DR. STEFAN SCHLOTT

BeOne Stuttgart GmbH

Java-Entwickler, Scala-Enthusiast, Linux-Jünger

Seit jeher begeistert für Security und Privacy

MAX RIECHELMANN

BeOne Stuttgart GmbH

Consultant, Linux Fan, vim Enthusiast

Begeistert sich für Betriebssysteme

WIESO AUTOMATISIERUNG?

Unsere Projektsituation - und unsere Wünsche

AUSGANGSSITUATION

Zwei Jenkins-Master

A

sehr heterogen,
viele Plugins

B

extrem viele
(gleichförmige) Projekte

AUSGANGSSITUATION

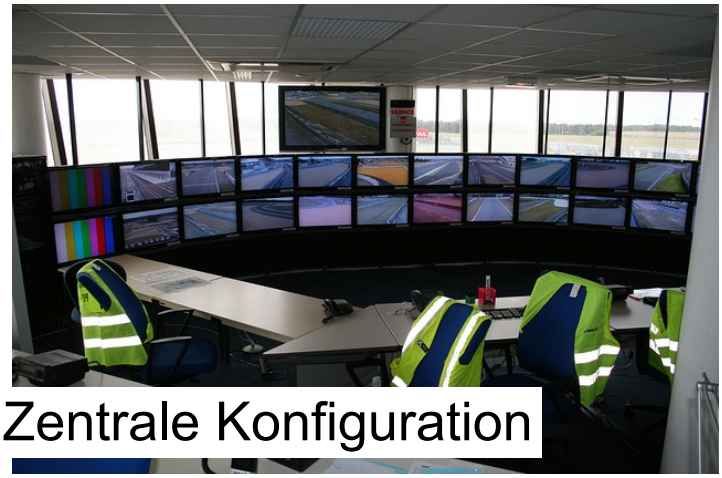
Angst vor Updates

Skalierungsproblem

Ablösung veralteter Plugins

Updates am „offenen Herzen“

ZIELE

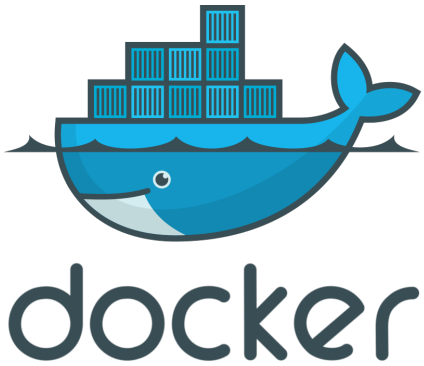


Zentrale Konfiguration

Replizierbarkeit

Testbarkeit

TOOL-AUSWAHL



Inzwischen: Entscheidung war gut, da auf Kubernetes / Openshift ausgerollt wird

ANSIBLE

```yaml
---
- name: Setup IAC-System
  hosts: localhost
  become: false
  vars_files:
  - "configuration/config.yml"

  tasks:
  - name: Force delete old docker image
    shell: |
      docker rmi --force docker-registry-default.example.com/project/iac_{{ configuration }}

  - name: Create build dir
    file:
      path: build
      state: directory

  - name: Copy from src to build folder
    command: "{{ item }}"
    with_items:
    - "cp -r src/ansible build/"
    - "cp -r src/Dockerfile build/"
    - "cp -r src/docker-root build/"
    - "cp -r src/hosts build/"

  - name: Copy configurations
    command: "{{ item }}"
    with_items:
    - "cp configuration/{{ configuration }}/setup.yml build/ansible/setup.yml"
    - "cp -r configuration/{{ configuration }}/vars build/ansible/"
    - "cp -r configuration/{{ configuration }}/jobs build/docker-root/"

  - name: Set correct file mode to executables
    file:
      path: "{{ item }}"
      mode: 0755
    with_items:
    - "build/docker-root/entrypoint.sh"
    - "build/docker-root/startjenkins.sh"
    - "build/ansible/setup.yml"
```

