Лабораторная работа №1

Управление версиями

Хохлачева Яна Дмитриевна, НПМмд-02-22

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 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

3 Выполнение лабораторной работы

3.1 Настройка github

Произвела первоначальную настройку github:

1. Создала учётную запись на https://github.com.
2. Заполнила основные данные на https://github.com.

3.2 Базовая настройка git

- Задала имя и email владельца репозитория:

Рис. 3.1: Username и email

- Настроила utf-8 в выводе сообщений git:
- Настроила верификацию и подписание коммитов git, задала имя начальной ветки (будем называть её master), параметр autocrlf, параметр safecrlf:

```
Yana@LAPTOP-BBI48HQD MINGW64 ~/Desktop
$ git config --global core.quotepath false

Yana@LAPTOP-BBI48HQD MINGW64 ~/Desktop
$ git config --global init.defaultBranch master

Yana@LAPTOP-BBI48HQD MINGW64 ~/Desktop
$ git config --global core.autocrlf input

Yana@LAPTOP-BBI48HQD MINGW64 ~/Desktop
$ git config --global core.safecrlf warn
```

Рис. 3.2: Дополнительные параметры

3.3 Создайте ключи ssh

- по алгоритму rsa с ключём размером 4096 бит:

```
$ ssh-keygen -t rsa -C "Yana Khokhlacheva"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Yana/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Yana/.ssh/id_rsa
Your public key has been saved in /c/Users/Yana/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:IFDtw+UKSFNKMNN6kBdnZHwyYhDgEOwE5WSBaAXHeDA Yana Khokhlacheva
The key's randomart image is:
+---[RSA 3072]----+
|/E#XB.           |
|B%XO+ o .        |
|+B++.*.o         |
|..o ..+..        |
| .   . oS        |
|      .          |
|                 |
|                 |
|                 |
+----[SHA256]-----+

Yana@LAPTOP-BBI48HQD MINGW64 ~/.ssh
$
```

Рис. 3.3: RSA SSH

- по алгоритму ed25519:

3.4 Создайте ключи pgp

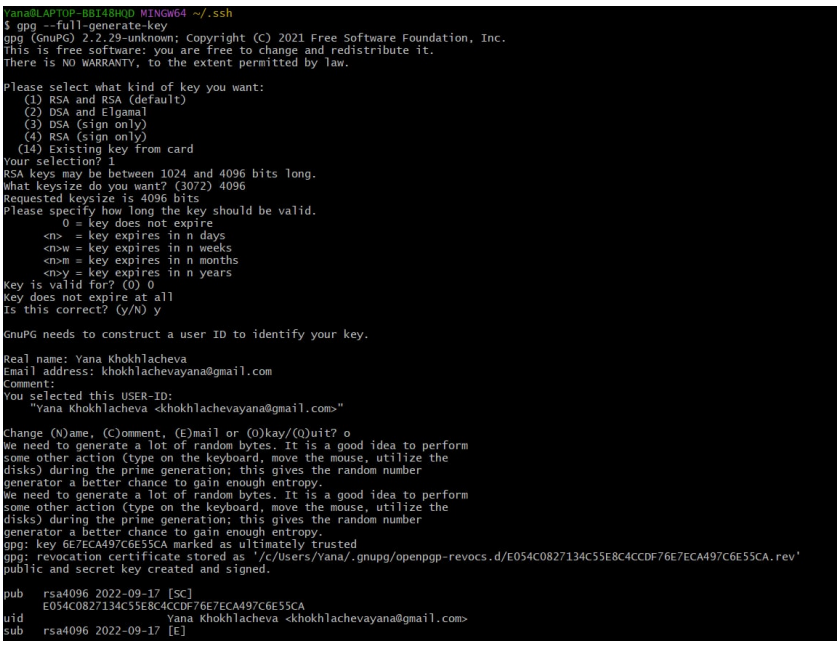

Рис. 3.4: GPG Key

3.5 Добавление PGP ключа в GitHub

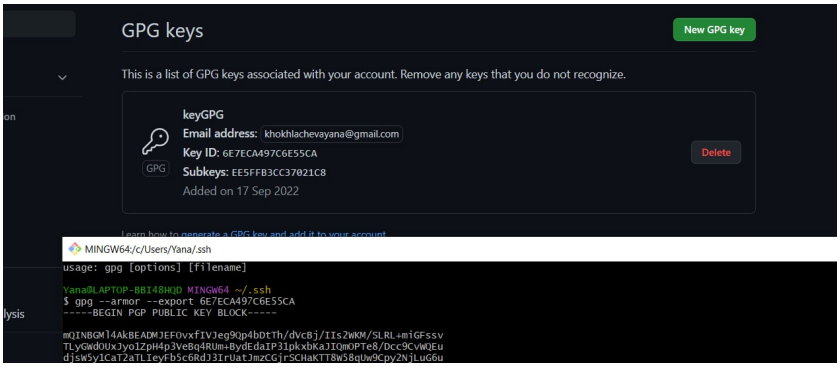

Рис. 3.5: PGP ключ в GitHub

3.6 Настройка автоматических подписей коммитов git

- Используя введёный email, указала Git применять его при подписи коммитов:

```
Yana@LAPTOP-BBI48HQD MINGW64 ~/.ssh
$ git config --global gpg.program "C:/Program Files (x86) /GnuPG/bin/gpg.exe"
```

Рис. 3.6: Подписи коммитов

```
Yana@LAPTOP-BBI48HQD MINGW64 ~/.ssh
$ git config --global user.signingkey 6E7ECA497C6E55CA

Yana@LAPTOP-BBI48HQD MINGW64 ~/.ssh
$ git config --global commit.gpgsign true
```

