Building Scalable Data Integration Pipelines Using Container Technology

Kasun Samarasinghe & Pierre-Andre Michel
CALIPHO Group
Swiss Institute of Bioinformatics
Geneva, Switzerland

Sessions

01

neXtProt Data Integration

02

Container Technology

03

Workflow Management with Airflow

04

Containerized Data Integration Pipelines

05

QA & Discussion

01

neXtProt Data Integration

Introduction – What is neXtProt

neXtProt: a **human protein** knowledge base

- Integrates knowledge collected from different data sources
- Visualizes the protein annotations in multiple views
- Full-text search engine
- Advanced search engine based on SPARQL (structured queries)
- REST API
- SPARQL endpoint: federated queries, LOD cloud
- Download from ftp (XML, RDF/ttl, fasta, PEFF, csv, …)

Isoform centric, maximizes usage of controlled vocabularies and ontologies

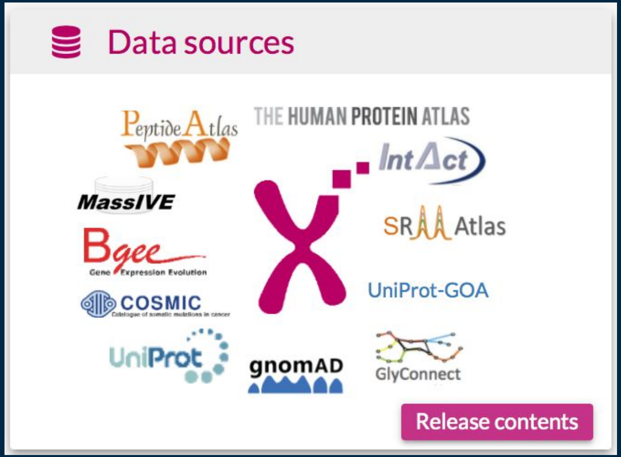
https://www.nextprot.org https://api.nextprot.org

https://snorql.nextprot.org https://sparql.nextprot.org/

ftp://ftp.nextprot.org/

Introduction – neXtProt content and main datasources

- Function: UniProt, GOA
- Interactions: UniProt, GOA, IntAct
- Expression: BGee, Human Protein Atlas
- Variants: UniProt, dbSNP, COSMIC, GnomAD
- Proteomics: UniProt, PeptideAtlas, SRMAtlas, MassIVE
- Gene mapping: UniProt, Ensembl
- ...



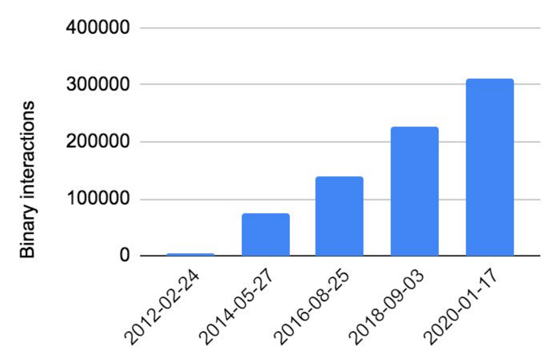
Introduction – Data volume over time / impact

