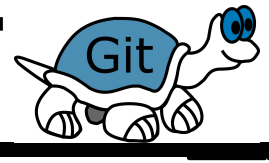
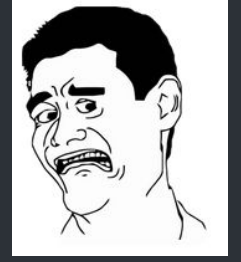
Система управления версиями Git

Автор: Кучук Михаил Сергеевич
e-mail: m.kuchuk@owen.ru

БЫСТРЫЙ СТАРТ



Что? Какой еще гит?



Git – распределенная система контроля версий. Проект появился в 2005 году для версионирования кода ядра Linux взамен платной системы BitKeeper. Название дано основателем проекта Торвальдсом и дословно подчеркивает эгоизм основателя Linux - «мерзавец».

Цели Git:

- высокая скорость работы;
- поддержка нелинейной разработки (тысячи параллельных веток);
- полная децентрализация;
- оптимизация для больших проектов (объем дискового пространства и быстродействие).

Git vs Subversion: преимущества

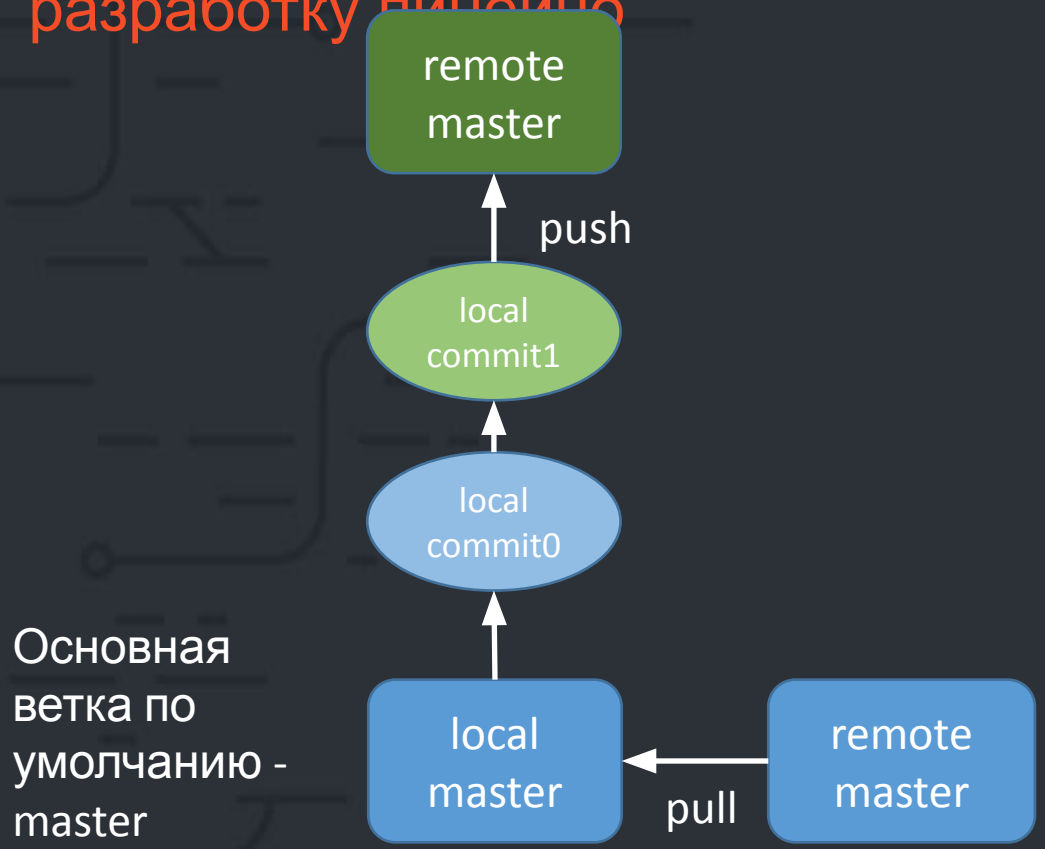
- возможность локальной работы с репозитарием без доступа к сети;
- хранение только изменений от версии к версии;
- быстрое создание/переключение веток;
- легкое слияние веток;
- расширенная работа с коммитами;
- поддержка сложных ветвлений;
- утилита TortoiseGit.

- линейная «человеческая» нумерация версии от младшей к старшей;
- поддержка бинарных файлов;
- доступно частичное клонирование -возможность выписать любой узел SVN;
- навигация по всем узлам;
- низкий порог вхождения;
- утилита TortoiseSVN.

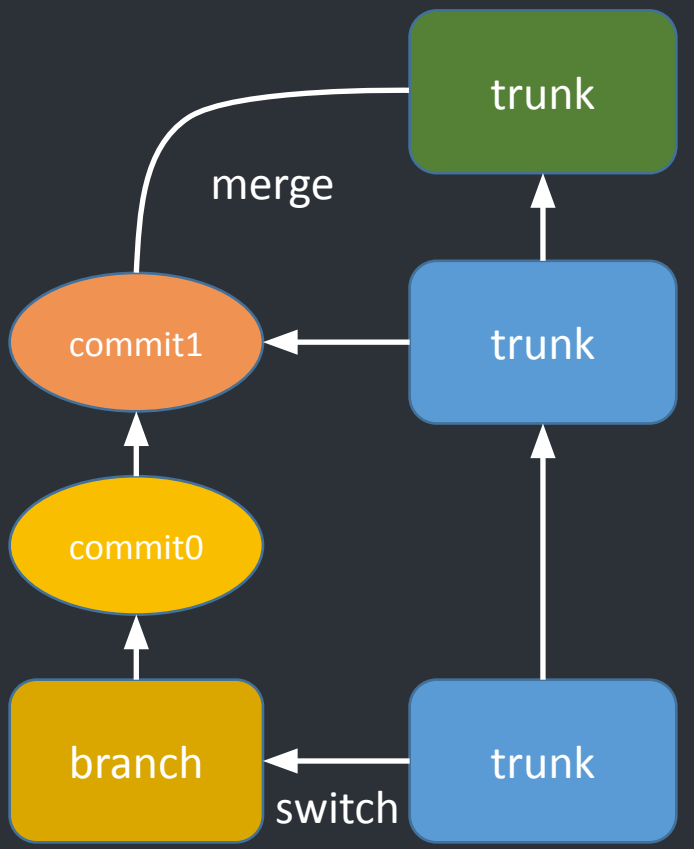
Git vs Subversion: идеология промежуточных коммитов

