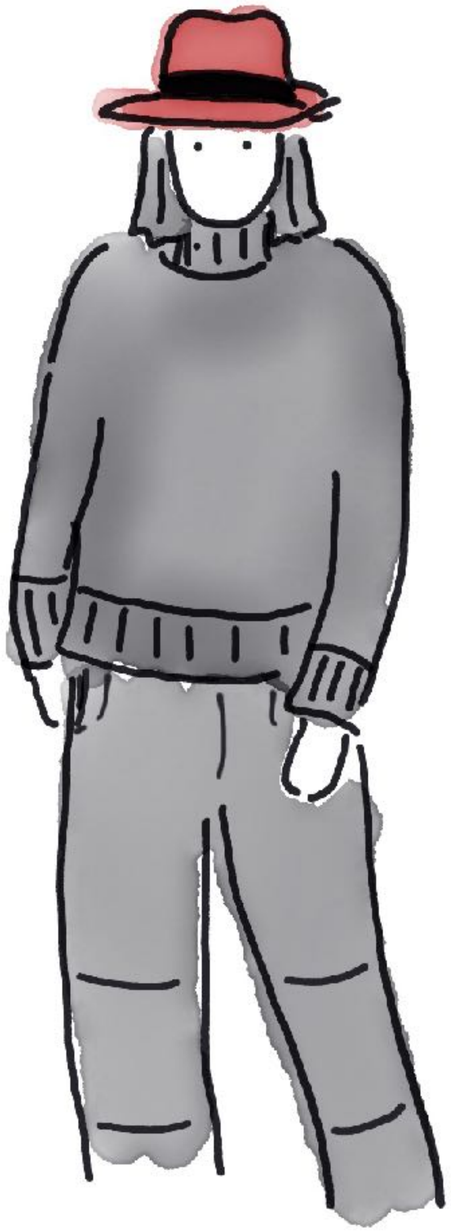
The Efficiency Paradox and How to Save Yourself and the World

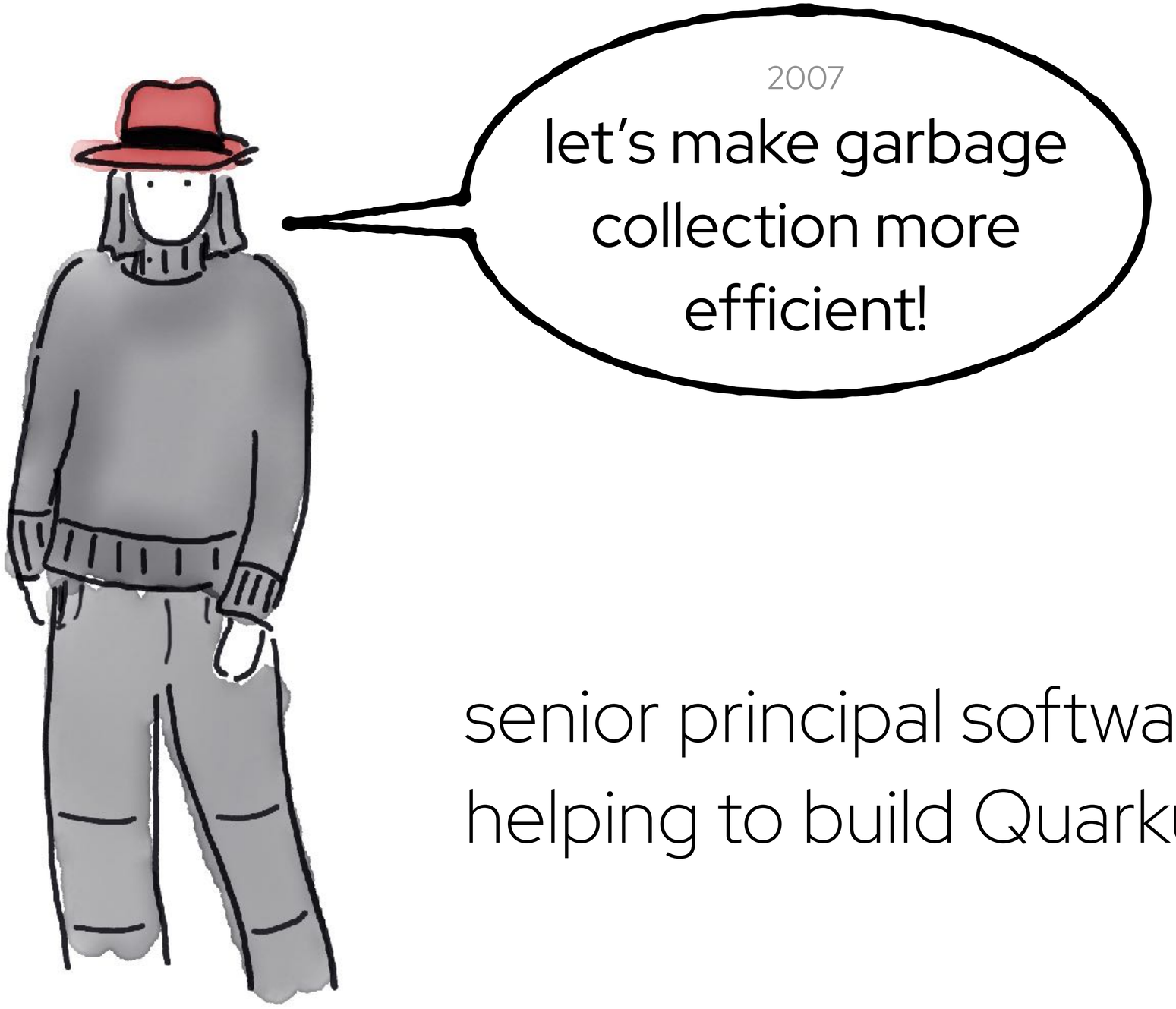
Holly Cummins

GOTO Copenhagen

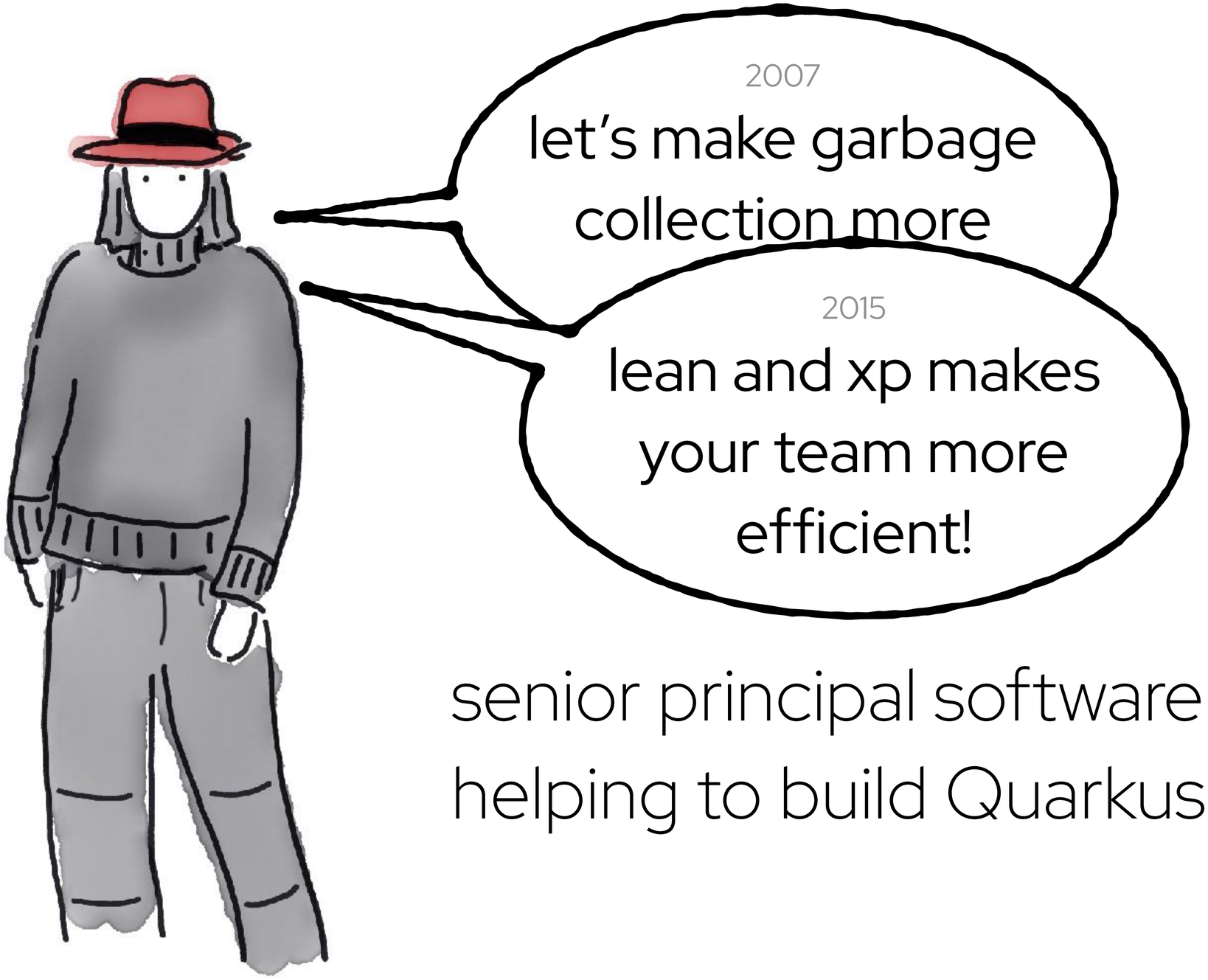


now

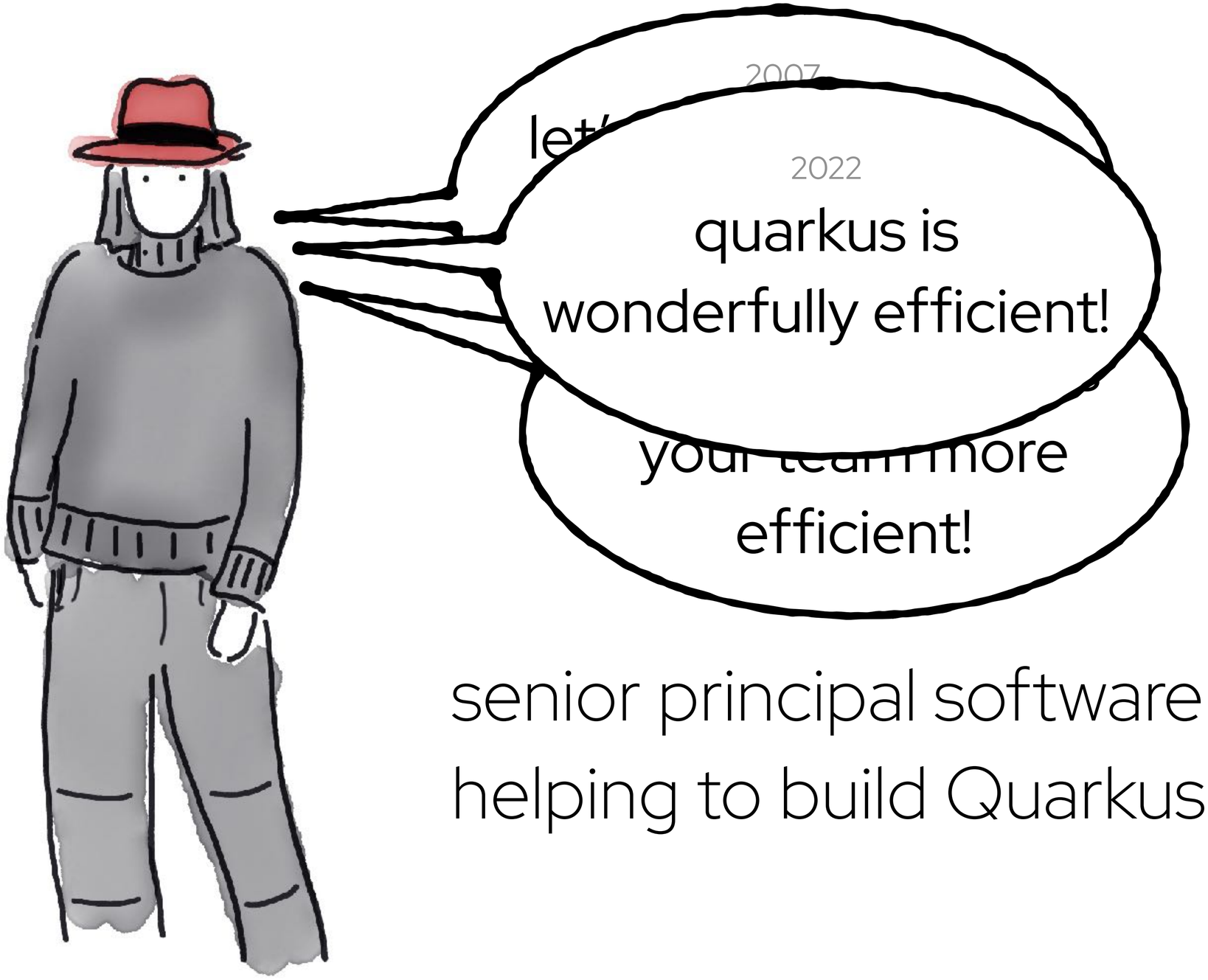
senior principal software engineer
helping to build Quarkus



senior principal software engineer
helping to build Quarkus

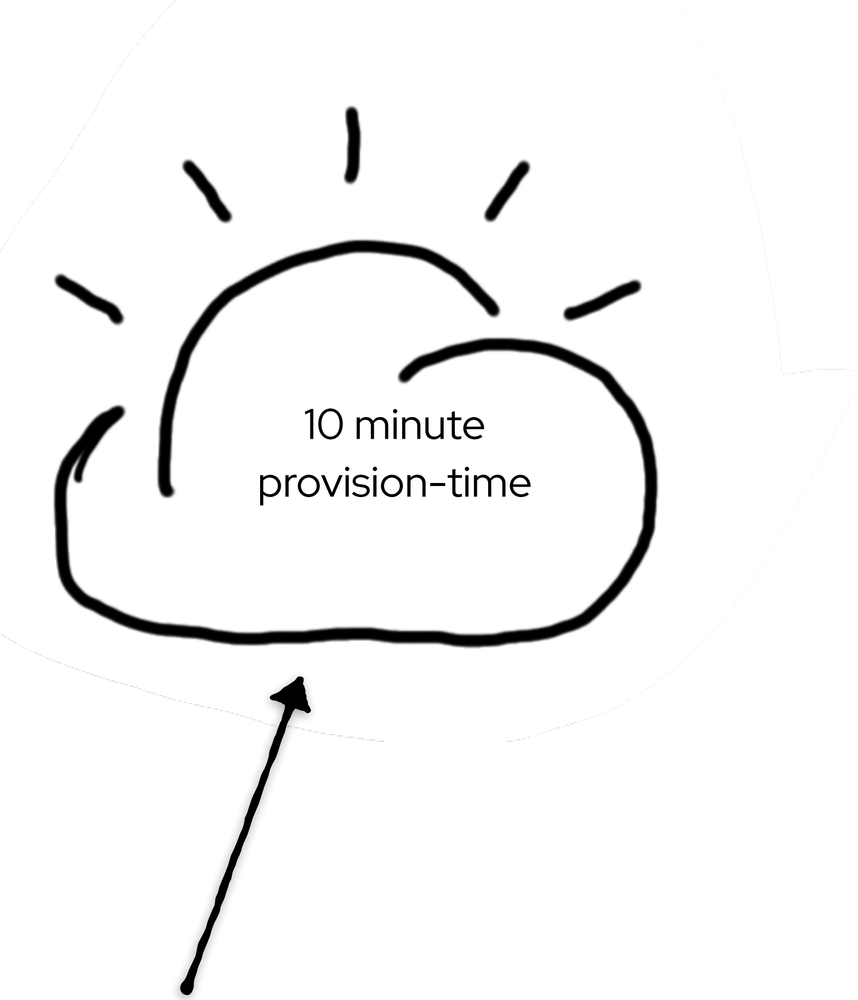


senior principal software engineer
helping to build Quarkus



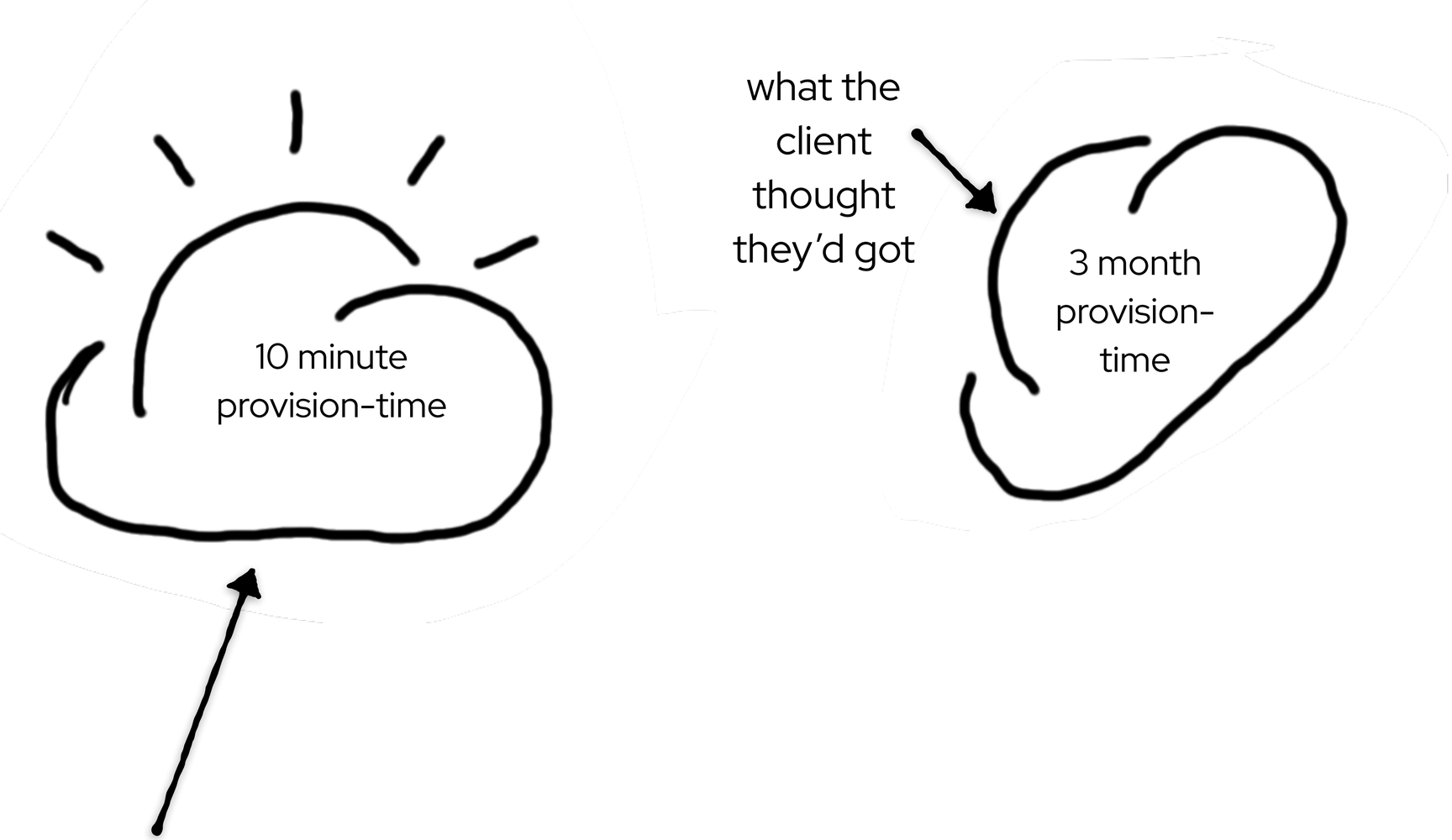
senior principal software engineer
helping to build Quarkus

“this provisioning software is broken”



what we sold

"this provisioning
software is broken"



what we sold

"this provisioning software is broken"

10 minute
provision-time

what we sold

what the
client
thought
they'd got

3 month
provision-
time

the reason

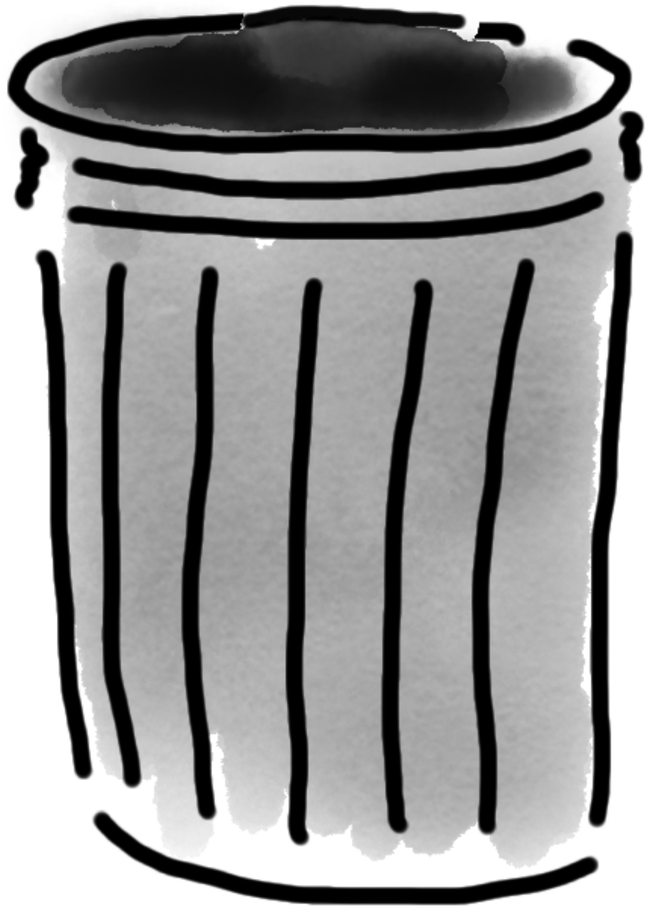
84-step
pre-approval process



"this provisioning
software is broken"

