Platform installation documentation

Documentation of Open Data sharing platform prototype
Platform installation

Jarosław Legierski
Henryk Rosa
Tomasz Janisiewicz

Flickr.com, jbozanowski, CC BY

The project is cofinanced by the National Centre for Research and Development within the framework of Social Innovations Program.
Table of contents

1    Aim of the project .......................................................... 3
2    Hardware and software requirements ........................................ 3
3    Platform installation documentation .......................................... 5
   3.1   Data Server (CKAN) Installation ....................................... 5
   3.2   Install Function Server (Java) ......................................... 25
   3.3   Install CKAN plugins.................................................... 32
4    Glossary – list of abbreviations ............................................. 36
5    References ......................................................................... 38
1 Aim of the project

The aim of the MUNDO [1] project is to diagnose the possibilities of exposure and to prepare data which can be shared by the City of Warsaw in the Open Data model, and first and foremost to build the exposure layer for the data in the form of an API exposure platform. The MUNDO project combines developing technologies for open data sharing with social methodology concerning the inclusion of various social groups in the activities aimed at improving the city life. The consortium, as part of its works, conducts activities to prepare the technological compound of the project's social part. Meanwhile, in the pilot part, the consortium will conduct a competition for web applications for Warsaw in order to convince the society that open data (and particularly public data) is essential for the development of the city and facilitation of its inhabitants’ functioning.

The innovation of the MUNDO project is its complexity and combination of technological framework with the methodology of social participation in the urban life. Another asset of the project is the possibility to replicate it in other Polish cities, as both open data exposure platform software and the methodology of conducting social activities will be shared for use by interested entities under free licenses (GPL, LGPL, FDL, CC).

2 Hardware and software requirements

The platform has the following components:

- Proxy Server (optional) – security layer responsible, among others, for traffic distribution
- Data Server – a server, based on CKAN, constituting: data catalog, web server and a repository of file and table data
- Function Server – server based on Jboss/Java, constituting middleware for API calls to data sources based on Web Services, database, etc. This system must ensure access to dynamic data, i.e. the data whose source are network services and databases. This subsystem is also responsible for limiting calls, buffering data, converting calls and converting data serialization in order to make the API format as coherent as possible for all data and functions of the Warsaw City Hall.
The following minimal requirements for platform installation were defined:

<table>
<thead>
<tr>
<th>Server</th>
<th>CPU</th>
<th>RAM</th>
<th>HD</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proxy server</td>
<td>2 cores</td>
<td>4 GB</td>
<td>40 GB</td>
<td>public IP address</td>
</tr>
<tr>
<td>Data server</td>
<td>2 cores</td>
<td>4 GB</td>
<td>40 GB + 100 GB with the option of expansion to 1 TB for data</td>
<td>In case of a lack of proxy server, public IP address</td>
</tr>
<tr>
<td>Function server</td>
<td>2 cores</td>
<td>4 GB</td>
<td>40 GB</td>
<td></td>
</tr>
</tbody>
</table>

And recommended hardware requirements:

<table>
<thead>
<tr>
<th>Server</th>
<th>CPU</th>
<th>RAM</th>
<th>HD</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proxy server</td>
<td>4 cores</td>
<td>8 GB</td>
<td>80 GB</td>
<td>public IP address</td>
</tr>
</tbody>
</table>
Tested operational systems: CentOS 6.6, CentOS 7.0, Ubuntu 12.4. Note: this documentation is dedicated to platform installation based on CentOS 7.0

## 3 Platform installation documentation

The following chapter presents the procedure of MUNDO platform installation, including a description of both Data Server and Function Server installation.

### 3.1 Data Server (CKAN) Installation

ISO used for VM installation

[http://isoredirect.centos.org/centos/7/isos/x86_64/CentOS-7.0-1406-x86_64-DVD.iso](http://isoredirect.centos.org/centos/7/isos/x86_64/CentOS-7.0-1406-x86_64-DVD.iso)

This document was created on the basis of documentation:  

OS version verification

```bash
cat /etc/*release*
```

#### 1. Installation of the required packages

Installation and activation of CentOS Continuous Release (CR) Repository

```bash
yum install centos-release-cr
```

*note – yum did not install any additional packages

*No package centos-release-cr available. Error: Nothing to do*

Description of CentOS Continuous Release (CR) Repository  

Update and reboot the system

```bash
yum update
shutdown -r now
```

Install wget and policycoreutils-python, which will be required later.

```bash
yum install wget policycoreutils-python
```

Additionally, install midnight commander mc (facultative)

```bash
yum install mc
```
Install and activate Extra Packages for Enterprise Linux (EPEL) Repository

```
rpm -Uvh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

Install the required packages

