Лабораторная работа №2
Первоначальна настройка git

Дисциплина Операционные системы

Осокин Георгий Иванович НММбд-02-22

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

2.1 Настройка GIT

Сконфигурируем `git` и создадим SSH ключ

```
[giosokin@localhost ~]$ git config --global user.name "giosokin"
[giosokin@localhost ~]$ git config --global user.email  "1132226517@pfur.ru"
[giosokin@localhost ~]$ git config --global core.quotepath false
[giosokin@localhost ~]$ git config --global core.autocrlf input
[giosokin@localhost ~]$ git config --global core.safecrlf warn
[giosokin@localhost ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/giosokin/.ssh/id_rsa): lab_key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in lab_key
Your public key has been saved in lab_key.pub
The key fingerprint is:
SHA256:We9KM4VhiH979taWCQXHLRboPLrIG7PkSiPGLumgfJQ giosokin@teapod
The key's randomart image is:
+---[RSA 4096]----+
|             .o..|
|       . .  ..oo.|
|      . . +o .o. |
|       . + ++  . |
|     .   S o.o..  |
|   E      ..+ .   |
|. ..+ o.+.=.+ ..o|
|o.o+ o +o=.* ..oo|
|.oo.. ..+..  ... |
+----[SHA256]-----+
[giosokin@localhost ~]$
```

Рис. 2.1: Конфигурация GIT

Создадим PGP ключ

Рис. 2.2: Создание PGP ключа

Настроим подпись коммитов нашим PGP ключом

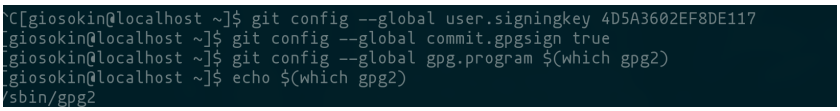

Рис. 2.3: Подпись коммитов PGP ключем

Добавим PGP ключ на гитхаб.

С помощью комманды `gpg --armor --export <PGP Fingerprint> | xclip -sel clip` скопируем PGP ключ И вставим в гитхаб

