Using Cognite Python SDK to read and write new datasets to CDF

[**Version Control** 2](#_Toc55391430)

[1. Introduction 3](#_Toc55391431)

[1.1 Stakeholders 3](#_Toc55391432)

[1.2. Requirements 3](#_Toc55391433)

[1.3. Vocabulary 3](#_Toc55391434)

[2. Goals 3](#_Toc55391435)

[3. Existing Solution 3](#_Toc55391436)

[4. Solution Overview 3](#_Toc55391437)

[5. Application architecture 4](#_Toc55391438)

[5.1 Interfaces 4](#_Toc55391439)

[6. Information architecture 4](#_Toc55391440)

[6.1 Data sources 4](#_Toc55391441)

[6.2 Data model 4](#_Toc55391442)

[6.3 Data architecture 4](#_Toc55391443)

[6.4 Data quality and data validation 4](#_Toc55391444)

[7. Deployment 4](#_Toc55391445)

[8. Security Architecture 5](#_Toc55391446)

[8.1 Identity and Authentication 5](#_Toc55391447)

[8.2 Authorization and access control 5](#_Toc55391448)

[9. Technical Architecture 5](#_Toc55391449)

[9.1 Environments 5](#_Toc55391450)

[10. Operations 5](#_Toc55391451)

[10.1 Monitoring and alerting 5](#_Toc55391452)

[10.2. Logging 5](#_Toc55391453)

[10.3. Job scheduling 5](#_Toc55391454)

[10.4. Roles and Responsibilities 5](#_Toc55391455)

[11. Known Gaps and Issues 6](#_Toc55391456)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version Control** | | | | |
| Version | Date | Change Descritpion | Responsible | Approver |
| 1.0 | 07.09 | Initial version | Vetle Nevland |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Introduction

*Problem statement - Here we shall describe what the problem is and what are we trying to achieve, source systems, use stories (functional requirements, business rules).*

The goal of this project is to demonstrate how to use the Cognite Python SDK to read a dataset from Cognite CLEAN, perform transformations using Cognite Functions written in Python, populate a new dataset with the transformed data, and finally deploying the dataset to CDF. The process is demonstrated on time series data of fluid volume percentage in a tank. Through the Python SDK we want to transform the data into a time series of drainage rate to facilitate leakage detection.

## Stakeholders

|  |  |  |
| --- | --- | --- |
| **#** | **Stakeholder** | **Role** |
| 1 | Stephanie Raoul | CoE Analysis dataset owner |
| 2 | Vetle Nevland | Development of Cognite Functions and deplyment of dataset |
| 3 | CDF Operation | Access granting and creation of dataset |

## 1.2 Requirements

*Use cases and requirements should be detailed here*

Running the project requires read/write access from/to CDF, Cognite Python SDK and basic knowledge of Python. Writing confidential data is not supported in this project. Potential use cases are broad; advanced Cognite Functions can be written in Python SDK to transform not only time series, but any of the CDF resource types.

## 1.3 Vocabulary

*Terminology which the target audience might be unfamiliar with and that this document understanding depends of. Some examples below.*

|  |  |
| --- | --- |
| **Term** | **Description** |
| CDF | Cognite Data Fusion |
| Asset | CDF resource type which acts as a digital representation of objects from the physical world |
| Time Series | CDF resource type to store data points in temporal sequence |
| SME | Subject Matter Expert |
| SDK | Software Development Kit |
| Dev | Development |
| Prod | Production |

# Goals

*KPIS*:

Demonstrate the versatility of Cognite’s Python SDK to perform advanced transformations on data, and integrate this with datasets in CDF.

Demonstrate the process on an existing time series in CDF to produce a new time series to help detect leakages from tanks, which provides value to assets in terms of decreased production loss and downtime.

KPIs:

* Provide daily average drainage rate from tanks

# Existing Solution

*Describe current solution if exists*

The solution already exists, but the usage of Cognite Python SDK is not well integrated in the workflow in Aker BP. To this end, a clear documentation of the extract-transform-deploy process is needed.

