



Aalto University
School of Science

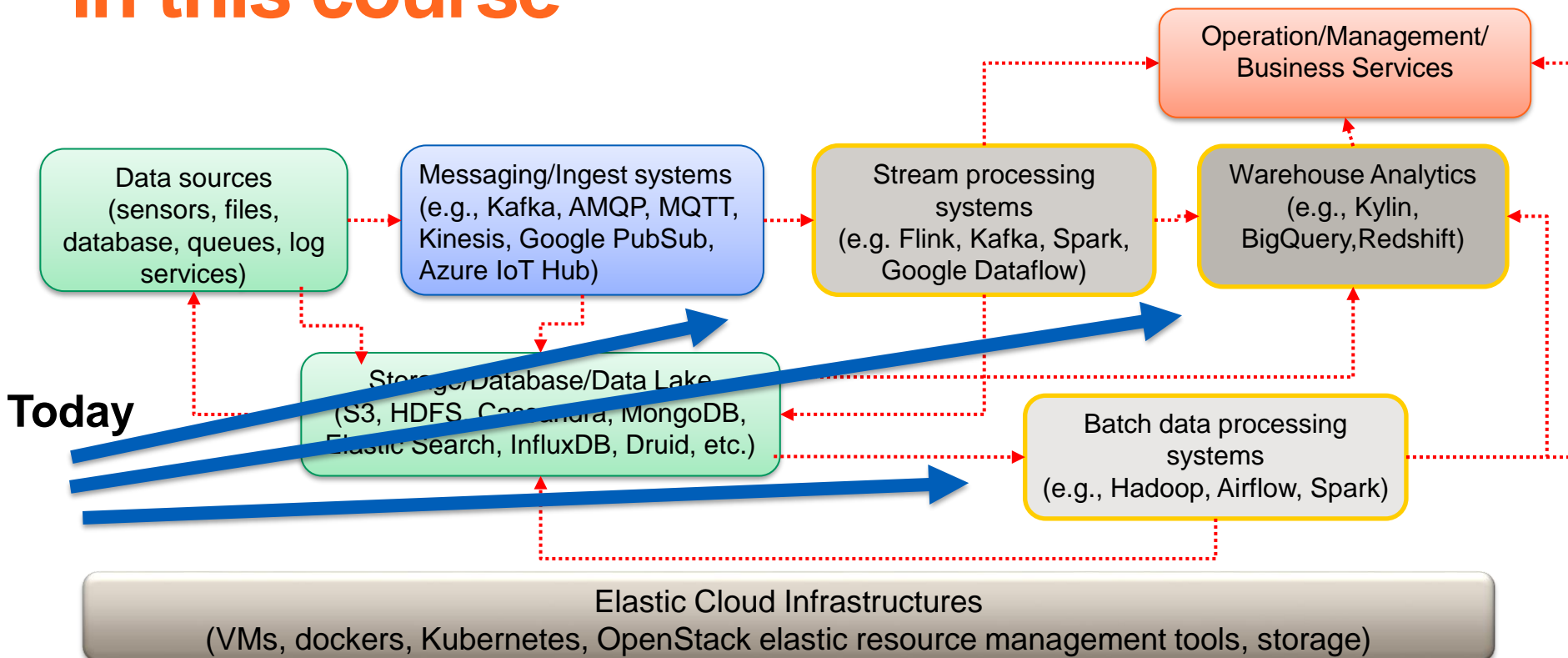
Workflows for Big Data Platforms

Hong-Linh Truong
Department of Computer Science
linh.truong@aalto.fi, <https://rdsea.github.io>

Learning objectives

- **Understand the role and use cases of workflows in big data platforms**
- **Understand key concepts and techniques in workflows and able to design workflows**
- **Able to apply common workflow technologies for practical work**

Big data at large-scale: the big picture in this course



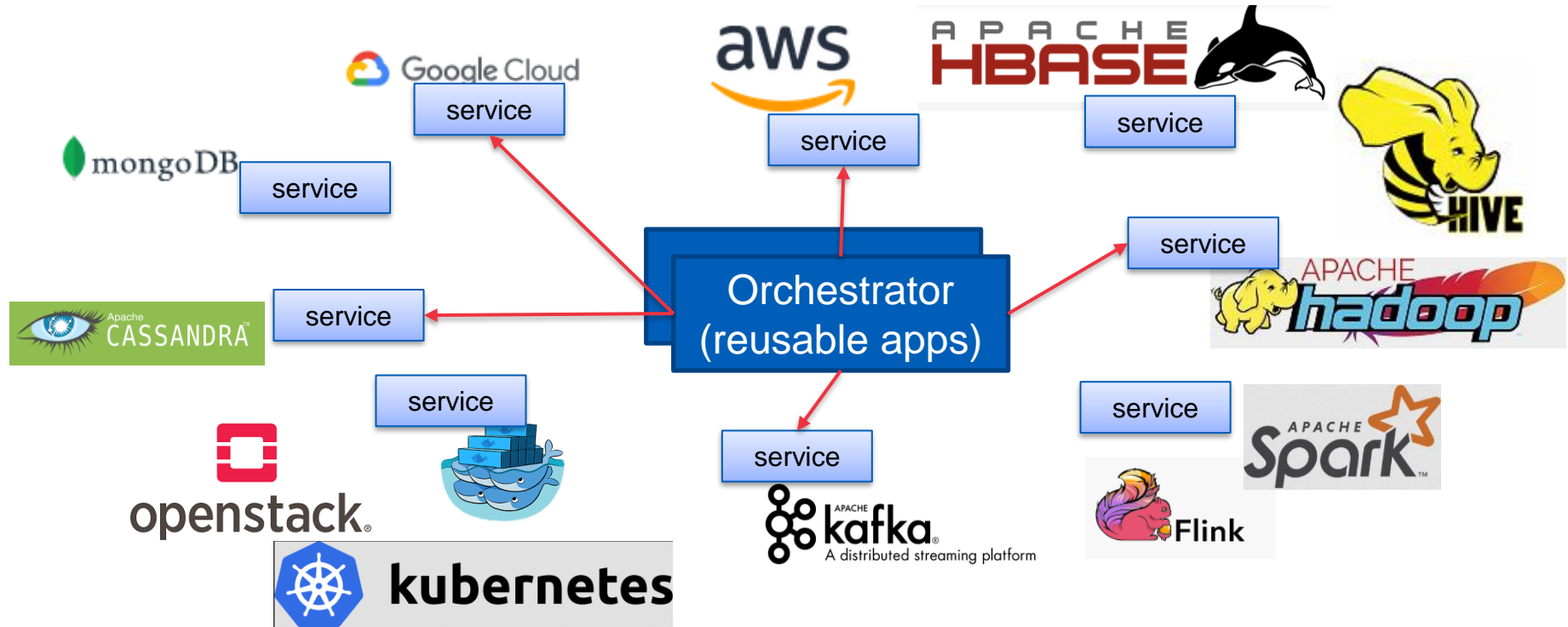
Tasks in big data platforms

- **Data collection and transformation**
 - data transfers, extraction, transformation,
- **Data processing, including machine learning**
 - data analytics, training, serving machine learning algorithms
- **Automation in big platform infrastructures**
 - service deployment, resource elasticity, backup/recovery
- **Business service integration with big data platforms**
 - integration with customer services, bringing insights from data analytics to business decision making

Many complex use cases

- **ETL, data cleansing and backup**
 - access and coordinate many different compute services, data sources, ingestion and extraction applications
- **Complex predictive maintenance**
 - coordination of machine learning pipelines and communication with humans/optimization services
- **Analytics-as a service**
 - metrics understanding, user activities analytics, customer understanding

Service Orchestration



Example of security data analytics

Security-related information and metrics from distributed customers

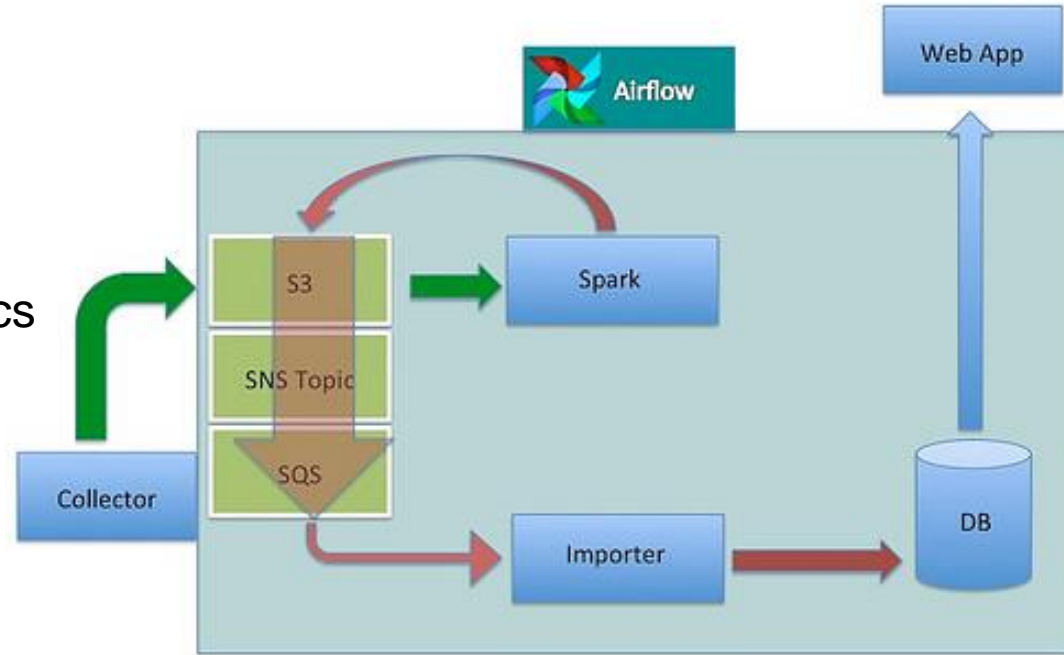


Figure Source: <http://highscalability.com/blog/2015/9/3/how-agari-uses-airbnbs-airflow-as-a-smarter-cron.html>

