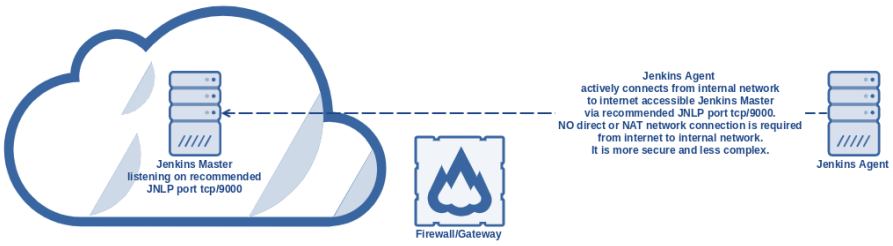
Executing Jenkins jobs when only one way network connection exists

Introduction

Usually, Jenkins master and Jenkins agent are located in the same network, therefore it is quite easy to adopt Jenkins agents using SSH. This time we will talk about the situation when Jenkins master and bunch of other DevOps tools are running in "network A" and Jenkins agents are running in "network B". Quite often connections from A to B are rejected, however, connections from B to A are accepted. There might be various reasons, in general, network B has elevated security expectations. The typical setup is Jenkins master located in public cloud and Jenkins agent(s) located in a private customer network. Private customer network might be needed in such because some sort of higher data privacy requirements must be matched or some specialised hardware is present in customer premises and it cannot be accessible directly from the public internet.

Solution

The recommended solution is master/agent setup, when the master runs in a public environment and agent is running on a server located in a private network. The agent is also called "on-premise executor" (OPE). Jobs are then performing jobs/tasks on such agent in internal/private networks. Jenkins Agent has an active connection from the internal network to internet accessible Jenkins Master via recommended JNLP port tcp/9000 and keeps listening to builds/jobs. Port can be changed, but we will keep talking about 9000. NO direct or NAT network connection is required from the internet to internal network. It is a secure and simple solution.



Requirements:

- Network
  - firewall opening for port tcp/9000 from relevant source agent(s) IP(s) in internal network towards internet in general (destination 0.0.0.0)
    - network layer (using TDS portal network functionality)
    - operating system layer (firewalld if needed)
- Jenkins master
  - running in public
  - listening on JNLP port tcp/9000
    - Go to https://jenkins.xxx.tds.customerx.com/configureSecurity (remember to use correct URL of your Jenkins master)
    - 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 master)
    - Set "Node name" to relevant name useful for you
    - Choose "Permanent"
    - Set "Remote root directory" to "/home/jenkinsope"
    - Set "Launch method" to "Launch agent by connecting it to the master" previously called "Launch agent via Java Web Start"
  - copy secret/token for connecting agent
    - Go to https://jenkins.xxx.tds.customerx.com/computer/XXX (remember to use correct URL of your Jenkins master and replace XXX with the name of your node)
    - You will see something like:

```
Run from agent command line:

java -jar agent.jar -jnlpUrl https://jenkins.xxx.tds.customerx.com/computer/test/slave-agent.jnlp -secret 8b2911d98400bad5d45635b812b5f2e8e7c1d216bbbae9422a3ba57c691bf762 -workDir "/home/jenkinsope"
Run from agent command line, with the secret stored in a file:

echo 8b2911d98400bad5d45635b812b5f2e8e7c1d216bbbae9422a3ba57c691bf762 > secret-file
java -jar agent.jar -jnlpUrl https://jenkins.xxx.tds.customerx.com/computer/test/slave-agent.jnlp -secret @secret-file -workDir "/home/jenkinsope"
```

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

- Jenkins agent node (slave) - or so-called "on-premise executor"
  - running on a server in the internal network(s)
  - agent service(s) with service auto-start to assure automatic re-connect to Jenkins master at any time even after server reboot
    - Install dependencies
      - CentOS

```
yum install java-1.8.0-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-8-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
```

      - Create service file **/tmp/jenkinsope.service**

jenkinsope.service

```
[Unit]
Description=Jenkins Slave 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/jenkinsope.service.d/local.conf
ExecStart=/usr/bin/java -Xms${JAVA_MEMORY} -Xmx${JAVA_MEMORY} -jar /usr/bin/agent.jar -jnlpUrl ${MASTER_URL}/computer/${SLAVE_NAME}/slave-agent.jnlp -secret ${SECRET} -workDir "${WORK_DIR}"
User=jenkinsope
Restart=always
RestartSec=10
StartLimitInterval=0

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

      - Create config file **/data/configs/jenkinsope.conf**

