



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных
технологий

Отчет по практической работе №1

по дисциплине «Технологии разработки программных приложений»

Тема практической работы: «Системы контроля версий»

Выполнил:

Студент группы ИКБО-41-23

Проверил:

Доцент кафедры МОСИТ,
кандидат технических наук, доцент
Жматов Д.В.

Москва 2025

Оглавление

ЧАСТЬ 1. ОСНОВНЫЕ КОМАНДЫ GIT	3
ЧАСТЬ 2. СИСТЕМЫ УПРАВЛЕНИЯ РЕПОЗИТОРИЯМИ	6
ЧАСТЬ 3. РАБОТА С ВЕТВЛЕНИЕМ И ОФОРМЛЕНИЕ КОДА	8
КОНТРОЛЬНЫЕ ВОПРОСЫ	16
ВЫВОД.....	17
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ	18

ЧАСТЬ 1. ОСНОВНЫЕ КОМАНДЫ GIT

1. Установить и настроить клиент git на своей рабочей станции.
2. Создать локальный репозиторий и добавить в него несколько файлов.
3. Внести изменения в один из файлов
4. Проиндексировать изменение и проверьте состояние.
5. Сделать коммит того, что было проиндексировано в репозиторий.
Добавить к коммиту комментарий.

6. Изменить еще один файл. Добавить это изменение в индекс `git`. Изменить файл еще раз. Проверить состояние и произведите коммит проиндексированного изменения. Теперь добавить второе изменение в индекс, а затем проверить состояние с помощью команды `git status`. Сделать коммит второго изменения.

7. Просмотреть историю коммитов с помощью команды `git log`. Ознакомиться с параметрами команды и использовать некоторые из них для различного формата отображения истории коммитов.

8. Вернуть рабочий каталог к одному из предыдущих состояний.

9. Изучить, как создавать теги для коммитов для использования в будущем.

10. Отмените некоторые изменения в рабочем каталоге (до и после индексирования)

11. Отменить один из коммитов в локальном репозитории.

ЧАСТЬ 2. СИСТЕМЫ УПРАВЛЕНИЯ РЕПОЗИТОРИЯМИ

1. Создать аккаунты на GitHub.

Уже 5 лет создан:

2. Создать репозиторий на GitHub и на локальной машине, согласно выбранной теме проекта

Репозиторий test:

3. Создайте несколько файлов на локальной машине при помощи консоли

4. Создайте SSH-ключ для авторизации

5. Свяжите репозиторий локальной машины с репозиторием на GitHub при помощи консоли

6. Создайте новую ветку в репозитории с помощью команды, произведите в ней какие-нибудь изменения, а после слейте с веткой main

7. Выполните цепочку действий в репозитории, согласно вариантам (Вариант 4):

1) Клонировать непустой удаленный репозиторий на локальную машину:

2) Создайте новую ветку и выведите список всех веток:

3) Произведите 3 коммитов в новой ветке в разные файлы:

4) Выгрузите изменения в удаленный репозиторий:

5) Откатите в новой ветке предпоследний коммит (в том числе в удаленном репозитории):

6) Выведите в консоли различия между веткой master и новой веткой:

7) Слейте новую ветку с master при помощи merge:

ЧАСТЬ 3. РАБОТА С ВЕТВЛЕНИЕМ И ОФОРМЛЕНИЕ КОДА

1. Сделайте форк репозитория в соответствии с вашим вариантом
2. Склонируйте его на локальную машину
3. Создайте две ветки `branch1` и `branch2` от последнего коммита в `master`

`free-for-dev` перемещена на рабочий стол:

4. Проведите по 3 коммита в каждую из веток, которые меняют один и тот же кусочек файла

Файл `README.md` до изменений:

README – Блокнот

Файл Правка Формат Вид Справка

free-for.dev

Developers and Open Source authors now have many services offering free t

This is a list of software (SaaS, PaaS, IaaS, etc.) and other offerings w

The scope of this particular list is limited to things that infrastructure

This list results from Pull Requests, reviews, ideas, and work done by 16

[![Track Awesome List](https://www.trackawesomelist.com/badge.svg)](https

****NOTE****: This list is only for as-a-Service offerings, not for self-host

Table of Contents

- * [Major Cloud Providers' Always-Free Limits](#major-cloud-providers)
- * [Cloud management solutions](#cloud-management-solutions)
- * [Analytics, Events, and Statistics](#analytics-events-and-statistics)
- * [APIs, Data and ML](#apis-data-and-ml)
- * [Artifact Repos](#artifact-repos)
- * [BaaS](#baas)
- * [Low-code Platform](#low-code-platform)
- * [CDN and Protection](#cdn-and-protection)
- * [CI and CD](#ci-and-cd)
- * [CMS](#cms)
- * [Code Generation](#code-generation)
- * [Code Quality](#code-quality)

