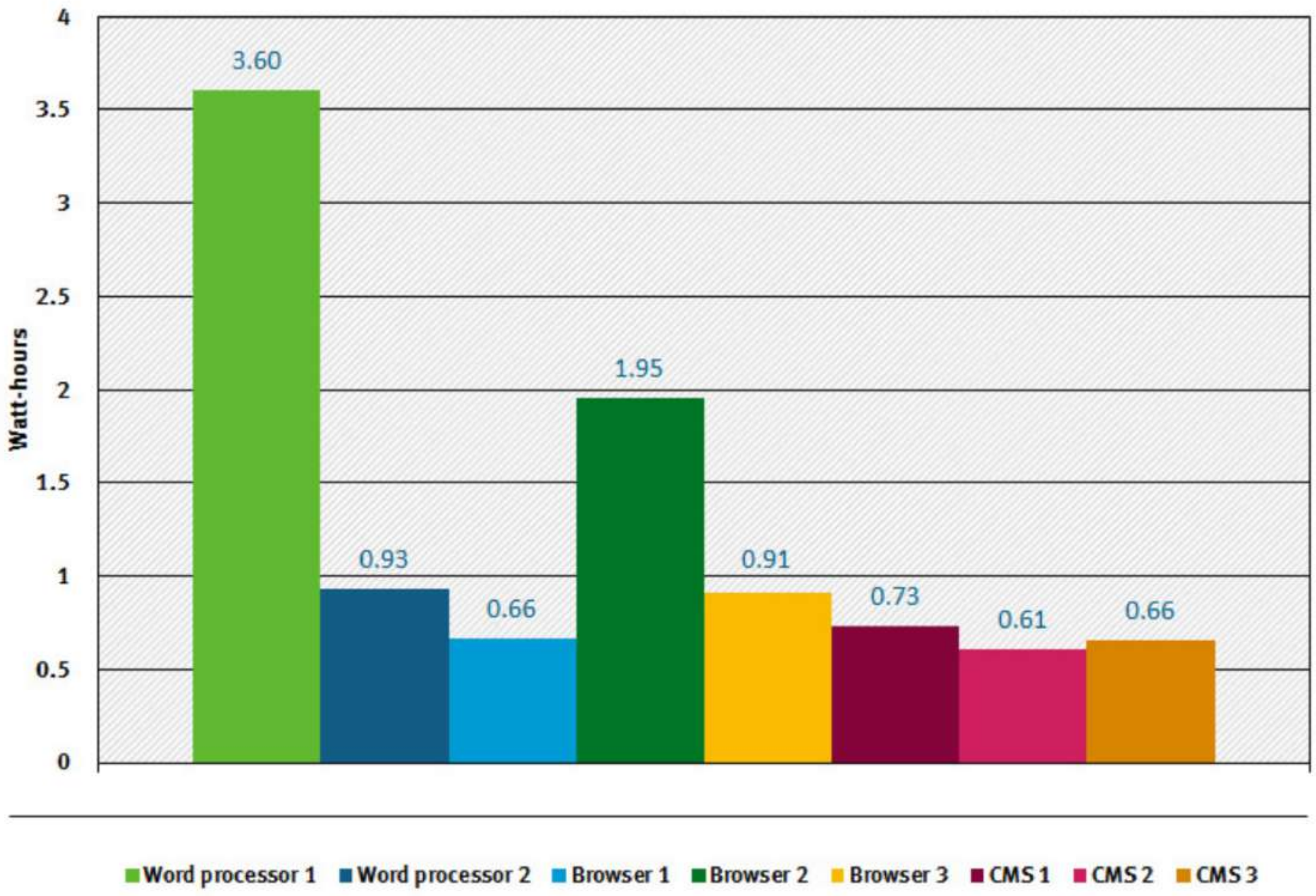
MAIBORNWOLFF

DevSusOps

Jochen Joswig

MAIBORNWOLFF

Energy consumed per use case scenario



[Gröger]

Questions

Questions!

How do I know how good I am?

How do I measure?

Which tools should I use?

How accurate is an approach?

How do I find potential optimization?

Who can impact my products sustainability?

Who is responsible?

How much effort is it?

How do I get management on board?

How do I get my team on board?

What do we have to do differently?

We are committed now what?

Do I need new processes?

What can XYZ do?

Does it really help?

Is it worth it?

Where can I find more information?

How do I raise awareness?

How can I educate my staff?

Are there certifications and how do I get certified?

My Qualifications

Jochen Joswig

With MaibornWolff since June 2020
Senior Full Stack Engineer & Project Lead

DevOps & Cloud-Native Department
Azure Cloud, Development, CI/CD, GreenOps

Topics
Client Work, Green in IT, R&D, Schools & Workshops

Volunteering
Green Software Foundation Champion, CNCF TAG env. Project Lead,
Green Software Development Manifesto Co-Author & Community Orga

