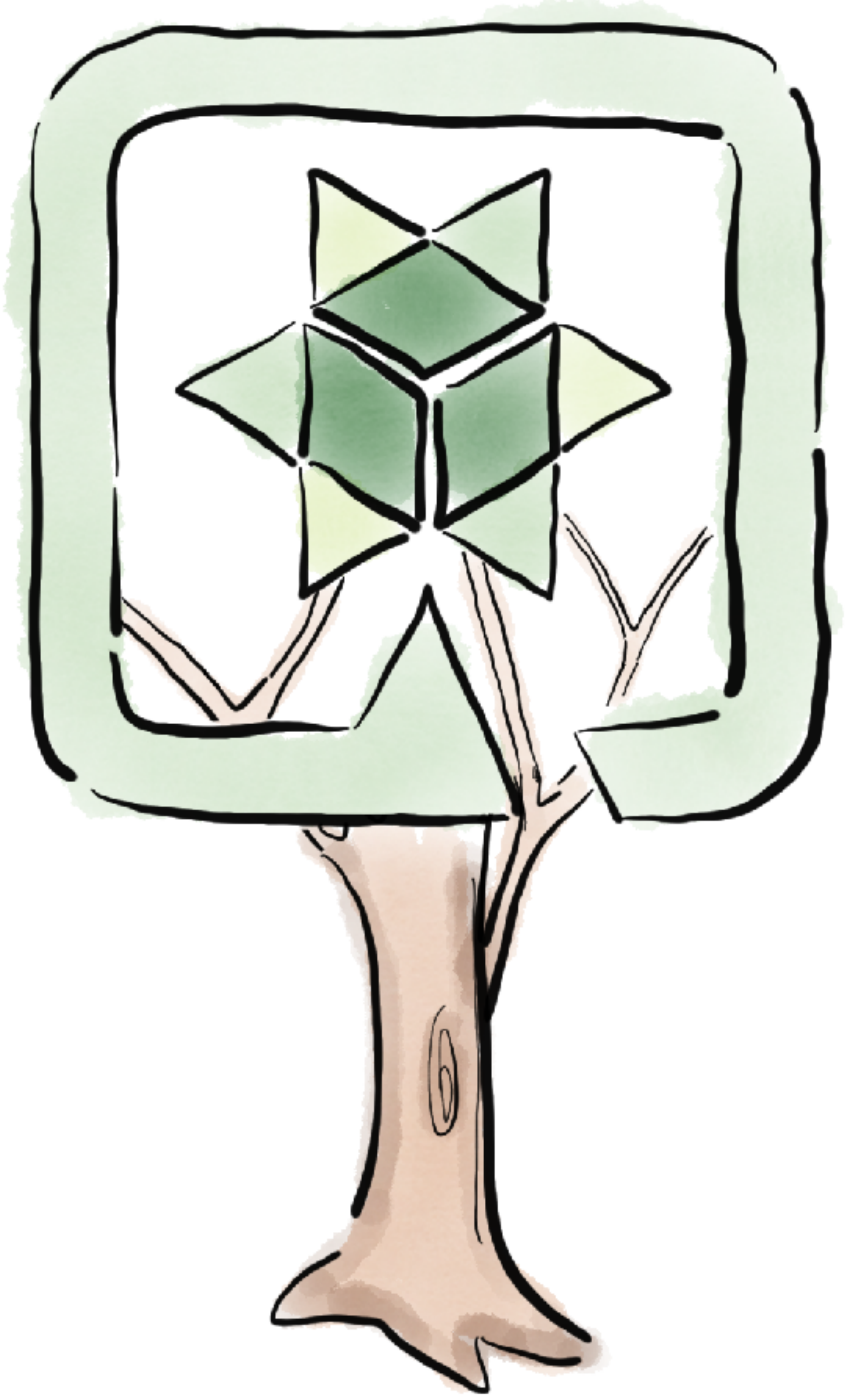


# These five tricks can make your apps greener cheaper & nicer

**Holly Cummins**

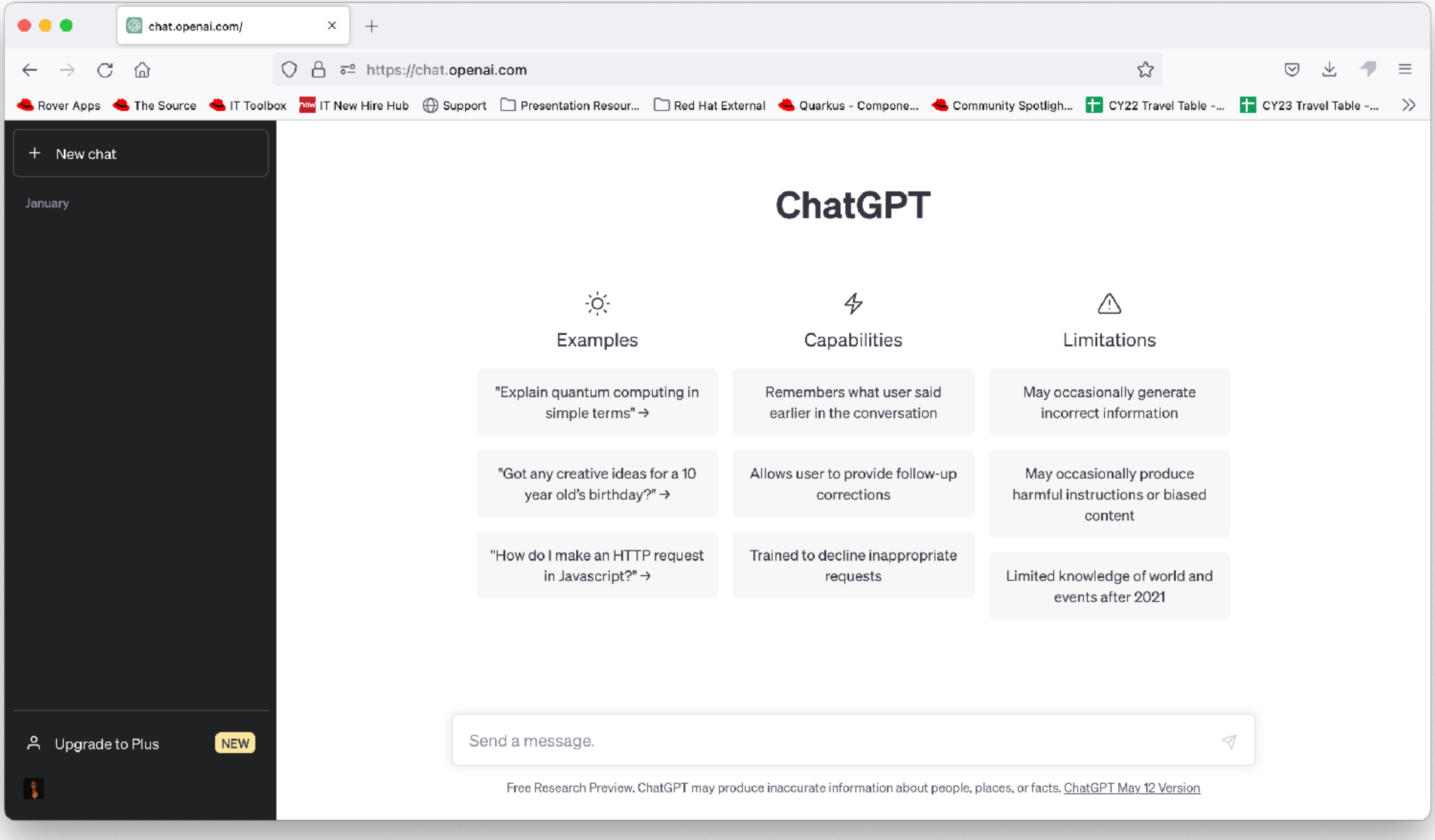
Senior Principal Software Engineer, Quarkus

**GOTO Aarhus**

May 24, 2023

@holly_cummins@hachyderm.io

Red Hat



chat.openai.com/

https://chat.openai.com

Rover Apps · The Source · IT Toolbox · IT New Hire Hub · Support · Presentation Resour... · Red Hat External · Quarkus - Compone... · Community Spotligh... · CY22 Travel Table -... · CY23 Travel Table -...

+ New chat

January

Upgrade to Plus NEW

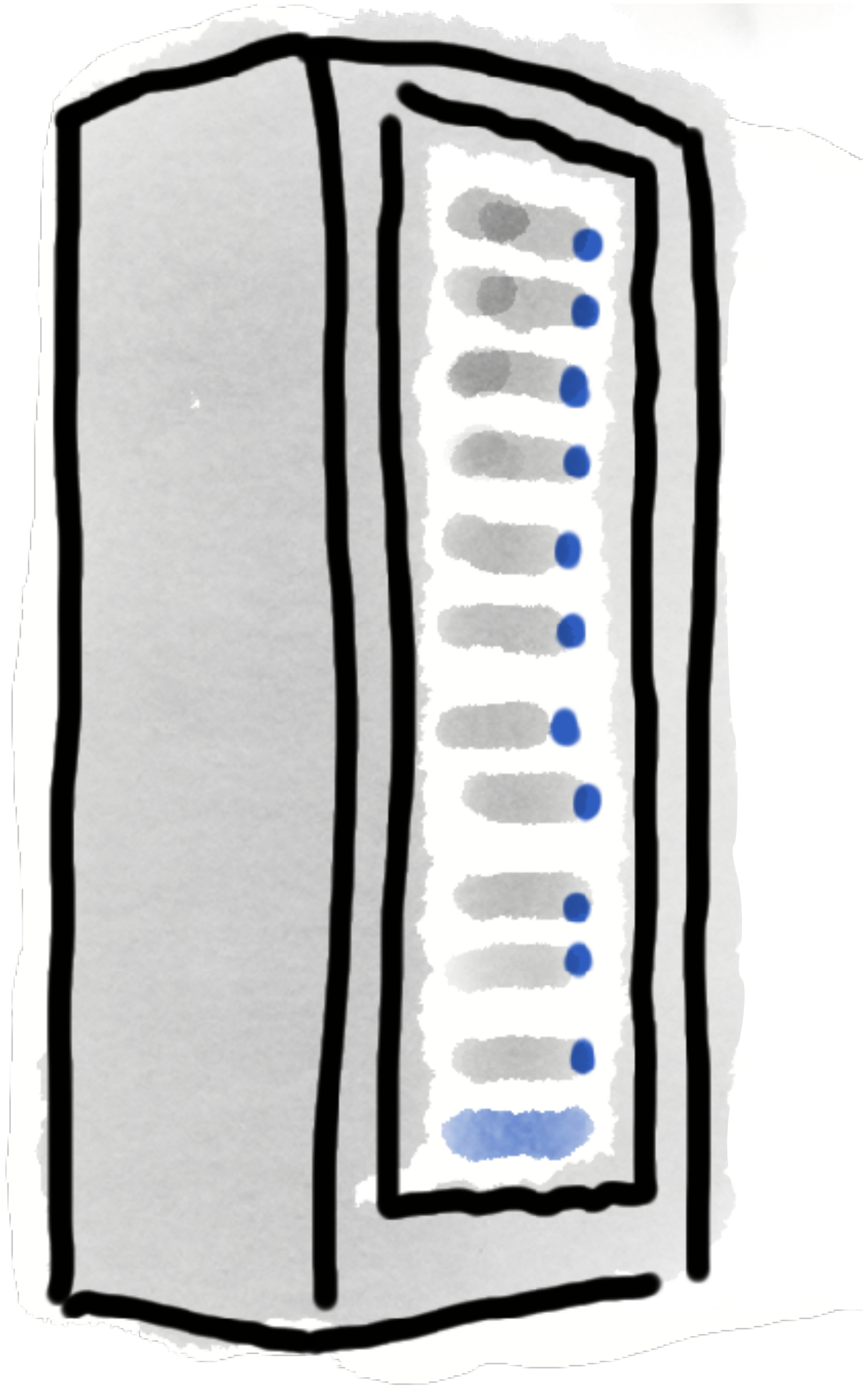
# ChatGPT

| Examples | Capabilities | Limitations |
|---|---|---|
| "Explain quantum computing in simple terms" → | Remembers what user said earlier in the conversation | May occasionally generate incorrect information |
| "Got any creative ideas for a 10 year old's birthday?" → | Allows user to provide follow-up corrections | May occasionally produce harmful instructions or biased content |
| "How do I make an HTTP request in Javascript?" → | Trained to decline inappropriate requests | Limited knowledge of world and events after 2021 |

Send a message.

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 12 Version

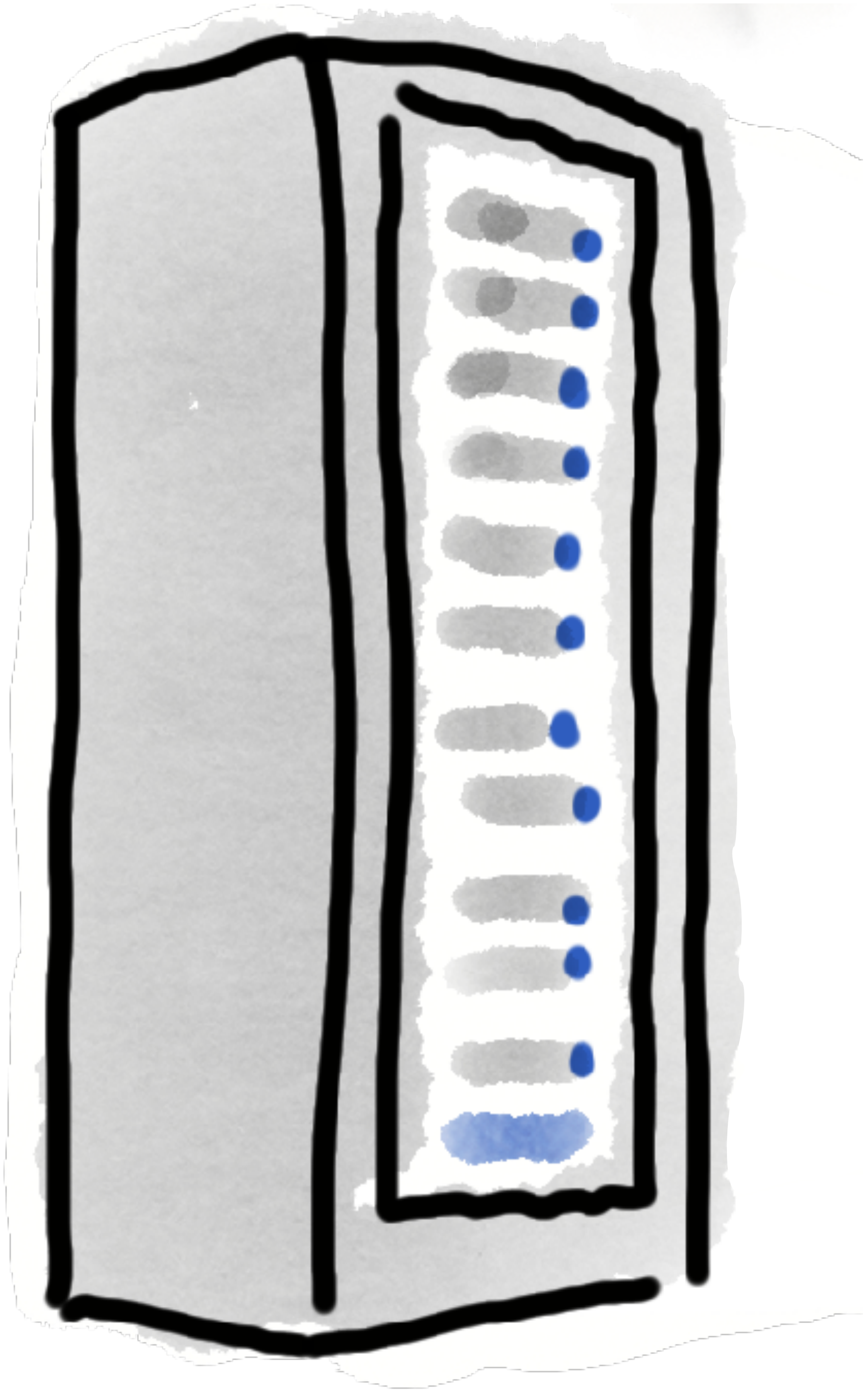
Dario Amodei and Daniel Hernandez. 2018. AI and Compute. https://openai.com/blog/ai-and-compute



compute for training large deep learning models

increase in 6 years

Dario Amodei and Daniel Hernandez. 2018. AI and Compute. https://openai.com/blog/ai-and-compute



compute for training large deep learning models

# 300,000-fold

increase in 6 years

it’s not just artificial intelligence

it’s not just artificial intelligence

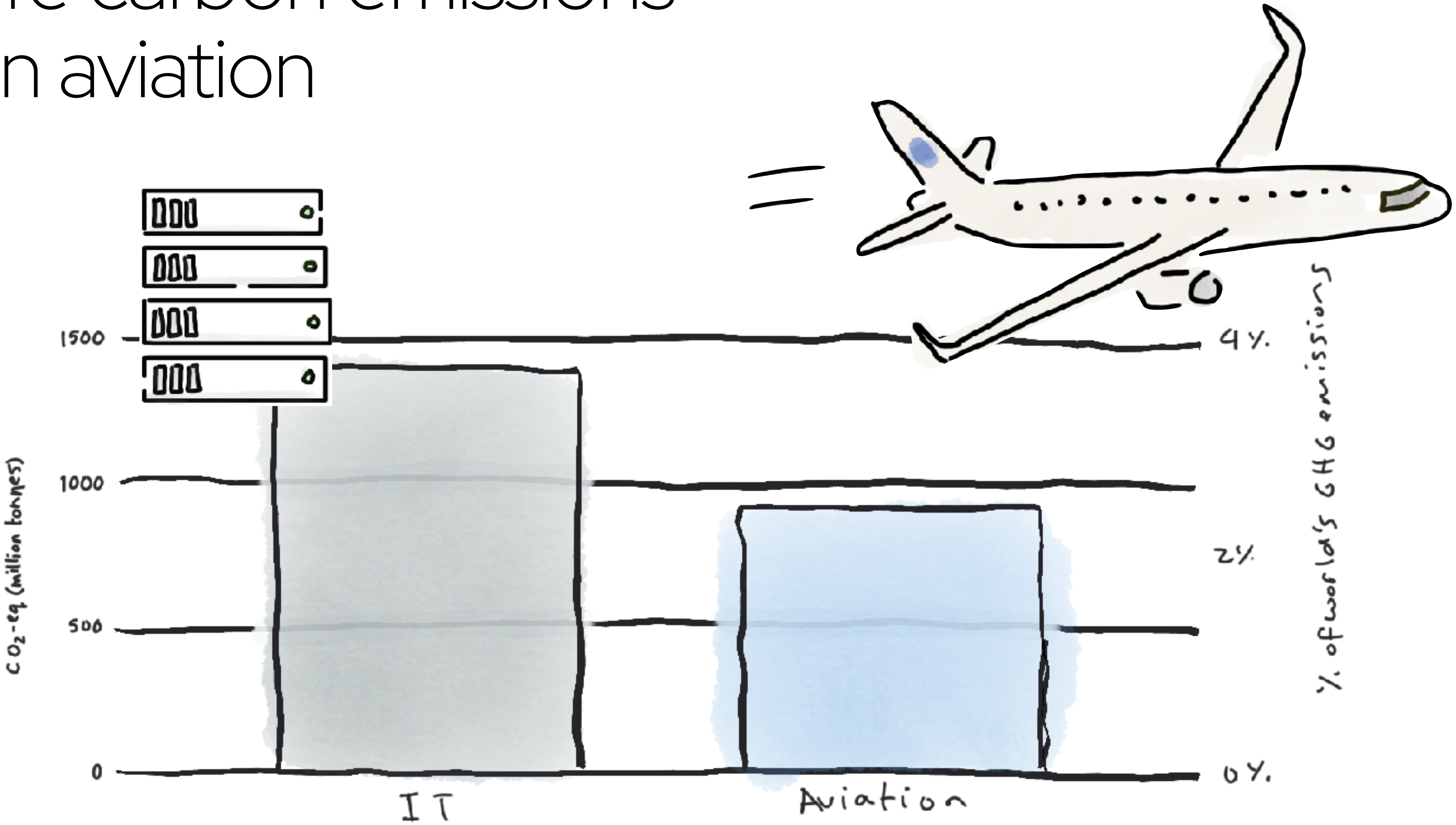
it’s not just cryptocurrency mining

it’s not just artificial intelligence

it’s not just cryptocurrency mining

**it’s all of us**

# the digital world creates more carbon emissions than aviation

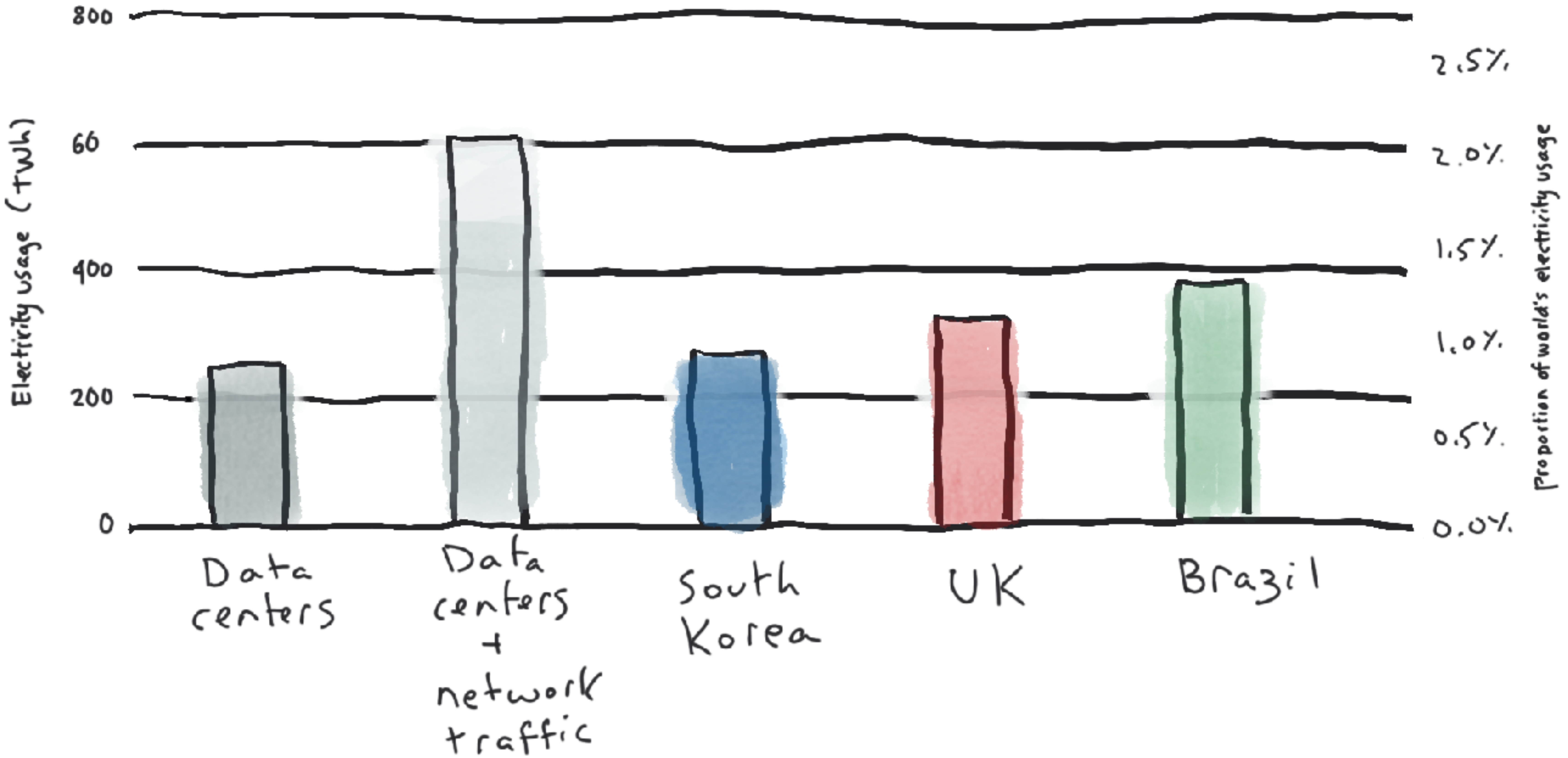


**Sources:**
https://www.greenit.fr/wp-content/uploads/2019/11/GREENIT_EENM_etude_EN_accessible.pdf
https://ourworldindata.org/ghg-emissions-by-sector

Lighter area represents high and low estimates, where available.

**Sources:**
https://www.iea.org/fuels-and-technologies/data-centres-networks
https://ourworldindata.org/grapher/electricity-demand?tab=table&country=USA~GBR~FRA~DEU~IND~BRA

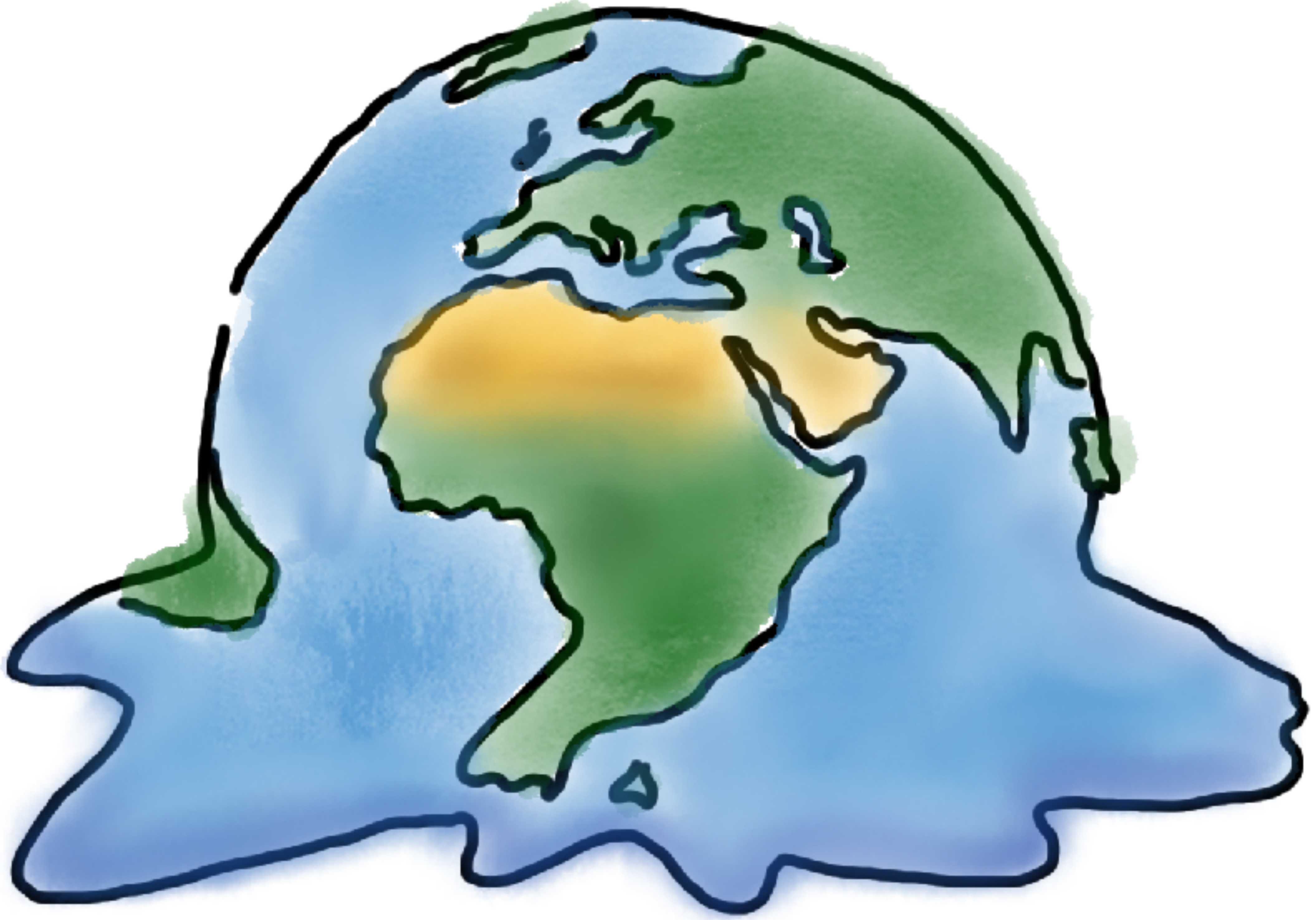
# data centres use as much electricity as a medium country



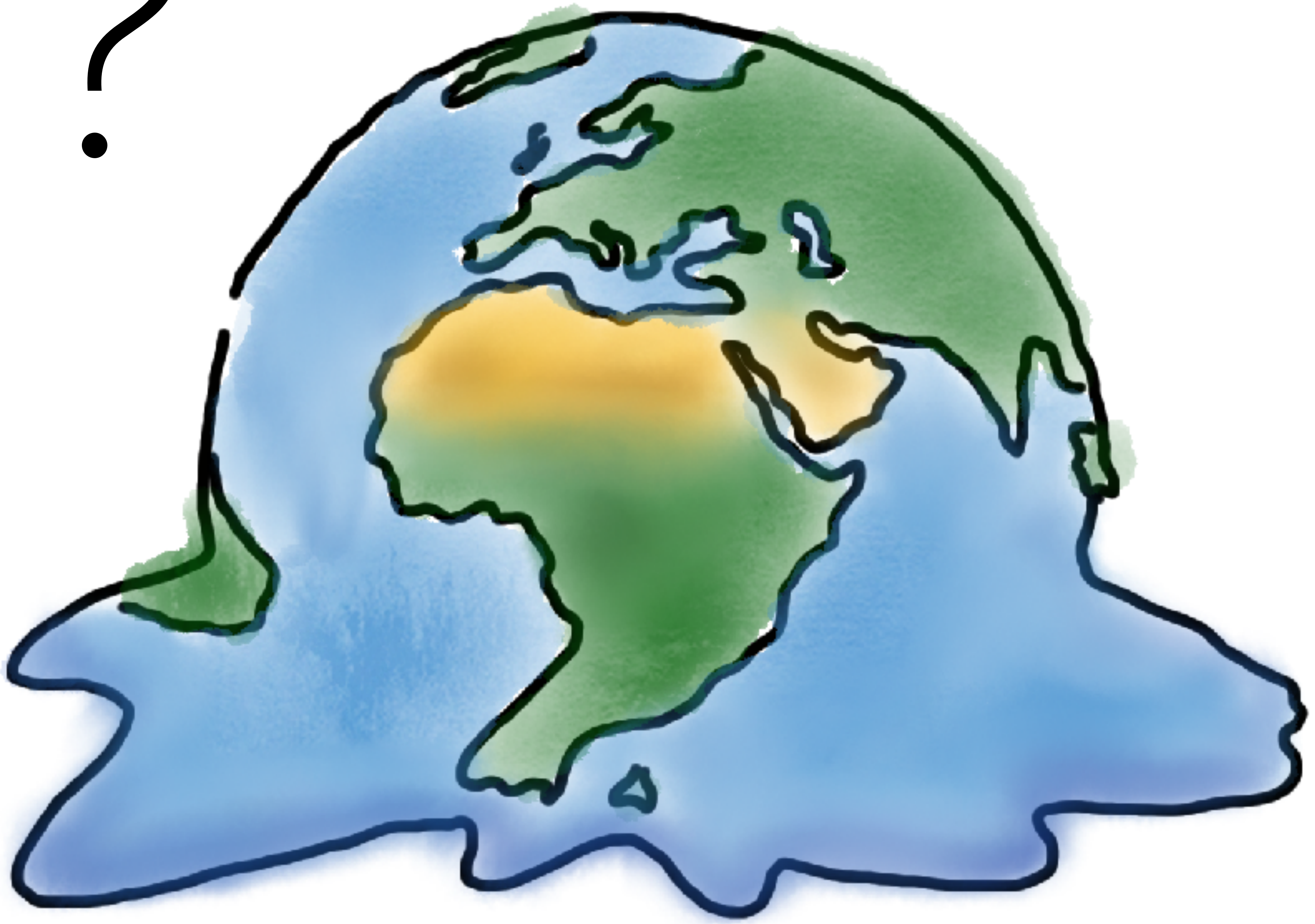
@holly_cummins

#RedHat

# aaaaaaaaaargh



# aaaaaaaaargh?



# be a solutionist

how do we do make solutions?

