# Работа с Git и CopyRelease

**Вся наша разработка разбита на репозитории.**

* InfoTaks – первая версия InfoTask
* InfoTask2 – вторая версия InfoTask
* AcesssApps – другие приложения Access (ServiceStend, ProjectBD, MakeArhiv и т.д.)

Локально каждый репозиторий содержится в своем каталоге (например, C:\AccessApps). Каталог репозитория кроме прочего всегда содержит подкаталоги Release, Debug и каталоги исходников C# (MainSources и т.п.). Разработка приложений Access производится в каталоге Debug.

## Схема обмена файлами.

***Git***

**Сервер**

**Папка**

**Release**

**Папка**

**Debug**

***CopyRelease***

***Pull***

***Push***

***Выполнить копирование***

***Копирование вручную***

***Commit***

**Получение данных с сервера** выполняется в следующей последовательности:

1. Операция **Commit** (в Git) – только для разработчиков C#, или если случайно открыли один из файлов Access в каталоге Release.
2. Операция **Pull** (в Git) – актуализация папки Release данными с сервера.
3. Операция **Выполнить копирование** (в CopyRelease) – обновление своей папки Debug из актуализированной Release

**Отправка на сервер своего творчества (разработчики Access):**

1. Операция **Pull** (в Git) – актуализация папки Release данными с сервера.
2. РУКАМИ (!) перенести своё творчество (файлы, папки и т.д.) из папки Debug в папку Release.
3. Операция **Commit**, она же Коммитить, она же Зафиксировать (в Git) – фиксация изменений.
4. Операция **Push** (в Git) – перенос своей папки Release на сервер.

**Обновление файла CopyRelease**

1. Открыть в Git репозиторий AccessApp.
2. Сделать **Pull**.
3. РУКАМИ (!) перенести файл CopyRelease.accdb из папки AccessApp\Release в папку Debug того репозитория, с которым собираемся работать (AcesssApps, InfoTask и т.п.)

**Отправка на сервер своего творчества (разработчики C#):**

1. Если хочется поделиться текущей версией с разработчиками Access, то нужно в VisualStudio построить решение как Release. Если просто хочется сохранить исходники на сервер, то строить решение как Release не нужно.
2. Операция **Commit**, она же Коммитить, она же Зафиксировать (в Git) – фиксация изменений.
3. Операция **Pull** (в Git) – актуализация папки Release данными с сервера.
4. Операция **Push** (в Git) – перенос своей папки Release на сервер.

## Выполнение операций в GitExtensions

**Открытие репозитория Git:**

1. Запустить **GitExtensions .**
2. Слева в списке **Недавние репозитории** выбрать репозиторий (AcesssApps, InfoTask2 и т.п.).
3. Если репозитория нет в списке, то нажать **Открыть репозиторий** и выбрать каталог репозитория.

**Операция Commit**, она же Коммитить, она же Зафиксировать – фиксация наработанных изменений

1. Нажать кнопку  (или Коммит, или Зафиксировать).
2. В появившемся окне слева нажать кнопку .
3. В окошке справа снизу описать изменения, сделанные с предыдущего коммита.
4. Нажать кнопку ****.
5. При разработке на C# имеет смысл не бояться и выполнять Commit постоянно, чтобы сохранять версию и если что откатываться. Выполнять даже если разрабатываемая программа не работает или нет связи с сервером.

**Операция Push**

1. Нажать кнопку  (Отправить).
2. В появившемся окне выбрать Push в Внешние:

**Server** – если заливается на сервер Ute,

**GitHub** – если заливается на сайт GitHib.com.

1. Оби ветви указать – **master** (если не используются другие ветви).Если нет в списке, то вписать вручную в поле).
2. Нажать .
3. Если выпало сообщение об ошибке, то его нужно внимательно прочитать.

* Возможно, сначала нужно выполнить операции **Commit** или **Pull**.
* Возможно, нет связи с сервером.

