Лабораторная работа №1. Работа с Git

Выполнила студентка группы НФИбд-03-19 Попова Юлия Дмитриевна 1032192876

Цель работы: познакомиться с основными возможностями Git и научиться работать с Markdown.

Команды Git, с которыми мы работали: * git init – создать git репозиторий из пустого каталога. * git status – проверить текущее состояние репозитория. * git add – проиндексировать изменения. Git теперь знает об изменении, но они пока не записаны в репозиторий. * git commit -m – сделать коммит, метка -m необходима, если нужно оставь комментарии в командной строке. * git log – вывести список изменений. * checkout – команда копирует снимок из репозитория в рабочий каталог. Необходима для возвращения назад в историю. * git tag v1 – создать тег первой версии. * git tag – проверить доступные теги. * git reset – отменить проиндексированные изменения перед коммитом. Команда сбрасывает буферную зону к HEAD, это очищает буферную зону от изменений, которые были проиндексированы. * git revert HEAD – создать новый коммит, который удаляет изменения, сохраненные предыдущем некорректным коммитом (предыдущим коммит отменяется). * git reset –hard v1 – команда, где параметр –hard указывает, что рабочий каталог должен быть обновлен в соответствии с новым HEAD ветки. * git tag -d oops – сборщик мусора, удаляющий теги и коммиты, на которые ссылается. * git commit –amend – изменить предыдущий коммит. * git mv – перемещать файлы. Команда удаляет файл из начального каталога, создает этот файл в новом каталоге, и файлы сразу проиндексированы и готовы к коммиту. * dir -C .git – посмотреть всю информацию git, которая расположена в каталоге .git * git cat-file – вывести определенные коммиты или каталоги по их хэшу. * git checkout -b style – создать новую ветку, где style – название новой ветки. * git merge – объединить (слить) изменения из двух веток в одну. * git rebase – при возникновении конфликтов в изменениях, разрешали конфликты вручную. Использовали команду git rebase. Результат выполнения похож на результат выполнения команды git merge слияния, но отличается дерево коммитов. * git clone – создать клон репозитория. * git branch -a – посмотреть ветки в репозитории, для отображения удаленных веток нужна метка -a. * git fetch – извлечь новые коммиты из удаленного репозитория, команда не сливает их с наработками в локальных ветках. * git pull – извлечь новые коммиты из удаленного репозитория и объединить (слить) изменения. * git branch –track – отследить удаленные ветки, возможно добавить удаленную ветку. * git clone –bare – создать чистый репозиторий (репозиторий, в котором нет рабочих каталогов).

- git remote add shared – добавить репозиторий к оригинальному репозиторию.
- git push shared master – отправить изменения в общий репозиторий.

Вывод: познакомились с основными возможностями Git, научились работать с Markdown.