Example of industrial retail forecast

| date | id | name | volume | price | cost | promo | category_net | margin | category1 | category2 | location | sales |
|------------|-----|---------|---------|-------|------|-------|--------------|--------|-----------|-----------|----------|-----------|
| 07/01/2018 | 100 | Chicken | 38144.0 | 3.79 | 2.7 | 0 | 451692.0 | 0.25 | Meat | Food | Helsinki | 144565.76 |
| 14/01/2018 | 100 | Chicken | 36420.0 | 3.79 | 2.66 | 0 | 414342.0 | 0.25 | Meat | Food | Helsinki | 138031.8 |
| 21/01/2018 | 100 | Chicken | 35322.0 | 3.79 | 2.66 | 0 | 381854.0 | 0.25 | Meat | Food | Helsinki | 133870.38 |

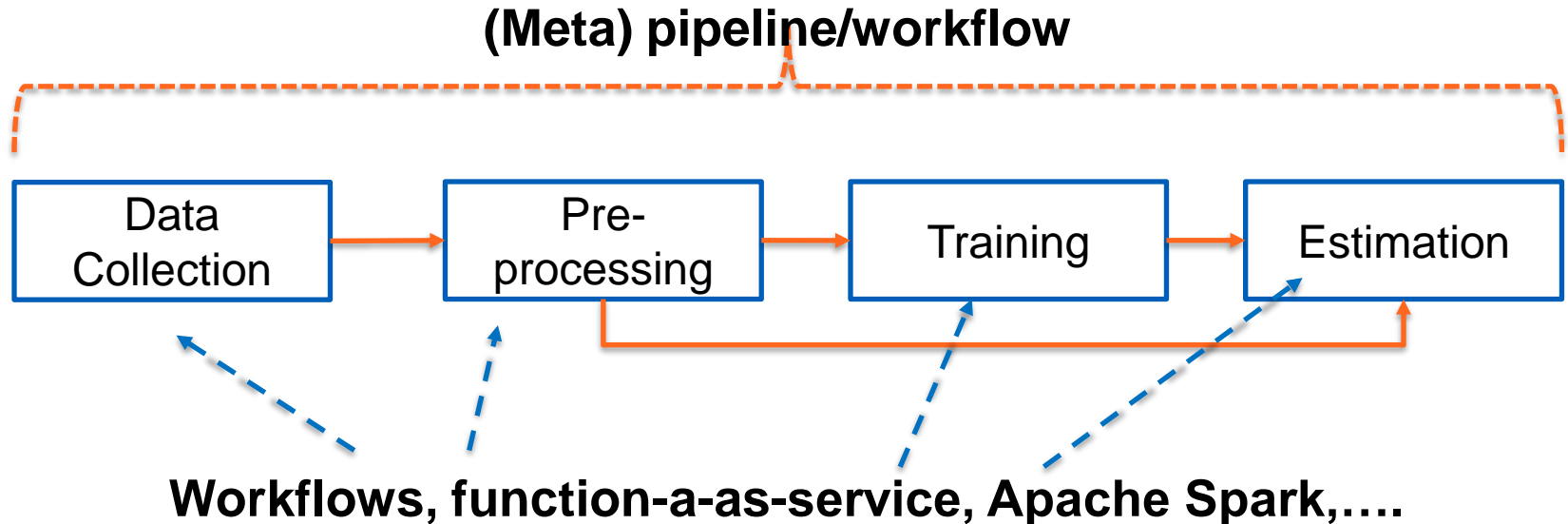
Sellforte: forecast where to put marketing information



Source: Kreics Kristis, „Quality of analytics management of data pipelines for retail forecasting“, Aalto CS Master thesis, 2019

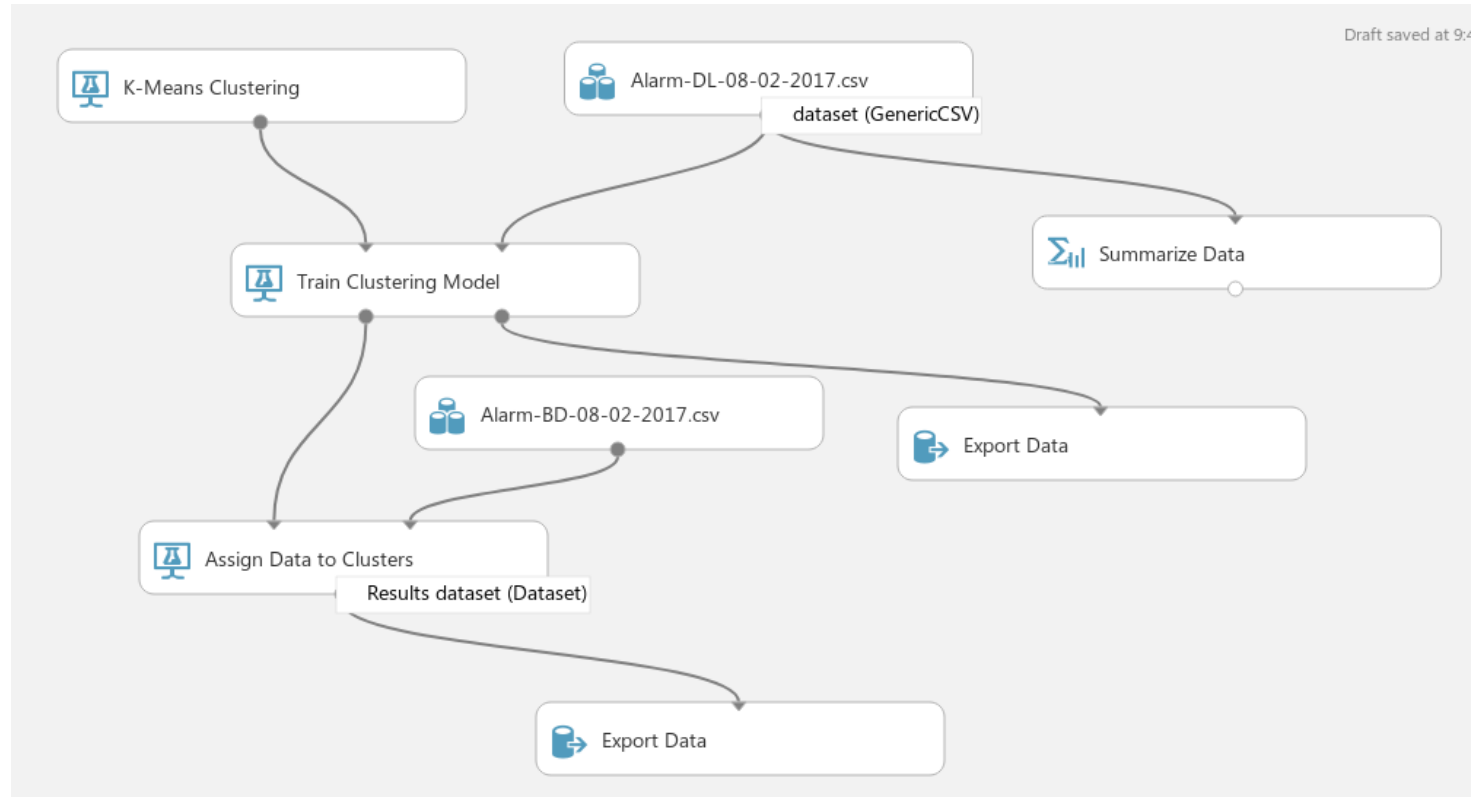
Example of ML workflows

- meta-workflow vs inside each phase: pipeline/workflow or other types of programs



Example of ML workflows

Azure ML



Workflows

A workflow specifies a process

- consists of a set of connected steps/tasks
- has steps/tasks carried out by diverse types of software services or humans, each performs a function
- can be automated with/without human intervention
- have data/control task dependencies
- can be reusable

Workflow technologies

- **Given many services offering different capabilities, we can combine them for different cases**
 - orchestration of capabilities from different services as the key!
 - reuse/customization of capabilities with a given set of services
- **Workflows are flexibly defined and changed**
 - services cannot be changed easily
 - but there are many ways to combine such services!
 - the integration is loosely coupled.

We have many workflows that are built in a flexible way for different goals

How to build the workflows and orchestrate tasks in these workflows?

