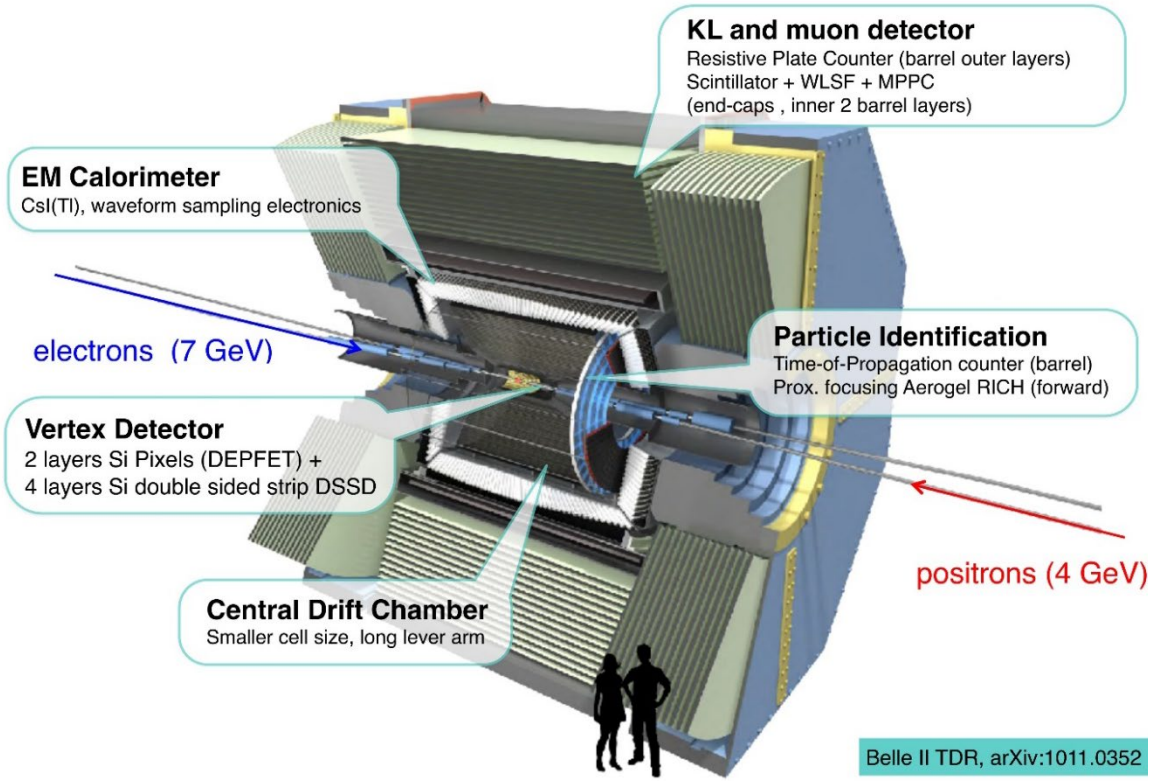
Automated Prompt Calibration for Belle II

Francis Pham*, David Dossett and Martin Sevior
University of Melbourne
For the Belle II Collaboration

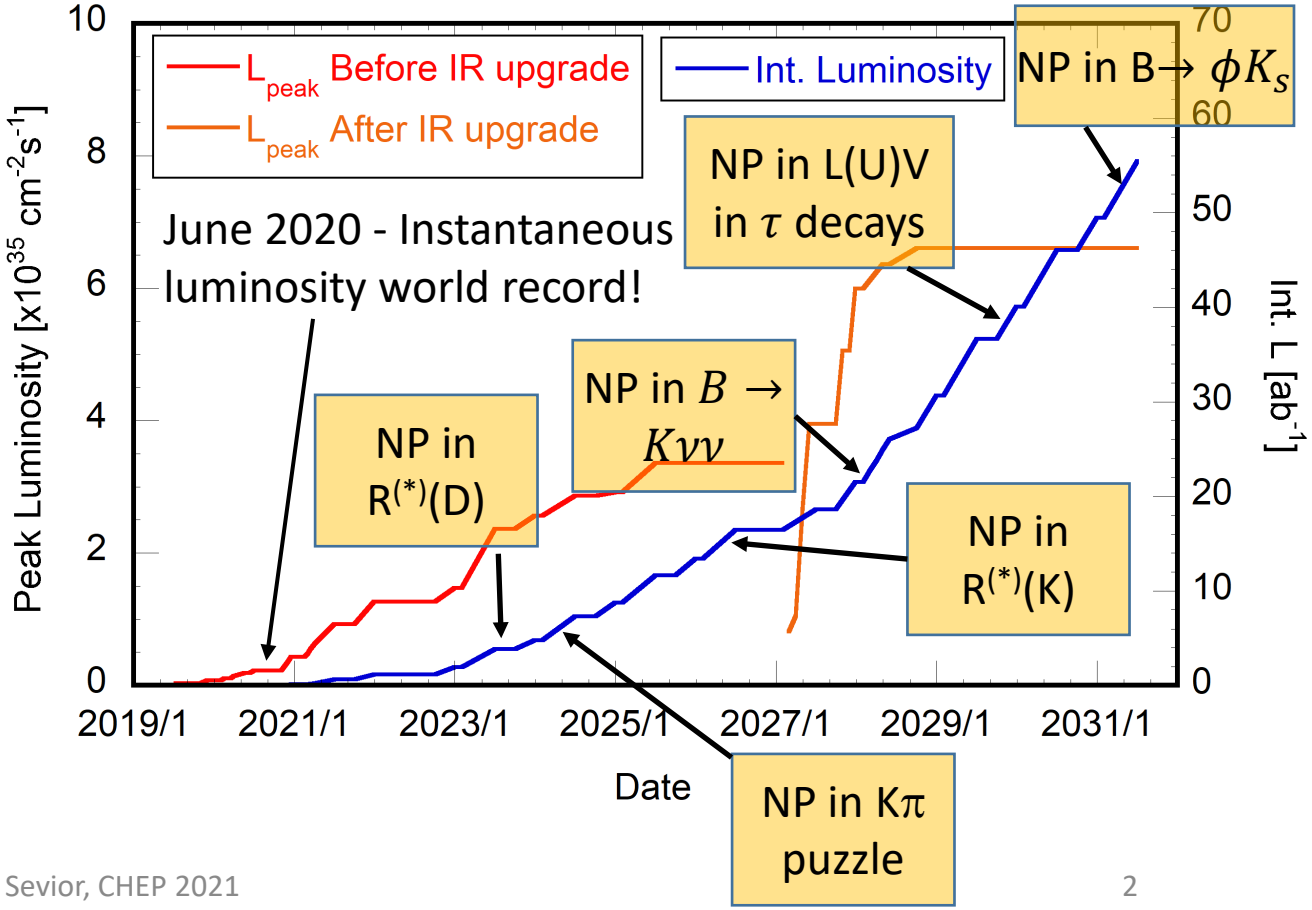
*Presenter

Belle II – Precision Flavour Physics

Belle II aims to collect 50 ab^{-1}
50 times more data than Belle!



