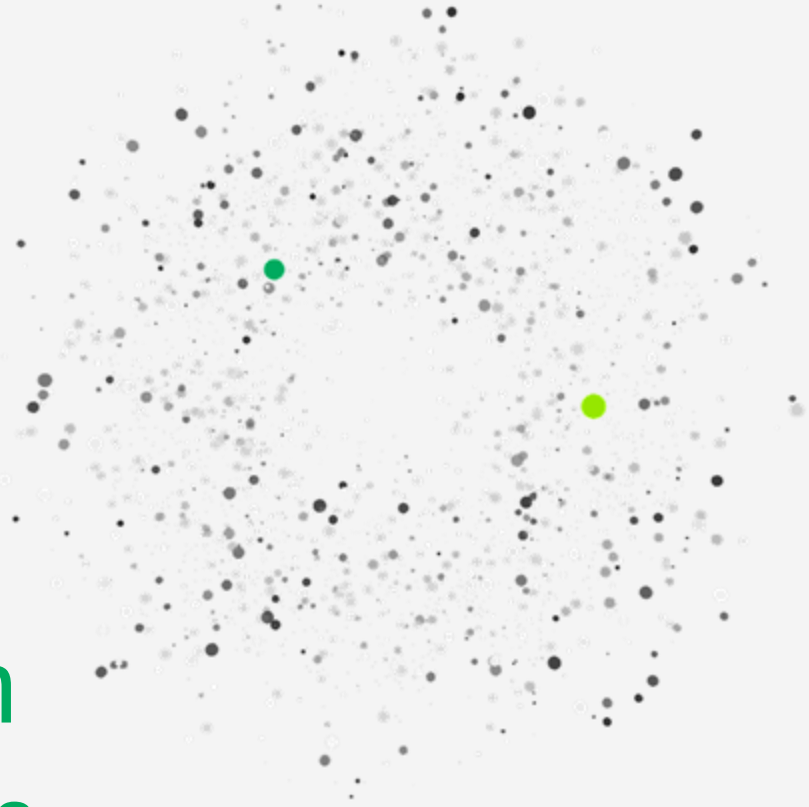


Data Pipeline Automatisation with Apache Airflow in Connection with the SAS LSAF Rest API

Frank Biedermann
10-Oct-2024

Agenda

- Background
- SAS Lifesciences Analytics Framework (LSAF)
- Apache Airflow
- SAS LSAF REST API
- Implementation Concept

01. Background

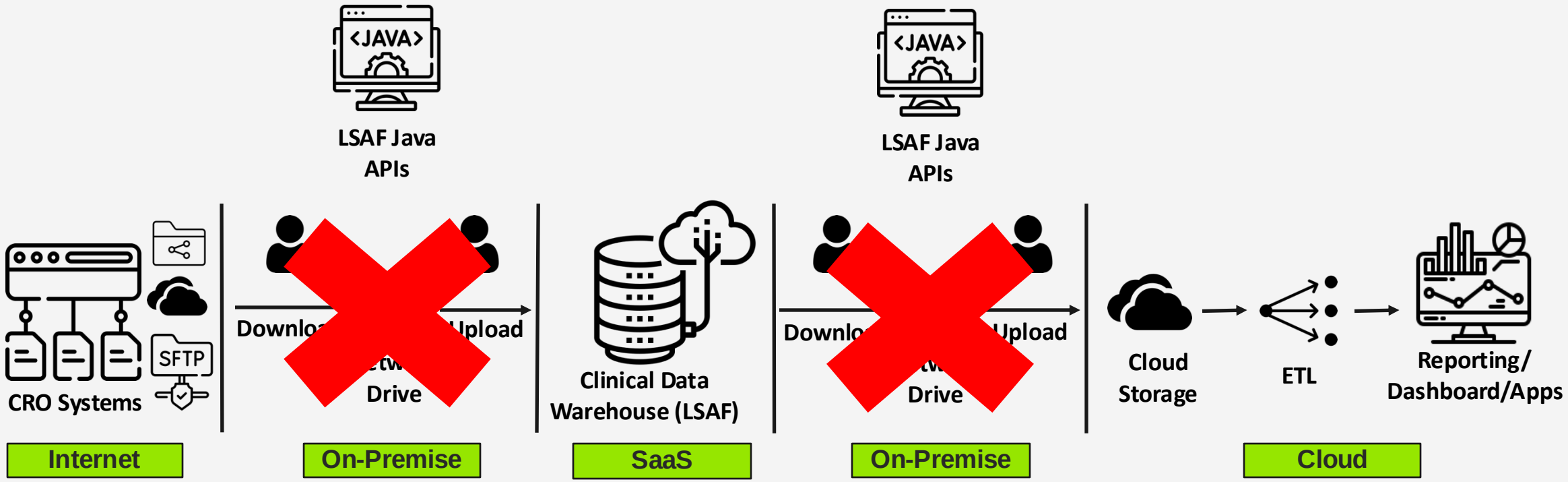
Background

- Study processes are completely outsourced
- A small team that has to process the data handling for several studies in parallel
- Data transfers from the source- to the target system are carried out manually

What is to be achieved:

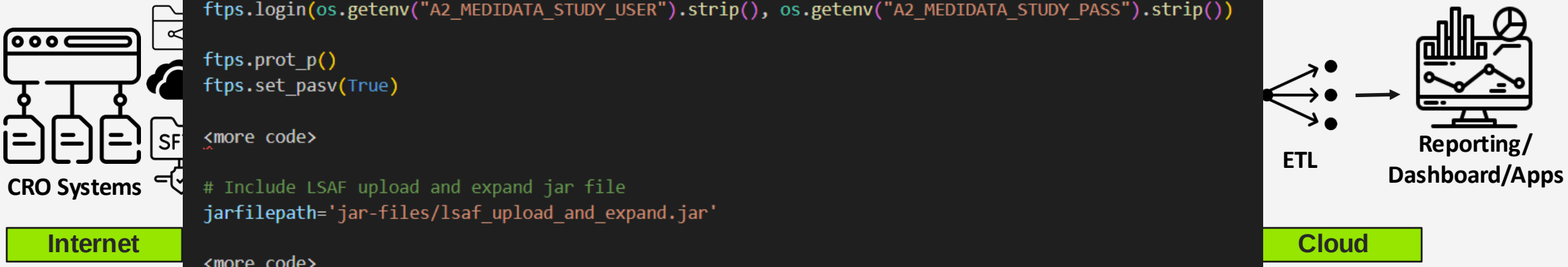
- Data transfers should take place automatically from the source- to the target system
- It should be a controlled process where it can be seen whether the data transfer has been completed, how often it has run, whether there were errors ...

Background


<JAVA>
LSAF Java APIs
<JAVA>
LSAF Java APIs
SFTP
CRO Systems
Drive
Clinical Data Warehouse (LSAF)
Drive
Cloud Storage
ETL
Reporting/ Dashboard/Apps
Internet
On-Premise
SaaS
On-Premise
Cloud

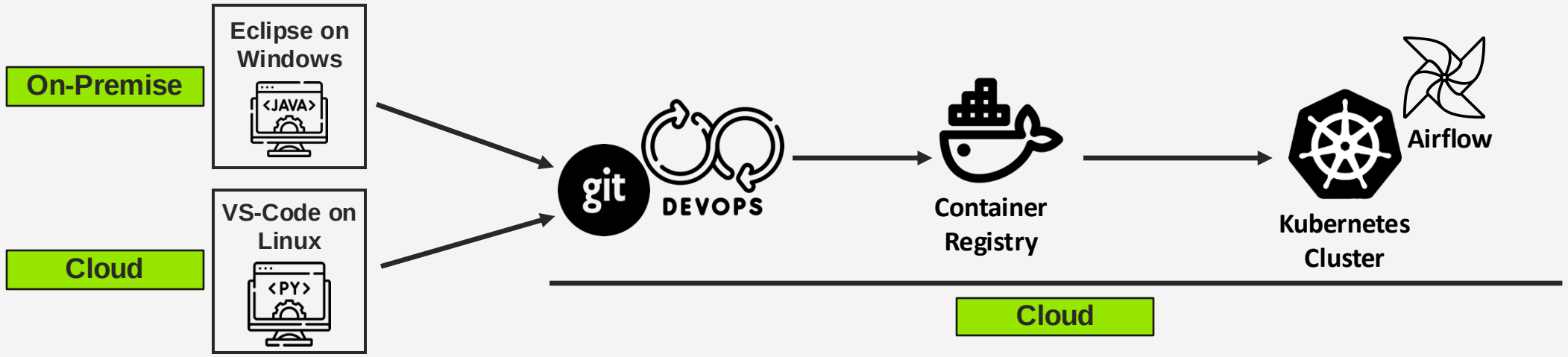
The first automation approach before the LSAF Rest APIs

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import pandas as pd, subprocess, os
from azure.storage.fileshare import ShareServiceClient, ShareFileClient
from azure.storage.file import FileService
from ftp_tls import ImplicitFTP_TLS

<more python modules>

# Download files from Medidata FTP Server
ftps = ImplicitFTP_TLS()
ftps.connect(host=os.getenv("A2_MEDIDATA_STUDY_HOST").strip(), port=990, timeout=5)
ftps.login(os.getenv("A2_MEDIDATA_STUDY_USER").strip(), os.getenv("A2_MEDIDATA_STUDY_PASS").strip())

ftps.prot_p()
ftps.set_pasv(True)

