Отчёт по лабораторной работе №2

Первоначальна настройка git

Сагдеров Камал

Содержание

# 1 Цель работы

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

# 2 Задание

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Настроить подписи git.
5. Зарегистрироваться на Github.
6. Создать локальный каталог для выполнения заданий по предмету.

# 3 Теоретическое введение

В результате выполнения лабораторной работы №2, мы познакомились с системой контроля версий.Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется. Система контроля версий имеют возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

# 4 Выполнение лабораторной работы

№1 Установка программного обеспечения Создаем учетную запись Github и заполняем основные данные. После устанавливаем программное обеспечение, устанавливаем gh. Затем переходим к базовой настройки git, где задаю свои данные (имя и email) репозитория. Настраиваю utf-8 в выводе сообщений git. Задаем имя начальной ветки, параметры (autocrlf, safecrlf) (рис. [1](#fig:001)).

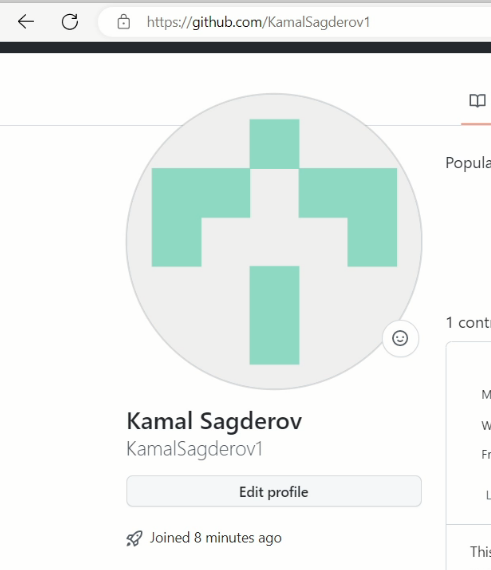


Figure 1: Создание учетной записи

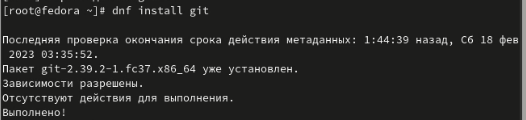


Figure 2: установка git

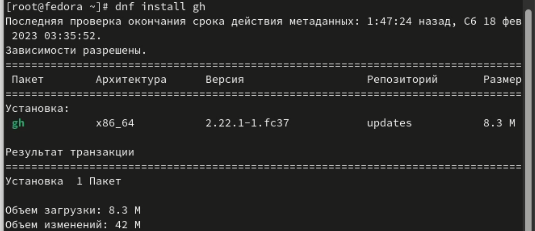


Figure 3: установка gh

№2 Базовая настройка git Задаем данные владельца, настраиваем utf-8 в выводе сообщений git, верификацию и подписание коммитов git и зададем имя начальной ветки. (рис. [4](#fig:004)).

Figure 4: Задаем имя и email владельца репозитория, настраиваю utf-8

Figure 4: Задаем имя и email владельца репозитория, настраиваю utf-8

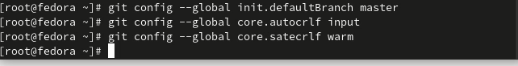


Figure 5: Настройка верификацию и подписание коммитов git

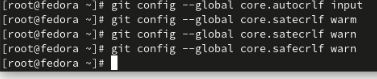


Figure 6: параметры autocrlf и safecrlf

№3 Создание ключей ssh и pgp

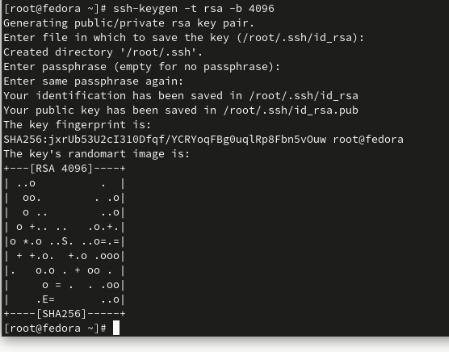


Figure 7: по алгоритму rsa

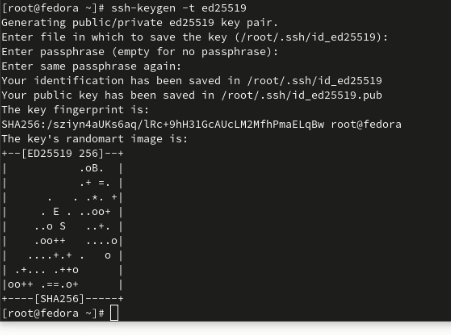


Figure 8: по алгоритму ed25519

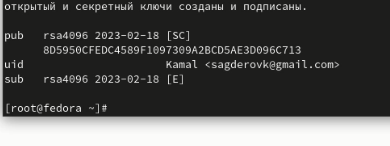


Figure 9: Генерация ключей

№4 Экспорт ключа Выводим список ключей и копируем отпечаток приватного ключа, а также экспортируем ключ в формате ASCII по его отпечатку (рис. [10](#fig:010)).

