

# CheatSheet: Jenkins & Groovy

## LANGUAGES

- PDF Link: [cheatsheet-jenkins-groovy-A4.pdf](#), Category: languages
- Blog URL: <https://cheatsheet.dennyzhang.com/cheatsheet-jenkins-groovy-A4>
- Related posts: [Jenkins CheatSheet](#), [#denny-cheatsheets](#)

File me Issues or star this repo.

## 1.1 Jenkins Pipeline

Name	Comment
Specify parameter to run jobs	<code>build job:'job1', parameters:[string(name:'name1', value:va1)]</code>
Run job in different agents	<code>node(\$agent_label) {...}</code>
Ask for user input	<code>stage('stage2'){ input "OK to go?" }</code>
Actively fail current pipeline job	<code>error("Build failed because of this and that..")</code>
Check whether property exists	<code>if (env.keep_failed_env)</code>
Jenkins Pipeline enable timestamps	<code>options{timestamps()}</code>
Set envs within a jenkins pipeline	<code>withEnv(["key1=\$var1"])</code>
Install plugin via groovy	<code>Hudson.instance.updateCenter.getPlugin(plugin).deploy().get()</code>
Keep previous job run via groovy	<code>buildDiscarder(logRotator(daysToKeepStr: '20', numToKeepStr: '60'))</code>
Keep going when previous stage has failed	<code>keep-going-with-errors.groovy</code>
Send slack notification in pipeline	<code>slack-notification.groovy</code>
Pass parameter across jenkins jobs	<code>jenkinsfile-pass-parameter.groovy</code>
Set timeout & retry	<code>jenkinsfile-timeout-retry.groovy</code>
Use finally to do cleanup	<code>jenkinsfile-finally.groovy</code>
Run jenkins jobs in a sequential way	<code>jenkinsfile-sequentially.groovy</code>
Run jenkins jobs in parallel	<code>jenkinsfile-parallelly.groovy</code>
Reference	Link: Syntax Reference, Link: Jenkins User Documentation
Reference	Link: Groovy Language Documentation
Reference	Link: Example, Link: Example

## 1.2 Config Jenkins Via Groovy

Name	Comment
Set timezone for jenkins	<code>timezone.groovy</code>
Set timezone for jenkins	<code>System.setProperty('org.apache.commons.jelly.tags.fmt.timeZone', 'America/Los_Angeles')</code>
Configure default view	<code>jenkins-views.groovy</code>
Configure Jenkins url	<code>jenkins-url.groovy</code>
Create a Jenkins user	<code>create-jenkins-user.groovy</code>
Groovy manages files/folders	<code>files-folder.groovy</code>
Configure max executors in Jenkins	<code>master-executors.groovy</code>
Configure slack plugin	<code>config-slack.groovy</code>
Configure pipeline shared libraries	<code>config-pipeline-library.groovy</code>
Get jenkins version from CLI	<code>java -jar /usr/share/jenkins/jenkins.war --version</code>
Reference	GitHub: <a href="#">cloudbees/jenkins-scripts</a> , GitHub: <a href="#">jenkinsci/pipeline-examples</a>

## 1.3 Jenkins Kubernetes Via Groovy

Name	Comment
Config jenkins kubernetes plugin	<code>jenkins-kubernetes-cloud.groovy</code>
Validate Kubernetes jenkins setup	<code>validate-kubernetes-cloud.groovy</code>
Kubernetes run with envs configured	<code>runWithEnvVariables.groovy</code>
Reference	GitHub: <a href="#">kubernetes-plugin pipeline examples</a>

## 1.4 Jenkins View Via Groovy

Name	Comment
Add a list of jobs by regexp to a view	<code>myView.setIncludeRegex(".*Integration.*"), addjobstoview-byregexp.groovy</code>
Create jenkins views and add jobs to it	<code>jenkins-views.groovy</code>
Add a view of build monitor view plugin	<code>build-monitor-views.xml</code>
Change view description in groovy	<code>myView.doSubmitDescription</code>

## 1.5 Jenkins Job Via Groovy

Name	Comment
List all my jenkins jobs	<code>println Jenkins.instance.projects.collect { it.name }</code>
List all jenkins jobs	<code>list-all-jobs.groovy</code>
Create and trigger a job	<code>create-jenkins-job.groovy</code>
Manage jenkins jobs	<code>manage-jenkins-jobs.groovy</code>
Cancel queued jenkins jobs by regexp	<code>kill-queued-jenkins.groovy</code>
Support HTML for job and parameter descriptions	Link: OWASP Markup Formatter Plugin