Первое изменение, первый коммит, branch1:

README – Блокнот

Файл Правка Формат Вид Справка

первое изменение 1 в branch1

free-for.dev

Developers and Open Source authors now have many services offering free t

This is a list of software (SaaS, PaaS, IaaS, etc.) and other offerings v

The scope of this particular list is limited to things that infrastructure


This list results from Pull Requests, reviews, ideas, and work done by 16

[![Track Awesome List](https://www.trackawesomelist.com/badge.svg)](https

****NOTE****: This list is only for as-a-Service offerings, not for self-host

Table of Contents

Второе изменение, второй коммит, branch1:

 README – Блокнот


Файл Правка Формат Вид Справка

первое изменение 1 в branch1
второе изменение 2 в branch1
free-for.dev

Developers and Open Source author

This is a list of software (SaaS,

Третье изменение, третий коммит, branch1:

 README – Блокнот


Файл Правка Формат Вид Справка

первое изменение 1 в branch1
второе изменение 2 в branch1
третье изменение 3 в branch1
free-for.dev

Developers and Open Source author

This is a list of software (SaaS,

Первое изменение, первый коммит, branch2:

 README – Блокнот

Файл Правка Формат Вид Справка

первое изменение 1 в branch2
free-for.dev

Developers and Open Source authors

This is a list of software (SaaS,

Второе изменение, второй коммит, branch2:

```
README – Блокнот
Файл  Правка  Формат  Вид  Справка
первое изменение 1 в branch2
второе изменение 2 в branch2
# free-for.dev

Developers and Open Source authors
```

Третье изменение, третий коммит, branch2:

```
README – Блокнот
Файл  Правка  Формат  Вид  Справка
первое изменение 1 в branch2
второе изменение 2 в branch2
третье изменение 3 в branch2
# free-for.dev

Developers and Open Source authors
```

5. Выполните слияние ветки branch1 в ветку branch2, разрешив конфликты при этом

Конфликт:

```
README – Блокнот
Файл  Правка  Формат  Вид  Справка
|<<<<<<< HEAD
первое изменение 1 в branch2
второе изменение 2 в branch2
третье изменение 3 в branch2
=====
первое изменение 1 в branch1
второе изменение 2 в branch1
третье изменение 3 в branch1
>>>>>>> branch1
# free-for.dev

Developers and Open Source authors
```

Разрешение конфликта:



README – Блокнот

Файл Правка Формат Вид Справка

первое изменение 1 в branch2

второе изменение 2 в branch2

третье изменение 3 в branch2

free-for.dev

Developers and Open Source authors

6. Выгрузите все изменения во всех ветках в удаленный репозиторий

7. Проведите еще 3 коммита в ветку branch1

Первое изменение, первый коммит, branch1:



README – Блокнот

Файл Правка Формат Вид Справка

первое изменение 1 в branch1
второе изменение 2 в branch1
третье изменение 3 в branch1

первое первое изменение 1 в branch1
free-for.dev

Developers and Open Source authors

Второй коммит, второе изменение, branch1:



README – Блокнот

Файл Правка Формат Вид Справка

первое изменение 1 в branch1
второе изменение 2 в branch1
третье изменение 3 в branch1

первое первое изменение 1 в branch1
второе второе изменение 2 в branch1
free-for.dev

Developers and Open Source authors

Третий коммит, третье изменение, branch1:



README – Блокнот

Файл Правка Формат Вид Справка

первое изменение 1 в branch1
второе изменение 2 в branch1
третье изменение 3 в branch1

первое первое изменение 1 в branch1
второе второе изменение 2 в branch1
третье третье изменение 3 в branch1
free-for.dev

Developers and Open Source authors

8. Склонируйте репозиторий еще раз в другую директорию

9. В новом клоне репозитории сделайте 3 коммита в ветку branch1

Первый коммит, первое изменение:

README – Блокнот
Файл Правка Формат Вид Справка
первое изменение 1 в branch1
второе изменение 2 в branch1
третье изменение 3 в branch1

первое первое первое изменение 1 в branch1
free-for.dev

Второй коммит, второе изменение:

README – Блокнот
Файл Правка Формат Вид Справка
первое изменение 1 в branch1
второе изменение 2 в branch1
третье изменение 3 в branch1

первое первое первое изменение 1 в branch1
второе второе второе изменение 2 в branch1
free-for.dev

Третий коммит, третье изменение:

README – Блокнот
Файл Правка Формат Вид Справка
первое изменение 1 в branch1
второе изменение 2 в branch1
третье изменение 3 в branch1

первое первое первое изменение 1 в branch1
второе второе второе изменение 2 в branch1
третье третье третье изменение 3 в branch1
free-for.dev

10. Выгрузите все изменения из нового репозитория в удаленный репозиторий

11. Вернитесь в старый клон с репозиторием, выгрузите изменения с опцией--force

12. Получите все изменения в новом репозитории

Конфликт:

```
README – Блокнот
Файл Правка Формат Вид Справка
первое изменение 1 в branch1
второе изменение 2 в branch1
третье изменение 3 в branch1

<<<<<<< HEAD
первое первое первое изменение 1 в branch1
второе второе второе изменение 2 в branch1
третье третье третье изменение 3 в branch1
=====
первое первое изменение 1 в branch1
второе второе изменение 2 в branch1
третье третье изменение 3 в branch1
>>>>>>> c444c05dce490e26b56274dbd96abb09c68954d7
# free-for.dev
```

Разрешение конфликта:

```
README – Блокнот
Файл Правка Формат Вид Справка
первое изменение 1 в branch1
второе изменение 2 в branch1
третье изменение 3 в branch1

первое первое первое изменение 1 в branch1
второе второе второе изменение 2 в branch1
третье третье третье изменение 3 в branch1
# free-for.dev
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

2. К какому типу систем контроля версий относится Git?

Git относится к распределительной системе контроля версий, в таких системах разработчик имеет локальную копию репозитория со всеми комитами

8. Что такое конфликт в Git? Как его решить и почему они бывают?

Конфликт это состояние репозитория, когда изменения в двух ветках затрагивают одни и те же строки в файле, и Git не может автоматически определить, какое изменение должно быть применено. Конфликт можно решить вручную, отредактировав файл и выбрав нужный вариант.

9. Как отменить слияние веток, если произошел конфликт?

Чтобы отменить слияние веток нужно воспользоваться командой `git merge --abort`, это вернёт всё в состояние до начала слияния.

11. Что делает команда `git status`?

Команда `git status` показывает текущее состояние рабочего каталога и индекса, это помогает понять какие файлы были изменены и какие файлы нужно проиндексировать.

15. Что делает команда `git show`?

Показывает информацию об объекте(коммите, дереве или теге) для которого мы вызываем эту команду, выводит изменения введенные в последнем коммите и его автора, дату.

19. Чем отличаются команды "`git push`" и "`git pull`"?

`git push` отправляет изменения локального репозитория в удалённый, а

`git pull` наоборот, забирает изменения из удалённого репозитория и объединяет с локальной веткой.

ВЫВОД

В ходе выполнения данной практической работы были изучены основные принципы работы с системой контроля версий Git, включая базовые команды, управление локальными и удалёнными репозиториями, работу с ветками и разрешение конфликтов.

Были рассмотрены такие важные аспекты, как создание и настройка локального репозитория, выполнение коммитов, откат изменений, работа с

тегами, а также взаимодействие с удалёнными репозиториями на GitHub. Отдельное внимание было уделено управлению ветвлением, слиянию изменений и разрешению возможных конфликтов.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. GIT: Создание прочной основы для эффективной разработки. Ч. 1. : учебное пособие / Д. В. Жматов .— Москва : РТУ МИРЭА , 2024
2. Турнецкая Е. Л., Аграновский А. В. Программная инженерия. Интеграционный подход к разработке» (Турнецкая, Е. Л. Программная инженерия. Интеграционный подход к разработке / Е. Л. Турнецкая, А. В. Аграновский. — Санкт-Петербург : Лань, 2023.