# DAGify - Enterprise Scheduler Migration Accelerator

Konrad Schieban, Strategic Cloud Engineer, Google

Tim Hiatt, Staff Technical Solutions Consultant, Google

Google Cloud

# DAGify Team

Konrad Schieban

Tim Hiatt

Joanna Rajaseharan

Harish S

Shreya Prabhu

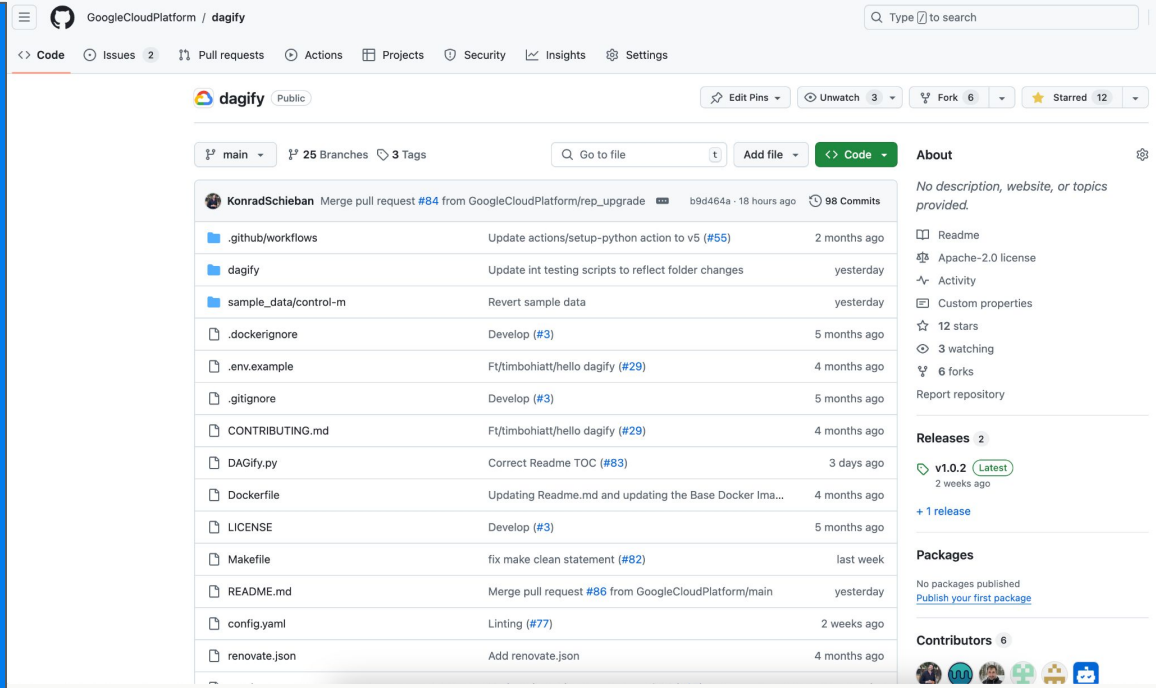
# Introduction

# What are we solving?

- Large Cloud Migrations often involve migrating enterprise workflow management systems.
- Enterprises have desire to move away from these systems when migrating to the Cloud.
- Migrating these workflows to Apache Airflow (e.g. on Cloud Composer) may require 1000's of human hours to convert proprietary scheduler formats to Python code.
- A burden that is too great for organizations to consider at scale.

# Introducing DAGify

- Convert exported workflow configuration from XML to Python Code for Airflow.
- Highly extensible with minimal to no code modification. Conversion controlled using YAML files.
- Open Source and encourages community collaboration.
- Provides a report that provides summary of the conversion and which job types could not be migrated.




