CEOS ODC Documentation

Oct 25, 2021

ODC PRODUCTS OVERVIEWS

1 CEOS ODC Products Overviews **3**
- 1.1 CEOS ODC UI . . . 3
- 1.2 CEOS ODC Notebooks . . . 3
- 1.3 CEOS ODC Utilities . . . 3
- 1.4 CEOS ODC Indexer . . . 3
- 1.5 CEOS ODC-GEE Integration Package . . . 4

2 Other ODC Products Overviews **5**
- 2.1 ODC Platform . . . 5

3 Environment Setup Guide **7**
- 3.1 Installation . . . 7
- 3.2 Testing . . . 8

4 ODC Database Installation Guide **9**
- 4.1 Prerequisites . . . 9
- 4.2 Installation . . . 9
- 4.3 External Connections . . . 10

5 CEOS ODC-GEE Technical Documentation **11**
- 5.1 Operation Requirements . . . 11
- 5.2 Adding New GEE Datasets as ODC Products . . . 11
- 5.3 Using GEE vs non-GEE Data . . . 12
- 5.4 GEE versus Alternative Datasources . . . 12

6 API Reference **15**
- 6.1 Data Info . . . 15
- 6.2 Data Masking/Cleaning . . . 15
- 6.3 Data Combining . . . 16
- 6.4 Data Transformations . . . 16
- 6.5 Visualization . . . 17
- 6.6 Dask . . . 18
- 6.7 Machine Learning . . . 18
- 6.8 EO Topics . . . 18
- 6.9 Export . . . 19
- 6.10 Mosaics . . . 19

7 Dependency Maps **21**

8 Obtaining Help **23**

This documentation covers CEOS Open Data Cube (ODC) deployments and related technologies.

CHAPTER

ONE

CEOS ODC PRODUCTS OVERVIEWS

1.1 CEOS ODC UI

The CEOS ODC UI (UI GitHub) is a Python Django webapp that uses the ODC to conduct some predefined analysis, such as cloud coverage estimation, water quality estimation, and coastal change detection.

The UI was designed to favor simplicity for casual users, but see the notebooks (mentioned below) before attempting to use the UI, since the UI is currently dated and harder to manage in comparison.

1.2 CEOS ODC Notebooks

The CEOS ODC Notebooks (Notebooks GitHub) is a collection of Jupyter Notebooks in a JupyterLabs environment uses the ODC to analyze satellite data.

Notebooks allow users to write code to conduct analyses. Some technologies such as widgets (from the Panel library) allow notebooks to be more easily usable - even by causal users (non-programmers).

1.3 CEOS ODC Utilities

The CEOS ODC Utilities (Utils GitHub) are a collection of Python files that provide several tools that conduct water detection, suspended matter estimation (water particulate pollution), coastline classification and change detection, and plotting - among other things.

1.4 CEOS ODC Indexer

The CEOS ODC Indexer (Indexer GitHub) provides convenient means of processing and indexing data for ODC environments.

1.5 CEOS ODC-GEE Integration Package

The CEOS ODC-GEE integration package (ODC-GEE GitHub) allows Google Earth Engine (GEE) data to be loaded into ODC environments.

This is particularly useful since GEE has many datasets and they can be "indexed" very quickly with ODC-GEE. Indexing a product only requires a product definition to be created (should be reformatted to look more like product definitions in the ODC Indexer) and added with `datacube product add` to allow an environment with ODC-GEE to be able to load data from it.

However, note that the throughput of the GEE API is quite slow, so loading data with ODC-GEE is also quite slow compared to loading data from data sources such as Amazon S3.

CHAPTER

TWO

OTHER ODC PRODUCTS OVERVIEWS

2.1 ODC Platform

The AMA (Analytical Mechanics Associates) Open Data Cube Platform (GitHub) allows creation of pre-defined JupyterLabs environments that use the Open Data Cube to analyze earth observation data. Each pre-defined environment is paired with an ODC index database that has some products and indexed data for those products relevant to that environment.

CHAPTER

THREE

ENVIRONMENT SETUP GUIDE

This document is a guide for setting up an environment to run the ODC - notably by installing Make, Docker, and Docker Compose.

3.1 Installation

3.1.1 Operating System

First, you will need to ensure you are in an Ubuntu 18.04 or 20.04 environment (20.04 recommended).

