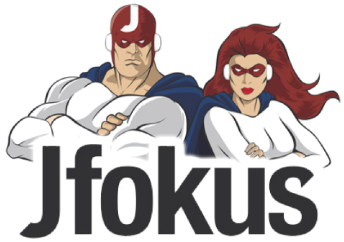
ninja tricks for

system scripting



by Andrey Adamovich @codingandrey

About me

Andrey Adamovich

- Bio: Developer, coach, speaker, author
- Company: Aestas/IT (http://aestasit.com)
- E-mail: andrey@aestasit.com
- Linkedin: http://www.linkedin.com/in/andreyadamovich
- Lanyrd: http://lanyrd.com/profile/andrey-adamovich
- GitHub: https://github.com/aadamovich
- SO: http://stackoverflow.com/users/162792/andrey-adamovich
- Twitter: @codingandrey, @aestasit

Exposed to operations

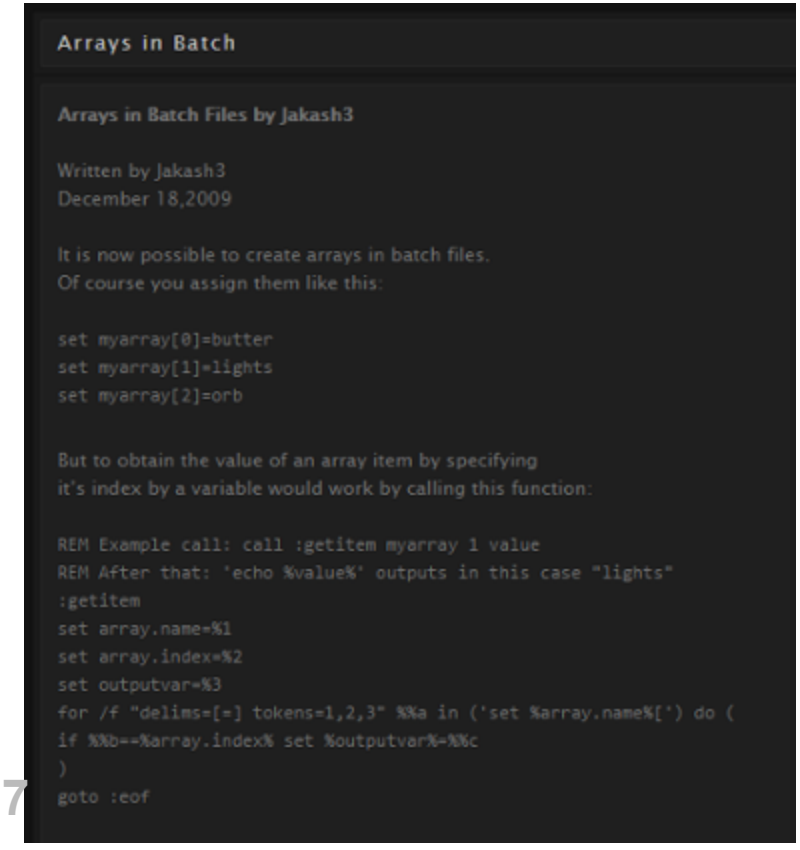
- Maintenance scripts (backup, cleaning, monitoring)
- Scheduled/CRON jobs (transfers, conversions, notifications)
- Automation tasks (deployment, packaging, testing)
- Web applications (status pages, back doors, dashboards)

Why Groovy?

Native system scripting

- Bash, Dash, Zsh, Fish, Csh etc.
- Bat, Cmd and, eventually, PowerShell (PoSH)

Arrays in BAT?