https://goo.gle/dagify-github

# Show me DAGify!

```
<JOB
JOBISN="0" APPLICATION="application-xx"
SUB_APPLICATION="australia-region" MEMNAME="app-xx-aus-reg-001"
JOBNAME="app-xx-aus-reg-001-daily" DESCRIPTION="Daily Application Run
Australia Region" CREATED_BY="Google PSO Demo App"
RUN_AS="gpso_user" PRIORITY="AA" CRITICAL="0" TASKTYPE="Command"
CYCLIC="0" NODEID="gcp.pso.demo.app.google.com" INTERVAL="00001M"
CMDLINE="exebatch.sh aus.google.com daily" CONFIRM="0" RETRO="0"
MAXWAIT="0" MAXRERUN="0" AUTOARCH="0" MAXDAYS="0" MAXRUNS="0"
WEEKDAYS="1,2,3,4,5" JAN="1" FEB="1" MAR="1" APR="1" MAY="1" JUN="1"
JUL="1" AUG="1" SEP="1" OCT="1" NOV="1" DEC="1" DAYS_AND_OR="O"
SHIFT="Ignore Job" SHIFTNUM="+00" SYSDB="0" JOBS_IN_GROUP="00000"
IND_CYCLIC="S" CREATION_USER="gpso_user"
CREATION_DATE="20160112" CREATION_TIME="103502"
CHANGE_USERID="emp8" CHANGE_DATE="20170714"
CHANGE_TIME="181328" RULE_BASED_CALENDAR_RELATIONSHIP="O"
APPL_TYPE="OS" MULTY_AGENT="N" USE_INSTREAM_JCL="N"
VERSION_OPCODE="N" IS_CURRENT_VERSION="Y" VERSION_SERIAL="3"
VERSION_HOST="gcp.online.com" CYCLIC_TOLERANCE="0"
CYCLIC_TYPE="C" PARENT_FOLDER="gcp-pso-core-apps">
   <VARIABLE NAME="%%SCRIPT_DIR" VALUE="/data/scripts/aus-scripts" />
   <VARIABLE NAME="%%DATE" VALUE="%%$YEAR-%%MNTH-%%DAY" />
   <SHOUT WHEN="NOTOK" URGENCY="V" DEST="ECS" MESSAGE="%%
failed, Needs Investigation" />
   <INCOND NAME="dependent-job-xxx-1" ODATE="ODAT" AND_OR="A" />
   <OUTCOND NAME="task-complete-flag" ODATE="ODAT" SIGN="+" />
</JOB>
```

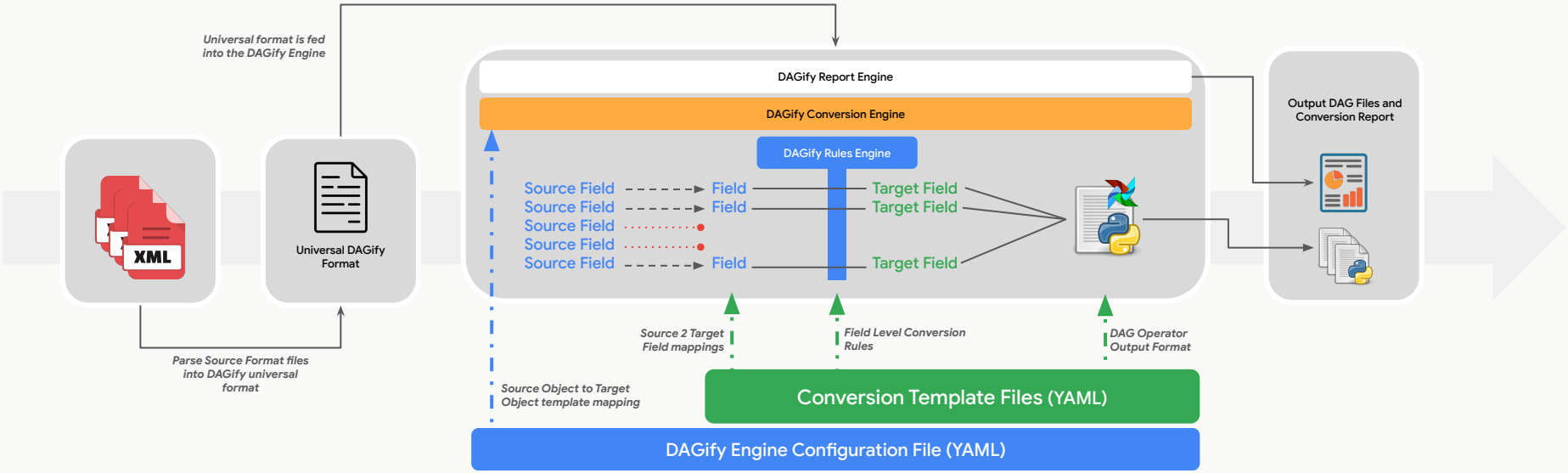
Input Source Workflow Definitions

```
import airflow
from airflow import DAG
from airflow.contrib.operators.ssh_operator
import SSHOperator,ComputeEngineSSHHook

with DAG(
   dag_id="JOB_DC_ONE",
   start_date=datetime.datetime( 2021, 1, 1),
) as dag:
# DAG Tasks
   JOB_APP_XX_AUS_REG_001_DAILY = SSHOperator(
       task_id='JOB_APP_XX_AUS_REG_001_DAILY' ,
       ssh_hook=Compute Engine SSH Hook(
         instance_name=UNKNOWN,
         zone=UNKNOWN,
         project_id=UNKNOWN,
         use_oslogin= True,
         command= "exebatch.sh aus.google.com
daily",
       dag=dag)
```