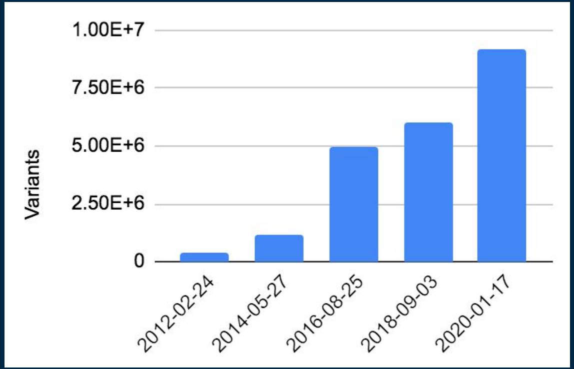
Still 20’000 proteins but

More datasources

More annotations

More terms & publications

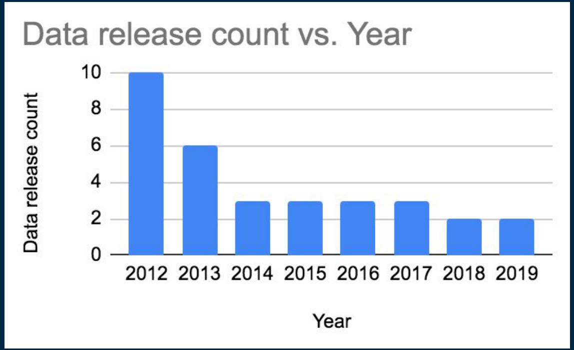




=> More dev & maintenance

=> Performance issues

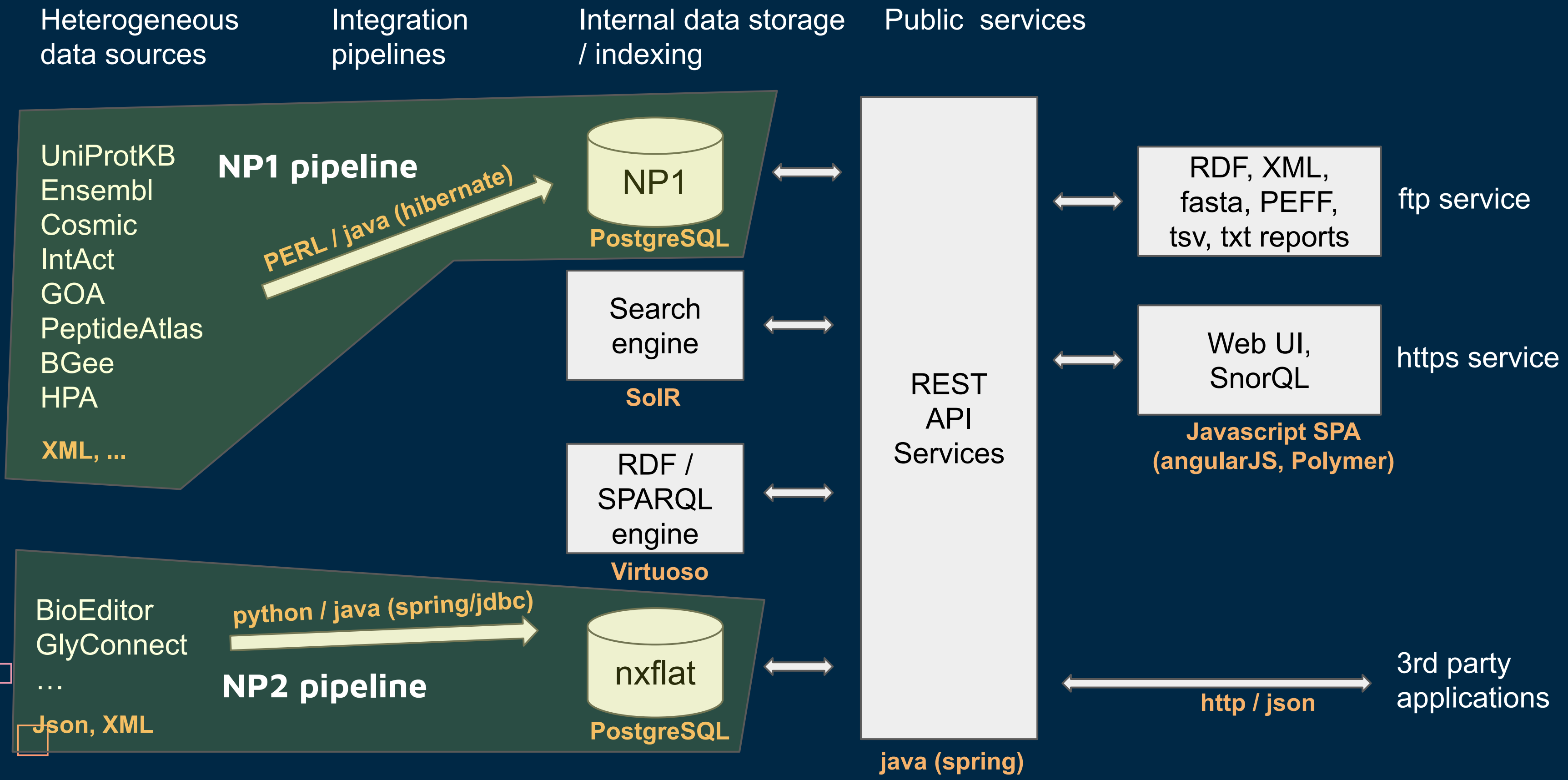
=> Less data releases / year



Website performance
:-)

Data integration performance
that’s the question !