We encourage Windows 10 users to use [Windows Subsystem for Linux Version 2 (WSL2)](https://docs.microsoft.com/en-us/windows/wsl/install-win10).

3.1.2 Make, Docker, Docker Compose

Now install Make in your Ubuntu environment: *apt-get update && apt-get install make -y*

For Windows users, install [Docker for Windows (Docker Desktop)](https://docs.docker.com/docker-for-windows/).

For Ubuntu users (**not WSL Ubuntu**), ensure you are a sudoer (user that can run *sudo*). Run the *sudo-ubuntu-install-docker* target in the *Makefile* in the top-level directory of a repository (e.g. [data_cube_notebooks](https://github.com/ceos-seo/data_cube_notebooks)) to install Docker:

```
`bash make sudo-ubuntu-install-docker `
```

After running the above command, logout and login again.

If that *Makefile* target is not available or it does not install Docker properly (e.g. the test listed below fails or the repository's environment will not start in Docker), run these commands as a block and then logout and login again.

```
`bash apt-get update && \ apt install -y docker.io curl && \ curl
-L "https://github.com/docker/compose/releases/download/1.27.4
docker-compose-Linux-x86_64" -o /usr/local/bin/docker-compose && \ chmod +x /
usr/local/bin/docker-compose && \ systemctl start docker && \ systemctl enable
docker && \ # The following steps are for enabling use # of the `docker`
command for the current user # without using `sudo` getent group docker ||
sudo groupadd docker && \ usermod -aG docker ${USER} `
```

3.2 Testing

Once Docker is installed, test it with the following command: *docker run hello-world*

You should see output beginning with *Hello from Docker!*.

CHAPTER

FOUR

ODC DATABASE INSTALLATION GUIDE

This document is a guide for installing and configuring the Open Data Cube index database - which the ODC will query to determine where to retrieve data from (e.g. local or remote GeoTIFFs).

4.1 Prerequisites

Follow the Environment Setup Guide if you have not done so for your environment yet.

4.2 Installation

Ultimately, the database should be a Postgres instance accessible from the ODC installations you want to access it from.

You will need to know:

- The host name (IP or domain name of the database)
- The database name (name of the database within Postgres to use for ODC)
- The user name (name of the Postgres user that will access the database)
- The password (password for the aforementioned user)

These values will need to be set in the *datacube.conf* file for all ODC installations that should index this database. See the ODC Environment Configuration documentation for information on setting up ODC configuration files.

We can't include full instructions for how to set up the database on all cloud providers, but the instructions for setting it up locally with Docker using Linux shell commands are included below.

4.2.1 Creating the database

The following commands should be run from the top-level directory (directory containing *Makefile*).

First we need to create a network for the Docker container. The database will only be accessible from other Docker containers on this machine in the *odc* Docker network.

Run the following command to do this: *make create-odc-network*

Next we need to create a filesystem volume for the database data so that it remains on the local filesystem and is not lost whenever the Docker container for the database terminates.

Run the following command to do this: *make create-odc-db-volume*

Now we need to create the Docker container for the database. In the *Makefile* file at the top-level directory, find the *create-odc-db* target. Here you will see the command to create the ODC database Docker container. You can replace the values for *POSTGRES_DB*, *POSTGRES_USER* and *POSTGRES_PASSWORD* with the desired Postgres database name, user name, and password. Note that if you change these settings, you will need to change the corresponding settings for applications that use them, like the CEOS ODC Notebooks or the CEOS ODC UI.

Once you are ready to start the database, run this command: *make create-odc-db*

4.3 External Connections

If you want this database to be accessible from anywhere, just specify the mapping of the Postgres port (*5432*) to the port on the host that it should be available from with the argument *-p <host port>:5432* after *run* and before *postgres* in the command for the *create-odc-db* target in the *Makefile* file. Use *-p 5432:5432* if possible.

ODC installations on this machine (e.g. the ODC UI) can be pointed to this index database by setting the *db_hostname* value in the *datacube.conf* file to the *name* of the database Docker container (*odc-db* in this case), but ODC installations outside this machine will set *db_hostname* to *<IP>:<port>*, where *IP* is the IP or domain name of the machine the database is running on and *port* is the host port specified previously.

