CODECO

Cognitive Decentralised
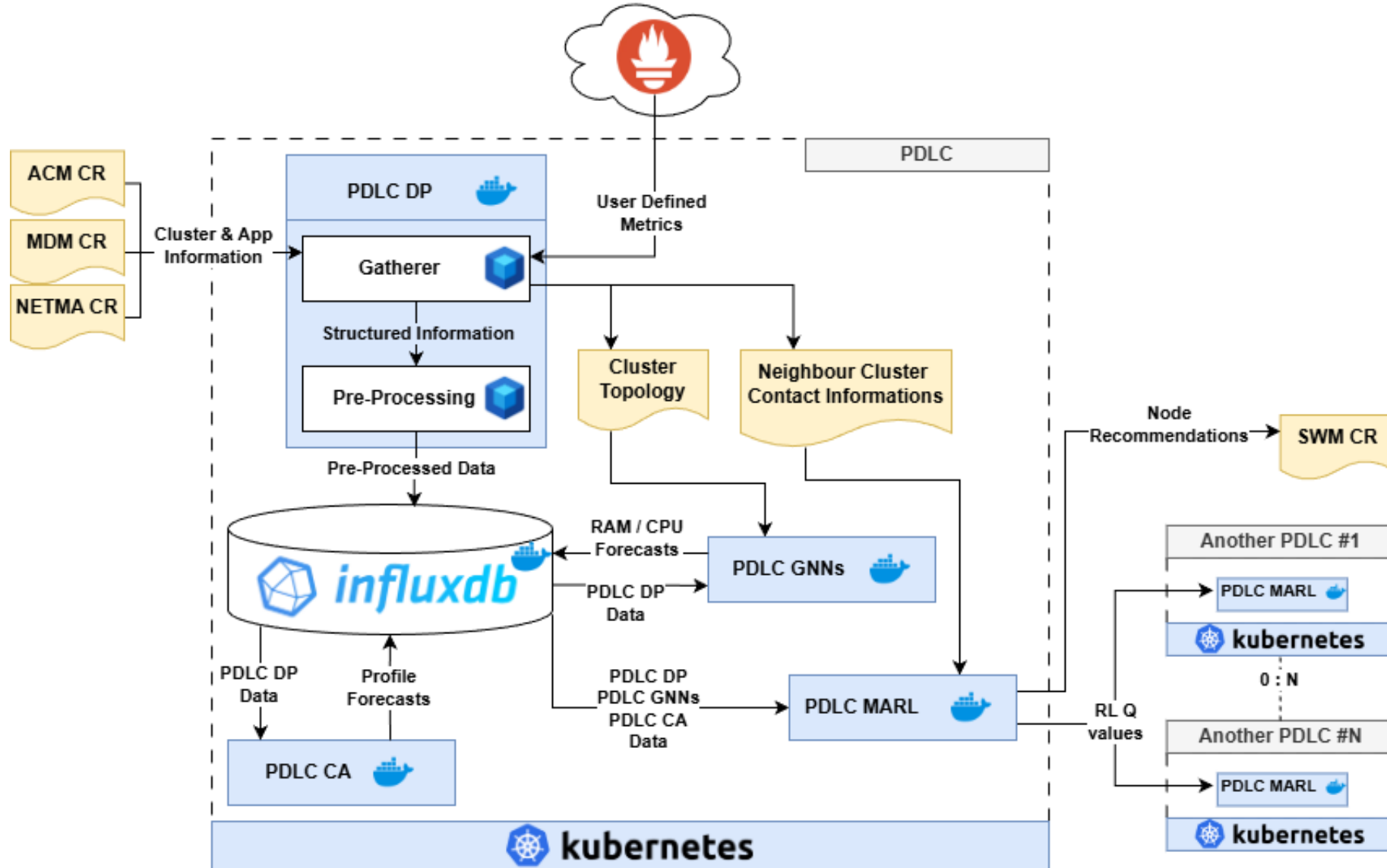Edge Cloud Orchestration