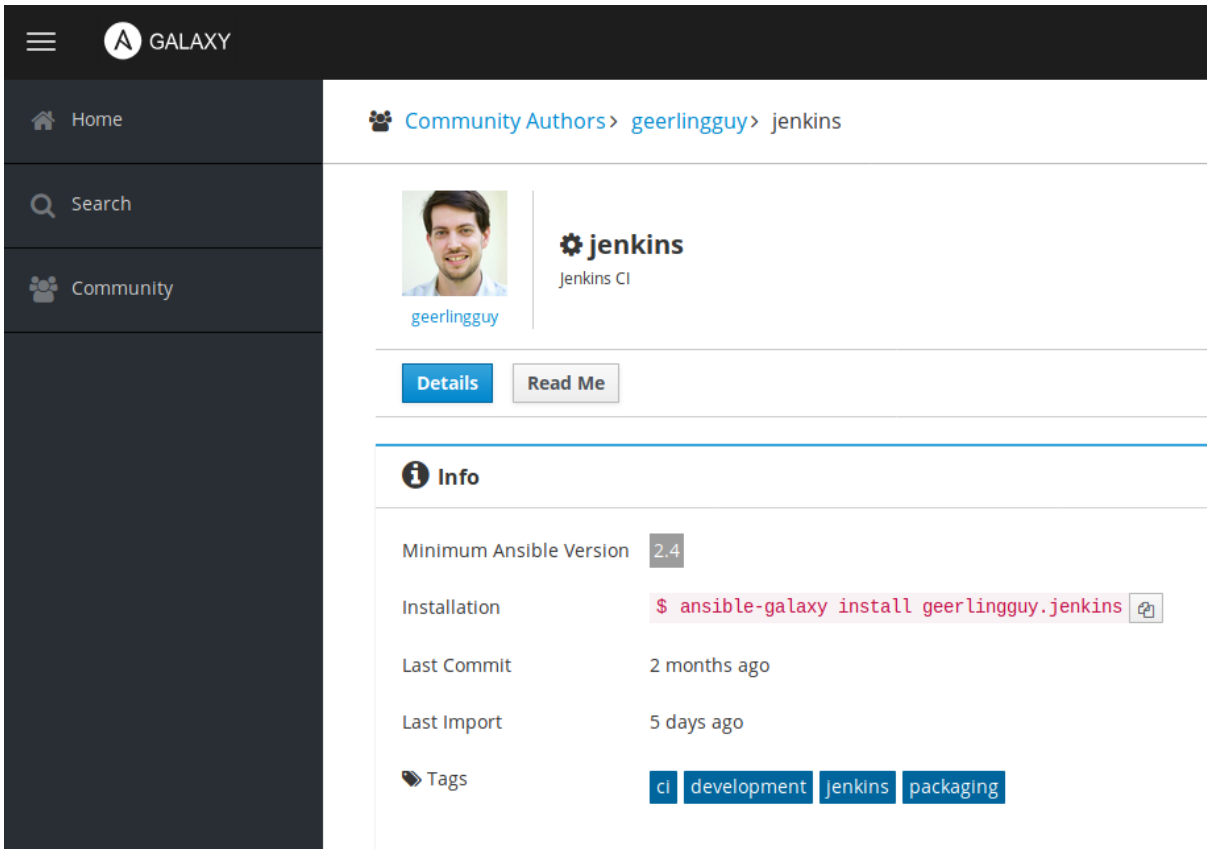
```
ansible-playbook playbook.yml
```

DOCKER

http://www.commitstrip.com/en/2016/06/24/how-to-host-a-coder-dinner-party/

JENKINS AUTOMATISCH AUFSETZEN

GEERLINGGUY JENKINS



GEERLINGGUY JENKINS

```
- name: Download specific Jenkins version.
  get_url:
    url: "{{ jenkins_pkg_url }}/ \\
      jenkins-{{ jenkins_version }}-1.1.noarch.rpm"
    dest: "/tmp/jenkins-{{ jenkins_version }}-1.1.noarch.rpm"
  when: jenkins_version is defined

...

- name: Install our specific version of Jenkins.
  package:
    name: "/tmp/jenkins-{{ jenkins_version }}-1.1.noarch.rpm"
    state: present
  when: jenkins_version is defined and specific_version.stat.exists
notify: configure default users
```

