
May 23–27, 2022
AIRFLOW
SUMMIT

Airflow Summit 2022

Choosing Apache Airflow over other Proprietary Tools for your Data Orchestration needs

Parnab Basak
Solutions Architect, AWS

About me

Parnab Basak
Solutions Architect @Amazon Web Services

- Over 18+ of IT experience in Energy, Utilities, FinTech & GovTech…
- ~2 Years in AWS as an SA serving our WWPS customers
- Specialist SME for Amazon Managed Workflow for Apache Airflow (Amazon MWAA)
- Airflow (MWAA) Blogpost Author

In my spare time: **I evolve to be a Movie Buff**

(Have Amazon Prime, Netflix, HBO Max, Apple TV+, Peacock, Disney+ AND MORE….)

** Wish to become a movie critic some day **

Agenda

Airflow Summit 2022

01 **Data Orchestration**
Definition & Benefits

02 **Tools**
Proprietary and OSS

03 **Compare**
Between my Pick of 4

04 **Similarities**
All are good tools

05 **Differences**
There are a few

06 **Unique Features**
Of Apache Airflow

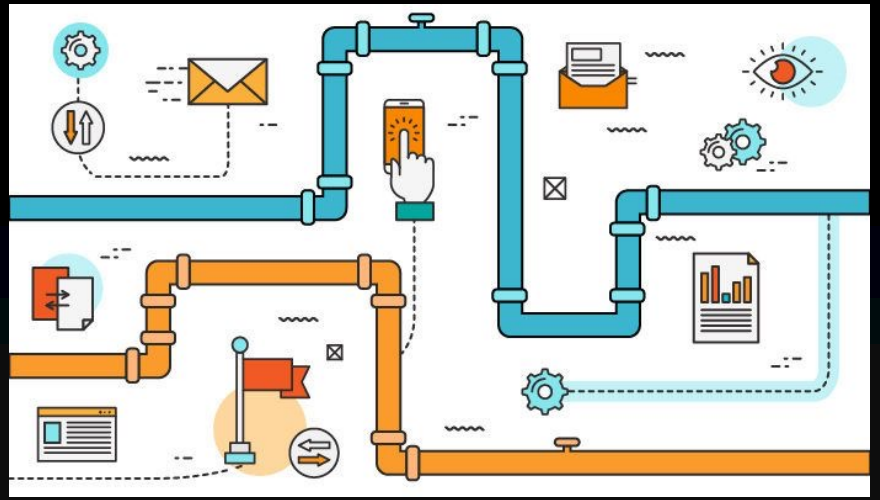
07 **Get Started**
With Apache Airflow

08 **Migrate**
To Apache Airflow

09 **Q and A**
You ask, I answer

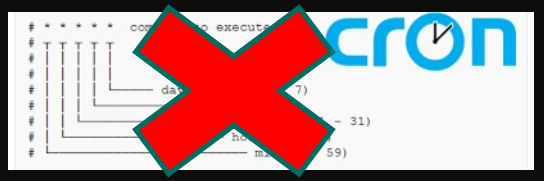
Data Orchestration

Data **orchestration** is the process of taking siloed data from multiple data storage locations, combining and organizing it, and making it available for data analysis tools. It can also include tasks like provisioning resources and monitoring.



