# Отчёт по тестовому заданию
# Для стажера в копрорацию EMC, отдел DevOps

Старался **Шкаруба Н.Е.**

---

## Требования:

1. Развернуть Jenkins LTS (https://jenkins.io/) на Linux.
2. Разграничить права доступа
3. Сделать pipeline job, которая предоставляет сервис сборки Дженкинс плагина на примере robotframework plugin: https://github.com/jenkinsci/robot-plugin:
    a. Пользователь должен иметь возможность выбрать версию JDK 7/8/9 с помощью которой собирать.
    b. Пользователь должен иметь возможность скачать собранный архив.
    c. (Доп.) Джоба должна запускать юнит тесты и отображать их результаты.
    d. (Доп.) Параллельные запуски джобы не должны приводить к конфликтам.

**Результатом работы** над заданием является подробный отчет о всех шагах выполнения с приложением скрипта джобы.
**Время выполнения**: 1 неделя (например, при получения задания в понедельник, вам нужно отправить отчет до начала вторника следующей недели).

## Выполнение работы:

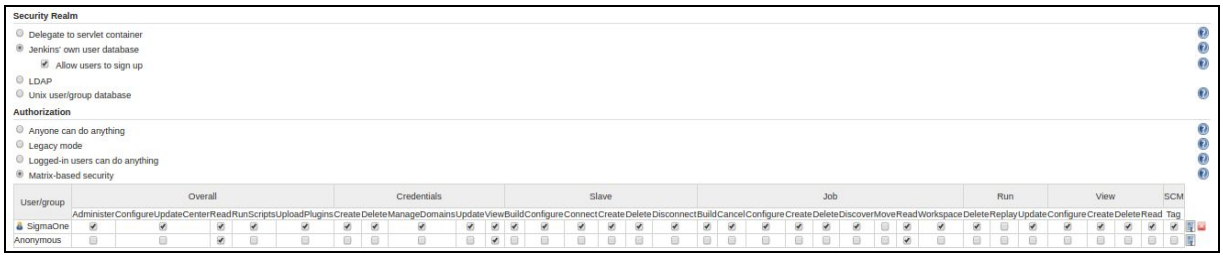
1. **Сбор информации о предметной области, а именно:**
    - Что такое Jenkins  - Continuous integration tool, или удобная система сборки и поддержания проекта
    - Что такое Jenkins плагины - Jenkins по своей сути, очень скромен по функционалу, но он отлично расширяется. Сообществом было написано очень много плагинов для удовлетворения различных потребностей.
    - Что такое Jenkins jobs - Джоба - Единица работы в jenkins. Пример джобы - сборка проекта или прогон тестов с ведением лога.
    - Просмотр гайда Jenkins tutorial (https://www.youtube.com/watch?v=Lxd6JMMxuwo)
    - Что такое Jenkins Pipleline plugin - Плагин, предоставляющий ещё один тип Джоб, а именно, Джобы, расширяемые Gradle скриптами. Нужен для того, чтобы меньше накликивать :)

    Теория будет изучена подробней к интервью!

2. **Развёртывание Jenkins на локальной машине:**
   Установка Jenkins, всех его зависимостей, и маленького сервлет-контейнера Winstone
   $ sudo apt-get install jenkins
   Удостоверение, что всё правильно установилось - Просмотр веб-интерфейса на localhost:8080. Всё работает.
   **Пункт 1 выполнен.**

3. **Разграничение прав доступа:**
   В гайде, который я просмотрел, автор делал себя администратором, а остальных пользователей лишь “зрителями”. Мне идея понравилась. Идём в Manage Jenkins -> Configure Global Security.
   Security Realm - отмечаем “Jenkins own database” и “Allow users to sign up”. Apply. Это всё нужно для того, чтобы Jenkins хранил пользователей у себя в бд, и можно было регистрироваться.
   Регистрируем себя в системе и входим. Теперь можно сделать себя администратором, а остальных пользователей лишь “зрителями”. Идём в Authorization - Matrix-based security. Выставляем себе все права, а остальным пользователям лишь права на чтение.

