Squash TF Execution Server Documentation

squahstest

Apr 02, 2020

Contents

CHAPTER 1

Administration

1.1 Execution Server - Administration

1.1.1 Execution Server installation

Warning: In **2.3.0-RELEASE version of the server**, we updated Jenkins to 2.190.1 version. Since 2.165 and newer version, Jenkins no longer supports the old (-remoting) mode in either the client or server. If you have installed a 2.3.0-RELEASE of the server and want to install an agent, you need to download **2.3.0-RELEASE (or newer) version of the agent**.

Headless install

Linux procedure

Pre-requisites

- A version of the java 8 jdk must be installed on the system.
- The bin directory from this JDK must be included in the PATH shell variable. If necessary, run the following command in the shell you'll be using :

```
export PATH=</path/to/the/JDK>/bin:$PATH
```

Procedure

To perform a headless server installation under linux, you need to perform the following steps :

- Create the installation parameter file.
- Launch the installer from the command line with the installation paramameter file path as argument.

> Installation parameter file

The parameter file can be generated from the attached template. Please make sure to adjust the following parameters to the real values on your system :

- **jdkPath** : Path to a java 8 JDK that will be used to run the server.
- **installPath** : Target installation path for the **Squash TF** server

`linux-headless-install-parms-template.xml`

> Installer execution

To actually install the **Squash TF** server, please run the following command :

```
java -jar <path/to/the/>squash-tf-execution-server-bundle-<version>-linux-installer.
↪jar <path/to/the/xml-installation-parameter-file>
```

Note:

You can now access to your execution server at : <host_url>:8080/jenkins
The default login is : admin / admin

Windows procedure

Pre-requisites

- A version of the java 8 jdk must be installed on the system.
- The bin directory from this JDK must be included in the PATH shell variable. If necessary, run the following command in the shell you'll be using :

```
set PATH=<\path\to\the\JDK>\bin;%PATH%
```

Procedure

To perform a headless server installation under linux, you need to perform the following steps :

- Create the installation parameter file.
- Launch the installer from the command line with the installation paramameter file path as argument.

> Installation parameter file

The parameter file Can be generated from the attached template. Please make sure to adjust the following parameters to the real values on your system :

- **jdkPath** : Path to a java 8 JDK that will be used to run the server.
- **installPath** : Target installation path for the **Squash TF** server
- **programGroup** : Title for the Squash TF server submenu.

`squash-tf-execution-server-auto-install-windows.xml`

> Installer execution

To actually install the **Squash TF** server, please run the following command :

```
java -jar <path\to\the\>squash-tf-execution-server-bundle-<version>-win64-installer.
↪jar <path\to\the\xml-installation-parameter-file>
```

Note:

You can now access to your execution server at : <host_url>:8080/jenkins
The default login is : admin / admin

GUI install

Pre-requisites

- A version of the java 8 jdk must be installed on the system.

- The bin directory from this JDK must be included in the PATH shell variable. If necessary run the following command in the shell you'll be using :
  - On Linux :

    ```
    export PATH=</path/to/the/JDK>/bin:$PATH
    ```

  - On Windows :

    ```
    set PATH=<\path\to\the\JDK>\bin;%PATH%
    ```

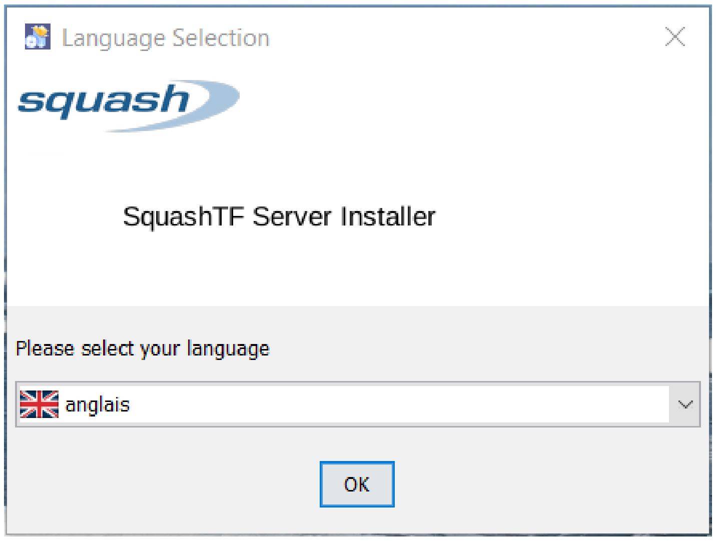
- Download the desired installer version of **Squash TF Execution Server** here

Procedure

1. Launch the graphical installer by a double click on the **Squash TF Execution Server** jar file, or launch it with the following command line :

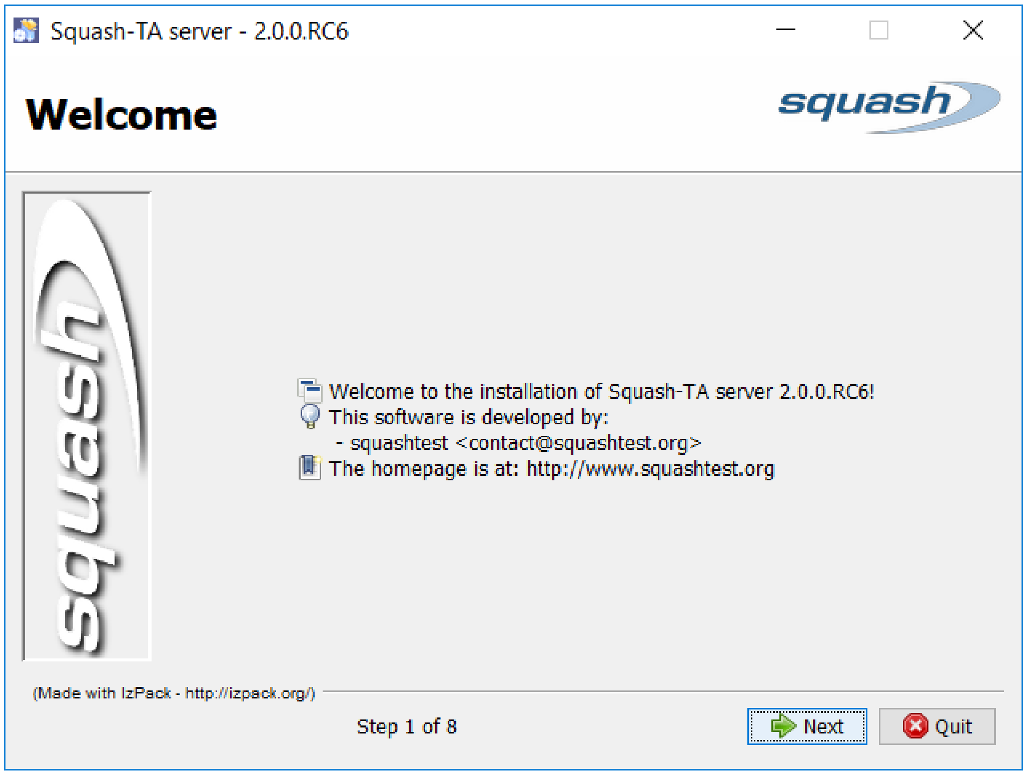
   ```
   java -jar <desired-squash-tf-version>.jar
   ```

   Then you should have the language selection screen :

   - Choose the desired language for the installer then click on `OK`

2. The *Welcome* screen appears :

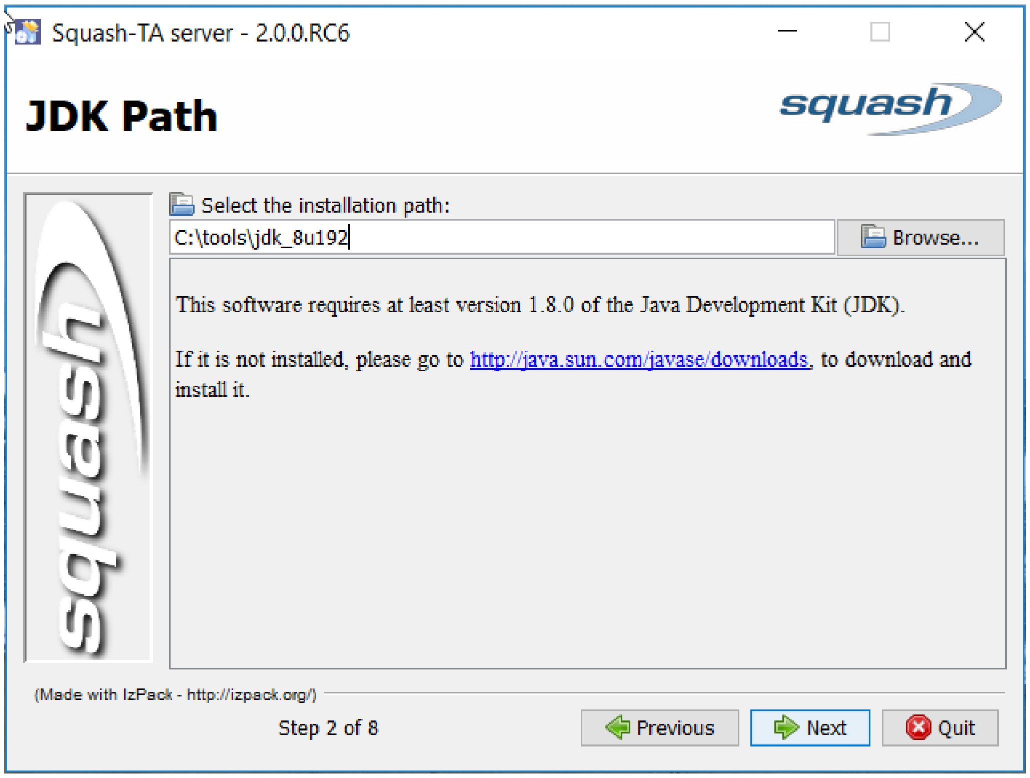

Squash-TA server - 2.0.0.RC6
Welcome
squash
squash
(Made with IzPack - http://izpack.org/)
Next
Quit

Welcome to the installation of Squash-TA server 2.0.0.RC6!
This software is developed by:
- squashtest <contact@squashtest.org>
The homepage is at: http://www.squashtest.org

Step 1 of 8

- Click on `Next`

3. The *JDK Path* screen appears. This screen allows you to define the path where you java jdk is installed.


Squash-TA server - 2.0.0.RC6

JDK Path

Select the installation path:

C:\tools\jdk_8u192

This software requires at least version 1.8.0 of the Java Development Kit (JDK).

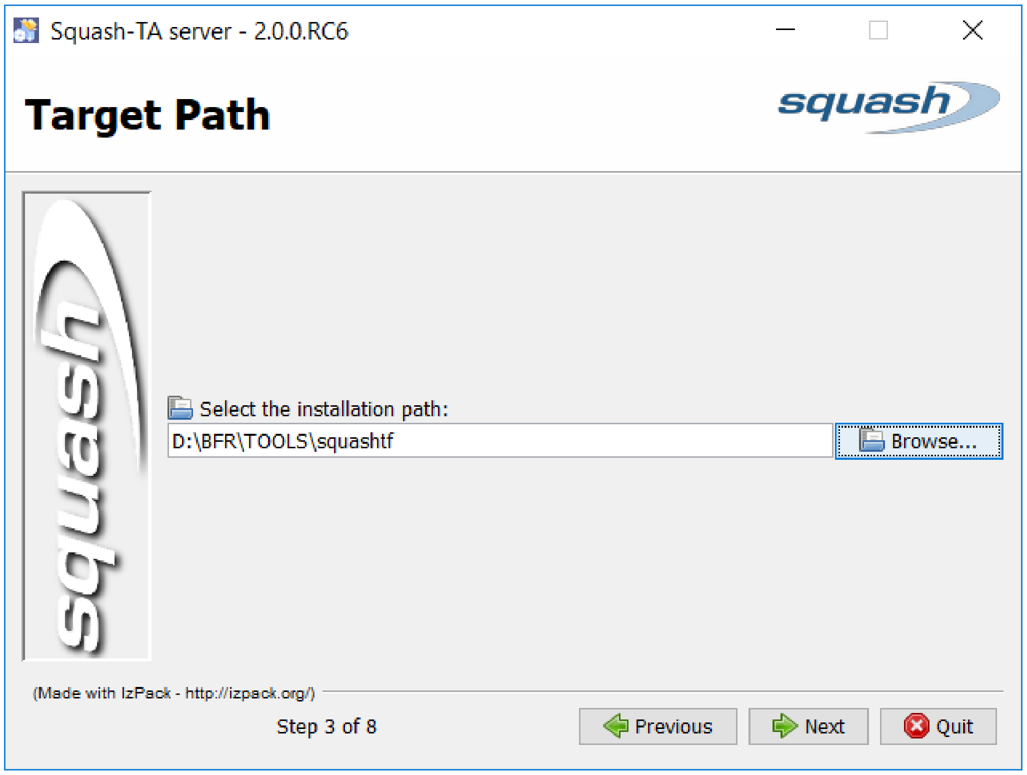
If it is not installed, please go to http://java.sun.com/javase/downloads, to download and install it.

(Made with IzPack - http://izpack.org/)

Step 2 of 8

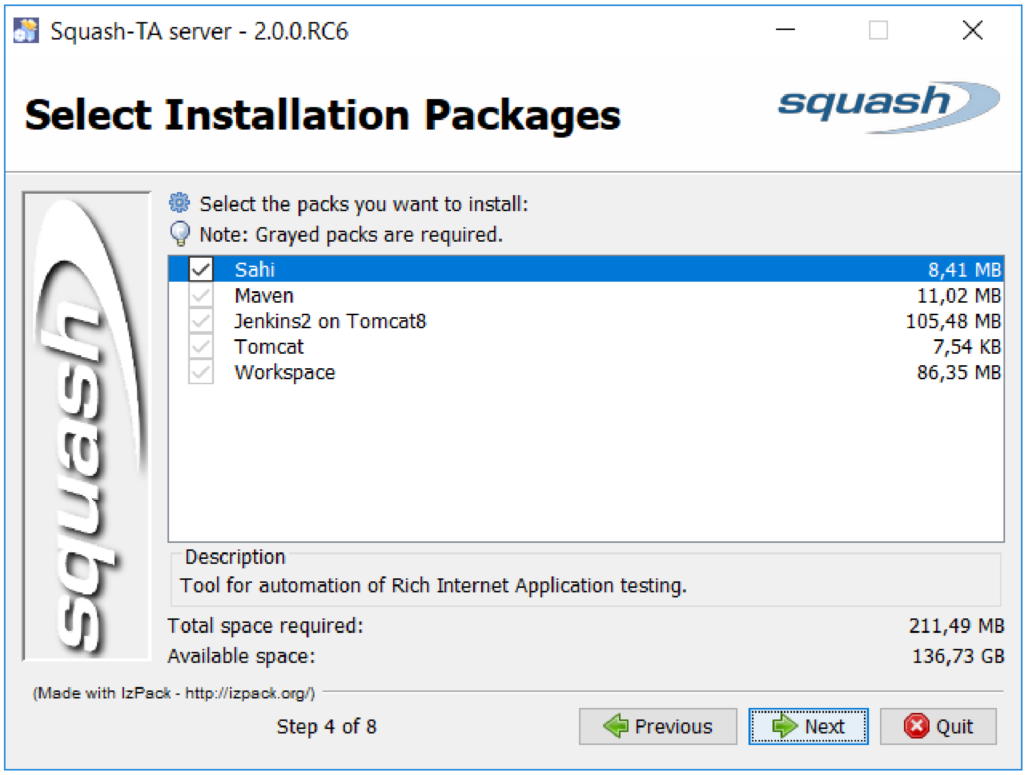
- Choose the path where your jdk is installed.
- Then click on `Next`

4. The *Target Path* Screen appears. It allows you to define the installation directory.


Squash-TA server - 2.0.0.RC6
Target Path
squash
Select the installation path:
D:\BFR\TOOLS\squashtf
Browse...
(Made with IzPack - http://izpack.org/)
Step 3 of 8
Previous
Next
Quit

- Define the installation directory (or keep the one proposed by default)
- Then click on `Next`

5. The *Select Installation Packages* appears. It allows you to choose which package you want to install. Currently, you only have the choice to install or not the sahi proxy.


Squash-TA server - 2.0.0.RC6

Select Installation Packages

Select the packs you want to install:
Note: Grayed packs are required.

✓	Sahi	8,41 MB
✓	Maven	11,02 MB
✓	Jenkins2 on Tomcat8	105,48 MB
✓	Tomcat	7,54 KB
✓	Workspace	86,35 MB

Description
Tool for automation of Rich Internet Application testing.

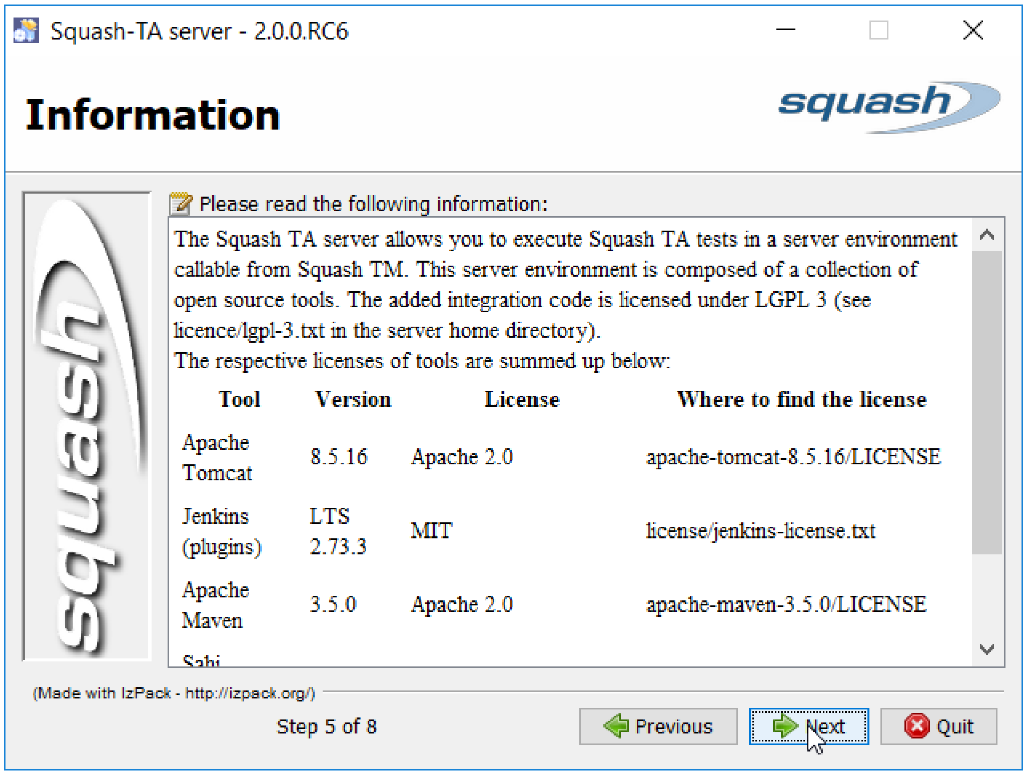
Total space required: 211,49 MB
Available space: 136,73 GB

(Made with IzPack - http://izpack.org/)

Step 4 of 8

- Do your choice
- Then click on `Next`

6. The *licenses information* screen appears. It gives you all the informations concerncing the license of the product embedded

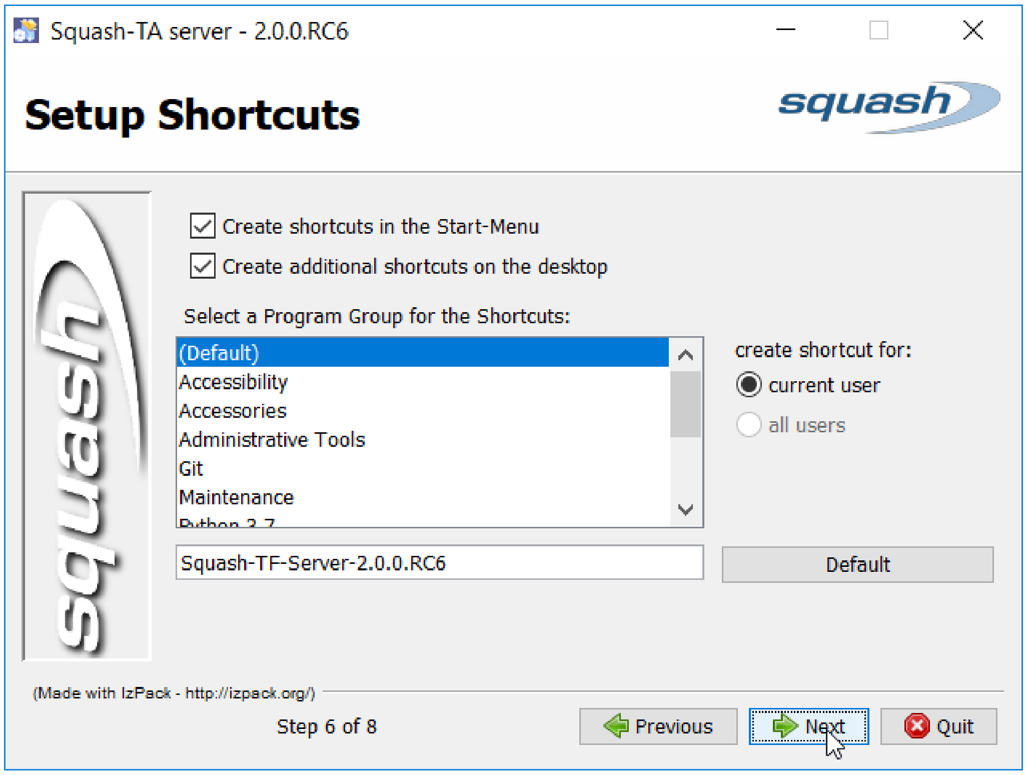

Squash-TA server - 2.0.0.RC6
Information
squash
Please read the following information:
(Made with IzPack - http://izpack.org/)
Step 5 of 8
Previous
Next
Quit

The Squash TA server allows you to execute Squash TA tests in a server environment callable from Squash TM. This server environment is composed of a collection of open source tools. The added integration code is licensed under LGPL 3 (see licence/lgpl-3.txt in the server home directory).
The respective licenses of tools are summed up below:

Tool	Version	License	Where to find the license
Apache Tomcat	8.5.16	Apache 2.0	apache-tomcat-8.5.16/LICENSE
Jenkins (plugins)	LTS 2.73.3	MIT	license/jenkins-license.txt
Apache Maven	3.5.0	Apache 2.0	apache-maven-3.5.0/LICENSE
Sahi			

- Click on `Next`

7. The *Setup Shortcuts* screen appears. It allows you to define your preferences concerning the shortcuts to create.



Squash-TA server - 2.0.0.RC6

Setup Shortcuts

☑ Create shortcuts in the Start-Menu

☑ Create additional shortcuts on the desktop

Select a Program Group for the Shortcuts:

(Default)
Accessibility
Accessories
Administrative Tools
Git
Maintenance
Python 3.7

create shortcut for:
● current user
○ all users

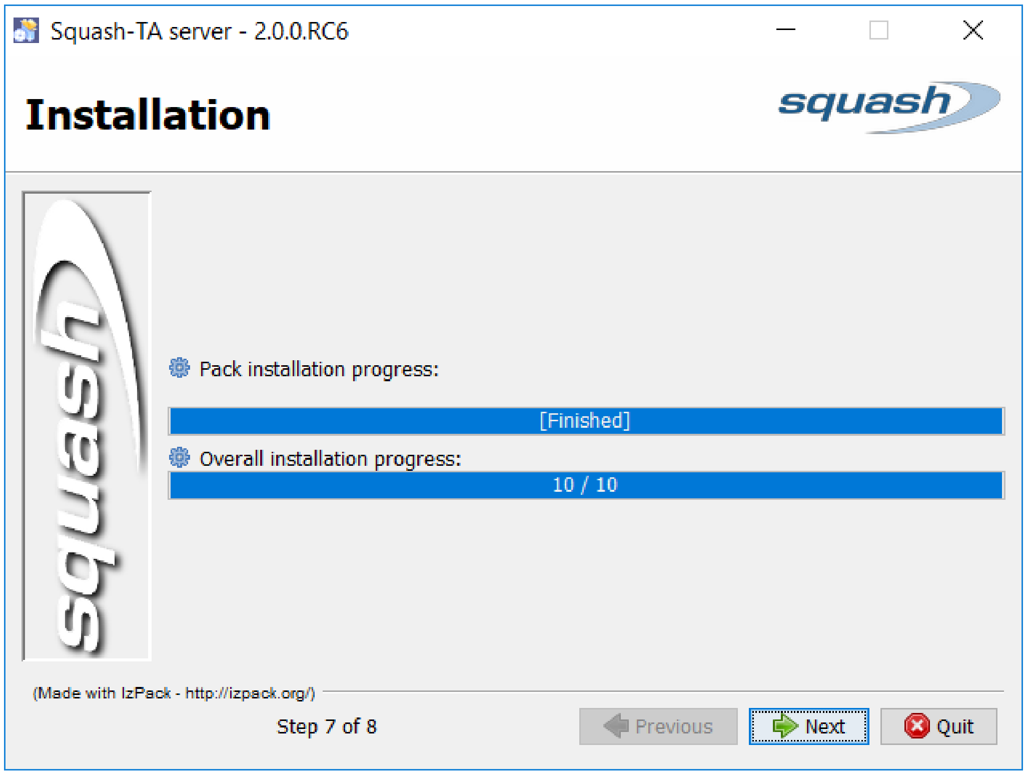
Squash-TF-Server-2.0.0.RC6

(Made with IzPack - http://izpack.org/)

Step 6 of 8

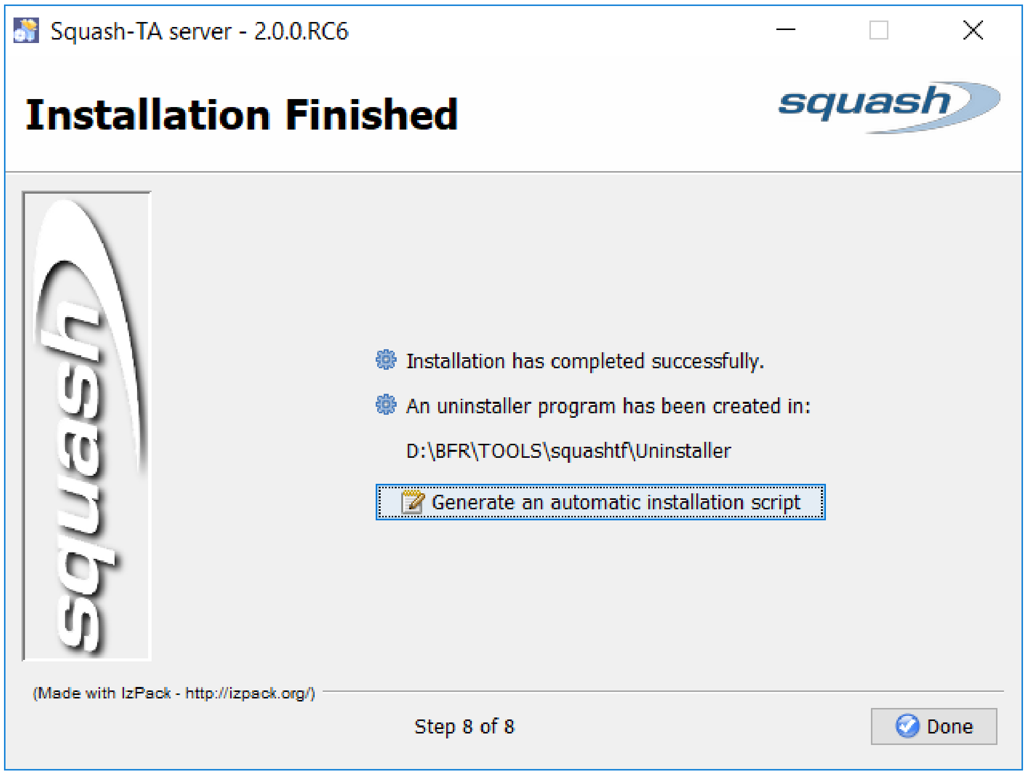
- Click on `Next`

8. The *installation tracking* screen appears. This screen show you the progression of the installation.


Squash-TA server - 2.0.0.RC6
Installation
squash
Pack installation progress:
[Finished]
Overall installation progress:
10 / 10
(Made with IzPack - http://izpack.org/)
Step 7 of 8
Previous
Next
Quit

- When the installation is finished, click on `Next`

9. The *Installation Finished* screen appears :


Squash-TA server - 2.0.0.RC6

Installation Finished

Installation has completed successfully.

An uninstaller program has been created in:

D:\BFR\TOOLS\squashtf\Uninstaller

Generate an automatic installation script
(Made with IzPack - http://izpack.org/)
Step 8 of 8
Done

On this screen there is a button `Generate an automatic installation script`. It offers you to save an xml file which you could use to reproduce the same installation (with the same configuration you chose in the previous screens). See the headless installation procedure for explanations on how do a new installation using this xml file.

- Then click on `Done`

Note:

You can now access to your execution server at : <host_url>:8080/jenkins
The default login is : admin / admin

Congratulations ! You have installed the Squash TF Execution Server

Docker

With version 2.0.0 and above, **Squash TF Execution Server** also comes with docker images.
Since version 2.2.0 those images are also available on dockerhub :
https://hub.docker.com/r/squashtest/squash-tf-execution-server
However, our images are still available as tarball in our repo (download at squashtest.com).

These images are fully parameterized execution servers ready to be deployed.

Below is the procedure to do so.

1. Retrieve the Docker image

   >> Retrieve from dockerhub

   ```
   docker pull squashtest/squash-tf-execution-server:{version}
   ```

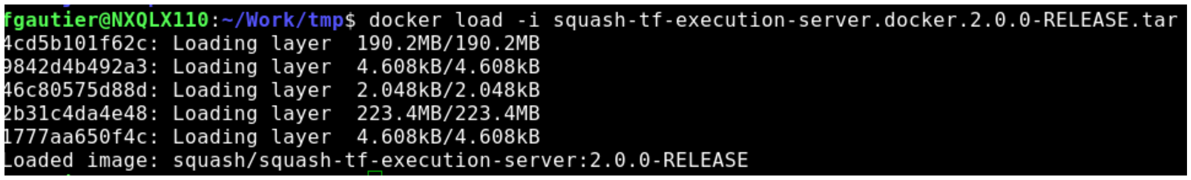
   where {version} is the downloaded server version.

   >> Retrieve from our artifacts repository

   - Download the Docker image of the execution server here
   - Load the image on your Docker setup with the command :

     ```
     docker load -i squash-tf-execution-server.docker.{version}.tar
     ```

   where {version} is the downloaded server version.

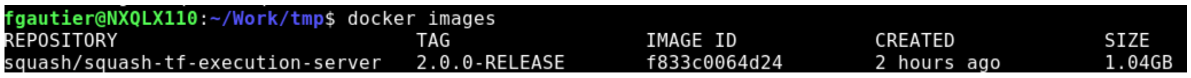
   ```
   fgautier@NXQLX110:~/Work/tmp$ docker load -i squash-tf-execution-server.docker.2.0.0-RELEASE.tar
   4cd5b101f62c: Loading layer  190.2MB/190.2MB
   9842d4b492a3: Loading layer  4.608kB/4.608kB
   46c80575d88d: Loading layer  2.048kB/2.048kB
   2b31c4da4e48: Loading layer  223.4MB/223.4MB
   1777aa650f4c: Loading layer  4.608kB/4.608kB
   Loaded image: squash/squash-tf-execution-server:2.0.0-RELEASE
   ```

   Warning: Do not execute the "docker import" command on the image archive as it will flatten it and result in the loss of all context data such as entry points.

2. Use the following command to check the correctness of the previous load :

   ```
   docker images
   ```

   On successful load the output looks like :

   ```
   fgautier@NXQLX110:~/Work/tmp$ docker images
   REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
   squash/squash-tf-execution-server   2.0.0-RELEASE       f833c0064d24        2 hours ago         1.04GB
   ```

3. Create a running container of the image using the "docker run" command

   >> Image from dockerhub

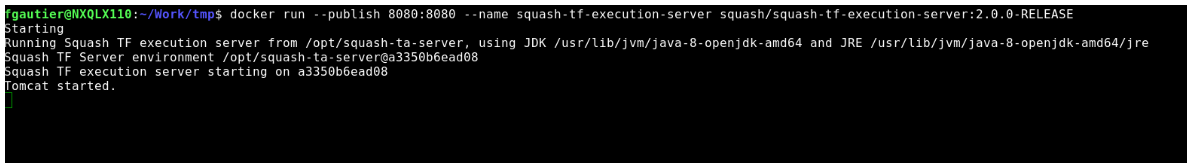
   ```
   docker run --publish 1234:8080 --name squash-tf-execution-server squashtest/
   ↪squash-tf-execution-server:{version}
   ```

   >> Image from our artifacts repository

```
docker run --publish 1234:8080 --name squash-tf-execution-server squash/squash-tf-
↪execution-server:{version}
```

Warning: The difference between the 2 run command is in the image name. The dockerhub one is **squashtest**/squash-tf-execution-server:{version}, the other is **squash**/squash-tf-execution-server:{version}

- The `--publish` option binds a port of the physical machine to the one of the soon to be created container. Here it binds the 1234 port of the physical machine to the 8080 port of the container. This is **mandatory** since the Jenkins installed on the image will listen to the 8080 of the container. If a Master/Slave architecture is planned you can open as many tcp port of the container as you wish by repeating the –publish option.
- The `--name` option is optional, but quite handy since it allows one to give a non arbitratry name to the container about to be be created.



```
fgautier@NXQLX110:~/Work/tmp$ docker run --publish 8080:8080 --name squash-tf-execution-server squash/squash-tf-execution-server:2.0.0-RELEASE
Starting
Running Squash TF execution server from /opt/squash-ta-server, using JDK /usr/lib/jvm/java-8-openjdk-amd64 and JRE /usr/lib/jvm/java-8-openjdk-amd64/jre
Squash TF Server environment /opt/squash-ta-server@a3350b6ead08
Squash TF execution server starting on a3350b6ead08
Tomcat started.
```

Note:

You can now access to your execution server at : <host_url>:<host_bound_port>/jenkins
The default login is : admin / admin

Congratulations ! You now have a Squash TF Execution Server running inside a Docker container and ready to execute its first tests.

Once the container is properly created and running, use the docker stop command to shut it down :

```
docker stop squash-tf-execution-server
```

Use the docker start command to restart your server properly and retain your previous configuration and non-volatile data :

```
docker start squash-tf-execution-server
```

1.1.2 Execution Server - Update

Execution Server - Generic Update

Overview

This page will help you upgrade your **Squash TF Execution Server** and your job collection to the latest version available.

Its instructions are generic and need to be followed for each update; however the process can have additionnal steps for some versions.

These specific instructions are documented in their own sections of this guide.

Note: The following procedure is identical whether your Execution Server runs on a Windows or a Linux computer.

Warning: We recommend updating step-by-step from your current version of **Squash TF execution server** to the next until you reach the latest.

If you wish to jump ahead to the latest release please read carefully the instructions dedicated to each version of the Execution Server you pass by.

The guide is divided in 3 major parts :

- First the process to update a physical install of **Squash TF Execution Server** on a local system.
- Second the process to update a Docker install.
- Third the various configurations to update in Jenkins. This part is necessary for both install.

Physical Server update

Setup

1. Download the version of **Squash TF** you wish to install here.

---

2. Stop your current server.

---

3. Install the new TF Execution Server in a separate location as a classic new installation.

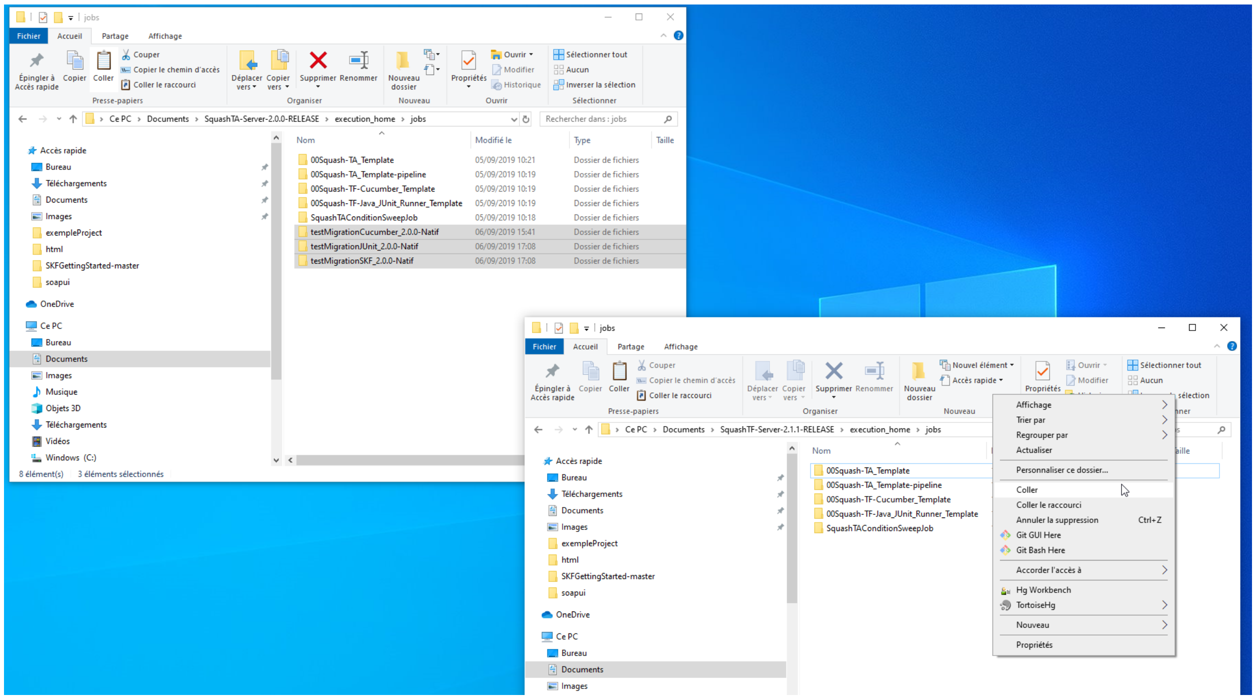
Server & jobs migration

4. Check the page dedicated to updating to the chosen version of the server and follow the specific instructions in the **Before migrating jobs** section.

---

5. Copy your custom jobs from **{current_server_path}/execution_home/jobs/** to **{new_server_path}/execution_home/jobs/**

> **Warning:** Do not copy Squash TF templates (beginning by **00**) and **SquashTAConditionSweepJobs**, so avoid copying the whole jobs directory.