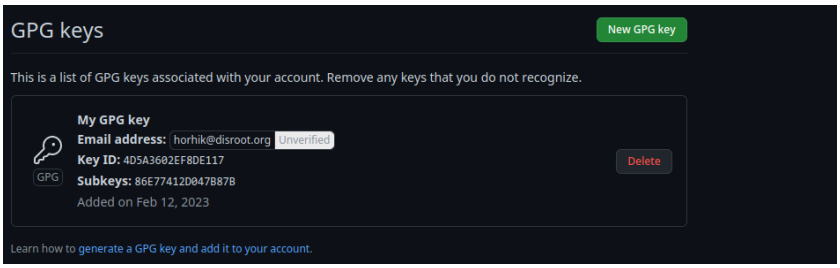

Рис. 2.4: Добавление ключа на гитхаб

Авторизируемся на гитхабе с помощью комманды `gh`

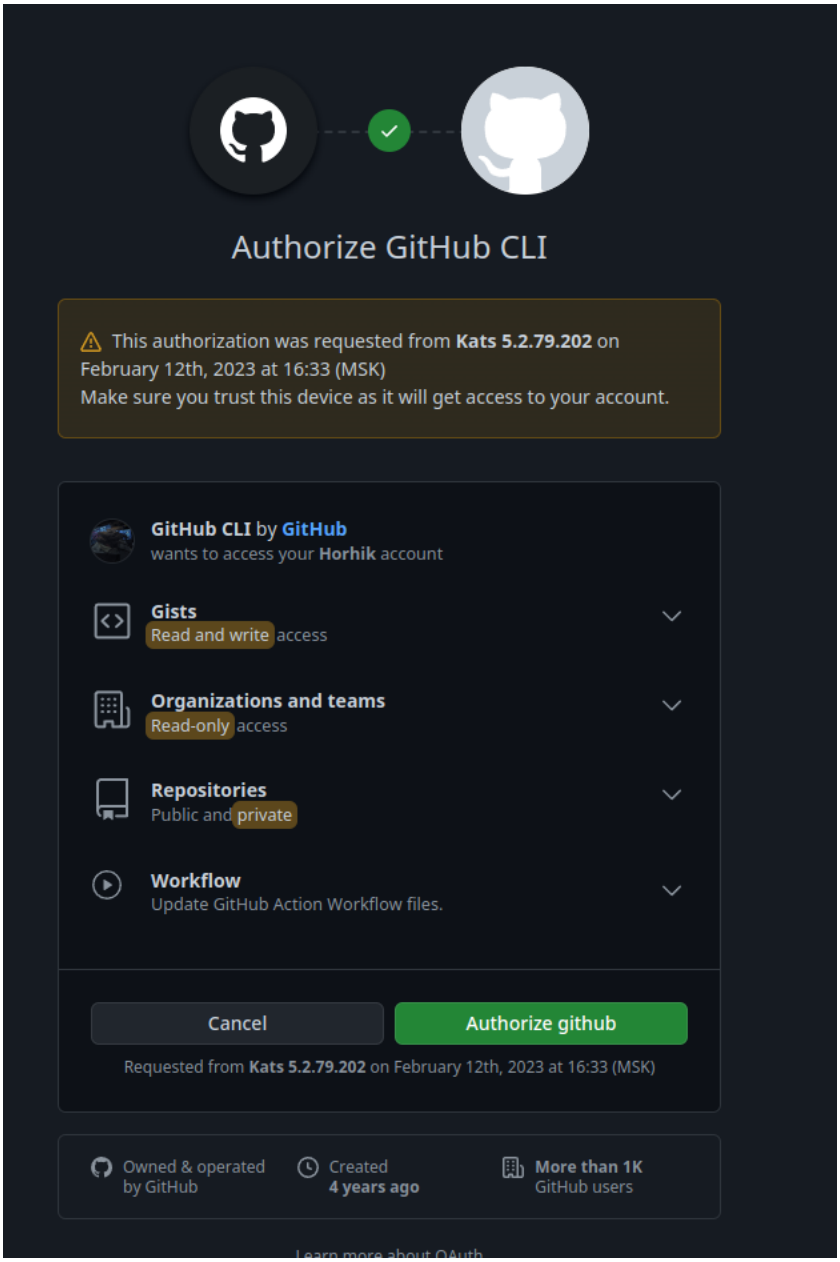

Рис. 2.5: Авторизация на гитхабе

2.2 Создание рабочего пространства

Перейдем в репозиторий с шаблоном и создадим из него шаблон

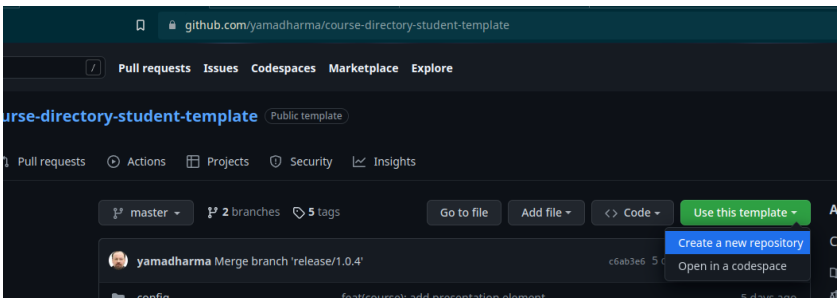

Рис. 2.6: Создание шаблона из шаблона

После создания шаблона, склонируем репозиторий на локальную машину

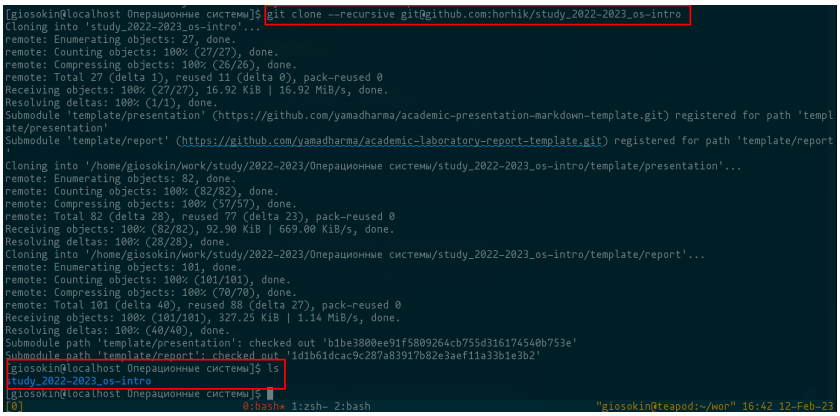

Рис. 2.7: Клонирование репозитория

Перейдем в репозиторий и удалим `package.json` Также создадим файл COURSE с текстом “os-intro”

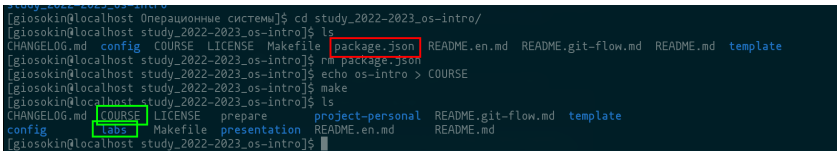

Рис. 2.8: Удаление фалйа package.json и создание файла COURSE

также запустим комманду `make`, которая сгенирирует файловую структуру. Убедимся коммандой `ls`

