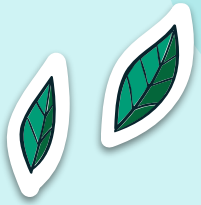
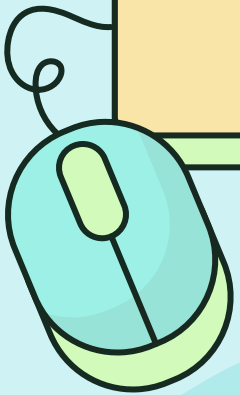
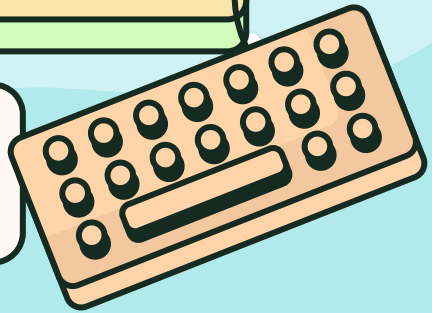
MONITORING CARBON EMISSIONS



making the impact of your python code visible



Jessica Greene & Chioma Onyekpere



JESSICA GREENE

She/Her

- Senior ML Engineer at Ecosia.org
- Pylady, community organiser, PySV board member, PSF Conduct WG member
- Career changer. Previously a Coffee roaster & Camera assistant.
- Knitter, gardener, Lego lover!

@sleepypioneer

CHIOMA ONYEKPERE

She/Her

- Software Engineer.
- From fear of “magic” dark screen with coloured text to becoming a magician.
- Tech collaborator. Happy to strengthen the underrepresented community through mentorship.
- Digital nomad