> Ce PC › Documents › SquashTA-Server-2.0.0-RELEASE › execution_home › jobs

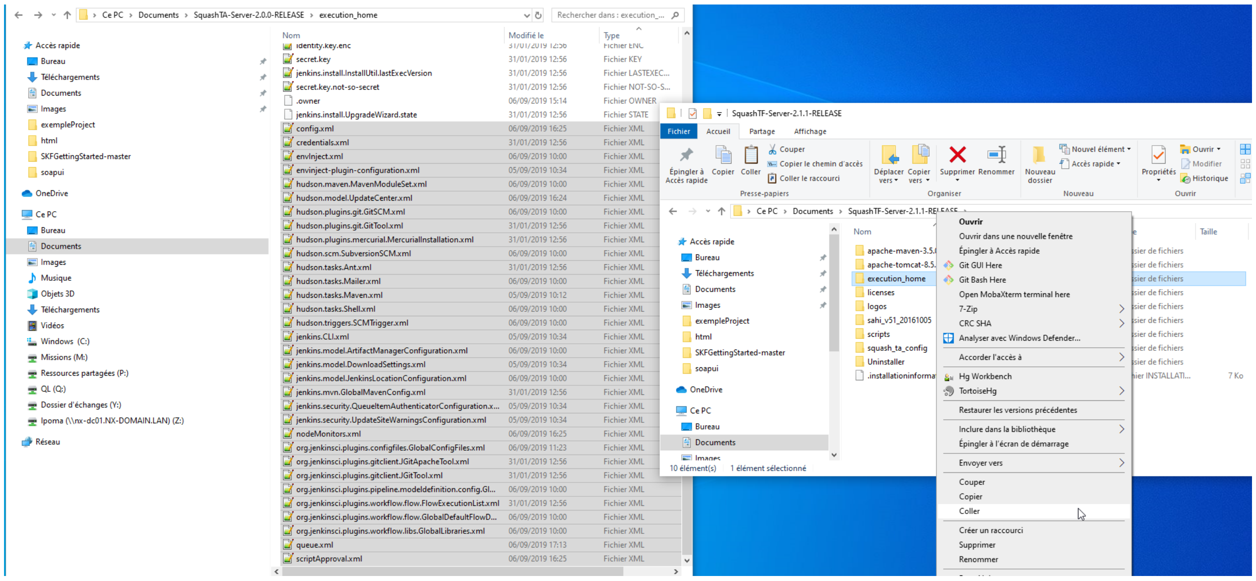
Nom	Modifié le	Type	Taille
00Squash-TA_Template	05/09/2019 10:21	Dossier de fichiers	
00Squash-TA_Template-pipeline	05/09/2019 10:19	Dossier de fichiers	
00Squash-TF-Cucumber_Template	05/09/2019 10:19	Dossier de fichiers	
00Squash-TF-Java_JUnit_Runner_Template	05/09/2019 10:19	Dossier de fichiers	
SquashTAConditionSweepJob	05/09/2019 10:18	Dossier de fichiers	
testMigrationCucumber_2.0.0-Natif	06/09/2019 15:41	Dossier de fichiers	
testMigrationJUnit_2.0.0-Natif	06/09/2019 17:08	Dossier de fichiers	
testMigrationSKF_2.0.0-Natif	06/09/2019 17:08	Dossier de fichiers	

> Ce PC › Documents › SquashTF-Server-2.1.1-RELEASE › execution_home › jobs

Nom
00Squash-TA_Template
00Squash-TA_Template-pipeline
00Squash-TF-Cucumber_Template
00Squash-TF-Java_JUnit_Runner_Template
SquashTAConditionSweepJob

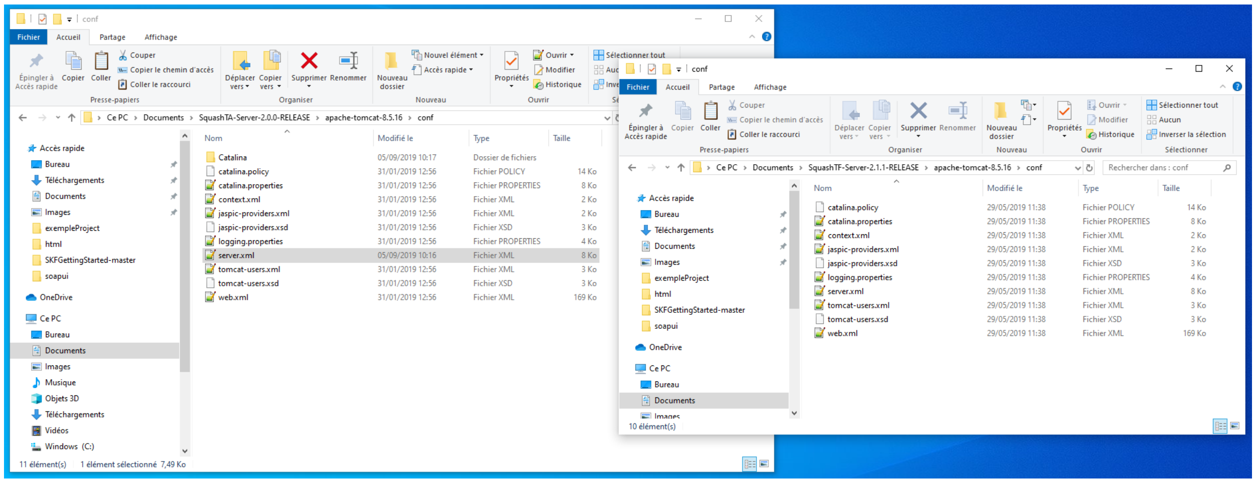
---

6. Copy and replace all .xml files from **{current_server_path}/execution_home/** to **{new_server_path}/execution_home/**


Ce PC > Documents > SquashTA-Server-2.0.0-RELEASE > execution_home
Rechercher dans : execution_...
SquashTF-Server-2.1.1-RELEASE
Ce PC > Documents > SquashTF-Server-2.1.1-RELEASE
apache-maven-3.5.
apache-tomcat-8.5.
execution_home
licenses
logs
sahi_v51_20161005
scripts
squash_ta_config
Uninstaller
.installationinforma
Ouvrir
Ouvrir dans une nouvelle fenêtre
Épingler à Accès rapide
Git GUI Here
Git Bash Here
Open MobaXterm terminal here
7-Zip
CRC SHA
Analyser avec Windows Defender...
Accorder l'accès à
Hg Workbench
TortoiseHg
Restaurer les versions précédentes
Inclure dans la bibliothèque
Épingler à l'écran de démarrage
Envoyer vers
Couper
Copier
Coller
Créer un raccourci
Supprimer
Renommer
10 élément(s)
1 élément sélectionné

Nom	Modifié le	Type
identity.key.enc	31/01/2019 12:56	Fichier ENC
secret.key	31/01/2019 12:56	Fichier KEY
jenkins.install.InstallUtil.lastExecVersion	31/01/2019 12:56	Fichier LASTEXEC...
secret.key.not-so-secret	31/01/2019 12:56	Fichier NOT-SO-S...
.owner	06/09/2019 15:14	Fichier OWNER
jenkins.install.UpgradeWizard.state	31/01/2019 12:56	Fichier STATE
config.xml	06/09/2019 16:25	Fichier XML
credentials.xml	31/01/2019 12:56	Fichier XML
envInject.xml	06/09/2019 10:00	Fichier XML
envinject-plugin-configuration.xml	05/09/2019 10:34	Fichier XML
hudson.maven.MavenModuleSet.xml	06/09/2019 10:00	Fichier XML
hudson.model.UpdateCenter.xml	06/09/2019 16:24	Fichier XML
hudson.plugins.git.GitSCM.xml	06/09/2019 10:00	Fichier XML
hudson.plugins.git.GitTool.xml	31/01/2019 12:56	Fichier XML
hudson.plugins.mercurial.MercurialInstallation.xml	31/01/2019 12:56	Fichier XML
hudson.scm.SubversionSCM.xml	06/09/2019 10:00	Fichier XML
hudson.tasks.Ant.xml	31/01/2019 12:56	Fichier XML
hudson.tasks.Mailer.xml	06/09/2019 10:00	Fichier XML
hudson.tasks.Maven.xml	05/09/2019 10:12	Fichier XML
hudson.tasks.Shell.xml	06/09/2019 10:00	Fichier XML
hudson.triggers.SCMTrigger.xml	06/09/2019 10:00	Fichier XML
jenkins.CLI.xml	05/09/2019 10:34	Fichier XML
jenkins.model.ArtifactManagerConfiguration.xml	06/09/2019 10:00	Fichier XML
jenkins.model.DownloadSettings.xml	05/09/2019 10:34	Fichier XML
jenkins.model.JenkinsLocationConfiguration.xml	06/09/2019 10:00	Fichier XML
jenkins.mvn.GlobalMavenConfig.xml	31/01/2019 12:56	Fichier XML
jenkins.security.QueueItemAuthenticatorConfiguration.x...	05/09/2019 10:34	Fichier XML
jenkins.security.UpdateSiteWarningsConfiguration.xml	05/09/2019 10:34	Fichier XML
nodeMonitors.xml	06/09/2019 16:25	Fichier XML
org.jenkinsci.plugins.configfiles.GlobalConfigFiles.xml	06/09/2019 11:23	Fichier XML
org.jenkinsci.plugins.gitclient.JGitApacheTool.xml	31/01/2019 12:56	Fichier XML
org.jenkinsci.plugins.gitclient.JGitTool.xml	31/01/2019 12:56	Fichier XML
org.jenkinsci.plugins.pipeline.modeldefinition.config.Gl...	06/09/2019 10:00	Fichier XML
org.jenkinsci.plugins.workflow.flow.FlowExecutionList.xml	31/01/2019 12:56	Fichier XML
org.jenkinsci.plugins.workflow.flow.GlobalDefaultFlowD...	06/09/2019 10:00	Fichier XML
org.jenkinsci.plugins.workflow.libs.GlobalLibraries.xml	06/09/2019 10:00	Fichier XML
queue.xml	06/09/2019 17:13	Fichier XML
scriptApproval.xml	06/09/2019 16:25	Fichier XML

7. Copy and replace the file **server.xml** located in **{current_server_path}/apache-tomcat-8.5.16/conf/** to **{new_server_path}/apache-tomcat-8.5.16/conf/**


Ce PC > Documents > SquashTA-Server-2.0.0-RELEASE > apache-tomcat-8.5.16 > conf
11 élément(s)
1 élément sélectionné 7,49 Ko
Ce PC > Documents > SquashTF-Server-2.1.1-RELEASE > apache-tomcat-8.5.16 > conf
Rechercher dans : conf
10 élément(s)

Nom	Modifié le	Type	Taille
Catalina	05/09/2019 10:17	Dossier de fichiers	
catalina.policy	31/01/2019 12:56	Fichier POLICY	14 Ko
catalina.properties	31/01/2019 12:56	Fichier PROPERTIES	8 Ko
context.xml	31/01/2019 12:56	Fichier XML	2 Ko
jaspic-providers.xml	31/01/2019 12:56	Fichier XML	2 Ko
jaspic-providers.xsd	31/01/2019 12:56	Fichier XSD	3 Ko
logging.properties	31/01/2019 12:56	Fichier PROPERTIES	4 Ko
server.xml	05/09/2019 10:16	Fichier XML	8 Ko
tomcat-users.xml	31/01/2019 12:56	Fichier XML	3 Ko
tomcat-users.xsd	31/01/2019 12:56	Fichier XSD	3 Ko
web.xml	31/01/2019 12:56	Fichier XML	169 Ko

Nom	Modifié le	Type	Taille
catalina.policy	29/05/2019 11:38	Fichier POLICY	14 Ko
catalina.properties	29/05/2019 11:38	Fichier PROPERTIES	8 Ko
context.xml	29/05/2019 11:38	Fichier XML	2 Ko
jaspic-providers.xml	29/05/2019 11:38	Fichier XML	2 Ko
jaspic-providers.xsd	29/05/2019 11:38	Fichier XSD	3 Ko
logging.properties	29/05/2019 11:38	Fichier PROPERTIES	4 Ko
server.xml	29/05/2019 11:38	Fichier XML	8 Ko
tomcat-users.xml	29/05/2019 11:38	Fichier XML	3 Ko
tomcat-users.xsd	29/05/2019 11:38	Fichier XSD	3 Ko
web.xml	29/05/2019 11:38	Fichier XML	169 Ko

Optional steps

8. If you wish you can copy and replace the **users** directory from **{current_server_path}/execution_home/** to **{new_server_path}/execution_home/**

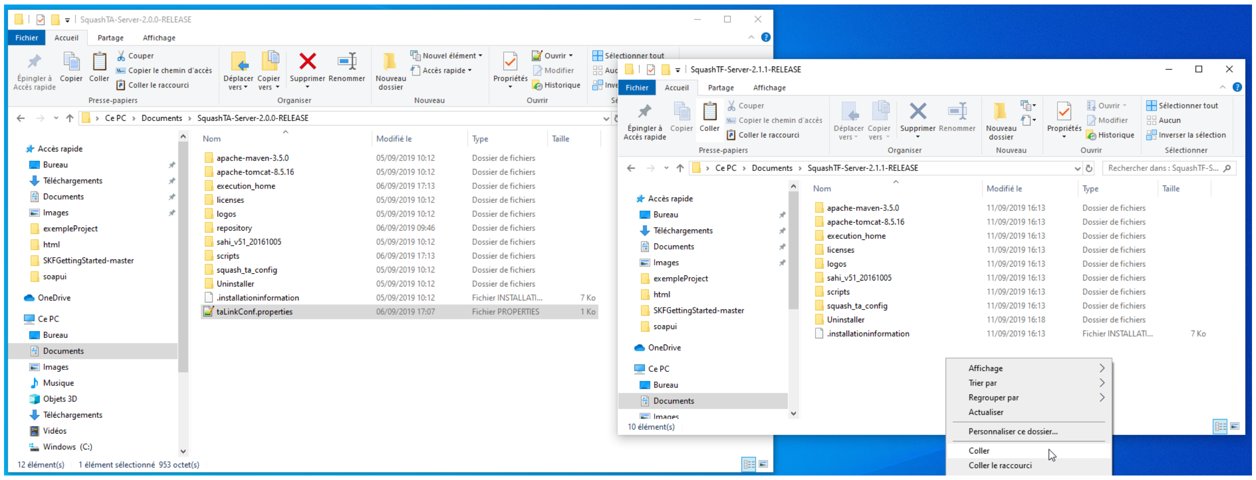
Tip: If this step isn't done the only user will be the administrator. By default -> login : admin // password : admin

9. If you wish you can copy Jenkins plugins you personally installed from **{current_server_path}/execution_home/plugins/** to **{new_server_path}/execution_home/plugins/**

Caution: Be sure to not overwrite one of the plugins we provide with a newer or older version to ensure compatibility. You could also re-install your plugins using the Jenkins interface later.

TF - TM connection update

10. Copy and replace the file **ta.linkConf.properties** from the root of your old execution server to the root of your new one


SquashTA-Server-2.0.0-RELEASE
Fichier
Accueil
Partage
Affichage
Ce PC > Documents > SquashTA-Server-2.0.0-RELEASE
SquashTF-Server-2.1.1-RELEASE
Ce PC > Documents > SquashTF-Server-2.1.1-RELEASE
Affichage
Trier par
Regrouper par
Actualiser
Personnaliser ce dossier...
Coller
Coller le raccourci

Nom	Modifié le	Type	Taille
apache-maven-3.5.0	05/09/2019 10:12	Dossier de fichiers	
apache-tomcat-8.5.16	05/09/2019 10:12	Dossier de fichiers	
execution_home	06/09/2019 17:13	Dossier de fichiers	
licenses	05/09/2019 10:12	Dossier de fichiers	
logos	05/09/2019 10:12	Dossier de fichiers	
repository	06/09/2019 09:46	Dossier de fichiers	
sahi_v51_20161005	05/09/2019 10:12	Dossier de fichiers	
scripts	06/09/2019 17:13	Dossier de fichiers	
squash_ta_config	05/09/2019 10:12	Dossier de fichiers	
Uninstaller	05/09/2019 10:12	Dossier de fichiers	
.installationinformation	05/09/2019 10:12	Fichier INSTALLATI...	7 Ko
taLinkConf.properties	06/09/2019 17:07	Fichier PROPERTIES	1 Ko

Nom	Modifié le	Type	Taille
apache-maven-3.5.0	11/09/2019 16:13	Dossier de fichiers	
apache-tomcat-8.5.16	11/09/2019 16:13	Dossier de fichiers	
execution_home	11/09/2019 16:13	Dossier de fichiers	
licenses	11/09/2019 16:13	Dossier de fichiers	
logos	11/09/2019 16:13	Dossier de fichiers	
sahi_v51_20161005	11/09/2019 16:13	Dossier de fichiers	
scripts	11/09/2019 16:13	Dossier de fichiers	
squash_ta_config	11/09/2019 16:13	Dossier de fichiers	
Uninstaller	11/09/2019 16:18	Dossier de fichiers	
.installationinformation	11/09/2019 16:13	Fichier INSTALLATI...	7 Ko

---

Docker update

If your **Squash TF Execution Server** is deployed in a Docker container the updating procedure is slightly different.

In order to keep your Jenkins jobs parameterized and your custom Jenkins configuration, you will need to extract data from you old container (default name : squash-ta-server) and provide it to your new one.

Warning: We recommend creating a backup of your current Execution Server container before beginning the update process to prevent the loss of data.

Tip: The files to keep during the update are the same as those transferred during a physical install.

If you are an experimented Docker user you can choose your preferred method to access and move the data between your containers.

Simply follow the *physical server update* process to know what files and directories to keep and transfer over.

Note: In this guide we will assume the name of the container running **Squash TF Execution Server** and the path to the server's files are the default (i.e. respectively squash-execution-server and /opt/squash-ta-server).

In this guide we will use the **docker cp** command exclusively to keep things simple but you could also use Docker Volumes or other means to access data from your containers.

The *cp* command can be used on a running container but we recommend stopping the **Squash TF Execution Server** container during the update process.

1. Copy the content of **/opt/squash-ta-server** from the container to the host system :

```
docker cp squash-execution-server:/opt/squash-ta-server {path_of_choice}/squash-server-copy
```

2. Check the page dedicated to updating to the chosen version of the server and follow the specific instructions in the **Before migrating jobs** section.

3. Filter the **execution_home** directory located in squash-server-copy :

   - Keep the **jobs** and **users** directories and all **.xml** files before deleting all the other files and directories in **execution_home**. If you use Jenkins Agents, you can also keep the **nodes** directory. To do so, use the following command :

   ```
   rm -r {path/to/squash-server-copy}/execution_home/!(\*.xml|"jobs"|
   ↪"users"|"nodes")
   ```

   - Delete the templates provided (beginning by **00** and **SquashTAConditionSweepJobs**) in the **jobs** directory.

4. Stop (if it was not already done) and rename your old **Squash TF** container to avoid naming conflict with the new one you will be creating shortly.

5. Download the newest version of **Squash TF Execution Server** Docker image from this page or on DockerHub.

6. Follow *these instructions* to launch the new **Squash TF Execution Server**.

7. Stop the new container for now :

```
docker stop squash-tf-execution-server
```

8. Copy the content of **{path/to/squash-server-copy}/execution_home** to the **execution_home** directory in the new container :

```
docker cp {/path/of/your/choice/squash-server-copy}/execution_home/. squash-execution-server:/opt/squash-ta-server/execution_home
```

---

9. Copy the server.xml file from **{path/to/squash-server-copy}/apache-tomcat-8.5.16/conf/** in the **/opt/squash-ta-server/apache-tomcat-8.5.16/conf/** in the new container :

```
docker cp {/path/of/your/choice/squash-server-copy}/apache-tomcat-8.5.16/conf/server.xml squash-execution-server:/opt/squash-ta-server/apache-tomcat-8.5.16/conf/
```

---

10. Copy any other data you wish to transfer over to the updated execution server.

---

11. If you wish you can follow step 9 or 10 of the *physical update process* to keep any Jenkins plugins you may have installed and/or update your connection with **Squash TM** .

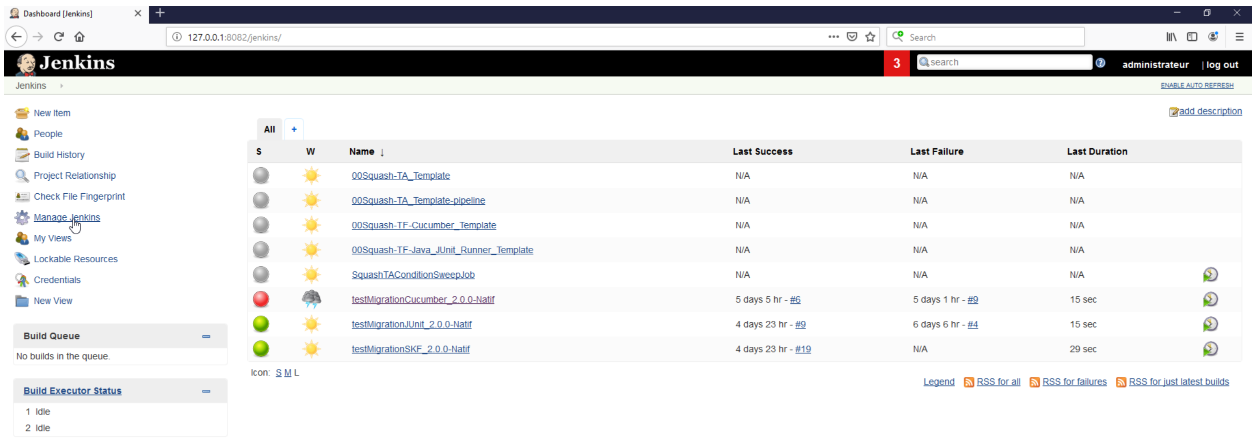
---

12. Use the docker start command to restart your server properly :

```
docker start squash-tf-execution-server
```

Jenkins configuration update

1. Start the new Execution Server and access the corresponding Jenkins page (login with an administrator account).



Build Queue

No builds in the queue.

Build Executor Status

1 Idle

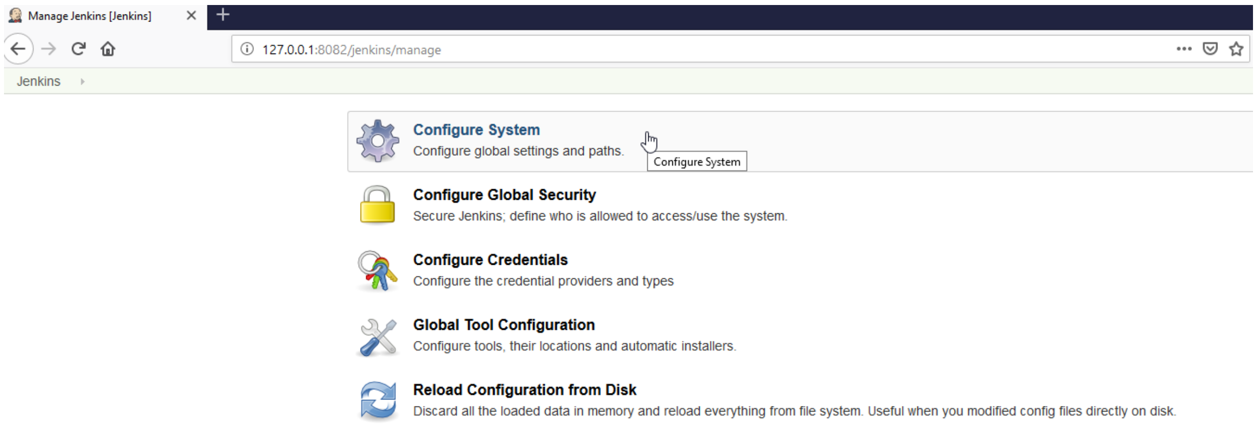
2 Idle

S	W	Name ↓	Last Success	Last Failure	Last Duration
●	☀	00Squash-TA_Template	N/A	N/A	N/A
●	☀	00Squash-TA_Template-pipeline	N/A	N/A	N/A
●	☀	00Squash-TF-Cucumber_Template	N/A	N/A	N/A
●	☀	00Squash-TF-Java_JUnit_Runner_Template	N/A	N/A	N/A
●	☀	SquashTAConditionSweepJob	N/A	N/A	N/A
🔴	⛈	testMigrationCucumber_2.0.0-Natif	5 days 5 hr - #6	5 days 1 hr - #9	15 sec
🟢	☀	testMigrationJUnit_2.0.0-Natif	4 days 23 hr - #9	6 days 6 hr - #4	15 sec
🟢	☀	testMigrationSKF_2.0.0-Natif	4 days 23 hr - #19	N/A	29 sec

Icon: S M L

Legend RSS for all RSS for failures RSS for just latest builds

2. Go to `Manage Jenkins` followed by `Configure system`:
   - Find the section `Global Pipeline Libraries` and the subsection `Source Code Management`.
   - In the field `Repository URL` replace the path of your old version with the new one (everything before **/execution_home/pipeline-libs/..**).
   - Check that the Jenkins url is the right one for your new execution server (**TM-TF** link).
   - Click on `Save`.


Manage Jenkins [Jenkins]

127.0.0.1:8082/jenkins/manage

Jenkins ›

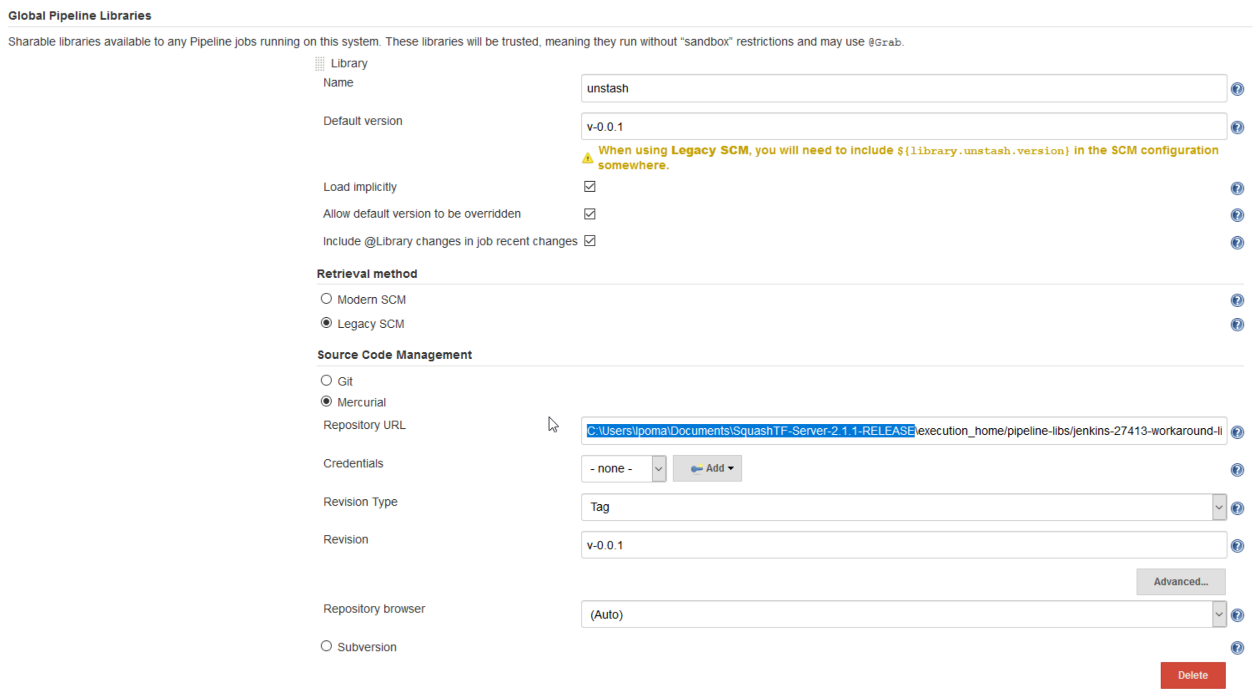
Configure System
Configure global settings and paths.

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

Configure Credentials
Configure the credential providers and types

Global Tool Configuration
Configure tools, their locations and automatic installers.

Reload Configuration from Disk
Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.


Global Pipeline Libraries

Sharable libraries available to any Pipeline jobs running on this system. These libraries will be trusted, meaning they run without "sandbox" restrictions and may use `@Grab`.

Library

Name	unstash
Default version	v-0.0.1
	When using **Legacy SCM**, you will need to include `${library.unstash.version}` in the SCM configuration somewhere.
Load implicitly	✓
Allow default version to be overridden	✓
Include @Library changes in job recent changes	✓

Retrieval method

Modern SCM [✓ Legacy SCM]

Source Code Management

Git [✓ Mercurial]

Repository URL	C:\Users\lpoma\Documents\SquashTF-Server-2.1.1-RELEASE\execution_home/pipeline-libs/jenkins-27413-workaround-li
Credentials	- none -
Revision Type	Tag
Revision	v-0.0.1
Repository browser	(Auto)

Subversion

3. Go to `Global Tool Configuration`:
   - Find the `Maven` section and click on **Maven installations…** if the section is empty.
   - In the field `MAVEN_HOME` replace the path of your old version with the new one (everything before **/apache-maven-3.5.0**).
   - Click on `Save`.

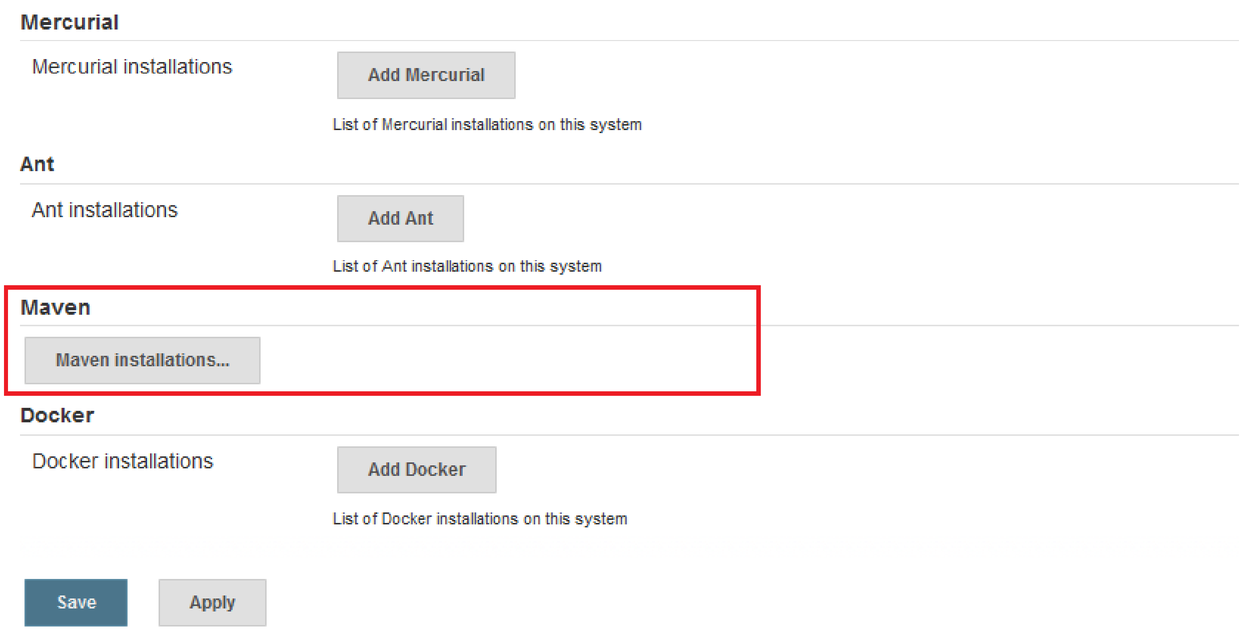
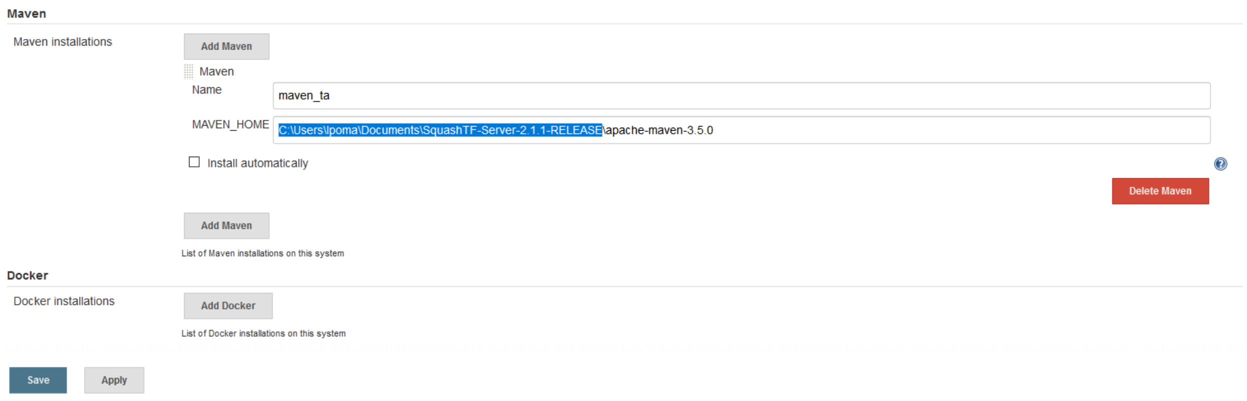


Fig. 1: Empty Maven section

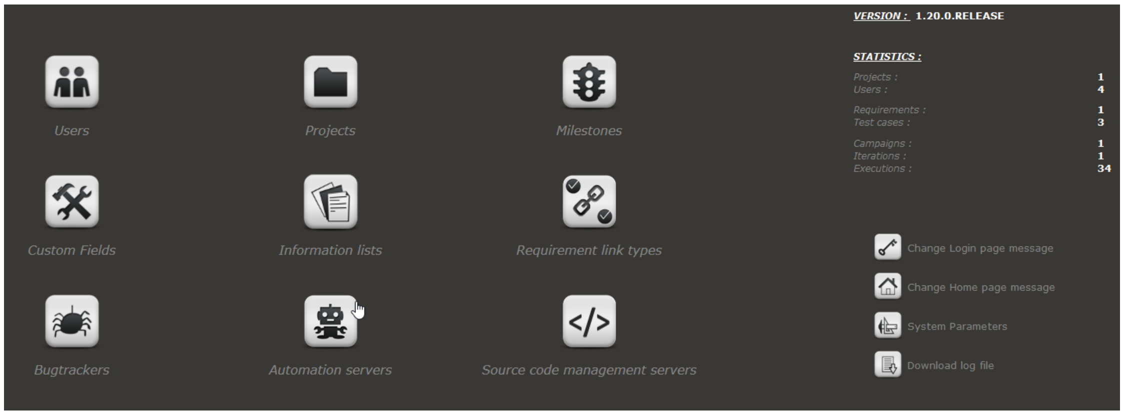


Maven
Name: maven_ta
MAVEN_HOME: C:\Users\poma\Documents\SquashTF-Server-2.1.1-RELEASE\apache-maven-3.5.0
☐ Install automatically

Fig. 2: New path

4. If you have some of your own Jenkins plugins that you didn't copy to the new plugin directory in the new execution server you can download them again at this time in the process.

5. In case of a **TM-TF** link remember to also check Jenkins url in Squash TM (`Administration` then `Automation Servers`):



VERSION : 1.20.0.RELEASE

STATISTICS :

Projects :	1
Users :	4
Requirements :	1
Test cases :	3
Campaigns :	1
Iterations :	1
Executions :	34

6. Check the page dedicated to updating to the chosen version of the server and follow the specific instructions in the **After migrating jobs** section.

7. Finally test your migrated jobs to confirm the update was successfull and your previously working jobs still behave as you want them to.

Jenkins Agents update

See *here*

Specific migration from 2.0.0 to 2.1.1

Before migrating jobs

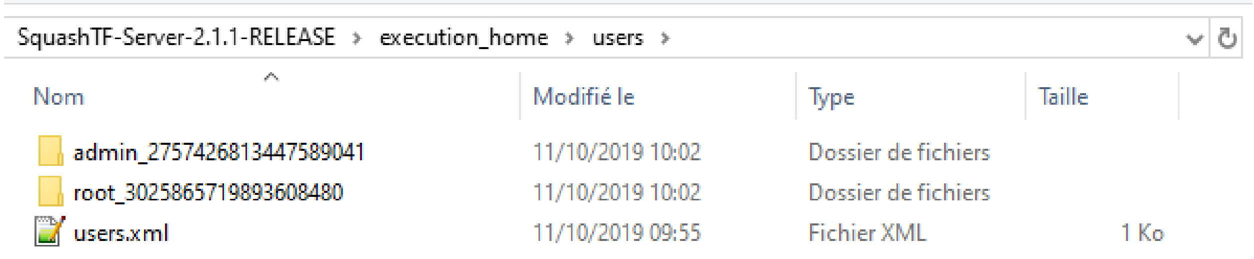
Updating Jenkins users

Due to security updates on Jenkins LTS 2.107 and 2.138 your users will have to be updated prior to starting the 2.1.1 version of **Squash TF Execution Server**.

1. Delete the **users** directory from **{new_server_path}/execution_home/**
2. Copy the **users** directory from **{current_server_path}/execution_home/** to **{new_server_path}/execution_home/** (keep a backup copy somewhere)
3. Check that there is NO **users.xml** file at the root of the new **users** directory. It will automatically be created later.

The users' updating process should automatically be done by Jenkins at the first start of the server, later in the generic process. You can comme back at this time to ensure the users' update was successful :

- All the users are visible on the people page on Jenkins.
- A **users.xml** file should appear in the **users** directory of your new **Squash TF Execution Server** with the list of all your users.
- Each directory corresponding to a user should have a number following the user's name.


SquashTF-Server-2.1.1-RELEASE › execution_home › users ›

Nom	Modifié le	Type	Taille
admin_2757426813447589041	11/10/2019 10:02	Dossier de fichiers	
root_3025865719893608480	11/10/2019 10:02	Dossier de fichiers	
users.xml	11/10/2019 09:55	Fichier XML	1 Ko

For more details about this process visit the Jenkins upgrade guide for versions 2.107 and 2.138

Warning: This auto update is entirely managed by Jenkins. If you encounter any problem with your users (missing entries, bad API token or credentials. . .) we recommend retrying the process by stopping the server and doing steps 1 to 3 of this guide again.

Note: You could also recreate all the users from scratch. To do so, keep the **users** directory provided in a new install of the **Squash TF Execution Server 2.1.1** instead of deleting it at step 2. Start the server to connect to Jenkins with the default administrator account (login : admin // password : admin) and recreate all your users from there.

Go back to the generic update procedure for now.

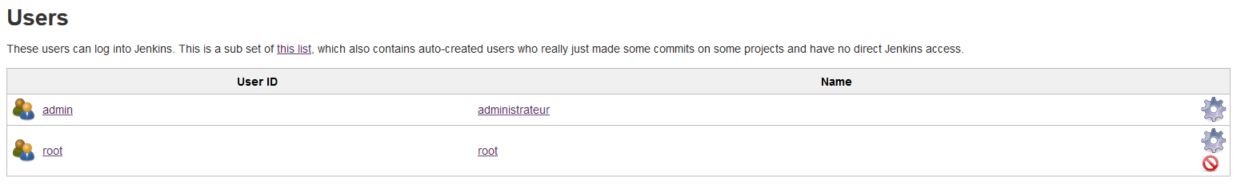
After migrating jobs

Verify users update

Prerequisites : You have followed the earlier instructions (in the **Before migrating jobs** section) to update Jenkins users.

Start your **Squash TF Execution Server 2.1.1** and connect to Jenkins as administrator.

Go to `Manage Jenkins` then to `Manage Users` and check that all your users appear in the list :



Users

These users can log into Jenkins. This is a sub set of this list, which also contains auto-created users who really just made some commits on some projects and have no direct Jenkins access.

User ID	Name
admin	administrateur
root	root

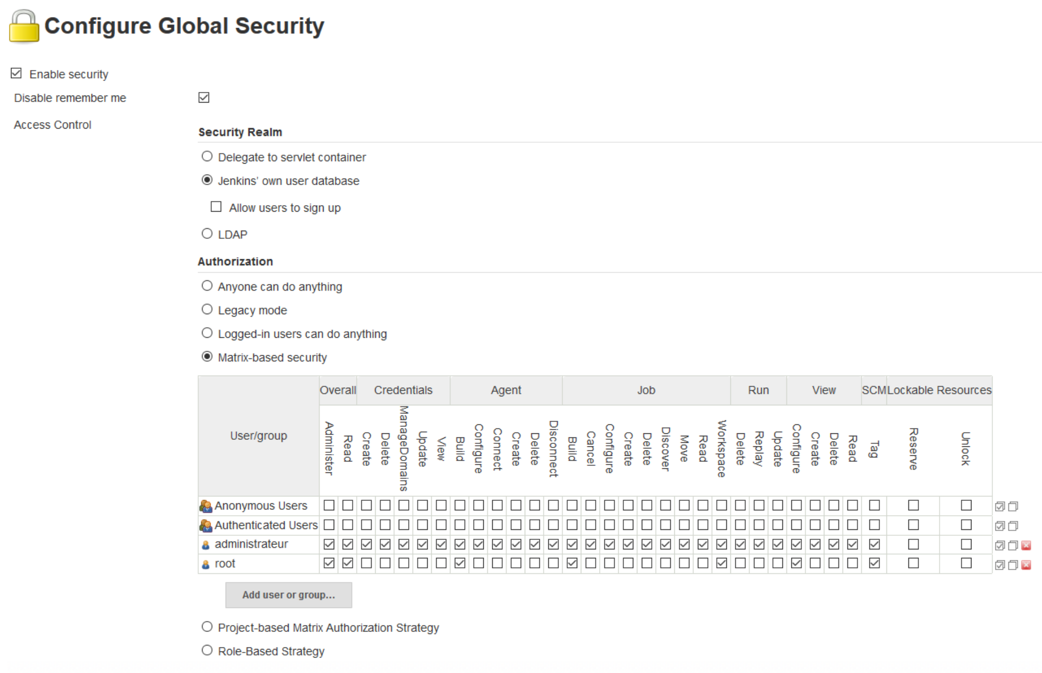
If some of your users are missing you should stop the server, delete the **users** directory in **execution_home** (v 2.1.1) and copy again the old **user** directory (v 2.0.0) in its place (See ***Before migrating jobs***).

Go back to `Manage Jenkins` then to `Configure Global Security` :

Check the authorization strategy used and verify the rights of each user in case of a matrix-based strategy.

If there is an error at this point you should retrieve a backup version of the 2.0.0 version of **execution_home** ({old_server_path}/execution_home) and copy the **config.xml** file to {new_server_path}/execution_home/.

(This config file is where the Jenkins authorization strategy is recorded).



Configure Global Security

☑ Enable security

Disable remember me ☑

Access Control

Security Realm

○ Delegate to servlet container
● Jenkins' own user database
☐ Allow users to sign up
○ LDAP

Authorization

○ Anyone can do anything
○ Legacy mode
○ Logged-in users can do anything
● Matrix-based security

User/group	Overall		Credentials					Agent						Job									Run			View				SCM	Lockable Resources	
	Administer	Read	Create	Delete	ManageDomains	Update	View	Build	Configure	Connect	Create	Delete	Disconnect	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace	Delete	Replay	Update	Configure	Create	Delete	Read	Tag	Reserve	Unlock
Anonymous Users	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Authenticated Users	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
administrateur	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☐
root	☑	☑	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☐	☐	☑	☐	☐	☐	☑	☐	☐	☐	☑	☐	☐

○ Project-based Matrix Authorization Strategy
○ Role-Based Strategy

Use an environment variable to define the maven local repository path

Overview

Previously, the local maven repository path was hardcoded by the installer during the installation process.

Starting from version 2.1.1 this path is define by using an environment variable. This change is not take into account by the generic migration and you have to do it yourself after.

Update the Maven Global Settings

Prerequisites : The generic migration should have been done.

Start your **Squash TF Execution Server 2.1.1** and connect to Jenkins as administrator.

Go to `Manage Jenkins` then to `Config File Management` :

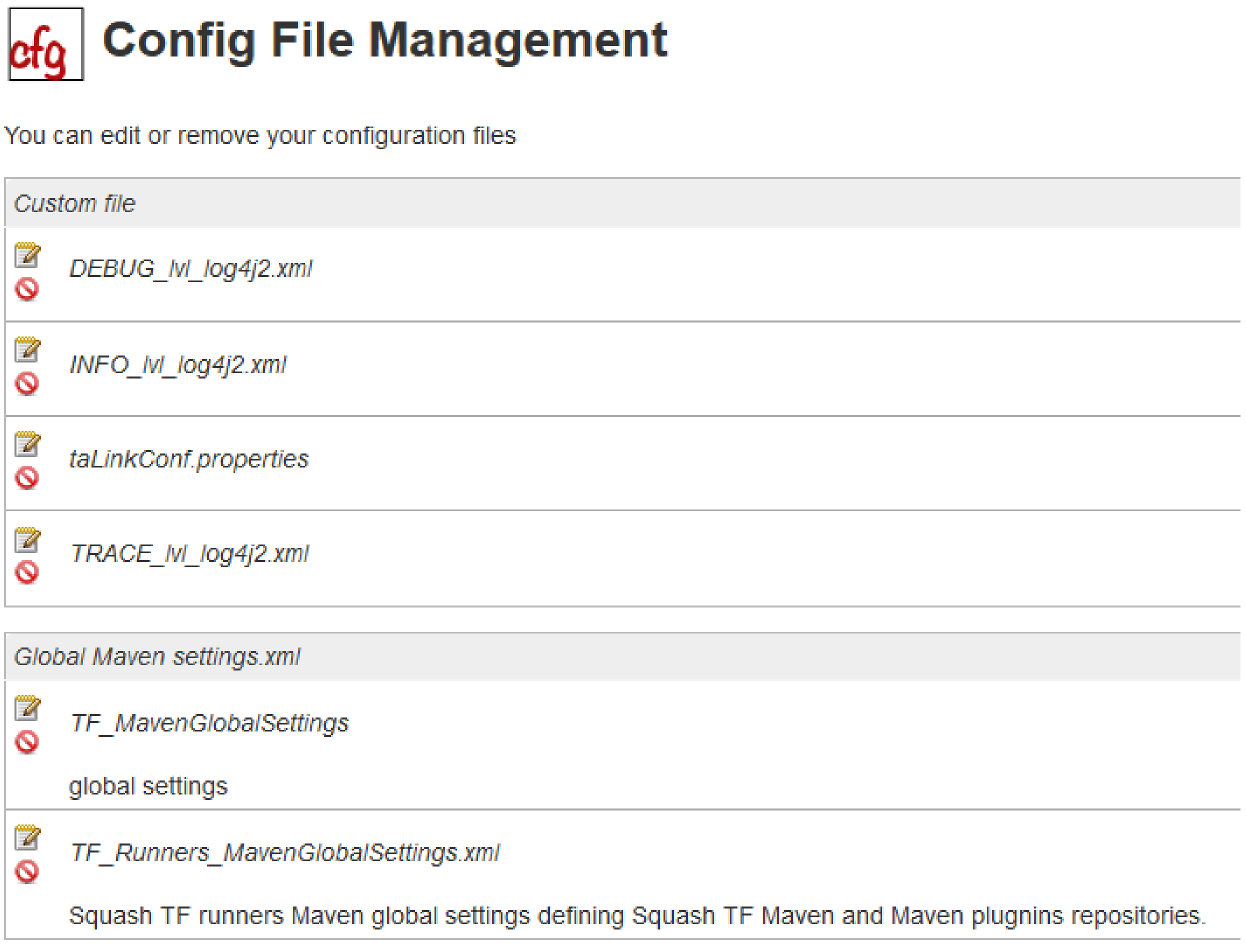
Open the file *TF_MavenGlobalSettings* and find the property <localRepository>.

Replace the line by :

```
<localRepository>${env.SQUASH_TA_HOME}/repository</localRepository>
```

and submit the changes.

Repeat this procedure for the file *TF_Runners_MavenGlobalSettings.xml* and submit again.

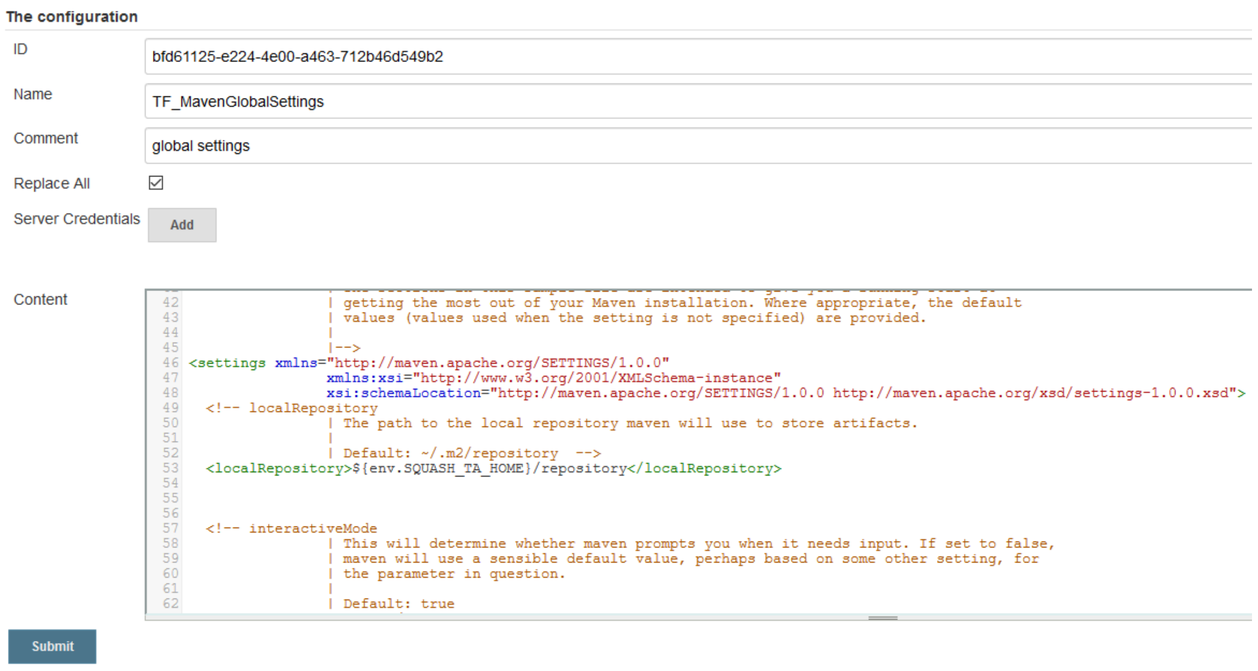


Config File Management

You can edit or remove your configuration files

Custom file
DEBUG_lvl_log4j2.xml
INFO_lvl_log4j2.xml
taLinkConf.properties
TRACE_lvl_log4j2.xml

Global Maven settings.xml
TF_MavenGlobalSettings global settings
TF_Runners_MavenGlobalSettings.xml Squash TF runners Maven global settings defining Squash TF Maven and Maven plugnins repositories.

