Оглавление

[Развёртывание проекта разработки 1С с использованием системы контроля версий Git 1](#_Toc476585664)

[1. Обзаводимся софтом 1](#_Toc476585665)

[Установка Git 1](#_Toc476585666)

[Первоначальная настройка Git 4](#_Toc476585667)

[Установка SourceTree 5](#_Toc476585668)

[Установка OneScript 8](#_Toc476585669)

[Установка дополнительных библиотек OneScript 9](#_Toc476585670)

[Скачивание библиотек 10](#_Toc476585671)

[Установка Gitsync 11](#_Toc476585672)

[2. Что нам стоит проект построить!? 11](#_Toc476585673)

[Создание и первоначальная настройка репозитория 11](#_Toc476585674)

[3. Готовим конфигурацию 1С и выгружаем её в Git 12](#_Toc476585675)

[Создание хранилища конфигурации 1С 12](#_Toc476585676)

[Синхронизация хранилища 1С с Git 13](#_Toc476585677)

[4. Между первым и вторым… (запускаем второй проект) 13](#_Toc476585678)

[5. А давайте синхронизироваться вместе! (пакетная синхронизация нескольких хранилищ) 14](#_Toc476585679)

[6. Призываем на помощь Precommit1C для работы с внешними отчётами 14](#_Toc476585680)

[7. Сборка конфигурации из исходных файлов с помощью packman 15](#_Toc476585681)

[8. Загрузка конфигурации в рабочий контур с помощью инструмента deployka 15](#_Toc476585682)

[Приложение. Наводим порядок в файловой системе 16](#_Toc476585683)

# Развёртывание проекта разработки 1С с использованием системы контроля версий Git

## 1. Обзаводимся софтом

Для запуска разработки конфигурации 1С с использованием Git нам потребуется:

* Собственно **Git**
* Любая клиентская оболочка для работы с Git. В принципе не обязательно, если предпочитаете работать с Git через командную строку. Я в своей работе пользуюсь **SourceTree**, поэтому далее буду описывать действия с его применением
* Свежая версия **OneScript**
* Дополнительная библиотека OneScript: **1commands**

### Установка Git

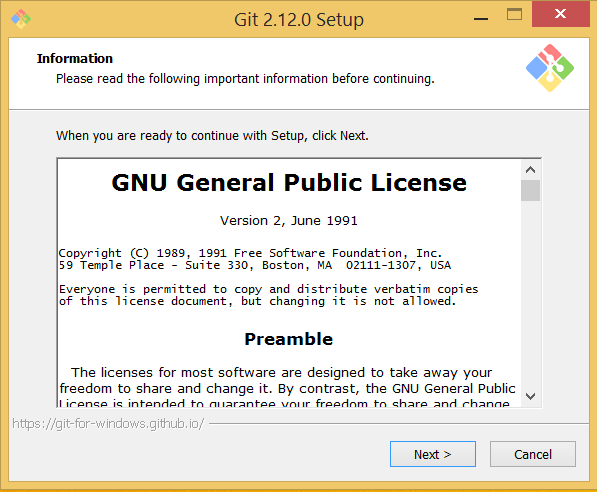
Скачиваем программу Git по одной из ссылок:

1. <https://githowto.com/ru>
2. <https://git-scm.com/>

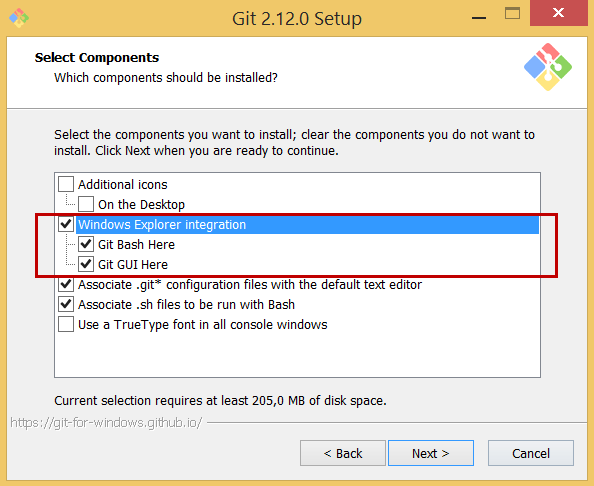
По первой ссылке также находится простой, доступный учебный курс по Git, благодаря которому можно быстро освоить работу с системой контроля версий. У меня прохождение всего курса заняло около 5 часов. По второй ссылке можно почитать более подробный учебник по Git.

После скачивания производим установку программы. Установку следует выполнять под правами администратора. Ниже описана последовательность шагов, актуальная для версии Git 2.12.0. При установке более поздних версий последовательность может отличаться.

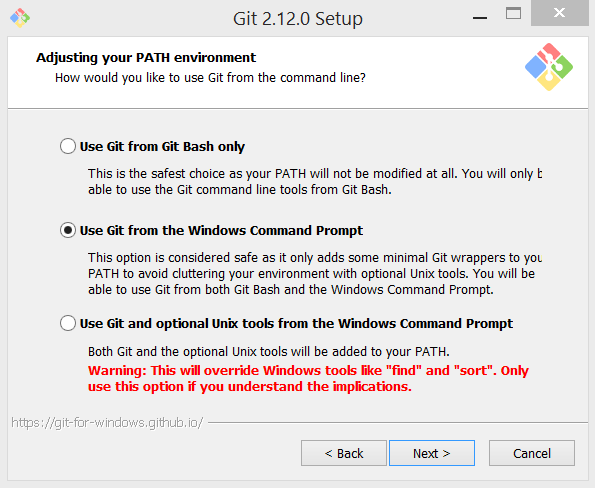
Шаг 1. Щелкаем Next.



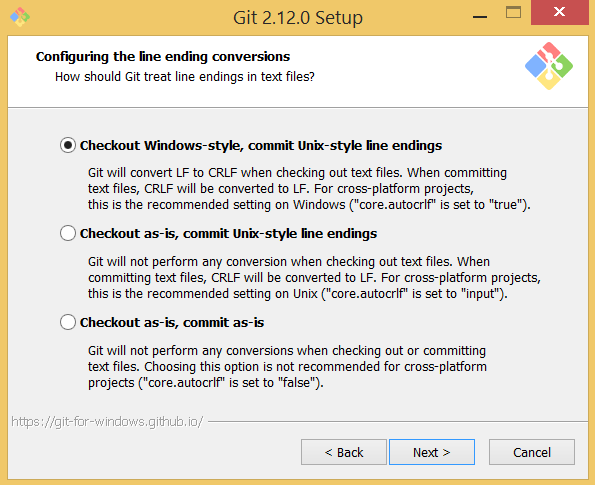
Шаг 2. Здесь можно оставить всё по умолчанию, но я снимаю флажки группы «Windows Explorer integration», так как лишние пункты контекстного меню во всех файловых папках мне ни к чему. (Правда, если вы планируете пользоваться ключами SSH, флажок «Git Bash Here» лучше оставить). После снятия флажков нажимаем Next.