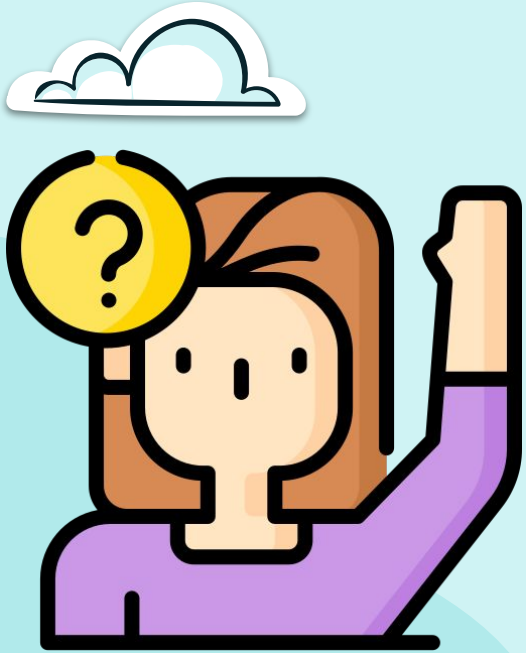
@simpcyclassy

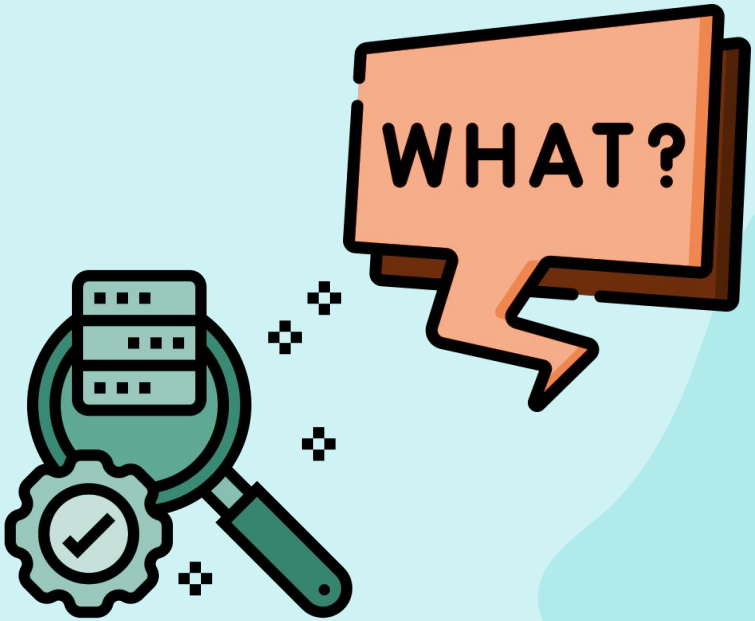
Agenda

1. What is Monitoring & Why is it important?
2. Overview of today's project repo
3. Exposing metrics with Prometheus Client
    a. **Challenge 1**: Expose base app metrics on /metrics endpoint
4. Adding custom metrics: What makes a meaningful metric?
    a. **Challenge 2**: Adding a custom metric with labels
5. Mid point QA and Break no.1
6. Inspecting metrics with Prometheus & Grafana
    a. **Challenge 3**: Query your metric with PromQL
    b. **Challenge 4:** Create a Grafana Dashboard
7. Carbon conscious development
    a. **Challenge 5**: Measure carbon emissions of our application
    b. **Challenge 6:** Adding carbon metrics to our Grafana Dashboard
8. Q&A

01

Monitoring: What is it?



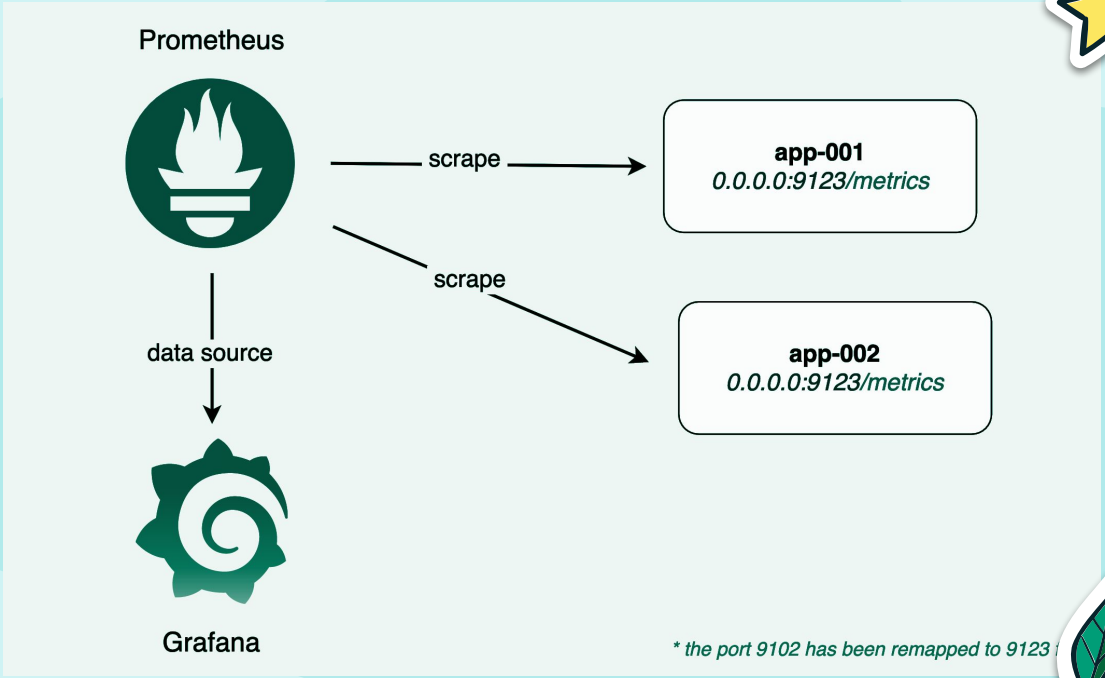


@sleepypioneer @simpcyclassy

Monitoring: Why is it important

@sleepypioneer @simpcyclassy

Monitoring: How?



