



Aalto University  
School of Science

# Workflows for Big Data Platforms

*Hong-Linh Truong*

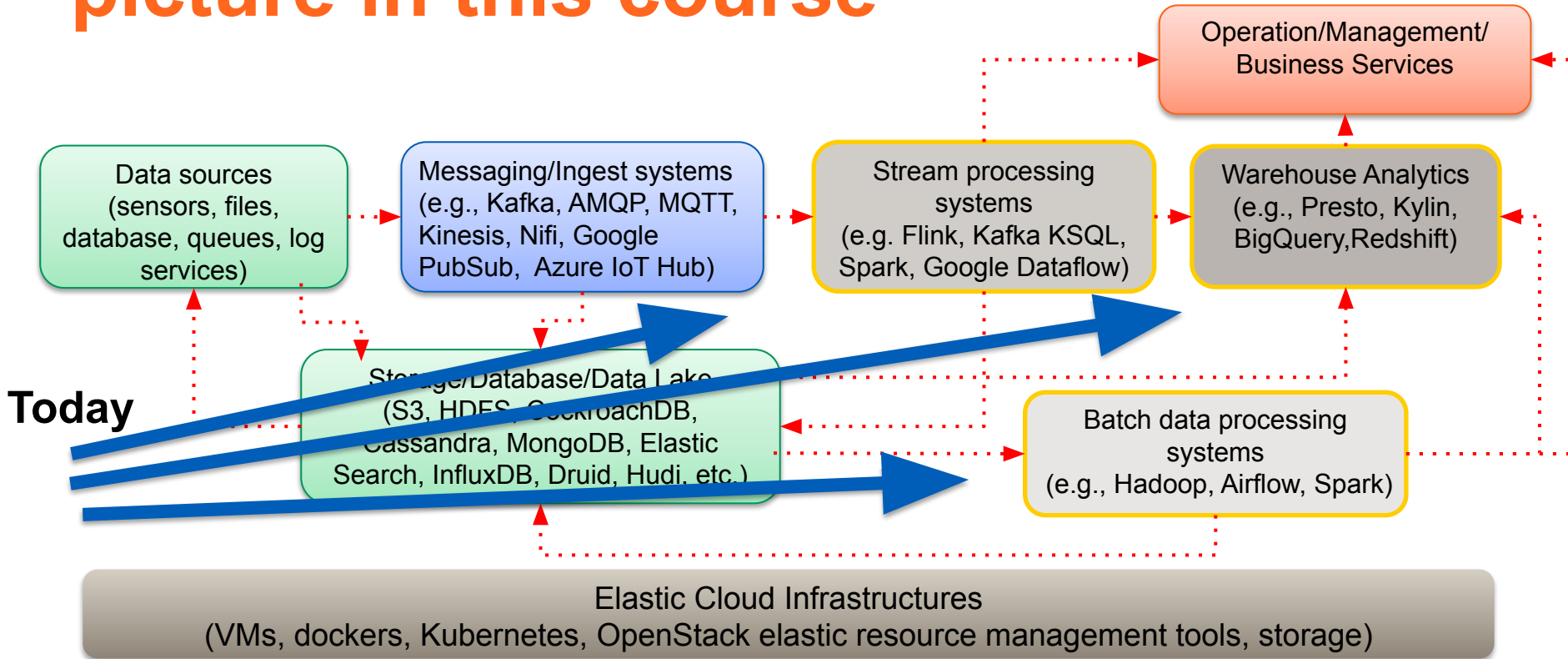
*Department of Computer Science*

*[linh.truong@aalto.fi](mailto: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

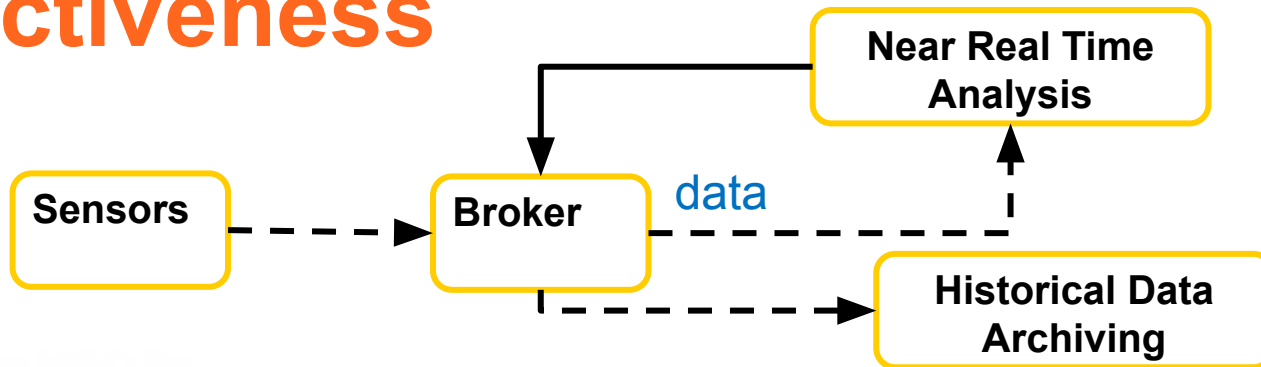
- **Loosely coupled data related tasks**
  - Data collection and transformation
    - *data transfers, extraction, transformation,*
  - Data processing, including machine learning
    - *data analytics, training, serving machine learning algorithms*
- **Loosely coupled platform automation tasks**
  -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

# Many complex use cases

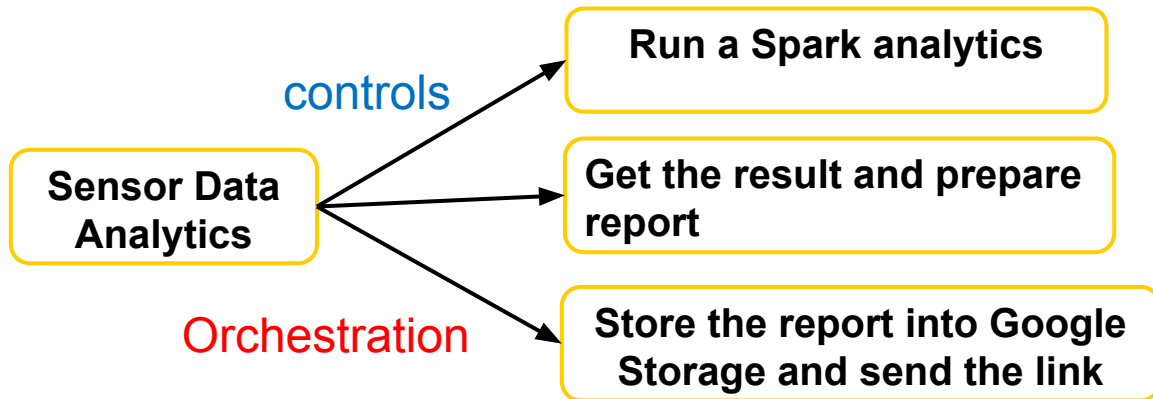
- **Deployment and configuration**
  - for executors/execution environments and platform components
- **ETL, data cleansing and backup**
  - access and coordinate many different compute services, data sources, ingestion and extraction applications
- **Predictive maintenance**
  - coordination of machine learning pipelines and communication with humans/optimization services
- **Analytics-as a service**
  - metrics understanding, user activities analytics, and customer understanding

