

CONTINUOUS INTEGRATION MIT JENKINS

KANYILDIZ MUHAMMEDHIZIR – 5YHITM

Inhaltsverzeichnis

1	Continuous Integration	1
1.1	Erklärung	1
1.2	Jenkins als CI	1
2	Intalaltion von Jenkins	2
2.1	Erklärung	2
2.2	Windows Install	2
2.3	Passwort finden und eingeben.....	2
2.4	Standard Installation	3
3	Oberfläche	4
3.1	Plugins.....	4
3.2	Suchen	4
3.3	Installieren	5
4	Install Packages.....	5
4.1	Pip	5
4.2	Nose	5
4.3	Pylint.....	6
5	Neues „Job“ erstellen.....	6
5.1	„Job“ Einstellungen	7
6	Fehler beim Builden.....	8

1 Continuous Integration

1.1 Erklärung

Continuous Integration (CI) ist ein Teil der modernen Software Entwicklung. CI stellt den Prozess dar, der das Bauen und Testen einer Anwendung abbildet. Mit Hilfe von CI lassen sich Fehler schneller finden und beheben. Die Idee der kontinuierlichen Integration ist es, dass die Entwickler frühzeitig und regelmäßig Änderungen in das Versionsmanagement einchecken. Diese Änderungen sollten funktionsfähig sein, sodass die gesamte Applikation auf Integrationsprobleme geprüft werden kann. Es ist somit die Verfügbarkeit einer lauffähigen Version gegeben, die dann z. B. für anderweitige Testzwecke oder Vertriebszwecke genutzt werden kann. Eine typische Anwendung sind sogenannte Nightly Builds, bei denen zu einer vorgegebenen Uhrzeit der aktuelle Programmcode übersetzt wird und dabei Tests mit der erstellten Software automatisch ausgeführt werden. Bei gefundenen Problemen kann ein Entwickler dann z. B. direkt per Mail über das gefundene Problem informiert werden.

1.2 Jenkins als CI

Jenkins (ehemals Hudson) ist ein webbasiertes Open Source Continuous Integration System. Es ist in Java geschrieben und plattformunabhängig. Die Basis von Jenkins unterstützt zahlreiche Werkzeuge darunter SVN, Ant, Maven sowie JUnit. Durch die Community können weitere Funktionen mit Hilfe von Plugins hinzugefügt werden. Somit lässt sich Jenkins für jedes Projekt individuell anpassen. Auch für Projekte mit anderen Sprachen/Technologien wie z. B. PHP, Ruby oder .NET ist Jenkins geeignet. Testwerkzeuge lassen sich über Plugins über die intuitive Benutzeroberfläche integrieren. Builds können durch verschiedene Auslöser gestartet werden: z. B. Änderung des CVS oder Zeitplan (z. B. Nightly Builds). Nightly Builds sind besonders bei Open Source Projekten zu finden und bedeutet, dass die Applikation nachts gebaut und getestet wird.

2 Intalaltion von Jenkins

2.1 Erklärung

Für Jenkins stehen bereits native Pakete bereit (Windows, Ubuntu/Debian, Mac OS X, uvm). Ebenso findet man es als Webarchiv (.war Datei). Diese kann alternativ auf weiteren Plattformen manuell installiert werden.

2.2 Windows Install

Für Windows Benutzer steht Jenkins als ZIP Archiv zur Verfügung, welches eine setup.exe enthält. Diese installiert bei Bedarf fehlende .NET Bibliotheken. Man wird grafisch durch die Installation geführt und kann Einstellungen wie z. B. Installationspfad angeben. Jenkins ist nun als Dienst vorhanden. Dieser lässt sich, wie in Windows üblich, starten und stoppen. Ist der Dienst gestartet, lässt sich Jenkins über den Port 8080 erreichen (<http://hostname:8080>).

2.3 Passwort finden und eingeben

In der Datei initialAdminPassword befindet sich der bestätigungs Passwort für den Adminitstrator.