# Solution Overview

*Describe the solution to the problem statement, including CDF resources and core functionalities as well as all infrastructure and resources used outside CDF which are used/deployed in the solution. Be sure to include diagrams which will make the understanding of the overall design easier to the reader.*

The solution involves extracting time series from CDF, perform transformations using Cognite Python SDK, deploy new dataset to akerbp-dev (Cognite Fusion Dev) tenant, run tests and eventually write the dataset to akerbp (Cognite Fusion Prod) tenant. The solution also details the process of granting required read/write access from/to CDF. The solution is summarized as follows:

1. Install the Cognite Python SDK to your development environment
2. Create an account at Cognite Hub
3. Apply for access to read datasets from and write new datasets to Cognite Fusion Dev (akerbp-dev)
4. Using Cognite Python SDK:
   1. Interactively authenticate with a Cognite Client
   2. Extract desired time series
   3. Create a Cognite Function that performs transformations on the input time series to produce a new time series of drainage rate
   4. Populate the new dataset (created by the CDF Operation team) with the new time series data
5. Deploy the new dataset in the akerbp-dev tenant
6. Apply for read/write access to CDF’s testing environment (akerbp-test)
7. Testing phase:
   1. Set up tests to validate quality of our solution
   2. Run the tests
   3. Proceed when all tests have passed
8. Apply for write access to CDF’s production environment (akerbp)
9. Finalize the dataset, ensure governance and deploy to the akerbp tenant

# Application architecture

The overall architecture and purpose of this solution, highlighting specific design or technological features and components. Include diagrams where appropriate.

Simplified, the architecture can be depicted as a workflow loop where Cognite Fusion Prod marks the start and end destination. Dataset of interest is extracted from Cognite Fusion Prod, transformed with Cognite Functions using the Python SDK, (what about contextualization?), written to Cognite Fusion Dev for further development and testing, and eventually deployed as a governed dataset to Cognite Fusion Prod. Once the dataset resides in Cognite Fusion Dev or Prod, time series can be visualized/analyzed in Grafana Dashboards.

## Interfaces

Describe if any.

The diagram below illustrates the architecture.

A diagram of a process

Description automatically generated

# Information architecture

*Define the information strategy or business solution through the definition of the company’s business information assets, their sources, structure, classification and associations.*

## Data extraction

Data is extracted from CDF CLEAN in the Cognite Fusion Prod tenant through the Python SDK:

* Show code snippet (client.time\_series.retrieve())
* client.time\_series.data.retrieve(external\_id=cdf\_ext\_id)

## Data sources

*Include all data sources, including CDF resources as well as any outside data used in the solution.*

The source of the dataset is a time series CDF resource, more specifically a PI time series, with external ID: EXT\_ID.

## Create new dataset

A particular dataset is to be dedicated to the time series produced in this project. As such, the CDF Operations team create a new dataset for us after our request for write access has been accepted.

## Data transformations

The data has already been transformed to a CDF time series resource type. The transformations performed herein are done through Cognite Functions using Cognite’s Python SDK. This includes aggregation, smoothing and calculation of derivative to obtain the daily average drainage rate as a new time series data object. The transformations are described in 2 – DataIntegrationArchitecture.

## Data model

*How the data will be modelled in CDF*

Apart from the Cognite Functions in the transformation steps, no modification is done on the source data, metadata and other data connected in a CDF model. Relationships with assets are sustained.

## Data quality and data validation

*Standards of data quality, operations, and Service Level Agreements (SLAs).*

Data quality and integrity is accounted for through SIT and UaT testing frameworks in the akerbp-test tenants as part of the testing phase. See separate documentation of testing.

# Deployment

*The necessary steps and tools to deploy the solution*

Deployment of the complete dataset is done internally in CDF with the necessary write access.

Solutions in the Grafana Dashboards are deployed as follows …

# Security Architecture

## Identity and Authentication

*On CDF, the solution follows the existing identity and authentication regime.*

The solution follows the existing identity and authentication regime as implemented in CDF. Interactive authentication is conducted through Cognite Python SDK.

## Authorization and access control

*For data ingestion into CDF, a service account, api key, access groups with the required capabilities, and finally a dataset in which the data will be written is required. Please provide the information below.*

There are no specific access restrictions on the produced dataset. General access permissions apply.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Stage** | **Tenant** | **Service account** | **Authorization** | **Data Set** |
| Development |  |  |  |  |
| Test |  |  |  |  |
| Prod |  |  |  |  |

# Technical Architecture

## Environments

*Technologies in which the solution is deployed and what runs where (ex: google cloud, windows server, etc)*

Transformations with Python SDK is performed in a Jupyter Notebook in VS Code through a virtual environment including the cognite-sdk package.

# Operations

## Monitoring and alerting

**Technical:**

**Business:**

## Logging

## Job scheduling

## Roles and Responsibilities

*Contacts: Tech Leader, SME and Support.*

# Known Gaps and Issues