Промежуточные локальные коммиты позволяют вести разработку линейно

Из-за отсутствия поддержки локальных коммитов требуется создание веток



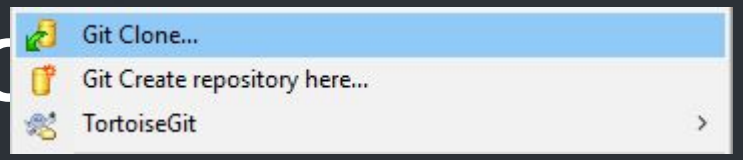
Основная ветка по умолчанию - master



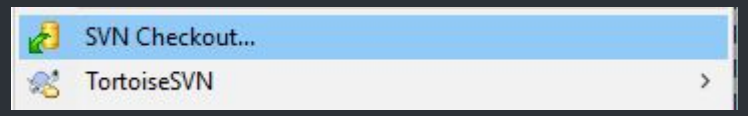
Основная ветка по умолчанию - trunk

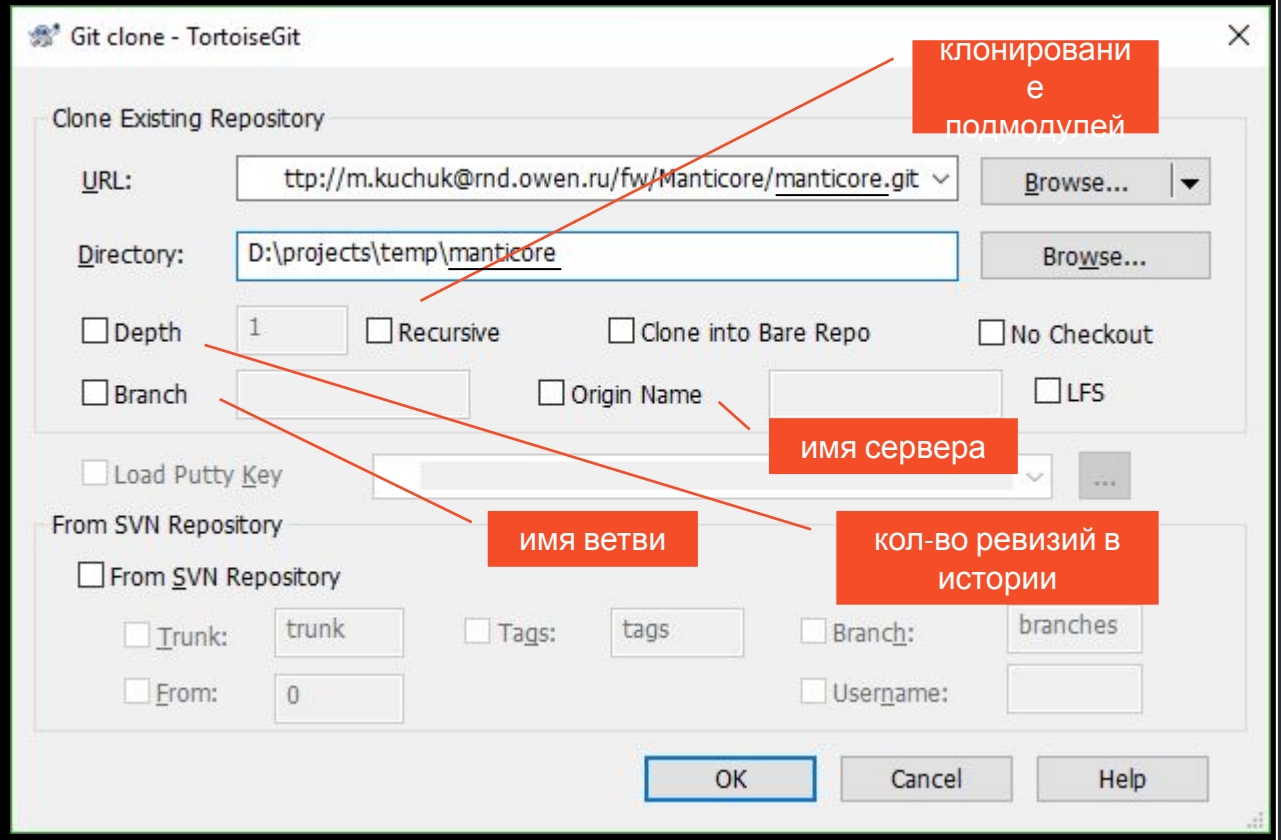
Git vs Subversion: получение исхо

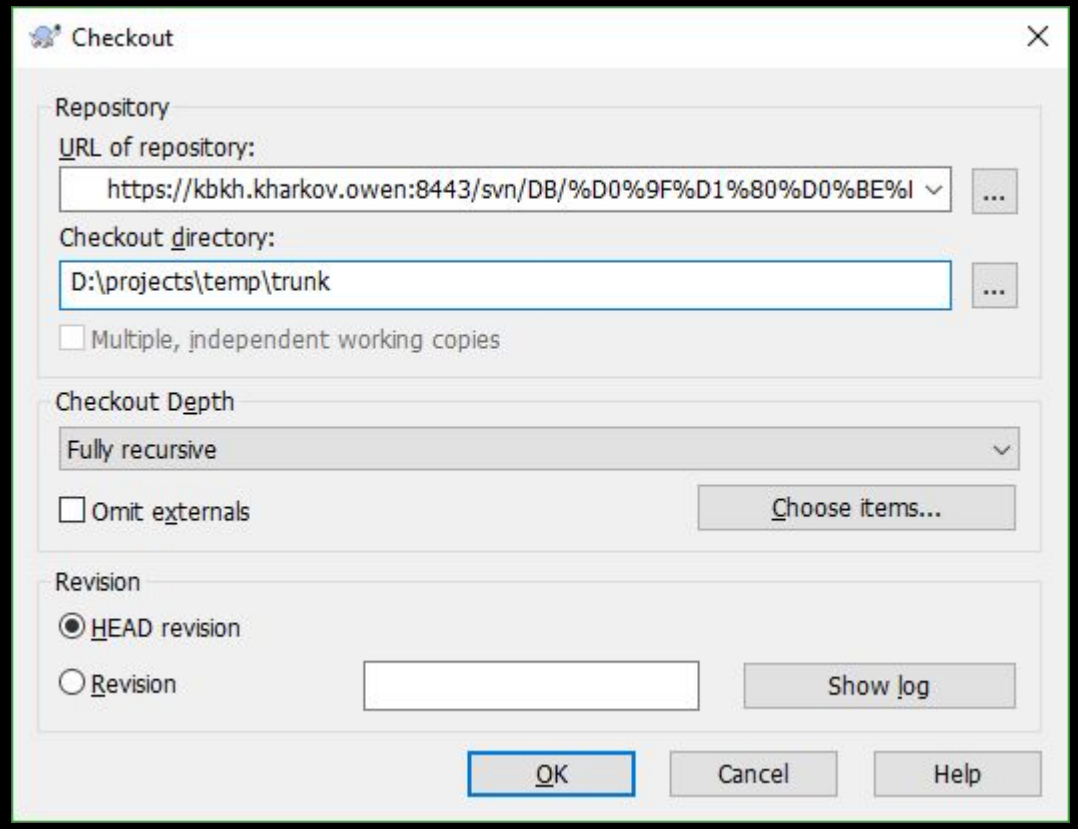
Git Clone



SVN Checkout



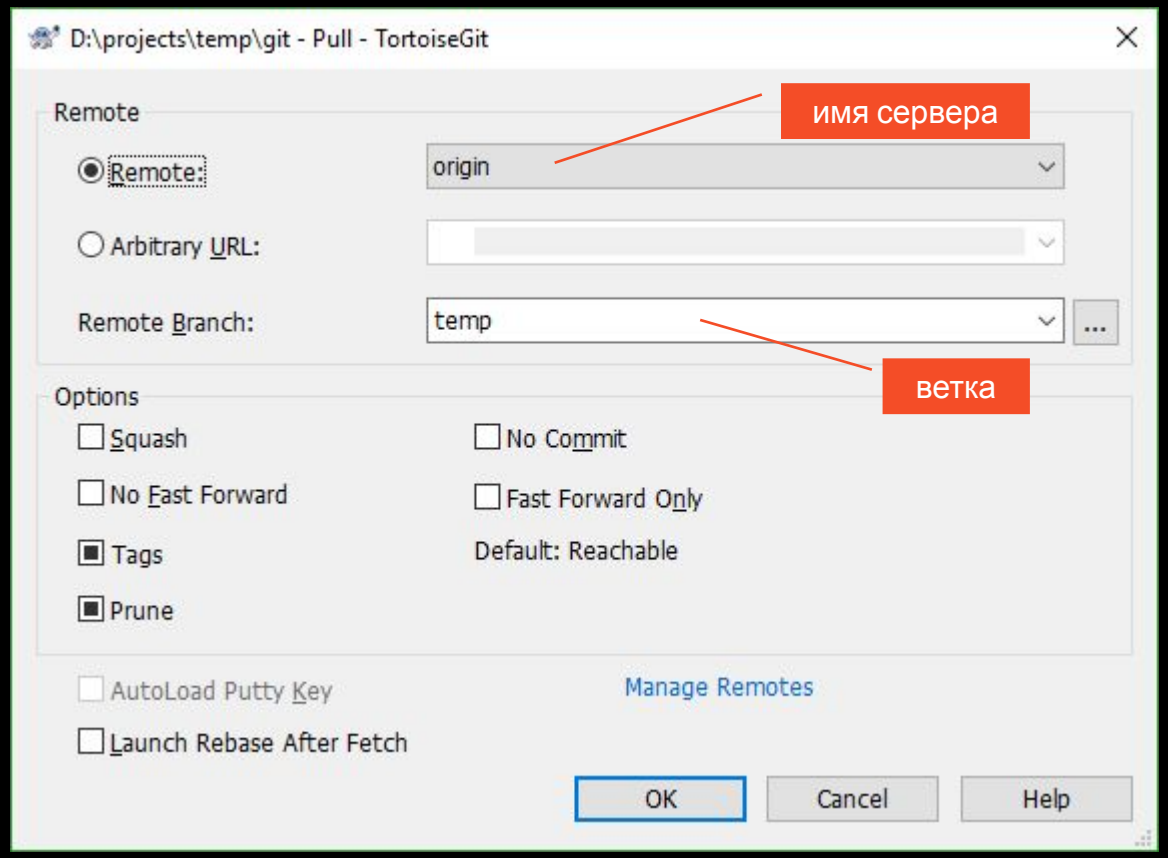




Git vs Subversion: получение

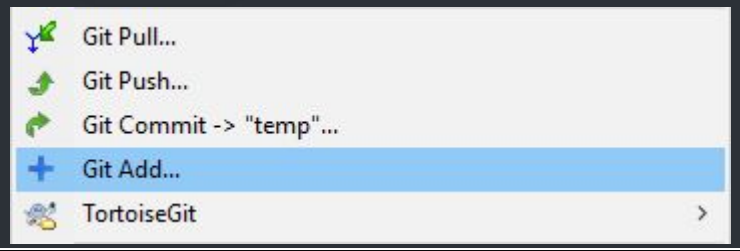
Git Pull из

SVN Update

