



Бесплатная электронная книга

УЧУСЬ jenkins

Free unaffiliated eBook created from
Stack Overflow contributors.

#jenkins

| | |
|------------------------------------------|-----------|
| | 1 |
| 1: | 2 |
| | 2 |
| | 2 |
| | 2 |
| 1.x 2.x | 2 |
| Examples..... | 4 |
| | 4 |
| jenkins (RPM)..... | 5 |
| - Nginx..... | 5 |
| | 6 |
| | 7 |
| | 7 |
| | 8 |
| Jenkins 2..... | 14 |
| 2: Jenkins Groovy Scripting | 16 |
| Examples..... | 16 |
| | 16 |
| | 16 |
| | 17 |
| | 17 |
| 3: Jenkins iOS | 19 |
| | 19 |
| | 19 |
| | 19 |
| Examples..... | 19 |
| | 19 |
| 4: iOS Shenzhen | 21 |
| Examples..... | 21 |
| iOS Shenzhen..... | 21 |
| 5: Git Jenkins | 22 |
| | |

| | |
|-------------------------------------------|-----------|
| Examples..... | 22 |
| | 22 |
| 6: | 31 |
| Examples..... | 31 |
| | 31 |
| | 31 |
| | 32 |
| 7: Jenkins Windows SSH GitHub..... | 35 |
| Examples..... | 35 |
| GitHub..... | 35 |
| PSEXEC.exe PS Tool Microsoft..... | 35 |
| SSH Jenkins, PSEXEC PSEXEC64..... | 35 |
| Jenkins..... | 36 |
| ~, | 38 |
| | 40 |

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [jenkins](#)

It is an unofficial and free jenkins ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official jenkins.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

глава 1: Начало работы с дженкинсами

замечания

[Jenkins](#) - инструмент непрерывной интеграции с открытым исходным кодом, написанный на Java. Проект был раздвоен из [Хадсона](#) после спора с [Oracle](#) .

Jenkins предоставляет услуги непрерывной интеграции для разработки программного обеспечения. Это серверная система, работающая в контейнере сервлетов, таком как Apache Tomcat. Он поддерживает инструменты SCM, включая AccuRev, CVS, Subversion, Git, Mercurial, Perforce, Clearcase и RTC и может выполнять проекты на основе Apache Ant и Apache Maven, а также произвольные сценарии оболочки и командные команды Windows. Первым разработчиком Дженкинса является [Кохсуке Кавагути](#) . Выпущенный под лицензией MIT, Дженкинс является свободным программным обеспечением.

Сборка может запускаться различными способами, в том числе инициироваться фиксацией в системе управления версиями, путем планирования через механизм, подобный cron, путем построения, когда другие сборки завершены, и путем запроса определенного URL-адреса сборки.

Версии

Дженкинс

| Версия | Дата выхода |
|--------|-------------|
| 1,656 | 2016-04-03 |
| 2,0 | 2016-04-20 |

Дженкинс 1.x против Дженкинса 2.x

Дженкинс (и по сей день) является системой непрерывной интеграции (CI), которая позволяет автоматизировать процесс разработки программного обеспечения, такой как создание кода для триггеров фиксации SCM. Однако растущая потребность в непрерывной доставке (CD) потребовала, чтобы Дженкинс эволюционировал для чистой системы CI к соединению CI и CD. Кроме того, потребность в индустриализации рабочих мест Дженкинса растет, и классические работы Jenkins 1.x `Freestyle/Maven jobs` стали слишком ограниченными для определенных потребностей.

Под Jenkins 1.xa плагин под названием `workflow-plugin` появился, чтобы позволить разработчикам писать код для описания заданий. Jenkins 2 идет дальше, добавляя встроенную поддержку `Pipeline as Code`. Главное преимущество заключается в том, что конвейеры, являющиеся файлами сценариев Groovy, могут быть более сложными, чем автономные пользовательские задания `freestyle`, и могут контролироваться версиями. Jenkins 2 также добавляет новый интерфейс, который позволяет легко визуализировать различные «этапы», определенные в конвейере, и следить за ходом всего конвейера, например, ниже:

Stage View

Average stage times:
(Average full run time: ~27y
220d)

Build the sudo
images for
installation

19s

#34

Mar 03

18:56

81
commits

1 min 34s

master

failed

#33

Dec 22

13:00

3
commits



17s

master

#32

Dec 22

12:33

20
commits



15s

master

#31

Dec 22

12:16

No
Changes



16s

master

#30

Dec 22

No



Red Hat Enterprise Linux (RHEL), CentOS, Fedora или Scientific Linux

Чтобы загрузить файл репозитория для стабильной версии:

```
sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo
```

Или, если вам нужны последние еженедельные выпуски:

```
sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat/jenkins.repo
```

Импорт открытого ключа:

```
sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key
```

Установите Jenkins с помощью yum:

```
sudo yum install jenkins
```

Jenkins требует java для запуска, чтобы установить его:

```
sudo yum install java
```

Для запуска / остановки / перезапуска использования дженкинсов:

```
sudo service jenkins start/stop/restart
```

Обновление jenkins (установки RPM)

1. Резервное копирование домашней директории jenkins
2. Замените jenkins.war в следующем месте с новым файлом WAR. / USR / Библиотека / Jenkins / jenkins.war`
3. Перезапустите Дженкинса
4. Проверьте прикрепленные плагины и отмените при необходимости
5. Перезагрузка конфигурации с диска

Примечание. Для обновлений Jenkins 2 для JENKINS_AJP_PORT="-1" сервера приложений причала отключите порт JENKINS_AJP_PORT="-1" (установите JENKINS_AJP_PORT="-1") в /etc/sysconfig/jenkins .

Настройка прокси-сервера Nginx

На самом деле Дженкинс работает на порту 8080. Мы можем установить прокси-сервер из порта 80 -> 8080, чтобы доступ к Jenkins был доступен через:

```
http://<url>.com
```

вместо стандартного

```
http://<url>.com:8080
```


Начните с установки Nginx.

```
sudo aptitude -y install nginx
```

Удалить настройки по умолчанию для Nginx

```
cd /etc/nginx/sites-available
```

```
sudo rm default ../sites-enabled/default
```

Создайте новый файл конфигурации

```
sudo touch jenkins
```

Скопируйте следующий код во вновь созданный файл `jenkins`.

```
upstream app_server {
    server 127.0.0.1:8080 fail_timeout=0;
}

server {
    listen 80;
    listen [::]:80 default ipv6only=on;
    server_name ;

    location / {
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_redirect off;

        if (!-f $request_filename) {
            proxy_pass http://app_server;
            break;
        }
    }
}
```

Создайте символическую связь между доступными сайтами и сайтами:

```
sudo ln -s /etc/nginx/sites-available/jenkins /etc/nginx/sites-enabled/
```

Перезапустите службу прокси-сервера Nginx

```
sudo service nginx restart
```

Теперь Дженкинс будет доступен из порта 80.

Установка плагина из внешнего источника

```
java -jar [Path to client JAR] -s [Server address] install-plugin [Plugin ID]
```

Клиент JAR должен быть JI-файлом CLI, а не тем же JAR / WAR, который запускает Jenkins. Уникальные идентификаторы можно найти на соответствующей странице плагинов на Wiki Jenkins CLI (<https://wiki.jenkins-ci.org/display/JENKINS/Plugins>)

Перемещение Дженкинса с одного компьютера на другой

Это помогло мне перейти от Ubuntu 12.04 (Jenkins ver. 1.628) к Ubuntu 16.04 (Jenkins, версия 1.651.2). Сначала я [установил Дженкинса из репозитория](#).

1. [Остановить оба сервера Jenkins](#)
2. Скопируйте `JENKINS_HOME` (например, `/var/lib/jenkins`) со старого сервера на новый. С консоли на новом сервере:

```
rsync -av username@old-server-IP:/var/lib/jenkins/ /var/lib/jenkins/
```
3. [Запустите новый сервер Jenkins](#)

Возможно, вам это не понадобится, но мне пришлось

- Manage Jenkins **И** Reload Configuration from Disk .
- Отключите и снова подключите все ведомые устройства.
- Убедитесь, что в Configure System > Jenkins Location Jenkins URL правильно назначен новому серверу Jenkins.

Настроить проект в Дженкинсе

Здесь мы будем проверять последнюю копию кода нашего проекта, запускать тесты и делать приложение вживую. Чтобы достичь этого, выполните следующие шаги:

1. Откройте Jenkins в браузере.
2. Нажмите ссылку « **Создать работу** » .
3. Введите название проекта и выберите ссылку « **Создать бесплатный проект** » .
4. Нажмите кнопку « **ОК** » .
5. В разделе «Управление **исходным кодом**» выберите поле рядом с инструментом управления исходным кодом. В моем случае я выбрал **Git** .

