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 П50-4-21  Игошев Р. В. | Проверил преподаватель  \_\_\_\_\_\_\_\_\_\_\_ К.С. Образцова  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 года |

Москва 2024

СОДЕРЖАНИЕ

[Конспект лекций. 3](#_Toc177148892)

[Практическая работа №1. 7](#_Toc177148893)

# Конспект лекций.

Каскадные модели ПО

Waterfall – каждый этап идет за другим, подробная документация.

SCRUM – в конце спринта имеется готовый продукт, гибкое изменение требований, быстрое реагирование на изменение трендов, бюджет не фиксирован.

Спринт – период в две недели, за который реализовывается какой-либо готовый продукт (окно, модуль и т.д.).

Основные этапы разработки ПО:

1. Анализ

2. Разработка

3. Сборка

4. Тестирование

5. Развертывание

6. Релиз (деплой)

7. Сопровождение

8. Мониторинг

DevOps – бесперебойная поставка ПО с помощью непрерывной интеграции рабочих процессов. Ускоряет этапы разработки, снижает ошибки, связанные с человеческим фактором.

DevSecOps – интегрирует безопасность в процессы разработки и автоматизации.

Инфраструктура как код – методология, которая предлагает использовать архитектурный подход к управлению инфраструктурными ресурсами.

CI/CD – методология разработки по, целью которой является автоматизация и ускорение процесса разработки. Цель – обеспечить быстрое время от идеи до развертывания новых функций или исправлений ошибок, с учетом меньшей вероятности ошибок и конфликтов.

Состав CI/CD

• Непрерывная интеграция

• Непрерывная доставка

• Непрерывное развертывание

Преимущества:

1. Ускорение процесса разработки

2. Улучшение качества ПО

3. Снижение рисков

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 использования системы контроля версий:

1. История изменений

2. Контроль версий

3. Коллективная работа

4. Ветвление и слияние

5. Отслеживание ошибок

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.

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.

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 коммита, 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.

Коммит (commit) представляет собой операцию, при которой изменения в файловой системе, внесенные программистом, сохраняются в репозитории проекта. Коммит фиксирует изменения в файле или наборе файлов, а также добавляет комментарии или описания к этим изменениям. Каждый коммит имеет уникальный идентификатор, который позволяет отслеживать историю изменений и восстанавливать предыдущие версии файлов. Коммиты также позволяют работать с несколькими программистами над одним проектом, объединяя изменения, внесенные каждым из них.

Git - это распределенная система управления версиями, которая используется для отслеживания изменений в коде программного обеспечения. Git позволяет разработчикам работать над проектом одновременно и совместно, сохраняя историю изменений и легко управляя версиями. Git также обеспечивает возможность ветвления и слияния кода, у также гибкую работу с удаленными репозиториями.

Staged - это когда проект на гите уже изменился с новыми файлами или измененными.

git init -создание нового репозитория Git для проекта

git status - вывод текущего статуса репозитория

git log - вывод логов коммитов проекта

git config --list вывод текущей конфигурации Git

git add <file> - добавление файлов из рабочего каталога в раздел проиндексированных файлов

git rm <file> - удаление отслеживаемых файлов из раздела проиндексированных файлов

git commit -m - запись индексированных репозиторий изменений в Git

git diff - вывод разницы между текущим не отслеживаемым состоянием и последним снимком репозитория

git diff --staged вывод разницы между текущим отслеживаемым

состоянием и последним снимком репозитория

git restore <file> Откатывает файл на последнюю зафиксированную версию

git reset --hard номер коммита - Возвращает проект к указанному коммиту, при этом полностью удаляет все коммиты после указанного безвозвратно.

git reset --soft номер коммита - Возвращает проект к указанному коммиту, при этом переводит все коммиты после указанного в отслеживаемую (staged) зону.

git reset --mixed номер коммита - Возвращает проект к указанному коммиту, при этом переводит все коммиты после указанного в неотслеживаемую (unstaged) зону.

# Практическая работа №1.

Тема: Работа с файлами в локальном репозитории Git.

1. Настройка конфигурации и вывод конфигурации Git.

Начать настройку конфигурации git следует с выполнения команд, расположенных снизу на изображении.

