DevOps World

Jenkins World

“Look ma, no hands”

Jenkins Configuration-as-Code

Who are we?

Name: Ewelina Wilkosz

Work: IT Consultant @ Praqma

Jenkins World 2018 Awards

Most Valuable Jenkins Contributor

@ewelinawilkosz @ewelinawilkosz ewe@praqma.com

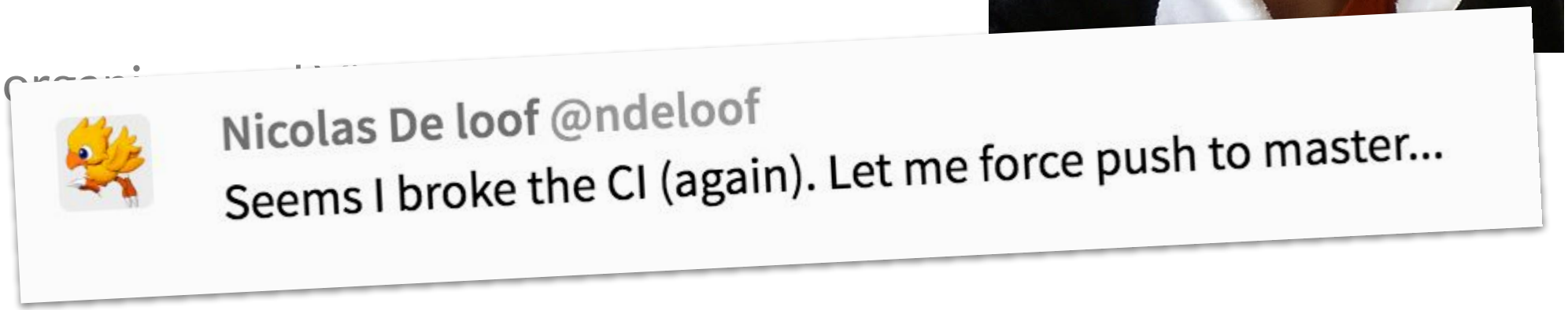
Who are we?

Name: Nicolas De Loof

Work: Hacker @ CloudBees

Jenkins contributor & Docker Captain

Conference organizer



@ndeloof

@ndeloof

ndeloof@cloudbees.com

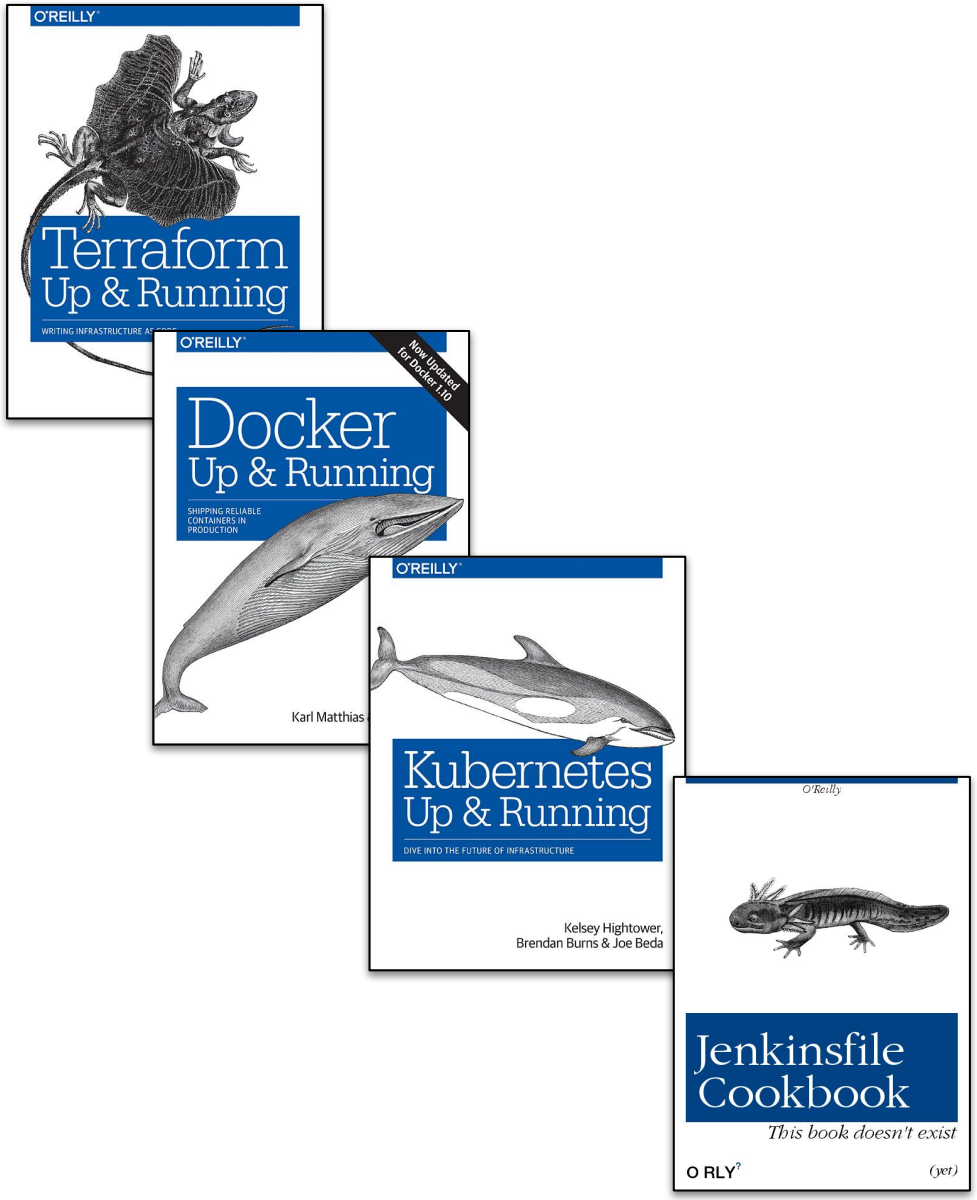
2018 is “ * as code”

