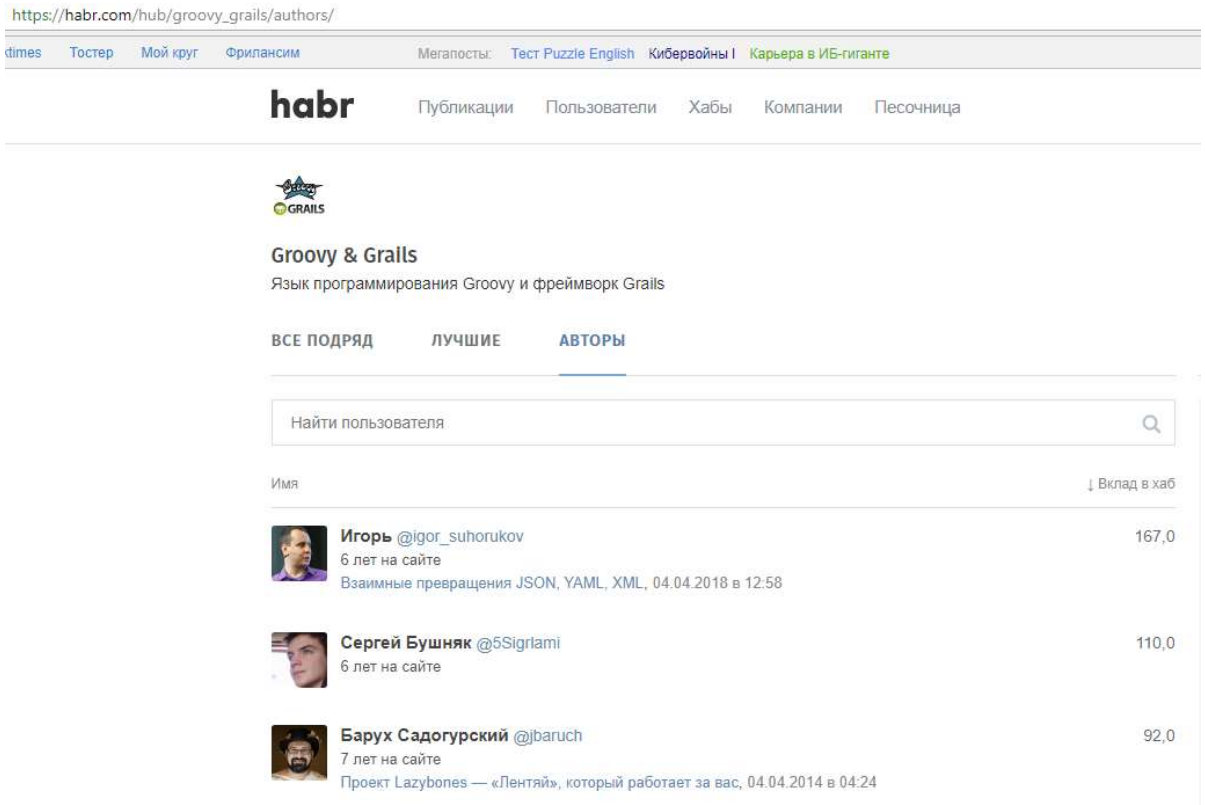
JavaCro'18

Groovy for cloud on steroids

Subjective opinion

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



Steroids prescription

GitHub: https://github.com/igor-suhorukov/groovy-grape-aether

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

Docker Hub: suhorukov/docker-groovy

https://hub.docker.com/r/suhorukov/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&useDefaultAWSCredentialsProviderChain=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

