Development of a flexible, multi-stage data pipeline

… for enhanced automation, quality control and observability

Christian Werner, Christof Lorenz
IMK-IFU, Campus Alpin, Karlsruhe Insitute of Technology
Contact: ✉️ christian.werner@kit.edu, 🐦 @cwerner76

Data Science Symposium No. 7 | Hereon, Geesthacht | 2020-06-27

Karlsruhe Institute of Technology

Current Status

… a patchwork of processes and workflows

- Heterogenuous data processing landscape
- No centralized control/ interface
- No/ limited data processing version control
- No/ limited data QA/ QC
- No centralized data catalog

🤨

Objectives

… there has to be a better way

- **Consolidate**
  Centralise services, jobs and scripts into a manageable system
- **Automate**
  Create data transformation pipelines that take care of data ingestion, auto-validation and transformation (data repositories)
- **Increase Consistency**
  A global scheduling ensures that processes run at defined times/ intervals or conditions
- **Add Scalability**
  The system should scale from few jobs to actual model processing (i.e. data generation for dashboards etc.)
- **Add Observability**
  System status, failure and progress should be easily observable, data owners be notified if things break or are out-of-norm

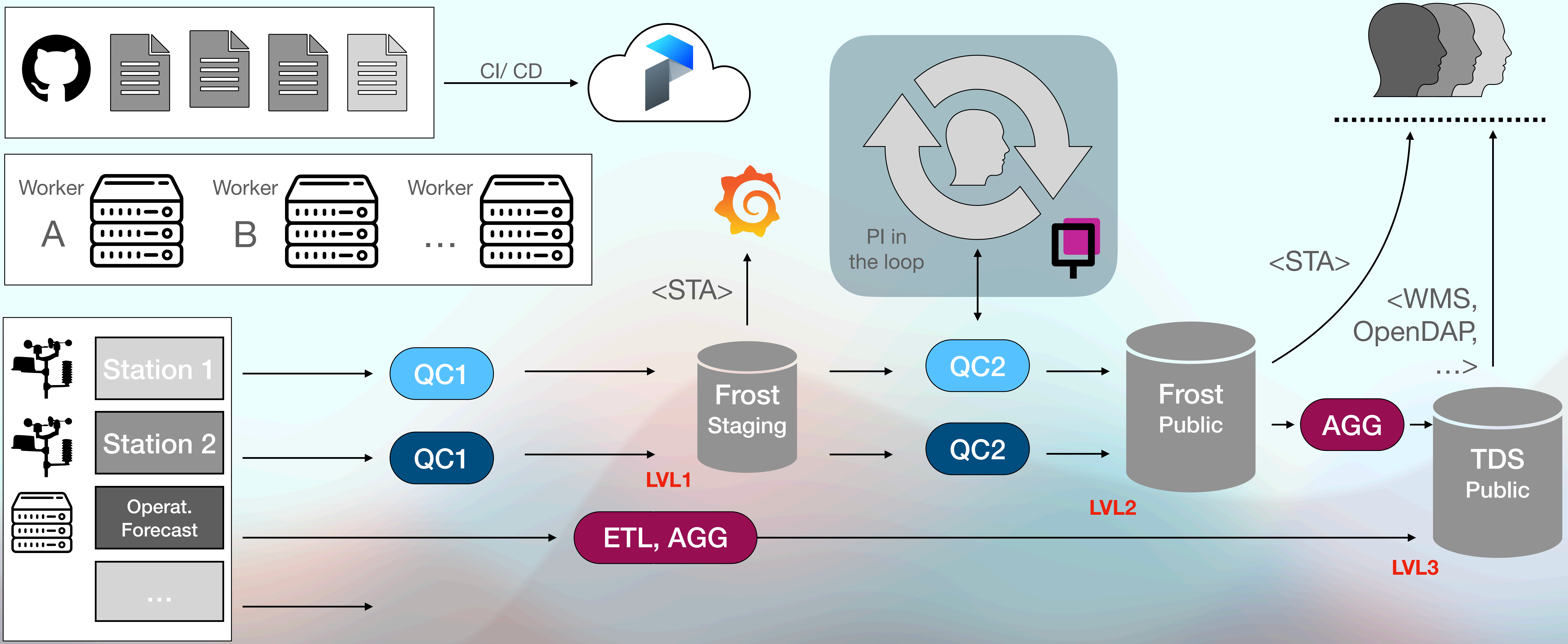
Proposed solution

… develop an flexible and scalabe dataflow framework

- Open-source
- Centralized workflows with Prefect
- Workflows as code (CI/CD pipelines after change)
- , Data Quality Checks with GreatExpectations and SaQC
- , Data publication via STA and THREDDs
- Observability via Prefect UI, Grafana, APIs