*

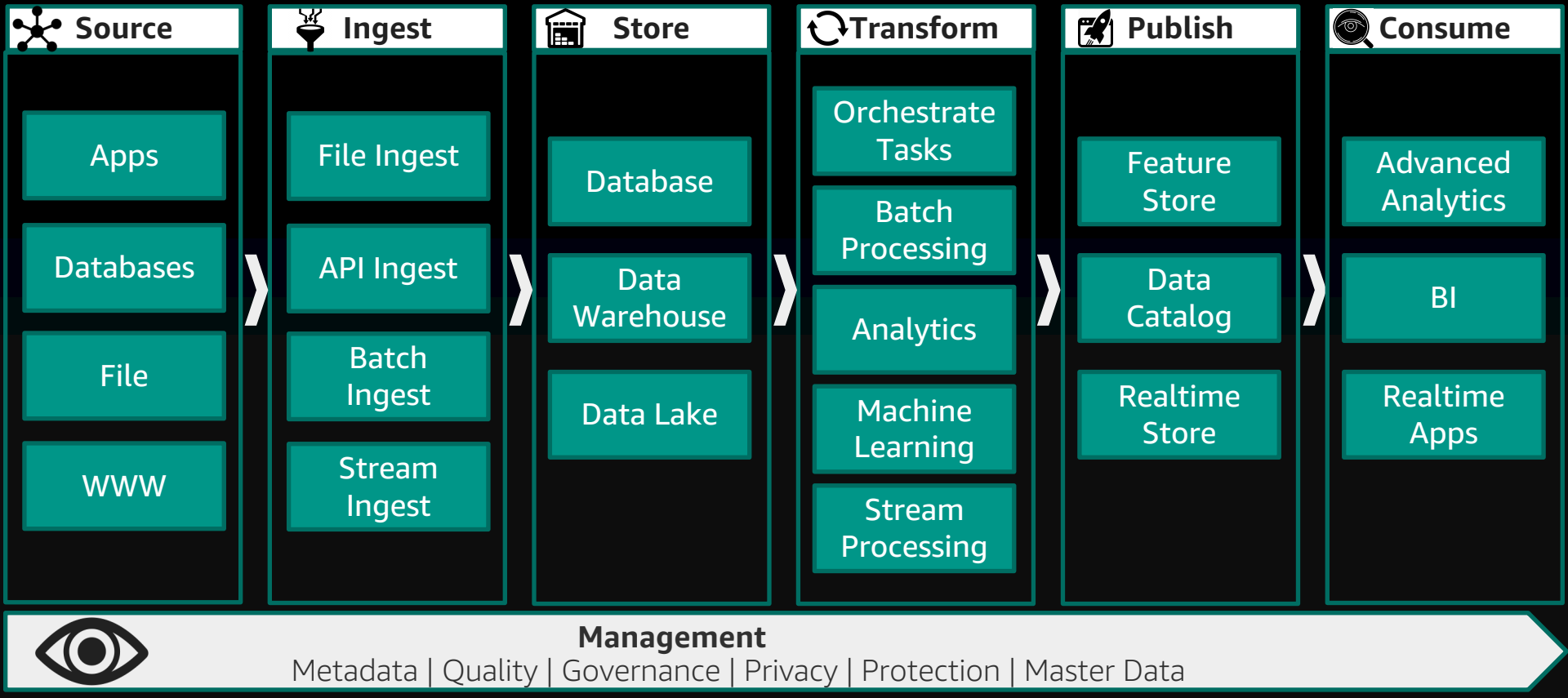
Data Pipelines, Workflows, Jobs



* Pic: https://www.bmc.com/blogs/workflow-orchestration/

Airflow Summit 2022

Data Pipeline



Benefits

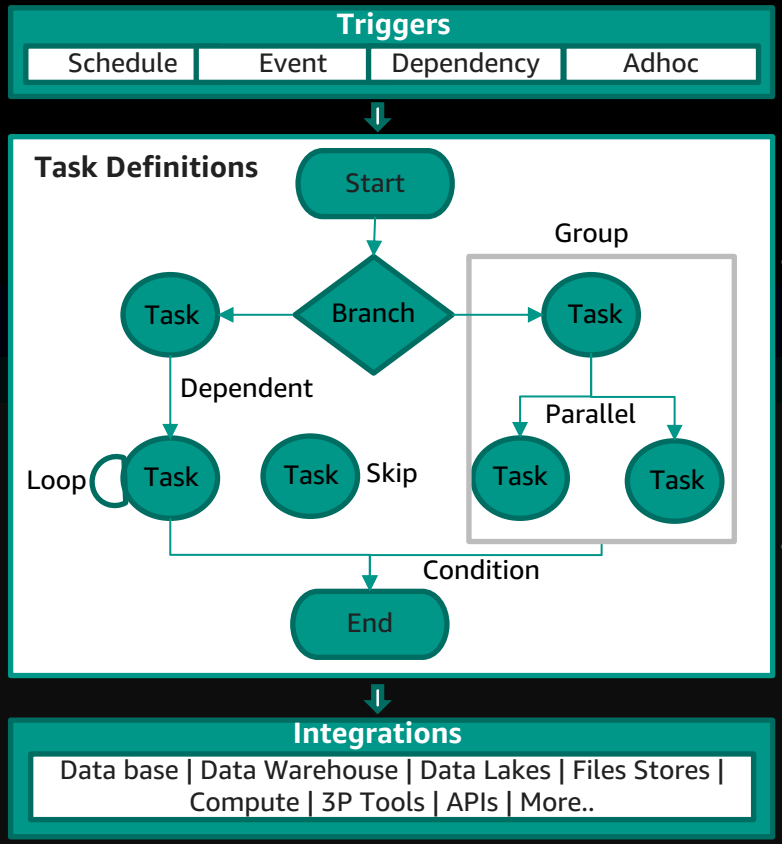
- Reduced human error
- Faster response to mission critical system problems
- More efficient allocation of resources

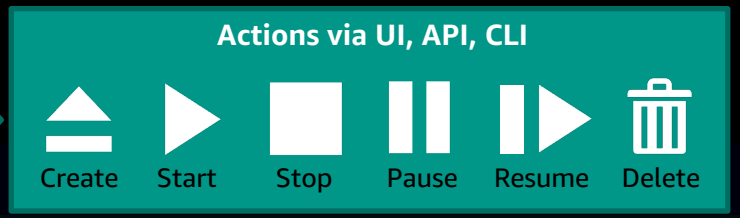
* As per EMA research

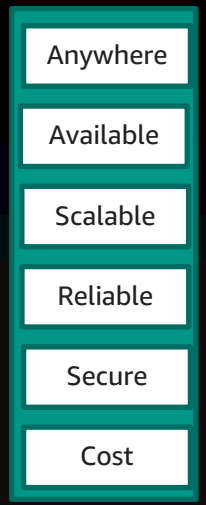
Persona based requirements

Business	Developer	Operations	Security
Availability	Programmable/ APIs	Admin	Compliance
Reliability	Testable – Local & Unit Test	Reporting/ Predictions	Access Control
Quality	Reusable/ Extensible	Alerting	At-Rest
Democratization	Integrations	Scalability	In-Transit
Simplicity	CI/CD	HA, DR	Audit
Cost Effective	Documentation/ Training	Infrastructure As Code / CM	Mitigation

The tool should







Available Orchestration tools

Airflow Summit 2022

Proprietary Tools

- Licensed from a vendor to install on a definite number of machines
- Source codes are not available publicly
- Customizations may come at an extra cost
- Specialized technical support, especially for enterprise clients
- Reliance on the vendor to continue to debug and improve the product

Red Hat
Ansible Automation
Platform

and more..

Available Orchestration tools

Open Source Software Tools

- Free to try, use, modify, redistribute
- Free community forums that offer support
- Open standards that increase transparency
- No vendor lock-in, No IP restrictions
- Easily scaled and extended

and more..

Lets Compare between Apache Airflow, Broadcom AutoSys, bmc Control-M and Spotify Luigi

History

Apache Airflow

- Started in October 2014 by Maxime Beauchemin at Airbnb.
- Open source from the very first commit and officially brought under the Airbnb GitHub and announced in June 2015
- Joined the Apache Software Foundation's Incubator program in March 2016

Broadcom AutoSys

- First developed by William Arntz and Walter Goodwin who created AutoSystems Ltd.
- Sold to Platinum Technology International in 1995
- Bought by Computer Associates in 1999
- Broadcom acquired CA in 2018 and known today as "AutoSys Workload Automation"

bmc Control-M

- Originally developed for scheduling jobs on mainframe computer systems, by an Israel-based company - New Dimension Software
- BMC software acquired New Dimension Software in **1999**
- "BMC Helix Control-M" - SaaS solution was released in December - **2020**

Spotify Luigi

- Created by Spotify mainly by Erik Bernhardsson and Elias Freider
- Initial commit on github/spotify/luigi on Nov 17, **2011**
- Open sourced in **2012**
- Spotify's Data Team maintains Luigi

