# Evaluationsbogen Continuous Delivery für Thoughtworks Go

Dieser Evaluationsbogen dient der Beurteilung von Werkzeugen für Continuous Delivery und dem Aufbau einer Deployment Pipeline.

Produkt: Thoughtworks Go

1. <http://www.thoughtworks-studios.com/docs/go/12.2/help/concepts_in_go.html>

## Erläuterung

### Punktevergabe

|  |  |
| --- | --- |
| 0 | Funktionalität wird nicht unterstützt / In der Eigenschaft nicht akzeptabel |
| 1 | Einfache Unterstützung der Funktionalität / In der Eigenschaft ausreichend |
| 2 | Durchschnittliche bzw. erwartete Unterstützung der Funktionalität / In der Eigenschaft gut |
| 3 | Überdurchschnittliche Unterstützung der Funktionalität / In der Eigenschaft sehr gut |

## Bewertungsbogen

|  |  |  |
| --- | --- | --- |
| Werkzeug | | Bewertung |
| Thourghtworks GO, Umsetzung für Testprojekt | |  |
| Funktionsumfang | |  |
| Schnittstellen | |  |
| Schnittstellen zu Versionierungssystemen | Subversion, Git, Mercurial, Perforce, Team-Foundation-Server |  |
| Artefakte-Repository | Go unterstützt Build- und Test-Artefakte, diese werden vom GO-Server verwaltet. Nach jedem Job können Artefakte, wie z.B. ein kompiliertes WAR-File, an eine |  |
| Unterstützung für die Commit-Stage | |  |
| Quellcode kompilieren | Unterstützt durch ANT, NAnt, Rake. Weitere können über Shell-Kommandos angestoßen werden. |  |
| Commit Test (Unit Test) | Siehe oben |  |
| Assemble | Siehe oben |  |
| Code Analyse | Siehe oben |  |
| Acceptance Stage | |  |
| Konfigurieren der Umgebung | Siehe oben |  |
| Deployen der Binärdateien | Siehe oben |  |
| Smoke Test | Siehe oben |  |
| Akzeptanztest | Siehe oben |  |
| Self-Service Deployment für Tester | |  |
| Konfigurieren der Umgebung | Siehe oben |  |
| Deployen der Binärdateien | Siehe oben |  |
| Smoke Test | Siehe oben |  |
| Kapazitätstest | Siehe oben |  |
| Unterstützung für die Produktionsumgebung | |  |
| Konfigurieren der Umgebung |  |  |
| Deployen der Binärdatein |  |  |
| Smoke Test |  |  |
| Report | |  |
| Auswertung und Beurteilung der Builds anhand von Testergebnissen und Laufzeiten |  |  |
| Funktionalitäten | |  |
| Liefern die Funktionen die erwarteten Ergebnisse? |  |  |
| Sind die Funktionen für den Aufbau einer Deployment-Pipeline geeignet? |  |  |
| Kann das System mit der bestehenden Infrastruktur zusammenarbeiten? |  |  |
| Wird der Zugriff unberechtigter verhindert? |  |  |
| Zuverlässigkeit | |  |
| Wie verhält sich das System bei Fehlerzuständen? |  |  |
| Benutzbarkeit | |  |
| Ist der Aufwand für Benutzer das Konzept des Systems zu verstehen gering? |  |  |
| Ist der Aufwand die Bedienung des Systems zu erlernen gering? |  |  |
| Übertragbarkeit | |  |
| Wie hoch ist der Aufwand das System an verschiedene Umgebungen anzupassen? |  |  |
| Wie hoch ist der Aufwand das System zu installieren? |  |  |
| Fazit | |  |
|  | | |

# Schritte

## Konzept von Toughtworks Go

Go Server:

Voraussetzungen für den Server:

* Unterstützte Systeme: Windows, Mac OSX, Linux, Solaris
* JRE6, 1-2GB RAM, 2GHz
* Separates Laufwerk für Artefakte

Go Agent:

Voraussetzungen für den Agent:

* Unterstützte Systeme: Windows, Mac OSX, Linux, Solaris
* JRE6, 128-256MB RAM, 2GHz

Go Client:

Voraussetzungen für den Client:

* Browser: Firefox 3+, Safari 3+, IE 7+8

## Installation

1. Anlegen von Virtuellen Maschinen:
   1. GO-Server: 1280 MB RAM, 8 GB HDD, 32-BIT CPU, Ubuntu
   2. GO-Client: 256 MB RAM, 8 GB HDD, 32-BIT, Ubuntu
   3. SVN-Server: 256 MB RAM, 8 GB HDD, 32-BIT CPU, Ubuntu
2. Installation von JAVA (<http://www.oracle.com/technetwork/java/javase/install-linux-64-self-extracting-140230.html>; <http://www.oracle.com/technetwork/java/javase/install-linux-self-extracting-142296.html>)
   1. Anlegen von Java Installationsverzeichnis (mkdir) /opt/java/32/
   2. Kopieren der von jre-6u31-linux-x32.bin nach /opt/java/32 mittels WinSCP
   3. Datei muss ausführbar gemacht werden (chmod +x jre-6u31-linux-i586.bin)
   4. Selbst-Extrahieren des Archives (sudo /opt/java/32 jre-6u31-linux-i586.bin)
   5. Umgebungsvariablen setzen: (export PATH=$PATH:/opt/java/32/jre1.6.0\_21/bin/)
3. Installation von Unzip „sudo apt-get install unzip”
4. SVN-Server
   1. Links:
      1. <http://www.pc-erfahrung.de/linux/netzwerk/subversion-unter-ubuntu.html>
   2. Installieren
   3. konfigurieren
      1. Verzeichnis anlegen (sudo mkldir /var/svn/repos)
      2. Nutzer ändern (sudo chown svnserver.svnserver /var/svn/repos)
      3. Repository anlegen (svnadmin create /var/svn/repos)
      4. Passwortdatei ändern: (vi /var/svn/repos/conf/svnserve.conf) Setzen des Parameters „password-db = passwd“. Damit wird die Datei „passwd“ genutzt, dort werden die Nutzer hineingeschrieben. Dies geschieht Klartext mit Nutzer und Passwort.
   4. Starten
      1. Server starten: (svnserve –d)
      2. Zugriff über svn://192.168.56.102:3690/var/svn/repos
5. Installation Go-Server
   1. Kopieren der Installationsdatei auf das System (go-server-12.2.0-15134.deb) (/var/tmp)
   2. Installation des Server „sudo dpkg -i go-server-12.2.0-15134.deb“ <http://www.thoughtworks-studios.com/docs/go/12.2/help/run_go_without_install.html>
   3. Go-Server starten: (Path für Java exportieren, java –jar go.jar, -XX:PermSize=256m und –XX:MaxPermSize=512m)
   4. <http://192.168.56.101:8153/go/>
6. Installation und Registrierung des Go-Agent
   1. Kopieren und Unzip der Go-Agent-Datei (/opt/go/)
   2. Client starten: (/opt/go/go-agent-12.2.0$ java –jar agent-bootstrapper.jar 192.168.56.101) Der Client ist dann über die Konsole aufrufbar und wird vom Server verwaltet.
7. Test-Projekt ANT Konfiguration
   1. <http://ant.apache.org/manual/index.html>
   2. SVN-Zugriff: svn://192.168.56.102:3690/var/svn/repos/HelloWorld
   3. Konfiguration Deploytask in ANT:
      1. <http://www.developer.com/java/other/article.php/998241/Building-with-Ant-Deployment-and-Distribution.htm>
8. Tomcat-6 Installieren und Konfigurieren:
   1. <https://help.ubuntu.com/11.04/serverguide/C/tomcat.html>
   2. <http://www.mkyong.com/tomcat/how-to-install-tomcat-in-ubuntu/>
   3. Installations-Pfad: /usr/share/tomcat6/bin und /etc/tomcat6/
   4. <http://192.168.56.102:8080/manager/html/> um die Management-Konsole aufzurufen
   5. User konfigurieren: sudo vi /etc/tomcat6/tomcat-users.xml
   6. Tomcat neustarten: sudo /etc/init.d/tomcat6 restart
9. JMeter Installation:
   1. Plug-In für ANT: <http://www.programmerplanet.org/pages/projects/jmeter-ant-task.php>
   2. Jmeter.zip nach /opt/jmeter/… entpackt
   3. Ant-jmeter.jar nach $ANT\_HOME/lib/ kopiert
10. JUnit Installation:
    1. Wird benötigt für die Commit Stage. Hier werden einfache Funktionalitäten der Anwendung getestet.
    2. Bibliotheken junit.jar und org.harmcrest.core.jar nach $Ant\_Home/lib/ kopiert
11. Pipeline-Setup
    1. Commit-Stage
       1. SVN-Material: SVN\_HelloWorld\_Autoupdate, svn://192.168.56.102:3690/var/svn/repos/HelloWorld, goserver
    2. Deploy-Stage

