govhack2015 Documentation

Release latest

November 17, 2015

Contents

target http://datawagovau.readthedocs.org/en/latest/?badge=latest

alt Documentation Status

This repository contains instructions to setup and deploy a data portal based on CKAN, plus some infrastructure to automate the information pipelines along the pyramid of data - information - knowledge - wisdom. Deployment documents the setup of a virtual machine, ready for the install of CKAN and the Data Science Workbench. GovHack 2015 contains the history of this project as a GovHack submission in July 2015. The last two chapters illustrate an alternative source install for CKAN and the workbench packages. Installing from source is superseded by datacats and docker, but might be advatageous for systems with single-sign-on and shared authentication.

CHAPTER 1

Deployment

This chapter explains the deployment of a Virtual Machine on the Amazon web services (AWS) cloud and hosting on a custom domain. The goal is a VM with ssh access which runs a default web page on a custom domain, and provide a scalable way to add additional services. We will build:

- AWS EC2 t2.medium VM with at least 16 GB, if possible 100 GB hdd (the default of 8 GB is too small for OS and docker images)
- AWS 100 GB SSD volume, formatted with btrfs, mounted in above instance at e.g. `/mnt/btrfsvol/` with symlinks to locations containing large files

All non-ephemeral data should live inside `/mnt/btrfsvol/`, which includes: * the datacats installation * the datacats data directory (soft-linked from default ~/.datacats) * the docker image dir (soft-linked from default /var/lib/docker) * Latex docs (> 1 GB)

The benefits: * the external volume can be snapshotted for backup&recovery * the external volume is btrfs-formatted, docker supports btrfs natively

1.1 AWS EC2 instance

In your AWS console, create a VM and a storage volume, attach the volume to the VM, create a static IP (ElasticIP). You will want to have an SSH keypair configured at AWS.

- Login to AWS management console, e.g. Sydney
- EC2 Dashboard
- Launch instance
- Ubuntu Server 14.04 LTS 64 bit (HVM), SSD Volume Type
- t2.medium
- Configure instance details
- Set shutdown behaviour to "stop", protect against accidental termination
- Set size of instance to 16 GB
- Add Storage: 100 GB SSD
- Tag instance (choose a name)
- Configure Security Group: SSH, HTTP, HTTPS (or select from existing security groups providing these three protocols)
- Review and Launch

- Select existing SSH key pair (or create new one)
- View instances

Test: **AWS console > EC2 > Instances** should list new instance

Format the external volume:

```
sudo su
apt-get -y install btrfs-tools libxml2-utils gdal-bin

fdisk -l
# will show external, as yet unformatted volume as e.g. "xvdb" as "no partition table found"
# your volume label will vary
mkfs.btrfs /dev/xvdb
mkdir -p /mnt/btrfsvol
echo "/dev/xvdb /mnt/btrfsvol  btrfs rw,relatime,ssd,space_cache  0 0" >> /etc/fstab
mount -a
```

Inspect new partition:

```
df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1       16G  1.1G   14G   8% /
none            4.0K     0  4.0K   0% /sys/fs/cgroup
udev            2.0G   12K  2.0G   1% /dev
tmpfs           396M  332K  395M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            2.0G     0  2.0G   0% /run/shm
none            100M     0  100M   0% /run/user
/dev/xvdb       100G  512K   98G   1% /mnt/btrfsvol
```

1.2 Folder structure

Locations of large files should be symlinked to the external volume `/mnt/btrfsvol/`. We will replace the following default locations with symlinks into the external volume:

- Apache's web app directory `/var/www`
- Docker's image directory `/var/lib/docker`
- A general purpose directory for projects, such as datacats
- Virtualenvwrapper's virtualenv directory (which we will configure in .bashrc) `/var/venvs`
- Datacats' data directory `~./datacats`
- User docs, including the sizeable Latex docs, will be moved from their original location `/usr/share/doc` to `/var/www/doc` and back-linked
- User ubuntu's home directory (accessible through RStudio Server) contains a symlink to `/mnt/btrfsvol/shiny-server`

Create custom folders:

```
sudo su

mkdir -p /mnt/btrfsvol/www /mnt/btrfsvol/docker /mnt/btrfsvol/projects /mnt/btrfsvol/venvs /mnt/btrfs

ln -s /mnt/btrfsvol/www /var/www
ln -s /mnt/btrfsvol/docker /var/lib/docker
```

```
ln -s /mnt/btrfsvol/projects /var/projects
ln -s /mnt/btrfsvol/venvs /var/venvs
ln -s /mnt/btrfsvol/datacats_data /home/ubuntu/.datacats

mv /usr/share/doc /var/www
ln -s /var/www/doc /usr/share/doc

chown -R ubuntu:www-data /mnt/btrfsvol /var/www /var/lib/docker /var/projects /var/venvs /home/ubuntu
```

1.3 Docker

Install docker into mounted btrfs volume, not the (16 GB) root volume, otherwise we'll run out of space (images for datacats + OS > 8 GB), and the docker install is not snapshotted. In the previous step, we have create a folder inside the btrfsvol and symlinked docker's install default `/var/lib/docker` to that folder. A symlink is preferred over a bind-mount via fstab, as docker will recognise and natively support btrfs through the symlink. Following the prompt after the docker installation, add your non-root user to the group "docker" so they won't have to sudo docker commands.

Install docker:

```
sudo su
wget -qO- https://get.docker.com/ | sh
usermod -aG docker ubuntu
```

You **need** to logout and login again to apply the group changes, else `docker pull` will fail, e.g. when pulling datacats images.

1.4 Web servers

Install nginx, use nginx to reverse proxy subdomains to ports. Note that we won't need an Apache web server as all installed packages come with their own web servers, but we'll include it to provide a hosting encironment for Apache apps. If your custom setup requires Apache, make sure to change the default port in its `/etc/apache2/ports.conf` from 80 (which is used by our nginx) to some unused port, e.g. 8000.

Install nginx and apache:

```
sudo apt-get install -y nginx apache2
```

Apache will be not used as web server unless your custom setup requires it; nginx will serve as reverse proxy for datacats, rstudio and rshiny. Apache's document dir `/var/www` is a symlink pointing to the btrfs volume at `/mnt/btrfsvol/www` before apache2 is installed. Apache's default listening port (80) will be changed to 8000 so nginx can listen on port 80.:

```
sudo sed -i 's/Listen 80/Listen 8000/g' /etc/apache2/ports.conf
sudo sed -i 's/<VirtualHost \*:80>/<VirtualHost \*:8000>/g' /etc/apache2/sites-enabled/000-default.co
```

Add new nginx site */etc/nginx/sites-enabled/base.conf*. The DOMAIN.TLD is e.g. *yes-we-ckan.org* and we're running an Apache web server as well:

- requests to yes-we-ckan.org on port 80 will be redirected to port 8000 (apache site 1)
- other subdomains can be added similarly, redirecting to ports 8002 ff.

`/etc/nginx/sites-enabled/base.conf`:

```
proxy_cache_path /tmp/nginx_cache levels=1:2 keys_zone=cache:30m max_size=250m;
proxy_temp_path /tmp/nginx_proxy 1 2;
server {
    server_name DOMAIN.TLD www.DOMAIN.TLD;
    listen 80;
    location / {
        proxy_pass http://127.0.0.1:8000;
    }
}
server {
    server_name ckan.DOMAIN.TLD;
    listen 80;
    client_max_body_size 2G;
    location / {
        proxy_pass http://127.0.0.1:5000/;
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_set_header Host $host;
        proxy_cache cache;
        proxy_cache_bypass $cookie_auth_tkt;
        proxy_no_cache $cookie_auth_tkt;
        proxy_cache_valid 30m;
        proxy_cache_key $host$scheme$proxy_host$request_uri;
        # In emergency comment out line to force caching
        # proxy_ignore_headers X-Accel-Expires Expires Cache-Control;
    }
}

server {
    server_name rstudio.DOMAIN.TLD;
    listen 80;
    location / {
        proxy_pass http://127.0.0.1:8787;
        proxy_redirect http://127.0.0.1:8787/ $scheme://$host/;
    }
}
server {
    server_name rshiny.DOMAIN.TLD;
    listen 80;
    location / {
        proxy_pass http://127.0.0.1:3838;
    }
}

server {
    server_name pycsw.DOMAIN.TLD;
    listen 80;
    location / {
        proxy_pass http://127.0.0.1:9000;
    }
}

server {
  server_name qgis.DOMAIN.TLD;
  listen 80;
  location / {
    proxy_pass http://127.0.0.1:8100;
    proxy_redirect http://127.0.0.1:8100/ $scheme://$host/;
  }
```

```
}

server {
  server_name openrefine.DOMAIN.TLD;
  listen 80;
  location / {
    proxy_pass http://127.0.0.1:3333;
    proxy_redirect http://127.0.0.1:3333/ $scheme://$host/;
  }
}

server {
  server_name ipython.DOMAIN.TLD;
  listen 80;
  location / {
    proxy_pass http://127.0.0.1:8888;
    proxy_redirect http://127.0.0.1:8888/ $scheme://$host/;
    proxy_set_header Origin http://127.0.0.1:8888;
  }
}
```