Infrastructure *as Code*

Environment *as Code*

Architecture *as Code*

CI/CD *as Code*



Manage Jenkins *as Code*

- Jenkins infrastructure
- Jenkins job configuration
- Jenkins system configuration

Jenkins infrastructure

Jenkins infrastructure

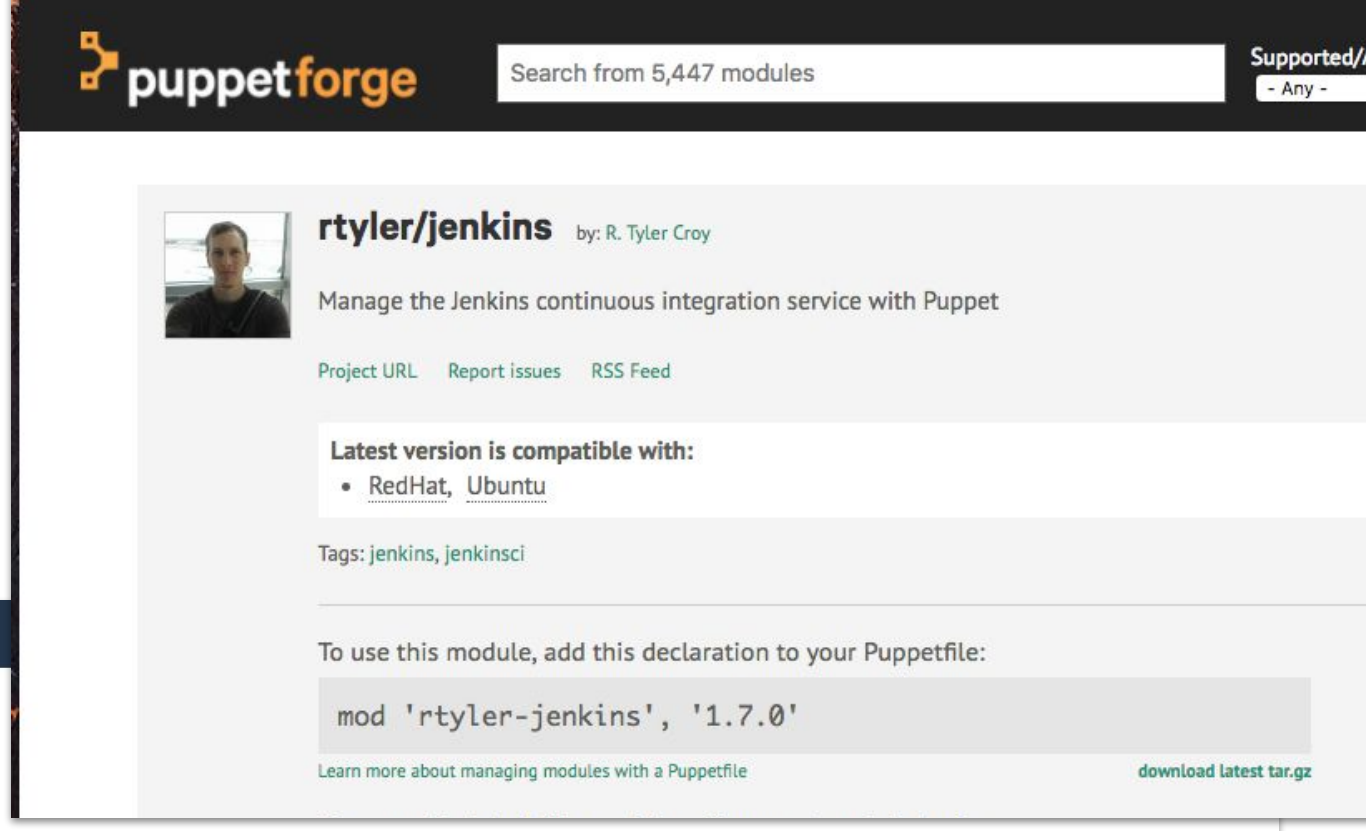
Using external tools

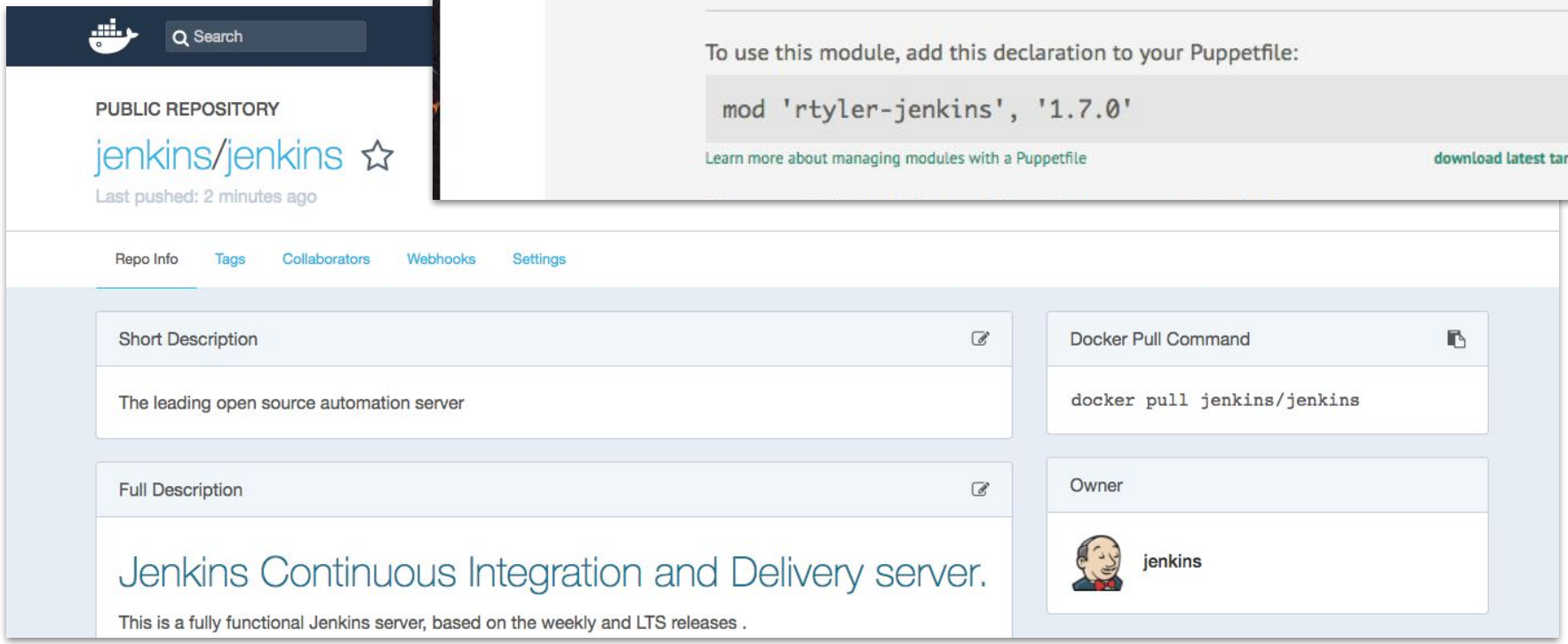
- Jenkins CLI
- REST API
- Python-jenkins
- Jenkins-client (Java, golang)
- ...

Jenkins infrastructure

Ansible, Chef, Puppet

Docker





jobs configuration

Jenkins job configuration

- JobDSL plugin (groovy)
- Job builder plugin (yaml)
- ...
- Jenkins Pipeline
  - Multibranch
  - Organizations folders

JobDSL

```
job('gr8 example') {
    scm {
        github 'sheehan/job-dsl-gradle-example'
    }
    triggers {
         scm 'H/5 * * * *'
    }
    steps {
        gradle 'clean test'
    }
    publishers {
        archiveJunit 'build/test-results/**/*.xml'
        extendedEmail 'mr.sheehan@gmail.com'
    }
}
```