# Recall: Orchestration and Reactiveness

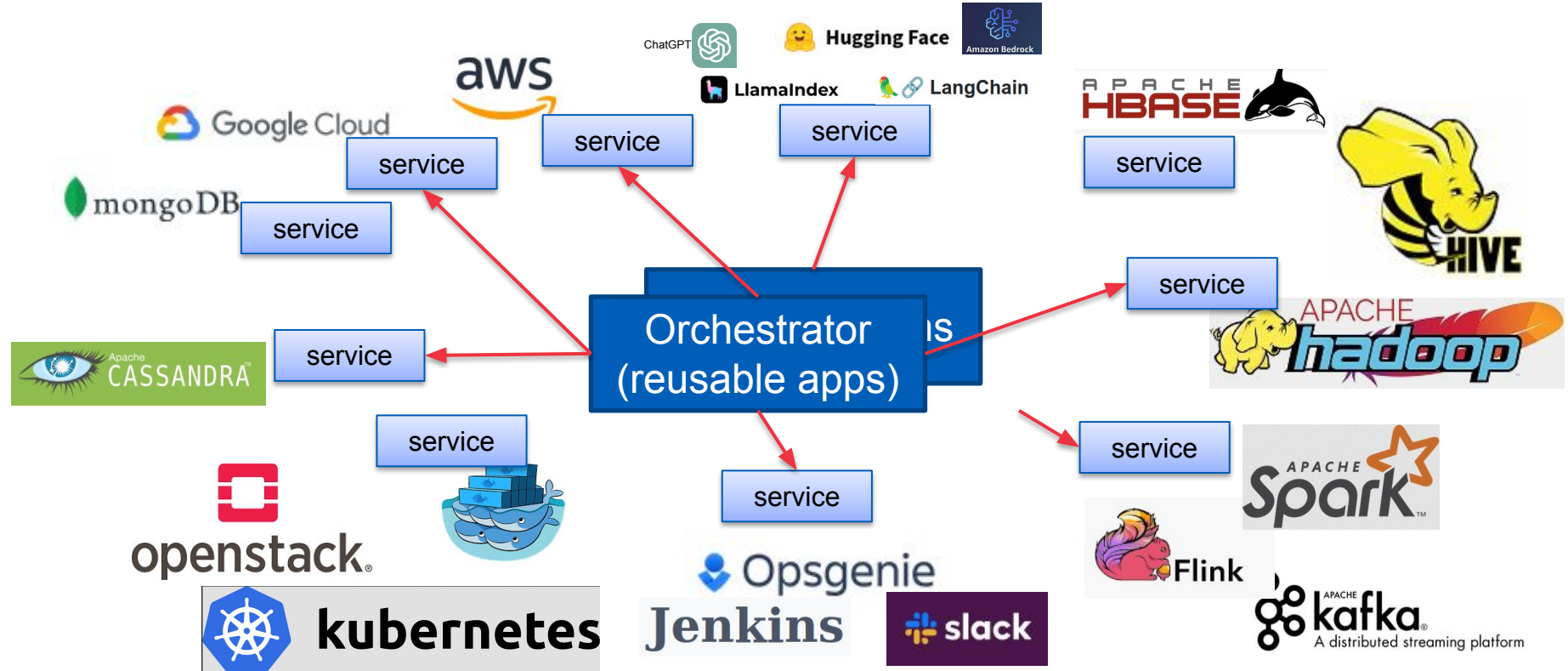
reactiveness



*Several design patterns you have done in previous lectures*

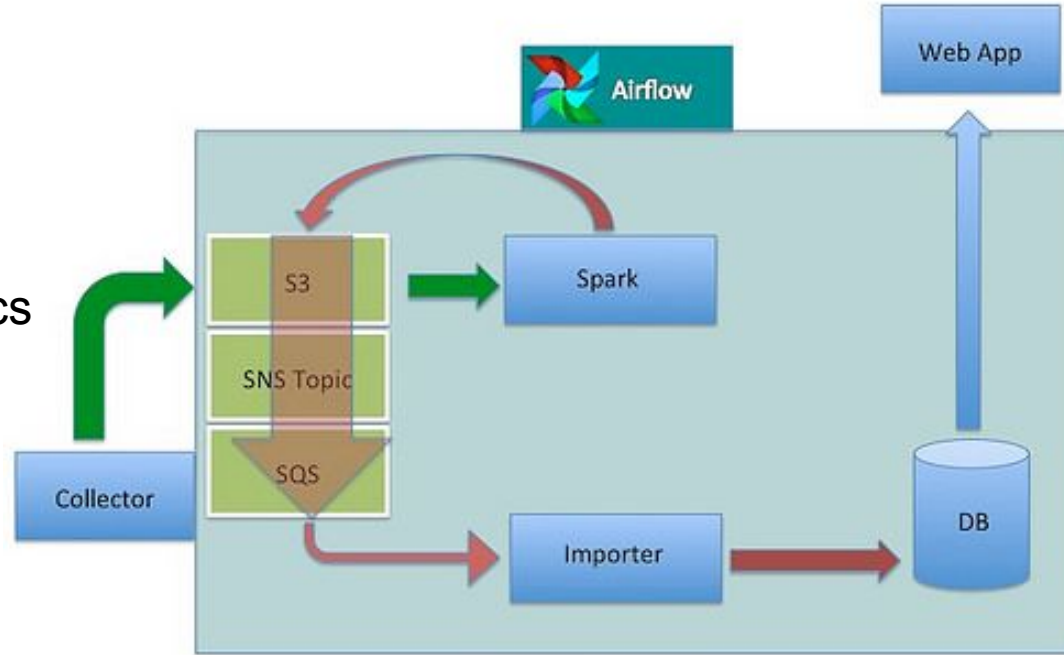


# Service orchestration in big data platforms: more than just with “big data”



# Example of security data analytics

Security-related information and metrics from distributed customers



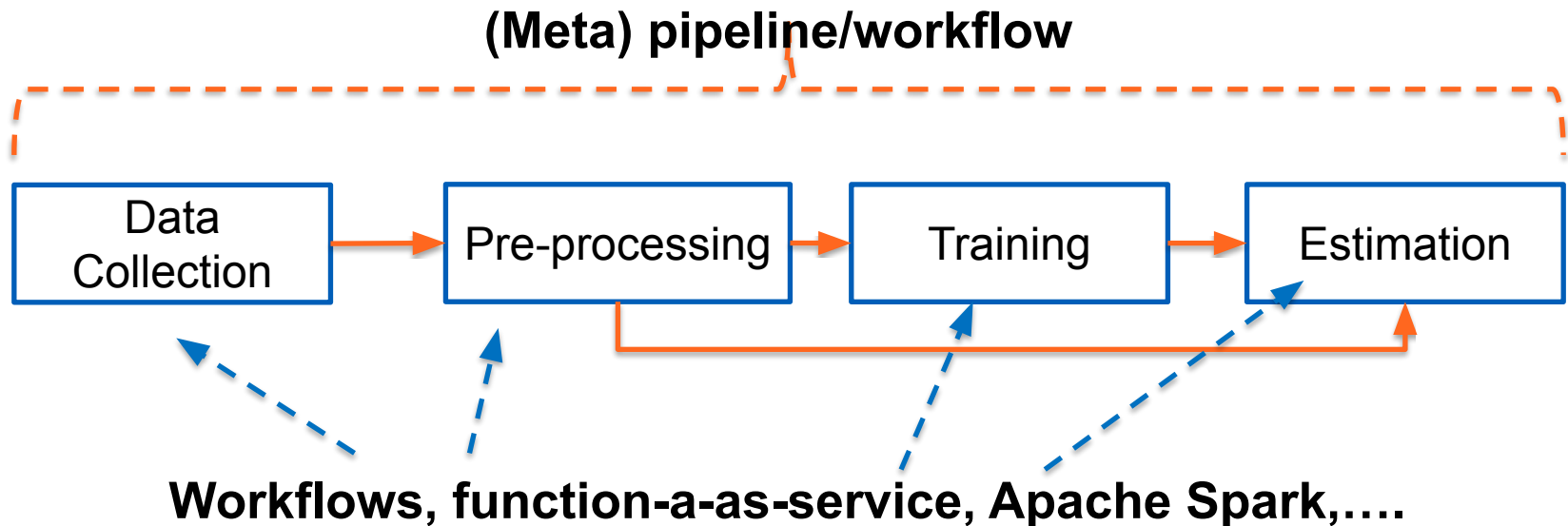
**Figure Source:**

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



# Example of ML workflows

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



# 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 Krists, „Quality of analytics management of data pipelines for retail forecasting“, Aalto CS Master thesis, 2019

# Workflows

## A workflow specifies a process

- consists of a set of connected tasks/activities
- has tasks/activities carried out by diverse types of software services or humans, each performs a function
- can be automated with/without human intervention
- has data/control task dependencies
- can be reusable (tasks, part of the workflow, and the whole workflow)

# 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
  - including IoT devices, robots, AI/LLM agents

- **Workflow**

- coordinate/orchestrate many tasks:
  - *the function of tasks **is not** really “carried out/executed” 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 workflow  $\Rightarrow$  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](https://en.wikipedia.org/wiki/Pipeline_%28computing%29)