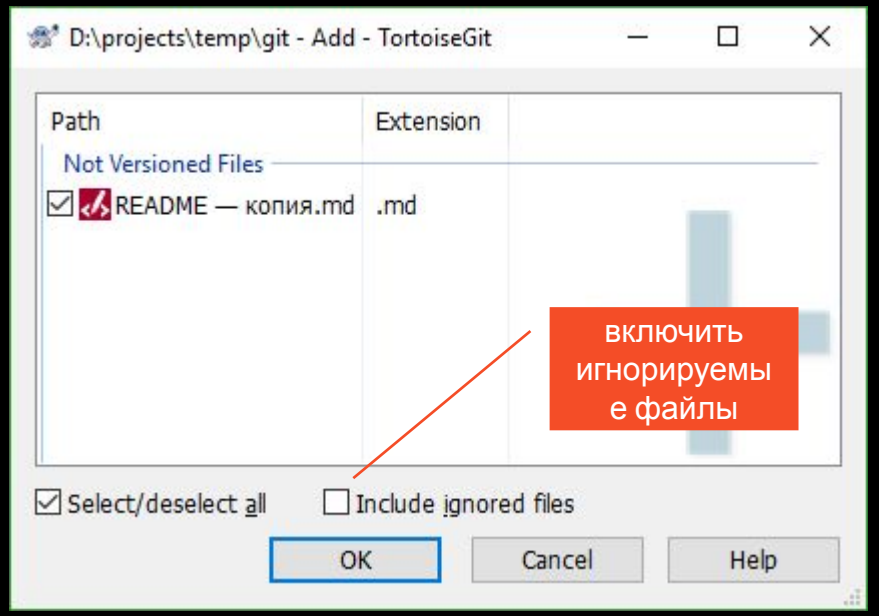


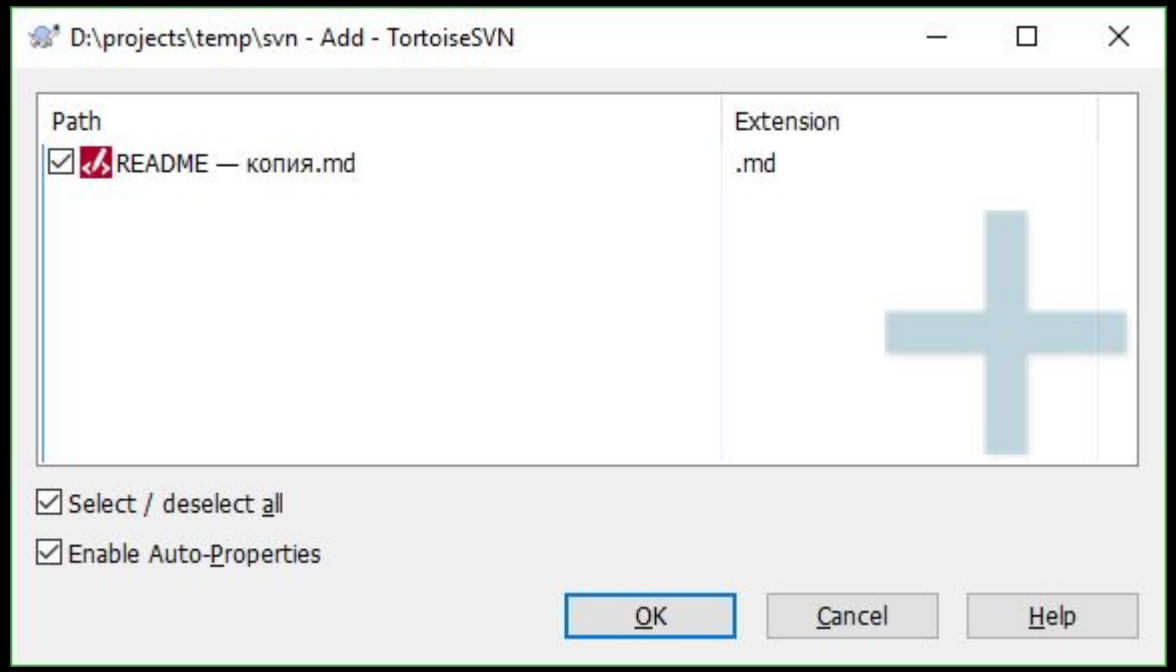
Git vs Subversion: добавление файлов

Git Add



SVN Add

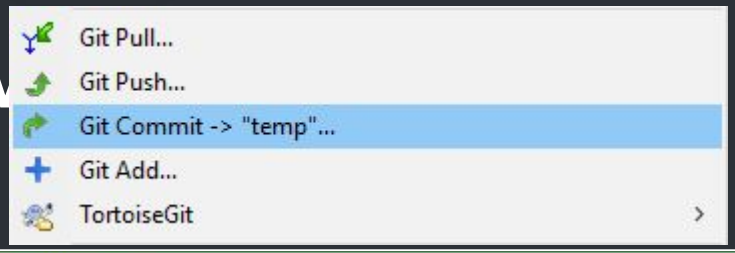




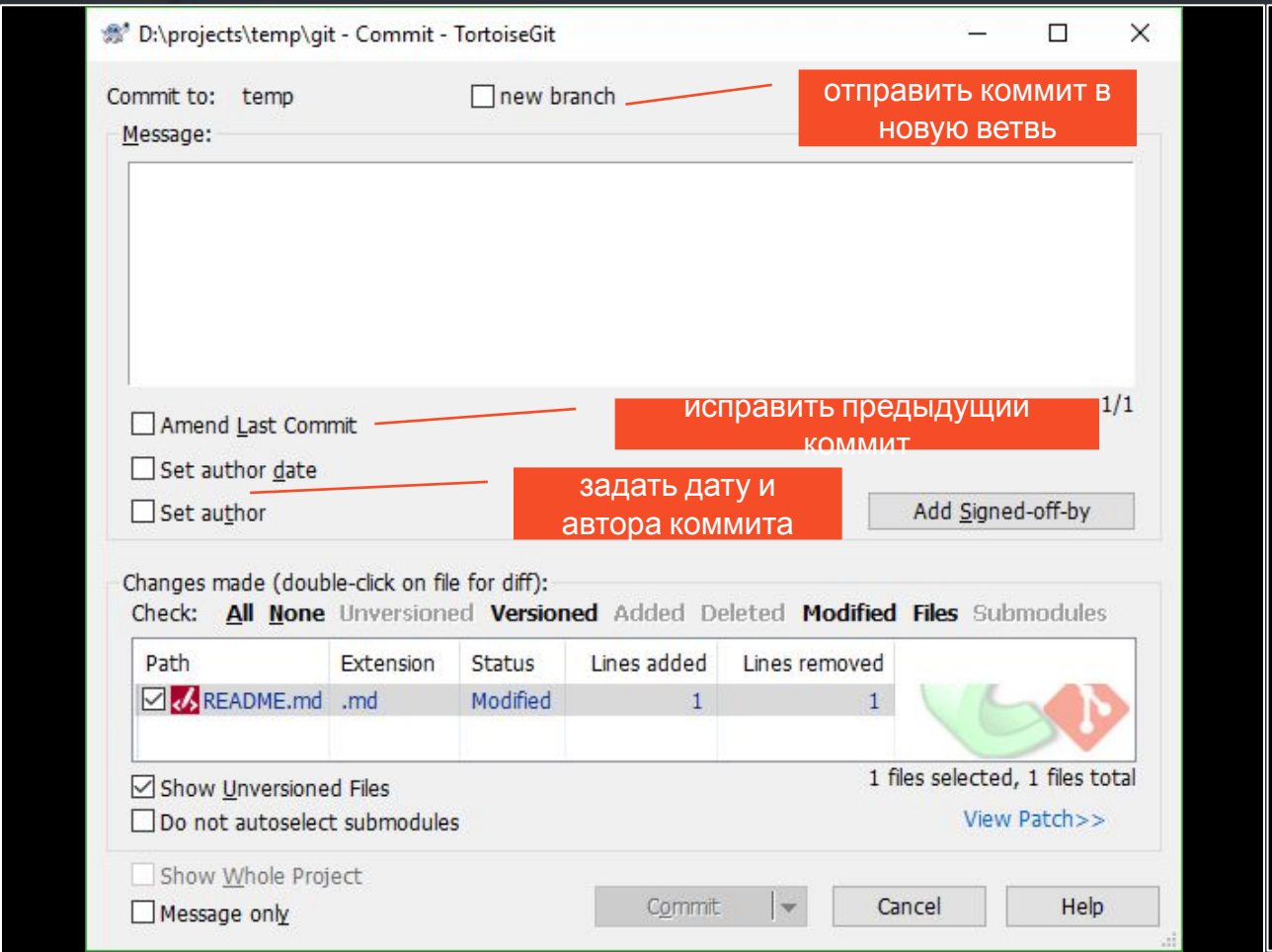
Git vs Subversion: фиксация

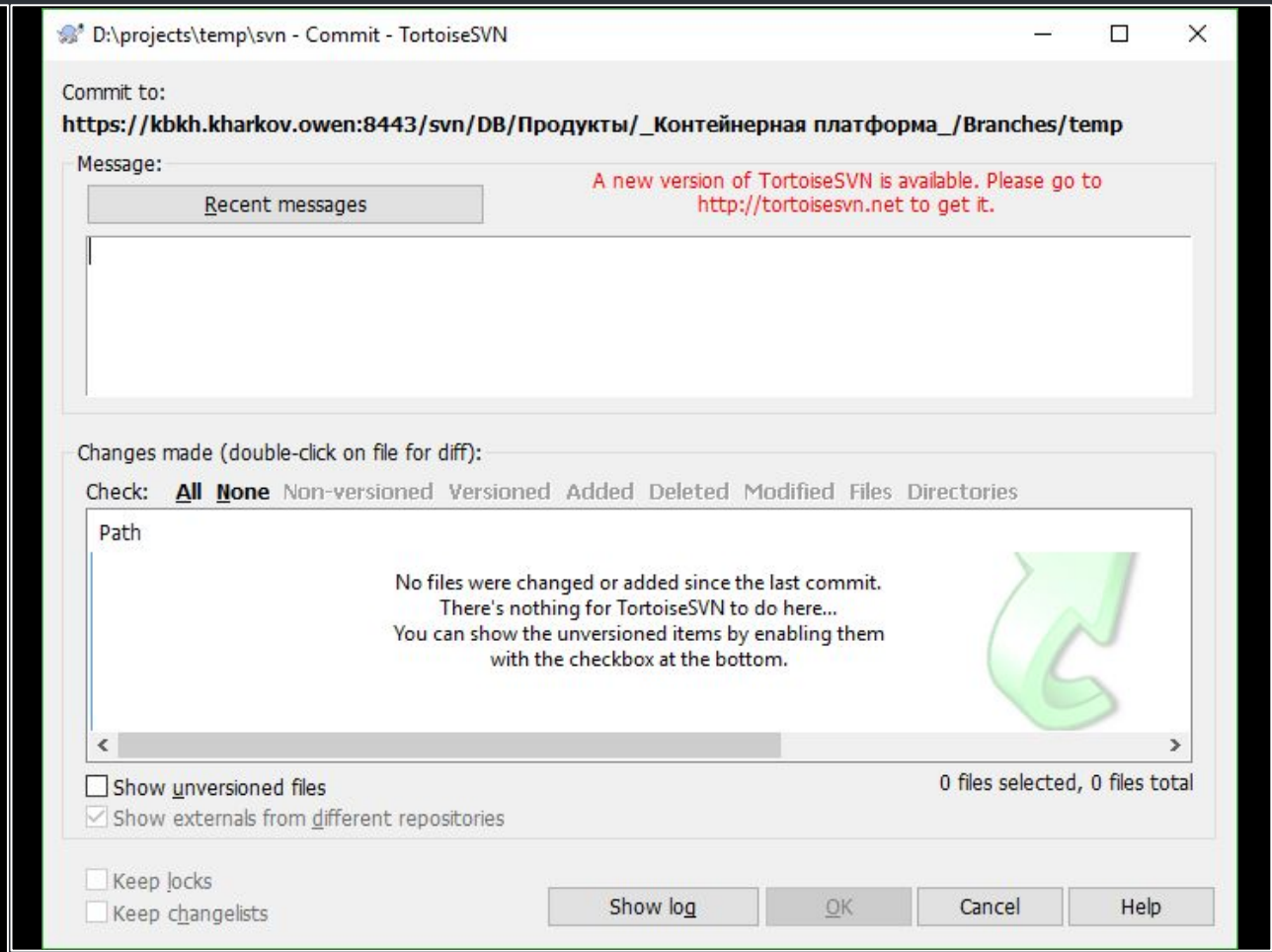
Git Commit

изм