PLUGINS BEHIND PROXY

Jenkins_plugin module fails with proxy #32440

Open **xlammertink** opened this issue on Nov 1, 2017 · 10 comments

xlammertink commented on Nov 1, 2017 • edited

ISSUE TYPE

- Bug Report

COMPONENT NAME

Module: jenkins_plugin

PLUGINS BEHIND PROXY

```
#!/usr/bin/env python
...
os.system("wget https://updates.jenkins.io/latest/" \
  + key + ".hpi -P" + plugin_dir + "> /dev/null 2> /dev/null")
file = key + ".hpi"
if os.path.isfile(os.path.join(plugin_dir, file)):
  z = zipfile.ZipFile(os.path.join(plugin_dir, file))
  for line in z.open("META-INF/MANIFEST.MF"):
    if line.startswith("Plugin-Dependencies:"):
      independencies = 1
      dependencies += line.rstrip() \
        .replace("Plugin-Dependencies: ", "")
  z.close()
  del z
...
```

CONFIGURATION AS CODE

```
jenkins:
  securityRealm:
    ldap:
      configurations:
        - groupMembershipStrategy:
            fromUserRecord:
              attributeName: "memberOf"
          inhibitInferRootDN: false
          rootDN: "dc=acme,dc=org"
          server: "ldaps://ldap.acme.org:1636"
```

CONFIGURATION AS CODE

Einfache Konfiguration mittels YAML

aber

nicht alle Optionen werden unterstützt.

BEISPIEL SCRIPT SECURITY

- **Problem:** Script Security blockt unbekannte scripts
- **Lösung:** Eigene scriptApproval.xml nutzen

```
<scriptApproval plugin="script-security@1.50">
  <approvedScriptHashes>
    <string>c7f1c59f27b48976b58025961c5456397666cd3b</string>
  </approvedScriptHashes>
  ...
</scriptApproval>
```

Sanitycheck: **http://jenkins-url.com/systemInfo**

Environment Variables

Name ↓	Value
_	/usr/bin/java
HOME	/home/
LANG	en_US.UTF-8
LOGNAME	
PATH	/sbin:/usr/sbin:/bin:/usr/bin
PWD	/
SHELL	/bin/bash
SHLVL	2
TERM	xterm
USER	

Plugins

Name ↓	Version	Enabled
ace-editor	1.1	true
analysis-core	1.96	true
analysis-model-api	5.0.2	true
ant	1.9	true
antisamy-markup-formatter	1.5	true
apache-httpcomponents-client-4-api	4.5.5-3.0	true
artifactory	3.2.2	true

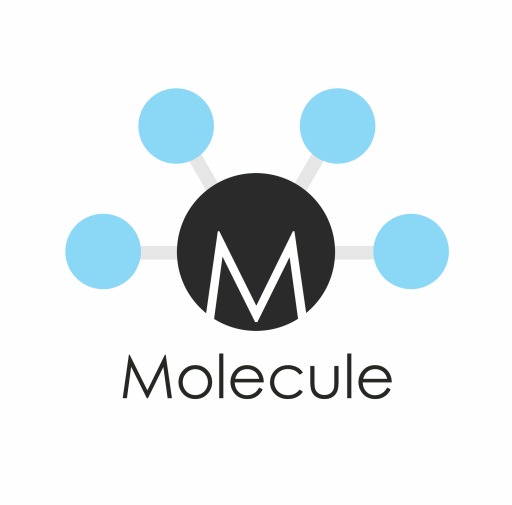
TESTING

- Aus Versionierung auschecken
- Bauen
- Nach Artifactory hochladen
- Ergebnis auf Korrektheit prüfen