<more code>

# Include LSAF upload and expand jar file
jarfilepath='jar-files/lsaf_upload_and_expand.jar'

<more code>

# Upload to LASF
inpath=row.zip_filename
outpath=row.target_path

subprocess.call(['java', '-jar', jarfilepath, inpath, outpath], stderr=subprocess.DEVNULL)
```



What were the reasons for improving the existing process

- Pipeline tool development with two different programming languages / development tools
- Internal fewer programmers with Java experience but more with Python skills
- Due to the update to the latest LSAF version, we would also have had to update the LSAF Java APIs in the tools and make code adjustments
- The programming concept also does not fit perfectly with our DevOps CI/CD approach



02. SAS LSAF

SAS Lifesciences Analytics Framework (LSAF)

- LSAF is an integrated system for managing, analyzing, and reporting clinical research information. It provides a centralised clinical trial data management environment with version control, audit trails and role-based access capabilities to ensure compliance with regulations such as 21 CFR Part 11.

- It has been developed with regulatory compliance in mind and meets the stringent legal requirements of organisations such as the FDA and EMA.

- Key Features
  - Version control, E-Signatures and detailed audit trail features for tracking the repository content
  - Role-based controlled access
  - Stores and processes data in SAS datasets or other formats that SAS can process
  - Provides personal workspaces for each user
  - Includes an Integrated Development Environment (IDE) for SAS (R) program development
  - Built-in workflows and job scheduler for process automation
  - Can be extended using JAVA-, SAS Macro-, REST APIs
  - Supports data autoloading from various sources via sFTP and HTTPS

03. Apache Airflow

What is Apache Airflow

- Initially developed by Airbnb 2015, Apache incubator project 2016 and a top-level project since 2019
- Open-source workflow management platform for data engineering pipelines
- Written in Python; Workflows are created via Python Script it is designed under the principle of "configuration as code“
- Directed acyclic graphs (DAGs) is used for to manage workflow orchestration
- Wide range of Airflow connectors and providers from/to external systems

What You Can Do with Apache Airflow

1. Automate Workflows
- Define workflows as Directed Acyclic Graphs (DAGs)
- Automate complex task sequences with ease

2. Schedule & Manage Tasks
- Set up task schedules with flexible intervals (e.g., hourly, daily)
- Monitor and track execution in real-time

3. Handle Dependencies
- Easily manage task dependencies to control execution order
- Ensure tasks run in the right sequence, even across systems

4. Scale Workflows
- Scale from simple to complex pipelines across multiple nodes
- Airflow’s extensible architecture supports custom plugins and integrations

What You Can Do with Apache Airflow

5. Integrate Seamlessly

- Integrate with databases, cloud services, and third-party APIs
- Use built-in operators to interact with Python, Bash, SQL, and more

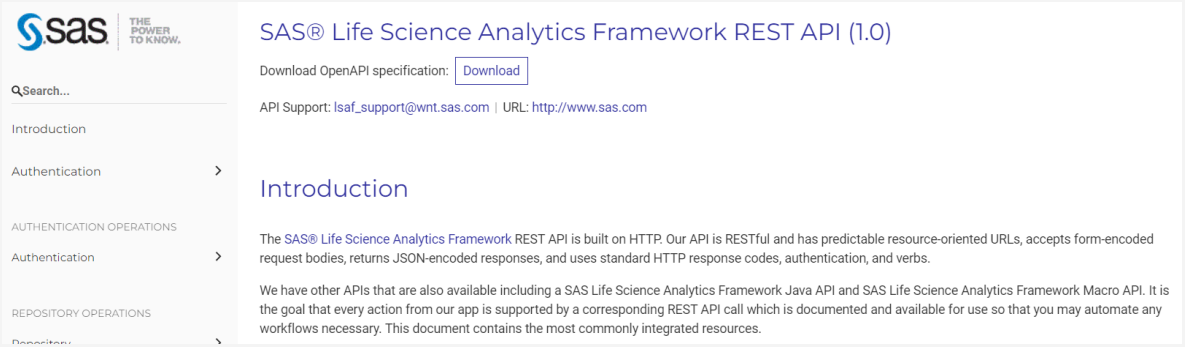
6. Track & Retry

- Get detailed logs of task execution
- Built-in retry mechanisms for failed tasks with alerts

04. SAS LSAF REST API

SAS LSAF REST API

- The REST API is available from SAS Life Science Analytics Framework version 5.4.1 in version 1.0
- The typical approach is to include the new functionality within the SAS Life Science Analytics Framework installation process
- With the SAS Life Science Analytics Framework version 5.4.2 (September 2024), the REST API was also updated to version 1.1, the version 1.1 is a patch, no new functions are added and supports only LSAF version 5.4.2
- The REST API documentation is available for LSAF customers via: https://<CUSTOMER_URL>/lsaf/api/reference



SAS LSAF REST API

- The REST API in version 1.0 already covers a large part of LSAF functions but is not yet at the same function level as the JAVA API

- The functions provided were sufficient for our use case, but a function for deleting files/folders would have been very helpful, which is currently not available in the functional scope of the REST API 1.0 / 1.1

- You need experience in dealing with REST APIs the documentation currently only provides examples for curl and SAS, but not for other programming languages e.g. R or Python

- The API Documentation is a pure text documentation with examples, it is not a Swagger documentation with API testing tool

05. Implementation Concept

Implementation Concept

- Check whether user access rights need to be adjusted in LSAF to use the Rest API
  Privilege: Access WebDav + password for Electronic Signature and WebDav Access must be set

- Identify the REST API function needed to replace the Java API uploader
  Authentication operations: Encrypt password, Log on, Log off
  Repository operations: Get file properties, Get container properties, Upload file

- Creation of a Python program to test the REST API calls

- Identification of potential security issues
  WebDav Username + Password in plain text

- Design and develop a concept to reuse REST API calls

- Design an develop a deployment concept

Implementation Concept

- Design and develop a concept to reuse REST API calls

The functions are saved in a separate python file: **cdw_utils.py**

The following functions are currently part of the file:

- **cdw_logon:** takes over the Encrypt Password and the Log On call
- **cdw_logoff:** takes over the Log Off call
- **cdw_get_file_properties:** takes over the Get File Properties call and is also used to find out if a file exists
- **cdw_get_container_properties:** takes over the Get Container Properties call and is also used to find out if a file exists
- **cdw_upload_file:** takes over the Upload File call and also takes over the setting of the correct file content type in LSAF
- **cdw_get_child:** takes over the Get Children call
- **cdw_download_file:** takes over the Download File call
- **cdw_execute_job:** takes over the Run job call

All functions can be controlled so that they run in the repository or in the workspace.

CDW = Clinical Data Warehouse

Implementation Concept

- **Function:** cdw_logon and cdw_logoff