Introduction – neXtProt integration architecture



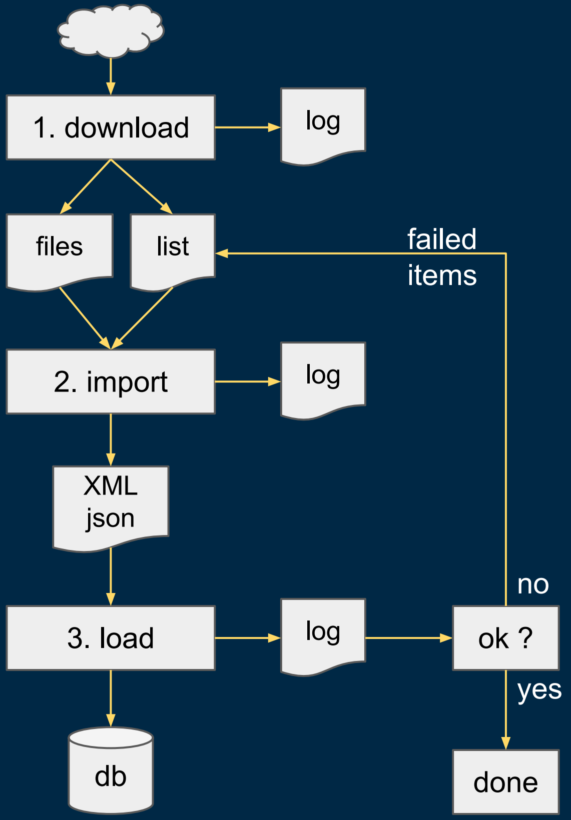
Introduction – integration pattern

For each datasource

1. Make a local copy on file system
   ant => bash, perl, python

2. Generate loadable file(s) using
   local rules, opt. parallelized
   ant => perl, scala, python

3. Load file(s) into database using
   global rules, opt. parallelized
   ant => java



Introduction – identified problems

Technical

- Access to file system on running multiple parallel processes
- High complexity of relational database
  - Many joins
  - Many integrity checks (unique keys, foreign keys)
  - Intricacy

Practical

- Manual checks required between 2 long processes
- …

Introduction – partial solutions considered

- Database schema simplifying
- Data partitioning, move to Oracle
- Ant, bash => Workflow management system
- Usage of multiple physical servers
- Cloud services ?

02

Container Technology

What is container technology

- Technology to package software components into an isolated runtime environment, where it can run its own

Container

- Main component of the containerization
- Encompasses an executable blob
- Runs on the container engine over the OS
- Resembles a virtual machine
- Lightweight and more separated from underlying hardware