TESTING

Teilmenge realer Jobs in Sandbox Umgebung testen

TESTING



Weiteres Vorgehen: Ansible Molecule

AUTOMATISCHES ERZEUGEN VON JOBS

MOTIVATION

- Zentrale Konfiguration
- Template Plugin ersetzen

ZENTRALE ANSIBLE KONFIGURATION

```
- hosts: localhost
  vars_files:
  - 'vars/vault.yml'
  vars:
    ...
    jenkins_version: "2.169"
    jenkins_plugins:
      - { id: 'job-dsl', version: '1.70' }
      - { id: 'anything-goes-formatter', version: '1.0' }
      - { id: 'subversion', version: '2.12.1' }
    ...
  roles:
    - role: geerlingguy.jenkins
    - role: task.docker
    - role: task.prodconfig
    ...
```

Type

Groovy template transformation

Script

```
<%

// global defines for the template
def branch = parent.parent.instance?.repoBranch ?: parent.parent.name
if (branch=='trunk') {
  branch = null
}
def sourceBasePath = 'base'
def repoUrl = parent.parent.instance?.repoName ?: 'https://example.com'

// major version for triggering debug builds
def majorVersion = branch ?:'1'

// trigger a downstream debug build?
boolean isDebug = (name == 'global_dmc' || name == 'global_obt')

def isRepoSpecific = [ 'dir', 'esi_nab' ].contains(name)
def repoSpecificInfo = isRepoSpecific ? '' (repository specific)'' : ''

def displayName = displayNames[name]

def componentTypeName = 'Debug Component'

def gradleJobTypeArgs = ''
if (name.startsWith('framework-')) {
  gradleJobTypeArgs = ''-Ptype=base-framework -Psubtype=${name.substring('framework-'.length())}''
} else {
  gradleJobTypeArgs = ''-Ptype=base-${name}''
}

def introduction = ''
def sourceUrl = ''
def sourcePath = '' // for excluding from commit notification, does not end with /


if (branch) {
  introduction = ''&lt;p&gt;This job builds &lt;strong&gt;${componentTypeName}
&quot;${displayName}&quot; (&quot;${name}&quot;)&lt;/strong&gt; from the development branch
&lt;strong&gt;${branch}&lt;/strong&gt;.${repoSpecificInfo}.&lt;br&gt;Note that the build scripts for this
component might be branched, see option in the branch folder.&lt;/p&gt;''
  sourceUrl = ''${repoUrl}/${sourceBasePath}/${name.replace('-', '/')}/branches/${branch}''
  sourcePath = ''/${sourceBasePath}/${name.replace('-', '/')}/branches/${branch}''
} else {
```

JOB-DSL

```
def jobs = ... // Each folder in SVN is a job in this case

// Create jobs from jobs array
jobs.each { currentjob ->
  job("${currentjob.branch}/base/${currentjob.name}") {
    scm {
      svn {
        location("{{ svnurl }}") {
          directory("build")
          ...
        }
        configure { xml ->
          xml / 'additionalCredentials' <<
            'hudson.scm.SubversionSCM_-AdditionalCredentials' {
              credentialsId "foobar"
            }
```

JENKINS CLI

```
- name: Create jobs from /job-dsl/
  template:
    src: "job-dsl.xml"
    dest: "/tmp/job-dsl_{{ item | basename }}.xml"
  with_fileglob: "/tmp/jobs/*"

- name: Create job-dsl seed
  shell: |
    java -jar {{ jenkins_jar_location }} -s http://localhost:8080/ \\
      create-job {{ item | basename }}  < "/tmp/job-dsl_\\
      {{ item | basename }}.xml"
  with_fileglob: "/jobs/*"
```

PERSISTENZ

PERSISTENZ LÖSUNG?

- Bei beenden des Containers Zustand speichern und nach Neustart wieder einspielen?
- Dateisystem welches nur History, Buildcounter etc. speichert?

Derzeit noch offen.

FAZIT

PROBLEME

DEPLOYMENT AUF OPENSHIFT

```
- name: Build iac-system docker image
  docker_image:
    name: >
          docker-registry-default.osh.example.com/
          project/iac_{{ configuration }}
    nocache: true
    debug: yes
    tag: latest
    push: yes
    path: build
    state: present
    buildargs:
      http_proxy: "{{ http_proxy }}"
      https_proxy: "{{ https_proxy }}"
      no_proxy: localhost
```

FAZIT

Jenkins auf Knopfdruck