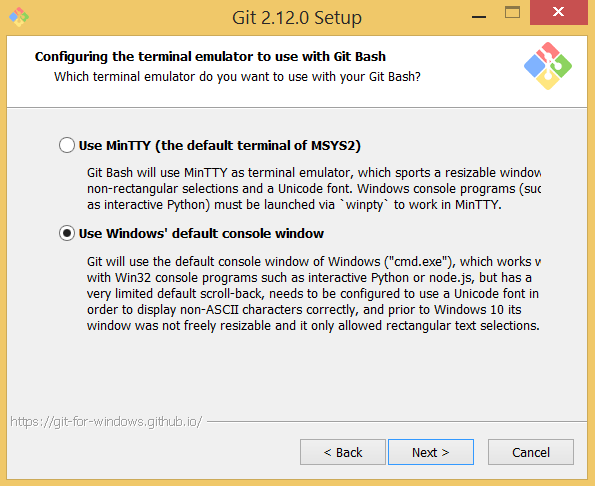
Шаг 3. Выбираем вариант «Use Git from the Windows Command Prompt» и нажимаем Next.



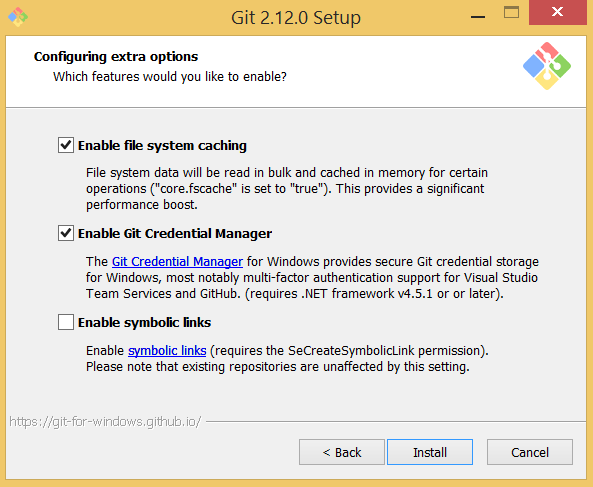
Шаг 4. Оставляем первый пункт, предложенный по умолчанию, и нажимаем Next. Правила коммита окончаний строк мы настроим после установки Git в его глобальных настройках.



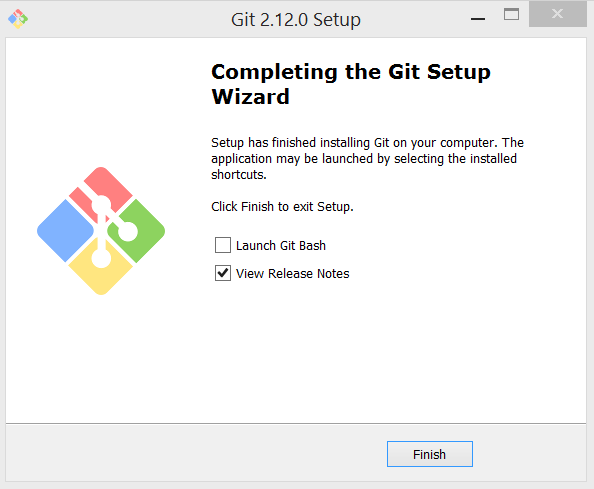
Шаг 5. Здесь необходимо выбрать вариант «Use Windows’ default console window», чтобы можно было использовать команды Git из командной строки. Нажимаем Next.



Шаг 6. Всё оставляем по умолчанию и жмём Install.



Завершение установки. Нажимаем Finish.



### Первоначальная настройка Git

Перед началом использования Git необходимо единожды выполнить некоторые глобальные настройки системы контроля версий. Все команды будем выполнять из командной строки.

**Внимание!**

Командную строку следует запускать под администратором, чтобы избежать ошибок прав доступа при выполнении некоторых команд.

Во-первых, зададим имя пользователя и адрес электронной почты, которыми будет идентифицироваться авторство коммитов в Git в локальном репозитории на текущей машине:

git config --global user.name "Your Name"

git config --global user.email "your\_email@whatever.com"

Во-вторых, установим настройки правил коммита окончаний строк, это в дальнейшем поможет избежать ошибок при коммите изменений в исходных файлах конфигураций 1С.

git config –-global core.quotepath false

git config –-global core.autocrlf false

git config –-global core.safecrlf false

**Внимание!**

Ключ core.autocrlf влияет на кроссплатформенные приложения. Поэтому, если есть необходимость коммитить с этой же машины код, который должен работать под Mac/Linux, необходимо проявлять осторожность.

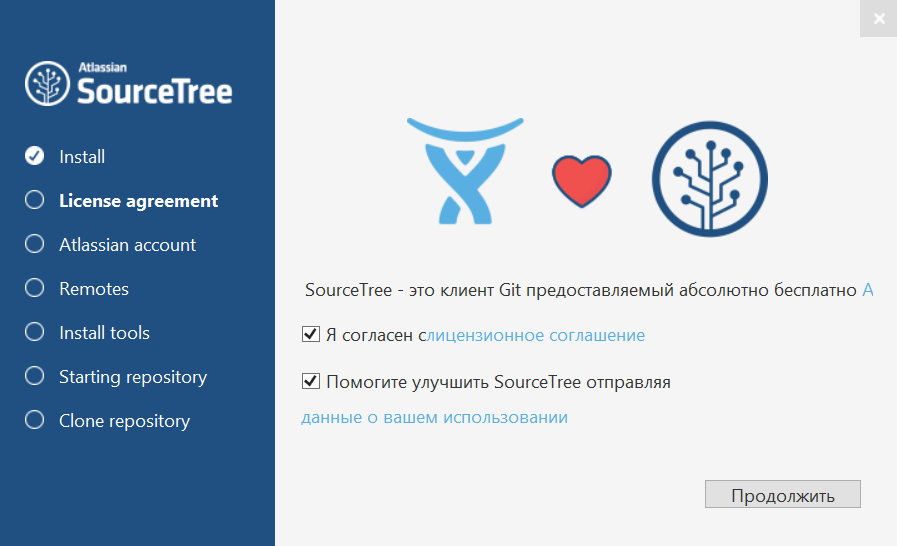
### Установка SourceTree

Скачиваем программу с официального сайта Atlassian:

<https://www.sourcetreeapp.com/>

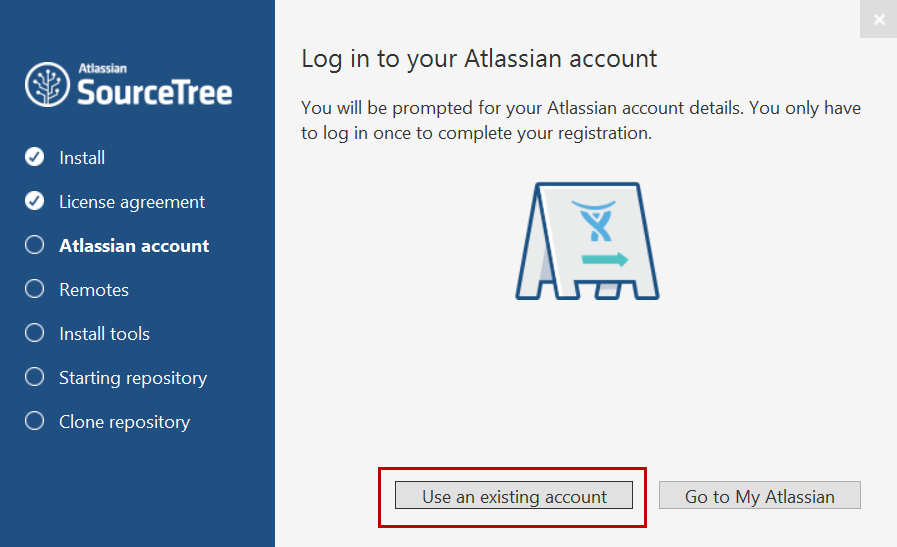
После скачивания производим установку программы. Установку следует выполнять под правами администратора. Ниже описана последовательность шагов, актуальная для версии SourceTree 1.10.20.1. При установке более поздних версий последовательность может отличаться.

Шаг 1. На первом шаге устанавливаем флажок о согласии с лицензионным соглашением и нажимаем «Продолжить».

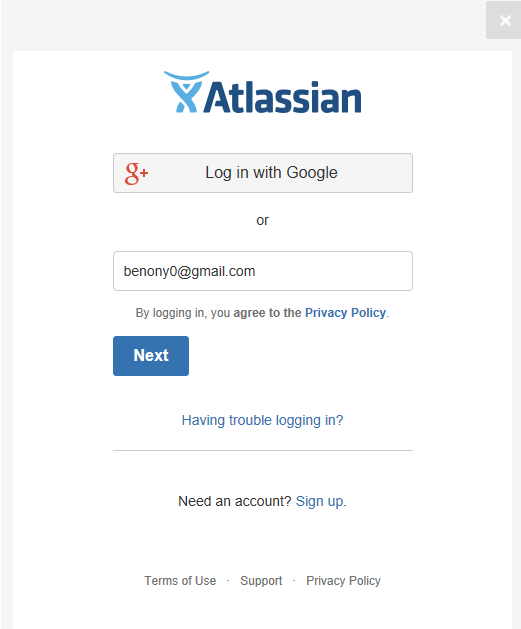


Шаг 2. На этом этапе SourceTree требует ассоциации с аккаунтом на BitBucket (<https://bitbucket.org/product>). Можно либо указать свой существующий аккаунт (должен быть зарегистрирован до установки программы), либо выбрать вариант регистрации нового аккаунта, тогда SourceTree поможет пройти необходимые шаги регистрации и произведет ассоциацию с новым аккаунтом. Далее рассматривается вариант, когда аккаунт на BitBucket уже зарегистрирован, и нам остаётся только к нему подключиться.

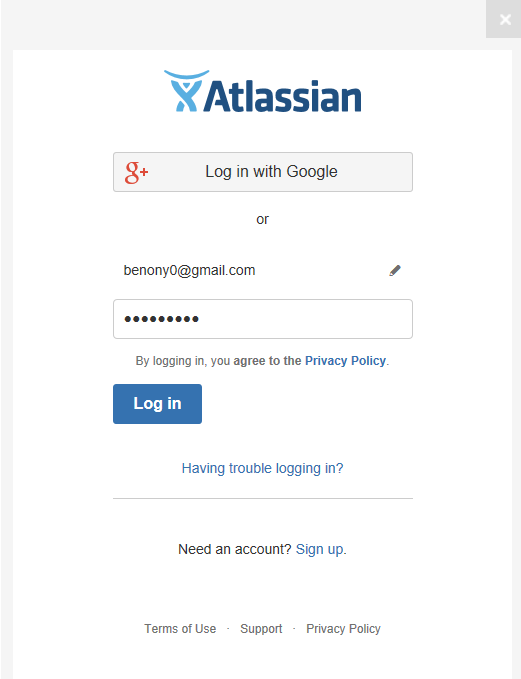
Нажимаем кнопку «Use an existing account».



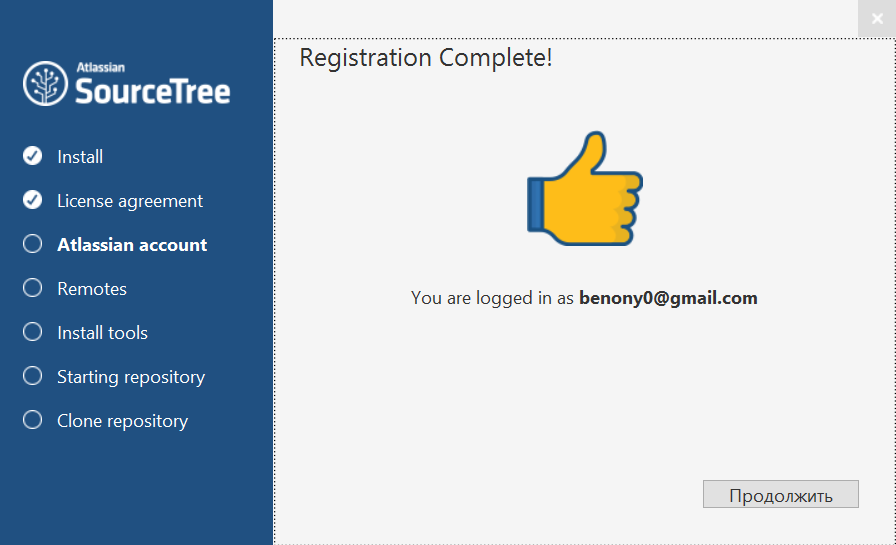
Откроется дополнительное окно для указания параметров авторизации. Вводим свой электронный адрес (который был задан при регистрации на BitBucket), нажимаем Next.