Operate at 40 times higher peak luminosity
Sensitivity to New Physics



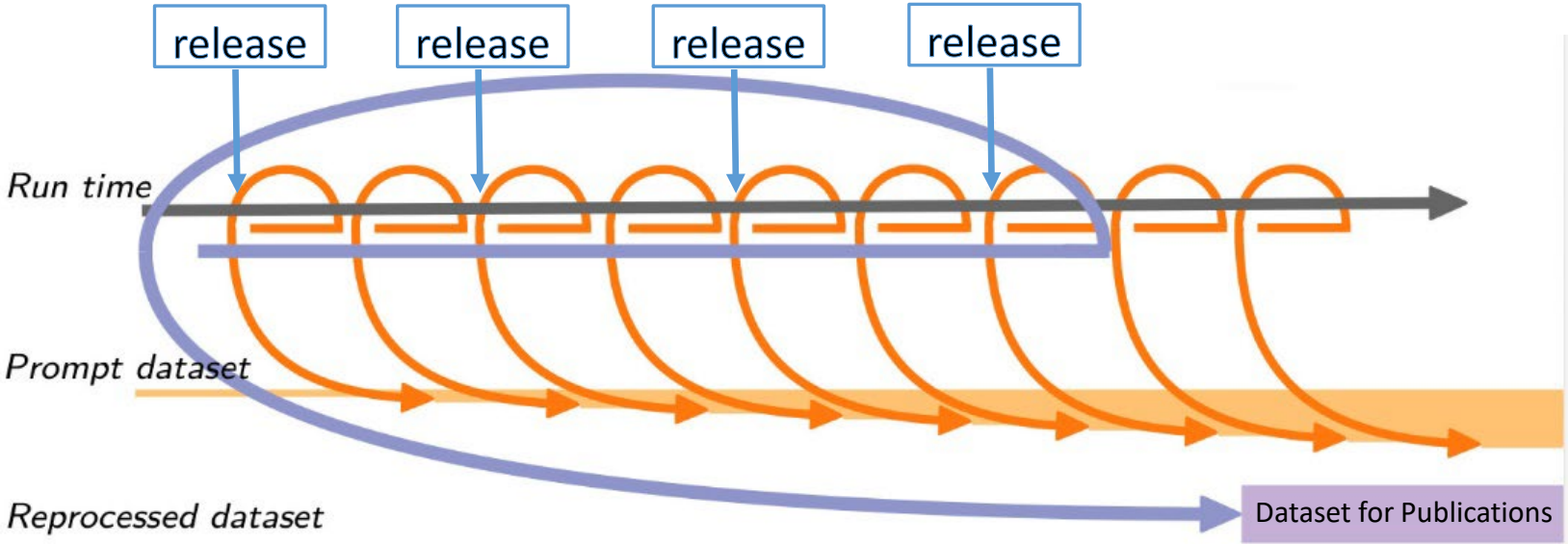
Belle II Calibration Strategy

Data is divided in intervals, “buckets”

- Approx. 9 fb^{-1}
- “Prompt” calibration
- Good for physics studies
- Runs at Brookhaven National Laboratory (BNL), America

“Reprocessing” for final physics

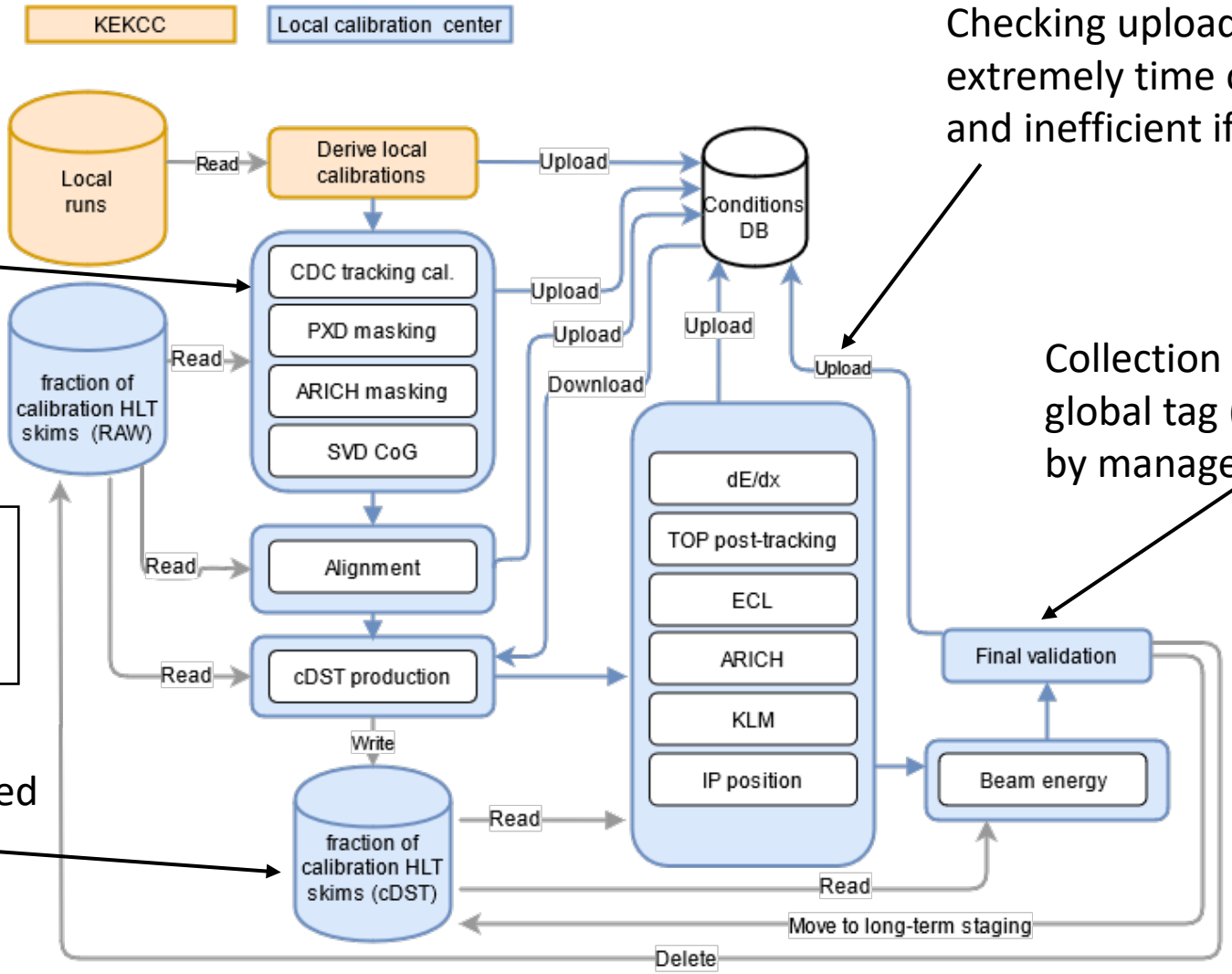
- Make a full recalibration
- Happens every 6-12 months
- Good for publication
- Runs at German Electron Synchrotron (DESY), Germany



Prompt Calibration Requirements

- High Level Trigger flag events passing various selections (hadron, muon-muon, etc) and this information can be used to produce calibration input files (skims).
- Per prompt bucket:
  - 7 calibration skims currently in used – 10 TB total storage
  - 2 fb^{-1} of each skim is required – fixed luminosity, so it will become a smaller fraction of total data set over time
  - Recreate calibration skims with tracking/alignment (cDST) – 13 TB
  - Calibration constant outputs (payloads) – 350 GB
  - Minimum 30 kHS06 (3k cores) for smooth prompt calibration
- Rucio used to automatically replicate calibration skims from KEK to BNL

Calibration Loop



Checking uploads are valid is extremely time consuming and inefficient if done manually

Calibrations have complex dependencies

Collection of payloads is called a global tag (GT) and must be verified by manager before final sign-off

Manager uses JIRA, a task management system, to inform experts to start their calibration

