Лабораторная работа №2

Управление версиями

Pavlova P.A.

Задание

Сделать базовую настройку git для дальнейшей работы.

Выполнение лабораторной работы

Установить git-flow в Fedora Linux (рис. 1.2)

image

Рис1.1 Листинг установки git-flow

image

Рис1.2 Установка git-flow в Fedora Linux

Установить gh в fedora Linux (рис. 1.4)

image

Рис1.3 Листинг установки gh

image

Рис1.4 Установка gh

Задать имя и email владельца репозитория (рис. 1.6)

image

Рис1.5 Листинг задания имени и email владельца репозитория

image

Рис1.6 Указывание имени и email владельца репозитория

Настроить utf-8 в выводе сообщений git (рис. 1.8)

image

Рис1.7 Листинг настройки utf-8

image

Рис1.8 Настройка utf-8 в выводе сообщений

Настройте верификацию и подписание коммитов git. – Зададим имя начальной ветки (будем называть её master). (рис. 1.10)

image

Рис1.9 Листинг задания начальной ветки

image

Рис1.10 Задание имени начальной ветки

Параметр autocrlf: (рис. 1.12)

image

Рис1.11 Листинг параметра autocrlf

image

Рис1.12 Параметр autocrlf

Параметр safecrlf:

image

Рис1.13 Листинг параметра safecrlf

image

Рис 1.14 Параметр safecrlf

Создайте ключи ssh. По алгоритму rsa с ключём размером 4096 бит: (рис. 2.2)

image

Рис2.1 Листинг создания ключа ssh

image

Рис2.2 Создание ключа ssh

По алгоритму ed25519 (рис. 2.4)

image

Рис2.3 Листинг алгоритма ed25519

image

Рис2.4 Алгоритм ed25519

Создайте ключи pgp. Генерируем ключ. (рис.3.2)

image

Рис3.1 Листинг генерации ключа

– Из предложенных опций выбираем: – тип RSA and RSA; – размер 4096; – выберите срок действия; значение по умолчанию— 0 (срок действия не истекает никогда). – GPG запросит личную информацию, которая сохранится в ключе: – Имя (не менее 5 символов). – Адрес электронной почты. – При вводе email убедитесь, что он соответствует адресу, используемому на GitHub. – Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым.

image

Рис3.2 Генерация ключа pgp

Добавление ключа pgp в GitHub. Выводим список ключей и копируем отпечаток приватного ключа: (рис. 4.2)

image

Рис4.1 Листинг вывода списка ключей

image

Рис4.2 Вывод списка ключей

Скопируйте ваш сгенерированный PGP ключ в буфер обмена: (рис. 4.3)

image

Рис4.3 Листинг копии сгенерированного ключа в буфер обмена

Перейдите в настройки GitHub (https://github.com/settings/keys), нажмите на кнопку New GPG key и вставьте полученный ключ в поле ввода. (рис. 4.4)

image

Рис4.4 Вставка полученного ключа в GitHub

Настройка автоматических подписей коммитов git. Используя введёный email, укажите Git применять его при подписи коммитов: (рис. 5.2)

image

Рис5.1 Листинг настройки автоматических подписей коммитов git

image

Рис5.2 Настройка автоматических подписей коммитов git.

Настройка gh. Авторизация (рис 6.2).

image

Рис6.1 Листинг авторизации

image

Рис6.2 Авторизация

Создание репозитория курса на основе шаблона. (рис. 7.2) Необходимо создать шаблон рабочего пространства. – Например, для 2021–2022 учебного года и предмета «Операционные системы» (код предмета os-intro) создание репозитория примет следующий вид:

image

Рис7.1 Листинг создания репозитория курса

Настройка каталога курса. Перейдите в каталог курса. Удалите лишние файлы. Создайте необходимые каталоги. Отправьте файлы на сервер. (рис. 7.5-7.6)

image

Рис7.1 Листинг перехода в каталог курса

image

Рис7.2 Листинг удаления лишних файлов

image

Рис7.3 Листинг создания необходимых каталогов

image

Рис7.4 Листинг отправки файлов на сервер

image

Рис7.5 Переход в каталог курса. Удаление лишних файлов. Создание необходимых каталогов.

image

Рис7.6 Отправка файлов на сервер

Выводы

Базовая конфигурация git была создана. Был создан локальный каталог для выполнения заданий по предмету.

Ответы на контрольные вопросы

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

VCS – программное обеспечение для облегчения работы с изменяющейся информацией. Применяется при работе нескольких человек над одним проектом. 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. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

- Хранилище - место хранения всех версий и служебной информации.
- Commit - синоним версии; процесс создания новой версии.
- История – изменения в репозитории.
- Рабочая копия - текущее состояние файлов проекта, основанное на версии, загруженной из хранилища (обычно на последней).

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

Централизованные: Простота использования. Вся история — всегда в едином общем хранилище. Нужно подключение к сети. Резервное копирование нужно только одному хранилищу. Удобство разделения прав доступа к хранилищу. Почти все изменения навсегда попадают в общее хранилище. Децентрализованные: Двухфазный commit: 1) запись в локальную историю; 2) пересылка изменений другим. Подключение к сети не нужно. Локальные хранилища могут служить резервными копиями. Локальное хранилище контролирует его владелец, но общее — администратор. Возможна правка локальной истории перед отправкой на сервер.

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

Создаётся новая версия в хранилище, делаем правки, после чего создаётся новая версия с правками. Так до конца работы над проектом.

5. Опишите порядок работы с общим хранилищем VCS.

Создаётся новая версия в хранилище. Каждый участник репозитория может получить локальную версию проекта и вносить изменения.

6. Каковы основные задачи, решаемые инструментальным средством git?

Хранение информации о всех изменениях в коде. Обеспечение удобства для 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
- 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
- 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
- отправка изменений конкретной ветки в центральный репозиторий: git push origin имя_ветки
- слияние ветки с текущим деревом: git merge --no-ff имя_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

8. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Локальный репозиторий – репозиторий, расположенный на локальном компьютере разработчика. Именно с ним работает программист. Удаленный репозиторий — репозиторий, находящийся на сервере. Это общий репозиторий, в который приходят все изменения.

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

- Ветка — это последовательность коммитов, в которой ведётся параллельная разработка какого-либо функционала.
- Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу.

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

Игнорируемые файлы обычно представляют собой файлы для конкретной платформы или автоматически созданные файлы из систем сборки. Игнорируемые файлы отслеживаются в специальном файле. gitignore, который регистрируется в корневом каталоге репозитория. Он использует шаблоны подстановки для сопоставления имен файлов с подстановочными знаками.