Jenkins agent setup

Introduction

This article describes one of the most common ways to setup Jenkins agent and connect it to Jenkins controller (master).

Setup options

- Launch agent by connecting it to the controller
    - registering agent on controller and then connecting that agent to controller
    - very popular in enterprises when complex network setups are present and one way connectivity does not allow population of agent via SSH, but actually agent must connect to controller
        - We have article describing most common problem of enterprise setups - Executing Jenkins jobs when only one way network connection exists
    - recommended for TDS users
    - detailed steps are described in chapter DeploymentofJenkinsagentandconnectingtocontroller
- Launch agents via SSH
    - connecting actively to agent and deploying agent there
    - requires admin credentials and privileges into agent system
    - it is OK for various scenarios, but other approaches are recommended
- Docker variant of Launch agents via SSH with SSH key injection
    - connecting actively to agent and deploying agent there, in this case using Docker
    - requires sufficient credentials and privileges in agent system to deploy Docker containers
- Others
    - there are other options potentially available thanks to plugins that can be installed.

Deployment of Jenkins agent and connecting to controller

Requirements

Network

- firewall opening for port 9000/tcp from relevant source agent(s) IP(s) or network segment/pool towards internet in general (sometimes referred as destination)
    - network layer (using TDS portal network functionality)
    - operating system layer (firewalld if needed)

Jenkins controller running in public

- Decide which port you are going to use as we use fixed setup in our case. We are choosing 9000 in this guide.
- Make sure to have TCP port 9000 opened:
    - on network/firewall layer - Firewall
        - firewall opening for port 9000/tcp from relevant source agent(s) IP(s) or network(s)

We do not recommend too wide opening for 0.0.0.0 when you have Jenkins deployed in public cloud accessible from internet as it could be unnecessarily yet another potential target of attacks.

    - in OS level, for example firewalld:

```
firewall-cmd --add-port=9000/tcp --permanent
firewall-cmd --reload
```

> This generic 9000/tcp opening is fine if you are controlling access of each and every source trusted IP or IP pool(s) via portal firewall settings. Otherwise we recommend narrowing down the scope to your trusted source IP or IP pool(s).

- Listening on JNLP port tcp/9000
  - Go to https://jenkins.xxx.tds.customerx.com/configureSecurity (remember to use correct URL of your Jenkins controller)
  - Set "TCP port for inbound agents" to Fixed:9000
  - Open "advanced" and choose "Inbound TCP Agent Protocol/4 (TLS encryption)" (deselect others if not relevant)
- node added according to the following steps
  - Go to https://jenkins.xxx.tds.customerx.com/computer/new (remember to use correct URL of your Jenkins controller)
  - Set "Node name" to relevant name useful for you, we will use "node01" for simplicity of this example. Here are some recommended examples for inspiration - short name (node01), FQDN (node01.xxx.tds.customerx.com).
  - Choose "Permanent"
  - Set "Remote root directory" to "/data/jenkins-agent"
  - Set "Launch method" to "Launch agent by connecting it to the controller" (previously called "Launch agent via Java Web Start")
  - Click "Save"
- now go to newly created node and copy secret/token for connecting agent
  - Go to https://jenkins.xxx.tds.customerx.com/computer/XXX (remember to use correct URL of your Jenkins controller and replace XXX with the name of your node)
  - You will see something like in very first block called "Run from agent command line: (Unix)":

```
curl -sO https://jenkins.xxx.tds.customerx.com/jnlpJars/agent.jar
java -jar agent.jar -jnlpUrl https://jenkins.xxx.tds.customerx.com/computer/node01/jenkins-agent.
jnlp -secret 8b2911d98400bad5d45635b812b5f2e8e7c1d216bbbae9422a3ba57c691bf762 -workDir "/data
/jenkins-agent"
```

Please copy only the secret, which is for example in this case "8b2911d98400bad5d45635b812b5f2e8e7c1d216bbbae9422a3ba57c691bf762"