@sleepypioneer @simpcyclassy

02

Git Repository

https://github.com/PyLadiesGermany/pycon-us-2024-workshop

@sleepypioneer @simpcyclassy

03 Challenge 1 - expose base metrics

1. Import **MetricsHandler** from the prometheus python client (**prometheus_client**)
2. Remember to remove **BaseHTTPRequestHandler**
3. Use the prometheus **MetricsHandler** as a base for your request handler
4. Once you've added the code, check it's working by running "make dev" and refresh the application a few times and check /metrics

https://github.com/PyLadiesGermany/pycon-us-2024-workshop

What is a metric? (base metrics)

```
# HELP process_cpu_seconds_total Total user and system CPU time spent in seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total 1.01
```

```
# HELP python_gc_collections_total Number of times this generation was collected
# TYPE python_gc_collections_total counter
python_gc_collections_total{generation="0"} 53.0
python_gc_collections_total{generation="1"} 4.0
python_gc_collections_total{generation="2"} 0.0
```

```
# HELP process_start_time_seconds Start time of the process since unix epoch in seconds.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1.60251632472e+09
```

@sleepypioneer @simpcyclassy

04 Defining Custom Metrics

- Useful when ‘out of the box’ metrics aren’t sufficient
- To define one, choose a data type and provide:
  - Base name
  - Description
  - Labels

```python
from prometheus_client import MetricsHandler, Counter, Histogram # type: ignore

HOST_NAME = "0.0.0.0"  # This will map to available port in docker
PORT_NUMBER = 8001
ELECTRICITY_MAP_API_KEY = getenv("ELECTRICITY_MAP_API_KEY")
ZONE = "DE"

# Prometheus counter to count number of requests by status and endpoint
requestCounter = Counter(
    "requests_total", "total number of requests", ["status", "endpoint"]
)


def fetch_carbon_intensity():
    tracker.start()
    r = (
        requests.get(
            carbon_intensity_url, auth=("auth-token", ELECTRICITY_MAP_API_KEY)
        )  # noqa
        if random.random() > 0.15
        else artificial_503()
    )
    requestCounter.labels(status=r.status_code, endpoint="/upstream").inc()
    _ = tracker.stop()
    if r.status_code == 200:
        return r.json()["carbonIntensity"]
    return 0
```

Defining Custom Metrics

- Useful when ‘out of the box’ metrics aren’t sufficient
- To define one, choose a data type and provide:
  - Base name
  - Description
  - Labels

```python
from prometheus_client import MetricsHandler, Counter, Histogram # type: ignore

HOST_NAME = "0.0.0.0"  # This will map to available port in docker
PORT_NUMBER = 8001
ELECTRICITY_MAP_API_KEY = getenv("ELECTRICITY_MAP_API_KEY")
ZONE = "DE"

# Prometheus counter to count number of requests by status and endpoint
requestCounter = Counter(
    "requests_total", "total number of requests", ["status", "endpoint"]
)


def fetch_carbon_intensity():
    tracker.start()
    r = (
        requests.get(
            carbon_intensity_url, auth=("auth-token", ELECTRICITY_MAP_API_KEY)
        )  # noqa
        if random.random() > 0.15
        else artificial_503()
    )
    requestCounter.labels(status=r.status_code, endpoint="/upstream").inc()
    _ = tracker.stop()
    if r.status_code == 200:
        return r.json()["carbonIntensity"]
    return 0
```

Defining Custom Metrics

- Useful when ‘out of the box’ metrics aren’t sufficient
- To define one, choose a data type and provide:
  - Base name
  - Description
  - Labels

```python
from prometheus_client import MetricsHandler, Counter, Histogram # type: ignore

HOST_NAME = "0.0.0.0"  # This will map to available port in docker
PORT_NUMBER = 8001
ELECTRICITY_MAP_API_KEY = getenv("ELECTRICITY_MAP_API_KEY")
ZONE = "DE"

# Prometheus counter to count number of requests by status and endpoint
requestCounter = Counter(
    "requests_total", "total number of requests", ["status", "endpoint"]
)


def fetch_carbon_intensity():
    tracker.start()
    r = (
        requests.get(
            carbon_intensity_url, auth=("auth-token", ELECTRICITY_MAP_API_KEY)
        )  # noqa
        if random.random() > 0.15
        else artificial_503()
    )
    requestCounter.labels(status=r.status_code, endpoint="/upstream").inc()
    _ = tracker.stop()
    if r.status_code == 200:
        return r.json()["carbonIntensity"]
    return 0
```

