Вопросы:

1. Что такое система контроля версий, для чего ее используют?  
2. Какой принцип хранения файлов использует Git?  
3. В каких трех основных состояниях файлы могут находиться в Git ?  
4. Что такое индексация файла? Какой командой она выполняется?  
5. Что такое фиксация файла? Какой командой она выполняется?  
6. Продемонстрируйте команду проверки выбранных настроек.  
7. Как инициализировать репозиторий в существующей папке?  
8. Как указать файлы, за которыми должна следить система?  
9. Как выполнить фиксацию изменений?  
10.Какой командой определить состояния файлов?  
11.Для чего создается файл .gitignore?Поясните его структуру.  
12.Как используется команда git diff?  
13.Как используется команда git commit?  
14.Как используется команда git log? Какие у нее есть параметры?  
15.Как используется команда git commit –amend?  
16.Как отобразить удаленные репозитории?  
17.Как извлечь данные из удаленного репозитория?  
18.Как отправить данные в удаленный репозитроий?  
19.Продемонстируйте создание новых веток в Git? Что такое  
ветвление? Что такое указатель HEAD?  
20.Как используется команда git checkout?  
21.Как выполнить включение изменений из одной ветки в другую?  
22.Какие проблемы могут быть при слиянии и как они разрешаются?  
23.Что такое GitLab?

Ответы:

1. Система управления версиями (от англ.  
VersionControl System, VCS или  
Revision Control System) — программное  
обеспечение для облегчения работы с  
изменяющейся информацией

2. В принципе, большинство других систем хранит информацию как список изменений (патчей) для файлов. Эти системы (CVS, Subversion, Perforce, Bazaar и другие) относятся к хранимым данным как к набору файлов и изменений, сделанных для каждого из этих файлов во времени

3.Модифицированное, индексирование и зафиксированное.

4. В процессе индексации мы добавляем изменения, которые войдут в следующий коммит. Команда git add.

5. Фиксация - сохранение снимка состояния проекта. Выполняется командой git commit.

6. core.symlinks=false

core.autocrlf=true

core.fscache=true

color.diff=auto

color.status=auto

color.branch=auto

color.interactive=true

help.format=html

rebase.autosquash=true

http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt

http.sslbackend=openssl

diff.astextplain.textconv=astextplain

filter.lfs.clean=git-lfs clean -- %f

filter.lfs.smudge=git-lfs smudge --skip -- %f

filter.lfs.process=git-lfs filter-process --skip

filter.lfs.required=true

credential.helper=manager

filter.lfs.smudge=git-lfs smudge -- %f

filter.lfs.process=git-lfs filter-process

filter.lfs.required=true

filter.lfs.clean=git-lfs clean -- %f

user.name=shpraha

user.email=yarik-yarik-01@mail.ru

:...skipping...

core.symlinks=false

core.autocrlf=true

core.fscache=true

color.diff=auto

color.status=auto

color.branch=auto

color.interactive=true

help.format=html

rebase.autosquash=true

http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt

http.sslbackend=openssl

diff.astextplain.textconv=astextplain

filter.lfs.clean=git-lfs clean -- %f

filter.lfs.smudge=git-lfs smudge --skip -- %f

filter.lfs.process=git-lfs filter-process --skip

filter.lfs.required=true

credential.helper=manager

filter.lfs.smudge=git-lfs smudge -- %f

filter.lfs.process=git-lfs filter-process

filter.lfs.required=true

filter.lfs.clean=git-lfs clean -- %f

user.name=shpraha

user.email=yarik-yarik-01@mail.ru

core.repositoryformatversion=0

core.filemode=false

core.bare=false

core.logallrefupdates=true

core.symlinks=false

core.ignorecase=true

remote.origin.url=https://github.com/treegoon/gitec.git

remote.origin.fetch=+refs/heads/\*:refs/remotes/origin/\*

branch.master.remote=origin

branch.master.merge=refs/heads/master

7. Войти в эту папку при помощи команды cd, далее воспользоваться командой git init, после этого будет создан локальный репозиторий.

8. Как указать файлы, за которыми должна следить система?   
Чтобы начать управление версиями существующих файлов (в противовес пустому каталогу), укажите файлы, за которыми должна следить система, и выполните первую фиксацию изменений. Для этого потребуется несколько команд git add, добавляющих файлы, за которыми вы хотите следить, а затем команда git commit:   
git add \*.txt   
git commit -m 'первоначальная версия проекта'

9.