@holly_cummins

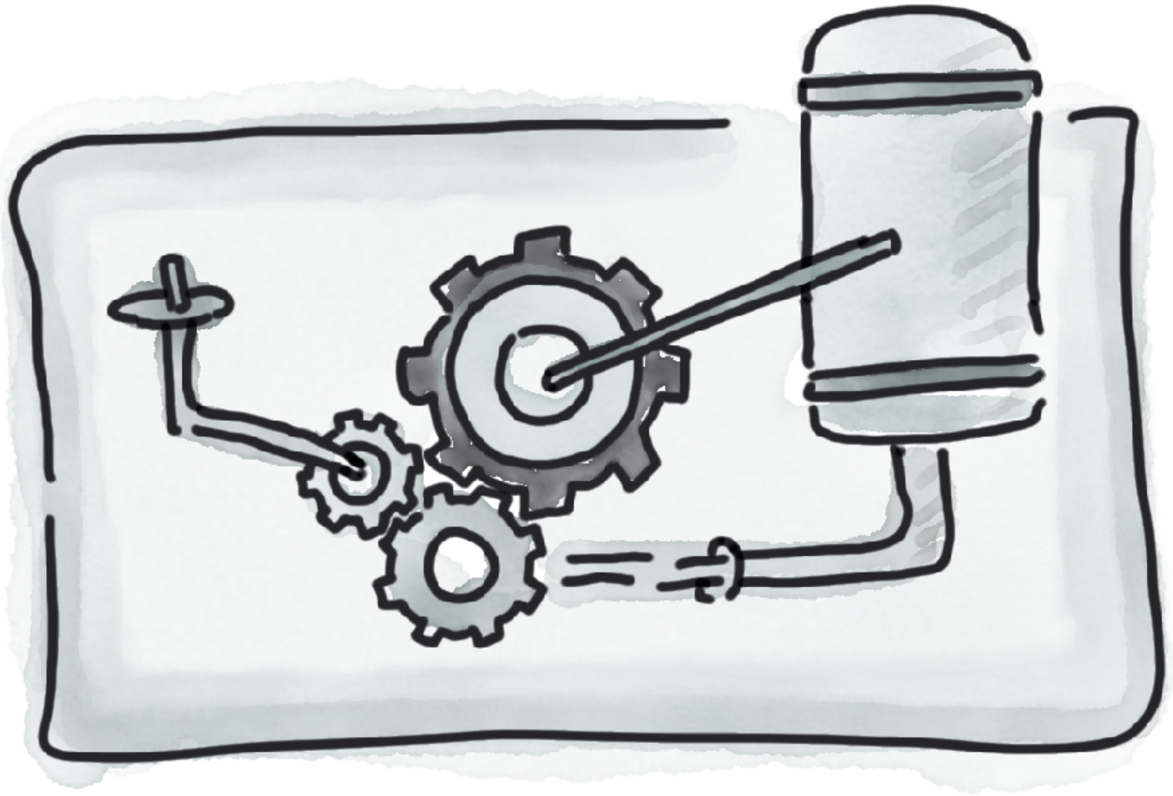
waste



waste is killing the planet

1700s

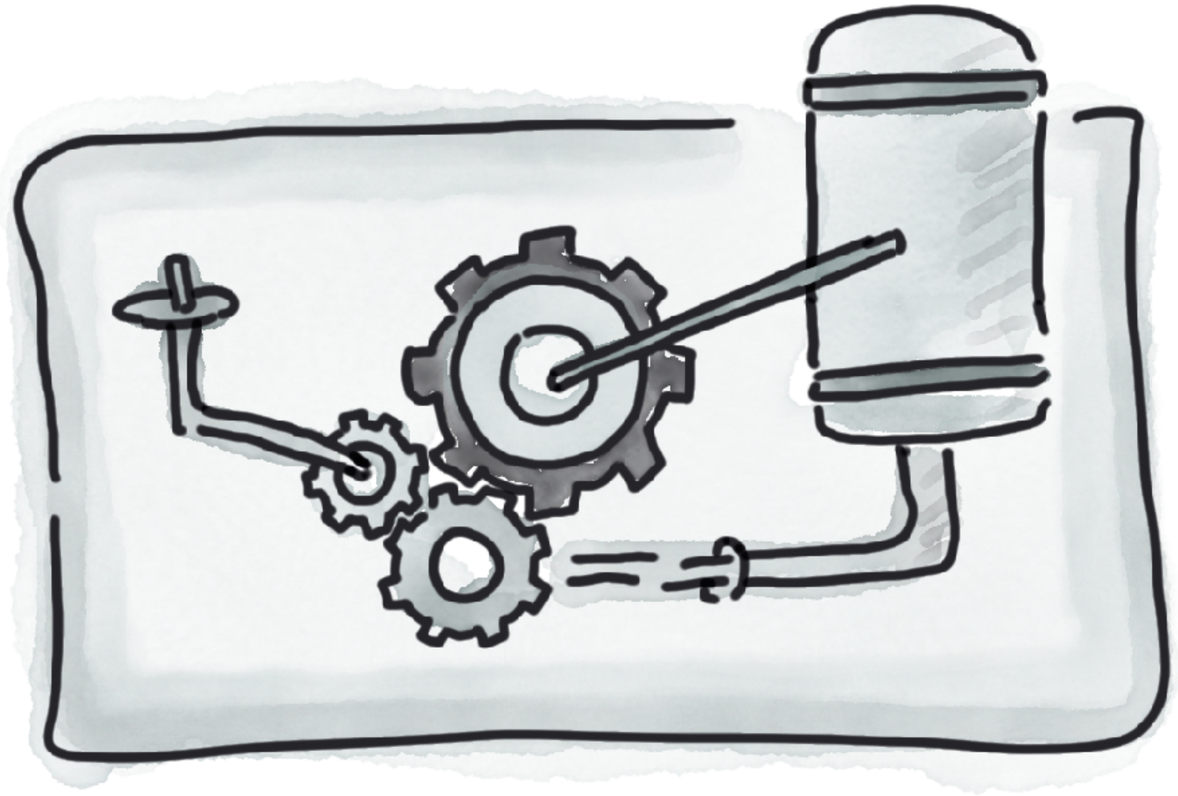
machine



1700s

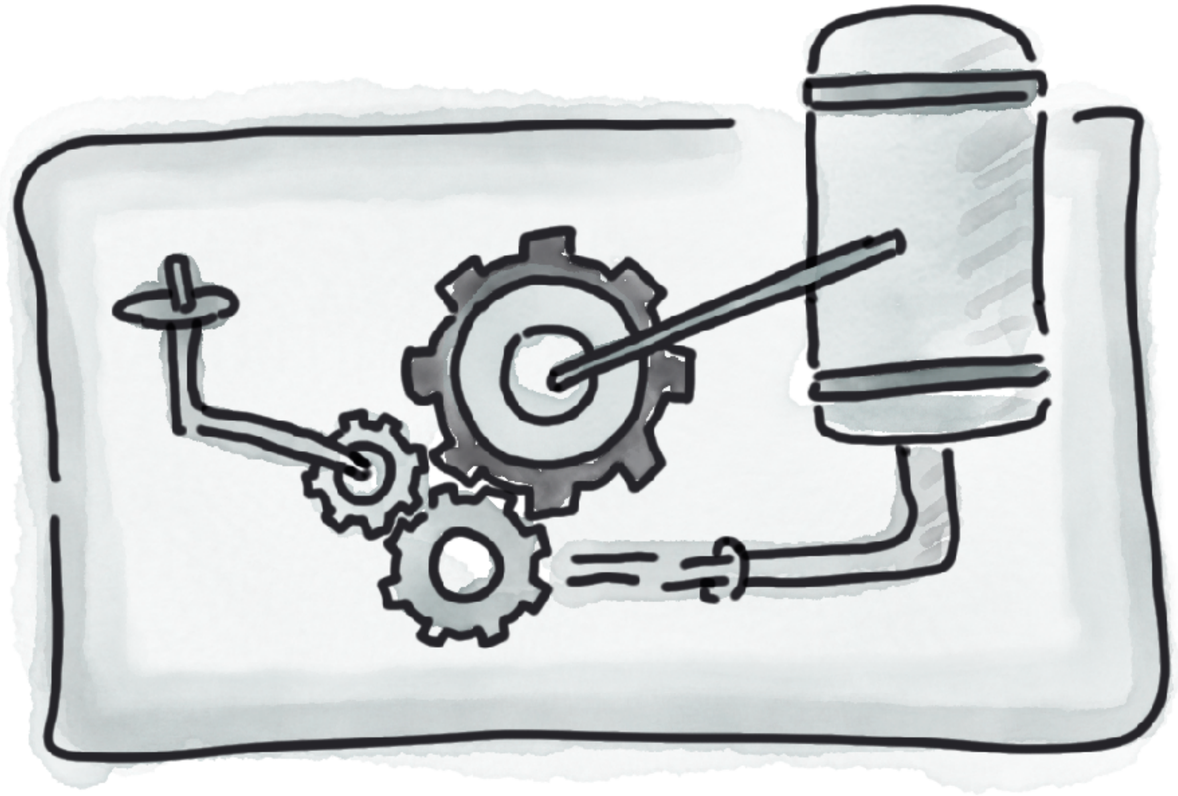
machine

energy →



1700s

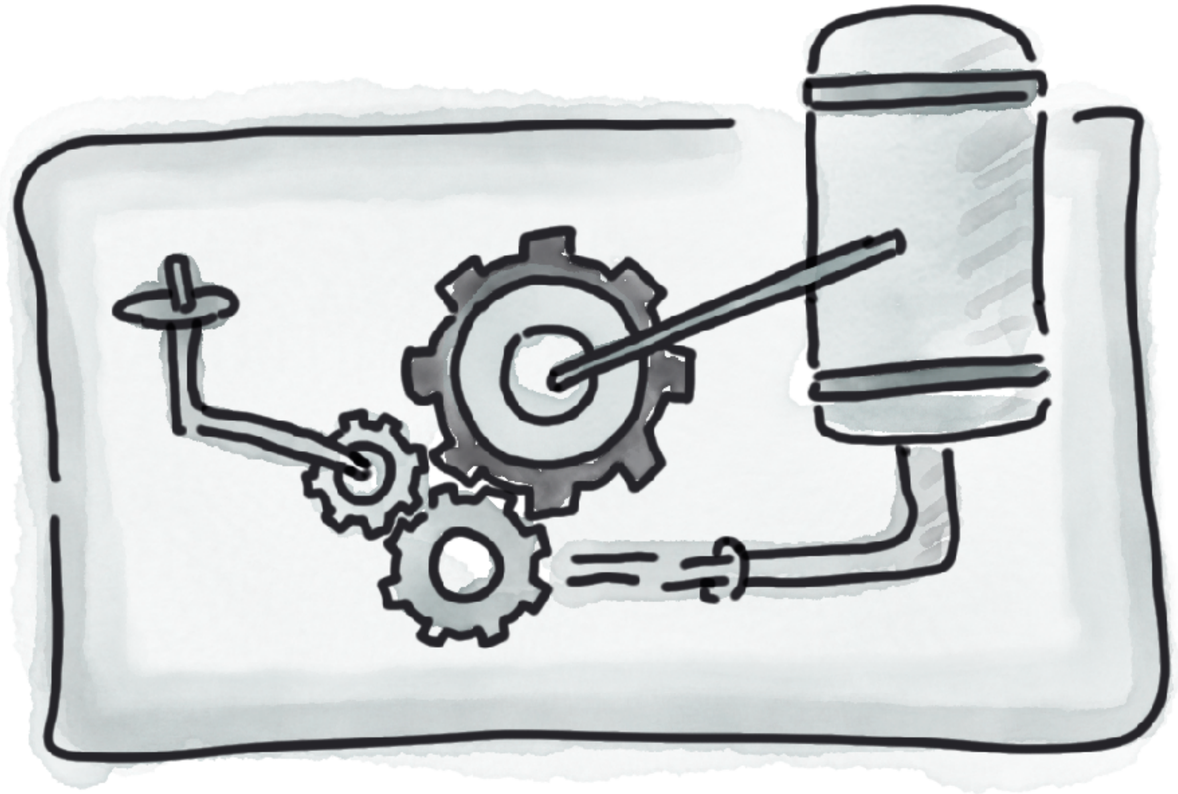
machine

energy →  → energy

1700s

machine

energy

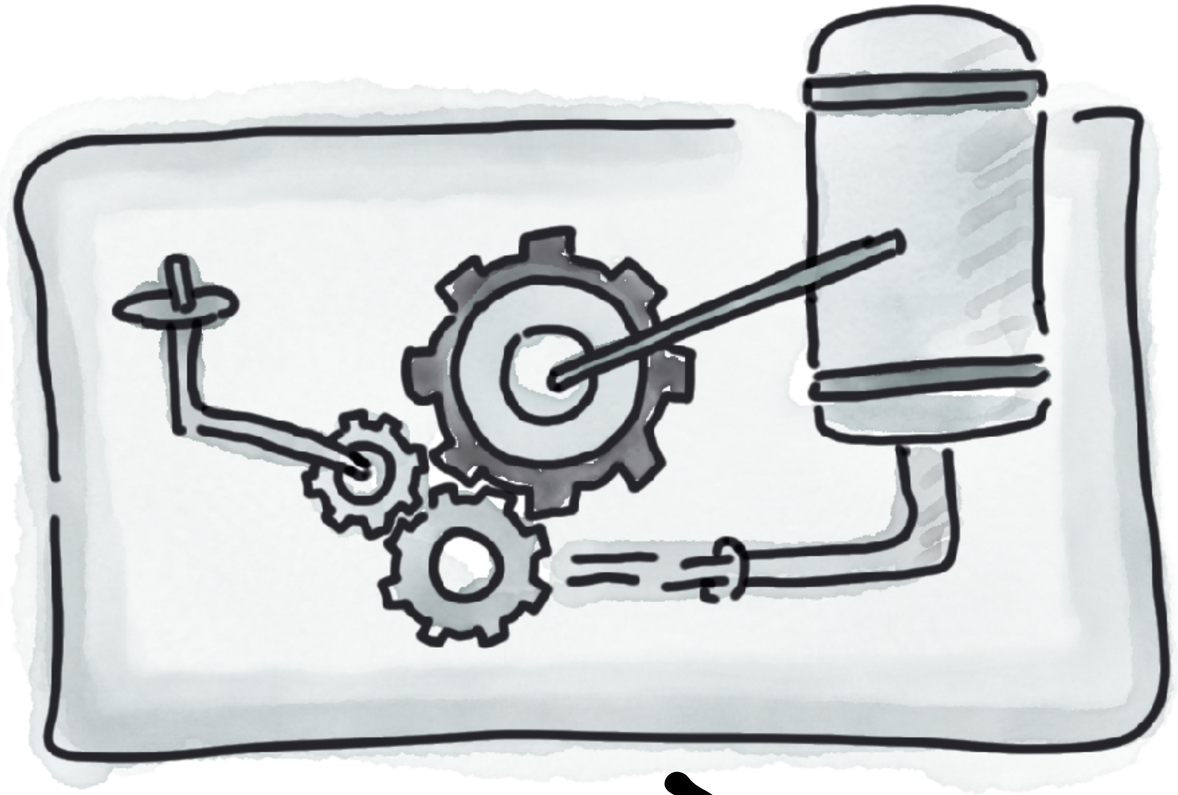


useful energy

1700s

machine

energy



useful energy

waste

1890s

processes



processes

time

money



1890s

processes

time
money



value

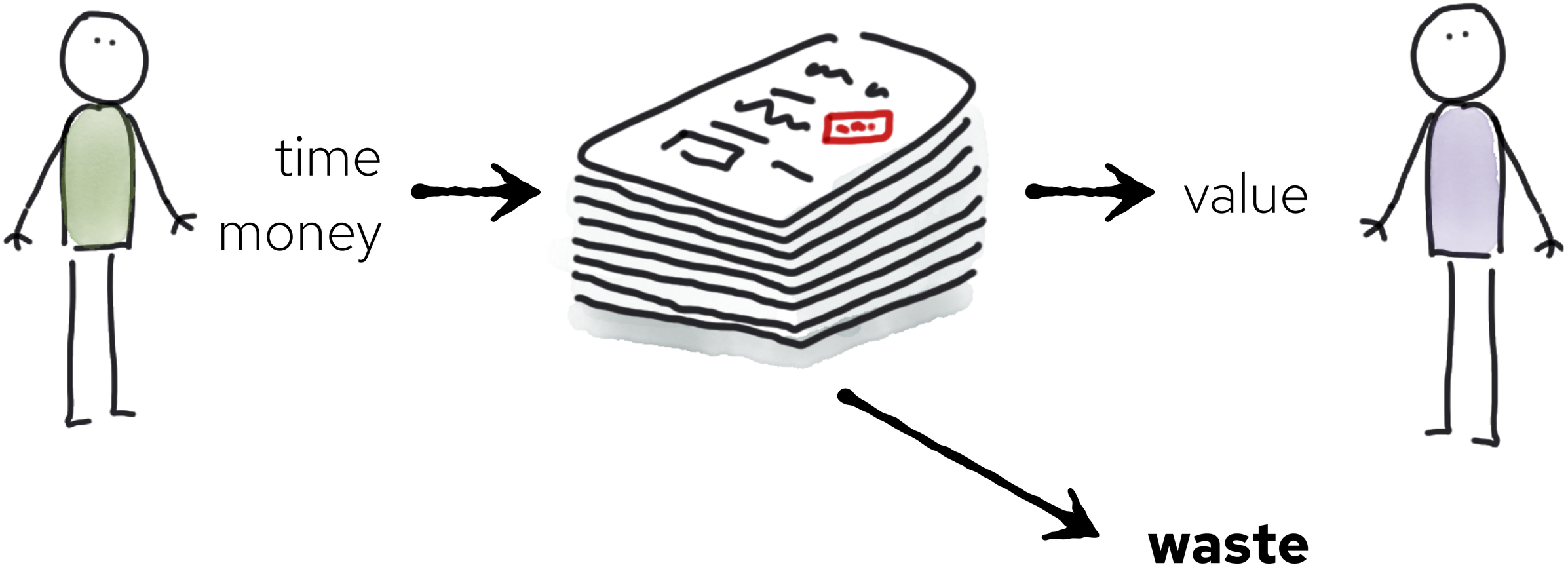
1890s

processes



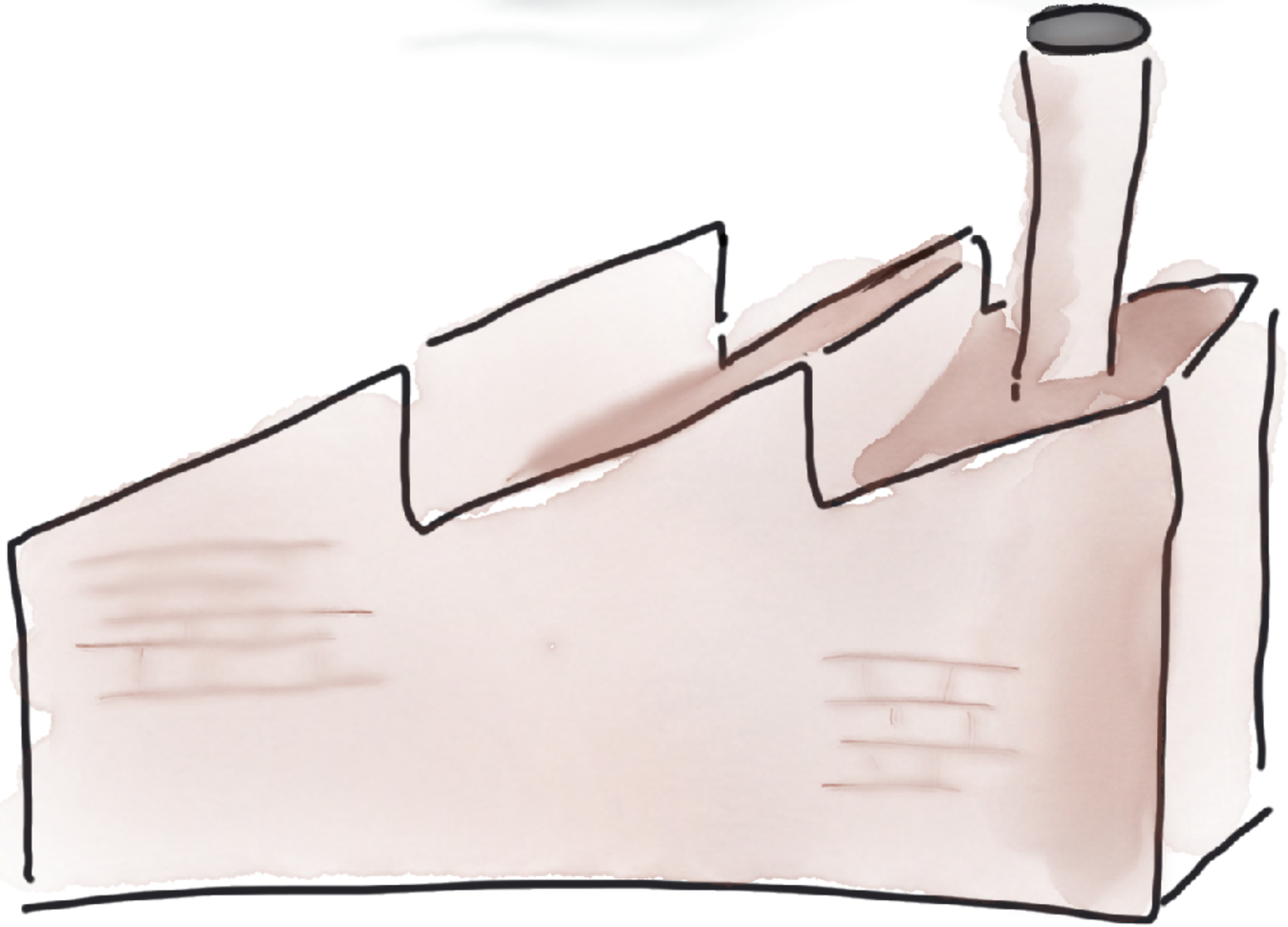
1890s

processes



1900s

manufacturing

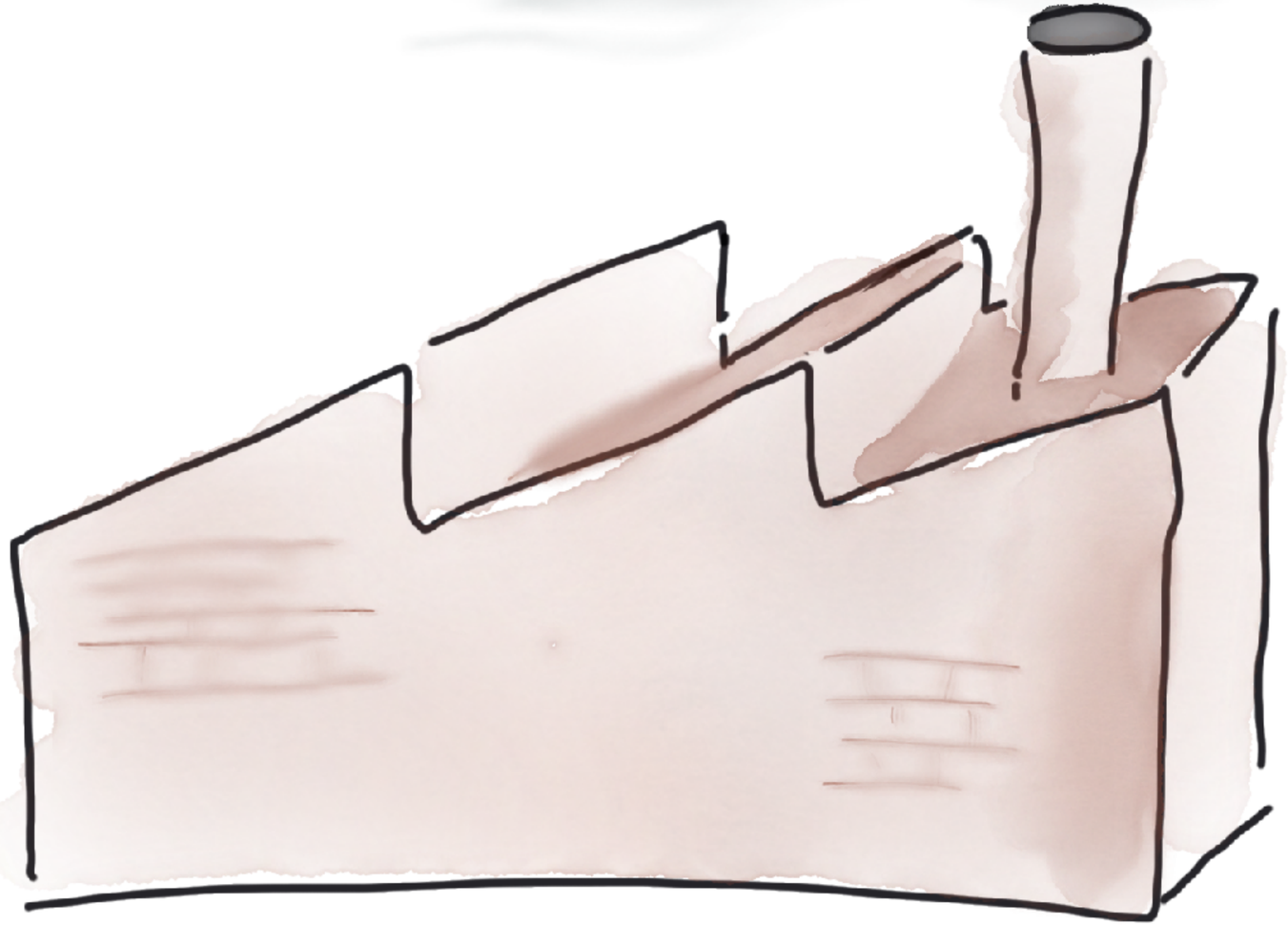


1900s

manufacturing

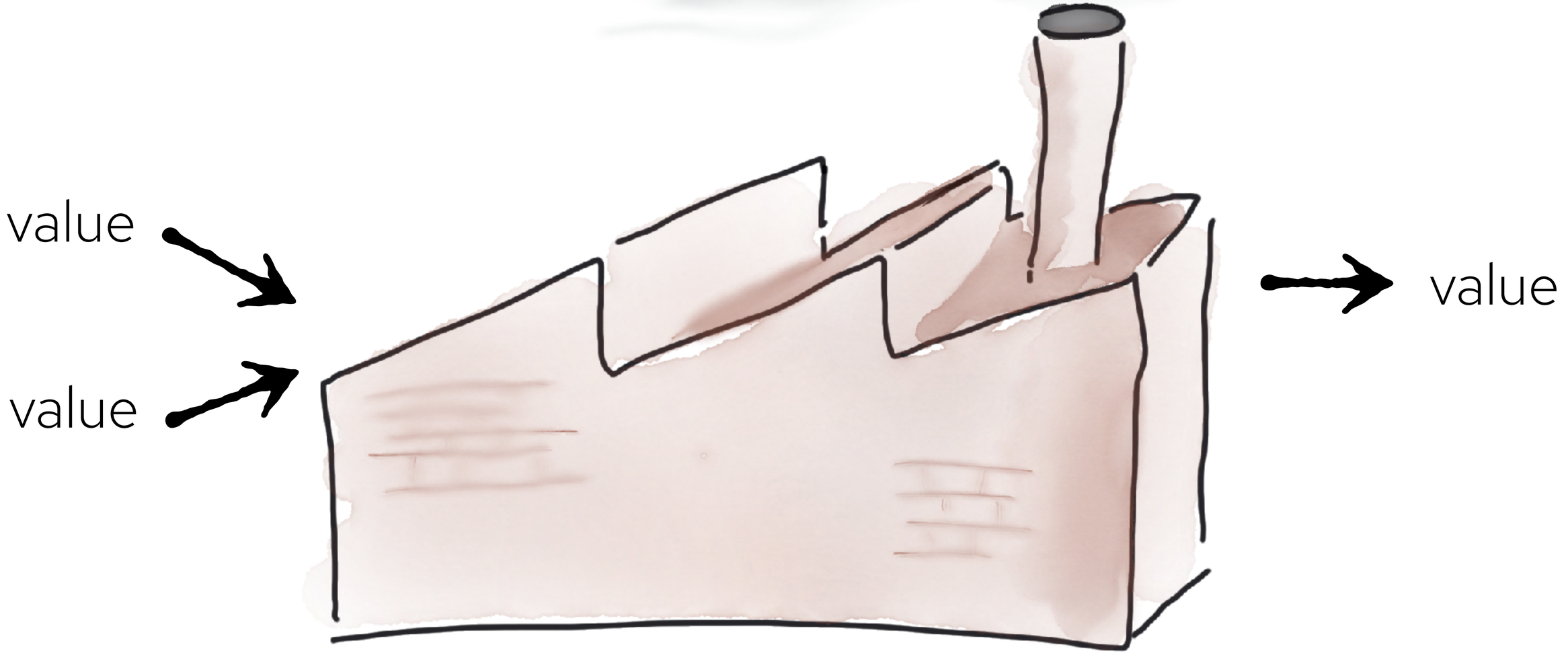
value

value



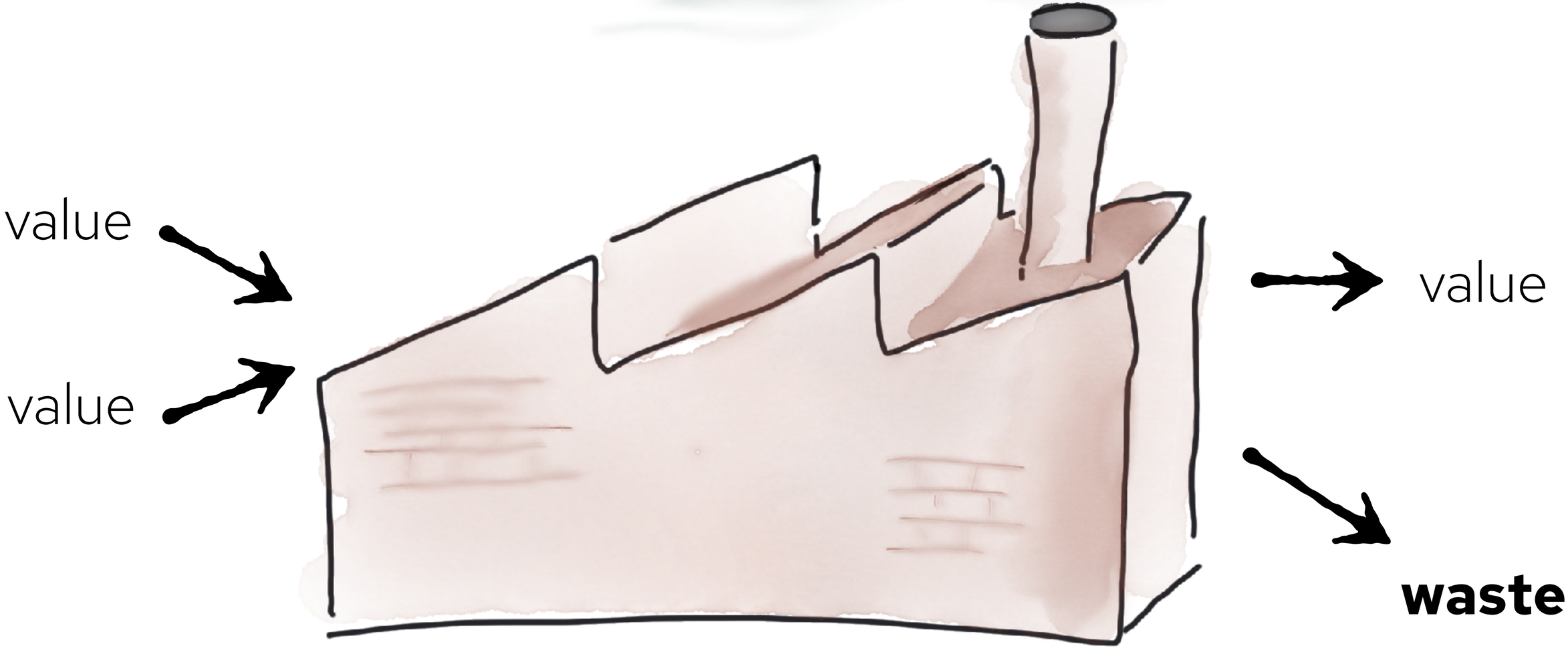
1900s

manufacturing



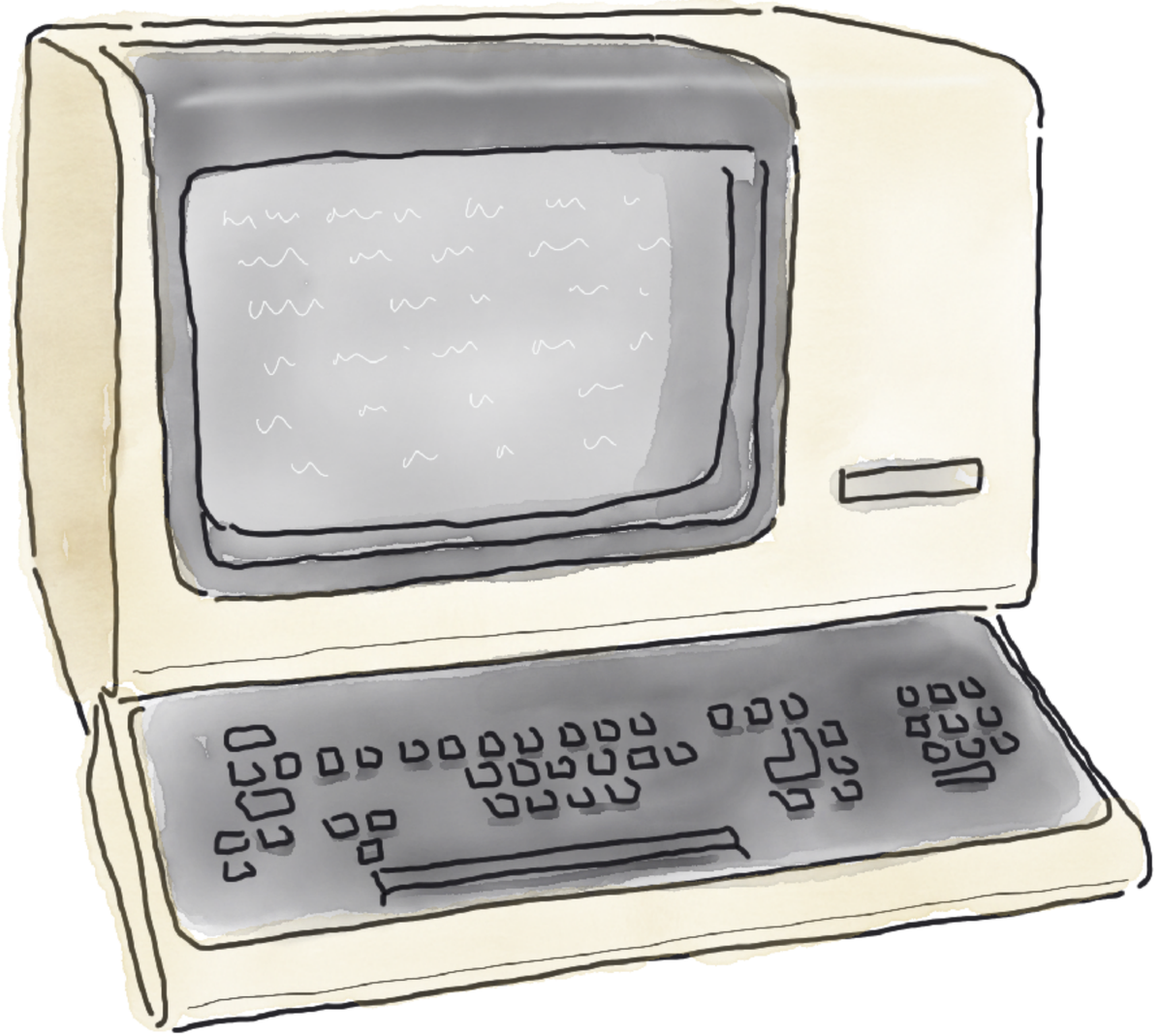
1900s

manufacturing



software

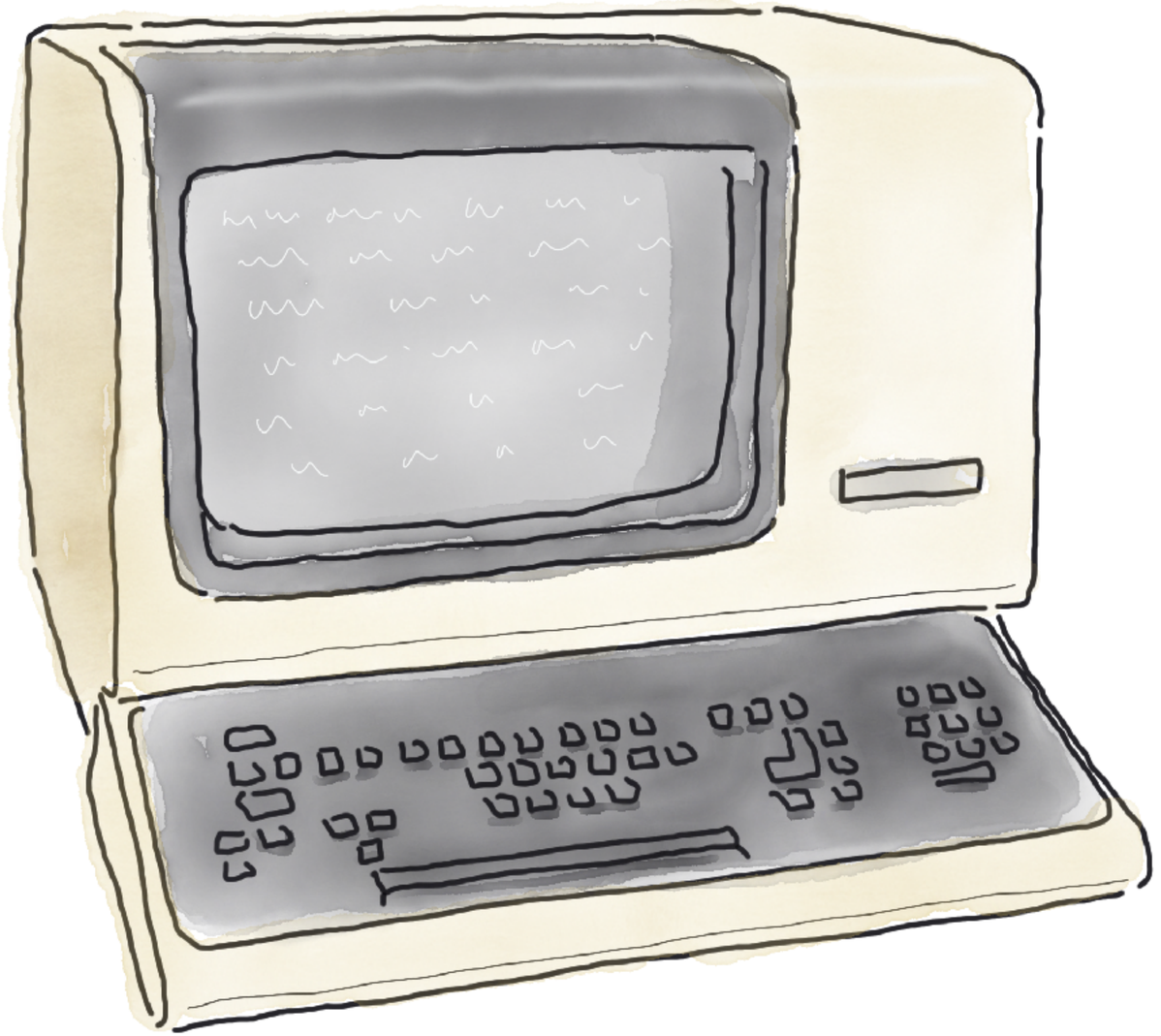
1960s



software

1960s

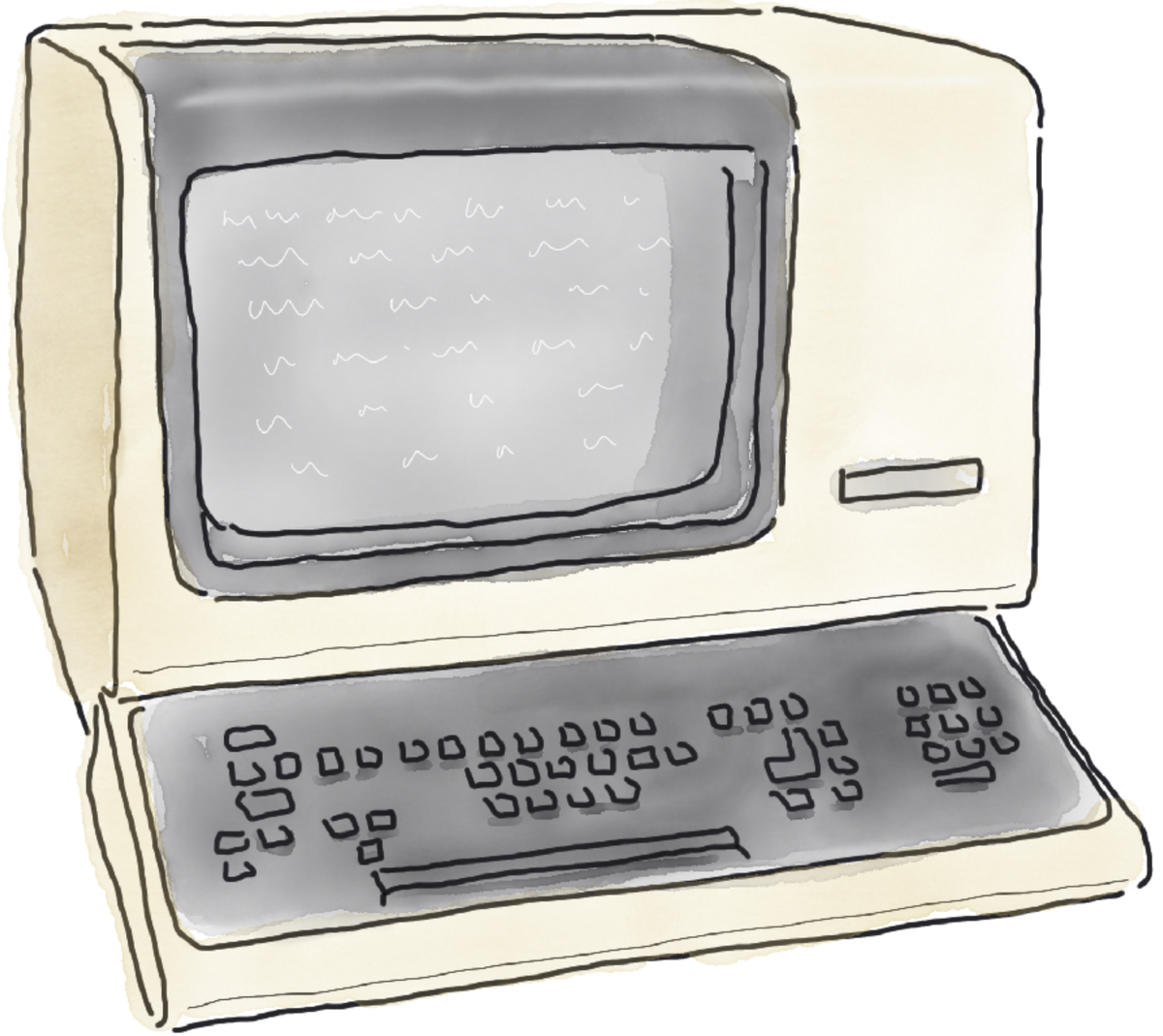
time
electricity
hardware



software

1960s

time
electricity
hardware

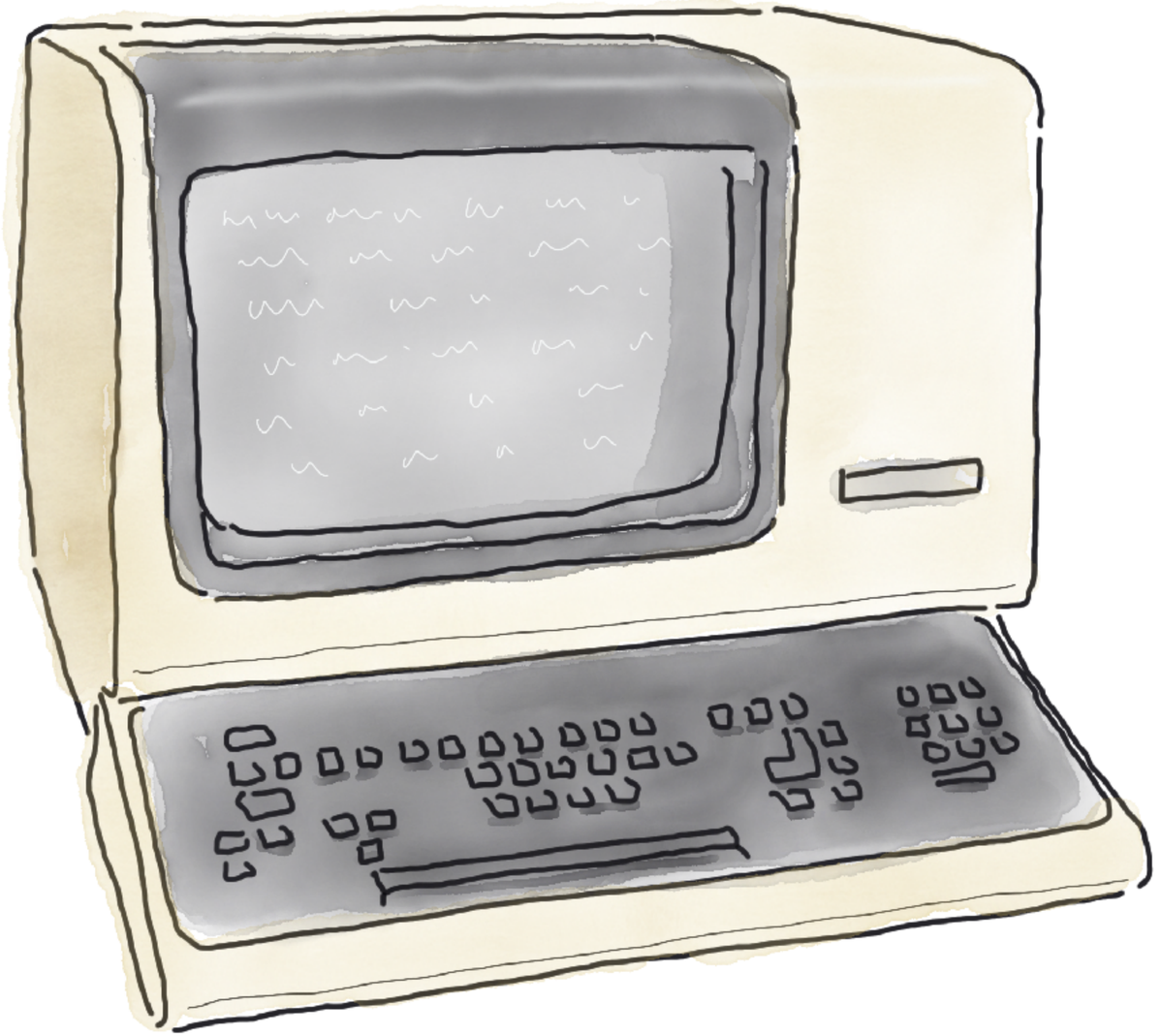


answers

software

1960s

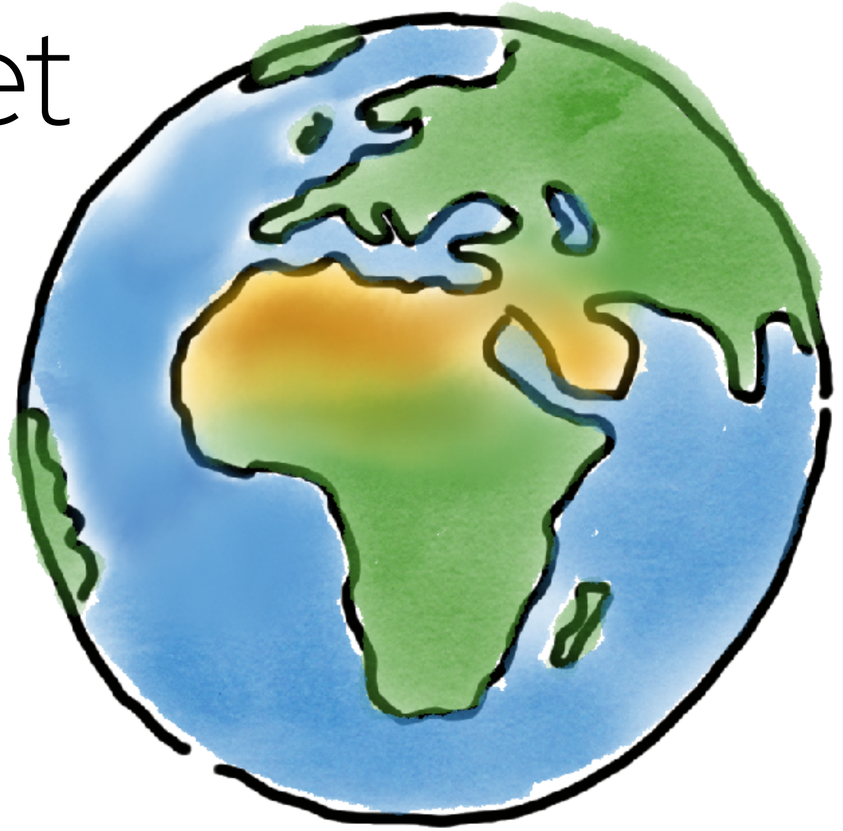
time
electricity
hardware



answers

waste

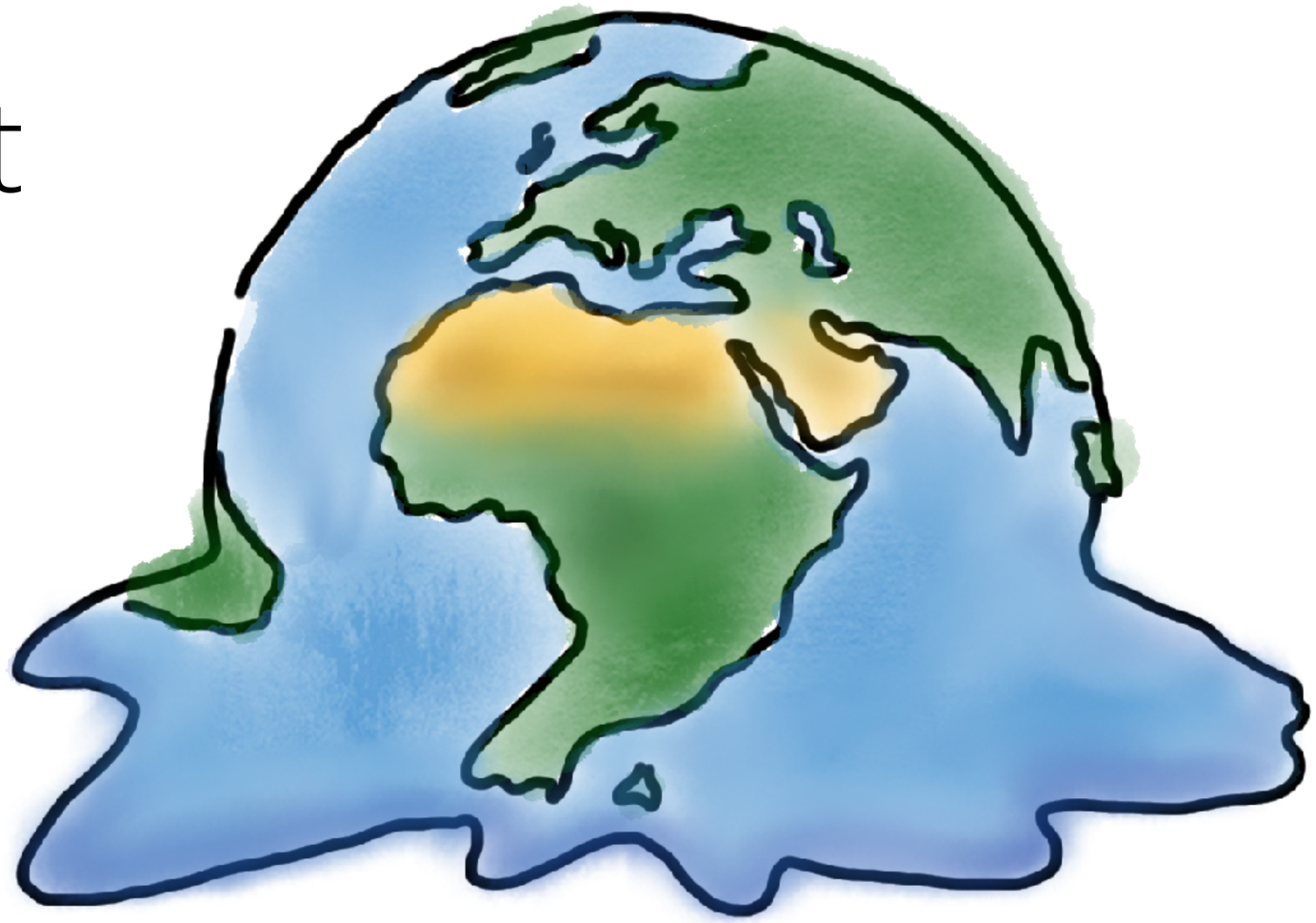
waste is killing the planet



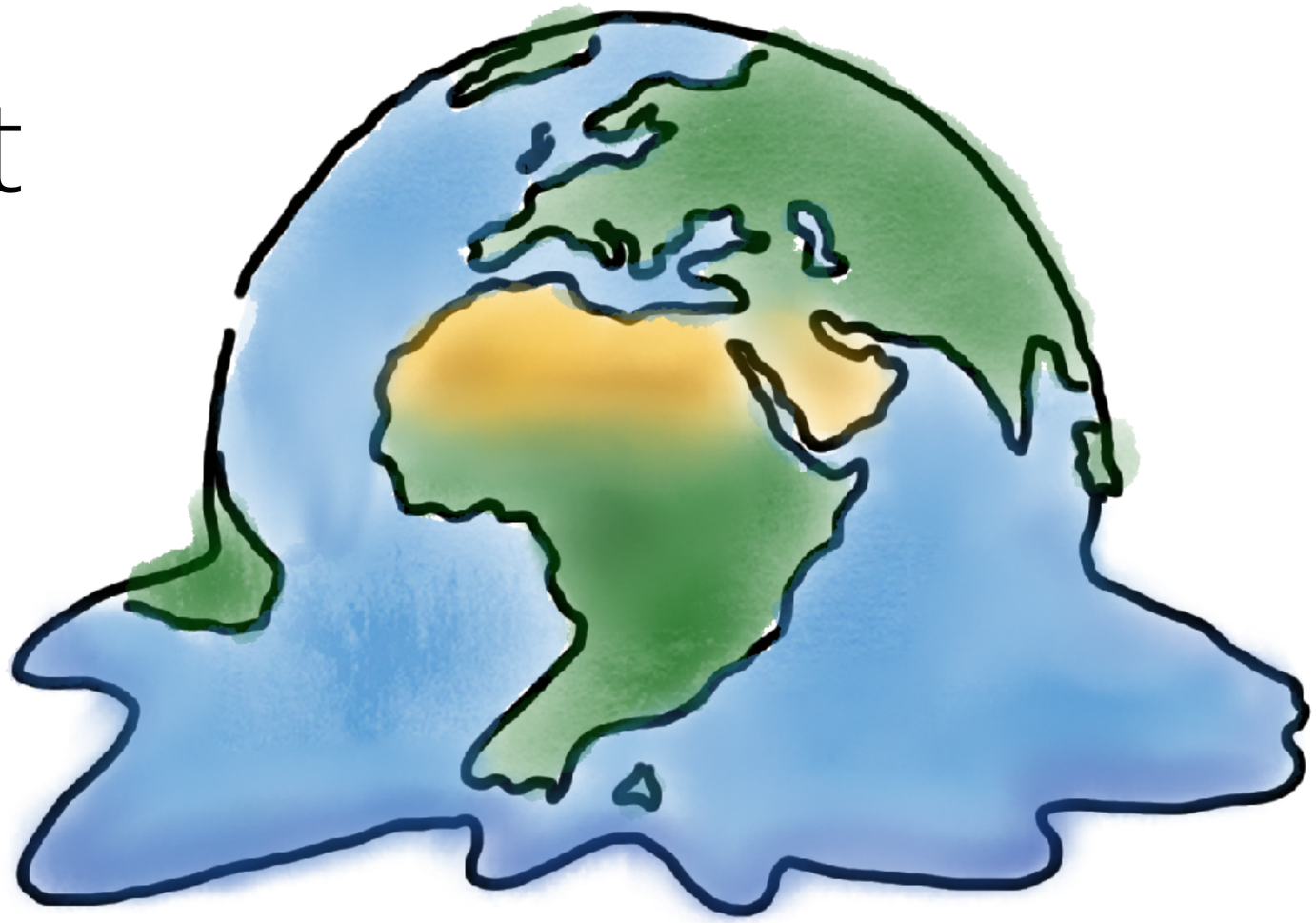
e-waste is killing the planet



energy waste is
killing the planet



slow code is
killing the planet







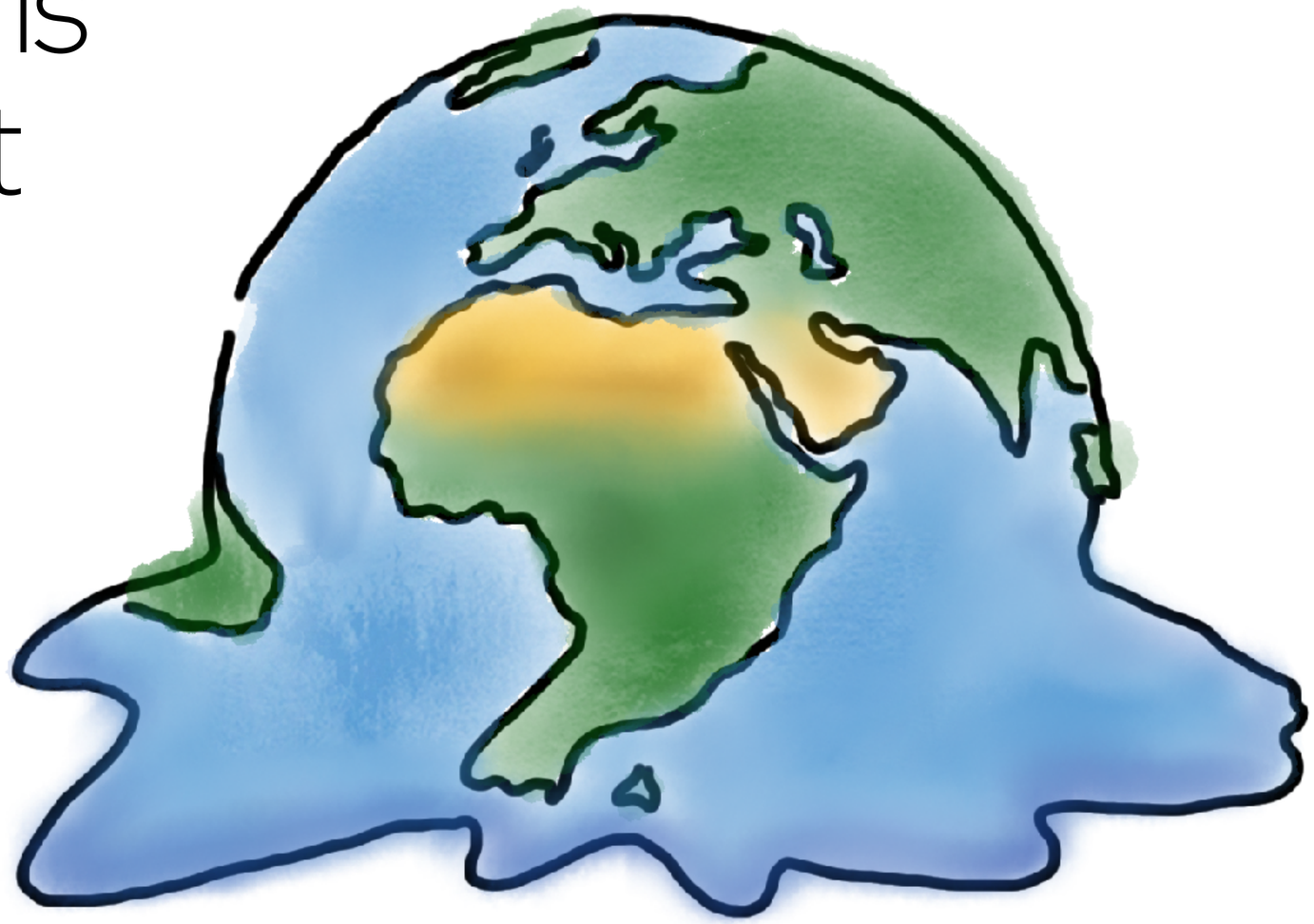
"my heart sank ... our new feature failed to load because of poor internet connectivity"

modern web is so inefficient it is **useless** for part of its audience



"my heart sank ... our new feature failed to load because of poor internet connectivity"

forgotten code is killing the planet

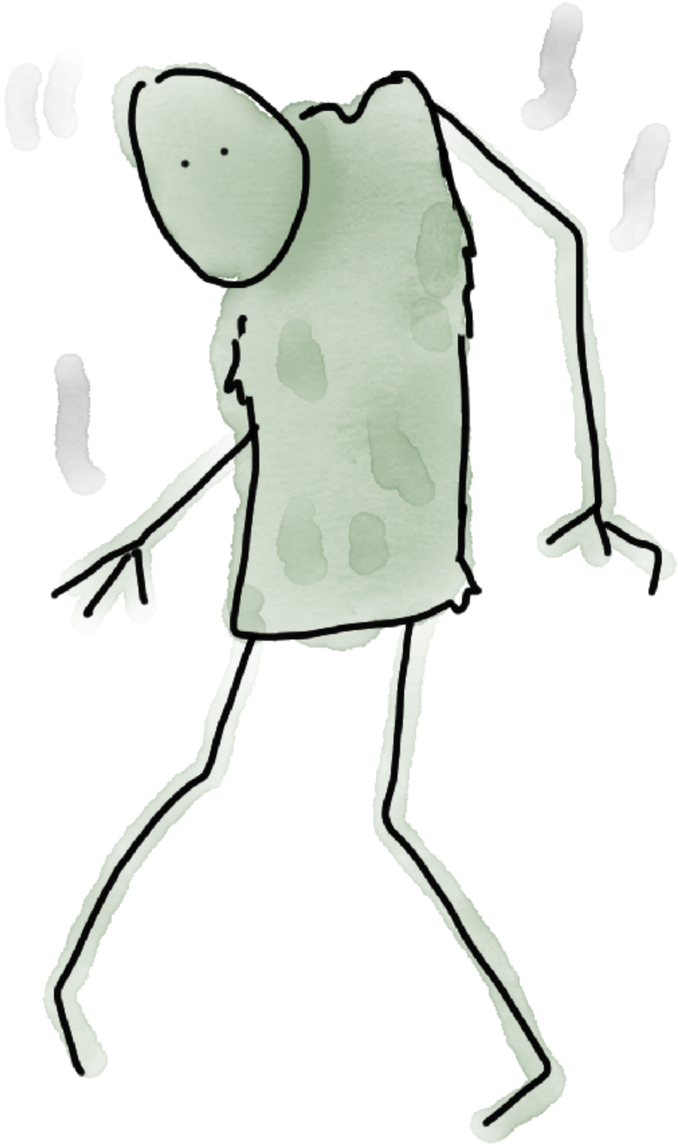


https://www.anthesisgroup.com/wp-content/uploads/2019/11/Comatose-Servers-Redux-2017.pdf

zombie servers

25%

of 16,000 servers
doing **no** useful work



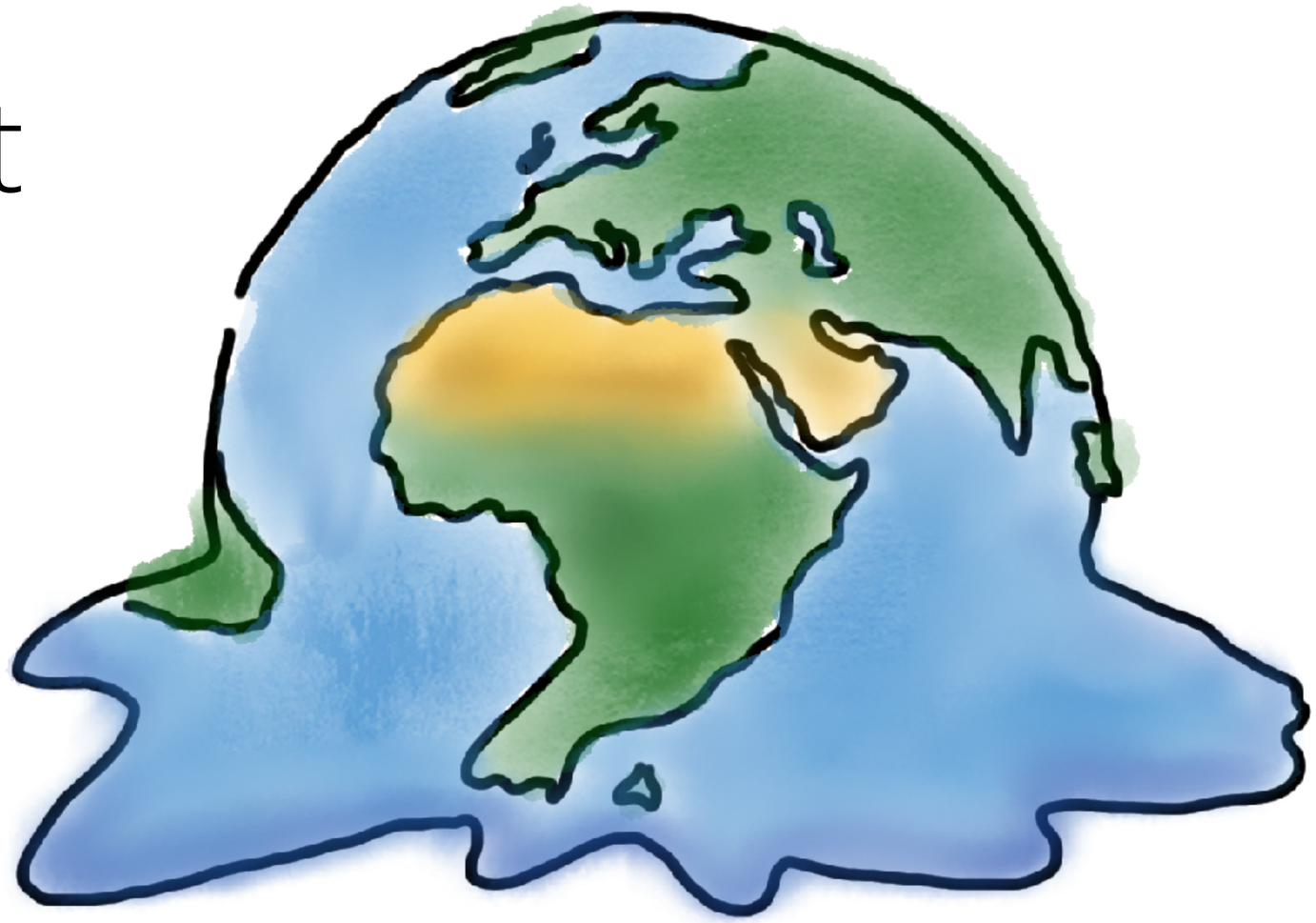
https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IB.pdf

the average server:

12 - 18%

of capacity

unused data is
killing the planet



https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IB.pdf

68%

of data is stored and never used

https://www.seagate.com/gb/en/news/news-archive/seagates-rethink-data-report-reveals-that-68-percent-of-data-available-to-businesses-goes-unleveraged-pr-master/

it’s not just money

it’s not just money

it’s not just electricity

it’s not just money

it’s not just electricity

it’s embodied carbon

it’s not just money

it’s not just electricity

it’s embodied carbon

it’s water



it’s not just money

it’s not just electricity

it’s embodied carbon

it’s water

it’s e-waste

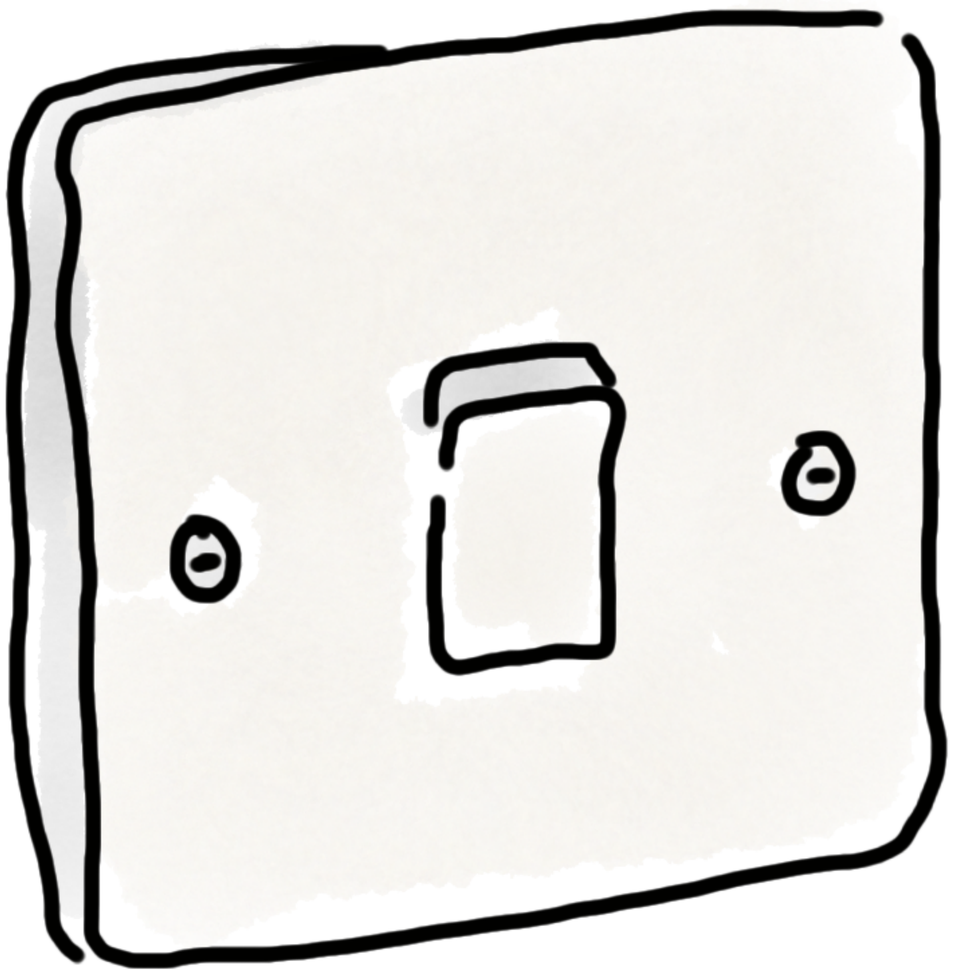


good news, everyone

we have **solutions**

making turning servers off as safe and easy as turning lights off.

and then automate it.

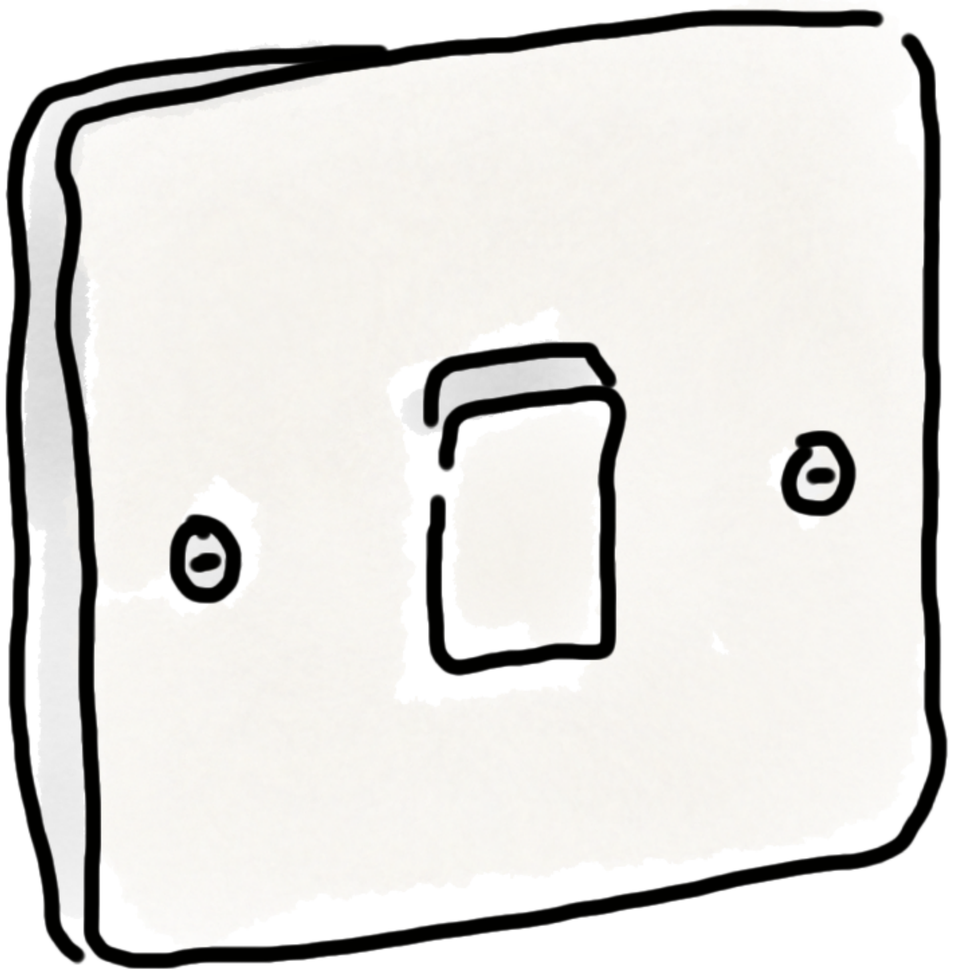


solution:

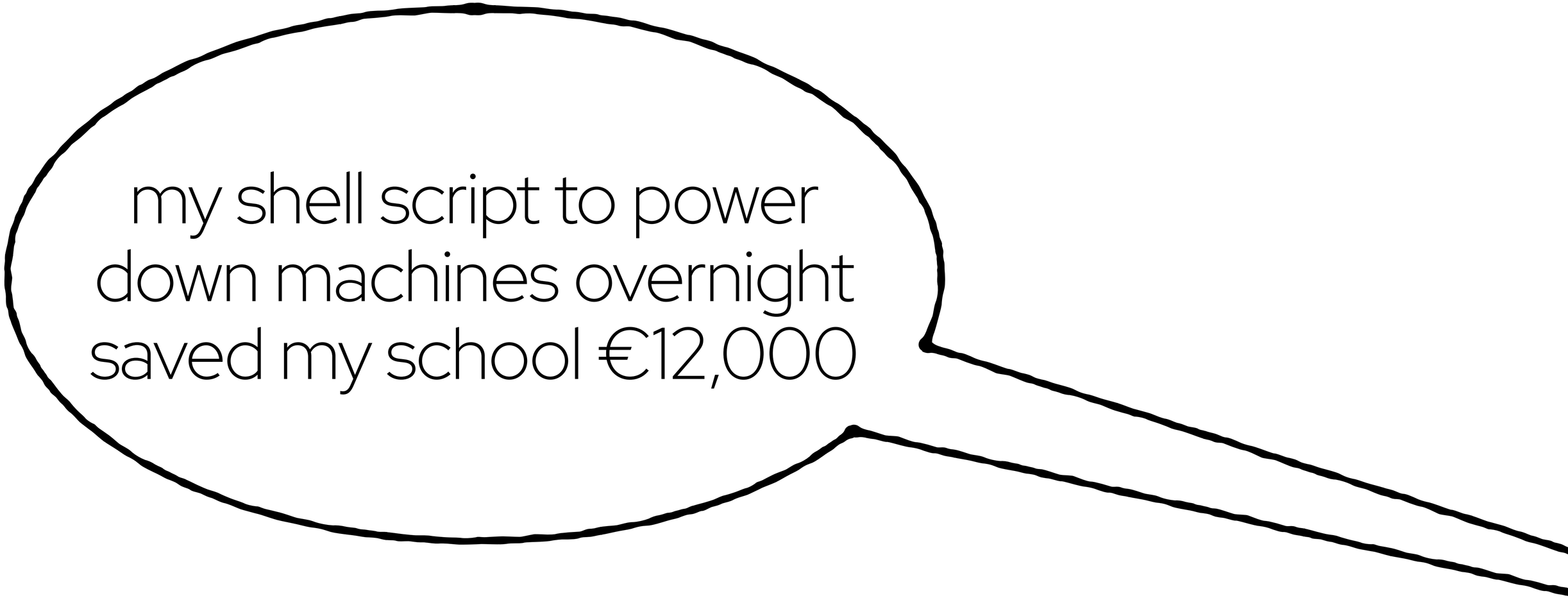
LightSwitchOps

making turning servers off as safe and easy as turning lights off.

and then automate it.