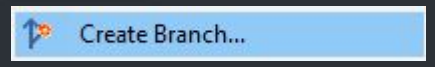
SVN Commit



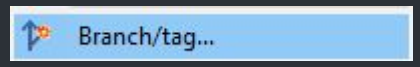


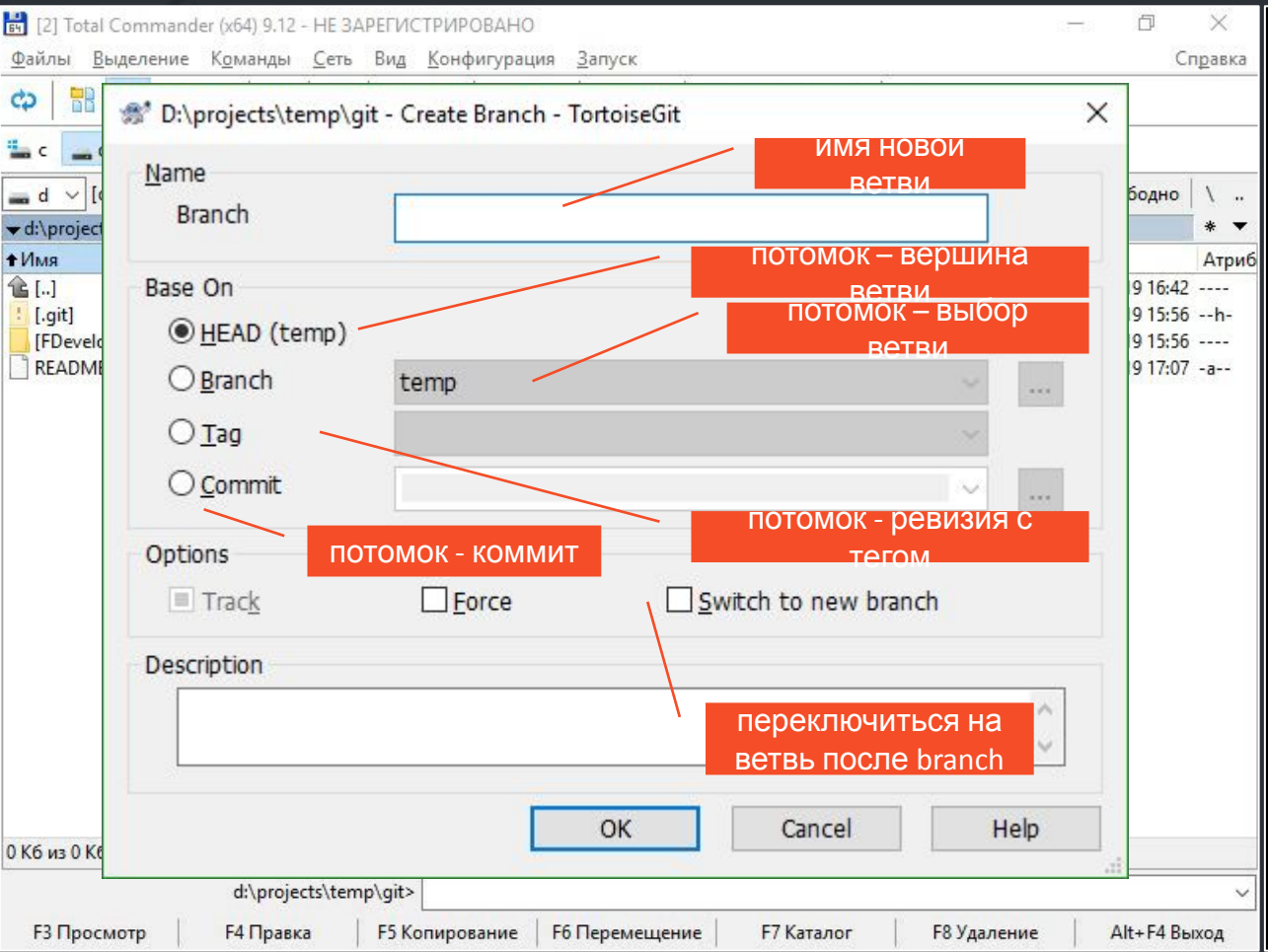
Git vs Subversion: создание веток

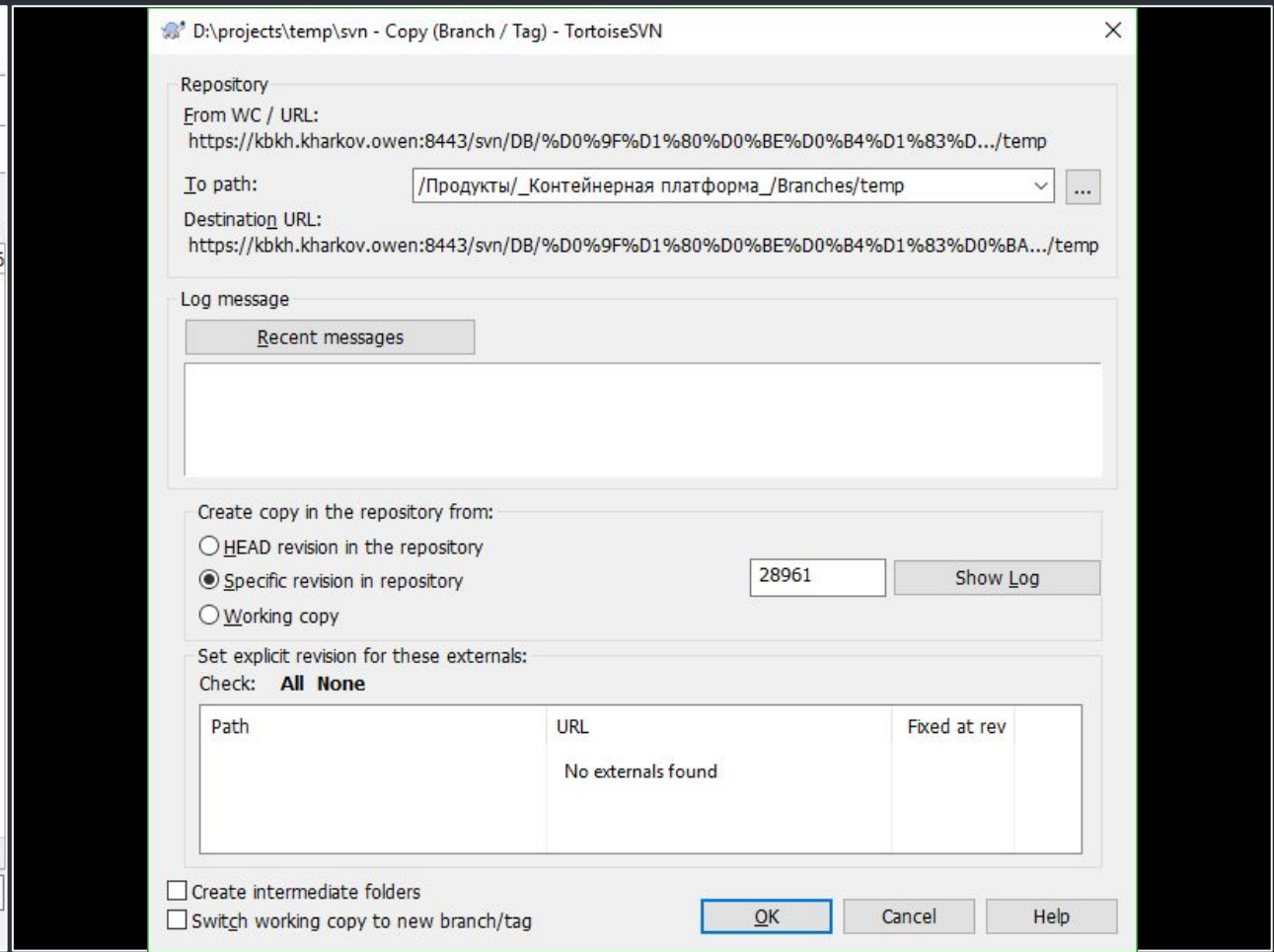
Git Branch



SVN Copy

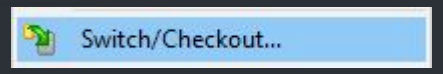




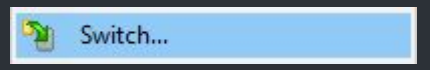