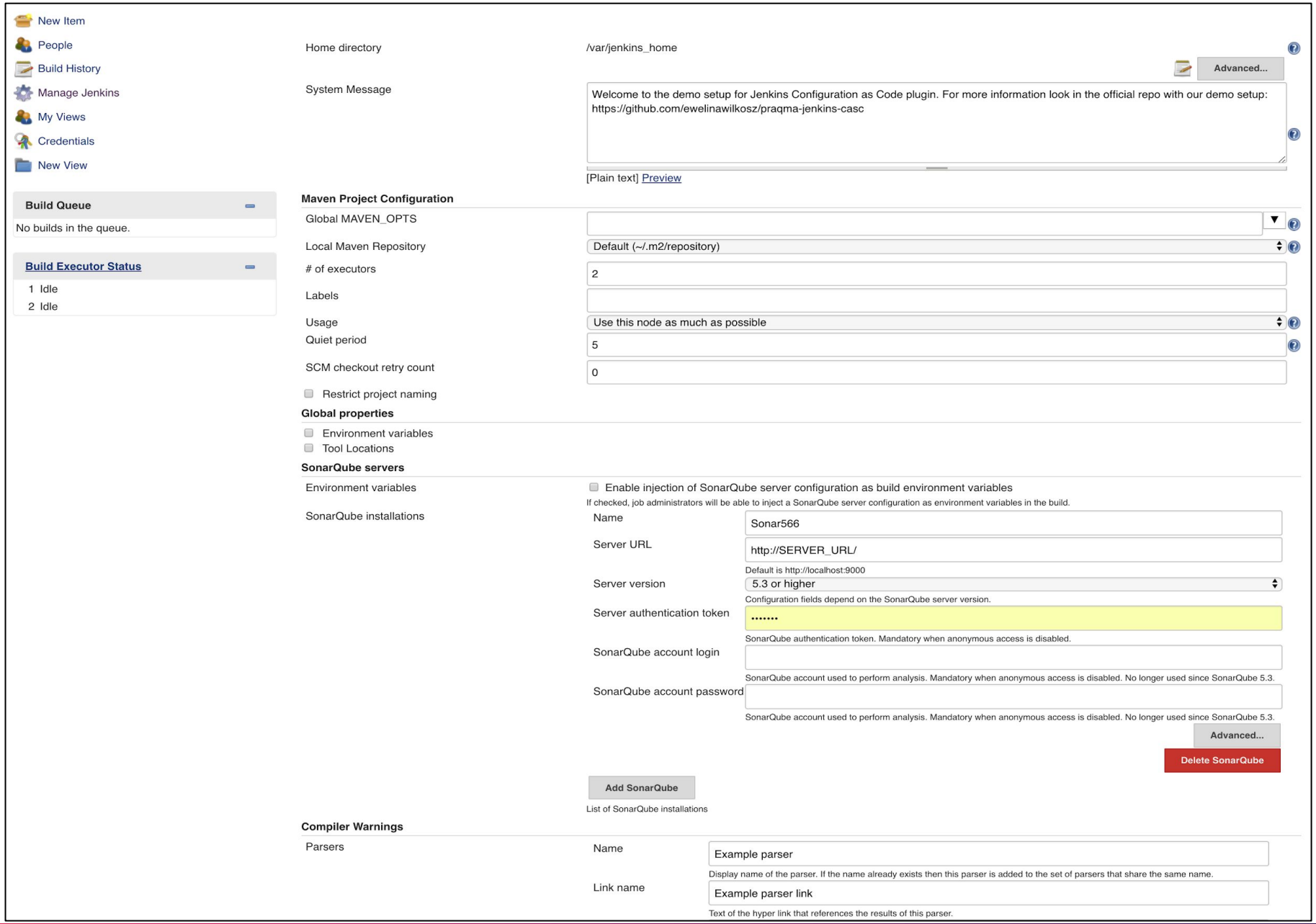
Jenkins master configuration

"Jenkins can be installed through native system packages, Docker, or run standalone by any machine with a Java Runtime Environment (JRE) installed..."

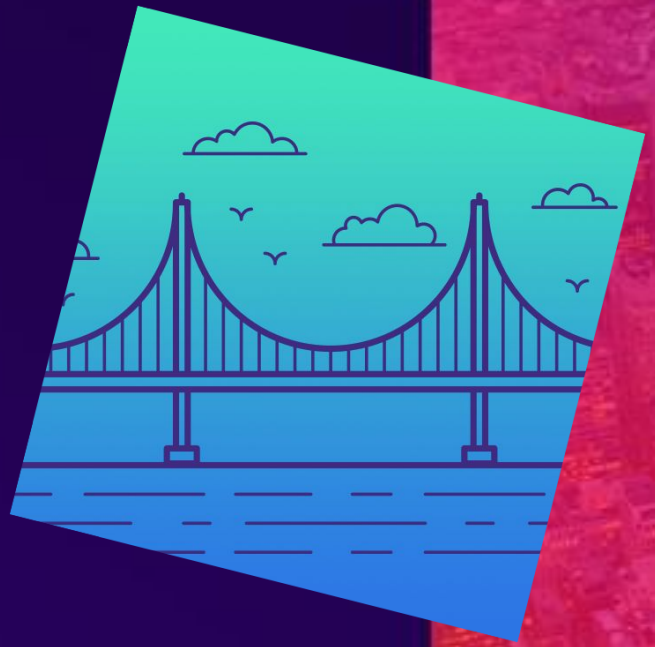
