Enumerating objects: 95, done.
remote: Counting objects: 100% (85/85), done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 95 (delta 34), reused 87 (delta 26), pack-reused 0
Получение объектов: 100% (95/95), 96.99 KiB | 475.00 KiB/c, готово.
Определение изменений: 100% (34/34), готово.
Клонирование в /home/evkuzn1/work/study/2023-2024/Операционные системы/os-intro/template/report...
remote: Enumerating objects: 126, done.
remote: Counting objects: 100% (126/126), done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 126 (delta 52), reused 188 (delta 34), pack-reused 0
Получение объектов: 100% (126/126), 335.80 KiB | 1.19 MiB/c, готово.
Определение изменений: 100% (32/32), готово.
Submodule path 'template/presentation': checked out '40a1761813e197d06e9443f1ca72c60a384f24c'
Submodule path 'template/report': checked out '7c31ab8e5d5f8cd267cabb8a19ef8828cd88e'
evkuzn1@fedora: Операционные системы$
```

Рис. 3.12: Создание репозитория

Ввожу фразу-пароль, установленную ранее (рис. 013).

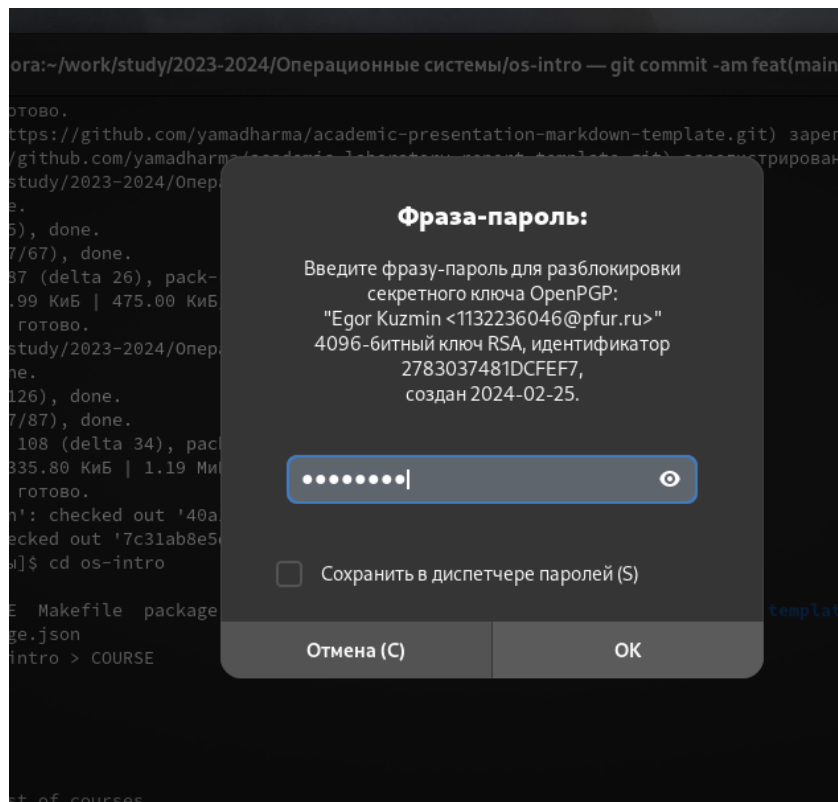


Рис. 3.13: Подтверждение пароля

Перехожу в каталог курса с помощью утилиты `cd`, там я удаляю лишние файлы с помощью утилиты `rm`, далее создаю необходимые каталоги используя `make`. В конце концов добавляю все новые файлы для отправки на сервер (сохраняю добавленные изменения) с помощью команды `git add` и комментирую их с помощью `git commit` (рис. 14).

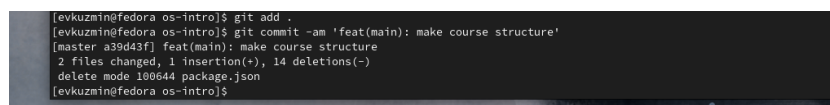


Рис. 3.14: Работа с каталогами

Отправляю файлы на сервер с помощью `git push` (рис. 015).

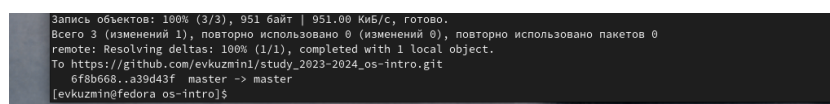


Рис. 3.15: Отправка файлов на сервер

4 Выводы

При выполнении данной лабораторной работы я приобрел практические навыки по применению средств контроля версий и работе с git.

5 Ответы на контрольные вопросы.

1. Системы контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Они позволяют хранить несколько версий изменяющейся информации, одного и того же документа, может предоставить доступ к более ранним версиям документа. Используется для работы нескольких человек над проектом, позволяет посмотреть, кто и когда внес какое-либо изменение и т. д. VCS применяются для: Хранения полной истории изменений, сохранения причин всех изменений, поиска причин изменений и совершивших изменение, совместной работы над проектами.
2. Хранилище – репозиторий, хранилище версий, в нем хранятся все документы, включая историю их изменения и прочей служебной информацией. commit – отслеживание изменений, сохраняет разницу в изменениях. История – хранит все изменения в проекте и позволяет при необходимости вернуться/обратиться к нужным данным. Рабочая копия – копия проекта, основанная на версии из хранилища, чаще всего последней версии.
3. Централизованные VCS (например: CVS, TFS, AccuRev) – одно основное хранилище всего проекта. Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет, затем добавляет изменения обратно в хранилище. Децентрализованные VCS (например: Git, Bazaar) – у каждого пользователя свой вариант репозитория (возможно несколько вариантов), есть возможность добавлять и забирать

изменения из любого репозитория. В отличие от классических, в распределенных (децентрализованных) системах контроля версий центральный репозиторий не является обязательным.

4. Сначала создается и подключается удаленный репозиторий, затем по мере изменения проекта эти изменения отправляются на сервер.
5. Участник проекта перед началом работы получает нужную ему версию проекта в хранилище, с помощью определенных команд, после внесения изменений пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются. К ним можно вернуться в любой момент.
6. Хранение информации о всех изменениях в вашем коде, обеспечение удобства командной работы над кодом.
7. 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 изменений: `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`

8. `git push -all` отправляем из локального репозитория все сохраненные изменения в центральный репозиторий, предварительно создав локальный репозиторий и сделав предварительную конфигурацию.
9. Ветвление - один из параллельных участков в одном хранилище, исходящих из одной версии, обычно есть главная ветка. Между ветками, т. е. их концами возможно их слияние. Используются для разработки новых функций.
10. Во время работы над проектом могут создаваться файлы, которые не следуют добавлять в репозиторий. Например, временные файлы. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл `.gitignore` с помощью сервисов.

Список литературы

Архитектура компьютеров и ОС/Электронный ресурс