absurdly simple scripts

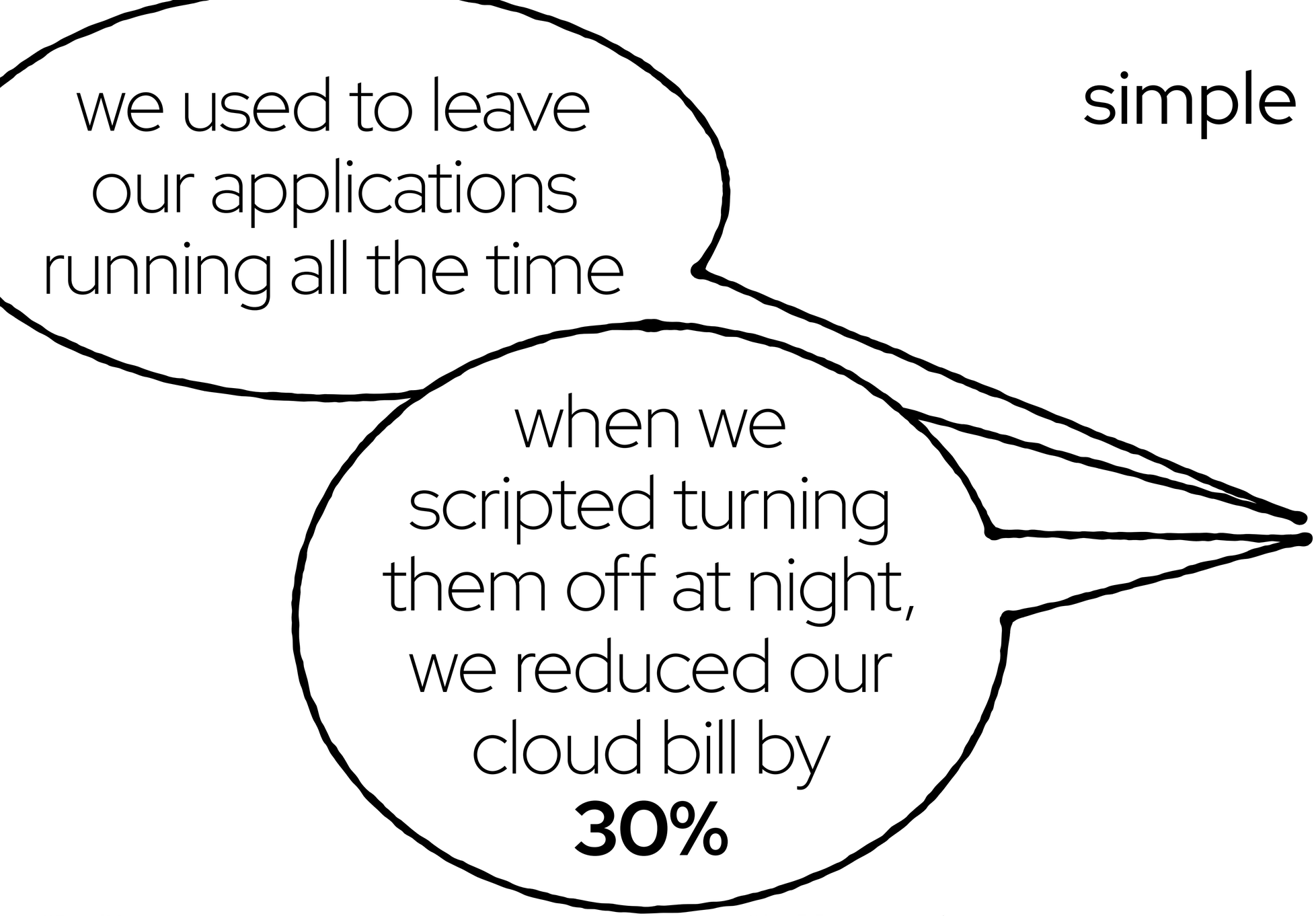




simple automation

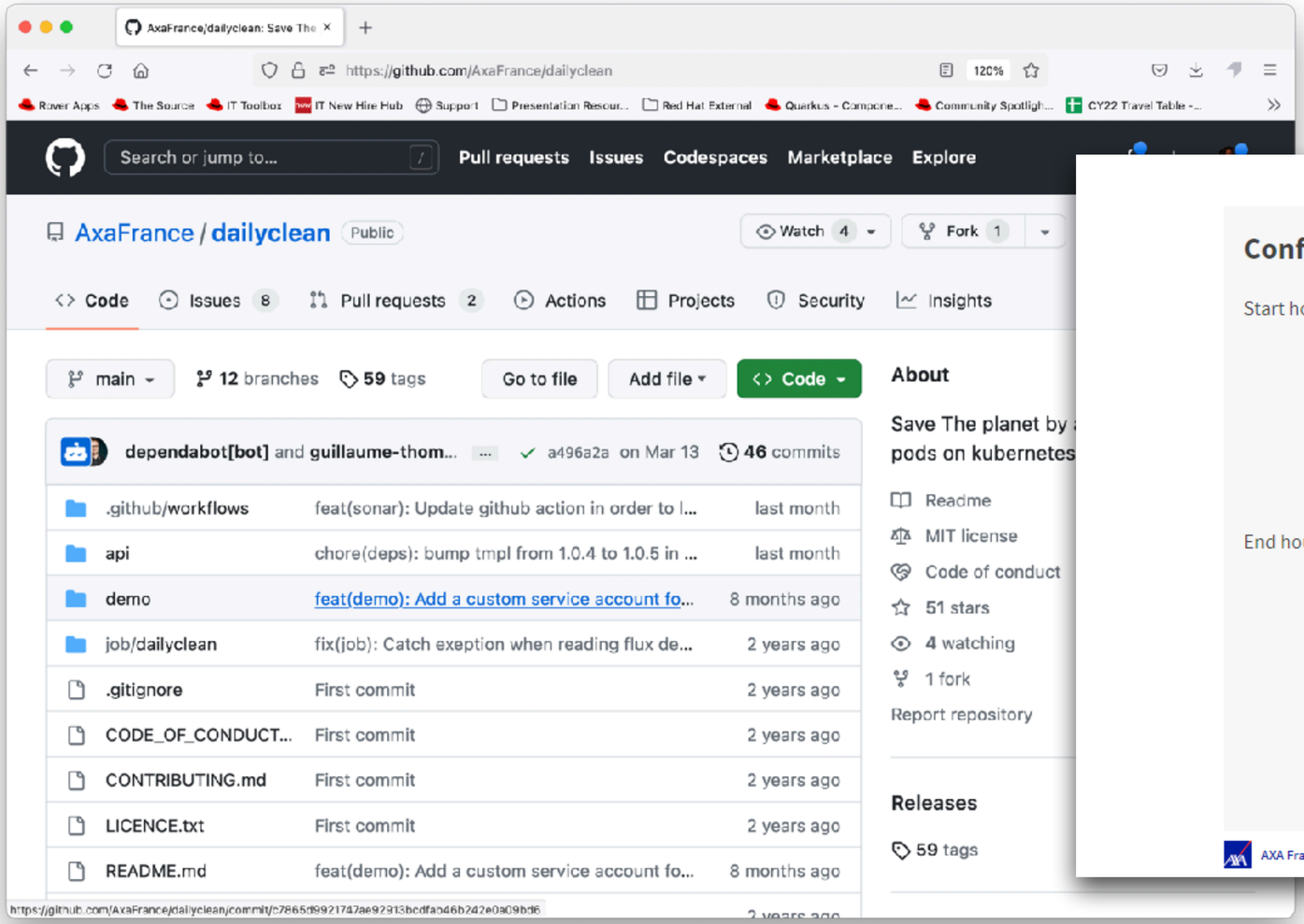
@darkandnerdy, Chicago DevOpsDays

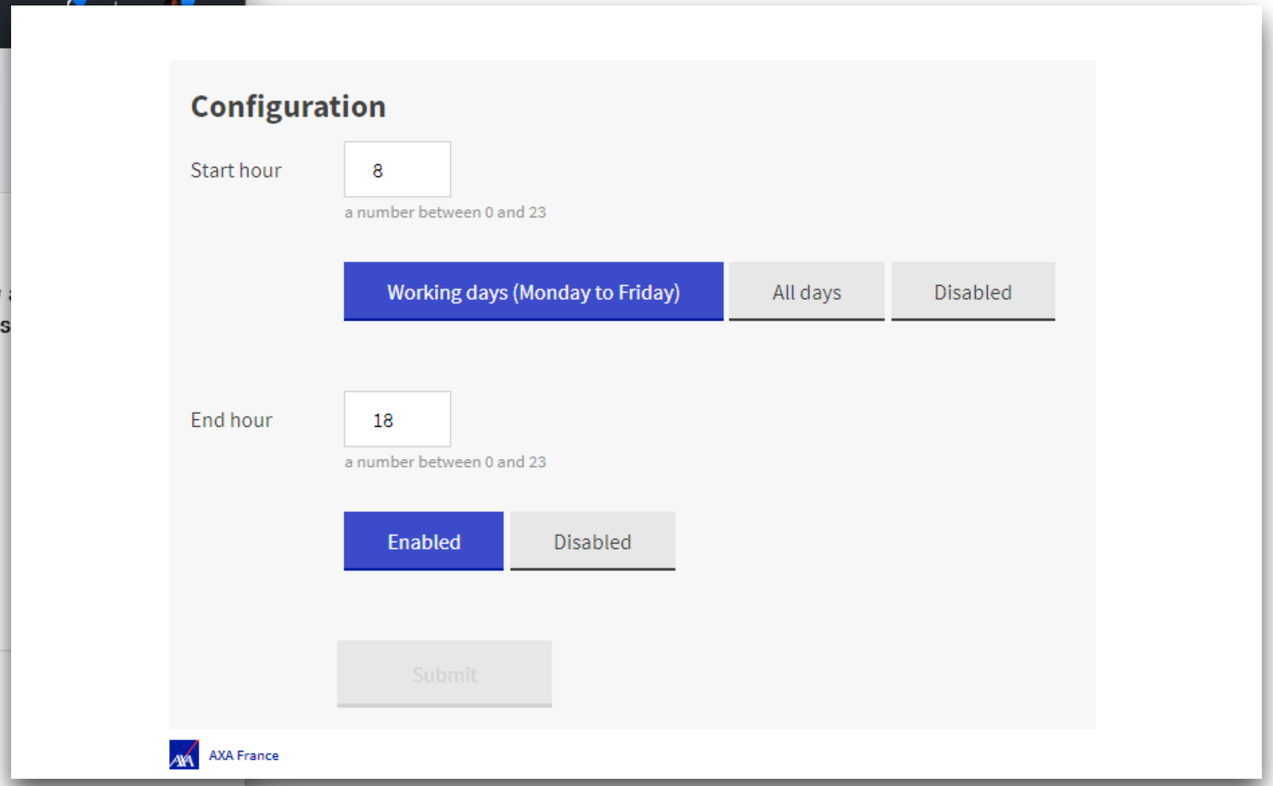
simple automation



@darkandnerdy, Chicago DevOpsDays

fancier scripts ...
with a frontend and a backend

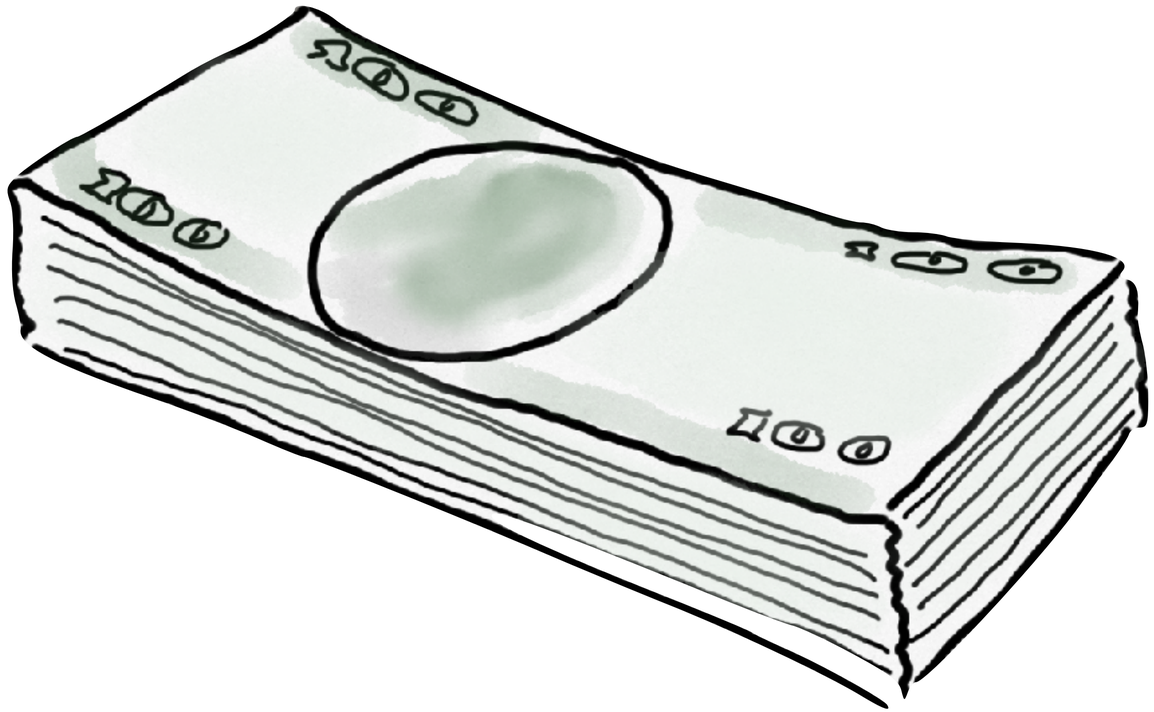




saves energy

saves energy

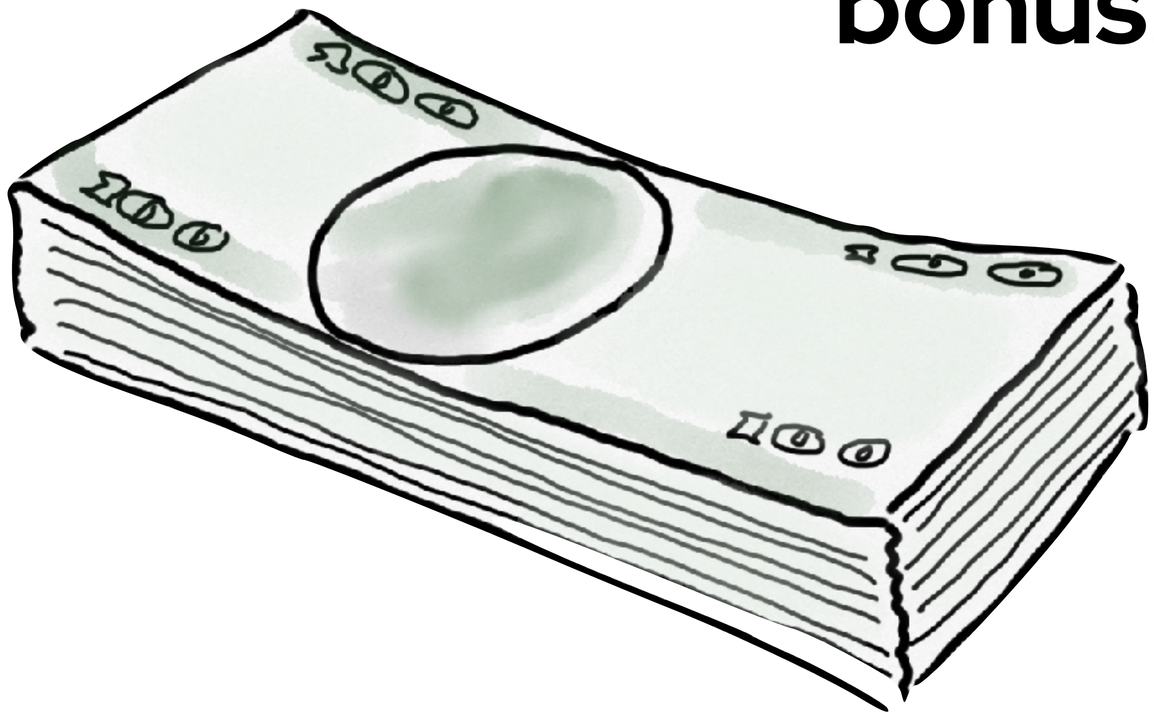
saves money



saves energy

saves money

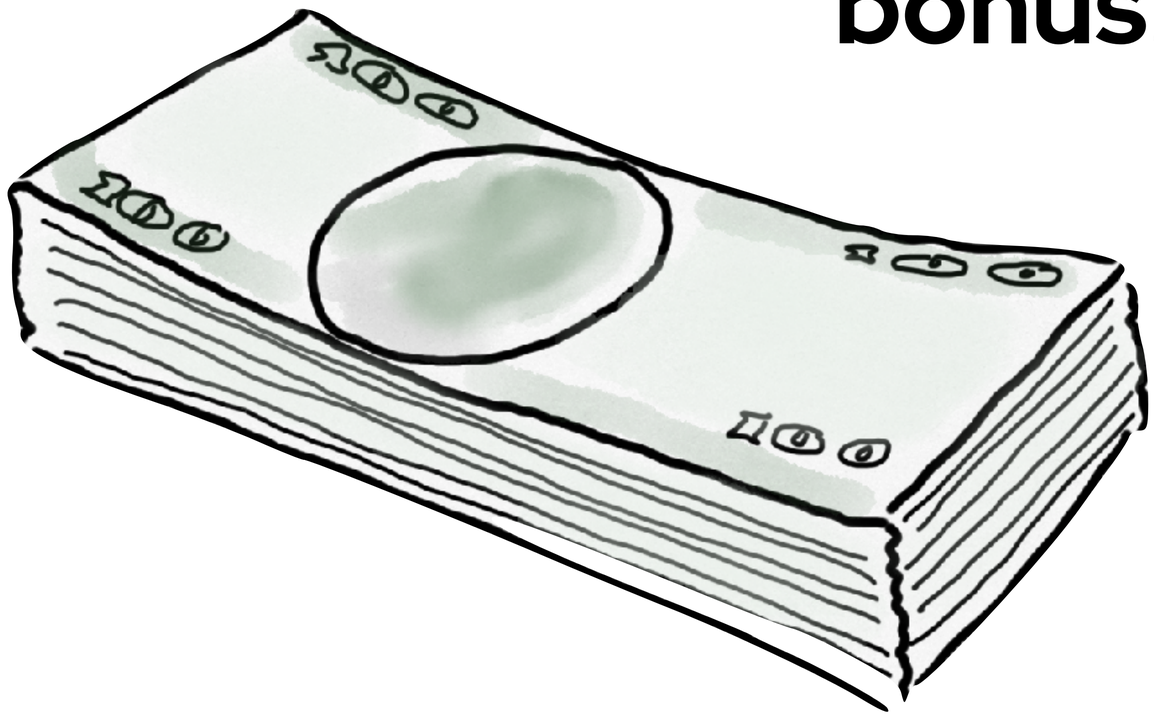
bonus: tests disaster recovery



saves energy

saves money

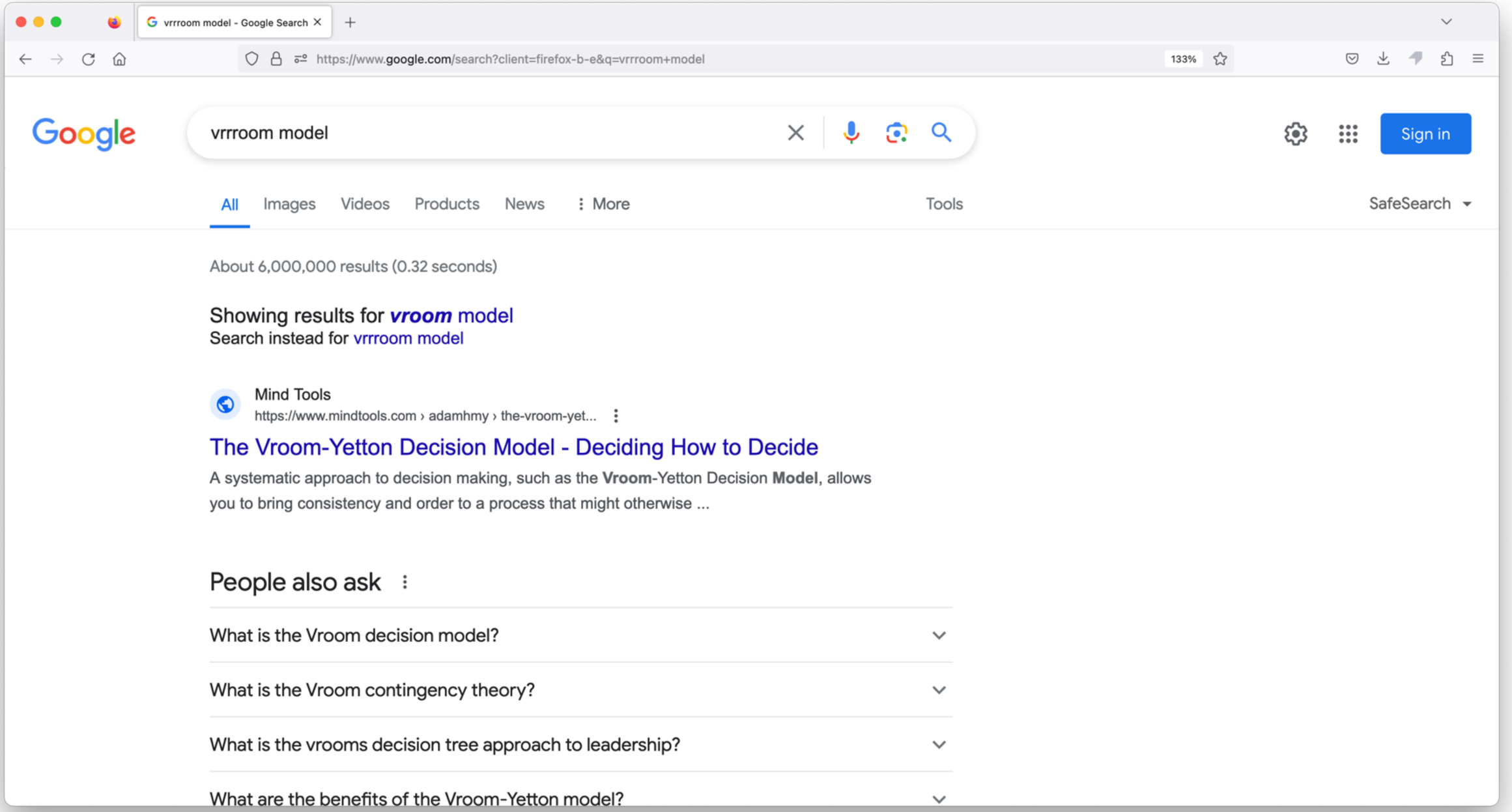
bonus: tests disaster recovery

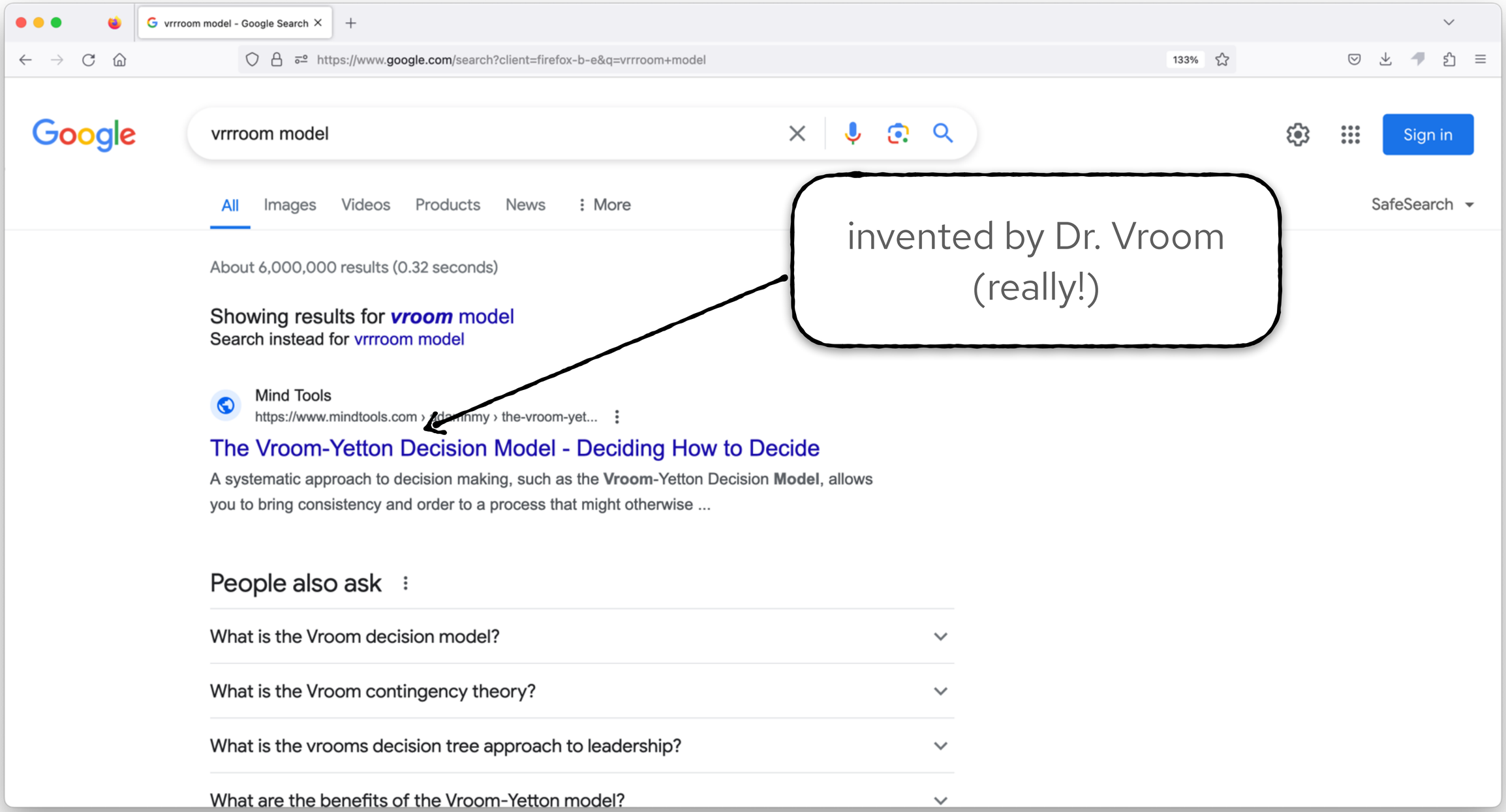


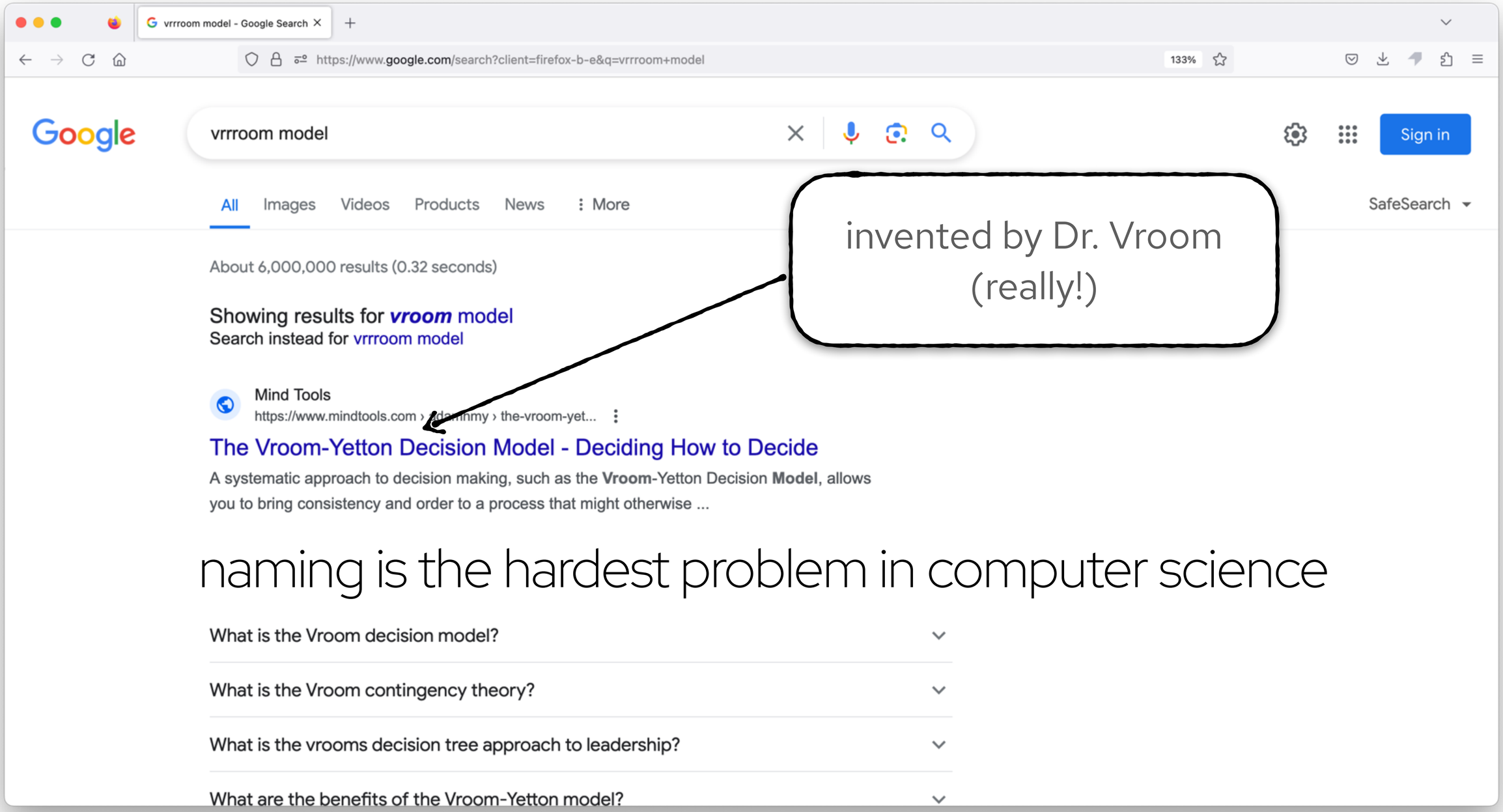
solution:

faster code

the vrroooom model

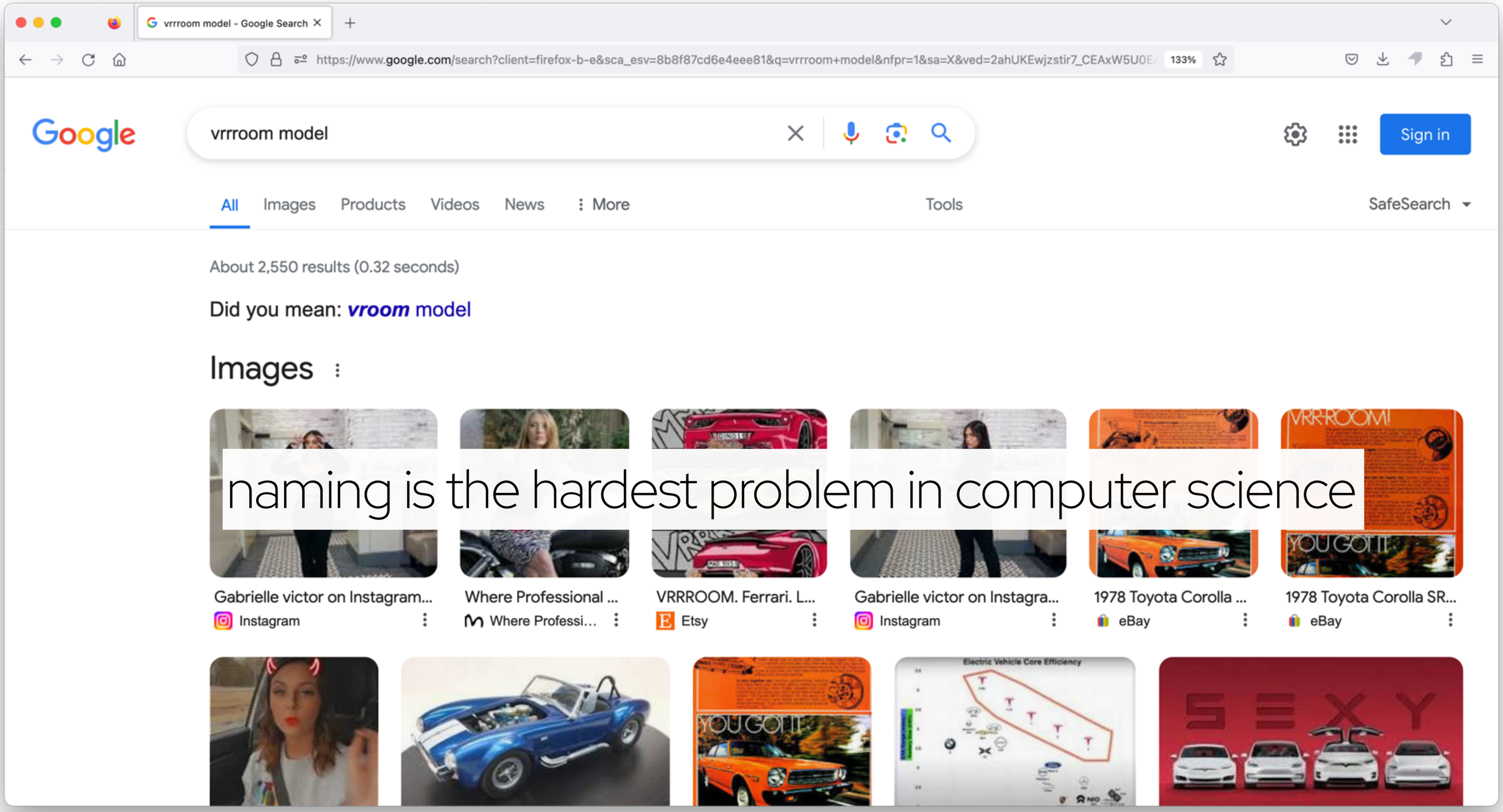

vrrroom model - Google Search
https://www.google.com/search?client=firefox-b-e&q=vrrroom+model
133%
Google
vrrroom model
Sign in
All
Images
Videos
Products
News
More
Tools
SafeSearch
About 6,000,000 results (0.32 seconds)
Showing results for vroom model
Search instead for vrrroom model
Mind Tools
https://www.mindtools.com › adamhmy › the-vroom-yet...
The Vroom-Yetton Decision Model - Deciding How to Decide
A systematic approach to decision making, such as the Vroom-Yetton Decision Model, allows you to bring consistency and order to a process that might otherwise ...
People also ask
What is the Vroom decision model?
What is the Vroom contingency theory?
What is the vrooms decision tree approach to leadership?
What are the benefits of the Vroom-Yetton model?


vrrroom model - Google Search
https://www.google.com/search?client=firefox-b-e&q=vrrroom+model
133%
Google
vrrroom model
Sign in
All
Images
Videos
Products
News
More
SafeSearch
About 6,000,000 results (0.32 seconds)
Showing results for vroom model
Search instead for vrrroom model
Mind Tools
https://www.mindtools.com › ... › the-vroom-yet...
The Vroom-Yetton Decision Model - Deciding How to Decide
A systematic approach to decision making, such as the Vroom-Yetton Decision Model, allows you to bring consistency and order to a process that might otherwise ...
People also ask
What is the Vroom decision model?
What is the Vroom contingency theory?
What is the vrooms decision tree approach to leadership?
What are the benefits of the Vroom-Yetton model?
invented by Dr. Vroom (really!)



invented by Dr. Vroom (really!)

naming is the hardest problem in computer science



naming is the hardest problem in computer science

@holly_cummins

my vrroooom model

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.

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-

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

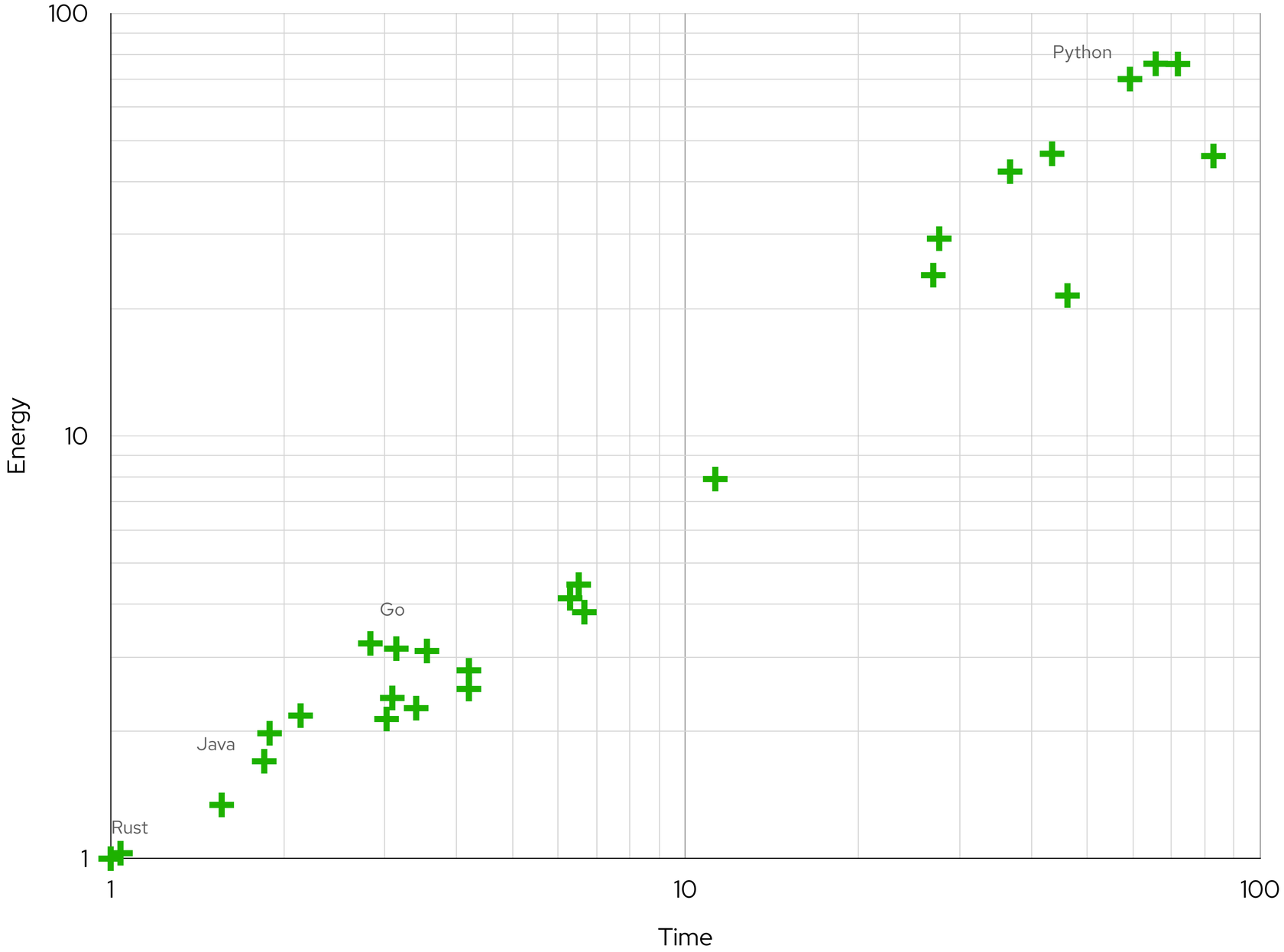
Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(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 efficiency across programming languages

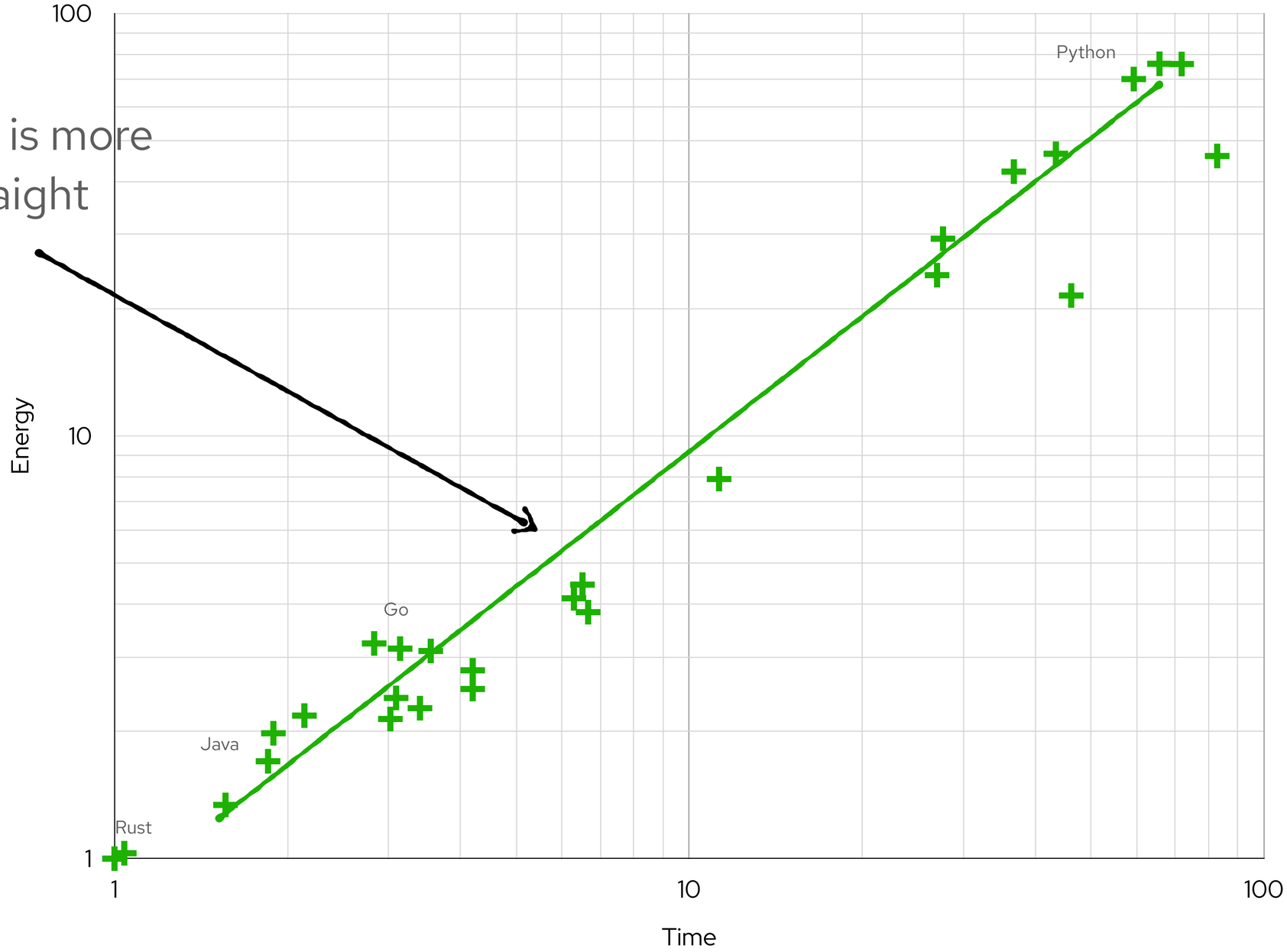


#RedHat

@holly_cummins

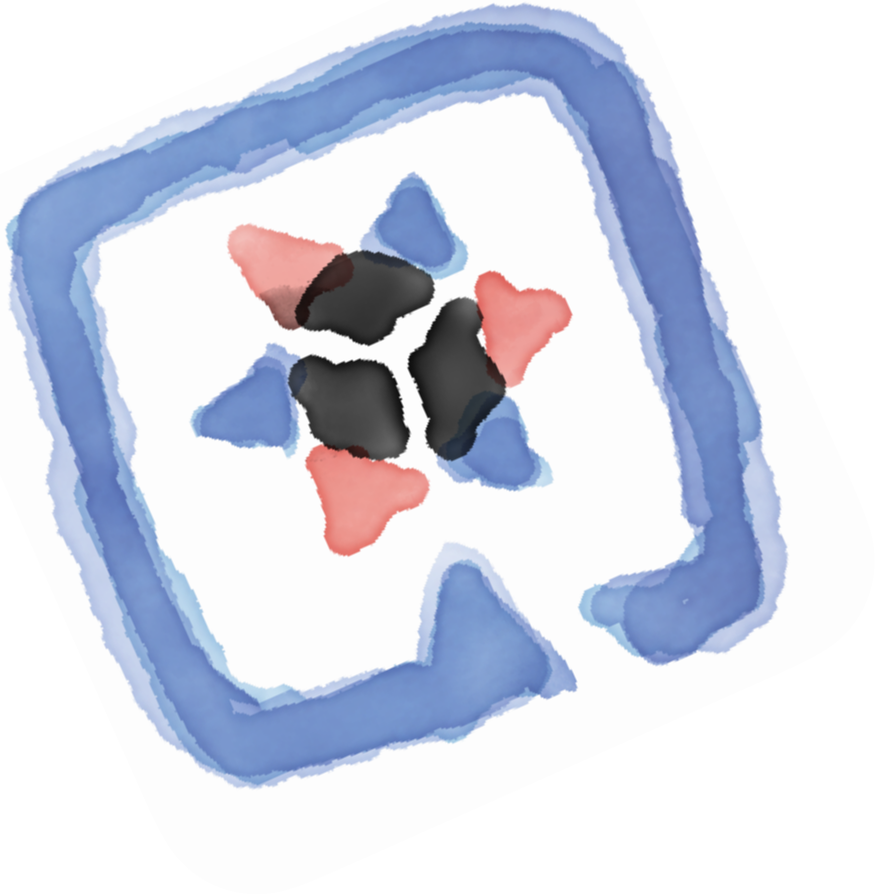
energy efficiency across programming languages

the trend line is more or less straight



case study

Quarkus



Java application frameworks were optimised for dynamism

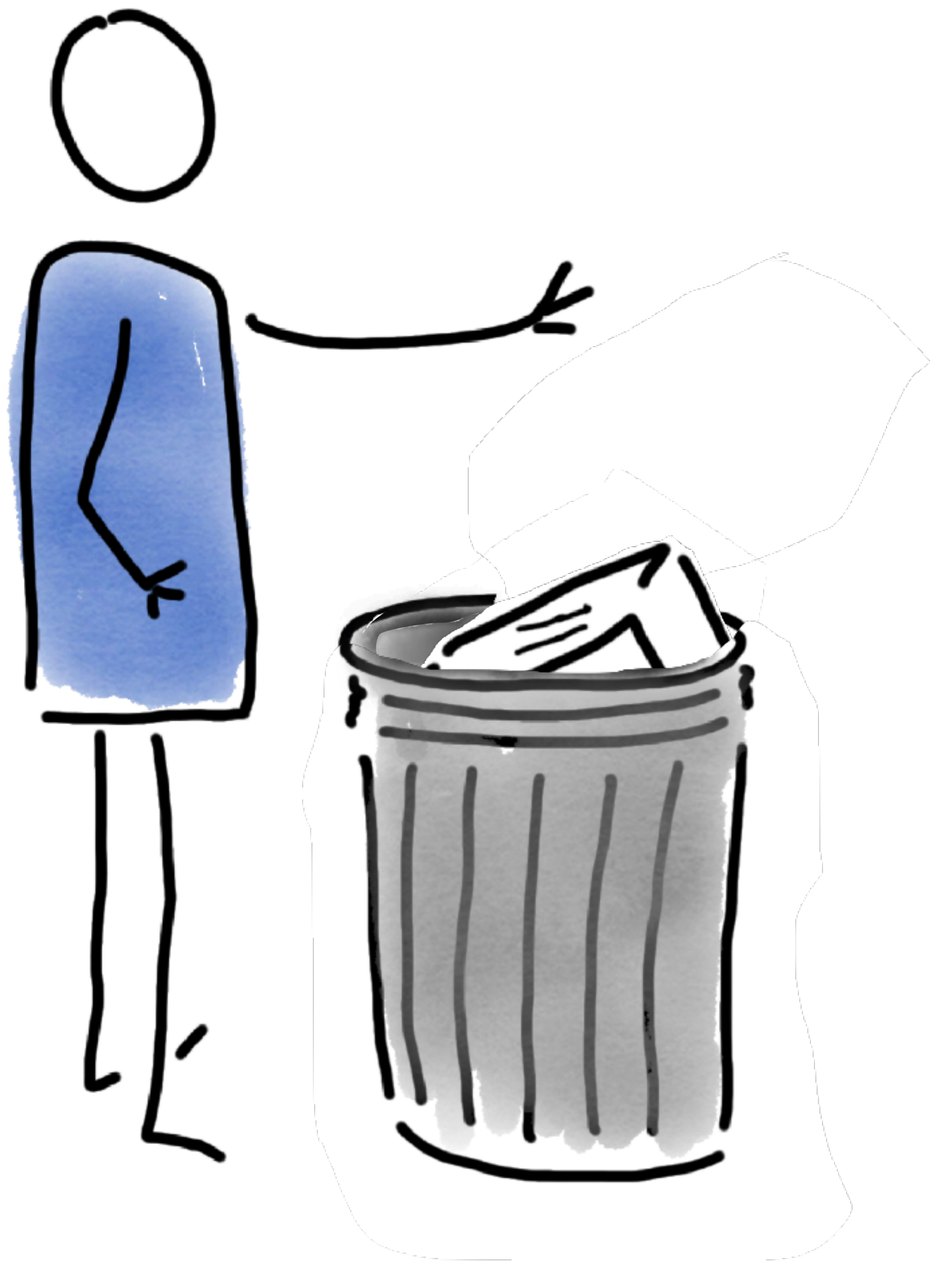
Java application frameworks were optimised for dynamism

dynamism has a **cost**



now

cloud apps are
immutable



now

cloud apps are
immutable

a highly dynamic
runtime in a container
is **pointless**



Java dynamism

build time

Java dynamism

build time *runtime*

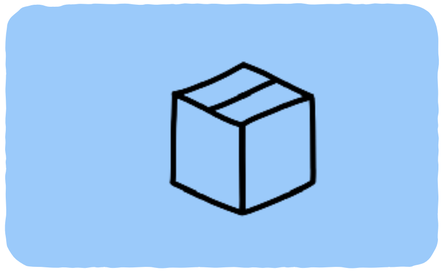
Java dynamism

build time *runtime*



Java dynamism

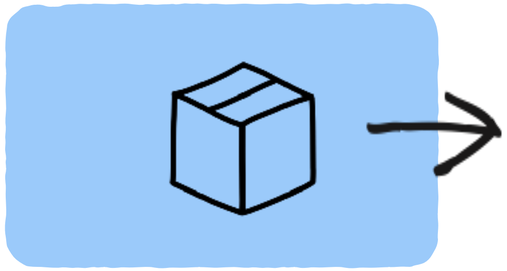
build time *runtime*



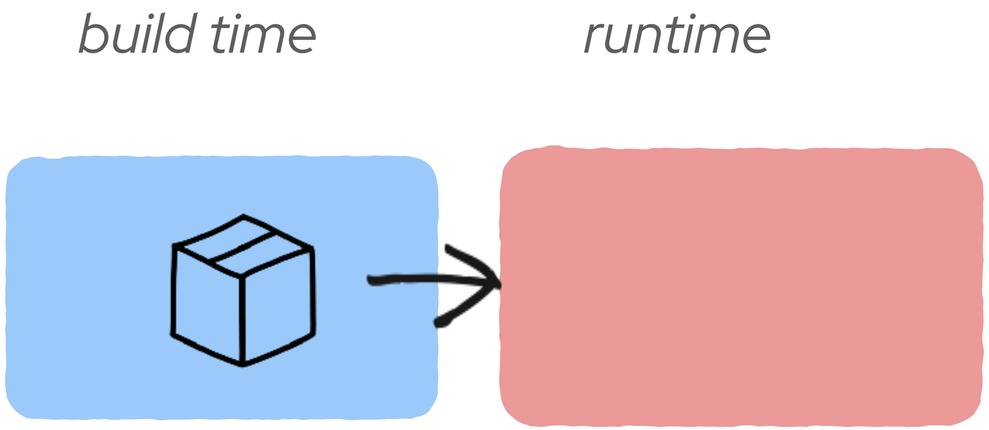
packaging
(maven, gradle...)

Java dynamism

build time *runtime*

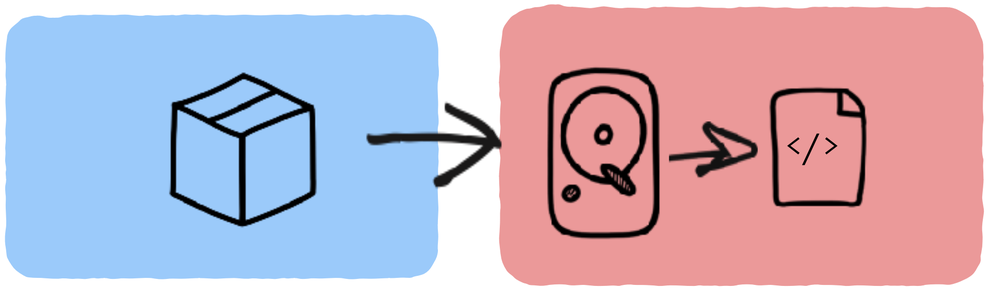


Java dynamism



Java dynamism

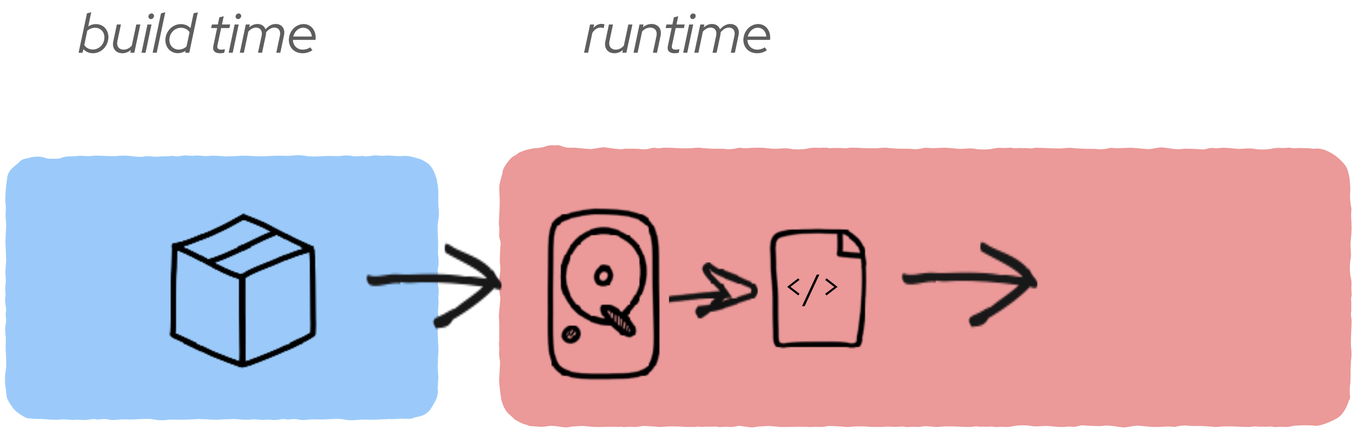
build time *runtime*



load and parse

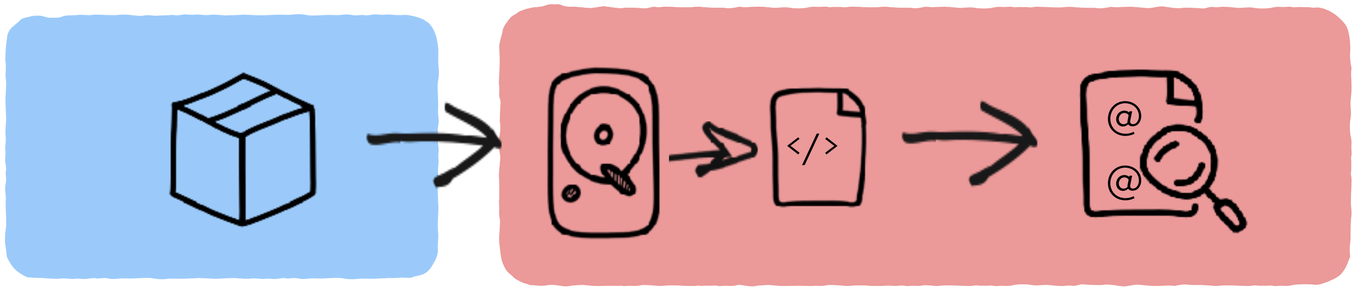
- config files
- properties
- yaml
- xml
- etc.

Java dynamism



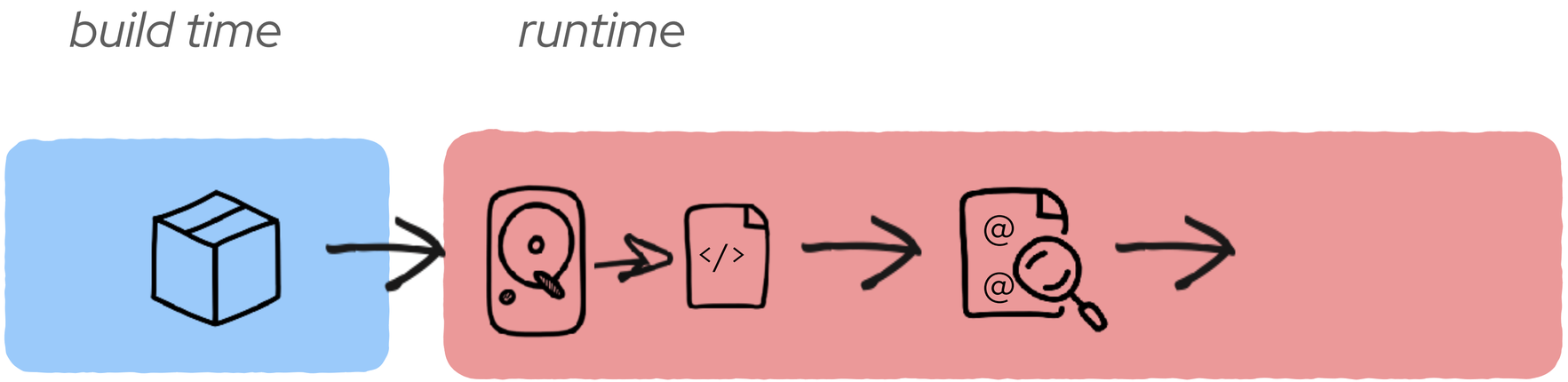
Java dynamism

build time *runtime*



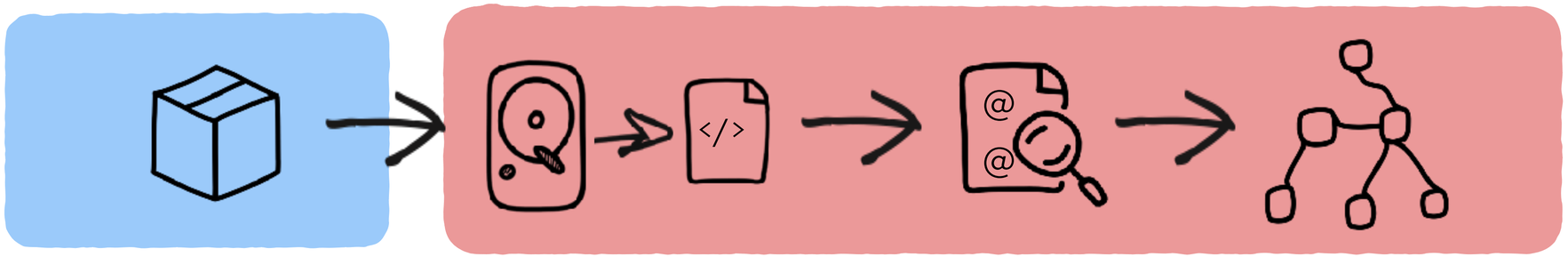
- classpath scanning and annotation discovery
- attempt to load class to enable/disable features

Java dynamism



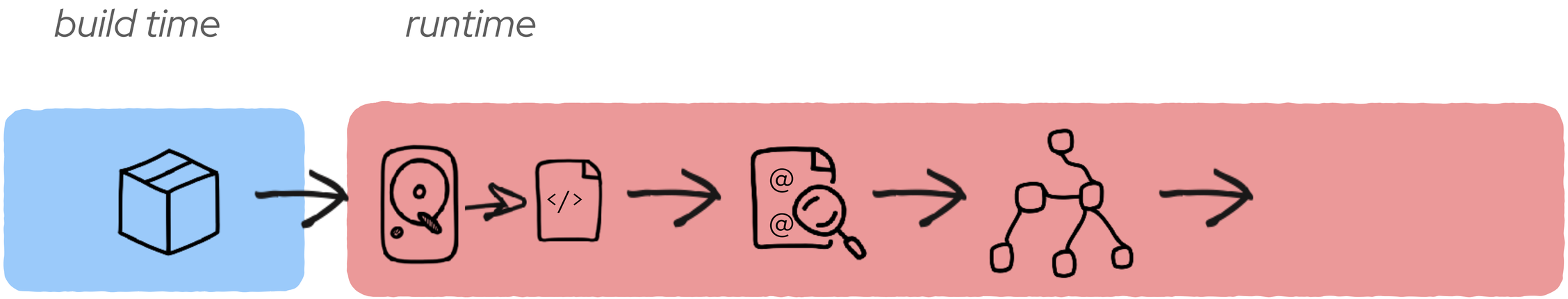
Java dynamism

build time *runtime*



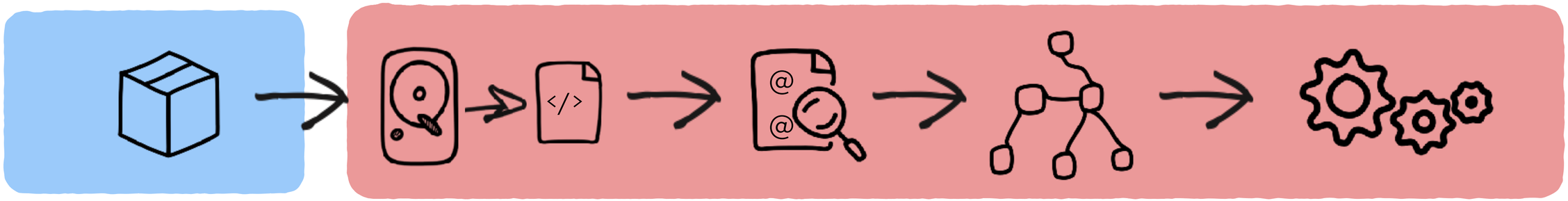
build a metamodel
of the world

Java dynamism



Java dynamism

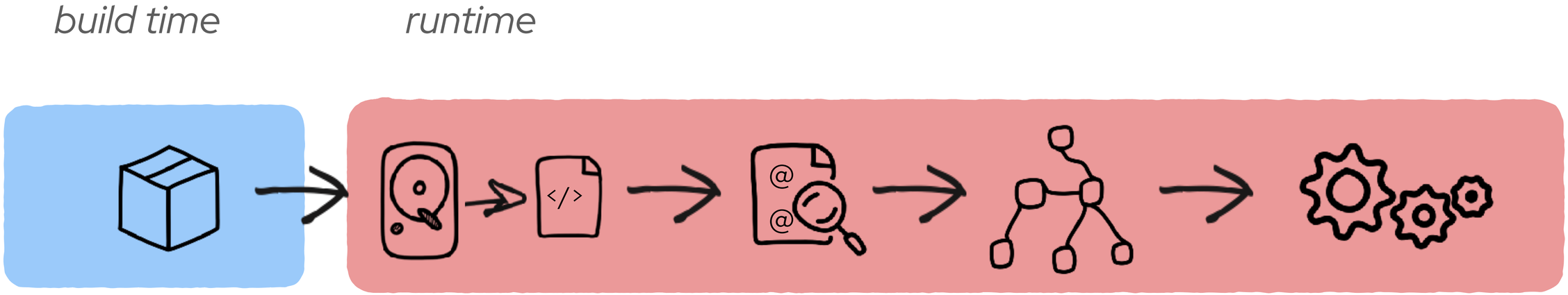
build time *runtime*



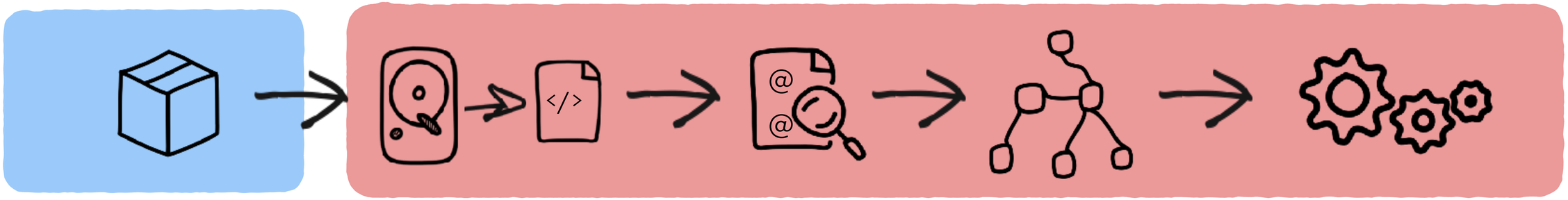
start

- thread pools
- I/O
- etc.

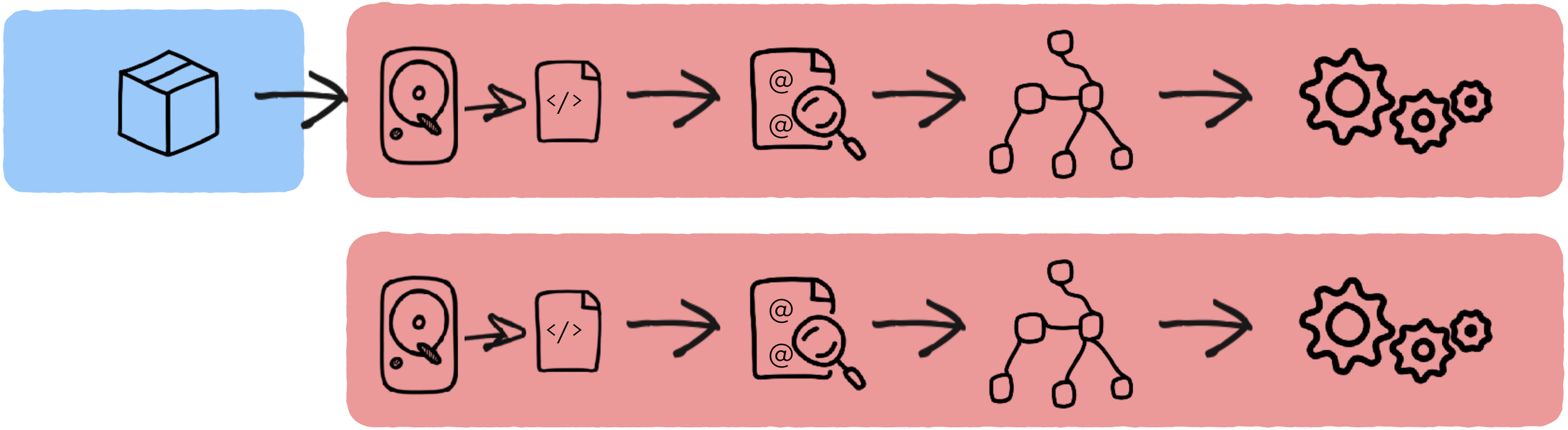
Java dynamism



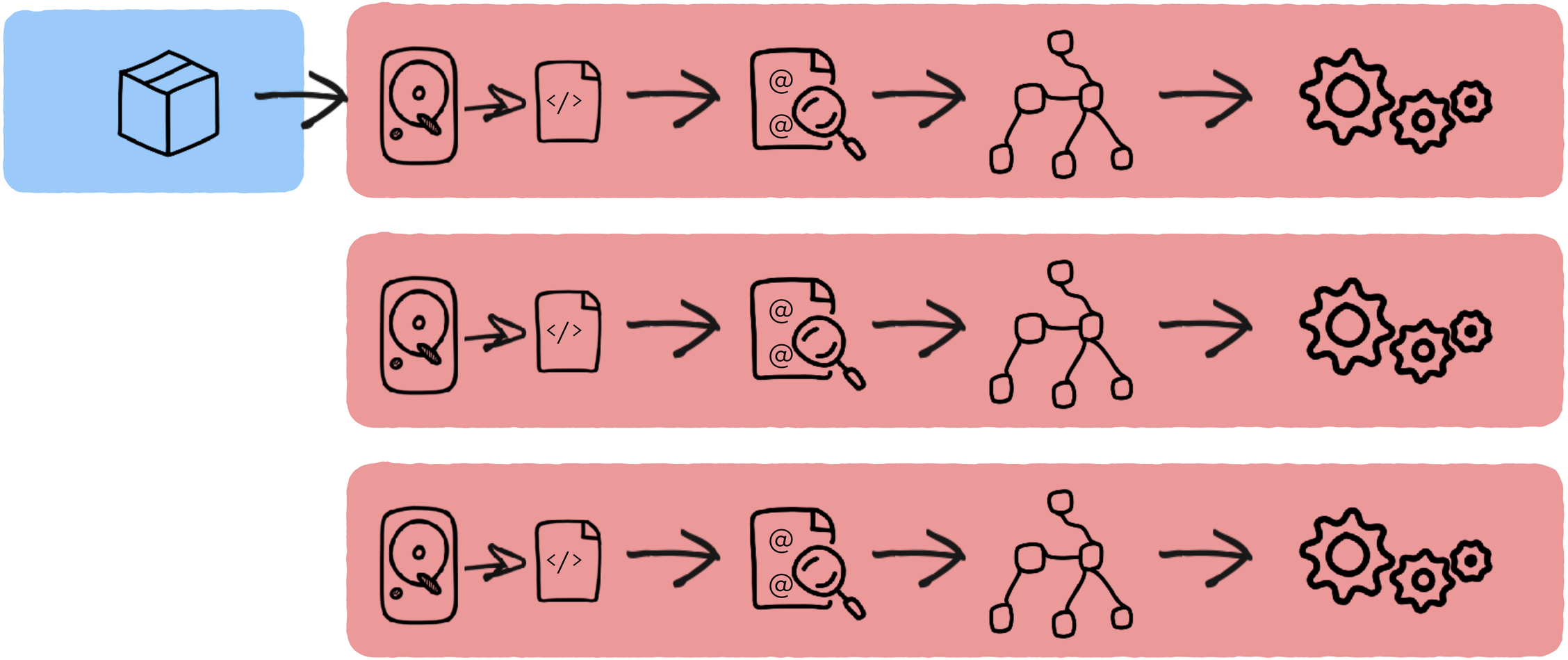
Java dynamism



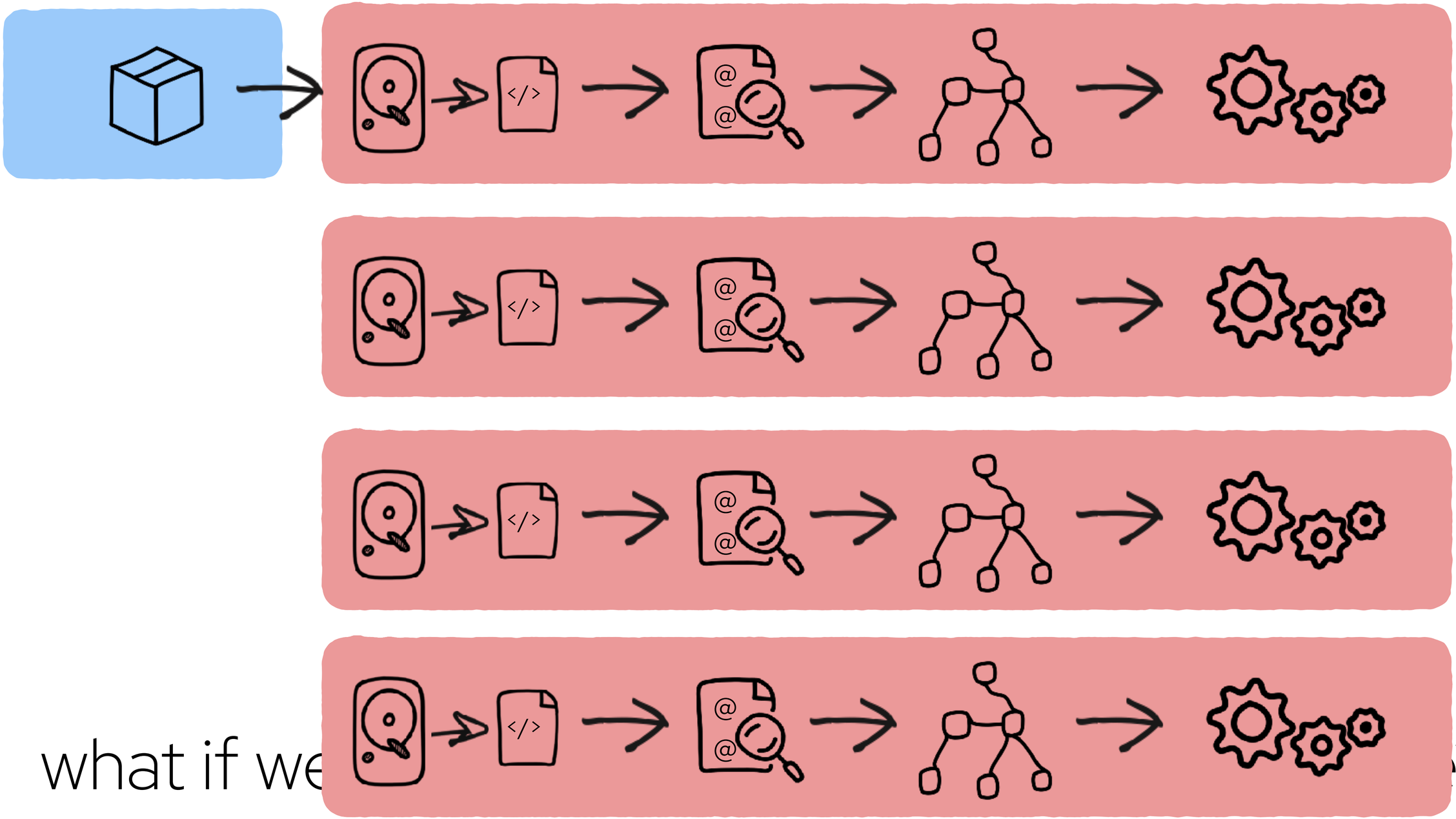
what if we start the application more than once?



what if we start the application more than once?

