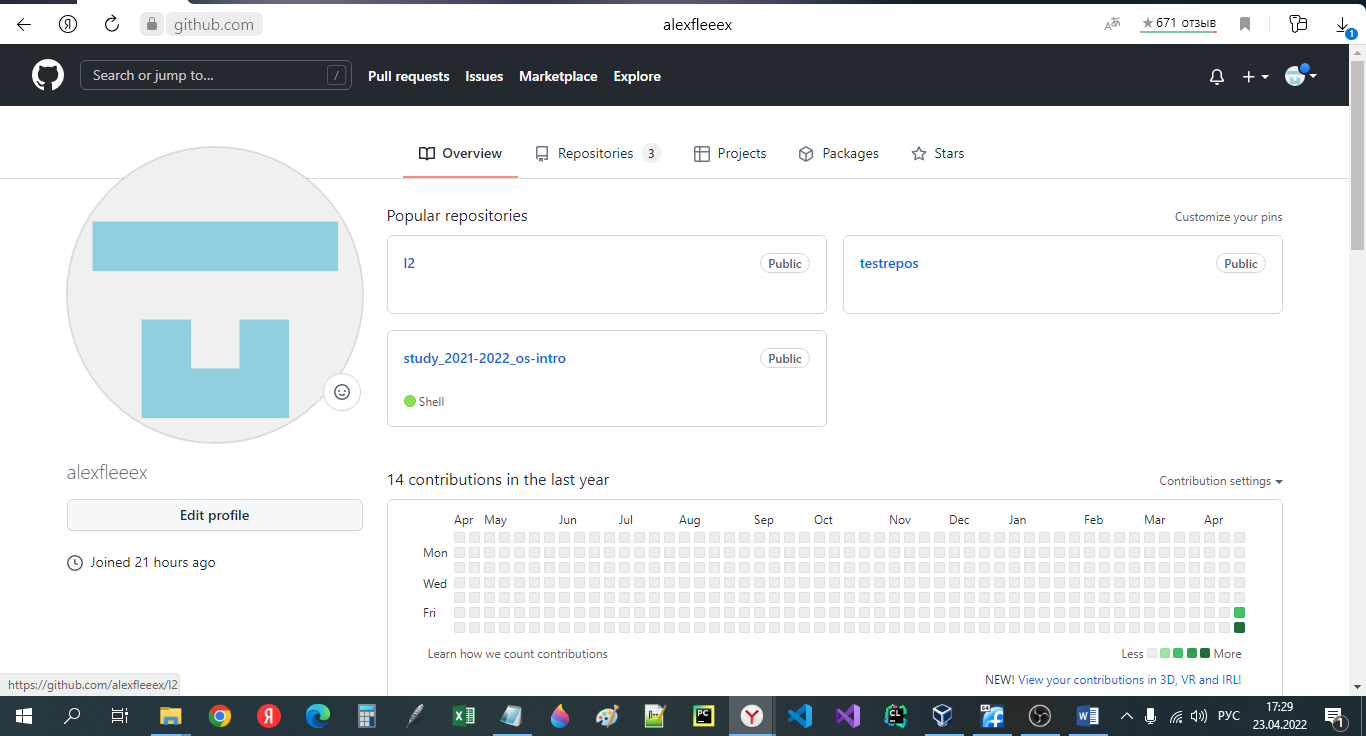
# **Отчет к лабораторной работе №3**

## **Common information**

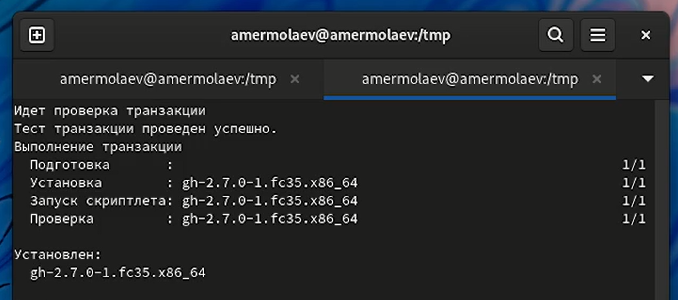
discipline: Операционные системы  
group: НПМбд-01-21  
author: Ермолаев А.М. — — ## **Цель работы** Изучить идеологию и применение средств контроля версий и освоить умения по работе с git. ## **Выполнение работы** Создадим аккаунт и заполним основные данные на сайте https://github.com.