Output Native Python DAG for Airflow

# DAGify under the hood



# DAGify Engine

DAGify is available as CLI tool that developers run from source or using a Container. For conversion of XML files, it executes the DAGify Engine.

**The DAGify Engine**

| Templates | Rules Engine | Configuration Files | Reporting Engine |
|---|---|---|---|
| Contain the source to target mappings for conversions | Small centralized python functions applied as on a per field mapping | User defined source type to template mapping | Captures detailed logs across the end to end conversion process |
| DAGify team and community contributed | Provided in DAGify template applied during conversion | YAML format for simplicity. | Detailed report output with all the captured details |
| YAML formatted for simplicity and extensibility and easy management of native python packages and version | Community extensible and highly customizable easily testable | Samples and defaults provided out of the box with DAGify | Provides developers with information on outstanding task and conversion challenges |

# Feature Overview

# DAG Dividers

If you have a large Control-M XML file you may consider splitting the Airflow tasks into multiple DAGs. DAGify has a command line parameter -d or --dag-divider which specifies by which XML attribute jobs should be divided by.

Examples:

```
./DAGify -d FOLDER
./DAGify -d APPLICATION
./DAGify -d SUB_APPLICATION
```

DAGify uses **ExternalTaskSensors** and **ExternalTaskMarkers** to implement dependencies.

# Cross-DAG Dependencies

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- # Copyright 2024 Google LLC
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
-->
<DEFTABLE xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="Folder.xsd">
<SMART_FOLDER FOLDER_NAME="fx_fld_001">
        <!-- Folder 001, Application 001, Sub Application 001, Job 001 -->
        <JOB APPLICATION="fx_fld_001_app_001" SUB_APPLICATION="fx_fld_001_app_001_subapp_001" JOBNAME="fx_fld_001_app_001_subapp_001_job_001"
                <OUTCOND NAME="fx_fld_001_app_001_subapp_001_job_001_ok" ODATE="ODAT" SIGN="+" />
        </JOB>
        <!-- Folder 001, Application 001, Sub Application 001, Job 002 -->
        <JOB APPLICATION="fx_fld_001_app_001" SUB_APPLICATION="fx_fld_001_app_001_subapp_001" JOBNAME="fx_fld_001_app_001_subapp_001_job_002"
                <INCOND NAME="fx_fld_001_app_001_subapp_001_job_001_ok" ODATE="ODAT" AND_OR="A" />
                <OUTCOND NAME="fx_fld_001_app_001_subapp_001_job_002_ok" ODATE="ODAT" SIGN="+" />
        </JOB>
        <!-- Folder 001, Application 002, Sub Application 001, Job 001 -->
        <JOB APPLICATION="fx_fld_001_app_002" SUB_APPLICATION="fx_fld_001_app_002_subapp_001" JOBNAME="fx_fld_001_app_002_subapp_001_job_001"
                <INCOND NAME="fx_fld_001_app_001_subapp_001_job_001_ok" ODATE="ODAT" AND_OR="A" />
                <INCOND NAME="fx_fld_001_app_001_subapp_001_job_002_ok" ODATE="ODAT" AND_OR="A" />
                <OUTCOND NAME="fx_fld_001_app_002_subapp_001_job_001_ok" ODATE="ODAT" SIGN="+" />
        </JOB>
</SMART_FOLDER>
</DEFTABLE>
```

`-d FOLDER_NAME` will generate a single DAG

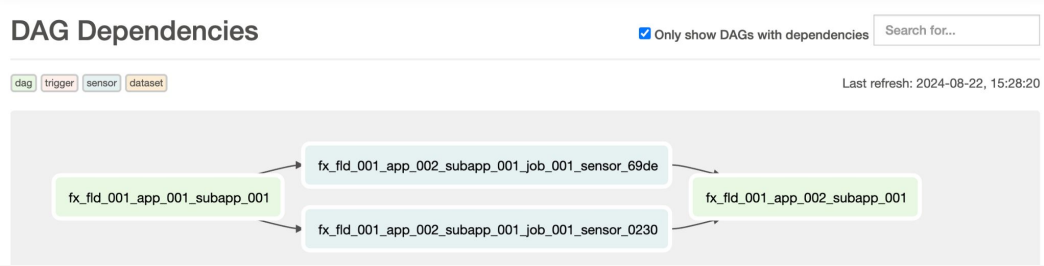
`-d APPLICATION` or `-d SUB_APPLICATION` will generate two DAGs

# Cross-DAG Dependencies