- **Two interpretations in practice:**

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

- **Note: sometimes the “data pipeline” here is just an abstract design**

# 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.



# 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.

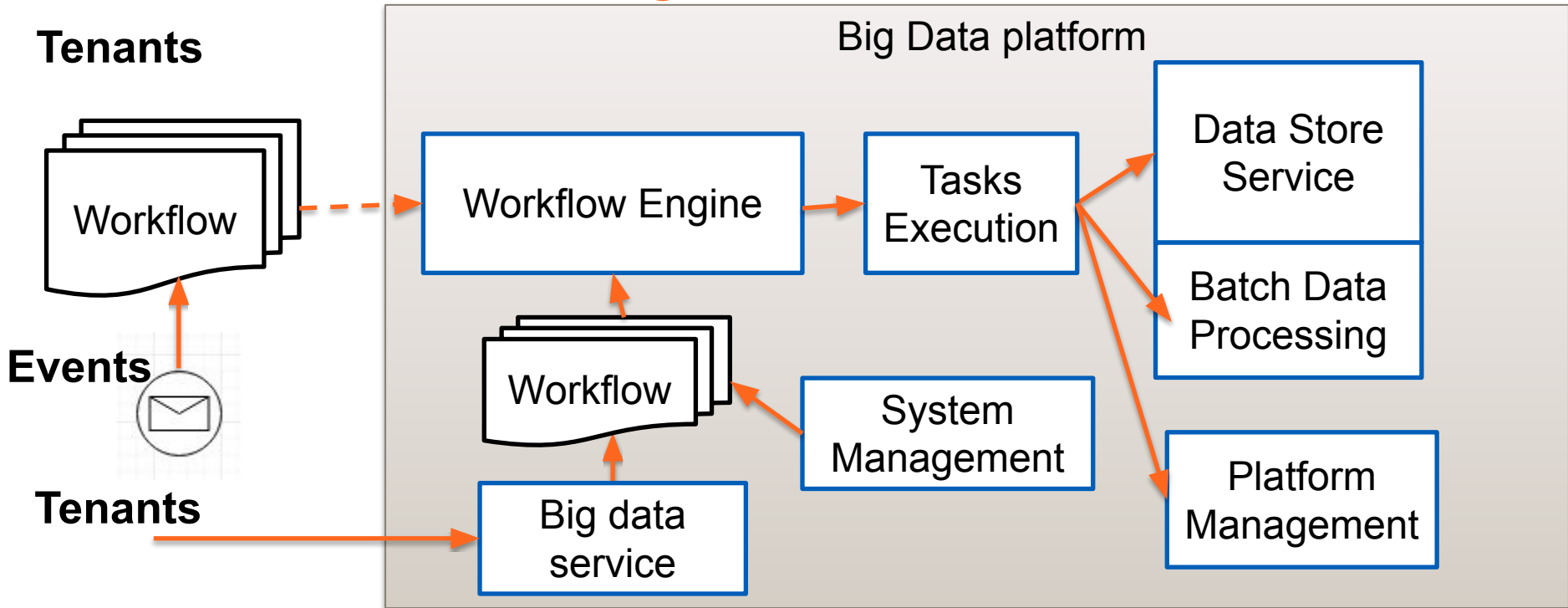
- **Workflow languages**

- structure/describe tasks, dataflows, and control flows