# green software foundation: principles

# green software foundation: principles

carbon awareness

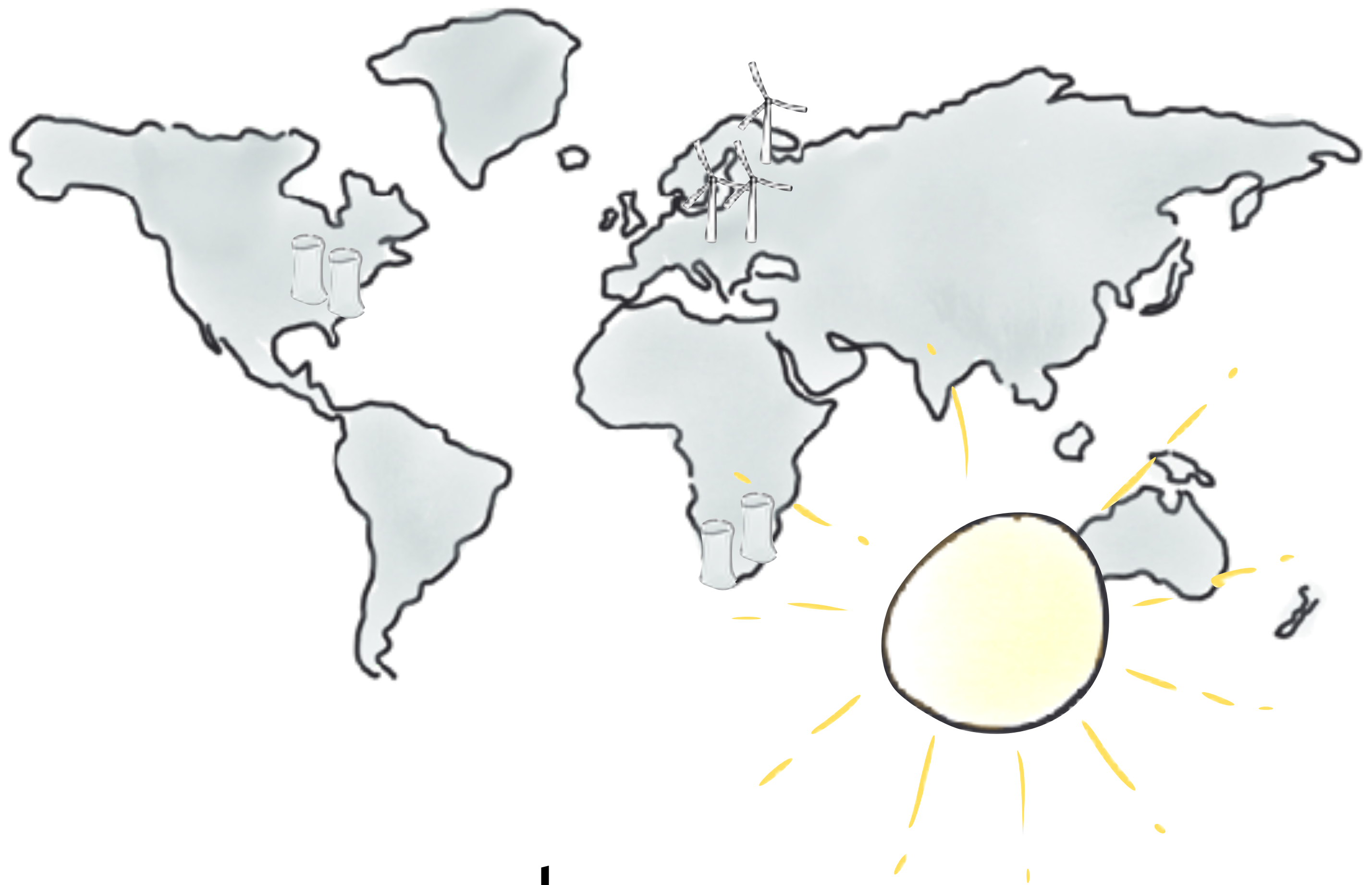
# green software foundation: principles

carbon awareness

where

# green software foundation: principles

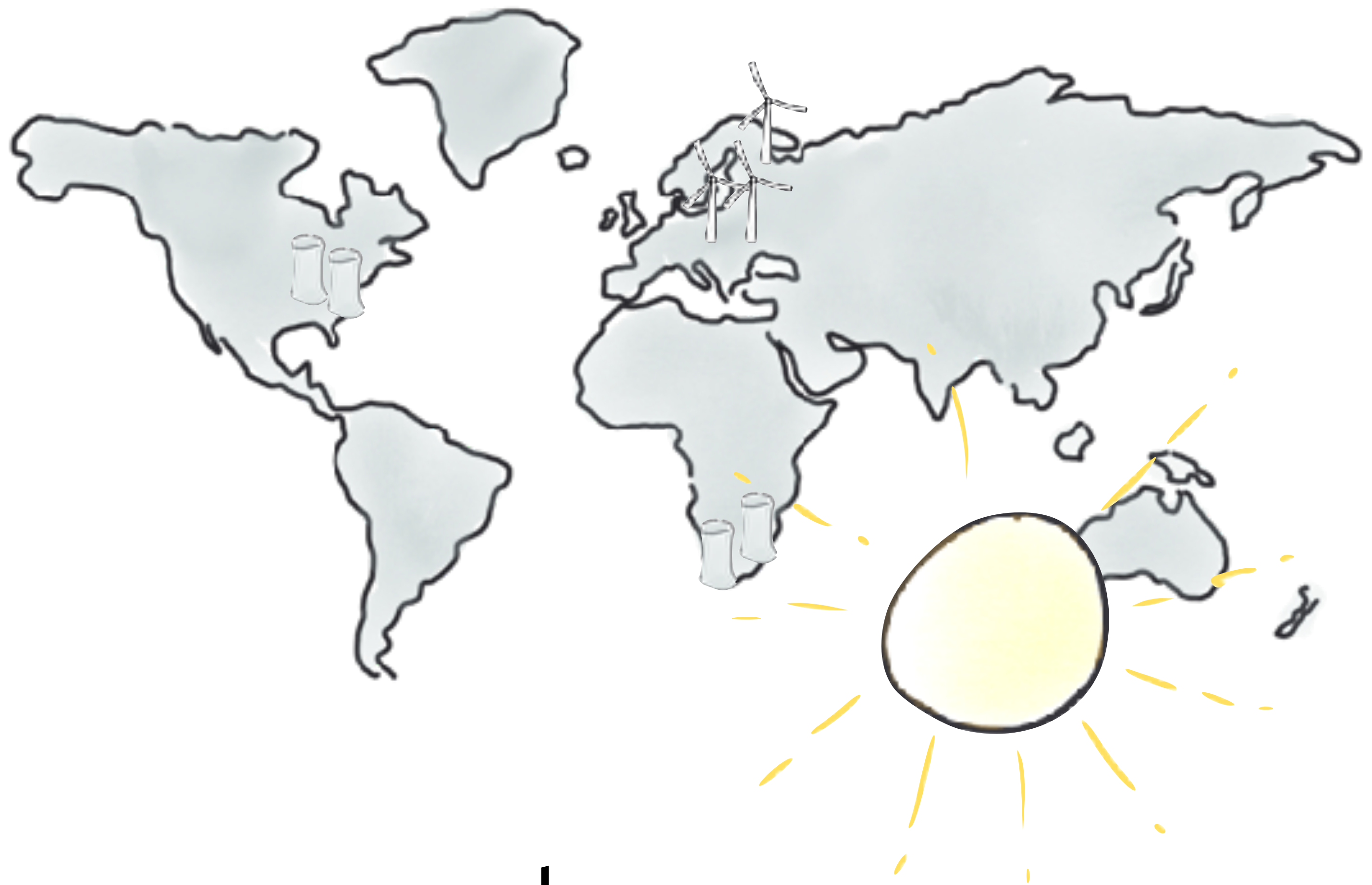
carbon awareness



where
when

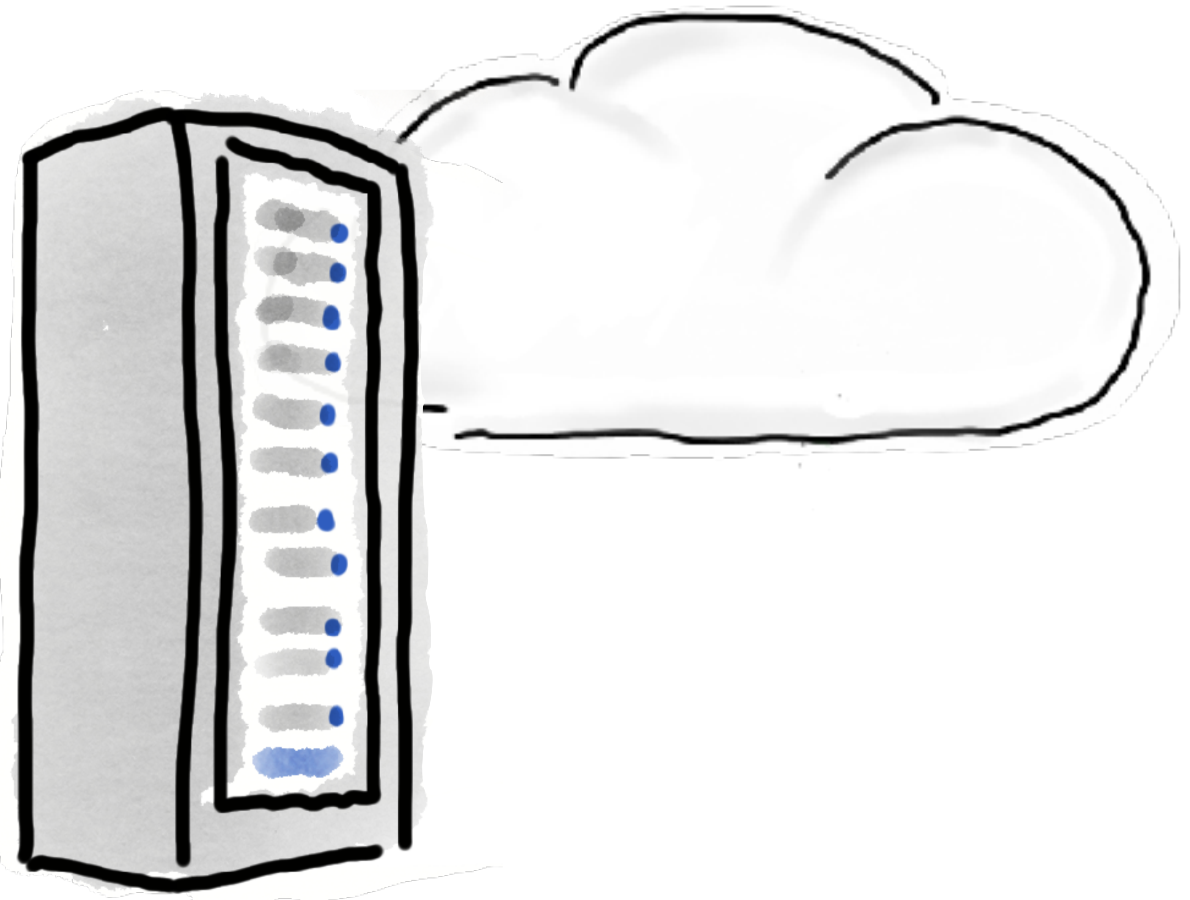
# green software foundation: principles

carbon awareness



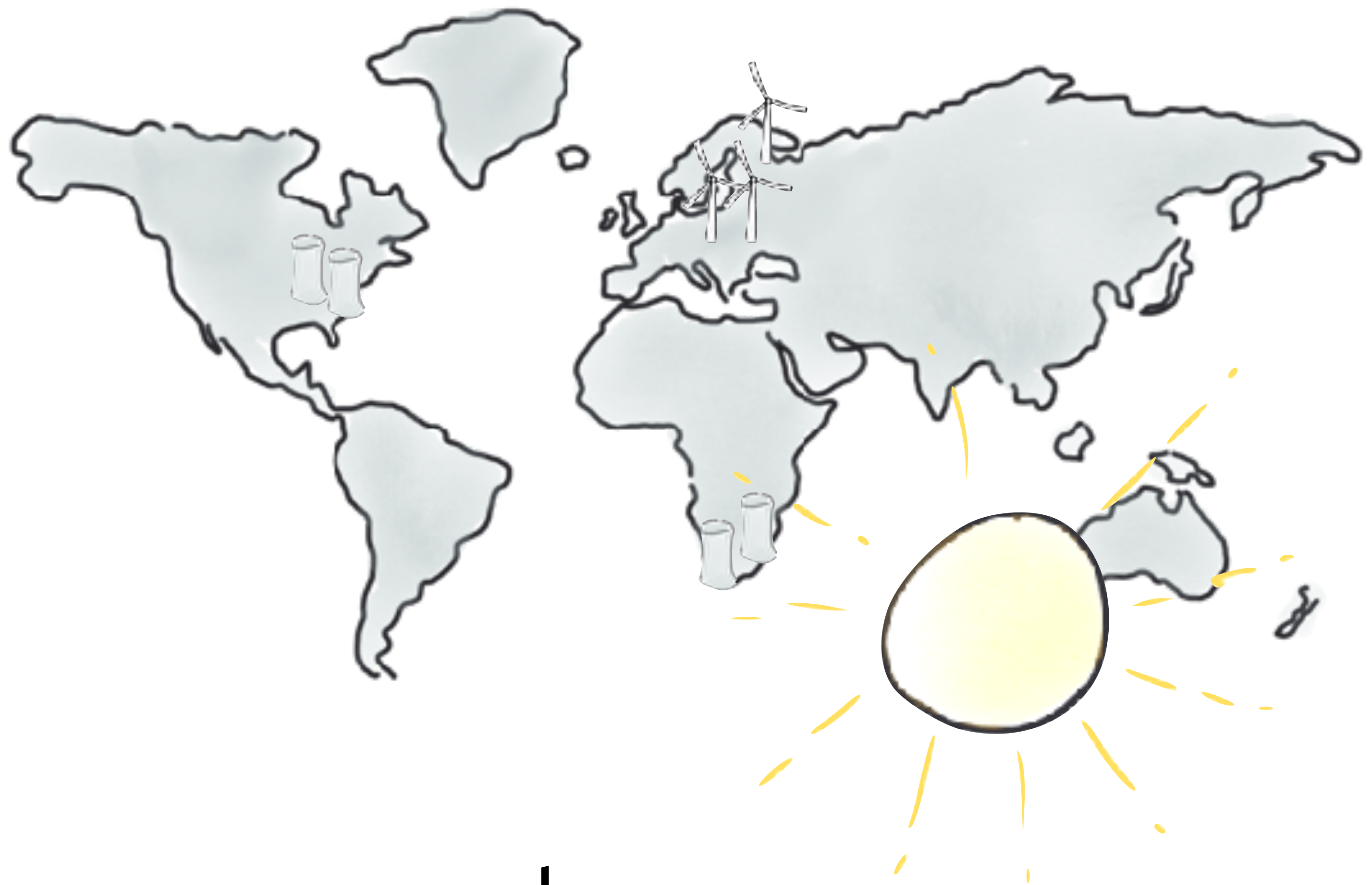
where
when

hardware efficiency



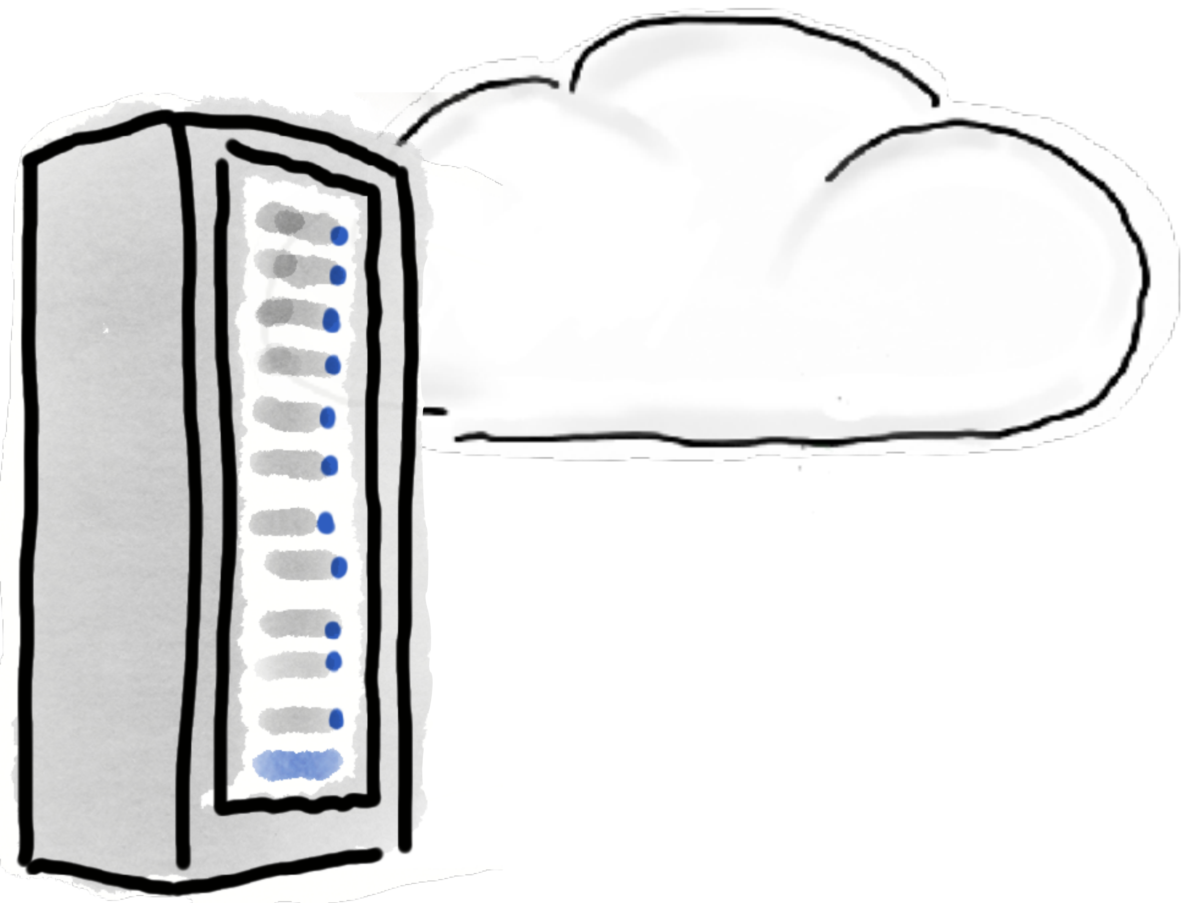
# green software foundation: principles

carbon awareness



where
when

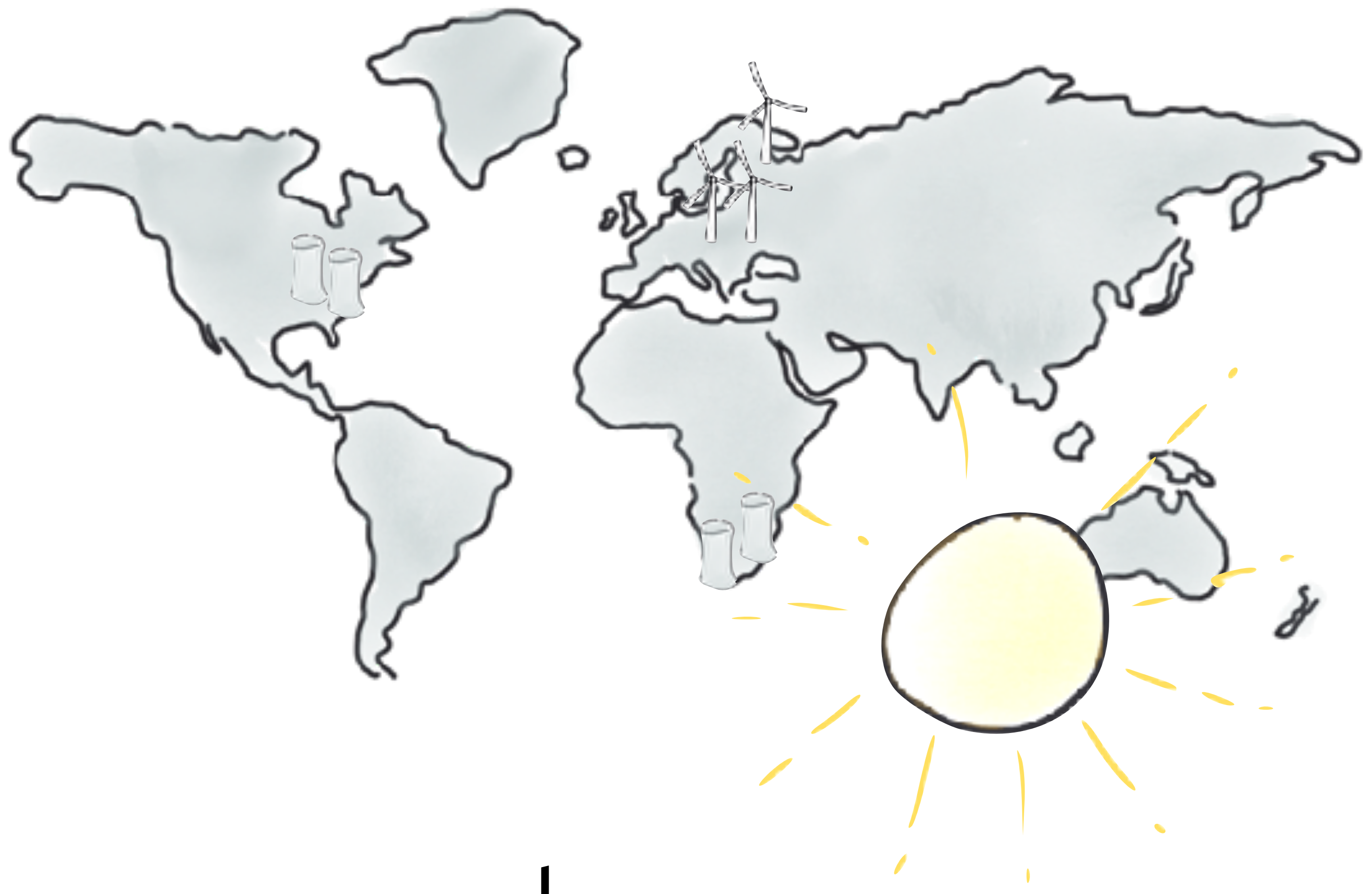
hardware efficiency



elasticity

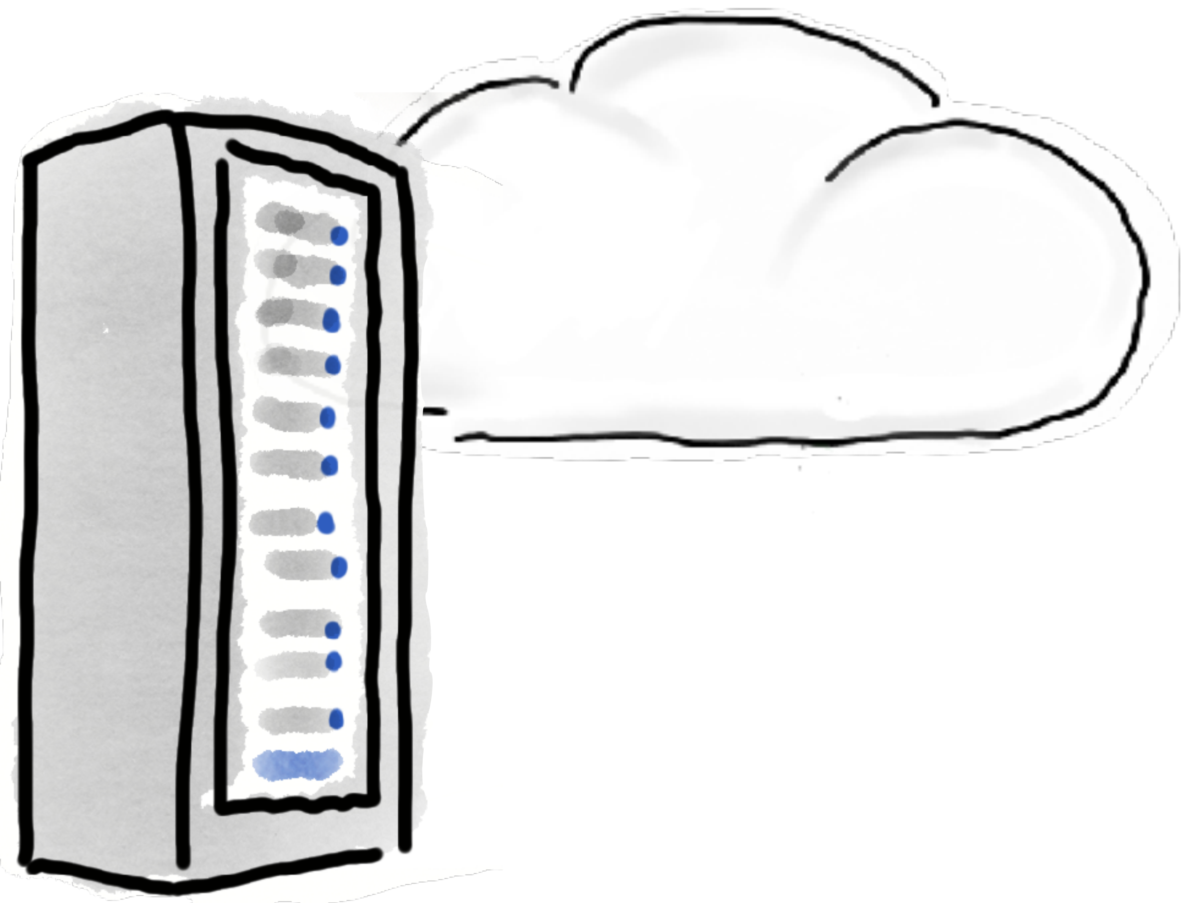
# green software foundation: principles

carbon awareness



where
when

hardware efficiency

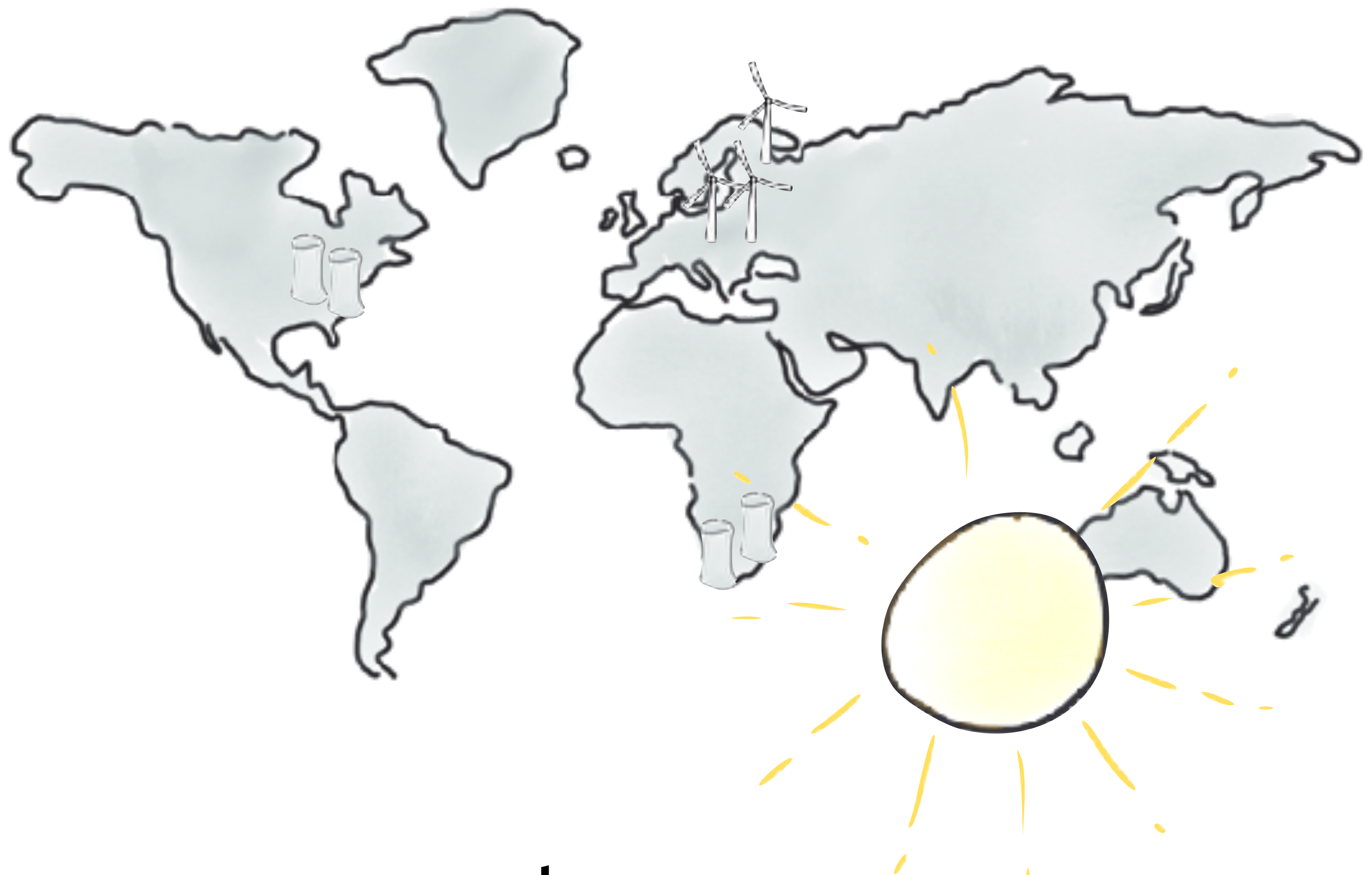


elasticity

utilisation

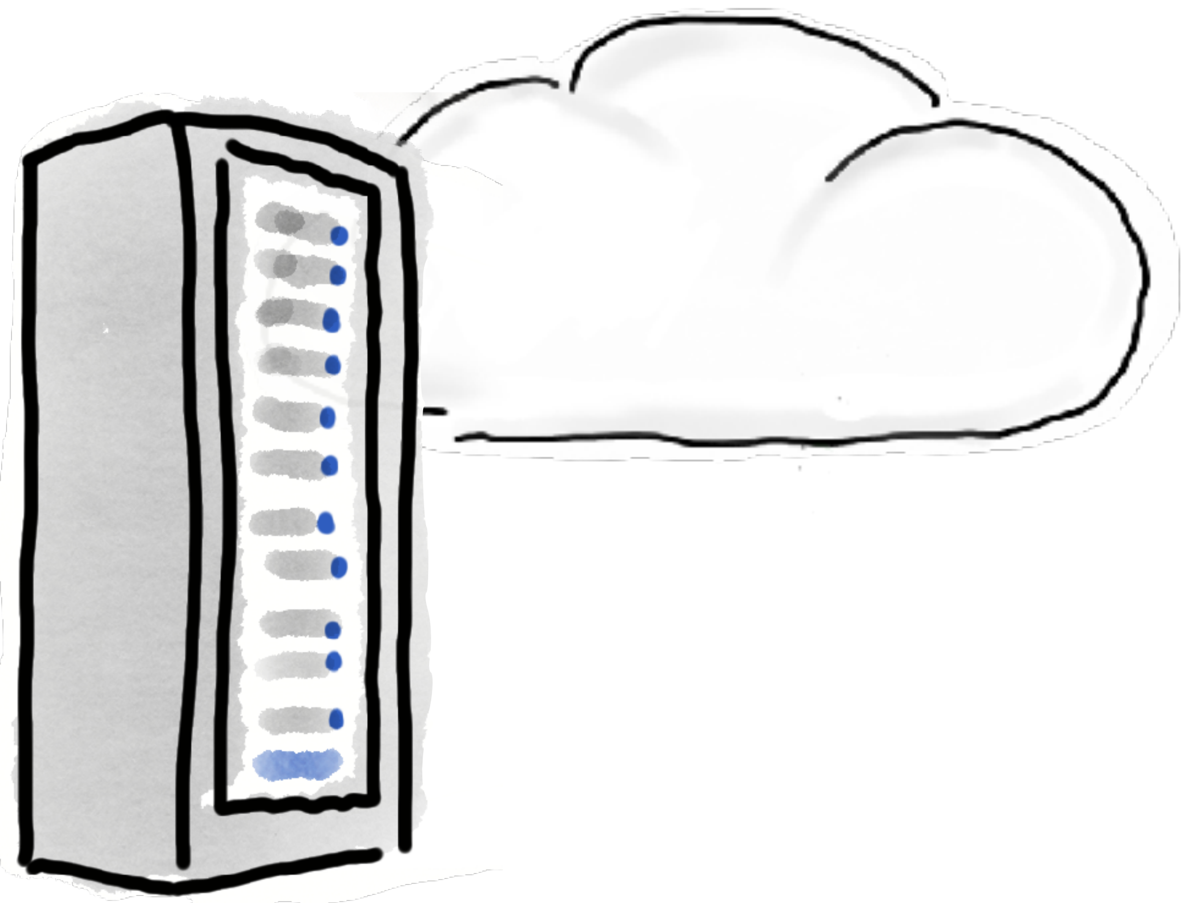
# green software foundation: principles

carbon awareness



where
when

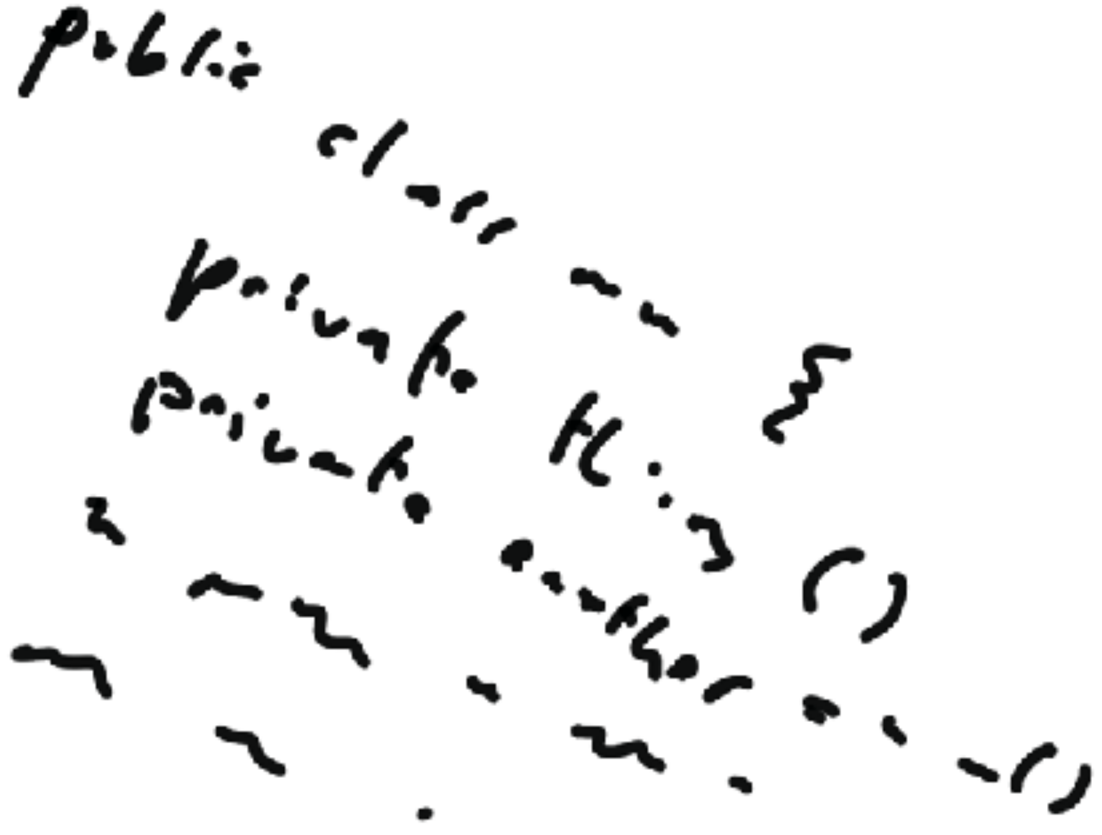
hardware efficiency



elasticity

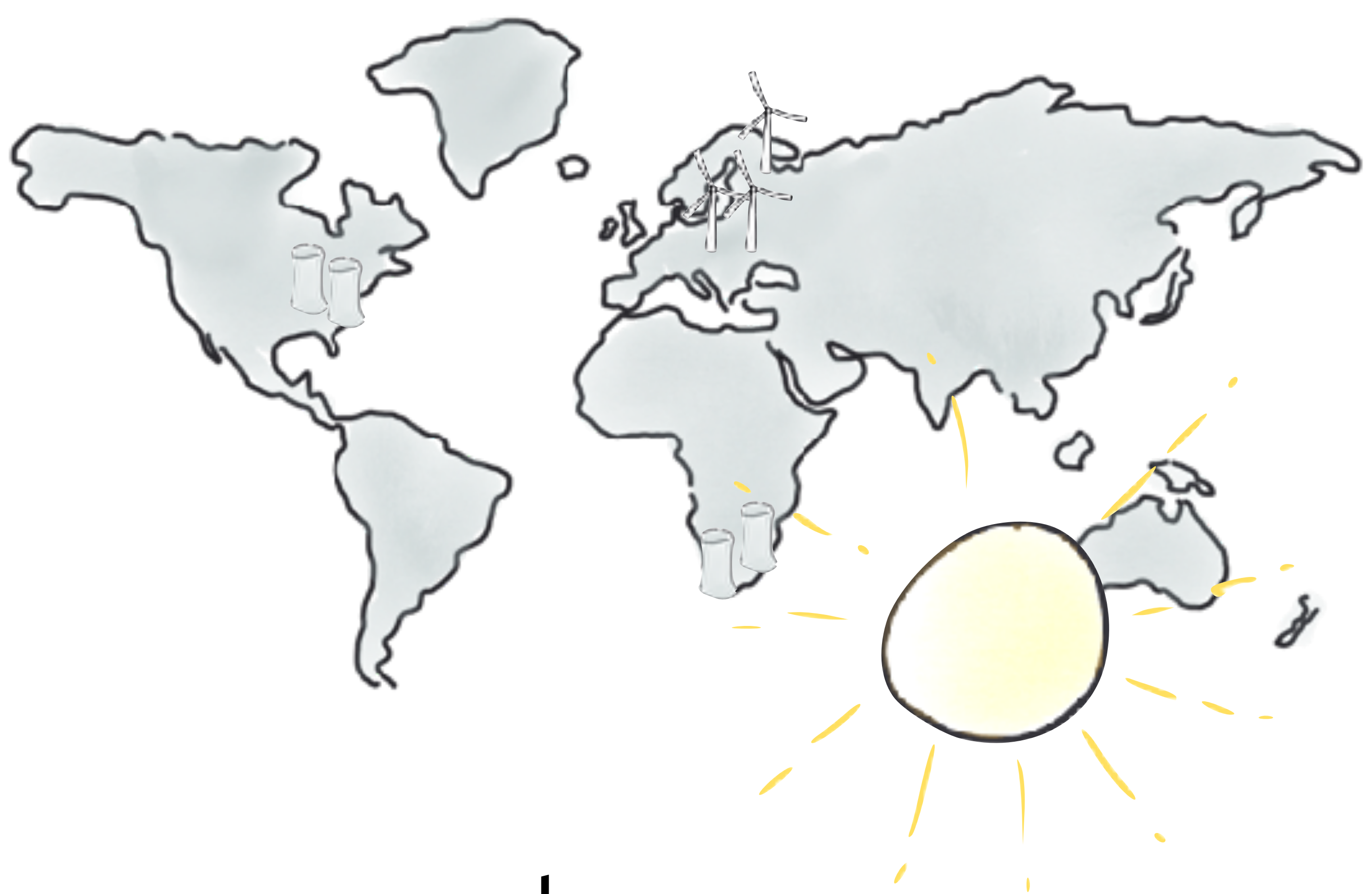
utilisation

electricity efficiency



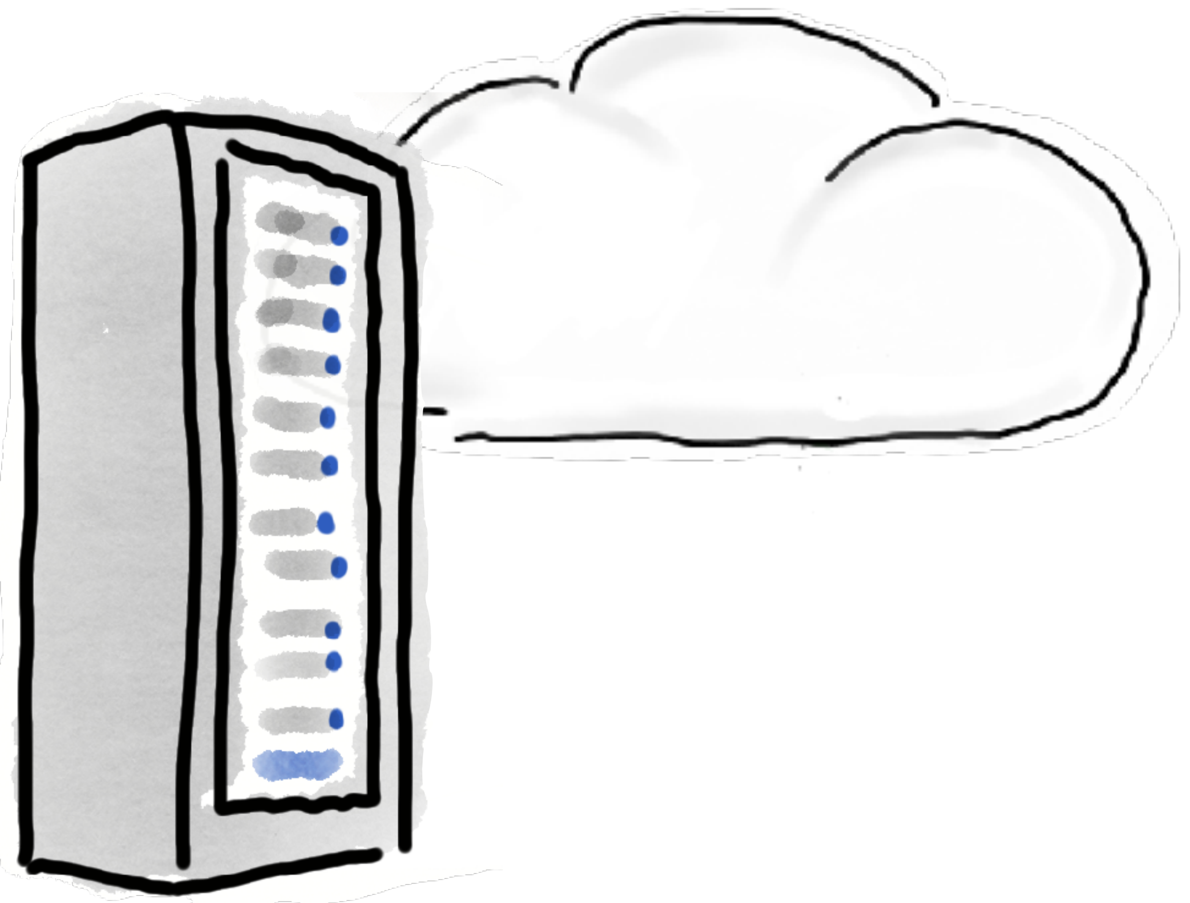
# green software foundation: principles

## carbon awareness



where
when

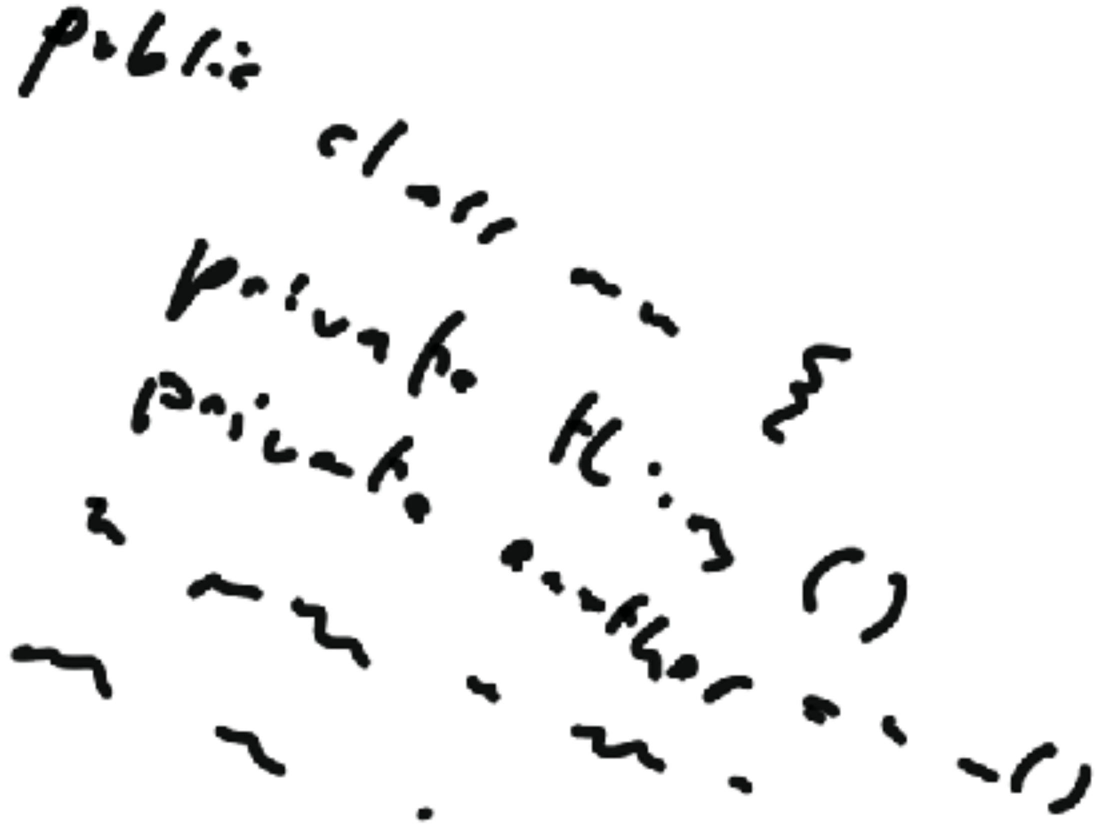
## hardware efficiency



elasticity

utilisation

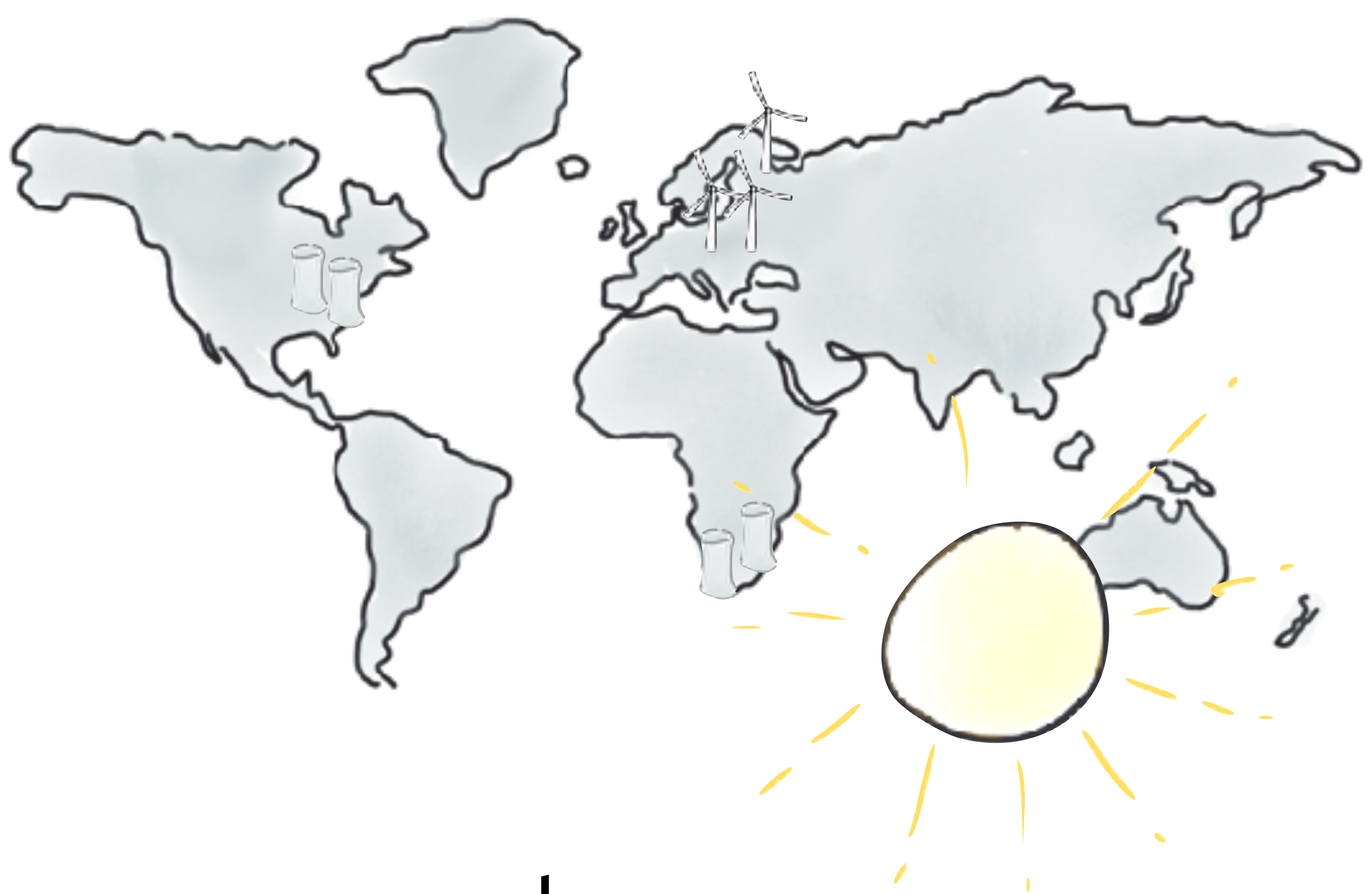
## electricity efficiency



algorithms

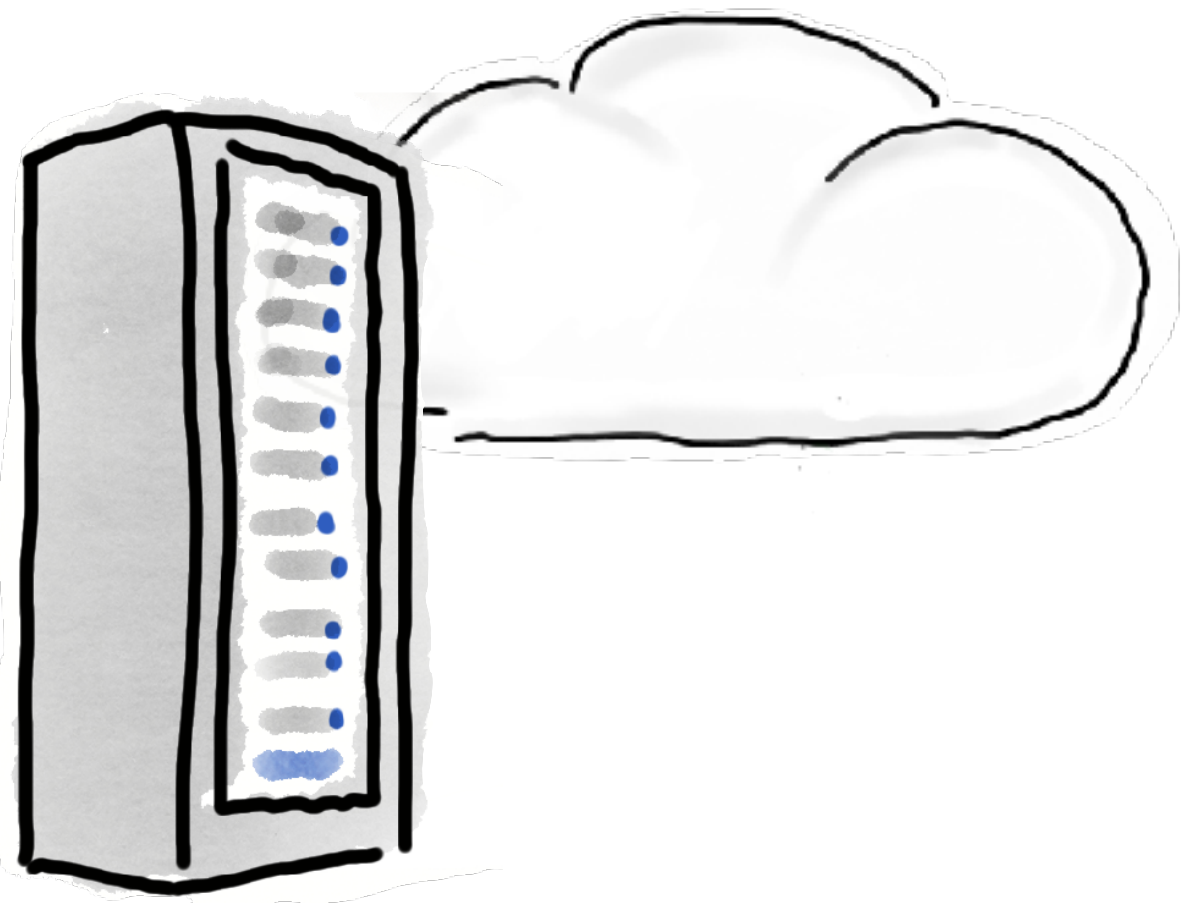
# green software foundation: principles

## carbon awareness



where
when

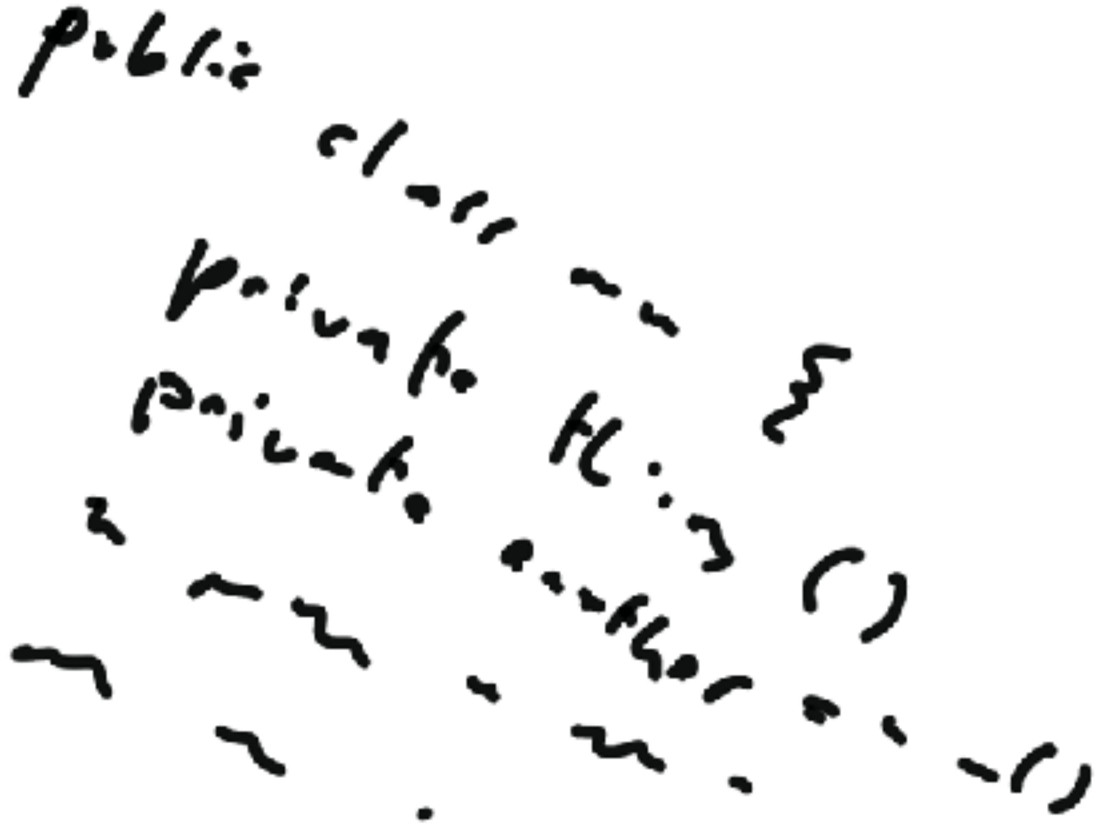
## hardware efficiency



elasticity

utilisation

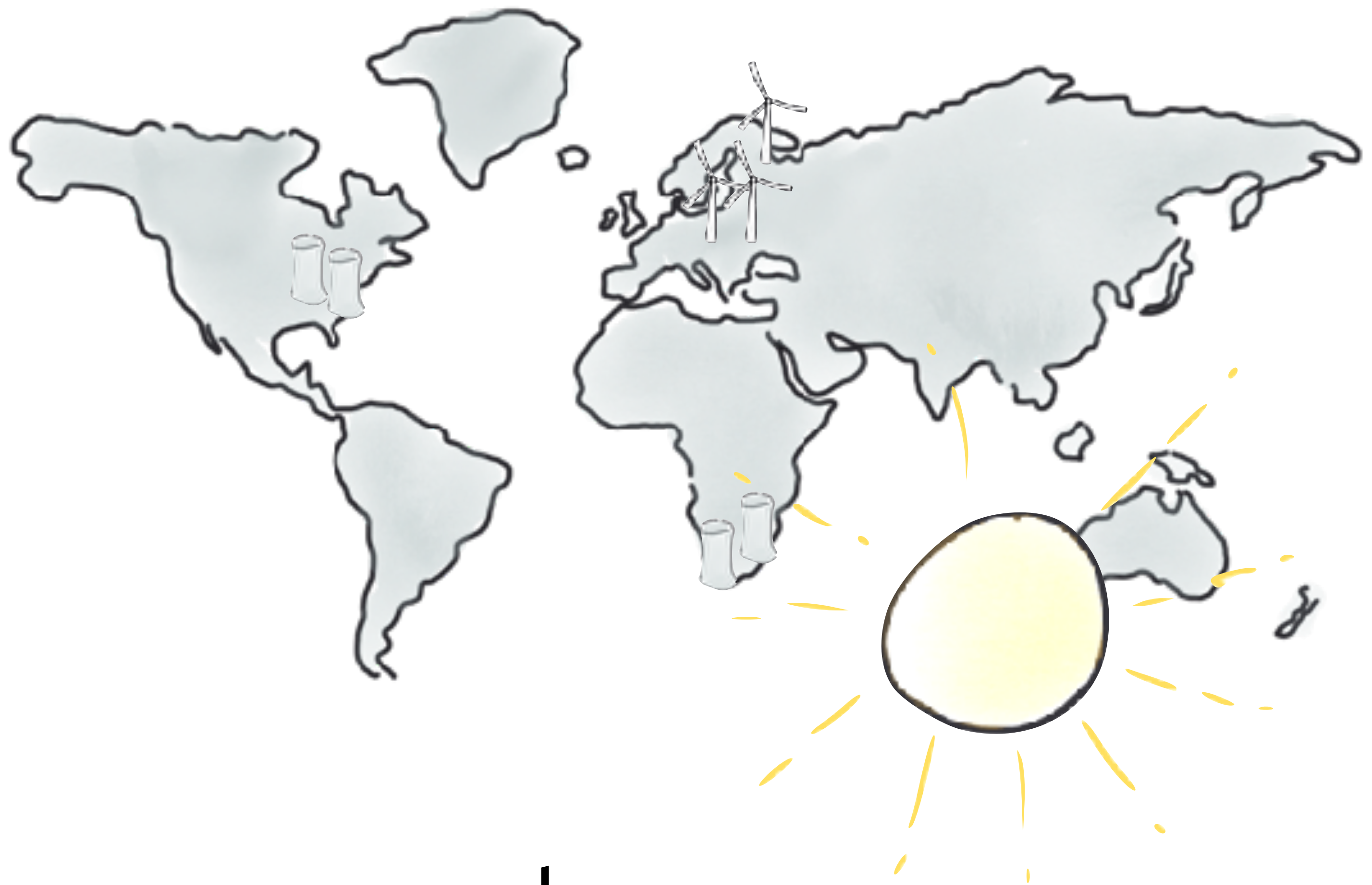
## electricity efficiency



algorithms
stack

# green software foundation: principles

## carbon awareness



where
when

## hardware efficiency

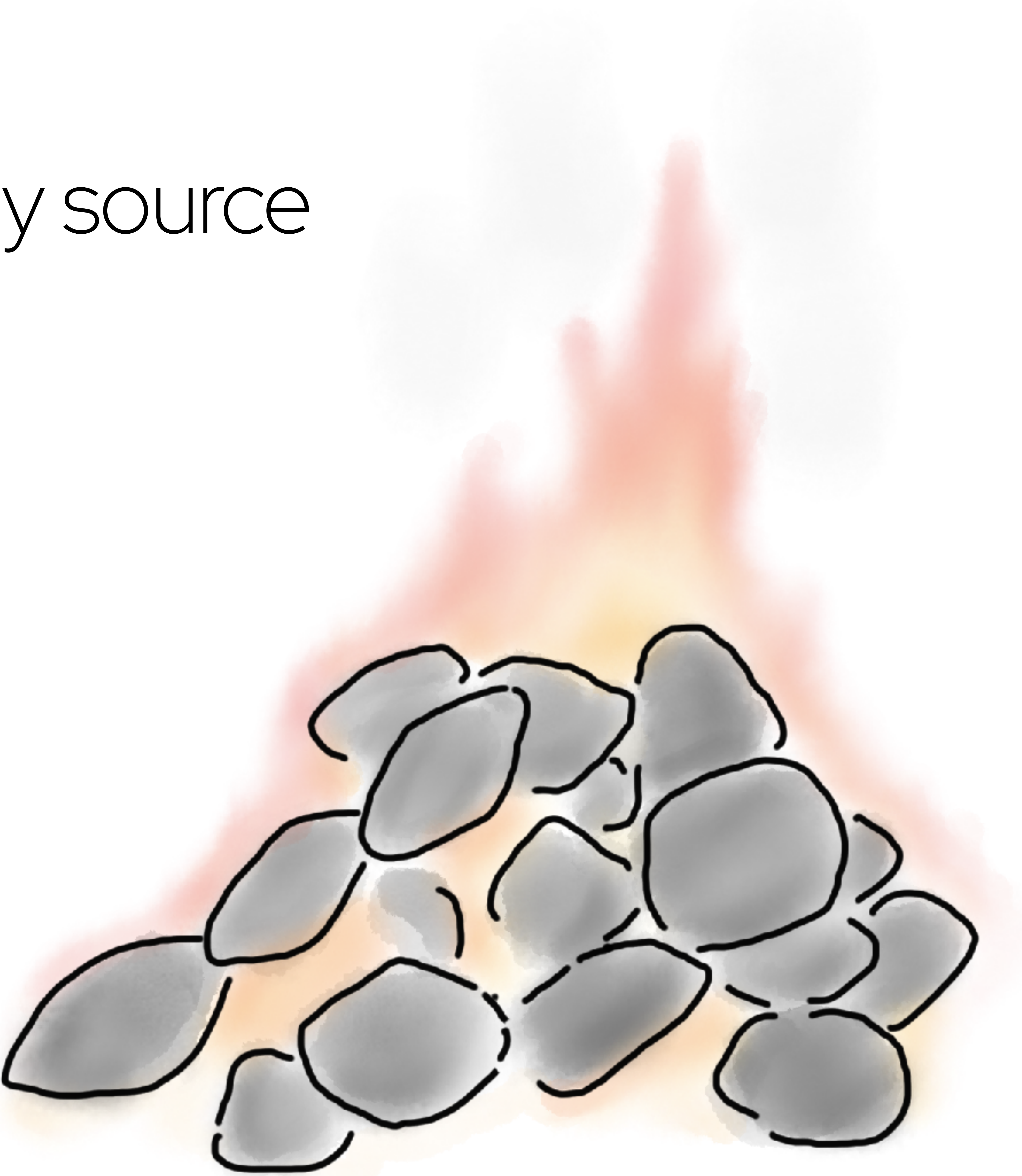
elasticity

utilisation

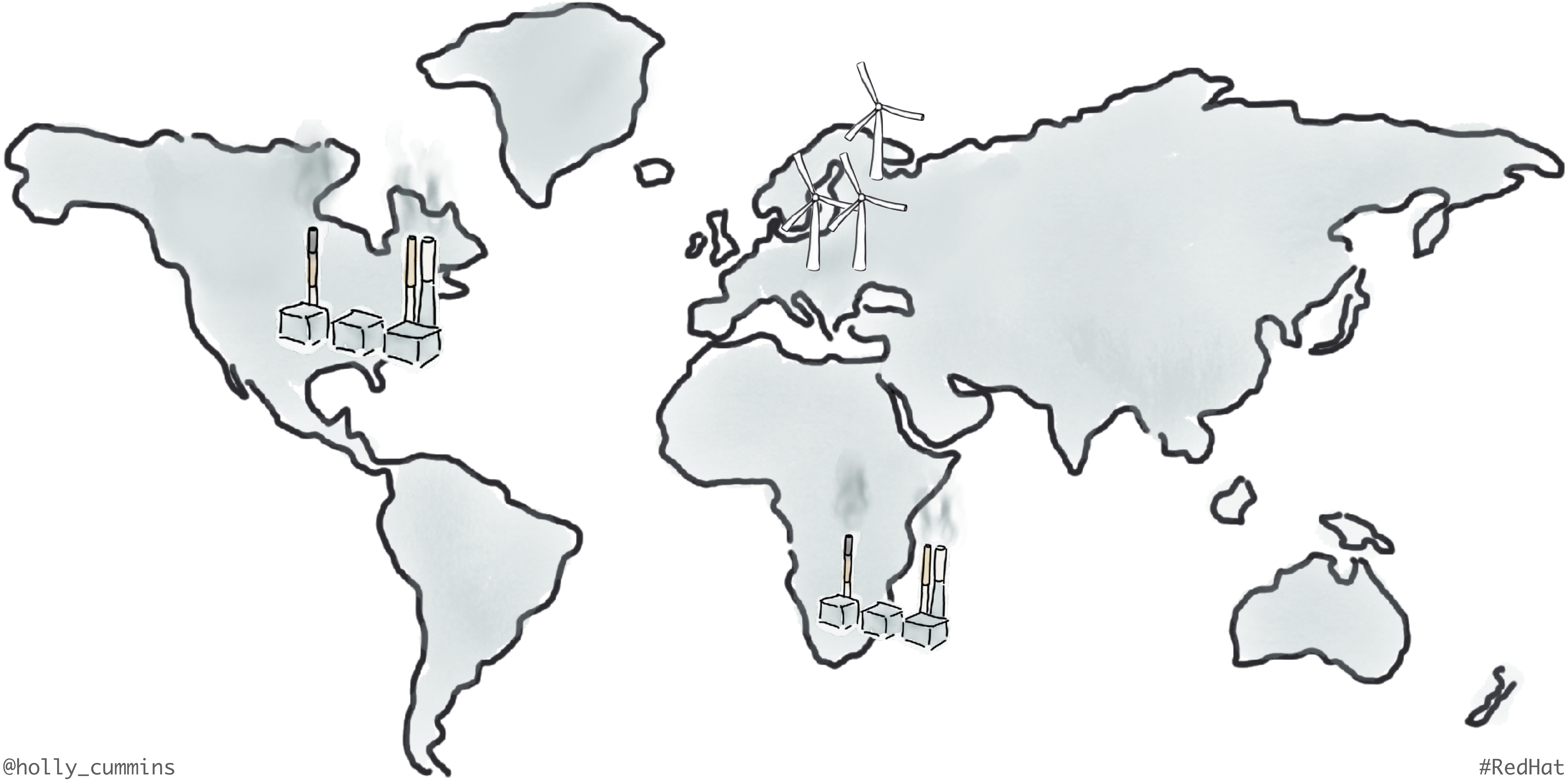
## electricity efficiency

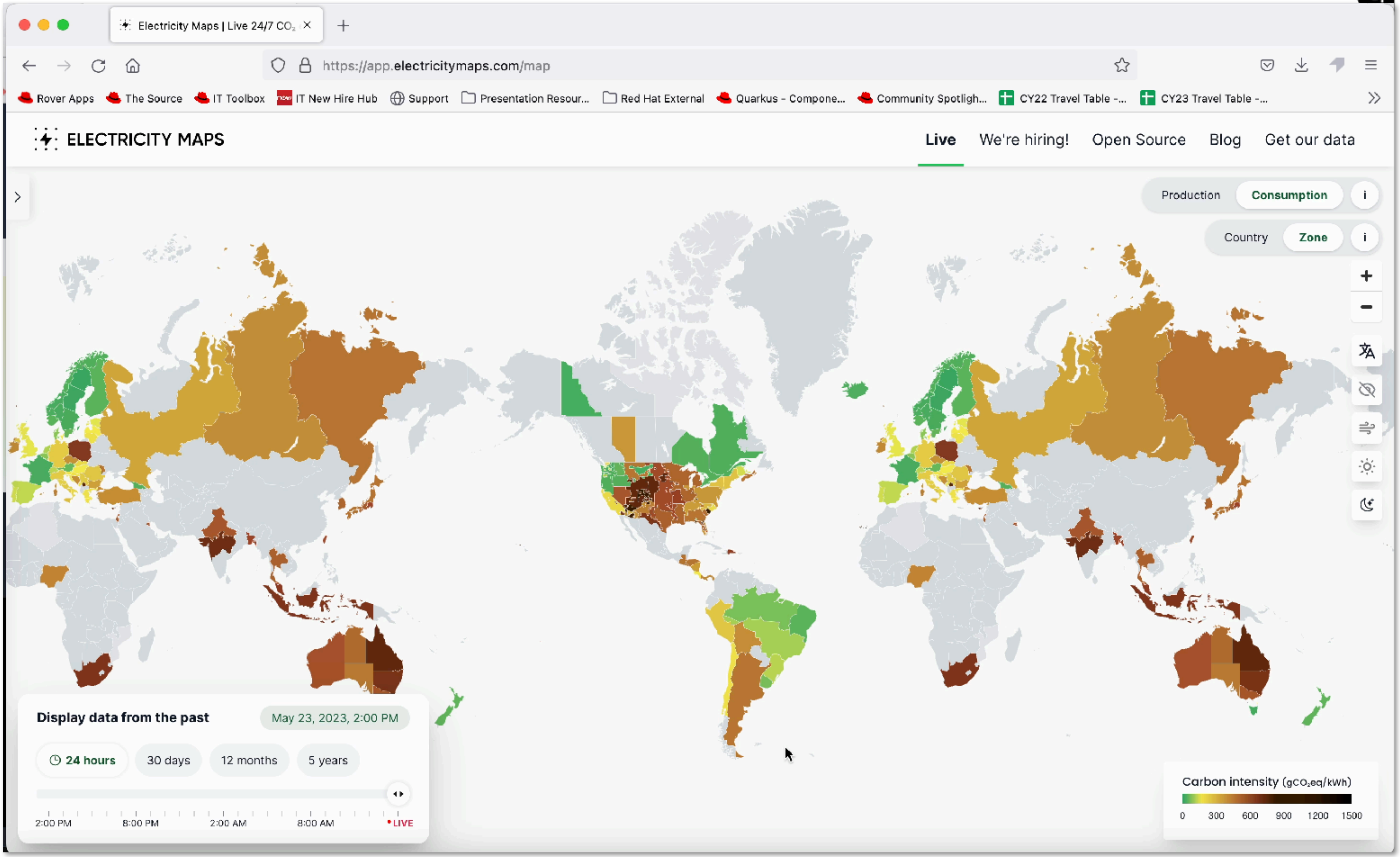
algorithms
stack

# trick 1: electricity source



# data center location matters





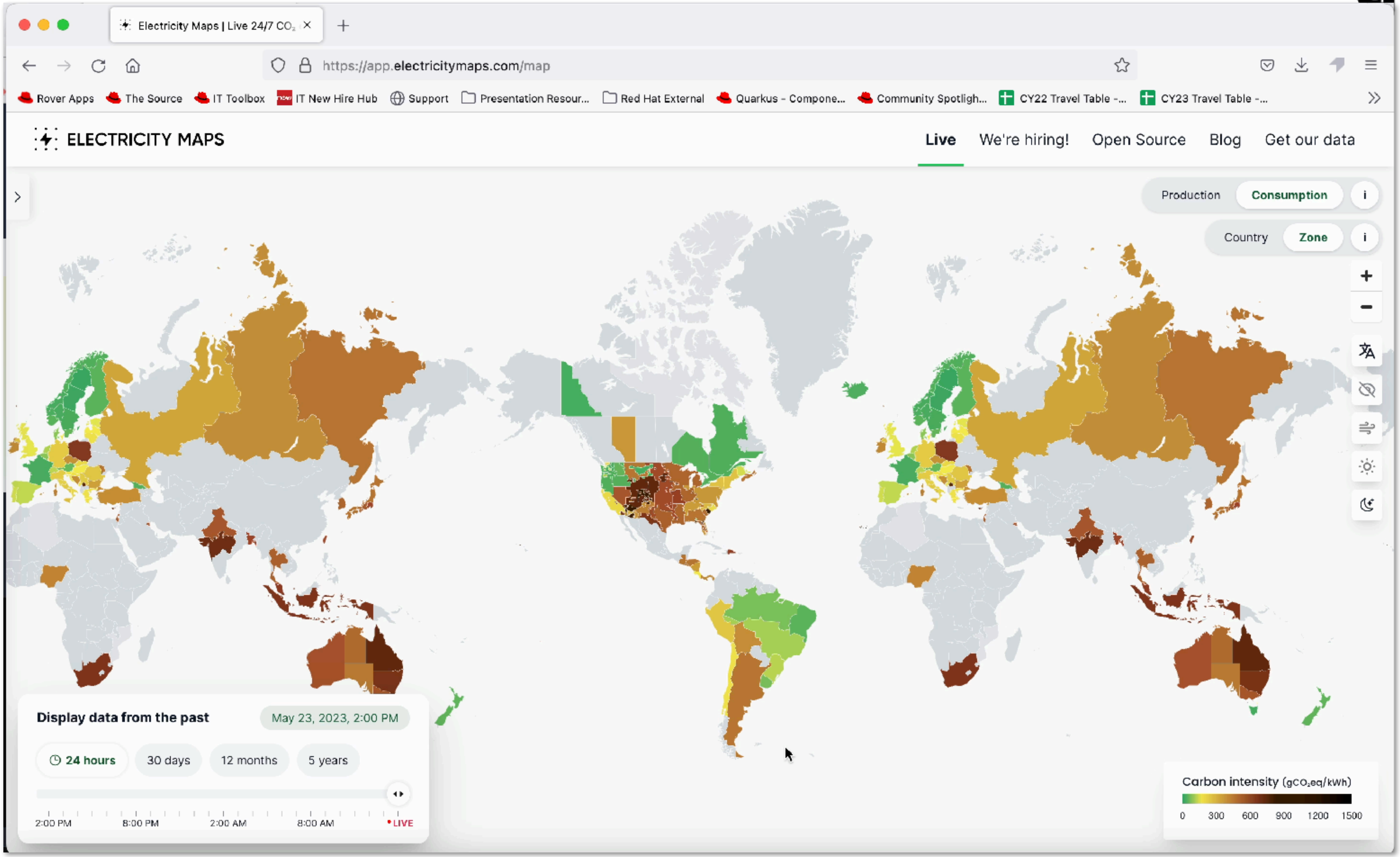
https://app.electricitymaps.com/map

Display data from the past: May 23, 2023, 2:00 PM

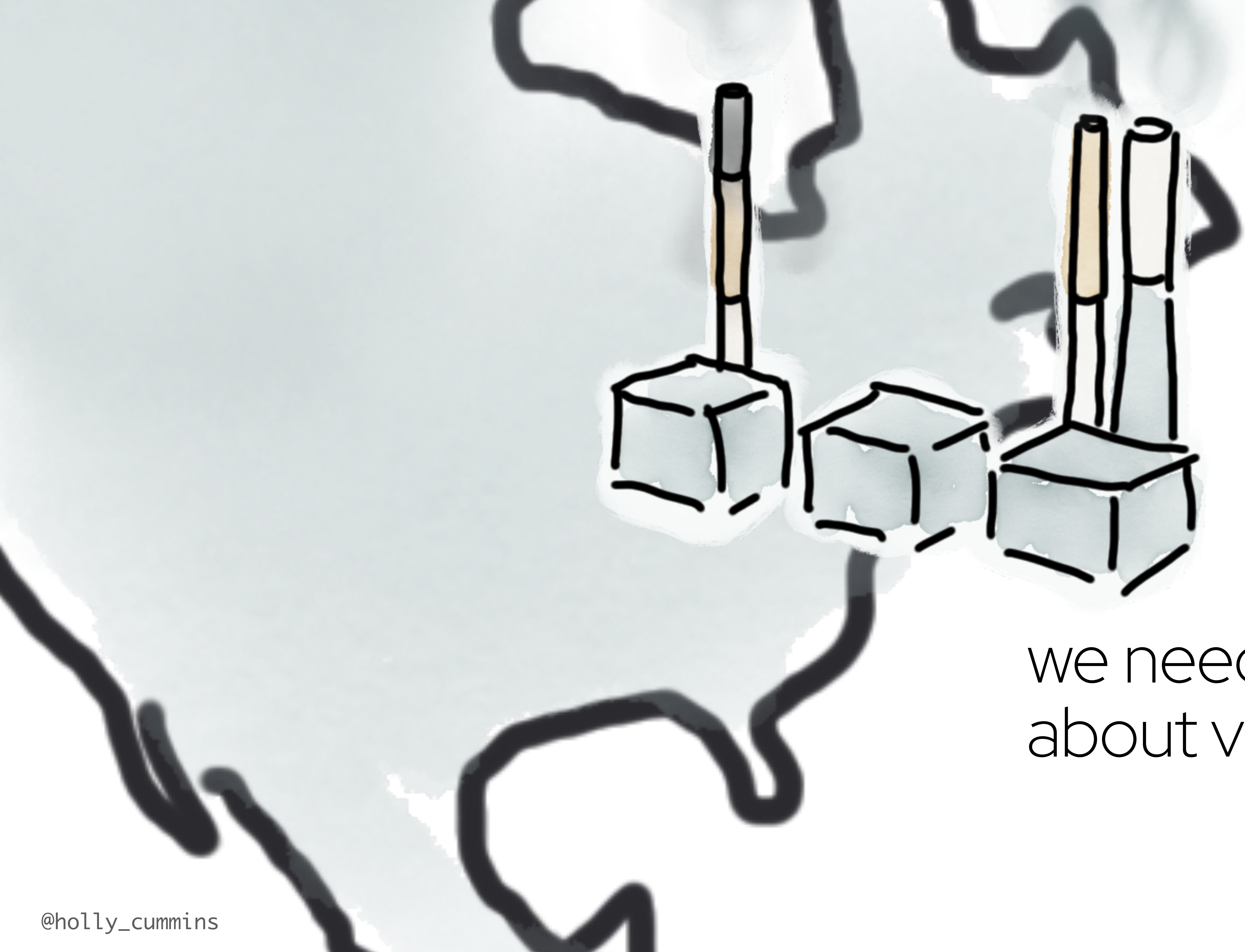
Consumption

Zone

24 hours



Display data from the past: May 23, 2023, 2:00 PM



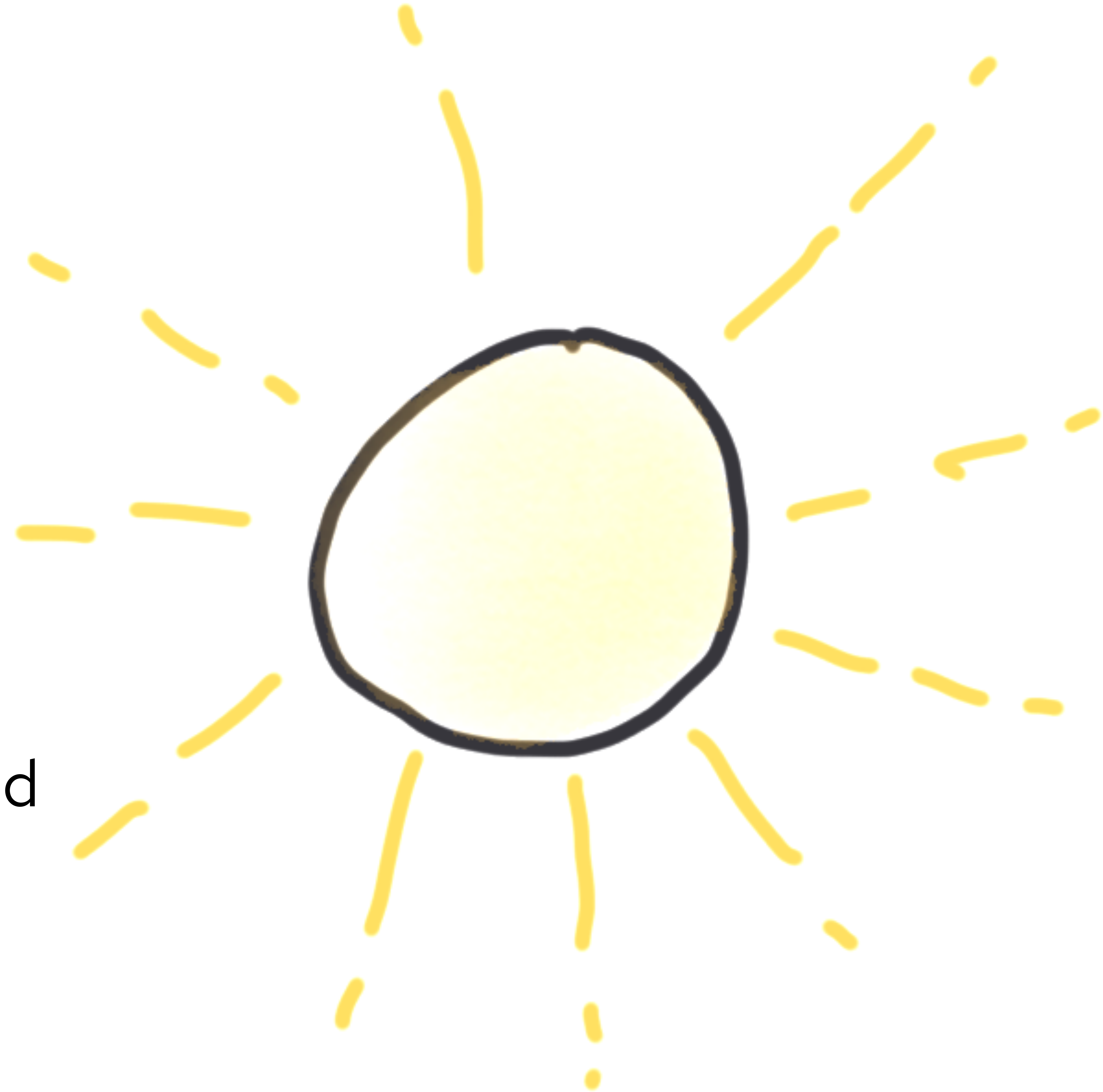
we need to talk about virginia

look at the sustainability information before choosing a hosting region look at the sustainability information before choosing a hosting region

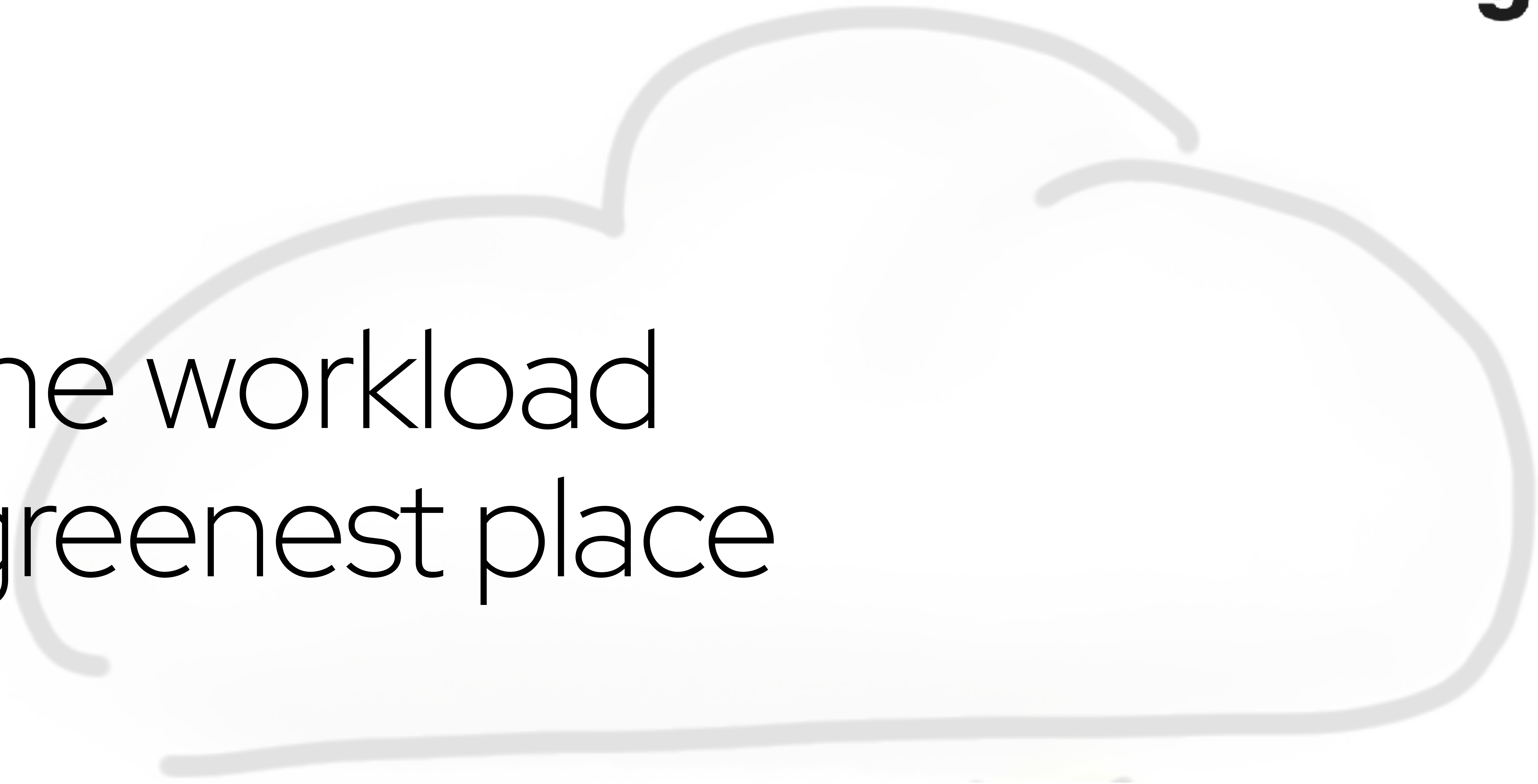
choose a cloud provider who make this easy

# time of day matters

- (most) renewables are  intermittent
- if grid load is high, shortfalls are filled by fossil fuels



# move the workload to the greenest place



# move the workload to the greenest place

# move the workload
# to the greenest place

a data problem

