

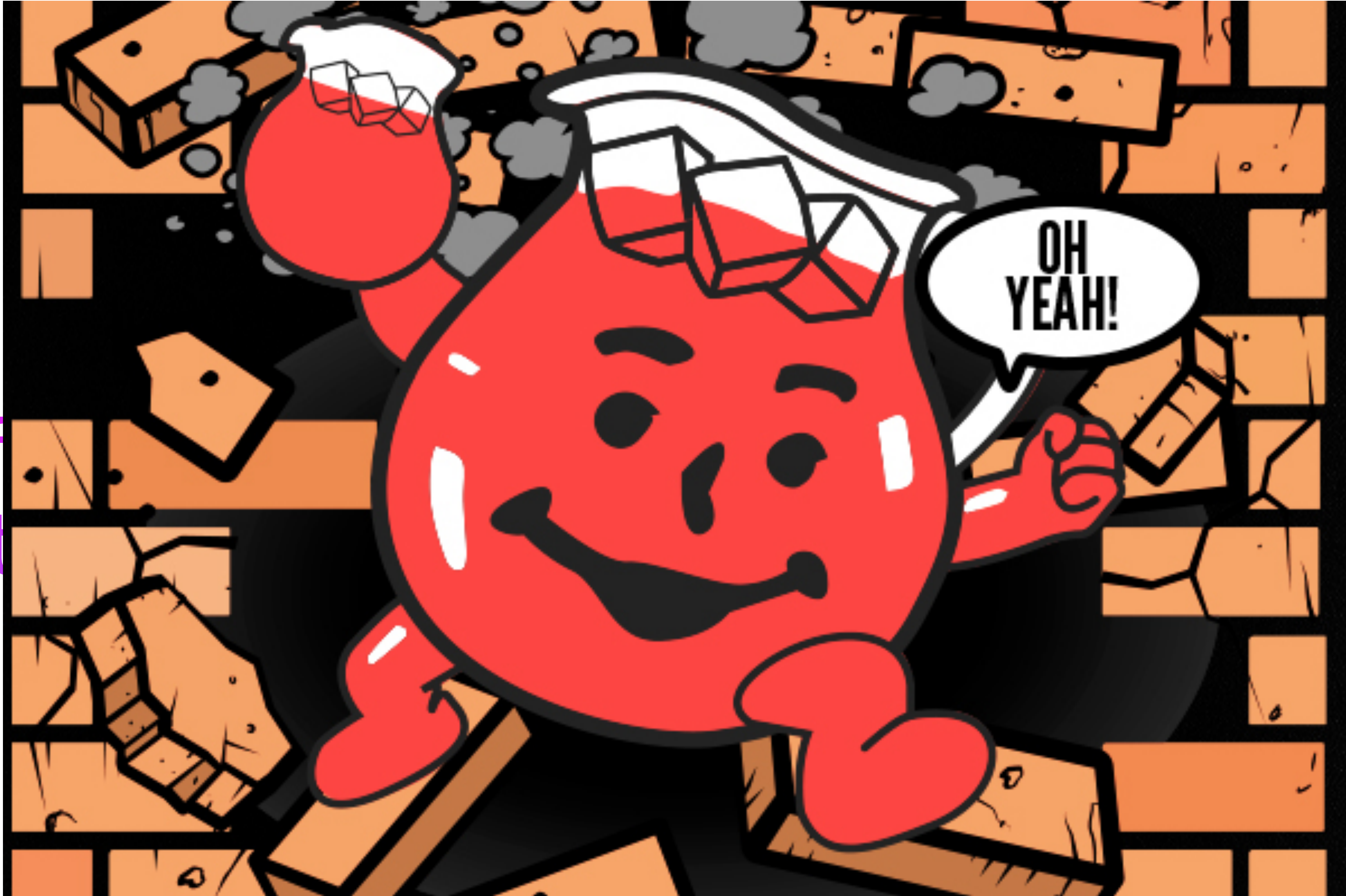


The Jump to Hyperspace

 Light speed, antientropy, & moving past the cloud 

I suppose it is tempting,
if the only tool you have is a **hammer**,
to treat everything as if it were a **nail**

~Abraham Maslow



if
t

r

[...] by **2025, 75% of data** will be processed **outside** the traditional data centre or cloud

~ Gartner (also Dell & IBM)

<https://www.gartner.com/smarterwithgartner/what-edge-computing-means-for-infrastructure-and-operations-leaders>

Brooklyn Zelenka

@expede



Brooklyn Zelenka

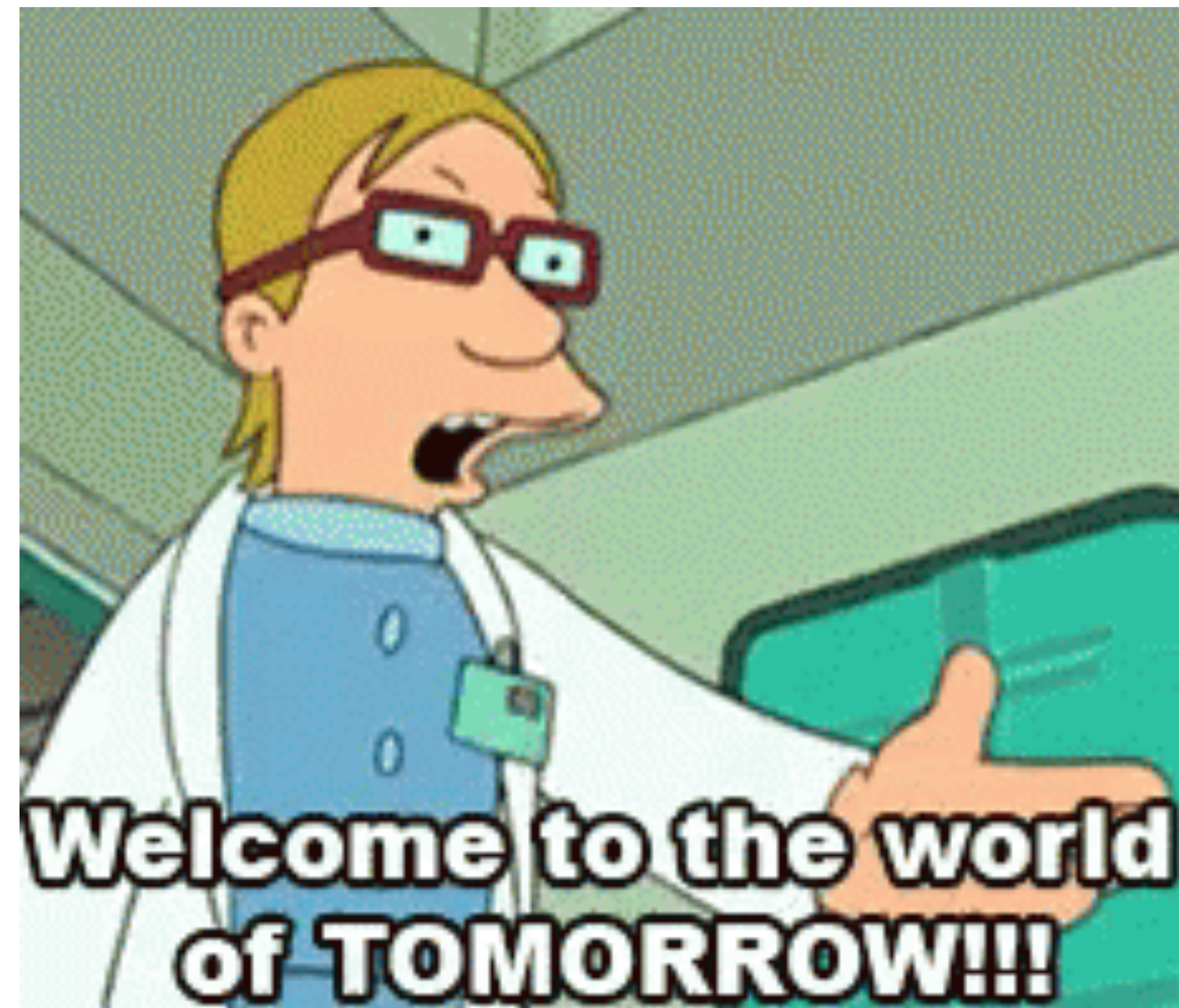
@expede

- CTO at Fission
 - <https://fission.codes>, @FISSIONCodes
 - Infra & browser SDK for "edge apps"
 - Local-first, E2EE/EAR, distributed, passwordless
- Standards: DIF, UCAN, Ethereum, Multiformats, others
- Elixir: Witchcraft, Algae, Exceptional, &c



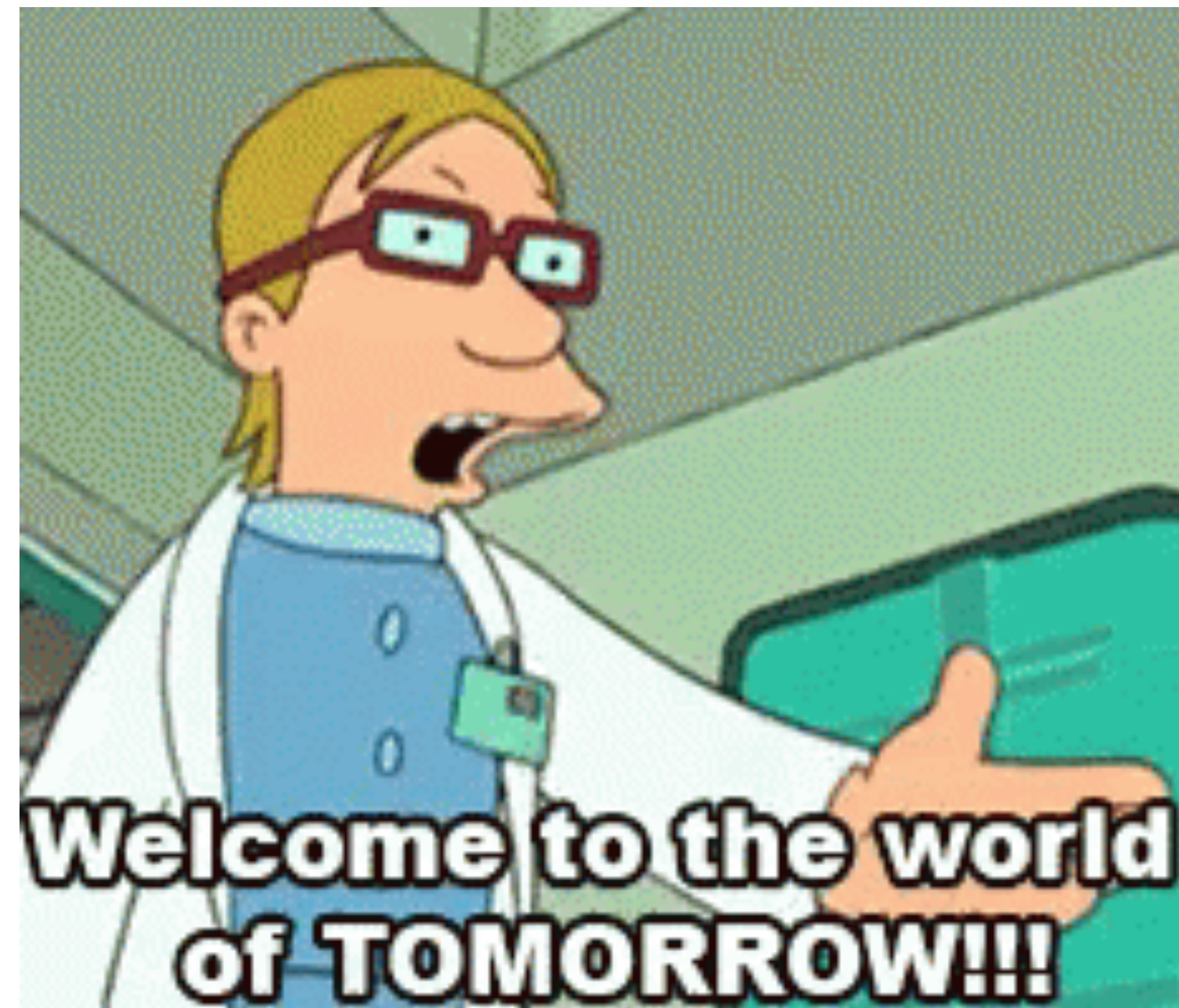
Meta 🌟

- R&D at Fission and others
- Future looking, emerging space
- Edge, web3, dweb
- Go "up a layer" from BEAM to the internet itself



Meta 🌟

- R&D at Fission and others
- Future looking, emerging space
- Edge, web3, dweb
- Go "up a layer" from BEAM to the internet itself



"Historical Reasons"

How We Got Here



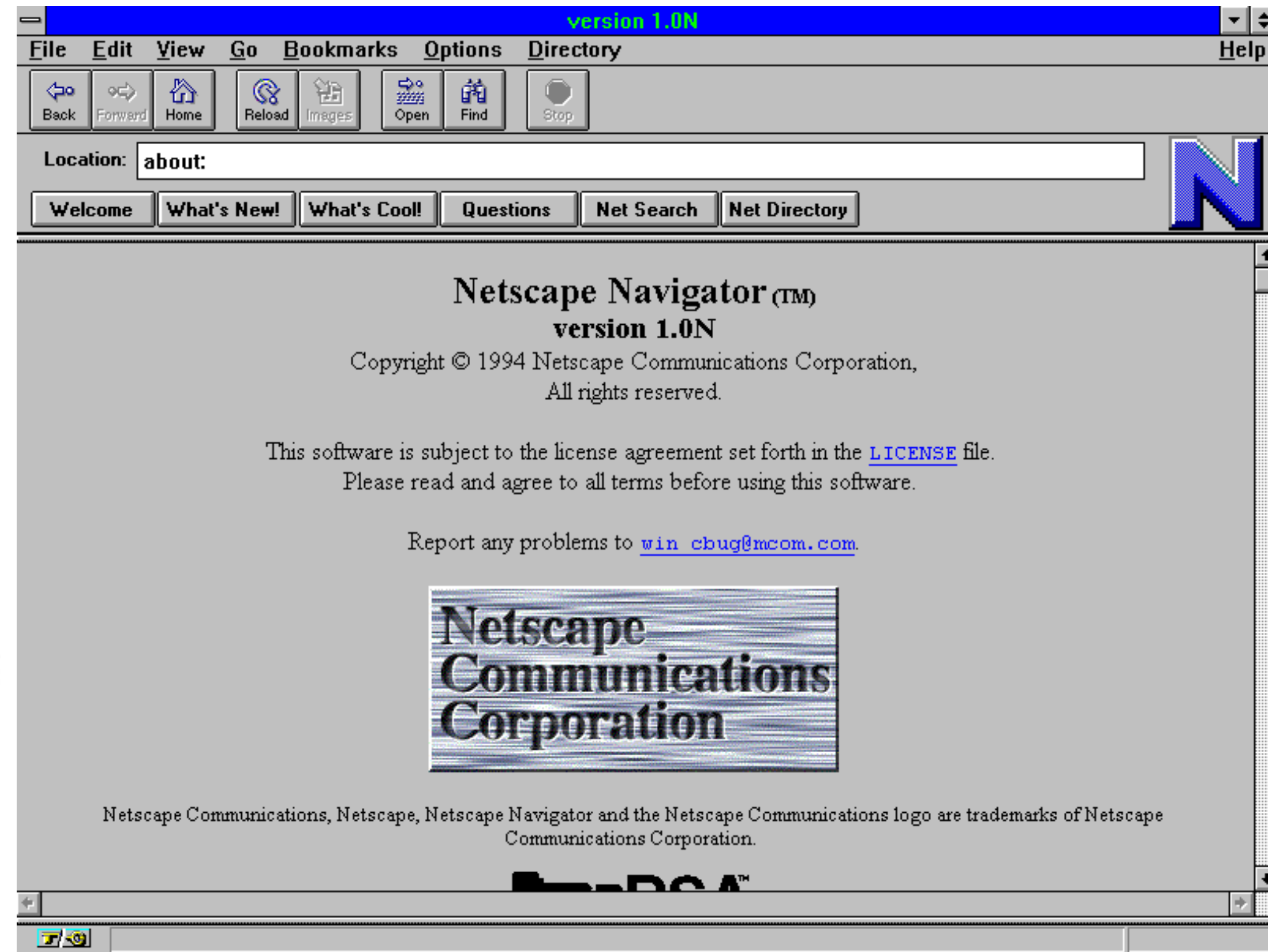
Motivation 🎭

1994 Set the Tone

Motivation 🎭

1994 Set the Tone

W3C®



Network Working Group
Request for Comments: 1105

K. Lougheed
cisco Systems
Y. Rekhter
T.J. Watson Research Center, IBM Corp.
June 1989

A Border Gateway Protocol (BGP)



THE MOST 1001100011011 PHONE.



NOKIA 2110

Some digital cellular phones are more digital than others. For GSM data transmission with your portable computer, the Nokia 2110 is the only phone to offer you almost unlimited compatibility and trouble-free connections with automatic error correction. The Nokia Cellular Data Card connects your Nokia 2110 phone to your PC or Macintosh via the PCMCIA slot. If you don't have this slot, or if you use a small palmtop organizer, what you need is the Nokia Data Card Expander. The Nokia 2110. The most compatible phone.

NOKIA
CONNECTING PEOPLE

Motivation 🎭

The Dream of the 90s is Alive on the Web 🎵

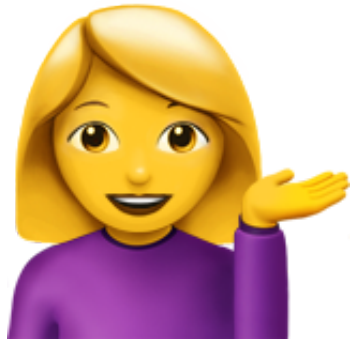
Motivation 🎭

The Dream of the 90s is Alive on the Web 🎵



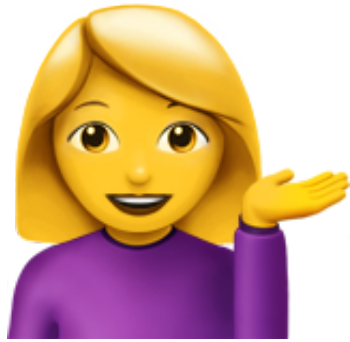
Motivation 🎭

The Dream of the 90s is Alive on the Web 🎵



Motivation 🎭

The Dream of the 90s is Alive on the Web 🎵



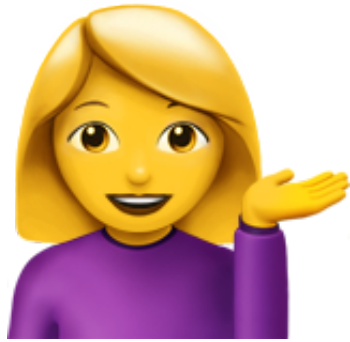
Motivation 🎭

The Dream of the 90s is Alive on the Web 🎵



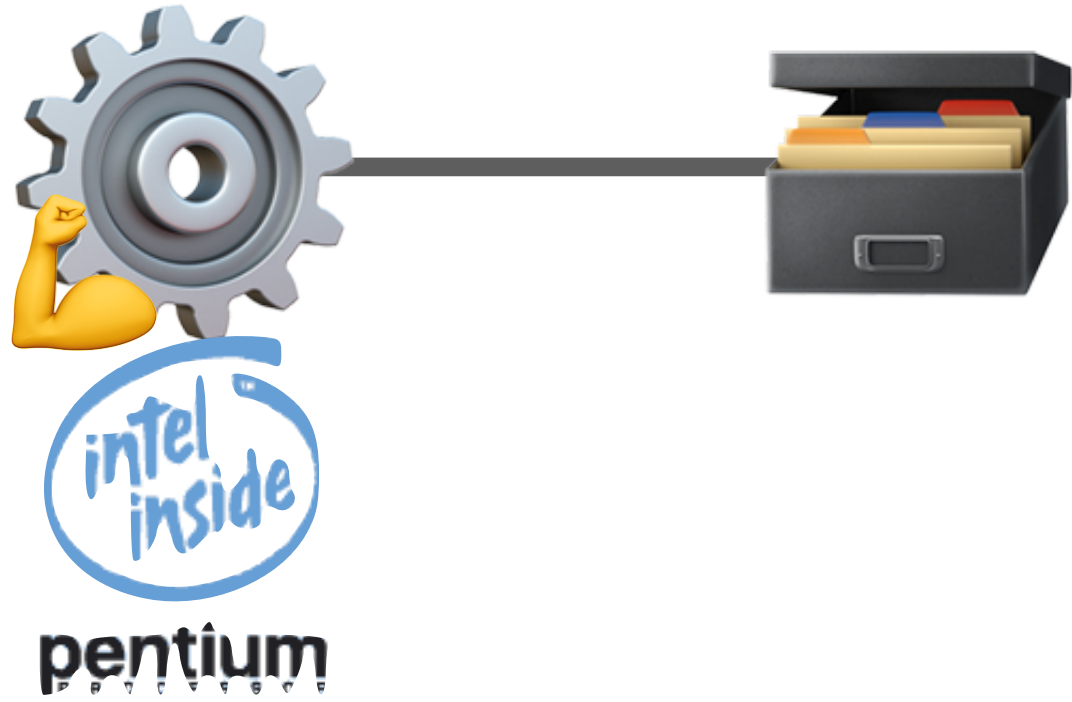
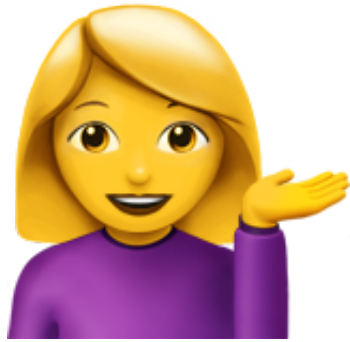
Motivation 🎭

The Dream of the 90s is Alive on the Web 🎵



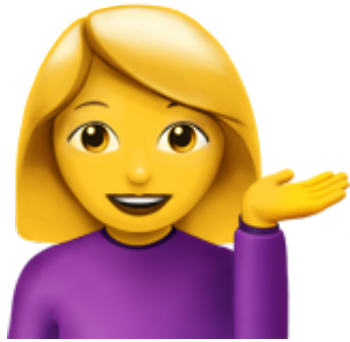
Motivation 🎭

The Dream of the 90s is Alive on the Web 🎵



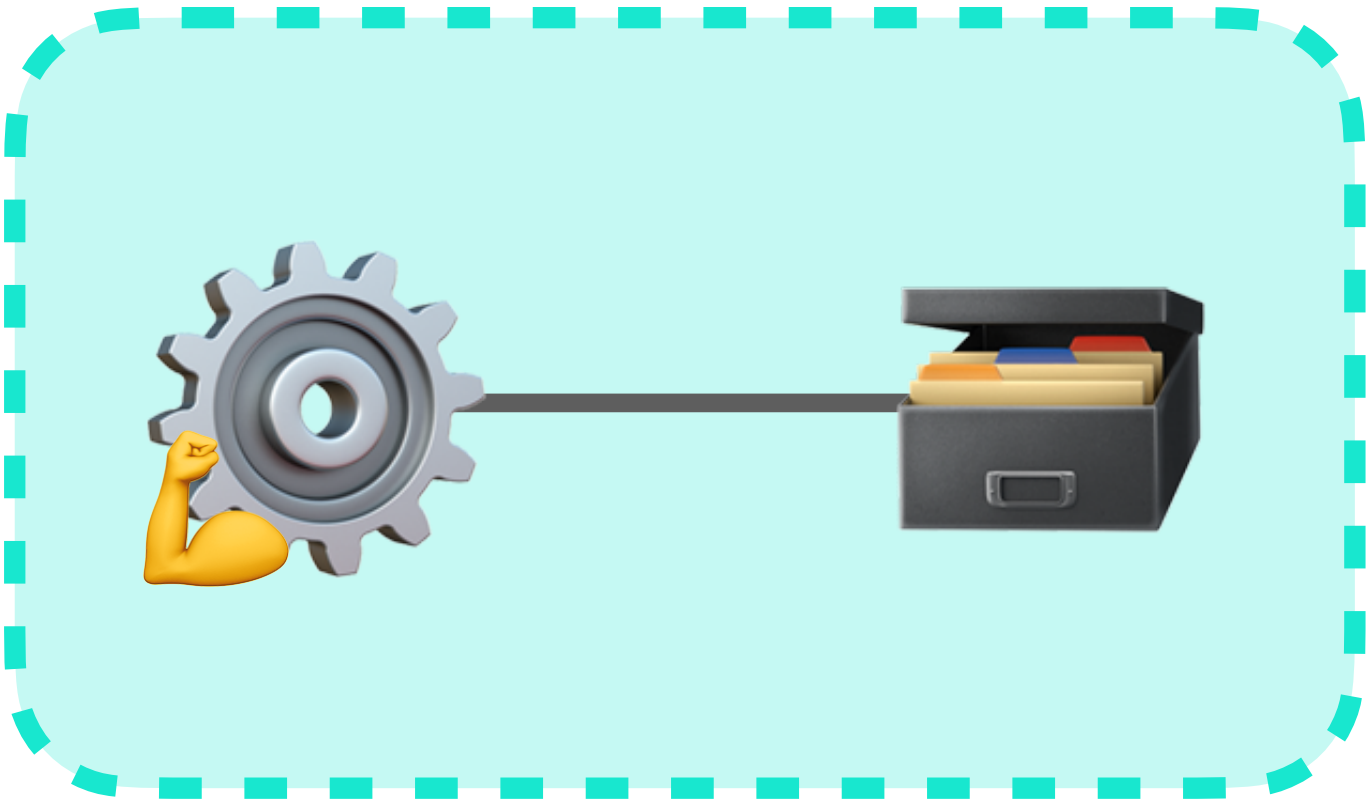
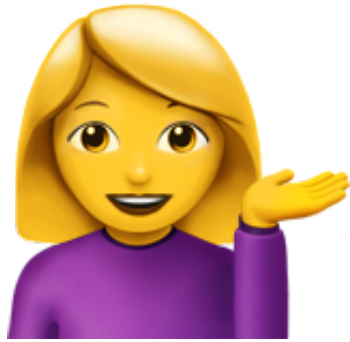
Motivation 🎭

The Dream of the 90s is Alive on the Web 🎵



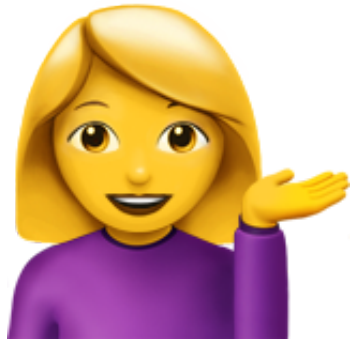
Motivation 🎭

The Dream of the 90s is Alive on the Web 🎵



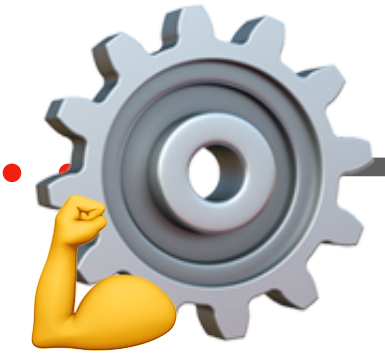
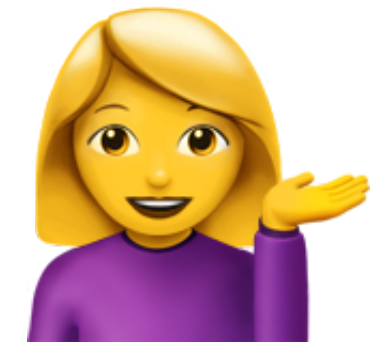
Motivation 🎭

The Dream of the 90s is Alive on the Web 🎵



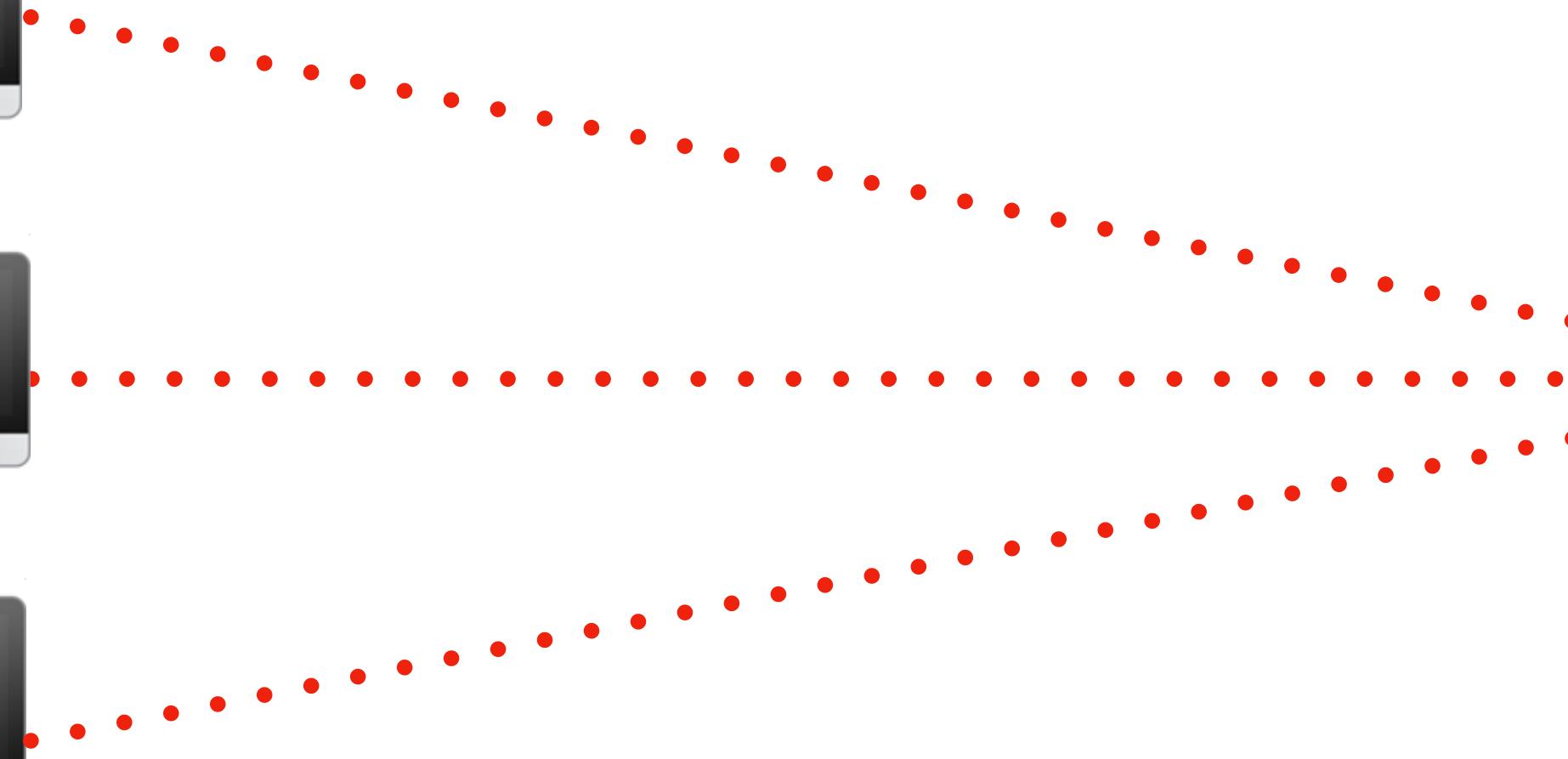
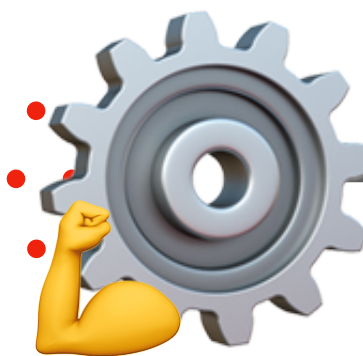
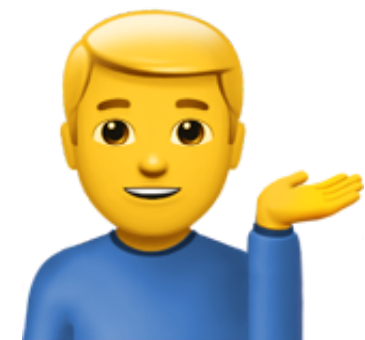
Motivation 🎭

The Dream of the 90s is Alive on the Web 🎵



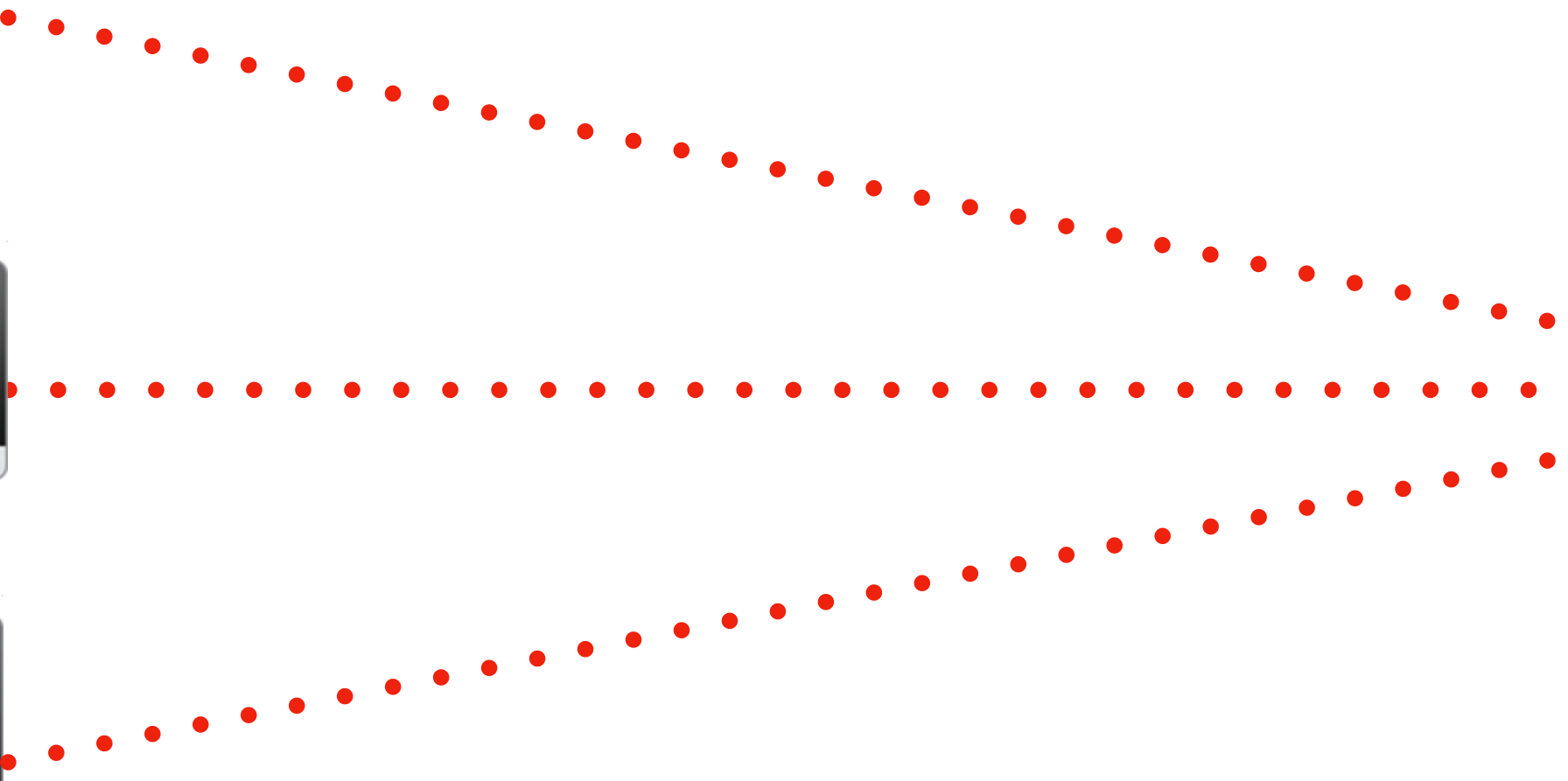
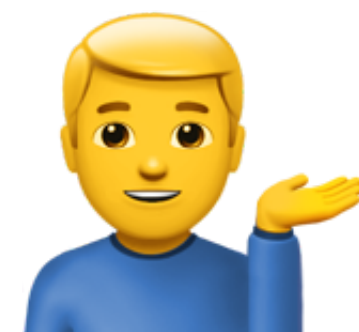
Motivation 🎭

The Dream of the 90s is Alive on the Web 🎵



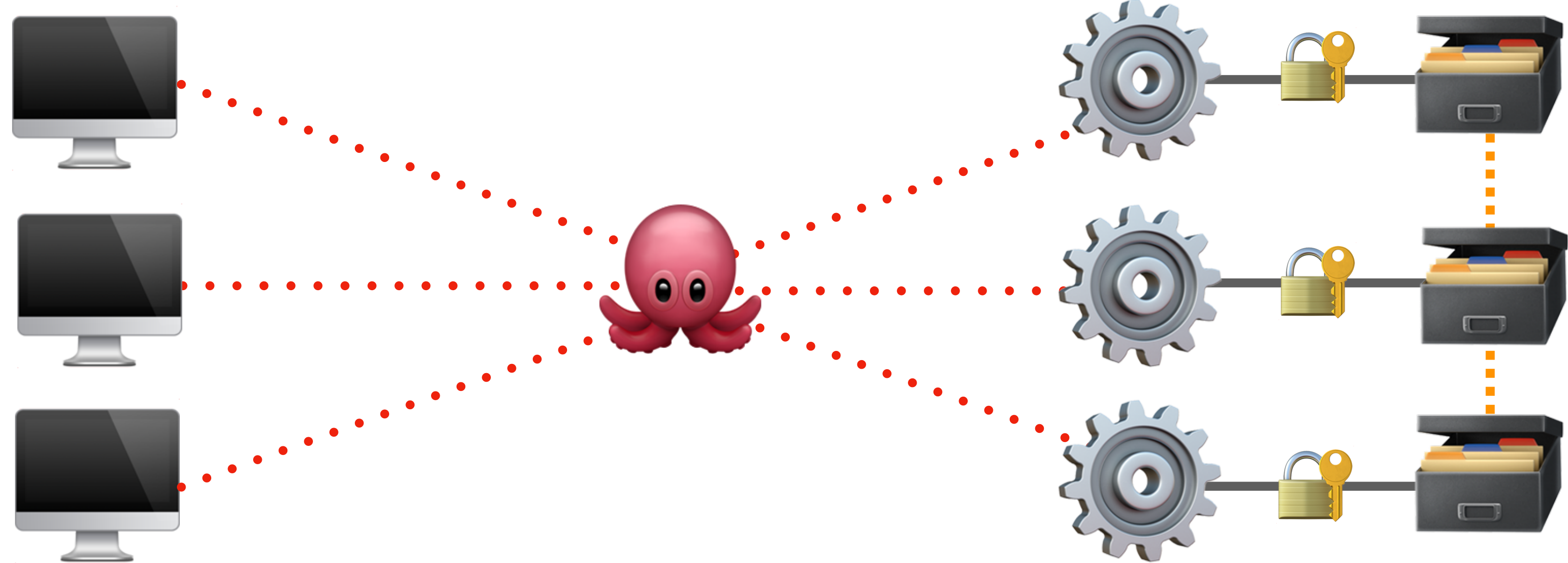
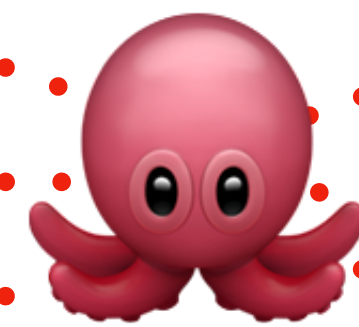
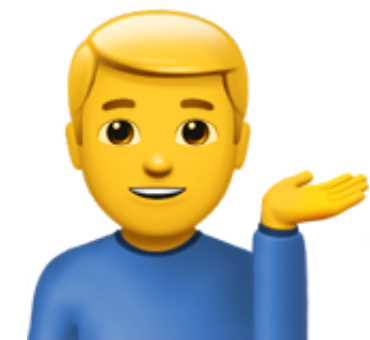
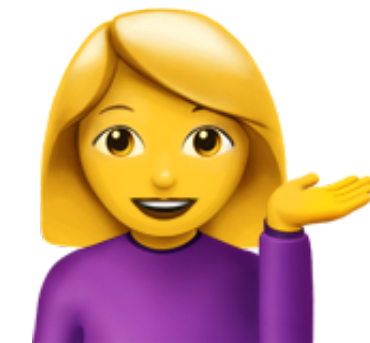
Motivation 🎭

The Dream of the 90s is Alive on the Web 🎵



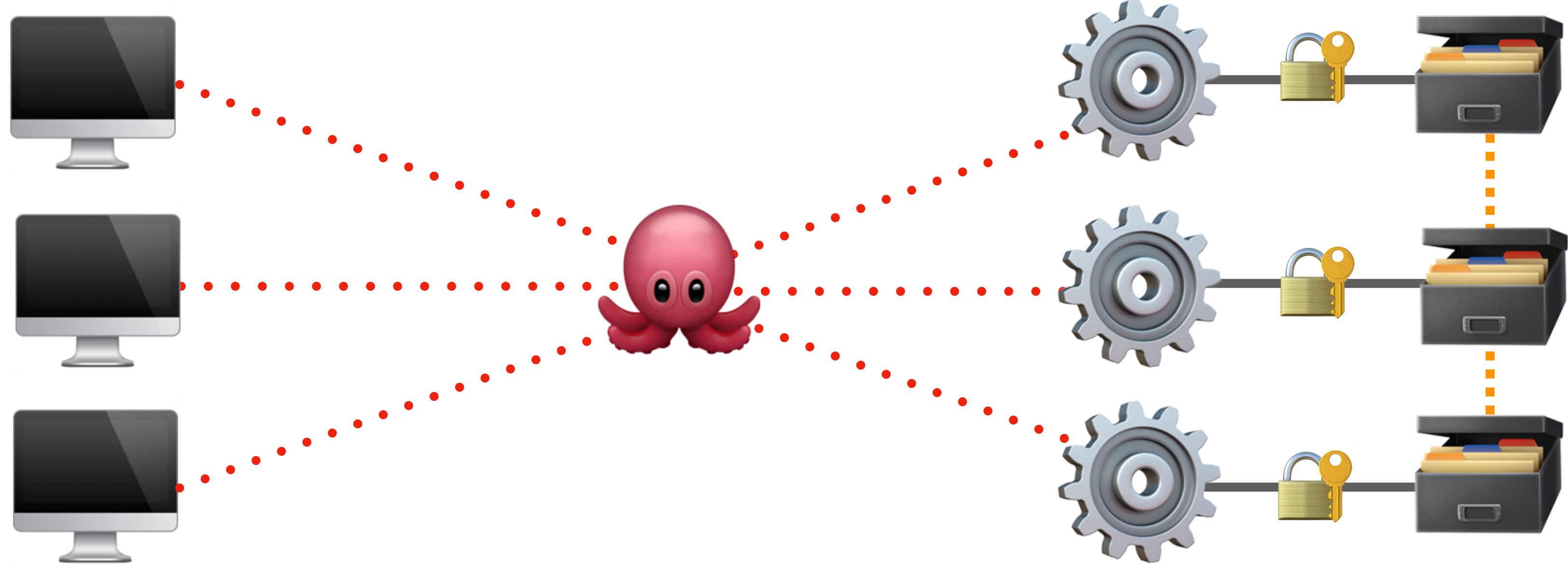
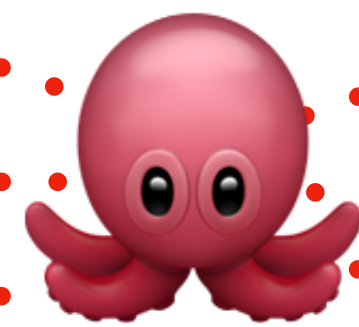
Motivation 🎭

Scaling Up



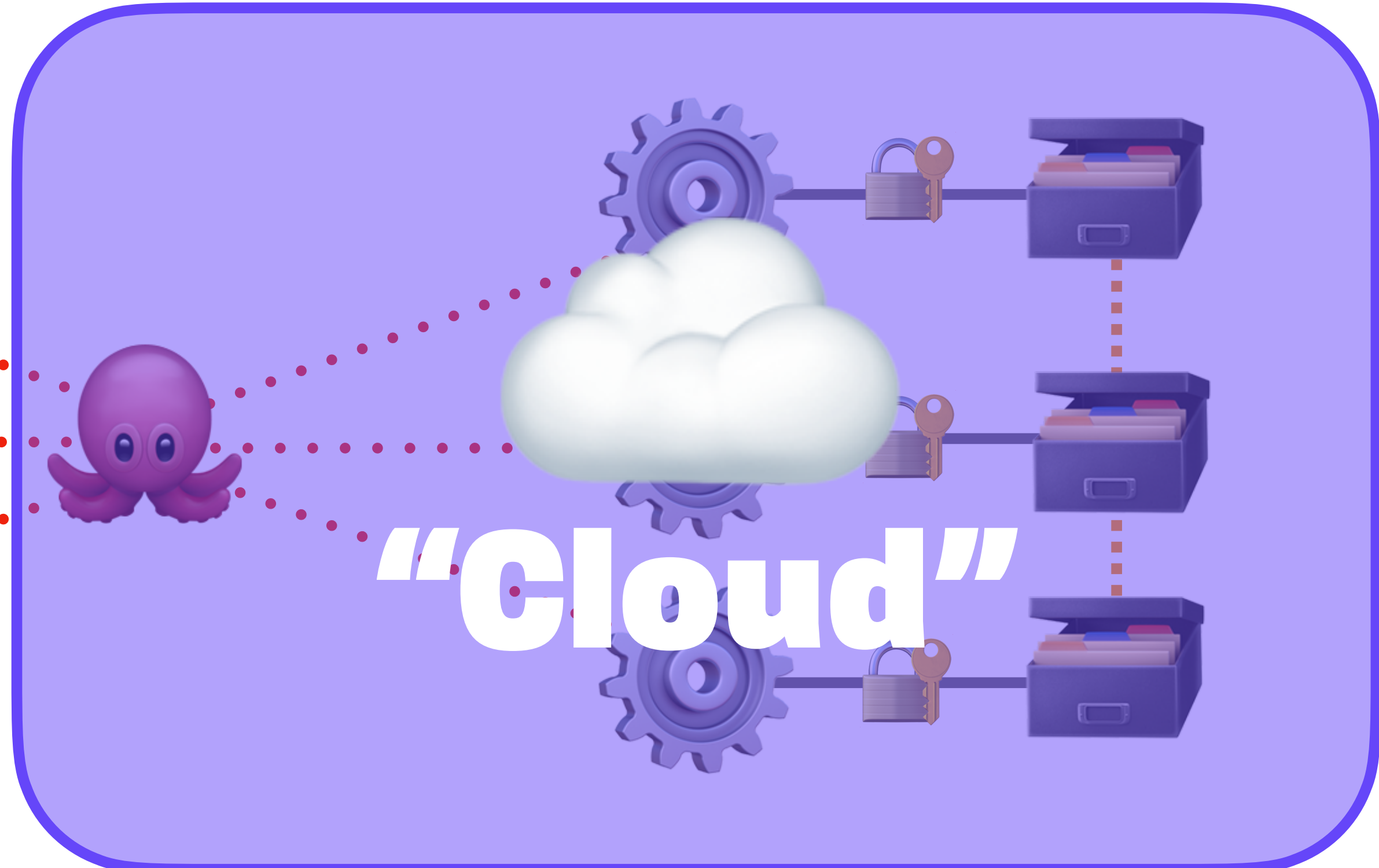
Motivation 🎭

Scaling Up



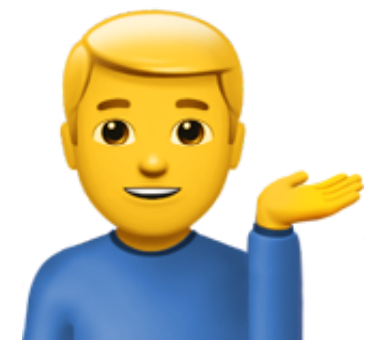
Motivation 🎭

Scaling Up



Motivation 🎭

Less is More



“Serverless”
(AKA more servers)

...and so it was for many years...

