Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет имени Г.В. Плеханова»

**МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ**

специальность 09.02.07 «Информационные системы и программирование»

Квалификация: Программист

ПРАКТИЧЕСКИЕ РАБОТЫ

ПО МДК 04.02 «Обеспечение качества функционирования компьютерных систем»

|  |  |
| --- | --- |
| Выполнил студент  группы П50-3-20  Бобряшов Евгений Викторович | Проверил преподаватель  \_\_\_\_\_\_\_\_\_\_\_ К.С. Образцова  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2023 года |

Москва 2023

Оглавление

**Элементы оглавления не найдены.**

Практическая работа №1

Тема: «Работа с git».

Цель работы: изучить работу с git для создания локального репозитория, изучить работу основных команд для работы с файлами в репозитории. Научиться создавать ветки а также работу с указателями HEAD.

1. Лекция:

**Каскадные модели разработки**

Waterfall:

* Каждый этап один за другим – нельзя вернуться назад
* Этапы циркулируют только после завершения последнего этапа

**Гибкие модели разработки**

Scrum:

* В конце спринта имеется готовый продукт
* Гибкое изменение требований
* Быстрое реагирование на изменение трендов
* Бюджет не фиксирован

**Основные этапы разработки ПО:**

1. Анализ
2. Разработка
3. Сборка
4. Тестирование
5. Развертывание
6. Релиз (деплой)
7. Сопровождение
8. Мониторинг

**DevOps**

ДевОпс – беспреребойная поставка ПО с помощью непрерывной интеграции рабочих процессов.

- Ускорение этапов разработки.

- Снижение ошибок, связанных с человеческим фактором.

**DevSecOps**

**-** Интегрирует безопасность в процессы разработки и автоматизации.

**Инфраструктура как код**

Методология, которая предлагает использовать архитектурный подход к управлению инфраструктурными ресурсами, а именно автоматизировать создание и управление инфраструктурными компонентами с помощью программного кода.

Основная идея заключается в том, что вся инфраструктура, такая как виртуальные машины, сетевые настройки, хранилища данных и другие ресурсы, описывается в виде кода, который может быть управляем исходными файлами, хранящимися в системе контроля версий.

**CI/CD**

Методология разработки программного обеспечения, целью которой является автоматизация и ускорение процесса разработки, тестирования и доставки программного продукта.

Цель CI/CD состоит в том, чтобы обеспечить быстрое время от идеи до развертывания новых функций или исправлений ошибок, с учетом меньшей вероятности ошибок и конфликтов, а также поддержания высокого уровня качества кода и процесса разработки.

**Состав**

- Непрерывная интеграция (Continuous Integration)

- Непрерывная доставка (Continuous Delivery)

- Непрерывное развертывание (Continuous Deployment)

**Преимущества внедрения DevOps в разработку**

1. Ускорение процесса разработки.

DevOps позволяет автоматизировать и упростить различные этапы разработки, такие как сборка, тестирование и развертывание. Это позволяет сократить время, затрачиваемое на каждый из этих этапов, и значительно увеличить производительность команды разработки.

1. Улучшение качества ПО

DevOps способствует автоматизации тестирования и контроля качества, что позволяет ыстрее выявлять и устранять ошибки и повышать стабильность и надежность разрабатываемого продукта.

1. Снижение рисков.

Devops позволяет осуществлять постоянный мониторинг и контроль процесса разработки. Это позволяет оперативно реагировать на возникающие проблемы и снижает риски, связанные с качетсвом, безопасностью и доступностью приложения.

1. Более быстрая доставка нового функционала

DevOps способствует автоматизации процесса развертывания и доставки нового функционала пользователям. Благодаря этому, команда разработки может выпускать обновления и новые версии приложения значительно быстрее и с меньшими затратами.

1. Более гибкий и масштабируемый процесс разработки.

DevOps позволяет легко внедрять изменения и масштабировать процесс разработки в соответствии с потребностями пользователей или рынка. Это позволяет более гибко реагировать на изменения условий и быстро адаптироваться к новым требованиям.

1. Снижение затрат на разработку и эксплуатацию

DevOps позволяет оптимизировать процессы разработки и эксплуатации, что сокращает затраты на труд и ресурсы. Это позволяет сделать разработку ПО более эффективной и экономичной.

1. Более высокая удовлетворенность пользователей.

Благодаря улучшенному качеству ПО, более быстрой доставке нового функционала и более оперативной поддержке, пользователи получают более удовлетворительный опыт использования приложения. Это повышает удовлетворенность пользователей и способствует удержанию и привлечению новых кдиентов.