```
yum install xml-commons git subversion mercurial postgresql-server postgresql-devel postgresql python-devel libxml2 libxml2-devel python-virtualenv gcc gcc-c++ make java-1.6.0-openjdk-devel java-1.6.0-openjdk tomcat6 xalan-j2 unzip policycoreutils-python mod_wsgi httpd
```

Note: in CentOS 7.0 repositories, there is no package called **tomcat6** – installation of tomcat is described in the subsequent chapters of this document.

### 2. CKAN Installation

Create a CKAN user. The ckan user is created with a shell of /sbin/nologin and a home directory /usr/lib/ckan on the basis of CKAN Deployment documentation [http://docs.ckan.org/en/ckan-2.0/install-from-source.html](http://docs.ckan.org/en/ckan-2.0/install-from-source.html)

```
useradd -m -s /sbin/nologin -d /usr/lib/ckan -c "CKAN User" ckan
```

The newly created directory /usr/lib/ckan should have appropriate read properties set so that the content can be hosted via httpd.

```
chmod 755 /usr/lib/ckan
```

Modify default properties for the newly created directory so that they can be hosted via httpd.

```
semanage fcontext --add --type httpd_sys_content_t "/usr/local/ckan(/.*)?"
```

```
restorecon -vR /usr/lib/ckan
```

Switch to the ckan user using su.

```
su -s /bin/bash - ckan
```

Install a dedicated, isolated Python environment.

```
virtualenv --no-site-packages default
```

Activate the newly installed Python environment.

```
. default/bin/activate
```

Download and install version 2.2.1 of CKAN in the dedicated, isolated Python environment.
pip install --ignore-installed -e 'git+https://github.com/ckan/ckan.git@ckan-2.2.1#egg=ckan'

Note: option **--ignore-installed** isn't compulsory, but may be useful if an earlier installation of CKAN exists.

```bash
pip install -r /usr/lib/ckan/default/src/ckan/requirements.txt
```

Return to the root user session using exit or pressing Ctrl + D
3. Configure PostgreSQL database

Enable PostgreSQL to start at system boot

`chkconfig postgresql on`

Initialize PostgreSQL database

`service postgresql initdb`

Edit `/var/lib/pgsql/data/pg_hba.conf` so that postgres can be accessed through the localhost interface and possibly through the network from other hosts

```
local  all   postgres         ident
local  all   all             trust
# IPv4 local connections:
host  all   all   127.0.0.1/32   md5
# IPv6 local connections:
host  all   all   ::1/128     md5
```

Note: for access from PostgressAdmin, add the network of host on which the PostgressAdmin application is installed

```
# IPv4 local connections:
host  all   all   10.10.10.0/24   trust
```

Edit `postgresql.conf` and set the address and port on which PostgreSQL will run

```
listen_addresses = '*'           # what IP address(es) to listen on; port = 5432  # (change requires restart)
```

Start PostgreSQL

`service postgresql start`

Switch to postgres user

`su - postgres`
List existing databases:

```bash
psql -l
```

Check that the encoding of characters in databases is UTF8; if database internationalisation is not set, it may cause problems when installing CKAN. Changing the encoding of PostgreSQL may mean deleting existing databases!

Create a new PostgreSQL user called `ckan_default` and set their password

```bash
createuser -S -D -R -P ckan_default
```

```
password
```

Create a new database in PostgreSQL called `ckan_default`

```bash
createdb -O ckan_default ckan_default -E utf-8
```

Exit postgres user session by pressing Ctrl + D or exit

**4. Modify CKAN configuration**

On the root user account, create a directory to contain the CKAN files:

```bash
mkdir -p /etc/ckan/default
```

```bash
chown -R ckan /etc/ckan/
```

Switch to ckan user and create a configuration file for CKAN:

```bash
su -s /bin/bash - ckan
```

```
. default/bin/activate
```

```
 cd /usr/lib/ckan/default/src/ckan
```

```bash
paster make-config ckan /etc/ckan/default/production.ini
```

Edit the `development.ini` file and change the following settings:

```ini
sqlalchemy.url = postgresql://ckan_default:pass@127.0.0.1/ckan_default
ckan.site_id = default
```

Set access to SOLR (for SOLR installed on Jetty - jetty-solr)

```ini
solr_url = http://127.0.0.1:8983
```

Set access to SOLR (for SOLR hosted on tomcat6 server)
solr_url = http://127.0.0.1:8080/solr/ckan-schema-2.0

Log out of ckan session by pressing Ctrl + D or exit.

5. Install and configure Apache SOLR

Install Java JDK

yum install java-1.6.0-openjdk-devel
yum install java-1.6.0-openjdk

Verify the installed version of Java environment

```
java -version
```

```
$ java -version
java version "1.7.0_25" Java(TM) SE Runtime Environment (build 1.7.0_25-b02) Java HotSpot(TM) 64-Bit Server VM (build 23.7-b01, mixed mode)
```

