Оглавление

[Установка 2](#_Toc520811611)

[Интеграция с GitHub 3](#_Toc520811612)

[Создание и настройка заданий 4](#_Toc520811613)

[1. Создать новое задание 4](#_Toc520811614)

[2. Указать систему контроля версиями 4](#_Toc520811615)

[3. Настроить триггеры сборки 5](#_Toc520811616)

[4. Настройка среды сборки 5](#_Toc520811617)

[5. Заполнение этапов сборки 6](#_Toc520811618)

[6. Послесборочные операции 6](#_Toc520811619)

[Примеры скриптов 10](#_Toc520811620)

[Общие подходы 10](#_Toc520811621)

[Автоматическая сборка приложения 1С после изменения в Git 10](#_Toc520811622)

[Автоматическая проверка кода 1С после изменения в Git 10](#_Toc520811623)

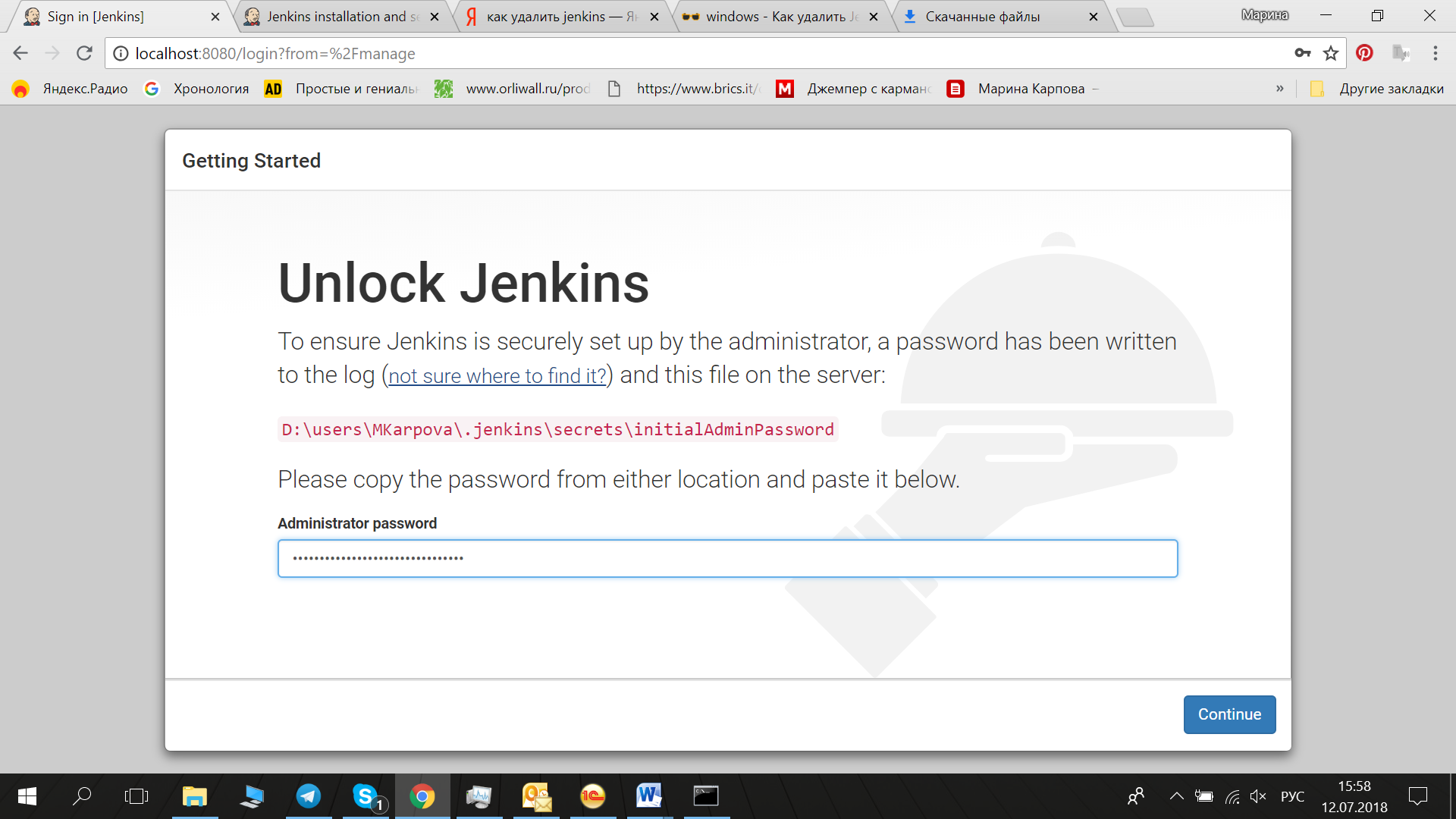
# Установка

1. Скачиваем дистрибутив с сайта: <https://jenkins.io/download/> для Windows.
2. Устанавливаем Jenkins как службу windows.
3. Если настроен внешний IP, обращаемся по нему, если нет, тогда по локальному адресу. Порт по умолчанию: «8080». Пример: <http://localhost:8080/>

Порт можно изменить в файле: C:\Program Files (x86)\Jenkins\ jenkins.xml. «httpPort=8080».

После изменения порта необходимо перезапустить службу «Jenkins».

