



JavaCro18

Groovy for cloud on steroids


# Subjective opinion

Information from this report is my subjective opinion based on my experience, knowledge, mistakes...

[https://habr.com/hub/groovy\\_grails/authors/](https://habr.com/hub/groovy_grails/authors/)

times Тостер Мой круг Фрилансим Мегалопсты: Тест Puzzle English Кибервойны I Карьера в ИБ-гиганте

**habr** Публикации Пользователи Хабы Компании Песочница




 **GRAILS**

**Groovy & Grails**  
Язык программирования Groovy и фреймворк Grails

ВСЕ ПОРЯД ЛУЧШИЕ **АВТОРЫ**

Найти пользователя

Имя Вклад в хаб

	<b>Игорь @igor_sukhorukov</b> 6 лет на сайте Взаимные преобразования JSON, YAML, XML, 04.04.2018 в 12:58	167,0
	<b>Сергей Бушняк @5Sigrtami</b> 6 лет на сайте	110,0
	<b>Барух Садогурский @jbaruch</b> 7 лет на сайте Проект Lazybones — «Лентяй», который работает за вас, 04.04.2014 в 04:24	92,0

Igor Sukhorukov  
Groovy for cloud on steroids



# Steroids prescription

GitHub: <https://github.com/igor-sukhorukov/groovy-grape-aether>

Maven: <https://repo1.maven.org/maven2/com/github/igor-sukhorukov/groovy-grape-aether/2.4.15/groovy-grape-aether-2.4.15.jar>

Docker Hub: suhorukov/docker-groovy

<https://hub.docker.com/r/sukhorukov/docker-groovy/>

# Groovy: run script from S3/HDFS

```
java -jar groovy-grape-aether-2.4.15.jar camel:/aws-s3:your_bucket  
?fileName=file_key&deleteAfterRead=false&region=US_WEST_2&useDefaultAWSCred  
entialsProviderChain=true
```

<http://camel.apache.org/components.html>

# Groovy: run script from Maven repository

```
java -jar groovy-grape-aether-2.4.15.jar  
mvn:/groupId:artifactId[:extension[:classifier]]:version[?custom_repository_URL]
```