Tasks and workflows

- **Diverse types of tasks**
 - task can be simple or complex (e.g., a task running an AI algorithm)
 - tasks are performed by software and humans
- **Workflow**
 - coordinate/orchestrate many tasks, *the function of tasks is not really “carried out” by workflows* → *orchestration/coordination*
 - workflow can be simple, like a pipeline of a sequence of tasks or complex with many forks/loops

Workflow and pipeline/data workflow

- **Data workflows → data pipeline**

” a pipeline is a set of data processing elements connected in series, where the output of one element is the input of the next one”

Source: https://en.wikipedia.org/wiki/Pipeline_%28computing%29

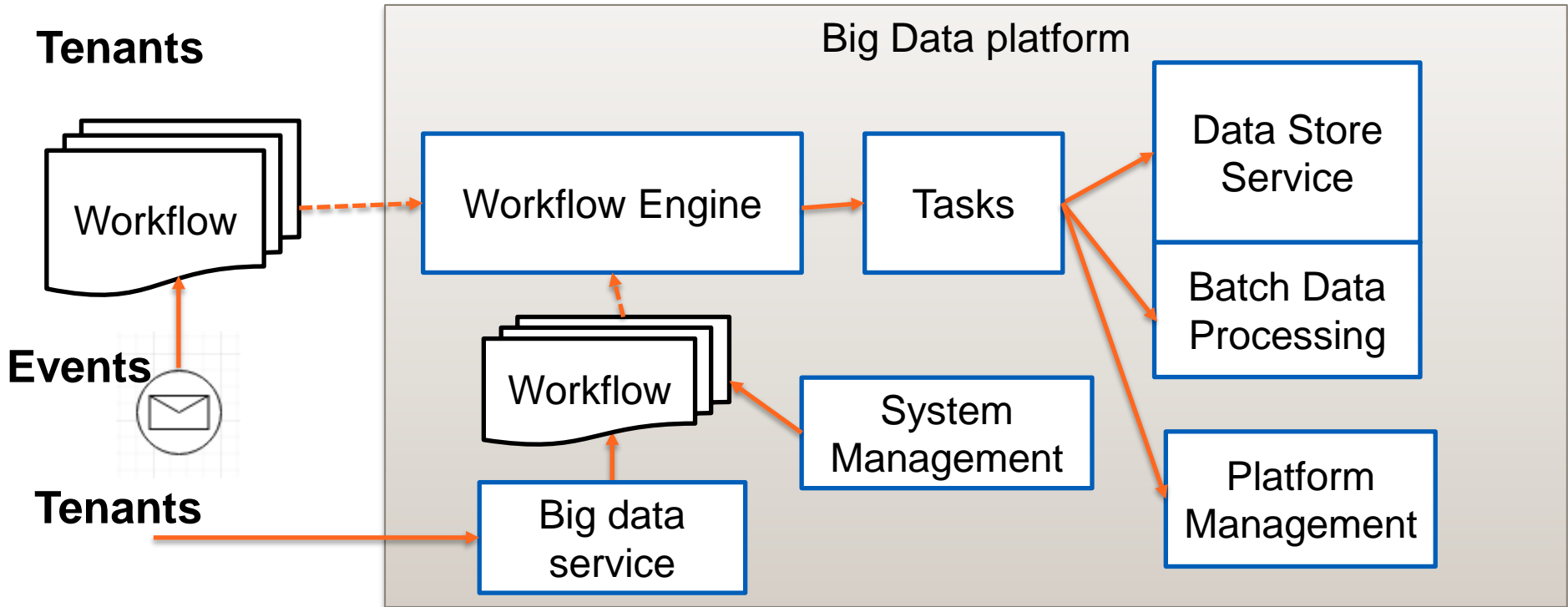
- **Two interpretations:**

- a pipeline is a simple workflow
- a pipeline coordinates different (sub)workflows

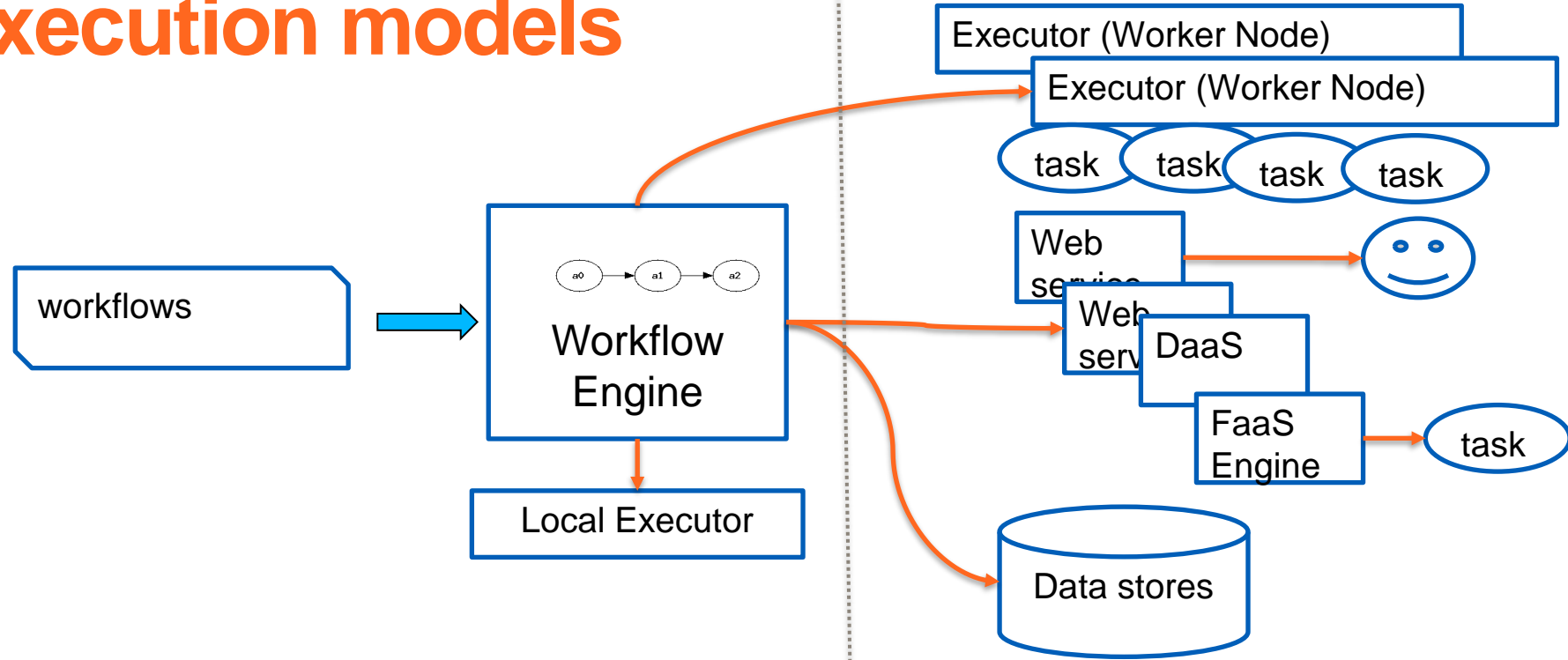
A long history – workflows are well-known!

- **Business workflows/processes**
 - business processes in enterprise computing (e.g., BI, ERP, and e-commerce)
- **Scientific workflows**
 - in scientific computing and high performance computing (e.g., bioinformatics, astrophysics, material science simulations)
- **Automation in system management**
 - at system level for automating infrastructure provisioning, system recovery, etc.

Workflows in big data platforms: more than analytics



Common workflow execution models



Key components

- **Tasks/Activities**

- describe a single work (it does not mean small)
- tasks can be carried out by humans, executables, scripts, batch applications, stream applications and other types of services.