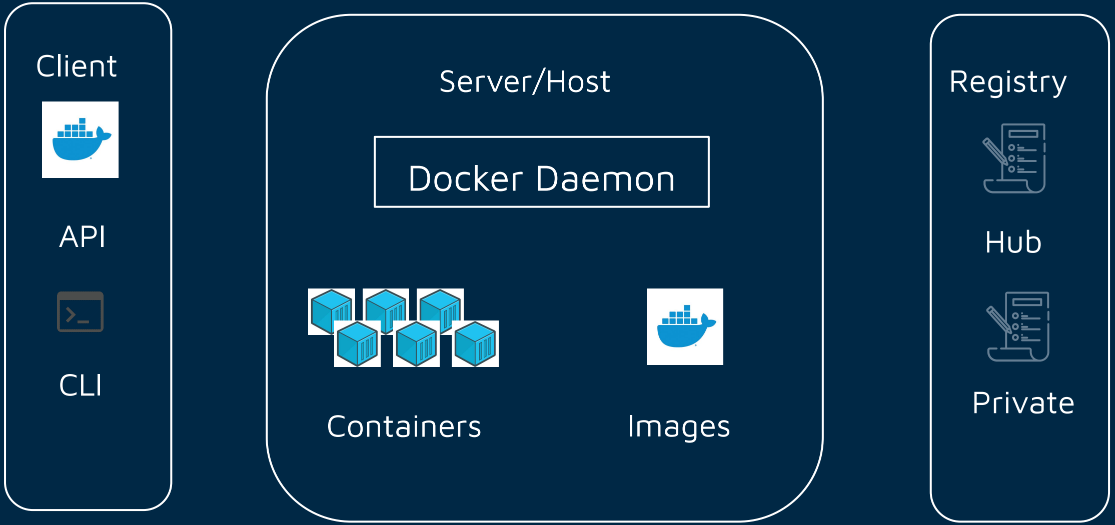
Use cases

- Isolated environments for different pieces of software component
- Many DevOps scenarios
    - Separate dev, test and prod environments
    - CI/CD related system compilation and building
- Deployment scenarios for fault tolerance and load balancing

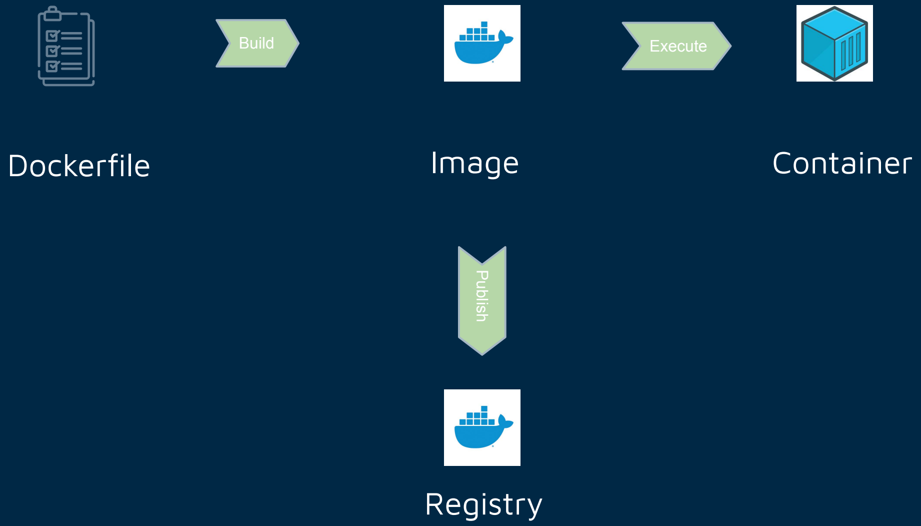
Docker

- Docker providers containerization
- Docker is a service which runs on the OS
- Docker service handles and interacts with hardware accordingly

Docker Architecture



Containerization process



Dockerfile

Build

Image

Execute

Container

Publish

Registry

02

Container Technology : Hands-On Session

Practicalities

- Hands-on sessions are done with the material at the git repo

  https://github.com/calipho-sib/dataintegration-tutorial

- Please clone this repository and follow the README files for ***s1.docker*** session

- Instructions are mostly for Linux based systems, so there can be problems with other OS s, we will try our best to help

Container Orchestration

- When multiple containers are running, management can be difficult
- Container orchestration handles the underlying complexity of managing multiple containers in a reactive mannar
- Guarantees properties such as load balancing, fault tolerance

03

Workflow Management with Airflow

Common Manual Workflow

- Consists of multiple processes, which run in sequence or in parallel
- Check logs for errors and failures
- Re-run them after manual intervention

Why workflow management?

- Could help automating/semi-automating a complex process
- To Streamline different steps of the process effectively (Schedule)
- Provides an easy way to achieve parallelism when required
- One place to oversee the whole operation

Why workflow management?