```python
# Airflow Upstream Task Dependencies (external dags)

fx_fld_001_app_002_subapp_001_job_001_sensor_69de = ExternalTaskSensor(
    task_id="fx_fld_001_app_002_subapp_001_job_001_sensor_69de",
    external_dag_id="fx_fld_001_app_001_subapp_001",
    external_task_id="fx_fld_001_app_001_subapp_001_job_001",
    dag=dag
)
fx_fld_001_app_002_subapp_001_job_001_sensor_69de >> fx_fld_001_app_002_subapp_001_job_001

fx_fld_001_app_002_subapp_001_job_001_sensor_0230 = ExternalTaskSensor(
    task_id="fx_fld_001_app_002_subapp_001_job_001_sensor_0230",
    external_dag_id="fx_fld_001_app_001_subapp_001",
    external_task_id="fx_fld_001_app_001_subapp_001_job_002",
    dag=dag
)
fx_fld_001_app_002_subapp_001_job_001_sensor_0230 >> fx_fld_001_app_002_subapp_001_job_001
```



# Scheduler Migration

*In Control-M each job can have its own schedule to run.*
*In Airflow,* the schedule is set at the DAG level, meaning all tasks within a DAG will adhere to the same schedule.

**Create Multiple DAGs:** The most straightforward solution is to create separate DAGs for tasks that need different schedules. Each DAG can have its own `schedule_interval`.

DAGify will convert the Control-M task schedule to a Cron expression and write it into the DAG definition.

**Attention:** DAGify will only take the schedule from the first job that it finds per dag and write it into the DAG definition.

# Reporting Engine

DAGify's reporting engine allows the generation of reports in txt and in JSON format.

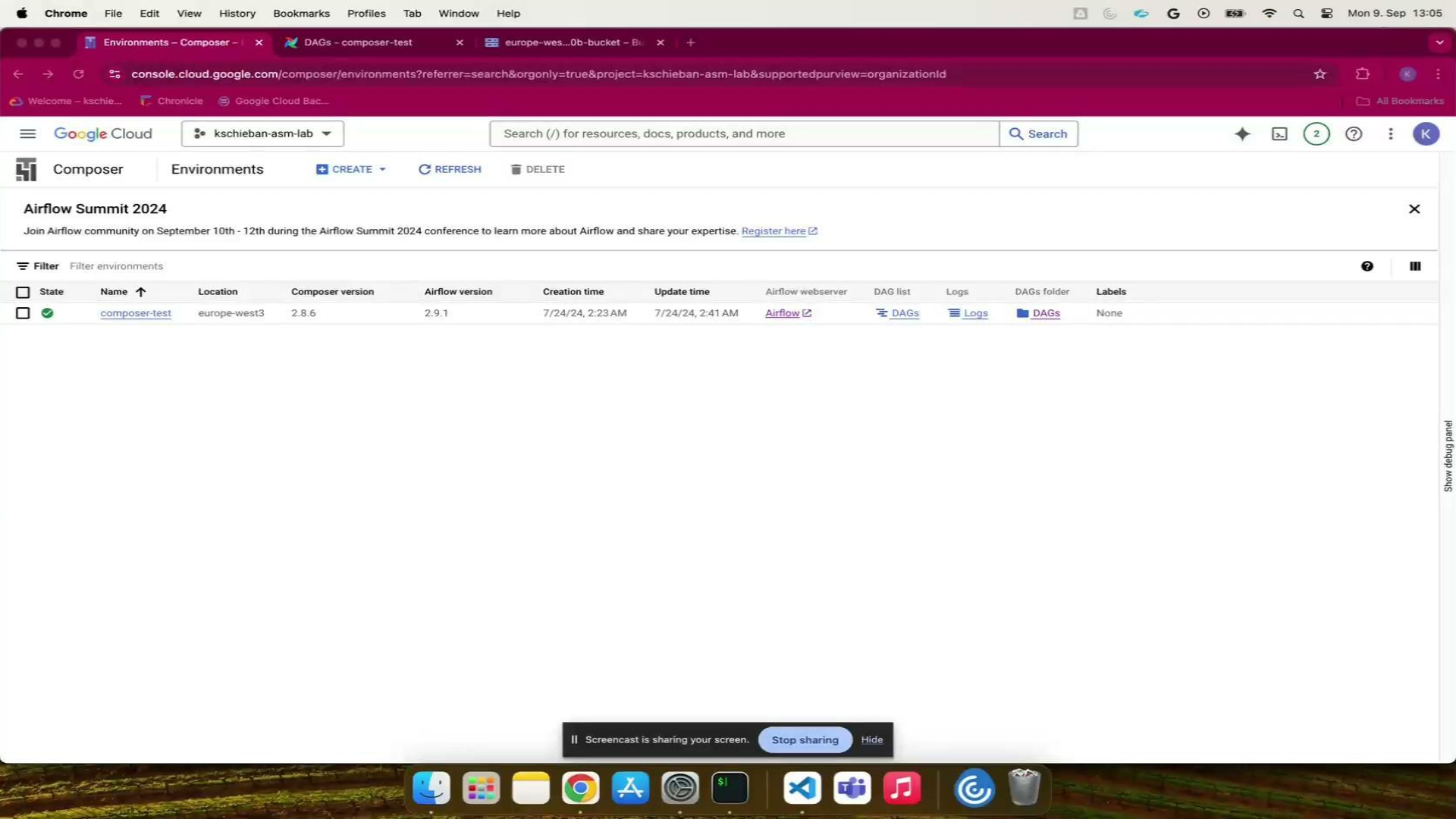
DAGify will for instance report on

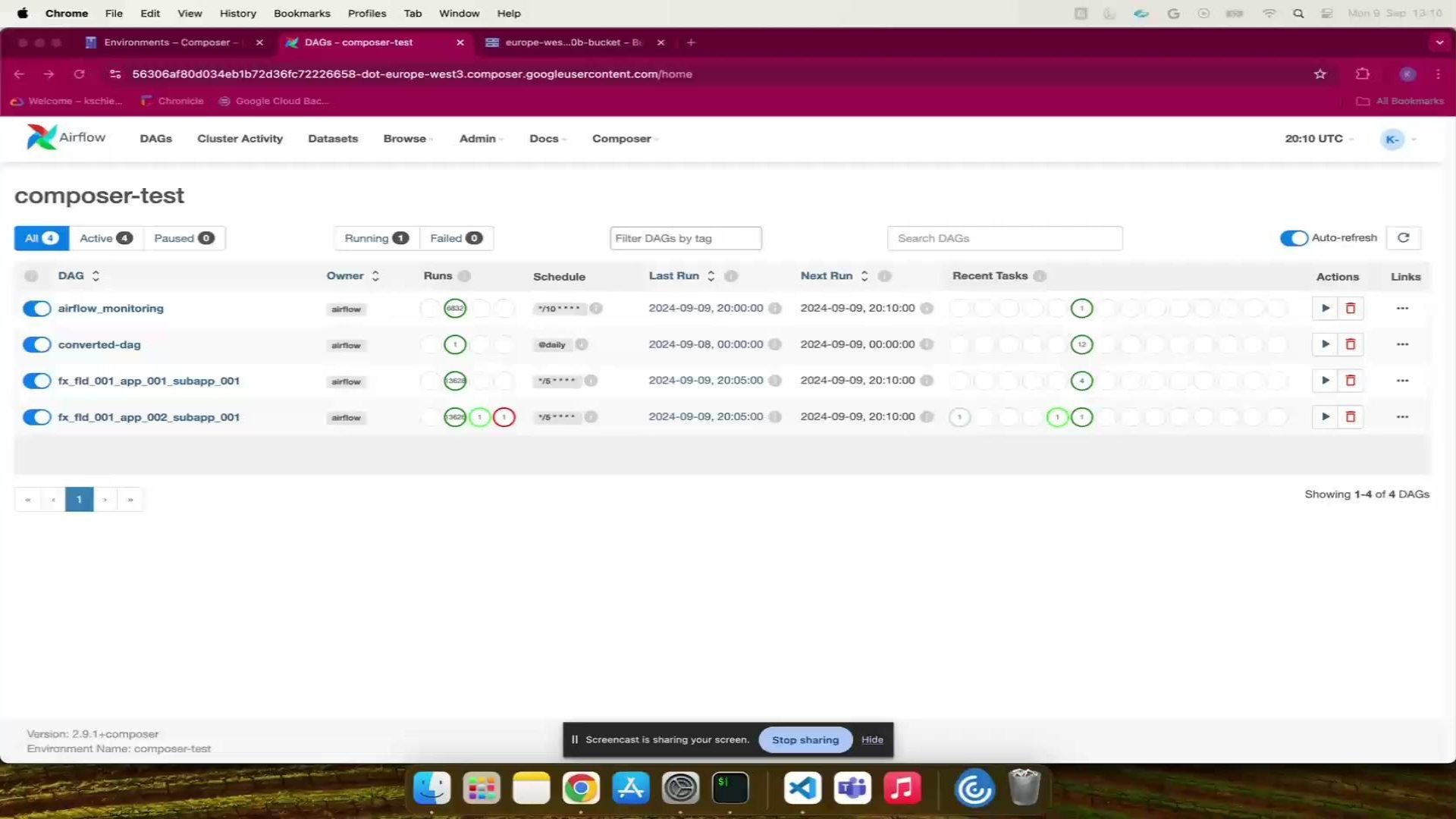
- amount of operations that it knows how to migrate
- amount of operations that it does not know how to migrate
- types of operations that it has found in the Control-M file
- the templates that are used for the conversion

