CKAN Notes

Table of contents:

CKAN Notes

This is a collection of notes on CKAN.

Install CKAN with Docker

In this section we will learn how to install CKAN with Docker.

Install CKAN with Package

In this section we will learn how to install CKAN with Package.

Install CKAN with Docker

In this section we will learn how to install CKAN with Docker.

What you'll need

- CKAN Docker clone master branch.

Install CKAN with Docker

step 1: Clone the CKAN Docker repository

```
git clone https://github.com/ckan/ckan-docker.git
```

step 2: Change directory to the CKAN Docker repository

```
cd ckan-docker
```

step 3: Create a `.env` file

```
cp .env.template .env
```

step 4: Edit the `.env` file

```
vim .env
```

step 5: modify the `.env` file according to your needs like:

```
# CKAN core
CKAN_VERSION=2.10.0
CKAN_SITE_ID=default
CKAN_SITE_URL=https://localhost:8443
CKAN___BEAKER__SESSION__SECRET=CHANGE_ME
```

```
CKAN___API_TOKEN__JWT__ENCODE__SECRET=string:CHANGE_ME
CKAN___API_TOKEN__JWT__DECODE__SECRET=string:CHANGE_ME
CKAN_SYSADMIN_NAME=ckan_admin
CKAN_SYSADMIN_PASSWORD=test1234
CKAN_SYSADMIN_EMAIL=your_email@example.com
```

step 6: Modify the `Dockerfile` in ckan folder

```
FROM ckan/ckan-base:2.10.3

# Install any extensions needed by your CKAN instance
# See Dockerfile.dev for more details and examples

# Copy custom initialization scripts
COPY docker-entrypoint.d/* /docker-entrypoint.d/

# Apply any patches needed to CKAN core or any of the built extensions (not the
# runtime mounted ones)
COPY patches ${APP_DIR}/patches

RUN for d in $APP_DIR/patches/*; do \
        if [ -d $d ]; then \
            for f in `ls $d/*.patch | sort -g`; do \
                cd $SRC_DIR/`basename "$d"` && echo "$0: Applying patch $f to
$SRC_DIR/`basename $d`"; patch -p1 < "$f" ; \
            done ; \
        fi ; \
    done
```

- here you can change the `CKAN_VERSION` according to your needs.
- you can also add any extensions needed by your CKAN instance.
- you can also add any patches needed to CKAN core or any of the built extensions (not the runtime mounted ones).

step 7: Build the CKAN image

```
docker-compose build
```

step 8: Run the CKAN image

```
docker-compose up -d
```

Install CKAN with Package

In this section we will learn how to install CKAN with Package.

What you'll need

- ubunutu version 20.4
- CKAN DOCS official docs.

CKAN Installation Guide on Ubuntu 20.04

Follow these steps to install CKAN on Ubuntu 20.04:

1. Login in as root user. If this is your first time, you will need to create a root password:

   ```
   sudo passwd root
   ```

2. Login as root user, enter password when prompted:

   ```
   sudo -i
   ```

3. Update ubuntu package index:

   ```
   sudo apt update
   ```

4. Install Ubuntu required packages for CKAN:

   ```
   sudo apt install -y libpq5 redis-server nginx supervisor
   ```

5. Download CKAN package. In this guide we are installing CKAN for Python 3:

   ```
   wget https://packaging.ckan.org/python-ckan_2.9-py3-focal_amd64.deb
   ```

6. Install additional needed packages:

   ```
   python3 --version
   sudo apt-get install python3.8-distutils
   ```

7. Install CKAN package:

   ```
   sudo dpkg -i python-ckan_2.9-py3-focal_amd64.deb
   ```

8. Install PostgreSQL:

   ```
   sudo apt install -y postgresql
   ```

9. Check that it installed correctly by running the following command and ensuring that the encoding of databases is UTF8:

   ```
   sudo -u postgres psql -l
   ```

10. Create a database user and create a password for the new user when prompted, replace `username` with the username of your choice:

    ```
    sudo -u postgres createuser -S -D -R -P username
    ```

11. Create a new PostgreSQL database owned by the new user, replace `ckan_default` with a database name of your choice and `username` with the username of the user just created:

    ```
    sudo -u postgres createdb -O username ckan_default -E utf-8
    ```

12. If you do not have vim installed you will need to install it with the following:

    ```
    apt install vim
    ```

13. Edit the CKAN configuration file and fill in the password, database, and database user for the database you've created:

    ```
    vim /etc/ckan/default/ckan.ini
    ```

    In this example our information is as follows:

    - Username: `ckanuser`
    - Password: `password`

- Database Name: `ckan_default`

14. Install Solr:

    ```
    sudo apt install -y solr-tomcat
    ```

15. Change the default port Tomcat runs on to the one expected by CKAN:

    ```
    vim /etc/tomcat9/server.xml
    ```

    Edit the following line:

    ```
    <Connector port="8080" protocol="HTTP/1.1"
    ```

    To:

    ```
    <Connector port="8983" protocol="HTTP/1.1"
    ```

16. Replace the default `schema.xml` file with a symlink to the CKAN schema file:

    ```
    sudo mv /etc/solr/conf/schema.xml /etc/solr/conf/schema.xml.bak
    sudo ln -s /usr/lib/ckan/default/src/ckan/ckan/config/solr/schema.xml
    /etc/solr/conf/schema.xml
    ```

17. Restart Solr:

    ```
    sudo service tomcat9 restart
    ```

18. Check that Solr is running by entering the following into your browser:

    ```
    http://localhost:8983/solr/
    ```

19. Edit the `solr_url` line in the CKAN configuration file to go to your Solr server:

```
vim /etc/ckan/default/ckan.ini
```

Example:

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

20. Also edit the following options in the CKAN configuration File:

    - `site_id=ckan_default`
    - `site_url=http://localhost`

21. Initialize CKAN database by running the following command:

```
sudo ckan db init
```

22. Reload the Supervisor daemon so the new processes are picked up:

```
sudo supervisorctl reload
```

23. Check the status of the processes:

```
sudo supervisorctl status
```

The following should appear with no errors:

```
ckan-datapusher:ckan-datapusher-00   RUNNING   pid 1963, uptime 0:00:12
ckan-uwsgi:ckan-uwsgi-00             RUNNING   pid 1964, uptime 0:00:12
ckan-worker:ckan-worker-00           RUNNING   pid 1965, uptime 0:00:12
```

24. Restart Nginx:

```
sudo service nginx restart
```

25. Enter the following into your web browser and the CKAN front page should appear:

```
http://localhost/
```

Congratulations! You have successfully installed CKAN! 👌

Additional CKAN Setup Steps

Follow these steps to create a super user, setup file uploads, activate the CKAN virtual environment, and update the dataproxy timeout:

1. To create a super user in CKAN, run the following command, replacing `username`, `email@gmail.com`, and `name` with your desired values:

   ```
   ckan -c /etc/ckan/default/ckan.ini sysadmin add username email=email@gmail.com
   name=name
   ```

   Example:

   ```
   ckan -c /etc/ckan/default/ckan.ini sysadmin add vivek email=vivek2292@gmail.com
   name=vivek
   ```

2. To setup CKAN's FileStore with local file storage:

   - Create the directory where CKAN will store uploaded files:

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

   - Add the following line to your CKAN config file, after the `[app:main]` line:

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

   - Set the permissions of your `ckan.storage_path` directory. For example if you're running CKAN with Nginx, then the Nginx's user (`www-data` on Ubuntu) must have read, write and execute permissions for the `ckan.storage_path`:

     ```
     sudo chown www-data /var/lib/ckan/default
     sudo chmod u+rwx /var/lib/ckan/default
     ```

- Restart your web server, for example to restart uWSGI on a package install:

  ```
  sudo supervisorctl restart ckan-uwsgi:*
  ```

3. To activate the CKAN virtual environment:

   ```
   . /usr/lib/ckan/default/bin/activate
   ```

4. To update the dataproxy timeout:

   - Open the `recline_view.js` file:

     ```
     vim
     /usr/lib/ckan/default/src/ckan/ckanext/reclineview/theme/public/recline_view.js
     ```

   - Update the following line:

     ```
     recline.Backend.DataProxy.timeout = 100000;
     ```

Note: You can also refer old docs for installation of CKAN with Package here

install Azure AD and configure on CKAN

You have just learned the How to install CKAN

install Azure AD and configure on CKAN

You have just learned the **How to install CKAN**

in this section we will learn how to install Azure AD on CKAN

What you'll need

- CKAN version 2.9 or above:
- Azure AD version 2.0 or above:
- ckan-msal compatible with CKAN 2.9.6 and CKAN 2.9.7

