
How Kubernetes Optimization can combat climate change
Annie Talvasto

Annie Talvasto

- CMO at VSHN

- CNCF Ambassador
- Azure MVP

- Kubernetes & CNCF meetup co-organizer
- Startup coach
- Co-host of Cloudgossip podcast - cloudgossip.net

Welcome to the session!

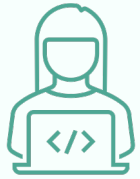
What will you learn?

Agenda

- Introduction
- Principles of Green Software
- Principles in Practice: Microservices
- Open Source Projects: KEDA
- Resources

People care about sustainability – but what about at work?

Sustainability & software development



ICT industry consistently adds 2 to 6% of emissions each year since 2007 – the same as airline industry.

Modern technologies require more and more compute power.

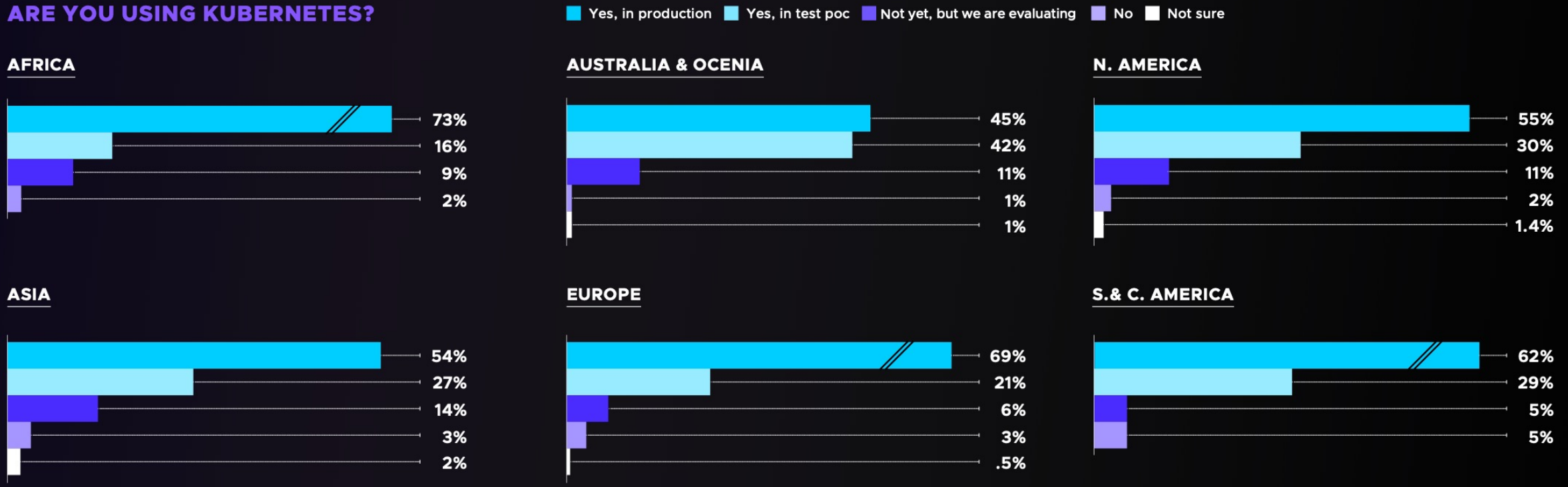
World is going to experience irreversible changes if climate change remains unchecked.

KUBERNETES HAS CROSSED THE ADOPTION CHASM TO BECOME A MAINSTREAM GLOBAL TECHNOLOGY

According to CNCF's respondents, 96% of organizations are either using or evaluating Kubernetes – a record high since our surveys began in 2016. Particularly interesting is the regional adoption of Kubernetes in production, with emerging technology hub Africa (73%) jumping ahead of other more established tech centers including Europe (69%) and North America (55%). Additionally, 93% of respondents are currently using, or planning to use, containers in production, echoing 92% in our **2020 survey**.

96% OF ORGANIZATIONS ARE EITHER USING OR EVALUATING KUBERNETES

ARE YOU USING KUBERNETES?



Search

- Home
- Getting started
- Concepts
  - Overview
  - Cluster Architecture
  - Containers
  - Workloads
  - Services, Load Balancing, and Networking
  - Storage
  - Configuration
    - **Configuration Best Practices**
    - ConfigMaps
    - Secrets
    - Resource Management for Pods and Containers
    - Organizing Cluster Access Using kubeconfig Files
    - Resource Management for Windows nodes

Configuration Best Practices

This document highlights and consolidates configuration best practices that are introduced throughout the user guide, Getting Started documentation, and examples.

This is a living document. If you think of something that is not on this list but might be useful to others, please don't hesitate to file an issue or submit a PR.

General Configuration Tips

- When defining configurations, specify the latest stable API version.
- Configuration files should be stored in version control before being pushed to the cluster. This allows you to quickly roll back a configuration change if necessary. It also aids cluster re-creation and restoration.
- Write your configuration files using YAML rather than JSON. Though these formats can be used interchangeably in almost all scenarios, YAML tends to be more user-friendly.
- Group related objects into a single file whenever it makes sense. One file is often easier to manage than several. See the guestbook-all-in-one.yaml file as an example of this syntax.
- Note also that many `kubectl` commands can be called on a directory. For example, you can call `kubectl apply` on a directory of config files.
- Don't specify default values unnecessarily: simple, minimal configuration will make errors less likely.
- Put object descriptions in annotations, to allow better introspection.

"Naked" Pods versus ReplicaSets, Deployments, and

Edit this page
Create child page
Create an issue
Print entire section

General Configuration Tips
"Naked" Pods versus ReplicaSets, Deployments, and Jobs
Services
Using Labels
Using kubectl

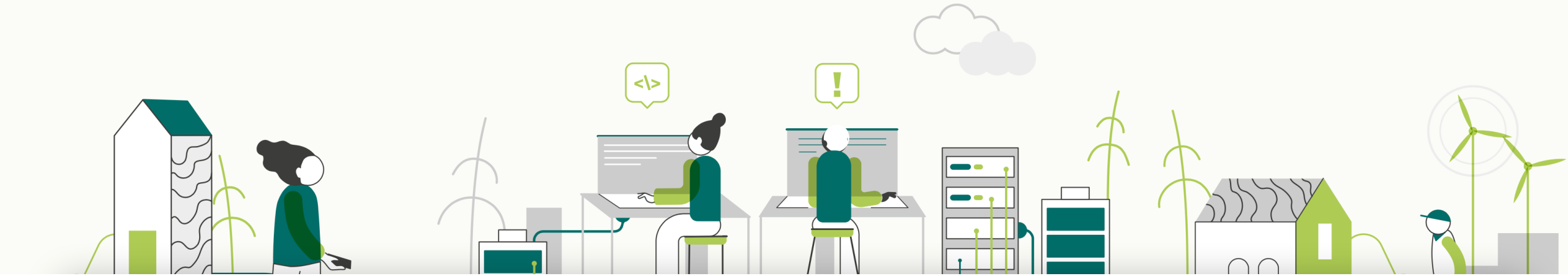
Principles of Green Software

We are building a trusted ecosystem of people, standards, tooling and best practices for

GREEN SOFTWARE

Sign up to our newsletter...