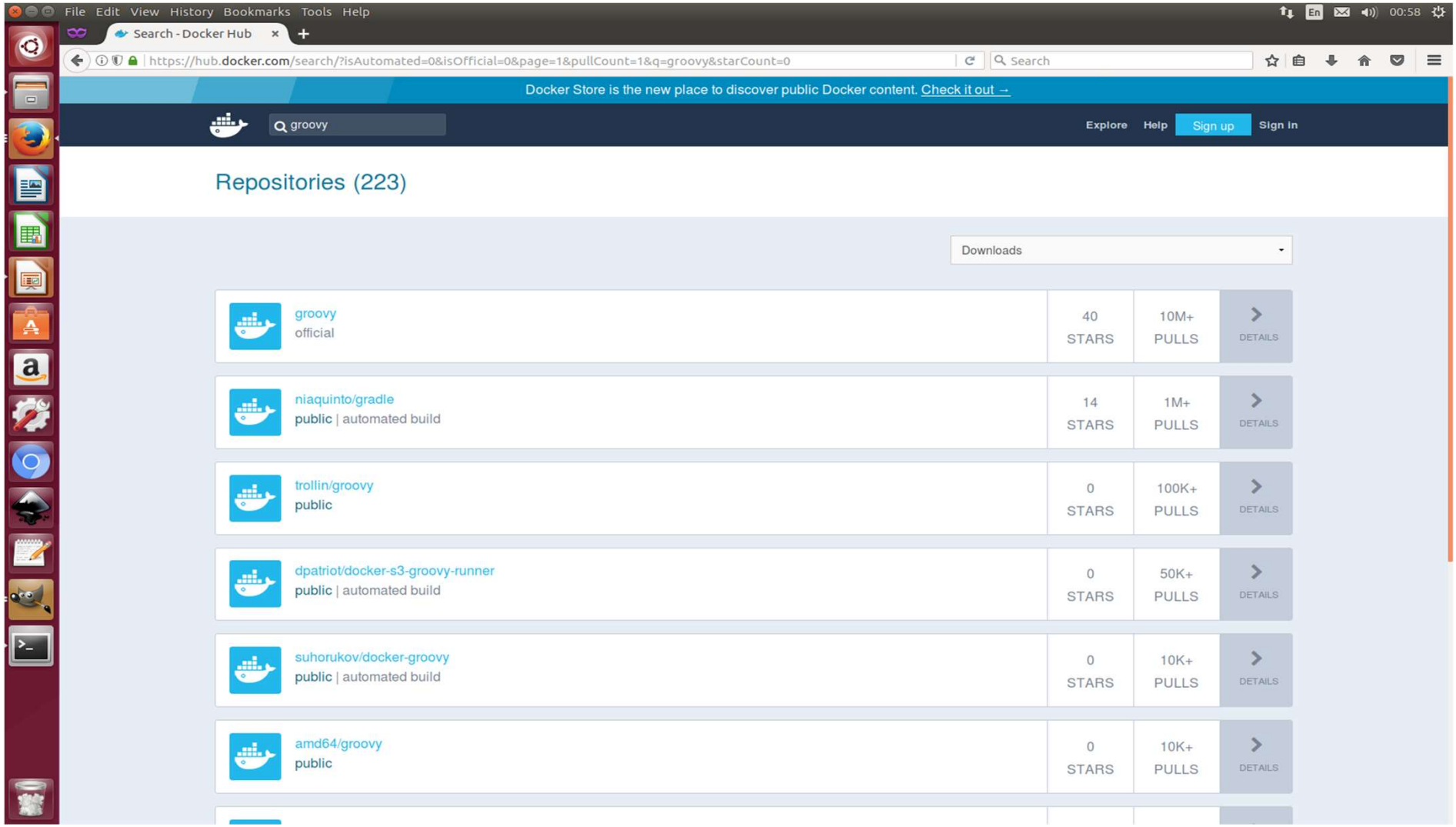
1. StreamHandlerFactory
   java.net.URL#setURLStreamHandlerFactory

2. -Djava.protocol.handler.pkgs=...
String className = packagePrefix + "." + protocol
+".Handler";