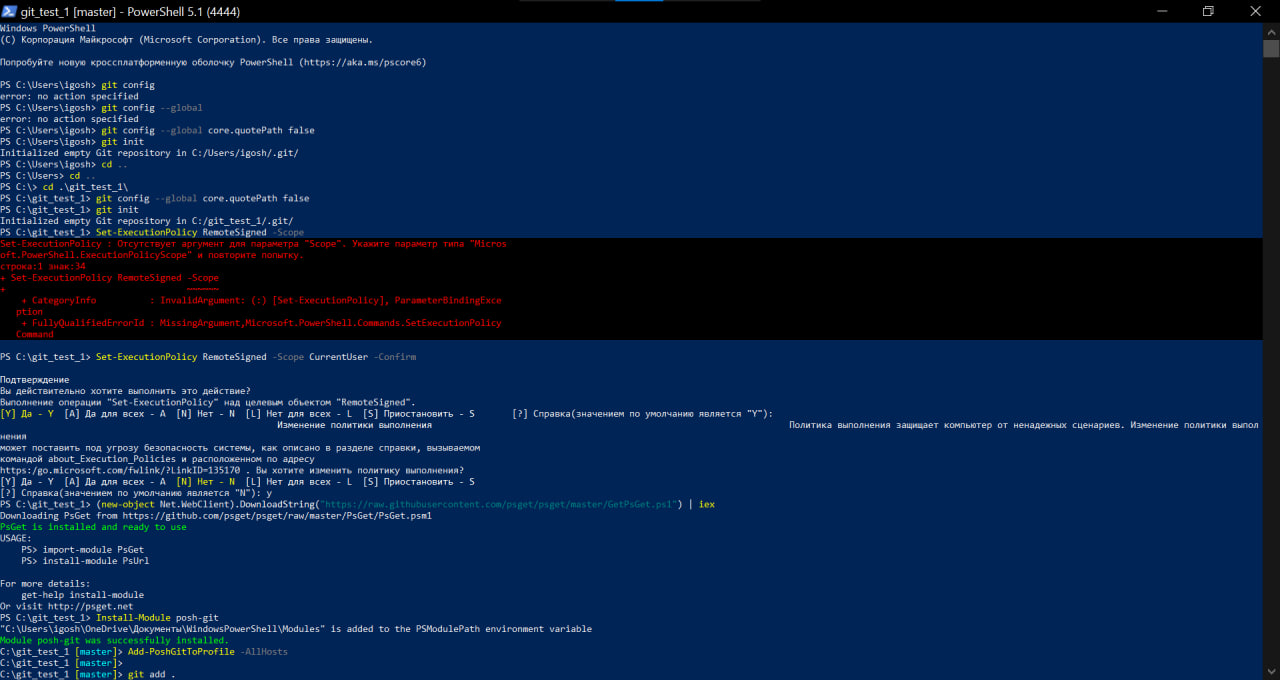


Рисунок 1 – Настройка конфигурации git

git config --global core.quotePath false – команда настраивает консоль таким образом, чтобы путь и названия файлов выводились без кавычек, сплошным текстом.

Set-ExecutionPolicy RemoteSigned -Scope CurrentUser –Confirm – позволяет выполнять PowerShell скрипты с подписью.

(new-object Net.WebClient).DownloadString("https://raw.githubusercontent.com/psget/psget/master/GetPsGet.ps1") | iex – устанавливает пакеты для PowerShell.

Install-Module posh-git – устанавливает пакет posh-git.

Add-PoshGitToProfile –AllHosts – эта команда добавляет posh-git в PowerShell для всех будущих сеансов.