...and so it was for many years...



Motivation 🎭

Natural Consequences 🌱

Motivation 🎭

Natural Consequences 🌱

- Single source of truth ("**the**" database)

Motivation 🎭

Natural Consequences 🌱

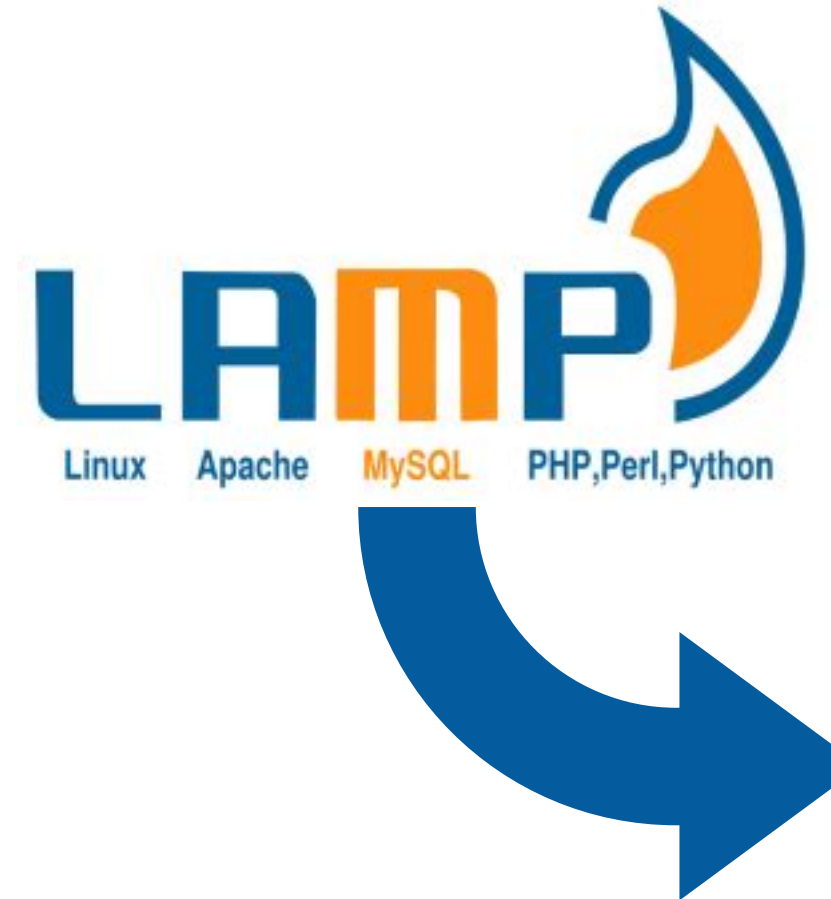
- Single source of truth ("**the**" database)
- Server-centric
 - "Full stack development"
 - DevOps, Docker, k8s
 - How to train enough engineers?



Motivation 🎭

Natural Consequences 🌱

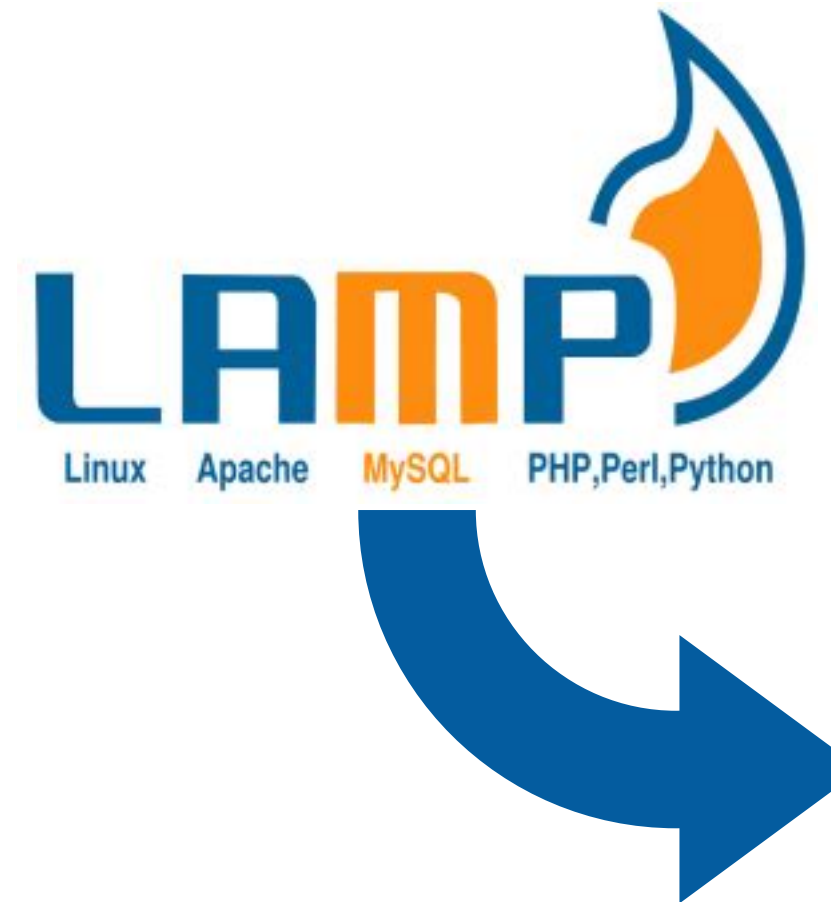
- Single source of truth ("**the**" database)
- Server-centric
 - "Full stack development"
 - DevOps, Docker, k8s
 - How to train enough engineers?



Motivation 🎭

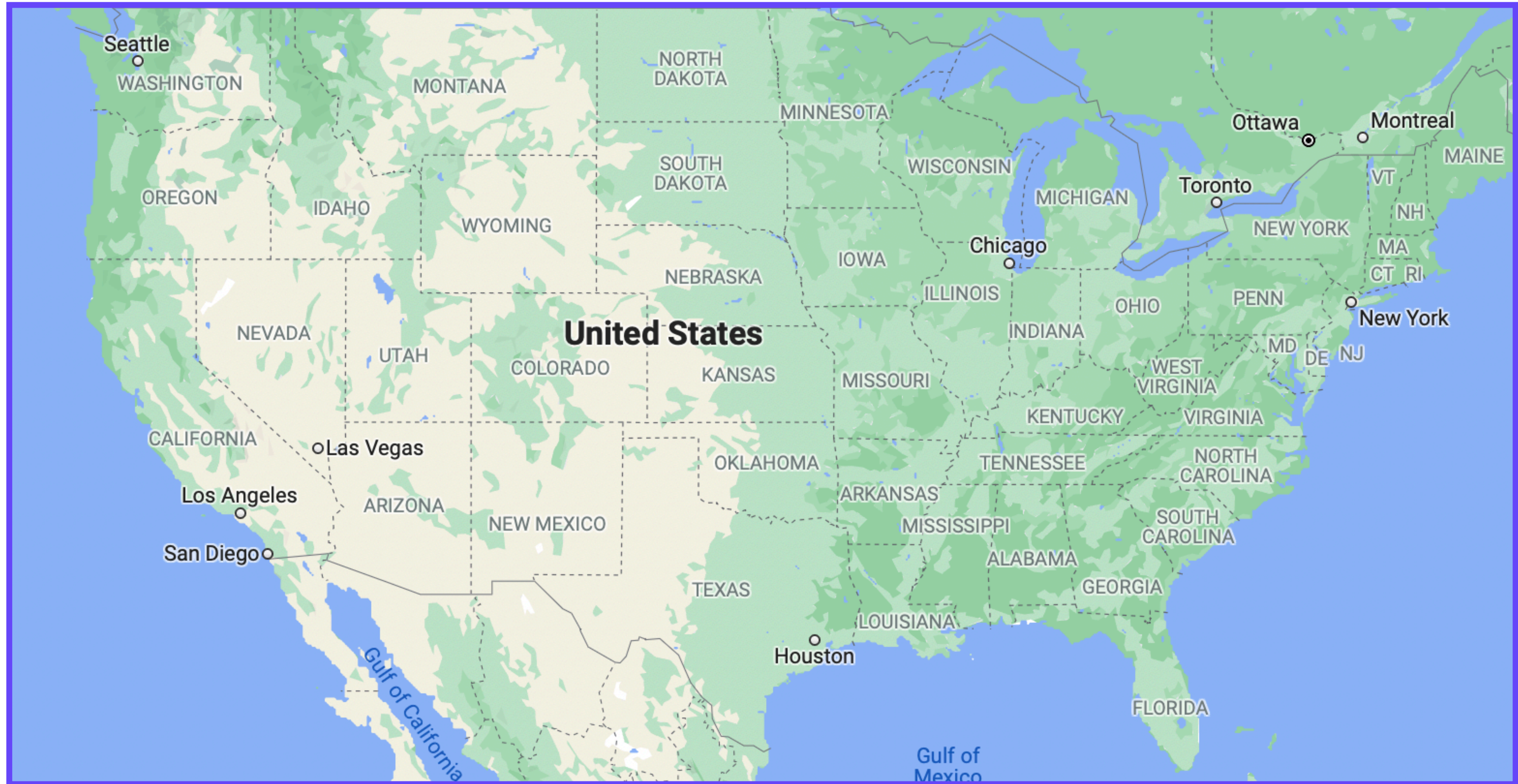
Natural Consequences 🌱

- Single source of truth ("**the**" database)
- Server-centric
 - "Full stack development"
 - DevOps, Docker, k8s
 - How to train enough engineers?
- Infrastructure Hegemony
 - AWS (60%), GCP, Azure