Sign up



Principles of Green software

- Carbon: Build applications that are carbon efficient.

Principles of Green software

- Carbon: Build applications that are carbon efficient.
- Electricity: Build applications that are energy efficient.

Principles of Green software

- Carbon: Build applications that are carbon efficient.
- Electricity: Build applications that are energy efficient.
  - Carbon Intensity: Consume electricity with the lowest carbon intensity.

Principles of Green software

- Carbon: Build applications that are carbon efficient.
- Electricity: Build applications that are energy efficient.
- Carbon Intensity: Consume electricity with the lowest carbon intensity.
- Embodied Carbon: Build applications that are hardware efficient.

Principles of Green software

- Carbon: Build applications that are carbon efficient.
- Electricity: Build applications that are energy efficient.
- Carbon Intensity: Consume electricity with the lowest carbon intensity.
- Embodied Carbon: Build applications that are hardware efficient.
- Energy Proportionality: Maximize the energy efficiency of hardware.

Principles of Green software

- Carbon: Build applications that are carbon efficient.
- Electricity: Build applications that are energy efficient.
- Carbon Intensity: Consume electricity with the lowest carbon intensity.
- Embodied Carbon: Build applications that are hardware efficient.
- Energy Proportionality: Maximize the energy efficiency of hardware.
- Networking: Reduce the amount of data and distance it must travel across the network.

Principles of Green software

- Carbon: Build applications that are carbon efficient.
- Electricity: Build applications that are energy efficient.
- Carbon Intensity: Consume electricity with the lowest carbon intensity.
- Embodied Carbon: Build applications that are hardware efficient.
- Energy Proportionality: Maximize the energy efficiency of hardware.
- Networking: Reduce the amount of data and distance it must travel across the network.
- Demand Shaping: Build carbon-aware applications.

Principles of Green software

- Carbon: Build applications that are carbon efficient.
- Electricity: Build applications that are energy efficient.
- Carbon Intensity: Consume electricity with the lowest carbon intensity.
- Embodied Carbon: Build applications that are hardware efficient.
- Energy Proportionality: Maximize the energy efficiency of hardware.
- Networking: Reduce the amount of data and distance it must travel across the network.
- Demand Shaping: Build carbon-aware applications.
- Measurement & Optimization: Focus on step-by-step optimizations that increase the overall carbon efficiency.

Principles of Green software

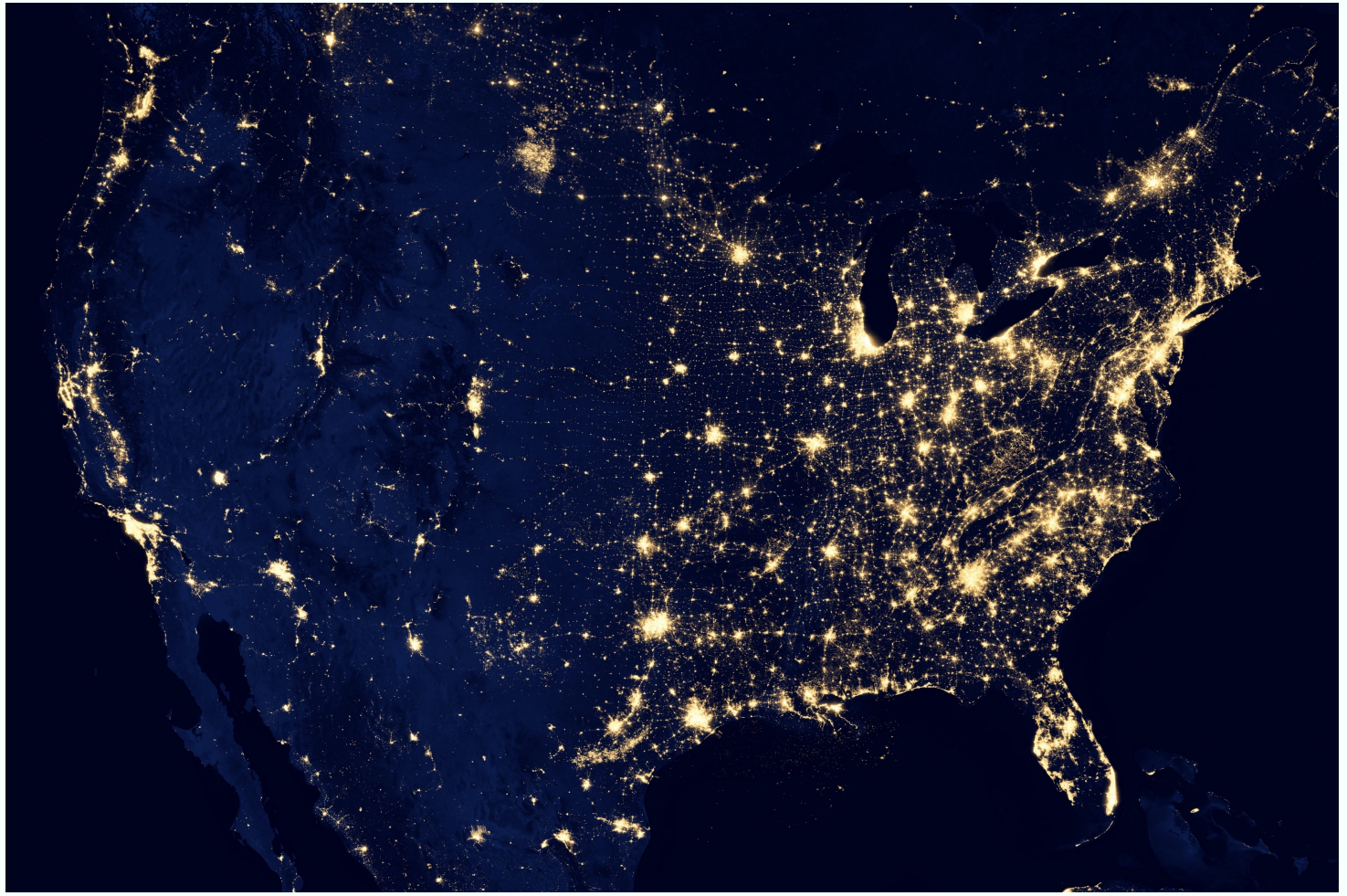
- Carbon: Build applications that are carbon efficient.
- Electricity: Build applications that are energy efficient.
- Carbon Intensity: Consume electricity with the lowest carbon intensity.
- Embodied Carbon: Build applications that are hardware efficient.
- Energy Proportionality: Maximize the energy efficiency of hardware.
- Networking: Reduce the amount of data and distance it must travel across the network.
- Demand Shaping: Build carbon-aware applications.
- Measurement & Optimization: Focus on step-by-step optimizations that increase the overall carbon efficiency.

