

Лабораторная работа №1

Управление версиями

Хохлачева Яна Дмитриевна, НПМмд-02-22

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Настройка github	7
3.2	Базовая настройка git	7
3.3	Создайте ключи ssh	8
3.4	Создайте ключи pgr	9
3.5	Добавление PGP ключа в GitHub	9
3.6	Настройка автоматических подписей коммитов git	9
3.7	Настройка gh	10
3.8	Создание репозитория курса на основе шаблона	11
4	Выводы	12
5	Ответы на контрольные вопросы	13
	Список литературы	15

Список иллюстраций

3.1	Username и email	7
3.2	Дополнительные параметры	8
3.3	RSA SSH	8
3.4	GPG Key	9
3.5	PGP ключ в GitHub	9
3.6	Подписи коммитов	10
3.7	Указание параметров	10
3.8	gh auth	10
3.9	Создание репозитория	11
3.10	Коммит в рабочую среду	11

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

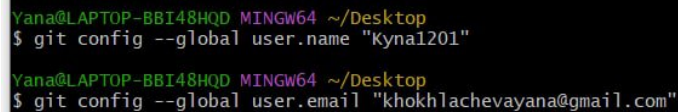
3.1 Настройка github

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:



```
Yana@LAPTOP-BBI48HQD MINGW64 ~/Desktop
$ git config --global user.name "Kyna1201"

Yana@LAPTOP-BBI48HQD MINGW64 ~/Desktop
$ git config --global user.email "khokhlachevayana@gmail.com"
```

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%X0+ o .       |
|+B++.*.o        |
|..O ..+..       |
|. . . oS         |
|                 |
|                 |
+---[SHA256]-----+
Yana@LAPTOP-BBI48HQD MINGW64 ~/.ssh
$

```

Рис. 3.3: RSA SSH

- по алгоритму ed25519:

3.4 Создайте ключи ргр

```
Yana@LAPTOP-BB148HQD MINGW64 ~/./ssh
$ gpg --full-generate-key
gpg (GnuPG) 2.2.29-unknown; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
(1) RSA and RSA (default)
(2) DSA and Elgamal
(3) DSA (sign only)
(4) RSA (sign only)
(14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 4096
Requested keysize is 4096 bits
Please specify how long the key should be valid.
    0 = key does not expire
    <n> = key expires in n days
    <n>w = key expires in n weeks
    <n>m = key expires in n months
    <n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: Yana Khokhlacheva
Email address: khokhlachevayana@gmail.com
Comments:
You selected this USER-ID:
    "Yana Khokhlacheva <khokhlachevayana@gmail.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key 6E7ECA497C6E55CA marked as ultimately trusted
gpg: revocation certificate stored as '/c:/Users/Yana/.gnupg/openpgp-revocs.d/E054C0827134C55E8C4CCDF76E7ECA497C6E55CA.rev'
public and secret key created and signed.

pub   rsa4096 2022-09-17 [SC]
      E054C0827134C55E8C4CCDF76E7ECA497C6E55CA
uid    Yana Khokhlacheva <khokhlachevayana@gmail.com>
sub    rsa4096 2022-09-17 [E]
```

Рис. 3.4: GPG Key

3.5 Добавление PGP ключа в GitHub

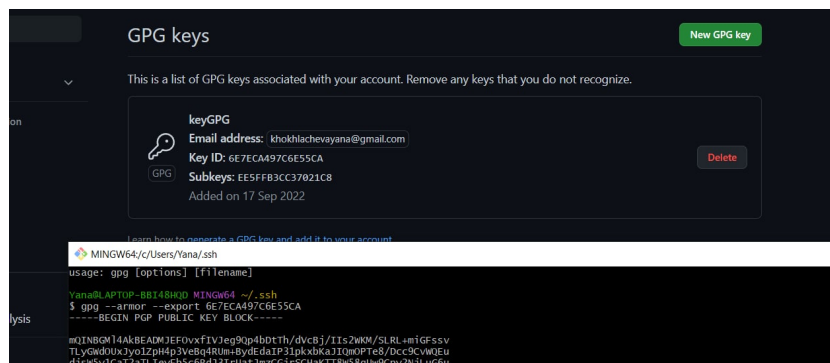


Рис. 3.5: PGP ключ в GitHub

3.6 Настройка автоматических подписей коммитов git

- Используя введённый 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

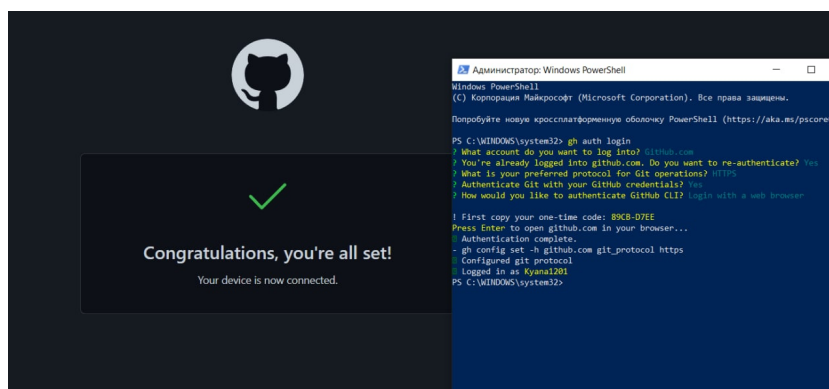


Рис. 3.8: gh auth

3.8 Создание репозитория курса на основе шаблона

