Installation von WASKA auf einem Debian GNU/Linux 4.0 Etch-System

MITWIRKENDE

	TITEL :	*REFERENCE* :
	Installation von WASKA auf einem Debian GNU/Linux 4.0 Etch-System	

AKTION	*NAME*	*DATUM*	*UNTERSCHRIFT*
VERFASST DURCH	Sascha L. Teichmann	25. November 2008	

VERSIONSGESCHICHTE

NUMMER	DATUM	BESCHREIBUNG	NAME
0.1	2008-07-15	Initiale Version	slt
0.2	2008-07-15	Korrekturen	fk
0.3	2008-07-16	Beschreibung der mod_wsgi- und Pylons-Installation erweitert.	slt
0.4	2008-07-16	Erläuterung zu mod_ssl erweitert.	slt
0.5	2008-07-16	Formatierungen und SSL-Zertifikate	fk
0.6	2008-08-22	Formatierungen und Anleitung für Testsystem	fk
0.7	2008-11-25	Entfernen PDF2XFA. Aktualisierung auf Version 1.4.1. Erweiterung SSL-Teil. Hinweis auf die Codierung des Datenbanknamens	ti

Inhaltsverzeichnis

Zusammenfassung

Dieses Dokument beschreibt die Installation von WASKA (Web-Applikations-Server für Kompetenzagenturen) auf einem Debian GNU/Linux 4.0 (Etch) System.

1 Einführung

WASKA läuft in der Produktiv-Installation auf einem Debian GNU/Linux 4.0 Etch-System. Dieses Handbuch beschreibt die notwendigen Schritte, um von einer Basis-Installation zu einem laufenden System mit Datenbanken und Web-Anwendung zu kommen.

Der erste Abschnitt beschreibt, die Vorbereitung des eingesetzten Datenbankmanagementsystem PostgreSQL für den Betrieb von WASKA.

Für jede Kompetenzagentur wird von WASKA zur Datentrennung eine separate Datenbank mit separaten Nutzern innerhalb eines PostgreSQL-Clusters verwaltet. Die Einrichtung der nötigen Strukturen wird im zweiten Abschnitt beschrieben.

Als Webanwendung benötigt WASKA einen Webserver. WASKA setzt hier den HTTP Server Apache in der Version 2.2 ein. Im dritten Abschnitt wird beschrieben, wie dieser zusammen mit den Modulen mod_ssl und mod_wsgi installiert wird. Ersteres dient der SSL-Verschlüssung, letzteres der Anbindung der Programmiersprache Python, in der die eigentliche Webanwendung geschrieben ist.

Im vierten Abschnitt wird die Einrichtung der eigentlichen Webapplikation WASKA beschrieben, die auf dem Python Web-Framework Pylons in der Version 0.96 beruht.

WASKA kann optional mit einem PDF-Import betrieben werden. Hierfür ist es nötig, zusätzlich eine Java-VM zu installieren und die Applikation auf die Kommunikation mit einem Hintergrundprozess vorzubereiten. Details hierzu sind im fünften Abschnitt beschrieben.

2 Einrichtung des Datenbankmanagement-Systems

Zum Betrieb von WASKA wird das Datenbankmanagementsystem PostgreSQL [1] mindestens in der Version 8.2 benötigt.

WICHTIG
Dies ist auf einem Debian GNU Linux 4.0 (etch) als Backport zu installieren, vgl. http://www.backports.org/dokuwiki/-doku.php?id=instructions
Entsprechend eingerichtet kann PostgreSQL über das Debian-Paketmanagement installiert werden:

```
$ su -
# apt-get install postgresql-8.3 postgresql-plpython-8.3
```

In der produktiven Installation für die Kompetenzagenturen läuft indes PostgreSQL in der Version 8.2. Dies hat für die eigentliche Anwendung keinerlei Belang. In dieser Version wird WAL-Shipping zu Replikationszwecken unterstützt. Die Installation dieser Version kann in Debian Etch durch einen Backport durchgeführt werden. Für WASKA wurden die hierfür nötigen Pakete selbst gebaut.

Nach erfolgreicher Installation des Datenbankservers ist dafür zu sorgen, dass dieser nur von den Rechnern erreicht werden kann, die auf ihn zugreifen müssen. Dies sind im Falle von WASKA nur die Rechner, auf denen die Webserver laufen. Die Client-Authentifikation kann über die pg_hba.conf [2] auf die entsprechenden Rechner eingeschränkt werden.