Too many things? Focus on these:

- Carbon efficiency
  - Energy efficiency
  - Hardware efficiency
  - Carbon awareness

Energy

Location sifting

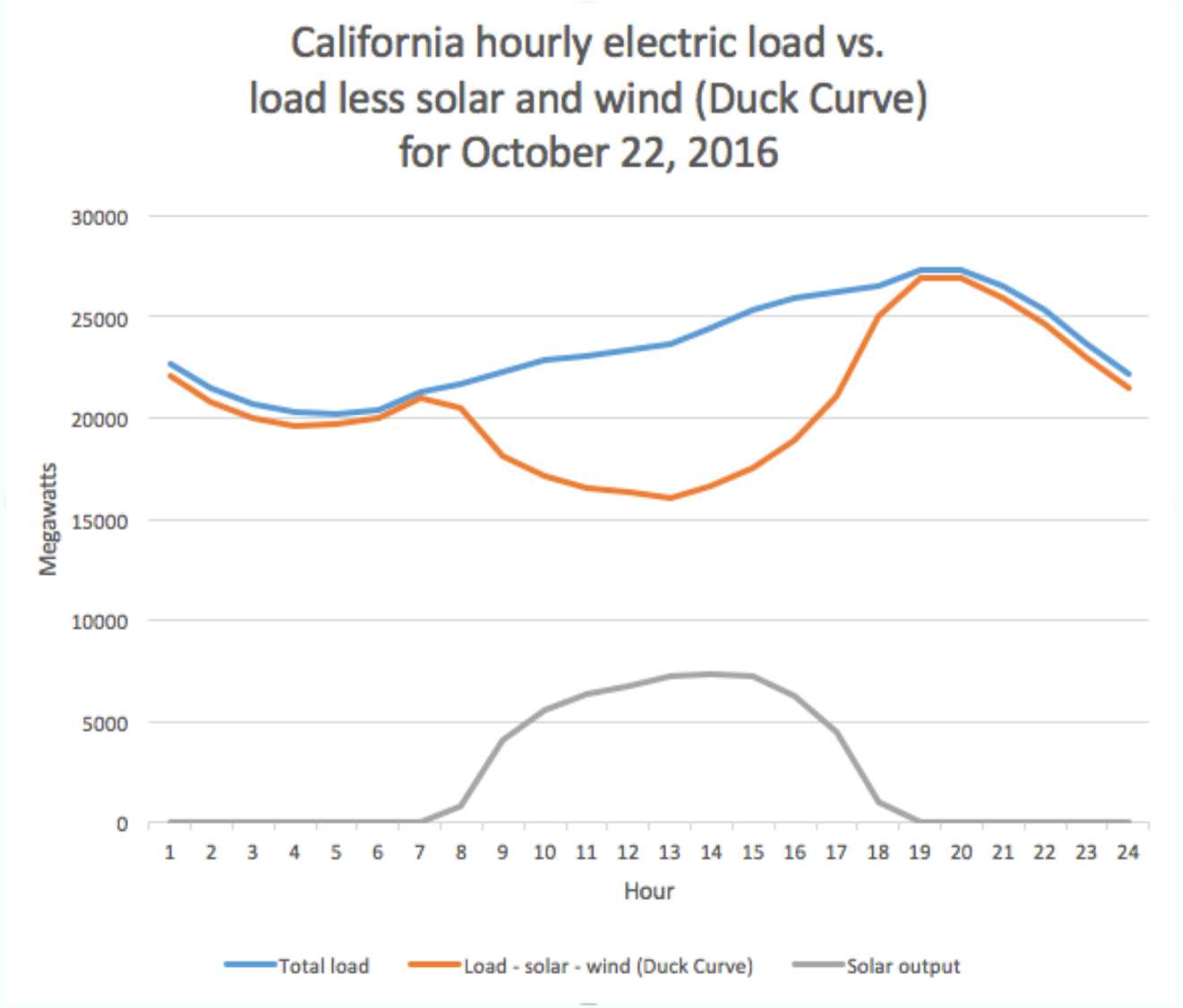


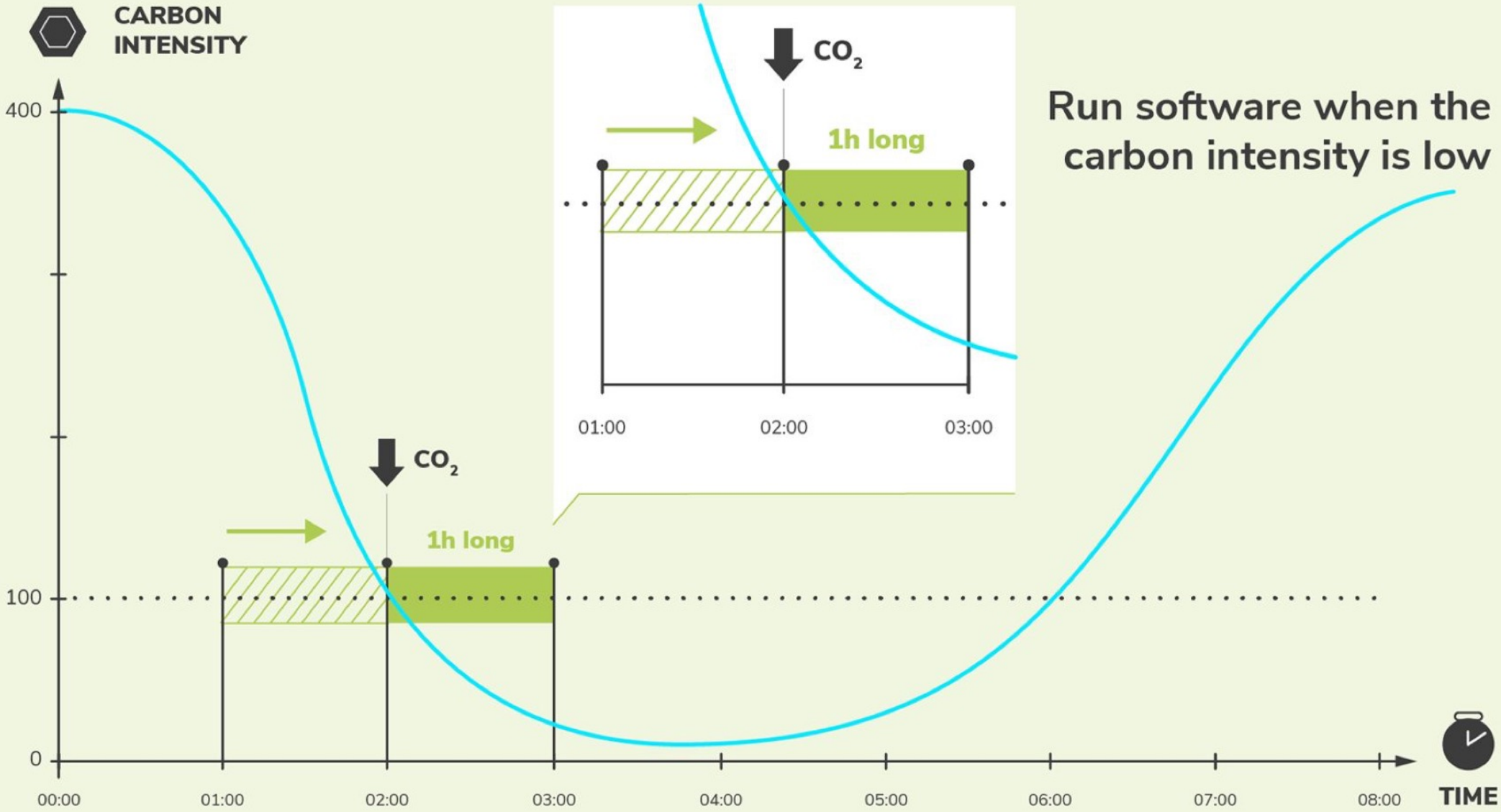
Varies by location

Green Software Foundation

greensoftware.org

Time sifting





Run software when the carbon intensity is low

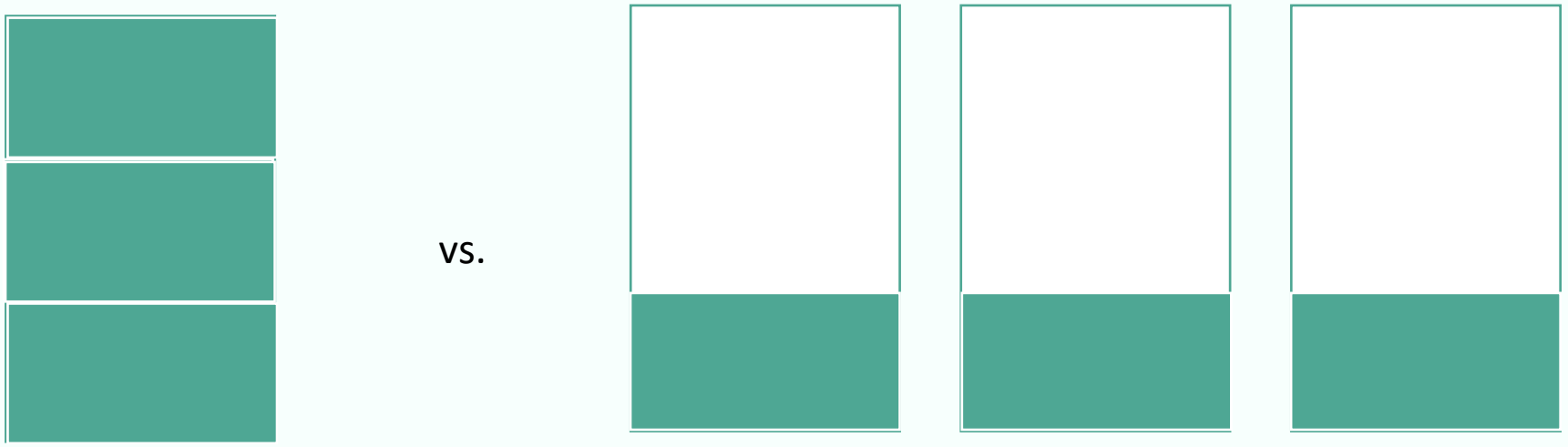
Green Software Foundation

greensoftware.org

Demand shaping

@AnnieTalvasto

Increasing utilization



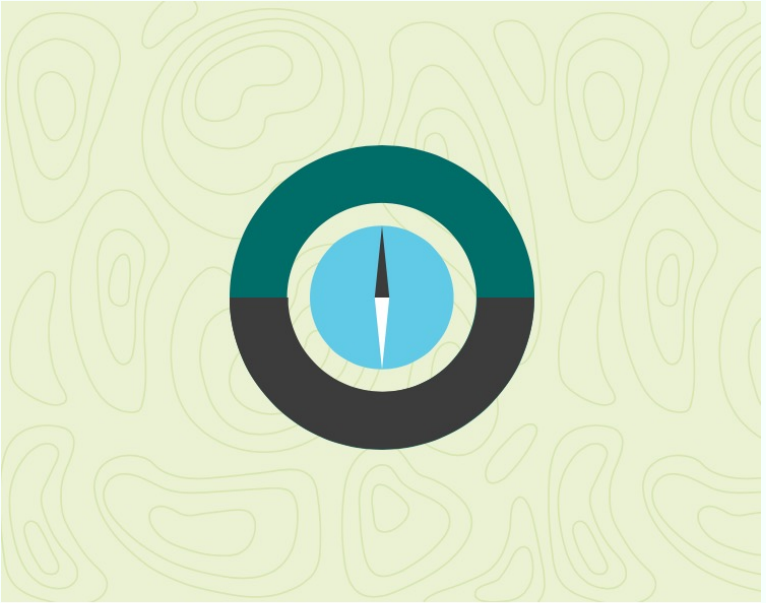
100%.

30%

30%

30%

Software Carbon Intensity – SCI

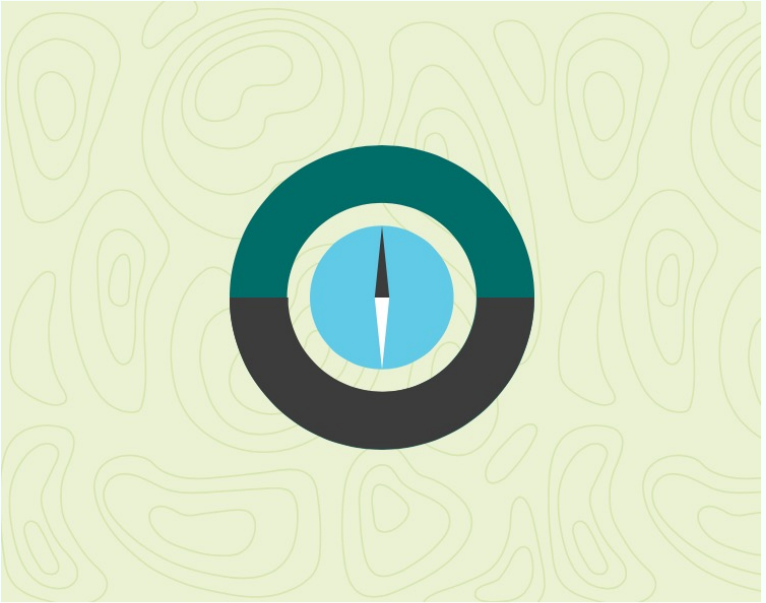


Software Carbon Intensity (SCI) Specification

The Software Carbon Intensity (SCI) Specification defines a methodology for calculating the rate of carbon emissions for a software system. The purpose is to help users and developers make informed choices about which tools, approaches, architectures, and services they use in the future. It is a score rather than a total; lower numbers are better than higher numbers, and reaching 0 is impossible.

Software Carbon Intensity – SCI

Deep dive in GSF GH:
https://github.com/Green-Software-Foundation/software_carbon_intensity/blob/main/Software_Carbon_Intensity/Software_Carbon_Intensity_Specification.md



Software Carbon Intensity (SCI) Specification

The Software Carbon Intensity (SCI) Specification defines a methodology for calculating the rate of carbon emissions for a software system. The purpose is to help users and developers make informed choices about which tools, approaches, architectures, and services they use in the future. It is a score rather than a total; lower numbers are better than higher numbers, and reaching 0 is impossible.