--- an enthusiast Jenkins user

“… but it has to be configured manually”

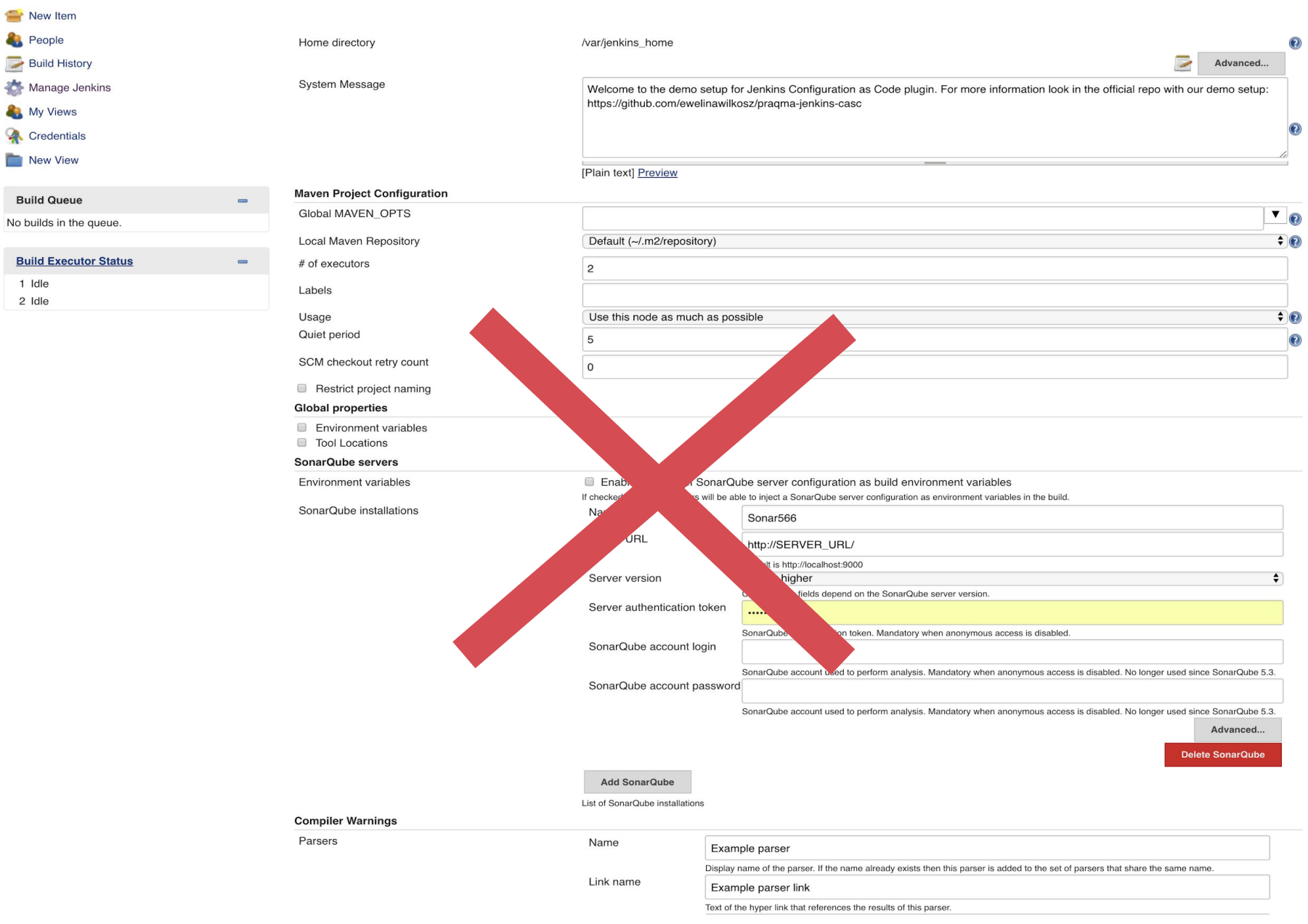
--- a not so enthusiast Jenkins user



looong scroll down



And we don’t (always) like that



So how do we solve it?