Укажите URL-адрес git-репо, например `git://github.com/example/example.git`

6. В разделе **триггеров Build** выберите радиоканал рядом с **SCM опроса** .
7. Предоставьте ***** в поле « **Расписание** » . Этот блок отвечает за запуск сборки через регулярные промежутки времени. ***** указывает, что задание будет запускаться каждую минуту для изменений в git репо.
8. В разделе « **Сборка** » нажмите кнопку « **Добавить шаг** » , а затем выберите вариант, по которому вы хотите построить проект. Я выбрал **Execute Shell** . В командной строке напишите команду для сборки, запуска тестов и развертывания в prod.
9. Прокрутите вниз и **выберите «Сохранить»** .

Итак, выше мы создали базовый проект в Jenkins, который будет запускать сборку каждую минуту для изменения вашего репозитория git. Примечание. Чтобы настроить сложный

проект, возможно, вам придется установить некоторые плагины в Jenkins.

Дженкинс полный Введение в одном месте

1. Дженкинс:

Jenkins - инструмент непрерывной интеграции с открытым исходным кодом, написанный на Java. Проект был раздвоен из Хадсона после спора с Oracle.

В двух словах, Jenkins является ведущим сервером автоматизации с открытым исходным кодом. Построенный с Java, он предоставляет сотни плагинов для поддержки строительства, тестирования, развертывания и автоматизации практически для любого проекта.

Особенности: Jenkins предлагает следующие основные функции из коробки, и многие другие могут быть добавлены через плагины:

Простота установки: просто запустите `java -jar jenkins.war`, разверните его в контейнере сервлетов. Нет дополнительной установки, нет базы данных. Предпочитаете установщик или собственный пакет? У нас есть и те. Простая конфигурация: Jenkins можно полностью настроить из своего дружественного веб-интерфейса с обширными проверками на лету и встроенной справкой. Богатая экосистема плагинов: Jenkins интегрируется практически с любым SCM или встроенным инструментом, который существует. Просмотр плагинов. Расширяемость: большинство частей Jenkins могут быть расширены и изменены, и легко создавать новые плагины Jenkins. Это позволяет вам настроить Jenkins в соответствии с вашими потребностями. Распределенные сборки: Jenkins может распространять сборки / тестовые нагрузки на несколько компьютеров с различными операционными системами. Создание программного обеспечения для OS X, Linux и Windows? Нет проблем.

Монтаж :

```
$ wget -q -O - https://jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt-key add -  
  
$ sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian binary/ >  
/etc/apt/sources.list.d/jenkins.list'  
$ sudo apt-get update  
$ sudo apt-get install jenkins  
to do more refer link :
```

Ссылка: <https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Ubuntu>

Ссылка: <http://www.vogella.com/tutorials/Jenkins/article.html>

Ссылка: <https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins>

Каталог JENKINS_HOME Дженкинсу необходимо пространство на диске для выполнения сборки и хранения архивов. Вы можете проверить это местоположение на экране

конфигурации Дженкинса. По умолчанию этот параметр установлен в `~/.jenkins`, но вы можете изменить его одним из следующих способов: Установите переменную окружения «JENKINS_HOME» в новый домашний каталог перед запуском контейнера сервлета.

Установите системное свойство «JENKINS_HOME» в контейнер сервлетов. Задайте запись среды JNDI «JENKINS_HOME» в новый каталог. Дополнительную информацию о том, как это сделать для контейнера, см. В сборке конкретных контейнеров. Вы можете изменить это местоположение после того, как вы использовали Дженкинса какое-то время. Для этого полностью остановите Дженкинса, переместите содержимое из старого JENKINS_HOME в новый дом, установите новый JENKINS_HOME и перезапустите Jenkins. JENKINS_HOME имеет довольно очевидную структуру каталогов, которая выглядит следующим образом:

JENKINS_HOME

```
+-- config.xml      (jenkins root configuration)
+-- *.xml           (other site-wide configuration files)
+-- userContent     (files in this directory will be served under your
http://server/userContent/)
+-- fingerprints    (stores fingerprint records)
+-- plugins          (stores plugins)
+-- workspace       (working directory for the version control system)
    +- [JOBNAME]     (sub directory for each job)
+-- jobs
    +- [JOBNAME]     (sub directory for each job)
        +- config.xml (job configuration file)
        +- latest     (symbolic link to the last successful build)
        +- builds
            +- [BUILD_ID] (for each build)
                +- build.xml (build result summary)
                +- log       (log file)
                +- changelog.xml (change log)
```

Jenkins Build Jobs:

Создание нового задания для сборки в Jenkins очень просто: просто нажмите на пункт меню «New Job» на панели инструментов Jenkins. Jenkins поддерживает несколько различных типов заданий построения, которые представлены вам, когда вы решите создать новое задание

Проект программного обеспечения Freestyle

Работы по созданию фристайла - это задания общего назначения, которые обеспечивают максимальную гибкость.

Проект Maven «Проект maven2 / 3» - это работа по сборке, специально адаптированная к проектам Maven. Дженкинс понимает файлы Maven pom и структуры проекта и может использовать информацию, полученную из файла pom, чтобы уменьшить работу, необходимую для настройки вашего проекта.

Workflow

Организует длительные действия, которые могут охватывать несколько подчиненных устройств. Подходит для строительства трубопроводов и / или организации сложных мероприятий, которые нелегко вписываются в тип работы в свободном стиле.

Мониторинг внешнего задания Задание задания «Мониторинг внешнего задания» позволяет следить за неинтерактивными процессами, такими как задания cron.

Задача многоконфигурации «Проект многоконфигурации» (также называемый «матричным проектом») позволяет запускать одно и то же задание сборки во многих разных конфигурациях. Эта мощная функция может быть полезна для тестирования приложения во многих разных средах, с разными базами данных или даже с разными машинами сборки.

1. Построение программного проекта (свободный стиль)

Дженкинс может использоваться для выполнения типичной работы сервера сборки, такой как непрерывная / официальная / ночная сборка, запуск тестов или выполнение некоторых повторяющихся пакетных задач. Это называется «программным проектом свободного стиля» в Дженкинсе. Настройка проекта. Перейдите на верхнюю страницу Jenkins, выберите «Новое задание», затем выберите «Построить проект со свободным стилем». Этот тип задания состоит из следующих элементов: необязательный SCM, например CVS или Subversion, где находится исходный код. дополнительные триггеры для управления, когда Дженкинс выполнит сборку. какой-то скрипт сборки, который выполняет сборку (ant, maven, shell-скрипт, командный файл и т. д.), где реальная работа выполняет дополнительные шаги для сбора информации из сборки, такие как архивирование артефактов и / или запись javadoc и test Результаты. необязательные шаги для уведомления других людей / систем с результатом сборки, таких как отправка электронной почты, мгновенных сообщений, обновление трекера и т. д.

Создает проекты, не связанные с источником. Иногда бывает необходимо создать проект просто для демонстрационных целей или недоступен доступ к хранилищу SVN / CVS. Выбрав для настройки проекта как «Нет» в разделе «Управление исходным кодом», вам необходимо:

1. Постройте проект хотя бы один раз (он не удастся), но Дженкинс создаст структуру `jenkins / workspace / PROJECTNAME /`
2. Скопируйте файлы проекта в `jenkins / workspace / PROJECTNAME /`
3. Постройте снова и настройте соответствующим образом

Дженкинс устанавливает переменные среды

Когда выполняется задание Дженкинса, он устанавливает некоторые переменные среды, которые вы можете использовать в своем сценарии оболочки, командах пакетной обработки, сценарии Ant или Maven POM. См. Список переменных, нажав на `ENVIRONMENT_VARIABLE`.

Настройка автоматических сборок

Сборка в Jenkins может запускаться периодически (по расписанию, указанному в конфигурации) или когда исходные изменения в проекте были обнаружены или их можно автоматически запускать, запросив URL-адрес:

[Http: // YOURHOST / Jenkins / Работа / Projectname / сборки](http://YOURHOST/Jenkins/Работа/Projectname/сборки)

Это позволяет подключать Jenkins к различным настройкам. Для получения дополнительной информации (в частности, с помощью включенной безопасности) см. API удаленного доступа.

Создает с помощью исходных изменений

Вы можете заставить Jenkins опросить вашу систему контроля версий за изменения. Вы можете указать, как часто Дженкинс опробовывает вашу систему контроля версий, используя тот же синтаксис, что и crontab в Unix / Linux. Однако, если ваш период опроса короче, чем требуется для опроса вашей системы контроля версий, вы можете получить несколько сборок для каждого изменения. Вы должны либо настроить период опроса дольше, чем время, затрачиваемое на опрос вашей системы контроля версий, или использовать триггер после фиксации. Вы можете просмотреть журнал опроса для каждой сборки, чтобы узнать, сколько времени потребовалось для опроса вашей системы.

В качестве альтернативы вместо опроса с фиксированным интервалом вы можете использовать триггер URL (описано выше), но с / polling вместо / build в конце URL-адреса. Это заставляет Дженкинс опробовать SCM для изменений, а не для немедленного создания. Это предотвращает запуск Jenkins сборки без каких-либо существенных изменений для коммитов, влияющих на модули или ветви, не связанные с заданием. При использовании / опросе задание должно быть настроено для опроса, но расписание может быть пустым.