1. При первом запуске Jenkins необходимо ввести пароль администратора из файла, расположенного по указанному пути:



Нажимаем «Continue».

1. Далее выбираем установку плагинов по - умолчанию (левая кнопка).
2. После окончательной установки система предложит создать пользователя, под которым будем работать в дальнейшем. Заполняем поля и нажимаем «Save and continue».
3. Установка завершена.

# Интеграция с GitHub

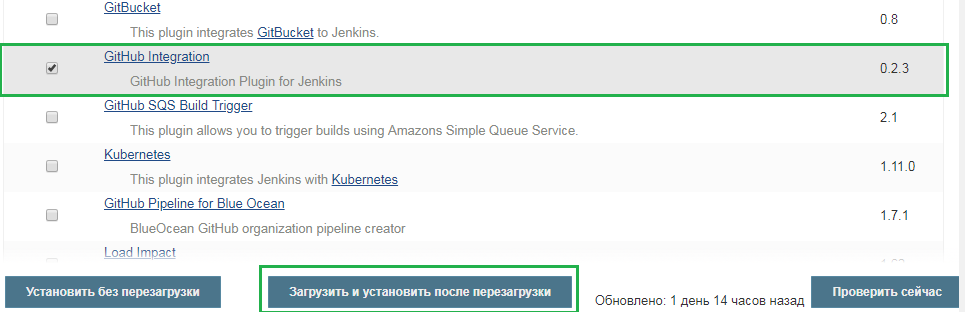
Существует два метода реализации взаимодействия Jenkins с GitHub:

* **Jenkins, по расписанию, опрашивает репозиторий GitHub** на наличие изменений.
* **GitHub уведомляет Jenkins, при изменении своего репозитория**. Для реализации данного метода необходимо:

1. Сервер, на котором установлен Jenkins, должен откликался при обращении по внешнему IP адресу на порт 8080. Порт Jenkins можно изменить, по - умолчанию, порт «8080».

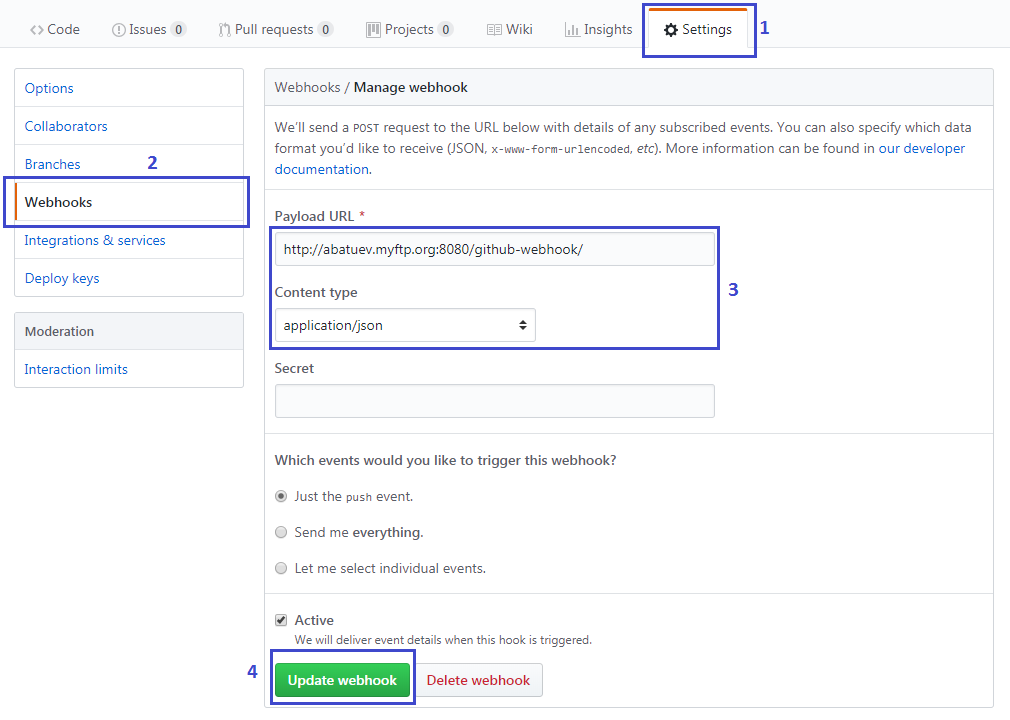
1. Установить на сервере Jenkins консоль «GitHub»: <https://git-scm.com/download/win>. Перезагрузить сервер Jenkins.
2. Установить в Jenkins плагин «[GitHub Integration](https://plugins.jenkins.io/github-pullrequest)»:

Раздел: «Настроить Jenkins» -> «Управление плагинами».