[1] http://www.postgresql.org
[2] http://www.postgresql.org/docs/8.2/static/auth-pg-hba-conf.html

3 Einrichtung einer Datenbank

WASKA setzt bis hinunter zur Datenbank auf klare Nutzerseparation. Jede Kompetenzagentur hat eine eigene Datenbank. Jeder Nutzer auf dem System hat ein eigenes Datenbank-Konto. Es wird gewährleistet, dass sich jeder Nutzer nur mit der Datenbank seiner Kompetenzagentur verbinden kann. Es gibt zwei Arten von Nutzern, sogenannte Administrations-Nutzer und Case-Manager. Erstere dienen der Verwaltung der Zuständigkeiten innerhalb der jeweiligen Kompetenzagentur; letzere sind für die inhaltliche Arbeit mit den Akten zuständig. Details zu diesem Rollenkonzept finden Sie im Administrations- und im Benutzungshandbuch.

Für die automatische Vergabe von Passworten bei der Einrichtung von Nutzerkonten in den Datenbanken ist das programm pwgen notwendig:

```
$ su -
# apt-get install pwgen
```

Um eine Datenbank mit diesem Rollenkonzept und den Datenstrukturen zur Speicherung der Daten zu erstellen, stellt die Intevation GmbH Installations-Skripte zur Verfügung. Wie diese im System genau zu hinterlegen sind, entnehmen Sie bitte der README-Datei, die dem Installations-Paket beliegt. TODO: Paket auf Server legen und referenzieren!

In einem PostgreSQL-Cluster können mehrere WASKA-Datenbanken betrieben werden. Um einen Cluster auf den Betrieb von WASKA vorzubereiten, muss das SQL-Skript waska-init-cluster.sql ausgeführt werden.

```
$ su - postgres
$ pqsl -f db_setup/waska-init-cluster.sql
```

Nachdem der Cluster vorbereitet ist, können nun die Datenbanken für die Kompetenzagenturen angelegt werden. Hierzu dient das Skript create_databases.sh. Dieses benötigt eine Datei KA_list mit einer Liste von Förderkennziffern der Kompetenzagenturen, für die Datenbanken eingerichtet werden sollen. Exemplarisch kann diese wie folgt aussehen:

```
$ cat KA_list
01jk00815
01jk00816
01jk00817
```

Diese würde zur Einrichtung der Datenbanken ka_01jk00815_db, ka_01jk00816_db und ka_01jk00817_db führen. Die Installation wird dann wie folgt durchgeführt:

```
$ db_setup/bin/create_databases.sh
```

Bei der Installation wird eine Datei result_list angelegt, die jeder der neu erstellten Datenbank ein sogenanntes Adm-Passwort zuordnet. Dies ist das Passwort für den initialen Administrations-Nutzer der Datenbank und muss der verantwortlichen Person der jeweiligen Kompetenzagentur gesichert zugestellt werden.

WICHTIG
Man beachte, dass die Datei result_list Zugangsdaten im Klartext enthält, die nicht in fremde Hände gelangen sollten. Diese Datei ist daher alsbald sicher aus dem Datei-System zu entfernen z.B. mit dem Kommando shred (1). Desweiteren ist darauf zu achten, dass diese Datei nicht ins Backup gerät.

4 Einrichtung des Webservers

Als Web-Server kommt bei WASKA ein Apache 2.2 zu Einsatz. Dieser kann unter Debian 4.0 Etch wie folgt installiert werden.

```
# apt-get install apache2
```

WASKA selbst ist mit dem Web-Framework Pylons [3] in der Version 0.96 entwickelt worden. Um dieses zusammen mit dem Apache-Webserver benutzen zu können, wird das zusätzliche Modul mod_wsgi [4] mindestens in der Version 1.3 benötigt. Auf dem WASKA-Produktiv-System werden hierfür qualitätsgesicherte Backports eingesetzt. Dies geschieht, um die Stabilität des Systems zu gewährleisten.

mod_wsgi kann unter anderem von Debian Backports [5] bezogen werden. Für den produktiven Einsatz wurde mod_wsgi jedoch selbst gebaut.

```
$ wget http://ftp.de.debian.org/debian/pool/main/m/mod-wsgi/mod-wsgi_1.3-1.dsc
$ wget http://ftp.de.debian.org/debian/pool/main/m/mod-wsgi/mod-wsgi_1.3.orig.tar.gz
$ wget http://ftp.de.debian.org/debian/pool/main/m/mod-wsgi/mod-wsgi_1.3-1.diff.gz

$ dpkg-source -x  mod-wsgi_1.3-1.dsc
$ cd mod-wsgi-1.3/
$ dpkg-checkbuilddeps # Angezeigte Abhängigkeiten nachinstallieren
$ fakeroot dpkg-buildpackage
```