Panagiotis Karamolegkos

CODECO's Experimentation & Validation Leader

University of Piraeus Research Centre

CODECO: PDLC

Date: 26/01/26

Venue: HIPEAC ML4CS Workshop

PDLC HIGH LEVEL ARCHITECTURE

PDLC DP

PDLC-DP aims to provide processed data treated with a range of techniques aimed to improved the quality and interpretability of data used by the rest of the components of PDLC.
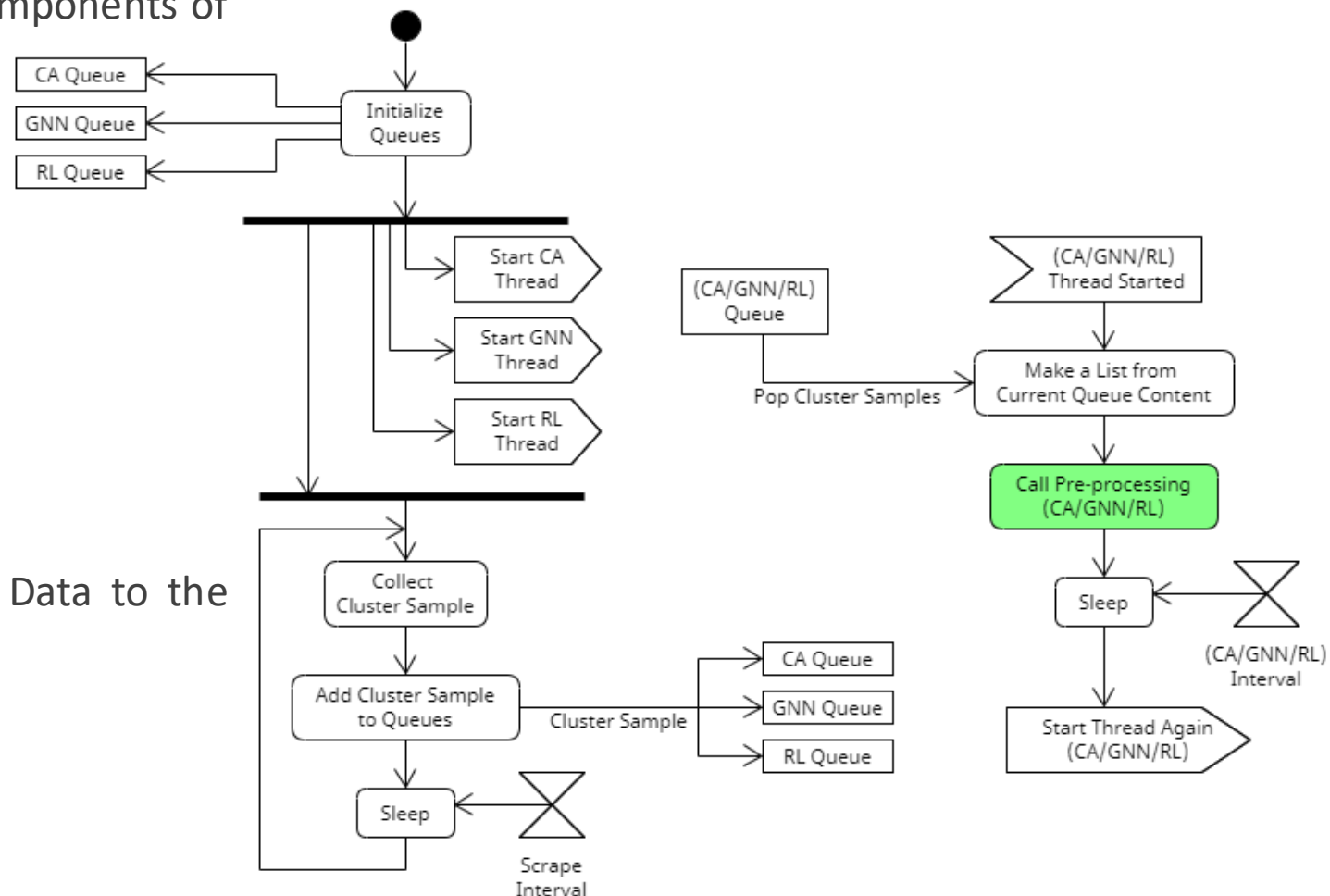
Gatherer:
- Gathers Data from Prometheus and CODECO

Pre-Processing:
- Data Cleaning
- Data Integration
- Data Transformation
- Data Normalization

