1.

Система контроля версий (Version Control System, VCS) представляет собой программное обеспечение, которое позволяет отслеживать изменения в документах, при необходимости производить их откат, определять, кто и когда внес исправления и т.п

Системы контроля версий можно разделить на две группы: распределенные и централизованные.

**Централизованные системы контроля версий**

Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществлялся через специальное клиентское приложение. В качестве примеров таких программных продуктов можно привести CVS, Subversion.

**Распределенные системы контроля версий**

Распределенные системы контроля версий (Distributed Version Control System, DVCS) позволяют хранить репозиторий (его копию) у каждого разработчика, работающего с данной системой. При этом можно выделить центральный репозиторий (условно), в который будут отправляться изменения из локальных и, с ним же эти локальные репозитории будут синхронизироваться. При работе с такой системой, пользователи периодически синхронизируют свои локальные репозитории с центральным и работают непосредственно со своей локальной копией. После внесения достаточного количества изменений в локальную копию они (изменения) отправляются на сервер. При этом сервер, чаще всего, выбирается условно, т.к. в большинстве DVCS нет такого понятия как “выделенный сервер с центральным репозиторием”.

2.

В таких системах, например CVS, Subversion и Perforce, есть центральный сервер, на котором хранятся все файлы под версионным контролем, и ряд клиентов, которые получают копии файлов из него. Много лет это было стандартом для систем контроля версий.

В таких системах как Git, Mercurial, Bazaar или Darcs клиенты не просто выгружают последние версии файлов, а полностью копируют весь репозиторий. Поэтому в случае, когда "умирает" сервер, через который шла работа, любой клиентский репозиторий может быть скопирован обратно на сервер, чтобы восстановить базу данных. Каждый раз, когда клиент забирает свежую версию файлов, он создаёт себе полную копию всех данных.

3.

По умолчанию Subversion использует модель копирование-изменение-слияние.

Модель копирование-изменение-слияние может выглядеть немного беспорядочно, но на практике все работает гладко. Пользователи могут работать одновременно, не тратя время на ожидание других. Обычно оказывается, что большинство одновременно вносимых изменений в файл вообще не пересекается; конфликты бывают редко. И время, потраченное на их разрешение, значительно меньше времени, отнимаемого блокировками системы.

4.

Команда «Фиксировать», в отличии от команды «Обновить», позволяет зафиксировать не весь файл, а лишь его часть.

5.

1) Создаём ветки master и dev

2) Создаём ветки для функций (feature branches)

3) Создаём ветки релиза

4) Создаём ветки Hotfix

Команда **git commit** берёт все данные, добавленные в индекс с помощью **git add**, и сохраняет их слепок во внутренней базе данных, а затем сдвигает указатель текущей ветки на этот слепок.

Команда **git reset**, как можно догадаться из названия, используется в основном для отмены изменений.

Команда **git add** добавляет содержимое рабочего каталога в индекс (staging area) для последующего коммита.

Команда **git status** показывает состояния файлов в рабочем каталоге и индексе: какие файлы изменены, но не добавлены в индекс; какие ожидают коммита в индексе. Вдобавок к этому выводятся подсказки о том, как изменить состояние файлов.

Команда **git diff** используется для вычисления разницы между любыми двумя Git деревьями.

Команда **git branch** — это своего рода "менеджер веток". Она умеет перечислять ваши ветки, создавать новые, удалять и переименовывать их.

Команда **git checkout** используется для переключения веток и выгрузки их содержимого в рабочий каталог.

Команда **git pull** работает как комбинация команд git fetch и git merge, т. е. Git вначале забирает изменения из указанного удалённого репозитория, а затем пытается слить их с текущей веткой.

Команда **git push** используется для установления связи с удалённым репозиторием, вычисления локальных изменений отсутствующих в нём, и собственно их передачи в вышеупомянутый репозиторий.