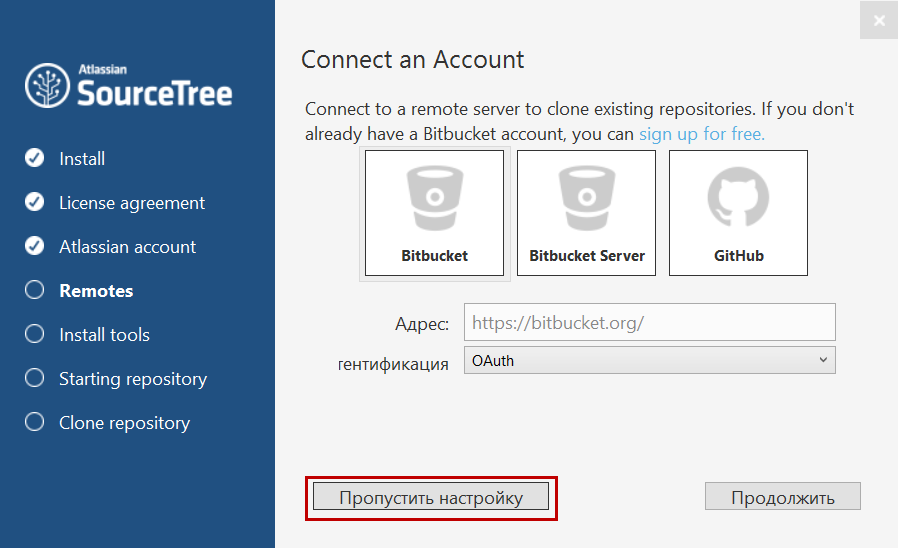
Далее вводим пароль и нажимаем Log in.



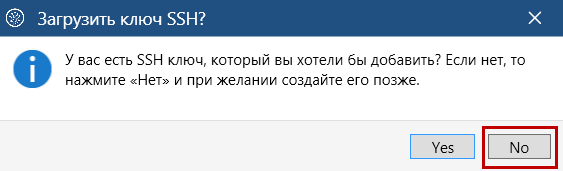
После проверки авторизации появится окно завершения шага ассоциации. Нажимаем «Продолжить».



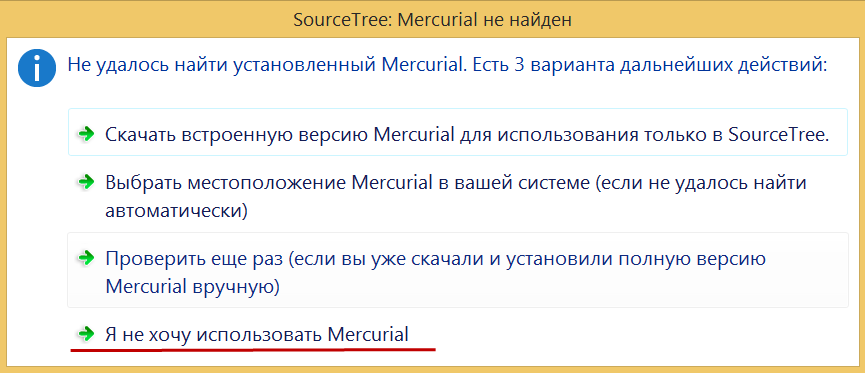
Шаг 3. Отказываемся от дополнительных настроек и нажимаем кнопку «Пропустить настройку».



На запрос об использовании ключа SSH отвечаем No (Нет).



На запрос о выборе варианта работы с Mercurial отвечаем «Я не хочу использовать Mercurial» (поскольку мы будем использовать Git).



После этого установка будет завершена, и откроется окно приложения SourceTree. Установка завершена.

**Немного про ключ SSH**

В своей папке пользователя в винде делаете и заходите в папку .ssh

Правая кнопка мыши -> Git Bash Here. В консоли набирайте: ssh-keygen -t rsa Три раза ентер (имя файла и пароль вводить НЕ НАДО)

В папке C:\Users\%Username%\.ssh появится пара файлов с именами id\_rsa и id\_rsa.pub

Файл, который называется id\_rsa.pub – это публичный SSH ключ, который можно залить в свой профиль на гитхаб/гитлаб/битбакет или локальный сервер-хранилище Git (тот же гитлаб дает такое), и который так же ассоциирует тебя, как конкретного пользователя. Второй файл не надо отправлять никому, кроме самого себя на другой свой компьютер, с которого вы собираетесь комиттить"

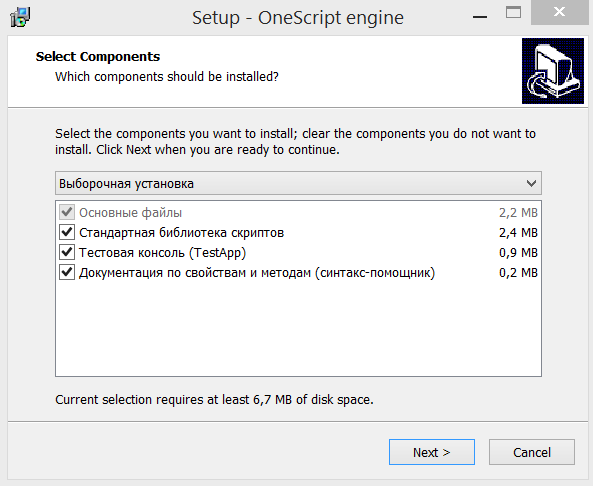
### Установка OneScript

Скачиваем программу по ссылке:

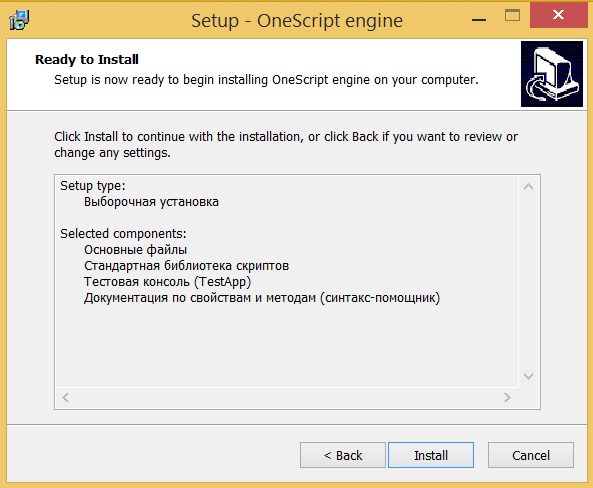
<http://oscript.io/>

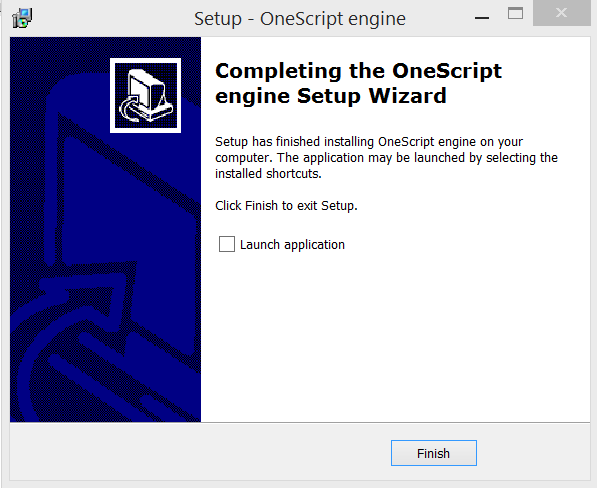
После скачивания производим установку программы. Установку следует выполнять под правами администратора. Ниже описана последовательность шагов, актуальная для версии OneScript 1.0.15. При установке более поздних версий последовательность может отличаться.

Шаг 1. Выбираем необходимые компоненты для установки. Обязательно выбираем первые два, остальное на усмотрение. Я себе ставлю всё, чтобы использовать возможности библиотек по максимуму. Нажимаем Next.



Шаг 2. Нажимаем Install, затем Finish, после чего установка завершена.



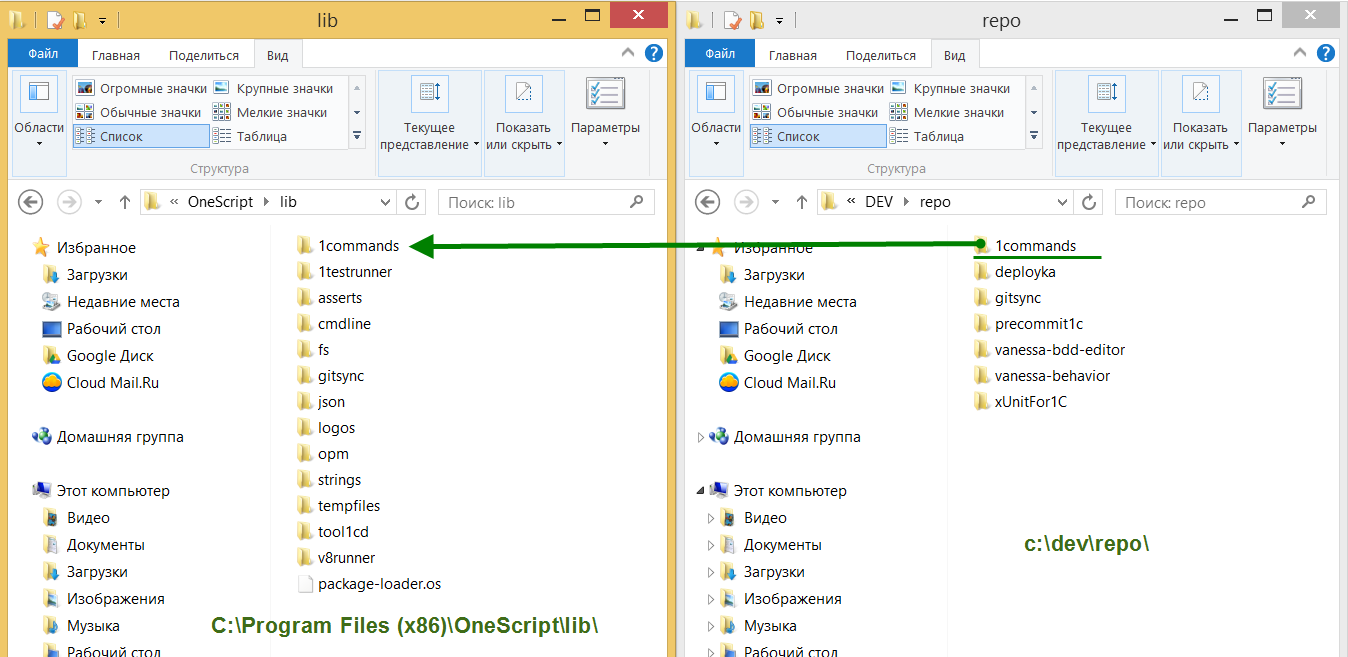


### Установка дополнительных библиотек OneScript

Для работы gitsync необходимо наличие библиотеки 1commands. Установку библиотек выполняет пакетный менеджер opm, входящий в поставку OneScript:

opm install 1commands

Если установка из командной строки прошла неудачно, библиотеку можно установить вручную. Для этого достаточно скопировать репозиторий 1commands в папку lib в каталоге с установленной программой OneScript. В следующей главе описывается процесс установки репозитория на локальный компьютер.