# move the workload
# to the greenest place

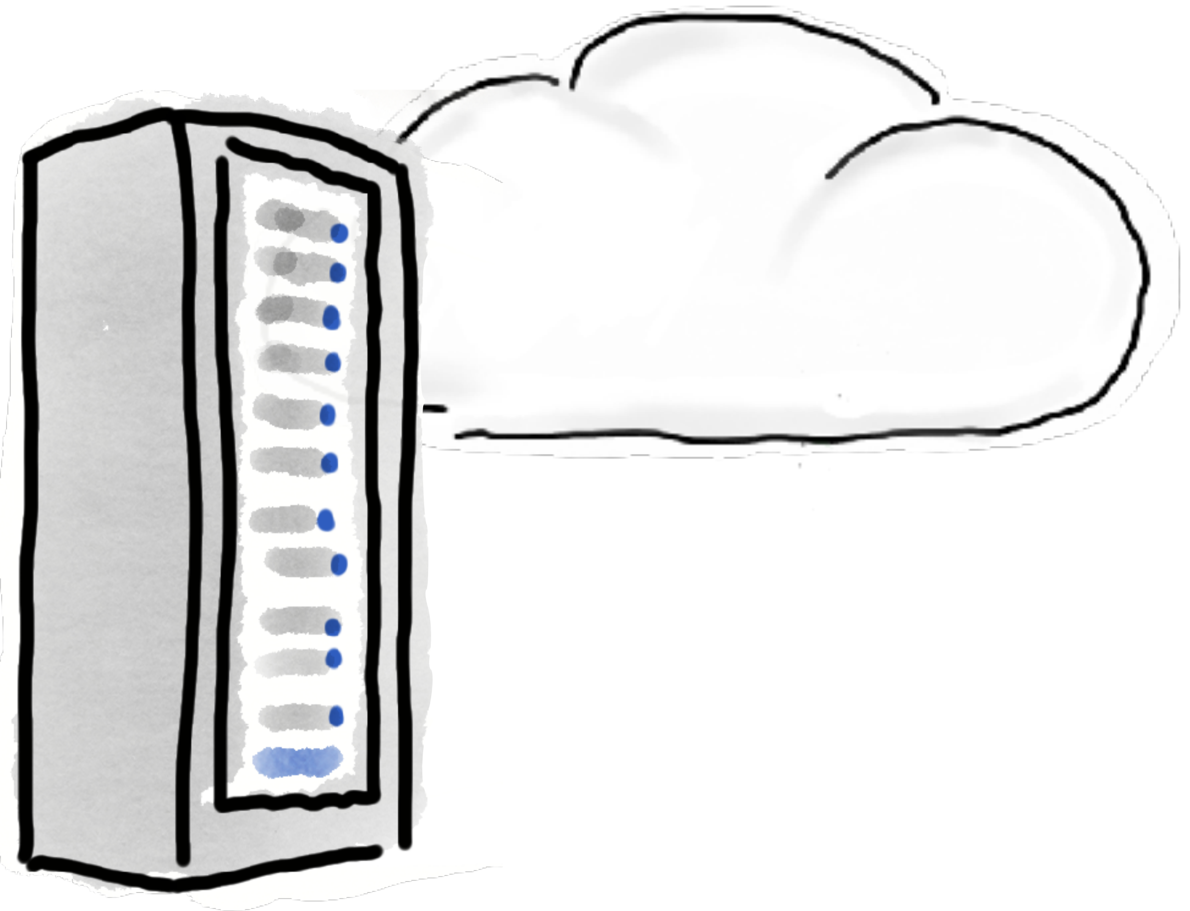
a data problem

an orchestration problem

# green software foundation: principles

carbon awareness

hardware efficiency

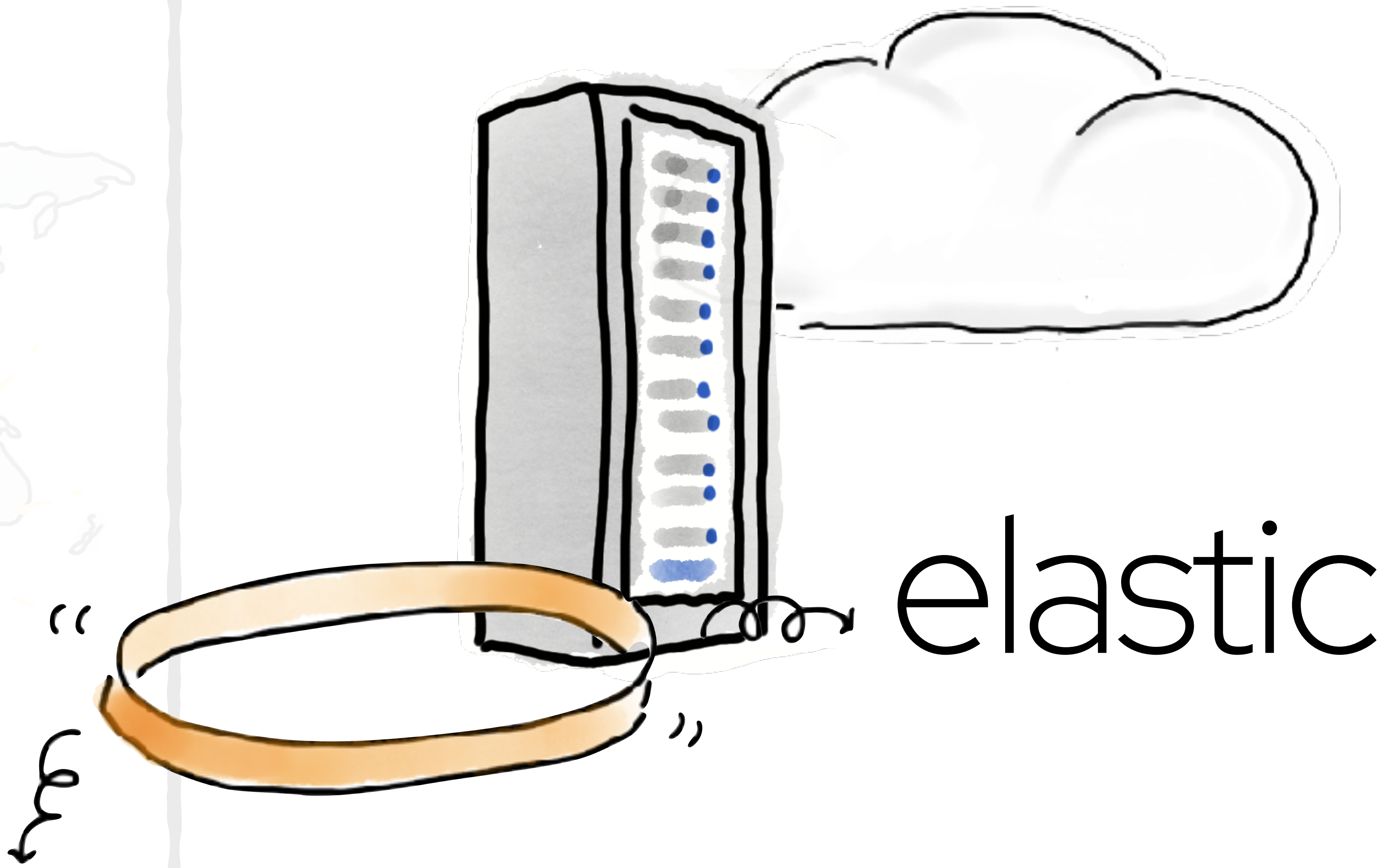


electricity efficiency

# green software foundation: principles

carbon awareness

hardware efficiency



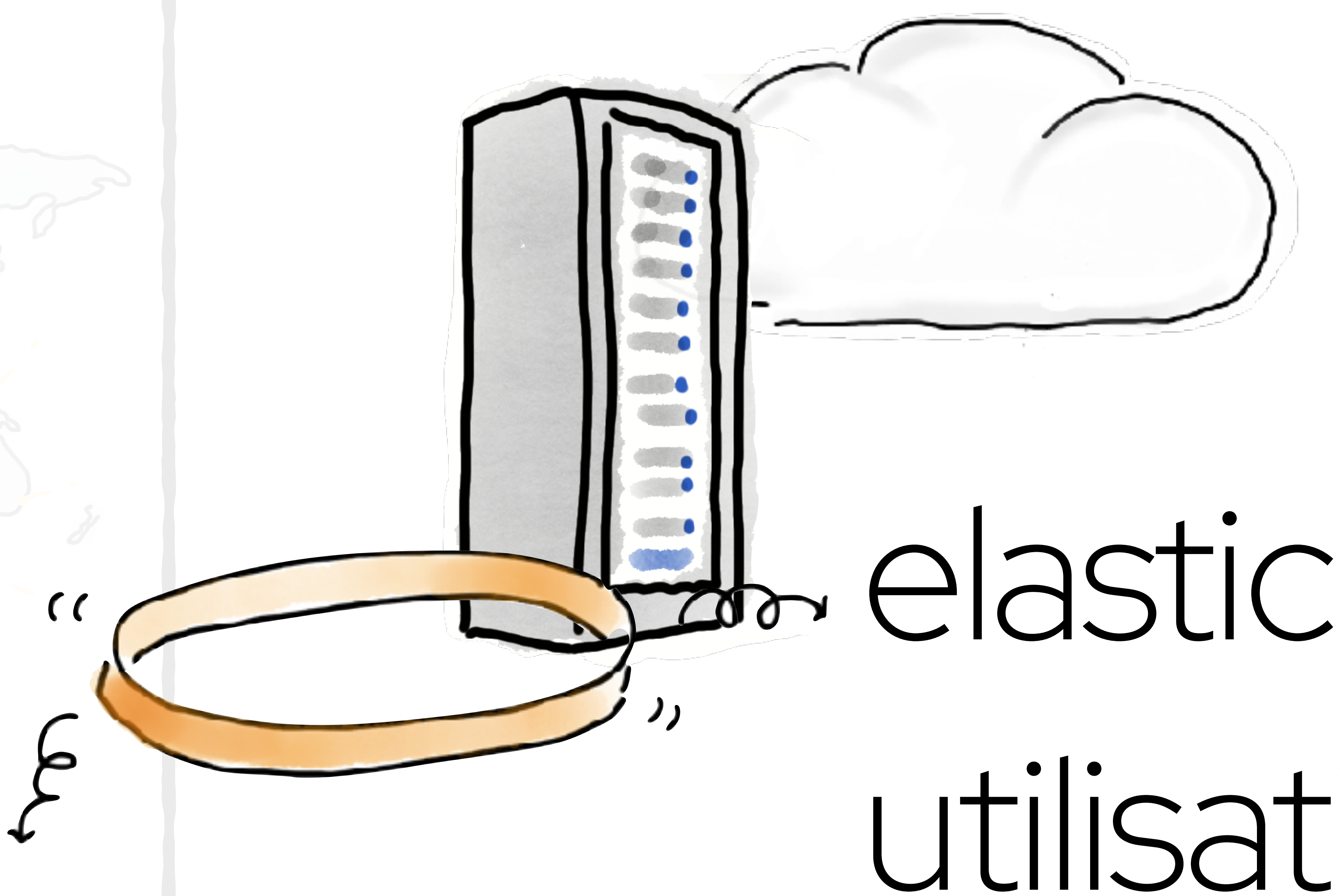
elasticity

electricity efficiency

# green software foundation: principles

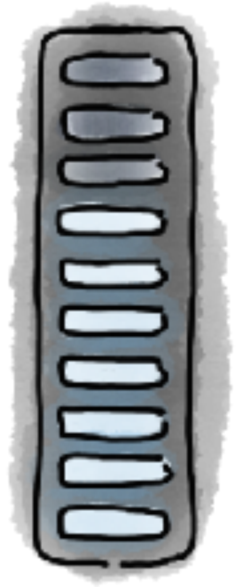
carbon awareness

hardware efficiency

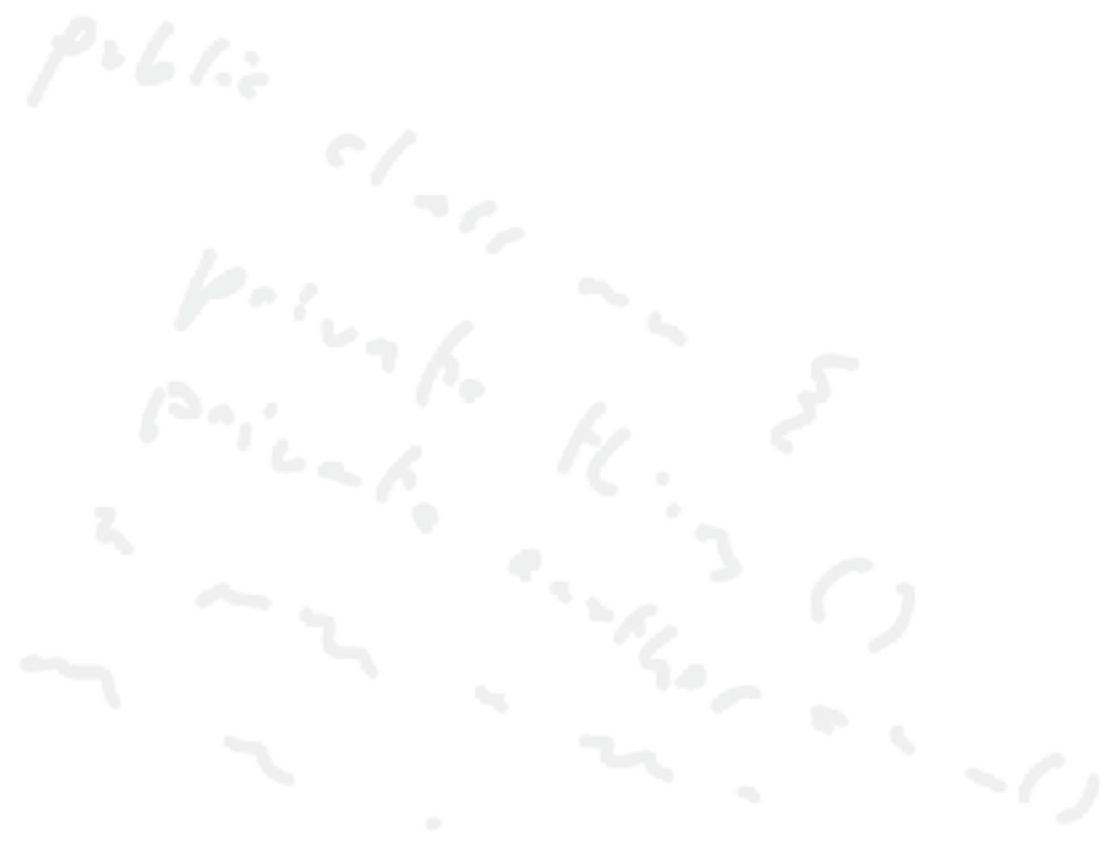


elasticity

utilisation



electricity efficiency

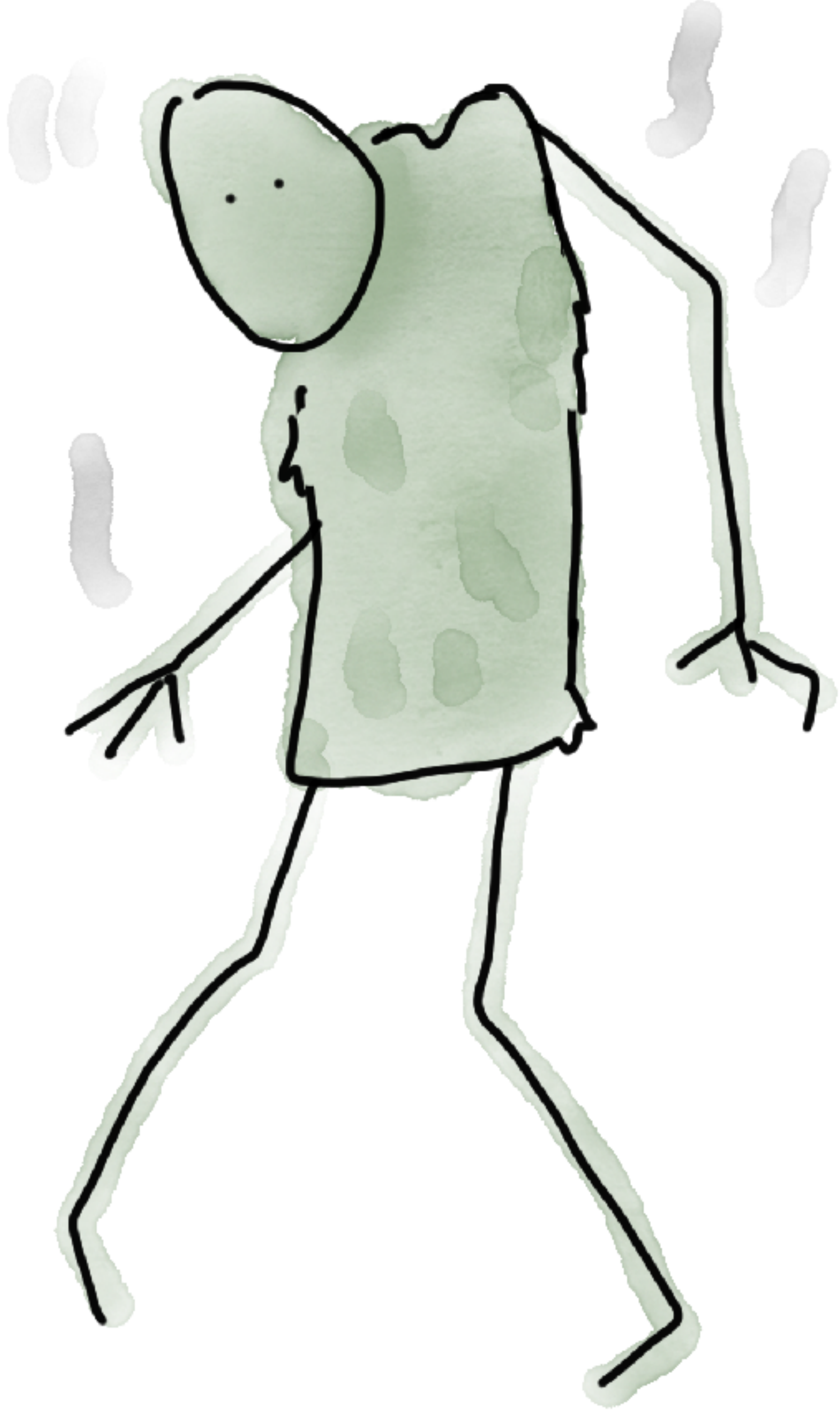


2017 server-survey

# 25%

doing **no** useful work

(16,000 sampled)

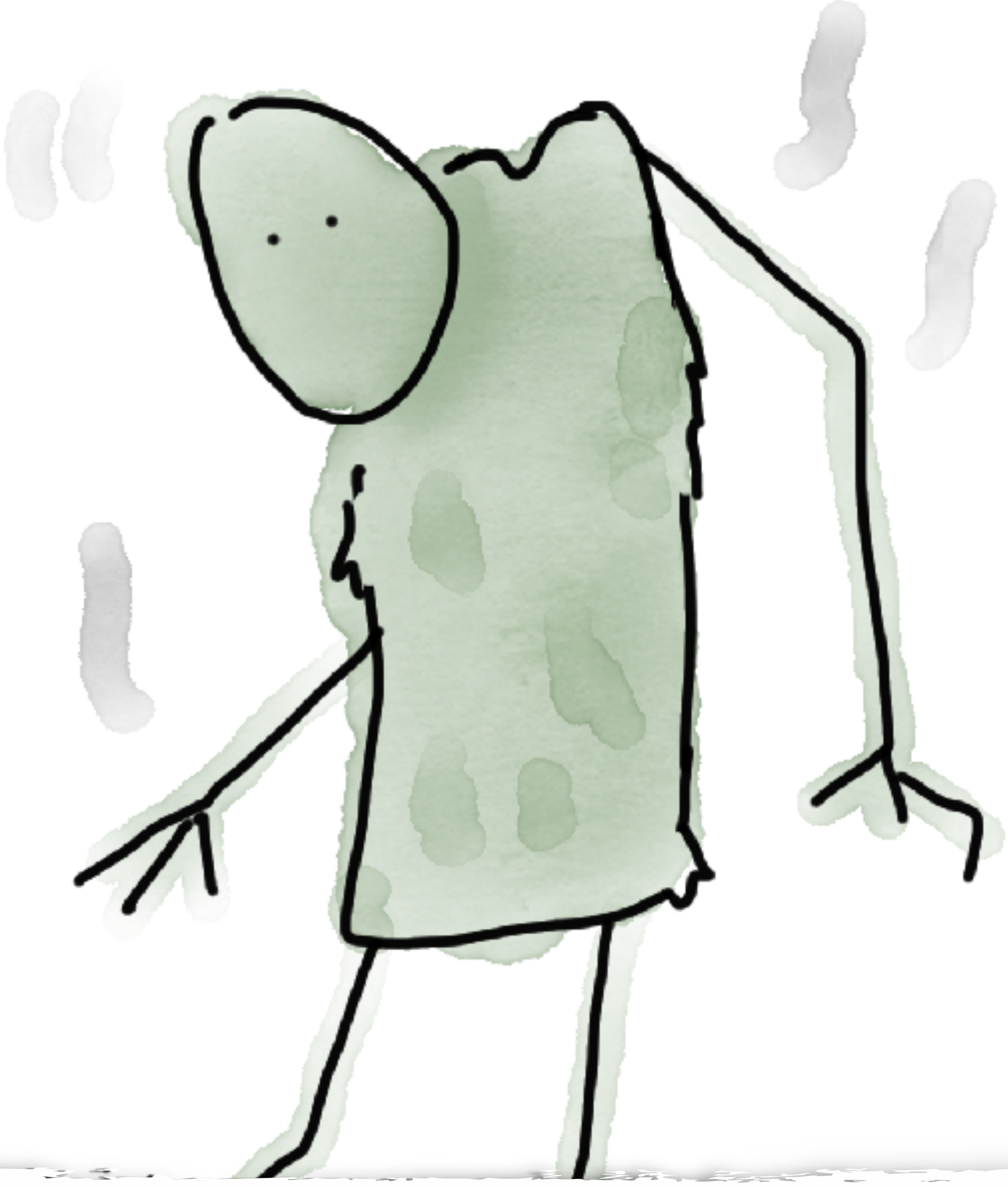


## 2017 server-survey

# 25%

doing **no** useful work

(16,000 sampled)



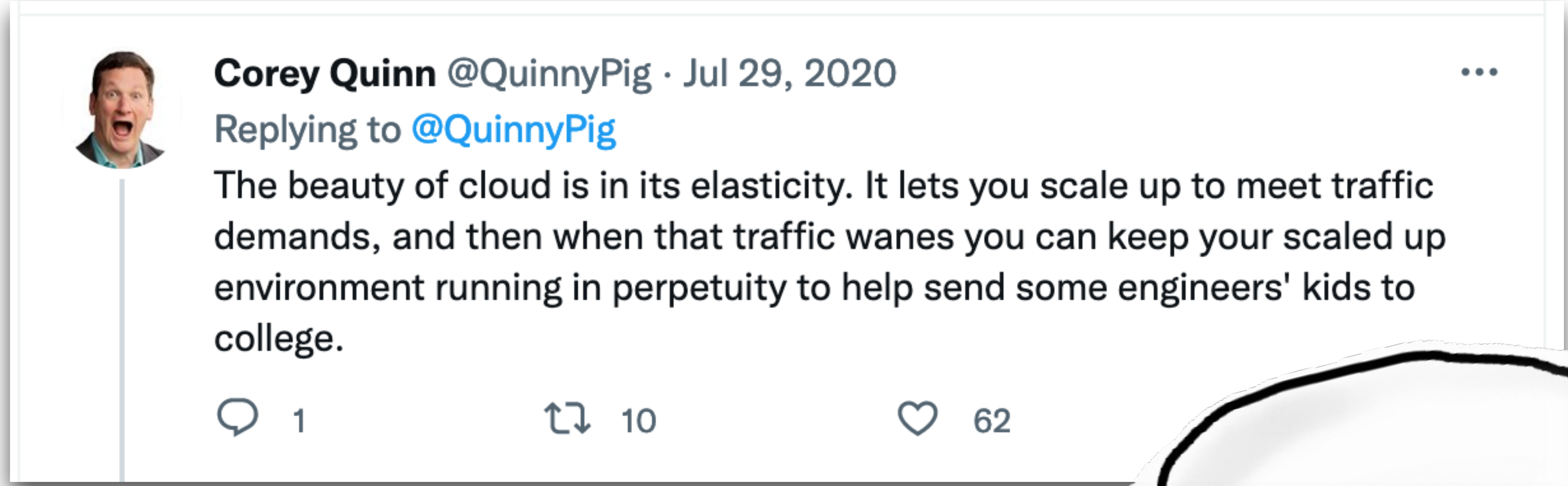
“perhaps someone forgot to turn them off” https://www.anthesisgroup.com/wp-content/uploads/2019/11/Comatose-Servers-Redux-2017.pdf

# 2014 server-survey

29%

active less than 5% of the time

(4,000 sampled)



**Corey Quinn** @QuinnyPig · Jul 29, 2020

Replying to @QuinnyPig

The beauty of cloud is in its elasticity. It lets you scale up to meet traffic demands, and then when that traffic wanes you can keep your scaled up environment running in perpetuity to help send some engineers' kids to college.

# cloud elasticity?

https://www.business2community.com/cloud-computing/overprovisioning-always-on-resources-lead-to-26-6-billion-in-public-cloud-waste-expected-in-2021-02381033

# cloud elasticity?

2021:

https://www.business2community.com/cloud-computing/overprovisioning-always-on-resources-lead-to-26-6-billion-in-public-cloud-waste-expected-in-2021-02381033

# cloud elasticity?

2021:

**$26.6 billion** wasted

https://www.business2community.com/cloud-computing/overprovisioning-always-on-resources-lead-to-26-6-billion-in-public-cloud-waste-expected-in-2021-02381033

# cloud elasticity?

2021:

**$26.6 billion** wasted

by always-on cloud instances

https://www.business2community.com/cloud-computing/overprovisioning-always-on-resources-lead-to-26-6-billion-in-public-cloud-waste-expected-in-2021-02381033

# elasticity

we used to leave our applications running all the time

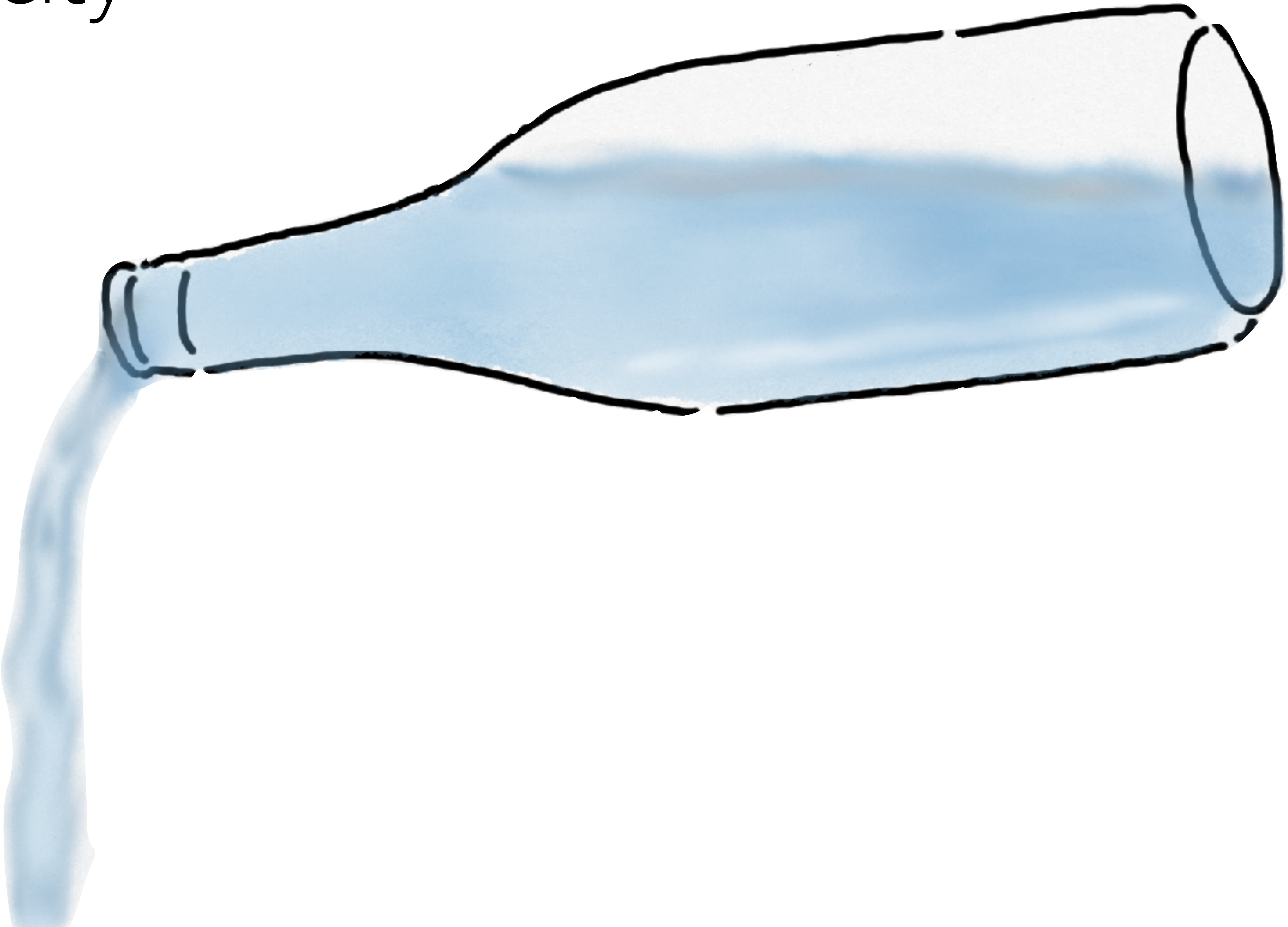
when we scripted turning them off at night, we reduced our cloud bill by 30%

@darkandnerdy, Chicago DevOpsDays

# it’s not just electricity

it’s not just electricity

it’s water



it's not just electricity

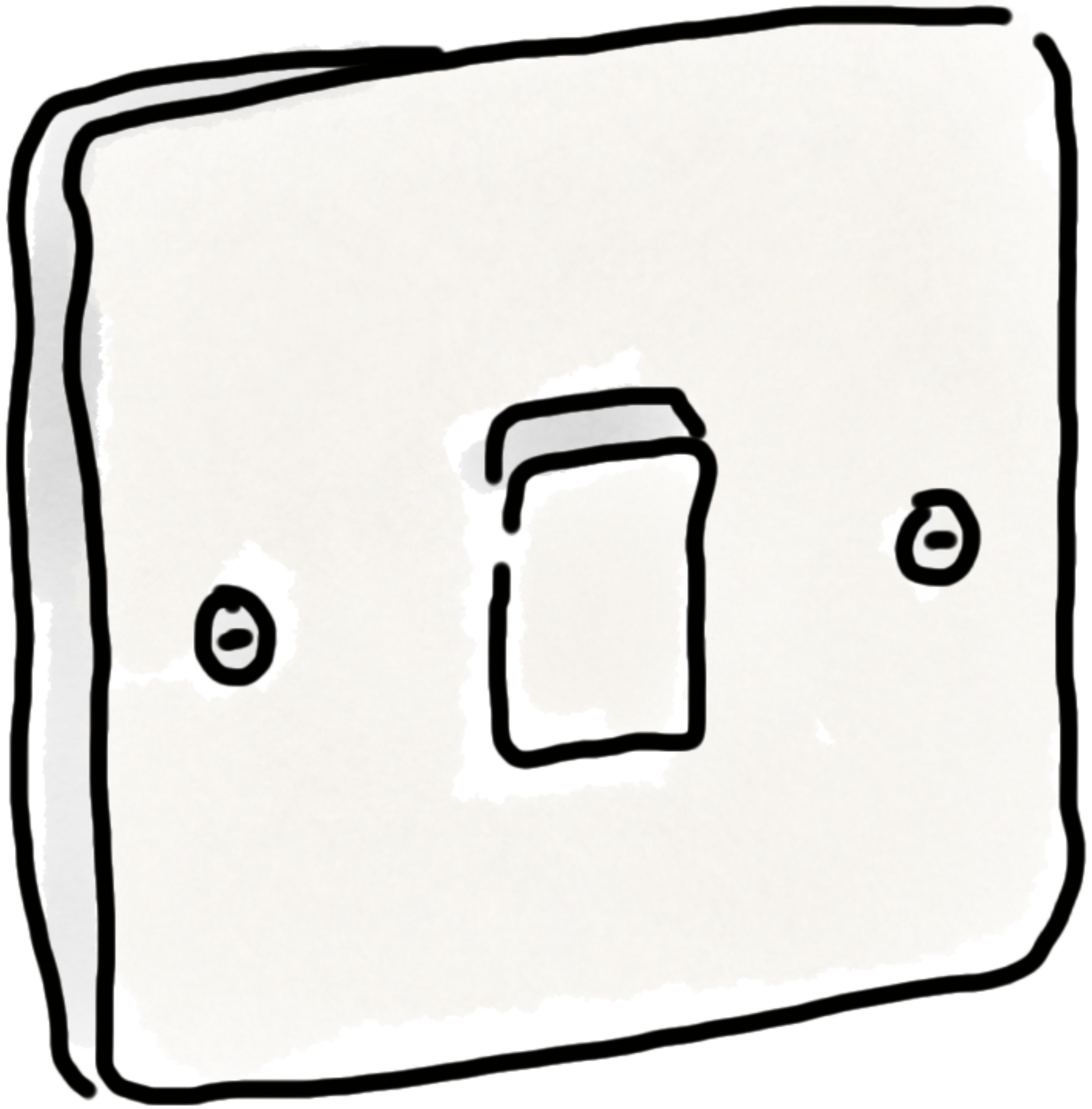
it's water

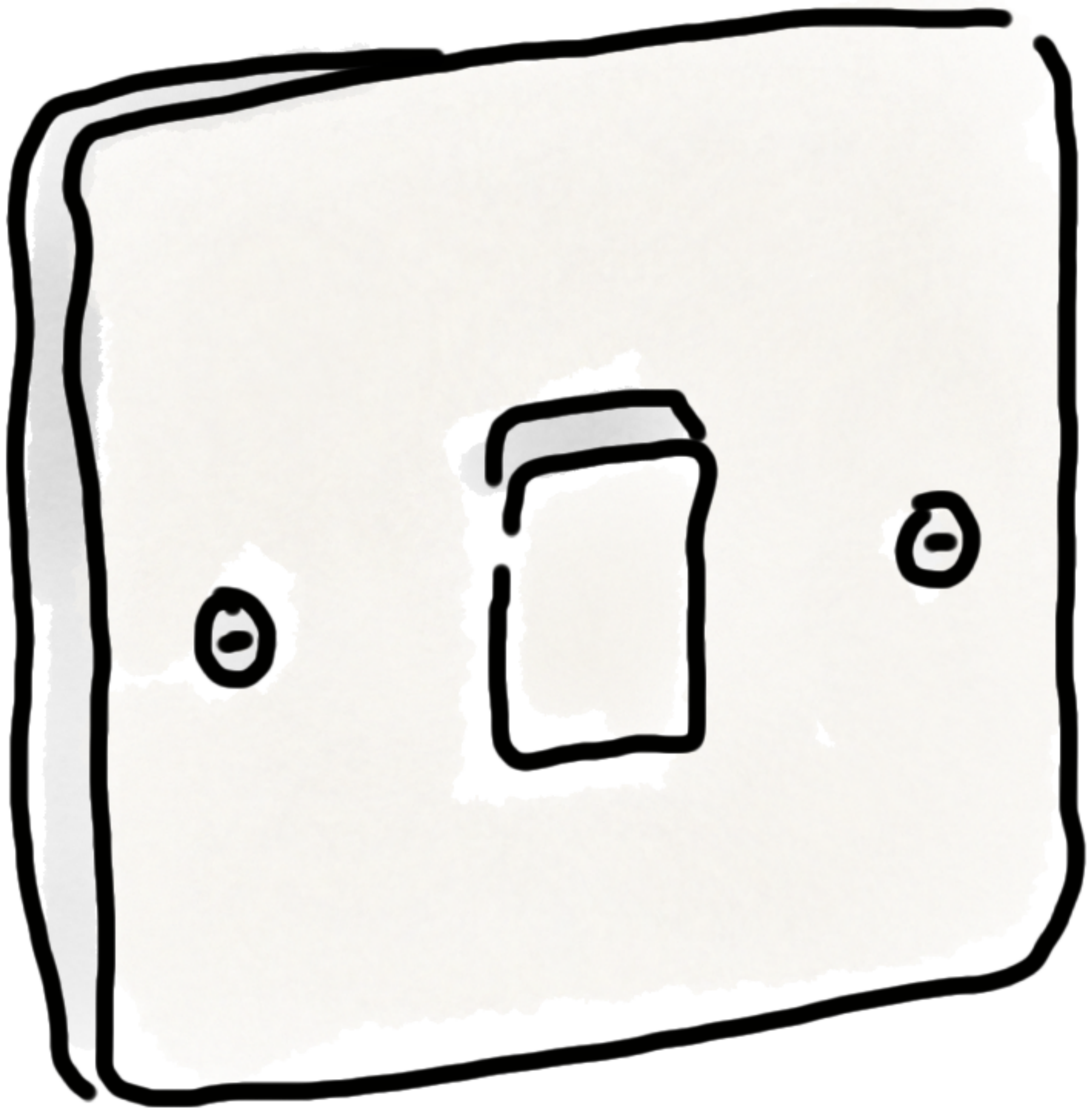
it's e-waste

@holly_cummins

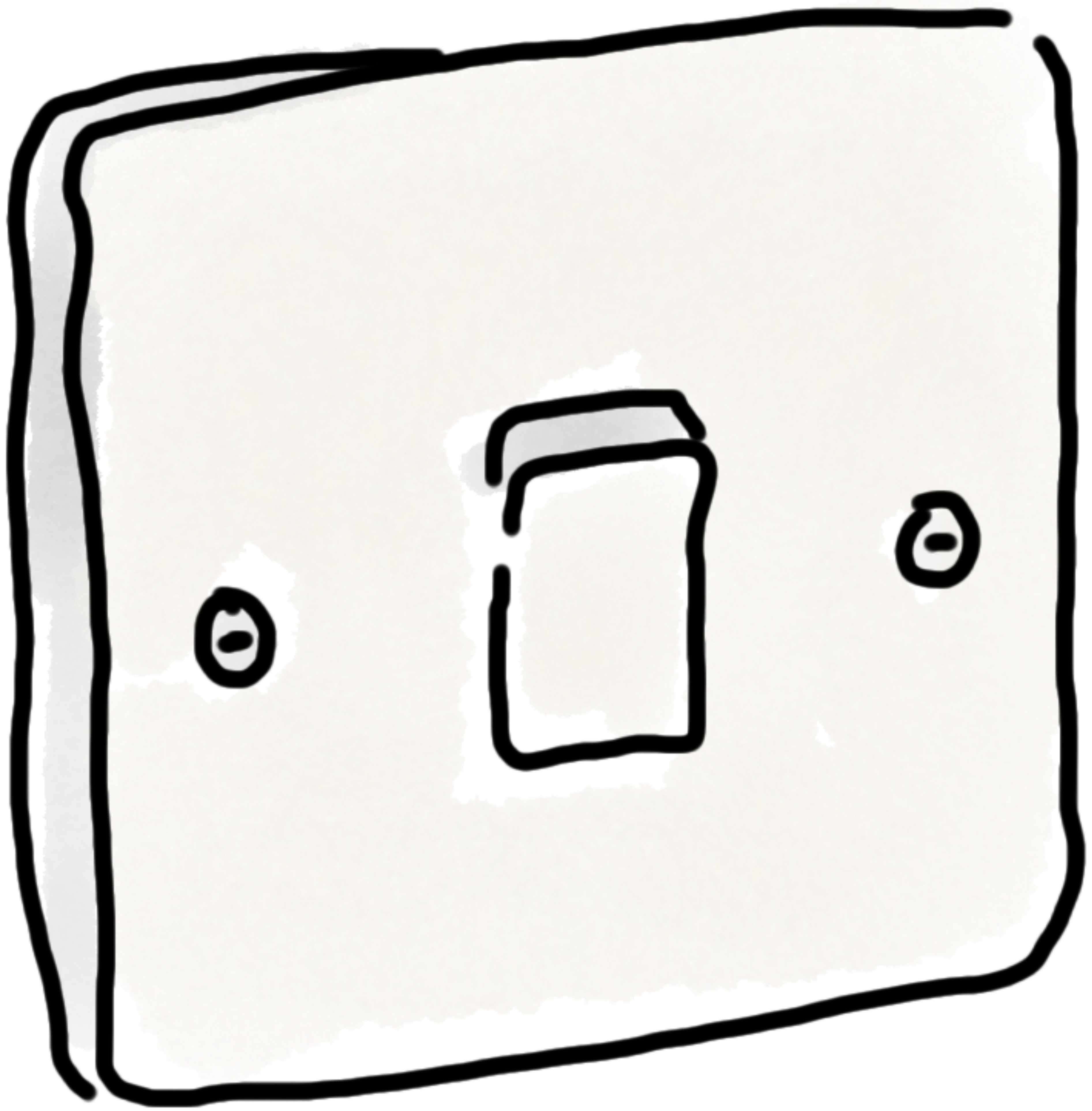
yes, turning applications off is scary

what if … turning applications off was no more scary than turning the lights off?