cDSTs need to be replicated for long term storage

Apache Airflow

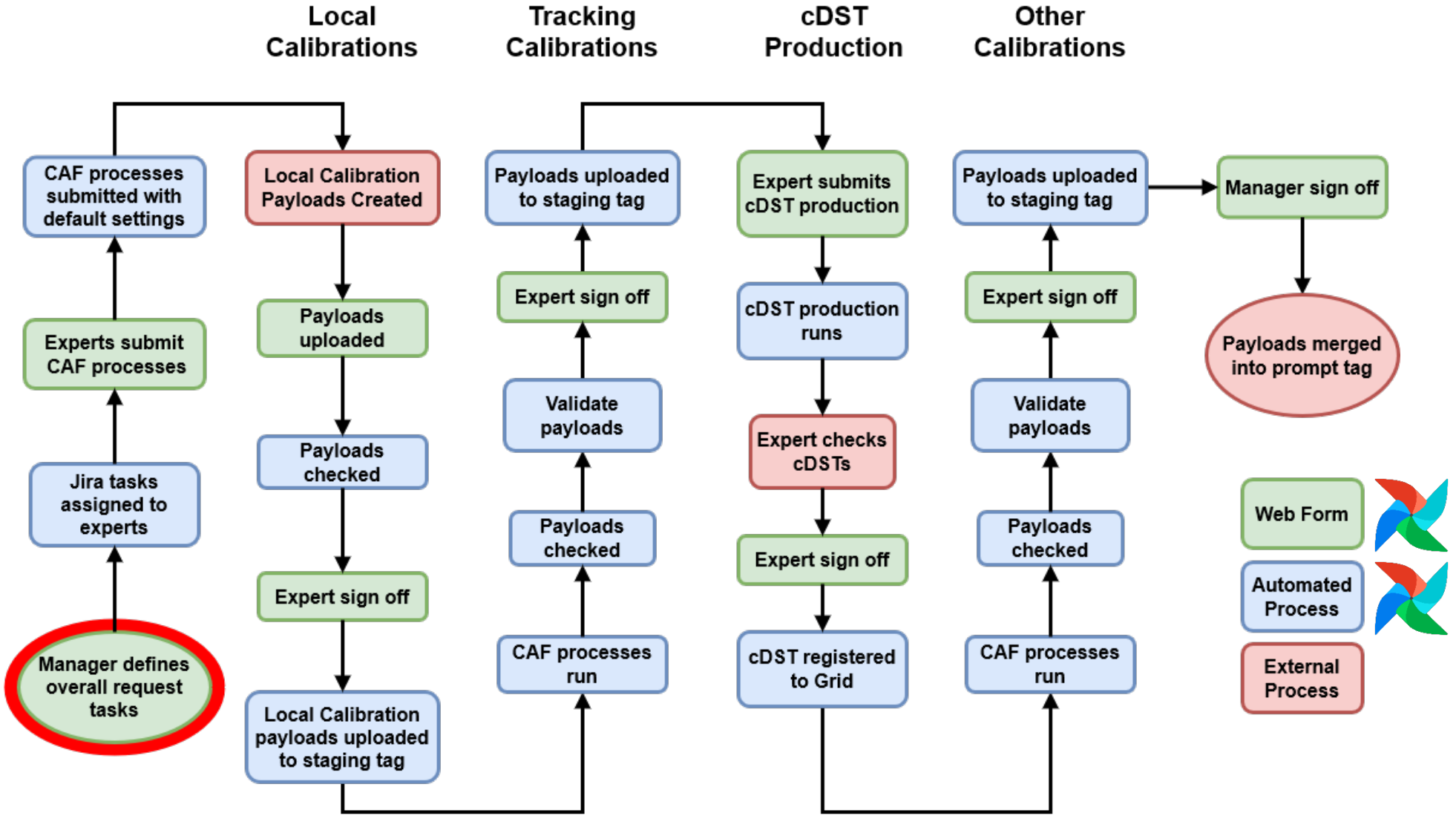
- Directed Acyclic Graph (DAG) workflow manager - it helps to automate scripts in order to perform tasks.
- Originally developed by Airbnb and released as open source Python 3

Comes with:

- Cron-like scheduler
- Plenty of task types if we need them (Bash, Python, SSH, SQL, …)
- Flask based admin web monitoring for tasks and database contents
- Easy plugin system for extension of website and core classes, e.g. LDAP authentication
- Active community for questions/extensions/new features

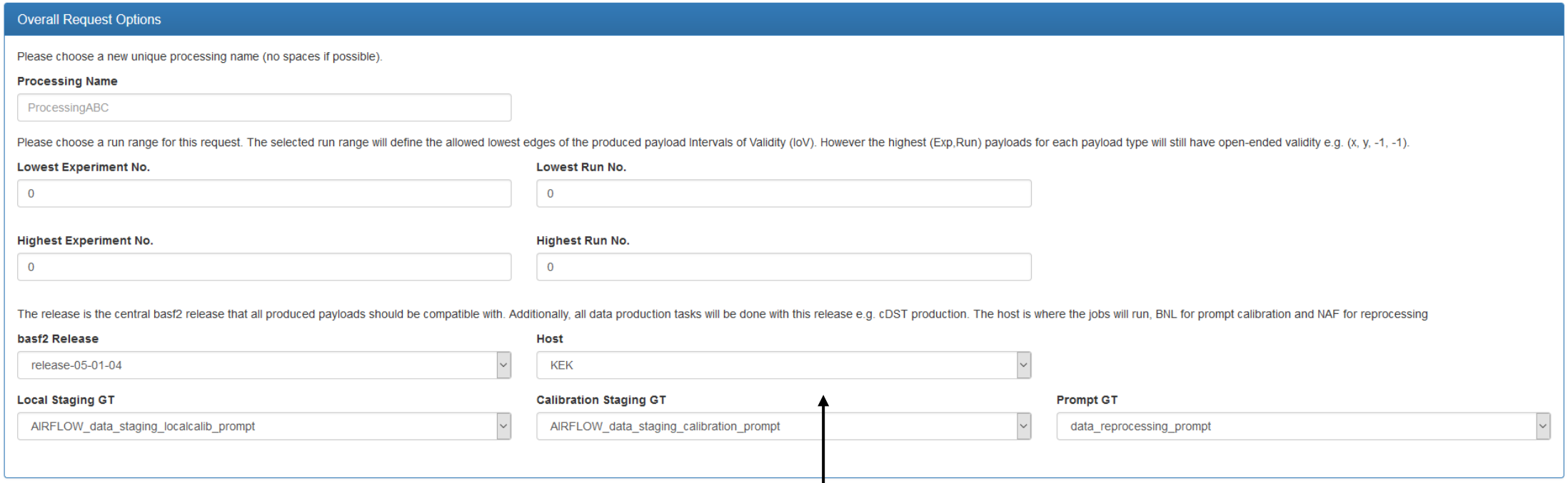
https://doi.org/10.1051/epjconf/202024502016

Automated Calibration



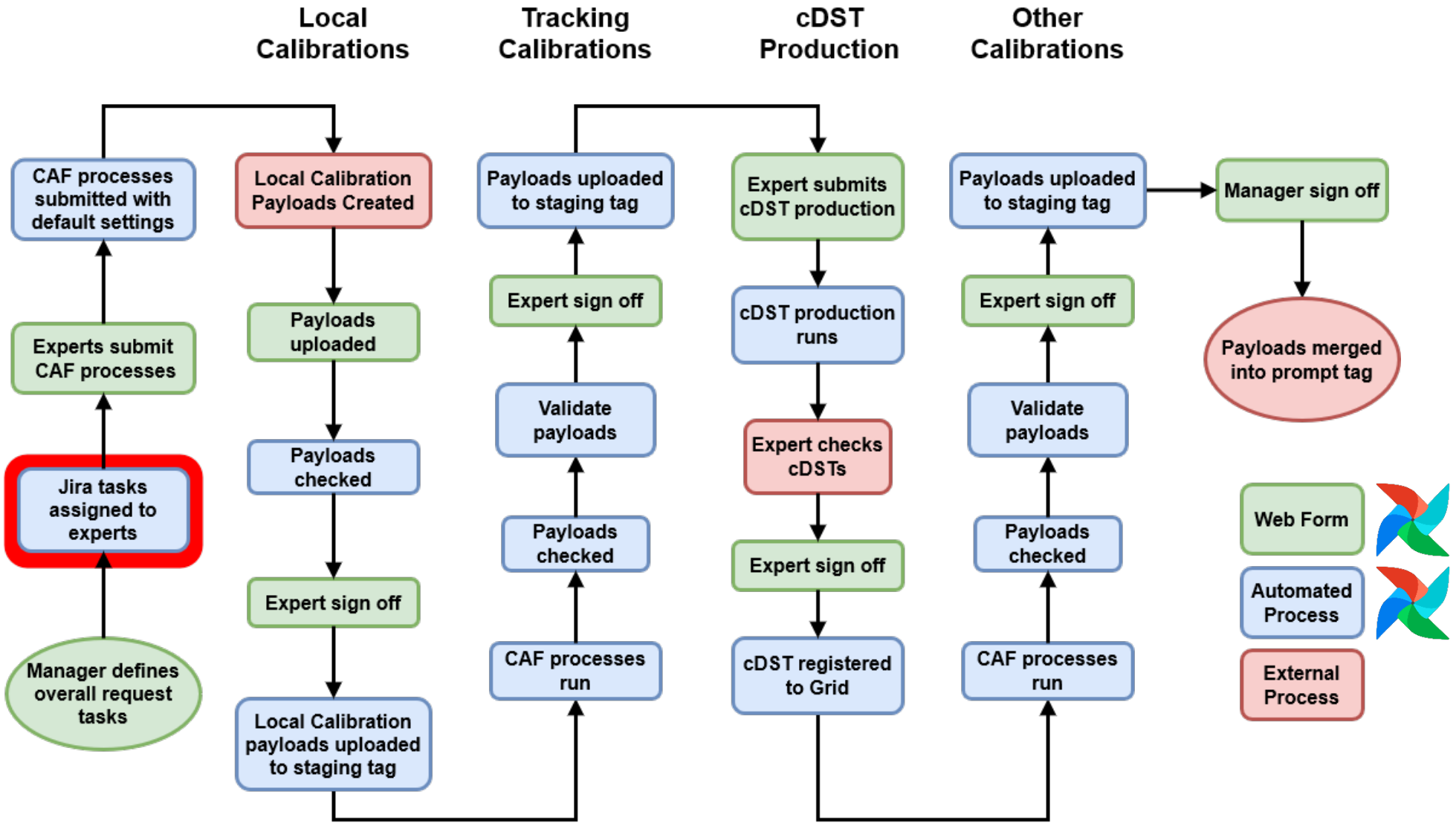
Starting Processing Request with Airflow