   Итоговая конфигурация Global security:



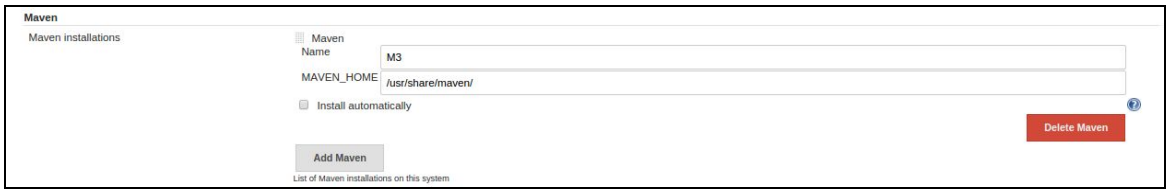

   **Пункт 2 выполнен.**

4. **Настройка окружения + установка необходимых плагинов**
   Для выолнения задания понядобятся два плагина - **Git plugin**(для интеграции с Гитом) и **Pipeline plugin**(для автоматизирования Джоб gradle скриптами)
   Manage Jenkins ->Plugin manager -> выбираем Pipeline plugin, Git plugin -> install

   Так, теперь необходимо указать пути к используемым утилитам. Для будущего выбора в джобе версий jdk, необходимо сконфигурировать их в Manage jenkins -> Configure System


   jdk 7 и 8 Jenkins умеет автоматически скачивать с сайта Oracle, а jdk 9 ещё оффициально не выпустили, поэтому я руками скачиваю jdk-9_Early_access_release и указываю к нему путь.

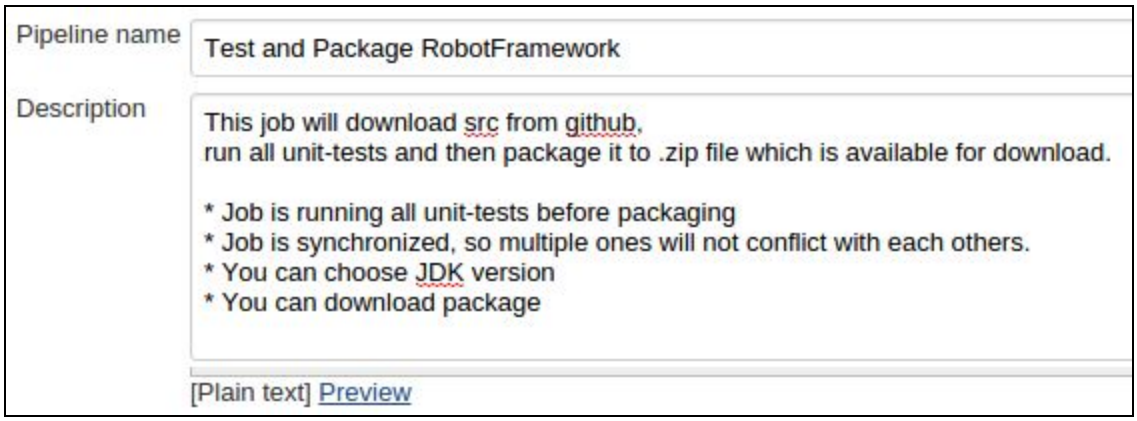
   Мэйвен у меня уже стоит, поэтому просто руками указываю к нему путь.



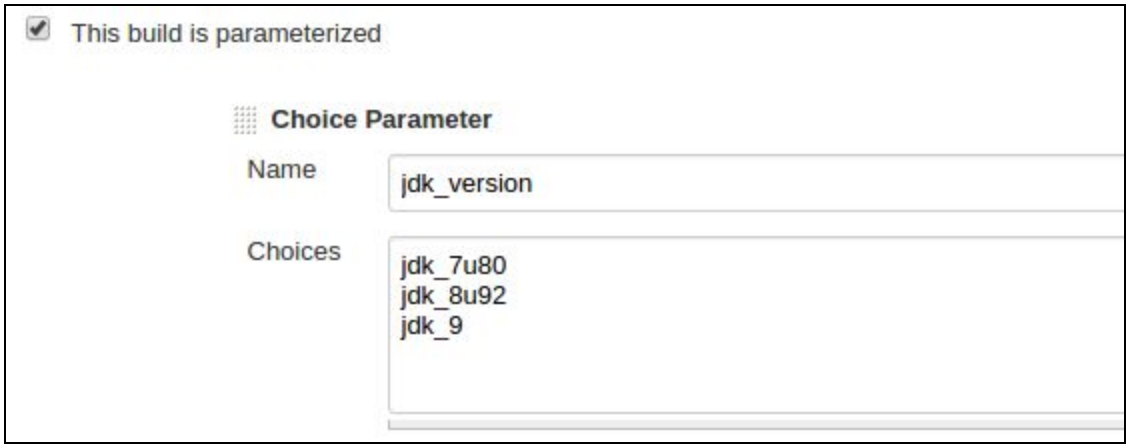

   Т.к. Я установил Git plugin, то Гит уже настроен.