Start apache2 and nginx services:: service apache2 start service nginx start

Apache's default page should run on port 8000. Run a test web server on port 8001 in the current directory with:

```
cd /tmp
python3 -m http.server 8001
```

`curl 127.0.0.1:8000` should show the Apache default page (source) `curl 127.0.0.1:8001` should show a "directory listing" (source) if the python3 http.server is still running. Mind that the AWS security group will only allow traffic on ports 22 (SSH), 80 (HTTP), and 443 (HTTPS).

1.5 Routing

In Amazon's Route53, create a hosted zone, add aliases to your domain name. In your domain registrar's management console, set your domain's name servers to those given to you by Route53 in your hosted zone.

- AWS management console > Route53 > Create Hosted Zone
- Name: your domain name, e.g. yes-we-ckan.org
- Add "A - IPv4 address" record with value (paste your Elastic IP) for both *yes-we-ckan.org* and **.yes-we-ckan.org*
- Make note of the NS (name servers)

1.6 Web domain

Buy a domain, e.g. yes-we-ckan.org and allow a day or two to activate. The domain registrar will allow to set custom name servers. Change the domain's DNS servers to your Route53 hosted zone's name servers (allow up to 48h to activate). E.g.:

> ns-1490.awsdns-58.org ns-521.awsdns-01.net ns-224.awsdns-28.com ns-1884.awsdns-43.co.uk

This will make requests to that domain use AWS's DNS servers, which will pick up Route 53's hosted zone pointing to your AWS VM. Nginx takes care of redirecting subdomains to internal ports.

- yes-we-ckan.org should show the Apache default page

- open-data.yes-we-ckan.org should show a "directory listing" if the python3 http.server is still running

1.7 Virtualenv

We'll install virtualenv and virtualenvwrapper as to keep installations of Python package requests contained in venv sandboxes, where they can't break pip. We will also append custom virtualenvwrapper settings to ubuntu's bashrc.:

sudo apt-get install -y python-pip sudo pip install virtualenvwrapper

cat <<EOF >> /home/ubuntu/.bashrc export WORKON_HOME=/var/venvs export PROJECT_HOME=/var/projects source /usr/local/bin/virtualenvwrapper.sh EOF

Apply with logout/login or `source ~/.bashrc`

1.8 Github SSH keys

As user ubuntu, run `ssh-keygen` and register the public key from `/home/ubuntu/.ssh/id_rsa.pub` in your Github and / or Bitbucket accounts to enable ssh authentication. This will enable you to push to your code repositories.

To make connecting a local terminal to a remote VM using SSH easier, add to your local ssh config `~/.ssh/config`:

```
Host ALIAS
    HostName ec2-IP.NODE.compute.amazonaws.com
    User ubuntu
    IdentityFile ~/.ssh/MYKEY.pem
```

This allows to SSH into HostName as User with IdentityFile by simply typing `ssh ALIAS`. Capitalised terms will differ.

1.9 Disaster recovery

On the AWS level, create snapshots of the VM and linked storage volumes. All servers should be shut down when the snapshots are taken. We recommend maintaining a rolling daily snapshot for 30 days.

On the VM level, we recommend to:

- Export all data from CKAN, RStudio home directories, RShiny app directories to Amazon S3
- Script setup of server (= automate the steps of this documentation) and import of data from S3
- Schedule the backup daily or as required
- Create Amazon Auto Scaling Group, min and max instance no = 1 to create an "auto-rebuild" instance

If the instance should go down, the auto scaling group will create a new instance and restore the data from the S3 backup.

1.10 Result

Now we'll have an Amazon AWS VM to which we can

- add servers running on custom ports (served through Apache2 or their own web servers),
- reverse proxy the ports in Nginx to subdomain names like my-subdomain.yes-we-ckan.org,

- `reload service nginx` to apply these changes.

CHAPTER 2

CKAN

This chapter documents a CKAN install using Datacats. Features:

- Datacats will be installed from source inside a virtualenv.
- The virtualenv will live in `/var/venvs/ckan`.
- The datacats installation and environments will live in `/var/projects/ckan`.
- The datacats data dir `~/.datacats` is symlinked to `/mnt/btrfsvol/datacats_data`.
- The directory `/var/lib/docker` contains all docker images.
- The directories `/var/venvs`, `/var/projects` and `/var/lib/docker` are symlinked to the external 100 GB volume `/mnt/btrfsvol/`.
- Nginx will be configured to reverse-proxy custom subdomain to servers running on local ports.
- The domain hosting redirects requests to the custom subdomains to the VM's static IP.

A note on conflicting pip and requests packages: If pip gets `ImportError: cannot import name IncompleteRead`, run `sudo easy_install requests==2.2.1`. To avoid this bug, we'll install datacats (and every other python-based project) into its own virtualenv, where they can have their preferred requests version, and the system can have its own, pip-compatible version (e.g. requests==2.2.1).

2.1 Directories and symlinks

With virtualennvwrapper installed and sourced from `~/.bashrc`, create virtualenv and project directories for datacats:

```
mkproject ckan
```

With our custom settings, this will create `/var/projects/ckan` as project directory, and `/var/venvs/ckan` for the virtualenv. It will also enable the virtualenv. Deactivate and reactivate to use the virtualenv's binaries rather than the system-wide ones. Create a symlink to `~/.datacats` *before* any datacats environment is created. Otherwise, `~/.datacats` will contain files owned by other users (root, postgres) and will have to be moved by the root user and chowned to the current user, while all datacats environments are stopped.

2.2 Datacats install

With the datacats virtualenv activated, clone the datacats repo and pull the Docker images:

```
workon ckan
(ckan)ubuntu@ip:/var/projects/ckan$

git clone https://github.com/datacats/datacats.git
cd datacats
python setup.py install
datacats pull -a
```

2.3 Datacats environments

Create an environment as per datacats docs:

```
(ckan)ubuntu@ip:/var/projects/ckan$
datacats create --ckan latest --site-url http://catalogue.alpha.data.wa.gov.au datawagovau 5000
```

This will create `/var/projects/ckan/datawagovau`, install ckan and run the server on the given port (here: 5000).

2.4 Reverse proxy the datacats environment

If the environment runs on e.g. port 5000, add this section to `/etc/nginx/sites-enabled/base.conf` to host the environment on a subdomain:

```
proxy_cache_path /tmp/nginx_cache levels=1:2 keys_zone=cache:30m max_size=250m;
proxy_temp_path /tmp/nginx_proxy 1 2;

server {
    server_name catalogue.alpha.data.wa.gov.au;
    listen 80;
    client_max_body_size 2G;
    location / {
        proxy_pass http://127.0.0.1:5000;
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_set_header Host $host;
        proxy_cache cache;
        proxy_cache_bypass $cookie_auth_tkt;
        proxy_no_cache $cookie_auth_tkt;
        proxy_cache_valid 30m;
        proxy_cache_key $host$scheme$proxy_host$request_uri;
    }
}
```

Test and apply with `sudo nginx configtest` and `sudo service nginx reload`. This will create a working CKAN without any further extensions. To enable the extensions, follow the next chapter.

2.5 Extensions

The following list of extensions displays their installation status on our example CKAN.

