# $CO_2$ Monitoring for Energy-Efficient Workloads in Kubernetes: A Data Provider for $CO_2$-Aware Migration

Ivo Hrib
ivo.hrib@gmail.com
Jožef Stefan Institute
Ljubljana, Slovenia

Jan Šturm
jan.sturm@ijs.s
Jožef Stefan Institute
Ljubljana, Slovenia

Oleksandra Topal
oleksandra.topal@ijs.si
Jožef Stefan Institute
Ljubljana, Slovenia

Maja Škrjanc
maja.skrjanc@ijs.si
Jožef Stefan Institute
Ljubljana, Slovenia

## Abstract

We present a $CO_2$ monitoring component developed within the FAME project's Energy Efficient Analytics Toolbox. The service continuously collects power usage for containerized workloads in Kubernetes via Kepler and fuses it with regional electricity-grid carbon intensity (e.g., ElectricityMaps) to compute per-workload $CO_2$ emission rates in $g\,s^{-1}$. Its primary role is to *store* accurate, timestamped emission values and expose them through lightweight APIs and an optional time-series database (TimescaleDB). It acts as a data provider consumed by external orchestration services, enabling $CO_2$-aware migration strategies across clusters and regions.

## Keywords

CO2 monitoring, Kubernetes, energy efficiency, carbon-aware computing, time-series storage, ElectricityMaps, Kepler

## 1 Introduction

Data centers are a significant contributor to global electricity demand. Beyond advances in hardware efficiency and renewable energy procurement, intelligent orchestration of workloads can reduce emissions by aligning computation with cleaner energy availability. A prerequisite for such carbon-aware orchestration is *reliable and accessible measurements* of workload-level emissions.

This paper introduces a $CO_2$ monitoring and storage service designed for Kubernetes environments. The service ingests pod/-container power data from Kepler [5], combines it with regional grid carbon intensity from ElectricityMaps [2], computes instantaneous emission rates, and *persists* the resulting time series. Unlike optimization or migration tools, this component deliberately restricts its scope: it provides measurements and exposes them via stable APIs for later consumption.

By decoupling measurement from decision-making, we ensure modularity and interoperability. External orchestrators such as the ATOS migration service in FAME D5.4 [3] can consume these metrics to implement $CO_2$-aware migration strategies without needing to handle the intricacies of measurement or data storage.

Our contributions are threefold: (i) a minimal but complete architecture for per-workload $CO_2$ measurement and storage in Kubernetes; (ii) a schema and REST API design that facilitates external consumption; and (iii) scenario-based evaluations demonstrating the potential of $CO_2$-aware workload migration.

Further testing will take place, utilizing real measurements and migrations from within the FAME framework, as to showcase the service's precise final capabilities, as opposed to benchmark tests.

### 1.1 Key-Idea

The key idea of our approach is to compute container-level $CO_2$ emissions by combining two complementary data sources: (i) instantaneous power consumption estimates from *Kepler*, and (ii) regional grid carbon intensity values from *ElectricityMaps*.

First, Kepler provides pod- and container-level telemetry in the form of estimated power usage $P(t)$, expressed in watts. This power signal is derived from eBPF-based kernel observations and model-based inference all provided by Keplers data source. Second, ElectricityMaps exposes a carbon intensity factor $I(t)$, expressed in $gCO_2/kWh$, corresponding to the bidding zone of the node on which the container executes.

We align these two signals in time and compute instantaneous emission rates by:

$$E(t) = P(t) \cdot \frac{I(t)}{3600}$$

where $E(t)$ is the $CO_2$ emission rate in $g\,s^{-1}$, $P(t)$ is container power in watts ($J\,s^{-1}$), and the division by 3600 converts the intensity factor from per-kWh to per-second units.

These per-container emission rates are then aggregated into a time series, optionally persisted in TimescaleDB, and exposed via a REST API. This composition allows downstream orchestration services to reason about the carbon impact of workloads at fine temporal and spatial granularity, enabling $CO_2$-aware migration strategies.