Создает по электронной почте (sendmail)

Если у вас есть учетная запись root вашей системы, и вы используете sendmail, я нашел ее проще всего настроить / etc / aliases и добавить следующую запись: jenkins-foo: "| / bin / wget -o / dev / null

[http: // YOURHOST / jenkins / job / PROJECTNAME / build "](http://YOURHOST/jenkins/job/PROJECTNAME/build)

а затем запустите команду «newaliases», чтобы сообщить об этом изменениям в sendmail. Всякий раз, когда кто-то отправляет электронное письмо в «jenkins-foo @ yoursystem», это вызовет новую сборку. Подробнее см. В настройке sendmail. Создает по электронной почте (qmail) С помощью qmail вы можете написать / var / qmail / alias / .qmail-jenkins следующим образом: | / bin / wget -o / dev / null [http: // YOURHOST / jenkins / job / PROJECTNAME / build "](http://YOURHOST/jenkins/job/PROJECTNAME/build)

2. Создание проекта Maven

Дженкинс предоставляет тип задания, посвященный Maven 2/3. Этот тип работы объединяет Дженкинса с Maven 2/3 и обеспечивает следующие преимущества по сравнению с более общим программным проектом свободного стиля.

Дженкинс разбирает Maven POMs, чтобы получить большую часть информации, необходимой для выполнения своей работы. В результате размер конфигурации резко сокращается.

Дженкинс слушает исполнение Maven и выясняет, что нужно делать, когда он сам по себе. Например, он автоматически записывает отчет JUnit, когда Maven запускает тестовую фазу. Или, если вы запустите цель javadoc, Дженкинс автоматически запустит javadoc.

Jenkins автоматически создает зависимости проекта между проектами, которые объявляют зависимости SNAPSHOT между собой. Увидеть ниже. Таким образом, в основном вам просто нужно настроить информацию SCM и какие цели вы хотите запустить, а Дженкинс все выяснит.

Этот тип проекта может автоматически предоставлять следующие функции:

Архивные артефакты, созданные сборкой

Опубликовать результаты тестирования

Запуск заданий для проектов, которые являются зависимыми от нисходящего потока

Разверните свои артефакты в репозиторий Maven

Результаты тестирования прорыва по модулю

Необязательно перестраивать только измененные модули, ускоряя ваши сборки

Автоматическое построение цепочки из зависимостей модулей

Дженкинс читает зависимости вашего проекта от вашего POM, и если они также построены на Jenkins, триггеры настроены таким образом, что новая сборка в одной из этих зависимостей автоматически запустит новую сборку вашего проекта. Дженкинс понимает все виды зависимостей в POM. А именно, родительский POM

```
<dependencies> section of your project
<plugins> section of your project
<extensions> section of your project
<reporting> section of your project
```

Этот процесс учитывает версии, поэтому вы можете иметь несколько версий / ветвей вашего проекта на одном Jenkins и правильно определять зависимости. **Обратите внимание**, что диапазоны версий зависимостей не поддерживаются, см. [<https://issues.jenkins-ci.org/browse/JENKINS-2787>][1] по этой причине.

Эта функция может быть отключена по требованию - см. Параметр конфигурации Сборка при каждом создании зависимости SNAPSHOT

Монтаж :

1. войдите в Управление Jenkins >> configure System

2. в maven tab "Нажмите на установку maven

Вы можете либо заставить Jenkins автоматически установить определенную версию Maven, либо указать путь к локальной установке Maven (вы можете настроить столько версий Maven для своих проектов сборки, сколько захотите, и использовать разные версии Maven для разных проектов. Если вы установите флажок «Установить автоматически», Jenkins загрузит и установит запрошенную версию Maven для вас и установит ее в каталог инструментов в домашнем каталоге Jenkins.

Как это использовать

Сначала вы должны настроить установку Maven (этот шаг можно пропустить, если вы используете DEV @ cloud). Это можно сделать, перейдя на экран конфигурации системы (Manage Jenkins-> Configure System). В разделе «Установка Maven»: 1) нажмите кнопку «Добавить», 2) укажите имя «Maven 3.0.3», а затем 3) выберите версию из раскрывающегося списка.

Теперь Jenkins автоматически установит эту версию в любое время (например, на любых новых машинах), загрузив ее из Apache и разархивируя ее.

Создайте новую работу Maven:

1. Нажатие «Новое задание / Новый элемент» слева
2. Дайте ему имя
3. Выберите «Build a Maven 2/3 project»
4. Сохраните свою работу

Теперь вам нужно настроить свою работу

1. Выберите SCM, который вы хотите использовать (например, с помощью git)
2. выберите цель maven для вызова
3. добавить URL-адрес репозитория и учетные данные.
4. проверить частное личное сообщение maven:

Вы также можете определить путь custome для этого же.

5. Проект строительства

Создайте свой проект, нажав на сборку сейчас и нажмите на индикатор выполнения в левой руке «Build Executor Status», чтобы посмотреть, как jenkins устанавливают Maven, проверяют ваш проект и строят его с помощью maven.

Логирование:

<https://wiki.jenkins-ci.org/display/JENKINS/Logging>

Консоль сценариев:

Полезно для устранения неполадок, диагностики или пакетных обновлений заданий. Jenkins предоставляет консоль скриптов, которая дает вам доступ ко всем внутренним компонентам Jenkins. Эти скрипты написаны в Groovy, и вы найдете некоторые образцы из них на этой [странице](#).

Настройте простой проект сборки с помощью сценария Jenkins 2

Здесь мы создадим Groovy-конвейер в Jenkins 2, чтобы сделать следующие шаги:

- Проверяйте каждые 5 минут, если новый код был зачислен в наш проект
- Код заказа из репозитория SCM
- Maven компилирует наш код Java
- Запустите наши интеграционные тесты и опубликуйте результаты

Вот шаги, которые мы сделаем:

1. Убедитесь, что у нас есть версия 2.0 Jenkins (вы можете проверить это в нижнем правом углу страницы), например:

A small screenshot showing the text "Jenkins ver. 2.6" in a blue font, likely from the Jenkins dashboard.

2. На домашней странице Дженкинса нажмите « **Новый элемент** »
3. Введите название проекта и выберите « **Трубопровод** »
4. В разделе « **Сборка триггеров** » выберите вариант **опроса SCM** и добавьте следующие расписания CRON за 5 минут: `* / 5 * * * *`
5. В разделе « **Трубопровод** » выберите « **Сценарий трубопровода** » или « **Сценарий трубопровода** » из **SCM**
6. Если на предыдущем шаге вы выбрали **Pipeline Script из SCM**, теперь вам нужно указать URL-адрес **репозитория** SCM (Git, Mercurial, Subversion) в URL-адрес **репозитория**, например `http://github.com/example/example.git`. Вам также необходимо указать **путь** к скрипту вашего файла сценария Groovy в репозитории example.git,

например, `pipelines/example.groovy`

7. Скопируйте следующий код Groovy либо непосредственно в окне сценария Groovy, если ранее вы нажали на **Pipeline Script** или в вашем `example.groovy` если вы выбрали **Pipeline Script из SCM**

```
node('remote') {
    // Note : this step is only needed if you're using direct Groovy scripting
    stage 'Checkout Git project'
    git url: 'https://github.com/jglick/simple-maven-project-with-tests.git'
    def appVersion = version()
    if (appVersion) {
        echo "Building version ${appVersion}"
    }

    stage 'Build Maven project'
    def mvnHome = tool 'M3'
    sh "${mvnHome}/bin/mvn -B -Dmaven.test.failure.ignore verify"
    step([$class: 'JUnitResultArchiver', testResults: '**/target/surefire-reports/TEST-*.xml'])
}
def version() {
    def matcher = readFile('pom.xml') =~ '<version>(.)</version>'
    matcher ? matcher[0][1] : null
}
```

Здесь вы должны теперь скомпилировать и протестировать свой первый проект Jenkins, используя трубопровод Jenkins 2 Groovy.

Прочитайте Начало работы с дженкинсами онлайн: <https://riptutorial.com/ru/jenkins/topic/919/начало-работы-с-дженкинсами>

глава 2: Jenkins Groovy Scripting

Examples

Создать пользователя по умолчанию

1. Создать groovy файл по пути `$JENKINS_HOME/init.groovy.d/basic-security.groovy`

В домашнем каталоге Ubuntu 16 Jenkins в каталоге `/var/lib/jenkins`

2. Поместить в файл следующий код

```
#!/groovy

import jenkins.model.*
import hudson.security.*

def instance = Jenkins.getInstance()

def hudsonRealm = new HudsonPrivateSecurityRealm(false)

hudsonRealm.createAccount("admin_name", "admin_password")
instance.setSecurityRealm(hudsonRealm)
instance.save()
```

3. Перезапустить службу Дженкинса
4. После запуска Jenkins вам нужно удалить `$JENKINS_HOME/init.groovy.d/basic-security.groovy`

Отключить мастер установки

1. Открыть Дженкинс конфигурации по умолчанию файл и добавить в `JAVA_ARGS` следующий ключ `-Djenkins.install.runSetupWizard=false`

В Ubuntu 16 файлов по умолчанию помещается в `/etc/default/jenkins`

2. Создать groovy файл по пути `$JENKINS_HOME/init.groovy.d/basic-security.groovy`

В домашнем каталоге Ubuntu 16 Jenkins в каталоге `/var/lib/jenkins`

3. Поместить в файл следующий код

```
#!/groovy

import jenkins.model.*
import hudson.util.*;
import jenkins.install.*;

def instance = Jenkins.getInstance()
```

```
instance.setInstallState(InstallState.INITIAL_SETUP_COMPLETED)
```

4. Перезапустить службу Дженкинса

5. После запуска Jenkins вам нужно удалить `$JENKINS_HOME/init.groovy.d/basic-security.groovy`

После этого Jenkins не просит вас подтвердить, что вы являетесь администратором, и вы не увидите страницу установки плагинов.