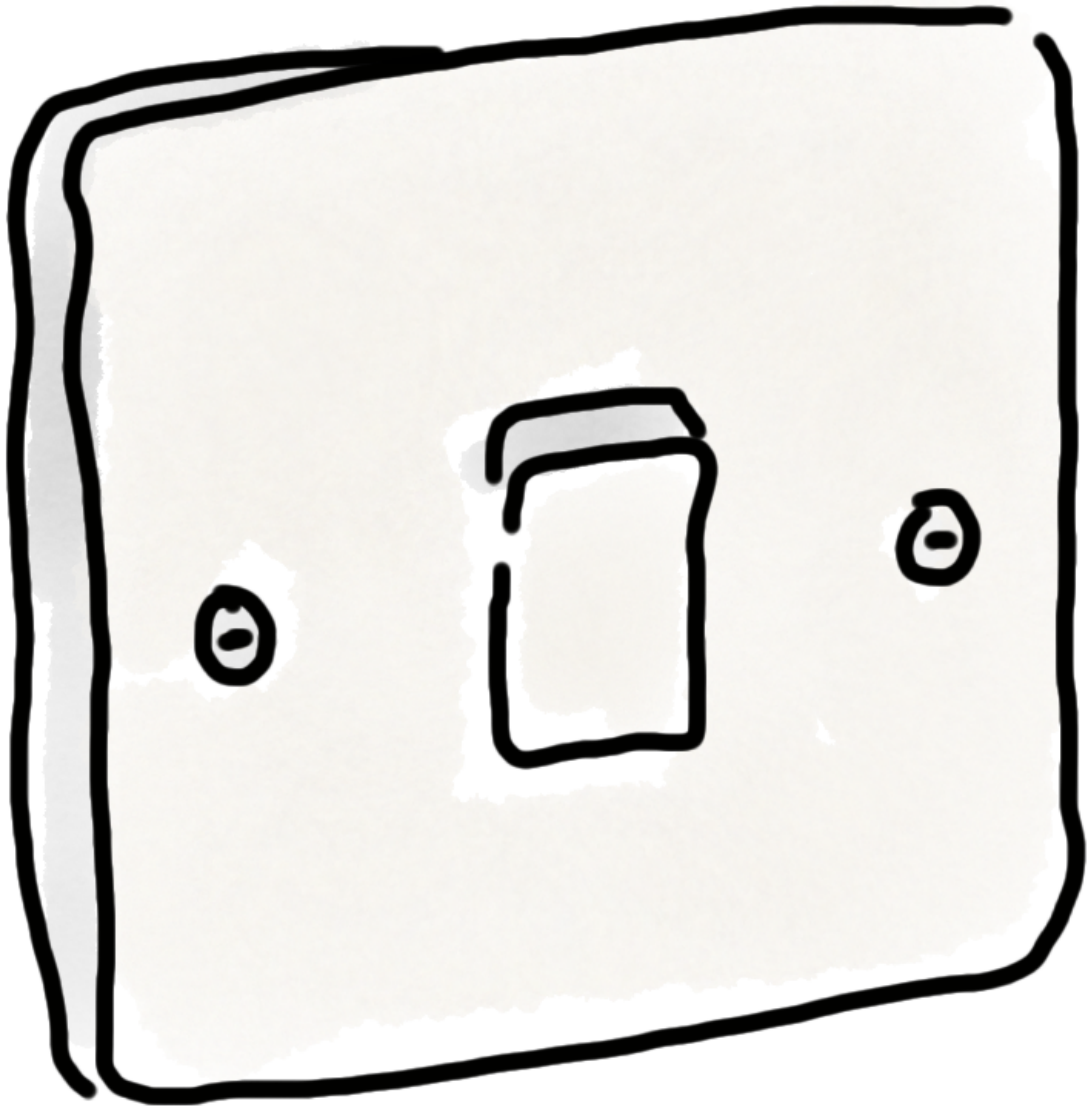


ultimate elasticity



ultimate elasticity

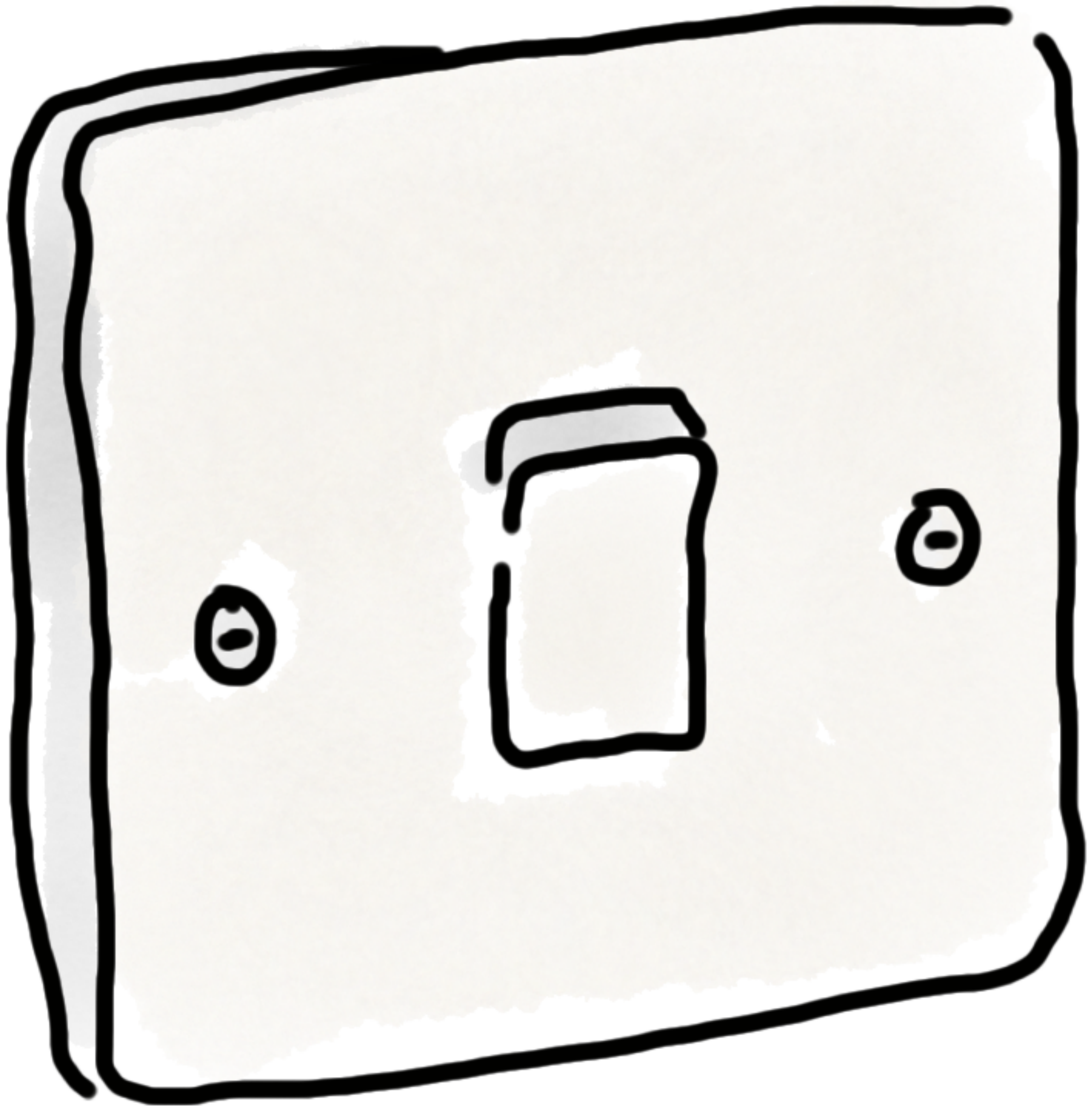
turning it off and on again must



ultimate elasticity

turning it off and on again must

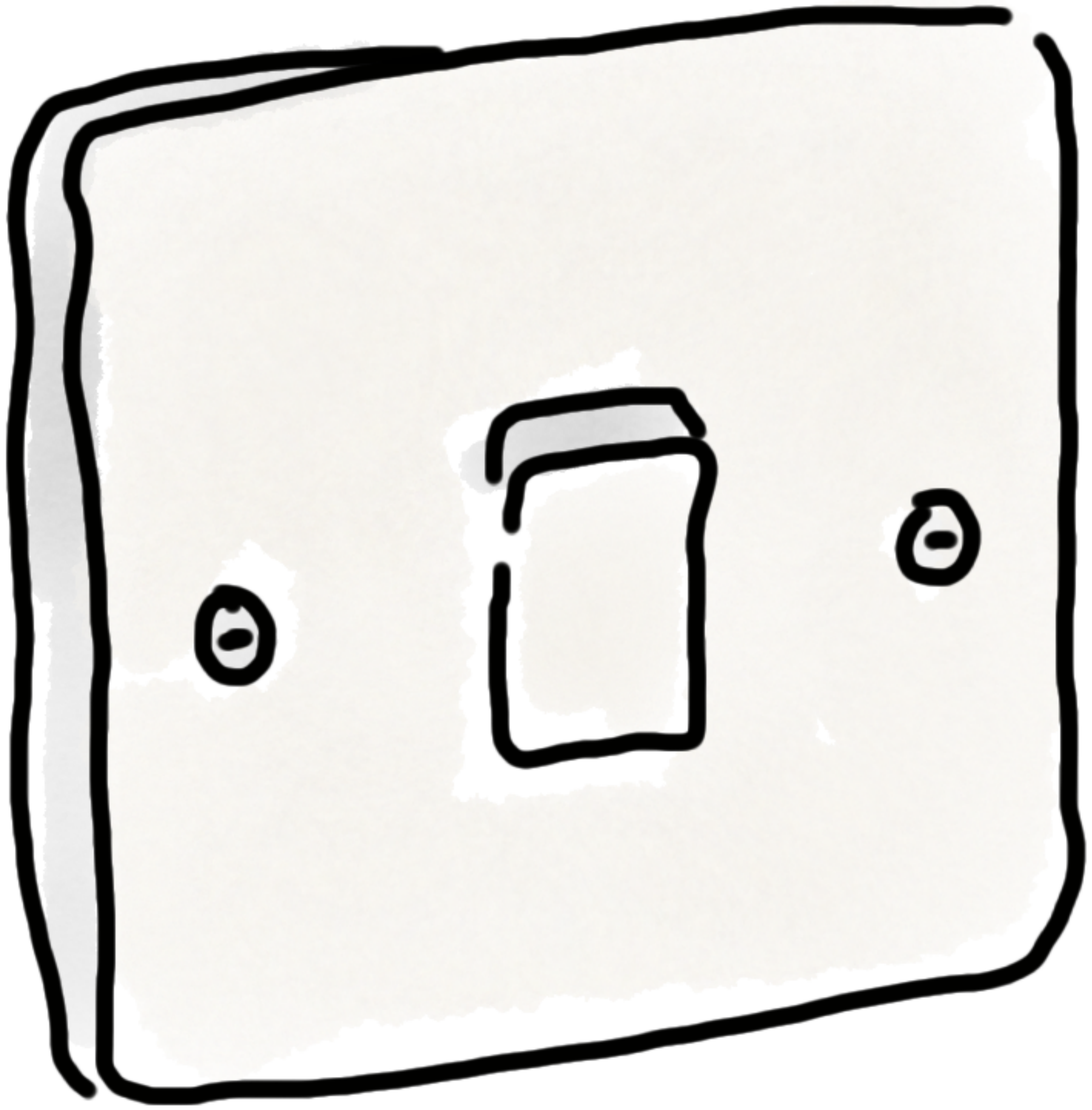
- be **fast**



ultimate elasticity

turning it off and on again must

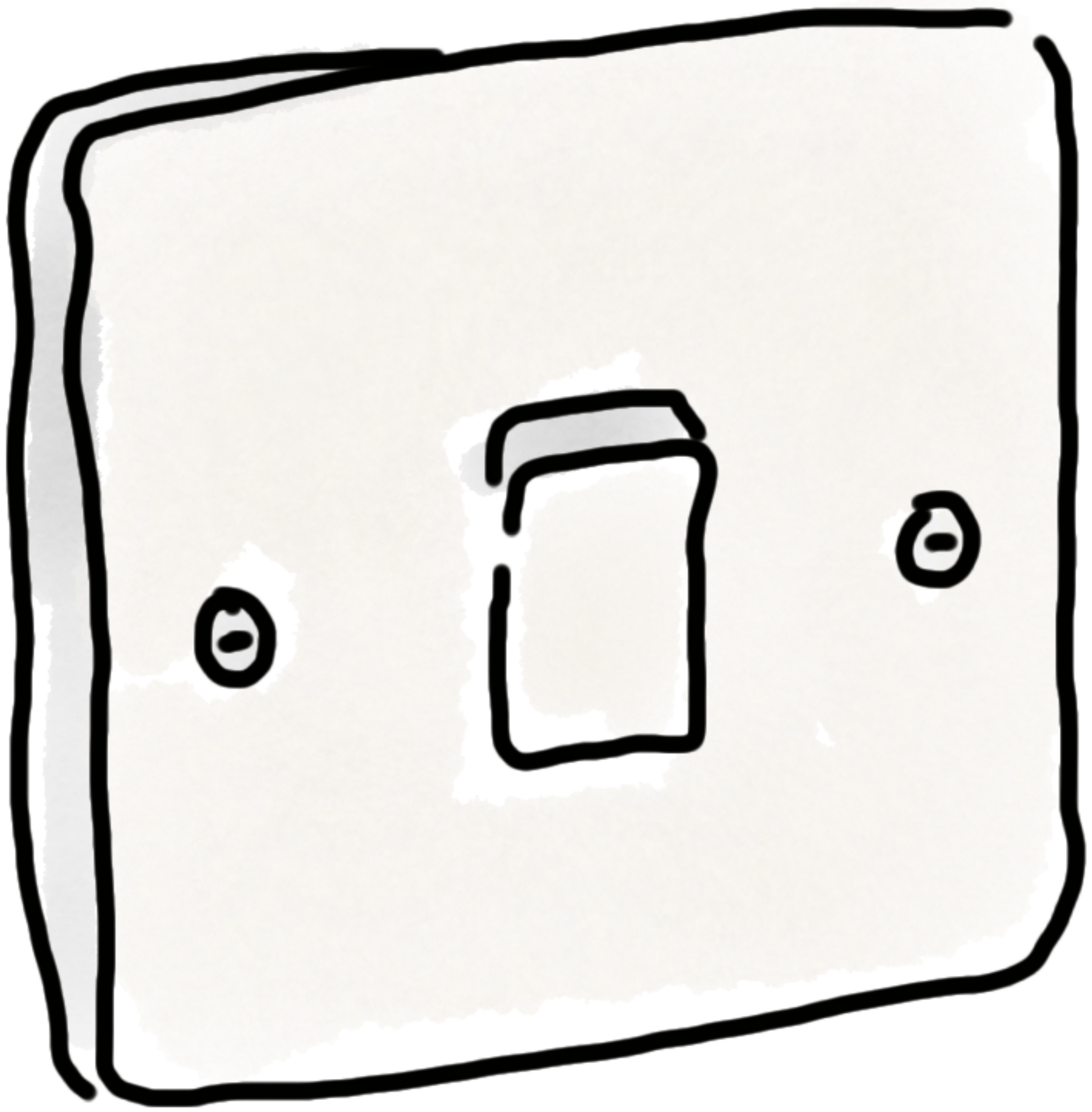
- be **fast**
- actually **work**



ultimate elasticity

turning it off and on again must

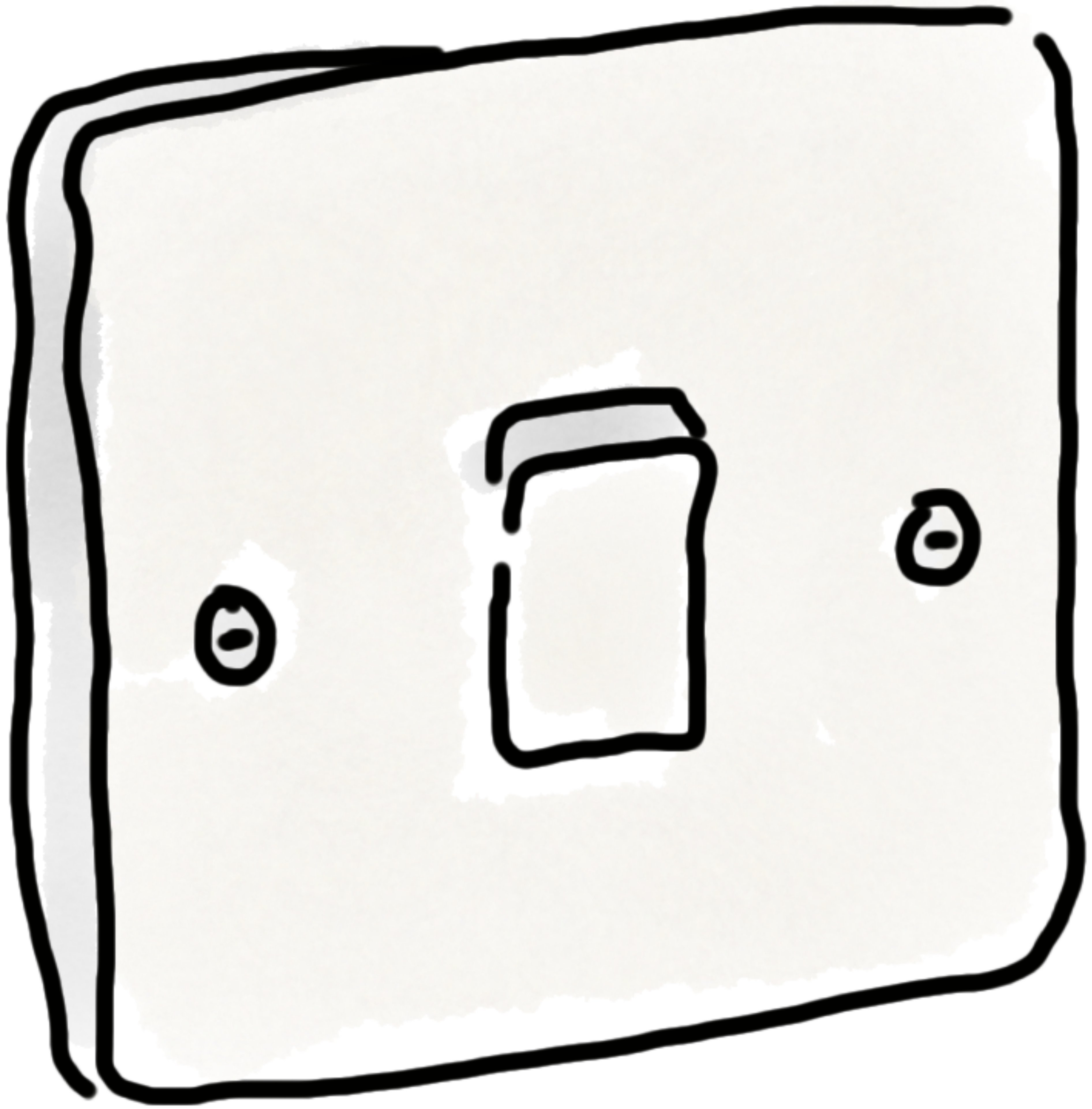
- be **fast**
- actually **work**
  - idempotency



ultimate elasticity

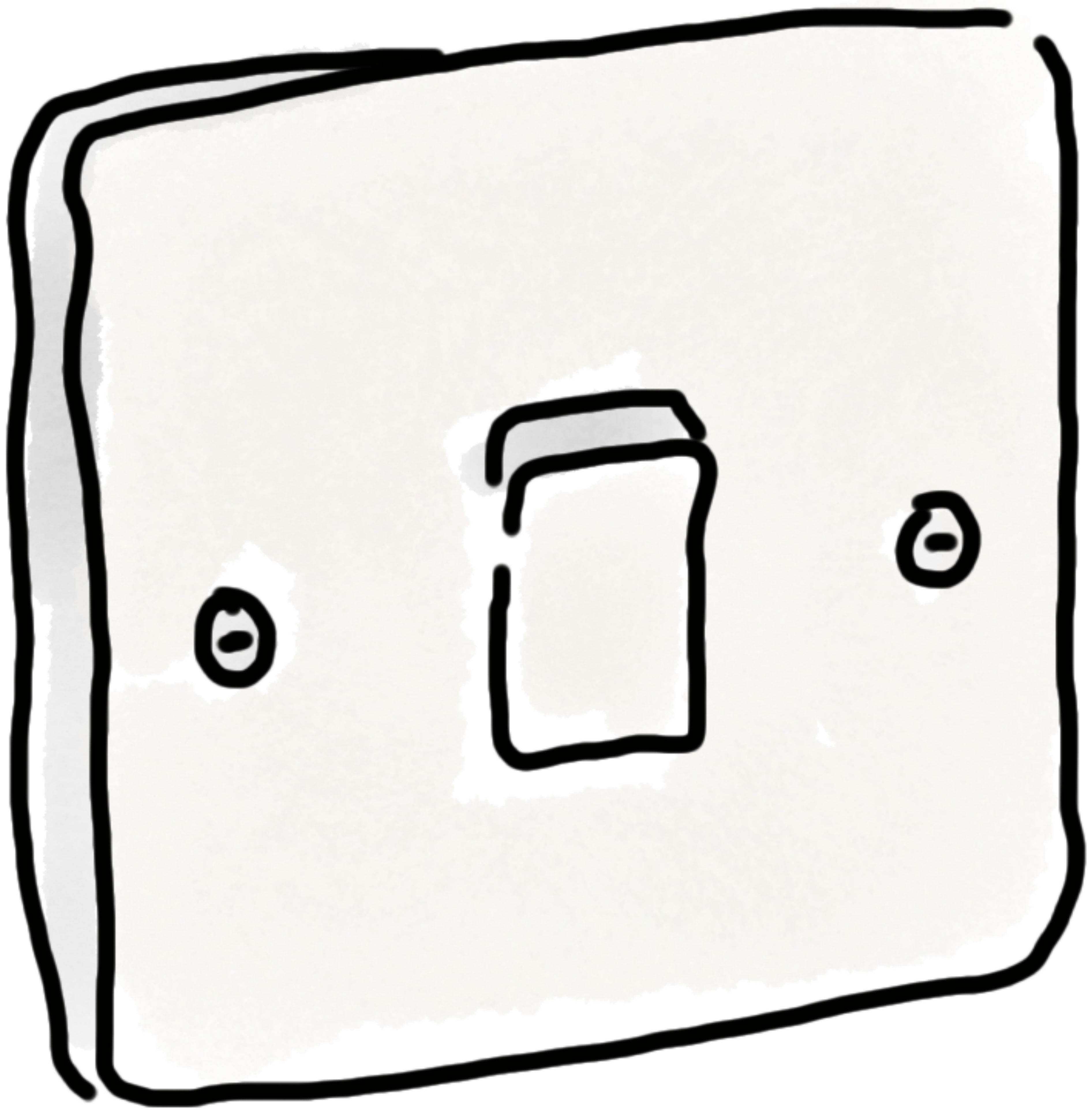
turning it off and on again must

- be **fast**
- actually **work**
  - idempotency
  - resiliency



ultimate elasticity

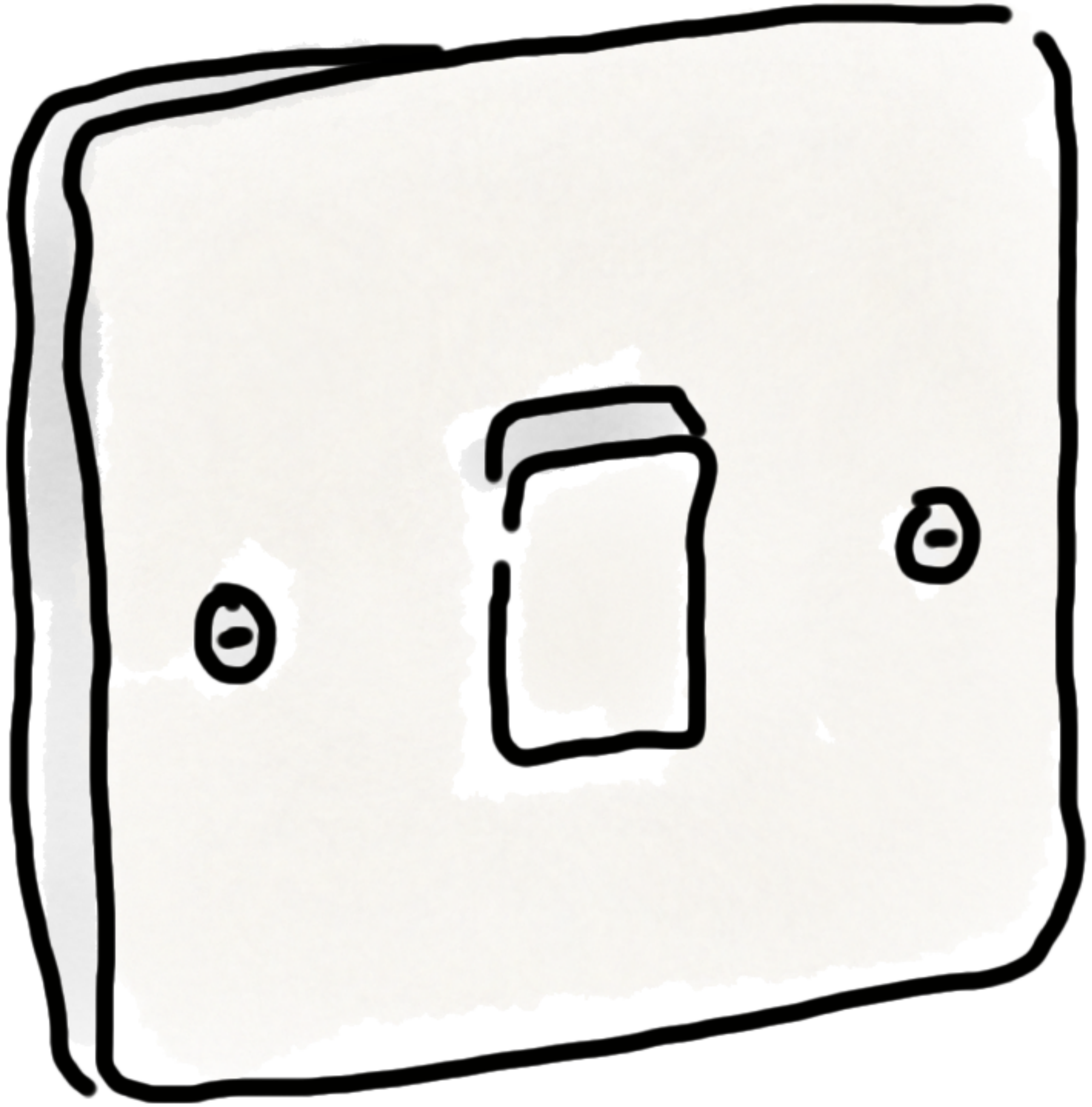
architect things to be turned off and on often



# trick 2:

# LightSwitchOps

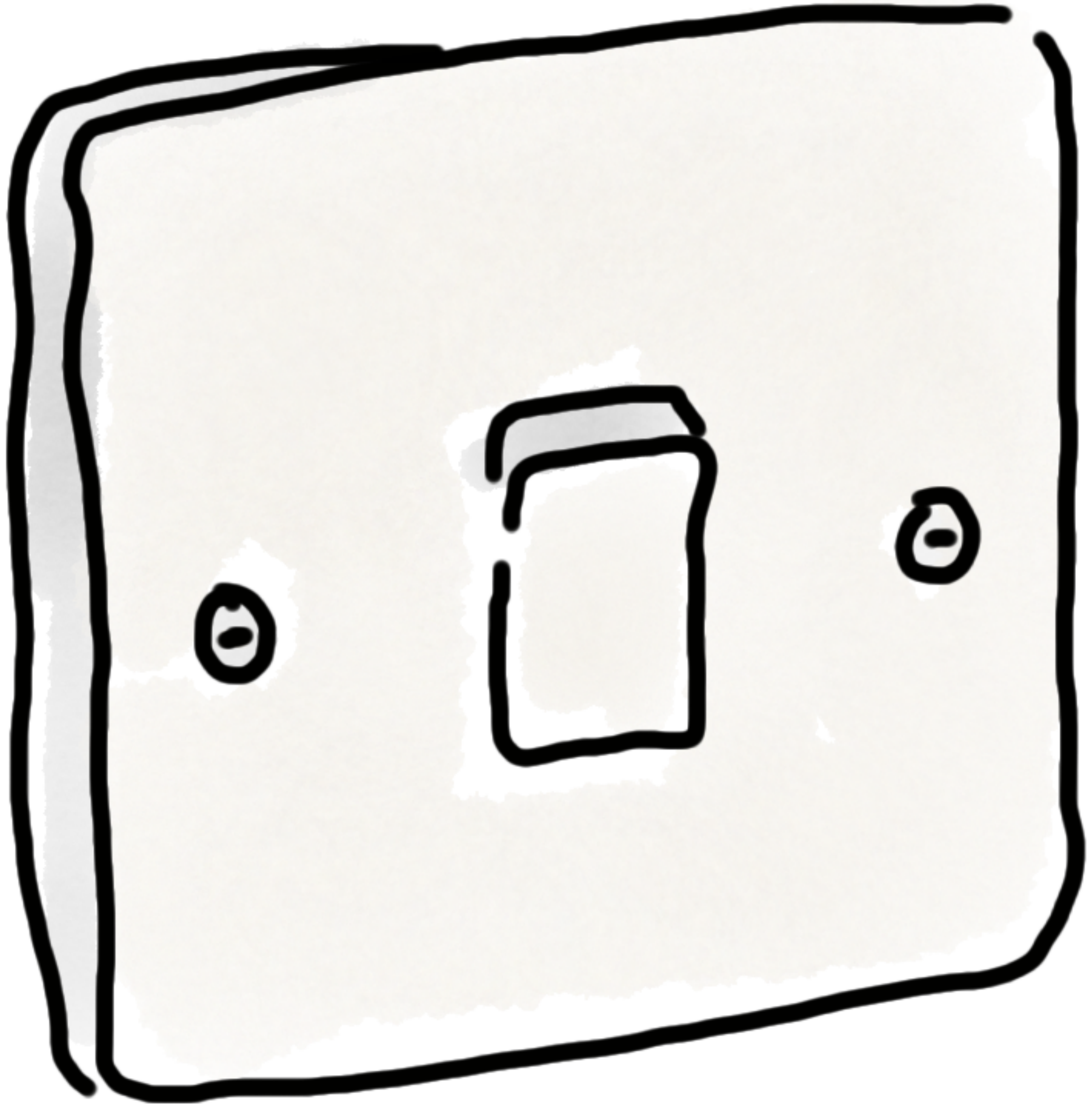
architect things to be turned off and on often

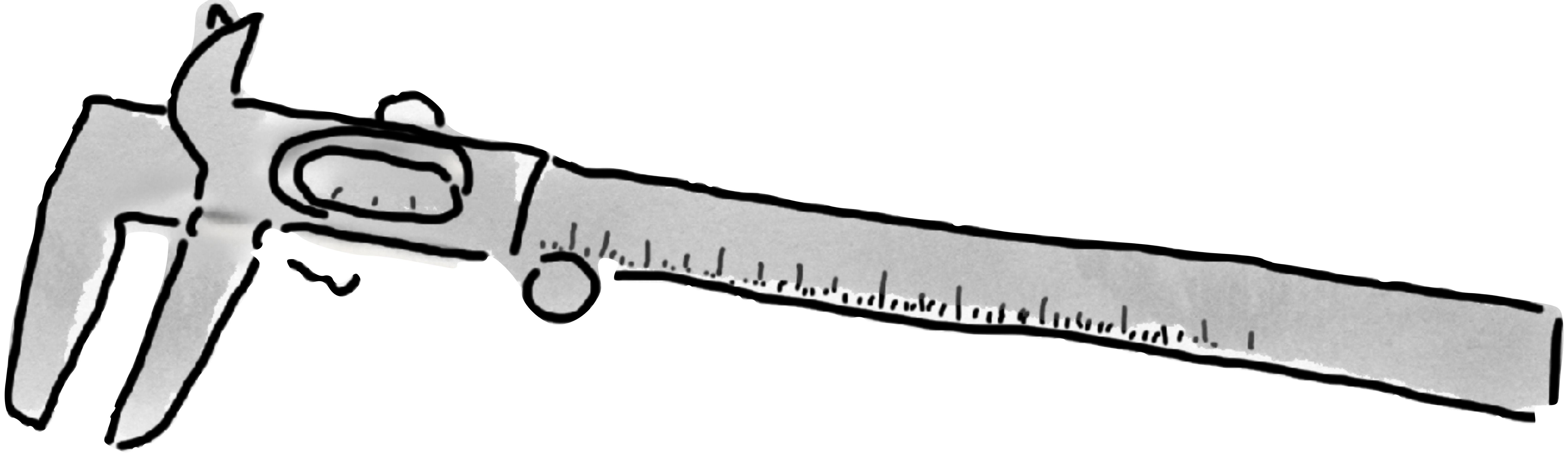


# trick 2:

# LightSwitchOps

architect things to be turned off and on often

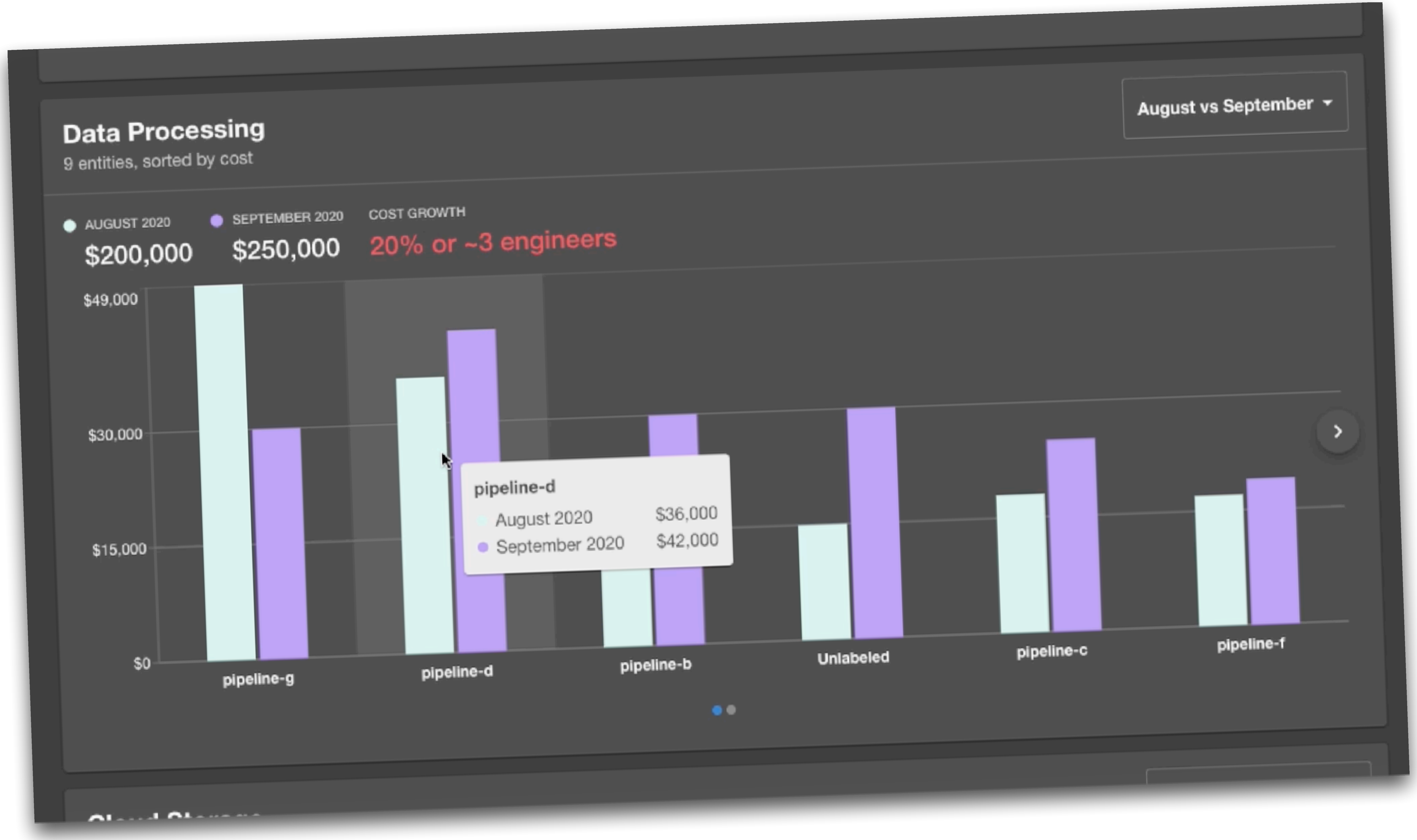




you can’t optimise what you can’t measure

# FinOps

figuring out who in your company forgot to turn off their cloud



## Data Processing

9 entities, sorted by cost

August vs September ▾

● AUGUST 2020 $200,000

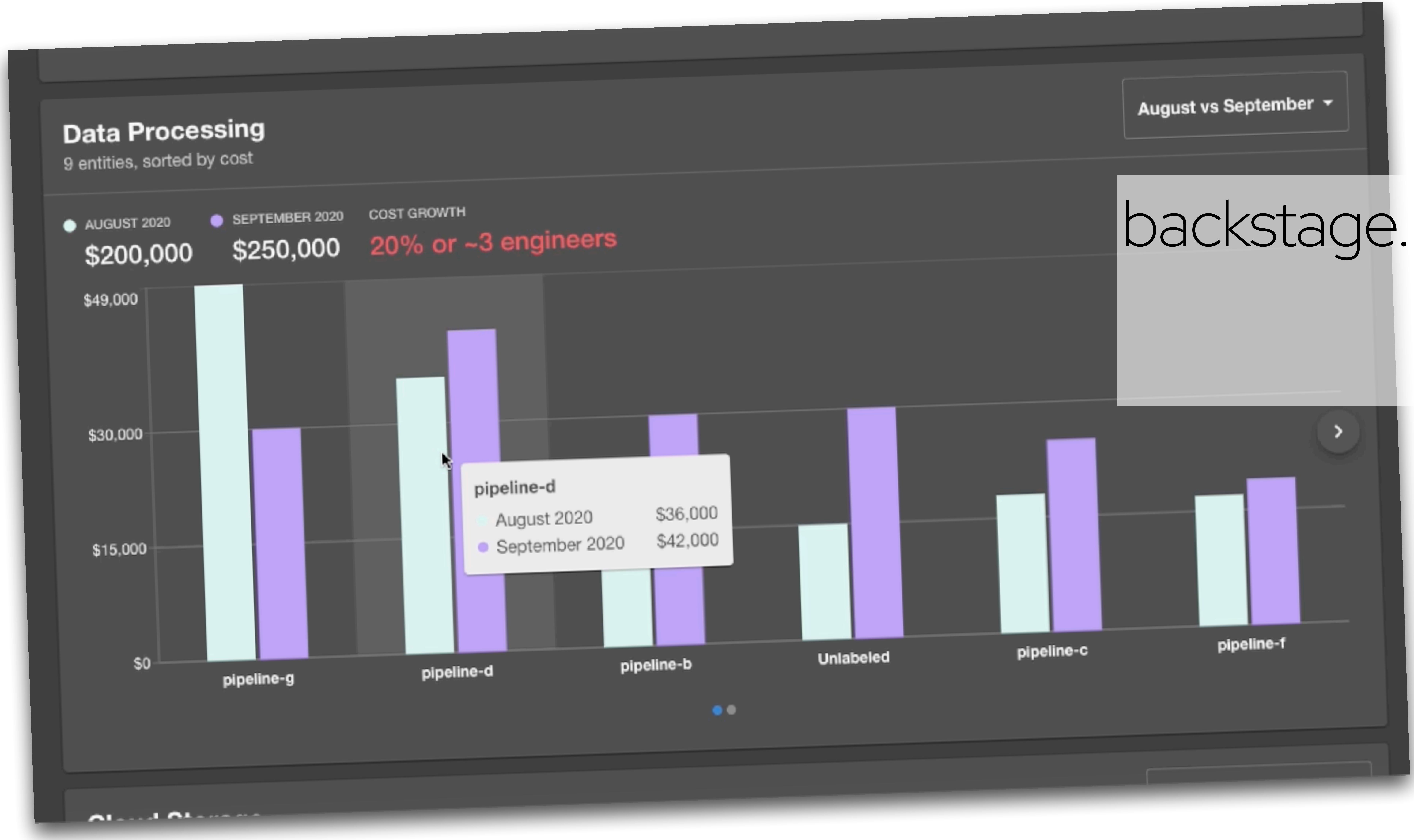
● SEPTEMBER 2020 $250,000

COST GROWTH 20% or ~3 engineers

pipeline-d

● August 2020 $36,000

● September 2020 $42,000



## Data Processing

9 entities, sorted by cost

● AUGUST 2020 $200,000

● SEPTEMBER 2020 $250,000

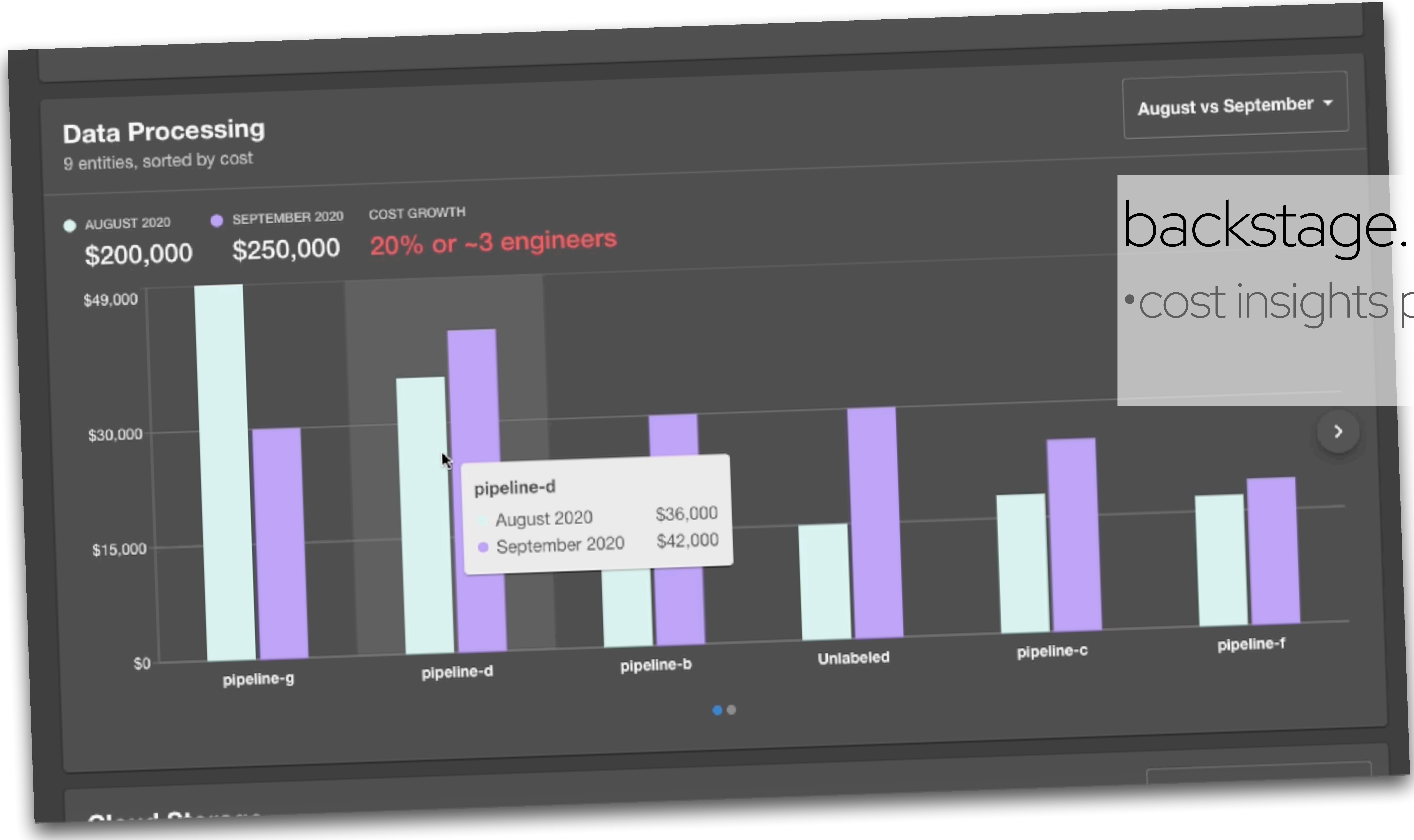
COST GROWTH 20% or ~3 engineers

pipeline-d

● August 2020 $36,000

● September 2020 $42,000

backstage.io



## Data Processing

9 entities, sorted by cost

August vs September

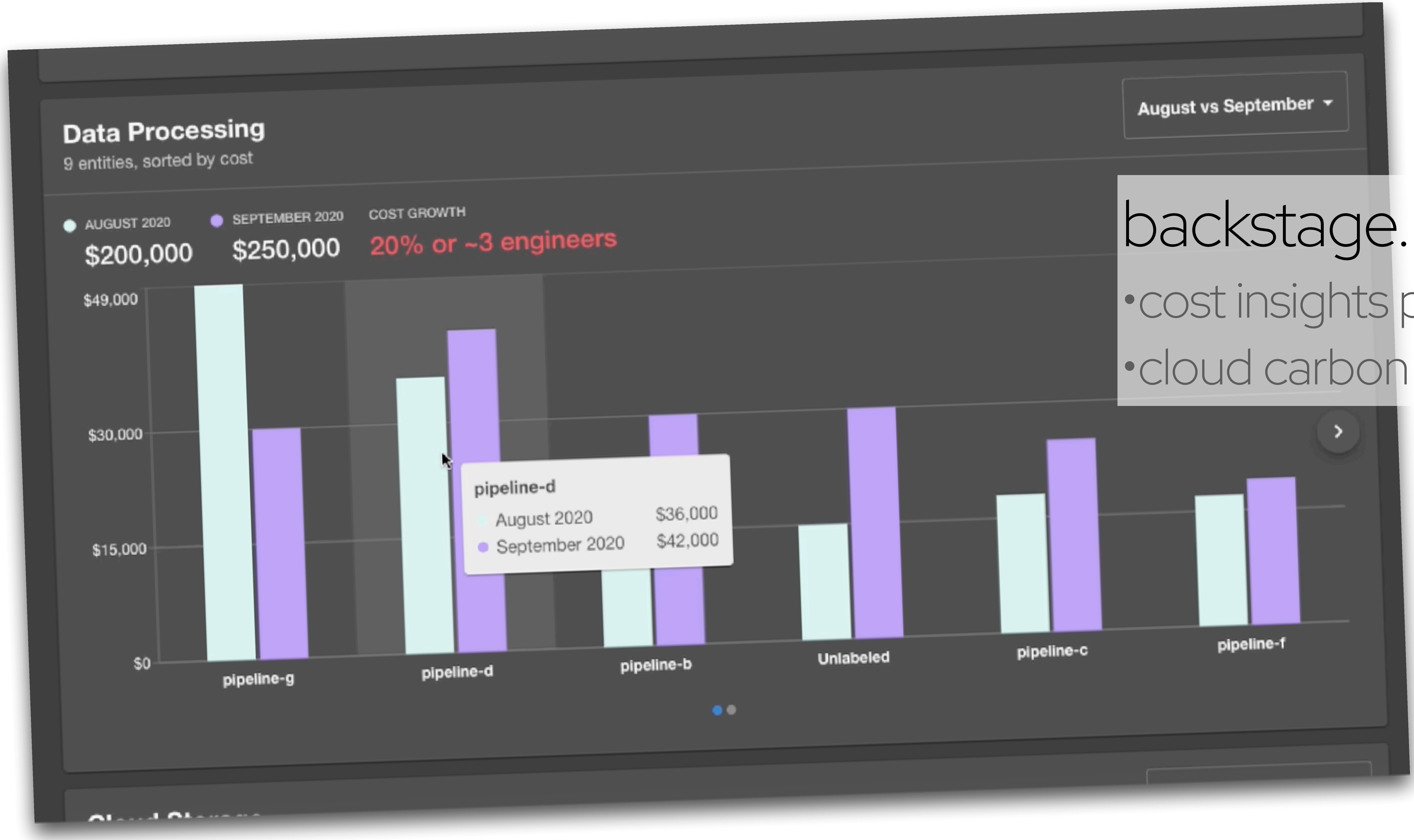
| ● AUGUST 2020 | ● SEPTEMBER 2020 | COST GROWTH |
|---|---|---|
| $200,000 | $250,000 | 20% or ~3 engineers |

