HACKING JENKINS

Miro Cupak (DNAstack), Oliver Gondza (Red Hat)

2015-02-07

TODAY'S TOPICS

1. REST API
2. CLI
3. Scripting
4. Groovy jobs

ENVIRONMENT SETUP

```
wget http://www.fi.muni.cz/~xcupak/download/hacking-jenkins-offline.zip # or copy from USB key
unzip hacking-jenkins-offline.zip
cd hacking-jenkins-offline
./start.sh # Jenkins should be available a http://localhost:8080/
```

Section 1

REST API

JENKINS EXPORTED API

- Jenkins data exposed in machine readable format
- available on most Jenkins URLs
- support for json, xml and python

JENKINS REST API

- numerous endpoints to interact with Jenkins
- configuration exposed via config.xml endpoint

Task 1.1

TRIGGER ACTIONS IN JENKINS

Task 1.1

GOAL

Use REST API to trigger build using `curl`.

HINTS

- endpoint is $JENKINS_URL/job/$JOB_NAME/build

Task 1.1

SOLUTION

```
curl -v -X POST http://localhost:8080/job/example_job/build
```

Task 1.2

MANIPULATE CONFIGURATION USING REST API

Task 1.2

GOAL

Update number of executors on given node via REST API.

HINTS

- fetch config.xml, modify by hand and send it back

Task 1.2

SOLUTION

```
curl http://localhost:8080/computer/example_slave/config.xml > example_slave.xml
vi example_slave.xml
curl -X POST http://localhost:8080/computer/example_slave/config.xml -d @example_slave.xml
```

Section 2

COMMAND-LINE INTERFACE

JENKINS CLI

- run predefined commands from console
- Manage Jenkins > Jenkins CLI

```
java -jar jenkins-cli.jar -s $JENKINS_URL help
```

Task 2.1

USING CLI COMMANDS

Task 2.1

BACKGROUND

Jenkins master does not respond through UI. It needs to be shut down before whole machine can be restarted.

Task 2.1

GOAL

Use Jenkins CLI to safely shutdown master.

HINTS

- have a look at `safe-shutdown` command

Task 2.1

SOLUTION

```
java -jar jenkins-cli.jar -s http://localhost:8080 safe-shutdown
```

Task 2.2

USING CLI TO MANIPULATE CONFIGURATION

Task 2.2

GOAL

Create copy of existing node using nothing but Jenkins CLI.

HINTS

- Fetch config.xml and use it to create new node.
- Commands `get-node` and `create-node`.

Task 2.2

SOLUTION

```
java -jar jenkins-cli.jar -s http://localhost:8080 get-node example_slave > example_slave.xml
java -jar jenkins-cli.jar -s http://localhost:8080 create-node new_slave < example_slave.xml
```

Section 3

SCRIPTING

JENKINS SCRIPT CONSOLE

- run arbitrary scripts on master and slave nodes
- useful for users, developers, **administrators**
- UI: Manage Jenkins > Script Console
- CLI:

```
java -jar jenkins-cli.jar -s http://jenkins/ groovysh
```

- HTTP POST:

```
curl -d "script=myscript.groovy" http://jenkins/script
```

Task 3.1

EXECUTING SCRIPTS ON SLAVES

Task 3.1

BACKGROUND

Our build is failing upon initialization. We suspect the machine running the slave does not have enough free memory.

Task 3.1

GOAL

Use example_slave's script console to find out how much free RAM it has.

HINTS

- slave's script console is located at http://localhost:8080/computer/example_slave/script
- read the script console help
- `free` shell command should do the trick - can you run it from Groovy?

Task 3.1

SOLUTION

```
println "free -m".execute().text
```

Task 3.2

EXECUTING SCRIPTS ON MASTER

Task 3.2

BACKGROUND

Labels are basically the only way to distinguish slaves in Jenkins. A good idea is to use feature labels, i.e. tag slaves according to what they provide (e.g. "rhel7 32b mem16g").

Task 3.2

GOAL

Use the script console to mark all Linux slaves as development machines, i.e. add "dev" label to all slaves with "linux" label.

HINTS

- a good place to start is the root of the Jenkins tree - `Jenkins.instance`
- look at Javadoc of `jenkins.model.Jenkins` to find out how to obtain the list of slaves
- slave labels are stored as a single space-delimited string
- don't forget to persist your changes

Task 3.2

SOLUTION

```
Jenkins.instance.getLabel("linux").nodes.each {
  it.labelString += " dev"
}

j.save()
```

Section 4

GROOVY JOBS

GROOVY JOBS

- freestyle projects with a Groovy build step
- provided by Groovy plugin
- 2 types of build steps:
  - groovy scripts
    - run in the slave's JVM
    - like running `groovy` command with a script
  - system groovy scripts
    - run inside master's JVM
    - have access to all the internal objects of Jenkins and can alter its state
- similar to script console

Task 4

CREATING SYSTEM JOBS

BACKGROUND

We're running a small Jenkins instance with a small number of slaves. Occasionally, slaves crash or need to undergo maintenance. We want to make sure that we have at least one online slave at any given time.

Task 4

GOAL

Create a job monitoring the number of active slaves. The job should run on master every hour and send an email to you whenever there are no online slaves.

HINTS

- see Javadoc of `hudson.model.Computer`
- use Mailer plugin to send the notifications
- unlike the script console, jobs don't do any automagic imports - this should do the trick:

```
import jenkins.model.*
import hudson.model.*
```

Task 4

SOLUTION

```
import jenkins.model.*
import hudson.model.*

def online = 0
for(computer in Jenkins.instance.computers){
  if(computer.isOnline()){
    online++
  }
}

return !(online<1)
```

THANK YOU.

Feedback: http://devconf.cz/f/152.

Slides: http://mcupak.github.io/hacking-jenkins-workshop/.