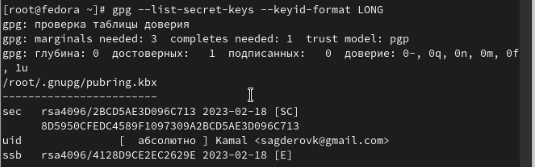


Figure 10: Вывод списка ключей и копирование отпечатка приватного ключа

№5 Добавление PGP ключа в GitHub

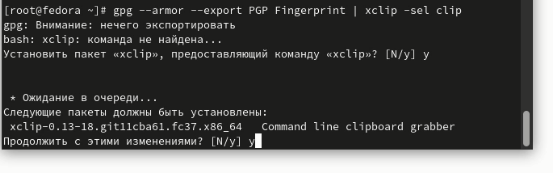


Figure 11: Добавление PGP ключа в GitHub

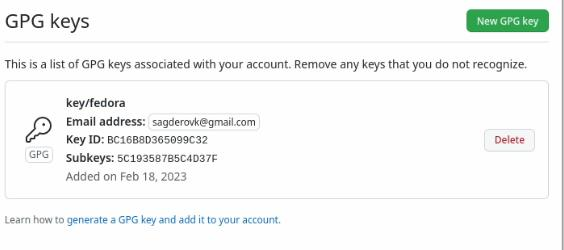


Figure 12: Создание GPG ключа

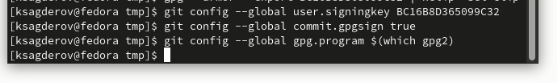


Figure 13: Настройка автоматических подписей коммитов git

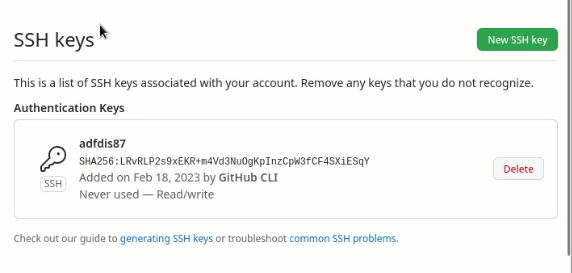


Figure 14: Настройка gh

№6 Создание репозитория курса на основе шаблона

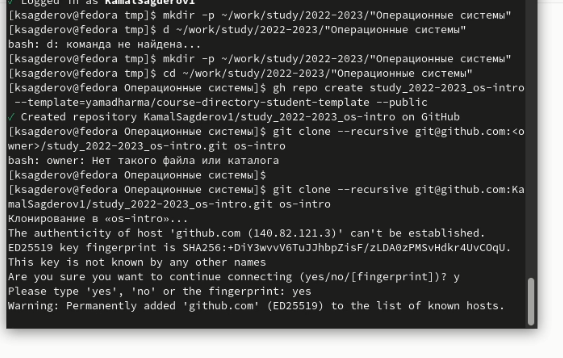


Figure 15: создание репозитория курса

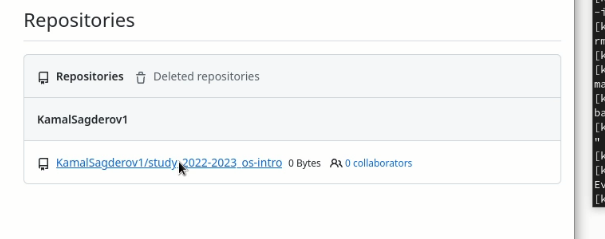


Figure 16: создание репозитория курса

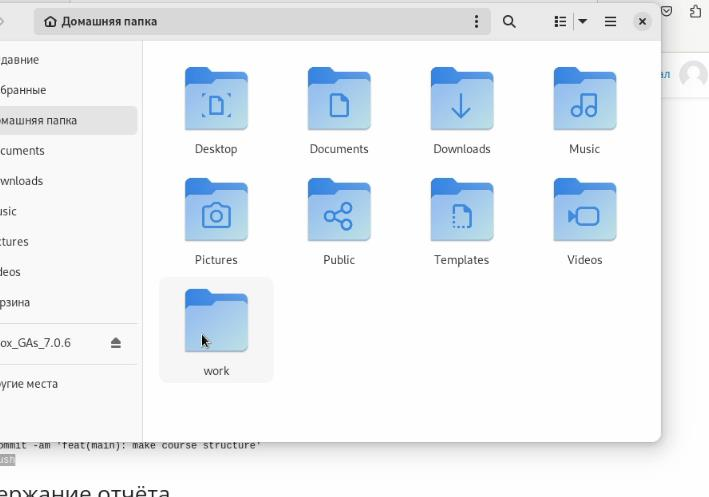


Figure 17: Создание каталога курса

№7 Контрольные вопросы 1.Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Система управления версиями — 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.