Defining Custom Metrics

- Useful when ‘out of the box’ metrics aren’t sufficient
- To define one, choose a data type and provide:
  - Base name
  - Description
  - Labels

```python
from prometheus_client import MetricsHandler, Counter, Histogram # type: ignore

HOST_NAME = "0.0.0.0"  # This will map to available port in docker
PORT_NUMBER = 8001
ELECTRICITY_MAP_API_KEY = getenv("ELECTRICITY_MAP_API_KEY")
ZONE = "DE"

# Prometheus counter to count number of requests by status and endpoint
requestCounter = Counter(
    "total number of requests", ["status", "endpoint"]
)


def fetch_carbon_intensity():
    tracker.start()
    r = (
        requests.get(
            carbon_intensity_url, auth=("auth-token", ELECTRICITY_MAP_API_KEY)
        )  # noqa
        if random.random() > 0.15
        else artificial_503()
    )
    requestCounter.labels(status=r.status_code, endpoint="/upstream").inc()
    _ = tracker.stop()
    if r.status_code == 200:
        return r.json()["carbonIntensity"]
    return 0
```

Defining Custom Metrics

- Useful when ‘out of the box’ metrics aren’t sufficient
- To define one, choose a data type and provide:
  - Base name
  - Description
  - Labels

```python
from prometheus_client import MetricsHandler, Counter, Histogram # type: ignore

HOST_NAME = "0.0.0.0"  # This will map to available port in docker
PORT_NUMBER = 8001
ELECTRICITY_MAP_API_KEY = getenv("ELECTRICITY_MAP_API_KEY")
ZONE = "DE"

# Prometheus counter to count number of requests by status and endpoint
requestCounter = Counter(
    "requests_total", "Total number of requests", ["status", "endpoint"]
)


def fetch_carbon_intensity():
    tracker.start()
    r = (
        requests.get(
            carbon_intensity_url, auth=("auth-token", ELECTRICITY_MAP_API_KEY)
        )  # noqa
        if random.random() > 0.15
        else artificial_503()
    )
    requestCounter.labels(status=r.status_code, endpoint="/upstream").inc()
    _ = tracker.stop()
    if r.status_code == 200:
        return r.json()["carbonIntensity"]
    return 0
```

@sleepypioneer @simpcyclassy

Defining Custom Metrics

- Useful when ‘out of the box’ metrics aren’t sufficient
- To define one, choose a data type and provide:
  - Base name
  - Description
  - Labels

```python
from prometheus_client import MetricsHandler, Counter, Histogram # type: ignore

HOST_NAME = "0.0.0.0"  # This will map to available port in docker
PORT_NUMBER = 8001
ELECTRICITY_MAP_API_KEY = getenv("ELECTRICITY_MAP_API_KEY")
ZONE = "DE"

# Prometheus counter to count number of requests by status and endpoint
requestCounter = Counter(
    "requests_total", "total number of requests", ["status", "endpoint"]
)


def fetch_carbon_intensity():
    tracker.start()
    r = (
        requests.get(
            carbon_intensity_url, auth=("auth-token", ELECTRICITY_MAP_API_KEY)
        )  # noqa
        if random.random() > 0.15
        else artificial_502()
    )
    requestCounter.labels(status=r.status_code, endpoint="/upstream").inc()
    _ = tracker.stop()
    if r.status_code == 200:
        return r.json()["carbonIntensity"]
    return 0
```