Рис. 3.7: Указание параметров

3.7 Настройка gh

- Процесс авторизации

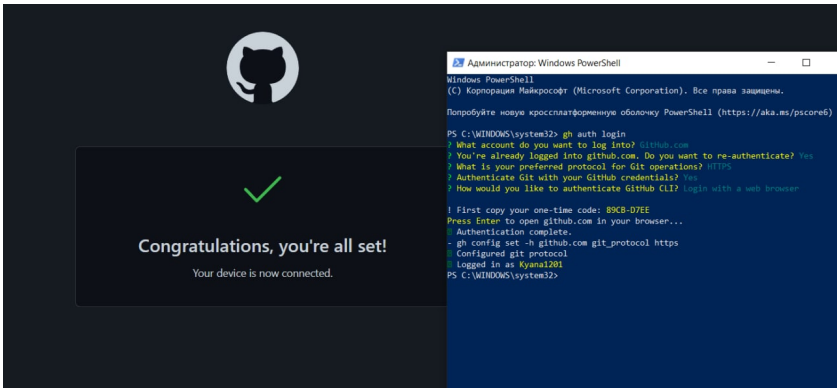

Рис. 3.8: gh auth

3.8 Сознание репозитория курса на основе шаблона

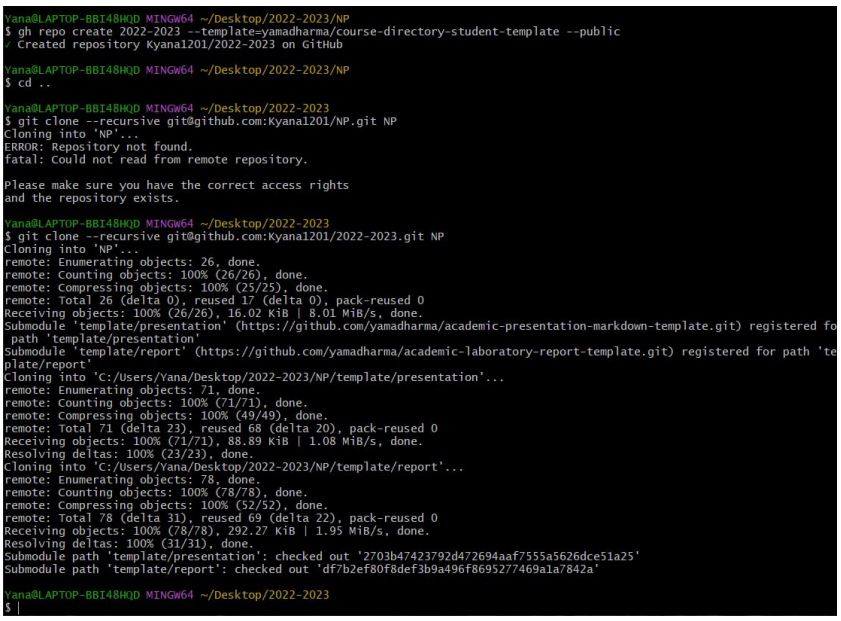

Рис. 3.9: Создание репозитория

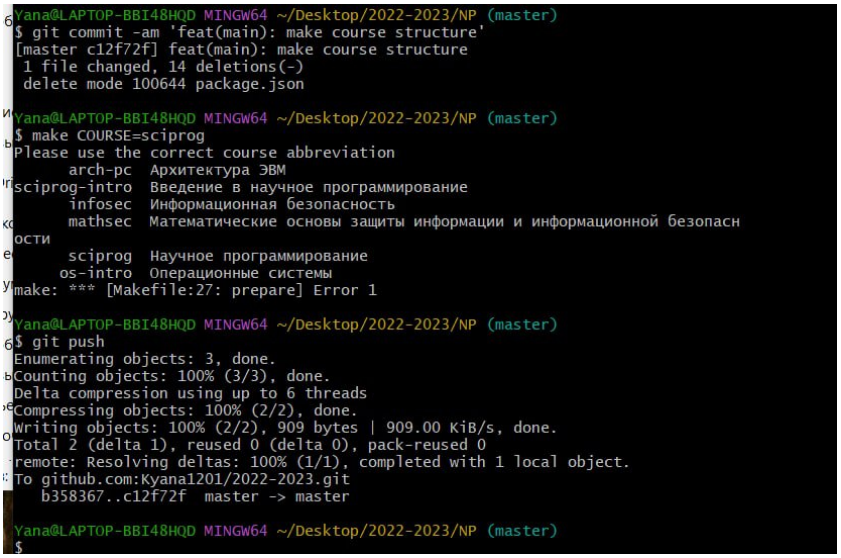

Рис. 3.10: Коммит в рабочую среду

4 Выводы

Таким образом в процессе лабораторной работы я изучила систему контроля версий git, ее идеологию и принципы.

5 Ответы на контрольные вопросы

1. Система контроля версий предназначена для ведения истории изменений. Каждое изменение добавляется через коммиты, и составляется дерево коммитов. В любой момент времени можно вернуться на любую ноду дерева

2. Хранилище - удаленный сервер, на котором хранится проект с гит файлами, commit - изменение в проект, которое затем должно быть подтверждено командой git push. История - дерево всех коммитов. Рабочая копия - создается с помощью git clone, копия на локальной машине, в которую вносятся изменения. Они могут быть загружены на сервер через коммиты.

3. Централизованные системы используют единственный сервер, содержащий все версии файлов, и некоторое количество клиентов, которые получают файлы из этого централизованного хранилища. Примеры: CVS, Subversion и Perforce.

4. При единоличной работе с хранилищем применяются такие же правила как и при работе с общим хранилищем.

5. При работе с общим хранилищем необходимо для каждой функции строго добавлять новую ветку feature, реализовывать её и слиять с веткой develop.

6. Защищает исходный код от потери, обеспечивает командную работу, помогает отменить изменения, распределённая работа.

7. git add, git commit, git push, git remote, git clone, git flow, git branch, git merge, git checkout, git pull, git init, git config

8. Если нужно вести систему контроля, но в целом мы не делимся кодом с командой, и нам не нужно иметь доступ к коду с разных устройств, которые практически никак не связаны с нашей локальной сетью, то можно использовать локальные репозитории. В противном случае нужно использовать удаленные репозитории.

9. Ветви - отдельные истории в СКВ, которые позволяют вести разработки параллельно. Над двумя ветками могут работать две разные комманды, а затем их можно слить в одну.

10. Через файл .gitignore. Потому что некоторые файлы могут быть слишком большими или отвечающими за определение конкретной локальной среды разработки.

Список литературы