Что представляет собой система *GIT*?

Ответ: Git — это распределенная система управления версиями, которая означает, что локальный клон проекта — это полный репозиторий управления версиями. Полнофункциональные локальные репозитории упрощают работу как в автономном, так и в удаленном режиме. Разработчики фиксируют свою работу локально, а затем синхронизируют копию репозитория с копией на сервере.

Перечислите возможности систем управления версиями.

### Ответ: 1) Локальные системы контроля версий

### Диаграмма локального контроля версий

Одной из популярных VCS была система RCS, которая и сегодня распространяется со многими компьютерами. RCS хранит на диске наборы патчей (различий между файлами) в специальном формате, применяя которые она может воссоздавать состояние каждого файла в заданный момент времени.

### 2) Централизованные системы контроля версий



Такой подход имеет множество преимуществ, особенно перед локальными VCS. Например, все разработчики проекта в определённой степени знают, чем занимается каждый из них. Администраторы имеют полный контроль над тем, кто и что может делать, и гораздо проще администрировать CVCS, чем оперировать локальными базами данных на каждом клиенте.

### 3) Распределённые системы контроля версий

Более того, многие DVCS могут одновременно взаимодействовать с несколькими удалёнными репозиториями, благодаря этому вы можете работать с различными группами людей, применяя различные подходы единовременно в рамках одного проекта. Это позволяет применять сразу несколько подходов в разработке, например, иерархические модели, что совершенно невозможно в централизованных системах.



Что такое commit и когда он выполняется?

Ответ: Commit/Коммит — это способ сохранения изменений в коде. Каждый commit содержит информацию о том, что было изменено в коде и кем были внесены эти изменения. Они позволяют разработчикам отслеживать изменения в своем (или чужом) коде и возвращаться к предыдущим версиям, если это необходимо.  
  
Когда разработчик вносит изменения в код, он в любой момент может сохранить их с помощью команды git commit - в этом случае новый коммит будет содержать информацию о внесенных изменениях и авторе этих изменений.

Как посмотреть историю коммитов?

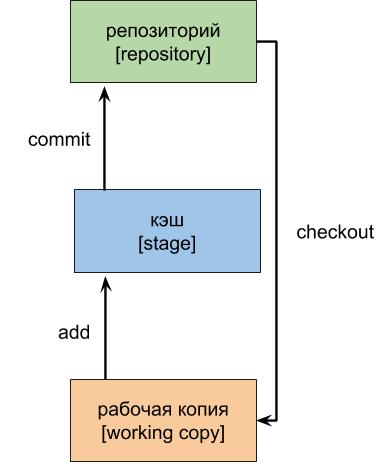
Ответ: Одним из основных и наиболее мощных инструментов для этого является команда git log. По умолчанию (без аргументов) git log перечисляет коммиты, сделанные в репозитории в обратном к хронологическому порядке — последние коммиты находятся вверху. Команда git log имеет очень большое количество опций для поиска коммитов по разным критериям.

Как создать пустой GIT-репозиторий?

Ответ: Это можно сделать с помощью команды git init. Данная команда создает репозиторий, то есть хранилище всех изменений файлов в этом каталоге. Чтобы система начала контроль версий файлов, необходимо добавить их в репозиторий. То есть, следующим шагом необходимо добавить, например, файл «Название файла» в созданный репозиторий и добавить коммит, говорящий о том, что текущая версия файла не содержит никакого программного кода, т.е. файл пустой.

Какую архитектуру имеет система контроля версий *GIT*?

Ответ: Система контроля версий *git*использует архитектуру трех деревьев. Схематично она выглядит так как показано на рисунке ниже.



Суть ее заключается в том, что дополнительно добавляется ещё одно место, которое можно назвать кэшем или *stage*в английской терминологии. Рабочая копия и репозиторий идейно не отличается от их аналогов в архитектуре двух деревьев.

Как создать новую ветку в *GIT* и соединить ветки?

Ответ: Создание ветки является одним из самых важных и удобных инструментов системы контроля версий является ветвление. Таким образом, один проект может разветвиться на несколько отдельно развивающихся ветвей, преследующих разные цели, оперирующие разными версиями файлов и так далее. Кроме того, в ходе разработки ветвей, можно устроить их слияние. С помощью команды git checkout -b «Название ветки» можно создать ветвь, а чтобы слить их нужно использовать команды git checkout «Созданная ветка» и git merge master.

Поясните понятие «staging area»

Ответ: Область подготовленных файлов (**staging area**) — это обычный файл, обычно хранящийся в каталоге Git, который содержит информацию о том, что должно войти в следующий коммит. Иногда его называют индексом (index), но в последнее время становится стандартом называть его областью подготовленных файлов. Область подготовленных файлов — это уже не рабочий каталог, но ещё и не коммит.

Как сравнить версии файла в *GIT*?

Ответ: Для сравнения версий файла в Git можно использовать команду "git diff". В терминале это должно выглядеть следующим образом: git diff "Название файла". Git покажет различия между текущей версией файла и последним коммитом в репозитории. Вы увидите добавленные, удаленные и измененные строки кода. Вы также можете указать определенные коммиты или ветки для сравнения, используя соответствующие параметры команды "git diff".