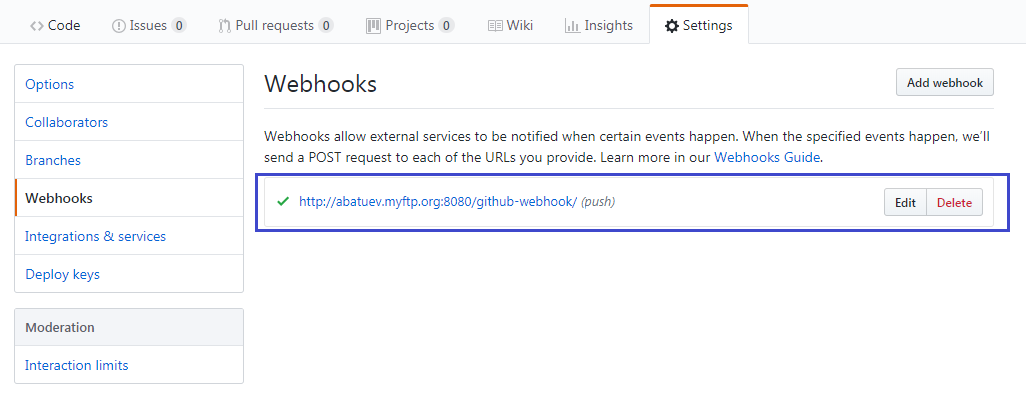


1. Настроить репозиторий в GitHub. Payload: «http://<IP>:<PORT>/github-webhook/».

Пример настройки:



Результат настройки:

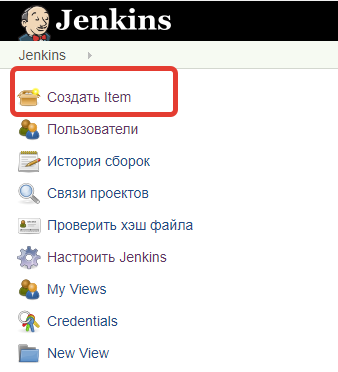


# Создание и настройка заданий

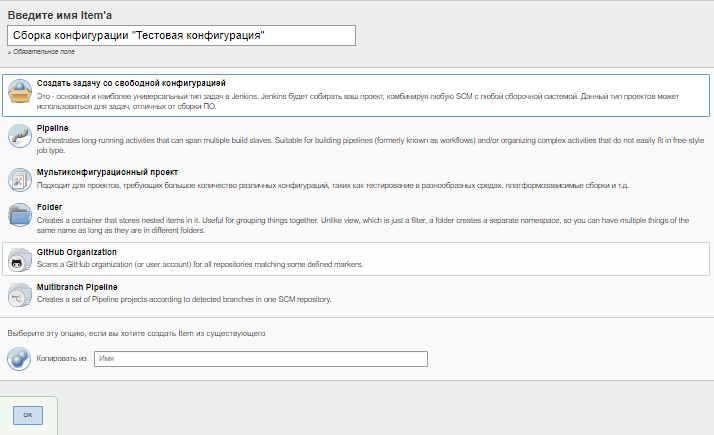
После внесения изменений в репозитории Git, сервер Jenkins запускает сборку готового приложения.