Arrays in Batch

Arrays in Batch Files by Jakash3

Written by Jakash3
December 18,2009

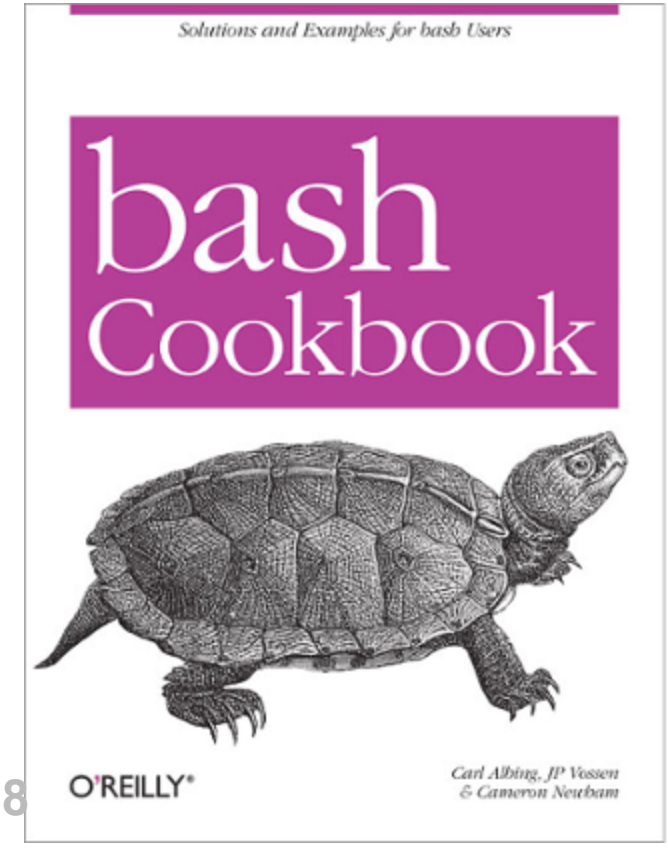
It is now possible to create arrays in batch files.
Of course you assign them like this:

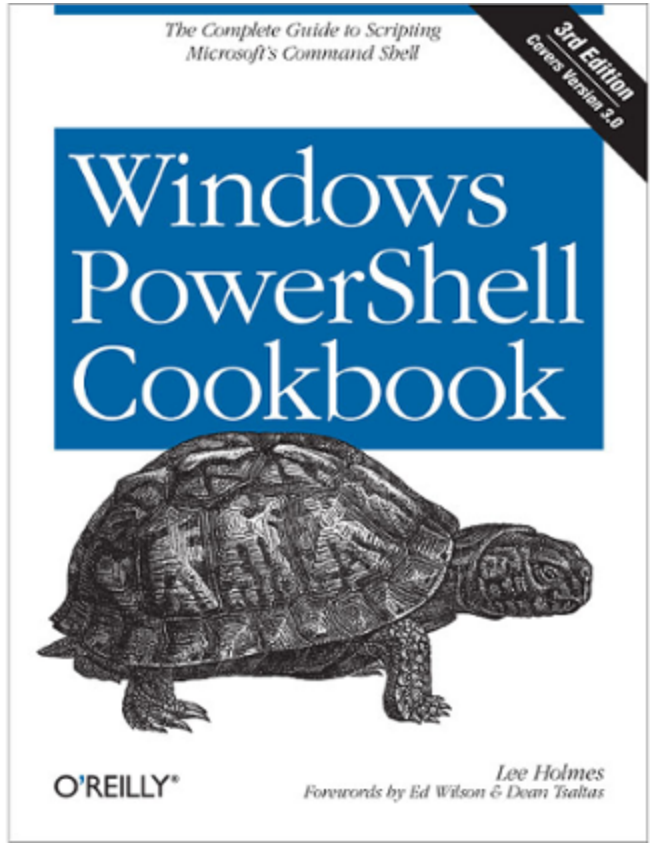
```
set myarray[0]=butter
set myarray[1]=lights
set myarray[2]=orb
```

But to obtain the value of an array item by specifying
it's index by a variable would work by calling this function:

```
REM Example call: call :getitem myarray 1 value
REM After that: 'echo %value%' outputs in this case "lights"
:getitem
set array.name=%1
set array.index=%2
set outputvar=%3
for /f "delims=[=] tokens=1,2,3" %%a in ('set %array.name%[') do (
if %%b==%array.index% set %outputvar%=%%c
)
goto :eof
```

BASH vs. PoSH


Solutions and Examples for bash Users
bash
Cookbook
O'REILLY
Carl Albing, JP Vossen
& Cameron Newham


The Complete Guide to Scripting
Microsoft's Command Shell
3rd Edition
Covers Version 3.0
Windows
PowerShell
Cookbook
O'REILLY
Lee Holmes
Forewords by Ed Wilson & Dean Tsaltas

Scripting languages

- Perl
- Python
- Ruby
- Lua
- Javascript
- ...

Multi-platform support

Does it work on Windows?