How Custom Metrics Look

```
# HELP requests_total Total number of requests
# TYPE requests_total counter
requests_total{endpoint="/carbon_intensity",status="200"} 1.0
```

Description

How Custom Metrics Look

```
# HELP requests_total Total number of requests
# TYPE requests_total counter
requests_total{endpoint="/carbon_intensity",status="200"} 1.0
```

Measurement type

How Custom Metrics Look

```
# HELP requests_total Total number of requests
# TYPE requests_total counter
requests_total{endpoint="/carbon_intensity",status="200"} 1.0
```

How Custom Metrics Look

```
# HELP requests_total Total number of requests
# TYPE requests_total counter
requests_total{endpoint="/carbon_intensity",status="200"} 1.0
```

Labels

How Custom Metrics Look

```
# HELP requests_total Total number of requests
# TYPE requests_total counter
requests_total{endpoint="/carbon_intensity",status="200"} 1.0
```

Metric value

Challenge 2 - custom metrics

1. Add a custom counter metric. It will count requests with the status code as a label.
2. Re-run the server by stopping it and then restarting it.
3. Refresh the main application page (/carbon_intensity) a few times and check /metrics

https://github.com/PyLadiesGermany/pycon-us-2024-workshop

05

QUESTIONS SO FAR?

06

Querying Metrics & building Dashboards

@sleepypioneer @simpcyclassy

Monitoring your metics

Prometheus

http://localhost:9090

Our Application

http://localhost:8001

Grafana

http://localhost:3000

docker-compose up

@sleepypioneer @simpcyclassy

Challenge 3 - PromQL queries

1. Run the app and prometheus by using the docker-compose command
2. Query your metrics in the prometheus UI using PromQL

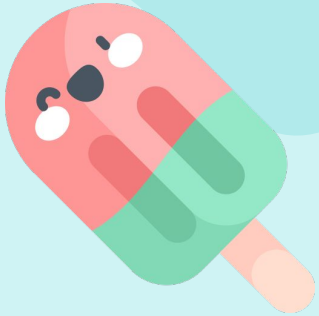
https://github.com/PyLadiesGermany/pycon-us-2024-workshop

Challenge 4 - build a dashboard

1. Run the app, prometheus and Grafana with “docker-compose up --build”
2. Go to localhost:3000
3. Create a Grafana Dashboard
4. Create a panel to visualise your custom metric (split by labels, irate or cumulative)

https://github.com/PyLadiesGermany/pycon-us-2024-workshop





Break (30 minutes)

07

Carbon Conscientious Development

@sleepypioneer @simpcyclassy

“The energy used in our digital consumption collectively emits the equivalent amount of carbon as the entire airline industry.”

Lindsay Vaughan,

Former Climate Impact Partners CEO

@sleepypioneer @simpcyclassy

Green Coding Principles

—Energy Efficient

Consume the least amount you can

—Hardware efficient

Do the most you can with the smallest amount of compute and use it longer

—Carbon awareness

Do more when energy is clean and less when it's burning fossil fuels

@sleepypioneer @simpcyclassy

Understanding your energy source

Fossil Fuels

Produces carbon emissions

Clean energy

Doesn't produce carbon emissions e.g. nuclear.