pipeline-d

| | |
|---|---|
| ● August 2020 | $36,000 |
| ● September 2020 | $42,000 |

# backstage.io

- cost insights plugin



Data Processing

9 entities, sorted by cost

August vs September

● AUGUST 2020 $200,000

● SEPTEMBER 2020 $250,000

COST GROWTH 20% or ~3 engineers

pipeline-d

● August 2020 $36,000

● September 2020 $42,000

# backstage.io

- cost insights plugin
- cloud carbon footprint plugin

# green software foundation: principles

carbon awareness

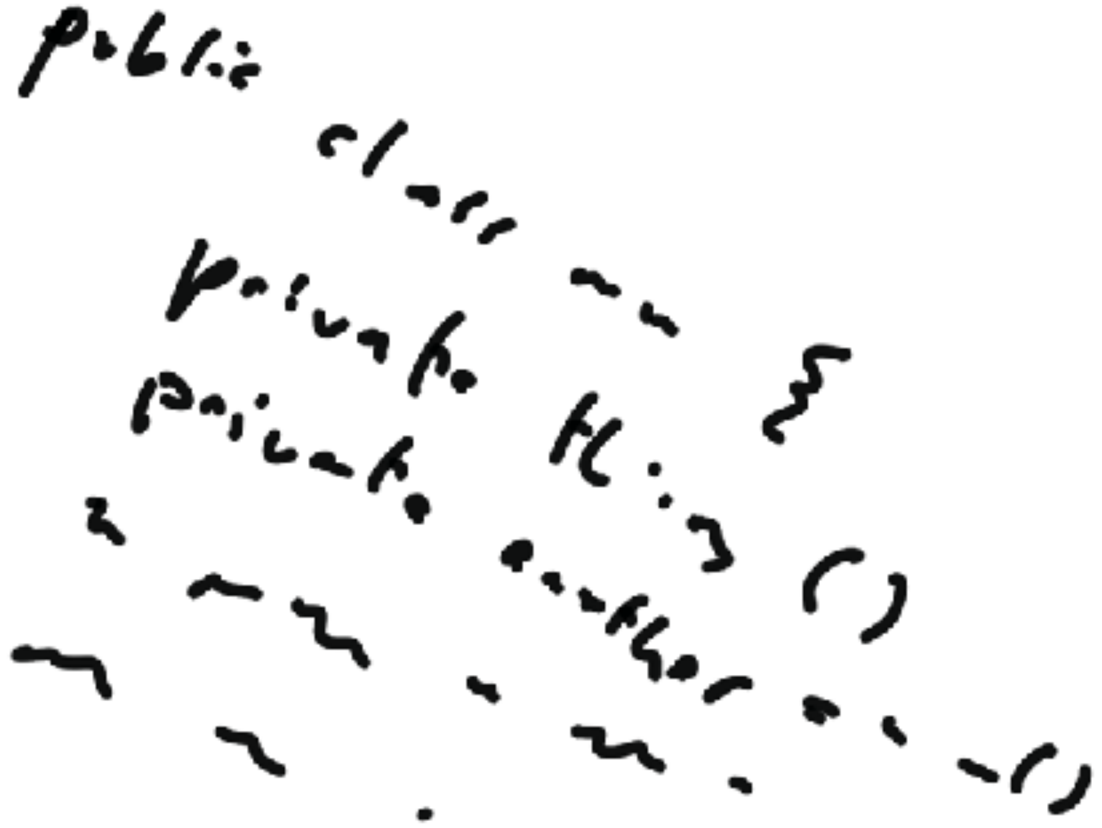
where
when

hardware efficiency

elasticity

utilisation

electricity efficiency

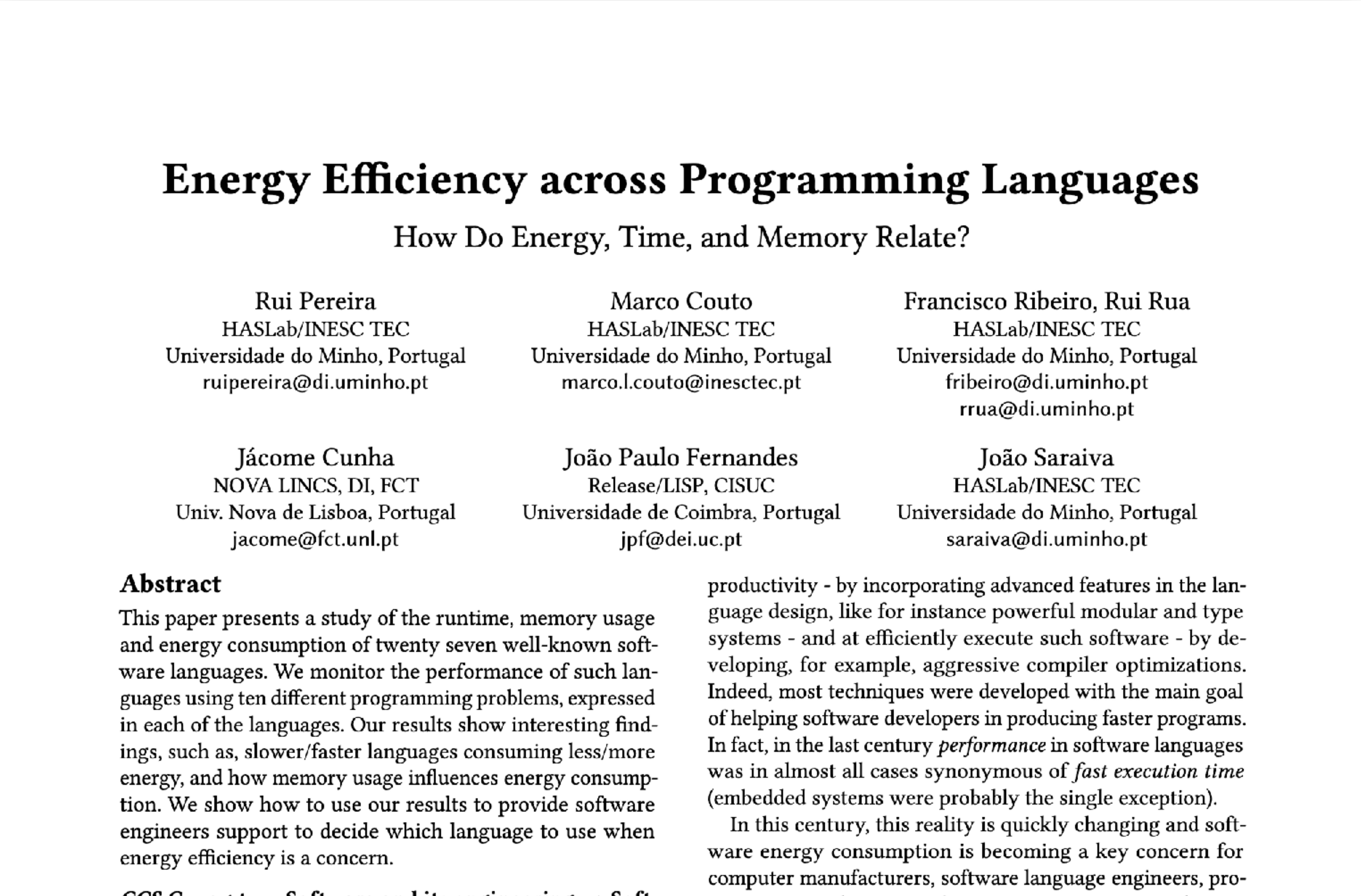


algorithms
stack

efficiency

what programming languages
use the **least** energy?

what programming languages use the **most** energy?



# 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

### Abstract

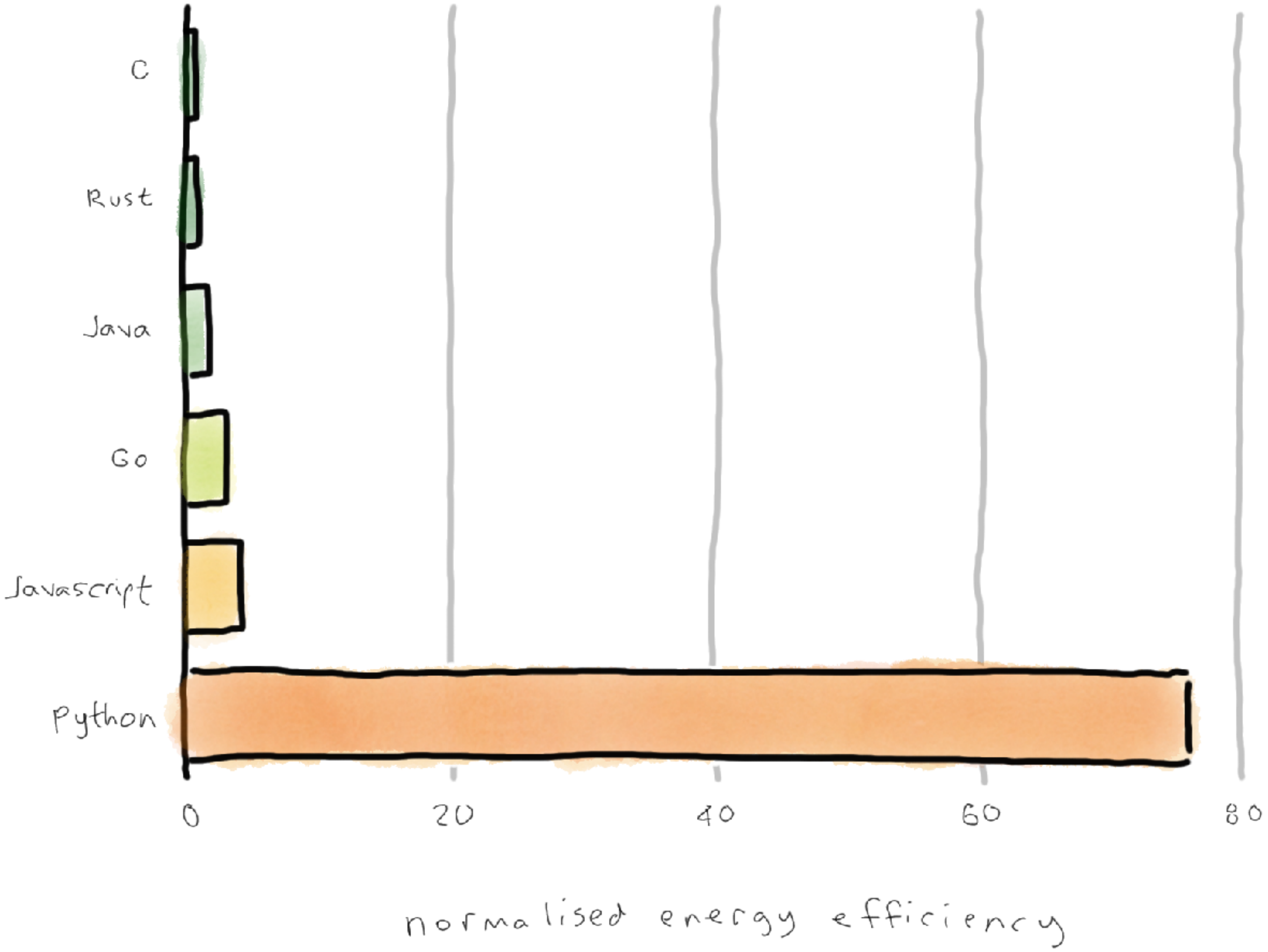
This paper presents a study of the runtime, memory usage and energy consumption of twenty seven well-known software languages. We monitor the performance of such languages using ten different programming problems, expressed in each of the languages. Our results show interesting findings, such as, slower/faster languages consuming less/more energy, and how memory usage influences energy consumption. We show how to use our results to provide software engineers support to decide which language to use when energy efficiency is a concern.

*CCS Concepts* • Software and its engineering → Soft-

productivity - by incorporating advanced features in the language design, like for instance powerful modular and type systems - and at efficiently execute such software - by developing, for example, aggressive compiler optimizations. Indeed, most techniques were developed with the main goal of helping software developers in producing faster programs. In fact, in the last century *performance* in software languages was in almost all cases synonymous of *fast execution time* (embedded systems were probably the single exception).

In this century, this reality is quickly changing and software energy consumption is becoming a key concern for computer manufacturers, software language engineers, pro-

# energy efficiency of programming languages



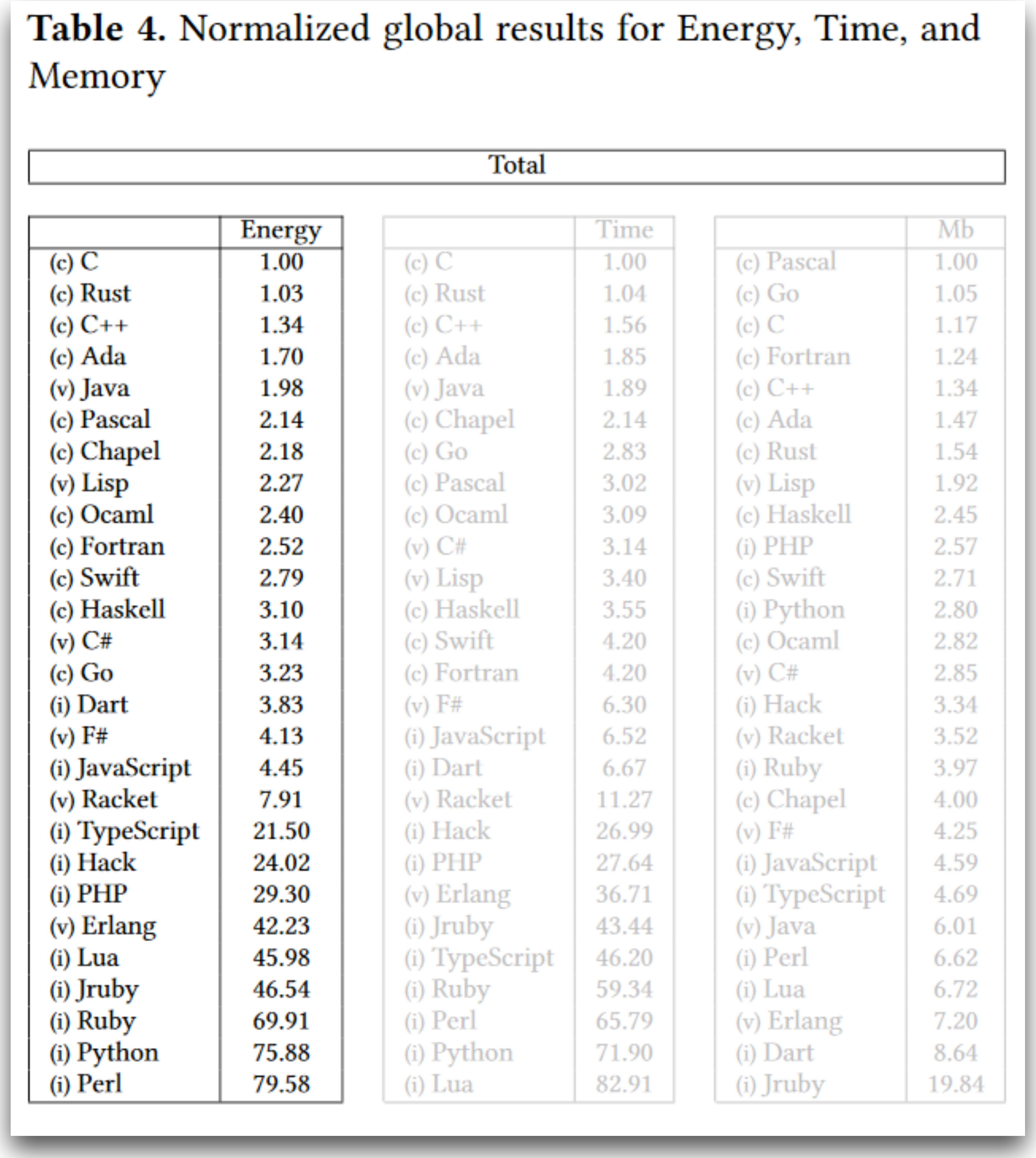


**Table 4.** Normalized global results for Energy, Time, and Memory

| Total | | | | | |
|---|---|---|---|---|---|
| | Energy | | Time | | Mb |
| (c) C | 1.00 | (c) C | 1.00 | (c) Pascal | 1.00 |
| (c) Rust | 1.03 | (c) Rust | 1.04 | (c) Go | 1.05 |
| (c) C++ | 1.34 | (c) C++ | 1.56 | (c) C | 1.17 |
| (c) Ada | 1.70 | (c) Ada | 1.85 | (c) Fortran | 1.24 |
| (v) Java | 1.98 | (v) Java | 1.89 | (c) C++ | 1.34 |
| (c) Pascal | 2.14 | (c) Chapel | 2.14 | (c) Ada | 1.47 |
| (c) Chapel | 2.18 | (c) Go | 2.83 | (c) Rust | 1.54 |
| (v) Lisp | 2.27 | (c) Pascal | 3.02 | (v) Lisp | 1.92 |
| (c) Ocaml | 2.40 | (c) Ocaml | 3.09 | (c) Haskell | 2.45 |
| (c) Fortran | 2.52 | (v) C# | 3.14 | (i) PHP | 2.57 |
| (c) Swift | 2.79 | (v) Lisp | 3.40 | (c) Swift | 2.71 |
| (c) Haskell | 3.10 | (c) Haskell | 3.55 | (i) Python | 2.80 |
| (v) C# | 3.14 | (c) Swift | 4.20 | (c) Ocaml | 2.82 |
| (c) Go | 3.23 | (c) Fortran | 4.20 | (v) C# | 2.85 |
| (i) Dart | 3.83 | (v) F# | 6.30 | (i) Hack | 3.34 |
| (v) F# | 4.13 | (i) JavaScript | 6.52 | (v) Racket | 3.52 |
| (i) JavaScript | 4.45 | (i) Dart | 6.67 | (i) Ruby | 3.97 |
| (v) Racket | 7.91 | (v) Racket | 11.27 | (c) Chapel | 4.00 |
| (i) TypeScript | 21.50 | (i) Hack | 26.99 | (v) F# | 4.25 |
| (i) Hack | 24.02 | (i) PHP | 27.64 | (i) JavaScript | 4.59 |
| (i) PHP | 29.30 | (v) Erlang | 36.71 | (i) TypeScript | 4.69 |
| (v) Erlang | 42.23 | (i) Jruby | 43.44 | (v) Java | 6.01 |
| (i) Lua | 45.98 | (i) TypeScript | 46.20 | (i) Perl | 6.62 |
| (i) Jruby | 46.54 | (i) Ruby | 59.34 | (i) Lua | 6.72 |
| (i) Ruby | 69.91 | (i) Perl | 65.79 | (v) Erlang | 7.20 |
| (i) Python | 75.88 | (i) Python | 71.90 | (i) Dart | 8.64 |
| (i) Perl | 79.58 | (i) Lua | 82.91 | (i) Jruby | 19.84 |

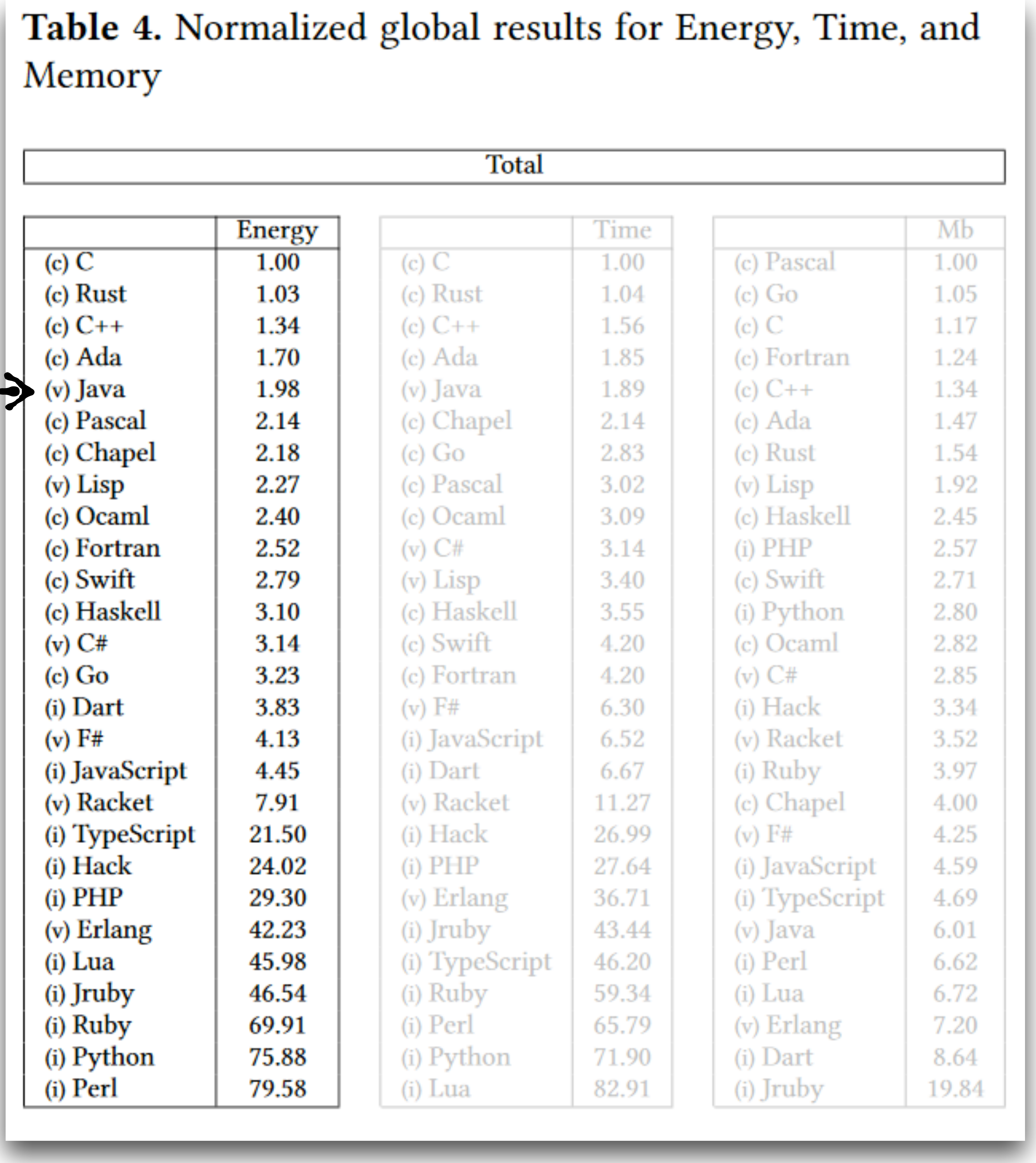
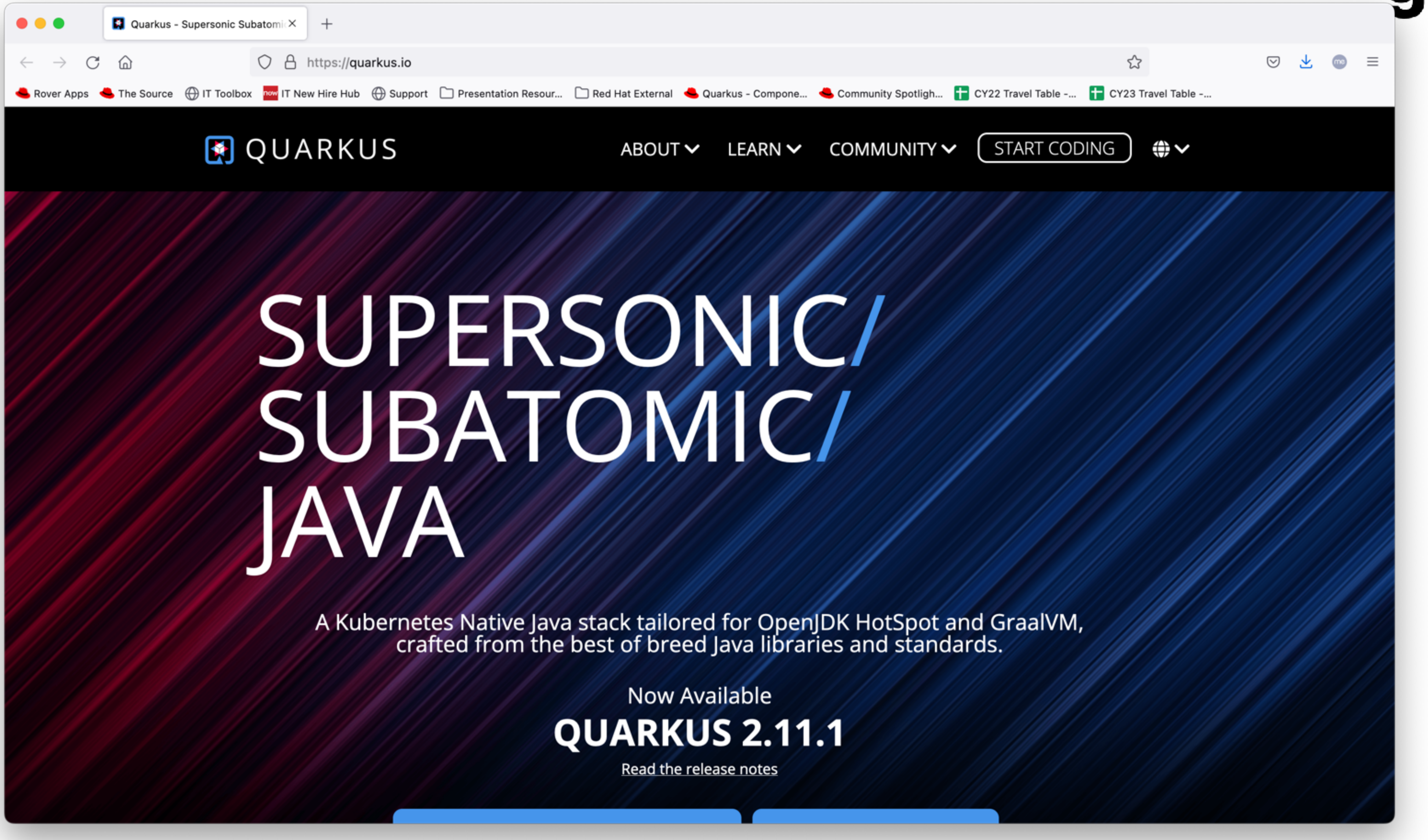


**Table 4.** Normalized global results for Energy, Time, and Memory

| Total | | | | | |
|---|---|---|---|---|---|
| | Energy | | Time | | Mb |
| (c) C | 1.00 | (c) C | 1.00 | (c) Pascal | 1.00 |
| (c) Rust | 1.03 | (c) Rust | 1.04 | (c) Go | 1.05 |
| (c) C++ | 1.34 | (c) C++ | 1.56 | (c) C | 1.17 |
| (c) Ada | 1.70 | (c) Ada | 1.85 | (c) Fortran | 1.24 |
| (v) Java | 1.98 | (v) Java | 1.89 | (c) C++ | 1.34 |
| (c) Pascal | 2.14 | (c) Chapel | 2.14 | (c) Ada | 1.47 |
| (c) Chapel | 2.18 | (c) Go | 2.83 | (c) Rust | 1.54 |
| (v) Lisp | 2.27 | (c) Pascal | 3.02 | (v) Lisp | 1.92 |
| (c) Ocaml | 2.40 | (c) Ocaml | 3.09 | (c) Haskell | 2.45 |
| (c) Fortran | 2.52 | (v) C# | 3.14 | (i) PHP | 2.57 |
| (c) Swift | 2.79 | (v) Lisp | 3.40 | (c) Swift | 2.71 |
| (c) Haskell | 3.10 | (c) Haskell | 3.55 | (i) Python | 2.80 |
| (v) C# | 3.14 | (c) Swift | 4.20 | (c) Ocaml | 2.82 |
| (c) Go | 3.23 | (c) Fortran | 4.20 | (v) C# | 2.85 |
| (i) Dart | 3.83 | (v) F# | 6.30 | (i) Hack | 3.34 |
| (v) F# | 4.13 | (i) JavaScript | 6.52 | (v) Racket | 3.52 |
| (i) JavaScript | 4.45 | (i) Dart | 6.67 | (i) Ruby | 3.97 |
| (v) Racket | 7.91 | (v) Racket | 11.27 | (c) Chapel | 4.00 |
| (i) TypeScript | 21.50 | (i) Hack | 26.99 | (v) F# | 4.25 |
| (i) Hack | 24.02 | (i) PHP | 27.64 | (i) JavaScript | 4.59 |
| (i) PHP | 29.30 | (v) Erlang | 36.71 | (i) TypeScript | 4.69 |
| (v) Erlang | 42.23 | (i) Jruby | 43.44 | (v) Java | 6.01 |
| (i) Lua | 45.98 | (i) TypeScript | 46.20 | (i) Perl | 6.62 |
| (i) Jruby | 46.54 | (i) Ruby | 59.34 | (i) Lua | 6.72 |
| (i) Ruby | 69.91 | (i) Perl | 65.79 | (v) Erlang | 7.20 |
| (i) Python | 75.88 | (i) Python | 71.90 | (i) Dart | 8.64 |
| (i) Perl | 79.58 | (i) Lua | 82.91 | (i) Jruby | 19.84 |

# what’s the most carbon-efficient java?



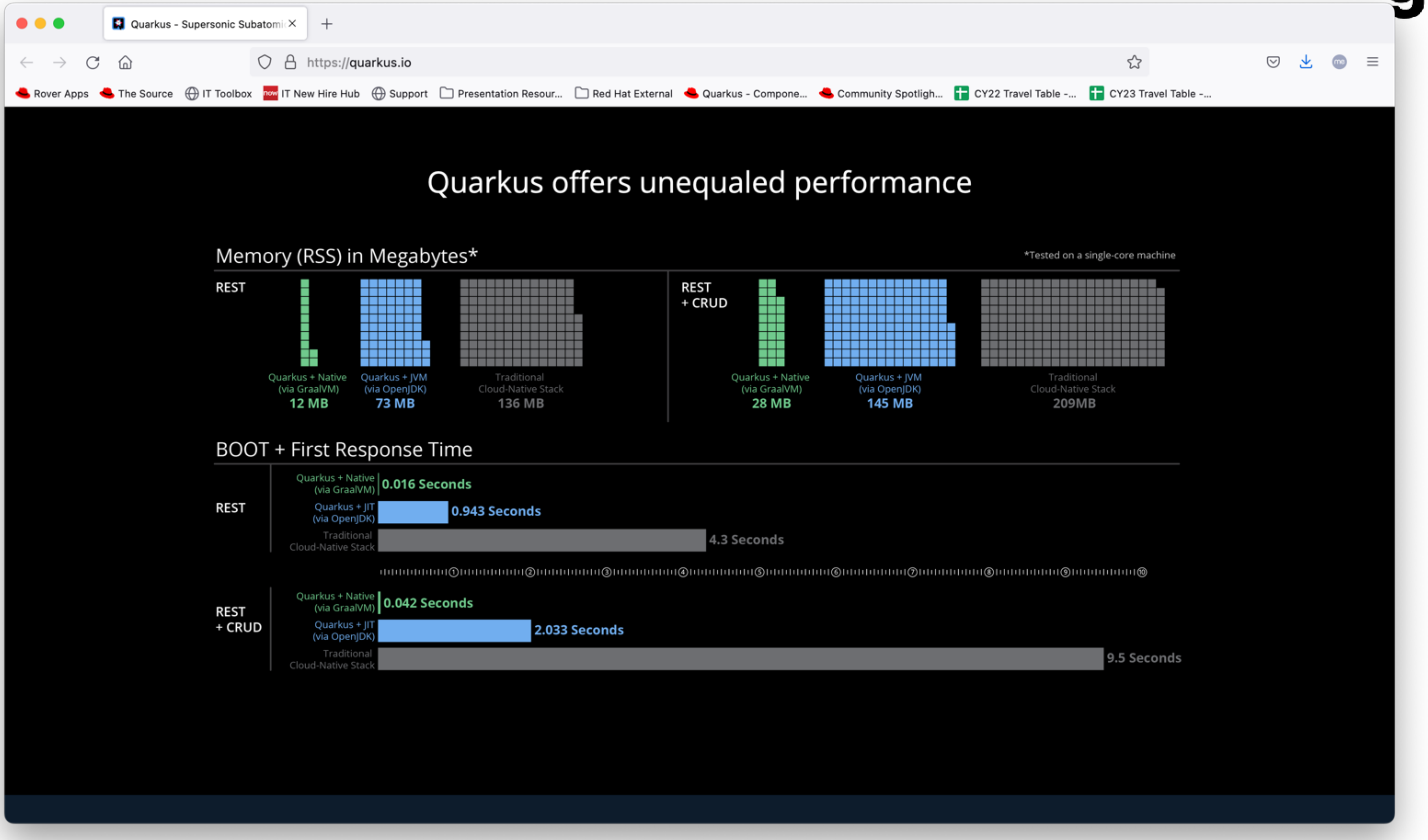
SUPERSONIC/
SUBATOMIC/
JAVA

A Kubernetes Native Java stack tailored for OpenJDK HotSpot and GraalVM, crafted from the best of breed Java libraries and standards.

Now Available

**QUARKUS 2.11.1**

Read the release notes


Quarkus - Supersonic Subatomi
https://quarkus.io
Rover Apps
The Source
IT Toolbox
IT New Hire Hub
Support
Presentation Resour...
Red Hat External
Quarkus - Compone...
Community Spotligh...
CY22 Travel Table -...
CY23 Travel Table -...

# Quarkus offers unequaled performance

## Memory (RSS) in Megabytes*

*Tested on a single-core machine

REST
Quarkus + Native (via GraalVM) 12 MB
Quarkus + JVM (via OpenJDK) 73 MB
Traditional Cloud-Native Stack 136 MB
REST + CRUD
Quarkus + Native (via GraalVM) 28 MB
Quarkus + JVM (via OpenJDK) 145 MB
Traditional Cloud-Native Stack 209MB

## BOOT + First Response Time

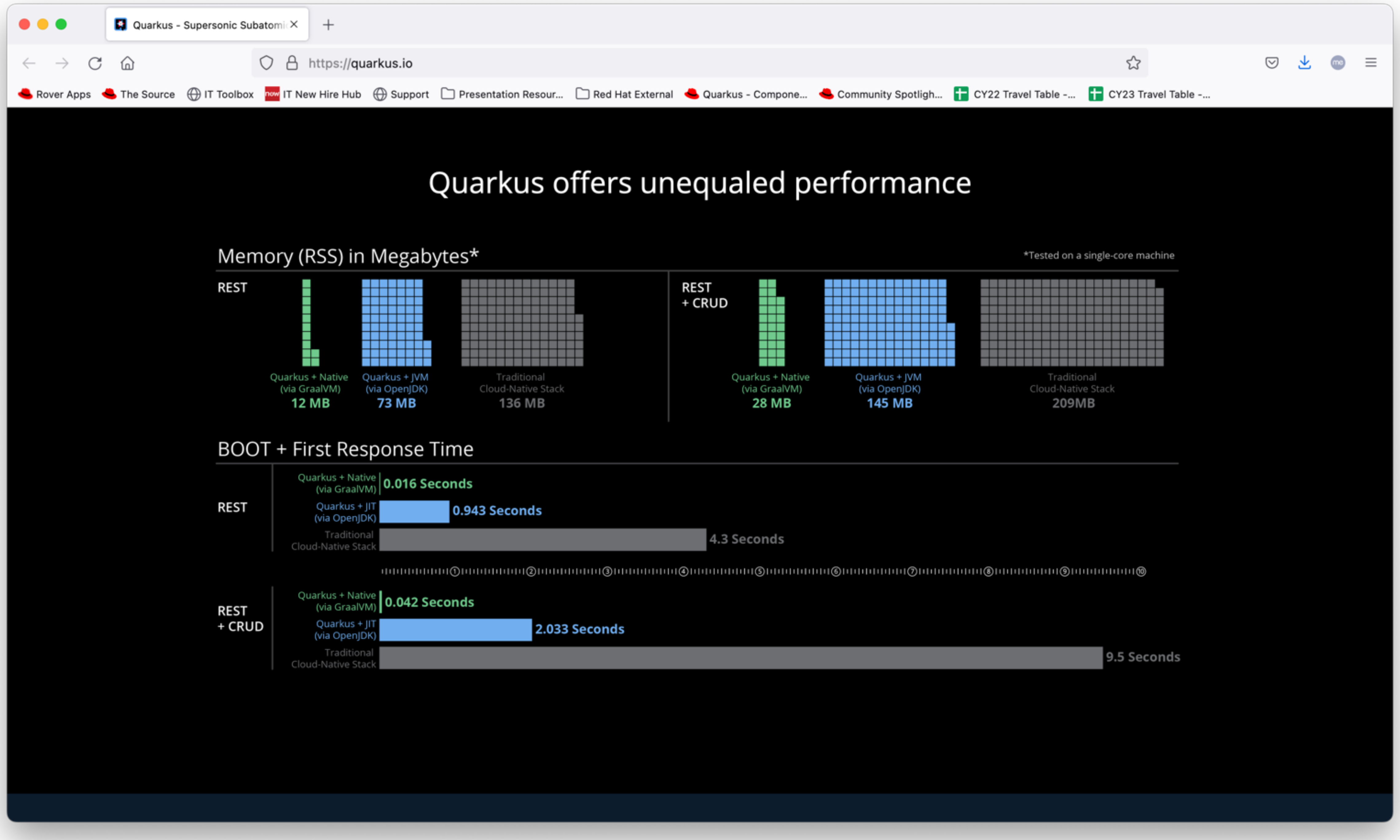
REST
Quarkus + Native (via GraalVM) 0.016 Seconds
Quarkus + JIT (via OpenJDK) 0.943 Seconds
Traditional Cloud-Native Stack 4.3 Seconds
1
2
3
4
5
6
7
8
9
10
REST + CRUD
Quarkus + Native (via GraalVM) 0.042 Seconds
Quarkus + JIT (via OpenJDK) 2.033 Seconds
Traditional Cloud-Native Stack 9.5 Seconds

goto; aarhus

@holly_cummins

#RedHat

# does being small and fast reduce carbon footprint?


Quarkus - Supersonic Subatomi
https://quarkus.io
Rover Apps
The Source
IT Toolbox
IT New Hire Hub
Support
Presentation Resour...
Red Hat External
Quarkus - Compone...
Community Spotligh...
CY22 Travel Table -...
CY23 Travel Table -...

Quarkus offers unequaled performance

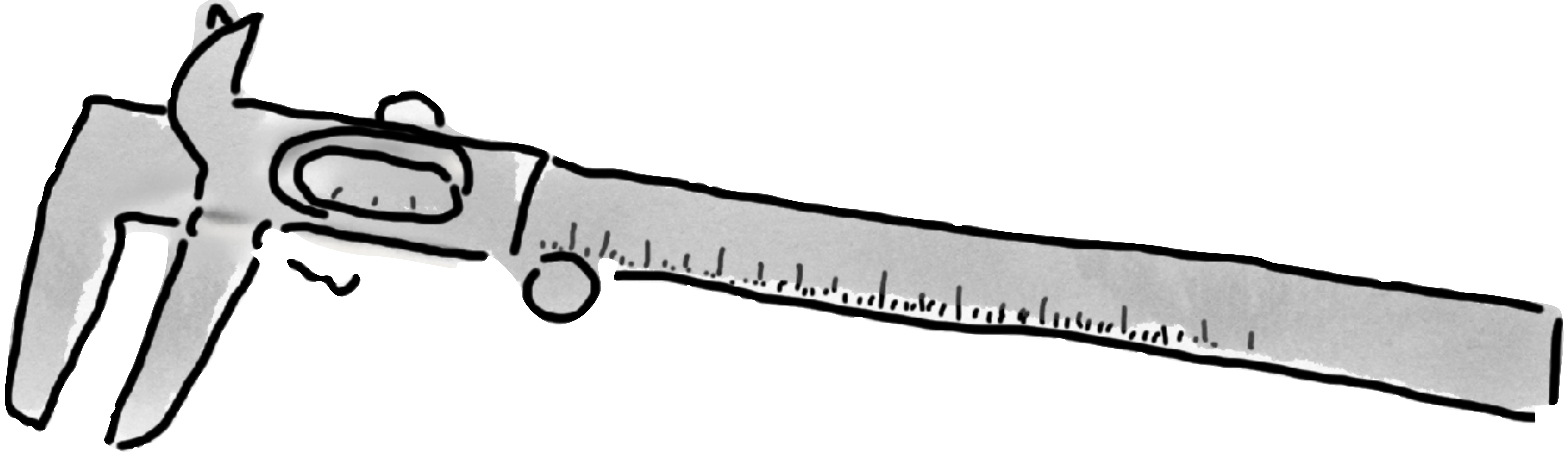
Memory (RSS) in Megabytes*

*Tested on a single-core machine

REST
Quarkus + Native (via GraalVM) 12 MB
Quarkus + JVM (via OpenJDK) 73 MB
Traditional Cloud-Native Stack 136 MB
REST + CRUD
Quarkus + Native (via GraalVM) 28 MB
Quarkus + JVM (via OpenJDK) 145 MB
Traditional Cloud-Native Stack 209MB

BOOT + First Response Time

REST
Quarkus + Native (via GraalVM) 0.016 Seconds
Quarkus + JIT (via OpenJDK) 0.943 Seconds
Traditional Cloud-Native Stack 4.3 Seconds
1
2
3
4
5
6
7
8
9
10
REST + CRUD
Quarkus + Native (via GraalVM) 0.042 Seconds
Quarkus + JIT (via OpenJDK) 2.033 Seconds
Traditional Cloud-Native Stack 9.5 Seconds



measure, don’t guess.

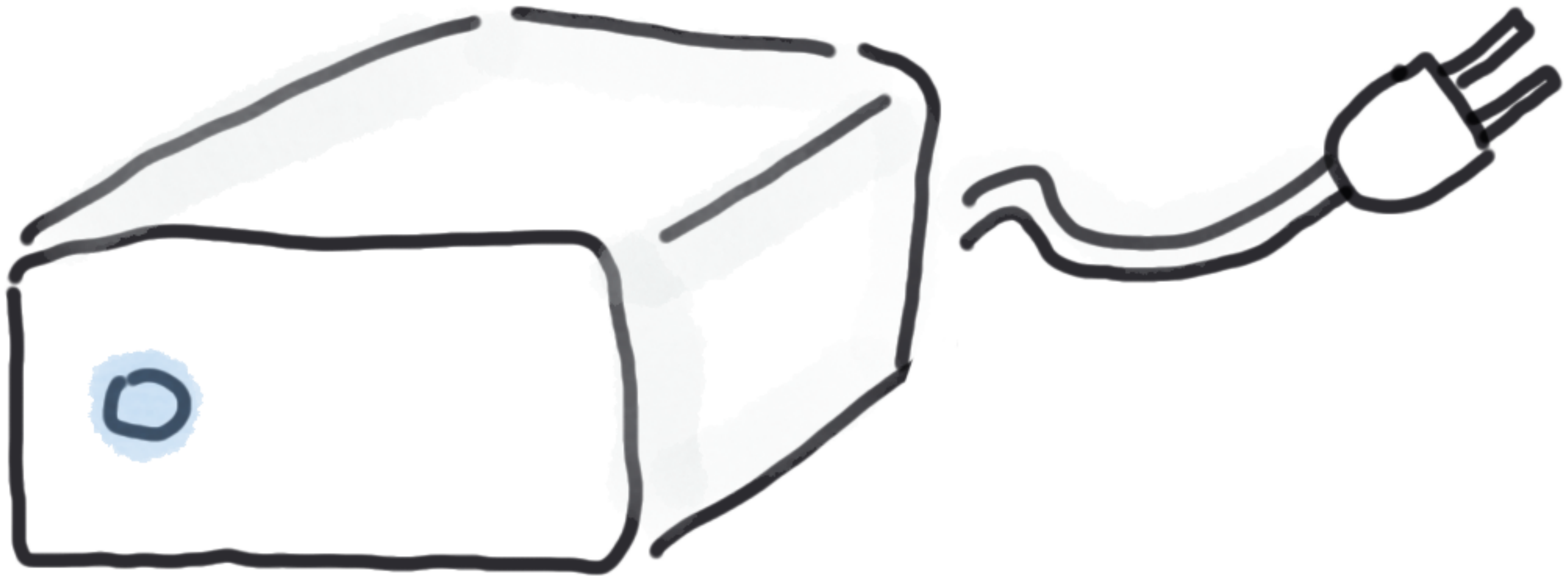
# digression:
# measuring carbon is **hard**

# step 1: measure power usage

# step 1: measure power usage

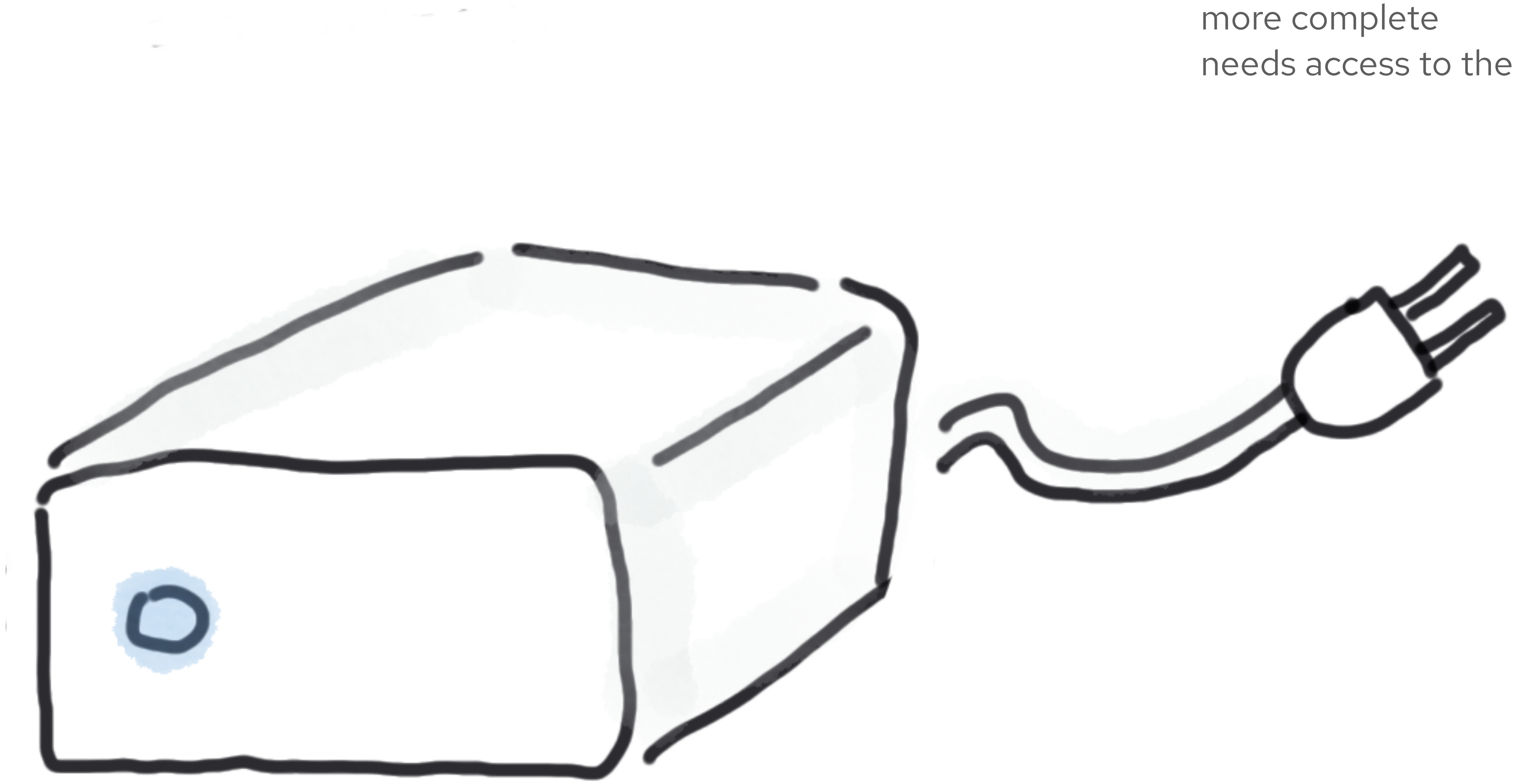
wall power measurement

# step 1: measure power usage



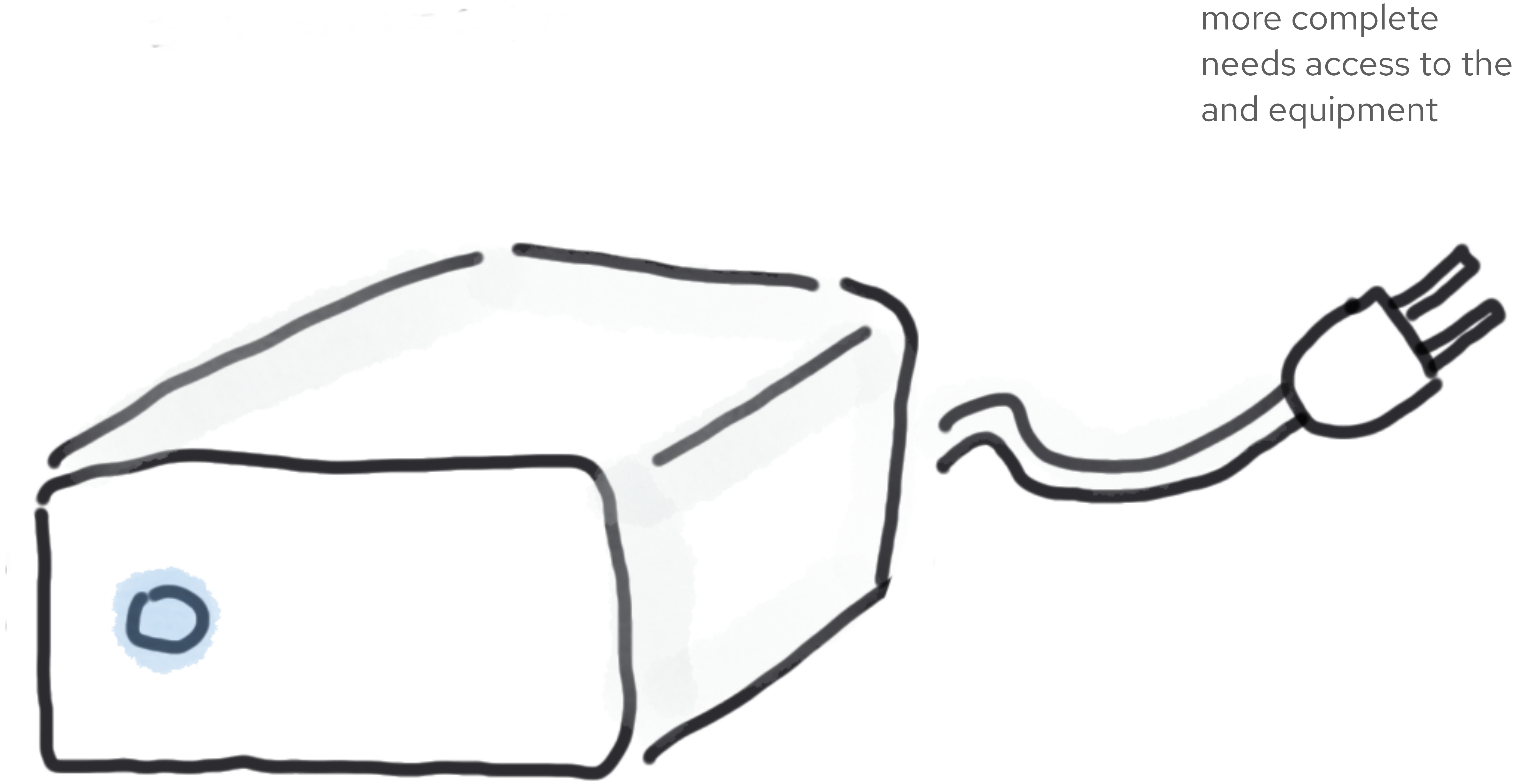
**wall power measurement**
more complete