screen

Установим git-flow в Fedora Linux при помощи терминала при помощи данных команд:

cd /tmp  
wget --no-check-certificate -q https://raw.github.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh  
chmod +x gitflow-installer.sh  
sudo ./gitflow-installer.sh install stable



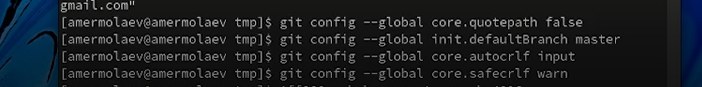
screen

Зададим имя и email владельца репозитория:

screen

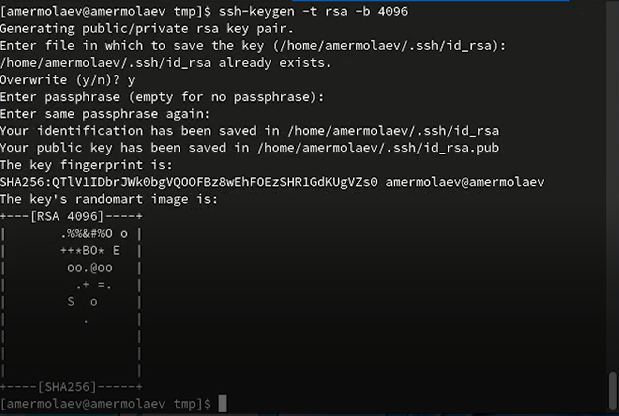
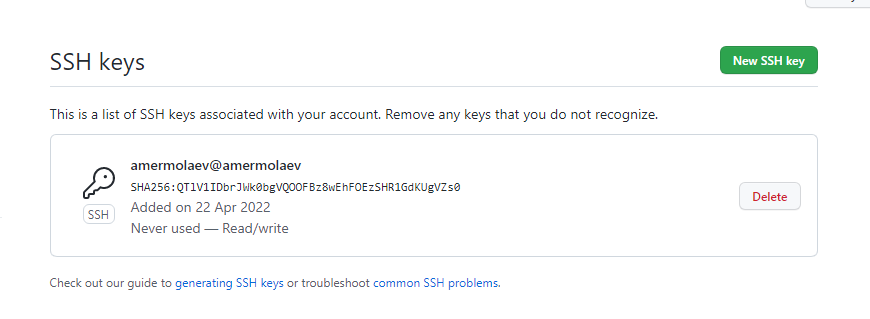
screen

Произведем другие базовые настройки.

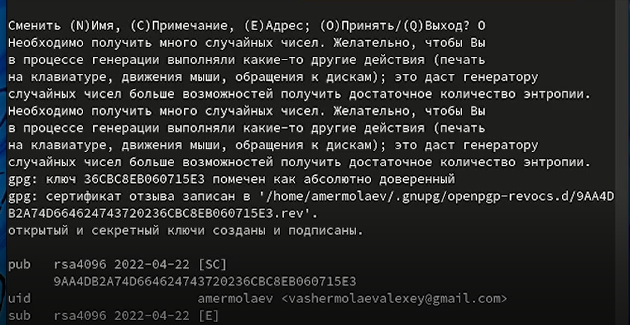


screen

Создадим ключ ssh. После этого скопируем ключ в буфер обмена и добавим его в поле «SSH Keys» в разделе Settings-SSH and GPG Keys.

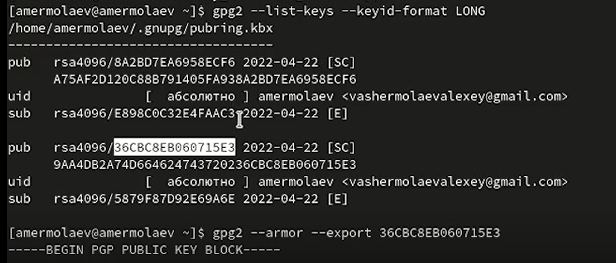
 

Создадим ключ gpg.



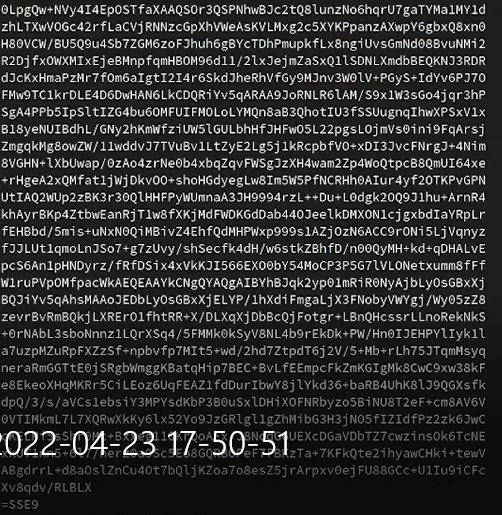
screen

После этого воспользуемся коммандой gpg2 --list-secret-keys --keyid-format=long, чтобы перечислить длинную форму ключей GPG, для которых у вас есть открытый и закрытый ключ.