**Операция Pull** – актуализация папки Release данными с сервера.

1. Нажать кнопку  (Open pull dialog).
2. В появившемся окне выбрать Pull из Внешние:

**Server** – если заливается c серверf Ute,

**GitHub** – если заливается c сайта GitHib.com.

1. Выбрать Внешнюю ветвь – **master** (если не используется другая ветвь).Если нет в списке, то вписать вручную в поле).
2. Опции слияния выбрать 1-й пункт (**merge**).
3. Нажать **.**
4. Если выпало сообщение об ошибке, то его нужно внимательно прочитать.

* Возможно, сначала нужно выполнить операцию **Commit**.
* Возможно, нужно решить конфликты.
* Возможно, нет связи с сервером.

**Разрешение конфликтов** возникших при операции Pull

1. Конфликт возникает, если два пользователя изменили один и тот же файл. В случае файла Access нужно понять, чьи изменения в приоритете.
2. Когда при операции Pull система спросит, нужно ли решить конфликты – ответить OK.
3. В появившейся форме щелкать поочередно на каждый файл в списке и выбирать **Выбрать наш** или **Выбрать их**, в зависимости от того, изменения какого пользователя приоритетнее. Для ускорения процесса можно использовать горячие клавиши Ctrl+1 и Ctrl+2.
4. В случае, если нужно решить конфликт между двумя файлами C# нужно воспользоваться инструментом слияния.
5. Когда все конфликты будут решены, программа предложит сделать коммит – нужно соглашаться.

**Откат версии** на определенный коммит

**Настройка внешних репозиториев для диалогов Pull и Push**

1. Открыть в GitExtensions репозиторий. Нажать Pull 
2. В окне Pull нажать **Репозитории**. Откроется окно редактирования списка репозиториев связанных с данным.
3. В списке слева выбирается репозиторий или добавляется новый. Справа задаются свойства.
4. Для репозитория задается название и URL.
5. Договоренность по имени следующая:

название репозитория с сервера Ute всгда задаем Server,

название репозитория с GitHub.com всегда задаем GitHub,

можно также добавить другие репозитории: локальный каталог, каталог на внешнем диске и т.п.

1. В поле Url для репозитория с сервера Ute указывается путь к каталогу на сервере. Серверные репозитории лежат в каталоге [\\Ute02\ute\Git\](file:///\\Ute02\ute\Git\).
2. Имя каталога репозитория = имя репозитория + Git (например InfoTask2Git или AccessAppsGit)
3. В поле Url для репозитория с сайта GitHub.com указывается либо URL скопированный с сайта. Настройка связи с GitHub описана ниже.
4. После изменения названия и Url нужно нажать кнопку **Сохранить**. Далее можно переходить к настройке следующего репозитория.
5. На вкладке **Поведение по умолчанию** можно при желании настроить работу с ветками репозитория.

## Выполнение операций в CopyRelease

**Запуск CopyRelease**

1. Если известно, что файл CopyRelease.accdb обновился его нужно залить через Pull в репозитории AccessApp, а затем скопировать из подкаталога Release каталога AccessApp в папку Debug репозитория, с которым идет работа.
2. Запустить CopyRelease.accdb из папки Debug репозитория.

**Загрузка профиля**

1. В главной форме CopyRelease в поле **Файл профилей** должен быть указан файл \\Ute02\ute\Git\CopyReleaseProfiles.accdb.
2. Если **Имя профиля** не совпадает с именем нужного профиля, нужно загрузить профиль. Для этого нужно нажать кнопку **Сохраненные профили**.
3. В появившейся форме выбрать профиль и нажать **Загрузить профиль**.

**Копирование данных** из папки Release в папку Debug (хотя возможны и другие варианты)

1. Для копирования данных нажать кнопку **Выполнить копирование**. Копирование будет производиться в соответствии с выбранным профилем.

## Настройка на репозиторий в GitHub

1. Заведение пользователя на GitHub, если он еще не заведен.

* Зайти на GiHub.com.
* Нажать Sign Up.
* Указать UserName, Email Address, придумать Password.
* Choose your personal plan – выбирать Unlimited public repositories for free.