Motivation 🎭

Sending a "Direct" Message



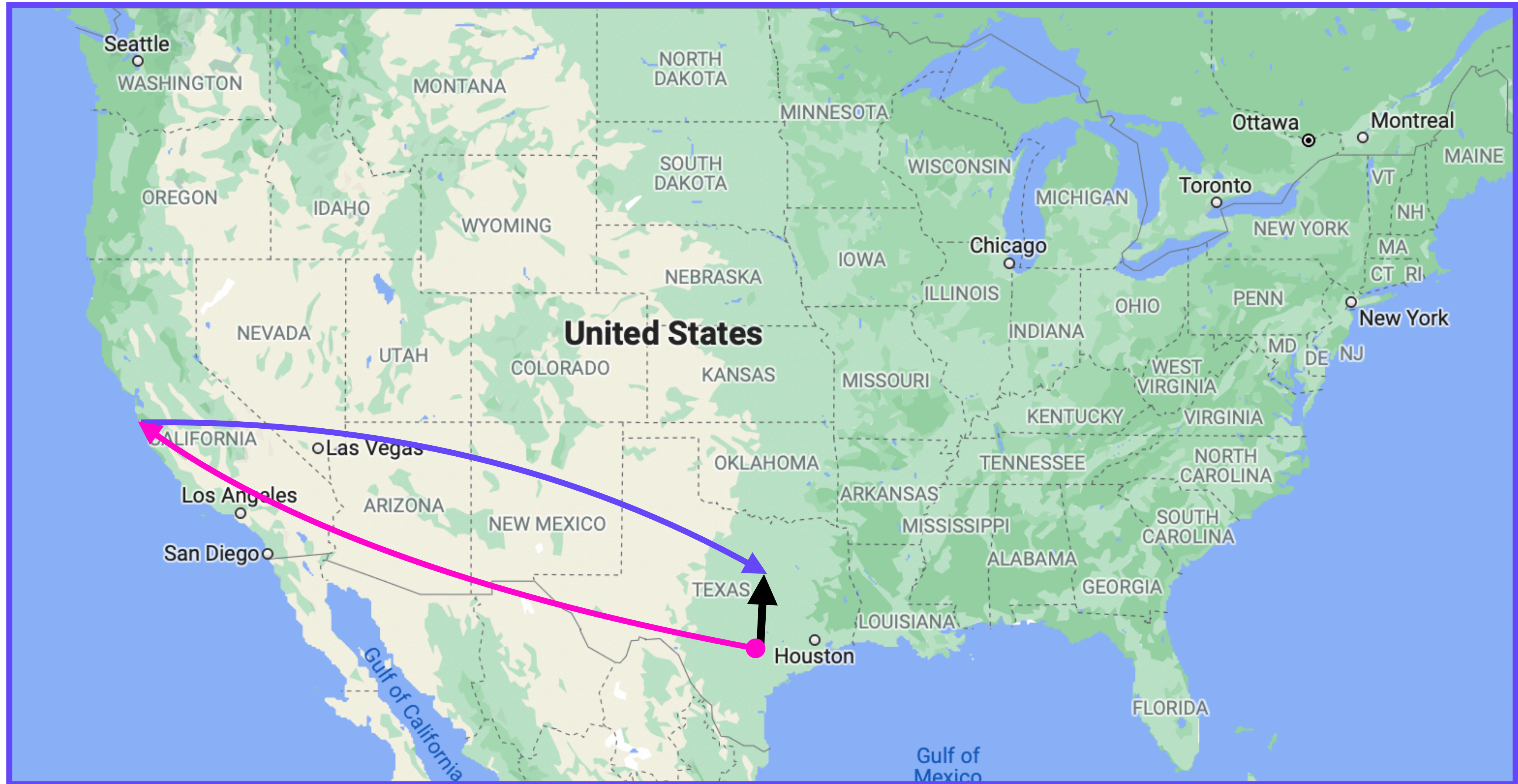
Motivation 🎭

Sending a "Direct" Message



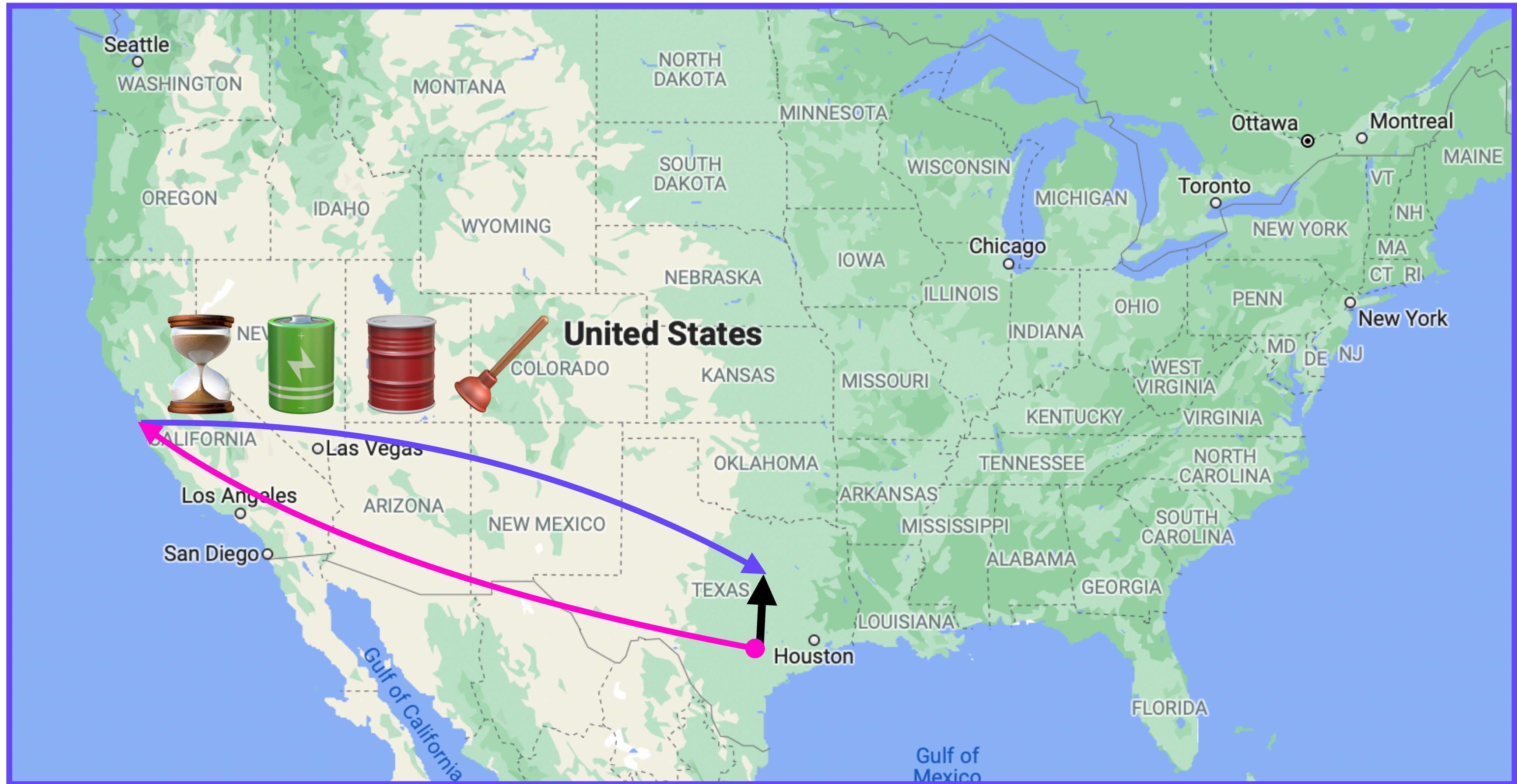
Motivation 🎭

Sending a "Direct" Message



Motivation 🎭

Sending a "Direct" Message



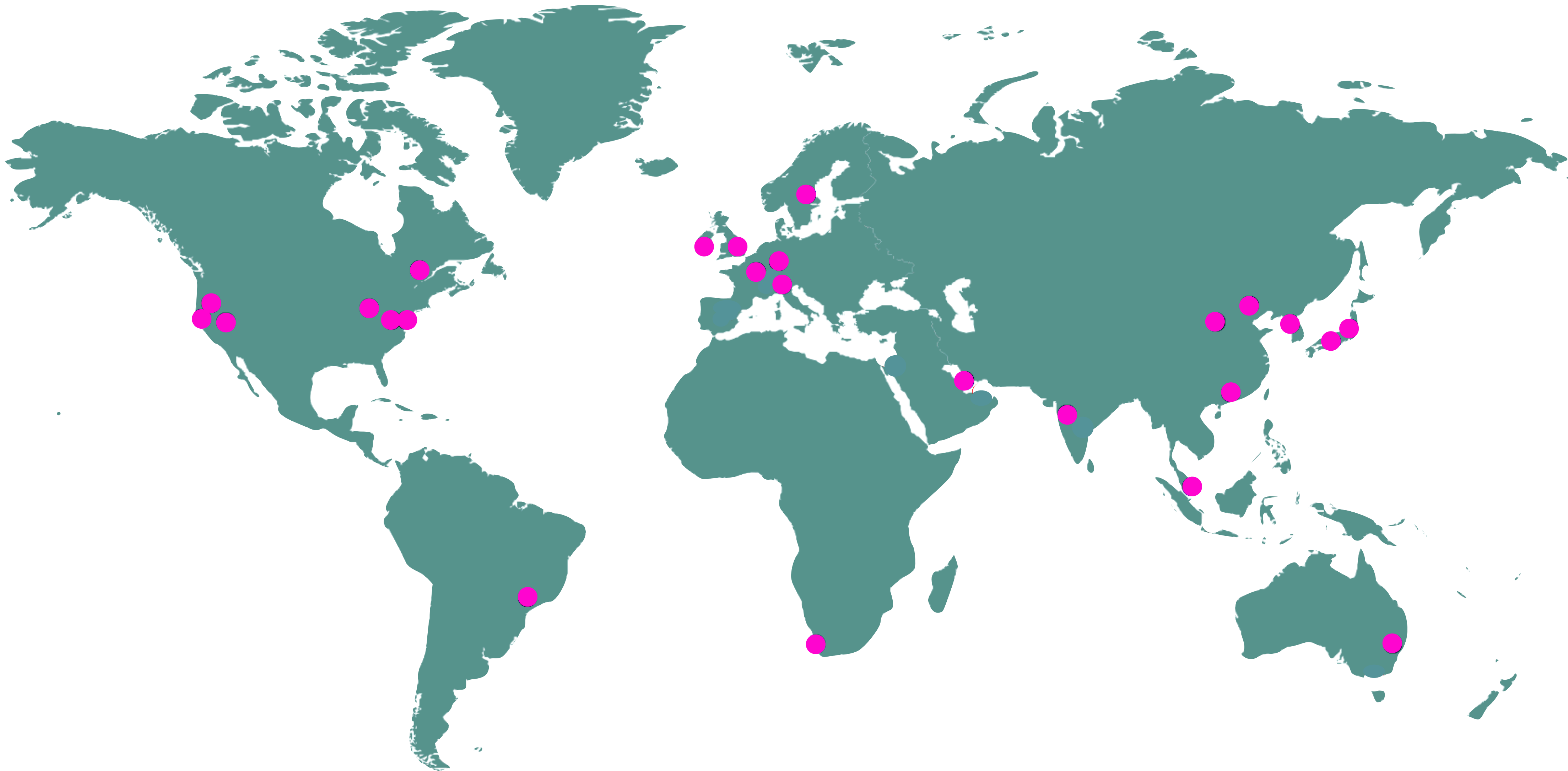
New Environment 🛰️

Users vs Cloud Infra



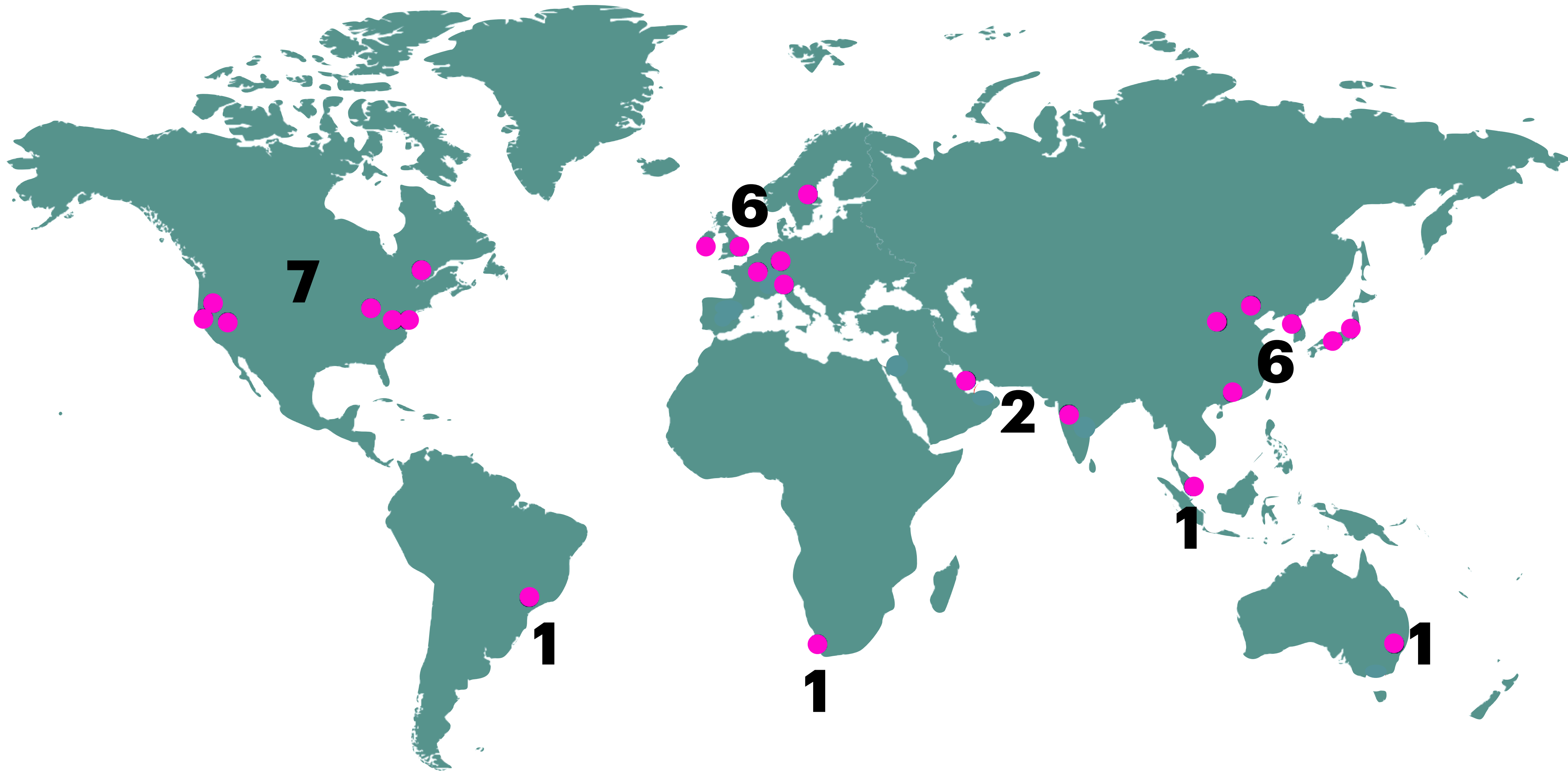
New Environment 🛰️

Users vs Cloud Infra



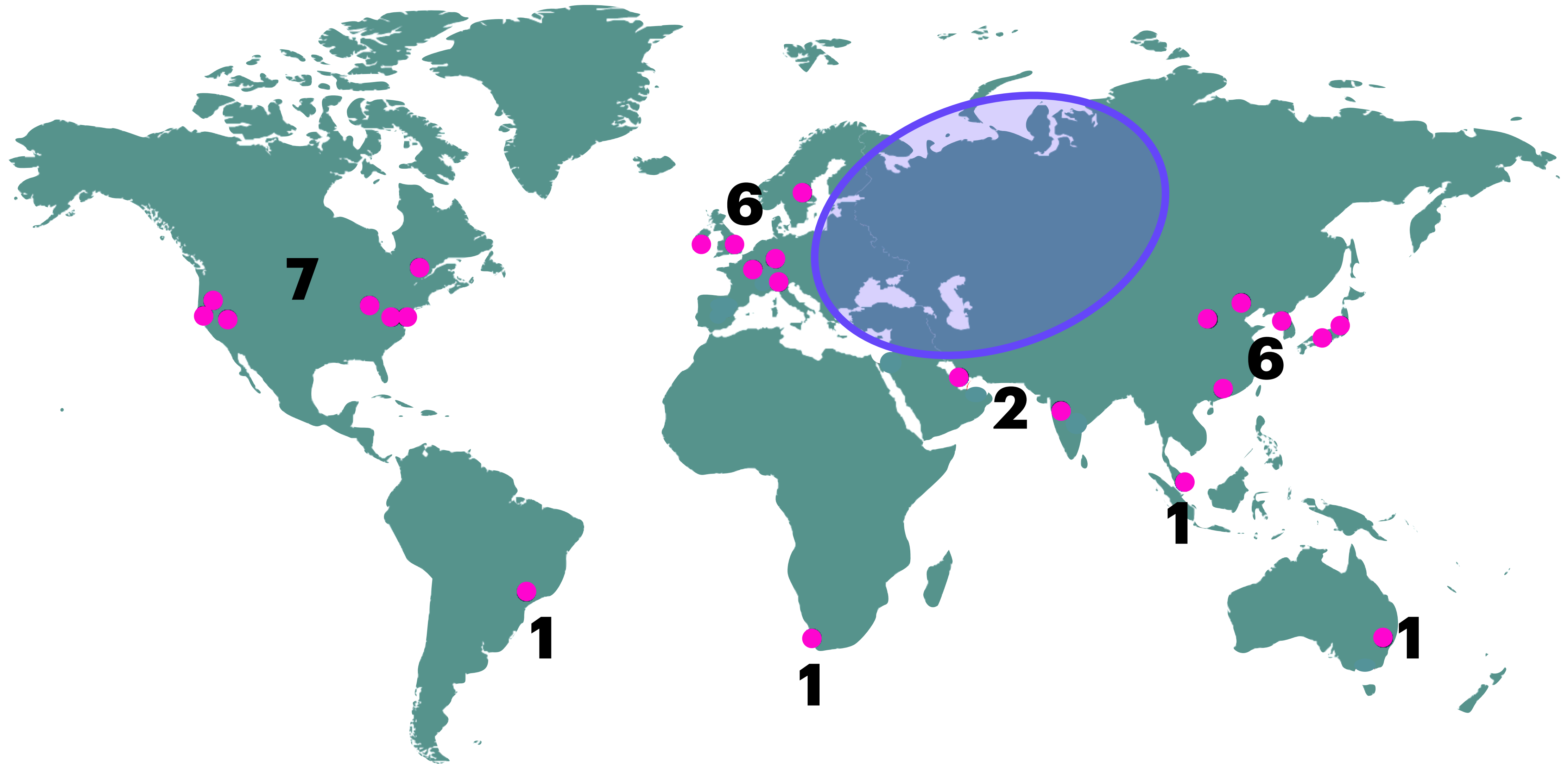
New Environment 🛰️

Users vs Cloud Infra



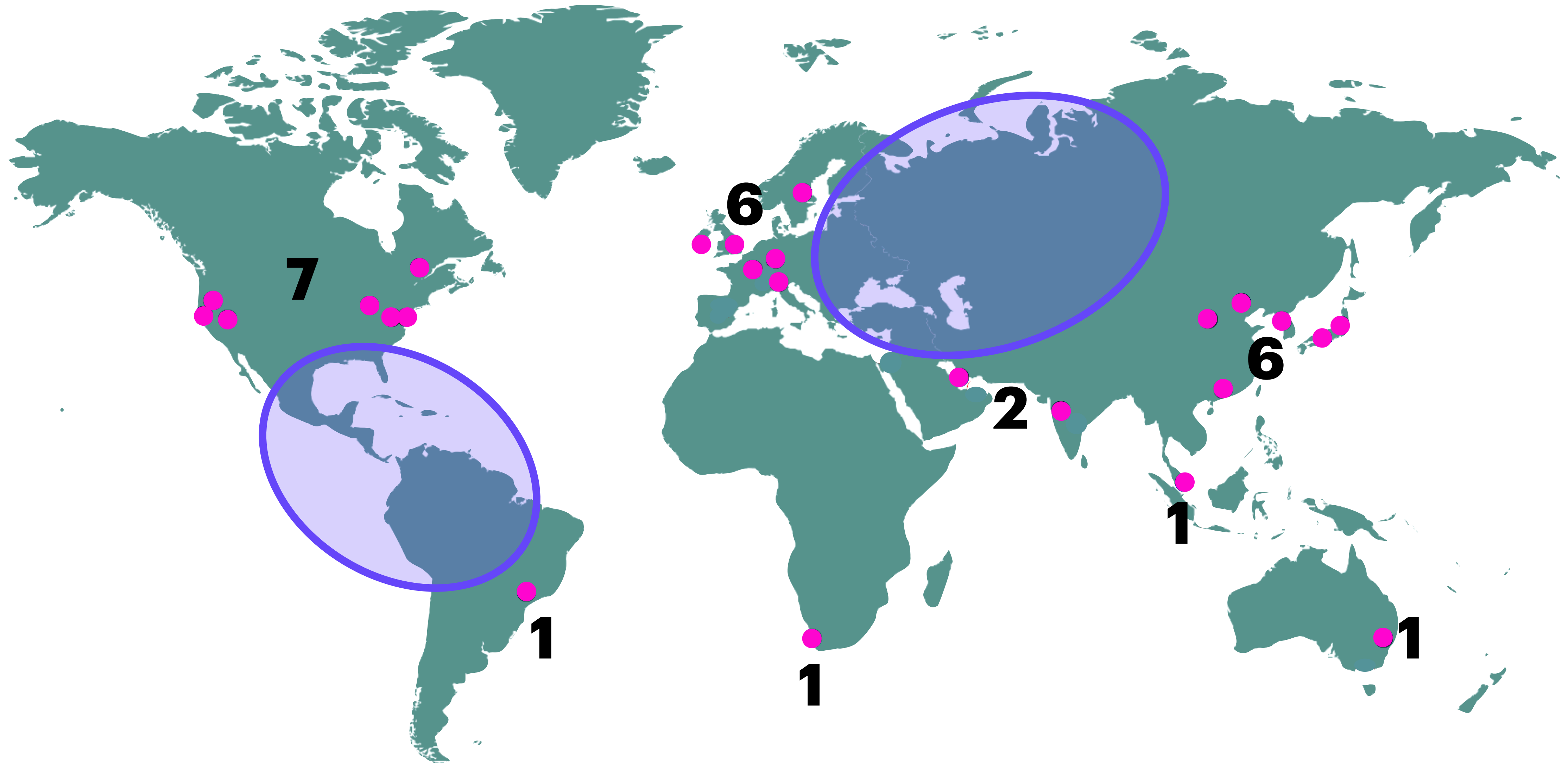
New Environment 🛰️

Users vs Cloud Infra



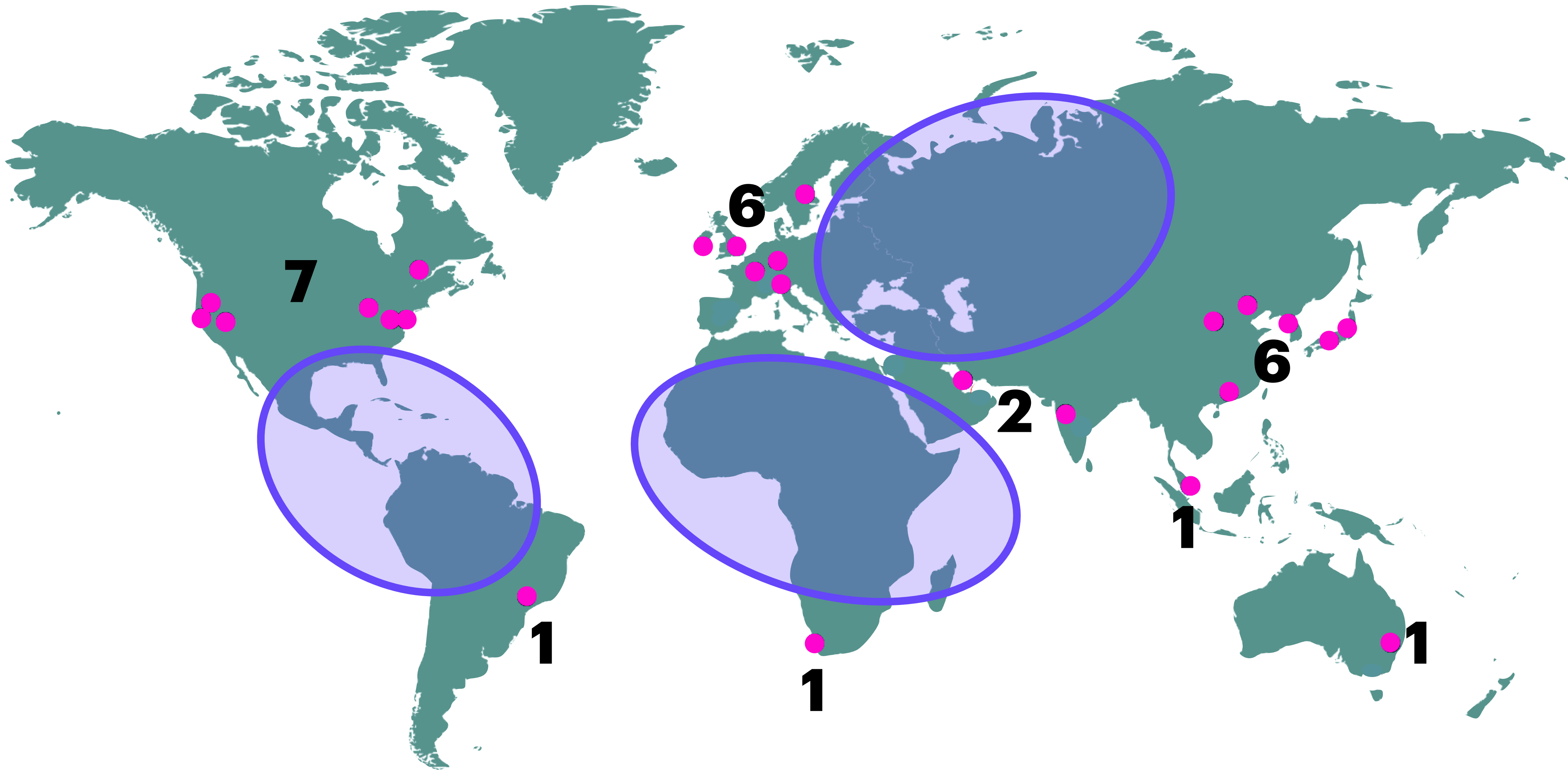
New Environment 🛰️

Users vs Cloud Infra



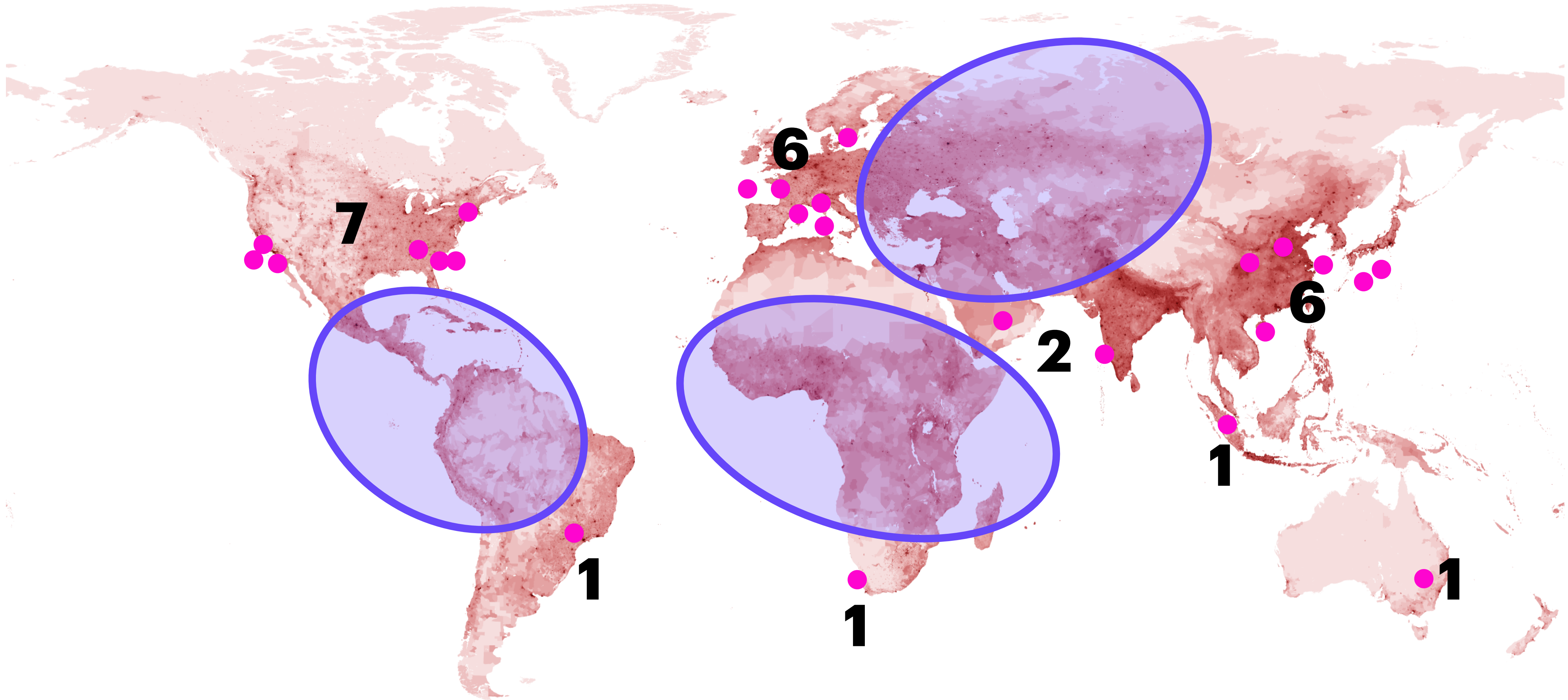
New Environment 🛰️

Users vs Cloud Infra



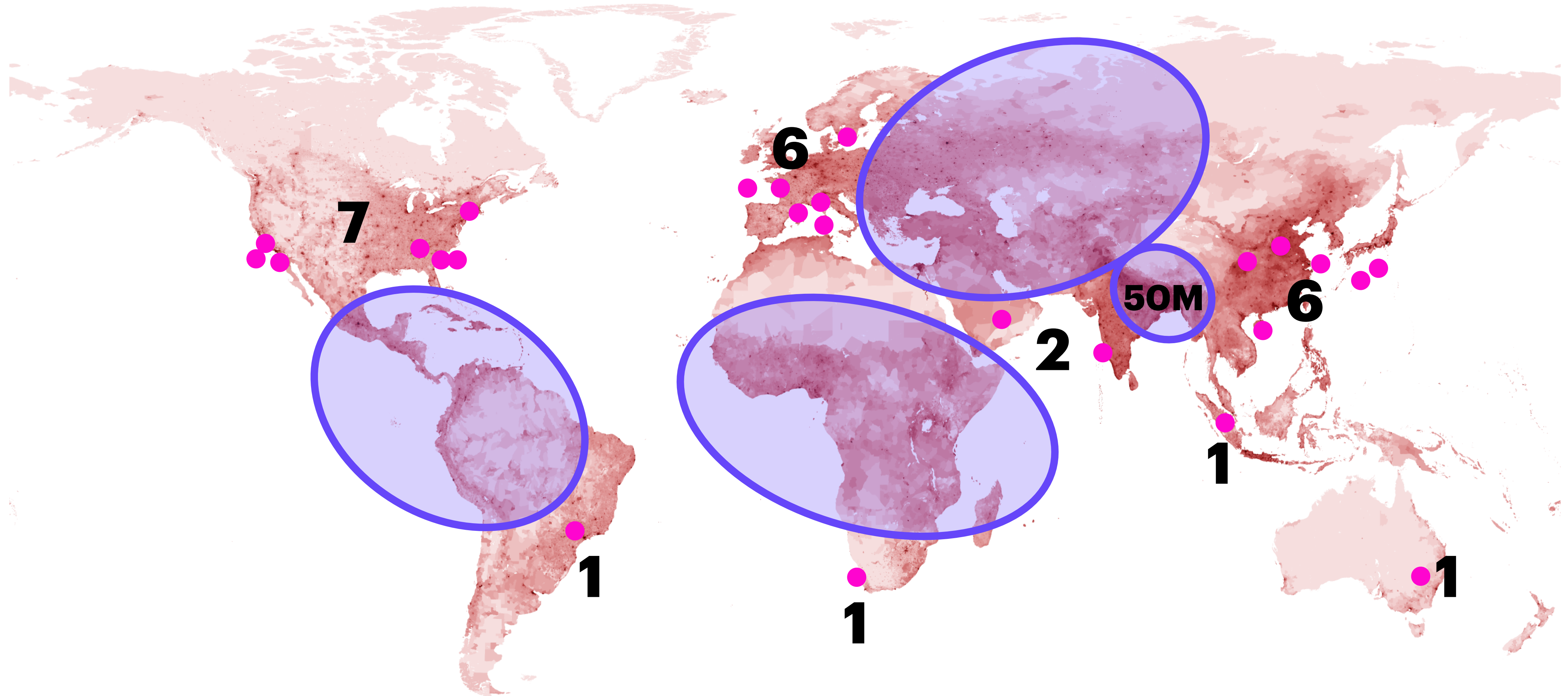
New Environment 🛰️

Users vs Cloud Infra



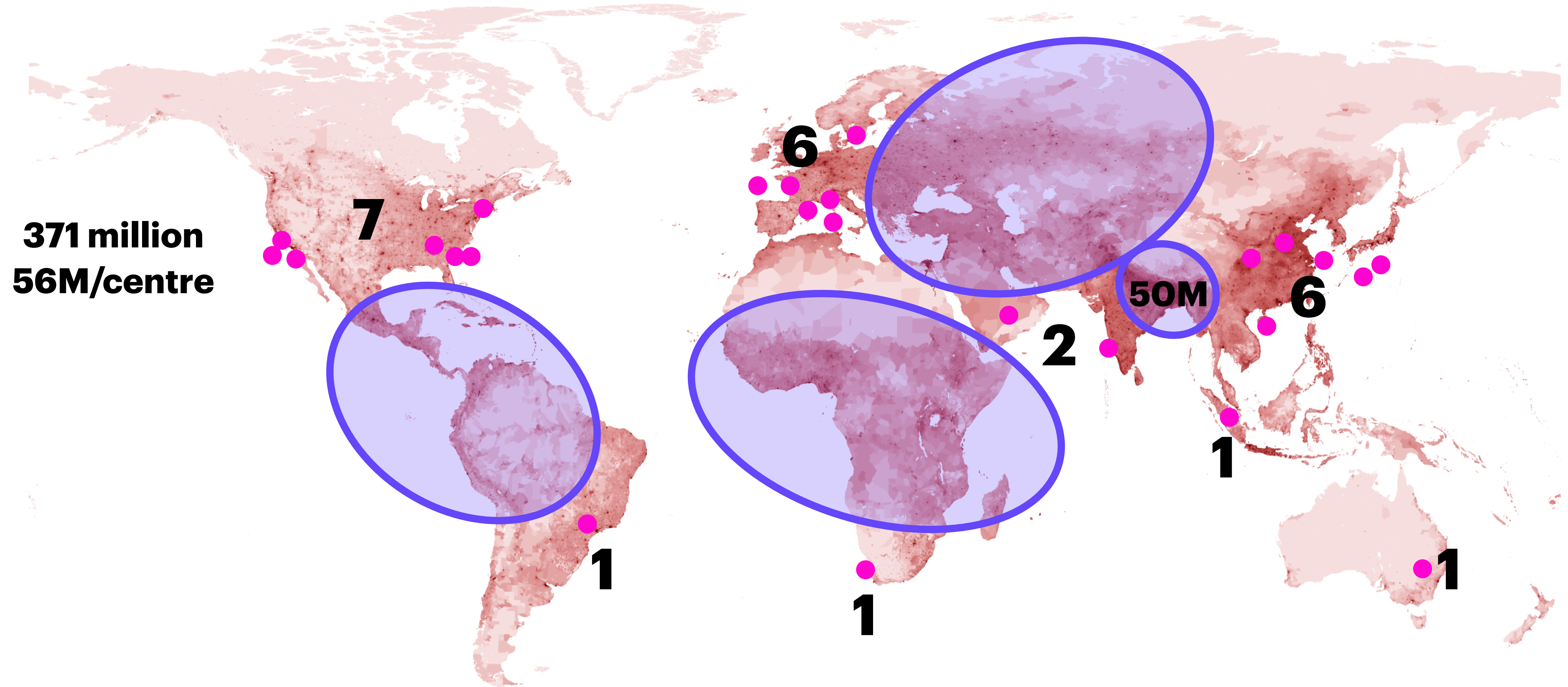
New Environment 🛰️

Users vs Cloud Infra



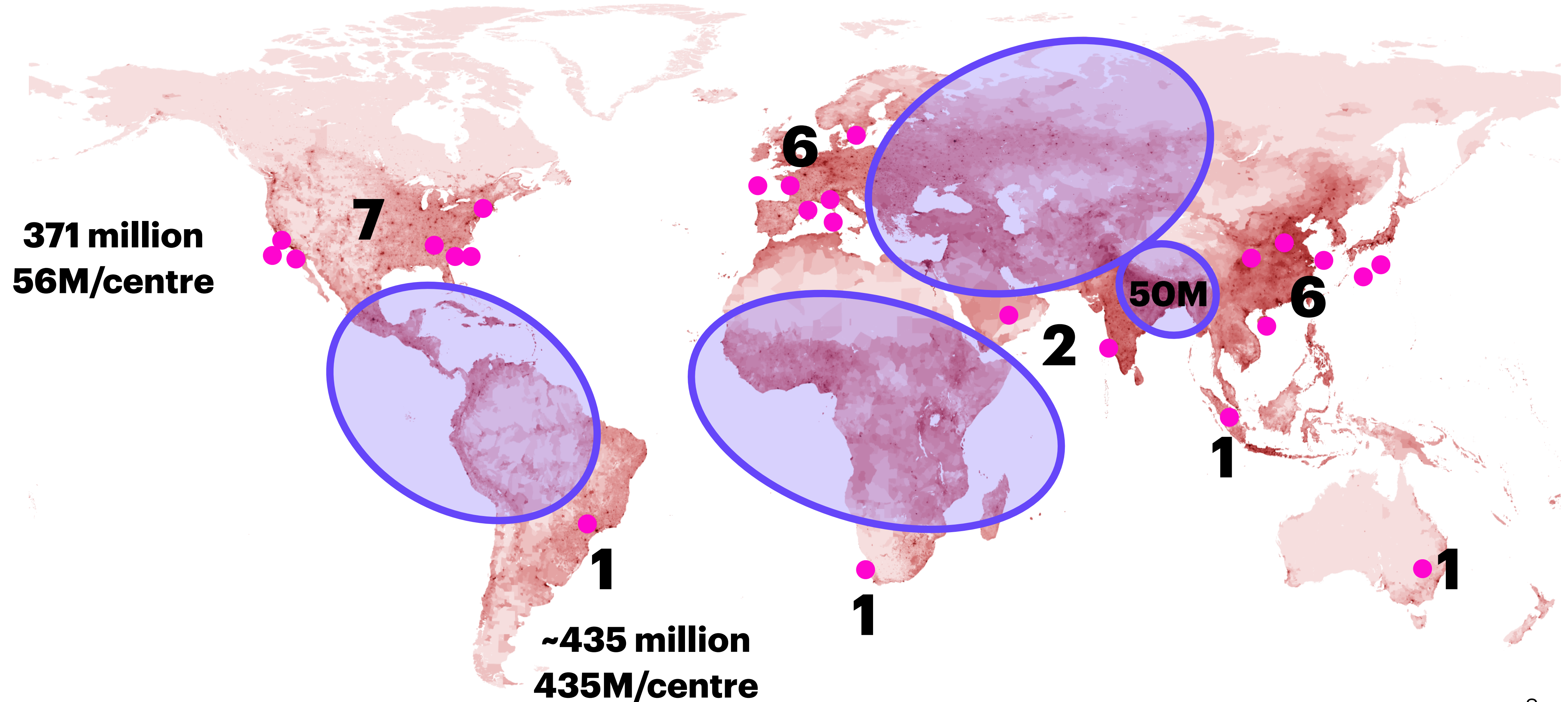
New Environment 🛰️

Users vs Cloud Infra



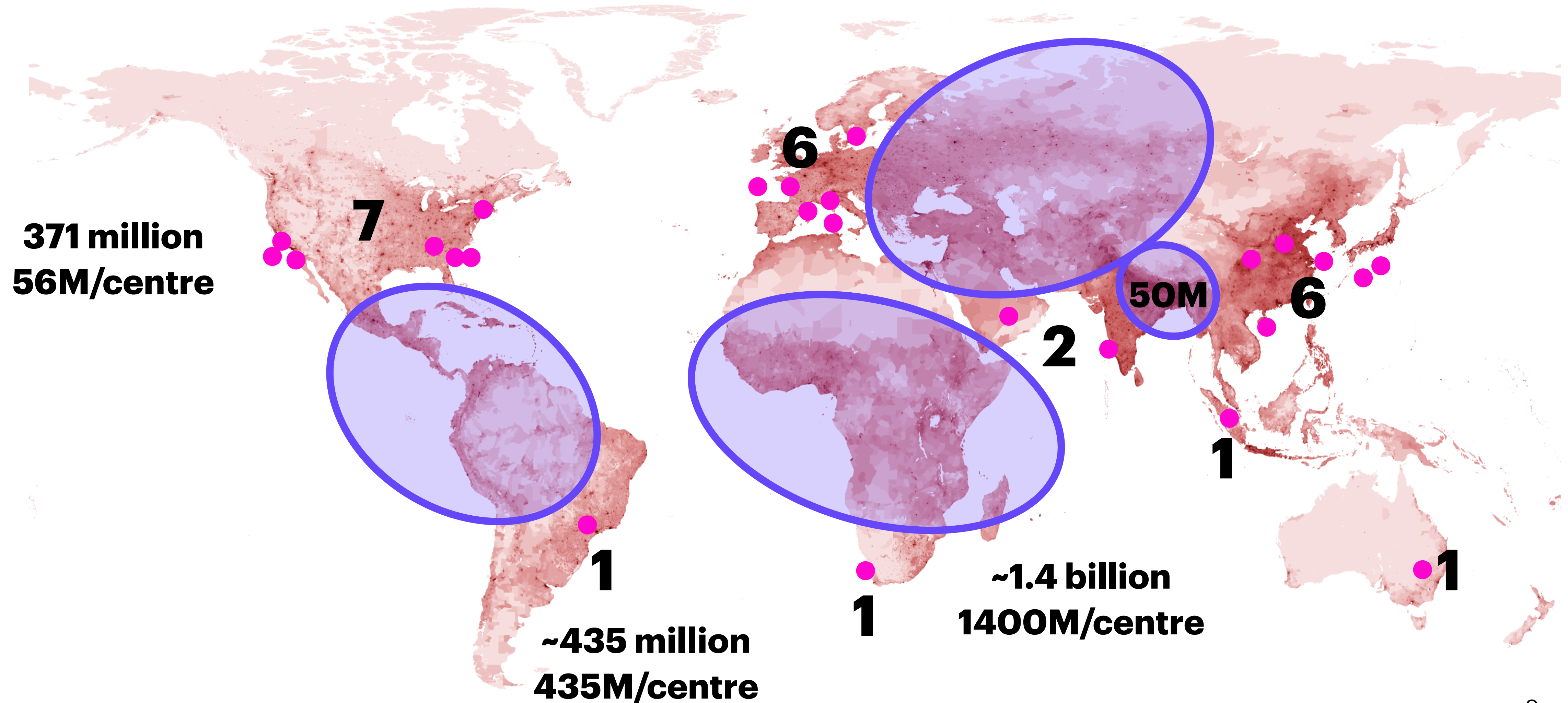
New Environment 🛰️

Users vs Cloud Infra



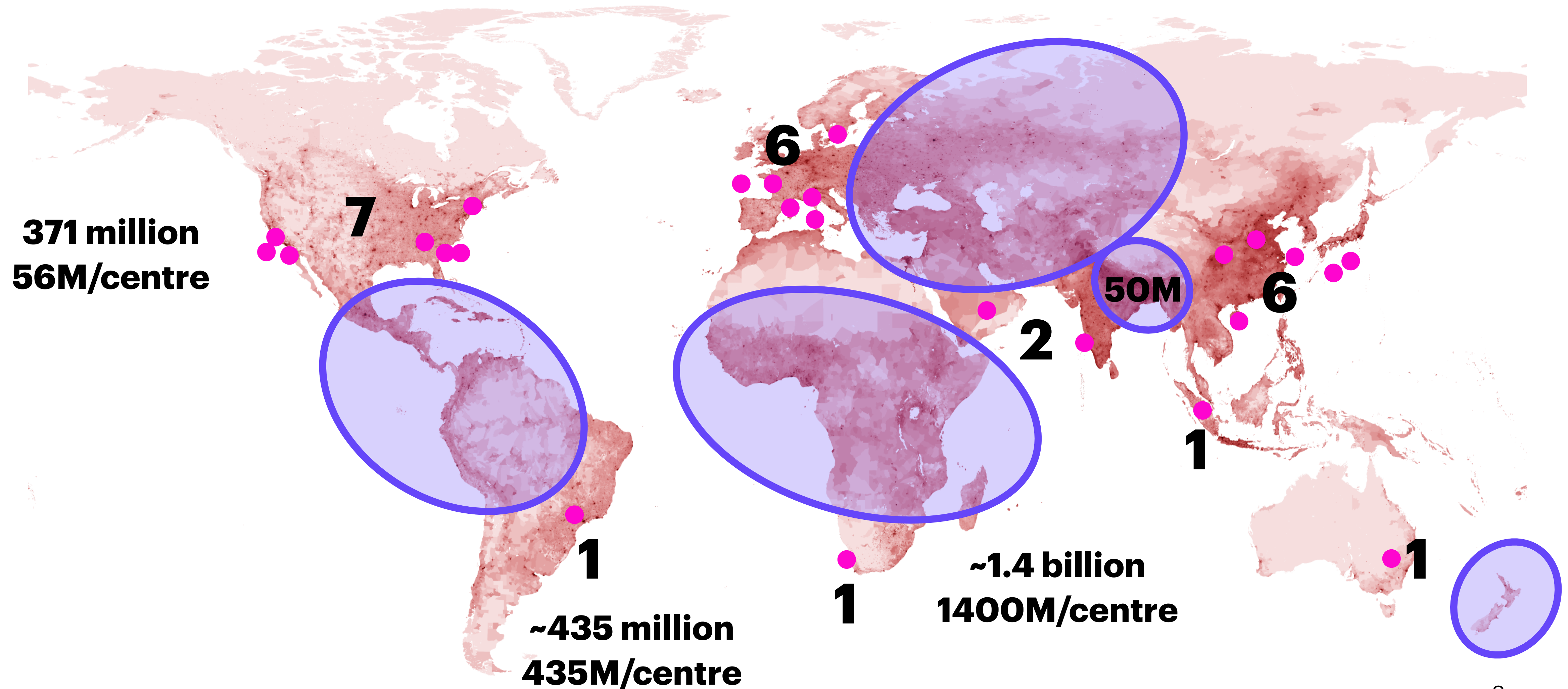
New Environment 🛰️

Users vs Cloud Infra



New Environment 🛰️

Users vs Cloud Infra



Video Killed the Radio Star

A New Environment



New Environment 

New Environment

New Environment 

New Environment

Then 

Now 









New Environment 🛰️

New Environment

| | Then 🖨️ | Now 🚀 |
|------|--------------|------------|
| Need | Convenient 🙋 | Critical 🚨 |











New Environment 

New Environment

| | Then  | Now  |
|----------|--|---|
| Need | Convenient  | Critical  |
| Location | Data Centre  | Powerful Clients (M1, IoT)    |













New Environment 

New Environment

| | Then  | Now  |
|----------|--|---|
| Need | Convenient  | Critical  |
| Location | Data Centre  | Powerful Clients (M1, IoT)    |
| Access |  |  |


















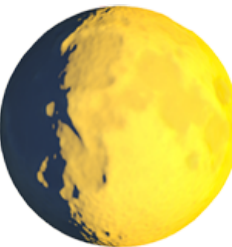
New Environment 

New Environment

| | Then  | Now  |
|------------|--|---|
| Need | Convenient  | Critical  |
| Location | Data Centre  | Powerful Clients (M1, IoT)    |
| Access |  |  |
| Bottleneck | Bandwidth  | Latency  |

New Environment 

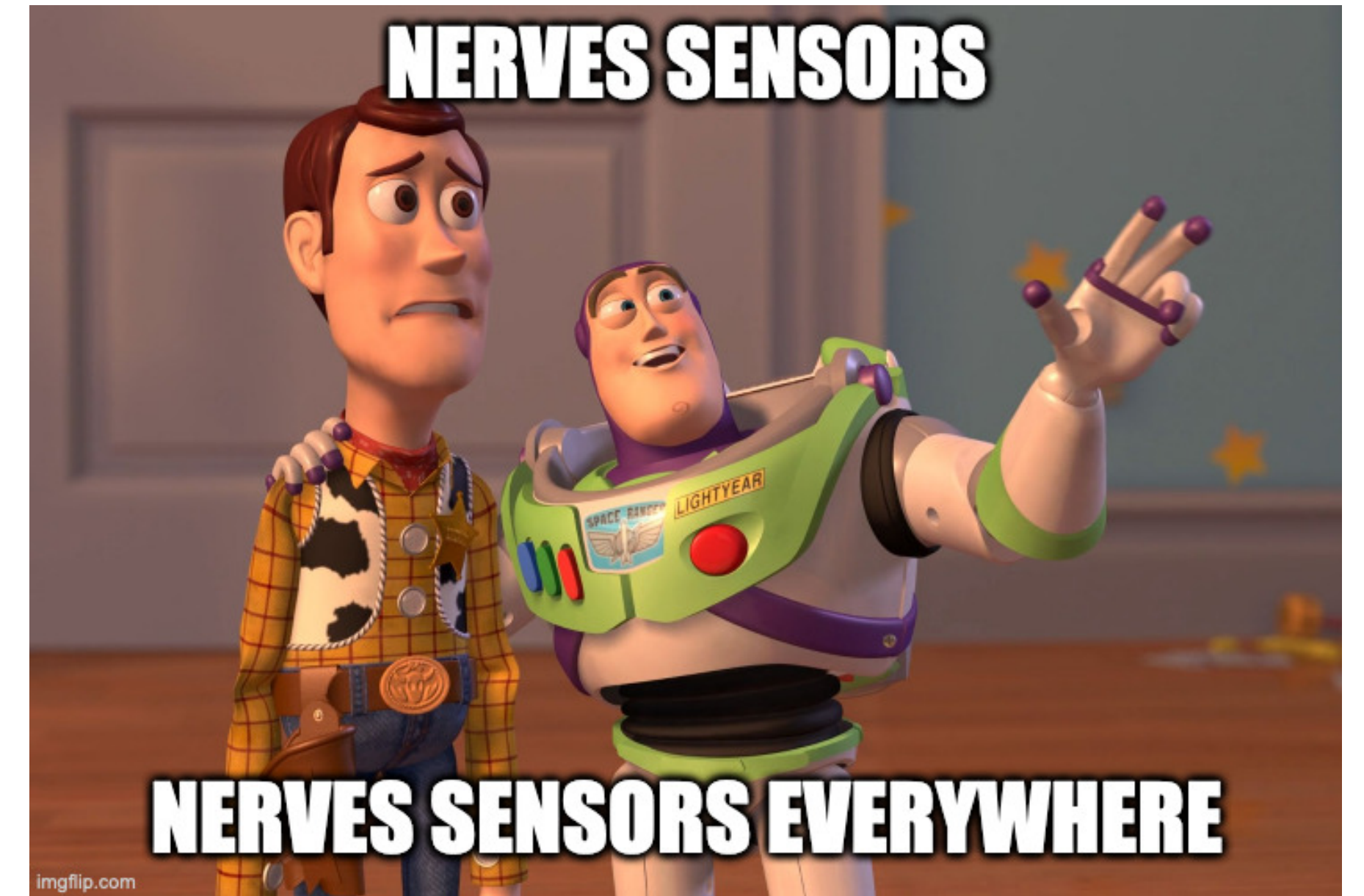
New Environment

| | Then  | Now  |
|------------|---|---|
| Need | Convenient  | Critical  |
| Location | Data Centre  | Powerful Clients (M1, IoT)    |
| Access |  |  |
| Bottleneck | Bandwidth  | Latency  |
| Market |   |    ...  |

High Volume 📣

Science Fiction → *Reality*

- Remote surgery
- Extended reality
- Autonomous vehicles
- Location transparency
- Competitive cloud gaming
- Realtime manufacturing
- Continuous ML



Literal brain surgery over 5G
3,000km ~ 1,900mi

Sensor data explosion will kill the cloud.
[...] **existing infrastructure** will not be able to
handle the **volumes or the rates**

We are absolutely going to return to a peer-to-peer
computing [...] not unlike **distributed computing**

We are going to move to a world of
data-centric programming.

~ a16z, "The End of Cloud Computing"

Sensor data explosion will kill the cloud.
[...] **existing infrastructure** will not be able to
handle the **volumes or the rates**

We are absolutely going to return to a peer-to-peer
computing [...] not unlike **distributed computing**



We are going to move to a world of
data-centric programming.

~ a16z, "The End of Cloud Computing"

Sensor data explosion will kill the cloud.
[...] **existing infrastructure** will not be able to
handle the **volumes or the rates**

We are absolutely going to return to a peer-to-peer
computing [...] not unlike **distributed computing**

We are going to move to a world of
data-centric programming.

~ a16z, "The End of Cloud Computing"

Sensor data explosion will kill the cloud.
[...] **existing infrastructure** will not be able to
handle the **volumes or the rates**

We are absolutely going to return to a peer-to-peer
computing [...] not unlike **distributed computing**

We are going to move to a world of
data-centric programming.

~ a16z, "The End of Cloud Computing"



Sensor data explosion will kill the cloud.
[...] **existing infrastructure** will not be able to
handle the **volumes or the rates**

We are absolutely going to return to a peer-to-peer
computing [...] not unlike **distributed computing**

We are going to move to a world of
data-centric programming.

~ a16z, "The End of Cloud Computing"



The World is Changing

Relativistic Computing



The ship that made the Kessel Run
in less than 12 parsecs

~ Han Solo

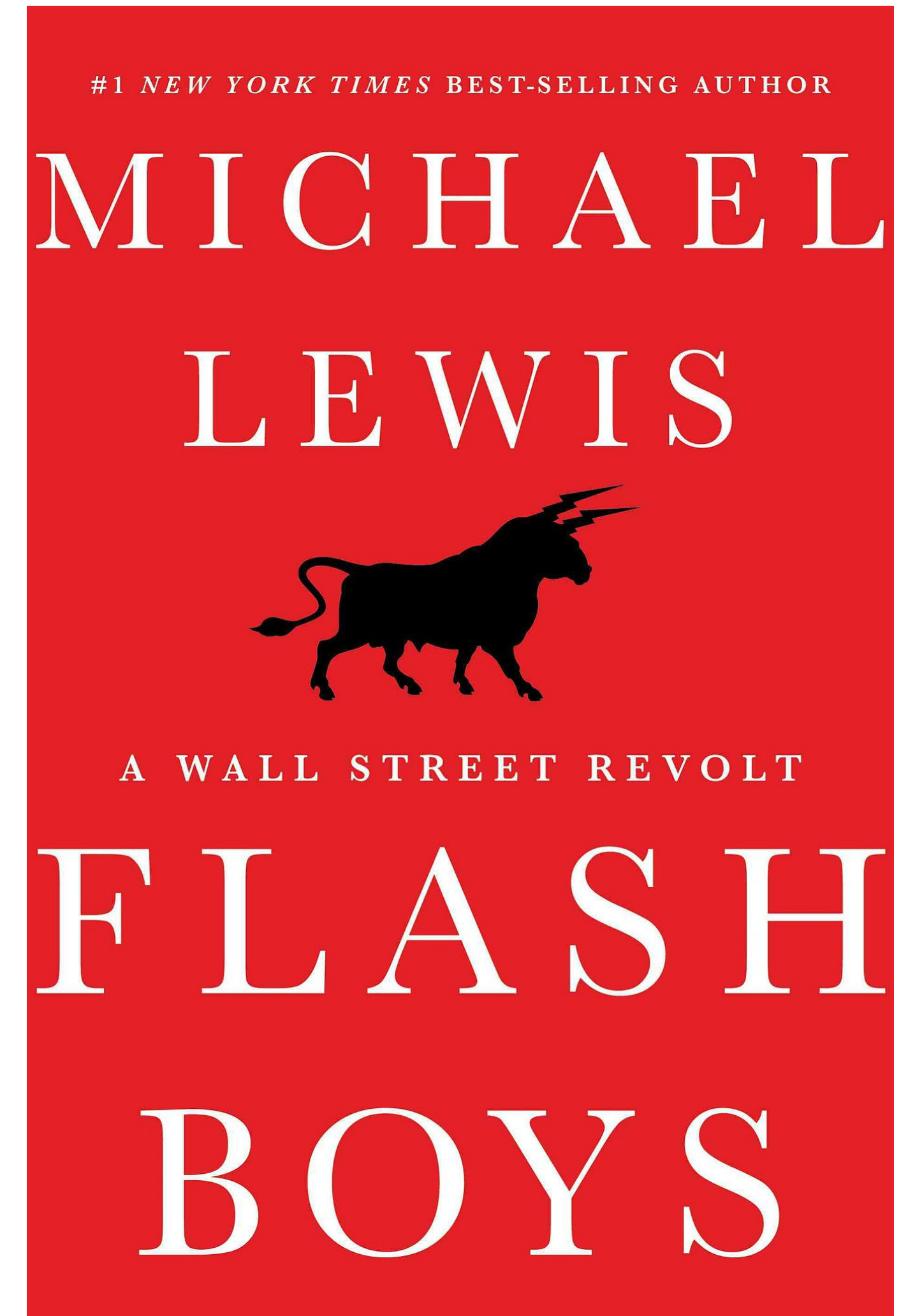
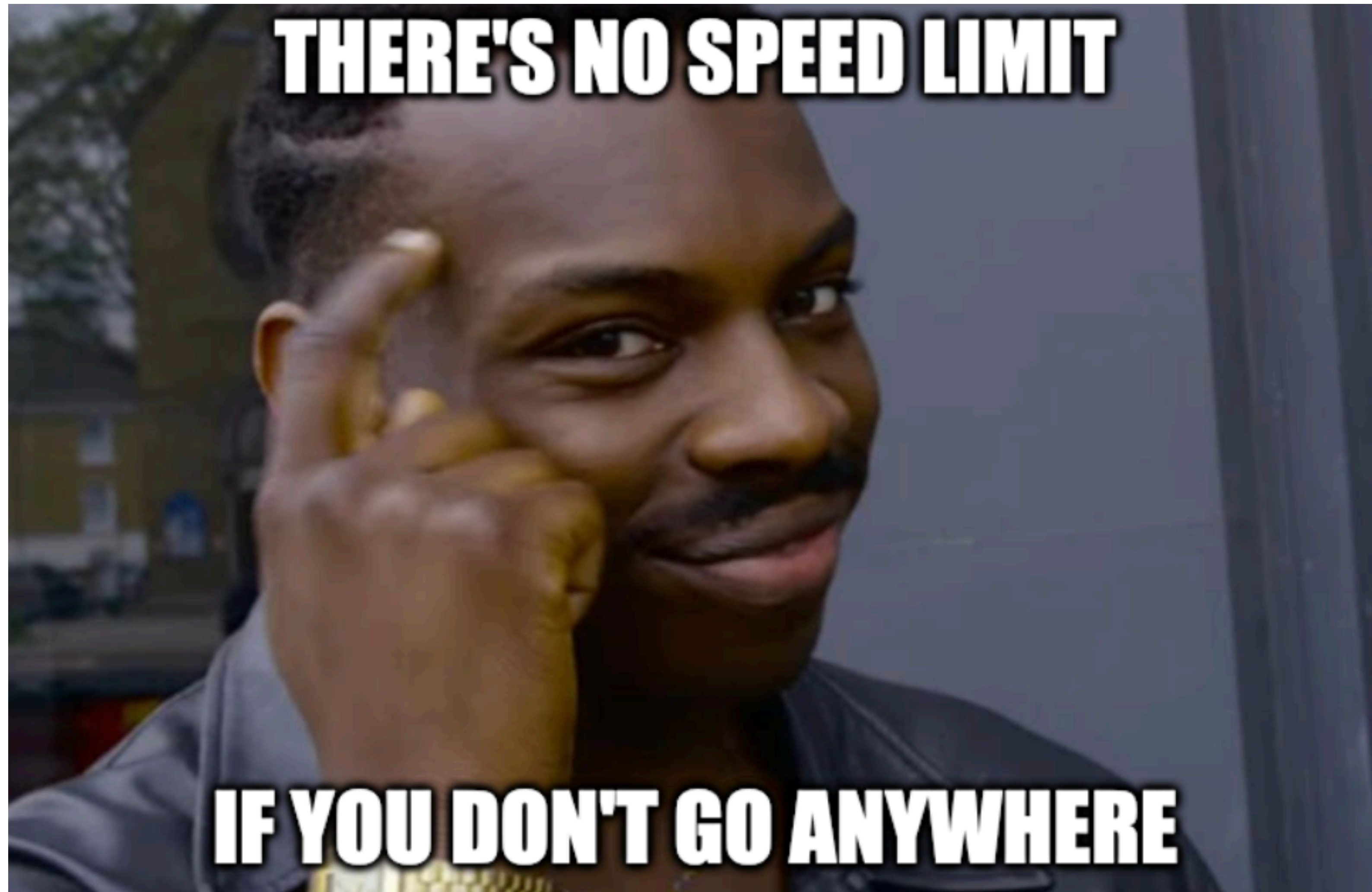


The ship that made the Kessel Run
in less than 12 parsecs

~ Han Solo

New Environment 🛰️

Moore's Law. But make it networked.



Low Latency 🐰

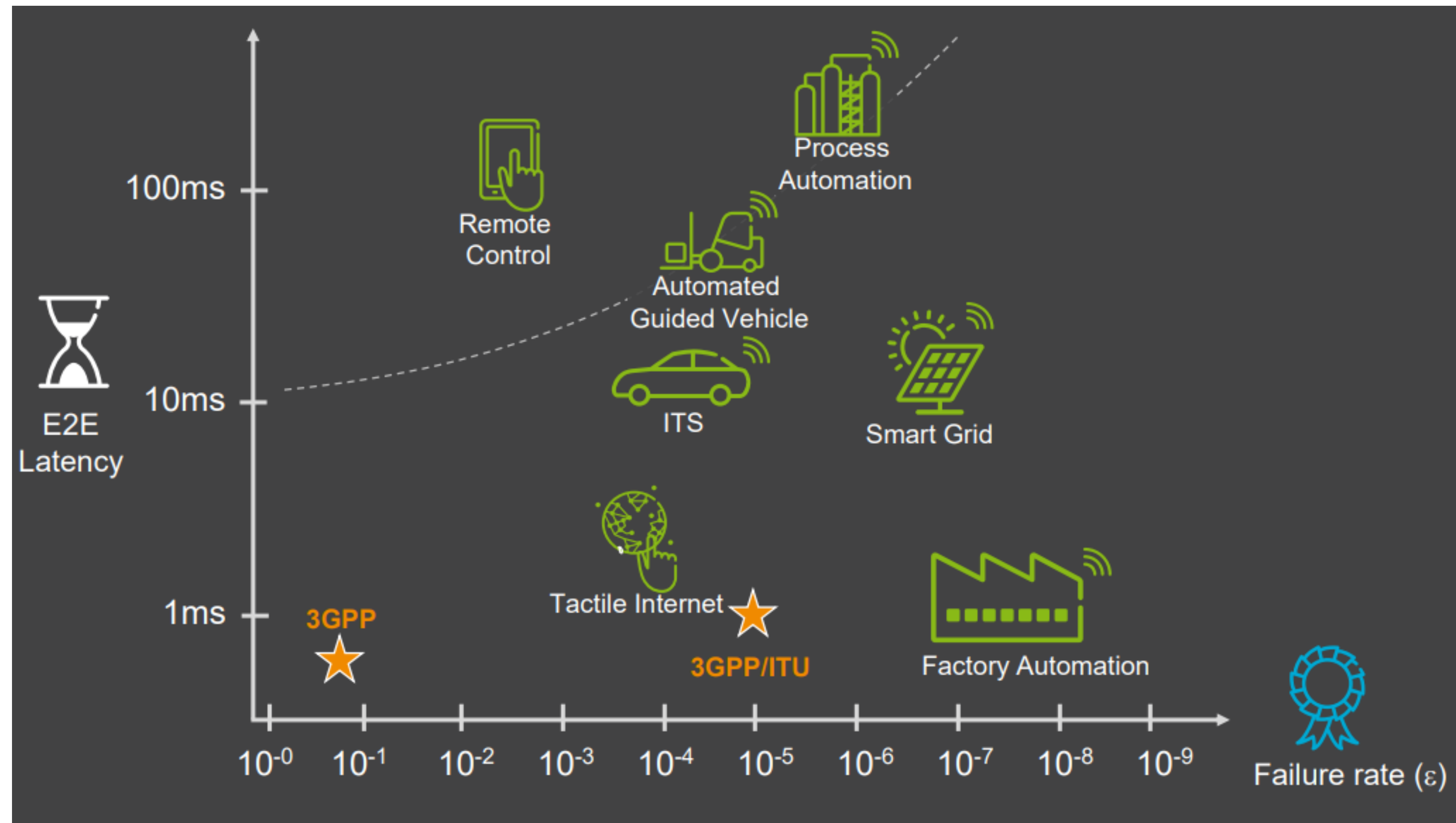
Latency is a Physical Limit 🚧

- Bandwidth max not even close
- Speed of light **causality**
- Edge dominates < 40ms
- Best at ~8ms
- 1ms applications exist
- "Ultra Reliable Low Latency"

Low Latency 🐰

Latency is a Physical Limit 🚧

- Bandwidth max not even close
- Speed of light **causality**
- Edge dominates < 40ms
- Best at ~8ms
- 1ms applications exist
- "Ultra Reliable Low Latency"



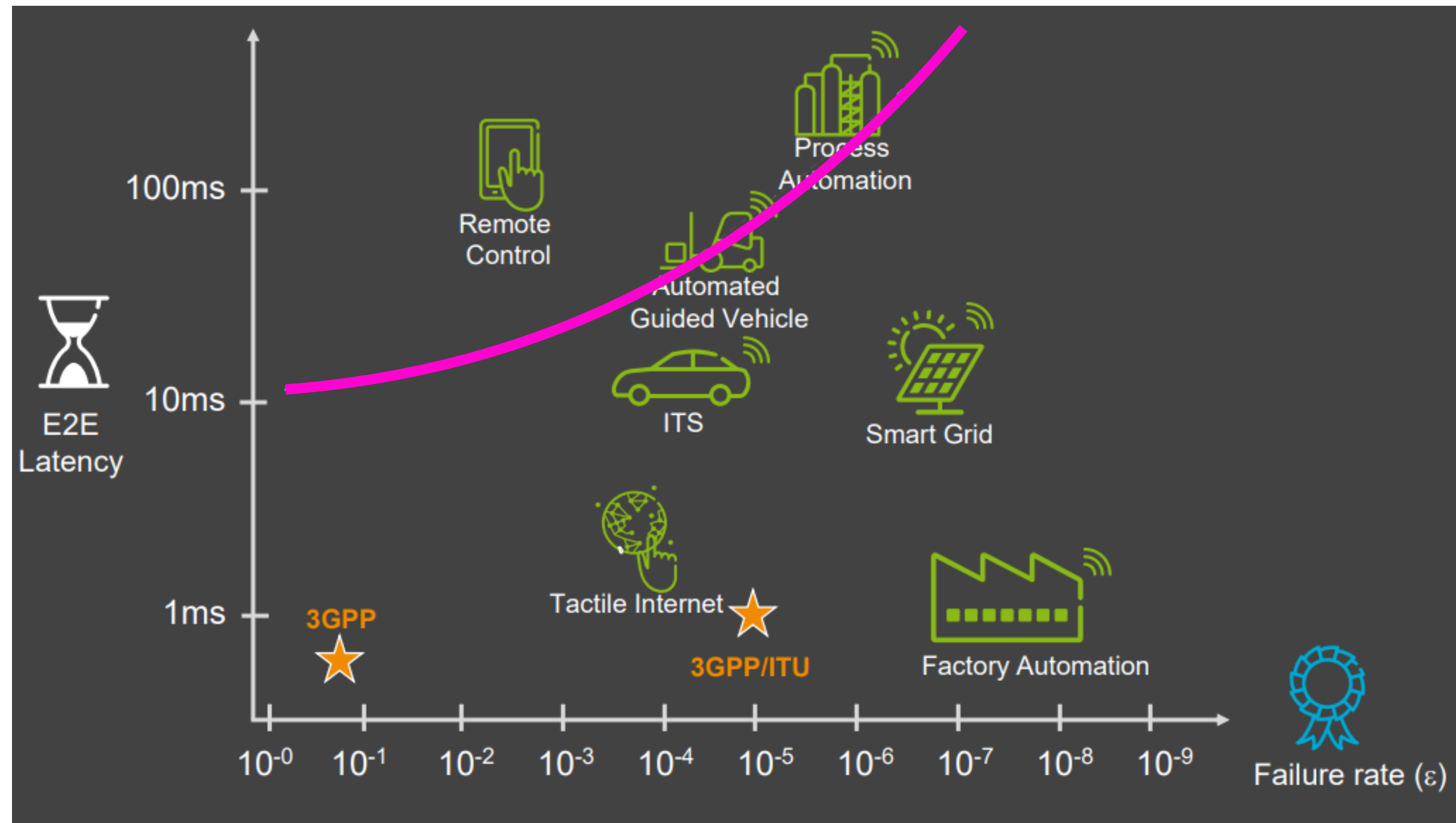
Source: Ericsson

http://cscn2017.ieee-cscn.org/files/2017/08/Janne_Peisa_Ericsson_CSCN2017.pdf

Low Latency 🐰

Latency is a Physical Limit 🚧

- Bandwidth max not even close
- Speed of light **causality**
- Edge dominates < 40ms
- Best at ~8ms
- 1ms applications exist
- "Ultra Reliable Low Latency"



Source: Ericsson

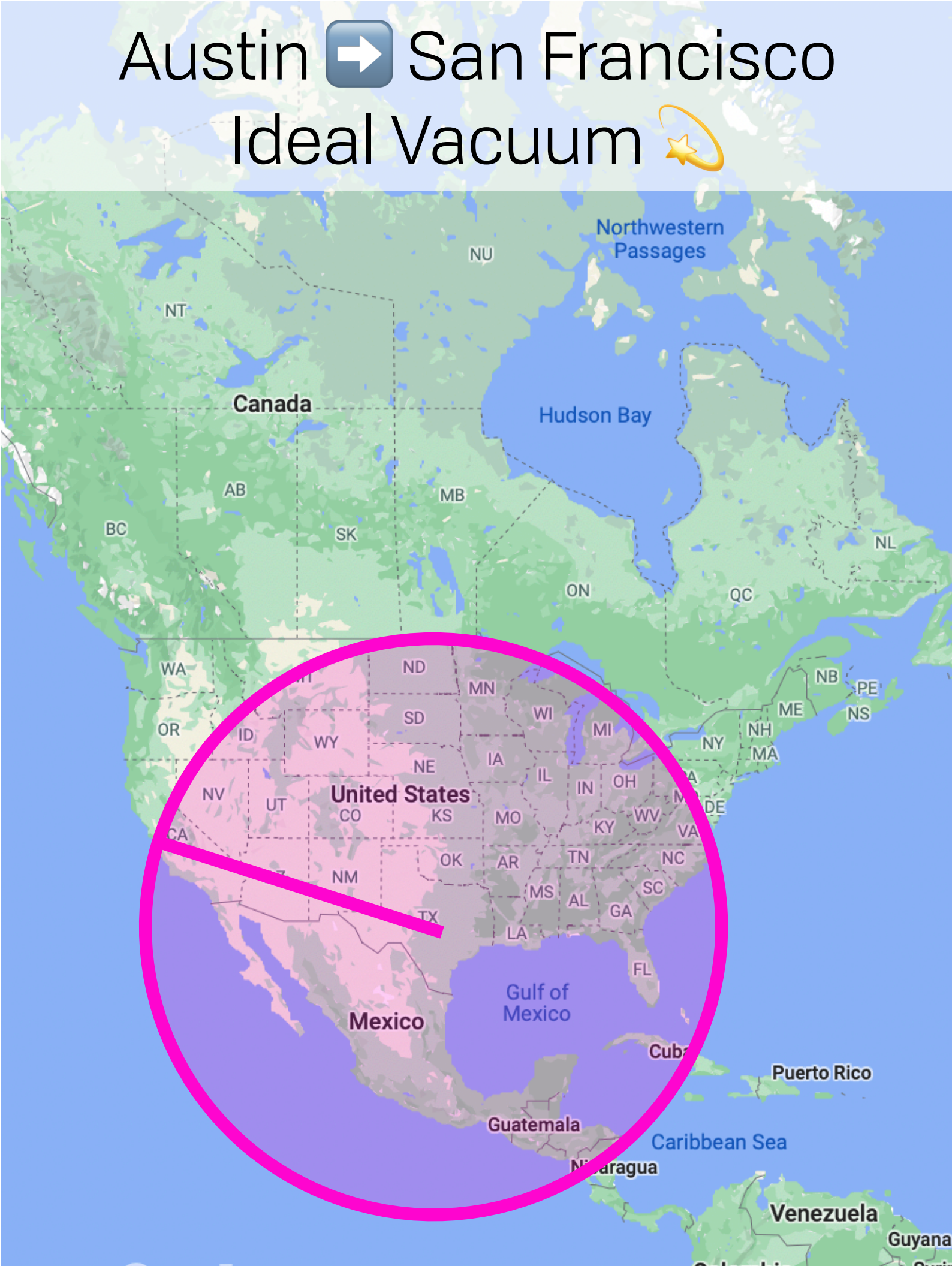
http://cscn2017.ieee-cscn.org/files/2017/08/Janne_Peisa_Ericsson_CSCN2017.pdf

Low Latency 

What 8ms Looks Like

Low Latency 🐰

What 8ms Looks Like



Low Latency 🐰

What 8ms Looks Like

Austin → San Francisco
Ideal Vacuum 🌟



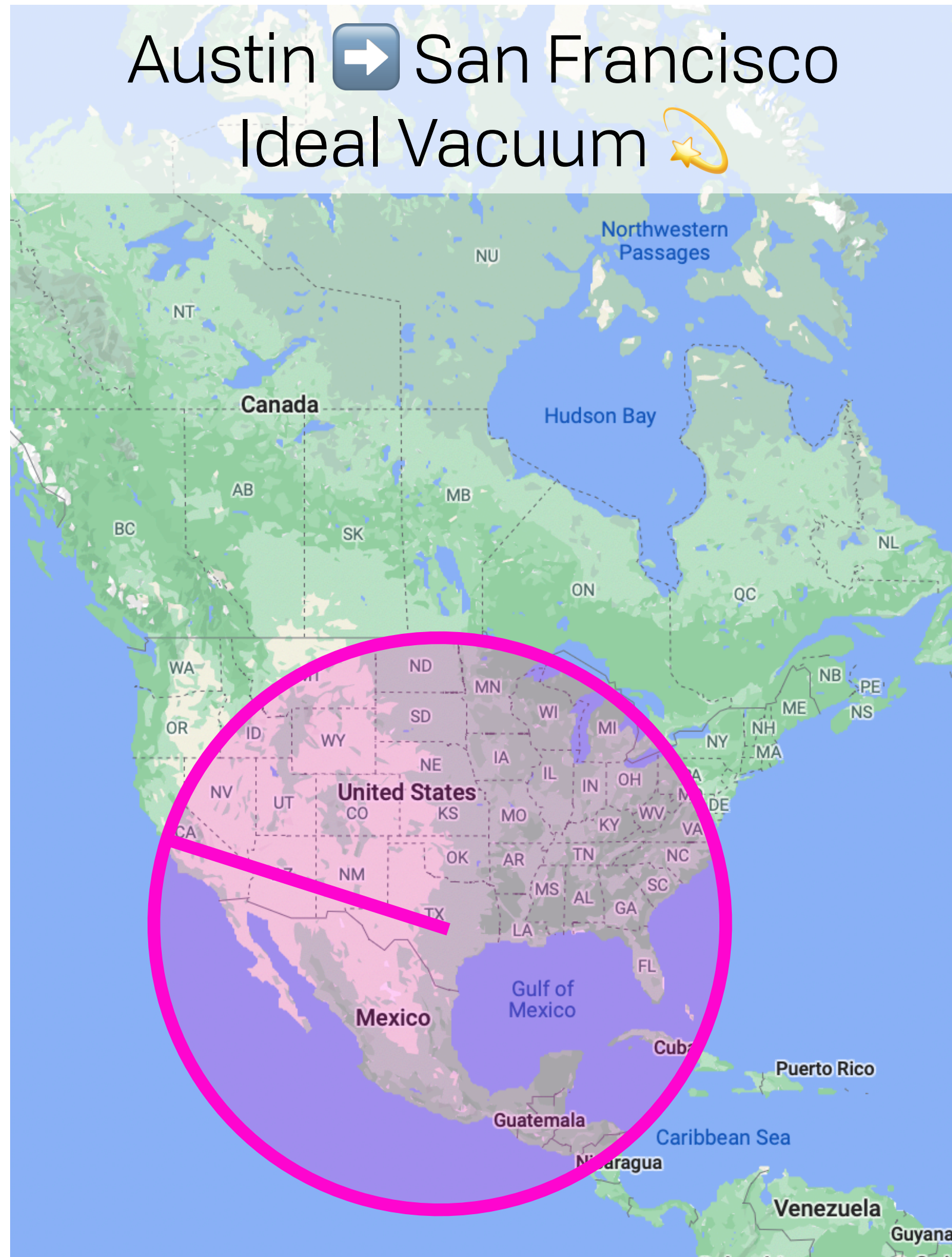
Austin ↻ (almost) Atlanta
Ideal Vacuum 🌟