The installation process is:

- installed: extension repo is downloaded and installed into the datacats environment

- active: extension is enabled in CKAN config
- working: extension actually works

Between the last two steps lies a varying amount of configuration to the environment, including but not limited to:

- database additions,
- running of servers (celery task queue, redis message queue, pycsw server etc.),
- addition of config files (pycsw, harvester),
- writing to weird and wonderful locations outside the installation directory (flickrapi being the worst offender).

All these additions have to be applied within the contraints of datacats' docker-based deployment approach.

Extension	Functionality	Status
ckanext-dcat	Metadata export as RDF	working
ckanext-pages	Static pages	working
ckanext-spatial	Georeferencing (DPaW widget), spatial search	fork working
ckanext-scheming	Custom metadata schema	fork working
ckanext-pdfview	PDF resource preview	working
ckanext-geoview	Spatial resource preview	working
ckanext-cesiumpreview	NationalMap preview	working
ckanext-harvest	Metadata harvesting	in dev, currently scripted
pycsw	CSW endpoint for CKAN	working
ckan-galleries	Image hosting on CKAN	some issues
ckanext-doi	DOI minting	in dev
ckanext-archiver	Resource file archiving	working
ckanext-qa	QA checks (e.g. has DOI)	working
ckanext-hierarchy	Hierarchical organisations	working
WA data licenses	WA data licensing	pending license list
ckanext-geopusher	SHP and KML to GeoJSON converter	working
ckanext-featuredviews	Showcase resource views	works in layout 1
ckanext-showcase	Replace featured items	working
ckanext-disqus	User comments	working
ckanext-datawagovautheme	Data.wa.gov.au theme	working
ckanapi	Python client for CKAN API	working
ckanR	R client for CKAN API	working

Note: Unless specified otherwise, all code examples are executed as non-root user "ubuntu" (who must be in the docker group) in the CKAN environment's directory, e.g.:

```
workon ckan
(ckan)ubuntu@ip:/var/projects/ckan/
# cd into datacats environment "test"
cd test/
(ckan)ubuntu@ip:/var/projects/ckan/test$
```

2.5.1 Download extensions

Run:

```
git config --global push.default matching

datacats install

# ckanext-spatial custom fork
```

```
git clone git@github.com:datawagovau/ckanext-spatial.git
cd ckanext-spatial
git remote add upstream https://github.com/ckan/ckanext-spatial.git
git fetch upstream
git merge upstream/master master -m 'merge upstream'
git push
cd ..

# ckanext-scheming custom fork
git clone git@github.com:florianm/ckanext-scheming.git
cd ckanext-scheming
git remote add upstream https://github.com/open-data/ckanext-scheming.git
git fetch upstream
git merge upstream/master master -m 'merge upstream'
git push
cd ..

#git clone https://github.com/datawagovau/ckanext-datawagovautheme.git
git clone git@github.com:datawagovau/ckanext-datawagovautheme.git

#git clone https://github.com/ckan/ckanext-pages.git
git clone https://github.com/datawagovau/ckanext-pages.git

# git clone https://github.com/ckan/ckanext-harvest.git
git clone git@github.com:datawagovau/ckanext-harvest.git

git clone https://github.com/ckan/ckanext-archiver.git
git clone https://github.com/datagovau/ckanext-cesiumpreview.git
git clone https://github.com/ckan/ckanext-dcat.git
git clone https://github.com/ckan/ckanext-disqus.git
git clone https://github.com/NaturalHistoryMuseum/ckanext-doi.git
git clone https://github.com/datacats/ckanext-featuredviews.git
#git clone https://github.com/DataShades/ckan-galleries.git
git clone https://github.com/ckan/ckanext-geoview.git
git clone https://github.com/datacats/ckanext-geopusher.git
git clone https://github.com/datagovuk/ckanext-hierarchy.git
git clone https://github.com/ckan/ckanext-pdfview.git
git clone https://github.com/ckan/ckanext-qa.git
git clone https://github.com/ckan/ckanext-showcase.git

git clone https://github.com/ckan/ckanapi.git
git clone https://github.com/geopython/pycsw.git

# pycsw dependencies
sudo apt-get install -y python-dev libxml2-dev libxslt-dev libgeos-dev
```

2.5.2 Manage dependency conflicts

Before running through this section, note that dependency conflicts are caused by multiple independently developed code bases of ckan and its plugins. Each code base pins third party library versions known to work at the time of release. Naturally, the most established extensions, e.g. spatial and harvesting, have the oldest dependencies, while brand new extensions, e.g. agls, require much newer libraries.

Note: currently, the setup works without this section.

Review possible collisions at http://rshiny.yes-we-ckan.org/ckan-pip-collisions/. Note, the following example lists dependencies current as of October 2015 and will outdate quickly. We recommend to research your own version

conflicts and use this example as a how-to guide, but with your own dependencies. In our example the following packages have differing, hard-coded requirements:

```
grep -rn --include="*requirements*" 'requests' .
grep -rn --include="*requirements*" 'six' .
grep -rn --include="*requirements*" 'lxml' .
grep -rn --include="*requirements*" 'python-dateutil' .
grep -rn --include="*requirements*" 'SQLAlchemy' .
```

We'll need to update all colliding requirement versions to one that works across all extensions. In our case, a simple bump to the highest mentioned version will work, such as with the perfectly backwards compatible `requests` library. In other cases, breaking changes between different dependency versions could require an upgrade to an actual extension.

Batch-modify version numbers as shown here work on our listed extensions at the time of writing. Modify to your actual needs. Warning - a mistake in this step could corrupt your installed code (including CKAN source), requiring to `git checkout` incorrectly modified files in each repo.:

```
grep -rl --include="*requirements*" 'requests' . | xargs sed -i 's/^.*requests.*$/requests==2.7.0/g'
grep -rl --include="*requirements*" 'six' . | xargs sed -i 's/^.*six^.*/six==1.9.0/g'
grep -rl --include="*requirements*" 'lxml' . | xargs sed -i 's/^.*lxml^.*/lxml==3.4.4/g'
grep -rl --include="*requirements*" 'python-dateutil' . | xargs sed -i 's/^.*python-dateutil^.*/pytho
grep -rl --include="*requirements*" 'SQLAlchemy' . | xargs sed -i 's/^.*SQLAlchemy.*$/SQLAlchemy==0.9

# review version numbers
grep -rn --include="*requirements*" 'requests' .
grep -rn --include="*requirements*" 'six' .
grep -rn --include="*requirements*" 'lxml' .
grep -rn --include="*requirements*" 'python-dateutil' .

# any other requirements conflicts?
cat `find . -name '*requirements*'` | sort | uniq
```

To fix issues with any dependency versions:

```
datacats shell
pip freeze | grep lchemy
pip install SQLAlchemy==0.9.6
exit
```

E.g., this is necessary when receiving this error on datacats reload:

```
File "/usr/lib/ckan/local/lib/python2.7/site-packages/geoalchemy2/comparator.py", line 52, in <module
class BaseComparator(UserDefinedType.Comparator):
AttributeError: type object 'UserDefinedType' has no attribute 'Comparator'
Starting subprocess with file monitor
```

2.5.3 Install extensions

To install all extensions and their dependencies in the site's environment, run:

```
datacats install
```

2.6 Modify datacats containers

Some extensions require modifications to the database, or additional servers, such as a message queue (redis) or a task runner (celery). Following ckanext-spatial docs and ckanext-harvest docs with datacats' paster command:

```
# (re)install postgis, add redis
datacats tweak --install-postgis
datacats tweak --add-redis
# datacats tweak --add-pycsw # soon
datacats reload
# pulls redis image

# initdb for spatial
cd ckanext-spatial
datacats paster spatial initdb
cd ..

# initdb for harvester, plus two celery containers, see also below
cd ckanext-harvest
datacats paster harvester initdb
datacats paster -d harvester gather_consumer
datacats paster -d harvester fetch_consumer
cd ..
```

Note: `git init` the theme extension (ckanext-SITEtheme) to preserve significant customisations.

2.7 Config

General procedure:

- Edit config *vim development.ini*, replace everything from "Authorization Settings" with settings below.
- Apply changes with *datacats reload*. That should be it!

