Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

**ПОЛЬЗОВАТЕЛЬСКИЕ ИНТЕРФЕЙСЫ ИНФОРМАЦИОННЫХ СИСТЕМ**

ОТЧЕТ

по практической работе

**«ОСНОВЫ GIT»**

Выполнила: ст. гр. 210902 Хоружик Е. В.

Проверил: Давыдович К. И.

Минск 2024

Git — это система контроля версий, которая позволяет отслеживать изменения в вашем коде с течением времени.

Репозиторий Git — это хранилище, в котором расположен ваш проект и его история. Это может быть локальное хранилище где-то на вашем компьютере или удаленное хранилище на сервисе типа GitHub или другом хостинге в Интернете. Репозиторий служит для отслеживания изменений в проекте, координации работы между несколькими людьми и отслеживания истории проекта.

Коммит содержит только информацию об изменениях, которые были внесены в репозиторий с момента последнего коммита. Он не содержит все файлы репозитория (если только это не первый коммит).

Ветка — это параллельная версия репозитория. Ветки позволяют вам работать над отдельными функциями вашего проекта, не влияя на основную версию. Закончив работу над новой фичей, вы можете объединить эту ветку с основной версией проекта.

Таблица 1 – Команды для подготовки к работе с Git

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git config --global user.name "*Your Name*" | Устанавливает имя для подписи изменений |
| git config --global user.email "*your\_email@whatever.com*" | Устанавливает электронную почту для подписи изменений |
| git config --global init.defaultBranch *main* | Устанавливает имя ветки по умолчанию (в данном случае main) |
| git config --global core.autocrlf true  git config --global core.safecrlf warn | Корректная обработка окончаний строк |

Таблица 2 – Команды для работы с папками и файлами

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| mkdir *folder\_name* | Создать папку с указанным именем |
| cd *folder\_name* | Переход в папку с указанным именем |
| touch *file\_name* | Создать файл с указанным именем |
| git mv *file\_name folder\_name /file\_name* | Перемещение файла |
| cd .. | Переход в родительскую директорию |
| pwd | Отображает полный путь к текущей директории |
| ls | Выводит список всех файлов и директорий в текущей директории |

Таблица 3 – Команды для создания локального репозитория

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git init | Создание Git-репозитория |

Таблица 4 – Проверка состояния

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git status | Проверка текущего состояния репозитория |

Таблица 5 – Внесение изменений

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git add *file\_name* | Добавление файла в область подготовленных файлов (изменения проиндексированы) |
| git add | Добавление всех изменений |
| git commit -m "*Initial Commit*" | Создание коммита с комментарием |
| git commit | Создание коммита с открытием редактора |

Таблица 6 – История

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git log | Получение списка произведенных изменений |
| git log *file\_name* | История изменений конкретного файла |
| git log --pretty=oneline | Однострочный формат |
| git log --oneline --max-count=2  git log --oneline --since="5 minutes ago"  git log --oneline --until="5 minutes ago"  git log --oneline --author="Your Name"  git log --oneline --all | Варианты просмотра истории |
| git log --pretty=format:"%h %ad | %s%d [%an]" --date=short | Конечный формат лога |
| git config --global format.pretty '%h %ad | %s%d [%an]'  git config --global log.date short | Настроить формат вывода истории по умолчанию |
| git show *tag* | Просмотр изменений в конкретном коммите (можно использовать любую ссылку на коммит, например, метку HEAD, хеш коммита, имя ветки или тега и т.д.) |
| git log --follow *folder\_name /file\_name* | Просмотр истории файла |
| git log --all --graph | Просмотр веток и их расхождения |

Таблица 7 – Получение старых версий

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git checkout <*hash*> | Обновит вашу рабочую директорию до любого предыдущего коммита |
| git checkout <*tag*> | Обновит вашу рабочую директорию до любого предыдущего коммита с указанным тегом |
| git checkout <*tag^*>  git checkout <*tag~1*> | Обновит вашу рабочую директорию до предыдущей версии |

Таблица 8 – Теги версий

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git tag *tag* | Создание тега |
| git tag | Просмотр доступных тегов |
| git log main --all | Просмотр тегов в логе |
| git tag -d *tag* | Удаление тега |

Таблица 9 – Отмена изменений

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git checkout *file\_name* | Отмена локальных изменений (до индексации) |
| git reset HEAD *file\_name* | Отмена проиндексированных изменений (перед коммитом) (сбрасывает область подготовки к HEAD) |
| git revert HEAD | Отмена коммита |
| git reset --hard *tag* | Сброс и удаление коммита (Параметр --hard указывает, что рабочая директория должна быть приведена к тому состоянию, которое соответствует HEAD-коммиту ветки) |

Таблица 10 – Изменение коммита

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git commit --amend -m "*Commit*" | Изменение предыдущего коммита |

Таблица 11 – Ветки

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git switch -c *branch*  (git checkout -b *branch*) | Создание ветки |
| git switch *branch* | Переключение на ветку с указанным именем |
| git branch | Список веток |
| git branch -a | Список удаленных веток |

Таблица 12 – Слияние

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git merge main | Слияние текущей ветки с веткой main |
| git merge --abort | Отмена слияния |

Таблица 13 – Перебазирование

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git rebase main | Перебазирование текущей ветки на ветку main |
| git rebase --continue | Позволяет продолжить процесс rebase после разрешения конфликтов |

Используйте команду rebase:

* Когда вы получаете изменения из удаленного репозитория и хотите применить их к своей локальной ветке.
* Если вы хотите, чтобы история коммитов была линейной и легко читаемой.

Не используйте команду rebase:

* Если текущая ветка является публичной и общей. Переписывание таких веток будет мешать работе других членов команды.
* Если важна точная история ветки коммитов (поскольку команда rebase переписывает историю коммитов).

Учитывая приведенные выше рекомендации, я предпочитаю использовать команду rebase для краткосрочных, локальных веток и команду merge для веток в публичном репозитории.

Таблица 14 – Клонирование репозитория

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git clone *work home* | Создание клона репозитория (оригинальный репозиторий work и клонированный репозиторий home) |

Таблица 15 – Удаленные репозитории

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git remote | Имя удаленного репозитория |
| git remote show *origin* | Отображение информации о удаленном репозитории под названием origin |
| git push *shared main* | Отправка изменений в общий репозиторий |

Таблица 16 – Подтягивание изменений

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git fetch | Подтягивает новые коммиты из удаленного репозитория, но не сливает их с вашими наработками в локальных ветках |
| git merge *origin*/*main* | Слияние подтянутых изменений в локальную ветку |
| git pull | Подтягивает и сливает изменения из удаленной ветки в текущую одним вызовом |

Таблица 17 – Удаленные ветки

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git branch --track style *origin*/*style* | Создание новой локальной ветки и настройка её отслеживания удаленной ветки |

Чистый репозиторий — это репозиторий, не имеющий рабочей директории. Он содержит только директорию .git, в которой Git хранит все свои внутренние данные. Основное предназначение таких репозиториев — быть центральным хранилищем, в которое разработчики могут отправлять и из которого могут получать данные. Поэтому в них нет смысла создавать рабочие файлы, они будут только впустую занимать место на диске.

Таблица 18 – Чистый репозиторий

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git clone --bare *work work.git* | Создание чистого репозитория |
| git remote add shared ../*work.git* | Добавить чистый репозиторий в качестве удаленного репозитория к оригинальному репозиторию |