Installation

To install ckanext-msal:

1. Activate your CKAN virtual environment, for example::

   ```
   . /usr/lib/ckan/default/bin/activate
   ```

2. Install the ckanext-msal Python package into your virtual environment::

   ```
   git clone https://github.com/boykoc/ckanext-msal.git
   cd ckanext-msal/
   python setup.py develop
   pip install -r requirements.txt
   ```

3. Add `msal` to the `ckan.plugins` setting in your CKAN config file (by default the config file is located at `/etc/ckan/default/production.ini`).

4. Edit `msal_config.py <https://github.com/ongov/ckanext-msal/blob/ckan_2.9.7_compatible/ckanext/msal/msal_config.py>`_ and replace the generic values with your specific credentials.

5. Restart CKAN. For example if you've deployed CKAN on Ubuntu::

   ```
   sudo service supervisor restart
   ```

---

if you are using docker

copy paste below code in your docker file and add `msal` in .env file

```
### Msal ###
RUN  pip3 install -e git+https://github.com/ongov/ckanext-msal.git#egg=ckanext-msal

COPY msal_config.py ${APP_DIR}/src/ckanext-msal/ckanext/msal/msal_config.py

```

then

```
docker compose -f docker-compose.yml build --no-cache
docker compose -f docker-compose.yml up -d
```

Configuration

your `msal_config.py` should look like this

```
# MSAL configurations.
AUTHORITY = "https://login.microsoftonline.com/d0139cb0-61c2-7362-9vf8c-e229cdf0fbed"
CLIENT_ID = "5b040eb5-a574-8464-c489bc25580d"
SCOPE = ["User.ReadBasic.All"]
REDIRECT_URI = "http://localhost:5000/getAToken"
CLIENT_SECRET = "P6D8Q~jhdsuhftIqZJs7N.k~EEOl~-axC"

# Plugin specific configurations.
EMAIL_DOMAINS = ["google.com","manishacharyagmail.onmicrosoft.com"]
```

- AUTHORITY: The authority URL for your tenant. For example: https://login.microsoftonline.com/your-tenant-name.onmicrosoft.com
- CLIENT_ID: The application ID of your app registered in Azure AD.
- SCOPE: The scopes required by your app. For example: ["User.ReadBasic.All"].
- REDIRECT_URI: The redirect URI of your app. For example: http://localhost:5000/getAToken.
- CLIENT_SECRET: The client secret of your app.
- EMAIL_DOMAINS: The list of email domains that are allowed to access CKAN. For example: ["google.com"].

install harvester and configure on CKAN

You have just learned the How to install and configure AZURE AD on CKAN

install harvester and configure on CKAN

You have just learned the **How to install and configure AZURE AD on CKAN**

in this section we will learn how to install and configure harvester on CKAN

What you'll need

- CKAN version 2.9 or above:
- ckanext-harvest compatible with CKAN 2.9.6 and above.

Installation with Docker

To install ckanext-harvest copy paste below code in your docker file.

```
### harvester ###
RUN pip3 install -e 'git+https://github.com/ckan/ckanext-harvest.git@master#egg=ckanext-
harvest' && \
    pip3 install -r ${APP_DIR}/src/ckanext-harvest/pip-requirements.txt && \
    pip3 install -r ${APP_DIR}/src/ckanext-harvest/dev-requirements.txt



# # Install Supervisor
RUN apk update && apk add --no-cache python3-dev py3-setuptools supervisor dcron
busybox-extras
RUN pip install --upgrade supervisor schedule future

# # Copy Supervisor configuration
COPY harvesting.conf /etc/supervisord.d/harvesting.conf



COPY harvester.py /srv/app
```

your `harvesting.conf` file should look like this

```
[program:ckan_gather_consumer]
command=ckan harvester gather-consumer
user=root
numprocs=1
stdout_logfile=/var/log/gather_consumer.log
stderr_logfile=/var/log/gather_consumer.err.log
autostart=true
autorestart=true
startsecs=10

[program:ckan_fetch_consumer]
command=ckan harvester fetch-consumer
user=root
numprocs=1
stdout_logfile=/var/log/fetch_consumer.log
stderr_logfile=/var/log/fetch_consumer.err.log
autostart=true
autorestart=true
startsecs=10
```