## 1.6 Jenkins Different Parameters

Name	Comment
string	<code>string(name: 'key1', defaultValue: 'Default value', description: 'some parameter')</code>
text	<code>text(name: 'key1', defaultValue: 'Default value', description: 'some parameter')</code>
boolean	<code>booleanParam(name: 'key1', defaultValue: true, description: 'some parameter')</code>
choice	<code>choice(name: 'key1', choices: 'One\nTwo\nThree\n', description: 'some parameter')</code>
password	<code>password(name: 'key1', defaultValue: 'SECRET', description: 'Enter a password')</code>
file	<code>file(name: 'key1', description: 'Choose a file to upload')</code>

## 1.7 Jenkins Security Via Groovy

Name	Comment
logged-in users can do anything	<code>logged-in-users.groovy</code>
Enable ldap in Jenkins	<code>enable-ldap.groovy</code>
Create a jenkins secret text	<code>create-secret-text.groovy</code>
Configure authorization in Jenkins	<code>matrix-authorization-strategy.groovy</code>
Jenkins skip wizzard when initialization	<code>-Djenkins.install.runSetupWizard=false</code>
Jenkins skip wizzard when initialization	<code>instance.setInstallState(InstallState.INITIAL_SETUP_COMPLETED)</code>
Slave To Master Access Control	<code>00-slave-to-master-access.groovy</code>
CSRF Protection	<code>00-csrf.groovy</code>
Add Jenkins permission	<code>jenkins-permission.groovy</code>
Disable CLI over Remoting	<code>00-disable-cli-remoting.groovy</code>
Disable jnlp	<code>jenkins.setSlaveAgentPort(-1)</code>
Access Control for Builds	<code>jenkins.security.QueueItemAuthenticatorConfiguration.xml</code>

## 1.8 Load Jenkins settings via folder copy

Name	Comment
Add default jobs	<code>Copy jobs/ /usr/share/jenkins/ref/jobs/</code>
Copy custom built plugins	<code>COPY plugins/*.hpi /usr/share/jenkins/ref/plugins/</code>
Use jenkins cli	<code>COPY config/jenkins.properties /usr/share/jenkins/ref/</code>
Add jenkins groovy scripts	<code>COPY config/*.groovy /usr/share/jenkins/ref/init.groovy.d/</code>
Configure Jenkins with some defaults	<code>COPY config/*.xml /usr/share/jenkins/ref/</code>
Install jenkins plugins	<code>/usr/local/bin/install-plugins.sh &lt; /usr/share/jenkins/ref/plugins.txt</code>

## 1.9 Jenkins Plugins

Plugin	Summary
Kubernetes Plugin	Jenkins plugin to run dynamic agents in a Kubernetes/Docker environment
Credentials Plugin	Load the ssh key
SiteMonitor Plugin	Monitor URLs
Timestamper Plugin	Add timestamp to job output
Dashboard View Plugin	Create dashboard
Log Parser Plugin	Parse the console output and highlight error/warning/info lines.
Build-timeout Plugin	Abort if job takes too long
Naginator Plugin	Retry failed a job
ThinBackup Plugin	Backup jenkins
JobConfigHistory Plugin	Backup job configuration
"Anything Goes" formatter	use JavaScript inside your project description
AnsiColor Plugin	Add support for ANSI escape sequences, including color, to Console Output
Build User Vars Plugin	Describe the user who started the build

## 1.10 Jenkins Git Via Groovy

Name	Comment
Git checkout code	git-checkout.groovy
Get all git commits since last success	git-commits-before-fail.groovy
List git tags and branches	git-list-tags-and-branches.groovy

## 1.11 Jenkins networking Via Groovy

Name	Comment
Get hostname	<code>println InetAddress.getLocalHost().canonicalHostName</code>
Get IP address	<code>println InetAddress.getLocalHost().hostAddress</code>
Get hostname by ip	<code>get-ip-by-hostname.groovy</code>
validate user input: ip address	<code>assert ip_address.matches("\\d{1,3}\\. \\d{1,3}\\. \\d{1,3}\\. \\d{1,3}")</code>

## 1.12 Jenkins with Kubernetes/Docker

Name	Comment
Kubernetes Plugin	Jenkins plugin to run dynamic agents in a Kubernetes/Docker environment
Config jenkins kubernetes plugin	<code>jenkins-kubernetes-cloud.groovy</code>
Cleanup for Docker stale containers/images/volumes	<code>docker-cleanup.groovy</code>