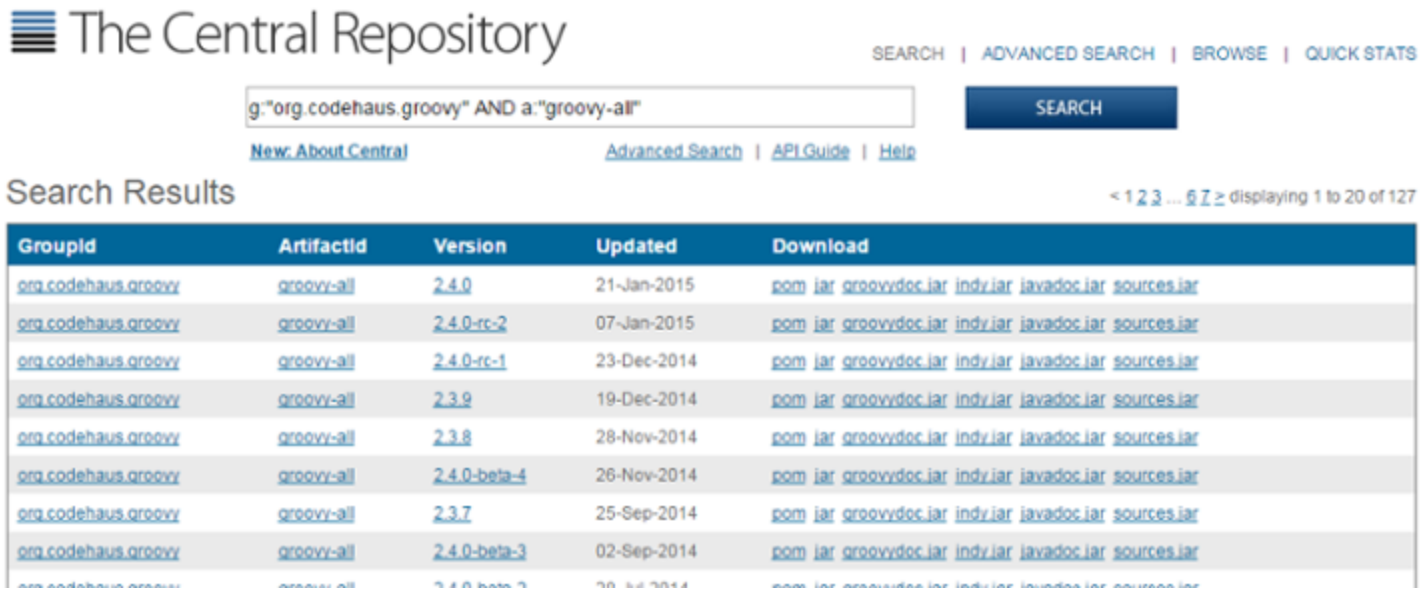
Operational simplicity

Maintainable code

Stable ecosystem

Rich extension library

Groovy is just a JAR



Reuse JVM knowledge

GVM

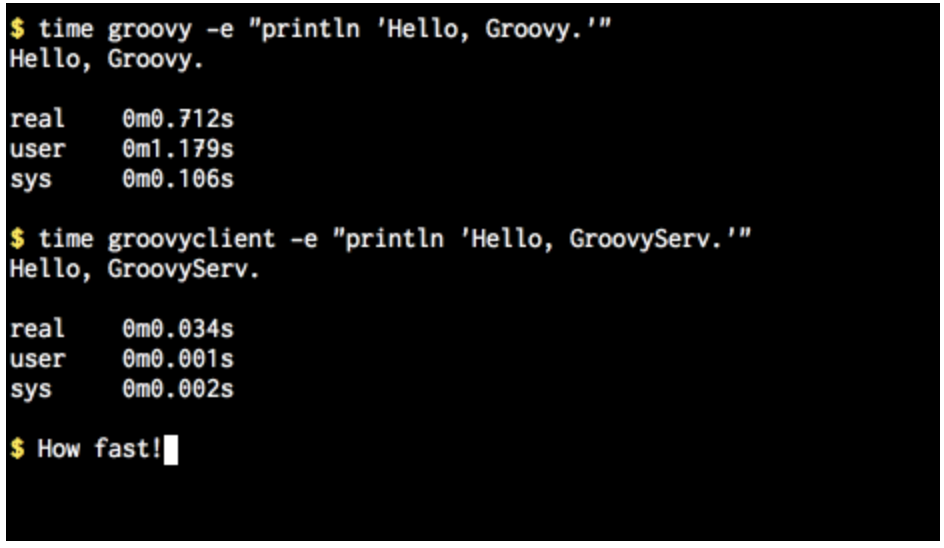
```
01. curl -s get.gvmtool.net | bash
02. gvm install groovy 2.3.6
03. gvm default groovy 2.3.6
04. gvm list groovy
```