`development.ini`:

```
## Authorization Settings
 ckan.auth.anon_create_dataset = false
 ckan.auth.create_unowned_dataset = false
 ckan.auth.create_dataset_if_not_in_organization = false
 ckan.auth.user_create_groups = true
 ckan.auth.user_create_organizations = false
 ckan.auth.user_delete_groups = true
 ckan.auth.user_delete_organizations = false
 ckan.auth.create_user_via_api = true
 ckan.auth.create_user_via_web = true
 ckan.auth.roles_that_cascade_to_sub_groups = admin editor member

 ## Search Settings
 ckan.site_id = default
 solr_url = http://solr:8080/solr

 ## CORS Settings
 ckan.cors.origin_allow_all = true

 ## Plugins Settings
 base = cesium_viewer resource_proxy datastore datapusher datawagovau_theme stats archiver qa feature
```

```
sch = scheming_datasets
rcl = recline_grid_view recline_graph_view recline_map_view
prv = text_view image_view recline_view pdf_view webpage_view
geo = geo_view geojson_view
spt = spatial_metadata spatial_query geopusher
hie = hierarchy_display hierarchy_form
dcat = dcat dcat_rdf_harvester dcat_json_harvester dcat_json_interface
hrv = harvest ckan_harvester csw_harvester
pkg = datapackager downloadtdf
ckan.plugins = %(base)s %(sch)s %(rcl)s %(prv)s %(dcat)s %(geo)s %(spt)s %(hrv)s %(hie)s
#%(pkg)s ## missing ckan branch datapackager

ckanext.geoview.ol_viewer.formats = wms wfs gml kml arcgis_rest gft
ckan.views.default_views = cesium_view %(prv)s geojson_view


# ckanext-scheming
scheming.dataset_schemas = ckanext.datawagovautheme:datawagovau_dataset.json
#scheming.organization_schemas = ckanext.datawagovautheme:datawagovau_organization.json

# ckanext-harvest
ckan.harvest.mq.type = redis
ckan.harvest.mq.hostname = redis
ckanext.spatial.harvest.continue_on_validation_errors= True

# ckanext-pages
ckanext.pages.organization = True
ckanext.pages.group = True
# disable to make space for static pages:
ckanext.pages.about_menu = True
ckanext.pages.group_menu = True
ckanext.pages.organization_menu = True

# ckanext-disqus
# add Engage to site > add a subaccount to your disqus account for this CKAN
# choose name = disqus.name
# settings > advanced >
# add %(site_url)s to trusted domains, e.g. catalogue.beta.data.wag.gov.au
disqus.name = xxxx

## Front-End Settings
ckan.site_title = Parks & Wildlife Data
ckan.site_logo = /logo.png
ckan.site_description =
ckan.favicon = /favicon.ico
ckan.gravatar_default = identicon
ckan.preview.direct = png jpg gif
ckan.preview.loadable = html htm rdf+xml owl+xml xml n3 n-triples turtle plain atom csv tsv rss txt
ckan.display_timezone = server
# package_hide_extras = for_search_index_only
#package_edit_return_url = http://another.frontend/dataset/<NAME>
#package_new_return_url = http://another.frontend/dataset/<NAME>
#licenses_group_url = http://licenses.opendefinition.org/licenses/groups/ckan.json
# ckan.template_footer_end =
ckan.recaptcha.version = 1
ckan.recaptcha.publickey = xxxx
ckan.recaptcha.privatekey = xxxx
```

```
## Internationalisation Settings
ckan.locale_default = en_AU
ckan.locale_order = en_AU pt_BR ja it cs_CZ ca es fr el sv sr sr@latin no sk fi ru de pl nl bg ko_KR
ckan.locales_offered =
ckan.locales_filtered_out = en_GB

## Feeds Settings
ckan.feeds.authority_name =
ckan.feeds.date =
ckan.feeds.author_name =
ckan.feeds.author_link =

## Storage Settings
ckan.storage_path = /var/www/storage
#ckan.max_resource_size = 10

## Datapusher settings
# Make sure you have set up the DataStore
ckan.datapusher.formats = csv xls xlsx tsv application/csv application/vnd.ms-excel application/vnd
ckan.datapusher.url = http://datapusher:8800

# Resource Proxy settings
ckan.max_resource_size = 1000000
ckan.max_image_size = 200000
ckan.resource_proxy.max_file_size = 31457280

## Activity Streams Settings
ckan.activity_streams_enabled = true
ckan.activity_list_limit = 31
#ckan.activity_streams_email_notifications = true
#ckan.email_notifications_since = 2 days
ckan.hide_activity_from_users = %(ckan.site_id)s

## Email settings
email_to = xxxx
error_email_from = xxxx
smtp.server = smtp.gmail.com:587
smtp.starttls = True
smtp.user = xxxx
smtp.password = xxxx
smtp.mail_from = xxxx

## Logging configuration
[loggers]
keys = root, ckan, ckanext
[handlers]
keys = console
[formatters]
keys = generic
[logger_root]
level = WARNING
handlers = console
[logger_ckan]
level = INFO
handlers = console
qualname = ckan
propagate = 0
[logger_ckanext]
```

```
level = INFO
handlers = console
qualname = ckanext
propagate = 0
[handler_console]
class = StreamHandler
args = (sys.stderr,)
level = NOTSET
formatter = generic
[formatter_generic]
format = %(asctime)s %(levelname)-5.5s [%(name)s] %(message)s
```

2.8 PyCSW

While our contribution is in development, we'll manually build and run a dockerised pycsw using our datacats fork:

```
cd /var/projects/ckan/datacats/docker/pycsw/
docker build -t datacats/pycsw .
docker run -d -p 9000:8000 -it datacats/pycsw python /var/www/pycsw/csw.wsgi
```

This will build a pycsw server image with harvesting enabled (transactions) for non-local IPs and run a pycsw server on localhost:9000. See also nginx settings in Deployment to expose the csw server publicly.

CHAPTER 3

Disaster Recovery

3.1 Examples

The following examples are taken from Ian Ward's talk at the CKANConf Ottawa 2015 as published on his blog. Exporting data (backup), restoring to same (disaster recovery) or remote (migration) instance:

```
ckanapi dump groups | ssh otherbox ckanapi load groups -p 3
ckanapi dump organizations | ssh otherbox ckanapi load organizations -p 3
ckanapi dump datasets | ssh otherbox ckanapi load datasets -p 3
```

Or pulling data from remote into local CKAN:

```
ckanapi dump datasets -r http://sourceckan | ckanapi load datasets -p 3
```

Track metadata using git:

```
ckanapi dump datasets > datasets.jsonl
git diff datasets.jsonl --stat
datasets.jsonl | 52 ++++++++++++++++++++++++++++++++++++++++++++++++++++----------------
1 file changed, 36 insertions(+), 16 deletions(-)
```

Summaries:

```
head -5 datasets.jsonl | jq .title
jq 'select(.organization.name!="nrcan-rncan")' -c datasets.jsonl | wc -l
```

Distributed loading:

```
split -n l/3 datasets.jsonl part
ckanapi load datasets -r http://web1 -a ... < partaa &
ckanapi load datasets -r http://web2 -a ... < partab &
ckanapi load datasets -r http://web3 -a ... < partac &
```

Create a database snapshot from inside a datacats environment:

```
(ckan)ubuntu@ip:/var/projects/ckan/SOURCE$ datacats paster db dump ../snapshot-DATE.sql
```

Notes: * SOURCE is the datacats CKAN environment to be backed up * DATE is the current datetime * datacats paster db will run in the /ckan source directory, so the prefix `../` will create the file in the current directory

3.2 Backup & Restore

3.2.1 Users

Ckanapi does not export users with password hashes. We'll have to migrate existin users manually.:

```
sudo su
sudo -u postgres pg_dump -a -O -t user -f user.sql ckan_private

# chown to datacats user and appropriate group
sudo chown ubuntu:www-data user.sql

# delete lines "default", "logged_in", "visitor" from user.sql

# print passwords in datacats env:
cat ~/.datacats/ENV_NAME/sites/primary/passwords.ini

(datacats venv)me@machine:/path/to/ENV_NAME datacats shell

#datacats shell
(ckan)shell@ENV_NAME:~$ psql -h db -d ckan -U postgres -f user.sql
# paste password for user "postgres"

(ckan)shell@ENV_NAME:~$ exit
```