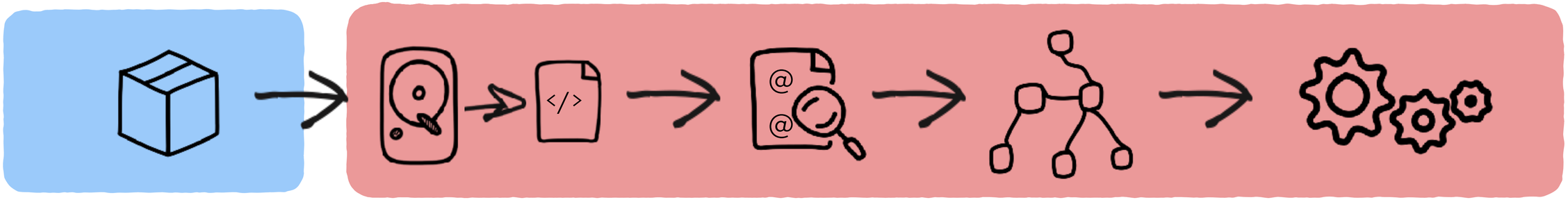


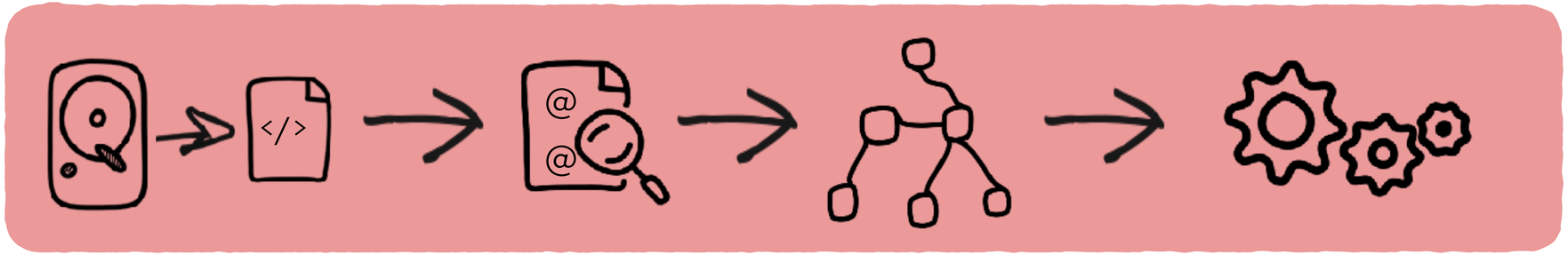
what if we start the application more than once?

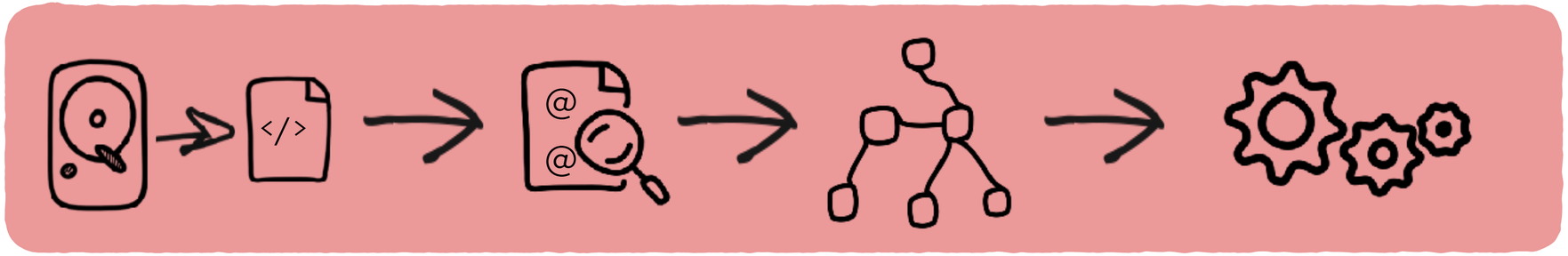


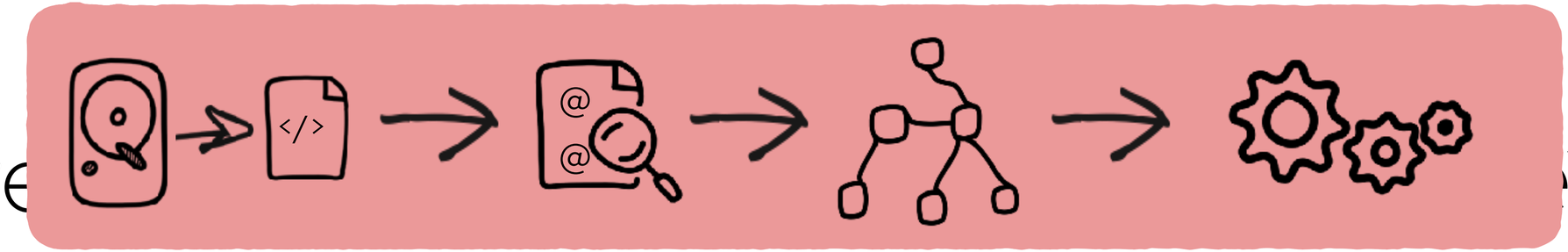
what if we

?



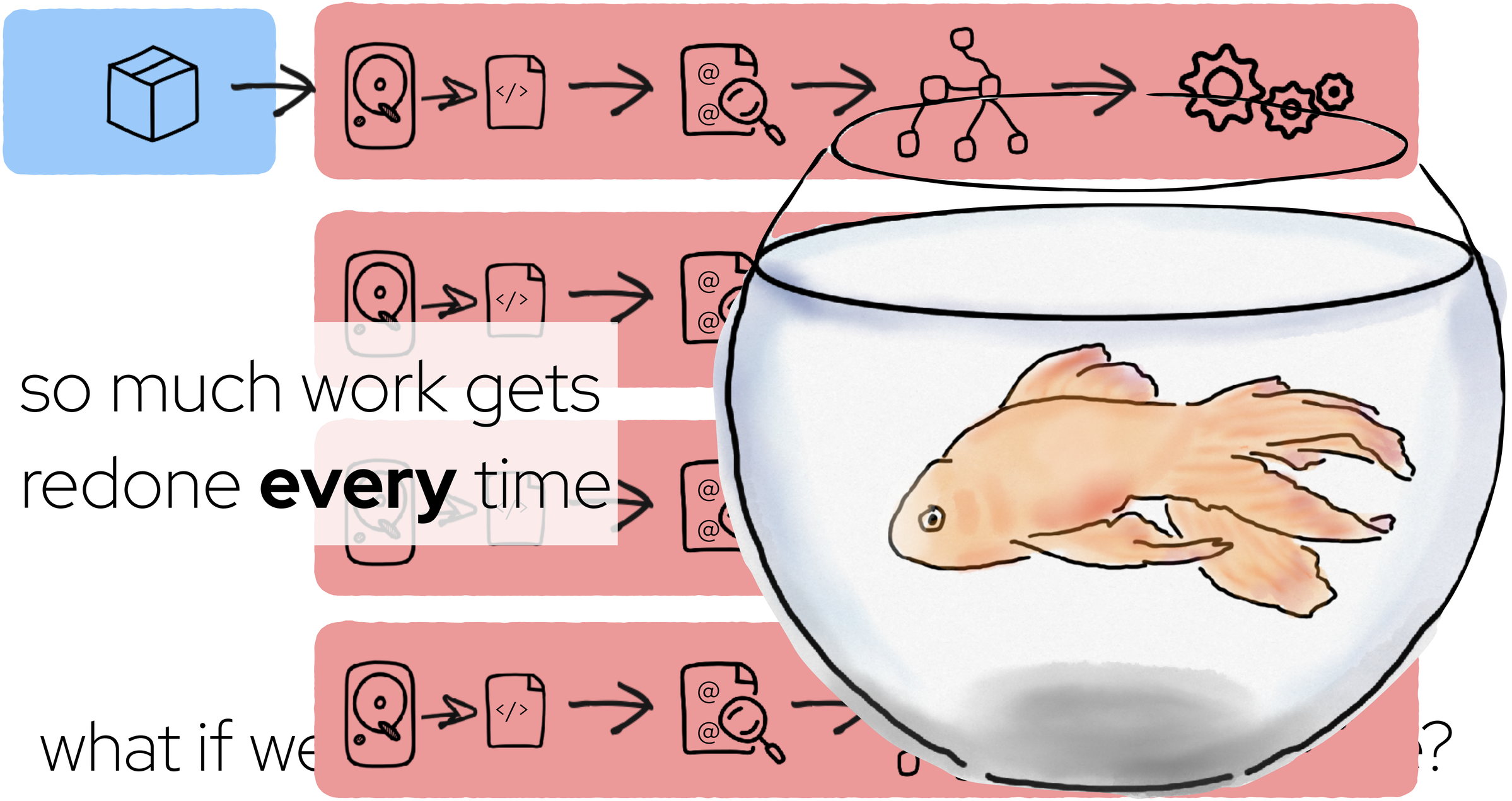






what if we

?



so much work gets redone **every** time

what if we

?

speed example: JTA auto-wiring

speed example: JTA auto-wiring

```
Class.forName(“LikelyJTAImplementation”);
```



speed example: JTA auto-wiring

```
Class.forName(“LikelyJTAImplementation”);
```

speed example: JTA auto-wiring

```
Class.forName(“LikelyJTAImplementation”);
Class.forName(“APossibleJTAImplementation”);
```



speed example: JTA auto-wiring

```
Class.forName(“LikelyJTAImplementation”);
Class.forName(“APossibleJTAImplementation”);
Class.forName(“AnotherJTAImplementation”);
```



speed example: JTA auto-wiring

```
Class.forName(“LikelyJTAImplementation”);
Class.forName(“APossibleJTAImplementation”);
Class.forName(“AnotherJTAImplementation”);
```

…



speed example: JTA auto-wiring

```
Class.forName(“LikelyJTAImplementation”);
Class.forName(“APossibleJTAImplementation”);
Class.forName(“AnotherJTAImplementation”);

 …

Class.forName(“NicheJTAImplementation”);
```



speed example: JTA auto-wiring

```
Class.forName(“LikelyJTAImplementation”);
Class.forName(“APossibleJTAImplementation”);
Class.forName(“AnotherJTAImplementation”);

 …

Class.forName(“NicheJTAImplementation”);
Class.forName(“VeryNicheJTAImplementation”);
```



speed example: JTA auto-wiring

```
Class.forName(“LikelyJTAImplementation”);
Class.forName(“APossibleJTAImplementation”);
Class.forName(“AnotherJTAImplementation”);

 …

Class.forName(“NicheJTAImplementation”);
Class.forName(“VeryNicheJTAImplementation”);
```



speed example: JTA auto-wiring

```
Class.forName("LikelyJTAImplementation");
Class.forName("APossibleJTAImplementation");
Class.forName("AnotherJTAImplementation");
...
Class.forName("NicheJTAImplementation");
Class.forName("VeryNicheJTAImplementation");
```

~129

auto-wiring attempts



speed example: JTA auto-wiring

```
Class.forName("LikelyJTAImplementation");
Class.forName("APossibleJTAImplementation");
Class.forName("AnotherJTAImplementation");
...
Class.forName("NicheJTAImplementation");
Class.forName("VeryNicheJTAImplementation");
```

~129

auto-wiring attempts

every single start.



it’s not just JTA

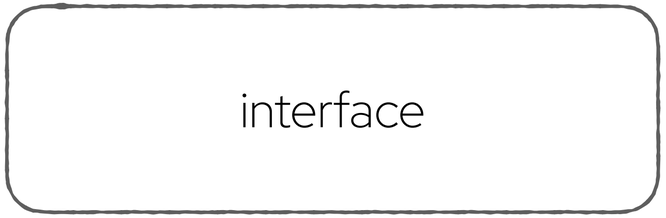
this happens for **lots** of internal service bindings

the true cost of loaded classes isn't just memory + start time

method dispatching:

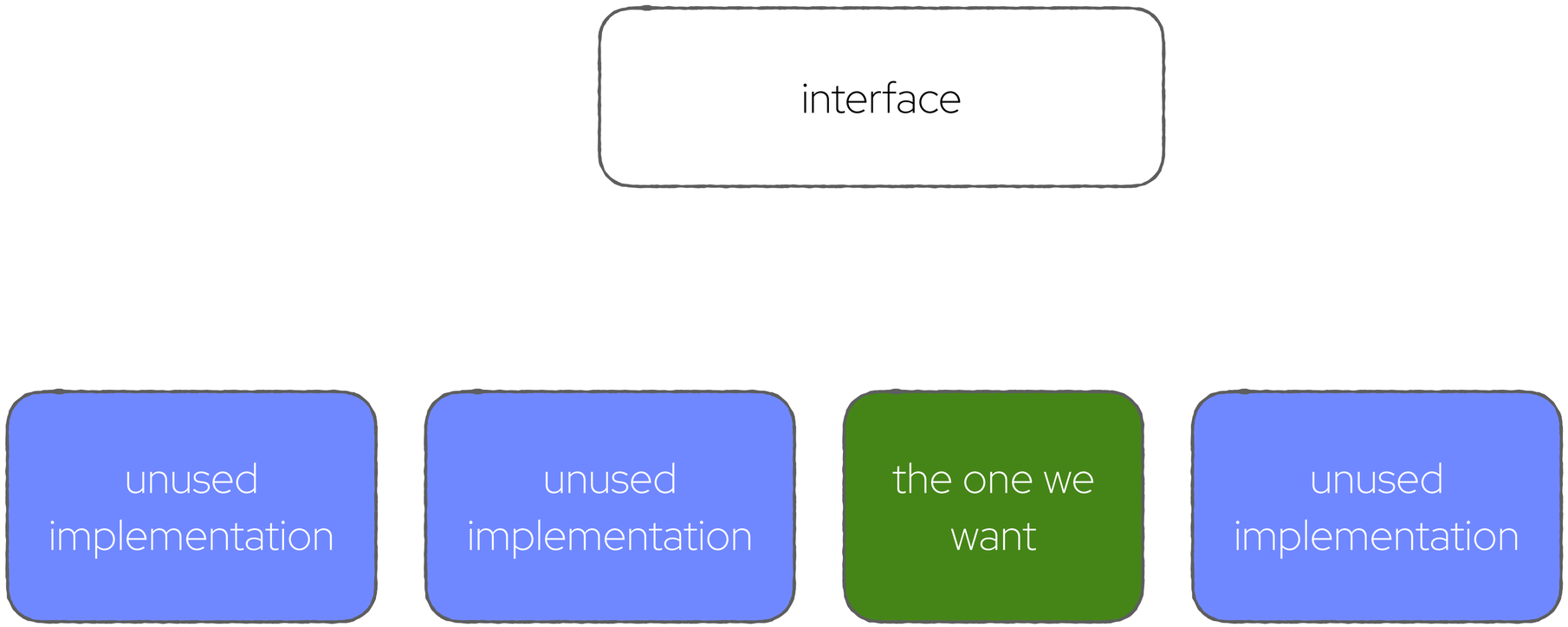
the true cost of loaded classes isn't just memory + start time

method dispatching:



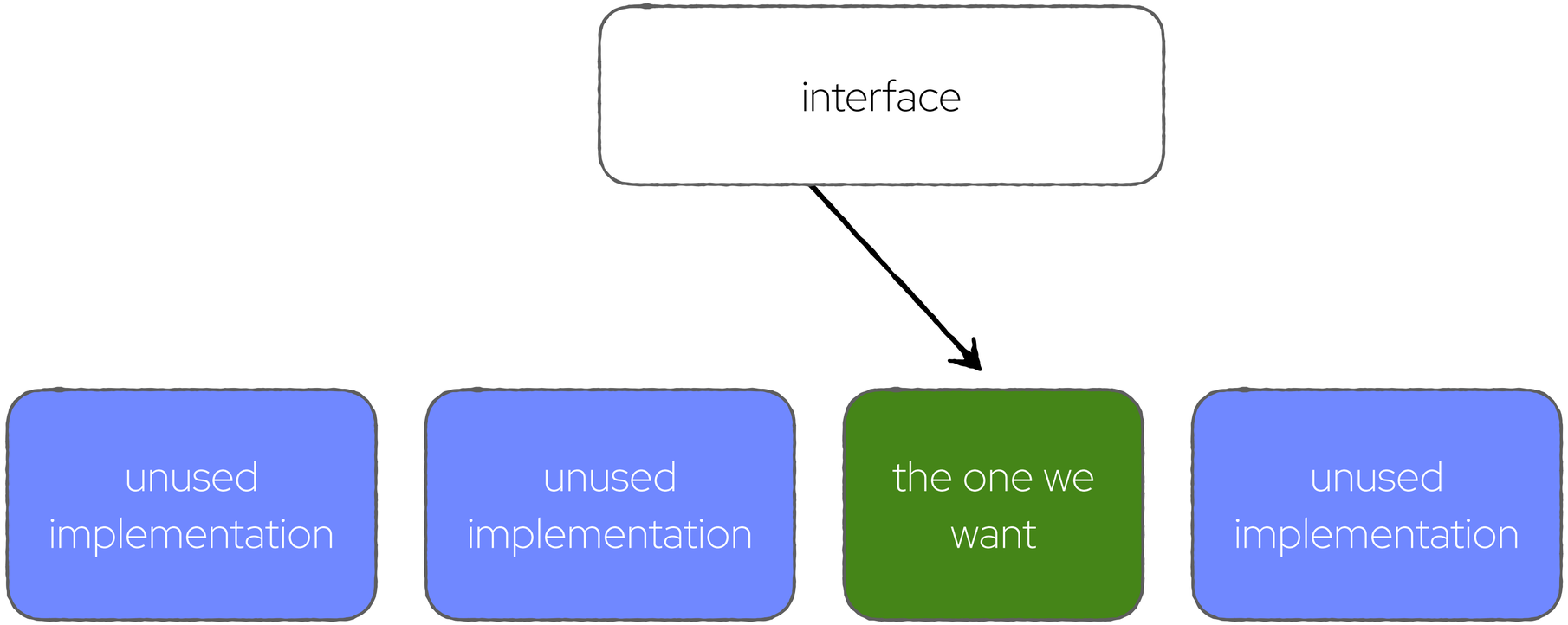
the true cost of loaded classes isn't just memory + start time

method dispatching:



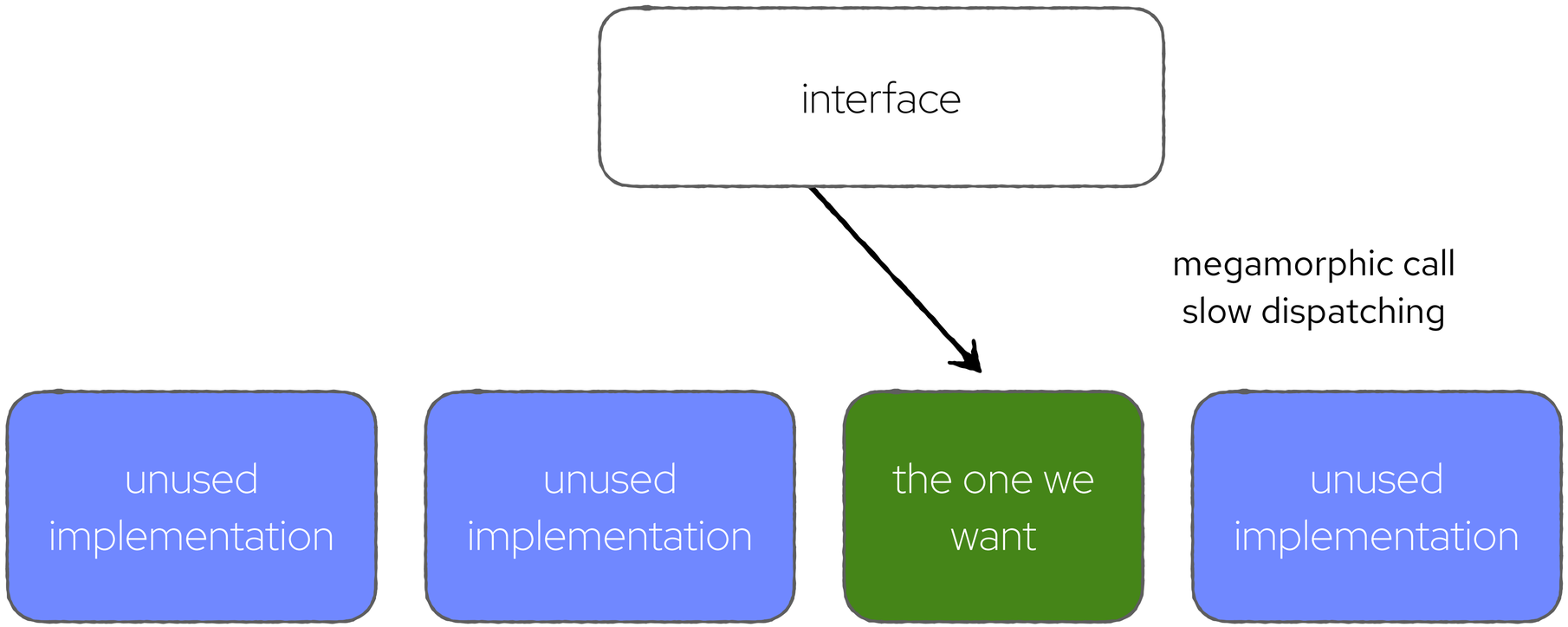
the true cost of loaded classes isn't just memory + start time

method dispatching:

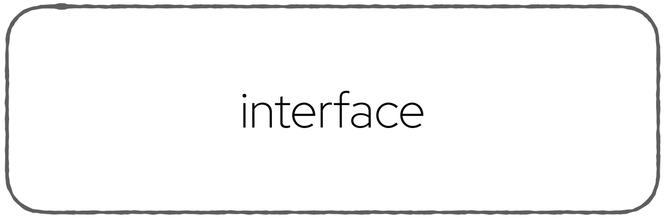


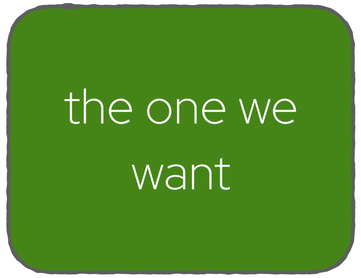
the true cost of loaded classes isn't just memory + start time

method dispatching:

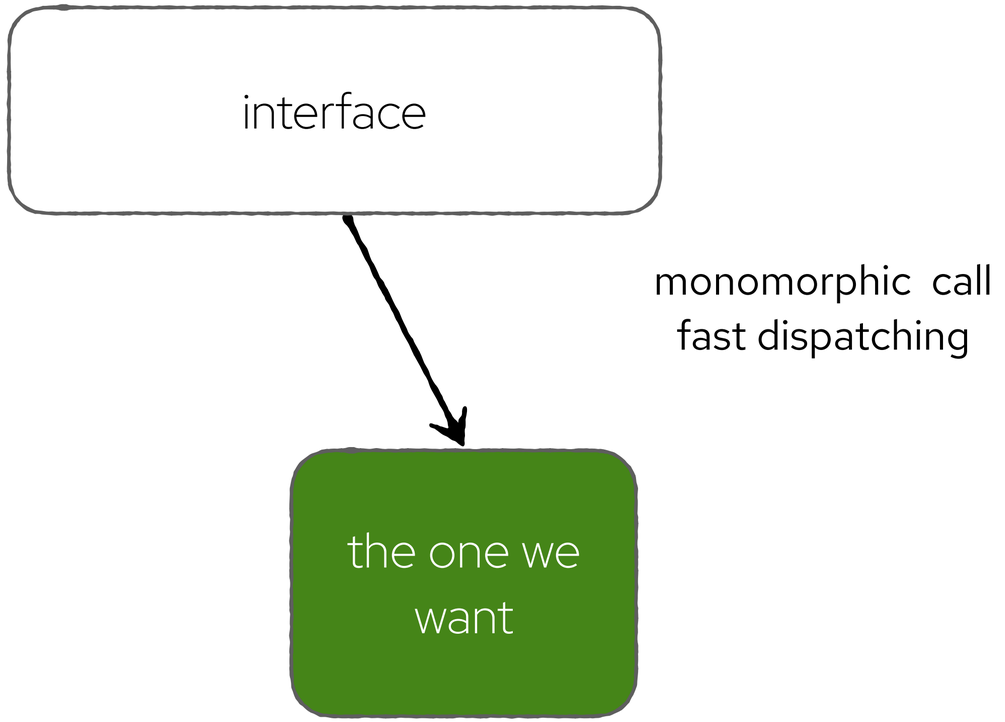


the true cost of loaded classes isn't just memory + start time





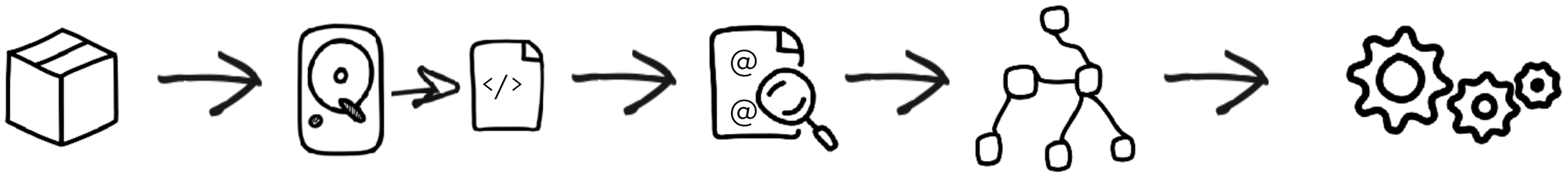
the true cost of loaded classes isn't just memory + start time



the true cost of loaded classes isn't just memory + start time

how do we fix all this?

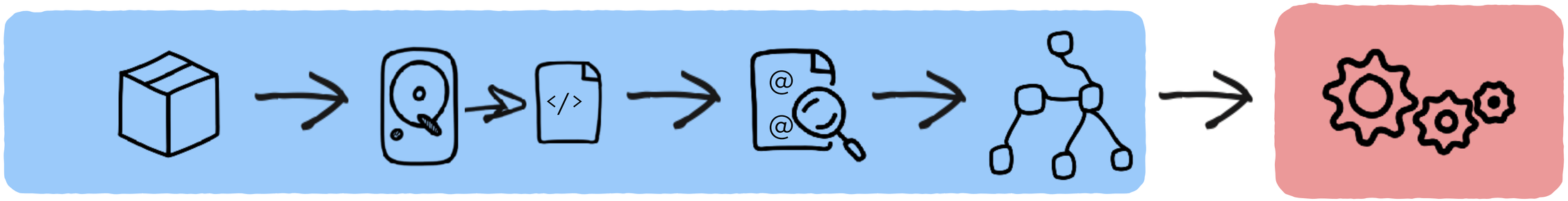
build time *runtime*



what if we initialize at build time?

build time

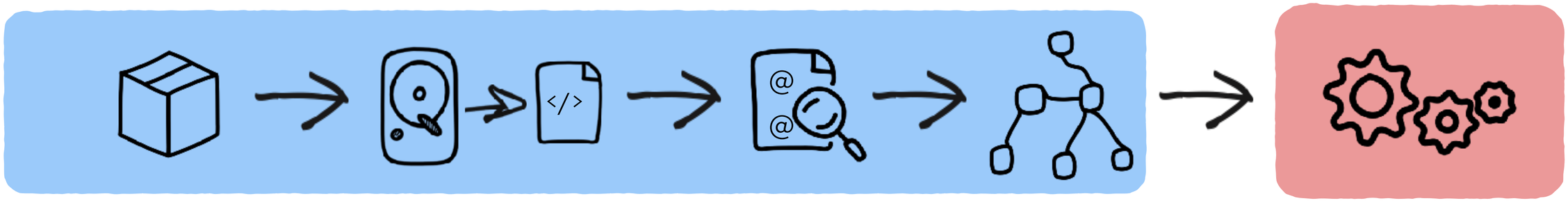
runtime



what if we initialize at build time?

build time

runtime



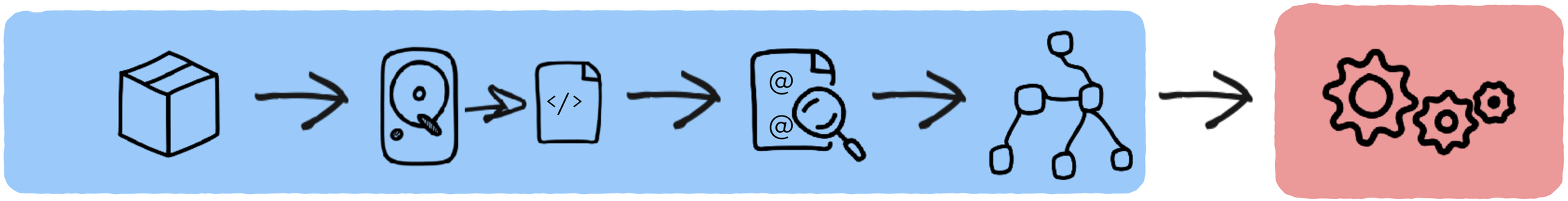
start

- thread pools
- I/O
- etc.

what if we initialize at build time?

build time

runtime

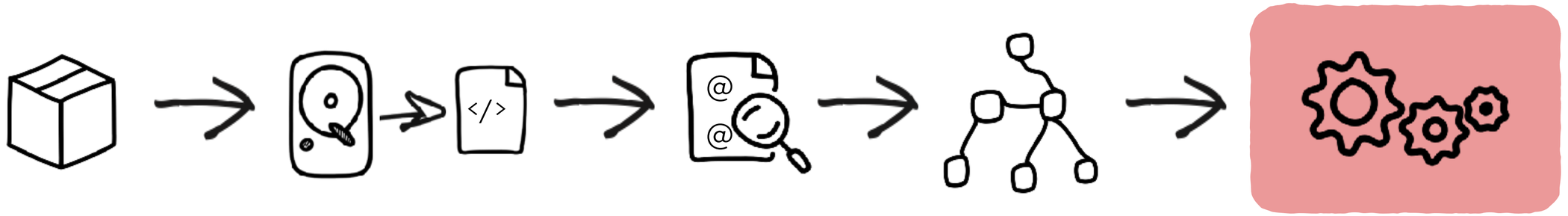


start

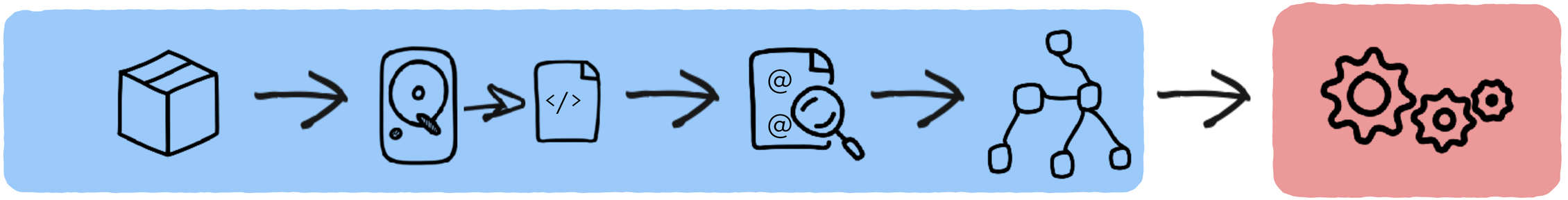
- thread pools
- I/O
- etc.

ready to do work!

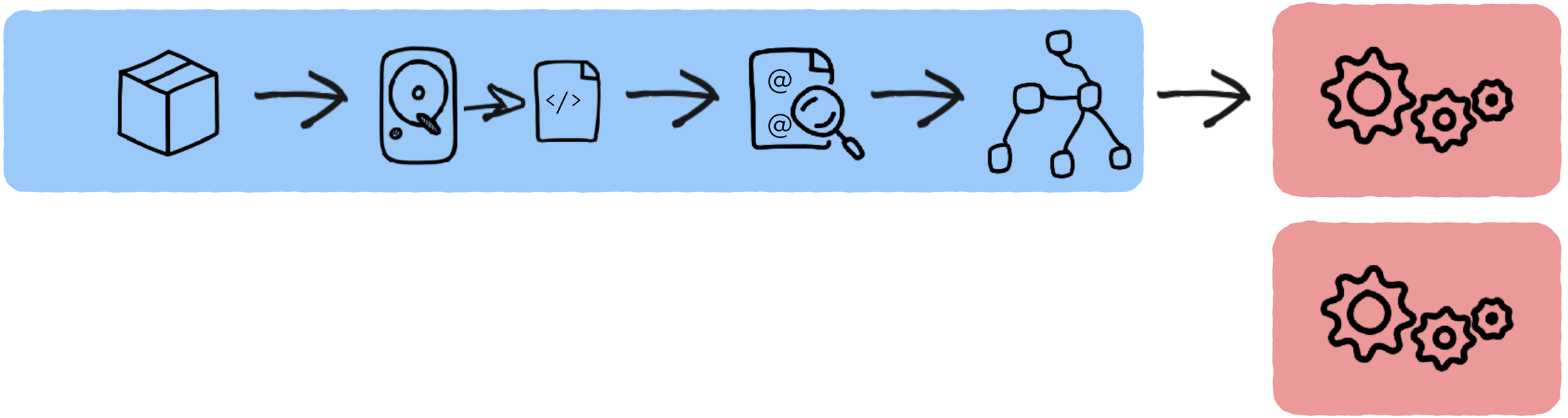
what if we initialize at build time?



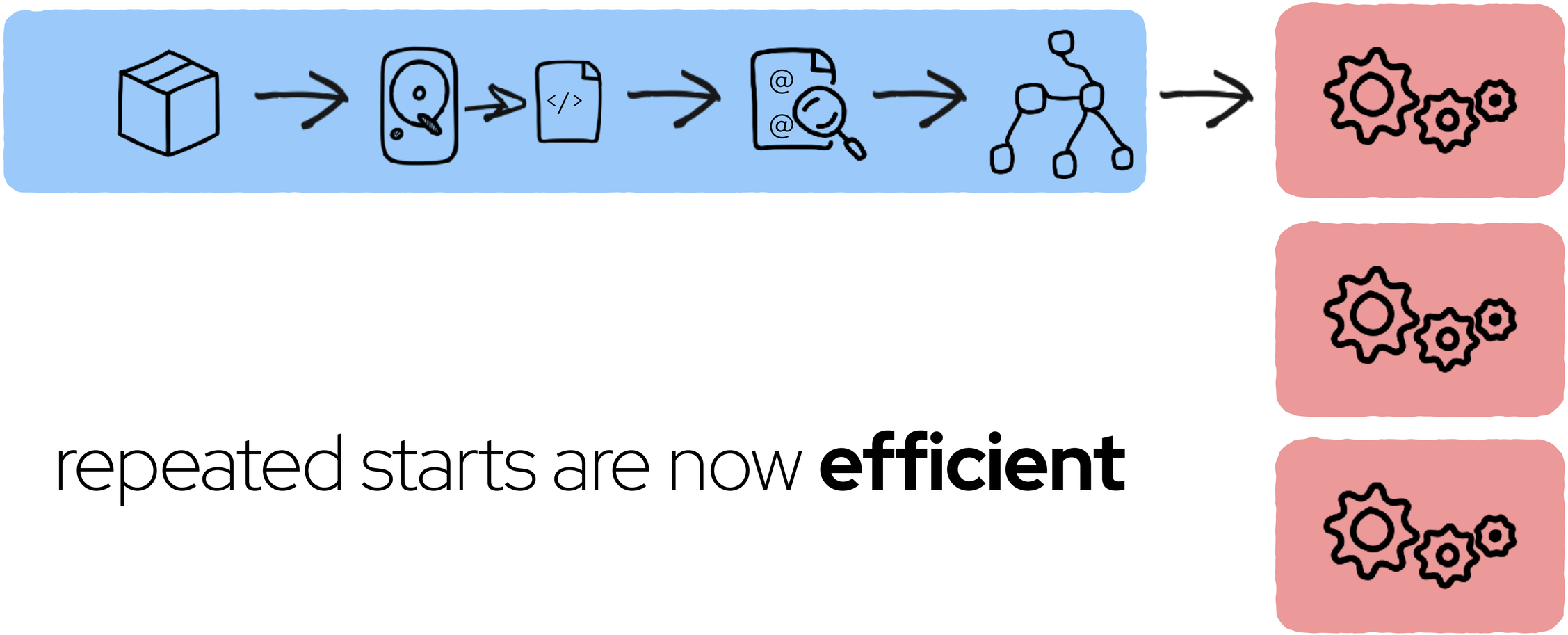
repeated starts are now **efficient**



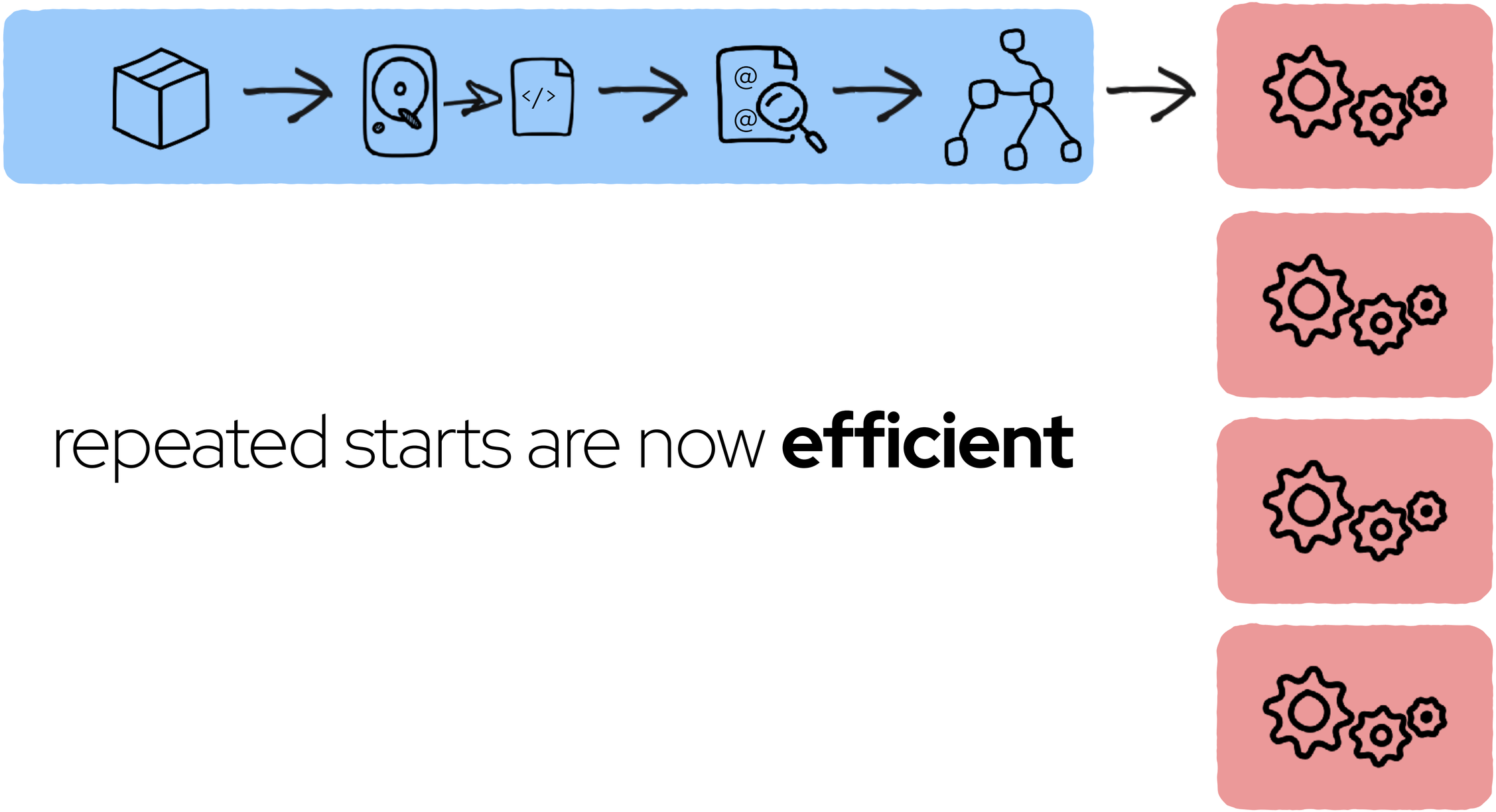
repeated starts are now **efficient**



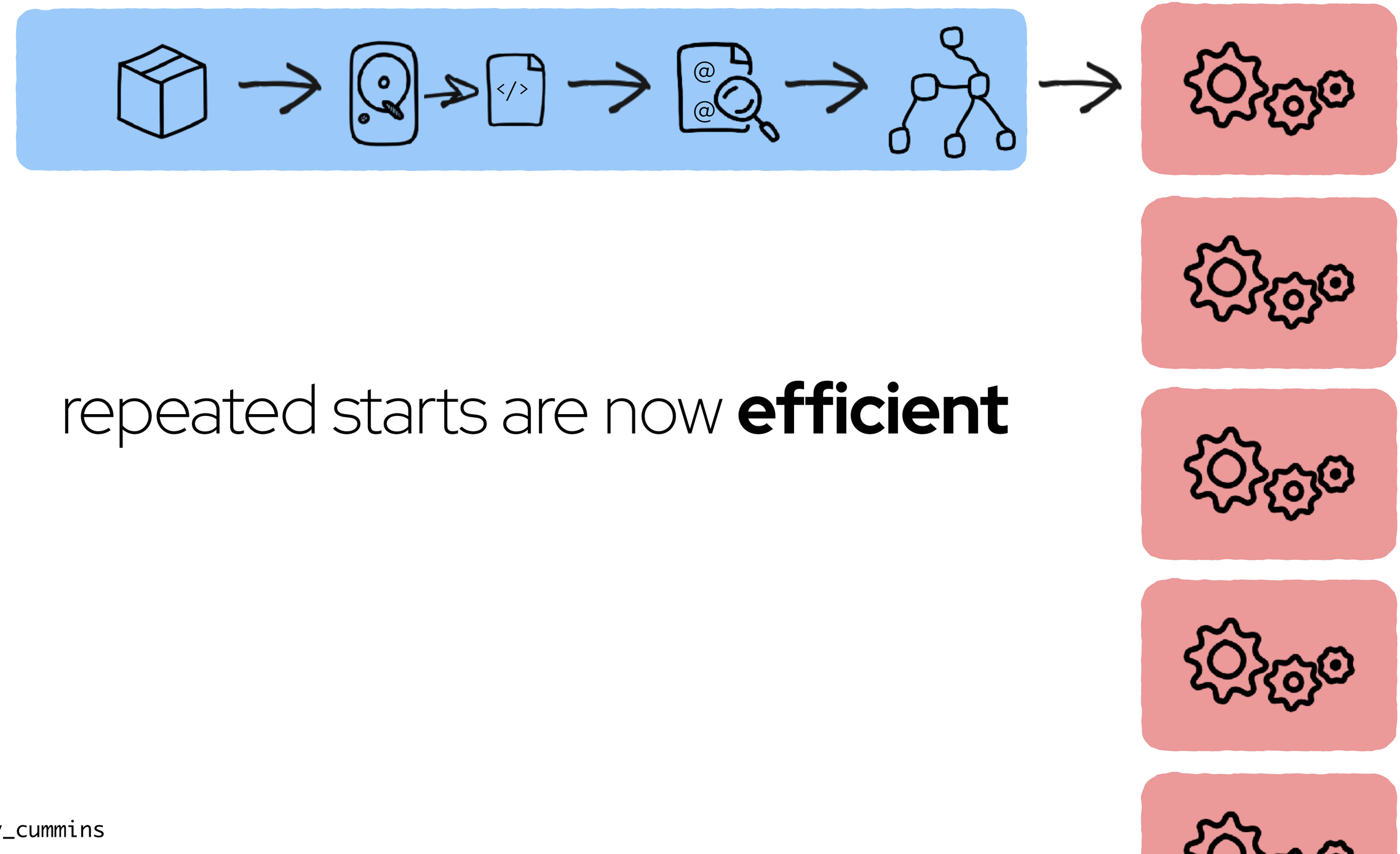
repeated starts are now **efficient**



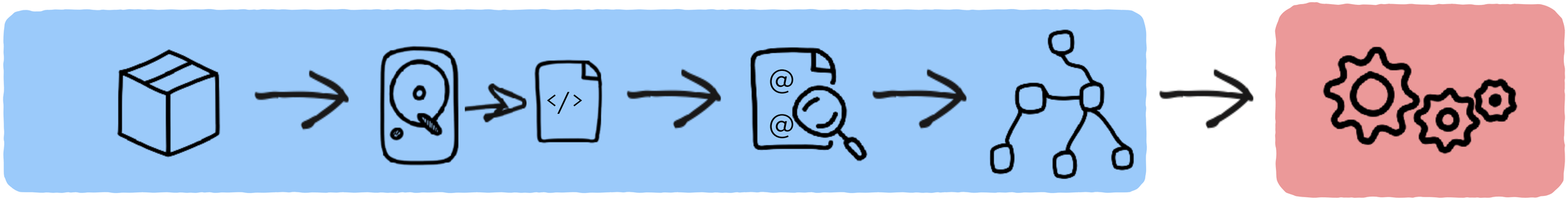
repeated starts are now **efficient**



repeated starts are now **efficient**



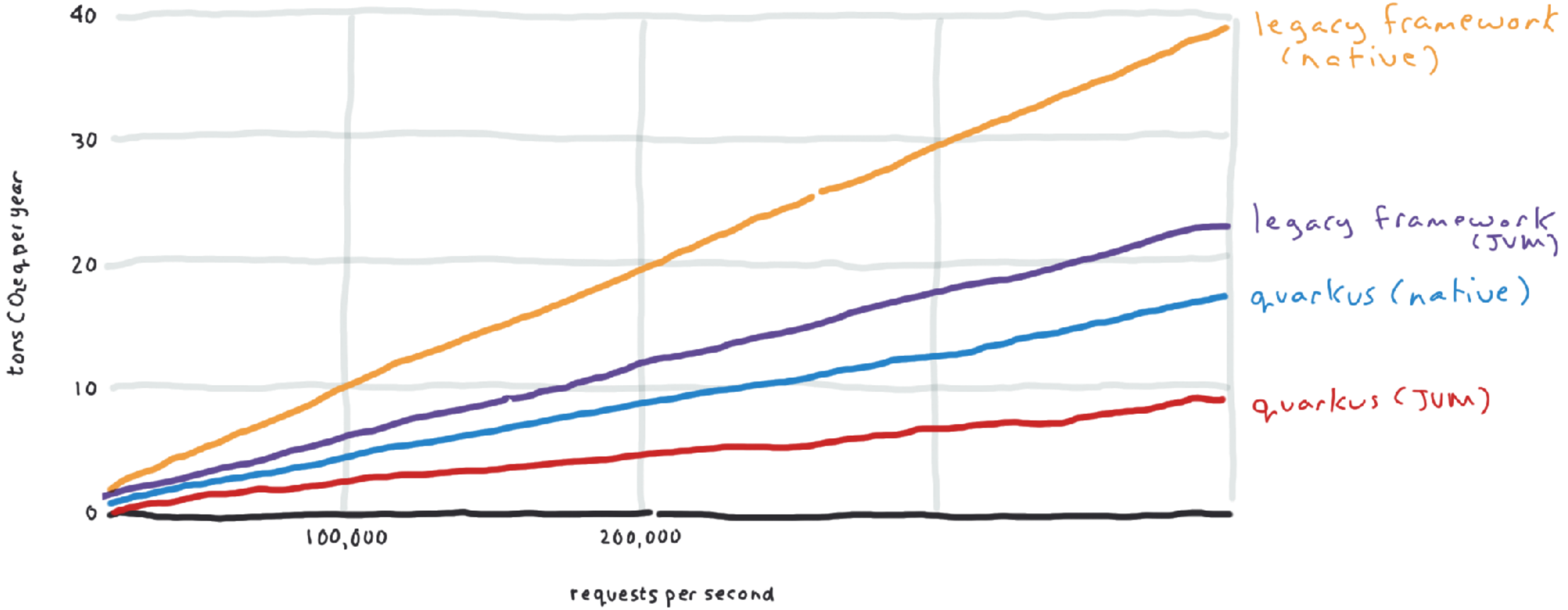
repeated starts are now **efficient**



repeated starts are now **efficient**

less wasted work ⟶

effect on energy consumption (RAPL measurements)



Setup:

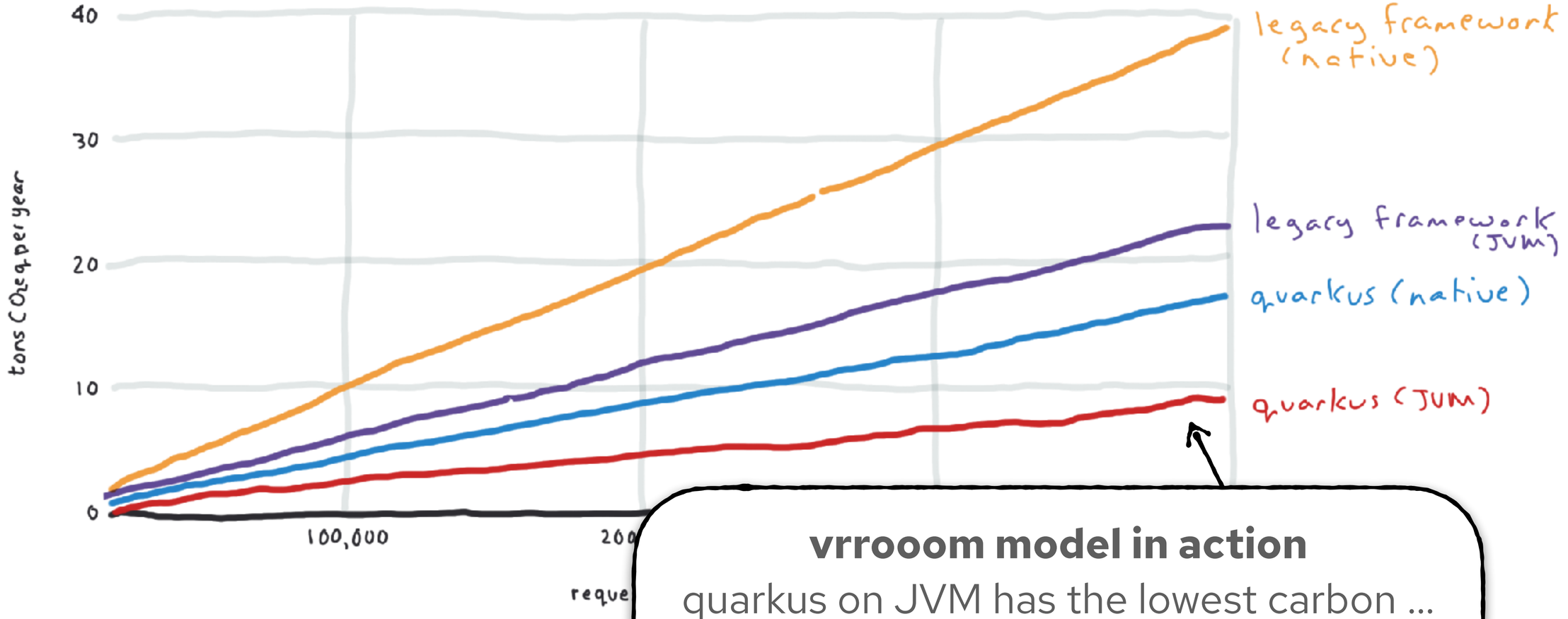
- REST + CRUD
- large heap
- RAPL energy measurement
- multiple instances to support high load

Assumptions:

- US energy mix

Source: John O'Hara

effect on energy consumption (RAPL measurements)



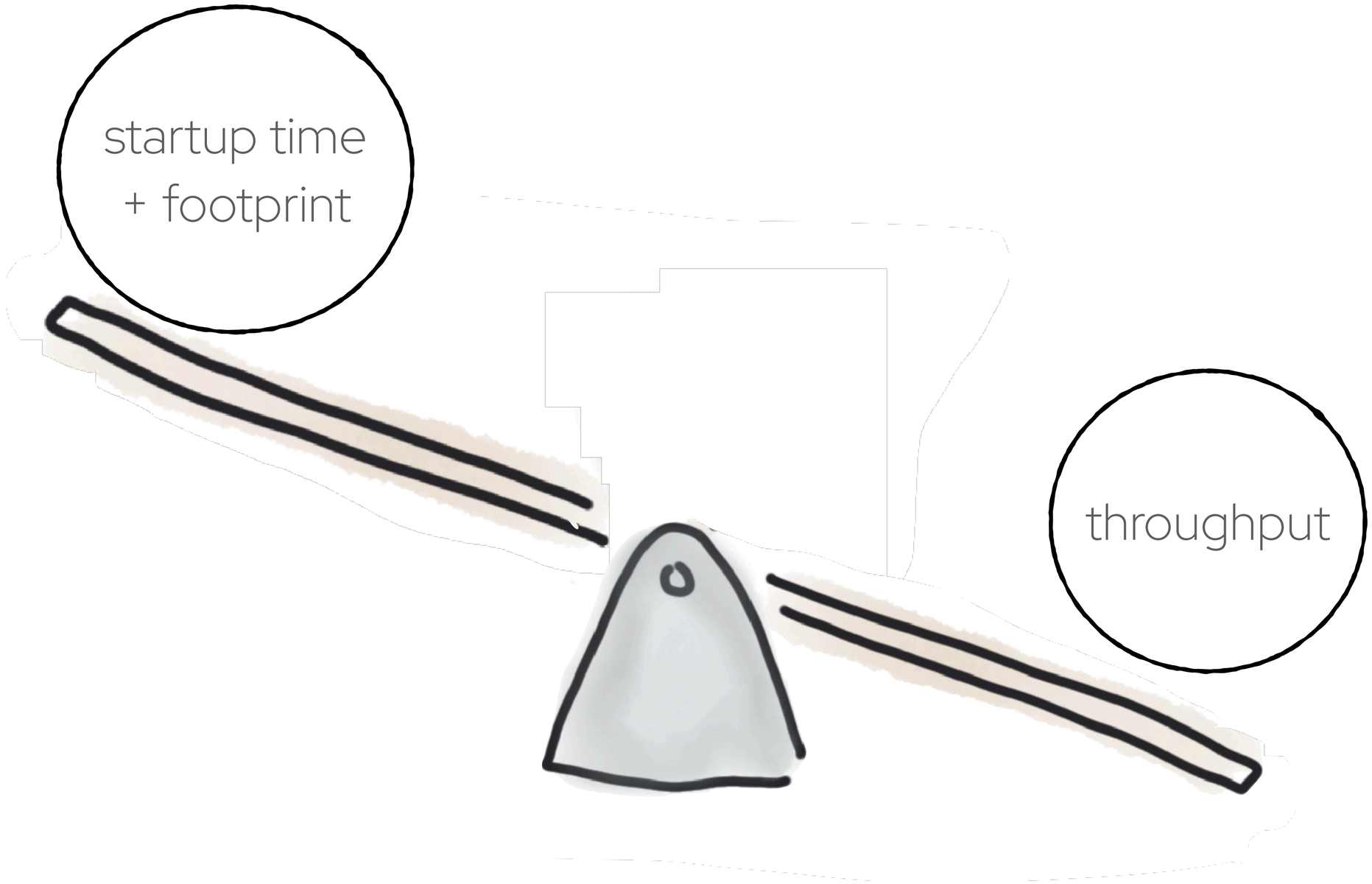
vrrooom model in action

quarkus on JVM has the lowest carbon ...
because it has the highest *throughput*

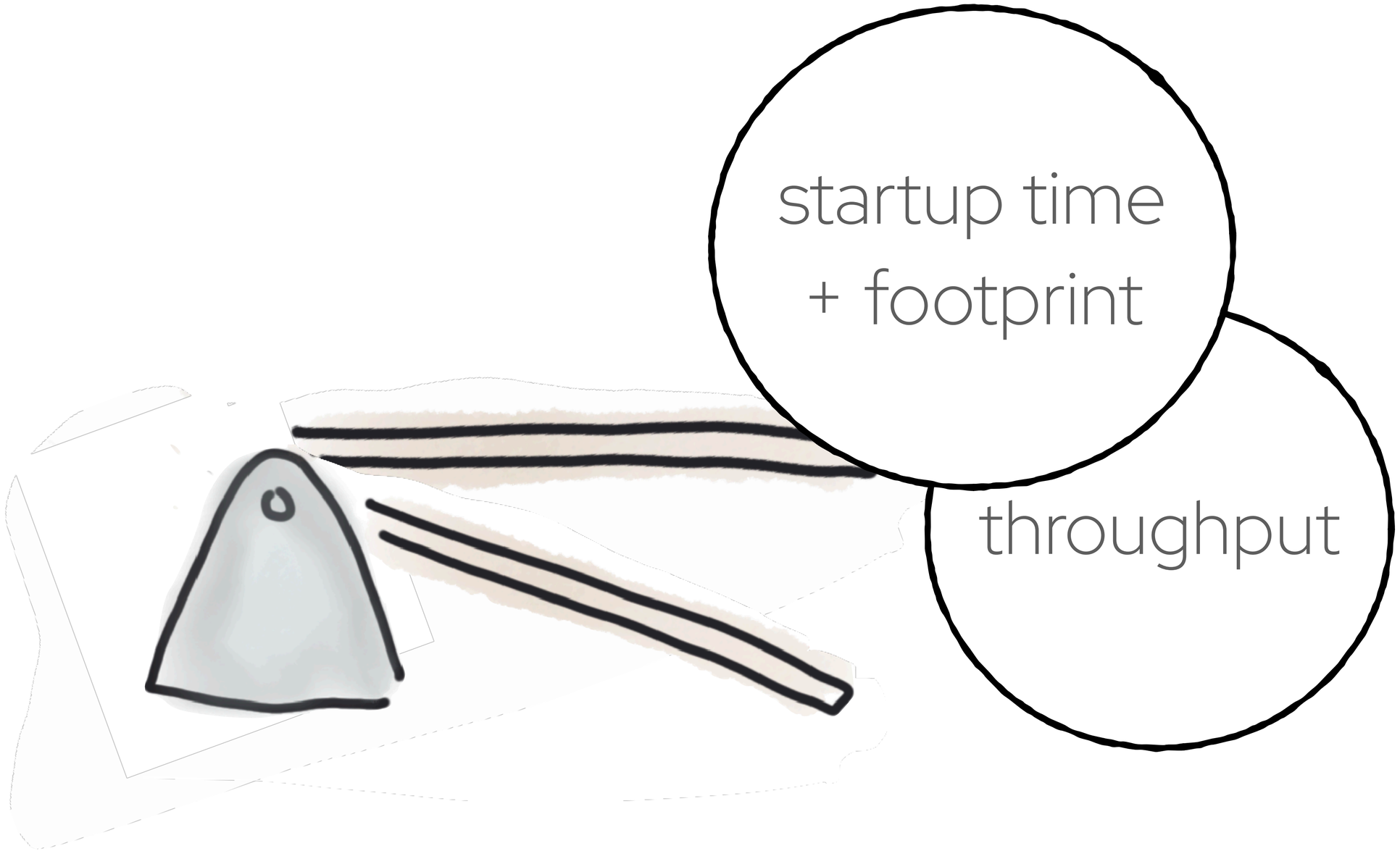
Setup:
- REST + CRUD
- large heap
- RAPL energy measurement

Assumptions:
- US energy mix

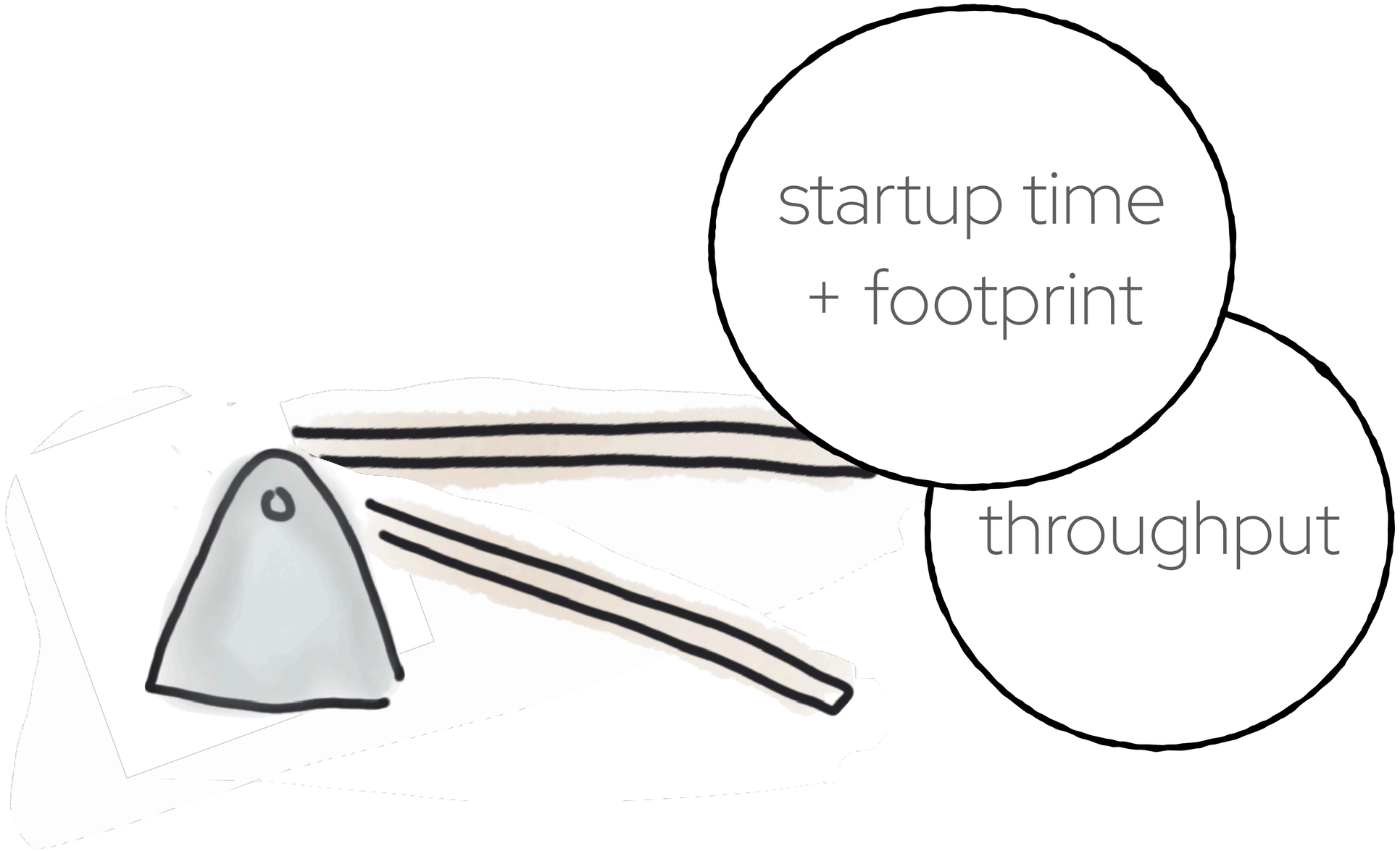
Source: John O'Hara


startup time
+ footprint
throughput

we beat the trade-off.

