JUST DO IT
HÖHERE ABTEILUNG FÜR
INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Protokoll

Continuous Integration

SEW
5BHIT 2015/16

Michael Weinberger

Version 1.0
Begonnen am 9. Februar 2016
Beendet am 16. Februar 2016

Note:
Betreuer: Dolezal/Vittori

Inhaltsverzeichnis

1 Einführung

"Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly. This article is a quick overview of Continuous Integration summarizing the technique and its current usage."M.Fowler

Schreibe fünf Testfälle für dein CSV-Projekt und lass diese mithilfe von Jenkins automatisch bei jedem Build testen!

- Installiere auf deinem Rechner bzw. einer virtuellen Instanz das Continuous Integration System Jenkins
- Installiere die notwendigen Plugins für Jenkins (Git Plugin, Violations, Cobertura)
- Installiere Nose und Pylint (mithilfe von pip)
- Integriere dein CSV-Projekt in Jenkins, indem du es mit Git verbindest
- Schreibe fünf Unit Tests für dein CSV-Projekt
- Konfiguriere Jenkins so, dass deine Unit Tests automatisch bei jedem Build durchgeführt werden inkl. Berichte über erfolgreiche/fehlgeschlagene Tests und Coverage
- Protokolliere deine Vorgehensweise (inkl. Zeitaufwand, Konfiguration, Probleme) und die Ergebnisse (viele Screenshots!)

Viel Spaß! :)

 Michael Weinberger

2 Durchführung

2.1 Jenkins installieren

Ich habe mich entschieden Jenkins in einer virtuellen Instanz zu installieren. Verwendet wurde Ubuntu 15.10 unter VMware Workstation 12. Folgende Befehle waren notwendig:

```
wget -q -O - https://jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt-key add -
sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian binary/ > /etc/apt/sources.list.d/jenkins.list'
sudo apt-get update
sudo apt-get install jenkins
```

Listing 1: Jenkins in Ubuntu installieren [1]

Sollte der Port 8080 bereits belegt sein und /etc/init.d/jenkins fehlschlägt, muss in /etc/default/jenkins der Wert bspw. auf 'HTTP_PORT=8081' gesetzt werden.

2.2 Installieren der notwendigen Plugins

Im Menüpunkt 'Jenkins verwalten' findet sich der Punkt 'Plugins verwalten'. Im Reiter 'Verfügbar' lassen sich die drei benötigten Plugins `Git Plugin, Violations & Cobertura` auswählen und installieren, Jenkins wird per Klick auf die Checkbox neu gestartet, damit die Erweiterungen auch ausgeführt werden.
Das Git Plugin integriert, wie der Name sagt, die Git-Unterstützung in Jenkins, zusätzlich habe ich das Github Authentication Plugin installiert, um eine Authentifizierung per Github OAuth zu gewährleisten. Violations überprüft den Code auf Verstöße gegen Coding-Richtlinien, und Cobertura stellt Coverage-Reports bereit.

2.3 Installieren von Nose und Pylint

```
sudo pip install nose
sudo pip install pylint
```

Listing 2: Nose und Pylint mit pip installieren

`Nose` kümmert sich um Unittests, und `Pylint` überprüft den Code auf Fehler, und versucht einen bestimmten Codestandard zu erreichen.

 Michael Weinberger

2.4 Integriere dein CSV-Projekt in Jenkins, indem du es mit Git verbindest

Die nächsten Schritte habe ich aus dem Tutorial übernommen. [2]

Unterpunkte:

- Our first job
- 1 - Where to get the source
- 2 - When to run the job
- 3 - What to run

  Wichtig! Hier die Shell je nach System selbst anpassen!
- 4 - Interpret the results

  Auch hier einige wichtige Schritte, dieses Mal um die Plugins richtig zu verwenden.
- Run the job