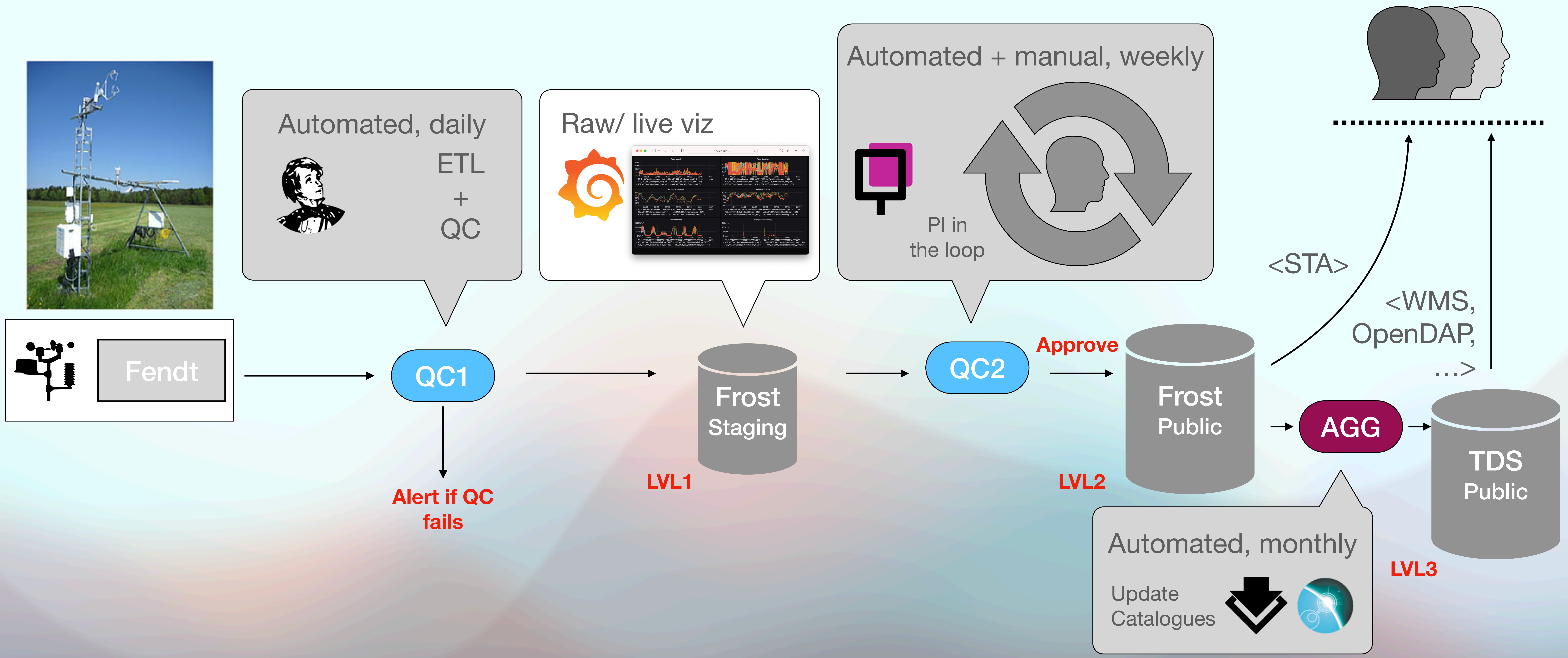
Workflow Orchestration

… schedule and control via Prefect



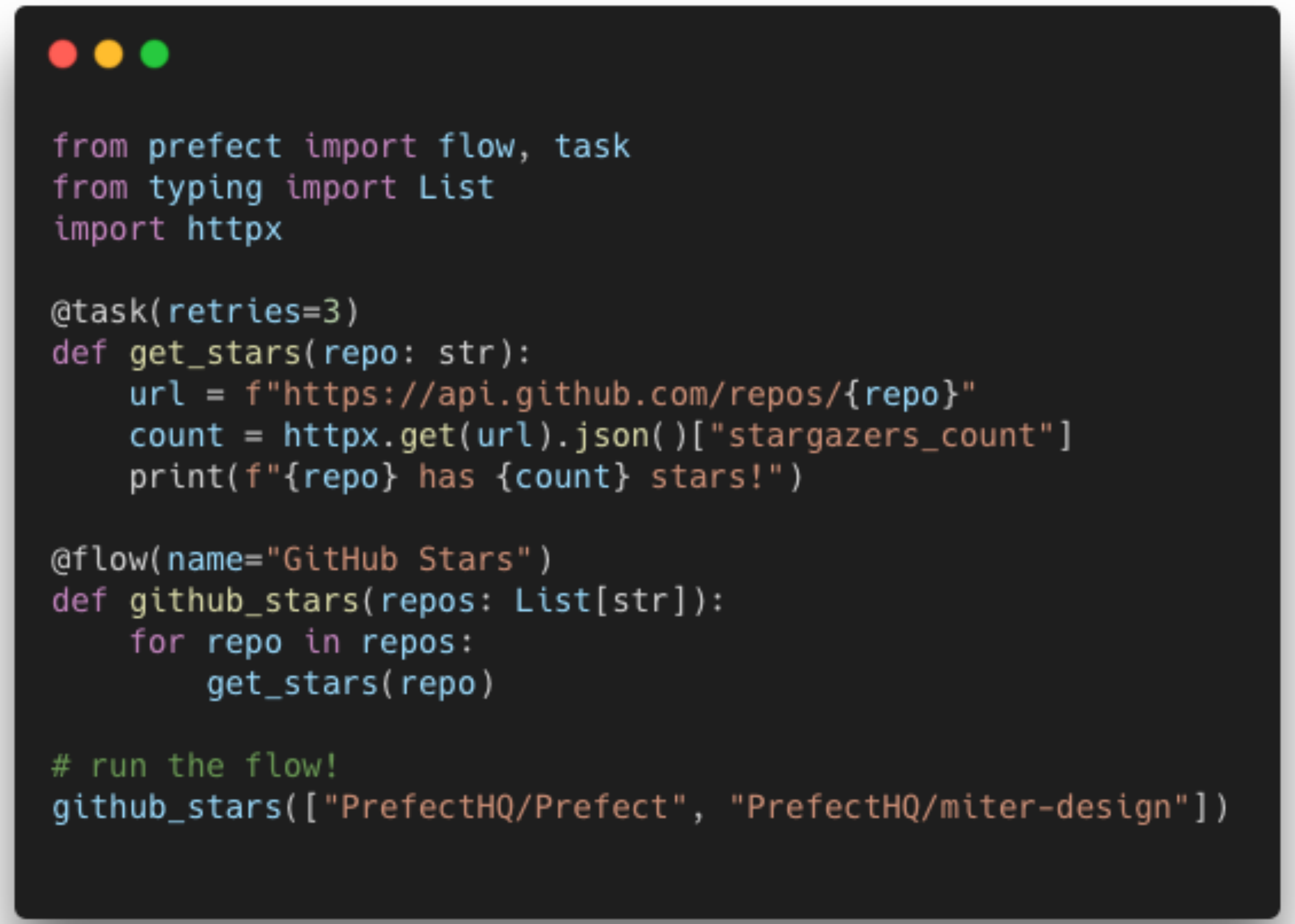
Example Workflow

TERENO micromet ingest, QA/ QC and publication



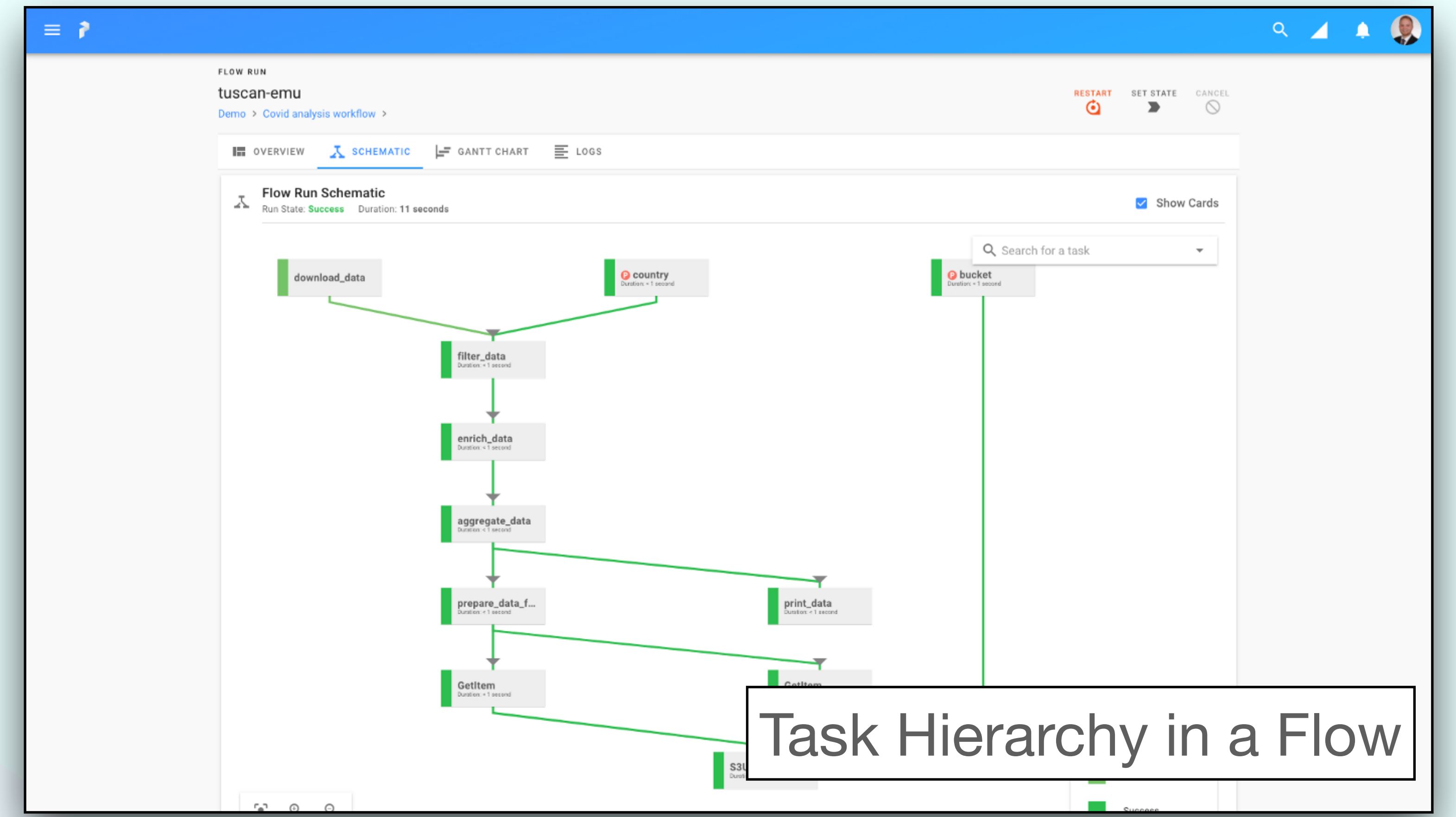
Prefect

Workflow definition (as code)

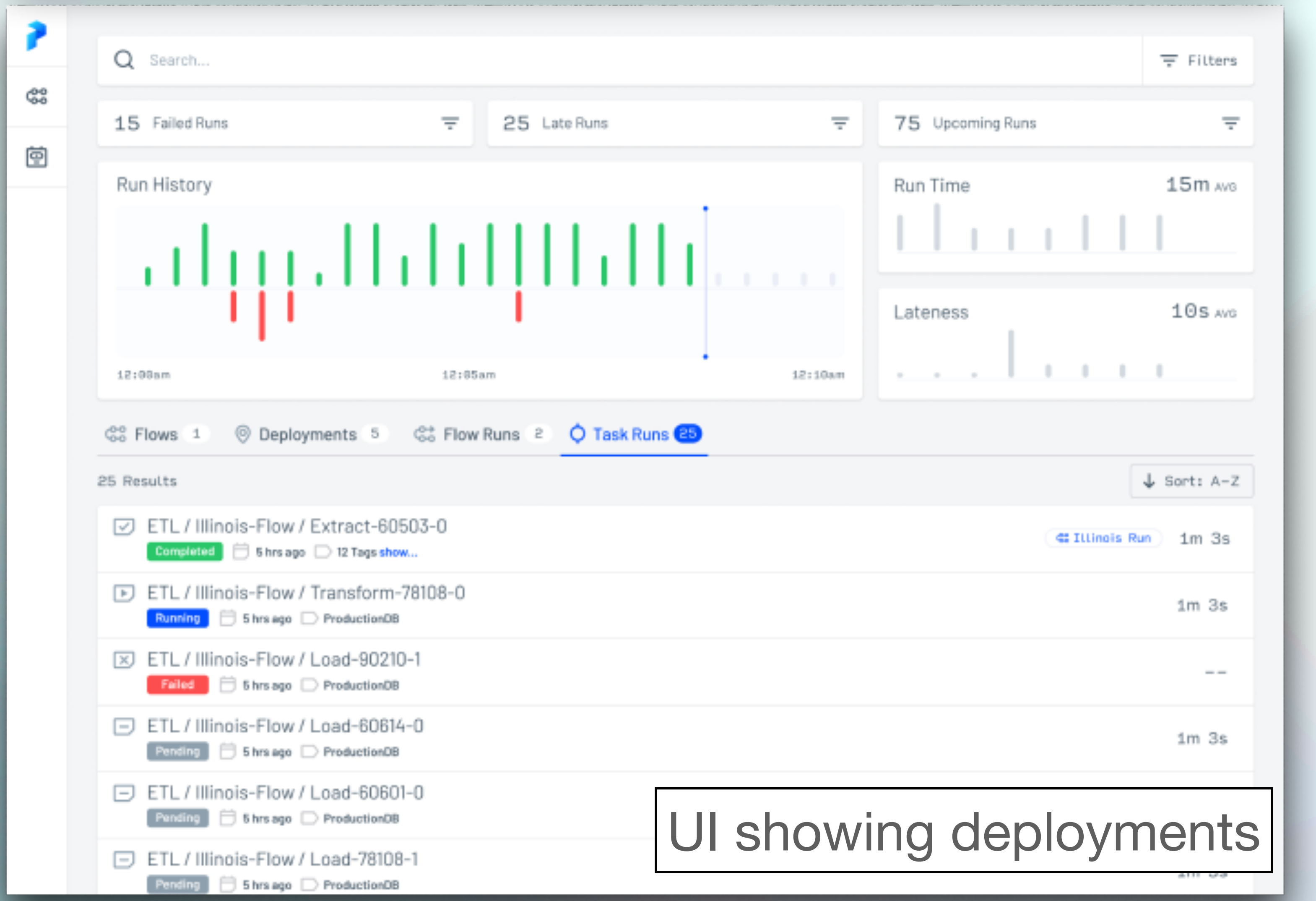


```python
from prefect import flow, task
from typing import List
import httpx

@task(retries=3)
def get_stars(repo: str):
    url = f"https://api.github.com/repos/{repo}"
    count = httpx.get(url).json()["stargazers_count"]
    print(f"{repo} has {count} stars!")

@flow(name="GitHub Stars")
def github_stars(repos: List[str]):
    for repo in repos:
        get_stars(repo)

# run the flow!
github_stars(["PrefectHQ/Prefect", "PrefectHQ/miter-design"])
```