```
JAVA_MEMORY=512m
MASTER_URL=https://jenkins.xxx.tds.customerx.com
SLAVE_NAME=XXX
SECRET=8b2911d98400bad5d45635b812b5f2e8e7c1d216bbbae9422a3ba57c691bf762
WORK_DIR=/home/jenkinsope
```

- Create script **/tmp/jenkinsope-install.sh**

```
useradd -m -s /bin/bash jenkinsope
mkdir -p /home/jenkinsope/.ssh
chmod 700 /home/jenkinsope/.ssh
touch /home/jenkinsope/.ssh/config
chmod 600 /home/jenkinsope/.ssh/*
chown jenkinsope:jenkinsope -R /home/jenkinsope/.ssh
source /data/configs/jenkinsope.conf
wget ${MASTER_URL}/jnlpJars/agent.jar -O /usr/bin/agent.jar
chmod 644 /usr/bin/agent.jar
install -D -m 644 /tmp/jenkinsope.service /usr/lib/systemd/system/jenkinsope.service
mkdir -p /etc/systemd/system/jenkinsope.service.d
echo "[Service]" > /etc/systemd/system/jenkinsope.service.d/local.conf
sed 's#^#Environment=#g' /data/configs/jenkinsope.conf >> /etc/systemd/system
/jenkinsope.service.d/local.conf
systemctl daemon-reload
systemctl restart jenkinsope
systemctl enable jenkinsope
systemctl status jenkinsope
```

- Run install script

```
chmod +x /tmp/jenkinsope-install.sh
/tmp/jenkinsope-install.sh
```

- Uninstalling agent (for cleanup purposes or if you messed up something)
  - Create script **/tmp/jenkinsope-uninstall.sh**

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

  - Run install script

```
chmod +x /tmp/jenkinsope-uninstall.sh
/tmp/jenkinsope-uninstall.sh
```

Inspired by:

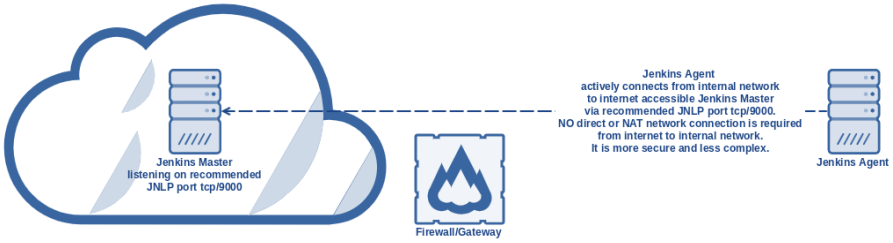
- service itself - https://github.com/jenkinsci/systemd-slave-installer-module/blob/master/src/main/resources/org/jenkinsci/modules/systemd_slave_installer/jenkins-slave.service
- service parameters/options - https://gist.github.com/dragolabs/05dfe1c0899221ce51204dbfe7feecbb
- way of service installing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

Introduction

Usually, Jenkins master and Jenkins agent are located in the same network, therefore it is quite easy to adopt Jenkins agents using SSH. This time we will talk about the situation when Jenkins master and bunch of other DevOps tools are running in "network A" and Jenkins agents are running in "network B". Quite often connections from A to B are rejected, however, connections from B to A are accepted. There might be various reasons, in general, network B has elevated security expectations. A typical setup is Jenkins master located in public cloud and Jenkins agent(s) located in a private customer network. Private customer network might be needed in such because some sort of higher data privacy requirements must be matched or some specialised hardware is present in customer premises and it cannot be accessible directly from the public internet.

Solution

The recommended solution is master/agent setup, when the master runs in a public environment and agent is running on a server located in a private network. The agent is also called "on-premise executor" (OPE). Jobs are then performing jobs/tasks on such agent in internal/private networks. Jenkins Agent has an active connection from the internal network to internet accessible Jenkins Master via recommended JNLP port tcp/9000 and keeps listening to builds/jobs. Port can be changed, but we will keep talking about 9000. NO direct or NAT network connection is required from the internet to internal network. It is a secure and simple solution.



Requirements:

- Network
  - firewall opening for port tcp/9000 from relevant source agent(s) IP(s) in internal network towards internet in general (destination 0.0.0.0)
    - network layer (using TDS portal network functionality)
    - operating system layer (firewalld if needed)
- Jenkins master
  - running in public
  - listening on JNLP port tcp/9000
    - Go to https://jenkins.xxx.tds.customerx.com/configureSecurity (remember to use correct URL of your Jenkins master)
    - 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 master)
    - Set "Node name" to relevant name useful for you
    - Choose "Permanent"
    - Set "Remote root directory" to "/home/jenkinsope"
    - Set "Launch method" to "Launch agent by connecting it to the master" previously called "Launch agent via Java Web Start"
  - copy secret/token for connecting agent
    - Go to https://jenkins.xxx.tds.customerx.com/computer/XXX (remember to use correct URL of your Jenkins master and replace XXX with the name of your node)
    - You will see something like:

```
Run from agent command line:

java -jar agent.jar -jnlpUrl https://jenkins.xxx.tds.customerx.com/computer/test/slave-
agent.jnlp -secret 8b2911d98400bad5d45635b812b5f2e8e7c1d216bbbae9422a3ba57c691bf762 -
workDir "/home/jenkinsope"
Run from agent command line, with the secret stored in a file:

echo 8b2911d98400bad5d45635b812b5f2e8e7c1d216bbbae9422a3ba57c691bf762 > secret-file
java -jar agent.jar -jnlpUrl https://jenkins.xxx.tds.customerx.com/computer/test/slave-
agent.jnlp -secret @secret-file -workDir "/home/jenkinsope"
```

    Please copy only the secret, which is for example in this case "8b2911d98400bad5d45635b812b5f2e8e7c1d216bbbae9422a3ba57c691bf762"
- Jenkins agent node (slave) - or so-called "on-premise executor"
  - running on a server in internal network(s)
  - agent service(s) with service auto-start to assure automatic re-connect to Jenkins master at any time even after server reboot
    - Install dependencies
      - CentOS

```
yum install java-1.8.0-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-8-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
```

  - Create service file **/tmp/jenkinsope.service**

jenkinsope.service

```
[Unit]
Description=Jenkins Slave 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/jenkinsope.service.d/local.conf
ExecStart=/usr/bin/java -Xms${JAVA_MEMORY} -Xmx${JAVA_MEMORY} -jar /usr/bin/agent.
jar -jnlpUrl ${MASTER_URL}/computer/${SLAVE_NAME}/slave-agent.jnlp -secret ${SECRET}
-workDir "${WORK_DIR}"
User=jenkinsope
Restart=always
RestartSec=10
StartLimitInterval=0

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

  - Create config file **/data/configs/jenkinsope.conf**

```
JAVA_MEMORY=512m
MASTER_URL=https://jenkins.xxx.tds.customerx.com
SLAVE_NAME=XXX
SECRET=8b2911d98400bad5d45635b812b5f2e8e7c1d216bbbae9422a3ba57c691bf762
WORK_DIR=/home/jenkinsope
```

  - Create script **/tmp/jenkinsope-install.sh**

```
useradd -m -s /bin/bash jenkinsope
mkdir -p /home/jenkinsope/.ssh
chmod 700 /home/jenkinsope/.ssh
touch /home/jenkinsope/.ssh/config
chmod 600 /home/jenkinsope/.ssh/*
chown jenkinsope:jenkinsope -R /home/jenkinsope/.ssh
source /data/configs/jenkinsope.conf
wget ${MASTER_URL}/jnlpJars/agent.jar -O /usr/bin/agent.jar
chmod 644 /usr/bin/agent.jar
install -D -m 644 /tmp/jenkinsope.service /usr/lib/systemd/system/jenkinsope.service
mkdir -p /etc/systemd/system/jenkinsope.service.d
echo "[Service]" > /etc/systemd/system/jenkinsope.service.d/local.conf
sed 's#^#Environment=#g' /data/configs/jenkinsope.conf >> /etc/systemd/system
/jenkinsope.service.d/local.conf
systemctl daemon-reload
systemctl restart jenkinsope
systemctl enable jenkinsope
systemctl status jenkinsope
```

  - Run install script

```
chmod +x /tmp/jenkinsope-install.sh
/tmp/jenkinsope-install.sh
```

- Uninstalling agent (for cleanup purposes or if you messed up something)
  - Create script **/tmp/jenkinsope-uninstall.sh**

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

  - Run install script

```
chmod +x /tmp/jenkinsope-uninstall.sh
/tmp/jenkinsope-uninstall.sh
```

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