Jenkins system configuration

- init.groovy
- scriptler
- scm-sync-configuration 😱

87 lines (77 sloc) | 4.12 KB

Raw | Blame | History

```groovy
// Initializes the Development folder, which is fully configurable by the user

import groovy.io.FileType
import com.synopsys.arc.jenkins.plugins.ownership.OwnershipDescription
import hudson.plugins.filesystem_scm.FSSCM
import jenkins.model.Jenkins
import com.cloudbees.hudson.plugins.folder.Folder
import org.jenkinsci.plugins.ownership.model.folders.FolderOwnershipHelper
import org.jenkinsci.plugins.workflow.cps.CpsFlowDefinition
import org.jenkinsci.plugins.workflow.cps.CpsScmFlowDefinition
import org.jenkinsci.plugins.workflow.job.WorkflowJob
import org.jenkinsci.plugins.workflow.libs.FolderLibraries
import org.jenkinsci.plugins.workflow.libs.LibraryConfiguration
import org.jenkinsci.plugins.workflow.libs.SCMRetriever

println("=== Initialize the Development folder")
if (Jenkins.instance.getItem("Development") != null) {
    println("Development folder has been already initialized, skipping the step")
    return
}

// Admin owns the root Development folder
def folder = Jenkins.instance.createProject(Folder.class, "Development")
FolderOwnershipHelper.setOwnership(folder, new OwnershipDescription(true, "admin"))

// Users get their own sandboxes
def folder2 = folder.createProject(Folder.class, "User")
FolderOwnershipHelper.setOwnership(folder2, new OwnershipDescription(true, "user"))

// Create a library for local Jenkins Pipeline Library Development
// if the Env Var is set and the directory is mapped
println("==== Initializing local Pipeline Library development dir")
File file = new File("/var/jenkins_home/pipeline-library/vars")
if (!file.exists()) {
    println("/var/jenkins_home/pipeline-library is not mapped, skipping")
    return
} else {
    println("/var/jenkins_home/pipeline-library is mapped, initializing the directory")
}
```


We’re not alone

- JENKINS-31094 (system-config-dsl)
- XML templating (seen at JenkinsWorld 2017)
- Various Groovy bindings
- Praqma's "JenkinsAsCodeReference"
- CloudBees CTO Office's prototype

cloudbees PRAQMA to **join forces**

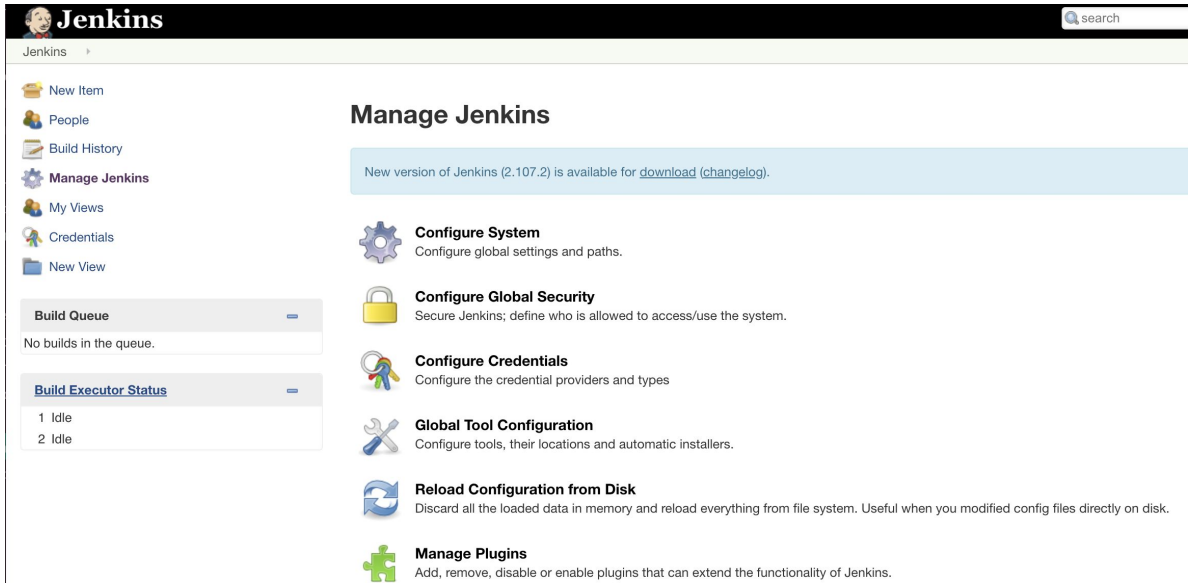
- Both had working prototypes last summer
- Praqma focusing on:
  - **real world** usage by customers
- CloudBees focusing on:
  - community **adoption**
  - out-of-the box support for our products

⇒ https://github.com/jenkinsci/configuration-as-code-plugin

Let’s make it as easy as possible

jenkins.yaml

```yaml
jenkins:
  systemMessage: "JCasC Demo"
  numExecutors: 1
  scmCheckoutRetryCount: 4
  mode: NORMAL
  securityRealm:
    local:
      allowsSignup: false
      users:
       - id: demoAdmin
         password: ${adminpw}
```