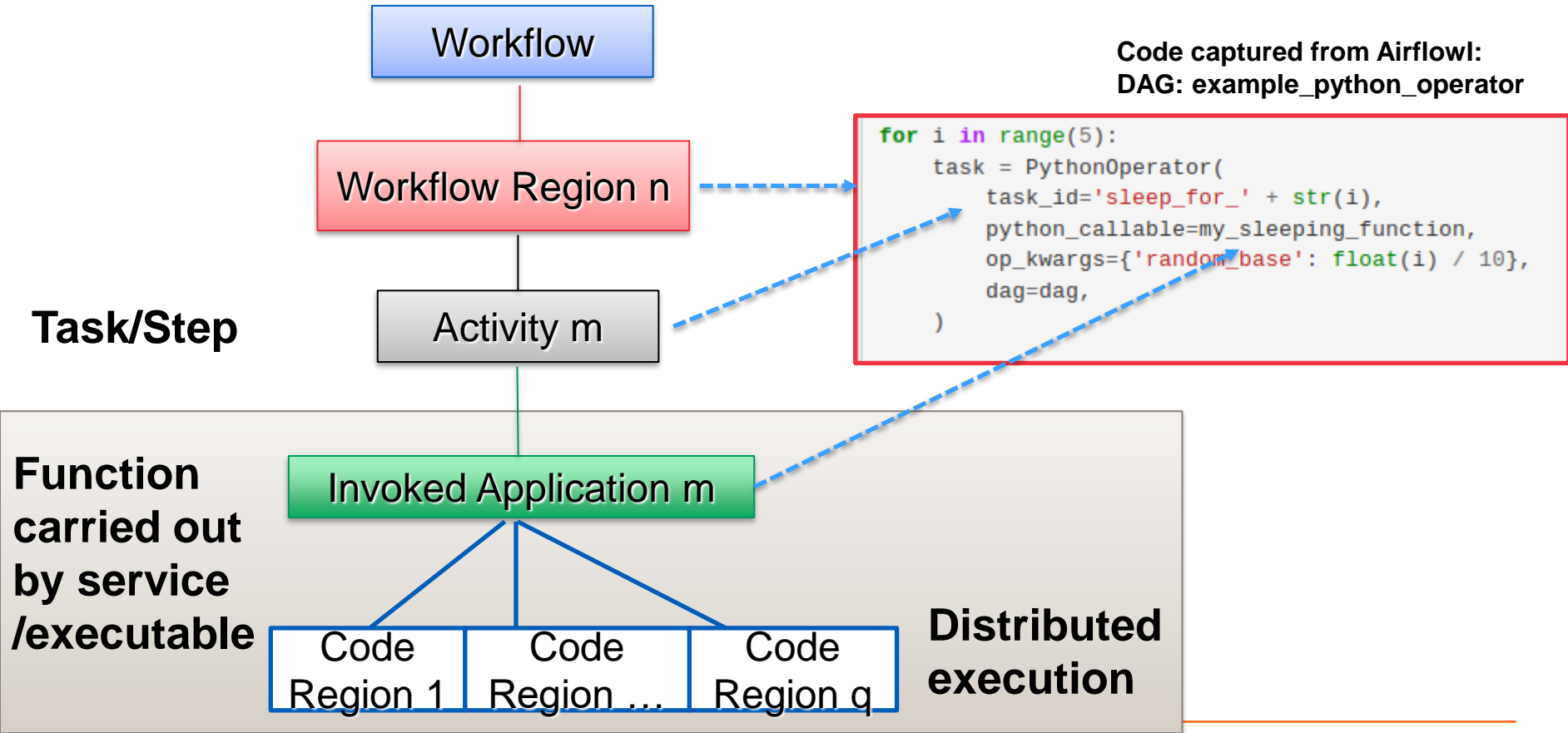
- **Workflow Languages**

- structure/describe tasks, dataflows, and control flows

- **Workflow Engines**

- execute the workflow by orchestrating tasks
- usually *call remote services* to run tasks

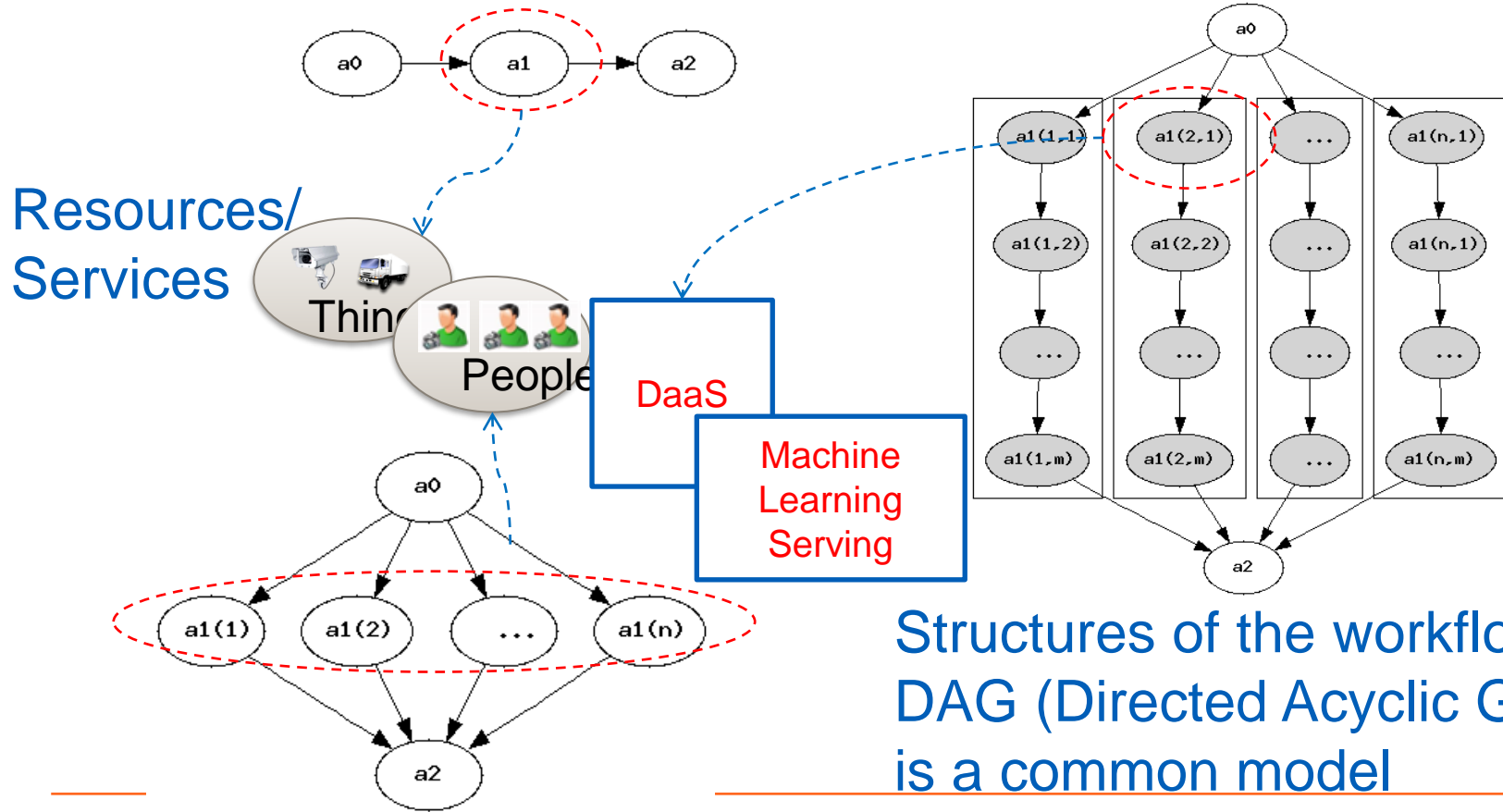
Structured view of workflows



Describing workflows

- **Programming languages with procedural code**
 - general- and specific-purpose programming languages, such as Java, Python, Swift
 - common ways in big data platforms for data analytics and system automation
- **Descriptive languages with declarative schemas**
 - BPEL and several languages designed for specific workflow engines
 - common in business and scientific workflows

Tasks orchestration



Runtime aspects

- **Parallel and distributed execution**
 - tasks are executed in different machines (by external invoked applications/services), **multiple running workflows**
- **Long running**
 - can be hours or weeks! → **pausing and resuming workflows are normal**
- **Checkpoint and recovery**
 - dealing with failures at different levels: **workflows and tasks retry/recovery**
- **Monitoring and tracking**
 - States and performance metrics: **queuing, running, idle, suspended, failed**
- **Stateful management**
 - dependencies among tasks w.r.t control and data, stateful tasks → **global services for managing states and data among tasks**

Select/build workflows in your platforms

- **Rich data services**
 - for data storing/retrieving tasks
- **Big data computation engines**
 - for data processing tasks with different workload: ML and (batch/stream) big data processing
- **Different underlying cloud/distributed computing infrastructures**
 - for resource management tasks and workflow infrastructures
- **REST APIs and message systems integration**
 - for widely integration with other services (e.g., business services)

Select/build workflows in your platforms

- **Scheduling**

- Scheduling in a large resource pool (e.g., using clusters)

- **Elasticity**

- Elasticity controls of virtualized resources (VMs/containers/Kubernetes) for executing tasks

- **Multiple levels of parallelism**

- Cluster level vs node level

- **Examples**

- Periodic cron schedules, backfill, opportunistic schedules
- Increase number of distributed workers/cluster sizes
- Heterogenous resources for tasks: lightweight compute nodes & high-end nodes

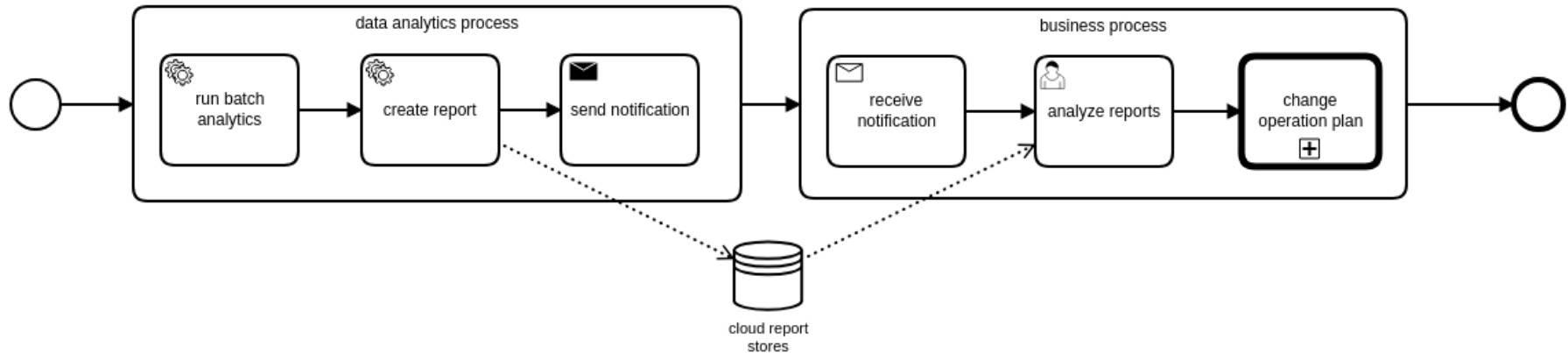
Wu, F., Wu, Q. & Tan, Y. Workflow scheduling in cloud: a survey. J Supercomput 71, 3373–3418 (2015). <https://doi.org/10.1007/s11227-015-1438-4>