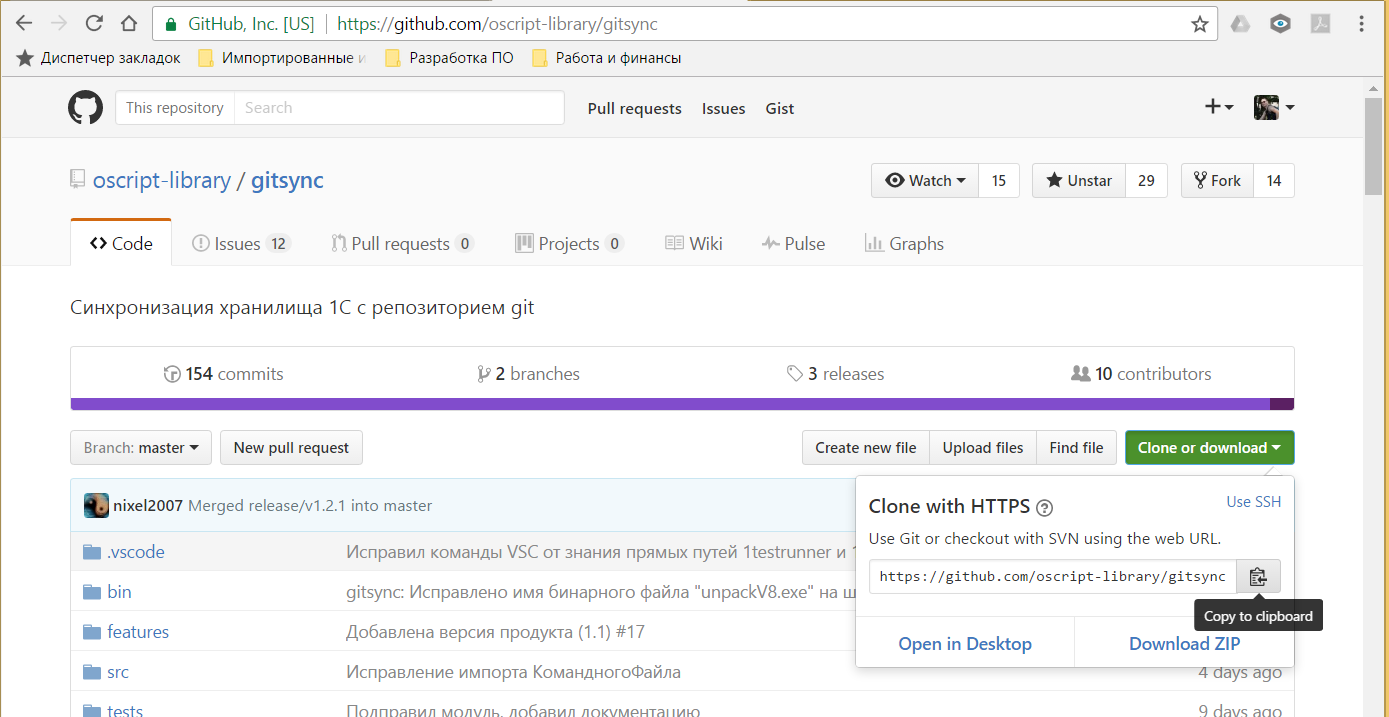


### Скачивание библиотек

Находим на Github сборник библиотек oscript-library, и скачиваем к себе репозиторий 1commands. Я предпочитаю хранить у себя клон репозитория в SourceTree, но можно просто скопировать zip-архив текущей версии.

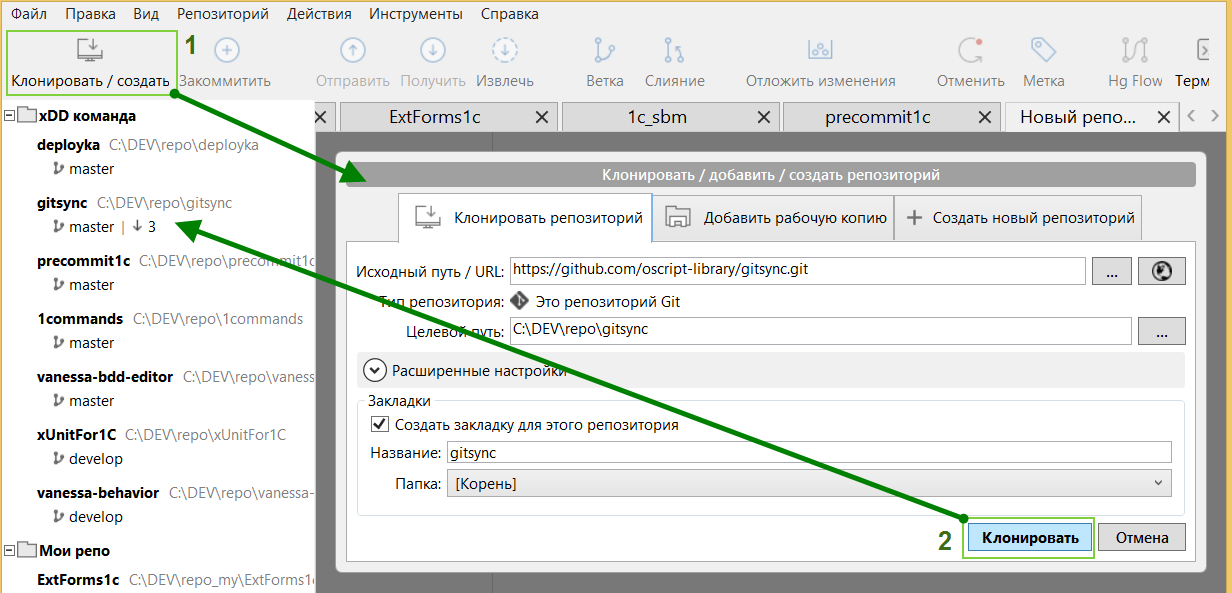
#### Как склонировать репозиторий с Github в SourceTree

На странице gitsync на Гитхабе нажимаем кнопку «Clone or download» и копируем URL репозитория с помощью кнопки справа от текстовой ссылки.



Затем, в SourceTree:

1. Нажимаем кнопку «Клонировать / создать» для создания нового репозитория.   
   На закладке «Клонировать репозиторий» вставляем из буфера скопированный URL, указываем целевой путь (где будет лежать репозиторий на локальной машине), название (если название по умолчанию не устраивает), и нажимаем кнопку «Клонировать».
2. После того, как программа склонирует репозиторий с Гитхаба (это займёт некоторое время), в списке репозиториев SourceTree появится новый пункт, а в целевом каталоге – файлы репозитория.



### Установка Gitsync

Устанавливаем Gitsync пакетным менеджером:

opm install gitsync

И проверяем успешность установки. Gitsync должен запускаться из командной строки:

gitsync help

#### Если что-то пошло не так

Если тест Гитсинха прошёл неудачно, необходимо проделать следующее:

1. Убедиться, что установлена свежая версия OneScript, при необходимости обновить
2. Обновить пакетный менеджер и библиотеки oscript:

opm update opm

opm update -all

## 2. Что нам стоит проект построить!?

В данном разделе описан процесс создания нового проекта разработки «с нуля». Если репозиторий уже существует и необходимо только подключиться к команде его разработчиков, достаточно [склонировать репо](#_Как_склонировать_репозиторий)зиторий и перейти к [следующему разделу](#_3._Подготовка_конфигурации).

### Создание и первоначальная настройка репозитория

1. Создаём репозиторий на удалённом хостинге, где собираемся его хранить. Это может быть Github, Gitlab, Bitbucket или что-то еще.
2. Клонируем репозиторий на рабочую машину.
3. Настраиваем файлы версий и пользователей в каталоге конфигурации. Вот тут немного остановимся:

Хорошим тоном при работе над проектом является хранение в одном репозитории не только исходных файлов самой разрабатываемой конфигурации, но и всех файлов, связанных с этим проектом (внешние отчёты и обработки, правила обмена, инструкции, тесты и т.д.). Поэтому сразу после клонирования репозитория, создаём в нём каталог config (или cfg, кому как удобно), в котором будем хранить исходники конфигурации.

Дополнительно следует создать два служебных файла: VERSION и AUTHORS. Первый создаём автоматически командой к Гитсинху:

gitsync set-version C:\DEV\repo\_ga\1c\_sbm\config 0

Здесь указанный каталог в первом параметре – это каталог для исходников конфигурации, который мы создали на предыдущем шаге внутри каталога репозитория проекта. Ноль во втором параметре означает, что первоначально мы выгружаем всё хранилище конфигурации, со всей историей изменений. После выполнения команды содержимое файла будет иметь вид xml с одним узлом. Значение в узле означает номер текущей версии хранилища, которое хранится в Git.