Main benefits

- Safety
- Traceability
- Speed
- Easy to use
- Easy to reuse

There are challenges

- *Manage configuration as **human-readable** config file(s)*
- *Self-describing model to **reflect Web UI***
- *Configure **all** jenkins initial setup (including plugins)*
- *Support most (*) plugins **without** extra development effort*
- *Generate **documentation** and **validation** tools (schema)*

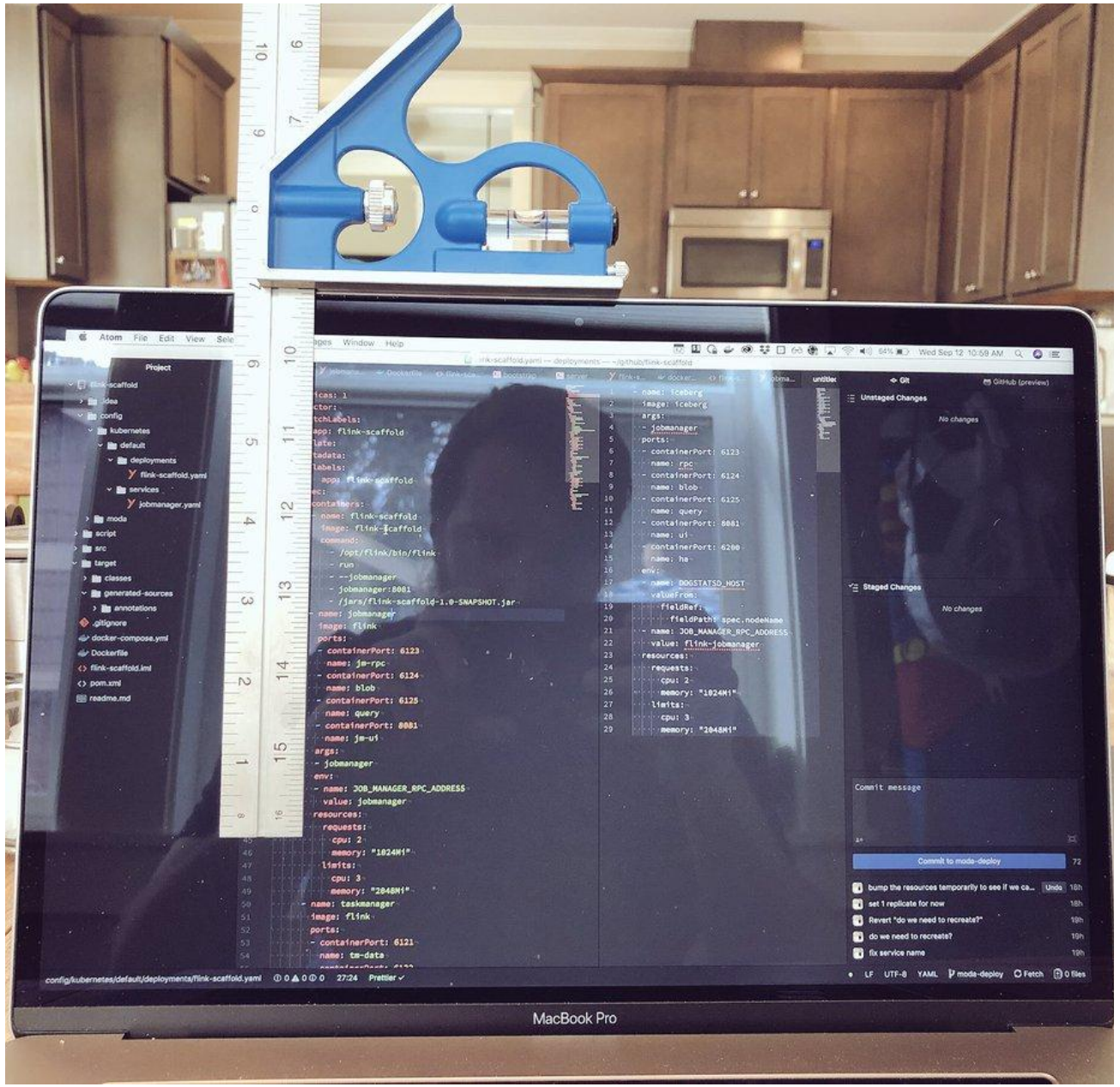
human-readable config file(s)

- Structured content
- Nothing language centric
  - No groovy / ruby / xx
- Readable and commentable

YAML ...

Indentation matters

photo credit :
Justin Palmer @Caged



30

Web UI as implicit documentation

Config element in web UI

==

Config element in YAML

“ No need to be a Jenkins expert to do it right”

-- Obi Wan Kenobi

Configure Jenkins in yaml

Obvious, isn't it ?

```yaml
jenkins:
   securityRealm:
       ldap:
           configurations:
               - server: ldap.acme.com
                 rootDN: dc=acme,dc=fr
                 managerPasswordSecret: ${LDAP_PASSWORD}
         cache:
             size: 100
            ttl: 10
         userIdStrategy: CaseSensitive
         groupIdStrategy: CaseSensitive

tool:
   git:
     installations:
         - name: git
         - path: /bin/git
```