3. (JDK >= 9) java.net.spi.URLStreamHandlerProvider

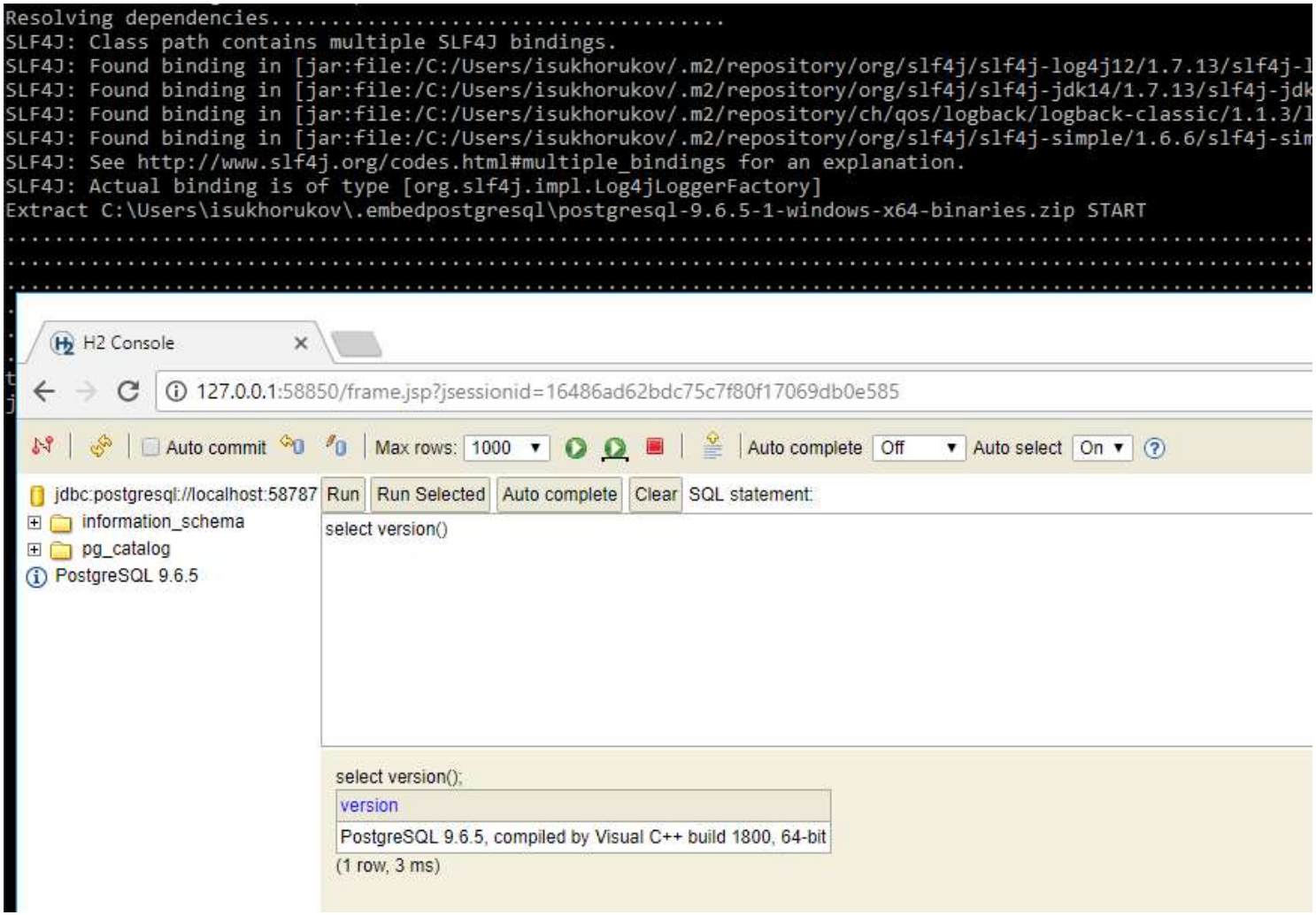
Groovy: docker

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



Grape: dependency manager embedded into Groovy

Portable PostgreSQL server with H2 web console



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