Outputs:
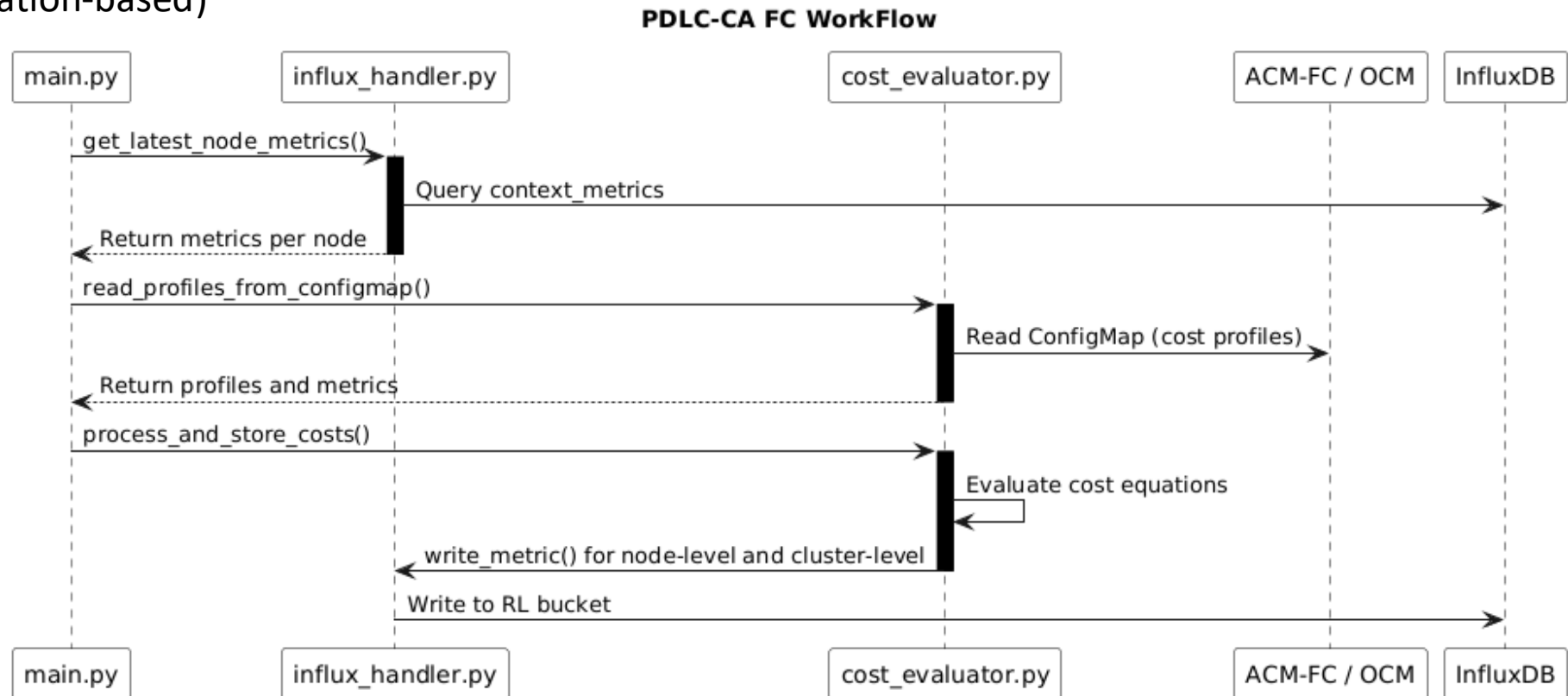- Provides in a Normalised Manner the above Data to the PDLC Subcomponents

PDLC CA

PDLC CA translates raw infrastructure metrics into decision-ready costs for scheduling and lifecycle management.