- The time-consuming task of creating dozens of JIRA tickets and setting their dependencies reduces to:



Multiple processing requests can run simultaneously
e.g. prompt at BNL while reprocessing at DESY

Automated Calibration



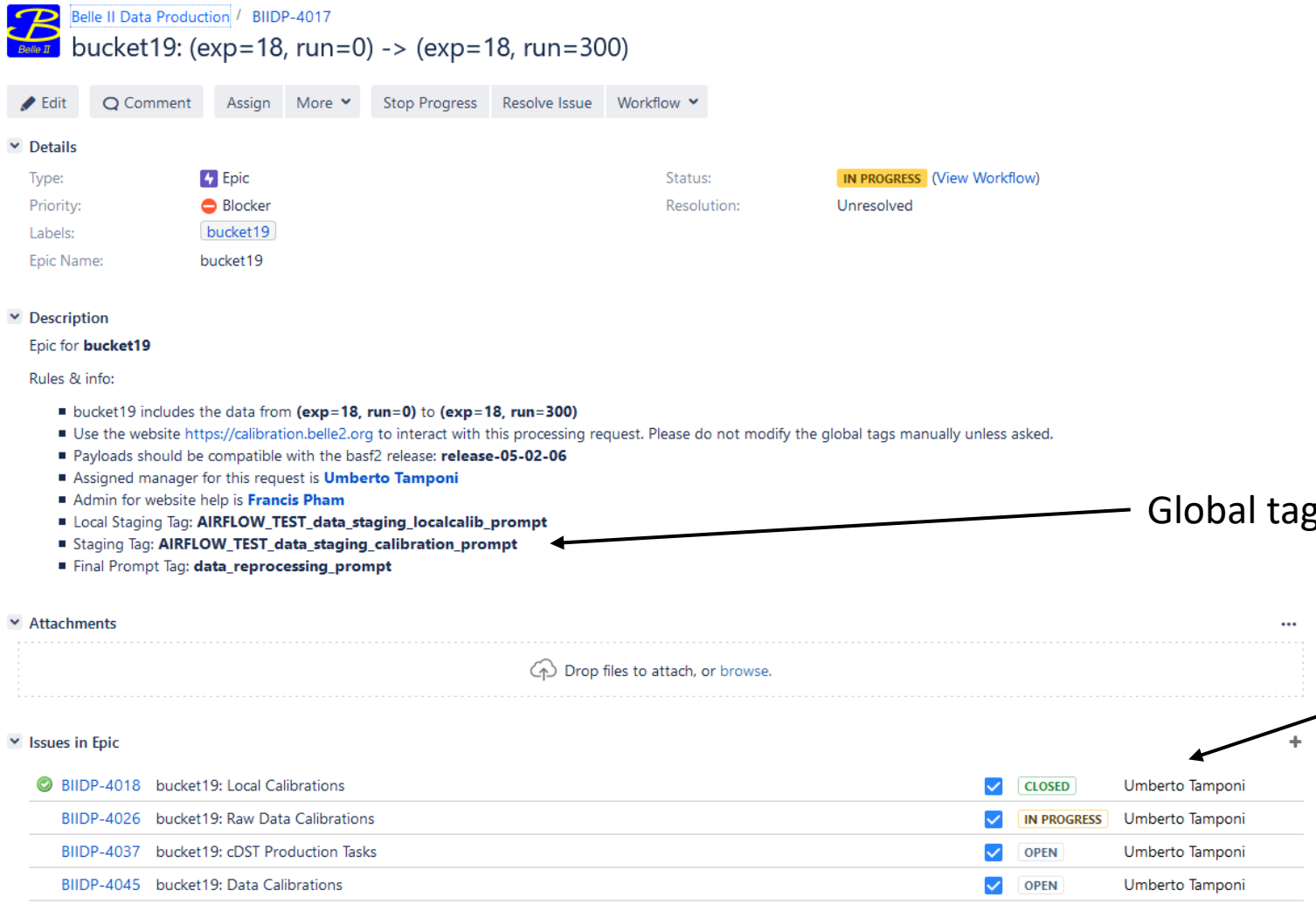
JIRA Task automatically assigned

Descriptions automatically filled

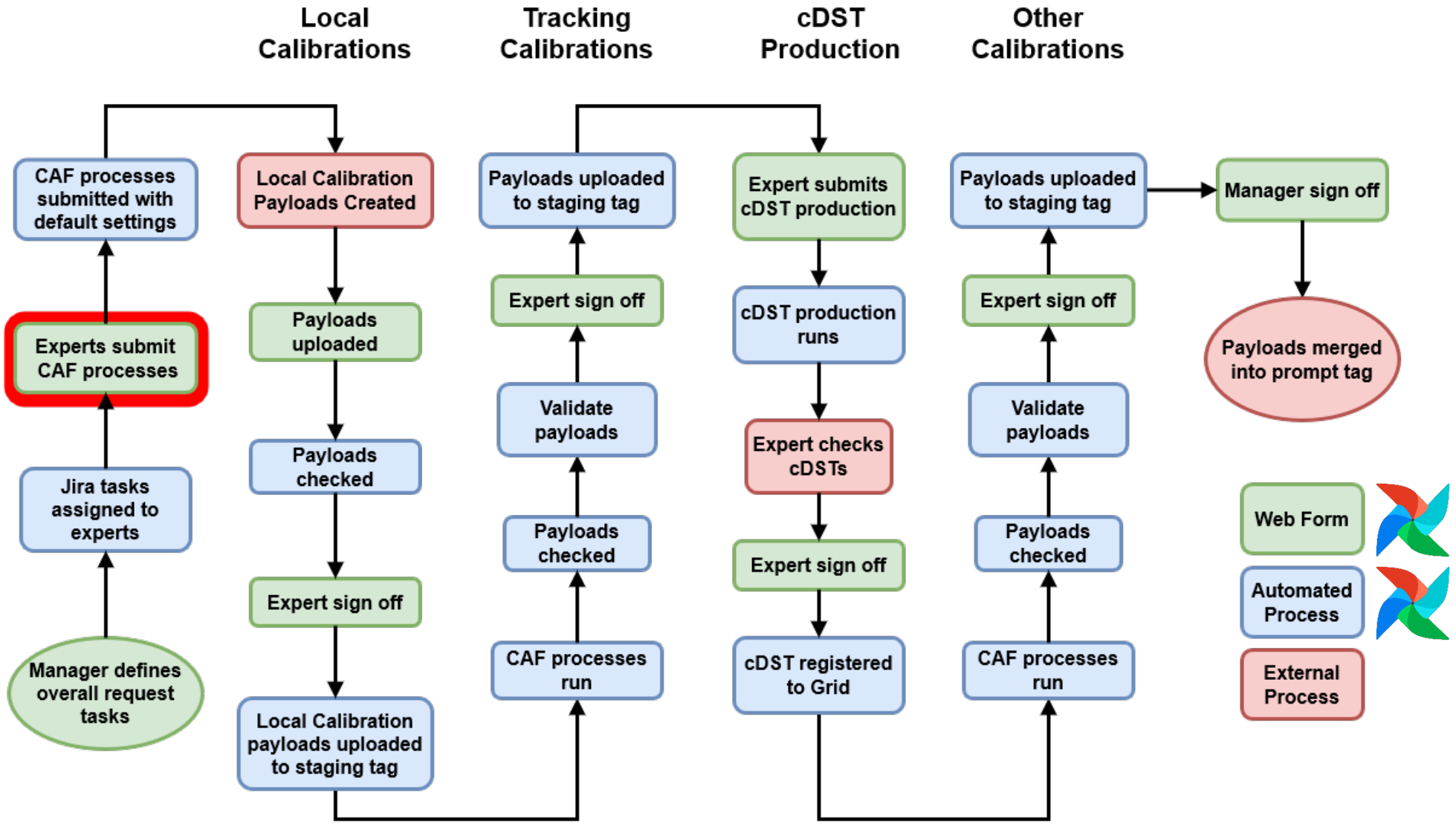
