**Datacube Installation and Setup:**

**Step-1: Requirements**

**Datacube** runs the best on Ubuntu version 20.04. So it is recommended to use **Ubuntu.**

There is also a third-party tested software called as OSGeoLive VM which is based on LUbuntu 22.04. This provides pre-configured applications for a range of geo-spatial use-cases that includes storage, publishing, viewing, analysis and manipulation of data.

For more: <https://live.osgeo.org/en/index.html>

The other requirements for the installation and setup are

* Python- 3.8+
* GDAL (libgdal-dev, gdal-bin, libgdal-doc)
* Rasterio – 1.3.2+
* Miniconda
* NETCDF(libnetcdf-dev, netcdf-doc, netcdf-bin)
* HDF(libhdf5-serial-dev, libhdf5-doc, hdf5-tools)
* Postgres (12+)
* Tornado 6.1

1. Add conda-forge to the package channels by running the following command:

**conda config –append channels conda-forge**

1. Create a conda environment with name odc\_env

**conda create –name odc\_env python=3.8 datacube**

1. Activate the odc environment

**conda activate odc\_env**

1. Once the odc\_env is activated, the terminal prompt should be similar to the below screenshot



1. Install the following python packages

**Jupyter, matplotlib, scipy, pytest-cov, hypothesis**

**Step-2 Postgres Database Configuration for testing:**

1. If the postgres has been newly installed for the purpose of datacube setup, then it is better to set the postgres user password
2. In a terminal type:

**sudo –u postgres psql postgres**

1. Set a password for the user “postgres” by entering the below command in the terminal

**\password postgres**

1. Create a database called “agdcintegration” for testing and try connecting to the database

**CREATE DATABASE agdcintegration;**

**psql –d agdcintegration**

This concludes the Postgres database setup. Now the next step would be datacube installation

**Step-3 Datacube Installation:**

1. Clone the datacube repository from GitHub. The URL is provided below:

<https://github.com/opendatacube/datacube-core.git>

1. We need to specify the database user and password for the ODC integration testing. To do this copy the database conf file using the below command

cp integration\_tests/agdcintegration.conf ~/.datacube\_integration.conf

1. Edit the ~/.datacube\_integration.conf with a text editor and add the following lines.

[datacube]

db\_hostname: /var/run/postgresql

db\_database: agdcintegration

db\_username: postgres

db\_password: {Password\_for\_the\_user\_postgres}

[integration]

db\_hostname: /var/run/postgresql

db\_database: agdcintegration

db\_username: postgres

db\_password: {Password\_for\_the\_user\_postgres}

The above lines are for Ubuntu based setup.

Before going to next step do

pip install moto

in etc/postgresql/12/main/postgresql.conf

uncomment line 60(listen\_address) and save

pip install -e .

run req.txt file in

<https://github.com/abhijeethreddy/datacube/blob/main/req.txt>

by pip install -r req.txt

**Step 4 Verification:**

1. Run the integration tests by running the below command

cd datacube-core

./check-code.sh integration\_tests

1. The integration tests will succeed if the database hostname and other credentials are provided accurately.

**Note:** 92% successful tests is an acceptable level to proceed with the data cube setup.

**Step-5 Database Setup for Datacube(Actual Datacube):**

1. Create a database named datacube using the following command

**CREATE DATABASE datacube;**

1. Now a configuration file needs to be created as datacube looks for a configuration file in **~/.datacube.conf** or in the location specified by **DATACUBE\_CONFIG\_PATH** path environment variable.

**Create the file and fill it up with the below details**

[datacube]

index\_driver: default

db\_database: datacube

*# A blank host will use a local socket. Specify a hostname (such as localhost) to use TCP.*

db\_hostname: /var/run/postgresql

*db\_username: postgres*

*db\_password: {postgres\_password}*

[test]

*# A "test" environment that accesses a separate test database.*

index\_driver: default

db\_database: datacube\_test

[null]

*# A "null" environment for working with no index.*

index\_driver: null

[local\_memory]

*# A local non-persistent in-memory index.*

*# Compatible with the default index driver, but resides purely in memory with no persistent database.*

*# Note that each new invocation will receive a new, empty index.*

index\_driver: memory

1. Now initialise the database schema by using the datacube system init. Run the following command

**datacube –v system init**

**Step-6 Product Definition:**

A product in odc could be considered as a set of properties that are common for the datasets.

It could be something like bands, CRS, resolution etc.

Sample Product yml file:

**name**: dem\_srtm

**metadata\_type**: eo3

**metadata**:

**product**:

**name**: dem\_srtm

**measurements**:

- **name**: elevation

**dtype**: int16

**nodata**: -32768.0

**units**: "metre"

You will be needing a product file to query the data. The product can be considered as a parameter that will help in filtration of data.

Add product by

datacube product add S2L2A\_eo3.yaml

<https://github.com/abhijeethreddy/datacube/blob/main/S2L2A_eo3.yaml>

**Step-7 Adding The satellite imagery to the file-system:**

* The next step would be to add the satellite images to the file system
* Create a parent directory in the hard disk in which the data is to be stored
* For Sentinel L2A data, the following link would be helpful in downloading data into the

[https://scihub.copernicus.eu/dhus/#/home](https://scihub.copernicus.eu/dhus/" \l "/home)

* Download the data into this parent directory. The data would be downloaded in .zip format. Extract it into the same directory.

**Step-8 Transforming Metadata to YAML:**

* For each tile there is a .SAFE file from which the metadata can be obtained. The metadata is initially provided in an XML format.
* This needs to be converted into the YAML format.
* For this conversion, use the below script:

**https://github.com/abhijeethreddy/datacube/blob/main/sen2cor\_new\_data.py**

run this command at extracted location of zip files

**for file in \*.SAFE; do python <sen2cor\_new\_data.py path> $file --output <./datasets output path>/ ;done;**

This will help you get the metadata for the datacube

* Note: This step required to go through the scripts directory in the **datacube-dataset-config** repository

**Step-9 Add datasets to the datacube:**

* The metadata YAML file obtained in the above step should now be added to the datacube
* This can be done by performing the command

Datacube dataset add \*.yaml

for adding multiple files

for file in \*.yaml ;do datacube dataset add $file; done;

**Note:** \*.yaml file is the name of the metadata file to be added.

**Installing datacube explorer**

pip install datacube-explorer

cubedash-gen --init --all

cubedash-run

check if products are available

**Step-10 Verify whether it works:**

* Now, one can verify whether this works by querying the datasets and the product.
* Try running the jupyter notebooks from the below URL:

[**https://github.com/GeoscienceAustralia/dea-notebooks/tree/develop/Beginners\_guide**](https://github.com/GeoscienceAustralia/dea-notebooks/tree/develop/Beginners_guide)

**Note:** When performing operations that involve latitude, longitude and time, make sure that these values are well within the dataset that we use for the datacube.