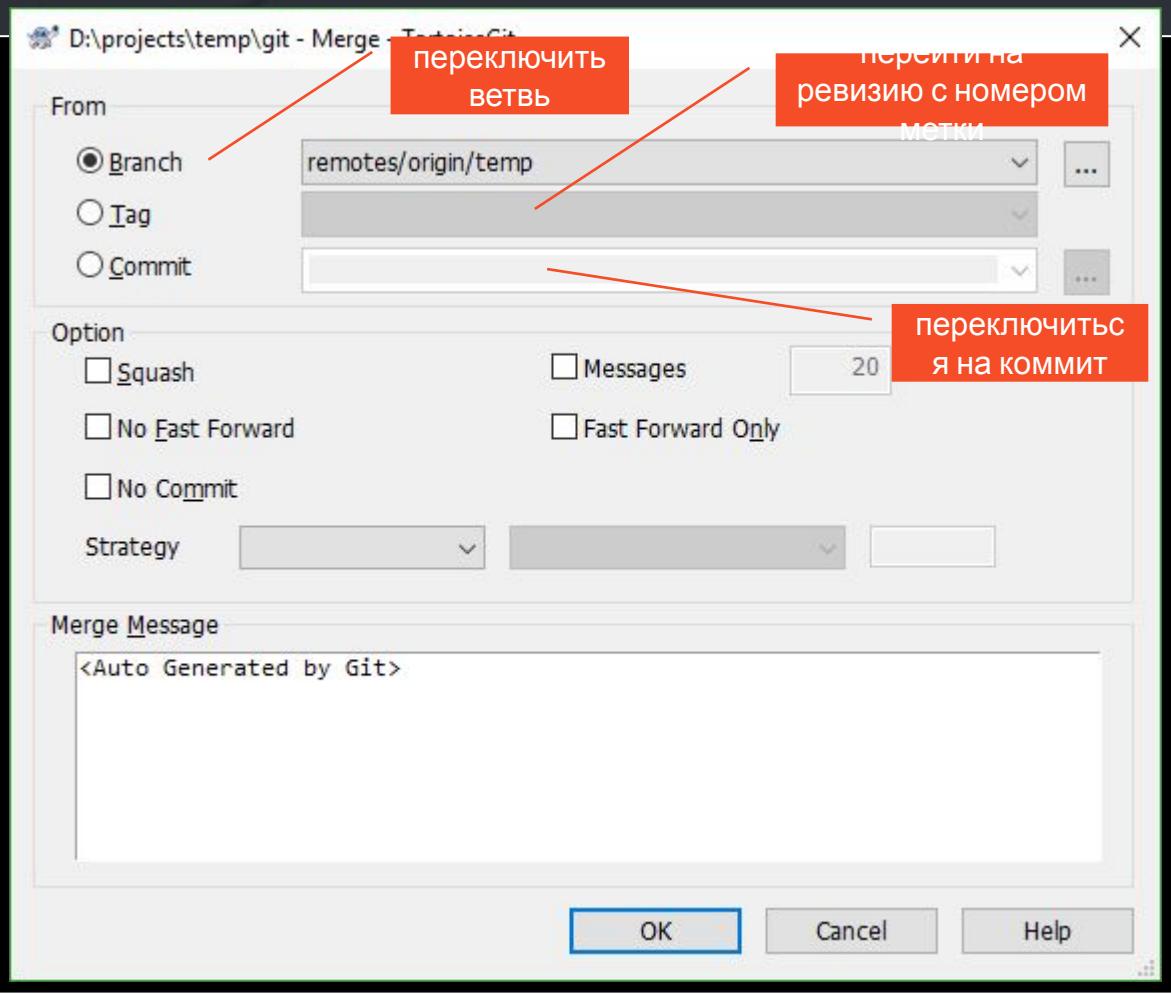
Git vs Subversion: переключение веток

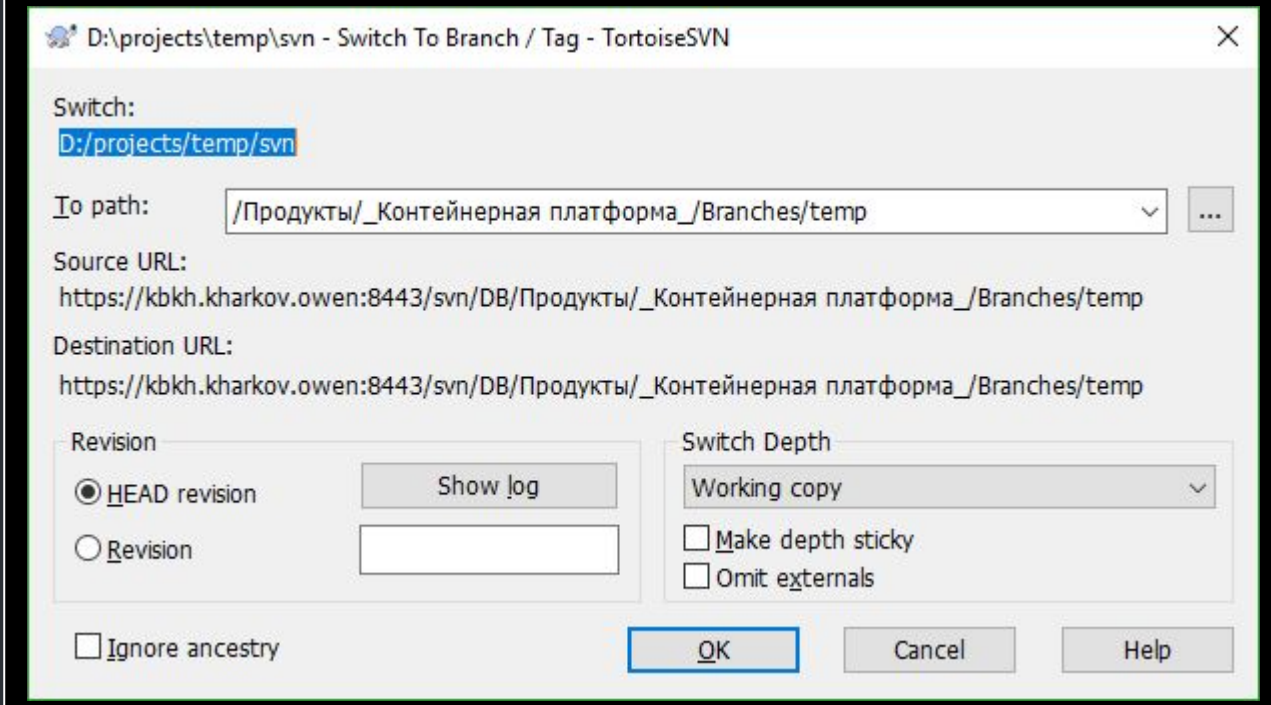
Git Checkout



SVN Switch







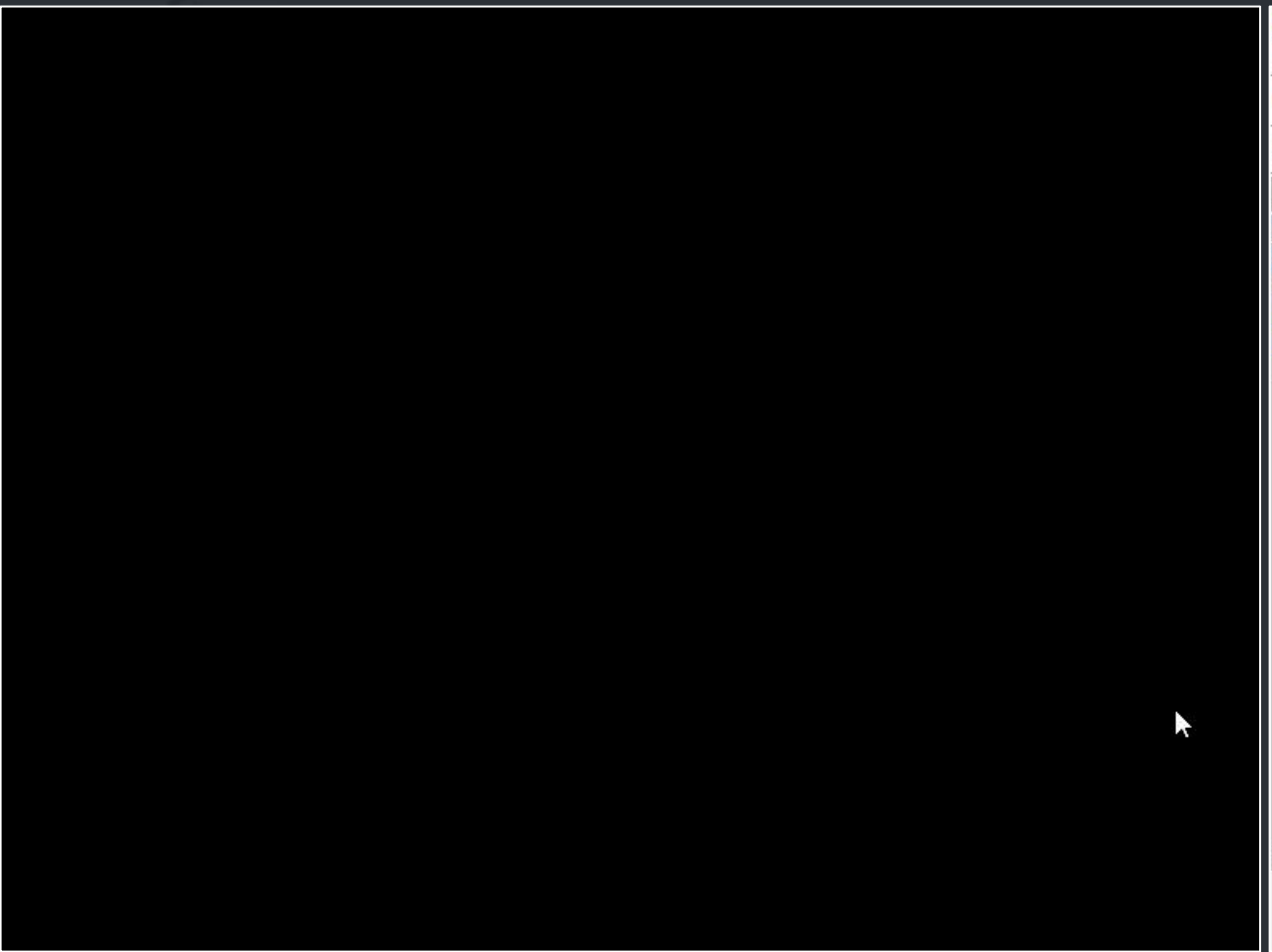
Git vs Subversion: просмотр истории

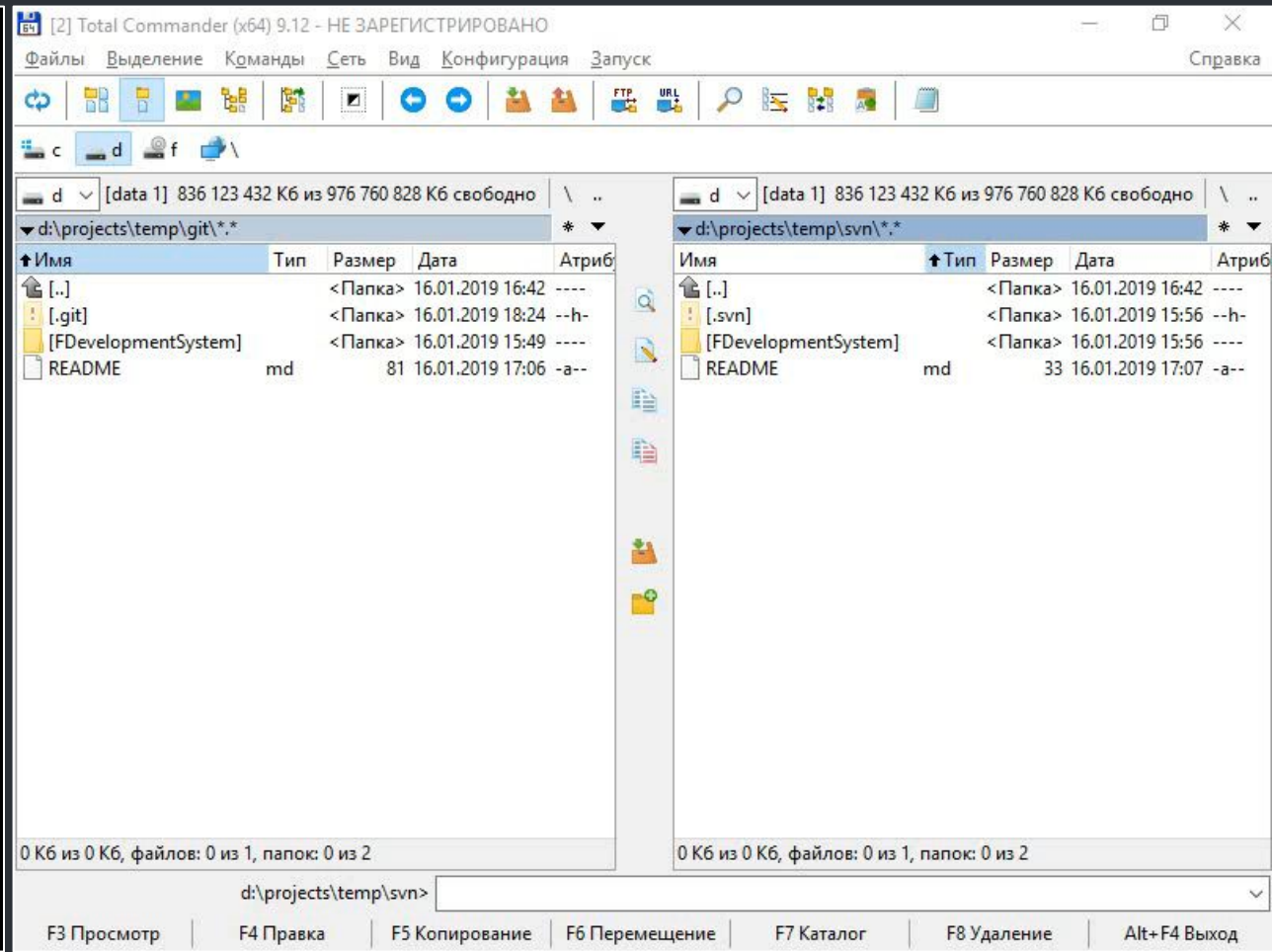
Git Log