1. Добавить на GiHub.com пользователя в организацию Uraltehenergo

* Одному из администраторов Uraltehenergo (например martugin) зайти на сайте в организацию Uraltehenergo.
* Нажать Invite someone. Поиском найти пользователя по имени.
* В окне пользователя выбрать **member** и нажать **Invite**.
* Приглашение придет пользователю на почту где его нужно будет подтвердить.

1. Общие настройки GitExtensions.

* Запустить GitExtensions.
* Выбрать пункт **Настройки** в меню Настройки.
* Слева выбрать **Глобальные настройки**. Указать Имя пользователя и Адрес электронной почты.
* Слева выбрать **SSH**. Переключатель Укажите ssh клиент установить в положение PuTTY.

1. Настройка Proxy

* Настройка Proxy нужна, если выход в Internet производится через Proxy, например в офисе УТЭ.
* В настройках GitExtension слева нужно выбрать **Plugins\Proxy Switcher**.
* Указать HttpProxy 192.168.30.7. HttpProxyPort 3128 (для УТЭ).
* Если настройки Proxy сменились, то их всегда можно посмотреть в настройках браузера.

1. Настройка автозапуска для Putty

* Нужно создать назначенное задание для запуска файла **plink.exe**.
* Для этого нужно запустить

Панель управления\Система и безопасность \Администрирование \ Планировщик заданий.

* Выбрать пункт **Создать простую задачу**. Следовать указаниям, каждый раз нажимая Далее >.
* Задать имя задачи PuTTY.
* Задать Триггер: запускать задачу при входе в Windows
* Задать Действие: Запустить программу
* Указать путь к файлу c:\Program Files (x86)\GitExtensions\PuTTY\plink.exe
* Завершить создание задачи. Нажать Готово.

1. Создание ключа для Putty

* В GitExtensions в меню Внешние репозитории\PuTTY выбрать Генерировать или импортировать ключ.
* В появившемся окне нажать **Generate**.
* Водить 20-30 секунд мышкой по серому пространству под индикатором процесса, пока индикатор не дойдет до конца.
* Нажать кнопку **Save private key** и сохранить куда-нибудь файлик с ключом
* Скопировать в буфер обмена все из верхнего большого серого поля Public key for pasting...

1. Ввести ключ в аккаунт GitHub

* На сайте GitHub.com открыть аккаунт Uraltehenergo.
* Слева в меню выбрать **SSH and GPG keys**.
* В появившейся форме нажать зеленую кнопку **New SSH key**.
* В поле Key вставить ключ из буфера обмена.
* В поле Title написать произвольное имя ключа, например имя пользователя.
* Нажать кнопку Add SSH key.

1. Получить URL репозитория

* На сайте GitHub.com открыть аккаунт Uraltehenergo.
* Слева в меню выбрать Repositories.
* В появившемся списке выбрать нужный репозиторий.
* В появившейся форме нажать кнопку Clone or downdload.
* Скопировать из белого поля адрес URL.

1. Клонировать репозиторий с GitHub (нужно, если на компьютере репозиторий еще не используется)

* В стартовой форме GitExtensions слева сверху нажать Клонировать репозиторий.
* В поле **Внешний репозиторий** для клонирования вставить URL репозитория.
* В поле **Назначение** указать родительский каталог для каталога репозитория.
* В поле **Подкаталог для создания** указать имя каталога репозитория.
* В поле ветвь указать master.
* Тип репозитория указать Личный репозиторий.
* Нажать кнопку **Загрузить SSH ключ**.
* Указать путь к файлику с ключом (private key), созданному в пункте 5.
* Нажать кнопку **Клонировать**.
* После этого в списке репозиториев окна Pull, желательно переименовать origin в GitHub

1. Настройка связи с GitHub для существующего репозитория

* Открыть в GitExtensions репозиторий. Нажать Pull
* В окне Pull нажать Репозитории.
* В списке слева выбрать репозиторий или добавить новый. Справа задаются свойства.
* Для репозитория задать Название – GitHub (договоренность такая у нас).
* В поле Url занести URL скопированный с GitHub (см. выше пункт 8).
* Нажать Сохранить. Нажать Закрыть.