Отчёт по лабораторной работе №2

Первоначальная настройка git

Аскеров Александр Эдуардович

Содержание

Список иллюстраций

1 Цель работы

Изучить идеологию и применение средств контроля версий и 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 Базовая настройка git

Сначала сделаем предварительную конфигурацию git. Откроем терминал и введём следующие команды, указав имя и email владельца репозитория.

```
[aeaskerov@fedora ~]$ git config --global user.name "<Alexander Askerov>"
[aeaskerov@fedora ~]$ git config --global user.email "<iqwertydragoni@gmail.com>"
[aeaskerov@fedora ~]$
```

Рис. 3.1: Создаём предварительную конфигурацию git

Настроим utf-8 в выводе сообщений git.

```
[aeaskerov@fedora ~]$ git config --global core.quotepath false
[aeaskerov@fedora ~]$
```

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

Зададим имя начальной ветки (будем называть её master).

```
[aeaskerov@fedora ~]$ git config --global init.defaultBranch master
[aeaskerov@fedora ~]$
```

Рис. 3.3: Задаём имя начальной ветки

Параметр autocrlf.

```
[aeaskerov@fedora ~]$ git config --global core.autocrlf input
```

Рис. 3.4: Параметр autocrlf

Параметр safecrlf.

```
[aeaskerov@fedora ~]$ git config --global core.safecrlf warn
```

Рис. 3.5: Параметр safecrlf

3.2 Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый).

```
[aeaskerov@fedora ~]$ ssh-keygen -C "Alexander Askerov iqwertydragoni@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/aeaskerov/.ssh/id_rsa): keys
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in keys
Your public key has been saved in keys.pub
The key fingerprint is:
SHA256:NoQV3k8QjrHbwQFK2CFOnEOuvcOJqFf6OYsoBzVIT1w Alexander Askerov iqwertydrag
oni@gmail.com
The key's randomart image is:
+---[RSA 3072]----+
|   .o=Eo.=o+.    |
| . *=o.= B o     |
|..o o.o = = .    |
|. o+   . o +     |
| ....   S . .    |
|.. o.o . .       |
|...o=            |
|o.+..o           |
|+o..+o           |
+----[SHA256]-----+
```

Рис. 3.6: Генерируем ключи

Ключи должны сохраниться в каталоге ~/.ssh/.

Далее необходимо загрузить сгенерированный открытый ключ. Для этого зайдём на сайт http://github.org/ под своей учётной записью и перейдём в меню Setting. После этого выберем в боковом меню SSH and GPG keys и нажмём кнопку New SSH key. Скопировав из локальной консоли ключ в буфер обмена cat ~/.ssh/id_rsa.pub | xclip -sel clip.