Task and flow definition as code



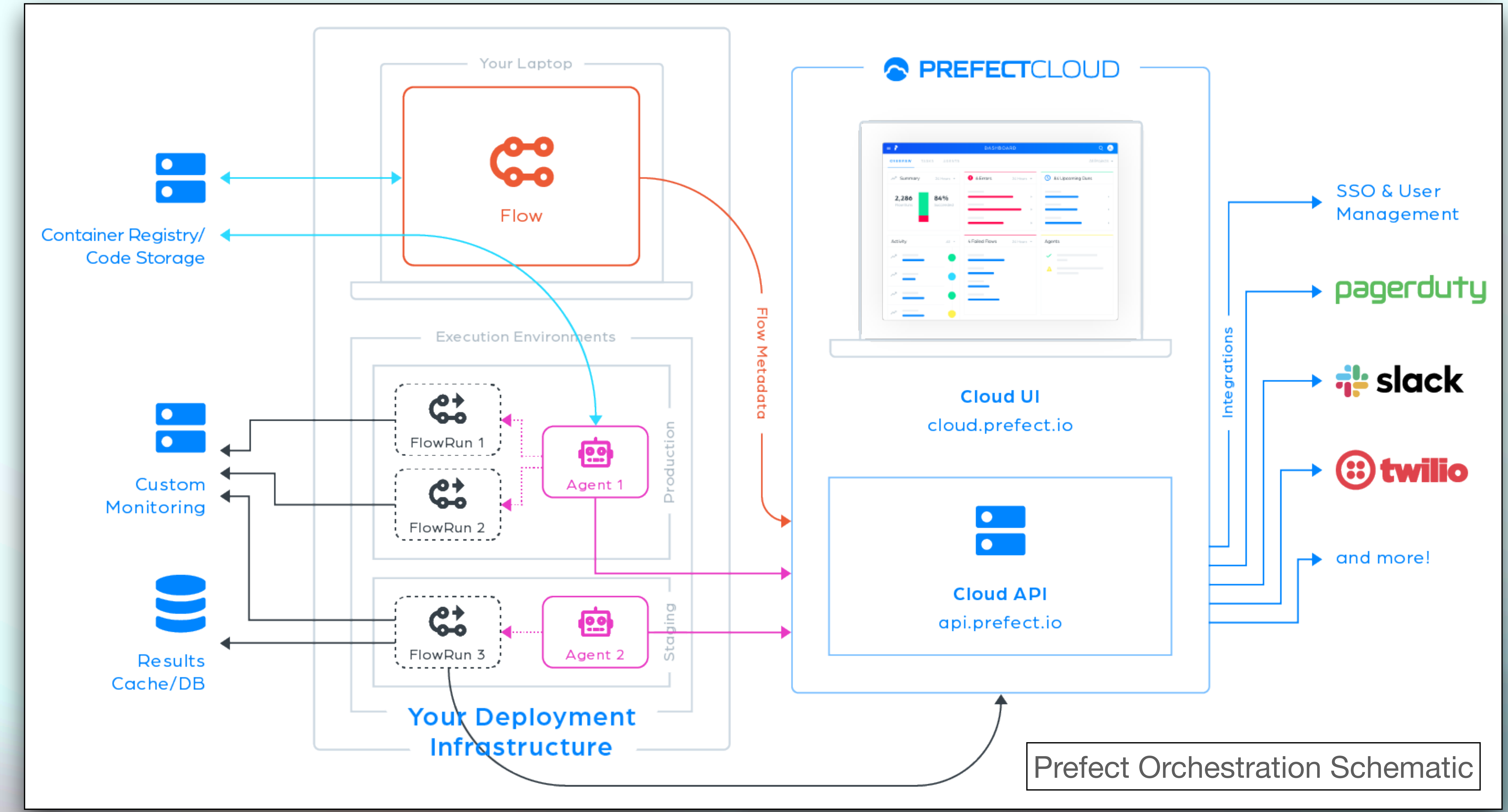
Task Hierarchy in a Flow


Search...
Filters
15 Failed Runs
25 Late Runs
75 Upcoming Runs
Run History
Run Time 15m AVG
Lateness 10s AVG
Flows 1
Deployments 5
Flow Runs 2
Task Runs 25
25 Results
Sort: A-Z
ETL / Illinois-Flow / Extract-60503-0
Completed
ETL / Illinois-Flow / Transform-78108-0
Running
ETL / Illinois-Flow / Load-90210-1
Failed
ETL / Illinois-Flow / Load-60614-0
Pending
ETL / Illinois-Flow / Load-60601-0
Pending
ETL / Illinois-Flow / Load-78108-1
Pending