CHAPTER

FIVE

CEOS ODC-GEE TECHNICAL DOCUMENTATION

5.1 Operation Requirements

The ODC-GEE integration package ([GitHub](https://github.com/ceos-seo/odc-gee)) has a few requirements to operate properly.

5.1.1 Operation Requirement 1: Supported Metadata Types

The ODC index database used by your environment must support the `eo3` metadata type. This can be checked by running *datacube metadata list* in your ODC environment. If eo3 is not listed, then you will need to add it. The default eo3 type definition can be found [here](https://github.com/opendatacube/datacube-core/blob/datacube-1.8.3/datacube/index/default-metadata-types.yaml). To add it, save the eo3 definition in a *.yaml* file then run the following command: *datacube metadata add <your_eo3.yaml>*.

5.1.2 Operation Requirement 2: Earth Engine Credentials

To use the ODC-GEE integration package, you must be registered as an Earth Engine developer. If not, you may submit an [application to Google](https://signup.earthengine.google.com/).

You will need [GEE service account credentials](https://developers.google.com/earth-engine/guides/service_account) for your account - specifically the private key JSON file. Put your version of this JSON file here: *odc_gee/config/credentials.json*.

(Optional) You can also create a JSON file defining the spatial areas associated with region names. An example is provided in *odc_gee/opt/config/odc-gee/regions.json*. Put your version of this JSON file here: *odc_gee/config/regions.json*.

5.2 Adding New GEE Datasets as ODC Products

To use a GEE dataset from the [Earth Engine Data Catalog](https://developers.google.com/earth-engine/datasets/), a new product must be created using the *new_product* command. Format: *new_product –asset <asset_id> <product_name.yaml>* where the *asset_id* is provided in the "Earth Engine Snippet" string on the dataset's page on the catalog and *<product_name.yaml>* is the path to the output YAML file containing the ODC product definition. For example, to index [Landsat 8 Level 2 Collection 2 Tier 1](https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC08_C02_T1_L2?hl=en) data: *new_product –asset LANDSAT/LC08/C02/T1_L2 ls8_l2_c2_gee.yaml*. The full process is as follows:

1. Run the *new_product* command to create the product definition.

2. Reformat the product definition to match a standard format, such as [this one from the ODC Indexer](https://github.com/ceos-seo/odc_manual_indexer/blob/develop/prod_defs/Landsat/collection_2/ls8_l2_c2.yaml).
3. Change the *aliases* field for the measurements as desired. Do NOT change the *name* field of any measurement - creating the product will fail if the *name* fields are changed.
4. Run *datacube product add <path-to-product-definition-file>* to add the product.

Notably, you will need to add a *storage* section with *crs* and *resolution* entries to avoid having to specify the *output_crs* and *resolution* each time data is loaded from the product.

After adding the product, it is a **non-indexed GEE product** (or [“undefined product”](https://github.com/ceos-seo/odc-gee#undefined-product)). It must be loaded using the [ODC-GEE *datacube.Datacube* wrapper class](https://github.com/ceos-seo/odc-gee#datacube-wrapper).

Once you have confirmed that data is loading for the newly defined product, you are welcome to add a pull request to the [ODC Indexer](https://github.com/ceos-seo/odc_manual_indexer) repository to add the new product definition to the *prod_defs* directory. Prefix the product definition file with *_google*.

The data can be indexed using the *index_gee* command, making it an **indexed GEE product**, but **this is deprecated**. Format: *index_gee –asset <asset_id> –product <product_name> [–latitude (lat1, lat2) –longitude (lon1, lon2) –time (YYYY-MM-DD, YYYY-MM-DD) –region <region_name>]* (for information on the optional arguments and others not listed here, run *index_gee –help*). For example, to index [Landsat 8 Level 2 Collection 2 Tier 1](https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC08_C02_T1_L2?hl=en) data for the United States: *index_gee –asset LANDSAT/LC08/C02/T1_L2 –product <product_name> –latitude (25.3168, 49.4885) –longitude (-125.2052, -66.6657)*. Data for these products can be loaded by the normal *datacube.Datacube* class.