PoSH GVM

```
01. > posh-gvm.psm1 cannot be loaded
02. > because the execution of scripts
03. > is disabled on this system.
```

Speed up!

GroovyServ


$ time groovy -e "println 'Hello, Groovy.'"
Hello, Groovy.
real 0m0.712s
user 0m1.179s
sys 0m0.106s
$ time groovyclient -e "println 'Hello, GroovyServ.'"
Hello, GroovyServ.
real 0m0.034s
user 0m0.001s
sys 0m0.002s
$ How fast!

Files, directories, and other animals

Reading

```
01. f = new File('test.txt')
02. String textContent = f.text
03. byte[] binaryContent = file.bytes
```

Writing I

```
01. f << 'hello, this is a test file'
02. f.text = new URL('http://server/my_data.json').text
03. file.bytes = [ 65, 66, 67, 68 ] as byte[]
```

Writing II

```
01. file.text = '''What's in a name? That which we call a rose
02. By any other name would smell as sweet.'''
03.
04. file << 'What\'s in a name? That which we call a rose\n'
05. file << 'By any other name would smell as sweet.'
```

Writing III

```
01. file.withWriter { Writer writer ->
02.   writer << 'What\'s in a name? That which we call a rose\n'
03.   writer << 'By any other name would smell as sweet.'
04. }
05. file.withOutputStream { OutputStream stream ->
07.   stream << 'What\'s in a name? That which we call a rose\n'
08.   stream << 'By any other name would smell as sweet.'
09. }
```

Line-by-line

```
01. def lines = f.readLines()
02. lines.each { String line ->
03.   println line
04. }
```

eachLine and eachByte

```
01. f.eachLine { String line ->
02.   println line
03. }
04. file.eachByte { int b ->
05.   ...
06. }
```

Filtering I

```
01. outputFile.withPrintWriter { writer ->
02.   inputFile.eachLine { line ->
03.     if (!line.startsWith('#')) {
04.       writer.println(line)
05.     }
06.   }
07. }
```

Filtering II

```
01. inputFile.filterLine(outputFile.newWriter()) { line ->
02.   !line.startsWith('#')
03. }
```

Filtering III

```
01. outputFile.withWriter { Writer writer ->
02.   inputFile.withReader { Reader reader ->
03.     reader.transformLine(writer) { String line ->
04.       line.replaceAll('\t', '  ')
05.     }
06.   }
07. }
```

Filtering IV

```
01. inputFile.withReader { reader ->
02.   reader.transformLine(outputFile.newWriter()) { line ->
03.     line.replaceAll('\t', '  ')
04.   }
05. }
```

Filtering V

```
01. outputFile.text = inputFile.text.replaceAll('\t', '  ')
```

Walking the path

eachFile and eachDir

```
01. f.eachFile { File file ->
02.   ...
03. }
04. f.eachDir { File dir ->
05.   ...
06. }
```

Recursive walking I

```
01. currentDir.eachFileRecurse { File file ->
02.   println file.name
03. }
04. currentDir.eachDirRecurse { File dir ->
05.   println dir.name
06. }
```

Recursive walking II

```
01. currentDir.traverse { File file ->
02.   if (file.name.endsWith('.groovy')) {
03.     println file.name
04.   }
05. }
```

Recursive walking III

```
01. currentDir.traverse(nameFilter: ~/.*\.groovy/) { File file ->
02.   println file.name
03. }
```

Recursive walking IV

```
01. import static groovy.io.FileType.*
02. ...
03. currentDir.traverse(
04.   type: FILES,
05.   nameFilter: ~/.*\.groovy/,
06.   excludeNameFilter: ~/^C.*$/) { File file ->
07.     println file.name
08. }
```