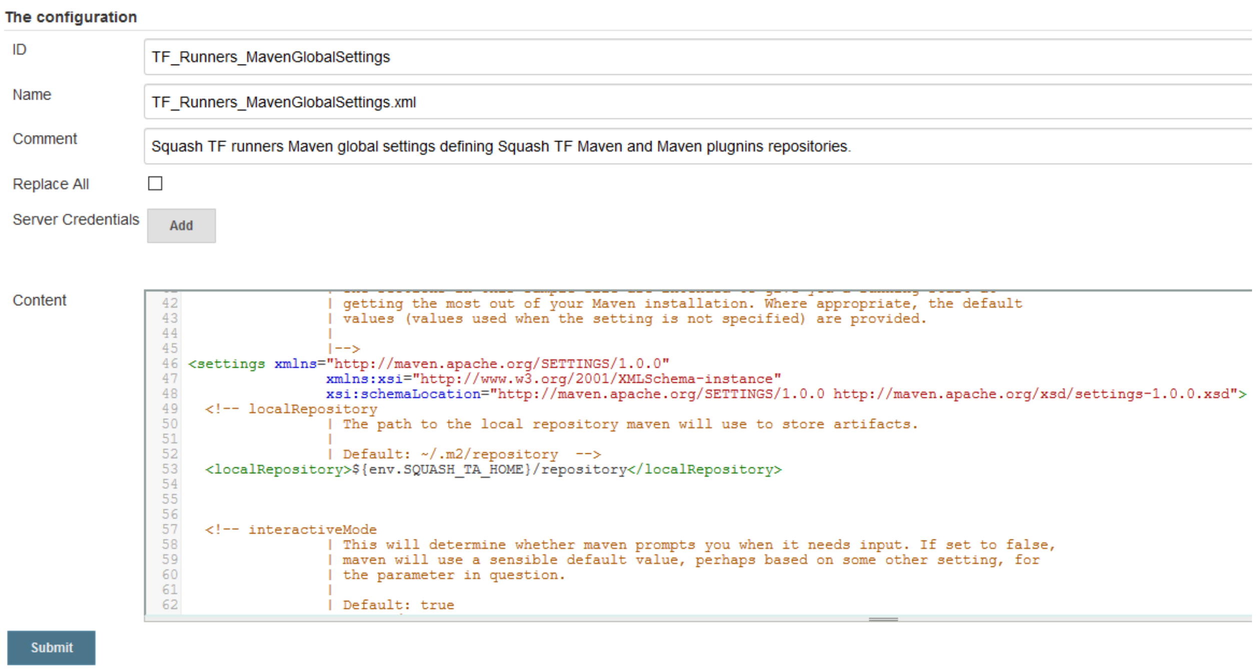


The configuration

ID	bfd61125-e224-4e00-a463-712b46d549b2
Name	TF_MavenGlobalSettings
Comment	global settings
Replace All	☑
Server Credentials	Add

Content

```
                 | getting the most out of your Maven installation. Where appropriate, the default
                 | values (values used when the setting is not specified) are provided.
                 |
                 |-->
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <!-- localRepository
                 | The path to the local repository maven will use to store artifacts.
                 |
                 | Default: ~/.m2/repository  -->
  <localRepository>${env.SQUASH_TA_HOME}/repository</localRepository>



  <!-- interactiveMode
                 | This will determine whether maven prompts you when it needs input. If set to false,
                 | maven will use a sensible default value, perhaps based on some other setting, for
                 | the parameter in question.
                 |
                 | Default: true
```



The configuration

ID	TF_Runners_MavenGlobalSettings
Name	TF_Runners_MavenGlobalSettings.xml
Comment	Squash TF runners Maven global settings defining Squash TF Maven and Maven plugnins repositories.
Replace All	☐
Server Credentials	Add

Content

```
                 | getting the most out of your Maven installation. Where appropriate, the default
                 | values (values used when the setting is not specified) are provided.
                 |
                 |-->
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <!-- localRepository
                 | The path to the local repository maven will use to store artifacts.
                 |
                 | Default: ~/.m2/repository  -->
  <localRepository>${env.SQUASH_TA_HOME}/repository</localRepository>



  <!-- interactiveMode
                 | This will determine whether maven prompts you when it needs input. If set to false,
                 | maven will use a sensible default value, perhaps based on some other setting, for
                 | the parameter in question.
                 |
                 | Default: true
```

New reports publication settings

Overview

The jobs template settings for the reports publication have changed in **Squash TF Execution Server 2.1.1**. This change is due to a problem in the report publication url in Jenkins which was breaking the **TM-TF** link

This change have been made to all the templates.

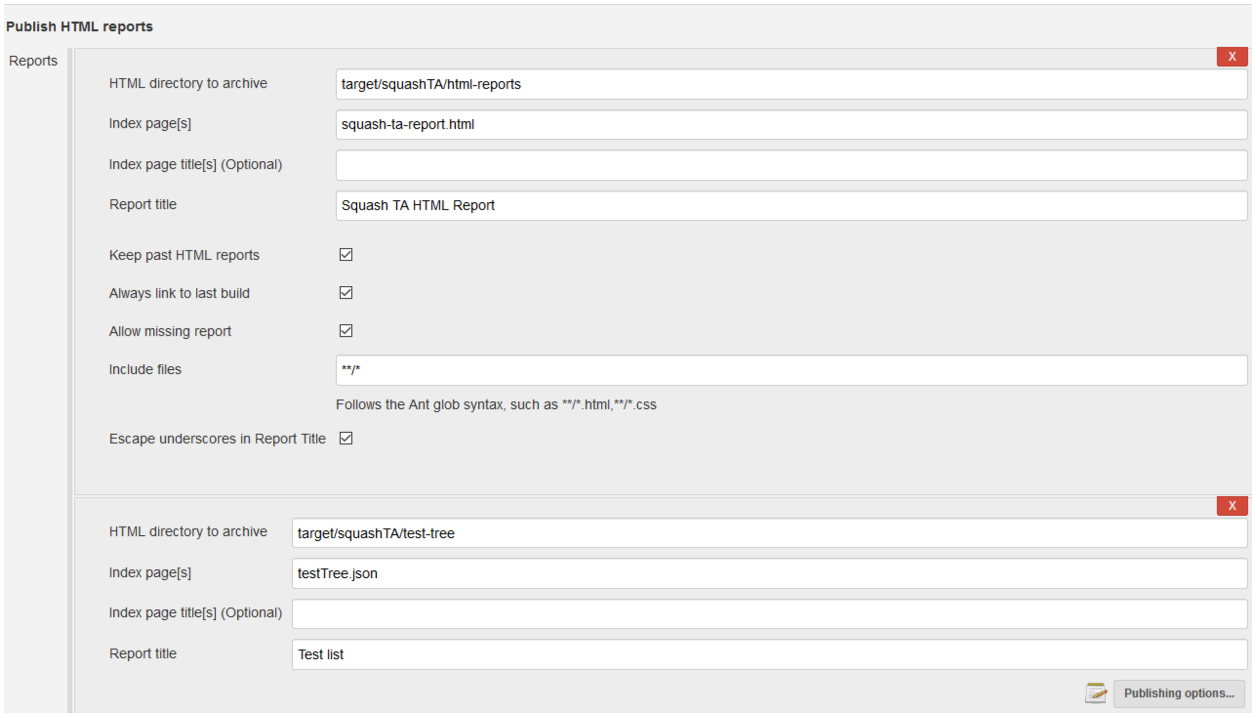
Already existing jobs will have to be updated manually.

Note: These new reports publication settings are mandatory in case of a **TF-TM** link configuration and highly recommended in other cases.

Update existing jobs

In your job, go to `Post-build Actions` :

- Replace all the spaces in the `Report title` fields for each report generated by the job.
- Click on `Publishing options` under each report form.
- The checkbox `Escape underscores in Report Title` now needs to be unchecked.


Publish HTML reports

Reports

HTML directory to archive	target/squashTA/html-reports
Index page[s]	squash-ta-report.html
Index page title[s] (Optional)	
Report title	Squash TA HTML Report
Keep past HTML reports	☑
Always link to last build	☑
Allow missing report	☑
Include files	**/*
	Follows the Ant glob syntax, such as **/*.html,**/*.css
Escape underscores in Report Title	☑

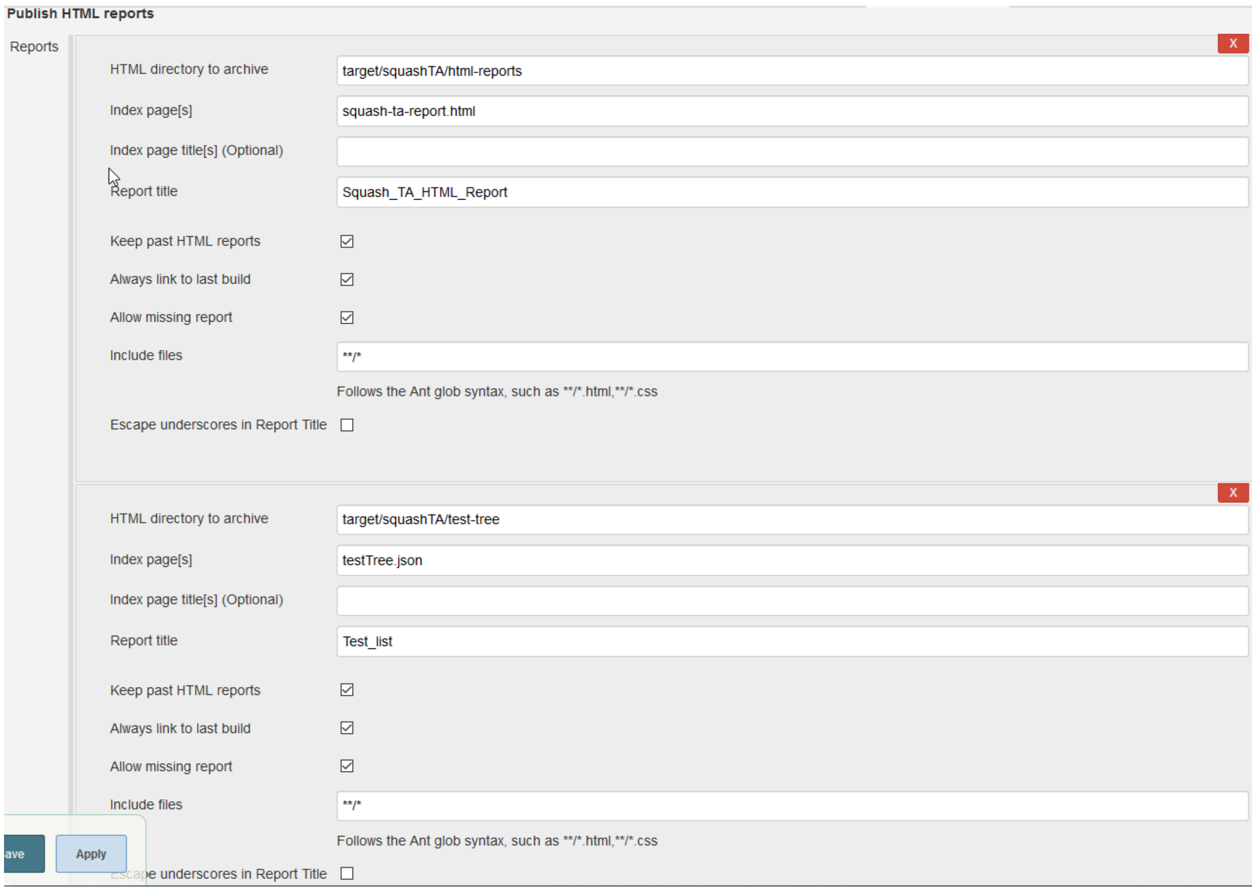
HTML directory to archive	target/squashTA/test-tree
Index page[s]	testTree.json
Index page title[s] (Optional)	
Report title	Test list

Fig. 3: Report forms before corrections

Final step

When all the procedures have been done you should restart your **Execution Server**, then reconnect to Jenkins and check that all the corrections have been successfully implemented.

Specific migration from 2.1.1 to 2.2.0



Publish HTML reports

Reports

HTML directory to archive	target/squashTA/html-reports
Index page[s]	squash-ta-report.html
Index page title[s] (Optional)	
Report title	Squash_TA_HTML_Report
Keep past HTML reports	☑
Always link to last build	☑
Allow missing report	☑
Include files	**/*
	Follows the Ant glob syntax, such as **/*.html,**/*.css
Escape underscores in Report Title	☐

HTML directory to archive	target/squashTA/test-tree
Index page[s]	testTree.json
Index page title[s] (Optional)	
Report title	Test_list
Keep past HTML reports	☑
Always link to last build	☑
Allow missing report	☑
Include files	**/*
	Follows the Ant glob syntax, such as **/*.html,**/*.css
Escape underscores in Report Title	☐

Fig. 4: Report forms after corrections

Before migrating jobs

No particular task needs to be done at this point for this update. Go back to the generic update procedure for now.

After migrating jobs

New reports publication settings

Overview

Since **Squash TF Execution Server 2.2.0**, we have changed our jobs template settings for the reports publication. Now the publication is done according to the operation executed :

- If it's a "list" operation, then we publish the tests list report.
- If it's a "run" operation, then we publish the tests execution report.

The change have been made to these templates :

- 00Squash-TA_Template.
- 00Squash-TF-Java_JUnit_Runner_Template.

Already existing jobs will have to be updated manually

Note: This new reports publication settings is mandatory for tests projects using :

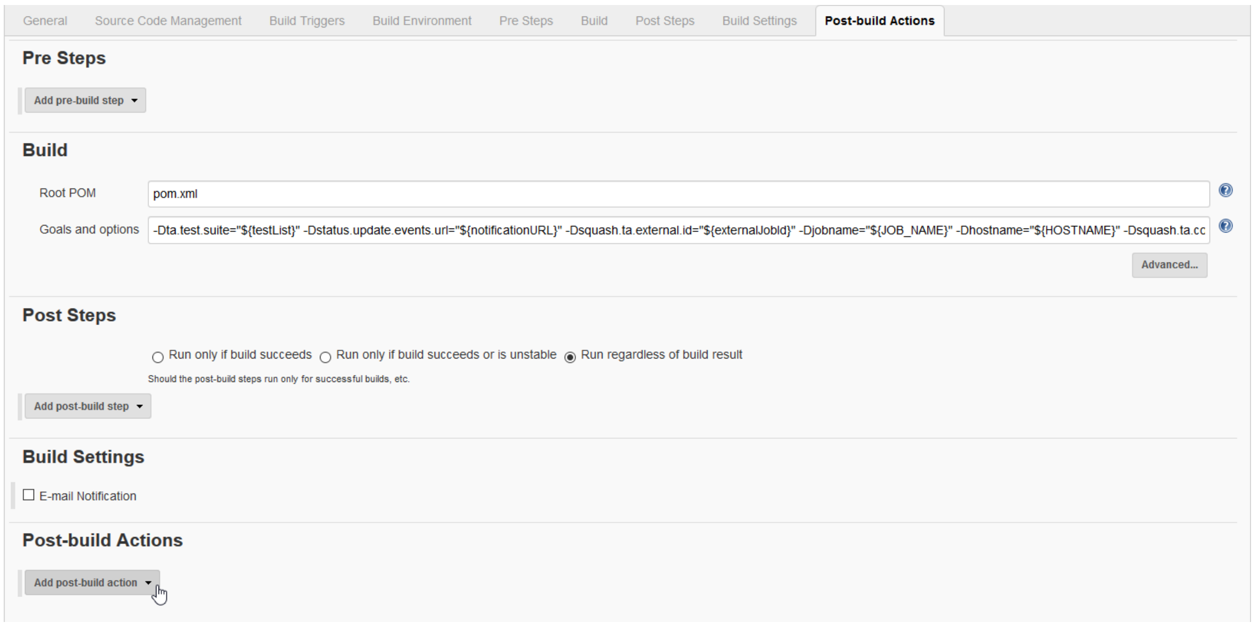
- **Squash TFJava Junit Runner** : all versions.
- **Squash TFKeyword framework (SKF)** : for version 1.13.0 and above.

Update existing jobs

Prerequisite

The `Flexible publish` plugin should have been installed in Jenkins (done in **Squash TF Execution Server 2.2.0**).

In your job, go to `Post-build Actions` and add a post-build action :



Pre Steps

Build

Root POM: pom.xml

Goals and options: -Dta.test.suite="${testList}" -Dstatus.update.events.url="${notificationURL}" -Dsquash.ta.external.id="${externalJobId}" -Djobname="${JOB_NAME}" -Dhostname="${HOSTNAME}" -Dsquash.ta.cc

Post Steps

Run only if build succeeds / Run only if build succeeds or is unstable / [✓ Run regardless of build result]

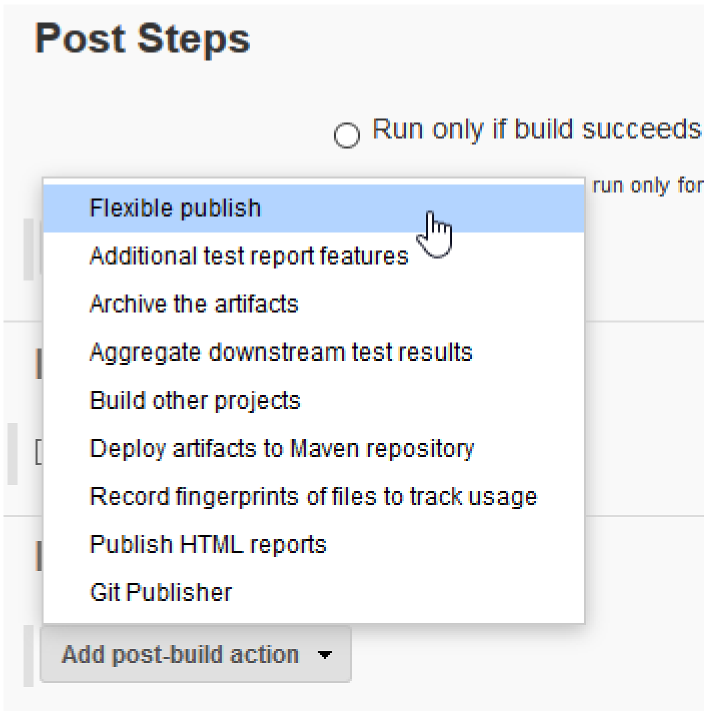
Should the post-build steps run only for successful builds, etc.

Build Settings

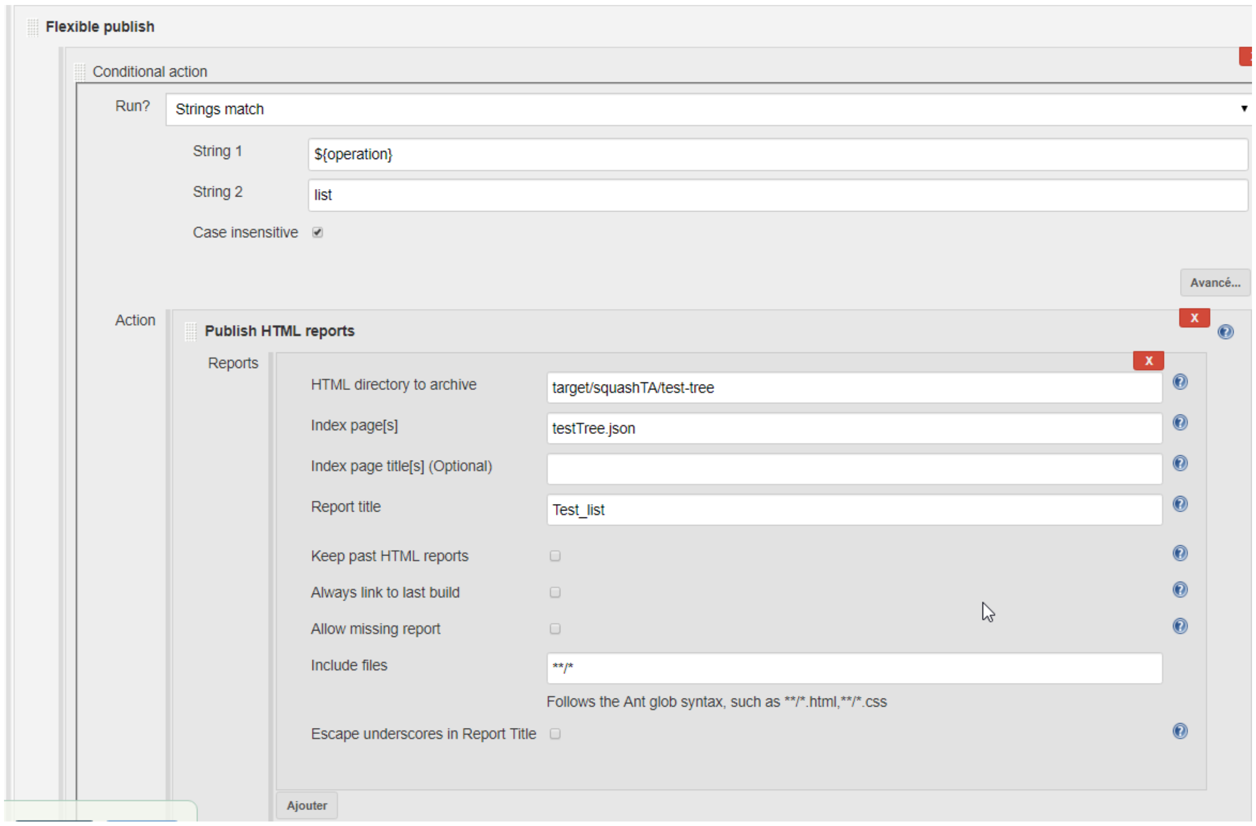
☐ E-mail Notification

Post-build Actions

Select `Flexible publish` :


Post Steps
Run only if build succeeds
run only for
Flexible publish
Additional test report features
Archive the artifacts
Aggregate downstream test results
Build other projects
Deploy artifacts to Maven repository
Record fingerprints of files to track usage
Publish HTML reports
Git Publisher
Add post-build action

Configure the conditional action for the goal **list** as shown in the screen below :


Flexible publish

Conditional action

Run? Strings match

String 1 ${operation}

String 2 list

Case insensitive ✓

Action

Publish HTML reports

Reports

HTML directory to archive target/squashTA/test-tree

Index page[s] testTree.json

Index page title[s] (Optional)

Report title Test_list

Keep past HTML reports ☐

Always link to last build ☐

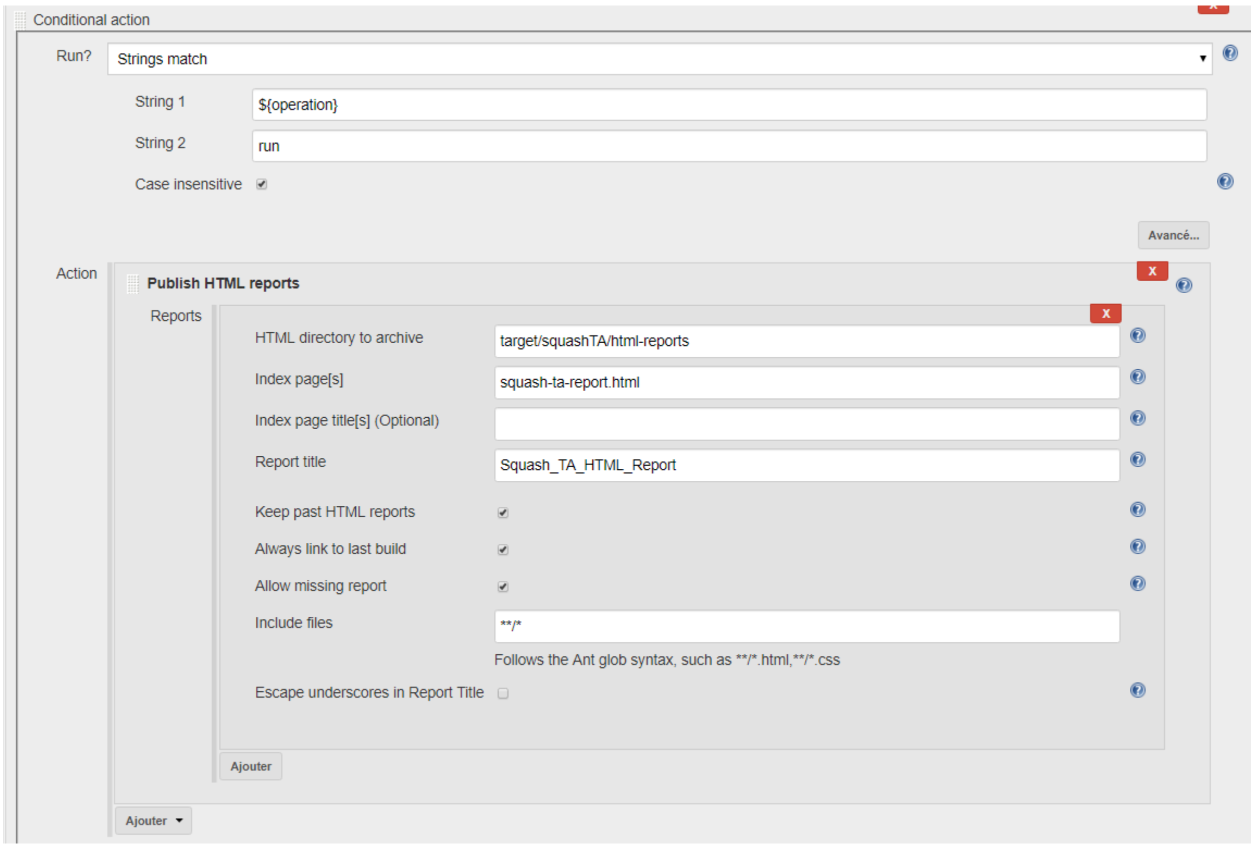
Allow missing report ☐

Include files **/*

Follows the Ant glob syntax, such as **/*.html,**/*.css

Escape underscores in Report Title ☐

Then add another conditional action and do the same for the goal **run** :


Conditional action

Run? Strings match

String 1 ${operation}

String 2 run

Case insensitive ✓

Action

Publish HTML reports

Reports

HTML directory to archive target/squashTA/html-reports

Index page[s] squash-ta-report.html

Index page title[s] (Optional)

Report title Squash_TA_HTML_Report

Keep past HTML reports ✓

Always link to last build ✓

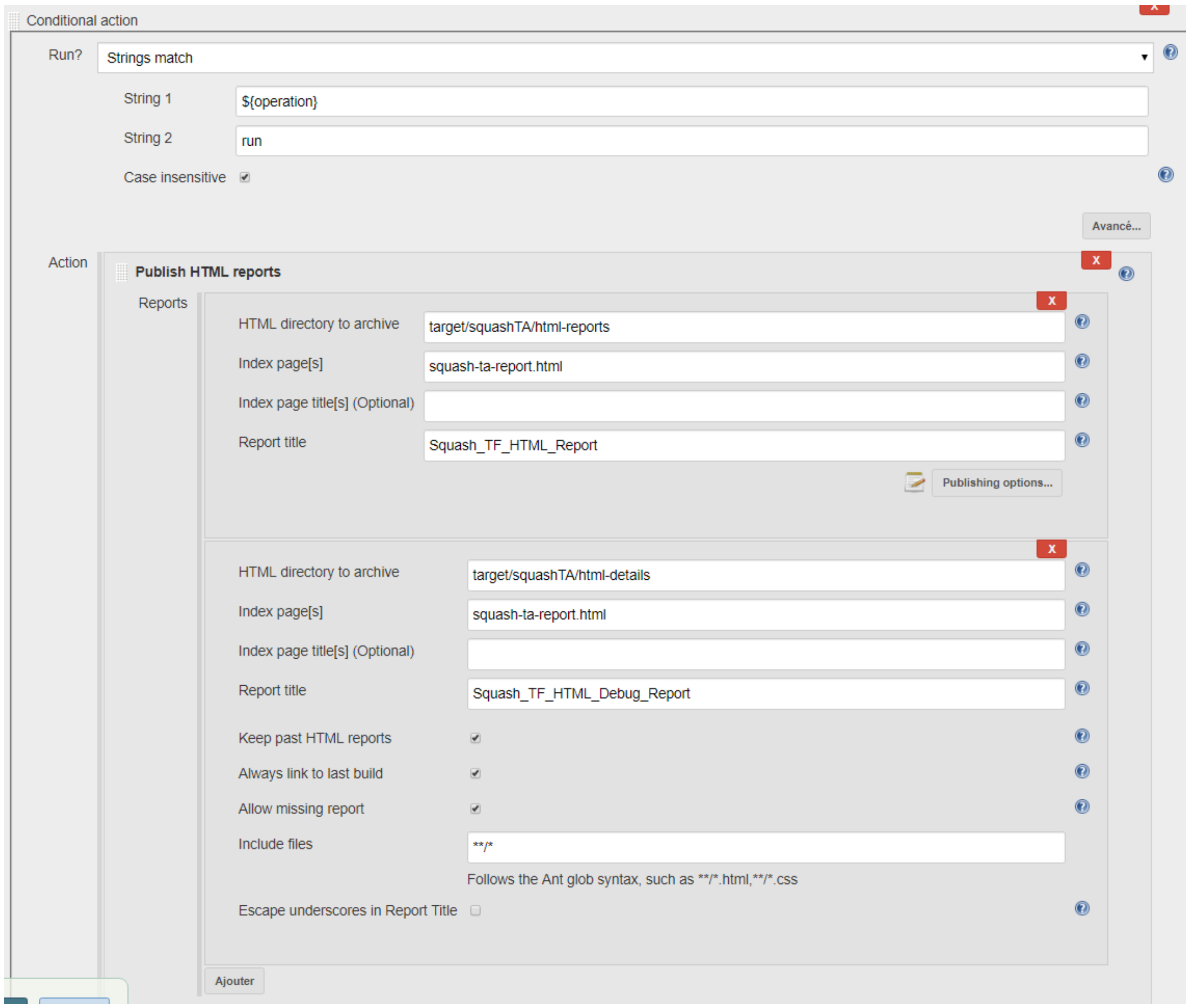
Allow missing report ✓

Include files **/*

Follows the Ant glob syntax, such as **/*.html,**/*.css

Escape underscores in Report Title ☐

For jobs which run tests using **Squash TFJava Junit Runner**, also add the "Squash_TF_HTML_Debug_Report" in the goal **run** :



Conditional action

Run? Strings match

String 1 ${operation}

String 2 run

Case insensitive ✓

Action

Publish HTML reports

Reports

HTML directory to archive target/squashTA/html-reports

Index page[s] squash-ta-report.html

Index page title[s] (Optional)

Report title Squash_TF_HTML_Report

HTML directory to archive target/squashTA/html-details

Index page[s] squash-ta-report.html

Index page title[s] (Optional)

Report title Squash_TF_HTML_Debug_Report

Keep past HTML reports ✓

Always link to last build ✓

Allow missing report ✓

Include files **/*

Follows the Ant glob syntax, such as **/*.html,**/*.css

Escape underscores in Report Title ☐

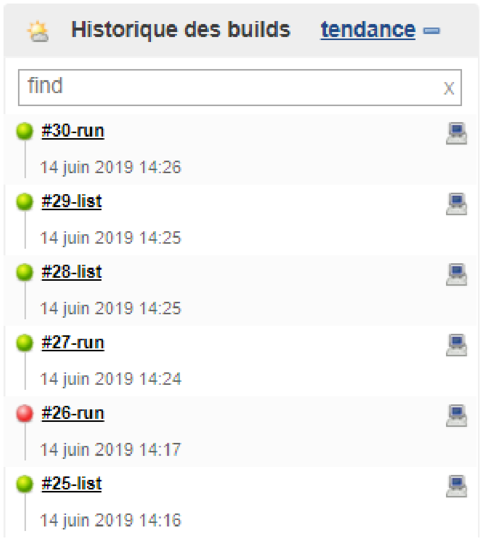
Finally delete all previous post build actions for publication.

Note: As explain in *the specific update documentation from 2.0.0 to 2.1.1*, the report title should contain "_" (and not a space), and `Escape underscores in Report Title` should be unchecked.

Rename job execution according to the executed operation

Overview

In **Squash TF Execution Server 2.2.0**, we introduce the renaming of a job execution (according to the executed operation). To do so, we use the Jenkins `Build Name and Description Setter` plugin. This plugin sets the display name of a build to something other than #1, #2, #3, . . . so that you can use an identifier that makes more sense in your context (see image).


Historique des builds tendance

find

#30-run
14 juin 2019 14:26

#29-list
14 juin 2019 14:25

#28-list
14 juin 2019 14:25

#27-run
14 juin 2019 14:24

#26-run
14 juin 2019 14:17

#25-list
14 juin 2019 14:16

Update existing jobs

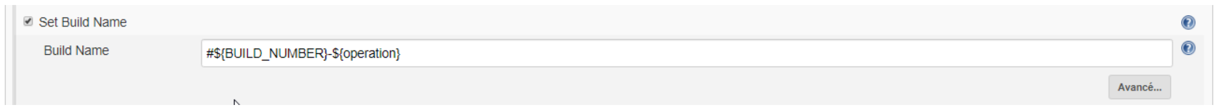
Prerequisite

The `Build Name and Description Setter` plugin should have been installed in Jenkins (done in **Squash TF Execution Server 2.2.0**).

In your job, go the `Build Environment` section and check the box `Set Build Name`.

In the `Build Name`, just add :

```
#${BUILD_NUMBER}-${operation}
```


Set Build Name

Build Name: #${BUILD_NUMBER}-${operation}

1.1.3 Execution Server - Troubleshooting

Insufficient Cache warnings in catalina logs

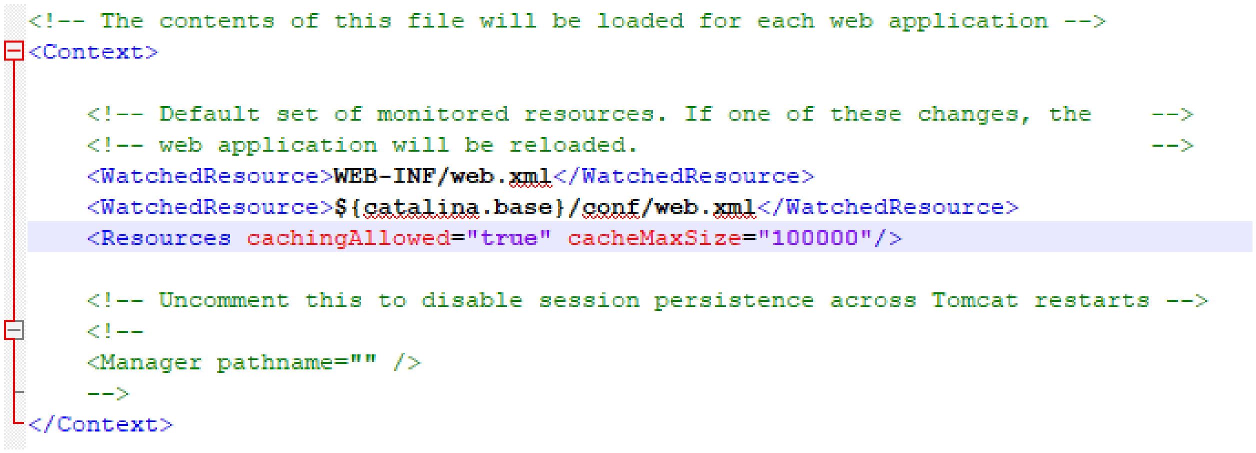
During the launch of the execution server, you might encounter a warning message stating that a resource could not be added because there was insufficient free space available in your cache memory (see the full message in the example below) :

```
08-Jul-2019 16:04:23.575 AVERTISSEMENT [Loading plugin Environment Injector Plugin v2.
↪1.6 (envinject)] org.apache.catalina.webresources.Cache.getResource Unable to add␣
↪the resource at [/WEB-INF/classes/org/jenkinsci/plugins/envinject/Messages_en.
↪properties] to the cache for web application [/jenkins] because there was␣
↪insufficient free space available after evicting expired cache entries - consider␣
↪increasing the maximum size of the cache
```

In order to fix this problem, you have to increase the cache max size. To do so, go in the Apache Tomcat Folder of your execution server and then go to the 'conf' folder.

Open context.xml and in the context block, add the following property :

```
<Resources cachingAllowed="true" cacheMaxSize="100000"/>
```



```
<!-- The contents of this file will be loaded for each web application -->
<Context>

    <!-- Default set of monitored resources. If one of these changes, the    -->
    <!-- web application will be reloaded.                                   -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>
    <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>
    <Resources cachingAllowed="true" cacheMaxSize="100000"/>

    <!-- Uncomment this to disable session persistence across Tomcat restarts -->
    <!--
    <Manager pathname="" />
    -->
</Context>
```

1.2 Execution Agent - Administration

1.2.1 Execution Agent installation

Squash TF Execution Server is Jenkins based. As such one can setup a master/slave architecture with clear separation of concerns. The Jenkins master schedules test related tasks whereas slave agents are solely responsible for their executions.

As the setting up of such architectures may be somewhat cumbersome, graphical agent installers are supplied as well as ready to run docker images.

Warning: In **2.3.0-RELEASE version of the server**, we updated Jenkins to 2.190.1 version. Since 2.165 and newer version, Jenkins no longer supports the old (-remoting) mode in either the client or server. If you have installed a 2.3.0-RELEASE of the server and want to install an agent, you need to download **2.3.0-RELEASE (or newer) version of the agent**.

Physical install

The graphical agent installers are meant for GNU-Linux-64 or Windows-64 environments. They will install and configure self-contained Jenkins Jnlp agents with **Squash TF** runners handling capabilities. More precisely they will :

1. Declare on the Jenkins master a new Jnlp node.
2. Retrieve the name and the secret of the created node.
3. Install the self contained agent and create a launch script.
4. Configure, on the Jenkins master, the newly created node properties such as the location of the embedded Maven.

Pre-requisites

The following pre-requisites are needed in order to perform the installation :

1. An open port on the Jenkins Master (**Squash TF Execution Server**) that can be used by the agent to connect to the master.
2. A Java Development kit 1.8 since, for licensing reasons, it is not possible to ship a our product with an embedded JDK.
3. A Java Virtual Machine with the corresponding environment variables JAVA_HOME, JRE_HOME and PATH set to launch the installer.
4. A Code source management client for the agent to retrieve tests sources code.
5. If GUI tests are in order, provide a way to keep active the graphical session. This may be a technical user account with the correct rights on Windows machines, or a X11 virtual frame buffer on GNU-Linux machines.

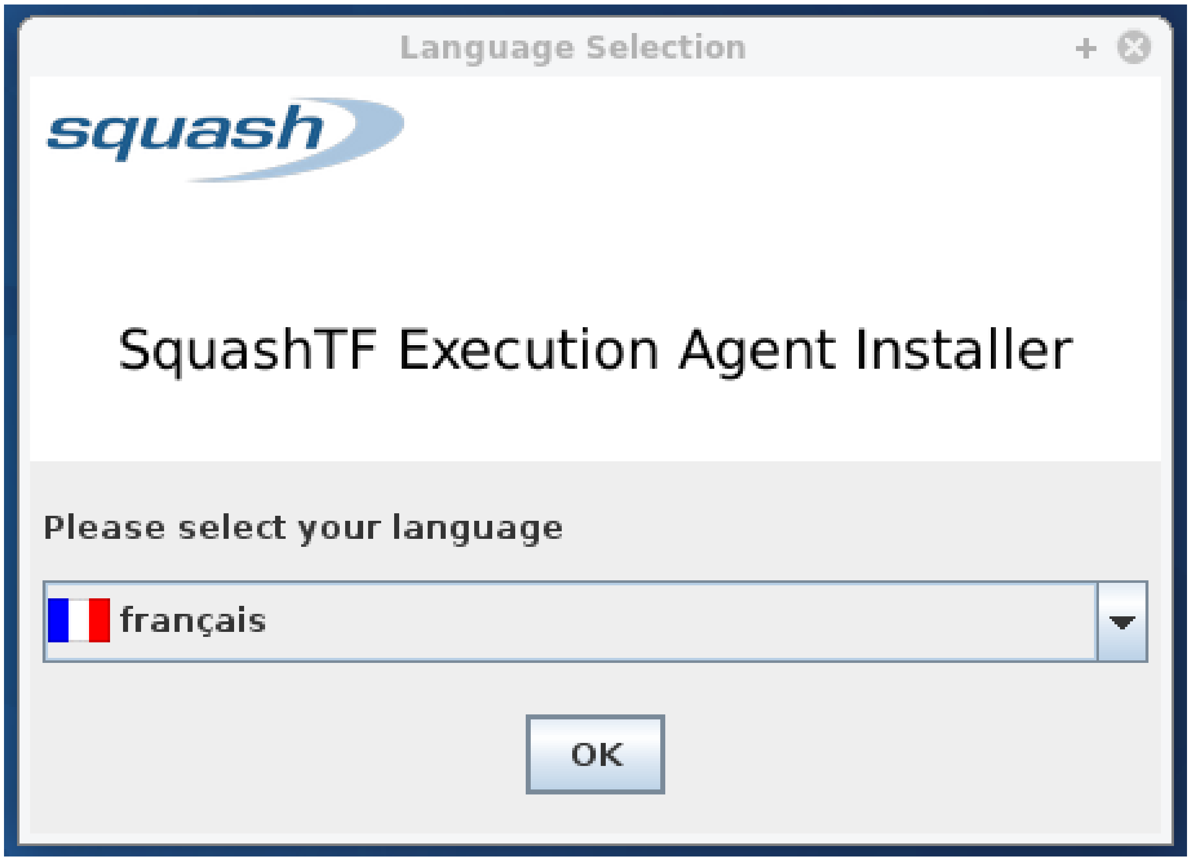
Procedure

Both Windows and GNU-Linux installers work the same. We display here below the common procedure :

1. The installers come as executable jar. So either double-click on the the jar or run the usual command :

   ```
   java -jar squash-tf-execution-agent-{version}-{os}.jar
   ```

   where {version} is the actual version of your execution server and {os} the operating system the agent will run on. One is then greeted by the following screen. Select the desired language and click on the OK button.