```python
import os, requests, pandas as pd, time

def cdw_logon():
    global headers

    r = requests.get(os.getenv("A2_RDAE_LSAF_PROD_URL").strip()+'encrypt', auth=(os.getenv("A2_RDAE_LSAF_PROD_USER").strip(), os.getenv("A2_RDAE_LSAF_PROD_PASS").strip()))
    encrypdpw = (r.content)

    if r.status_code == 200:
        r = requests.post(os.getenv("A2_RDAE_LSAF_PROD_URL").strip()+'logon', auth=(os.getenv("A2_RDAE_LSAF_PROD_USER").strip(), encrypdpw))
        xauthtoken = (r.headers['X-Auth-Token'])
        headers = {'X-Auth-Token': xauthtoken}
        print("Note: Successfully logged on to LSAF")
        return headers
    else:
        print('Error: Not successfully logged on to LSAF')


def cdw_logoff():
    r = requests.post(os.getenv("A2_RDAE_LSAF_PROD_URL").strip()+'logoff',  headers=headers)
    if r.status_code == 200:
        print("Note: Successfully logged off from LSAF")
    else:
        print('Error: Not successfully logged out from LSAF')
```

CDW = Clinical Data Warehouse

Implementation Concept

- **Function:** cdw_get_file_properties and cdw_get_container_properties

```
def cdw_get_file_properties(_lsafpath, _lsaffilename):
    params = {'component': 'properties',}
    r = requests.get(os.getenv("A2_RDAE_LSAF_PROD_URL").strip()+'repository/files'+_lsafpath+'/'+_lsaffilename, params=params, headers=headers)
    getproperties = pd.json_normalize(r.json())
    if r.status_code == 200:
        print('Note: Successfully get properties for '+_lsaffilename+' from LSAF')
        return getproperties
    elif r.status_code == 404:
        getproperties=getproperties[['httpStatusCode', 'message', 'details']]
        print('Error: Not successfully get properties for '+_lsaffilename+' from LSAF')
        print('Error: Message = '+getproperties['httpStatusCode'].to_string(index=False)+' '+getproperties['message'].to_string(index=False))


def cdw_get_container_properties(_lsafpath):
    params = {'component': 'properties',}
    r = requests.get(os.getenv("A2_RDAE_LSAF_PROD_URL").strip()+'repository/containers'+_lsafpath, params=params, headers=headers)
    getproperties = pd.json_normalize(r.json())
    if r.status_code == 200:
        print('Note: Successfully get properties for '+_lsafpath+' from LSAF')
        return getproperties
    elif r.status_code == 404:
        getproperties=getproperties[['httpStatusCode', 'message', 'details']]
        print('Error: Not successfully get properties for '+_lsafpath+' from LSAF')
        print('Error: Message = '+getproperties['httpStatusCode'].to_string(index=False)+' '+getproperties['message'].to_string(index=False))
```

CDW = Clinical Data Warehouse

Implementation Concept

- **Function:** cdw_upload_file

```python
def cdw_upload_file(_lsafpath, _lsaffilename, localpath):
    _fileName, _fileExtension = os.path.splitext(_lsaffilename)
    _fileExtension=_fileExtension.replace('.','').strip()

    # Set file content type
    _fileExtensionErr=0
    if _fileExtension == 'xlsx':
        _lsafUplContType = 'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet'
    elif _fileExtension == 'xls':
        _lsafUplContType = 'application/vnd.ms-excel'
    elif _fileExtension == 'xlsm':
        _lsafUplContType = 'application/vnd.ms-excel.sheet.macroenabled.12'
    elif _fileExtension == 'csv':
        _lsafUplContType = 'text/csv'
    elif _fileExtension == 'txt':
        _lsafUplContType = 'text/plain'
    elif _fileExtension == 'docx':
        _lsafUplContType = 'application/vnd.openxmlformats-officedocument.wordprocessingml.document'
    elif _fileExtension == 'doc':
        _lsafUplContType = 'application/msword'
    elif _fileExtension == 'rtf':
        _lsafUplContType = 'application/rtf'
    elif _fileExtension == 'pdf':
        _lsafUplContType = 'application/pdf'
    elif _fileExtension == 'msg':
        _lsafUplContType = 'application/vnd.ms-outlook'
    elif _fileExtension == 'pptx':
        _lsafUplContType = 'application/vnd.openxmlformats-officedocument.presentationml.presentation'
    elif _fileExtension == 'ppt':
        _lsafUplContType = 'application/vnd.ms-powerpoint'
    elif _fileExtension == 'pptm':
        _lsafUplContType = 'application/vnd.ms-powerpoint.presentation.macroenabled.12'
    elif _fileExtension == 'xml':
        _lsafUplContType = 'application/xml'
    elif _fileExtension == 'log':
        _lsafUplContType = 'application/octet-stream'
    elif _fileExtension == 'lst':
        _lsafUplContType = 'application/octet-stream'
    elif _fileExtension == 'sas':
        _lsafUplContType = 'application/x-sas'
    elif _fileExtension == 'sas7bcat':
        _lsafUplContType = 'application/x-sas-catalog'
    elif _fileExtension == 'sas7bdat':
        _lsafUplContType = 'application/x-sas-data'
    elif _fileExtension == 'cpt':
        _lsafUplContType = 'application/octet-stream'
    elif _fileExtension == 'xpt':
        _lsafUplContType = 'application/octet-stream'
    else:
        print('Error: Unknown file extension')
        _fileExtensionErr=1