# step 1: measure power usage



**wall power measurement**
more complete
needs access to the wall

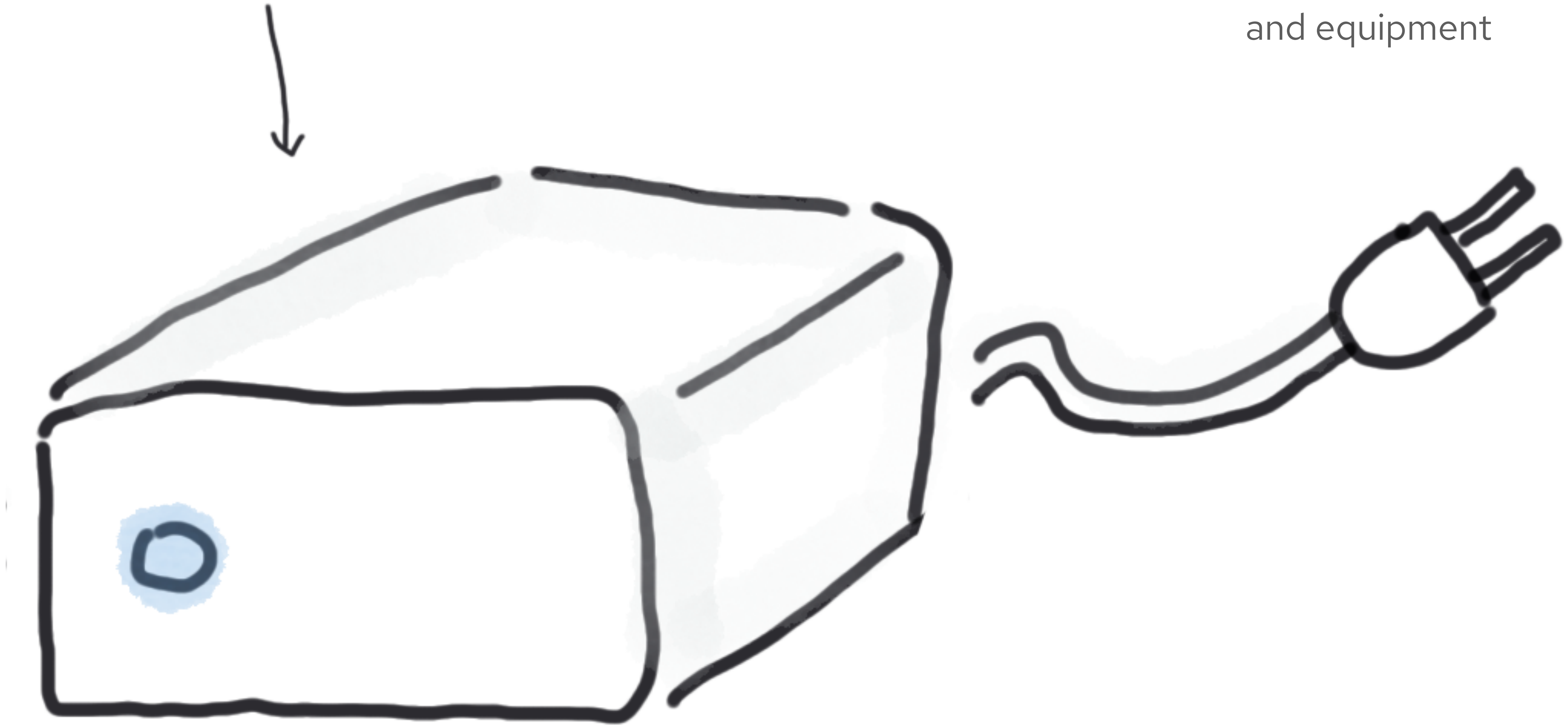
# step 1: measure power usage



**wall power measurement**
more complete
needs access to the wall
and equipment

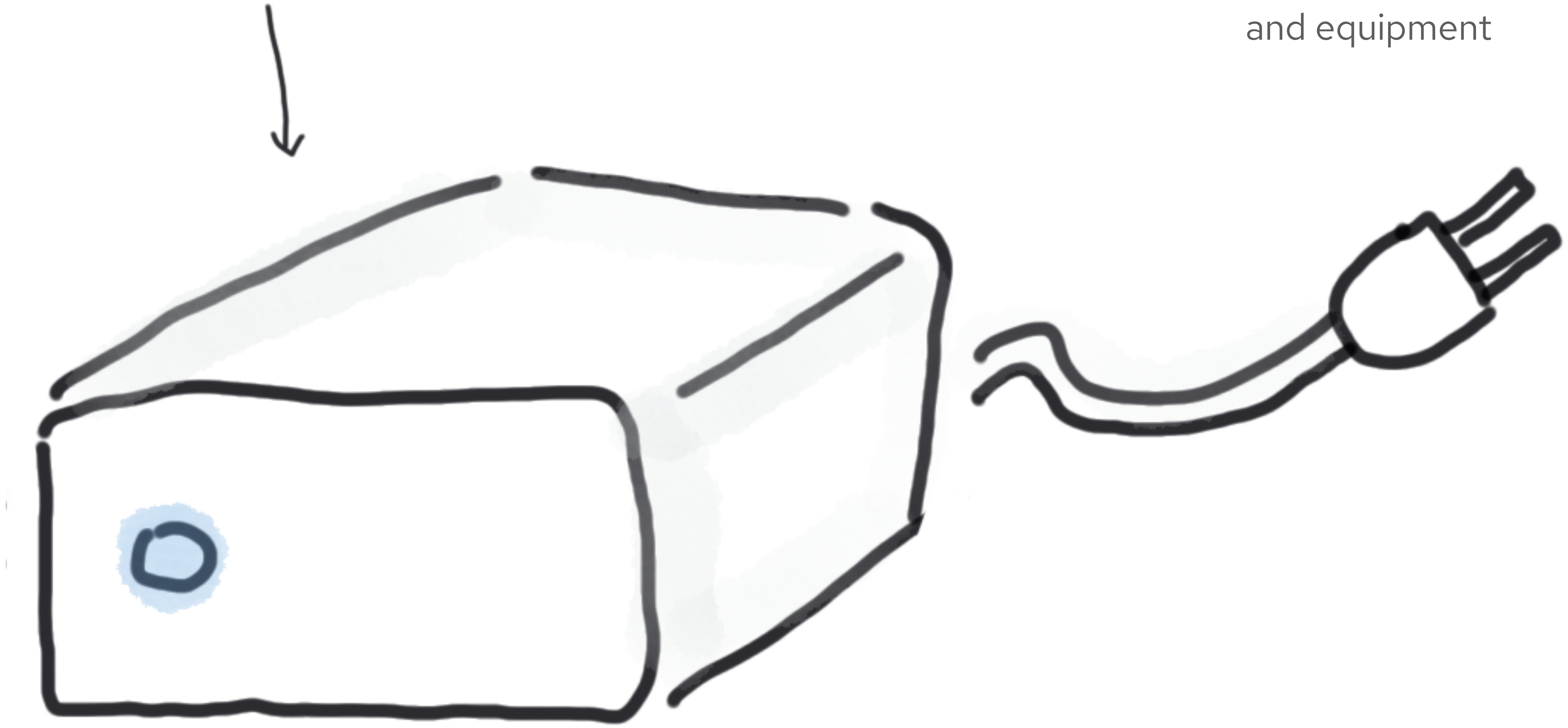
# step 1: measure power usage

**RAPL**



**wall power measurement**
more complete
needs access to the wall
and equipment

# step 1: measure power usage



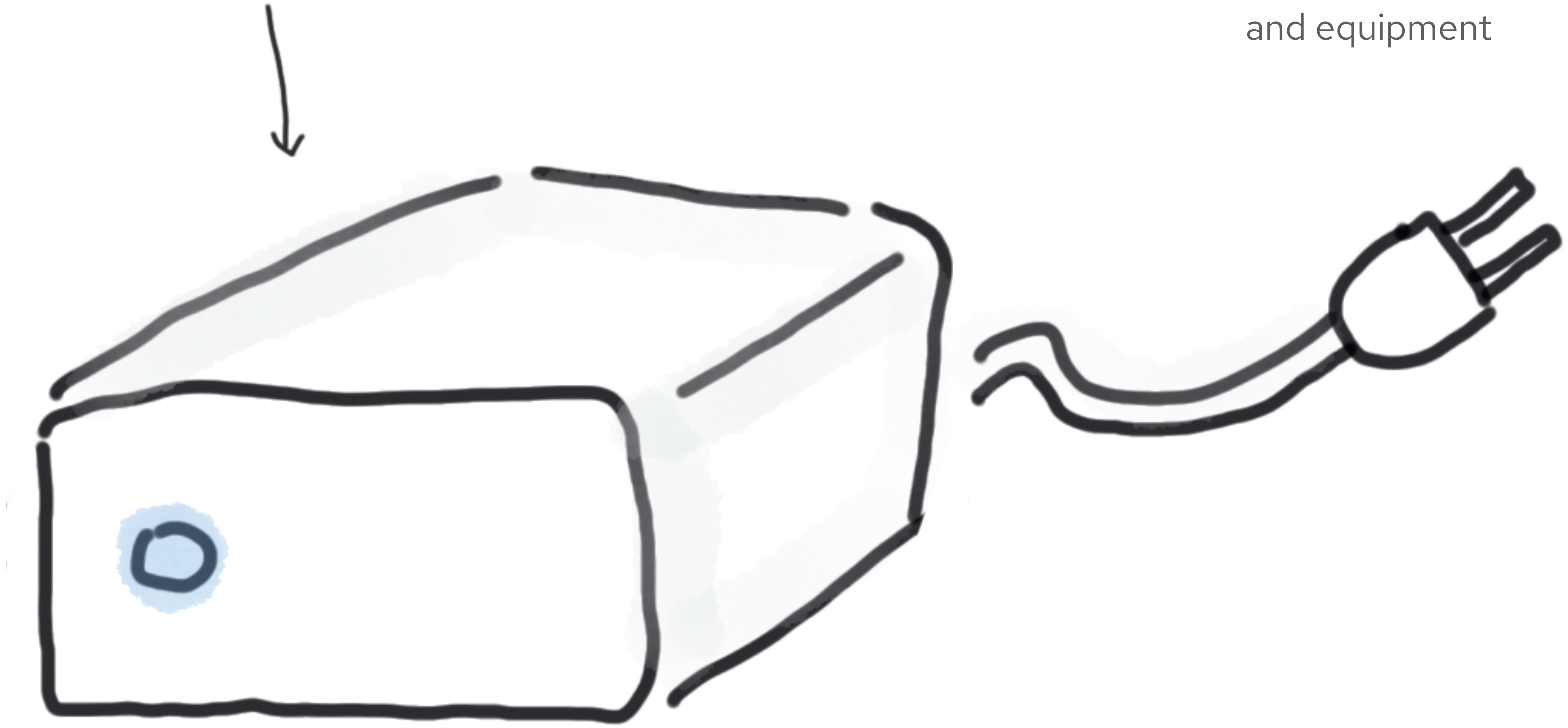
**RAPL**
programmatically accessible

**wall power measurement**
more complete
needs access to the wall
and equipment

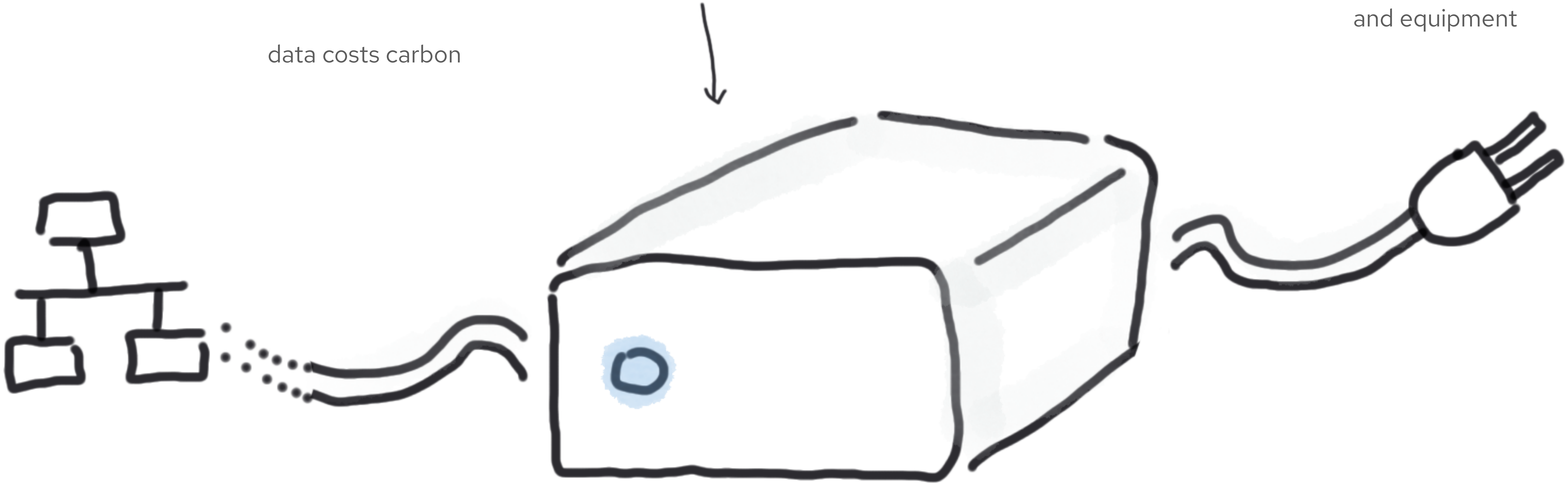
# step 1: measure power usage

**RAPL**
programmatically accessible
misses some components

**wall power measurement**
more complete
needs access to the wall
and equipment



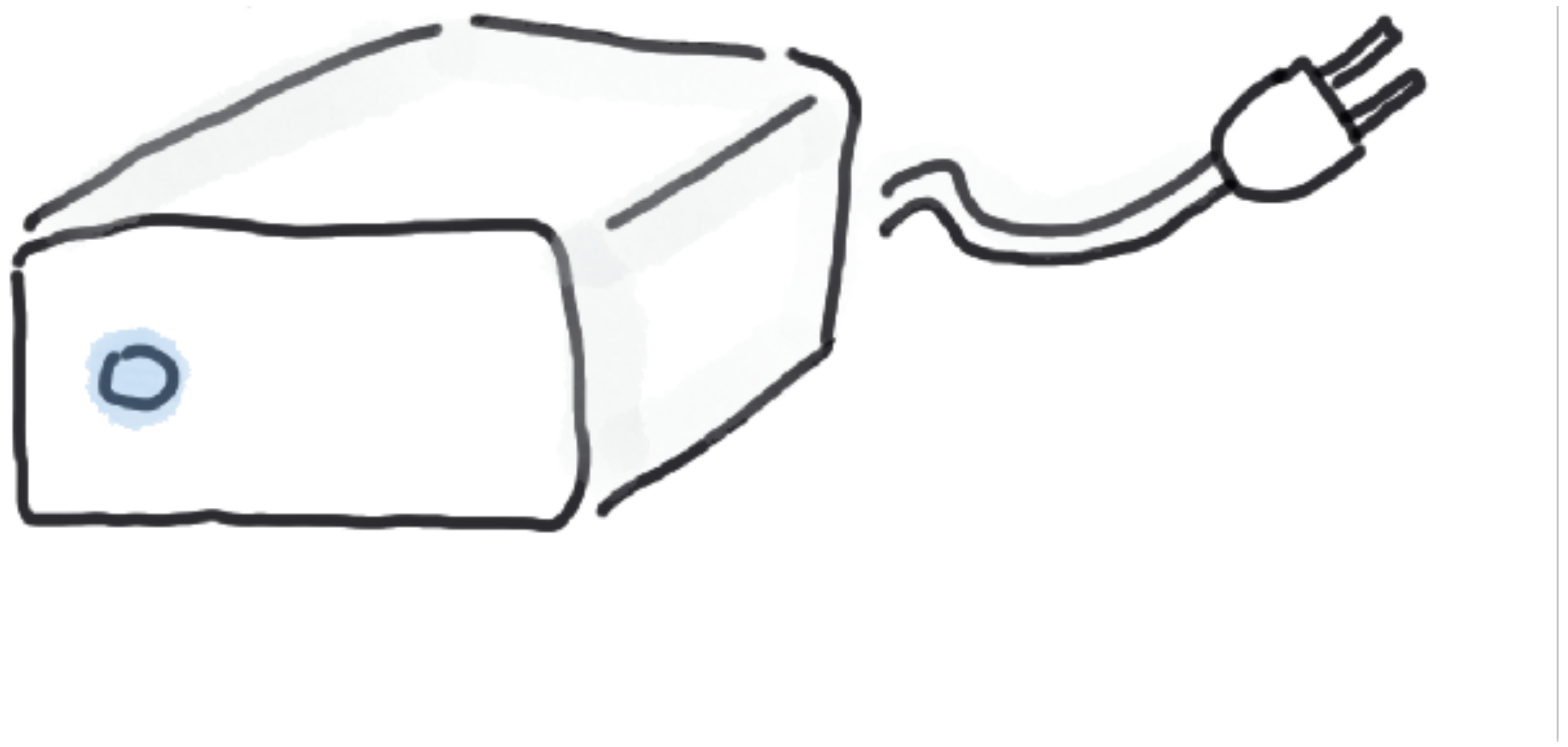
# step 1: measure power usage

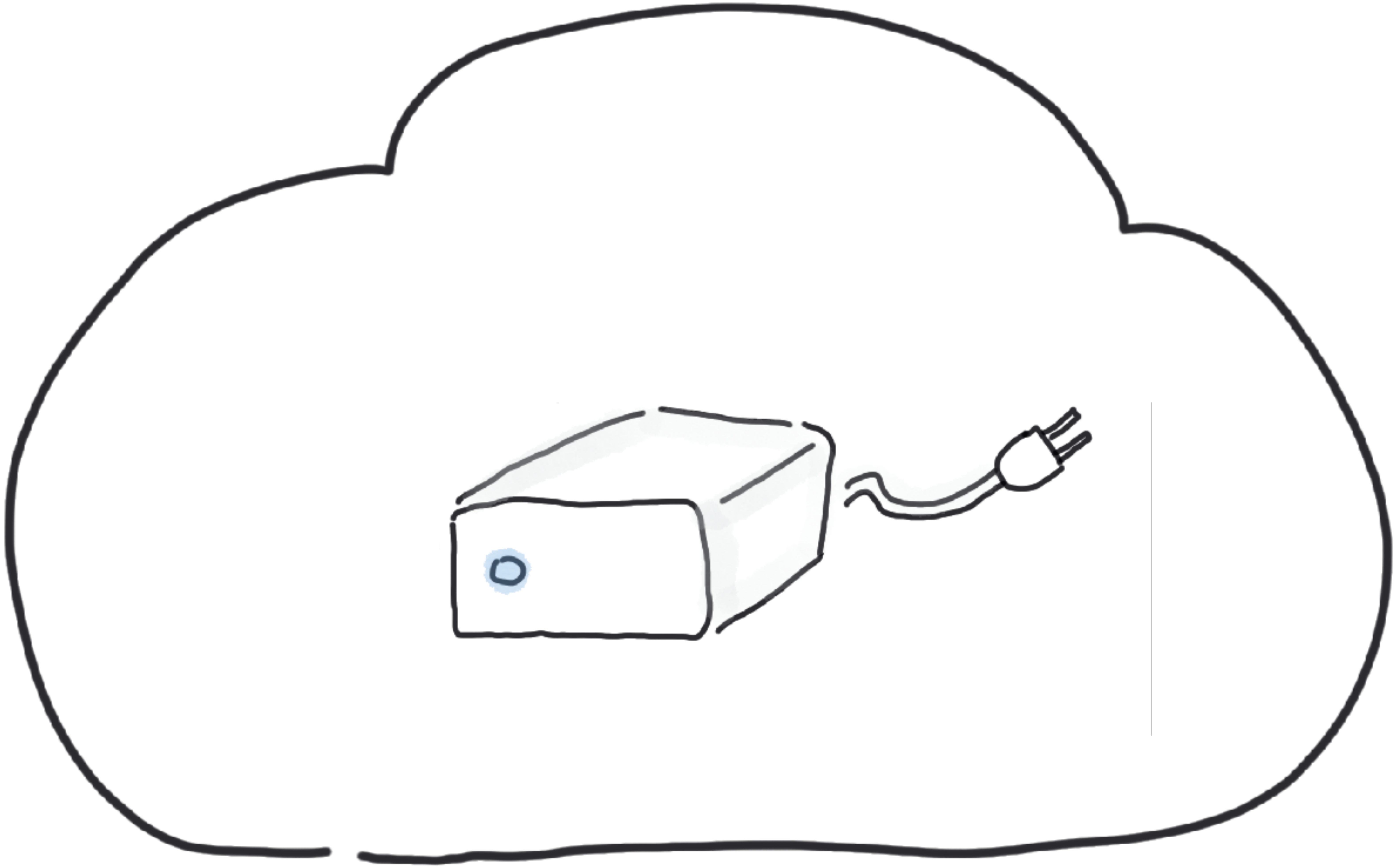


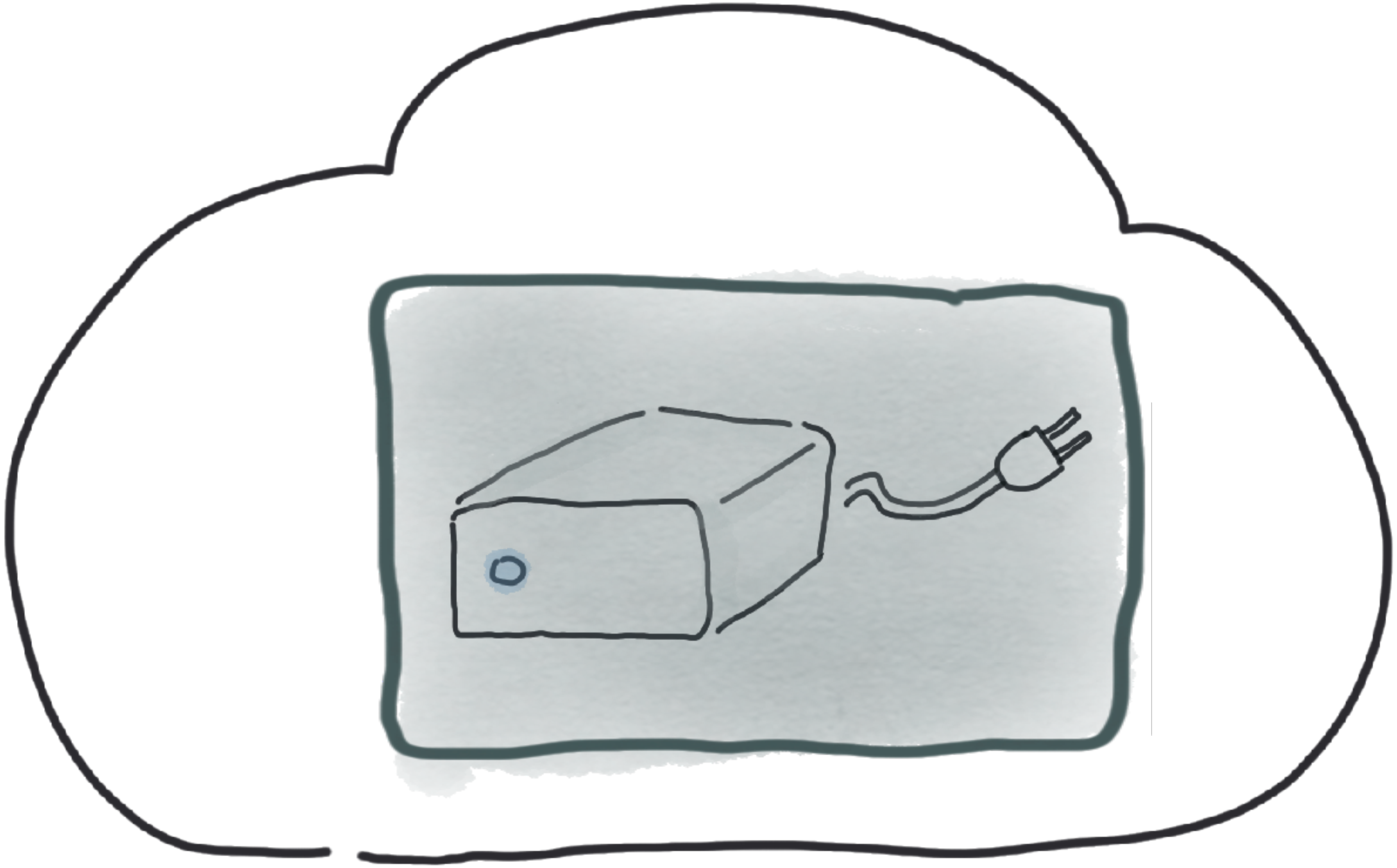
data costs carbon

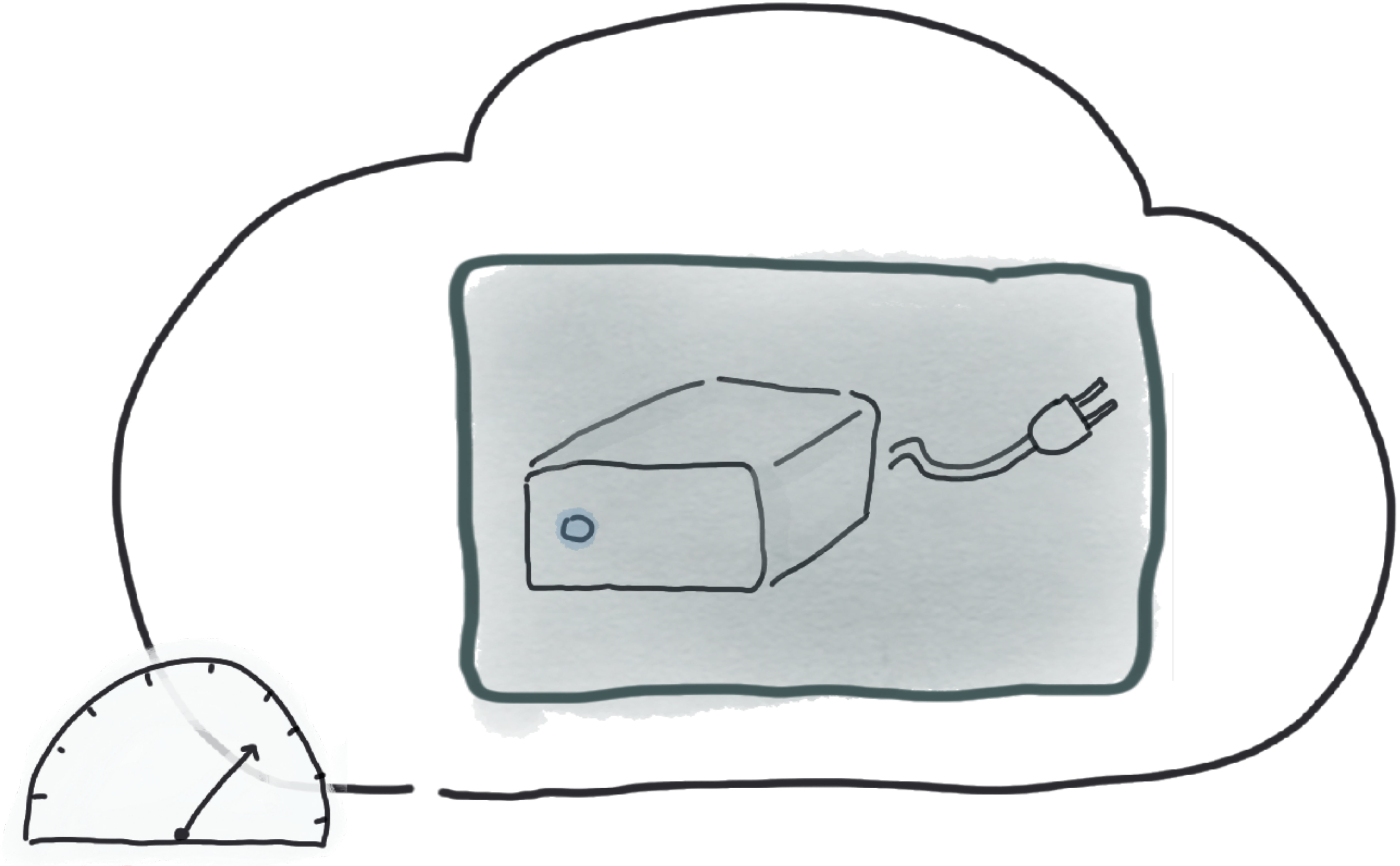
**RAPL**
programmatically accessible
misses some components

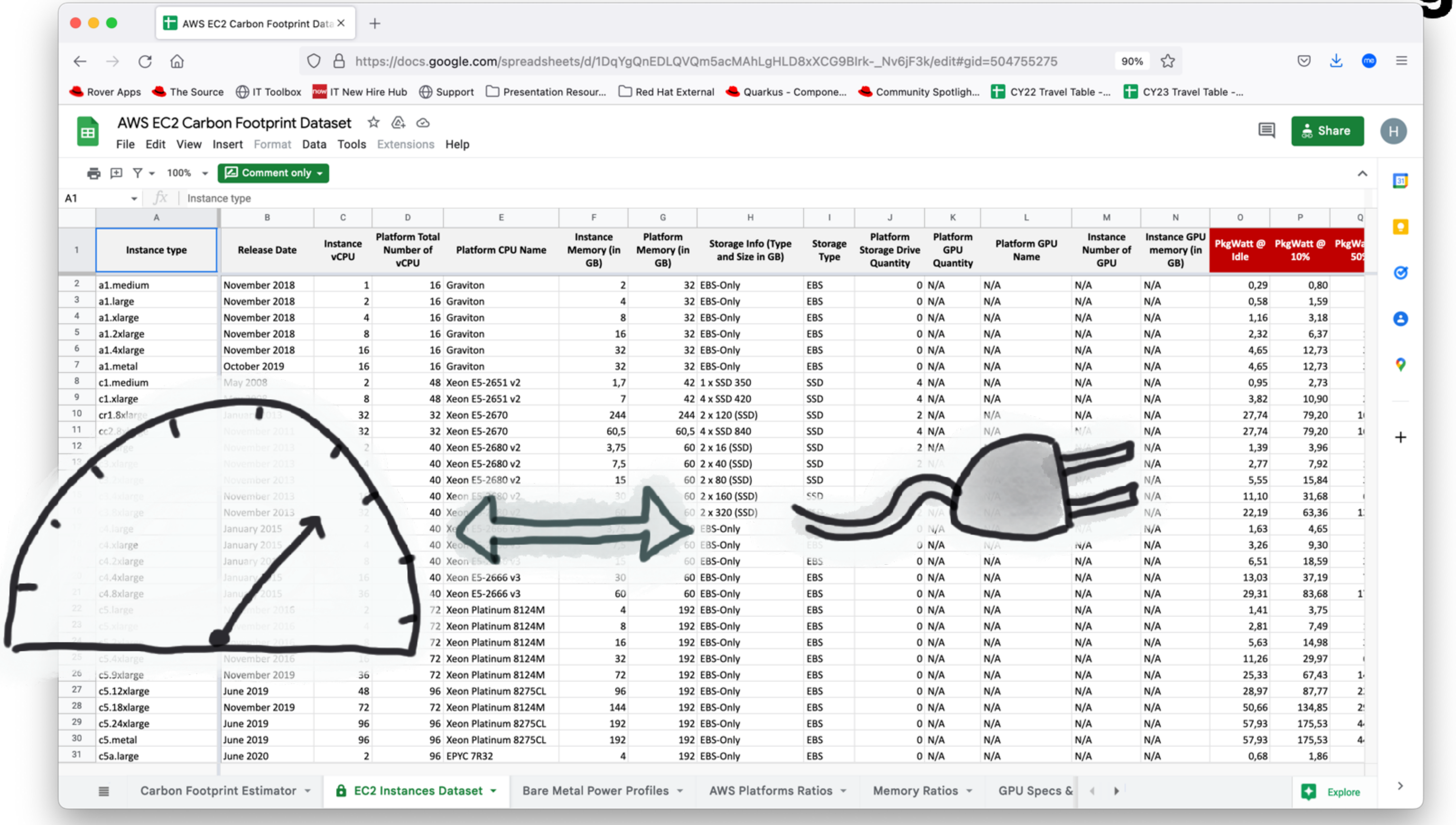
**wall power measurement**
more complete
needs access to the wall
and equipment







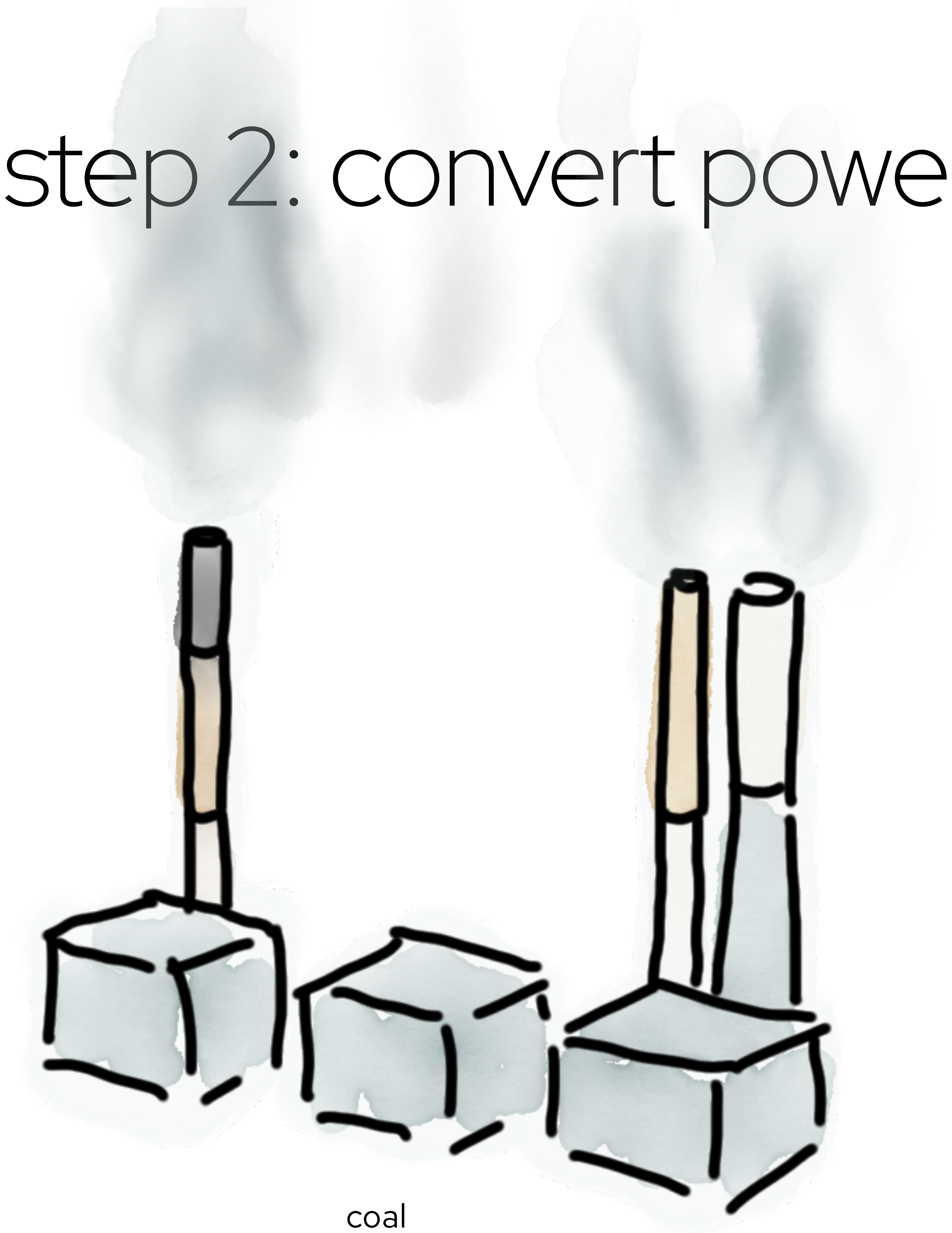

load



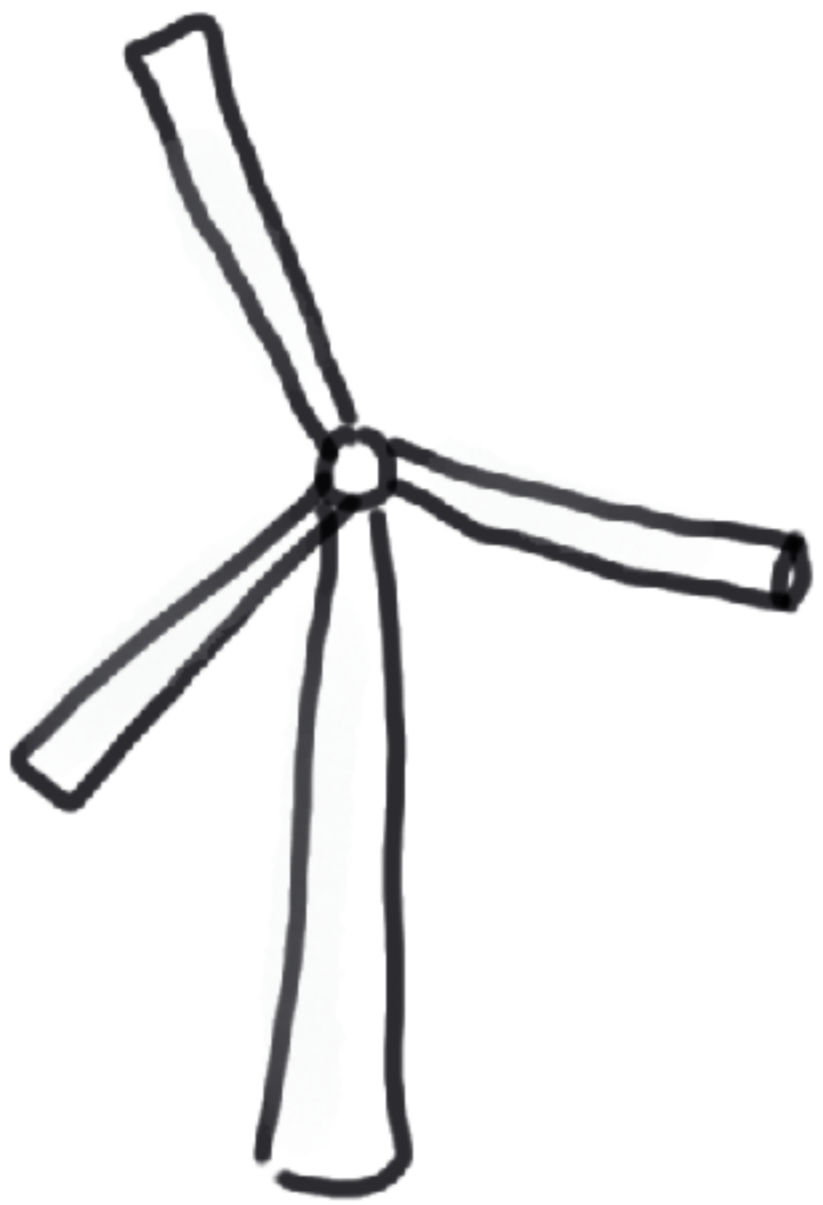
| | Instance type | Release Date | Instance vCPU | Platform Total Number of vCPU | Platform CPU Name | Instance Memory (in GB) | Platform Memory (in GB) | Storage Info (Type and Size in GB) | Storage Type | Platform Storage Drive Quantity | Platform GPU Quantity | Platform GPU Name | Instance Number of GPU | Instance GPU memory (in GB) | PkgWatt @ Idle | PkgWatt @ 10% |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | a1.medium | November 2018 | 1 | 16 | Graviton | 2 | 32 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 0,29 | 0,80 |
| 3 | a1.large | November 2018 | 2 | 16 | Graviton | 4 | 32 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 0,58 | 1,59 |
| 4 | a1.xlarge | November 2018 | 4 | 16 | Graviton | 8 | 32 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 1,16 | 3,18 |
| 5 | a1.2xlarge | November 2018 | 8 | 16 | Graviton | 16 | 32 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 2,32 | 6,37 |
| 6 | a1.4xlarge | November 2018 | 16 | 16 | Graviton | 32 | 32 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 4,65 | 12,73 |
| 7 | a1.metal | October 2019 | 16 | 16 | Graviton | 32 | 32 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 4,65 | 12,73 |
| 8 | c1.medium | May 2008 | 2 | 48 | Xeon E5-2651 v2 | 1,7 | 42 | 1 x SSD 350 | SSD | 4 | N/A | N/A | N/A | N/A | 0,95 | 2,73 |
| 9 | c1.xlarge | May 2008 | 8 | 48 | Xeon E5-2651 v2 | 7 | 42 | 4 x SSD 420 | SSD | 4 | N/A | N/A | N/A | N/A | 3,82 | 10,90 |
| 10 | cr1.8xlarge | January 2013 | 32 | 32 | Xeon E5-2670 | 244 | 244 | 2 x 120 (SSD) | SSD | 2 | N/A | N/A | N/A | N/A | 27,74 | 79,20 |
| 11 | cc2.8xlarge | November 2011 | 32 | 32 | Xeon E5-2670 | 60,5 | 60,5 | 4 x SSD 840 | SSD | 4 | N/A | N/A | N/A | N/A | 27,74 | 79,20 |
| 12 | c3.large | November 2013 | 2 | 40 | Xeon E5-2680 v2 | 3,75 | 60 | 2 x 16 (SSD) | SSD | 2 | N/A | [illegible] | N/A | N/A | 1,39 | 3,96 |
| 13 | c3.xlarge | November 2013 | 4 | 40 | Xeon E5-2680 v2 | 7,5 | 60 | 2 x 40 (SSD) | SSD | 2 | N/A | [illegible] | N/A | N/A | 2,77 | 7,92 |
| 14 | c3.2xlarge | November 2013 | [illegible] | 40 | Xeon E5-2680 v2 | 15 | 60 | 2 x 80 (SSD) | SSD | [illegible] | [illegible] | N/A | N/A | N/A | 5,55 | 15,84 |
| 15 | c3.4xlarge | November 2013 | 16 | 40 | Xeon E5-2680 v2 | 30 | 60 | 2 x 160 (SSD) | SSD | [illegible] | [illegible] | N/A | [illegible] | N/A | 11,10 | 31,68 |
| 16 | c3.8xlarge | November 2013 | 32 | 40 | Xeon E5-2680 v2 | 60 | 60 | 2 x 320 (SSD) | [illegible] | 2 | N/A | N/A | [illegible] | N/A | 22,19 | 63,36 |
| 17 | c4.large | January 2015 | 2 | 40 | Xeon E5-2666 v3 | 3,75 | [illegible] | EBS-Only | [illegible] | 0 | N/A | N/A | N/A | N/A | 1,63 | 4,65 |
| 18 | c4.xlarge | January 2015 | 4 | 40 | Xeon E5-2666 v3 | 7,5 | 60 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 3,26 | 9,30 |
| 19 | c4.2xlarge | January 2015 | 8 | 40 | Xeon E5-2666 v3 | 15 | 60 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 6,51 | 18,59 |
| 20 | c4.4xlarge | January 2015 | 16 | 40 | Xeon E5-2666 v3 | 30 | 60 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 13,03 | 37,19 |
| 21 | c4.8xlarge | January 2015 | 36 | 40 | Xeon E5-2666 v3 | 60 | 60 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 29,31 | 83,68 |
| 22 | c5.large | November 2016 | 2 | 72 | Xeon Platinum 8124M | 4 | 192 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 1,41 | 3,75 |
| 23 | c5.xlarge | November 2016 | 4 | 72 | Xeon Platinum 8124M | 8 | 192 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 2,81 | 7,49 |
| 24 | c5.2xlarge | November 2016 | 8 | 72 | Xeon Platinum 8124M | 16 | 192 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 5,63 | 14,98 |
| 25 | c5.4xlarge | November 2016 | 16 | 72 | Xeon Platinum 8124M | 32 | 192 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 11,26 | 29,97 |
| 26 | c5.9xlarge | November 2019 | 36 | 72 | Xeon Platinum 8124M | 72 | 192 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 25,33 | 67,43 |
| 27 | c5.12xlarge | June 2019 | 48 | 96 | Xeon Platinum 8275CL | 96 | 192 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 28,97 | 87,77 |
| 28 | c5.18xlarge | November 2019 | 72 | 72 | Xeon Platinum 8124M | 144 | 192 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 50,66 | 134,85 |
| 29 | c5.24xlarge | June 2019 | 96 | 96 | Xeon Platinum 8275CL | 192 | 192 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 57,93 | 175,53 |
| 30 | c5.metal | June 2019 | 96 | 96 | Xeon Platinum 8275CL | 192 | 192 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 57,93 | 175,53 |
| 31 | c5a.large | June 2020 | 2 | 96 | EPYC 7R32 | 4 | 192 | EBS-Only | EBS | 0 | N/A | N/A | N/A | N/A | 0,68 | 1,86 |

Source: Teads EC2 instances carbon dataset

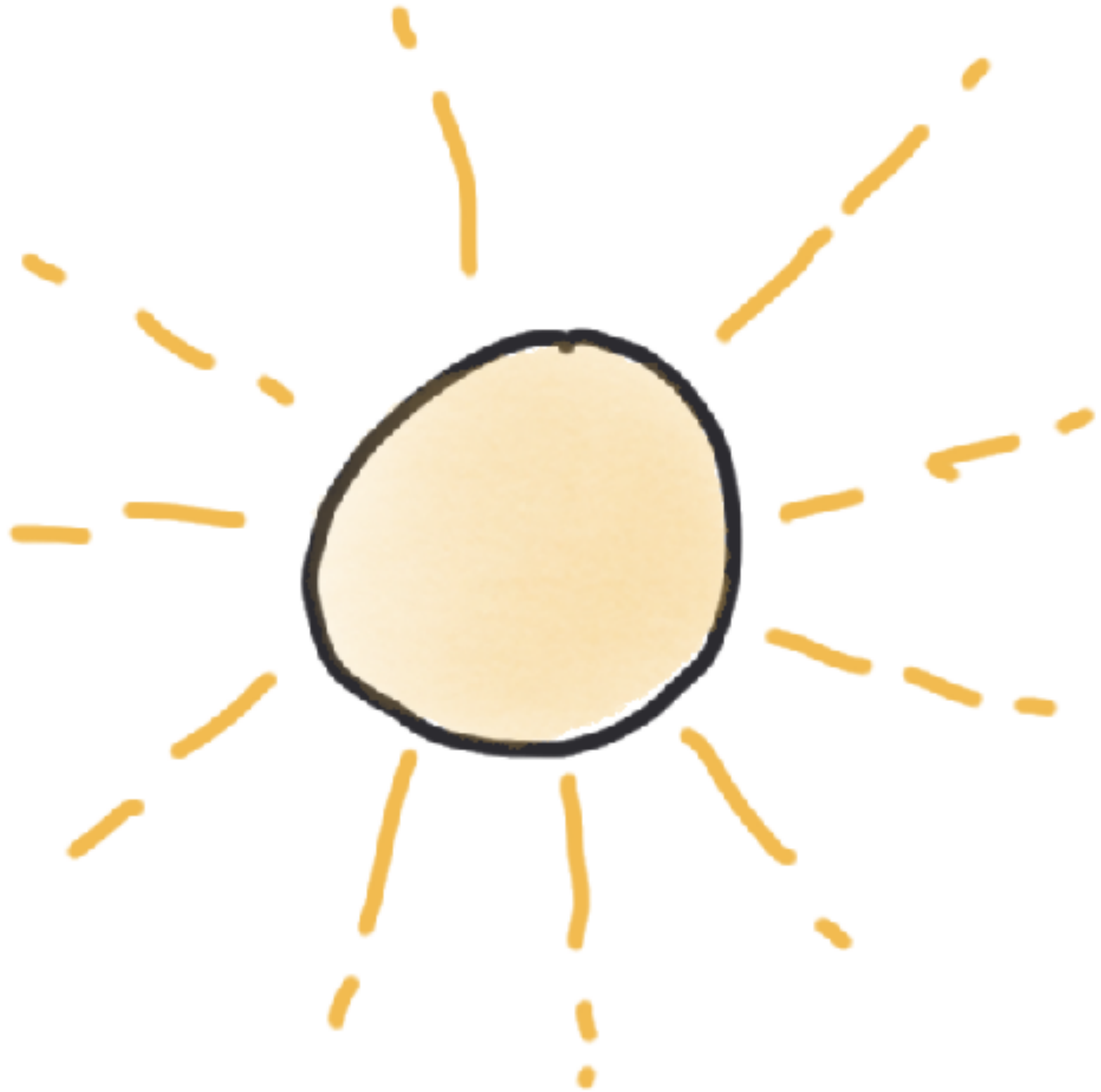
# step 2: convert power usage to carbon



coal

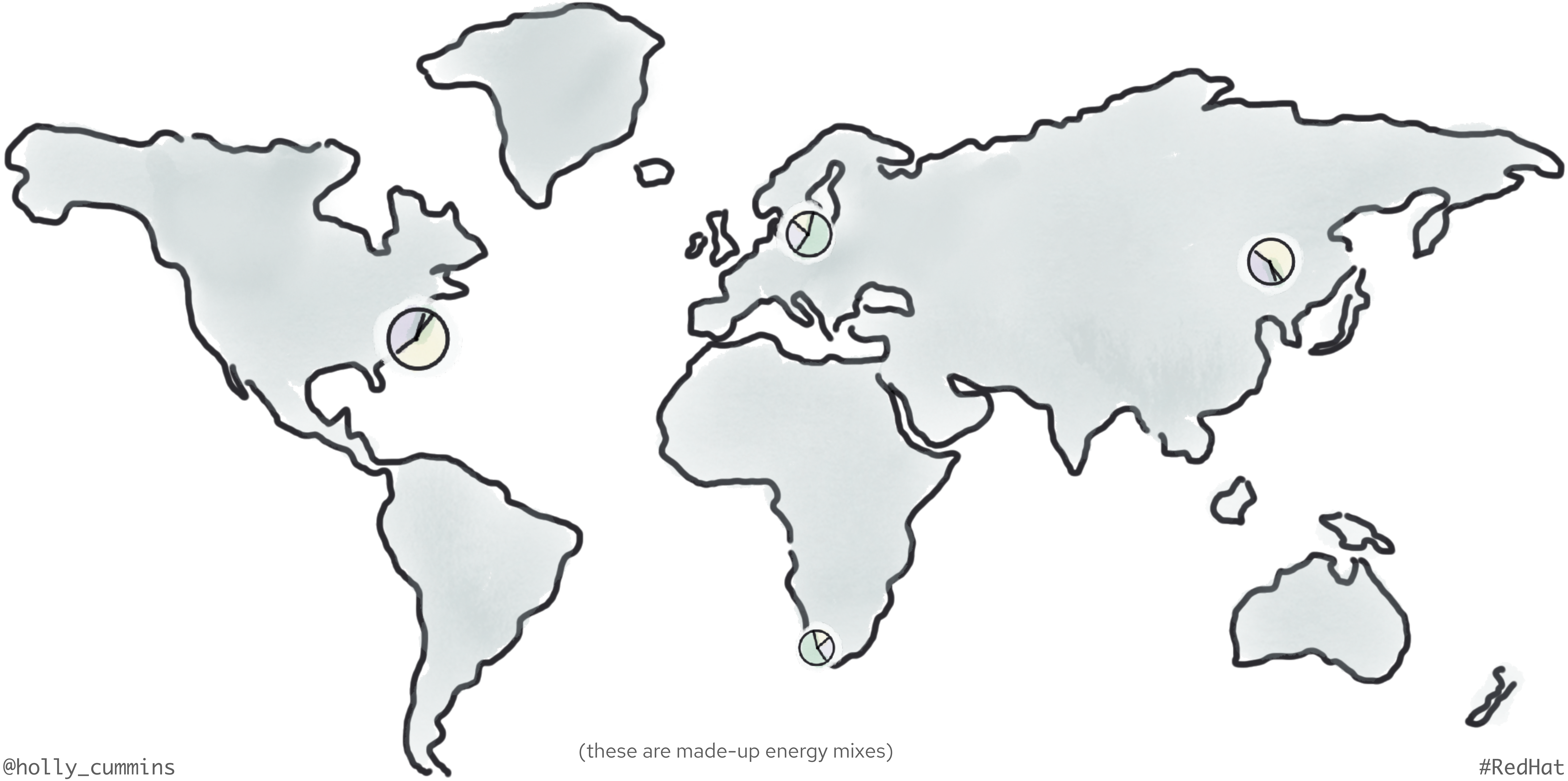


wind



solar

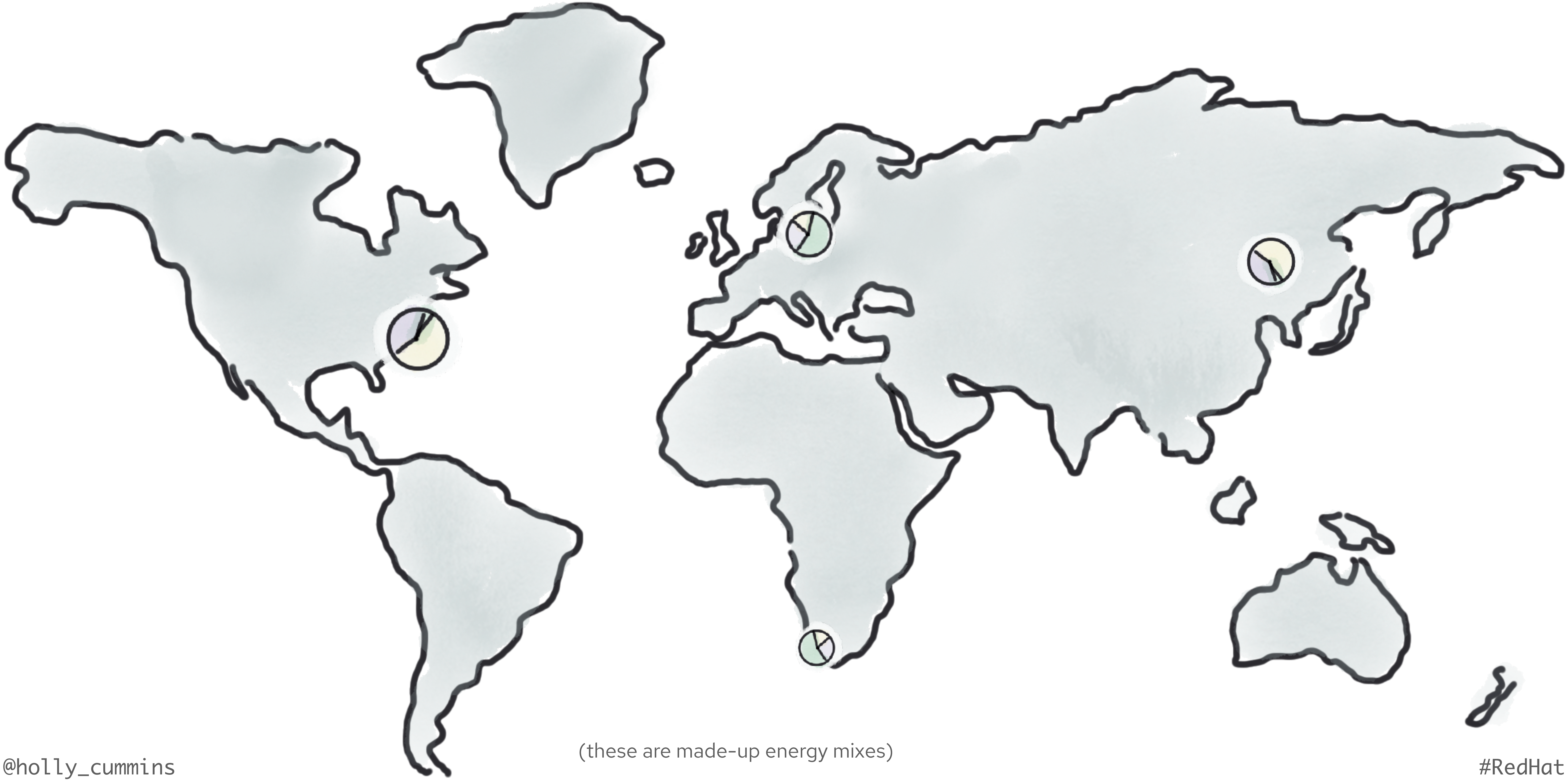
# published energy mixes



(these are made-up energy mixes)