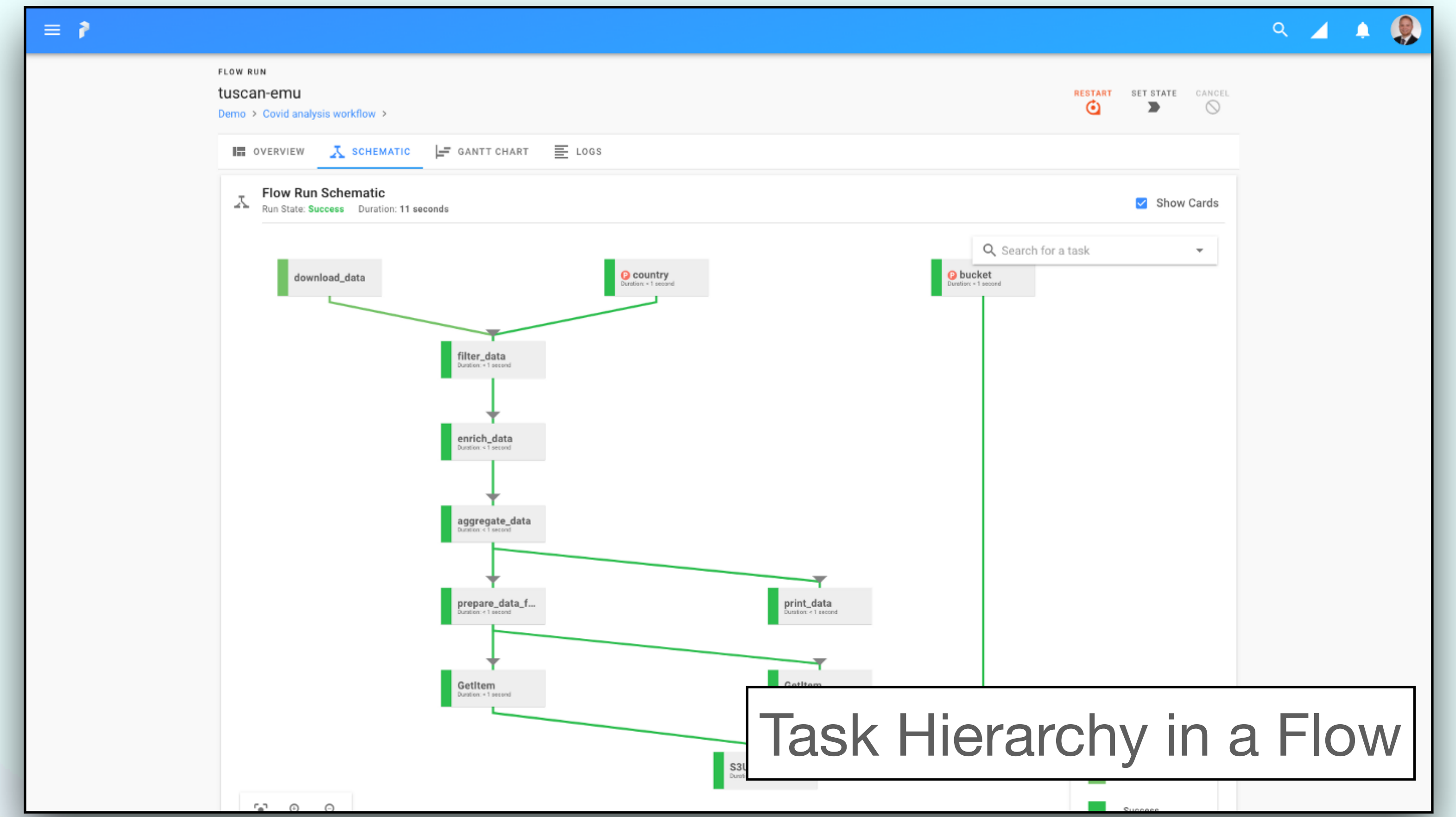
UI showing deployments

Prefect

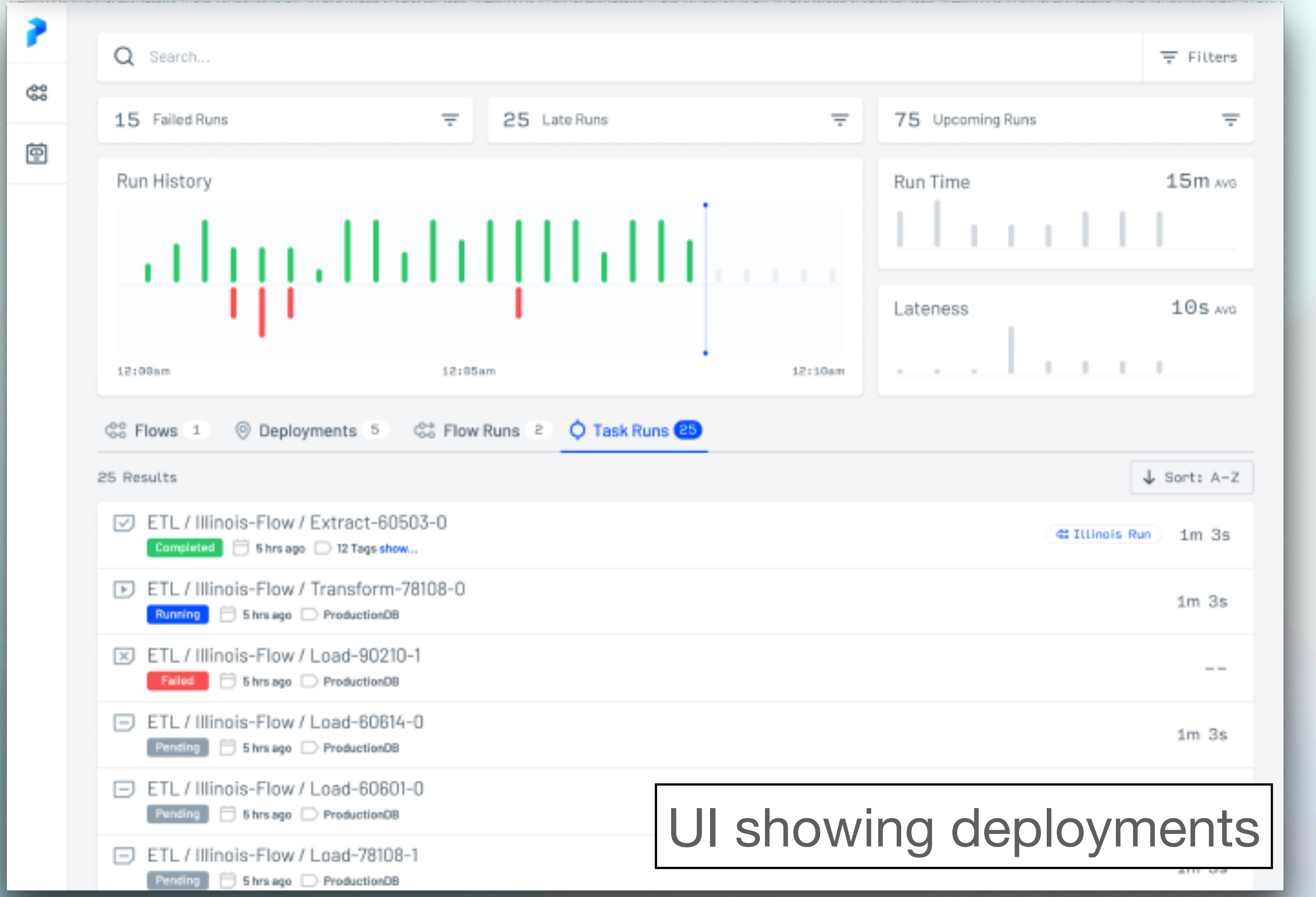
Workflow orchestration



Prefect Orchestration Schematic



Task Hierarchy in a Flow



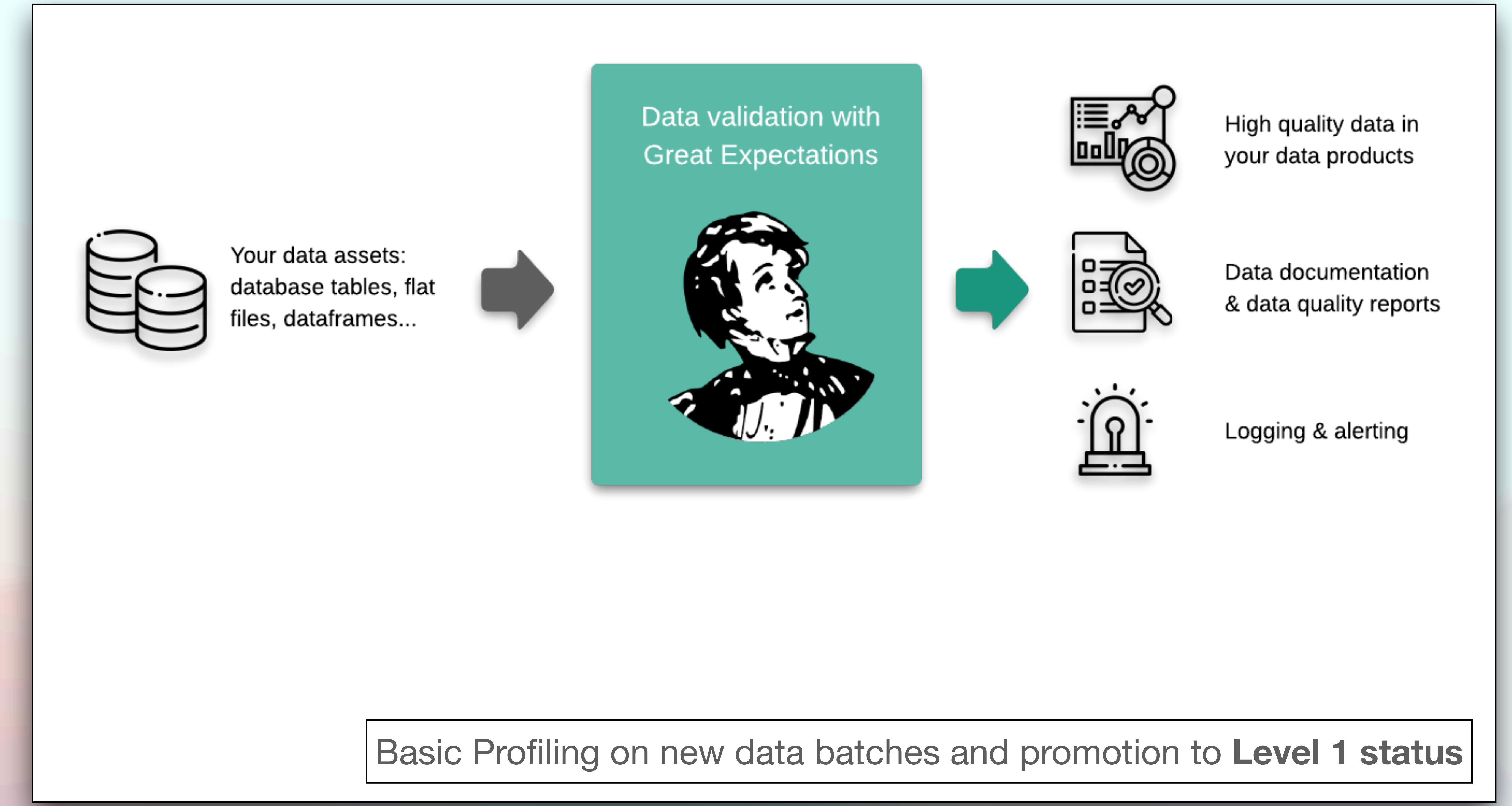
ETL / Illinois-Flow / Extract-60503-0 — Completed — 5 hrs ago — 12 Tags show... — Illinois Run — 1m 3s
ETL / Illinois-Flow / Transform-78108-0 — Running — 5 hrs ago — ProductionDB — 1m 3s
ETL / Illinois-Flow / Load-90210-1 — Failed — 5 hrs ago — ProductionDB — --
ETL / Illinois-Flow / Load-60614-0 — Pending — 5 hrs ago — ProductionDB — 1m 3s
ETL / Illinois-Flow / Load-60601-0 — Pending — 5 hrs ago — ProductionDB
ETL / Illinois-Flow / Load-78108-1 — Pending — 5 hrs ago — ProductionDB