Как получить информацию о примере Дженкинса

Откройте консоль сценариев экземпляров jenkins [http:// yourJenkins: следующий порт / сценарий](http://yourJenkins:следующий порт / сценарий) - пример того, как получить информацию об этом экземпляре. скопируйте код в консоль и нажмите «Запустить».

```
/* This scripts shows how to get basic information about Jenkins instance */
def jenkins = Jenkins.getInstance()
println "Jenkins version: ${jenkins.getVersion()}"
println "Available JDKs: ${jenkins.getInstance().getJDKs()}"
println "Connected Nodes:"
jenkins.getNodes().each{
    println it.displayName
}
println "Configured labels: ${jenkins.getLabels()}"
```

В этом примере вы увидите информацию о версии Jenkins, JDK, агентах (slaves) и ярлыках.

Как получить информацию о работе Дженкинса

Откройте консоль сценариев экземпляров jenkins [http:// yourJenkins: порт / сценарий](http://yourJenkins:порт / сценарий), приведенный ниже, является примером того, как получить информацию о конкретной работе. скопируйте код в консоль, измените имя job на требуемое задание и нажмите «Запустить».

```
/*This script shows how to get basic information about a job and its builds*/
def jenkins = Jenkins.getInstance()
def jobName = "myJob"
def job = jenkins.getItem(jobName)

println "Job type: ${job.getClass()}"
println "Is building: ${job.isBuilding()}"
println "Is in queue: ${job.isInQueue()}"
println "Last successfull build: ${job.getLastSuccessfulBuild()}"
println "Last failed build: ${job.getLastFailedBuild()}"
println "Last build: ${job.getLastBuild()}"
println "All builds: ${job.getBuilds().collect{ it.getNumber() }}"
```

сначала мы получаем экземпляр объекта Jenkins, а затем с помощью этого экземпляра получаем объект задания (item). из объекта задания мы можем получить различную

информацию, такую как: она в настоящее время строится, находится ли она в очереди, последней строкой, последней строкой по статусу и намного больше.

Прочитайте Jenkins Groovy Scripting онлайн: <https://riptutorial.com/ru/jenkins/topic/7562/jenkins-groovy-scripting>

глава 3: Настройка Jenkins для автоматизации сборки iOS.

Вступление

Теперь вы можете определить процесс непрерывной интеграции и непрерывной доставки (**CI / CD**) как код с Jenkins 2.0 для ваших проектов в iOS 10. Мероприятия, такие как сборка, тестирование, покрытие кода, стиль проверки, отчеты и уведомления, могут быть описаны только в одном файле.

Чтобы прочитать полную статью, перейдите на [Pipeline в Jenkins 2.0 как Код для iOS 10 и XCode 8](#)

параметры

| параметр | подробности |
|-------------------|---------------------------------------------------------------------------------------------------|
| node ('iOS Node') | Узел Дженкинса с Mac OS. Если Jenkins установлен в Mac OS, используйте <code>node { }</code> |

замечания

Статья написана на обоих языках: английском и испанском.

Examples

Пример таблицы времени

Исходный код можно [клонировать](#) или [загружать из GitHub](#) для его проверки.

```
node('iOS Node') {  
  
    stage('Checkout/Build/Test') {  
  
        // Checkout files.  
        checkout([  
            $class: 'GitSCM',  
            branches: [[name: 'master']],  
            doGenerateSubmoduleConfigurations: false,  
            extensions: [], submoduleCfg: [],  
            userRemoteConfigs: [[  
                name: 'github',  
                url: 'https://github.com/mmorejón/time-table.git'  
            ]]  
        ])  
    }  
}
```

```

    ]]
  })

  // Build and Test
  sh 'xcodebuild -scheme "TimeTable" -configuration "Debug" build test -destination
"platform=iOS Simulator,name=iPhone 6,OS=10.1" -enableCodeCoverage YES |
/usr/local/bin/xcpretty -r junit'

  // Publish test results.
  step([$class: 'JUnitResultArchiver', allowEmptyResults: true, testResults:
'build/reports/junit.xml'])
}

stage('Analytics') {

  parallel Coverage: {
    // Generate Code Coverage report
    sh '/usr/local/bin/slather coverage --jenkins --html --scheme TimeTable
TimeTable.xcodeproj/'

    // Publish coverage results
    publishHTML([allowMissing: false, alwaysLinkToLastBuild: false, keepAll: false,
reportDir: 'html', reportFiles: 'index.html', reportName: 'Coverage Report'])

  }, Checkstyle: {

    // Generate Checkstyle report
    sh '/usr/local/bin/swiftlint lint --reporter checkstyle > checkstyle.xml || true'

    // Publish checkstyle result
    step([$class: 'CheckStylePublisher', canComputeNew: false, defaultEncoding: '',
healthy: '', pattern: 'checkstyle.xml', unhealthy: ''])
  }, failFast: true|false
}

stage ('Notify') {
  // Send slack notification
  slackSend channel: '#my-team', message: 'Time Table - Successfully', teamDomain: 'my-
team', token: 'my-token'
}
}

```

Прочитайте [Настройка Jenkins для автоматизации сборки iOS](https://riptutorial.com/ru/jenkins/topic/8868/настройка-jenkins-для-автоматизации-сборки-ios-). онлайн:

<https://riptutorial.com/ru/jenkins/topic/8868/настройка-jenkins-для-автоматизации-сборки-ios->

глава 4: Настройка автоматизации сборки для iOS с использованием Shenzhen

Examples

Настройка автоматизации сборки iOS с использованием Shenzhen

Часть I: настройка компьютера Mac для использования shenzhen

Перейти к терминалу

Установить Шэньчжэнь

```
sudo gem install shenzhen
```

```
sudo gem install nomad-cli
```

Загрузить утилиту командной строки XCode

```
xcode-select --install
```

Всплывающее окно отображается с приведенным ниже текстом

Для команды xcode-select требуются средства разработки командной строки. Вы хотите установить инструменты сейчас? "

Нажмите - Установить

Создать каталог проекта

gitclone ваш проект

```
git clone https://akshat@bitbucket.org/company/projectrepo.git
```

Построить проект, используя команду ниже

```
ipa build --verbose
```

PS: Если вы видите какие-либо ошибки, выберите профиль Active Provisioning Profile и зафиксируйте файлы проекта. и снова выполните `ipa build --verbose` .

Прочитайте Настройка автоматизации сборки для iOS с использованием Shenzhen онлайн: <https://riptutorial.com/ru/jenkins/topic/8002/настройка-автоматизации-сборки-для-ios-с-использованием-shenzhen>

глава 5: Настройка автоматической загрузки Git на успешную сборку в Jenkins

Вступление

В этом документе вы найдете инструкции по настройке задания Jenkins, которое позволяет пользователю настраивать автоматическое нажатие на успешную сборку. Операцию push можно контролировать пользователем. Пользователь может выбрать, хотите ли они выполнить операцию автоматического нажатия на успешную сборку или нет.

Examples

Настройка автозапуска

Создайте задание сборки (согласно вашему требованию). В этом примере я создал работу по фристайлу (AutoPush) для выполнения сборки ANT.

Мы собираемся создать две переменные: PUSH (параметр выбора) и TAG_NUMBER (параметр String).

Мы можем выбрать значение ДА или НЕТ для PUSH, это решит, следует ли нажимать код на тег или нет при успешной сборке.

Мы можем указать имя тега (пример 1.0.1) для TAG_NUMBER для создания нового тега (например, 1.0.1) в удаленном репозитории с тем же именем или указать существующее имя тега для обновления существующего тега.

Project AutoPush

This build requires parameters:

PUSH

YES ▼

Controls whether to push the code to a new release tag or not.

TAG_NUMBER

1.0.1

Enter a new tag number to create a new tag or enter an existing tag number to update an existing tag.

Build

Теперь перейдем к настройке задания.

1. Установите флажок «Этот проект параметризован» и создайте параметр выбора «PUSH» и укажите YES и NO в качестве выбора. Этот параметр решит, хотите ли вы

нажимать код на определенный тег / релиз или нет.