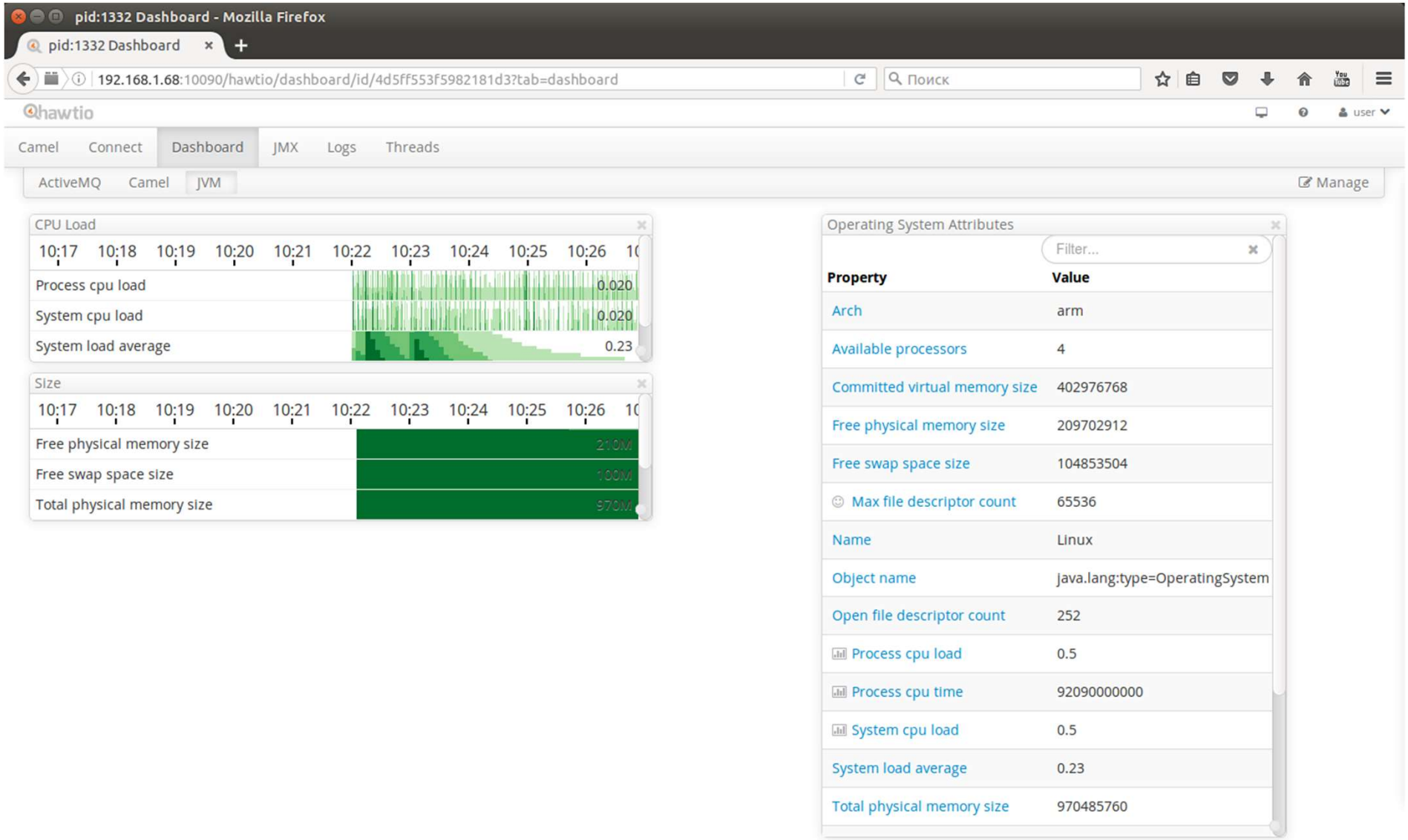
Groovy:

```
def hawtIoConsole = MavenClassLoader.usingCentralRepo().forMavenCoordinates('io.hawt:hawtio-app:2.0.0').loadClass('io.hawt.app.App')

Thread.currentThread().setContextClassLoader(hawtIoConsole.getClassLoader())

hawtIoConsole.main('--port','10090')
```