Система контроля версий применяется при работе : - нескольких человек над одним проектом. - полной истории изменений - причин всех производимых изменений - Откат изменений, если что-то пошло не так - Поиск причины и ответственного за появления ошибок в программе - Совместная работа группы над одним проектом

2.Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. repository - хранилище файлов, ссылок на изменения в файлах, в нем хранятся все документы вместе с историей их изменения и другой служебной информацией. commit - отслеживание изменений, сохраняет разницу в изменениях История - хранит все изменения в проекте и позволяет при необходимости обратиться к нужным данным. Рабочая копия- «снимок» содержимого репозитория, плюс некоторая служебная информация. Изменив содержимое рабочей копии, разработчик фиксирует сделанные изменения в репозитории. Как правило, фиксация сопровождается небольшим текстовым комментарием, описывающим сделанные изменения.

3.Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные VCS - это системы контроля версий, которые имеют один основной удаленный репозиторий, к которому все пользователи имеют доступ. Они базируются на предположении, что репозиторий является истинным и достоверным. Примерами таких VCS являются Subversion и CVS.

Децентрализованные VCS - это системы контроля версий, которые не имеют основного удаленного репозитория. Вместо этого каждый пользователь имеет локальную копию репозитория, которая может быть синхронизирована с копиями 1. Централизованные:

* Subversion (SVN)
* CVS
* Perforce

1. Децентрализованные:

* Git
* Mercurial
* Bazaar

1. Опишите действия с VCS при единоличной работе с хранилищем.

* Создание локального хранилища: Используя команду git init, можно создать локальное хранилище.
* Клонирование хранилища: Если хранилище уже существует в удаленном репозитории, можно использовать команду git clone для клонирования хранилища на локальную машину.
* Добавление файлов: Используя команду git add, можно добавлять файлы в хранилище.
* Коммитирование изменений: Используя команду git commit, можно коммитить изменения в хранилище.
* Просмотр истории коммитов: Используя команду git log

1. Опишите порядок работы с общим хранилищем VCS.
2. Начните с создания репозитория VCS. Это может быть на сервере или на локальном компьютере.
3. Добавьте в репозиторий все необходимые файлы.
4. Создайте коммит для хранения изменений.
5. Обменяйтесь изменениями с другими участниками проекта с помощью пулл-реквестов.
6. Объедините изменения в одну версию.
7. Обновите код на локальной машине.
8. Проверьте изменения и подтвердите их.
9. Отправьте изменения в общее хранилище.

6.Каковы основные задачи, решаемые инструментальным средством git? 1 - хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки 2 - обеспечение удобства командной работы над кодом

1. Назовите и дайте краткую характеристику командам git.
2. git init - Инициализация нового репозитория Git в текущем каталоге.
3. git clone - Клонирование удаленного репозитория в текущий каталог.
4. git add - Добавление файлов в репозиторий для последующего коммита.
5. git commit - Сохранение изменений в репозитории.
6. git push - Отправка изменений из локального репозитория в удаленный репозиторий.
7. git pull - Загрузка изменений из удаленного репозитория в локальный репозиторий.
8. git status - Просмотр текущего состояния репозитория.
9. git diff - Просмотр изменений, внесенных
10. Приведите примеры использования при работе с локальным и удалённым репозиториями. git push - all(push origin master/любой branch) $ git fetch - загрузка изменений с удаленного репозитория $ git push - отправка изменений в удаленный репозиторий
11. что такое и зачем могут быть нужны ветви (branches)? Ветви (branches) — это варианты исходного кода, которые могут быть использованы для разработки различных версий программы. Они позволяют разработчикам и тестировщикам работать над одним проектом одновременно, не пересекаясь друг с другом. Кроме того, они позволяют отслеживать изменения в исходном коде и настраивать процесс разработки.
12. Как и зачем можно игнорировать некоторые файлы при commit? Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов.

# 5 Выводы

В результате выполнения лабораторной работы я научился создавать учетную запись GitHub, устанавливать и работать с программным обеспечением. Также научился базовой настройки git и работать с серверами репозиториями, добавлять SSH/PGP ключи в свою учетную запись Github