☒ This project is parameterized

Choice Parameter

Name

PUSH

Choices

YES
NO

Description

Controls whether to push the code to a new release tag or not.

[Plain text]

[Preview](#)

2. Затем создайте параметр String под названием «TAG_NUMBER», используя этот параметр, мы можем указать новый номер тега для создания нового тега или указать существующий номер тега для обновления существующего тега.

String Parameter

Name

TAG_NUMBER

Default Value

Description

Enter a new tag number to create a new tag or enter an existing tag number to update an existing tag.

[Plain text]

[Preview](#)

Add Parameter

3. В разделе «Управление исходным кодом» выберите Git и укажите URL-адрес репозитория. Этот репозиторий содержит исходный код, который вы собираетесь создать, и после успешной сборки тег release будет создан в том же хранилище.

Source Code Management

☐ None

☒ Git

Repositories

Repository URL

Credentials



Advanced

Add Repository

Branches to build

Branch Specifier (blank for 'any')

Add Branch

4. После добавления сведений о репозитории щелкните по расширенному и укажите имя в репозитории, который позже будет передан в плагин Git Publisher для идентификации репозитория.

Source Code Management

☐ None

☒ Git

Repositories

Repository URL

Credentials



Advanced

Add Repository

Branches to build

Branch Specifier (blank for 'any')

Add Branch

Source Code Management

☐ None

☒ Git

Repositories

Repository URL

Credentials

Name

Refspec

Branches to build

Branch Specifier (blank for 'any')

5. Затем добавьте шаг сборки. В этом примере я создаю проект ANT.

Build

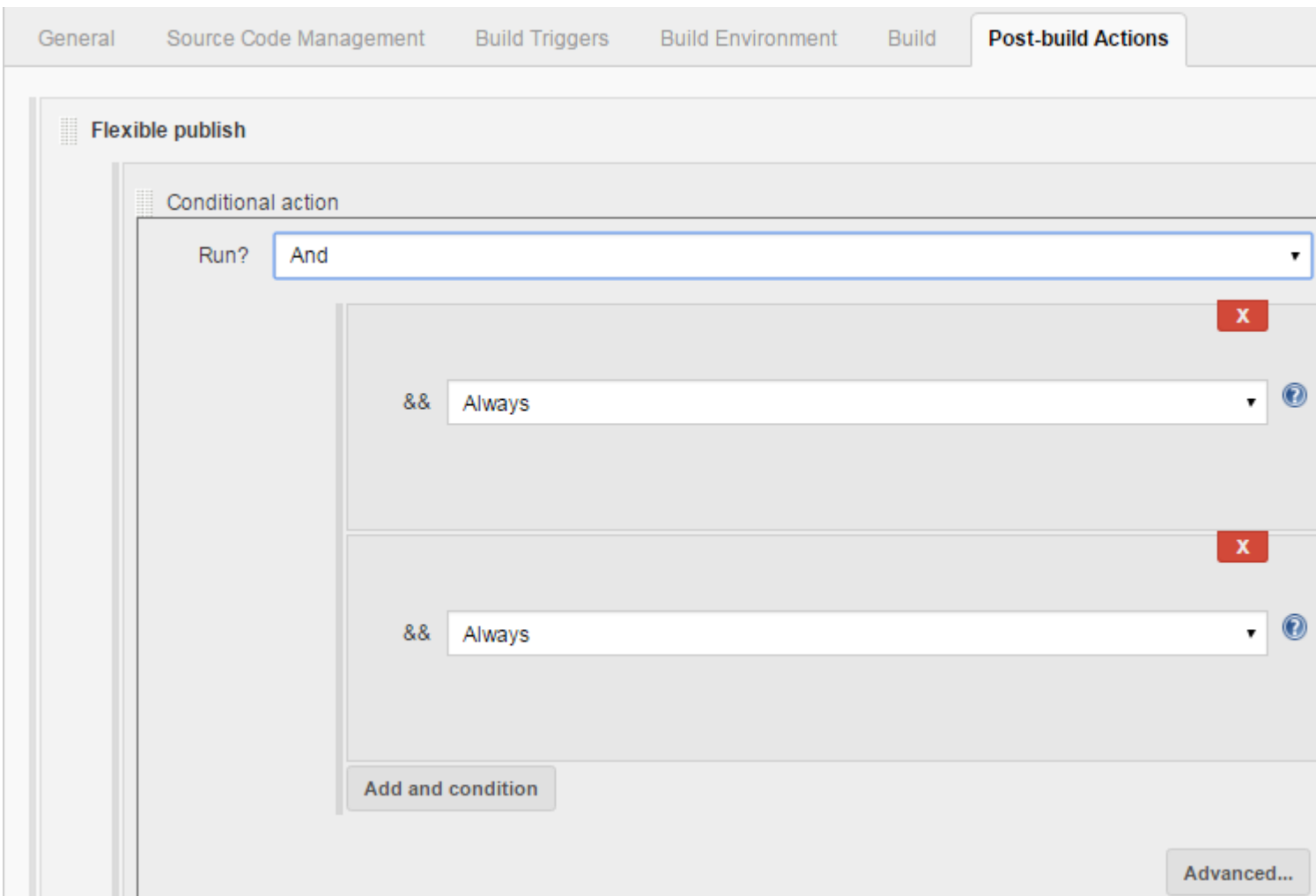
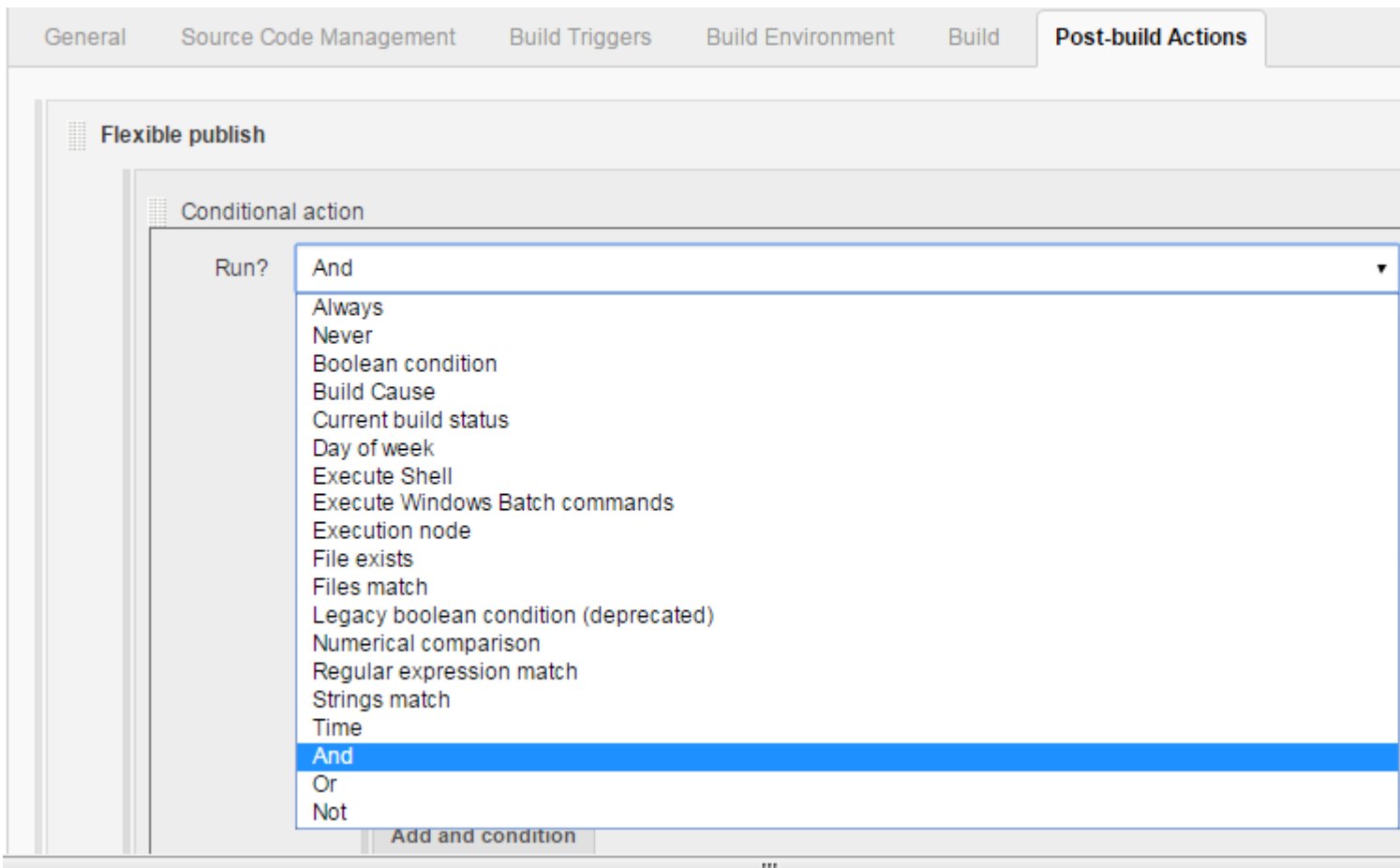
Execute shell