- Data integration processes are complex
- These processes have to be scheduled and executed properly
- Should be done in a resource and time efficient manner

Apache Airflow

- An open source framework
- Define workflows combining different processes
- Workflows are defined as Directed Acyclic Graphs (DAG)

Airflow DAGs

- DAG comprises of multiple processes
- A process is represented with an operator
- Operator is an abstraction for execution

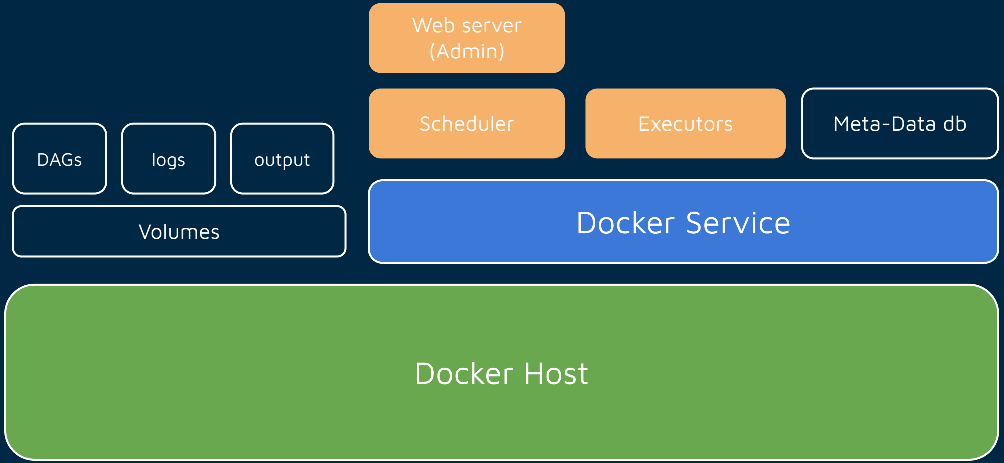
Airflow Operators

- Operators are abstract units which can be use to execute a process
- Command line operator, HTTP Operator and many more
- DAG can support both sequential and parallel execution patterns

Airflow Executors

- An executor is a process unit on which DAG is executed
- Sequential executor, Celery executor and others

Airflow Components in docker-compose Setup



03

Workflow Management with Airflow : Hands-on Session

Practicalities

- Hands-on sessions are done with the material at the git repo

  https://github.com/calipho-sib/dataintegration-tutorial

- Please clone this repository and follow the README files for ***s2.airflow*** session

- Instructions are mostly for Linux based systems, so there can be problems with other OSs, we will try our best to help

04

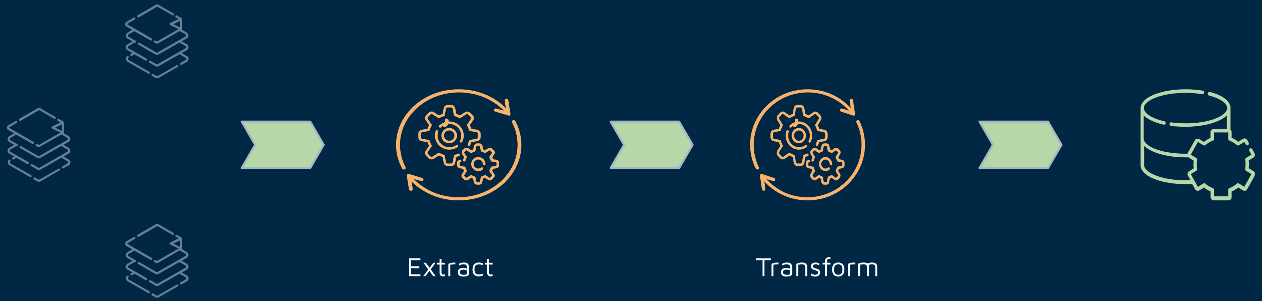
Containerized Data Integration Pipelines

Airflow Executors

- An executor is a process unit on which DAG is executed
- Sequential executor, Celery executor and others