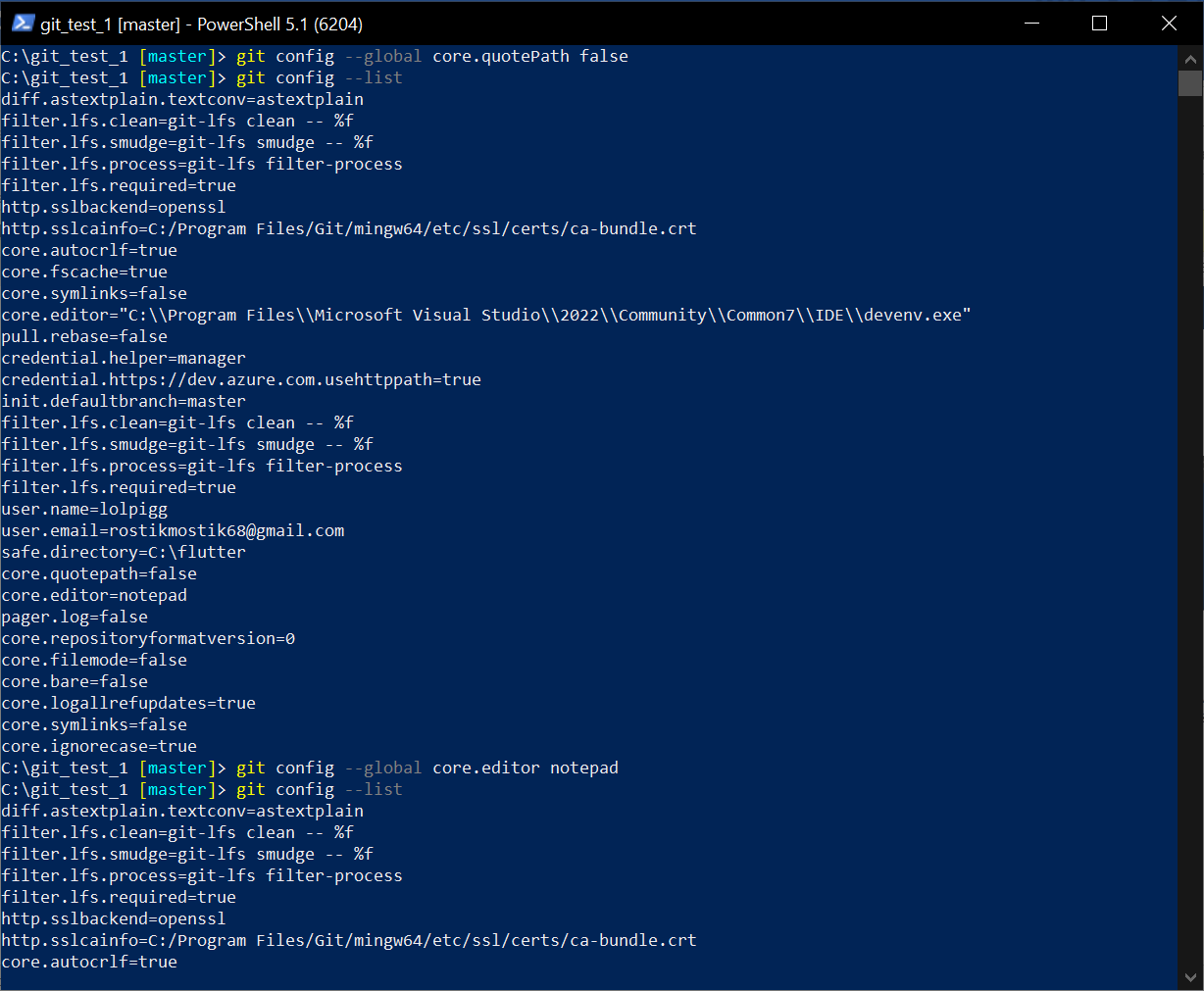


Рисунок 2 – Вывод конфигурации git

Теперь командой git config –list можно вывести конфигурацию git и рассмотреть настройки подробнее.

1. Создание локального репозитория

Локальный репозиторий создается в папке с помощью команды git init, результат которой показан на изображении ниже.