Конец лекции.

Лекция №2

Что такое система контроля версий и для чего она нужна

Система контроля версий - это специальное программное обеспечение, которое используется для управления изменениями в файловой системе, отслеживания и контроля версий документов или кода программы. Она позволяет разработчикам и команде проекта работать над файлами вместе, отслеживать изменения, управлять их версиями и восстанавливаться к предыдущим состояниям проекта.

Преимущества применения системы контроля версий

1. История изменений

2. Контроль версий

3. Коллективная работа

4. Ветвление и слияние

5. Отслеживание ошибок

Распределенная VS Централизованная системы контроля версий

Архитектура: Централизованная система контроля версий имеет единую центральную базу данных, где хранятся все файлы и изменения. В распределенной системе каждый пользователь имеет собственную копию репозитория, включая всю историю изменений.

Работа в офлайн-режиме: Распределенная система позволяет работать в офлайн-режиме, так как каждый пользователь имеет полную копию репозитория. В централизованной системе требуется подключение к центральному серверу для выполнения большинства операций.

Управление правами доступа: В централизованной системе управление правами доступа к репозиторию обычно происходит на уровне центрального сервера. В распределенной системе каждый пользователь может выполнять операции с собственной копией репозитория, что позволяет более гибко управлять правами доступа.

Скорость работы: Распределенная система обычно быстрее выполняет операции, так как все данные хранятся локально у каждого пользователя. В централизованной системе операции происходят через удаленное соединение, что может быть медленнее при большом количестве пользователей.

snapshot-based

Система контроля версий с механизмом снимков (например, Git, Mercurial) хранят все версии файлов и директорий в форме полных копий (снимков) состояния проекта в определенный момент времени. Когда вы делаете коммит, система фиксирует все измененные файлы и создает новый снимок, который включает в себя полные копии всех измененных файлов. Это позволяет быстро переключаться между разными версиями проекта и осуществлять обход старых коммитов.

delta-based

Система контроля версий со списком изменений (например, Subversion) хранят только разницу (delta) между последующими версиями файлов. Когда вы делаете коммит, система сохраняет только изменения, сделанные в файлах, относительно предыдущего коммента, в виде списка изменений. На самом деле, система часто хранит несколько версий файла, чтобы быстрее обрабатывать запросы, основанные на списке изменений. При переключении на предыдущие версии проекта, СКВ применяет все изменения последовательно, чтобы восстановить запрошенную версию проекта.

Конец лекций.

Для начала проверяем текущую версию гита, и, так как доступно обновление, скачиваем его:

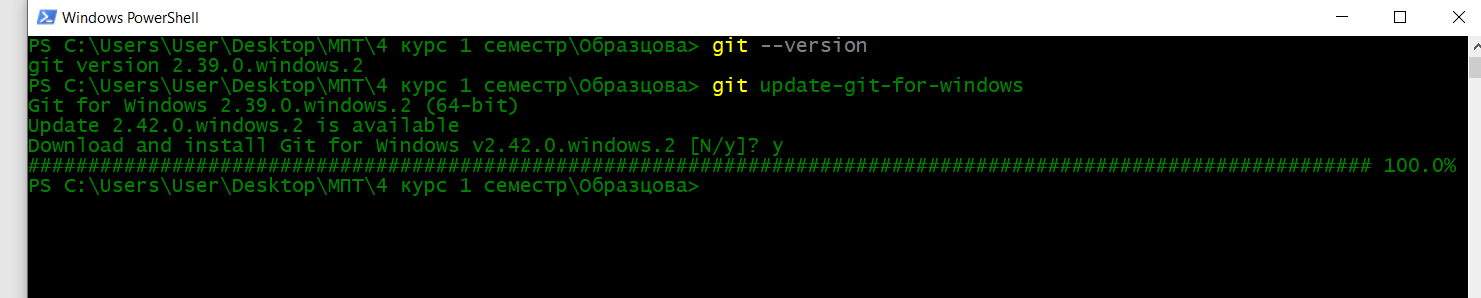


Рисунок 1. Обновление гита

1. Далее, настроим конфигурацию гита, для этого устанавливаем имя и почту пользователя при помощи следующих команд:

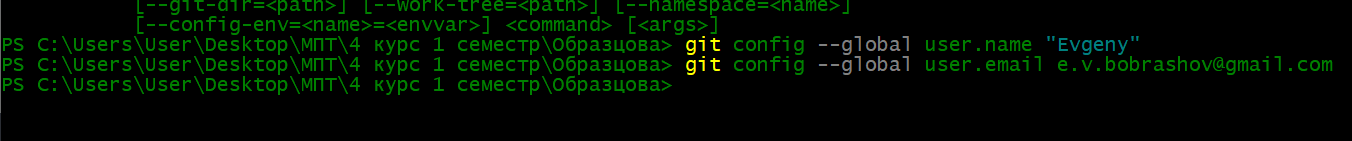


Рисунок 2. Настройка конфигурации

Также отключим замену некоторых символов восьмиричными кодами при помощи «core.quotePath» и

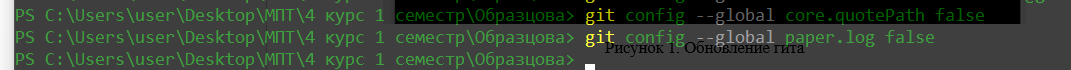


Рисунок . Продолжение настройки конфигурации

1. Затем, создадим локальный репозиторий при помощи команды «git init»:

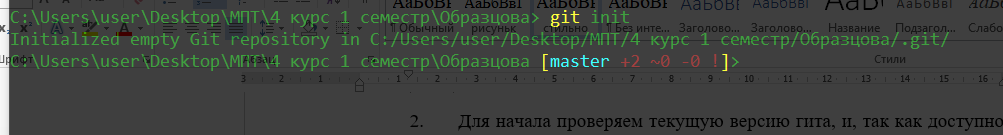


Рисунок . Создание локального репозитория

1. Используем команды «git status» для проверки текущего статуса репозитория, а также «git log» для вывода логов коммита проекта:

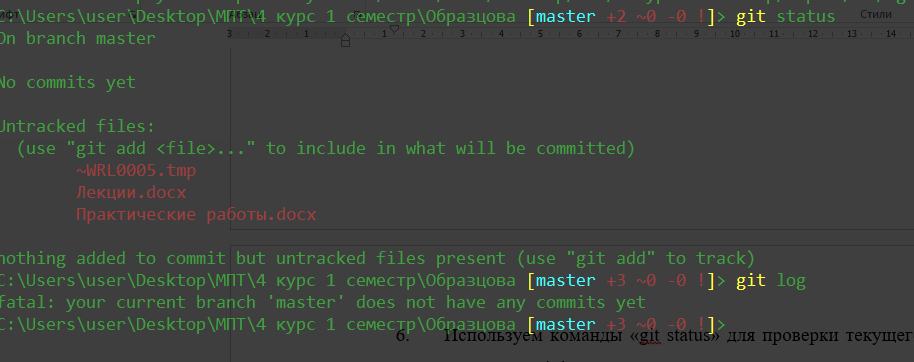


Рисунок . Статус репозитория и логи

Проверить текущую конфигурацию гита можно при помощи ключа –list:

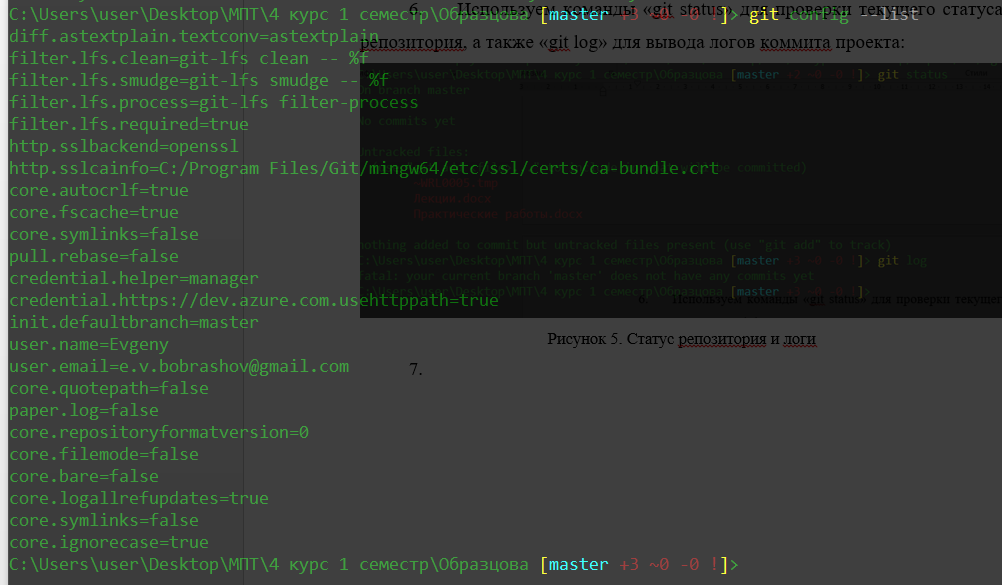


Рисунок . Текущая конфигурация гита

Затем, добавим файлы из текущего каталога в раздел проиндексированных файлов при помощи «git add», а затем запишем эти файлы в репозиторий при помощи «git commit»:

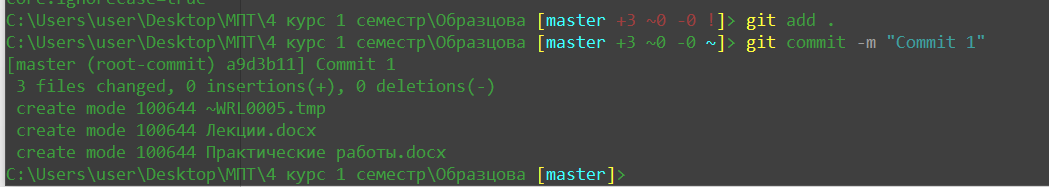


Рисунок . Добавление файлов и их запись

Попробуем команду «git rm». Она позволяет удалить файл из списка проиндексированных:

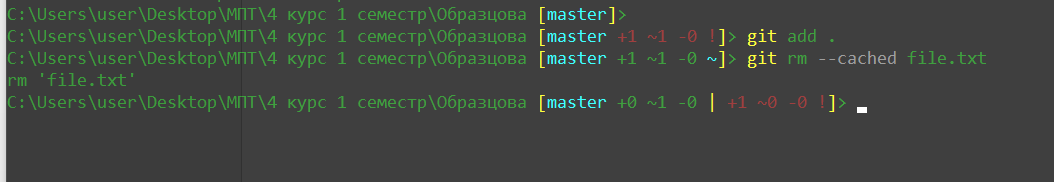


Рисунок . Удаление из индексированных файлов

Проверим логи проекта:

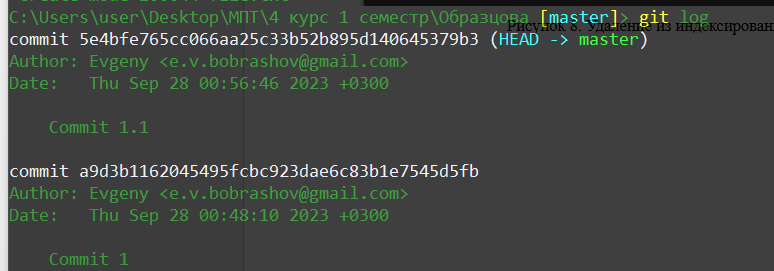


Рисунок . Повторный вывод логов

Также мы можем использовать блокнот или Visual Code для коммитов, для примера используем блокнот, для этого воспользуемся командой «core.editor notepad». Затем добавим файл в индексируемые и пропишем commit и у нас откроется блокнот:



Рисунок . Использование блокнота для коммитов

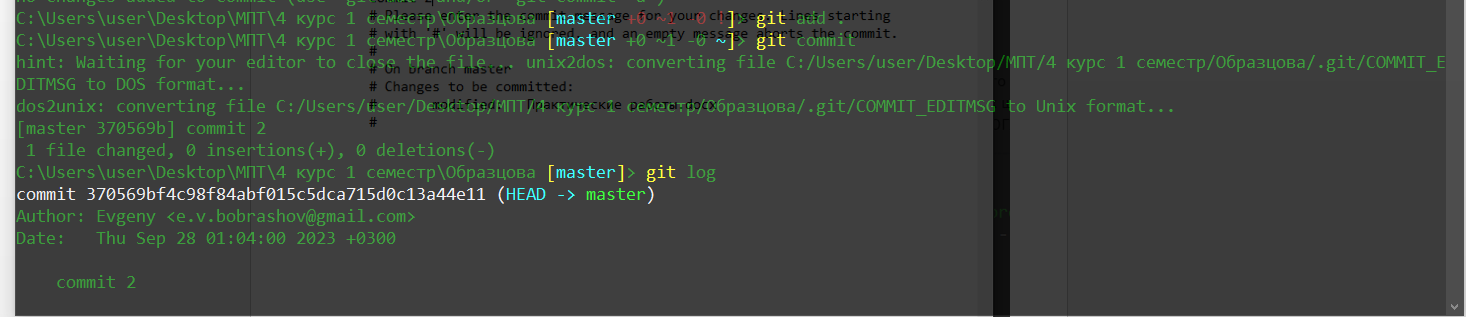


Рисунок . Команды для коммита в блокноте

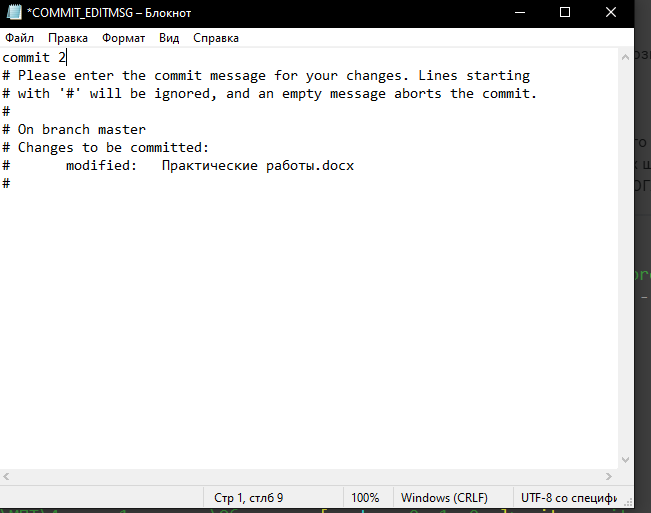


Рисунок . Коммит в блокноте

Воспользуемся Visual Code, а также добавим один файл в список индексированных, и при помощи команды «git diff» выведем разницу между текущим не отслеживаемым состоянием репозитория и последним снимком репозитория:

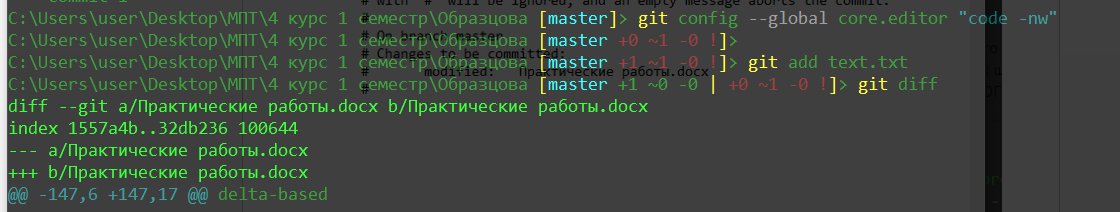


Рисунок . Вывод разницы состояний

Для вывода разницы уже между отслеживаемым состоянием и последним снимком репозитория указываем ключ «--staged»:

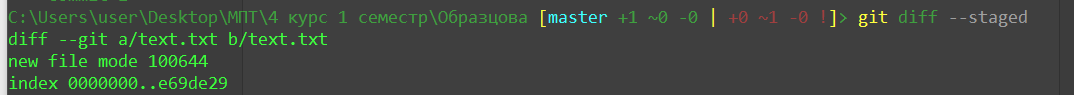


Рисунок . Вывод разницы состояний (отслеживаемого)

1. Для создания ветки используем команду «git branch» и указываем название ветки, без указания названия команда выведет нам список веток. Для удаления ветки используется ключ «-d». «checkout» позволит нам переключаться между ветками а с ключом «-b» можно сразу создать ветку:

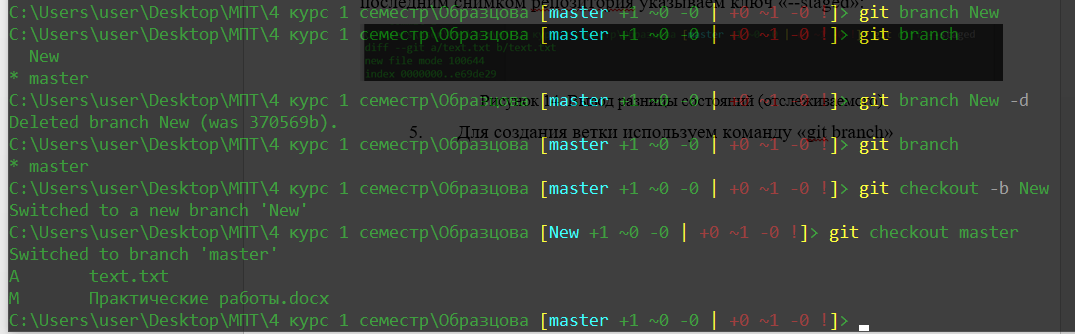


Рисунок . Работа с ветками

1. Перейдём к работе с указателями HEAD, команда «git reset --hard» возвращает проект к указанному коммиту, при этом полностью удаляя все коммиты после указанного:



Рисунок . Hard reset