```
alternatives --config java
```

If the default Java version is different than 1.6, set 1.6

yum install net-tools

6. Install Tomcat6

Download Tomcat6

```
 cd /opt


 Extract

 tar zxfv apache-tomcat-6.0.41.tar.gz

 Move the content to tomcat directory

 groupadd tomcat

 useradd -g tomcat -d /opt/tomcat tomcat chown -R
tomcat.tomcat /opt/tomcat

 cd /opt
```
mv apache-tomcat-6.0.41/* tomcat

Note: tomcat user should have read rights to tomcat directory contents (to some files (logs write +w))

chown -R tomcat /opt/tomcat/
chown -R tomcat /opt/tomcat/*

Set environment variables by creating and editing the tomcat file in /etc/profile.d/

vi /etc/profile.d/tomcat

File contents:

#!/bin/bash
CATALINA_HOME=/opt/tomcat
PATH=$CATALINA_HOME/bin:$PATH
export PATH CATALINA_HOME
export CLASSPATH=.  

Set permissions for tomcat.sh
chmod +x /etc/profile.d/tomcat

Change the environment variables by executing
source /etc/profile.d/tomcat

Set permissions to scripts controlling the tomcat service

chmod +x /opt/tomcat/bin/startup.sh
chmod +x /opt/tomcat/bin/shutdown.sh
chmod +x /opt/tomcat/bin/catalina.sh

Start Tomcat server:

$CATALINA_HOME/bin/startup.sh

Check if tomcat server runs on port 8080

netstat -an | grep 8080

Set firewall rules in CentOS 7.0 system:
firewall-cmd --zone=public --add-port=8080/tcp --permanent firewall-cmd --reload

Configure tomcat admin and user accounts

Edit file conf/tomcat-users.xml and put users' account data between the tags <tomcat-users> </tomcat-users>

vi $CATALINA_HOME/conf/tomcat-users.xml

[...]
<!--
NOTE: The sample user and role entries below are wrapped in a comment and thus are ignored when reading this file. Do not forget to remove <!... ...> that surrounds them.
-->
<!--
<role rolename="tomcat"/>
<role rolename="role1"/>
</role>
<user username="tomcat" password="tomcat" roles="tomcat"/>
<user username="both" password="tomcat" roles="tomcat,role1"/>
<user username="role1" password="tomcat" roles="role1"/>
</user>
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<role rolename="admin-gui"/>
<role rolename="admin-script"/>
<user username="admin" password="admin" roles=""manager-gui,manager-script,manager-jmx,manager-status,admin-gui,admin-script"">
</user>
</tomcat-users>

In the configuration above, username=admin and password=admin were set

Restart tomcat server

$CATALINA_HOME/bin/shutdown.sh
$CATALINA_HOME/bin/startup.sh

Create and edit file /etc/rc.d/init.d/tomcat
#!/bin/bash

#chkconfig: 2345 95 20
#description: tomcat
#processname: tomcat

CATALINA_HOME=/opt/tomcat
PATH=$CATALINA_HOME/bin:$PATH
export PATH CATALINA_HOME
export CLASSPATH=

case $1 in
    start)
        /bin/su tomcat $CATALINA_HOME/bin/startup.sh
        ;;
    stop)
        /bin/su tomcat $CATALINA_HOME/bin/shutdown.sh
        ;;
    restart)
        /bin/su tomcat $CATALINA_HOME/bin/shutdown.sh
        /bin/su tomcat $CATALINA_HOME/bin/startup.sh
        ;;
    status)
        netstat -an | grep 8080
        ;;
esac

exit 0

Edit sudoers file and add the following line at the end

tomcat ALL=/opt/tomcat/bin/shutdown.sh, /opt/tomcat/bin/startup.sh

Set autostart for Tomcat service
chkconfig --add tomcat
chkconfig --level 234 tomcat on
chkconfig --list tomcat
tomcat 0:off 1:off 2:on 3:on 4:on 5:off 6:off

Verify – check logs
less /opt/tomcat/logs/catalina.out
7. Install SOLR

Note: CKAN does not use the newest version of Apache SOLR and requires version 1.4.1 of SOLR

Download and extract Apache SOLR

curl http://archive.apache.org/dist/lucene/solr/1.4.1/apache-solr-1.4.1.tgz | tar xzf -

Create directories to contain multiple SOLR cores.

curl http://archive.apache.org/dist/lucene/solr/1.4.1/apache-solr-1.4.1.tgz | tar xzf -

mkdir -p /usr/share/solr/core0 /usr/share/solr/core1 /var/lib/solr/data/core0 /var/lib/solr/data/core1 /etc/solr/core0 /etc/solr/core1

Copy Apache SOLR war file to the desired location

cp apache-solr-1.4.1/dist/apache-solr-1.4.1.war /usr/share/solr

Copy the example Apache SOLR configuration to the core0 directory.

cp -r apache-solr-1.4.1/example/solr/conf /etc/solr/core0

Edit configuration file, /etc/solr/core0/conf/solrconfig.xml, by modifying the line:

<dataDir>${dataDir}</dataDir>

Copy core0 configuration to core1.

cp -r /etc/solr/core0/conf /etc/solr/core1

Create a symbolic link between the configurations in /etc and /usr

ln -s /etc/solr/core0/conf /usr/share/solr/core0/conf
ln -s /etc/solr/core1/conf /usr/share/solr/core1/conf

Remove the provided schema from the two configured cores and link the schema files in the CKAN source.

rm -f /etc/solr/core0/conf/schema.xml

ln -s /usr/lib/ckan/default/src/ckan/ckan/config/solr/schema-2.0.xml /etc/solr/core0/conf/schema.xml
rm -f /etc/solr/core1/conf/schema.xml
ln -s /usr/lib/ckan/default/src/ckan/ckan/config/solr/schema-1.4.xml /etc/solr/core1/conf/schema.xml
Create a new file `/opt/tomcat/conf/Catalina/localhost/solr.xml`, with the following contents:

```xml
<Context docBase="/usr/share/solr/apache-solr-1.4.1.war" privileged="true" allowLinking="true" crossContext="true">
    <Environment name="solr/home" type="java.lang.String" value="/usr/share/solr" override="true" />
</Context>
```

Create a new file `/usr/share/solr/solr.xml`, with the following contents:

```xml
<solr persistent="true" sharedLib="lib">
    <cores adminPath="/admin/cores">
        <core name="ckan-schema-2.0" instanceDir="core0"> <property name="dataDir" value="/var/lib/solr/data/core0" /></core>
        <core name="ckan-schema-1.4" instanceDir="core1"> <property name="dataDir" value="/var/lib/solr/data/core1" /></core>
    </cores>
</solr>
```

Make tomcat user the owner of the solr directory.

```
chown -R tomcat:tomcat /usr/share/solr /var/lib/solr
```

8. Create tables in the CKAN database

Log to ckan user, activate the isolated Python environment, and change directory to default/src/ckan.

```
su -s /bin/bash - ckan
. default/bin/activate
cd default/src/ckan
```

Initialise the CKAN platform database.

```
paster db init -c /etc/ckan/default/production.ini
```

You should see Initialising DB: SUCCESS.

9. Link to who.ini

Activate Python virtual environment

```
su -s /bin/bash - ckan
```
. default/bin/activate

cd default/src/ckan

who.ini (the Repoze.who configuration file) must be accessible in the same directory as CKAN configuration file, so create a symbolic link to it:

```
ln -s /usr/lib/ckan/default/src/ckan/who.ini /etc/ckan/default/who.ini
```

10. Create a WSGI file

Create WSGI script in /etc/ckan/default/apache.wsgi. apache.wsgi file should have the following contents.

```python
import os

activate_this = os.path.join('/usr/lib/ckan/default/bin/activate_this.py')
execfile(activate_this, dict(file =activate_this))

from paste.deploy import loadapp

config_filepath = os.path.join(os.path.dirname(os.path.abspath(file__)),
              'production.ini')

from paste.script.util.logging_config import fileConfig
fileConfig(config_filepath)

application = loadapp('config:%s' % config_filepath)
```

The modwsgi Apache module will redirect requests to your web server to this WSGI script file. The script file then handles those requests by directing them to your CKAN instance.

Log out of ckan user session by pressing Ctrl + D or exit.
11. Create configuration file for Apache server

Create a configuration file for Apache server `/etc/httpd/conf.d/ckan_default.conf`, with the following contents

```
WSGISocketPrefix /var/run/wsgi

<VirtualHost 0.0.0.0:80>
    ServerName 10.10.10.10
    ServerAlias www.default.ckanhosted.com
    WSGIScriptAlias / /etc/ckan/default/apache.wsgi

    # Pass authorization info on (needed for rest api).
    WSGIPassAuthorization On

    # Deploy as a daemon (avoids conflicts between CKAN instances).
    WSGIDaemonProcess ckan_default display-name=ckan_default processes=2 threads=15

    WSGIProcessGroup ckan_default

    ErrorLog /var/log/httpd/ckan_default.error.log
    CustomLog /var/log/httpd/ckan_default.custom.log combined

    <Directory /etc/ckan/default>
        Require all granted
    </Directory>

    <Directory /usr/lib/ckan/default/bin/>
        Require all granted
    </Directory>

</VirtualHost>
```

Modify ServerName and ServerAlias

12. Configure Apache server

Enable httpd to make network connections

```
setsebool -P httpd_can_network_connect 1
```

Enable httpd to start on system boot

