Лабораторная работа 2

Операционные системы

Мухина Даниила Александровича

Содержание

# 1 Цель работы

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

# 2 Задание

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

# 3 Выполнение лабораторной работы

Установка git и gh командами dnf install git и dnf install gh

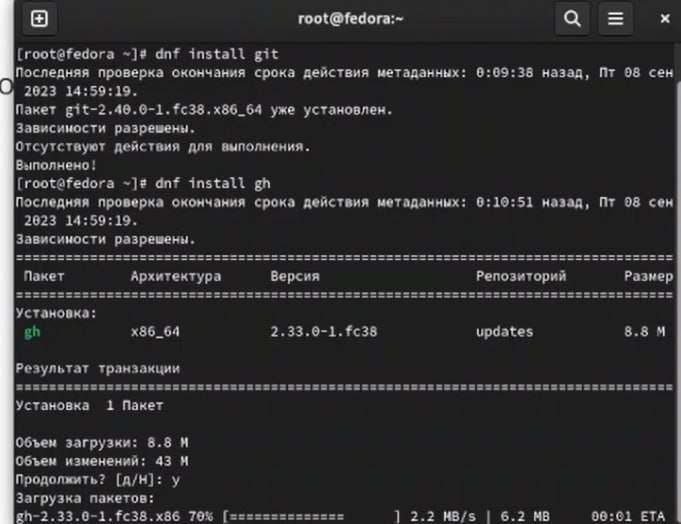


Figure 1: Установка программного обеспечения

Базовая настройка git командами

git config –global user.name “Name Surname” git config –global user.email “work@mail” git config –global core.quotepath false git config –global init.defaultBranch master git config –global core.autocrlf input git config –global core.safecrlf warn [Базовая настройка git] (image/2.jpg){#fig:002 width=70%}

Создание ssh ключа командами по алгоритму rsa с ключём размером 4096 бит: ssh-keygen -t rsa -b 4096 по алгоритму ed25519: ssh-keygen -t ed25519

[Создание ssh ключа] (image/3.jpg){#fig:003 width=70%}

Создание ключа pgp командами gpg –full-generate-key

[Создание pgp ключа] (image/4.jpg){#fig:004 width=70%}

Добавление pgp ключа в GitHub командами gpg –list-secret-keys –keyid-format LONG gpg –armor –export | xclip -sel clip

[Добавление Pgp ключа в GitHUb] (image/5.jpg){#fig:005 width=70%}

Настройка автоматических подписей коммитов git командами git config –global user.signingkey git config –global commit.gpgsign true git config –global gpg.program $(which gpg2)

[Настройка автоматических коммитов git] (image/6.jpg){#fig:006 width=70%}

Создание репозитория курса на основе шаблона командами

mkdir -p ~/work/study/2022-2023/“Операционные системы” cd ~/work/study/2022-2023/“Операционные системы” gh repo create study\_2022-2023\_os-intro –template=yamadharma/course-directory-student-template –public git clone –recursive git@github.com:/study\_2022-2023\_os-intro.git os-intro

[Создание репозитория курса на основе шаблона командами ] (image/7.jpg){#fig:007 width=70%}

Настройка каталога курса командами cd ~/work/study/2022-2023/“Операционные системы”/os-intro rm package.json echo os-intro > COURSE make git add . git commit -am ‘feat(main): make course structure’ git push

[Настройка каталога курса] (image/8.jpg){#fig:008 width=70%} # Ответы на контрольные вопросы 1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Это программное обеспечение для облегчения работы с изменяющейся информацией. VCS 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. 2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище (repository), или репозитарий, — место хранения всех версий и служебной информации. Commit («[трудовой] вклад», не переводится) — синоним версии; процесс создания новой версии. История – место, где сохраняются все коммиты, по которым можно посмотреть данные о коммитах. Рабочая копия – текущее состояние файлов проекта, основанное на версии, загруженной из хранилища. 3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные VCS: одно основное хранилище всего проекта и каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно. Децентрализованные VCS: у каждого пользователя свой вариант (возможно не один) репозитория. 4. Опишите действия с VCS при единоличной работе с хранилищем. 5. Опишите порядок работы с общим хранилищем VCS. 6. Каковы основные задачи, решаемые инструментальным средством git? Git — это система управления версиями. У Git две основных задачи: первая —хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом. 7. Назовите и дайте краткую характеристику командам git. git –version (Проверка версии Git) git init (Инициализировать ваш текущий рабочий каталог как Git-репозиторий) git clone https://www.github.com/username/repo-name (Скопировать существующий удаленный Git-репозиторий) git remote (Просмотреть список текущих удалённых репозиториев Git) git remote -v (Для более подробного вывода) git add my\_script.py (Можете указать в команде конкретный файл). git add . (Позволяет охватить все файлы в текущем каталоге, включая файлы, чье имя начинается с точки) git commit -am “Commit message” (Вы можете сжать все индексированные файлы и отправить коммит). git branch (Просмотреть список текущих веток можно с помощью команды branch) git –help (Чтобы узнать больше обо всех доступных параметрах и командах) git push origin master (Передать локальные коммиты в ветку удаленного репозитория). 8. Приведите примеры использования при работе с локальным и удалённым репозиториями. 9. Что такое и зачем могут быть нужны ветви (branches)? Ветки 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. 10. Как и зачем можно игнорировать некоторые файлы при commit? Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты.

# 4 Выводы

В ходе выполнения лабораторной работы изучили идеологию и применение средств контроля версий, а также освоили умения по работе с git. # Список литературы{.unnumbered}