Jenkins CLI

> More up-to-date information is available in the Jenkins documentation at https://jenkins.io/doc/book/managing/cli/

Jenkins has a built-in command line interface that allows you to access Jenkins from a script or from your shell. This is convenient for automation of routine tasks, bulk updates, trouble diagnosis, and so on.

This interface is accessed via the Jenkins CLI client, which is a Java JAR file distributed with Jenkins.

Obtaining the CLI client

You can download the JAR file for the client from the URL "`/jnlpJars/jenkins-cli.jar`" on your Jenkins server, e.g. `https://jenkins.example.com/jnlpJars/jenkins-cli.jar`
In theory, the CLI JAR is dependent on the version of Jenkins, but in practice, we expect to be able to retain compatibility between different versions of Jenkins. In case of problems, just re-download the latest JAR from your Jenkins server.

Running a CLI command

The general syntax is as follows (the design is similar to tools like svn/git):

```
java -jar jenkins-cli.jar [-s JENKINS_URL] command [options...]
[arguments...]
```

`JENKINS_URL` can be specified via the environment variable `$JENKINS_URL`. This environment variable is automatically set when Jenkins fork a process during builds, which allows you to use Jenkins CLI from inside the build without explicitly configuring the URL.

For details on authentication methods ("`-auth`" parameter or `JENKINS_USER_ID` and `JENKINS_API_TOKEN` environment variables), see Jenkins CLI > Client connection modes > HTTP connection mode.

Getting help

The list of the available commands depends on the server you are talking to. Visit `https://jenkins.example.com/cli` or use '`help`' command to list them all:

```
java -jar jenkins-cli.jar -s https://jenkins.example.com help [command]
```

More detailed help for individiual commands can be found by adding the command name after `help` (e.g. `help build`).
The same information is available via the web UI, by clicking on a command name on the Jenkins CLI page.

Extending the CLI

Plugins installed on Jenkins server can add custom CLI commands. See Writing CLI commands for more details.

Working with Credentials

> Jenkins accounts must have the Overall/Read account permission to access the CLI.

1.576 and later

Whenever the CLI tries to to connect to the Jenkins server, it offers the before mentioned SSH keys. When the user has those keys but doesn't want to use them to authenticate, it's possible to use the `-noKeyAuth` argument to skip being prompted for the key's password. This way the CLI will never try to use available SSH keys.

1.419 and later

If your Jenkins requires authentication, you should set up public key authentication. Login from the web UI and go to `http://yourserver.com/me/configure`, then set your public keys in the designated text area. When connecting to the server, the CLI will look for `~/.ssh/identity`, `~/.ssh/id_dsa`, `~/.ssh/id_rsa` and use those to authenticate itself against the server. Alternatively, the `-i` option can be used to explicitly specify the location of the private key.

See the middle of this guide for how to generate SSH key pair, if you don't have one yet.

If you have used PuttyGen to generate your keys, you will have to convert them to openssh format. Otherwise Jenkins might silently ignore your keys and you will be *Authenticated as: anonymous*.

To use the `-i` option the syntax is as follows:

```
java -jar jenkins-cli.jar [-s JENKINS_URL] [-i PRIVATE_KEY] command
[options...] [arguments...]
```

> For compatibility reasons, unless you use the `-i` option, failure to authenticate by itself does not constitute a fatal error. It will instead try to execute the command anyway, as the anonymous user.

Before 1.419

If your Jenkins requires authentication, use `--username` and `--password` or `--password-file` options to specify the credentials. To avoid doing this for every command, you can also use the `login` CLI command once (with the same credentials parameters), and after that you may use other commands without specifying credentials.
Note that not every authentication type supports these parameters for credentials. Prior to version 1.373, only authentication in Jenkins' own database was supported. As of 1.373, LDAP is also supported. If the CLI reports these are invalid parameters, file an issue for your authentication type and ask them to extend `AbstractPasswordBasedSecurityRealm` instead of directly from `SecurityRealm` to get support for these parameters.

Change History: Note that a security hole in CLI commands was fixed in version 1.371, and that CLI `login` did not work properly for many commands until 1.375.

Connection mechanism

Jenkins CLI clients and Jenkins server establishes the communication in the following fashion.

1. Jenkins listens on a TCP/IP port configured under "TCP port for JNLP agents" in the system configuration page. This single port is used for both agents and CLI.
2. Jenkins advertises this port number as a special HTTP header (if disabled, this header will not be present).
3. CLI client will make an HTTP request to the top page of Jenkins, looking for this header.
4. If the header is found and the TCP/IP port is identified, the client will attempt to connect to this URL.
5. If that fails (for example, if there's a reverse proxy and Jenkins runs on a different host, or if a firewall blocks access to this TCP/IP port), or if the header is not found, it will fall back to the communication mechanism that uses two simultaenous HTTP connections.

> **Use 1.427 for the fallback behavior**
>
> Up until 1.426, if the server advertises a separate TCP/IP port, then a client failure to connect to this port was fatal. Since 1.427, the client is improved to fall back to HTTP-based mechanism. See JENKINS-10611

Configuring TCP/IP port for CLI and agents.