UI showing deployments

Quality Control I

Great Expectations - Data Profiling and initial QC

- Initial automated profiling on existing valid data
- Define “expectations”
- Checks preformed at ingest time (8am), alert if missing data and other expectations are not met
- Valid data ingested into FROST staging server [LVL1]

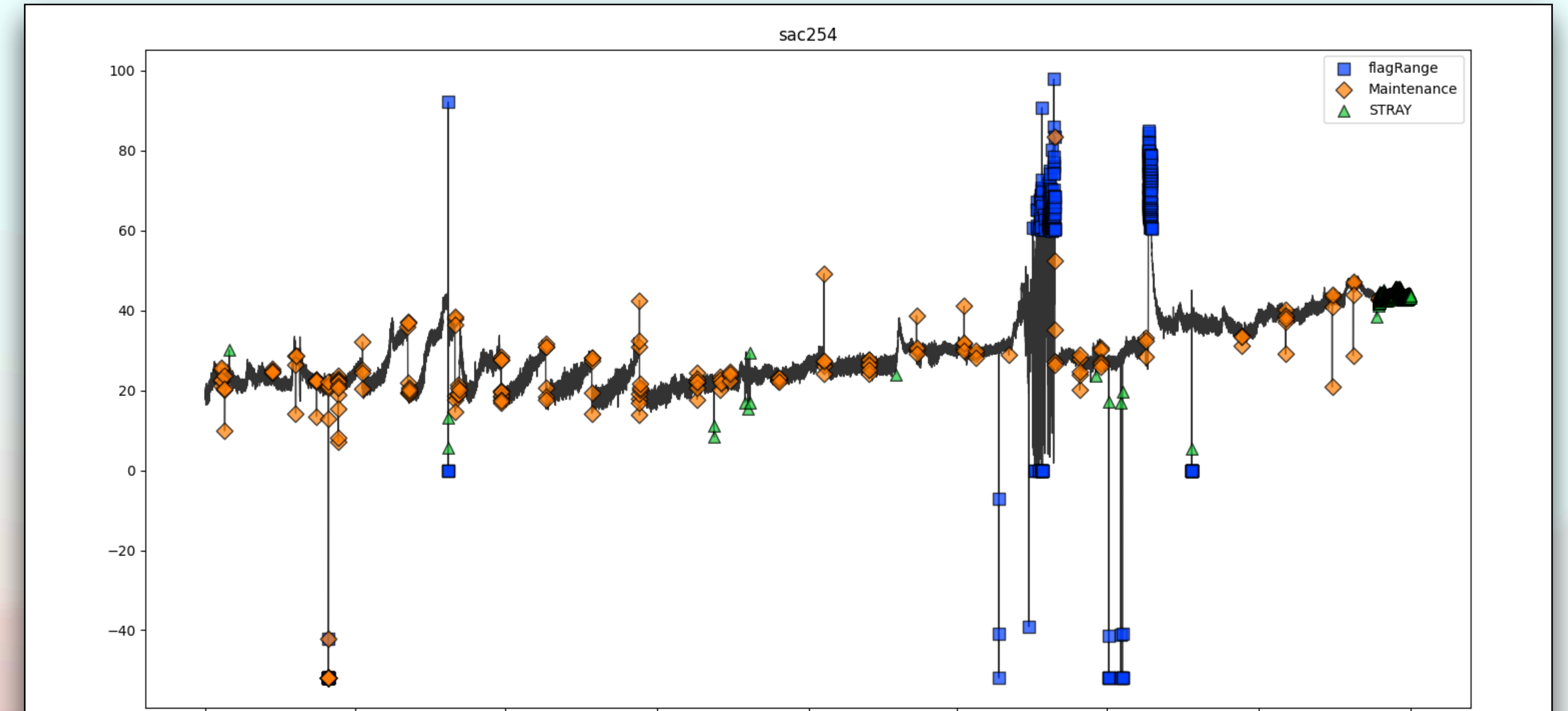

Your data assets: database tables, flat files, dataframes...
Data validation with Great Expectations
High quality data in your data products
Data documentation & data quality reports
Logging & alerting

Basic Profiling on new data batches and promotion to **Level 1 status**

Quality Control II

SaQC - measurement/ project-specific QC

- Manual intervention step at weekly interval
- Domain knowledge required
- Checks performed on time-span of data (data drift detection, outliers etc.)
- After approval data accessible via STA from 2nd FROST server [LVL2]

