Лабораторная работа № 2.

Знакомство с системой управления версиями git

Цель работы: познакомить студентов со структурой локального репозитория git и основными операциями, доступными в нем. Работа выполняется в режиме терминала. После выполнения очередного действия следует проверять состояние (статус) репозитория.

Задание

1. Войти в систему со своим логином/паролем и открыть эмулятор терминала.

2. Создать в домашнем каталоге каталог localrepo и в нем два файла — first.txt и second.txt. Просмотреть и зафиксировать детальную информацию о каталоге localrepo (сделать копию экрана или вывести в файл и сохранить для отчета).

3. Перейти в каталог localrepo. Инициализировать в каталоге localrepo репозиторий git (команда git init). Просмотреть и зафиксировать детальную информацию о каталоге localrepo. **Как изменился каталог после создания репозитория?** Изучить изменения — вывести на экран содержимое каталога .git, проверить статус репозитория (git status).

4. Добавить (команда git add) оба файла для сохранения в репозитории в следующем коммите. Проверить статус репозитория. Найти в каталоге .git файл index (в .git не переходить!). Зафиксировать время сохранения файла index.

5. Зафиксировать (сохранить) изменения в репозитории (git commit). Вывести на экран информацию о коммите (git show). Проверить статус репозитория. Проверить время сохранения файла index и сравнить его с предыдущим значением. **Сделайте вывод о роли файла index при выполнении фиксации изменений в репозитории и почему команда git add называется «добавление в индекс».**

6. Создать новый файл third.txt и зафиксировать его в репозитории. Отредактировать third.txt и зафиксировать в репозитории без предварительной индексации (команда commit с ключом -a).

7. Внести изменения в каждый из трех файлов. Проверить статус репозитория.

8. Для первого файла отменить последние изменения (git checkout). Второй и третий файлы проиндексировать. Проверить статус.

9. Отказаться от сохранения изменений второго файла (git reset HEAD), а изменения в третьем — зафиксировать в репозитории. Проверить статус. Выполнить команду, которая **уберет second.txt из статуса** (см. п. 8).

10. Вернуть последний коммит для внесения дополнительных изменений: внести изменения в файл, добавить их в индекс, затем выполнить команду git commit - -amend, при этом изменить текст комментария. Проверить статус и убедиться, что предыдущая редакция коммита отменена.

11. Просмотреть историю изменений репозитория (git log) — подробную и краткую.

12. Просмотреть историю изменений (логи) отдельного (любого) файла. Сравнить полученный результат с результатом, полученным для всего репозитория.

13. Изменить первый и второй файл, первый добавить в индекс. Для каждого файла сравнить текущую версию (с изменениями) и последнюю зафиксированную версию, для этого подобрать вариант команды git diff, соответствующий состоянию файла.

14. Сохранить все изменения в репозитории.

15. Переименовать файл first.txt средствами операционной системы (команда mv) — f1.txt, second.txt – средствами репозитория (git mv)— f2.txt.

16. Сделать копию третьего файла средствами операционной системы — f3.txt (команда cp).

17. Создать файл fourth.txt и добавить его в индекс.

18. Создать файл fifth.txt (можно пустой), но в индекс не добавлять.

19. Вывести на экран статус репозитория а) подробный, б) краткий. Изучить обозначения, используемые краткой формой представления статуса репозитория. Определить, какие изменения надо добавить в индекс для последующей фиксации, индексировать их и зафиксировать в репозитории.

20. Настроить игнорирование некоторых файлов, для этого создать в рабочем каталоге файл .gitignore и добавить в него шаблоны (маски) для имен файлов, которые репозиторий не должен замечать (они не будут отображаться в статусе). Задать три варианта шаблона имен файлов: 1) с символом *, 2) с [ ], 3) с ? (знак вопроса).

21. Создать 3 файла с именами, удовлетворяющими этим шаблонам. Объяснить, почему эти имена соответствуют конкретному шаблону. Проверить статус репозитория и оценить результаты игнорирования файлов.

22. Проверить статус репозитория; если есть незафиксированные изменения, зафиксировать.

23. Удалить файл f2.txt средствами файловой системы. Проверить статус. Восстановить файл из репозитория.

24. Удалить файл third.txt командой git rm. Проверить статус. Зафиксировать удаление файла.

25. Отредактировать и добавить в индекс файл fifth.txt. Затем отредактировать его еще раз и проверить статус. **Какие изменения будут зафиксированы последующим коммитом? Что надо сделать, чтобы все изменения были зафиксированы?**

26. Отредактировать файл fifth.txt еще раз, добавить в индекс и зафиксировать, а затем исключить файл из списка отслеживаемых репозиторием. После выполнения операции файл должен остаться в каталоге!

27. Изменить файл fourth.txt, добавить в индекс, а затем удалить файл, несмотря на незафиксированные изменения — для этого подобрать подходящий вариант команды git rm.