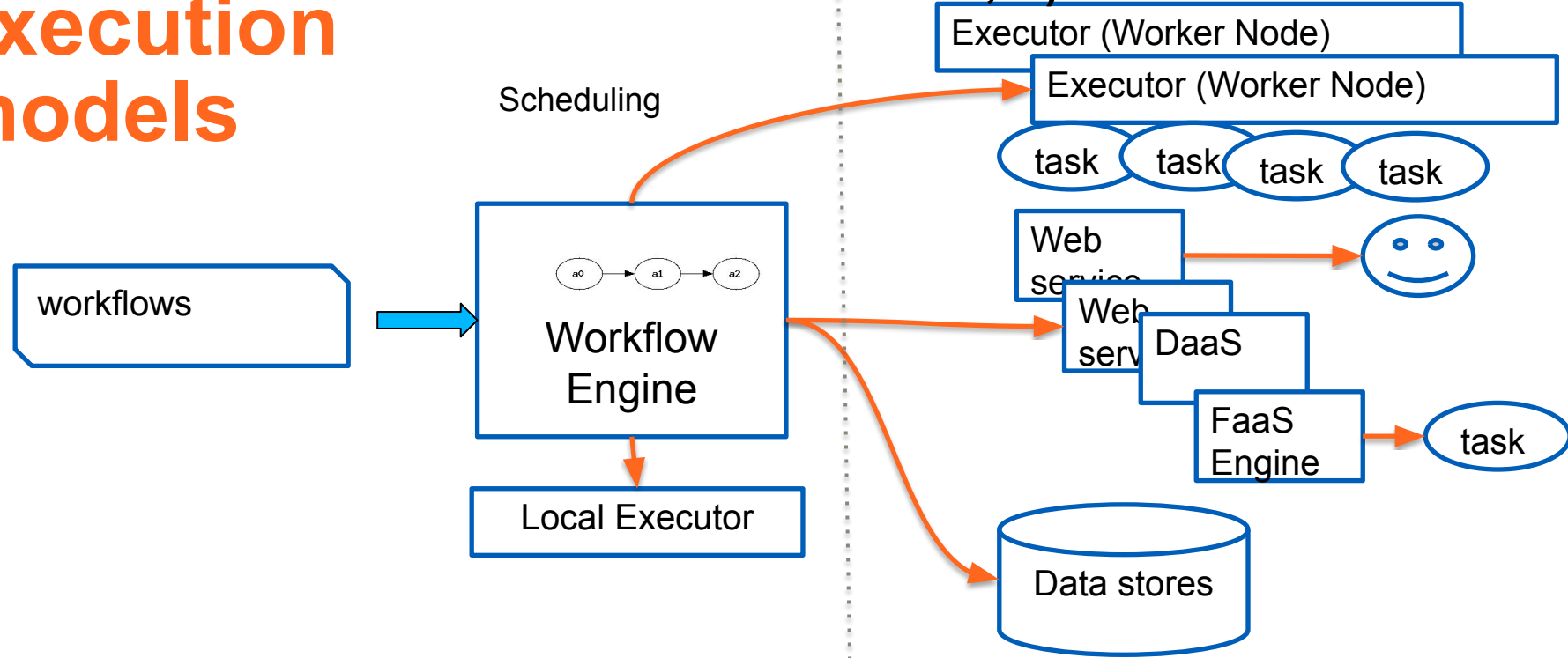
- **Workflow engines**

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

# Workflows in big data platforms: more than analytics

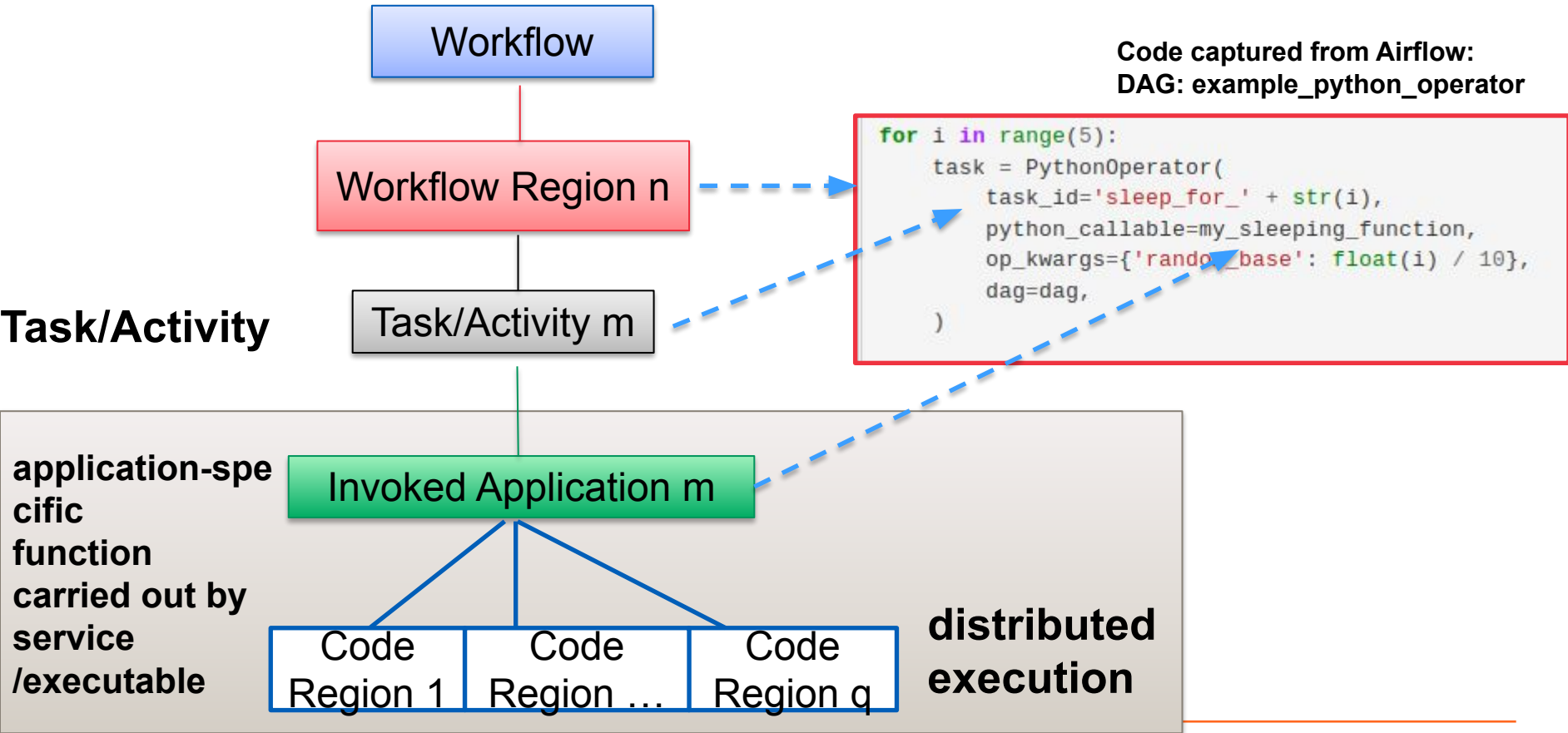


# Common workflow execution models



Major works are carried out in distributed nodes (manage by Kubernetes, Celery, Dask, Ray, Slurm,...)