Contact
jochen.joswig@maibornwolff.de

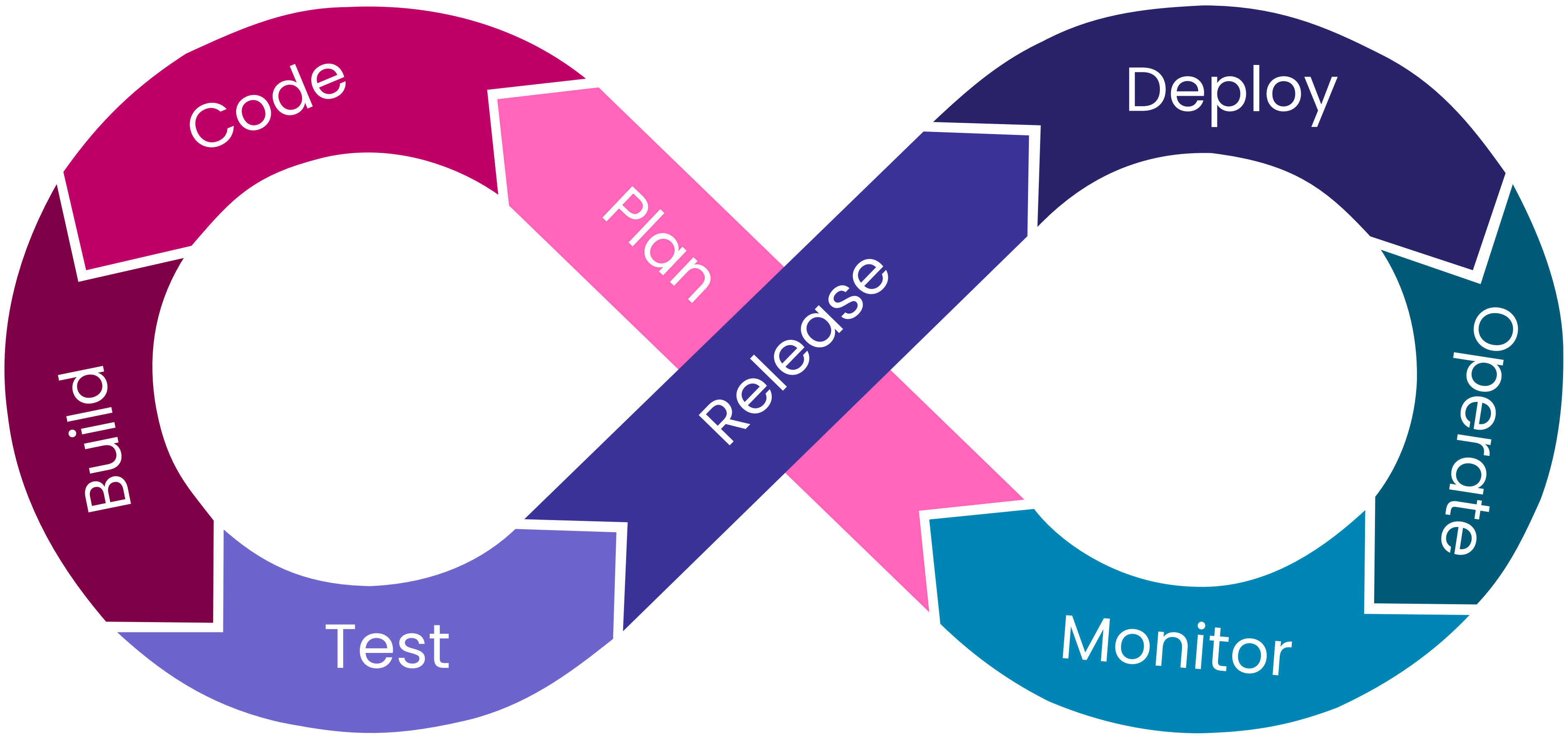
Challenges

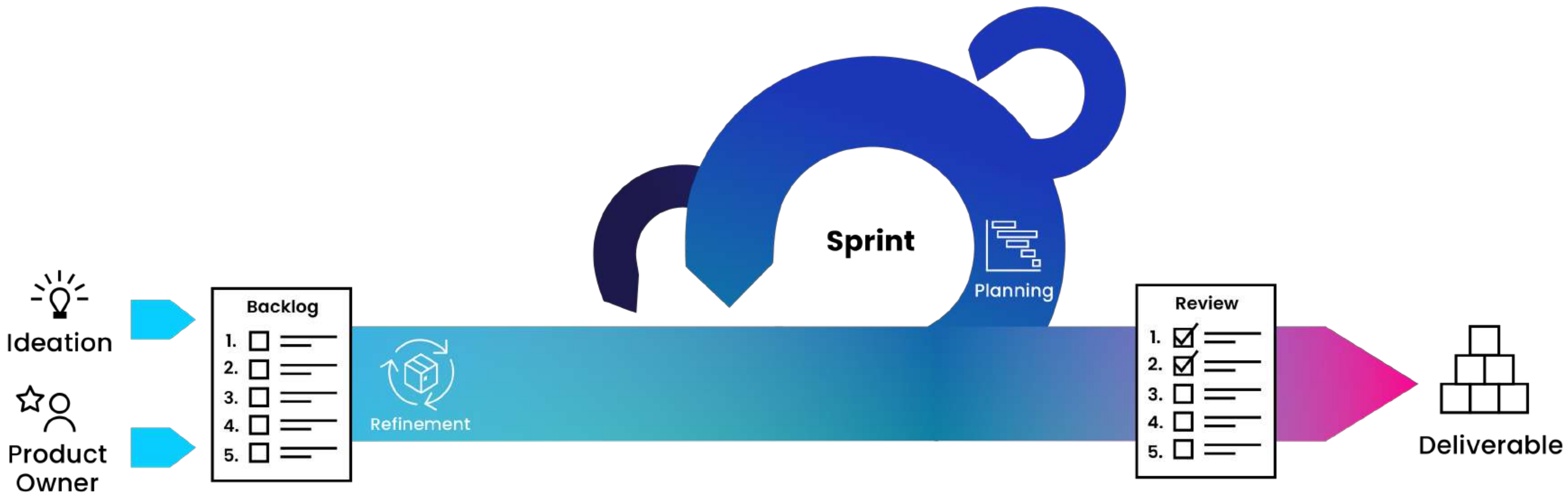
The challenge of finding a good approach

- Must be easy to rollout and apply
- Must be predictable in outcome and cost
- Must be fast and agile
- Must be flexible and adaptable
- Must ensure other common software quality criteria
- Must facilitate knowledge exchange
- Must ensure sustainability optimization

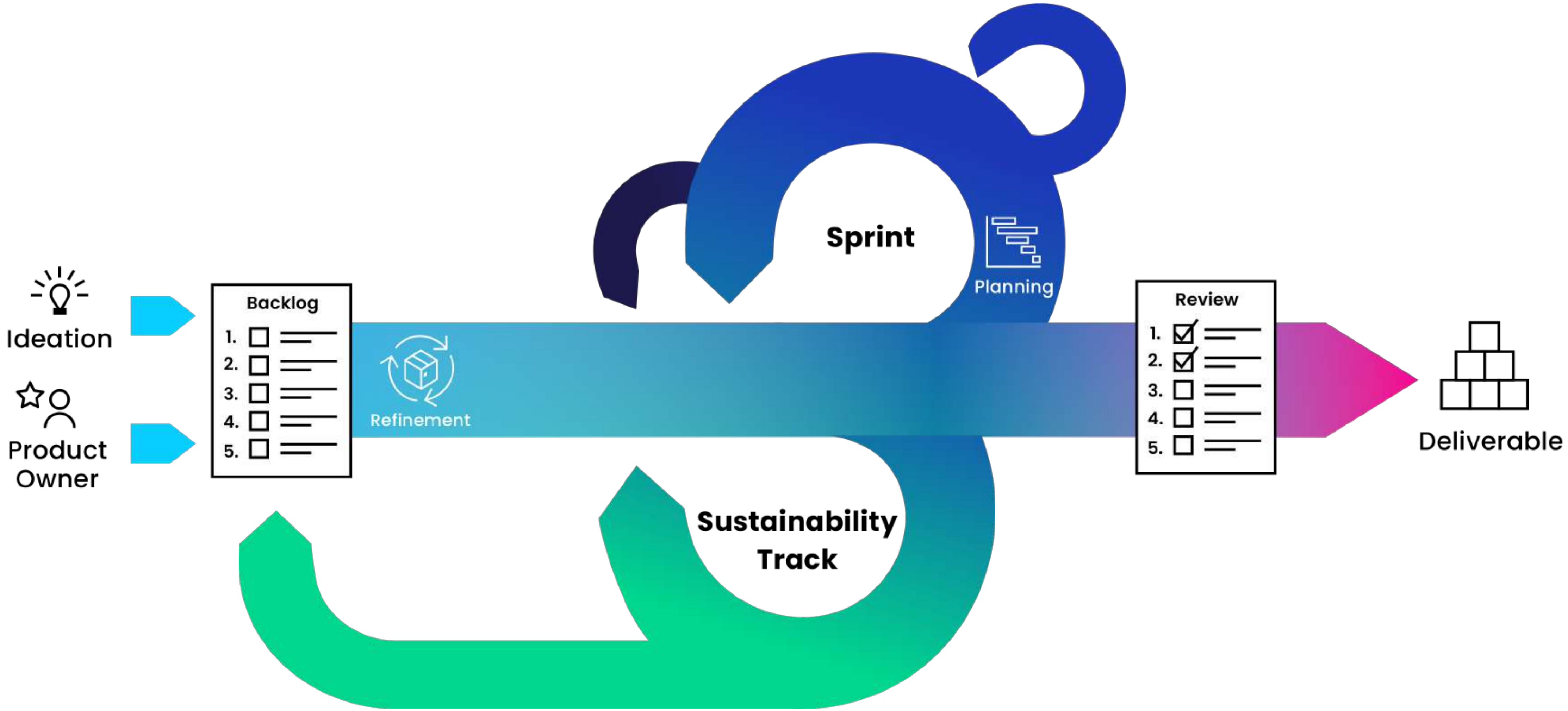
State of the Art

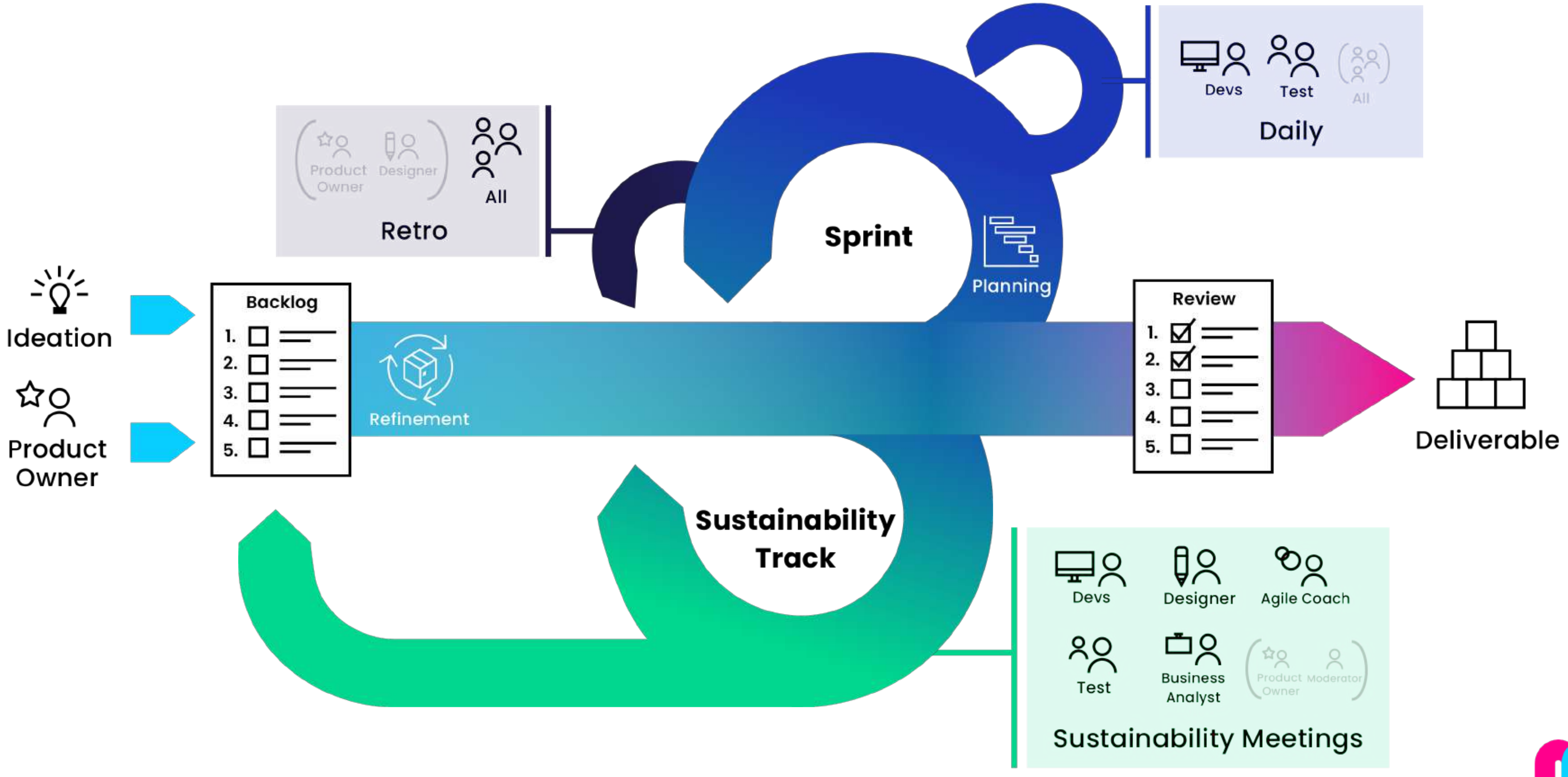
DevOps





Ideation
Product Owner
Backlog
1.
2.
3.
4.
5.
Refinement
Sprint
Planning
Review
1.
2.
3.
4.
5.
Deliverable


Ideation
Product Owner
Backlog
1.
2.
3.
4.
5.
Refinement
Sprint
Planning
Sustainability Track
Review
1.
2.
3.
4.
5.
Deliverable



Product Owner
Designer
All
Retro
Devs
Test
All
Daily
Sprint
Planning
Ideation
Product Owner
Backlog
1.
2.
3.
4.
5.
Refinement
Review
1.
2.
3.
4.
5.
Deliverable
Sustainability Track
Devs
Designer
Agile Coach
Test
Business Analyst
Product Owner
Moderator
Sustainability Meetings

Code

Code - Recommendation

Handbook of sustainable design of digital services

Strategy

The project strategy stage makes it possible to determine the relevance and the challenges of the project.