```
chkconfig httpd on
```
Start httpd

service httpd start

12. Configure iptables

Enable rule to open port 80.

`firewall-cmd --zone=public --add-port=80/tcp --permanent firewall-cmd --reload`

13. Connect to CKAN web interface

Start browser and type domain name or server’s IP address – access to CKAN website should be enabled.

14. Activate Filestore

Create a directory in which CKAN will store files uploaded to the platform.

`mkdir -p /var/lib/ckan/default`

Add the following line to the CKAN configuration file (production.ini) in the section [app:main]

`ckan.storage_path = /var/lib/ckan/default`

Set permissions for the `ckan.storage_path` directory. For example: for Apache, the user on which httpd demon is running (apache user for CentOS 7.0) must have read, write and execute permissions.

`chown apache /var/lib/ckan/default`

`chmod u+rwx /var/lib/ckan/default`

Reload web server

`service httpd reload`

Filestore test using API is performed by calling CKAN API using curl

`curl -H Authorization: your-api-key' 'http://yourhost/api/action/resource_create' --form upload=@filetoupload --form package_id=my_dataset`

For example, add resource for the dataset with package_id=85ee85fd-09ef-404d-9edf-f2aace7e2420

Note: the contents of package_id for the given data set can be found in the resource’s URL, e.g.: `http://10.10.10.10/dataset/package_id/resource/resource_id/dscn0001.jpg`
curl -H 'Authorization: cde10f4f-2d84-407c-86d7-b0281eb8723b'
http://10.10.10.10/api/action/resource_create --form
upload=@Miasta.jpg --form package_id=85ee85fd-09ef-404d-9edf-f2aace7e2420

In order to overwrite a file with its new version, use resource_update() function

15. Activate DataStore

Edit CKAN configuration file production.ini and add datastore plugin to the ckan.plugins list

ckan.plugins = datastore

Configure database for datastore

Switch to postgres user

su - postgres

List existing databases:

psql -l

Check that the encoding of characters in databases is UTF8; if database internationalisation is not activated, it may cause problems. Changing the encoding of PostgreSQL may mean deleting existing databases!

Create user and database for datastore

Create a datastore_default user

createuser -S -D -R -P -l datastore_default

Password

Create database (owner: ckan_default) with the example name datastore_default:

createdb -O ckan_default datastore_default -E utf-8

Give permissions to datastore_default

Edit set_permissions.sql (/usr/lib/ckan/default/src/ckan/ckanext/datastore/bin)

-- name of the main CKAN database
\set maindb ckan_default

-- the name of the datastore database
\set datastore\_db datastore\_default

-- username of the ckan postgres user
\set ckanuser ckan\_default

-- username of the datastore user that can write
\set wuser ckan\_default

-- username of the datastore user who has only read permissions
\set rouser datastore\_default

Start set\_permissions.sql script

psql postgres -f /usr/lib/ckan/default/src/ckan/ckanext/datastore/bin/set\_permissions\.sql

Verify – list the existing databases

psql -l

Edit CKAN configuration file (production.ini), set users who have access to datastore and their passwords.

/etc/ckan/default/production.ini

ckan.datastore.write_url =
    postgresql://ckan\_default:pass@localhost/datastore\_default
ckan.datastore.read_url =
    postgresql://datastore\_default:pass@localhost/datastore\_default

Activate connections from other hosts with the postgresql database – modify pg\_hba\.conf file by adding

/var/lib/pgsql/data/pg\_hba.conf

host all all 10.10.0/24 trust

Restart postgresql service

service postgresql restart

Open 5432 for firewall

firewall\-cmd --zone=public --add-port=5432/tcp --permanent
firewall\-cmd --reload

Test datastore by CKAN JSON API

curl -X GET "http://127.0.0.1/api/3/action/datastore\_search?resource\_id=_table\_metadata"

The result of this function should be json returned without errors

Add resource to the datastore via JSON CKAN API
curl -X POST http://127.0.0.1/api/3/action/datastore_create -H "Authorization: {YOUR-API-KEY}" -d "{"resource":{"package_id":"{PACKAGE-ID}"}, "fields": [{"id": "a"}, {"id": "b"}], "records": [{"a": 1, "b": "xyz"}, {"a": 2, "b": "zzz"}]}

Dataset address (package_id to which data is added is in bold)

http://10.10.10.10/dataset/2280bc40-1e6b-4256-be57-8b5e471aa4a3/resource/c9c09817-3a08-4662-b647-b15d72f9271c/download/file.txt

curl -X POST http://10.72.1.80/api/3/action/datastore_create -H "Authorization: c5203da7-e1f1-40f9-89de-358de6cbb6d4" -d "{"resource":{"package_id":"128a74d4-84ec-4cee-83d3-5147f88f7245"}, "fields": [{"id": "a"}, {"id": "b"}], "records": [{"a": 1, "b": "xyz"}, {"a": 2, "b": "zzz"}]}

Executing the function above should result in adding another resource (table) to the dataset

16. Install and configure DataPusher – automatically add data to DataStore

Install DataPusher plugin
Create a virtual Python environment (virtualenv) for datapusher
virtualenv /usr/lib/ckan/datapusher

Create a directory for datapusher sources
mkdir /usr/lib/ckan/datapusher/src
cd /usr/lib/ckan/datapusher/src

Clone datapusher sources from github
git clone -b stable https://github.com/ckan/datapusher.git

Install DataPusher
cd datapusher
/usr/lib/ckan/datapusher/bin/pip install -r requirements.txt
/usr/lib/ckan/datapusher/bin/python setup.py develop

Copy configuration file for Apache server
cp deployment/datapusher/etc/httpd/conf.d/datapusher.conf
Modify datapusher.conf file – log location and directory permissions, among others