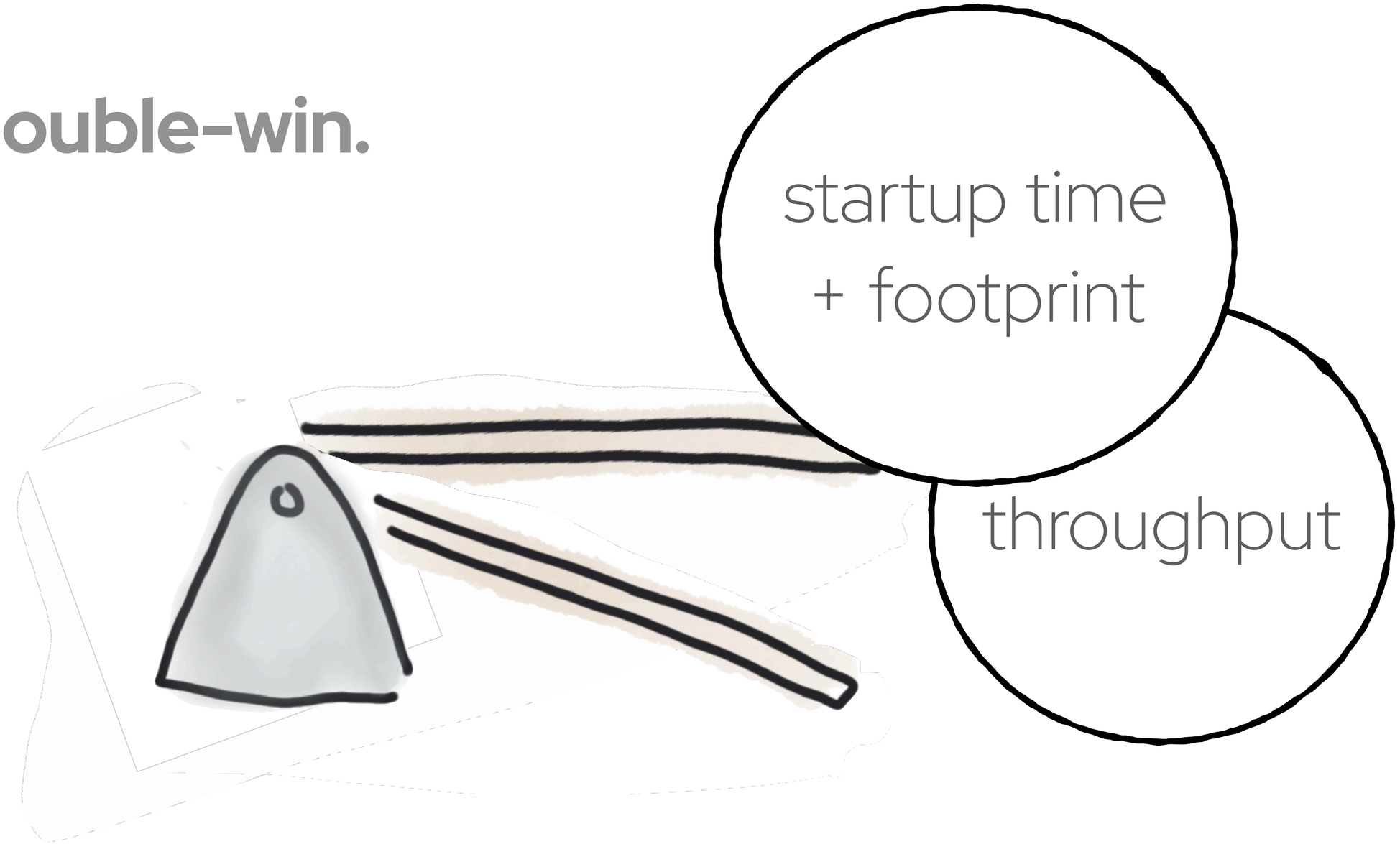


we beat the trade-off.



we beat the trade-off.

it's a double-win.



see saw folded in half - Google

https://www.google.com/search?client=firefox-b-e&sca_esv=824cd7f292a9a46d&q=see+saw+folded+in

133%

Google

see saw folded in half

Sign in

All | Images | Products | Videos | News | More | Tools | SafeSearch



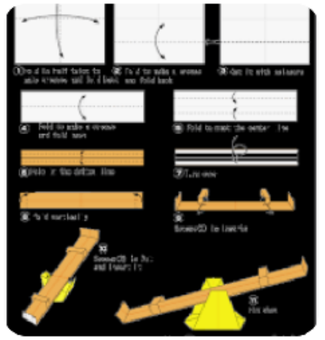
BBAE

Collapsable See-Saw - BBAE



BBAE

Collapsable See-Saw - BBAE



Origami Center

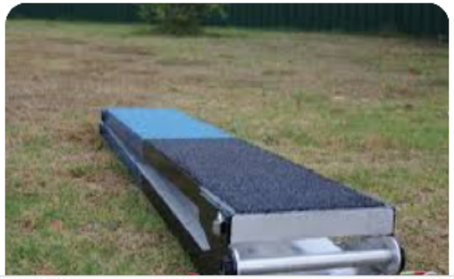
A Seesaw

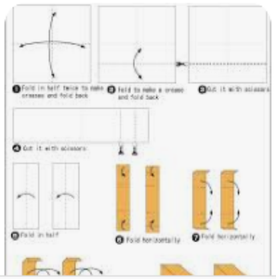


BBAE

Collapsable See-Saw - BBAE





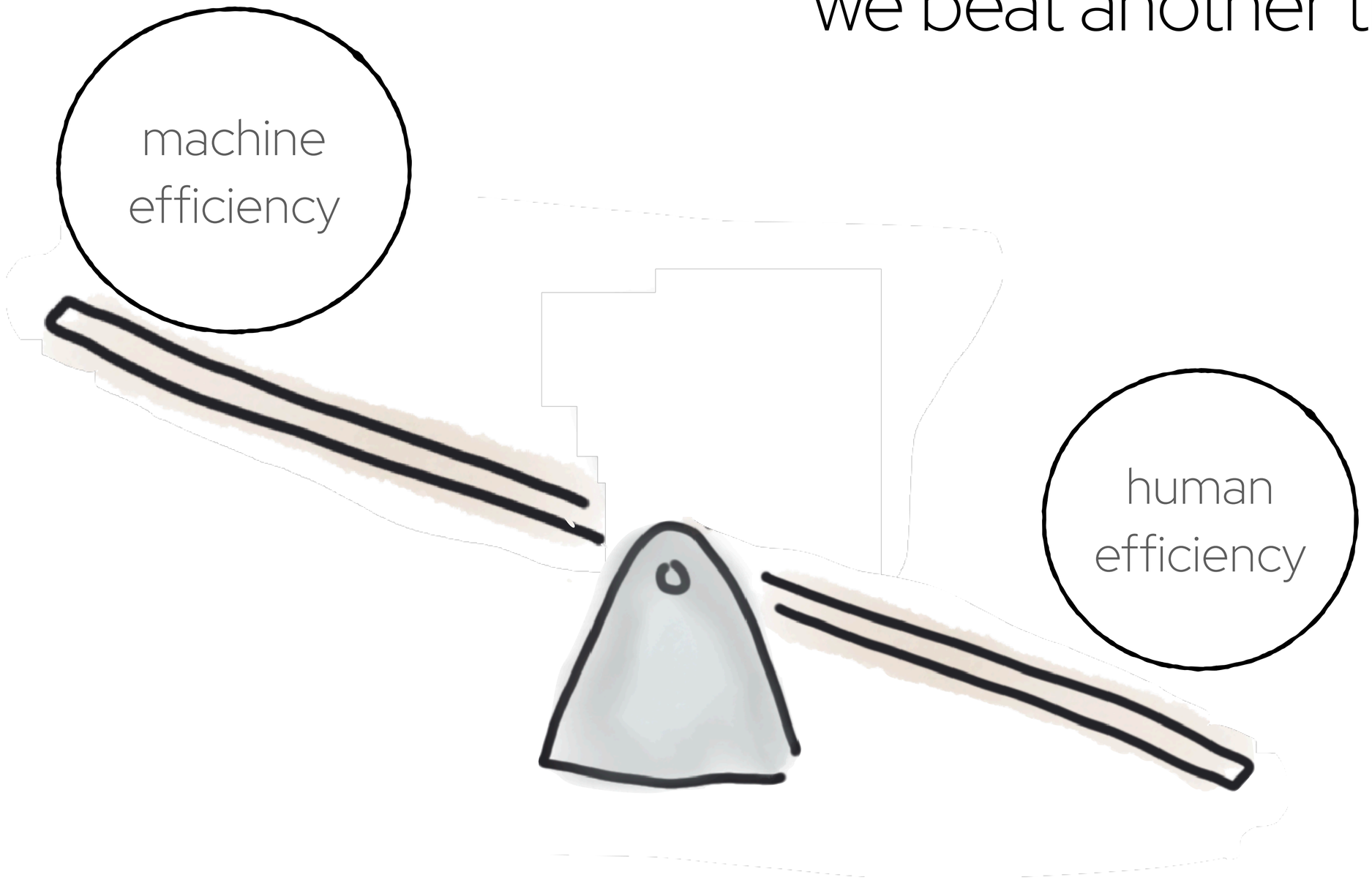




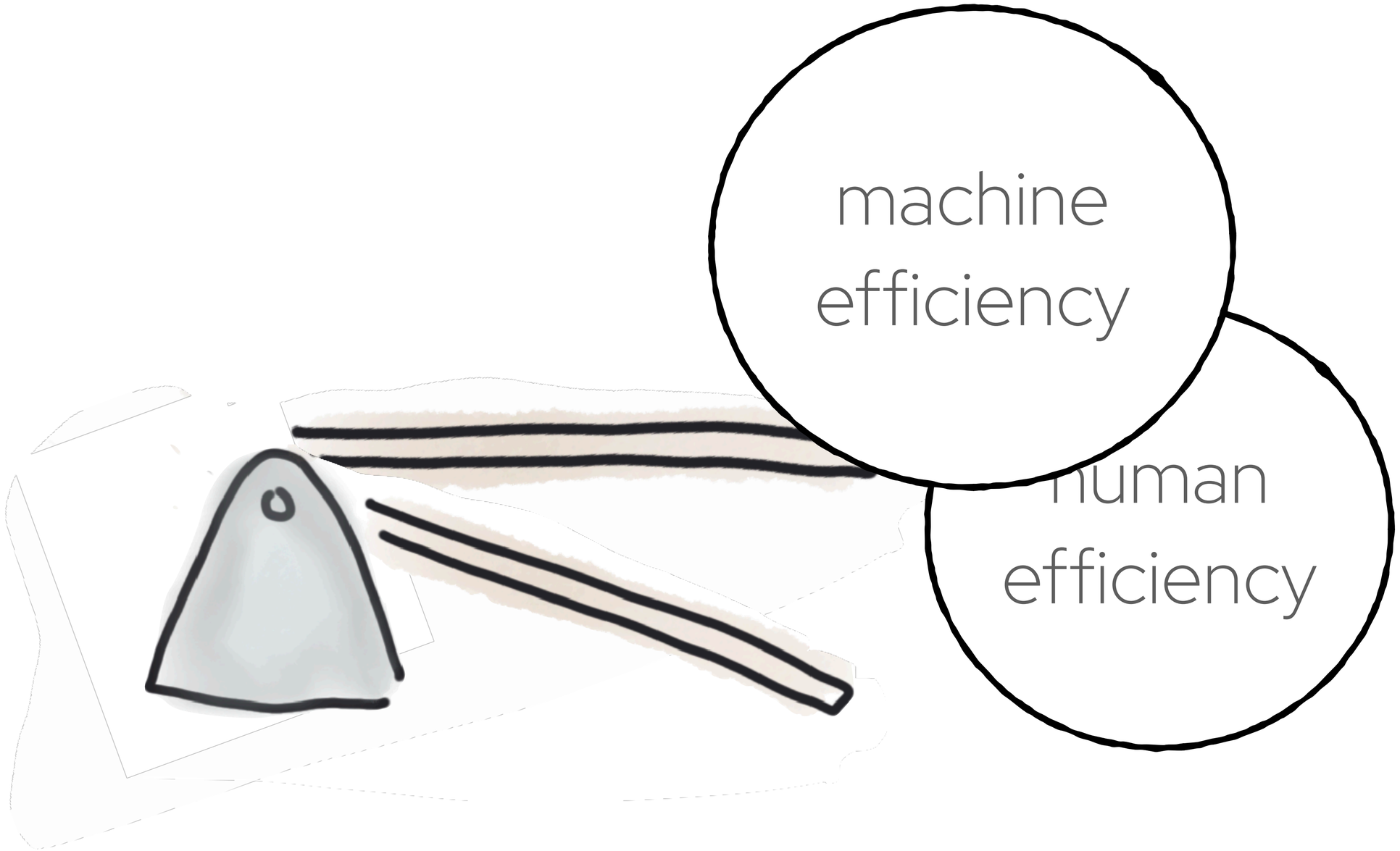


https://www.google.com/imgres?q=see saw folded in half&imgurl=https://www.bbae.com.au/uploads/1/4/6/8/14685954...UKEwjoqf_s25SFAxUjUEEAHR_6CVQQM3oECEwQAA..i&w=1100&h=733&hcb=2&ved=2ahUKEwjoqf_s25SFAxUjUEEAHR_6CVQQM3oECEwQAA

we beat another trade-off.

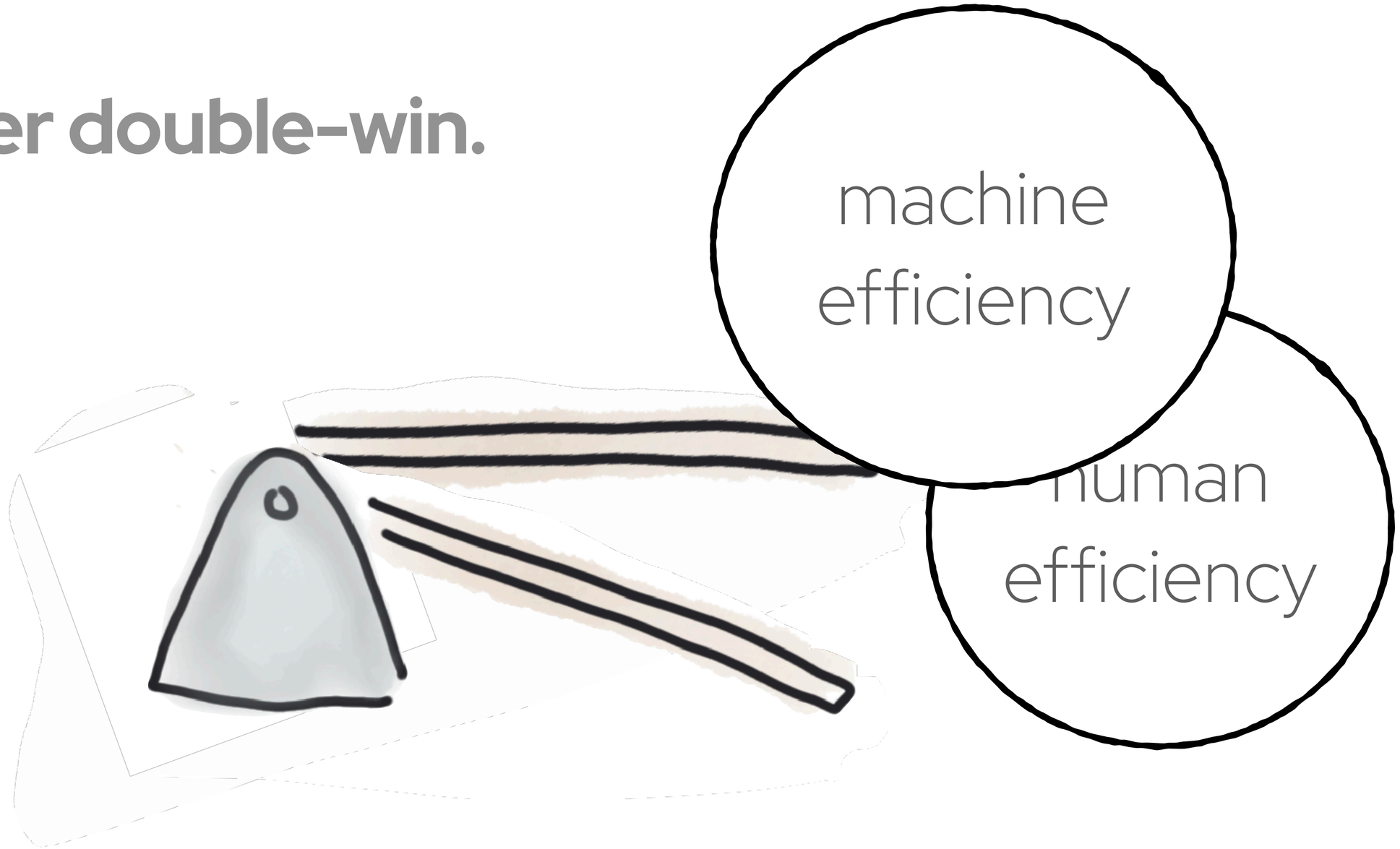


we beat another trade-off.



we beat another trade-off.

another double-win.



boilerplate code


boilerplate code

example: logging

```
package com.example;
import org.jboss.logging.Logger;

public class Thing {
    private static final Logger log = Logger.getLogger(Thing.class);

    public void doSomething() {
        log.info("It works!");
    }
}
```

example: logging

```
package com.example;
import io.quarkus.logging.Log;

public class Thing {
    private static final Logger log = Logger.getLogger(Thing.class);

    public void doSomething() {
        Log.info("It works!");
    }
}
```

example: hibernate

```
@ApplicationScoped
public class GreetingRepository {

    public Entity findByName(int name) {
        return find("name", name).firstResult();
    }

    void persist(Entity entity) {}


    void delete(Entity entity) {}


    Entity findById(Id id) {}


    List<Entity> list(String query, Sort sort, Object... params) {
        return null;
    }

    Stream<Entity> stream(String query, Object... params) {
        return null;
    }

    long count() {
        return 0;
    }

    long count(String query, Object... params) {
        return 0;
    }

}
```

example: hibernate

```
@ApplicationScoped
public class GreetingRepository {

    public Entity findByName(int name) {
        return find("name", name).firstResult();
    }

    void persist(Entity entity) {}


    void delete(Entity entity) {}


    Entity findById(Id id) {}


    List<Entity> list(String query, Sort sort, Object... params) {
        return null;
    }

    Stream<Entity> stream(String query, Object... params) {
        return null;
    }

    long count() {
        return 0;
    }

    long count(String query, Object... params) {
        return 0;
    }

}
```

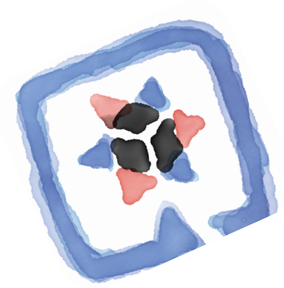
example: hibernate with panache

```
@ApplicationScoped
public class GreetingRepository implements PanacheRepository<Greeting> {

    public Entity findByName(int name) {
        return find("name", name).firstResult();
    }

}
```

DAO

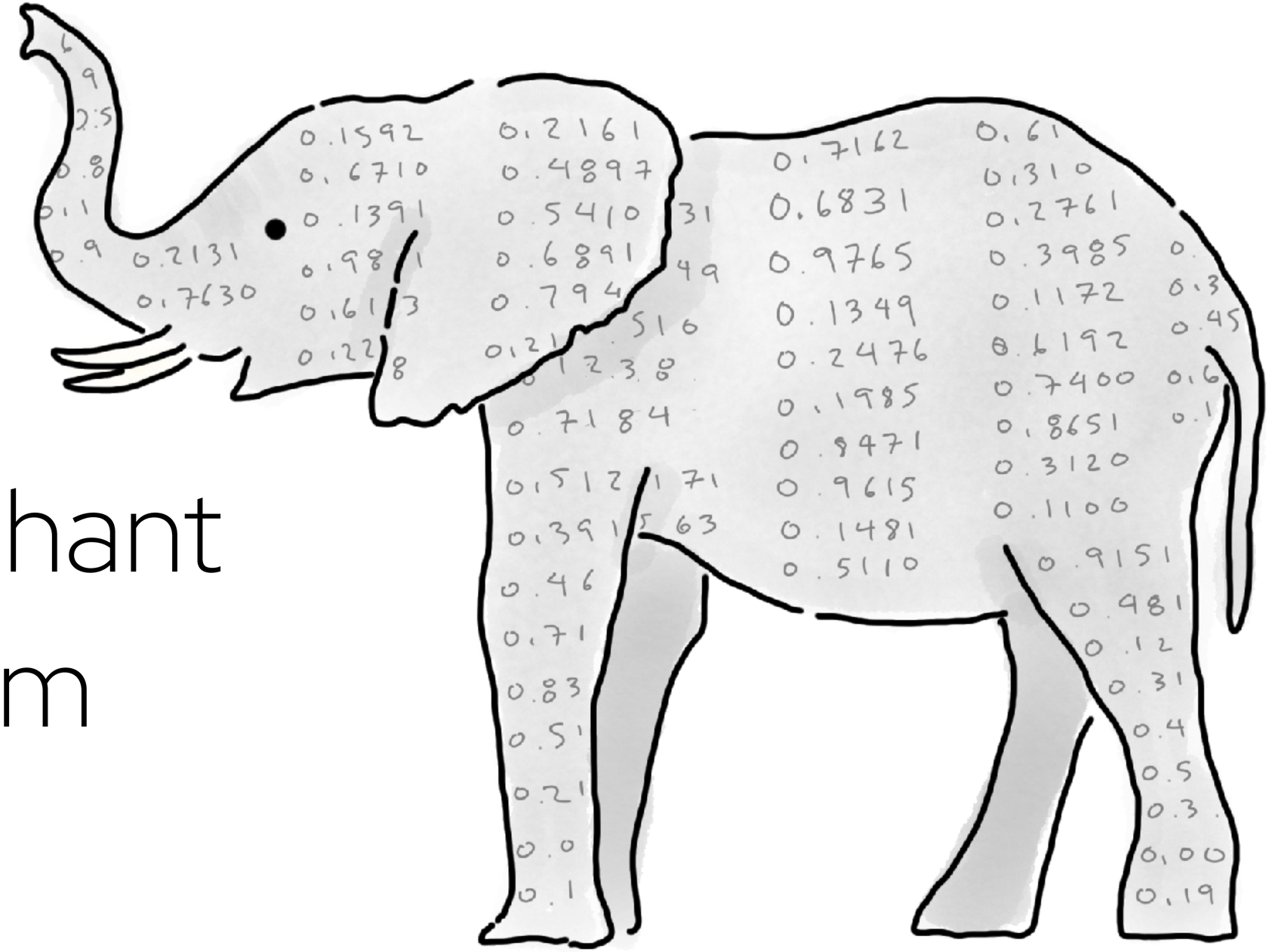


developer joy

no regrets solution:

we got rid of dynamism we didn't even want, and made **everything** better

the ai-lephant in the room



https://www.sify.com/ai-analytics/the-hilarious-and-horrifying-hallucinations-of-ai/

https://www.sify.com/ai-analytics/the-hilarious-and-horrifying-hallucinations-of-ai/

“What is the world record for crossing the English channel entirely on foot?”

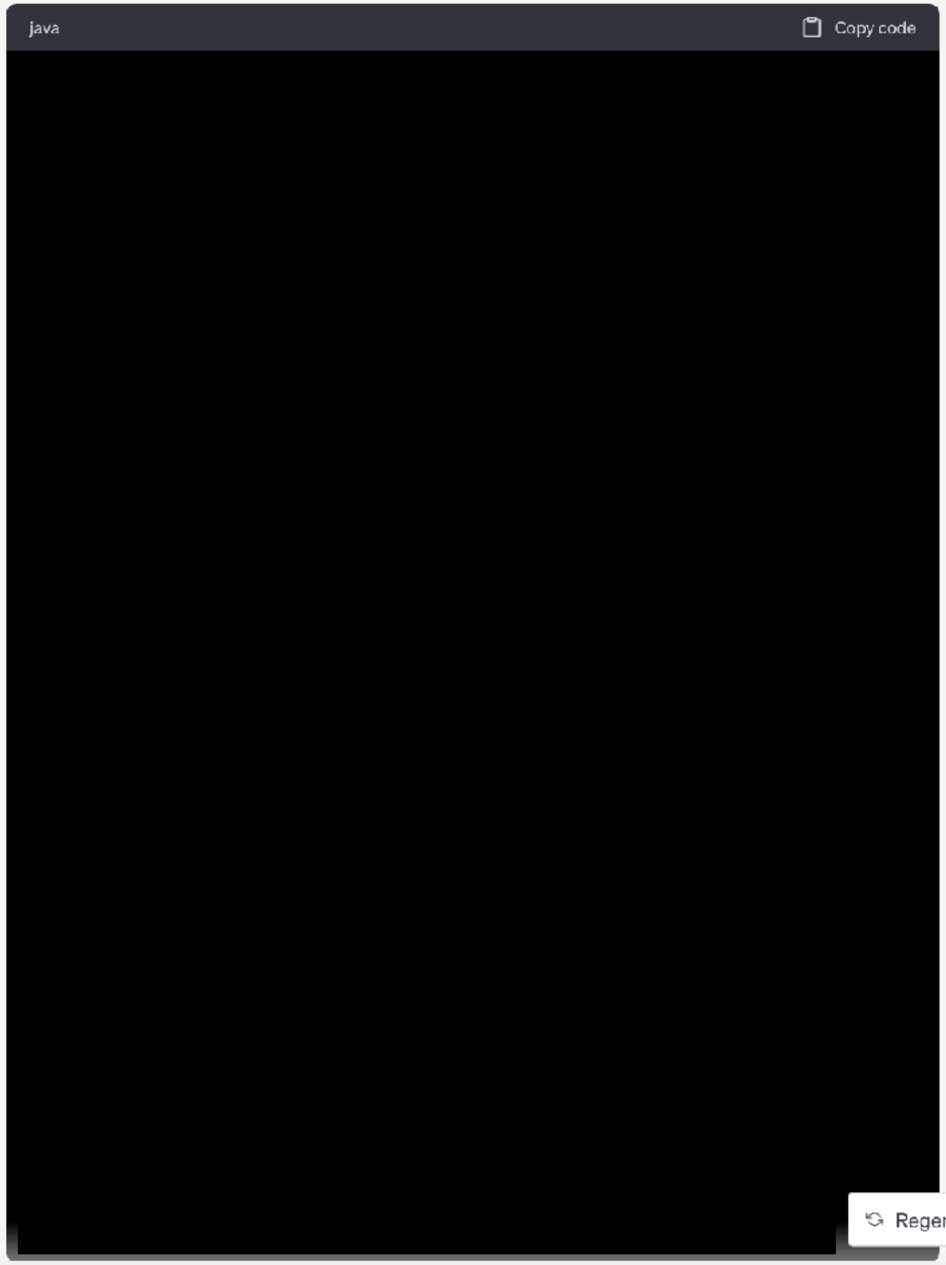
https://www.sify.com/ai-analytics/the-hilarious-and-horrifying-hallucinations-of-ai/

“What is the world record for crossing the English channel entirely on foot?”

“The world record for crossing the English Channel entirel
held by Christof Wandr
who completed the cr
and 51 minutes on A



Sure, here's a Quarkus class that allows FOSSCOMM attendees to rate talks out of 10, including the speaker name, rating, and optional comments. This class represents a `TalkRating` entity that you can use for this purpose:



Sure, here's a Quarkus class that allows FOSSCOMM attendees to rate talks out of 10, including the speaker name, rating, and optional comments. This class represents a `**TalkRating**` entity that you can use for this purpose:

java Copy code

```java
import io.quarkus.hibernate.orm.panache.PanacheEntity;

import javax.persistence.Entity;

@Entity
public class TalkRating extends PanacheEntity {
    private String speakerName;
    private int rating;
    private String comments;
```

Regenerate

Sure, here's a Quarkus class that allows FOSSCOMM attendees to rate talks out of 10, including the speaker name, rating, and optional comments. This class represents a `**TalkRating**` entity that you can use for this purpose:

java Copy code

```java
import io.quarkus.hibernate.orm.panache.PanacheEntity;

import javax.persistence.Entity;

@Entity
public class TalkRating extends PanacheEntity {
    private String speakerName;
    private int rating;
    private String comments;

    public String getSpeakerName() {
        return speakerName;
    }

    public void setSpeakerName(String speakerName) {
        this.speakerName = speakerName;
    }
```

Regenerate

Sure, here's a Quarkus class that allows FOSSCOMM attendees to rate talks out of 10, including the speaker name, rating, and optional comments. This class represents a `**TalkRating**` entity that you can use for this purpose:

java Copy code

```java
import io.quarkus.hibernate.orm.panache.PanacheEntity;

import javax.persistence.Entity;

@Entity
public class TalkRating extends PanacheEntity {
    private String speakerName;
    private int rating;
    private String comments;

    public String getSpeakerName() {
        return speakerName;
    }

    public void setSpeakerName(String speakerName) {
        this.speakerName = speakerName;
    }

    public int getRating() {
        return rating;
    }

    public void setRating(int rating) {
        this.rating = rating;
    }
```

Regenerate

Sure, here's a Quarkus class that allows FOSSCOMM attendees to rate talks out of 10, including the speaker name, rating, and optional comments. This class represents a `TalkRating` entity that you can use for this purpose:

java Copy code

```java
import io.quarkus.hibernate.orm.panache.PanacheEntity;

import javax.persistence.Entity;

@Entity
public class TalkRating extends PanacheEntity {
    private String speakerName;
    private int rating;
    private String comments;

    public String getSpeakerName() {
        return speakerName;
    }

    public void setSpeakerName(String speakerName) {
        this.speakerName = speakerName;
    }

    public int getRating() {
        return rating;
    }

    public void setRating(int rating) {
        this.rating = rating;
    }

    public String getComments() {
        return comments;
    }

    public void setComments(String comments) {
        this.comments = comments;
    }
```

Regenerate

Sure, here's a Quarkus class that allows FOSSCOMM attendees to rate talks out of 10, including the speaker name, rating, and optional comments. This class represents a `TalkRating` entity that you can use for this purpose:

java Copy code

```java
import io.quarkus.hibernate.orm.panache.PanacheEntity;

import javax.persistence.Entity;

@Entity
public class TalkRating extends PanacheEntity {
    private String speakerName;
    private int rating;
    private String comments;

    public String getSpeakerName() {
        return speakerName;
    }

    public void setSpeakerName(String speakerName) {
        this.speakerName = speakerName;
    }

    public int getRating() {
        return rating;
    }

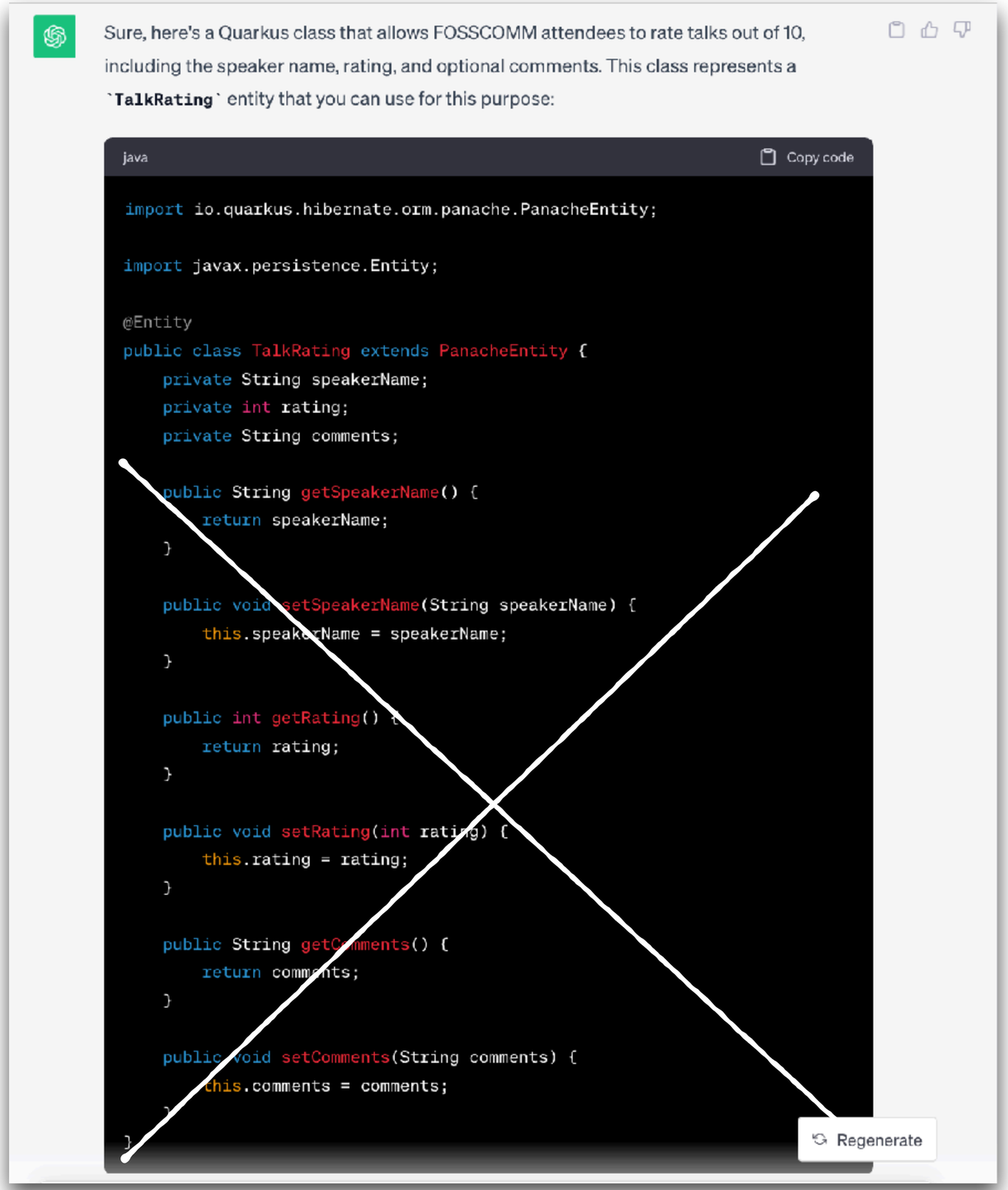
    public void setRating(int rating) {
        this.rating = rating;
    }

    public String getComments() {
        return comments;
    }

    public void setComments(String comments) {
        this.comments = comments;
    }
}
```

Regenerate

70%
unnecessary code



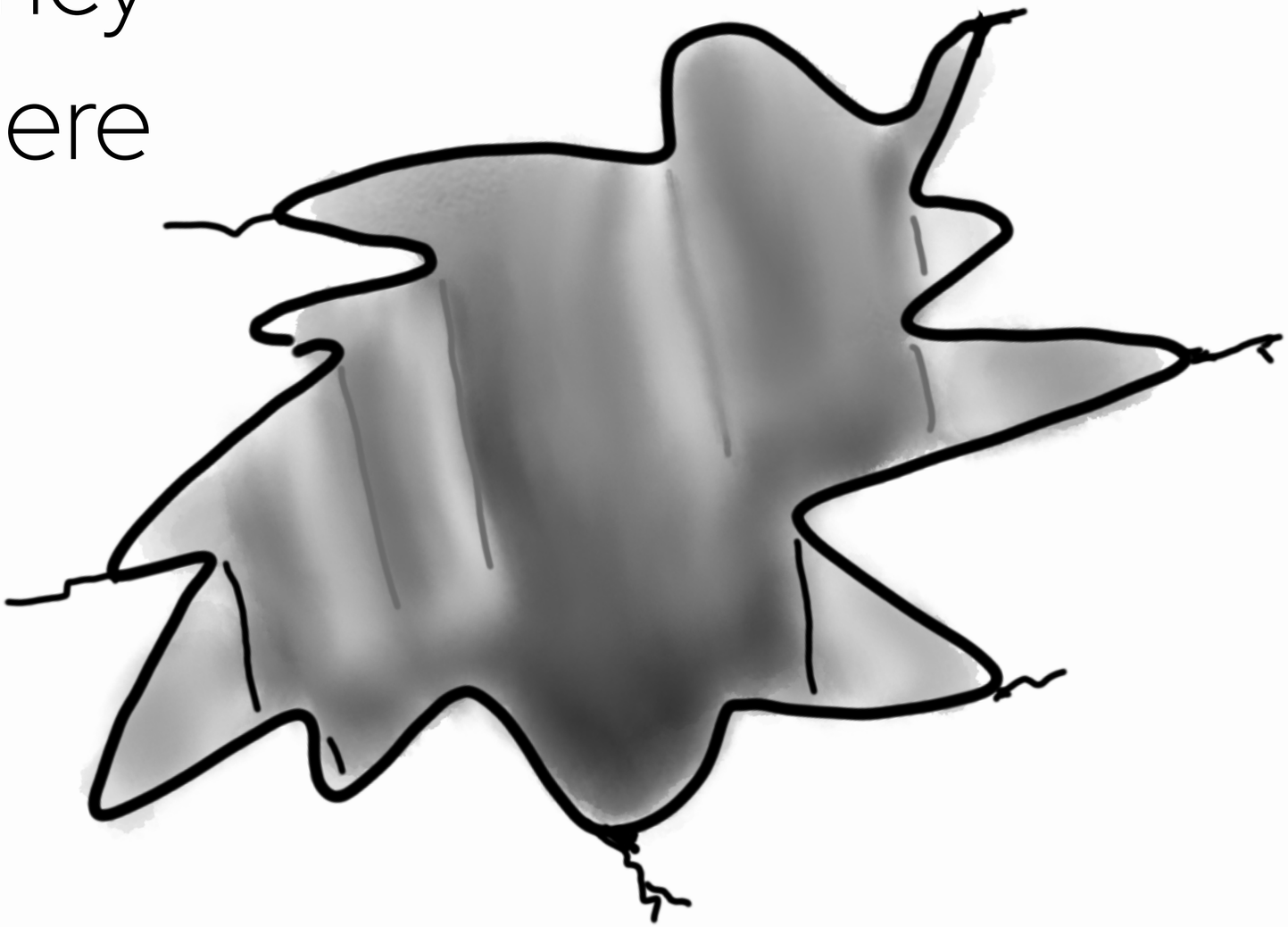
“illusion of efficiency”

we have a **solution**

solution:

- small fine-tuned models
- small model + RAG
- hybrid: symbolic reasoning + model

illusions of efficiency
are everywhere



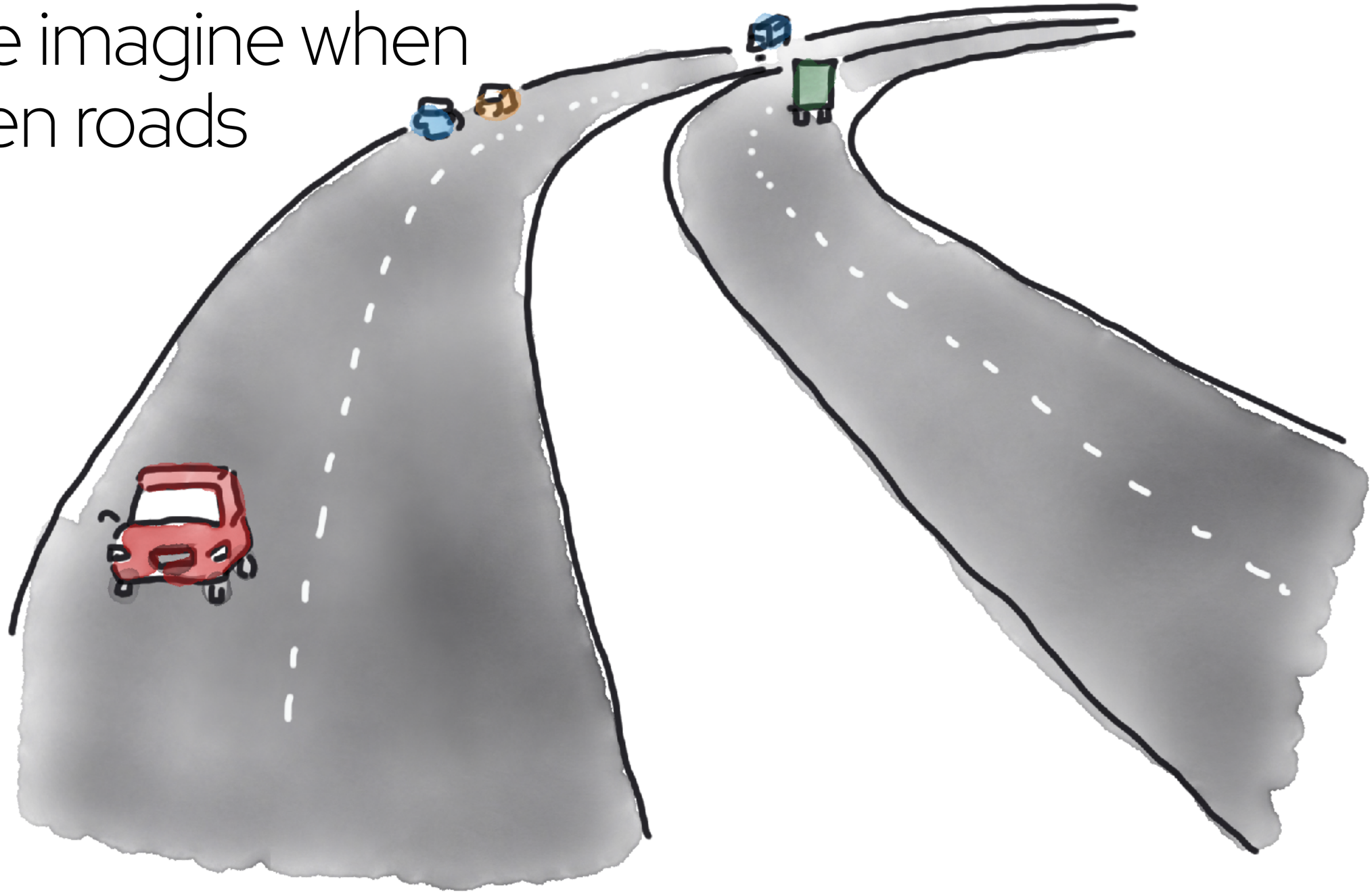

Yash Zaveri · 1st
Partner & Managing Director at Flash Ventures
11m
I just heard of a founder who monitors his co-founder's and employee's productivity via a Whoop group. The team is collectively averaging 5.5 hours of sleep per night.


Yash Zaveri · 1st
Partner & Managing Director at Flash Ventures
11m
I just heard of a founder who monitors his co-founder's and employee's productivity via a Whoop group. The team is collectively averaging 5.5 hours of sleep per night. If you haven't implemented this already at your portfolio companies to monitor founder work ethic, I strongly recommend it. #team #productivity #sleep #work #blessed

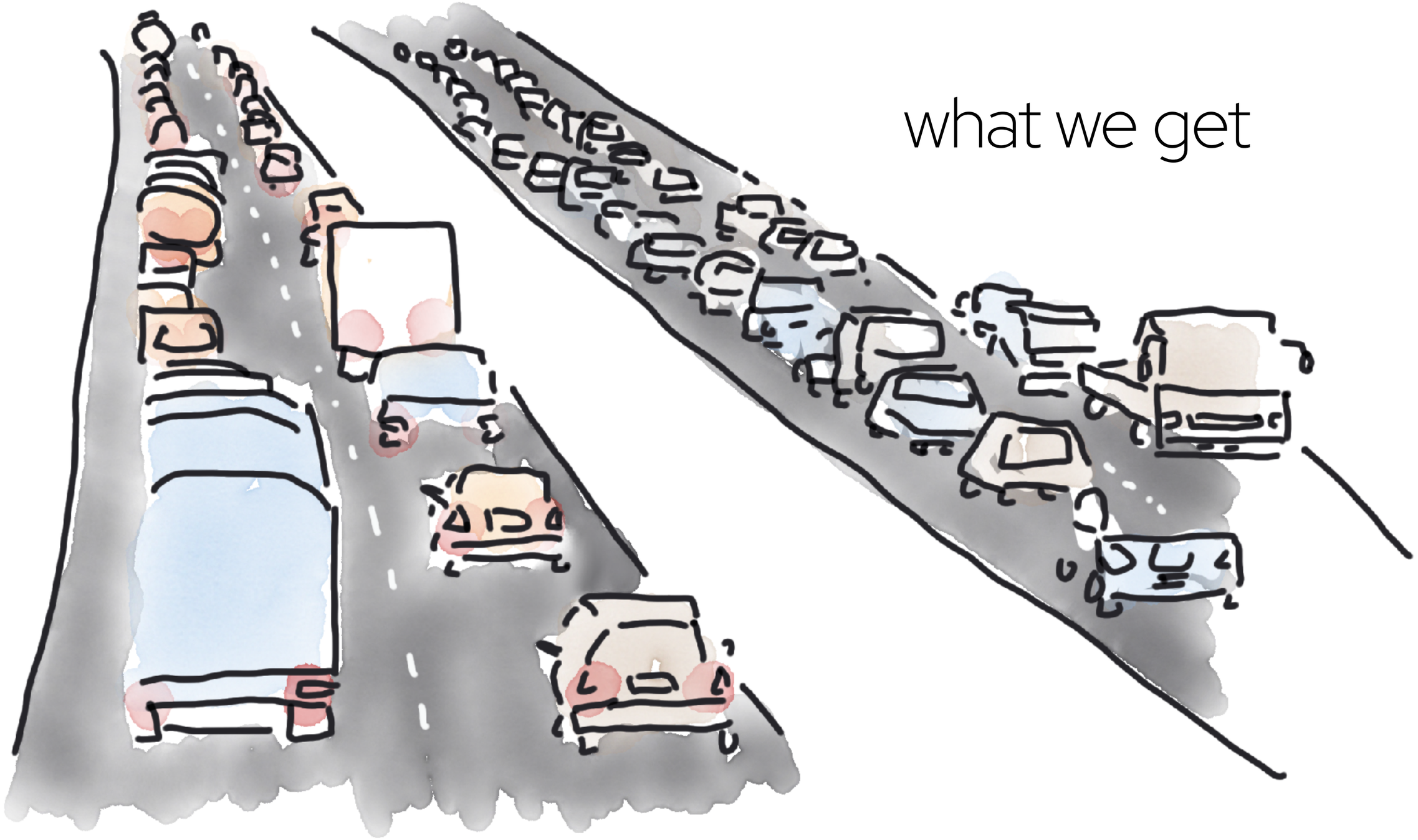
how efficient
can we get?