Jenkins agent node (slave) or so called on-premise executor

- This agent can be running on a server next to your Jenkins controller/master or in the internal network(s)
- agent service(s) with service auto-start to assure automatic re-connect to Jenkins controller at any time even after server reboot
  - Install dependencies
    - CentOS 7

```
yum install java-11-openjdk-devel git -y
# you can install also other dependencies that will be required for your jobs
```

    - CentOS 8/9

```
yum install java-17-openjdk-devel git -y
# you can install also other dependencies that will be required for your jobs
```

    - Ubuntu

```
apt-get update; apt-get install openjdk-11-jdk git -y
# you can install also other dependencies that will be required for your jobs
```

  - Installing agent
    - Prepare a folder for config

```
mkdir -p /data/configs
mkdir -p /opt/jenkins-agent
```

    - Create service file **/opt/jenkins-agent/jenkins-agent.service**

jenkinsope.service

```
[Unit]
Description=Jenkins Agent - On Premise Executor
Wants=network.target
After=network.target

[Service]
# EnvironmentFile cannnot be used on Debian/Ubuntu anymore - Reference: https://github.com
/varnishcache/pkg-varnish-cache/issues/24
# So we are using drop-in config /etc/systemd/system/jenkins-agent.service.d/local.conf
ExecStart=/usr/bin/java -Xms${JAVA_MEMORY} -Xmx${JAVA_MEMORY} -jar /opt/jenkins-agent/agent.
jar -jnlpUrl ${CONTROLLER_URL}/computer/${NODE_NAME}/jenkins-agent.jnlp -secret ${SECRET} -
workDir "${WORK_DIR}"
User=jenkins-agent
Restart=always
RestartSec=10
StartLimitInterval=0

[Install]
WantedBy=multi-user.target
```

- Create config file **/data/configs/jenkins-agent.conf**

```
JAVA_MEMORY=512m
CONTROLLER_URL=https://jenkins.xxx.tds.customerx.com
NODE_NAME=XXX
SECRET=8b2911d98400bad5d45635b812b5f2e8e7c1d216bbbae9422a3ba57c691bf762
WORK_DIR=/data/jenkins-agent
```

