Конспект по Git / **"Что? ~~Где? Когда?"~~ Зачем? Как работает?"**

1. **Что такое система контроля версий и зачем она нужна?**

Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов.

1. **Как происходит инициализация git?**

С помощью команды $ git init в необходимой папке, которая становится репозиторием

1. **Какие изменения происходят при этом в файловой системе?**

Создается скрытая папка «.git», в которой находятся все файлы Git’a

1. **Из каких логических частей состоит git?**
2. Рабочий каталог
3. Область подготовленных файлов
4. Каталог Git’a

1. Что такое коммит? Для чего он нужен?

Это команда Git для записи индексированных изменений в репозиторий.

2. Какие данные сохраняет коммит в GIT?

Коммит будет включать текущие состояния индексированных файлов плюс последние сохраненные состояния неиндексированных (но отслеживаемых) файлов. Обратите внимание: коммит включает в себя не изменения (дельты, патчи) относительно предыдущего коммита, а "снимок" (англ. shapshot) текущего состояния рабочей области.

3. Что делают команды git add . и git add \* ? В чем между ними разница?

Команда git add принимает параметром путь к файлу или каталогу. Если это каталог, команда рекурсивно добавляет (индексирует) все файлы в данном каталоге.

git add \* добавит все пути, которые являются результатом расширения оболочки \*, тогда как git add . сообщит git добавить текущий каталог.

git add \* не будет добавлять пути, начинающиеся с ., поскольку расширение оболочки \* считает, что это "скрытые" пути.

git add \* также отменит всю операцию и покажет сообщение об ошибке, если какой-либо расширенный путь в настоящее время игнорируется git. Git считает это ошибкой, когда явно указывается игнорируемый путь, без флага -f (force), показывающего, что действительно необходимо добавить проигнорированный путь.

4. Можно ли разбить один коммит на несколько? Если да, как это сделать?

По данному вопросу у меня есть два ответа^

Первый, более логичный, состоит в понятии "ветвление", осуществляемом с помощью команды git branch. С помощью создания веток получится как бы разделить коммит.

Однако я нагуглил еще одну заметку, когда объединенный коммит разделяют на части, причем, как я понял, делают это именно физически с помощью действия edit ttps://clck.ru/GzEtK

5. Как прицепить указатель ветки к нужному коммиту?

С помощью команды git branch -f , используя в качестве аргументов указатель ветки и хэш/относительную ссылку коммита

6. Что такое режим Detaching HEAD? (отрыв башки)? Как оно работает?

HEAD - это символическое имя текущего выбранного коммита — это, по сути, тот коммит, над которым мы в данный момент работаем. HEAD всегда указывает на последний коммит из локального дерева.

Обычно HEAD указывает на имя ветки.

Переходим к сути вопроса:

Отделение (detaching) HEAD означает лишь присвоение его не ветке, а конкретному коммиту с помощью того же checkout

7. Как работают относительные ссылки? Для чего они нужны?

Относительные ссылки позволяют проще передвигаться по дереву git. Они обозначают исходную точку и количество движений от исходной точки. Благодаря относительным ссылкам, не нужно идентифицировать каждый коммит по хэшу, чтобы обратиться к нему.

Еще одна из наиболее распространённых целей, для которых используются относительные ссылки - это перемещение веток (указателей веток). Можно напрямую прикрепить ветку к коммиту при помощи опции -f. Например, команда:

git branch -f master HEAD~3

Переместит (принудительно) ветку master на три родителя назад от HEAD.