Starting and Accessing Jenkins

Starting Jenkins

See also

- Features controlled by system properties
- Administering Jenkins

The easiest way to execute Jenkins is through the built in Jetty servlet container. You can execute Jenkins like this:

```
$ java -jar jenkins.war
```

Of course, you probably want to send the output of Jenkins to a log file, and if you're on Unix, you probably want to use **nohup**:

```
$ nohup java -jar jenkins.war > $LOGFILE 2>&1
```

Accessing Jenkins

To see Jenkins, simply bring up a web browser and go to URL **`http://mySer ver:8080`** where mySer ver is the name of the system running Jenkins.

Top of page

Command Line Parameters

Jenkins normally starts up using port 8080. However, if you have other web services starting up you might find that this port is already taken. You can specify a different port by using `--httpPort=$HTTP_PORT` where `$HTTP_PORT` is the port you want Jenkins to run on. Other command line parameters include:

Command Line Parameter	Description
--help	Displays all available options and their usage
--httpPort=$HTTP_PORT	Runs Jenkins listener on port $HTTP_PORT using standard *http* protocol. The default is port 8080. To disable (because you're using *https*), use port `-1`. This option does not impact the root URL being generated within Jenkins logic (UI, JNLP files, etc.); it is defined by the Jenkins URL specified in the global configuration.
--httpListenAddress=$HTTP_HOST	Binds Jenkins to the IP address represented by $HTTP_HOST. The default is 0.0.0.0 — i.e. listening on all available interfaces. For example, to only listen for requests from localhost, you could use: --httpListenAddress=127.0.0.1
--httpsPort=$HTTPS_PORT	Uses HTTPS protocol on port $HTTPS_PORT. This option does not impact the root URL being generated within Jenkins logic (UI, JNLP files, etc.); it is defined by the Jenkins URL specified in the global configuration.
--httpsListenAddress=$HTTPS_HOST	Binds Jenkins to listen for HTTPS requests on the IP address represented by $HTTPS_HOST.
--http2Port=$HTTP_PORT	Uses HTTP/2 protocol on port $HTTP_PORT. This option does not impact the root URL being generated within Jenkins logic (UI, JNLP files, etc.); it is defined by the Jenkins URL specified in the global configuration.
--http2ListenAddress=$HTTPS_HOST	Binds Jenkins to listen for HTTP/2 requests on the IP address represented by $HTTPS_HOST.
--prefix=$PREFIX	Runs Jenkins to include the $PREFIX at the end of the URL. For example, to make Jenkins accessible at **`http://mySer ver:8080/jenkins`**, set --prefix=/jenkins
--ajp13Port=$AJP_PORT	Runs Jenkins listener on port $AJP_PORT using standard *AJP13* protocol. The default is port 8009. To disable (because you're using *https*), use port `-1`.
--ajp13ListenAddress=$AJP_HOST	Binds Jenkins to the IP address represented by $AJP_HOST. The default is 0.0.0.0 — i.e. listening on all available interfaces.
--argumentsRealm.passwd.$ADMIN_USER	Sets the password for user $ADMIN_USER. If Jenkins security is turned on, you must log in as the $ADMIN_USER in order to configure Jenkins or a Jenkins project. NOTE: You must also specify that this user has an *admin* role. (See next argument below).
--argumentsRealm.roles.$ADMIN_USER=admin	Sets that $ADMIN_USER is an administrative user and can configure Jenkins if Jenkins' security is turned on. See Sec uring Jenkins for more information.

--sessionTimeout=$SESSION_TIMEOUT	Sets the http session timeout value to $SESSION_TIMEOUT minutes. Default to what webapp specifies, and then to 60 minutes
--useJmx	Enable Jetty JMX (See documentation)
-Xdebug -Xrunjdwp:transport=dt_socket,address=$DEBUG_PORT,server=y,suspend=n	Sets debugging on and you can access debug on $DEBUG_PORT.
--logfile=$LOG_PATH/winstone_`date +"%Y-%m-%d_%H-%M"`.log	Logging to desired file
-XX:PermSize=512M -XX:MaxPermSize=2048M -Xmn128M -Xms1024M -Xmx2048M	referring to these options for Oracle Java

Jenkins passes all (or just leading parameters until the first Jenkins-specific parameter?) command line parameters to the Winstone servlet container, so you can get more information by looking at the Winstone Command Line Parameter Reference and jenkinsci/winstone.