Performance Profiles (Policy Layer)
- Greenness (predefined)
- Resilience (predefined)
- UserDefined (custom, equation-based)

Zero-Code Customization

PDLC-CA FC WorkFlow

PDLC GNNS

PDLC GNNs uses the state of the system in order to create forecasts of CPU and RAM consumption across the cluster so that they can be used as input of the **PDLC MARL** subcomponent.
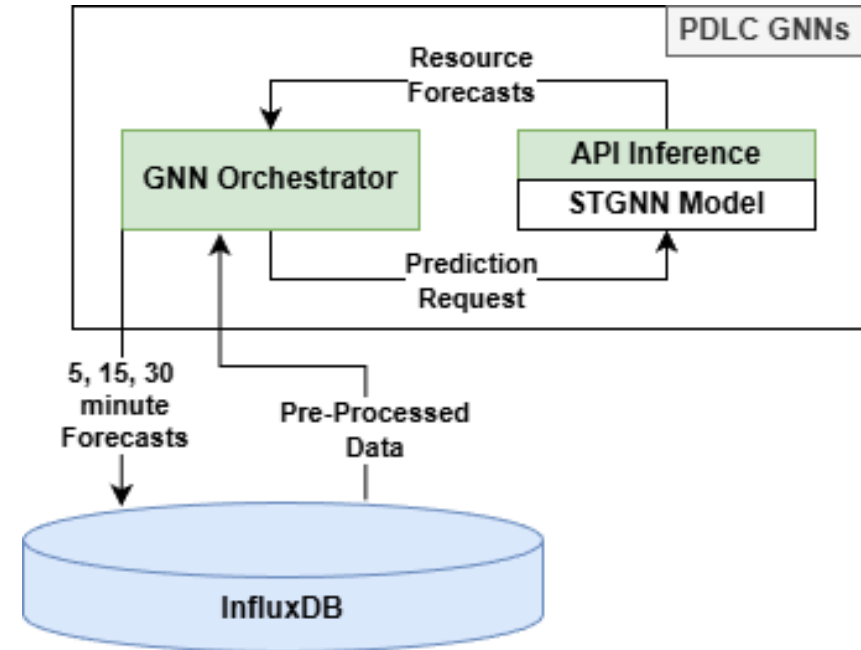
Input data:
- Gathers data from PDLC DP, obtaining consumption of CPU and RAM across all nodes in the cluster.

Functioning:
- Queries API with historical data.
- Model is selected dynamically depending on the number of nodes in the cluster.
- Predicts future values on CPU and RAM consumption based on historical observations by using spatio-temporal graphs.

Outputs:
- Forecasted values are written in csv file depending on the forecast time (e.g. 5,15 or 30 minutes) so that **MARL** can use them.

PDLC MARL 1/2

PDLC MARL uses the current (or forecasted if available) state of the system together with custom metrics such as greenness in order to provide node recommendations to optimize the chosen performance profile (e.g. reduce energy consumption in the system).
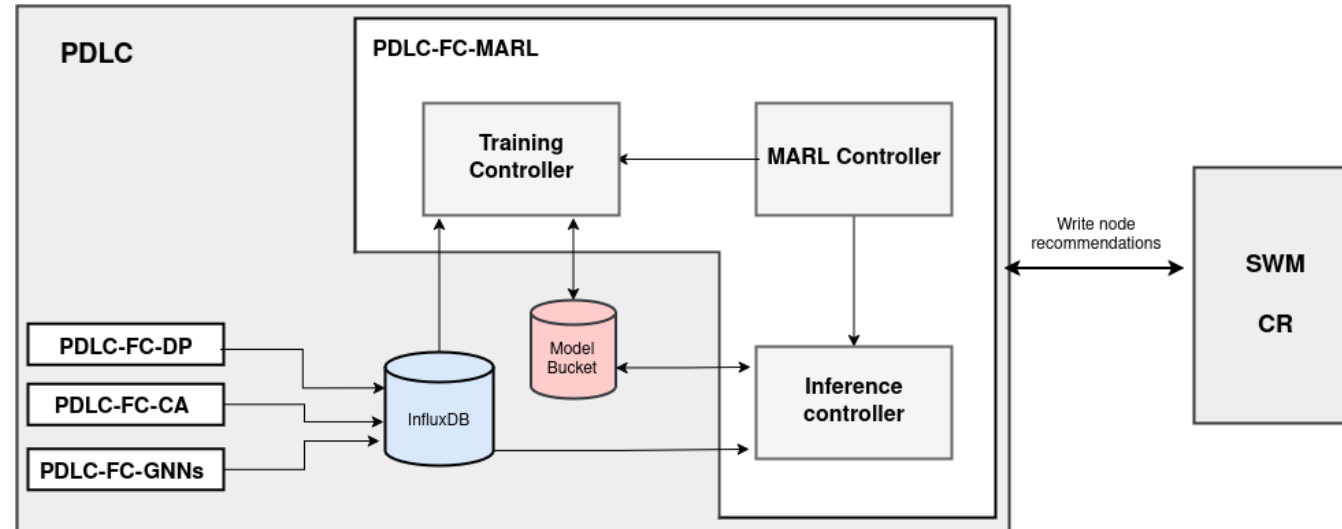It serves as the central component of PDLC.