# java.net.URL resolvers

## 1. StreamHandlerFactory

`java.net.URL#setURLStreamHandlerFactory`

## 2. `-Djava.protocol.handler.pkgs=...`

`String className = packagePrefix + "." + protocol + ".Handler";`

## 3. (JDK $\geq$ 9) `java.net.spi.URLStreamHandlerProvider`

# Groovy: docker

```
docker run -d -p 8080:8080 suhorukov/docker-groovy URL(script-path)
```

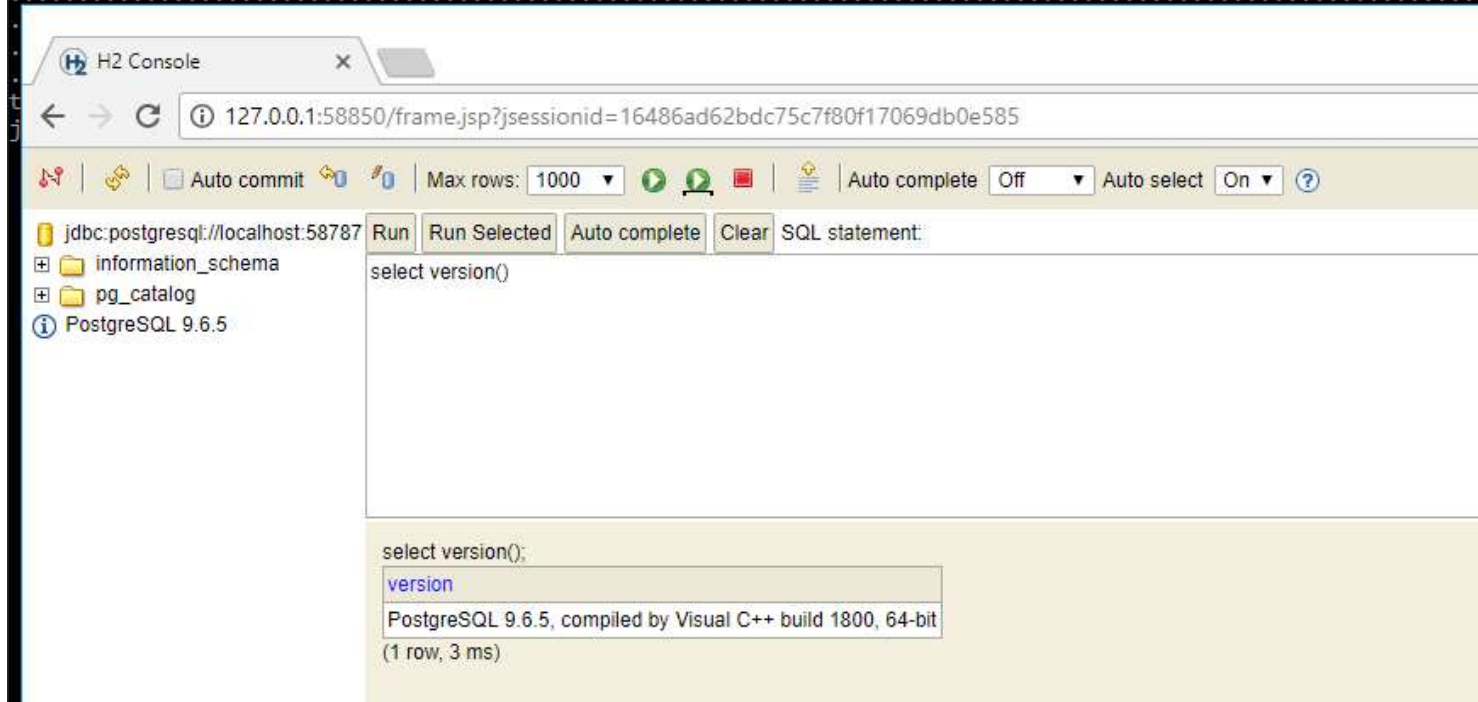
The screenshot shows a web browser window displaying the Docker Hub search results for 'groovy'. The search results are sorted by 'Downloads' and show a list of repositories. The top repository is 'groovy/official' with 40 stars and 10M+ pulls. Other repositories include 'niaquinto/gradle', 'trollin/groovy', 'dpatriot/docker-s3-groovy-runner', 'suhorukov/docker-groovy', and 'amd64/groovy'.

Repository	Stars	Pulls	Details
groovy/official	40	10M+	> DETAILS
niaquinto/gradle public   automated build	14	1M+	> DETAILS
trollin/groovy public	0	100K+	> DETAILS
dpatriot/docker-s3-groovy-runner public   automated build	0	50K+	> DETAILS
suhorukov/docker-groovy public   automated build	0	10K+	> DETAILS
amd64/groovy public	0	10K+	> DETAILS

# Grape: dependency manager embedded into Groovy

## Portable PostgreSQL server with H2 web console

```
Resolving dependencies.....
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/C:/Users/isukhorukov/.m2/repository/org/slf4j/slf4j-log4j12/1.7.13/slf4j-log4j12-1.7.13.jar]
SLF4J: Found binding in [jar:file:/C:/Users/isukhorukov/.m2/repository/org/slf4j/slf4j-jdk14/1.7.13/slf4j-jdk14-1.7.13.jar]
SLF4J: Found binding in [jar:file:/C:/Users/isukhorukov/.m2/repository/ch/qos/logback/logback-classic/1.1.3/logback-classic-1.1.3.jar]
SLF4J: Found binding in [jar:file:/C:/Users/isukhorukov/.m2/repository/org/slf4j/slf4j-simple/1.6.6/slf4j-simple-1.6.6.jar]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Extract C:\Users\isukhorukov\embedpostgresql\postgresql-9.6.5-1-windows-x64-binaries.zip START
.....
```



The screenshot shows the H2 Console web interface. The browser address bar displays the URL `127.0.0.1:58850/frame.jsp?jsessionid=16486ad62bdc75c7f80f17069db0e585`. The interface includes a toolbar with options for `Auto commit`, `Max rows: 1000`, `Auto complete: Off`, and `Auto select: On`. The left sidebar shows the database structure with `jdbc:postgresql://localhost:58787` and `PostgreSQL 9.6.5`. The main area shows the SQL statement `select version();` and the execution result: `version`, `PostgreSQL 9.6.5, compiled by Visual C++ build 1800, 64-bit`, and `(1 row, 3 ms)`.