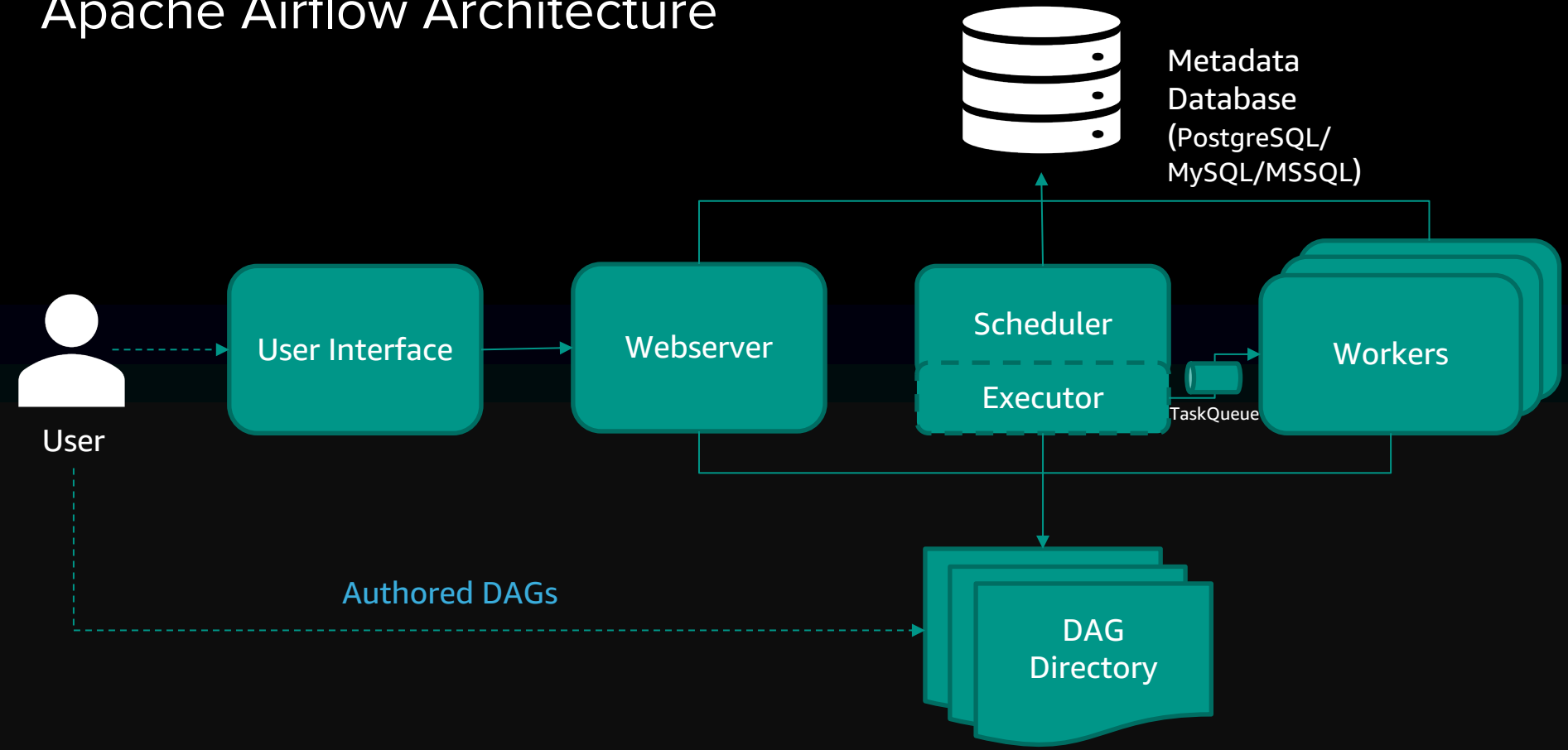
Similarities

OS Support	Availability	Security
Multi-platform Cloud & On-prem, Hybrid	HA, DR	SSO (LDAP/AD), RBAC, SSL

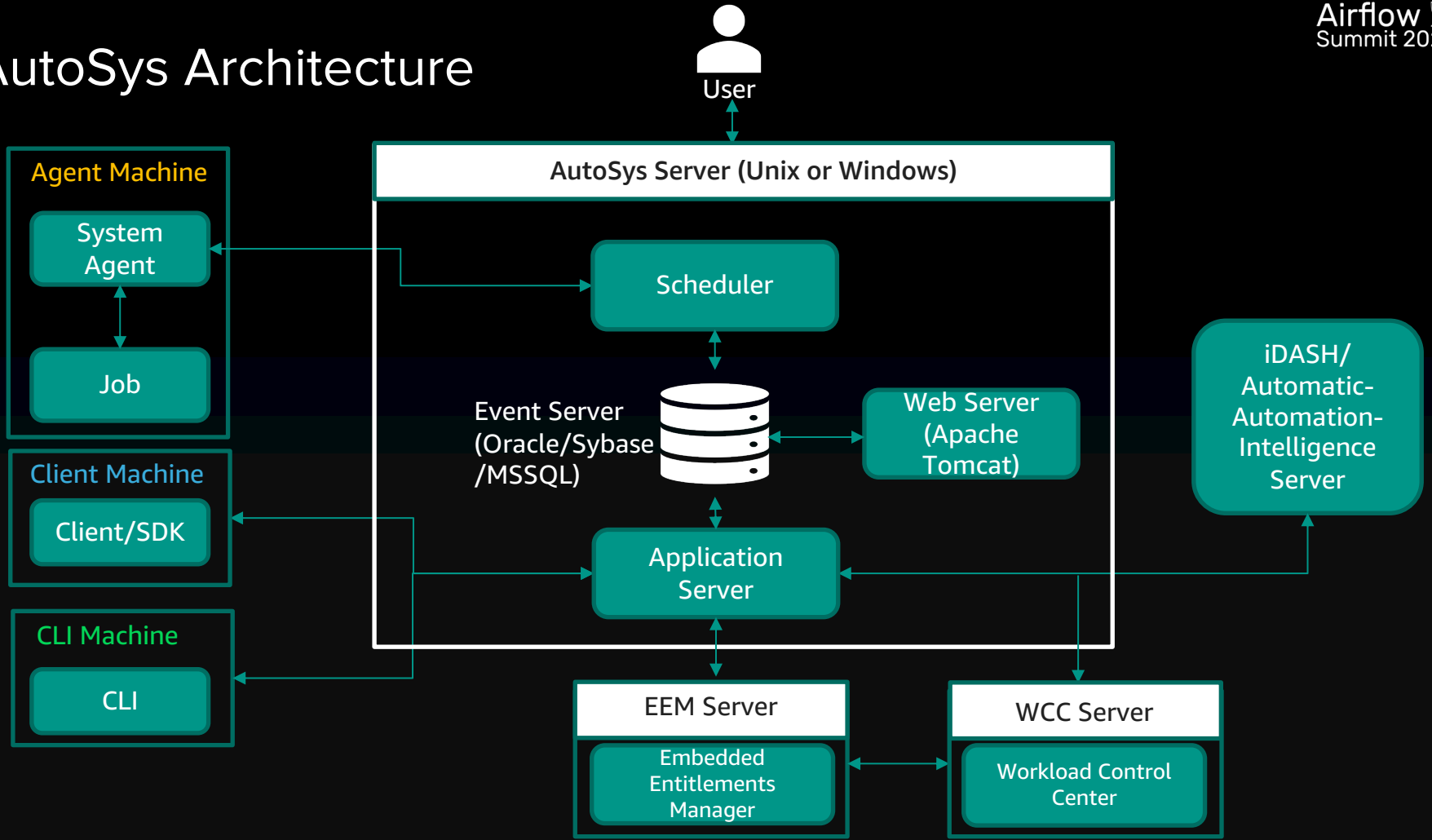
Interaction	Insights	Integration
Rich UI, REST API, CLI	History, Audit Trails, Dependency Graph	DBs, Data Warehouses, Data Lakes, Hadoop, 3Ps