5.3 Using GEE vs non-GEE Data

To load data from non-GEE products, use the *datacube.Datacube* class as always:

`python from datacube import Datacube dc = datacube.Datacube() `

To load data from GEE products, use the *odc_gee.earthengine.Datacube* class:

`python from odc_gee.earthengine import Datacube as GEE_Datacube dc = GEE_Datacube() `

To load data from indexed GEE products (remember **this is deprecated**), use the *datacube.Datacube* class.

5.4 GEE versus Alternative Datasources

These are the benefits and penalties of loading data from Google Earth Engine through the ODC-GEE module instead of from other datasources such as S3.

Benefits:

1. Data does not need to be indexed before loading, which allows new datasets to be added and queried quickly, which allows faster prototyping. This also results in a much smaller ODC index database.
2. There is no cost to loading data from GEE.

Penalties:

1. Data has a very low throughput - just a few MiB per second.

2. (Advanced) Using *dask_chunks* in the *load()* call does not work as expected. Normally, loads specifying the *dask_chunks* parameters will not immediately load data - instead creating the plan to load data with Dask. Instead, the data immediately begins loading as if *dask_chunks* was not specified. This can be problematic for datasets that are larger than the amount of available memory. This problem does not occur when loading data using the normal *datacube.Datacube* class.

CHAPTER

SIX

API REFERENCE

6.1 Data Info

`dc_load.is_dataset_empty`
`dc_load.get_product_extents`
`dc_load.get_overlapping_area`
`dc_load.find_desired_acq_inds`
`dc_utilities.get_range`
`data_stats.find_gaps`

6.2 Data Masking/Cleaning

6.2.1 General

These masking utilities are for generic use with xarrays.

`clean_mask.xarray_values_in`
`clean_mask.create_circular_mask`
`shapefile_mask.shapefile_mask`

6.2.2 Dataset-specific

Landsat

These masking utilities serve to clean Landsat data.

We recommend using `landsat_clean_mask_full` for simplicity and `landsat_qa_clean_mask` if you need to be specific about what is masked (e.g. cloud shadow).

All of these functions except `landsat_clean_mask_invalid` require the QA data (often called `pixel_qa` as a measurement for Landsat products in ODC).

`clean_mask.landsat_clean_mask_invalid`
`clean_mask.landsat_qa_clean_mask`
`clean_mask.landsat_clean_mask_full`

Sentinel-2

These masking utilities serve to clean Sentinel-2 data.

`clean_mask.sentinel2_fmask_clean_mask`

6.3 Data Combining

`dc_load.match_prods_res`
`dc_load.match_dim_sizes`
`dc_load.get_product_extents`

6.4 Data Transformations

6.4.1 Aggregation/Rescaling

These utilities allow binning, grouping, and rescaling features (such as resolution) beyond those offered by xarray, or having simplified interfaces, or both.

Note that `xr_scale_res` is a simpler interface to `xr_interp`.

`aggregate.xr_sel_time_by_bin`
`dc_load.reduce_on_day`
`aggregate.xr_scale_res`
`aggregate.xr_interp`

6.4.2 Kernel-Based Filters

There are 2 kinds of kernel-based filters offered here: seletive and non-selective.

Selective filters apply to only some data points. These include: [`raster_filter.lone_object_filter`]

Non-selective filters apply to all data points. These include: [`stats_filter_3d_composite_2d`, `stats_filter_2d`]

`raster_filter.lone_object_filter`
`raster_filter.stats_filter_3d_composite_2d`
`raster_filter.stats_filter_2d`

6.4.3 Conversion

Sometimes data needs to be transformed to more closely match another dataset (e.g. converting Landsat 8 Level 2 Collection 2 data to approximate Landsat 8 Level 2 Collection 1 data to accomodate algorithms that only support the latter).

`dc_utilities.convert_range`

6.5 Visualization

6.5.1 2D Data Display

`plotter_utils.xarray_imshow`
`dc_rgb.rgb`

6.5.2 Plotting

`plotter_utils.xarray_time_series_plot`
`plotter_utils.binary_class_change_plot`
`plotter_utils.create_discrete_color_map`
`plotter_utils.create_gradient_color_map`
`plotter_utils.impute_missing_data_1D`

6.5.3 Figure Sizing

`figure_ratio` is often used to set the size of *matplotlib* figures and axes (created by `matplotlib.pyplot.Figure()` or `matplotlib.pyplot.subplots()`).

