# Jenkins 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.

# java -jar jenkins-cli.jar -s <http://localhost:8080/jenkins/> help --username admin --password admin﻿

## 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 insid

## Download the Java jar

The best place to download the Jenkins CLI jar is from your own Jenkins server. This means you'll have the matching CLI version.

Visit the /cli endpoint, such as <http://myjenkins/cli>, and it will give you a link to download the jar (the /jnlpJars/jenkins-cli.jar endpoint).

Download that jar.

It will also tell you how to view all the commands available:

$ java -jar jenkins-cli.jar -s http://myjenkins help

## Running commands against a secure Jenkins (basic auth or ssh keys)

If your Jenkins is secured, and it probably is so that you control who can create jobs, run jobs, view results, then the command above may fail and ask for credentials.

You have two options: provide --username & --password options to every command or provide -i option and provide the path to the ssh private key that matches the public key you provided to your user account.

For example:

$ java -jar jenkins-cli.jar -s http://myjenkins help --username me --password mypassword

$ java -jar jenkins-cli.jar -s http://myjenkins help -i ~/.ssh/id\_rsa

Sadly, the Jenkins CLI doesn't remember these flags and you have to pass them every single time.

I'm not a security expert, but the former option is the weaker security option. The mypassword value will be stored in your shell history and possibly also exposed over network traffic (since the example <http://myjenkins> uses the non-encrypted http protocol).

# Disable Security in Jenkins for JenkinsCli

# G:\Jenkins\Jenkinscli>java -jar jenkins-cli.jar -s <http://localhost:8080/> help --username admin --password admin ERROR: Bad Credentials. Search the server log for d0c37ac9-a06a-4dc2-aea7-491f9d1154af for more details.

# goto manage jenkins - configure global security. Uncheck enable security and try again. If error still persist try with some diff version.﻿

enkins has a built-in command line interface that allows users and administrators to access Jenkins from a script or shell environment. This can be convenient for scripting of routine tasks, bulk updates, troubleshooting, and more.

The command line interface can be accessed over SSH or with the Jenkins CLI client, a .jar file distributed with Jenkins. The SSH approach is preferred over the CLI client as it is considered more secure.

## Using the CLI

By default Jenkins will boot with a randomly assigned SSH port, which administrators may choose to override in the [Configure System](https://jenkins.io/doc/book/managing/system-configuration/#ssh-server) page. In order to determine the randomly assigned SSH port, inspect the headers returned on a Jenkins URL, for example:

% curl -Lv https://JENKINS\_URL/login 2>&1 | grep 'X-SSH-Endpoint'

< X-SSH-Endpoint: localhost:53801

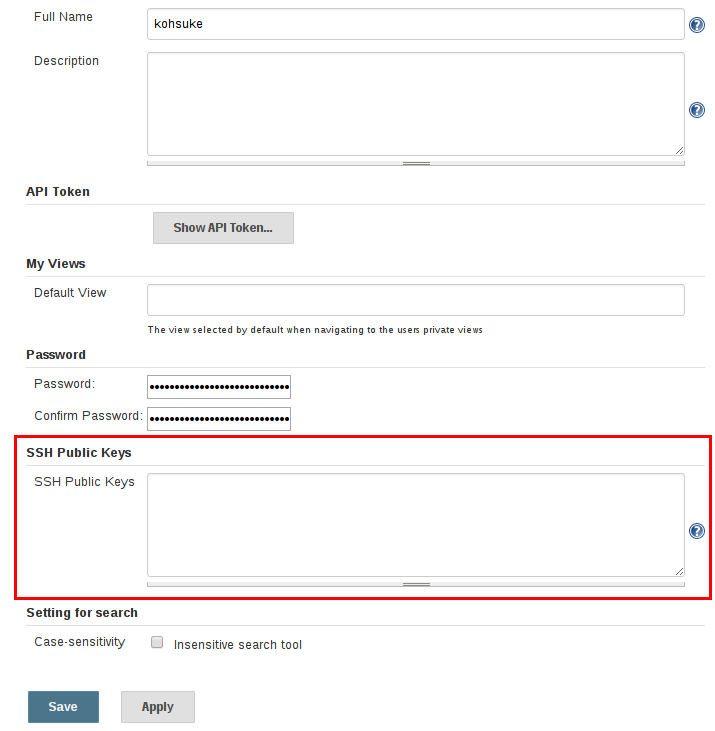
%

With the random SSH port (53801 in this example), and [Authentication](https://jenkins.io/doc/book/managing/cli/#authentication) configured, any modern SSH client may securely execute CLI commands.

### Authentication

Whichever user used for authentication with the Jenkins master must have the Overall/Read permission in order to access the CLI. The user may require additional permissions depending on the commands executed.