Principles of Green Software in Practice: Microservices

@AnnieTalvasto

Microservices example (or Web Queue Worker)

- Optimize network traffic
- Increase your compute utilization
- Reduce your number of Microservices
- Optimize your database
- Understand your latency limitations

Optimize Your Network Traffic

- Traffic & architecture
- Caching headers
- CDN (Content delivery network)
- Reduce the size and optimize your bundles and static assets.
- Compression and decompression for data

Increase Your Compute Utilization

- Update workload distribution and compute resources - use less resources at a higher utilization.
- Smaller virtual machines
- PaaS
- Auto-scaling or burst capabilities
- Physical tiers & logical layers

Reduce the Number of Microservices

- Microservices architecture
- Combining services
- If two or more microservices are highly coupled, consider co-locating to reduce network congestion and latency
- Use Regions with a lower carbon intensity
- Use languages and technology stacks that optimize the efficiency of a specific microservices function.

Energy Efficiency across Programming Languages

Energy Efficiency across Programming Languages

How Do Energy, Time, and Memory Relate?

Rui Pereira
HASLab/INESC TEC
Universidade do Minho, Portugal
ruipereira@di.uminho.pt

Marco Couto
HASLab/INESC TEC
Universidade do Minho, Portugal
marco.l.couto@inesctec.pt

Francisco Ribeiro, Rui Rua
HASLab/INESC TEC
Universidade do Minho, Portugal
fribeiro@di.uminho.pt
rrua@di.uminho.pt

Jácome Cunha
NOVA LINCS, DI, FCT
Univ. Nova de Lisboa, Portugal
jacome@fct.unl.pt

João Paulo Fernandes
Release/LISP, CISUC
Universidade de Coimbra, Portugal
jpf@dei.uc.pt

João Saraiva
HASLab/INESC TEC
Universidade do Minho, Portugal
saraiva@di.uminho.pt

	Energy
(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

	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

Energy Efficiency across Programming Languages

Energy Efficiency across Programming Languages

How Do Energy, Time, and Memory Relate?

Rui Pereira
HASLab/INESC TEC
Universidade do Minho, Portugal
ruipereira@di.uminho.pt

Marco Couto
HASLab/INESC TEC
Universidade do Minho, Portugal
marco.l.couto@inesctec.pt

Francisco Ribeiro, Rui Rua
HASLab/INESC TEC
Universidade do Minho, Portugal
fribeiro@di.uminho.pt
rrua@di.uminho.pt

Jácome Cunha
NOVA LINCS, DI, FCT
Univ. Nova de Lisboa, Portugal
jacome@fct.unl.pt

João Paulo Fernandes
Release/LISP, CISUC
Universidade de Coimbra, Portugal
jpf@dei.uc.pt

João Saraiva
HASLab/INESC TEC
Universidade do Minho, Portugal
saraiva@di.uminho.pt

	Energy
(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

	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

Optimize your Database

- Choose your database well
- Ensure you are using the best database for interacting with your data set
  - If no easy solution - reduntant copies
- Index
- Evaluate and optimize your queries
- Database cache

Understand your latency limits

- Think about what you actually need for your application
- Request/response cycle
- Worker processes vs web processes
- Worker processes in a region with lower carbon intensity.
- Delay worker process to run when the carbon intensity is the lowest

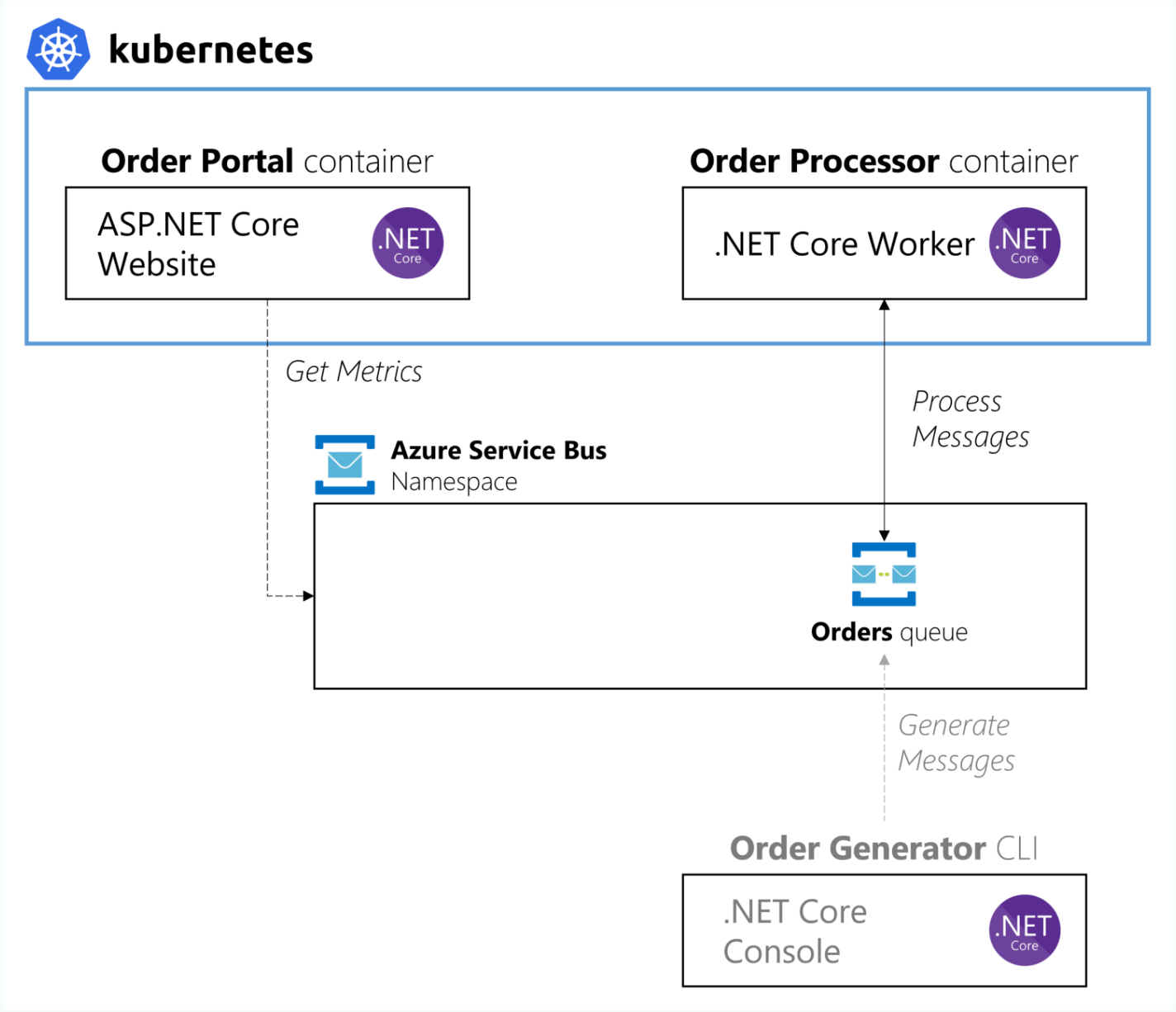
Autoscaling

What is Keda?