3.2.2 Groups, orgs, datasets

Run after migrating users:

```
ckanapi dump groups --all -p 3 -r http://source-ckan.org > groups.jsonl
ckanapi dump organizations --all -p 3 -r http://source-ckan.org > organizations.jsonl
ckanapi dump datasets --all -p 6 -r http://source-ckan.org > datasets.jsonl

ckanapi load groups -p 3 -I groups.jsonl -r http://target-ckan.org -a API_KEY
ckanapi load organizations -p 3 -I organizations.jsonl -r http://target-ckan.org -a API_KEY
ckanapi load datasets -p 3 -I datasets.jsonl -r http://target-ckan.org -a API_KEY
```

3.2.3 Files

Run any time:

```
sudo rsync -Pavvr DATACATS_DATADIR/SOURCE/sites/primary/files/ DATACATS_DATADIR/TARGET/sites/primary/
sudo chown -R ubuntu:www-data DATACATS_DATADIR/TARGET/sites/primary/files/
```

- The DATACATS_DATADIR/ defaults to `~/.datacats`, but in our installation lives at `/var/projects/datacats_data/`
- SOURCE and TARGET are two datacats CKAN environments
- `datacats paster` runs inside the datacats environment with prefix `/project/` for the current work dir
- http://target-ckan.org has an admin API_KEY

3.3 Archive

Create a scheduled job to dump the db as well as the users, groups, orgs, datasets and rsync the dumps as well as the files (resource storage) to a secure place.

CHAPTER 4

Operations

The following sections illustrate operational use of extensions after successful installation.

4.1 Quality Assurance (ckanext-qa)

Create a supervisor script:

```
# run the celery daemon (stops with each datacats reload)
datacats paster -d celeryd

# run the qa update
cd ckanext-qa
datacats paster qa update
cd ..
```

4.2 Geo-referencing a dataset (ckanext-spatial)

Watch this video to get started.

4.3 Adding a static page (ckanext-pages)

Click on the pages icon, add content, save.

4.4 CKAN to CKAN Harvesting (ckanext-harvest)

If there's no custom ckanext-scheming metadata schema enabled, plain CKAN harvesting will work.

Create a harvest source at `/harvest` with

- url `http://SOURCE.yes-we-ckan.org`,
- source type `CKAN`,
- update frequency `always`,
- configuration:

```
{"api_version": 1,
```

"default_extras":{"url":"{harvest_source_url}/dataset/{dataset_id}"}, "override_extras": "false", "user":"local-sysadmin-username", "read_only": true, "remote_groups": "create", "remote_orgs": "create", "clean_tags":"true", "override_extras": true}

- save.

On the shell inside the ckanext-harvest extension's folder:

```
ckanext-harvest$ datacats paster -d harvester gather_consumer
ckanext-harvest$ datacats paster -d harvester fetch_consumer
ckanext-harvest$ datacats paster harvester job-all
ckanext-harvest$ datacats paster harvester run
```

This will run the two gather/fetch-consumer celery containers in the background (-d flag), reharvest the source (job-all does the same as clicking "reharvest" on every data source), and run the harvest job.

Note: The two daemonised consumer containers will have to be restarted manually after each `datacats reload`.

Known problem: The source and target CKAN instances must have the exact same schema, either the default, or the exact same custom schema. Otherwise, the "extra" fields will collide with ckanext-scheming's "extra" fields, with an error such as:

nature-reserves-proposals-beeliar-wetlands-park-sw-corridor-wl19850899

Invalid package with GUID nature-reserves-proposals-beeliar-wetlands-park-sw-corridor-wl19850899: {'extras': [{'key': ['There is a schema field with the same name']}, {'key': ['There is a schema field with the same name']}]}

Notably, a custom extra field "spatial", or a scheming field "spatial" will also not work.

4.5 Harvesting between catalogues with custom CKAN dataset schemas

Note: This section describes (only partly implemented) functionality under development.

Custom metadata schemas via ckanext-scheming can be harvested using our fork of ckanext-harvest.

Create a harvesting job, e.g. on catalogue.alpha.data.wa.gov.au:

- Url http://landgate.alpha.data.wa.gov.au/
- Source type: CKAN (we overwrote the ckanharvester)
- Update frequency: manual
- Configuration: (for ckanext-datawagovautheme:datawagovau_dataset.json):

```
{
```

"api_version": 1, "user":"florianm", "remote_groups": "create", "remote_orgs": "create", "clean_tags":false, "force_all":true, "disable_extras":true, "default_fields":{

"data_portal":"{harvest_source_url}", "landing_page":"{harvest_source_url}/dataset/{dataset_id}"

}, "field_mapping":{ "doi": "doi", "citation": "citation", "published_on": "published_on", "last_updated_on": "last_updated_on", "update_frequency": "update_frequency", "data_temporal_extent_begin": "data_temporal_extent_begin", "data_temporal_extent_end": "data_temporal_extent_end", "spatial": "spatial"

```
        }
    }
```

- Organization: landgate (exists on catalogue.alpha)

Then run inside the ckanext-harvest extension folder:

```
(ckan)ubuntu@ip:/var/projects/ckan/datawagovau/ckanext-harvest$
datacats paster -d harvester gather_consumer
datacats paster -d harvester fetch_consumer
datacats paster harvester job-all
datacats paster harvester run
```

- Every `datacats reload` requires a restart of the gather and fetch consumer.
- `job-all` is the shortcut for clicking "Reharvest" on all harvest sources.
- `run` starts all pending jobs, and refreshes the status of running jobs.

4.6 Harvesting WMS

We will harvest metadata from a WMS GetCapabilities statement into a local pycsw server, then harvest that pycsw server into CKAN using the CSW harvester.

Create a file *test-wms.xml* for the official pycsw example WMS:

```
<?xml version="1.0" encoding="UTF-8"?>
<Harvest
  xmlns="http://www.opengis.net/cat/csw/2.0.2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 http://schemas.opengis.net/csw/2.0.2/CSW-p
  service="CSW" version="2.0.2">
  <Source>http://webservices.nationalatlas.gov/wms/1million</Source>
  <ResourceType>http://www.opengis.net/wms</ResourceType>
  <ResourceFormat>application/xml</ResourceFormat>
</Harvest>
```

Landgate's SLIP Classic WMS, e.g. *slip_classic.xml*:

```
<?xml version="1.0" encoding="UTF-8"?>
<Harvest xmlns="http://www.opengis.net/cat/csw/2.0.2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 http://schemas.opengis.net/csw/2.0.2/CSW-p
  service="CSW" version="2.0.2">
  <Source>http://srss-dev.landgate.wa.gov.au/slip/wmspublic.php</Source>
  <ResourceType>http://www.opengis.net/wms</ResourceType>
  <ResourceFormat>application/xml</ResourceFormat>
</Harvest>
```

In the pycsw folder, run:

```
(datacats)ubuntu@ip:/var/projects/datacats/public/pycsw$
python bin/pycsw-admin.py -c post_xml -u http://localhost:9000/pycsw/csw.py -x test-wms.xml
python bin/pycsw-admin.py -c post_xml -u http://localhost:9000/pycsw/csw.py -x slip_classic.xml
```

to receive:

```
Initializing static context
Executing HTTP POST request test-wms.xml on server http://localhost:8086/pycsw/csw.py
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- pycsw 2.0-dev -->
<csw:HarvestResponse xmlns:csw30="http://www.opengis.net/cat/csw/3.0" xmlns:fes20="http://www.opengis
Done
```

In CKAN, create a harvest source (e.g. through the GUI at `/harvest`) following the harvest docs:

- url: http://pycsw.beta.data.wa.gov.au
- Source type: CSW Server
- Update frequency: Daily (creates a new harvest job at given frequency, which has to be run by *datacats paster harvester run*)
- Configuration:

```
{"default_tags":["wms_harvested"],
```

"clean_tags":true, "default_groups":["test"], "remote_groups":"create", "override_extras": false, "read_only":true, "force_all":true}

Notes on configuration:

- `clean_tags` makes tags url-safe
- `override_extras` does not influence whether remote extras will be created locally (which will fail if ckanext-scheming is installed)
- `force_all` will update even unchanged datasets

4.7 Custom WMS harvesting

The following example is a real-life use case of harvesting GeoServer 1.8 WMS/WFS endpoints into our customised dataset schema `datawagovau_dataset.json`.