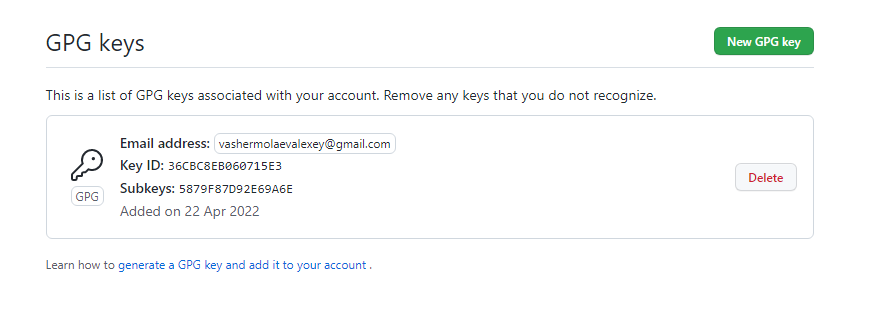
screen

Затем воспользуемся командой gpg --armor --export <идентификатор ключа>, чтобы вывести ключ в терминал.



screen

Скопируем свой ключ GPG, начиная с —–BEGIN PGP PUBLIC KEY BLOCK—–и заканчивая —–END PGP PUBLIC KEY BLOCK—–, и добавим его в поле «GPG Keys» в разделе Settings-SSH and GPG Keys.

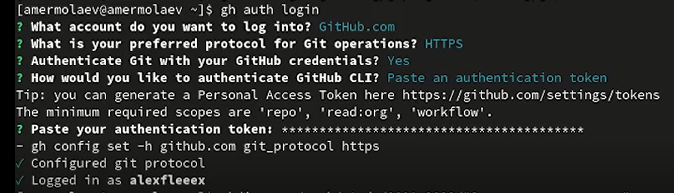


screen

Настроим автоматические подписи коммитов git.

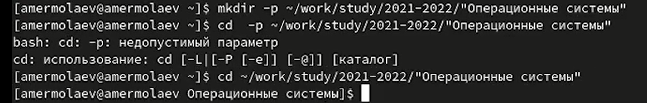
git config --global user.signingkey <PGP Fingerprint>  
git config --global commit.gpgsign true  
git config --global gpg.program $(which gpg2)

Авторизуемся и настроим gh, параллельно получив специальный токен.



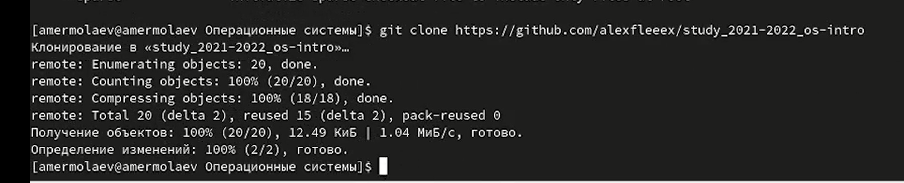
screen

Создадим шаблон рабочего пространства.

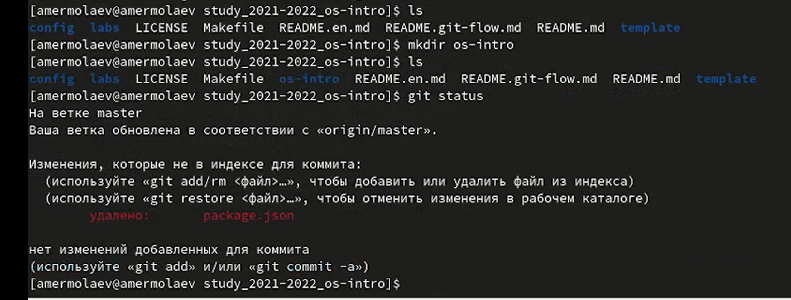


screen

После этого скопируем шаблон в собственный удаленный репозиторий и загрузим файлы в собственный каталог.