Dannach ist ein Debian-Paket libapache2-mod-wsgi_1.3-1_i386.deb entstanden. Das Bauen des Paketes sollte auf einem externen System durchgeführt werden, da es nicht ratsam ist eine Bauumgebung auf dem produktiven System zu installieren. Das Paket kann dann wie folgt auf dem produktiven System installiert werden:

```
# dpkg -i libapache2-mod-wsgi_1.3-1_i386.deb
# a2enmod mod-wsgi
```

Die Pylons-Umgebung lässt sich mit Easy Install [6] installieren. Wir empfehlen allerdings, dies auf einem System ausserhalb des produktiven System zu tun, da bei dieser Installationmethode ggf. Verbindung zu externen Servern aufgebaut werden, um fehlendene Abhängigkeiten nachzuladen. Im produktiven Einsatz in WASKA geschah dies auf dem Staging-System, die eingentlich Installation auf dem Produktiv-System geschieht dann mittels:

```
# cd /tmp
# sh setuptools-0.6c7-py2.4.egg
# tar xfvj Pylons-eggs.tar.bz2
# easy_install -H None -f /tmp/eggs Pylons
```

Wobei die Pakete setuptools-0.6c7-py2.4.egg und Pylons-eggs.tar.bz2 wärend oben genannter Methode entstanden sind und auf das produktive System kopiert wurden. Zugriff auf weitere Server findet so nicht mehr statt.

4.1 Einrichtung SSL Verbindung

Der produktive WASKA-Webserver ist exklusiv über das HTTPS-Protokoll erreichbar. Unverschlüsselte HTTP-Verbindungen werden nicht zugelassen. Ferner ist es zwingend erforderlich, sich mit einem gültigen klienten-seitigen Zertifikat am Server anzumelden, dass von einer eingetragenen CA ausgestellt wurde.

[3] http://pylonshq.com/
[4] http://code.google.com/p/modwsgi/
[5] http://www.backports.org/
[6] http://peak.telecommunity.com/DevCenter/EasyInstall

Hierfür ist es zunächst nötig, in der Konfigurationsdatei /etc/apache2/httpd.conf den Eintrag 'Listen 80' durch 'Listen 443' zu ersetzen. Im der Konfiguartionsdatei des virtuellen Hosts (/etc/apache2/sites-available/default) ist folgendes hinzuzufügen:

```
SSLEngine On
SSLCipherSuite HIGH:MEDIUM
SSLCertificateFile    /etc/apache2/ssl/server.crt
SSLCertificateKeyFile /etc/apache2/ssl/server.key
SSLCACertificateFile  /etc/apache2/ssl/cacert.pem
SSLVerifyClient       require
SSLVerifyDepth        3
SSLOptions            StdEnvVars
```

Dies erzwingt die Nutzung von SSL und Client-Zertifikaten. Es werden zudem nur bessere Cipher-Algorithmen unterstützt. SSLCertificate(File|KeyFile) sind entsprechend zu erzeugen und zu signieren. SSLCACertificateFile ist eine konkatenierte Liste von CA-PEMs, die für die Überprüfung benutzt werden sollen. In der produktiven WASKA-Installation werden nur Client-Zertifikate der Zertifizierungsstelle der KDVZ [7] angenommen. Die Überprüfungskette hin zur Wurzel wird auf 3 beschänkt. SSLOptions StdEnvVars sorgt dafür, dass in CGI- und WSGI-Anwendungen im Environment detailierte Informationen über das hereinkommende Zertifikat zur Verfügung stehen. Eine detailiere Beschreibung dieser Parameter finden Sie unter [8].

Inhalte des Zertifikates können auch direkt durch den apache ausgewertet werden, dies ist in die Wurzel-Location einzutragen (ebenfalls /etc/apache2/sites-available/default):

```
<Location />
    SSLRequire %{SSL_CLIENT_S_DN_OU} eq "WASKA" \
    and %{SSL_CLIENT_VERIFY} eq "SUCCESS"
</Location>
```

Dies erzwingt, nur Anfragen von Klienten zu beantworten, die ein Zertifikat mit Eintrag WASKA im SSL_CLIENT_S_DN_OU vorweisen können und die erfolgreich validiert wurden.