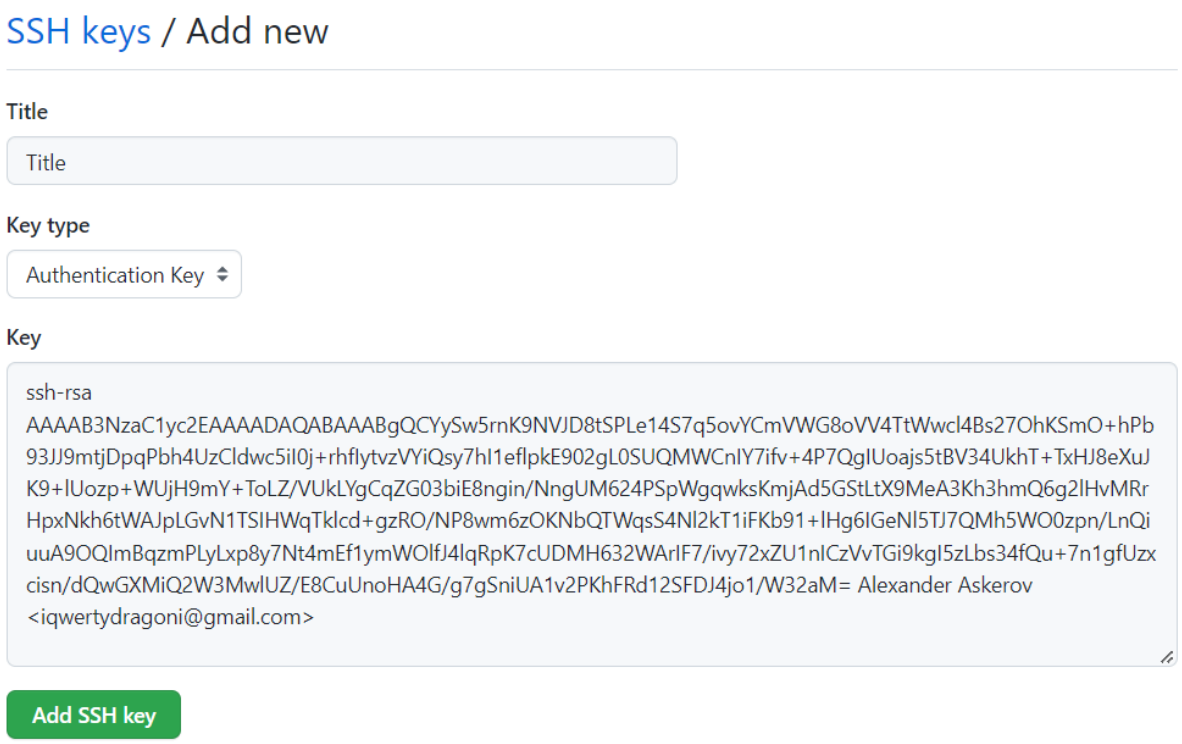

Рис. 3.7: Загружаем сгенерированный открытый ключ

3.3 Установка gh

Установим gh.

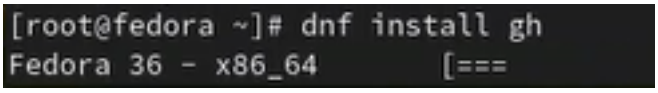

Рис. 3.8: Установка gh

3.4 Настройка gh

Авторизуемся и ответим на вопросы.

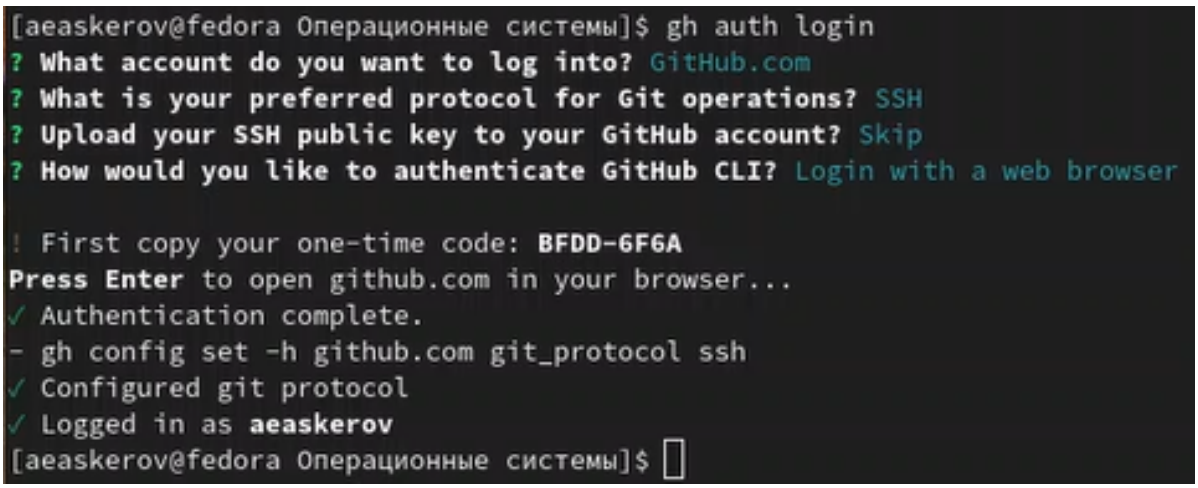

Рис. 3.9: Настройка gh

3.5 Настройка автоматических подписей коммитов git

Используя введёный email, укажем Git применять его при подписи коммитов.

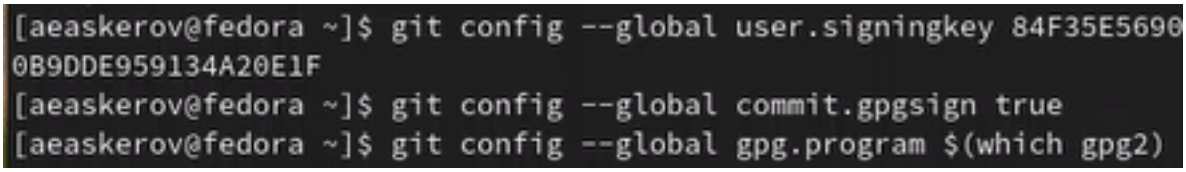

Рис. 3.10: Настройка автоматических подписей коммитов git

3.6 Создание pgp ключа

Сгенерируем pgp ключ.

```
[aeaskerov@fedora ~]$ gpg --full-generate-key
gpg (GnuPG) 2.3.4; 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.
```

Рис. 3.11: Генерация pgp ключа

3.7 Добавление pgp ключа в GitHub

Выведем список ключей и скопируем отпечаток приватного ключа.

```
[aeaskerov@fedora ~]$ gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
```

Рис. 3.12: Вывод списка ключей

Скопируем появившийся ключ и добавим его на GitHub.

