

Отчёт по лабораторной работе №2

Отчет подготовил:
Архипов Александр Сергеевич
группы НБИбд-03

Содержание

- 1. Цель работы
- 2. Теоретические знания, которые пригодятся.
- 3. Выполнение лабораторной работы
 - 1. Установка git на виртуальную машину
 - 2. Создание SSH ключа
- 4. Задание

Цель работы

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

Теоретические знания, которые пригодятся

- *Системы контроля версий (Version Control System, VCS)* применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.

Примеры использования git

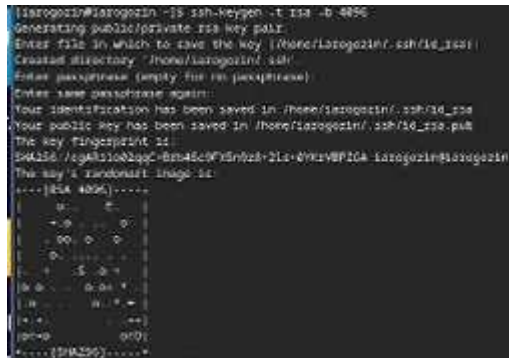
- Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` с различными опциями.
- Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.

Выполнение лабораторной работы

- 1. Зададим имя и email владельца репозитория:
 - `git config --global user.name "Name Surname"` `git config --global user.email "work@mail"`
- 2. Настроим utf-8 в выводе сообщений git:
 - `git config --global core.quotePath false`
- 3. Зададим имя начальной ветки (будем называть её master):
 - `git config --global init.defaultBranch master`

2. Создание SSH ключа

- 4. Создаём ключи:
 - по алгоритму *rsa* с ключём размером 4096 бит:
 - `ssh-keygen -t rsa -b 4096`
 - по алгоритму *ed25519*:
 - `ssh-keygen -t ed25519`

A terminal window showing the execution of the 'ssh-keygen' command. The user enters 'ssh-keygen -t rsa -b 4096'. The terminal prompts for a file to save the key, a directory to create, and a passphrase. The user enters '/home/larogozin/.ssh' for the directory and an empty passphrase. The terminal confirms the key generation and displays the key fingerprint and the path to the public key file.

```
larogozin@larogozin:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/larogozin/.ssh/id_rsa):
Create a directory? [y/n] y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/larogozin/.ssh/id_rsa
Your public key has been saved in /home/larogozin/.ssh/id_rsa.pub
The key fingerprint is:
SHA256: /GK1100290C-Bm460F3dnt28-21a-0YrvBP10A-larogozin@larogozin
The key's randomart image is:
[256 4096]
+-----+
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
+-----+
[SHA256].....
```

2.1 Создаём ключи по двум алгоритмам

- 5. Скопируем созданный SSH-ключ в буфер обмена командой:
`xclip -i < ~/.ssh/id_ed25519.pub`
Далее откроем настройки своего аккаунта на GitHub и перейдем в раздел SSH and GPG keys.
Нажмём кнопку new SSH key.
Добавим в поле Title название этого ключа, например, ed25519@hostname.
Вставим из буфера обмена в поле Key ключ.
Нажмём кнопку Add SSH key.

Верификация коммитов с помощью *PGP*

- 6. Генерируем ключ
`gpg --full-generate-key`
- Из предложенных опций выбираем:
 - тип *RSA and RSA*;
 - размер 4096;
 - выберите срок действия; значение по умолчанию — 0 (срок действия не истекает никогда).
- GPG запросит личную информацию, которая сохранится в ключе:
 - Имя (не менее 5 символов).
 - Адрес электронной почты.
- 7. Выводим список ключей и копируем отпечаток приватного ключа:
`gpg --list-secret-keys --keyid-format LONG`
- Экспортируем ключ в формате ASCII по его отпечатку:
`gpg --armor --export <PGP Fingerprint>`

Подписывание КОММИТОВ git

- 8. Используя введённый email, укажите Git применять его при подписи КОММИТОВ:

```
git config --global user.signingkey <PGP Fingerprint>
```

```
git config --global commit.gpgsign true
```

```
git config --global gpg.program $(which gpg2)
```


Режим бдительности (vigilant mode)

- 9. На GitHub есть настройка vigilant mode. Включается это в настройках в разделе *SSH and GPG keys*. Установим метку на *Flag unsigned commits as unverified*.

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.

Выполнение

- 1. Установим *git*:
`dnf install git`
- 2. Установим *gh*
`dnf install gh`
- 3. Для начала необходимо авторизоваться
`gh auth login`

Создание репозитория курса на основе шаблона

- **4.** `mkdir -p ~/work/study/2022-2023/"Операционные системы"`
`cd ~/work/study/2022-2023/"Операционные системы"`
`gh repo create study_2022-2023_os-intro --`
`template=yamadharm/course-directory-student-template --public`
`git clone --recursive git@github.com:<owner>/study_2022-2023_os-`
`intro.git os-intro`

Настройка каталога курса

- 5. Перейдём в каталог курса:

```
cd ~/work/study/2022-2023/"Операционные системы"/os-intro
```

- 6. Удалим лишние файлы:

```
rm package.json
```

- Создадим необходимые каталоги:

```
echo os-intro > COURSE
```

```
make
```

- Отправим файлы на сервер:

```
git add .
```

```
git commit -am 'feat(main): make course structure'
```

```
git push
```

Вывод

- Сегодня я научился создавать репозиторий на гитхабе и настраивать его для лабораторных работ.