Global tags emptied

Dozens of JIRA tickets Created with dependencies included

Correct expert assigned

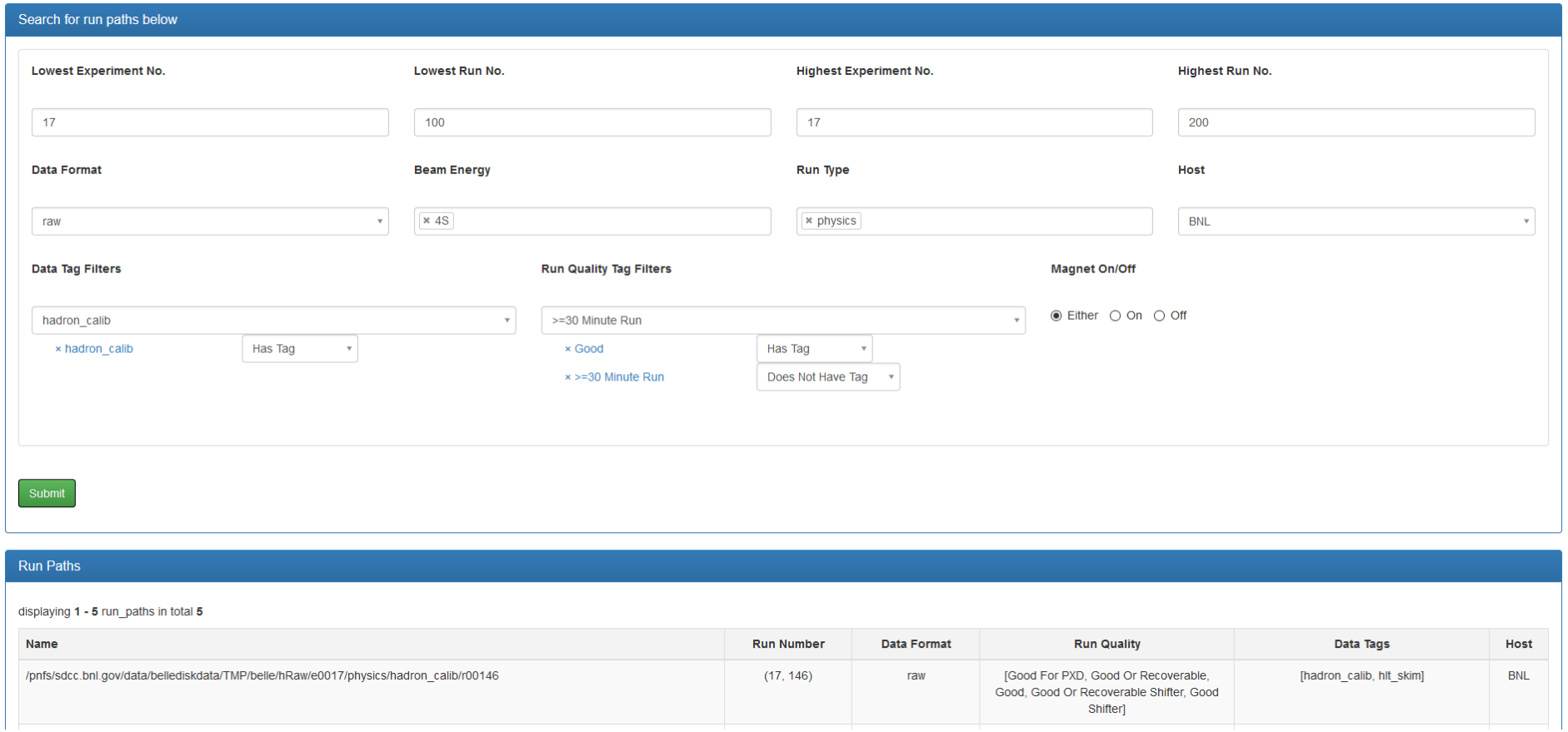


Automated Calibration

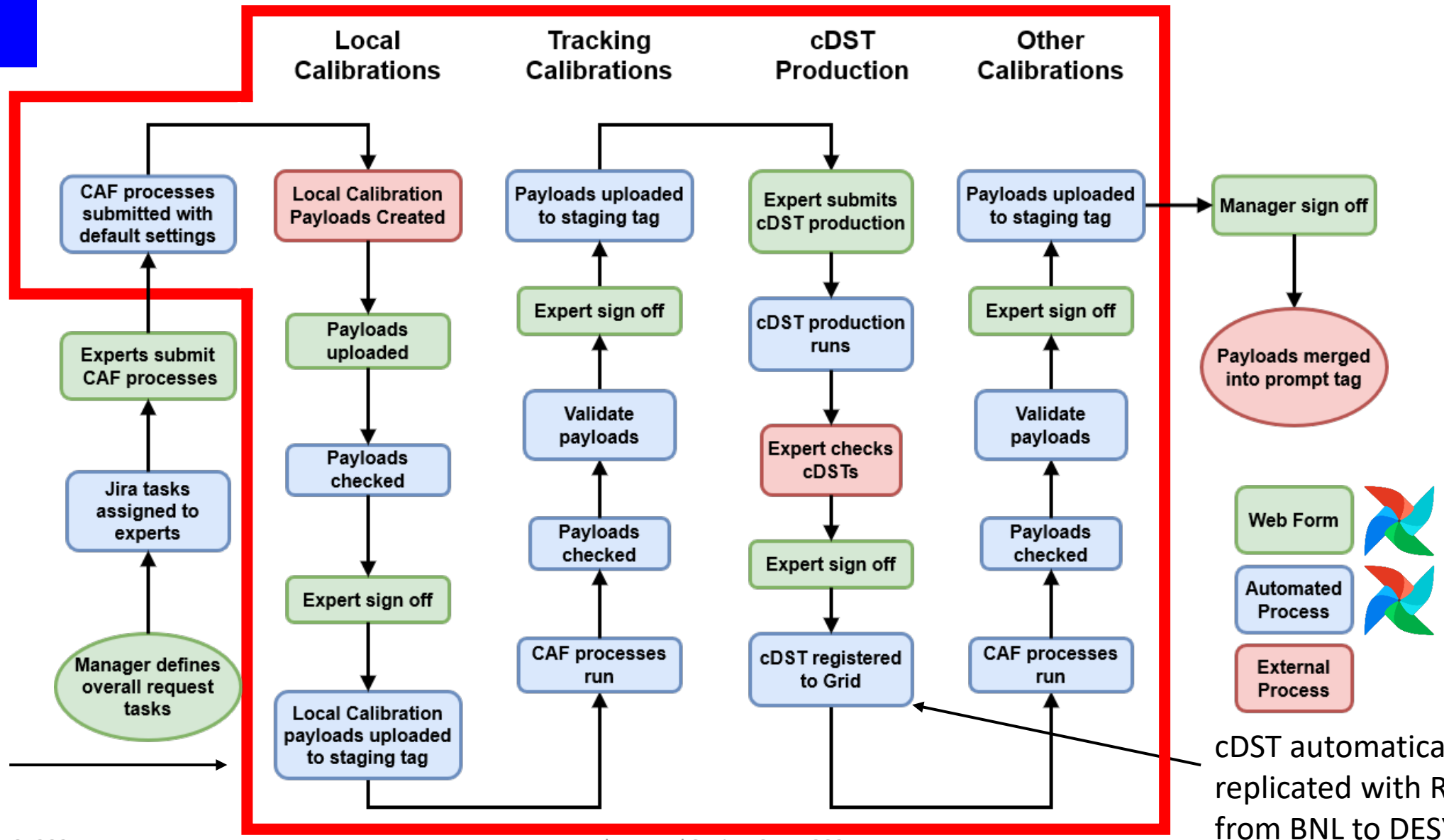


Starting a Calibration with Airflow

- The tedious and often error prone task of creating data input files for a calibration reduces to:



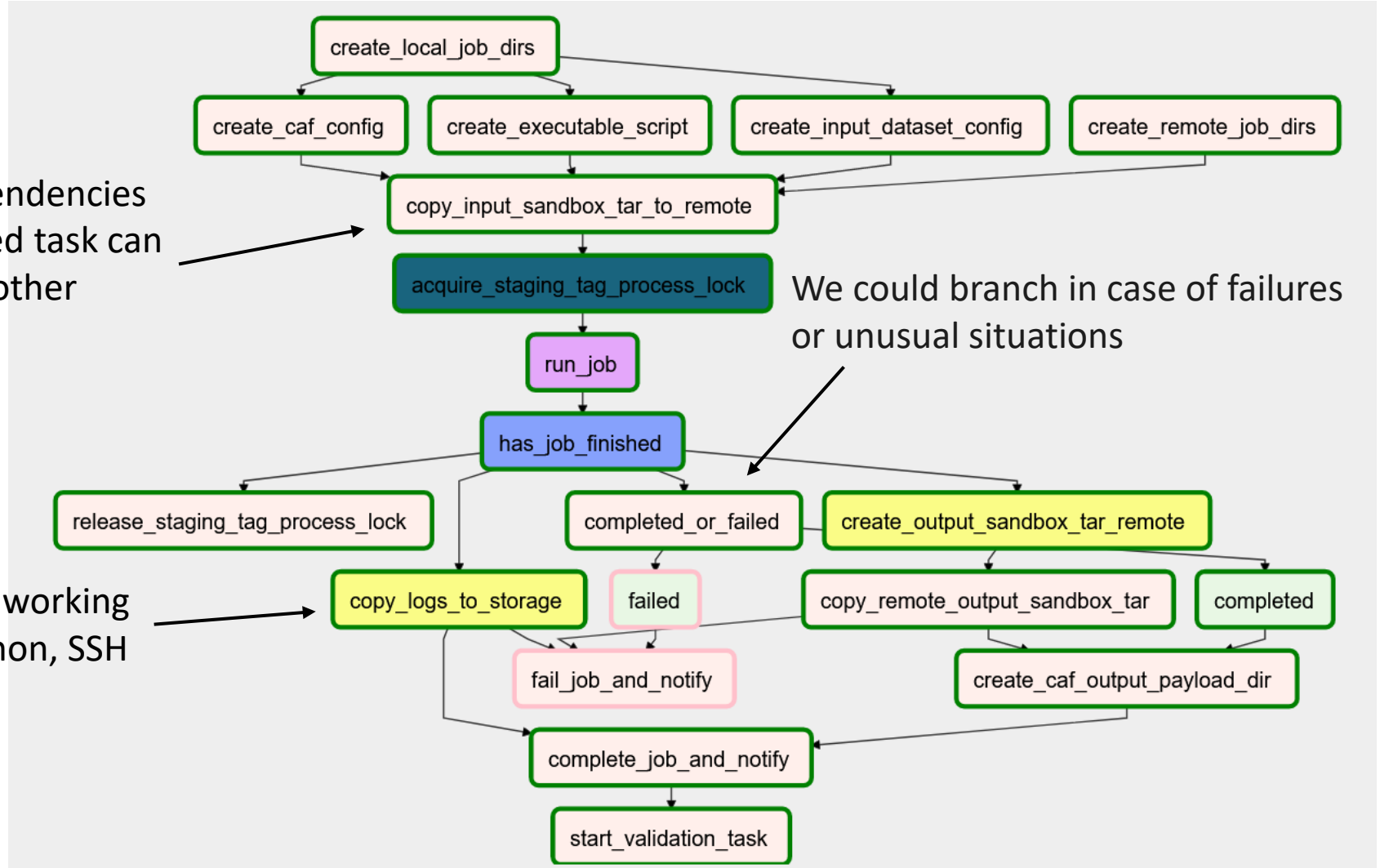
Automated Calibration



Remaining steps automated with DAGs!

cDST automatically replicated with Rucio from BNL to DESY

A basic DAG example: Job submission



Won't run all dependencies fulfilled – one failed task can be set to block another from starting

We could branch in case of failures or unusual situations

Different task types working together: Bash, Python, SSH

Pham and Sevior, CHEP 2021

Prompt Experience with Airflow

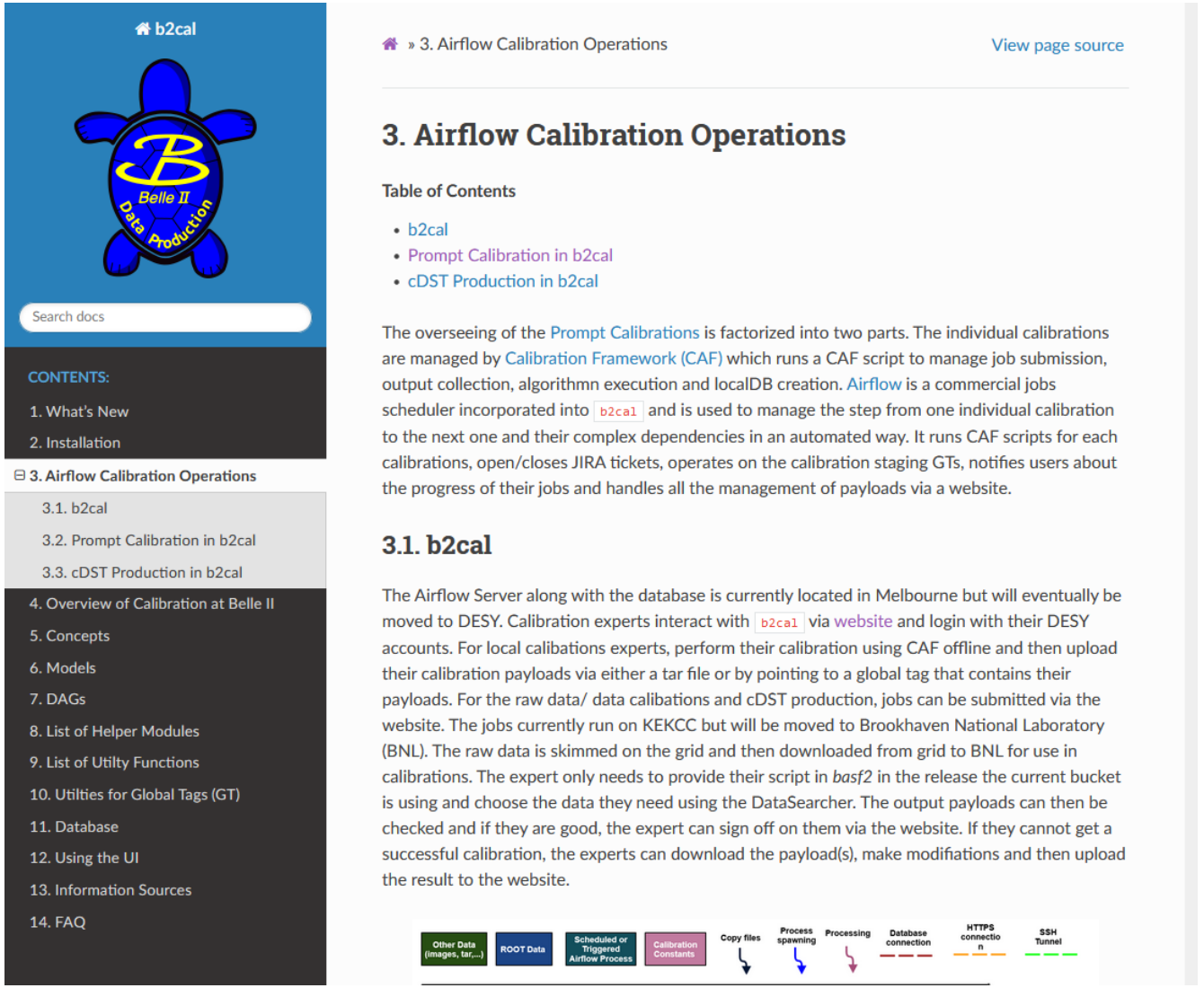
- Time taken to perform prompt calibration at BNL continues to decrease as we gain more experience
- Signing off is largest down-time, to be automated in current bucket

Prompt at BNL	Time for bucket (days)	Time from data taken to calibrated (days)
Bucket #1 (10.2 fb^{-1})	43	44
Bucket #2 (10.6 fb^{-1})	24	28
Bucket #3 (11.3 fb^{-1})	21	35

About 1 month from data taken to calibrated!