# Grape: dependency manager embedded into Groovy

<http://docs.groovy-lang.org/latest/html/documentation/grape.html>

```
java -jar groovy-grape-aether-2.4.15.jar https://raw.githubusercontent.com/igor-suhorukov/database-quickstart/master/postgresql.groovy
```

```
@Grab('com.h2database:h2:1.4.197')
@Grab('com.github.igor-suhorukov:postgresql-runner:2.5')
@Grab('org.slf4j:slf4j-simple:1.6.6')
import com.github.igorsuhorukov.postgresql.PostgresqlService;

def postgresqlService = new PostgresqlService()
postgresqlService.start()
addShutdownHook {
    println 'shutdown postgresql server'
    postgresqlService.close()
}

String[] h2Args = ['-url', postgresqlService.getJdbcConnectionUrl(), '-driver', 'org.postgresql.Driver']
org.h2.tools.Console.main(h2Args)
```

# Maven Grape resolver

```
@Grab(group='org.codehaus.plexus', module='plexus-archiver', version='2.10.2')
import org.codehaus.plexus.archiver.tar.TarGZipUnArchiver
import com.github.igorsuhorukov.smreed.dropship.MavenClassLoader;
@Grab(group='org.codehaus.plexus', module='plexus-container-default', version='1.6')
import org.codehaus.plexus.logging.console.ConsoleLogger;
```

```
@Grab(group='org.codehaus.plexus', module='plexus-archiver', version='2.10.2')
import org.codehaus.plexus.archiver.tar.TarGZipUnArchiver
import com.github.igorsuhorukov.smreed.dropship.MavenClassLoader;
@Grab(group='org.codehaus.plexus', module='plexus-container-default', version='1.6')
import org.codehaus.plexus.logging.console.ConsoleLogger;

def artifact = 'crate'
def version = '0.54.1'

def userHome= System.getProperty('user.home')
def destDir = new File("${userHome}/.crate-io")

def crateIoDir= new File(destDir, "${artifact}-${version}");
if(!crateIoDir.exists()){
    destDir.mkdirs()
    String sourceFile = MavenClassLoader.using("https://dl.bintray.com/crate/crate/").getArtifactUrlsCollection("io.crate:${artifact}:tar.gz:${version}",
null).get(0).getFile()
    final TarGZipUnArchiver unArchiver = new TarGZipUnArchiver()
    unArchiver.setSourceFile(new File(sourceFile))
    unArchiver.enableLogging(new ConsoleLogger(ConsoleLogger.LEVEL_DEBUG,"Logger"))
    unArchiver.setDestDirectory(destDir)
    unArchiver.extract()

    def crateCfg = new File("${crateIoDir.absolutePath}/config/crate.yml")
    crateCfgText = crateCfg.text
    crateCfg.withWriter { w ->
        w << crateCfgText.replace('# es.api.enabled: false', 'es.api.enabled: true')
    }
}

def proc = "${crateIoDir.absolutePath}/bin/crate".execute()

proc.consumeProcessOutput(System.out, System.err)
proc.waitFor()
```

<https://habr.com/post/274315/>

# Maven ClassLoader / ServiceLoader

Chromedriver(native part) loading from maven repository

```
def chromedriver =  
MavenClassLoader.usingCentralRepo().resolveArtifact("com.github.igor-  
suhorukov:chromedriver:bin:linux64:2.34").getFile()  
  
System.setProperty("webdriver.chrome.driver", chromedriver)
```