- Default Kubernetes Scaling is not well suited for event driven applications.
  - Keda: Event driven scale controlling that can run inside any kubernetes cluster -> Can monitor rate of the events to pre-emptively act before CPU is affected.
- You can install it into new or existing clusters.
- Provides 30+ built-in scalers, but you can build your own
  - Support for external scaler, external push or Metrics API

Demo

KEDA:
Scaling .NET Core worker with Azure Service Bus


kubernetes
Order Portal container
ASP.NET Core Website
.NET Core
Order Processor container
.NET Core Worker
.NET Core
Get Metrics
Process Messages
Azure Service Bus
Namespace
Orders queue
Generate Messages
Order Generator CLI
.NET Core Console
.NET Core

Sponsored Keynote: Building a Sustainable, Carbon-Aware Cloud: Scale Workloads and Re... Jorge Palma

CNCF [Cloud Native Computing Foundation]
100K subscribers

Subscribe

6 Share Clip Save

244 views 7 days ago

Sponsored Keynote: Building a Sustainable, Carbon-Aware Cloud: Scale Workloads and Reduce Emissions - Jorge Palma, Principal PM Lead, Microsoft Azure

When we think about sustainability in the technology space, we know that reducing emissions is essential, even as we face greater demand to build scalable appli Show more

BLOG / COMMUNITY POST

CNCF WG Environmental Sustainability

By Max Körbächer + Leonard Pahlke

May 31, 2022

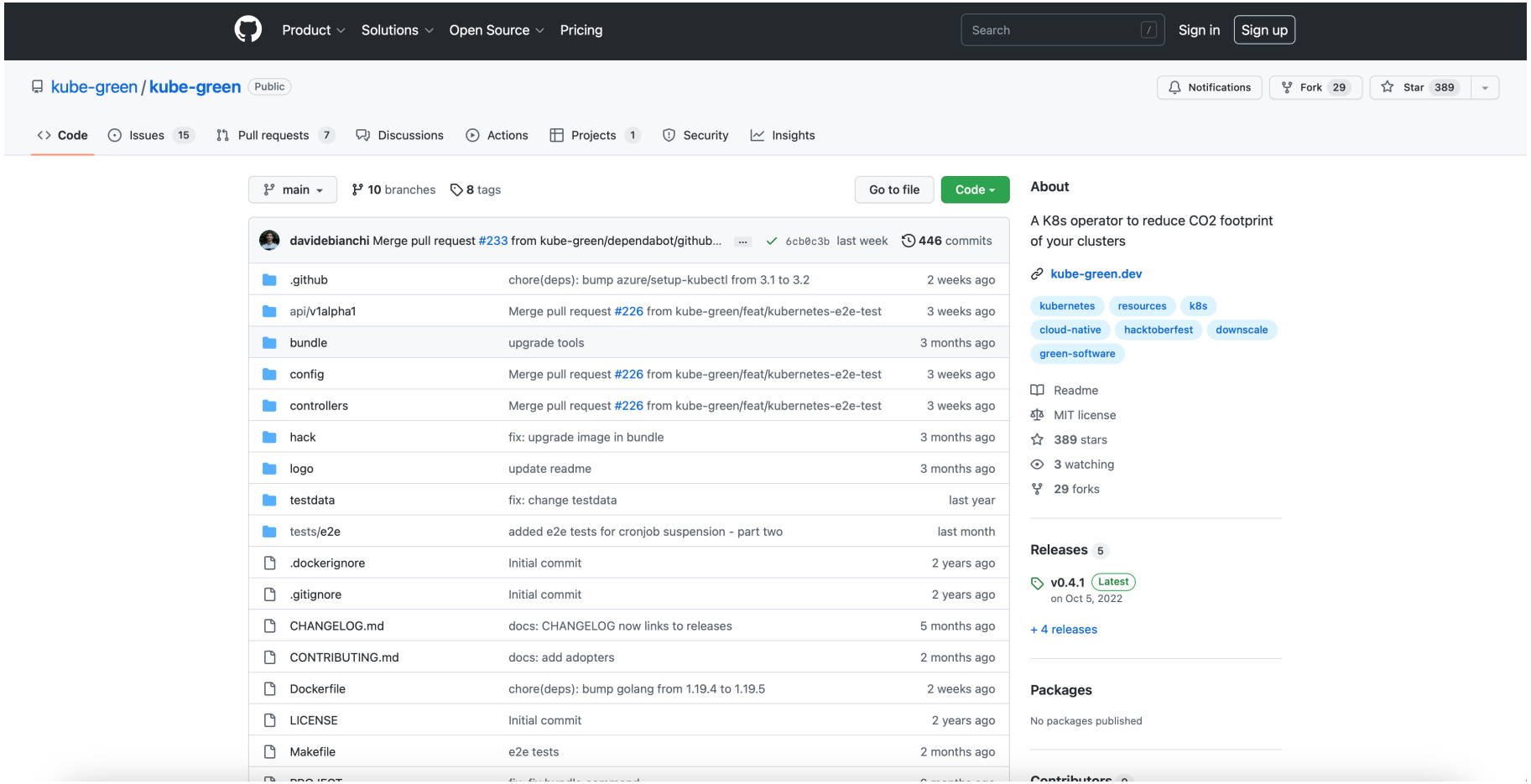
*Community post by **Max Körbächer**, Co-Founder of Liquid Reply, and **Leonard Pahlke**, Consultant at Liquid Reply*

We are pleased to announce that we have established a new working group for environmental sustainability. Our mission is to promote sustainability awareness and develop a culture within the CNCF landscape to establish sustainability best practices and standards.

Environmental sustainability is a pressing issue for humanity, and with simultaneous digitization, software is playing an increasingly important role. Making software ecologically sustainable is becoming more and more vital.

Data Centers, and therefore every compute resource accessible, are currently using 2% of the world's energy. This is conservatively expected to grow to hold 1-2% in the next couple of years, up to 12% by 2040. The growth in energy for computing is outpacing the global growth in energy production. The contributing factors include an explosion in data, the emergence of energy-intensive workloads such as AI and, the flattening of Moore's law. The energy consumption

Greenkube



@AnnieTalvasto

How many CO2 is produced by pod?

This calculations are based on the following assumptions:

- Emissions for cloud server using 100% green electricity: **160 Kg CO2eq / year and server** (from goclimate.com)
- Cluster node of 2 cpu. We approximate **1 node is 1 server**
- **15 pods per node**

With this assumption, the mean consumption of CO2 per pod in a year is 160 / 15 = **11 Kg CO2eq / year per pod**.