Job Constructs	Scheduling	Jobs
Sequential, Parallel, Branching, Loops, Skips	Event, Schedule, Dependency, Manual	Groups/Nesting, Meta-data, Dynamic Jobs

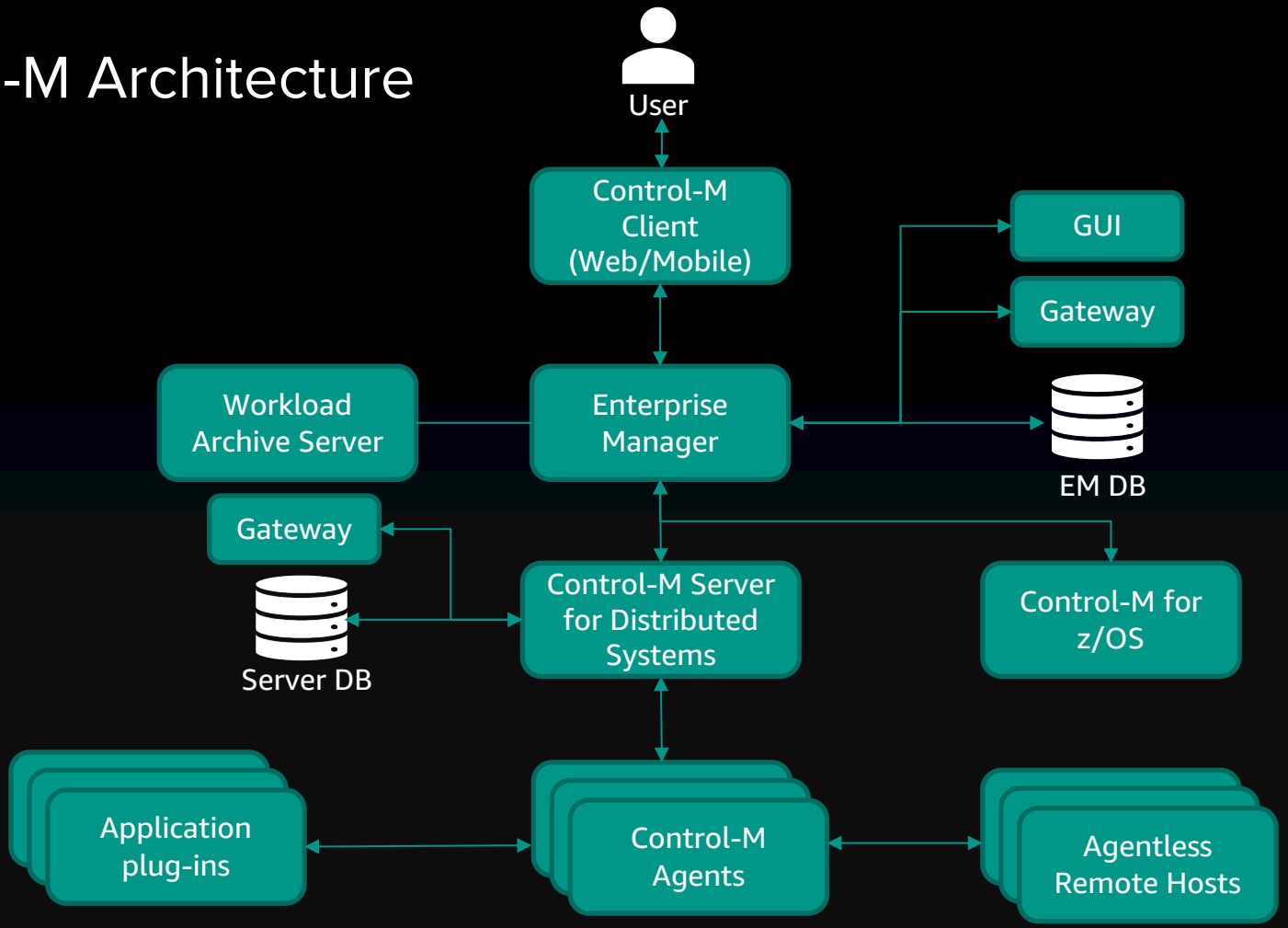
Apache Airflow Architecture



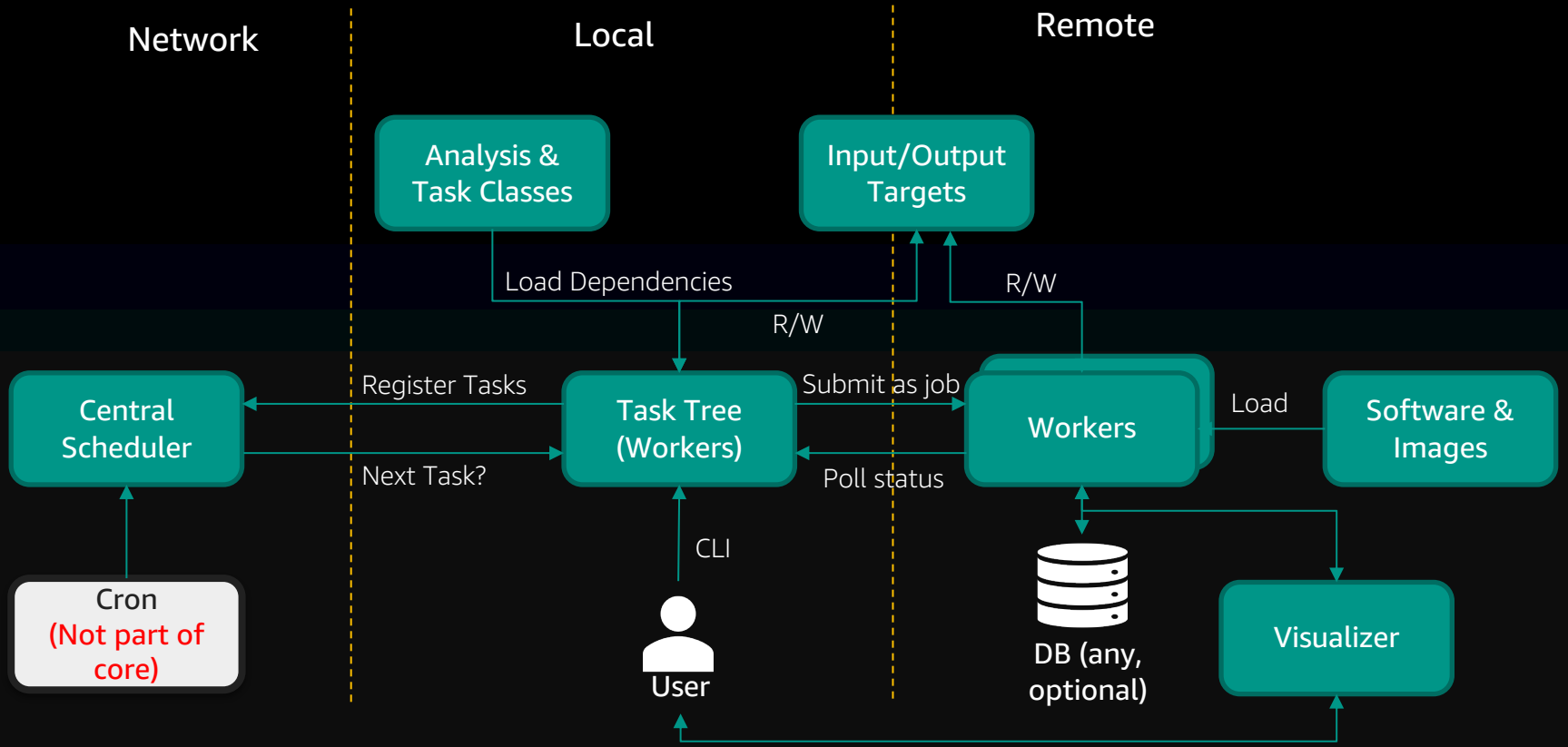
AutoSys Architecture



Control-M Architecture



Luigi Architecture



Install

Apache Airflow	Broadcom AutoSys
Agent-less. Use Connections, Hooks & Operators (Core and part of community provider packages)	Must install an agent (ex: Workload Automation Agent for UNIX, Linux, or Windows) and agent plug-ins (ex: Database Agent plug-in) on target system
bmc Control-M	**Spotify Luigi**
Agent based. Involves installing a Server, Agent and client installs along with Application Plug-ins and add-ons	Agent-Less. *pip install luigi* Create a physical connection in python to connect to DBs

Job Definition

Apache Airflow

Python

- **Default Args** = a constructor
- **DAG instantiate** = id, schedule
- **Tasks** = Operators
- **Dependencies** = upstream, downstream

Broadcom AutoSys

Job Information Language

- job_type
- command, machine
- owner
- date_conditions, days_of week, start_times
- alarm_if_fail

bmc Control-M

JSON/XML

- **Defaults** = define a parameter once
- **Job/When** = scheduling criteria
- **Job** = Tasks
- **Flow** = define order dependency

Spotify Luigi

Python

Luigi task = A class that contains methods:

- ***param*** = *parameters for the task*
- ***requires()*** = *specify task dependencies*
- ***run()*** = logic for execution
- ***output()*** = returns the artifacts generated
- ***optional input()*** = input from other tasks.

Apache Airflow Code Example

```python
from airflow.contrib.sensors.file_sensor import FileSensor
from airflow.operators.dummy_operator import DummyOperator

import datetime
from datetime import date, timedelta
import airflow

default_args = {
    "depends_on_past": False,
    "start_date": airflow.utils.dates.days_ago(1),
    "retries": 1,
    "retry_delay": datetime.timedelta(hours=5),
}
today = datetime.datetime.today()
yesterday = date.today() - timedelta(days=1)

with airflow.DAG("file_sensor_example", default_args=default_args,
                 schedule_interval= "*/5 * * * *") as dag:

    start_task = DummyOperator(task_id="start")
    stop_task = DummyOperator(task_id="stop")
    sensor_task = FileSensor(task_id="file_sensor_task",
                             poke_interval=30,
                             fs_conn_id= "<path>",
                            filepath= "<file or directory name>")

start_task >> sensor_task >> stop_task
```