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, the initialAdminPassword has been written to the log (not sure where to find it?) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Continue

filepath-filters.d	05.03.2017 19:03	File folder	
whitelisted-callables.d	05.03.2017 19:03	File folder	
hudson.util.Secret	05.03.2017 19:03	SECRET File	1 KB
jenkins.model.Jenkins.crumbSalt	05.03.2017 19:03	CRUMBSALT File	1 KB
jenkins.security.ApiTokenProperty.seed	05.03.2017 19:03	SEED File	1 KB
master.key	05.03.2017 19:03	KEY File	1 KB
org.jenkinsci.main.modules.instance_id...	05.03.2017 19:03	KEY File	1 KB
slave-to-master-security-kill-switch	05.03.2017 19:03	File	1 KB

2.4 Standard Installation

Bei dieser Installation wurde die Standard installation deshalb ausgewählt, weil dies schon einige der Erforderlichen Plugins enthält. Und ein neuer Benutzer wird gleich danach angelegt für das ein und ausloggen.

Getting Started

Getting Started

✓ Folders Plugin	✓ OWASP Markup Formatter Plugin	build timeout plugin	Credentials binding Plugin	OWASP Markup Formatter Plugin Folders Plugin ** Structs Plugin ** JUnit Plugin OWASP Markup Formatter Plugin PAM Authentication plugin ** OWASP Markup Formatter Plugin
Timestamp	Workspace Cleanup Plugin	Ant Plugin	Gradle Plugin	
Pipeline	GitHub Organization Folder Plugin	Pipeline: Stage View Plugin	Git	
Subversion Plug-in	SSH Slaves plugin	Matrix Authorization Strategy Plugin	✓ PAM plug	
LDAP Plugin	Email Extension Plugin	Mailer Plugin		