Specifications

The specifications group together the elements of the project framework, the means implemented, the objectives and constraints of the project over the entire lifespan of the target product. Regardless of the type of project management: AGILE or...

Ux/Ui

The stages and methods of designing digital services to define the best solutions for interactions with the user.

Contents

All elements of a digital service available to the end user.

Frontend

The stages and methods of designing digital services to define the best solutions for interactions with the user.

Architecture

It defines all the typologies of common technical service components which are interposed between the application components and the hardware components to manage these physical resources: components for managing local technical resources (OS, ...

Backend

The backend represents the computer translation of business processes, the technical means and data implemented for their use, as well as all the external interactions implemented for their realization.

Hosting

Allowing remote users to use a digital service.

[GR491, The Handbook of sustainable design of digital services | ISIT]()

Azure Well-Architected Framework

Sustainability workload documentation

In partnership with the Green Software Foundation, we've developed this set of recommendations for optimizing Azure workloads. This documentation helps you plan your path forward, improve your sustainability posture, and create new business value while reducing your operational footprint.

[Sustainability workload documentation - Microsoft Azure Well-Architected Framework | Microsoft Learn]

Code - Recommendation

AWS Well-Architected Framework

Sustainability Pillar - AWS Well-Architected Framework

PDF | RSS

Publication date: **October 3, 2023** (Document revisions)

This whitepaper focuses on the sustainability pillar of the Amazon Web Services (AWS) Well-Architected Framework. It provides design principles, operational guidance, best-practices, potential trade-offs, and improvement plans you can use to meet sustainability targets for your AWS workloads.

Sustainability Pillar - AWS Well-Architected Framework - Sustainability Pillar (amazon.com)

Code - Recommendation

GSF Patterns

Green Software Patterns

Summary

An online open-source database of software patterns reviewed and curated by the Green Software Foundation across a wide range of categories. You can be confident that applying any of our published and live patterns will reduce your software emissions.

Any software practitioner can find the patterns related to their field, technology, or domain. Anyone can submit a pattern that triggers a detailed review process by the Foundation.

[Green Software Patterns | Green Software Patterns]

Code - Recommendation

Blue Angle

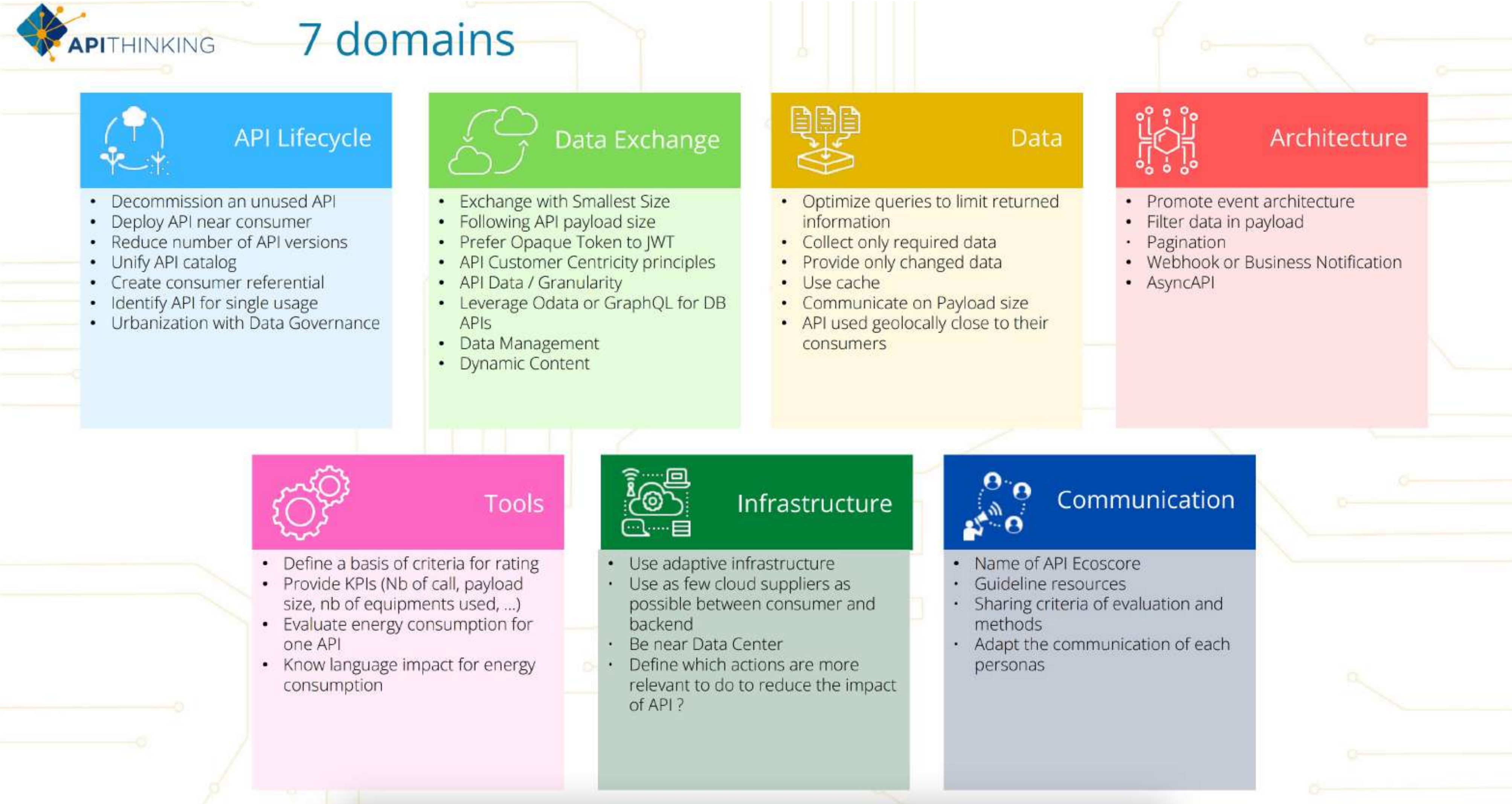
Resource and Energy-Efficient Software Products

DE-UZ 215

DE-UZ 215-202001-en-Criteria-V3.pdf (blauer-engel.de)

Code - Recommendation

Sustainable API Design



[CAT_API_Sustainable_IT.pptx (collectif-api-thinking.com)]

Code – Tools

Static Code Analysis

Greenspector

ecoCode

How green is your code?

Example 1

Total

	Energy (J)
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time (ms)
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

	Mb
(c) Pascal	1.00
(c) Go	1.05
(c) C	1.17
(c) Fortran	1.24
(c) C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	2.71
(i) Python	2.80
(c) Ocaml	2.82
(v) C#	2.85
(i) Hack	3.34
(v) Racket	3.52
(i) Ruby	3.97
(c) Chapel	4.00
(v) F#	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84

[Pereira]

Example 2

				Difference (%)	
Pattern	System	Overall Time	Energy (J)	Time	Energy
Facade	"Clean"	15,40	395,60	1,80	2,50
	Pattern	15,70	405,60		
Abstract Factory	"Clean"	13,50	342,10	14,20	15,90
	Pattern	15,40	396,60		
Observer	"Clean"	15,10	373,70	0,90	0,10
	Pattern	15,20	373,90		
Decorator	"Clean"	15,20	374,00	132,40	133,60
	Pattern	35,40	873,80		
Prototype	"Clean"	11,20	271,80	33,00	33,20
	Pattern	14,90	362,00		
Template Method	"Clean"	15,00	366,40	0,30	0,10
	Pattern	15,10	366,70		

[Bunse]