Java:

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

Maven ClassLoader: hawtio web console



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

Write result in clouds

FTPS

S3

STOMP

XMPP

HDFS

```
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

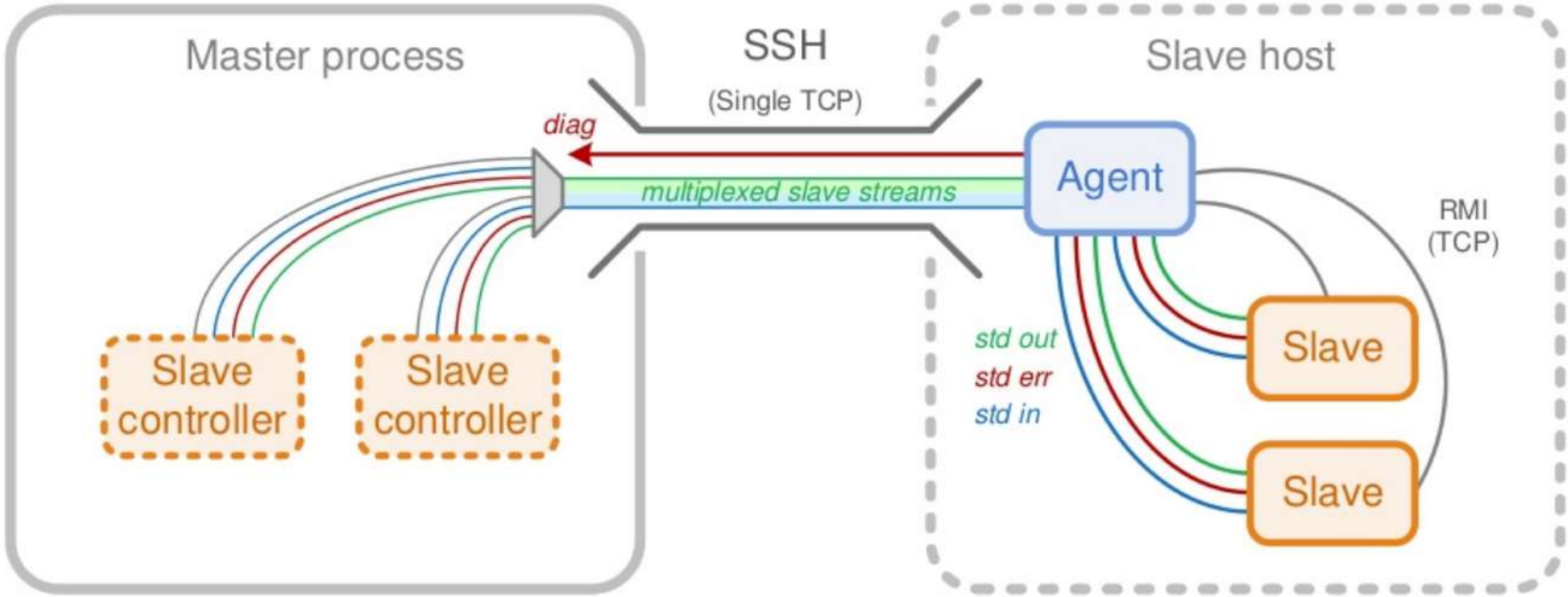
Groovy in legacy environments. SSH

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 debuger)

Death clock is ticking on slave though

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


?
Q
&
A
!

JavaCro'18

Thank you!

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