5. **Создание Джобы, выполнение дополнительных пунктов**
   Jenkins - New Item -> Pipeline.

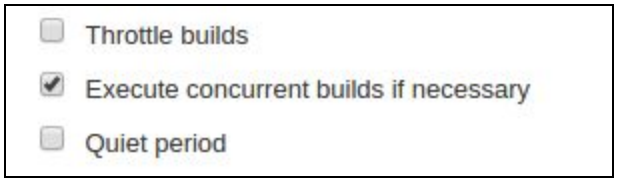
   Настройка описания



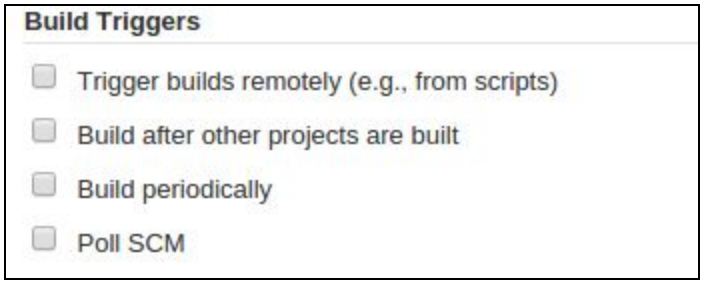

   Настройка Параметров билда. Задача была - выбирать версию jdk. Указываем имя параметра и 3 текстовых значения. В последствии jdk_version будет доступен как текстовая переменная gradle скрипта.




   Выставляем  Execute concurrent builds if necessary, для доп.задания.



Триггеры, автомтически запускающие тест, я решил опустить




Наконец, сам скрипт джобы:

```
node {
  // Download git repository
  git url: 'https://github.com/jenkinsci/robot-plugin'

  // Change PATH, MAVEN, JAVA_HOME env variables only in this scope
  withEnv(["PATH+MAVEN=${tool 'M3'}/bin", "PATH+JAVA_HOME=${tool jdk_version}/bin"]) {
    // Execute verify mvn goal - build and test the project
    sh "mvn -B -Dmaven.test.failure.ignore verify"
    // Archive artifacts
    step([$class: 'ArtifactArchiver', artifacts: '**/target/*.jar', fingerprint: true])
    // Archive test results
    step([$class: 'JUnitResultArchiver', testResults: '**/target/surefire-reports/TEST-*.xml'])
  }
}
```

Что есть примечательного в скрипте?

- node {} - как-бы пространсто исполнения. Для каждой ноды создётся по workspace директории.
- withEnv(["PATH+MAVEN=${tool 'M3'}/bin", "PATH+JAVA=${tool jdk_version}/bin"]) {} - устанавливает переменные окружения PATH, MAVEN и JAVA_HOME в соответствующие им значения в определённой видимости - не глобально(как было бы без withEnv), а лишь внутри фигурных скобок. Это нужно, чтобы параллельно исполняющиеся скрипты не портили друг другу переменные окружения.
  **Доп.Пункт 3.d выполнен.**
- withEnv([“PATH+JAVA=${tool jdk_version}/bin”]) - путь к jdk изымается из переменной-параметра jdk_version, которую выбирает пользователь.
  **Пункт 3.a выполнен.**
- mvn verify - исполнить все цели до verify(интеграционные тесты), что включает в себя и прогон юнит-тестов.
  **Доп.Пункт 3.c выполнен.**
- Ключи mvn:
  - -B указывает на неинтерактивный мод(Что важно, т.к. Jenkins - утилита автоматизации)
  - -Dmaven.test.failure.ignore - указывает мэйвену, что при наличии ошибкок при тестах, необходимо исполнять остальные цели, и package в том числе.
- step([$class: 'ArtifactArchiver', artifacts: '**/target/*.jar', fingerprint: true]) - Архивировать все .jar фаилы из target, чтобы пользователь мог их скачать.
  **Пункт 3.b выполнен**
- step([$class: 'JUnitResultArchiver', testResults: '**/target/surefire-reports/TEST-*.xml']) - Логировать результаты тестов, чтобы пользователь мог их посмотреть.

