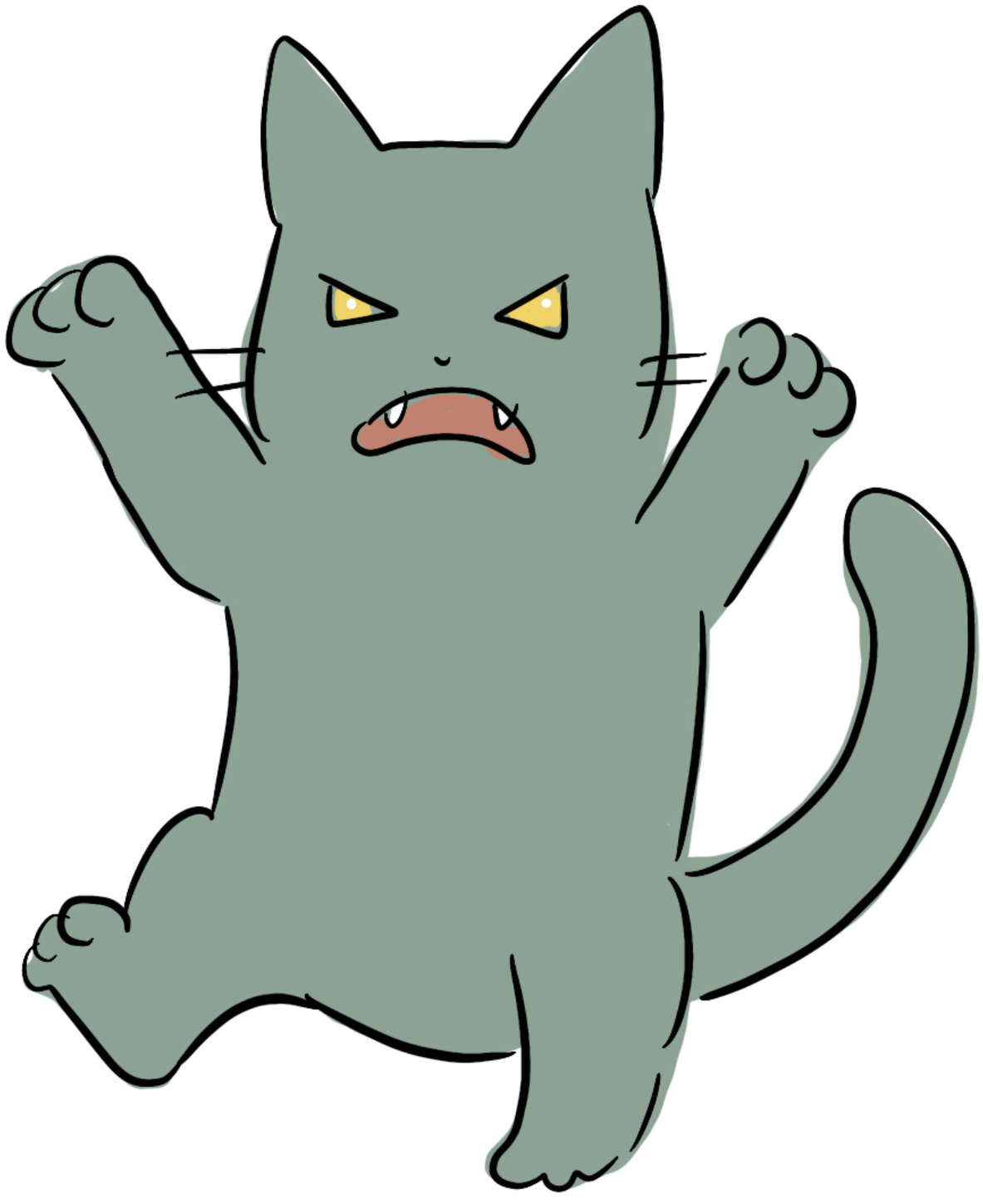
why are DISTRIBUTED SYSTEMS so hard?

@deniseyu21

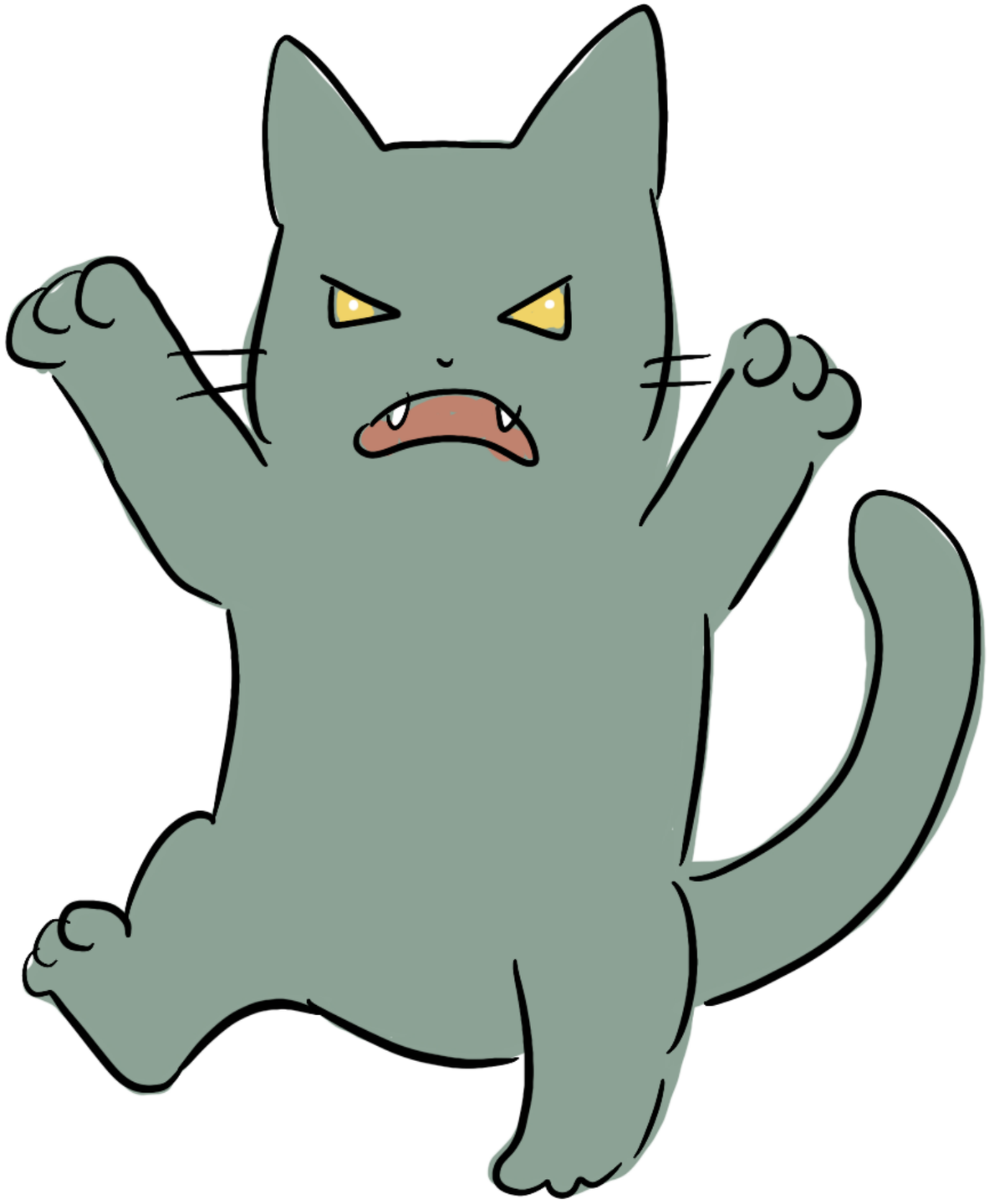
deniseyu.io

why even bother learning about distributed computing in 2020?

@deniseyu21



Services



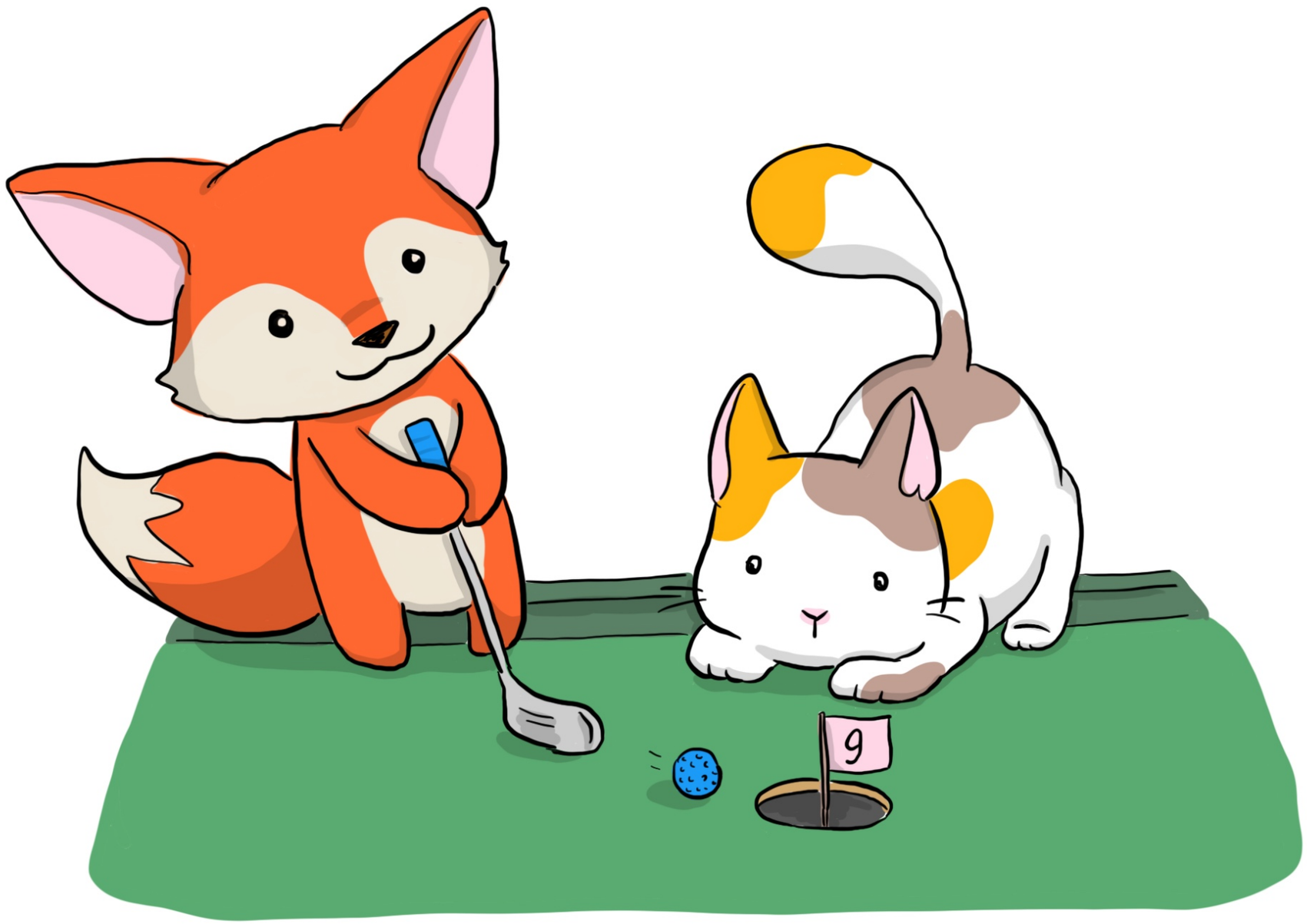
Services

Microservices

@deniseyu21

Why build smol services?

@deniseyu21



Sociotechnical goals

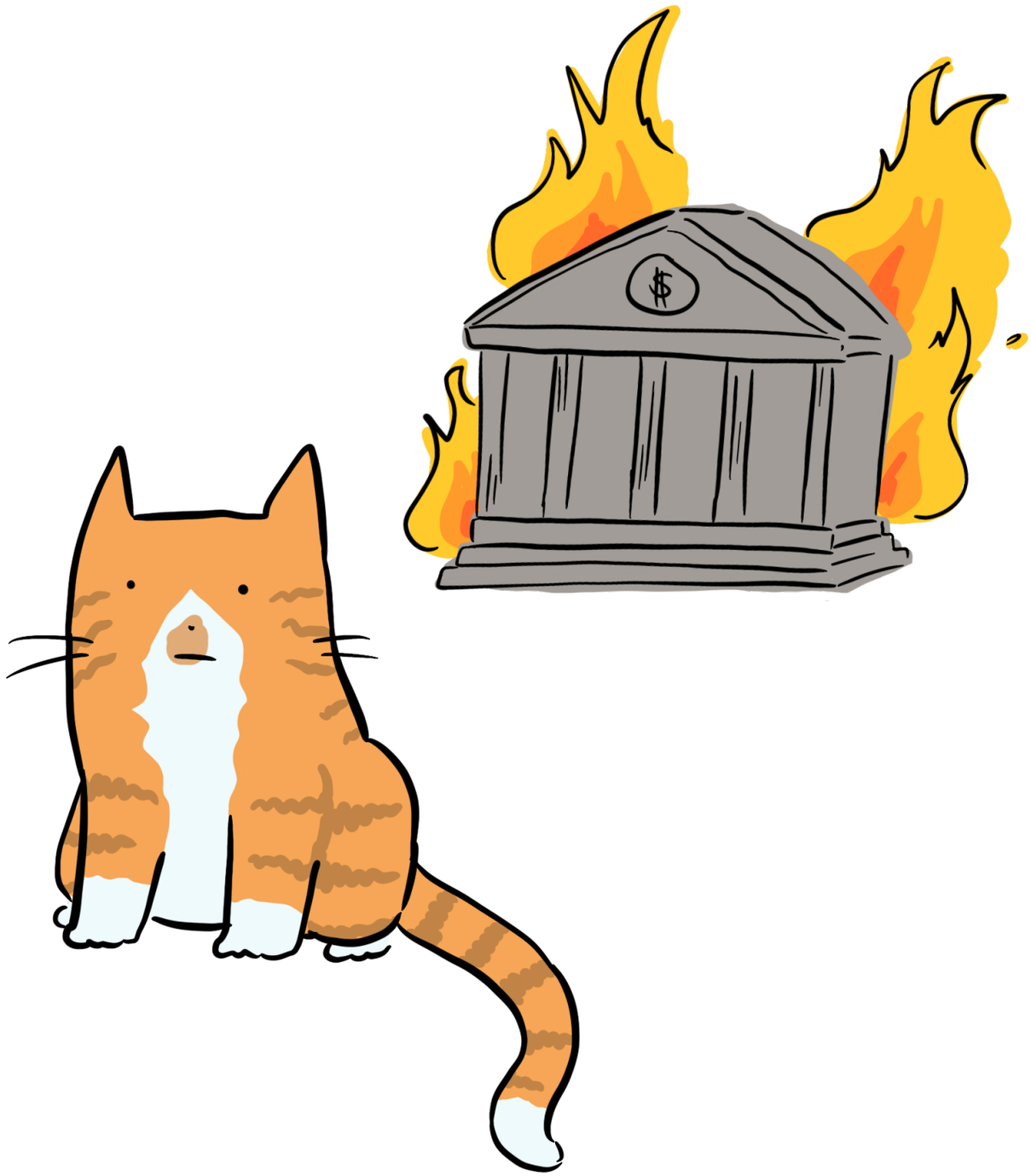
@deniseyu21

Engineering team boundaries => domain boundaries

@deniseyu21

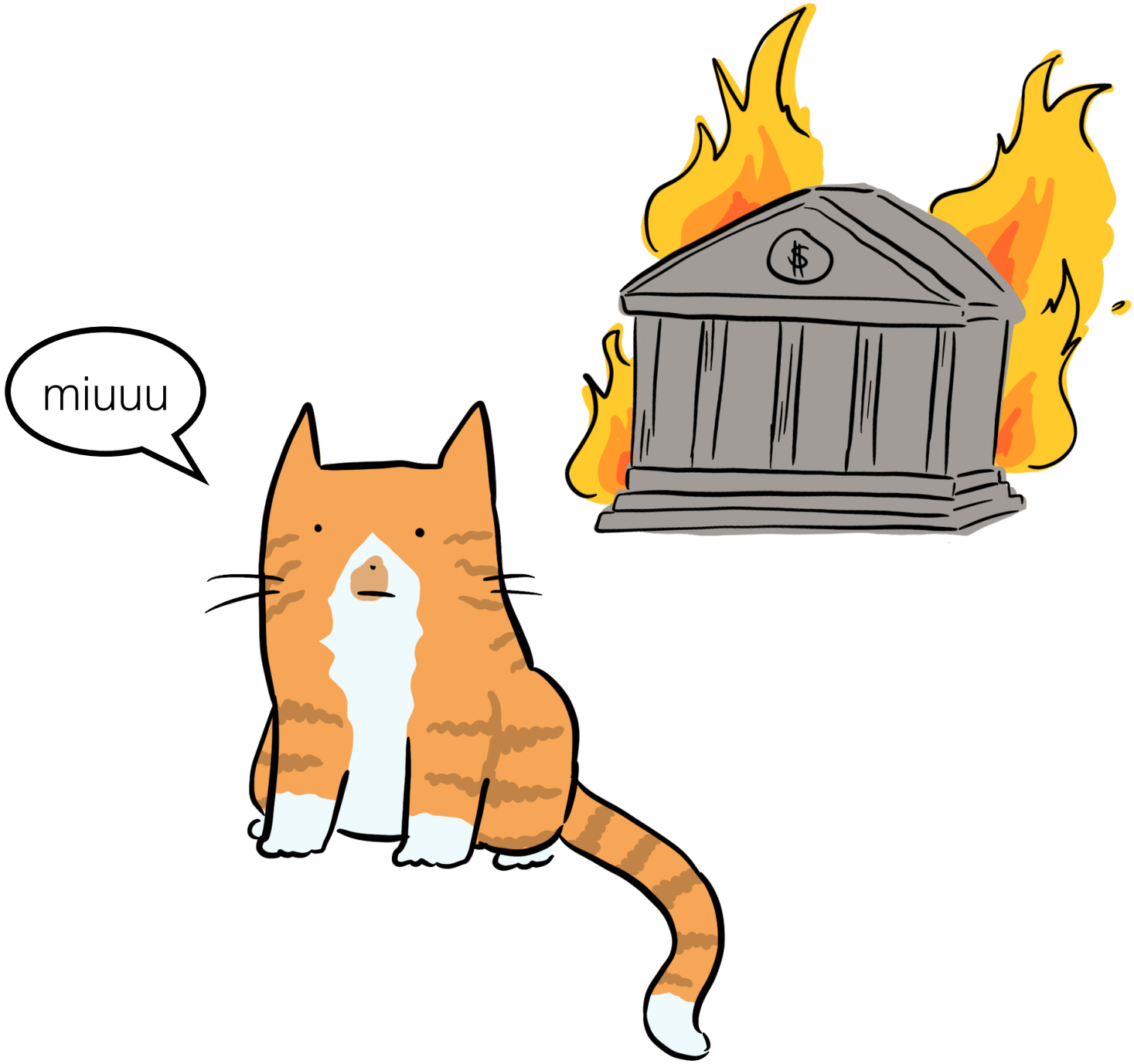
Teams can release & deploy independently

@deniseyu21



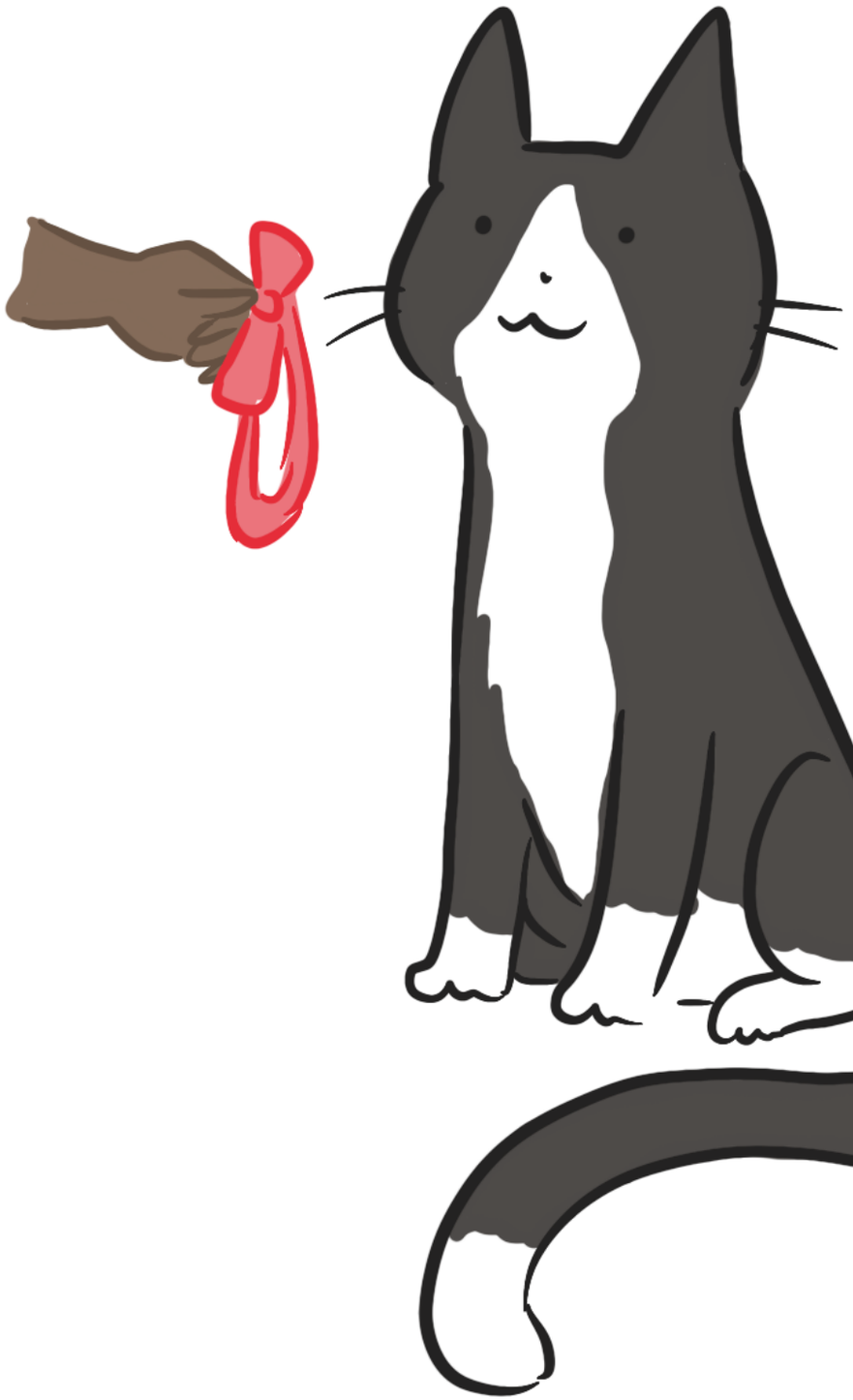
We improve the resiliency of our critical systems…

@deniseyu21

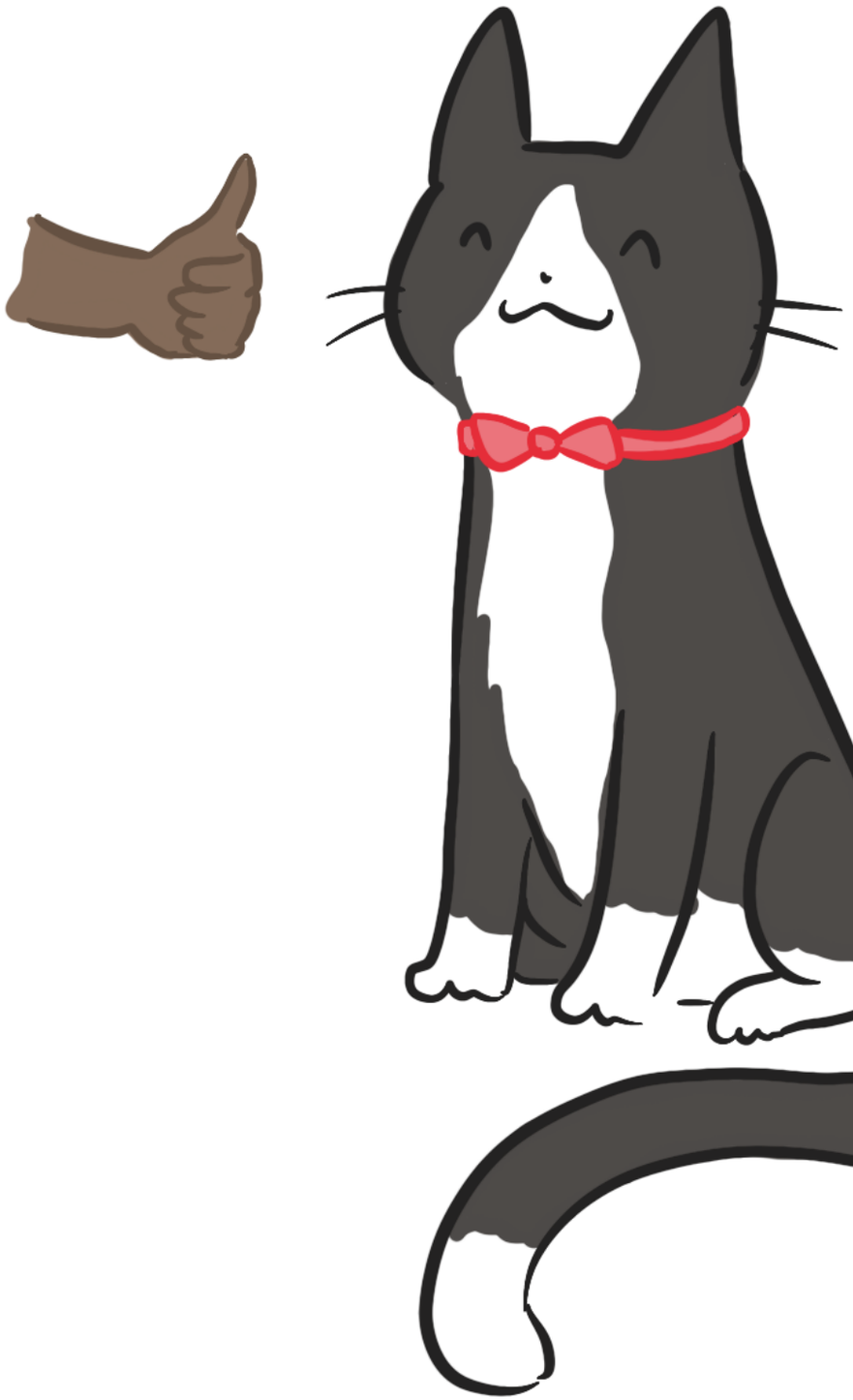


…against *cat*astrophic, show-stopping events

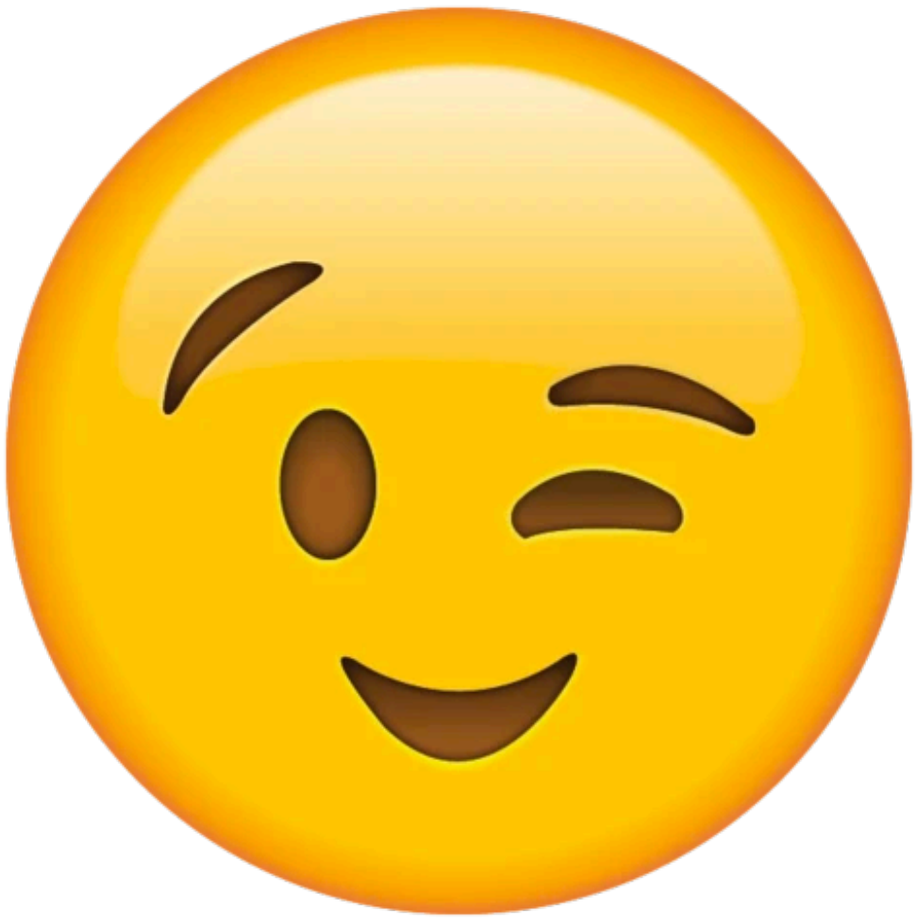
@deniseyu21

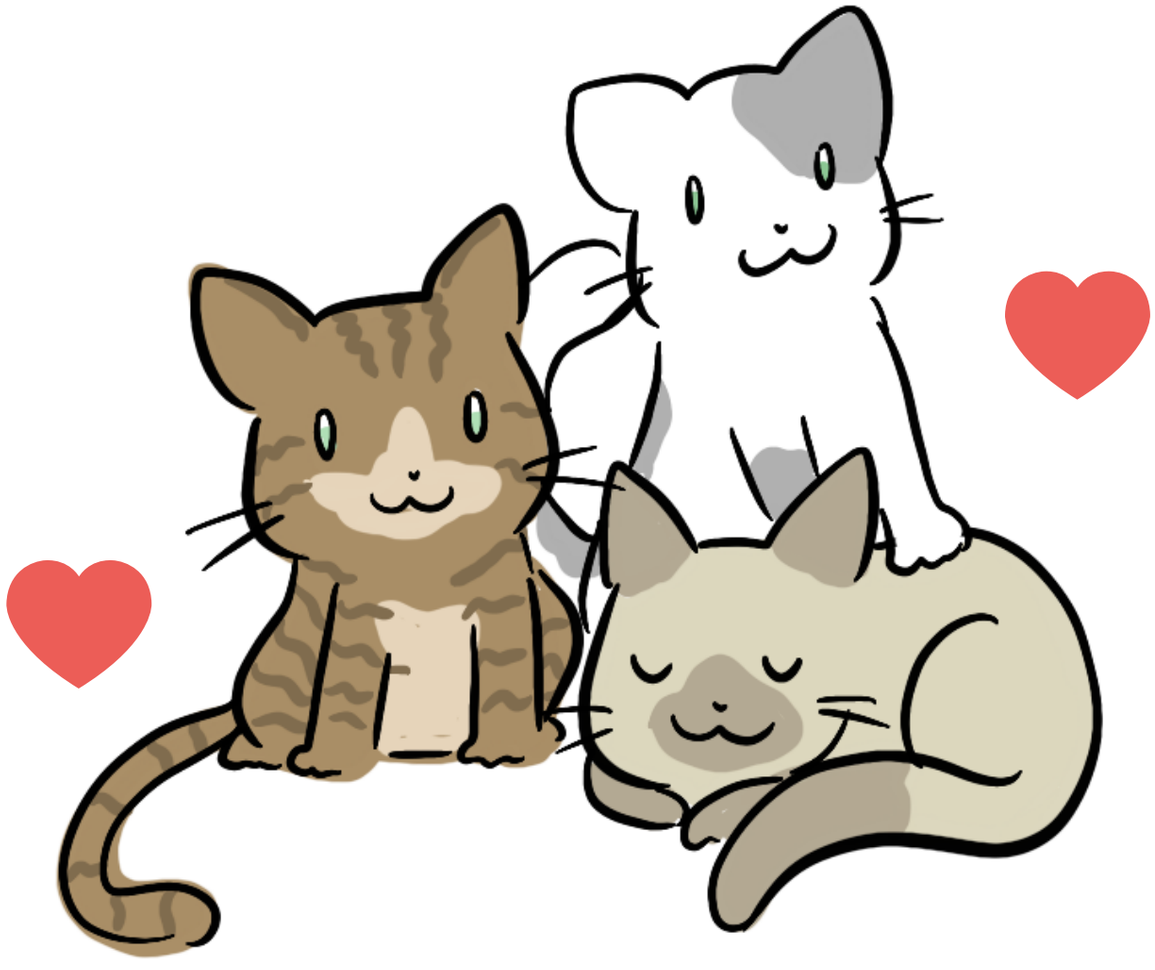


After all, we can change one microservice,



without disturbing or changing any others.