## 2 Background and Related Work

**Components of our approach.** Our service integrates two external data sources to produce fine-grained $CO_2$ emission signals. *Kepler* is an open-source project that estimates the energy consumption of containerized workloads in Kubernetes by leveraging eBPF-based telemetry and machine learning models. It exposes per-container power and energy metrics that can be consumed by higher-level services. *ElectricityMaps* provides real-time and historical carbon intensity data for electricity grids, expressed in $gCO_2/kWh$. By fusing Kepler's workload-level power estimates with regional carbon intensity factors from ElectricityMaps, our system produces a continuous stream of container-level $CO_2$

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
*Information Society 2025, Ljubljana, Slovenia*
© 2025 Copyright held by the owner/author(s).
https://doi.org/10.70314/is.2025.sikdd.24

emission data. This stream can then be consumed by orchestration or scheduling components for migration and placement decisions.

**Carbon-aware computing.** Prior research demonstrates the potential of carbon-aware strategies, such as shifting workloads across time or regions to align with lower-carbon electricity supplies [4]. Such approaches rely on access to reliable, fine-grained emission signals to inform scheduling policies.

**Existing monitoring tools.** Several open-source frameworks exist for energy and carbon monitoring. For example, *CodeCarbon* [1] and *Scaphandre* [6] estimate workload emissions, but they rely on hardware-specific telemetry, such as Intel's Running Average Power Limit (RAPL) counters. This dependence limits portability to Intel CPUs and makes integration across heterogeneous infrastructures challenging. In contrast, our design—built on Kepler and ElectricityMaps—remains hardware-agnostic: eBPF enables container-level monitoring without vendor-specific counters, while ElectricityMaps provides global coverage of carbon intensity signals. This combination makes our service applicable in diverse Kubernetes environments and datacenter setups.

**Time-series storage.** Finally, for persistence, we optionally employ TimescaleDB, which extends PostgreSQL with hypertables and compression optimized for telemetry data [7]. Nevertheless, the service can also operate in buffer-only mode when persistent storage is not required.

**Positioning.** This paper positions our monitoring service as a foundational measurement substrate for carbon-aware orchestration in Kubernetes environments. By combining hardware-agnostic energy estimates with real-time grid carbon data, it extends the applicability of carbon-aware scheduling beyond the limitations of prior approaches.

# 3 Design and Implementation

## 3.1 Architecture

The component runs as a Kubernetes deployment. Workers collect power metrics from Kepler, fetch grid intensity values, compute emissions, and either persist results in TimescaleDB or serve them from memory. A REST API provides read-only access to historical and recent emissions.

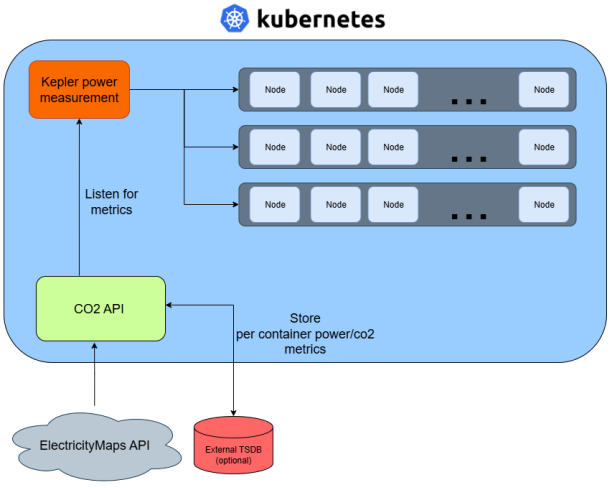



**Figure 1: System architecture**

## 3.2 Data Model

Each emission record is structured as a tuple that captures both workload identifiers and measurement values. This schema is designed to balance expressiveness with minimal storage overhead, while ensuring compatibility with external orchestration services.

- `ts` (timestamp, UTC): the precise moment when the measurement was taken, enabling time-series alignment across nodes and regions.
- `namespace`, `pod`, `container`: identifiers for locating the workload within the Kubernetes hierarchy, which is essential for container-level granularity and reproducibility.
- `node`, `region`, `country_iso2`: metadata that ties the container execution to its physical and geographical context. This supports carbon-aware decisions that depend on grid intensity differences across regions.
- `power_w`, `energy_j`: raw telemetry provided by Kepler, describing both instantaneous power and accumulated energy consumption.
- `intensity_g_per_kwh`: regional grid carbon intensity retrieved from ElectricityMaps, serving as the multiplier that translates energy into emissions.
- `co2_g_per_s`: the computed emission signal, representing the core value consumed by orchestrators.
- `source_version`: versioning tag for tracking provenance of measurements and external data dependencies.