Mainak Adhikari, Tarachand Amgoth, and Satish Narayana Srirama. 2019. A Survey on Scheduling Strategies for Workflows in Cloud Environment and Emerging Trends. ACM Comput. Surv. 52, 4, Article 68 (August 2019), 36 pages. <https://doi.org/10.1145/3325097>

Select/build workflows in your platforms

- **Integration**

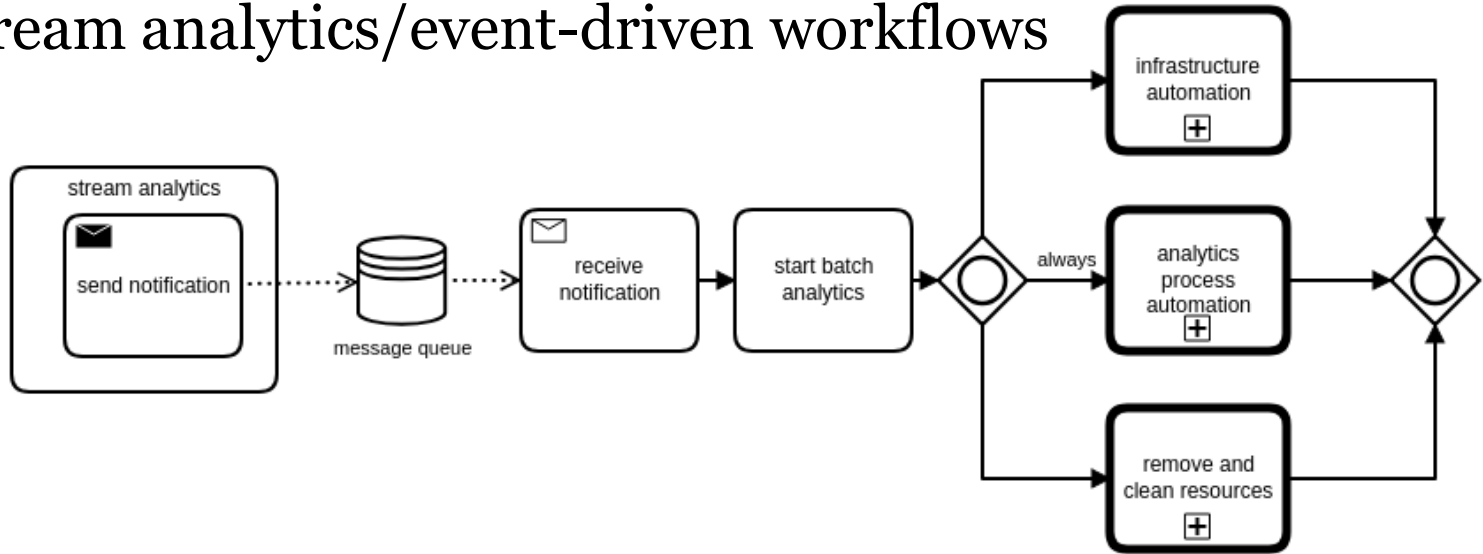
- Data analytics processes and business processes
- Include human-in-the-loop



Select/build workflows in your platforms

■ Integration

- Multiple types of workflows for services/infrastructure provisioning and analytics
- Stream analytics/event-driven workflows



Existing frameworks for your study

- **Apache Oozie**
 - designed to work with Hadoop: orchestrating Hadoop jobs
- **Serverless-based: Function-as-a-Service**
 - e.g., Microsoft, Google, AWS serverless/function-as-a-service
- **Apache Airflow**
 - a generic workflow framework
- **Argo Workflows**
 - Container-native workflow engine
- **Uber Cadence (<https://cadenceworkflow.io/>)/Camunda (<https://camunda.com/>)**
 - Connecting to business activities+ human in the loop

Example with Apache Airflow

<https://airflow.apache.org>

Airflow overview

- **Originally from Airbnb**
- **Features**
 - Dynamic, **extensible**, scalable workflows
 - Programmable language-based workflows
 - *Write workflows as procedural code*
- **Good and easy to study to understand concepts of workflows/data pipeline**
- **Google Cloud Composer is a cloud-provided version of Airflow**
 - <https://cloud.google.com/composer/>

Many connectors

| | | |
|-------------------|--|-----------------------|
| async | <code>pip install 'apache-airflow[async]'</code> | Async worker classes |
| celery | <code>pip install 'apache-airflow[celery]'</code> | CeleryExecutor |
| cloudant | <code>pip install 'apache-airflow[cloudant]'</code> | Cloudant hook |
| crypto | <code>pip install 'apache-airflow[crypto]'</code> | Encrypt connection p |
| devel | <code>pip install 'apache-airflow[devel]'</code> | Minimum dev tools re |
| devel_hadoop | <code>pip install 'apache-airflow[devel_hadoop]'</code> | Airflow + dependenci |
| druid | <code>pip install 'apache-airflow[druid]'</code> | Druid related operatc |
| gcp | <code>pip install 'apache-airflow[gcp]'</code> | Google Cloud Platfor |
| github_enterprise | <code>pip install 'apache-airflow[github_enterprise]'</code> | GitHub Enterprise au |
| google_auth | <code>pip install 'apache-airflow[google_auth]'</code> | Google auth backend |
| hdfs | <code>pip install 'apache-airflow[hdfs]'</code> | HDFS hooks and ope |
| hive | <code>pip install 'apache-airflow[hive]'</code> | All Hive related oper |
| jdbc | <code>pip install 'apache-airflow[jdbc]'</code> | JDBC hooks and ope |
| kerberos | <code>pip install 'apache-airflow[kerberos]'</code> | Kerberos integration |

| | | |
|------------|---|---|
| kubernetes | <code>pip install 'apache-airflow[kubernetes]'</code> | Kubernetes Executor |
| ldap | <code>pip install 'apache-airflow[ldap]'</code> | LDAP authentication |
| mssql | <code>pip install 'apache-airflow[mssql]'</code> | Microsoft SQL Server |
| mysql | <code>pip install 'apache-airflow[mysql]'</code> | MySQL operators and |
| oracle | <code>pip install 'apache-airflow[oracle]'</code> | Oracle hooks and ope |
| password | <code>pip install 'apache-airflow[password]'</code> | Password authentica |
| postgres | <code>pip install 'apache-airflow[postgres]'</code> | PostgreSQL operator |
| qds | <code>pip install 'apache-airflow[qds]'</code> | Enable QDS (Qubole |
| rabbitmq | <code>pip install 'apache-airflow[rabbitmq]'</code> | RabbitMQ support as |
| redis | <code>pip install 'apache-airflow[redis]'</code> | Redis hooks and sens |
| s3 | <code>pip install 'apache-airflow[s3]'</code> | <code>S3KeySensor</code> , <code>S3Prefi</code> |
| samba | <code>pip install apache-airflow[samba]'</code> | <code>airflow.operators.hiv</code> |
| slack | <code>pip install 'apache-airflow[slack]'</code> | <code>airflow.operators.sla</code> |
| ssh | <code>pip install 'apache-airflow[ssh]'</code> | SSH hooks and Oper |
| vertica | <code>pip install 'apache-airflow[vertica]'</code> | Vertica hook support |

From <https://airflow.apache.org/installation.html#extra-packages>

Cloud integration and big data support

- **Several supports with known cloud providers**
 - Microsoft Azure
 - Amazon Web Services
 - Databricks
 - Google Cloud Platform
- **Big data supports**
 - Hadoop, Hive, Druid, Presto
- **Distributed execution**

Airflow workflow structure

- **Workflow is a DAG (Direct Acyclic Graph)**
 - a workflow consists of a set of activities represented in a DAG
 - workflow and activities are **programed using Python** – structures described in code
- **Workflow activities are described by Airflow operator objects**
 - tasks are created when instantiating operator objects

