

Отчет по лабораторной работе № 2

Операционные системы

Чувакина Мария Владимировна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Базовая настройка git.	7
3.2	Создание ключа SSH.	8
3.3	Создание ключа GPG.	8
3.4	Регистрация на Github.	10
3.5	Добавление ключа GPG в Github.	10
3.6	Настройка подписи Git.	11
3.7	Настройка gh.	12
3.8	Создание репозитория курса на основе шаблона.	12
4	Выводы	15
5	Ответы на контрольные вопросы.	16

Список иллюстраций

3.1	Задаю имя и email владельца репозитория	7
3.2	Настройка utf-8 в выводе сообщений git	7
3.3	Задаю имя начальной ветки	7
3.4	Задаю параметры autocrlf и safecrlf	7
3.5	Генерация ssh ключа по алгоритму rsa	8
3.6	Генерация ssh ключа по алгоритму ed25519	8
3.7	Генерация ключа	9
3.8	Защита ключа GPG	9
3.9	Вывод списка ключей	10
3.10	Копирование ключа в буфер обмена	10
3.11	Настройки Github	11
3.12	Добавление нового PGP ключа	11
3.13	Настройка подписей Git	11
3.14	Авторизация в gh	12
3.15	Создание репозитория	13
3.16	Перемещение между директориями	13
3.17	Удаление файлов и создание каталогов	13
3.18	Отправка файлов на сервер	14

Список таблиц

1 Цель работы

Цель данной лабораторной работы – изучение идеологии и применение средств контроля, освоение умения по работе с git.

2 Задание

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Настроить подписи git.
5. Зарегистрироваться на Github.
6. Создать локальный каталог для выполнения заданий по предмету.

3 Выполнение лабораторной работы

3.1 Базовая настройка git.

Задаю в качестве имени и email владельца репозитория свои имя, фамилию и электронную почту (рис.1).

```
mvchuvakina@dk2n24 ~ $ git config --global user.name "Maria Chuvakina"
mvchuvakina@dk2n24 ~ $ git config --global user.email "maria.chuvakina@mail.ru"
```

Рис. 3.1: Задаю имя и email владельца репозитория

Настраиваю utf-8 в выводе сообщений git для их корректного изображения (рис.2).

```
mvchuvakina@dk2n24 ~ $ git config --global core.quotepath false
```

Рис. 3.2: Настройка utf-8 в выводе сообщений git

Начальной ветке задаю имя master (рис.3).

```
mvchuvakina@dk2n24 ~ $ git config --global init.defaultBranch master
```

Рис. 3.3: Задаю имя начальной ветки

Задаю параметры autocrlf и safecrlf для корректного отображения конца строки (рис.4).

```
mvchuvakina@dk2n24 ~ $ git config --global core.autocrlf input
mvchuvakina@dk2n24 ~ $ git config --global core.safecrlf warn
```

Рис. 3.4: Задаю параметры autocrlf и safecrlf

3.2 Создание ключа SSH.

Создаю ключ ssh размером 4096 бит по алгоритму rsa (рис.5).

```
mvchuvakina@dk2n24 ~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/m/v/mvchuvakina/.ssh/id_rsa):
/afs/.dk.sci.pfu.edu.ru/home/m/v/mvchuvakina/.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 /afs/.dk.sci.pfu.edu.ru/home/m/v/mvchuvakina/.ssh/id_rsa
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/m/v/mvchuvakina/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:K0Sxq951oAVeYyJgha5ZcE2o7voBqJM0rx5mGC0dqEg mvchuvakina@dk2n24
The key's randomart image is:
+---[RSA 4096]-----+
|  . . . |
|  .+  o |
| ..o + = + |
|o* + + * . |
|OEB + S |
|B@  o o o |
|%+.  o o . |
|+=o . o . |
|-=o . . |
+---[SHA256]-----+
mvchuvakina@dk2n24 ~$
```

Рис. 3.5: Генерация ssh ключа по алгоритму rsa

Создаю ключ ssh по алгоритму ed25519 (рис.6).

```
+---[SHA256]-----+
mvchuvakina@dk2n24 ~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/m/v/mvchuvakina/.ssh/id_ed25519):
/afs/.dk.sci.pfu.edu.ru/home/m/v/mvchuvakina/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/m/v/mvchuvakina/.ssh/id_ed25519
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/m/v/mvchuvakina/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:fYpvJ9QGV23+aEhpyymajp1C8WW4mk9x00Ss/LPLR0Q mvchuvakina@dk2n24
The key's randomart image is:
+---[ED25519 256]---+
|  . |
|  o |
|  o o |
|  .*.o+ = |
|  oE+oB.+ .. |
|  ..oo=ooB o . |
|  .oo++o . |
|  +B+oo . |
|  .+X=.o |
+---[SHA256]-----+
mvchuvakina@dk2n24 ~$
```

Рис. 3.6: Генерация ssh ключа по алгоритму ed25519

3.3 Создание ключа GPG.

