

March, 2020

An IBM TechDoc from

IBM Z DevOps Acceleration Program

Setting up the CI Pipeline Agent on IBM Z as a Started Task

Security Setup for CI Slaves on IBM Z

Dennis Behm

dennis.behm@de.ibm.com

Mathieu Dalbin

mathieu.dalbin@fr.ibm.com

Abstract

Learn how to setup a CI pipeline agent as a Started Task. Sample RACF definitions provided to setup the necessary requirements to launch the agent.

DocID: TR109995 / Release 1.0

Table of Contents

Introduction.....	3
Triggering builds as a CI pipeline user	3
Setting up a technical user	4
Running the Jenkins agent as a Started Task.....	5
Two methods to launch the Jenkins agent	5
Configuring the inbound agent method for the agent.....	5
1. Defining the agent	5
2. Defining the TCP port	6
3. Retrieving the launch commands for the agent	6
4. Downloading the agent binary and transferring it to USS.....	6
5. Setting up RACF Security	7
6. Defining the STC procedure	8
7. Customizing USS and WLM parameters	9
8. Starting the agent	10

Introduction

Several CI orchestrators, like Jenkins, or TeamCity, follow a server / agent architecture. The agent (also called the slave) runs on the target build machines. Some orchestrators, such as Jenkins, support z/OS as a target environment for the agent. In this document, we focus on z/OS and use Jenkins in the text as an example.

In a standard setup, the agent is launched from the master (also called the server) through SSH. Such a launch method is convenient as it is dynamic and does not require specific installation steps on the target environment. However, with this standard setup to launch the agent, the task is not easily identifiable within SDSF (System Display and Search Facility). Additionally, it appears as a Unix subtask of the user, which requires the TSO user ID and password stored on the Jenkins server to launch the agent.

Besides the standard setup, the agent can be launched as a Started Task (STC), which does not require SSH access. Therefore, credentials are not stored on Jenkins. In addition, the process can be embedded to the z/OS system automation.

This TechDoc outlines the required definitions to launch the agent as a Started Task (STC).

Triggering builds as a CI pipeline user

The pipeline is an automated process. The best practice is to create a CI pipeline user to drive the pipeline build on IBM Z as follows:

- It is expected that the CI pipeline jobs runs under a dedicated technical user ID.
- The agent has exclusive R+W permissions on the build workspaces in Unix System Services and the target build data sets. It is recommended that developers only have read permission on build workspaces and the build data sets.
- The agent performs all the necessary steps of the pipeline (building, code inspection, unit testing, packaging) under the authority of the technical user ID that launched it.
- It is recommended to use a dedicated HFS file system to manage the build workspaces to avoid impacting other activities on the system.

Setting up a technical user

This section provides a generic RACF definition for the technical user for the CI agent.

Environment prerequisites for the CI pipeline user are as follows:

- Minimum of 512 MB region size, 1 GB recommended.
- OMVS Segment

```
ADDGROUP STCGROUP OMVS(GID(#group-id)) DATA('GROUP WITH OMVS SEGMENT FOR
STARTED TASKS')

ADDUSER CIUSER DFLTGROUP(STCGROUP) NOPASSWORD NAME('CI AGENT')
OMVS(UID(#user-id) HOME(#homeDirectory) PROGRAM(/bin/sh)) DATA('CI AGENT')

RDEFINE STARTED CIAGENT.* DATA('CI BUILD AGENT') STDATA(USER(CIUSER)
GROUP(STCGROUP) TRUSTED(NO))

SETROPTS RACLIST(STARTED) REFRESH
```

Comment: Ensure that the started task user ID is protected by specifying the NOPASSWORD keyword.

The provided generic RACF definitions should be tailored to meet the installation's requirements, by defining specific value for attributes **#group-id**, **#user-id** and **#homeDirectory**.