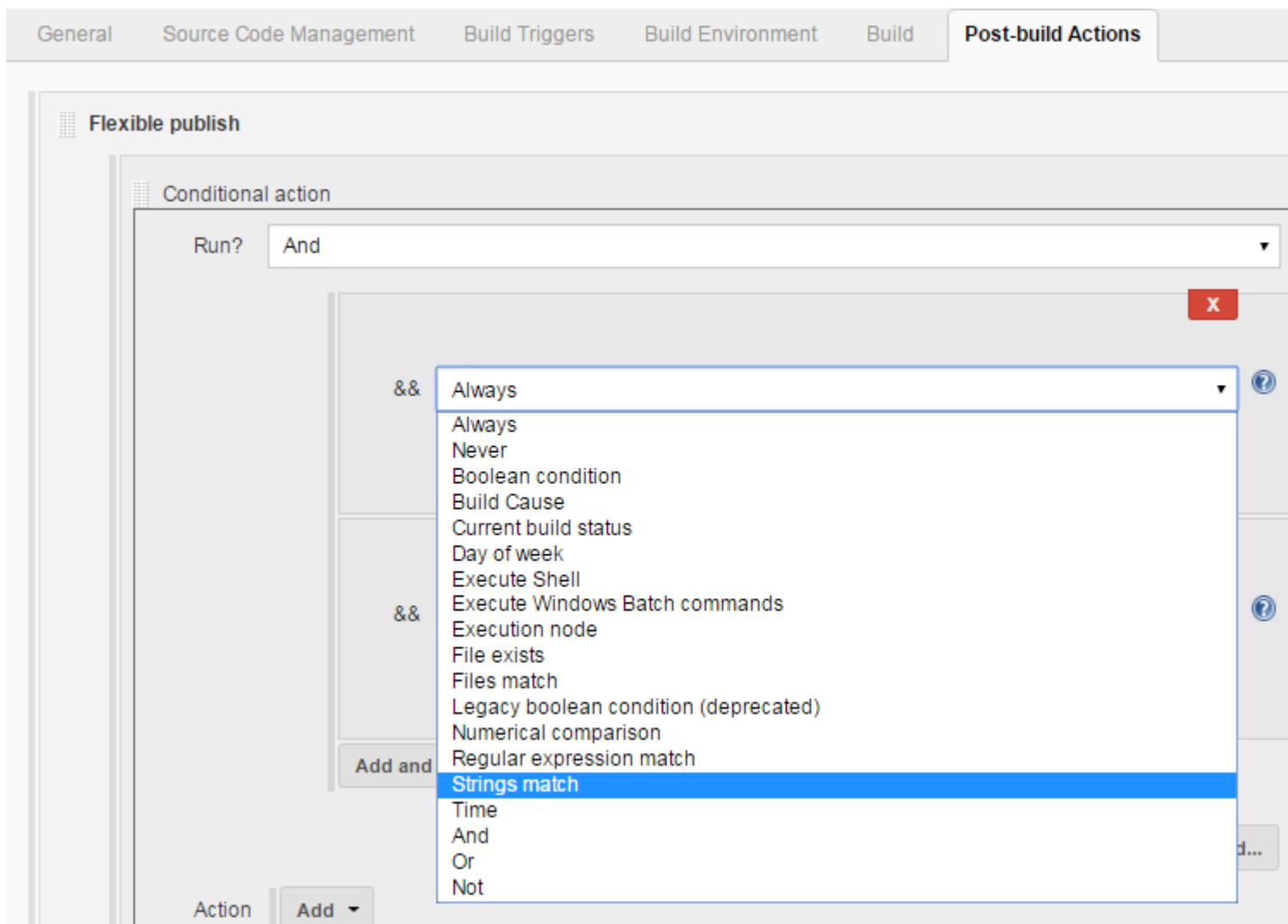
Command

```
cd /data/core/proj/  
ant |
```

See [the list of available environment variables](#)

6. Теперь в разделе «Действия после сборки» выберите плагин «Flexi Publish». Выберите значение «И» в раскрывающемся списке для условного действия (Запустить?). Затем выберите «String Match» из раскрывающегося списка для условия Run (&&).





7. После выбора соответствия строки укажите \$ PUSH как значение String 1 и YES как значение String 2. Поэтому, когда вы запустите сборку, если вы выберете значение PUSH как ДА, он будет сравнивать строку 1 (= \$ PUSH) и строку 2 (= YES) и запустить операцию нажатия Git, и если вы выберете «НЕТ», это не будет активировать операцию нажатия Git.

```
Choose the value of PUSH -> YES OR NO -> Chosen value "YES"
then, $PUSH = YES
AS String 1 = $PUSH => String 1 = YES
Again, String 2 = YES, hence String 2 == String 1 (String match)
Then, trigger the Git push action.
```

General Source Code Management Build Triggers Build Environment Build **Post-build Actions**

Run? And

Strings match

String 1 \$PUSH

String 2 YES

Case insensitive ☐

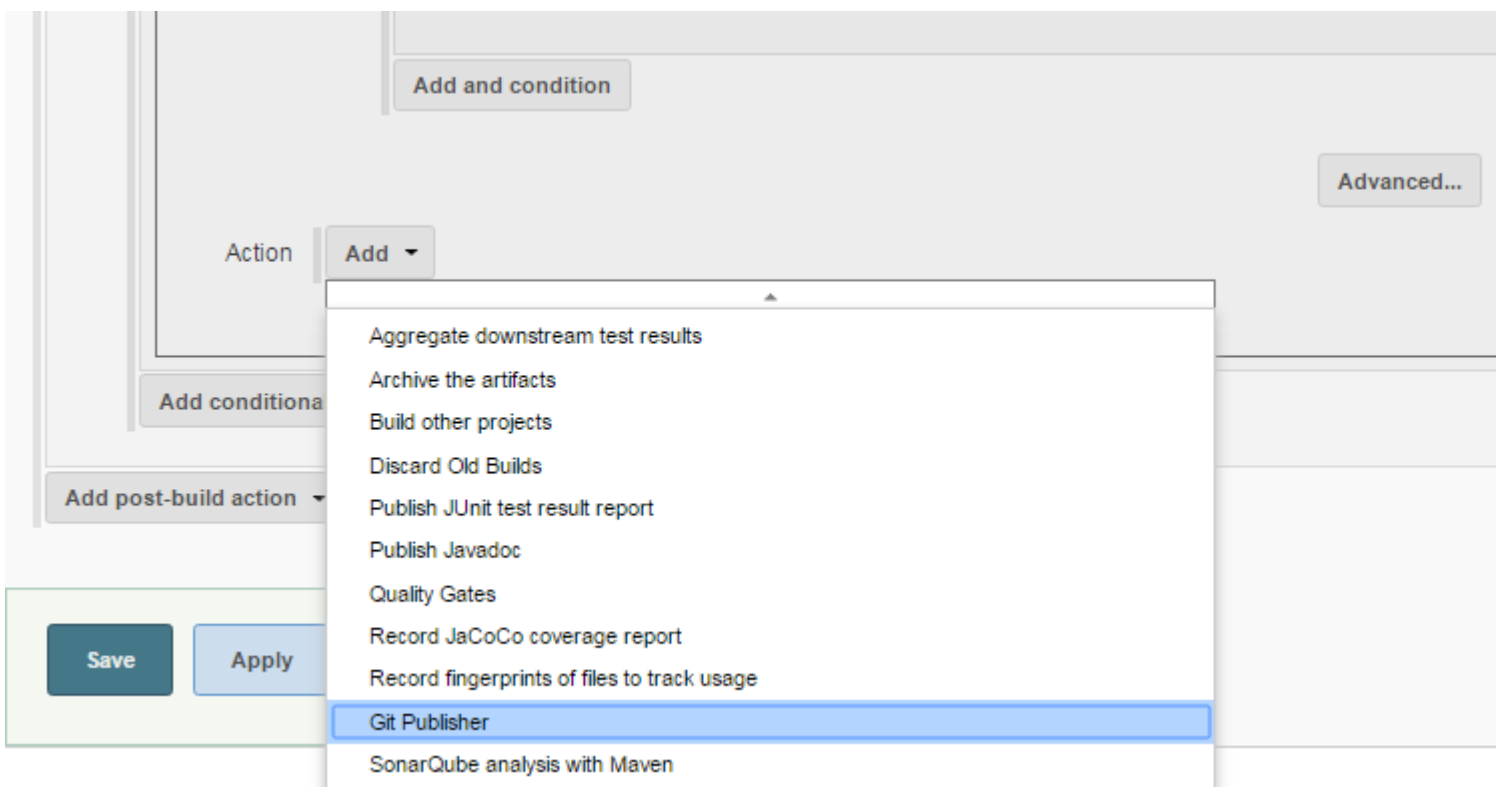
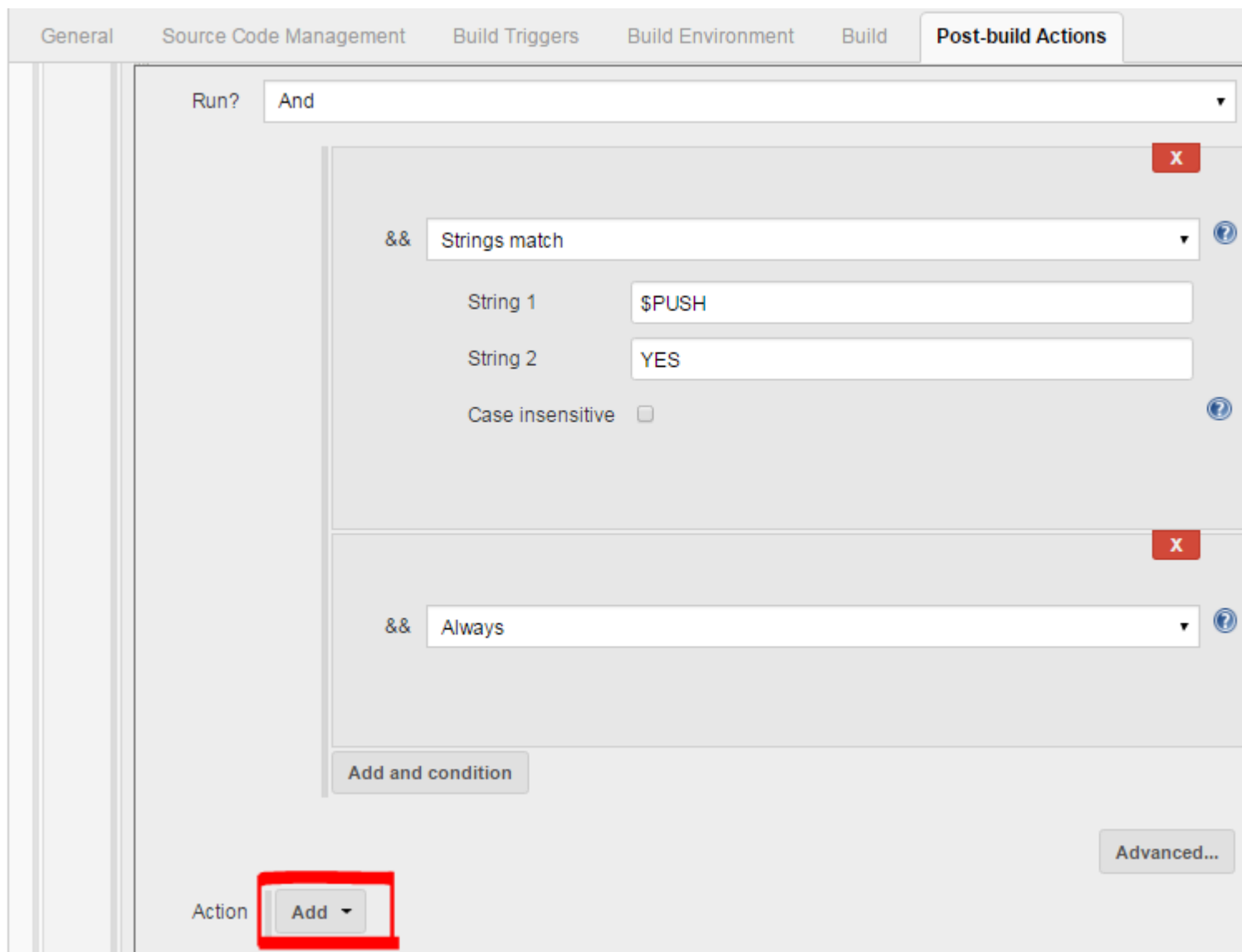
Always