## Создать новое задание

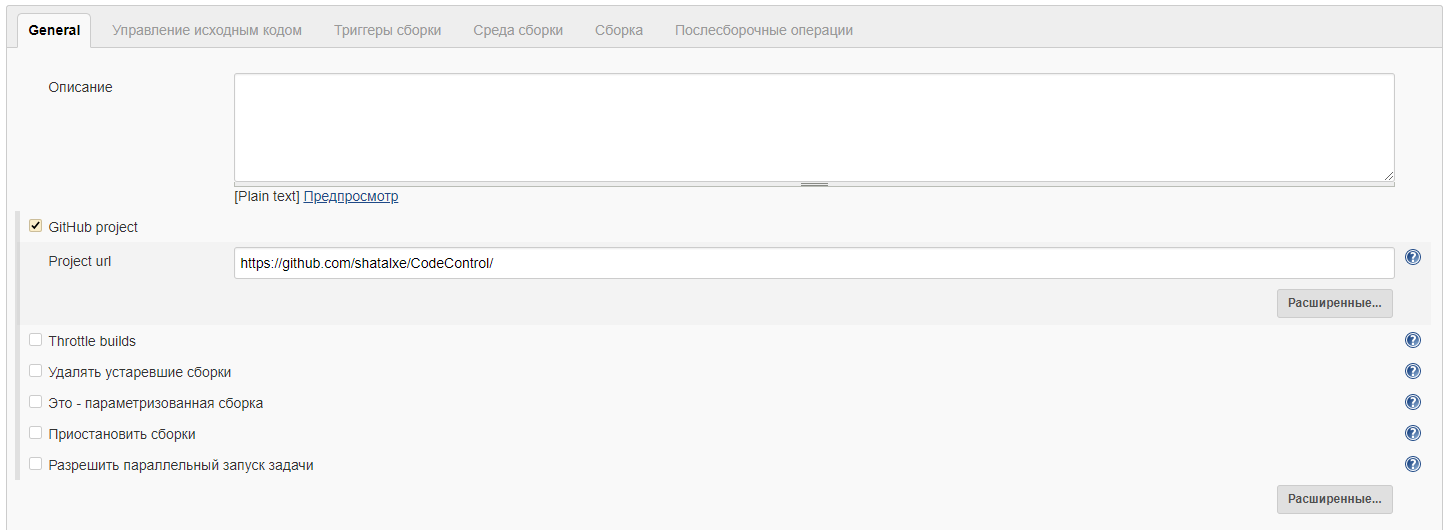
* 1. Создать Item



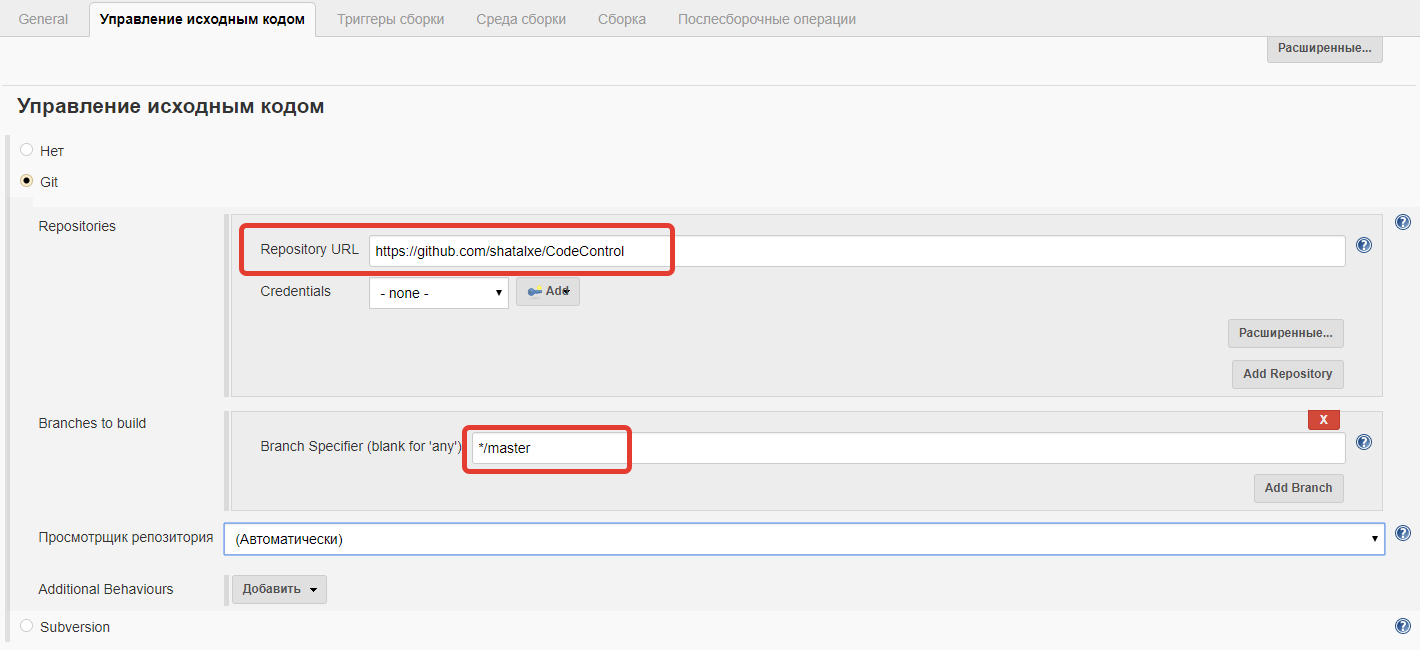
* 1. Выбрать **создать задачу со свободной конфигурацией**, задать имя, **Ok**