Генерирую ключ GPG, затем выбираю тип ключа RSA and RSA, задаю максимальную длину ключа: 4096, оставляю неограниченный срок действия ключа.

Далее отвечаю на вопросы программы о личной информации. (рис.7).

```
[GnuPG]
mvchuvakina@dk2n24 ~ $ gpg --full-generate-key
gpg (GnuPG) 2.2.42; Copyright (C) 2023 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Выберите тип ключа:
  (1) RSA и RSA (по умолчанию)
  (2) DSA и Elgamal
  (3) DSA (только для подписи)
  (4) RSA (только для подписи)
  (14) Имеющийся на карте ключ
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (3072) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
  0 = не ограничен
  <n> = срок действия ключа - n дней
  <n>w = срок действия ключа - n недель
  <n>m = срок действия ключа - n месяцев
  <n>y = срок действия ключа - n лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (у/Н) у

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Мария
Адрес электронной почты: maria.chuvakina@mail.ru
Примечание:
Используется таблица символов 'utf-8'.
Вы выбрали следующий идентификатор пользователя:
  "Мария <maria.chuvakina@mail.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? O
```

Рис. 3.7: Генерация ключа

Ввожу фразу-пароль для защиты нового ключа (рис.8).

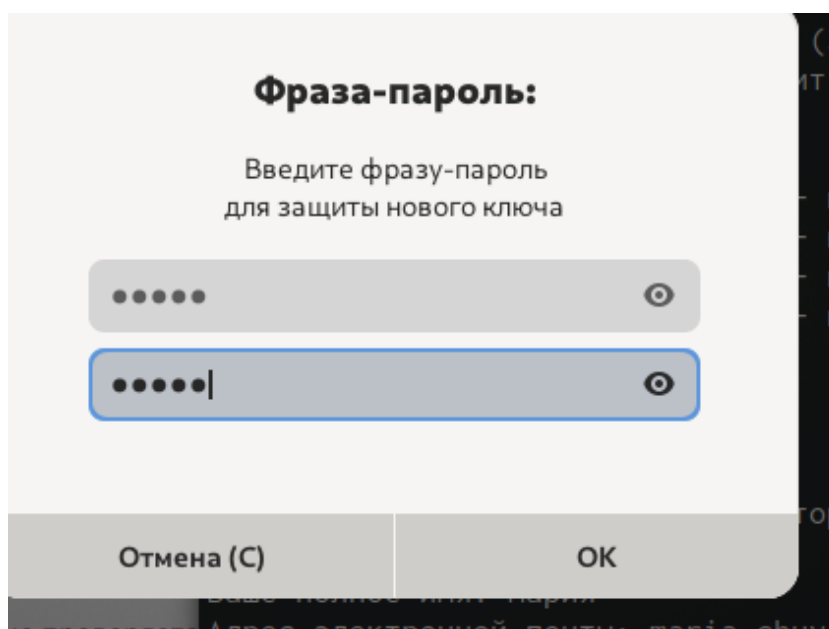


Рис. 3.8: Защита ключа GPG

3.4 Регистрация на Github.

У меня уже был создан аккаунт на Github, соответственно, основные данные аккаунта я так же заполняла и проводила его настройку, поэтому просто захожу в свой аккаунт.

3.5 Добавление ключа GPG в Github.

Вывожу список созданных ключей в терминал, ищу в результате запроса отпечаток ключа (последовательность байтов для идентификации более длинного, по сравнению с самим отпечатком, ключа), он стоит после знака слеша, копирую его в буфер обмена. (рис.9).

```
mvchuvakina@dk2n24 ~$ 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: 2 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 2u
/afs/.dk.sci.pfu.edu.ru/home/m/v/mvchuvakina/.gnupg/pubring.kbx
-----
sec   rsa4096/7E6C02CF7752B664 2024-02-14 [SC]
      91A318EC18AF051CB3DAA9307E6C02CF7752B664
uid   [ абсолютно ] Мария <maria.chuvakina@mail.ru>
ssb   rsa4096/8F9D4021E3571F25 2024-02-14 [E]

sec   rsa4096/05AC2C39D42A923E 2024-02-16 [SC]
      B94BCC4EE49A2FF5B5E17B6505AC2C39D42A923E
uid   [ абсолютно ] Мария <maria.chuvakina@mail.ru>
ssb   rsa4096/B69163433E62CA1B 2024-02-16 [E]
```

