Лабораторные работы по Git

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc17899)

[Лабораторная работа №1 «Первоначальная настройка git» 4](#_Toc17900)

[Лабораторная работа № 2 «Игнорирование, сравнение, удаление и](#_Toc17901)

[перемещение файлов» 5](#_Toc17902)

[Лабораторная работа №3 «Просмотр истории коммитов» 6](#_Toc17903)

[Лабораторная работа №4 «Отмена изменений. Работа с метками» 7](#_Toc17904)

[Лабораторная работа №5 «Ветвление. Конфликты» 8](#_Toc17905)

[Лабораторная работа №6 «Прятанье» 9](#_Toc17906)

[Лабораторная работа №7 «Работа с удаленным репозиторием» 10](#_Toc17907)

[Методические указания по выполнению самостоятельной работы 12](#_Toc17908)

Рекомендуемая литература .................................................................................... 13

## Введение

Лабораторные работы по данной части дисциплины имеют целью: закрепление теоретического материала, получение навыков самостоятельной работы с системой контроля версий Git.

## Лабораторная работа №1 «Первоначальная настройка git»

*Тема: Первоначальная настройка git. Инициализация каталога. Состояния фалов в git. Первый коммит.*

*Цель работы: провести первоначальную настройку системы контроля версии git, после установки инициализировать каталог для работы, разобраться с существующими состояниями файлов в git, сделать первый коммит.*

В состав git'а входит утилита **git config**, которая позволяет просматривать и устанавливать параметры, контролирующие все аспекты работы git'а и его внешний вид.

Первое, что необходимо сделать после установки git'а, — указать имя и адрес электронной почты. Это важно, потому что каждый коммит в git'е содержит эту информацию, и она включена в коммиты, передаваемые разработчиками, и не может быть далее изменена.

Для того чтобы начать использовать git для существующего проекта, необходимо перейти в проектный каталог и в командной строке ввести **git init**.

Эта команда создаёт в текущем каталоге новый подкаталог с именем .git содержащий все необходимые файлы репозитория — основу git-репозитория. На этом этапе проект ещё не находится под версионным контролем. Данная команда инициализирует возможность работы с git, но не вносит файлы под контроль.

**Порядок выполнения работы**

1. Изучить теоретическую часть работы.
2. Зайти в папку лабораторной работы и в ней создать папку соответствующую инициалам студента на английском языке. Например, для студента Иванов Петр Петрович, папка будет иметь имя IPP.
3. Провести инициализацию репозитория в созданной папке. Для этого, открыть программу **Git Bash**, перейти в созданную папку (для перемещения используется команда **cd**).
4. Установить настройки имени и e-mail’а, не используя опцию **--global.**
5. Создать в папке файл my\_first\_file.txt и проиндексировать его.
6. Сделать первый коммит.
7. Открыть файл my\_first\_file.txt и добавить в него строчку “test row”. Проиндексировать изменения.
8. Создать новый файл my\_second\_file.txt. Проиндексировать изменения.
9. Сделать второй коммит.
10. Продемонстрировать преподавателю ход работы, ответить на уточняющие вопросы.

## Лабораторная работа № 2 «Игнорирование, сравнение, удаление и

# перемещение файлов»

*Тема: Игнорирование файлов. Сравнение изменений. Удаление и перемещение файлов.*

*Цель работы: научиться исключать файлы, которые нет необходимости вести в системе контроля версий. Получить практические навыки сравнения проделанных изменений в файлах.*

Зачастую, имеется группа файлов, которые не только нет необходимости автоматически добавлять в репозиторий, но и видеть в списках неотслеживаемых. К таким файлам обычно относятся автоматически генерируемые файлы (различные логи, результаты сборки программ и т.п.). В таком случае, необходимо создать файл *.gitignore* с перечислением шаблонов соответствующих таким файлам.

К шаблонам в файле .gitignore применяются следующие правила:

* Пустые строки, а также строки, начинающиеся с # (символ комментария), игнорируются.
* Можно использовать стандартные glob шаблоны.
* Можно заканчивать шаблон символом слэша (/) для указания каталога.
* Можно инвертировать шаблон, использовав восклицательный знак (!) в качестве первого символа.