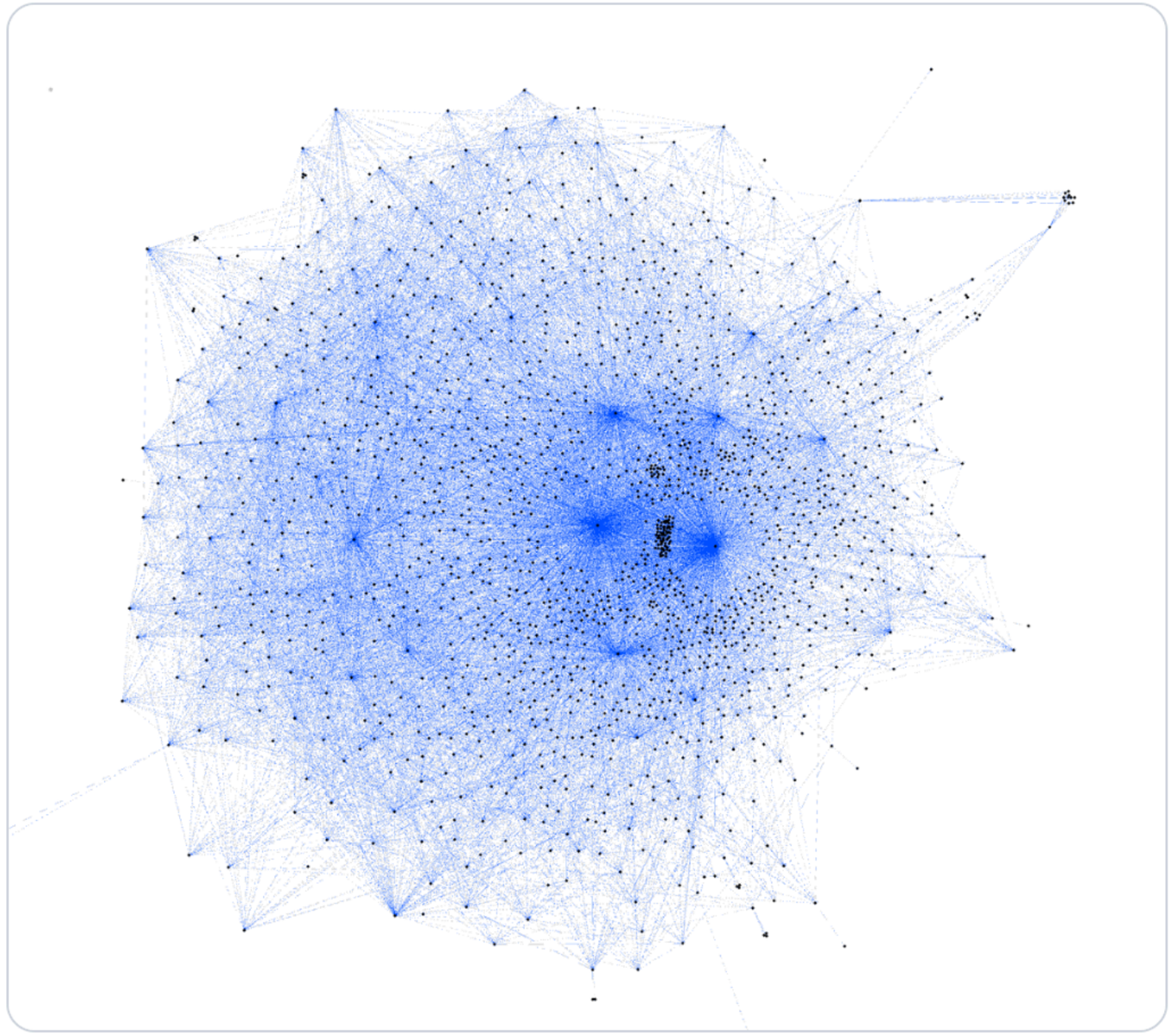
There are more reasons! But enough of that now…

@deniseyu21

Everyone who runs microservices is
at the helm of a large distributed system

Jack Kleeman
@JackKleeman

1500 microservices at @monzo; every line is an enforced network rule allowing traffic

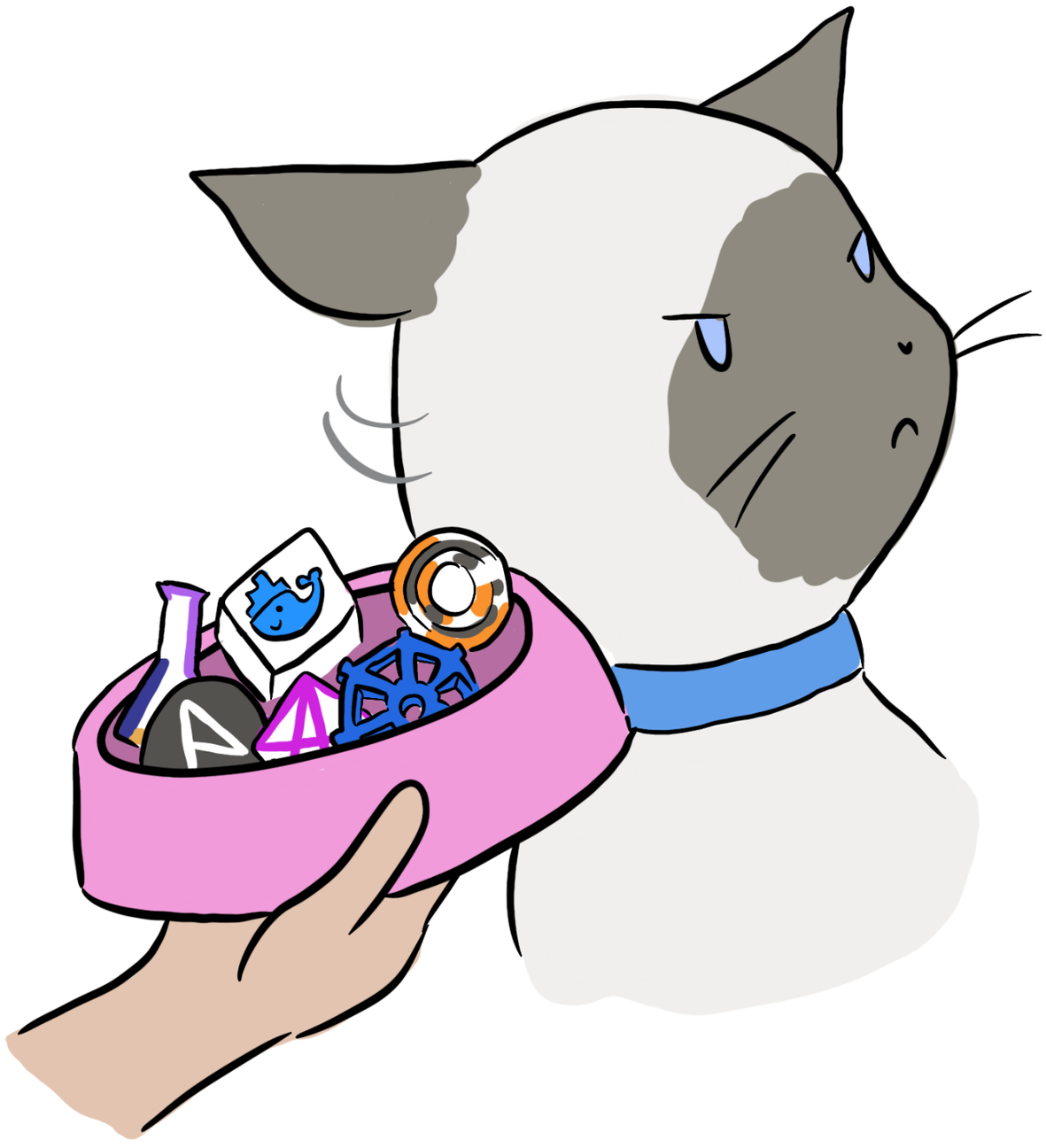


3:47 PM · Nov 1, 2019 · Twitter Web App

642 Retweets **2.7K** Likes

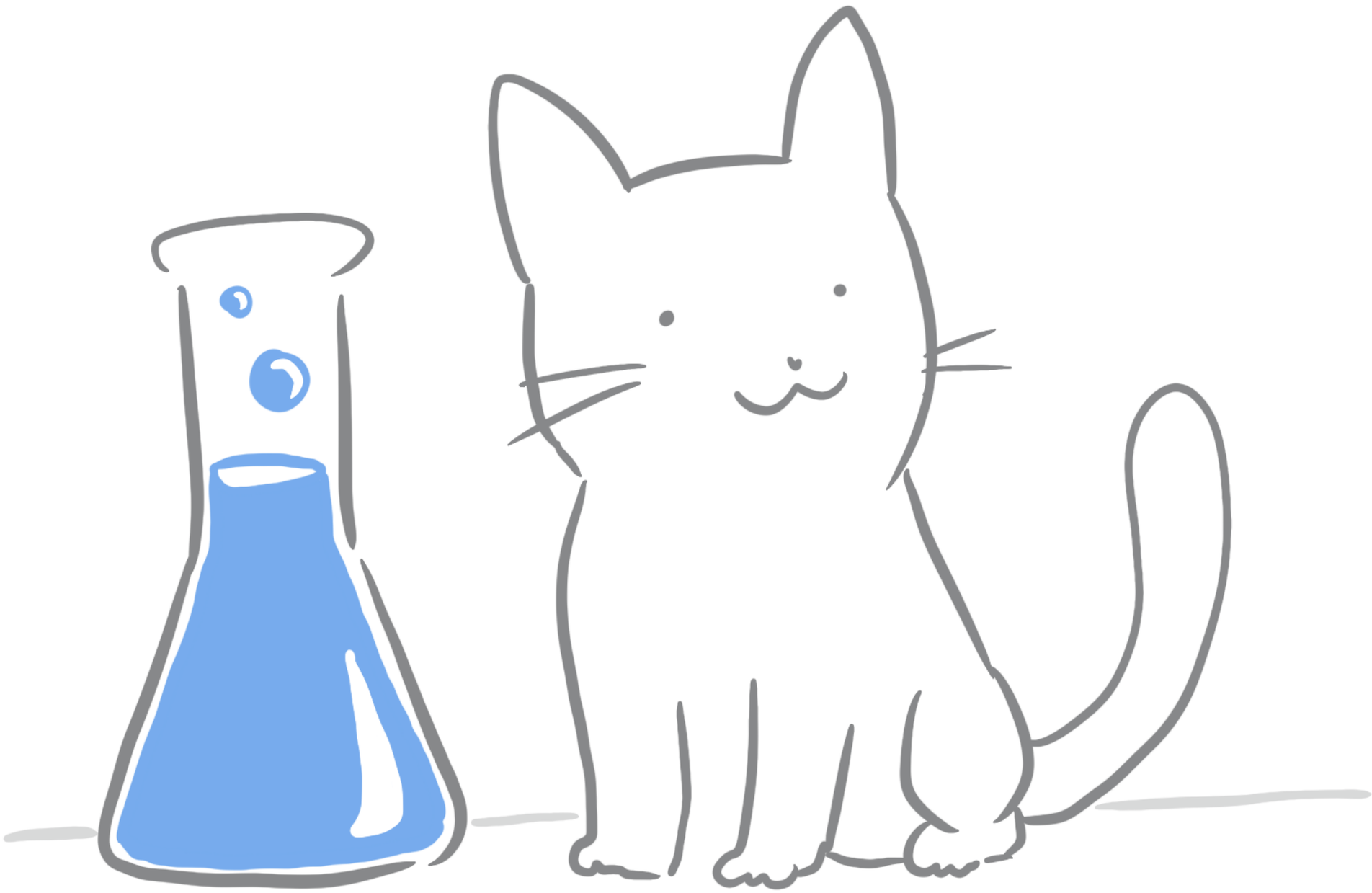
Specific technologies will come and go

@deniseyu21



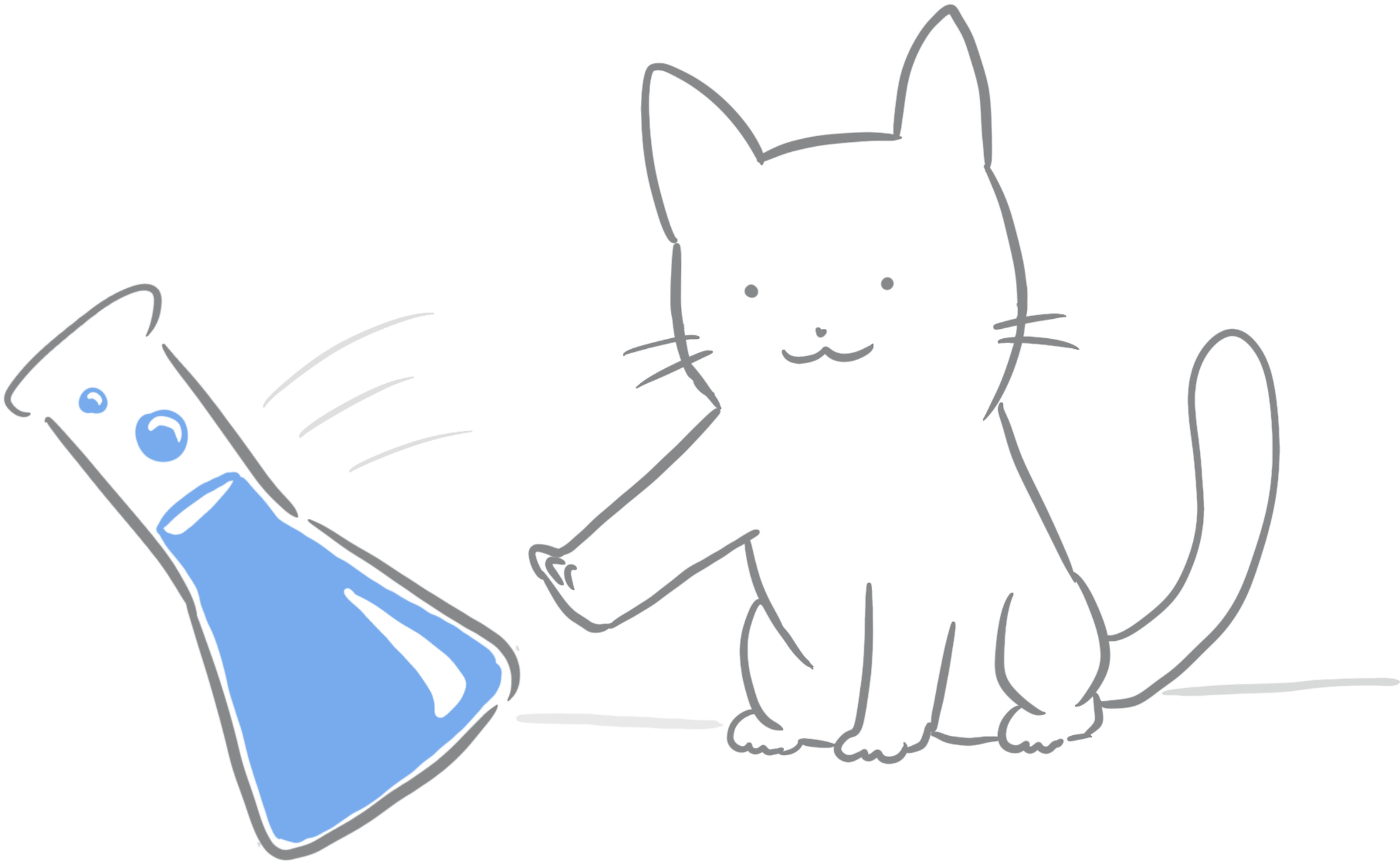
and trends are fickle

@deniseyu21



but fundamentals haven’t changed all that much

@deniseyu21



and knowledge is the best way to future-proof systems

@deniseyu21

It’s also really fun to learn about distributed computing!

@deniseyu21


hello
#QCon!
@deniseyu21
deniseyu.io/qcon

Senior Engineer @ GitHub Community & Safety

hello #QCon!

@deniseyu21

deniseyu.io/qcon

Senior Engineer @ GitHub Community & Safety

TORONTO-BASED

hello #QCon!

@deniseyu21

deniseyu.io/qcon

Senior Engineer @ GitHub Community & Safety

TORONTO-BASED

hello #QCon!

@deniseyu21

deniseyu.io/qcon

Tech-doodling

deniseyu.io/art

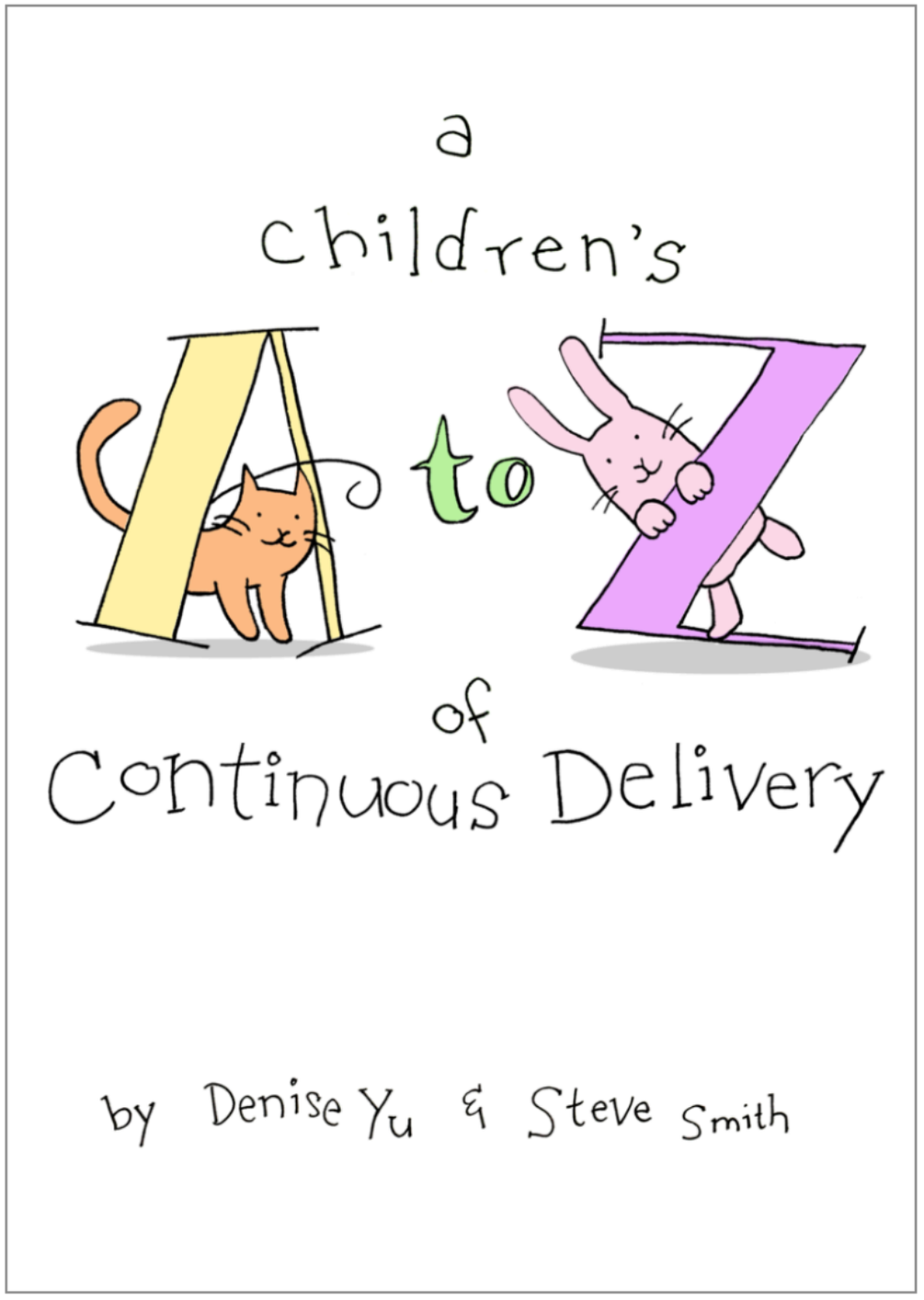
A Children's A to Z of Continuous Delivery

Denise Yu and Steve Smith

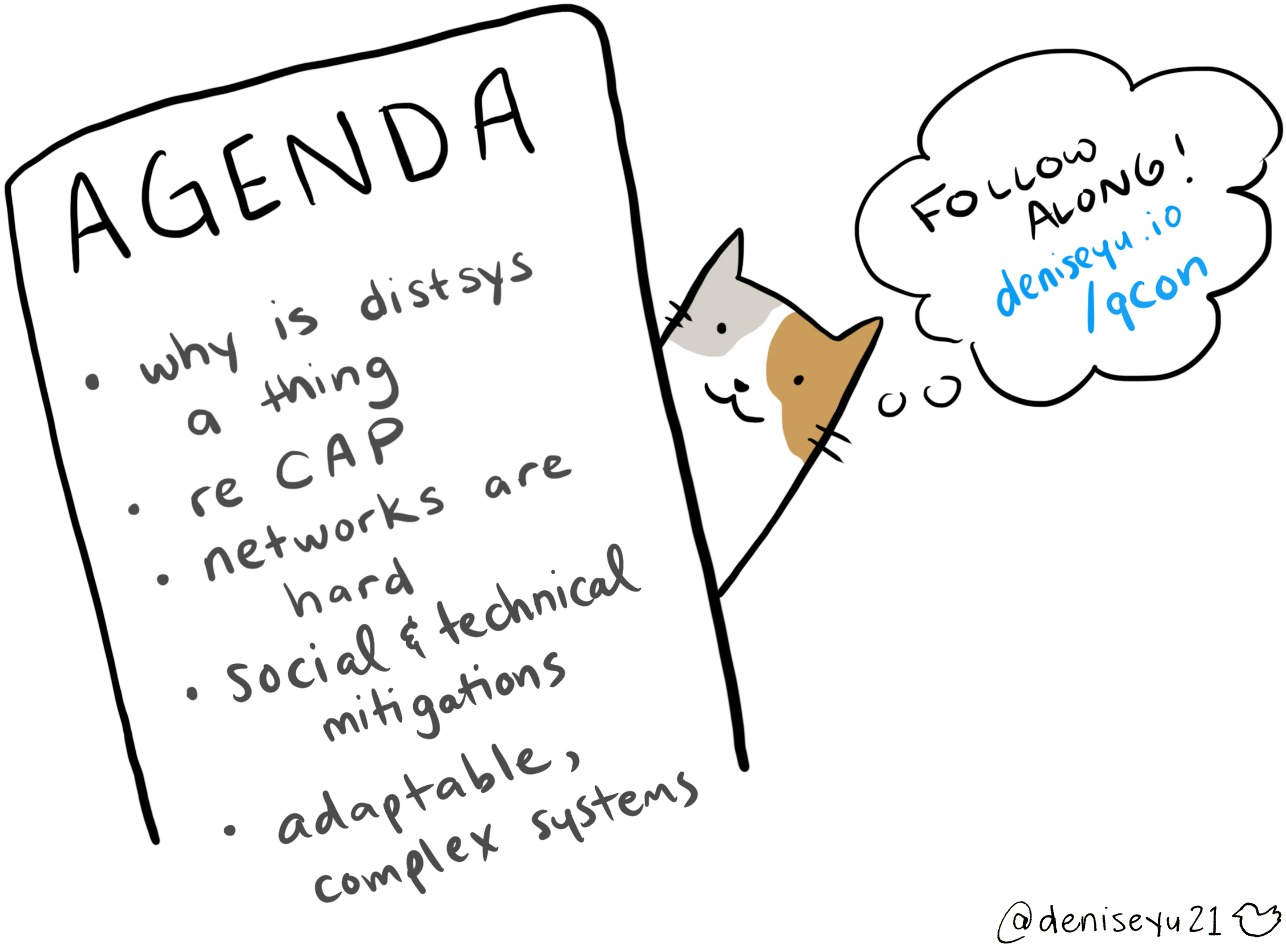
A friendly introduction to key concepts in Continuous Delivery, for all ages

Read Free Sample

Table Of Contents



leanpub.com/achildrensatozofcontinuousdelivery


AGENDA
• why is distsys a thing
• re CAP
• networks are hard
• social & technical mitigations
• adaptable, complex systems
FOLLOW ALONG!
denisey u.io
/qcon
@denisey u21

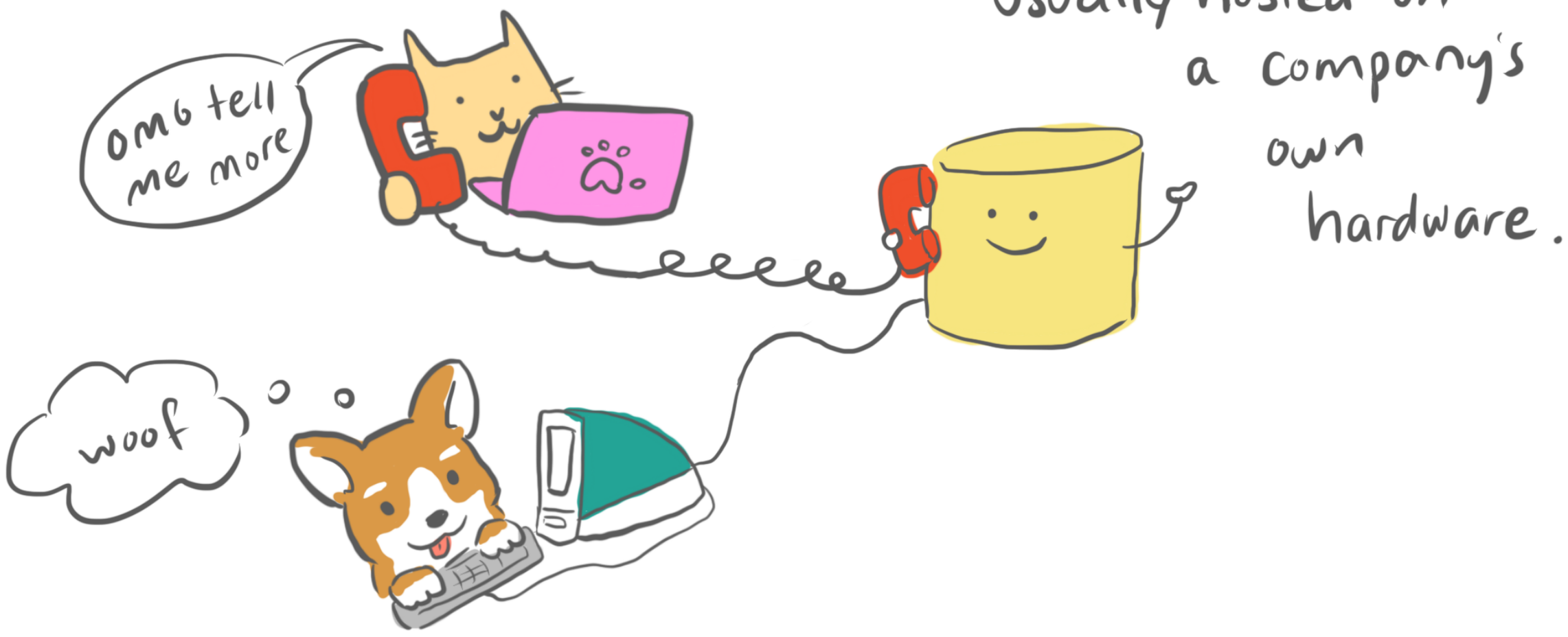
A long time ago,

in a datacenter not too far away...

A long time ago,

in a datacenter not too far away...

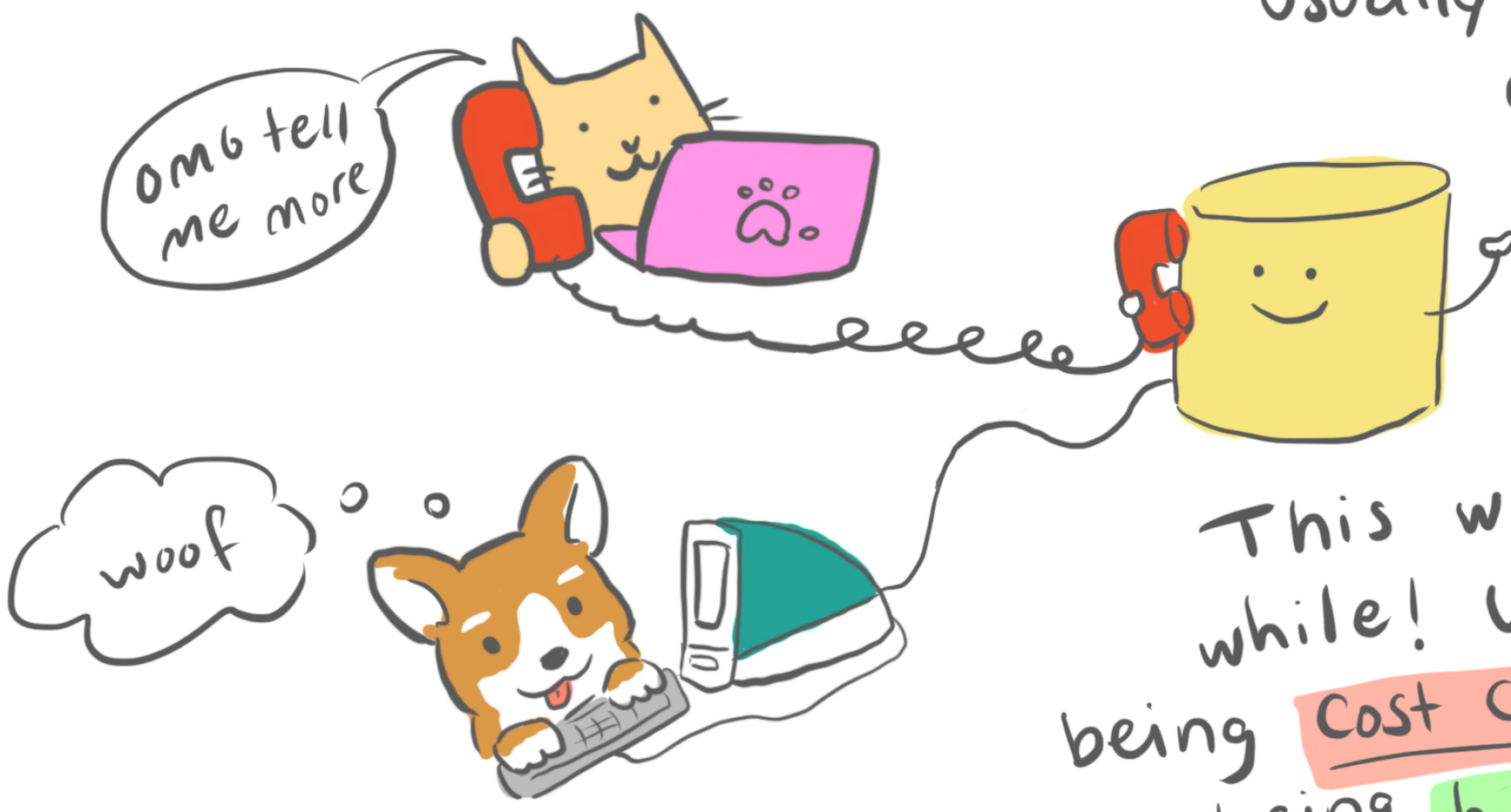
All business applications talked to one database,
usually hosted on
a company's
own
hardware.



A long time ago,

in a datacenter not too far away...

All business applications talked to one database, usually hosted on a company's own hardware.



This worked for a while! Until IT stopped being Cost centers & started being business enablers.

@deniseyu21

Data storage & retrieval needs evolved as software became business differentiators.

Data storage & retrieval needs evolved as software became business differentiators.

Business analysis
& data warehouses

Data storage & retrieval needs evolved as software became business differentiators.

Business analysis & data warehouses

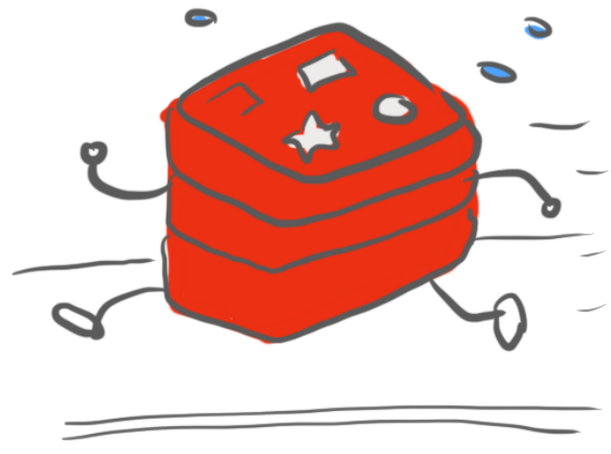
ML & Natural Language processing

Data storage & retrieval needs evolved as software became business differentiators.