Airflow operators/tasks

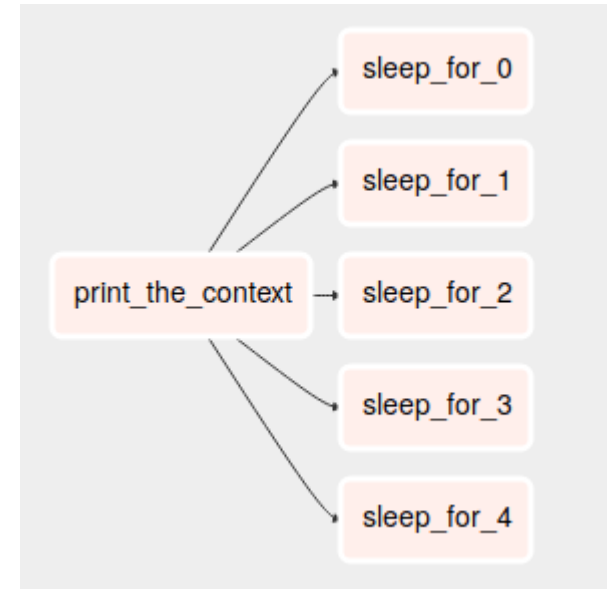
- **Tasks are implemented using operators**
- **Rich set of operators**
 - we can program different kinds of tasks and integrate with different systems
- **Different types of operators for workflow activities**
 - BashOperator, PythonOperator, EmailOperator, HTTPOperator, SqlOperator, Sensor, DockerOperator, HiveOperator, S3FileTransferOperator, PrestoToMysqlOperator, SlackOperator
- **Remember:**
 - such operators will be executed by corresponding services

Example of operators

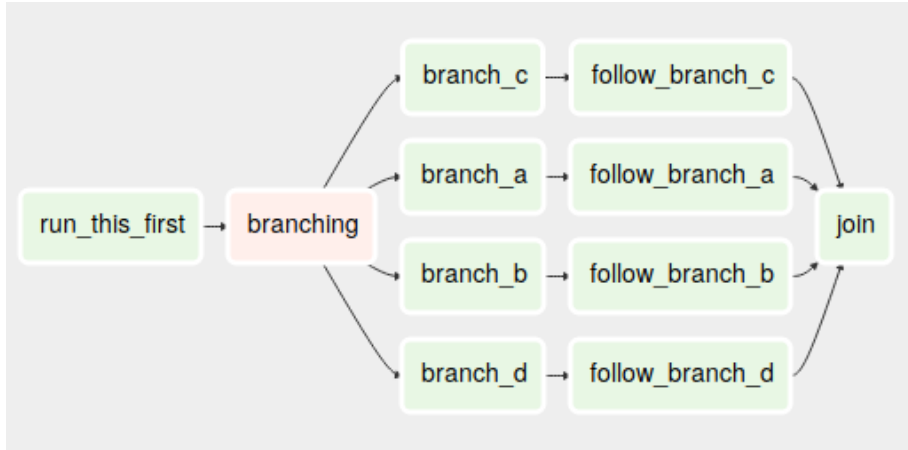
- High-level structure is mapped to python and suitable operators

```
for i in range(5):  
    task = PythonOperator(  
        task_id='sleep_for_' + str(i),  
        python_callable=my_sleeping_function,  
        op_kwargs={'random_base': float(i) / 10},  
        dag=dag,  
    )
```

Code and figures captured from Airflow UI:
DAG: example_python_operator
schedule: None



Example of branching



```
run_this_first = DummyOperator(
    task_id='run_this_first',
    dag=dag,
)

options = ['branch_a', 'branch_b', 'branch_c', 'branch_d']

branching = BranchPythonOperator(
    task_id='branching',
    python_callable=lambda: random.choice(options),
    dag=dag,
)
run_this_first >> branching

join = DummyOperator(
    task_id='join',
    trigger_rule='one_success',
    dag=dag,
)
```

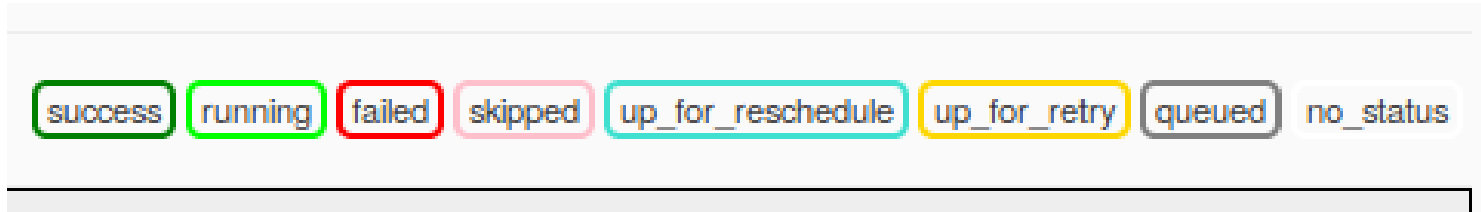
Code and figures captured from Airflow UI
DAG: example_branch_operator
schedule: @daily

Scheduling and execution

- **You can schedule the workflow like a cron job**
 - execute once, every minutes, hours, ...
- **Trigger from external**
 - tasks can be triggered as normal (upstream tasks finishes, dependencies)
 - or specific triggers
- **Very suitable ingestion and batch analytics job managements**
 - the ingestion and analytics are done within tasks

Task lifecycle

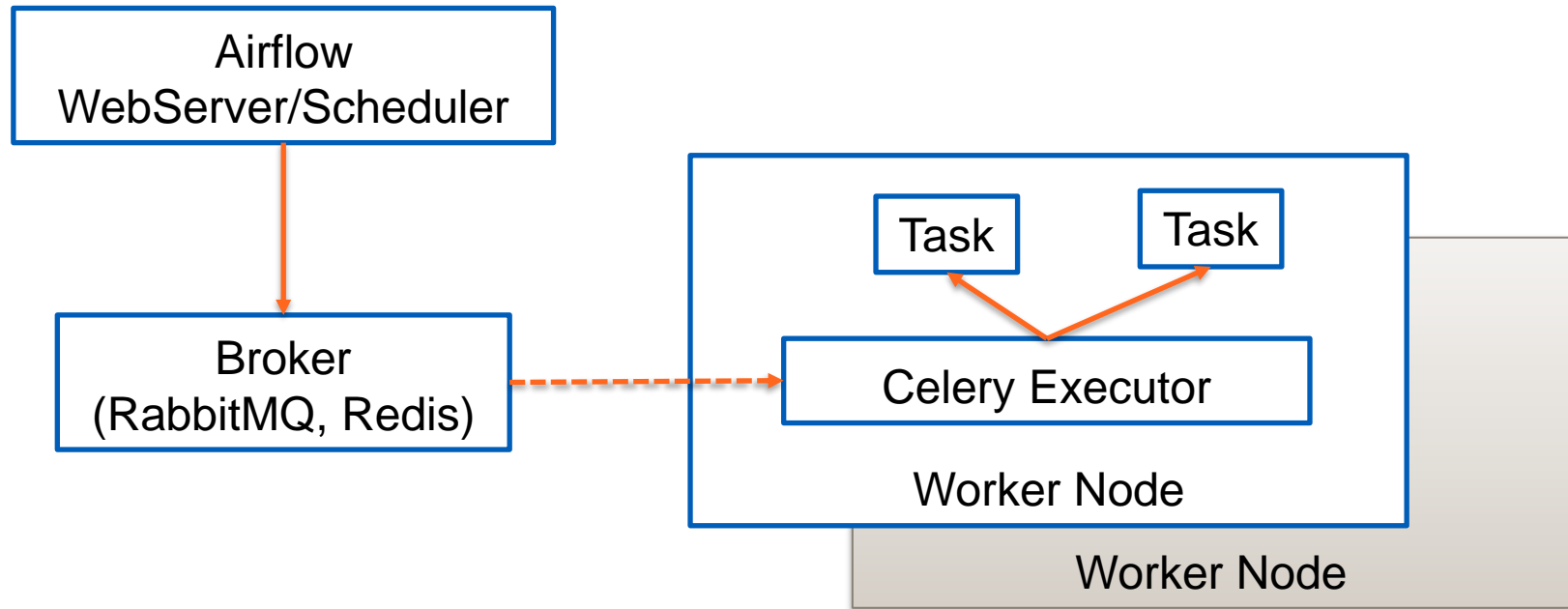
Different states



- Performance metrics can be determined based on states and structures
- Interesting in performance analytics?
 - Check <https://doi.org/10.1016/j.future.2007.01.003>

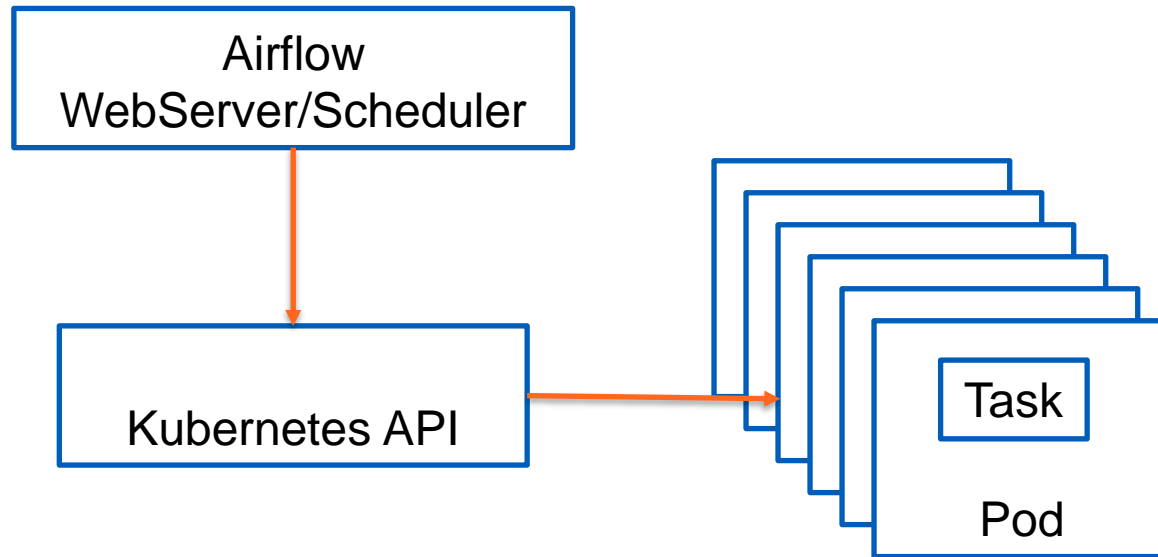
Distributed tasks

You can scale Airflow using workers deployed in different nodes managed by Celery (<http://www.celeryproject.org>)