Откроется окно настройки задания



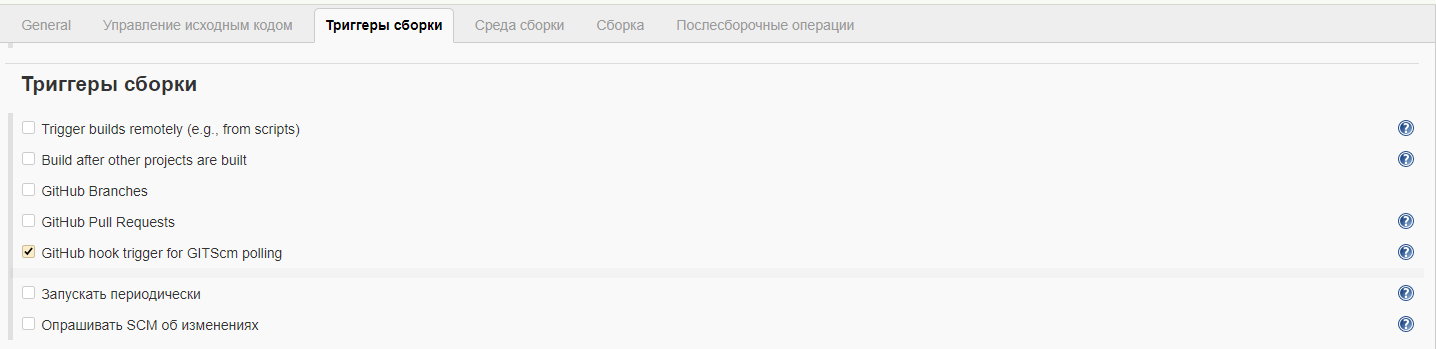
## Указать систему контроля версиями



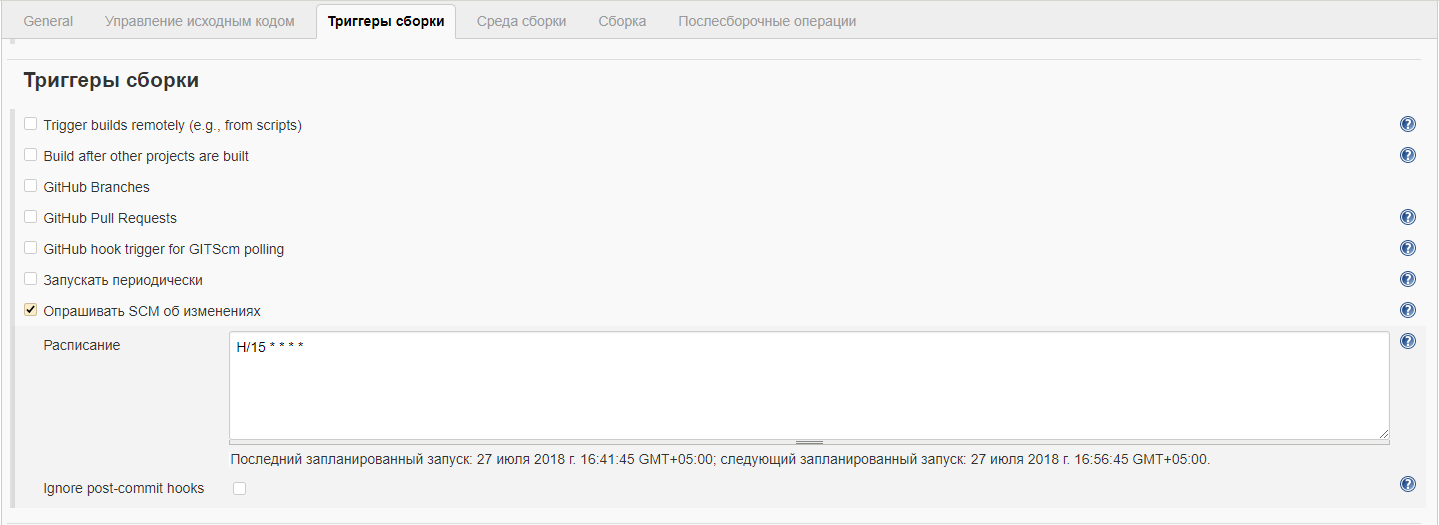
Для срабатывания задания по коммиту в любую ветку, в **поле Branch Specifier** поставить \*