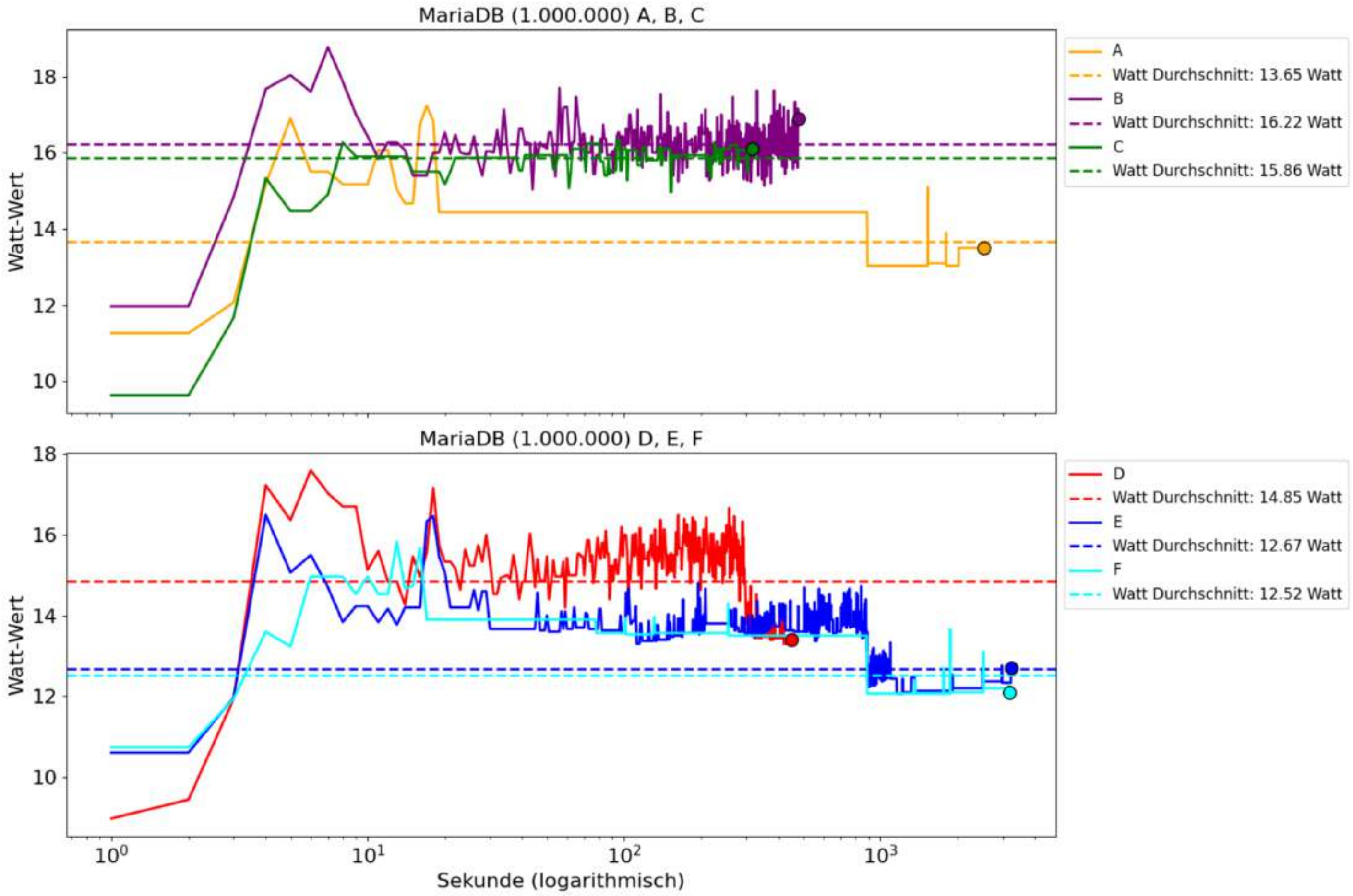
Example 3

Gains are possible: The experiments with Orange and Greenspector© have shown that energy and resource gains were possible. In return of some corrections, we have obtained between 5 and 7% of gain in energy and other resources. As the workload to work on the correction was limited, we estimate we can have more gain.

[Vautier]

Code

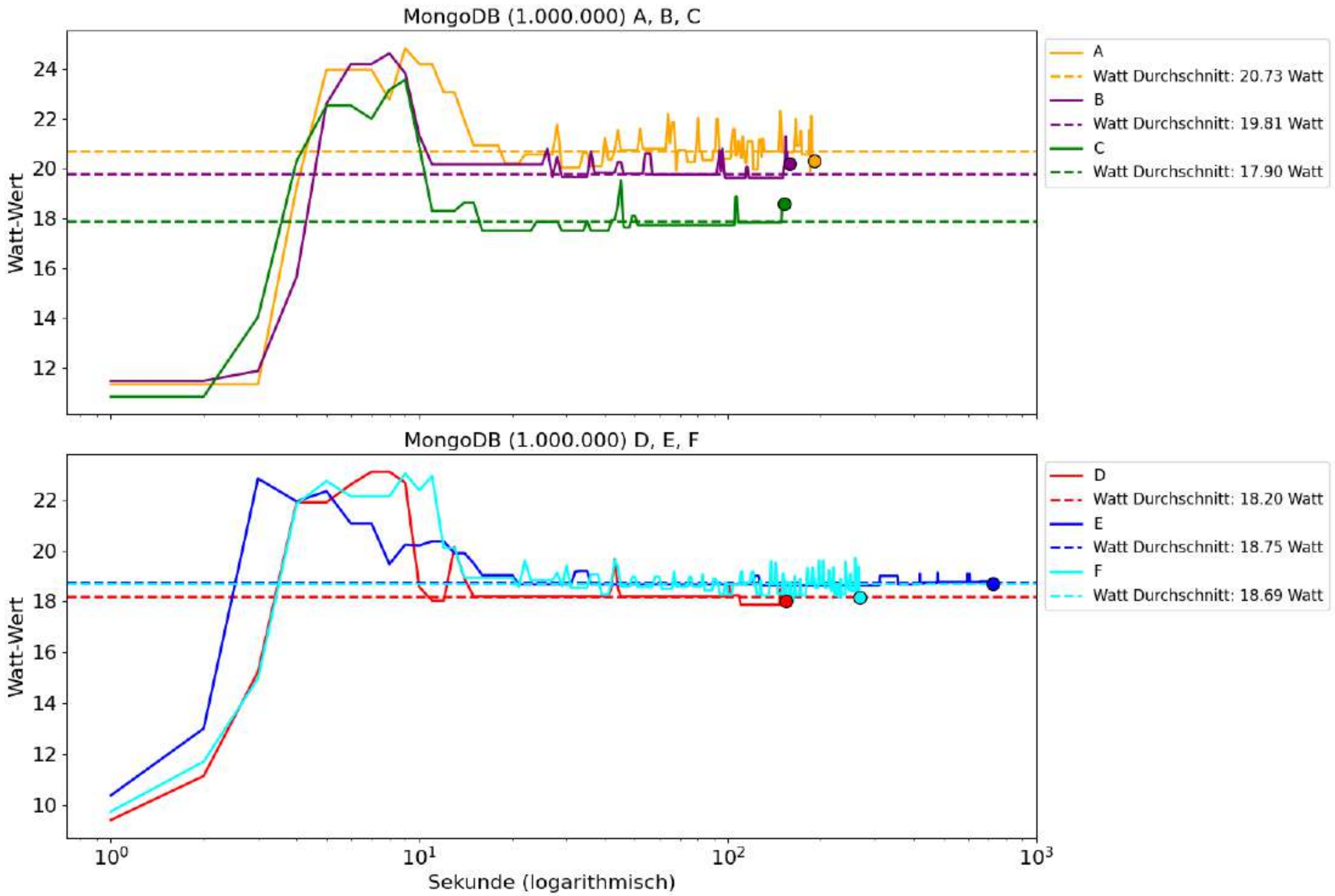
Example 4



[Jonuzi]

Code

Example 4



[Jonuzi]

Code

Example 4

Workload A: 50 % READ, 50 % UPDATE

Tabelle 5.3: Workload A - Auswertung

A	**Maria 100k**	**Maria 1M**	**Mongo 100k**	**Mongo 1M**
ShellyPlug in Wh	0,4377	9,3284	0,1259	1,0855
Watt durchschn.	14,6192	13,5523	20,5203	20,7138
RAPL in Wh	0,0785	1,5741	0,0427	0,338
Laufzeit in s	108,1357	2478,771	22,088	188,658
Durchsatz in op/s	924,78	410,5233	4528,617	5301,593
READ Latenz in µs	571,05	1085,573	199,55	176,1733
UPDATE Latenz in µs	1563,69	3845,2867	220,6033	194,89

Workload C: 100 % READ

Tabelle 5.5: Workload C - Auswertung

C	**Maria 100k**	**Maria 1M**	**Mongo 100k**	**Mongo 1M**
Shelly Plug in Wh	0,0797	1,3963	0,1019	0,7559
Watt durchschn.	17,968	15,8658	19,615	17,8871
RAPL in Wh	0,0261	0,3908	0,0371	0,2546
Laufzeit in s	15,966	316,8567	18,6927	152,0557
Durchsatz in op/s	6265,99	3156,4067	5350,29	6577,05
READ Latenz in µs	155,52	314,54	177,2833	149,6667

[Jonuzi]

Build

Build

best practices

- Minify
- Compress
- Build changed code only

Slim toolkit

You can find the examples in a separate repository: https://github.com/slimtoolkit/examples

Node.js application images:

- from ubuntu:14.04 - 432MB => 14MB (minified by **30.85X**)
- from debian:jessie - 406MB => 25.1MB (minified by **16.21X**)
- from node:alpine - 66.7MB => 34.7MB (minified by **1.92X**)
- from node:distroless - 72.7MB => 39.7MB (minified by **1.83X**)

Python application images:

- from ubuntu:14.04 - 438MB => 16.8MB (minified by **25.99X**)
- from python:2.7-alpine - 84.3MB => 23.1MB (minified by **3.65X**)
- from python:2.7.15 - 916MB => 27.5MB (minified by **33.29X**)
- from centos:7 - 647MB => 23MB (minified by **28.57X**)
- from centos/python-27-centos7 - 700MB => 24MB (minified by **29.01X**)
- from python2.7:distroless - 60.7MB => 18.3MB (minified by **3.32X**)

Ruby application images:

- from ubuntu:14.04 - 433MB => 13.8MB (minified by **31.31X**)
- from ruby:2.2-alpine - 319MB => 27MB (minified by **11.88X**)
- from ruby:2.5.3 - 978MB => 30MB (minified by **32.74X**)

Go application images:

- from golang:latest - 700MB => 1.56MB (minified by **448.76X**)
- from ubuntu:14.04 - 531MB => 1.87MB (minified by **284.10X**)
- from golang:alpine - 258MB => 1.56MB (minified by **165.61X**)
- from centos:7 - 615MB => 1.87MB (minified by **329.14X**)