Extending File I

```
01. File.metaClass.safeDelete = {
02.   if (exists()) {
03.     if (isDirectory()) {
04.       if (!deleteDir()) {
05.         def msg = "Unable to delete a directory: ${name}"
06.         throw new IOException(msg)
07.       }
08.     ...
```

Extending File II

```
01.     ...
02.     } else {
03.       if (!delete()) {
04.         def msg = "Unable to delete a file: ${name}"
05.         throw new IOException(msg)
06.       }
07.     }
08.   }
09. }
```

Extending File III

```
01. File f = new File('test.txt')
02. f.safeDelete()
03. File d = new File('test_dir')
04. d.safeDelete()
```

External commands

```
01. def exitValue = "ls -l".execute().exitValue()
02. if (!exitValue) {
03.   println "Command failed with exit code: ${exitValue}"
04. }
```

Defensive scripting

Do you like NPEs?

Even simple things can cause NPE

Java:

```
01. System.out.println(user.
02.                        getAddress().
03.                        getCity())
```

Groovy:

```
01. println user.
02.           address.
03.             city
```

Safe navigation operator (?.)

```
01. println user?.
02.             address?.
03.               city
```

Elvis operator (?:)

```
01. println user?.
02.           address?.
03.             city ?: "No idea"
```

Closures for DSLs

Imaginary DSL

```
01. transaction {
02.   request {
03.     id     = '2L'
04.     name   = 'PUT'
05.     amount = 25
06.   }
07.   request {
08.     ...
09.   }
10. }
```

DSL: transaction() method

```
01. def transaction(Closure cl) {
02.   def processor = new RequestProcessor()
03.   try {
04.     processor.init()
05.     cl.delegate = processor
06.     cl()
07.     sendRequests()
08.   } finally {
09.     processor.cleanup()
10. }}
```

DSL: request() method

```
01. class RequestProcessor {
02.   def requests = []
03.   def request(Closure cl) {
04.     def request = new Request()
05.     requests << request
06.     cl.delegate = request
07.     cl()
08.   }
09. ...
```

DSL: Request structure

```
01. class Request {
02.   String id
03.   String name
04.   String amount
05. }
```

Data manipulation

XML

```
01. import groovy.util.XmlSlurper
02. def xmlSource = new File('shakespeare.xml')
03. def bibliography = new XmlSlurper().parse(xmlSource)
04. println bibliography.author
05. bibliography.play
06.                 .findAll { it.year.toInteger() > 1592 }
07.                 .each { println it.title }
```

JSON

```
01. import groovy.json.JsonSlurper
02. def reader = new FileReader('ui.json')
03. def ui = new JsonSlurper().parse(reader)
04. ui.items.each { println it.type }
05. println ui.items[0]
06.           .axes
07.           .find {
08.             it.fields.contains('y')
09.           }.title
```

Grab your stuff!

@Grab: import

```
01. @Grab('org.apache.httpcomponents:httpclient:4.2.1')
02. import org.apache.http.impl.client.DefaultHttpClient
03. import org.apache.http.client.methods.HttpGet
```

@Grab: variable

```
01. @Grab('org.apache.httpcomponents:httpclient:4.2.1')
02. def httpClient =
03.   new org.apache.http.impl.client.DefaultHttpClient()
```

@Grab: method

```
01. @Grab('org.apache.httpcomponents:httpclient:4.2.1')
02. def getHttpClient() {
03.   new org.apache.http.impl.client.DefaultHttpClient()
04. }
```

@Grab: class

```
01. @Grab('org.apache.httpcomponents:httpclient:4.2.1')
02. class Searcher {
03.   def httpClient
04.   Searcher() {
05.     httpClient =
06.       new org.apache.http.impl.client.DefaultHttpClient()
07.   }
08. }
```

@Grab: multiple

```
01. @Grapes([
02.     @Grab('org.apache.httpcomponents:httpclient:4.2.1'),
03.     @Grab('org.ccil.cowan.tagsoup:tagsoup:1.2')])
04. class Searcher { ... }
```

@Grab: repository

```
01. @GrabResolver(name='codehaus',
02.               root='http://repository.codehaus.org/')
03. class Searcher { ... }
```

@Grab: exclude

```
01. @GrabExclude(group='commons-codec',
02.              module='commons-codec')
03. class Searcher { ... }
```