Add and condition

Advanced...

Action Add

8. Теперь нажмите «Добавить раскрывающийся список», чтобы добавить действие издателя Git, которое будет запущено на основе условия соответствия строки.



9. После выбора Git Publisher выполните настройку следующим образом:

The screenshot shows the Jenkins configuration page for the 'Git Publisher' action. The interface is divided into several sections: 'Push Only If Build Succeeds' (checked), 'Merge Results' (unchecked), 'Force Push' (checked), and 'Tags'. The 'Tags' section is expanded, showing a 'Conditional action' configuration. The 'Tag to push' is set to '\${TAG_NUMBER}', the 'Tag message' is empty, 'Create new tag' is checked, 'Update new tag' is checked, and the 'Target remote name' is 'stableTag'. There is an 'Add Tag' button. Below this, the 'Branches' section is also expanded, showing a 'Conditional action' configuration. The 'Branch to push' is set to 'master', and the 'Target remote name' is 'stableTag'. There is an 'Add Branch' button. At the bottom left, there are 'Save' and 'Apply' buttons.

После настройки сохраните задание, и все будет готово.

Прочитайте [Настройка автоматической загрузки Git на успешную сборку в Jenkins онлайн](https://riptutorial.com/ru/jenkins/topic/8972/настройка-автоматической-загрузки-git-на-успешную-сборку-в-jenkins):
<https://riptutorial.com/ru/jenkins/topic/8972/настройка-автоматической-загрузки-git-на-успешную-сборку-в-jenkins>

глава 6: Плагин для стратегии ролей

Examples

конфигурация

Управление ролями

Глобальные роли. Создание ролей с выбранным набором функций Jenkins, например. Обычно для проекта разработки могут быть созданы 2 роли.

1. Разработчик-Глобальная роль может быть установлена только в **целом** : Чтение
2. ProjectOwner - Глобальная роль может быть установлена в **целом** : Чтение

Это ограничивает разработчика и владельца проекта доступ ко всем функциям Jenkins.



Manage and Assign Roles

Global roles

| Role | Overall | | | | | Credentials | | | | |
|--------------------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|
| | Administer | ConfigureUpdateCenter | Read | RunScripts | UploadPlugins | Create | Delete | ManageDomains | Update | View |
| <input checked="" type="checkbox"/> Developer | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> ProjectOwner | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> admin | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |

Role to add



Add

Роли проекта. Создавайте роли, ограничивая доступ пользователей к функциям задания и учетных данных jenkins с помощью регулярных выражений.

Например, для проекта разработки «MyProjectA»; владельцы проектов должны иметь полные разрешения на работу, и разработчикам необходимо создать доступ к заданиям Jenkins. Поэтому мы создаем ниже роли:

- **ProjectA_admin** - проверить все параметры в разделе «Работа». *Создание, отмена, настройка, создание, удаление, обнаружение, перемещение, чтение, рабочее пространство*
- **ProjectA_dev** - параметры проверки Сборка, Отмена, Чтение, Рабочее пространство в

Project roles

| Role | Pattern | Credentials | | | | | Job | | | | | | |
|--------------------------------------------------------------------------------------------------|---------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|
| | | Create | Delete | ManageDomains | Update | View | Build | Cancel | Configure | Create | Delete | Discover | Move |
|  ProjectA_admin | MyProjectA.*. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
|  ProjectA_dev | MyProjectA.*. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

Role to add

Pattern

Add

Чтобы ограничить над проектами соответствующих владельцев проектов и разработчиков, все задания должны следовать заранее определенному шаблону.

Предположим, что для MyProjectA требуются 3 задания создания jenkins:
MyProjectA_Dev_Build, MyProjectA_QA_Build, MyProjectA_Nightly_Sonar_Analysis

Чтобы ограничить владельца проекта и разработчиков проекта «MyProjectA» выше заданий сборки, укажите «Шаблон» как **MyProjectA.*.**

Назначить роли

Помогает назначать пользователей или группы проектов соответствующим глобальным или проектным ролям. Например, чтобы назначить глобальную роль разработчика Gautam для разработчика, укажите имя пользователя «Gautam», нажмите «Добавить» и установите флажок рядом с «Гаутом» и ниже глобальной роли разработчика.



Assign Roles

Global roles

| User/group | Developer | ProjectOwner | admin |
|------------|-------------------------------------|--------------------------|-------------------------------------|
| admin | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| Anonymous | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| gautam | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

User/group to add

Add

Project roles

| User/group | ProjectA_admin | ProjectA_dev |
|------------|--------------------------|-------------------------------------|
| Anonymous | <input type="checkbox"/> | <input type="checkbox"/> |
| gautam | <input type="checkbox"/> | <input checked="" type="checkbox"/> |

User/group to add

Add

Аналогичным образом добавьте пользователя в роли проекта и выберите соответствующие роли проекта, чтобы назначить требуемые роли проекта.