Rust application images:

- from rust:1.31 - 2GB => 14MB (minified by **147.16X**)

Java application images:

- from ubuntu:14.04 - 743.6 MB => 100.3 MB

[GitHub - slimtoolkit/slim: Slim(toolkit)](https://github.com/slimtoolkit/slim)

Eco CI

[Eco CI | green-coding.io](https://www.green-coding.io)

Compression

In the results of the table we can see, that the network carbon emission savings through compression of **115.13 J** and **3412.8 J** are way higher than the compression & decompress costs of **~10 J** ad **~0.5 J**

Does gzip save CO2? | green-coding.io

Test

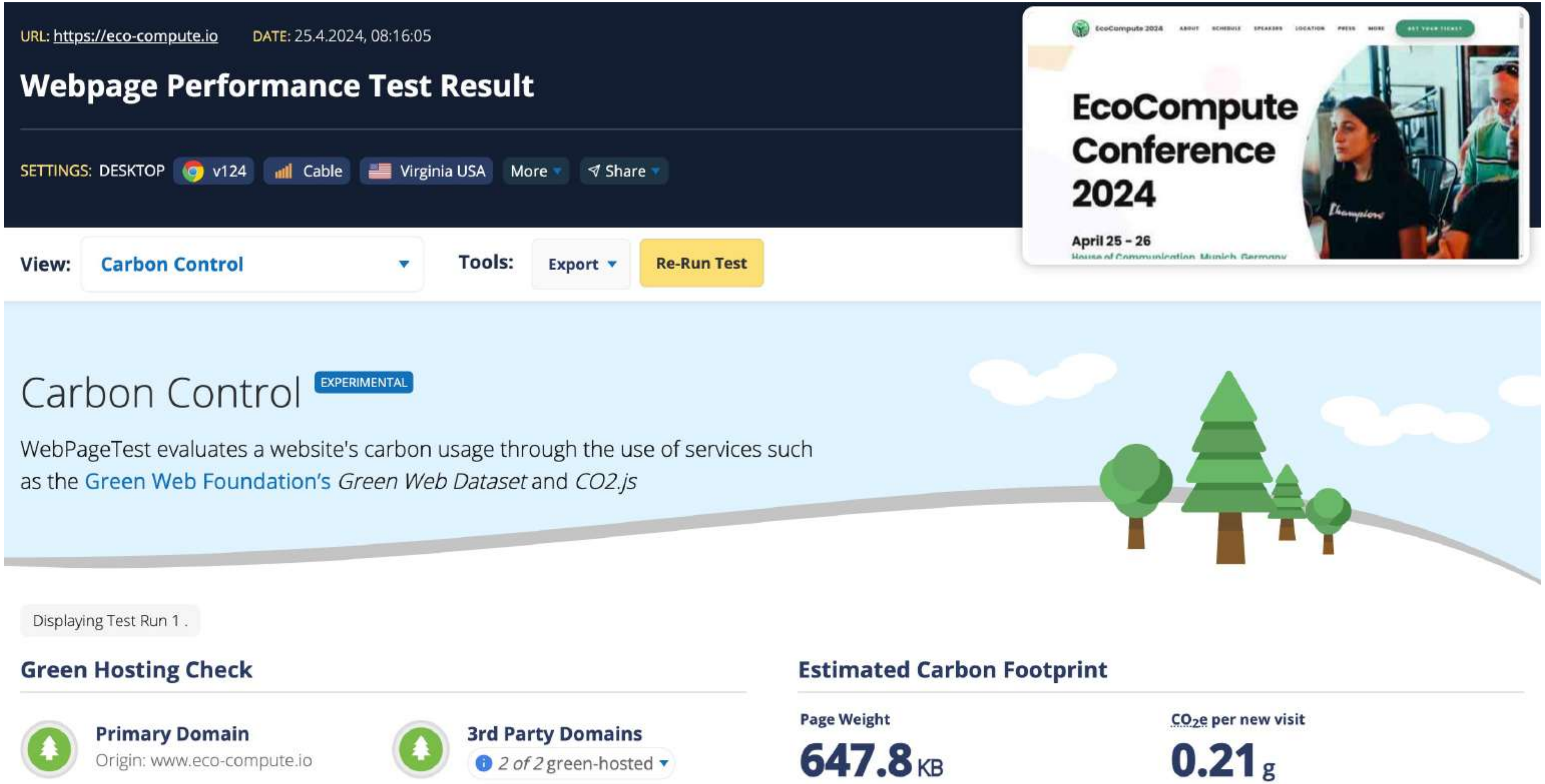
Test

best practices

- Only test changed code
- Turn off Dev/Test environments when no one is working
- Test environmental factors

Test - Tools

Web Tester



[WebPageTest - Website Performance and Optimization Test]()

Test - Tools

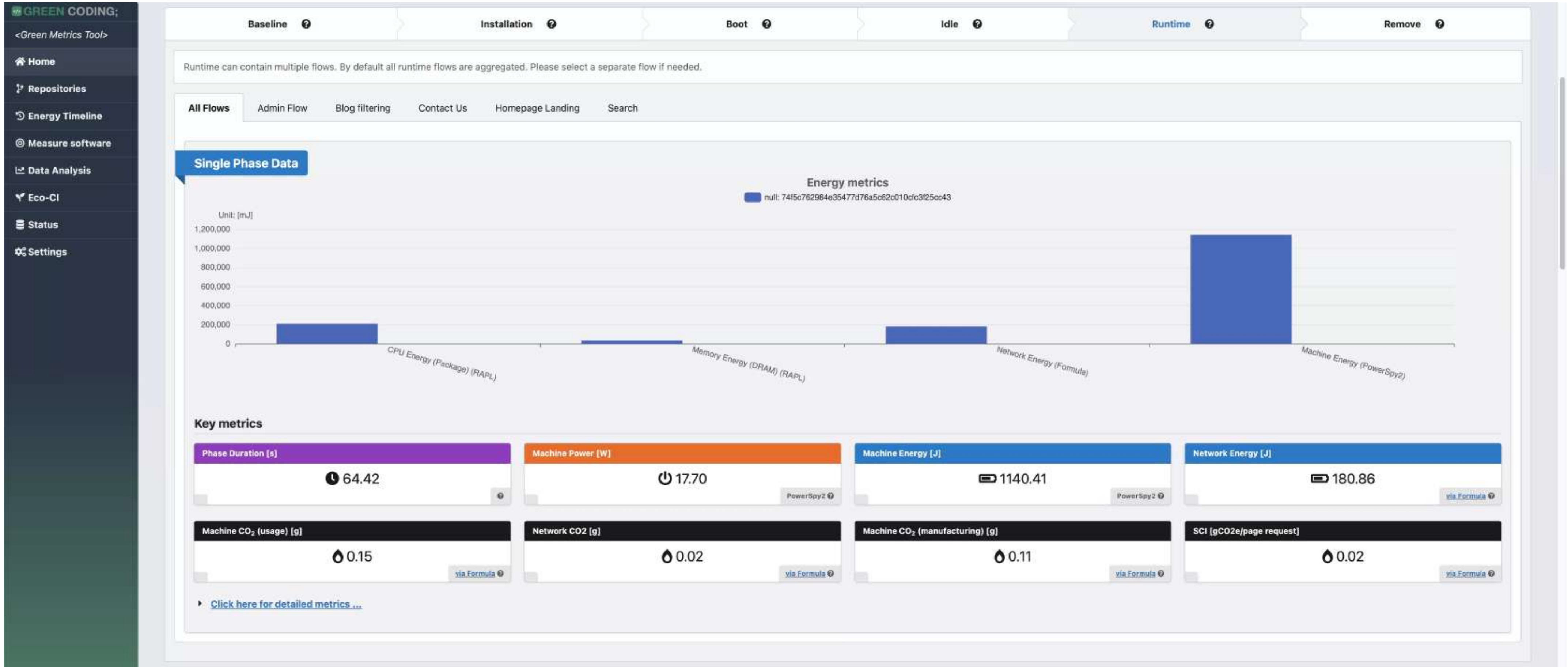
kube-green

kube-green

A Project by Davide Bianchi

[GitHub - kube-green/kube-green: A K8s operator to reduce CO2 footprint of your clusters]()

Green Metrics Tool



[Green Metrics Tool | green-coding.io]

Release

Release

best practices

- Allow Configuration and set Eco-friendly defaults
- Have an end-of-life strategy
- Ensure support for old hardware
- Clean up left-overs