Low Latency 🐰

What 8ms Looks Like

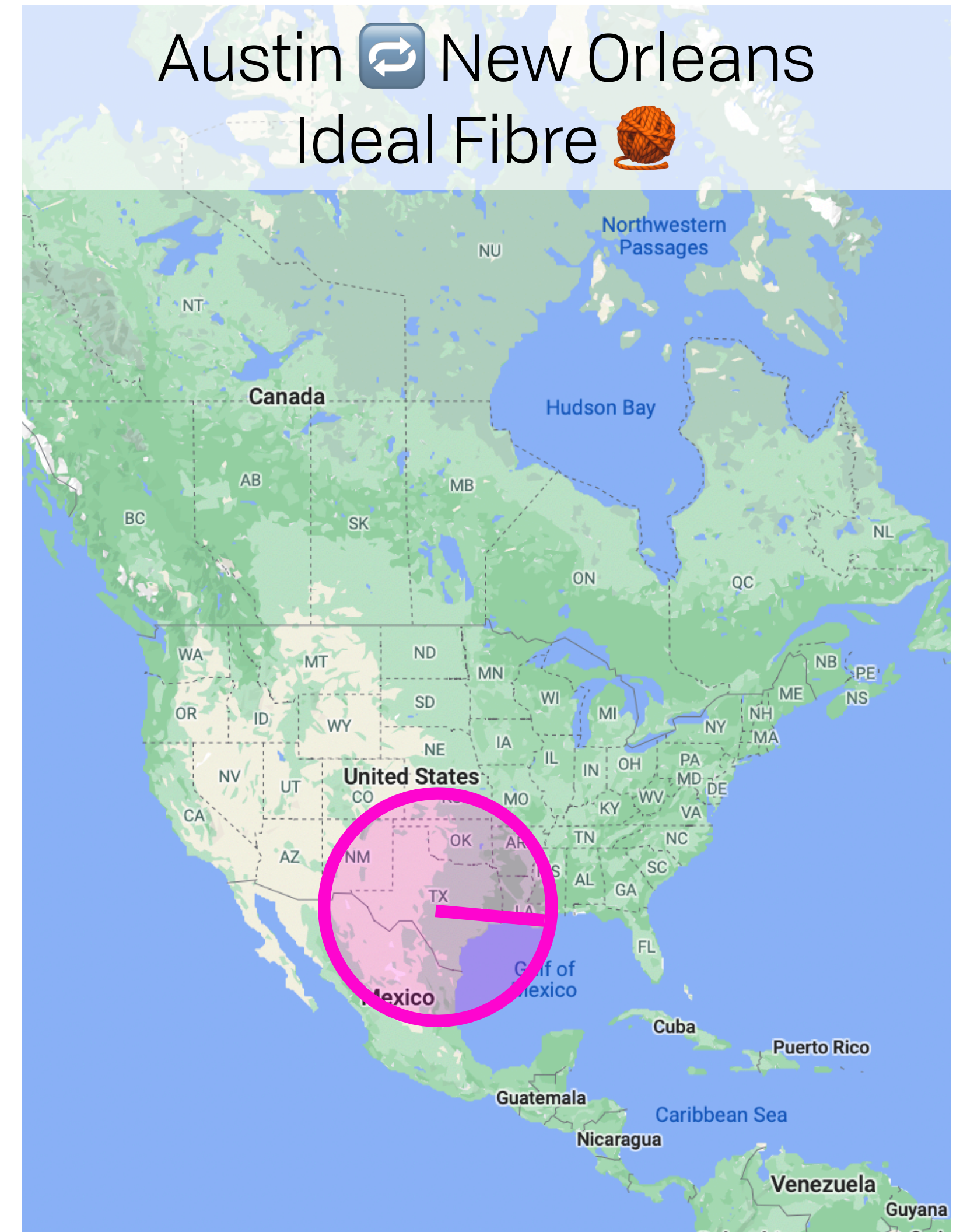
Austin → San Francisco
Ideal Vacuum 🌟



Austin ↔ (almost) Atlanta
Ideal Vacuum 🌟

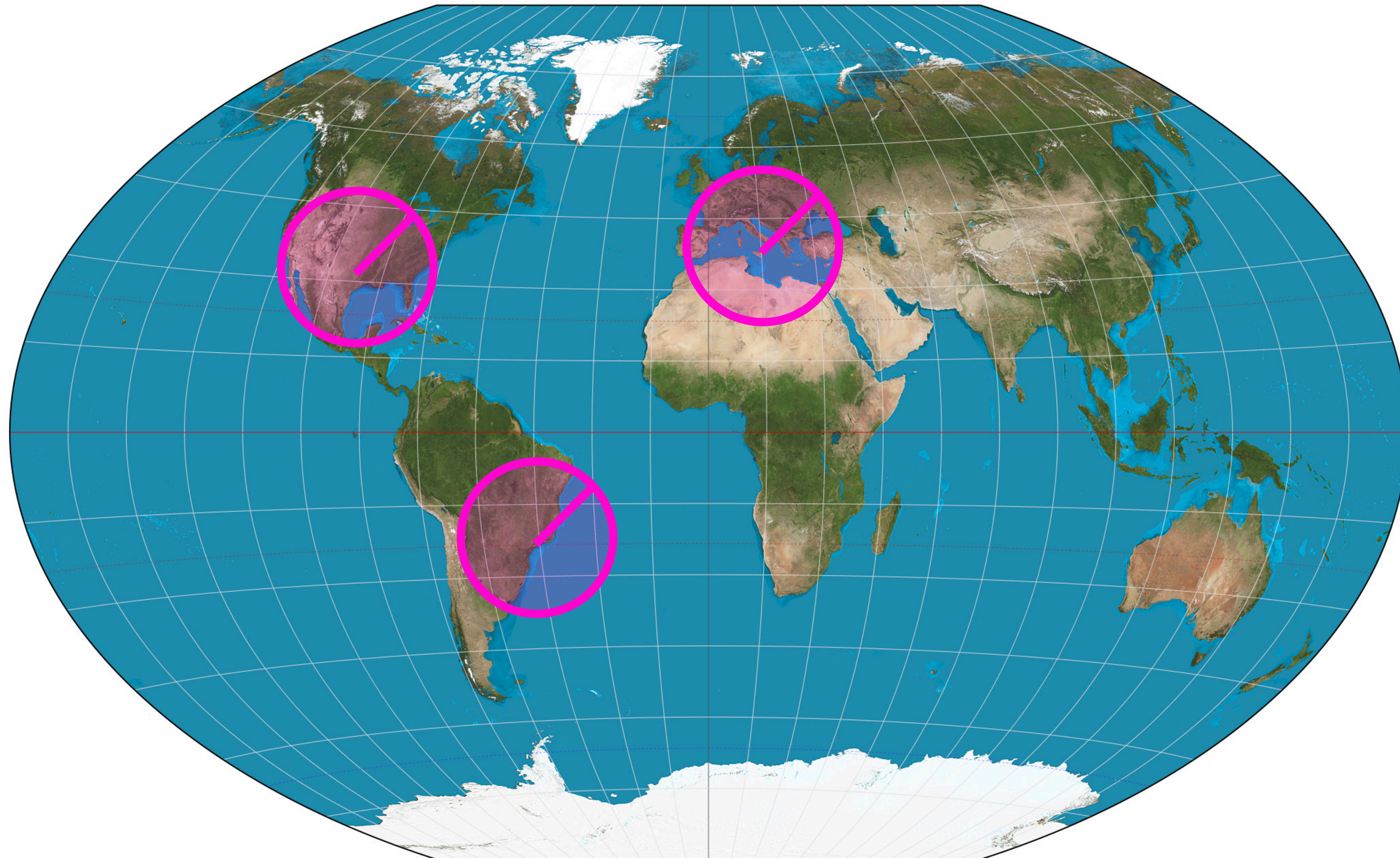


Austin ↔ New Orleans
Ideal Fibre 🧶



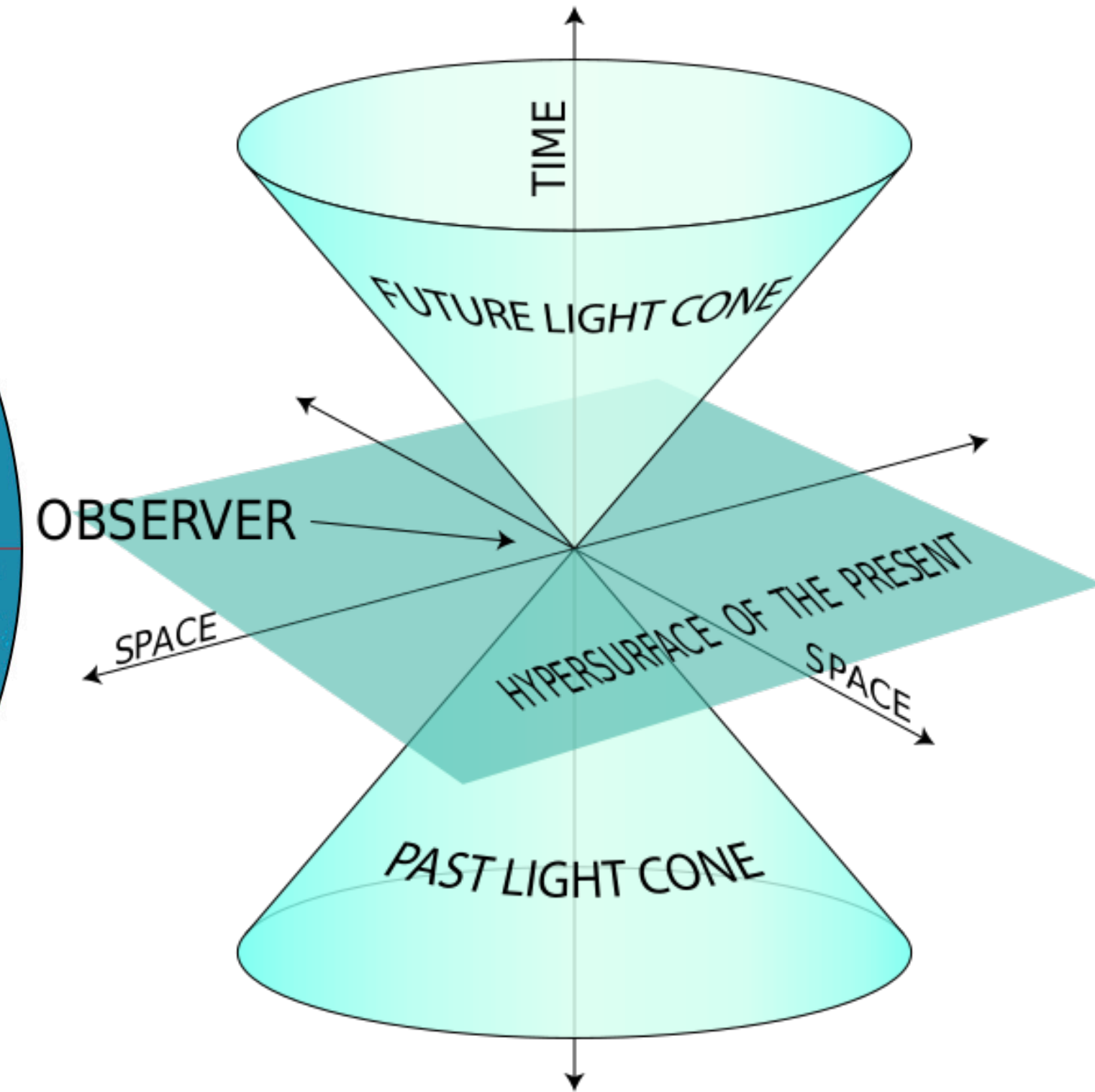
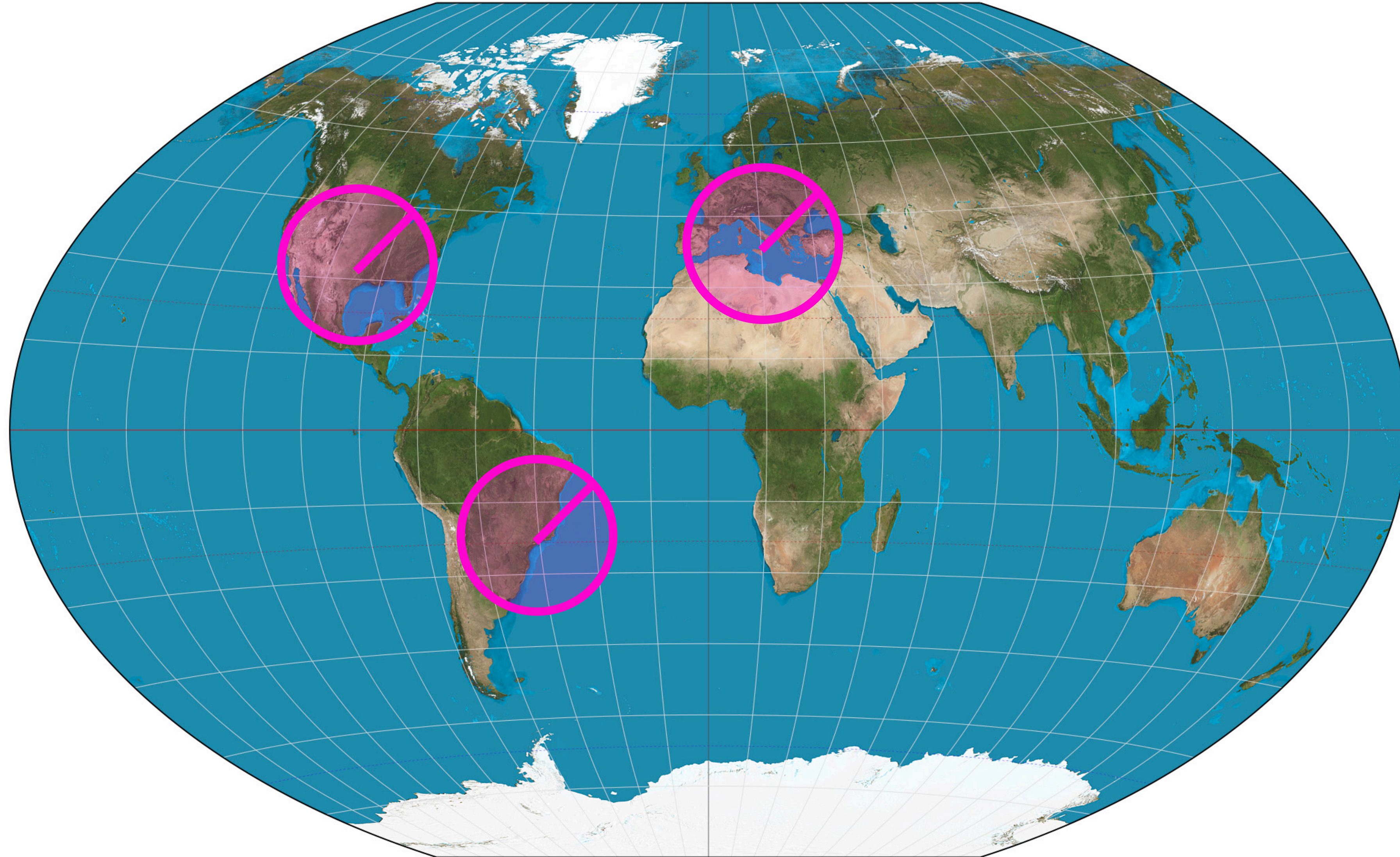
Low Latency 🐰

Causal Islands 🏖️ 🌴



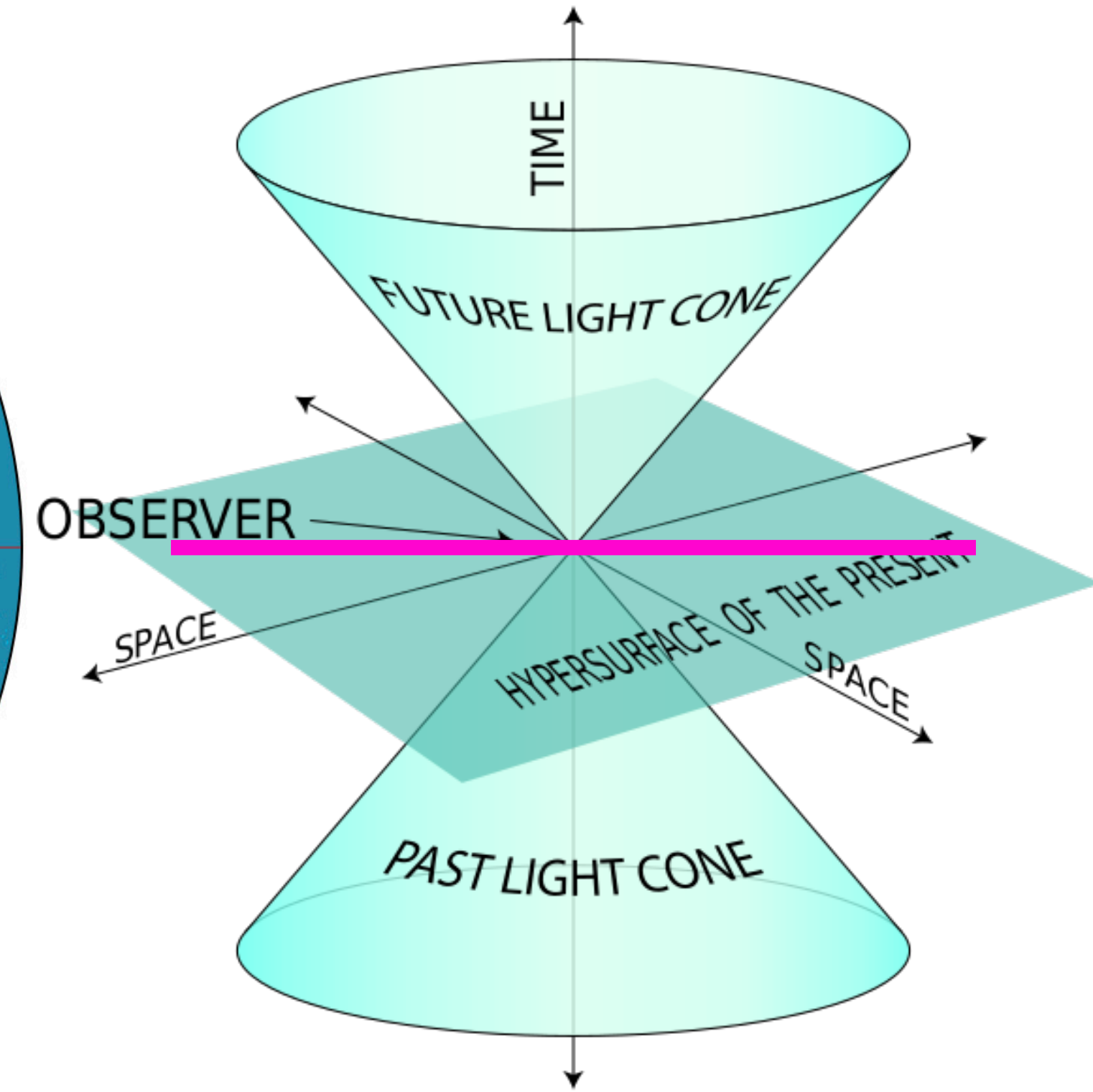
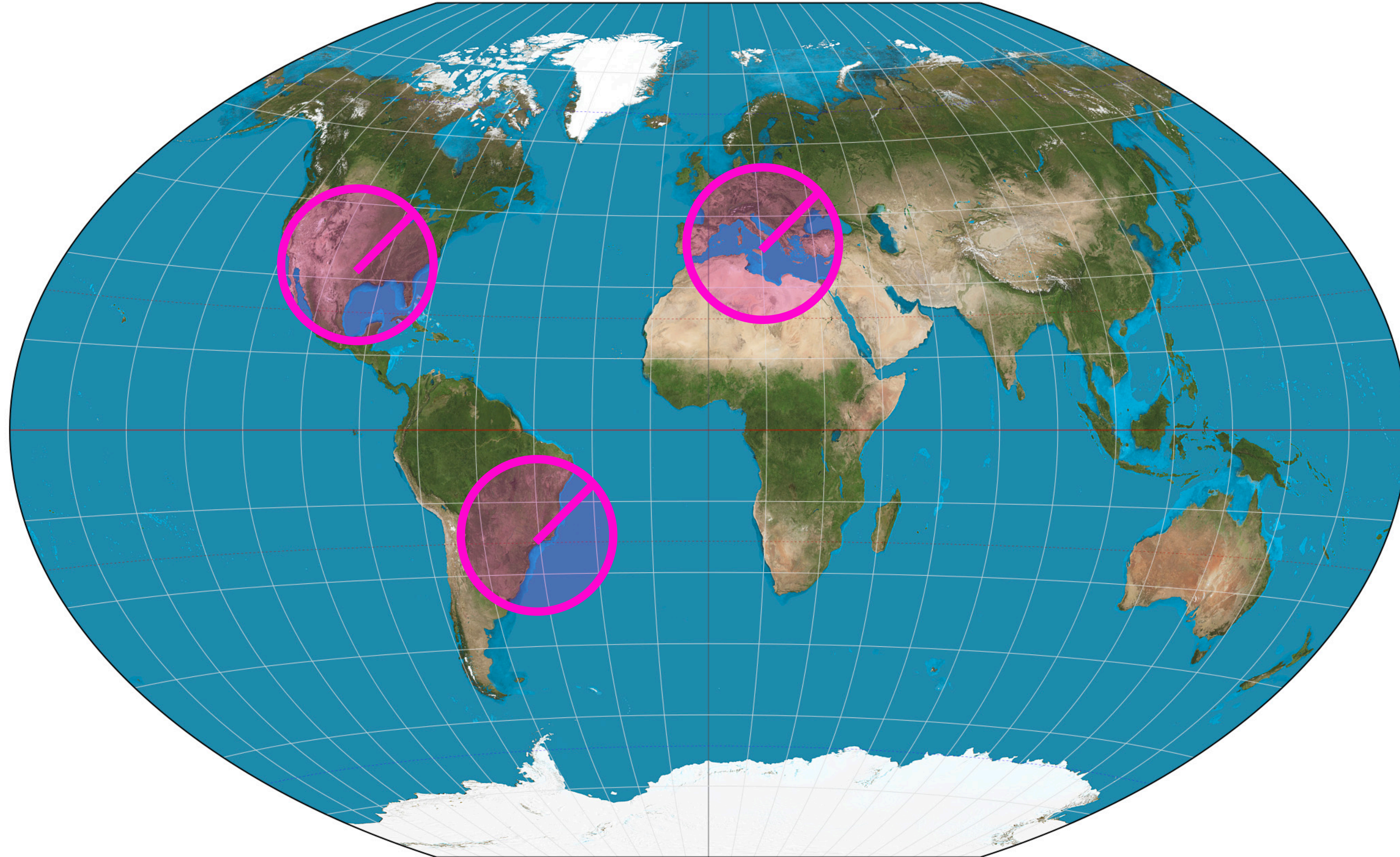
Low Latency 🐰

Causal Islands 🏖️ 🌴



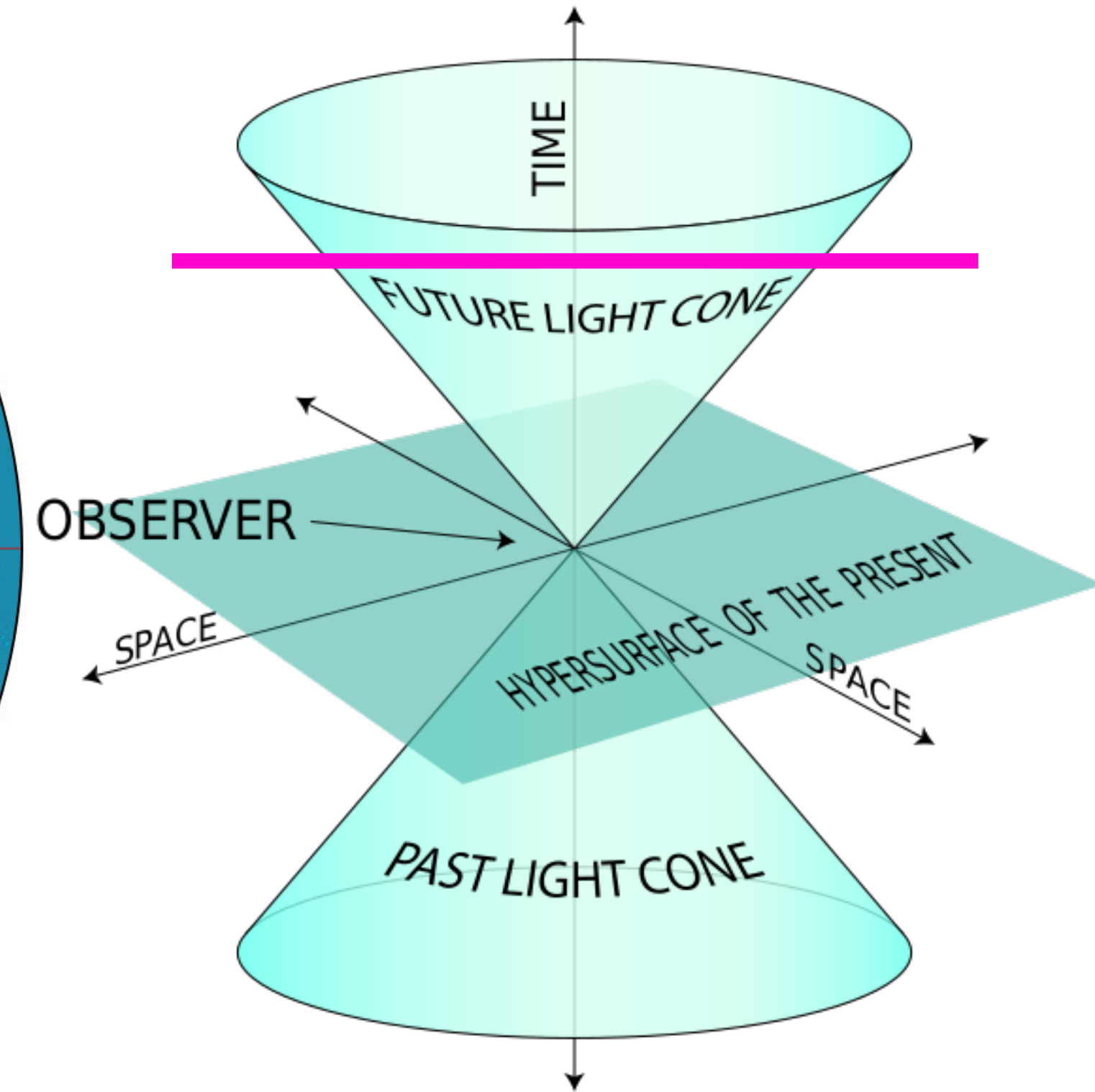
Low Latency 🐰

Causal Islands 🏖️ 🌴



Low Latency 🐰

Causal Islands 🏖️ 🌴






New Environment 

Friendly Neighbourhood Compute

New Environment 




Friendly Neighbourhood Compute

- 5G networks & Starlink 
- Straight line point-to-point
- PoP directly on the tower  



New Environment 

Friendly Neighbourhood Compute

- 5G networks & Starlink 
 - Straight line point-to-point
 - PoP directly on the tower  
- Walmart's Edge
 - 90% Americans < 10mi [16km]



STARLINK



High Volume 

A New Topology

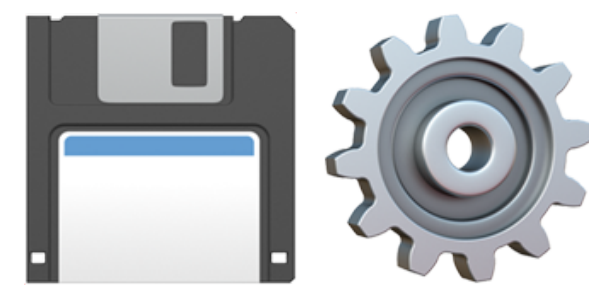
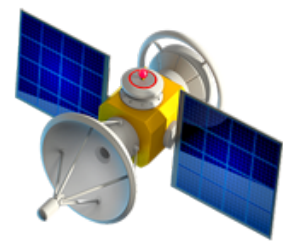
High Volume 

A New Topology



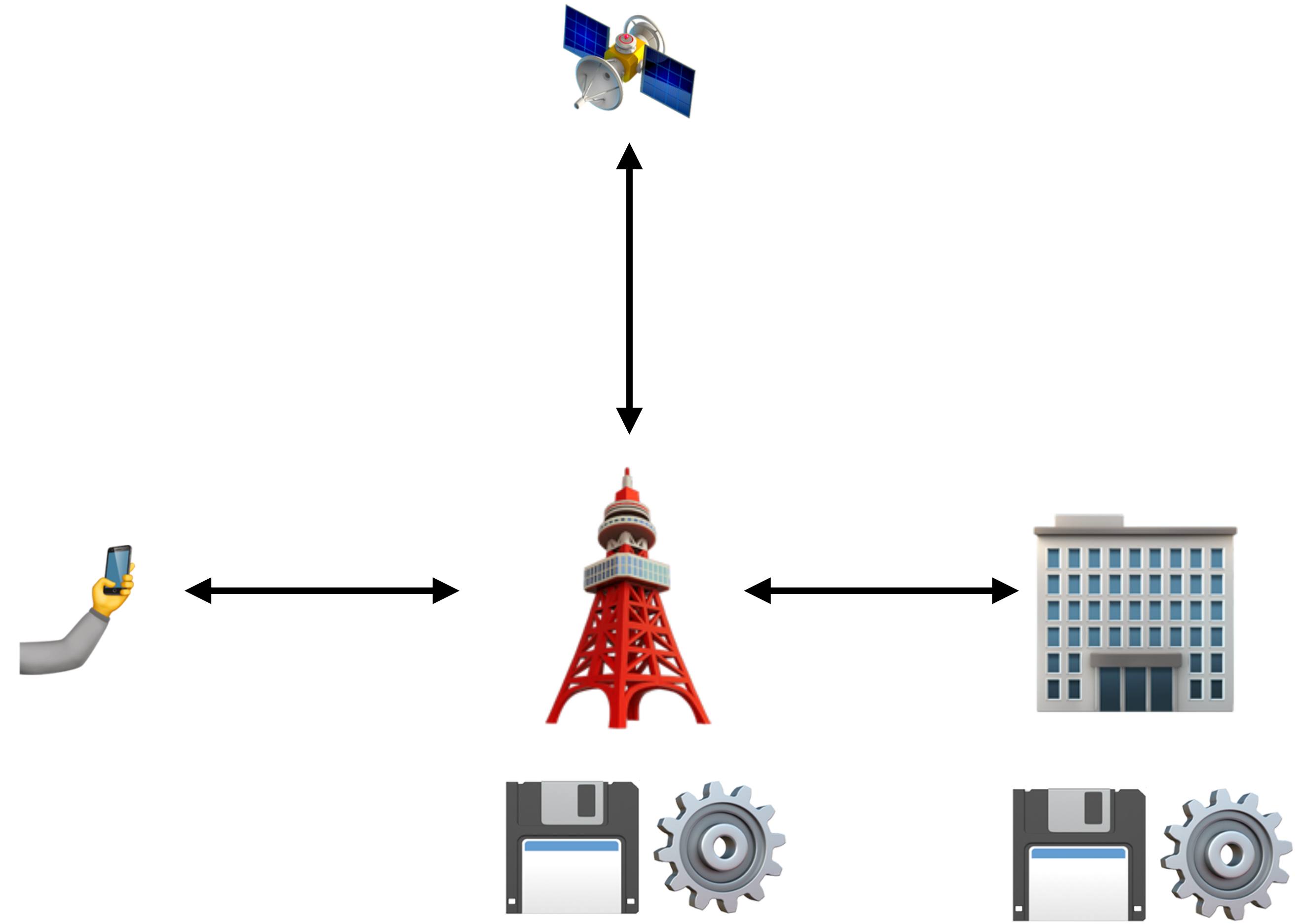
High Volume 📣

A New Topology



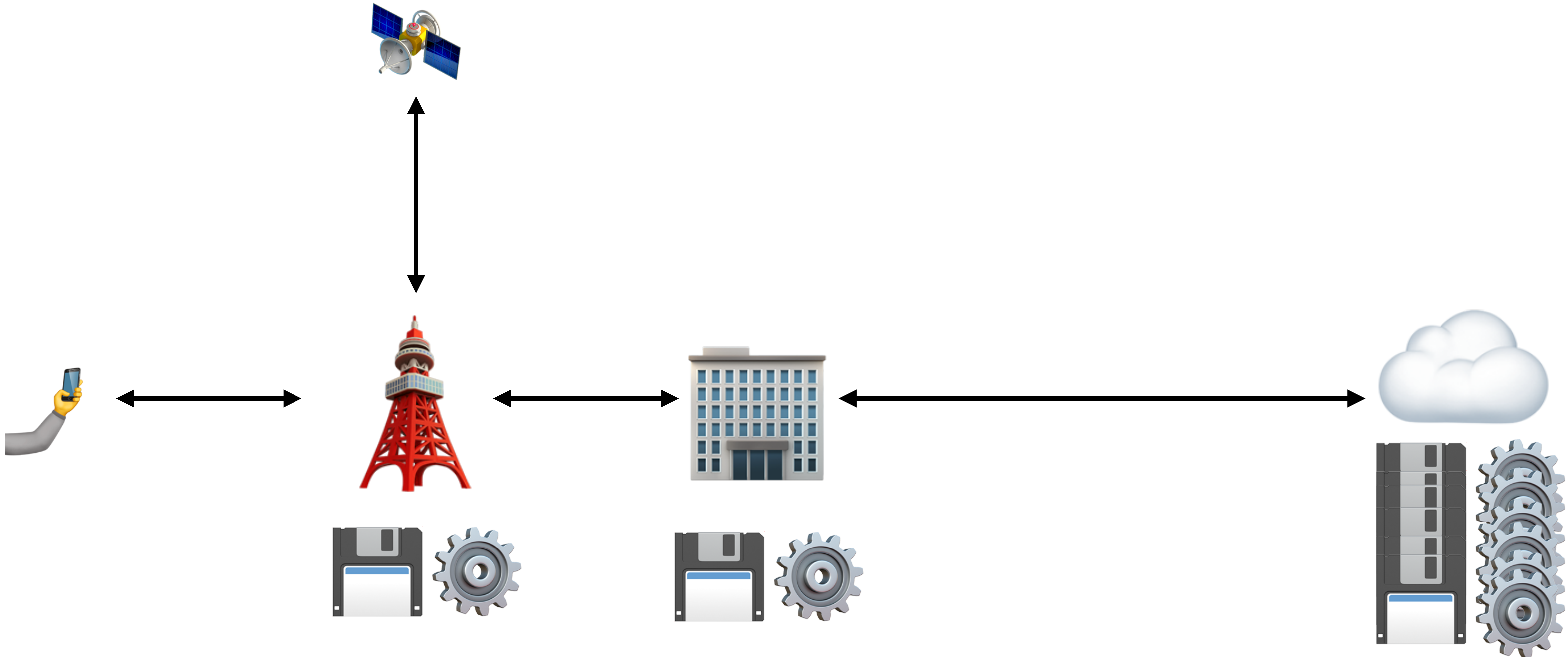
High Volume 📢

A New Topology



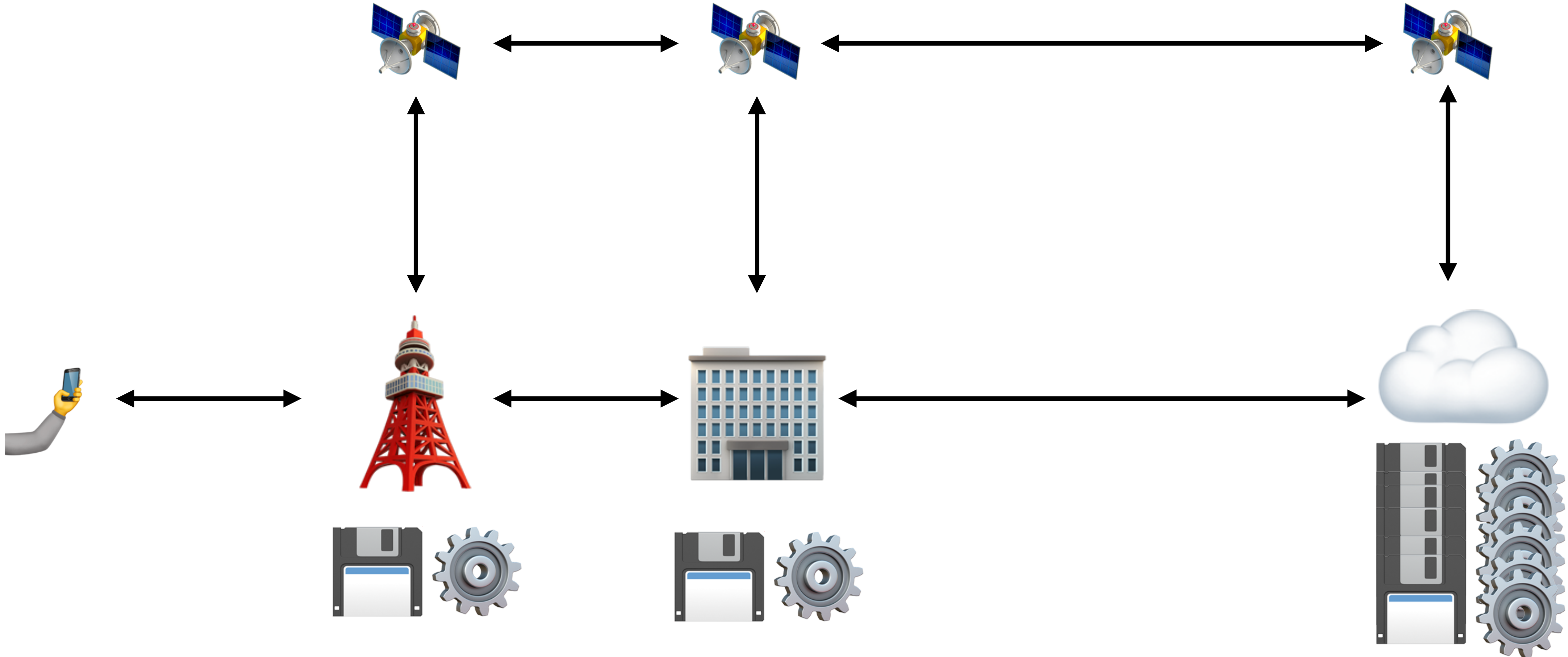
High Volume 📢

A New Topology



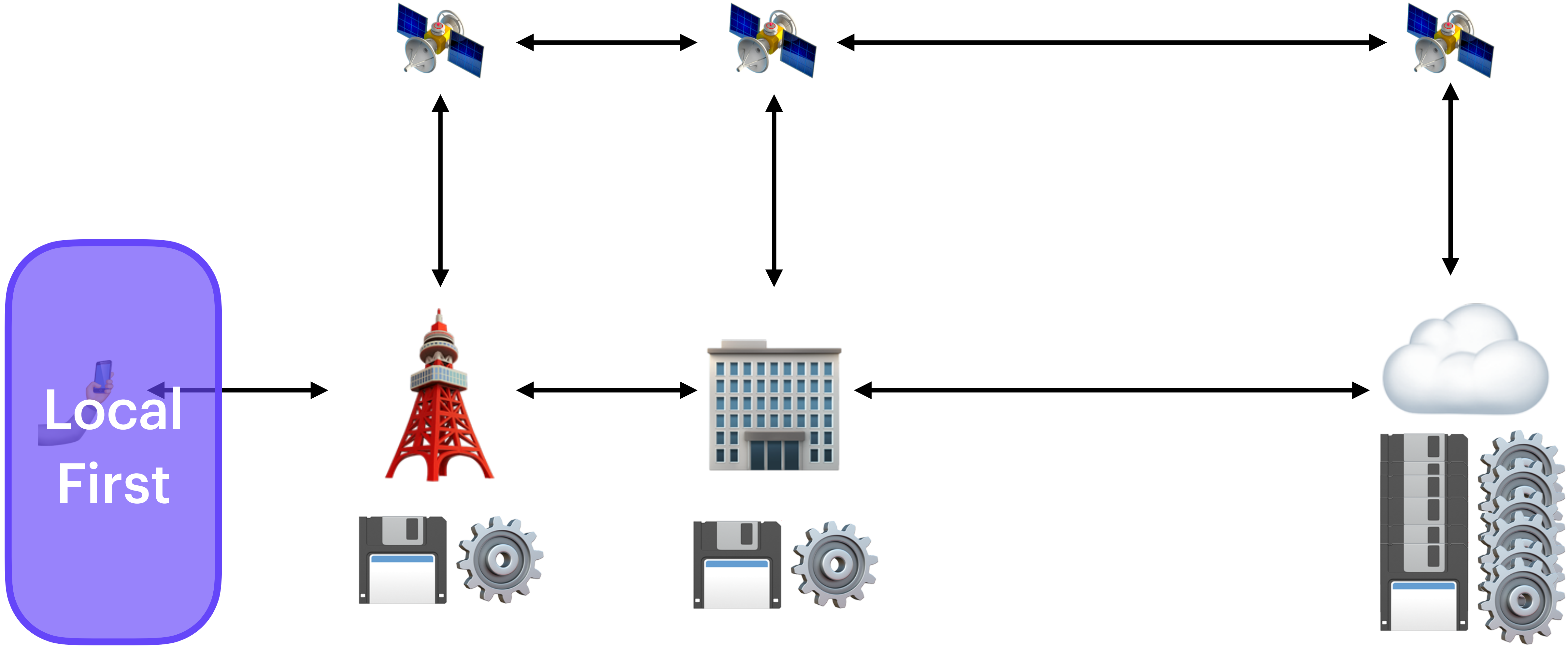
High Volume 

A New Topology



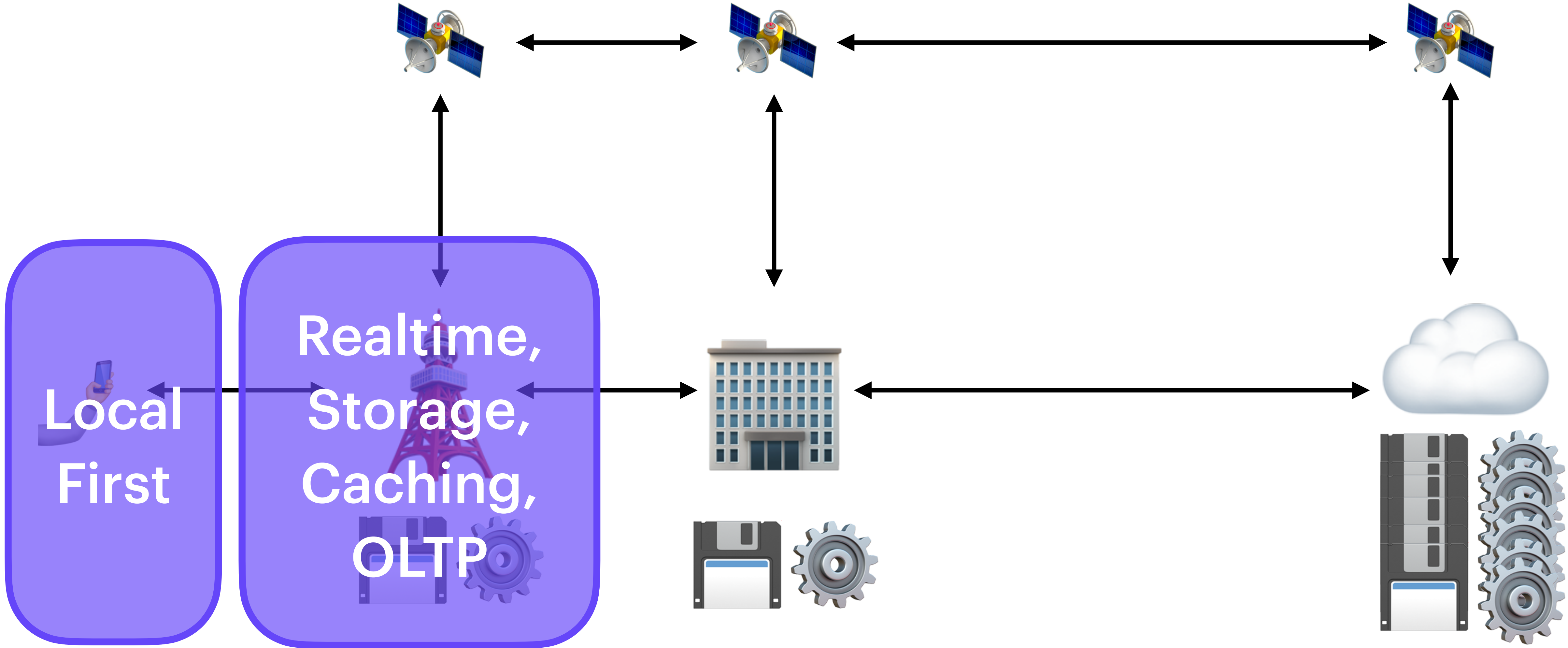
High Volume 

A New Topology



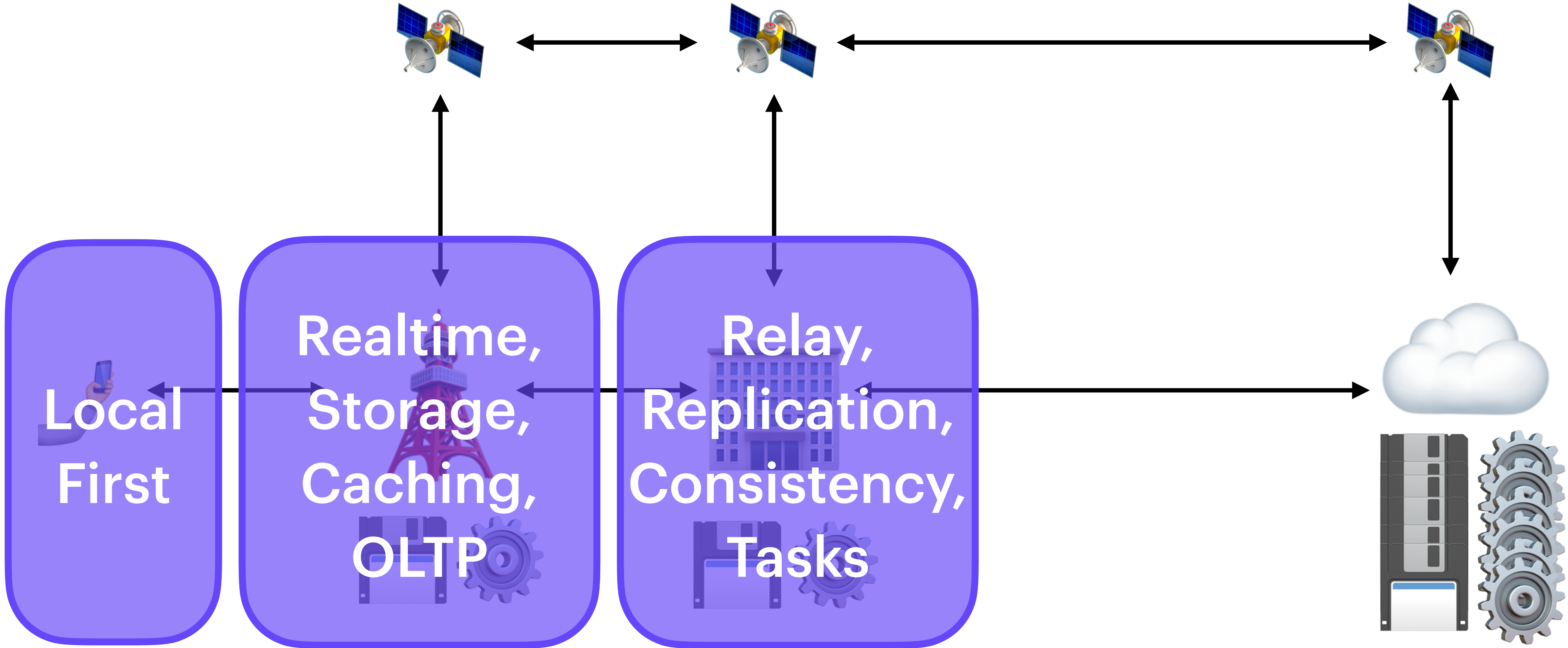
High Volume 

A New Topology



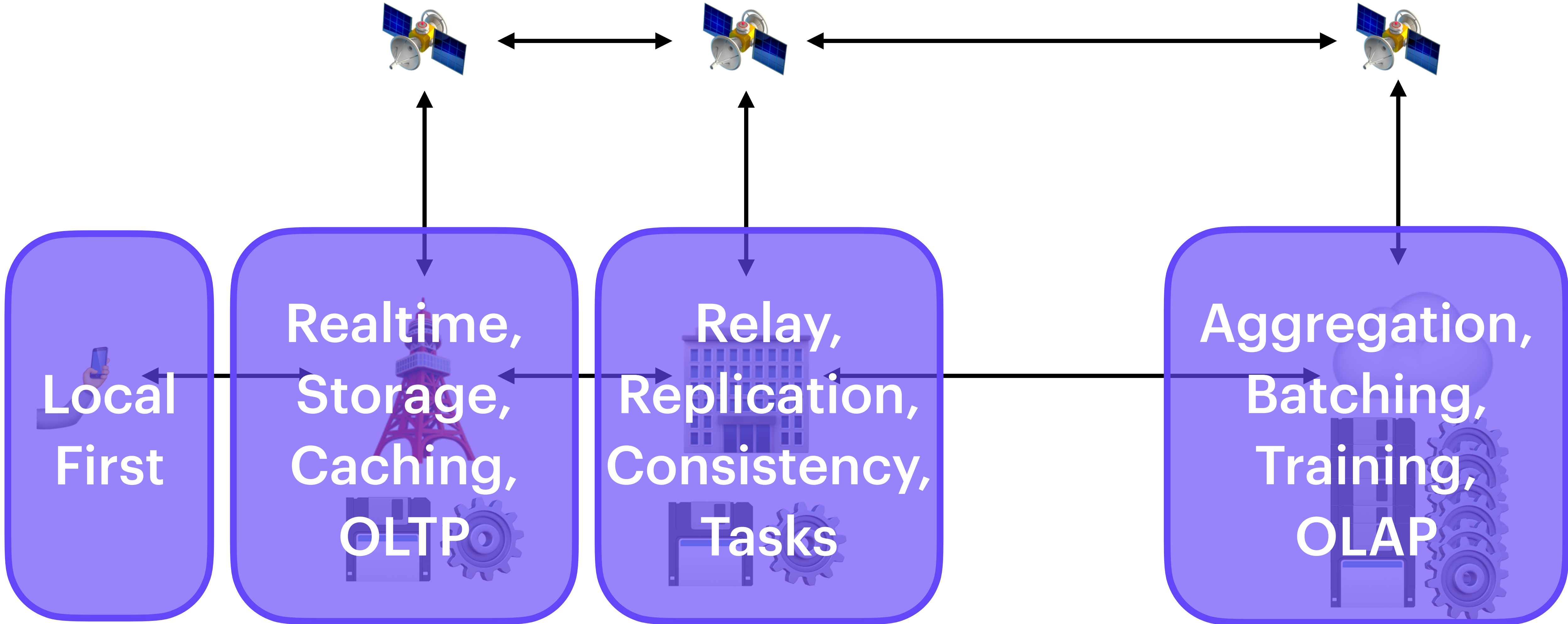
High Volume 

A New Topology



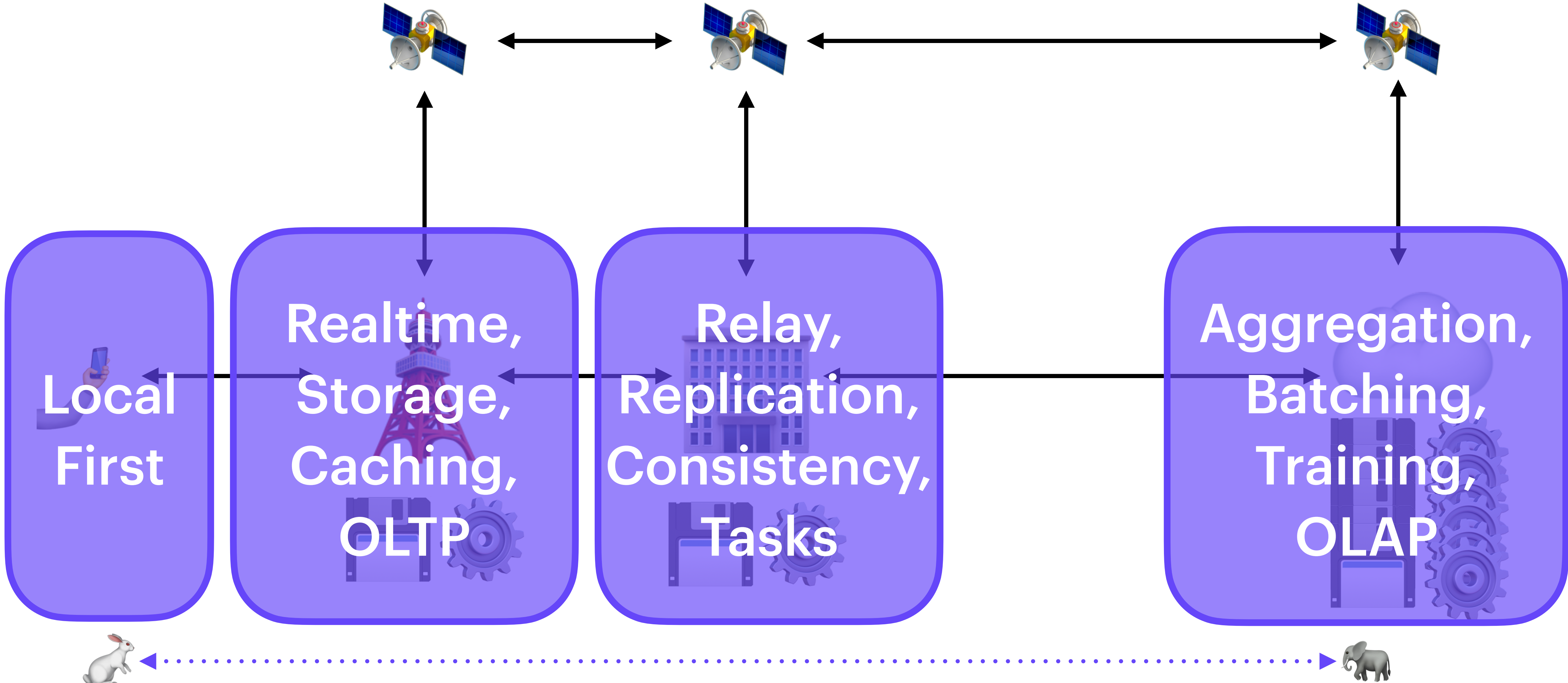
High Volume 

A New Topology



High Volume 

A New Topology



Functional Programming

On the Edge



Instead of [...] “which database would be best to hold presences?”, we could ask “how can we best **replicate data in a distributed system without the user having to worry about it?**”