your `harvester.py` file should look like this

```
import schedule
import time
import subprocess



def run_harvester():
    # Redirect logs to a file
    with open("harvester.log", "a") as log_file:
        # Execute the command "ckan harvester run" and redirect stdout and stderr to the log file
        subprocess.run(["ckan", "harvester", "run"], stdout=log_file, stderr=log_file)
```

```
# Schedule the job to run every 5 minutes
schedule.every(5).minutes.do(run_harvester)

# Run the scheduler continuously
while True:
    schedule.run_pending()
    time.sleep(1)
```

in your `.env` file add below lines

```
CKAN__PLUGINS = harvest ckan_harvester

CKAN__HARVEST__MQ__TYPE=redis
CKAN__HARVEST__MQ__HOSTNAME=redis
CKAN__HARVEST__MQ__PORT=6379
CKAN__HARVEST__MQ__REDIS_DB=0
```

some of the important commands to run harvester

```
python3 harvester.py &
```

harvester official documentation

refer this link https://github.com/ckan/ckanext-harvest#command-line-interface

some above commads meaning:

- geather consumer: This command will gather all the harvest sources and create jobs for each of them. It will also create jobs for the harvest sources that are scheduled to run periodically.
- fetch consumer: This command will fetch the harvest objects for each of the jobs created by the gather consumer. It will also create jobs for the harvest objects that are scheduled to run periodically.
- Supervisor in Harvester: in Harvester supervisor is used to run the harvester in background. Supervisor is a client/server system that allows its users to monitor and control a number of processes on UNIX-like operating systems. It shares some of the same goals of programs like launchd, daemontools, and runit.

> Unlike some of these programs, it is not meant to be run as a substitute for init as “process id 1”. Instead it is meant to be used to control processes related to a project or a customer, and is meant to start like any other program at boot time.

creating harvest source using ckan UI

- go to your ckan and login as admin then to `/harvest` page and click on `Add Harvest Source` button
- fill the form and click on `save` button

Here is an example of a configuration object (the one that must be entered in the configuration field):

```
{
 "api_version": 1,
 "default_tags": [{"name": "geo"}, {"name": "namibia"}],
 "default_groups": ["science", "spend-data"],
 "default_extras": {"encoding":"utf8", "harvest_url": "
{harvest_source_url}/dataset/{dataset_id}"},
 "override_extras": true,
 "organizations_filter_include": [],
 "organizations_filter_exclude": ["remote-organization"],
 "user":"harverster-user",
 "api_key":"<REMOTE_API_KEY>",
 "read_only": true,
 "remote_groups": "only_local",
 "remote_orgs": "create"
}
```

read more here

install Schemming on CKAN

in this section we will learn how to install Schemming on CKAN

install Schemming on CKAN

in this section we will learn how to install Schemming on CKAN

What you'll need

- CKAN version 2.9 or above:
- ckanext-scheming compatible with CKAN 2.9.6 and above.
- official documentation of ckanext-scheming.

Installation with Docker

To install ckanext-scheming copy paste below code in your docker file.

```
### Scheming ###
RUN  pip3 install -e 'git+https://github.com/ckan/ckanext-
scheming.git@master#egg=ckanext-scheming'

COPY translate.json /srv/app/src/ckanext-scheming/ckanext/scheming

COPY package_form.html /srv/app/src/ckanext-
scheming/ckanext/scheming/templates/scheming/package/snippets/package_form.html
```

add below code in your `ckan.ini` file

```
scheming.dataset_schemas = ckanext.scheming:translate.json
scheming.presets = ckanext.scheming:presets.json
```