Configure ALL jenkins initial setup

No hand on keyboard

No click on web UI

to deploy

a fully working Jenkins master

Support ALL plugins

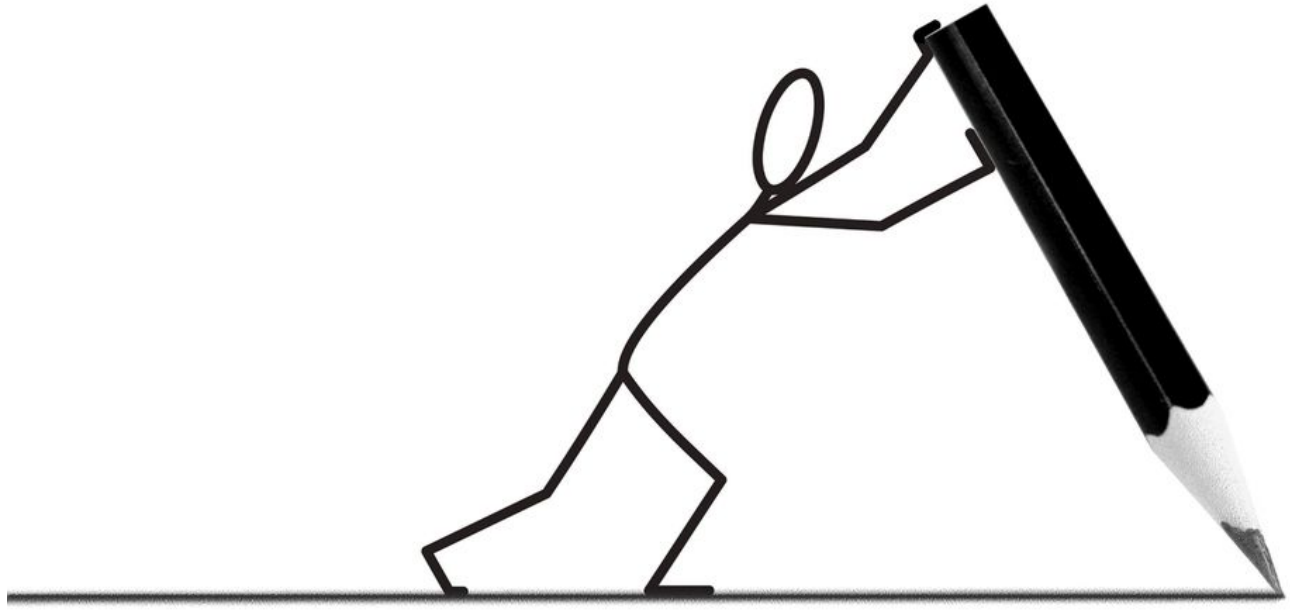
- No need to write glue code for every supported plugin
- Most(*) plugins supported out of the box
- Others can bundle adapter code

!! we require configuration-as-code-support plugin to be installed, for now !!

(*) could require some minor changes

Generate documentation and validation tools

- Can validate without running a test master
- IDE support



Here comes JCasC

Jenkins Configuration-as-Code

Where to find it?!

https://github.com/jenkinsci/configuration-as-code-plugin

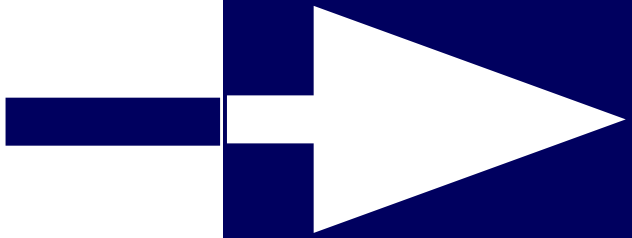
Implementation details and guide for plugin developers available in plugin's github repository

DEMO 😅

How it works

Live Jenkins instance

Core + plugins



Data model

- Yaml parser
- Doc generator
- Schema validator

40

Introspection

Jenkins-core 2.xx + plugins [git:3.7.0, ...]

- Jenkins root instance
- Descriptors (global configuration)
- + Special component with CasC support

=> hierarchical data model, trying to mimic Jenkins UI

Requirements

Target components need to follow some basic design rules

We rely on UI data binding mechanism (@DataBound)

Component to directly parse StaplerRequest / JsonObject can't be introspected

- Recommendations to plugin developers
  github.com/jenkinsci/configuration-as-code-plugin/blob/master/PLUGINS.md
- Pull requests on major plugins we want to support
  github.com/jenkinsci/mailer-plugin/pull/39

Doc/Schema Generation

JENKISN/plugin/configuration-as-code/

Jenkins Configuration as Code Reference

jenkins

agentProtocols

list of String

authorizationStrategy

AuthorizationStrategy

- loggedInUsersCanDoAnything
- legacy
- projectMatrix
- globalMatrix
- unsecured

clouds

list of Cloud

crumbIssuer

CrumbIssuer

- standard

disableRememberMe

boolean

labelString

String

markupFormatter

In such places as project description, user description, view description, and build description, Jenkins configuration determines how such free-form text is converted to HTML. By default, Jenkins treats the of the backward compatibility.)

While this is convenient and people often use it to load <iframe>, <script>. and so on to mash up data

JENKINS/plugin/configuration-as-code/schema


```
{
  $schema: "http://json-schema.org/draft-06/schema#",
  id: "http://jenkins.io/configuration-as-code#",
  description: "Jenkins Configuration as Code",
  type: "object",
- properties: {
  - jenkins: {
        type: "object",
        ref: "#/definitions/jenkins.model.Jenkins"
    },
  - credentials: {
        type: "object",
        ref: "#/definitions/com.cloudbees.plugins.credentials.CredentialsStore"
    },
  - : {
        type: "object",
        ref: "#/definitions/com.cloudbees.plugins.credentials.GlobalCredentialsConfiguration$Category"
    },
  - security: {
        type: "object",
        ref: "#/definitions/jenkins.model.GlobalConfigurationCategory$Security"
    },
  - unclassified: {
        type: "object",
        ref: "#/definitions/jenkins.model.GlobalConfigurationCategory$Unclassified"
    },
  - Tool: {
        type: "object",
        ref: "#/definitions/jenkins.tools.ToolConfigurationCategory"
    },
  - mavenmoduleset: {
        type: "object",
        ref: "#/definitions/hudson.maven.MavenModuleSet$DescriptorImpl"
    },
  - masterBuild: {
        type: "object",
        ref: "#/definitions/jenkins.model.MasterBuildConfiguration"
    },
  - quietPeriod: {
        type: "object",
        ref: "#/definitions/jenkins.model.GlobalQuietPeriodConfiguration"
    },
```

Corner cases

Some components hardly fit this model

For those we can develop dedicated Configurator adapter classes.