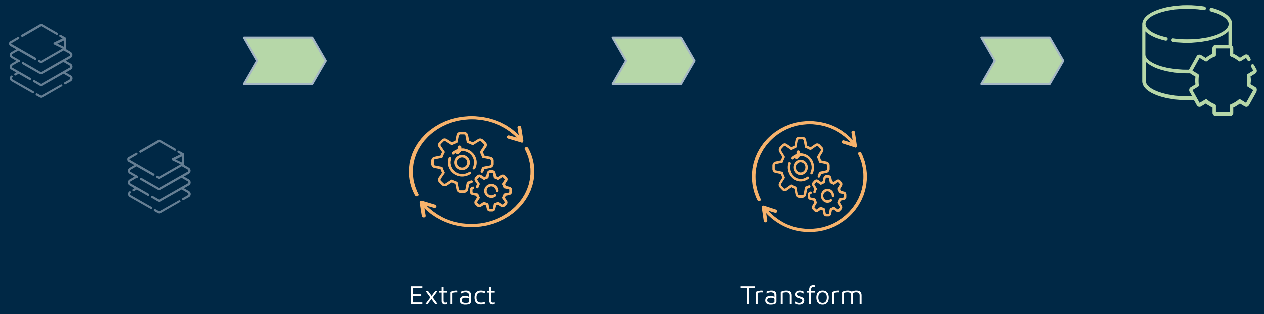
Data Integration Pipelines

- A complex workflow, a common pattern is Extract, Transform and Load
- Workflow processes are scheduled by Airflow scheduler



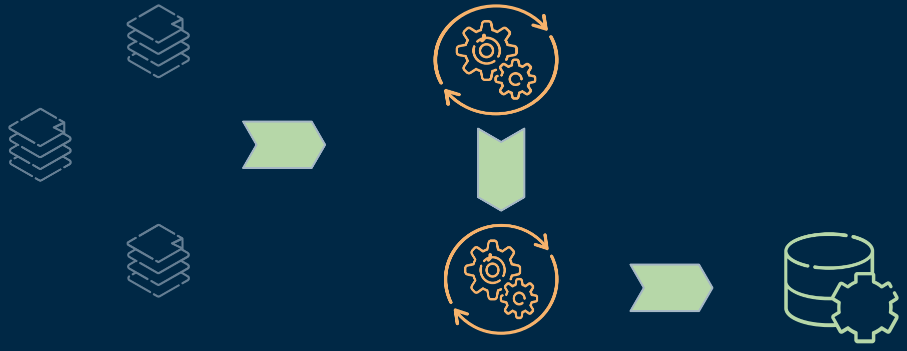
Data Integration Pipelines

- Some parts of the workflow could be processed in parallel
- Parallel components can be executed on containers achieving parallelism