your `translate.json` file should look like this

translate.json

```
{
    "scheming_version": 2,
    "dataset_type": "formpages",
    "about": "A reimplementation of the default CKAN dataset schema",
    "about_url": "http://github.com/ckan/ckanext-scheming",
    "dataset_fields": [
        {
            "start_form_page": {
                "title": "Required Fields",
                "description": "These fields provide detailed metadata about the
dataset."
            },
            "field_name": "title",
            "label": "Title",
            "preset": "title",
            "form_placeholder": "eg. A descriptive title"
        },
        {
            "field_name": "name",
            "label": "URL",
            "preset": "dataset_slug",
            "form_placeholder": "eg. my-dataset"
        },
        {
            "field_name": "owner_org",
            "label": "Organization",
            "preset": "dataset_organization"
        },
        {
            "start_form_page": {
                "title": "English",
                "description": "These fields provide detailed metadata about the
dataset."
            },
            "field_name": "en_title",
            "label": "English Dataset Title",
            "preset": "title",
            "form_placeholder": "eg. A descriptive title"
        },
        {
            "field_name": "notes",
            "label": "English Description",
            "form_snippet": "markdown.html",
            "form_placeholder": "eg. Some useful notes about the data"
        },
        {
            "start_form_page": {
                "title": "French dataset",
                "description": "These fields provide detailed metadata about the
dataset."
            },
            "field_name": "fr_title",
            "label": "Freanch Dataset Title",
            "preset": "title",
            "form_placeholder": "eg. A descriptive title"
        },
        {
            "field_name": "fr_notes",
            "label": "French Description",
            "form_snippet": "markdown.html",
            "form_placeholder": "eg. Some useful notes about the data"
        },
        {
            "start_form_page": {
                "title": "Spanish dataset",
                "description": "These fields provide detailed metadata about the
dataset."
            },
            "field_name": "es_title",
            "label": "Spanish Dataset Title",
            "preset": "title",
            "form_placeholder": "eg. A descriptive title"
        },
        {
            "field_name": "es_notes",
            "label": "Spanish Description",
            "form_snippet": "markdown.html",
            "form_placeholder": "eg. Some useful notes about the data"
        },
        {
            "start_form_page": {
                "title": "Russian dataset",
                "description": "These fields provide detailed metadata about the
dataset."
            },
            "field_name": "ru_title",
            "label": "Russian Dataset Title",
            "preset": "title",
            "form_placeholder": "eg. A descriptive title"
        },
        {
            "field_name": "ru_notes",
            "label": "Russian Description",
            "form_snippet": "markdown.html",
            "form_placeholder": "eg. Some useful notes about the data"
        },
        {
            "start_form_page": {
                "title": "Chinese dataset",
                "description": "These fields provide detailed metadata about the
dataset."
            },
            "field_name": "zh_title",
            "label": "Chinese Dataset Title",
            "preset": "title",
            "form_placeholder": "eg. A descriptive title"
        },
        {
            "field_name": "zh_notes",
            "label": "Chinese Description",
            "form_snippet": "markdown.html",
            "form_placeholder": "eg. Some useful notes about the data"
        },
        {
            "start_form_page": {
                "title": "Arabic dataset",
                "description": "These fields provide detailed metadata about the
dataset."
            },
            "field_name": "ar_title",
            "label": "Arabic Dataset Title",
            "preset": "title",
            "form_placeholder": "eg. A descriptive title"
        },
        {
            "field_name": "ar_notes",
            "label": "Arabic Description",
            "form_snippet": "markdown.html",
            "form_placeholder": "eg. Some useful notes about the data"
        }
    ],
    "resource_fields": [
        {
            "field_name": "url",
            "label": "URL",
            "preset": "resource_url_upload"
        },
        {
            "field_name": "name",
            "label": "Name",
            "form_placeholder": "eg. January 2011 Gold Prices"
        },
        {
            "field_name": "description",
            "label": "Description",
            "form_snippet": "markdown.html",
            "form_placeholder": "Some useful notes about the data"
        },
        {
            "field_name": "format",
            "label": "Format",
            "preset": "resource_format_autocomplete"
        }
    ]
}
```