Challenges:

- Custom WMS with authentication for publicly available layers (you right that read)
- We harvest a proxy which provides our (secret) credentials to access the public WMS/WFS layers
- Custom conventions for extracting groups and organizations from WMS and from assumptions
- Custom methods to extract dataset title, dataset ID and publication date from WMS layer name

This harvester contains too many unique assumptions and requires context, and the WMS endpoint will be superseded soon. Therefore, it is implemented as iPython Notebook, with the functions and operations cleanly separated. This brings as benefits:

- faster turn-around (Shift-Enter) than customising extension and going through redis queues
- can still be refactored into an extension when appropriate
- self-documenting
- separates confidential credentials into separate config file
- shapshots online at http://catalogue.alpha.data.wa.gov.au/dataset/slip-harvesting
- follows harvester logic of building dicts from harvested source, and uploading dicts to CKAN

The following sections contain work in progress on not yet working extensions.

4.8 Mint DOIs (ckanext-doi)

- Fork ckanext-doi
- Register with ANDS to mint DOI at http://ands.org.au/services/cmd-registration.html
- Customise ckanext-doi to support both (existing) Datacite and (add) ANDS minting
- CKAN config parameters

4.9 Media gallery (ckan-galleries)

Add `dfmp` to plugins. Results:

- dfmp theme overrides datacats theme (undesirable, submitted as ckan-galleries #5 at https://github.com/DataShades/ckan-galleries/issues/5)
- one of the layout options crashes ckan
- ‘‘import‘‘ing a flickr pool does not seem to have any effect (needs investigation)
- twitter account is hard-coded and blocked due to over-use

Will use as soon as bugs are fixed upstream.

4.10 Harvesting WMS notes

- VicRoads AGO http://vicroadsopendata.vicroadsmaps.opendata.arcgis.com/data.json
- Dcat example https://raw.githubusercontent.com/ckan/ckanext-dcat/master/examples/dataset.json
- Meteoswiss harvester example https://github.com/ogdch/ckanext-meteoswiss

Setup ckanext-spatial spatial harvesters:

```
cd ckanext-spatial
datacats paster ckan-pycsw setup -p ../pycsw.cfg
cd ..

cd pycsw
python csw.wsgi &
cd ..


http://127.0.0.1:PORT/?service=CSW&version=2.0.2&request=GetCapabilities
```

[pycsw docs](http://geopython.github.io/pycsw-workshop/docs/intro/intro-exercises.html#metadata-harvesting)

Author: Keith Moss, Landgate WA:

```
python pycsw/bin/pycsw-admin.py -c post_xml -u http://localhost:8000/pycsw/csw.py -x pycsw/post.xml

# Woo! PyCSW harvested the example service!

# Got this error from lxml:
# http://docs.ckan.org/projects/ckanext-spatial/en/latest/install.html#when-running-the-spatial-harve
```

```
# So, continue_on_validation_errors wasn't being picked up from production.ini

vim ckanext-spatial/ckanext/spatial/harvesters/base.py
# Just commented out the block that aborts


# Harvesting SLIP Classic
nano pycsw/post-slip-classic.xml

python pycsw/bin/pycsw-admin.py -c post_xml -u http://localhost:8000/pycsw/csw.py -x pycsw/post-slip-

curl -x https://USER:PASS@www2.landgate.wa.gov.au/ows/wmspublic -X POST -d post-slip-classic.xml http

vim pycsw/post-firewatch.xml
python pycsw/bin/pycsw-admin.py -c post_xml -u http://localhost:8000/pycsw/csw.py -x pycsw/post-firew

# List records
# http://127.0.0.1:PORT/?request=GetRecords&service=CSW&version=2.0.2&resultType=results&outputSchema

# Get a record
# http://127.0.0.1:PORT/?outputFormat=application%2Fxml&service=CSW&outputSchema=http%3A%2F%2Fwww.iso

# Dumb reverse proxy to Classic to work around the "pycsw doesn't do secured services" thing
# http://gis.stackexchange.com/questions/103191/getting-error-after-trying-to-harvest-from-geoserver-
python pycsw/bin/pycsw-admin.py -c post_xml -u http://localhost:PORT/pycsw/csw.py -x pycsw/post-slip-
```

CSW ISO19139 XML validation issues (pyCSW?) https://github.com/ngds/ckanext-ngds/issues/442 CSIRO on CKAN harvesting https://www.seegrid.csiro.au/wiki/Infosrvices/CKANHarvestingGuide

4.11 data.wa.gov.au theming extension

This section documents the additions we made to the theming extension beyond the obvious CSS overrides. If you have installed the extension, the following steps are already done.

First, we copied a few templates we needed to override:

```
mkdir -p ckanext-datawagovautheme/ckanext/datawagovautheme/templates/package/
mkdir -p ckanext-publictheme/ckanext/datawagovautheme/templates/organization/
cp ckan/ckan/templates/organization/read_base.html ckanext-datawagovautheme/ckanext/datawagovautheme/
cp ckan/ckan/templates/package/search.html ckanext-datawagovautheme/ckanext/datawagovautheme/template
cp ckan/ckan/templates/package/read_base.html ckanext-datawagovautheme/ckanext/datawagovautheme/templ
```

Fix CKAN organisation activity stream bug (now fixed upstream):

- `vim cckanext-datawagovautheme/ckanext/datawagovautheme/templates/organization/read_bas`
- Add "offset=0" to as per ckan/ckan#2466 https://github.com/ckan/ckan/issues/2466:

```
    {{ h.build_nav_icon('organization_activity', _('Activity Stream'), id=c.group_dict.name, offset=
```

Add spatial search (ckanext-spatial):

- Enable the spatial search widget http://docs.ckan.org/projects/ckanext-spatial/en/latest/spatial-search.html#spatial-search-widget:
- `vim ckanext-datawagovautheme/ckanext/datawagovautheme/templates/package/search.html`

Add to block `secondary_content`:

```
{% snippet "spatial/snippets/spatial_query.html", default_extent="[[-40,110], [0, 130]]" %}
```

CSS fixes should be obsolete once our spatial widget (with fixed libraries and CSS) gets merged. To render the dataset search icon visible, add to the CKAN CSS:

```
.leaflet-draw-draw-rectangle, .leaflet-draw-draw-polygon {height:22px; width:22px;}
```

Add spatial preview (ckanext-spatial):

- If *not* using the custom dataset schema (which we are in this example,

enable the dataset extent map: * `vim ckanext-datawagovautheme/ckanext/datawagovautheme/templates/packa`
* Add to block `secondary_content`:

```
{% set dataset_extent = h.get_pkg_dict_extra(c.pkg_dict, 'spatial', '') %}
{% if dataset_extent %}
  {% snippet "spatial/snippets/dataset_map_sidebar.html", extent=dataset_extent %}
{% endif %}
```

4.12 Contribute to Datacats

This section is not part of the setup workflow and can be skipped.

4.12.1 Update a local docker image

To update a datacats image locally, as reported at https://github.com/datacats/datacats/issues/210:

```
$ docker run -datacats/web apt-get install -y whatever
$ docker ps -lq
6f51fba7febb
$ docker commit 6f51fba7febb datacats/web
```

This image will be used until you do a datacats pull the next time. You can do run/commit as many times as you'd like, but if you're making a lot of changes you'll want to change the actual Dockerfiles in the source and rebuild them:

```
cd datacats/docker
docker build -t datacats/web .
```

4.12.2 Report a bug

Submit working additions as a new datacats issue at https://github.com/datacats/datacats/issues/new.

4.12.3 Install extensions selectively

`datacats install` installs all downloaded extensions in an environment directory. To install an extension individually, run:

```
datacats shell
cd ckanext-EXTENSION
python setup.py develop
exit
```

This will install the extension into the datacats environment. Running `python setup.py develop` outside the datacats shell will not install the extension into the datacats environment.

CHAPTER 5

Data Science Workbench

5.1 RStudio Server

Follow the rocker-rstudio wiki at https://github.com/rocker-org/rocker/wiki/Using-the-RStudio-image to run the RStudio Server image:

```
mkdir -p /var/projects/rstudio-server
docker run -d \
-p 8787:8787 \
-e USER=<username> \
-e PASSWORD=<password> \
-v /var/projects/rstudio-server/:/home/ \
rocker/ropensci
```

Login to [RStudio Server]() with above set username and password and install a few packages:

```
install.packages(c('shinyIncubator','markdown','whisker','Hmisc','qcc','httr','RCurl','curl','devtool
devtools::install_github("ropensci/ckanr")
ckanr::ckanr_setup(url="http://catalogue-beta.data.wa.gov.au/")
```

Follow rocker-rstudio's instructions at https://github.com/rocker-org/rocker/wiki/Using-the-RStudio-image#multiple-users to add new users for your (public-facing) RStudio Server.