Jenkins 2.32.3

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

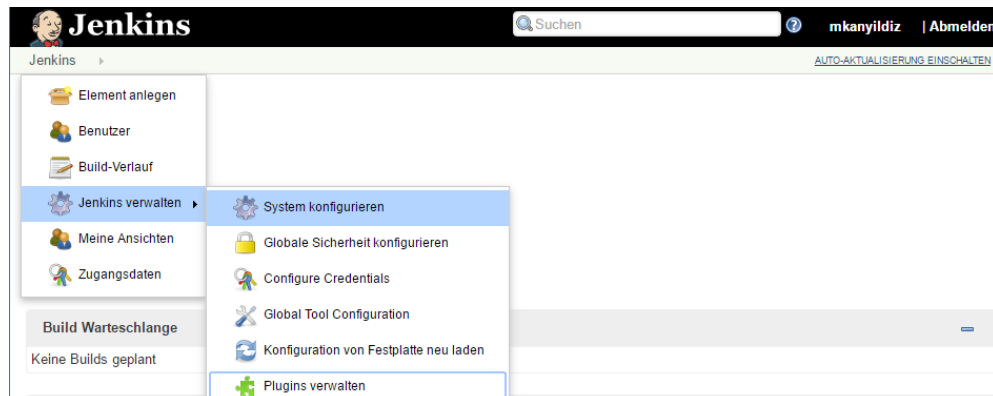
Jenkins 2.32.3

Continue as admin

3 Oberfläche

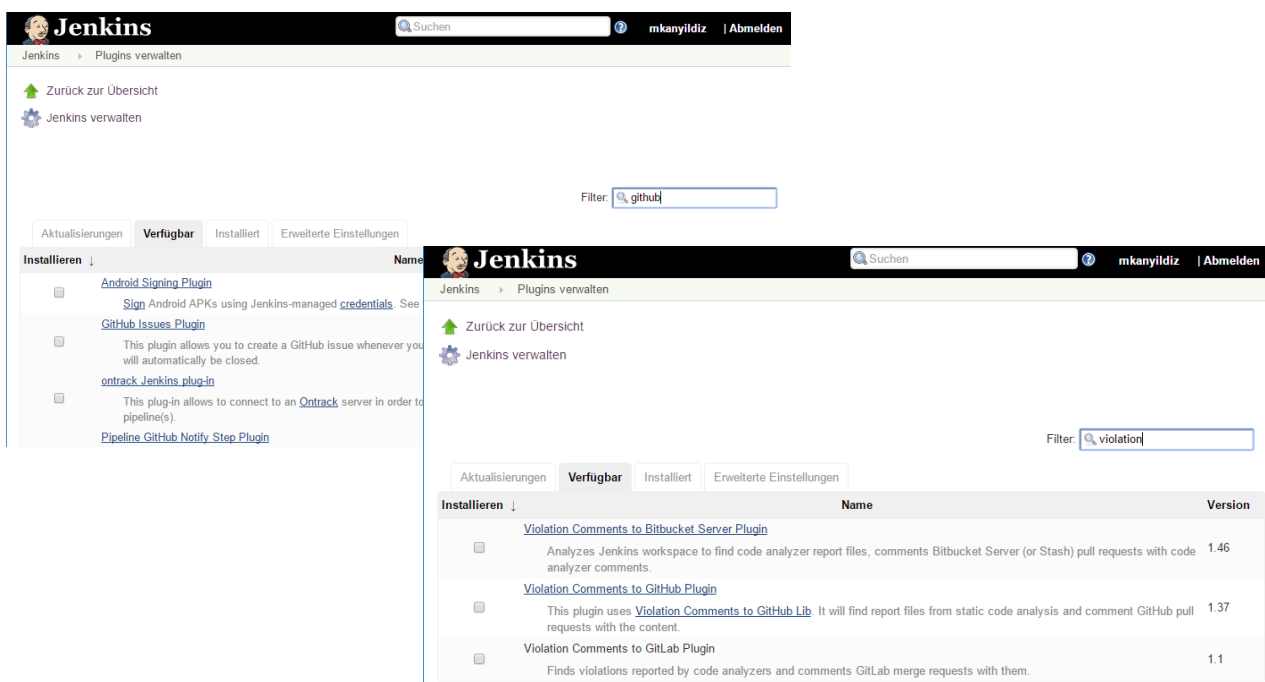
3.1 Plugins

Um zusätzliche Plugins zu installieren, muss der Administrator auf Jenkins -> Jenkins verwalten -> Plugins verwalten gehen. (Wie auf dem Screenshot zusehen)



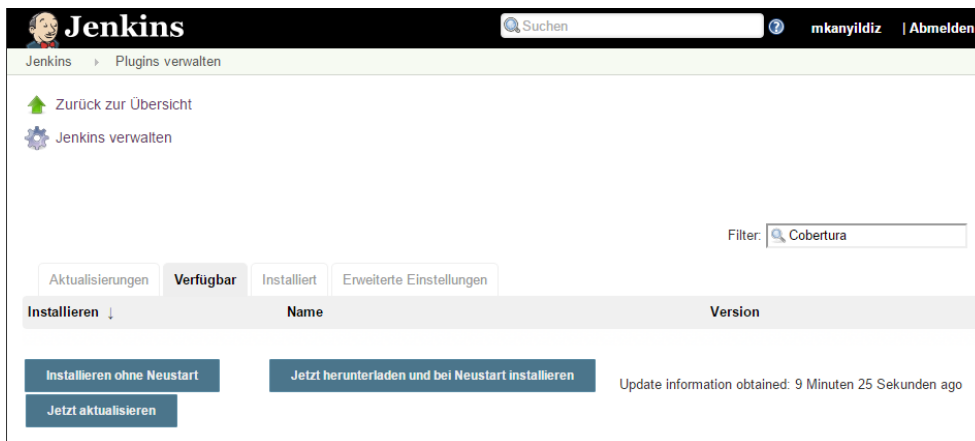
3.2 Suchen

Um bestimmte Plugins zu suchen muss der Administrator auf den Tab Verfügbar gehen und in die Suchleiste den namen des Plugins eingeben. (Git Plugin, Violations, Cobertura)



3.3 Installieren

Die Installation eines Plugins erfolgt einfach durch einen Knopfdruck. (Installation ohne Neustart oder Installation mit Neustart.)