```
ErrorLog /var/log/httpd/datapusher.error.log
CustomLog /var/log/httpd/datapusher.custom.log combined

<Directory /etc/ckan/>
Require all granted
</Directory>

<Directory /usr/lib/ckan/default/bin/>
Require all granted
</Directory>
```

Open 8800 port for the httpd service and in the firewall settings

```
semanage port -a -t http_port_t -p tcp 8800
```

```
firewall-cmd --zone=public --add-port=8800/tcp --permanent
firewall-cmd --reload
```

Copy wsgi file for DataPusher

```
cp deployment/datapusher.wsgi /etc/ckan/
```

Copy standard settings for DataPusher

```
cp deployment/datapusher_settings.py /etc/ckan/ Activate VirtualHost in httpd.conf file
```

Add and modify the following lines in httpd.conf file

```
Listen 80
Listen 8800
```

And add at the end of httpd.conf file

```
<VirtualHost *:80>
    ServerAdmin admin@linuxdrops.com
    ServerName ckan

    ProxyRequests Off
    <Proxy *>
        Order deny,allow
        Allow from all
    </Proxy>
    ProxyPass / http://localhost:8080/
    ProxyPassReverse / http://localhost:8080/
</VirtualHost>
```

```
<Location />
    Order allow,deny
    Allow from all
</Location>
```
<VirtualHost *:8800>
  ServerAdmin admin@linuxdrops1.com
  ServerName datapusher

  ProxyRequests Off
  
  <Proxy *>
  Order deny,allow
  Allow from all
  </Proxy>

  ProxyPass / http://localhost:8080/
  ProxyPassReverse / http://localhost:8080/

  ErrorLog logs/datapusher_error_log
  CustomLog logs/datapusher-access_log common
</VirtualHost>

WSGISocketPrefix /var/run/wsgi

Test DataPusher – in a browser, connect with CKAN host on port 8800
http://10.10.10.10:8800/
As a result, the following JSON should be returned

```json
{
  "help": "Get help at:
  http://ckan-service-provider.readthedocs.org/.
}
```

Add or modify the following lines in the CKAN configuration file development.ini
ckan.datapusher.url = http://10.10.10.10:8800/
ckan.site_url = http://10.10.10.10
ckan.plugins = <other plugins> datapusher
3.2 Install Function Server (Java)

This chapter describes the installation of Function Server system components

1. Install Java 7

```bash
sudo apt-get install oracle-java7-installer
```

2. Install JBOSS 7.1.1

Download jboss 7.1.1
```bash
```

Extract jboss 7.1.1
```bash
sudo tar xfvz jboss-as-7.1.1.Final.tar.gz
```

Copy to directory
```bash
sudo mv jboss-as-7.1.1.Final /usr/local/share/jboss
```

Add jboss user
```bash
sudo adduser jboss
```

Change permissions for jboss directory
```bash
sudo chown -R jboss /usr/local/share/jboss
sudo chgrp -R jboss /usr/local/share/jboss
```

3. Add user-administrator using Management Console