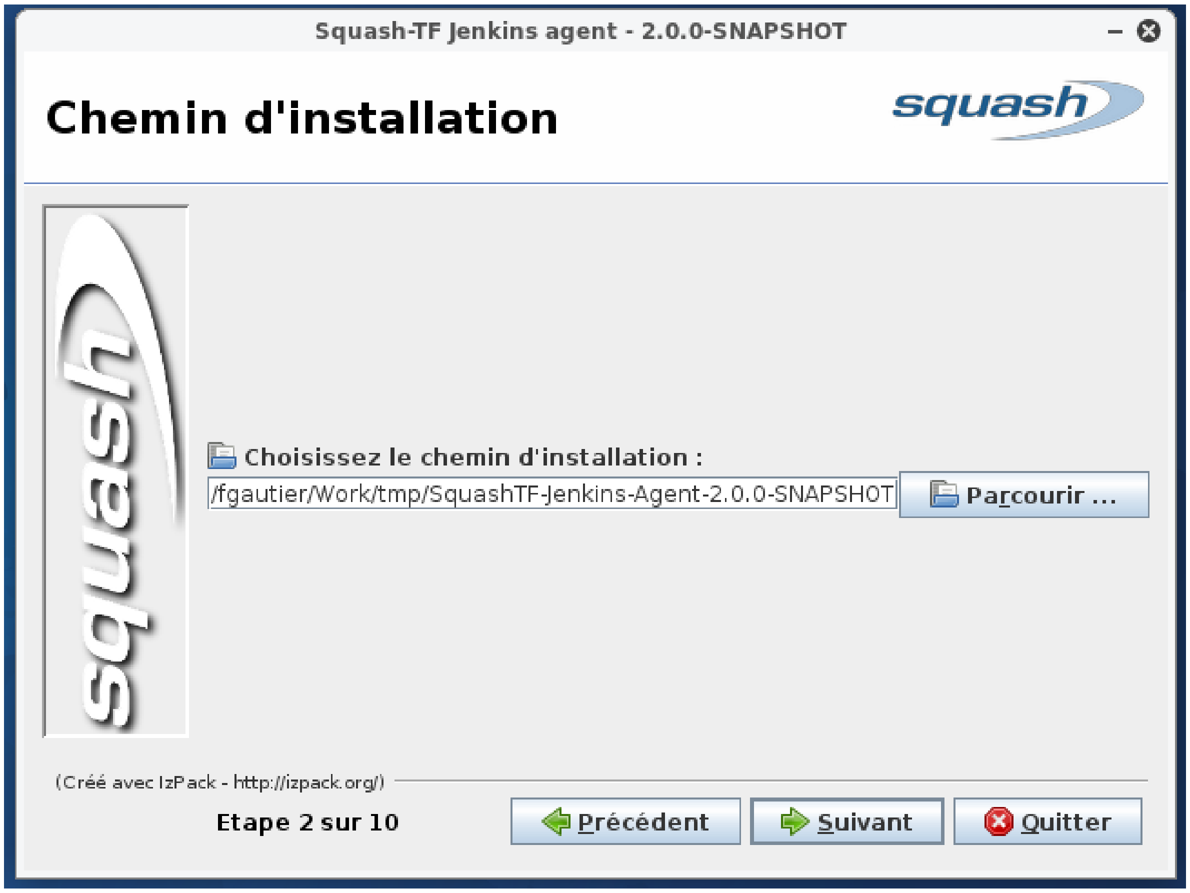
   

   SquashTF Execution Agent Installer

   Please select your language

   français

2. Fill in the installation directory and click on the next button.


Squash-TF Jenkins agent - 2.0.0-SNAPSHOT

Chemin d'installation

Choisissez le chemin d'installation :

/fgautier/Work/tmp/SquashTF-Jenkins-Agent-2.0.0-SNAPSHOT

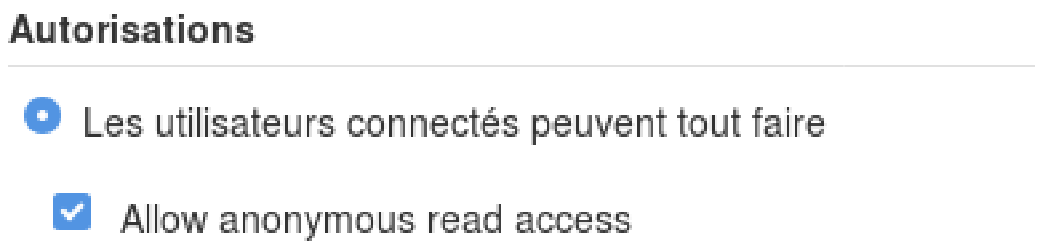
3. Go through various non-ambiguous steps until you're prompted to fill in the URL of your execution server.

   WARNING : Before going any further the Jenkins master should be configured so it allows "anonymous read access". Indeed at this step the installer will try to interact with the Jenkins API. If anonymous reading is not allowed, the installer will consider that the given URL is false. To configure it :

   a. Log in administrator mode into Jenkins.

   b. Navigate to "Administer Jenkins/Configure global security".

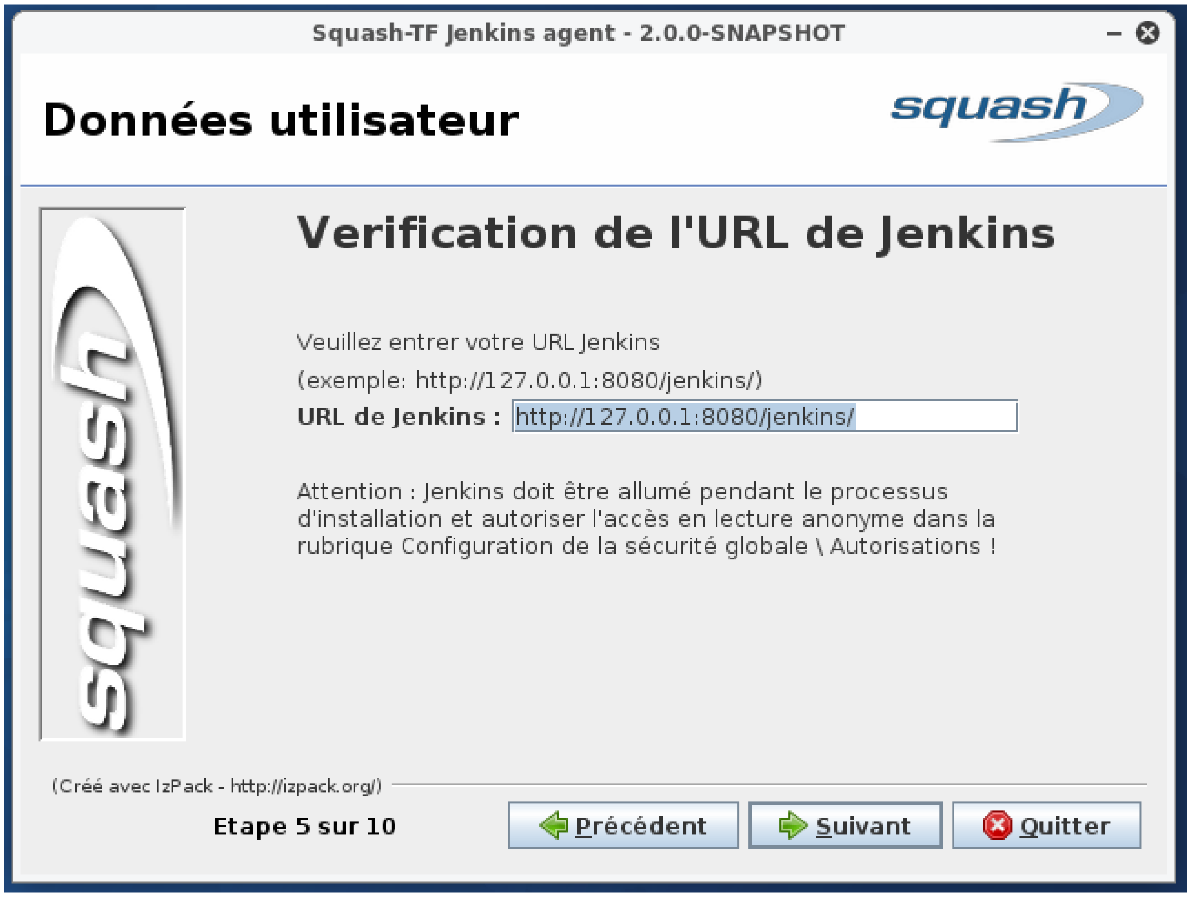
   c. In the "Autorisations" section, allow anonymous access reads.


Autorisations

● Les utilisateurs connectés peuvent tout faire

✓ Allow anonymous read access

Once this is done, come back to the installer screen and fill in the Jenkins URL.



Données utilisateur

Verification de l'URL de Jenkins

Veuillez entrer votre URL Jenkins

(exemple: http://127.0.0.1:8080/jenkins/)

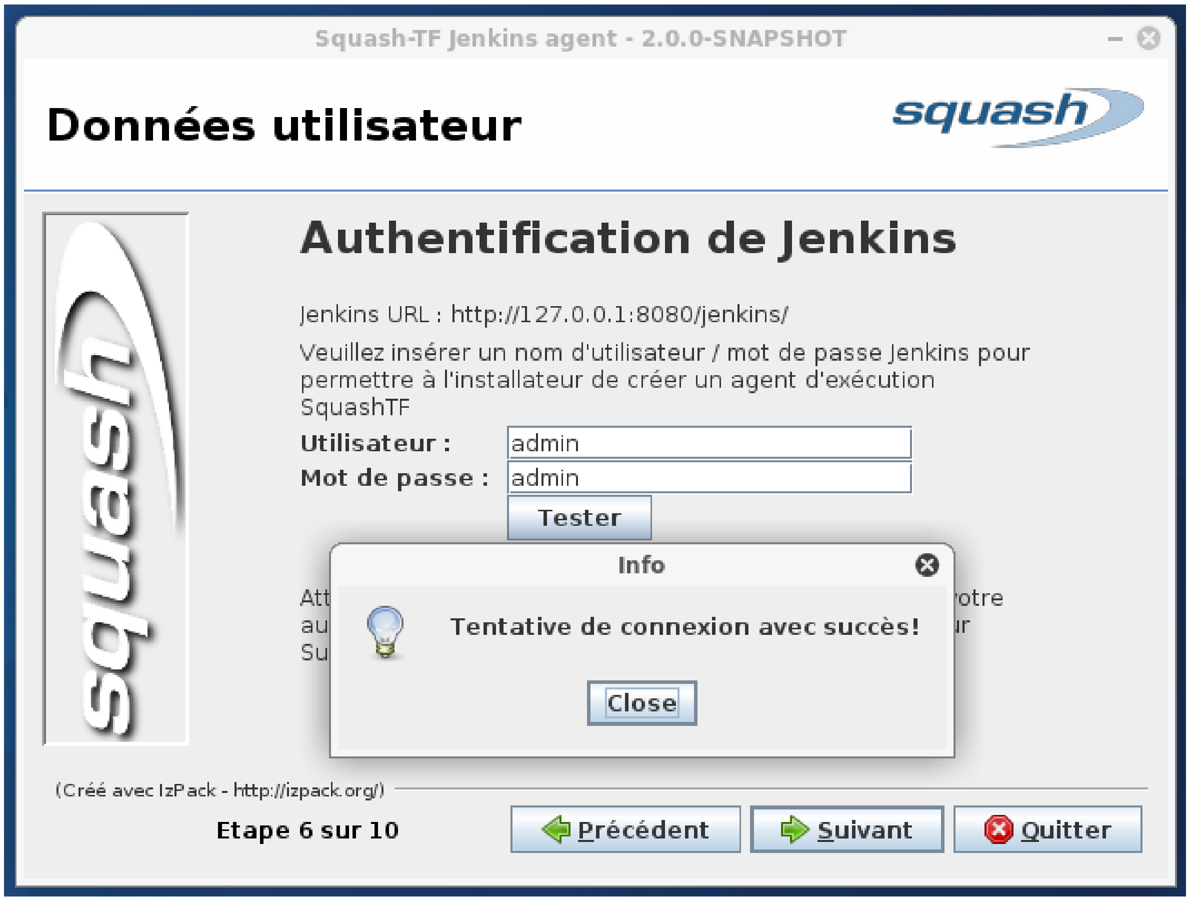
URL de Jenkins : http://127.0.0.1:8080/jenkins/

Attention : Jenkins doit être allumé pendant le processus d'installation et autoriser l'accès en lecture anonyme dans la rubrique Configuration de la sécurité globale \ Autorisations !

(Créé avec IzPack - http://izpack.org/)

Etape 5 sur 10

4. Fill in Jenkins administrator credentials, and verify the correcteness of the pair URL/credential by clicking the “Test” button.


Squash-TF Jenkins agent - 2.0.0-SNAPSHOT
squash
squash
Tester
Info
Close
(Créé avec IzPack - http://izpack.org/)
Précédent
Suivant
Quitter

Données utilisateur

Authentification de Jenkins

Jenkins URL : http://127.0.0.1:8080/jenkins/

Veuillez insérer un nom d'utilisateur / mot de passe Jenkins pour permettre à l'installateur de créer un agent d'exécution SquashTF

Utilisateur : admin

Mot de passe : admin

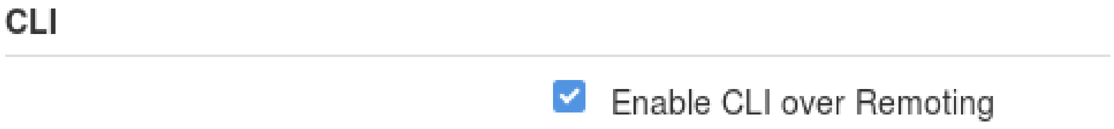
Tentative de connexion avec succès!

Etape 6 sur 10

5. Next the installer will perform the actual remote declaration of the node on the Jenkins master. Hence, the latter **must** be configured to accept remote control :

   a. Navigate to "Administer Jenkins/Configure global security"

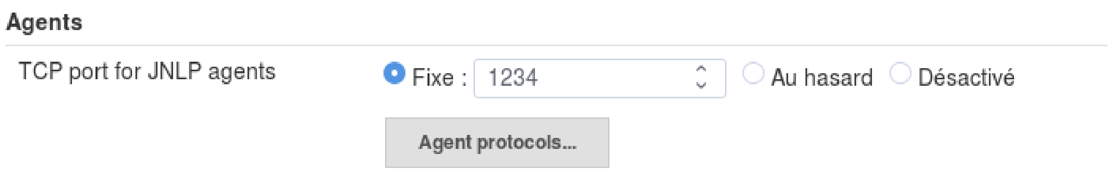
   b. Autorise (**temporarily**) remote access to Jenkins CLI.



CLI

✓ Enable CLI over Remoting

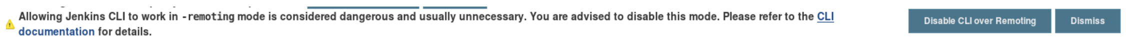
   c. Fill in the port onto which Jnlp agents can connect.



Agents

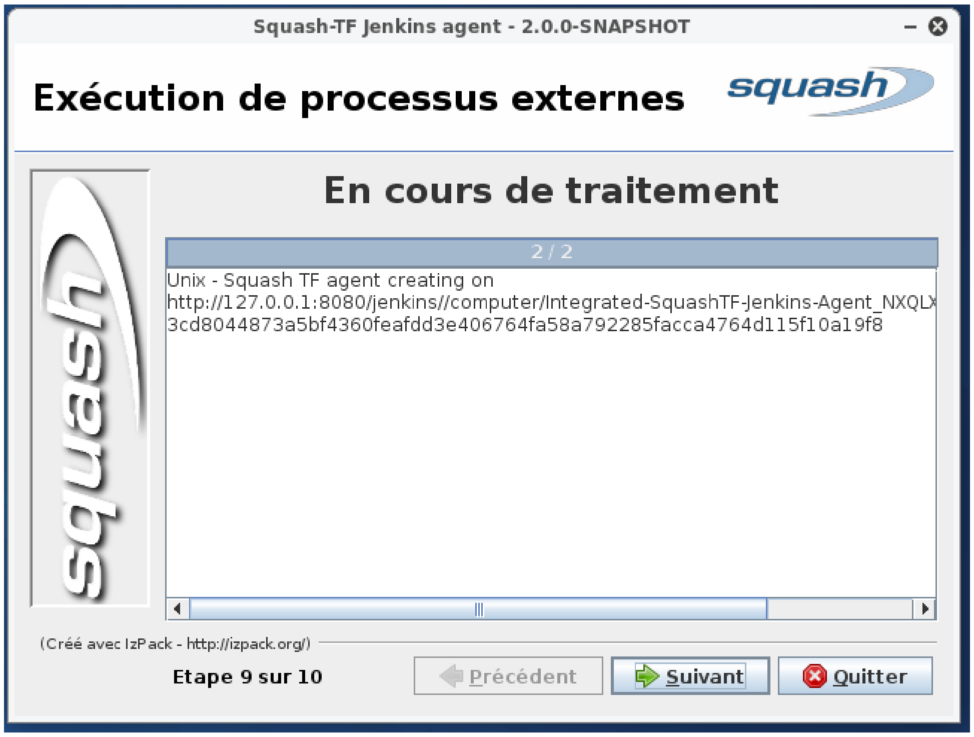
TCP port for JNLP agents [✓ Fixe : 1234] Au hasard Désactivé

   d. Click the "save" button. An alert message should then appear stating that allowing remote access to the CLI is not safe. That's normal. Just think to disable it after completion of this procedure.



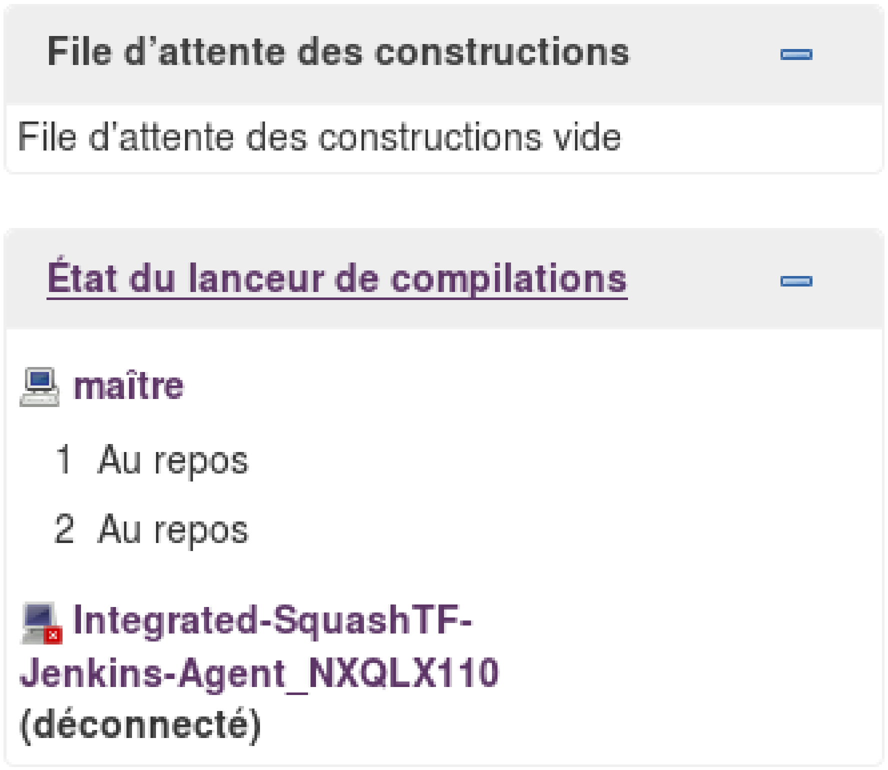
Allowing Jenkins CLI to work in `-remoting` mode is considered dangerous and usually unnecessary. You are advised to disable this mode. Please refer to the CLI documentation for details.

Once this is done, come back to the installer screen and click “next”. The following screen states that everything went well.


Squash-TF Jenkins agent - 2.0.0-SNAPSHOT
Exécution de processus externes
En cours de traitement
2 / 2
(Créé avec IzPack - http://izpack.org/)
Etape 9 sur 10
Précédent
Suivant
Quitter

```
Unix - Squash TF agent creating on
http://127.0.0.1:8080/jenkins//computer/Integrated-SquashTF-Jenkins-Agent_NXQLX
3cd8044873a5bf4360feafdd3e406764fa58a792285facca4764d115f10a19f8
```

6. Go through the following steps until installation is finished.
7. The newly created agent node is now visible in the left Jenkins menu. It appears as disconnected. This is perfectly normal. Indeed the slave has not been launched yet.



File d'attente des constructions

File d'attente des constructions vide

État du lanceur de compilations

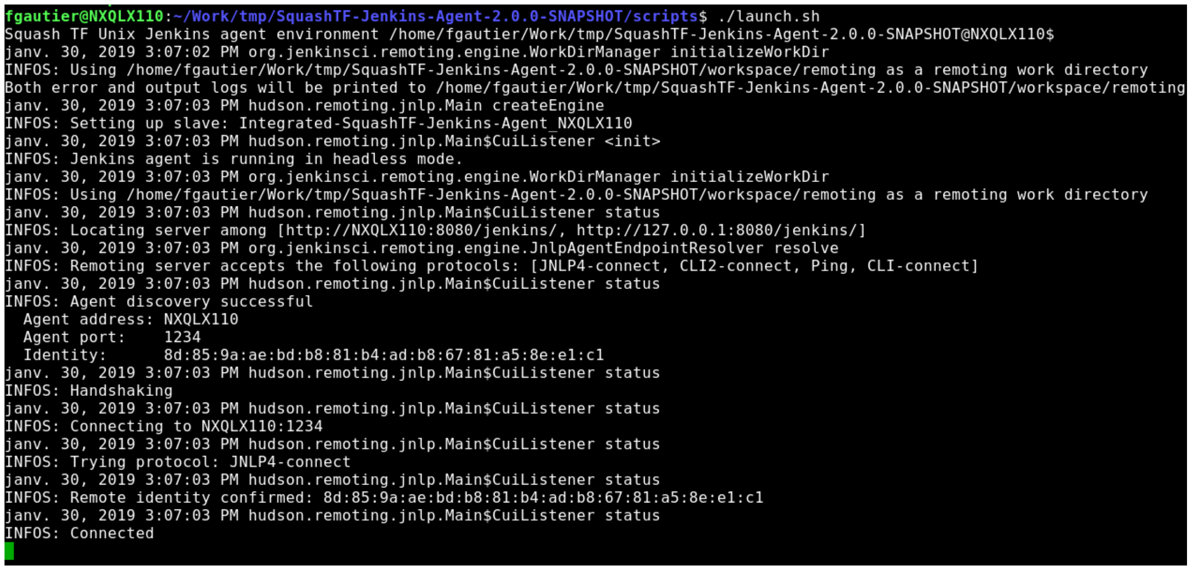
maître

1 Au repos

2 Au repos

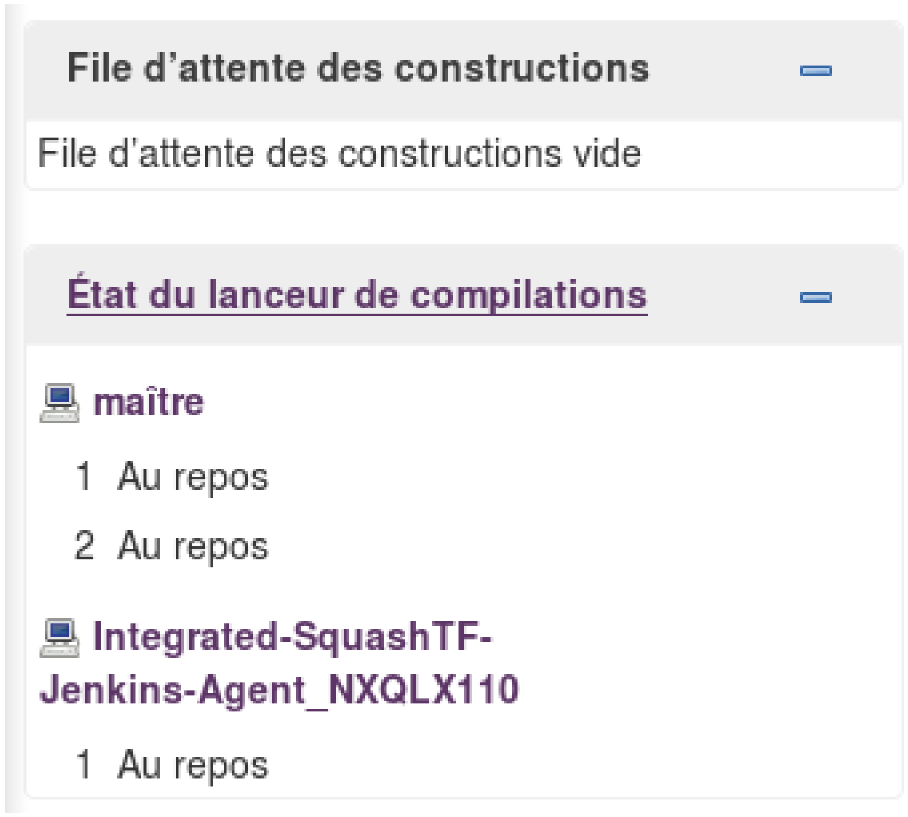
Integrated-SquashTF-Jenkins-Agent_NXQLX110 (déconnecté)

8. Launch the agent by executing either {install-dir}\scripts\lauch.cmd on Windows or {install-dir}/scripts/launch.sh on GNU-Linux, where {install-dir} is the directory one chose to install the agent. “INFOS: Connected” is then displayed on the console signaling that the agent is launched properly.



```
fgautier@NXQLX110:~/Work/tmp/SquashTF-Jenkins-Agent-2.0.0-SNAPSHOT/scripts$ ./launch.sh
Squash TF Unix Jenkins agent environment /home/fgautier/Work/tmp/SquashTF-Jenkins-Agent-2.0.0-SNAPSHOT@NXQLX110$
janv. 30, 2019 3:07:02 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFOS: Using /home/fgautier/Work/tmp/SquashTF-Jenkins-Agent-2.0.0-SNAPSHOT/workspace/remoting as a remoting work directory
Both error and output logs will be printed to /home/fgautier/Work/tmp/SquashTF-Jenkins-Agent-2.0.0-SNAPSHOT/workspace/remoting
janv. 30, 2019 3:07:03 PM hudson.remoting.jnlp.Main createEngine
INFOS: Setting up slave: Integrated-SquashTF-Jenkins-Agent_NXQLX110
janv. 30, 2019 3:07:03 PM hudson.remoting.jnlp.Main$CuiListener <init>
INFOS: Jenkins agent is running in headless mode.
janv. 30, 2019 3:07:03 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFOS: Using /home/fgautier/Work/tmp/SquashTF-Jenkins-Agent-2.0.0-SNAPSHOT/workspace/remoting as a remoting work directory
janv. 30, 2019 3:07:03 PM hudson.remoting.jnlp.Main$CuiListener status
INFOS: Locating server among [http://NXQLX110:8080/jenkins/, http://127.0.0.1:8080/jenkins/]
janv. 30, 2019 3:07:03 PM org.jenkinsci.remoting.engine.JnlpAgentEndpointResolver resolve
INFOS: Remoting server accepts the following protocols: [JNLP4-connect, CLI2-connect, Ping, CLI-connect]
janv. 30, 2019 3:07:03 PM hudson.remoting.jnlp.Main$CuiListener status
INFOS: Agent discovery successful
  Agent address: NXQLX110
  Agent port:    1234
  Identity:      8d:85:9a:ae:bd:b8:81:b4:ad:b8:67:81:a5:8e:e1:c1
janv. 30, 2019 3:07:03 PM hudson.remoting.jnlp.Main$CuiListener status
INFOS: Handshaking
janv. 30, 2019 3:07:03 PM hudson.remoting.jnlp.Main$CuiListener status
INFOS: Connecting to NXQLX110:1234
janv. 30, 2019 3:07:03 PM hudson.remoting.jnlp.Main$CuiListener status
INFOS: Trying protocol: JNLP4-connect
janv. 30, 2019 3:07:03 PM hudson.remoting.jnlp.Main$CuiListener status
INFOS: Remote identity confirmed: 8d:85:9a:ae:bd:b8:81:b4:ad:b8:67:81:a5:8e:e1:c1
janv. 30, 2019 3:07:03 PM hudson.remoting.jnlp.Main$CuiListener status
INFOS: Connected
```

On the Jenkins master the created node appears now as at “rest”.



File d'attente des constructions

File d'attente des constructions vide

État du lanceur de compilations

maître

1 Au repos

2 Au repos

Integrated-SquashTF-Jenkins-Agent_NXQLX110

1 Au repos

The procedure is reproducible for each agent node one wants to create. For security purposes, please ensure , once this is complete, to finally disable remote access to Jenkins CLI.

Docker setup

With version 2.1.0 and above, various Docker images of **Squash TF Execution Agents** are available.
Since the 2.2.0 version, our docker images are available from dockerhub : https://hub.docker.com/u/squashtest
However, our images are still available as tarball in our repo (download at squashtest.com).

Those come in three "flavors" :

- squash-tf-execution-agent : The base image.

  It brings the tools needed to be ran as a Jnlp agent and run headless **Squash TF** test. The image is built on the the open-jdk image and is thus Debian based. The image includes a JDK, a Maven, a Git client, a Mercurial client. And last but not least it contains the jars and configured entry-point to run it as a Jnlp agent.

- squash-tf-chrome-ui-execution-agent : Chrome-GUI testing specialized image.

  This specialization provides additionally a working graphical session and comes equipped with a guaranteed compatible Chrome/Chromedriver couple. The graphical stack is built using Xvfb as a fully in-memory X-11 server. Hence no graphical components should be needed on the underlying machine servicing the Docker host and running the images.

- squash-tf-firefox-ui-execution-agent : Firefox-GUI testing specialized image.

  Same as the chrome-ui image but comes with a guaranteed compatible Firefox/Geckodriver couple instead.

Credits : The docker images with in-memory X-server implementation is heavily based on work done by Stephen Fox.

Pre-requisites

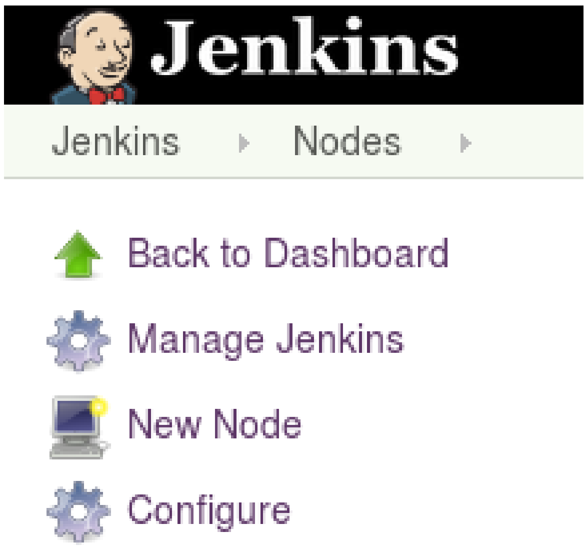
The following pre-requisites are needed in order to perform the installation :

- A Jnlp open port on the Jenkins Master (Squash TF Execution Server) that can be used by the agent to connect to the master.
- A Docker compatible operating system on the machine(s) where the images will be running.
- A valid Docker installation with a Docker host.

Procedure

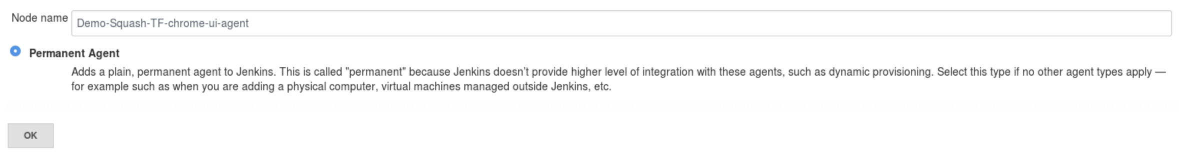
The setup procedure is flavor independent[1]. For the sake of example, screenshots displayed below are those from a **Squash TF Chrome-ui agent** setup.

1. The first step is to declare a new node in Jenkins administration section. Go to "Manage Jenkins/Manage Nodes" and click on "New Node" on the left panel.



2. Fill in wanted agent name and select Permanent agent radio button.

[1] No evident link has been found with Quantum Chromodynamics. Yet, we let the demonstration of the invariance under SU(3) transformations as an exercise to the reader.



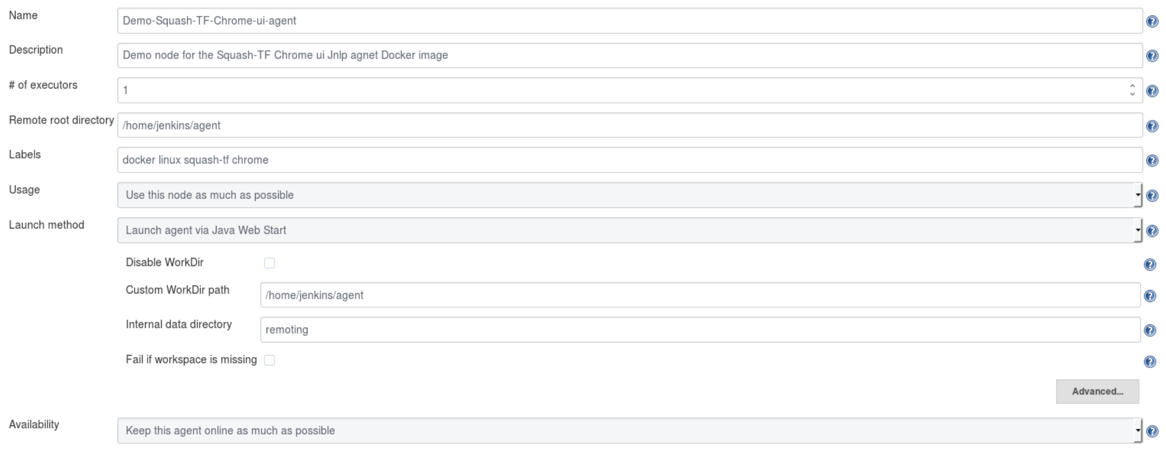
Node name: Demo-Squash-TF-chrome-ui-agent

● **Permanent Agent**
Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

OK

3. The images are built to use /home/jenkins/agent as the working directory. Thus fill in distant working dir section and Remote root directory with "/home/jenkins/agent".

4. Add as many labels as you wish and a description if wanted. As the Jenkins documentation states : *Labels (or tags) are used to group multiple agents into one logical group (..) Labels do not necessarily have to represent the operating system on the agent; you can also use labels to note the CPU architecture, or that a certain tool is installed on the agent.*

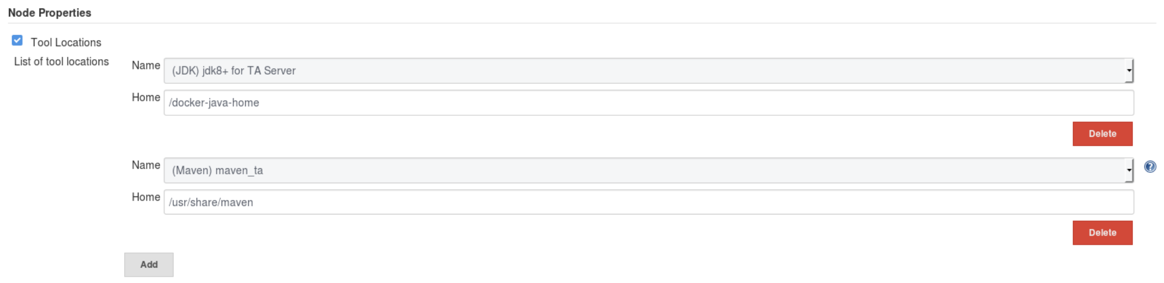


Name	Demo-Squash-TF-Chrome-ui-agent
Description	Demo node for the Squash-TF Chrome ui Jnlp agnet Docker image
# of executors	1
Remote root directory	/home/jenkins/agent
Labels	docker linux squash-tf chrome
Usage	Use this node as much as possible
Launch method	Launch agent via Java Web Start
Disable WorkDir	☐
Custom WorkDir path	/home/jenkins/agent
Internal data directory	remoting
Fail if workspace is missing	☐
Availability	Keep this agent online as much as possible

5. The agent may not share with the execution server vital – for **Squash TF** tests executions – tools locations. The aforementioned locations can be overridden using the dedicated section and the following values :

   a. JDK home : /docker-java-home

   b. Maven home: /usr/share/maven



Node Properties

☑ Tool Locations
List of tool locations

Name	(JDK) jdk8+ for TA Server
Home	/docker-java-home
Name	(Maven) maven_ta
Home	/usr/share/maven

To further clarify tools locations Mercurial and Git are to be found at :

   c. Mercurial : /usr/bin/hg

   d. Git : /usr/bin/git

6. The agent may not share with the execution server vital – for **Squash TF** tests executions – environment variables. Yet, the aforementioned variables are already set in the dockerfile. Hence, there is no need to redefine them in the concerned section. Nonetheless, for information purposes, here are the values of a few of these variables :

   a. JENKINS_HOME : /home/jenkins/agent

   b. SQUASH_TA_HOME : /home/jenkins/agents

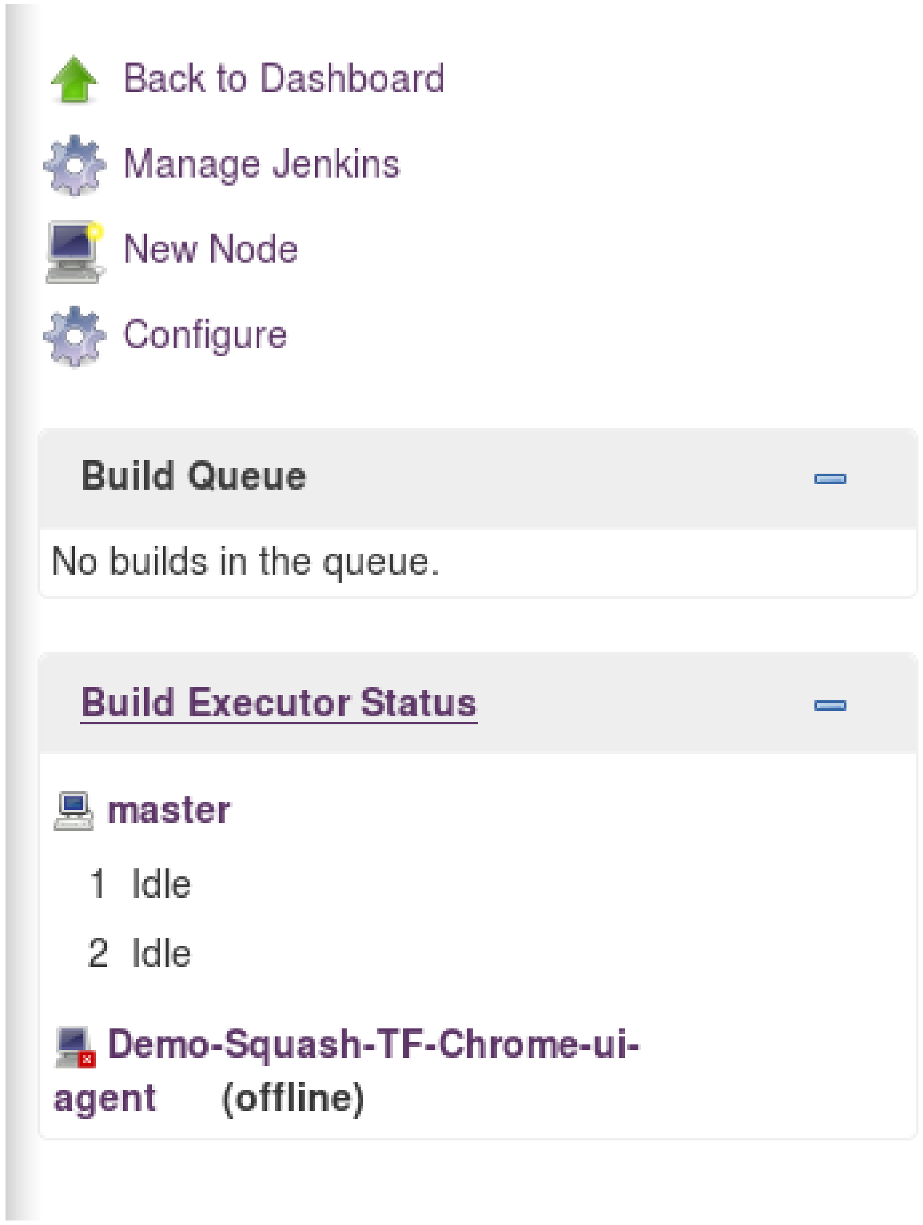
   c. MAVEN_HOME : /usr/share/maven

   d. MAVEN_CONFIG : /home/jenkins/.m2

   e. JAVA_HOME : /docker-java-home

---

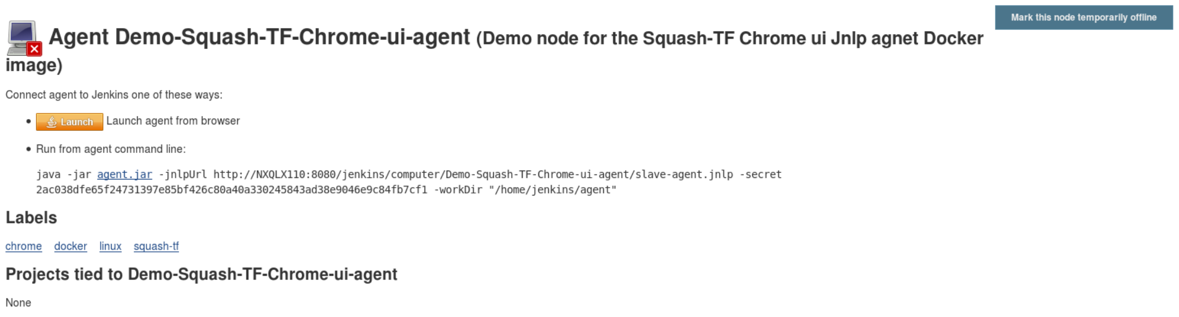
7. Click on the "save" button and note the secret of your newly declared node. The newly declared node should now appear as offline in the left panel.



No builds in the queue.

Demo-Squash-TF-Chrome-ui-agent (offline)

Click on the newly created node to retrieve its "secret" phrase. It will prove necessary when launching the agent.



Agent Demo-Squash-TF-Chrome-ui-agent (Demo node for the Squash-TF Chrome ui Jnlp agnet Docker image)

Connect agent to Jenkins one of these ways:

- Launch agent from browser
- Run from agent command line:

```
java -jar agent.jar -jnlpUrl http://NXQLX110:8080/jenkins/computer/Demo-Squash-TF-Chrome-ui-agent/slave-agent.jnlp -secret
2ac038dfe65f24731397e85bf426c80a40a330245843ad38e9046e9c84fb7cf1 -workDir "/home/jenkins/agent"
```

Labels

chrome docker linux squash-tf

Projects tied to Demo-Squash-TF-Chrome-ui-agent

None

8. You've finished the node declaration. Now you now have to download the agent docker image. Two possibilities :
   - using a docker images from dockerhub.
   - using a docker images package as tarball from our artifacts repository.

>> Download the agent image from dockerhub

To donwload the desired image :

```
docker pull squashtest/squash-tf-{flavor}-execution-agent.docker.{version}
```

where {flavor} corresponds to the type of agent you're interested in and {version} is the version of the agent. We recommend you to choose the **version** corresponding to your **Squash TF execution server** for full compatibility. For the sake of information, the Jnlp agent jars used to build the base image are the one provided by Jenkins itself.

>> Download the agent image from our artifacts repository

a. Download the desired image here. Alternatively one can access the image directly from our repository using :

```
wget http://repo.squashtest.org/distribution/squash-tf-{flavor}-execution-
↪agent.docker.{version}.tar
```

where {flavor} corresponds to the type of agent you're interested in and {version} is the version of the agent. We recommend you to choose the **version** corresponding to your **Squash TF execution server** for full compatibility. For the sake of information, the Jnlp agent jars used to build the base image are the one provided by Jenkins itself.

b. Load the downloaded docker image in your set up using the following command :

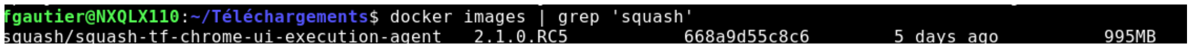
```
docker load -i squash-tf-{flavor}-execution-agent.docker.{version}.tar
```

where {flavor} corresponds to the type of agent you're interested in and {version} is the version of the agent.

It is to be noted that one should "**NOT**" use the docker import command as it will flatten all layers and will render unusable the image.

The image should now be visible in your available docker images. This can be checked using :

```
docker images
```



```
fgautier@NXQLX110:~/Téléchargements$ docker images | grep 'squash'
squash/squash-tf-chrome-ui-execution-agent    2.1.0.RC5            668a9d55c8c6          5 days ago            995MB
```

9. Finally run the docker image using :

>> For the image from dockerhub

```
docker run --name demo-tf-agent --user jenkins --env "JENKINS_AGENT_NAME={agent_
↪name}" --env "JENKINS_SECRET={secret}" squashtest/squash-tf-{flavor}-execution-
↪agent:{version} -url http://{jenkins_url}
```

The command should be understood as following :

a. docker run : Runs a docker container.

b. –name demo-tf-agent : The nickname of the container that will be created. Can be chosen to one's liking. If not set a random name will be assigned to the container.

c. –user jenkins : The technical user with proper rights to run Jenkins.

d. –env "JENKINS_AGENT_NAME={agent_name}" : Enables one to specify the name of the declared node in Jenkins. Modify {agent_name} accordingly.

e. –env "JENKINS_SECRET={secret}" : Enables one to specify the secret of the declared node in Jenkins. Modify {secret} accordingly.

f. squashtest/squash-tf-{flavor}-execution-agent:{version} : Selects the image one wants to run, where {flavor} is the type of image one is interested in and {version} the version targeted.

g. -url http://{jenkins_url} : Enables one to specify the URL of the Jenkins master server.

