Лабораторная работа №2

Управление версиями

Тараканов Борис Александрович студ 1032212284

Содержание

# Цель работы

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

# Задание

– Создать базовую конфигурацию для работы с git. – Создать ключ SSH. – Создать ключ PGP. – Настроить подписи git. – Зарегистрироваться на Github. – Создать локальный каталогдля выполнения заданий по предмету.

# Теоретическое введение

Системы контроля версий (Version Control System,VCS) применяются при работе несколь- кихчеловекнадоднимпроектом.Обычноосновноедеревопроектахранитсявлокальном 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 этотребуется. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов.Выполнение боль- шинства 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 сохранятьнеполнуюверсиюизменённыхфайлов,апроизводитьтакназываемуюдельта- компрессию—сохранятьтолько изменения между последовательными версиями,что позволяет уменьшить объём хранимых данных. Системы контроля версий поддерживаютвозможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом.Можно объединить (слить) изменения,сделанные разными участниками (авто- матически или вручную),вручную выбрать нужную версию,отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяетдругим пользователям получитьрабочую копию или препятствуетизменению рабочей копии файла средствами файловой системы ОС,обеспечиваятаким образом, привилегированный доступтолько одному пользователю,работающему с файлом. Системы контроля версийтакже могутобеспечиватьдополнительные,более гибкие функциональные возможности.Например,они могут поддерживать работу с нескольки- ми версиями одного файла,сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви.Крометого,обычно доступна информация отом,кто из участников,когда и какие изменения вносил.Обычнотакого 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 распределён- ных — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

# Выполнение лабораторной работы

#*1)Базовая настройка GIT* Произведем базовую настройку Github. Для этого создадим профиль на сайте GitHub (У меня профиль уже был, поэтому нужна была только настройка) (рис. 1)

Рис. 1

Рис. 1

#Базовая настройка (рис. 2): 1)Зададим имя пользователя и электронный адрес 2)настроим utf-8 в выводе сообщений git 3)Настроим верефикацию и подписание коммитов git 4)Зададим имя начальной ветки (master) 5)Изменим параметры autocrlf и safecrlf

Рис. 2

Рис. 2

#Создание ключей: Создаём ключ SSH. В терминале вводим данную команду: **ssh-keygen -t rsa -b 4096** Далее во всех пунктах пользуемся клавишей Enter и получаем наш ключ. (Рис. 3)

Рис. 3

Рис. 3

#Подключение ключей к Github: Скопированный ключ нужно добавить на github. Для этого переходим на сайте в раздел “Settings” и выбираем “SSH and GPG keys”. Как только мы выбрали SHH keys, даём название нашему ключу и копируем ключ из терминала. В моём случае был создан ключ под названием OS (операционные системы). Возвращаемся в наш терминал и настраиваем gh командой: **gh auth login** Во всех пунктах выбираем y(yes). По полученной ссылке переходим в браузер на виртуальной машине и вводим код из терминала (находится перед ссылкой).(Рис. 4)

Рис. 4

Рис. 4

#Создание репозитория на Github: Создаём репозиторий курса на основе шаблона. Все нужные команды для создания были в указаниях к лабораторной работе. В 4 команде, вместо , указываем своё имя профиля на github.(Рис. 5)

Рис. 5

Рис. 5

#Настройка каталога: Настраиваем каталог курса. Для этого переходим в него командой: Далее командой ls проверяем, что мы в него перешли. В каталоге “os-intro” нам потребуется удалить файл “package.json”. Снова командой ls проверяем успешное выполнение удаление файла. (Рис. 6)

Рис. 6

Рис. 6

#Синхронизация с сервером: Создаём необходимые каталоги и отправляем наши файлы на сервер. (Рис. 7) **make COURSE=os-intro** **git add .** **git commit -am ‘feat(main): make course structure’** **git push**

Рис. 7

Рис. 7

# Выводы

В ходе выполнения лабораторной работы изучили идеологию и применение средств контроля версий, а также освоили умения по работе с git.