Jenkins Script Console

Jenkins Script Console

Jenkins features a nice Groovy script console which allows one to run arbitrary Groovy scripts within the Jenkins master runtime or in the runtime on agents.

> **Security warnings**
>
> It is *very* **important** to understand all of the following points because it affects the integrity of your Jenkins installation. The Jenkins Script Console:
>
> - Access is controlled by the `RunScripts` permission. Older versions of the Matrix Authorization Strategy Plugin allow non-Admin users to be granted this permission. Matrix Authorization Strategy Plugin version 1.5 and later fixed this issue. If any authorization strategy allows this permission to be granted to users other than Admins, then extreme care should be taken not to allow non-admins.
> - Is a web-based Groovy shell into the Jenkins runtime. Groovy is a very powerful language which offers the ability to do practically anything Java can do including:
>   - Create sub-processes and execute arbitrary commands on the Jenkins master and agents.
>   - It can even read files in which the Jenkins master has access to on the host (like `/etc/passwd`)
>   - Decrypt credentials configured within Jenkins.
> - Offers no administrative controls to stop a User (or Admin) once they are able to execute the Script Console from affecting all parts of the Jenkins infrastructure. Granting a normal Jenkins user Script Console Access is essentially the same as giving them Administrator rights within Jenkins.
> - Can configure any Jenkins setting. It can disable security, reconfigure security, even open a backdoor on the host operating system completely outside of the Jenkins process. Due to the mission critical importance many organizations place on Jenkins in their infrastructure this point is especially important because it would allow an attacker to move laterally within infrastructure with little effort.
> - Is so powerful because it was originally intended as a debugging interface for Jenkins developers but has since grown into an interface used by Jenkins Admins to configure Jenkins and debug Jenkins runtime issues.
>
> Because of the power offered by the Jenkins Script Console, Jenkins and its agents should never be run as the `root` user (on Linux) or system administrator on any other flavor of OS. Videos linked in this wiki page demonstrate and discuss security warnings.
>
> **Be sure to secure your Jenkins instance using known good community practices.**

Table of Contents

Multiple contexts

The Jenkins Script Console can run either on the master or any configured agents.

Running Script Console on the master

This feature can be accessed from "Manage Jenkins" > "Script Console". Or visit the sub-URL `/script` on your Jenkins instance.

Running Script Console on agents

Visit "Manage Jenkins" > "Manage Nodes". Select any node to view the status page. In the menu on the left, a menu item is available to open a "Script Console" on that specific agent.

Run scripts from master Script Console on agents

It's also possible to run scripts from the master Script Console on individual agents. The following script is an example running a script on agents from the master Script Console.

Script executes code on agent from Master Script Console

```
import hudson.util.RemotingDiagnostics
import jenkins.model.Jenkins

String agent_name = 'your agent name'
//groovy script you want executed on an agent
groovy_script = '''
println System.getenv("PATH")
println "uname -a".execute().text
'''.trim()

String result
Jenkins.instance.slaves.find { agent ->
    agent.name == agent_name
}.with { agent ->
    result = RemotingDiagnostics.executeGroovy(groovy_script,
agent.channel)
}
println result
```

Remote access

A Jenkins Admin can execute groovy scripts remotely by sending an HTTP POST request to `/script/` url or `/scriptText/`.

curl example via bash

```
curl -d "script=<your_script_here>" https://jenkins/script
# or to get output as a plain text result (no HTML)
curl -d "script=<your_script_here>" https://jenkins/scriptText
```

Also, Jenkins CLI offers the possibility to execute groovy scripts remotely using `groovy` command or execute groovy interactivelly via `groovysh`. However, once again curl can be used to execute groovy scripts by making use of bash Command Substitution. In the following example `somescript.groovy` is a groovy script in the current working directory.

curl submitting groovy file via bash

```
curl --data-urlencode "script=$(< ./somescript.groovy)"
https://jenkins/scriptText
```

If security is configured in Jenkins, then curl can be provided options to authenticate using the `curl --user` option.

curl submitting groovy file providing username and password via bash

```
curl --user 'username:password' --data-urlencode "script=$(<
./somescript.groovy)" https://jenkins/scriptText
```

Here is the equivalent command using python, not curl.

python submitting groovy file providing username and password

```
with open('somescript.groovy', 'r') as fd:
        data = fd.read()
    r = requests.post('https://jenkins/scriptText', auth=('username',
'password'), data={'script': data})
```

Remote access with CSRF protection enabled

There's an extra step which must be performed to configure Jenkins via the Script Console when CSRF Protection is enabled. The extra step is to get a CSRF token. The token provides an extra security measure in Jenkins to ensure the script console is not being configured from an unauthorized source. It basically comes down to a two step process.

1. Authenticate and get a CSRF token for submitting script console scripts.
2. Authenticate and *use the CSRF token* when submitting script console scripts.

Here's an example. Get a CSRF token.

Getting CSRF token and saving it to a bash variable

```
mytoken=$(curl --user 'username:password' -s
https://jenkins/crumbIssuer/api/json | python -c 'import
sys,json;j=json.load(sys.stdin);print j["crumbRequestField"] + "=" +
j["crumb"]')
```

More examples of getting a CSRF token can be found in the Remote access API wiki page.