This schema ensures that each record is self-contained, interpretable across clusters, and suitable for longitudinal analysis in time-series databases.

## 3.3 API Endpoints

The service exposes a lightweight REST API, designed to be easily consumed by external orchestrators or monitoring pipelines. The API emphasizes read-only access to maintain reliability and auditability.

- `GET /api/containers`: returns the set of containers currently monitored by the service, allowing orchestrators to discover available emission signals.
- `POST /api/emissions`: fetches recent emission values in bulk. This endpoint is optimized for dashboards or monitoring agents that need timely updates with low overhead. Requires a specified time range to return said emissions.
- `POST /api/emissions/by-container`: queries the emission history of a specific container, Similarly requires a time range, as well as the names of specific containers for which to fetch data.
- `GET /api/schema`: provides the data schema including units and field definitions. This enables clients to validate their assumptions and facilitates long-term interoperability across versions.

By standardizing access patterns, the API makes it possible for external services to reliably retrieve emissions information without depending on internal implementation details.

# 4 Evaluation

We now present evaluations based on benchmarks and scenario analyses conducted in the FAME project [3]. The goal was to assess whether exposing real-time $CO_2$ signals can enable meaningful emission reductions when coupled with migration strategies.

## 4.1 Benchmark Test

In a simple benchmark using busybox, a lightweight Linux container, the optimal $CO_2$ emissions achieved were significantly lower than the mean observed values. The key performance indicator (KPI) was defined as a 200% improvement, corresponding to a 66.6% reduction compared to baseline. Results show that this threshold can be achieved and often surpassed. The baseline is, for lack of a better, metric defined as the mean of emissions across all tracked countries with available resources for migration.

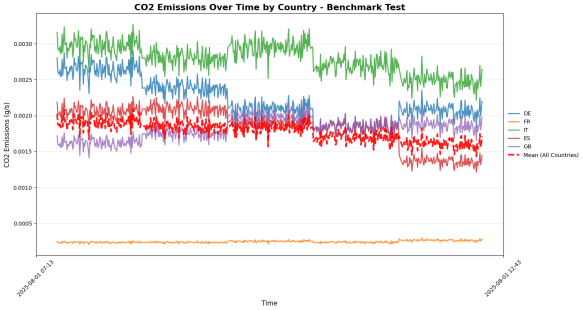



**Figure 2: Small timeframe emissions of a benchmark Busybox for testing purposes. We can see noticably low emisions for France, which can be explained by heavy reliance on nuclear power, as can be seen in [2]**

## 4.2 Scenario-Based Evaluation

Scenarios simulate workload migrations across subsets of European countries. Each scenario randomly selects 4–7 countries from a pool of 28, representing constrained deployment options. The system attempts to minimize emissions within these subsets.The abbreviations (e.g., FR, DE, SI) correspond to ISO-3166 country codes representing different electricity regions. We employed random sampling of countries to simulate the heterogeneity faced by cloud and edge providers operating across multiple regions. This choice enables us to reflect migration challenges where workloads are moved not only between datacenters but also across electricity grids with diverse carbon intensities. While random sampling is a simplification, it provides statistically representative insights into the variability of emission factors. We showcase oure results through the following 5 scenarios:

- Scenario 1 (IS, CZ, BG, RO, AT, SE): 88.2% ± 2.1% reduction.
- Scenario 2 (DE, PL, GR, LV): 72.8% ± 5.6% reduction.
- Scenario 3 (GB, LT, SI, DE, AT, GR): 78.0% ± 1.7% reduction.
- Scenario 4 (ES, FR, GB, PL, HU, LT, SE): 89.6% ± 1.1% (best case).
- Scenario 5 (LV, ES, HU, LT): 32.4% ± 12.7% (worst case).
- All Countries: 87.7% ± 1.7% reduction (ideal case).

Across all scenarios, at least one migration was executed per window, with an average emission reduction of 74.8%. These results confirm that even under limited availability, $CO_2$-aware migration strategies yield substantial benefits.