    if _fileExtensionErr==0:
        _lsafUplStatement=(os.getenv("A2_RDAE_LSAF_PROD_URL").strip()+'repository/files'+_lsafpath+'/'+_lsaffilename +"?action=upload&overwrite=true")
        _lsafUplFile={'uploadFile':(_lsaffilename, open(localpath+'/'+_lsaffilename, 'rb'), _lsafUplContType)}

        uploadExecute = requests.put(_lsafUplStatement, files=_lsafUplFile, headers=headers)

        print("Note: Upload status code for file: "+ _lsafpath+'/'+_lsaffilename + " = " + str(uploadExecute.status_code))
```

We take the single file upload because we have noticed that if content type is not specified it is not set correctly by LSAF.
Furthermore, there is a bug with the Upload and Expand ZIP call in the REST API 1.0 (SAS Problem Note 70918), which has now been fixed with version 1.1.

CDW = Clinical Data Warehouse

Implementation Concept

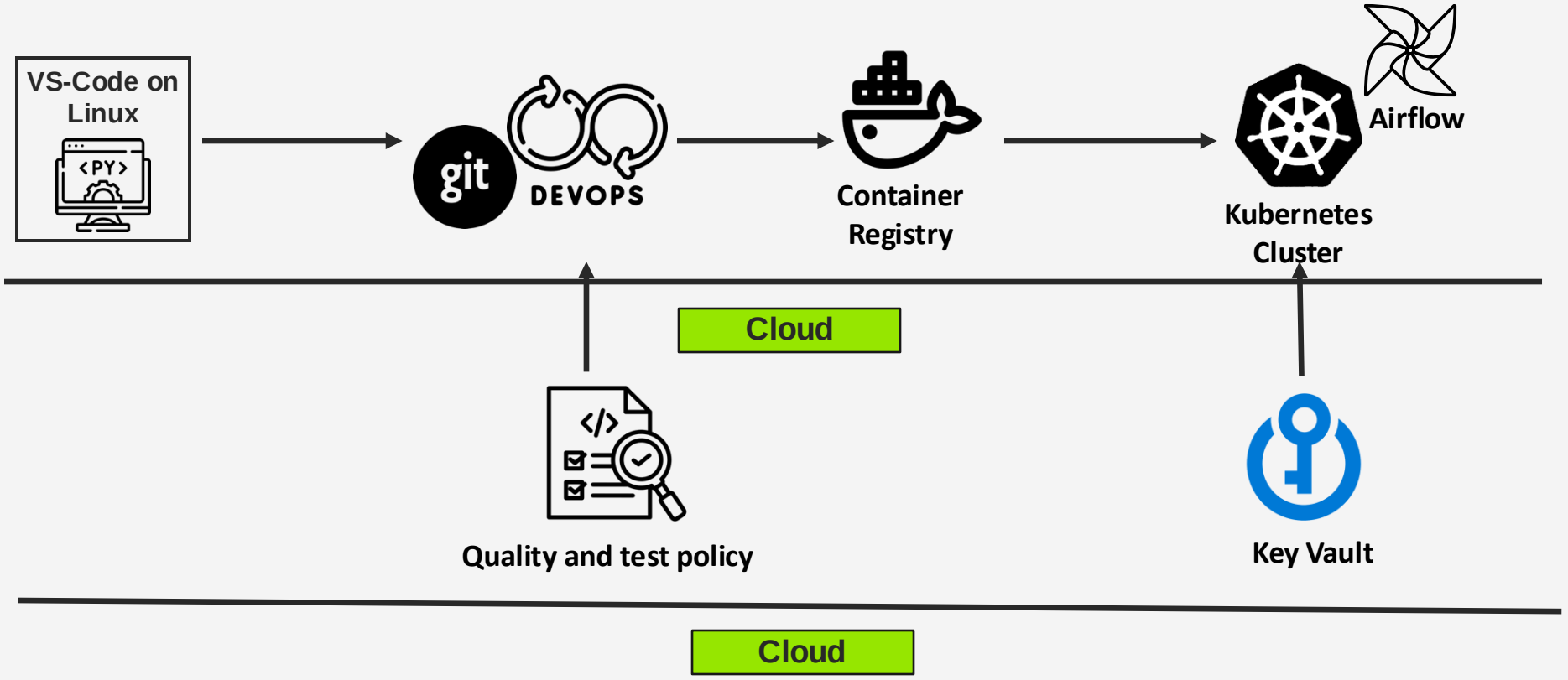
- Example of how to call the functions in other python programs

```
import pandas as pd, subprocess, os
from azure.storage.fileshare import ShareServiceClient, ShareClient, ShareDirectoryClient, ShareFileClient
from azure.storage.file import FileService
from ftp_tls import ImplicitFTP_TLS
from cdw_utils import cdw_logon, cdw_logoff, cdw_upload_file, cdw_get_container_properties

<more python modules>

# Download files from Medidata SFTP Server
ftps = ImplicitFTP_TLS()
ftps.connect(host=os.getenv("A2_MEDIDATA_STUDY_HOST").strip(), port=990, timeout=5)
ftps.login(os.getenv("A2_MEDIDATA_STUDY_USER").strip(), os.getenv("A2_MEDIDATA_STUDY_PASS").strip())

ftps.prot_p()
ftps.set_pasv(True)