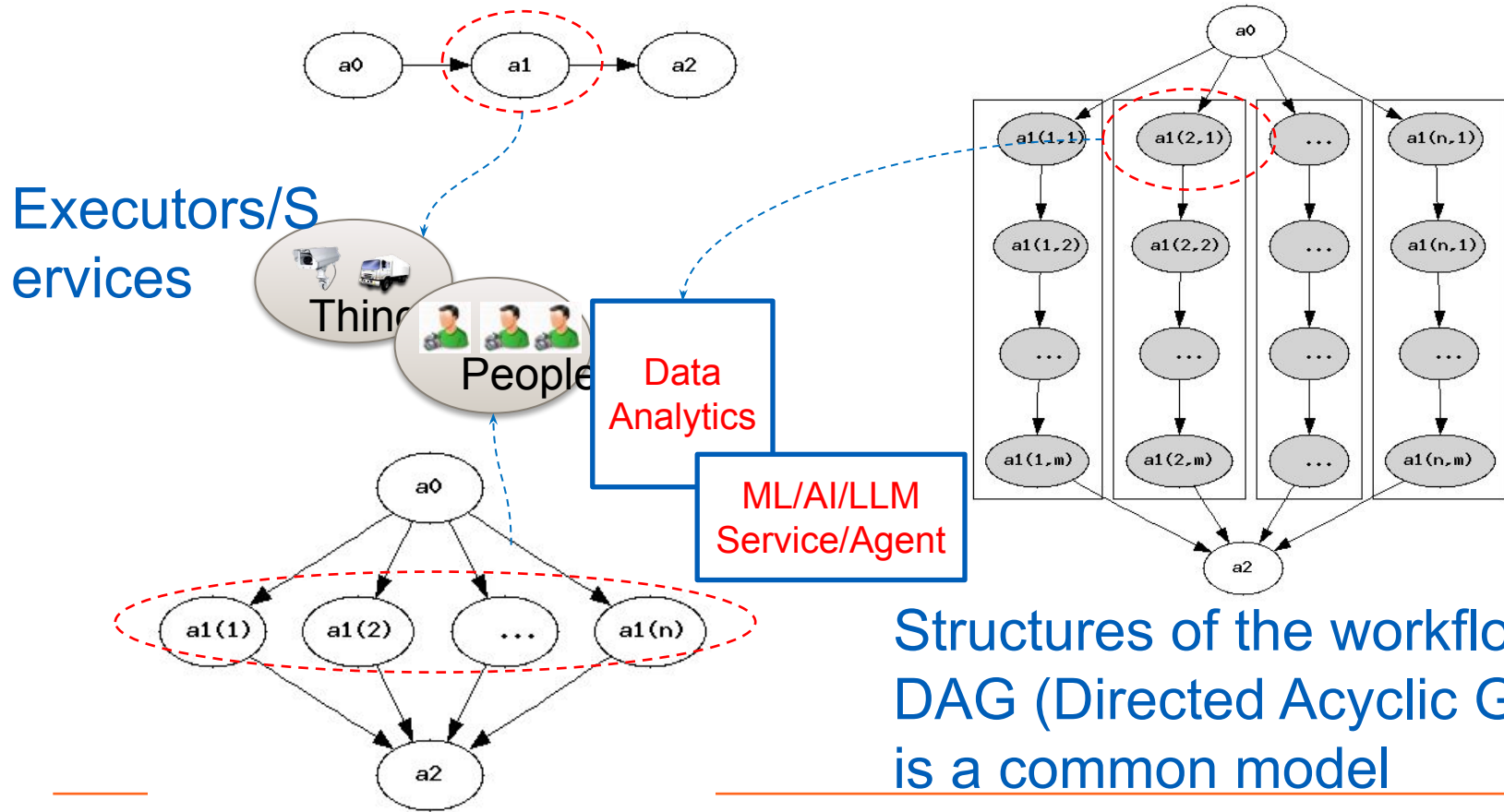
# Structured view of workflows



# 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
  - data processing workflows: common in data science and data analysis
- **Descriptive languages with declarative schemas**
  - BPEL, YAML, JSON and several languages designed for specific workflow engines
  - common in business and scientific workflows

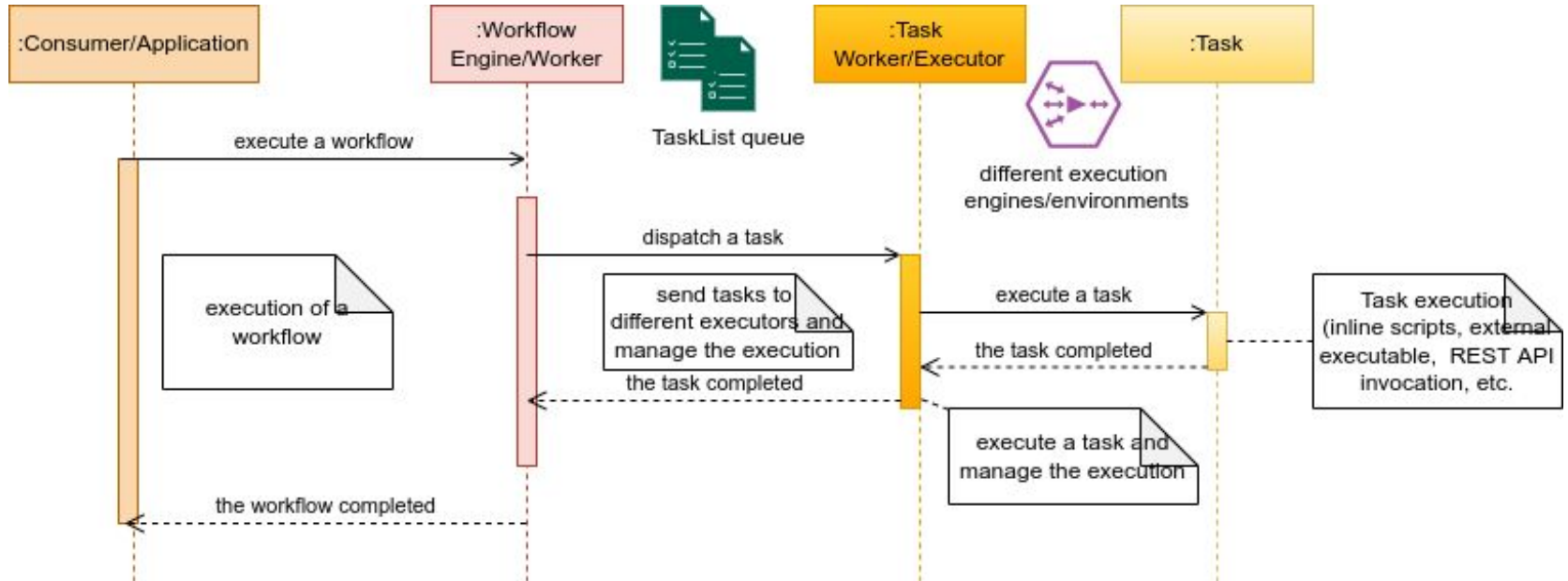
# Task dependencies & orchestration



# Modern data workflows in clouds

- **Invoked applications**
  - Cloud services: with diverse types of APIs and protocols
  - Serverless functions
  - AI/ML/LLM agents/services
- **Workers/Executors**
  - Distributed workers in data centers
    - *interfaces via APIs and messaging systems*
  - Containers and distributed task executors
  - Cloud orchestration
- **Python-based for data analytics and ML workflows**
  - mainly for data analysis

# Complex execution model across distributed machines



**Challenges in managing dependencies, failures and performance**



# Runtime aspects

- **Parallel and distributed execution**

- tasks are executed in different machines (by external invoked applications/services) ⇒ multiple running workflows in the same system

- **Long and/or periodic 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

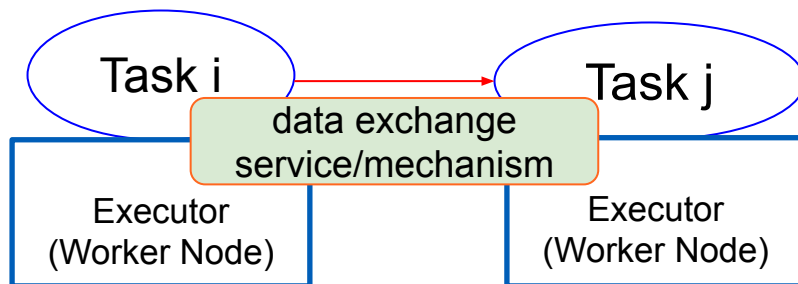
# Select/build workflows in your platforms

- **Rich data services**
  - for data storing/retrieving tasks
- **Big data computation engines**
  - for data processing tasks with different workloads: 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)

# State and exchange data

- How do tasks exchange data?