Imported Libraries

Default Arguments/dictionary of default parameters

Unique identifier + schedule interval

Task definitions

Dependency

AutoSys JIL example

Export

autosys.jil

Airflow Summit > autosys.jil

```
1   insert_job: file-watcher-job
2   job_type: FW
3   machine: localhost
4   owner: me@aws
5   permission: mx
6   date_conditions: 1
7   days_of_week: all
8   start_times: “15:00, 14:00”
9   watch_file: <Some file path location>
10  watch_interval: 10
11  description: “Some description of the job”
12  std_out_file: /tmp/std_out
13  std_err_file: /tmp/std_err
14  alarm_if_fail: 1
15  profile: /tmp/.profile
```

Job type definitions

Client machine details

Schedule interval

Job specifics

Control-M Example

Drag & Drop via UI

```
{
    "Defaults" : {
        "Application" : "SampleApp",
        "SubApplication" : "SampleSubApp",
        "RunAs" : "USERNAME",
        "Host" : "HOST",
        "Job": {
            "When" : {
                "Months": ["JAN", "OCT", "DEC"],
                "MonthDays":["22","1","11"],
                "WeekDays":["MON","TUE", "WED", "THU", "FRI"],
                "FromTime":"0300",
                "ToTime":"2100"
            }
        }
    },
    "AutomationAPISampleFlow": {
        "Type": "Folder",
        "Comment" : "Code reviewed by John",
        "CommandJob": {
            "Type": "Job:Command",
            "Command": "echo my 1st job"
        },
        "ScriptJob": {
            "Type": "Job:Script",
            "FilePath":"SCRIPT_PATH",
            "FileName":"SCRIPT_NAME"
        },
        "Flow": {
            "Type": "Flow",
            "Sequence": ["CommandJob", "ScriptJob"]
        }
    }
```

Client machine details

Schedule interval

Job specifics

Dependency

Luigi Code Example

luigi.py

Airflow Summit > luigi.py > ...

```python
from datetime import date, timedelta
import luigi

class HelloWorld(luigi.Task):
    date = luigi.DateParameter(default=date.today() - timedelta(days=1))

    # no upstream requirements at all
    def requires(self):
        return None

    # creates a local file as output
    def output(self):
        return luigi.LocalTarget('helloworld.txt')

    # the actual job to perform
    def run(self):
        with self.output().open('w') as outfile:
            outfile.write('Hello World!\n')

if __name__ == '__main__':
    luigi.run()
```

Imported Libraries

Dependency

Job specifics

File System Events

Apache Airflow

FileSensor

- *FileSensor - Waits for a file or folder to land in a filesystem*
- *S3KeySensor* - AWS
- *WasbBlobSensor* - Azure
- *GCSObjectExistenceSensor* - GCP

Broadcom AutoSys

File Watcher/File Trigger Job

- The file reaches the minimum file size that is specified in the watch_file_min_size attribute.
- The file reaches a “steady state” during the polling interval

bmc Control-M

File Watcher (ctmfw)

- File transfer activity (min size)
- File Creation
- File Deletion

Spotify Luigi

None available natively

Have to be triggered externally based on file event

Autoscaling

Apache Airflow

Workers Autoscale

- *Parallelism*
- *Concurrency (max_active_tasks)*
- *min-workers*
- *max-workers*

Broadcom AutoSys

Agent based. Container scaling

- Agents can run in docker containers
- Run jobs on the container, autoscale tasks
- Auto register/un-register the Agent container from the server

bmc Control-M

Agent based. Container scaling

- Agents can run in docker containers
- Run jobs on the container, Autoscale tasks
- Auto register/un-register the Agent container from the server

Spotify Luigi

Workers Autoscale

Based on PENDING tasks count as determined by the Central Scheduler

Calendar

Apache Airflow

- Cron expression
- Time deltas - *Waits for a timedelta after the run's data interval (TimeDeltaSensor)*
- Timetables - *custom schedules using Python*

Broadcom AutoSys

- Standard Calendar - *Lists fixed dates*
- Cycle - *Lists date ranges (periods)*
- Extended Calendar - *Specifies complex criteria to generate a schedule based on logic*

bmc Control-M

- Regular Calendar - *specific dates, such as, days of the month, and days of the week in a selected year, holidays*
- Periodic Calendar - *different calendar periods other than months and days (quarter)*
- Rule Based Calendar - *specific complex rules. (3 days before the end of the month)*

Spotify Luigi

No native concept of scheduling

Luigi does not include its own triggering, so you have to rely on an external scheduler such as crontab to actually trigger the workflows.

Job Queue Priority

Apache Airflow

Pool + priority_weights parameter

- *default_pool = 128 slots*
- *priority_weights* = any arbitrary integer (default is 1). Higher values get higher priority in the executor queue.

Pools are meant to control parallelism for Task Instances

Broadcom AutoSys

priority attribute

priority_level = Defines the queue priority of the job. The lower the value, the higher the priority; 0 signifies to run the job immediately, regardless of the current machine load
Default: 0

sendevent -E CHANGE_PRIORITY -q queue_priority

bmc Control-M

PRIORITY* property

Available *priority levels* are Critical, High, Normal, Low, and Lowest. The default value for the PRIORITY* property is Normal.

pause a running job > Update priority > resume

Spotify Luigi

priority

Tasks with a higher priority value will be picked before tasks with a lower priority value. No predefined range of priorities, you can choose whatever (int or float) values. The default value is 0.

Alerting

Apache Airflow

Notifications

- *default_args/BaseOperator*
  'email_on_failure' :True
  'email': ['noreply@aws.com']
- *Custom Notifications (DAG or task level)*
  on_failure_callback, on_success_callback
- *SlackWebhookOperator*

Broadcom AutoSys

send_notification attribute

set the *send_notification* attribute value to **y** and specify the *notification_template* and the *notification_emailaddress*, *notification_emailaddress_on_success*, or *notification_emailaddress_on_terminated* attributes in your job definition.

bmc Control-M

Shout for job + Destination

- sent out before a job ends
- sent out after a job ends

Spotify Luigi

luigi.notifications module

[email] receiver=foo@bar.baz

send_email, send_error_email, send_email_smtp, send_email_ses, send_email_sns

SLA Management

Apache Airflow

SLA

- Define a callback method
- Pass the callback method to DAG
  sla_miss_callback
- Define the SLA duration on task(s)/DAG
  sla=timedelta(seconds=5)

monitor SLA miss in the Airflow UI

Broadcom AutoSys

CA Workload Automation iDash SLAs

- A separate web-based solution to install
- Generates alerts for SLA deadlines that are at risk of being missed, are predicted to be missed, or have been missed
- Executes automated recovery actions in response to alerts

bmc Control-M

Batch Impact Manager

- Separate utility that needs to be installed
- Alerts on potential delays
- Can set deadline to finish, should be completed by since start, notification action
- Analyze why by Filtering on Critical Path using Analysis Viewpoint option