```bash
su - jboss
cd /usr/local/share/jboss/bin
jboss@mundo-be:/usr/local/share/jboss/bin$ ./add-user.sh
```

Enter the details of the new user to add Realm (ManagementRealm):
Username: jbossAdmin
Password:
Re-enter Password:
About to add user 'jbossAdmin' for realm 'ManagementRealm'
Is this correct yes/no? yes
Added user ‘jbossAdmin’ to file ‘/usr/local/share/jboss/standalone/configuration/mgmt-users.properties’

Added user ‘jbossAdmin’ to file ‘/usr/local/share/jboss/domain/configuration/mgmt-users.properties’

4. Management console – activate access for external connections

Edit standalone.xml file

$ vi /usr/local/share/jboss/standalone/configuration/standalone.xml

changing

<!!--<socket-binding name="management-http" interface="management"
port="${jboss.management.http.port:9990}"/>-->

<socket-binding name="management-http" interface="public"
port="${jboss.management.http.port:9990}"/>

5. Configure jboss as service – (note: depending on Linux distribution)

Copy script jbosas7 to /etc/init.d/
Modify $JBOSS_HOME and $JBOSS_OTPS Set
execute mode

$ sudo chmod +x /etc/init.d/jbossas7

Start/stop/log tests
Note: it’s best to execute them as jboss user

service jbossas7 start

service jbossas7 stop

service jbossas7 log

6. Configure database for Java backend application (PostgreSQL)

Use the same PostgreSQL database that was used for CKAN, only create a new instance.
Add a new user and database.

sudo -u postgres createuser -S -D -R -P java_backend
sudo -u postgres createdb -O java_backend java_backend -E utf-8

Optionally, enable connection with the database from other IP networks sudo vi
/etc/postgresql/9.3/main/pg_hba.conf
sudo vi /etc/postgresql/9.3/main/postgresql.conf
sudo service postgresql restart
6. Install PostgreSQL driver for JBOSS and add a new DataSource

Install the jdbc driver for PostgreSQL using administrator panel’s GUI
Runtime Manage deployments – choose jdbc driver from the file

Figure 2 Install PostgreSQL driver

In the next step, enable the driver

Figure 3 Install PostgreSQL driver
Configure the connection to the database

In Runtime Datasources menu, choose Datasources Add

**Figure 4 Configure connection to PostgreSQL**

Set its parameters:

**Data source parameters Name:**

java_backend

**JNDI:** java:jboss/datasources/java_backend

**Driver:** postgresql-9.2-1002.jdbc3.jar

**Connection URL:** jdbc:postgresql://10.10.10.10:5432/java_backend

**User:** java_backend

**Password:** according to the previous point

And perform activation (enable)
7. Install mundo-java-backend application

Upload the application in the form of mundo-java-backend.war file onto the server using Jboss Management Console

Runtime – Manage Deployments – Add

---

Figure 5 Configure connection to PostgreSQL

Figure 6 Install mundo-java-backend application
Indicate the mundo-java-backend.war file

Figure 7 Install mundo-java-backend application

And activate (enable)

Figure 8 Install mundo-java-backend application
Figure 9 Install mundo-java-backend application

Figure 10 Install mundo-java-backend application
Verify installation

After going to http://server_ip_address:8080/cbr/mundo-java-backend/ the log-in page should show:

![MUNDO BACKEND](image)

Figure 11 Install mundo-java-backend application

3.3 Install CKAN plugins

In order to ensure correct cooperation between Data Server and Function Server, a number of extensions for the CKAN system were created, which ensure support in: transmitting communication between DS and FS, configuring datasets and resources and showing help in the form of a webpage (snippets) from the CKAN server

<table>
<thead>
<tr>
<th>No.</th>
<th>Extension</th>
<th>Supported data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>mundo</td>
<td>Common component for all other extensions</td>
</tr>
<tr>
<td>2</td>
<td>wmsstore</td>
<td>Raster maps from WMS resources</td>
</tr>
<tr>
<td>3</td>
<td>wfsstore</td>
<td>Vector maps from WFS resources</td>
</tr>
<tr>
<td>4</td>
<td>wsstore</td>
<td>Network services – queues</td>
</tr>
<tr>
<td>5</td>
<td>dbstore</td>
<td>Databases – on-line trams</td>
</tr>
<tr>
<td>6</td>
<td>19115store</td>
<td>API of the 19115 reporting system</td>
</tr>
</tbody>
</table>
1. Install plugins

Copy the plugin directory to /usr/lib/ckan/default/src/

2. Change permissions for the directory content

chown -R ckan /usr/lib/ckan/default/src/ckanext-mundo/*
chown -R ckan /usr/lib/ckan/default/src/ckanext-wmsstore/* chown -R ckan /usr/lib/ckan/default/src/ckanext-wfsstore/* chown -R ckan /usr/lib/ckan/default/src/ckanext-wsstore/*
chown -R ckan /usr/lib/ckan/default/src/ckanext-dbstore/*
chown -R ckan /usr/lib/ckan/default/src/ckanext-19115store/*