Green energy

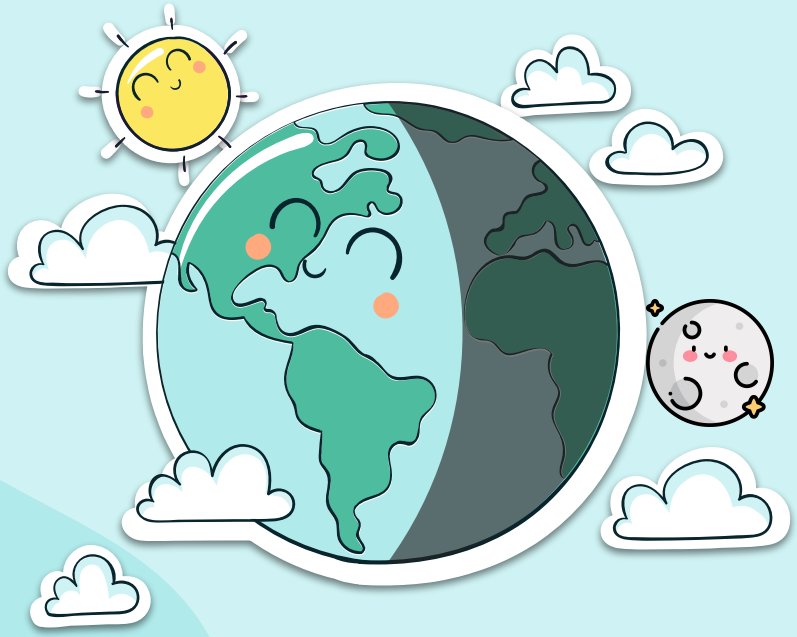
Sources from nature

Renewable energy

Sources will not expire e.g. solar, wind

@sleepypioneer @simpcyclassy

Demand Shifting



@sleepypioneer @simpcyclassy

Allocating emissions

Scope 1

Emissions from burning fossil fuels to make our products

Scope 2

Emissions from electricity generated on our behalf, to run our products

Scope 3

Emissions from activity in our supply chain

Software Carbon Intensity (SCI) Specification

- Reducing an SCI score is only possible through the elimination of emissions.
- It is possible to calculate an SCI score for any software application, from a large, distributed cloud system to a small monolithic open source library, any on-premise application, or even a serverless function.

SCI github

@sleepypioneer @simpcyclassy

08

Measuring Emissions in our Code

"If you can't measure it, you can't improve it." - Peter Drucker

@sleepypioneer @simpcyclassy

Code Carbon

AI can benefit society in many ways but, given the energy needed to support the computing behind AI, these benefits can come at a high environmental price.

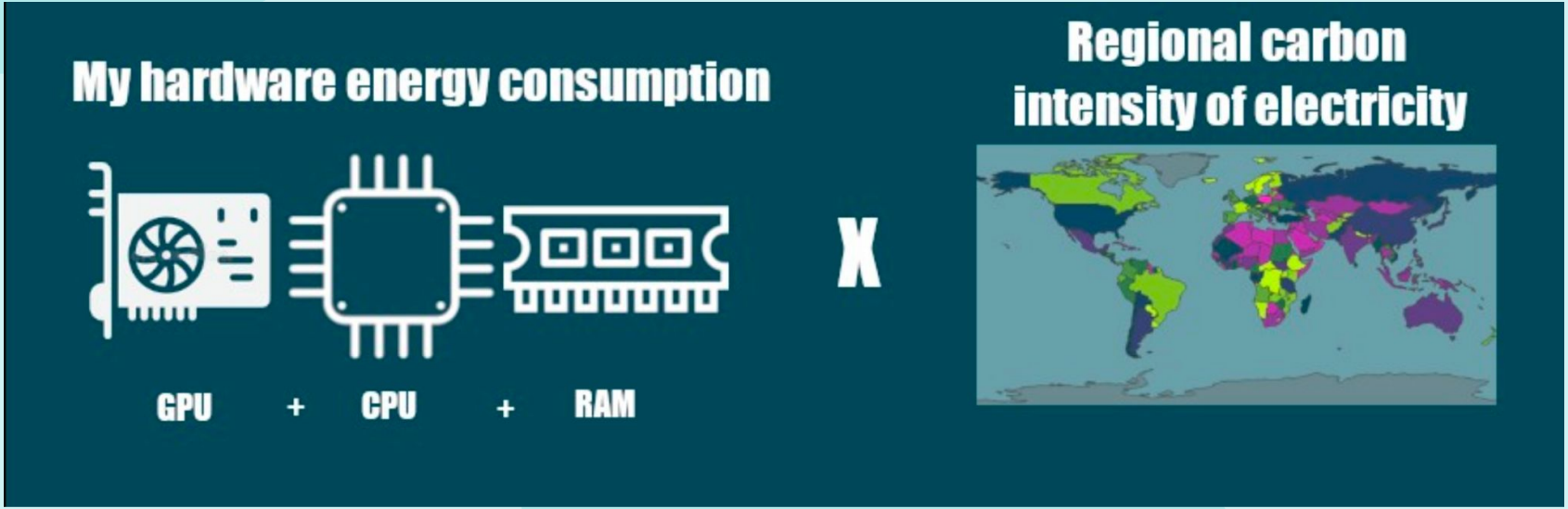
CodeCarbon is a lightweight software package that seamlessly integrates into your Python codebase. It estimates the amount of carbon dioxide (CO2) produced by the cloud or personal computing resources used to execute the code.