Distributed tasks

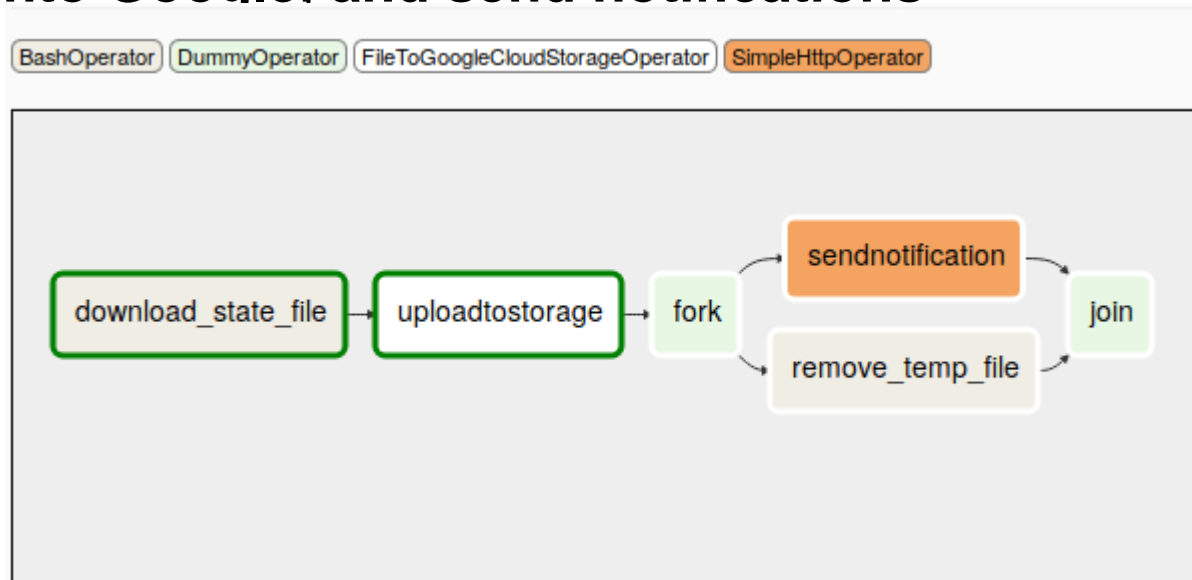
You can scale Airflow to run tasks in Kubernetes



**Google Cloud
Composer: use
Kubernetes**

Example

Scenarios: scan various local servers, obtain log files, store log files into Google. and send notifications



Example for uploading state logs

```
fork = DummyOperator(
    task_id='fork',
    trigger_rule='one_success',
    dag=dag
)

join = DummyOperator(
    task_id='join',
    trigger_rule='one_success',
    dag=dag
)

t_downloadlogtocloud= BashOperator(
    task_id="download_state_file",
    bash_command=downloadlogscript,
    dag = dag
)

t_removefile = BashOperator(
    task_id='remove_temp_file',
    bash_command=removetempfile,
    dag=dag,
)
```

```
## change it suitable to your setting
t_analytics= FileToGoogleCloudStorageOperator(
    task_id="uploadtostorage",
    src=destination_file,
    dst=gcsdir,
    bucket='mybdpairflow',
    google_cloud_storage_conn_id='gcsmybdp',
    dag = dag
)

## change it suitable for your setting
t_sendresult =SimpleHttpOperator(
    task_id='sendnotification',
    method='POST',
    http_conn_id='notificationserver',
    endpoint='api/logUpdate',
    data=json.dumps({"source_file": source_file}),
    headers={"Content-Type": "application/json"},
    dag = dag
)
```

In our GIT course (tutorials)

Example for uploading state logs

upstream task

```
'''  
the dependencies among tasks  
'''  
t_downloadlogtocloud >> t_analytics  
t_analytics >> fork  
fork >> t_sendresult  
t_analytics >> fork  
fork >> t_removefile  
t_removefile >> join  
t_sendresult >> join
```

downstream
task

Monitoring UI

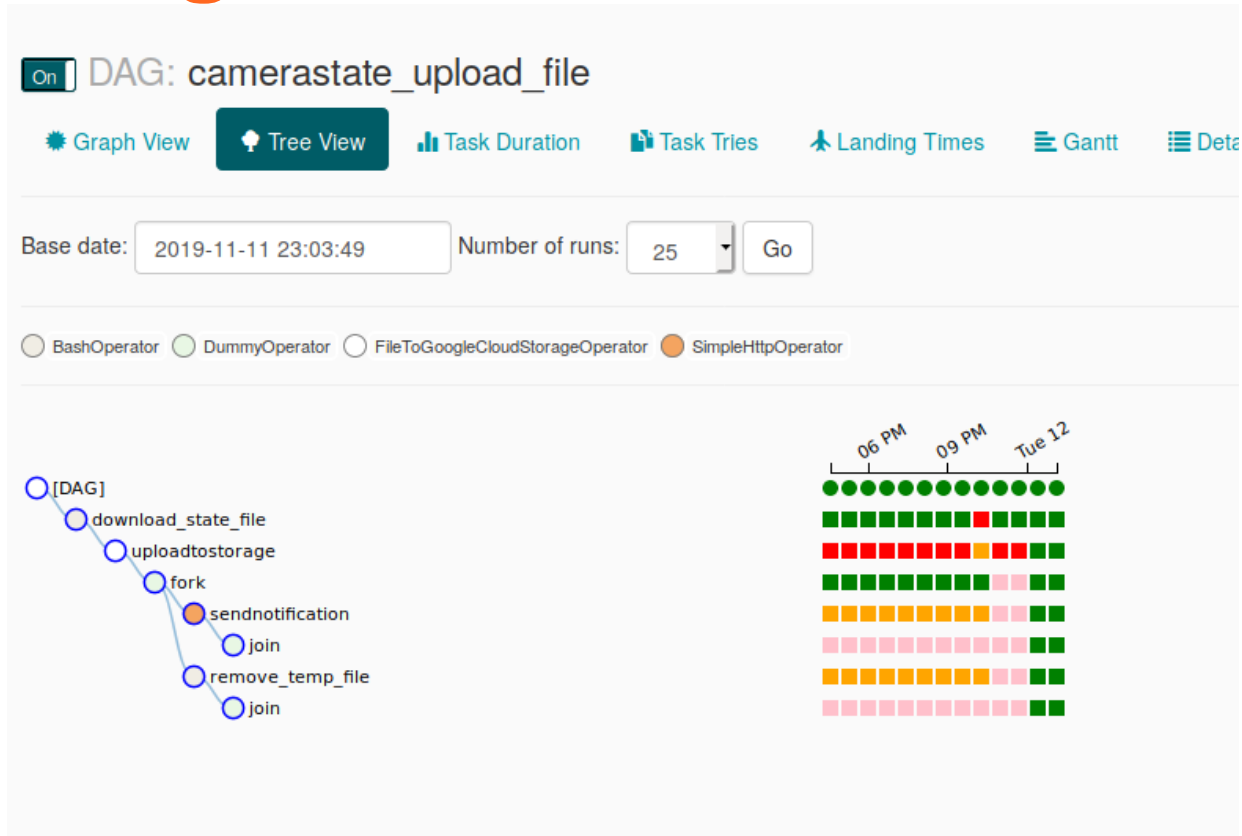
DAGs

Search:

| | ⓘ | DAG | Schedule | Owner | Recent Tasks ⓘ | Last Run ⓘ | DAG Runs ⓘ | Links |
|--|-----|---|----------------|------------------|----------------|--------------------|------------|-------|
| | On | camerastate_upload_file | None | hong-linh-truong | | 2019-11-11 23:03 ⓘ | | |
| | Off | example_bash_operator | 0 0 * * * | Airflow | | 2019-11-11 14:47 ⓘ | | |
| | Off | example_branch_dop_operator_v3 | */1 * * * * | Airflow | | | | |
| | Off | example_branch_operator | @daily | Airflow | | | | |
| | Off | example_http_operator | 1 day, 0:00:00 | Airflow | | 2019-11-11 14:48 ⓘ | | |
| | Off | example_passing_params_via_test_command | */1 * * * * | airflow | | | | |
| | Off | example_plg_operator | None | Airflow | | | | |
| | Off | example_python_operator | None | Airflow | | | | |
| | Off | example_short_circuit_operator | 1 day, 0:00:00 | Airflow | | | | |
| | Off | example_skip_dag | 1 day, 0:00:00 | Airflow | | | | |
| | Off | example_subdag_operator | @once | Airflow | | | | |
| | Off | example_trigger_controller_dag | @once | airflow | | | | |
| | Off | example_trigger_target_dag | None | Airflow | | | | |
| | Off | example_xcom | @once | Airflow | | | | |
| | Off | latest_only | 4:00:00 | airflow | | | | |
| | Off | latest_only_with_trigger | 4:00:00 | airflow | | | | |
| | Off | test_utils | None | airflow | | | | |
| | Off | tutorial | 1 day, 0:00:00 | Airflow | | | | |