With Elixir, **you are empowered** to tackle problems that in other platforms would feel impossible to solve

~ Chris McCord, What Makes Phoenix Presence Special

On the Edge



Phoenix LiveView

On the Edge 

Phoenix LiveView

Users 

Client 

WSS / REST / GraphQL 

Controller Logic 

Data Store 

DevOps 

Developer 



On the Edge 

Phoenix LiveView

Users 

Client 

WSS / REST / GraphQL 

Controller Logic 

Data Store 

DevOps 

Developer 



On the Edge 

Phoenix LiveView

Users 

Client 

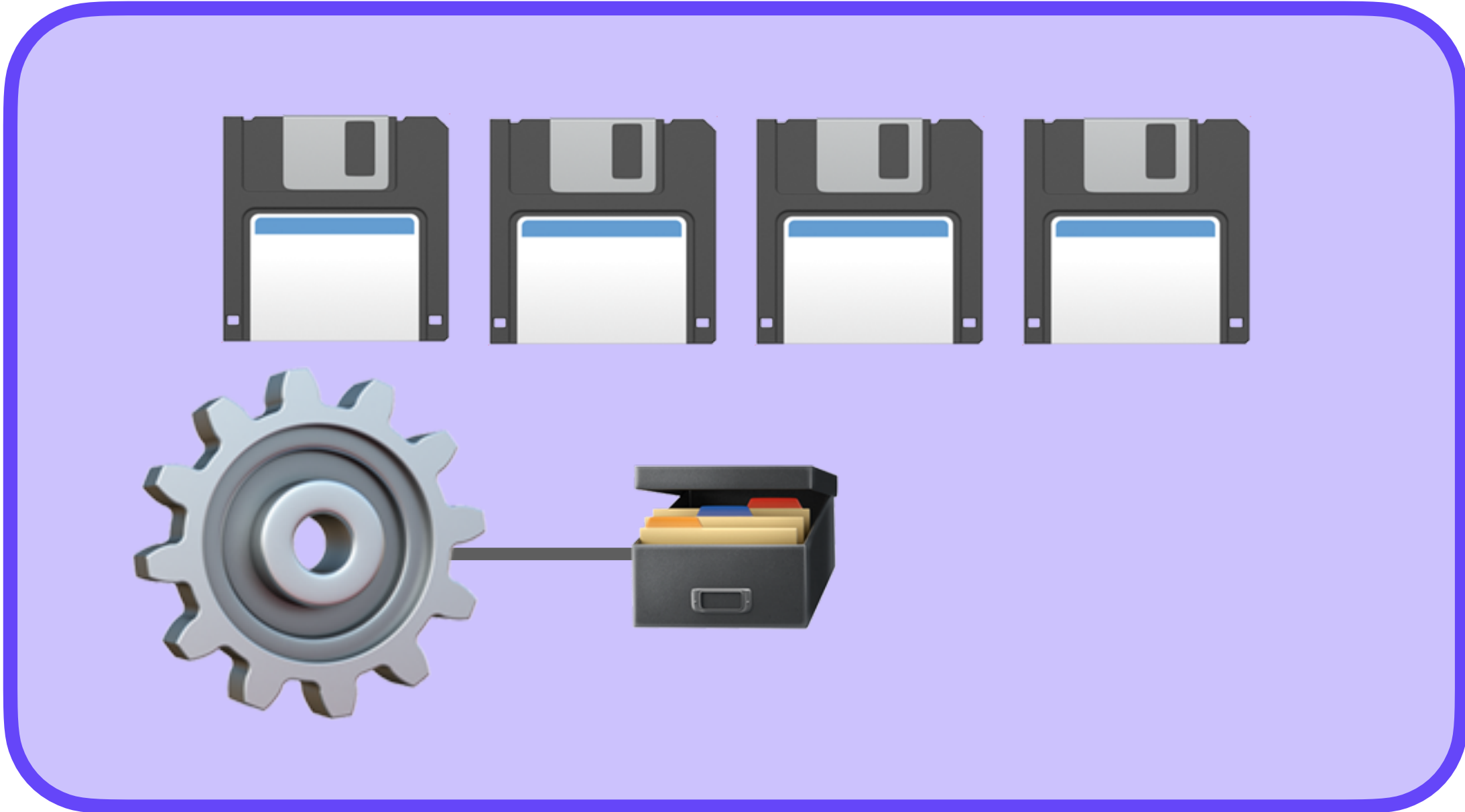
WSS / REST / GraphQL 

Controller Logic 

Data Store 

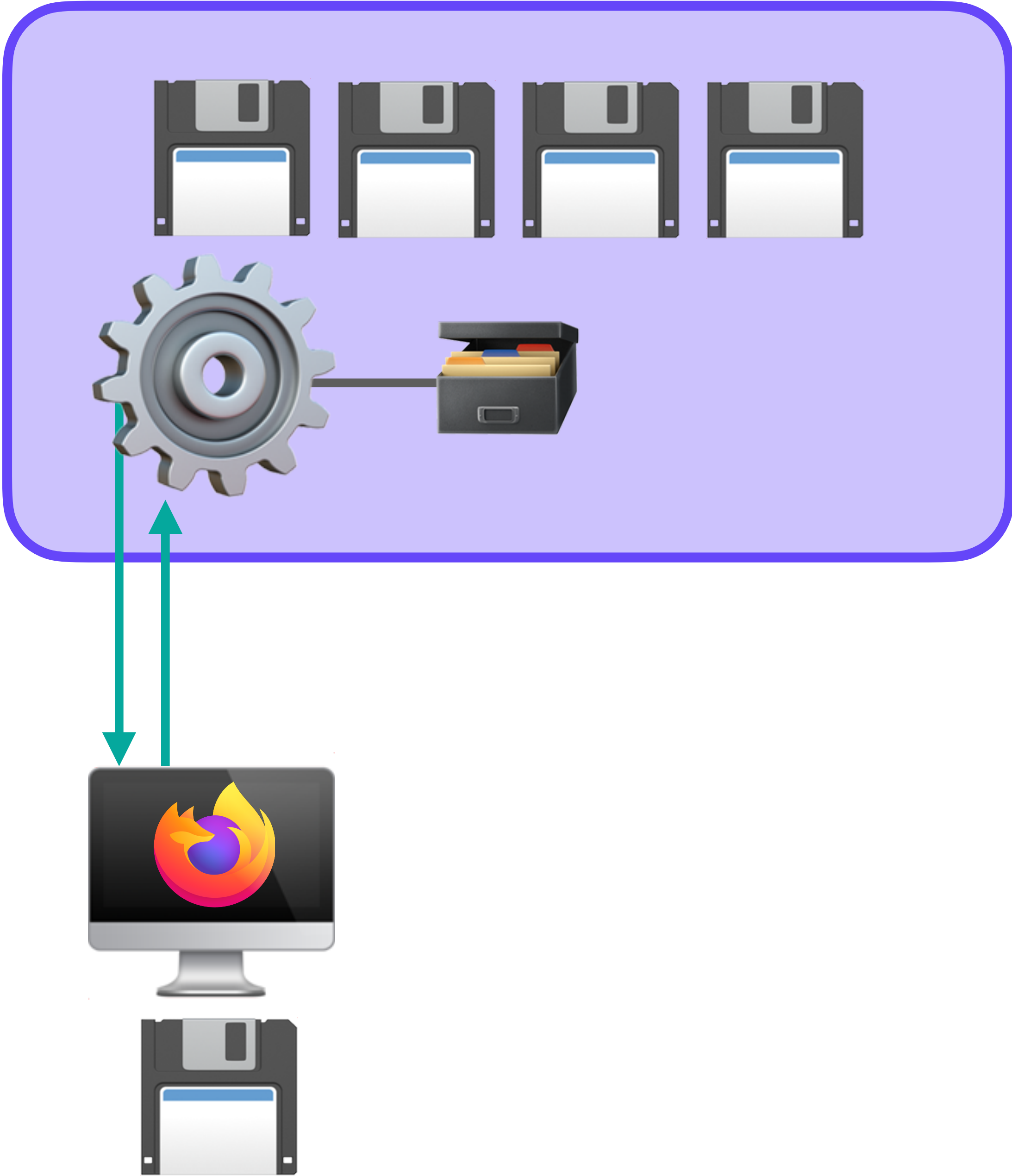
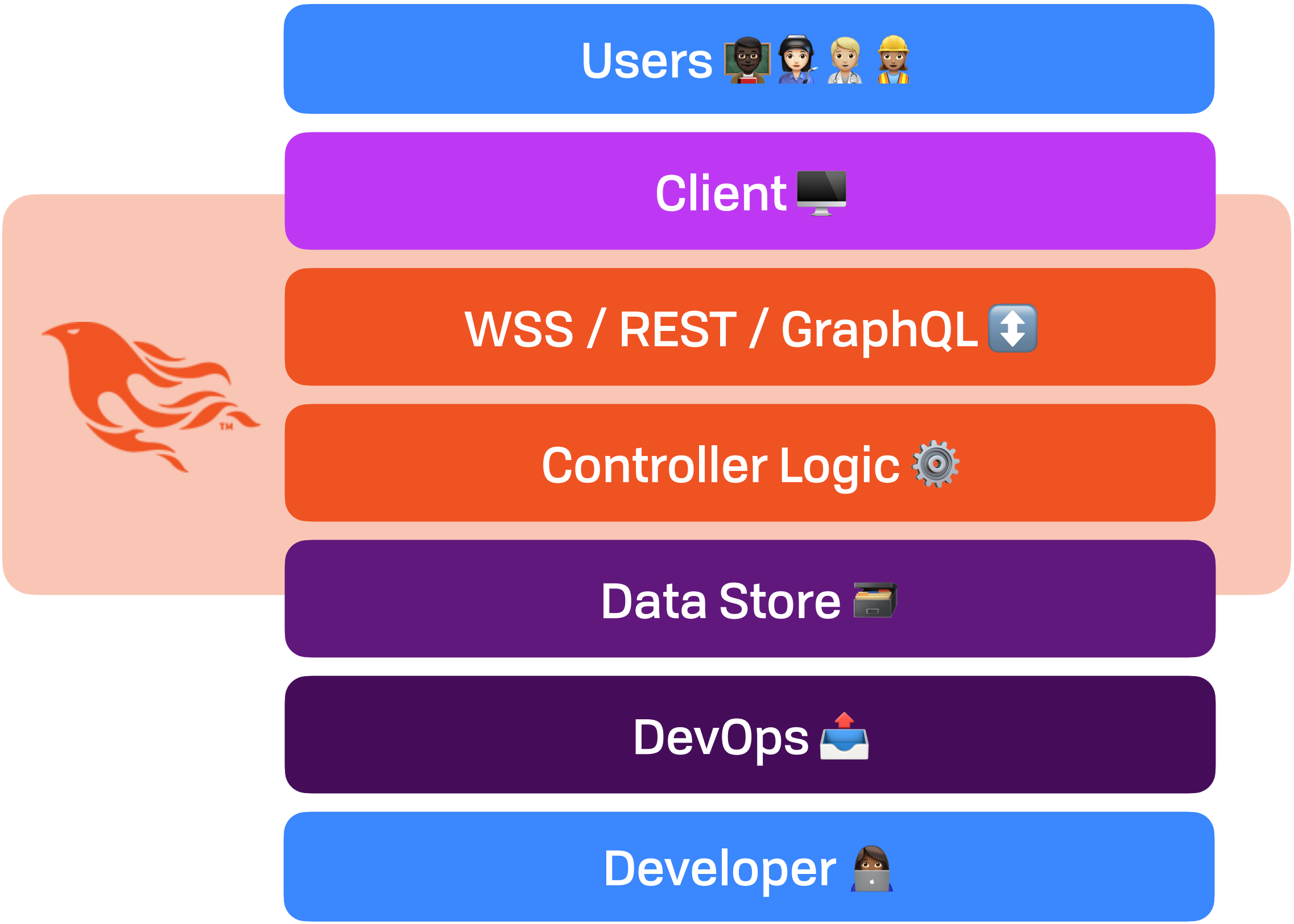
DevOps 

Developer 



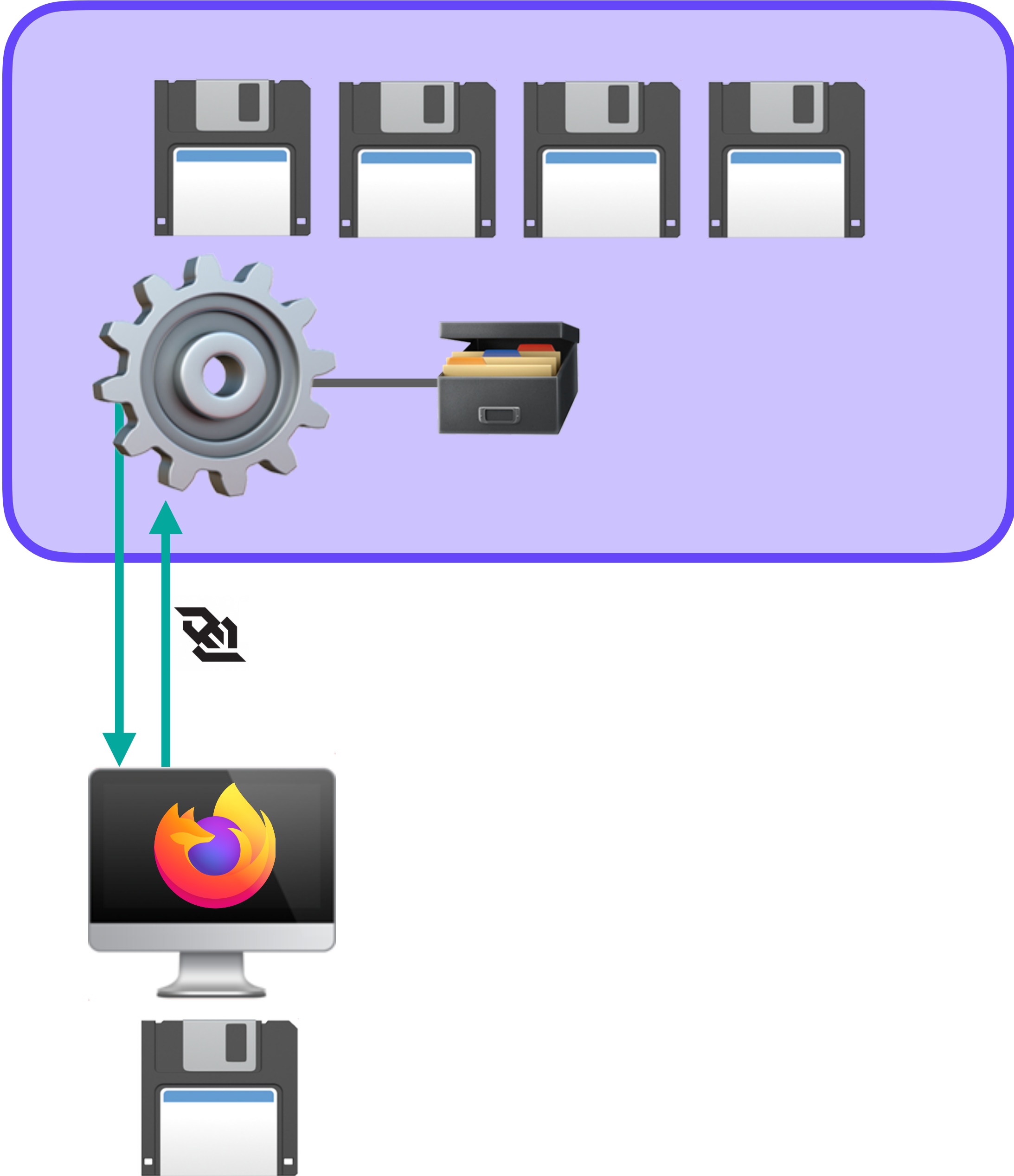
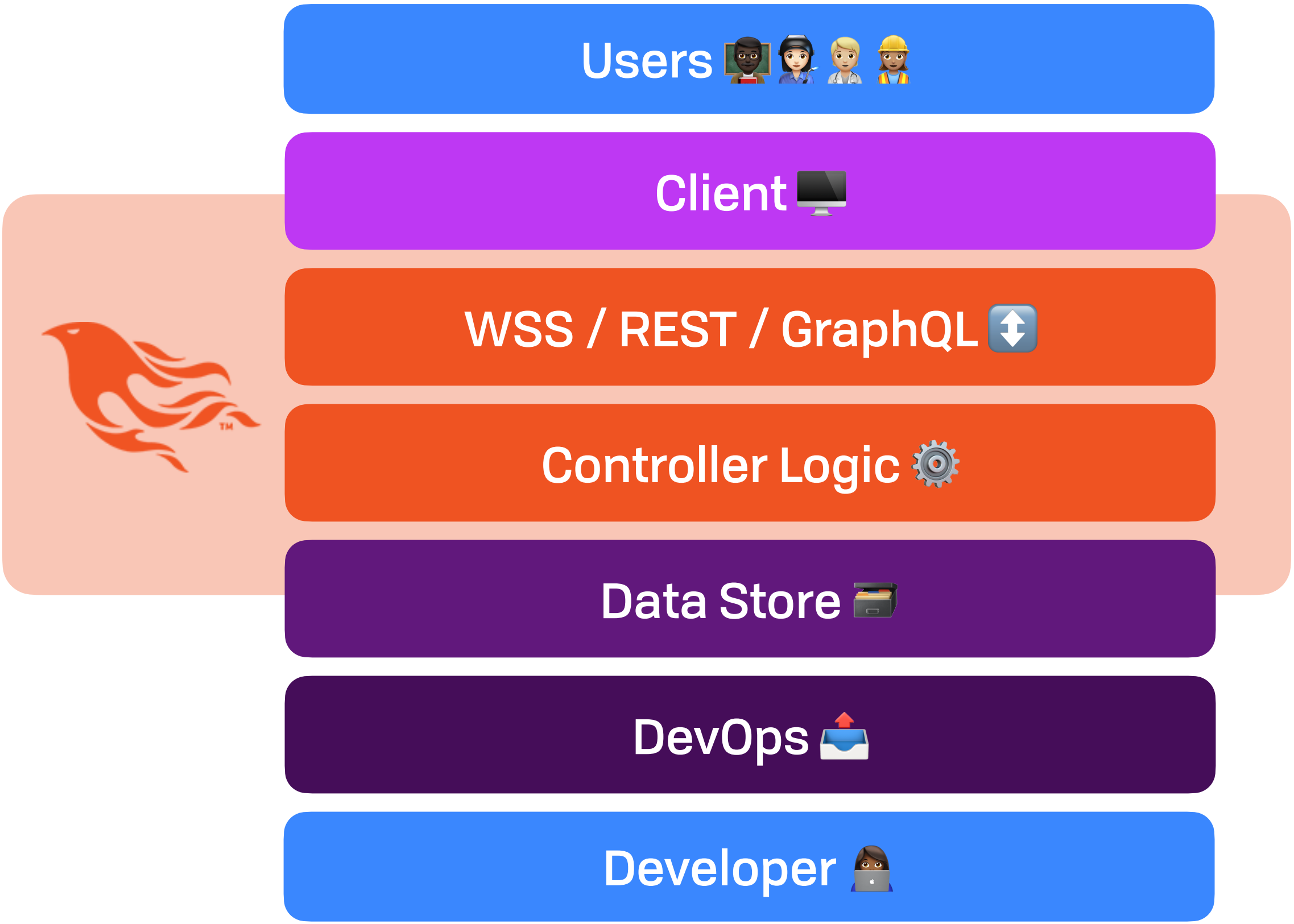
On the Edge 

Phoenix LiveView



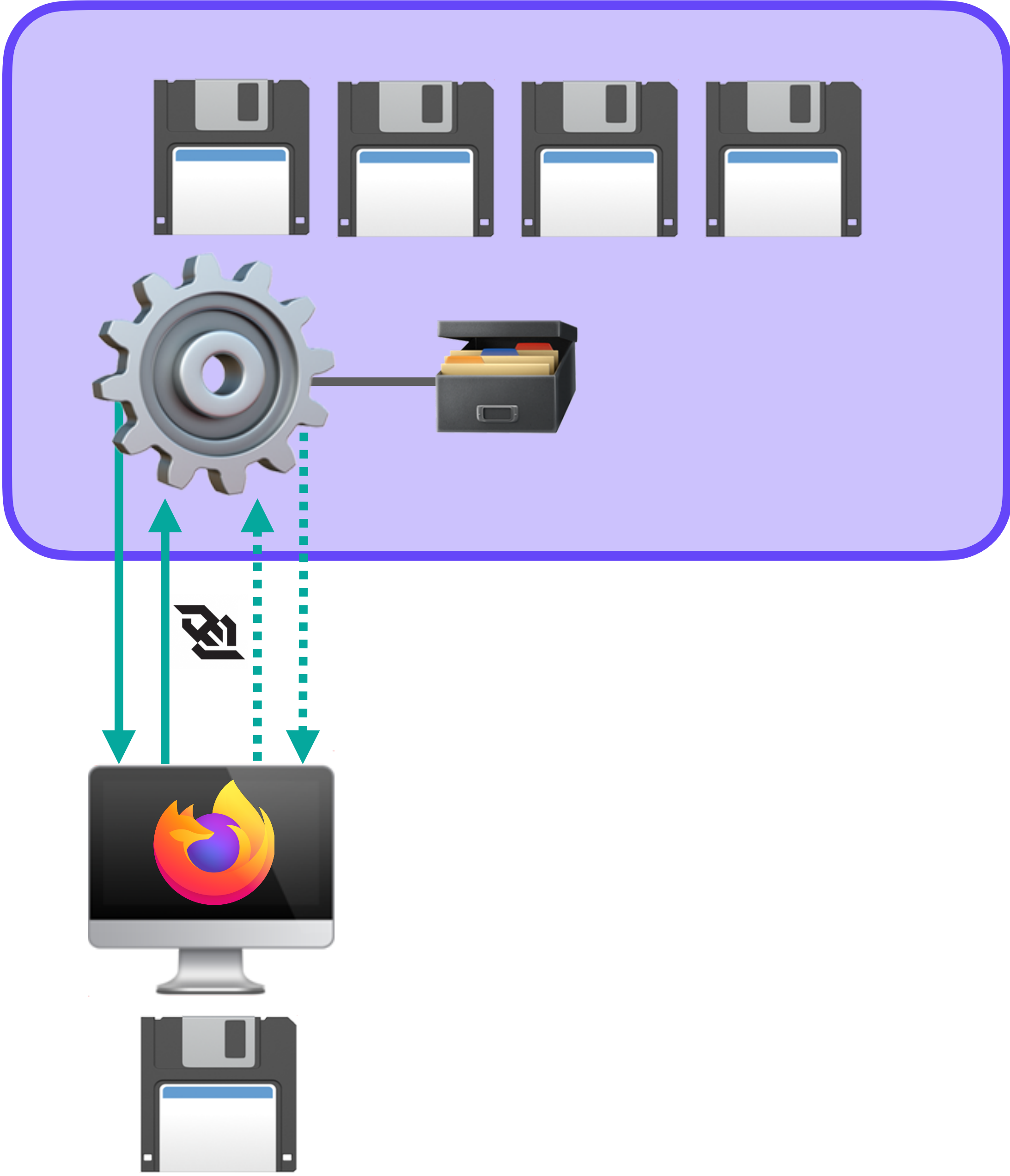
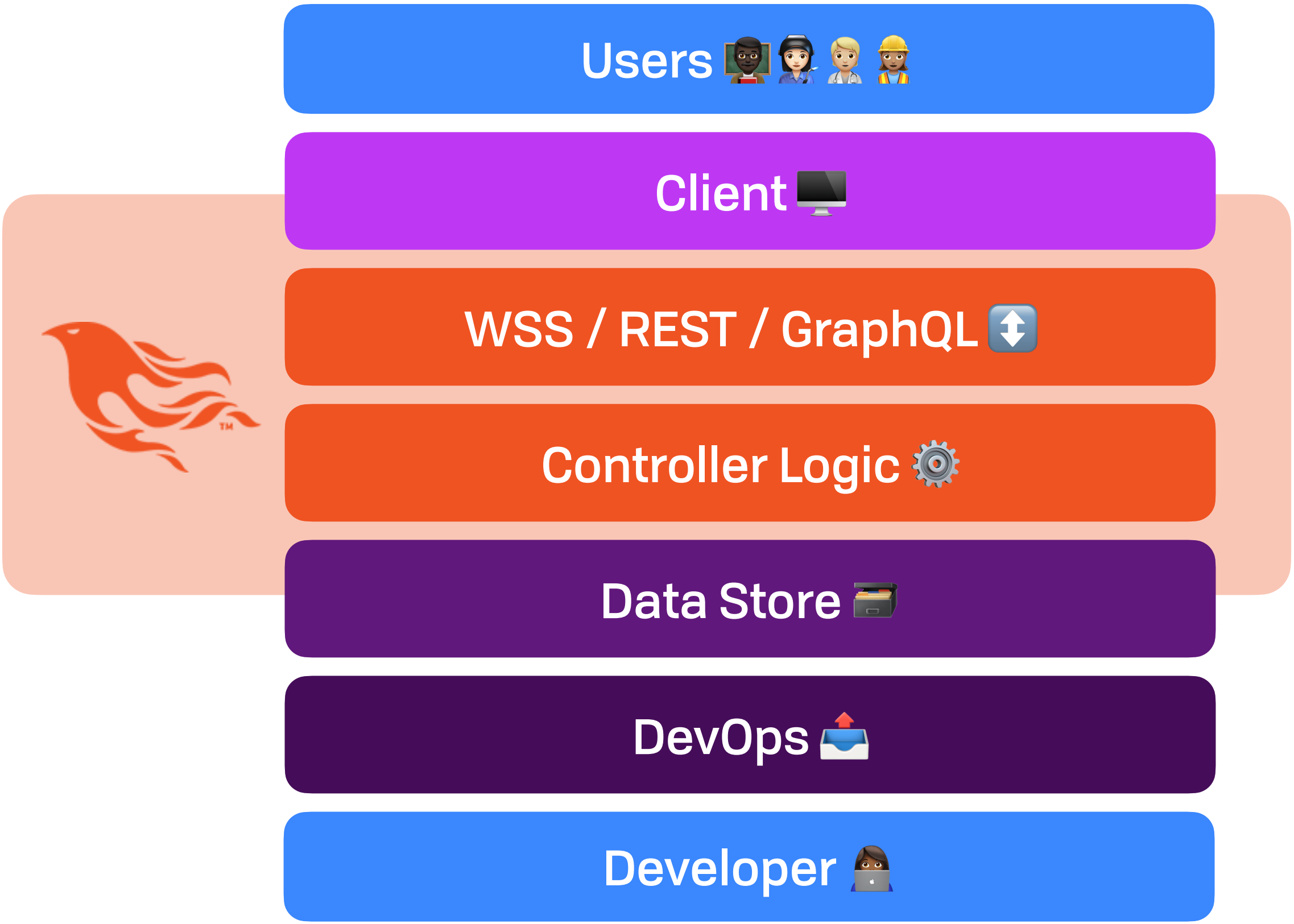
On the Edge 

Phoenix LiveView



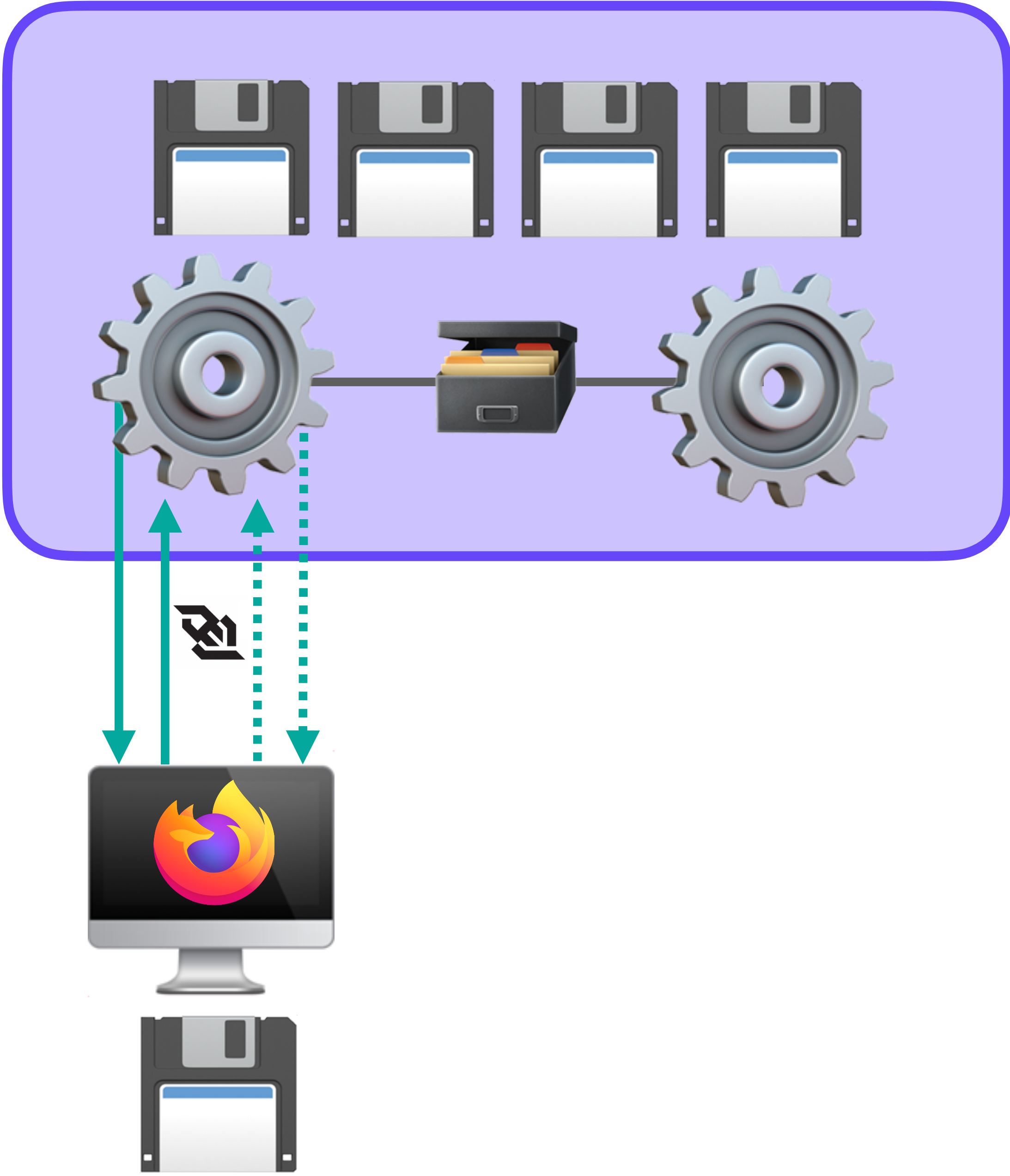
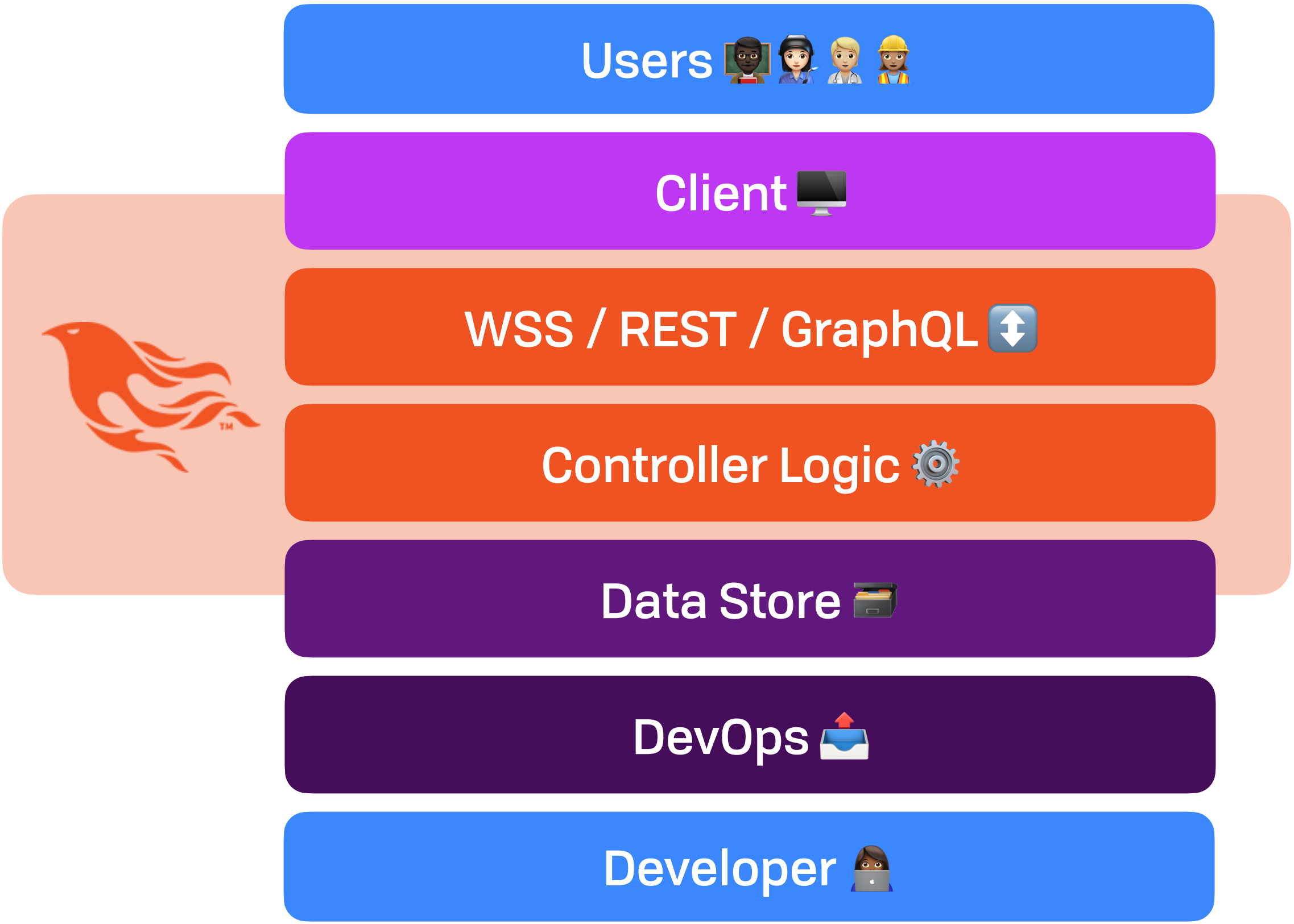
On the Edge 

Phoenix LiveView



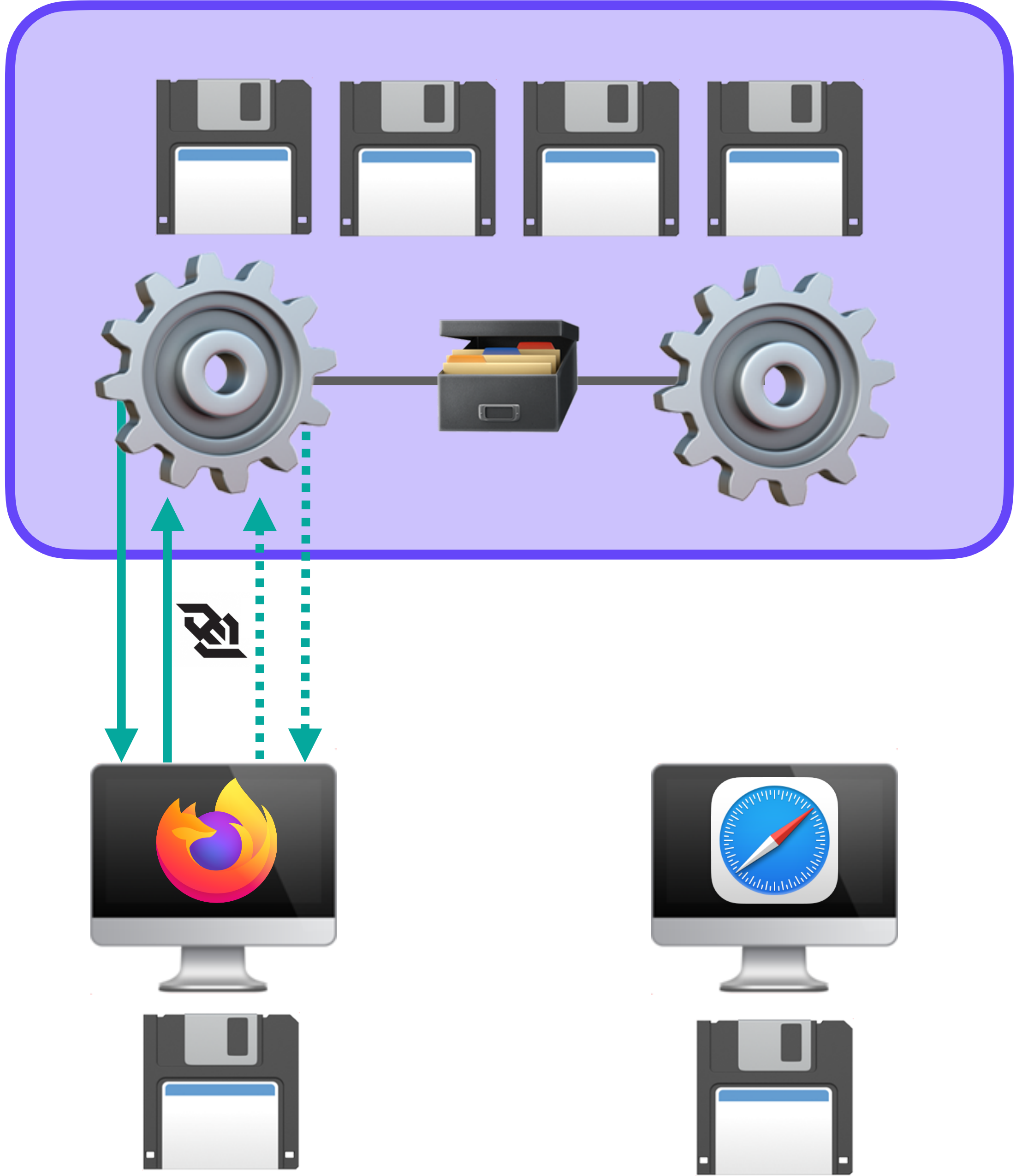
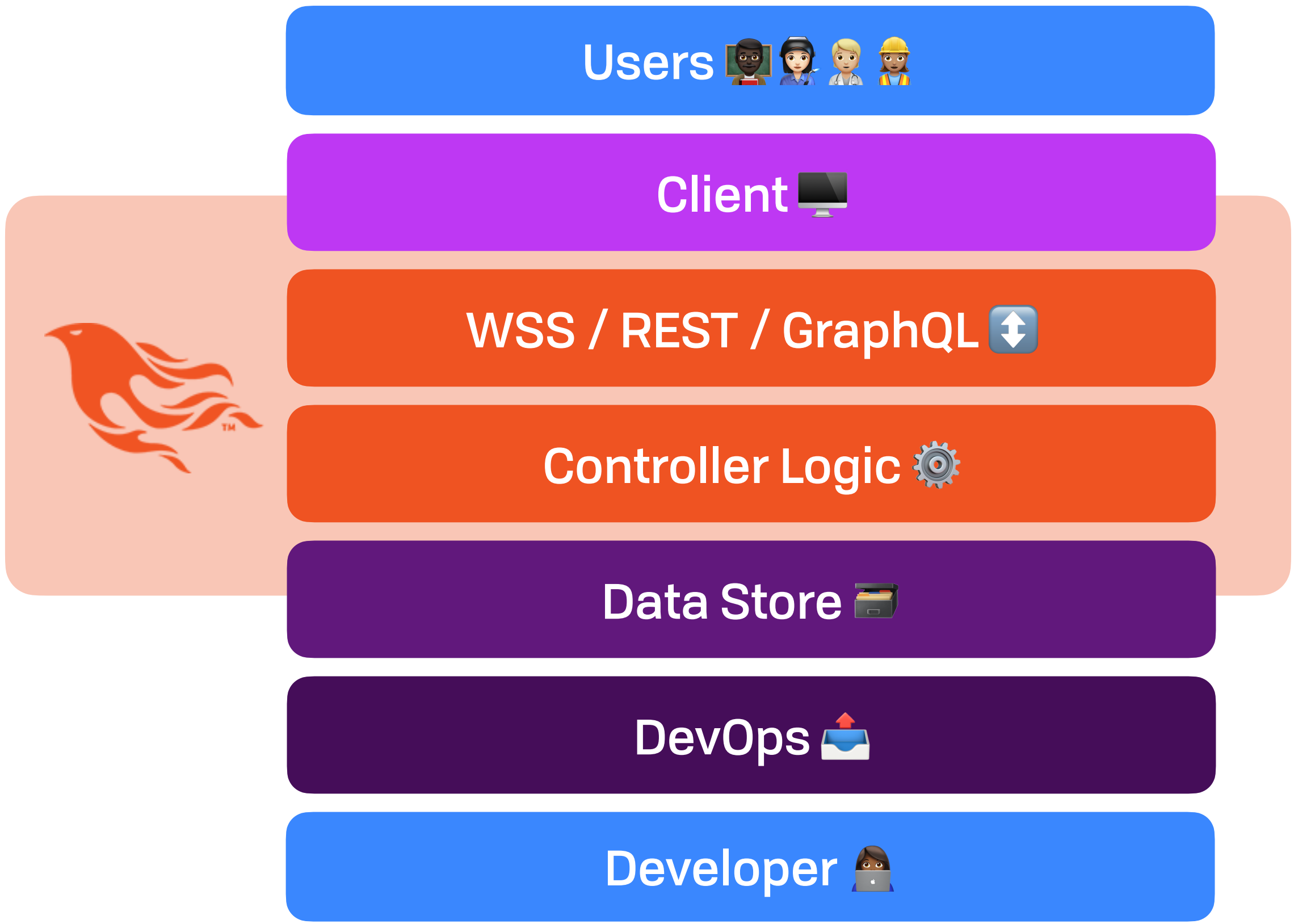
On the Edge 

