1. **Что такое git?**

Система управления версиями — программное обеспечение для облегчения работы с изменяющейся информацией.

1. **В чем отличие git от других систем контроля версий?**

Концептуально, большинство других систем хранят информацию в виде списка изменений в файлах. Эти системы представляют хранимую информацию в виде набора файлов и изменений, сделанных в каждом файле, по времени (обычно это называют контролем версий, основанным на различиях).

Git не хранит и не обрабатывает данные таким способом. Вместо этого, подход Git’а к хранению данных больше похож на набор снимков миниатюрной файловой системы. Каждый раз, когда вы делаете коммит, то есть сохраняете состояние своего проекта в Git’е, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок. Для увеличения эффективности, если файлы не были изменены, Git не запоминает эти файлы вновь, а только создаёт ссылку на предыдущую версию идентичного файла, который уже сохранён. Git представляет свои данные как, скажем, поток снимков.

1. **Назовите три возможных состояния файлов в git. В чем их различия?**

В Git'е файлы могут находиться в одном из трёх состояний: зафиксированном, изменённом и подготовленном. "Зафиксированный" значит, что файл уже сохранён в вашей локальной базе. К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы. Подготовленные файлы — это изменённые файлы, отмеченные для включения в следующий коммит.

**4. Что такое индекс? Для чего он используется?**

Перед сохранением любого файла Git вычисляет контрольную сумму, и она становится индексом этого файла. Поэтому невозможно изменить содержимое файла или каталога так, чтобы Git не узнал об этом. Эта функциональность встроена в сам фундамент Git'а и является важной составляющей его философии. Если информация потеряется при передаче или повредится на диске, Git всегда это выявит.

1. **Какая команда служит для добавления файлов в следующий коммит?**

git add

1. **Для чего используется команда merge? В чем отличие merge от rebase?**

В Git'е есть два способа включить изменения из одной ветки в другую: merge (слияние) и rebase(перемещение).

Наиболее простое решение для объединения веток, как мы уже выяснили, команда merge. Эта команда выполняет слияние между двумя последними снимками состояний.

Однако, есть и другой путь: вы можете взять изменения и применить их поверх. В Git'е это называется **перемещение** (rebasing). При помощи команды rebase вы можете взять все изменения, которые попали в коммиты на одной из веток, и повторить их на другой.

1. **Для чего используется команда fetch? В чем отличие команды fetch от pull?**

$ git fetch [remote-name]

Данная команда связывается с указанным удалённым проектом и забирает все те данные проекта, которых у вас ещё нет. После того как вы выполнили команду, у вас должны появиться ссылки на все ветки из этого удалённого проекта. Теперь эти ветки в любой момент могут быть просмотрены или слиты.

Если у вас есть ветка, настроенная на отслеживание удалённой, то вы можете использовать команду git pull, чтобы автоматически получить изменения из удалённой ветви и слить их со своей текущей ветвью. Этот способ может для вас оказаться более простым или более удобным.

1. **Какая команда служит для фиксации изменений?**

git commit

1. **Какая команда позволяет просмотреть информацию о текущем индексе репозитория?**
2. **Для чего служит команда stash?**

Команда **git stash** предназначена для того, чтобы поместить текущие изменения, которые вы выполнили в файлах, в отдельное хранилище, и вернуть файлы к исходному состоянию. То есть **git stash** **прячет** изменения в файлах и сохраняет эти изменения отдельно, чтобы потом можно было их вернуть.

1. **Какими способами можно создать локальный репозиторий?**

Первый подход — импорт в Git уже существующего проекта или каталога. Второй — клонирование уже существующего репозитория с сервера.

**12. Для чего нужен .gitignore файл?**

Зачастую, у вас имеется группа файлов, которые вы не только не хотите автоматически добавлять в репозиторий, но и видеть в списках неотслеживаемых. К таким файлам обычно относятся автоматически генерируемые файлы (различные логи, результаты сборки программ и т.п.). В таком случае, вы можете создать файл .gitignore. с перечислением шаблонов соответствующих таким файлам.

**13. Для чего используются ветки в git? Как создать/удалить ветку?**

Что же на самом деле происходит, когда вы создаете ветку? Всего лишь создается новый указатель для дальнейшего перемещения. Допустим вы хотите создать новую ветку с именем “testing” Вы можете это сделать командой git branch

**14. Каким образом перенести изменения с одной ветки в другую?**

git merge - команда для объединения веток

**15. Какая команда позволяет отобразить удаленные репозитории, связанные с текущим локальным?**

Для того, чтобы просмотреть список настроенных удалённых репозиториев, вы можете запустить команду git remote. Она выведет названия доступных удалённых репозиториев.

**16. Каким образом можно получить изменения из удаленного репозитория в локальный?**

Когда вы хотите поделиться своими наработками, вам необходимо отправить (push) их в главный репозиторий. Команда для этого действия простая: git push

**17. Что такое pull request?**

Люди просто могут создавать свои собственные ветвления (fork), вносить туда изменения, а затем отправлять свои внесенные изменения в оригинальный репозиторий проекта путем создания запроса на принятие изменений (Pull Request), сами же запросы на принятие изменений (Pull Request) будут описаны далее. Запрос на принятие изменений (Pull Request) откроет новую ветвь с обсуждением отправляемого кода, и автор оригинального проекта, а так же другие его участники, могут принимать участие в обсуждения предлагаемых изменений до тех пор, пока автор проекта не будет ими доволен, после чего автор проекта может добавить предлагаемые изменения в проект.

**18. Какая команда позволяет отобразить историю репозитория?**

После того как вы создадите несколько коммитов, или же вы склонируете репозиторий с уже существующей историей коммитов, вы, вероятно, захотите оглянуться назад и узнать, что же происходило с этим репозиторием. Наиболее простой и в то же время мощный инструмент для этого — команда git log.

$ git init

Эта команда создаёт в текущей директории новую поддиректорию с именем .git, содержащую все необходимые файлы репозитория — основу Git-репозитория

Если вы хотите добавить под версионный контроль существующие файлы (в отличие от пустого каталога), вам стоит добавить их в индекс и осуществить первый коммит изменений. Добиться этого вы сможете запустив команду git add несколько раз, указав индексируемые файлы, а затем выполнив git commit