**Дополнительные вопросы**

1. Что такое рабочая копия (working copy)?

Рабочая копия - это копия репозитория с основного сервера, с которой в последствии планируется работа и внесение изменений в некоторые файлы.

Рабочая копия создается командой git clone <ссылка на репозиторий>.

1. Что такое ревизия? И есть ли отличие от коммита?

Ревизия - это фиксация любого изменения в системе контроля версий. Коммит - это процесс создания ревизии. Сама ревизия может содержать дельту изменения или снимок.

1. Что такое Stage area (index) в git?

С помощью промежуточной обработки появляется возможность вносить отдельные небольшие последовательные изменения. Предположим, что была произведена работа над file1.txt и file2.txt. Их можно добавить в stage area сразу командой git add –A. Но если необходимо срочно загрузить изменения первого файла на сервер, то git add file1.txt и продолжить работу с другим файлом.

1. Чем отличаются snapshot-based от diff-based системы контроля версий?

Снимки - это файлы после изменений. Дельта изменений - это разница содержимого между начальным файлом и файлом, который был изменен.

1. Что такое отмена? (подумайте, какие есть варианты)

Существуют несколько операций отмены:

1) допустим, что при коммите не были включены дополнительные файлы или добавлен неправильный комментарий, тогда, чтобы переопределить коммит выполняется команда с параметром --amend: git commit –amend.

2) Отмена индексации файла. После того, как добавили файл в stage area, можно его исключить командой git reset <file>.

3) Отмена изменений в файле. Например, когда не хотим сохранять изменения в файле, тогда git checkout -- <file>. Локальный файл заменится версией из последнего коммита.

4) Отмена индексации файла с помощью git restore. Аналогично п.2, только команда git restore –staged <file>.

5) Откат измененного файла с помощью git restore. Аналогично п.3. Если не хотим сохранять изменения в файле, то git restore <file>.

1. Что такое ветки (branch)? Помимо формального описания, подумай, как можно было бы описать своими словами, для чего нужна ветка и что она и из себя представляет?

Ветки нужны для того, чтобы работать с версиями репозиториев. Ветка представляет из себя указатель на последний коммит. При работе с веткой коммиты применяются только в данной ветке. Это дает возможность разработчикам удобно работать с проектом. При необходимости, можно производить слияние веток.

1. Что такое конфликты (conflict)? Как они возникают? Сколько версий файла при конфликте? Какие это версии? Как разрешаются конфликты?

Конфликты - это решения разработчиков, которые не могут существовать вместе без дополнительных поправок. Конфликты могут возникнуть на этапе слияния.

Например, в один и тот же файл вносится изменение. Изменение от одного разработчика затрагивает ту же часть кода, что и у другого. Если первый разработчик сделал коммит изменений файла в репозиторий сервера, то второй при по пытке сделать коммит получит уведомление о конфликте.

Git добавляет в конфликтующие файлы специальные маркеры конфликтов для исправления их вручную.

Решить их можно путем заменой блоков или использования графического редактора merge-tool.

1. Rebase vs merge

Merge - слияние двух веток. Допустим, есть ветка master и test, произошедшая от последнего commit ветки master. С веткой master работали и вносили некоторые изменения, делали commit. Появилась необходимость слияния ветки master в ветку test. При использовании команды слияния в ветке test создается новый коммит, в котором учитываются изменения мастера.

Преимущество в том, что эта операция не уничтожает история коммитов ветки master.

Недостаток в том, что если ведется активная работка с master и в ветке test, к тому же операция слияния будет производиться часто, то ветка test “засорится” историей коммитов и будет непонятно какие конкретно изменения производились в файле.

Rebase - перемещение ветки. Та же ситуация с веткой master и test. При использовании способа rebase происходит ситуация, когда к ветке master “присоединяется” ветка test. Последний коммит ветки мастера образуется на основе дельты изменений между коммитом, от которого произошла ветка тест и первым коммитом ветки test. Последний коммит после rebase - коммит ветки test. Т.е. история “перезатирается” и остаются актуальные данные об последних коммитах. Это есть преимущество. С такой историей коммитов удобно работать.

Однако, постоянное использование Rebase может скрыть контекст коммитов от разработчиков. Поэтому необходимо сочетать использование Rebase и Merge.

1. Что такое cherry-pick? И когда, на твой взгляд, может пригодиться?

Cherry-pick позволяет перенести 1 коммит из одной ветки в другую.

Команда может пригодиться в случае, если необходимо быстро внести изменения в основную ветку, работая при этом в отдельной, где есть прочие изменения, которые не должны попасть в основную ветку.

При работе с двумя ветками, если требуется в одной из ветки внести отдельный коммит из другой, то пользуются данной командой.

Восстановление или сохранения потерянных коммитов. Возможно, что работа с одной веткой давно прекратилась и данные на ней неактуальны, но есть один коммит, который может понадобиться при ведении другой ветки.