Documentation

- Easy to self document most of the code with Sphinx
- Host the static docs alongside the Airflow web-server



Conclusion

- Airflow is immensely flexible and easy to code
- Calibration at Belle II using Airflow to automate task such as JIRA ticket creation, job submission, cDST registration, etc.
- Use of webforms greatly simplify the process of data selection, job monitoring and processing request initialisation
- Automated calibration successfully implemented at KEK, NAF and BNL
- Time taken for each calibration has decreased as a result of automation and will continue to decrease as more tasks become automated
- Quickly approaching zero down-time due to manual intervention, physics data will becomes available as quickly as computing resources will allow!


Belle II

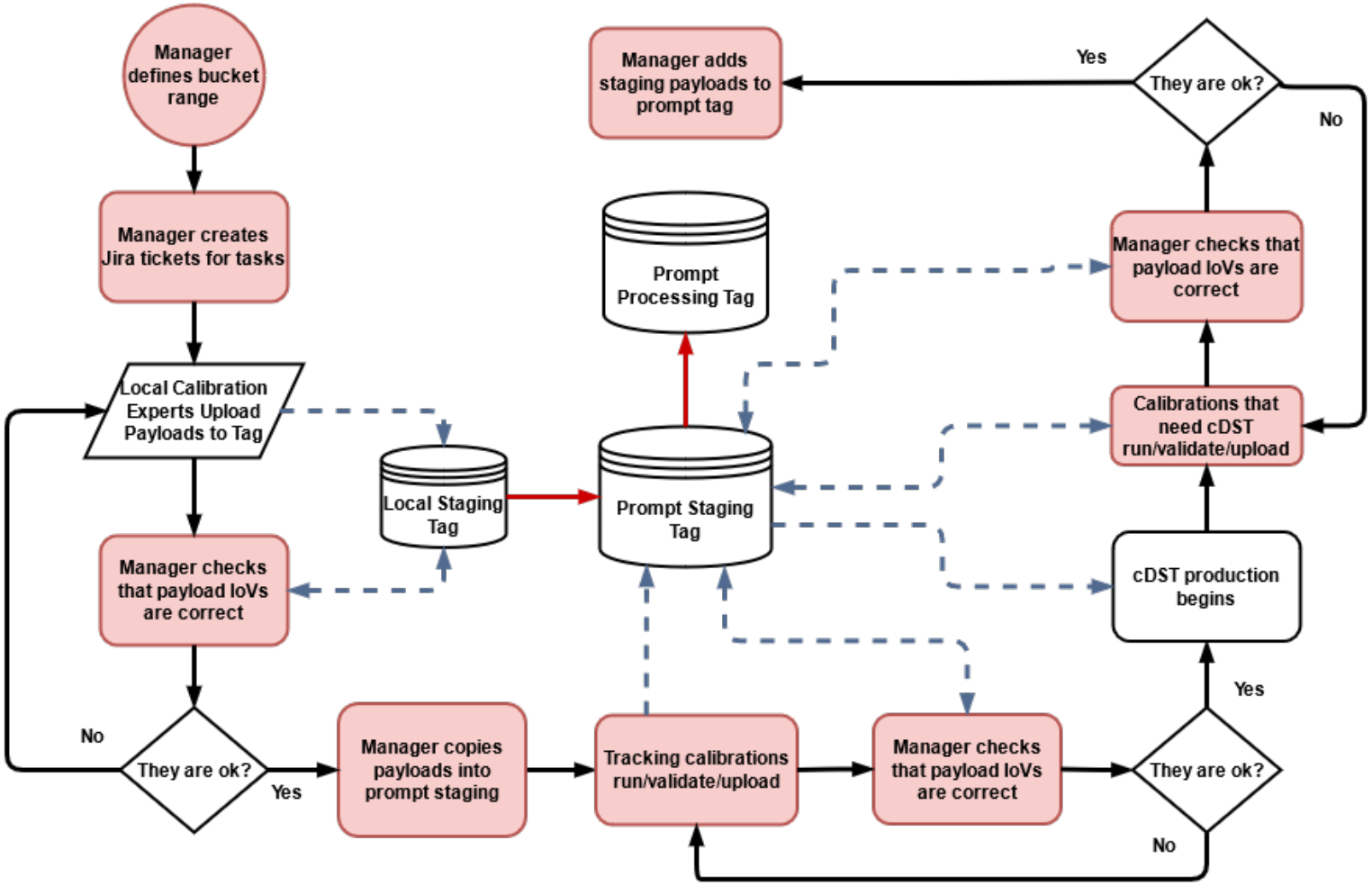

POSTERA CRESCAM LAUDE
THE UNIVERSITY OF
MELBOURNE

Backup

Belle II Jargon

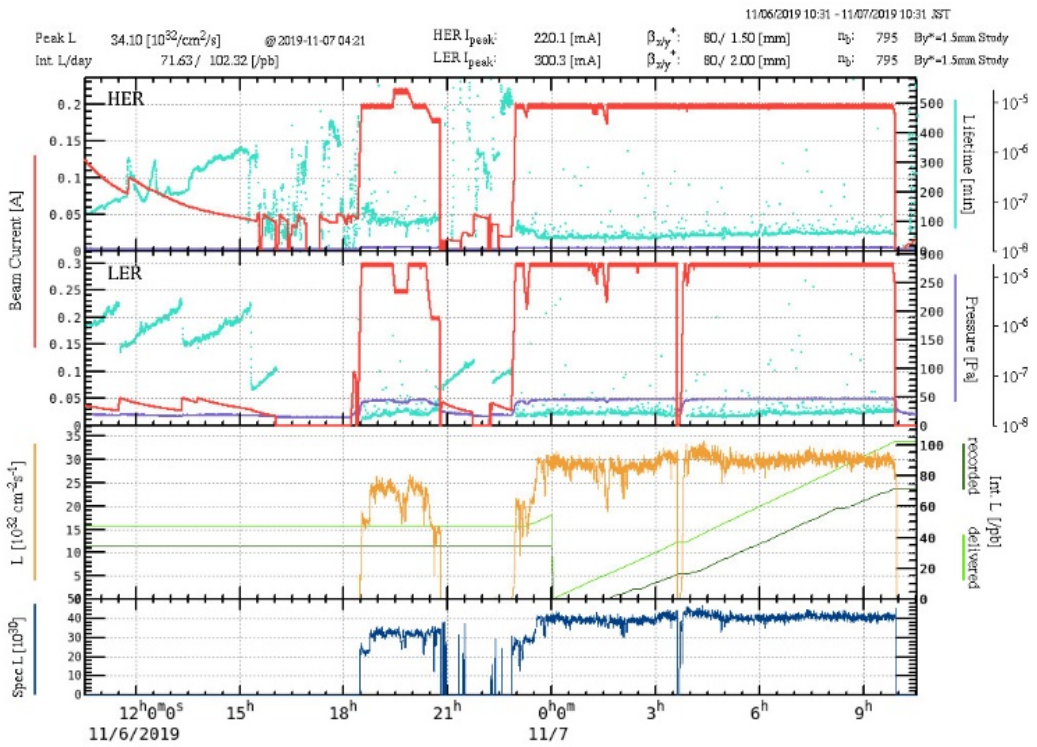
- mDST – basic file format for analysis (after reconstruction)
- cDST – Calibration data format, available after initial tracking alignment
- JIRA – task or issue tracking system
- Run – an amount of time with very similar detector conditions ~ 1-8 hours
- Experiment – a longer period of time of operation with similar detector configuration ~ 1-6 months
- Global Tag – String that defines a collection of database payloads in the basf2 conditions database
- Payload – a file containing data in the conditions database
- Interval of Validity (IoV) – a run range defining a payload's validity e.g. (exp_low=1, run_low=1, exp_high=1, run_high=10)
- Calibration and Alignment Framework (CAF) – basf2 framework for producing payloads from input data
- Prompt calibration – best first attempt at calibrations that can be done before mDST production
- Prompt processing – the first mDST production for user analysis