Jevon's paradox

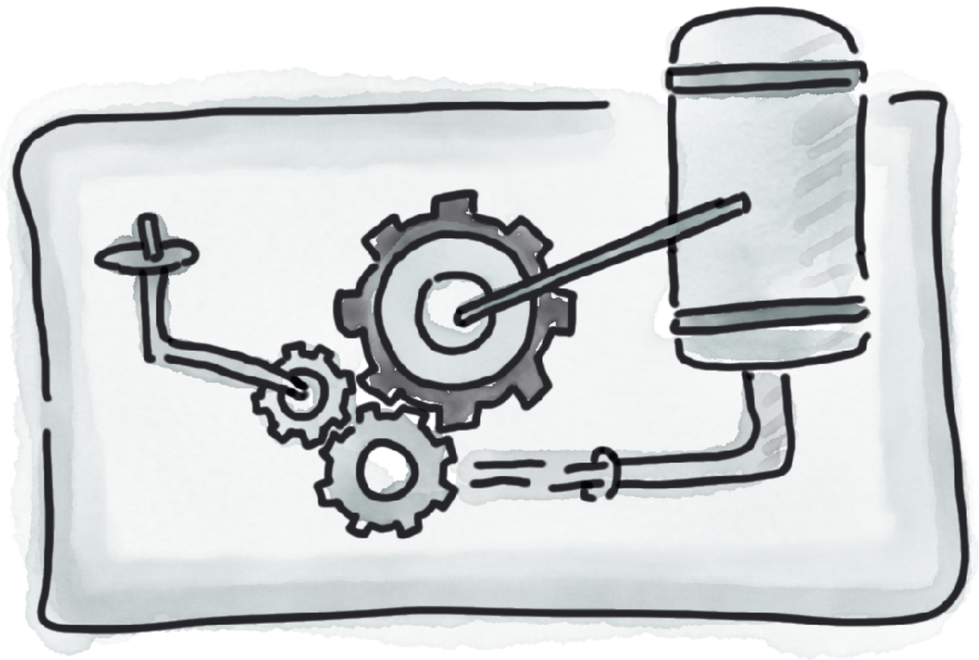
what we imagine when we widen roads



what we get

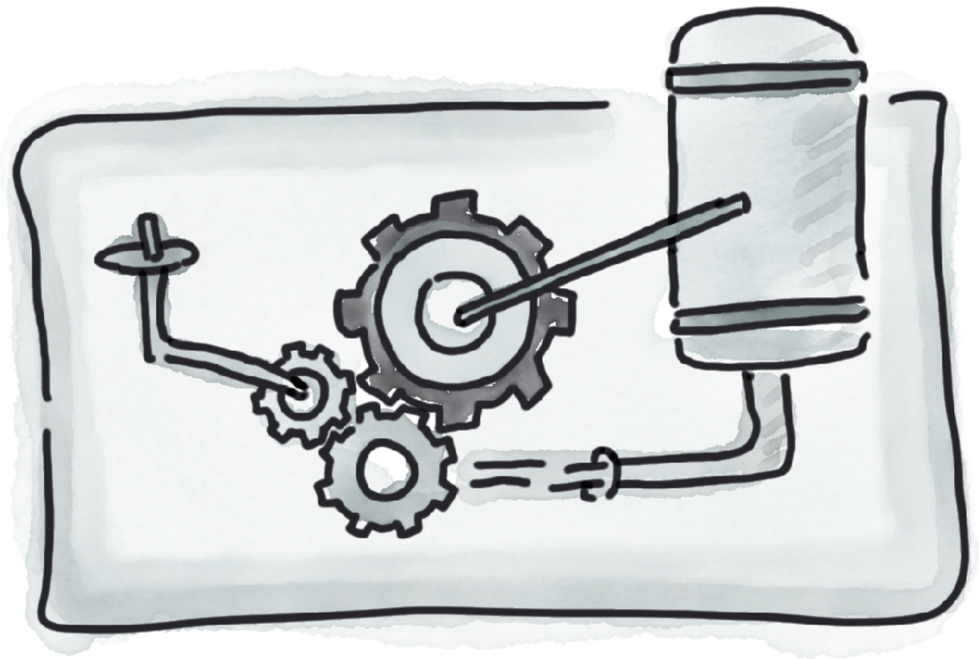


limit 2



even for machines, there is a

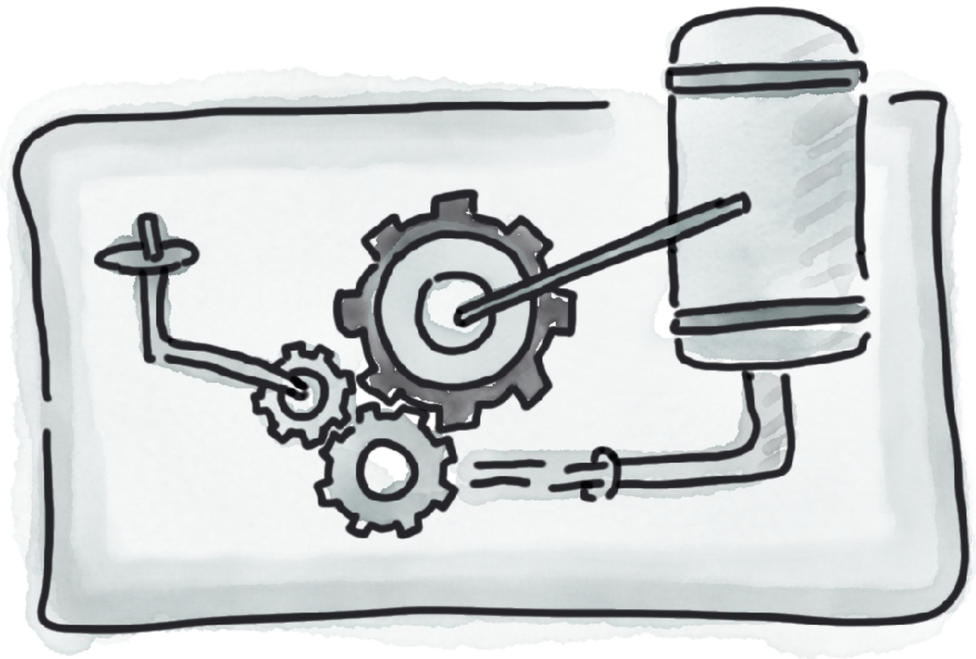
limit to efficiency



even for machines, there is a

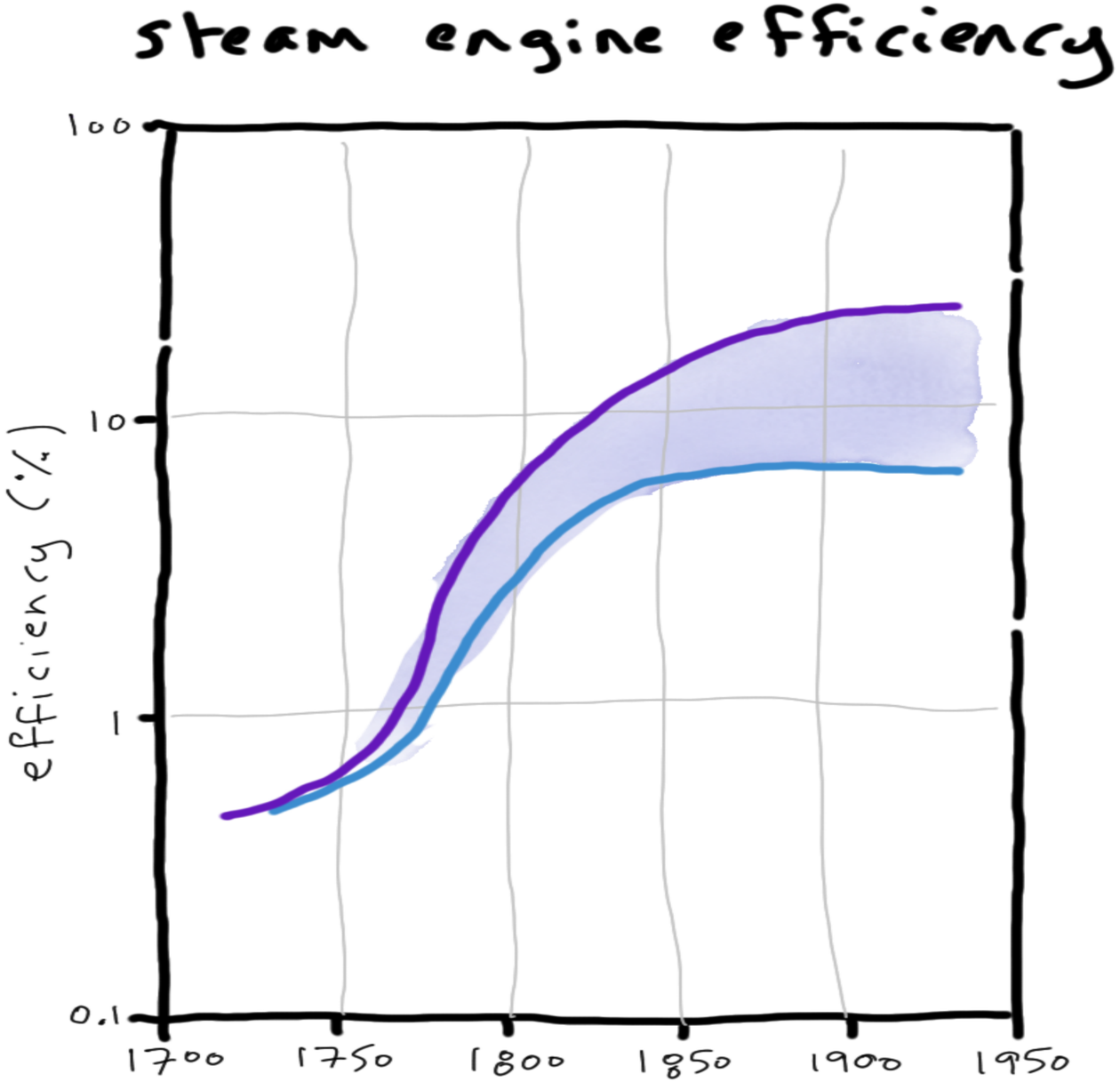
limit to efficiency

...and it's lower than you think



source: http://geosci.uchicago.edu/~moyer/GEOS24705/2016/Assignments/PS7.pdf

physics

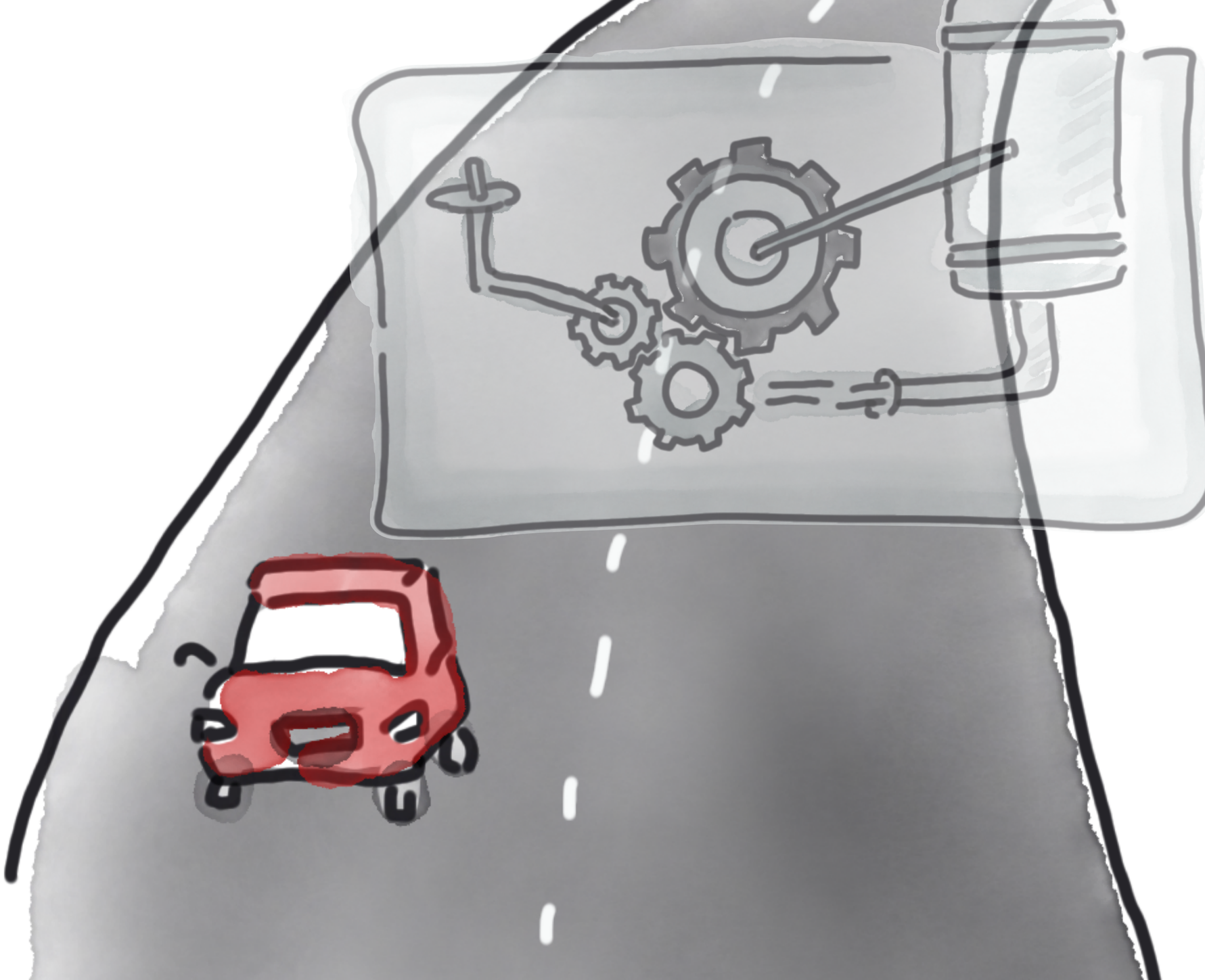


limit 2



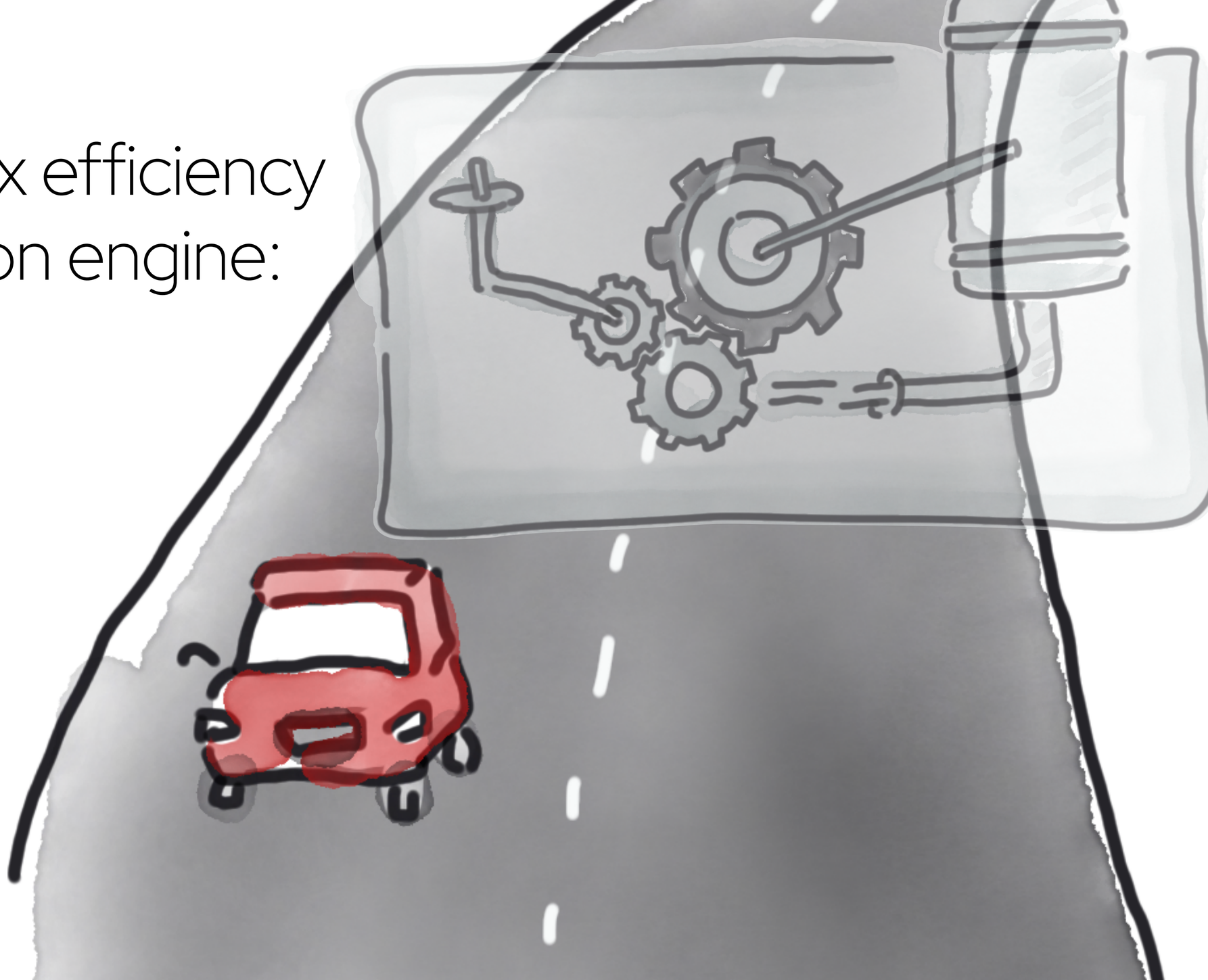
@holly_cummins

limit 2



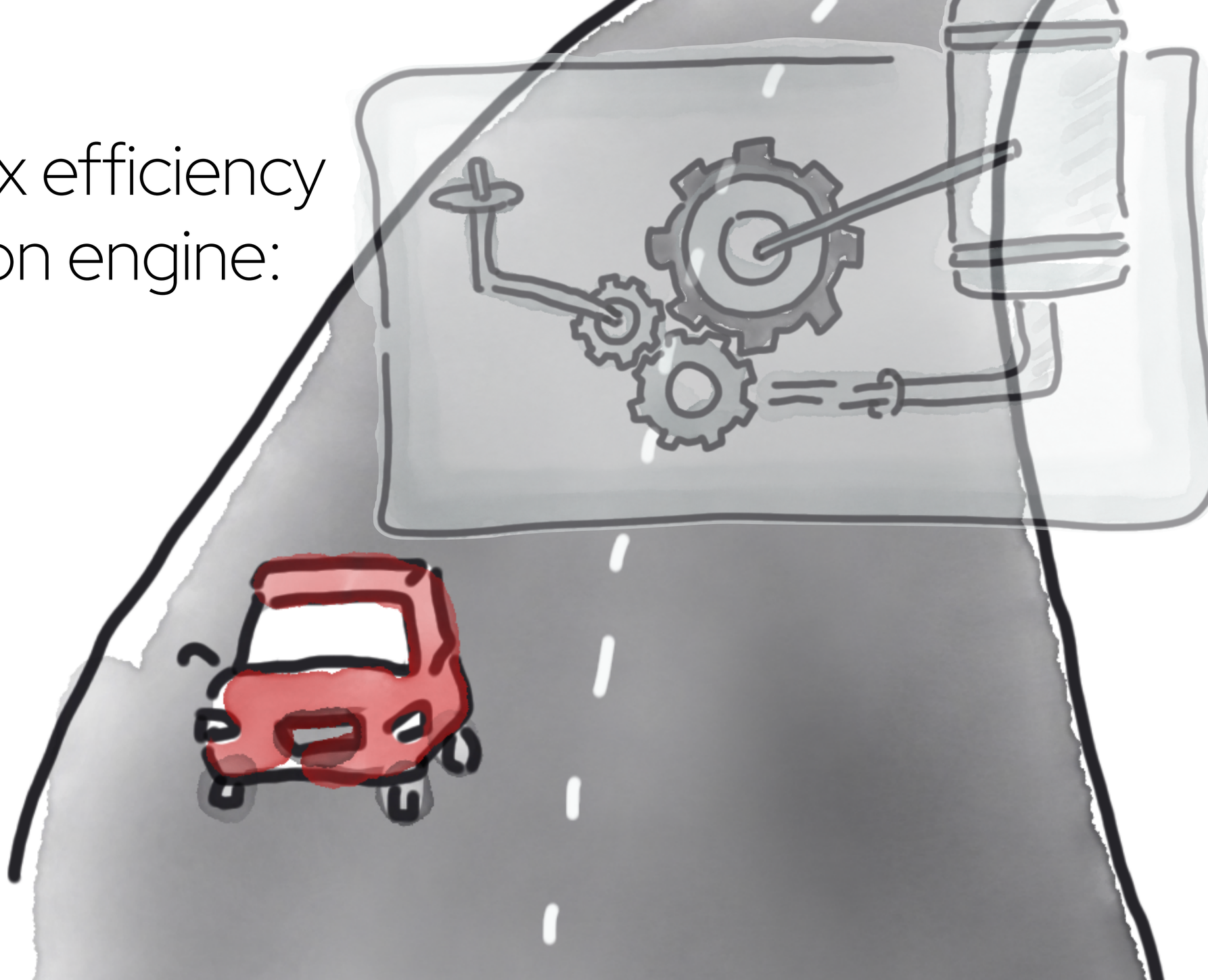
@holly_cummins

theoretical max efficiency
of a combustion engine:



theoretical max efficiency
of a combustion engine:

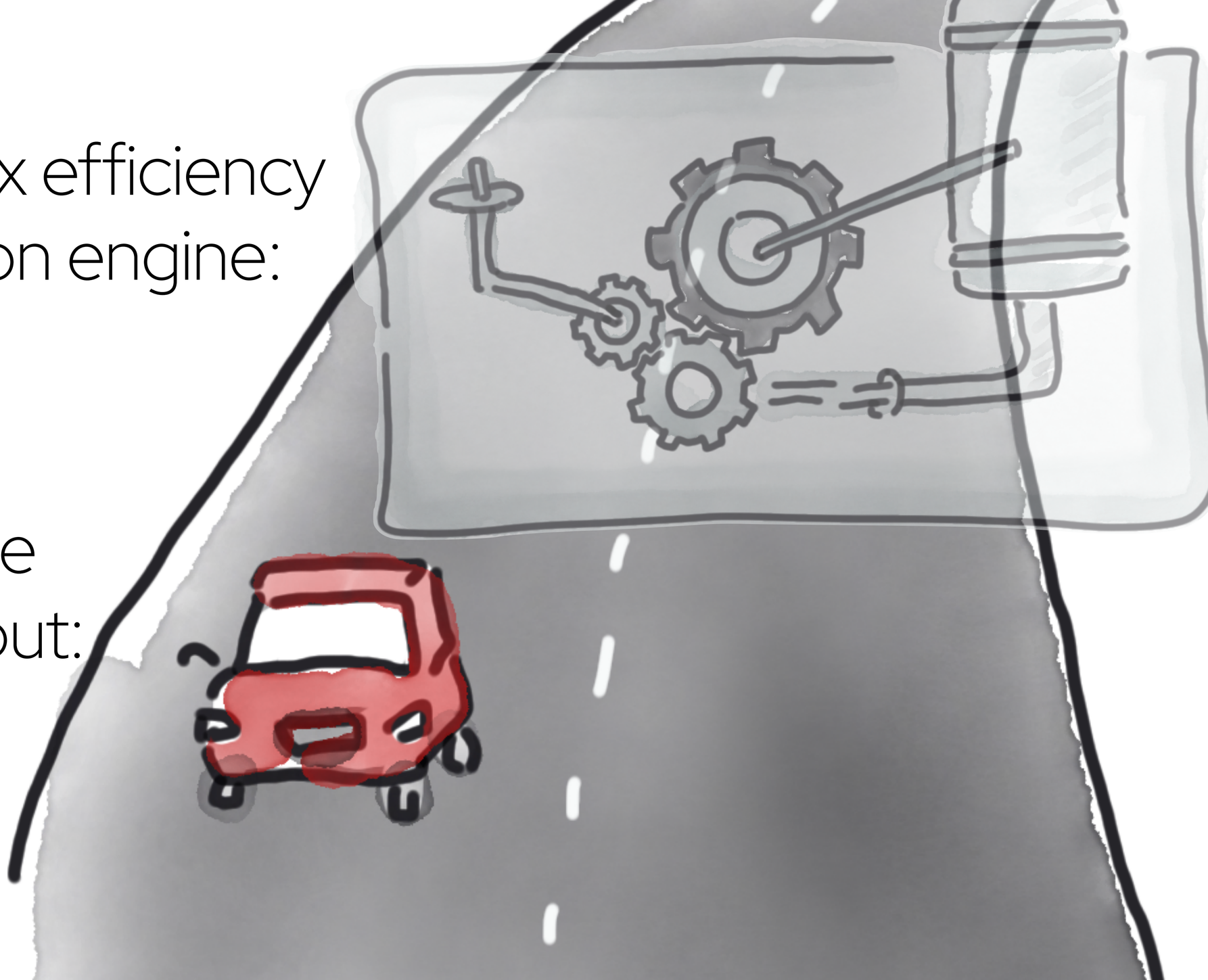
37%



theoretical max efficiency of a combustion engine:

37%

... so the engine doesn't wear out:

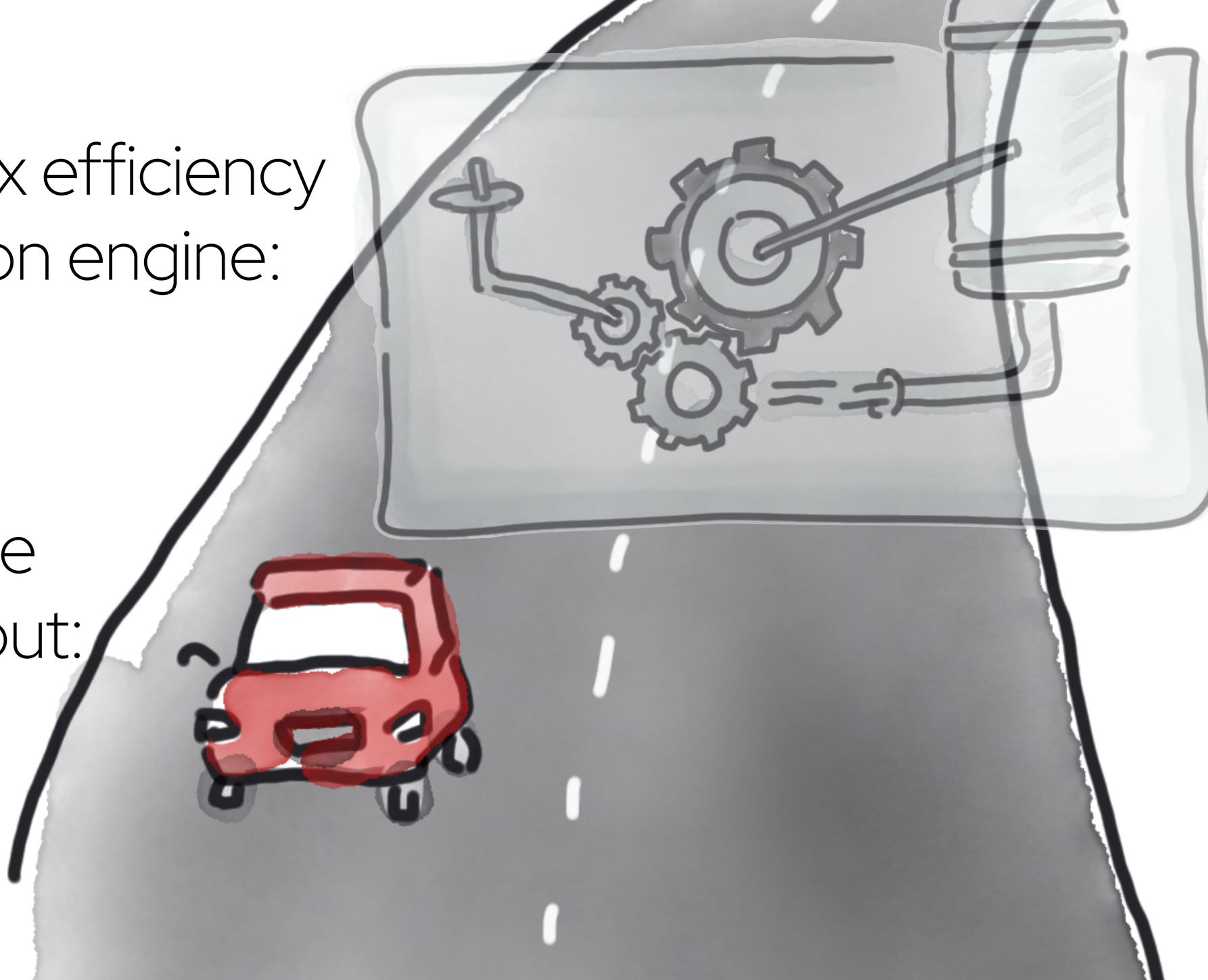


theoretical max efficiency of a combustion engine:

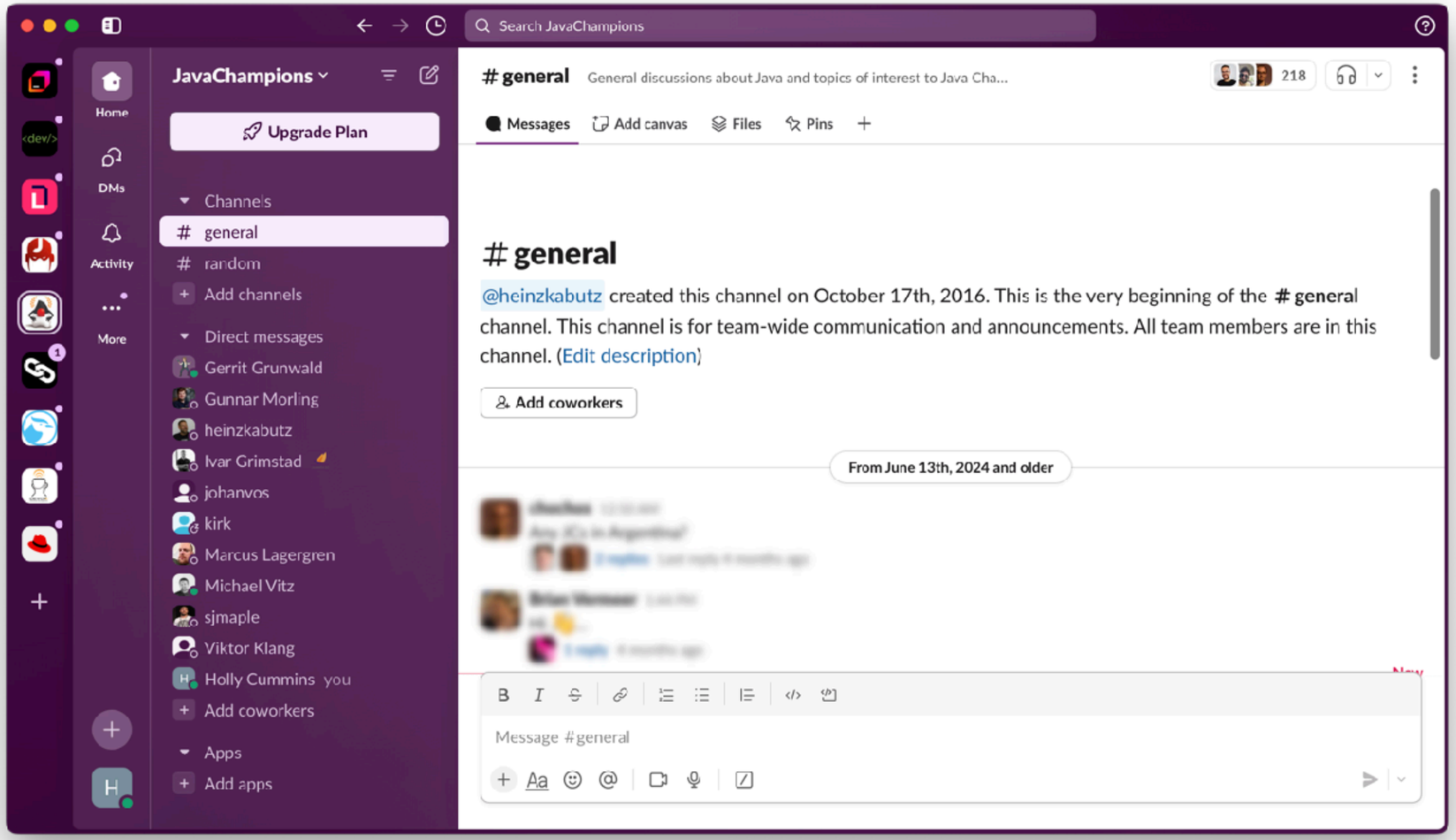
37%

... so the engine doesn't wear out:

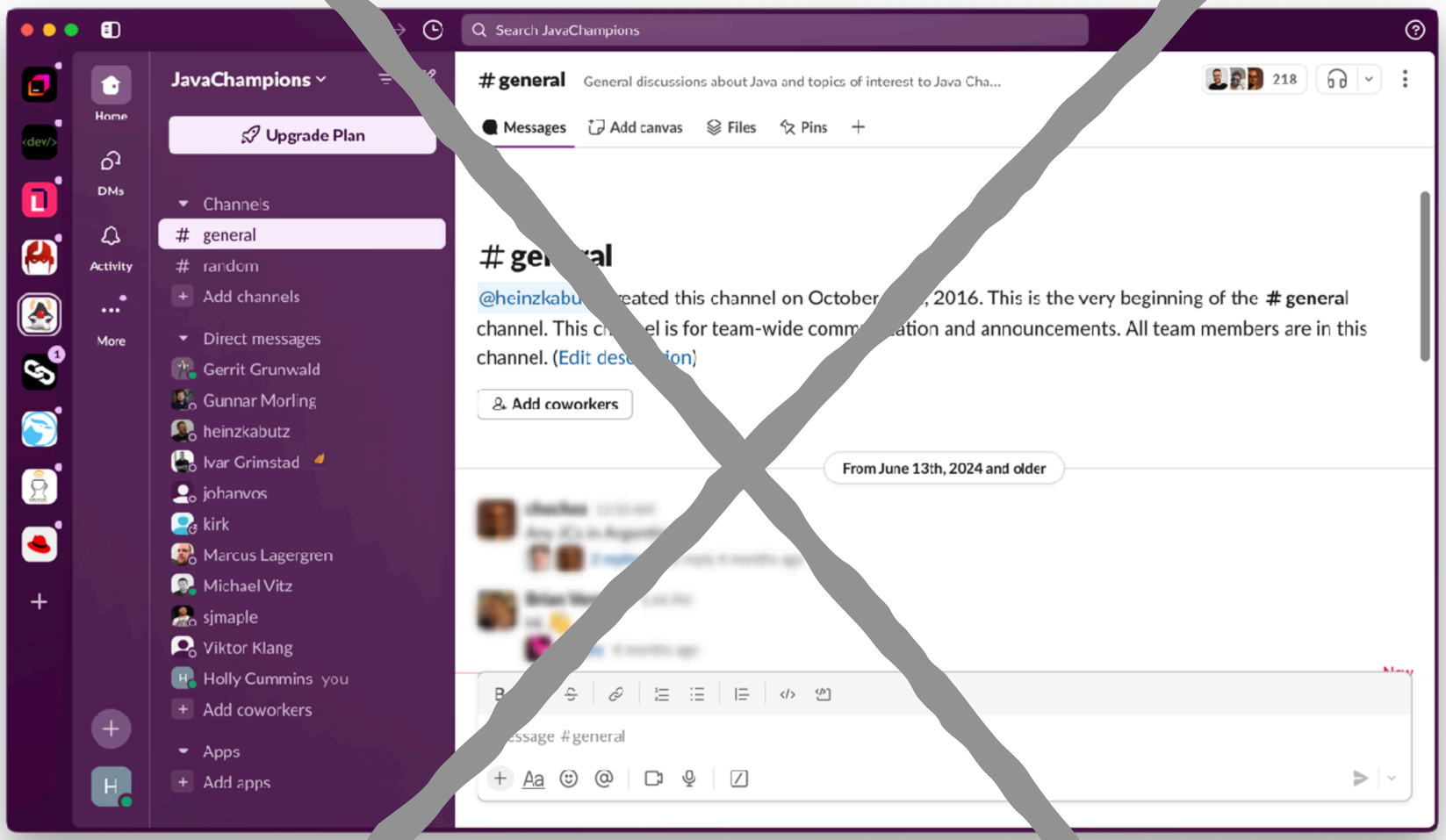
~20%



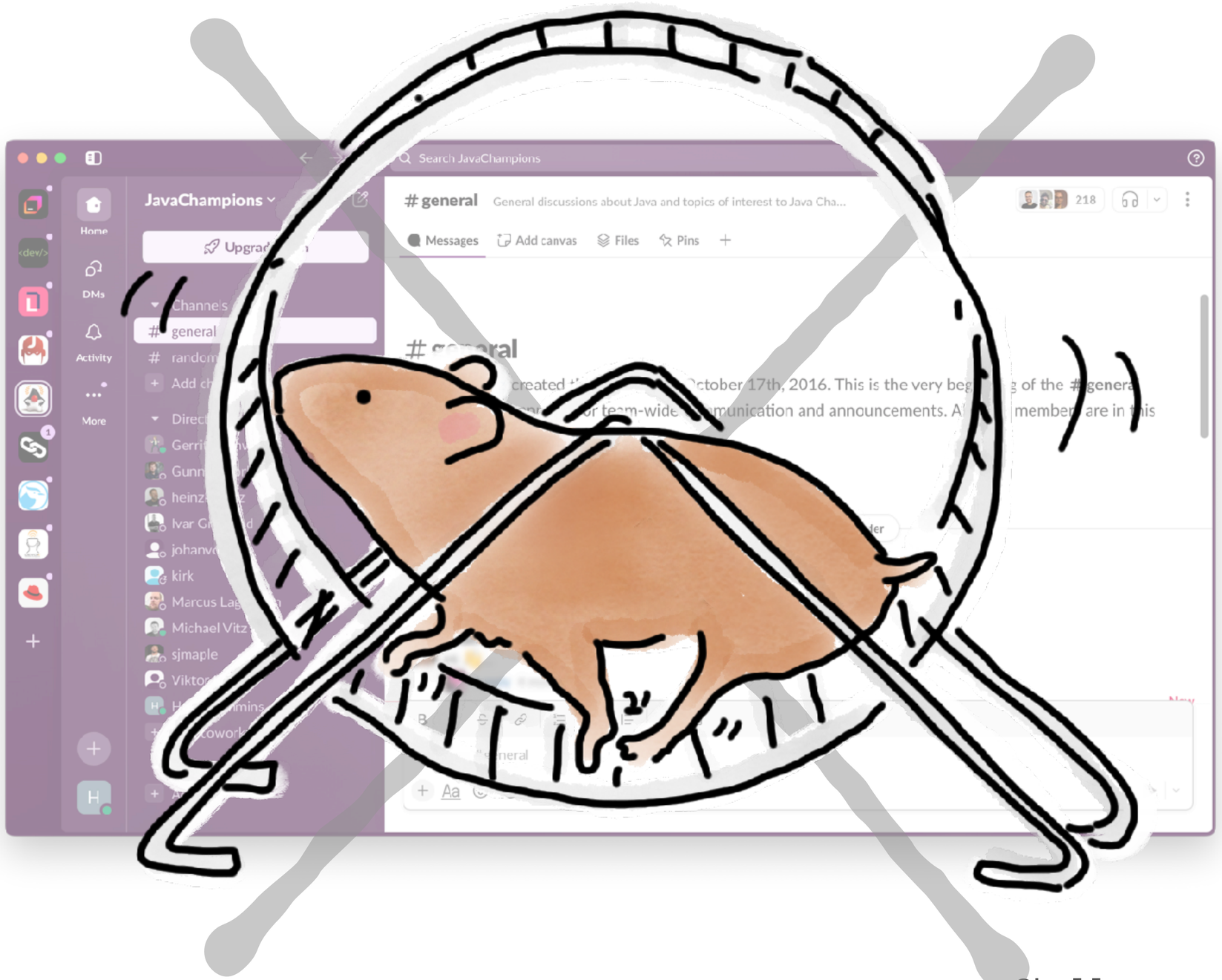
the value of slack



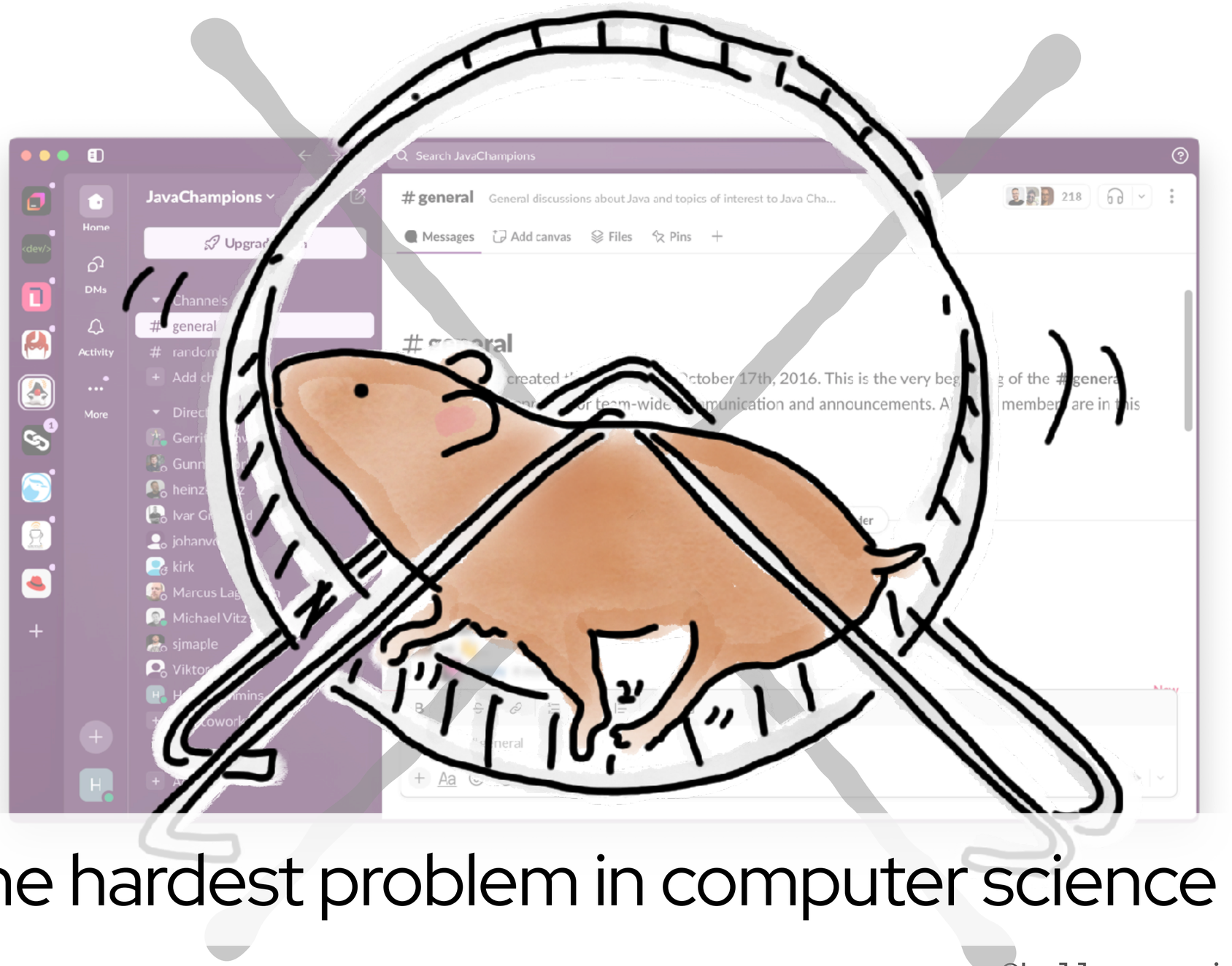
the value of slack



the value of slack



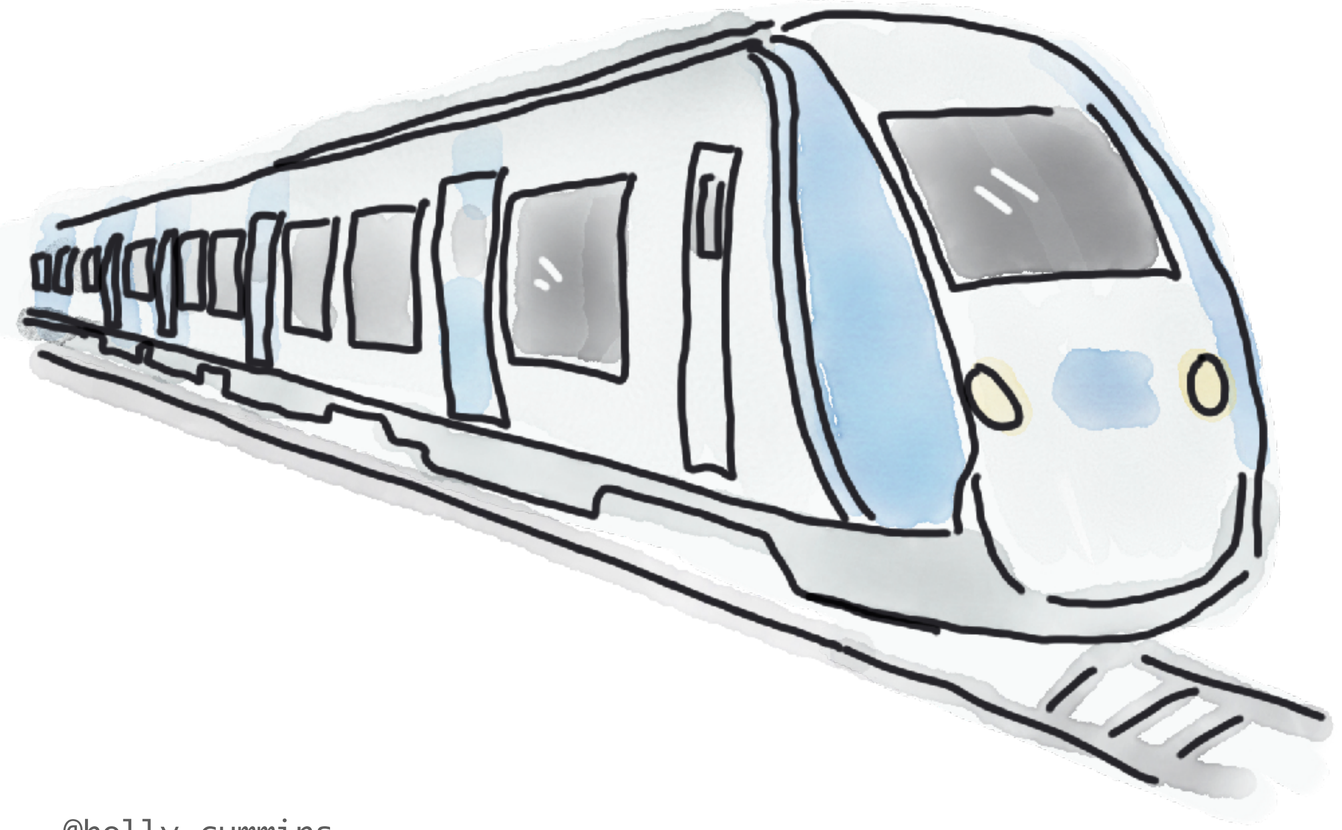
the value of slack

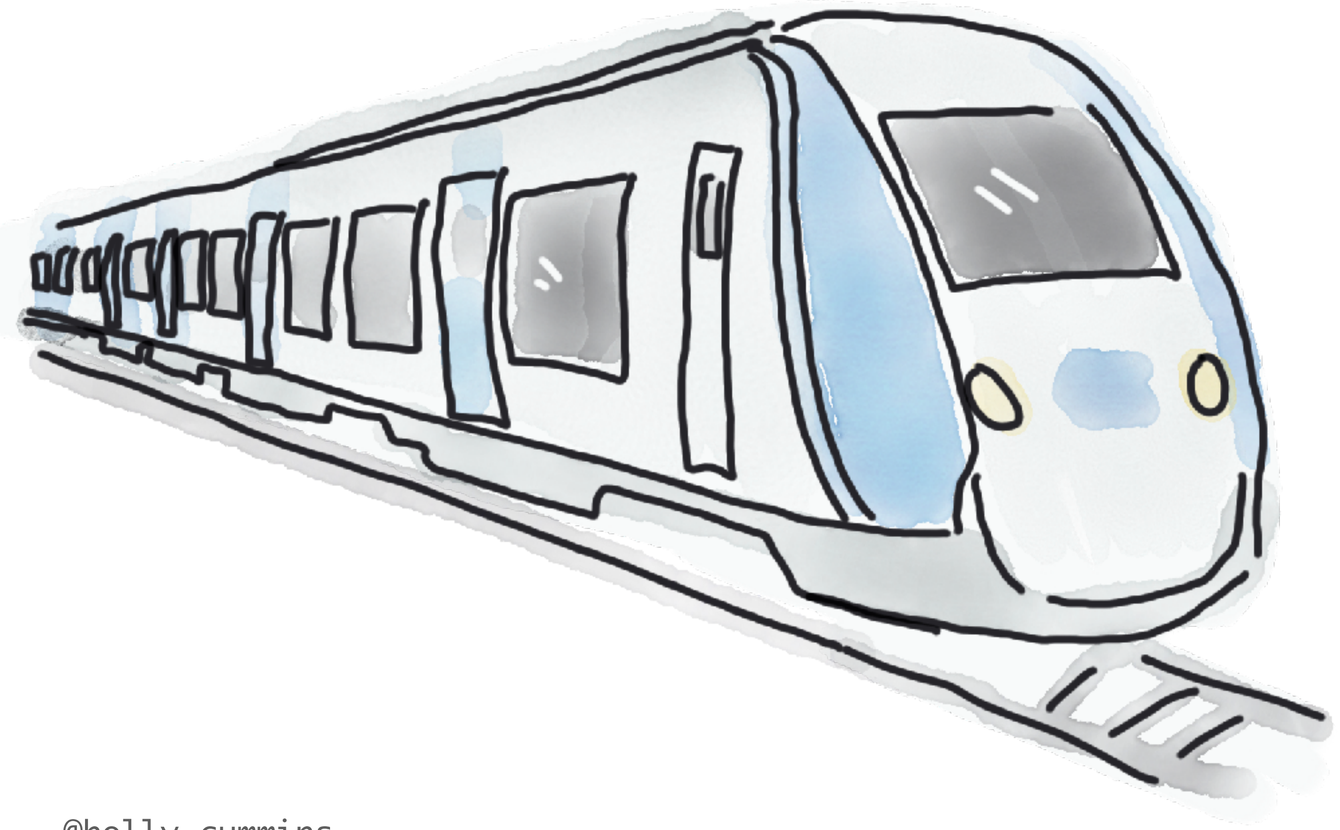


naming is still the hardest problem in computer science

the value of slack







speed



speed
vs
resiliency



less efficient

(more legs than needed)



less efficient

(more legs than needed)



highly efficient

(optimum number of legs)



less efficient

(more legs than needed)



resilient

highly efficient

(optimum number of legs)



less efficient

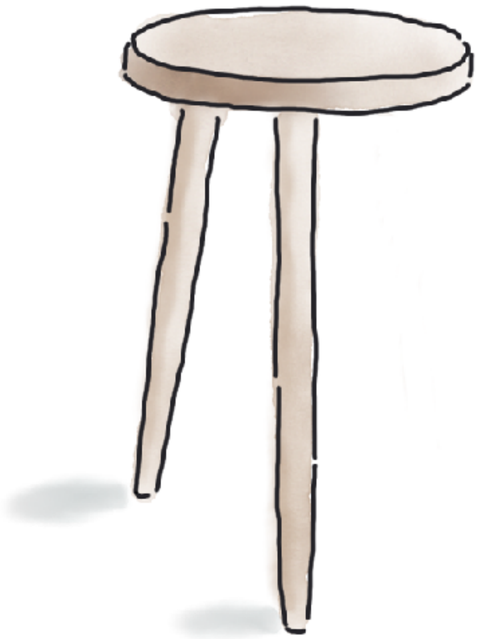
(more legs than needed)



resilient

highly efficient

(optimum number of legs)



less efficient

(more legs than needed)



resilient

highly efficient

(optimum number of legs)



no resiliency

less efficient
(more legs than needed)

highly efficient
(optimum number of legs)



resiliency
lowers
efficiency



resilient

no resiliency



it’s the same for people



it's the same for people

all work and no play is ...
not efficient

Adrian Cockcroft boosted

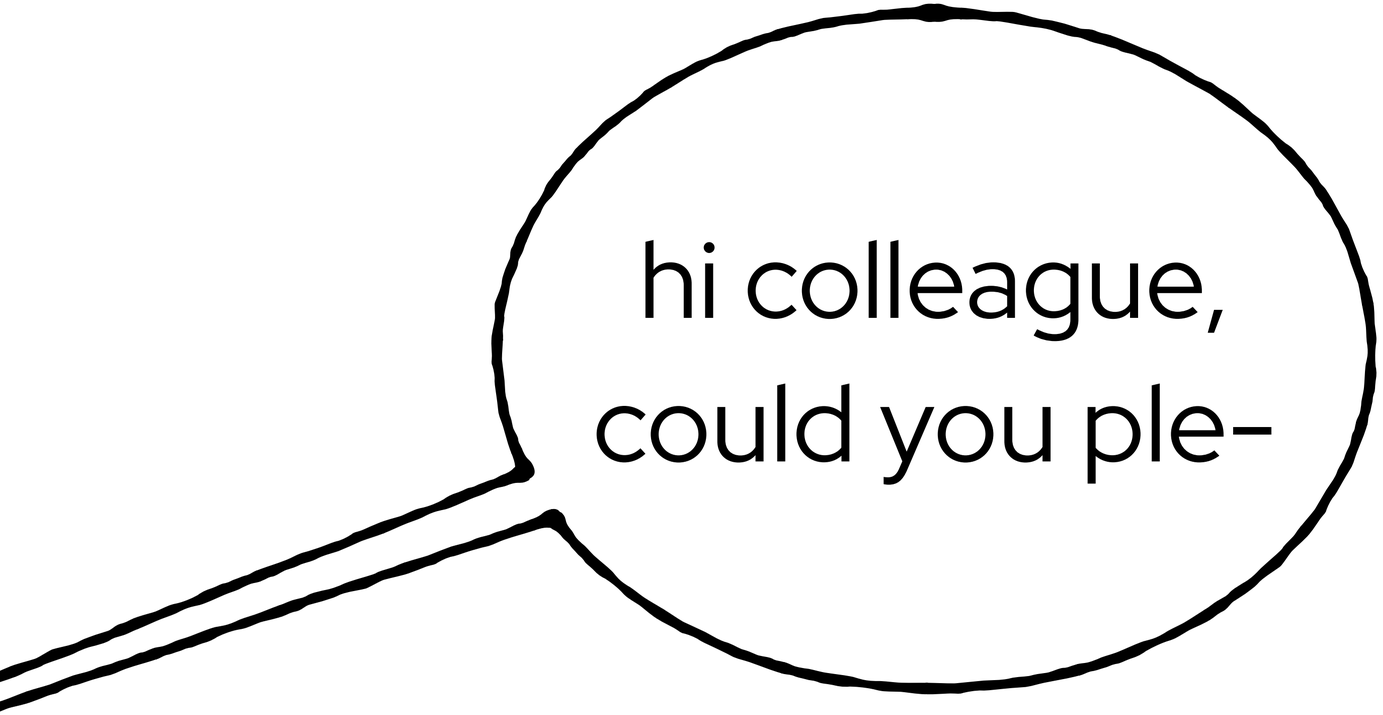


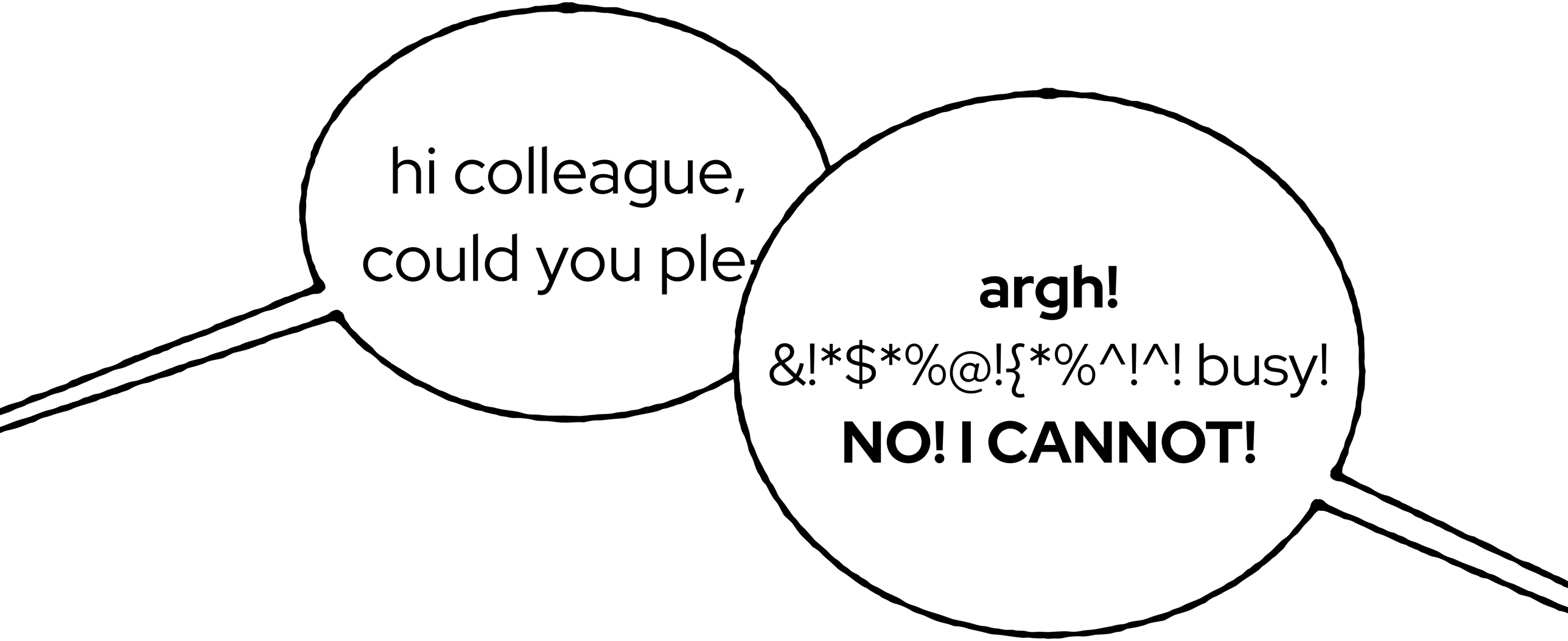
Matt Weagle
@mweagle

2h

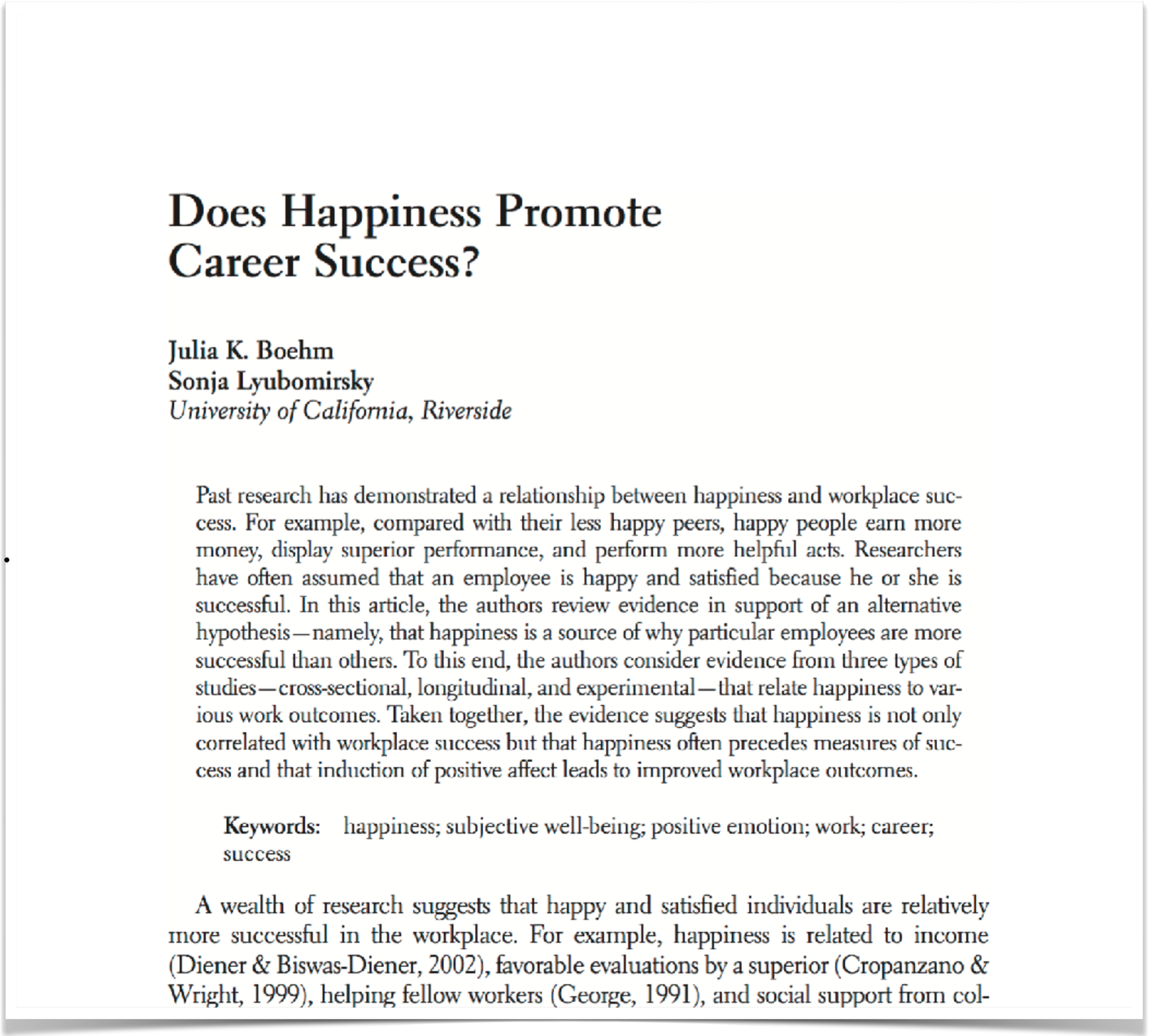
That…sounds like a problem.