Show log

SVN Log

Show log

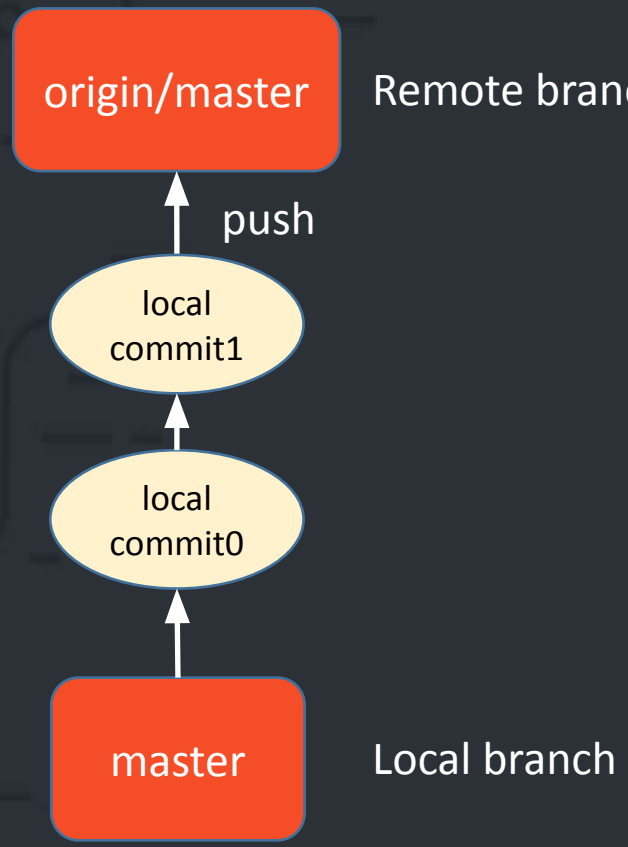


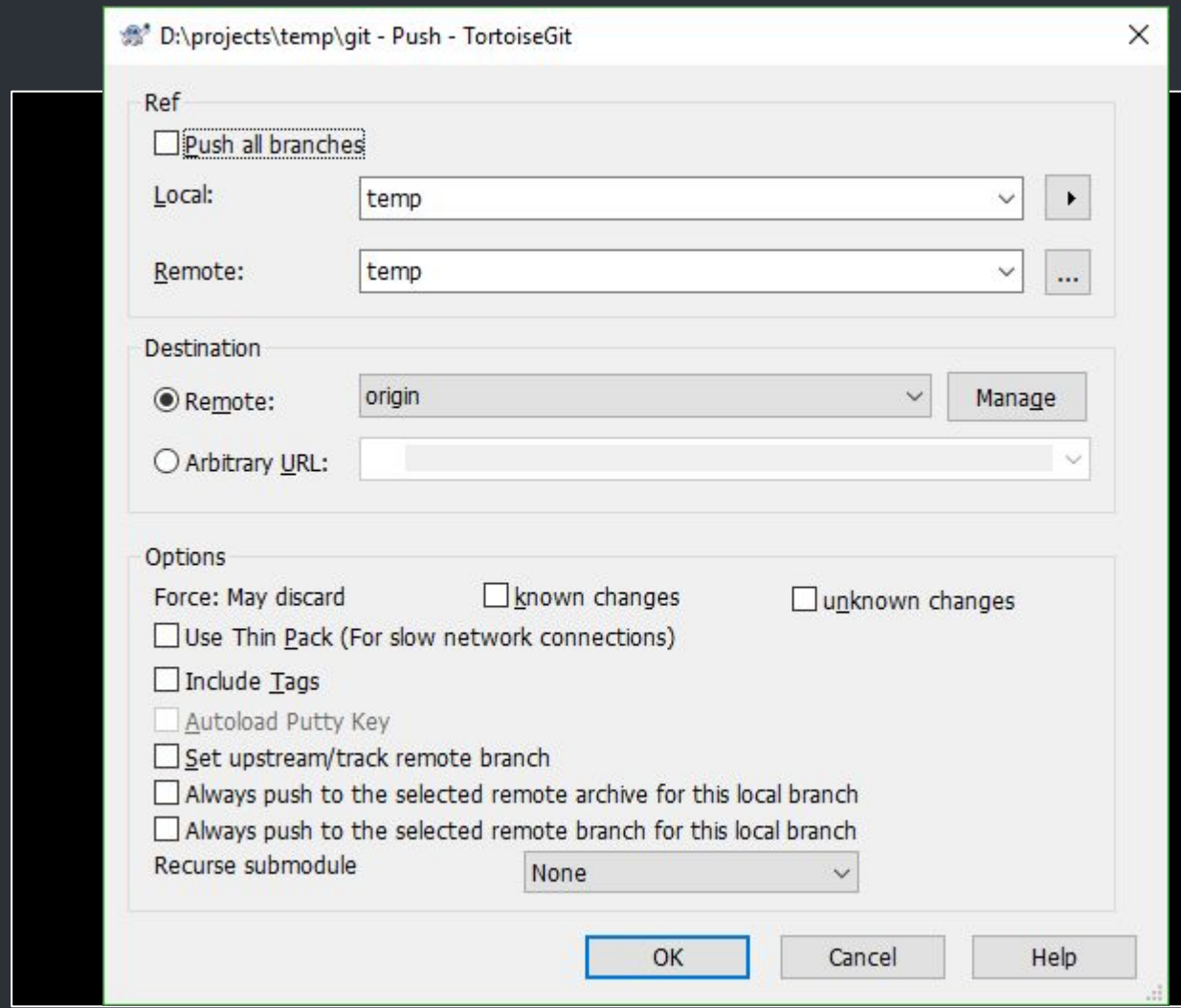


Git: обновление удаленного репозитария

Git Push синхронизирует локальные коммиты с удаленным (remote) сервером. URL-сервера назначается воспринимаемый alias – по умолчанию имя сервера origin.