Data Integration Patterns

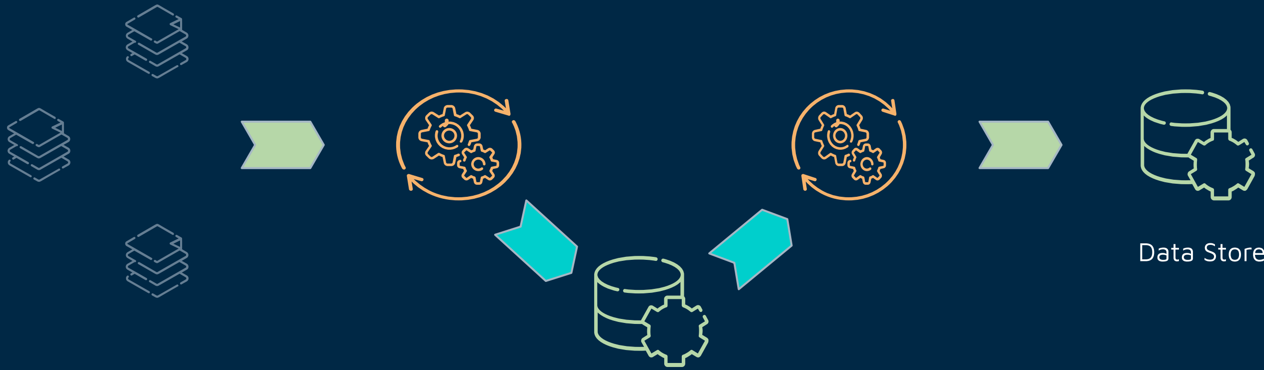
- Synchronous Communication Pattern



Data Store

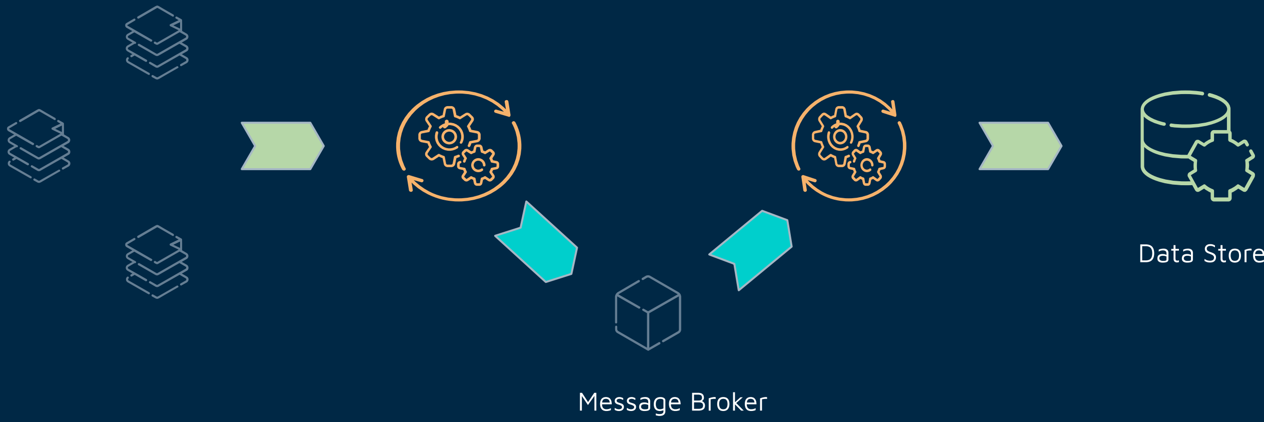
Data Integration Patterns

- Asynchronous Communication Pattern with I/O



Data Integration Patterns

- Asynchronous Communication Pattern with Message Passing



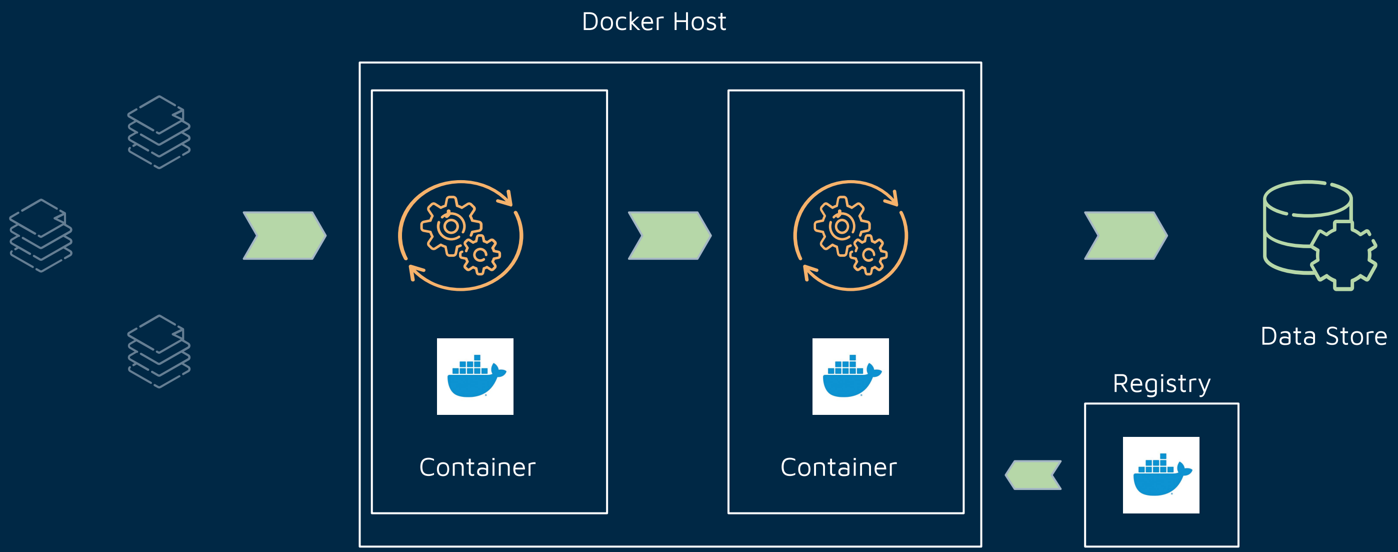
Advanced execution options in Airflow

- Celery is a work management system
- Celery resource has to be allocated in advance