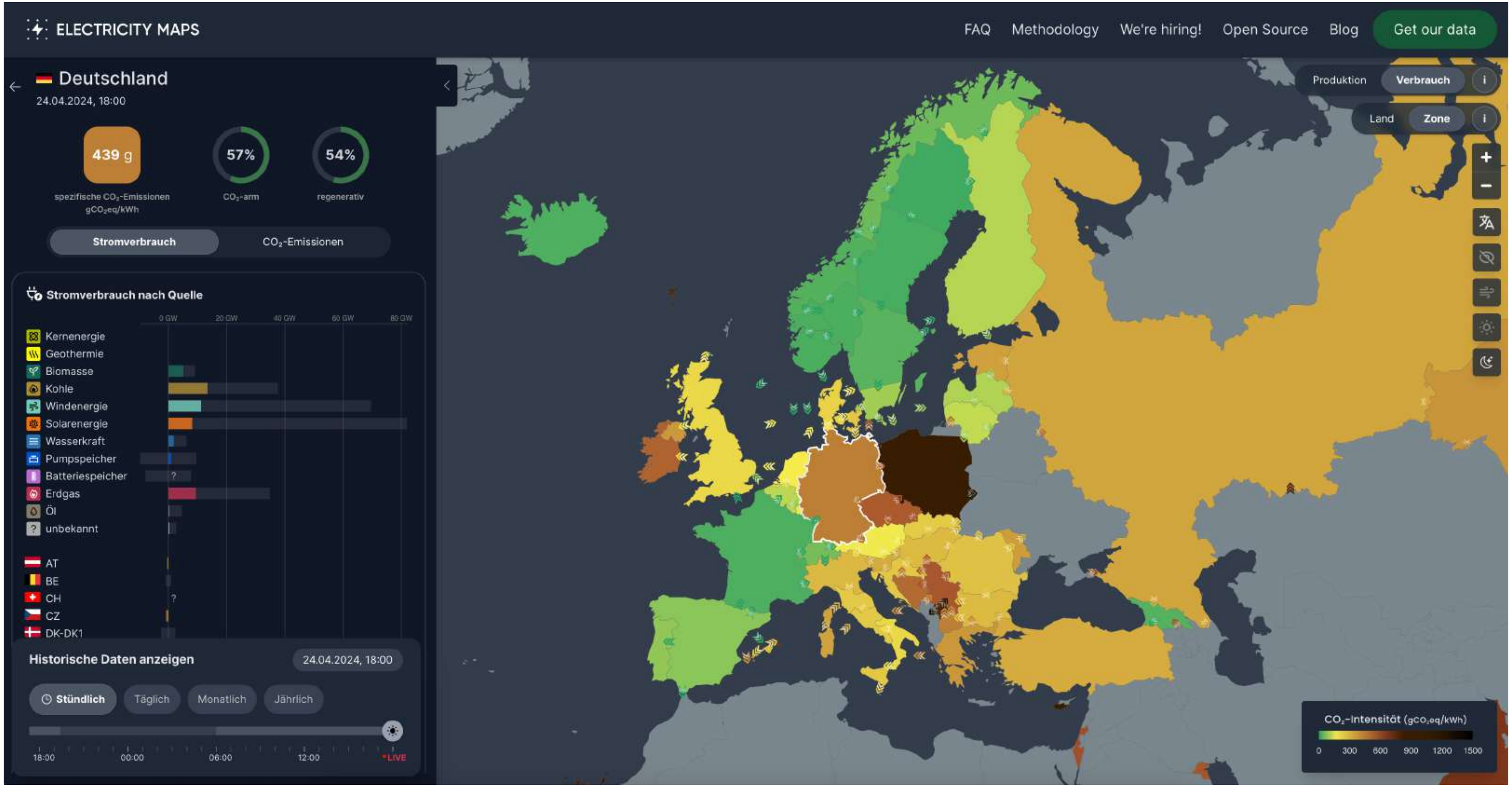
Deploy

best practices

- Only deploy changes worth deploying
- Consider the sustainability of the deploy location
- Deploy during times of curtailment
- Use declarative Infrastructure as Code

Deploy

carbon awareness



[Electricity Maps | CO_2-Emissionen des Stromverbrauchs in Echtzeit]

Operate

Operate

best practices

- Right size your infrastructure
- Pay attention to Scaling
- Consider carbon awareness

Operate - Tool

Keda

Features

Autoscaling Made Simple

Bring rich scaling to every workload in your Kubernetes cluster

Event-driven

Intelligently scale your event-driven application

Built-in Scalers

Catalog of 50+ built-in scalers for various cloud platforms, databases, messaging systems, telemetry systems, CI/CD, and more

Multiple Workload Types

Support for variety of workload types such as deployments, jobs & custom resources with `/scale` sub-resource

Reduce environmental impact

Build sustainable platforms by optimizing workload scheduling and scale-to-zero

Extensible

Bring-your-own or use community-maintained scalers

Vendor-Agnostic

Support for triggers across variety of cloud providers & products

Azure Functions Support

Run and scale your Azure Functions on Kubernetes in production workloads

KEDA | Kubernetes Event-driven Autoscaling

Operate - Example

Branch magazine

[Branch Magazine: A Just and Sustainable Internet for All - Branch (climateaction.tech)]

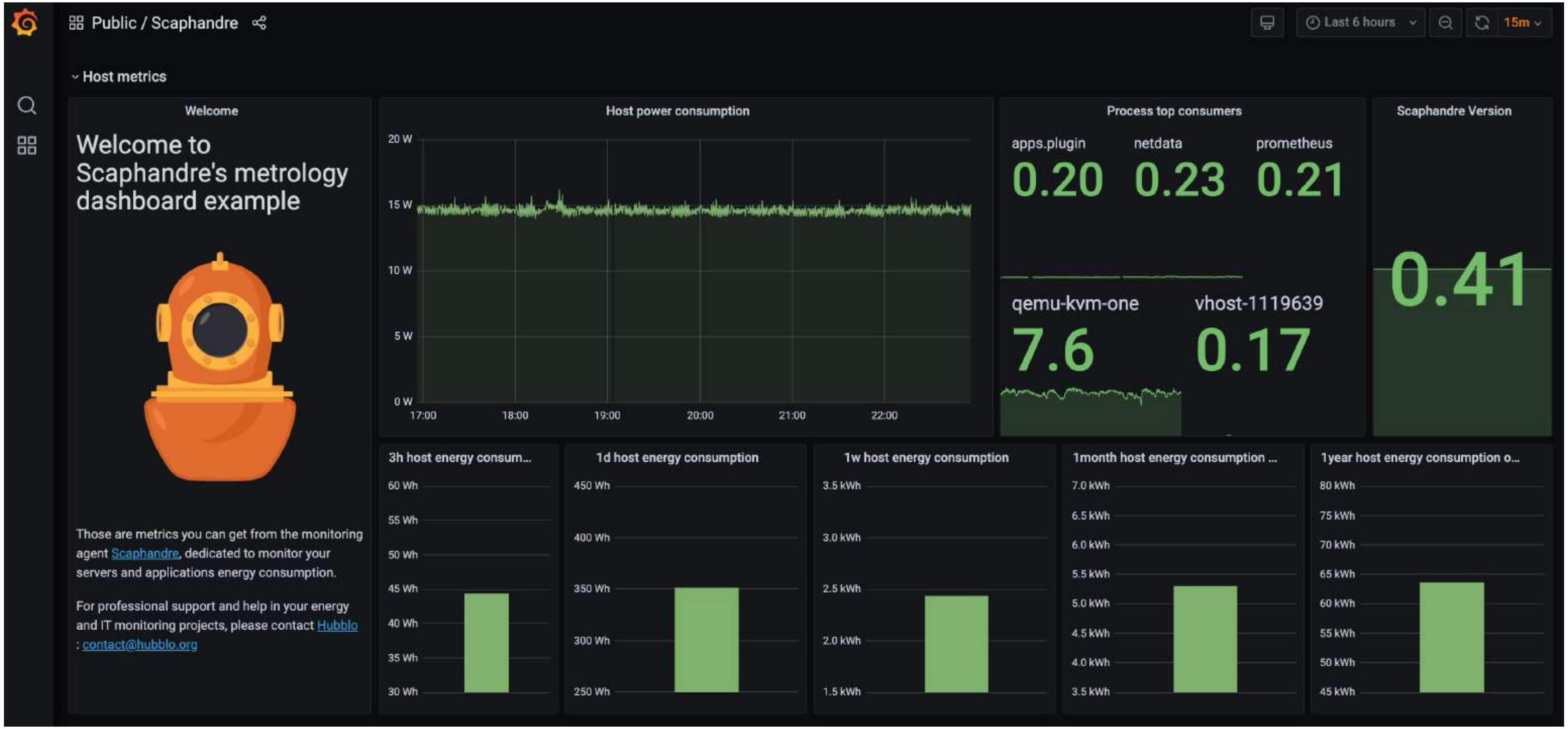
Monitor

Monitor

best practices

- Delete old data
- Only collect relevant data
- monitor environmental factors/impact