Showing 1 to 18 of 18 entries

Monitoring UI



Example with serverless computing/function-as-a-service

Serverless/Function-as-a-service

- **Key principles**
 - running code without managing complex back-end server/application server infrastructures
 - tasks in your application: described as functions
 - *As you typically see in your Java/Python/JavaScript code*
 - functions are uploaded to FaaS engines and will be executed based on different triggers (e.g., direct call or events)
- **Event-driven triggers!**
 - triggered from HTTP calls, messages (brokers), storage events (e.g. new files are stored)

Note: Serverless technologies are lectured *in CS-E4190 - Cloud Software and Systems*

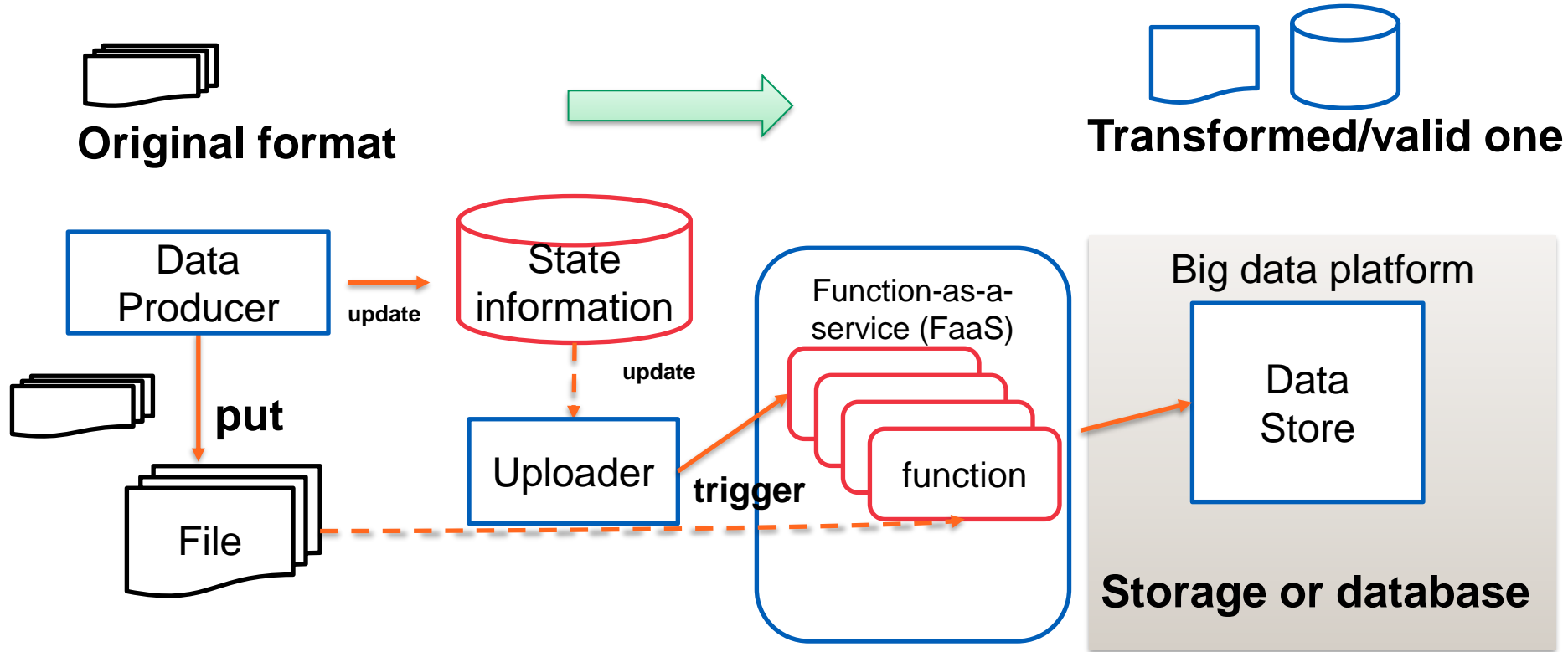
Function-as-a-service and workflows

- **Basic tasks can be implemented as a function**
 - follow the function-as-a-service model (serverless)
 - FaaS engine is a service for executing tasks
- **Workflow coordination with additional features**
 - using coordination engines: explicitly coordinate functions by calling functions executed by FaaS engines
 - implicitly coordinate functions –as a workflows - using connectors and triggers

Using function-as-a-service in clouds

- **Azure**
 - Azure Function for tasks
 - Azure durable functions (<https://docs.microsoft.com/en-us/azure/azure-functions/durable/durable-functions-orchestrations>)
- **Amazon Web services**
 - AWS Lambda for tasks/functions
 - AWS Step Functions (<https://aws.amazon.com/step-functions/>)

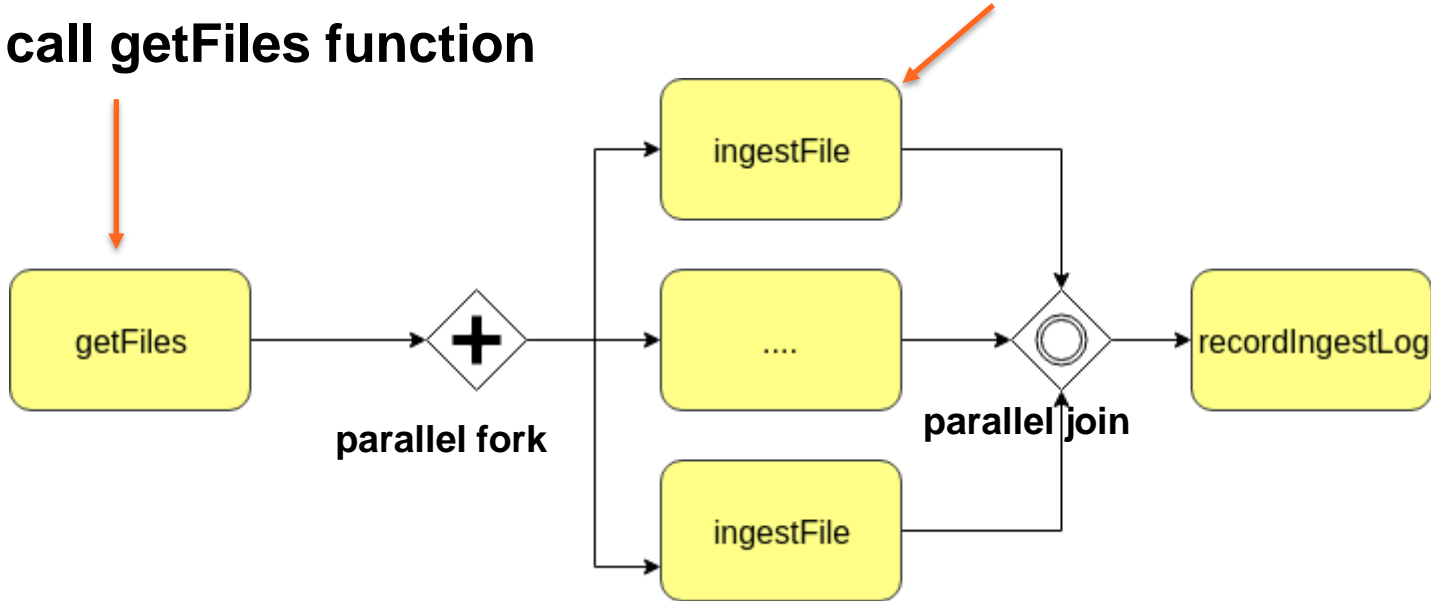
Recall: data ingestion



Example of Azure durable functions

Activity call ingestFile function

Activity call getFiles function



Code

```
module.exports = df.orchestrator(function* (context, req) {
  const tasks = [];

  //....
  //....

  const files = yield context.df.callActivity("getFiles", username);
  for (const file in files) {
    console.log("Deal with file "+file);
    tasks.push(yield context.df.callActivity("ingestFile", file));
  }
  return tasks;
});
```

Pros and cons

- **Pros:**

- work well in big data ecosystems offered with cloud providers
- easy to integrate with other cloud services

- **Cons**

- lock-in with big cloud providers
 - *Difficult if you must change services in your ecosystem*
- limited coordination/workflow engines compared with well-known scientific/data-intensive analytics workflows

Summary

- **Focus:**
 - practical programming with:
 - *Apache Airflow: for data analytics and platform management*
 - *Function-as-a-service: for service integration*
 - *Kubeflow: for machine learning with big data platforms (if you like ML)*
- **Action:**
 - hands-on and work on concrete examples
 - *Try to see if you can implement previous use cases/scenarios in your work with workflows*
 - offering workflows as a service in your platform!

Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io