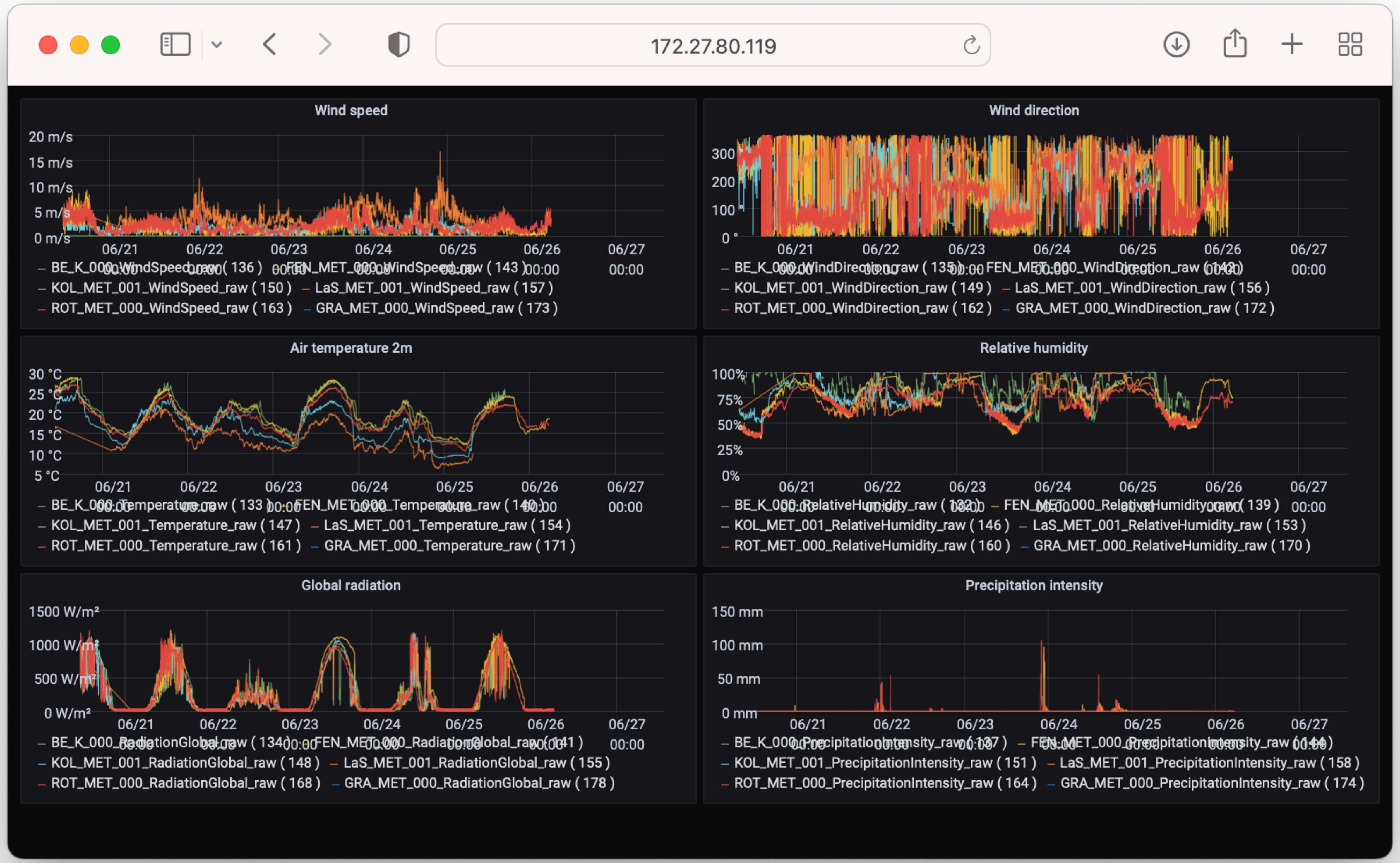


Detailed Profiling and Data Promotion to **Level 2 status**

Data Access

Sensor Things API

- Data access via STA API (REST or via code, i.e. stantic)
- (Grafana) Dashboards