CO2 Calculator

CO2 per pods per year (kg CO2eq)	11
Total number of pods	100
Total pods when kube-green active	100
Hour of sleep per week	128

Results

Total (Kg CO2eq/week)

without kube-green: 21

with kube-green: 5

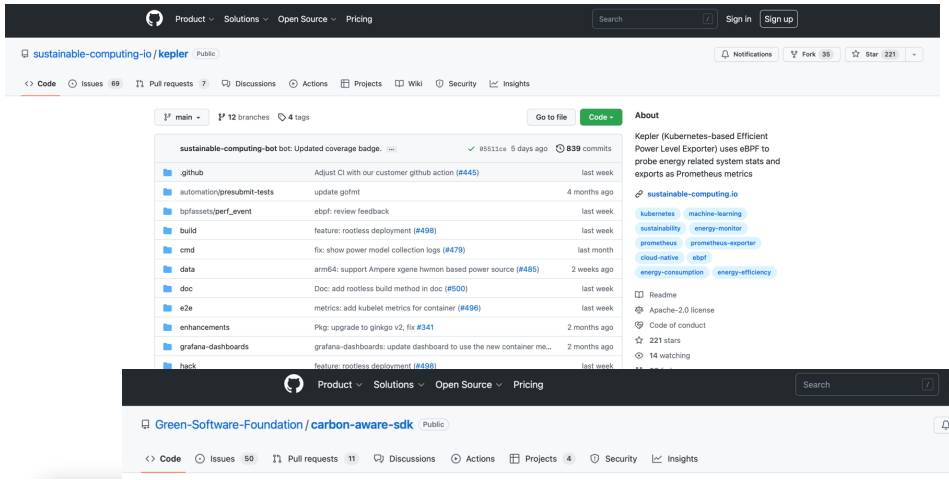
Difference: -16

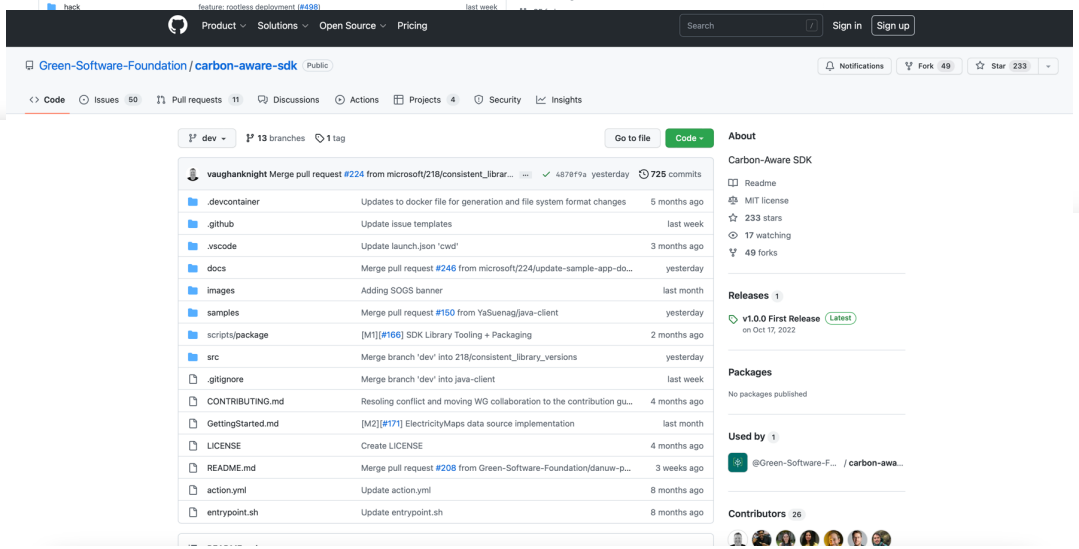
Edit this page

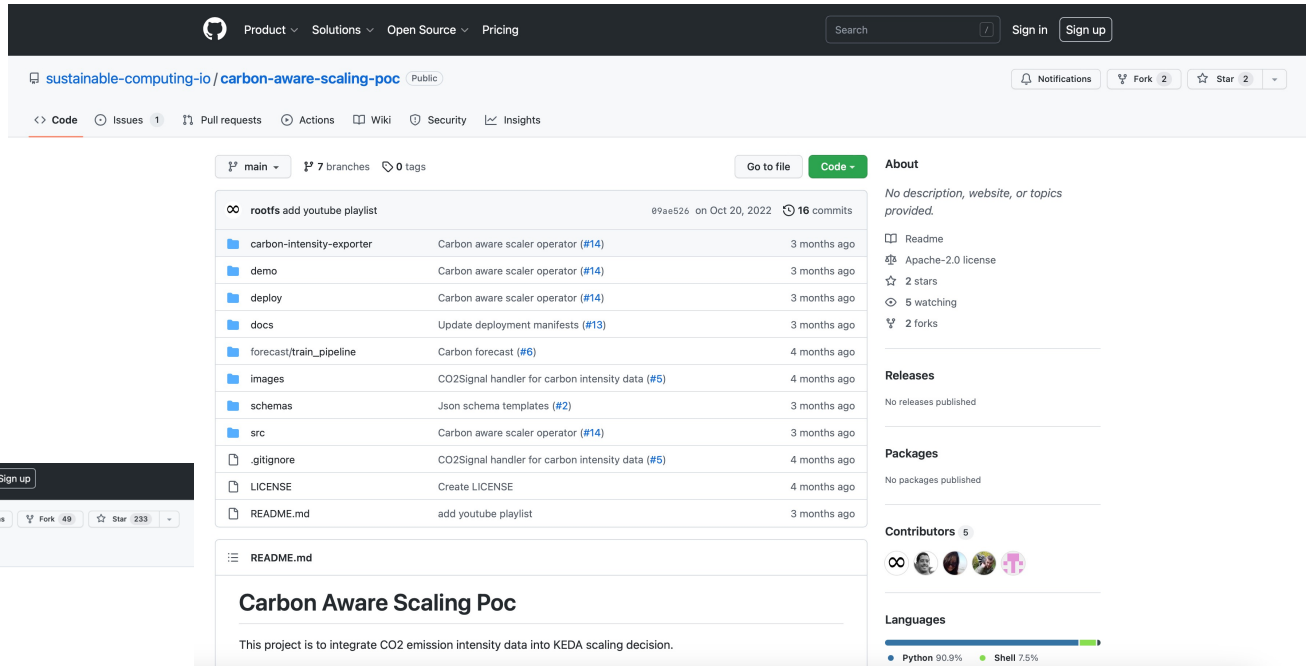
Previous
« API Reference

Next
Adopters »