Glob-шаблоны представляют собой упрощённые регулярные выражения используемые командными интерпретаторами. Символ \* соответствует 0 или более символам; последовательность [abc] — любому символу из указанных в скобках (в данном примере a, b или c); знак вопроса (?) соответствует одному символу; [0-9] соответствует любому символу из интервала (в данном случае от 0 до 9).

**Порядок выполнения работы**

1. Изучить теоретическую часть работы.
2. Продолжить работу с созданным репозиторием на первой лабораторной работе.
3. Создать папку temp в своем репозитории.
4. Создать папку log и добавить в нее 2 файла: main.html и some.tmp.
5. Создать файл .gitignore и добавить в игнорирование папку temp и файлы с расширением .tmp из папки log.
6. Закоммитить добавление файла .gitignore.
7. Внести изменения в файл my\_first\_file.txt, добавив строчку “row to index”, проиндексировать данные изменения. Еще раз внести изменения в файл, добавив строчку “row no index”.
8. Посмотреть индексированные и неиндексированные изменения используя команду **git diff**.
9. Удалить файл my\_first\_file.txt, зафиксировать данное удаление.
10. Переименовать файл my\_second\_file.txt в my\_first\_file.txt, зафиксировать изменение.
11. Продемонстрировать преподавателю ход работы, ответить на уточняющие вопросы.

## Лабораторная работа №3 «Просмотр истории коммитов»

*Тема: Просмотр истории коммитов, команда git log.*

*Цель работы: освоить механизм работы с командой git log для получения информации об истории коммитов.*

После того как будет создано несколько коммитов, вероятнее всего появится необходимость просмотреть, что же происходило с этим репозиторием. Наиболее простой и в то же время мощный инструмент для этого — команда **git log**.

По умолчанию, без аргументов, **git log** выводит список коммитов созданных в данном репозитории в обратном хронологическом порядке. То есть самые последние коммиты показываются первыми.

Один из наиболее полезных параметров — это **-p**, который показывает дельту (разницу/diff), привнесенную каждым коммитом. Также можно использовать **-2**, что ограничит вывод до 2-х последних записей.

**Порядок выполнения работы**

1. Изучить теоретическую часть работы.
2. Продолжить работу с созданным репозиторием.
3. Изучить возможности команды **git log**, выполнить различные варианты вывода информации и ее отбора.
4. Выполнить задание согласно варианту.
5. Продемонстрировать преподавателю ход работы, ответить на уточняющие вопросы.

**Варианты**

|  |  |
| --- | --- |
| **Номер** | **Задание** |
| 1 | Вывести коммиты, автором которых являетесь Вы, за последний месяц. |
| 2 | Вывести все коммиты в формате: короткий хеш, автор, комментарий. |
| 3 | Вывести все коммиты, в сообщении которых присутствует слово my. |
| 4 | |  |  | | --- | --- | | Вывести все коммиты за текущий месяц с информацией о том, какие | | | файлы были изменены. |  | |
| 5 | Вывести информацию о первом коммите в системе, с выводом дельты |
|  | |  | | --- | | (diff). | |
| 6 | |  |  | | --- | --- | | Вывести коммиты сделанные за последний месяц назад, но исключая | | | последнюю неделю. |  | |
| 7 | |  |  | | --- | --- | | Вывести информацию о коммитах в формате: автор, дата, список изме- | | | ненных файлов. |  | |
| 8 | |  |  | | --- | --- | | Вывести коммиты, автором которых являетесь Вы, с выводом дельты | | | (diff). |  | |
| 9 | |  |  | | --- | --- | | Вывести коммиты, в которых происходили изменения файла | | | my\_first\_file.txt, за последние 2 недели. |  | |
| 10 | |  | | --- | | Вывести последние 3 коммита в формате: автор, комментарий. | |
| 11 | |  |  | | --- | --- | | Вывести информацию о первом коммите в формате: дата, автор, ком- | | | ментарий, а также список измененных файлов. |  | |
| 12 | |  |  | | --- | --- | | Вывести коммиты, автором которых являетесь Вы, со списком изменен- | | | ных файлов. |  | |
| 13 | |  |  | | --- | --- | | Вывести все коммиты за текущий месяц в формате: сокращенный хеш, | | | дата, комментарий. |  | |
| 14 | |  |  | | --- | --- | | Вывести все коммиты в формате: e-mail автора, дата коммита, хеши ро- | | | дительских коммитов. |  | |
| 15 | |  |  | | --- | --- | | Вывести коммиты, в которых происходили изменения файла | | | my\_first\_file.txt. |  | |