Input data:
- Gathers data from all the rest of the PDLC subcomponents.
- It periodically obtains the assignment plan created by SWM (future usage) via K8s API.
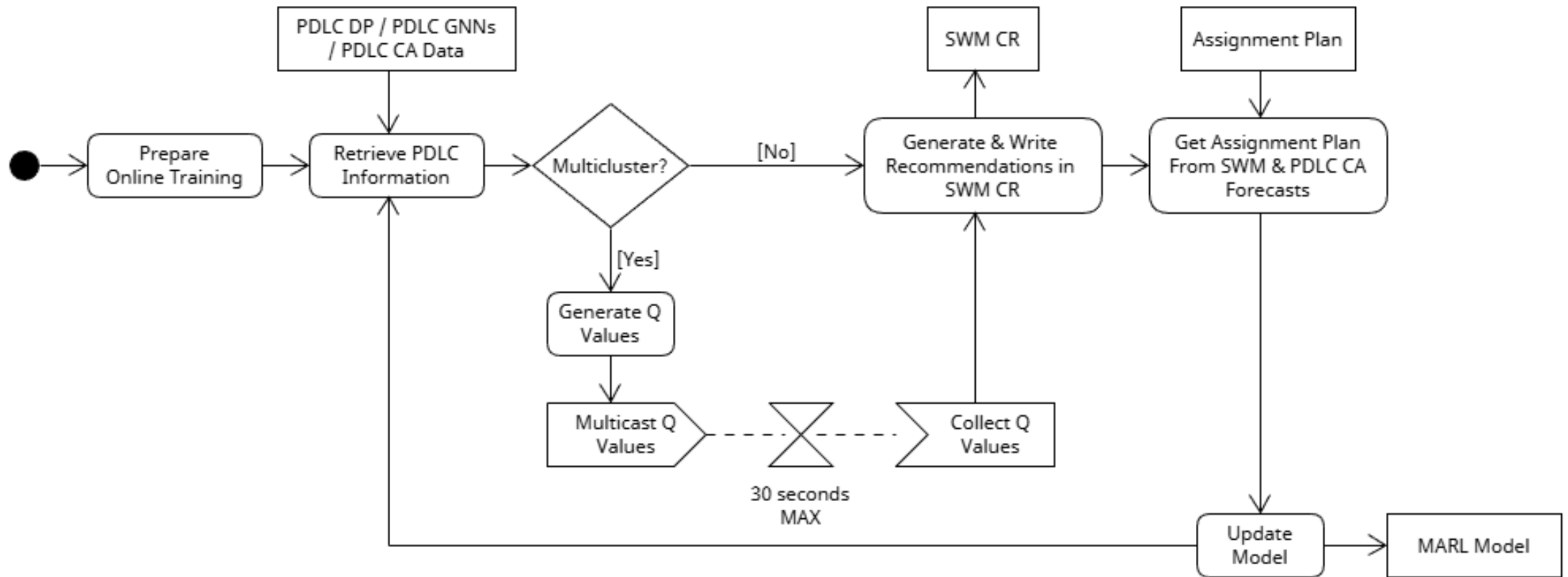
Functioning:
- Model is selected dynamically depending on the number of nodes in the cluster.
- Creates node recommendations based on data received.
- Collaborates in the Training of Neighborhood PDLC MARL subcomponents.