<more code>

# LSAF login
headers = cdw_logon()

<more code>

# Get LSAF container properties
getproperties = cdw_get_container_properties('<path in lsaf>')

<more code>

# Upload files to LSAF
<loop>
cdw_upload_file('<path in lsaf>', '<filename>', '<localpath>')
</loop>

<more code>

# LSAF logout
cdw_logoff()
```

CDW = Clinical Data Warehouse

Implementation Concept

- Design an develop a deployment concept

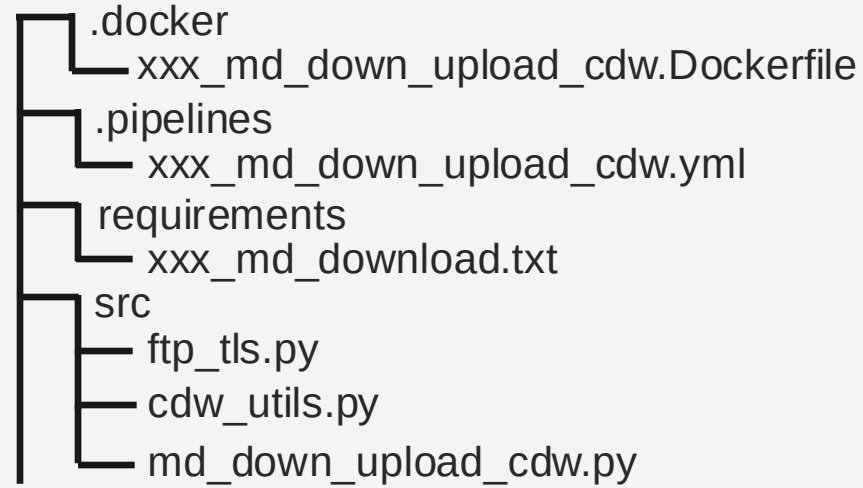


Implementation Concept – GIT and Docker

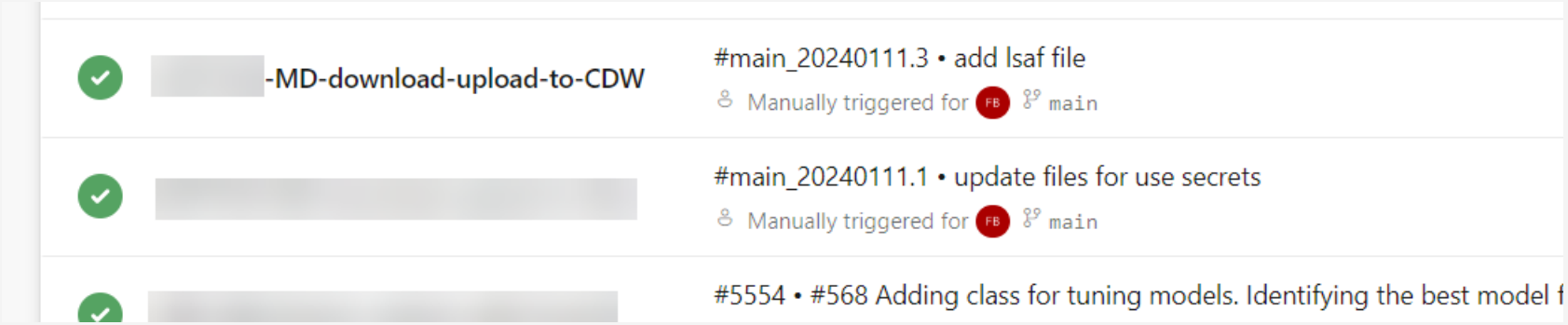
xxx_md_down_upload_cdw.Dockerfile

```
FROM python:3.10.10-slim-buster

RUN apt-get update && apt-get install -y --no-install-recommends \
    gnupg2 curl apt-transport-https g++ gcc \
    unixodbc-dev libgssapi-krb5-2 jq vim wget \
    && rm -rf /var/lib/apt/lists/*