5.2 RShiny Server

Following the rocker-rshiny wiki https://github.com/rocker-org/shiny:

```
mkdir -p /var/projects/shiny-logs
mkdir -p /var/projects/shiny-apps

docker run -d \
-p 3838:3838 \
-v /var/projects/shiny-apps/:/srv/shiny-server/ \
-v /var/projects/shiny-logs/:/var/log/ \
-v /usr/local/lib/R/site-library/:/usr/local/lib/R/site-library/ \
rocker/shiny

cd /var/projects/shiny-apps
git clone git@github.com:florianm/ckan-pip-collisions.git
```

- Add shiny apps into `/var/projects/shiny-server`.

- Find logs from rocker-rshiny in `/var/projects/shiny-logs`.
- Install R packages as required by your rshiny apps into `/usr/local/lib/R/site-library`:

  sudo su apt-get -y install r-base r-base-dev gdebi-core texlive-full htop ncdu libcurl4-openssl-dev libxml2-dev

  sudo su - -c "R -e "install.packages(c('bitops','caTools','colorspace','RColorBrewer','scales','ggplot2','rjson','markdown','knitr',
  'shiny','shinyIncubator','markdown','whisker','Hmisc','dplyr','tidyr','lubridate','qcc','httr','RCurl','curl','devtools'),
  repos='http://cran.rstudio.com/');devtools::install_github('ropensci/ckanr')""

  # or as root in an R session: shopping_list = c('bitops','caTools','colorspace','RColorBrewer','scales','ggplot2','rjson','markdown'
  'shiny','shinyIncubator','markdown','whisker','Hmisc','dplyr','tidyr','lubridate','qcc','httr','RCurl','curl','devtools')
  install.packages(shopping_list, repos='http://cran.rstudio.com/') devtools::install_github('ropensci/ckanr')

CKAN-o-Sweave Using Rstudio Server, follow https://github.com/datawagovau/ckan-o-sweave to fork yourself a nice cold CKAN-o-Sweave. Read the instructions in the README to get started, and read the example report for an in-depth explanation.

Following the RShiny Server docs at https://support.rstudio.com/hc/en-us/articles/200552326-Configuring-the-Server/, add to `/etc/nginx/sites-enabled/base.conf` (substituting DOMAIN.TLD with your domain):

```
server {
  server_name rshiny.DOMAIN.TLD;
  listen 80;
  location / {
    proxy_pass http://127.0.0.1:3838;
  }
}
server {
  server_name rstudio.DOMAIN.TLD;
  listen 80;
  location / {
    proxy_pass http://127.0.0.1:8787;
    proxy_redirect http://127.0.0.1:8787/ $scheme://$host/;
  }
}
```

5.3 OpenRefine

Using https://github.com/SpazioDati/docker-openrefine 's container:

```
mkdir -p /var/projects/openrefine
docker run -d -p 3333:3333 -v /var/projects/openrefine/:/mnt/refine/ spaziodati/openrefine
```

nginx conf:

```
server {
  server_name openrefine.DOMAIN.TLD;
  listen 80;
  location / {
    proxy_pass http://127.0.0.1:3333;
    proxy_redirect http://127.0.0.1:3333/ $scheme://$host/;
  }
}
```

Integration suggestion:

- CSV and XLS resources menu: "OpenRefine" link
- Use OpenRefine API to create new project at openrefine.beta.data.wa.gov.au using resource url

Or OpenRefine View: create OR projects for all data resources with `paster views openrefine` and persist OR urls in resource view

5.4 IPython Notebook Server

Using https://github.com/ipython/docker-notebook/tree/master/scipyserver or https://github.com/jupyter/docker-demo-images:

```
mkdir -p /var/projects/ipython
docker run -d -p 8888:8888 -e "PASSWORD=MakeAPassword" -e "USE_HTTP=1" -v /var/projects/ipython/:/not
```

Current issue: neither connect to kernel. Investigate Beaker notebook.

Homework: https://ipython.org/ipython-doc/1/interactive/public_server.html

Future developments: spawn github-authenticated single-user servers on demand using

- https://github.com/jupyter/dockerspawner
- https://github.com/jupyter/oauthenticator
- https://github.com/jupyter/jupyterhub

Or use hosted service like https://www.dominodatalab.com/

5.5 Quantum GIS Server

https://github.com/opengisch/docker-qgis-server-webclient Run:

```
mkdir /var/projects/qgis
docker pull opengisch/qgis-server-webclient
docker run -v /var/projects/qgis:/web -p 8100:80 -d -t opengisch/qgis-server-webclient
```

nginx conf:

```
server {
  server_name qgis.DOMAIN.TLD;
  listen 80;
  location / {
    proxy_pass http://127.0.0.1:8100;
    proxy_redirect http://127.0.0.1:8100/ $scheme://$host/;
  }
}
```

5.6 QGIS plugin idea: CKAN dataset shopping basket

- CKAN API > filter by file type (WMS, WFS etc) > extract URLs and metadata > create .qgis project file
- Place .qgis project file into */var/projects/qgis* which the qgis docker image mounts.
- Follow https://github.com/opengisch/docker-qgis-server-webclient.

5.7 Taverna

Pull and run the Taverna Server Docker image:

```
docker pull taverna/taverna-server
sudo docker run -p 8080:8080 -d taverna/taverna-server
```

5.8 Map

This section will setup a local copy of NationalMap, which will come in handy for running behind firewalls, where the official NationalMap can't access datasets for preview.

1. Clone, install and run NationalMap as per NationalMap docs:

   sudo apt-get install -y git-core gdal-bin

   curl -sL https://deb.nodesource.com/setup_0.12 | sudo bash -

   sudo apt-get install -y nodejs

   sudo npm install -g gulp

   /var/projects/$ clone https://github.com/NICTA/nationalmap.git

   /var/projects/nationalmap/$ sudo npm install

   /var/projects/nationalmap/$ gulp

2. Create a supervisord config `/etc/supervisor/conf.d/nmap.conf`:

   [program:nmap]

   user=www-data

   stopasgroup=true

   autostart=true

   autorestart=true

   directory=/mnt/projects/nationalmap

   command=/usr/bin/npm start

3. Configure local data sources as required, e.g. DPaW sources

4. Start NationalMap

   sudo supervisorctl stop nmap

   # modify datasources/xx.json

   /var/projects/nationalmap/$ gulp

   gulp && sudo supervisorctl start nmap

CHAPTER 6

GovHack 2015

This project started out as one of several products entered by team *Bangers n' Mashup* for their project *yes we CKAN* at the GovHack 2015:

- The data portal demo open-data.yes-we-ckan.org,
- a data science workbench (an R Studio Server) rstudio.yes-we-ckan.org,
- an interactive application server (an R Shiny Server) rshiny.yes-we-ckan.org, running
- an example app timeseries explorer (ts code) visualizing open government data (one example dataset),
- and a whirlwind video tour of the whole stack.

6.1 Disclaimer

- The timeseries explorer was created before the GovHack and serves only as an illustration of an interactive, data-driven application to bridge the gap between raw data and meaningful visualisation.
- The datacats CKAN install was created after the GovHack, as a critical bug with a Docker version rendered datacats unusable during the contest, but was fixed one day afterwards.
- This documentation (submitted **'version'**_) was hosted on readthedocs.org after the contest and is under active development.
- The video was modified after the contest: the audio was shifted by ca. 200ms to counter the audio lag introduced by the hosting on vimeo.com, and a few words have been edited in the introductory slides. However, the general content and message have not been altered.

6.2 Details

The installation documented in *Alternatives* demonstrates the most low-level hands-on way of installing a CKAN data catalogue, an installation from source. There are several quicker, but less customisable ways:

- installation from package,
- installation from Docker image,
- using PaaS services like http://www.datacats.com/ or CKAN Galvanize http://datashades.com/ckan-galvanize/our-ckan-managed-platform/,
- contracting a third party to implement any of the above methods.