## Лабораторная работа №4 «Отмена изменений. Работа с метками»

*Тема: Отмена внесенных изменений. Работа с метками.*

*Цель работы: научиться отменять сделанные изменения, работать с метками.*

Одна из типичных отмен происходит тогда, когда коммит сделан слишком рано, например, не были добавлены какие-либо файлы, или перепутан комментарий к коммиту. Если необходимо сделать этот коммит ещё раз, можно выполнить **git commit** с опцией **–amend.**

Эта команда берёт индекс и использует его для коммита. Если после последнего коммита не было никаких изменений (например, приведенная команда была запущена сразу после предыдущего коммита), то состояние проекта будет абсолютно таким же и всё, что изменится, это комментарий к коммиту.

Git использует два основных типа меток: легковесные и аннотированные. Легковесная метка — это что-то весьма похожее на ветку, которая не меняется — это

|  |  |
| --- | --- |
| просто указатель на определённый коммит. А вот аннотированные метки хранятся в | |
| базе данных Git'а как полноценные объекты. Они имеют контрольную сумму, со- | |
| держат имя поставившего метку, e-mail и дату, имеют комментарий и могут быть | |
| подписаны и проверены с помощью GNU Privacy Guard (GPG). Обычно рекоменду- | |
| ется создавать аннотированные метки, чтобы иметь всю перечисленную информа- | |
| цию; но если необходимо сделать временную метку или по какой-то причине нет | |
| необходимости сохранять остальную информацию, то для этого годятся и легковес- | |
| ные метки. |  |

**Порядок выполнения работы**

1. Изучить теоретическую часть работы.
2. Продолжить работу с созданным репозиторием.
3. Создать три файла: 1.txt, 2.txt, 3.txt.
4. Проиндексировать первый файл и сделать коммит с комментарием “add

1.txt file”.

1. Проиндексировать второй и третий файлы.
2. Удалить из индекса второй файл.
3. Перезаписать уже сделанный коммит с новым комментарием “add 1.txt

and 3.txt”

1. Создать аннотированную метку с названием v0.01.
2. Создать легковесную ветку указывающую на первый коммит в репозитории.
3. Продемонстрировать преподавателю ход работы, ответить на уточняющие вопросы.

## Лабораторная работа №5 «Ветвление. Конфликты»

*Тема: Работа с ветками, решение конфликтов.*

*Цель работы: научиться создавать ветки, перемещаться по ним, объединять и удалять их. Решать конфликты слияния.*

Ветка в git'е — это просто легковесный подвижный указатель на один из

|  |  |  |
| --- | --- | --- |
| коммитов. Ветка по умолчанию в git'е называется | **master** | . Когда происходит созда- |
| ние коммита на начальном этапе, доступна ветка | **master** | , указывающая на послед- |

ний сделанный коммит. При каждом новом коммите она сдвигается вперёд автоматически. Для того чтобы создать новую ветку используется команда **git branch**.

Эта команда создаст новый указатель на тот самый коммит, на котором сейчас находится git.