Whether using the CLI via SSH, or with the CLI client, both rely primarily on SSH-based public/private key authentication. In order to add an SSH public key for the appropriate user, navigate to https://JENKINS\_URL/user/USERNAME/configure and paste an SSH public key into the appropriate text area.



### Common Commands

------------

## Triggering a job build against a secure Jenkins (avoiding a 1.5 year old open bug!)

To trigger building a job you only need to know its name.

$ java -jar jenkins-cli.jar -s http://myjenkins build 'My Awesome Jenkins Job' -i ~/.ssh/id\_rsa

With a couple of bonus flags, you can have the CLI block and stream all the console output into your terminal:

$ java -jar jenkins-cli.jar -s http://myjenkins build 'My Awesome Jenkins Job' -i ~/.ssh/id\_rsa -s -v

BUT, the above may not immediately work for you. Why? You could read this 1.5 year-old Jenkins ticket ([JENKINS-11024](https://issues.jenkins-ci.org/browse/JENKINS-11024?focusedCommentId=177140)); or you can skip to the workaround with me!

Go to the /configure endpoint (<http://myjenkins/configure>) to edit your global Jenkins settings.

Under "Authorization", you want to turn on "Matrix-based security" and then for the "anonymous" user enable "Read" mode for the "Job" section.

You can now successfully trigger jobs with the commands above!

## Passing parameters when triggering a job build

Job parameters are a very handy concept. Perhaps you've only ever used Jenkins or another CI system to automatically run builds when a remote SCM/git repo changes. You can also trigger builds manually from within Jenkins. And whilst you're doing that, your job can prompt for parameters to the build.

For example, we have a job named similar to "Deploy XYZ App". It has the git repo hardcoded in the job like a normal build, but when you press "Build", it shows a list of options: string fields, drop-down lists, etc.

When the job runs, you can use these values anywhere within your job's configuration. Its very cool.

But how to pass those same parameters via the CLI? You use the -p key=value flag for each parameter you want to pass.

So our finished product might look like:

$ java -jar jenkins-cli.jar -s http://myjenkins build 'Deploy XYZ App' -i ~/.ssh/id\_rsa -s -v -p target\_env=api.cloudfoundry.com -p branch=master

## create new job from templates

In this article, we will talk about how to create a Jenkins job with a template by some commands.

### Instructions

1. Get the Jenkins CLI client package

|  |  |
| --- | --- |
| 1 | $ wget http://jenkins/jnlpJars/jenkins-cli.jar |

2. Check which commands supported by the client package [Exception 1,2]

|  |  |
| --- | --- |
| 1 | $ java -jar jenkins-cli.jar -s http://jenkins/ help |

3. List all jobs under the view: tools

|  |  |
| --- | --- |
| 1 | $ java -jar jenkins-cli.jar -s http://jenkins/ list-jobs tools |

4. Get the configuration of the job: template

|  |  |
| --- | --- |
| 1 | $ java -jar jenkins-cli.jar -s http://jenkins/ get-job template > template.xml |

5. Create a new job based on the configuration

|  |  |
| --- | --- |
| 1 | $ java -jar jenkins-cli.jar -s http://jenkins/ create-job new\_job\_name < new\_job\_name.xml |

6. Run groovy script

|  |  |
| --- | --- |
| 1 | $ java -jar jenkins-cli.jar -s http://jenkins/ groovy scripts/add\_job\_to\_view.groovy |

If there are any parameters in the script, just as:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | import jenkins.model.\*  if (args.length != 2 ) {  println "Error on arguments!"  }  def jobName = args[0] ?: 'a\_job'  def viewName = args[1] ?: 'a\_view'  println jobName + ' ' + viewName  def v = Jenkins.instance.getView(viewName)  def i = Jenkins.instance.getItemByFullName(jobName)  if (v && i) {  v.add(i)  } |

pass the parameters as:

|  |  |
| --- | --- |
| 1 | $ java -jar jenkins-cli.jar -s http://jenkins/ groovy scripts/add\_job\_to\_view.groovy JOB\_NAME VIEM\_NAME |

7. Build a job

|  |  |
| --- | --- |
| 1 | $ java -jar jenkins-cli.jar -s http://jenkins/ build new\_job\_name |

8. Diable a job

|  |  |
| --- | --- |
| 1 | $ java -jar jenkins-cli.jar -s http://jenkins/ disable-job new\_job\_name |