Business analysis & data warehouses

ML & Natural Language processing

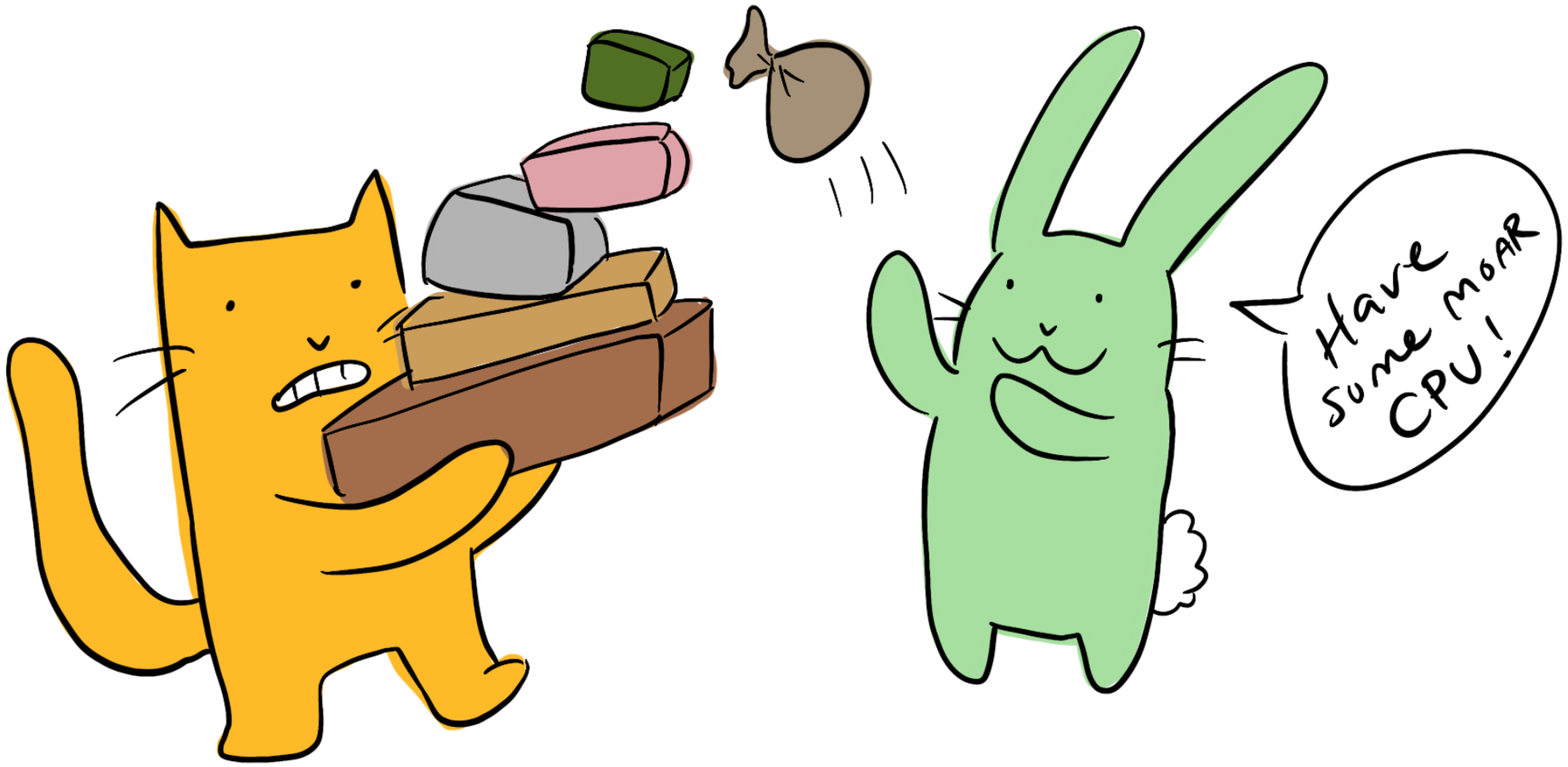
Faster, bigger queries!



@deniseyu21

So we scaled vertically...

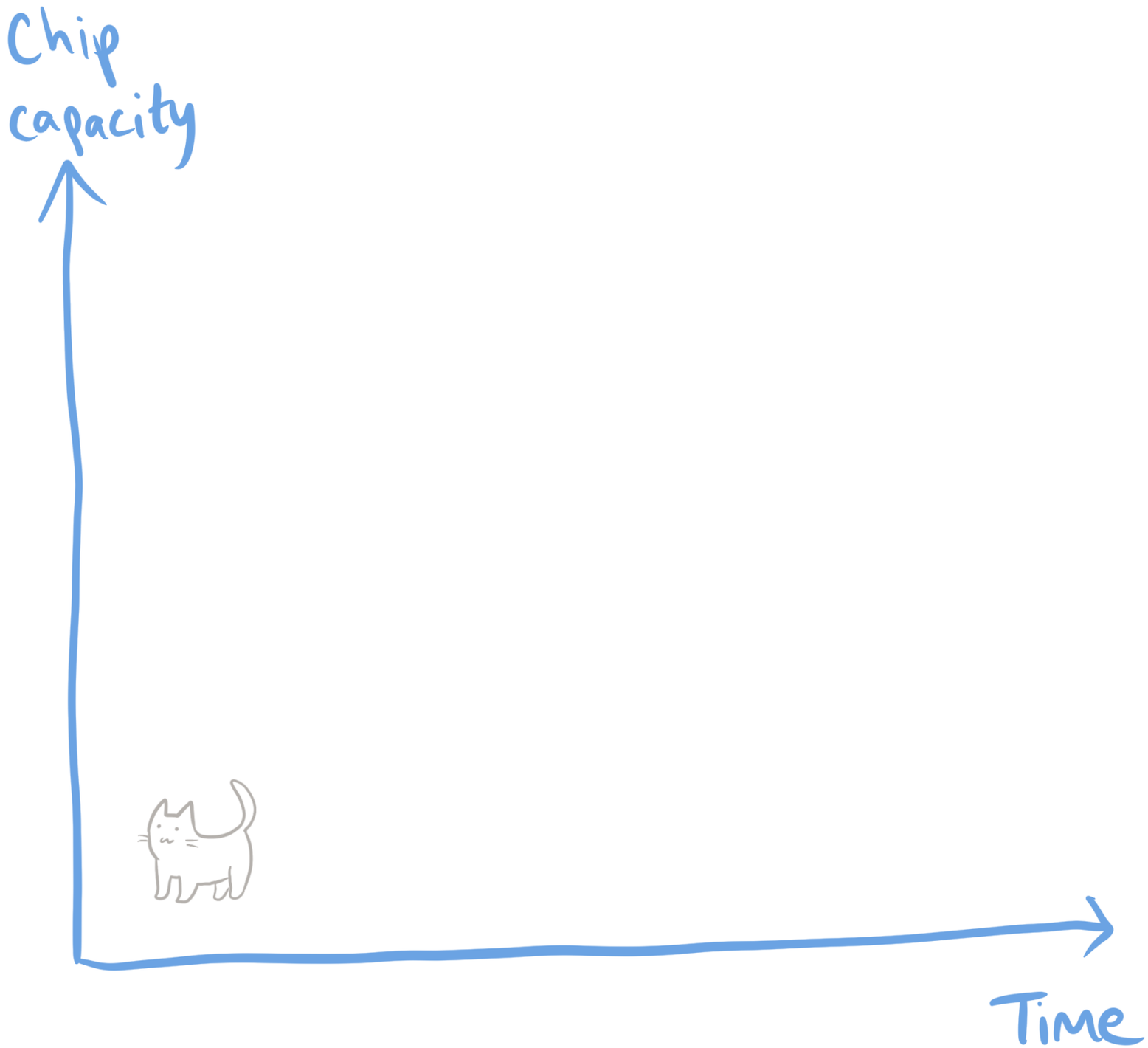
So we scaled vertically...



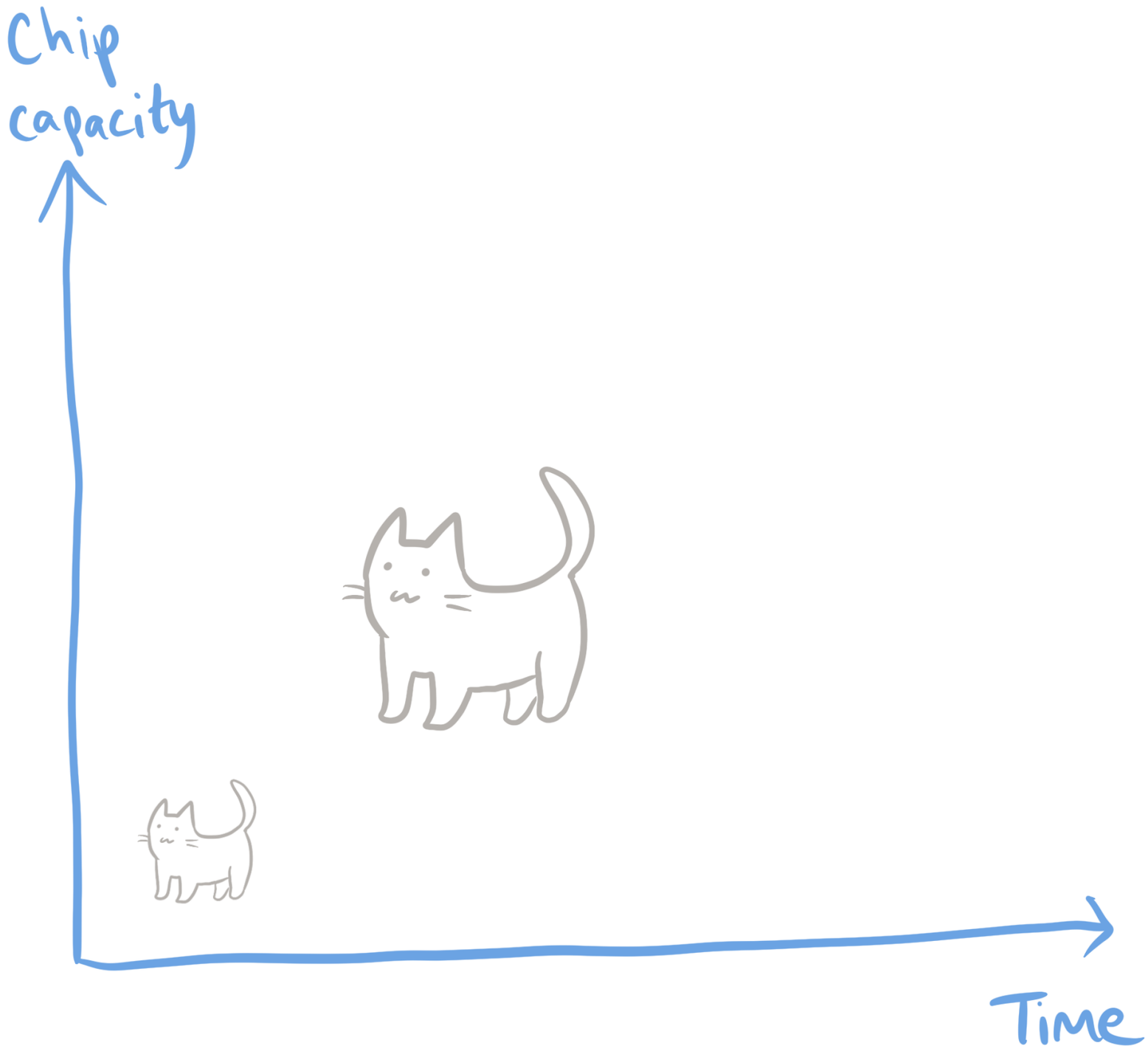
until unit economics (or physics) caught up.

@deniseyu21

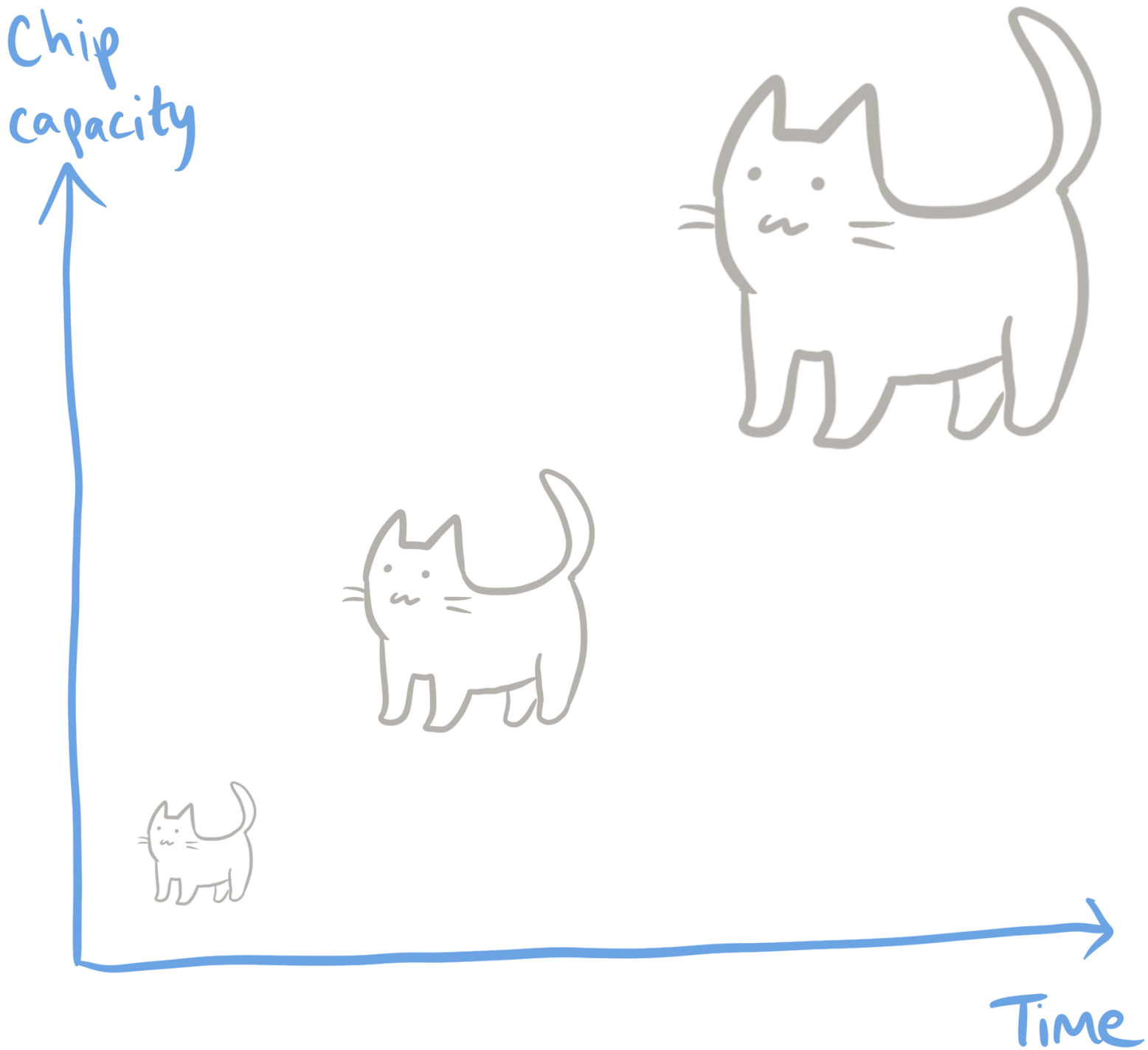
Moore's Law



Moore's Law



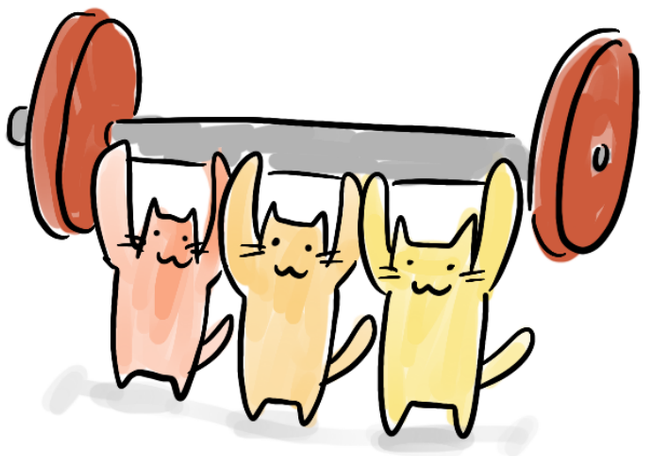
Moore's Law



@deniseyu21

REASONS TO HORIZONTALLY DISTRIBUTE:

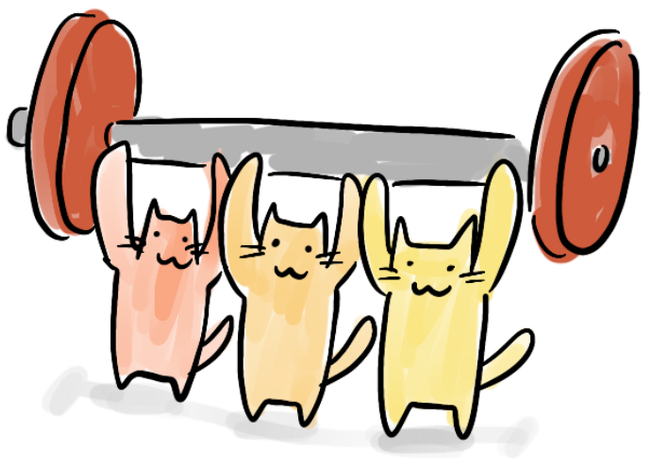
REASONS TO HORIZONTALLY DISTRIBUTE:



Scalability:

one machine cannot handle request or data size

REASONS TO HORIZONTALLY DISTRIBUTE:



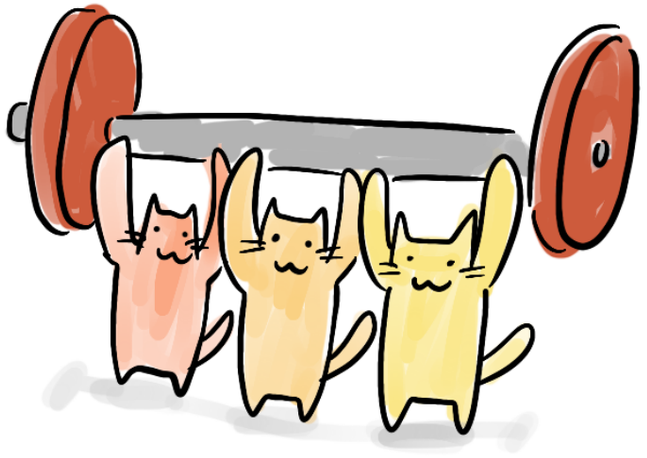
Scalability:

one machine cannot handle request or data size

Availability:

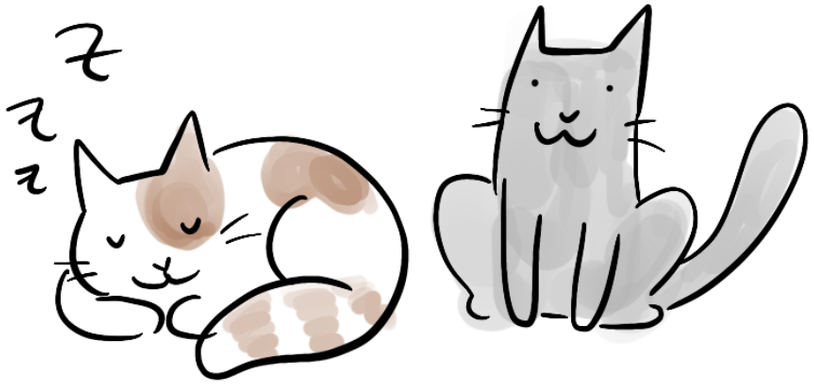
if one machine goes down, others keep working

REASONS TO HORIZONTALLY DISTRIBUTE:



Scalability:

one machine cannot handle request or data size



Availability:

if one machine goes down, others keep working

Latency:

go faster when data is stored geographically closer to users

@deniseyu21

You may have heard the term "shared nothing" architecture:

You may have heard the term "shared nothing" architecture:

This is mostly sensible, but it does force design trade-offs.

@deniseyu21

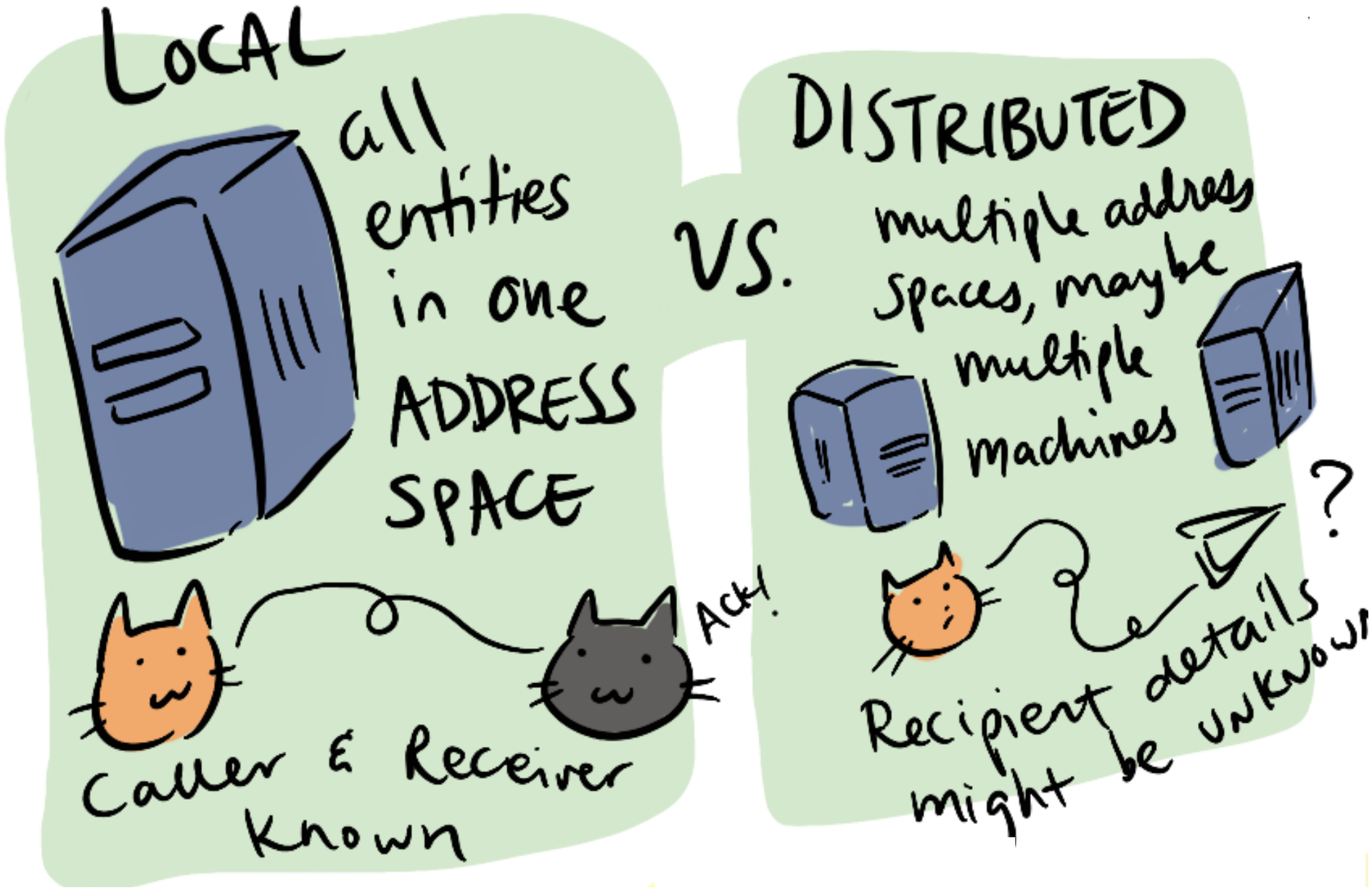
You may have heard the term "shared nothing" architecture:

"Don't communicate by sharing memory; share memory by communicating."

— Rob Pike

@deniseyu21

what does it actually mean to run a distributed system?


LOCAL
all entities in one ADDRESS SPACE
Caller & Receiver Known
Ack!
VS.
DISTRIBUTED
multiple address spaces, maybe multiple machines
?
Recipient details might be unknown

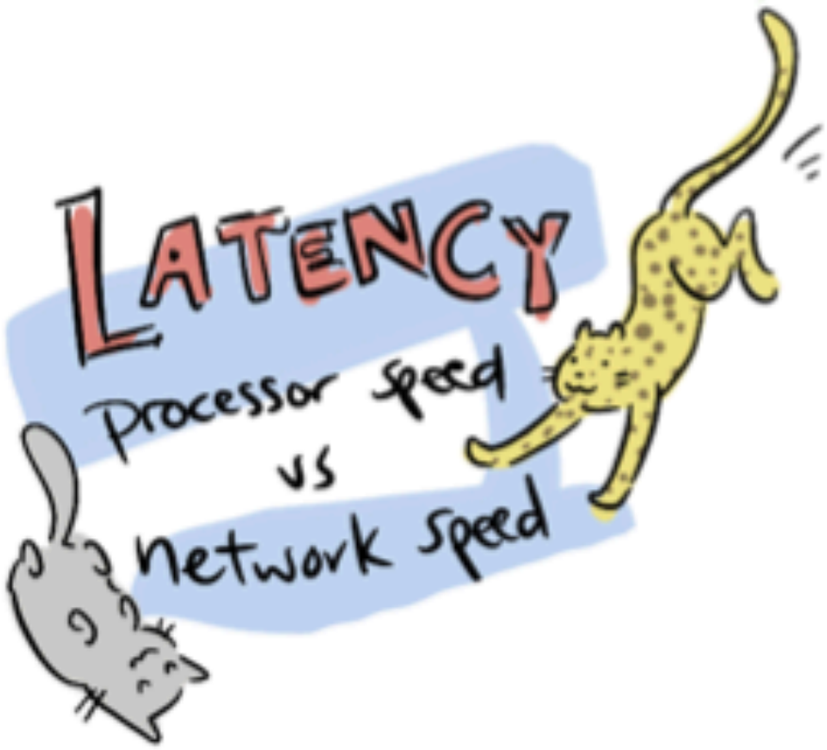
@deniseyu21

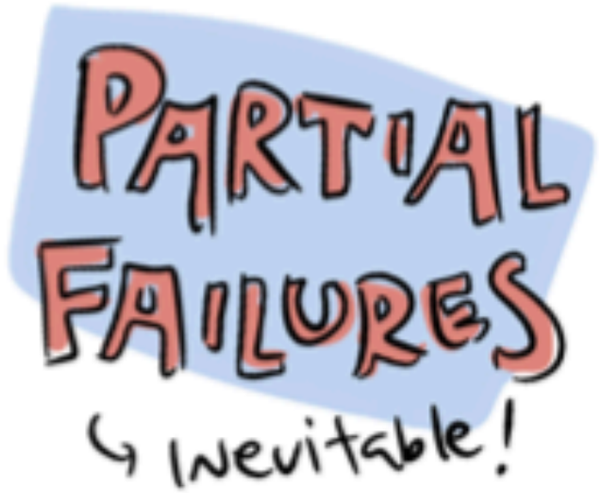
a note on distributed computing

1994

@deniseyu21

THE HARD PARTS!





@deniseyu21

Many machines,
running many processes,
and "only message-passing
via unreliable networks
with variable delays,
and the system may
suffer from partial failures..."

Martin Kleppmann,
"Designing Data Intensive Applications"

?

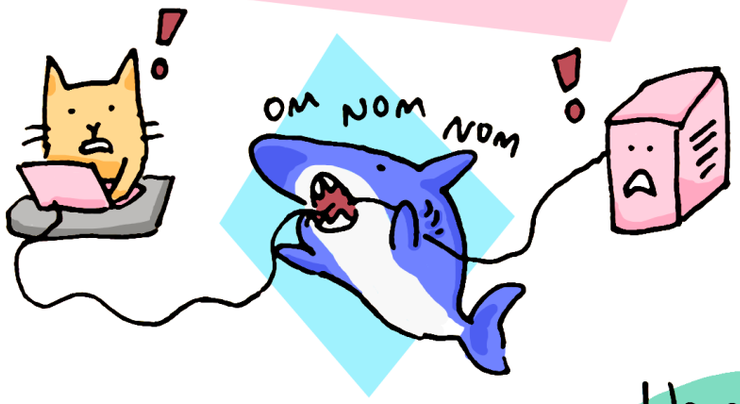
@deniseyu21

the 8 Fallacies of Distributed Computing

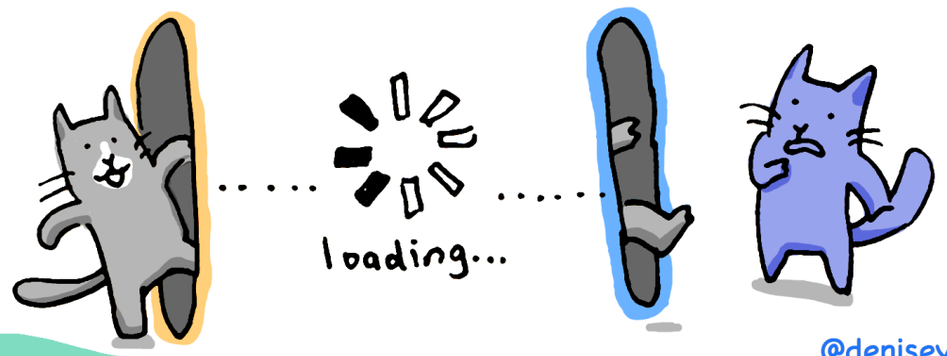
@deniseyu21

Originally formulated by L. Peter Deutsch & Colleagues at Sun Microsystems in 1994; #8 added in 1997 by James Gosling

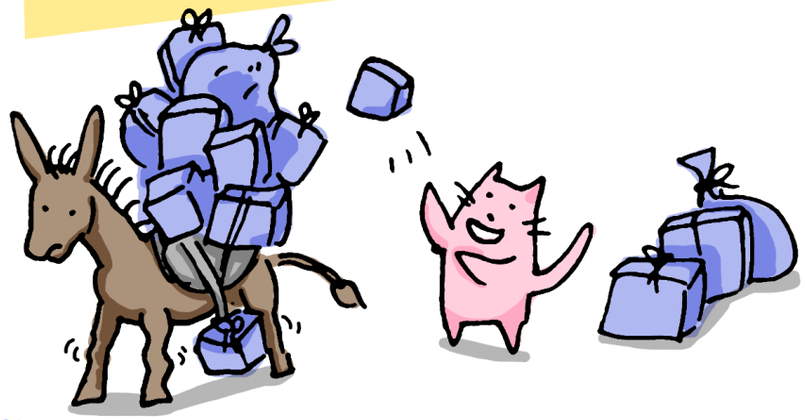
1) The network is reliable



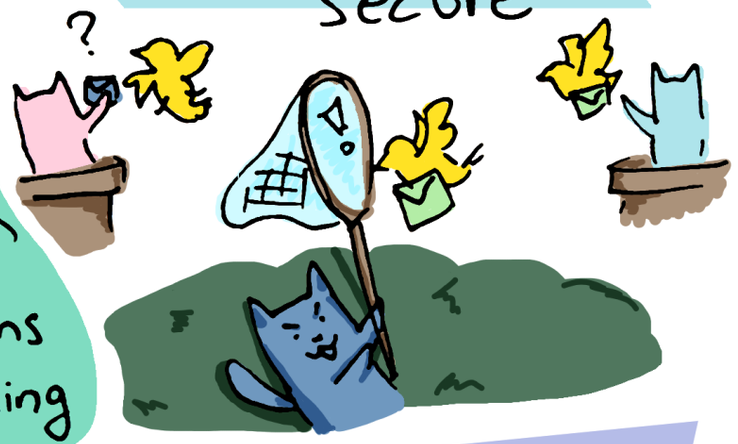
2) Latency is ZERO



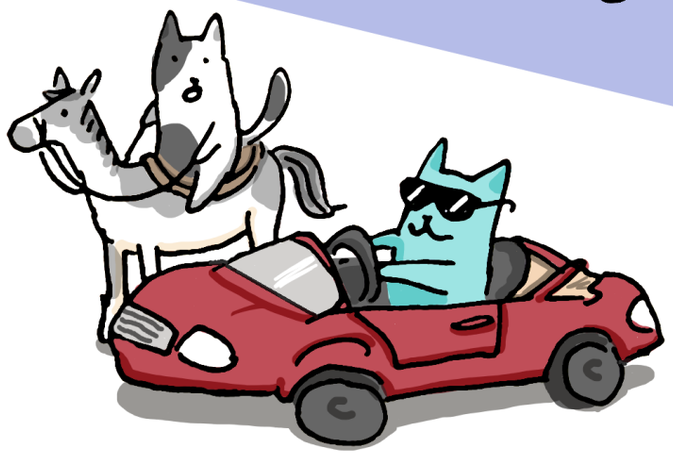
3) Bandwidth is infinite



4) The network is Secure



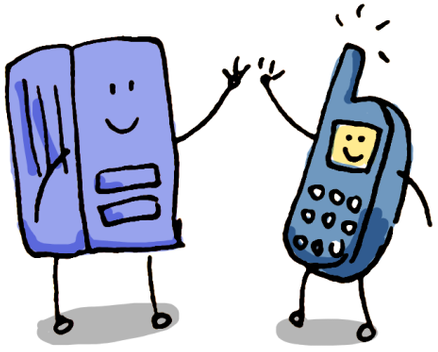
5) Topology doesn't change



6) There is only one administrator

7) Transport costs $0

8) The network is homogeneous

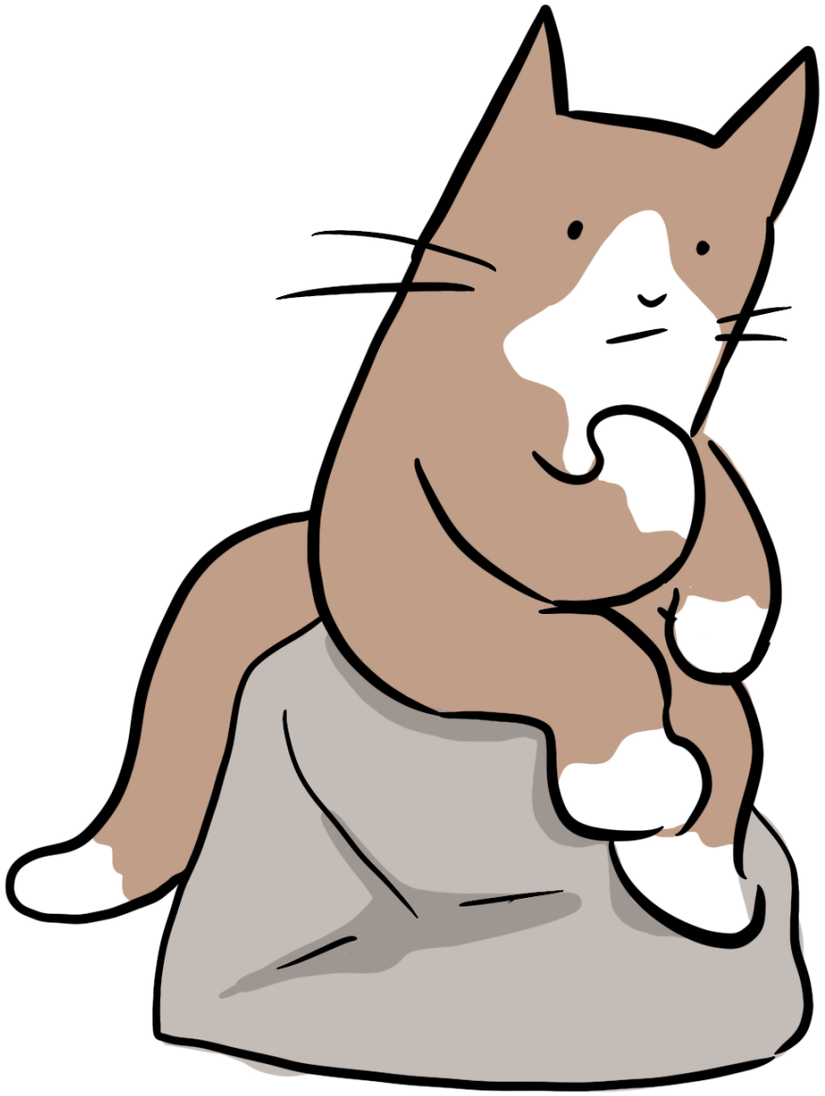


1 The network is reliable

So much unreliability!