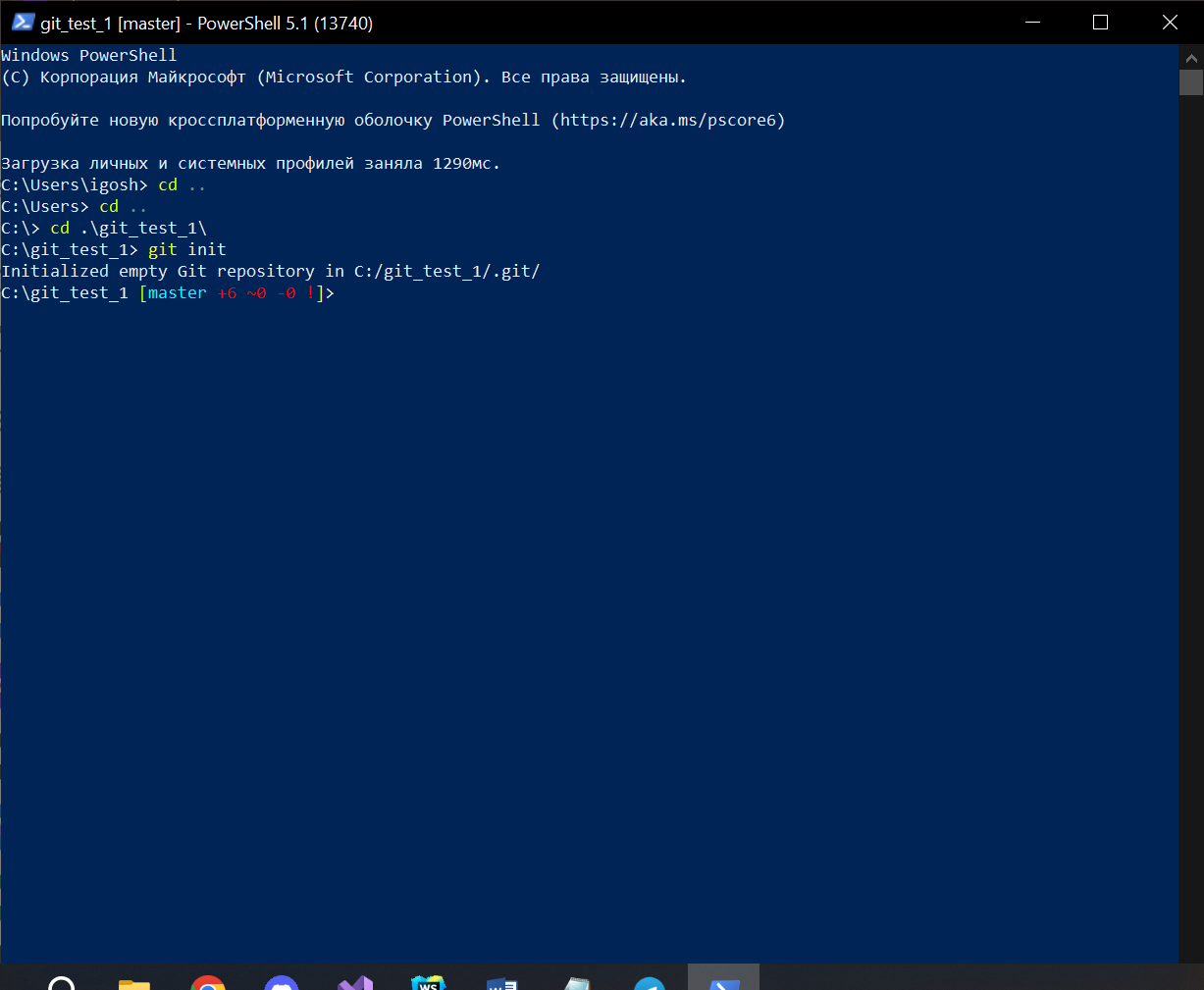


Рисунок 3 – Создание локального репозитория в папке

1. Демонстрация работы основных команд для работы с файлами в репозитории

Для начала отслеживания файла необходимо использовать команду git add “имя файла”, или же git add . для начала отслеживания всего содержимого папки разом. Удалить из отслеживания файл можно командой git rm “имя файла”. Сохранить изменения в отслеживании можно используя команду git commit, которая сохранит изменения в новом коммите.

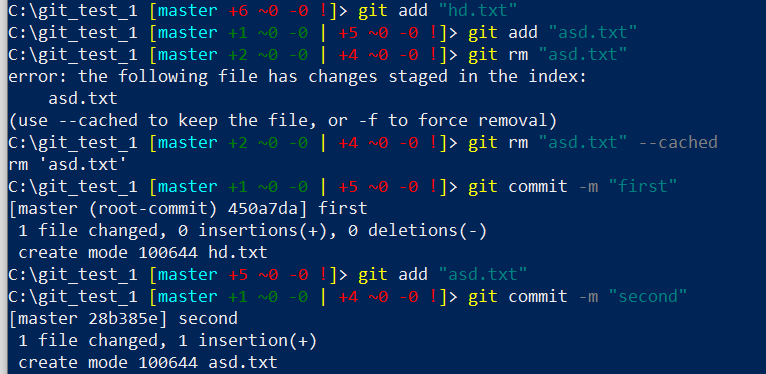


Рисунок 4 – Добавление файлов в отслеживание, удаление оттуда, создание коммита

Командой git log можно вывести все созданные коммиты. Текущая ветка будет выделена надписью HEAD -> “название ветки”. Сама же надпись будет напротив активного коммита.