Airflow Docker Operator

- Airflow has a Docker operator which can execute docker containers in the DAG
- Docker operator loads an image from a given docker registry
- Executes it on the docker host

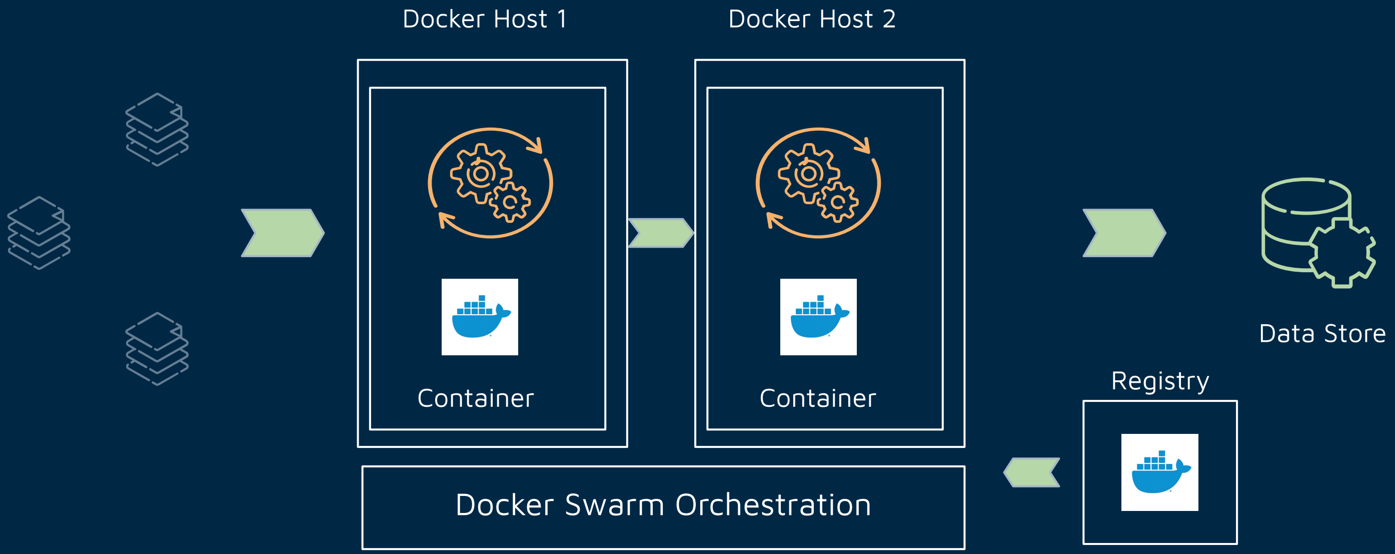
Airflow Docker Operator



Airflow Docker Swarm Operator

- Airflow supports to get the power of container orchestration
- Docker swarm operator executes containers over a docker swarm
- Hence it manages the underlying orchestration complexity
- Swarm provides fault tolerance, load balancing and other swarm features

Airflow Docker Swarm Operator



04

Containerized Data Integration Pipelines : Hands-on Session

Practicalities

- Hands-on sessions are done with the material at the git repo

  https://github.com/calipho-sib/dataintegration-tutorial

- Please clone this repository and follow the README files for ***s2.airflow*** session

- Instructions are mostly for Linux based systems, so there can be problems with other OSs, we will try our best to help

05

Q & A and Discussion

Few thoughts

- Any experiences on data integration systems?
- Any thoughts/experiences on deploying systems on external cloud services?
- External cloud services costs vs SIB resources?
- …..

Do you have any questions?

kasun.samarasinghe@sib.swiss
Pierre-Andre.Michel@sib.swiss

THANKS