1 Плюсы GiT

1. Главная польза в том, что есть возможность откатить какие либо изменения, которые повлияли на работоспособность проекта.
2. Также гит является децентрализованной системой, что позволяет работать в отсутствии основного сервера.
3. Есть возмонжость вести групповые проекты и синхронизировать локально работу нескольких программистов. Есть возможность слияния различных веток проекта, которые были сделаны различными программистами

2 Как пользоваться гитом?

Гит - это набор скриптов, соответственно его необходимо установить.

1. Скачиваем, устанавливаем
2. нихуя не работает
3. если не помогла перезагрузка - то все, пизда
4. много гемора по входу в PATH

3 Почему пользуемся консолью?

1. Единый интерфейс на всех устройствах (до сих пор хуесосит окошки как систему)
2. В различных IDE работает все одинаково, т.к. везде есть терминал
3. Работает быстрее, т.к. исключается работа лишняя перед взаимодействием с гитом
4. В командной строке быстрее работать (как он говорит в несколько раз)
5. Подходит для удаленной работы на другом сервере/компьютере
6. Открытый исходный код (сложнее навредить), нет никакой зависимости от бизнеса/корпорации

Единственный плюс GUI - привычка.

4 Практическая часть

4.1 Репозиторий

Основа гита - репозиторий. Это папка, в которой есть дополнительная папка .git. Если необходимо в папке создать git репозиторий необходимо ввести git init. Наш репозиторий существует до тех пор, пока существует папка .git. Соответственно ее не трогать, пока не понимаешь, зачем конкретно ее удаляеью.

4.2 Создание и добавление файла

При создании нового файла/изменении текущего, он напишет, что он их не отслеживает. Команда git status даст информацию о всех файлах репозитория. Пока файлы "untracked" - изменения в них не отслеживаются. При написании команды git add - мы можем добавить в отслеживание. Т.е. добавим в состояние "staged". Далее с помощью команды git commit мы сделаем коммит, но выдаст ошибку, т.к. не было названия коммита. Для того, чтобы назвать коммит - необходимо указать флаг -m, который позволит далее в кавычках название коммита.

4.3 Политика названия коммитов

Существует политика называния коммита, которая заставляет называть их как то, зачем было сделано что либо. Крайне желательно придерживаться общего стиля т.е. можно указывать название изменения, но необходимо четко стандартизировать коммитов.

После выполнения команды он даст ответ, что были внесены изменения.

Написав теперь git status - мы получим, что он перестал быть "untracked"

4.4 Бинарные файлы

В гите нет необходимости сохранять бинарники, т.к. впустую тратит место, и при этом могут не воспроизвестись на компьютере.

4.5 Игнорирование файлов

Для игнорирования различных файлов необходимо указать их в файл .gitignore. Можно добавлять командой git add *.cpp все файлы имеющие определенный формат (в данном случае .cpp). Также с помощью команды

git restore –staged main.cpp можно не коммитить данный файл. Если написать просто git restore main.cpp - можно вернуть предыдущее изменение, которое в данном файле было до этого. Таким образом можно убирать из коммита изменения файла.

Если изменить файл, сделать git add, а далее еще раз изменить файл - то он будет сразу в 2х состояниях (1 - то что еще не добавлено второе изменение и 2 - то что изменение первое уже добавленно в коммит). Соответственно необходимо после каждого изменения файла его добавлять.

4.6 Настройки гита

В файле gitconfig указаны настройки гита (в том числе ваше имя фамилия и почта) крайне рекоммендуется указывать!!!

4.7 История проекта

Для того, чтобы отобразить историю проекта более кратко - можно использовать git log (–oneline), которая позволит уменьшить информацию показанную.

4.8 Хэш коммита

Каждый коммит обеспечивается своим хэшем, который составляется и зависит от даты и кучи прочей лабуды.

4.9 Откат изменений

Есть два варианта различного отката изменений.

revert Данный способ изменения позволяет оставить коммиты, но создать новый, который отменяет все изменения, внесенные в коммитах. Мы можем только добавлять изменения (не можем удалять коммиты)

При выполнении revert "название коммита" он создаст новый коммит

reset При выполнении данной штукенции гит делает вид, что данного коммита будто бы и не было. Есть два варианта использования данной комманды - –soft HEAD-1 В данной ситуации мы откатываемся на 1 коммит от последнего коммита. Можно сделать таким образом: откатиться мягко на 5 коммитов и обьеденить их. Мягкий коммит сохраняет изменения, которые были в этот момент сделаны, а hard - удаляет изменения. Но при ошибке можно выполнить комманду git ref log можно просмотреть

все хэши коммитов, и с помощью reset –hard "хэш" - вернуться к данному коммиту. Можно делать reset вперед.

4.10 Вопросы

Как не закидывать бинарники? Можно условиться как либо называть бинарники, можно скидывать все бинарники в определенную директорию, которая будет игнорироваться.

Зачем гит сейчас? Можно домашку сохранять, резюме, лучше не комментировать дофига, а сохранять с гитом через кучу коммитов много разных версий приложения.

Можно ли игнорить папки? Да, можно, просто необходимо название добавить в .gitignore

На винде боль? Да, боль, но в целом норм, если поставишь гит - то комммуникация будет одинаковой с линуксом, просто установи линукс.

4.11 Итоги

Сегодня не были затронуты ветки, коммандное взаимодействия программистов, т.к. они требуют понимания простейших вещей. Мы научились взаимодействовать с гитом, основным его коммандам. Следующее будет про ветки и в целом - коммандное.