Then use the `mytoken` environment variable to submit the token along with your authentication to the script console.

curl example submitting script with username, password, and CSRF token

```
curl --user 'username:password' -d "$mytoken" --data-urlencode
"script=$(<./somescript.groovy)" https://jenkins/scriptText
```

Additionally, you can `curl` the root of the Jenkins API to determine if CSRF protection is enabled.

Use curl to determine if CSRF protection is enabled in Jenkins

```
curl --user 'username:password' -s https://jenkins/api/json 2> /dev/null
| python -c 'import sys,json;exec "try:\n  j=json.load(sys.stdin)\n
print str(j[\"useCrumbs\"]).lower()\nexcept:\n  pass"'
```

The above command will return `true` or `false`. If CSRF protection is enabled then it will return `true`.

Shortcut key on script console to submit

You can submit a script without mouse. Jenkins has a shortcut key which enables to submit with keyboard.

- Windows / Linux : Ctrl + Enter
- Mac : Command + Enter

Video Tutorials and additional learning materials

Here are some recorded videos on the Jenkins Script Console:

- Jenkins World 2017: Mastering the Jenkins Script Console - 44 minutes - sample usage and security discussion
- LA Jenkins Area Meetup 2016 - Hacking on Jenkins Internals - Jenkins Script Console - 39 minutes - sample usage

To expand your ability to write script console scripts the following references are recommended:

- Learn Groovy - Learning Groovy is useful for more than writing scripts for the Script Console. Groovy is also relevant for other features of Jenkins like Pipelines and shared pipeline libraries, the Groovy Plugin, the Job DSL plugin, and many other plugins which utilize Groovy (see section "Plugins enabling Groovy usage" in this wiki page).
- Write Groovy scripts for Jenkins with Code completion - The gist of this is to create a Maven project within your IDE and to depend on org.jenkins-ci.main:jenkins-core (and any other plugins that you expect present). You can then write a Groovy script with code completion of Jenkins API objects and methods.

Example Groovy scripts

Out of date scripts

Due to the nature of Groovy scripts accessing Jenkins source code directly, Script Console scripts are easily out of date from the Jenkins source code. It is possible to run a script and get exceptions because public methods and interfaces in Jenkins core or Jenkins plugins have changed. Keep this in mind when trying out examples. Jenkins is easily started from a local development machine via the following command:

Starting a local copy of Jenkins

```
export JENKINS_HOME="./my_jenkins_home"
java -jar jenkins.war
```

Use CTRL+C to stop Jenkins. It is not recommended to try Script Console examples in a production Jenkins instance.

The following repositories offer solid examples of Groovy scripts for Jenkins.

- CloudBees jenkins-scripts repository.
- Jenkins CI jenkins-scripts repository under the `scriptler/` directory (scripts for the Scriptler Plugin).
- Sam Gleske's jenkins-script-console-scripts repository.
- Sam Gleske's jenkins-bootstrap-shared repository under the `scripts/` directory.
- Some scripts at JBoss.org.

Browse all Scriptler Plugin Groovy Scripts and **please share your scripts with the Scriptler Plugin.**

- Activate Chuck Norris Plugin — This script activates Chuck Norris plugin for all jobs in your Jenkins server
- Add a Maven Installation, Tool Installation, Modify System Config
- Add a new label to slaves meeting a condition — This script shows how to alter the slave nodes' label membership. In this case we create a new label if the existing label contains a string. It has been tested from the Jenkins command window.
- Add notification plugin to every job — This script will add the Notification Plugin to every job.
- Allow broken build claiming on every jobs — With the following simple script, you can activate the option on every jobs of your server in just one go.
- Batch-Update Mercurial branch that is checked out — Updates for multiple jobs which branch will be checked out from Hg
- Bulk rename projects
- Change JVM Options in all Maven tasks of Freestyle Jobs — This script find all Maven Tasks registered in freestyle jobs and replace JVM Options by a new value.
- Change publish over SSH configuration
- Change SCMTrigger for each project to disable during the night and the week-end — This script lets you easily change all jobs running every minutes so that it gets disabled between 21:00 and 07:00 and on Saturday and Sunday.
- Change Version-Number in SVN-path
- Clone all projects in a View — This script enumerates all projects belonging to a specific view and clones them.
- Convert standard mail notifications to use the Mail-Ext Publisher plugin — This script replace mail notifications in all projects by Mail-Ext publisher plugin and re-uses existing recipients.
- Delete .tmp files left in workspace-files — This scripts deletes all the tmp files left in workspace-files directory after the build. On windows servers, this seems pretty common so we run this script on daily basis.
- Delete workspace for all disabled jobs — Deletes the workspace for all disabled jobs to save space
- Disable all jobs — This script disables all jobs in your Jenkins server
- Display Information About Nodes — This scripts displays a bunch of information about all the slave nodes.