>> For the image from our artifact repository

```
docker run --name demo-tf-agent --user jenkins --env "JENKINS_AGENT_NAME={agent_
↪name}" --env "JENKINS_SECRET={secret}" squash/squash-tf-{flavor}-execution-
↪agent:{version} -url http://{jenkins_url}
```

The command should be understood as following :

a. docker run : Runs a docker container.

b. –name demo-tf-agent : The nickname of the container that will be created. Can be chosen to one's liking. If not set a random name will be assigned to the container.

c. –user jenkins : The technical user with proper rights to run Jenkins.

d. –env "JENKINS_AGENT_NAME={agent_name}" : Enables one to specify the name of the declared node in Jenkins. Modify {agent_name} accordingly.

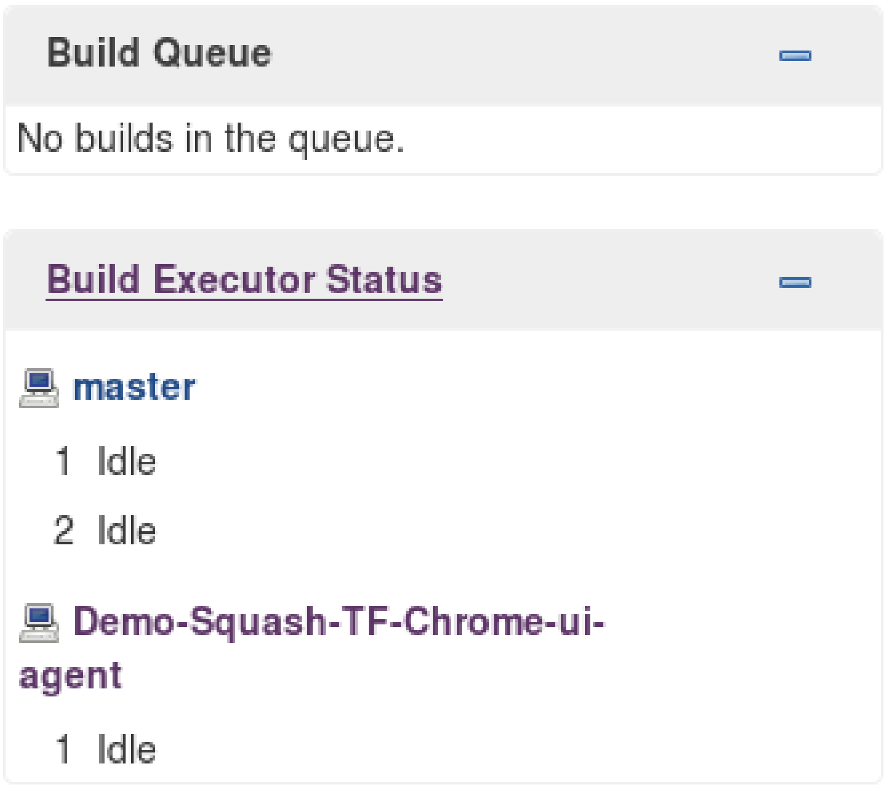
e. –env "JENKINS_SECRET={secret}" : Enables one to specify the secret of the declared node in Jenkins. Modify {secret} accordingly.

f. squash/squash-tf-{flavor}-execution-agent:{version} : Selects the image one wants to run, where {flavor} is the type of image one is interested in and {version} the version targeted.

g. -url http://{jenkins_url} : Enables one to specify the URL of the Jenkins master server.

Warning: The difference between the 2 run command is in the image name. The dockerhub one is **squashtest**/squash-tf-{flavor}-execution-agent:{version}, the other is **squash**/squash-tf-{flavor}-execution-agent:{version}

Once done, the node should appear as online on the Jenkins interface and its build queue should be in "idle".



Build Queue

No builds in the queue.

Build Executor Status

master

1 Idle

2 Idle

Demo-Squash-TF-Chrome-ui-agent

1 Idle

The agent is now ready to execute its first **Squash TF** tests.

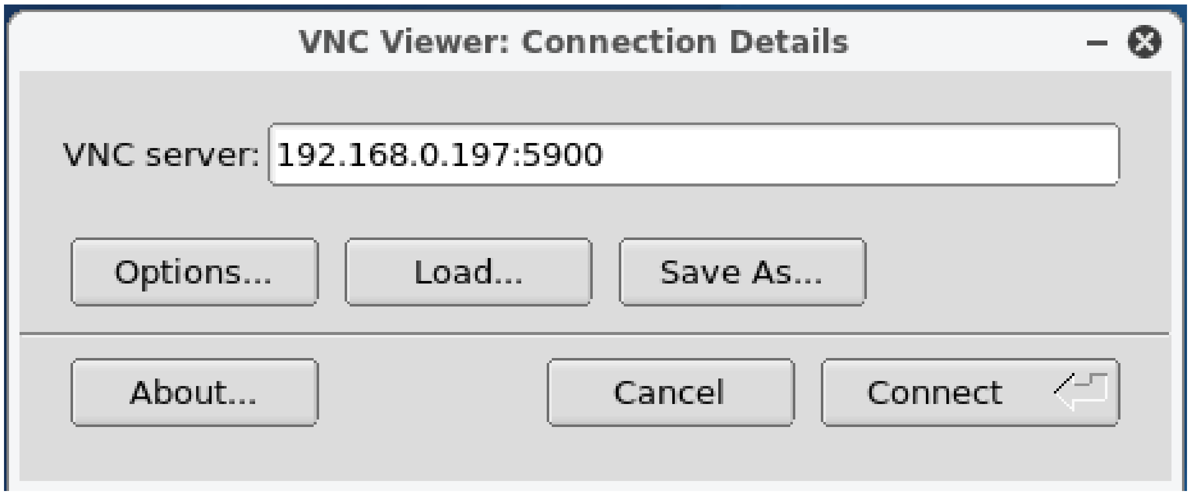
Particularity of graphical session providing docker images

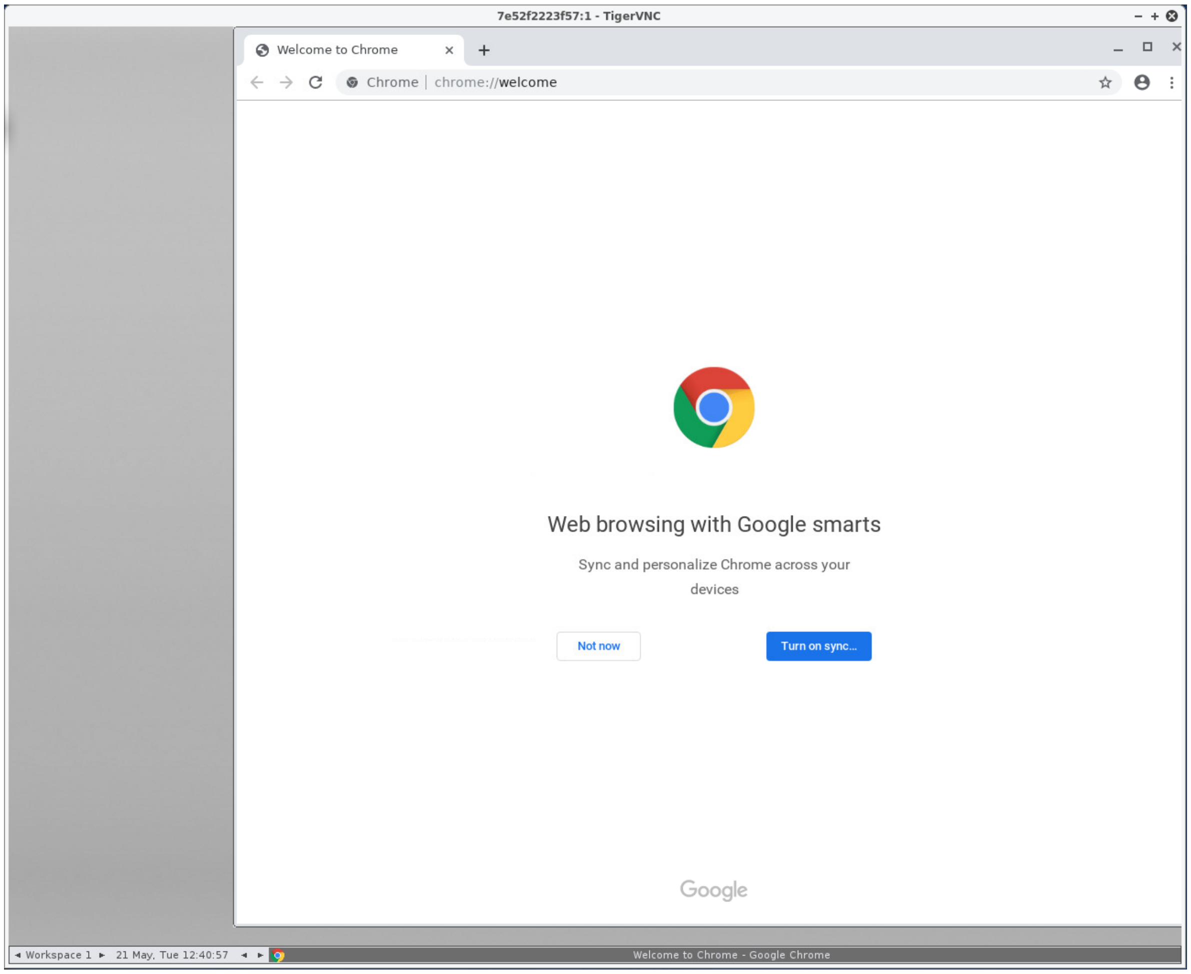
The docker images with X11 have some particularities :

1. The latter images can be ran in "debug mode" using the –x11 vnc-debug option. In such a mode a VNC (Virtual Network Computing) server is also launched allowing remote access to the graphical session of the running container. If launched the server listen on port 5900. One should thus bind the container 5900 port to a physical port of the machine hosting Docker. The full command is then :

```
docker run -p 5900:5900 --name demo-tf-agent --user jenkins --env "JENKINS_AGENT_
↪NAME={agent_name}" --env "JENKINS_SECRET={secret}" squash/squash-tf-{flavor}-
↪execution-agent:{version} -url http://{jenkins_url} --x11-vnc-debug
```

Once the container launched in debug mode, its graphical session can be accessed using a VNC client targeting the Docker hosting machine on the binded port. For example using vncviewer :


VNC Viewer: Connection Details
VNC server: 192.168.0.197:5900
Options...
Load...
Save As...
About...
Cancel
Connect


7e52f2223f57:1 - TigerVNC
Welcome to Chrome
Chrome | chrome://welcome
Web browsing with Google smarts
Sync and personalize Chrome across your devices
Not now
Turn on sync...
Google
Workspace 1
21 May, Tue 12:40:57
Welcome to Chrome - Google Chrome

2. Due to Chrome own limitations the user running it shoud be “Privileged”. This can be set using the –privileged options in the docker command. The command to run the Chrome flavor of the agent is then :

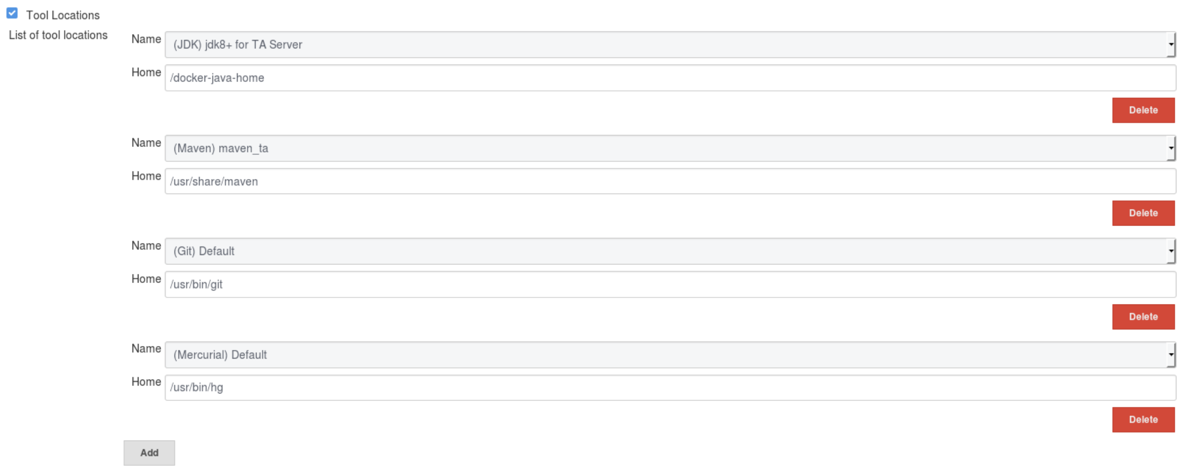
```
docker run --name demo-tf-agent --user jenkins --privileged --env "JENKINS_AGENT_
↪NAME={agent_name}" --env "JENKINS_SECRET={secret}" squash/squash-tf-chrome-ui-
↪execution-agent:{version} -url http://{jenkins_url}
```

Both of these options can be run simultaneously.

Troubleshooting

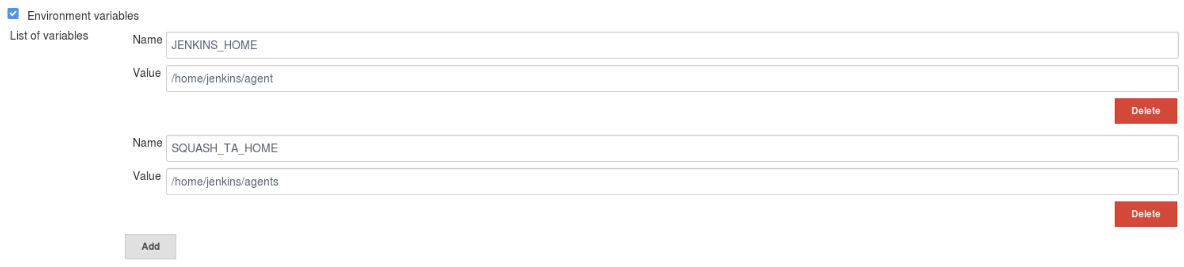
Here is a non exhaustive list of possible slight configuration/integration issues.

1. Problems linked to tools : Java, Maven, git, mercurial is not found, problems while checking out project, etc… Jenkins master tools configuration may be interfering with the agent one. Try overriding their location using the "tools location" section in the node configuration page on Jenkins.



Name	(JDK) jdk8+ for TA Server
Home	/docker-java-home
Name	(Maven) maven_ta
Home	/usr/share/maven
Name	(Git) Default
Home	/usr/bin/git
Name	(Mercurial) Default
Home	/usr/bin/hg

2. Something is wrong with the environment, cannot find any parasable pom, working directory is weirdly set. Jenkins master environment variables configuration may be interfering with the agent one. Try overriding their values using the "environment variables" section in the node configuration page on Jenkins.



Name	JENKINS_HOME
Value	/home/jenkins/agent
Name	SQUASH_TA_HOME
Value	/home/jenkins/agents

3. Jenkins is not reachable since it is behind a firewall. Use the Jenkins anticipated mechanism and specify the environment variable –env "JENKINS_TUNNEL : HOST:PORT" for a tunnel to route TCP traffic to Jenkins host, when jenkins can't be directly accessed over network

4. Our own tests revealed that too many layers of virtualization may render the use of the images unstable. For instance we found that on our setup, trying to run the images on the "virtualization sandwich" Windows 10/Virtualbox/Debian 9/Docker results in an immediate crash of the container without any further ado.

5. Time problems in reports. By default our containers are in UTC. This could lead to unexpected time value in the produced reports. You could solve this problem by specifying a timezone in your docker run command line. Example :

```
--env "TZ=Europe/Paris"
```

6. When you execute a test with a job using the Chrome agent which fails when it tries to open Chrome, and you have an error stack which looks like :

```
org.squashtest.ta.framework.exception.InstructionRuntimeException: Junit test␣
↪execution for [engine:junit-jupiter]/[class:XXX]/[method:YYY] failed on error :␣
↪unknown error: Chrome failed to start: crashed
(unknown error: DevToolsActivePort file doesn't exist)
(The process started from chrome location /usr/bin/google-chrome is no longer␣
↪running, so ChromeDriver is assuming that Chrome has crashed.)
```

Then have you run your container in privileged mode as explain in the doc ? See : *Particularity of graphical session providing docker images*

1.2.2 Execution Agent - Update

Overview

Using a master-agent architecture to run your jobs in Jenkins can be done in a few combinations : depending on the platform on which your Execution Server runs (Windows, Linux, docker container) and on which the Execution Agents you wish to update will be, the procedure can be slightly different.

As such this guide will have a two parts structure :

- Tranferring nodes directory
- Jenkins configuration update

Each step in these parts will specify if it concerns the server or the agent, and each type of install (Windows, Linux, docker container) will be explained if there are differences.

Prerequisites

A **Squash TF Execution Server** up to date with the version you will update your agent to.

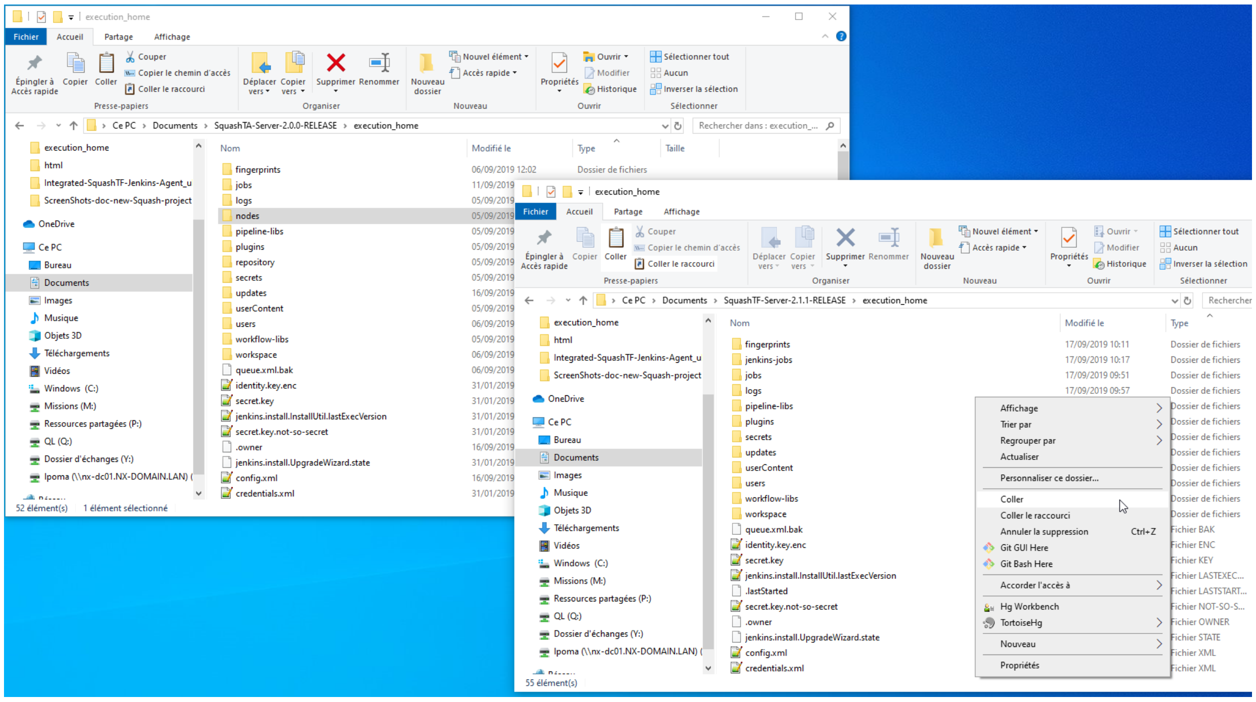
Transferring nodes directory

Tip: If you followed the Execution Server update process (physical or docker) you may have already transferred the **nodes** directory from the old Execution Server to the new one.

In that case you can skip ahead to the 4th step.

1. Stop your current **Squash TF Execution Server** and the **Jenkins Agent** you wish to update :
   - Using your method of choice in a physical install of the Execution Server (Linux or Windows).
   - Using the **docker stop** command in case your Execution Server is running in a docker container.

2a. On a physical install of the Execution Server simply copy the **nodes** directory from **{current_server_path}/execution_home/** to **{new_server_path}/execution_home/**



2b. On a docker install of the Execution Server use the **docker cp** command to copy the **nodes** directory from your old container to the host system, then from the host system to the new container.

```
docker cp {old-container}:/opt/squash-ta-server/execution_home/nodes/. {path_
↪of_choice}/nodes-backup
docker cp {path_of_choice}/nodes-backup/. {new-container}:/opt/squash-ta-
↪server/execution_home/nodes
```

---

3. Restart your new Execution Server :
   - Using your method of choice in a physical install of the Execution Server (Linux or Windows).
   - Using the **docker start** command in case your Execution Server is running in a docker container.

---

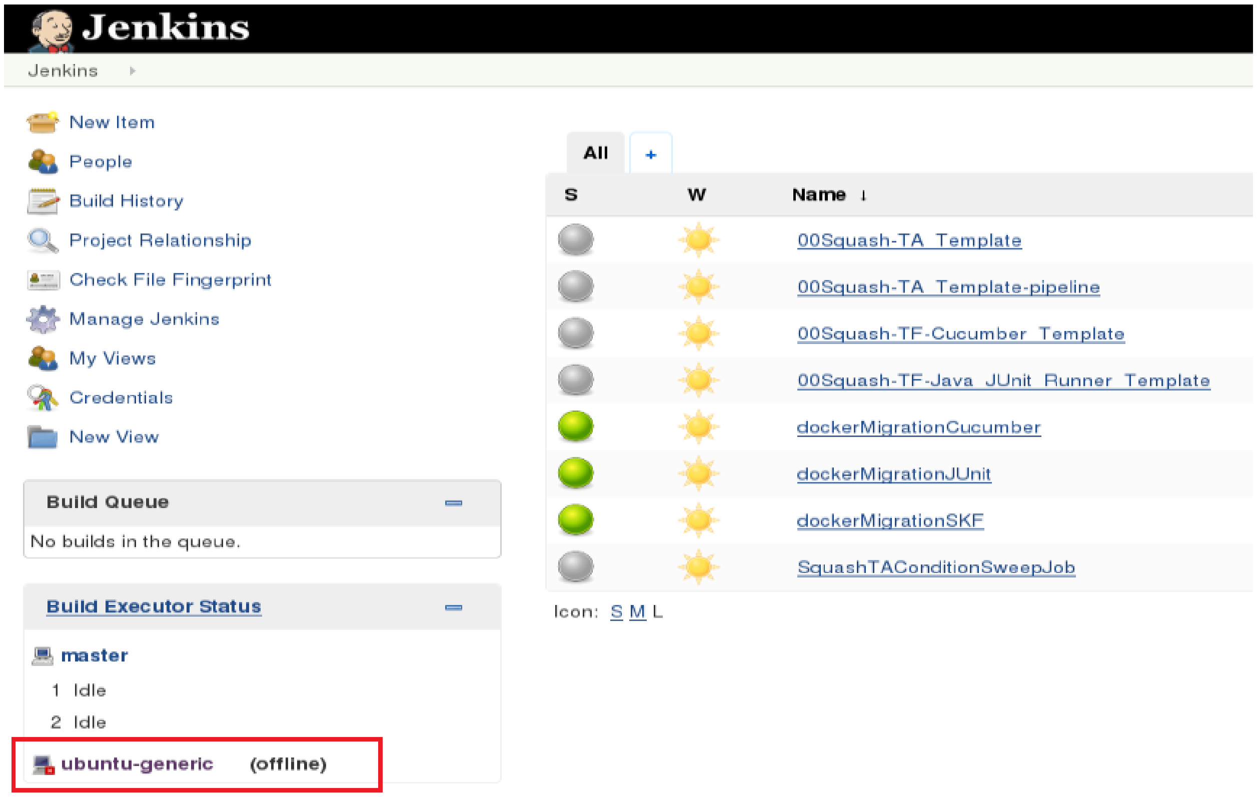
4. Choose and download the version of the **Squash TF Execution Agent** you wish to install on this page.

5a. If you downloaded a physical installer of the agent :

Launch the installer of your new **Squash TF Execution Agent** and follow the *setup procedure*.

5b. If you chose to use our docker image :

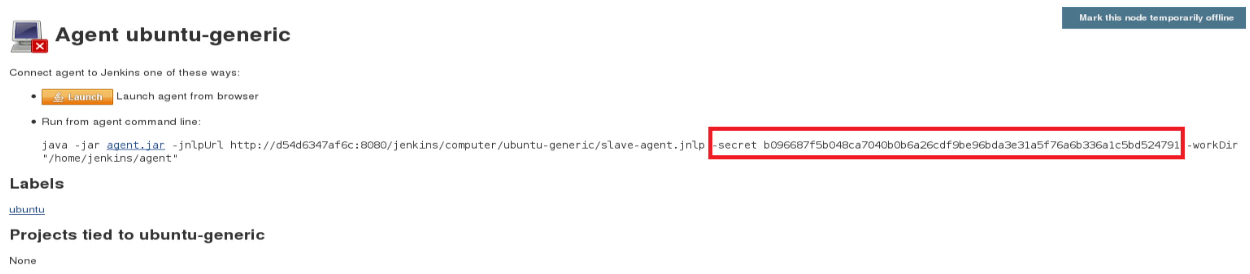
- With the new **Execution Server** running connect to Jenkins as an administrator and click on the name of your old agent on the left of the screen.
- Recover the **secret** of the node and keep it for later.



No builds in the queue.

master
1 Idle
2 Idle
ubuntu-generic (offline)

S	W	Name ↓
●	☀	00Squash-TA_Template
●	☀	00Squash-TA_Template-pipeline
●	☀	00Squash-TF-Cucumber_Template
●	☀	00Squash-TF-Java_JUnit_Runner_Template
●	☀	dockerMigrationCucumber
●	☀	dockerMigrationJUnit
●	☀	dockerMigrationSKF
●	☀	SquashTAConditionSweepJob



Mark this node temporarily offline

Agent ubuntu-generic

Connect agent to Jenkins one of these ways:

- Launch Launch agent from browser
- Run from agent command line:

```
java -jar agent.jar -jnlpUrl http://d54d6347af6c:8080/jenkins/computer/ubuntu-generic/slave-agent.jnlp -secret b096687f5b048ca7040b0b6a26cdf9be96bda3e31a5f76a6b336a1c5bd524791 -workDir
"/home/jenkins/agent"
```

Labels

ubuntu

Projects tied to ubuntu-generic

None

Jenkins node update

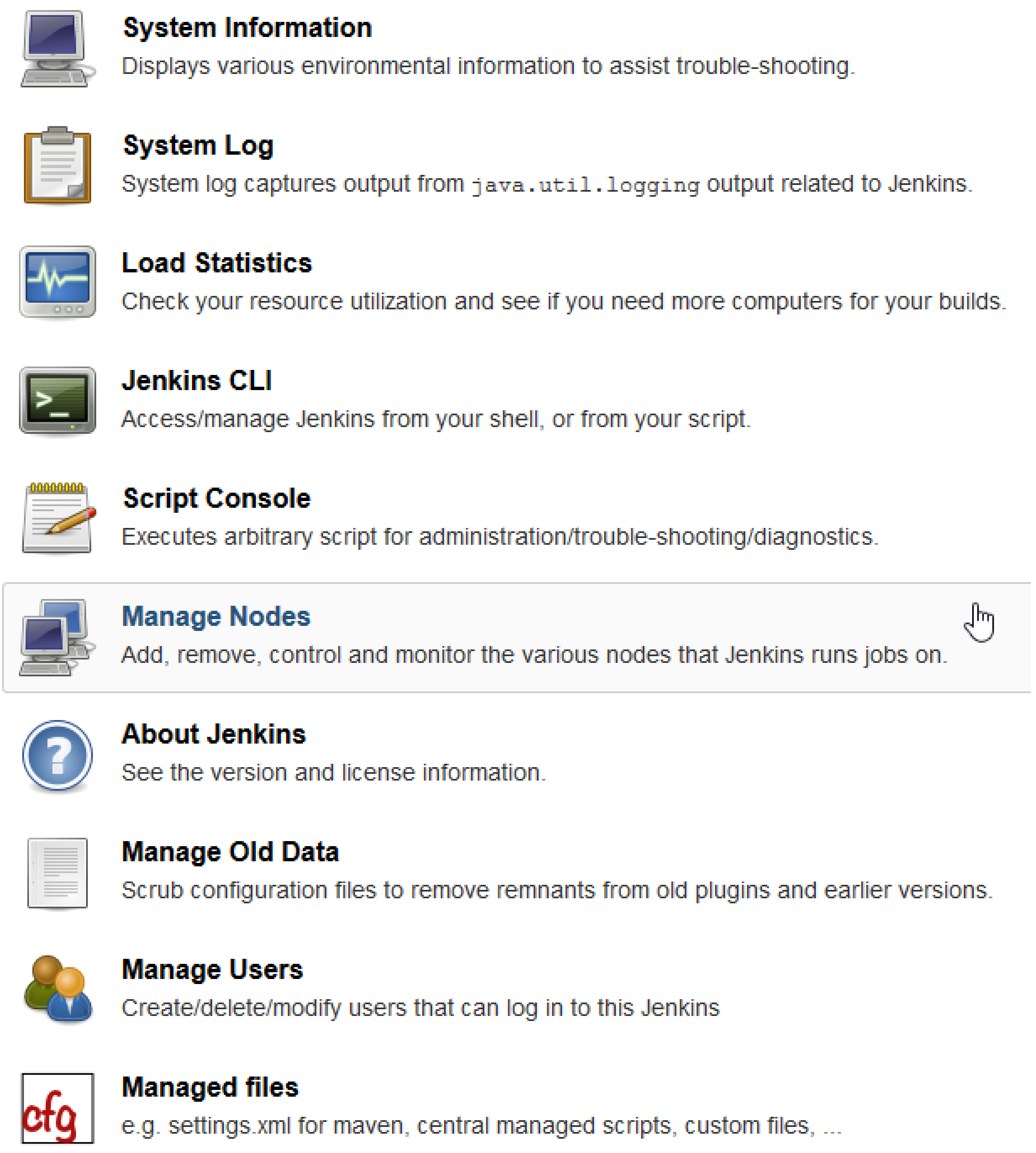
For this section the new Execution Server should be running, either on your machine (Windows or Linux) or in a docker container.

You should also be connected to the corresponding Jenkins as an administrator.

The rest of the procedure will distinguish between a *physically installed* Agent and a an Agent in a *docker container*. The type of install of the **Execution server** is no longer important in this part.

Physical installer

1. Go to `Manage Jenkins` then to `Manage Nodes`.



System Information
Displays various environmental information to assist trouble-shooting.

System Log
System log captures output from `java.util.logging` output related to Jenkins.

Load Statistics
Check your resource utilization and see if you need more computers for your builds.

Jenkins CLI
Access/manage Jenkins from your shell, or from your script.

Script Console
Executes arbitrary script for administration/trouble-shooting/diagnostics.

Manage Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

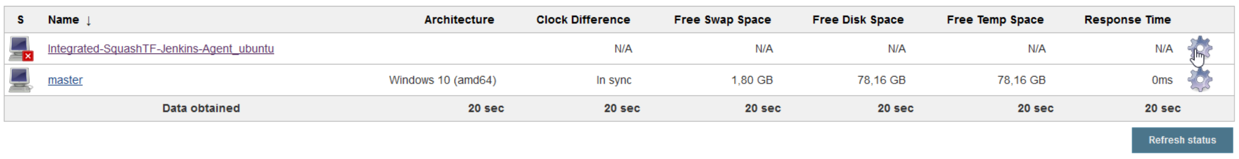
About Jenkins
See the version and license information.

Manage Old Data
Scrub configuration files to remove remnants from old plugins and earlier versions.

Manage Users
Create/delete/modify users that can log in to this Jenkins

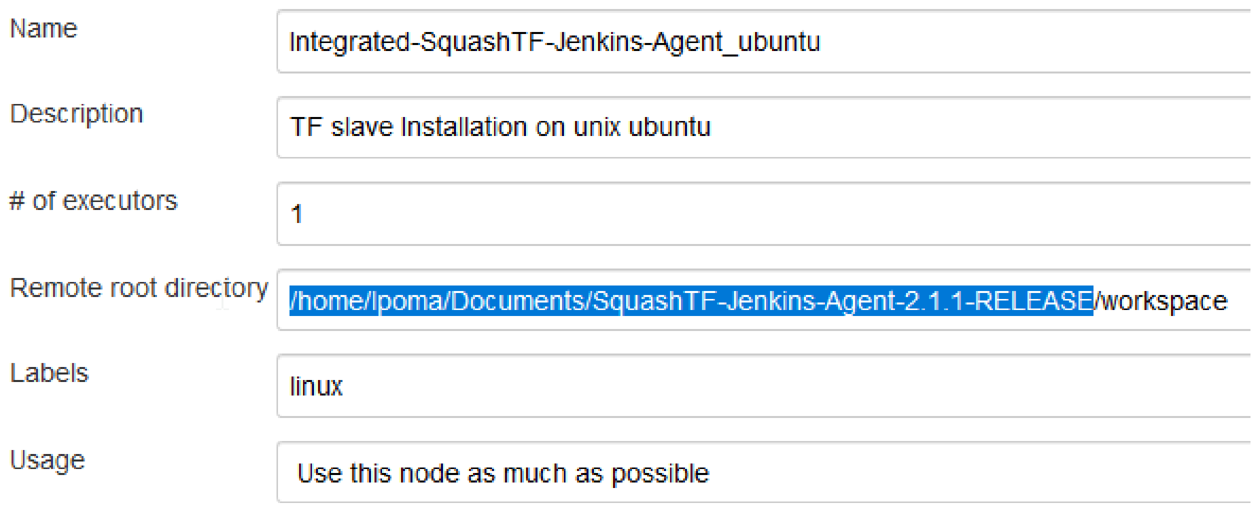
Managed files
e.g. settings.xml for maven, central managed scripts, custom files, ...

2. Click on the **cogwheel** icon on the right of the agent of the agent your are updating to access its configuration screen.



S	Name ↓	Architecture	Clock Difference	Free Swap Space	Free Disk Space	Free Temp Space	Response Time	
	Integrated-SquashTF-Jenkins-Agent_ubuntu		N/A	N/A	N/A	N/A	N/A	
	master	Windows 10 (amd64)	In sync	1,80 GB	78,16 GB	78,16 GB	0ms	
	Data obtained	20 sec	20 sec	20 sec	20 sec	20 sec	20 sec	

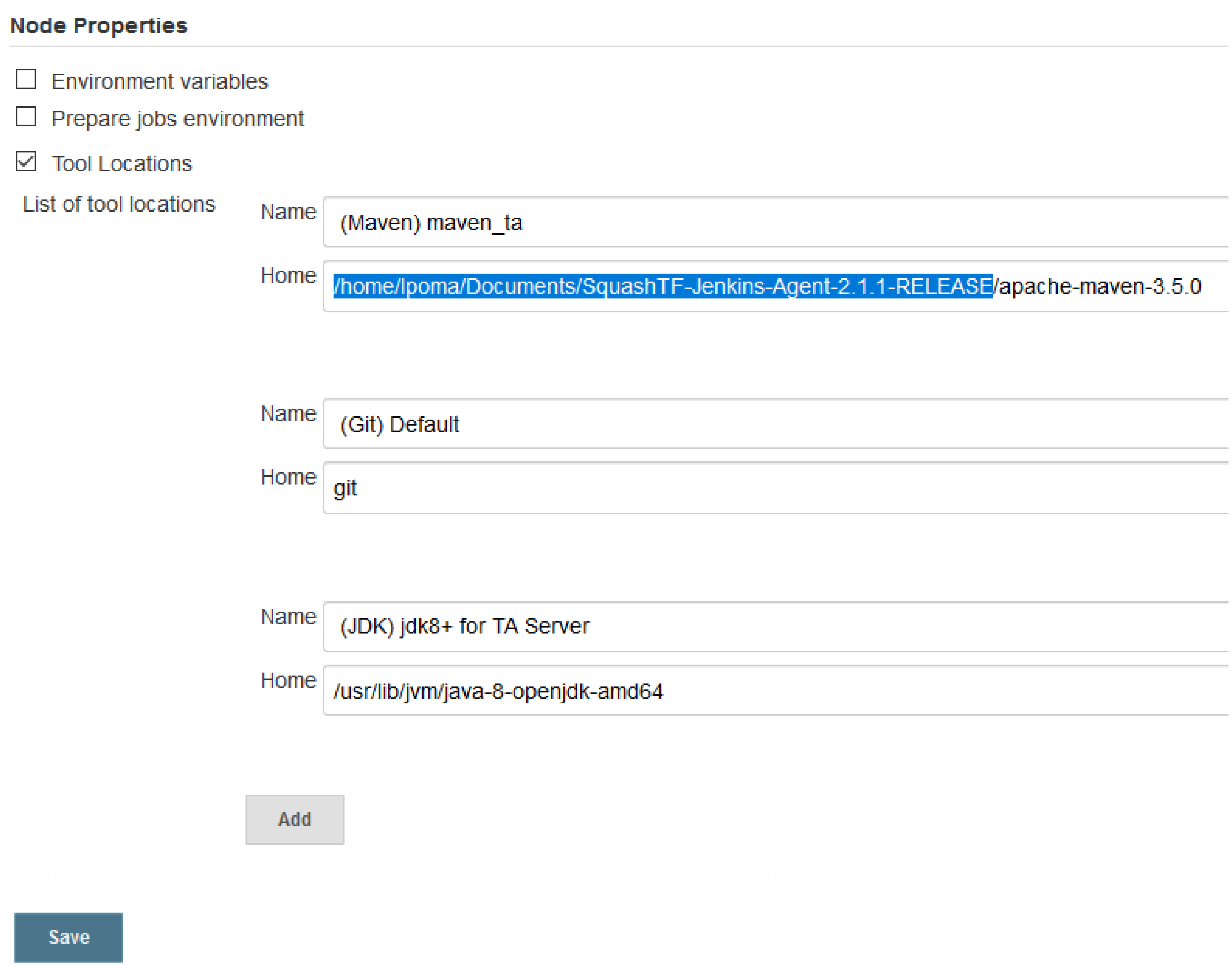
3. In the field `Remote root directory` replace the path of your old version with the new one (everything before **/workspace**).



Name	Integrated-SquashTF-Jenkins-Agent_ubuntu
Description	TF slave Installation on unix ubuntu
# of executors	1
Remote root directory	/home/lpoma/Documents/SquashTF-Jenkins-Agent-2.1.1-RELEASE/workspace
Labels	linux
Usage	Use this node as much as possible

4. On the same screen find the section `Node Properties` and the sub-section `List of tool locations`.

5. In the field `Home` under `(Maven) maven_ta` replace the path of your old version with the new one (everything before **/apache-maven-3.5.0**).



Node Properties

☐ Environment variables

☐ Prepare jobs environment

☑ Tool Locations

List of tool locations

Name	(Maven) maven_ta
Home	/home/lpoma/Documents/SquashTF-Jenkins-Agent-2.1.1-RELEASE/apache-maven-3.5.0
Name	(Git) Default
Home	git
Name	(JDK) jdk8+ for TA Server
Home	/usr/lib/jvm/java-8-openjdk-amd64

6. Check that all your custom configurations are correct then click on **Save**.

7. Restart your **Squash TF Execution Server** and launch the agent, then run your jobs to make sure the update was sucessfull.

Docker image

1. Follow step 9 and step 10 from *these* setup instructions to link the agent container to the server container (you can read steps 1 to 8 for context).

2. Run your jobs on the updated Jenkins agent to make sure the update was successfull.

CHAPTER 2

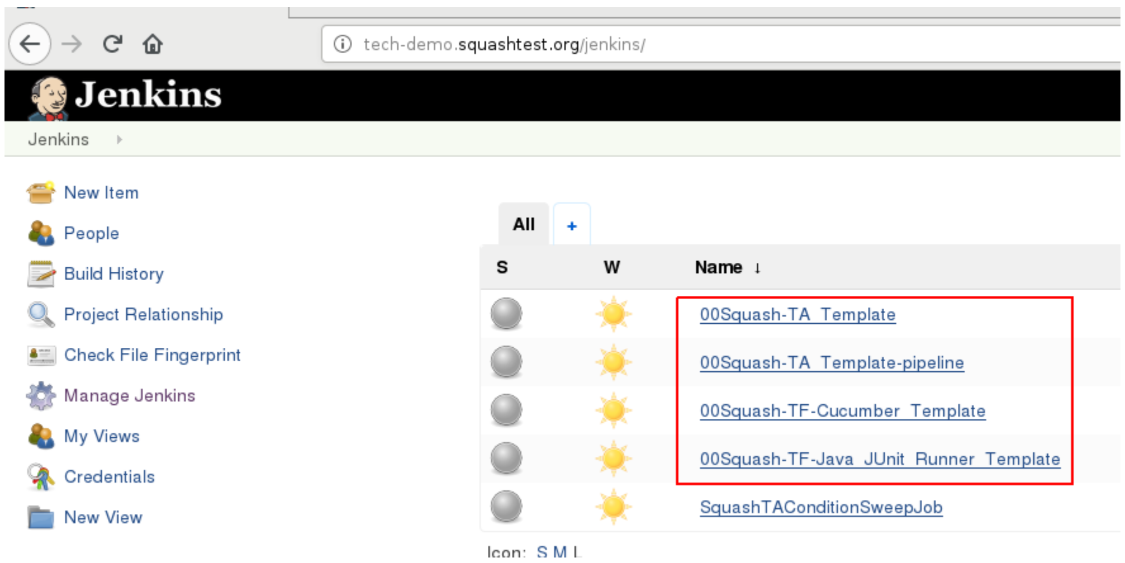
Use Squash TF Execution Server

2.1 Create a job in Squash TF Execution Server

NB : this procedure requires authentication with a user account with job creation rights

2.1.1 Choosing the template

The **Squash TF Execution Server** offers various templates, depending on the test you want to run.