Be Careful with Command Line Parameters

Jenkins ignores command line parameters it doesn't understand instead of producing an error. Be careful when using command line parameters and make sure you have the correct spelling. For example, the parameter needed for defining the Jenkins administrative user is **`--argumentsRealm`** and not **`--argumentRealm`**.

A very simple init script

The following script is for Ubuntu based systems. The RedHat Jenkins distribution contains a startup script.

```
#!/bin/sh

DESC="Jenkins CI Server"
NAME=jenkins
PIDFILE=/var/run/$NAME.pid
RUN_AS=jenkins
COMMAND="/usr/bin/java -- -jar /home/jenkins/jenkins.war"

d_start() {
        start-stop-daemon --start --quiet --background --make-pidfile --pidfile $PIDFILE --chuid $RUN_AS --exec
$COMMAND
}

d_stop() {
        start-stop-daemon --stop --quiet --pidfile $PIDFILE
        if [ -e $PIDFILE ]
                then rm $PIDFILE
        fi
}

case $1 in
        start)
        echo -n "Starting $DESC: $NAME"
        d_start
        echo "."
        ;;
        stop)
        echo -n "Stopping $DESC: $NAME"
        d_stop
        echo "."
        ;;
        restart)
        echo -n "Restarting $DESC: $NAME"
        d_stop
        sleep 1
        d_start
        echo "."
        ;;
        *)
        echo "usage: $NAME {start|stop|restart}"
        exit 1
        ;;
esac

exit 0
```

Using HTTPS with an existing certificate

If you're setting up Jenkins using the built-in Winstone server and want to use an existing certificate for HTTPS:

```
--httpPort=-1 --httpsPort=443 --httpsKeyStore=path/to/keystore --httpsKeyStorePassword=keystorePassword
```

The keystore should be in JKS format (as created by the JDK 'keytool') and the keystore and target key must have the same password. (Placing the keystore arguments after Jenkins-specific parameters does not seem to work; either they are not forwarded to Winstone or Winstone ignores them coming after unknown parameters. So, make sure they are adjacent to the working `--httpsPort` argument.)

If your keystore contains multiple certificates (e.g. you are using CA signed certificate) Jenkins might end-up using a incorrect one. In this case you can con vert the keystore to PEM and use following command line options:

```
--httpPort=-1 --httpsPort=443 --httpsCertificate=path/to/cert --httpsPrivateKey=path/to/privatekey
```

Passing the Command Line Parameters to an instance on a Mac OSX (Currently is Mavericks 10.9.4) that uses launchctl (rather than using Jenkins.jar to start up)

In this example, we set the Jenkins server to listen for HTTPS on port 8443. Note that we do not disable the httpPort by passing in -1. So in this example, your server would answer on both http and https. We also assume that the user has already created the keystore (see the "Using SSL" section from http://wiki.wocommunity.org/display/documentation/Installing+and+Configuring+Jenkins)

```
sudo launchctl unload /Library/LaunchDaemons/org.jenkins-ci.plist

sudo defaults write /Library/Preferences/org.jenkins-ci httpsPort 8443
sudo defaults write /Library/Preferences/org.jenkins-ci httpsKeyStore /path/to/your/keystore/file
sudo defaults write /Library/Preferences/org.jenkins-ci httpsKeyStorePassword <keystore password>

sudo launchctl load /Library/LaunchDaemons/org.jenkins-ci.plist
```

Using HTTP/2

With Java 8 (should be included per default in Java 9), you need to include alpn boot jar in the bootclasspath. The alpn boot jar depends on your jvm version. Have a look here https://www.eclipse.org/jetty/documentation/current/alpn-chapter.html#alpn-versions to figure which version to use.