Теперь запушим на гитхаб изменения. До этого исполнив `git add` и `git commit` с названием коммита `feat(main): make course structure`

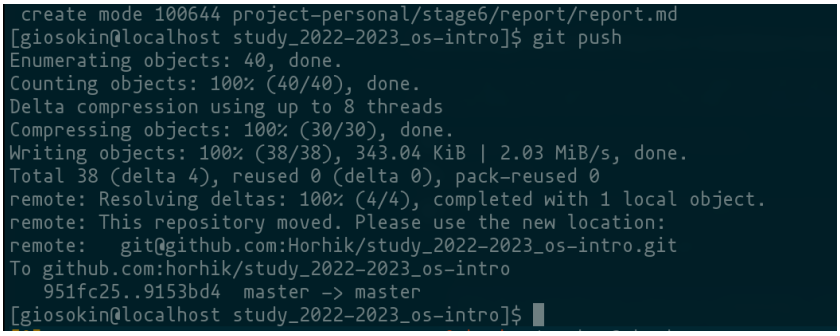

Рис. 2.9: Git push

Зайдем на гитхаб и убедимся что изменения запустились

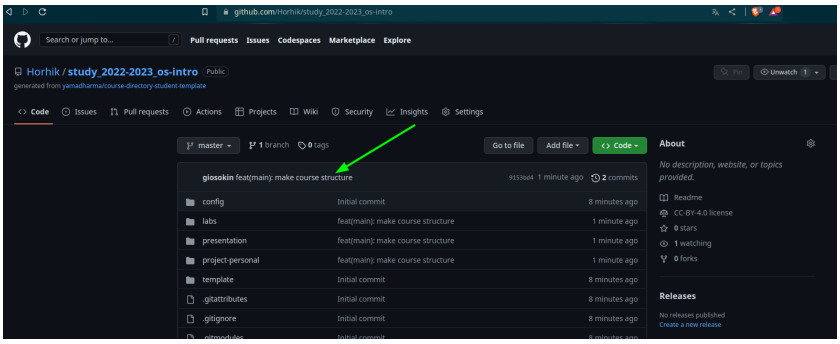

Рис. 2.10: Просмотр нового коммита на гихабе

3 Ответы на контрольные вопросы

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

Системы контроля версий применяются для фиксации изменений в программе и коллективной разработки

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

Хранилище: хранилище всех изменений и версий Commit: Единица изменения История: Последовательность коммитов Рабочая копия: Последняя версия

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

Децентрализованные VCS скачивают на хост всю цепочку изменений, а централизованные позволяют только подключаться к полноценному хранилищу и запрашивать конкретную версию.

3.0.0.4 Опишите действия с VCS при единоличной работе с хранилищем.

Пользователь может склонировать версию на свою машину, внести изменения и создать ветку

3.0.0.5 Опишите порядок работы с общим хранилищем VCS.

Пользователь запрашивает у хранилища определенную версию программы, VCS выдает ему эту версию, он вносит туда изменения и оправляет их обратно. VCS сохраняет изменения новой версии

3.0.0.6 Каковы основные задачи, решаемые инструментальным средством git?

1. Разработка больших проектов коммандами
2. Контроль изменений множества файлов и откат изменений
3. Возможность сохранять несколько состояний разработки программы

3.0.0.7 Назовите и дайте краткую характеристику командам git.

`git add` - добавить изменения в коммит
`git commit` - сохранить изменения присвоив им название
`git push` - отправить изменения на удаленный репозиторий
`git config` - изменение параметров git

3.0.0.8 Приведите примеры использования при работе с локальным и удалённым репозиториями.

В большом проекте разработчики будут клонировать локальный репозиторий себе на устройства, благодоря чему смогут работать параллельно.

Если же репозиторий слишком большой, разработчики могут подключаться к нему удаленно и править код в новых ветках и т.п.

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

Ветви могут быть нужни для совместной работы над проектом, для разработки нескольких функций программы одновременно, что бы не конфликтовать файлами при параллельной разработки программы несколькими людьми

3.0.0.10 Как и зачем можно игнорировать некоторые файлы при commit?

Файлы стоит вносить в .gitignore, это сделанно для того, что бы не загружать репозиторий лишними файлами или избегать утечки личной информации (баз данных и так далее)

4 Выводы

Мы изучили идеологию применения средств контроля версий и осовили базовые комманды утилиты git.