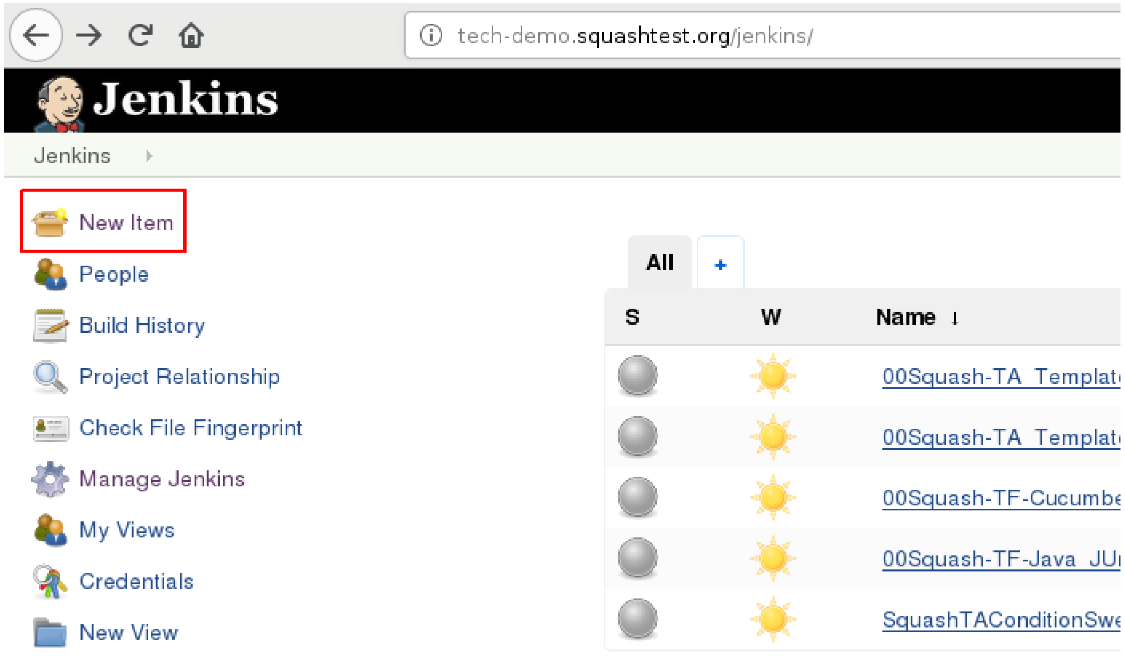


You may make your choice according to the following table :

Test project	Template
Junit 4 based test project (including Selenium)	`00Squash-TF-Java_JUnit_Runner_Template`
Junit 5 based test project (including Selenium)	`00Squash-TF-Java_JUnit_Runner_Template`
Test project managed from Squash TM's Gherkin Test cases	`00Squash-TF-Cucumber_Template`
Robot Framework test project	`00Squash-TF_Robotframework_Template`
Keywork Framework[1] test project	`00Squash-TA_Template`
Keywork Framework[1] test project with two execution phases[2]	`00Squash-TA_Template-pipeline`

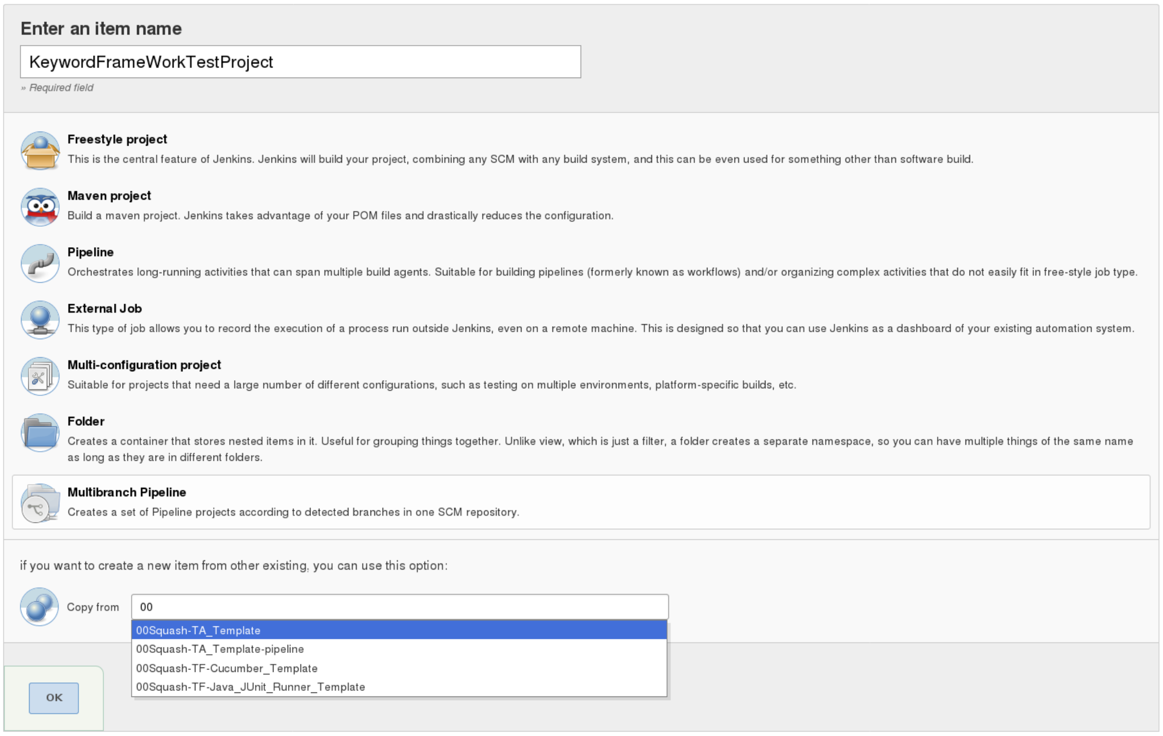
2.1.2 Creating the job

Once you have chosen the appropriate template, create your job from this template through the **New Item** menu.



1. Enter a name for your test job
2. Type the name of the template in the **Copy from** field at the bottom of the new item page. Once you've begun to type auto-completion will trigger and you'll be able to choose from existing templates, then click on the *OK* button to finish job creation.

[1] The **Keyword Framework** is the new version of our testing framework derived from the **Squash TA DSL** framework.
[2] This pipelined job class allows an asynchronous wait between two test phases. Between these two test phases, the pipeline is active but no executor is allocated. The trigger for phase II is a SQL query. The current version only allows tests triggered from **Squash TM**.



Enter an item name

KeywordFrameWorkTestProject

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

External Job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

if you want to create a new item from other existing, you can use this option:

Copy from: 00

00Squash-TA_Template
00Squash-TA_Template-pipeline
00Squash-TF-Cucumber_Template
00Squash-TF-Java_JUnit_Runner_Template

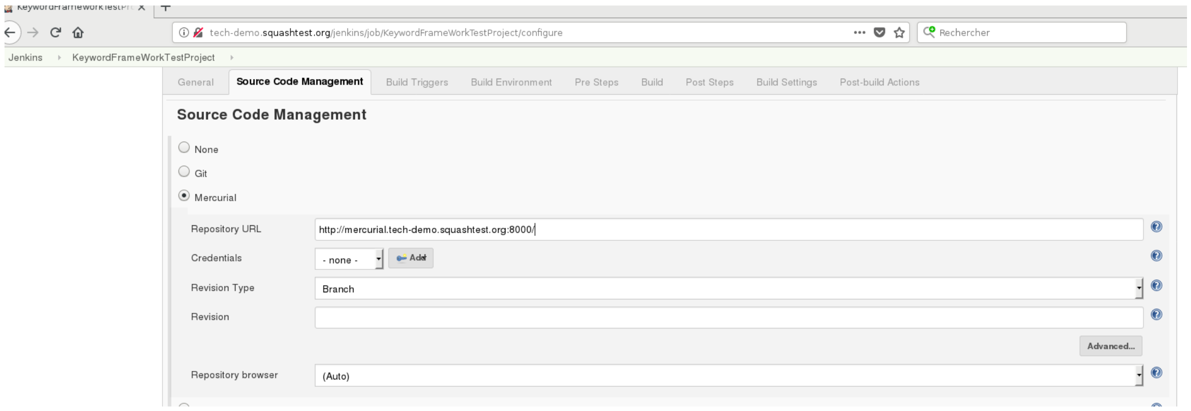
OK

2.1.3 Setting the job up

After creating the job, you have to set it up.

Setting a Squash TF job up (general case)

1. The main required settings for a Squash TF job are the Source Code Management parameters. The job needs to known how to get the test project sources. The illustration below shows the setup of a mercurial hosted project, but any SCM will do as long as it is supported in jenkins. Once you've entered the SCM parameters, hit the **Save** button.



tech-demo.squashtest.org/jenkins/job/KeywordFrameWorkTestProject/configure

Jenkins › KeywordFrameWorkTestProject ›

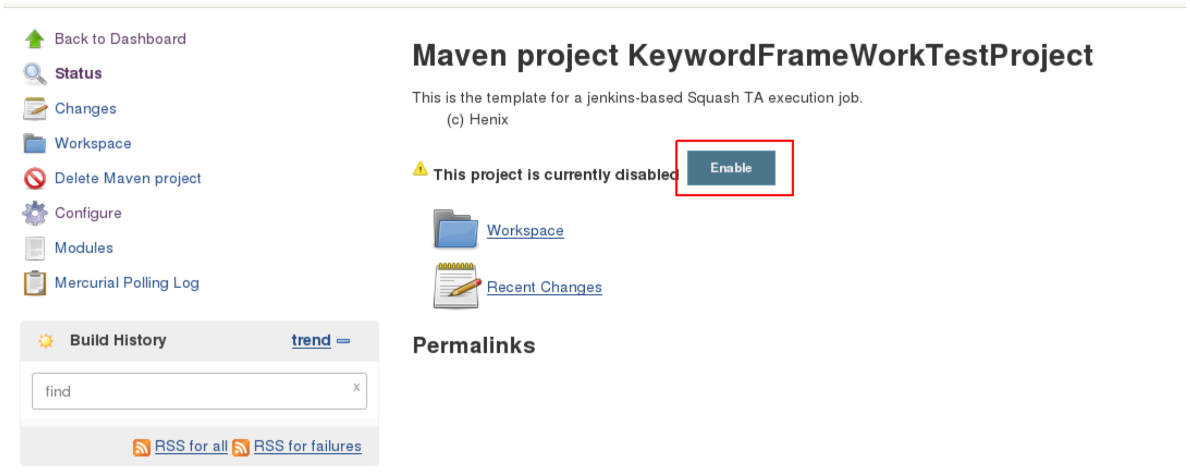
General | Source Code Management | Build Triggers | Build Environment | Pre Steps | Build | Post Steps | Build Settings | Post-build Actions

Source Code Management

None [✓ Mercurial] Git

Repository URL	http://mercurial.tech-demo.squashtest.org:8000/
Credentials	- none -
Revision Type	Branch
Revision	
Repository browser	(Auto)

2. The next - and last - step is to enable the job by clicking on the **Enable** button.



Maven project KeywordFrameWorkTestProject

This is the template for a jenkins-based Squash TA execution job.
(c) Henix

This project is currently disabled

Setting a Robot framework job up

1. First you have to follow the *general case setup describe below*
2. For a job based on the Squash TF Robot Framework job template, one more configuration step is needed. As Robot Framework tests needs an execution environment with python, then by default the job is configured to only run on an executor which have the label `robotFW`. Out of the box Squash TF Execution Server doesn't have any executor with this label. So before launch a Robot framework job you need to :

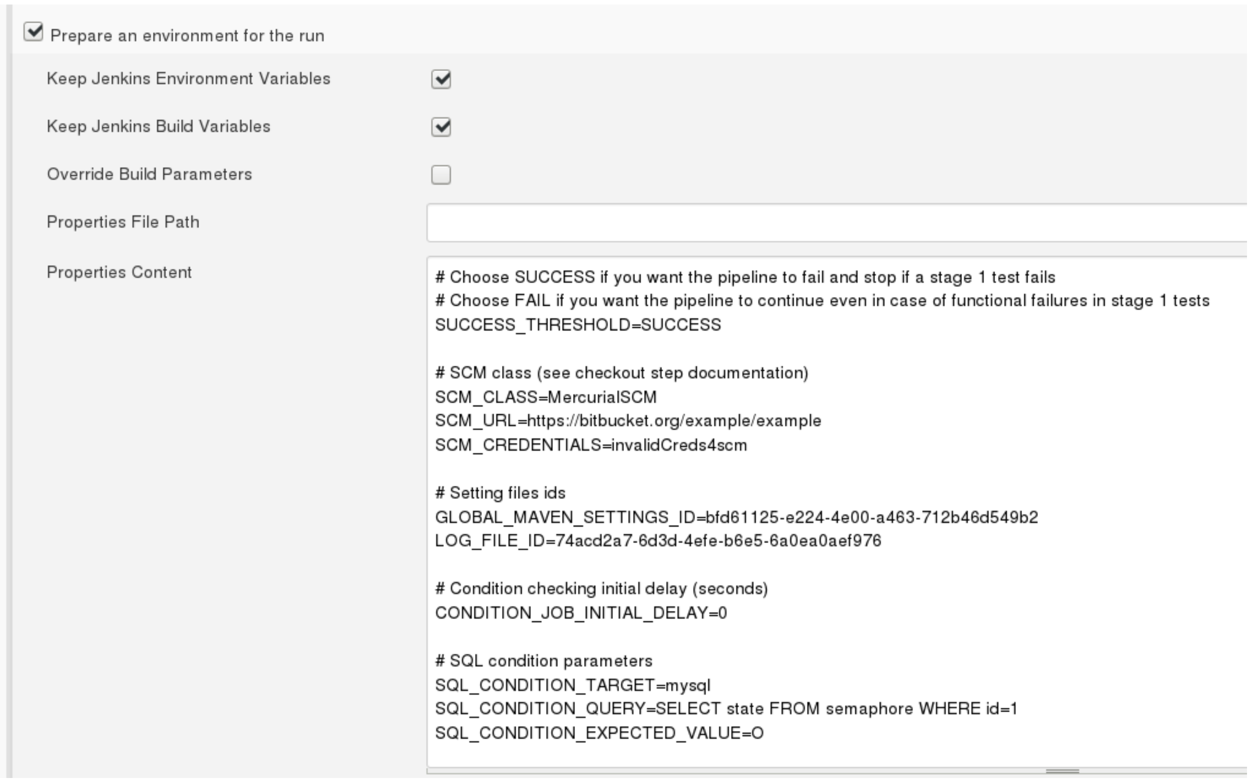
   - Have an execution environment with python and robot installed on it (on the master or on an agent) (see Squash TF Robot Framework Runner prerequisites for more details)
   - Give to this executor the label `robotFW`. To do so :
     - Click on `Manage Jenkins` (left menu)
     - Click on `Manage Nodes` (left menu)
     - Select the desired Node
     - Click on `Configure` (left menu)
     - In the `Labels` field add `robotFW` (multiple labels are separated by a space)
     - Click on the `Save` button

   You're now ready to execute a RobotFramework job

   Note: You may need to wait few minutes before Jenkins recognize this node as having the `robotFW` label.

Setting a two-phased Keyword Framework job up

Most parameters for the two-phased Keyword DSL pipeline are setup through the Environment injector plugin.



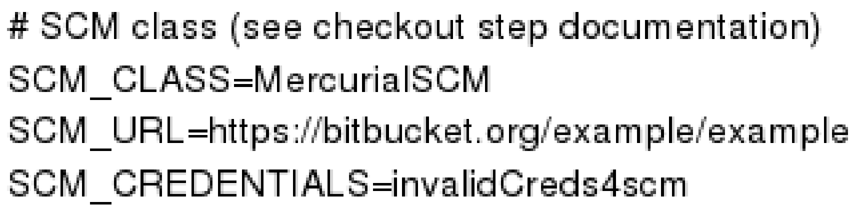
☑ Prepare an environment for the run

Keep Jenkins Environment Variables	☑
Keep Jenkins Build Variables	☑
Override Build Parameters	☐
Properties File Path	
Properties Content	

```
# Choose SUCCESS if you want the pipeline to fail and stop if a stage 1 test fails
# Choose FAIL if you want the pipeline to continue even in case of functional failures in stage 1 tests
SUCCESS_THRESHOLD=SUCCESS

# SCM class (see checkout step documentation)
SCM_CLASS=MercurialSCM
SCM_URL=https://bitbucket.org/example/example
SCM_CREDENTIALS=invalidCreds4scm

# Setting files ids
GLOBAL_MAVEN_SETTINGS_ID=bfd61125-e224-4e00-a463-712b46d549b2
LOG_FILE_ID=74acd2a7-6d3d-4efe-b6e5-6a0ea0aef976

# Condition checking initial delay (seconds)
CONDITION_JOB_INITIAL_DELAY=0

# SQL condition parameters
SQL_CONDITION_TARGET=mysql
SQL_CONDITION_QUERY=SELECT state FROM semaphore WHERE id=1
SQL_CONDITION_EXPECTED_VALUE=O
```

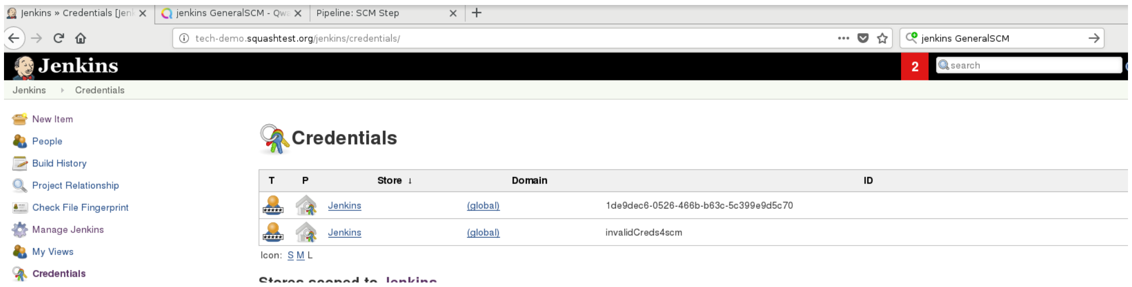
1. SCM parameters

   The first thing you'll want to set is the SCM part.



```
# SCM class (see checkout step documentation)
SCM_CLASS=MercurialSCM
SCM_URL=https://bitbucket.org/example/example
SCM_CREDENTIALS=invalidCreds4scm
```

   - `SCM_CLASS` : This first version of the pipeline only supports the use of the mercurial SCM. We plan to support other SCM providers compatible with the jenkins CSM plugin in the future : as they come up, you'll be able to select other values for the `SCM_CLASS` parameter.
   - `SCM_URL` : As shown, you may specify a repository URL through the `SCM_URL` parameter (the pipeline pulls from branch default).
   - `SCM_CREDENTIALS` : This parameter holds a credential identifier. The value must match the ID of a registered jenkins credential record (see below).



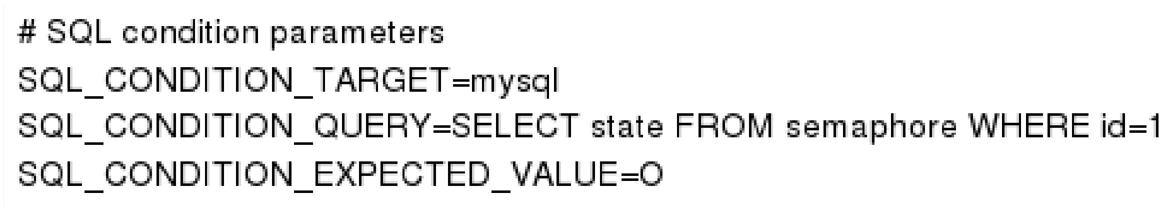
Credentials

T	P	Store ↓	Domain	ID
		Jenkins	(global)	1de9dec6-0526-466b-b63c-5c399e9d5c70
		Jenkins	(global)	invalidCreds4scm

Icon: S M L

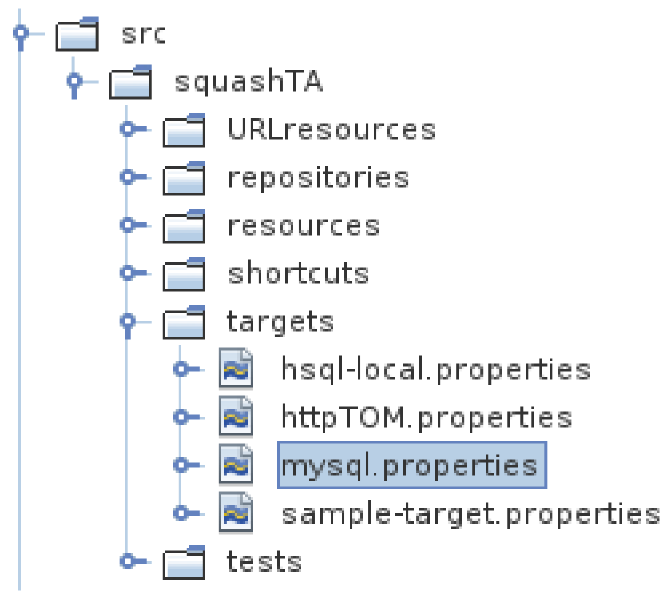
2. SQL condition parameters

   The trigger of the second test execution phase is a SQL query. Here is how you may set this condition up :



```
# SQL condition parameters
SQL_CONDITION_TARGET=mysql
SQL_CONDITION_QUERY=SELECT state FROM semaphore WHERE id=1
SQL_CONDITION_EXPECTED_VALUE=0
```

   - `SQL_CONDITION_TARGET` : this parameter must match the name of a target defined by the Keyword Framework[1] test project.



```
src
  squashTA
    URLresources
    repositories
    resources
    shortcuts
    targets
      hsql-local.properties
      httpTOM.properties
      mysql.properties
      sample-target.properties
    tests
```

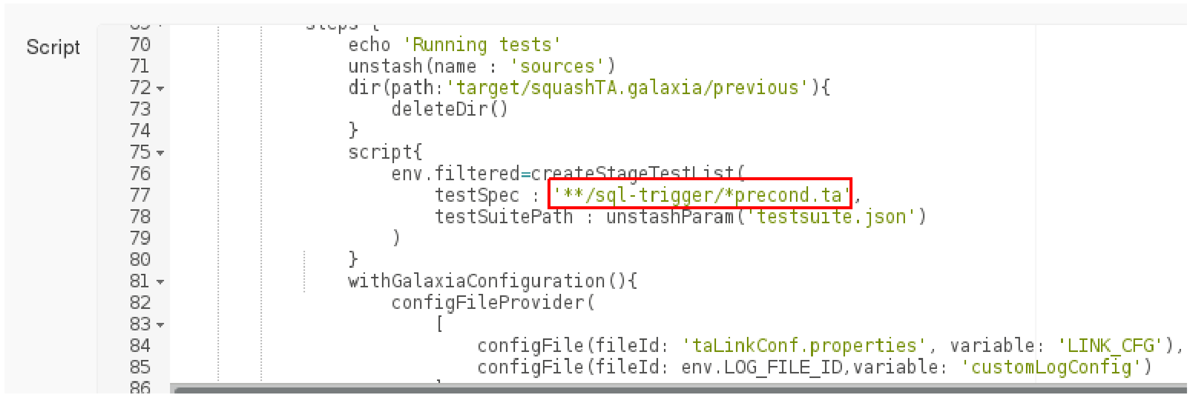
   - `SQL_CONDITION_QUERY` : this parameter defines the SQL Query executed when evaluating the condition. It may be any SQL query that selects a unique value (one column, one row). More columns and rows may be fetched but they will be ignored. We advise you to avoid this and only fetch the expected value for clarity.
   - `SQL_CONDITION_EXPECTED_VALUE` : this parameter is compared to the query result. If they match, the condition is considered fulfilled and test execution resumes for phase two.

3. Selection of phase I and phase II tests Phase I and phase II tests are chosen among tests triggered by the received **Squash TM** execution order. They are picked using two filters. The default filters are, respectively :

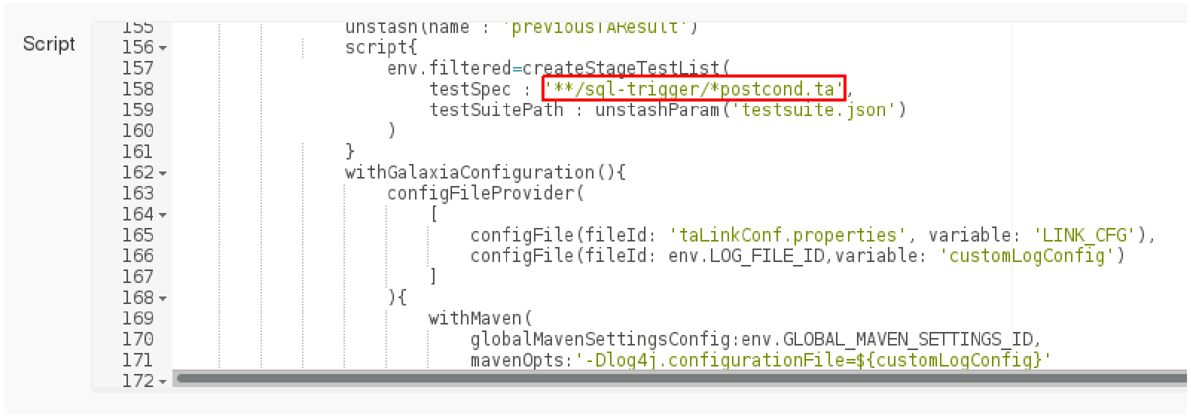
Phase	Filter definition	Selected tests
I	**/sql-trigger/*precond.ta	Any test in a sub-directory named sql-trigger with a name ending with `precond.ta`
II	**/sql-trigger/*postcond.ta	Any test in a sub-directory named sql-trigger with a name ending with `postcond.ta`

   As of now[3] , the only way to change this is to search the pipeline code for the `**/sql-trigger/*precond.ta` and `**/sql-trigger/*postcond.ta` strings, (each exists in one place only) and edit them. They follow the ant-like wildcard format, where * means 'anything but the file separator, including an emtpy string', and ** means 'any thing, even path separators'.

[3] changing the phase test filters will be made easier in the near future. This enhancement is planned for version *2.1.0-RELEASE*, which should be published around April 2019.



```
Script
                echo 'Running tests'
                unstash(name : 'sources')
                dir(path:'target/squashTA.galaxia/previous'){
                    deleteDir()
                }
                script{
                    env.filtered=createStageTestList(
                        testSpec : '**/sql-trigger/*precond.ta',
                        testSuitePath : unstashParam('testsuite.json')
                    )
                }
                withGalaxiaConfiguration(){
                    configFileProvider(
                        [
                            configFile(fileId: 'taLinkConf.properties', variable: 'LINK_CFG'),
                            configFile(fileId: env.LOG_FILE_ID,variable: 'customLogConfig')
```

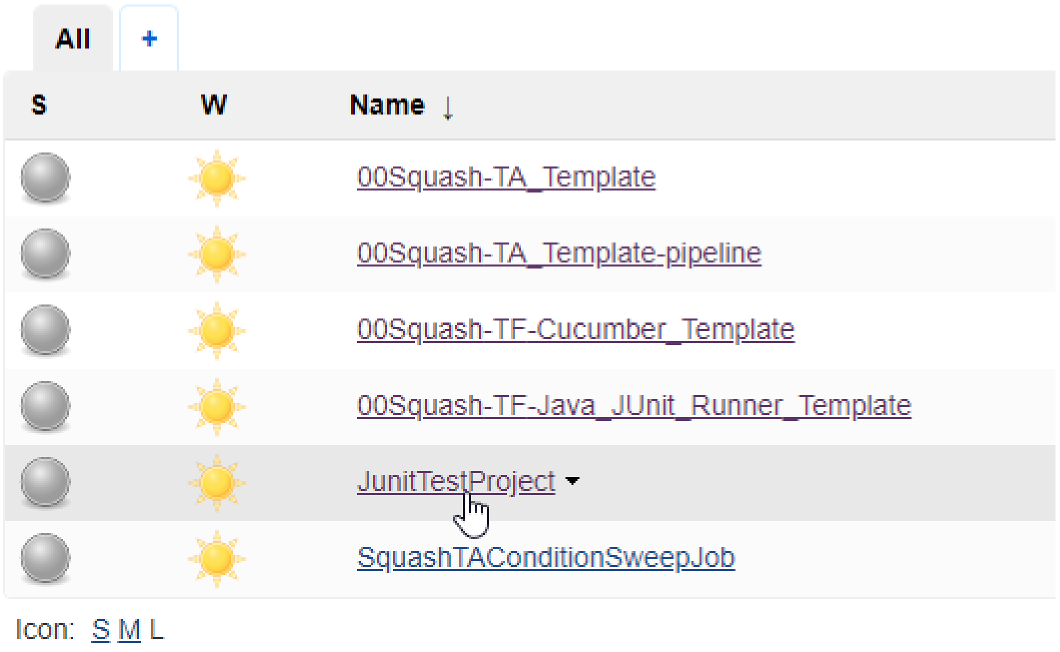


```
Script
                unstash(name : 'previousTAResult')
                script{
                    env.filtered=createStageTestList(
                        testSpec : '**/sql-trigger/*postcond.ta',
                        testSuitePath : unstashParam('testsuite.json')
                    )
                }
                withGalaxiaConfiguration(){
                    configFileProvider(
                        [
                            configFile(fileId: 'taLinkConf.properties', variable: 'LINK_CFG'),
                            configFile(fileId: env.LOG_FILE_ID,variable: 'customLogConfig')
                        ]
                    ){
                        withMaven(
                            globalMavenSettingsConfig:env.GLOBAL_MAVEN_SETTINGS_ID,
                            mavenOpts:'-Dlog4j.configurationFile=${customLogConfig}'
```

2.2 Use a job based on Java Junit runner job template

2.2.1 Selecting the job

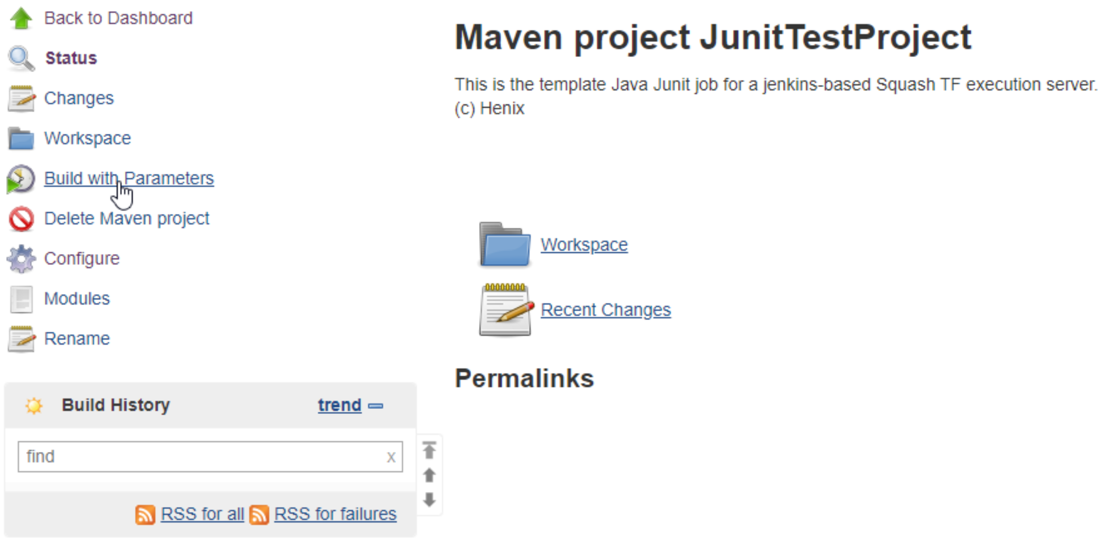
Once you've *created a job using the Squash-TA Template*, select it in the list of jobs available :


All
+
S
W
Name ↓
00Squash-TA_Template
00Squash-TA_Template-pipeline
00Squash-TF-Cucumber_Template
00Squash-TF-Java_JUnit_Runner_Template
JunitTestProject
SquashTAConditionSweepJob
Icon: S M L

On the job's page, you'll be able to launch a build, access various informations regarding the job or make modifications.

Note: You'll have to be logged in as an administrator to modify your job (rename, delete or configure) or to launch a build.

To launch a build, click on *'Build with Parameters'* :

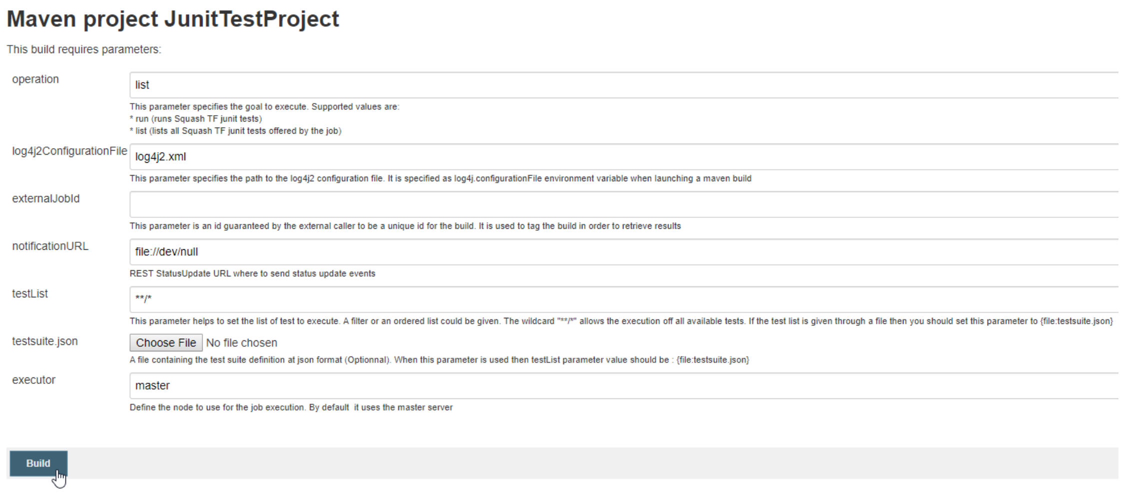


Maven project JunitTestProject

This is the template Java Junit job for a jenkins-based Squash TF execution server.
(c) Henix

2.2.2 Setting the parameters of the build

You can then configure some of the parameters of the build :



Maven project JunitTestProject

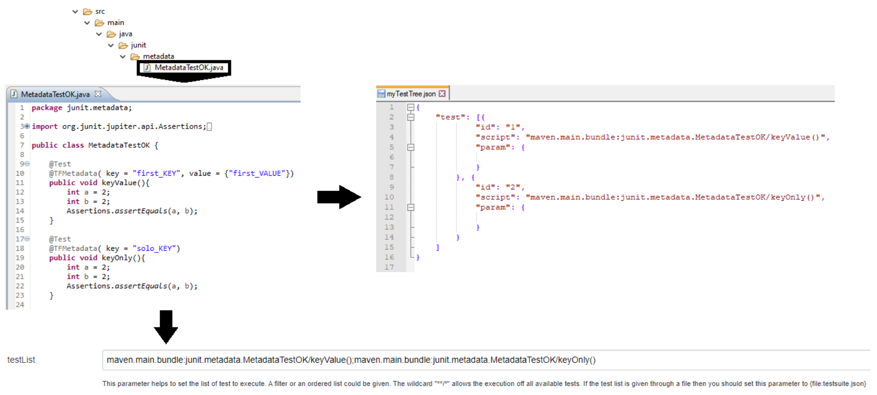
This build requires parameters:

operation	list This parameter specifies the goal to execute. Supported values are: * run (runs Squash TF junit tests) * list (lists all Squash TF junit tests offered by the job)
log4j2ConfigurationFile	log4j2.xml This parameter specifies the path to the log4j2 configuration file. It is specified as log4j.configurationFile environment variable when launching a maven build
externalJobId	 This parameter is an id guaranteed by the external caller to be a unique id for the build. It is used to tag the build in order to retrieve results
notificationURL	file://dev/null REST StatusUpdate URL where to send status update events
testList	**/* This parameter helps to set the list of test to execute. A filter or an ordered list could be given. The wildcard "**/*" allows the execution off all available tests. If the test list is given through a file then you should set this parameter to {file:testsuite.json}
testsuite.json	Choose File No file chosen A file containing the test suite definition at json format (Optionnal). When this parameter is used then testList parameter value should be : {file:testsuite.json}
executor	master Define the node to use for the job execution. By default it uses the master server

Operation and TestList

You can specify two types of goals to execute in the *'operation'* field: **list** or **run**.

> **> list** : This will generate a *json* file listing all the tests present in your project.
>
> This file is used by **Squash TM** in the context of the **TM-TF** link to determine the tests that can be executed.
>
> **> run** : This will run all the tests specified in the *'testList'* field or in a user created *json* file.
>
> If you are using the **TM-TF** link, **Squash TM** will generate and transmit to **Squash TF** a *testsuite.json* file containing the list of tests to execute. In that case you don't have to alter the field *'testList'* or specify a *json* file.
>
> Otherwise, if you want to provide manually to your **run** the list of tests to execute, you can procede in two ways :
>
> - Enter the path to the tests you wish to execute (see this page for details on Junit tests naming scheme), separated by a semicolon (and **no space** after the semicolon), in the *'testList'* field. You can also have the build execute all the tests present in you project, using : **/*.
> - Provide a *json* type file (by clicking on the *'Choose File'* button on the *'testsuite.json'* line) containing the list of tests you wish to execute, and fill the *'testList'* field with : **{file:testsuite.json}**.
>
>   **Exemple of each method** :



```java
package junit.metadata;

import org.junit.jupiter.api.Assertions;

public class MetadataTestOK {

    @Test
    @TFMetadata( key = "first_KEY", value = {"first_VALUE"})
    public void keyValue(){
        int a = 2;
        int b = 2;
        Assertions.assertEquals(a, b);
    }

    @Test
    @TFMetadata( key = "solo_KEY")
    public void keyOnly(){
        int a = 2;
        int b = 2;
        Assertions.assertEquals(a, b);
    }

```

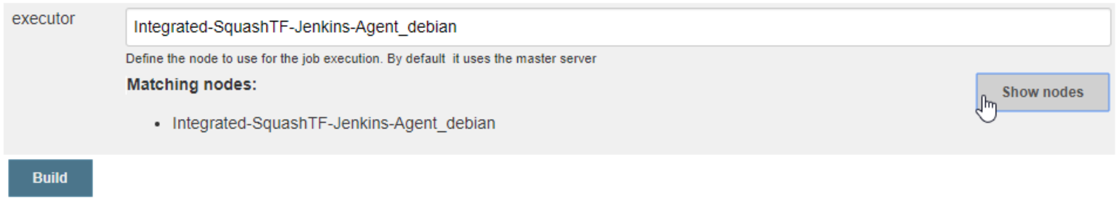
```json
{
    "test": [{
            "id": "1",
            "script": "maven.main.bundle:junit.metadata.MetadataTestOK/keyValue()",
            "param": {

            }
        }, {
            "id": "2",
            "script": "maven.main.bundle:junit.metadata.MetadataTestOK/keyOnly()",
            "param": {

            }
        }
    ]
}
```

testList: maven.main.bundle:junit.metadata.MetadataTestOK/keyValue();maven.main.bundle:junit.metadata.MetadataTestOK/keyOnly()

This parameter helps to set the list of test to execute. A filter or an ordered list could be given. The wildcard "**/*" allows the execution off all available tests. If the test list is given through a file then you should set this parameter to {file:testsuite.json}

Executor

If you want to launch the build on an **Squash TF** agent located on a distant machine and properly configured (see *this page* for the agent installation), enter the exact name of the agent, or its label, in the *'executor'* field. Click on the *'Show nodes'* button on the bottom right of the field to validate that you've entered a correct name :



executor: Integrated-SquashTF-Jenkins-Agent_debian

Define the node to use for the job execution. By default it uses the master server

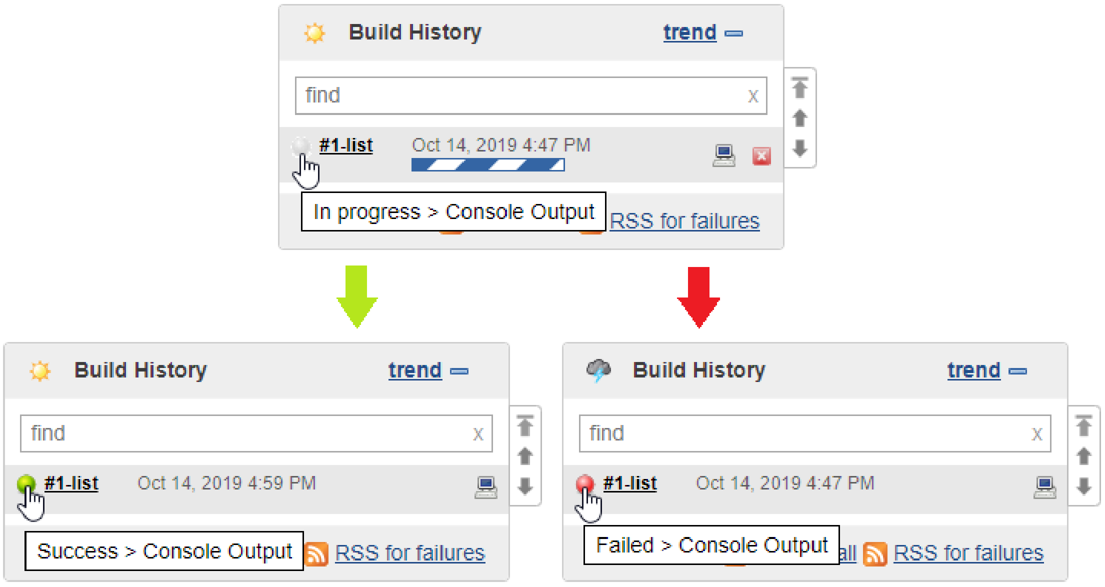
Matching nodes:

- Integrated-SquashTF-Jenkins-Agent_debian

2.2.3 Launching a build

Once you've specified the parameters of your build, click on the *'Build'* button to launch the build.

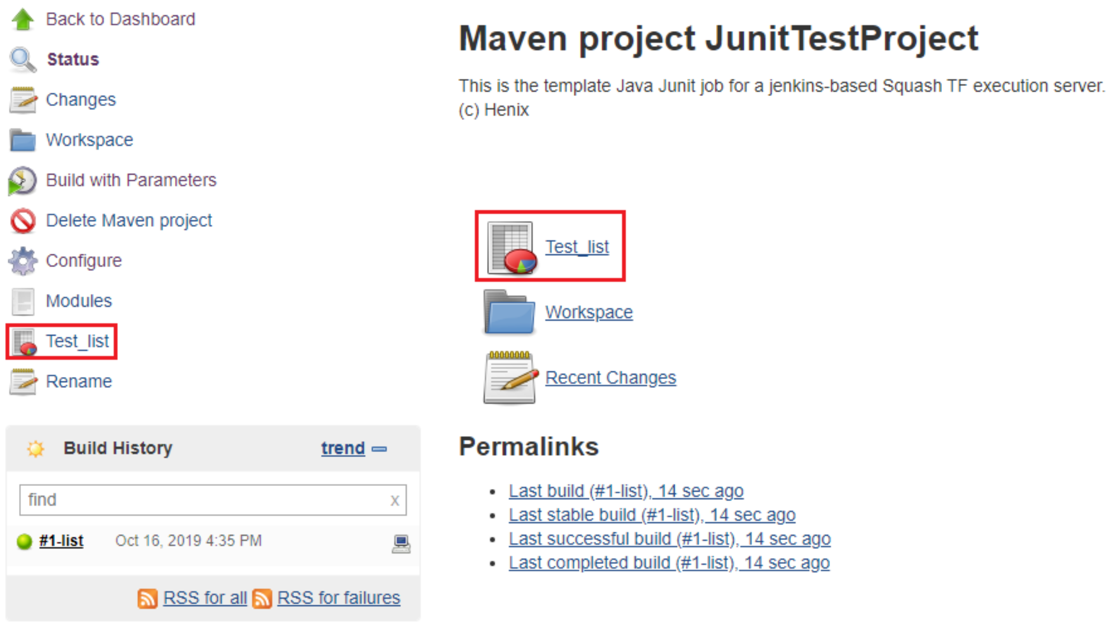
Clicking on the dot (grey, red or green depending on the status of the build) next to the build name in the *'Build History'* window will show the console output.