your `package_form.html` file should look like this

package_form.html

```
{% extends 'package/new_package_form.html' %}

{% block stages %}
{%- set pages = h.scheming_get_dataset_form_pages(dataset_type) -%}
{%- if pages -%}
{%- set active_page = data.get('_form_page', 1) | int -%}


<ol class="stages stage-1">
    {%- for p in pages -%}
    {%-set pg_url = h.url_for(dataset_type +
    ('.scheming_edit_page' if form_style == 'edit' else '.scheming_new_page'),
    package_type=dataset_type,
    id=data.name or data.id,
    page=loop.index) -%}

    <li class="{{
          'first ' if loop.first else ''}}{{
          'active ' if loop.index == active_page else '' }}"
        style="width:{{ 100 / (loop.length + (0 if form_style == 'edit' else 1)) }}%">
        <span class="highlight" style="padding-right:0">
            {% if '//' not in pg_url.strip() %}
            <a href="{{
              h.url_for(dataset_type +
                  ('.scheming_edit_page' if form_style == 'edit' else
'.scheming_new_page'),
                package_type=dataset_type,
                id=data.name or data.id,
                page=loop.index)
            }}">{{ h.scheming_language_text(p.title) }}</a>
            {%
            else %}{{ h.scheming_language_text(p.title) }}{% endif %}
        </span>
    </li>
    {%- endfor -%}
    {%- if form_style != 'edit' -%}
    <li class="last {{ s2 }}" style="width:{{ 100 / (pages | length + 1) }}%">
        {% if s2 != 'complete' %}
        <span class="highlight">{{ _('Add data') }}</span>
        {% else %}
        {% if s1 == 'active' %}
        {# stage 1 #}
        <button class="highlight" name="save" value="go-resources" type="submit">{{
_('Add data') }}</button>
        {% else %}
        {% link_for _('Add data'), named_route='dataset.new', class_="highlight" %}
        {% endif %}
        {% endif %}
    </li>
    {%- endif -%}
</ol>
{%- else -%}
{{ super() }}
{%- endif -%}
{% endblock %}

{% block errors %}
{%- if errors -%}
{%- set schema = h.scheming_get_dataset_schema(dataset_type) -%}
{%- snippet 'scheming/snippets/errors.html',
errors=errors, fields=schema.dataset_fields,
entity_type='dataset', object_type=dataset_type -%}
{%- endif -%}
{% endblock %}

{% block basic_fields %}
```

```
{%- if not dataset_type -%}
<p>
    dataset_type not passed to template. your version of CKAN
    might not be compatible with ckanext-scheming
</p>
{%- endif -%}

{%- set schema = h.scheming_get_dataset_schema(dataset_type) -%}
{%- set pages = h.scheming_get_dataset_form_pages(dataset_type) -%}
{%- if pages -%}
{%- set active_page = data.get('_form_page', 1) | int -%}
{%- set fields = pages[active_page - 1]['fields'] -%}
{%- else -%}
{%- set fields = schema.dataset_fields -%}
{%- endif -%}
{%- for field in fields -%}
{%- if field.form_snippet is not none -%}
{%- if field.field_name not in data %}
{# Set the field default value before rendering but only if
it doesn't already exist in data which would mean the form
has been submitted. #}
{% if field.default_jinja2 %}
{% do data.__setitem__(
field.field_name,
h.scheming_render_from_string(field.default_jinja2)) %}
{% elif field.default %}
{% do data.__setitem__(field.field_name, field.default) %}
{% endif %}
{% endif -%}
{%- snippet 'scheming/snippets/form_field.html',
field=field,
data=data,
errors=errors,
licenses=c.licenses,
entity_type='dataset',
object_type=dataset_type
-%}
{%- endif -%}
{%- endfor -%}

{%- if pages -%}
<input type="hidden" name="_ckan_phase" value="{{ active_page }}" />
{%- elif 'resource_fields' not in schema -%}
<!-- force controller to skip resource-editing step for this type -->
<input type="hidden" name="_ckan_phase" value="" />
{%- endif -%}
```

```
{% endblock %}

{% block metadata_fields %}
{% endblock %}

{% block save_button_text %}
{%- set pages = h.scheming_get_dataset_form_pages(dataset_type) -%}
{%- if pages and form_style == 'edit' -%}
{%- set active_page = data.get('_form_page', 1) | int -%}
{{ _('Update {page}').format(page=h.scheming_language_text(pages[active_page-1].title))
}}
{%- elif pages -%}
{{ _('Next') }}
{%- else -%}
{{ super() }}
{%- endif -%}
{% endblock %}
```

How to install Azure AD on CKAN

You have just learned the **How to install CKAN**

in this section we will learn how to install Azure AD on CKAN

What you'll need

- CKAN version 2.9 or above:
- Azure AD version 2.0 or above:
- ckan-msal compatible with CKAN 2.9.6 and CKAN 2.9.7