Our approach hopes to demonstrate our lessons learnt:

- It is possible to setup and host a working CKAN from scratch within a day.
- A source install, while not advisable for production use, allows to fix bugs and add new functionality, and contribute these improvements back to the CKAN community.
- We demonstrate how to scale the install to include and host other useful servers like R Studio Desktop and R Shiny Server.
- We include a scalable way to host a multi-tenant CKAN install following the WA Department of Parks and Wildlife's working multi-tenant setup. The description of the multi-tenant install was created before the GovHack and is not to be considered for judging. However, we hope it provides value for real-world use past the GovHack event.

6.3 Future directions

This repository will be updated as work progresses on the CKAN installation. We plan to include installation examples using Datacats (plus Docker-based installs of R Studio Server and R Shiny Server) as well as Datashades' CKAN Galvanize.

CHAPTER 7

CKAN Source install

This chapter provides instructions to setup a CKAN instance from source inside an existing VM with special attention to managing storage needs. An alternative installation is based on datacats / Docker. The datacats method is preferable over a source install for production use, while the source install allows more flexibility, e.g. to use an existing database or SolR instance.

Generally, it follows the CKAN source install at http://docs.ckan.org/en/latest/maintaining/installing/install-from-source.html and the Department of Parks and Wildlife's multi-tenant CKAN, see https://twitter.com/opendata/status/555760171017056256.

7.1 Install system dependencies

pip install virtualenvwrapper apt-get install python-dev postgresql libpq-dev python-pip python-virtualenv git-core openjdk-7-jdk virtualenvwrapper python-pastescript build-essential git redis-server postgresql-9.3-postgis-2.1 libxml2-dev libxslt1-dev libgeos-c1 libjts-java apache2 libapache2-mod-wsgi libapache2-mod-rpaf unzip

Have virtualenvwrapper installed.

7.2 virtualenv

As non-root user:

```
mkproject ckan
# with virtualenv "ckan" activated = workon ckan
mkdir -p /var/projects/ckan/src
cd /var/projects/ckan/src
```

7.3 Clone CKAN code from custom fork

add ssh public key to github profile git clone git@github.com:florianm/ckan.git git remote add upstream https://github.com/ckan/ckan.git

git fetch upstream git merge upstream/master master -m 'merge upstream' git push

7.4 Database

Change data dir to external volume:

```
service postgresql stop
mkdir /var/www/pgdata
mv /var/lib/postgresql/9.3/main/ /var/www/pgdata/
ln -s /var/www/pgdata/main /var/lib/postgresql/9.3/main
chown -R postgres:postgres /var/lib/postgresql/9.3/main
service postgresql start
```

Setup database for CKAN:

```
sudo -u postgres psql -l
sudo -u postgres createuser -S -D -R -P ckan_default
sudo -u postgres createuser -S -D -R -P -l datastore_default
sudo -u postgres createdb -O ckan_default ckan_private -E utf-8
sudo -u postgres createdb -O ckan_default ckan_public -E utf-8
sudo -u postgres createdb -O ckan_default datastore_private -E utf-8
sudo -u postgres psql -d ckan_private -f /usr/share/postgresql/9.3/contrib/postgis-2.1/postgis.sql
sudo -u postgres psql -d ckan_private -f /usr/share/postgresql/9.3/contrib/postgis-2.1/spatial_ref_sy
sudo -u postgres psql -d ckan_private -c "ALTER TABLE spatial_ref_sys OWNER TO ckan_default;ALTER TAB
```

7.5 SolR

cd /tmp sudo su wget http://archive.apache.org/dist/lucene/solr/4.10.2/solr-4.10.2.tgz tar -xvf solr-4.10.2.tgz cp -r solr-4.10.2/example /opt/solr

cp /opt/solr/solr/collection1 /opt/solr/solr/ckan sed -i "s/collection1/ckan/g" /opt/solr/solr/ckan/core.properties ln -s /var/projects/ckan/src/ckan/ckan/config/solr/schema.xml /opt/solr/solr/ckan/conf/schema.xml

Download JTS-1.13, unpack and copy .jars to solr to enable spatial search:

```
wget http://downloads.sourceforge.net/project/jts-topo-suite/jts/1.13/jts-1.13.zip
unzip jts-1.13.zip
cp *.jar /opt/solr/lib/
cp *.jar /opt/solr/solr-webapp/webapp/WEB-INF/lib/
```

`/etc/supervisor/conf.d/solr.conf`:

```
[program:solr]
autostart=True
autorestart=True
directory=/opt/solr
command=/usr/bin/java -Dsolr.solr.home=/opt/solr/solr -Djetty.logs=/opt/solr/logs -Djetty.home=/opt/s
```

Reload supervisorctl to start solr.

Test:

```
curl 127.0.0.1:8983/solr/ckan/select/?fl=*,score&sort=score%20asc&q={!geofilt%20score=distance%20filt
```

7.6 Data store and datapusher

Follow the CKAN docs, but install datapusher into the project directory with:

```
mkvirtualenv datapusher
```

This will create `/var/projects/datapusher` for files, and `/var/venvs/datapusher` for the virtualenv.

7.7 Spatial referencing

Install ckanext-spatial, or use DPaW's special spatial widgets. DPaW's widgets are live in action at their public data catalogue release candidate http://data-demo.dpaw.wa.gov.au/, and a video is available at https://vimeo.com/116324887.

7.8 Custom dataset schema

DPaW uses custom schemas, also live at their public data catalogue release candidate http://data-demo.dpaw.wa.gov.au/, but for simplicity's sake not enabled in the yes-we-ckan instance.

7.9 Multi-tenant install

Follow the Department of Parks and Wildlife's guide at https://twitter.com/opendata/status/555760171017056256. Deployment diagram: http://data-demo.dpaw.wa.gov.au/dataset/da79b533-2813-42fb-a455-d9b32998abbf/resource/ae26b6f2-5ad5-46f1-b91d-3d40a19a77c4/download/CKANMultiSite.pdf

CHAPTER 8

RStudio and RShiny package install

This chapter describes the installation of R packages and servers to create a data science workbench. Deployments using Docker, e.g. https://github.com/rocker-org/rocker, are easier to install, however, using system users (e.g. AD users provided via winbind) for the RStudio user management requires a non-docker install.

Assuming the VM runs Ubuntu 14.04 "Trusty Tahr":

- add R repo to `/etc/apt/sources.list`
- install system packagegs
- install R libraries
- download and install RStudio Server and RWhiny Server

  sudo su

  echo "deb http://cran.csiro.au/bin/linux/ubuntu trusty/" >> /etc/apt/sources.list

  apt-key adv –keyserver keyserver.ubuntu.com –recv-keys E084DAB9 apt-get update && apt-get -y upgrade apt-get -y dist-upgrade reboot

  apt-get -y install r-base r-base-dev gdebi-core texlive-full htop ncdu libcurl4-openssl-dev libxml2-dev

  sudo su - -c "R -e "install.packages(c('shiny', 'shinyIncubator','markdown','whisker','Hmisc','dplyr','tidyr','lubridate','qcc','httr' repos='http://cran.rstudio.com/');devtools::install_github("ropensci/ckanr")""

  wget -O /tmp/rstudio.deb https://download2.rstudio.org/rstudio-server-0.99.467-amd64.deb gdebi -n /tmp/rstudio.deb wget -O /tmp/shiny.deb https://download3.rstudio.org/ubuntu-12.04/x86_64/shiny-server-1.4.0.721-amd64.deb gdebi -n /tmp/shiny.deb

  /home/ubuntu/shiny-serve$ git clone git@github.com:florianm/shiny-timeseries.git

8.1 Nginx

Configure as in Data Science Workbench.

Notes on Shiny's app dir:

- RStudio Server lets ubuntu access files only in own home directory `/home/ubuntu`.
- RShiny Server can use any directory as app directory, default is `/srv/shiny-server`.
- The external volume has enough space (100GB) and can be snapshotted (btrfs).
- The correct folders have already been created in the DevOps chapter.
- `/srv/shiny-server` is a symlink to `/mnt/btrfsvol/shiny-server`.

- The user ubuntu's home directory contains another symlink to `/srv/shiny-server`.