It then shows developers how they can lessen emissions by optimizing their code or by hosting their cloud infrastructure in geographical regions that use renewable energy sources

https://codecarbon.io/

@sleepypioneer @simpcyclassy

Measure the power consumption and estimate the C02 equivalent



https://mlco2.github.io/codecarbon/

Challenge 5 - carbon metrics

1. Import EmissionsTracker from the CodeCarbon library (**from codecarbon import EmissionsTracker**)

2. Declare your Tracker and point to the Prometheus gateway

```
tracker = EmissionsTracker(
   project_name="python-app",
   save_to_prometheus=True,
   prometheus_url="http://pushgateway:9091",
)
```

3. Start (and stop) the track in your code

https://github.com/PyLadiesGermany/pycon-us-2024-workshop

Challenge 6 - visualising emissions

1. Run the app, prometheus and Grafana with “docker-compose up --build”
2. Go to localhost:3000
3. Create a new Grafana Dashboard
4. Create a panel to visualise your carbon metrics (they will be prefixed with *codecarbon_)*

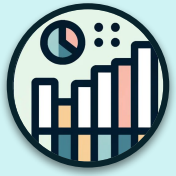
https://github.com/PyLadiesGermany/pycon-us-2024-workshop

Joining the green code movement!

Build climate impact into your day to day work and make the world a better place one line of code at a time

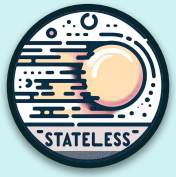
@sleepypioneer @simpcyclassy

Green coding tips



Avoid tracking unnecessary data

Avoid an excessive DOM size



Implement stateless design

Defer offscreen images

Optimise storage & CPU utilization

Deprecate GIFs for animated content

https://patterns.greensoftware.foundation/tags/role-software-engineer

@sleepypioneer @simpcyclassy

Efficient Containerization

Docker files

```
FROM ubuntu:latest
COPY . .
ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update -y
RUN apt-get upgrade -y
RUN apt-get install vim -y
RUN apt-get install net-tools -y
RUN apt-get install dnsutils -y
```

green-coding-is-a-matter-of-code-quality

@sleepypioneer @simpcyclassy

Efficient Containerization

Docker files

Use smaller base images

Copy over code that will change last (to utilise caching)

```
FROM ubuntu:latest
COPY . .
ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update -y
RUN apt-get upgrade -y
RUN apt-get install vim -y
RUN apt-get install net-tools -y
RUN apt-get install dnsutils -y
```

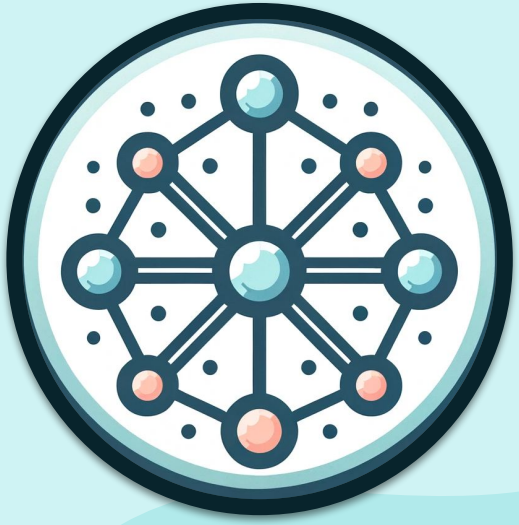
Run imports first to utilise caching & run them together