## 1.13 Groovy Common Errors/Exceptions

Name	Comment
Illegal class name	JVM doesn't like class names with a hyphen

## 1.14 Groovy Basic

Name	Comment
Get environment variables	<code>get-env.groovy</code> , <code>println env.WORKSPACE</code>
Groovy execute command	<code>execute-command.groovy</code>
Get data type of a variable	<code>myObject.getClass()</code>
Print stdout	<code>print.groovy</code> echo 'Action is done', <code>println "Hello World"</code>
Use boolean parameter	<code>if (istru == "false") {...}</code>
Basic integer caculation	<code>def a = 3, b = 7; println "\$a + \$b = \${a + b}"</code>
Run groovy online	SaaS: Groovy Web console
Run groovy script from Jenkins	Link: Jenkins Script Console
Reference	Link: Apache Groovy

## 1.15 Groovy String/Regex

Name	Comment
Check string startsWith	<code>assert s.startsWith("\t")</code>
Trim whitespaces	<code>s=s.trim()</code>
Concat string	<code>first = 'Joe'; last = 'Smith'; println("Name: \$first \$last")</code>
Convert list to string	<code>l.join(";")</code>
Create string with multi-lines	<code>multi-line-string.groovy</code>
Convert string to list	<code>split-string.groovy</code>
Convert string to json	<code>string-to-json.groovy</code>
Remove tags	<code>input.replaceAll("\\&lt;.*?&gt;", "")</code>
Regex match	<code>regex-match.groovy</code>
Regex case insensitive	<code>(item.name == ~/(?i).*NSX.* /)</code>
Reference	Regular Expressions in Groovy

## 1.16 Groovy Array

Name	Comment
Iterate a list	<code>for(item in [1,2,3,4]){ println item }</code>
Iterate a list	<code>(1..3).each { println "Number \${it}" }</code>
Add item to list	<code>def alist = [10, 9, 8]; alist &lt;&lt; 7</code>
List size	<code>def alist = [10, 9, 8]; alist.size()</code>
Split string with delimiter	<code>'1128-2'.tokenize('-')</code>

## 1.17 Groovy File

Name	Comment
Read file into a string	<code>String fileContents = new File('/tmp/test.txt').text</code>
Read file content as a variable	<code>def env = System.getenv(), def content = readFile("/tmp/test.txt")</code>
Write file in pipeline	<code>writeFile file: "output/my.txt", text: "This is a test"</code>
Read a property file	<code>def conf = readProperties file: "\${env.WORKSPACE}@script/my.properties"</code>
Read and write json files	<code>json-file.groovy</code>
Obtain a relative path	<code>relative-path.groovy</code>

## 1.18 Groovy Shell Command

Name	Comment
Run shell and get output	<code>def out = sh script: command, returnStdout: true</code>
Run shell and get exit code	<code>def status = sh script: command, returnStatus: true</code>

## 1.19 Groovy Dictionary

Name	Comment
Create a map	<code>def m = ['fruit':'Apple', 'veggie':'Carrot']</code>
Add an item to map	<code>m.put('denny','hello')</code>
Check if key exists	<code>m.containsKey('key1')</code>
Loop a map	<code>loop-map.groovy</code>

## 1.20 Groovy json

Name	Comment
Convert string to json	<code>string-to-json.groovy</code>
Convert dictionary to json	<code>dict-to-json.groovy</code>
Read and write json files	<code>json-file.groovy</code>

## 1.21 Groovy Date

Name	Comment
Date to string	<code>new Date().format("yyyy-MM-dd'T'HH:mm:ss'Z'", TimeZone.getTimeZone("UTC"))</code>
String to date	<code>Date.parse("yyyy-MM-dd'T'HH:mm:ss'Z'", "2001-01-01T00:00:00Z")</code>
String to date	<code>Date.parse("yyyy-MM-dd'T'HH:mm:ssZ", "2001-01-01T00:00:00+0000")</code>

## 1.22 Jenkins Agent

Name	Comment
Check jenkins slave jar version	check-slave-jar-version.groovy
Find dead executors and remove them	find-dead-executors.groovy
Set env for each agent	set-agent-env.groovy

## 1.23 Jenkins Maintenance

Name	Comment
Delete jenkins job by regexp	delete-job-by-regexp.groovy
Deploy Jenkins via docker	<a href="https://hub.docker.com/r/jenkins/jenkins/">https://hub.docker.com/r/jenkins/jenkins/</a>
Clean up old builds	Link: CloudBees Best Strategy for Disk Space Management

## 1.24 More Resources

<http://groovy-lang.org/documentation.html#gettingstarted>

<https://github.com/fabric8io/jenkins-docker>

License: Code is licensed under MIT License.