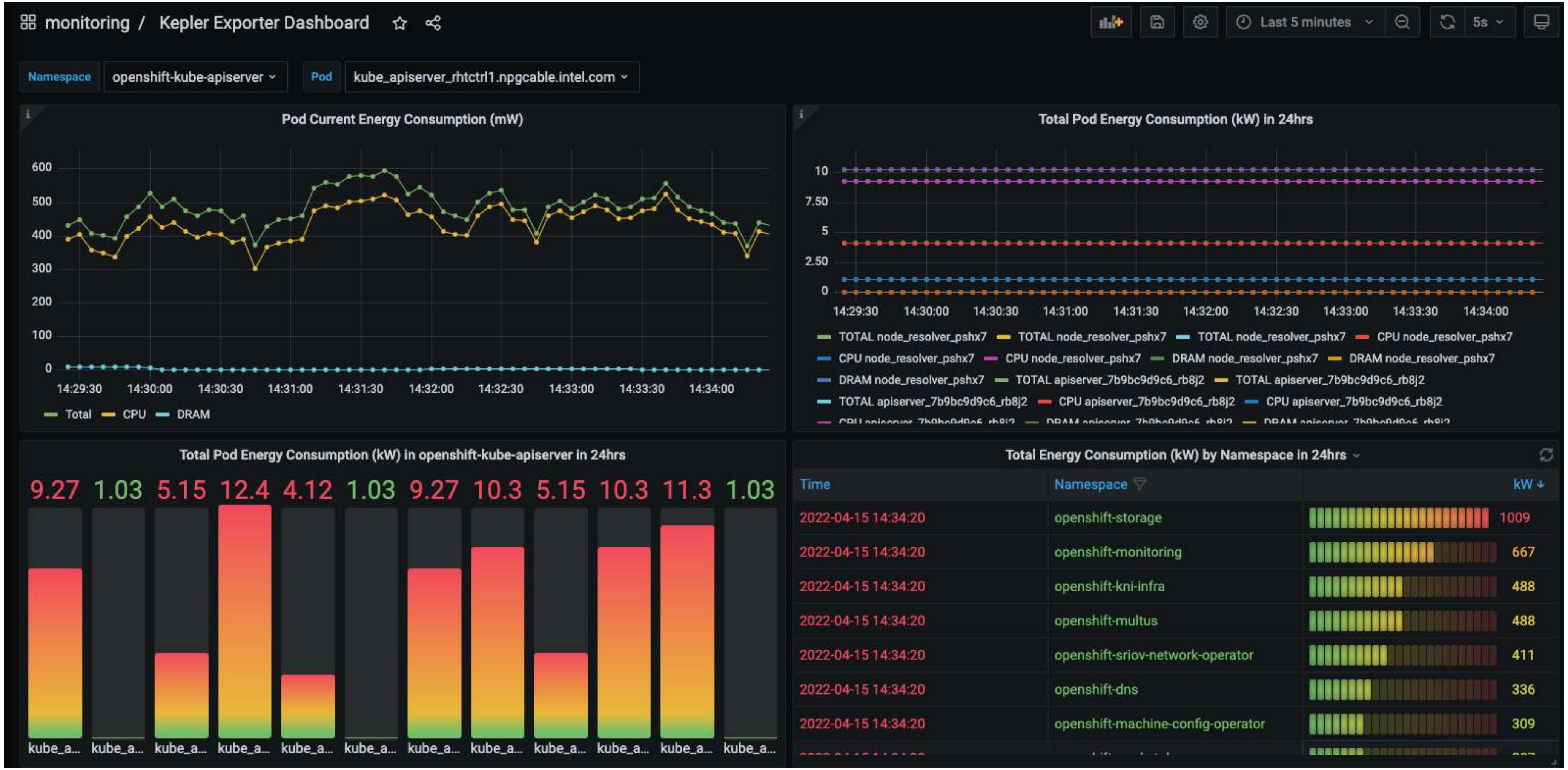
Scaphandre



[GitHub - hubblo-org/scaphandre: ⚡ Energy consumption metrology agent]

Monitor - Tools

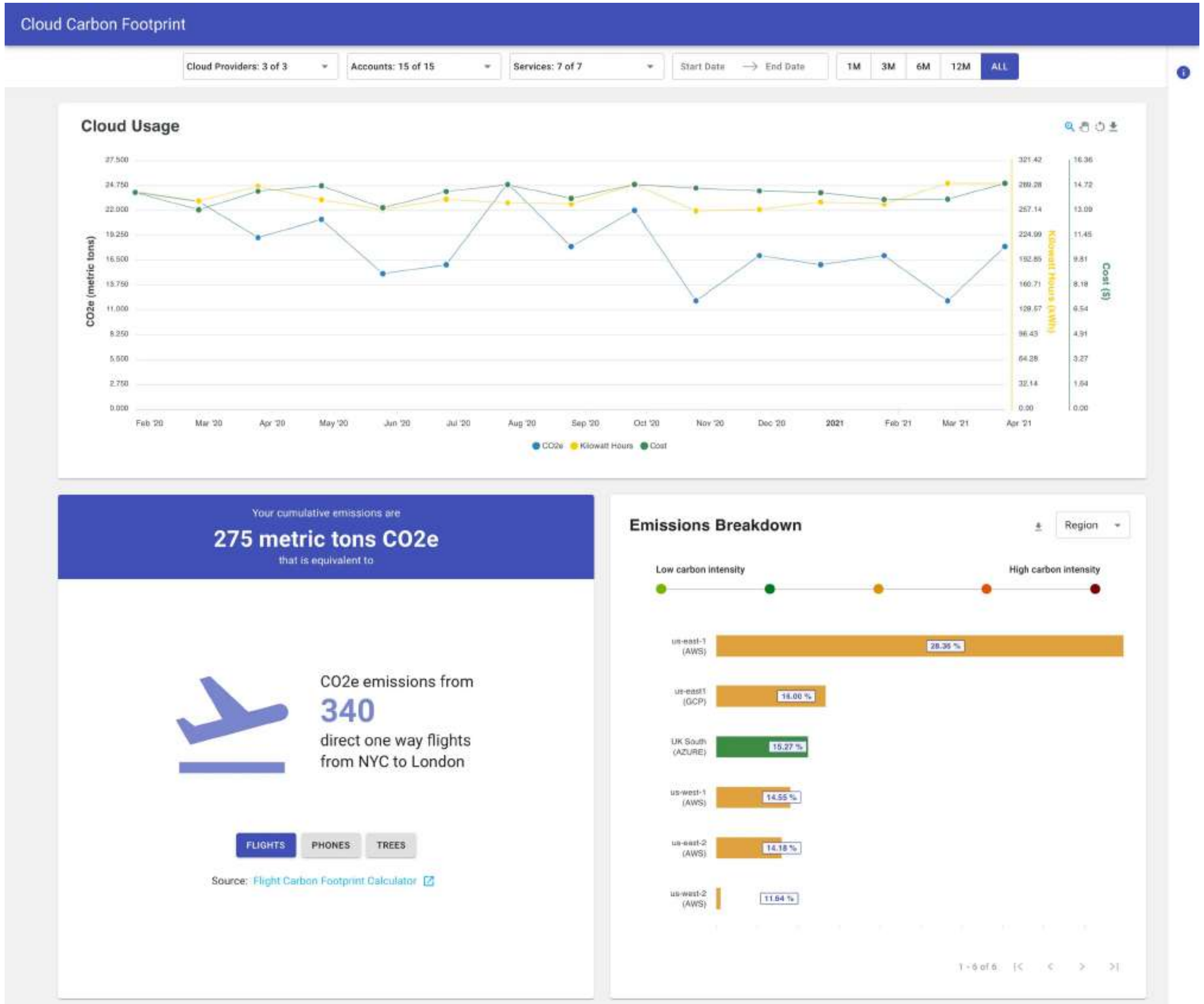
Kepler



[GitHub - sustainable-computing-io/kepler: Kepler (Kubernetes-based Efficient Power Level Exporter)]()

Monitor - Tools

Cloud Carbon Footprint



[Cloud Carbon Footprint - An open source tool to measure and analyze cloud carbon emissions]()

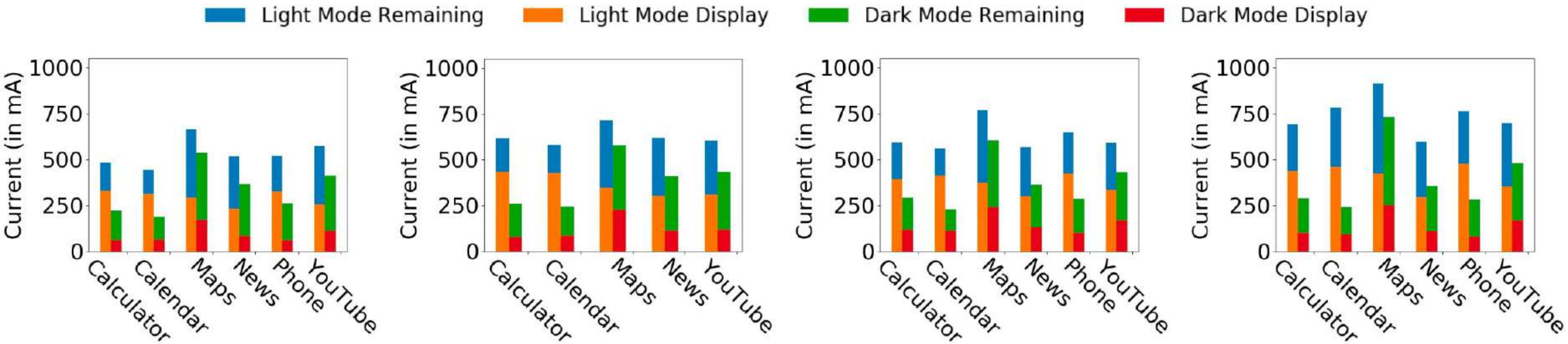
Plan

best practices

- Optimize User Journeys
- Find solutions for the Jevons Paradox
- Design with user and device in mind
- Make the ecosystem a stakeholder
- Use media and formats appropriately

Plan - Example

Dark Mode



(a) Pixel 2 (b) Moto Z3 (c) Pixel 4 (d) Pixel 5

Figure 11: Total phone and OLED power draw under light mode and dark mode on the four phones.