- Create script **/opt/jenkins-agent/jenkins-agent-install**

```
#!/bin/bash
set -e
AGENT_APP_HOME=/opt/jenkins-agent
SRV_USER_NAME=jenkins-agent
SRV_USER_ID="900"
SRV_GROUP_NAME=jenkins-agent
SRV_GROUP_ID="900"
WORK_DIR_DEFAULT="/data/jenkins-agent"
AGENT_SERVICE_NAME="jenkins-agent"
AGENT_SERVICE_FILE="/usr/lib/systemd/system/$AGENT_SERVICE_NAME.service"
AGENT_SERVICE_CFG_FOLDER="/etc/systemd/system/$AGENT_SERVICE_NAME.service.d"
MAINCONFIG=/data/configs/jenkins-agent.conf

# Making sure obsolete jenkinsope config is migrated when found
OPE_OLD_MAINCONFIG=/data/configs/jenkinsope.conf
[ -f $OPE_OLD_MAINCONFIG ] && mv -f $OPE_OLD_MAINCONFIG $MAINCONFIG

# Loading configuration
source $MAINCONFIG

# Detecting and migrating obsolete jenkinsope account and data
OPE_OLD_USER=jenkinsope
OPE_OLD_HOME=/home/$OPE_OLD_USER
if [ -d $OPE_OLD_HOME ];then
  sed -i "s#^WORK_DIR.*#WORK_DIR=$WORK_DIR_DEFAULT#g" $MAINCONFIG
  sed -i 's#^MASTER_URL#CONTROLLER_URL#g' $MAINCONFIG
  # Re-loading configuration with new values
  source $MAINCONFIG
  systemctl stop jenkinsope
  sleep 10
  rm -rf /etc/systemd/system/jenkinsope.service.d
  rm -f /usr/lib/systemd/system/jenkinsope.service
  mv $OPE_OLD_HOME $WORK_DIR
  userdel -rf $OPE_OLD_USER
```

```
fi

# Detecting and migrating obsolete jenkins-agent account and data from /home to /data
JA_OLD_HOME=/home/$SRV_USER_NAME
if [ -d $JA_OLD_HOME ] && [ $JA_OLD_HOME != $WORK_DIR ];then
  sed -i "s#^WORK_DIR.*#WORK_DIR=$WORK_DIR_DEFAULT#g" $MAINCONFIG
  # Re-loading configuration with new values
  source $MAINCONFIG
  systemctl stop $AGENT_SERVICE_NAME
  sleep 10
  rm -rf $AGENT_SERVICE_CFG_FOLDER
  rm -f $AGENT_SERVICE_FILE
  mv $JA_OLD_HOME $WORK_DIR
fi

echo "Preparing basic home folder structure"
mkdir -p $WORK_DIR/.ssh
chmod 700 $WORK_DIR/.ssh
touch $WORK_DIR/.ssh/config
chmod 600 $WORK_DIR/.ssh/*
ls -lah $WORK_DIR

echo "Preparing group"
getent group $SRV_GROUP_NAME || groupadd --gid $SRV_GROUP_ID $SRV_GROUP_NAME
getent group $SRV_GROUP_ID || groupmod -g $SRV_GROUP_ID $SRV_GROUP_NAME

echo "Preparing user"
getent passwd $SRV_USER_NAME || useradd --shell /bin/bash --uid $SRV_USER_ID --gid
$SRV_GROUP_ID --create-home --home-dir /data/jenkins-agent $SRV_USER_NAME
getent passwd $SRV_USER_NAME && usermod --shell /bin/bash --uid $SRV_USER_ID --gid
$SRV_GROUP_ID --home /data/jenkins-agent $SRV_USER_NAME
getent passwd $SRV_USER_NAME

echo "Changing ownership of home/work folder ($WORK_DIR) and its content (can take long
time with many files)..."
chown $SRV_USER_NAME:$SRV_USER_NAME -R $WORK_DIR
ls -lah $WORK_DIR

echo "Installing agent"
if [ -f $AGENT_SERVICE_FILE ];then
  echo "Previous service found, making sure it is stopped..."
  systemctl stop $AGENT_SERVICE_NAME
  sleep 5
fi
rm -f $AGENT_APP_HOME/agent.jar
curl -s ${CONTROLLER_URL}/jnlpJars/agent.jar -o $AGENT_APP_HOME/agent.jar
chmod 644 $AGENT_APP_HOME/agent.jar
install -D -m 644 $AGENT_APP_HOME/$AGENT_SERVICE_NAME.service $AGENT_SERVICE_FILE

# Check existence of JDK 11/17 and hard code it into service unit, otherwise attempt to use
generic java
JDK11="/usr/lib/jvm/java-11/bin/java"
JDK17="/usr/lib/jvm/java-17/bin/java"
if [ -f "$JDK17" ];then
  echo "Setting discovered OpenJDK 17 to be used by Jenkins agent."
  sed -i "s#/usr/bin/java#$JDK17#" $AGENT_SERVICE_FILE
else
  if [ -f "$JDK11" ];then
    echo "Setting discovered OpenJDK 11 to be used by Jenkins agent as alternative."
    sed -i "s#/usr/bin/java#$JDK11#" $AGENT_SERVICE_FILE
  fi
  # Otherwise using generic java
fi

mkdir -p $AGENT_SERVICE_CFG_FOLDER
echo "[Service]" > $AGENT_SERVICE_CFG_FOLDER/local.conf
sed 's#^#Environment=#g' $MAINCONFIG >> $AGENT_SERVICE_CFG_FOLDER/local.conf
systemctl daemon-reload
systemctl restart $AGENT_SERVICE_NAME
systemctl enable $AGENT_SERVICE_NAME
# Using mandatory sleep otherwise checking status too early ends with error
```

```
sleep 10
systemctl status $AGENT_SERVICE_NAME

echo "0 4 * * * root $AGENT_APP_HOME/jenkins-agent-install" > /etc/cron.d/jenkins-agent-update
```

- Run install script

```
chmod +x /opt/jenkins-agent/jenkins-agent-install
/opt/jenkins-agent/jenkins-agent-install
```

