

Отчет по лабораторной работе №2

Начало работы с github

Попова Елизавета Сергеевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Ответы на контрольные вопросы	12
4	Выводы	15
	Список литературы	16

Список иллюстраций

2.1	Подключение и настройка github	6
2.2	Создание ssh ключей	7
2.3	Генерация pgr ключа	8
2.4	Ввод данных для pgr ключа	8
2.5	Вывод pgr ключа	9
2.6	Настройка github	9
2.7	Подключение к github	10
2.8	Копируем данных с github	10
2.9	Изменение данных	10
2.10	Выгрузка данных на github	11

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

Поключаемся к github и настраиваем его (рис. 2.1)

```
[espopova@espopova ~]$ git config --global user.name "Lisa Popova"  
[espopova@espopova ~]$ git config --global user.email "timliza123@gmail.com"  
[espopova@espopova ~]$ git config --global core.quotepath false  
[espopova@espopova ~]$ git config --global init.defaultBranch master  
[espopova@espopova ~]$ git config --global core.autocrlf input  
[espopova@espopova ~]$ git config --global core.safecrlf warn  
[espopova@espopova ~]$ ssh-keygen -t rsa -b 4096
```

Рис. 2.1: Подключение и настройка github

Сгенерируем и выведем ssh ключи (рис. 2.2)

```
+---[RSA 4096]---+
|
|  .
|  . * .
|  . * S .
|  . o . + . o .
|  . + . + . E o . +
|  + = * B . o . + o + .
|  . * & = o . o o = + .
|
+-----[SHA256]-----+
[espopova@espopova ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/espopova/.ssh/id_ed25519):
/home/espopova/.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 /home/espopova/.ssh/id_ed25519.
Your public key has been saved in /home/espopova/.ssh/id_ed25519.pub.
The key fingerprint is:
SHA256:MYHr2VYrNF+I9dm+cBtVEwOv6cUqOKompML86RaVHbk espopova
The key's randomart image is:
+---[ED25519 256]---+
|
|  .o . .o .
|  .o . . .o
|  o . = o o o .o
|  o . E+oo + = .
|  . . +S+ o + +
|  o o + + o *
|  o o . . . + = +
|  . + o . . . . o
|  . + + o . .
|
+-----[SHA256]-----+
[espopova@espopova ~]$
```

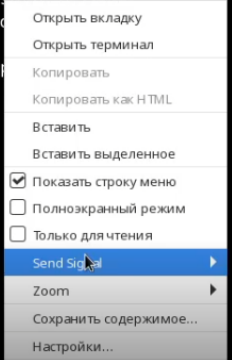


Рис. 2.2: Создание ssh ключей

Сгенерируем pgr ключ (рис. 2.3)

```
[espopova@espopova ~]$ gpg --full-generate-key
gpg (GnuPG) 2.3.8; 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.

Выберите тип ключа:
  (1) RSA and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) *default*
 (10) ECC (только для подписи)
 (14) Existing key from card
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? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.
```

Рис. 2.3: Генерация pgp ключа

Задаем данные для pgp ключа (рис. 2.4)

```
Ваше полное имя: Elizaveta
Адрес электронной почты: timliza123@gmail.com
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Elizaveta <timliza123@gmail.com>"
```

Рис. 2.4: Ввод данных для pgp ключа

Выведем наш pgp ключ (рис. 2.5)


```

pub  rsa4096 2023-02-17 [SC]
     D8DA04CEE8BF65AD9BC2DCEE8075E310BAF7EA95
uid      Elizaveta <timliza123@gmail.com>
sub  rsa4096 2023-02-17 [E]

[espopova@espopova ~]$ 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
/home/espopova/.gnupg/pubring.kbx
-----
sec  rsa4096/353DFC9F517A81E9 2023-02-17 [SC]
     5AD6E772FCB5B24C773BE8CE353DFC9F517A81E9
uid      [ абсолютно ] Elizaveta <timliza123@gmail.com>
ssb  rsa4096/AB7DCE94879BCD75 2023-02-17 [E]

sec  rsa4096/8075E310BAF7EA95 2023-02-17 [SC]
     D8DA04CEE8BF65AD9BC2DCEE8075E310BAF7EA95
uid      [ абсолютно ] Elizaveta <timliza123@gmail.com>
ssb  rsa4096/35D70B2524D42E55 2023-02-17 [E]

```

Рис. 2.5: Вывод pgp ключа

Продолжаем настройку github (рис. 2.6)

```

[espopova@espopova ~]$ gpg --armor --export 353DFC9F517A81E9 | xclip -sel clip
[espopova@espopova ~]$ git config --global user.signingkey 353DFC9F517A81E9
[espopova@espopova ~]$ git config --global commit.gpgsign true
[espopova@espopova ~]$ git config --global gpg.program $(which gpg2)
[espopova@espopova ~]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? 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: 4980-DE4B
Press Enter to open github.com in your browser...