```
Jobtypes converted: 1/2
Percentage of Jobtypes converted: 50.0%
Percentage of Jobtypes not converted: 50.0%
Jobs converted: 11/12
Percentage of Jobs converted: 91.67%
Percentage of Jobs not converted: 8.33%

+-------------------
```

# Demo


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon 9. Sep 13:05
Environments – Composer –
DAGs - composer-test
europe-wes...0b-bucket – Bu
console.cloud.google.com/composer/environments?referrer=search&orgonly=true&project=kschieban-asm-lab&supportedpurview=organizationId
Welcome – kschie...
Chronicle
Google Cloud Bac...
All Bookmarks
Google Cloud
kschieban-asm-lab
Search (/) for resources, docs, products, and more
Search
Composer
Environments
CREATE
REFRESH
DELETE
Airflow Summit 2024
Join Airflow community on September 10th - 12th during the Airflow Summit 2024 conference to learn more about Airflow and share your expertise. Register here
Filter
Filter environments
State
Name
Location
Composer version
Airflow version
Creation time
Update time
Airflow webserver
DAG list
Logs
DAGs folder
Labels
composer-test
europe-west3
2.8.6
2.9.1
7/24/24, 2:23 AM
7/24/24, 2:41 AM
Airflow
DAGs
Logs
DAGs
None
Screencast is sharing your screen.
Stop sharing
Hide
Show debug panel


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Environments – Composer –
DAGs - composer-test
europe-wes...0b-bucket – Bu
56306af80d034eb1b72d36fc72226658-dot-europe-west3.composer.googleusercontent.com/home
Welcome – kschie...
Chronicle
Google Cloud Bac...
All Bookmarks
Airflow
DAGs
Cluster Activity
Datasets
Browse
Admin
Docs
Composer
20:10 UTC
K-
composer-test
All 4
Active 4
Paused 0
Running 1
Failed 0
Filter DAGs by tag
Search DAGs
Auto-refresh
DAG
Owner
Runs
Schedule
Last Run
Next Run
Recent Tasks
Actions
Links
airflow_monitoring
airflow
6832
*/10 * * * *
2024-09-09, 20:00:00
2024-09-09, 20:10:00
converted-dag
airflow
1
@daily
2024-09-08, 00:00:00
2024-09-09, 00:00:00
12
fx_fld_001_app_001_subapp_001
airflow
3628
*/5 * * * *
2024-09-09, 20:05:00
2024-09-09, 20:10:00
4
fx_fld_001_app_002_subapp_001
airflow
3629
*/5 * * * *
2024-09-09, 20:05:00
2024-09-09, 20:10:00
« ‹ 1 › »
Showing 1-4 of 4 DAGs
Version: 2.9.1+composer
Environment Name: composer-test
Screencast is sharing your screen.
Stop sharing
Hide

# Questions?

**Contact:**

- linkedin.com/in/konrad-schieban-1730b986
- kschieban@google.com

**Blog:** https://goo.gle/dagify-blog
**Github:** https://goo.gle/dagify-github

Github

Blog