HTTP

HTTPBuilder: import

```
01. @Grab(
02.   group='org.codehaus.groovy.modules.http-builder',
03.   module='http-builder',
04.   version='0.6'
05. )
06. import groovyx.net.http.*
```

HTTPBuilder: instantiation

```
01. def baseUrl = 'http://api.duckduckgo.com'
02. def queryString = 'q=groovy&format=json&pretty=1'
03. def http = new HTTPBuilder(baseUrl)
```

HTTPBuilder: request

```
01. http.request(Method.POST) {
02.   send ContentType.URLENC, queryString
03.   response.success = { response, reader ->
04.     println response.statusLine
05.     println reader.text
06.   }
07.   response.failure = { response ->
08.     println response.statusLine
09.   }
10. }
```

SQL

SQL: settings

```
01. import groovy.sql.Sql
02. def dbSettings = [
04.   url: 'jdbc:hsqldb:hsql://localhost/cookingdb',
05.   driver: 'org.hsqldb.jdbcDriver',
06.   user: 'sa',
07.   password: ''
08. ]
```

SQL: query

```
01. def sql = Sql.newInstance(dbSettings)
02. sql.eachRow('SELECT * FROM COOKBOOK') { cookbook ->
04.   printf '%-20s%s\n',
05.          cookbook.id,
06.          cookbook[1]
07. }
```

Groovy as a server (GaaS?)

```
01. > groovy -l 4444 -e "println new Date()" &
02. > telnet localhost 4444
03. hey groovy give me the date
04. Wed Feb 04 10:03:23 EET 2015
```

Ratpack

Ratpack: import

```
01. @Grab("io.ratpack:ratpack-groovy:0.9.4")
02. import static ratpack.groovy.Groovy.ratpack
```

Ratback: script body

```
01. ratpack {
02.   handlers {
03.     get {
04.       response.send "This is the app root (try: /date)"
05.     }
06.     get("date") {
07.       response.send new Date().toString()
08.     }
09.   }
10. }
```

Sshoogr

Sshoogr: import

```
01. @Grab(
02.   group='com.aestasit.infrastructure.sshoogr',
03.   module='sshoogr',
04.   version='0.9.16')
05. import static com.aestasit.ssh.DefaultSsh.*
```

Sshoogr: defaults

```
01. defaultUser    = 'root'
02. defaultKeyFile = new File('secret.pem')
03. execOptions {
04.   verbose      = true
05.   showCommand  = true
06. }
```

Sshoogr: connection

```
01. remoteSession {
02.   url = 'user2:654321@localhost:2222'
03.   exec 'rm -rf /tmp/*'
04.   exec 'touch /var/lock/my.pid'
05.   remoteFile('/var/my.conf').text = "enabled=true"
06. }
```

Sshoogr: multi-line content

```
01. remoteFile('/etc/yum.repos.d/puppet.repo').text = '''
02.   [puppet]
03.   name=Puppet Labs Packages
04.   baseurl=http://yum.puppetlabs.com/el/
05.   enabled=0
06.   gpgcheck=0
07. '''
```

Sshoogr: file copying

```
01. remoteSession {
02.   scp {
03.     from { localDir "$buildDir/application" }
04.     into { remoteDir '/var/bea/domain/application' }
05.   }
06. }
```

Sshoogr: command result

```
01. def result = exec(command: '/usr/bin/mycmd',
02.   failOnError: false, showOutput: false)
03. if (result.exitStatus == 1) {
04.   result.output.eachLine { line ->
05.     if (line.contains('WARNING')) {
06.       throw new RuntimeException("First warning: ${line}")
07.     }
08.   }
09. }
```

Sshoogr: shortcuts

```
01. if (ok('/usr/bin/mycmd')) {
02.   ...
03. }
04. if (fail('/usr/bin/othercmd')) {
05.   ...
06. }
```

Sshoogr: tunnels

```
01. tunnel('1.2.3.4', 8080) { int localPort ->
02.   def url = "http://localhost:${localPort}/flushCache"
03.   def result = new URL(url).text
04.   if (result == 'OK') {
05.     println "Cache is flushed!"
06.   } else {
07.     throw new RuntimeException(result)
08.   }
09. }
```