Outputs:
- Writes node recommendations in SWM CRs via K8s python API.


```
nodeRecommendations:
    codeoco-master: 1.0
    codeco-worker1: 0.9
    codeco-worker2: 0.0
```
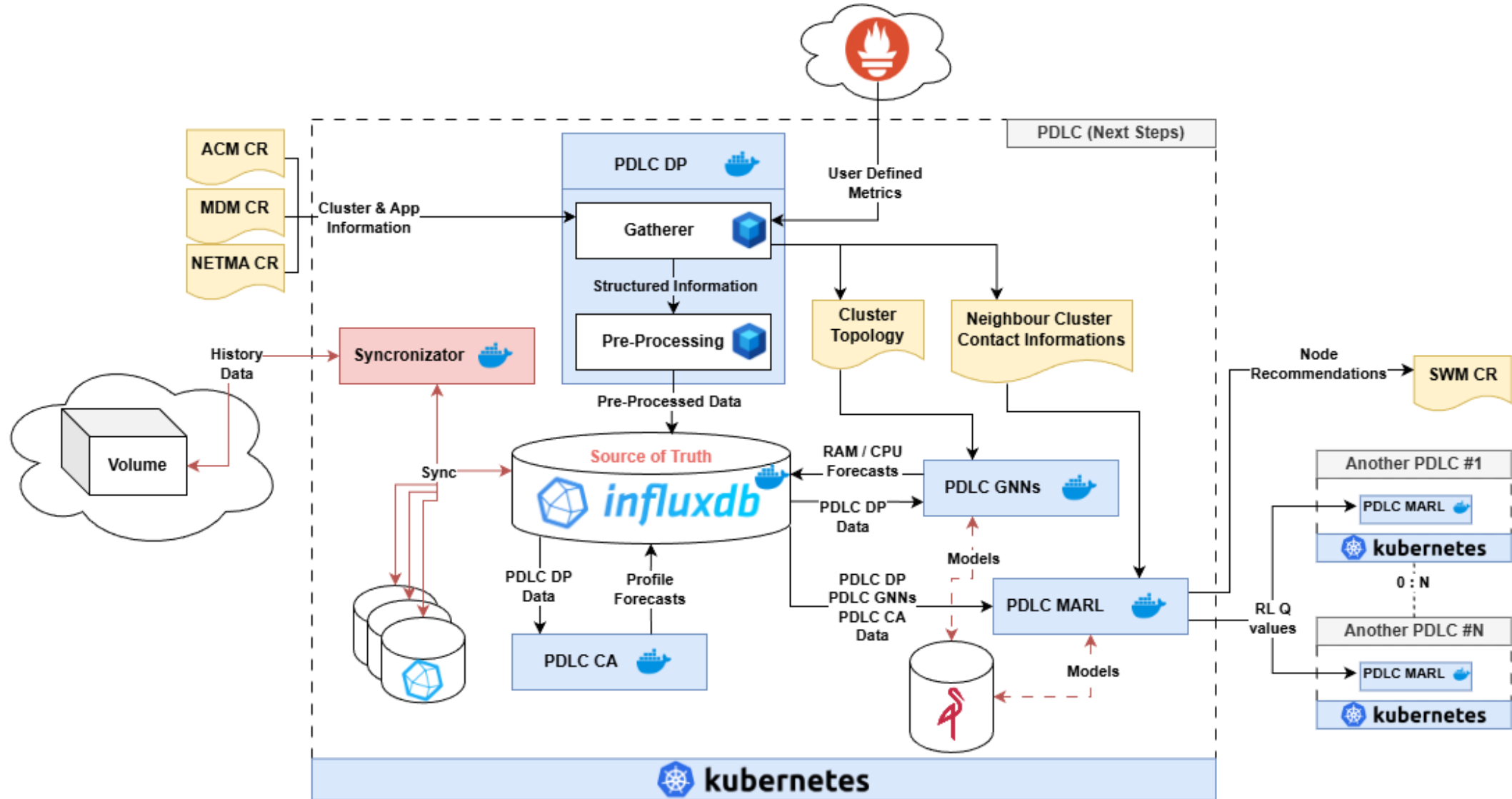
PDLC MARL Activity Diagram

Works on my machine™
(Let's see it run)

PDLC VISION & NEXT STEPS

THANK YOU!

Questions?

Panagiotis Karamolegkos

www.karamolegkos.dev

pkaram@unipi.gr

MEET OUR CONSORTIUM