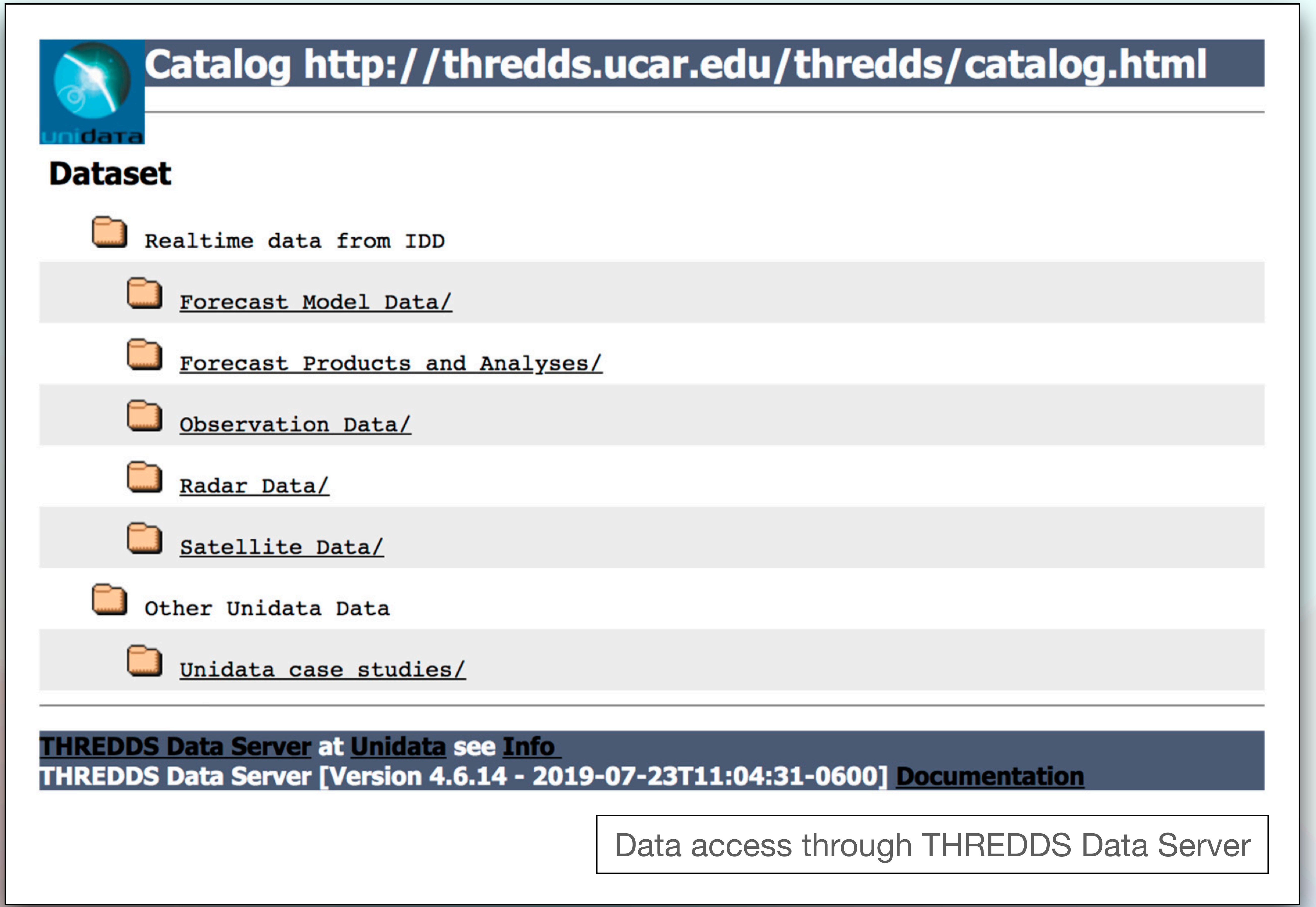

172.27.80.119
Wind speed
20 m/s
15 m/s
10 m/s
5 m/s
0 m/s
06/21 00:00
06/22 00:00
06/23 00:00
06/24 00:00
06/25 00:00
06/26 00:00
06/27 00:00
BE_K_000_WindSpeed_raw (136)
FEN_MET_000_WindSpeed_raw (143)
KOL_MET_001_WindSpeed_raw (150)
LaS_MET_001_WindSpeed_raw (157)
ROT_MET_000_WindSpeed_raw (163)
GRA_MET_000_WindSpeed_raw (173)
Wind direction
300
200
100
0
BE_K_000_WindDirection_raw (135)
FEN_MET_000_WindDirection_raw (142)
KOL_MET_001_WindDirection_raw (149)
LaS_MET_001_WindDirection_raw (156)
ROT_MET_000_WindDirection_raw (162)
GRA_MET_000_WindDirection_raw (172)
Air temperature 2m
30 °C
25 °C
20 °C
15 °C
10 °C
5 °C
BE_K_000_Temperature_raw (133)
FEN_MET_000_Temperature_raw (140)
KOL_MET_001_Temperature_raw (147)
LaS_MET_001_Temperature_raw (154)
ROT_MET_000_Temperature_raw (161)
GRA_MET_000_Temperature_raw (171)
Relative humidity
100%
75%
50%
25%
0%
BE_K_000_RelativeHumidity_raw (132)
FEN_MET_000_RelativeHumidity_raw (139)
KOL_MET_001_RelativeHumidity_raw (146)
LaS_MET_001_RelativeHumidity_raw (153)
ROT_MET_000_RelativeHumidity_raw (160)
GRA_MET_000_RelativeHumidity_raw (170)
Global radiation
1500 W/m²
1000 W/m²
500 W/m²
0 W/m²
BE_K_000_RadiationGlobal_raw (134)
FEN_MET_000_RadiationGlobal_raw (141)
KOL_MET_001_RadiationGlobal_raw (148)
LaS_MET_001_RadiationGlobal_raw (155)
ROT_MET_000_RadiationGlobal_raw (168)
GRA_MET_000_RadiationGlobal_raw (178)
Precipitation intensity
150 mm
100 mm
50 mm
0 mm
BE_K_000_PrecipitationIntensity_raw (137)
FEN_MET_000_PrecipitationIntensity_raw (144)
KOL_MET_001_PrecipitationIntensity_raw (151)
LaS_MET_001_PrecipitationIntensity_raw (158)
ROT_MET_000_PrecipitationIntensity_raw (164)
GRA_MET_000_PrecipitationIntensity_raw (174)

Data access and live plotting of Level 1/2 data via STA

Data Access

THREDDS Data Server

- Aggregated datasets [LVL3] (csv, netCDF, monthly/ annual)
- Automatically expand data catalog
- Intake catalog
- Harvested into searchable Geoserver and/ or Data Portal



Catalog http://thredds.ucar.edu/thredds/catalog.html

unidata

Dataset

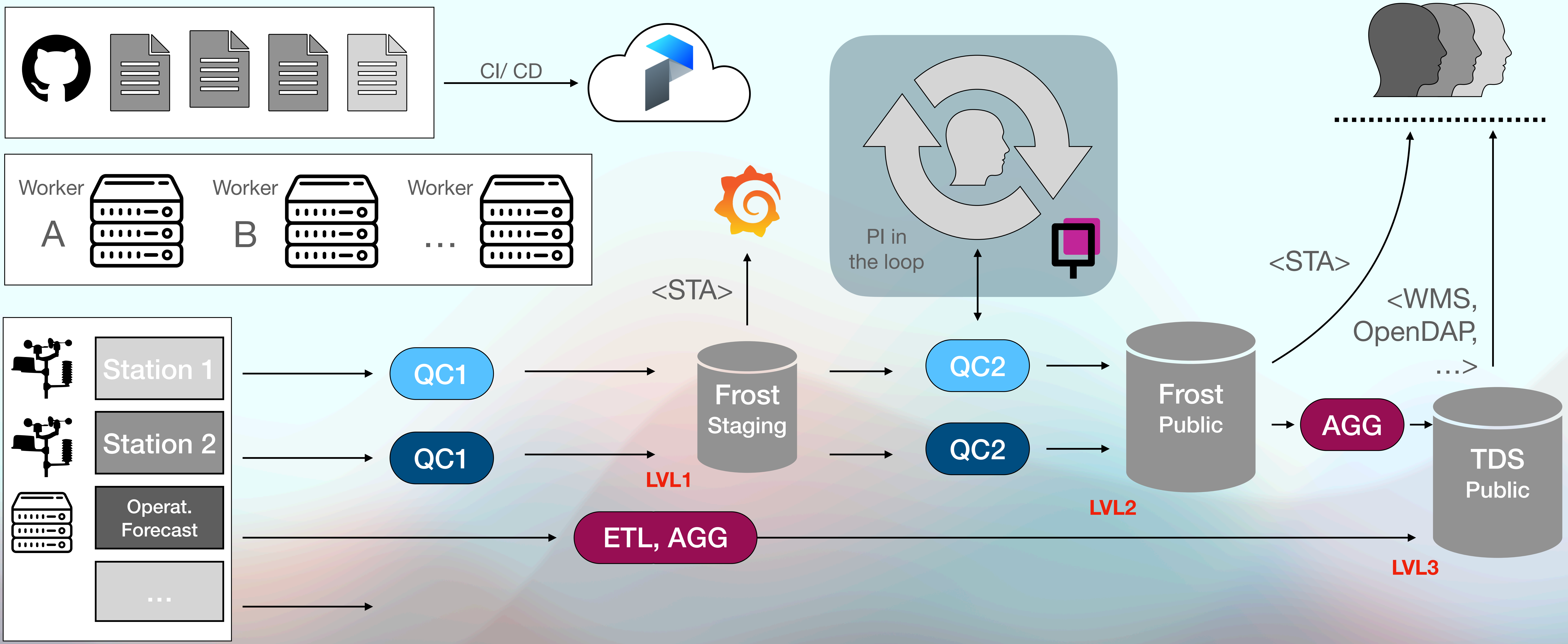
- Realtime data from IDD
  - Forecast Model Data/
  - Forecast Products and Analyses/
  - Observation Data/
  - Radar Data/
  - Satellite Data/
- Other Unidata Data
  - Unidata case studies/

THREDDS Data Server at Unidata see Info
THREDDS Data Server [Version 4.6.14 - 2019-07-23T11:04:31-0600] Documentation

Data access through THREDDS Data Server

Workflow Orchestration

… schedule and control via Prefect



References

- FROST Server: https://fraunhoferiosb.github.io/FROST-Server
- Great Expectations: https://greatexpectations.io
- Intake: https://intake.readthedocs.io
- OGC Sensor Things API: https://www.ogc.org/standards/sensorthings
- Prefect: https://prefect.io
- Stantic: https://cwerner.github.io/stantic
- System for automated Quality Control (SaQC): https://rdm-software.pages.ufz.de/saqc
- THREDDS Data Server: https://www.unidata.ucar.edu/software/tds