

DevOps World



Jenkins World



Jean-Marc Meessen

CloudBees
DevOps Consultant

Featured Speaker

Dec
5

Automating Jenkins (re)installation: some thoughts, tips, and tricks



Lisbon | December 3 - 5, 2019



Automating Jenkins (re)installation:

some thoughts, tips, and tricks

Presentation available at: <https://jmMeessen.github.io/slides/jw-eu-2019>

Hello !!

- Jean-Marc MEESEN:



Who are you?

What is configuration Management?





Pet versus Cattle



Automation, automation, automation!

- Frees precious time
- Repeatable
- Best emergency/repair tool
- Best way to avoid any malicious modification

In Source Control

- Visibility
- Peer review
- History
- Versioned \Rightarrow Revertable

Why should CI/CD systems be
handled as a Pet ?

Automation objectives

- Provision new CI/CD cluster (or major components)
 - efficiently
 - repeatably
 - consistant
- Update the system
 - ex: change a setting, add a plugin

Automation objectives

- Peer-review mechanism for configuration changes
 - Keeps the audit/compliance team happy
- Easily manage very large CI/CD cluster
- Properly document the system
- support CI/CD power users
 - behind the scene warranty for creativity

Configuration Management philosophies

Golden Image

- in the early days
 - a lot of work to maintain
 - messy
 - "one size fits nobody"

Configuration Scripting

- Scripts solved a lot of these problems
 - added
 - readability
 - versioning
- At first ad hoc (bash) scripting
- then Chef, Puppet, Ansible, etc.

Golden Image revisited

- Docker/Containers
 - Golden Image new momentum
 - very short start time
 - image definition description files (dockerfiles)
 - particularly adapted to the Cloud scheduler (ex K8S)

But no silver bullet

- reality lies between
 - generalization (general purpose images)
 - need for fine grained customizations to adapt to the local constrains

Jenkins configuration vectors



Direct file System manipulation

- classical way to configure a system
- copying/updating files on the file system (JENKINS_HOME)
- Typical Ansible modules.
 - copy
 - template
 - lineinfile
 - xml

File system vector: Pro

- easy/natural for tools like Ansible

File system vector: Con

- lot of reverse engineering required
- stability of these undocumented configuration is not guaranteed.
 - particularly plugins configuration

Command Line Interfaces

- two types
 - REST API
 - Jenkins CLI

REST API

- using HTTP requests to GET, PUT, POST and DELETE data.

```
curl -X POST "<jenkinsURL>/testProject/build" --user jmm:<password|token>
```

REST API - CSRF protection

- be aware of CSRF protection (should be on, isn't it?)
 - session hijacking
 - requires a token or "crumb" when using password
 - not required when using an API Token

REST API

- To learn more:
 - <https://wiki.jenkins.io/display/JENKINS/Remote+access+API>
 - <https://wiki.jenkins.io/display/JENKINS/Authenticating+scripted+clients>

Jenkins CLI

- Traditional way, via the `jenkins-cli.jar`
- To list the very functions list (dependant of installed plugin):
 - view it in "Manage Jenkins → Jenkins CLI"
 - or simply use "help" CLI command.

Jenkins CLI - Classic

```
java -jar jenkins-cli.jar -http -s $JENKINS_URL -auth $USERNAME:$API_token command ...
```

Much better:

```
java -jar jenkins-cli.jar -http -s $JENKINS_URL -auth @FILE command ...
```

Jenkins CLI - SSH

- A simple SSH can also be used.
- Requires to enable the build-in SSH server and assign a port
 - watch your firewalls and reverse proxy

```
ssh -l jmm -i ~/.ssh/id_rsa -p 10200 my-jenkins-server help
```

Jenkins CLI - More details

- <https://jenkins.io/doc/book/managing/cli/>

Summary

- Rich set of API
- Easy to use in Ansible for example
- Initial user and credential is a tough problem to solve
 - SSH authentication can be automated
- CLI does a better job at controlling parameter
- CLI makes blocking calls
- CLI commands are better documented
- Parsing results is tricky

Recommendation

- Use CLI
- Use CLI with SSH if you can (networking)
- Consider executing commands from target host.

Groovy Scripts

- Richest way to configure Jenkins
 - Taps into Jenkins native language
- Need developer skills
- Documentation not easy to find
 - See this *Knowledge Base article* on how to access the javadocs
- Make them idempotent !

How to use Groovy Script

- via the script console
- at startup, as init-script
 - placed in `$JENKINS_HOME/init.groovy.d/`
 - executed in lexical order
- via the CLI

Groovy Scripts from the CLI

```
cat my_script.groovy | {{ CLI_command }} groovy =
```

Docker Container

- Can automate the configuration of some parts
 - ex: pre-loading plugins
- But does not solve all the problems
- a little out of the scope of this presentation

Jenkins Configuration as Code

- First developed and tested in OSS realm
 - Implementation on CloudBees product is ongoing
-
- Declarative method, yaml based
 - Loaded on reboot or with a CLI command

JCasC Example


```
jenkins:
  securityRealm:
    ldap:
      configurations:
        - inhibitInferRootDN: false
          managerDN: "uid=idm,ou=Administrators,dc=example,dc=com"
          managerPasswordSecret: "{{ ldap_admin_passw }}"
          rootDN: "dc=example,dc=com"
          server: "ldap://{{ full_agent_docker_dns_name }}:389"
      disableMailAddressResolver: false
      disableRolePrefixing: true
      groupIdStrategy: "caseInsensitive"
      userIdStrategy: "caseInsensitive"
```

Current Status

- In technical preview
 - Masters configuration work
 - CloudBees functionality in the works
 - Waiting for RBAC support
- Centralized CasC management from CJOC

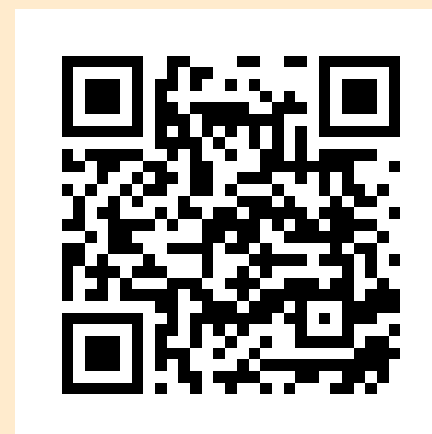
Thank You !




 @jm_meessen

 jmMeessen

Slides: <https://jmMeessen.github.io/slides/jw-eu-2019>



Source on : <https://github.com/jmMeessen/slides/tree/jw-eu-2019>