You have to configure global security in order to select the port (rather than system configuration). Using a fixed port allows you to configure your firewall more easily.

Commons problems

Operation timed out

```
$ java -jar jenkins-cli.jar -s YOUR_SERVER_URL help
Exception in thread "main" java.net.ConnectException: Operation timed
out
        at java.net.PlainSocketImpl.socketConnect(Native Method)
        at java.net.PlainSocketImpl.doConnect(PlainSocketImpl.java:351)
        at java.net.PlainSocketImpl.connectToAddress(PlainSocketImpl.
java:213)
        at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:200)
        at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:432)
        at java.net.Socket.connect(Socket.java:529)
        at java.net.Socket.connect(Socket.java:478)
        at java.net.Socket.<init>(Socket.java:375)
        at java.net.Socket.<init>(Socket.java:189)
        at hudson.cli.CLI.<init>(CLI.java:97)
        at hudson.cli.CLI.<init>(CLI.java:82)
        at hudson.cli.CLI._main(CLI.java:250)
        at hudson.cli.CLI.main(CLI.java:199)
```

Check that the JNLP port is opened if you are using a firewall on your server. You can configure its value in Jenkins configuration. By default it is set to use a random port.

java.io.EOFException

```
$ java -jar jenkins-cli.jar -s YOUR_SERVER_URL login
Exception in thread "main" java.io.EOFException
        at java.io.DataInputStream.readBoolean(DataInputStream.java:227)
        at hudson.cli.Connection.readBoolean(Connection.java:90)
        at hudson.cli.CLI.authenticate(CLI.java:360)
        at hudson.cli.CLI._main(CLI.java:255)
        at hudson.cli.CLI.main(CLI.java:199)
```

If on the server side you have such logs (perhaps with another security manager)

```
INFO: Accepted connection #54 from /88.171.115.235:60876
Exception in thread "Thread-3518" java.lang.
UnsupportedOperationException: Not giving you the password
        at com.atlassian.crowd.integration.acegi.user.CrowdUserDetails.
getPassword(CrowdUserDetails.java:52)
        at hudson.model.User.impersonate(User.java:250)
        at org.jenkinsci.main.modules.cli.auth.ssh.SshCliAuthenticator.
authenticate(SshCliAuthenticator.java:44)
        at hudson.cli.CliManagerImpl$1.run(CliManagerImpl.java:99)
```

This issues was fixed in Jenkins 1.424

WARNING: No header 'X-SSH-Endpoint' returned by Jenkins

You may get this error, which usually prevents you from using SSH CLI due to

~~JENKINS-45651~~ - X-SSH-Endpoint is not provided on top page when 401/403 are returned CLOSED

but if you are using nginx as a proxy you could trick it to add this information instead of Jenkins.

```
add_header 'X-SSH-Endpoint' 'jenkins.example.com:50022' always;
```

If you hit this, please be sure you add a comment on that bug in order to assure that it is reopened and fixed correctly.

java.io.IOException / javax.net.ssl.SSLHandshakeException (SSL certificate issue)

You may face this issue if the certificate is not trusted, e.g. self-signed certificate.

```
bash-4.1$ java -jar jenkins-cli.jar -s https://jenkins.example.com/ help
Exception in thread "main" java.io.IOException: Failed to connect to
https://jenkins.example.com/
        at hudson.cli.CLI.getCliTcpPort(CLI.java:274)
        at hudson.cli.CLI.<init>(CLI.java:134)
        at hudson.cli.CLIConnectionFactory.connect(CLIConnectionFactory.
java:72)
        at hudson.cli.CLI._main(CLI.java:469)
        at hudson.cli.CLI.main(CLI.java:384)
Caused by: javax.net.ssl.SSLHandshakeException: sun.security.validator.
ValidatorException: \
PKIX path building failed: sun.security.provider.certpath.
SunCertPathBuilderException: \
unable to find valid certification path to requested target
        at sun.security.ssl.Alerts.getSSLException(Alerts.java:192)
        at sun.security.ssl.SSLSocketImpl.fatal(SSLSocketImpl.java:1682)
        at sun.security.ssl.Handshaker.fatalSE(Handshaker.java:257)
        at sun.security.ssl.Handshaker.fatalSE(Handshaker.java:251)
        at sun.security.ssl.ClientHandshaker.serverCertificate
(ClientHandshaker.java:1168)
        at sun.security.ssl.ClientHandshaker.processMessage
(ClientHandshaker.java:153)
        at sun.security.ssl.Handshaker.processLoop(Handshaker.java:609)
        at sun.security.ssl.Handshaker.process_record(Handshaker.java:
545)
        at sun.security.ssl.SSLSocketImpl.readRecord(SSLSocketImpl.java:
930)
        at sun.security.ssl.SSLSocketImpl.performInitialHandshake
(SSLSocketImpl.java:1175)
        at sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.
java:1202)
        at sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.
java:1186)
        at sun.net.www.protocol.https.HttpsClient.afterConnect
(HttpsClient.java:440)
        at sun.net.www.protocol.https.
AbstractDelegateHttpsURLConnection.connect
(AbstractDelegateHttpsURLConnection.java:185)
        at sun.net.www.protocol.https.HttpsURLConnectionImpl.connect
(HttpsURLConnectionImpl.java:153)
        at hudson.cli.CLI.getCliTcpPort(CLI.java:272)
        ... 4 more
Caused by: sun.security.validator.ValidatorException: \
PKIX path building failed: sun.security.provider.certpath.
SunCertPathBuilderException: \
unable to find valid certification path to requested target
        at sun.security.validator.PKIXValidator.doBuild(PKIXValidator.
```

```
java:324)
        at sun.security.validator.PKIXValidator.engineValidate
(PKIXValidator.java:224)
        at sun.security.validator.Validator.validate(Validator.java:235)
        at sun.security.ssl.X509TrustManagerImpl.validate
(X509TrustManagerImpl.java:147)
        at sun.security.ssl.X509TrustManagerImpl.checkServerTrusted
(X509TrustManagerImpl.java:230)
        at sun.security.ssl.X509TrustManagerImpl.checkServerTrusted
(X509TrustManagerImpl.java:270)
        at sun.security.ssl.ClientHandshaker.serverCertificate
(ClientHandshaker.java:1147)
        ... 15 more
Caused by: sun.security.provider.certpath.SunCertPathBuilderException: \
unable to find valid certification path to requested target
        at sun.security.provider.certpath.SunCertPathBuilder.engineBuild
(SunCertPathBuilder.java:197)
        at java.security.cert.CertPathBuilder.build(CertPathBuilder.
java:255)
        at sun.security.validator.PKIXValidator.doBuild(PKIXValidator.
java:319)
        ... 21 more
```