28. Подготовить отчет на основе полученных результатов и ответить на вопросы.

Справочный материал

Системы управления версиями

Система управления версиями (система контроля версий) — это программный продукт, предназначенный для сохранения истории изменений файлов с целью последующего возврата в случае необходимости к более ранней версии. По структуре системы управления версиями делятся на локальные (Рис. 1), централизованные (Рис. 2) и децентрализованные (распределенные, Рис. 3).

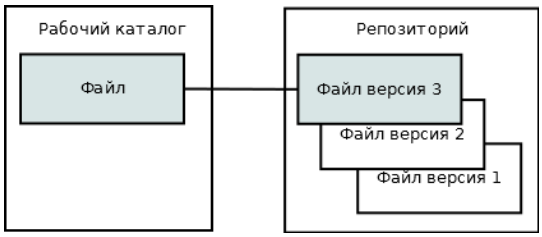

Рис. 1 Локальные системы управления версиями

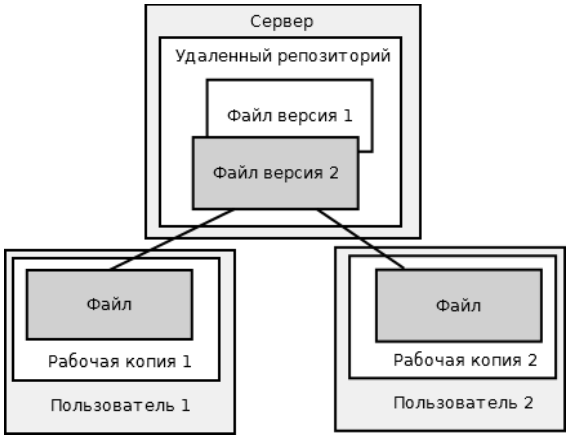

Рис. 2 Централизованные системы управления версиями

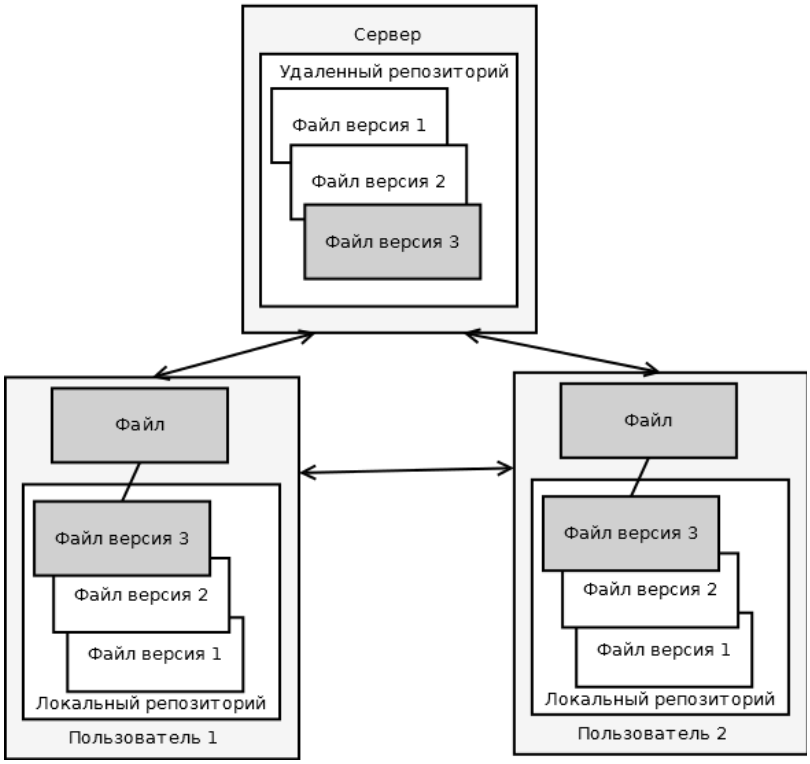

Рис. 3 Децентрализованные системы управления версиями

Краткая история git

Система git появилась в 2005 году как инструмент для разработки операционной системы Linux. Разработчики git ставили перед собой цель создать децентрализованную систему управления версиями со следующими характеристиками: быстродействие, простое проектное решение, мощная поддержка нелинейной разработки (большое количество ветвлений), полностью распределенная система, возможность эффективной работы с большими проектами и объемами данных.

На основе git созданы сервисы для хранения репозиториев: GitHub – наиболее популярный сервис, сервисы для локальных сетей GitLab, Gitea.

Команды git

git help – справочная информация о git.

git help <команда> – справочная информация о команде.

git init <каталог>- инициализация репозитория git в указанном каталоге.

git status – просмотр текущего состояния репозитория: есть изменения, изменения добавлены в индекс (для фиксации в репозитории), все изменения сохранены.

Просмотр состояния репозитория в краткой форме: git status -s.