How can we even know what is true about the state of the world?

@deniseyu21



This is an epistomology problem!

@deniseyu21

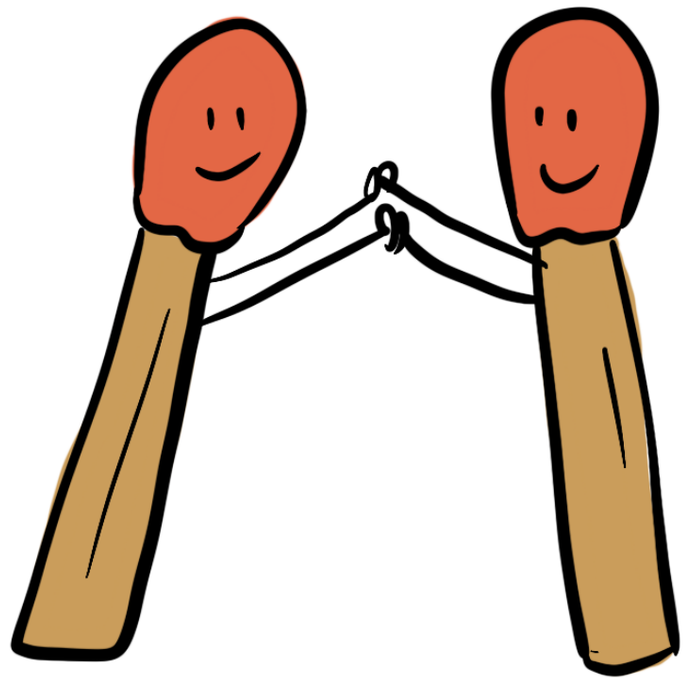
there are
TWO SCHOOLS
of epistemic PHILOSOPHY

FOUNDATIONALISM

Fundamental truths
like math first principles

FOUNDATIONALISM

Fundamental truths
like math first principles

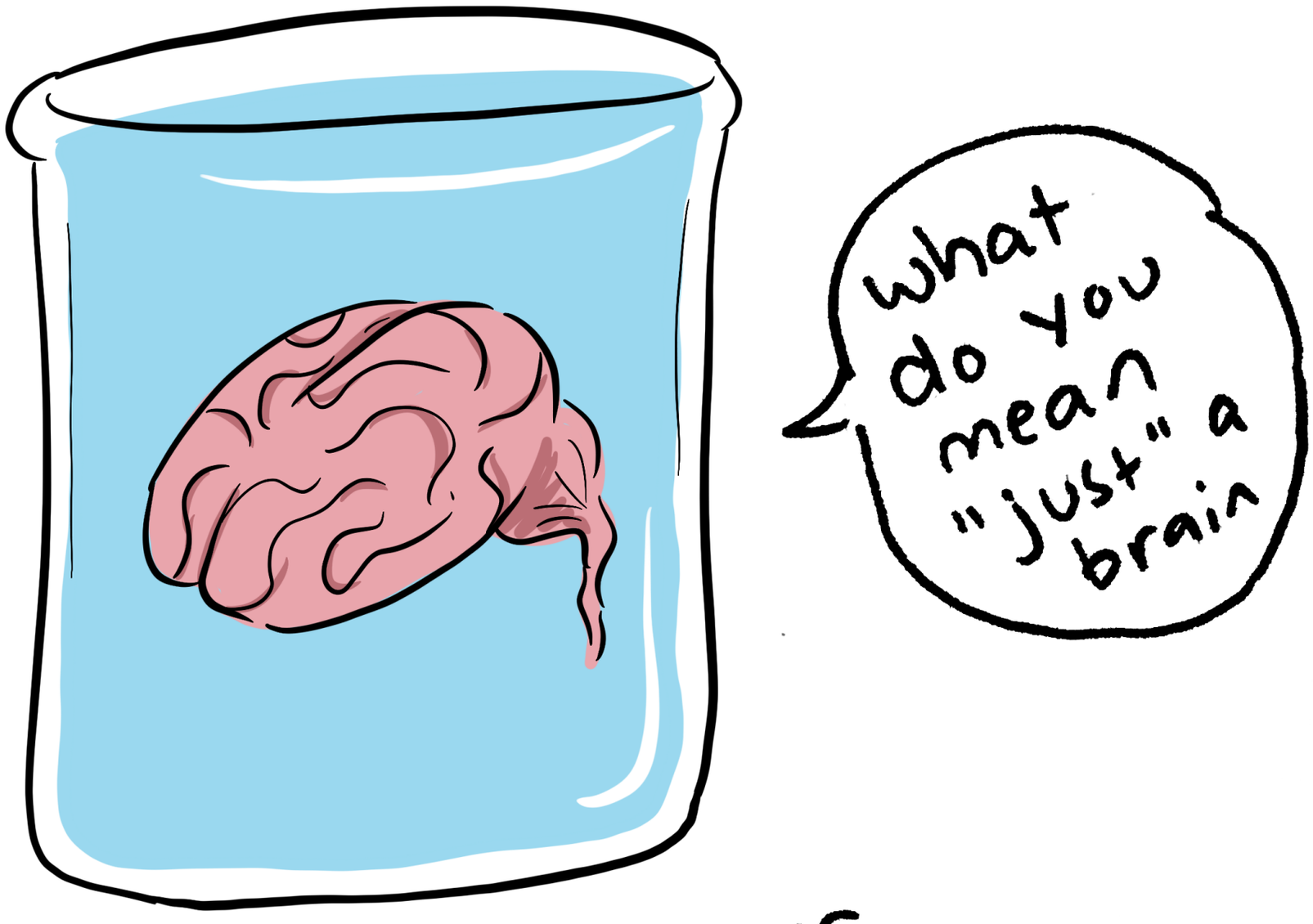


COHERENTISM

Logical, interlocking,
mutually-reinforcing
truths like
matchsticks

@deniseyu21

AND WHAT IF WE'RE ALL

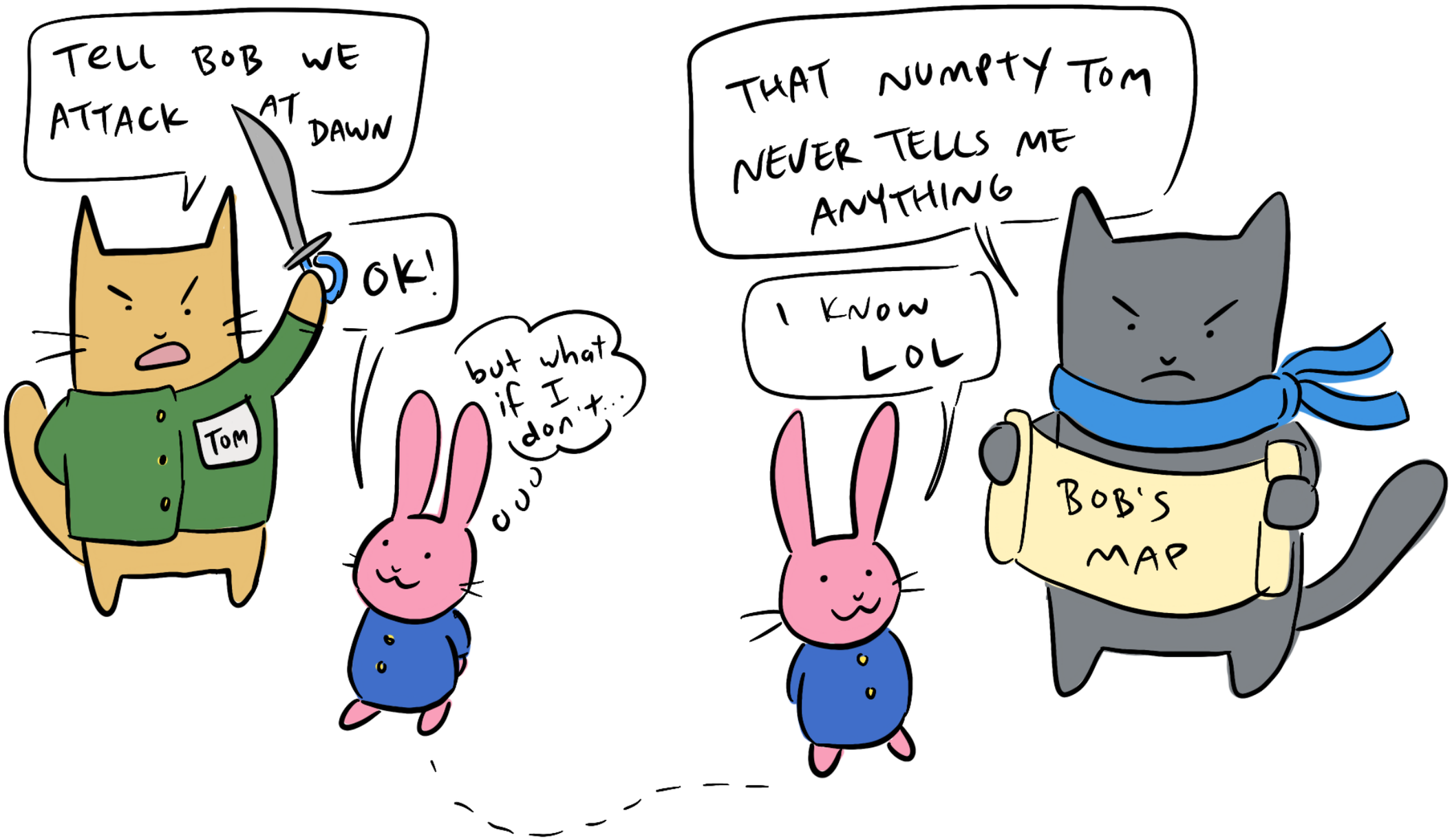


JUST BRAINS IN VATS

#skeptics

@deniseyu21

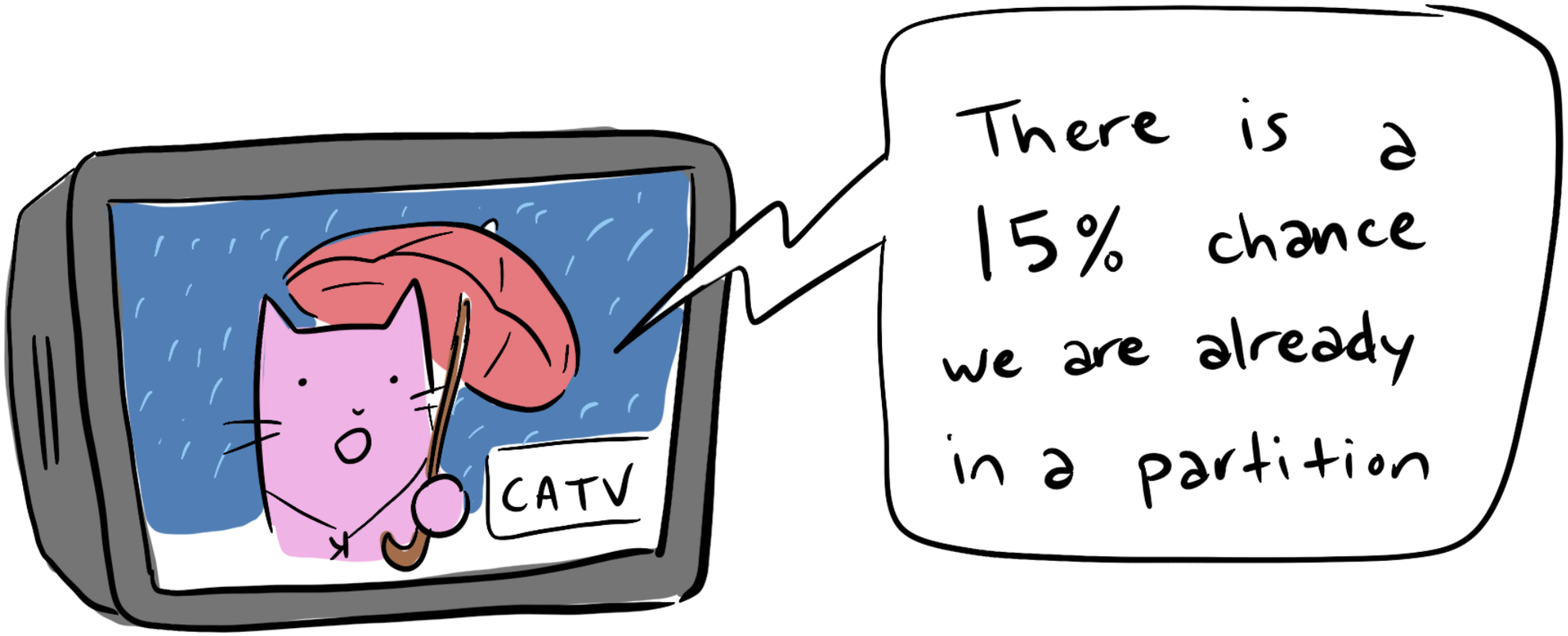
The Byzantine Generals Problem



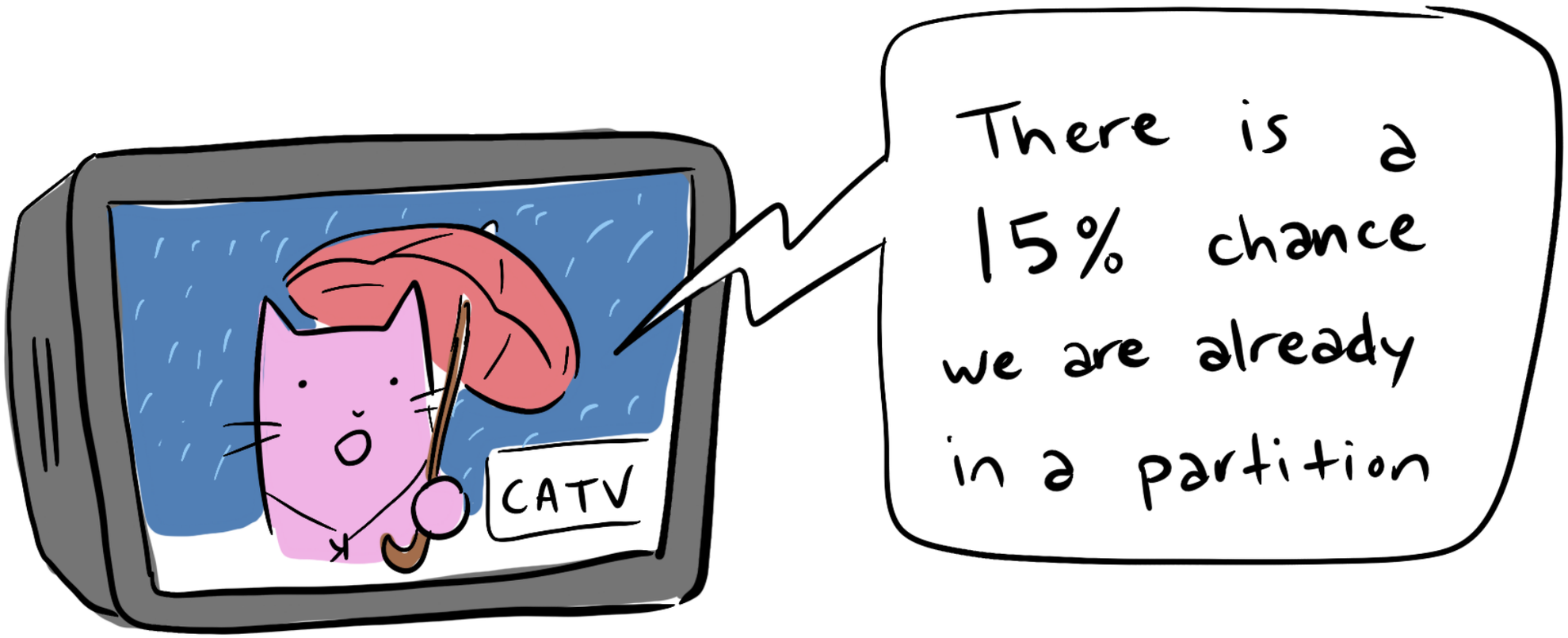
@deniseyu21

MONITOR & observe

@deniseyu21



There are lots we can't know. But in distributed computing, we can know one thing:



There are lots we can't know. But in distributed computing, we can know one thing:

Shit's gonna fail

@deniseyu21

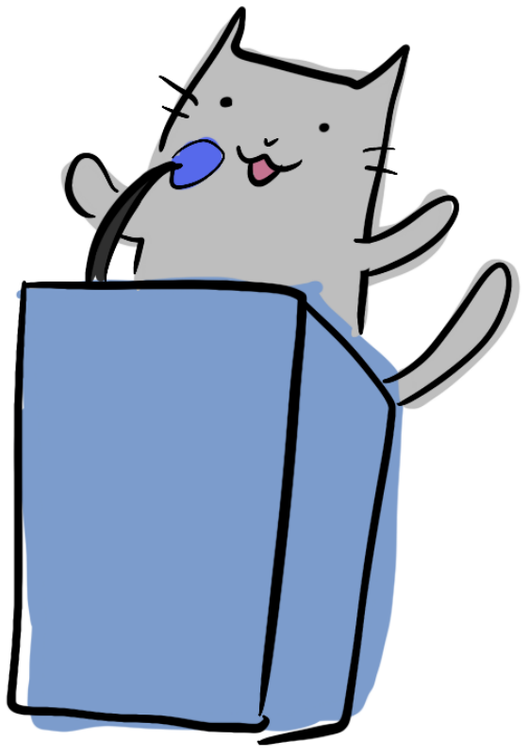
CAT

THEOREM

@deniseyu21

2000

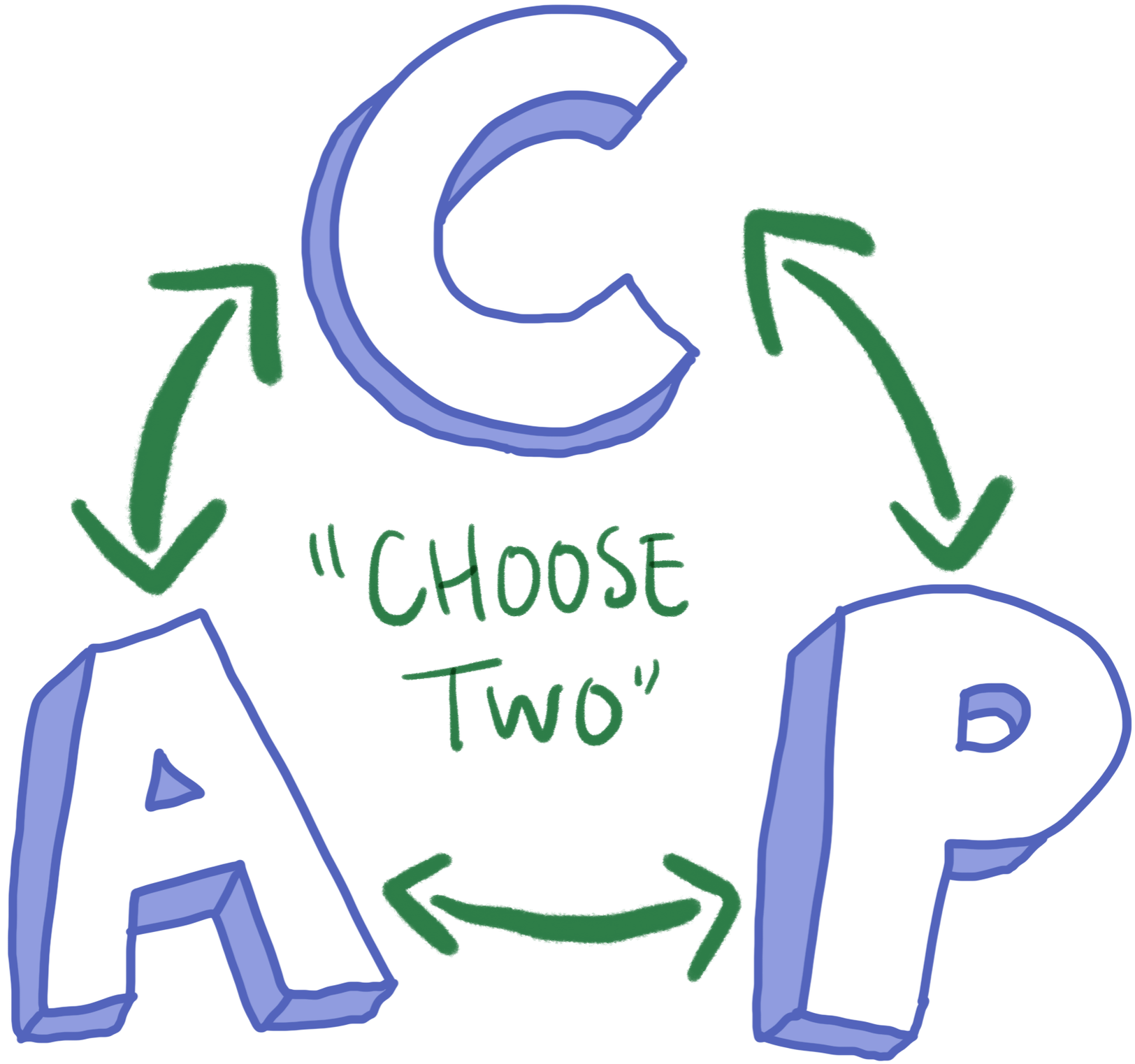
DR ERIC BREWER



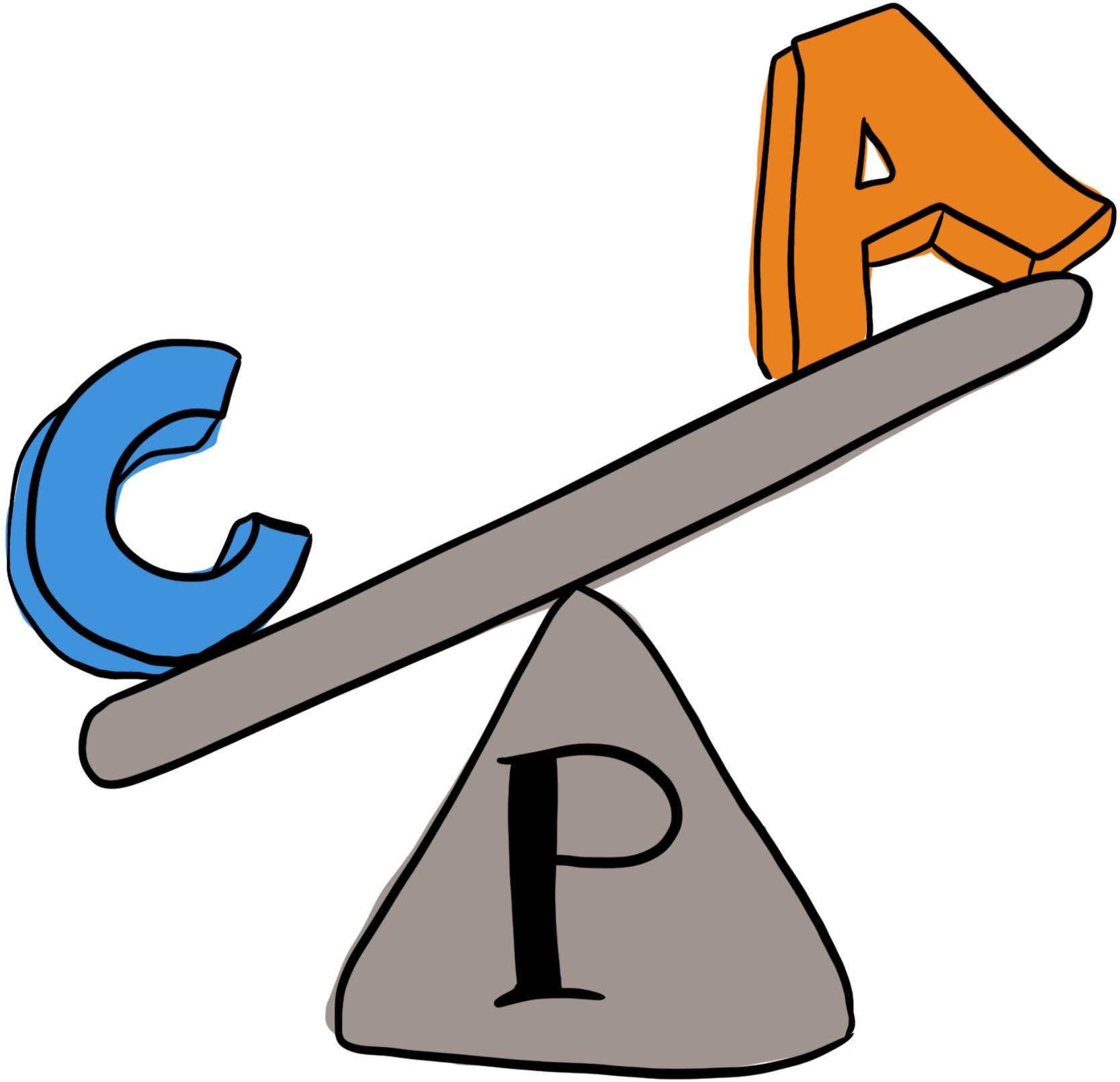
"Towards Robust Distributed Systems"

@ Principles of Computing Conf

@deniseyu21

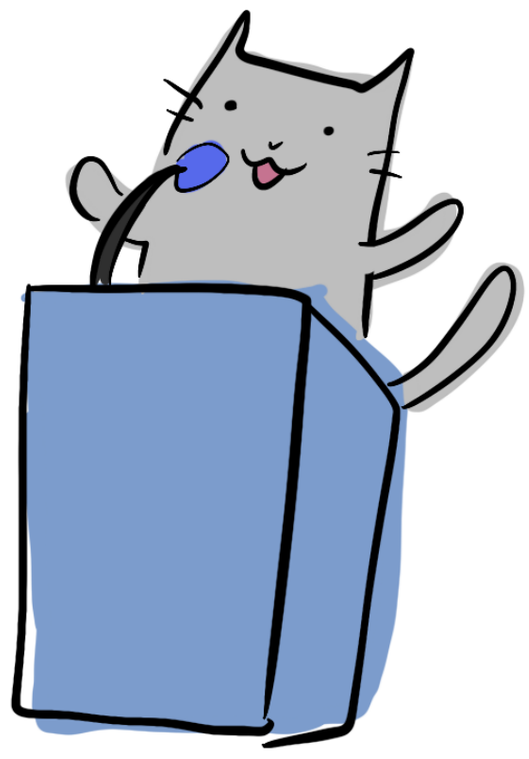

C
"CHOOSE
TWO"
A
P

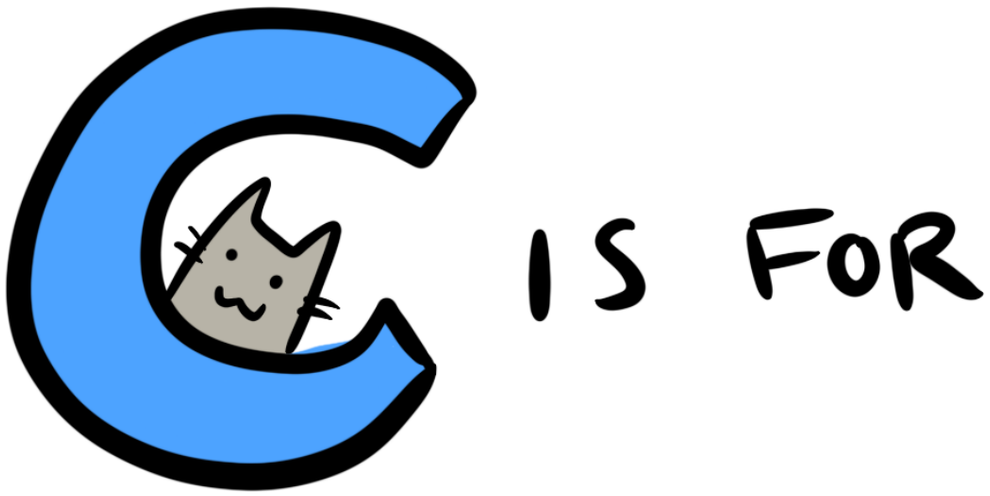

NO!

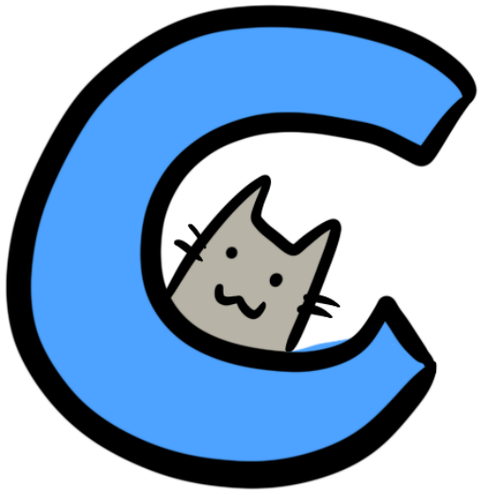

C
A
P

“Does choosing consistency and availability (CA) as the "2 of 3" make sense? As some researchers correctly point out, exactly what it means to forfeit P is unclear.”