Рис. 3.9: Вывод списка ключей

Ввожу в терминале команду, с помощью которой копирую сам ключ GPG в буфер обмена, за это отвечает утилита xclip. (рис.10).

```
mvchuvakina@dk2n24 ~$ gpg --armor --export <PGP Fingerprint> | xclip -sel clip
bash: синтаксическая ошибка рядом с неожиданным маркером «|»
mvchuvakina@dk2n24 ~$ gpg --armor --export 7E6C02CF7752B664 | xclip -sel clip
mvchuvakina@dk2n24 ~$
```

Рис. 3.10: Копирование ключа в буфер обмена

Открываю настройки Github, ищу среди них добавление GPG ключа (рис.11).

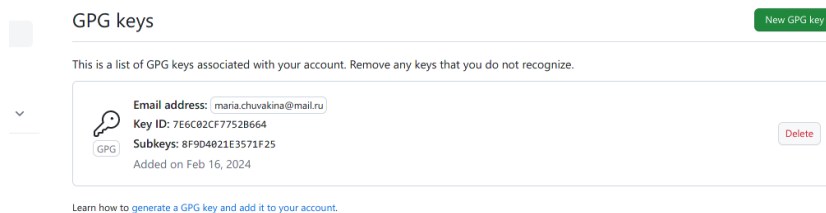


Рис. 3.11: Настройки Github

Нажимаю на “New GPG key” и вставляю в поле ключ из буфера обмена (рис.12).

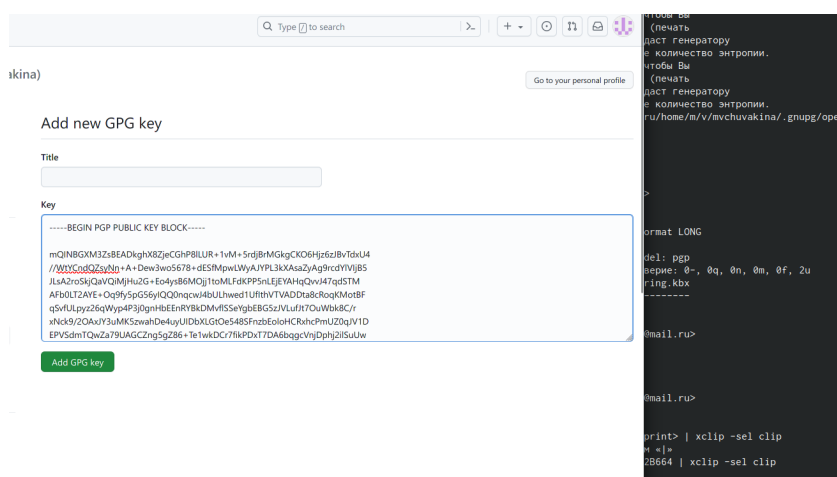


Рис. 3.12: Добавление нового PGP ключа

Я добавила ключ GPG на Github.

3.6 Настройка подписи Git.

Настраиваю автоматические подписи коммитов git: используя введенный ранее email, указываю git, использую его при создании подписей коммитов.(рис.13).

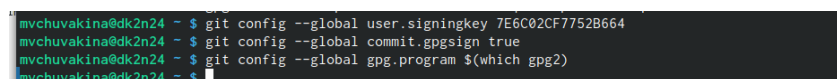


Рис. 3.13: Настройка подписей Git

3.7 Настройка gh.

Начинаю авторизацию в gh, отвечаю на наводящие вопросы от утилиты, в конце выбираю авторизоваться через браузер. Далее вижу сообщение о завершении авторизации под именем mvchuvakina. (рис.14).

```
mvchuvakina@dk2n24 ~/work/study/2023-2024/Операционные системы $ cd
mvchuvakina@dk2n24 ~ $ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: EE61-F5AC
Press Enter to open github.com in your browser...
[14179:14235:0214/185300.170636:ERROR:nss_util.cc(357)] After loading Root Certs, loaded==false: NSS error code: -8018
[14179:14179:0214/185313.240145:ERROR:object_proxy.cc(576)] Failed to call method: org.freedesktop.ScreenSaver.GetActive: object_path= /org/freedesktop/ScreenSaver: org.freedesktop.DBus.Error.NotSupported: This method is not part of the idle inhibition specification: https://specifications.freedesktop.org/idle-inhibit-spec/latest/
[14179:14179:0214/185313.336254:ERROR:object_proxy.cc(576)] Failed to call method: org.gnome.ScreenSaver.GetActive: object_path= /org/gnome/ScreenSaver: org.freedesktop.DBus.Error.ServiceUnknown: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name org.gnome.Shell.ScreenShield was not provided by any .service files
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
```

Рис. 3.14: Авторизация в gh

3.8 Создание репозитория курса на основе шаблона.