Phoenix LiveView



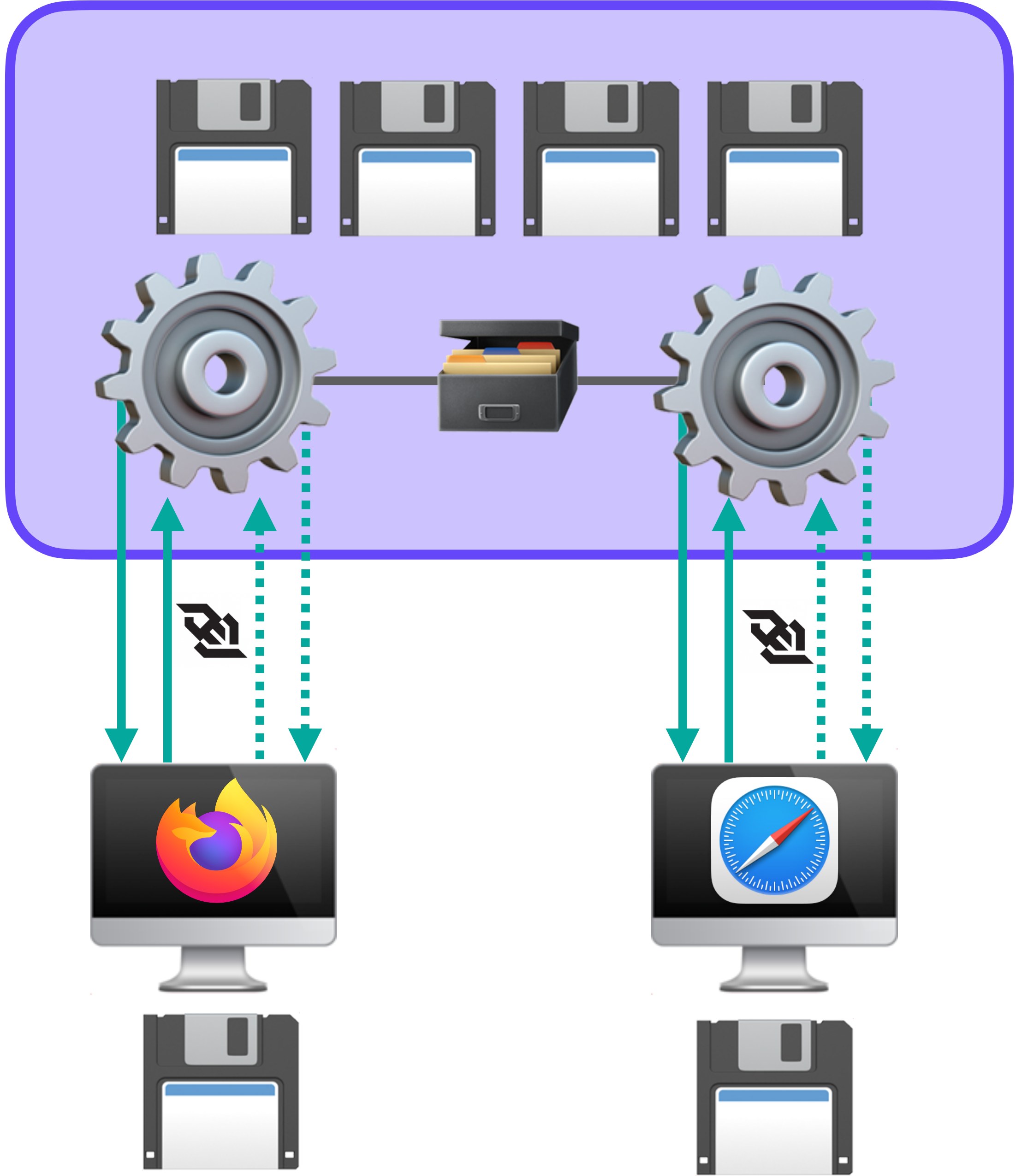
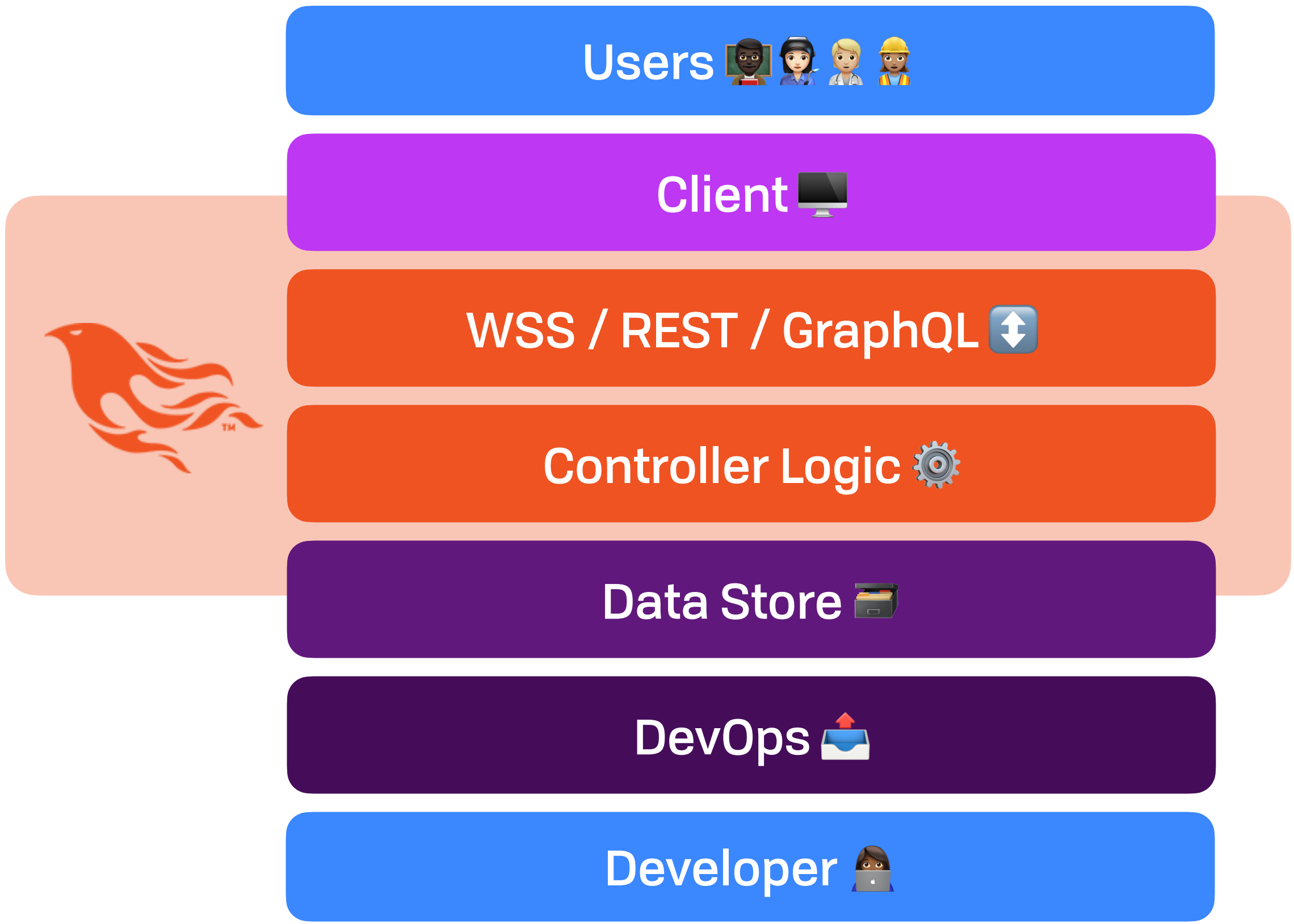
On the Edge 

Phoenix LiveView



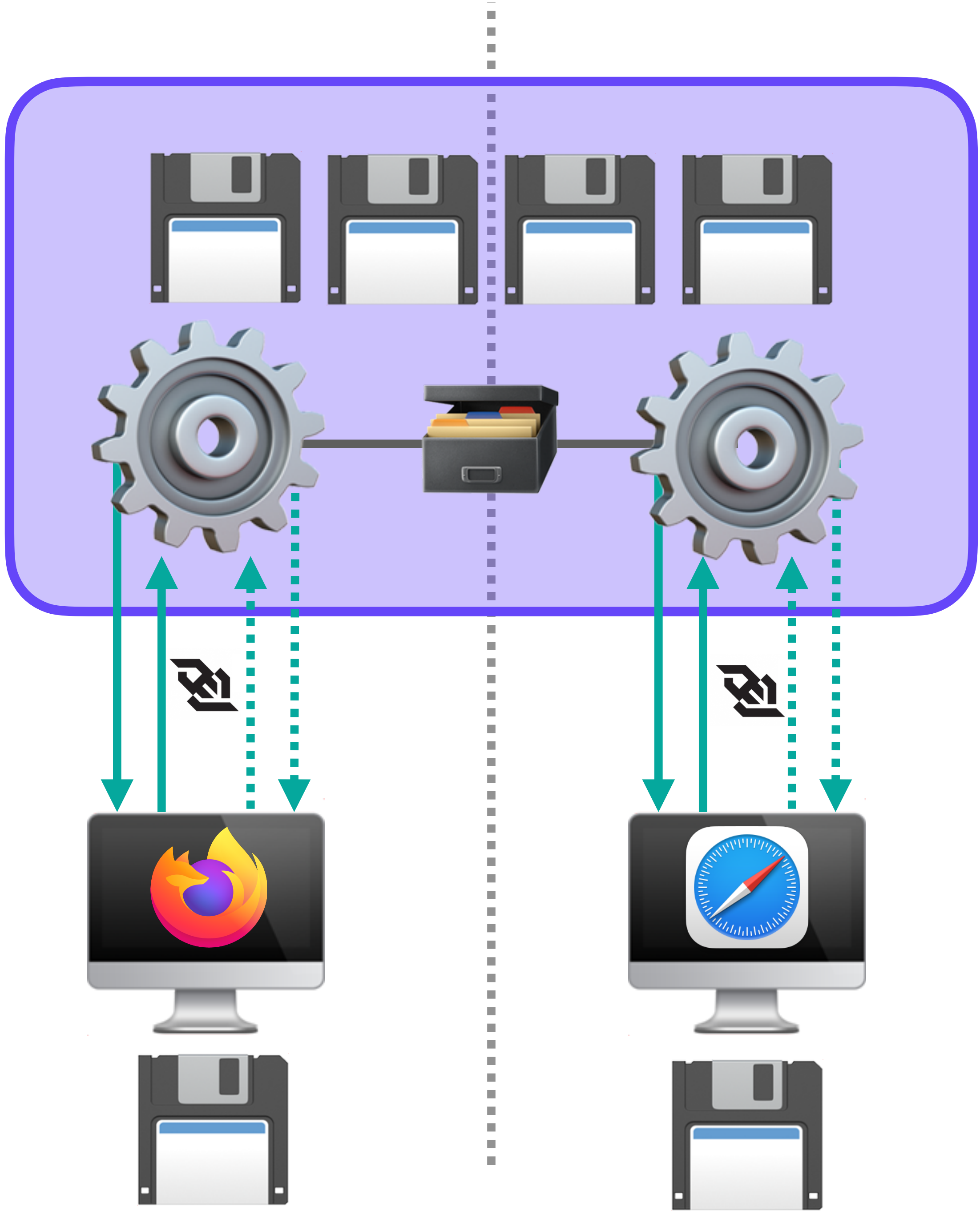
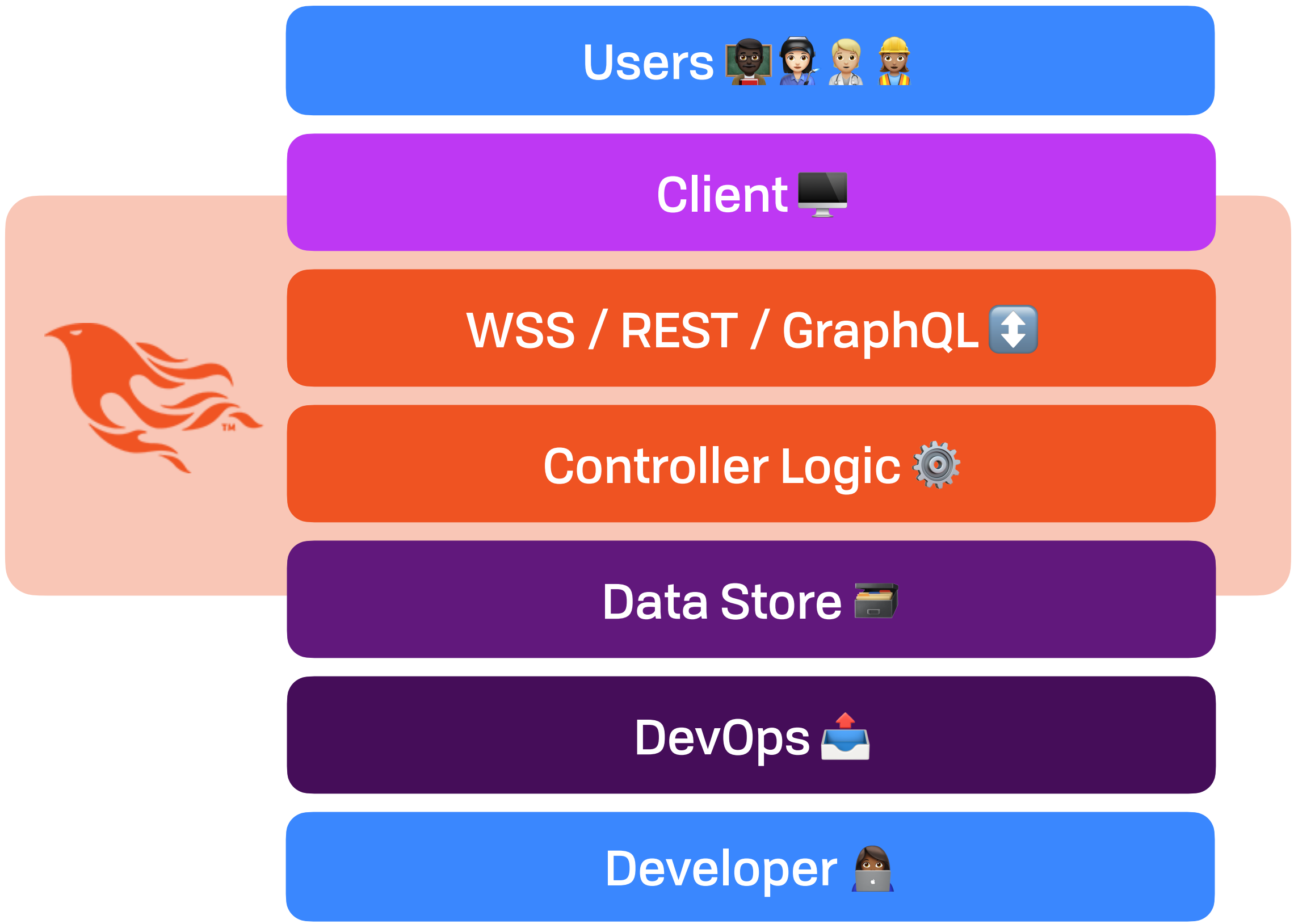
On the Edge 

Phoenix LiveView



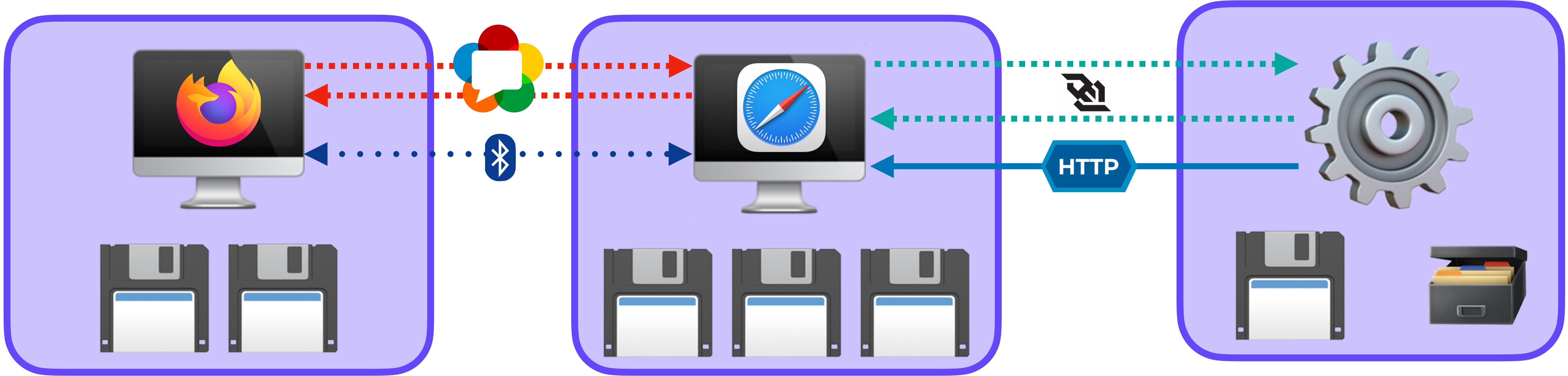
On the Edge 

Phoenix LiveView



On the Edge 

Phoenix Inside Out



Mo Distributed Mo Problems


| Property | Consequence |
|------------------------------|------------------------------------|
| Casual islands | No single source of truth! |
| Local first | How to do coordination? |
| Replicate/run on any machine | ACLs fail (for reads, many writes) |

On the Edge 

CAP → *PACELC*  


On the Edge 

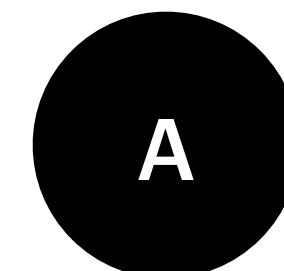
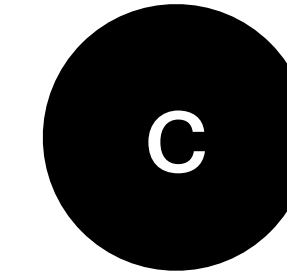
CAP → *PACELC*  

- If network partition, either:
 - Availability (A)  **Uptime!**
 - Consistency (C)

On the Edge 

CAP → *PACELC*  

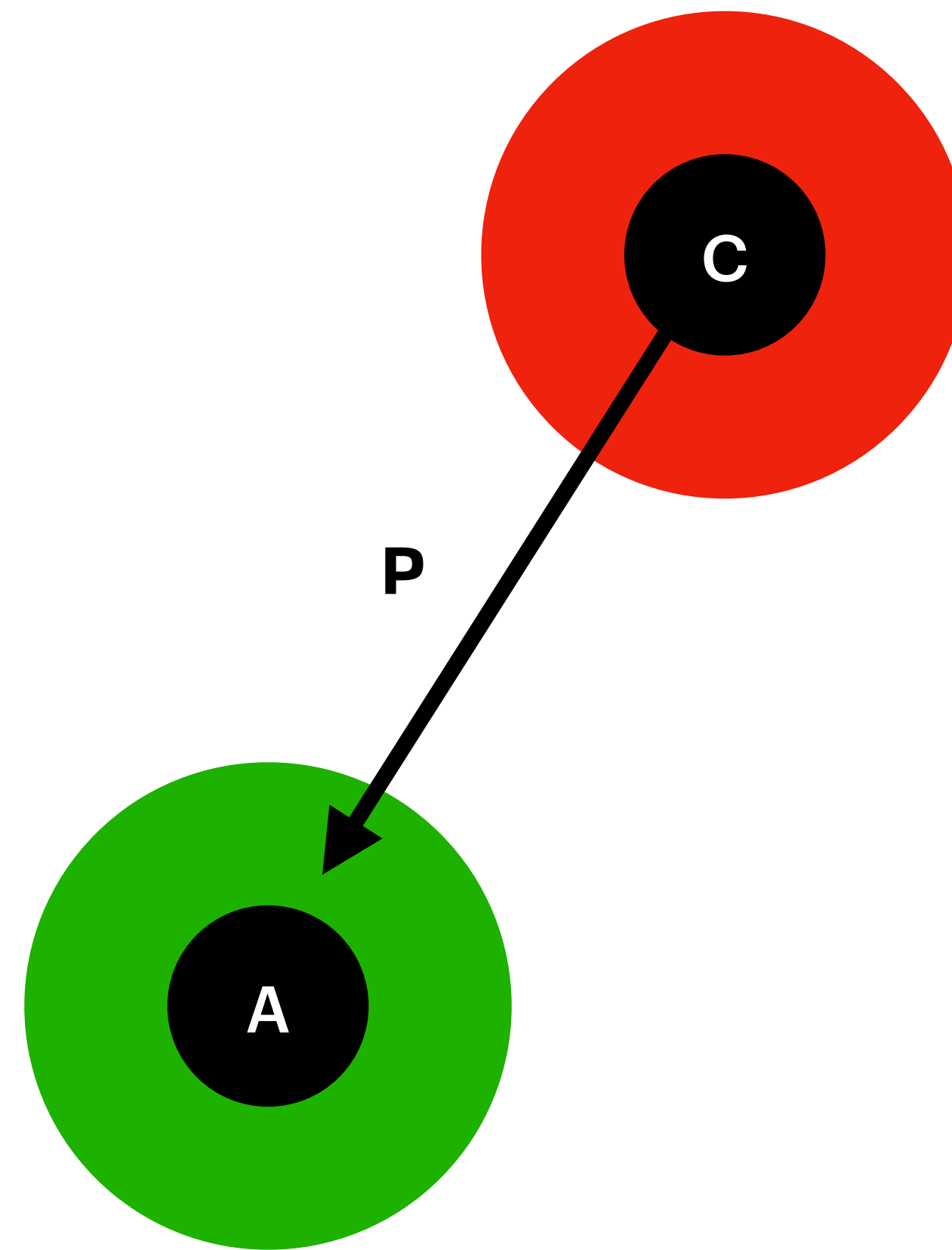
- If network partition, either:
 - Availability (A)  **Uptime!**
 - Consistency (C)



On the Edge 🧗

CAP → *PACELC* 📦 🦌

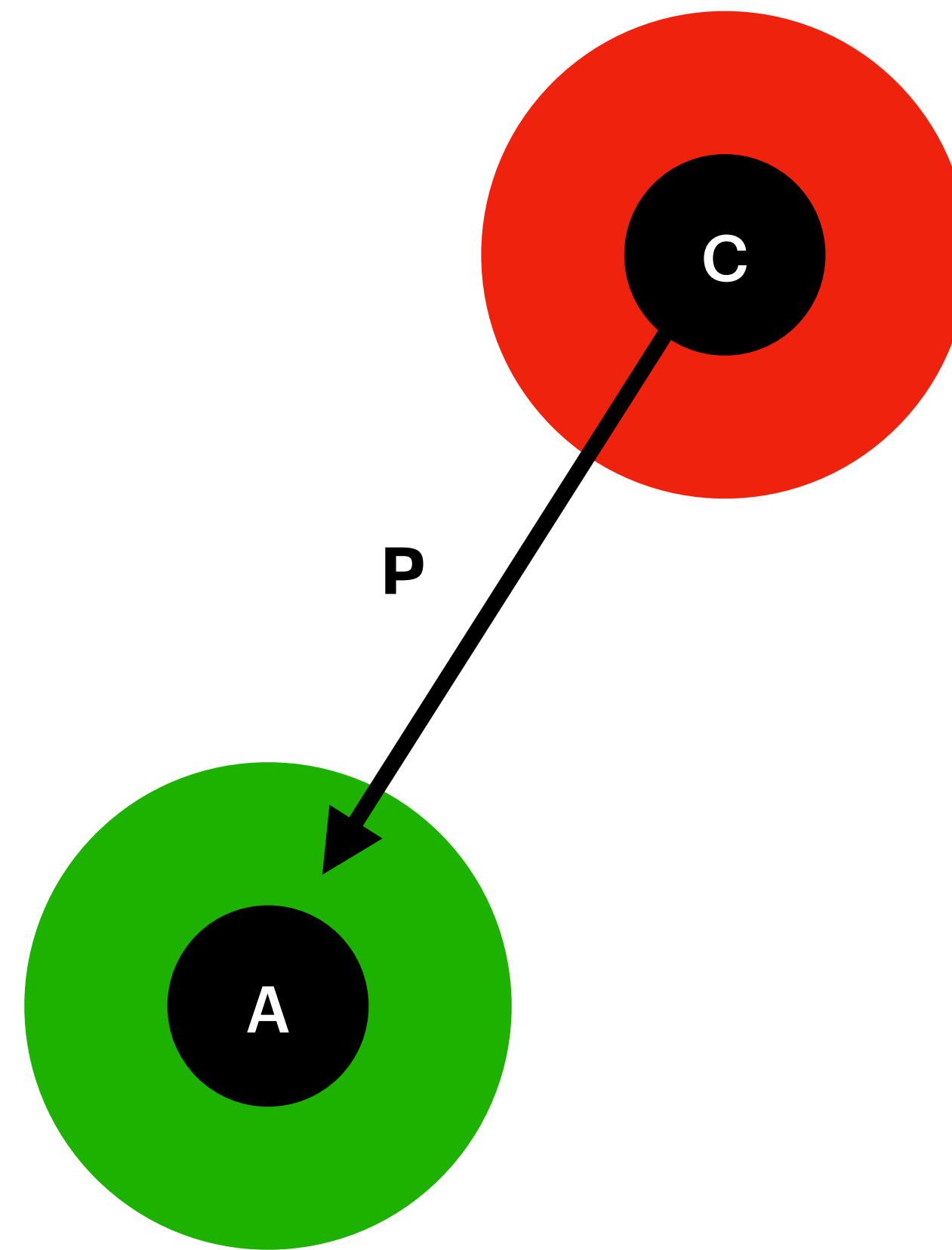
- If network partition, either:
 - Availability (A) ✅ **Uptime!**
 - Consistency (C)



On the Edge 🧑‍🏭

CAP → *PACELC* 📦 🦌

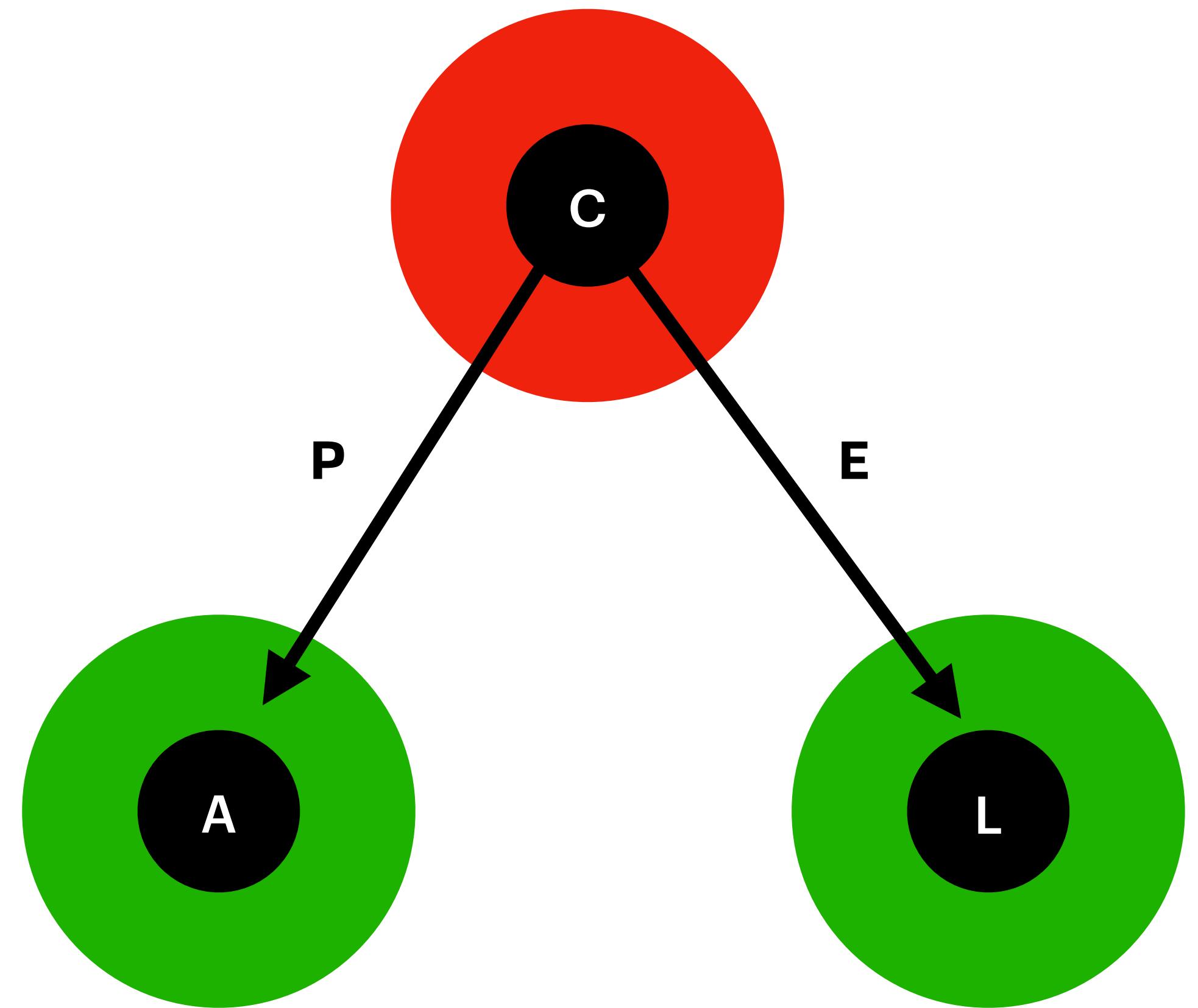
- If network partition, either:
 - Availability (A) ✅ **Uptime!**
 - Consistency (C)
- Else (E) running normally, either:
 - Latency (L) ✅ **Speed!**
 - Consistency (C)



On the Edge 🧗

CAP → *PACELC* 📦 🦌

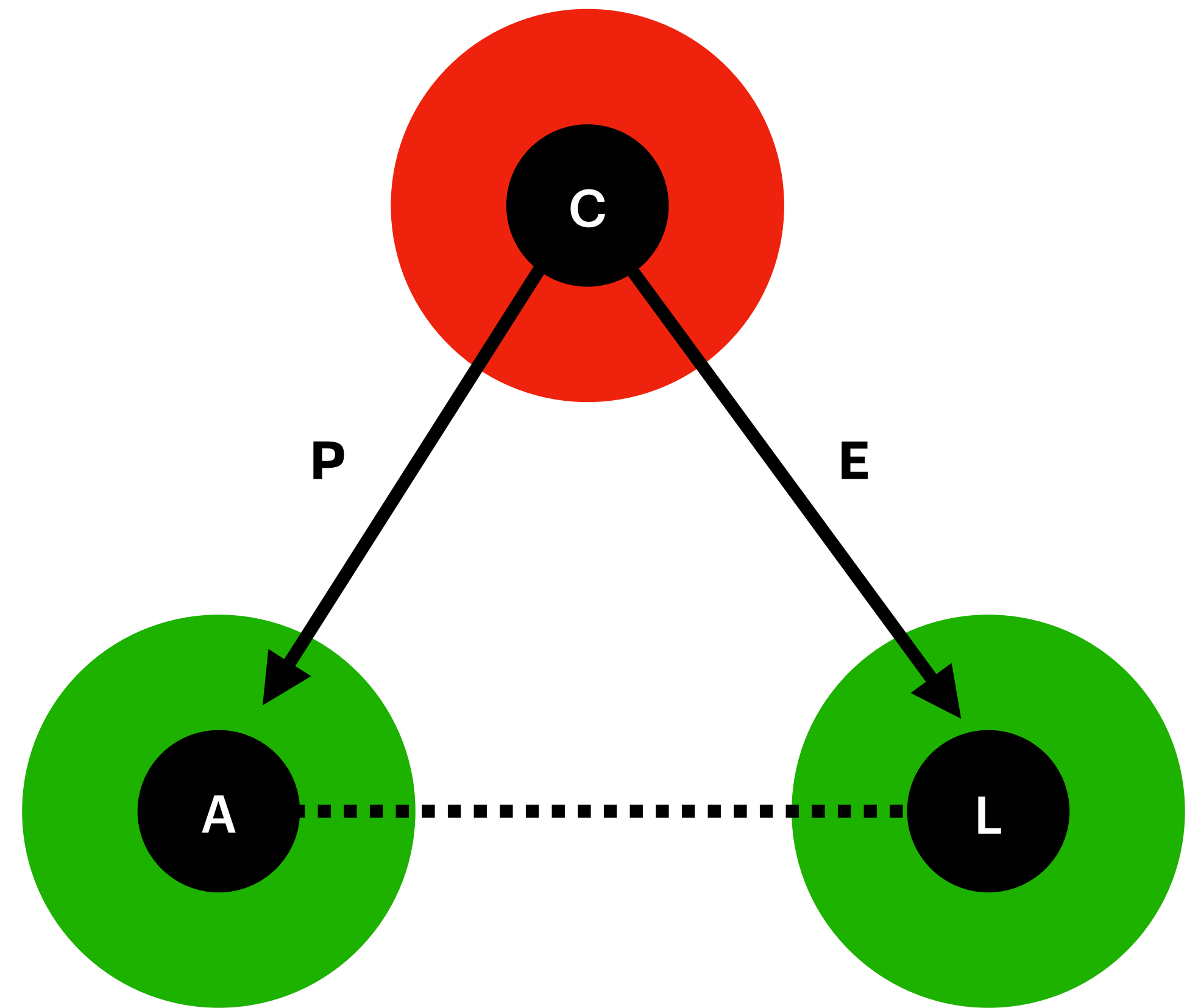
- If network partition, either:
 - Availability (A) ✅ **Uptime!**
 - Consistency (C)
- Else (E) running normally, either:
 - Latency (L) ✅ **Speed!**
 - Consistency (C)



On the Edge 🧗

CAP → *PACELC* 📦 🦌

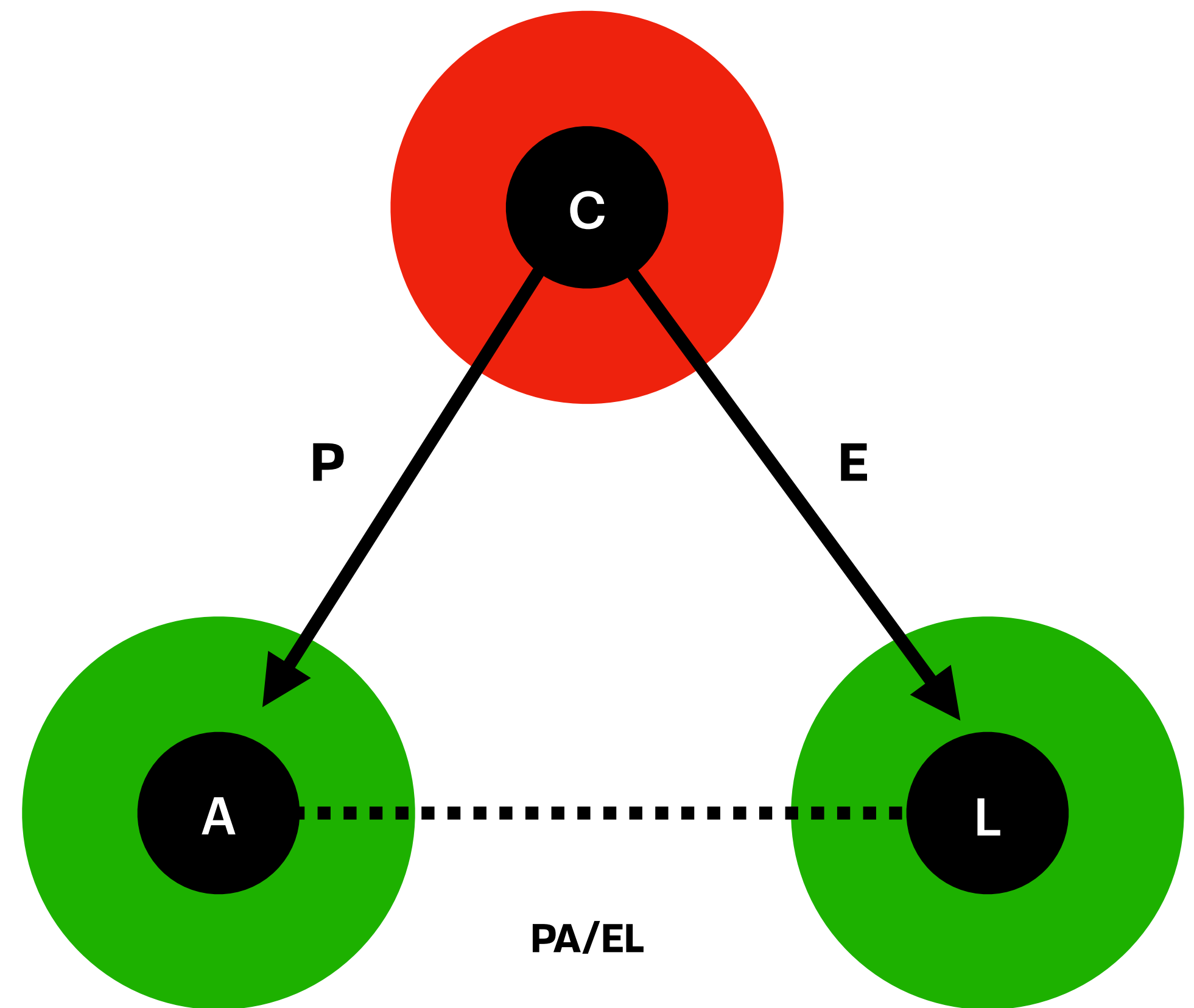
- If network partition, either:
 - Availability (A) ✅ **Uptime!**
 - Consistency (C)
- Else (E) running normally, either:
 - Latency (L) ✅ **Speed!**
 - Consistency (C)



On the Edge 🧗

CAP → *PACELC* 📦 🦌

- If network partition, either:
 - Availability (A) ✅ **Uptime!**
 - Consistency (C)
- Else (E) running normally, either:
 - Latency (L) ✅ **Speed!**
 - Consistency (C)

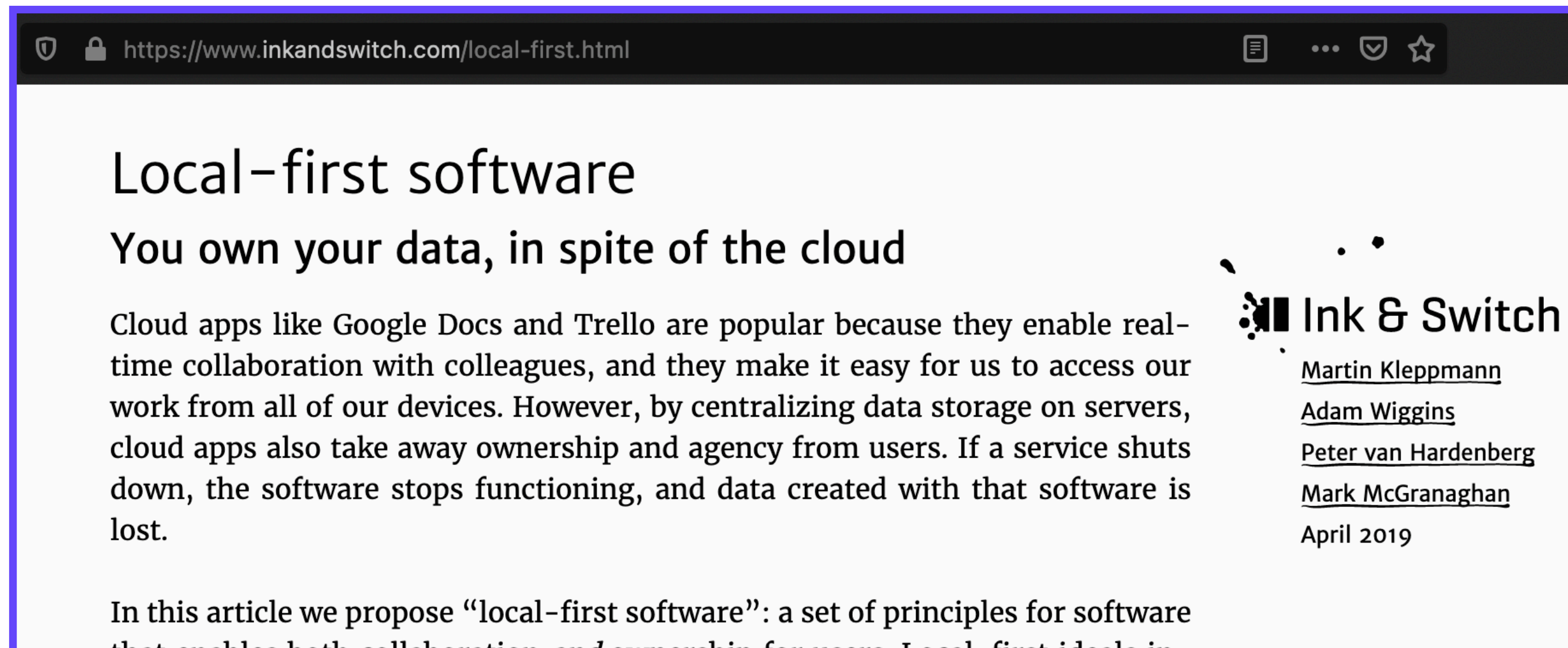


On the Edge 

New Assumptions, New Approach

New Assumptions, New Approach

- New assumptions → new architecture → new features
- Local-first → network efficient
- Data can run anywhere → commons networks



The screenshot shows a web browser window with the address bar displaying <https://www.inkandswitch.com/local-first.html>. The main content of the page is an article titled "Local-first software" with the subtitle "You own your data, in spite of the cloud". The article text discusses the popularity of cloud apps like Google Docs and Trello, but notes that they centralize data storage on servers, taking away ownership and agency from users. It states that if a service shuts down, the software stops functioning, and data created with that software is lost. The article is attributed to Ink & Switch, with authors listed as Martin Kleppmann, Adam Wiggins, Peter van Hardenberg, and Mark McGranaghan. The date is April 2019. The bottom of the article begins with "In this article we propose 'local-first software': a set of principles for software that enables both collaboration and ownership for users. Local-first ideals in".

On the Edge



New Tools

On the Edge 

New Tools

- Conflict-free Replicated Data Type (CRDT)
- Software Transactional Memory (STM)
- Differential Datalog (DDlog)
- Merkleization (hash chaining)
- Cryptographic compute (MPC/FHE/NIZK)
- REST < bidirectional sync < relative data views

On the Edge 

New Tools

- Conflict-free Replicated Data Type (CRDT)
- Software Transactional Memory (STM)
- Differential Datalog (DDlog)
- Merkleization (hash chaining)
- Cryptographic compute (MPC/FHE/NIZK)
- REST < bidirectional sync < relative data views

On the Edge 

New Tools

- Conflict-free Replicated Data Type (CRDT)
- Software Transactional Memory (STM)
- Differential Datalog (DDlog)
- Merkleization (hash chaining)
- Cryptographic compute (MPC/FHE/NIZK)
- REST < bidirectional sync < relative data views

On the Edge 

New Tools

- Conflict-free Replicated Data Type (CRDT)
- Software Transactional Memory (STM)
- Differential Datalog (DDlog)
- Merkleization (hash chaining)
- Cryptographic compute (MPC/FHE/NIZK)
- REST < bidirectional sync < relative data views