ADD src/requirements/ md_download.txt requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Copy files
WORKDIR /src
COPY src/ftp_tls.py /src
COPY src/cdw_utils.py /src
COPY src/md_down_upload_cdw.py /src
```

GIT Repository Structure



Azure DevOps CI/CD Pipeline

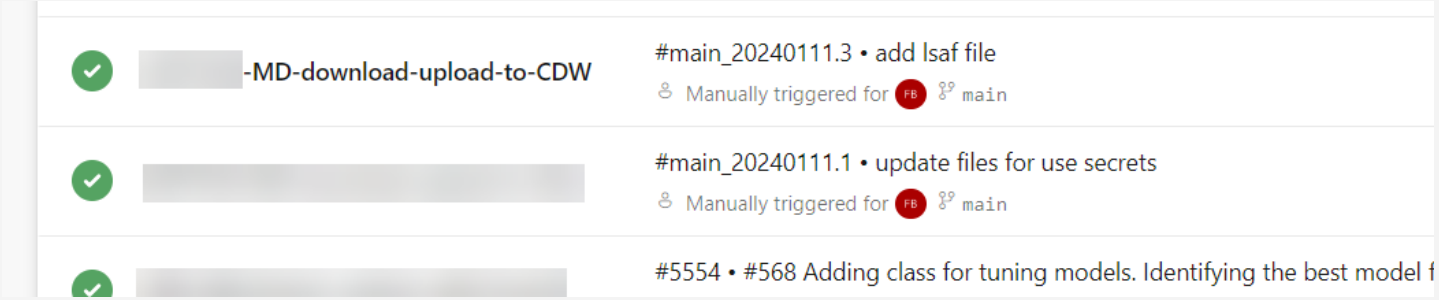


Implementation Concept – Azure DevOps

xxx_md_down_upload_cdw.yml

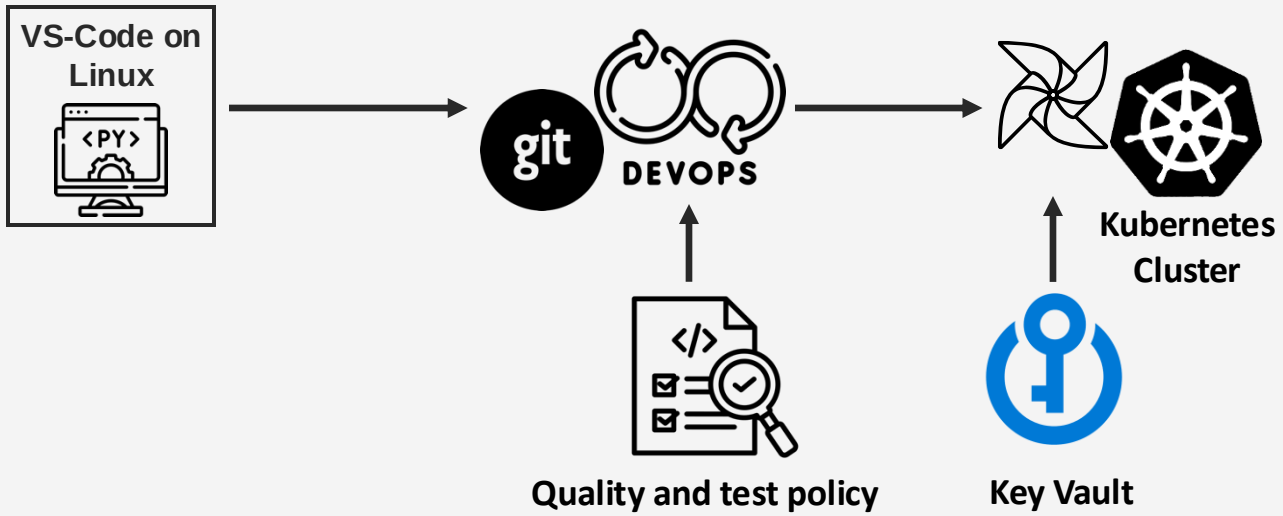
```yaml
name: $(SourceBranchName)_$(Date:yyyyMMdd)$(Rev:.r)

pr: none
trigger: none

variables:
  - group: build_jobs_vg

pool:
  vmImage: ubuntu-latest

stages:
- stage: 'CreateImage'
  displayName: 'Create Image'
  jobs:
  - job: "BuildAndPushImage"
    displayName: "Build and push base image"
    timeoutInMinutes: 0
    steps:
    - checkout: self
      lfs: true
    - task: Docker@2
      displayName: Login to ACR
      inputs:
        command: login
        containerRegistry: $(REGISTRY_SVC)
    - task: Docker@2
      displayName: Build and push an image to container registry
      inputs:
        command: buildAndPush
        repository:  _md_down_upload_cdw
        buildContext: .
        dockerfile: src/.docker/ _md_down_upload_cdw.Dockerfile
        containerRegistry: $(REGISTRY_SVC)
        tags: |
          $(Build.SourceVersion)
          latest
          v1
    - task: PublishBuildArtifacts@1
      displayName: 'Publish Artifact: drop'
```

Azure DevOps CI/CD Pipeline



Implementation Concept – Apache Airflow

Airflow requires its own GIT repository, in which dags, plugins ... must be stored in a predefined structure.



Furthermore, another Azure DevOps CI/CD pipeline is required for this, in our case this process is automated, i.e. as soon as a new DAG has been pushed into a feature branch in GIT, it must be released by a reviewer, as soon as this has happened a merge into the main branch is triggered, the CI/CD pipeline runs and creates a new AirFlow workflow based on the DAG information.

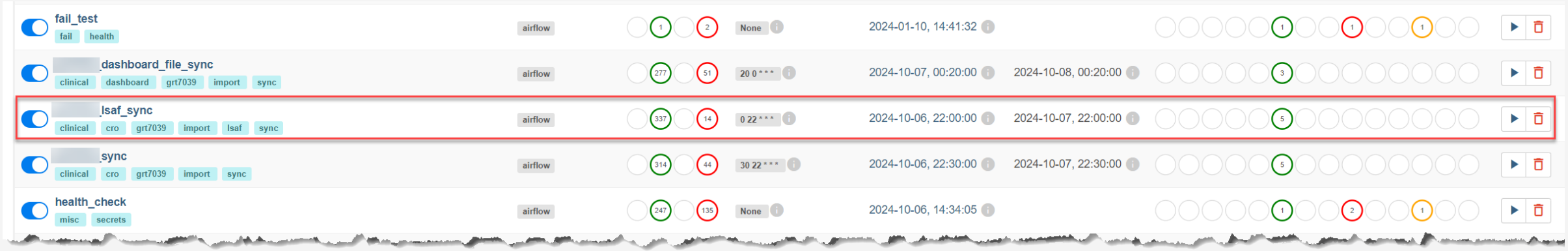
Implementation Concept – Apache Airflow

DAG file: xxx_md_down_upload_cdw.py

```python
import airflow.models.baseoperator as airopts
from airflow import DAG
from grt.ops import kube_script_op, st_op
from grt.scheduling import get_schedule
from grt.utils import default_args_for