Spotify Luigi

No out-of-the-box solution

Have to be custom-built using Python

Forecast Reports

Apache Airflow

Insights only for past runs

- Calendar View - see trends of the overall success/failure rate of runs over time.
- Gantt Chart - analyse task duration and overlap
- Task Duration - duration of your different tasks over the past N runs

Broadcom AutoSys

Forecast Reports

- Displays information about predicted workflow
- Forecast reports help you identify problems with the predicted workflow to resolve them before they occur or to plan changes in the workflow.

bmc Control-M

Forecast

- Add-on component
- A visual calendar that displays all the dates on which the job will be scheduled.
- Estimated time execution window for each and every job
- Trend analysis displayed as a histogram

Spotify Luigi

No out-of-the-box solution

Have to be custom-built using Python

- Luigi Visualizer shows basic info (Status, Priority, Time)
- Luigi Task status shows dependency graph

Job Source Versioning

Apache Airflow

No Built-in Support

- Have to use external SCM tools like Github, BitBucket, AWS Code Commit etc. for DAG code versioning
- Have to use other CD tools like Jenkins, CircleCI, Github Actions etc. for deployment to *{AIRFLOW_HOME}/dags*

Broadcom AutoSys

No Built-in Support

- Have to use external SCM tools like Github, BitBucket, AWS Code Commit etc. for JIL definition versioning
- Have to use other CD tools like Jenkins, CircleCI, Github Actions etc. for deployment using CLI commands

bmc Control-M

Built in Check-in/Check-out

- Display changes between Job versions
- Restore a previous version of a job
- Restore a deleted job
- Audit Report – User that changed the job

Workload Change Manager

Spotify Luigi

No Built-in Support

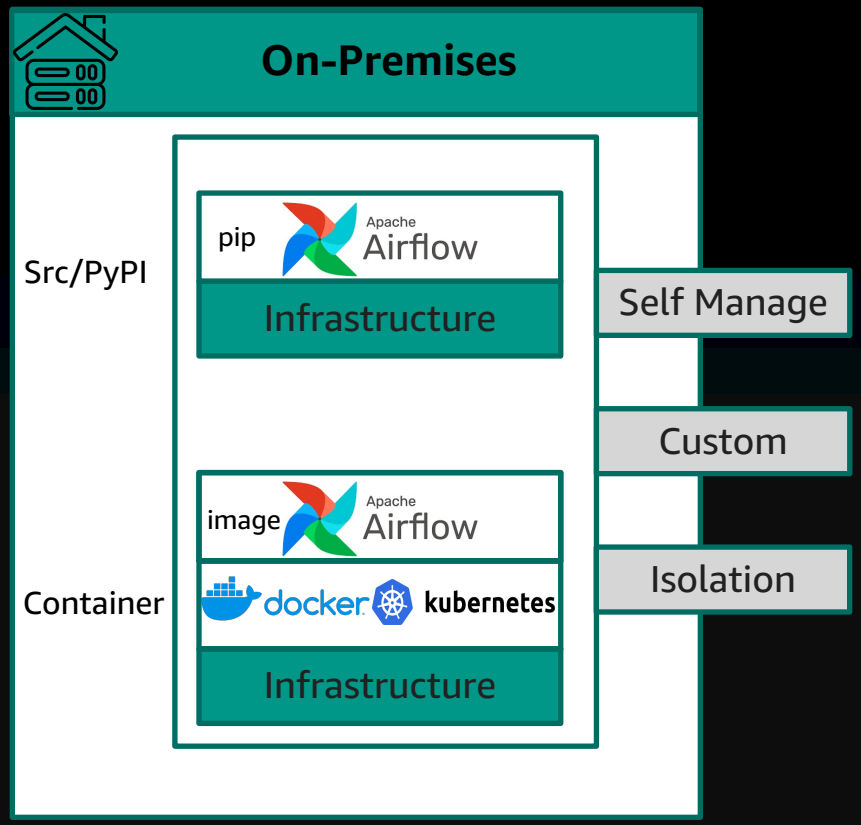
- Have to use external SCM tools like Github, BitBucket, AWS Code Commit etc. for Python code versioning
- Have to use other CD tools like Jenkins, CircleCI, Github Actions etc. for deployment using commands