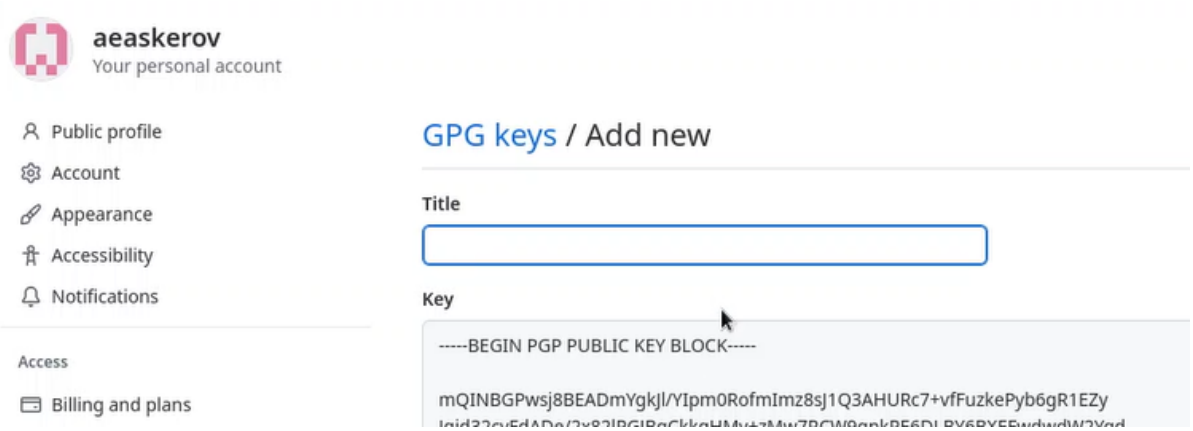

Рис. 3.13: Добавление pgp ключа в GitHub

3.8 Создание репозитория курса на основе шаблона

Создадим каталог для лабораторных работ.

```
[aeaskerov@fedora ~]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
[aeaskerov@fedora ~]$
```

Рис. 3.14: Создание каталога «Операционные системы»

Перейдём в созданный каталог.

```
[aeaskerov@fedora ~]$ cd ~/work/study/2022-2023/"Операционные системы"
```

Рис. 3.15: Переход в каталог «Операционные системы»

Введём команды для создания репозитория.

```
[aeaskerov@fedora Операционные системы]$ gh repo create study_2022-2023_os-intro
 --template=yamadharma/course-directory-student-template --public
✓ Created repository aeaskerov/study_2022-2023_os-intro on GitHub
[aeaskerov@fedora Операционные системы]$ git clone --recursive git@github.com:ae
askerov/study_2022-2023_os-intro.git os-intro
```

Рис. 3.16: Создание репозитория

3.9 Настройка каталога курса

Перейдём в каталог курса.

```
[aeaskerov@fedora Операционные системы]$ cd ~/work/study/2022-2023/"Операционные
 системы"/os-intro
```

Рис. 3.17: Переход в созданный каталог курса

Удалим лишние файлы.

```
[aeaskerov@fedora os-intro]$ rm package.json
```

Рис. 3.18: Удаление лишних файлов

Создадим необходимые каталоги.

```
[aeaskerov@fedora os-intro]$ echo os-intro > COURSE
[aeaskerov@fedora os-intro]$ make
```

Рис. 3.19: Создание необходимых каталогов

Отправим файлы на сервер.

```
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:aeaskerov/study_2022-2023_os-intro.git
```

Рис. 3.20: Отправка файлов на сервер

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

Рис. 3.21: Команды для отправки файлов на сервер

3.10 Контрольные вопросы

1) Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

Системы контроля версий (VCS) — это программные инструменты, которые помогают управлять изменениями исходного кода с течением времени. Они позволяют разработчикам отслеживать изменения, возвращаться к предыдущим версиям и сотрудничать с другими разработчиками. Они также позволяют хранить исходный код и делиться им с другими.

2) Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

Хранилище — это центральное место, где хранится и управляется исходный код. Это основная область хранения для всех версий проекта.

Commit — это сохраненная версия проекта. Он представляет собой снимок проекта в определенный момент времени. Коммиты используются для отслеживания изменений кода с течением времени.

История: история проекта — это запись всех коммитов, сделанных в репозиторий. Он показывает, как проект развивался с течением времени и какие изменения были внесены разработчиками.

Рабочая копия — это локальная копия репозитория. Она используется разработчиками для внесения изменений в код, не затрагивая основной репозиторий. Затем изменения могут быть зафиксированы в репозитории, когда они будут готовы.

3) Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

Централизованные системы контроля версий (CVCS) — это системы, в которых все файлы и изменения хранятся на одном сервере. Все пользователи получают доступ к одному и тому же серверу для фиксации, отправки и получения изменений. Примеры CVCS включают Subversion (SVN) и Perforce.