WICHTIG
Über das Feld *SSL_CLIENT_S_DN_CN* im Client-Zertifikat wird festgelegt, mit welcher Datenbank sich der Nutzer verbinden wird. Die Angaben in dem Zertifikat sind dabei in der Form *GRP: KA FKZ TESTDB* anzugeben. "TESTDB"gibt hier den Namen der Datenbank an, die übrigen Werte können beliebig getauscht werden, solange die Trennung bestehen bleibt.

Abschliessend muss für eine funktionierende SSL-Verschlüsselung das SSL-Modul mod_ssl aktiviert werden.

```
# a2enmod ssl
# /etc/init.d/apache2 restart
```

Die Zertifikate werden anhand von Sperrlisten auf ihre Gültigkeit überprüft. Hier muss ein regelmässiger Abgleich mit den Sperrlisten (revocation lists) der Zertifizierungsstelle (Certificate Authority) stattfinden.

5 Einrichtung der Webanwendung

WASKA sollte als exklusive Anwendung auf dem Web-Server laufen. Um dies zu gewährleisten, ist der Skriptpfad in der Konfigurationsdatei des virtuellen Hosts (/etc/apache2/sites-available/default) zu beschränken:

[7] http://cas.kdvz.de
[8] http://httpd.apache.org/docs/2.2/mod/mod_ssl.html

```
WSGIScriptAlias / /usr/local/wsgi/scripts/waska.wsgi
```

Weitere Einstellungen zur Optimierung des Web-Servers finden Sie in der README-Datei im Installationspaket.

Die Anwendung selbst wird von der Intevation GmbH als Tar-Archiv ausgeliefert. Dieses muss an geeigneter Stelle im Dateibaum entpackt werden. Auf dem produktiven System ist dies das Verzeichnis /usr/local/wsgi/scripts/.

```
# mkdir -p /usr/local/wsgi/scripts/
# cd /usr/local/wsgi/scripts/
# tar xfvz waska_1_4_1.tar.gz
# chown -R root:www-data waska_1_4_1
# find waska_1_4_1 -type d -print0 | xargs -0 chmod 750
# find waska_1_4_1 -type f -print0 | xargs -0 chmod 640
# ln -s waska_1_4_1 waskaweb
```

Die Applikation selbst wird über die Datei waskaweb/production_wsgi.ini konfiguriert. Hier ist es vor allem wichtig, dass der Rechner und der Port angeben werden, auf dem der Datenbank-Server läuft:

```
...
db_host = 127.0.0.1 # IP-Adresse des Datenbank-Servers
db_port = 5432
...
```

Weitere optimierende Einstellungen entnehmen Sie auch hier bitte der README-Datei des Installationspaketes. Nach den Einstellungen kann der Web-Server neu gestartet werden.

6 Installation einer Test-/Entwicklerversion

Die Installation von mpuls WASKA für den produktiven Einsatz hängt von den Rahmenbedingungen bzgl. Infrastruktur und Anforderungen an Datensicherheit und Datenschutz ab. Eine allgemeine Empfehlung kann nicht gegeben werden, der Betrieb erfordert fortgeschrittene Kenntnisse über Administration der eingesetzten Komponenten.

Im Folgenden wird die Installation von mpuls WASKA auf einem Debian GNU/Linux 4.0 (etch) für Tests und Entwicklungen dargestellt.

WARNUNG
Die dargestellte Installation ist *nicht* für den produktiven Einsatz mit realen personenbezogenen Daten geeignet! Einige Sicherheitskonzepte sind abgeschwächt, der produktive Einsatz erfordert fortgeschrittene Kenntnisse über Administration der eingesetzten Komponenten!