You can download it from (with alpn boot version 8.1.12.v20180117): https://repo.maven.apache.org/maven2/org/mortbay/jetty/alpn/alpn-boot/8.1.12.v20180117/alpn-boot-8.1.12.v20180117.jar

Then you have to include it on jvm start:

```
java -Xbootclasspath/p:alpn-boot-8.1.12.v20180117.jar -jar target/jenkins.war --http2Port=9090
```

Configuring https certificates with Windows

Creating a certificate for use within Jenkins. This used a stock Jenkins 1.612 installation on Windows Server 2008 R2 Standard 64-bit. This creates a certificate signed by a Certificate Authority such as Digicert, if making your own certificate skip steps 3, 4, and 5.

This process utilizes Java's keytool, however you do not have to perform a separate Java installation if you don't need it. Jenkins packages a JRE with it when you do the installation, located in `C:\Program Files (x86)\Jenkins\jre\bin`

Step 1: Create a new keystore on your server. This will place a 'keystore' file in your current directory.

```
C:\Program Files (x86)\Jenkins\jre\bin>keytool -genkeypair -keysize 2048 -keyalg RSA -alias jenkins -keystore
keystore
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: server-name.your.company.com
What is the name of your organizational unit?
[Unknown]: Your City
What is the name of your organization?
[Unknown]: Your company name
What is the name of your City or Locality?
[Unknown]: Your city
What is the name of your State or Province?
[Unknown]: Your State
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=server-name.your.company.com, OU=Your City, O=Your company name, L=Your City, ST=Your State, C=US correct?
[no]: yes

Enter key password for <jenkins>
(RETURN if same as keystore password):
```

Step 2: Verify the keystore was created (your fingerprint will vary)

```
C:\Program Files (x86)\Jenkins\jre\bin>keytool -list -keystore keystore
Enter keystore password:

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

jenkins, May 6, 2015, PrivateKeyEntry,
Certificate fingerprint (SHA1): AA:AA:AA:AA:AA:AA:AA:AA:AA:AA:AA:AA:AA:AA:AA:AA:AA:AA:AA:AA
```

Step 3: Create the certificate request. This will create a 'certreq.csr' file in your current directory.

```
C:\Program Files (x86)\Jenkins\jre\bin>keytool -certreq -alias jenkins -keyalg RSA -file certreq.csr -ext
SAN=dns:server-name,dns:server-name.your.company.com -keystore keystore
Enter keystore password:
```

Step 4: Use the contents of the `certreq.csr` file to generate a certificate from your certificate provider. Request a SHA-1 certificate (SHA-2 is untested but will likely work). If using DigiCert, download the resulting certificate as Other format "a .p7b bundle of all the certs in a .p7b file".

Step 5: Add the resulting .p7b into the keystore you created above.

```
C:\Program Files (x86)\Jenkins\jre\bin>keytool -import -alias jenkins -trustcacerts -file
response_from_digicert.p7b -keystore keystore
Enter keystore password:
Certificate reply was installed in keystore
```

Step 6: Copy the 'keystore' file to your Jenkins secrets directory. On a stock installation, this will be at

```
C:\Program Files (x86)\Jenkins\secrets
```

Step 7: Modify the <arguments> section of your `C:\Program Files (x86)\Jenkins\jenkins.xml` file to reflect the new certificate. Note: This example disables http via `--httpPort=-1` and places the server on 8443 via `--httpsPort=8443`.

```
<arguments>-Xrs -Xmx256m -Dhudson.lifecycle=hudson.lifecycle.WindowsServiceLifecycle -jar "%BASE%\jenkins.war"
--httpPort=-1 --httpsPort=8443 --httpsKeyStore="%BASE%\secrets\keystore" --httpsKeyStorePassword=your.password.
here</arguments>
```

Step 8: Restart the jenkins service to initialize the new configuration.

```
net stop jenkins
net start jenkins
```

Step 9: After 30-60 seconds, Jenkins will have completed the startup process and you should be able to access the website at https://server-name.your.company.com:8443; Verify the certificate looks good via your browser's tools. If the service terminates immediately, there's an error somewhere in your configuration. Useful error information can be found in:

```
C:\Program Files (x86)\Jenkins\jenkins.err.log
C:\Program Files (x86)\Jenkins\jenkins.out.log
```

Top of page