```

Рис. 2.6: Настройка github

Подключение пришло успешно (рис. 2.7)

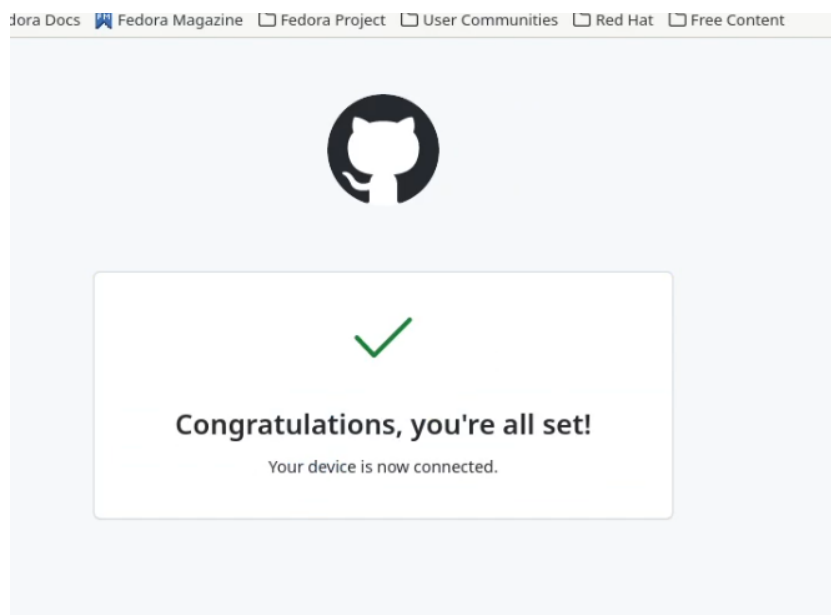


Рис. 2.7: Подключение к github

Копируем пространство для учебы в наш репозитори, а затем на виртуальную машину (рис. 2.8)

```
[esporova@esporova Операционные системы]$ git clone --recursive https://github.com/chistachill/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 27 (delta 1), reused 11 (delta 0), pack-reused 0
Получение объектов: 100% (27/27), 16.94 KiB | 2.42 MiB/c, 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 «/home/esporova/work/study/2022-2023/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 82 (delta 20), reused 72 (delta 23), pack-reused 0
Получение объектов: 100% (82/82), 92.90 KiB | 537.00 KiB/c, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/home/esporova/work/study/2022-2023/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 101 (delta 40), reused 80 (delta 27), pack-reused 0
Получение объектов: 100% (101/101), 327.25 KiB | 2.40 MiB/c, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be3800ee91f5809264cb755d316174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef1a3b1e3b2'
```

Рис. 2.8: Копируем данных с github

Редактируем данные (рис. 2.9)

```
[esporova@esporova Операционные системы]$ cd ~/work/study/2022-2023/"Операционные системы"/os-intro
[esporova@esporova os-intro]$ rm package.json
[esporova@esporova os-intro]$ echo os-intro > COURSE
[esporova@esporova os-intro]$ make
```

Рис. 2.9: Изменение данных

Выгружаем всё на github (рис. 2.10)

```
[espopova@espopova os-intro]$ git add .  
[espopova@espopova os-intro]$ git commit -am 'feat(main): make course structure'  
error: gpg не удалось подписать данные  
fatal: сбой записи объекта коммита  
[espopova@espopova os-intro]$ ^C  
[espopova@espopova os-intro]$ git commit -am 'feat(main): make course structure'  
error: gpg не удалось подписать данные  
fatal: сбой записи объекта коммита  
[espopova@espopova os-intro]$ git push
```

Рис. 2.10: Выгрузка данных на github

3 Ответы на контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Система управления версиями (также используется определение «система контроля версий», от англ. Version Control System, VCS или Revision Control System)— программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище (repository), или репозиторий, — место хранения файлов и их версий, служебной информации. Версия (revision), или ревизия,— состояние всего хранилища или отдельных файлов в момент времени («пункт истории»). Commit («трудовой вклад», не переводится) — процесс создания новой версии; иногда синоним версии. Рабочая копия (working copy) — текущее состояние файлов проекта (любой версии), полученных из хранилища и, возможно, измененных.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Децентрализованные VCS: У каждого пользователя свой вариант (возможно не один) репозитория. Присутствует возможность добавлять и забирать изменения из любого репозитория (Git, Mercurial, Bazaar). Централизованные VCS : Одно основное хранилище всего проекта. Каждый пользователь копирует

- себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно (Subversion, CVS, TFS, VAULT, AccuRev)
4. Опишите действия с VCS при единоличной работе с хранилищем.
 5. Опишите порядок работы с общим хранилищем VCS.
 6. Каковы основные задачи, решаемые инструментальным средством git? У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.
 7. Назовите и дайте краткую характеристику командам git. git init - создание репозитория git add (имена файлов) - Добавляет файлы в индекс git commit – выполняет коммит проиндексированных файлов в репозиторий git status – показывает какие файлы изменились между текущей стадией и HEAD. Файлы разделяются на 3 категории: новые файлы, измененные файлы, добавленные новые файлы git checkout (sha1 или метка) - получение указанной версии файла git push – отправка изменений в удаленный репозиторий git fetch – получение изменений из удаленного репозитория git clone (remote url) - клонирование удаленного репозитория себе
 8. Приведите примеры использования при работе с локальным и удалённым репозиториями.
 9. Что такое и зачем могут быть нужны ветви (branches)? Ветка (англ. branch) — это последовательность коммитов, в которой ведётся параллельная разработка какого-либо функционала Основная ветка– master Ветки в GIT. Показать все ветки, существующие в репозитории git branch. Создать ветку git branch имя. Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.
 10. Как и зачем можно игнорировать некоторые файлы при commit? Игнориру-

емые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты. Вот некоторые распространенные примеры таких файлов: кэши зависимостей, например содержимое `node_modules` или `packages`; скомпилированный код, например файлы `.o`, `.рус` и `.class` ; каталоги для выходных данных сборки, например `bin`, `out` или `target`; файлы, сгенерированные во время выполнения, например `.log`, `.lock` или `.tmp`; скрытые системные файлы, например `.DS_Store` или `Thumbs.db`; личные файлы конфигурации IDE, например `.idea.workspace.xml`.

4 Выводы

Мы научились работать и настраивать систему github

Список литературы