Sshoogr: prefix/suffix

```
01. prefix('sudo ') {
02.   exec 'rm -rf /var/log/abc.log'
03.   exec 'service abc restart'
04. }
05. suffix(' >> output.log') {
06.   exec 'yum -y install nginx'
07.   exec 'yum -y install mc'
08.   exec 'yum -y install links'
09. }
```

Summary

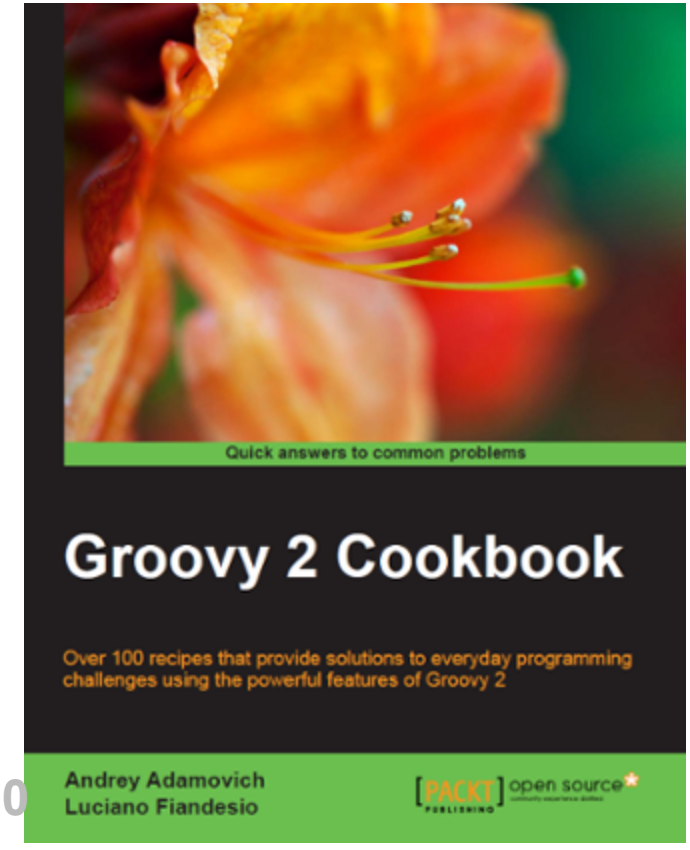
Take-aways

- Groovy scripts are portable
- Groovy scripts are concise
- Well, the name is groovy as well!

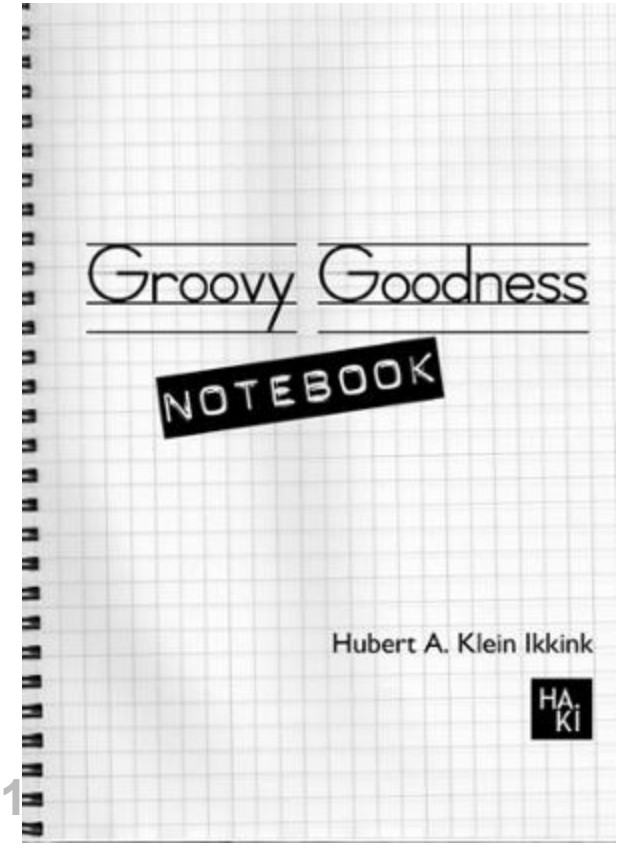
Reading material

http://groovy-lang.org/

Groovy 2 Cookbook



Groovy Goodness



Questions?

Thank you!

Happy scripting!