2.2.4 Build outputs

Test list

After the first **list** build has been performed, the generated *'Test_list'* of the last **list** build executed will be available on the job's page :



Maven project JunitTestProject

This is the template Java Junit job for a jenkins-based Squash TF execution server.
(c) Henix

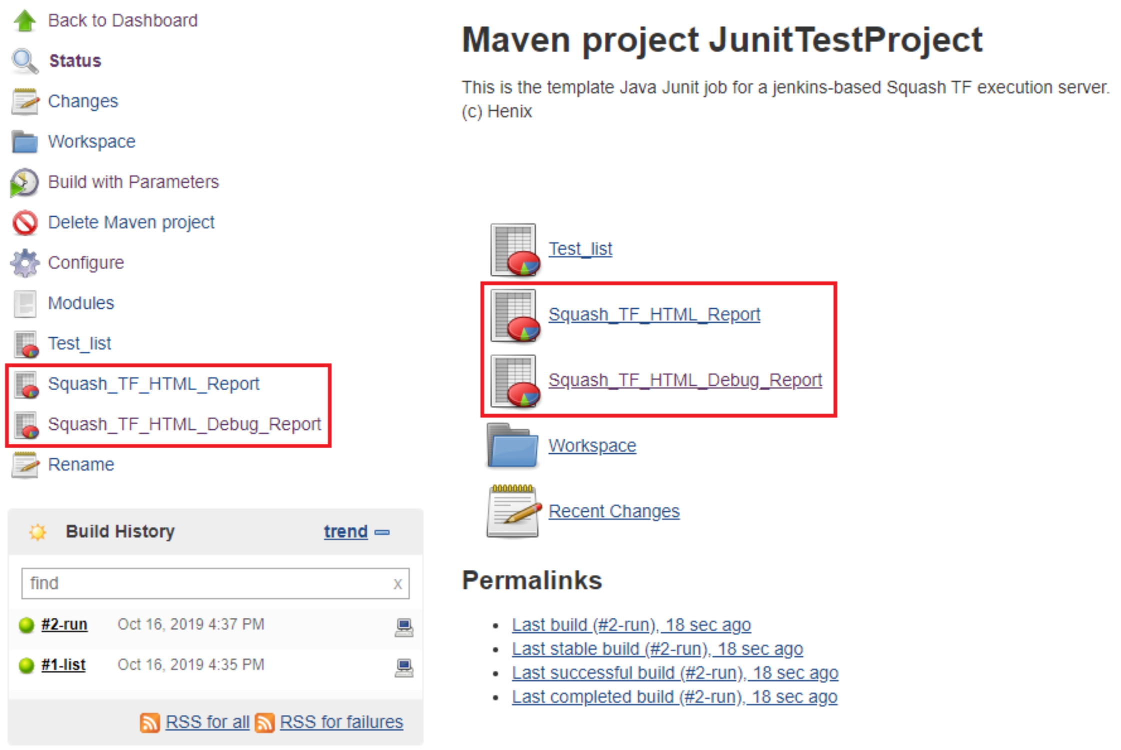
Permalinks

- Last build (#1-list), 14 sec ago
- Last stable build (#1-list), 14 sec ago
- Last successful build (#1-list), 14 sec ago
- Last completed build (#1-list), 14 sec ago

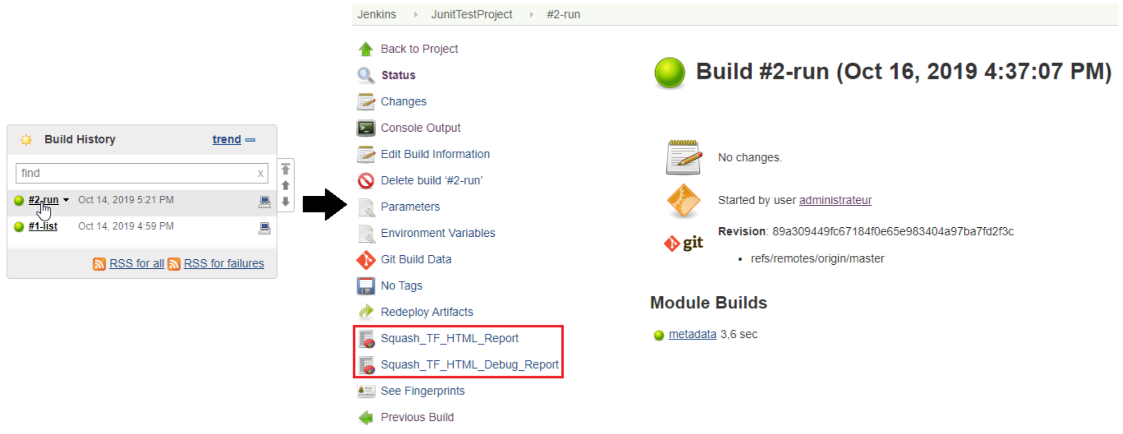
This is the *json* file fetched by **Squash TM** if using the **TM-TF** link.

HTML reports

In the same manner, the generated *'Squash_TF_HTML_Report'* and *'Squash_TF_HTML_Debug_Report'* of the last **run** build executed will be available on the job's page :


Back to Dashboard
Status
Changes
Workspace
Build with Parameters
Delete Maven project
Configure
Modules
Test_list
Squash_TF_HTML_Report
Squash_TF_HTML_Debug_Report
Rename
Build History
trend
find
#2-run Oct 16, 2019 4:37 PM
#1-list Oct 16, 2019 4:35 PM
RSS for all
RSS for failures
Maven project JunitTestProject
This is the template Java Junit job for a jenkins-based Squash TF execution server.
(c) Henix
Test_list
Squash_TF_HTML_Report
Squash_TF_HTML_Debug_Report
Workspace
Recent Changes
Permalinks
Last build (#2-run), 18 sec ago
Last stable build (#2-run), 18 sec ago
Last successful build (#2-run), 18 sec ago
Last completed build (#2-run), 18 sec ago

If you click directly on a **run** build's name, you'll have access to its page with the corresponding **run**'s HTML reports :

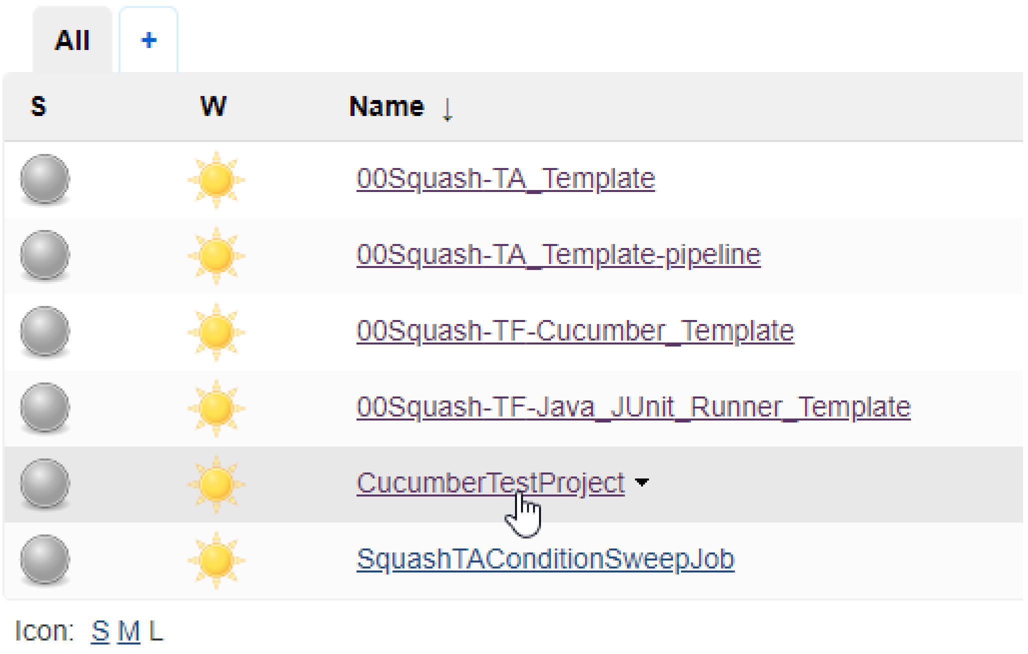


Note: For more indepth details about the Junit runner, please consult its dedicated section.

2.3 Use a job based on Cucumber runner job template

2.3.1 Selecting the job

Once you've *created a job using the Squash-TA Template*, select it in the list of jobs available :



All +

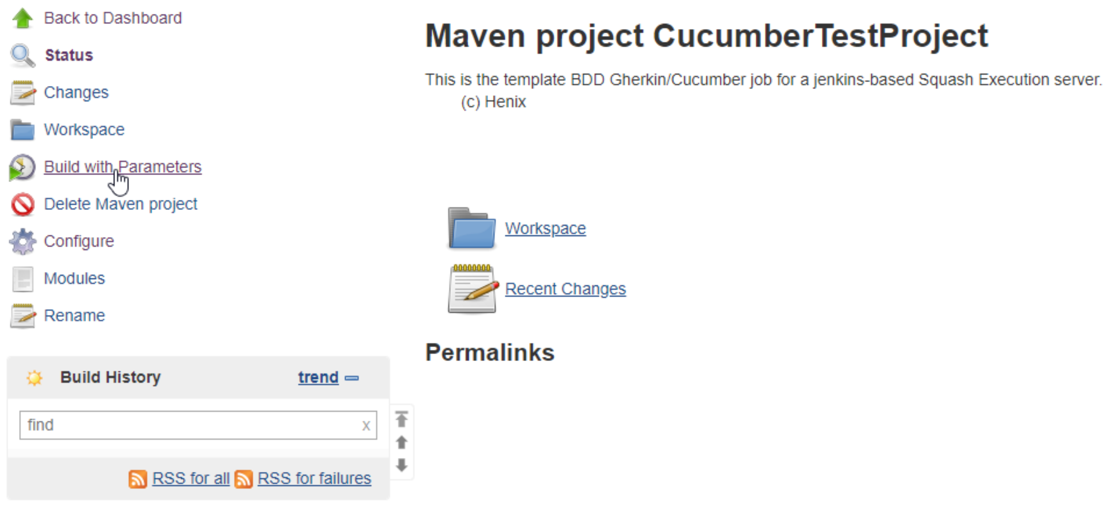
S	W	Name ↓
●	☀	00Squash-TA_Template
●	☀	00Squash-TA_Template-pipeline
●	☀	00Squash-TF-Cucumber_Template
●	☀	00Squash-TF-Java_JUnit_Runner_Template
●	☀	CucumberTestProject
●	☀	SquashTAConditionSweepJob

Icon: S M L

On the job's page, you'll be able to launch a build, access various informations regarding the job or make modifications.

Note: You'll have to be logged in as an administrator to modify your job (rename, delete or configure) or to launch a build.

To launch a build, click on *'Build with Parameters'* :

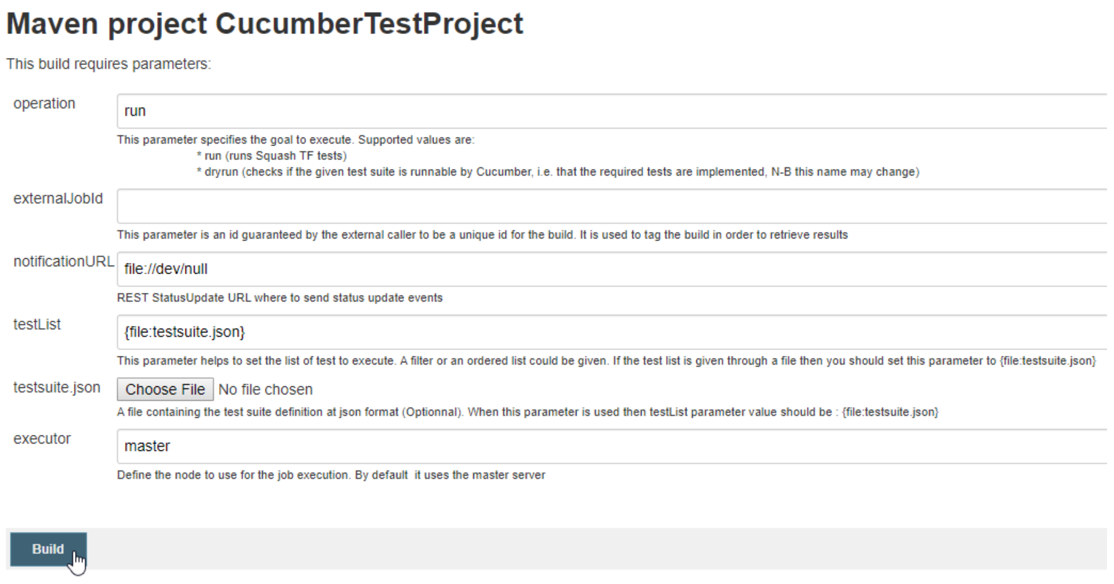


Maven project CucumberTestProject

This is the template BDD Gherkin/Cucumber job for a jenkins-based Squash Execution server.
(c) Henix

2.3.2 Setting the parameters of the build

You can then configure some of the parameters of the build :



Maven project CucumberTestProject

This build requires parameters:

operation: run

This parameter specifies the goal to execute. Supported values are:
* run (runs Squash TF tests)
* dryrun (checks if the given test suite is runnable by Cucumber, i.e. that the required tests are implemented, N-B this name may change)

externalJobId:

This parameter is an id guaranteed by the external caller to be a unique id for the build. It is used to tag the build in order to retrieve results

notificationURL: file://dev/null

REST StatusUpdate URL where to send status update events

testList: {file:testsuite.json}

This parameter helps to set the list of test to execute. A filter or an ordered list could be given. If the test list is given through a file then you should set this parameter to {file:testsuite.json}

testsuite.json: Choose File No file chosen

A file containing the test suite definition at json format (Optionnal). When this parameter is used then testList parameter value should be : {file:testsuite.json}

executor: master

Define the node to use for the job execution. By default it uses the master server

Operation and TestList

You can specify two types of goals to execute in the *'operation'* field: **dryrun** or **run**.

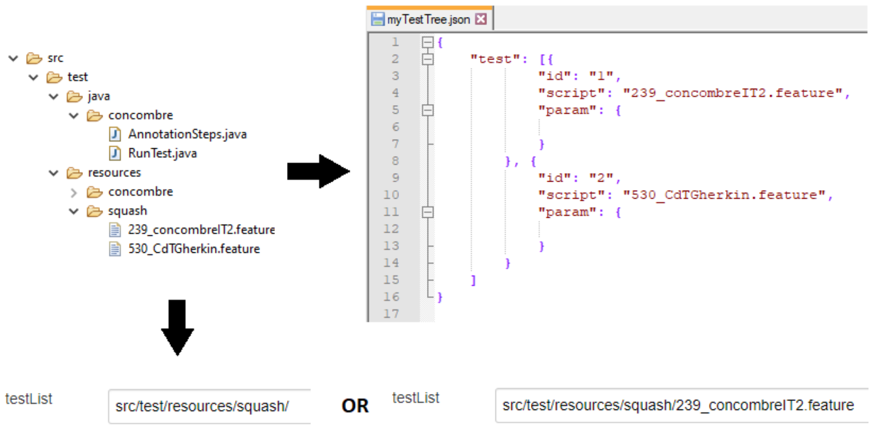
> **> dryrun** : This will check if the tests specified in the *'testList'* field or in a user created *json* file are runnable by Cucumber (meaning that they are implemented).
>
> **> run** : This will run all the tests specified in the *'testList'* field or in a user created *json* file.

If you are using the **TM-TF** link, **Squash TM** will generate and transmit to **Squash TF** a *testsuite.json* file containing the list of tests to execute. In that case you don't have to alter the field *'testList'* or specify a *json* file.

Otherwise, if you want to provide manually to your **run** the list of tests to execute, you can procede in two ways :

- Enter the relative path (to the root of your project) of a single test (*.feature* type file) you wish to execute in the *'testList'* field. Alternatively, you can specify the path of a folder containing several test files. You can also have the build execute all the tests present in you project, by leaving the *'testList'* field blank.
- Provide a *json* type file (by clicking on the *'Choose File'* button on the *'testsuite.json'* line) containing the list of tests you wish to execute, and fill the *'testList'* field with : **{file:testsuite.json}**.

  Exemple of each method :



```
src
  test
    java
      concombre
        AnnotationSteps.java
        RunTest.java
    resources
      concombre
      squash
        239_concombreIT2.feature
        530_CdTGherkin.feature
```

myTestTree.json

```
{
    "test": [{
        "id": "1",
        "script": "239_concombreIT2.feature",
        "param": {

        }
    }, {
        "id": "2",
        "script": "530_CdTGherkin.feature",
        "param": {

        }
    }
    ]
}
```

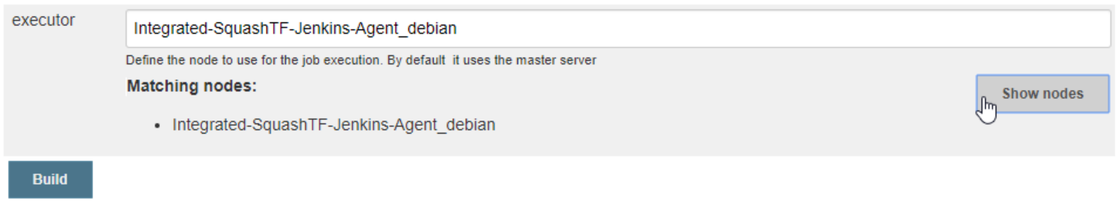
testList: src/test/resources/squash/

OR

testList: src/test/resources/squash/239_concombreIT2.feature

Executor

If you want to launch the build on an **Squash TF** agent located on a distant machine and properly configured (see *this page* for the agent installation), enter the exact name of the agent, or its label, in the *'executor'* field. Click on the *'Show nodes'* button on the bottom right of the field to validate that you've entered a correct name :



executor: Integrated-SquashTF-Jenkins-Agent_debian

Define the node to use for the job execution. By default it uses the master server

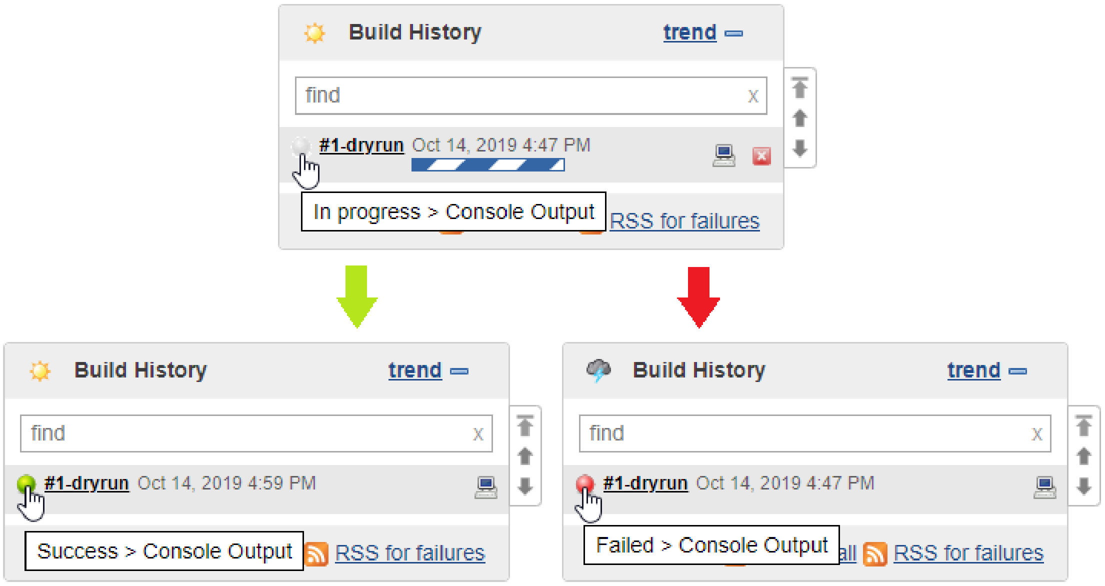
Matching nodes:

- Integrated-SquashTF-Jenkins-Agent_debian

2.3.3 Launching a build

Once you've specified the parameters of your build, click on the *'Build'* button to launch the build.

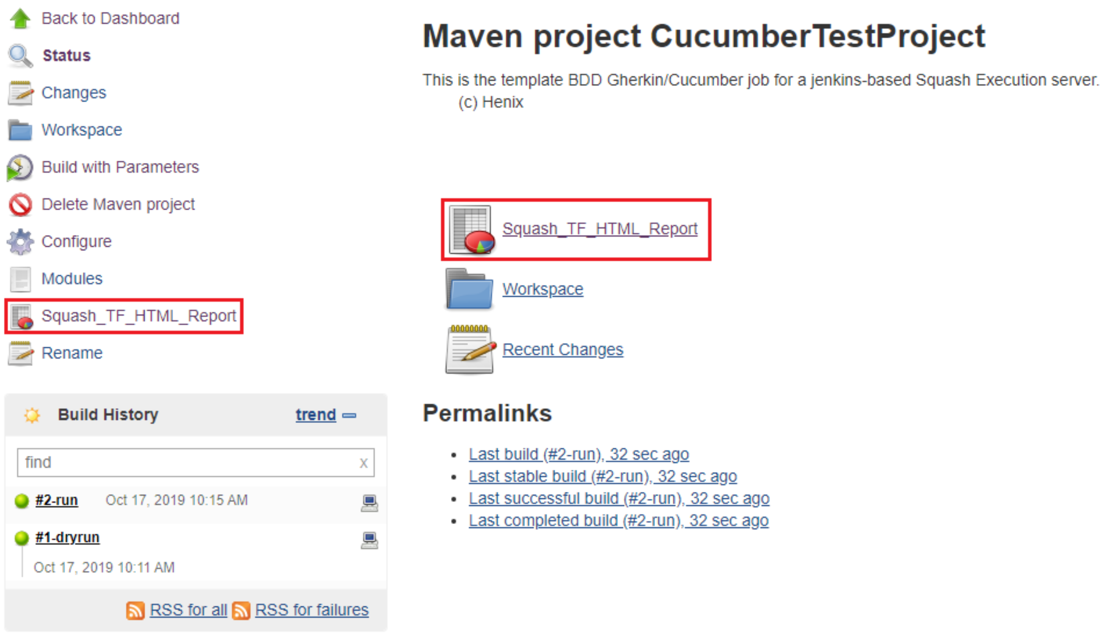
Clicking on the dot (grey, red or green depending on the status of the build) next to the build name in the *'Build History'* window will show the console output.



2.3.4 Build outputs

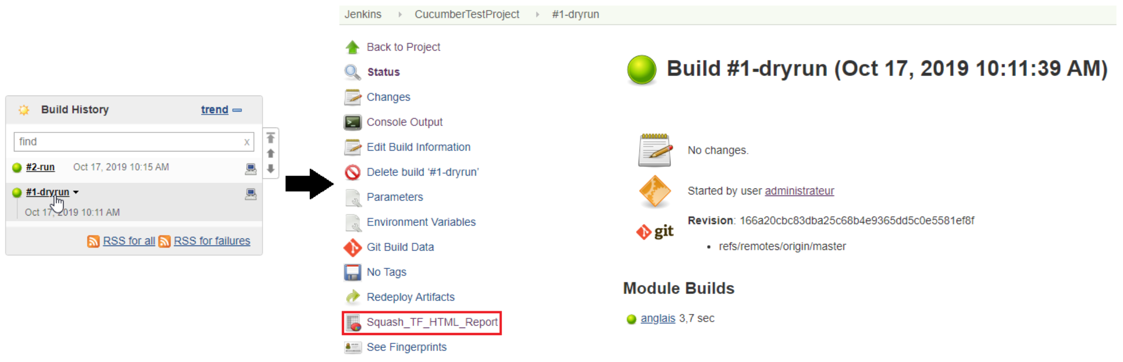
HTML report

After the first build (**dryrun** or **run**) has been performed, the generated *'Squash_TF_HTML_Report'* of the last build executed will be available on the job's page :


Back to Dashboard
Status
Changes
Workspace
Build with Parameters
Delete Maven project
Configure
Modules
Squash_TF_HTML_Report
Rename
Build History
trend
find
#2-run Oct 17, 2019 10:15 AM
#1-dryrun
Oct 17, 2019 10:11 AM
RSS for all
RSS for failures
Maven project CucumberTestProject
This is the template BDD Gherkin/Cucumber job for a jenkins-based Squash Execution server.
(c) Henix
Squash_TF_HTML_Report
Workspace
Recent Changes
Permalinks
Last build (#2-run), 32 sec ago
Last stable build (#2-run), 32 sec ago
Last successful build (#2-run), 32 sec ago
Last completed build (#2-run), 32 sec ago

The content of the HTML report will differ according to the type of build (**dryrun** or **run**) that has been executed.

If you click directly on a build's name, you'll have access to its page with the corresponding HTML report :



Note: For more indepth details about the Cucumber runner, please consult its dedicated section.

2.4 Use a job based on Robot Framework runner job template

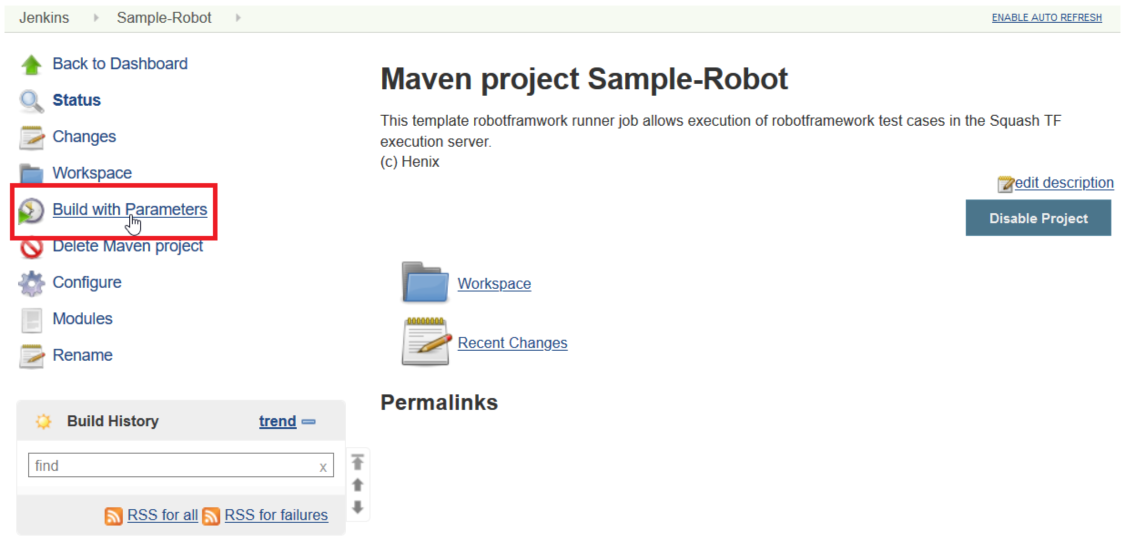
2.4.1 Execute a job

Launching a build

Once you've *created a job using the Squash TF Robot Framework job template*, select it in the list of available jobs.

On the job's page, you'll be able to launch a build, access various informations regarding the job or make modifications.

To launch a build, click on `Build with Parameters` :



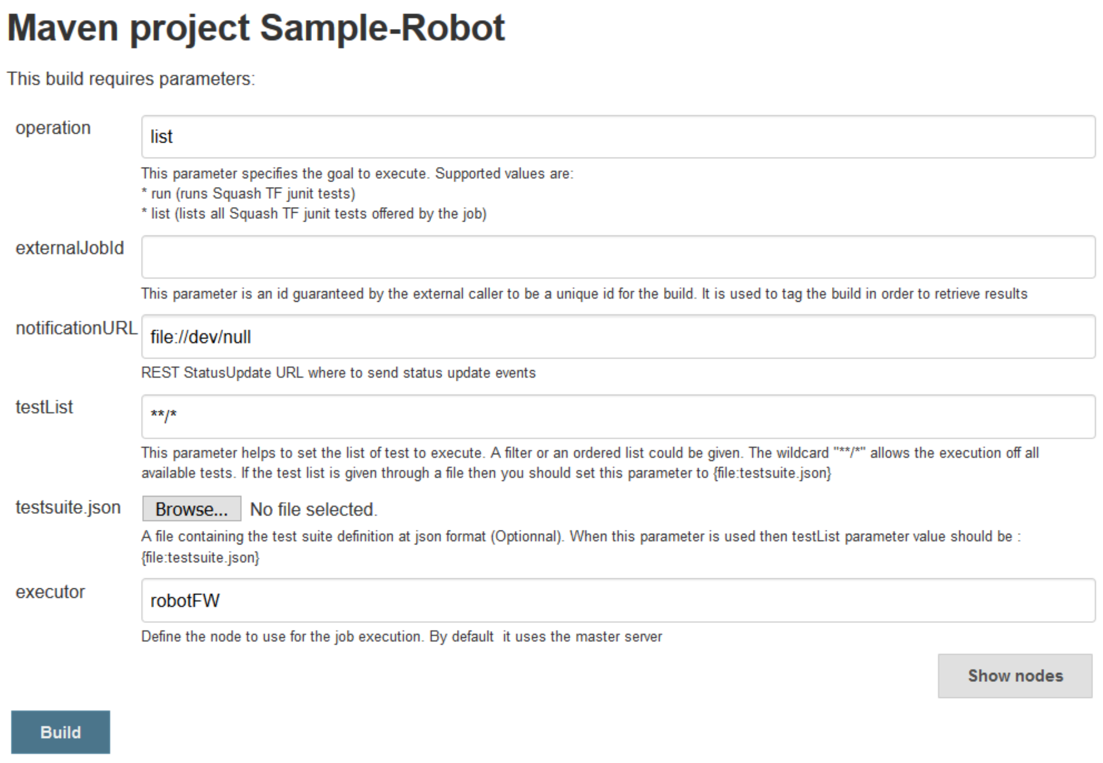
Maven project Sample-Robot

This template robotframwork runner job allows execution of robotframework test cases in the Squash TF execution server.
(c) Henix

Permalinks

Setting the parameters of the build

You can then configure some of the parameters of the build :



Maven project Sample-Robot

This build requires parameters:

operation	list
	This parameter specifies the goal to execute. Supported values are: * run (runs Squash TF junit tests) * list (lists all Squash TF junit tests offered by the job)
externalJobId	
	This parameter is an id guaranteed by the external caller to be a unique id for the build. It is used to tag the build in order to retrieve results
notificationURL	file://dev/null
	REST StatusUpdate URL where to send status update events
testList	**/*
	This parameter helps to set the list of test to execute. A filter or an ordered list could be given. The wildcard "**/*" allows the execution off all available tests. If the test list is given through a file then you should set this parameter to {file:testsuite.json}
testsuite.json	Browse... No file selected.
	A file containing the test suite definition at json format (Optionnal). When this parameter is used then testList parameter value should be : {file:testsuite.json}
executor	robotFW
	Define the node to use for the job execution. By default it uses the master server

> Operation and TestList

You can specify two types of goals to execute in the `operation` field : **list** or **run**.

- **list** : This will generate a *json* file listing all the tests present in your project. This file is used by **Squash TM** in the context of the **TM-TF** link.
- **run** : This will run all the tests specified in the `testList` field or in a user created *json* file. If you are using the **TM-TF** link, **Squash TM** will generate and transmit to **Squash TF** a *testsuite.json* file containing the list of tests to execute. In that case you don't have to alter the field `testList` or specify a *json* file.

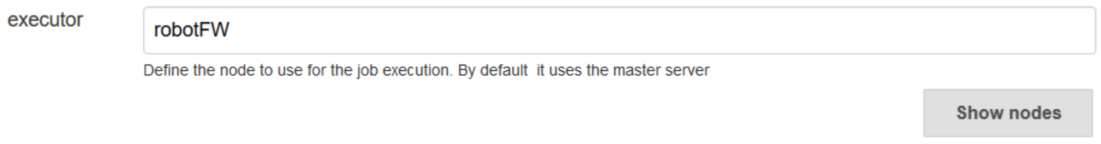
  Otherwise, if you want to provide manually to your **run** the list of tests to execute, you can procede in two ways :

  – Enter the path to the tests cases you wish to execute, separated by a semicolon (and **no space** after the semicolon), in the `testList` field. For the path, use the test suite and test case name provided in the json test list (generated by the **list** goal) to prevent problems. You can also execute all the tests present in you project by using in the `testList` field : ****/***.

  – Provide a *json* type file (by clicking on the `Choose File` button on the `testsuite.json` line) containing the list of tests you wish to execute, and fill the `testList` field with : **{file:testsuite.json}**.

> Executor

If you want to launch the build on specific **Squash TF** agent executor, enter the exact name of the agent, or its label, in the `executor` field. In this robot framework template job, a default label `robotFW` is set by default (see *Robot Framework job setup* for explanation on this default label). Click on the `Show nodes` button on the bottom right of the field to validate that you've entered a correct name :

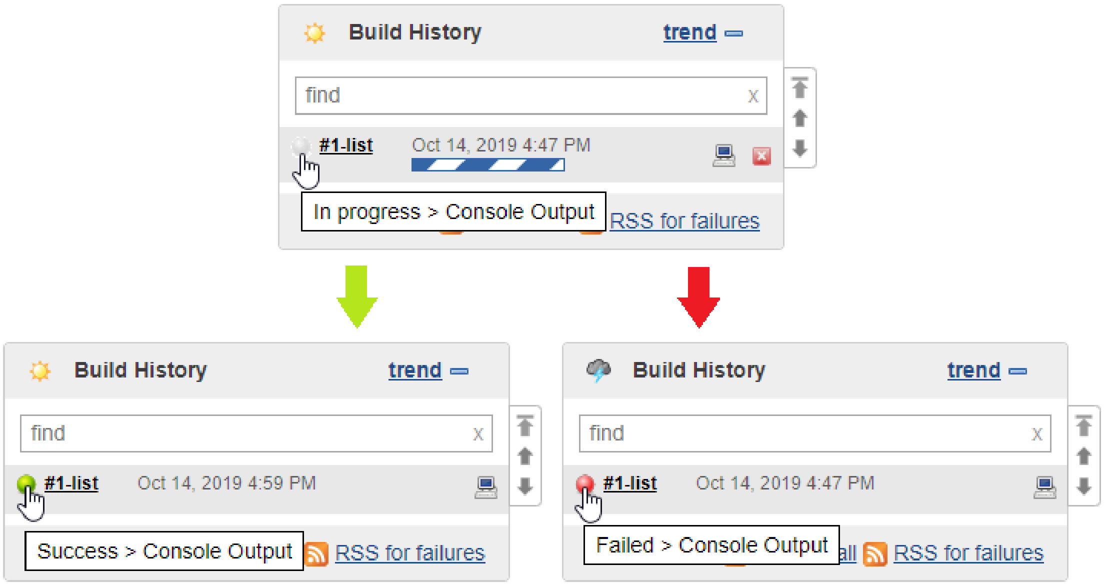

executor: robotFW

Define the node to use for the job execution. By default it uses the master server

Start the build

Once you've specified the parameters of your build, click on the *'Build'* button to launch the build.

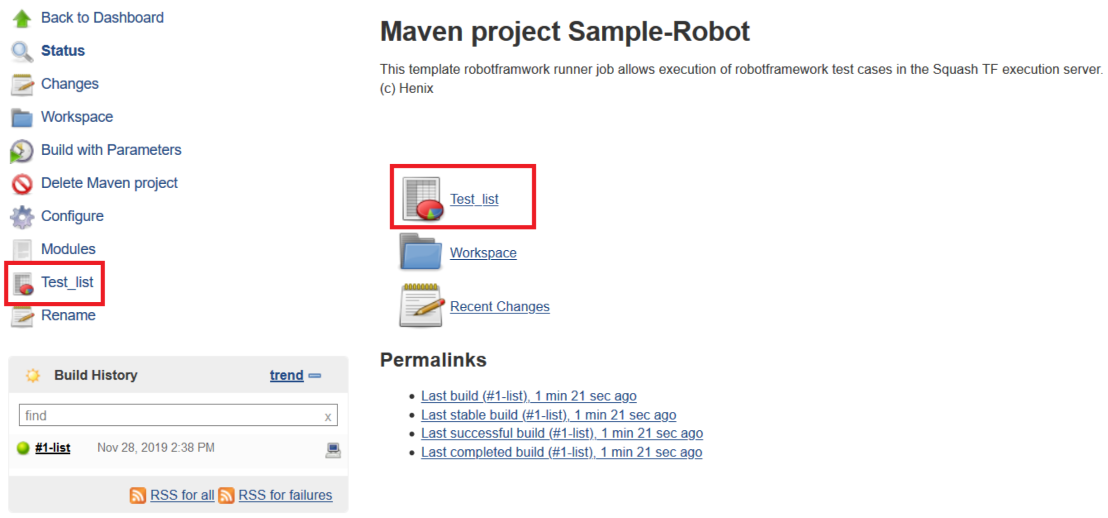
Clicking on the dot (grey, red or green depending on the status of the build) next to the build name in the *'Build History'* window will show the console output.



2.4.2 Build outputs

Test list

After the first build of type **list** has been performed, the generated **Test_list** of the last **list** build executed will be available on the job's page :



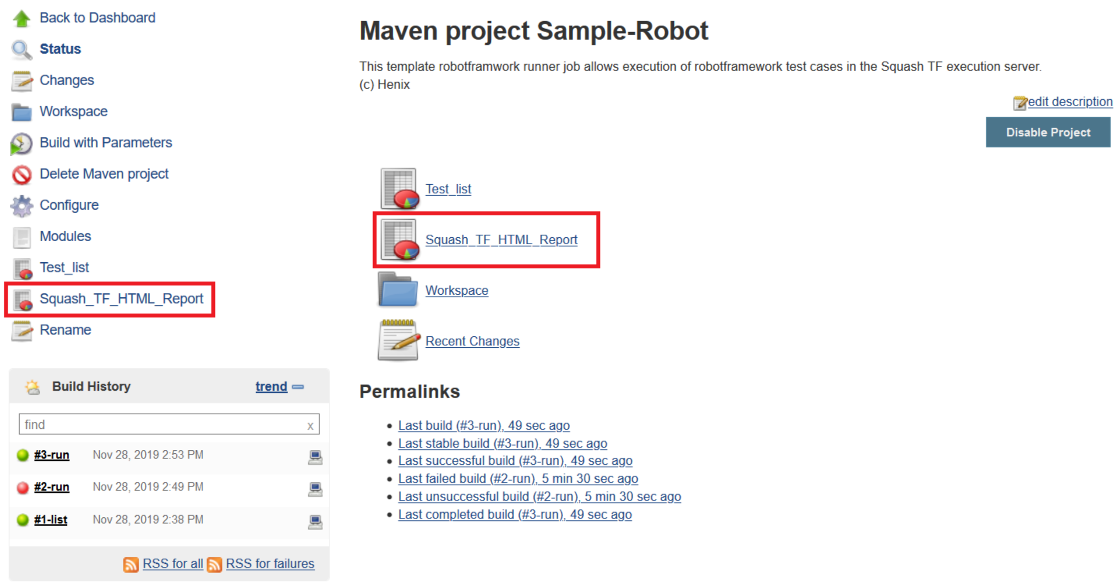
This template robotframwork runner job allows execution of robotframework test cases in the Squash TF execution server.
(c) Henix

- Last build (#1-list), 1 min 21 sec ago
- Last stable build (#1-list), 1 min 21 sec ago
- Last successful build (#1-list), 1 min 21 sec ago
- Last completed build (#1-list), 1 min 21 sec ago

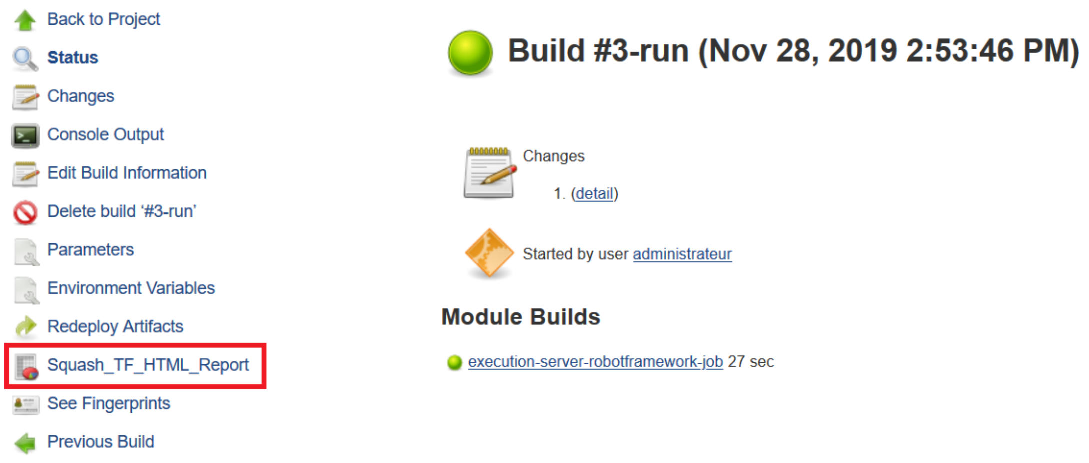
This is the *json* file fetched by **Squash TM** if using the **TM-TF** link.

HTML reports

In the same manner, the **Squash_TF_HTML_Report** generated by the last **run** build executed will be available on the job's page :


Back to Dashboard
Status
Changes
Workspace
Build with Parameters
Delete Maven project
Configure
Modules
Test_list
Squash_TF_HTML_Report
Rename
Build History
trend
find
#3-run Nov 28, 2019 2:53 PM
#2-run Nov 28, 2019 2:49 PM
#1-list Nov 28, 2019 2:38 PM
RSS for all
RSS for failures
Maven project Sample-Robot
This template robotframwork runner job allows execution of robotframework test cases in the Squash TF execution server.
(c) Henix
edit description
Disable Project
Test_list
Squash_TF_HTML_Report
Workspace
Recent Changes
Permalinks
Last build (#3-run), 49 sec ago
Last stable build (#3-run), 49 sec ago
Last successful build (#3-run), 49 sec ago
Last failed build (#2-run), 5 min 30 sec ago
Last unsuccessful build (#2-run), 5 min 30 sec ago
Last completed build (#3-run), 49 sec ago

Those run html reports are also accessible directly in the **run** build's page (which is convenient for the historization) :

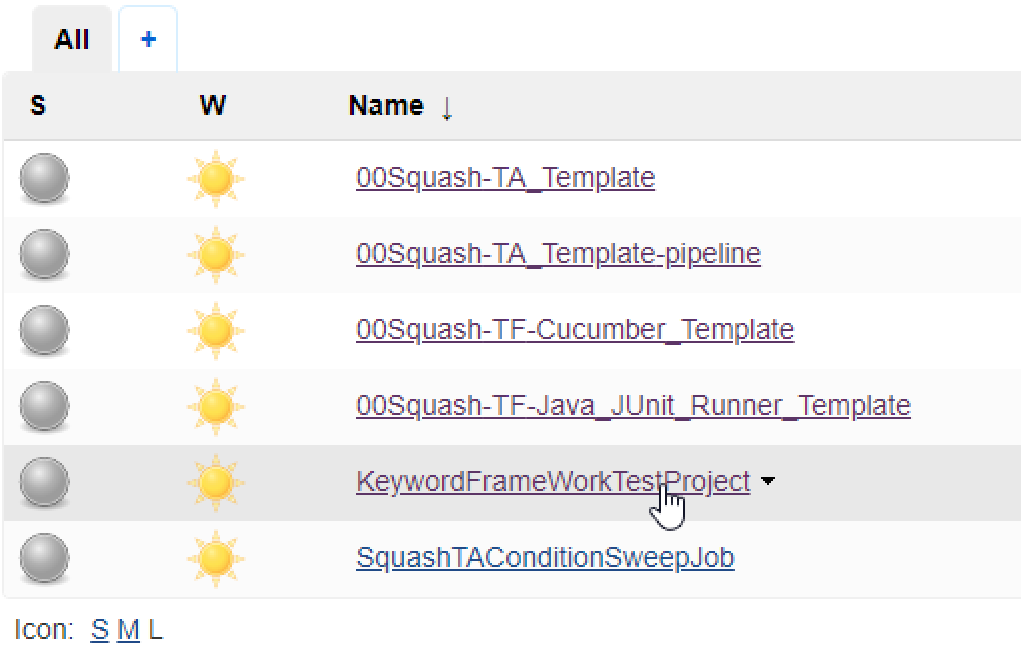


Note: For more indepth details about the Robot Framework runner, please consult its dedicated section.

2.5 Use a job based on TA / SKF job template

2.5.1 Selecting the job

Once you've *created a job using the Squash-TA Template*, select it in the list of jobs available :

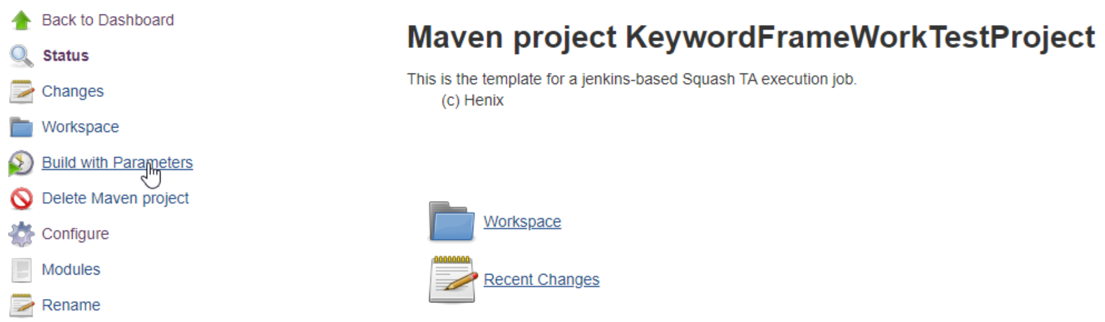


S	W	Name ↓
●	☀	00Squash-TA_Template
●	☀	00Squash-TA_Template-pipeline
●	☀	00Squash-TF-Cucumber_Template
●	☀	00Squash-TF-Java_JUnit_Runner_Template
●	☀	KeywordFrameWorkTestProject
●	☀	SquashTAConditionSweepJob

On the job's page, you'll be able to launch a build, access various informations regarding the job or make modifications.

Note: You'll have to be logged in as an administrator to modify your job (rename, delete or configure) or to launch a build.