**Порядок выполнения работы**

1. Изучить теоретическую часть работы.
2. Продолжить работу с созданным репозиторием.
3. Создать новую ветку my\_first\_branch.
4. Перейти на ветку и создать новый файл in\_branch.txt, закоммитить изменения.
5. Вернуться на ветку master.
6. Создать и сразу перейти на ветку new\_branch.
7. Сделать изменения в файле 1.txt, добавить строчку “new row in 1.txt file”, закоммитить изменения.
8. Перейти на ветку master и слить ветки master и my\_first\_branch, после чего слить ветки master и new\_branch.
9. Удалить ветки my\_first\_branch и new\_branch. 10. Создать ветки branch\_1 и branch\_2.
10. Перейти на ветку branch\_1 и изменить файл 1.txt, удалить все содержимое и добавить текст “fix in 1.txt”, изменить файл 3.txt, удалить все содержимое и добавить текст “fix in 3.txt”, закоммитить изменения.
11. Перейти на ветку branch\_2 и также изменить файл 1.txt, удалить все содержимое и добавить текст “My fix in 1.txt”, изменить файл 3.txt, удалить все содержимое и добавить текст “My fix in 3.txt”, закоммитить изменения.
12. Слить изменения ветки branch\_2 в ветку branch\_1.
13. Решить конфликт файла 1.txt в ручном режиме, а конфликт 3.txt используя команду git mergetool c утилитой Meld.
14. Продемонстрировать преподавателю ход работы, ответить на уточняющие вопросы.

## Лабораторная работа №6 «Прятанье»

*Тема: Механизм прятанья.*

*Цель работы: научиться использовать механизм прятанья, а также расширить знания в управлении веток.*

|  |  |  |
| --- | --- | --- |
|  | Часто возникает такая ситуация, что пока идет работа над частью своего про- | |
| екта, всё находится в беспорядочном состоянии, а нужно переключить ветки, чтобы | | |
| немного поработать над чем-то другим. Проблема в том, что делать коммит с напо- | | |
| ловину доделанной работой только для того, чтобы позже можно было вернуться в | | |
| это же состояние не хотелось бы. Ответ на эту проблему — команда **git stash**. | |  |

Прятанье поглощает грязное состояние рабочего каталога, то есть изменённые отслеживаемые файлы и изменения в индексе, и сохраняет их в стек незавершённых изменений, которые потом в любое время можно снова применить.

**Порядок выполнения работы**

1. Изучить теоретическую часть работы.
2. Продолжить работу с созданным репозиторием.
3. Проверить какие ветки слиты с веткой master, а какие нет.
4. Удалить все ветки слитые с master.
5. Создать новую ветку work и перейти в нее.
6. Изменить файл 1.txt.
7. Спрятать данные изменения.
8. Развернуть обратно данные изменения с опцией --index.
9. Развернуть спрятанные изменения в новую ветку.
10. Продемонстрировать преподавателю ход работы, ответить на уточняющие вопросы.

## Лабораторная работа №7 «Работа с удаленным репозиторием»

*Тема: Работа с удаленным репозиторием. Github.com.*

*Цель работы: научиться работать с удаленным репозиторием, использовать платформу github.com.*

Если необходимо получить копию существующего репозитория Git, например, проекта, в котором разработчик планирует поучаствовать, то необходимо использовать команду **git clone**. Каждая версия каждого файла из истории проекта забирается (*pulled*) с сервера, когда выполняется команда **git clone**. Фактически, если серверный диск выйдет из строя, можно использовать любой из клонов на любом из клиентов, для того чтобы вернуть сервер в то состояние, в котором он находился в момент клонирования.

Клонирование репозитория осуществляется командой **git clone [url]**.

**Порядок выполнения практической работы**

1. Изучить теоретическую часть работы.
2. Продолжить работу с созданным репозиторием.
3. Пройти регистрацию на сайте github.com.
4. Настроить доступ к github по SSH.
5. Создать репозиторий на github.com.
6. Синхронизировать локальный репизиторий с удаленным.
7. Продемонстрировать преподавателю ход работы, ответить на уточняющие вопросы.

## Методические указания по выполнению самостоятельной работы

Самостоятельная работа предусмотрена учебным планом. Цель самостоятельной работы студента в рамках курса — закрепление и расширение знаний, полученных во время проведения аудиторных занятий.

Согласно рабочей программе в качестве самостоятельной работы для студентов являются следующие виды:

* проработка лекционного материала;
* подготовка к лабораторным работам; - выполнение контрольных работ; - выполнение индивидуального задания.

Проработка лекционного материала осуществляется студентом с использованием конспекта лекций и рекомендуемых учебников.

Для подготовки к лабораторным работам студент должен изучить теоретический материал согласно теме лабораторной работы, подготовиться к практической части реализации за компьютером.