“When I led Splunk’s Observability business, 90% of the SRE team resigned within six months due to burnout from on-call duties.”


hi colleague,
could you ple–


hi colleague,
could you ple-
argh!
&!*$*%@!{*%^!^! busy!
NO! I CANNOT!

research shows:
not-work is good for business.



Does Happiness Promote Career Success?

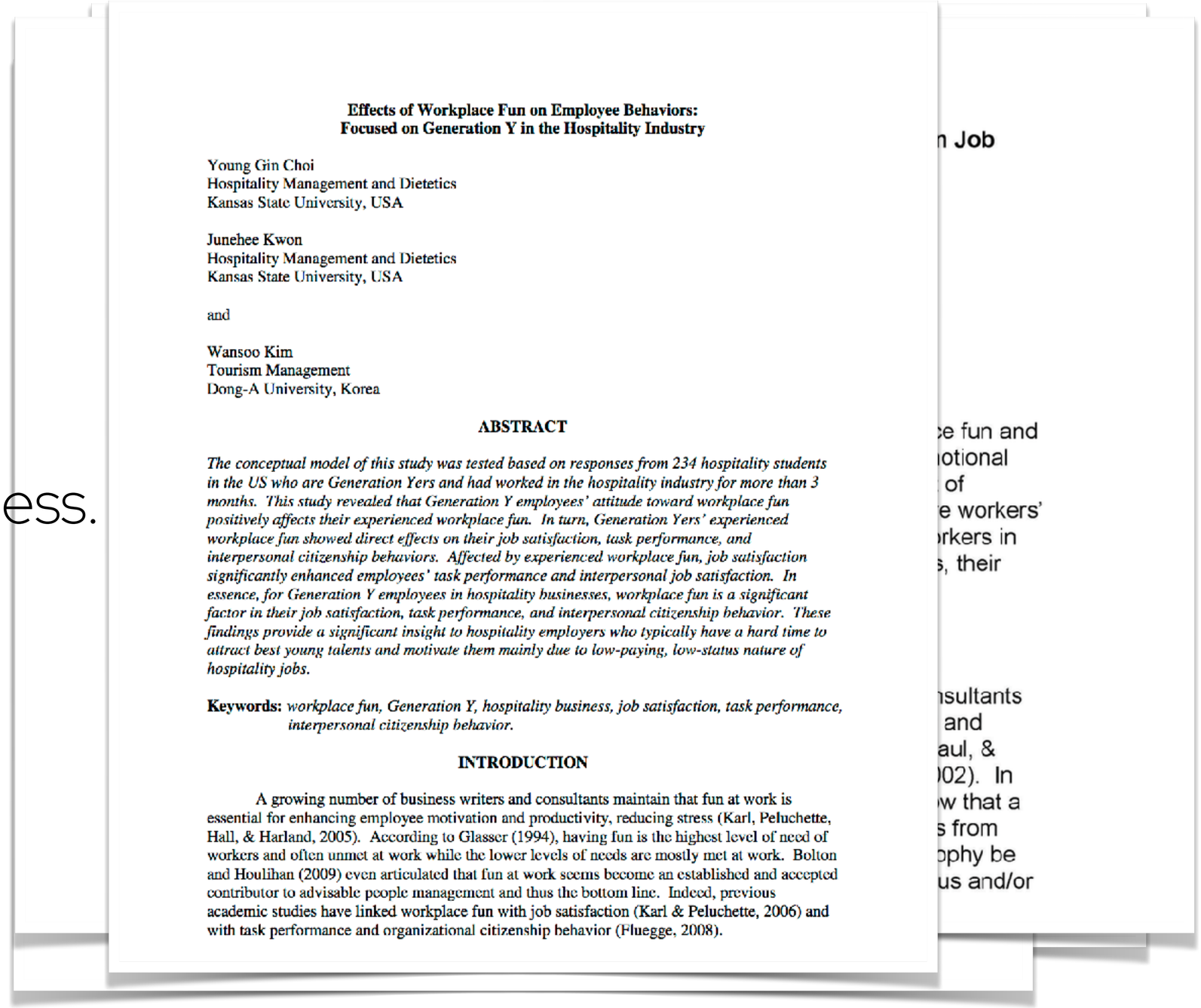
Julia K. Boehm
Sonja Lyubomirsky
University of California, Riverside

Past research has demonstrated a relationship between happiness and workplace success. For example, compared with their less happy peers, happy people earn more money, display superior performance, and perform more helpful acts. Researchers have often assumed that an employee is happy and satisfied because he or she is successful. In this article, the authors review evidence in support of an alternative hypothesis—namely, that happiness is a source of why particular employees are more successful than others. To this end, the authors consider evidence from three types of studies—cross-sectional, longitudinal, and experimental—that relate happiness to various work outcomes. Taken together, the evidence suggests that happiness is not only correlated with workplace success but that happiness often precedes measures of success and that induction of positive affect leads to improved workplace outcomes.

Keywords: happiness; subjective well-being; positive emotion; work; career; success

A wealth of research suggests that happy and satisfied individuals are relatively more successful in the workplace. For example, happiness is related to income (Diener & Biswas-Diener, 2002), favorable evaluations by a superior (Cropanzano & Wright, 1999), helping fellow workers (George, 1991), and social support from col-

research shows:
not-work is good for business.



Effects of Workplace Fun on Employee Behaviors:
Focused on Generation Y in the Hospitality Industry

Young Gin Choi
Hospitality Management and Dietetics
Kansas State University, USA

Junehee Kwon
Hospitality Management and Dietetics
Kansas State University, USA

and

Wansoo Kim
Tourism Management
Dong-A University, Korea

ABSTRACT

The conceptual model of this study was tested based on responses from 234 hospitality students in the US who are Generation Yers and had worked in the hospitality industry for more than 3 months. This study revealed that Generation Y employees' attitude toward workplace fun positively affects their experienced workplace fun. In turn, Generation Yers' experienced workplace fun showed direct effects on their job satisfaction, task performance, and interpersonal citizenship behaviors. Affected by experienced workplace fun, job satisfaction significantly enhanced employees' task performance and interpersonal job satisfaction. In essence, for Generation Y employees in hospitality businesses, workplace fun is a significant factor in their job satisfaction, task performance, and interpersonal citizenship behavior. These findings provide a significant insight to hospitality employers who typically have a hard time to attract best young talents and motivate them mainly due to low-paying, low-status nature of hospitality jobs.

Keywords: *workplace fun, Generation Y, hospitality business, job satisfaction, task performance, interpersonal citizenship behavior.*

INTRODUCTION

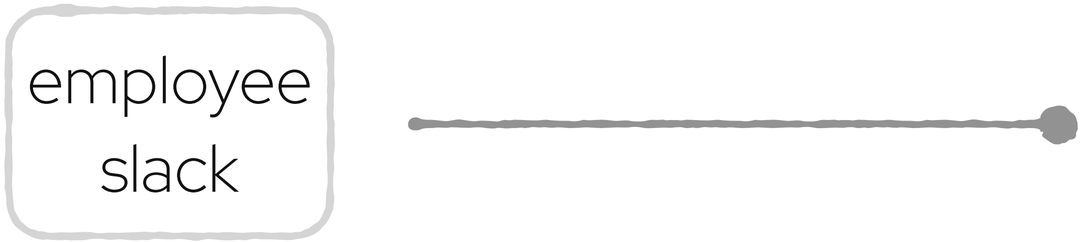
A growing number of business writers and consultants maintain that fun at work is essential for enhancing employee motivation and productivity, reducing stress (Karl, Peluchette, Hall, & Harland, 2005). According to Glasser (1994), having fun is the highest level of need of workers and often unmet at work while the lower levels of needs are mostly met at work. Bolton and Houlihan (2009) even articulated that fun at work seems become an established and accepted contributor to advisable people management and thus the bottom line. Indeed, previous academic studies have linked workplace fun with job satisfaction (Karl & Peluchette, 2006) and with task performance and organizational citizenship behavior (Fluegge, 2008).

studies show...

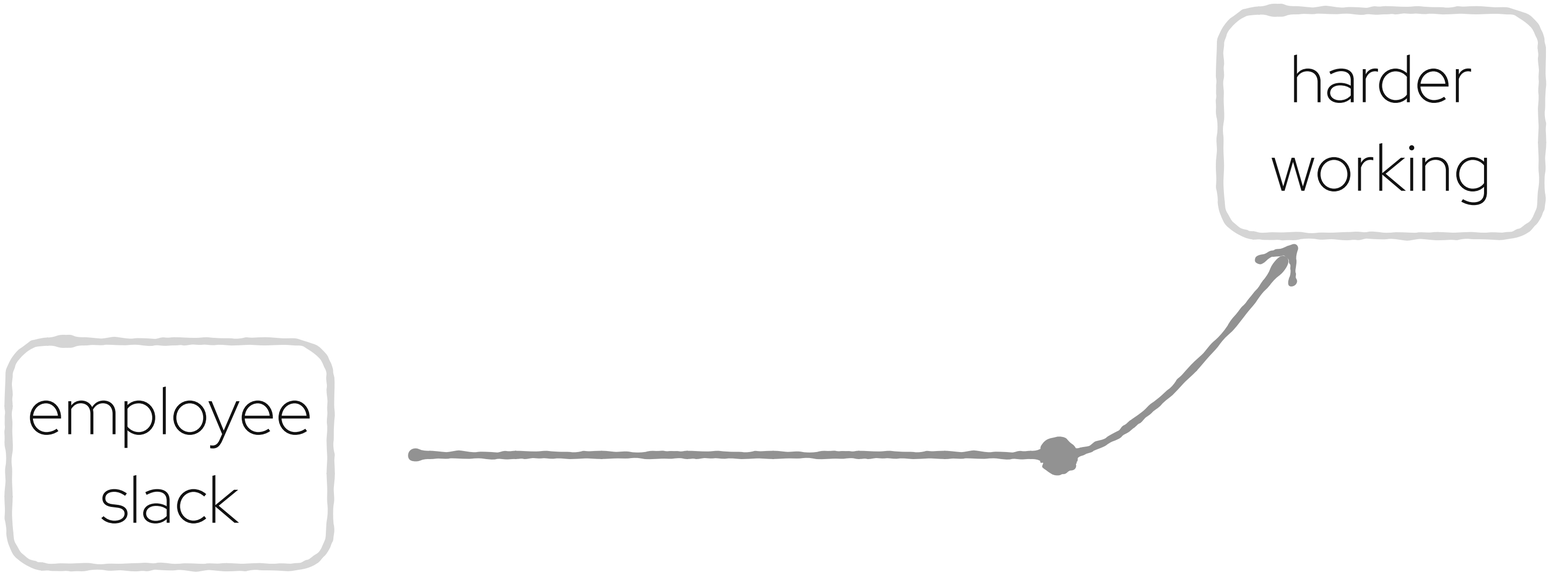
studies show...

employee
slack

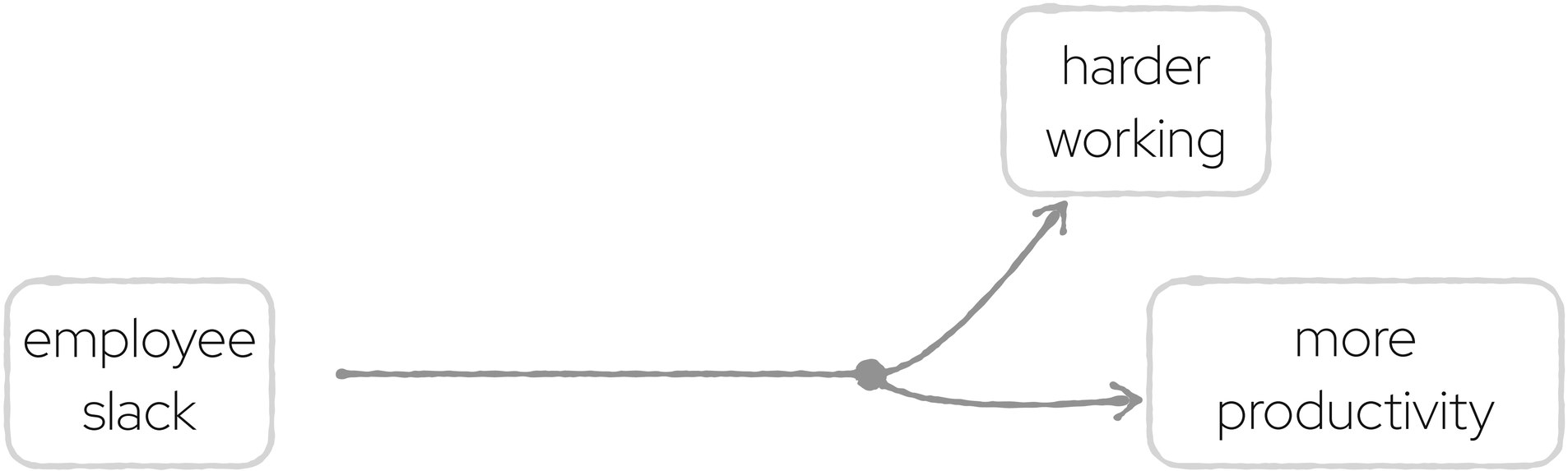
studies show...



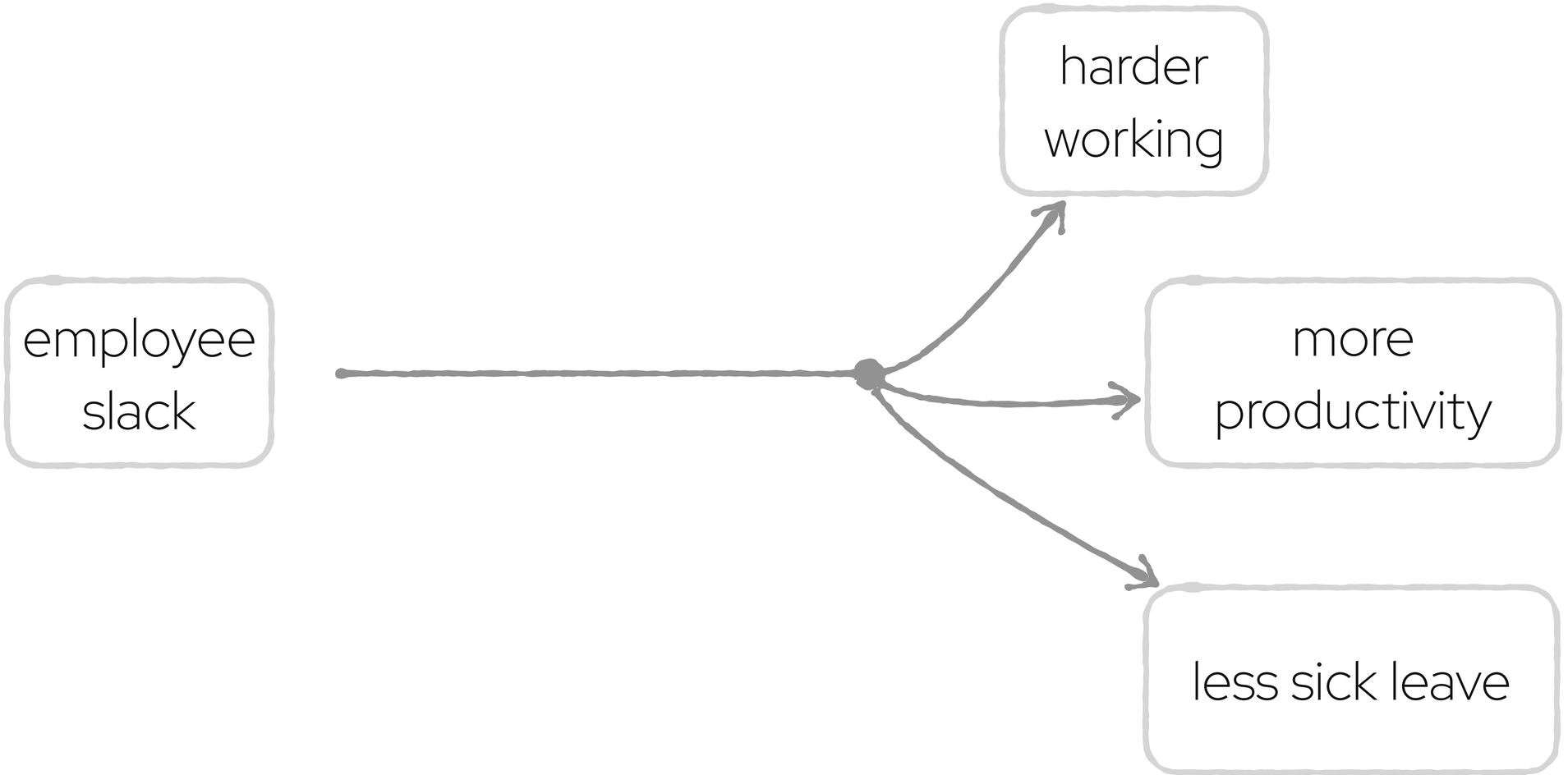
studies show...



studies show...



studies show...



Harvard Business Review

PRODUCTIVITY

Positive Intelligence

by Shawn Achor

FROM THE JANUARY–FEBRUARY 2012 ISSUE

SUMMARY SAVE SHARE HH TEXT SIZE PRINT $8.95 BUY COPIES

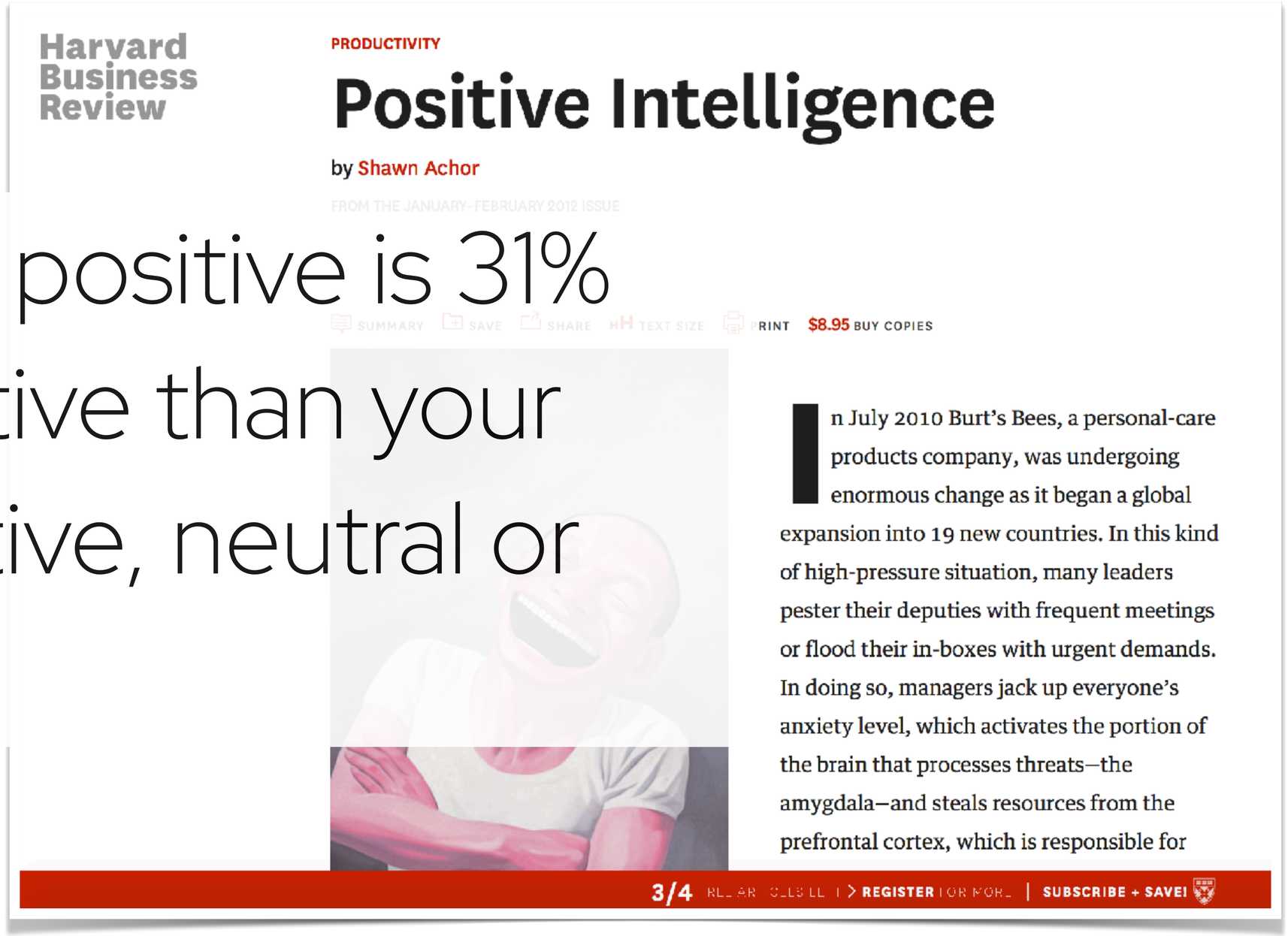


In July 2010 Burt's Bees, a personal-care products company, was undergoing enormous change as it began a global expansion into 19 new countries. In this kind of high-pressure situation, many leaders pester their deputies with frequent meetings or flood their in-boxes with urgent demands. In doing so, managers jack up everyone's anxiety level, which activates the portion of the brain that processes threats—the amygdala—and steals resources from the prefrontal cortex, which is responsible for

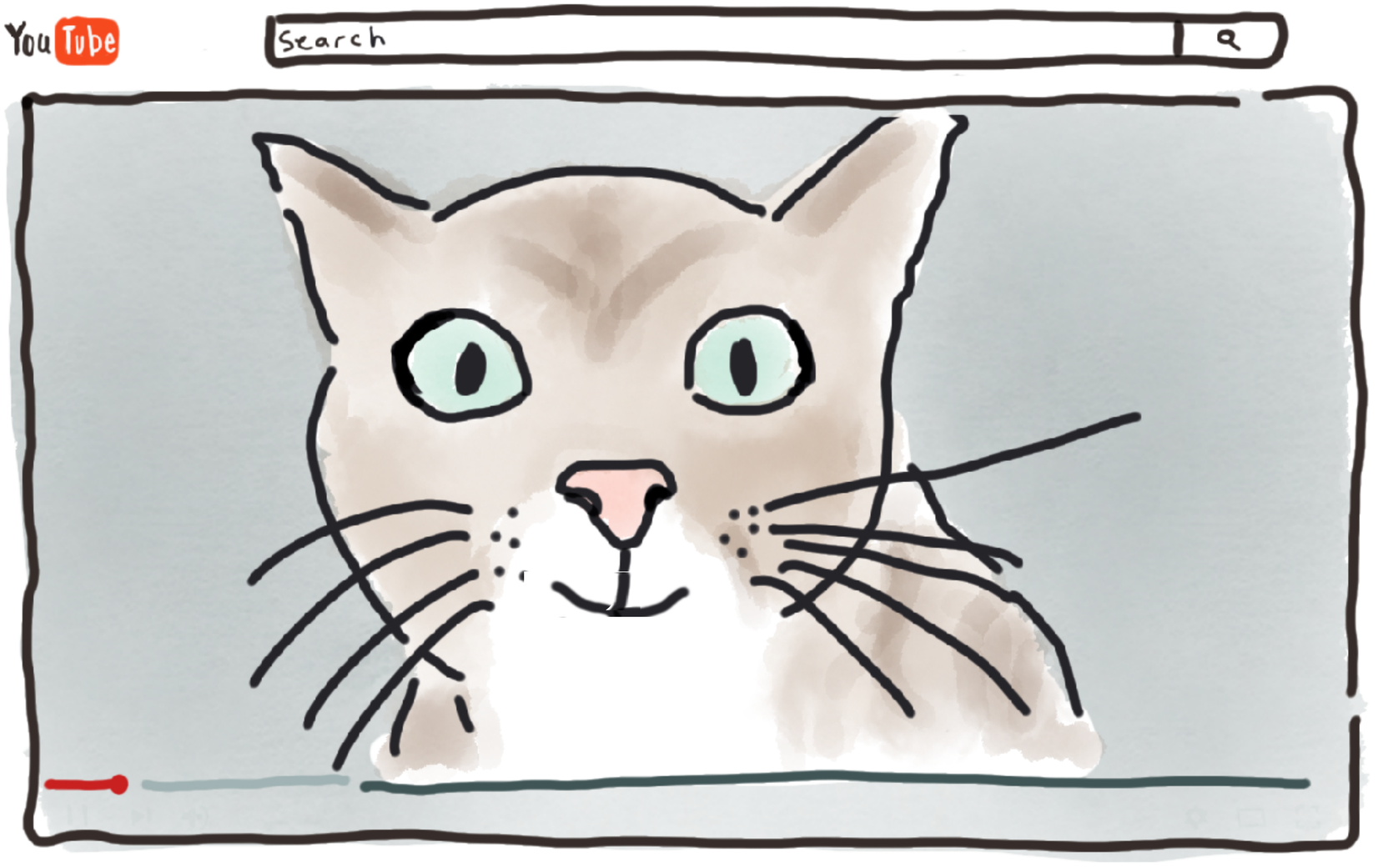
3/4 READ ... > REGISTER FOR MORE | SUBSCRIBE + SAVE!

https://hbr.org/2012/01/positive-intelligence

"Your brain at positive is 31% more productive than your brain at negative, neutral or stressed. "



https://hbr.org/2012/01/positive-intelligence


YouTube
Search

Happiness and Productivity

Andrew J. Oswald*, Eugenio Proto**, and Daniel Sgroi**

*University of Warwick, UK, and IZA Bonn, Germany
**University of Warwick, UK

JOLE 3rd Version: 10 February 2014

Emails: andrew.oswald@warwick.ac.uk; e.proto@warwick.ac.uk; daniel.sgroi@warwick.ac.uk
JEL Classification: D03, J24, C91
Keywords: Well-being; productivity; happiness; personnel economics.
Address: Department of Economics, University of Warwick, Coventry CV4 7AL, United Kingdom.
Telephone: (+44) 02476 523510
Acknowledgements: For their suggestions, we thank the referees and the editor Paul Oyer. For fine research assistance, and valuable discussions, we are indebted to Malena Digiuni, Alex Dobson, Stephen Lovelady, and Lucy Rippon. For advice, we would like to record our deep gratitude to Alice Isen. Insightful suggestions were provided by seminar audiences in Berlin, Birmingham, Bonn, Leicester, Glasgow, HM Treasury London, LSE, Maastricht, PSE Paris, Warwick, York, and Zurich. Special thanks also go to Johannes Abeler, Eve Caroli, Emanuele Castano, Andrew Clark, Alain Cohn, Ernst Fehr, Justina Fischer, Bruno Frey, Dan Gilbert, Amanda Goodall, Greg Jones, Graham Loomes, Rocco Macchiavello, Michel Marechal, Sharun Mukand, Steve Pischke, Nick Powdthavee, Tommaso Reggiani, Daniel Schunk, Claudia Senik, Tania Singer, and Luca Stanca. The first author thanks the University of Zurich for its hospitality and is grateful to the ESRC for a research professorship. The ESRC (through CAGE) and the Leverhulme Trust also provided research support.

https://wrap.warwick.ac.uk/63228/7/WRAP_Oswald_681096.pdf



Happiness and Productivity

Andrew J. Oswald*, Eugenio Proto**, and Daniel Sgroi**

*University of Warwick, UK, and IZA Bonn, Germany
**University of Warwick, UK

JOLE 3rd Version: 10 February 2014

Emails: andrew.oswald@warwick.ac.uk; e.proto@warwick.ac.uk; daniel.sgroi@wa wick.ac.uk
JEL Classification: D03, J24, C91
Keywords: Well-being; productivity; happiness; personnel economics.
Address: Department of Economics, University of Warwick, Coventry CV4 7AL, U ited Kingdom.
Telephone: (+44) 02476 523510
Acknowledgements: For their suggestions, we thank the referees and the editor Pau Oyer. For fine research assistance, and valuable discussions, we are indebted to Malena Digiun , Alex Dobson, Stephen Lovelady, and Lucy Rippon. For advice, we would like to record our deep gratitude to Alice Isen. Insightful suggestions were provided by seminar audiences in Berl n, Birmingham, Bonn, Leicester, Glasgow, HM Treasury London, LSE, Maastricht, PSE Paris, Wa wick, York, and Zurich. Special thanks also go to Johannes Abeler, Eve Caroli, Emanuele Castano Andrew Clark, Alain Cohn, Ernst Fehr, Justina Fischer, Bruno Frey, Dan Gilbert, Amanda Goodall, Greg Jones, Graham Loomes, Rocco Macchiavello, Michel Marechal, Sharun Mukand, Steve Pischke, Nick Powdthavee, Tommaso Reggiani, Daniel Schunk, Claudia Senik, Tania Singer, and Luca Stanca. The first author thanks the University of Zurich for its hospitality and is grateful to the ESRC for a research professorship. The ESRC (through CAGE) and the Leverhulme Trust also provided research support.

"Individuals [who just watched a comedy video] have approximately

12%

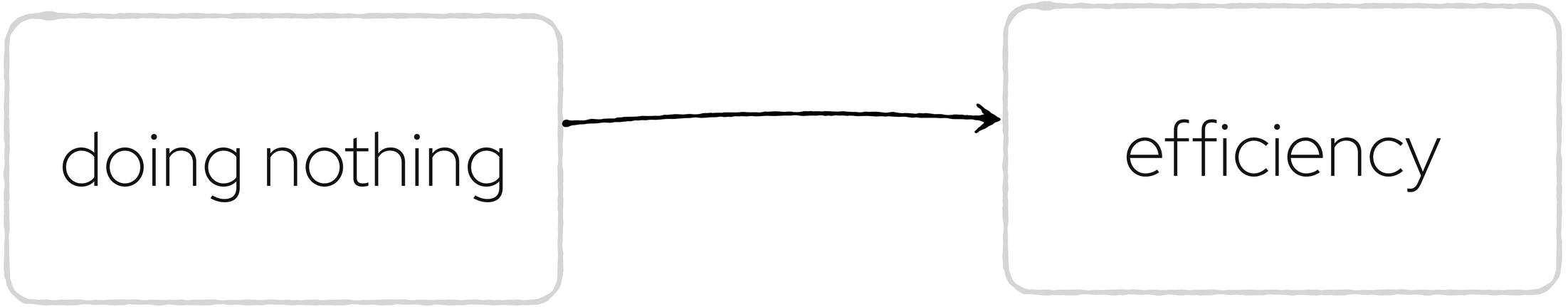
greater productivity."

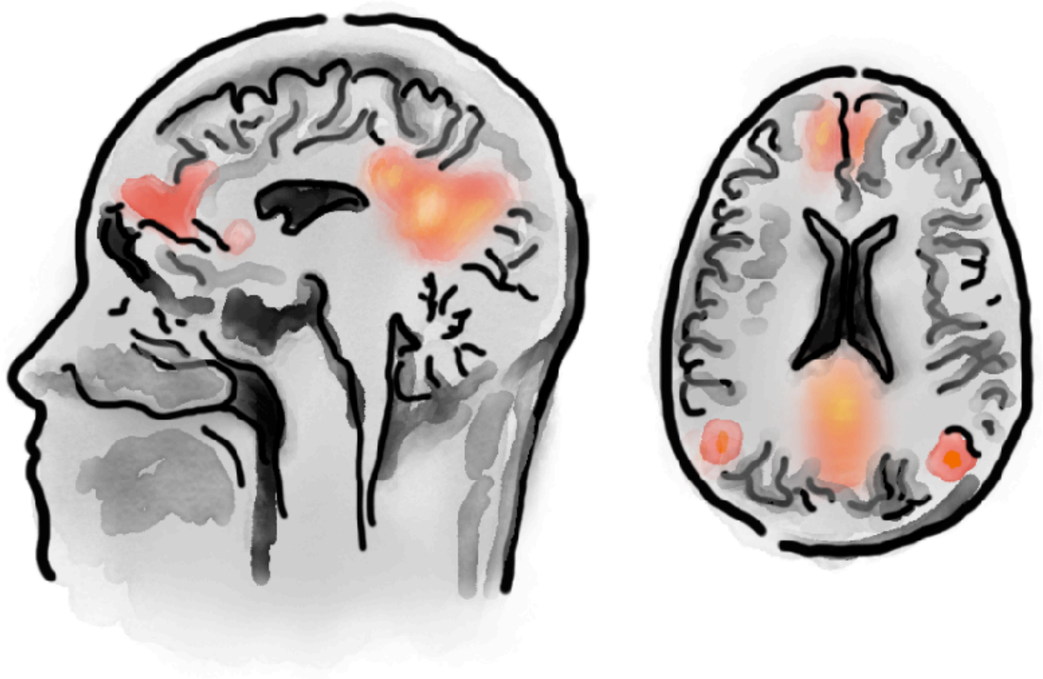
https://wrap.warwick.ac.uk/63228/7/WRAP_Oswald_681096.pdf

play


play
efficiency

doing nothing


doing nothing
efficiency



the default mode network

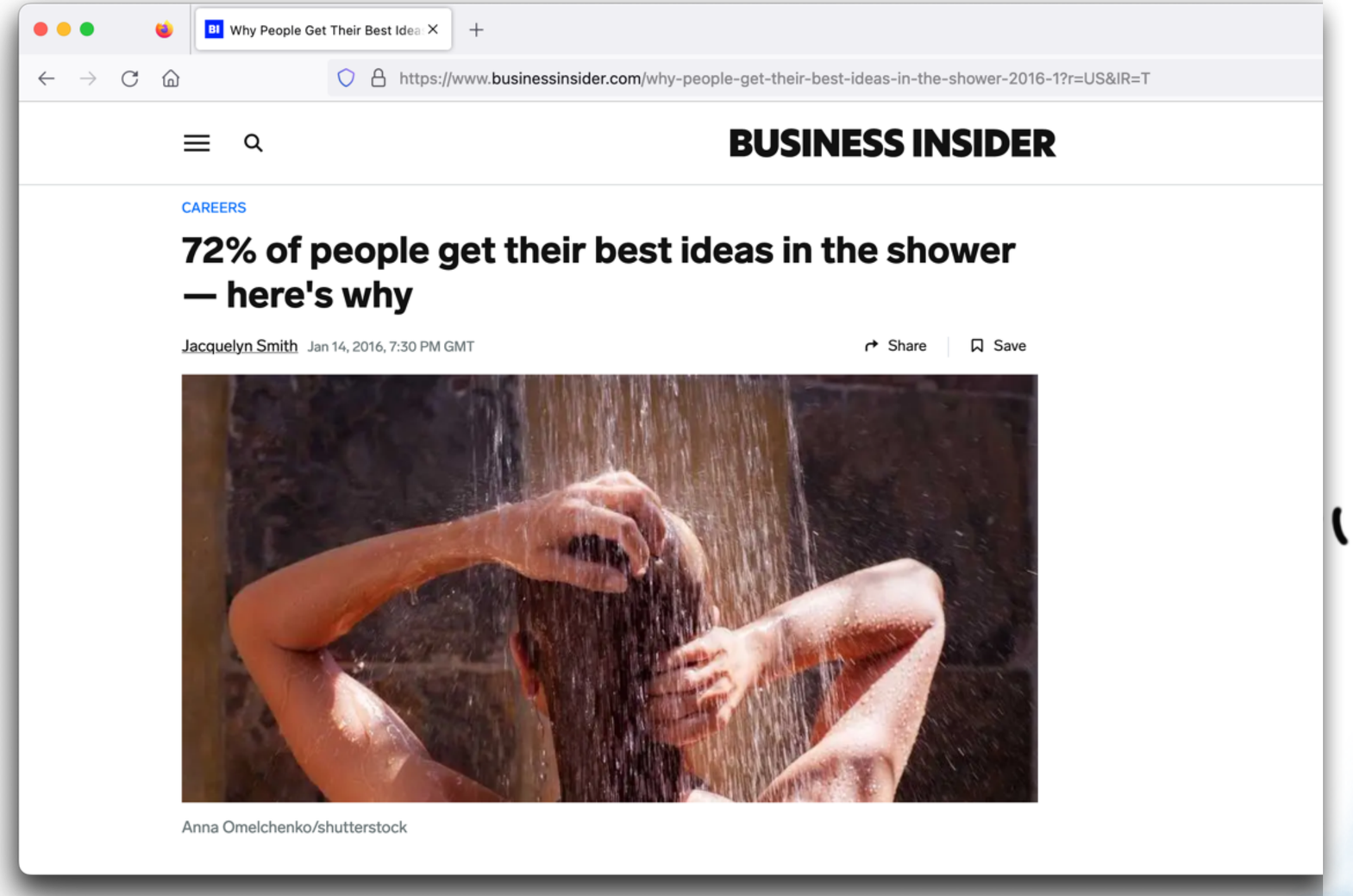
idle minds can solve hard problems



@holly_cummins

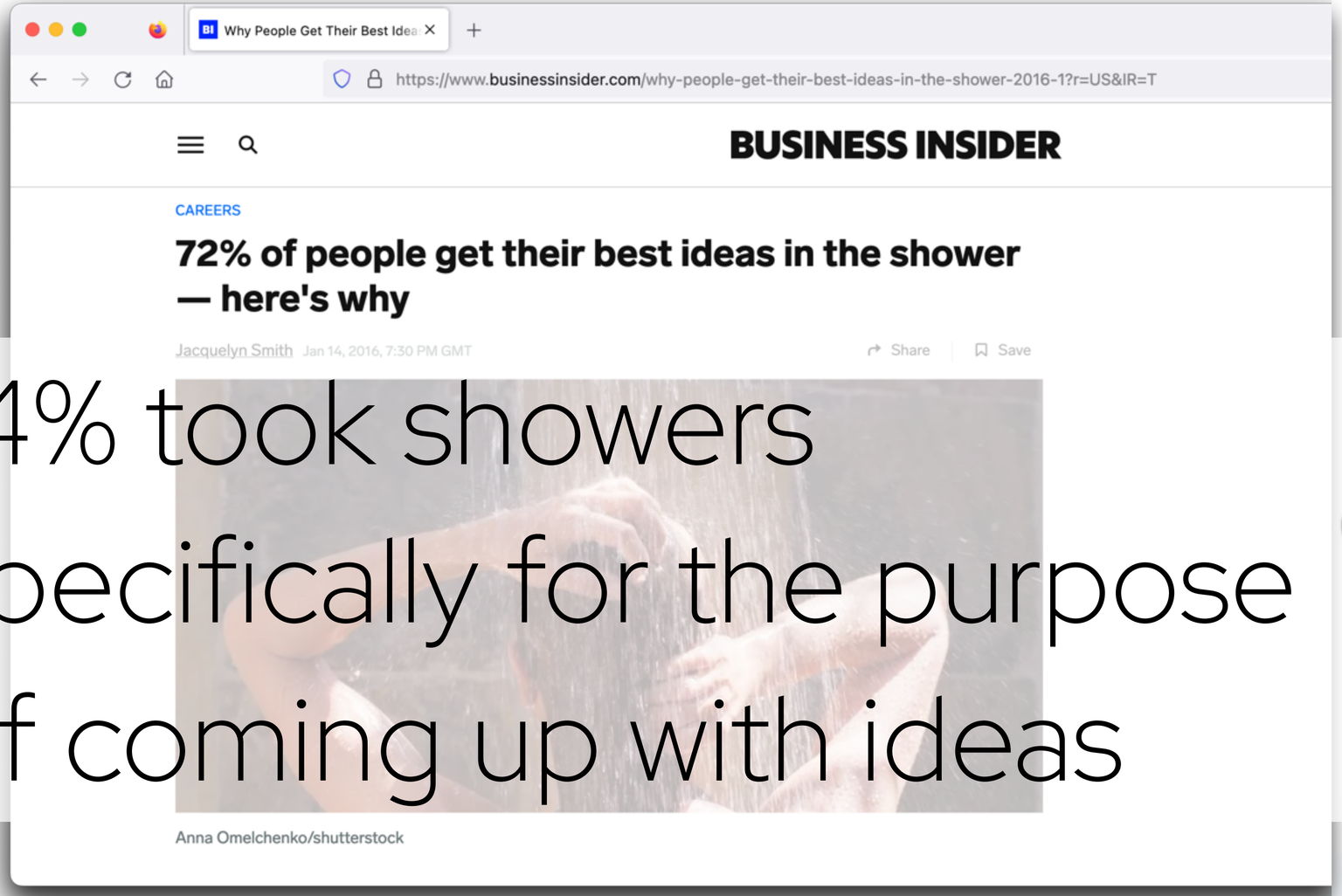


@holly_cummins





@holly_cummins



14% took showers specifically for the purpose of coming up with ideas



psychology says we need idle time; maths agrees



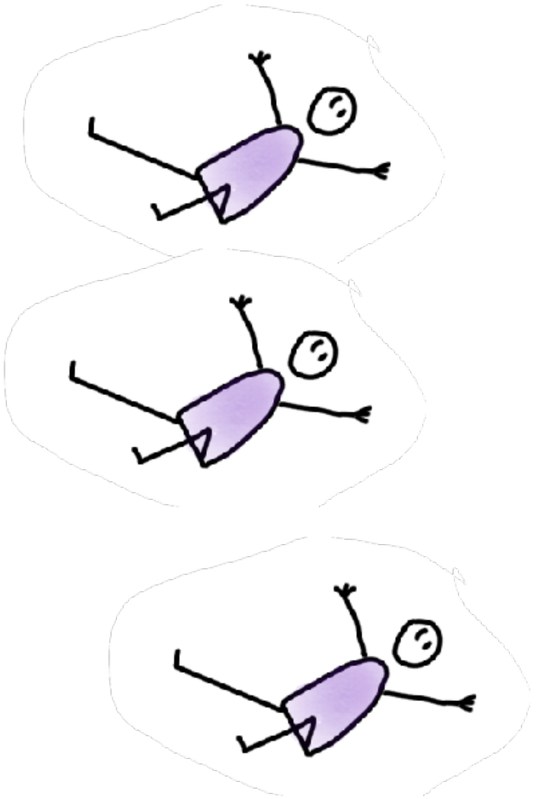
psychology says we need idle time; maths agrees