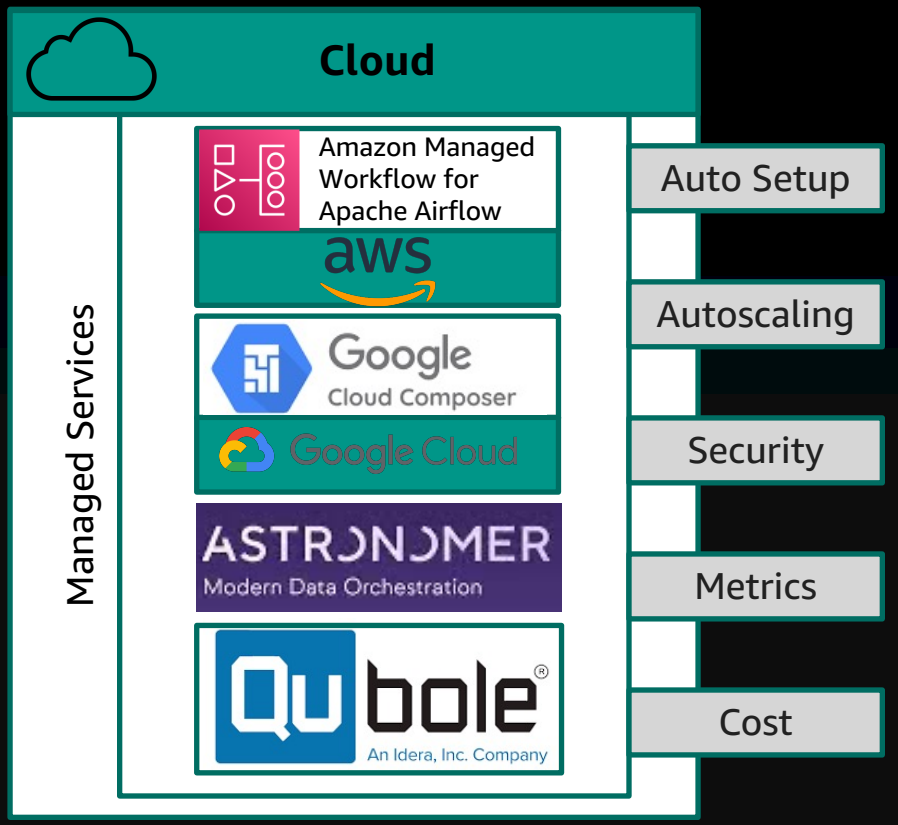
Features Unique to Apache Airflow

Airflow Summit 2022

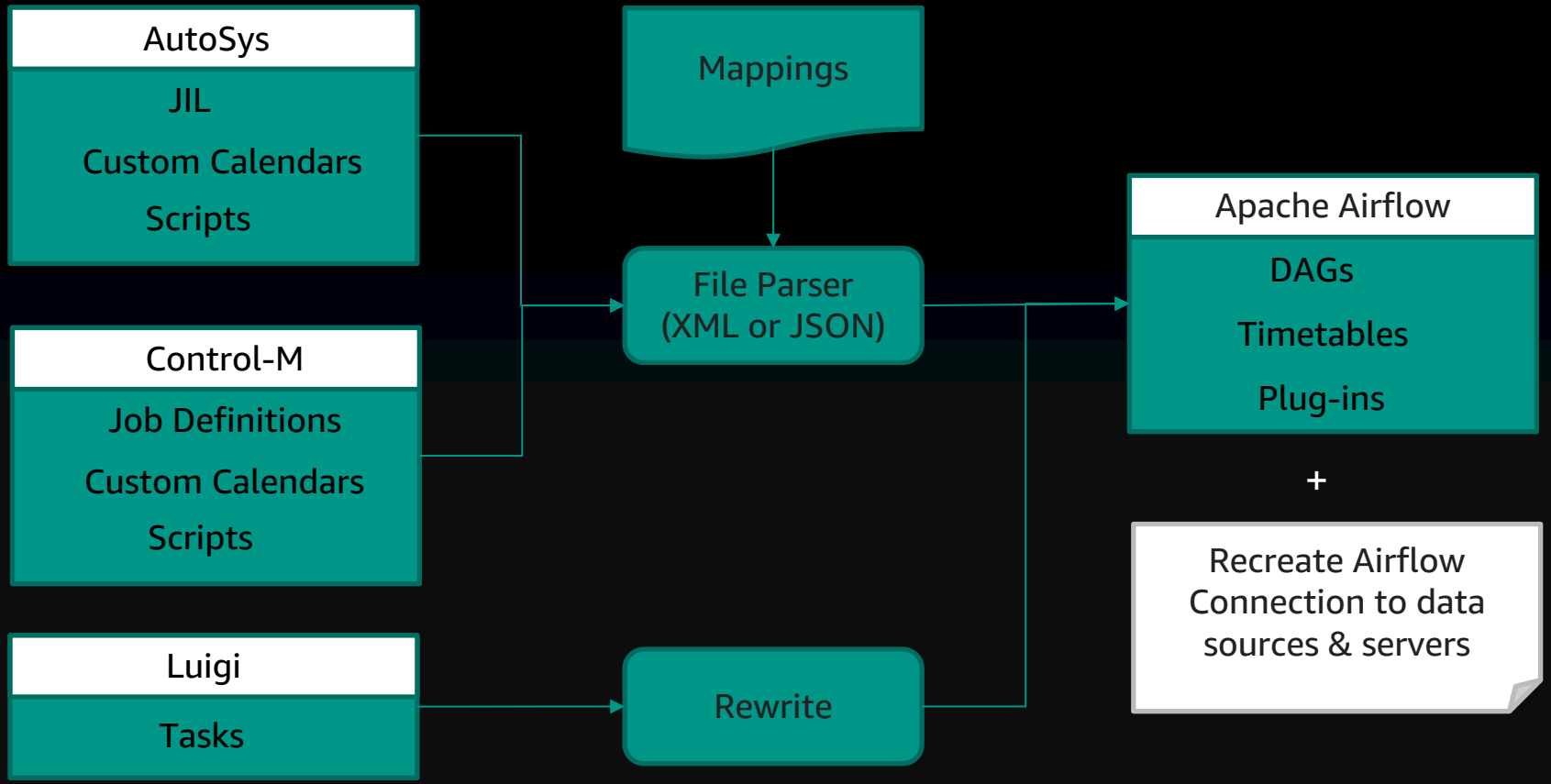
- Extensible
- Run Locally
- Catchup & Backfill
- Swappable Executors
- Deferrable Operators & Triggers
- Unit Tests
- Custom Secrets Backend
- Data Lineage
- Templating

How to get started





How to Migrate



Airflow Summit 2022

In Summary

- Use a data workflow orchestration tool to easily Build, Define, Schedule, Manage, and Monitor production workflows, ensuring visibility, reliability, and improving SLAs
- Use Apache Airflow to:
    - Programmatically author, schedule and monitor workflows
    - It is Easy to Use, Scalable, Dynamic, Extensible & Elegant
    - Open Source, so no punitive licensing fees and you can Customize it
    - Provides Robust Integrations to current infrastructure and Extend to next-gen technologies
    - Comparable features (and more) than proprietary tools
- Use a Managed service to focus on authoring, scheduling, and monitoring your workflows as opposed to provisioning resources.
- Refactor/re-architect from existing proprietary tools to improve operating cost, agility, performance, and scalability

Questions for Me?

Post them in the Chat
Feel free to voice them as well

- And If you don't remember them now:
- https://www.linkedin.com/in/parnab-basak/

Thank You!

Enjoy the other summit talks as well.