## Настроить триггеры сборки

* Срабатывание по оповещению GitHub’а



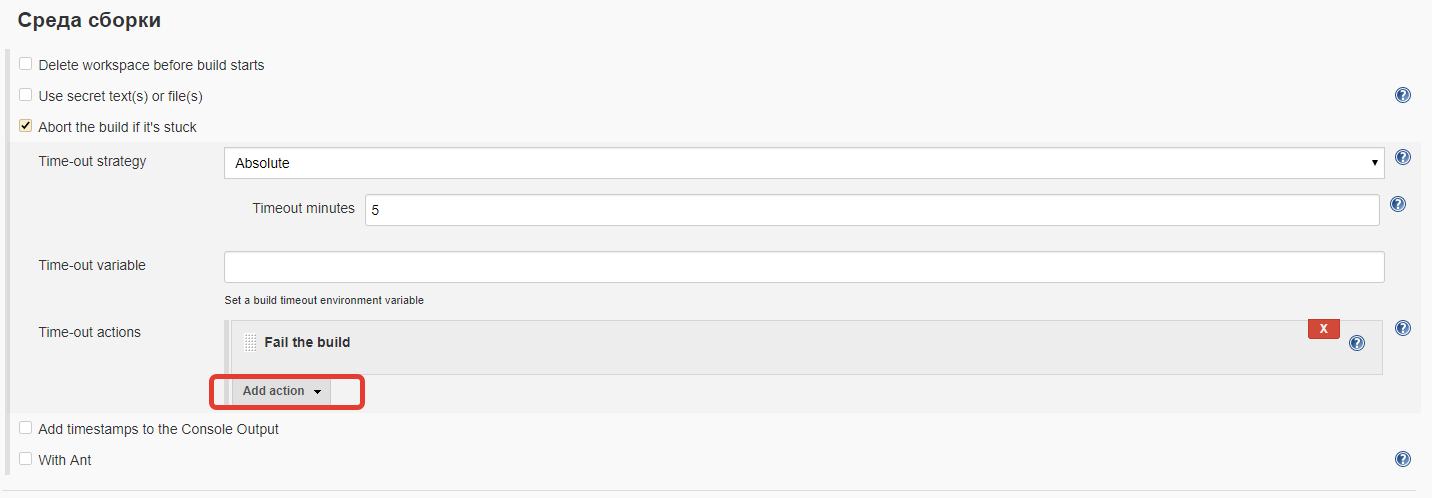
* Опрос Github по расписанию



## Настройка среды сборки

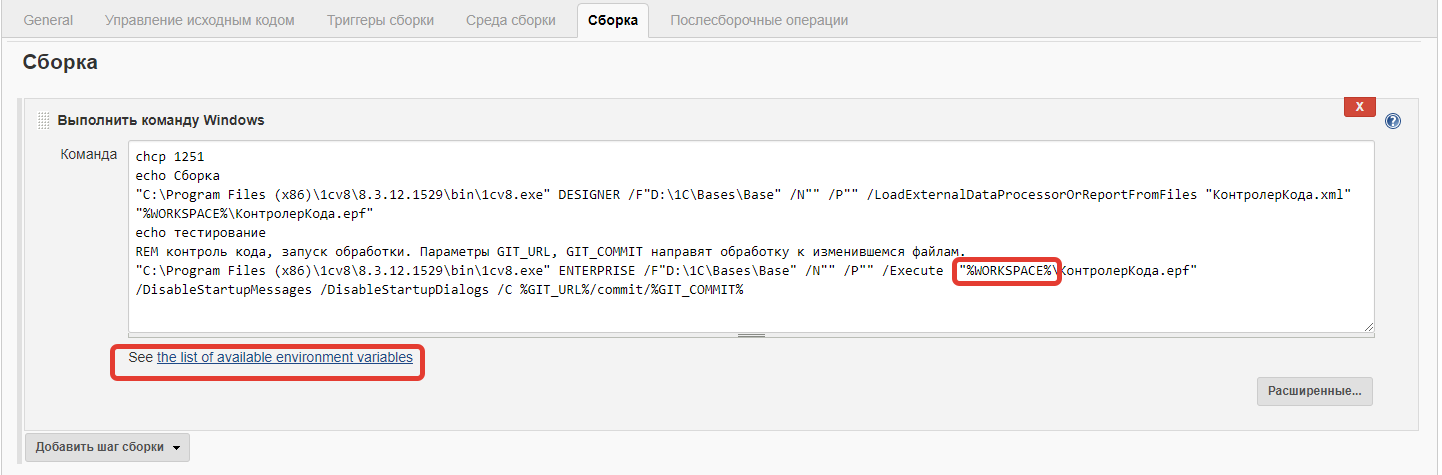
Для установки прекращения выполнения задания по таймауту, выставить флаг **Abort the build if it's stuck.**

Для того, чтобы сборка считалась неуспешной (например, чтобы разослать уведомления) нужно добавить событие (Add action) **Fail to build.**



## Заполнение этапов сборки

**Выполнить команду Windows**



Во время выполнения задания Jenkins по тексту скрипта соберет и выполнит bat-файл из **рабочего каталога** (WORKSPACE).

*Расположение* *bat-файла C:\Windows\TEMP\jenkins<id>.bat*

*Расположение* *рабочего каталога C:\Program Files (x86)\Jenkins\workspace\<имя проекта>\*

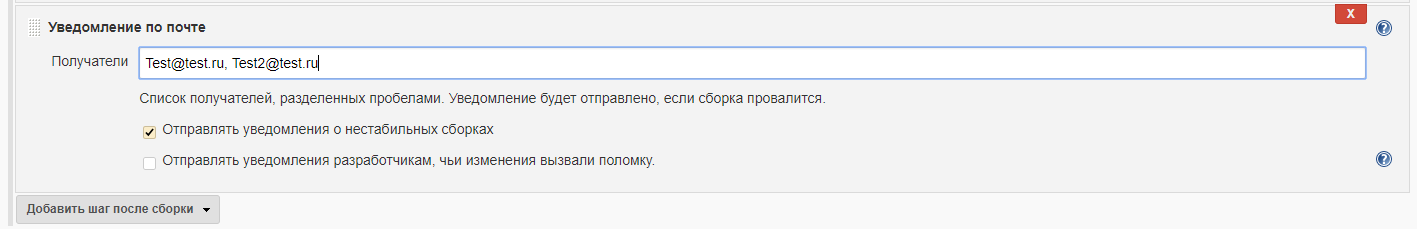
Допускается использование переменных. Список переменных **See the list of available environment variables.**

## Послесборочные операции

**Уведомление по почте**

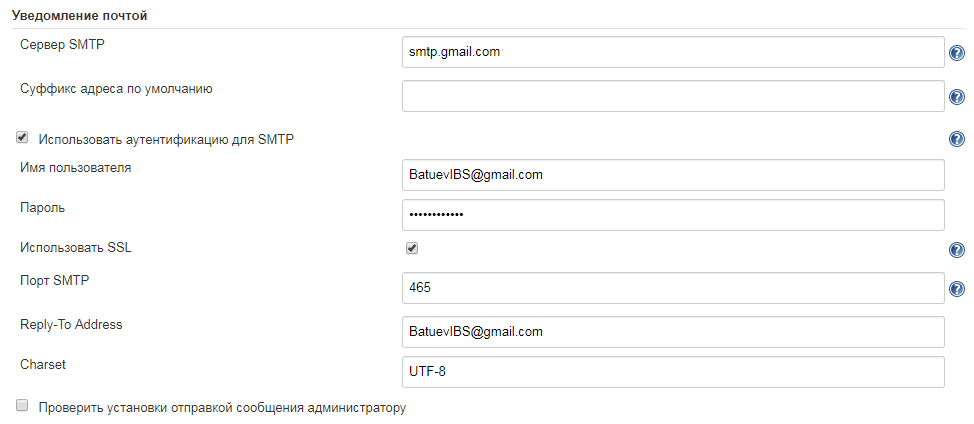
Данное уведомление срабатывает, когда сборка завершилась с ошибкой. При возникновении ошибки, сценарий выставит ненулевой %ERRORLEVEL%.

Указанному списку получателей будет отправлен лог выполнения.



Для настройки отправителя, необходимо в конфигурации Jenkins’а настроить раздел «Настроить Jenkins» -> «Конфигурация системы» -> «Уведомление почтой».