3. Install and activate mundo plugin

. /usr/lib/ckan/default/bin/activate
cd /usr/lib/ckan/default/src/ckanext-mundo
python setup.py develop

4. Install and activate wmsstore plugin

. /usr/lib/ckan/default/bin/activate
cd /usr/lib/ckan/default/src/ckanext-wmsstore python setup.py develop

5. Install and activate wfsstore plugin

. /usr/lib/ckan/default/bin/activate
cd /usr/lib/ckan/default/src/ckanext-wfsstore python setup.py develop

6. Install and activate wsstore plugin

. /usr/lib/ckan/default/bin/activate
cd /usr/lib/ckan/default/src/ckanext-wsstore python setup.py develop
7. Install and activate dbstore plugin

. /usr/lib/ckan/default/bin/activate
cd /usr/lib/ckan/default/src/ckanext-dbstore python setup.py develop

8. Install and activate 19115store plugin

. /usr/lib/ckan/default/bin/activate
cd /usr/lib/ckan/default/src/ckanext-19115store python setup.py develop

9. Install and activate dbstore plugin

. /usr/lib/ckan/default/bin/activate
cd /usr/lib/ckan/default/src/ckanext-dbstore python setup.py develop

10. Activate CKAN plugins

In the file /etc/ckan/default/production.ini, in the section plugins, add: mondo – required for the other plugins

wmsstore – WMS handling
wfsstore – WFS handling
wsstore – WS handling
dbstore – DB handling
19115store – 19115 handling
dbtimetable – handling Public Transport Authority timetables

11. Restart the Apache server

service httpd stop
service httpd start
or
service httpd restart
12. Verify installation

After accessing the main page, CKAN server should not return any errors.
### 4 Glossary – list of abbreviations

<table>
<thead>
<tr>
<th>No.</th>
<th>Abbreviation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>2</td>
<td>GPL</td>
<td>GNU Public License – free and open software license</td>
</tr>
<tr>
<td>3</td>
<td>LGPL</td>
<td>Lesser GNU Public License – a weaker GNU general public license, general public license for libraries</td>
</tr>
<tr>
<td>4</td>
<td>FDL</td>
<td>Free Documentation License</td>
</tr>
<tr>
<td>5</td>
<td>CC</td>
<td>Creative Commons</td>
</tr>
<tr>
<td>6</td>
<td>CKAN</td>
<td>Comprehensive Knowledge Archive Network – a web-based system which is a data repository and exposes open data created by the Open Knowledge Foundation</td>
</tr>
<tr>
<td>7</td>
<td>WS</td>
<td>Web Service based on SOA/SOAP or ROA/REST</td>
</tr>
<tr>
<td>8</td>
<td>SOA</td>
<td>Service-Oriented Architecture</td>
</tr>
<tr>
<td>9</td>
<td>SOAP</td>
<td>Simple Object Access Protocol - XML-based protocol for calling remote access to objects</td>
</tr>
<tr>
<td>10</td>
<td>ROA</td>
<td>Resource-oriented architecture</td>
</tr>
<tr>
<td>11</td>
<td>REST</td>
<td>Representational State Transfer – software architecture model</td>
</tr>
<tr>
<td>12</td>
<td>MD</td>
<td>metadata (data about data)</td>
</tr>
<tr>
<td>13</td>
<td>WMS</td>
<td>Web Map Service – map-sharing standard created by the Open Geospatial Consortium (OGC), enabling sharing raster maps using HTTP interface</td>
</tr>
<tr>
<td>14</td>
<td>WFS</td>
<td>Web Feature Service map-sharing standard created by the Open Geospatial Consortium (OGC), enabling sharing vector maps using HTTP interface</td>
</tr>
<tr>
<td>15</td>
<td>DS</td>
<td>Data set</td>
</tr>
<tr>
<td>16</td>
<td>FS</td>
<td>Function Set – a set of functions, e.g. performed by a web service</td>
</tr>
<tr>
<td>17</td>
<td>APIKey</td>
<td>Alphanumeric code transferred to the API server by computer programs calling API</td>
</tr>
<tr>
<td>18</td>
<td>RDF</td>
<td>Resource Description Framework – language/method enabling description of Web resources, with XML-based syntax, developed by W3C</td>
</tr>
<tr>
<td>19</td>
<td>Dublin Core</td>
<td>(Dublin Core Metadata Element Set, DC, DCES) – general metadata standard. Accepted as ISO 15836-2003 standard.</td>
</tr>
<tr>
<td>20</td>
<td>DCAT</td>
<td>W3C Data Catalog Vocabulary – standard of catalog scheme for data</td>
</tr>
<tr>
<td>21</td>
<td>CSV</td>
<td>Comma separated values – a file in which values are separated with commas</td>
</tr>
<tr>
<td>22</td>
<td>CMS</td>
<td>content management system</td>
</tr>
<tr>
<td>23</td>
<td>SPARQL</td>
<td>SPARQL Protocol And RDF Query Language</td>
</tr>
<tr>
<td>24</td>
<td>OWL</td>
<td>Web Ontology Language – language with XML-based syntax and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>description logic-based semantics</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>25</td>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>26</td>
<td>SSL</td>
<td>Secure Socket Layer</td>
</tr>
</tbody>
</table>
5 References


[4] Install Tomcat 6 on CentOS or RHEL http://www.davidghedini.com/pg/entry/install_tomcat_6_on_centos