2.5 Konfiguriere Jenkins so, dass deine Unit Tests automatisch bei jedem Build durchgeführt werden inkl. Berichte über erfolgreiche / fehlgeschlagene Tests und Coverage

Ein erster Build wird fertiggestellt mit folgendem Output:

```
Started by an SCM change
Building in workspace /var/lib/jenkins/jobs/csv/workspace
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/mweinberger-tgm/Continuous-Integration.git # timeout=10
Fetching upstream changes from https://github.com/mweinberger-tgm/Continuous-Integration.git
> git --version # timeout=10
> git -c core.askpass=true fetch --tags --progress https://github.com/mweinberger-tgm/Continuous-Integration.git
+refs/heads/*:refs/remotes/origin/*
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 8c6b0b6d136c67dbdda6e213fe310bfde36b7321 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 8c6b0b6sdfg235sfg324da6e213fe310bfde36b7321
> git rev-list 01c4c4cfef6607d5vfbwedsdfdf235a5507d901e9873 # timeout=10
[workspace] $ /bin/sh -xe /tmp/hudson6843542523459213337267.sh
-------------------------------------------------------------------
Ran 5 tests in 0.040s

...

```

```
28
Mit Shell:

PYTHONPATH=''
Nosetests-3.4 --with-xunit --all-modules --traverse-namespace --with-coverage --cover-package=
    CSVimportPython --cover-inclusive
33 python -m coverage xml
pylint -f parseable -d I0011,R0801 Main.py | tee pylint.out
```

Listing 3: Build #1

Daraufhin war das gewünschte Ergebnis sichtbar. Link zum Repo [3]

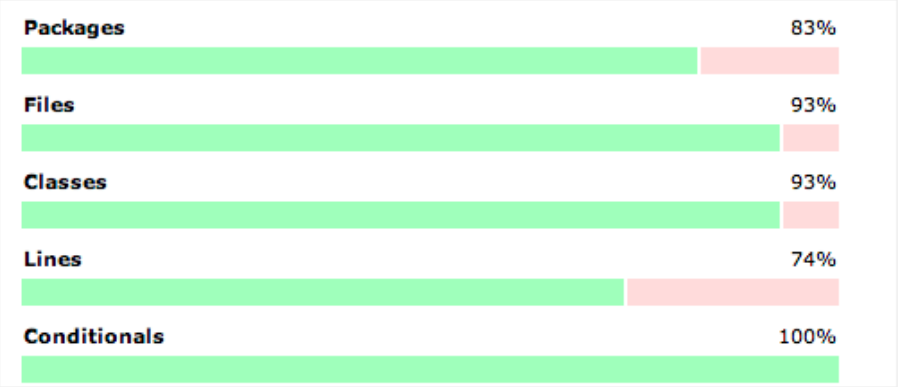

Abbildung 1: Beispiel Code Coverage

2.6 Zeitaufwand

Installation *5 min*, Konfiguration *1 h*, Troubleshooting *30 min*, Neuimport und Builden *15 min*

2.7 Probleme

Diese Aufgabe wurde zuerst in Linux gelöst, was aufgrund von enormen Dependencies kaum möglich war. Auch nach ca. 500 MB an Paketen haben gerne noch weitere gefehlt. Unter Windows gab es (komischerweise) weniger Probleme, die Builds waren auf Anhieb erfolgreich, dank richtiger Installation.

Literaturverzeichnis

[1] Jenkins Wiki. Jenkins installieren. `https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Ubuntu`. zuletzt besucht: 13.02.2016.

[2] Steve. Automated python unit testing, code coverage and code quality analysis with jenkins - part 3. `http://bhfsteve.blogspot.co.at/2012/04/automated-python-unit-testing-code_27.html`. zuletzt besucht: 13.02.2016.

[3] Michael Weinberger 5BHIT. Repository zur aufgabe. `https://github.com/mweinberger-tgm/Continuous-Integration.git`. zuletzt besucht: 13.02.2016.

Listings

Abbildungsverzeichnis