[Dash]

Plan - Example

Favicons

Plan - Example

Favicons

Plan - Example

Favicons

Original

13KB

6.143 gCO2eq

32km

Option A

2KB

946 gCO2eq

4.6km

Option B

1.7KB

804 gCo2eq

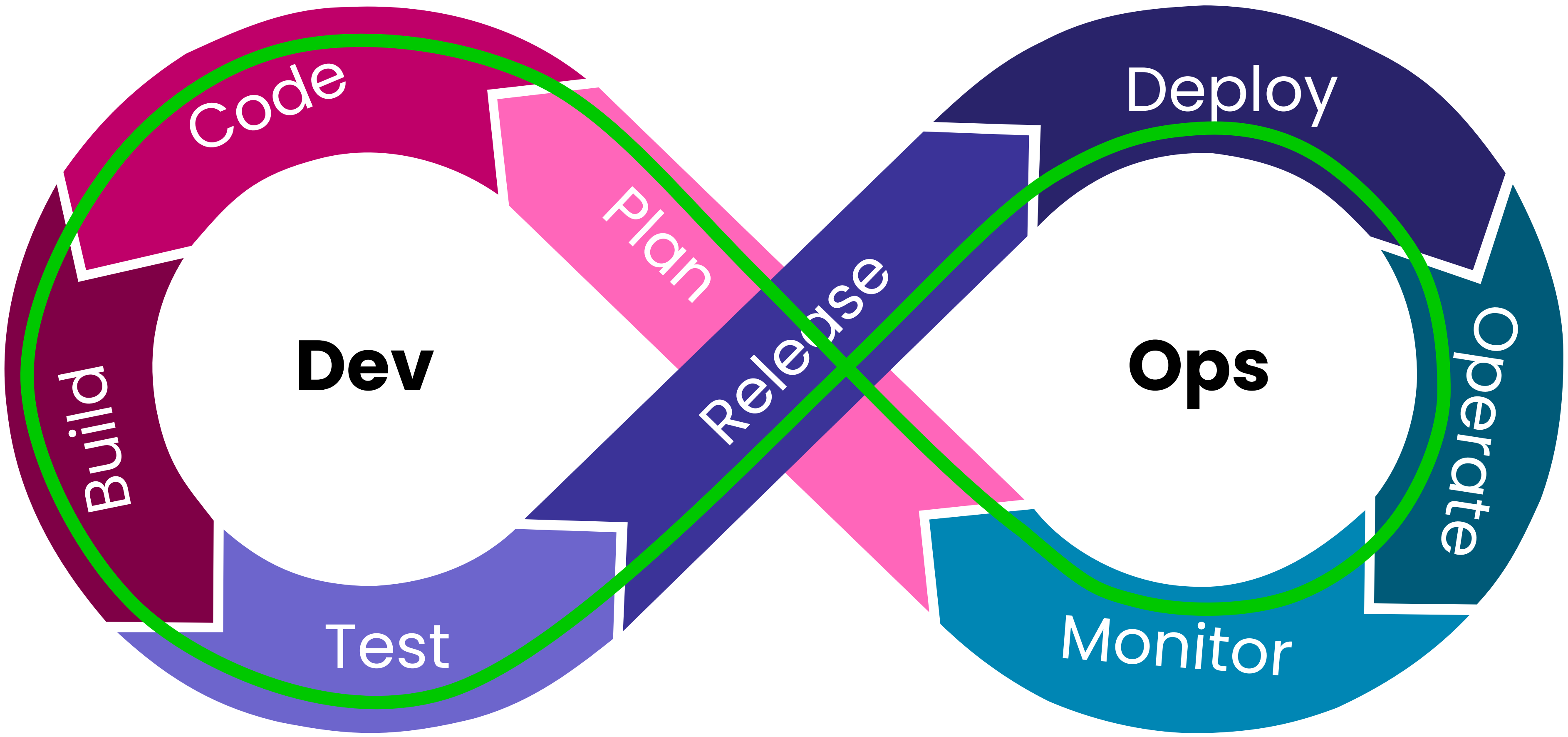
4km

Plan - Example

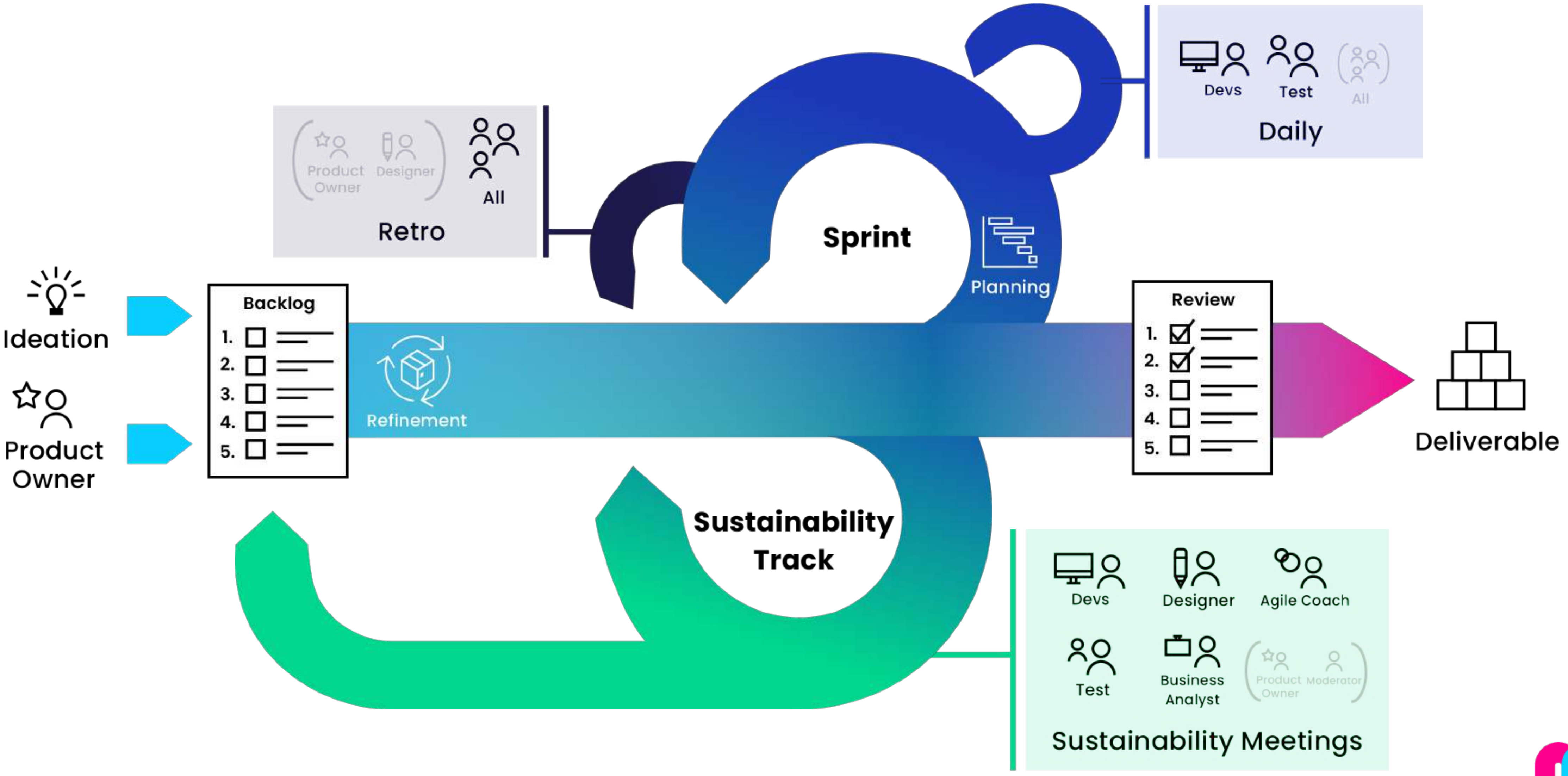
Favicons

Take away

DevSusOps


Code
Plan
Build
Dev
Test
Release
Deploy
Ops
Operate
Monitor

DevSusOps



Use Sustainability Meetings to introduce sustainability efforts into every part of your software and development process

Read more: [Jochen Joswig on LinkedIn: Download: So kommt Nachhaltigkeit ins agile Projekt] (atm only available in German)

Green Software Development Principals

Transparent

Minimal

Efficient

Aware

Green Software Development Principals & Patterns

Transparent

- Accountability
- Monitoring
- Reporting
- Observability
- Carbon-Budget
- Scoreboards

Minimal

- Scale to Zero
- Cloud Resources
- Data collection
- Data retention
- Compression
- Bundle Size
- Feature Set
- Request frequency
- Request body size
- User interaction
- Dependencies

Efficient

- Programming language
- Algorithms
- Implementation
- Dependencies
- Caching
- Hardware Use
- Device Lifespan
- Useful work vs. utilization
- Perfection???

Aware

- Carbon Awareness
  - Demand Shifting
  - Temporal
  - Spatial
- Demand Shaping
- User Behavior
- Configurability
- Jevons Paradox

LinkedIn

Thank you for your attention

Jochen Joswig

MAIBORNWOLFF

References & Sources

References

- Bunse, C., & Stiemer, S. (2013). On the energy consumption of design patterns.
- Dash, P., & Hu, Y. C. (2021, June). How much battery does dark mode save? An accurate OLED display power profiler for modern smartphones. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services* (pp. 323-335).
- Gröger, J., Köhler, A., Naumann, S., Filler, A., Guldner, A., Kern, E., ... & Maksimov, Y. (2018). Entwicklung und Anwendung von Bewertungsgrundlagen für ressourceneffiziente Software unter Berücksichtigung bestehender Methodik-Abschlussbericht. *UBA TEXTE*, *105*.
- Jonuzi, F. (2024). Analyse des Energiebedarfs von Datenbankmanagementsystemen. (pp. 29)
- Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J. P., & Saraiva, J. (2021). Ranking programming languages by energy efficiency. *Science of Computer Programming*, *205*, 102609.
- Vautier, M., & Philippot, O. (2016, September). Is “software eco-design” a solution to reduce the environmental impact of electronic equipments?. In *2016 Electronics Goes Green 2016+(EGG)* (pp. 1-6). IEEE.