**OPEN DATA CUBE SETUP**

**1.Add the conda-forge channel**

conda config --add channels conda-forge

conda-forge is a community effort that provides conda packages for a wide range of software.

2. **Create a virtual environment in conda**

conda create --name cubeenv python=3.6 datacube

An environment consists of a certain Python version and some packages. Consequently, if you want to **develop or use applications with different Python or package version requirements**, you need to set up different environments.

The two most popular tools for setting up environments are:

* [**PIP**](https://pip.pypa.io/en/stable/)(a Python package manager; funnily enough, it stands for “Pip Installs Packages”) with **[virtualenv](https://virtualenv.pypa.io/en/stable/)** (a tool for creating isolated environments)
* **Conda**(a package and environment manager)
* Inside a new Conda installation, the root environment is activated by default, so you can use it without activation.
* In other cases, if you want to use an environment (for instance manage packages, or run Python scripts inside it) you need to first activate it.

**CONDA SEARCH FOR A PACKAGE** -- You can search for available packages

conda search $SEARCH\_TERM

e.g.- Search for a specific package named 'scikit-learn': --- conda search scikit-learn

Search for packages containing 'scikit' in the package name: --- conda search \*scikit\*

3. **Activate the virtual environment**

source activate cubeenv

4. **Install other packages**

conda install jupyter matplotlib scipy

**Database Setup**

After installing postgreSQL, launch PostgreSQLPortable.exe

To prepare the database for first use, enter the following commands in the PostgrSQL Portable window, substituting “u12345” with your windows login user-ID:

create role u12345 superuser login;

create database datacube;

createdb datacube

delete the database by running dropdb datacube

run postgresql on cmd

psql -U postgres:

asks for pwd: \*\*\*\*\*

**CREATE DB** : create database dbName ; {; is important}

To enter a database -- \c database name.

**ADD CONFIG FILE (**VERY VERY IMP**)**

Add a datacube.config file in C:/Users/SinghCoder (template from documentation).

Add in it details asked.

And then to check if datacube working properly

Go to anaconda prompt > conda activate cubeenv >datacube system check (No error => successful connection)

**User creation**

Create role <username> superuser login

**{IMP:: username ko “” me daal de if Capital letters hai isme}**

After that go in conda prompt

Activate env

Type datacube system check

Then datacube system init

You can connect to multiple indexes from 1 python process.

While initialising db instance , it loads configuration options from 1 or more config files (options define which indexes are available and parameters reqd to connect to them)

Two indexes available - standard PostgreSQL backed index, and the other is an extension to the standard index, with additional support for data stored in the S3 AIO format.

 type of index driver to use is defined by the *index\_driver* option in each section of the user config file.

The runtime config specifies configuration options for the current user, such as available Data Cube instances and which to use by default.

When using the datacube, it will use your default environment unless you specify one explicitly

eg. **with** Datacube(env='staging') **as** dc:

When you load data into the Data Cube, all you are doing is recording the existence of and detailed metadata about the data into the **index**. None of the data itself is copied, moved or transformed. This is therefore a relatively safe and fast process.

to load in some data. This step is performed using the **datacube** command line tool.

**INDEXING DATA**

Loading the data into datacube == recording the existence of and detailed metadata about the data into the index. None of the data itself is copied, moved or transformed. This is therefore a relatively safe and fast process.

**STEPS TO ADD DATA**

1. Create a new Product – which describes data.

We create a product definition (yaml doc)

It is used to give datacube a info about what type of data it is and what it should do with it.

A Product Definition provides a short **name**, a **description**, some basic source **metadata** and (optionally) a list of **measurements** describing the type of data that will be contained in the Datasets of its type. In Landsat Surface Reflectance, for example, the measurements are the list of bands.

The **measurements** is an ordered list of data, which specify a **name** and some **aliases**, a data type or **dtype**, and some options extras including what type of **units** the measurement is in, a **nodata** value, and even a way of specifying **bit level descriptions** or the **spectral response** in the case of reflectance data.

Examples of some yaml files --- <https://github.com/opendatacube/datacube-core/tree/develop/docs/config_samples/dataset_types>

**Load products in datacube:--**

datacube product add <path-to-product-definition-yml>

**Metadata**

Every dataset requires a metadata document describing what the data represents and where it has come from, as well has what format it is stored in. At a minimum, you need the dimensions or fields your want to search by, such as lat, lon and time, but you can include any information you deem useful.

It is typically stored in YAML documents, but JSON is also supported. It is stored in the index for searching, querying and accessing the data.

The data from Geoscience Australia already comes with relevant files (named ga-metadata.yaml), so no further steps are required for indexing them.

For third party datasets <https://datacube-core.readthedocs.io/en/latest/ops/prepare_scripts.html#prepare-scripts>

**IMP: To add metadata types information in the database:**

Command is : datacube metadata add <File path {like default\_metadata\_types.yaml}> instead of datacube product add <File path>

**POSTGRESQL COMMAND**

\dt – to view all the existing tables

Select \* from agdc.dataset\_type shows what you added via datacube add product…..

After adding product definition, add metadata yaml file corresponding to your actual data .

This yaml file can be generated using python script <https://opendatacube.readthedocs.io/en/latest/_downloads/aaf3a3c01cf57e78d674a773c6597878/ls_usgs_prepare.py>

ls\_usgs\_prepare.py using command

python <path to this script file> --ouput <name of yaml file with extension> <path to folder containing your mtl.txt file>

After generating this yaml file add it to datacube using command

**datacube dataset add --auto-match <path-to-dataset-document-yaml>**

**DJANGO ADD INDEXES INFORMATION TO DATABASE**

In models.py create a class named indices having two fields , indexName and indexFormula

Then python manage.py makemigrations

Python manage.py mirate

Then to play within shell (API Django provides)

python manage.py shell

from myapp.models import indices

indices.objects.all()

i = indices ( indexName=’NDVI’ , indexFormula = ‘(nir - r)/(nir + r)’ )

i.save()

i.id

i.indexFormula