Синтаксис: git push [сервер] [ветка]

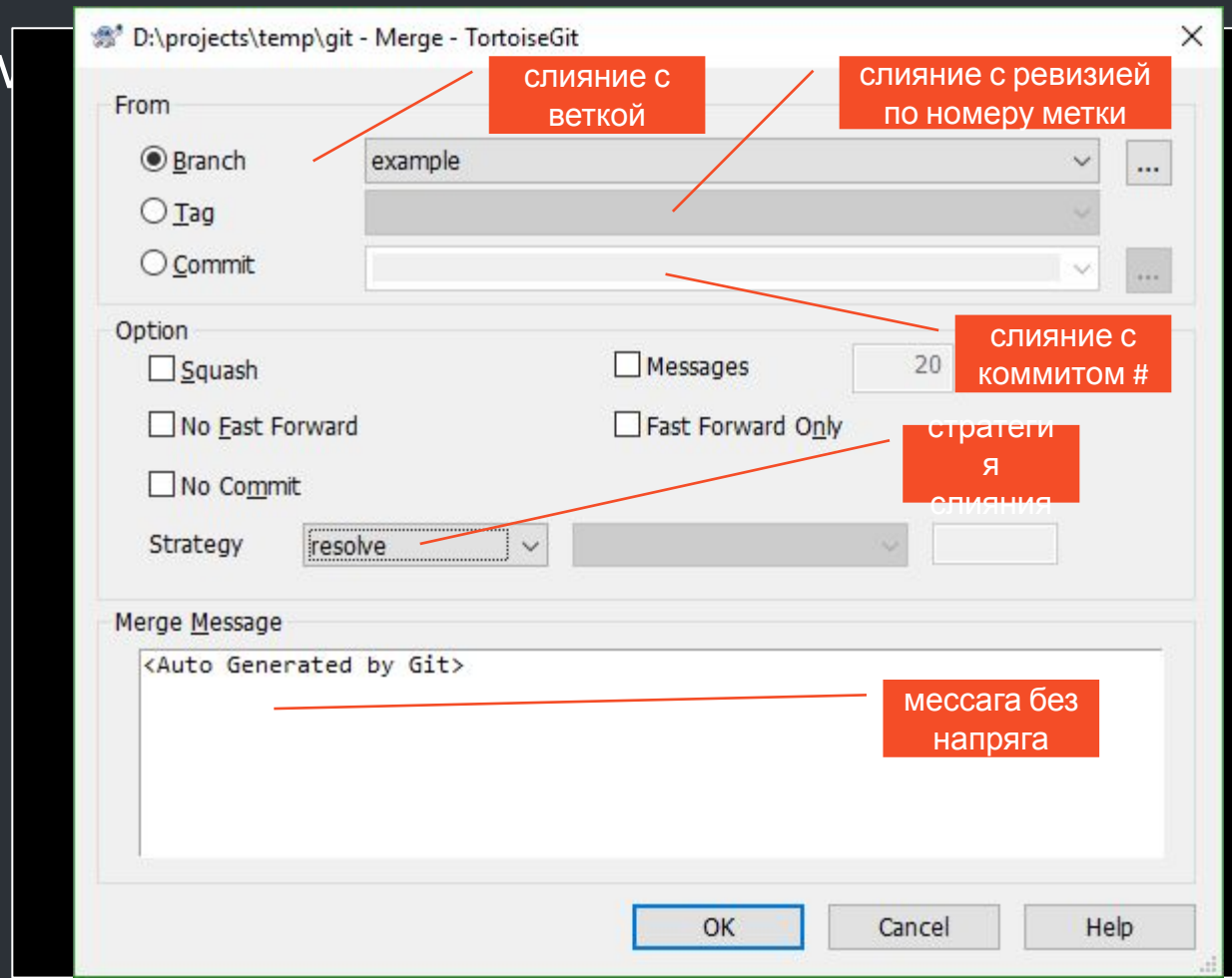




Git: слияние

Git Merge – слияние нескольких веток. Существуют различные стратегии слияния:

resolve – трехстороннее слияние с общим предком;
recursive - трехстороннее слияние с построением виртуального предка для сложных случаев resolve;
ours – слияние веток с игнорированием истории выбранной ветки;
subtree – слияние с помещением ветки в отдельный каталог (создание дерева проектов);
octopus – слияние более двух веток с двумя и более предками.



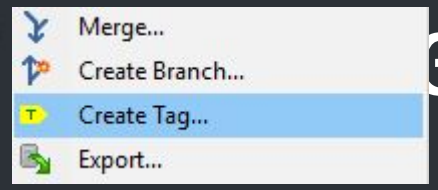
Git: слияние

При слиянии с созданием коммита ветвь перемещается вперед с присвоением уникального хэша, HEAD указывает на вершину.

Опции слияния:
Squash – объединить историю ветви в один коммит и поместить узел слияния до HEAD (HEAD не меняется);
No Fast Forward (перемотка вперед) – не переносить историю ветви в основной ствол при отсутствии в нем конфликтующих изменений, иными словами все равно сохранить историю в отдельной ветви;
No commit – сохранить результат слияния, но не фиксировать его, позволяя внести исправления перед коммитом;
Messages – создать сообщение о слиянии по истории указанного количества коммитов;
Fast Forward Only – попытаться провести слияние только методом перемотки вперед
с выражением наследованной ветви

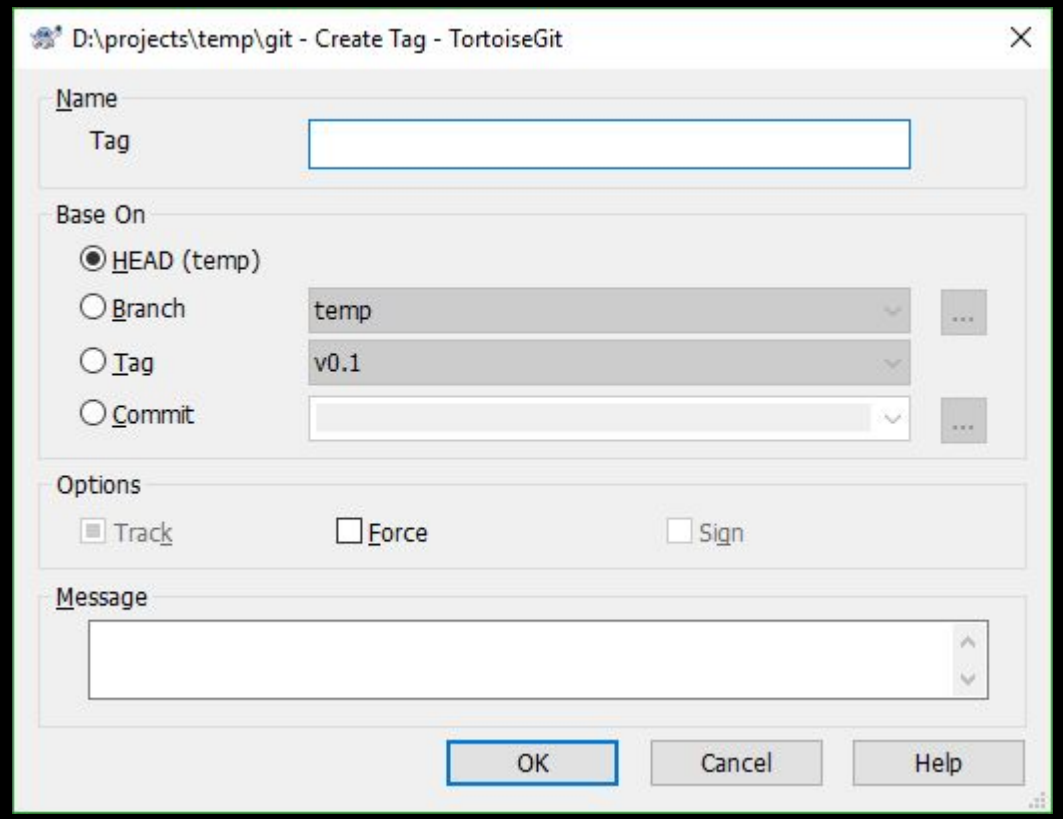
Git: создание меток

Git Tag



Метки создаются по:

- HEAD
- Ветке
- Существующему тегу
- Номеру ревизии



Git: полезности

- Git Init – создание локального репозитария
- Git Checkout – многозадачная команда (переключить/отменить)
- Git Stash – спрятать неиндексированные изменения с возможностью из последующего восстановления
- Git Submodule – работа с внешними репозиториями (аналог svn:externals)
- Git Status – состояние репозитария: наличие новых файлов, пометки удалено/изменено/конфликтует
- Git Fetch – синхронизация хранилища
- Gitignore – добавление по маске или конкретных файлов, не включаемых в индекс и не отображаемых по команде Status (все исключения хранятся в файлу .gitignore в корне репозитария)

Справка: https://git-scm.com/book/ru/v2