By **value** or by **reference** exchange?



Possibilities:

- shared file systems
- global/common storage systems
- key/value database
- external processes for pulling data

Some important aspects:

- Asynchronous task execution in which the task results must be retrieved with a different method
- Idempotency (retries, fault-tolerance) and caching
- SerDe (serialization and deserialization)
- Performance of shared systems/services for data exchange
- Deployment consideration

# Select/build workflows in your platforms

- **Scheduling**

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

- **Elasticity**

- Elasticity controls of virtualized resources (workers/executors, e.g. VMs/containers/Kubernetes) for executing tasks

- **Multiple levels of concurrency/parallelism**

- Cluster level vs node level

- **Examples**

- Periodic cron schedules, backfill, opportunistic schedules
- Increase number of distributed workers/cluster sizes
- Heterogeneous 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>

# Monitoring

- **Understand the states**
  - Workflow level vs task/activity level vs invoked applications level
- **Multiple levels of instrumentation and monitoring**
  - Workflow Engine/Worker
  - Scheduler
  - Task worker/executor
  - External services/applications

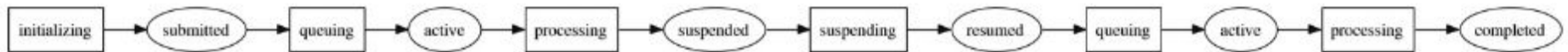
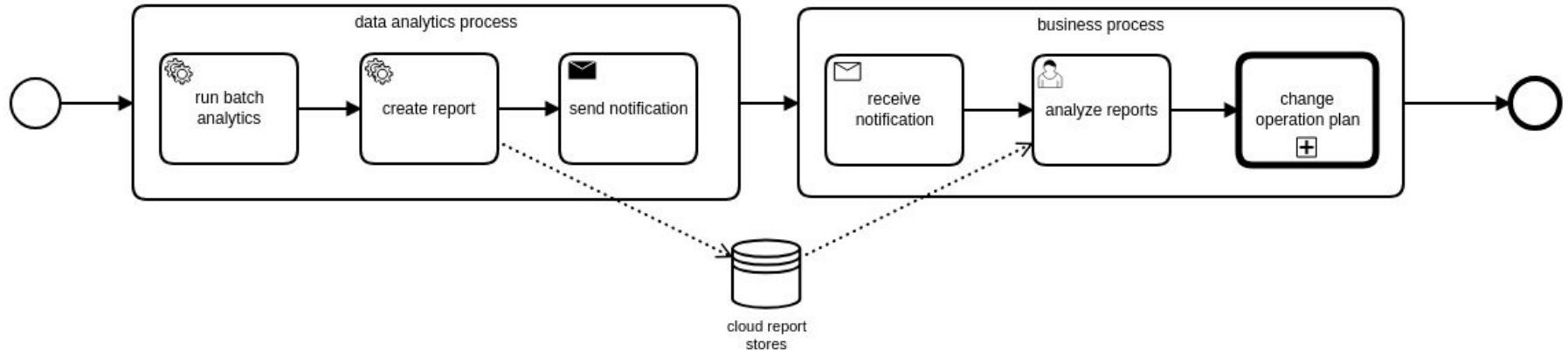


Fig. 3. Discrete process model of the tracing execution of an activity. □ represents an execution phase, ○ represents an event.

# Select/build workflows in your platforms

- **Integration**

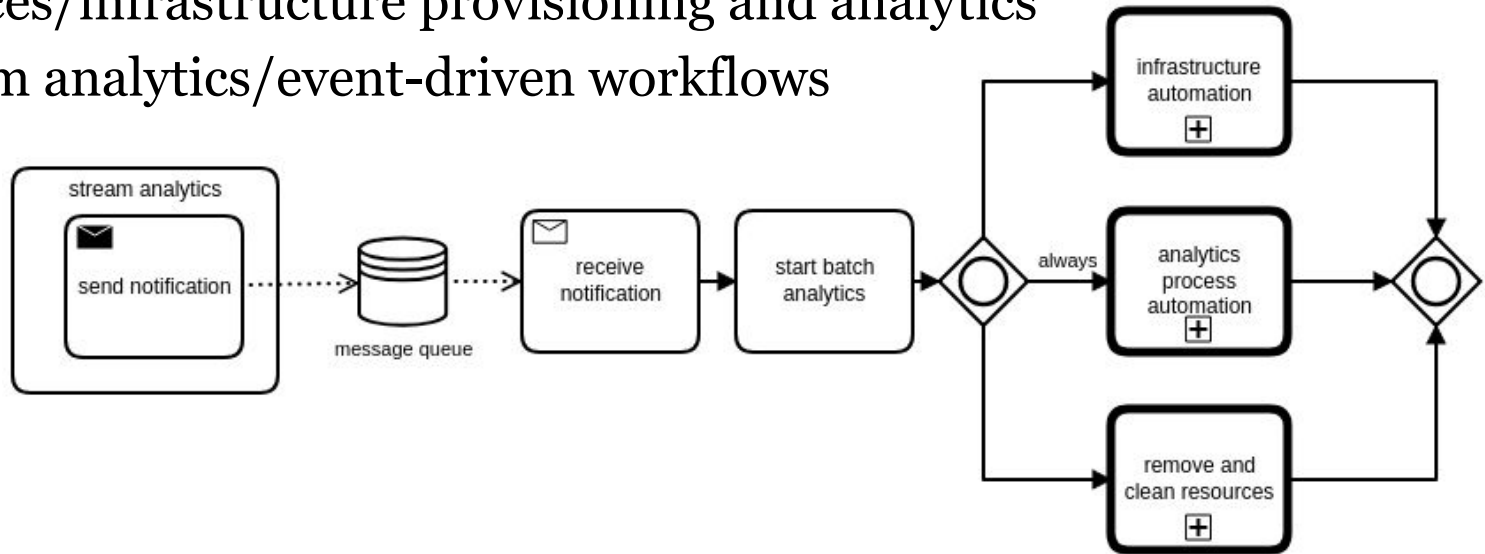
- data analytics processes and business processes
- include human-in-the-loop



# 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:**
  - for both data processing and service management

# Many connectors

- Airbyte
- Alibaba
- Amazon
- Apache Beam
- Apache Cassandra
- Apache Drill
- Apache Druid
- Apache HDFS
- Apache Hive
- Apache Kylin
- Apache Livy
- Apache Pig
- Apache Pinot
- Apache Spark
- Apache Sqoop
- Asana
- Celery
- IBM Cloudant
- Kubernetes
- Databricks
- Datadog
- DBT cloud
- Dingding
- Discord
- Docker
- Elasticsearch
- Exasol
- Facebook
- File Transfer Protocol (FTP)
- Github
- Google
- gRPC
- Hashicorp
- Hypertext Transfer Protocol (HTTP)
- Influx DB
- Internet Message Access Protocol (IMAP)
- Java Database Connectivity (JDBC)
- Jenkins
- Jira
- Microsoft Azure
- Microsoft PowerShell Remoting Protocol (PSRP)
- Microsoft SQL Server (MSSQL)
- Windows Remote Management (WinRM)
- MongoDB
- MySQL
- Neo4J
- ODBC
- OpenFaaS
- Opsgenie
- Oracle
- Pagerduty
- Papermill
- Plexus
- PostgreSQL
- Presto
- Qubole
- Redis
- Salesforce
- Samba
- Segment
- Sendgrid
- SFTP
- Singularity
- Slack
- Snowflake
- SQLite
- SSH
- Tableau
- Telegram
- Trino
- Vertica
- Yandex
- Zendesk