Fixing The Leaky Pipes

Identity & Access



Fixing the Leaky Pipes 

ACL Read & Write

Fixing the Leaky Pipes 🚿

ACL Read & Write



Fixing the Leaky Pipes 🚿

ACL Read & Write



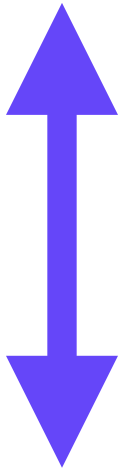
Fixing the Leaky Pipes 🚿

ACL Read & Write



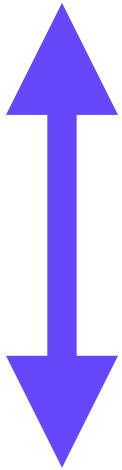
Fixing the Leaky Pipes 

ACL Read & Write



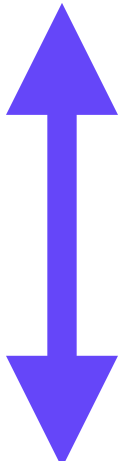
Fixing the Leaky Pipes 

ACL Read & Write



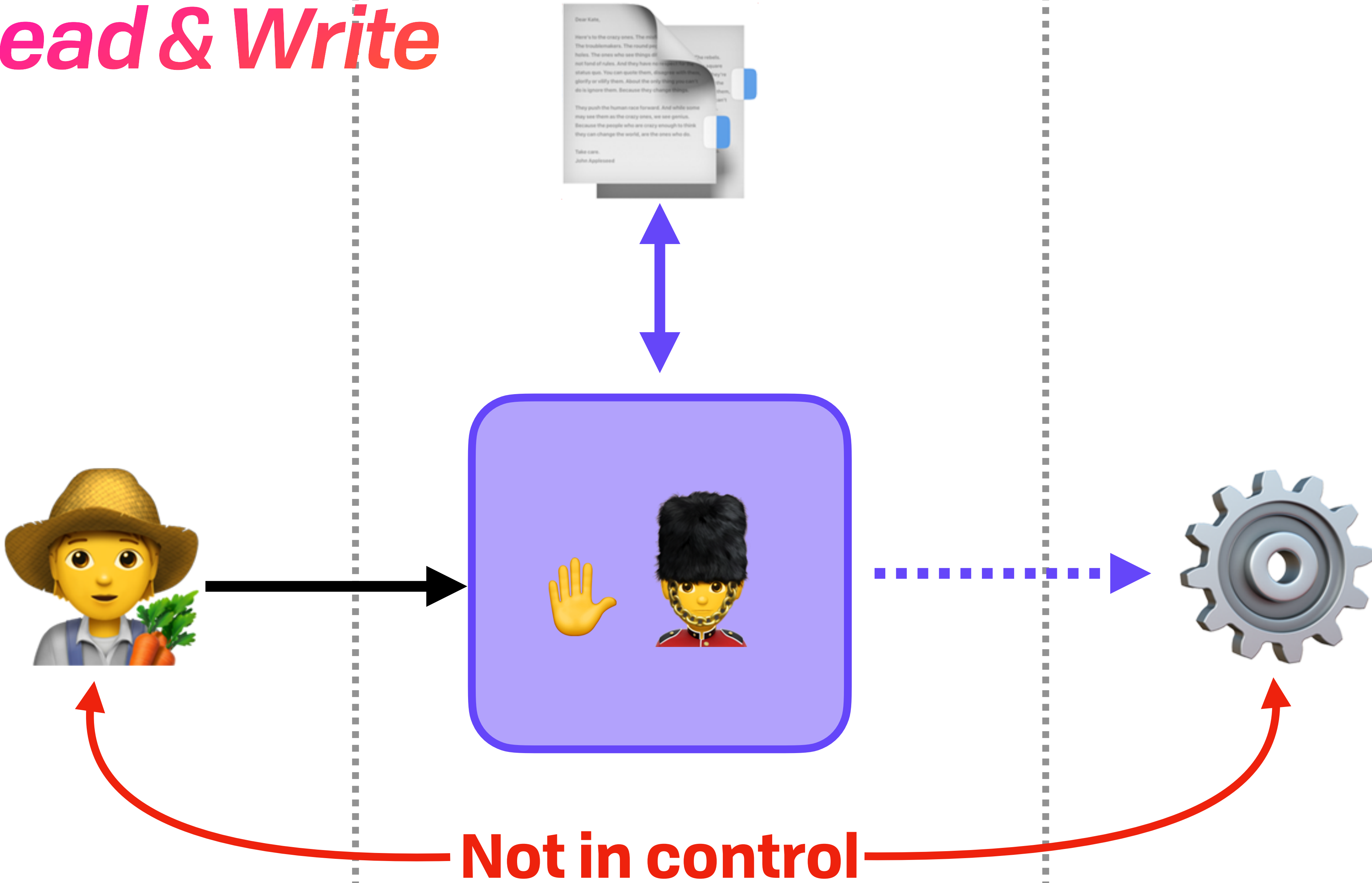
Fixing the Leaky Pipes 

ACL Read & Write



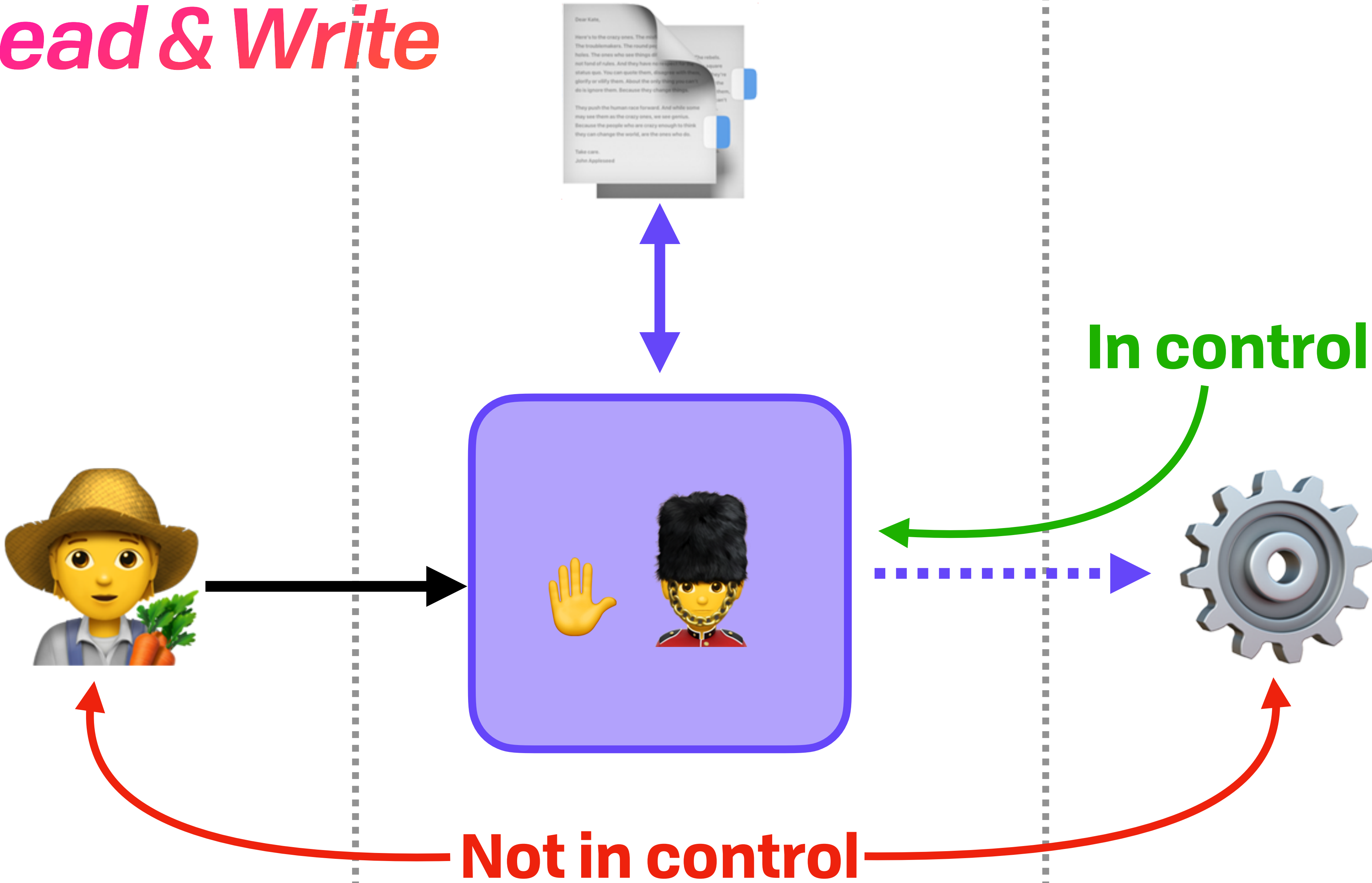
Fixing the Leaky Pipes 

ACL Read & Write



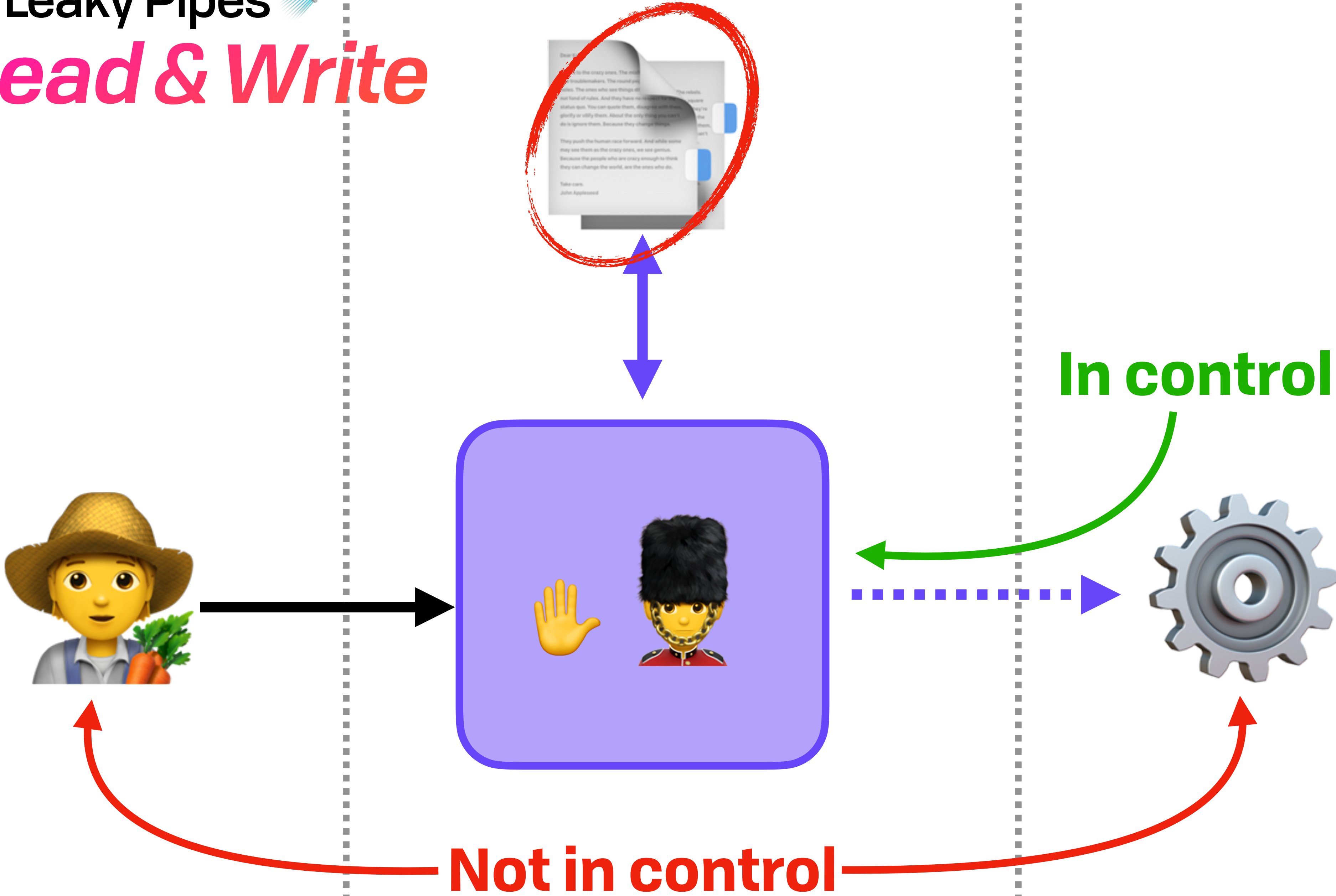
Fixing the Leaky Pipes 🚿

ACL Read & Write



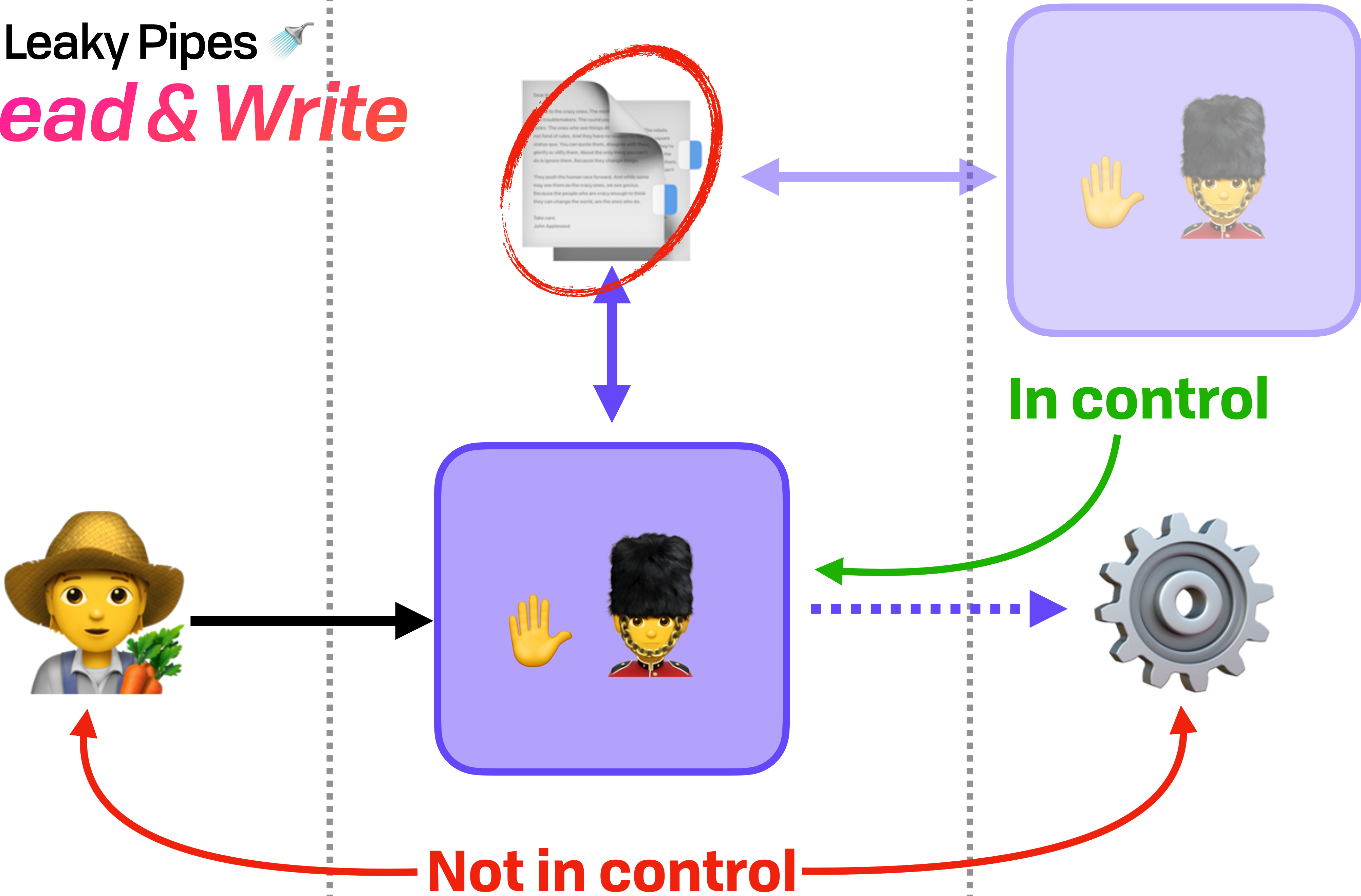
Fixing the Leaky Pipes 

ACL Read & Write



Fixing the Leaky Pipes 

ACL Read & Write



Fixing the Leaky Pipes 

OCAP for Mutation

Fixing the Leaky Pipes 🚿

OCAP for Mutation



Fixing the Leaky Pipes 🚿

OCAP for Mutation



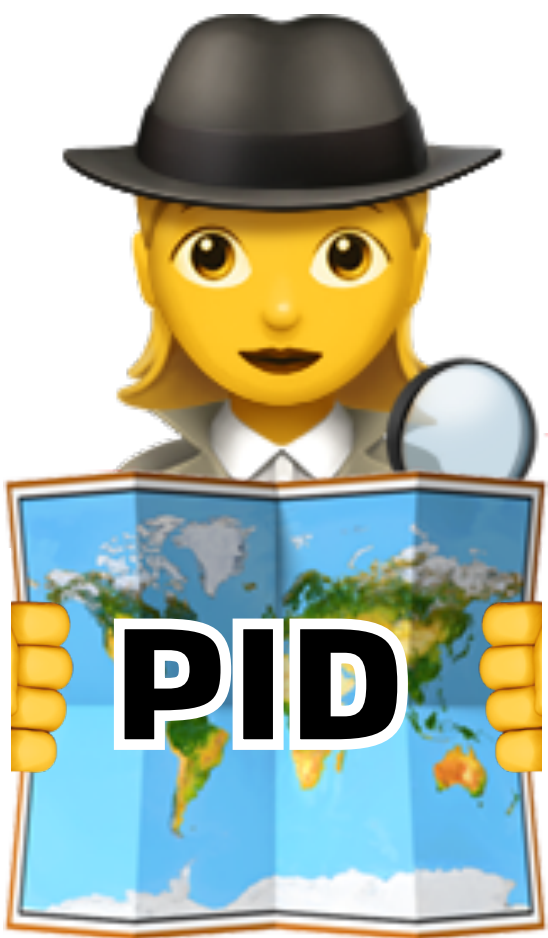
Fixing the Leaky Pipes 🚿

OCAP for Mutation



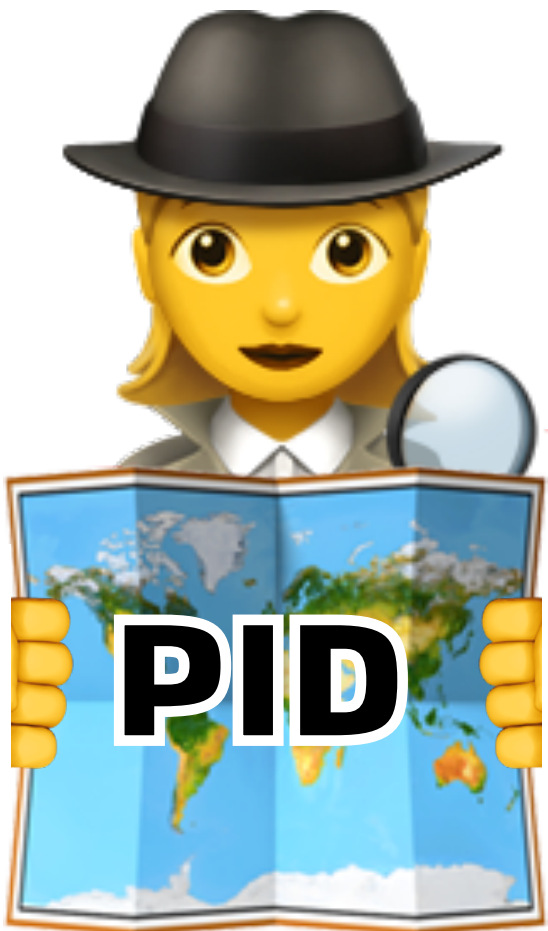
Fixing the Leaky Pipes 🚿

OCAP for Mutation



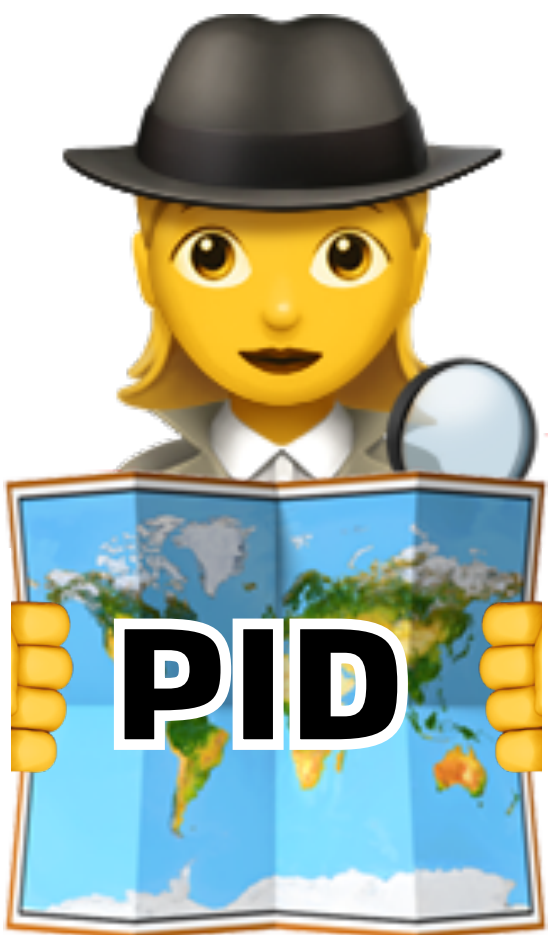
Fixing the Leaky Pipes 🚿

OCAP for Mutation



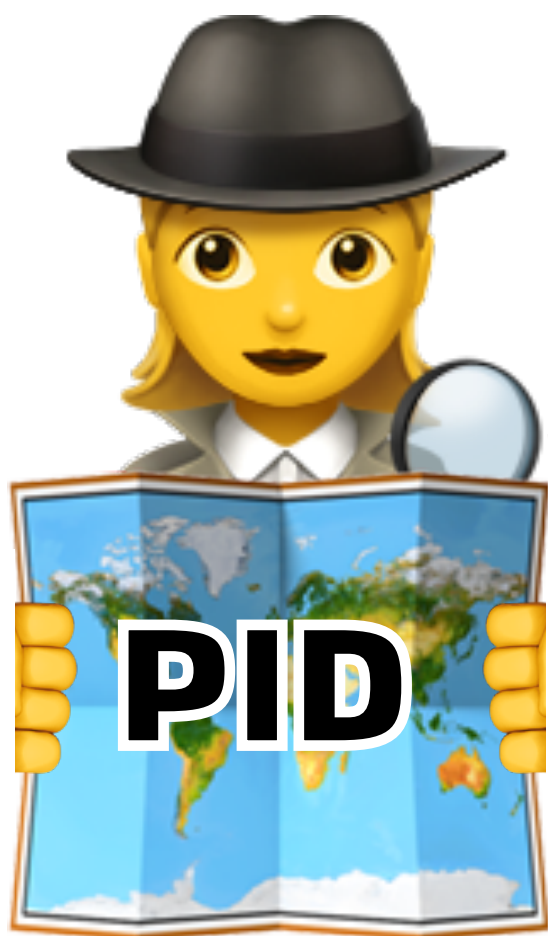
Fixing the Leaky Pipes 🚿

OCAP for Mutation



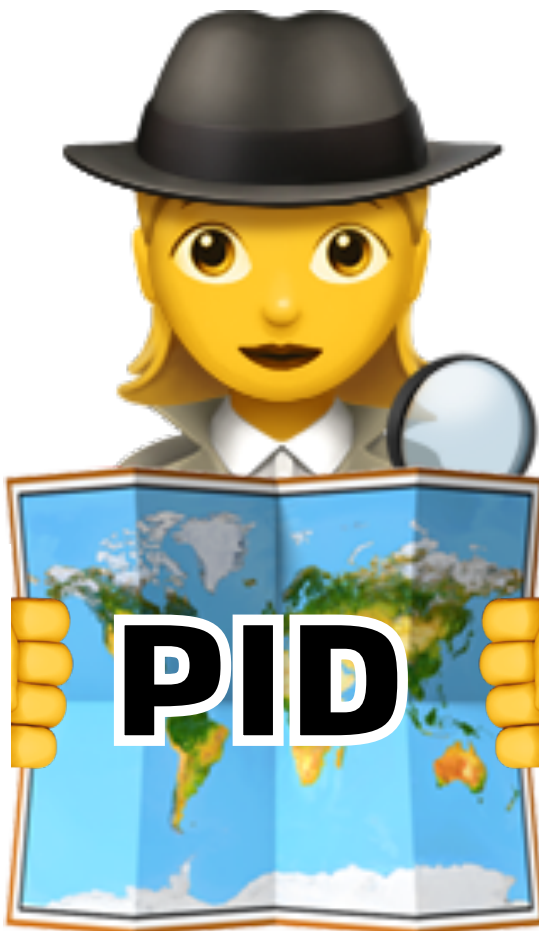
Fixing the Leaky Pipes 🚿

OCAP for Mutation



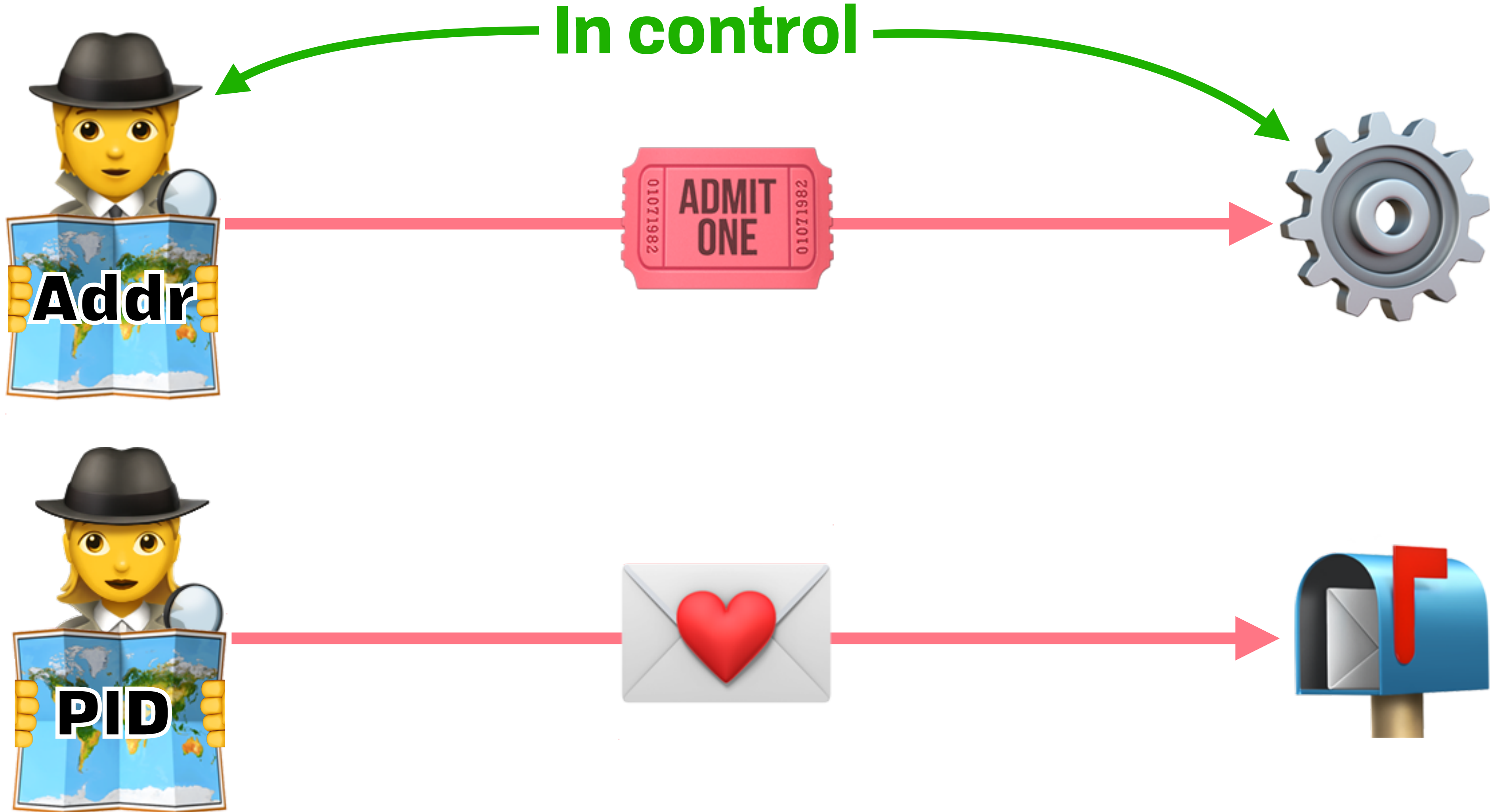
Fixing the Leaky Pipes 🚿

OCAP for Mutation



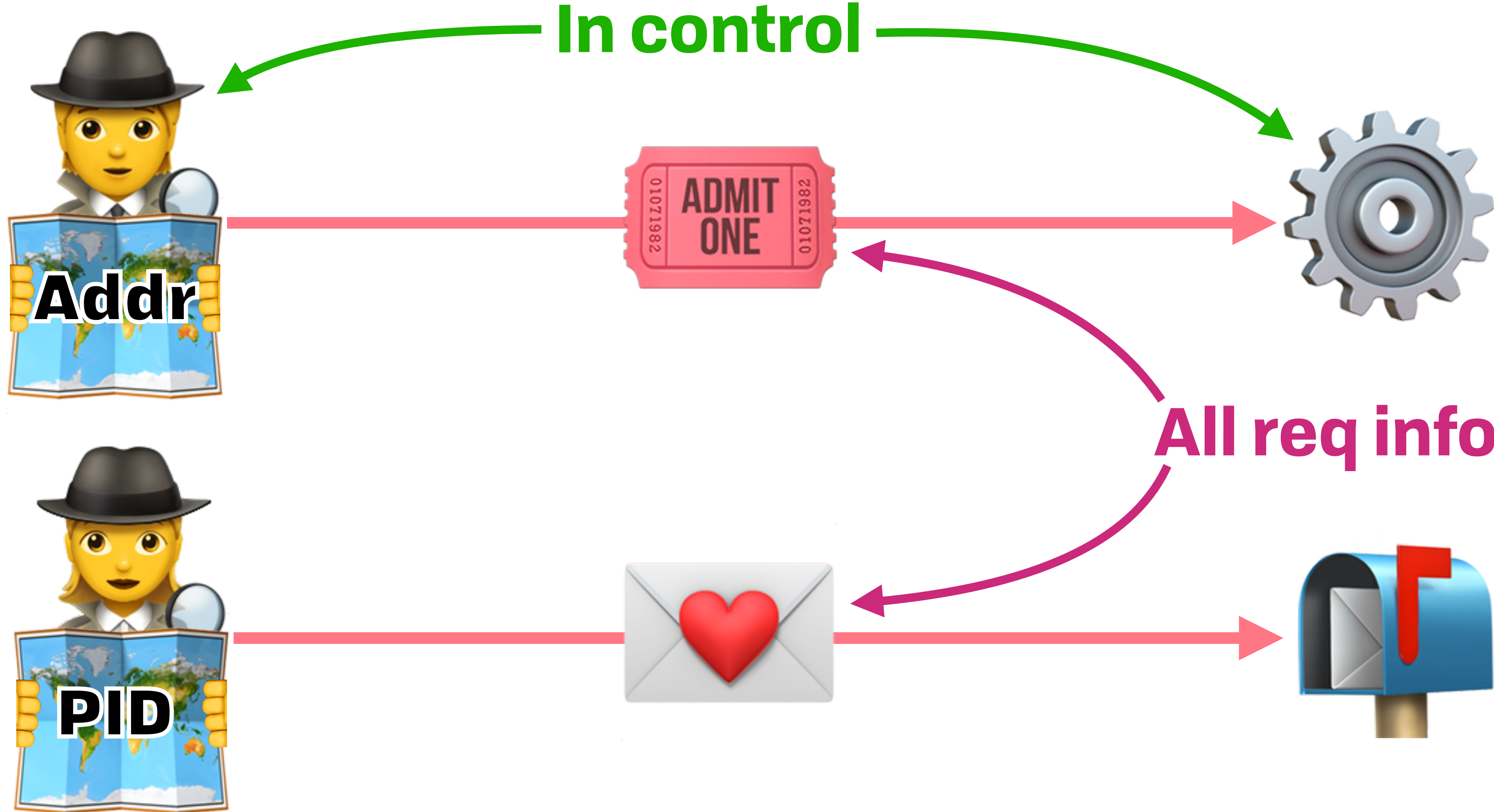
Fixing the Leaky Pipes 🚿

OCAP for Mutation



Fixing the Leaky Pipes 🚿

OCAP for Mutation



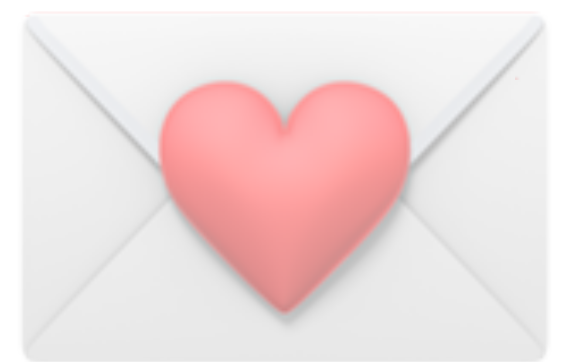
Fixing the Leaky Pipes 🚿

OCAP for Mutation



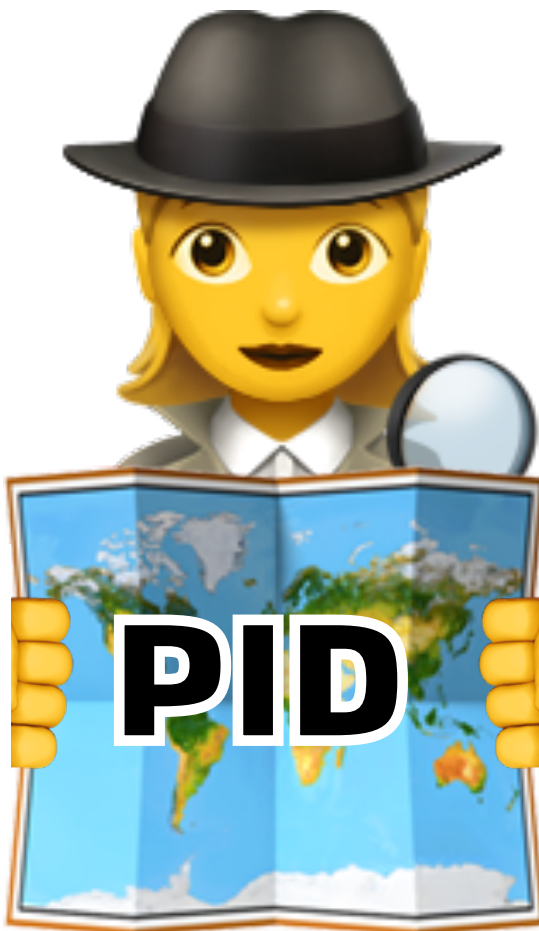
Fixing the Leaky Pipes 🚿

OCAP for Mutation



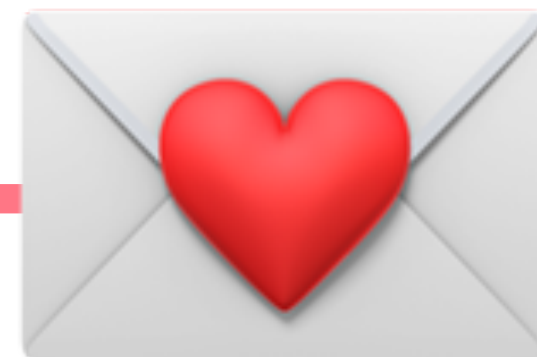
Fixing the Leaky Pipes 🚿

OCAP for Mutation



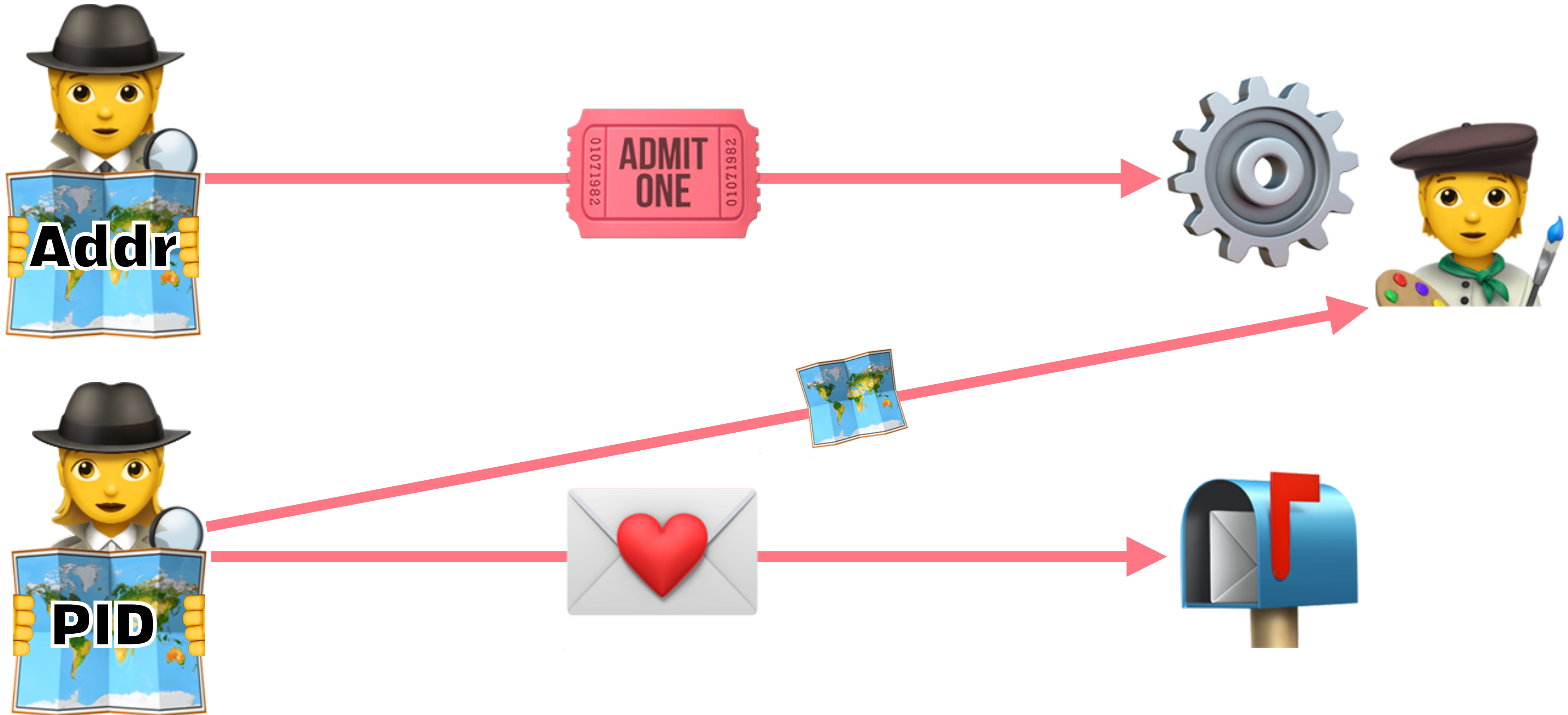
Fixing the Leaky Pipes 🚿

OCAP for Mutation



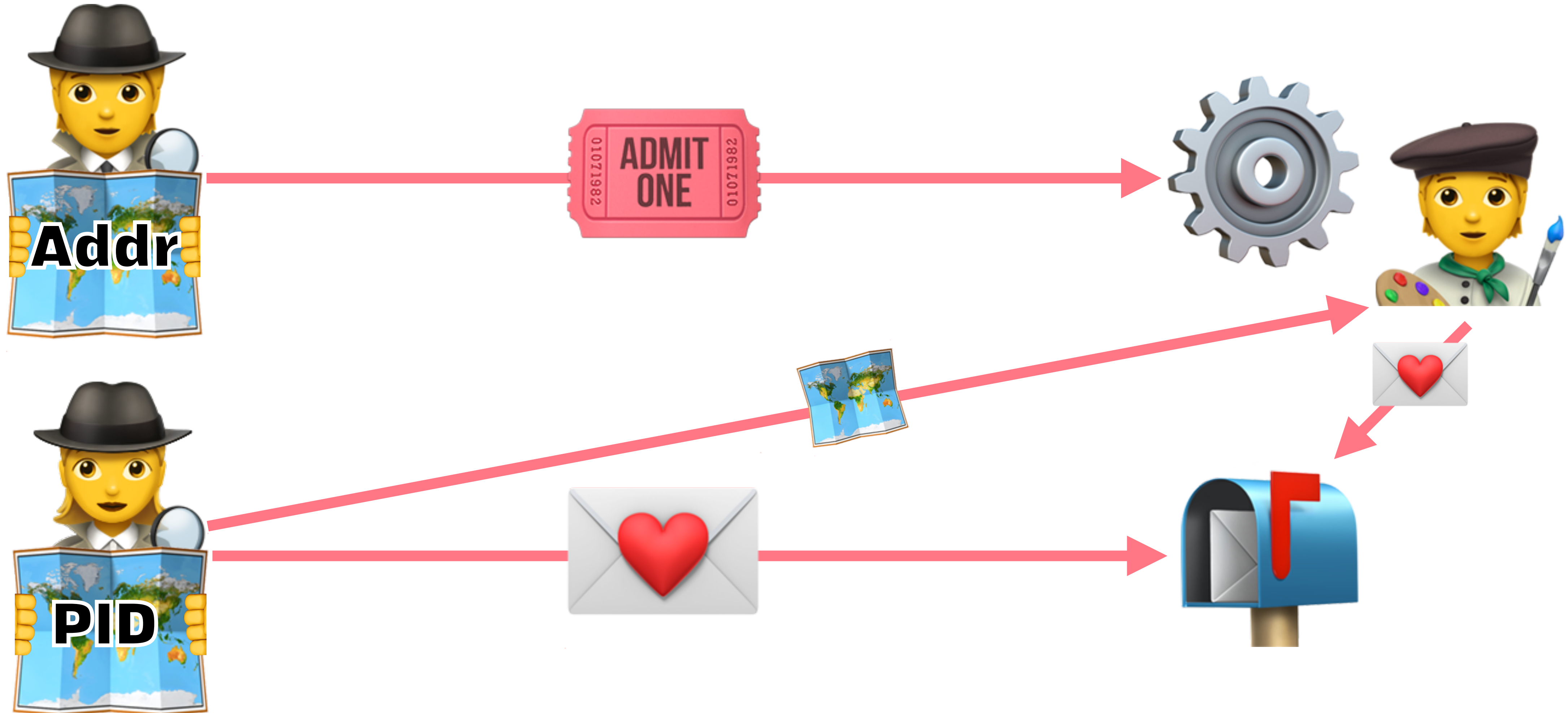
Fixing the Leaky Pipes 🚿

OCAP for Mutation



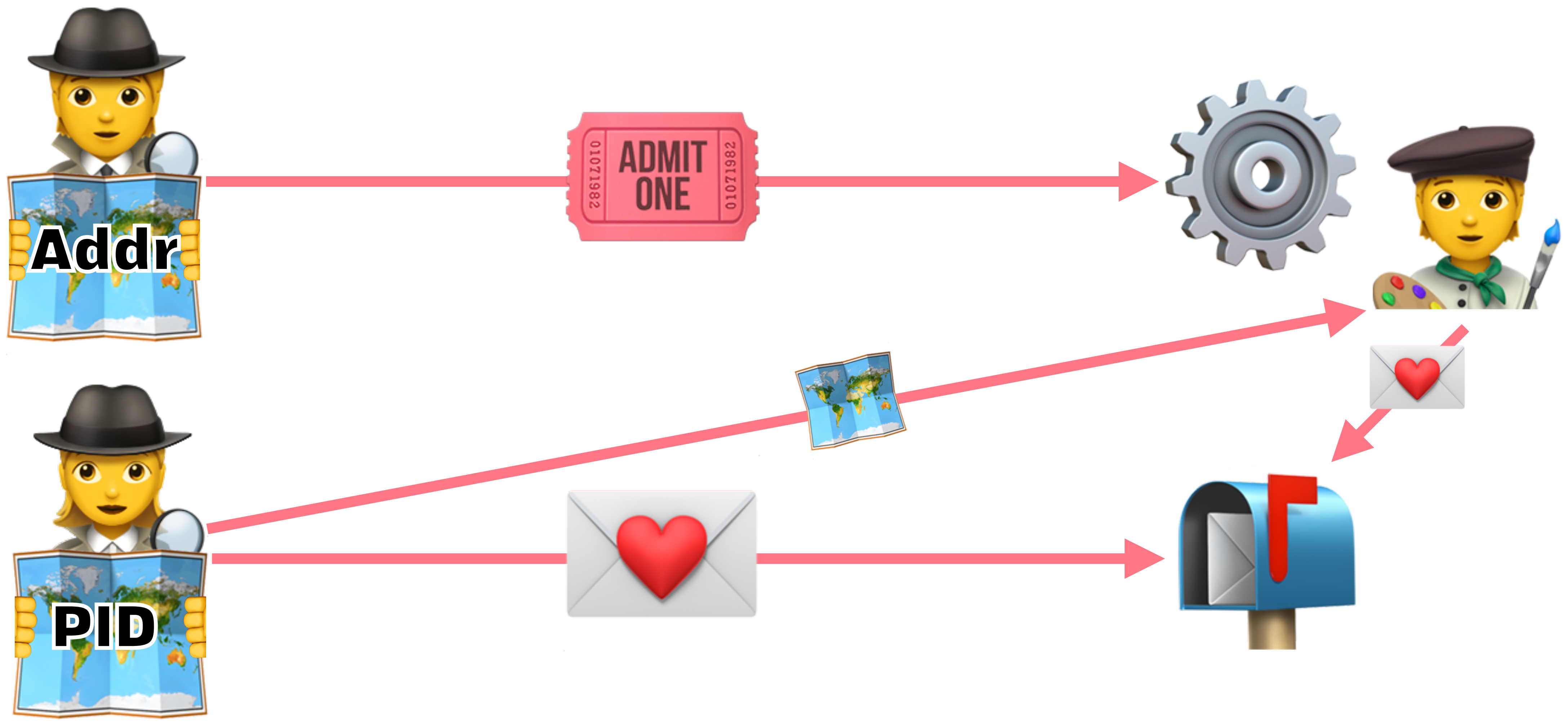
Fixing the Leaky Pipes 🚿

OCAP for Mutation



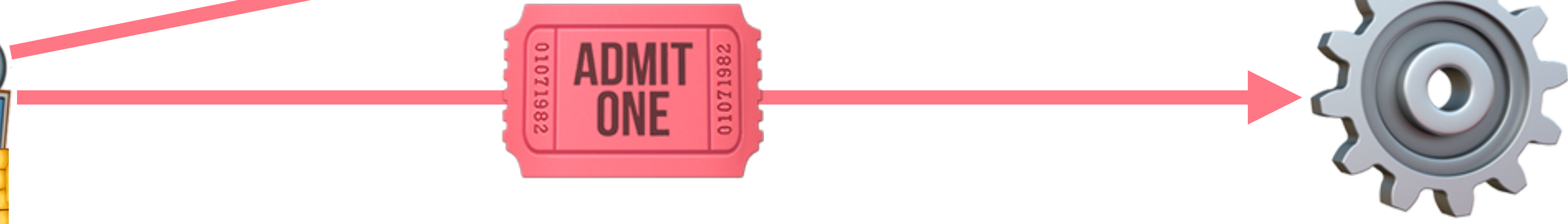
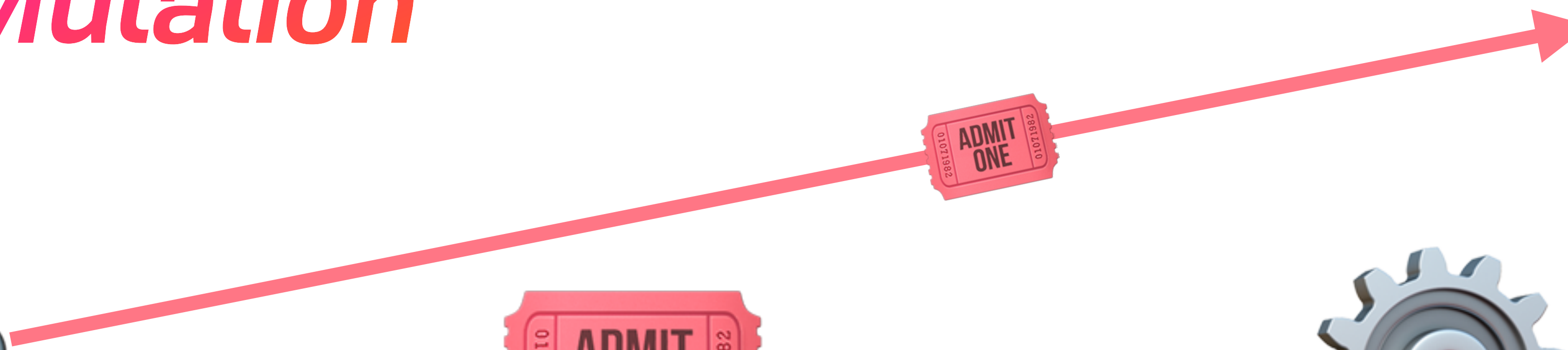
Fixing the Leaky Pipes 🚿

OCAP for Mutation



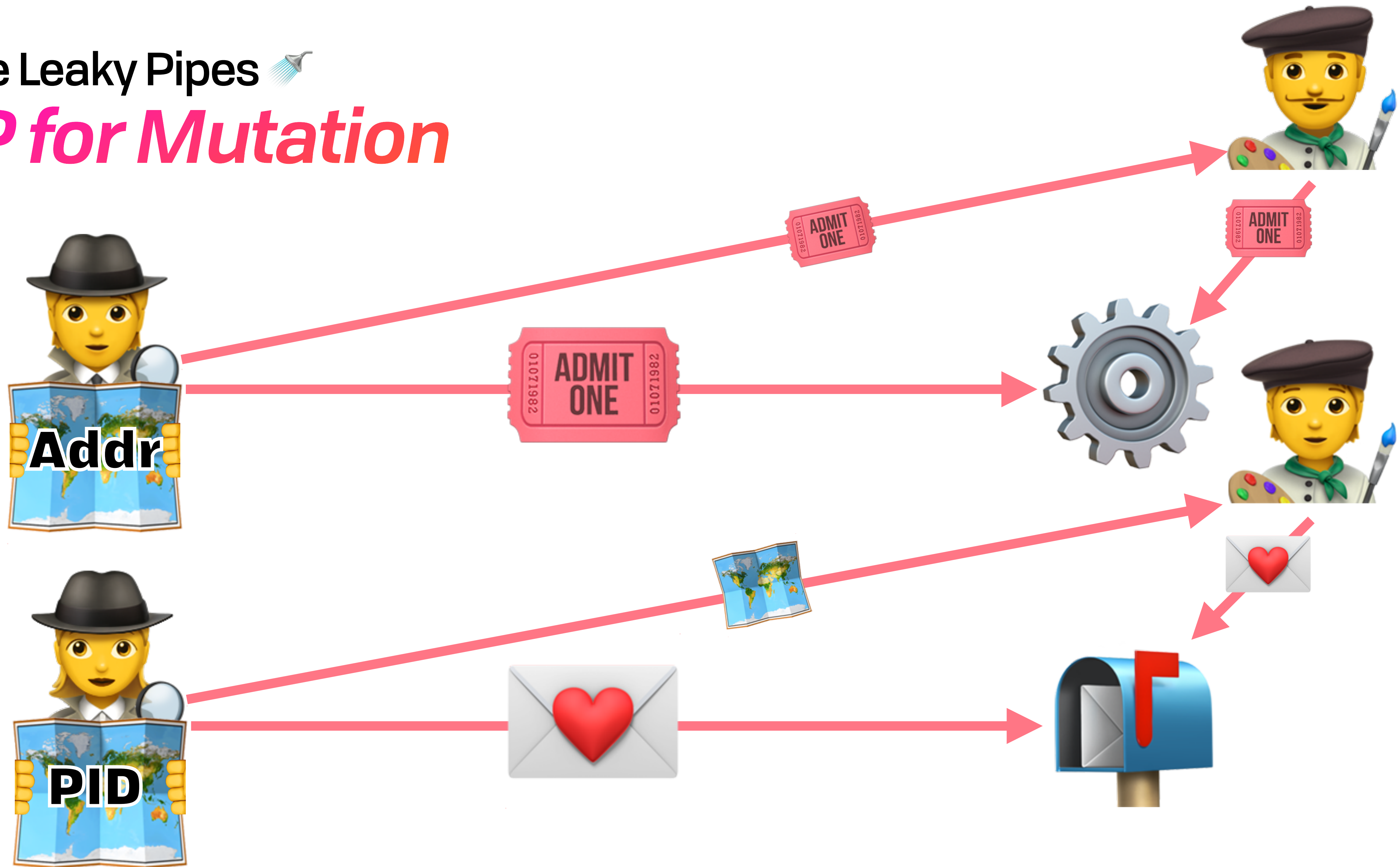
Fixing the Leaky Pipes 🚿

OCAP for Mutation



Fixing the Leaky Pipes 🚿

OCAP for Mutation



Fixing the Leaky Pipes 

Rights Amplification

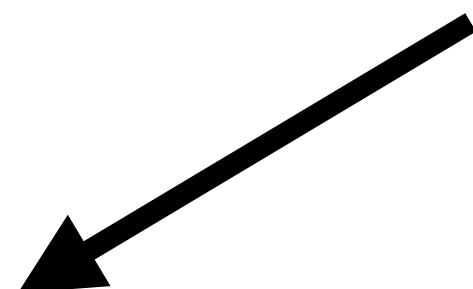
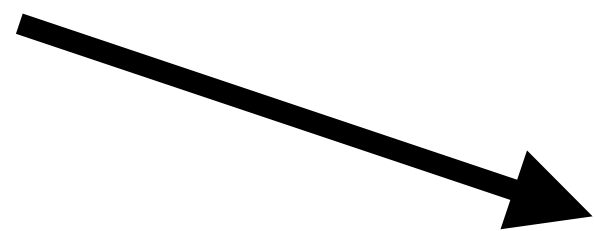
Fixing the Leaky Pipes 🚿