More projects

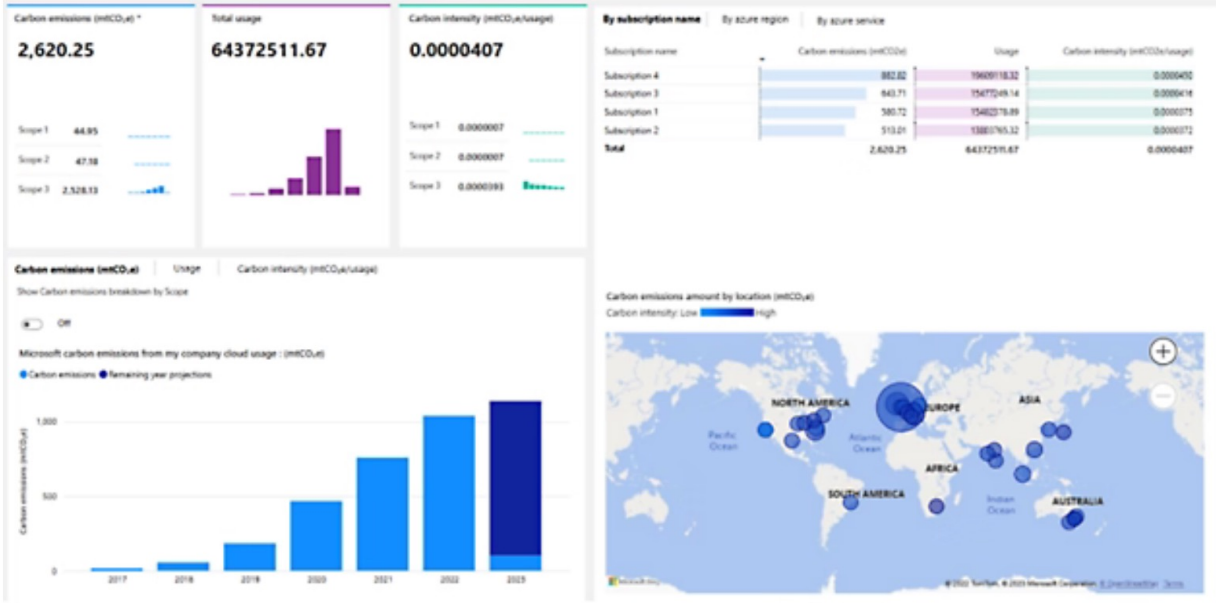


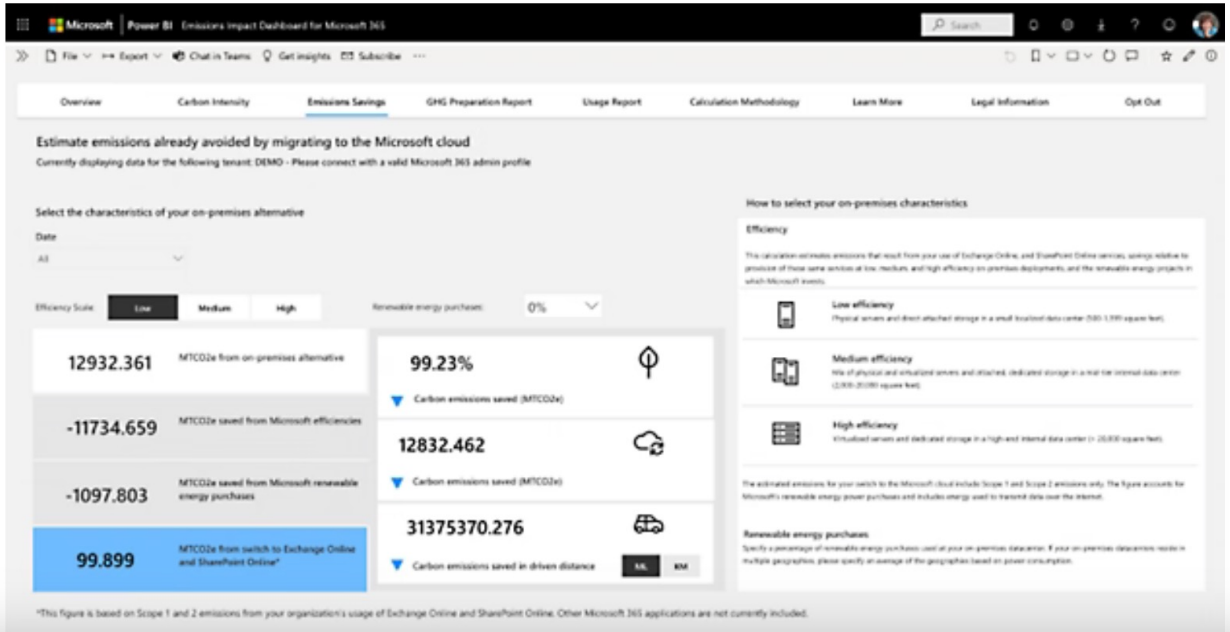




Gain transparency into the carbon impact of your cloud usage

- Track direct and indirect greenhouse gas emissions related to your cloud usage.
- Understand your emissions avoided over time through our datacenter efficiency.
- Easily share your findings, shown in CO2-equivalent metric tons, through cloud data export.
- **New!** Access your Azure emissions data with the Microsoft Cloud for Sustainability API (Preview).





Estimate emissions that you've avoided

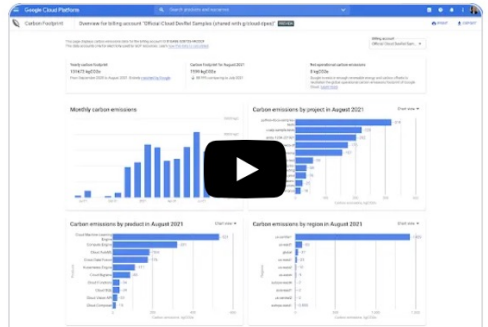
Calculate the emissions your organization has already avoided by moving on-premises workloads to the Microsoft Cloud.

Carbon Footprint

Measure, report, and reduce your cloud carbon emissions.

Try it for free

- Include gross carbon emissions data in reports and disclosures
- Visualize carbon insights via dashboards and charts
- Reduce the gross emissions of cloud applications and infrastructure



VIDEO

Watch this session and learn how to measure carbon emissions on Google Cloud

8.02

BENEFITS

Accurately measure your gross carbon footprint

View the gross, location-based emissions that derive from your Google Cloud usage, providing transparency into emissions associated with your cloud applications. Your net operational emissions are zero.

Track the emissions profile of cloud projects

Monitor your gross cloud emissions over time by project, product, and region—giving IT teams and developers metrics that can help them improve their carbon footprint.

Share detailed methodology with reviewers

Our detailed calculation methodology is published so that reviewers and reporting teams can verify that their emissions data meets GHG Protocol.

Reminder:

Use best practices from other fields creatively and as fits!

Benefits

- Organizational improvements
- Improved performance
- Cost savings
- Sustainability goals

Saving the world !

Wrap up

- Introduction
- Principles of Green Software
- Principles in Practice: Microservices
- CNCF Open Source Projects: KEDA
- Resources

Resources

- Slides & materials:
  - Github.com/annietalvasto

Resources

- **Green Software Foundation**
  - https://greensoftware.foundation
  - Youtube, Podcast, blog etc
  - A lot of upcoming pro

- **CNCF Environmental Sustainability TAG**
  - A lot of projects upcoming: How to Get Involved in CNCF Environmental Sustainability TAG - Marlow Weston & Huamin Chen - https://www.youtube.com/watch?v=XFZZ9YfOyI8
  - Slack channel (CNCF Slack): https://cloud-native.slack.com/archives/C03F270PDU6
  - High level view: https://github.com/orgs/cncf/projects/10/views/1
  - Best practices: https://github.com/cncf/tag-env-sustainability/issues/54

Resources

- Lightning Talk: Green(Ing) CI/CD: A Sustainability Journey with GitOps - Niki Manoledaki, Weaveworks (Kepler, Flux)
  - https://www.youtube.com/watch?v=rgvHbG3U_c4&t
- Using kube-green with Argo CD
  - https://akuity.io/blog/argo-cd-kube-green/
- Carbon Aware Kubernetes article
  - https://greensoftware.foundation/articles/carbon-aware-kubernetes
- Green software patterns https://patterns.greensoftware.foundation/catalog/cloud/

Conclusion

Thank you!


when you remember to bring your reusable bag to the grocery store
STARECAT.COM