# published energy mixes

... but methodologies are not open

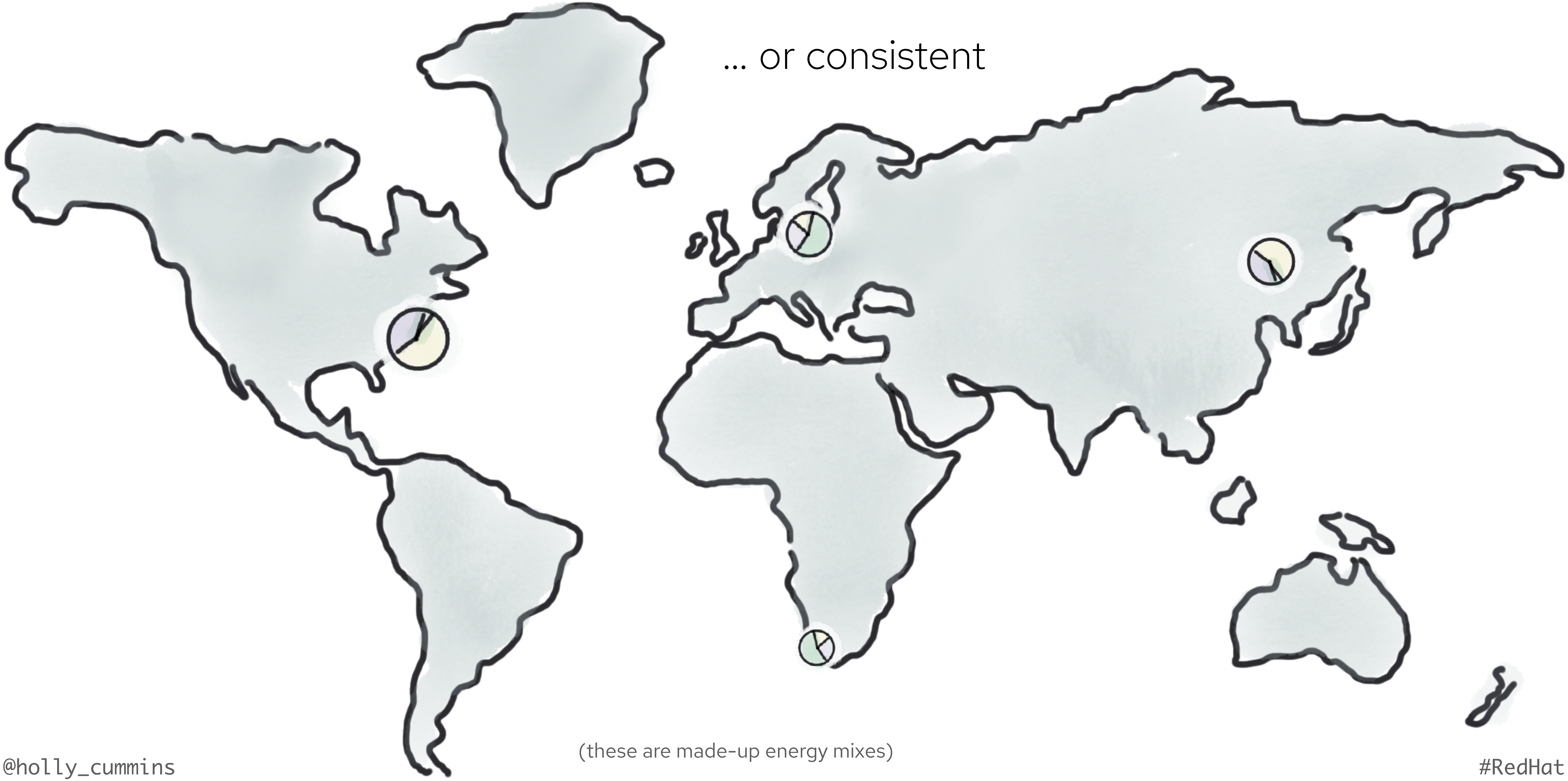


(these are made-up energy mixes)

# published energy mixes

... but methodologies are not open

... or consistent



(these are made-up energy mixes)

# step 3: embedded carbon

(manufacturing has costs)

simpler models

all models are wrong, some are useful

# trick 3: vrrrrrooooooooom model*

* a made-up name

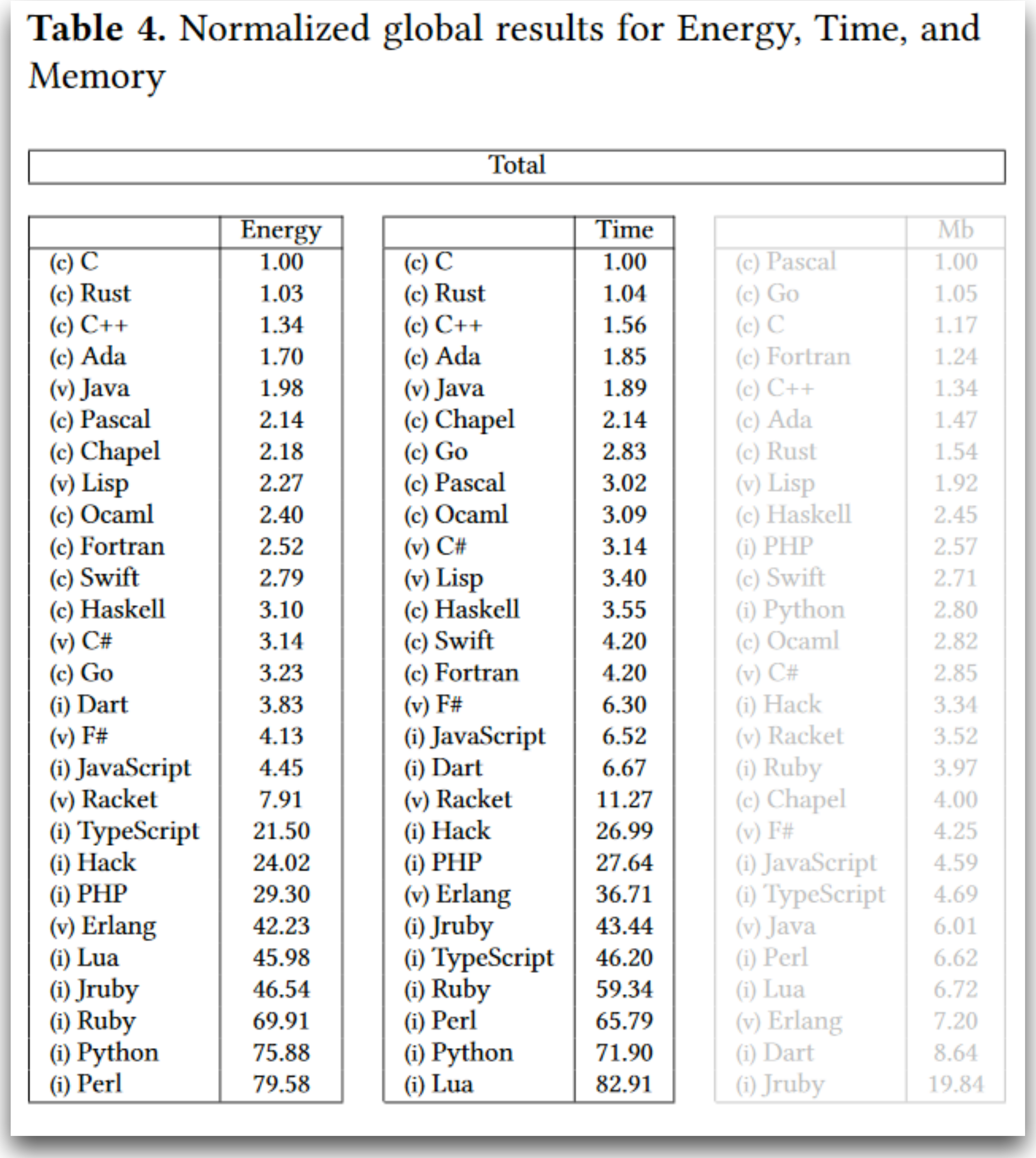


**Table 4.** Normalized global results for Energy, Time, and Memory

Total

|  | Energy |  | Time |  | Mb |
|---|---|---|---|---|---|
| (c) C | 1.00 | (c) C | 1.00 | (c) Pascal | 1.00 |
| (c) Rust | 1.03 | (c) Rust | 1.04 | (c) Go | 1.05 |
| (c) C++ | 1.34 | (c) C++ | 1.56 | (c) C | 1.17 |
| (c) Ada | 1.70 | (c) Ada | 1.85 | (c) Fortran | 1.24 |
| (v) Java | 1.98 | (v) Java | 1.89 | (c) C++ | 1.34 |
| (c) Pascal | 2.14 | (c) Chapel | 2.14 | (c) Ada | 1.47 |
| (c) Chapel | 2.18 | (c) Go | 2.83 | (c) Rust | 1.54 |
| (v) Lisp | 2.27 | (c) Pascal | 3.02 | (v) Lisp | 1.92 |
| (c) Ocaml | 2.40 | (c) Ocaml | 3.09 | (c) Haskell | 2.45 |
| (c) Fortran | 2.52 | (v) C# | 3.14 | (i) PHP | 2.57 |
| (c) Swift | 2.79 | (v) Lisp | 3.40 | (c) Swift | 2.71 |
| (c) Haskell | 3.10 | (c) Haskell | 3.55 | (i) Python | 2.80 |
| (v) C# | 3.14 | (c) Swift | 4.20 | (c) Ocaml | 2.82 |
| (c) Go | 3.23 | (c) Fortran | 4.20 | (v) C# | 2.85 |
| (i) Dart | 3.83 | (v) F# | 6.30 | (i) Hack | 3.34 |
| (v) F# | 4.13 | (i) JavaScript | 6.52 | (v) Racket | 3.52 |
| (i) JavaScript | 4.45 | (i) Dart | 6.67 | (i) Ruby | 3.97 |
| (v) Racket | 7.91 | (v) Racket | 11.27 | (c) Chapel | 4.00 |
| (i) TypeScript | 21.50 | (i) Hack | 26.99 | (v) F# | 4.25 |
| (i) Hack | 24.02 | (i) PHP | 27.64 | (i) JavaScript | 4.59 |
| (i) PHP | 29.30 | (v) Erlang | 36.71 | (i) TypeScript | 4.69 |
| (v) Erlang | 42.23 | (i) Jruby | 43.44 | (v) Java | 6.01 |
| (i) Lua | 45.98 | (i) TypeScript | 46.20 | (i) Perl | 6.62 |
| (i) Jruby | 46.54 | (i) Ruby | 59.34 | (i) Lua | 6.72 |
| (i) Ruby | 69.91 | (i) Perl | 65.79 | (v) Erlang | 7.20 |
| (i) Python | 75.88 | (i) Python | 71.90 | (i) Dart | 8.64 |
| (i) Perl | 79.58 | (i) Lua | 82.91 | (i) Jruby | 19.84 |

these two columns
are **almost** the same

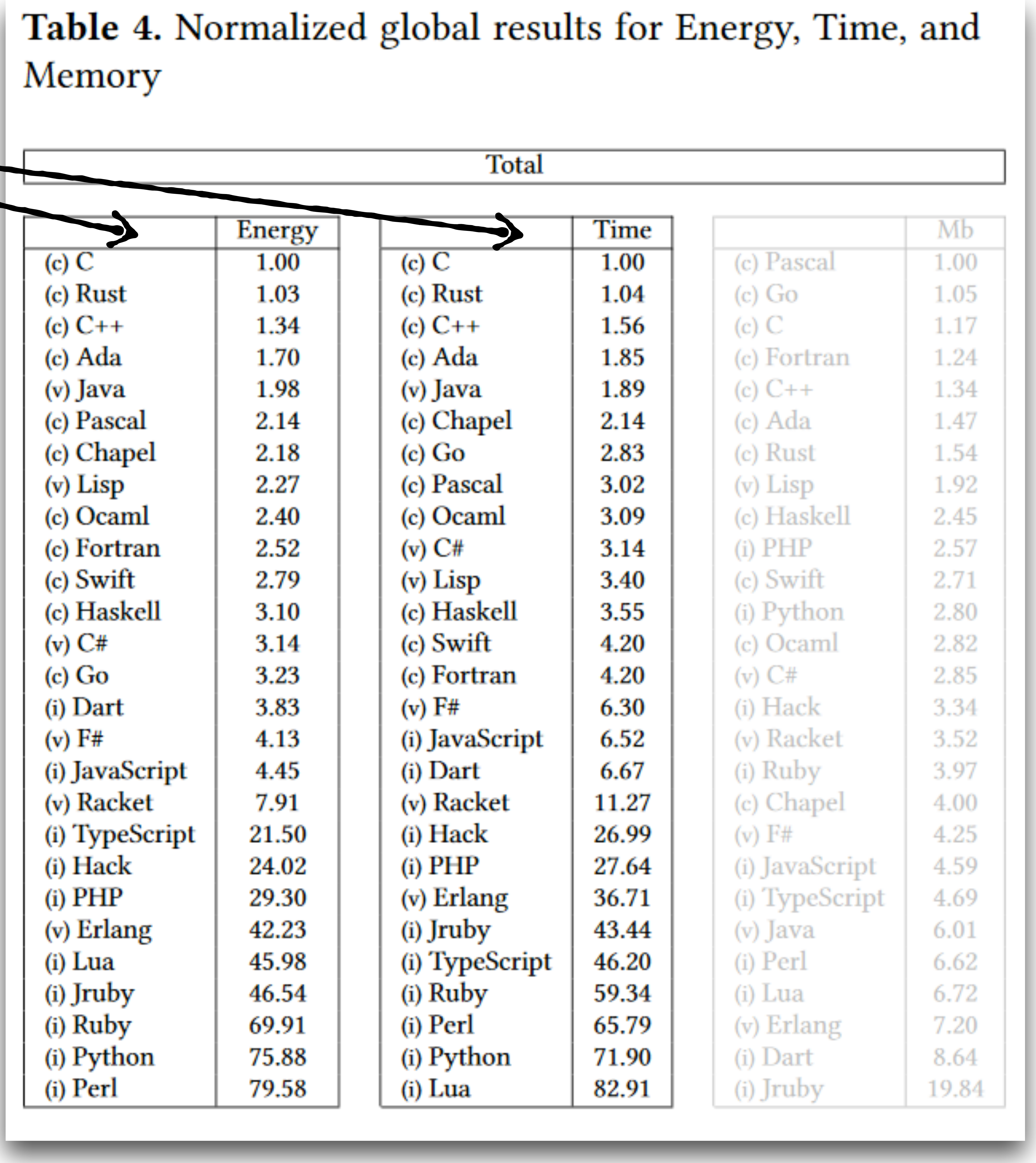


**Table 4.** Normalized global results for Energy, Time, and Memory

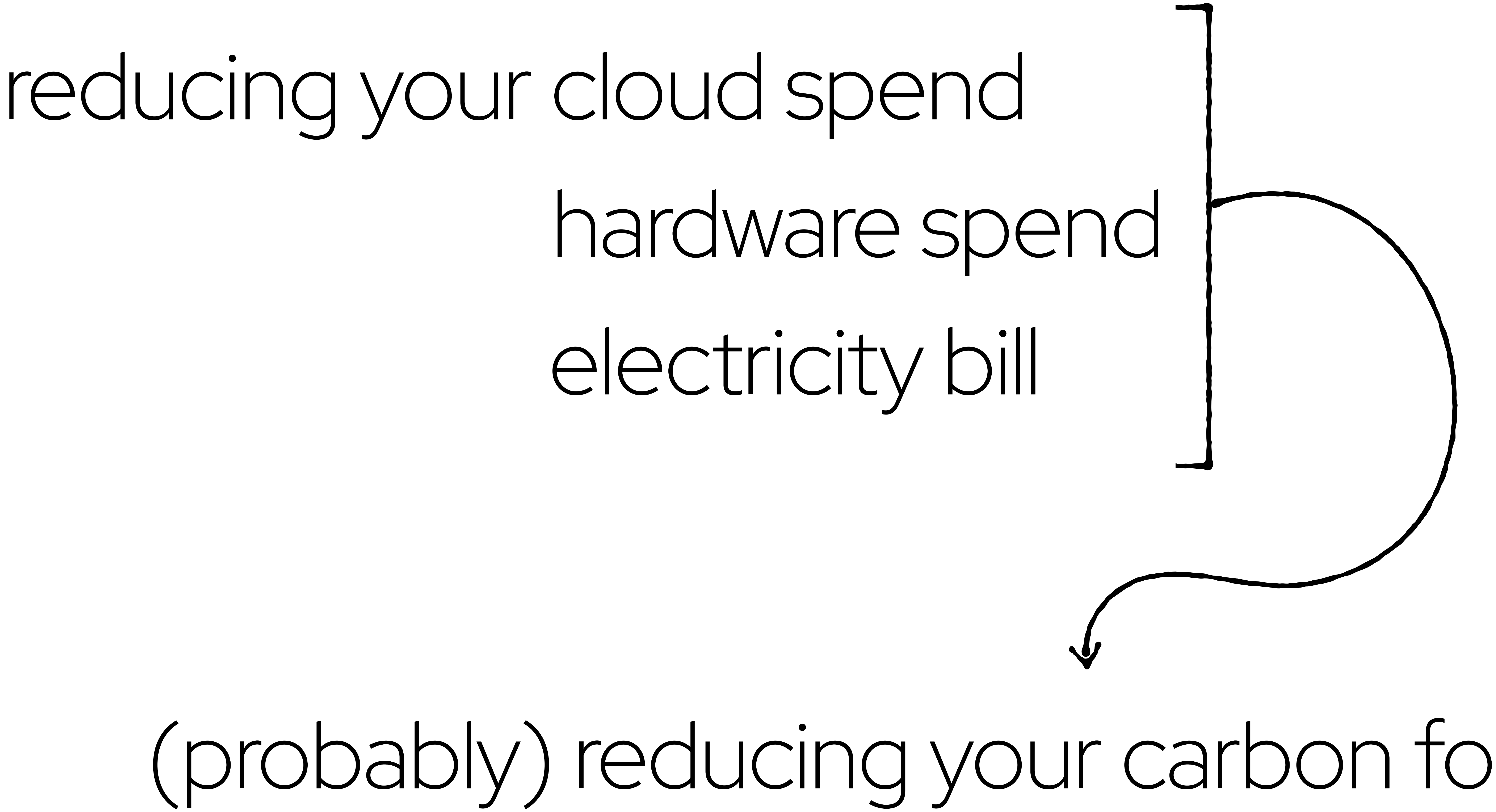
| Total | | | | | |
|---|---|---|---|---|---|
| | Energy | | Time | | Mb |
| (c) C | 1.00 | (c) C | 1.00 | (c) Pascal | 1.00 |
| (c) Rust | 1.03 | (c) Rust | 1.04 | (c) Go | 1.05 |
| (c) C++ | 1.34 | (c) C++ | 1.56 | (c) C | 1.17 |
| (c) Ada | 1.70 | (c) Ada | 1.85 | (c) Fortran | 1.24 |
| (v) Java | 1.98 | (v) Java | 1.89 | (c) C++ | 1.34 |
| (c) Pascal | 2.14 | (c) Chapel | 2.14 | (c) Ada | 1.47 |
| (c) Chapel | 2.18 | (c) Go | 2.83 | (c) Rust | 1.54 |
| (v) Lisp | 2.27 | (c) Pascal | 3.02 | (v) Lisp | 1.92 |
| (c) Ocaml | 2.40 | (c) Ocaml | 3.09 | (c) Haskell | 2.45 |
| (c) Fortran | 2.52 | (v) C# | 3.14 | (i) PHP | 2.57 |
| (c) Swift | 2.79 | (v) Lisp | 3.40 | (c) Swift | 2.71 |
| (c) Haskell | 3.10 | (c) Haskell | 3.55 | (i) Python | 2.80 |
| (v) C# | 3.14 | (c) Swift | 4.20 | (c) Ocaml | 2.82 |
| (c) Go | 3.23 | (c) Fortran | 4.20 | (v) C# | 2.85 |
| (i) Dart | 3.83 | (v) F# | 6.30 | (i) Hack | 3.34 |
| (v) F# | 4.13 | (i) JavaScript | 6.52 | (v) Racket | 3.52 |
| (i) JavaScript | 4.45 | (i) Dart | 6.67 | (i) Ruby | 3.97 |
| (v) Racket | 7.91 | (v) Racket | 11.27 | (c) Chapel | 4.00 |
| (i) TypeScript | 21.50 | (i) Hack | 26.99 | (v) F# | 4.25 |
| (i) Hack | 24.02 | (i) PHP | 27.64 | (i) JavaScript | 4.59 |
| (i) PHP | 29.30 | (v) Erlang | 36.71 | (i) TypeScript | 4.69 |
| (v) Erlang | 42.23 | (i) Jruby | 43.44 | (v) Java | 6.01 |
| (i) Lua | 45.98 | (i) TypeScript | 46.20 | (i) Perl | 6.62 |
| (i) Jruby | 46.54 | (i) Ruby | 59.34 | (i) Lua | 6.72 |
| (i) Ruby | 69.91 | (i) Perl | 65.79 | (v) Erlang | 7.20 |
| (i) Python | 75.88 | (i) Python | 71.90 | (i) Dart | 8.64 |
| (i) Perl | 79.58 | (i) Lua | 82.91 | (i) Jruby | 19.84 |

energy consumption (sort of, mostly)
is proportional to execution time

# trick 4: economic model*

# trick 4: economic model*

* "economic input-output life cycle assessment"



reducing your cloud spend

hardware spend

electricity bill

(probably) reducing your carbon footprint*

* if you keep other factors the same

# worked(ish) example:
# what’s the carbon footprint of ChatGPT?

# worked(ish) example:
# what’s the carbon footprint of ChatGPT?

worked(ish) example:
what's the carbon footprint of ChatGPT?

$50,000-

$700,000

running costs per day

https://www.businessinsider.com/how-much-chatgpt-costs-openai-to-run-estimate-report-2023-4

worked(ish) example:
what's the carbon footprint of ChatGPT?

$50,000-
$700,000

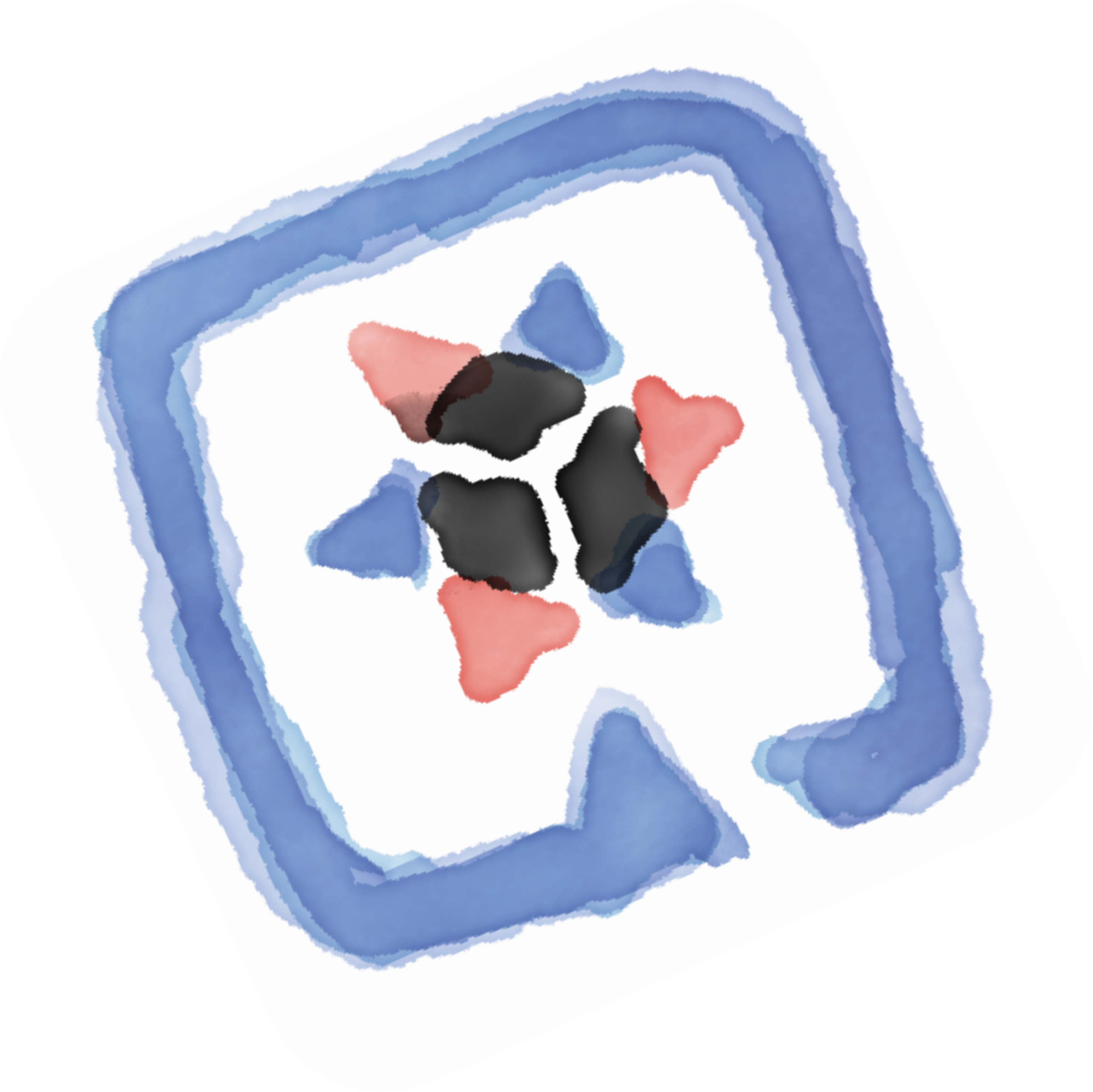
running costs per day

https://www.businessinsider.com/how-much-chatgpt-costs-openai-to-run-estimate-report-2023-4

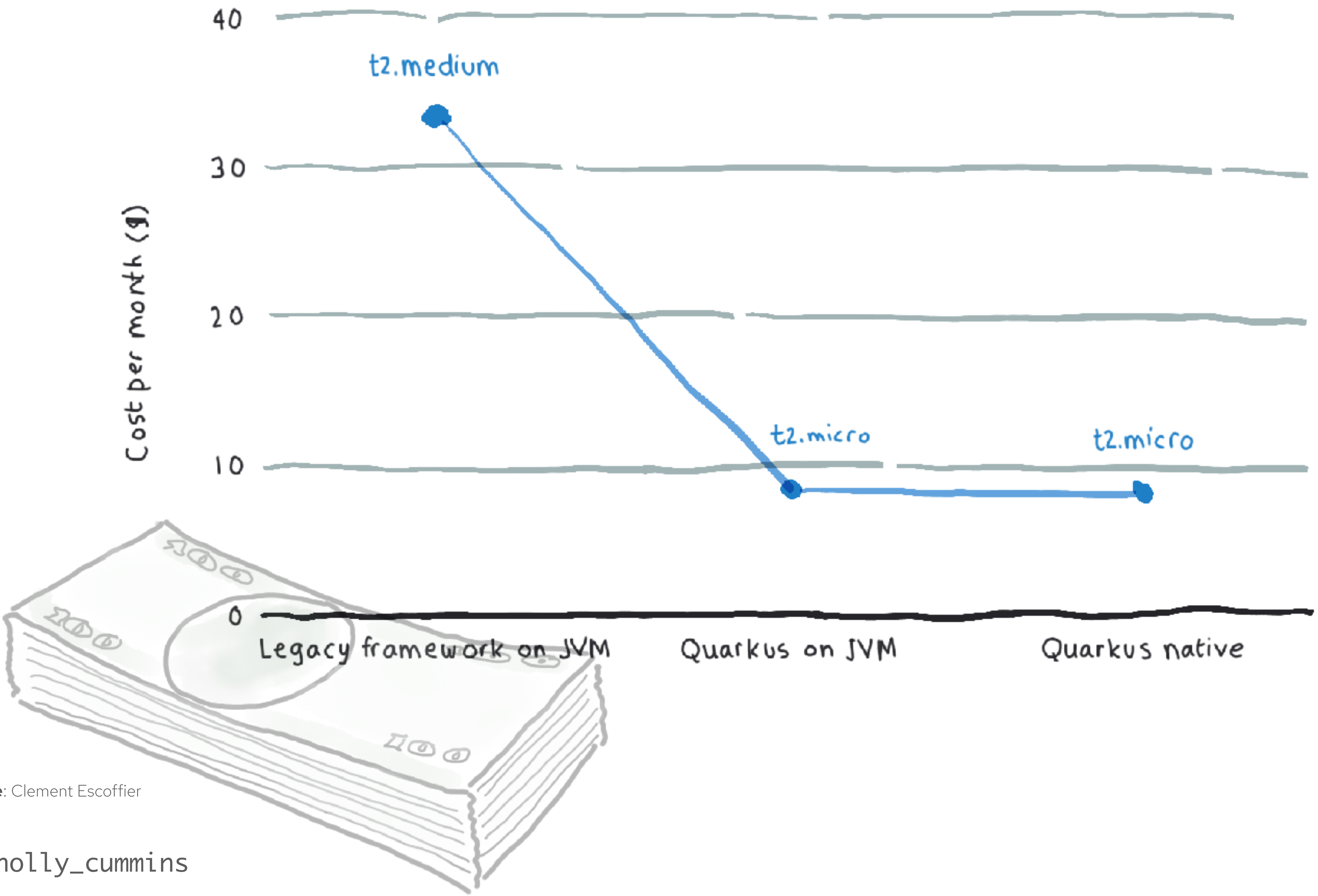
$3,000,000-
$50,000,000

training costs

so … quarkus?



# cost impact of framework choice



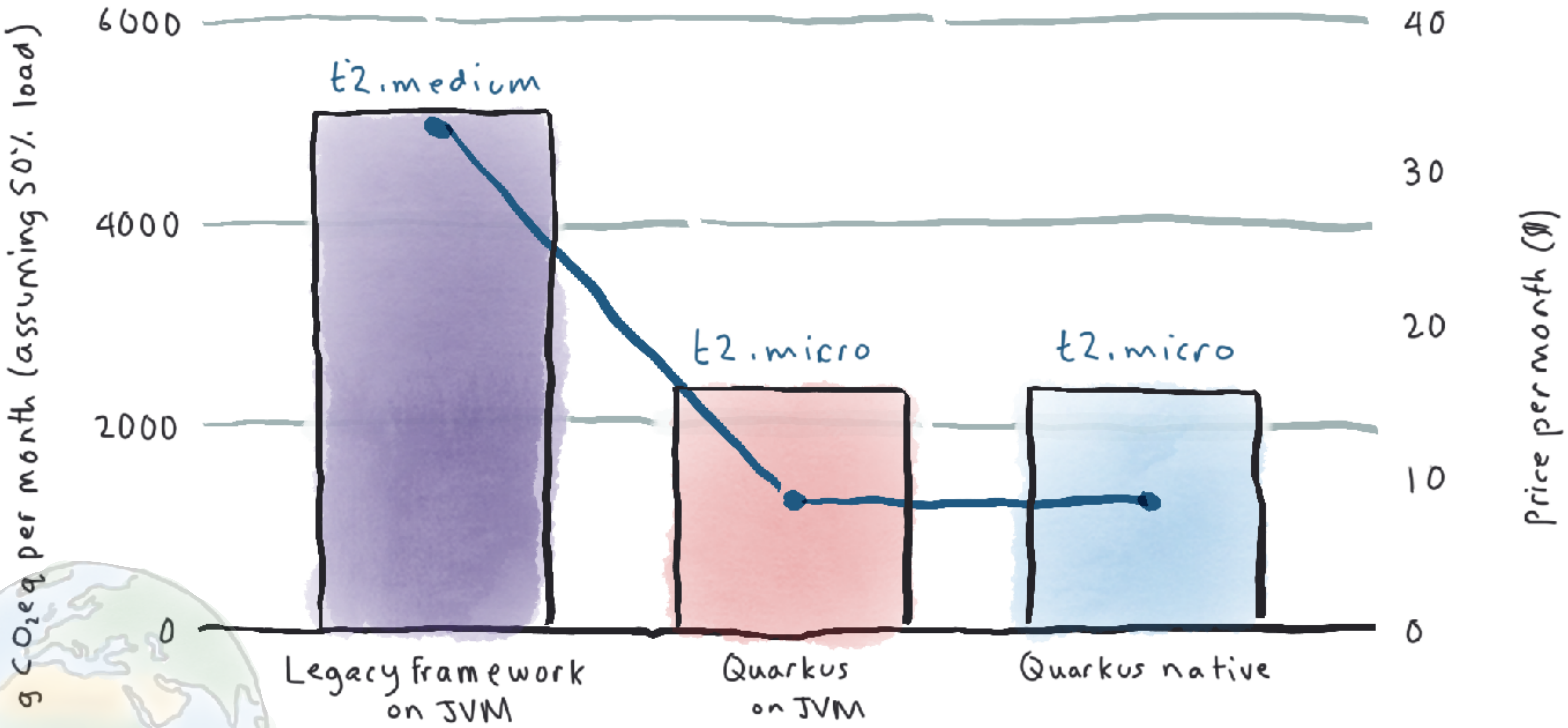
**Setup:**

- 800 requests/second, over 20 days
- SLA > 99%
- AWS instances

**Assumptions:**

- Costs are for us-east-1 data centre

**Source**: Clement Escoffier



Source: Clement Escoffier x Teads

**Setup:**

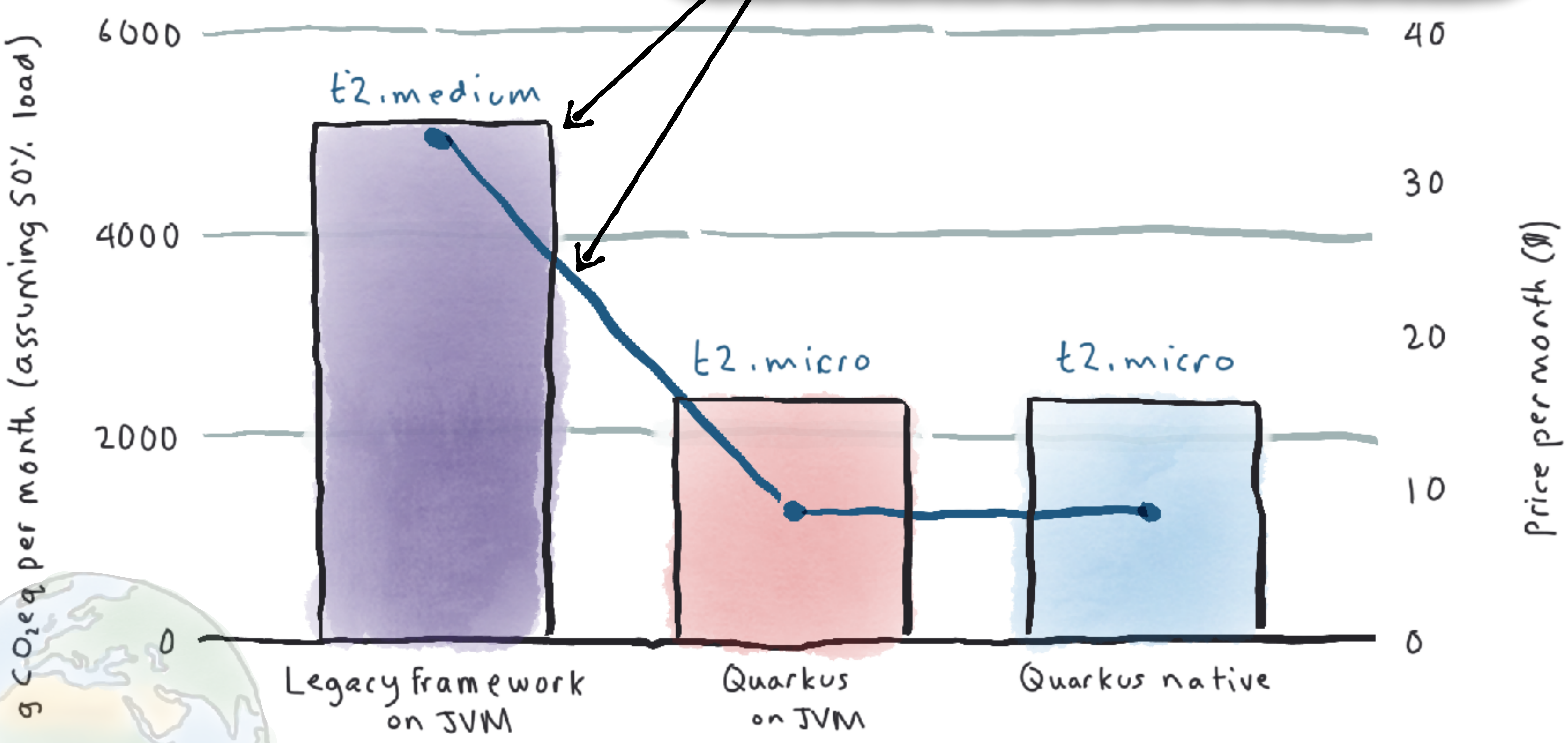
- 800 requests/second, over 20 days
- SLA > 99%

**Assumptions:**

- 50% load
- us-east-1 data centre
- Teads dataset

**economic model in action:**
the *cost* and *carbon* metrics are (roughly) the same

goto; aarhus



Source: Clement Escoffier x Teads

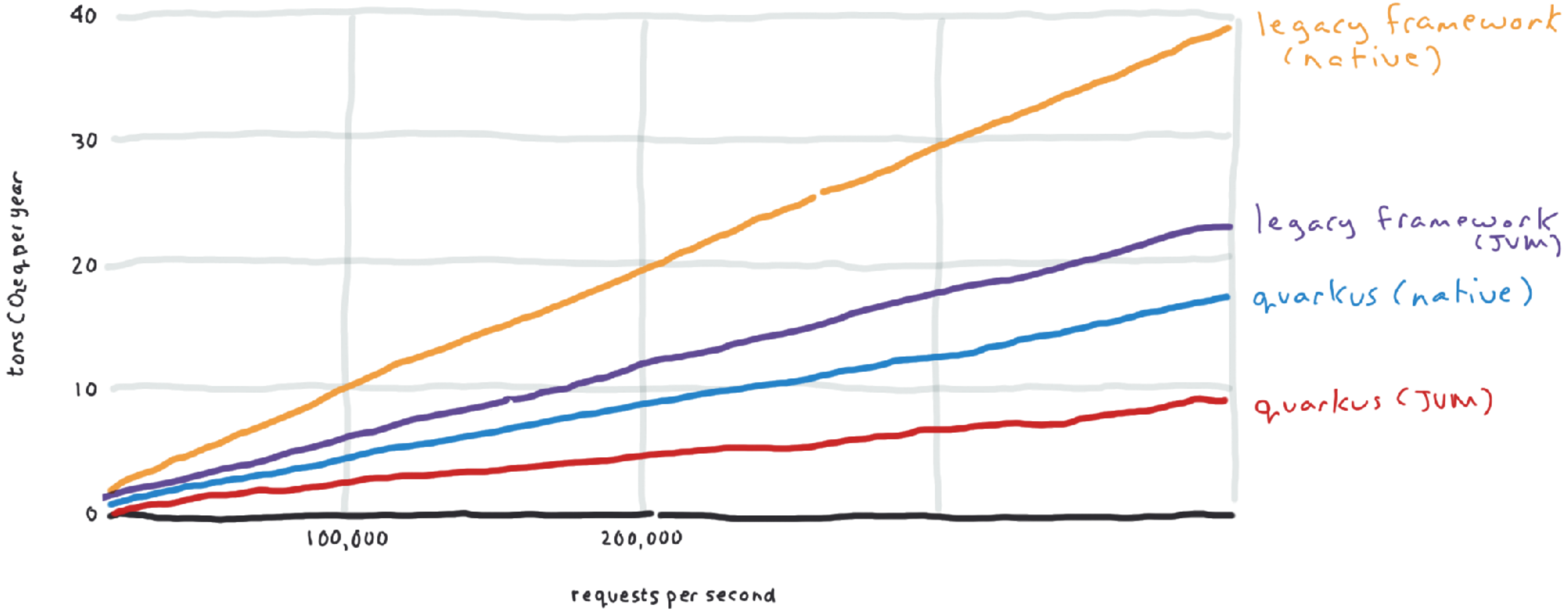
**Setup:**

- 800 requests/second, over 20 days
- SLA > 99%

**Assumptions:**

- 50% load
- us-east-1 data centre
- Teads dataset

# climate impact as a function of load



**Setup:**

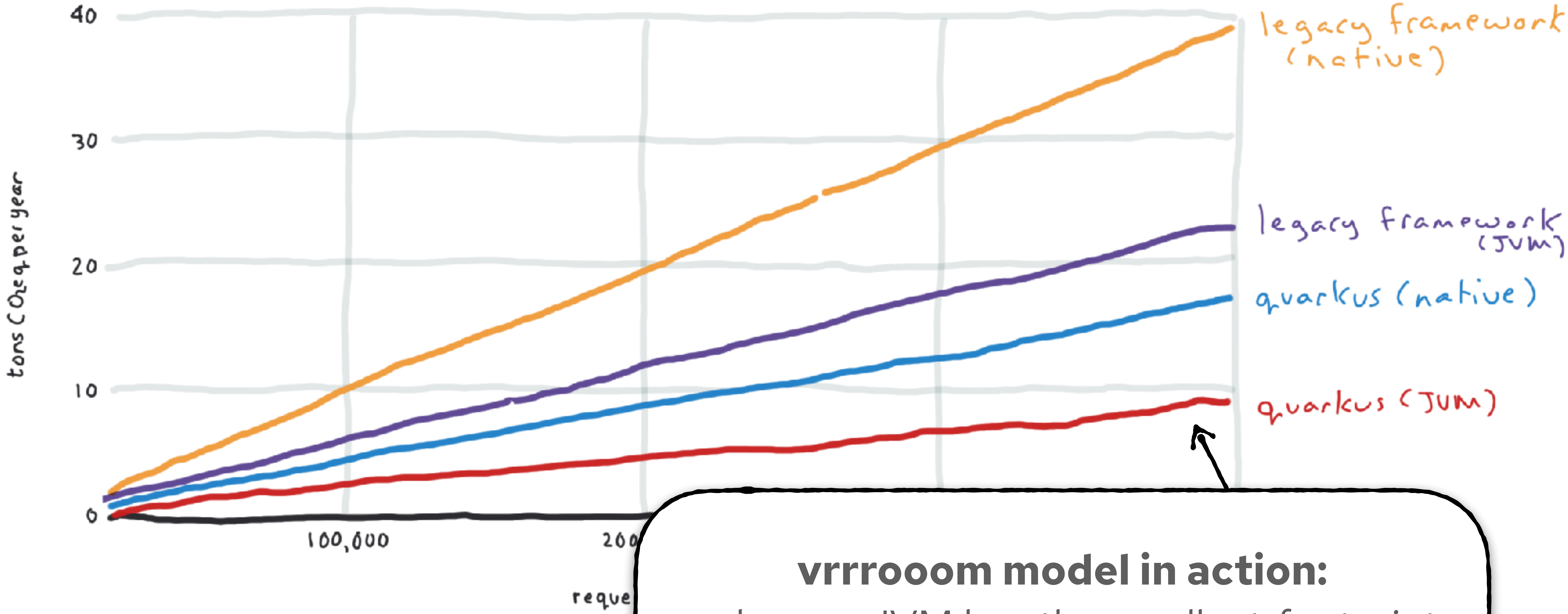
- REST + CRUD
- large heap
- RAPL energy measurement
- multiple instances to support high load

**Assumptions:**

- US energy mix

**Source**: John O'Hara

# climate impact as a function of load



**vrrrooom model in action:**
quarkus on JVM has the smallest *footprint* …
because it has the highest *throughput*

**Setup:**
- REST + CRUD
- large heap
- RAPL energy measurem
- multiple instances to support high load

**Source**: John O'Hara

@holly_cummins

#RedHat

# what about memory?

**more complete model:**

both throughput (execution time) and memory contribute to carbon footprint

# what about memory?

**more complete model:**

both throughput (execution time) and memory contribute to carbon footprint

quarkus is most efficient for

- startup time
- throughput
- memory (RSS size)

# trick 5: use quarkus

# trick 5: use quarkus

quarkus ‘automatically’ saves

# trick 5: use quarkus

quarkus ‘automatically’ saves

- time

# trick 5: use quarkus

quarkus ‘automatically’ saves

- time
- money

# trick 5: use quarkus

quarkus ‘automatically’ saves

- time
- money
- carbon (~2x)

# trick 5: use quarkus

quarkus ‘automatically’ saves

- time
- money
- carbon (~2x)
- … even when Spring compatibility libraries are used (almost no code changes except dependencies and tests)

# trick 5: use quarkus

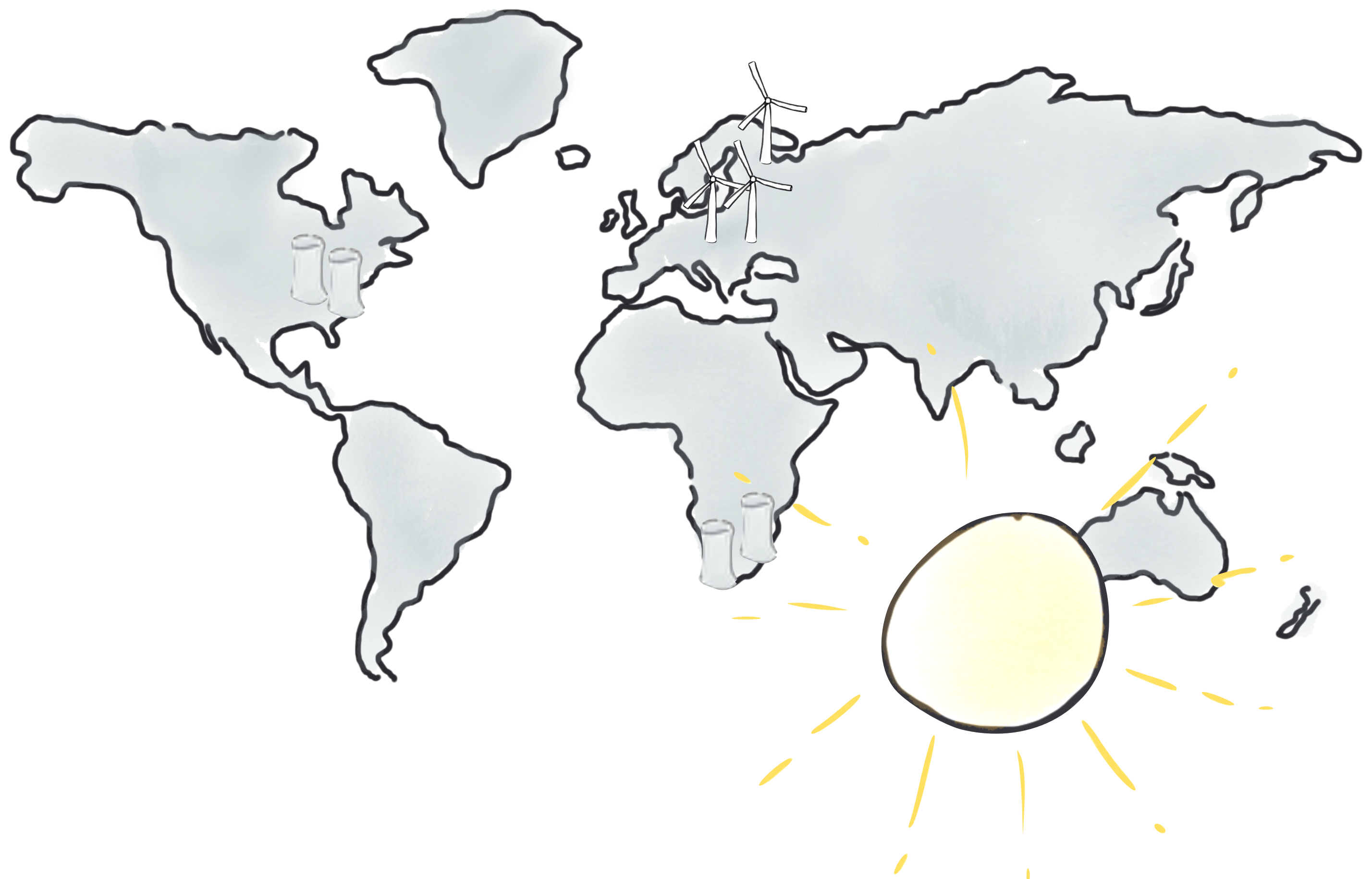
quarkus ‘automatically’ saves

- time
- money
- carbon (~2x)
- … even when Spring compatibility libraries are used (almost no code changes except dependencies and tests)

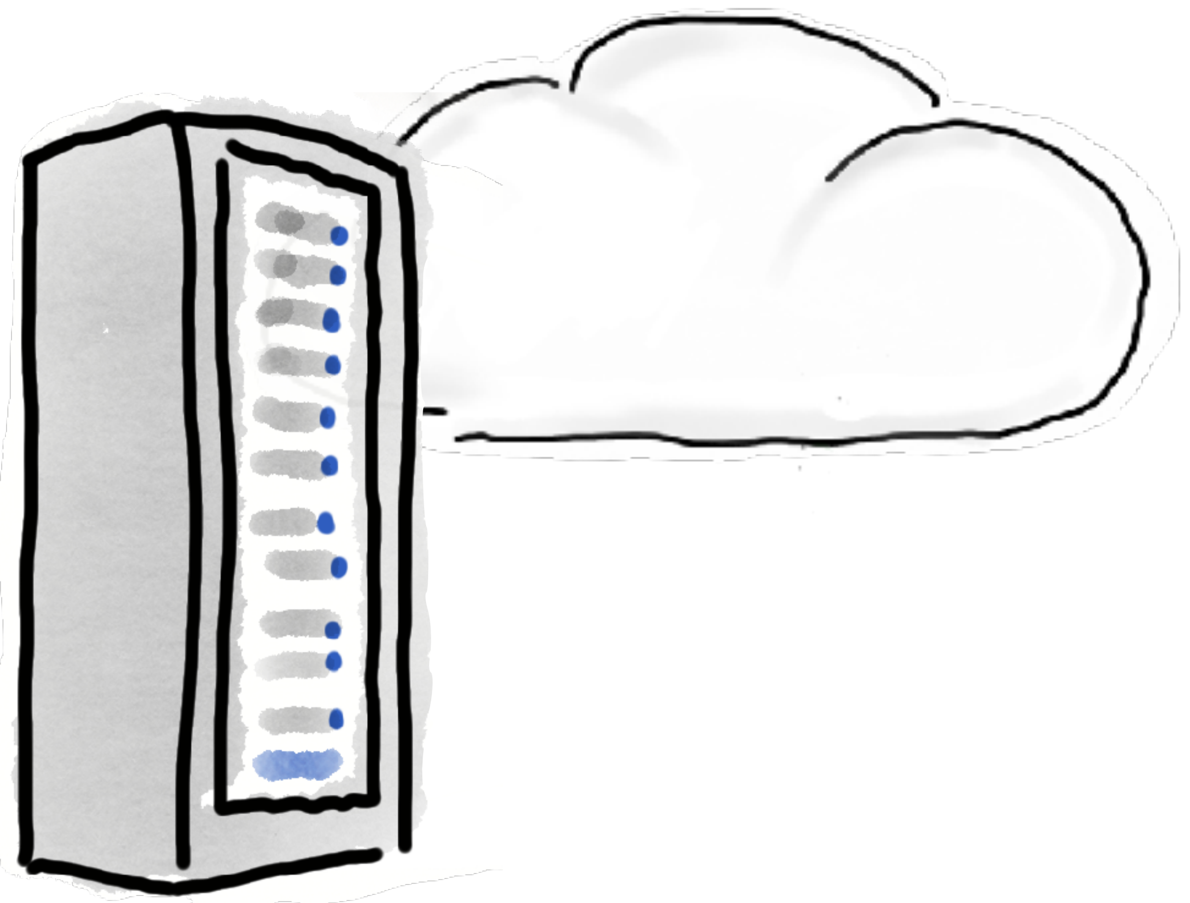
# aaaaaaaaargh?