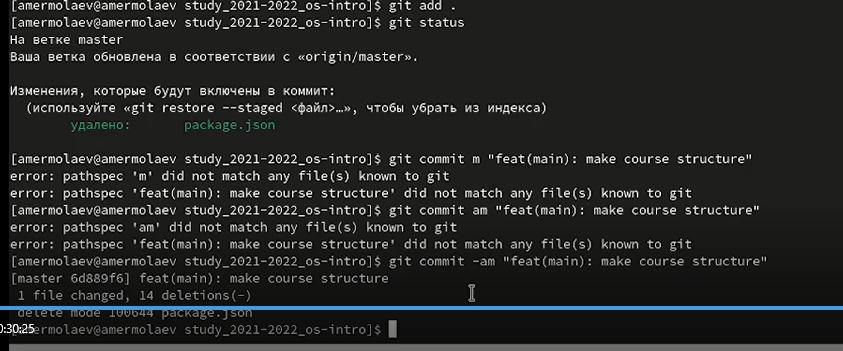
screen 

Удалим файлы и создадим новый каталог os-intro. Проверим статус контроля версий при помощи команды git status.



screen

Произведем отправку на сервер при помощи команд add, commit и push.



screen

## **Ответы на контрольные вопросы**

### *Вопрос 1*

Система контроля версий Git - набор программ командной строки, которые можно использовать посредством ввода команды git с различными опциями. Системы контроля версий (VCS) применяются при работе нескольких человек над одним проектом. ### *Вопрос 2* Хранилище – сервер, куда пользователь размещает новую версию проекта. При этом предыдущие версии не удаляются из центрального хранилища; к ним можно вернуться в любой момент. Сервер сохраняет только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Коммит – команда сохранения определенной версии в локальном репозитории в рамках системы управления версиями Git.

История версии содержит информацию об изменениях и служебную информацию.

Рабочая копия - созданная клиентской программой локальная копия части данных из хранилища. ### *Вопрос 3* Централизованные системы используют архитектуру клиент - сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Например, данная стратегия реализована с Subversion.

В децентрализованных системах каждый узел принимает свое собственное решение, и конечное поведение системы является совокупностью решений отдельных узлов. Bitcoin является отличным примером. ### *Вопрос 4* Локальный репозиторий создается в два этапа. Сначала устанавливается предварительная конфигурация при помощи команд git config –global user.name “Имя Фамилия” и git config –global user.email “work@mail”. Для инициализации локального репозитория в каталоге необходимо воспользоваться командой git init. ### *Вопрос 5* Для работы с общим хранилищем VCS необходимо сгенерировать ssh и pgp ключи, разместив их в соответствующих полях системы. Если говорить о Git, то для отправки изменений на сервер надо будет воспользоваться командой git push. ### *Вопрос 6* Git решает две основные задачи: хранение информации о всех изменениях в вашем коде, начиная с самой первой строчки, и обеспечение удобства командной работы над кодом. ### *Вопрос 7* Наиболее часто используемые команды git:  
- git init – создание основного дерева репозитория - git pull – получение обновлений (изменений) текущего дерева из центрального репозитория - git push – отправка всех произведённых изменений локального дерева в центральный репозиторий - git status – просмотр списка изменённых файлов в текущей директории: - git diff – просмотр текущих изменения - git add – добавление файла в раздел сохранения изменений - git commit - сохранение добавленных изменений - git checkout/ branch – работа с ветками ### *Вопрос 8* Локальный репозиторий:

git config --global user.name "Имя Фамилия"  
git config --global user.email "work@mail"  
cd  
mkdir tutorial  
cd tutorial  
git init  
echo 'hello world' > hello.txt  
git add hello.txt  
git commit -am 'Новый файл'

Удаленный репозиторий:

mkdir -p ~/work/study/2021-2022/"Операционные системы"  
cd ~/work/study/2021-2022/"Операционные системы"  
gh repo create study\_2021-2022\_os-intro --template=yamadharma/course-directory-student-template --public  
git clone --recursive git@github.com:<owner>/study\_2021-2022\_os-intro.git os-intro  
cd ~/work/study/2021-2022/"Операционные системы"/os-intro  
rm package.json  
make COURSE=os-intro  
git add .  
git commit -am 'feat(main): make course structure'  
git push

### *Вопрос 9*

Ветки в Git 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. ### *Вопрос 10* Во время работы над проектом так или иначе могут создаваться файлы (например, объектные файлы или файлы библиотек), которые не требуется добавлять в последствии в репозиторий. Для таких файлов можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore.

## **Вывод**

В рамках выполнения работы я изучил идеологию и применение средств контроля версий и освоил умения по работе с git.