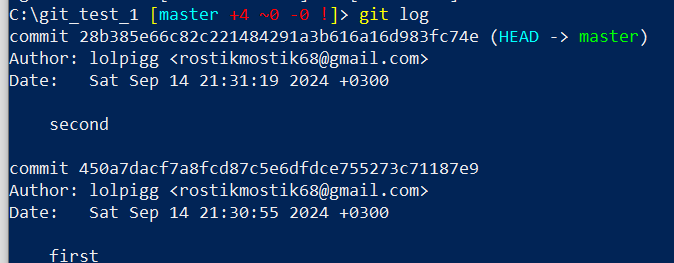


Рисунок 5 – Вывод логов коммитов

Команды git reset с параметрами позволяют перемещаться между коммитами, параметры позволяют определенным образом взаимодействовать с ними, soft просто меняет активный коммит, hard удаляет же все коммиты после выбранного.

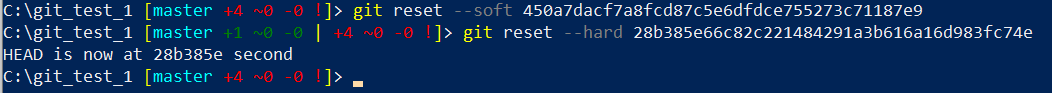


Рисунок 6 – Мягкий откат к коммиту и жесткий откат

1. Работа с указателями HEAD

Указатель HEAD показывает активную ветку, помогая в навигации по репозиторию. Создать новую ветку можно командой git checkout –b “название ветки”. В примере ниже используется изменение файла через команду echo, которая вставляет в файл определенный текст, после чего создается коммит, и все это на второй ветке. Далее выводится информация логов в одну линию через git log с параметром oneline, где видно указание второй ветки как активной. Далее можно переключиться обратно на основную ветку с помощью команды git checkout master.

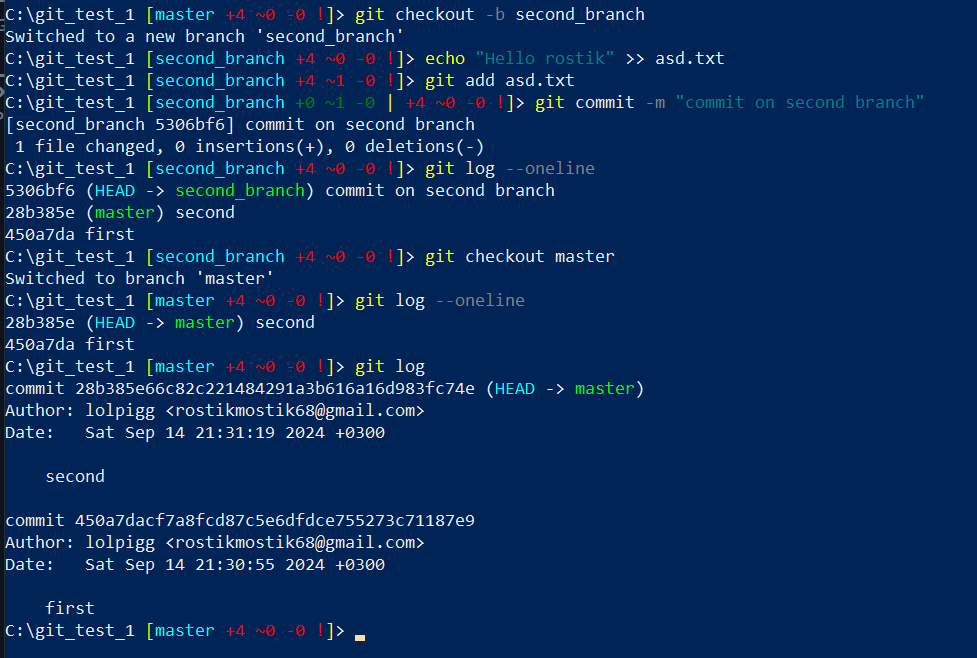


Рисунок 7 – Создание второй ветки, вывод указателей HEAD

Вывод: Продемонстрирована настройка и вывод конфигурации Git. Создан локальный репозиторий. Продемонстрирована работа основных команд для работы с файлами в репозитории.