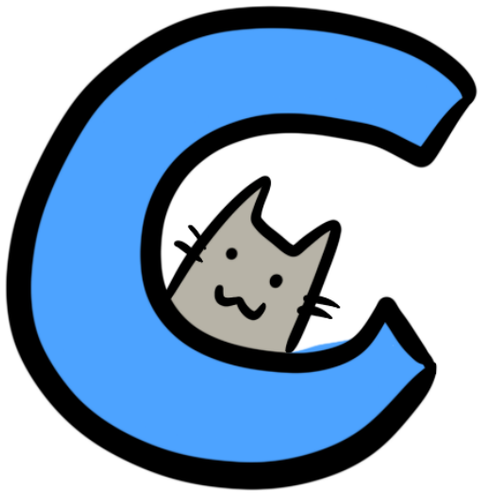
Eric Brewer, *CAP Twelve Years Later: How the 'Rules' Have Changed*



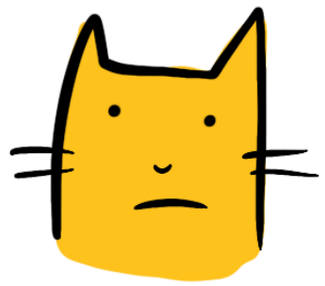

C IS FOR



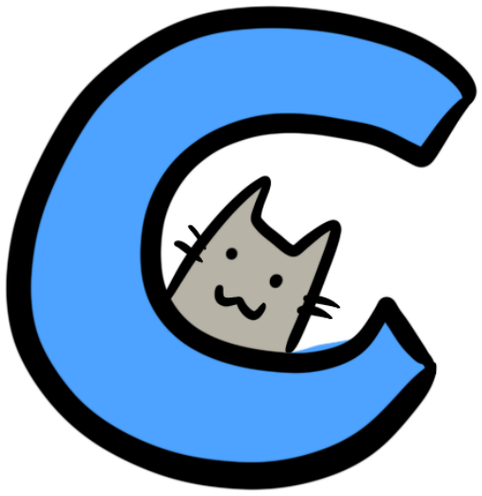
IS FOR LINEARIZABILITY



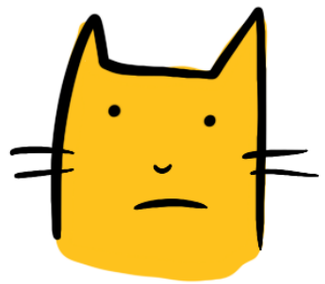
IS FOR LINEARIZABILITY

t_0  cat.state = 'hungry'

t_1  cat.state = 'full'

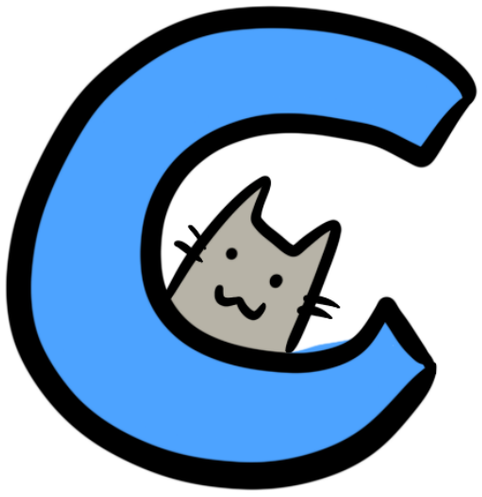


IS FOR LINEARIZABILITY

t_0  cat.state = 'hungry'

t_1  cat.state = 'full'

All nodes must have t_1 if anyone showed t_1



IS FOR LINEARIZABILITY

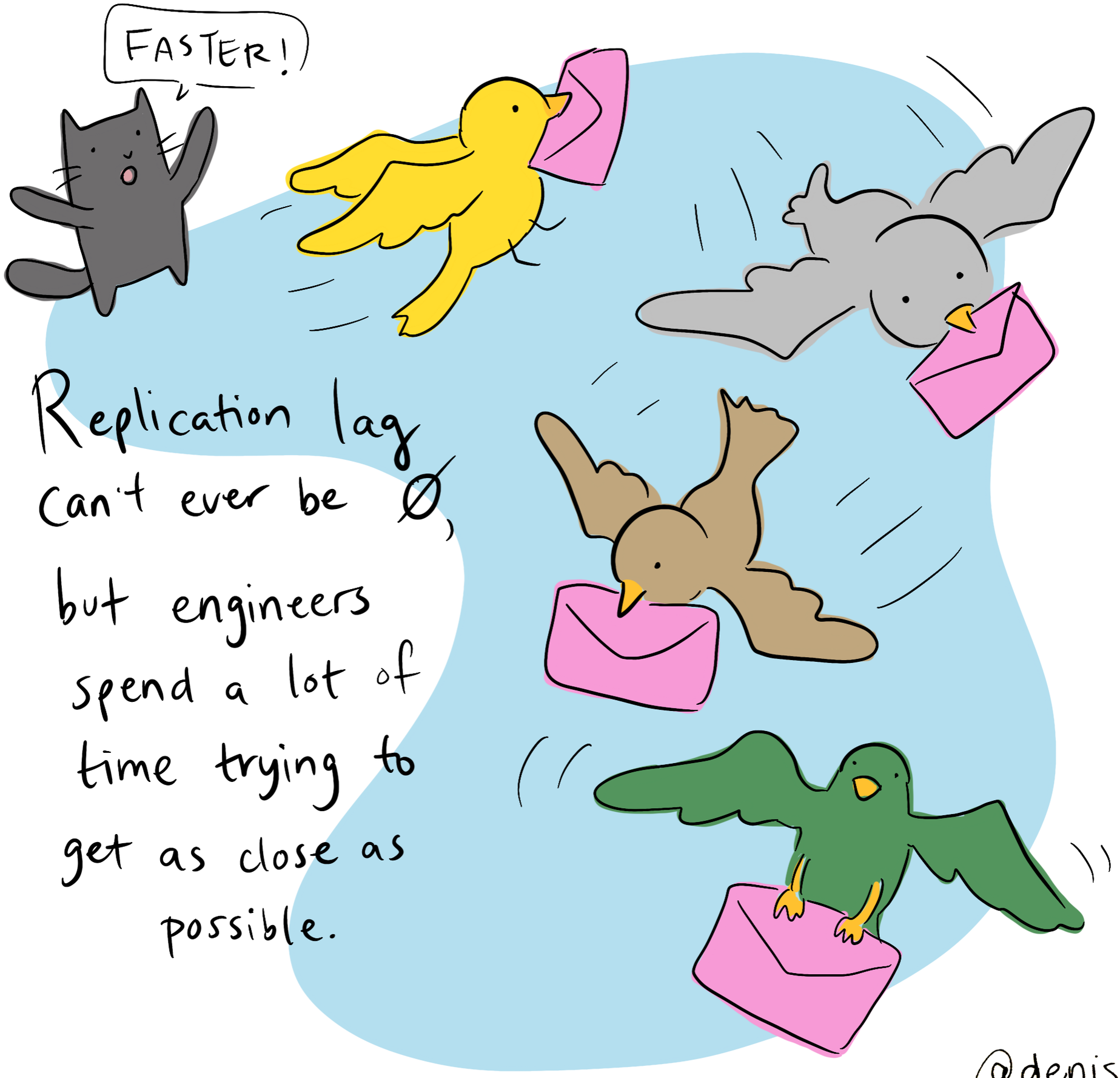
t_0 cat.state = 'hungry'

t_1  cat.state = 'full'

All nodes must have t_1 if anyone showed t_1

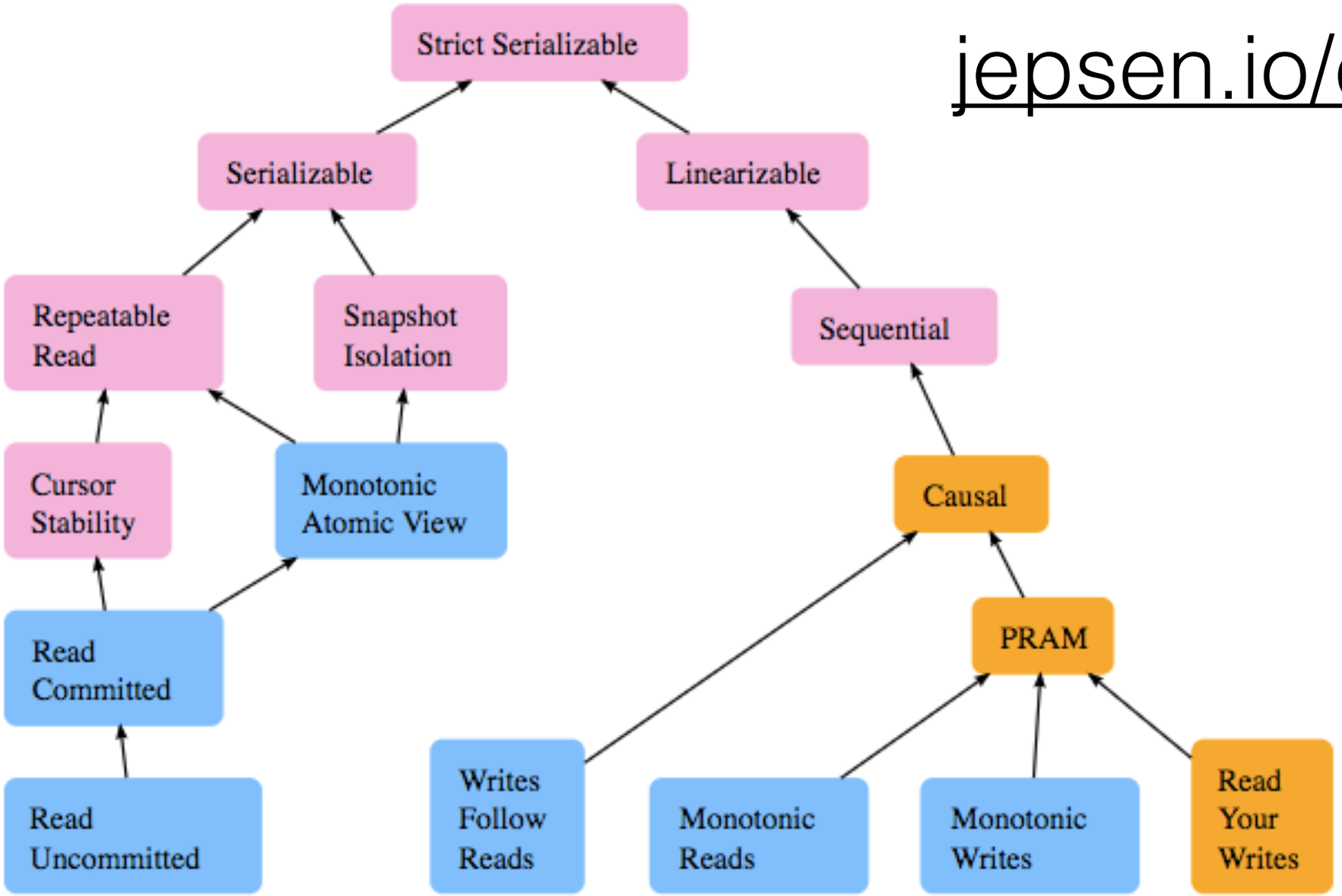
This is really hard! Instant & universal replication.

@deniseyu21


FASTER!
Replication lag
can't ever be ∅,
but engineers
spend a lot of
time trying to
get as close as
possible.
@deniseyu21

BTW, eventual consistency doesn't count.

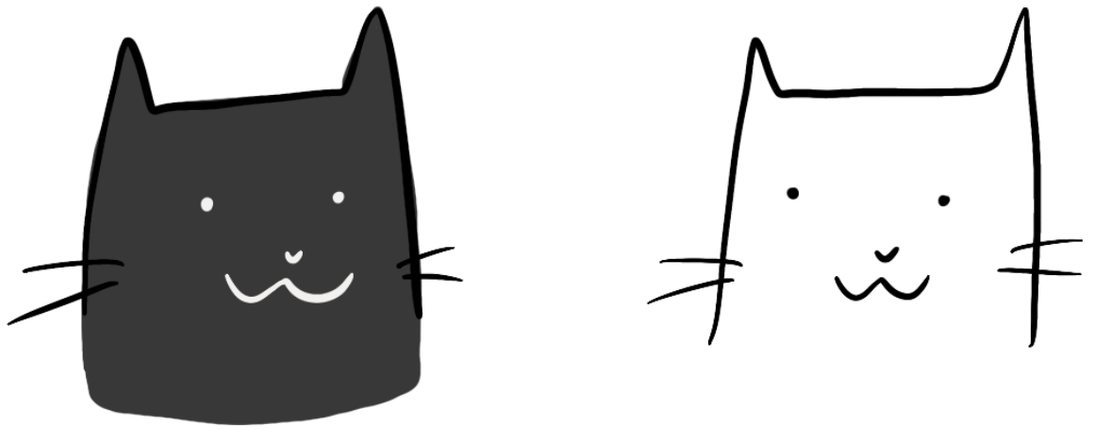
jepsen.io/consistency



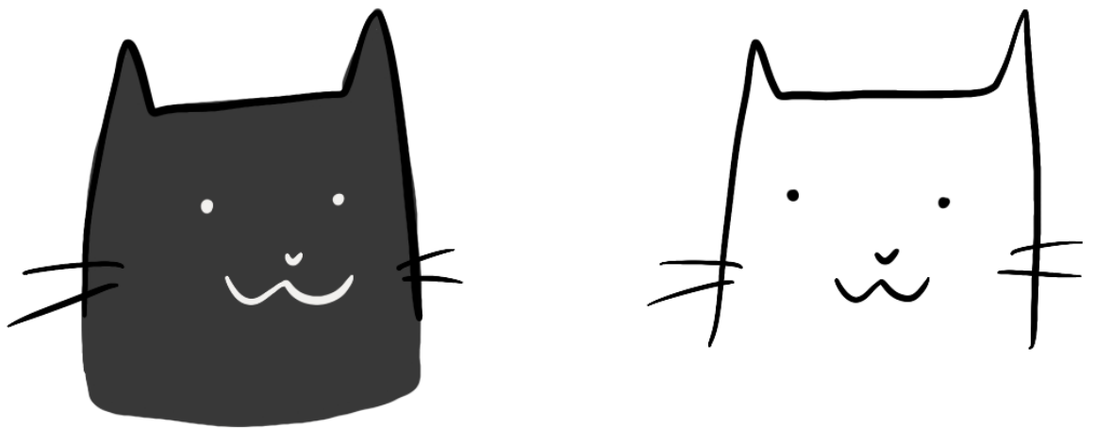
Legend

Unavailable	Not available during some types of network failures. Some or all nodes must pause operations in order to ensure safety.
Sticky Available	Available on every non-faulty node, so long as clients only talk to the same servers, instead of switching to new ones.
Total Available	Available on every non-faulty node, even when the network is completely down.

A IS FOR AVAILABILITY



A IS FOR AVAILABILITY



We tend to think of availability as a binary state. BUT — reality is much messier!

Because LATENCY

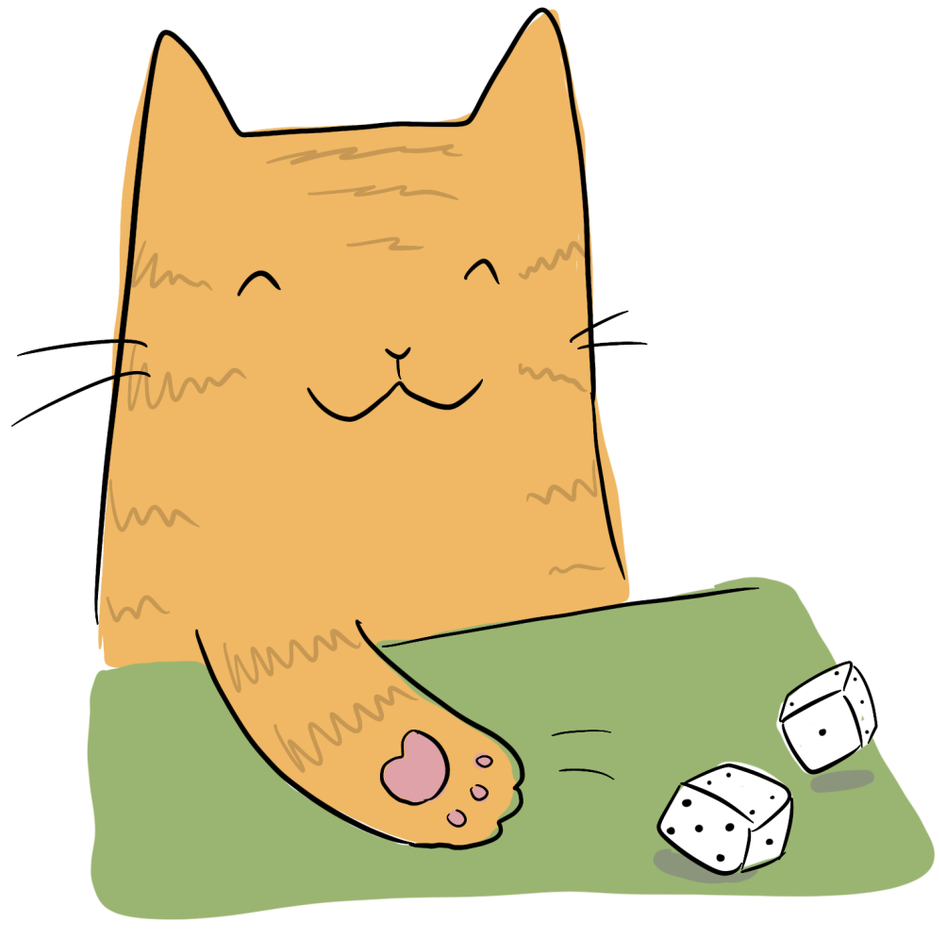
@deniseyu21

How can we know if a node is unresponsive...
or just slow?

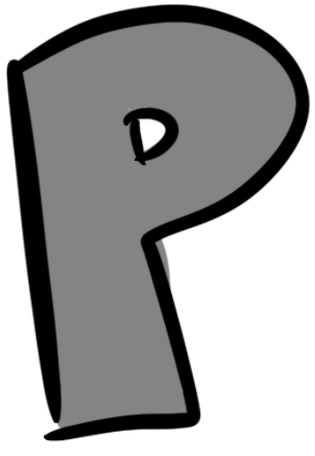
Network latency wasn't part of the original CAP formulation.

@deniseyu21

Determining a timeout limit is a very scientific process



yu21



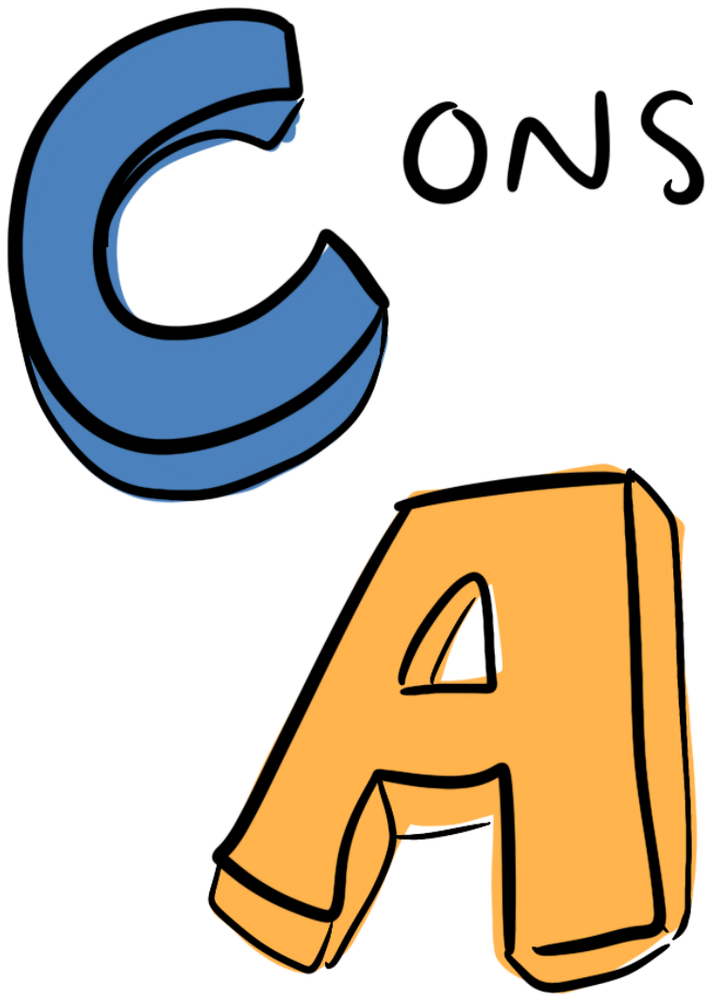
IS FOR PARTITION TOLERANCE

Network partitions occur when network connectivity between two datacenters (running your nodes!) is interrupted!

P IS FOR PARTITION TOLERANCE

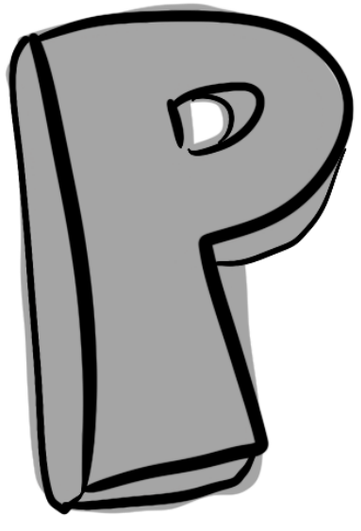
During a partition, your nodes might as well be on opposite sides of a wormhole: there is no way to know what's happening on the other side.

@deniseyu21



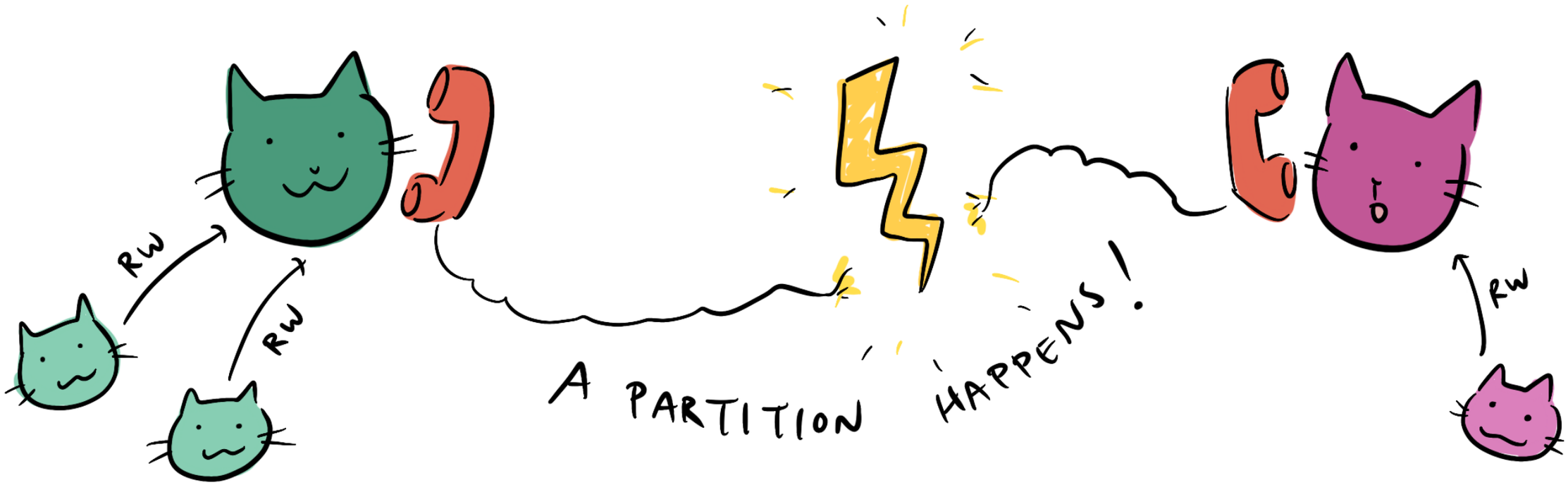
CONSISTENCY *

AVAILABILITY

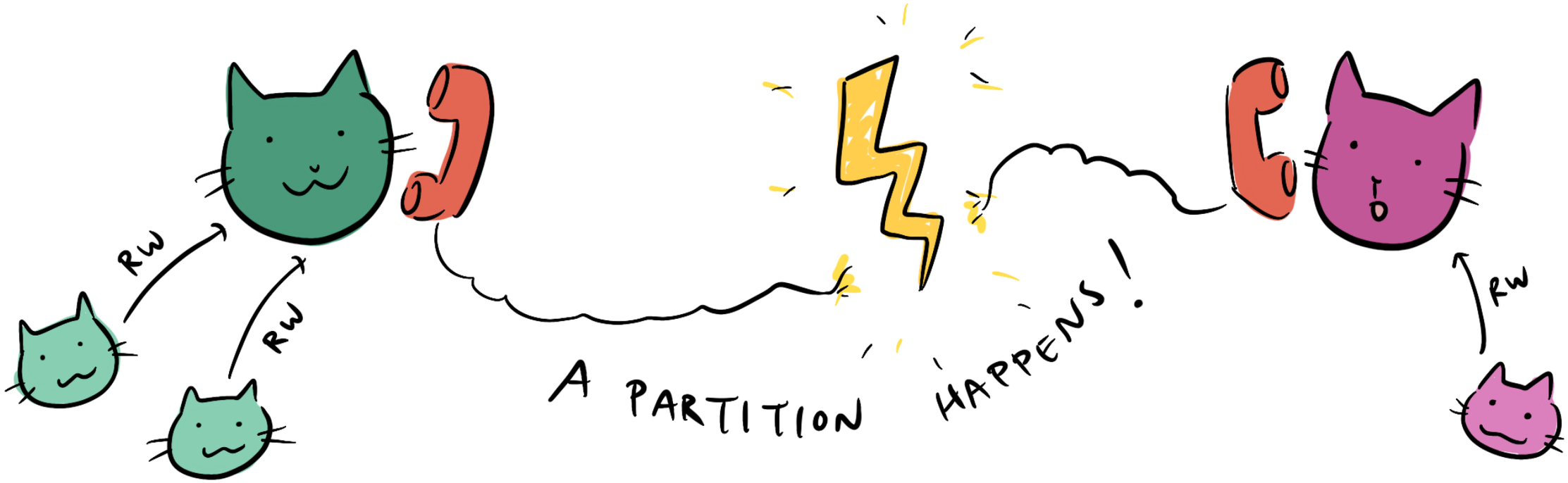


Partition tolerance

PROOF OF CAP THEOREM



PROOF OF CAP THEOREM

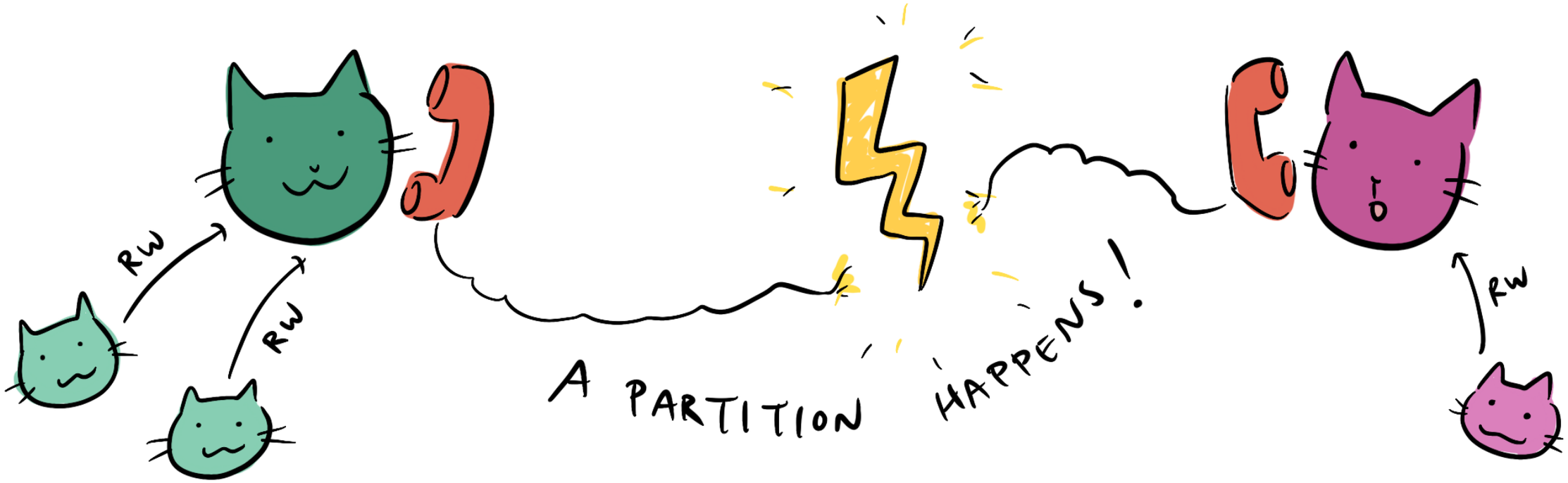


OPTION 1

Let clients keep R/W
in both sides of split

✗ LINEARIZABILITY

PROOF OF CAP THEOREM



OPTION 1

Let clients keep R/W in both sides of split

✗ LINEARIZABILITY

OPTION 2

Stop writing in one side until partition ends

✗ AVAILABILITY

@deniseyu21

PARTITION TOLERANCE

Network partitions
are inevitable.

How inevitable?