queueing theory says systems need to run under-capacity to function



queuing theory basics

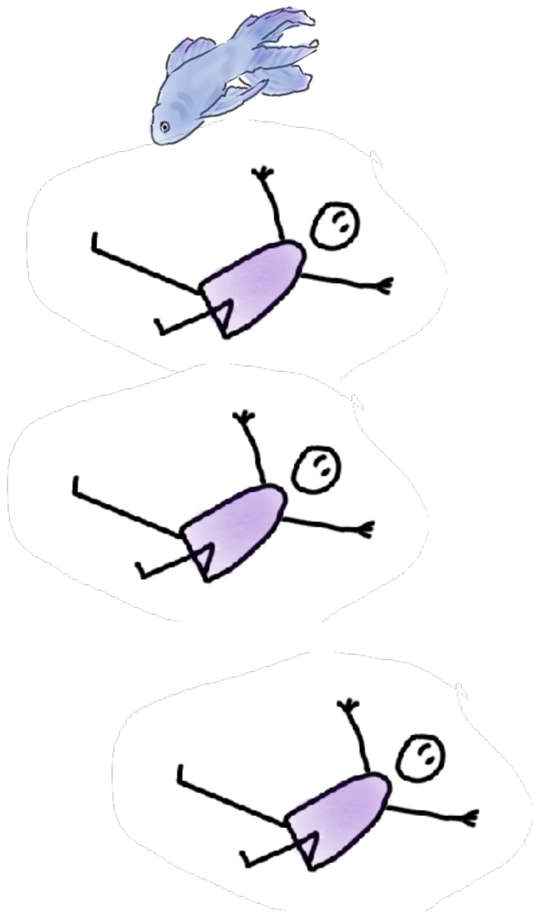
queuing theory basics



arrival process

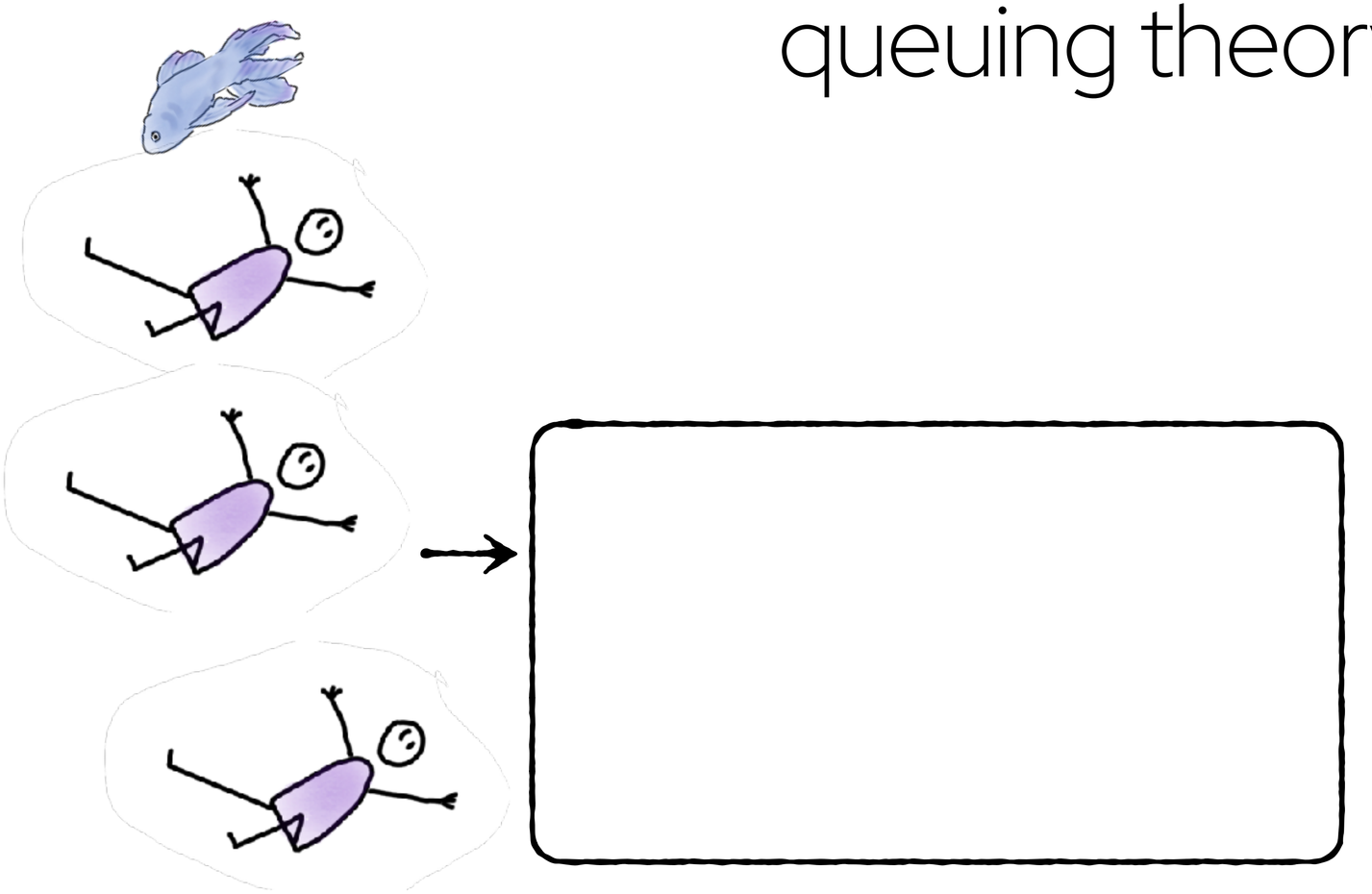
Poisson distribution

queuing theory basics



arrival process

queuing theory basics

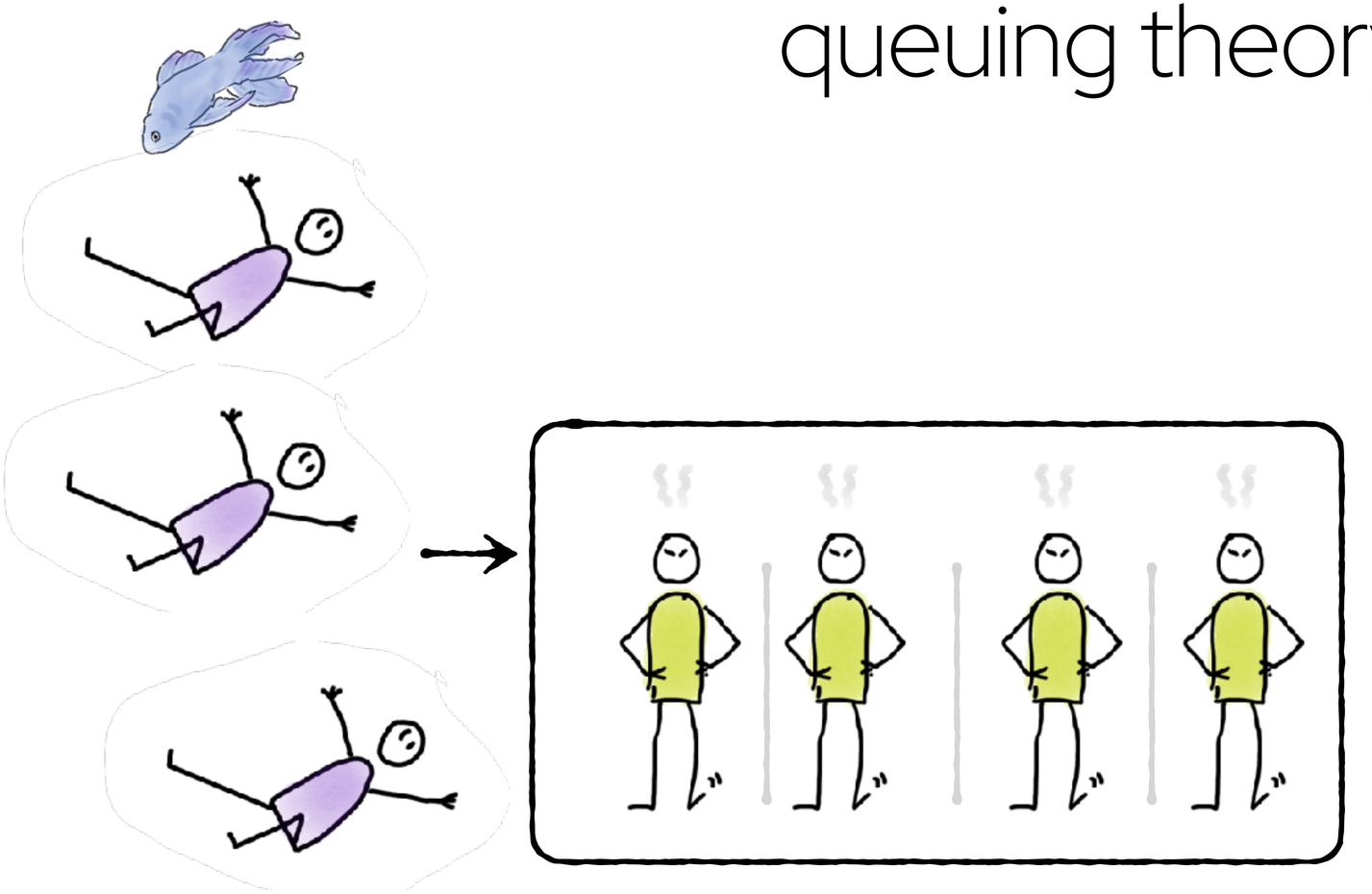


arrival process

queue

Poisson distribution

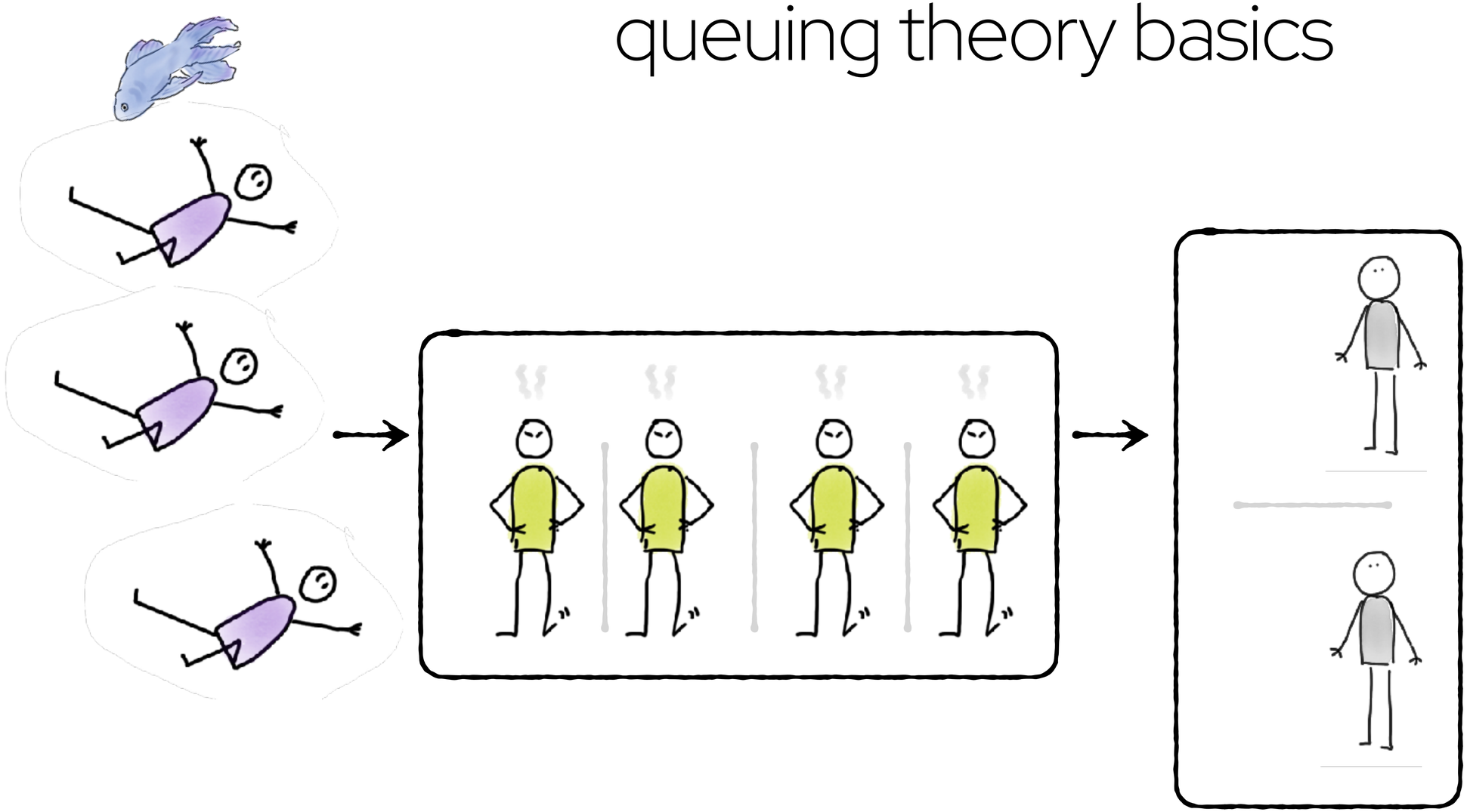
queuing theory basics



arrival process

queue

queuing theory basics



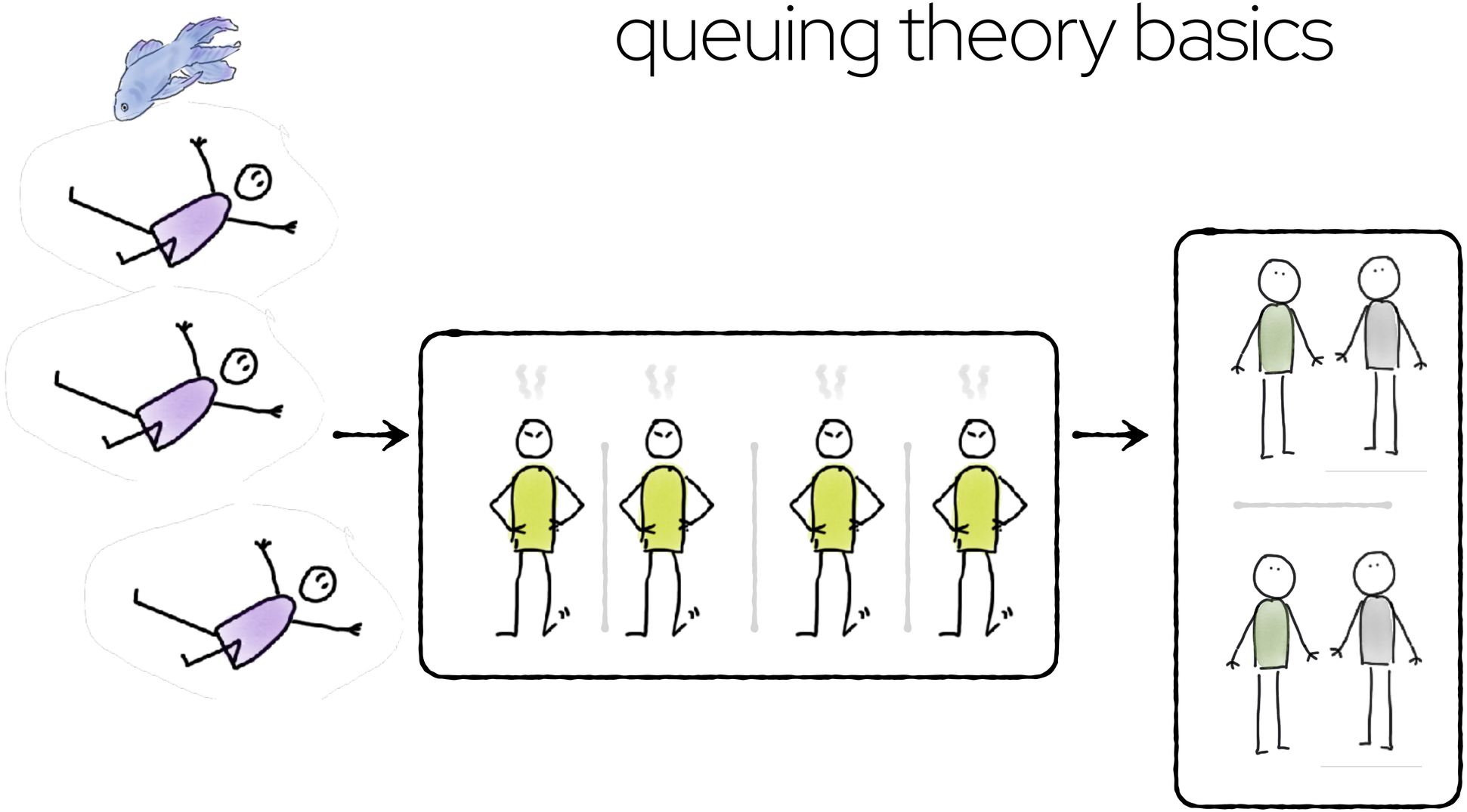
arrival process

queue

servers

Poisson distribution

queuing theory basics



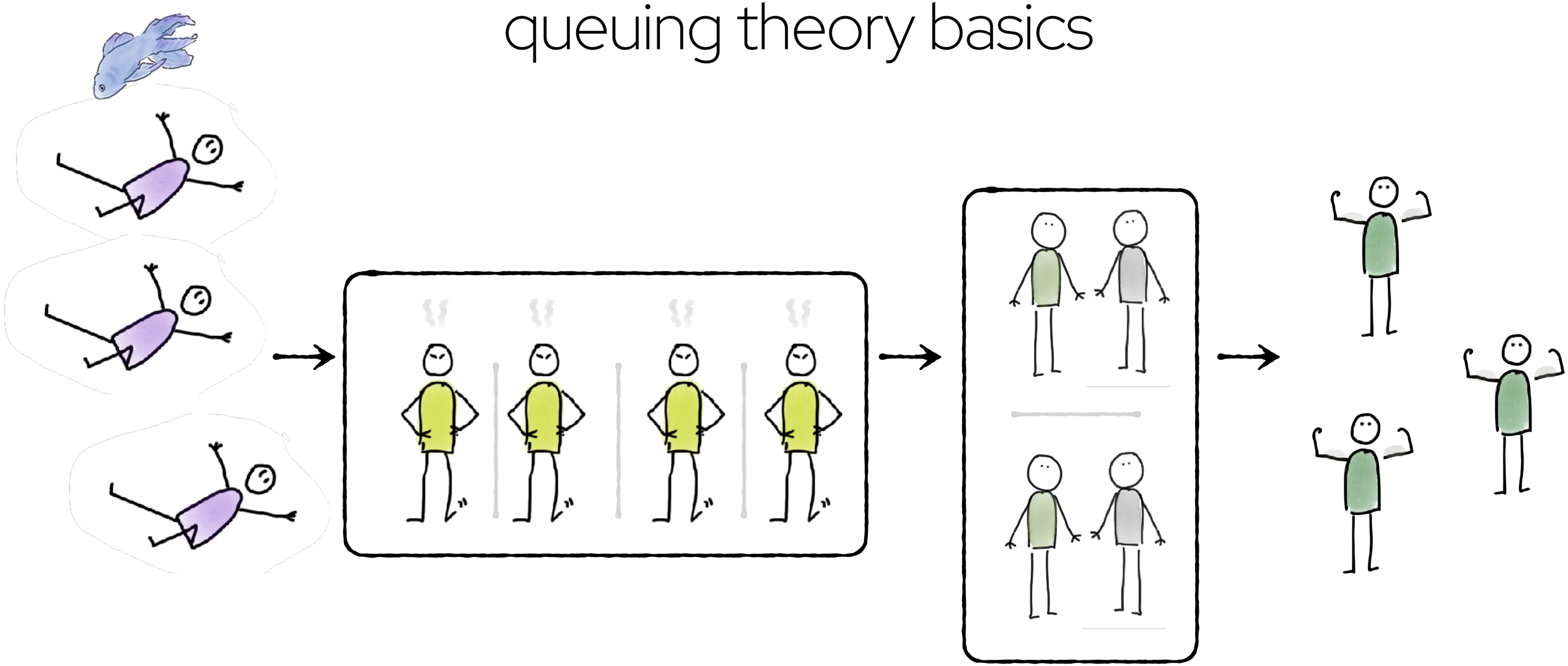
arrival process

queue

servers

Poisson distribution

queuing theory basics



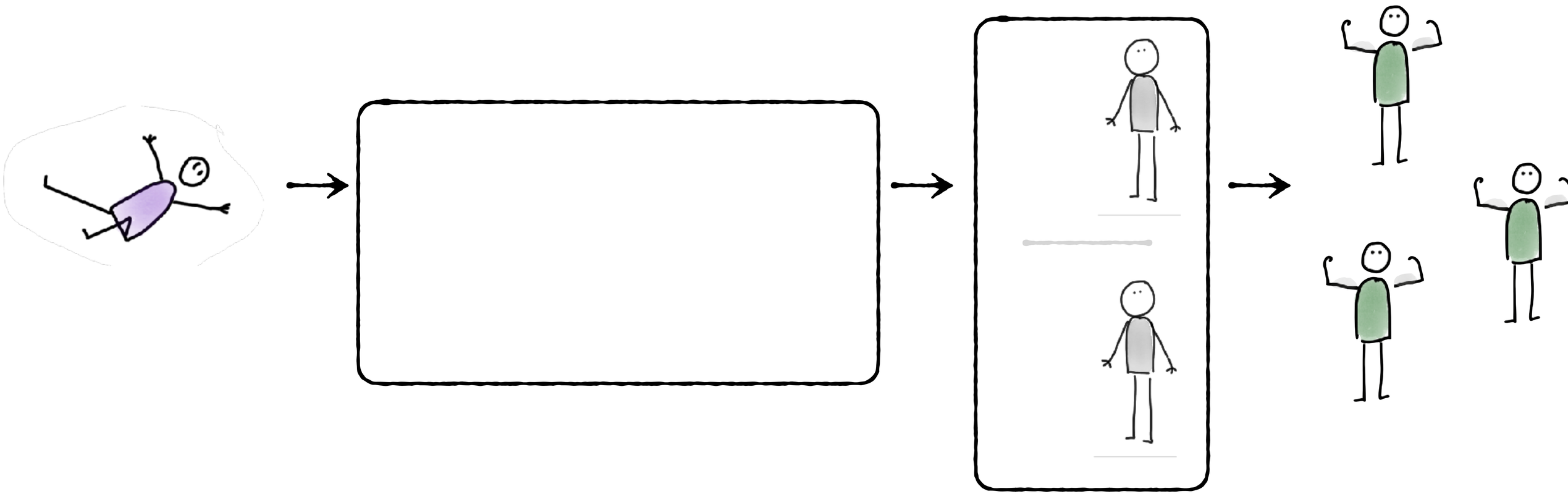
arrival process

queue

servers

completed work

if arrival rates are low, servers will be idle



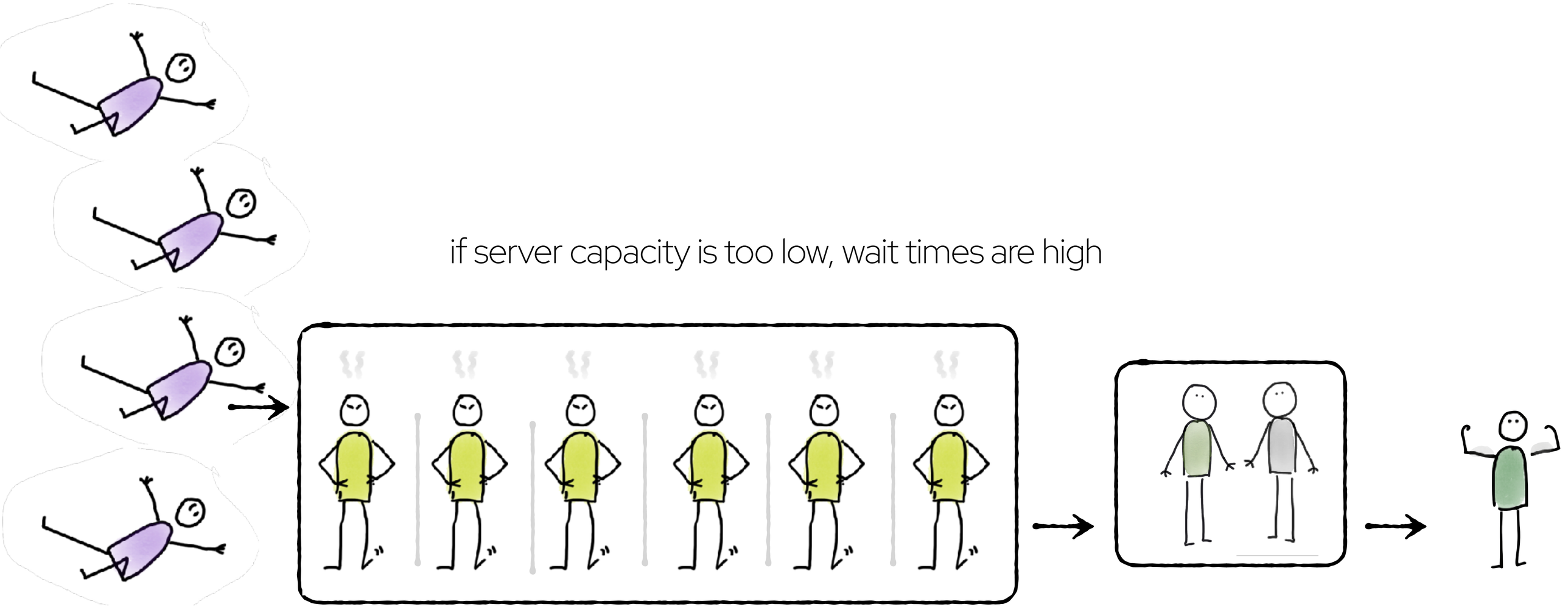
arrival process

queue

servers

completed work

if server capacity is too low, wait times are high



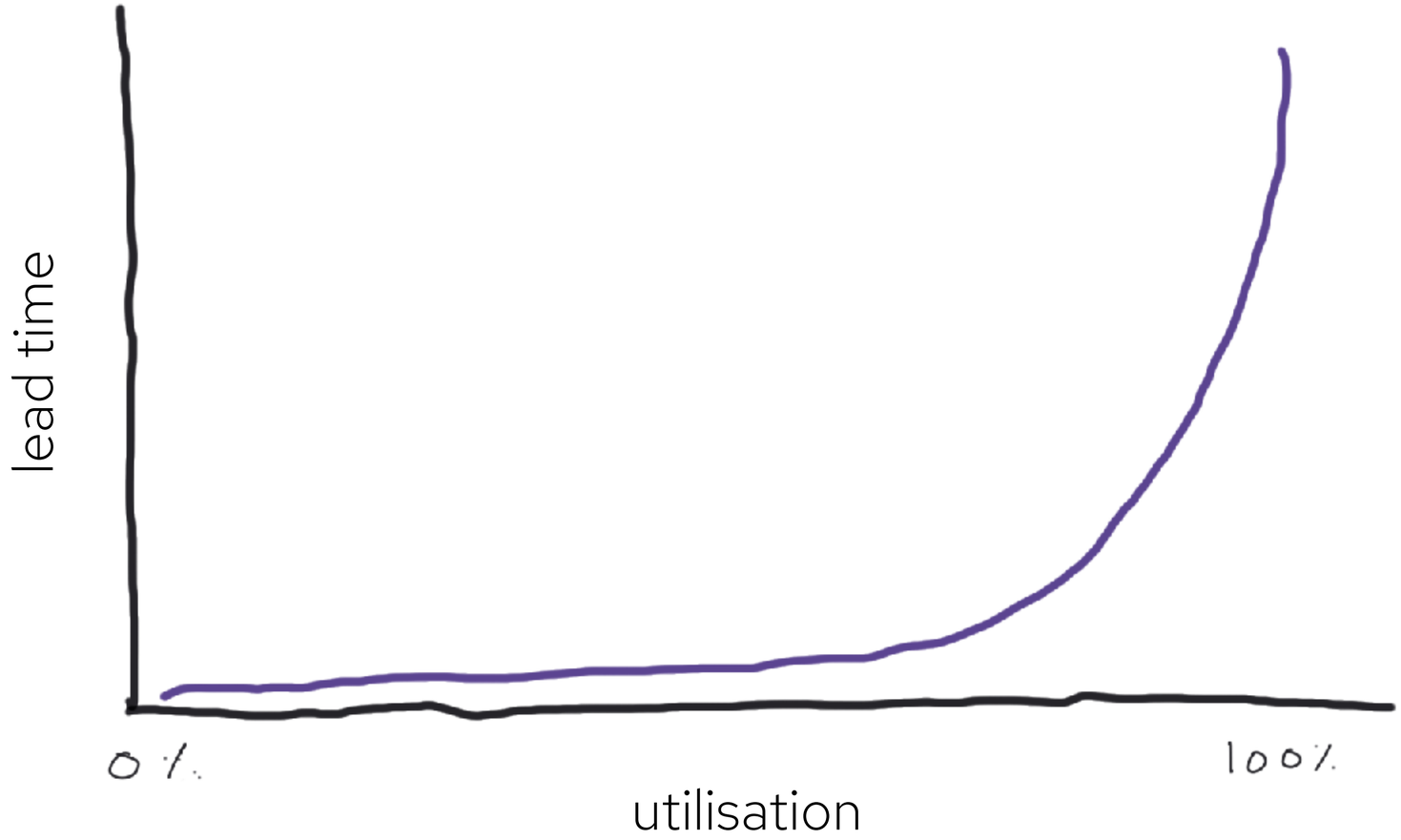
arrival process

queue

servers

completed work

http://brodzinski.com/2015/01/slack-time-value.html



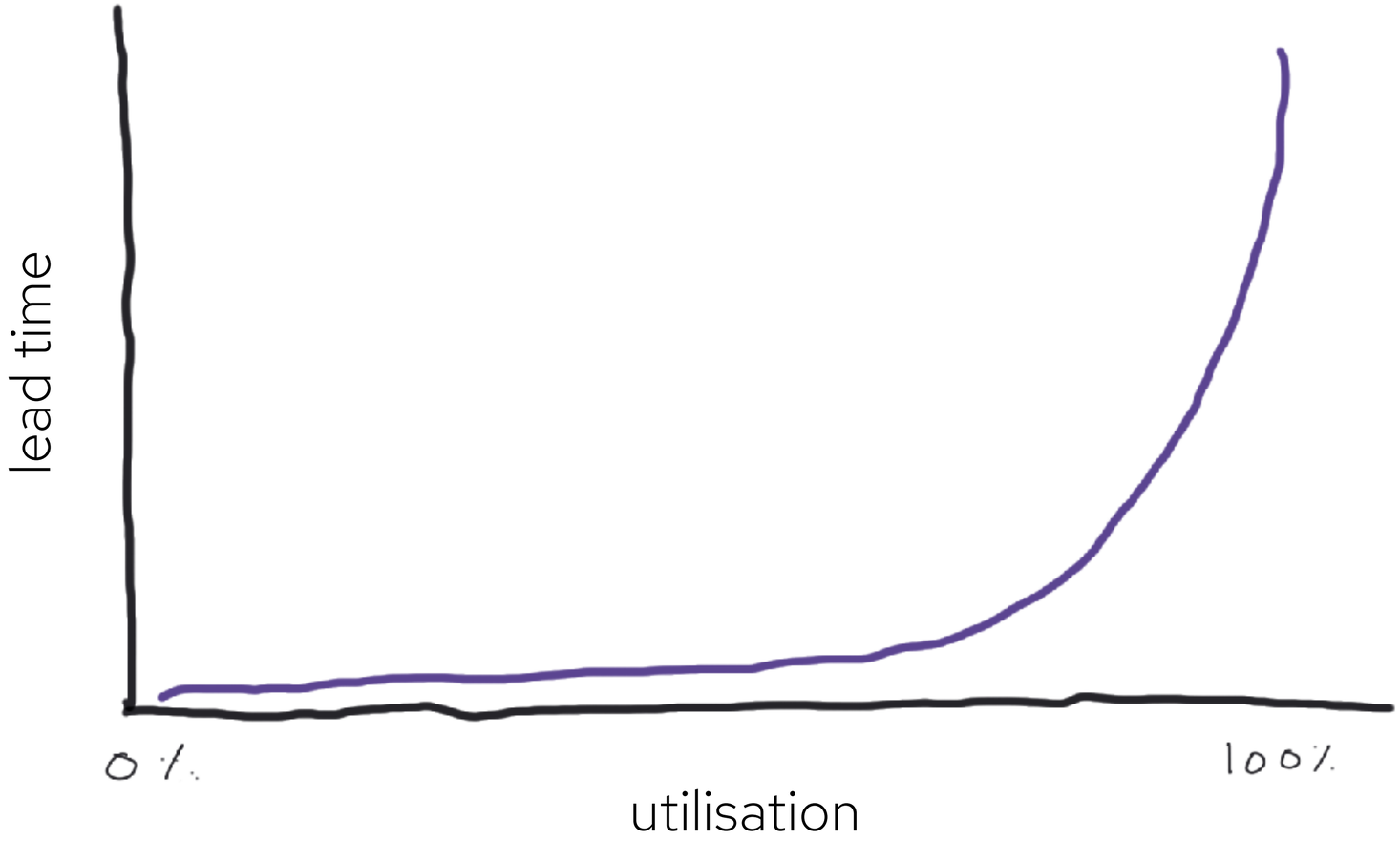


assuming Poisson
distribution of arrivals

http://brodzinski.com/2015/01/slack-time-value.html

80% utilisation → 90% utilisation:

wait times **double**

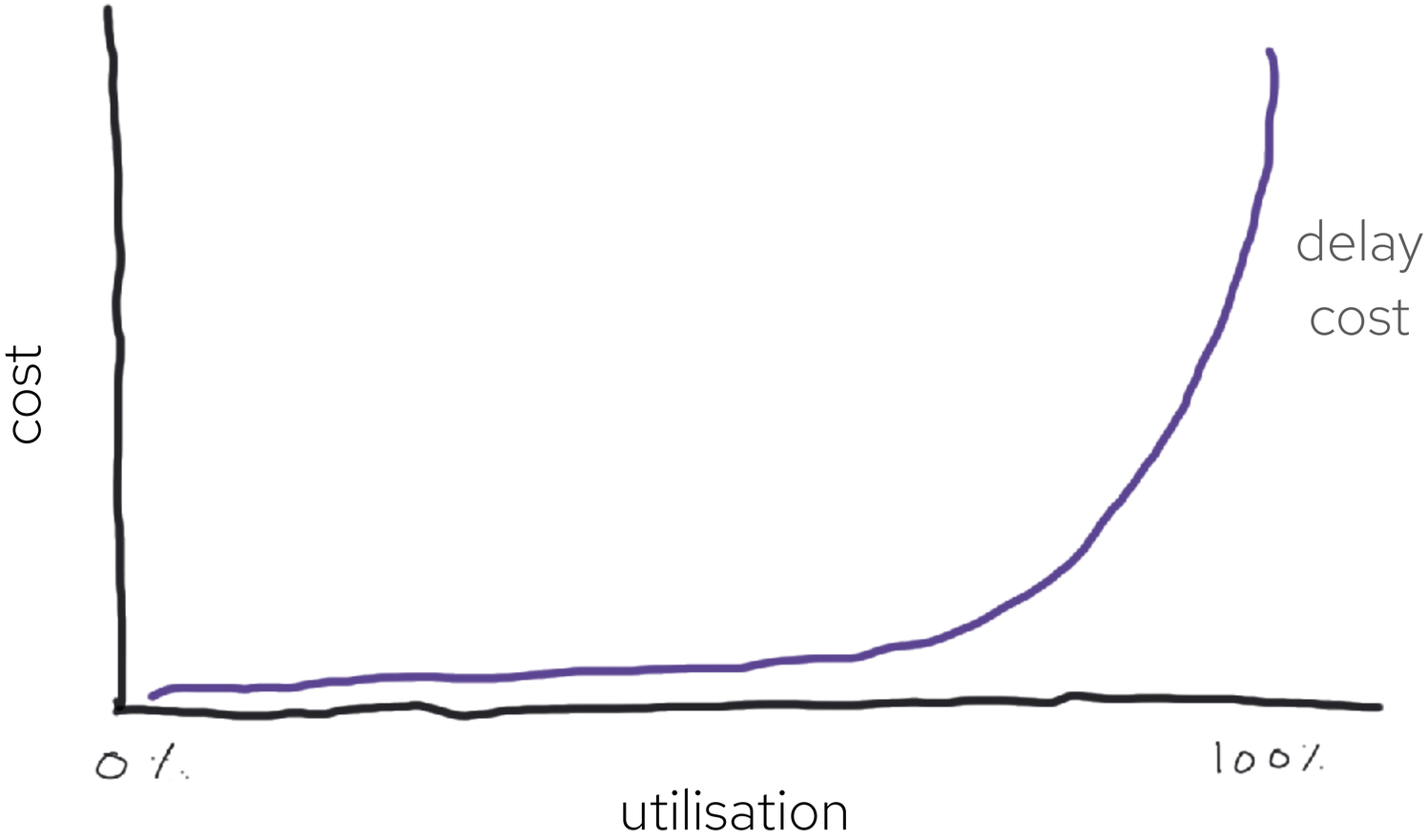




assuming Poisson distribution of arrivals

http://brodzinski.com/2015/01/slack-time-value.html

80% utilisation → 90% utilisation:
wait times **double**

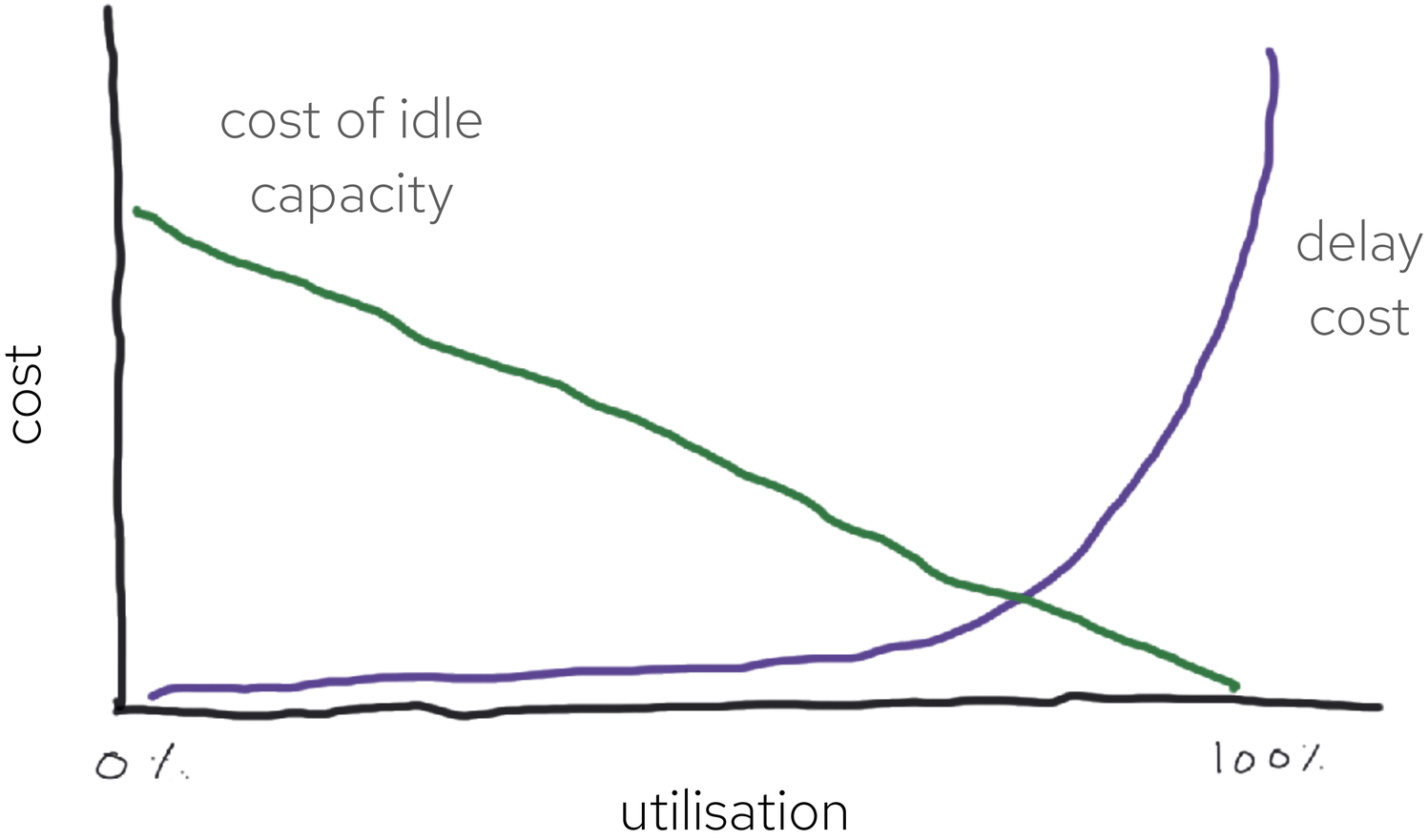




assuming Poisson
distribution of arrivals

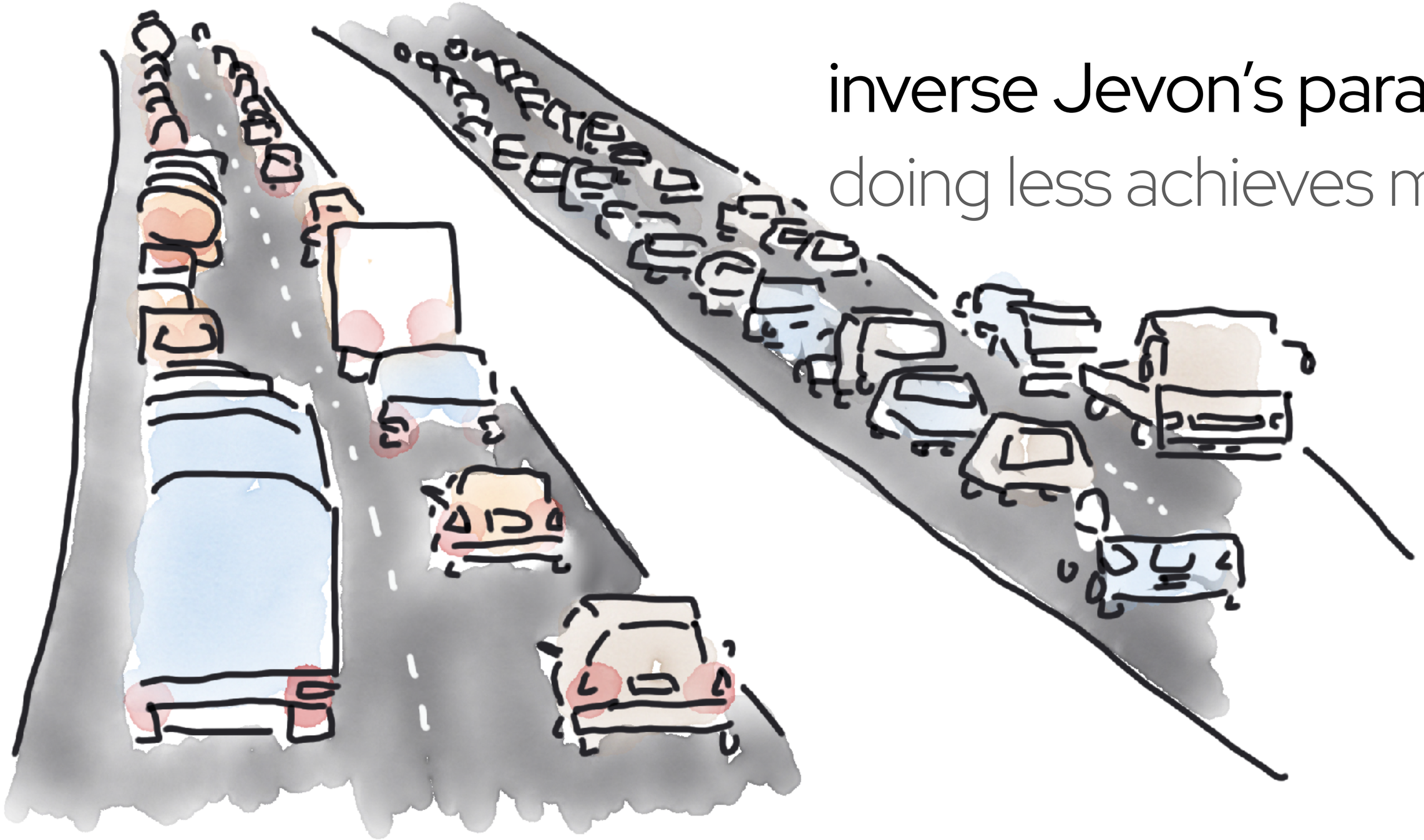
http://brodzinski.com/2015/01/slack-time-value.html

80% utilisation → 90% utilisation:
wait times **double**



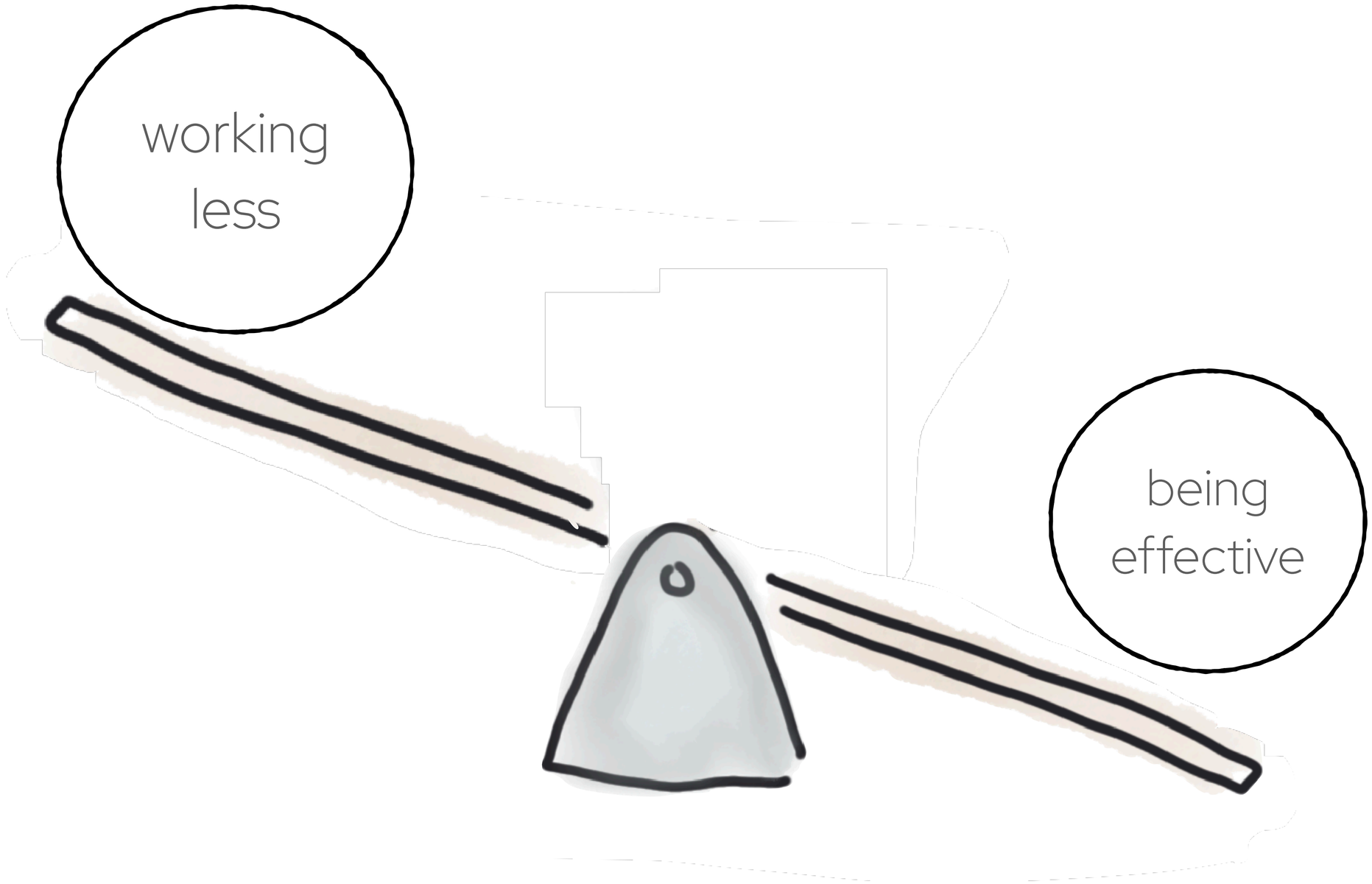


assuming Poisson distribution of arrivals



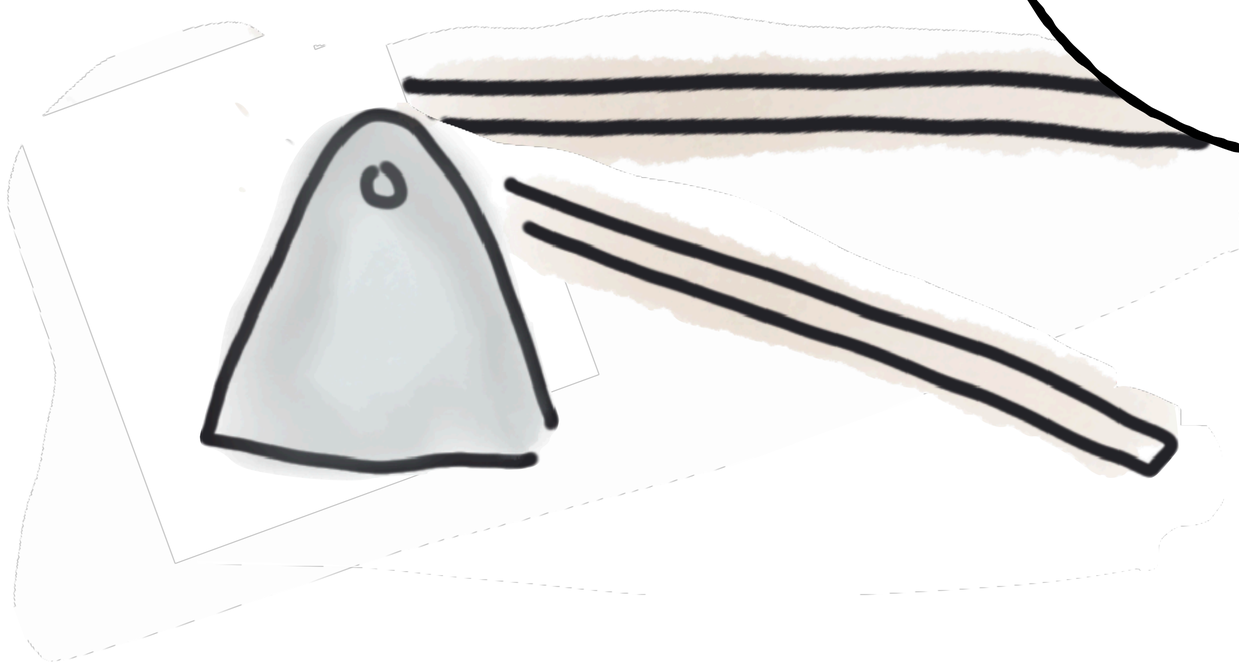
inverse Jevon's paradox

doing less achieves more

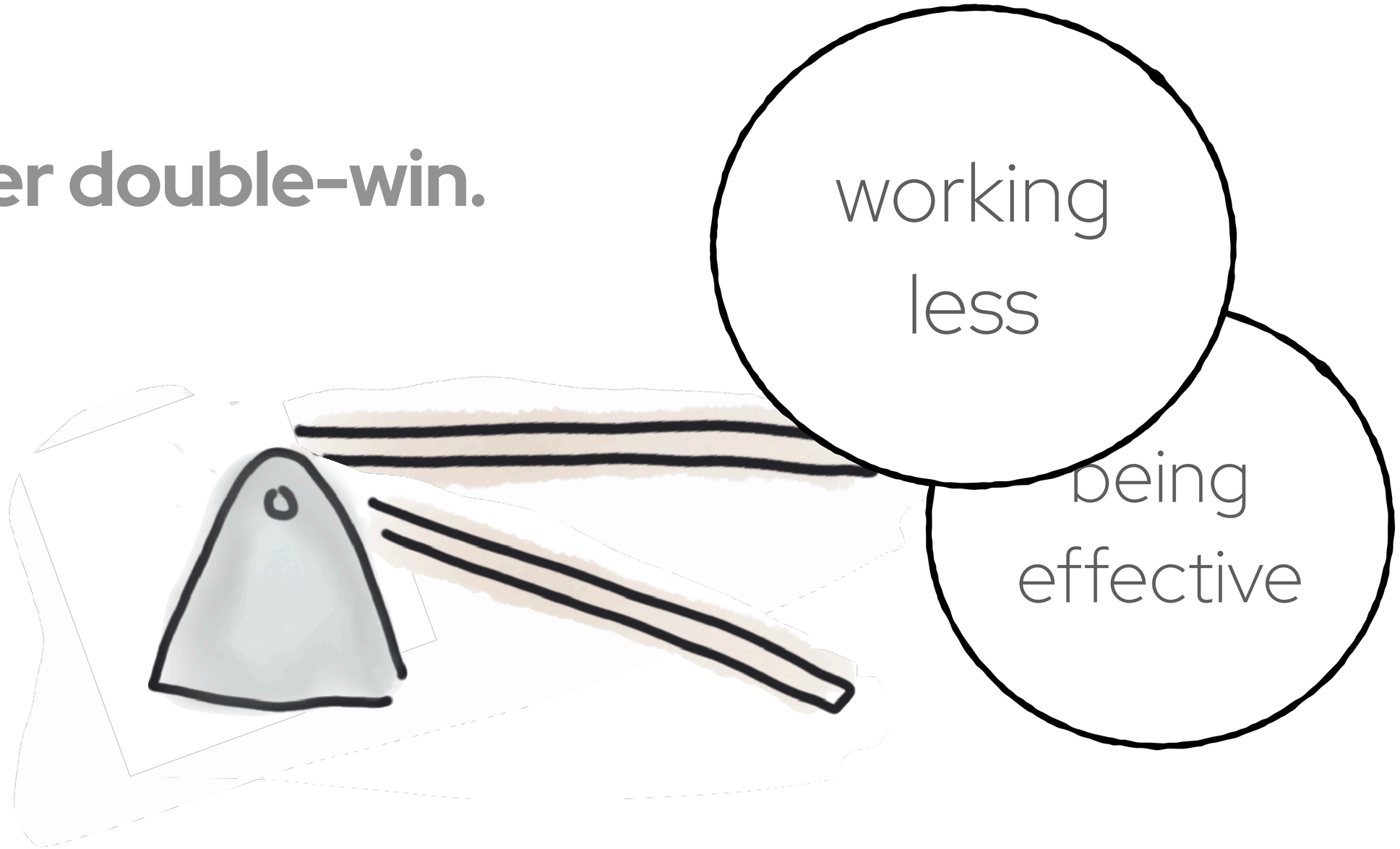

working
less
being
effective

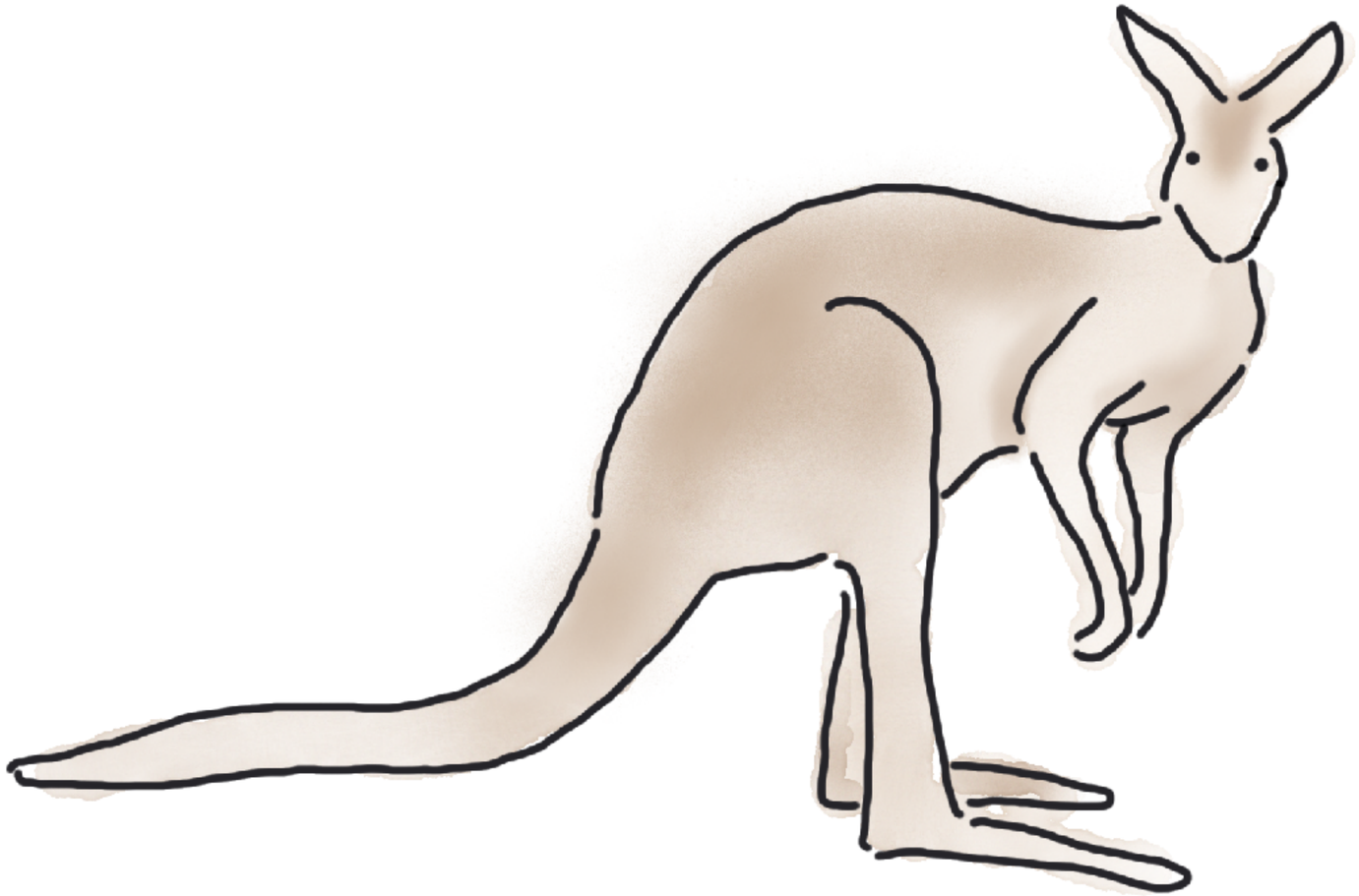
working less

being effective



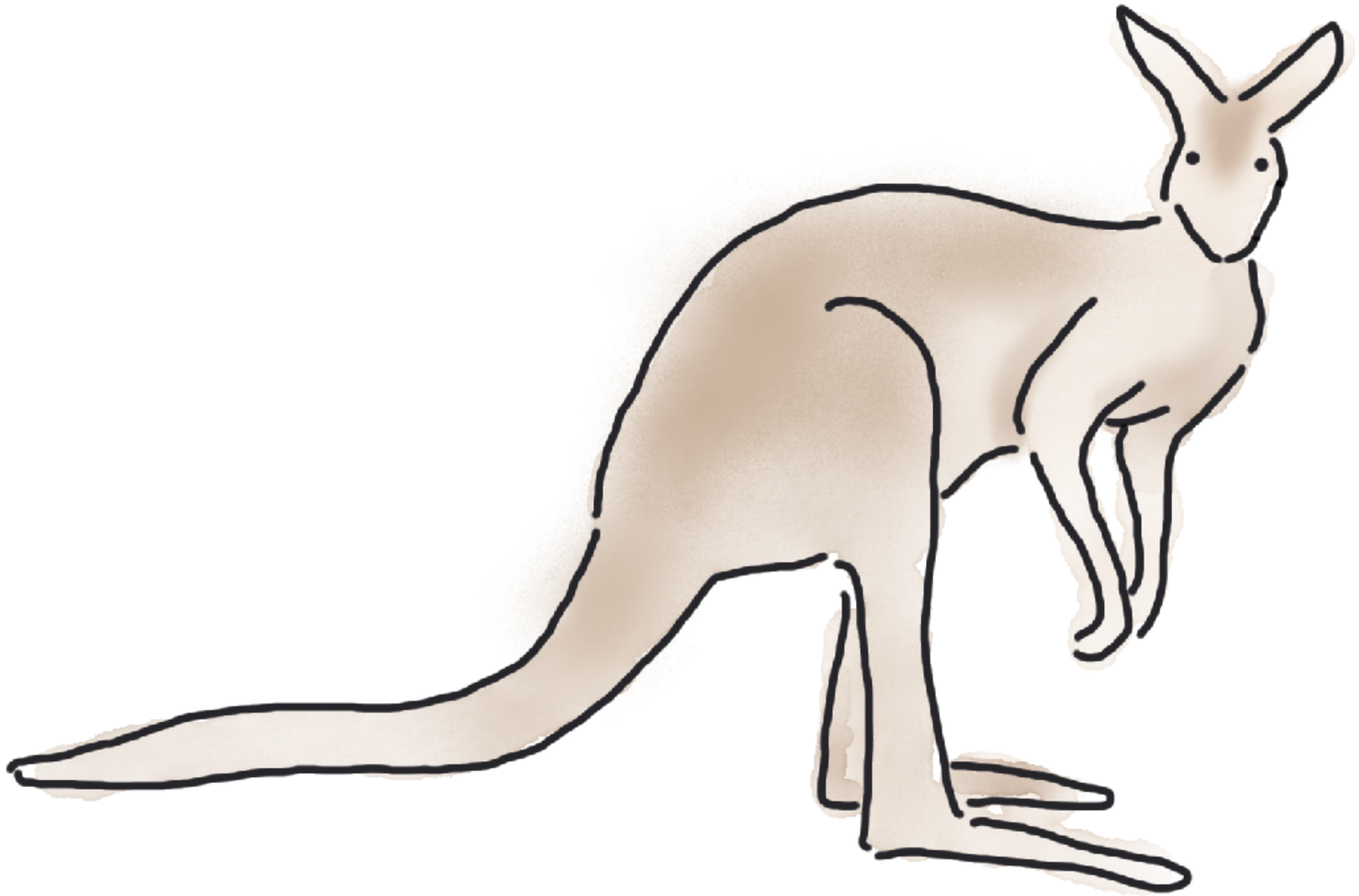
another double-win.





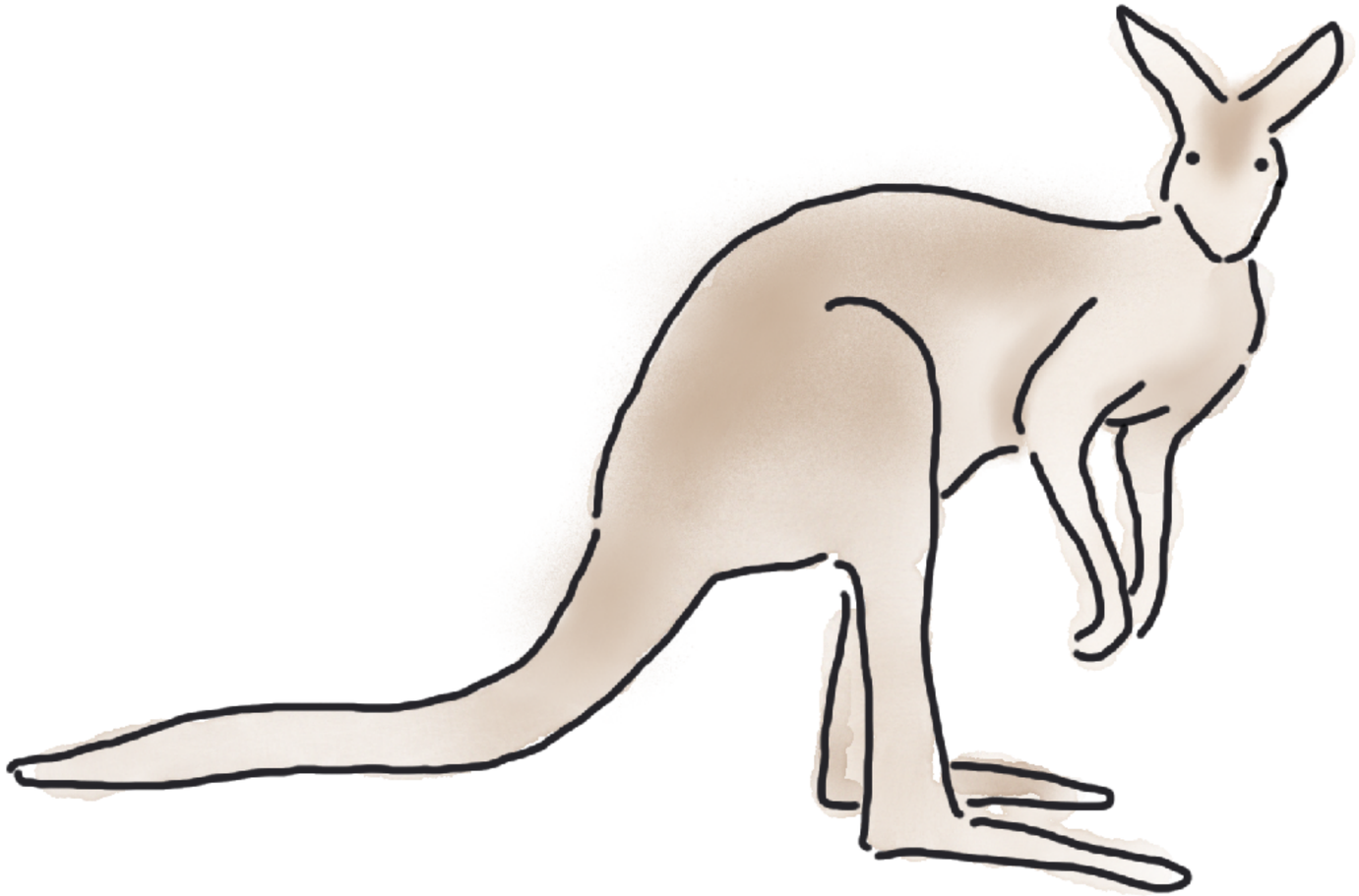
slack has an image problem

most efficient land animal



slack has an image problem

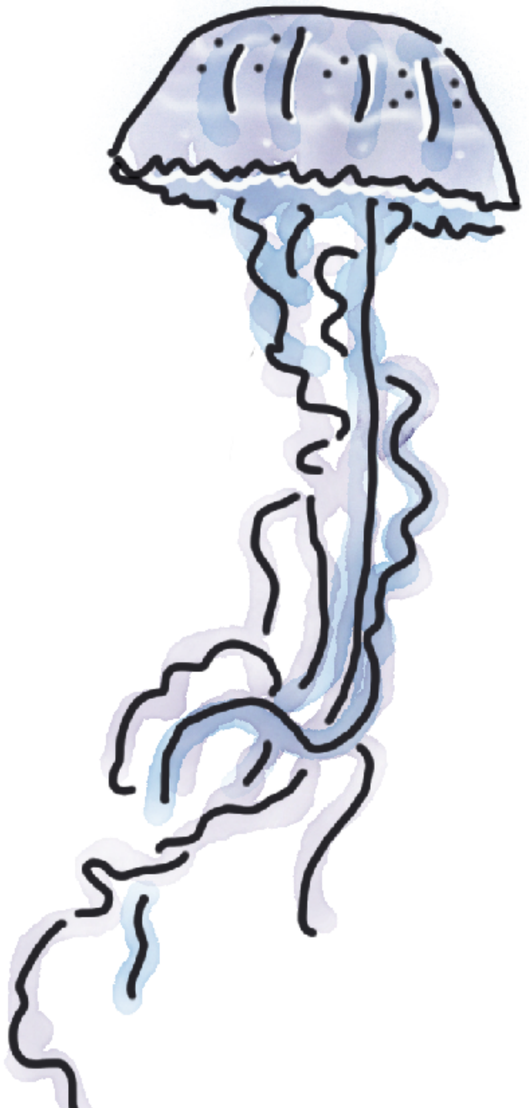
most efficient land animal
cool and bouncy



slack has an image problem

most efficient land animal
cool and bouncy

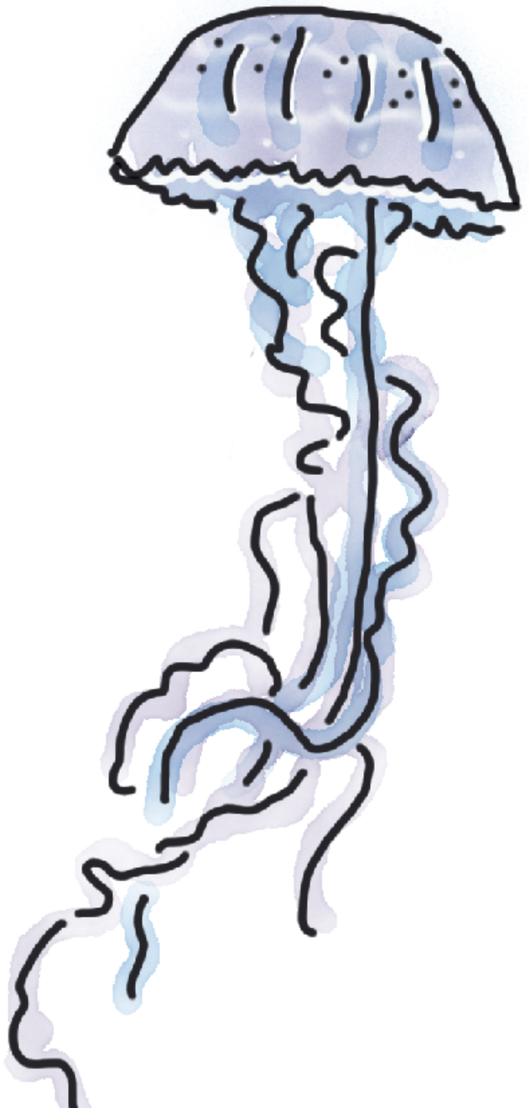
uncool and floppy



slack has an image problem

most efficient land animal
cool and bouncy

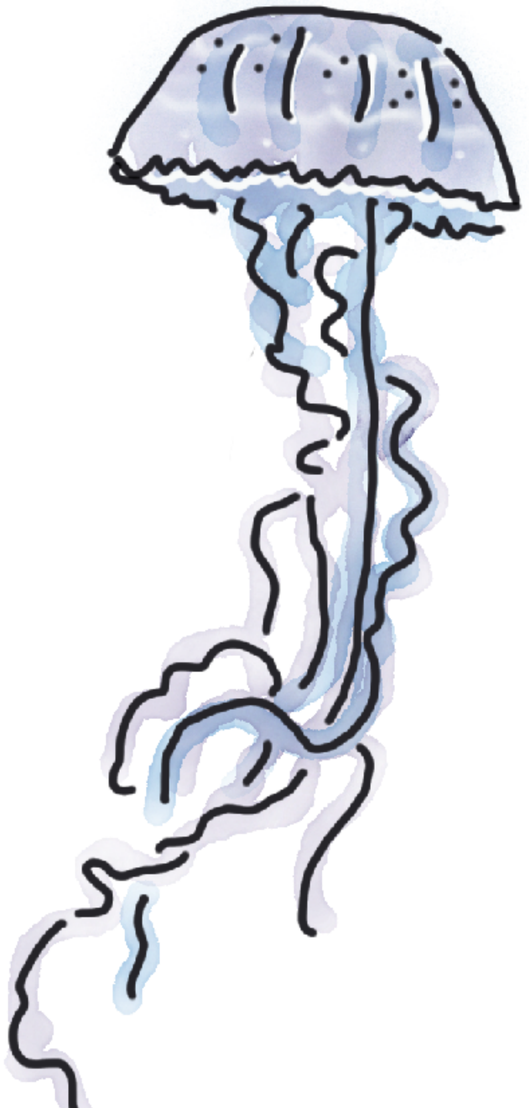
uncool and floppy
whole body is basically slack



slack has an image problem

most efficient land animal
cool and bouncy

uncool and floppy
whole body is basically slack
most efficient animal



slack has an image problem

hygge

hygge

“lazy cosy developers”



tl;dpa

(too long; didn't pay attention)

tl;dpa

(too long; didn't pay attention)

- don't accept **waste**; our world deserves better

tl;dpa

(too long; didn't pay attention)

- don't accept **waste**; our world deserves better
- work **less**; achieve more

tl;dpa

(too long; didn't pay attention)

- don't accept **waste**; our world deserves better
- work **less**; achieve more
- happiness is **not** waste

tl;dpa

(too long; didn't pay attention)

- don't accept **waste**; our world deserves better
- work **less**; achieve more
  - happiness is **not** waste
  - idleness is **not** waste

tl;dpa

(too long; didn't pay attention)

- don't accept **waste**; our world deserves better
- work **less**; achieve more
  - happiness is **not** waste
  - idleness is **not** waste
- look for double-wins **everywhere**

goto;

thank you!

Rate this session in the GOTO Guide

and claim your reward

slides



https://hollycummins.com/efficiency-goto/