### Exceptions:

1. Description:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | Exception in thread "main" java.io.IOException: No X-Jenkins-CLI2-Port among [null, Vary, Date, Content-Length, Keep-Alive, Set-Cookie, Connection, Content-Type, X-Powered-By, Server]  at hudson.cli.CLI.getCliTcpPort(CLI.java:281)  at hudson.cli.CLI.<init>(CLI.java:128)  at hudson.cli.CLIConnectionFactory.connect(CLIConnectionFactory.java:72)  at hudson.cli.CLI.\_main(CLI.java:449)  at hudson.cli.CLI.main(CLI.java:378)  Suppressed: java.io.EOFException: unexpected stream termination  at hudson.remoting.ClassicCommandTransport.create(ClassicCommandTransport.java:100)  at hudson.remoting.Channel.<init>(Channel.java:392)  at hudson.remoting.Channel.<init>(Channel.java:388)  at hudson.remoting.Channel.<init>(Channel.java:349)  at hudson.remoting.Channel.<init>(Channel.java:345)  at hudson.remoting.Channel.<init>(Channel.java:333)  at hudson.cli.CLI.connectViaHttp(CLI.java:159)  at hudson.cli.CLI.<init>(CLI.java:132)  ... 3 more |

#### Solution: check the port which Jenkins is running on, put 8080 on URL should fix, just as:

|  |  |
| --- | --- |
| 1 | $ java -jar jenkins-cli.jar -s http://jenkins:8080/ help |

2. Description:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | Failed to authenticate with your SSH keys.  hudson.security.AccessDeniedException2: anonymous is missing the ExtendedRead permission  at hudson.security.ACL.checkPermission(ACL.java:54)  at hudson.model.AbstractItem.checkPermission(AbstractItem.java:441)  at hudson.cli.GetJobCommand.run(GetJobCommand.java:46)  at hudson.cli.CLICommand.main(CLICommand.java:229)  at hudson.cli.CliManagerImpl.main(CliManagerImpl.java:92)  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)  at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)  at java.lang.reflect.Method.invoke(Method.java:616)  at hudson.remoting.RemoteInvocationHandler$RPCRequest.perform(RemoteInvocationHandler.java:275)  at hudson.remoting.RemoteInvocationHandler$RPCRequest.call(RemoteInvocationHandler.java:256)  at hudson.remoting.RemoteInvocationHandler$RPCRequest.call(RemoteInvocationHandler.java:215)  at hudson.remoting.UserRequest.perform(UserRequest.java:118)  at hudson.remoting.UserRequest.perform(UserRequest.java:48)  at hudson.remoting.Request$2.run(Request.java:326)  at hudson.remoting.InterceptingExecutorService$1.call(InterceptingExecutorService.java:72)  at hudson.cli.CliManagerImpl$1.call(CliManagerImpl.java:63)  at hudson.remoting.InterceptingExecutorService$2.call(InterceptingExecutorService.java:95)  at java.util.concurrent.FutureTask$Sync.innerRun(FutureTask.java:334)  at java.util.concurrent.FutureTask.run(FutureTask.java:166)  at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1110)  at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:603)  at java.lang.Thread.run(Thread.java:679) |

#### Solution: add client public ssh key to the Jenkins server at: http://jenkins/user/USERNAME/configure | SSH Public Keys.

3. Description:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | hudson.security.AccessDeniedException2: <USERNAME> is missing the Job/ExtendedRead permission  at hudson.security.ACL.checkPermission(ACL.java:54)  at hudson.model.AbstractItem.checkPermission(AbstractItem.java:446)  at hudson.cli.GetJobCommand.run(GetJobCommand.java:46)  at hudson.cli.CLICommand.main(CLICommand.java:234)  at hudson.cli.CliManagerImpl.main(CliManagerImpl.java:92)  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)  at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)  at java.lang.reflect.Method.invoke(Method.java:606)  at hudson.remoting.RemoteInvocationHandler$RPCRequest.perform(RemoteInvocationHandler.java:300)  at hudson.remoting.RemoteInvocationHandler$RPCRequest.call(RemoteInvocationHandler.java:281)  at hudson.remoting.RemoteInvocationHandler$RPCRequest.call(RemoteInvocationHandler.java:240)  at hudson.remoting.UserRequest.perform(UserRequest.java:118)  at hudson.remoting.UserRequest.perform(UserRequest.java:48)  at hudson.remoting.Request$2.run(Request.java:328)  at hudson.remoting.InterceptingExecutorService$1.call(InterceptingExecutorService.java:72)  at hudson.cli.CliManagerImpl$1.call(CliManagerImpl.java:63)  at hudson.remoting.InterceptingExecutorService$2.call(InterceptingExecutorService.java:95)  at jenkins.util.ContextResettingExecutorService$2.call(ContextResettingExecutorService.java:46)  at java.util.concurrent.FutureTask.run(FutureTask.java:262)  at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)  at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)  at java.lang.Thread.run(Thread.java:745) |

#### Solution: Jenkins server –> Manage Jenkins –> Configure Global Security to add related permissions.