In the first year of a Google cluster's life, it will experience

5 rack failures

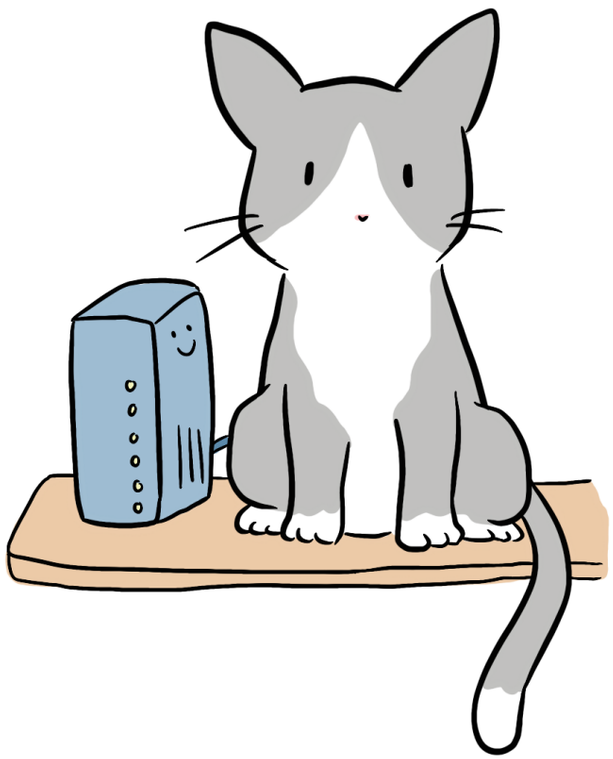
3 router failures

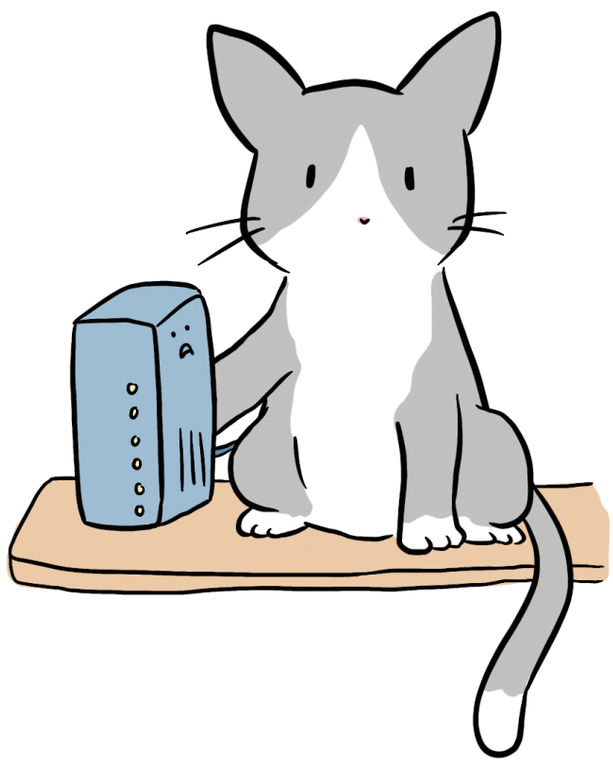
8 network maintenances

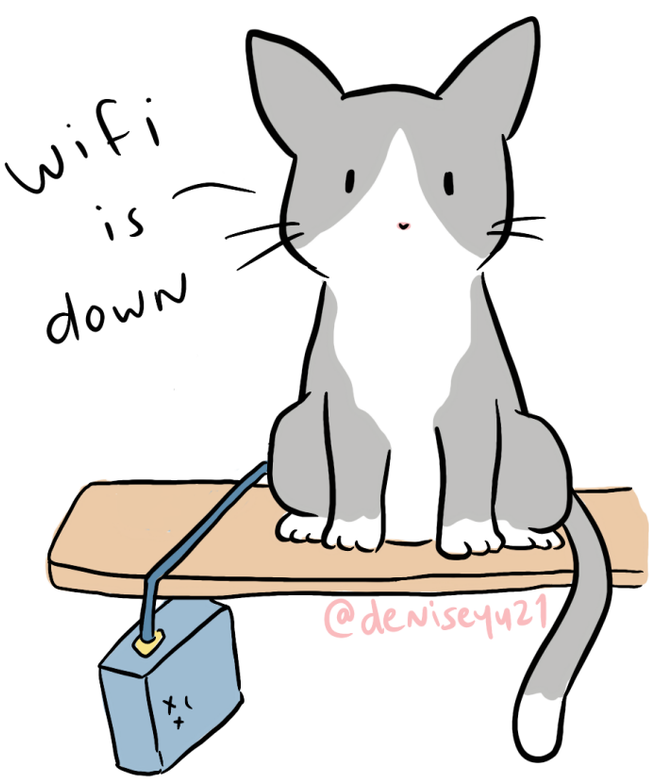
(Jeffrey Dean)

Hardware will fail

Hardware will fail

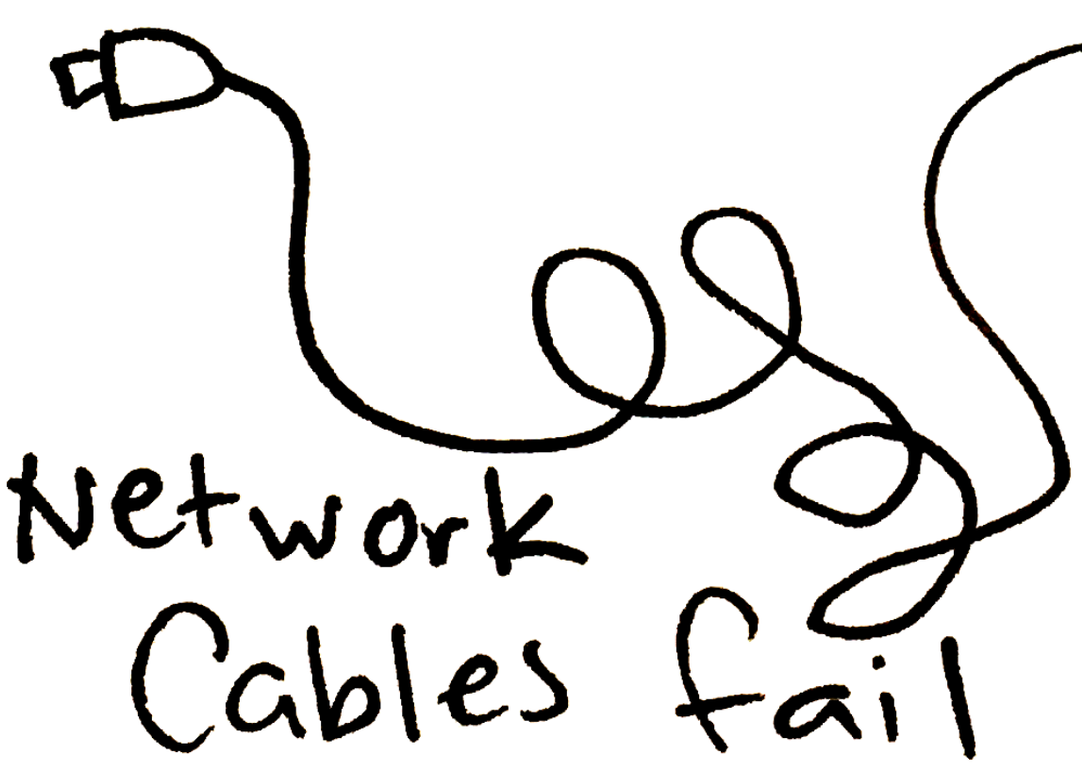






@deniseyu21

Hardware will fail



Network Cables fail

@deniseyu21

Hardware will fail

@deniseyu21

POLICY —

It's official: Sharks no longer a threat to subsea Internet cables

First known cable shark attacks were in 1985.

DAVID KRAVETS - 7/10/2015, 5:16 PM

Software will
behave weirdly

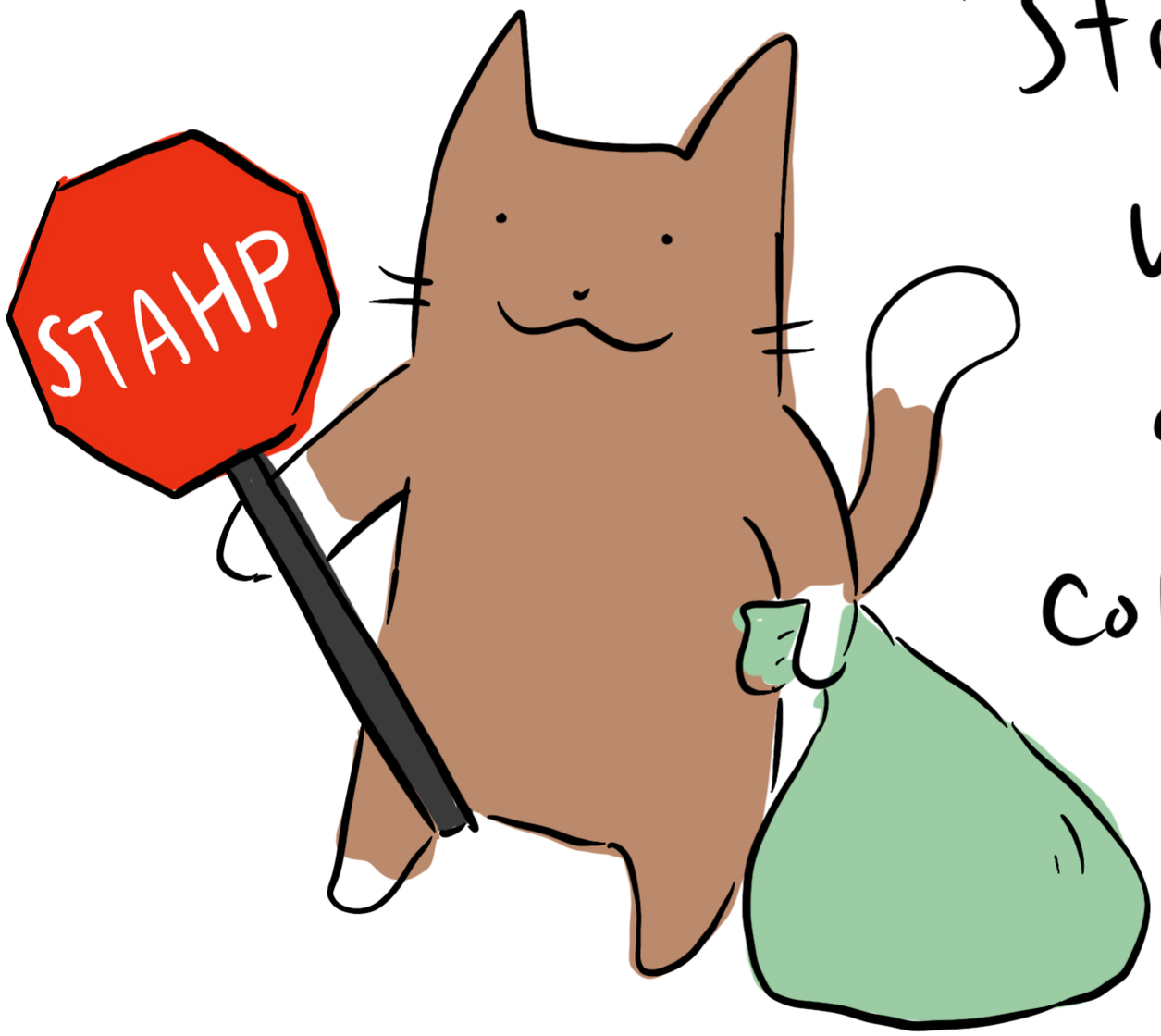
Software will behave weirdly

"Bursty" VMs borrow resources from each other—

the Noisy Neighbor Problem

deniseyu21

Software will behave weirdly

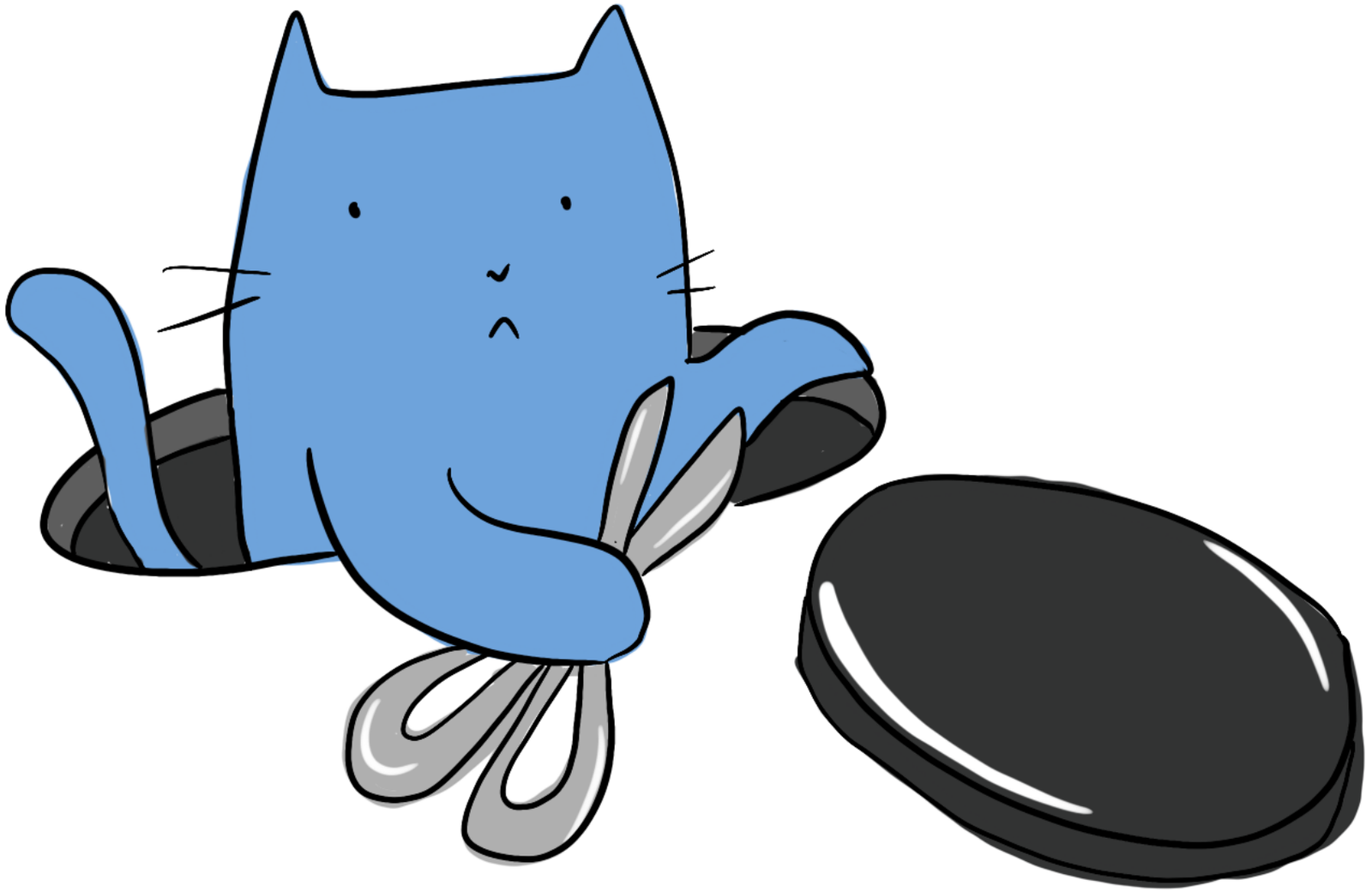


"Stop the WORLD" garbage Collection

@deniseyu21

Network glitches

@deniseyu21



@deniseyu21

@deniseyu21

🌕🦊and his hair was perfect 🦊
@palvaro

another highlight:

Me: so... what is the hard thing about distributed systems? if you had to pick one word...

🌕🦊and his hair was perfect 🦊
@palvaro

another highlight:

Me: so... what is the hard thing about distributed systems? if you had to pick one word...

Student 1: uncertainty?

Me: *beaming* GOOD. *writes it on the board*

🌕🦊and his hair was perfect 🦊
@palvaro

another highlight:

Me: so... what is the hard thing about distributed systems? if you had to pick one word...

Student 1: uncertainty?

Me: *beaming* GOOD. *writes it on the board*

Student 2: Docker

🌕🦊and his hair was perfect 🦊
@palvaro

another highlight:

Me: so... what is the hard thing about distributed systems? if you had to pick one word...

Student 1: uncertainty?

Me: *beaming* GOOD. *writes it on the board*

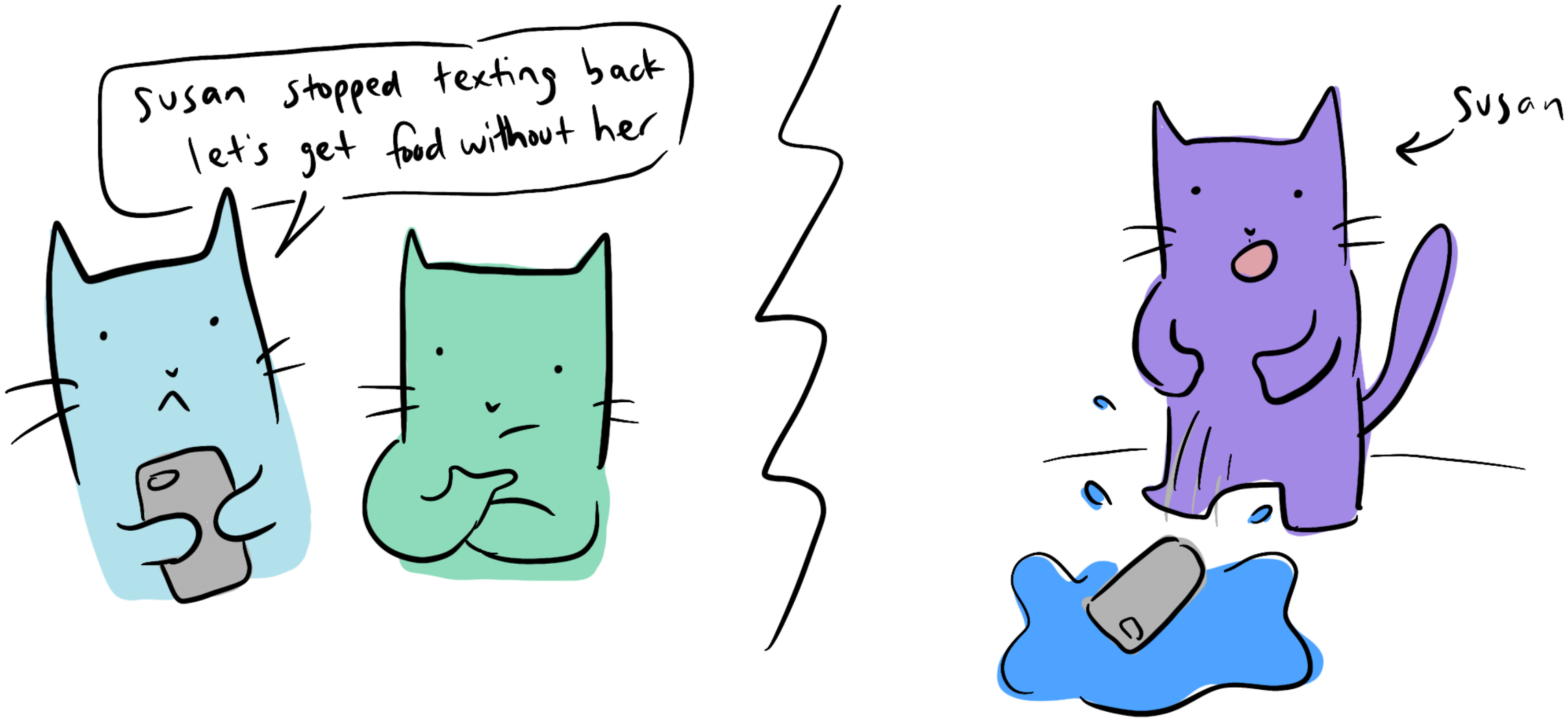
Student 2: Docker

Student 1: actually that's better, take mine off

6:33 PM · Oct 20, 2019 · Twitter Web App

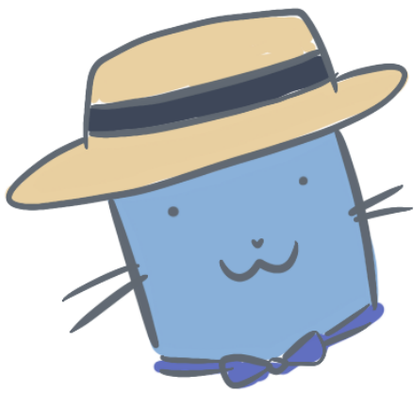
why does any of this matter?

Some part of every system is always at risk of failing



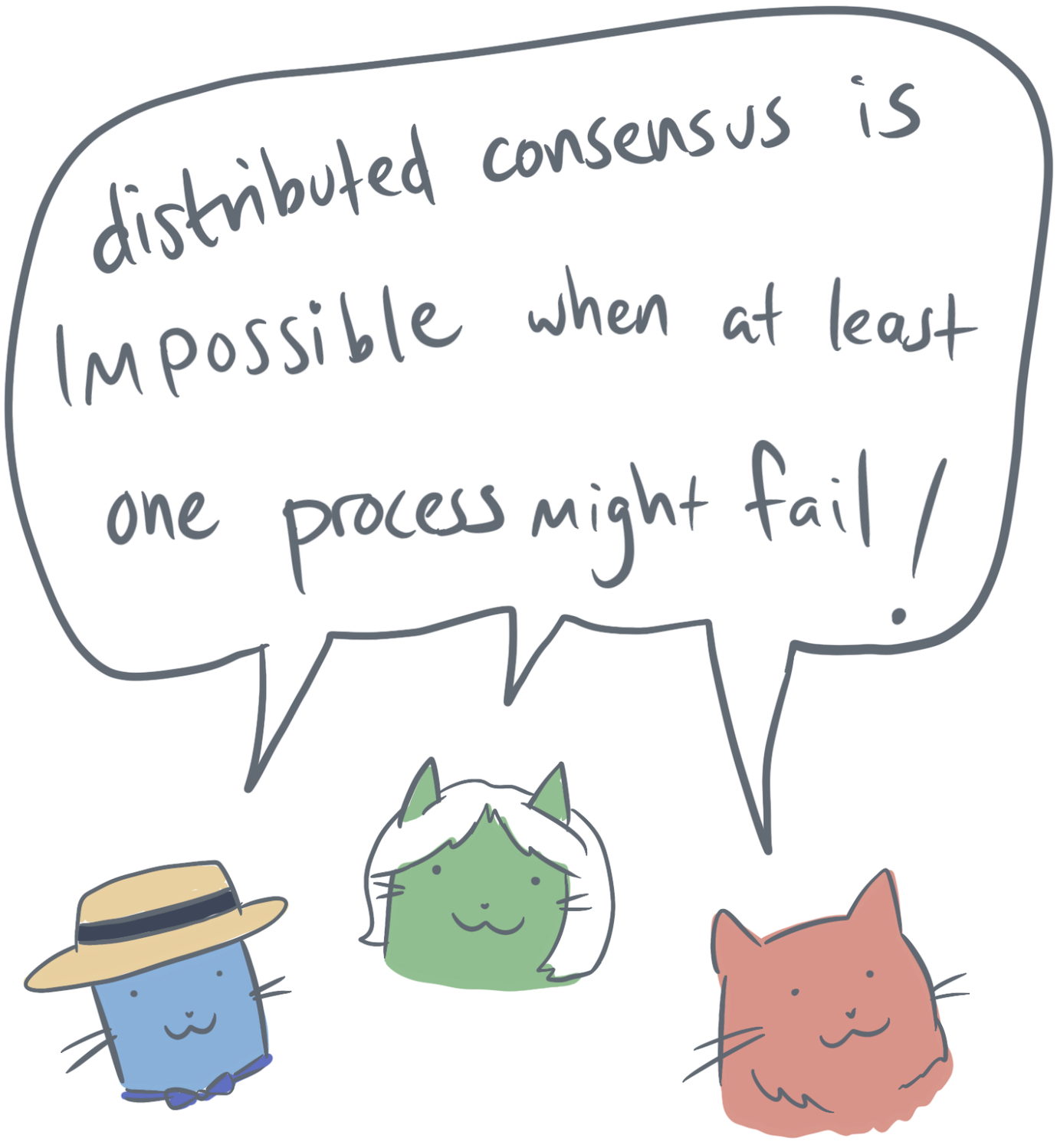
@deniseyu21

Fischer Lynch Paterson



Correctness result

@deniseyu21



“Impossibility of Distributed Consensus with One Faulty Process, 1985.

@deniseyu21

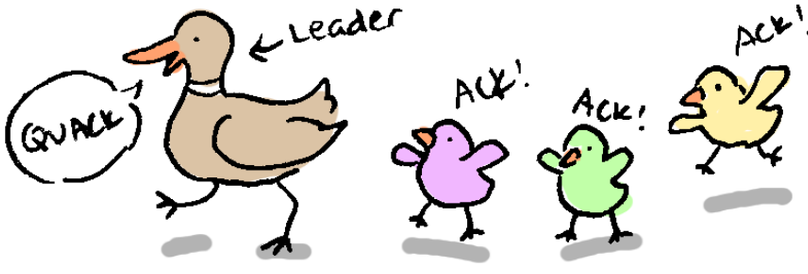
to manage uncertainty, we have mitigation strategies

limit who can write

at any point in time

what is AUTOMATIC LEADER FOLLOWER ?

LEADER-FOLLOWER* is a common pattern for REPLICATING data across nodes in a cluster. If a follower node fails, it's NO BIG DEAL usually, because other nodes can continue to serve READ requests!



STEP ONE

Detect that a node failed! most databases have a default TIMEOUT (ex. 30s)

If a node has no activity for this time, it is considered to be offline

WHAT IF THE LEADER NODE FAILS?

the cluster must initiate a FAILOVER to choose a new leader!

STEP TWO

From the remaining followers, elect a new leader!

Becky's data is most up to date I vote Becky

Becky

YEH OK I'll do it

Some databases have a CONTROLLER NODE who chooses

He has ONE JOB!

STEP THREE

Tell the world about the new LEADER

I AM THE NEW LEADER!

yea Becky we know

THIS WAY

Traffic will be routed to the new leader for all future write requests

* You may have encountered "master-slave replication". I prefer not to use this term, because as technologists, our language matters. We should choose terms that are inclusive & don't cause harm.

@deniseyu21
sketch-ops.tumblr.com

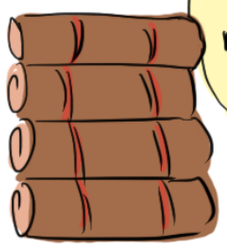
make rules for how many "yes" votes is enough to proceed

how does RAFT work?

RAFT is a CONSENSUS ALGORITHM

used in many projects in the real world - ex. etcd

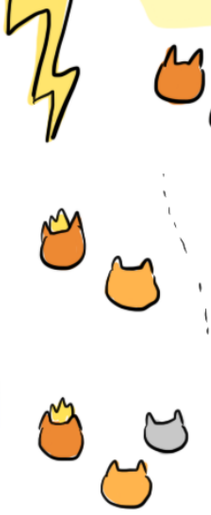
a process for getting multiple machines to "agree"!



- RAFT doesn't stand for anything. It's a bunch of logs tied together.

WHY? Distributed systems come with uncertainty. Achieving consensus is important so data can be replicated!

WHAT ABOUT NETWORK PARTITIONS?!



In a partition, if the leader falls on the smaller side, a new leader is elected within the majority side. When the partition ends, messages received by the minority side are discarded, and those nodes converge their state to match the majority's.

WHEN A NEW WRITE HAPPENS:

(Any node can write!)

Message is forwarded to leader node

Message is queued

Followers copy leader

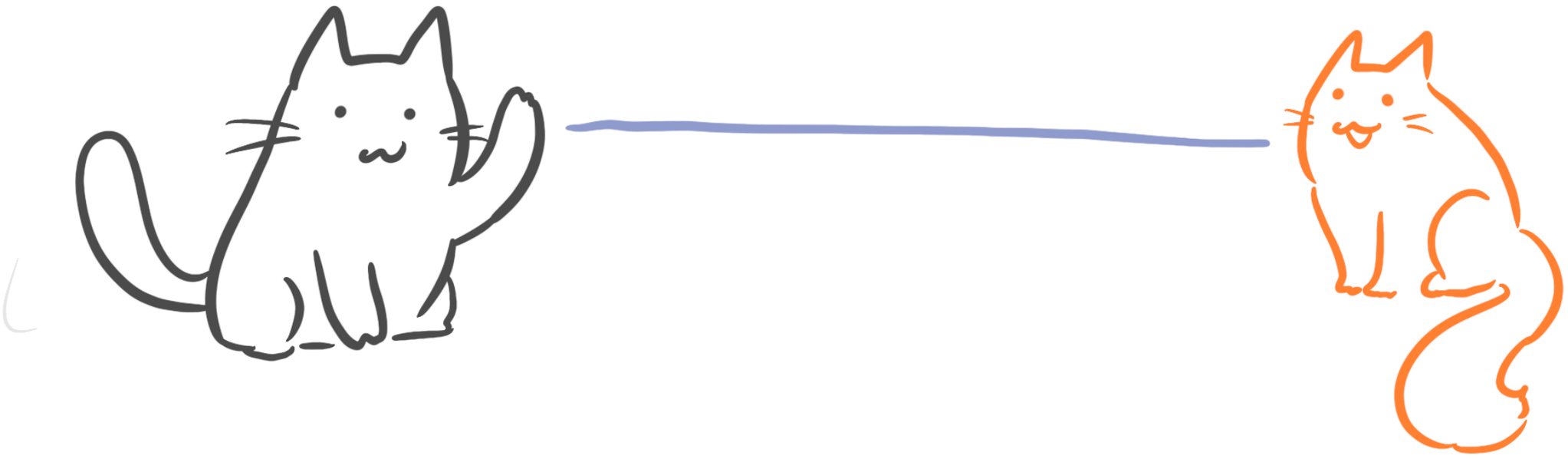
Leader "commits" the new data

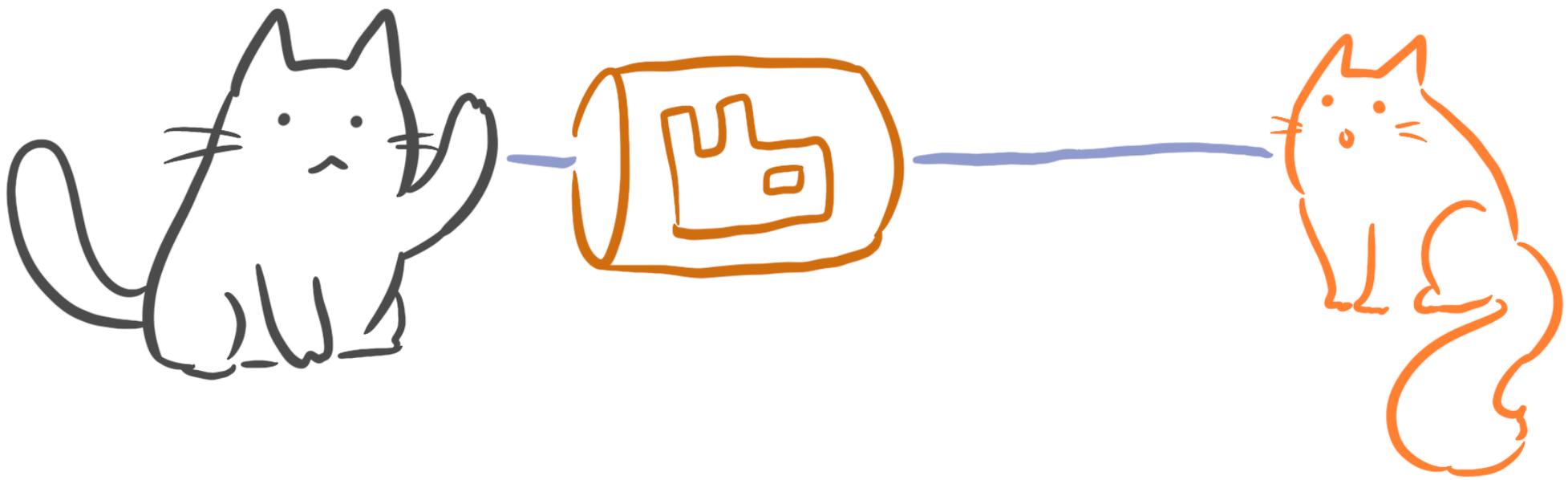
& followers increment their state counter