Обозначения, используемые краткой формой git status:

A – новый файл добавлен в индекс,

M – файл изменен,

D – файл удален,

R – файл переименован,

C – файл скопирован,

? - файл не отслеживается.

git show – команда выводит на экран информацию о различных объектах git; по умолчанию выводит информацию о последнем коммите.

git show <хеш-код коммита> - просмотр коммита, сделанного ранее. Хеш-код — это число, вычисляемое по специальному алгоритму для каждого коммита и состоящее из сорока шестнадцатеричных чисел; используется для обращения к коммиту.

git add <файл и ли каталог>– добавление изменений в индекс; изменения, добавленные в индекс, будут сохранены в репозитории последующим коммитом.

git add file1 file2 file3 – добавление нескольких изменений в индекс.

git commit – сохранение изменений, добавленных в индекс, в репозитории; в процессе выполнения команды автоматически будет открыт текстовый редактор (установленный в настройках клиента git по умолчанию), и надо будет ввести комментарий к коммиту и сохранить его.

git commit -m “комментарий” - сохранение в репозитории изменений и комментария к ним.

git commit -a - фиксация в репозитории неиндексированного файла, отслеживаемого репозиторием.

git commit - -amend – отмена последнего коммита и внесение в него изменений; команда применяется для внесения изменений в комментарий и фиксации дополнительных изменений в файле. Надо помнить, что совсем отменить коммит (т. е. удалить) нельзя!

git checkout <файл> - отменить изменение файла, не добавленного в индекс.

git reset HEAD <файл> - отменить добавление файла в индекс.

git log <файл> - просмотр истории изменений указанного файла или каталога. Команда имеет опции для настройки вывода git log - -pretty =”строка_формата”. В строке формата указываются параметры вывода:

%h – сокращенный хеш-код коммита,

%s – строка комментария,

%an -имя автора,

%ae – электронная почта автора,

%cn – имя создателя версии,

%ce – электронная почта создателя версии,

%cd – дата создания версии,

%cr – дата создания версии в относительном формате.

Примеры команды git log

git log - -pretty=”%h %s” - вывод сокращенного хеша и комментария,

git log - -pretty=short – краткая информация о коммите.

git mv <путь/исходный_файл> <путь/конечный_файл>– перемещение или переименование файла средствами репозитория.

git mv ./file1 ./file2 – переименование файла, находящегося в текущем каталоге.

git mv ./file1 ./subdir/ - перемещение файла в подкаталог с сохранением имени.

git rm <путь/удаляемый_файл> - удаление файла. Следующим коммитом файл будет удален как из отслеживания репозиторием, так и из каталога. Если файл следует сохранить, но отменить его отслеживание, то следует выполнить команду git rm - -cached <путь/удаляемый_файл>. Если файл пред удалением был изменен и проиндексирован, то выполняется команда безусловного удаления git rm -f <путь/удаляемый_файл>.

git diff <путь/файл> - просмотр различий в текущей версии файла по сравнению с последней зафиксированной. В таком виде команда покажет изменения для неиндексированного файла. Если файл проиндексирован, то увидеть изменения можно командой git diff - -staged <путь/файл>. При выводе результата зафиксированный файл обозначается как a, а измененный как b.

Игнорирование файлов

Под игнорированием понимают способность репозитория не обращать внимания на некоторые файлы, например, файлы, являющиеся результатом обработки других файлов, т. е. файлы, которые могут быть удалены, а затем созданы заново. В разработке программного обеспечения к таким файлам относятся результаты компиляции — файлы типа *.o, *.obj и т. д. Игнорируемые файлы не отслеживаются репозиторием и не отображаются в статусе.

Для настройки репозитория на игнорирование надо создать в рабочем каталоге файл .gitignore и поместить в него список игнорируемых файлов. В списке могут быть как имена файлов, так и шаблоны имен (маски).

Примеры шаблонов имен файлов:

*.a – игнорировать все файлы с расширением a,

!abcd.a – исключить из списка игнорирования файл с именем abcd.a,

*.[abc] – игнорировать файлы с расширением a, b, c,

a?.?? - игнорировать файлы с именами, состоящими из двух символов, первый из которых — a, и расширением из двух символов.

текст — комментарий в файле .gitignore.

Вопросы к защите

1. Что такое подкаталог .git в рабочем каталоге?

2. Что делает команда git add?

3. Что такое индексированный файл в git?

4. В файл внесены изменения, он добавлен в индекс. После этого снова были внесены изменения. Надо ли файл добавлять в индекс еще раз?

5. Какие изменения автоматически добавляются в коммит (без предварительного индексирования)?

6. Как отменить удаление файла, если файл отслеживается репозиторием и удален а) средствами файловой системы, б) средствами репозитория, но удаление еще не зафиксировано?

7. Некоторый файл был изменён, затем проиндексирован и снова изменен. Как сравнить его текст с последней зафиксированной версией, если в нем есть как индексированные, так и неиндексированные изменения?