`plotter_utils.figure_ratio`

6.5.4 Animation

`xr_animation` is from Geoscience Australia's utilities here.

`plotter_utils.xr_animation`

6.6 Dask

dask.create_local_dask_cluster

6.7 Machine Learning

6.7.1 Clustering

dc_clustering.kmeans_cluster_dataset
dc_clustering.birch_cluster_dataset
dc_clustering.get_frequency_counts

6.8 EO Topics

6.8.1 Urbanization

urbanization.NDBI
urbanization.DBSI

6.8.2 Fires

vegetation.NBR

6.8.3 Vegetation

vegetation.EVI
vegetation.EVI2
vegetation.NDVI
vegetation.SAVI

6.8.4 Water

Water Detection

dc_water_classifier.NDWI
dc_water_classifier.wofs_classify

Water Quality

dc_water_quality.tsm

Coastlines

dc_coastal_change. compute_coastal_change

6.8.5 Landslides

dc_slip.compute_slip

6.8.6 Land Classification

dc_fractional_coverage_classifier. frac_coverage_classify

6.9 Export

import_export.export_xarray_to_netcdf
import_export.export_xarray_to_multiple_geotiffs
import_export.export_xarray_to_geotiff

6.10 Mosaics

dc_mosaic.create_min_max_var_mosaic
dc_mosaic.create_mosaic
dc_mosaic.create_mean_mosaic
dc_mosaic.create_median_mosaic
dc_mosaic.create_hdmedians_multiple_band_mosaic

CHAPTER

SEVEN

DEPENDENCY MAPS

Below is a map of CEOS ODC environments and their dependency chains.

- Each cell is a Docker repository (which can have many images).
- The names have this format: **<image-name>[:version] (repo-path: (<hyperlink-to-repo-or-Dockerfile-in-repo>))**
- The image version (**[:version]**) is optional, as denoted by the square brackets ([]).
- If the **repo-path** is to a repo instead of a Dockerfile in a repo, check the **build** directory in the repo for the Dockerfile.
- Note that **image** and **image:latest** (or even **image:odc<odc_version>**) may or may not be identical (diagram is a simplification).

Table 1: CEOS ODC Dependency Map

	ubuntu:20.04		
	|		
	ubuntu-base (repo-path: datacube-base/ubuntu-base)		
	|		
	datacube-base (repo-path: datacube-base)		
v-	|	-v	
data_cube_notebooks (repo-path: data_cube_notebooks)	data_cube_ui (repo-path: data_cube_ui)	odc_manual_indexer (repo-path: odc_manual_indexer)	-v
|	-v ->	-v	|
odc_drone_paper_notebooks (repo-path: odc_drone_paper_notebooks)	odc_training_notebooks (repo-path: odc_training_notebooks)	odc_va_cube_notebooks (repo-path: odc_va_cube_notebooks)	odc_manual_indexer_init (repo-path: odc_db_init)
|->	|	<-|	|
	odc_platform (repo-path: odc_platform)	<-	<-|

CHAPTER

EIGHT

OBTAINING HELP

If you encounter issues with the Open Data Cube or derived products - whether CEOS ODC products (notebooks, UI, etc.) or otherwise - that are not documented in the corresponding repositories (for CEOS repositories, check files in the *README.md* file at the root directory and the files in the *docs* directories - including the FAQ sections at the bottoms of some documents) and you are unable to diagnose and fix those issues on your own, follow these steps to obtain assistance:

1. Search for your question in the GIS Stack Exchange with the *open-data-cube* tag. You can find such tagged questions *here <https://gis.stackexchange.com/questions/tagged/open-data-cube>*. If a question similar to yours has already been asked, search for a suitable answer in that question's webpage. If you can not find a suitable answer, contine to step 2. If no such question exists or if the question is missing some information that you think may be important regarding your issue, create a new question with the *open-data-cube* tag.
2. State your problem in the ODC Slack workspace in the most appropriate channel (use that link to join the workspace if you have not already). Use the **#general** channel if no other channel is more appropriate for your question. Link to the question on GIS Stack Exchange. When you receive an answer, add that answer to the question page on GIS Stack Exchange if the answerer is not available to do so (ask them if they would like to so they can get credit). This preserves the answer in a publicly searchable way - which is useful for remembering answers to one's own past questions and benefits the community.