## 4.3 Insights

The best-performing scenario demonstrates that careful selection of even a limited number of regions can approach the effectiveness of full global availability. Conversely, the poorest-performing scenario illustrates the dependency on geographic flexibility. Overall, results validate that exposing reliable $CO_2$ signals through our service empowers orchestration layers to meet or exceed environmental KPIs.

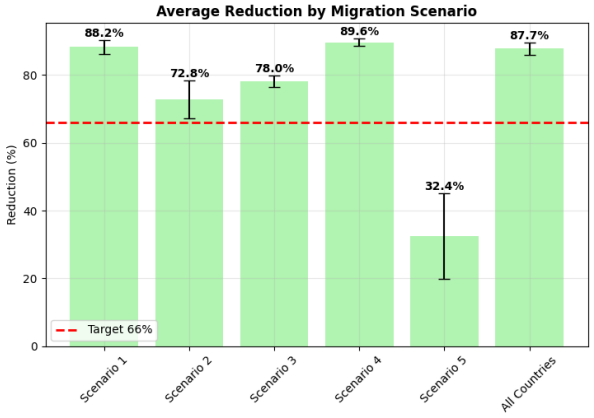



**Figure 3: Plot of average reductions per scenario**

## 5 Limitations and Future Work

The reported emissions are estimates subject to the accuracy of both Kepler's models and grid intensity data. As a result, the benchmark tests previously performed may not fully capture all possible scenarios, as grid dependency may sometimes force suboptimal migrations in the CO2-system as per resource availability. Resolution is limited by the update frequency of intensity sources, and storage requirements increase with sampling granularity.

We considered only a single baseline, defined as the mean $CO_2$ emissions across all tracked countries. While this provides a general reference point, it is not directly comparable to region-specific benchmarks and may obscure finer-grained differences. Future work should therefore incorporate multiple baselines, such as per-country averages or established benchmarks from the literature, and assess statistical significance relative to them.

Our benchmark scenarios were simplified to ensure reproducibility and interpretability. Although random sampling of countries illustrates the variability in energy mixes, it does not fully capture the operational constraints of datacenter migrations or multi-cloud scheduling. More complex benchmarks with realistic workloads and infrastructure heterogeneity would further validate the applicability of our approach.

Finally, while implementation details such as REST endpoints and TimescaleDB integration were reported for transparency, their evaluation was not the main focus of this study. Additional experimentation with scalability and deployment overhead would strengthen the case for adoption in production environments.

Future work will focus on service options to adjust granularity and tackle scalability issues within the service as well as broader evaluation.

## 6 Conclusion

We presented a Kubernetes-native $CO_2$ monitoring service that provides real-time emissions data through stable APIs. Evaluations demonstrate that when coupled with migration strategies, these metrics enable significant emission reductions, often surpassing KPI thresholds. Future work will include integration with more compute-intensive workloads, multi-source intensity aggregation, and cryptographic provenance for auditability.

## Acknowledgements

This work was supported by the FAME project under the European Union's Horizon Europe programme. We thank the Kepler community and colleagues who contributed feedback during testing.

For all online resources cited, the date of access has been included to ensure reproducibility and traceability.

## References

[1] CodeCarbon Project Contributors. 2025. Codecarbon: track and reduce the carbon footprint of your computing. Accessed: 25 September 2025. https://mlco2.github.io/codecarbon/.

[2] Electricity Maps ApS. 2025. Electricity maps: real-time carbon intensity of electricity consumption. Accessed: 25 September 2025. https://app.electricitymaps.com.

[3] European Union Horizon Europe Programme. 2025. Fame project: federated and multicloud enablers for green computing. https://www.fame-horizon.eu/the-project/. Accessed: 25 September 2025. (2025).

[4] Google. 2020. Our data centers now work harder when the sun shines and wind blows. Accessed: 25 September 2025. https://blog.google/inside-google/infrastructure/data-centers-work-harder-sun-shines-wind-blows/.

[5] Kepler Project Contributors. 2025. Kepler: kubernetes-based efficient power level exporter. Accessed: 25 September 2025. https://github.com/sustainable-computing-io/kepler.

[6] Scaphandre Project Contributors. 2025. Scaphandre: energy monitoring agent for linux servers. Accessed: 25 September 2025. https://github.com/hubblo-org/scaphandre.

[7] Timescale Inc. 2025. Timescaledb: an open-source time-series database. Accessed: 25 September 2025. https://www.timescale.com.