- Display job parameters — This scripts displays the parameters for all the jobs along with their default values (if applicable).
- Display jobs group by the build steps they use
- Display list of projects that were built more than 1 day ago. — This script to display list of projects that were built more than 1 day ago.
- Display mail notifications recipients — This script displays for all jobs the list of mail recipients used for notifications.
- Display monitors status — Jenkins uses monitors to validate various behaviors. If you dismiss one, Jenkins will never propose you to reactivate it. This script allows you to check the status of all monitors and to reactive them.
- Display the number of jobs using SCM Polling from Freestyle, Pipeline and Maven
- Display timer triggers — This scripts displays the timer triggers for all the jobs in order to better arrange them.
- Display Tools Location on All Nodes — This script can help to get Jenkins tools localtion on all your slaves
- Enable Timestamper plugin on all jobs — With the following simple script, you can activate the option on every jobs of your server in just one go.
- Failed Jobs — This scripts displays a list of all failed jobs. Addon: restart them.
- Find builds currently running that has been executing for more than N seconds
- Grant Cancel Permission for user and group that have Build permission — This script will go through all groups and users in both Global security and per job security settings.
- Invalidate Jenkins HTTP sessions — This script can monitor and invalidate HTTP sessions if there are many open ones on your server.
- Manually run log rotation on all jobs — Runs log rotation on all jobs to free space
- Monitor and Restart Offline Slaves — This script can monitor and restart offline nodes if they are not disconnected manually.
- Monitoring Scripts — Several scripts to display data about http sessions, threads, memory, JVM or MBeans, when using the Monitoring plugin.
- Parameterized System Groovy script — This script will demonstrate how to get parameters in a system groovy script.
- Preselect username in Maven Release Build
- Printing a list of credentials and their IDs
- Remove all disabled modules in Maven jobs — To remove all disabled modules in Maven jobs
- Remove Deployed Artifacts Actions — This script is used to remove the Deployed Artifacts list that is uselessly stored for each build by the Artifact Deployer Plugin.
- Remove Git Plugin BuildsByBranch BuildData — This script is used to remove the static list of BuildsByBranch that is uselessly stored for each build by the Git Plugin.
- Set GitBlitRepositoryBrowser with custum settings on all repos — This scripts allows to update the repo browser. Can be adapted to any other browser, not only gitblit.
- Update maven jobs to use the post build task to deploy artifacts — This script updates all maven jobs having a deploy goal by install and activate the post build step to deploy artifacts at the end of the build
- Update SVN Browser
- Wipe out workspaces of all jobs — This script wipes out the workspaces of all jobs on your Jenkins server
- Wipe workspaces for a set of jobs on all nodes — The script wipes workspaces of certain jobs on all nodes.

Plugins enabling Groovy usage

Page:Config File Provider Plugin — Adds the ability to provide configuration files (i.e., settings.xml for maven, XML, groovy, custom files, etc.) loaded through the Jenkins UI which will be copied to the job's workspace.

Page:Global Post Script Plugin — Execute a global configured groovy script after each build of each job managed by the Jenkins. This is typical for cases when you need to do something based on a shared set of parameters, such as triggering downstream jobs managed by the same Jenkins or remote ones based on the parameters been passed to the parameterized jobs.

Notice: jython script support removed since 1.1.0

Page:Groovy plugin — This plugin adds the ability to directly execute Groovy code.

Page:Groovy Postbuild Plugin — This plugin executes a groovy script in the Jenkins JVM. Typically, the script checks some conditions and changes accordingly the build result, puts badges next to the build in the build history and/or displays information on the build summary page.

Page:Groovy Remote Control Plugin — This plugin provides Groovy Remote Control's receiver, and allows to control external application from Jenkins.

Page:Matrix Groovy Execution Strategy Plugin — A plugin to decide the execution order and valid combinations of matrix projects.

Page:Pipeline Classpath Step Plugin — Pipeline DSL step to add path to the groovy classpath

Page:Scriptler Plugin — Scriptler allows you to store/edit groovy scripts and execute it on any of the slaves/nodes... no need to copy/paste groovy code anymore.

Page:SnowGlobe Plugin — This plugin provides the ability to define Infrastructure as Code. Create, update and tear down clusters of related docker containers for builds, testing or continuous delivery.

Snowglobe plugin for Jenkins

This allows Jenkins jobs to control a SnowGlobe instance (see https://nirima.github.io/SnowGlobe/).

Operations

The operations are relatively simple:

Clone

```
snowglobe_clone createAction: true, sourceId: 'ci-template', targetId: 'new-globe-name'
```

Set Variables

```
snowglobe_set_variables globeId: 'my-globe', variables: 'key="value"'
```

Get Variables

```
data = snowglobe_get_variables  globeId: 'my-globe'
```

Apply

```
snowglobe_apply createAction: true, globeId: 'my-globe'
```

State

```
data = snowglobe_state createAction: false, globeId: 'my-globe'
```

Destroy

```
snowglobe_clone remove: true, globeId: 'my-globe'
```

Remove: set to true to also remove the SnowGlobe after destruction.

In all cases - createAction controls whether to add an action to the build, which will also remove the globe when the CI build is complete.