To launch a build, click on *'Build with Parameters'* :

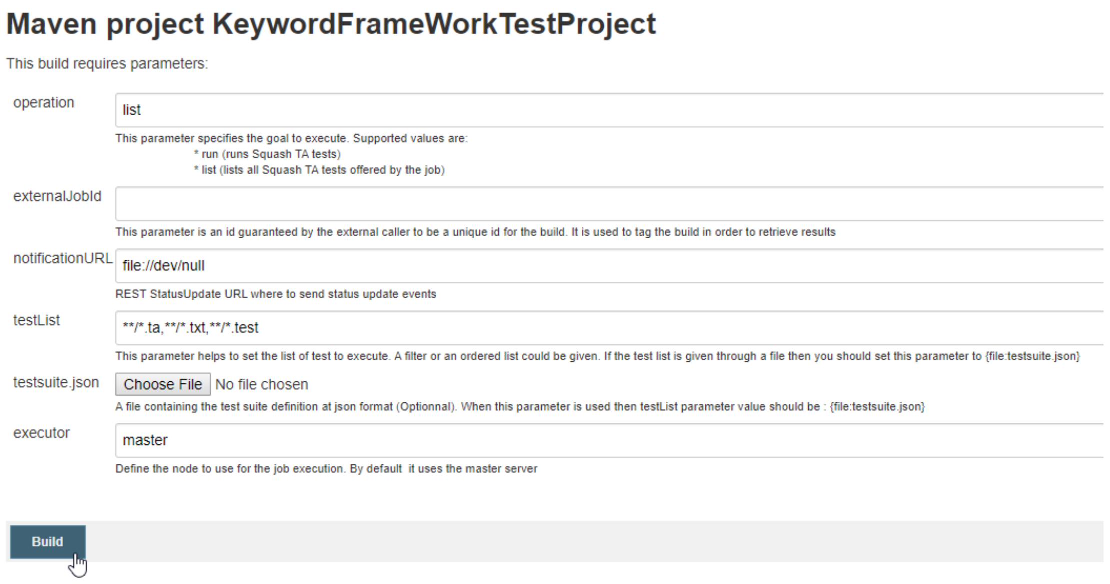


Maven project KeywordFrameWorkTestProject

This is the template for a jenkins-based Squash TA execution job.
(c) Henix

2.5.2 Setting the parameters of the build

You can then configure some of the parameters of the build :



Maven project KeywordFrameWorkTestProject

This build requires parameters:

operation	list This parameter specifies the goal to execute. Supported values are: * run (runs Squash TA tests) * list (lists all Squash TA tests offered by the job)
externalJobId	 This parameter is an id guaranteed by the external caller to be a unique id for the build. It is used to tag the build in order to retrieve results
notificationURL	file://dev/null REST StatusUpdate URL where to send status update events
testList	**/*.ta,**/*.txt,**/*.test This parameter helps to set the list of test to execute. A filter or an ordered list could be given. If the test list is given through a file then you should set this parameter to {file:testsuite.json}
testsuite.json	No file chosen A file containing the test suite definition at json format (Optionnal). When this parameter is used then testList parameter value should be : {file:testsuite.json}
executor	master Define the node to use for the job execution. By default it uses the master server

Operation and TestList

You can specify two types of goals to execute in the *'operation'* field: **list** or **run**.

> list : This will generate a *json* file listing all the tests present in your project.

This file is used by **Squash TM** in the context of the **TM-TF** link to determine the tests that can be executed.

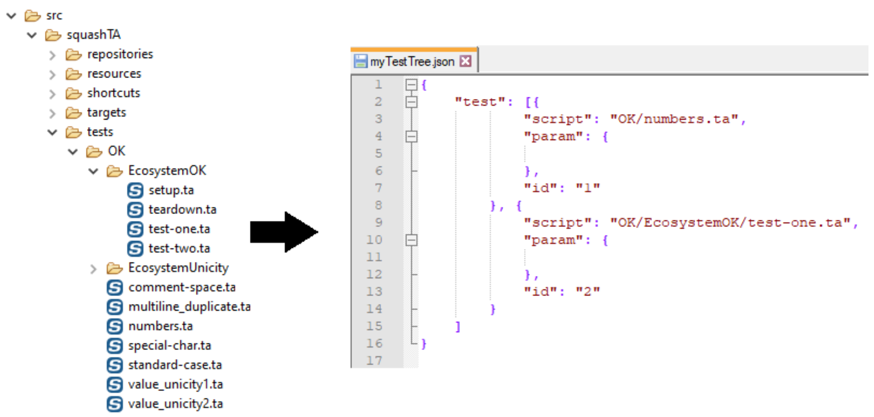
> run : This will run all the tests specified in the *'testList'* field or in a user created *json* file.

If you are using the **TM-TF** link, **Squash TM** will generate and transmit to **Squash TF** a *testsuite.json* file containing the list of tests to execute. In that case you don't have to alter the field *'testList'* or specify a *json* file.

Otherwise, if you want to provide manually to your **run** the list of tests to execute, you can procede in two ways :

- Enter the relative path (to the *'tests'* folder of your project) of the test files you wish to execute, separated by a comma (and **no space** after the comma), in the *'testList'* field. You can specify the path of a folder containing test files, using : **path/to/tests/***. You can also have the build execute all the test files (.ta, .txt or .test) in all the subfolders of the *'tests'* folder of you project, using : ****/*.{ta, txt or test}**.
- Provide a *json* type file (by clicking on the *'Choose File'* button on the *'testsuite.json'* line) containing the list of tests you wish to execute, and fill the *'testList'* field with : **{file:testsuite.json}**.

  Exemple of a user created json file :



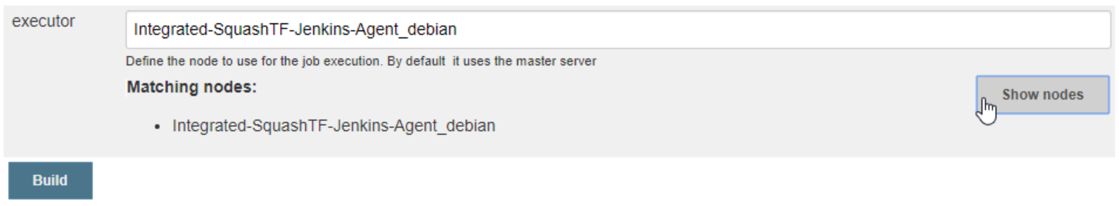
```
src
  squashTA
    repositories
    resources
    shortcuts
    targets
    tests
      OK
        EcosystemOK
          setup.ta
          teardown.ta
          test-one.ta
          test-two.ta
        EcosystemUnicity
        comment-space.ta
        multiline_duplicate.ta
        numbers.ta
        special-char.ta
        standard-case.ta
        value_unicity1.ta
        value_unicity2.ta
```

myTestTree.json

```
{
    "test": [{
        "script": "OK/numbers.ta",
        "param": {

        },
        "id": "1"
    }, {
        "script": "OK/EcosystemOK/test-one.ta",
        "param": {

        },
        "id": "2"
    }
    ]
}
```

Executor

If you want to launch the build on an **Squash TF** agent located on a distant machine and properly configured (see *this page* for the agent installation), enter the exact name of the agent, or its label, in the *'executor'* field. Click on the *'Show nodes'* button on the bottom right of the field to validate that you've entered a correct name :



executor: Integrated-SquashTF-Jenkins-Agent_debian

Define the node to use for the job execution. By default it uses the master server

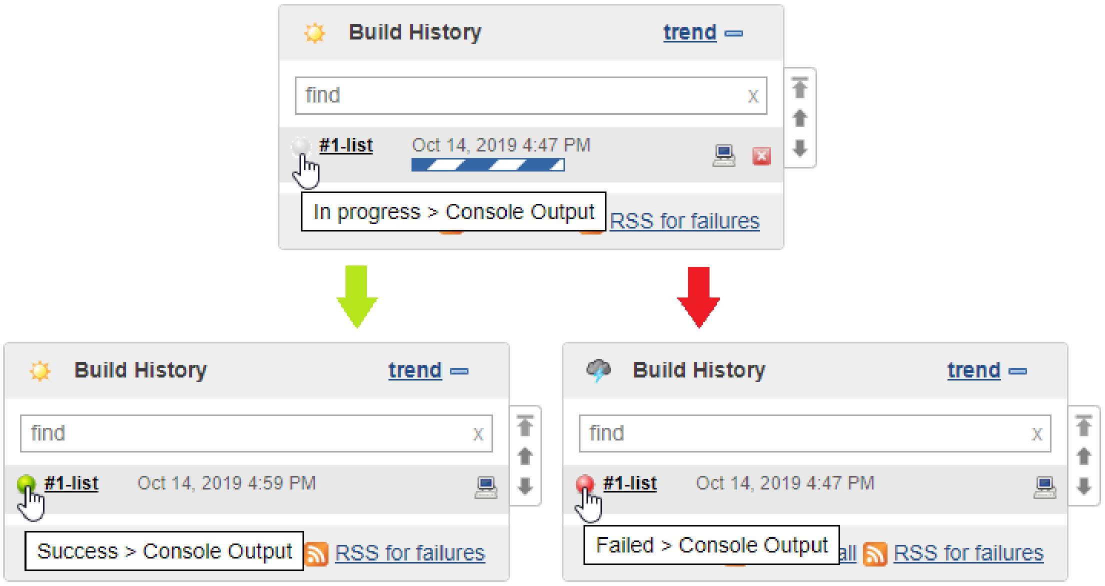
Matching nodes:

- Integrated-SquashTF-Jenkins-Agent_debian

2.5.3 Launching a build

Once you've specified the parameters of your build, click on the *'Build'* button to launch the build.

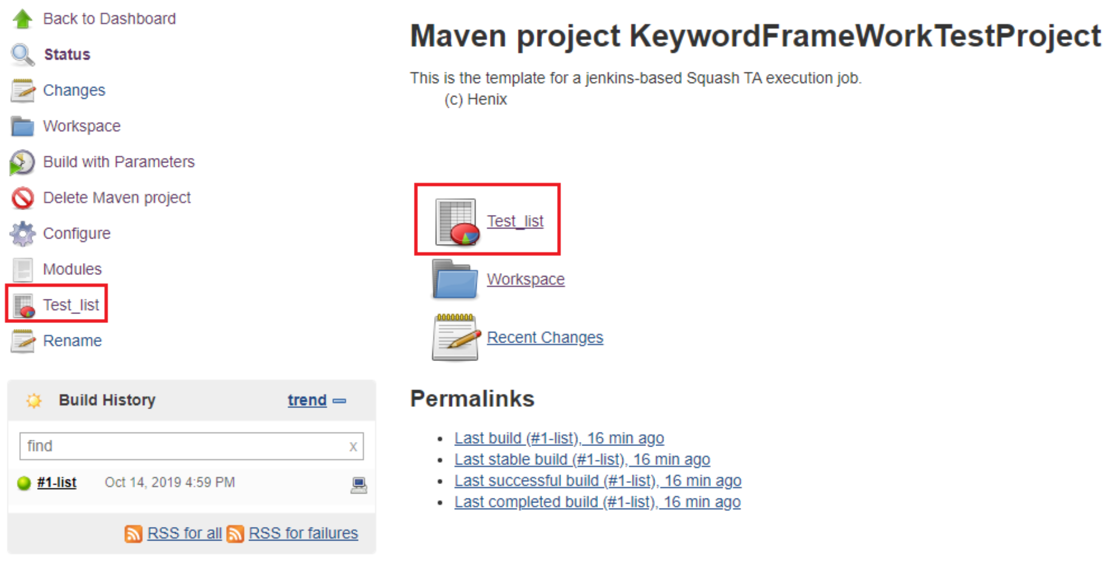
Clicking on the dot (grey, red or green depending on the status of the build) next to the build name in the *'Build History'* window will show the console output.



2.5.4 Build outputs

Test list

After the first **list** build has been performed, the generated *'Test_list'* of the last **list** build executed will be available on the job's page :



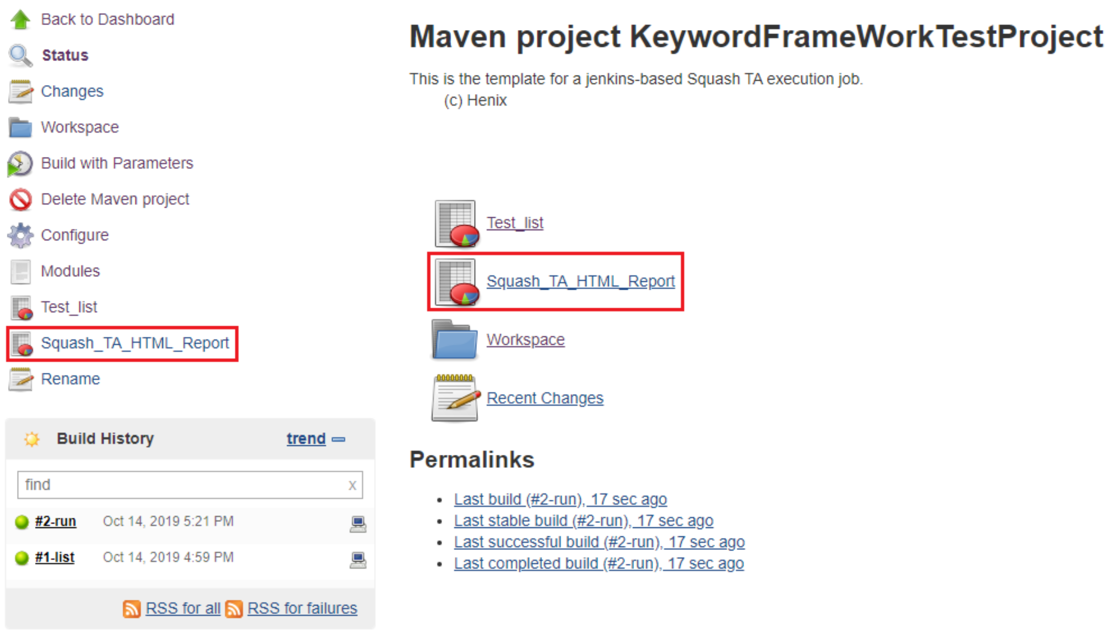
This is the template for a jenkins-based Squash TA execution job.
(c) Henix

- Last build (#1-list), 16 min ago
- Last stable build (#1-list), 16 min ago
- Last successful build (#1-list), 16 min ago
- Last completed build (#1-list), 16 min ago

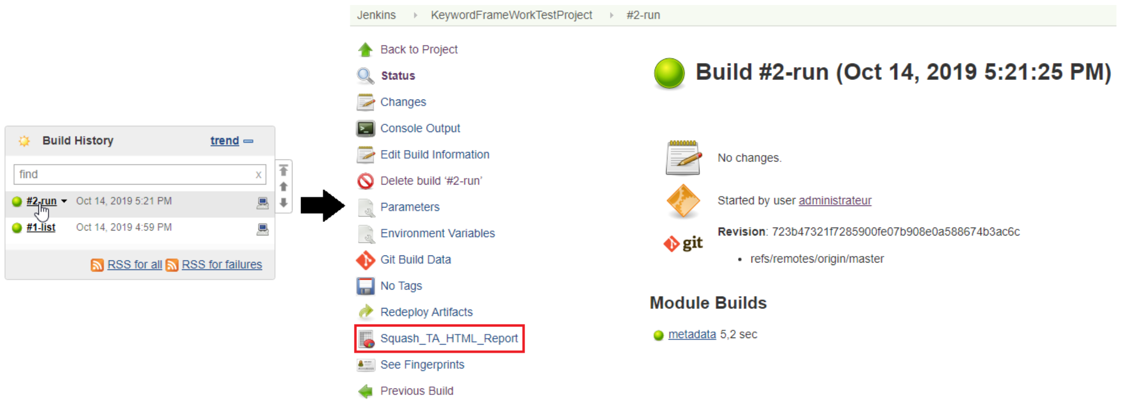
This is the *json* file fetched by **Squash TM** if using the **TM-TF** link.

HTML report

In the same manner, the generated *'Squash_TA_HTML_Report'* of the last **run** build executed will be available on the job's page :


Back to Dashboard
Status
Changes
Workspace
Build with Parameters
Delete Maven project
Configure
Modules
Test_list
Squash_TA_HTML_Report
Rename
Build History
trend
find
#2-run Oct 14, 2019 5:21 PM
#1-list Oct 14, 2019 4:59 PM
RSS for all RSS for failures
Maven project KeywordFrameWorkTestProject
This is the template for a jenkins-based Squash TA execution job.
(c) Henix
Test_list
Squash_TA_HTML_Report
Workspace
Recent Changes
Permalinks
Last build (#2-run), 17 sec ago
Last stable build (#2-run), 17 sec ago
Last successful build (#2-run), 17 sec ago
Last completed build (#2-run), 17 sec ago

If you click directly on a **run** build's name, you'll have access to its page with the corresponding **run**'s HTML report :



Note: For more indepth details about the Squash Keyword Framework, please consult its dedicated section.

Create a job

Use a job based on:

- *Java Junit runner job template*
- *Cucumber Java runner job template*
- *Robot Framework job template*
- *SKF / TA job template*

CHAPTER 3

Associate Squash TM and Squash TF

3.1 Overview

Squash TF could be used as **Squash TM** automated test execution server.

The process is as follows:

- In **Squash TM**, select a list of test case and launch the execution
- **Squash TM** ask **Squash TF** to execute a list of automated tests
- **Squash TF** execute the list of test
- **Squash TF** generate a report
- **Squash TF** send back status and report to TM
- Results are then available in TM

Of course as prerequisite, you should have created the automated tests and associate them to the **Squash TM** test cases you request the execution.

Before using this process some configuration is needed on both part

3.2 Configure the TM - TF Link

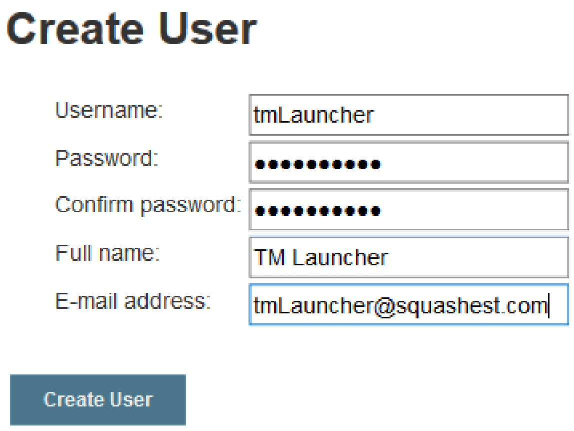
3.2.1 Configure TF

- *Create a user with the rights to launch job execution remotely*
- *Configuring Squash TM callback*
- *Configuring JENKINS_URL*

Create a user with the rights to launch job execution remotely

On **Squash TF Execution Server** you have to create a user (ex : tmLauncher) able to launch remotely the execution of the job(s) who handle the tests.

- Login with a user having admin rights on your **Squash TF Execution Server**
- Go to `Manage Jenkins \ Manage Users`
- Click on `Create User` in the left menu
- Fill the field :



Create User

Username:	tmLauncher
Password:	●●●●●●●●●●
Confirm password:	●●●●●●●●●●
Full name:	TM Launcher
E-mail address:	tmLauncher@squashest.com

- Click on the button `Create User`

Note: This user credentials will be needed when you will define this automation server in **Squash TM**. Keep them near you.

Warning: By default `Logged-in users can do anything` strategy is selected for access control in Jenkins. You may want to use another access control strategy. To do so, go to `Manage Jenkins \ Configure Global Security` section `Access Control`. In such a case, don't forget to give the rights to launch job execution remotely to the user we've just created.

Configuring Squash TM callback

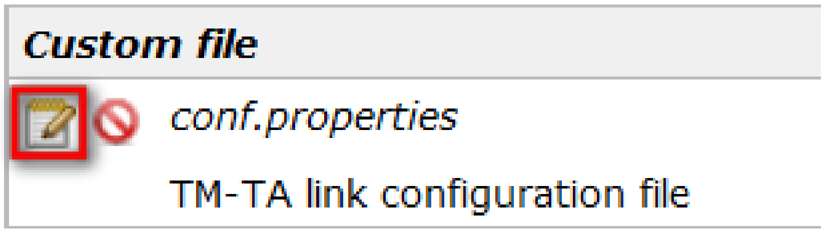
When **Squash TM** asks the execution of automated tests to **Squash TF** Execution Server, the **Squash TM** request also contains it's callback URL.

Squash TF Execution Server use this callback URL in two ways :

- As base url to send to **Squash TM** feedback about the progress of the execution
- To retrieve the right credentials to use to send feedback to **Squash TM**

On **Squash TF Server** side, the association between the callback URL and the credentials is done in a properties file. To modify this file:

- In your jenkins interface go to `Manage jenkins / Managed files`
- Edit the file taLinkConf.properties



- In this file, you should define an endpoint for each TM Server using this **Squash TF Execution Server**. For each endpoint you have to define the **Squash TM** callback URL and the credentials of a **Squash TM** User authorized to send feedback to **Squash TM**. For example a TM Server which callback URL is : http://myServer:8080/squash and the credentials are tmFeedbackLogin / tmFeedbackPassword then you should define the three properties below:

```
endpoint.1=http://myServer:8080/squash
endpoint.1.login=tmFeedbackLogin
endpoint.1.password=tmFeedbackPassword
```

Note: The credentials associated to the callback URL are those of a user define in **Squash TM**. This Squash TM user should belong to the `Test Automation Server` group in order to have right to send feedback to Squash TM. See the TM configuration page for more details.

Warning: The callbackURL define in Squash TM *configuration file* should **perfectly** match the one define in your endpoint. If you wonder why, the answer is Squash TM put the callbackURL from its configuration file in the execution request it sends to Squash TF Execution Server. And this last one, use it to retrieve the good endpoint. So if they not match, no endpoint is found.

Configuring JENKINS_URL

Squash TF provides the URL of the execution report to Squash TM. To do so, the JENKINS_URL variable should be set as follows:

- Go to “Manage Jenkins / Configure system”
- In the “Jenkins location” section, set the correct value for the “Jenkins URL” property
- Click on “Save” or “Apply”

Note: When you first start Jenkins, even if the “JENKINS_URL” field contains the right value, you have to click on “Save” or “Apply”. Otherwise the JENKINS_URL property won’t be set.

3.2.2 Configure TM

- *Add a Squash TF Execution Server to Squash TM*
- *Configure a Project to use automation*

Configure Squash TM properties

1. Open the `squash.tm.cfg.properties` file located in the conf folder of Squash TM installation folder (ex: C:\Squash-TM\conf).
2. Find the line with `tm.test.automation.server.callbackurl` and uncomment it
3. Add the url of **Squash TM** (ex: http://192.168.2.138:8080/squash). This url will be used by **Squash TF** Server to notify **Squash TM** of the execution progress.
4. Restart **Squash TM**.

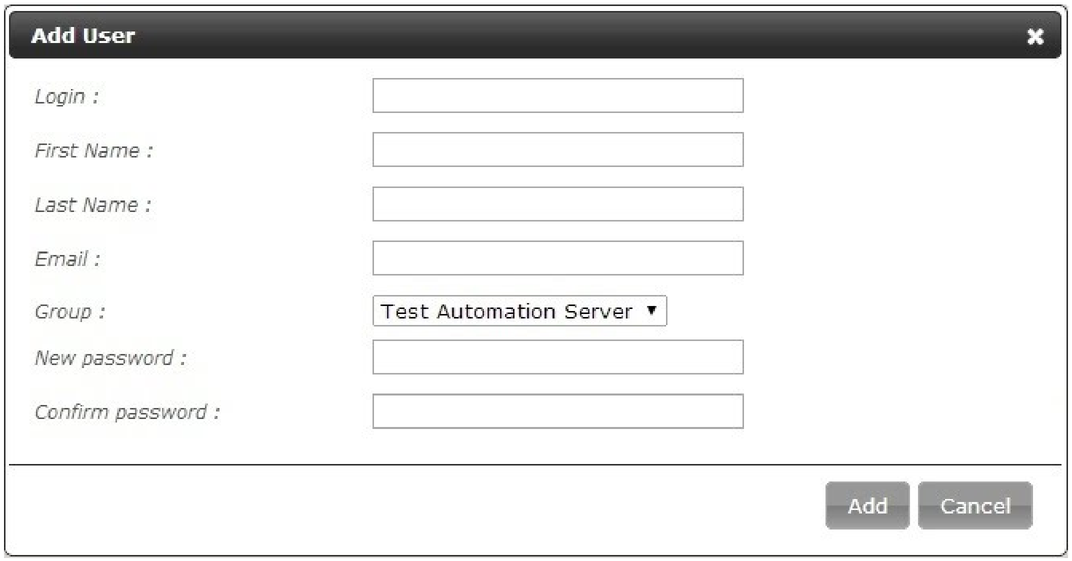
Warning: The **Squash TM** URL used in this configuration should be accessible from **Squash TF** Server location

Warning: If you update this **Squash TM** callback URL, then don't forget to update all the **Squash TF** execution server using this *endpoint*

Create a `Test Automation Server` user

In **Squash TM** there is a special user group called `Test Automation Server` which has the right to send automated test execution feedback to TM from the outside. To link a **Squash TM** server with a **Squash TF** server, you have to create a user which belong to this special group. To do so :

1. In **Squash TM**, click on the link `[Administration]` (in the upper corner) then click on `[Users]`.
2. Add a new user with the button `[Add New User]`. The `[Add User]` popup shows up :


Add User

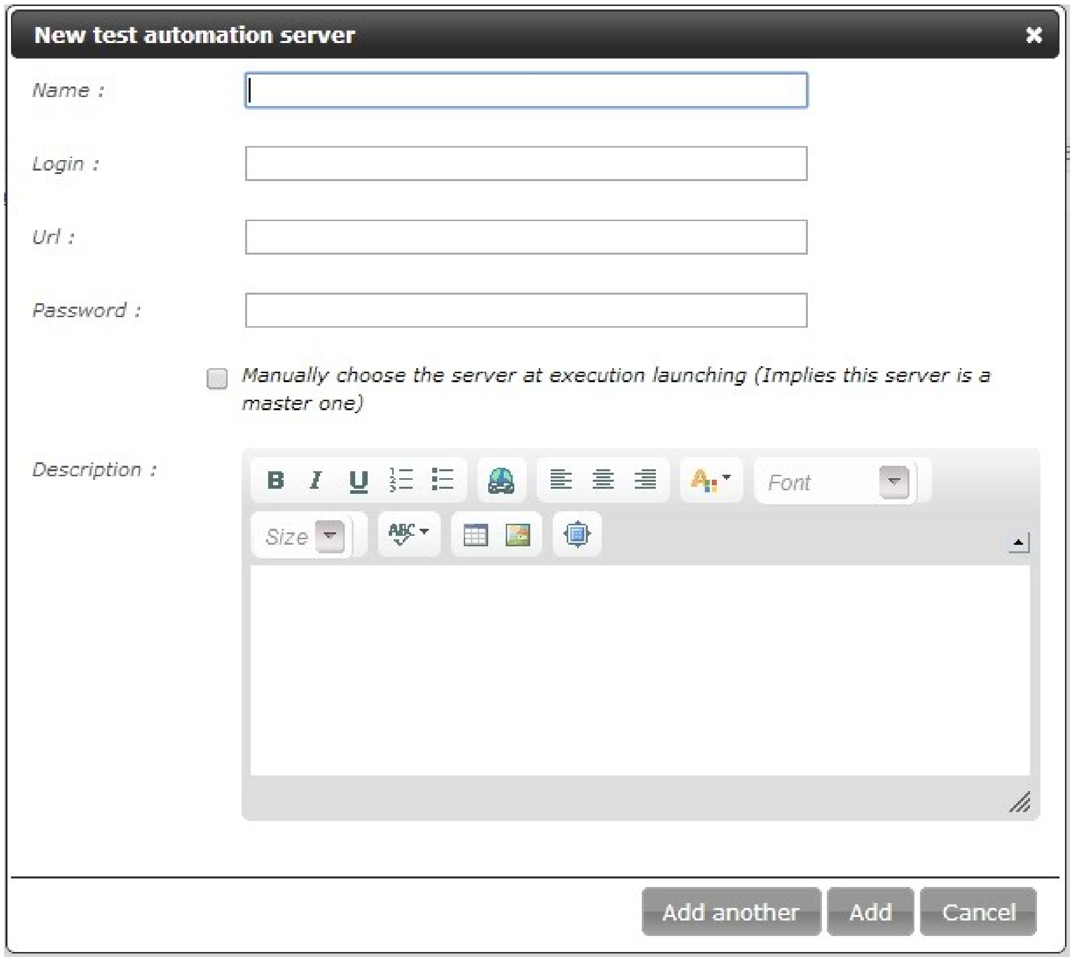
Login :	
First Name :	
Last Name :	
Email :	
Group :	Test Automation Server
New password :	
Confirm password :	

3. In the `Group` ComboBox choose `Test Automation Server`.
4. Fill the `Login` field with the login you have configured in the **Squash TF** `conf.properties` *file*.
5. Fill the `Password` field with the password you configured in the **Squash TF** `conf.properties` *file*.

Warning: The credentials of this user is used on **Squash TF** side to define the *endpoint in the conf.properties*. If you update the credentials of this user, then don't forget to update the execution server endpoint which use it.

Add a Squash TF Execution Server to Squash TM

1. In **Squash TM**, click on the link `[Administration]` (in the upper corner) then click on `Automation servers`
2. Add a new server with the button `[Add a server]`
3. The `[New test automation server]` popup shows up :



New test automation server

Name :

Login :

Url :

Password :

☐ Manually choose the server at execution launching (Implies this server is a master one)

Description :

4. Fill the `URL` field with the **Squash TF** url (ex : http://192.168.2.138:9080/jenkins).
5. Fill the `Login` field with the login of the user in **Squash TF** dedicated to automation.

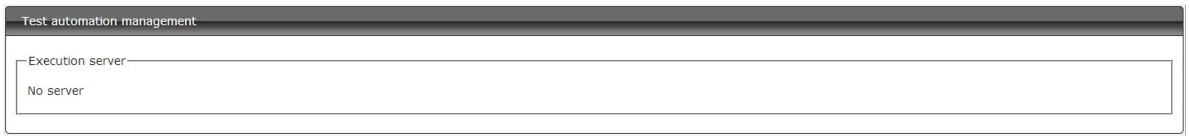
6. Fill the `Password` field with the password of the user in **Squash TF** dedicated to automation.
7. If you use a Master server with slave(s), you can check the `Manually choose the server at execution lanching (Implies this server is a master one)` checkbox. This option will allow you to choose, each time you run your tests, on which agent they run.
8. If you want to add another server, click `[Add another]` and repeat steps 4-8, otherwise click `[Add]`.

Attention: Login must be unique for each URL.

Configure a Project to use automation

> Link an Automated Server to a Project

1. Click on the link `[Administration]` (in the upper corner) then click on `[Projects]`.
2. Select an existing project, scroll down to `Test automation management`.

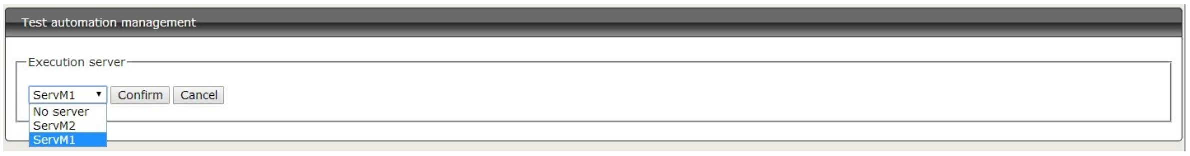


Test automation management

Execution server

No server

3. Click on `No server`. You'll see the list of available servers.



Test automation management

Execution server

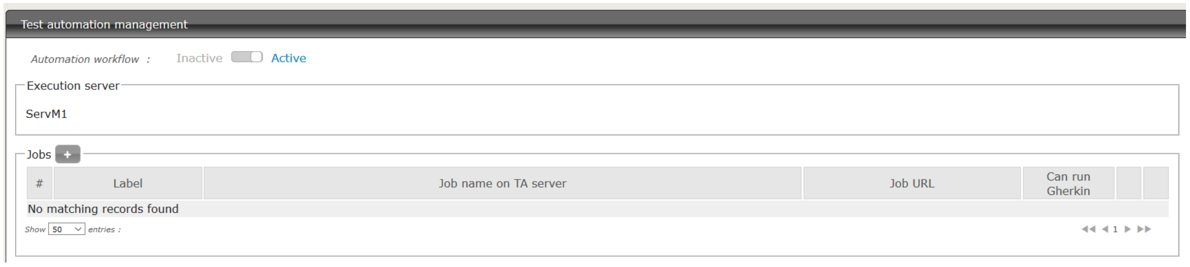
ServM1 Confirm Cancel

No server
ServM2
ServM1

4. Choose the server and `[Confirm]`.
5. In the `Test automation management` you should see a new part called `Jobs`.

> Link a Squash TF job to a Squash TM project

1. Click on the link `Administration` (in the upper corner) then click on `Projects`.
2. Select an existing project with an automation server associated, scroll down to `Test automation management`.



Test automation management

Automation workflow : Inactive Active

Execution server

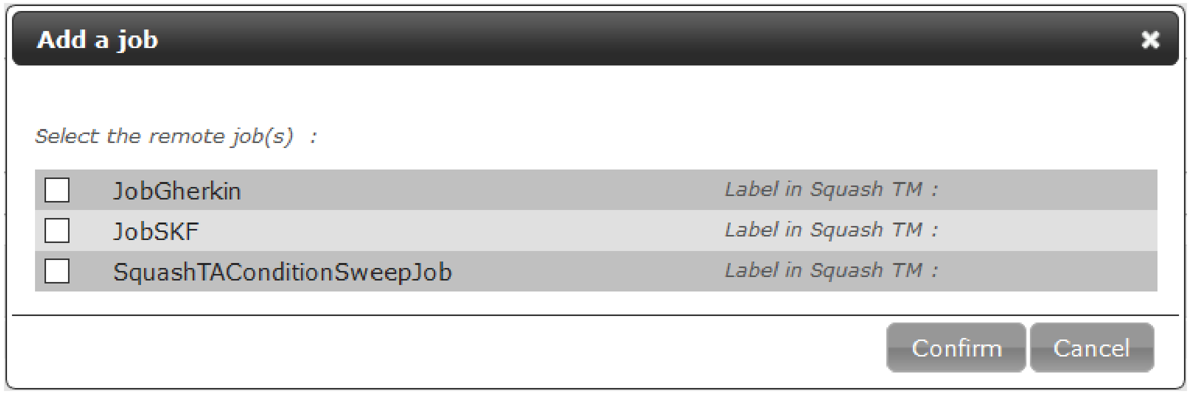
ServM1

Jobs +

#	Label	Job name on TA server	Job URL	Can run Gherkin		
No matching records found						

Show 50 entries :

3. Click on `[+]`. The New job popup will shows up with all the jobs you have in **Squash TF**.

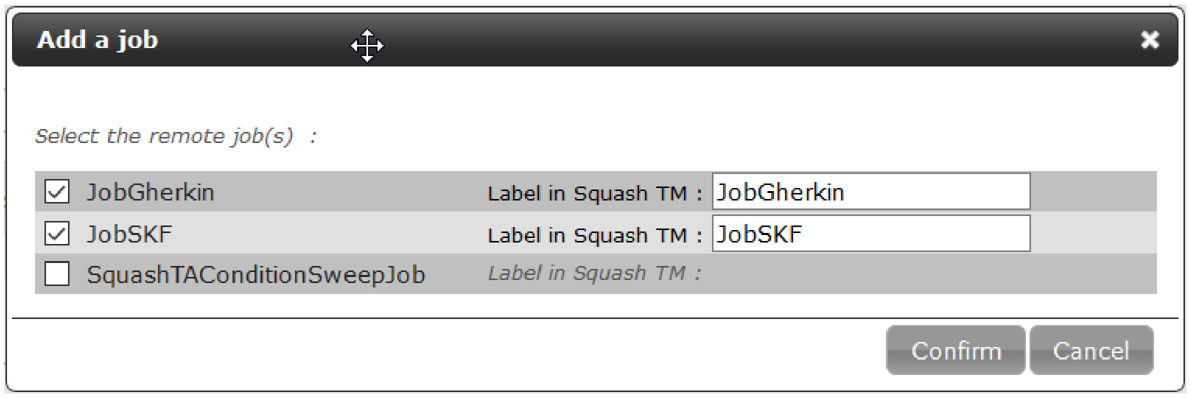


Add a job

Select the remote job(s) :

☐	JobGherkin	Label in Squash TM :
☐	JobSKF	Label in Squash TM :
☐	SquashTAConditionSweepJob	Label in Squash TM :

4. Select the job(s) you want to add. You can change their label in **Squash TM** (by default the name are the associated **Squash TF** job name).



Add a job

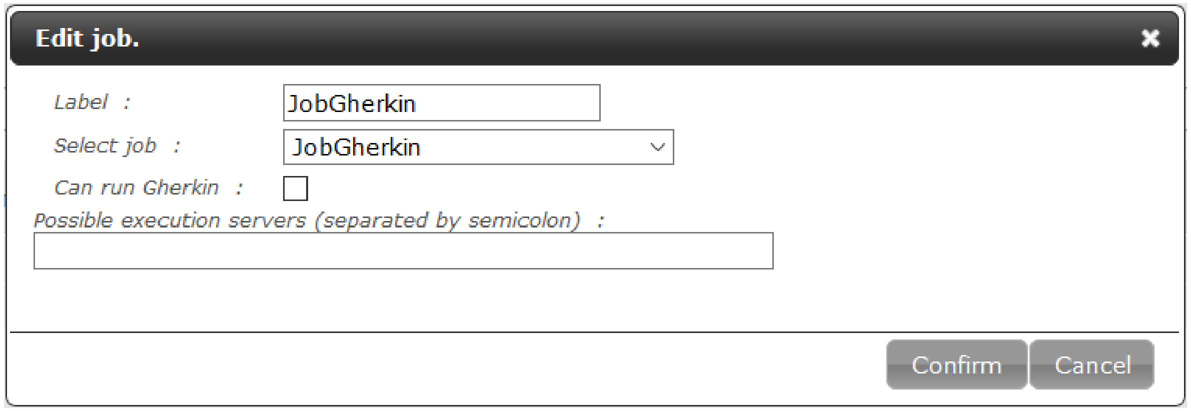
Select the remote job(s) :

✓	JobGherkin	Label in Squash TM : JobGherkin
✓	JobSKF	Label in Squash TM : JobSKF
☐	SquashTAConditionSweepJob	Label in Squash TM :

Attention: Job's name can't be blank and must be unique.

5. Click `[Confirm]`.

6. You can edit your job (pencil button) if you want to further configure it.



Edit job.

Label : JobGherkin

Select job : JobGherkin

Can run Gherkin : ☐

Possible execution servers (separated by semicolon) :

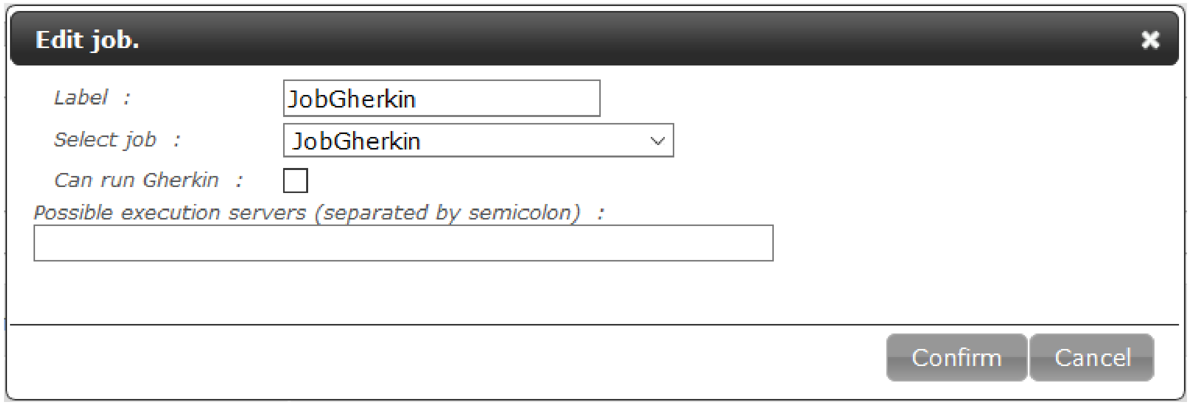
7. In the field `Possible execution servers`, you can choose which `slaves` servers can be used for this job. Type their name separated with semicolon (ex: SlaveServ1; SlaveServ2).

> **Attention:** The Possible execution servers field is only for SLAVES servers, the master will alway be displayed in the list of possible servers.

> Enable Gherkin execution for a job

When you linked a **Squash TF** job to a **Squash TM** project, by default in the job configuration in TM, the `Can run Gherkin` option is set to No. In order to this job can run gherkin you have to enable it.

1. In the project administration page section `Test automation management`, edit your job (pencil button) :



Edit job.

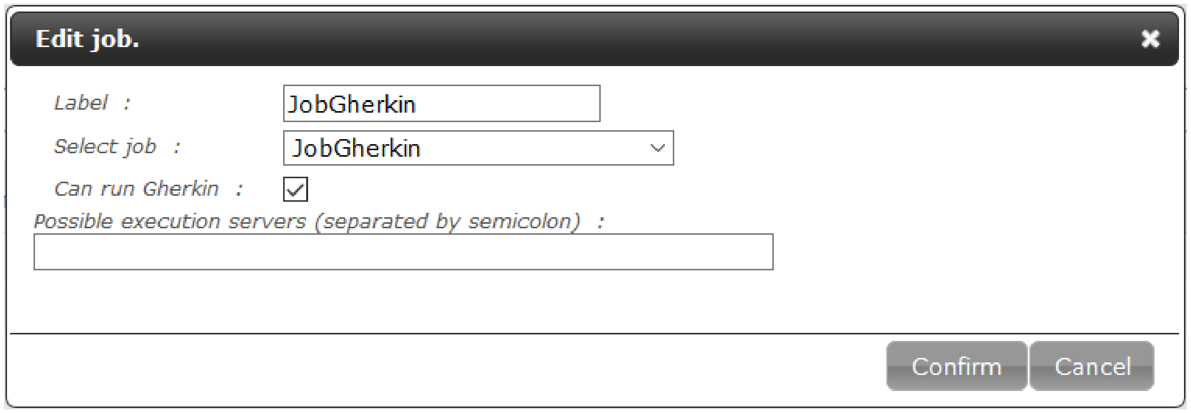
Label : JobGherkin

Select job : JobGherkin

Can run Gherkin : ☐

Possible execution servers (separated by semicolon) :

2. Check the box `Can run gherkin`



Edit job.

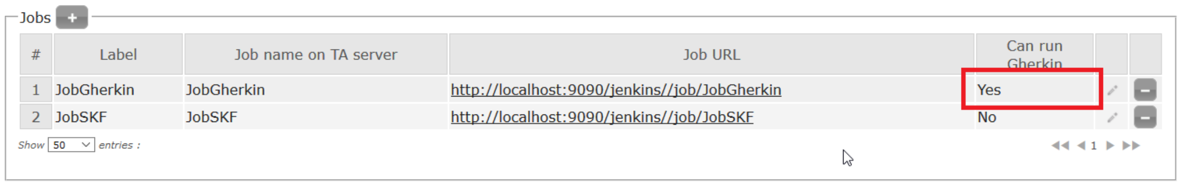
Label : JobGherkin

Select job : JobGherkin

Can run Gherkin : ✓

Possible execution servers (separated by semicolon) :

3. Click on `Confirm`



Jobs

#	Label	Job name on TA server	Job URL	Can run Gherkin		
1	JobGherkin	JobGherkin	http://localhost:9090/jenkins//job/JobGherkin	Yes		
2	JobSKF	JobSKF	http://localhost:9090/jenkins//job/JobSKF	No		

Show 50 entries :

Now the project is related to **Squash TF** server. In this project you can relate automated test script to TM test cases. You can then execute these automated tests from the campaign space and read their execution results.

- **Configure Squash TF:** *Configure TF*
- **Configure Squash TM:** *Configure TM*

CHAPTER 4

Overview

Squash TF Execution Server is our server to execute test on distributed environments

It use the master - agents architecture :

- the master: It's the scheduler which handle the jobs to execute.
- the agents: The handle execution on multiple environments or technologies:
  - environments: development, acceptance, integration, ...
  - technologies:
    - OS : windows /linux
    - Browser : chrome/firefox/...

Your jobs are on the master. We provide some sample jobs you can duplicate to create your project job according to the type of execution you want to do. See usage part of the documentation for more information on job creation. In this job we define :

- Your job parameters
- The way to retrieve your automated tests project sources
- The command line to execute your test
- The post action to publish the reports

Squash TF is based on jenkins which bring us all its ecosystem capacity:

- scm connectors (git, mercurial, svn, ...) to retrieve tests project sources
- distributed execution through master agent architecture
- pipeline
- report publishing
- API Rest for job remote launch