Децентрализованные системы контроля версий (DVCS) — это системы, в которых у каждого пользователя есть собственная локальная копия репозитория. Изменения можно зафиксировать в локальном репозитории, а затем отправить в репозитории других пользователей. Примеры DVCS включают Git и Mercurial.

4) Опишите действия с VCS при единоличной работе с хранилищем.

При единоличной работе с системой контроля версий пользователь может фиксировать изменения в своем локальном репозитории. Это включает в себя добавление новых файлов, редактирование существующих файлов и удаление файлов. Затем пользователь может отправить эти изменения в центральный

репозиторий, где они будут храниться для доступа других пользователей. Пользователь также может получать изменения из центрального репозитория, который будет обновлять их локальный репозиторий последними изменениями, сделанными другими пользователями.

5) Опишите порядок работы с общим хранилищем VCS.

1. Нужно создать локальный репозиторий. Первым шагом к работе с системой контроля версий является создание локального репозитория. Это можно сделать либо путем клонирования существующего репозитория с удаленного сервера, либо путем создания нового репозитория с нуля.
2. Нужно внести изменения. После создания локального репозитория пользователи могут вносить изменения в свои локальные файлы. Это может включать добавление новых файлов, редактирование существующих файлов и удаление файлов.
3. Нужно зафиксировать изменения: после внесения изменений в локальный репозиторий пользователи могут зафиксировать свои изменения. Это сохранит изменения в локальном репозитории и позволит их отслеживать.
4. Нужно отправить изменения: после внесения изменений в локальный репозиторий пользователи могут отправить эти изменения в центральный репозиторий. Это сохранит изменения, чтобы другие могли получить доступ и обновить свои локальные репозитории последними изменениями.
5. Нужно извлечь изменения: пользователи также могут извлекать изменения из центрального репозитория. Это обновит их локальный репозиторий последними изменениями, внесенными другими пользователями.

6) Каковы основные задачи, решаемые инструментальным средством git?

1. Контроль версий: Git позволяет пользователям отслеживать изменения в своем коде с течением времени и при необходимости возвращаться к предыдущим версиям.

2. Ветвление: Git позволяет пользователям создавать несколько веток проекта для разных целей, таких как разработка, тестирование и производство.
3. Слияние: Git позволяет пользователям объединять разные ветки вместе.
4. Совместная работа: Git упрощает совместную работу нескольких пользователей над проектом, позволяя им отправлять и извлекать изменения из репозиториев друг друга.
5. Безопасность: Git обеспечивает безопасный способ хранения исходного кода и управления им с помощью шифрования и аутентификации.

7) Назовите и дайте краткую характеристику командам git.

1. git init: инициализирует новый локальный репозиторий Git.
2. git clone: клонирует существующий репозиторий из удаленного источника.
3. git add: Добавляет файлы в промежуточную область для фиксации.
4. git commit: фиксирует изменения в локальном репозитории.
5. git push: отправляет изменения в удаленный репозиторий.
6. git pull: извлекает изменения из удаленного репозитория.
7. git status: Проверяет текущий статус репозитория.
8. git branch: создает ветки в репозитории и управляет ими.

8) Приведите примеры использования при работе с локальным и удалённым репозиториями.

Локальный репозиторий:

1. git init: инициализирует новый локальный репозиторий Git.
2. git add: Добавляет файлы в промежуточную область для фиксации.
3. git commit: фиксирует изменения в локальном репозитории.
4. git status: Проверяет текущий статус репозитория.
5. ветка git: создает ветки в репозитории и управляет ими.

Удаленный репозиторий:

1. git clone: клонирует существующий репозиторий из удаленного источника.
2. git push: отправляет изменения в удаленный репозиторий.
3. git pull: извлекает изменения из удаленного репозитория.

9) Что такое и зачем могут быть нужны ветви (branches)?

Ветки в системе git используются для разделения разных версий проекта. Это позволяет разработчикам работать над различными функциями или исправлениями ошибок, не затрагивая основной код. Ветки также можно использовать для экспериментов с новыми идеями или тестирования новых функций, не влияя на основной проект. Это упрощает отслеживание изменений и сотрудничество с другими разработчиками.

10) Как и зачем можно игнорировать некоторые файлы при commit?

Git позволяет пользователям игнорировать определенные файлы во время фиксации, добавляя их в файл .gitignore. Этот файл указывает, какие файлы следует игнорировать при фиксации изменений. Это может быть полезно для игнорирования файлов, которые не нужно отслеживать, таких как временные файлы или файлы конфигурации. Его также можно использовать для предотвращения случайной фиксации конфиденциальной информации, такой как пароли или ключи API.

4 Выводы

Изучены идеология и применение средств контроля версий и приобретены умения по работе с git.