1. Перед тем, как зафиксировать изменения отдельных файлов, нужно добавить файлы в набор этих изменений. Воспользуйтесь командой git add имя-файла. Если вы хотите сохранить все изменения разом, вводите git add -A.
2. Теперь мы можем сделать **коммит**, то есть зафиксировать все сохранённые изменения и дать им название. Это делается с помощью команды git commit -m "ваше сообщение".
3. Чтобы отправить свои изменения (коммиты) в репозиторий на GitHub, введите команду git push origin название-текущей-ветки, где origin означает репозиторий, который был склонирован на компьютер, то есть ваш форк.

10. Команда, которая определяет состояние файлов: git status

11. В большинстве проектов есть файлы или целые директории, в которые мы не хотим (и, скорее всего, не захотим) коммитить. Мы можем удостовериться, что они случайно не попадут в git add -A при помощи файла .gitignore

1)Создайте вручную файл под названием .gitignore и сохраните его в директорию проекта.

2)Внутри файла перечислите названия файлов/папок, которые нужно игнорировать, каждый с новой строки.

3)Файл .gitignore должен быть добавлен, закоммичен и отправлен на сервер, как любой другой файл в проекте.

Файл .gitignore, исключающий все перечисленное выше, будет выглядеть так:

\*.log

build/

node\_modules/

.idea/

my\_notes.txt

Символ слэша в конце некоторых линий означает директорию (и тот факт, что мы рекурсивно игнорируем все ее содержимое). Звездочка, как обычно, означает шаблон.

12. Для вывода изменений в файлах по сравнению с последним коммитом, используется git diff

Если вы изменили какие-нибудь файлы в вашем рабочем каталоге и добавили один или несколько из них в индекс (с помощью git add), то команда git diff не покажет изменения в этих файлах. Чтобы показать изменения в файлах, включая файлы, добавленные в индекс, используется ключ --cached: git diff –cached

13. Коммиты — базовое понятие во всех системах контроля версий, поэтому совершатся  
он должен легко и по возможности быстро. В самом своем простом виде достаточно  
после индексации набрать:  
git commit  
Если индекс не пустой, то на его основе будет совершен коммит, после чего  
пользователя попросят прокомментировать вносимые изменения вызовом команды  
edit. Сохраняемся и Коммит готов.  
Есть несколько ключей, упрощающих работу с git commit:  
git commit -a — совершит коммит, автоматически индексируя изменения в файлах  
проекта. Новые файлы при этом индексироваться **не будут**! Удаление же файлов  
будет учтено.  
git commit -m «commit comment» — комментируем коммит прямо из командной строки  
вместо текстового редактора.  
git commit FILENAME — внесет в индекс и создаст коммит на основе изменений  
единственного файла.

14. git log — разнообразная информация о коммитах в целом, по отдельным файлам и различной глубины погружения в историю. Иногда требуется получить информацию об истории коммитов, коммитах, изменивших отдельный файл; коммитах за определенный отрезок времени и так далее. Для этих целей используется команда git log.  
Простейший пример использования, в котором приводится короткая справка по всем  
коммитам, коснувшимся активной в настоящий момент ветки:

git log  
Получить подробную информацию о каждом в виде патчей по файлам из коммитов  
можно, добавив ключ -p (или -u):  
git log -p  
Статистика изменения файлов, вроде числа измененных файлов, внесенных в них  
строк, удаленных файлов вызывается ключом --stat:  
git log --stat  
За информацию по созданиям, переименованиям и правам доступа файлов отвечает ключ  
git log --summary

15.

#### Изменение последнего коммита

Если вы что-либо недоглядели в последнем коммите, то отредактировать его не составит никакого труда. Все, что нужно это добавить изменения обычным образом:

git add .

Затем закоммитить изменения с параметром --amend (amend /əˈmɛnd/ — вносить поправки, исправлять, улучшать):

git commit --amend

16. Чтобы просмотреть, какие удалённые серверы у вас уже настроены, следует выполнить команду git remote.

17. $ git fetch [remote-name]

18. $ git push origin master

19. $git branch "имя", Ветвление означает, что вы отклоняетесь от основной линии разработки и продолжаете работу, не вмешиваясь в основную линию. HEAD - указатель на локальную ветку, на которой вы находитесь.

20. Вводится имя ветки, на которую нужно переключиться

21. $git merge

22. Отличия при слиянии решаются вручную путем изменения файлов

23. GitLab — сайт и система управления репозиториями кода для Git.