Если вы заметили ниже скриншоты, вы можете видеть, что пользователь gautam имеет доступ только к проектам, начинающимся с MyProjectA. Кроме того, доступ пользователя ограничен для сборки и настройки.

| All | | MyProjectA | | |
|-----|---|---------------------------------------------------|--------------|--------------|
| S | W | Name | Last Success | Last Failure |
| | | MyProjectA Dev Build | N/A | N/A |
| | | MyProjectA Nightly Sonar Analysis | N/A | N/A |
| | | MyProjectA QA Build | N/A | N/A |

 [Back to Dashboard](#)

 [Status](#)

 [Changes](#)

 [Workspace](#)

 [Build Now](#)

Project MyProjectA_Dev_Build

 [Workspace](#)

 [Recent Changes](#)

 **Build History** [trend](#) 

 [RSS for all](#)  [RSS for failures](#)

Permalinks

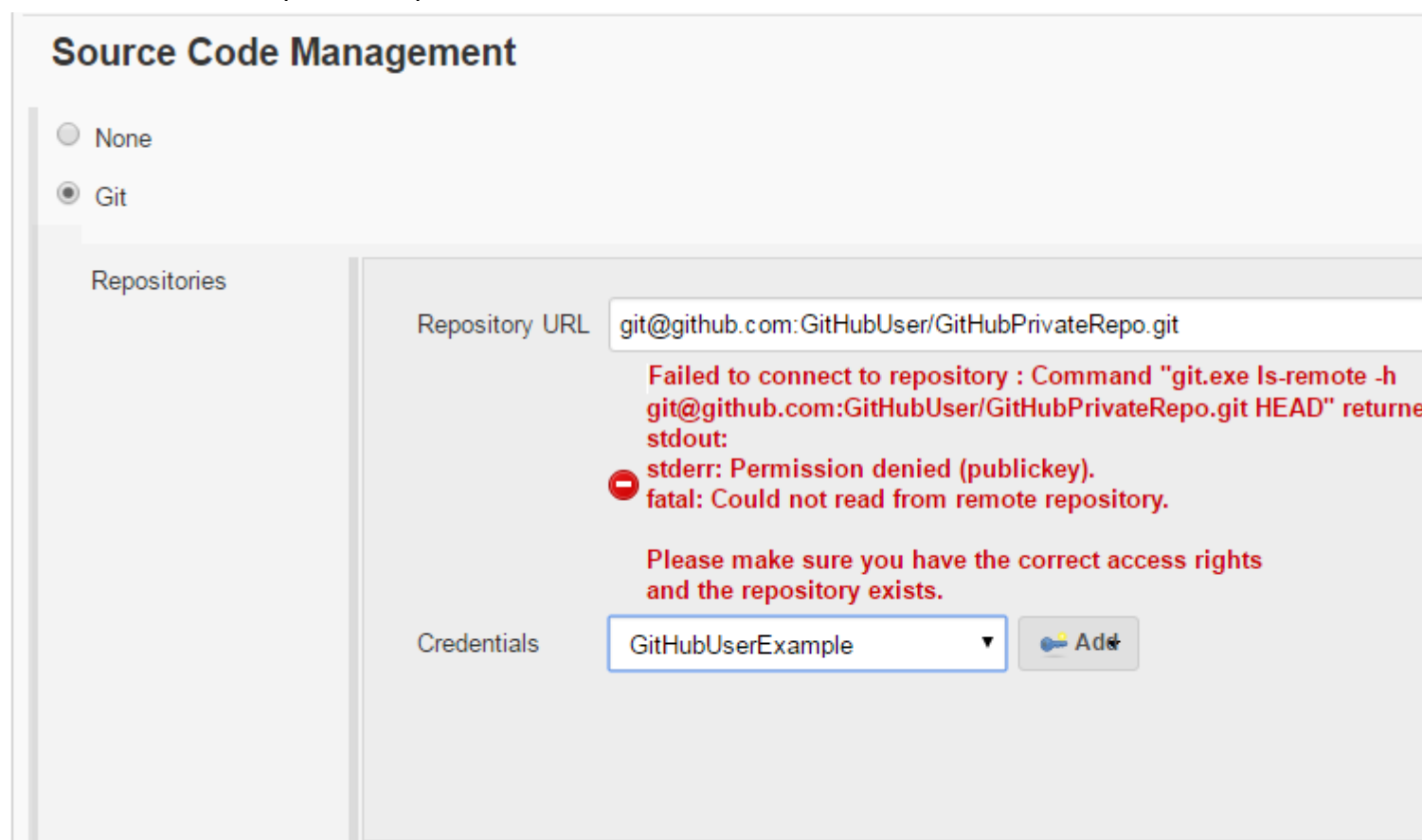
Прочитайте Плагин для стратегии ролей онлайн: <https://riptutorial.com/ru/jenkins/topic/5741/плагин-для-стратегии-ролей>

глава 7: Установите Jenkins на Windows с поддержкой SSH для частных репозиториев GitHub

Examples

Неисправность запросов GitHub

Вне коробки установки Jenkins с плагинами Git и SSH не будут работать при попытке вытащить частный репозиторий из GitHub.



PSEXec.exe PS Tool от Microsoft

Первым шагом для исправления этой проблемы я нашел загрузку [PSTools](#) и извлечение инструментов в удобное место на сервере сборки (например, c: \ Programs \ PSTools, там я извлек мой).

Создайте новый ключ SSH только для Jenkins, используя PSEXec или PSEXec64

1. Сначала откройте командную строку и «Запуск от имени администратора».
2. После открытия командной строки перейдите в каталог PSTools.
3. Из командной строки нам нужно запустить git-bash с помощью PSEXec или PSEXec64 в качестве локальной службы, которую Jenkins запускает на сервере сборки по умолчанию.
4. Мы будем использовать ключ -i для запуска PSEXec в качестве интерактивного и -s для запуска git-bash в качестве локальной службы
5. Следуйте инструкциям по созданию ssh-ключа в GitHub - [Генерация нового ключа SSH и добавление его в ssh-agent](#)
6. Если вы находитесь в 64-битной системе Windows, скопируйте папку .ssh в C: \ Windows \ SysWOW64 \ config \ systemprofile.ssh (это не было необходимо для моей 64-битной системы Windows, но там, где некоторые инструкции, указывающие файлы .ssh следует хранить там, что-то иметь в виду, если у вас все еще есть проблемы).
7. Добавьте общедоступный ключ SSH к вашим ключам github.

Your Commandline should look similar to this:

```
C:\Programs\PSTools> PSEXec.exe -i -s C:\Programs\Git\git-bash
```

Создайте учетные данные Jenkins

Твердая часть закончилась! Теперь просто создайте учетные данные, которые будут использоваться в Jenkins. Используйте свое имя пользователя и парольную фразу, используемую для создания ключа SSH.



Jenkins Credentials Provider: Jenkins



Add Credentials

Domain

Global credentials (unrestricted)

Kind

SSH Username with private key

Scope

Global (Jenkins, nodes, items, all child items, etc)

Username

GitHubUserExample

Private Key

☐ Enter directly

☐ From a file on Jenkins master

☒ From the Jenkins master ~/.ssh

Passphrase

.....|

ID

Description

Add

Cancel

Это то, что теперь должно выглядеть (с вашим собственным реестром github и именем пользователя:

Source Code Management

☐ None

☒ Git


Repositories

Repository URL

Credentials [Add](#)

Запустите запрос на тест-тест, чтобы проверить его и выполнить.

Сохраните и запустите запрос на тест-драйв, и у вас больше не будет никаких проблем с тем, что Jenkins использует SSH на вашей машине для сборки Windows.



Jenkins

Jenkins ▶ ▶

[Back to Dashboard](#)

[Status](#)

[Changes](#)


[Workspace](#)


[Build Now](#)

[Delete Project](#)

[Configure](#)


[Move](#)

 [Workspace](#)


 [Recent Changes](#)

Permalinks

- [Last build \(#1\). 47 min ago](#)
- [Last stable build \(#1\). 47 min ago](#)
- [Last successful build \(#1\). 47 min ago](#)
- [Last completed build \(#1\). 47 min ago](#)

 **Build History** [trend](#)

x

 **#1** Oct 25, 2016 10:31 AM

 [RSS for all](#)  [RSS for failures](#)

Прочитайте Установите Jenkins на Windows с поддержкой SSH для частных репозиториев GitHub онлайн: <https://riptutorial.com/ru/jenkins/topic/7626/установите-jenkins-на-windows-с-поддержкой-ssh-для-частных-репозиториев-github>

кредиты

| S. No | Главы | Contributors |
|-------|-------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1 | Начало работы с дженкинсами | acalb , Community , Gautam Jose , Girish Kumar , Katu , Pablo , Pom12 , Priyanshu Shekhar , Rogério Peixoto , S.K. Venkat , Seb , Tyler |
| 2 | Jenkins Groovy Scripting | RejeeshChandran , serieznyi , Tidhar Klein Orbach |
| 3 | Настройка Jenkins для автоматизации сборки iOS. | Manuel Morejón |
| 4 | Настройка автоматизации сборки для iOS с использованием Shenzhen | Ichthyocentaurs |
| 5 | Настройка автоматической загрузки Git на успешную сборку в Jenkins | ANIL |
| 6 | Плагин для стратегии ролей | Gautam Jose |
| 7 | Установите Jenkins на Windows с поддержкой SSH для частных репозиторий GitHub | Riana |