4 Install Packages

4.1 Pip

```
root@sew:/home/sew# apt-get install python
```

4.2 Nose

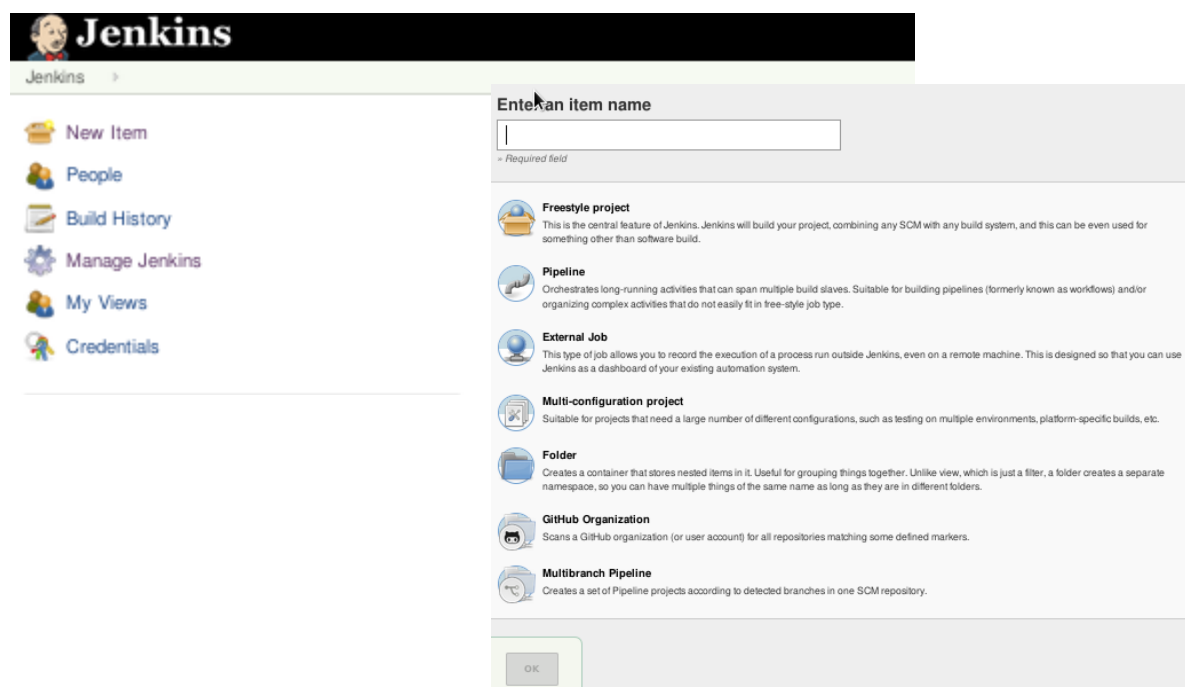
```
root@sew:/home/sew# pip install nose
Collecting nose
  Downloading nose-1.3.7-py2-none-any.whl (154kB)
    100% |#####| 163kB 1.4MB/s
Installing collected packages: nose
Successfully installed nose-1.3.7
root@sew:/home/sew#
```