Сначала создаю директорию с помощью утилиты mkdir и флага -p, который позволяет установить каталоги на всем указанном пути. После этого с помощью утилиты cd перехожу в созданную директорию. После этого клонирую репозиторий к себе в директорию (рис.15).

```

mvchuvakina@dk2n24 ~/work/study/2023-2024/Операционные системы $ git clone --recursive https://github.com/mvchuvakina/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), 18.60 Киб | 634.00 Киб/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/afs/.dk.sci.pfu.edu.ru/home/m/v/mvchuvakina/work/study/2023-2024/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 95, done.
remote: Counting objects: 100% (95/95), 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 Киб | 1.04 Миб/с, готово.
Определение изменений: 100% (34/34), готово.
Клонирование в «/afs/.dk.sci.pfu.edu.ru/home/m/v/mvchuvakina/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% (87/87), done.
remote: Total 126 (delta 52), reused 108 (delta 34), pack-reused 0
Получение объектов: 100% (126/126), 335.80 Киб | 1.32 Миб/с, готово.
Определение изменений: 100% (52/52), готово.
Submodule path 'template/presentation': checked out '40a1761813e197d00e8443ff1ca72c60a304f24c'
Submodule path 'template/report': checked out '7c31ab8e5dfa8cdb2d67caeb8a19ef8028ced88e'
mvchuvakina@dk2n24 ~/work/study/2023-2024/Операционные системы $

```

Рис. 3.15: Создание репозитория

Перехожу в каталог курса с помощью утилиты `cd`, проверяю содержание каталога с помощью утилиты `ls` (рис.16).

```

mvchuvakina@dk2n24 ~/work/study/2023-2024/Операционные системы $ cd os-intro
mvchuvakina@dk2n24 ~/work/study/2023-2024/Операционные системы/os-intro $ ls
CHANGELOG.md  config  COURSE  LICENSE  Makefile  README.en.md  README.git-flow.md  README.md  template
mvchuvakina@dk2n24 ~/work/study/2023-2024/Операционные системы/os-intro $

```

Рис. 3.16: Перемещение между директориями

Удаляю лишние файлы с помощью утилиты `rm`, далее создаю необходимые каталоги используя `makefile` (рис.17).

```

mvchuvakina@dk2n24 ~/work/study/2023-2024/Операционные системы/os-intro $ echo os-intro > COURSE
mvchuvakina@dk2n24 ~/work/study/2023-2024/Операционные системы/os-intro $ make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare       Generate directories structure
  submodule      Update submodules
mvchuvakina@dk2n24 ~/work/study/2023-2024/Операционные системы/os-intro $ git add

```

Рис. 3.17: Удаление файлов и создание каталогов

Добавляю все новые файлы для отправки на сервер с помощью команды `git add` и комментирую из с помощью `git commit`. Отправляю файлы на сервер с помощью `git push` (рис.18).

```

mvchuvakina@dk2n24 ~/work/study/2023-2024/Операционные системы/os-intro $ git add .
mvchuvakina@dk2n24 ~/work/study/2023-2024/Операционные системы/os-intro $ git commit -am 'feat(main): make course structure'
[master d88e8ba] feat(main): make course structure
 2 files changed, 1 insertion(+), 14 deletions(-)
 delete mode 100644 package.json
mvchuvakina@dk2n24 ~/work/study/2023-2024/Операционные системы/os-intro $ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 285 байтов | 285.00 КиБ/с, готово.
Всего 3 (изменений 1), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/mvchuvakina/study_2023-2024_os-intro.git
   0e1f392..d88e8ba  master -> master
mvchuvakina@dk2n24 ~/work/study/2023-2024/Операционные системы/os-intro $

```

Рис. 3.18: Отправка файлов на сервер

4 Выводы

При выполнении данной лабораторной работы я изучила идеологию и применение средств контроля версий, освоила умение по работе с git.

5 Ответы на контрольные вопросы.

1. Системы контроля версий (VCS) - программное обеспечение работы с изменяющейся информацией. Они позволяют хранить несколько версий изменяющейся информации, одного и того же документа, может предоставлять 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 и 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 (децентрали-

зованных) 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. Используется для разработки новых функций.
10. Во время работы над проектом могут создаваться файлы, которые не следует добавлять в репозиторий. Например, временные файлы. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл `.gitignore` с помощью сервисов.