Maven service loader:

```
import com.github.igorsuhorukov.service.MavenServiceLoader  
  
def drivers = MavenServiceLoader.loadServices("mysql:mysql-connector-java:8.0.11",  
Driver.class)  
  
def connection = drivers.iterator().next().connect("jdbc:mysql://localhost/test", null)
```

<https://habr.com/post/317578/>

# Maven ClassLoader

Dynamic fetch and run hawtio console from Maven repository

## Groovy:

```
def hawtloConsole =  
MavenClassLoader.usingCentralRepo().forMavenCoordinates('io.hawt:hawtio-  
app:2.0.0').loadClass('io.hawt.app.App')  
  
Thread.currentThread().setContextClassLoader(hawtloConsole.getClassLoader())  
  
hawtloConsole.main('--port', '10090')
```

## Java:

```
Class<?> hawtloConsole =  
MavenClassLoader.usingCentralRepo().forMavenCoordinates("io.hawt:hawtio-  
app:2.0.0").loadClass("io.hawt.app.App");  
Thread.currentThread().setContextClassLoader(hawtloConsole.getClassLoader());  
Method main = hawtloConsole.getMethod("main", String[].class); main.invoke(null,  
(Object) new String[]{"--port", "10090"});
```

# Maven ClassLoader: hawtio web console

The screenshot displays the hawtio web console interface. The browser address bar shows the URL: `192.168.1.68:10090/hawtio/dashboard/id/4d5ff553f5982181d3?tab=dashboard`. The console has tabs for Camel, Connect, Dashboard, JMX, Logs, and Threads. The Dashboard tab is active, and the JVM sub-tab is selected. The interface shows two main panels:

**CPU Load**

Time	Process cpu load	System cpu load	System load average
10:17			
10:18			
10:19			
10:20			
10:21			
10:22	0.020	0.020	0.23
10:23			
10:24			
10:25			
10:26			

**Size**

Time	Free physical memory size	Free swap space size	Total physical memory size
10:17			
10:18			
10:19			
10:20			
10:21			
10:22	210M	100M	970M
10:23			
10:24			
10:25			
10:26			

**Operating System Attributes**

Property	Value
Arch	arm
Available processors	4
Committed virtual memory size	402976768
Free physical memory size	209702912
Free swap space size	104853504
Max file descriptor count	65536
Name	Linux
Object name	java.lang:type=OperatingSystem
Open file descriptor count	252
Process cpu load	0.5
Process cpu time	92090000000
System cpu load	0.5
System load average	0.23
Total physical memory size	970485760

# Read binary data from webcam

```
import groovy.swing.SwingBuilder

import javax.imageio.ImageIO
import javax.swing.*

def swing = new SwingBuilder()
swing.edt {
    frame(title: 'Webcam protocol', defaultCloseOperation: JFrame.EXIT_ON_CLOSE, pack:
true, show: true) {
        vbox {
            swing.panel() {
                def webcamStream = new URL('camel:/webcam:spycam?resolution=HD720').openStream()

                label(new JLabel(new ImageIcon(ImageIO.read(webcamStream))))
            }
        }
    }
}
```



## Write result in clouds

S3  
HDFS  
FTPS  
STOMP XMPP

```
def connection = new
URL("camel:/file:..?fileName=url_out.txt").openConnection()
def resultStream = connection.getOutputStream()
resultStream.withWriter { Writer writer ->
    writer << "hello world"
}
```

# Distributed locks, executors, queue etc.

## Hazelcast

<http://docs.hazelcast.org/docs/latest-dev/manual/html-single/index.html#distributed-data-structures>

`java.util.concurrent.locks.Lock`

`java.util.concurrent.Semaphore`

`java.util.concurrent.CountDownLatch`

`java.util.concurrent.atomic.AtomicLong`

`java.util.concurrent.BlockingQueue`

`java.util.concurrent.ExecutorService`



## Groovy in legacy environments. Maven repository

Maven repository became de facto standard in enterprise environment. So maven Grape resolver leverage existing infrastructure settings from `~/.m2/settings.xml` and `~/.m2/settings-security.xml`.