green-coding-is-a-matter-of-code-quality

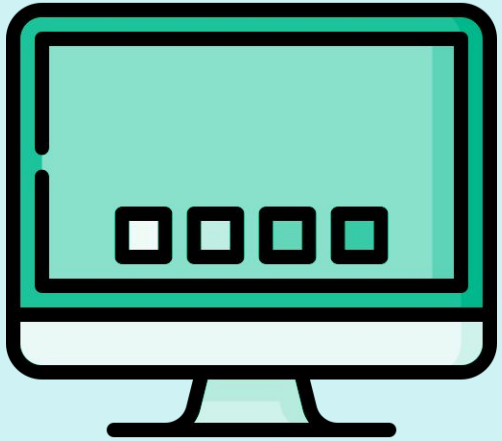
@sleepypioneer @simpcyclassy

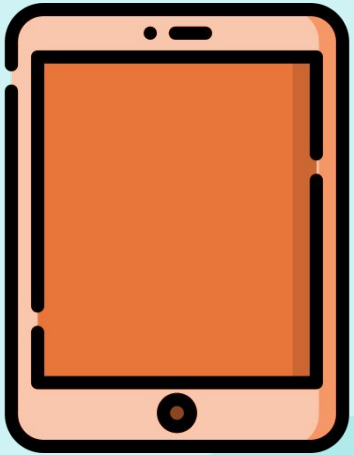
Network Efficiency

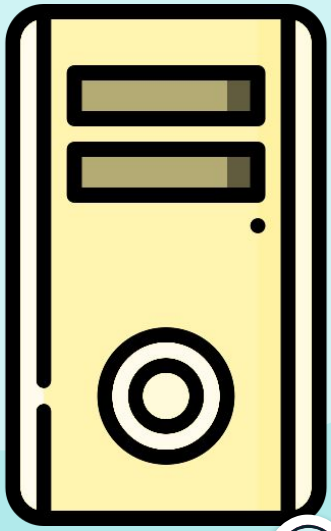
Each connected device consumes electricity, we aim therefore to reduce the amount of data sent and the distance it needs to travel across the network.



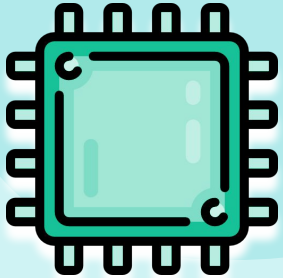
Embodied carbon

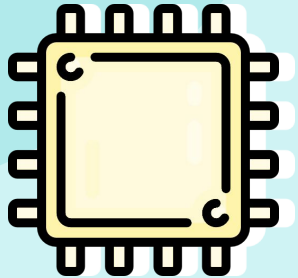




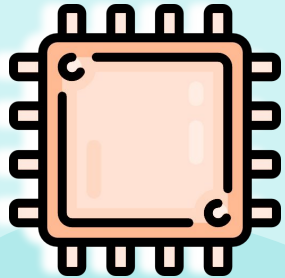


Hardware efficiency





The hardware your software is running on uses some of the electricity for operational overhead. This is called power usage efficiency (PUE) in the cloud space.



@sleepypioneer @simpcyclassy

Resources

- https://prometheus.io
- https://prometheus.io/docs/practices/histograms/
- https://grafana.com/
- https://tomgregory.com/the-four-types-of-prometheus-metrics/
- https://github.com/ecosia/pycon22-prometheus-workshop
- https://codecarbon.io/
-

kube-green

A Project by Davide Bianchi

@sleepypioneer @simpcyclassy

08

Q&A

Thanks!

@sleepypioneer @simpcyclassy