To be able to override jobnames of forked USS address spaces, the CIUSER requires READ permission to the BPX.JOBNAME resource in class FACILITY. Please see the restrictions for _BPX environment variables, so that _BPX_JOBNAME can be applied. Before the job name can be changed, the invoker must have appropriate privileges. The privileges include either superuser authority or READ permission to the BPX.JOBNAME FACILITY class profile. See details in IBM Knowledge Center:

https://www.ibm.com/support/knowledgecenter/SSLTBW_2.4.0/com.ibm.zos.v2r4.bpxb200/bpxenv.htm

The following RACF commands provide this privilege to the CIUSER:

```
PERMIT BPX.JOBNAME CLASS(FACILITY) ID(CIUSER) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Running the Jenkins agent as a Started Task

This section describes how to run the Jenkins agent on z/OS (USS) as a Started Task.

Two methods to launch the Jenkins agent

Jenkins offers two methods to start the agent:

- The Jenkins master establishes a connection to the target z/OS environment through SSH. This requires that the user ID and password are stored on the Jenkins master. Binaries are transferred to the target environment and the agent is started. This is the standard setup.
- Alternatively, the agent can be installed on the target z/OS environment and then establishes a connection to the Jenkins master. It does not require SSH access and credentials are not stored on the Jenkins master. This option is known as **launching an inbound agent**. For more information, see <https://github.com/jenkinsci/remoting/blob/master/docs/inbound-agent.md>

To run the Jenkins agent as a Started Task, it must be launched with the inbound agent method.

Configuring the inbound agent method for the agent

The following procedure describes how to set up a Jenkins inbound agent and configure it in z/OS.

1. Defining the agent

Logon to the Jenkins master and define a new agent. Set the launch method to “Launch agent by connecting it to the master” and uncheck the “Use WebSocket”.

Disabling the use of WebSocket requires to define a TCP Port, on which Jenkins Server will listen for incoming connections, which we will do in the next step.



Launch method: Launch agent by connecting it to the master

☐ Disable WorkDir

Custom WorkDir path:

Internal data directory: remoting

☐ Fail if workspace is missing

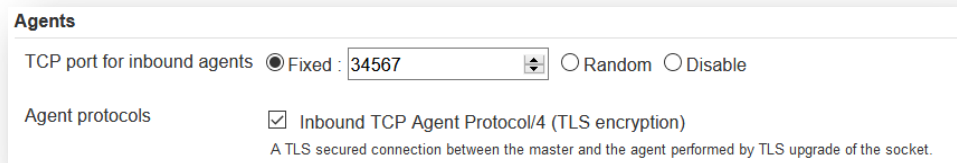
☐ Use WebSocket

Warning: Please avoid the use of WebSockets. A loop has been detected in the implementation of the WebSocket on z/OS JVM, **which causes high CPU consumption**.

2. Defining the TCP port

To define the incoming TCP Port where Jenkins will listen, go to the Jenkins's Global Security Configuration (*Manage Jenkins --> Configure Global Security*).

In the *Agents* section, pick an available TCP port and check the use of TLS encryption:



Agents

TCP port for inbound agents ☒ Fixed : ☐ Random ☐ Disable


Agent protocols ☒ Inbound TCP Agent Protocol/4 (TLS encryption)
A TLS secured connection between the master and the agent performed by TLS upgrade of the socket.

3. Retrieving the launch commands for the agent

Return to the agent configuration page and retrieve the *launch command* provided by Jenkins to start the agent on the build machine. The commands include a secret and all necessary parameters.



Connect agent to Jenkins one of these ways:

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

```
java -jar agent.jar -jnlpUrl http://10.3.20.156:8080/computer/E0LE53STC/slave-agent.jnlp -secret 9107e7f943f01cc0b88d636342bfd9cd273decf1641cc5cce69d76a75d2e1af6 -workDir "/var/jenkins"
```

Run from agent command line, with the secret stored in a file:

```
echo 9107e7f943f01cc0b88d636342bfd9cd273decf1641cc5cce69d76a75d2e1af6 > secret-file
java -jar agent.jar -jnlpUrl http://10.3.20.156:8080/computer/E0LE53STC/slave-agent.jnlp -secret @secret-file -workDir "/var/jenkins"
```

4. Downloading the agent binary and transferring it to USS

From the agent configuration page, download the .jar file, which might be shown as agent.jar or remoting.jar, and transfer it to the work directory in USS.

The agent JAR file must be present prior to starting the STC.

5. Setting up RACF Security

To comply with standard setup of z/OS Security Server depending on your installation and security requirements, define a RACF user to run the Started Task.

The following RACF commands define a specific group and a user 'JENKINS', with NOPASSWORD (to prevent this user to log on TSO), and associate this user to a resource of the STARTED class:

```
ADDGROUP JENKINS OMVS(GID(14001)) DATA('Group for Jenkins agents')

ADDUSER JENKINS DFLTGRP(JENKINS) NOPASSWORD NAME('Jenkins agent')
OMVS(UID(14001) HOME(/u/jenkins) PROGRAM(/bin/sh)) DATA('Jenkins Agent')

RDEFINE STARTED JENKINS.* DATA('Jenkins agent STC') STDATA(USER(JENKINS)
GROUP(JENKINS) TRUSTED(NO))

SETROPTS RACLIST(STARTED) REFRESH
```

This user has its home directory in /u/jenkins, but all the files retrieved through the Jenkins agent will be located in the work directory on USS. This work directory should be defined before the agent accesses it. Also, ownership of the home directory and the work directory should be given to the 'jenkins' user defined through the Security Server commands shown above. Other users might need to be given read access to this work directory to check build results and logs of the CI pipeline.

6. Defining the STC procedure

Define a STC PROC in your preferred library of the PROCLIB concatenation. The EXEC statement is a call to BPXBATCH, which will in turn call your tailored shell script.

```
//JENKINS PROC
//*****
//RUN      EXEC PGM=BPXBATCH,DYNAMNBR=30,REGION=0M,TIME=1440,
//      PARM='SH /u/jenkins/start.sh'
//STDOUT   DD PATH='/u/jenkins/stdout',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=SIRWXU
//STDERR   DD PATH='/u/jenkins/stderr',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=SIRWXU
//STDENV   DD *
__BPX_JOBNAME=JENAGNT
__BPX_SHAREAS=YES
/*
```

Next, create the launch shell script `/u/jenkins/start.sh`. The following sample script `start.sh` contains the necessary commands to set up environment variables, including the launch command provided by Jenkins master to start the agent:

```
#!/bin/sh

. /usr/lpp/dbb/v1r0/conf/gitenv.sh
. /usr/lpp/dbb/v1r0/conf/dbbenv.sh
export JAVA_HOME=/usr/lpp/java/J8.0_64/
export IBM_JAVA_ENABLE_ASCII_FILETAG=ON
cd /var/jenkins

curl --output remoting.jar http://10.3.20.156:8080/jnlpJars/agent.jar
/usr/lpp/java/J8.0_64/bin/java -Dfile.encoding=ISO-8859-1 -
Xnoargsconversion -jar /var/jenkins/remoting.jar -jnlpUrl
http://10.3.20.156:8080/computer/EOLE53STC/slave-agent.jnlp -secret
9107e7f943f01cc0b88d636342bfd9cd273decf1641cc5cce69d76a75d2elaf6 -workDir
"/var/jenkins" -text
```

Tailor this shell script to add specific configurations, for instance invocations of additional scripts or defining new environment variables.

To download the latest version of the Jenkins agent from the Jenkins server, a curl command can be used, as shown in the above start.sh script. Curl is available as part of Rocket Open Source Languages and Tools for z/OS offering and should be available on the target system. Using this curl command in the startup script ensures compatibility between the Jenkins server and the Jenkins agent, which is downloaded every time the STC is recycled.

7. Customizing USS and WLM parameters

The build agent is expected to run with a high priority, as the triggering of builds is a critical part of a CI pipeline on mainframe. Consider setting the appropriate WLM settings for the STC.

Using BPXBATCH to launch the start script of the Jenkins Agent in USS will create a new address space, which uses the WLM OMVS classification rules. To facilitate the classification of jobs in WLM, two environment variables can be used: `_BPX_JOBNAME` to specify the name of the process, and `_BPX_ACCT_DATA` to specify the accounting information of the process. In the sample JENKINS PROC listed above, the `_BPX_JOBNAME` for the new forked process is specified in the inline STDENV DD card and has the value JENAGNT.

In this configuration, jobname of all subsequent forks created by the Jenkins agent will start with JENAGNT prefix, followed by a number from 1 to 9.

To identify the forked processes of the Jenkins agent and to relate them to Jenkins jobs, the `_BPX_JOBNAME` environment variable can be used. This can be done in the Jenkins pipeline definition by providing a value to the `env._BPX_JOBNAME` variable:

```
env._BPX_JOBNAME = "JENJOB"
```

Additionally, consider setting the `_BPX_JOBNAME` environment variable in groovyz script to highlight the Dependency-Based Build operations in the Jenkins pipeline:

```
export _BPX_JOBNAME=JENBUILD
```

Optionally, set the environment variable `_BPX_SHAREAS` to YES, which allows the sharing of address space for forked USS processes. Please be aware that the execution of groovyz will set `_BPX_SHAREAS` for this address space to NO, in order to support Multi-threaded MVSJob operations in subsequent forked processes.

See below the results of this customization as displayed in SDSF, during the execution of a Jenkins pipeline building a z/OS application with Dependency-Based Build:

JOBNAME	Workload	SrvClass
JENAGNT	OMVS_WKL	UNIX
JENBUILD	OMVS_WKL	UNIX
JENJOB	OMVS_WKL	UNIX
JENJOB	OMVS_WKL	UNIX
JENJOB	OMVS_WKL	UNIX
JENJOB1	OMVS_WKL	UNIX
JENJOB2	OMVS_WKL	UNIX
JENJOB3	OMVS_WKL	UNIX
JENKINS	STC_WKL	OPSDEF

In this configuration, JENKINS is the initial Started Task, JENAGNT is the address space executing the Jenkins agent, JENJOB address spaces are executing tasks of the Jenkins pipeline and JENBUILD is executing the groovy script corresponding to the Dependency-Based Build operation.

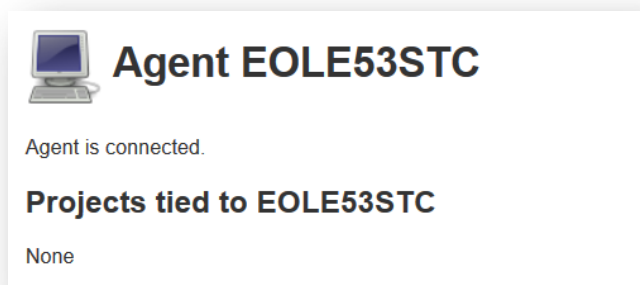
To meet with your WLM specifications, _BPX_JOBNAME and _BPX_ACCT_DATA can be used to filter and define WLM policies and goals.

8. Starting the agent

Start the STC by issuing the MVS START command: **/S JENKINS** command:

JENKINS	JENKINS	RUN	STC07054	STC	L0	FF	315	0.00	0.00
JENKINS1	*OMVSEX		STC07055	STC	L0	FF	366	0.00	0.00
JENKINS2	STEP1		STC07056	STC	L0	FF	2537	0.00	0.00
JENKINS3	*OMVSEX		STC07057	STC	IN	F0	21T	0.00	0.00

Within the Jenkins user interface, the agent is now listed as connected in the Jenkins master.



The agent can now be used in subsequent Jenkins pipelines, to perform actions on z/OS USS, such as Git operations, DBB builds, and shell script invocations.

To stop the agent, just cancel the Started Task.