From <https://airflow.apache.org/docs/>

# Cloud integration and big data support

- **Several supports with known cloud providers**
  - Microsoft Azure
  - Amazon Web Services
  - Databricks
  - Google Cloud Platform (Google Composer)
- **Big data supports**
  - Hadoop, Hive, Druid, Presto
- **Distributed execution**
  - Celery, Dask, and Kubernetes

# Airflow workflow structure

- **Workflow is a DAG (Direct Acyclic Graph)**
  - a workflow consists of a set of activities/tasks represented in a DAG
  - workflow and activities are **programed using Python**
    - the workflow 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, SimpleHttpOperator, BaseSQLOperator, BaseSensorOperator, DockerOperator, HiveOperator, SparkSubmitOperator, SageMakerTrainingOperator, PrestoToMysqlOperator, SlackAPIPostOperator
- **Remember:**
  - such operators will be executed by corresponding services

# Example for uploading state logs

```
--
92 t_download_data = PythonOperator(
93     task_id="download_data",
94     python_callable=download_data,
95     op_kwargs={'source_file':source_file, 'dest_file':temp_dest_file},
96     dag=dag,
97 )
98 #the dest file from the download task will be used for anal
99 t_basic_aggregation = PythonOperator(
100     task_id='alarm_analytic',
101     python_callable=basic_aggregation,
102     op_kwargs={'input_file':temp_dest_file, 'report_destinat
103     dag=dag,
104 )
105 t_uploadgcs = LocalFilesystemToGCSOperator(
106     task_id="upload_local_file_to_gcs",
107     src=report_destination,
108     dst=gcs_dest_file,
109     bucket=GCS_CONF["bucket"],
110     gcp_conn_id=GCS_CONF["gcp_conn_id"],
111     dag = dag
112 )
117 t_insert_data_warehouse = PythonOperator(
118     task_id='insert_data_warehouse',
119     python_callable=data_to_bigquery,
120     op_kwargs={'input_data_src':f'file://{report_destination}',
121               'table_id':BIGQUERY_CONF["table_id"],
122               'project_id':BIGQUERY_CONF["project_id"],
123               'credentials':credentials},
124     dag=dag)
125
126
127 t_msnotification = PythonOperator(
128     task_id='teams_notification',
129     python_callable=post_notification,
130     op_kwargs={'gcs_report':gcs_file_url, 'teams_webhook':teams_webhook},
131     dag=dag)
132
133
134 t_clean_data = PythonOperator(
135     task_id='data_cleansing',
136     python_callable=clean_data,
137     op_kwargs={'dest_files':[temp_dest_file,report_destination]}},
138     dag=dag)
139
```

**In our GIT course (tutorials)**

# Example

upstream task

```
139
140 """
141 the dependencies among tasks
142
143 now you have to remember how different tasks exchange data:
144 - they pass data via files and you use a local file system, but
145 task A and task B are not executed in the same machine
146 - they pass data via a global data storage, then some upload/download of data
147 must be implemented.
148
149 thus, you have to see the task implementation in detail. This example, basically,
150 works only for local or file sharing systems as we implement download, check quality,
151 clean data, etc. using local file systems.
152 """
153
154 t download data >> t basic aggregation >> t uploadqcs >> t insert data warehouse
155 t insert data warehouse >> t msnotification >> t clean data
156
```

downstream task

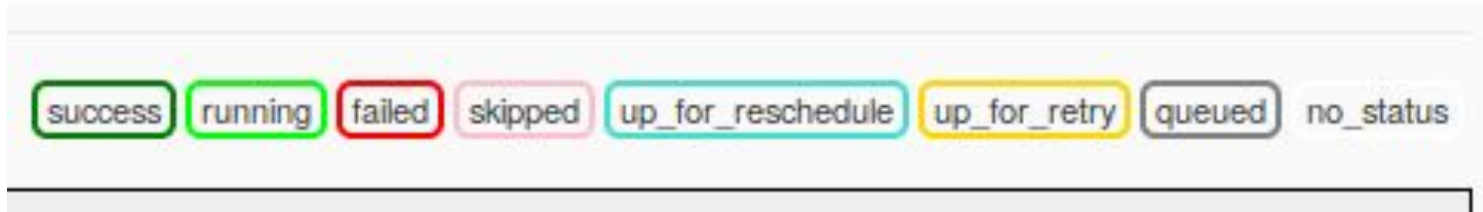


# Scheduling and execution

- **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
  - schedule based on analytics needs