6. **Тестирование смены версии jdk**

Заменяю разработанный скрипт на следующий скрипт-тест для тестирования:

```
node {
  withEnv(["PATH+MAVEN=${tool 'M3'}/bin", "PATH+JAVA_HOME=${tool jdk_version}/bin"]) {
    sh 'java -version'
  }
  sh 'java -version'
}
```

**Результат:**

GUI


Jdk_7u80:

```
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
[Test and Package RobotFramework@2] Running shell script
+ java -version
java version "1.7.0_80"
Java(TM) SE Runtime Environment (build 1.7.0_80-b15)
Java HotSpot(TM) 64-Bit Server VM (build 24.80-b11, mixed mode)
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] sh
[Test and Package RobotFramework@2] Running shell script
+ java -version
java version "1.8.0_72"
Java(TM) SE Runtime Environment (build 1.8.0_72-b15)
Java HotSpot(TM) 64-Bit Server VM (build 25.72-b15, mixed mode)
```

Jdk_8u92:

```
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
[Test and Package RobotFramework] Running shell script
+ java -version
java version "1.8.0_92"
Java(TM) SE Runtime Environment (build 1.8.0_92-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.92-b14, mixed mode)
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] sh
[Test and Package RobotFramework] Running shell script
+ java -version
java version "1.8.0_72"
Java(TM) SE Runtime Environment (build 1.8.0_72-b15)
Java HotSpot(TM) 64-Bit Server VM (build 25.72-b15, mixed mode)
```

Jdk_9-ea:

```
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
[Test and Package RobotFramework@2] Running shell script
+ java -version
java version "9-ea"
Java(TM) SE Runtime Environment (build 9-ea+121)
Java HotSpot(TM) 64-Bit Server VM (build 9-ea+121, mixed mode)
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] sh
[Test and Package RobotFramework@2] Running shell script
+ java -version
java version "1.8.0_72"
Java(TM) SE Runtime Environment (build 1.8.0_72-b15)
Java HotSpot(TM) 64-Bit Server VM (build 25.72-b15, mixed mode)
```

Как видно, версии jdk правильные, причём за пределами withenv() {}, установлен мой домашний jdk.

## 7. Проверка итогового функционала

Так, в конце концов, наша Джоба должна выполнять следующее:

- Скачивать исходники с Гитхаба
- Прогонять юнит-тесты и логировать результат
- Собирать проект, предоставлять пользователю возможность скачать архив с артефактами

**Итоговый лог Джобы:**

Скачивание репозитория:

```
[Pipeline] git
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/jenkinsci/robot-plugin # timeout=10
Fetching upstream changes from https://github.com/jenkinsci/robot-plugin
 > git --version # timeout=10
 > git -c core.askpass=true fetch --tags --progress https://github.com/jenkinsci/robot-plugin +refs/heads/*:refs/remotes/origin/*
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision c0385d5e05a3adfc1ffcb05b6c70c9e441b61979 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f c0385d5e05a3adfc1ffcb05b6c70c9e441b61979 # timeout=10
 > git branch -a -v --no-abbrev # timeout=10
 > git branch -D master # timeout=10
 > git checkout -b master c0385d5e05a3adfc1ffcb05b6c70c9e441b61979
 > git rev-list c0385d5e05a3adfc1ffcb05b6c70c9e441b61979 # timeout=10
```

Прогон тестов:

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running hudson.plugins.robot.RobotPublisherSystemTest
Running hudson.plugins.robot.RobotProjectActionTest
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 11.692 sec - in hudson.plugins.robot.RobotProjectActionTest
Running hudson.plugins.robot.tokens.RobotFailTokenMacroTest
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.131 sec - in hudson.plugins.robot.tokens.RobotFailTokenMacroTest
Running hudson.plugins.robot.tokens.RobotPassPercentageTokenMacroTest
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.021 sec - in hudson.plugins.robot.tokens.RobotPassPercentageTokenMacroTest
Running hudson.plugins.robot.tokens.RobotPassTokenMacroTest
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.041 sec - in hudson.plugins.robot.tokens.RobotPassTokenMacroTest
Running hudson.plugins.robot.tokens.RobotFailedCasesTokenMacroTest
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.177 sec - in hudson.plugins.robot.tokens.RobotFailedCasesTokenMacroTest
Running hudson.plugins.robot.tokens.RobotPassRatioTokenMacroTest
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.054 sec - in hudson.plugins.robot.tokens.RobotPassRatioTokenMacroTest
Running hudson.plugins.robot.tokens.RobotReportLinkTokenMacroTest
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.011 sec - in hudson.plugins.robot.tokens.RobotReportLinkTokenMacroTest
Running hudson.plugins.robot.tokens.RobotTotalTokenMacroTest
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.01 sec - in hudson.plugins.robot.tokens.RobotTotalTokenMacroTest
Running hudson.plugins.robot.model.RobotSuiteResultTest
Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.06 sec - in hudson.plugins.robot.model.RobotSuiteResultTest
Running hudson.plugins.robot.model.RobotResultTest
Tests run: 30, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.363 sec - in hudson.plugins.robot.model.RobotResultTest
Running hudson.plugins.robot.graph.RobotGraphHelperTest
Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.566 sec - in hudson.plugins.robot.graph.RobotGraphHelperTest
Running hudson.plugins.robot.RobotPublisherTest
Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.635 sec - in hudson.plugins.robot.RobotPublisherTest
Running InjectedTest
Tests run: 32, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 159.012 sec - in InjectedTest
Tests run: 15, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 300.02 sec - in hudson.plugins.robot.RobotPublisherSystemTest

Results :

Tests run: 121, Failures: 0, Errors: 0, Skipped: 0
```

Сборка проекта в jar:

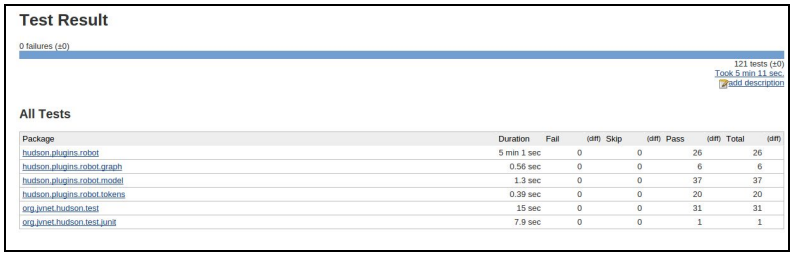
```
[INFO]
[INFO] --- maven-license-plugin:1.7:process (default) @ robot ---
[INFO] Generated /var/lib/jenkins/workspace/Test and Package RobotFramework/target/robot/WEB-INF/licenses.xml
[INFO]
[INFO] --- maven-hpi-plugin:1.106:hpi (default-hpi) @ robot ---
[INFO] Generating /var/lib/jenkins/workspace/Test and Package RobotFramework/target/robot/META-INF/MANIFEST.MF
[INFO] Building jar: /var/lib/jenkins/workspace/Test and Package RobotFramework/target/robot.jar
[INFO] Exploding webapp...
[INFO] Copy webapp webResources to /var/lib/jenkins/workspace/Test and Package RobotFramework/target/robot
[INFO] Assembling webapp robot in /var/lib/jenkins/workspace/Test and Package RobotFramework/target/robot
[INFO] Generating hpi /var/lib/jenkins/workspace/Test and Package RobotFramework/target/robot.hpi
[INFO] Building jar: /var/lib/jenkins/workspace/Test and Package RobotFramework/target/robot.hpi
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 5:51.605s
[INFO] Finished at: Sat Jun 04 18:17:45 MSK 2016
[INFO] Final Memory: 31M/241M
[INFO] ------------------------------------------------------------------------
```

Конец лога - успешное заверение Джобы:

```
[Pipeline] step
Archiving artifacts
Recording fingerprints
[Pipeline] step
Recording test results
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Возможность скачивания артефакта в Job Status -> Last successful artifact:


Возможность просмотров результата тестов в Job Status -> Last test result:




Так, все пункты выполнил. Надеюсь, было достаточно подробно :) Всё остальное смогу объяснить при встрече + выучу теорию лучше.
**Спасибо за внимание!**