```
Yana@LAPTOP-BBI48HQD MINGW64 ~/Desktop/2022-2023/NP
$ gh repo create 2022-2023 --template=yamadharma/course-directory-student-template --public
/ Created repository Kyana1201/2022-2023 on GitHub

Yana@LAPTOP-BBI48HQD MINGW64 ~/Desktop/2022-2023/NP
$ cd ..

Yana@LAPTOP-BBI48HQD MINGW64 ~/Desktop/2022-2023
$ git clone --recursive git@github.com:Kyana1201/NP.git NP
Cloning into 'NP'...
ERROR: Repository not found.
Fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

Yana@LAPTOP-BBI48HQD MINGW64 ~/Desktop/2022-2023
$ git clone --recursive git@github.com:Kyana1201/2022-2023.git NP
Cloning into 'NP'...
remote: Enumerating objects: 26, done.
remote: Counting objects: 100% (26/26), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 26 (delta 0), reused 17 (delta 0), pack-reused 0
Receiving objects: 100% (26/26), 16.02 KiB | 8.01 MiB/s, done.
Submodule 'template/presentation' (https://github.com/yamadharma/academic-presentation-markdown-template.git) registered for path 'template/presentation'
Submodule 'template/report' (https://github.com/yamadharma/academic-laboratory-report-template.git) registered for path 'template/report'
Cloning into 'C:/Users/Yana/Desktop/2022-2023/NP/template/presentation'...
remote: Enumerating objects: 71, done.
remote: Counting objects: 100% (71/71), done.
remote: Compressing objects: 100% (49/49), done.
remote: Total 71 (delta 23), reused 68 (delta 20), pack-reused 0
Receiving objects: 100% (71/71), 88.89 KiB | 1.08 MiB/s, done.
Resolving deltas: 100% (23/23), done.
Cloning into 'C:/Users/Yana/Desktop/2022-2023/NP/template/report'...
remote: Enumerating objects: 78, done.
remote: Counting objects: 100% (78/78), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 78 (delta 31), reused 69 (delta 22), pack-reused 0
Receiving objects: 100% (78/78), 292.27 KiB | 1.95 MiB/s, done.
Resolving deltas: 100% (31/31), done.
Submodule path 'template/presentation': checked out '2703b47423792d472694aaf7555a5626dce51a25'
Submodule path 'template/report': checked out 'df/b2ef80f8def3b9a496f8695277469a1a/842a'

Yana@LAPTOP-BBI48HQD MINGW64 ~/Desktop/2022-2023
$ |
```

Рис. 3.9: Создание репозитория

```
Yana@LAPTOP-BBI48HQD MINGW64 ~/Desktop/2022-2023/NP (master)
$ git commit -am 'feat(main): make course structure'
[master c12f72f] feat(main): make course structure
1 file changed, 14 deletions(-)
delete mode 100644 package.json

Yana@LAPTOP-BBI48HQD MINGW64 ~/Desktop/2022-2023/NP (master)
$ make COURSE=scipro
Please use the correct course abbreviation
arch-pc Архитектура ЭВМ
scipro-intro Введение в научное программирование
infosec Информационная безопасность
mathsec Математические основы защиты информации и информационной безопасн
ости
scipro Научное программирование
os-intro Операционные системы
make: *** [Makefile:27: prepare] Error 1

Yana@LAPTOP-BBI48HQD MINGW64 ~/Desktop/2022-2023/NP (master)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 6 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 909 bytes | 909.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:Kyana1201/2022-2023.git
b358367..c12f72f master -> master

Yana@LAPTOP-BBI48HQD MINGW64 ~/Desktop/2022-2023/NP (master)
$
```

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 сливать 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 команды, 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