see https://issues.jenkins-ci.org/browse/JENKINS-12629 for a way to trust a self-signed certificate (rather than using `-noCertificateCheck` option)

```
JENKINS_HOST=jenkins.example.com
JENKINS_PORT=443
JENKINS_URL=https://${JENKINS_HOST}:${JENKINS_PORT}
KEYSTOREFILE=myKeystore
KEYSTOREPASS=changeme

# get the SSL certificate
openssl s_client -connect ${JENKINS_HOST}:${JENKINS_PORT} </dev/null |
sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > ${JENKINS_HOST}.
cer
# create a keystore and import certificate
keytool -import -noprompt -trustcacerts -alias ${JENKINS_HOST} -file
${JENKINS_HOST}.cer -keystore ${KEYSTOREFILE} -storepass ${KEYSTOREPASS}
# verify that the certificate is listed
keytool -list -v -keystore ${KEYSTOREFILE} -storepass ${KEYSTOREPASS}
# get jenkins-cli
wget --no-check-certificate ${JENKINS_URL}/jnlpJars/jenkins-cli.jar
# test access
alias jcli="java -Djavax.net.ssl.trustStore=${KEYSTOREFILE} -Djavax.net.
ssl.trustStorePassword=${KEYSTOREPASS} -jar jenkins-cli.jar -s
${JENKINS_URL,,}"
jcli help
# ... or set this in your ~/.bashrc
export JAVA_TOOL_OPTIONS="-Djavax.net.ssl.trustStore=${KEYSTOREFILE} -
Djavax.net.ssl.trustStorePassword=${KEYSTOREPASS}"
java -jar jenkins-cli.jar -s ${JENKINS_URL} help
```

java.io.IOException: No X-Jenkins-CLI2-Port

```
java.io.IOException: No X-Jenkins-CLI2-Port among [X-Jenkins, null,
Server, X-Content-Type-Options, Connection, X-You-Are-In-Group, X-
Hudson, X-Permission-Implied-By, Date, X-Jenkins-Session, X-You-Are-
Authenticated-As, X-Required-Permission, Set-Cookie, Expires, Content-
Length, Content-Type]
        at hudson.cli.CLI.getCliTcpPort(CLI.java:284)
        at hudson.cli.CLI.<init>(CLI.java:128)
        at hudson.cli.CLIConnectionFactory.connect(CLIConnectionFactory.
java:72)
        at hudson.cli.CLI._main(CLI.java:473)
        at hudson.cli.CLI.main(CLI.java:384)
        Suppressed: java.io.IOException: Server returned HTTP response
code: 403 for URL: http://citest.gce.px/cli
                at sun.net.www.protocol.http.HttpURLConnection.
getInputStream0(HttpURLConnection.java:1840)
                at sun.net.www.protocol.http.HttpURLConnection.
getInputStream(HttpURLConnection.java:1441)
                at hudson.cli.FullDuplexHttpStream.<init>
(FullDuplexHttpStream.java:78)
                at hudson.cli.CLI.connectViaHttp(CLI.java:152)
                at hudson.cli.CLI.<init>(CLI.java:132)
                ... 3 more
```

Solution: Go to Manage Jenkins -> Configure Global Security -> "TCP port for JNLP agents": choose fixed or random