# Dag
args = default_args_for(
    desc="        raw data sync for the studies                    to LSAF trigger updates once daily at 11 P.M..",
    schedule=get_schedule("        _lsaf_sync"),
    tags=["clinical", "        ', "cro", "lsaf", "import", "sync"],
)
with DAG("        _lsaf_sync", **args):
    start_op, end_op = st_op()

    _01_op = kube_script_op(
        identifier="        01",
        image="a2acrrdae.azurecr.io/        _md_down_upload_cdw:latest",
        cmds=["python"],
        arguments=["md_down_upload_cdw.py", "--trial-id=        01"],
    )

    _02_op = kube script_op(
        identifier="|        02",
        image="a2acrrdae.azurecr.io/        _md_down_upload_cdw:latest",
        cmds=["python"],
        arguments=["md_down_upload_cdw.py", "--trial-id=        02"],
    )

    _03_op = kube script_op(
        identifier="|        _03",
        image="a2acrrdae.azurecr.io/        _md_down_upload_cdw:latest",
        cmds=["python"],
        arguments=["md_down_upload_cdw.py", "--trial-id=        _03"],
    )

    # Dag composition
    trial_ops = [        _01_op,        _02_op,        _03_op]
    airopts.chain(start_op, trial_ops, end_op)
```

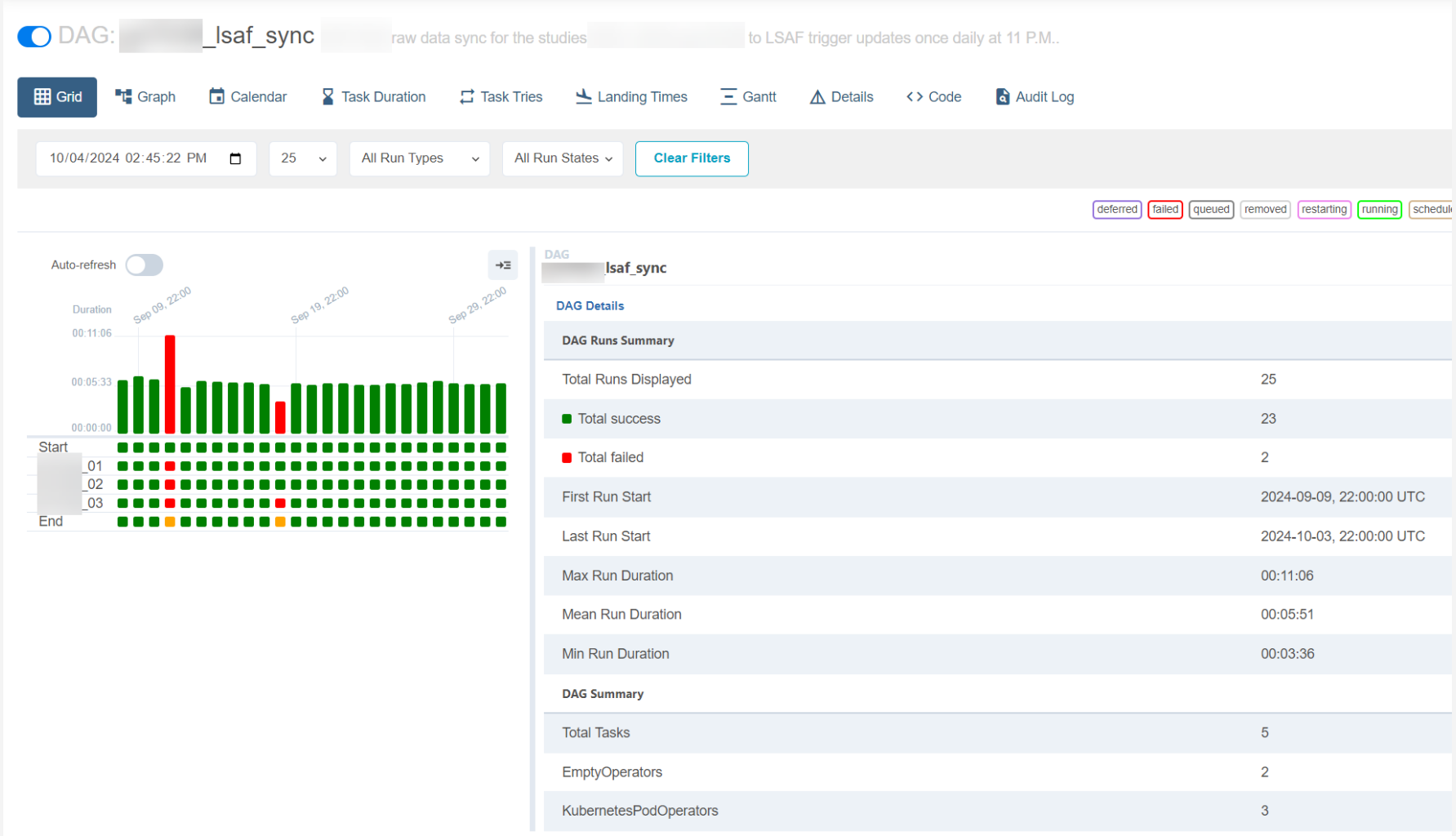
Implementation Concept – Apache Airflow

The Airflow Dashboard



Implementation Concept – Apache Airflow

The Airflow DAG view



Thank You


GRÜNENTHAL