# green software foundation: principles

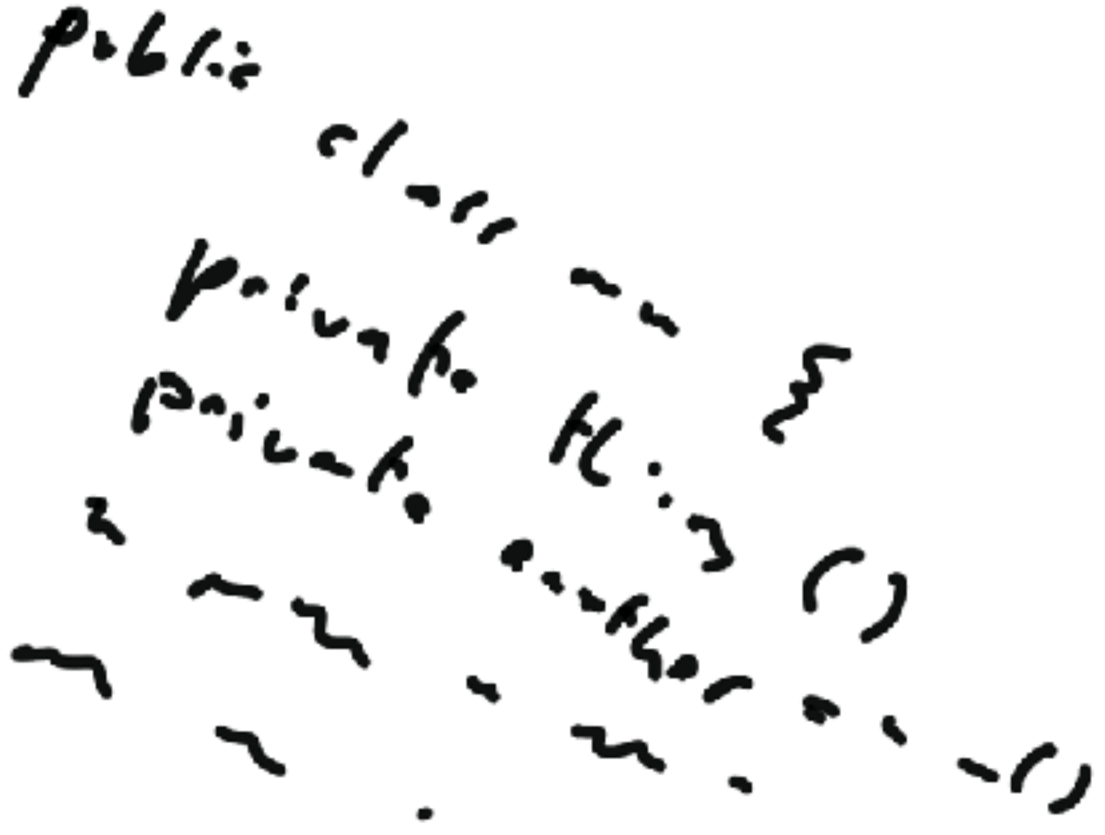
carbon awareness



hardware efficiency



electricity efficiency



utility

# utility 🤩

is using this carbon giving **value**?

# “no-regrets” solutions

# co-benefits

co-benefits

the double win

co-benefits

the double win

win-win

co-benefits

the double win

win-win

1 + 1 = 3

co-benefits

the double win

win-win

1 + 1 = 3

twofer

co-benefits

the double win

win-win

1 + 1 = 3

twofer

überwinden

co-benefits

the double win

win-win

1 + 1 = 3

twofer

überwinden

climate solutions can
make **everything** better

# remember the zombie servers?

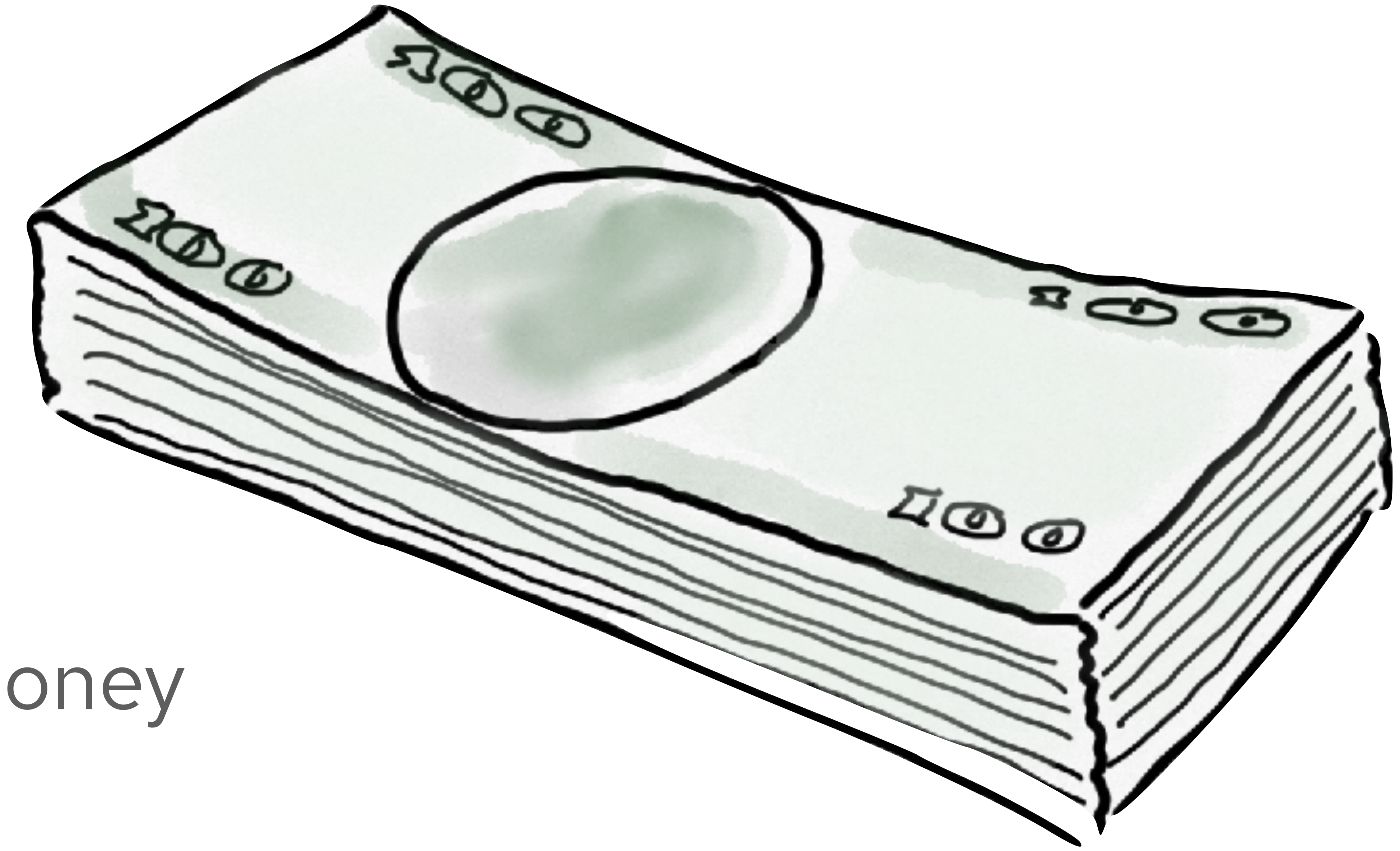
https://www.business2community.com/cloud-computing/overprovisioning-always-on-resources-lead-to-26-6-billion-in-public-cloud-waste-expected-in-2021-02381033

# remember the zombie servers?

what else could we do
with that $26.6 billion
of wasted cloud spend?

https://www.business2community.com/cloud-computing/overprovisioning-always-on-resources-lead-to-26-6-billion-in-public-cloud-waste-expected-in-2021-02381033

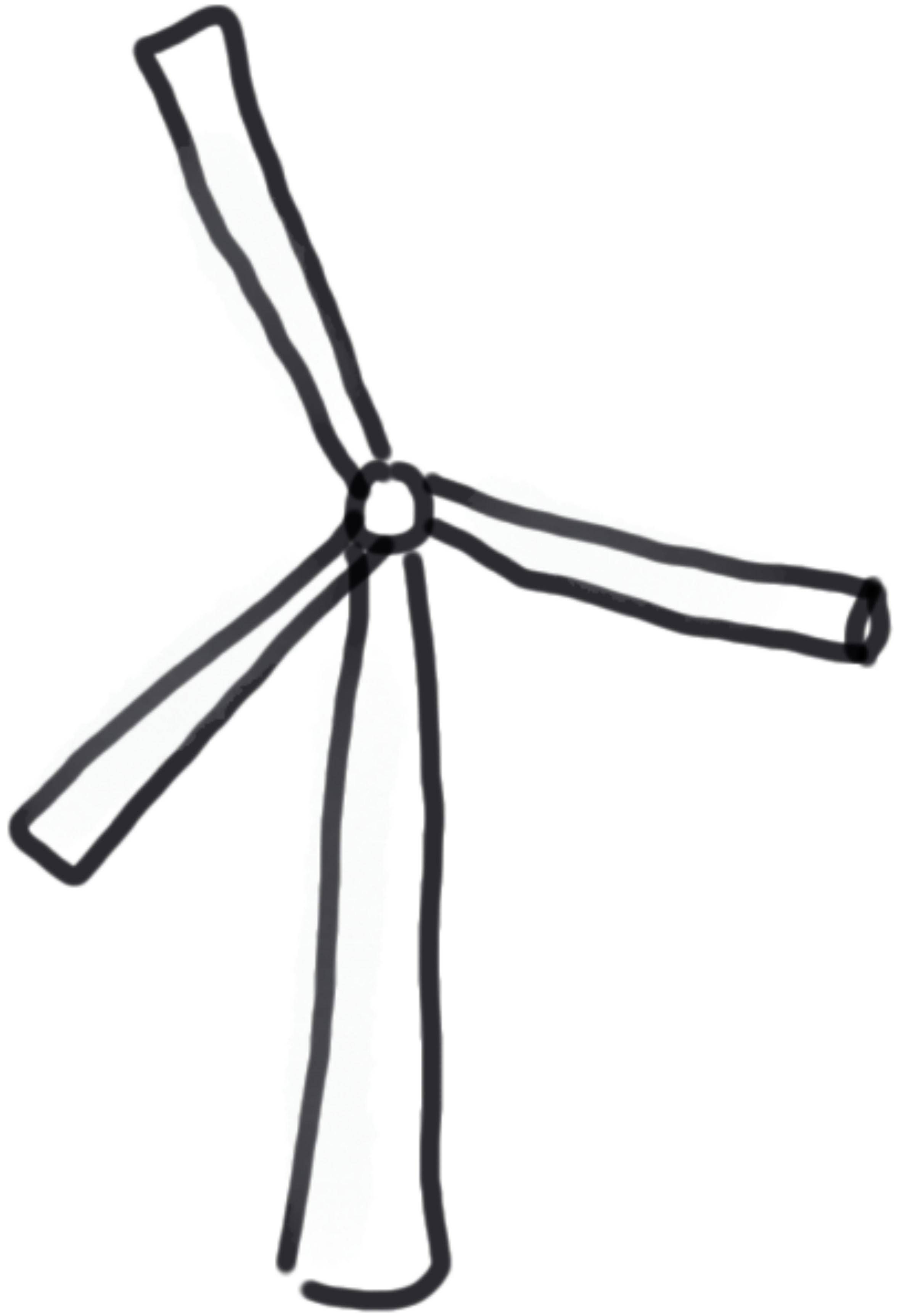
# the double-win



turning things off saves a lot of money

# the double-win

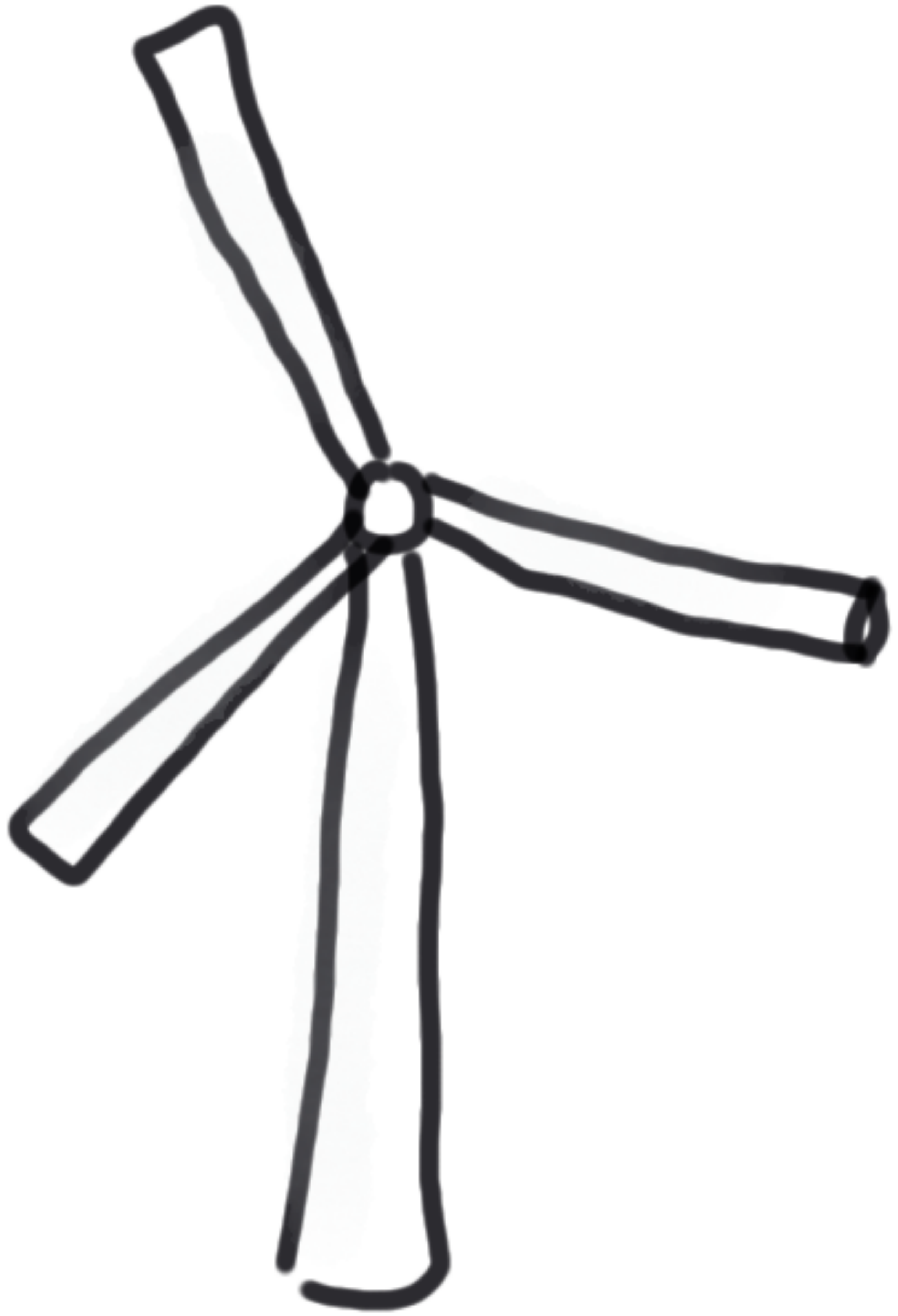
renewable electricity is 9x cheaper



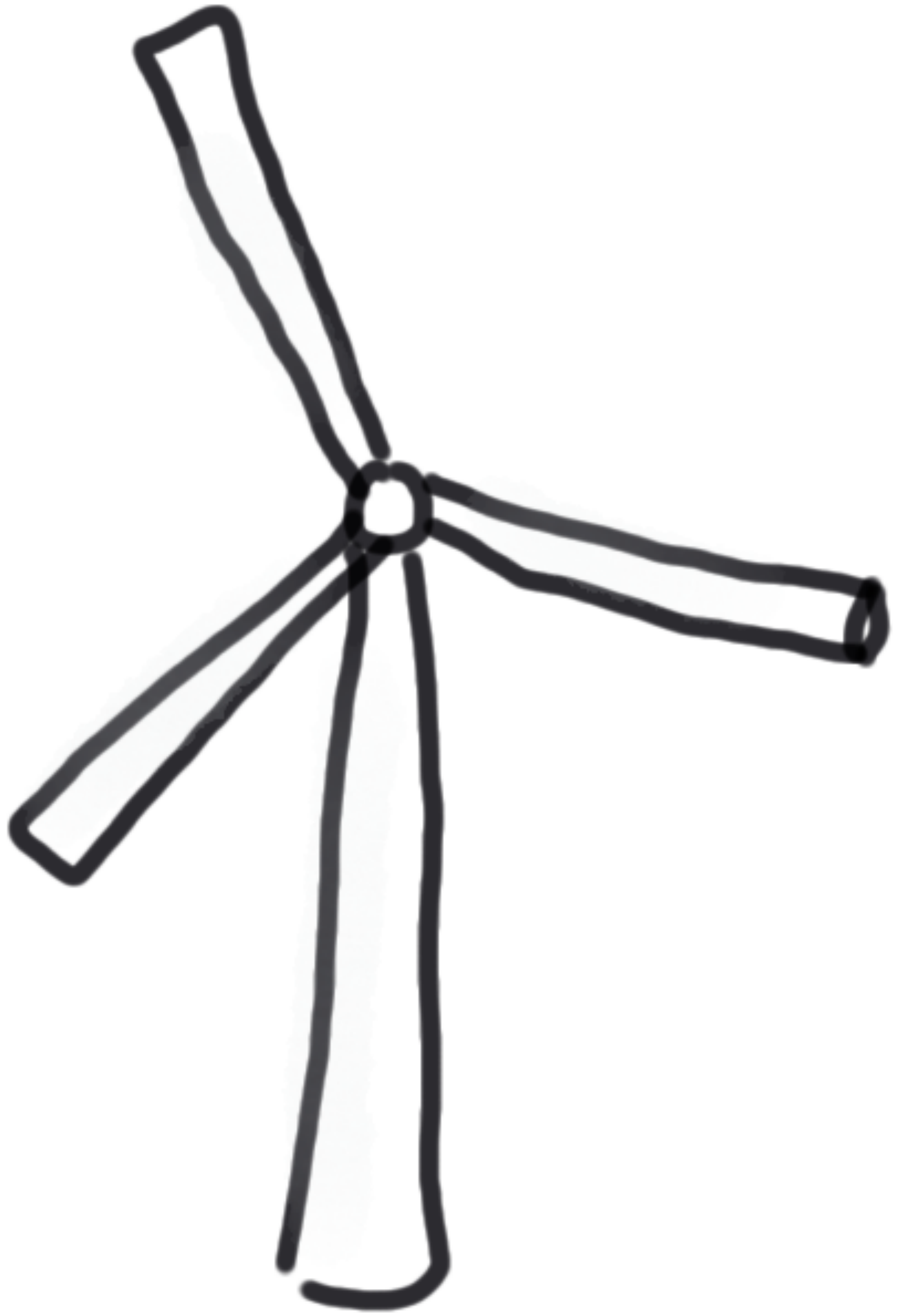
# the double-win

renewable electricity is 9x cheaper

hosting in Montreal:



# the double-win

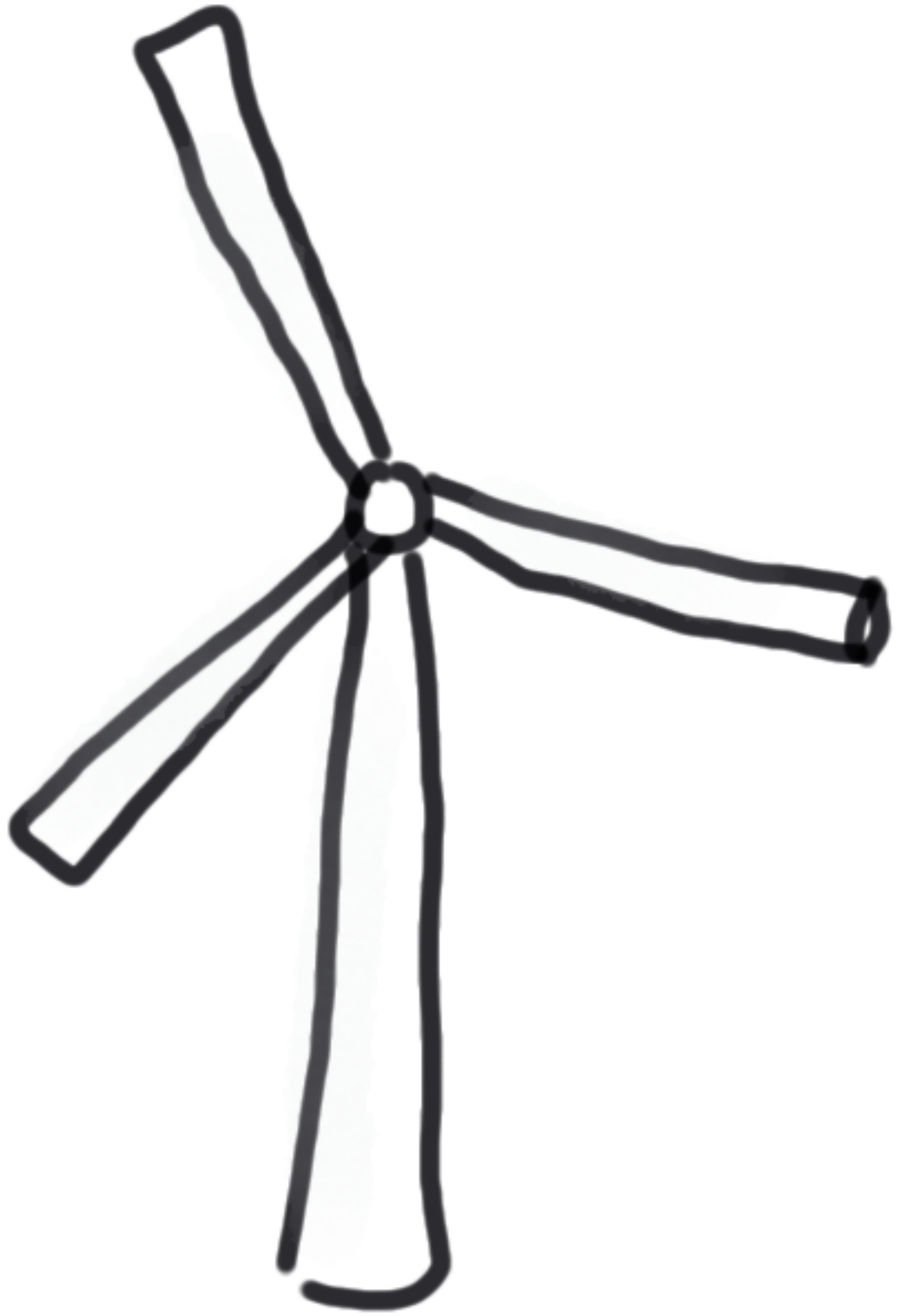


renewable electricity is 9x cheaper

hosting in Montreal:

88% less carbon than the same workload in London

# the double-win



renewable electricity is 9x cheaper

hosting in Montreal:

88% less carbon than the same workload in London

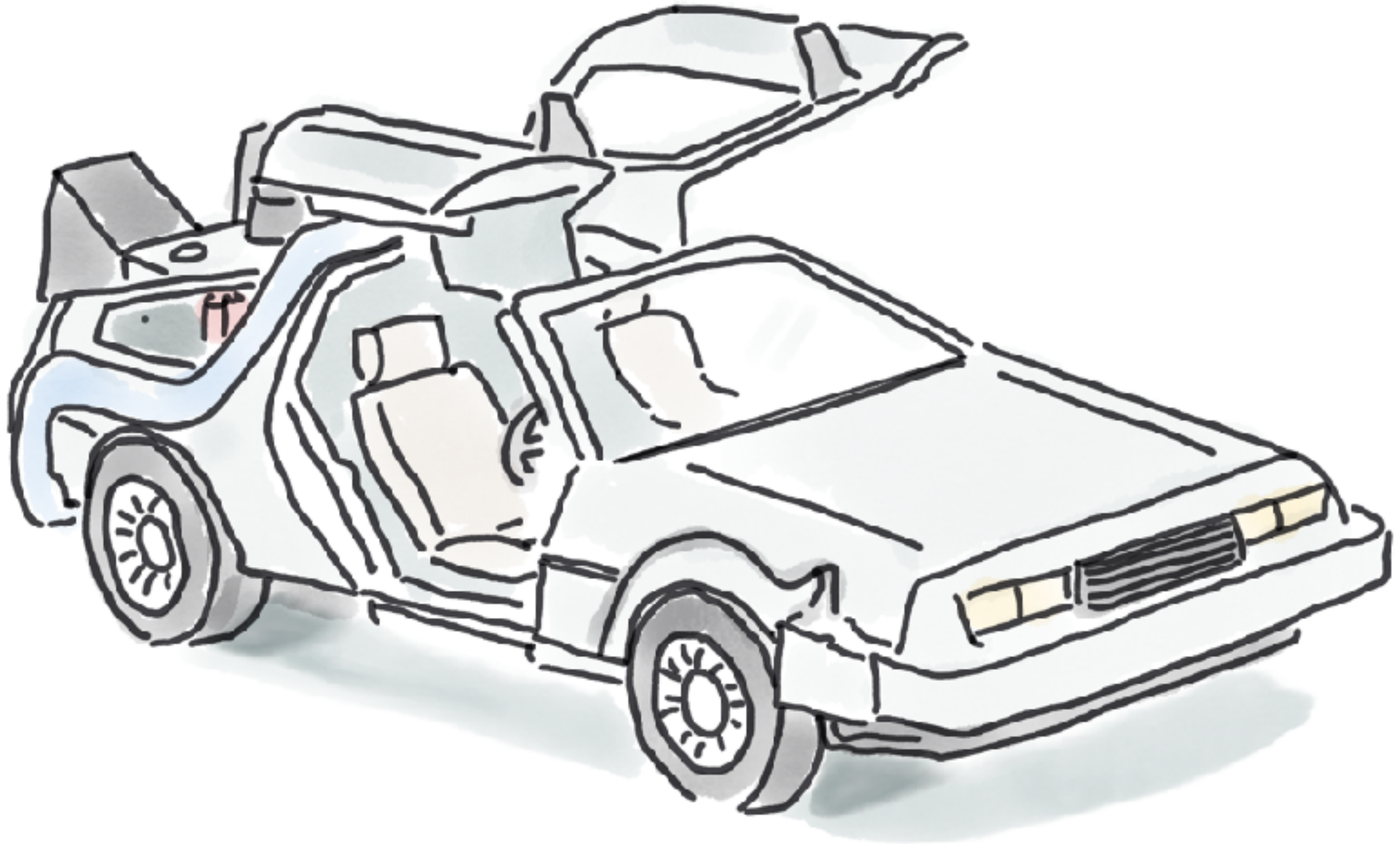
**and** it's 15% cheaper

# remember the vrrrrrooooooooom model?

(probably not, it was a made-up name)

# car:

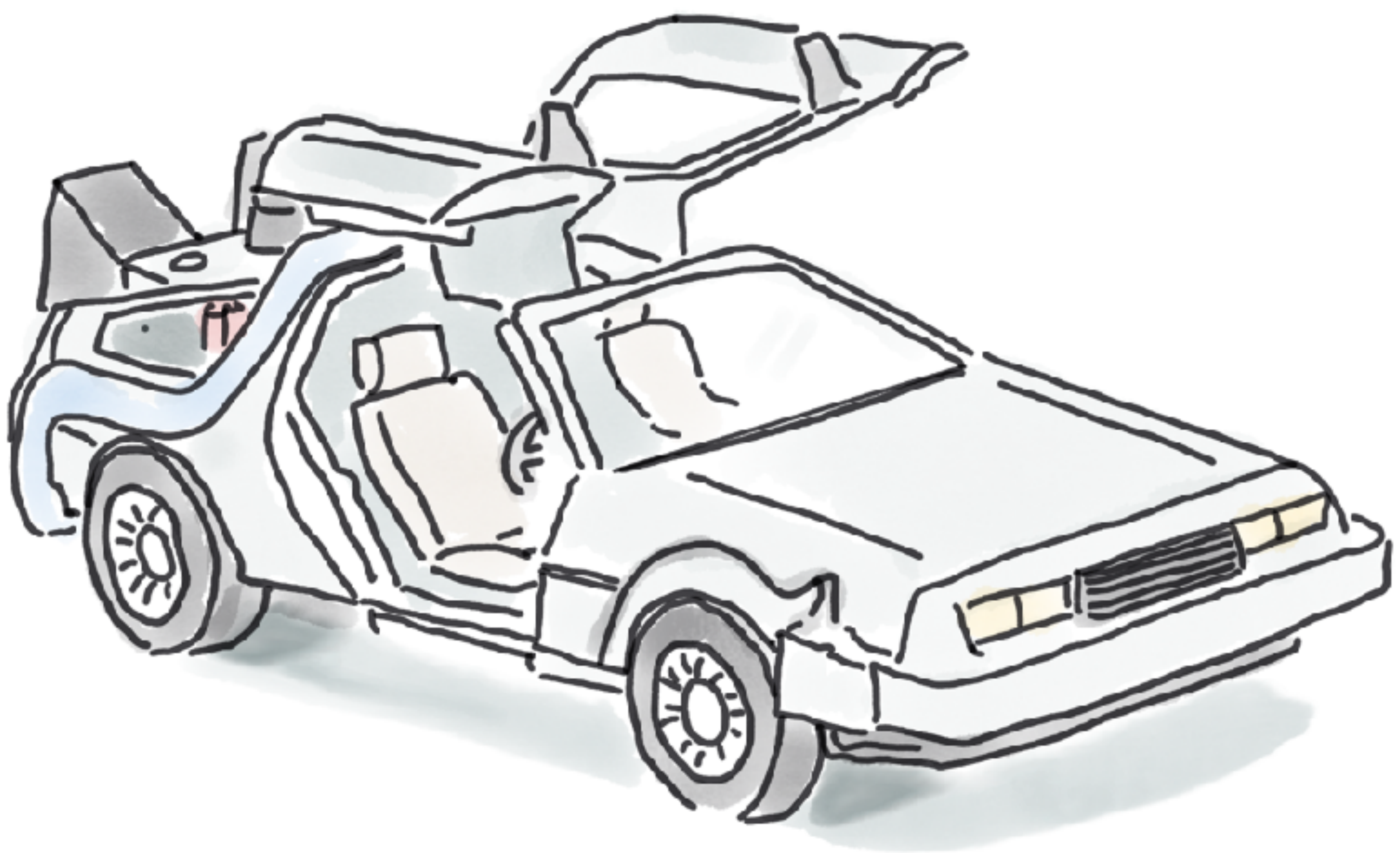
high max speed means **high** fuel usage per mile travelled



we need a new mental model for ‘virtuous’

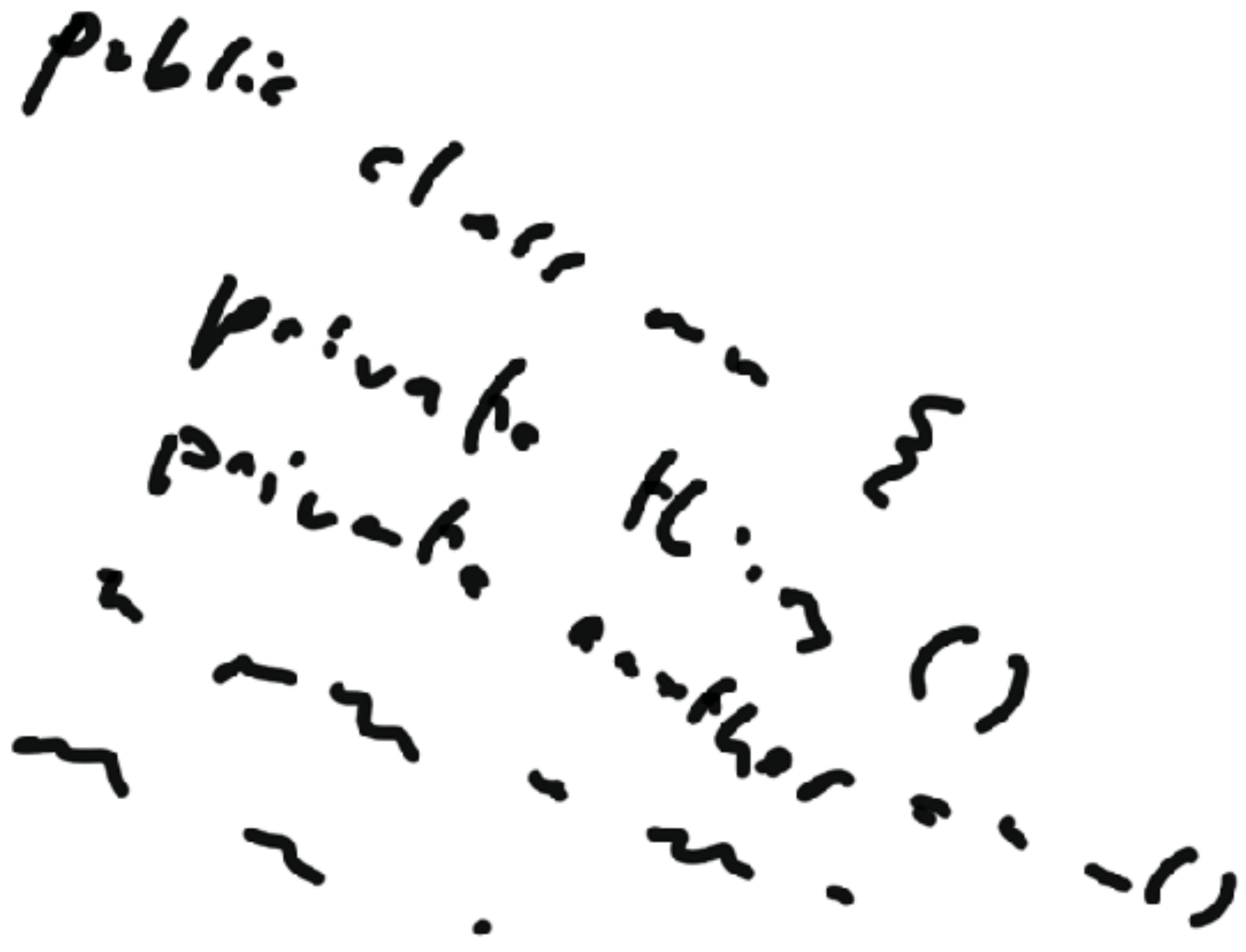
**car:**

high max speed means **high** fuel usage per mile travelled



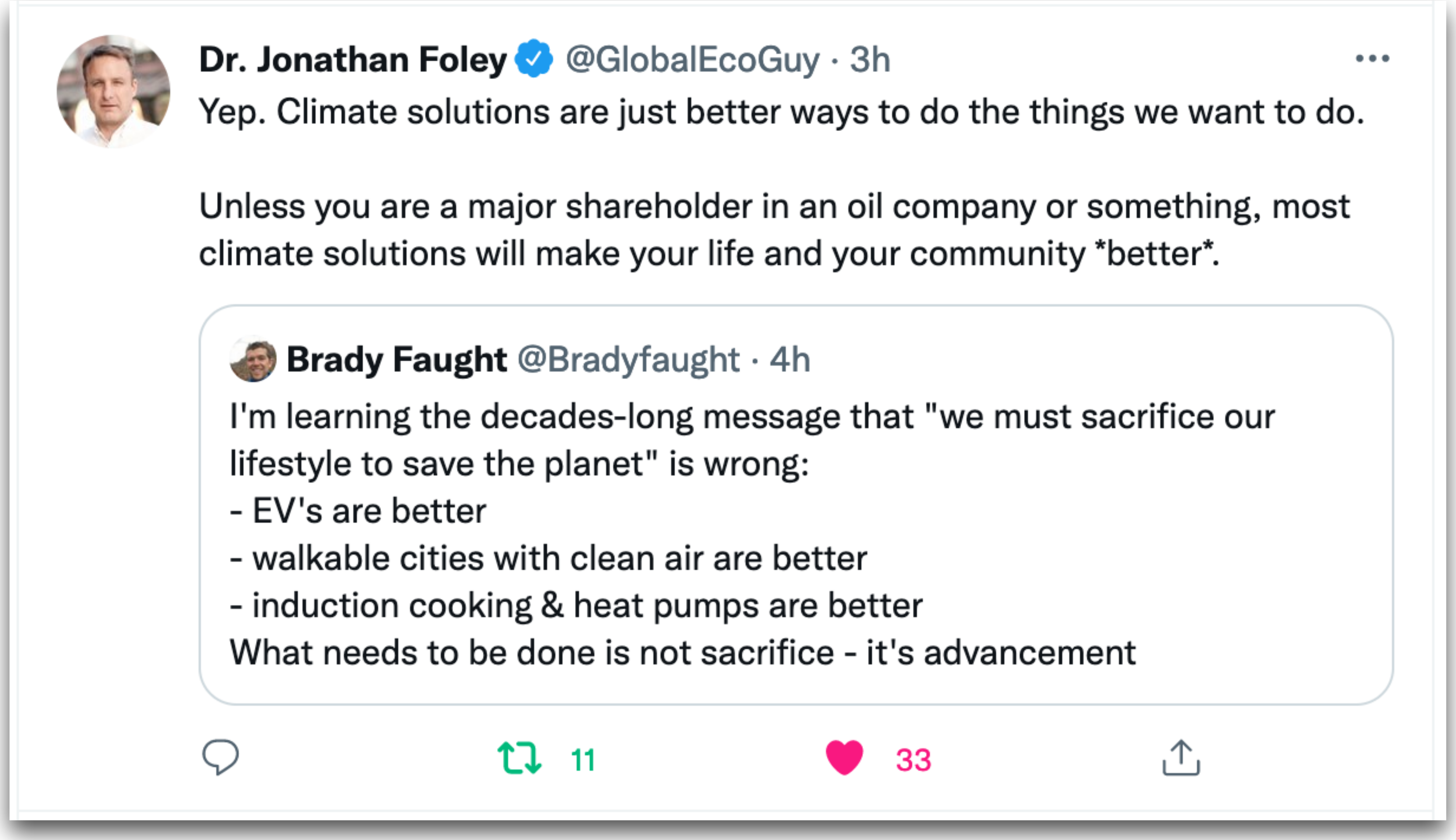
**software:**

high max transactions means **low** carbon per transaction



we need a new mental model for 'virtuous' goto; aarhus

# the double-win



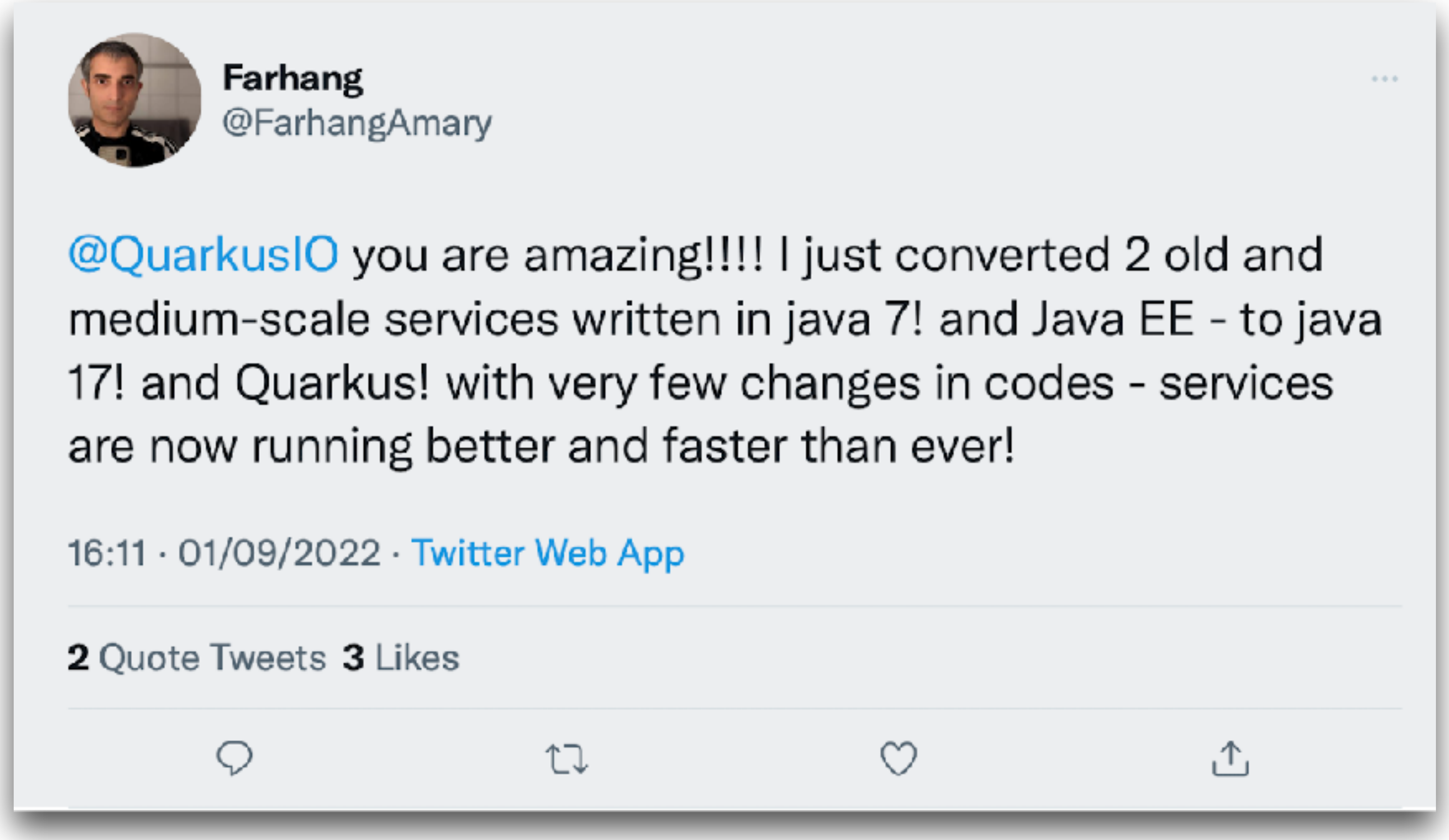
**Dr. Jonathan Foley** @GlobalEcoGuy · 3h

Yep. Climate solutions are just better ways to do the things we want to do.

Unless you are a major shareholder in an oil company or something, most climate solutions will make your life and your community *better*.

> **Brady Faught** @Bradyfaught · 4h
>
> I'm learning the decades-long message that "we must sacrifice our lifestyle to save the planet" is wrong:
> - EV's are better
> - walkable cities with clean air are better
> - induction cooking & heat pumps are better
> What needs to be done is not sacrifice - it's advancement

# the double-win



Farhang
@FarhangAmary

@QuarkusIO you are amazing!!!! I just converted 2 old and medium-scale services written in java 7! and Java EE - to java 17! and Quarkus! with very few changes in codes - services are now running better and faster than ever!

16:11 · 01/09/2022 · Twitter Web App

**2** Quote Tweets **3** Likes

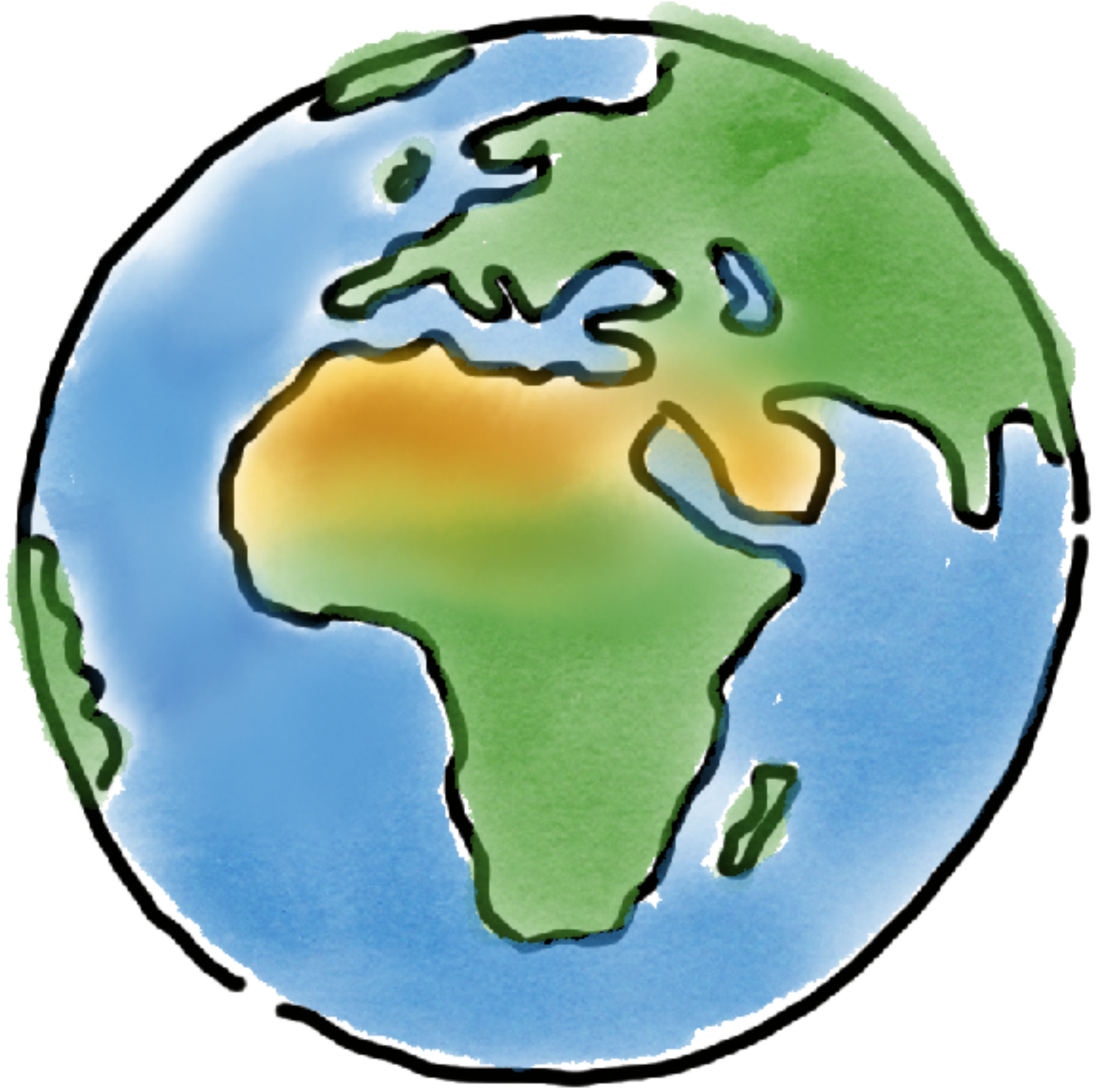
"this is not sacrifice. it's advancement."

– Dr. Jonathan Foley

# tl;dpa

(too long; didn't pay attention)

- **trick 1:** choose your hosting wisely
- **trick 2:** architect to be able to turn stuff off (LightSwitchOps)
- **trick 3:** the vrrrooooom model says faster is greener
- **trick 4:** the economic model says cheaper is greener
- **trick 5:** choose a fast and energy-efficient framework, such as quarkus



we all make a difference

Don’t forget to

**vote for this session**

in the **GOTO Guide app**

goto; aarhus

# thank you

@holly_cummins@hachyderm.io

slides