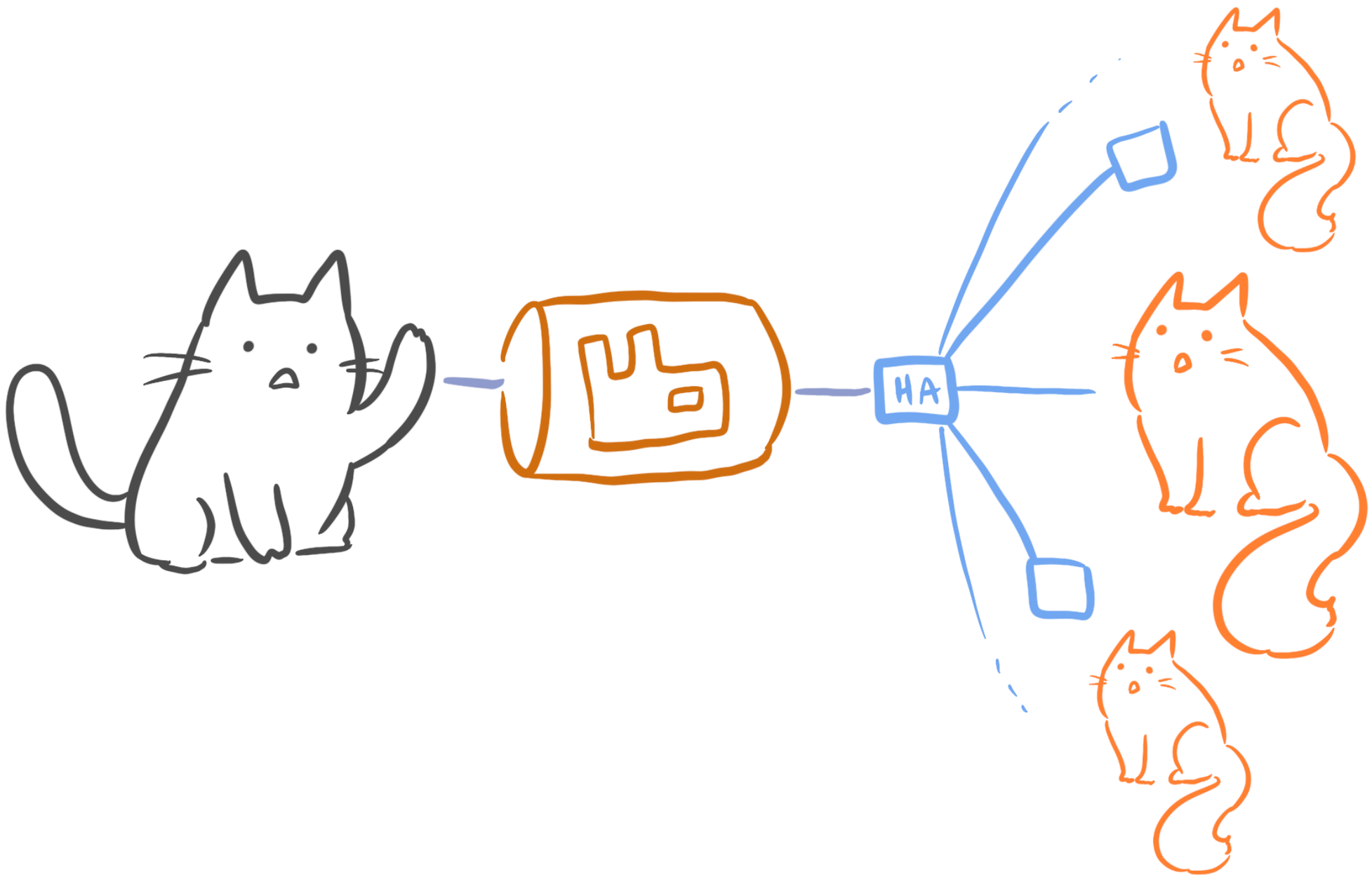
@deniseyu21 with @mt165

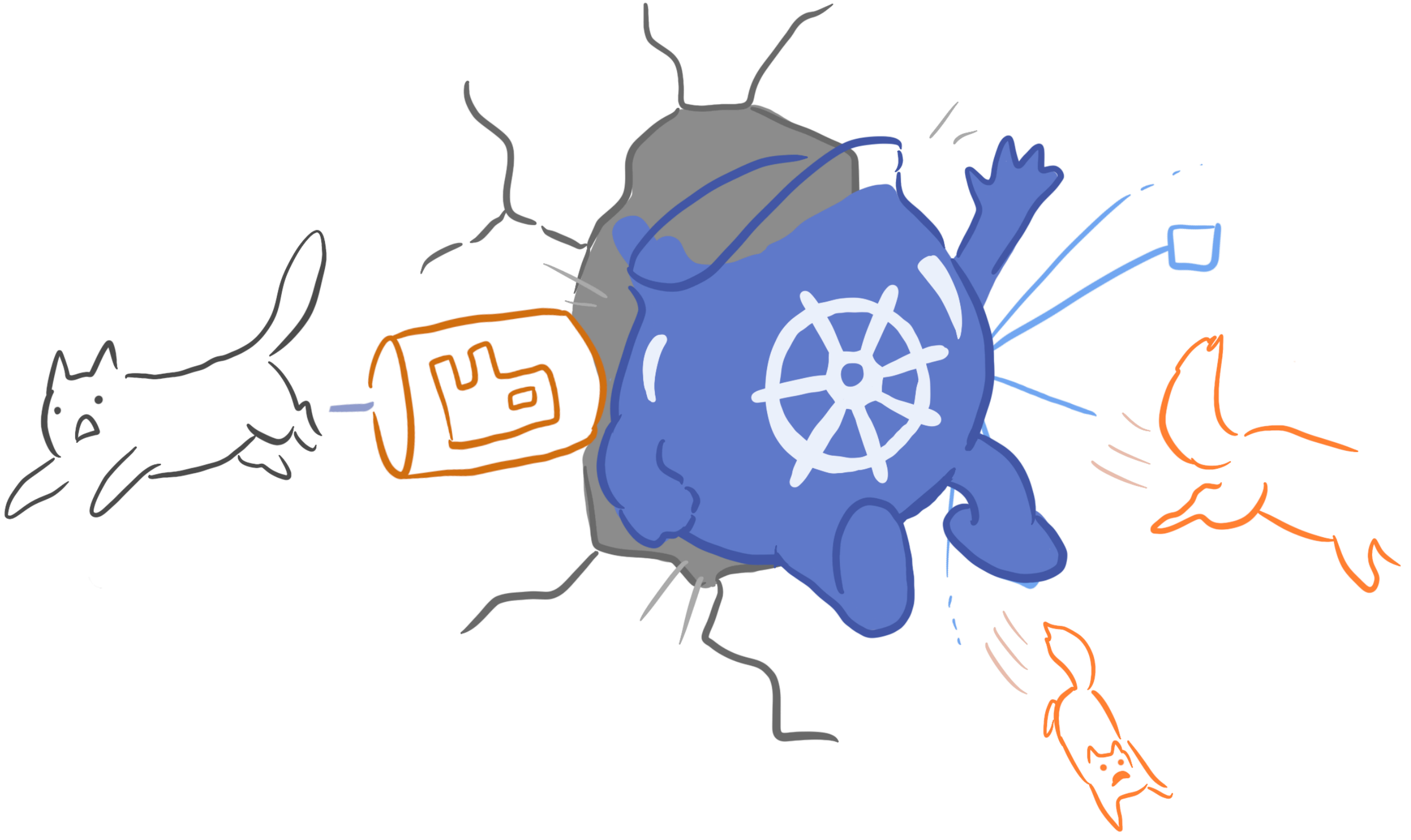
what is even harder than getting machines to agree?

humans






HA

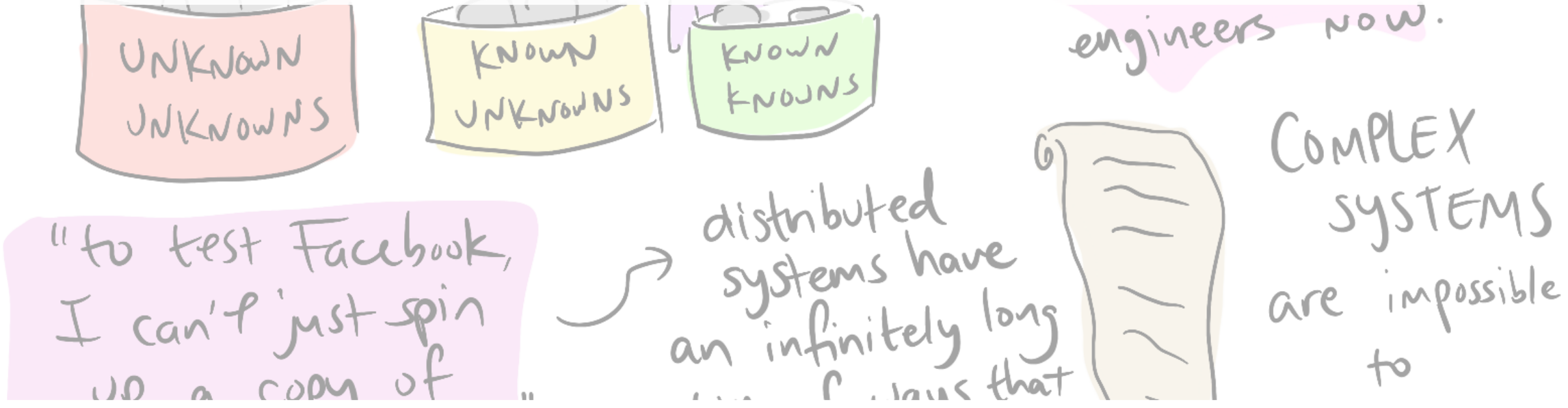


Humans introduce uncertainty

OBSERVABILITY

"Systems are getting more complicated, mental models are getting tougher to build, and we're all distributed systems engineers now."

Charity Majors (@mipsytipsy)



how do we manage growing complexity?

understand where

COGNITIVE COMPLEXITY

comes from

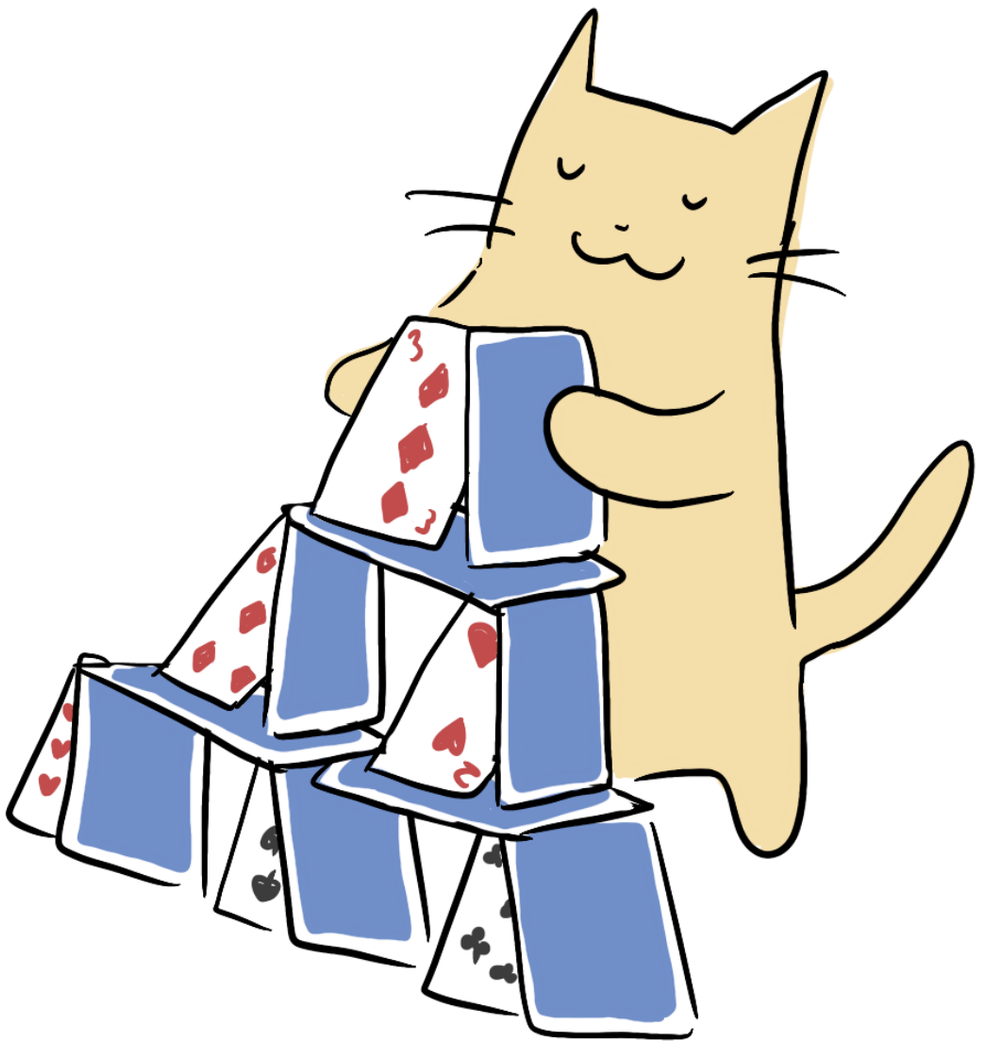
Woods' theorem:

"as the complexity of a system increases, the accuracy of any single agent's own model of that system decreases rapidly."

"Coping With Complexity: the psychology of human behavior in complex systems." Dr. David Woods, 1988.

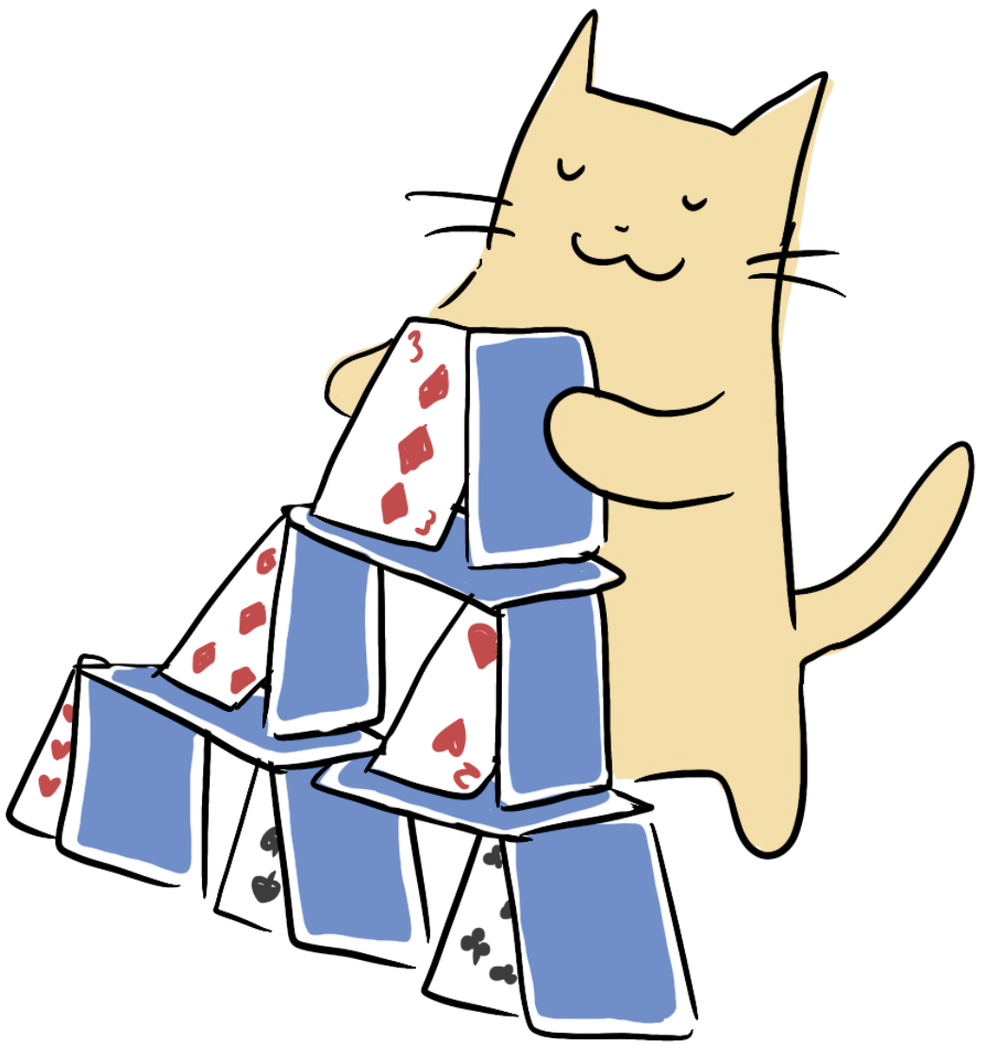
BUILDING MENTAL MODELS

in theory:



BUILDING MENTAL MODELS

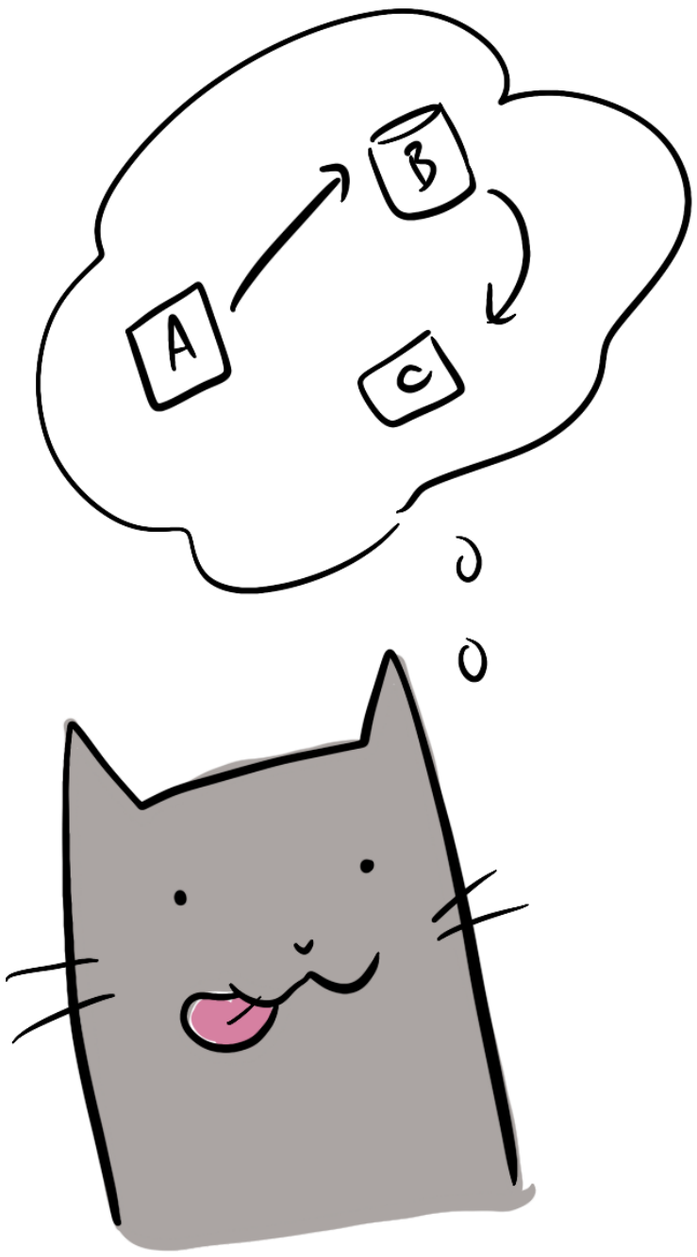
in theory:

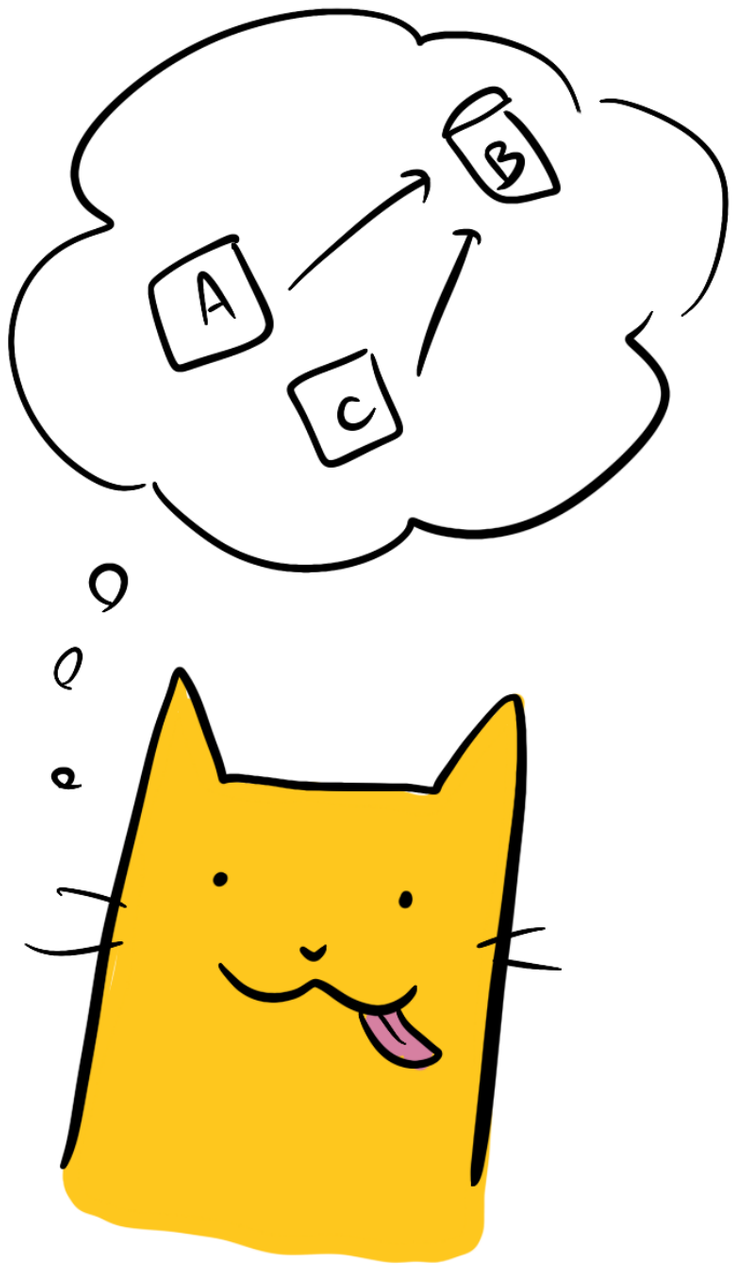


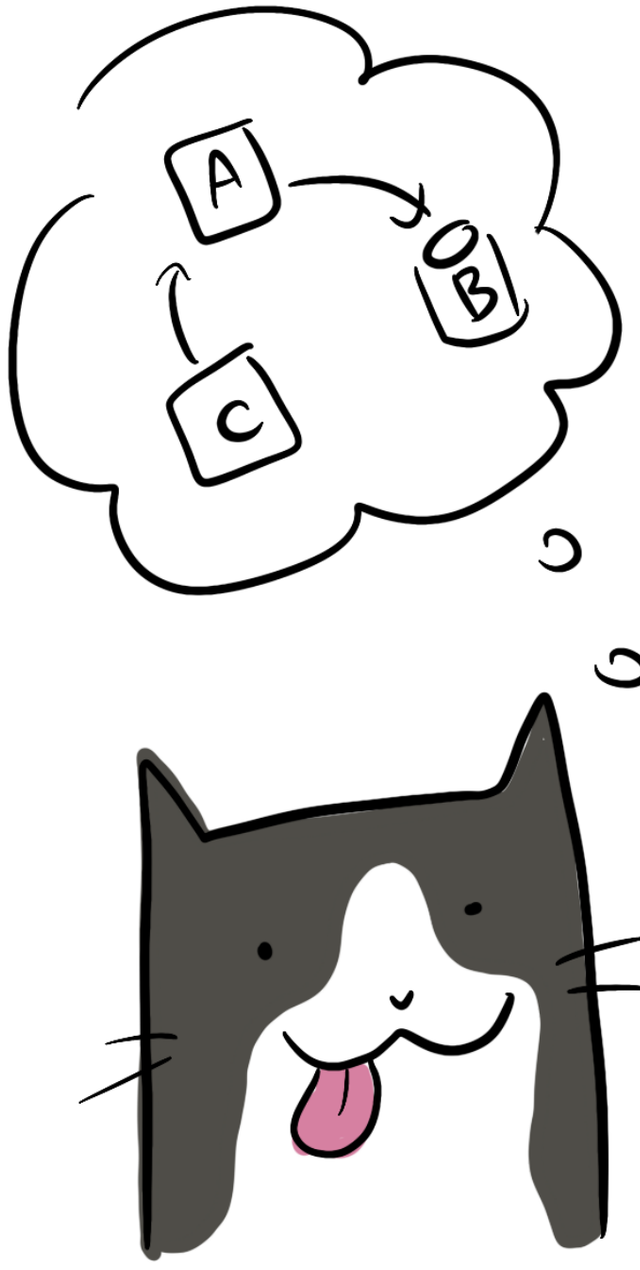
in practice:

@deniseyu21

Mental models are hard to Compare,
which makes them hard to calibrate



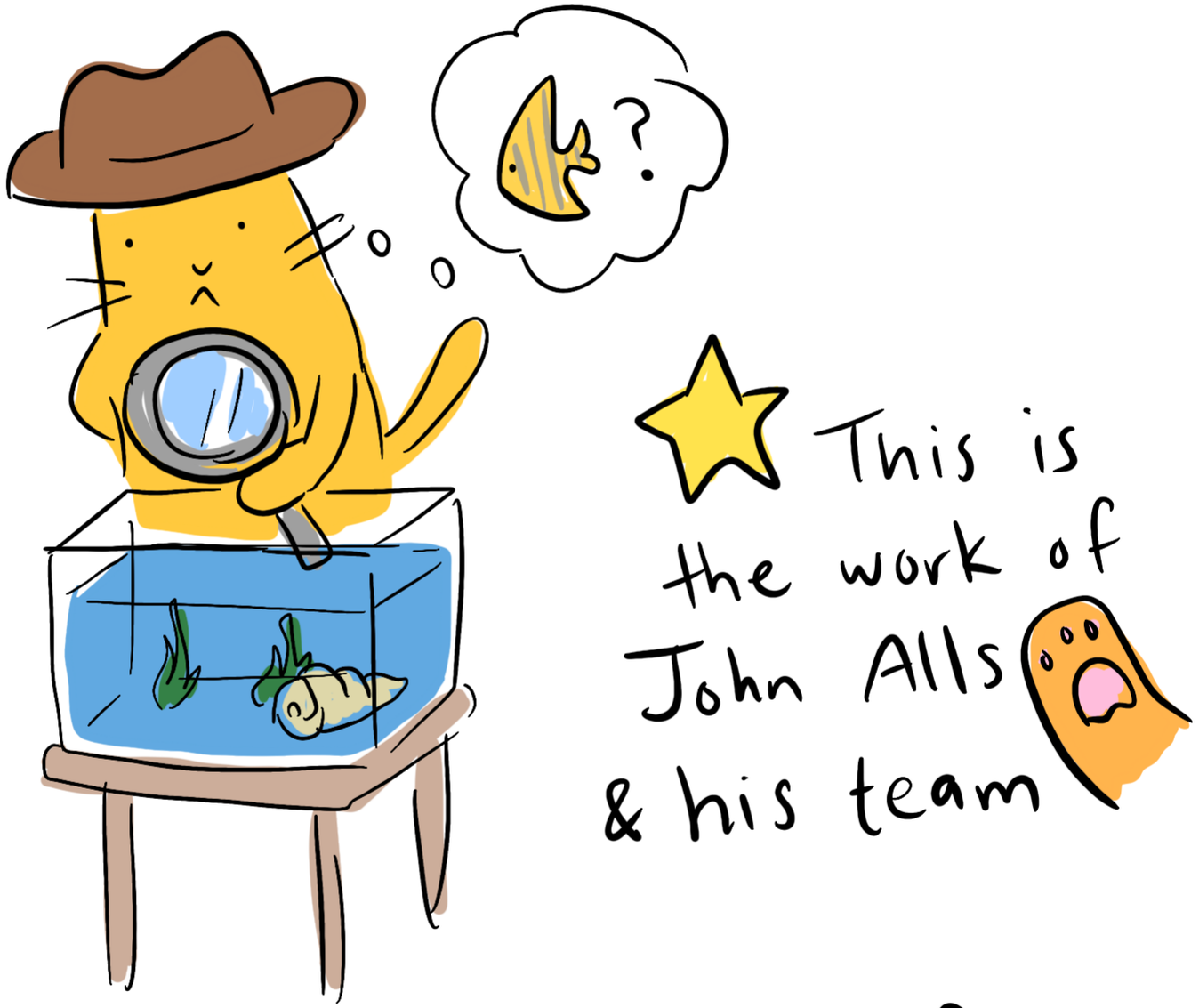




Mental models are hard to Compare, which makes them hard to calibrate

@deniseyu21

INCIDENT ANALYSIS



IS particularly great for mental model calibration

@deniseyu21

Art created for zine.incidentlabs.io

Blameless discussions optimized for learning

@deniseyu21

"Counterfactuals don't belong in postmortems."

Will Gallego
@wcgallego

@deniseyu21

"What seemed reasonable at the time?"

"Running Excellent Retrospectives", by Courtney Eckhardt & Lex Nova, SRECon Americas 2019

Don't accept HUMAN ERROR as the root cause. Dig deeper!

@deniseyu21

Unintuitive design?

- [ ] check if you do not not not not not not wish to receive emails

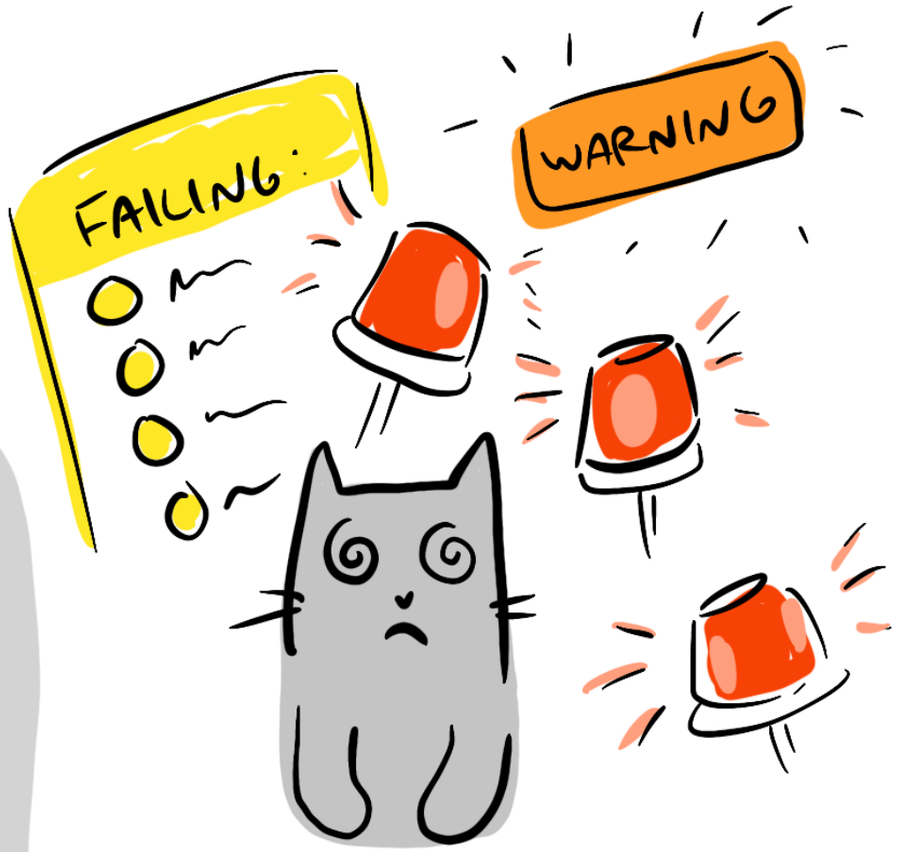
Unintuitive design?

☐ check if you do not not not not not not not wish to receive emails

Alert fatigue?

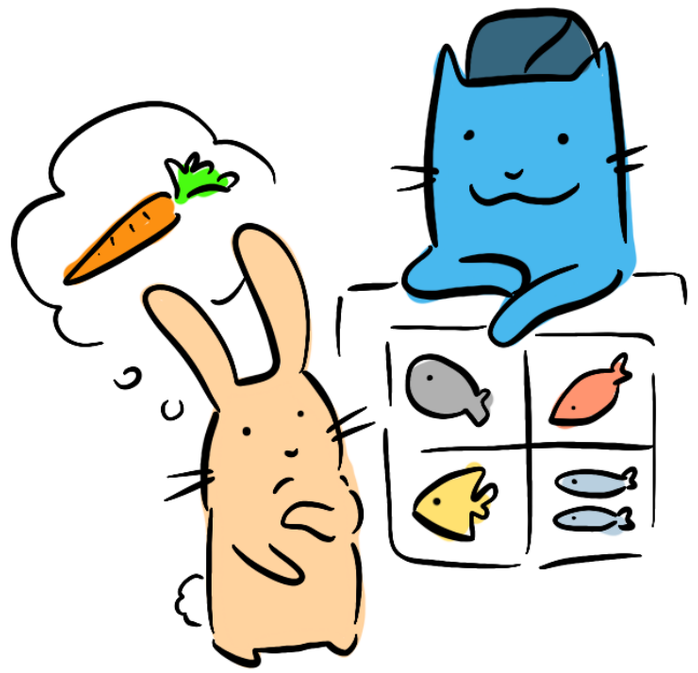
Unintuitive design?

