Лабораторная работа №2

НПИбд-01-25 №1032252598

Иванова Ангелина Олеговна

Содержание

# 1. Цель работы

Цель данной дабораторной работы – изучить идеологии и применение средств контроля версий, и получить практические навыки по работе с системой контроля версий git.

# 2. Теоретическое введение

## 2.1 Системы контроля версий. Общие понятия

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 yужную 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 центральныйрепозиторий 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.

## 2.2 Система контроля версий Git

Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями. Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.

## 2.3 Основные команды git

Наиболее часто используемые команды git представлены в таблице.

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

## 2.4 Стандартные процедуры работы при наличии центрального

репозитория Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория (при этом в локальное дерево до начала этой процедуры не должно было вноситься изменений):

git checkout master

git pull

git checkout -b имя\_ветки

Затем можно вносить изменения в локальном дереве и/или ветке. После завершения внесения какого-то изменения в файлы и/или каталоги проекта необ- ходимо разместить их в центральном репозитории. Для этого необходимо проверить, какие файлы изменились к текущему моменту:

git status

При необходимости удаляем лишние файлы, которые не хотим отправлять в центральный репозиторий. Затем полезно просмотреть текст изменений на предмет соответствия правилам ведения чистых коммитов:

git diff

Если какие-либо файлы не должны попасть в коммит, то помечаем только те файлы, изменения которых нужно сохранить. Для этого используем команды добавления и/или удаления с нужными опциями:

git add имена\_файлов

git rm имена\_файлов

Если нужно сохранить все изменения в текущем каталоге, то используем:

git add .

Затем сохраняем изменения, поясняя, что было сделано:

git commit -am “Some commit message”

и отправляем в центральный репозиторий:

git push origin имя\_ветки

или

git push

# 3. Выполнение лабораторной работы

## 3.1 Настройка github

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

|  |
| --- |
| Созданный аккаунт в Github  Рисунок 1: Созданный аккаунт в Github |

## 3.2 Базовая настройка git

Сначала сделаем предварительную конфигурацию git. Открыли терминал и ввели следующие команды, указав имя IvanovaAngelina и e-mail 1032252598@pfur.ru.

|  |
| --- |
| Команды с именем пользователя и e-mail  Рисунок 2: Команды с именем пользователя и e-mail |

Настроили utf-8 в выводе сообщений git.

|  |
| --- |
| Команда для настройки utf-8  Рисунок 3: Команда для настройки utf-8 |

Задали имя начальной ветки (назвали её master), а также установили параметры autocrlf и safecrlf.

|  |
| --- |
| Задание начальной ветки и установка параметров  Рисунок 4: Задание начальной ветки и установка параметров |

## 3.3 Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозиториев сгенерировали приватный и открытый ключи, которые сохранились в каталоге ~/.ssh/.

|  |
| --- |
| Генерация ключа  Рисунок 5: Генерация ключа |

Загрузили сгенерированный открытый ключ. Зашли на сайт http://github.org/ под своей учётной записью и перешли в меню Setting . После этого выбрали в боковом меню SSH and GPG keys и нажали кнопку New SSH key. Скопировали из локальной консоли ключ в буфер обмена

|  |
| --- |
| Копирование ключа  Рисунок 6: Копирование ключа |

Вставили ключ в появившееся на сайте поле и указали для ключа имя IvanovaAngelina. После чего нажали на кнопку Add SSH Key. Таким образом, успешно добавили ключ.

|  |
| --- |
| Подтверждение успешного добавления ключа  Рисунок 7: Подтверждение успешного добавления ключа |

## 3.4 Создание рабочего пространства и репозитория курса на основе шаблона

Открыли терминал и создали каталог для предмета «Архитектура компьютера»

|  |
| --- |
| Создание директории  Рисунок 8: Создание директории |

## 3.5 Создание репозитория курса на основе шаблона

Перешли на станицу репозитория с шаблоном курса https://github.com/yamadharma/cour se-directory-student-template. Далее выбрали Use this template. В открывшемся окне задали имя репозитория study\_2025–2026\_arh-pc и создали репозиторий, нажав кнопку ‘Create repository from template’

|  |
| --- |
| Проверка создания репозитория  Рисунок 9: Проверка создания репозитория |

Открыли терминал и перешли в каталог курса

|  |
| --- |
| Переход в каталог  Рисунок 10: Переход в каталог |

Клонировали созданный репозиторий

|  |
| --- |
| Клонирование репозитория  Рисунок 11: Клонирование репозитория |

## 3.6 Настройка каталога курса

Перешли в каталог курса.

|  |
| --- |
| Переход в каталог  Рисунок 12: Переход в каталог |

Создали необходимые каталоги.

|  |
| --- |
| Создание каталогов 1  Рисунок 13: Создание каталогов 1 |

|  |
| --- |
| Создание каталогов 2  Рисунок 14: Создание каталогов 2 |

Проверили правильность создания иерархии рабочего пространства в локальном репозитории и на странице github.

|  |
| --- |
| Проверка правильность создания  Рисунок 15: Проверка правильность создания |

## 3.7 Задание для самостоятельной работы

Создали отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs/lab02/report).

|  |
| --- |
| Создание отчета по 2 лабораторной  Рисунок 16: Создание отчета по 2 лабораторной |

Скопировали отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства с помощью команды mc

|  |
| --- |
| Копирование отчета по 2 лабораторной  Рисунок 17: Копирование отчета по 2 лабораторной |

Загрузили файлы на github.

|  |
| --- |
| Загрузка файлов на github  Рисунок 18: Загрузка файлов на github |

# 4. Выводы

Изучили идеологии и применение средств контроля версий, и получили практические навыки по работе с системой контроля версий git. Создали аккаунт github, связали со своим устройством, создали рабочее пространство и репозиторий и загрузили отчеты по первым лабораторным работам. ::: {#refs} :::