Manual Calibration

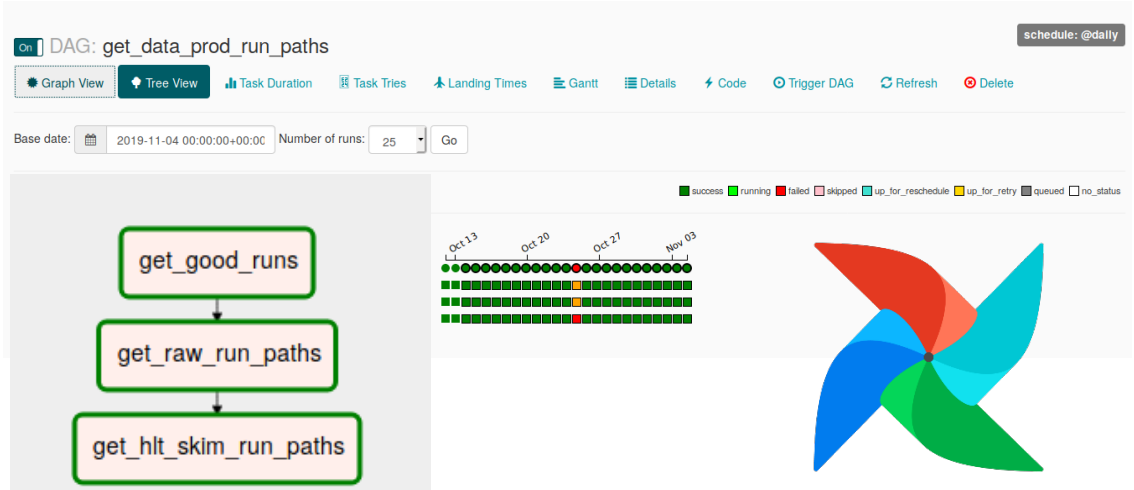


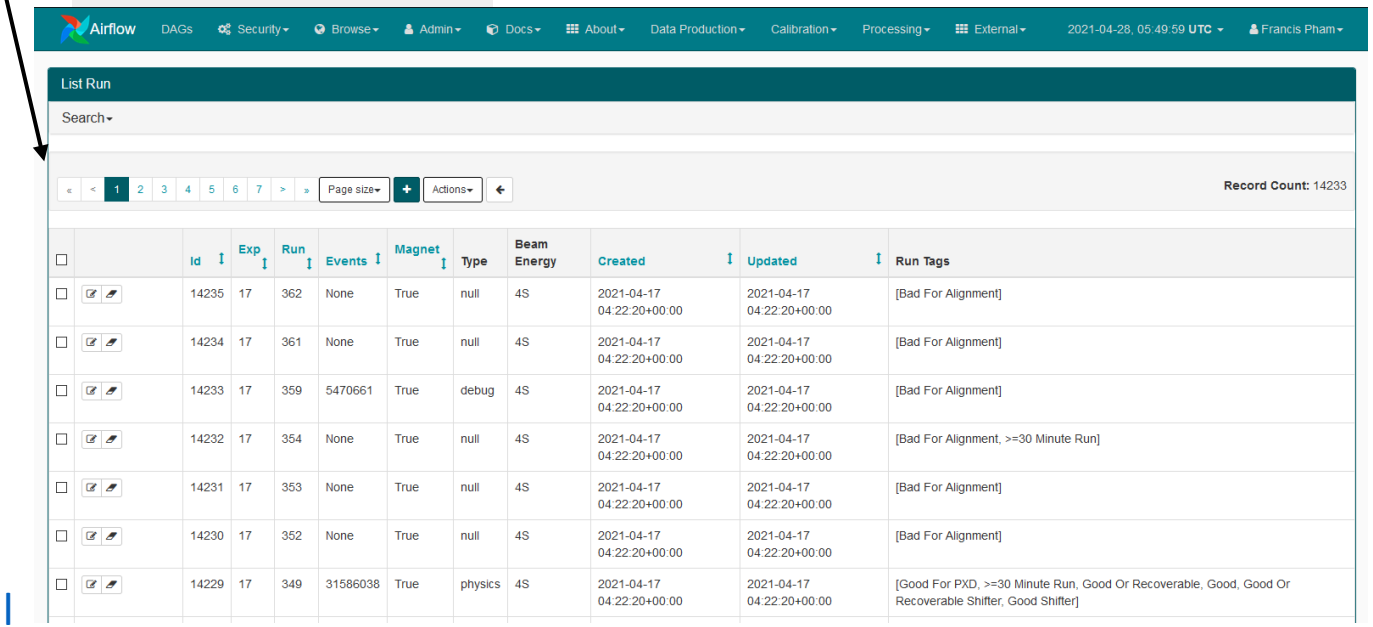
Automatic Run Collection

Run information collected and updated daily, flags runs at ‘Good’, ‘Bad’, etc. based on runDB



http://www-linac.kek.jp/skekb/snapshot/dailysnap.html



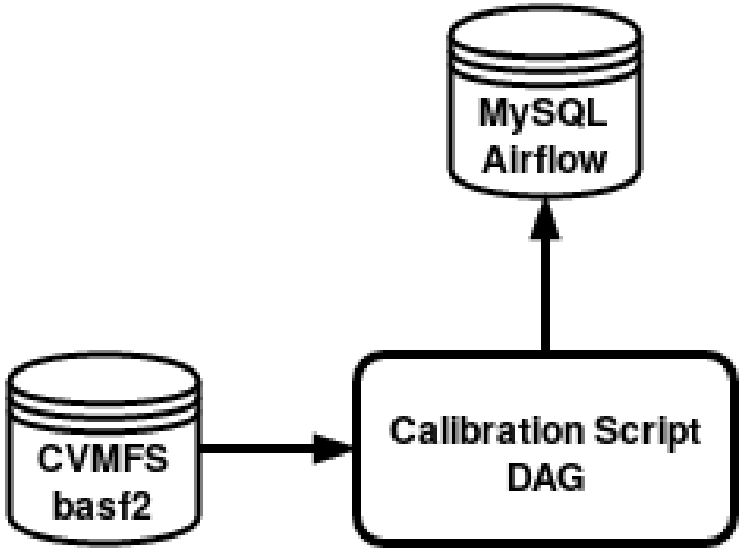


	Id	Exp	Run	Events	Magnet	Type	Beam Energy	Created	Updated	Run Tags
	14235	17	362	None	True	null	4S	2021-04-17 04:22:20+00:00	2021-04-17 04:22:20+00:00	[Bad For Alignment]
	14234	17	361	None	True	null	4S	2021-04-17 04:22:20+00:00	2021-04-17 04:22:20+00:00	[Bad For Alignment]
	14233	17	359	5470661	True	debug	4S	2021-04-17 04:22:20+00:00	2021-04-17 04:22:20+00:00	[Bad For Alignment]
	14232	17	354	None	True	null	4S	2021-04-17 04:22:20+00:00	2021-04-17 04:22:20+00:00	[Bad For Alignment, >=30 Minute Run]
	14231	17	353	None	True	null	4S	2021-04-17 04:22:20+00:00	2021-04-17 04:22:20+00:00	[Bad For Alignment]
	14230	17	352	None	True	null	4S	2021-04-17 04:22:20+00:00	2021-04-17 04:22:20+00:00	[Bad For Alignment]
	14229	17	349	31586038	True	physics	4S	2021-04-17 04:22:20+00:00	2021-04-17 04:22:20+00:00	[Good For PXD, >=30 Minute Run, Good Or Recoverable, Good, Good Or Recoverable Shifter, Good Shifter]

Collecting basf2 versions

- **Automatically find available prompt calibrations via basf2**

- Defining prompt calibration scripts in basf2 allows us to enforce release + payload compatibility
- Available scripts, contact details, input data types, default input data filters, … can be automatically assigned in Airflow's database
- They are presented as options when defining the processing request



Current (manual) prompt calibration

- Run range of new prompt processing declared
- Jira used to make an ordered/blocking set of assigned tasks
- Experts produce payloads and upload to a “staging” global tag
- cDST production manually run when tracking calibrations finished
- Once all Jira tasks are marked as completed the contents of the staging tag are merged with the main prompt processing tag
- mDST production can begin
- See poster by Marco Milesi “Calibration and prompt processing of data for the Belle II experiment” - https://indico.cern.ch/event/773049/contributions/3474791/

Core Airflow Concepts

Hook: A Python class for managing a connection to an external service e.g. JiraHook

Operator: A Python class that defines a task type to be executed e.g. BashOperator

Sensor: A special Operator type that re-schedules occasionally while waiting for a value to be True

DAG: A dynamic collection of operators that have dependencies between them

Jinja2 template: A string which contains templates to be automatically replaced by the Airflow environment when the operator executes e.g.

”{{ conf['webserver']['webserver_port'] }}” → “8080”

Apache Airflow

https://airflow.apache.org/

Flask framework

https://flask.palletsprojects.com/en/1.1.x/

Jinja2

https://jinja.palletsprojects.com/en/2.10.x/

SQLAlchemy

https://docs.sqlalchemy.org/en/13/