Второй файл AUTHORS необходимо создать вручную. Он представляет собой текстовый файл без расширения, и содержит в себе перечень соответствий имён пользователей хранилища конфигурации 1С именам пользователей Git, от имени которых будут производиться коммиты.



**Рекомендация**

Для работы с файлами проекта рекомендуется использовать универсальный «всеядный» редактор, например **Notepad++** (<https://notepad-plus-plus.org/>), или **Visual Studio Code** (<https://code.visualstudio.com/>). Описание работы с редактором в настоящей инструкции не предусмотрено.

**Внимание!**

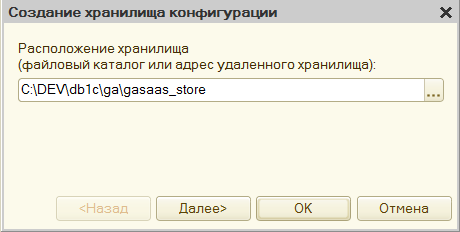
При редактировании русскоязычного текста, помещаемого в Git, необходимо использовать кодировку UTF-8

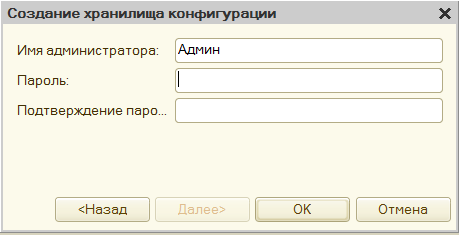
## 3. Готовим конфигурацию 1С и выгружаем её в Git

Gitsync не работает напрямую с конфигурацией 1С, а оперирует хранилищем кофигурации. Поэтому следующим шагом создаём локальное хранилище 1С, с которым будет связана разрабатываемая нами конфигурация.

### Создание хранилища конфигурации 1С

В режиме конфигуратора выбираем пункт меню «Конфигурация – Хранилище конфигурации – Создать хранилище». На первом шаге указываем каталог, в котором будет находиться хранилище, на втором – имя пользователя хранилища и его пароль, и нажимаем кнопку «ОК». Дожидаемся создания хранилища, в конце на запрос подключиться к нему отвечаем утвердительно.





### Синхронизация хранилища 1С с Git

Теперь осталось выгрузить конфигурацию из хранилища в Git:

cd c:\dev

gitsync export db1c\ga\gasaas\_store repo\_ga\gasaas\config

Это основная команда, которая будет использоваться на регулярной основе для синхронизации хранилища 1С в Git. При ее выполнении gitsynс производит коммиты в git по каждой версии хранилища, с указанием автора коммита из сопоставленного имени пользователя хранилища 1С из файла AUTHORS.

Если что-то осталось непонятным, справку о работе gitsync можно посмотреть на странице Гитхаба: <https://github.com/oscript-library/gitsync>.

## 4. Между первым и вторым… (запускаем второй проект)

После того, как вся инфраструктура подготовлена, убедимся на примере, что развернуть новый проект 1С в Git совсем не сложно. А заодно рассмотрим еще один способ, который предоставляет gitsync для разворачивания проекта.

1. [Создаём хранилище конфигурации](#_Создание_хранилища_конфигурации) и подключаемся к нему;
2. Создаём репозиторий проекта на Gitlab, [клонируем его локальную копию](#_Как_склонировать_репозиторий);
3. Заводим каталог cfg, и с помощью команды gitsync init выполняем первоначальную настройку:

gitsync init db1c\my\lbpchs\_store repo\_my\lbpchs\cfg

После этого в каталоге cfg для исходных файлов конфигурации будут созданы и заполнены файлы AUTHORS и VERSION, при этом в файле AUTORS будут вписаны имена всех пользователей, определённых в хранилище конфигурации:



Останется только правильно прописать имена и электронные адреса пользователей, от имени которых будут производиться соответствующие коммиты в git:



1. Инициализируем первоначальную версию хранилища:

gitsync set-version repo\_my\lbpchs\cfg 0

1. Выгружаем конфигурацию в git:

gitsync export db1c\my\lbpchs\_store repo\_my\lbpchs\cfg

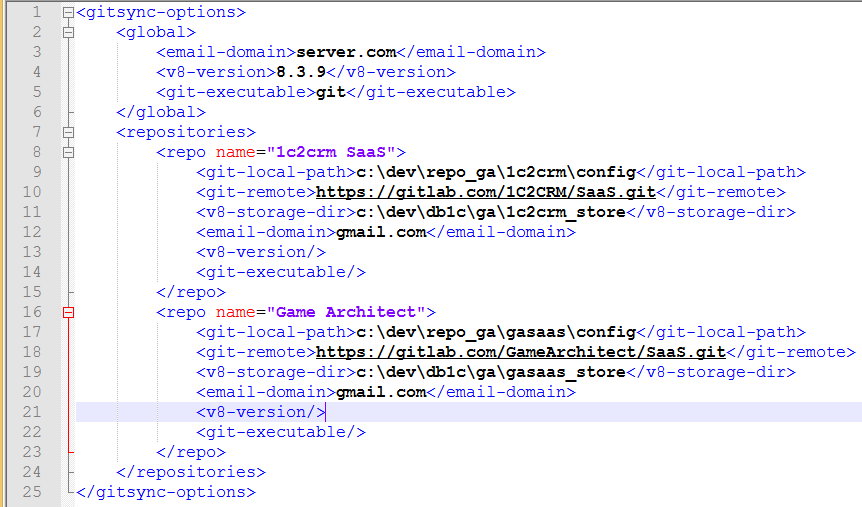
1. Отправляем новые коммиты в удалённый репозиторий

## 5. А давайте синхронизироваться вместе! (пакетная синхронизация нескольких хранилищ)

Теперь, когда у нас в работе находится одновременно два проекта, было бы здорово выгружать все изменения двух конфигураций в Git одним нажатием. Gitsync и тут готов нам помочь:

gitsync all c:\dev\db1c\configs.xml

В качестве параметра команде all передаётся путь к файлу xml с описанием всех конфигураций, которые необходимо синхронизировать в Git «одним пакетом». Пример файла и описание его структуры находится в каталоге репозитория самого gitsync, и называется **config-example.xml**. Готовый же файл для наших двух проектов выглядит так:



## 6. Призываем на помощь Precommit1C для работы с внешними отчётами

Этот инструмент помогает автоматизировать процесс разборки/сборки внешних отчётов/обработок 1С из исходных файлов в репозиторий Git. Исходные файлы по умолчанию складываются в каталог src репозитория. Синхронизация отчёта/обработки с git происходит автоматически в момент исполнения команды git commit.