☐ check if you do not not not not not not wish to receive emails

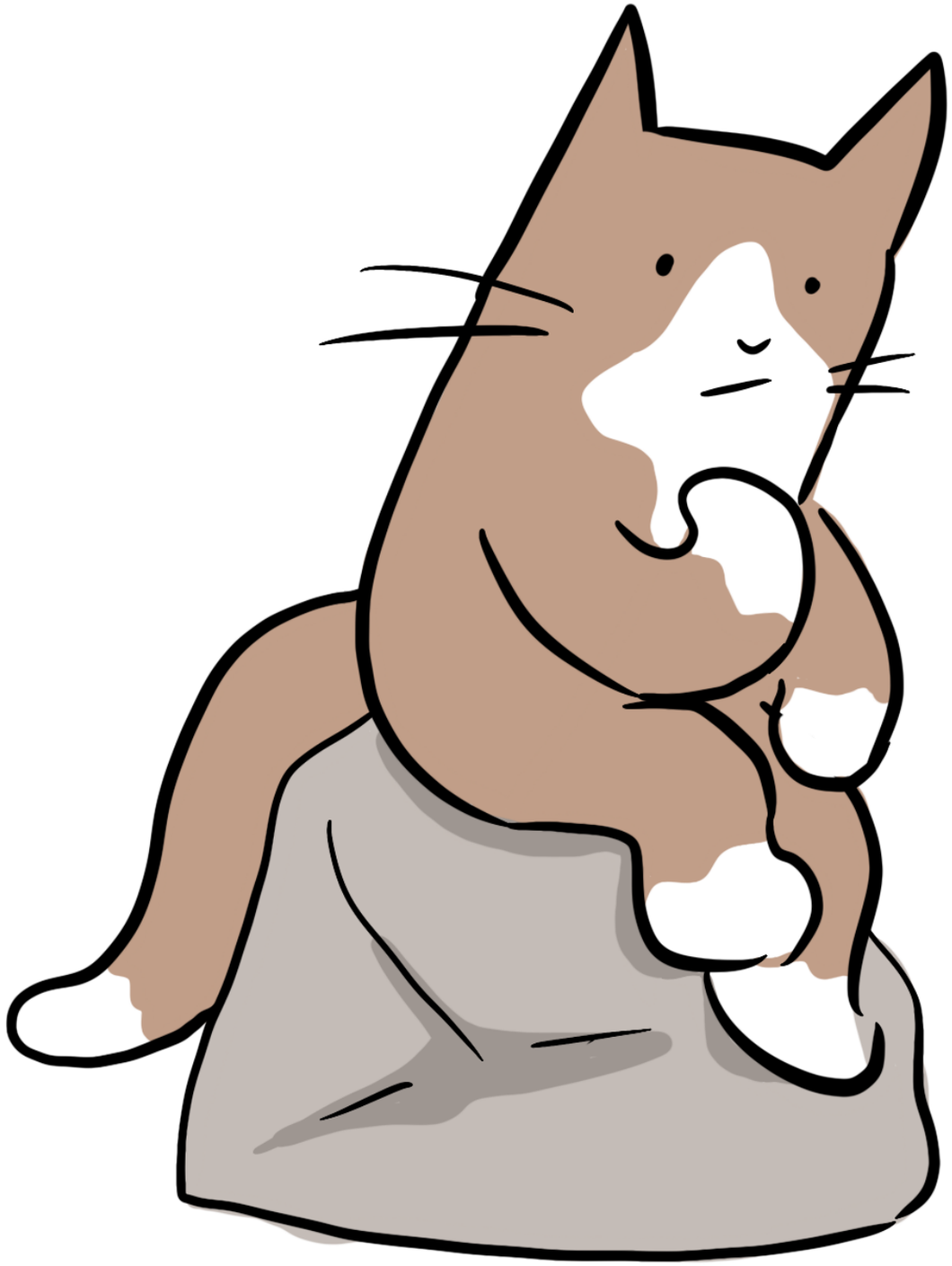


Alert fatigue?

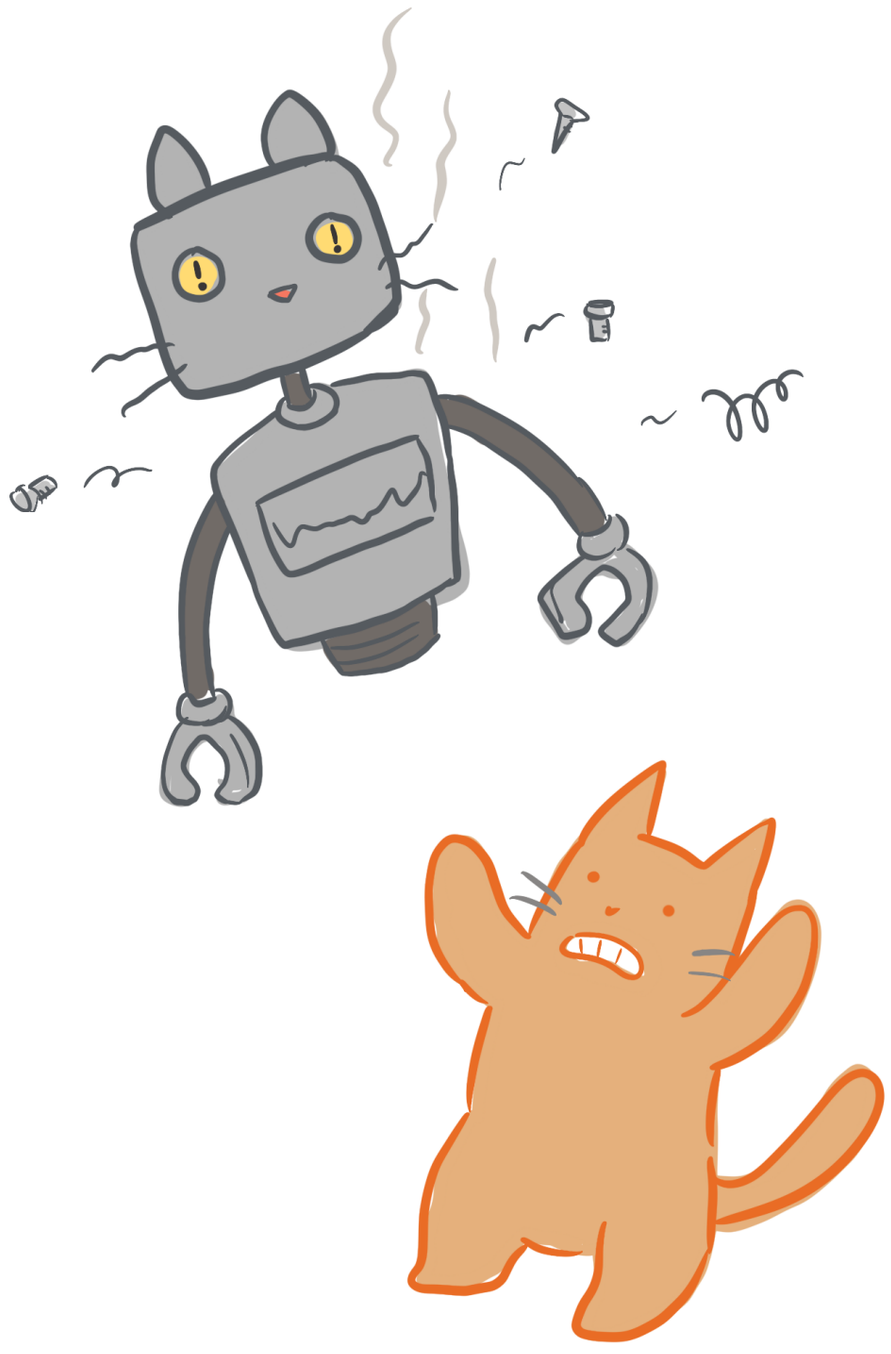
Not understanding the users' assumptions and needs?



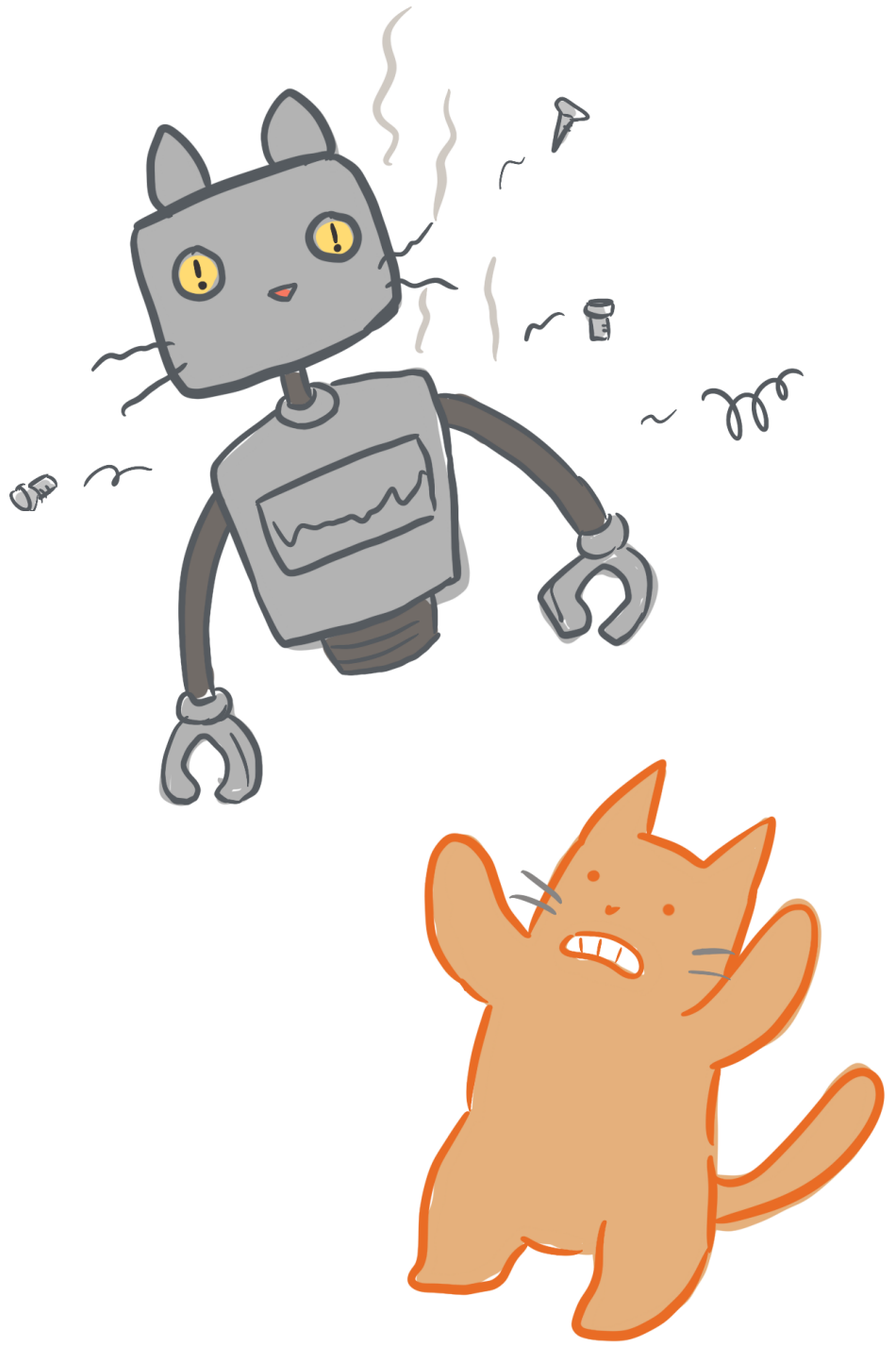
@deniseyu21



“The more we depend on technology and push it to its limits,

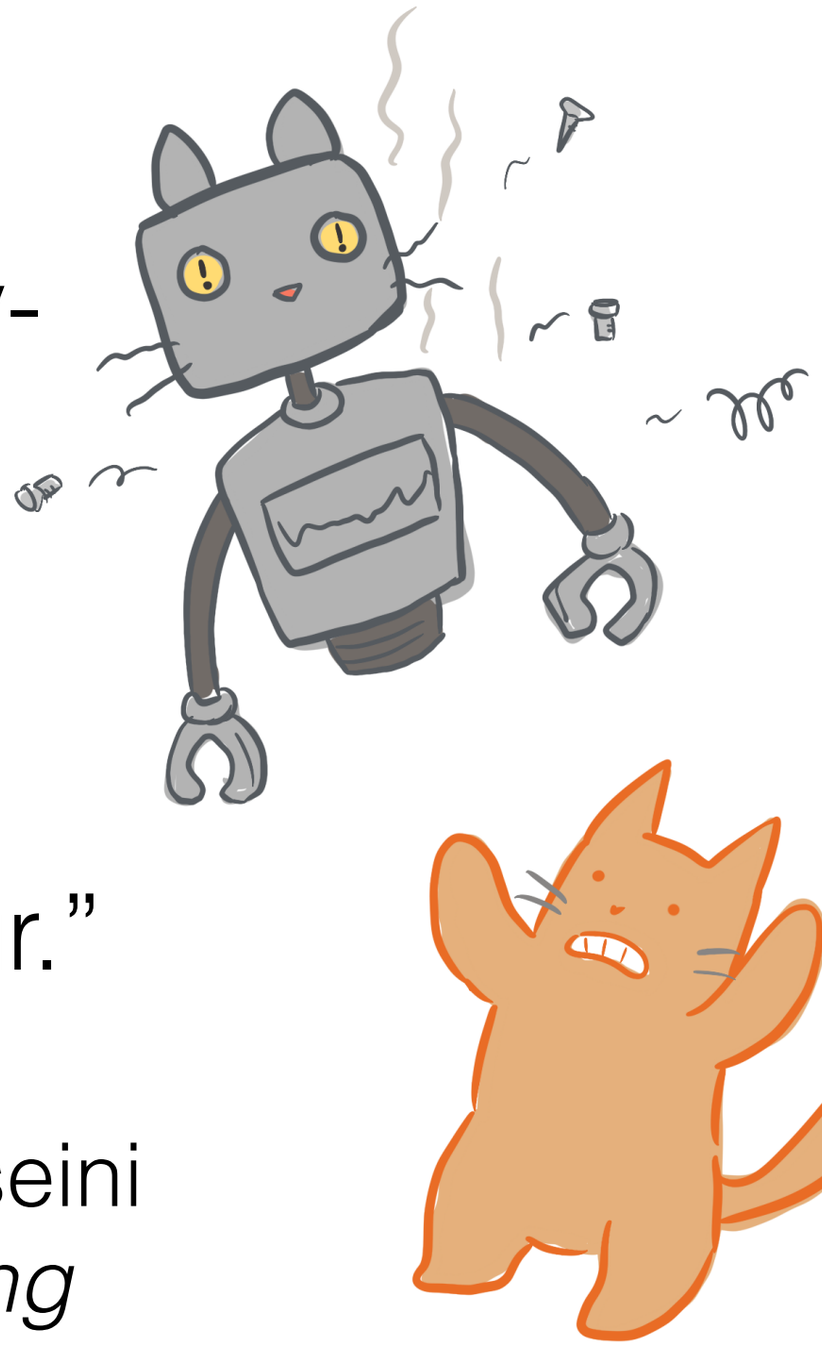


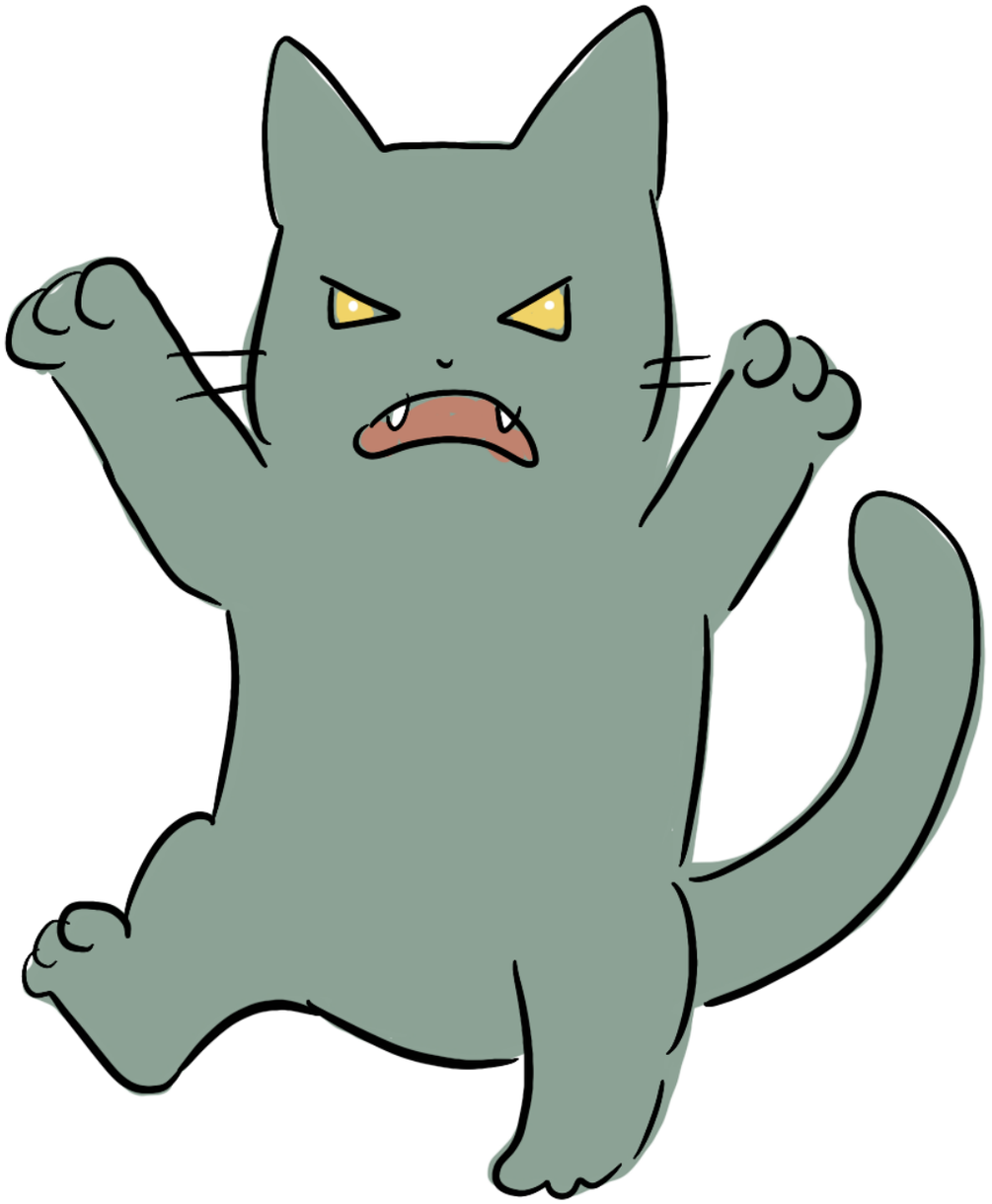
“The more we depend on technology and push it to its limits, the more we need highly skilled, well-trained, well-practiced people to make systems resilient,



“The more we depend on technology and push it to its limits, the more we need highly-skilled, well-trained, well-practiced people to make systems resilient, acting as the last line of defence against the failures that will inevitably occur.”

Baxter, Rooksby, Wong, Khajeh-Hosseini
“The Ironies of Automation… still going strong at 30?” (2012)





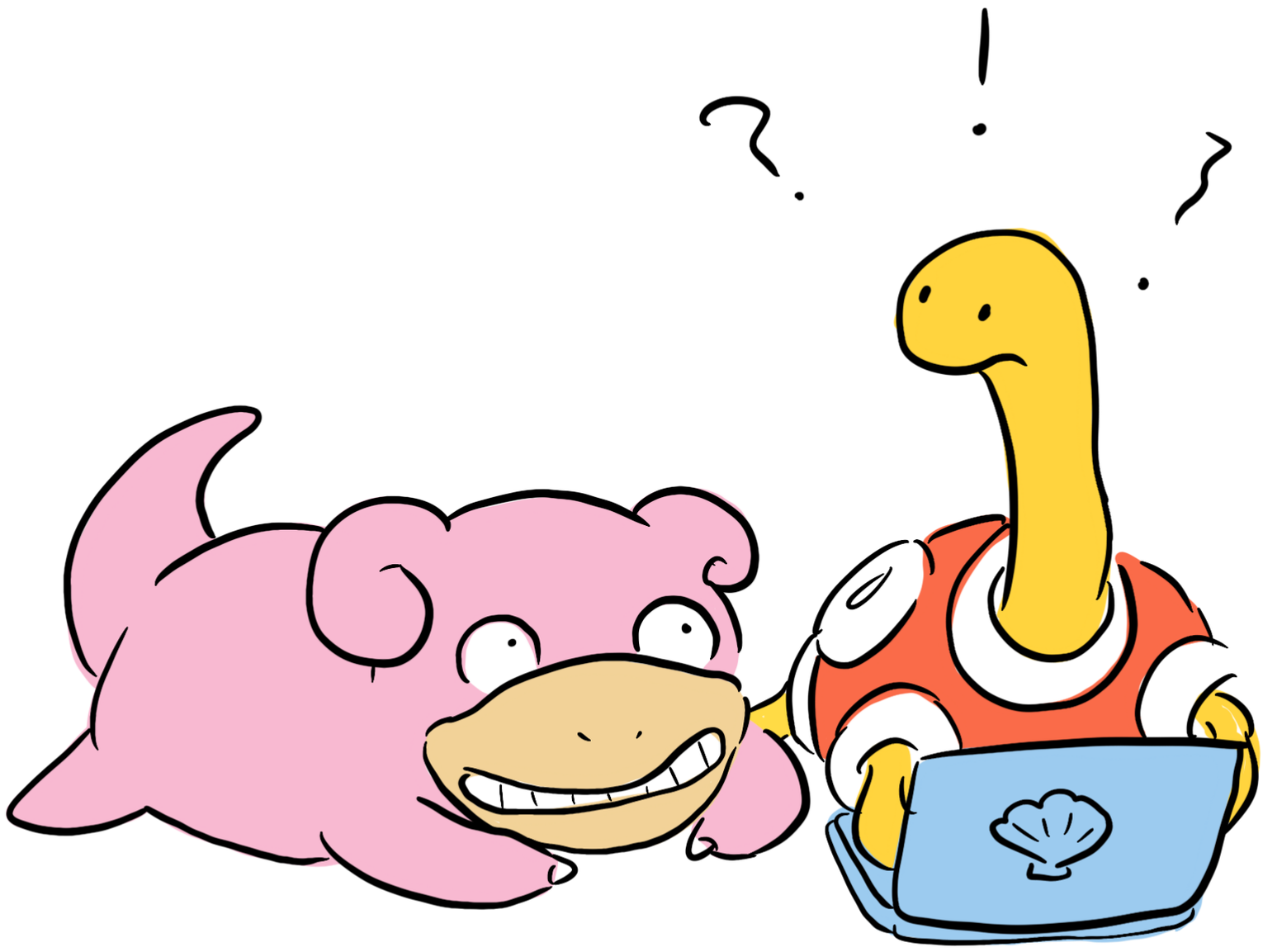
In pursuit of the sociotechnical benefits of distributed, decoupled microservices,

we borrow against

'inherent Complexity

but we can learn

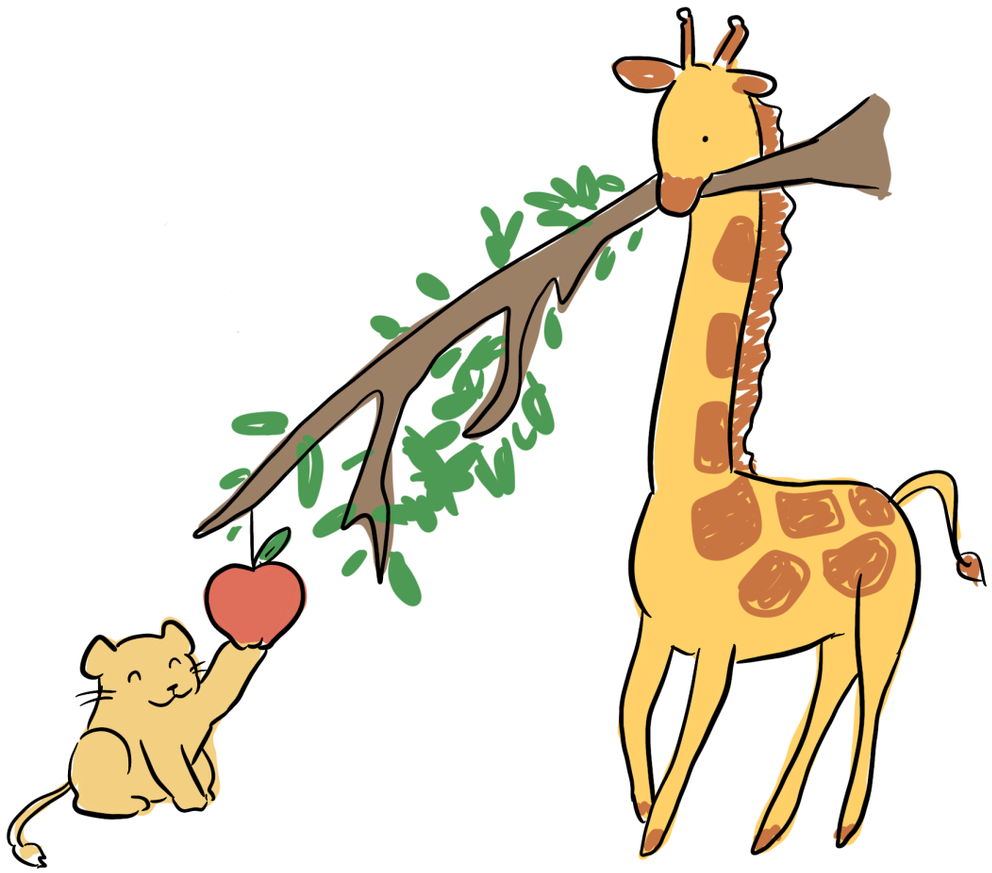
and adapt



Challenge yourself to empathize with each user

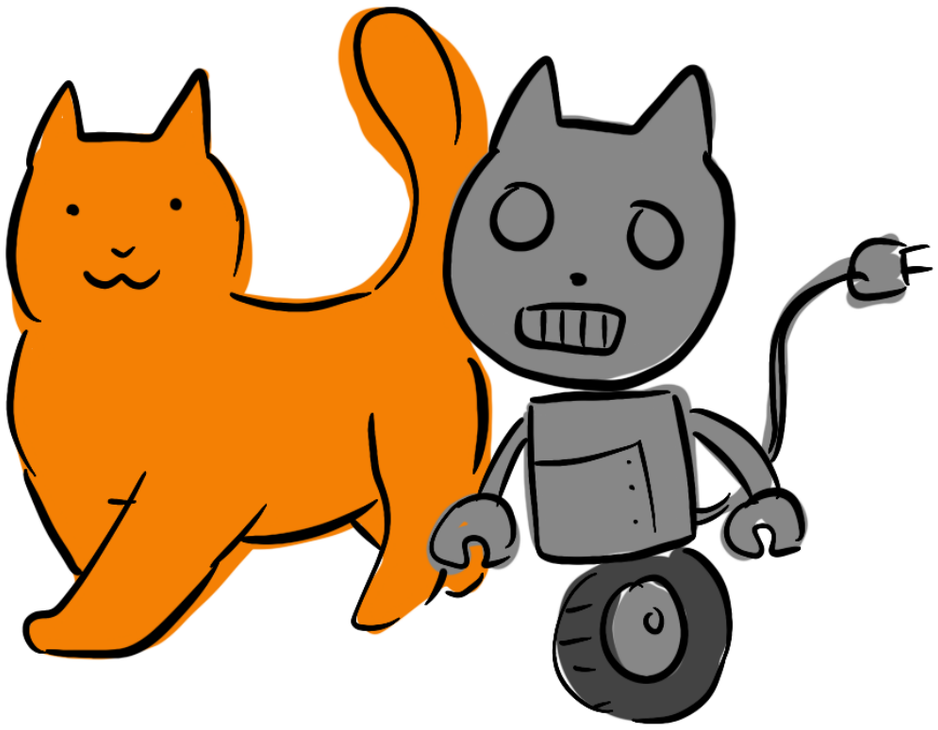
@deniseyu21

Our design choices should make life simpler for the humans operating our systems



@deniseyu21

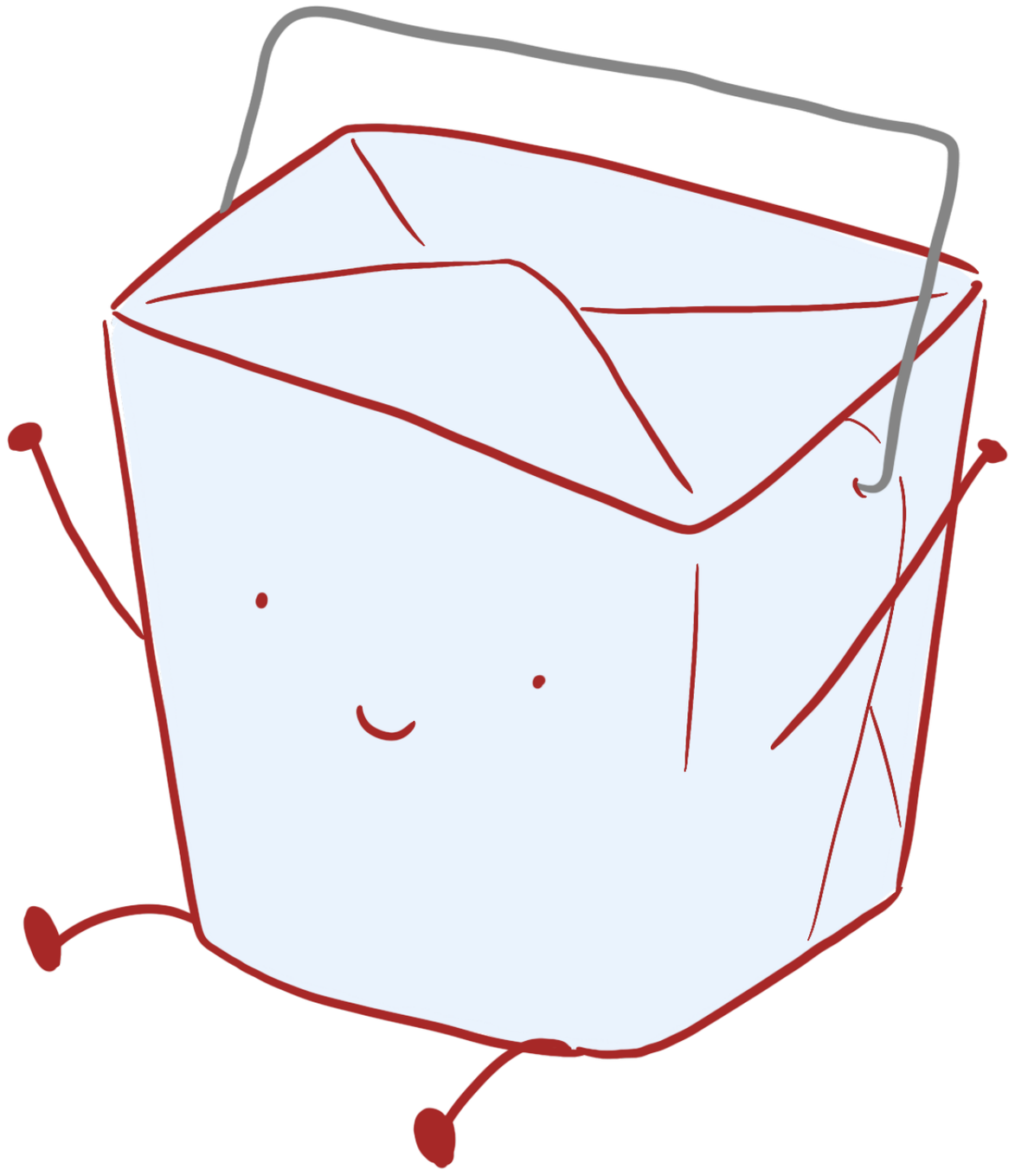
we owe it to our end users &
our teams to understand
& design for the whole system



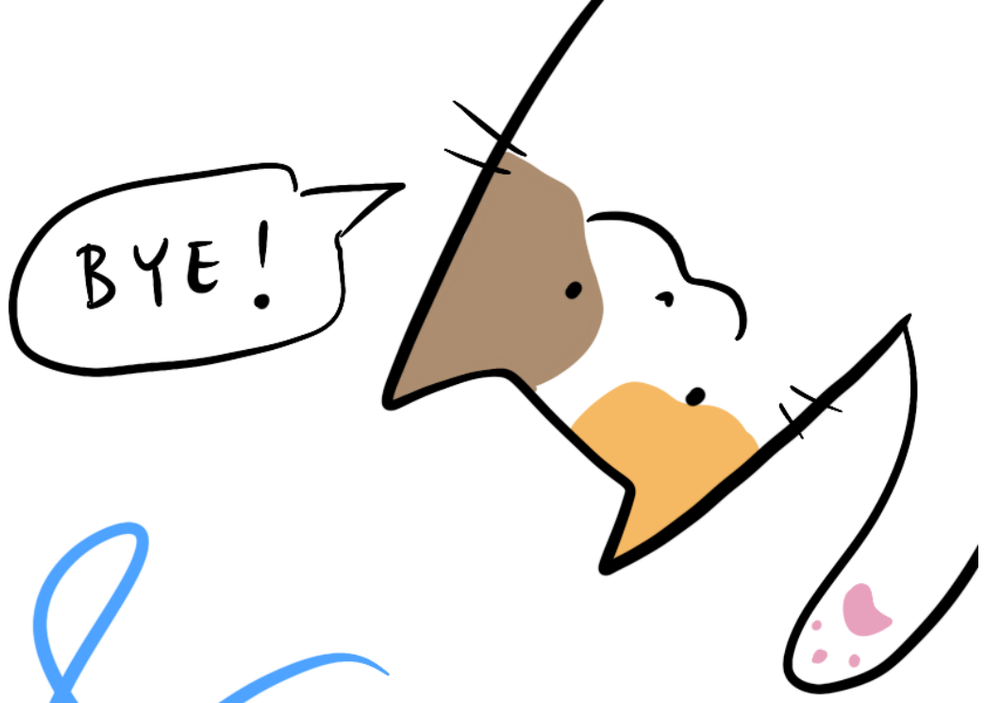
including the fleshy human parts.

@deniseyu21

Thanks
Nicky &
QCon
London!



@deniseyu21



Slides &

references

denise.yu.io/qcon