It is possible to override maven settings by jvm `-D...` system properties

`mavenSettings.offline=true`

`mavenSettings` — path to `settings.xml`

`mavenSettingsSecurity` — path to `settings-security.xml`

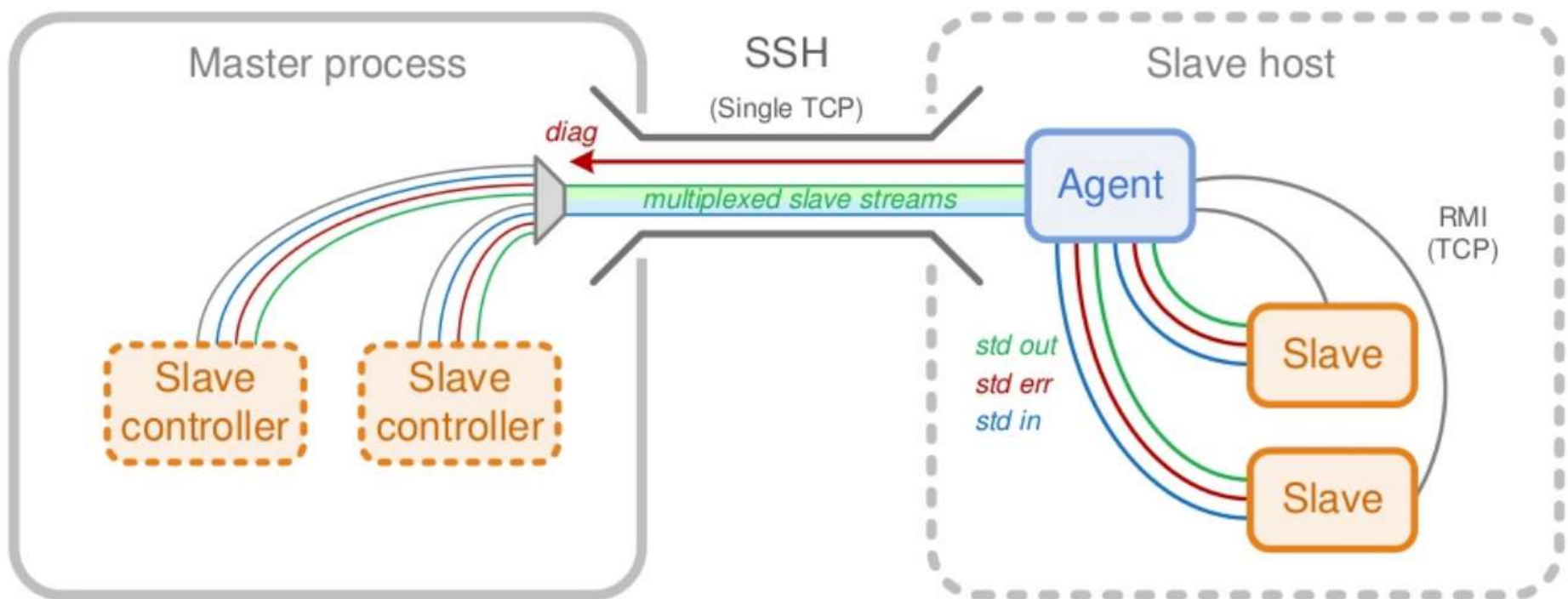
# Groovy in legacy environments. SSH

Java RMI over SSH

# Groovy in legacy environments. Nanocloud

Java RMI over SSH

<https://github.com/gridkit/nanocloud>



# Groovy in legacy environments. Nanocloud

NanoCloud requirements:

- SSHd
- Java present

Works though NAT and firewalls

Works on Amazon EC2

Works everywhere where SSH works

## Groovy in legacy environments. Nanocloud

Master JVM kills slave processes, unless

- SSH session was interrupted
- someone kill -9 master JVM
- master JVM has crashed (e.g. under debugger)

Death clock is ticking on slave though

- if master is not responding
- slave process will terminate itself



# JavaCro18

Thank you!

igor.suhorukov@gmail.com  
github.com/igor-suhorukov



GOOD NIGHT,  
PROGRAMMERS!

{turnoff.us}