РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: операционные системы

Студент: Тозе Виктор Ф

Группа:НФИбд-02-21

МОСКВА

2022 г.

Цель работы

– 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.

Ход работы

1. **Настройка github**

Создали учётную запись на https://github.com и Заполнили основные данные на https://github.com.

1. **Установка программного обеспечения**

** Установка git-flow и gh в Fedora Linux

1. **Базовая настройка git**

Зададили имя и email владельца репозитория:

Настроили utf-8 в выводе сообщений git:

Настроили верификацию и подписание коммитов git. Задавили имя начальной ветки (будем называть её master):

– Параметр autocrlf и Параметр safecrlf:

1. Создание ключи ssh и gpg

Ключ ssh

Ключ gpg

1. **Добавление PGP ключа в GitHub**

Перейдили в настройки GitHub (https://github.com/settings/keys), нажали на

кнопку New GPG key и вставьте полученный ключ в поле ввода

1. **Настройка автоматических подписей коммитов gi**

Используя введёный email, указывали Git применять его при подписи коммитов

1. **Настройка gh**

1. **Шаблон для рабочего пространства**

Сознание репозитория курса на основе шаблона

Необходимо создать шаблон рабочего пространства.

Например, для 2021–2022 учебного года и предмета «Операционные системы» (код

предмета os-intro) создание репозитория примет следующий вид:

Перейдили в каталог курса и удаляем файл rm package.json

–Создали необходимые каталоги и Отправили файлы на сервер

Контрольные вопросы:

\1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

Система контроля версий (VCS) — это место хранения кода. 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.

Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы.

\2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией

Commit («[трудовой] вклад», не переводится) — процесс создания новой версии

Рабочая копия (working copy) — текущее состояние файлов проекта, основанное на версии, загруженной из хранилища (обычно на последней).

Версия (revision), или ревизия, — состояние всех файлов на определенный момент времени, сохраненное в репозитарии, с дополнительной информацией

\3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. (Пример — Wikipedia.)

В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. (Пример — Bitcoin)

\6. Каковы основные задачи, решаемые инструментальным средством git?

У Git есть две основные задачи: хранить информацию обо всех изменениях в коде, начиная с самой первой строчки, и обеспечить удобства командной работы над кодом.

\7. Назовите и дайте краткую характеристику командам git.

– создание основного дерева репозитория: git init – получение обновлений (изменений) текущего дерева из центрального репозитория: git pull – отправка всех произведённых изменений локального дерева в центральный репозиторий: git push – просмотр списка изменённых файлов в текущей директории: git status – просмотр текущих изменения: git diff – сохранение текущих изменений: – добавить все изменённые и/или созданные файлы и/или каталоги: git add . – добавить конкретные изменённые и/или созданные файлы и/или каталоги: git add имена_файлов – удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): git rm имена_файлов – сохранение добавленных изменений: – сохранить все добавленные изменения и все изменённые файлы: git commit -am 'Описание коммита' – сохранить добавленные изменения с внесением комментария через встроенный редактор: git commit – создание новой ветки, базирующейся на текущей: git checkout -b имя_ветки – переключение на некоторую ветку: git checkout имя_ветки (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) – отправка изменений конкретной ветки в центральный репозиторий: git push origin имя_ветки – сл ияние ветки с текущим деревом: git merge —no-ff имя_ветки – удаление ветки: – удаление локальной уже слитой с основным деревом ветки: git branch -d имя_ветки – принудительное удаление локальной ветки: git branch -D имя_ветки – удаление ветки с центрального репозитория: git push origin :имя_ветки

\9. Что такое и зачем могут быть нужны ветви (branches)?

‘Git branch’ – это команда для управления ветками в репозитории Git.

Ветка – это просто «скользящий» указатель на один из коммитов. Когда мы создаём новые коммиты, указатель ветки автоматически сдвигается вперёд, к вновь созданному коммиту.

Ветки используются для разработки одной части функционала изолированно от других. Каждая ветка представляет собой отдельную копию кода проекта. Ветки позволяют одновременно работать над разными версиями проекта.

Ветвление («ветка», branch) — один из параллельных участков истории в одном хранилище, исходящих из одной версии (точки ветвления). Ветки нужны для того, чтобы программисты могли вести совместную работу над проектом и не мешать друг другу при этом.

\10. Как и зачем можно игнорировать некоторые файлы при commit?

Игнорируемые файлы обычно представляют собой файлы, специфичные для платформы, или автоматически созданные из сборочных систем. Временно игнорировать изменения в файле можно командой: git update-index — assume-unchanged