Установка утилиты precommit1c на локальный компьютер:

opm install precommit1c

Установка precommit1c в целевой репозиторий (установка производится в текущий каталог):

cd c:\dev\repo\_ga\1c2crm

precommit1c --install

Precommit1c готом в использованию.

Подробная видео-инструкция по использованию Precommit1C представлена в первых 32 минутах вебинара:

<http://infostart.ru/webinars/521983/>

**Внимание!**

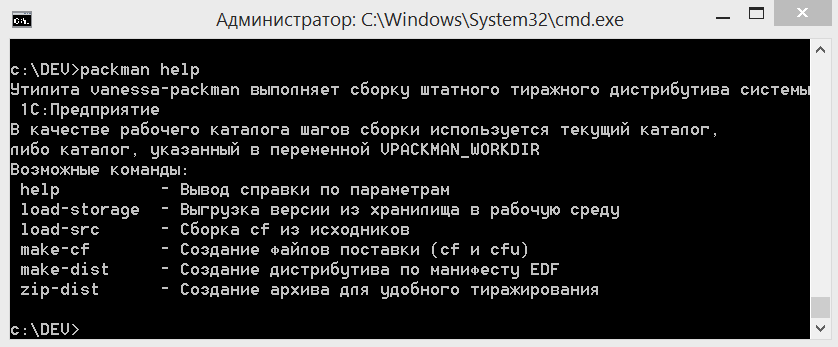
В последних версиях платформы 8.3.9 (начиная с 8.3.9.2033), при разборе внешней обработки на исходники, появляется предупреждение об ограничении безопасности, описанное здесь: <http://forum.infostart.ru/forum86/topic163993/>. Описанные методы лечения не сработают, поскольку открываемая база является временной и создаётся "на лету" в момент коммита. На платформе 8.3.9.1850 ошибки нет. Обещают устранить в 8.3.10.

## 7. Сборка конфигурации из исходных файлов с помощью packman

Для сборки конфигурации из исходных файлов используется инструмент автоматизации packman. Для начала установим его и проверим на работоспособность:

opm install packman

packman help



По умолчанию конфигурация собирается в текущий каталог, поэтому сначала переключаемся в удобное место, после чего даём команду на сборку:

cd c:\dev

packman load-src repo\_my\lbpchs\cfg

После выполнения команды, в рабочем каталоге **c:\dev** будет создан подкаталог **.packman\v8r\_TempDB** с временной информационной базой, собранной из исходников. Далее даём команду на создание файла конфигурации этой базы:

packman make-cf

Теперь в подкаталоге **.packman** находится файл конфигурации **1cv8.cf**, который можно использовать для обновления рабочего контура (или любого другого контура).

## 8. Загрузка конфигурации в рабочий контур с помощью инструмента deployka

**Внимание!**

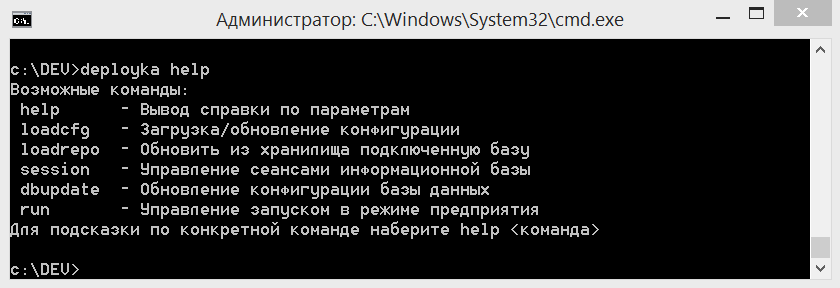
На текущий момент (27.02.2017) в работе Деплойки существует известная ошибка, исправление которой еще жд ём. Описанная ниже последовательность действий будет работать после её устранения, пока же полученную \*.CF-ку придётся натягивать на продуктив ручками.

Подробно про этот инструмент рассказывается на вебинаре: <http://infostart.ru/webinars/564260/>

Установка инструмента:

opm install deployka

deployka help

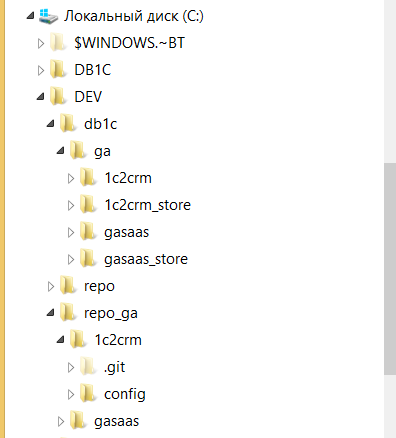


Обновление нашей рабочей базы из собранного дистрибутива:

deployka loadcfg <Строка подключения к рабочей базе> c:\dev\.packman

## Приложение. Наводим порядок в файловой системе

В ходе разработки по промышленным стандартам используется достаточно много софта и файлов, которые необходимо как-то у себя хранить и структурировать. Поэтому заранее следует продумать структуру каталогизации информации на своём жёстком диске. Приведу пример того, как я это сделал у себя. Я не претендую на образцово-показательный эталон, главное, чтобы стало понятно, с какого рода информацией предстоит работать, и о чем стоит озаботиться.



В корне диска – каталог **DEV**, в котором хранится вообще всё, что хоть как-то связано с разработкой. В нём лежат каталоги **db1c** (информационные базы, они же конфигурации) и **repo** (локальные хранилища репозиториев Git). Этих каталогов может быть несколько, в зависимости от направления разработки. Например: **repo** – связанные с моей основной работой; **repo\_my** – разработка для сообщества и личные наработки; и т.д.

Каждый репозиторий – это уже непосредственно то, что хранится в Гите и связано с внешним репо на хостинге Гитлаба или ещё где-то. Внутри каждого репо также завожу папки под отдельные нужны проекта:

* config или cfg – исходники конфы 1С
* extforms – внешние отчёты и обработки
* exchenge – правила обмена
* feature – фичи BDD-сценариев
* и т.д.

Каталог с базами, в свою очередь, аналогично делится на подкаталоги по направлениям разработки, а в тех уже лежат пары папок под каждый проект: одна для самой конфигурации, вторая – для ее хранилища.

Все наименования папок называю латинскими буквами, слитно, … в общем по правилам MS DOS, и как можно короче. Это поможет меньше заморачиваться на написании команд в командной строке, а при разборе конфы на исходники будет меньше вероятности, что длина полного пути к какому-нибудь файлу перевалит за установленные ограничения Windows.