Under the hood

Root Elements → RootElementConfigurator

```yaml
jenkins:
  securityRealm:
      ldap:
        configurations:
          - server: ldap.acme.com
            rootDN: dc=acme,dc=fr
            managerPasswordSecret: ${LDAP_PASSWORD}
        cache:
          size: 100
         ttl: 10
        userIdStrategy: CaseSensitive
        groupIdStrategy: CaseSensitive

tool:
  git:
    installations:
      - name: git
      - path: /bin/git
```


46

Root Element

- JenkinsConfigurator
  "`jenkins`" → Jenkins.instance root object
- GlobalConfigurationCategoryConfigurator
  "`tools`", "`security`", … → Descriptors grouped by categories
- DescriptorRootElementConfigurator
  Uncategorized Descriptors *with* a global configuration page
  "`mailer`", …
- CredentialsRootConfigurator
  "`credentials`" → Glue code for credentials plugin (more on this later)

47

Child element → Attribute

```yaml
jenkins:
  securityRealm:
    ldap:
      configurations:
        - server: ldap.acme.com
          rootDN: dc=acme,dc=fr
          managerPasswordSecret: ${LDAP_PASSWORD}
      cache:
        size: 100
        ttl: 10
      userIdStrategy: CaseSensitive
      groupIdStrategy: CaseSensitive
```


48

Attribute

Configurator do describe a target component as a set of Attributes

Attribute handle :

- Name
- Type (inferred by reflection on generics)
- Multiplicity (Collection<x>)
- Setting value

Generic Attribute

writable JavaBean property | DataBound constructor parameter

```
public void setSecurityRealm(SecurityRealm securityRealm) {
```

```
jenkins:
  securityRealm:
    ldap:
```

SecurityRealm is an ExtensionPoint (abstract)

Configuration-as-Code need to identify implementation

Extension point implementation

SecurityRealm is an ExtensionPoints => candidates implementations:

LegacySecurityRealm	→ @Symbol("legacy") →	`legacy`
HudsonPrivateSecurityRealm	→ @Symbol("local") →	`local`
ActiveDirectorySecurityRealm	→ ActiveDirectory →	`activedirectory`
LDAPSecurityRealm	→ LDAP →	`ldap`

```
jenkins:
  securityRealm:
    ldap:
```


51

Build target Component

```
@DataBoundConstructor public LDAPSecurityRealm(
        List<LDAPConfiguration> configurations,
        boolean disableMailAddressResolver,
        CacheConfiguration cache,
        IdStrategy userIdStrategy,
        IdStrategy groupIdStrategy)
```

+ DataBoundSetters

```
jenkins:
  securityRealm:
    ldap:
      configurations:
        ...
      cache:
        size: 100
        ttl: 10
      userIdStrategy: CaseSensitive
      groupIdStrategy: CaseSensitive
```


52

Corner cases

- Setter method defined for internal needs / backward compatibility
  We exclude @Deprecated and @Restricted

- [WiP] Technical facing Property name : “labelString”
  We support @Symbol on setters

- Not a Describable / Internal model is … weird for end-user
  Custom Configurator | Attribute implementation

53

Custom Configurator, a.k.a “Glue Code”

Sample : expose a **user-friendly** credentials model

```
credentials:
 system:
  domainCredentials:
   # global credentials
   - credentials:
      - certificate:
          scope:    SYSTEM
          id:       ssh_private_key
          password: ${SSH_KEY_PASSWORD}
          keyStoreSource:
           fileOnMaster:
             keyStoreFile: /docker/secret/id_rsa
```

CredentialsRootConfigurator
custom code

A fake Attribute "system"
to expose DomainCredentials (List)
with custom setter implementation:

```
target.setDomainCredentialsMap(
        DomainCredentials.asMap(value)
)
```

Status

1.0 is there !
… even 1.3 (released last week)

We welcome Feedback !

- jenkins-users mailing list
- jenkinsci/configuration-as-code gitter
- github issues

56

Features

- Read configuration from local drive or url, REST API or CLI
- Reload configuration (Manage Jenkins → Configuration as Code → Reload)
- Export existing jenkins instance configuration into yaml (*here be dragons*)
- Compatibility dashboard :
  https://issues.jenkins-ci.org/secure/Dashboard.jspa?selectPageId=17346
  Please report issues with "`jcasc-compatibility`" label

+ Additionally docker demo setup (which can be easily adapted for different than demo purpose): https://github.com/Praqma/praqma-jenkins-casc

JEP-201

Make this **THE** configuration component for Jenkins community

https://github.com/jenkinsci/jep/blob/master/jep/201/README.adoc

Give it a try

Report missing plugin support / broken features

Contribute test cases (easy) or fixes (not so easy :P)

chat on gitter

How to talk to us?

- github issues working well for reporting problems
- we're monitoring Jenkins Users, Jenkins Developers mailing lists

but...

- **gitter** channel is a place to go to: https://gitter.im/jenkinsci/configuration-as-code-plugin

Questions?

DevOps World

Jenkins World

Thank you!