- Uninstalling agent (for cleanup purposes or if you messed up something)
  - Create script **/opt/jenkins-agent/jenkins-agent-uninstall**

```
systemctl disable jenkins-agent
systemctl stop jenkins-agent
rm -f /usr/lib/systemd/system/jenkins-agent.service
rm -rf /etc/systemd/system/jenkins-agent.service.d
systemctl daemon-reload
userdel -r jenkins-agent
rm -rf /data/jenkins-agent
```

  - Run install script

```
chmod +x /opt/jenkins-agent/jenkins-agent-uninstall
/opt/jenkins-agent/jenkins-agent-uninstall
```

Inspired by

- service itself - https://github.com/jenkinsci/systemd-slave-installer-module/blob/master/src/main/resources/org/jenkinsci/modules/systemd_slave_installer/jenkins-slave.service
- service parameters/options - https://gist.github.com/dragolabs/05dfe1c0899221ce51204dbfe7feecbb
- way of service installing - https://gist.github.com/michaelneale/9635744

Troubleshooting

Java Runtime class files recognition errors

Jenkins agent installing procedure tries to be smart enough to detect and use correct Java (OpenJDK 11 or 17), however in some cases it might fail.

Symptoms

- service jenkins-agent is constantly failing

```
[root@node01 ~]# systemctl status jenkins-agent
 jenkins-agent.service - Jenkins Agent - On Premise Executor
   Loaded: loaded (/usr/lib/systemd/system/jenkins-agent.service; enabled; vendor preset: disabled)
  Drop-In: /etc/systemd/system/jenkins-agent.service.d
           local.conf
   Active: activating (auto-restart) (Result: exit-code) since Sun 2023-10-08 13:40:53 UTC; 1s ago
  Process: 9960 ExecStart=/usr/bin/java -Xms${JAVA_MEMORY} -Xmx${JAVA_MEMORY} -jar /opt/jenkins-agent
/agent.jar -jnlpUrl ${CONTROLLER_URL}/computer/${NODE_NAME}/jenkins-agent.jnlp -secret ${SECRET} -
workDir ${WORK_DIR} (code=exited, status=1/FAILURE)
 Main PID: 9960 (code=exited, status=1/FAILURE)

Oct 08 13:40:53 loadgen systemd[1]: Unit jenkins-agent.service entered failed state.
Oct 08 13:40:53 loadgen systemd[1]: jenkins-agent.service failed.
```

- following errors can be spotted in journal

```
journalctl -fu jenkins-agent

Oct 08 13:51:57 loadgen systemd[1]: jenkins-agent.service holdoff time over, scheduling restart.
Oct 08 13:51:57 loadgen systemd[1]: Stopped Jenkins Agent - On Premise Executor.
Oct 08 13:51:57 loadgen systemd[1]: Started Jenkins Agent - On Premise Executor.
Oct 08 13:51:57 loadgen java[11258]: Error: A JNI error has occurred, please check your installation and
try again
Oct 08 13:51:57 loadgen java[11258]: Exception in thread "main" java.lang.UnsupportedClassVersionError:
hudson/remoting/Launcher has been compiled by a more recent version of the Java Runtime (class file
version 55.0), this version of the Java Runtime only recognizes class file versions up to 52.0
Oct 08 13:51:57 loadgen java[11258]: at java.lang.ClassLoader.defineClass1(Native Method)
...
Oct 08 13:51:57 loadgen java[11258]: at sun.launcher.LauncherHelper.checkAndLoadMain(LauncherHelper.java:
621)
Oct 08 13:51:57 loadgen systemd[1]: jenkins-agent.service: main process exited, code=exited, status=1
/FAILURE
Oct 08 13:51:57 loadgen systemd[1]: Unit jenkins-agent.service entered failed state.
Oct 08 13:51:57 loadgen systemd[1]: jenkins-agent.service failed.
```

Workaround

- Switch default java to JDK 11 or 17 interactively:

  ```
  update-alternatives --config java
  ```

  or in non-interactive way:

  ```
  alternatives --set java java-11-openjdk.x86_64

  # OR

  alternatives --set java java-17-openjdk.x86_64
  ```

- Inform other users of system that you made that change, as it might affect builds that were dependent on previously configured java version.