4.3 Pylint

```
root@sew:/home/sew# pip install pylint
Collecting pylint
  Downloading pylint-1.6.5-py2.py3-none-any.whl (577kB)
    100% |#####| 583kB 1.8MB/s
Collecting mccabe (from pylint)
  Downloading mccabe-0.6.1-py2.py3-none-any.whl
Collecting astroid<1.5.0,>=1.4.5 (from pylint)
  Downloading astroid-1.4.9-py2.py3-none-any.whl (213kB)
    100% |#####| 215kB 3.2MB/s
Collecting configparser; python_version == "2.7" (from pylint)
  Downloading configparser-3.5.0.tar.gz
Collecting backports.functools-lru-cache; python_version == "2.7"
  Downloading backports.functools_lru_cache-1.3-py2.py3-none-any.
Requirement already satisfied: six in /usr/lib/python2.7/dist-pac
(int)
Collecting isort>=4.2.5 (from pylint)
  Downloading isort-4.2.5-py2.py3-none-any.whl (40kB)
    100% |#####| 40kB 1.8MB/s
Collecting lazy-object-proxy (from astroid<1.5.0,>=1.4.5->pylint)
  Downloading lazy_object_proxy-1.2.2-cp27-cp27mu-manylinux1_x86_
    100% |#####| 61kB 5.9MB/s
Collecting wrapt (from astroid<1.5.0,>=1.4.5->pylint)
```

5 Neues „Job“ erstellen

Um ein neuen Job zu srstellen, geht man zuerst auf den Side-Menü und als nächstes auf New Item. Danach wird Freestyle Projekt ausgewählt und auf OK gedrückt.



5.1 „Job“ Einstellungen

General

Source Code Management

Build Triggers

Build Environment

Build

Post-build Actions

Project name

GitBruch

Description

[\[Plain text\]](#) [Preview](#)

☐ Discard old builds

☐ GitHub project

☐ This project is parameterized

☐ Throttle builds

☐ Disable this project

☐ Execute concurrent builds if necessary

Source Code Management

☐ None

☒ Git

Repositories

Repository URL

https://github.com/mkanyildiz01/GitBruchJenkins

Credentials

- none -

Add

Name

GitBruchJenkins

Refspec

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/master/bruch

Add Branch

Repository browser

(Auto)

Additional Behaviours

Add

☐ Subversion

Build Triggers

☐ Trigger builds remotely (e.g., from scripts)
☐ Build periodically
☒ Poll SCM

Schedule

⚠ Do you really mean "every minute" when you say "*****"? Perhaps you meant "H ****" to poll once per hour

Would last have run at Sonntag, 05. März 2017 20:24 Uhr MEZ; would next run at Sonntag, 05. März 2017 20:24 Uhr MEZ.

☐ Ignore post-commit hooks
☐ Build after other projects are built

Build

Execute shell

Command

```
PYTHONPATH=""  
nosetests --with-xunit --all-modules --traverse-namespace --with-coverage --cover-package=project1  
python -m coverage xml --include=project1*  
pylint -f parseable -d I0011,R0801 project1 | tee pylint.out
```

See [the list of available environment variables](#)

Advanced...

Add build step

6 Fehler beimi Builden

The screenshot shows the Jenkins web interface for a project named 'Project GitBruch'. The top navigation bar includes the Jenkins logo and the breadcrumb 'Jenkins > GitBruch >'. On the left, a sidebar contains links to 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', 'Git Polling Log', and 'Move'. The main content area is titled 'Project GitBruch' and features links to 'Workspace' and 'Recent Changes'. Below this, a 'Permalinks' section lists 'Last build (#1), 0.32 sec ago'. A 'Build History' panel is open, showing a search bar with 'find', a table with one entry '#1' dated 'Mar 5, 2017 8:26 PM', and RSS feeds for 'all' and 'failures'.

Jenkins

Jenkins > GitBruch >

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build Now](#)

[Delete Project](#)

[Configure](#)

[Git Polling Log](#)

[Move](#)

Project GitBruch

[Workspace](#)

[Recent Changes](#)

Permalinks

- [Last build \(#1\), 0.32 sec ago](#)

Build History [trend](#)

find x

#1 Mar 5, 2017 8:26 PM

[RSS for all](#) [RSS for failures](#)