Лабораторная работа №2

НКАбд-06-23

Улитина Мария Максимовна

Содержание

# 1 Цель работы

Изучить идеологию и применение средств контроля версий. 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

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.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

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

## 4.1 Установка git и gh

Установим git (рис. 1).

git

Рис. 1: git

Установим gh (рис. 2).

gh

Рис. 2: gh

## 4.2 Базовая настройка git

Зададим имя и email владельца репозитория (рис. 3).

имя и email

Рис. 3: имя и email

Настроим utf-8 в выводе сообщений git (рис. 4).

utf-8

Рис. 4: utf-8

Зададим имя начальной ветки (рис. 5).

имя начальной ветки

Рис. 5: имя начальной ветки

Настроим параметы autocrlf и safecrlf (рис. 6).

autocrlf и safecrlf

Рис. 6: autocrlf и safecrlf

## 4.3 Создание ключа ssh

По алгоритму rsa создадим ключ размером 4096 бит (рис. 7).

rsa

Рис. 7: rsa

И по алгоритму ed25519 (рис. 8).

ed25519

Рис. 8: ed25519

## 4.4 Ключи pgp

Создадим ключ pgp (рис. 9).

ключ pgp

Рис. 9: ключ pgp

Из предложенных опций выбираем тип ключа и его размер(рис. 10).

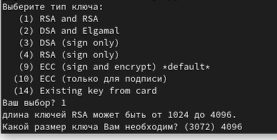


Рис. 10: тип ключа

## 4.5 Добавление PGP ключа в GitHub

Выводим список ключей и копируем отпечаток приватного ключа: (рис. 11).

Список ключей

Рис. 11: Список ключей

Скопируем полученный ключ и добавим его на github.

## 4.6 Настройка автоматических подписей коммитов git

Используя введёный email, укажем Git применять его при подписи коммитов: (рис. 12).

Подпись коммитов

Рис. 12: Подпись коммитов

## 4.7 Настройка gh

Авторизируемся в gh (рис. 13).

Авторизируемся в gh

Рис. 13: Авторизируемся в gh

## 4.8 Создание репозитория курса на основе шаблона

Создадим шаблон для рабочего пространства (рис. 14).

Шаблон для рабочего пространства

Рис. 14: Шаблон для рабочего пространства

(рис. 15).

Шаблон для рабочего пространства

Рис. 15: Шаблон для рабочего пространства

(рис. 16).

Шаблон для рабочего пространства

Рис. 16: Шаблон для рабочего пространства

## 4.9 Настройка каталога курса

Перейдем в каталог курса (рис. 17).

Каталог курса

Рис. 17: Каталог курса

Удалим лишние файлы (рис. 18).

Удаление

Рис. 18: Удаление

Создадим необходимые каталоги (рис. 19).

Создание каталога

Рис. 19: Создание каталога

(рис. 20).

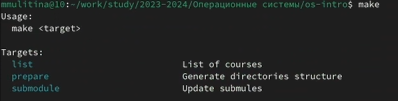


Рис. 20: Создание каталога

Отправим файлы на сервер (рис. 21).

Отправление файлов на сервер

Рис. 21: Отправление файлов на сервер

(рис. 22).

Отправление файлов на сервер

Рис. 22: Отправление файлов на сервер

# 5 Контрольные вопросы

1. 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.
2. Хранилище - единый репозиторий для хранения файлов. Commit - сохранить все добавленные изменения и все изменённые файлы. История - история изменений репозитория. Рабочая копия - копия проекта, основанная на версии из хранилища.
3. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным. Классические - CVS, Subversion. Распределенные - Git, Bazaar, Mercurial.
4. Создание и подключение удаленного репозитория, отправка изменений проекта на сервер.
5. Системы контроля версий могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.
6. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями.
7. Создание основного дерева репозитория git init, получение обновлений текущего дерева из центрального репозитория git pull, отправка всех произведённых изменений локального дерева в центральный репозиторий git push, просмотр списка измененных файлов в текущей директории git status, просмотр текущих изменений, сохранение текущих и добавленных изменений git diff.
8. git push - отправление из локального репозитория на удаленный.
9. Для разработки отдельных частей проекта.
10. Для игнорирования файлов, которые не стоит добавлять в репозиторий. С помощью конфигурационного файла gitignore.

# 6 Выводы

В процессе выполнения работы я изучила применение средств контроля версий Git.

# Список литературы

1. Лабораторная работа №2.