Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технолиогий**

**Лабораторная работа №1**

Выполнил:

Студент 2 курса 3 группы ФИТ

Авхачёв Евгений Сергеевич

**2020 г.**

**Ответы на вопросы:**

1. Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии.
2. Она позволяет вернуть файлы к состоянию, в котором они были до изменений, вернуть проект к исходному состоянию, увидеть изменения, увидеть, кто последний менял что-то и вызвал проблему, кто поставил задачу и когда и многое другое.
3. Основное отличие Git от любой другой СКВ (включая Subversion и её собратьев) — это подход к работе со своими данными. Концептуально, большинство других систем хранят информацию в виде списка изменений в файлах. Эти системы (CVS, Subversion, Perforce, Bazaar и т.д.) представляют хранимую информацию в виде набора файлов и изменений, сделанных в каждом файле, по времени.
4. У Git есть три основных состояния, в которых могут находиться ваши файлы: зафиксированное (committed), изменённое (modified) и подготовленное (staged).
5. Чтобы проиндексировать его, необходимо выполнить команду git add. Это многофункциональная команда, она используется для добавления под версионный контроль новых файлов, для индексации изменений, а также для других целей, например для указания файлов с исправленным конфликтом слияния.
6. Фиксация файла обычно производится с помощью команды git-commit.
7. Чтобы посмотреть все установленные настройки и узнать где именно они заданы, нужно использовать команду git config –list –show-origin.
8. Когда вы инициализируете репозиторий командой git init, Git создаёт ветку с именем master по умолчанию.
9. Чтобы начать слежение за текущим проектом, нужно перейти в папку этого проекта и ввести команду git init.

Пока контроль ваших файлов отсутствует. Чтобы начать управление версиями существующих файлов (в противовес пустому каталогу), укажите файлы, за которыми должна следить система, и выполните первую фиксацию изменений. Для этого потребуется несколько команд git add, добавляющих файлы, за которыми вы хотите следить, а затем команда git commit.

1. Фиксация изменений производиться с помощью команды git commit.
2. Для определения состояния файлов следует использовать команду git status.
3. Зачастую, у вас имеется группа файлов, которые вы не только не хотите автоматически добавлять в репозиторий, но и видеть в списках неотслеживаемых. К таким файлам обычно относятся автоматически генерируемые. В таком случае, вы можете создать файл .gitignore с перечислением шаблонов соответствующих таким файлам.
4. git diff сама по себе не показывает все изменения сделанные с последнего коммита — только те, что ещё не проиндексированы.
5. Команда git commit берёт все данные, добавленные в индекс с помощью git add, и сохраняет их слепок во внутренней базе данных, а затем сдвигает указатель текущей ветки на этот слепок.
6. Команда git log  используется для просмотра истории коммитов, начиная с самого свежего и уходя к истокам проекта. По умолчанию, она показывает лишь историю текущей ветки, но может быть настроена на вывод истории других, даже нескольких сразу, веток. Также её можно использовать для просмотра различий между ветками на уровне коммитов.
7. Практически во всех главах книги эта команда используется для демонстрации истории проекта.

Использование опций -p и --stat для получения представления об изменениях в каждом коммите, а также -pretty и --oneline для настройки формата вывода этой команды — более полным и подробным или кратким.

1. Для того, чтобы просмотреть список настроенных удалённых репозиториев, вы можете запустить команду git remote.
2. Для извлечения данных из удалённого репозитория, нужно использовать команду git clone.
3. Для того, чтобы отправить данные в удалённый репозиторий, нужно импользовать команду git push <remote-name> <branch-name>.
4. Команда, позволяющая отобрать удалённые репозитории git remote show origin.
5. Команда git fetch забирает данные в ваш локальный репозиторий, но не сливает их с какими-либо вашими наработками и не модифицирует то, над чем вы работаете в данный момент. Git pull, как правило, извлекает (fetch) данные с сервера, с которого вы изначально склонировали, и автоматически пытается слить (merge) их с кодом, над которым вы в данный момент работаете.
6. git pull — это, по сути, **команда** git **fetch**, после которой сразу же следует git merge. git **fetch** получает изменения с сервера и сохраняет их в каталог.
7. Git Merge и Git Rebase преследуют одну и ту же цель. Они предназначены для интеграции изменений из одной ветки в другую. Слияние — обычная практика для разработчиков, использующих системы контроля версий. Независимо от того, созданы ли ветки для тестирования, исправления ошибок или по другим причинам, слияние фиксирует изменения в другом месте. Rebase — еще один способ перенести изменения из одной ветки в другую. Rebase сжимает все изменения в один «патч». Затем он интегрирует патч в целевую ветку.  
   В отличие от слияния, перемещение перезаписывает историю, потому что она передает завершенную работу из одной ветки в другую. В процессе устраняется нежелательная история.
8. Запрос на принятие изменений (Pull Request) откроет новую ветвь с обсуждением отправляемого кода, и автор оригинального проекта, а так же другие его участники, могут принимать участие в обсуждении предлагаемых изменений до тех пор, пока автор проекта не будет ими доволен, после чего автор проекта может добавить предлагаемые изменения в проект.
9. Git log - команда позволяющая отобразить историю репозитория.
10. Для того, чтобы создать новую ветку, нужно воспользоваться командой git branch «name». Используя ветвление, Вы отклоняетесь от основной линии разработки и продолжаете работу независимо от неё, не вмешиваясь в основную линию. **HEAD** — это **указатель** на текущий коммит.
11. Команда git checkout используется для переключения между ветками и коммитами.
12. Для того, чтобы добавить изменения из одной ветки в другую, нужно перейти в основную ветку и применить команду git merge «name branch».
13. При слиянии веток, может произойти конфликт слияния (merge conflict) и его нужно будет как-то разрешить. Чтобы решить проблему, нужно либо исправить файлы вручную или выбрать одну из версий этого файла.
14. GitLab — это онлайн-сервис, предназначенный для работы с **git-репозиториями**.