# Task lifecycle

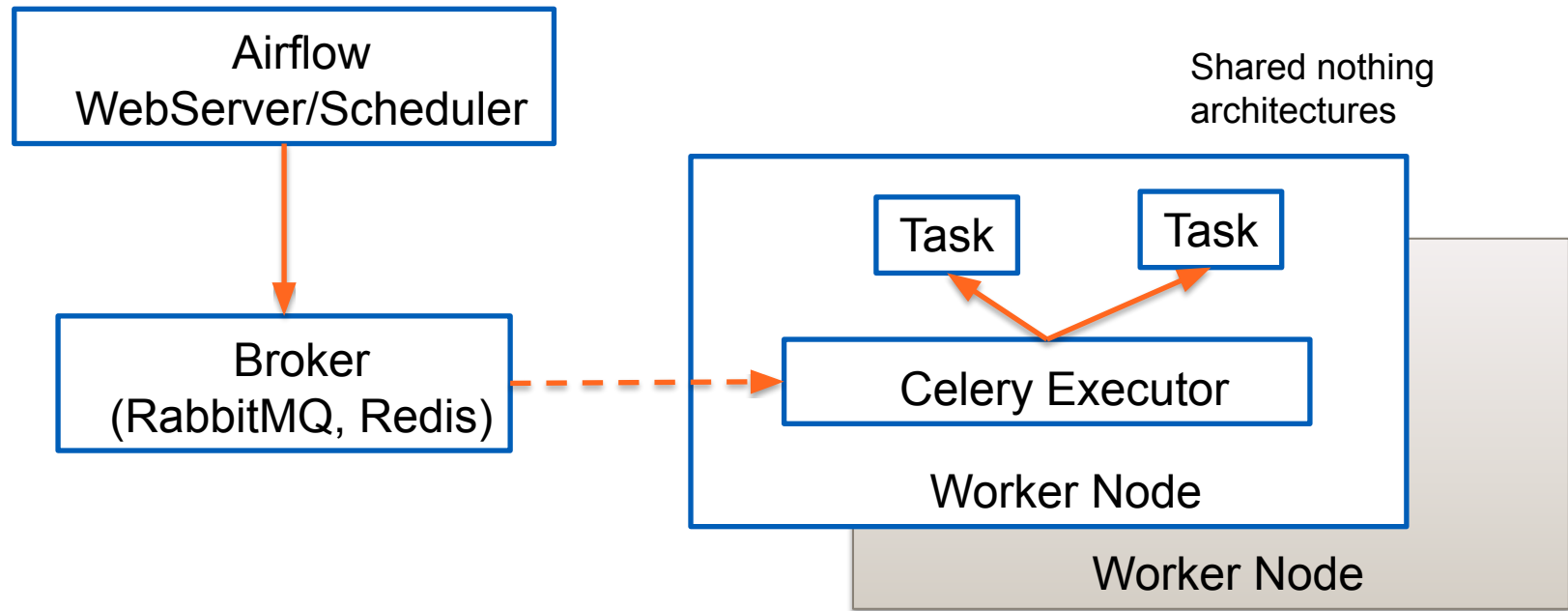
## Different states



- 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>

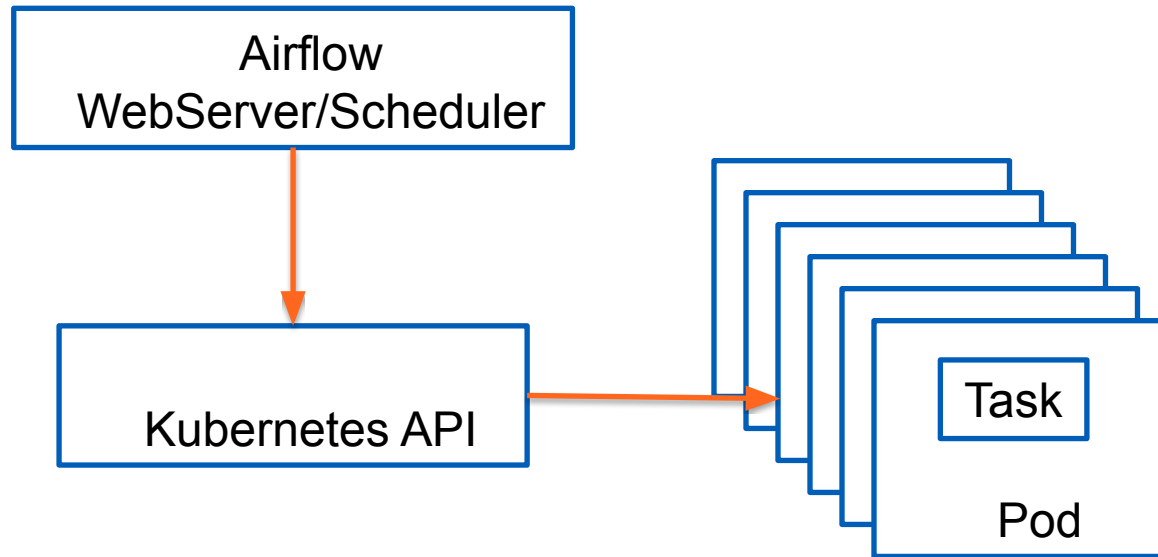
# Distributed tasks

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



# Distributed tasks

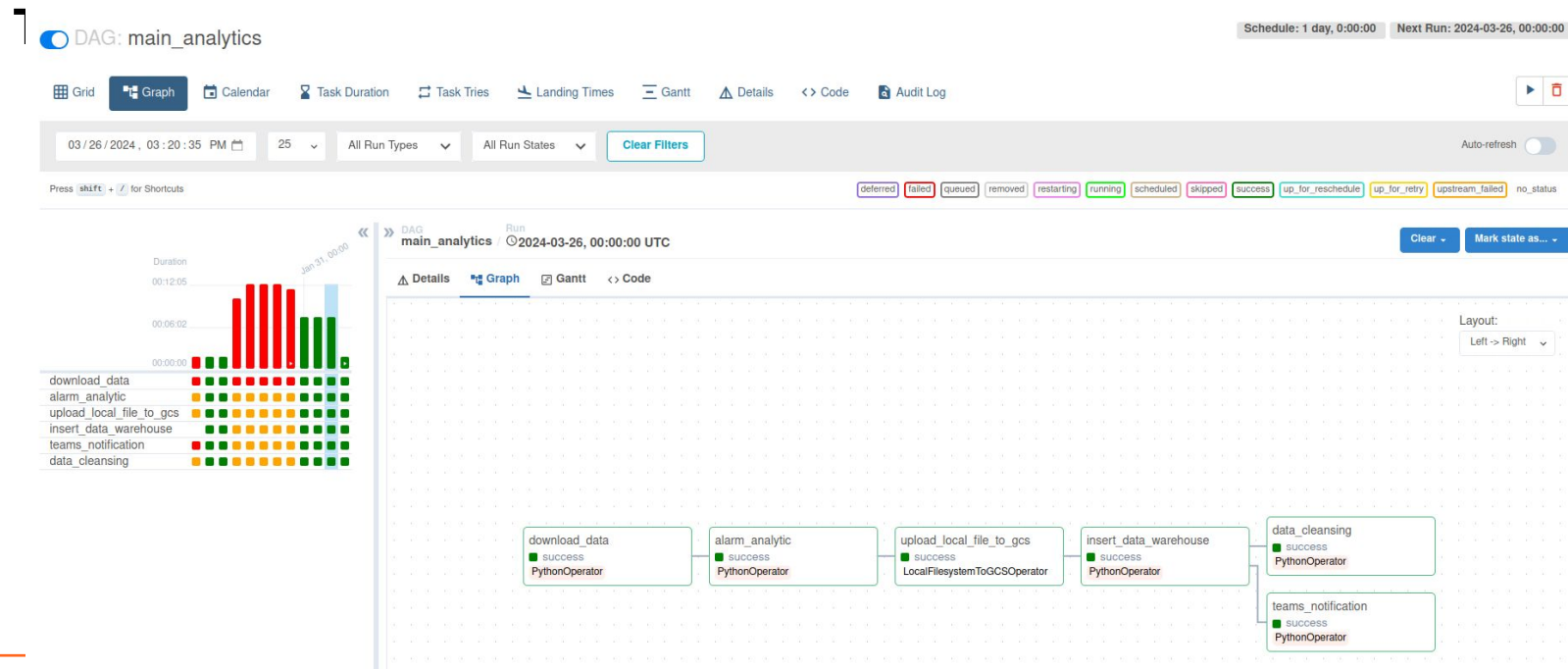
You can scale Airflow to run tasks in Kubernetes



**Google Cloud  
Composer: use  
Kubernetes**

# Monitoring UI

download data from various services (via pre-defined API/endpoints)  $\Rightarrow$  analyze data  $\Rightarrow$  store results to Google storage and BigQuery  $\Rightarrow$  send notifications about the result to



# Summary

- **Focus:**

- practical programming with:
  - *Apache Airflow: for data analytics and platform management*
  - *Serverless workflows using function-as-a-service: e.g., AWS Steps*
  - *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!
  - *suggest to do some hands-on by configuring and deploying Airflow*

# Thanks!

**Hong-Linh Truong**  
**Department of Computer Science**

**rdsea.github.io**