Пример настройки отправки уведомления с почтового сервера «gmail.com»:

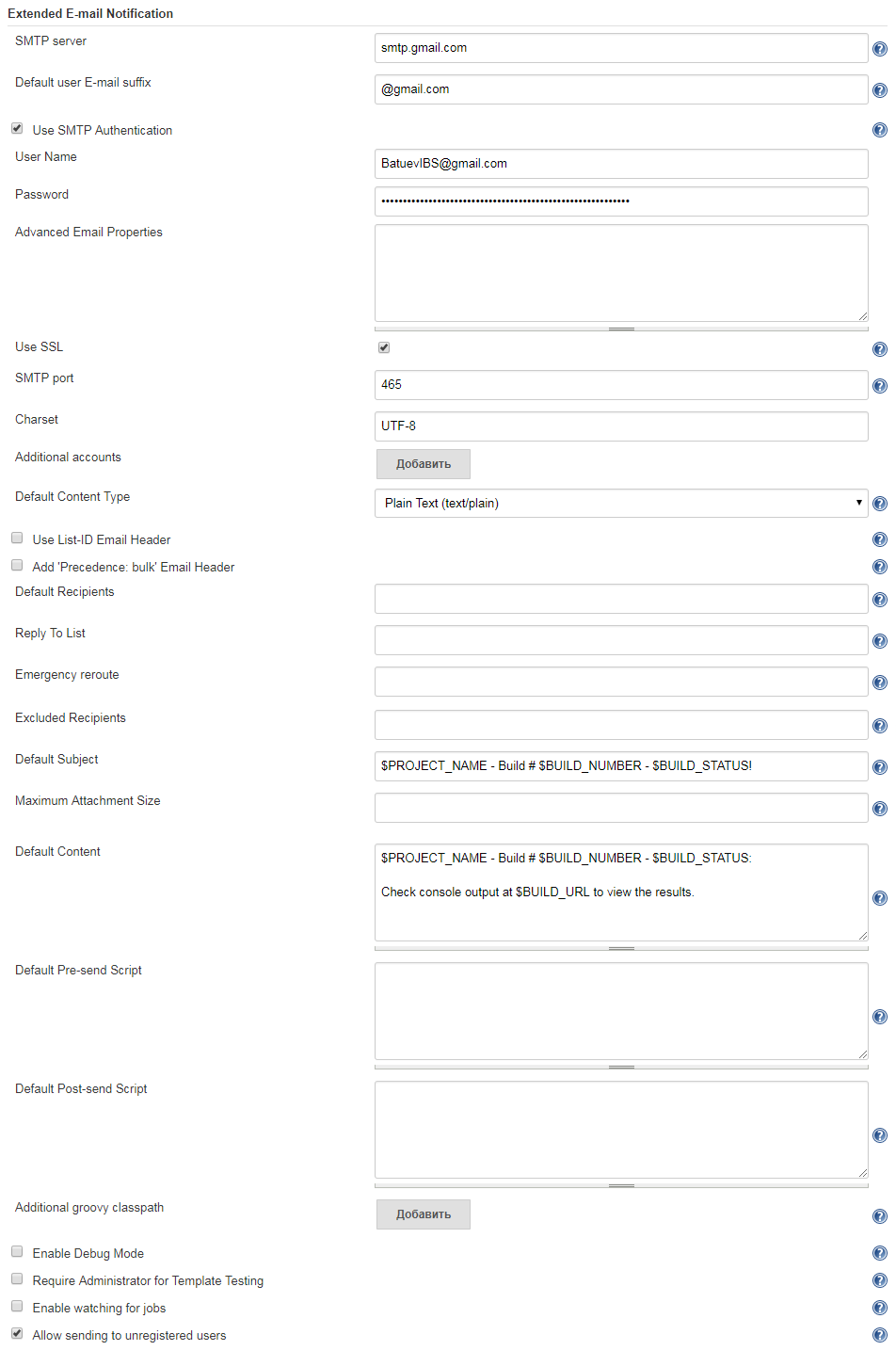


**Editable Email Notification**

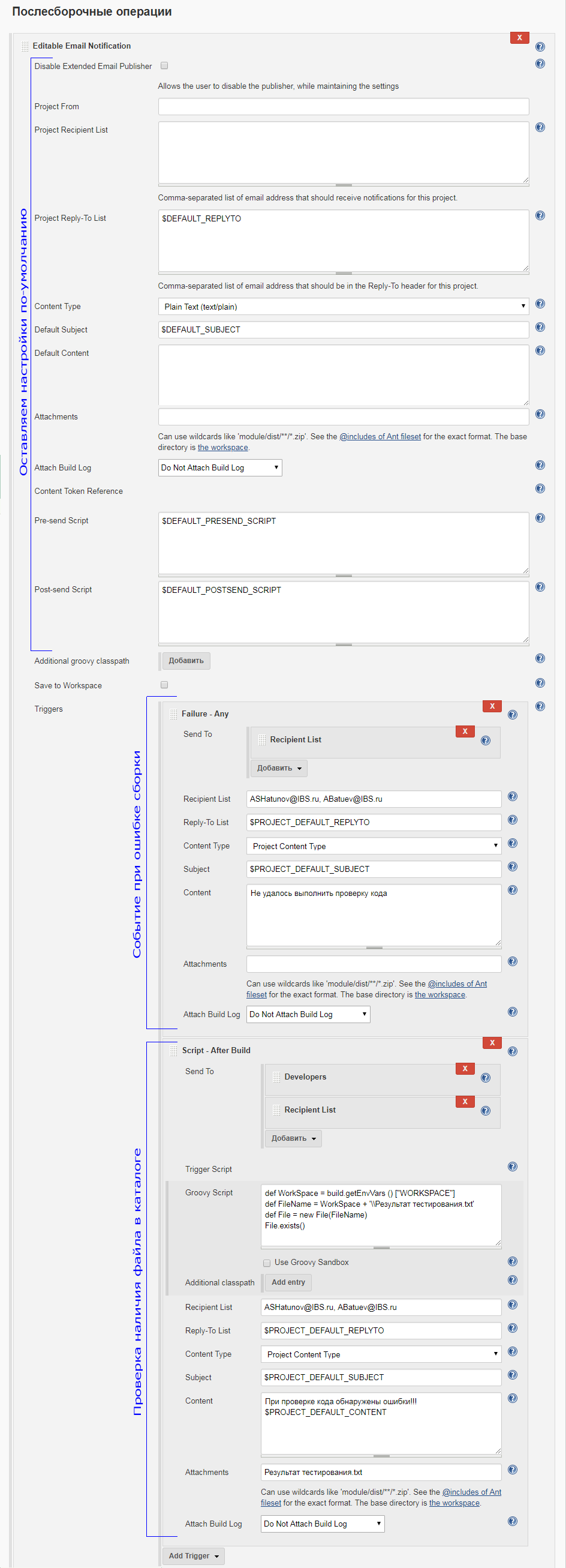
Это расширенное уведомление, которое можно настроить под определенные события.

Необходимо заполнить настройки по умолчанию в разделе «Настроить Jenkins» -> «Конфигурация системы» -> «Extended E-mail Notification».

Пример настройки отправки расширенных уведомлений с почтового сервера «gmail.com»:



В послесборочных операциях можно указать различные события (triggers), при выполнении которых сработает данное уведомление:



В тригеррах можно переопределить настройки по умолчанию (список получателей, тему, содержание).

В содержании можно указывать следующие конструкции:

* **${SCRIPT, template="robot.groovy"}** – выполняет скрипт на языке «Groovy» и помещает результат в содержимое письма. Скрипт должен находиться в каталоге «email-templates» домашней директории «Jenkins». Например: «C:\Program Files (x86)\Jenkins\email-templates\»
* **${FILE,path="результат тестирования.txt"}** – помещает в содержание письма содержимое файла, находящегося в домашней директории проекта. Например: «C:\Program Files (x86)\Jenkins\workspace\Run 1C\».

Триггер **«Failure Any»** срабатывает при любой ошибке во время сборки проекта.

Триггер **«Script – After build»** выполняет скрипт после сборки на языке «Groovy», на примере описан алгоритм поиска файла в домашней директории проекта. Последняя строка алгоритма должна содержать «true» или «false».

Пример скрипта Groovy (проверка наличия файла в домашнем каталоге сборки):