Mit "vim [Dateiname]ßind jeweils zu bearbeitende Dateien benannt, die Änderungen sind mit "+"(hinzugefügt) bzw. "(entfernt) gekennzeichnet.

```
#
# Download notwendiger Pakete von http://mpuls.wald.intevation.org
# siehe "Released Files"
#
cd /tmp
wget http://wald.intevation.org/frs/download.php/490/mpuls-waska-1.4.1.tar.bz2
tar -jxvf mpuls-waska-1.4.1.tar.bz2
```

```
#
# Installation von PostgreSQL 8.3 aus Debian Backports:
#
vim /etc/apt/sources.list
  + deb http://www.backports.org/debian etch-backports main contrib non-free
apt-get install -t etch-backports postgresql-8.3 postgresql-plpython-8.3

apt-get install python-psycopg2 pwgen

#
# Installation des Apache Web-Servers und Einrichtung der Zertifikate
# (vgl. unten). Der Server akzeptiert _JEDES_ Zertifikat, das im
# Subject OU=WASKA enthält!
#
apt-get install apache2 libapache2-mod-wsgi
mkdir /etc/apache2/ssl
cp /tmp/server.* /etc/apache2/ssl/
vim /etc/apache2/sites-available/default
  +        SSLEngine On
  +        SSLVerifyClient       optional_no_ca
  +        SSLCipherSuite HIGH:MEDIUM
  +        SSLCertificateFile    /etc/apache2/ssl/server.crt
  +        SSLCertificateKeyFile /etc/apache2/ssl/server.key
  +        SSLOptions StdEnvVars
  +        SSLSessionCacheTimeout 1800
  +
  +        <Location />
  +                SSLRequire %{SSL_CLIENT_S_DN_OU} eq "WASKA"
  +        </Location>
  +
  +        WSGIScriptAlias / /usr/local/wsgi/scripts/waska.wsgi
  +        WSGIDaemonProcess waska processes=1 threads=25
  +        WSGIProcessGroup waska
a2enmod ssl
vim /etc/apache2/ports.conf
  -Listen 80
  +Listen 4444

#
# Installation von Pylons und weiteren Paketen
#
apt-get install python-xml python-excelerator
cd /tmp/mpuls-waska-1.4.1
wget http://peak.telecommunity.com/dist/ez_setup.py
python2.4 ez_setup.py
tar xfvj Pylons-eggs.tar.bz2
easy_install -H None -f /tmp/mpuls-waska-1.4.1/eggs Pylons

#
# Einrichtung der Datenbank(en)
#
# ALS BENUTZER postgres
cd /tmp/
tar zxvf mpuls-waska-1.4.1/db_setup-1.4.1.tar.gz
cd /tmp/db_setup
psql -f waska-init-cluster.sql
vim KA_list
  +01jk00816
bin/create_databases.sh

#
```

```
# Installation der Anwendung
#
mkdir -p /usr/local/wsgi/scripts
cd /usr/local/wsgi/scripts
tar zxvf /tmp/waskaweb_1.4.1.tar.gz
chown -R root:www-data waskaweb_1_4_1
find waskaweb_1_4_1 -type d -print0 | xargs -0 chmod 750
find waskaweb_1_4_1 -type f -print0 | xargs -0 chmod 640
ln -s waskaweb_1_4_1 waskaweb
cp /tmp/waska.wsgi .
vim waskaweb/production_wsgi.ini
  -db_host     = 192.168.11.17
  -db_port     = 5434
  +db_host     = 127.0.0.1
  +db_port     = 5433
  +evaluation_server = 0
chgrp www-data /usr/local/wsgi/scripts/waska.wsgi
mkdir /usr/local/wsgi/waska_data
chown root:www-data /usr/local/wsgi/waska_data
chmod 770 /usr/local/wsgi/waska_data
```

Für den Betrieb von WASKA (auch für Tests und Entwicklung) sind Zertifikate notwendig. Allerdings sind hier Zertifikate mit einer vollständigen PKI (Public-Key-Infrastruktur) mit Zertifizierungsstelle (CA, Trustcenter) zu äufwändig, daher stellen wir im Folgenden die Generierung selbst-signierter Zertifikate vor:

```
#
# Zertifikate erstellt mit OpenSSL:
apt-get install openssl

  # Server
  cd /etc/apache2/ssl/
  openssl req -new -x509 \
    -keyout server.key \
    -out server.crt \
    -days 365 -nodes

  # Client - Die Datei client.p12 ist in die Zertifikatsspeicher der jeweiligen
  #          Web-Browser zu importieren.
  openssl req -new -x509 \
   -keyout client.pem \
   -out client.pem \
   -subj '/C=DE/ST=./O=mpuls WASKA Demo/OU=WASKA/OU=WASKA Demo 00816/CN=GRP: KA FKZ 01 ↩
       jk00816/emailAddress=verantwortlich@example.mail' \
   -days 365 -nodes

  openssl pkcs12 \
    -export -clcerts \
    -in client.pem \
    -out client.p12
```