**def WorkSpace = build.getEnvVars () ["WORKSPACE"]**

**def FileName = WorkSpace + '\\Результат тестирования.txt'**

**def File = new File(FileName)**

**File.exists()**

Все остальные триггеры отличаются только событиями, которые их вызывают.

# Примеры скриптов

## Общие подходы

chcp 1251 – установить кодировку Windows-1251

В качестве пути к выгруженной конфигурации (обработке/отчету) достаточно указать текущий каталог, т.к. команда выполняется в рабочем каталоге (WORKSPACE), куда загружаются файлы из GitHub

Каталог сохранения должен быть доступен пользователю, от которого запущена служба Jenkins

Сложные конструкции следует комментировать (REM).

Параметры, заключенные в %% при создании bat-файла, будут заменены значением.

## Автоматическая сборка приложения 1С после изменения в Git

chcp 1251

"C:\Program Files (x86)\1cv8\8.3.12.1529\bin\1cv8.exe" DESIGNER /F"D:\1С\Bases\Base" /N"" /P"" /LoadExternalDataProcessorOrReportFromFiles "КонтролерКода.xml" "D:\КонтролерКода.epf"

## Автоматическая проверка кода 1С после изменения в Git

chcp 1251

"C:\Program Files (x86)\1cv8\8.3.12.1529\bin\1cv8.exe" DESIGNER /F"D:\1С\Bases\Base" /N"" /P"" /LoadExternalDataProcessorOrReportFromFiles "КонтролерКода.xml" "%WORKSPACE%\КонтролерКода.epf"

REM контроль кода, запуск обработки. Параметры GIT\_URL, GIT\_COMMIT направят обработку к изменившемся файлам.

"C:\Program Files (x86)\1cv8\8.3.12.1529\bin\1cv8.exe" ENTERPRISE /F"D:\1С\Bases\Base" /N"" /P"" /Execute "%WORKSPACE%\КонтролерКода.epf" /DisableStartupMessages /DisableStartupDialogs /C %GIT\_URL%/commit/%GIT\_COMMIT%