Rights Amplification



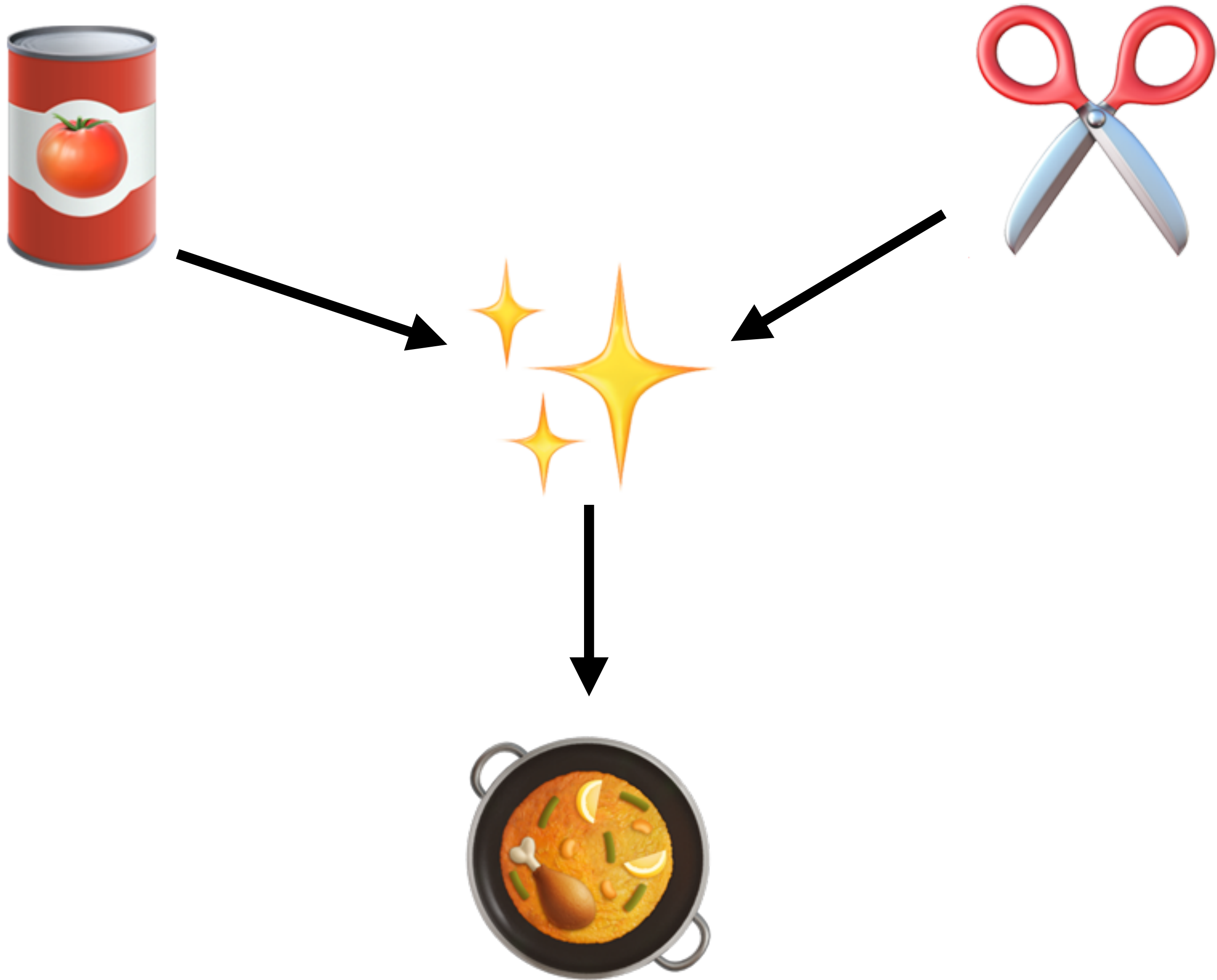
Fixing the Leaky Pipes 🚿

Rights Amplification



Fixing the Leaky Pipes 🚿

Rights Amplification

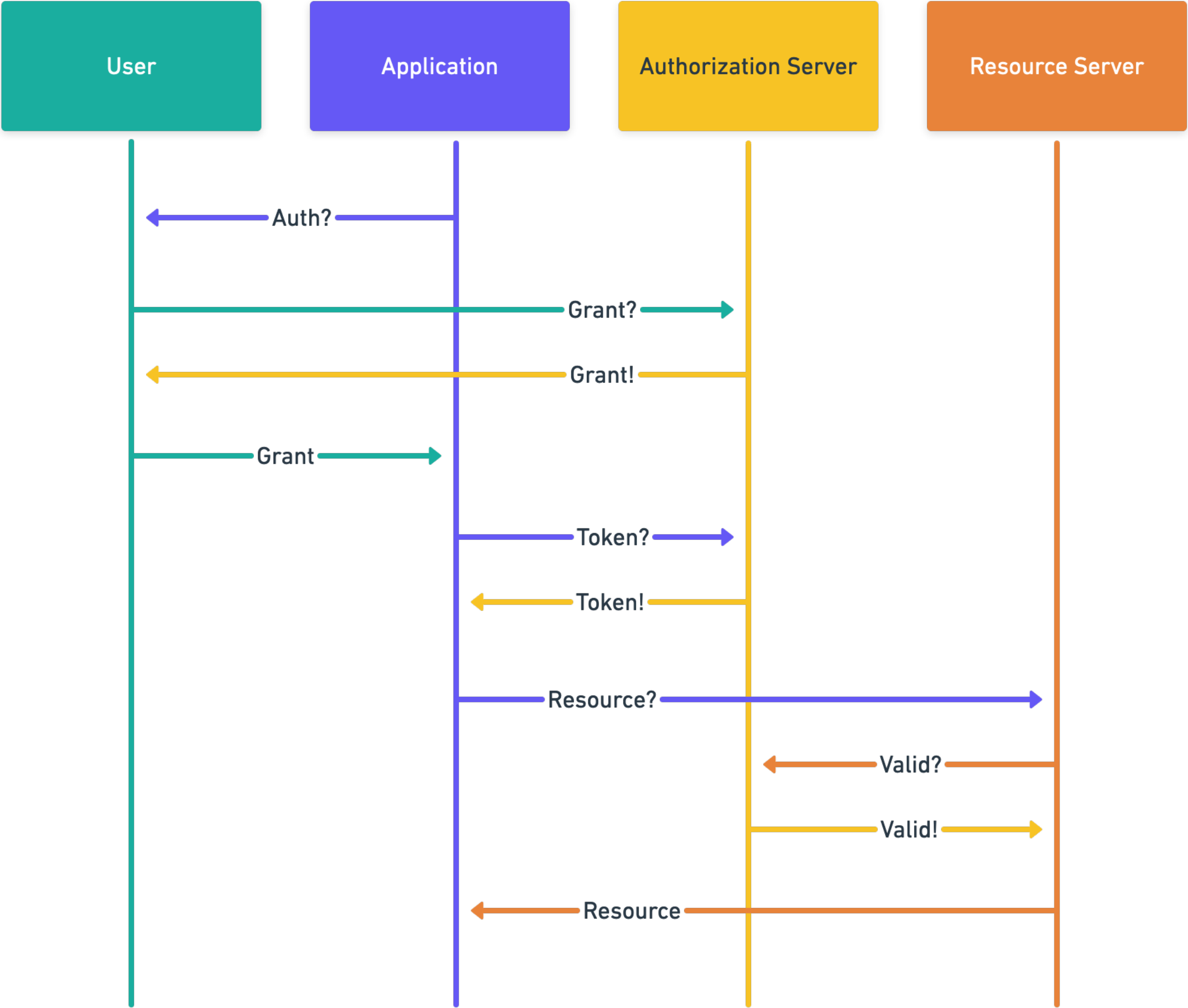


Fixing the Leaky Pipes 

Faster, Stronger, More Streamlined

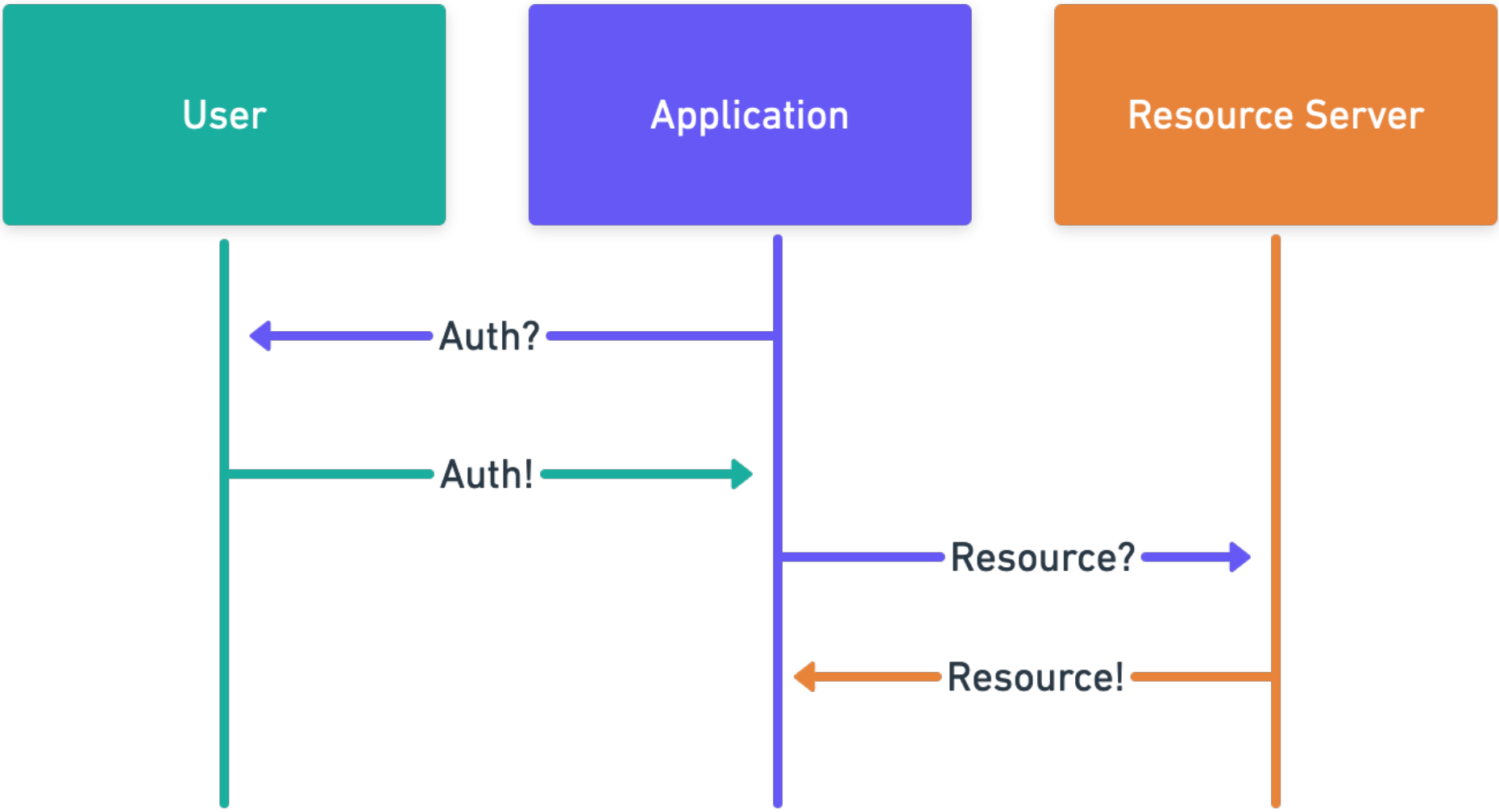
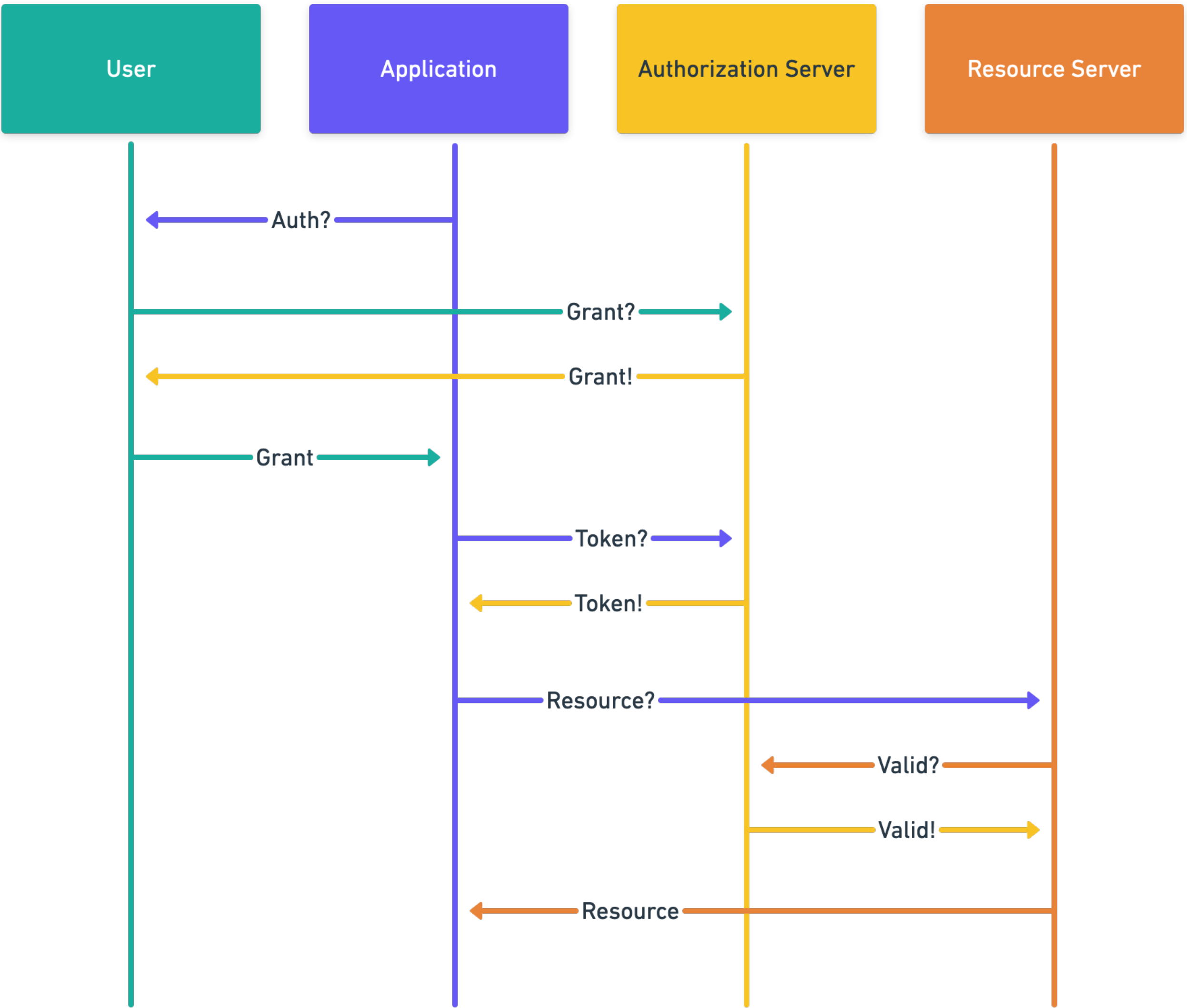
Fixing the Leaky Pipes 🚿

Faster, Stronger, More Streamlined



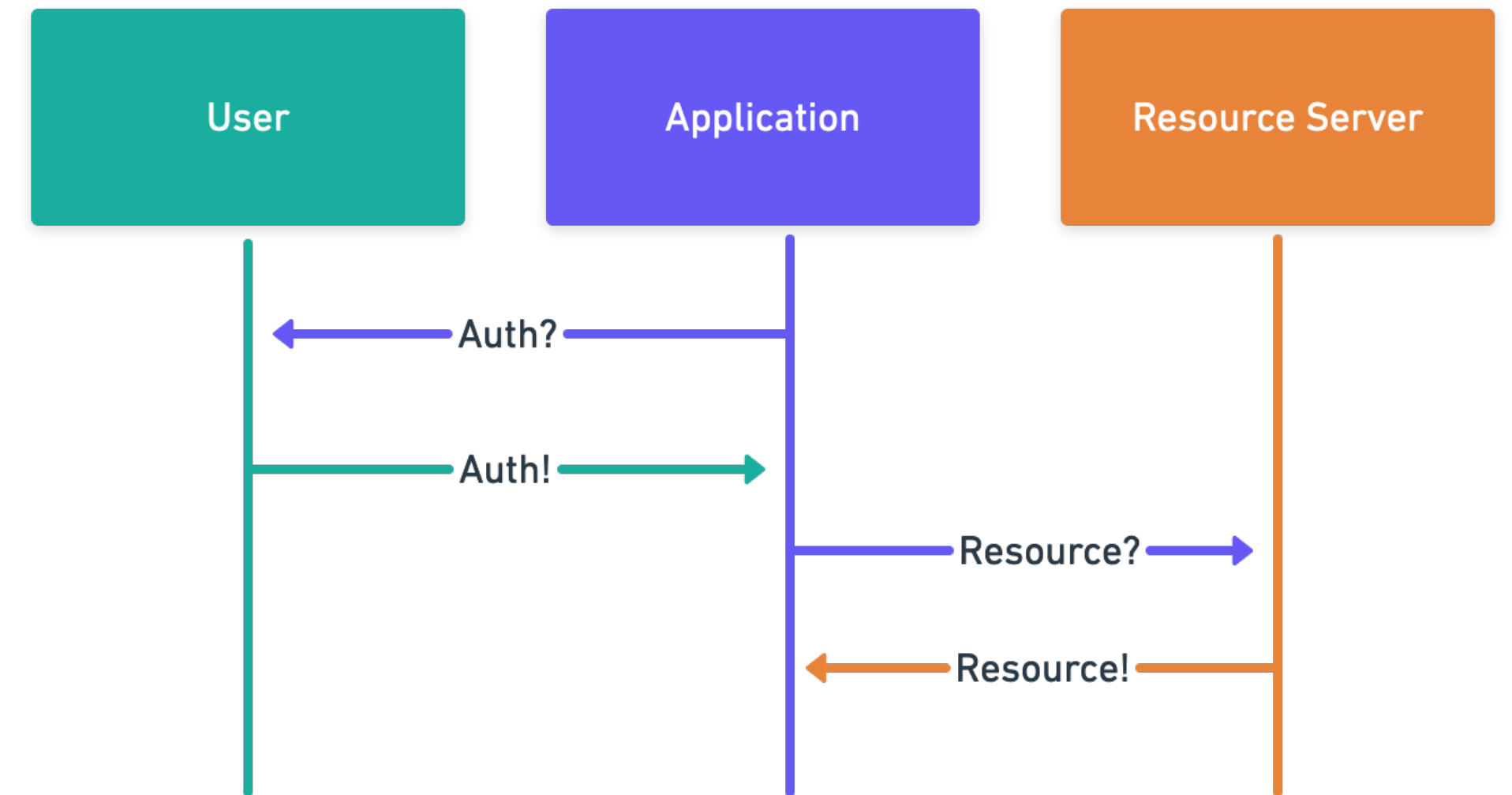
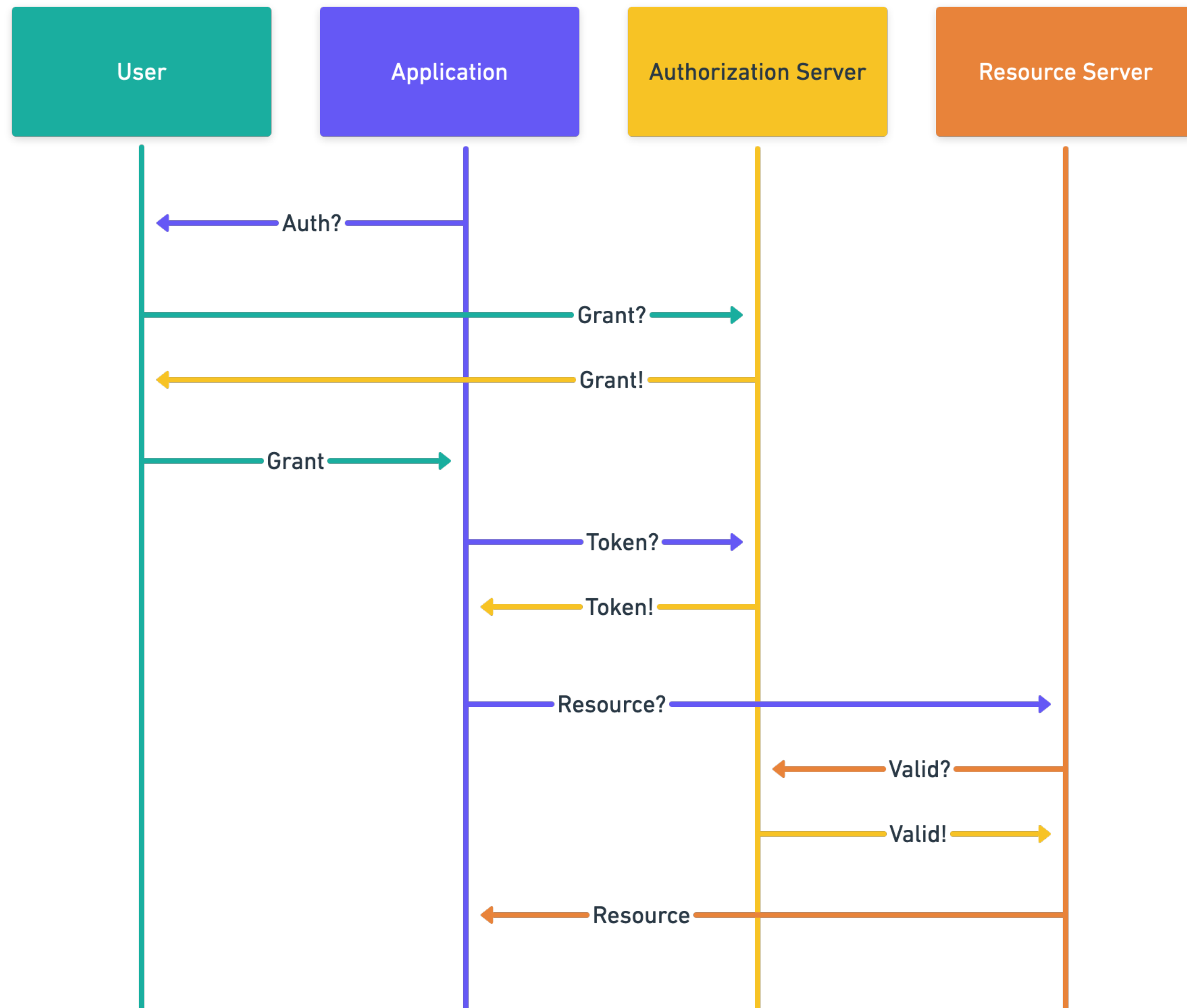
Fixing the Leaky Pipes 🚿

Faster, Stronger, More Streamlined



Fixing the Leaky Pipes

Faster, Stronger, More Streamlined



Universal, verifiable, user originated



Fixing the Leaky Pipes

JWT++

```
{
  "aud": "did:key:zStEZpzSMtTt9k2vszgvCwF4fLQSyA15W5AQ4z3AR6Bx4eFJ5crJFbuGxKmbma4",
  "iss": "did:key:z5C4fuP2DDJChhMBCwAkpYUMuJZdNWWH5NeYjUyY8btYfzDh3aHwT5picHr9Ttjq",

  "nbf": 1611204719,
  "exp": 1611300000,

  "fct": [
    {
      "sha256": "B94D27B9934D3E08A52E52D7DA7DABFAC484EFE37A5380EE9088F7ACE2EFCDE9",
      "msg": "hello world"
    }
  ]

  "att": [
    {
      "wnfs": "boris.fission.name/public/photos/",
      "cap": "OVERWRITE"
    },
    {
      "email": "boris@fission.codes",
      "cap": "SEND"
    }
  ],

  "prf": [
    "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsInVhdDI6IjAuMS4wIn0.eyJhdWQiOiJkaWQ6a2V5Omp"
  ]
}
```

Fixing the Leaky Pipes

JWT++

```
{
  "aud": "did:key:zStEZpzSMtTt9k2vszgvCwF4fL00SvA15W5A04z3AR6Bx4eFJ5crJFbuGxKmbma4",
  "iss": "did:key:z5C4fuP2DDJChhMBCwAkpYUMuJZdNWWH5NeYjUyY8btYfzDh3aHwT5picHr9Ttjq",
  "nbf": 1611204719,
  "exp": 1611300000,
  "fct": [
    {
      "sha256": "B94D27B9934D3E08A52E52D7DA7DABFAC484EFE37A5380EE9088F7ACE2EFCDE9",
      "msg": "hello world"
    }
  ]
  "att": [
    {
      "wnfs": "boris.fission.name/public/photos/",
      "cap": "OVERWRITE"
    },
    {
      "email": "boris@fission.codes",
      "cap": "SEND"
    }
  ],
  "prf": [
    "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsInVhdDI6IjAuMS4wIn0.eyJhdWQiOiJkaWQ6a2V5Omp"
  ]
}
```

Fixing the Leaky Pipes

JWT++

```
{
  "aud": "did:key:zStEZpzSMtTt9k2vszgvCwF4fL00SvA15W5A04z3AR6Bx4eFJ5crJFbuGxKmbma4",
  "iss": "did:key:z5C4fuP2DDJChhMBCwAkpYUMuJZdNWWH5NeYjUyY8btYfzDh3aHwT5picHr9Ttjq",

  "nbf": 1611204719,
  "exp": 1611300000,

  "fct": [
    {
      "sha256": "B94D27B9934D3E08A52E52D7DA7DABFAC484EFE37A5380EE9088F7ACE2EFCDE9",
      "msg": "hello world"
    }
  ]

  "att": [
    {
      "wnfs": "boris.fission.name/public/photos/"
      "cap": "OVERWRITE"
    },
    {
      "email": "boris@fission.codes",
      "cap": "SEND"
    }
  ],

  "prf": [
    "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsInVhdDI6IjAuMS4wIn0.eyJhdWQiOiJkaWQ6a2V5bnp"
  ]
}
```

Fixing the Leaky Pipes

JWT++

```
{
  "aud": "did:key:zStEZpzSMtTt9k2vszgvCwF4fL00SvA15W5A04z3AR6Bx4eFJ5crJFbuGxKmbma4",
  "iss": "did:key:z5C4fuP2DDJChhMBCwAkpYUMuJZdNWWH5NeYjUyY8btYfzDh3aHwT5picHr9Ttjq",

  "nbf": 1611204719,
  "exp": 1611300000,

  "fct": [
    {
      "sha256": "B94D27B9934D3E08A52E52D7DA7DABFAC484EFE37A5380EE9088F7ACE2EFCDE9",
      "msg": "hello world"
    }
  ]

  "att": [
    {
      "wnfs": "boris.fission.name/public/photos/"
      "cap": "OVERWRITE"
    },
    {
      "email": "boris@fission.codes",
      "cap": "SEND"
    }
  ],

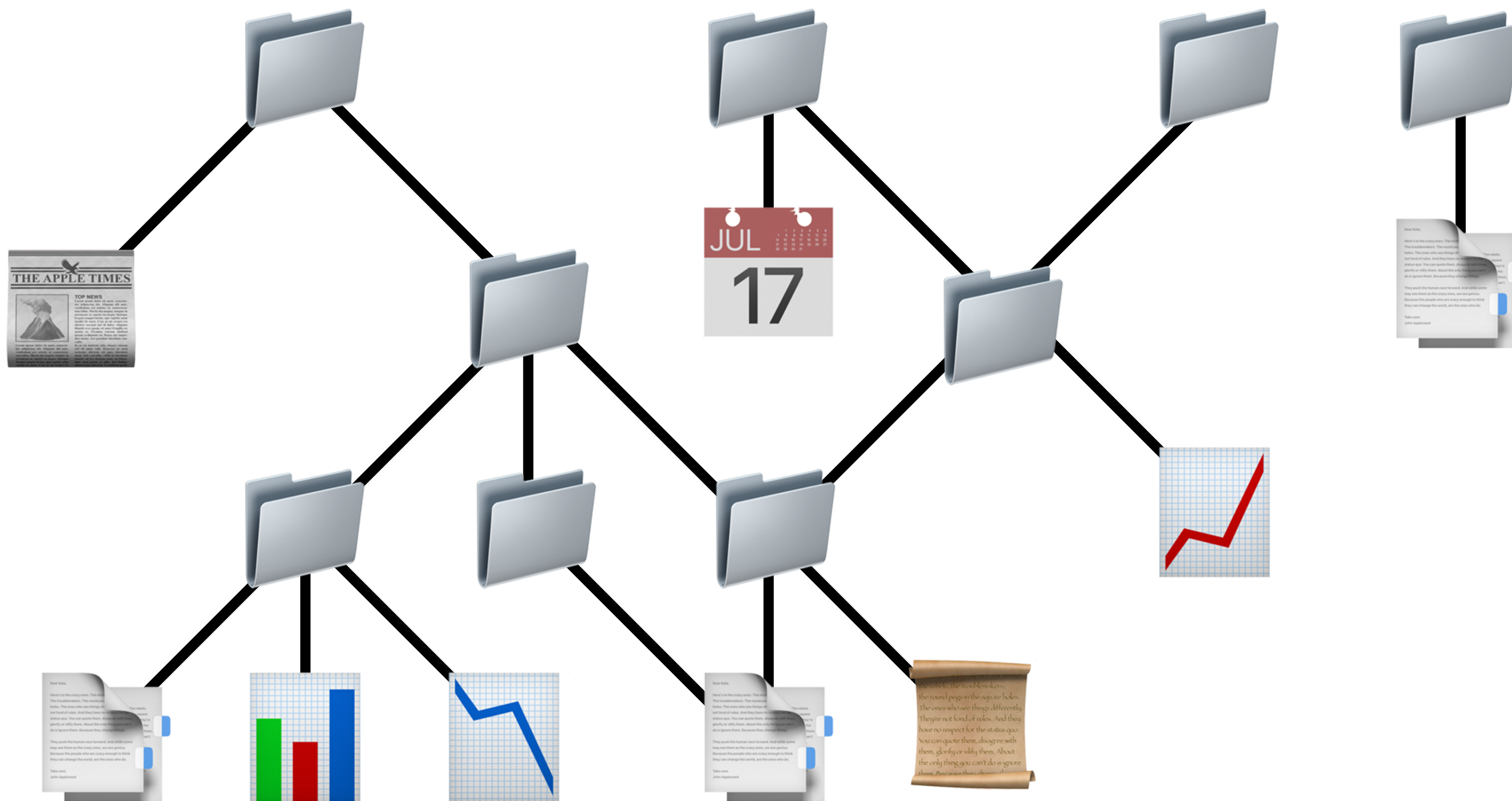
  "prf": [
    "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJkaWQ6a2V5Y8btYfzDh3aHwT5picHr9Ttjq",
    "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJkaWQ6a2V5Y8btYfzDh3aHwT5picHr9Ttjq"
  ]
}
```

Fixing the Leaky Pipes 

Reading the Universal Dataspace  

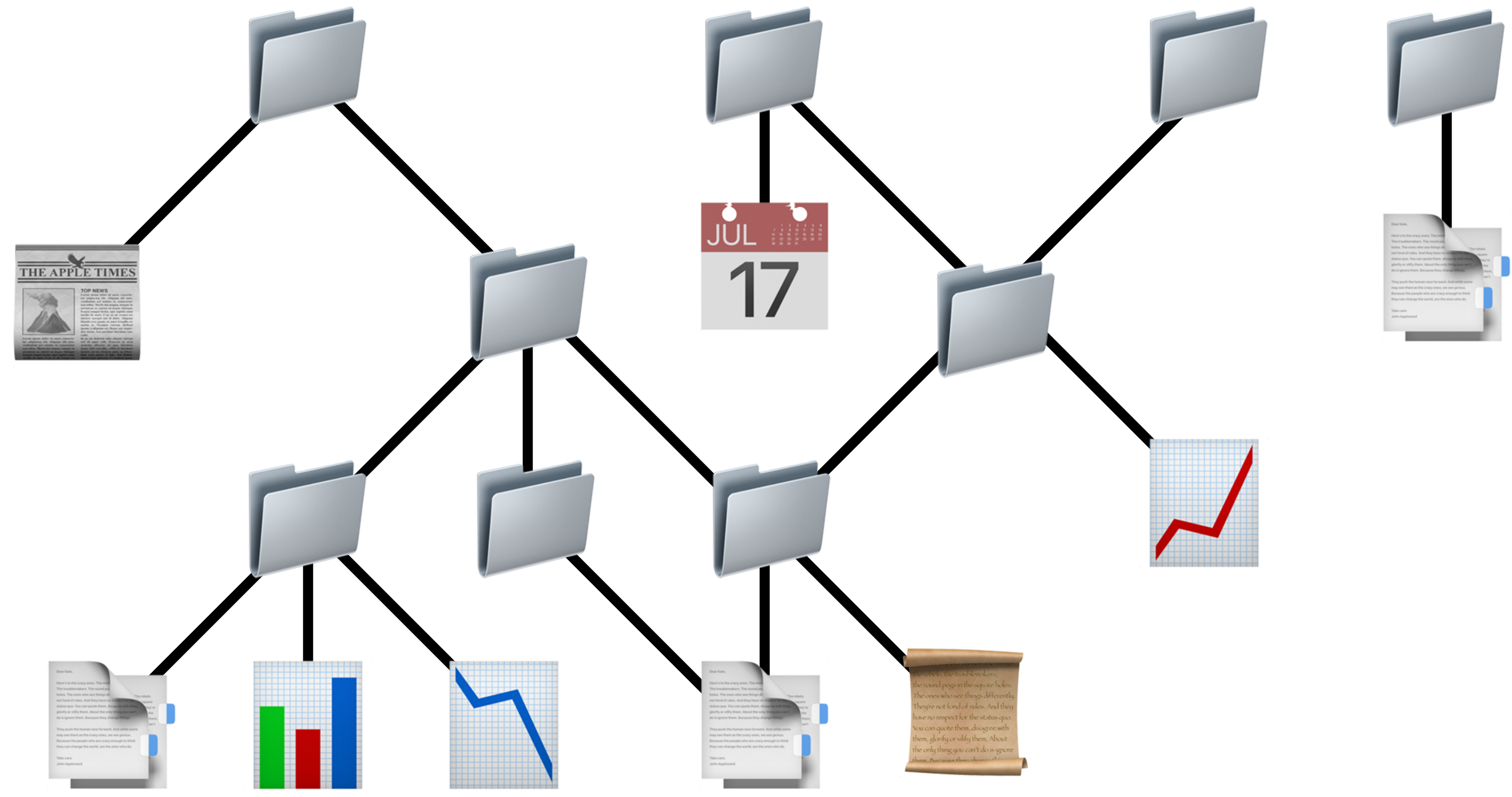
Fixing the Leaky Pipes 🚿

Reading the Universal Dataspace 🌐 🎱



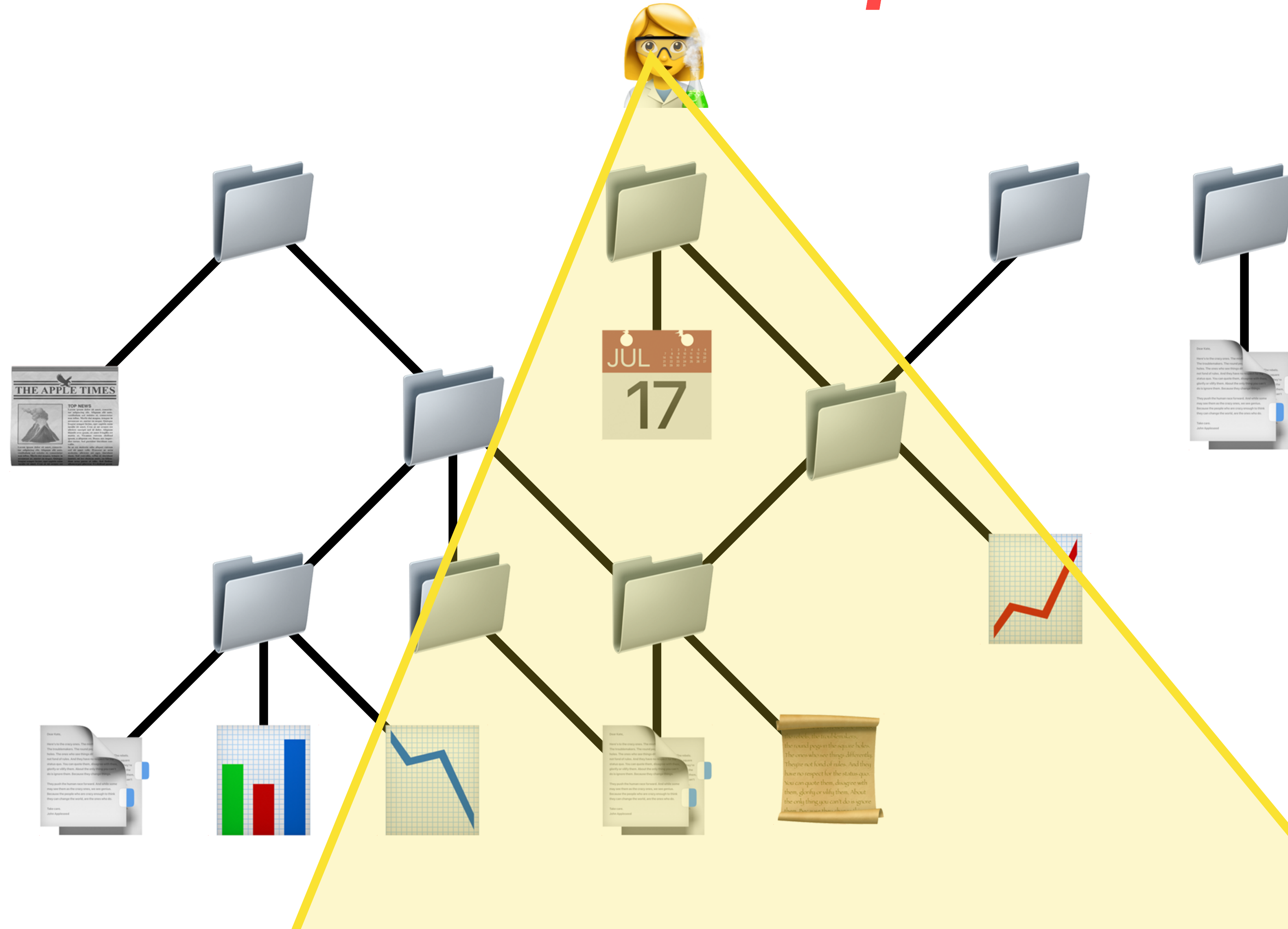
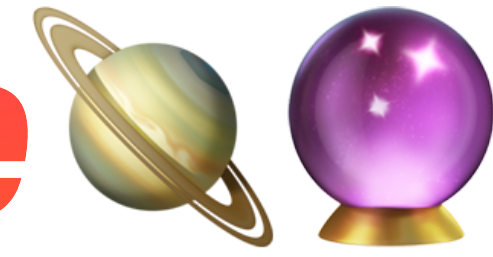
Fixing the Leaky Pipes

Reading the Universal Dataspace



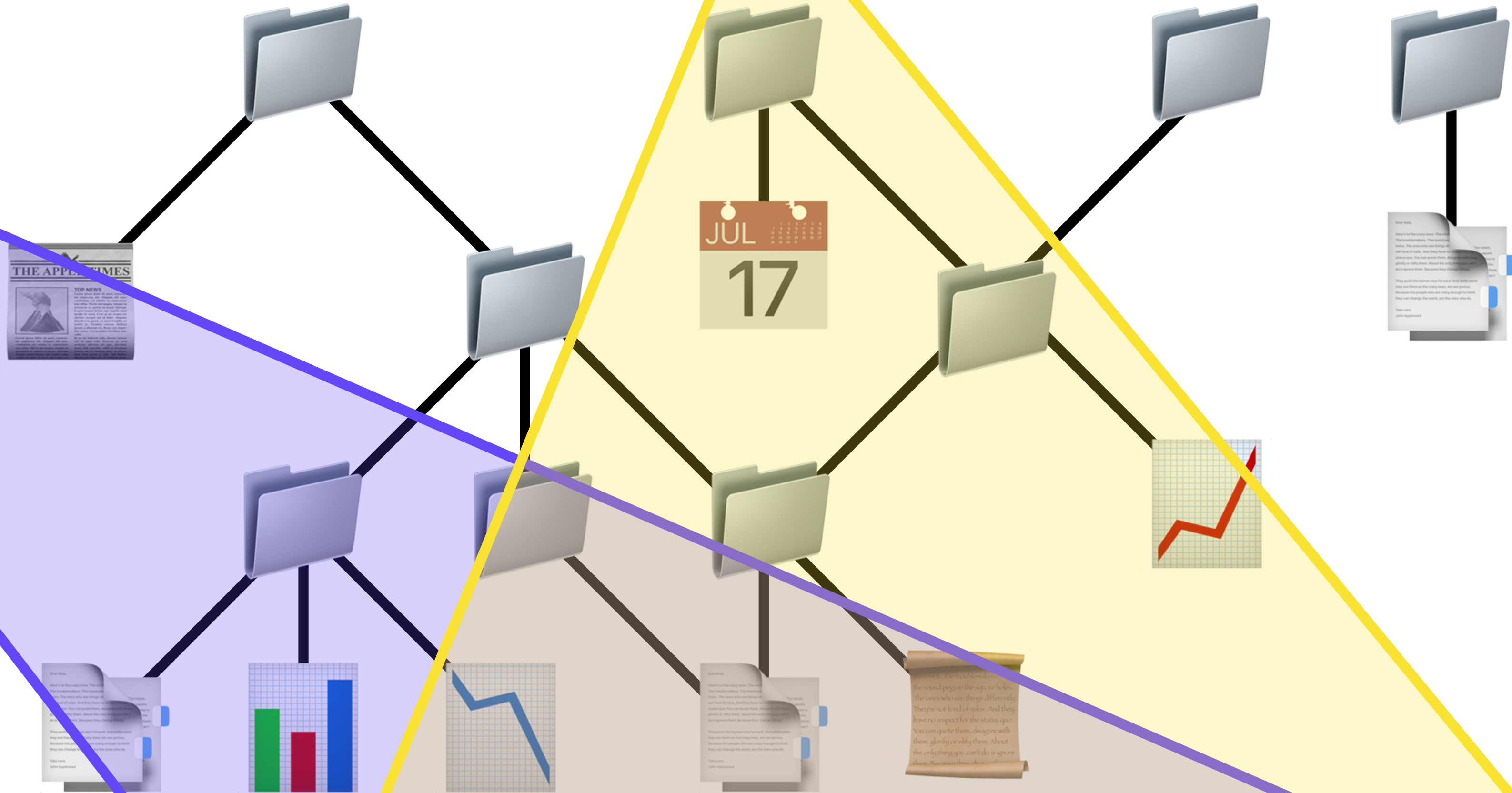
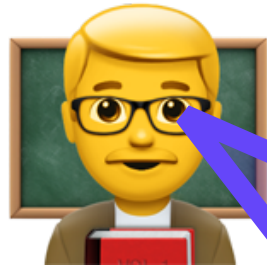
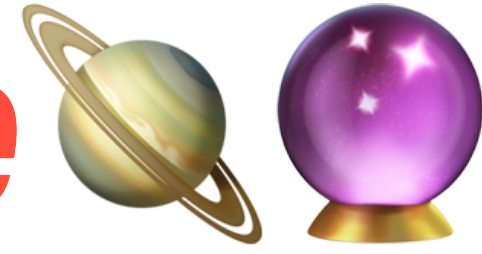
Fixing the Leaky Pipes

Reading the Universal Dataspace



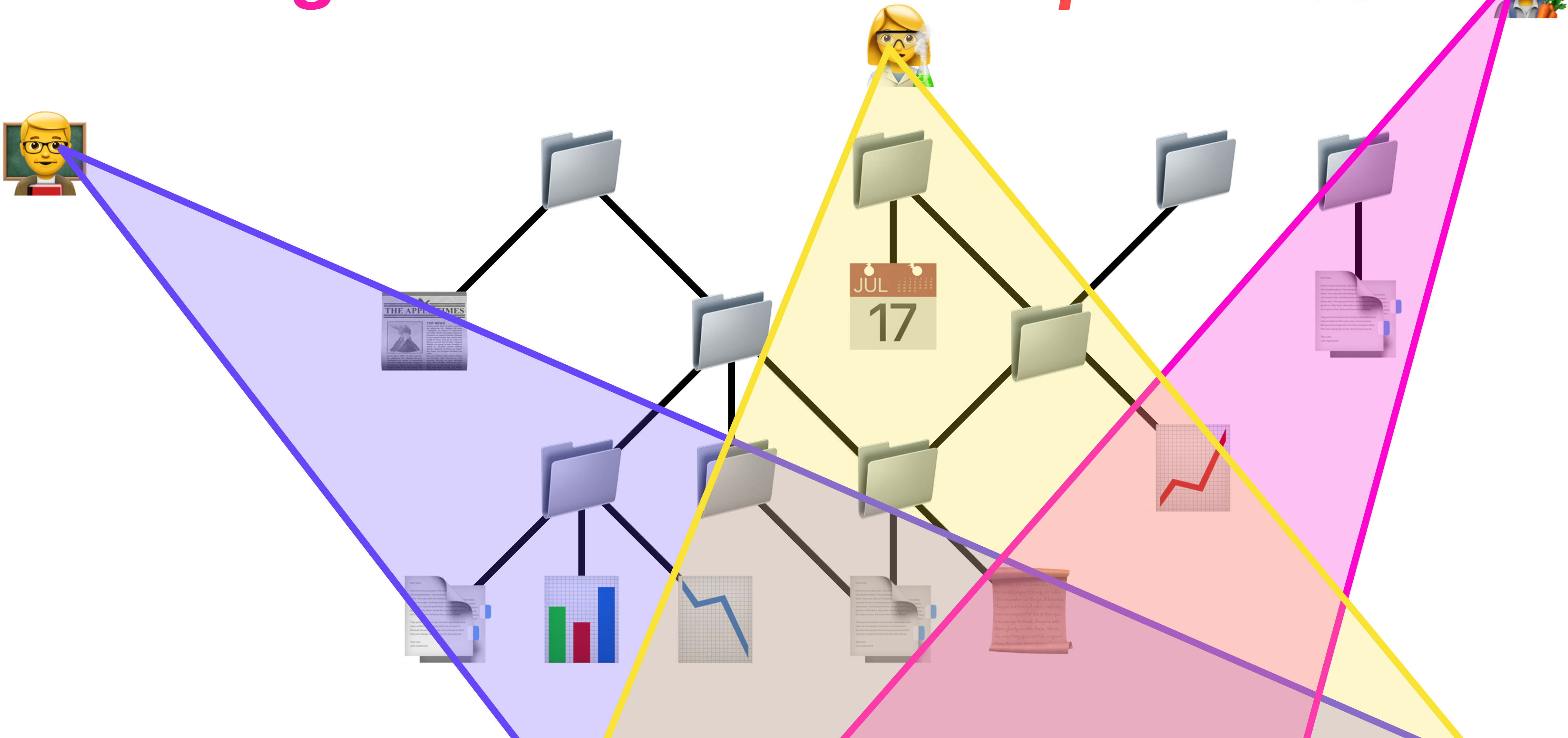
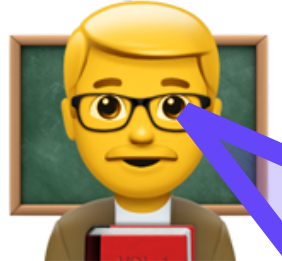
Fixing the Leaky Pipes

Reading the Universal Dataspace



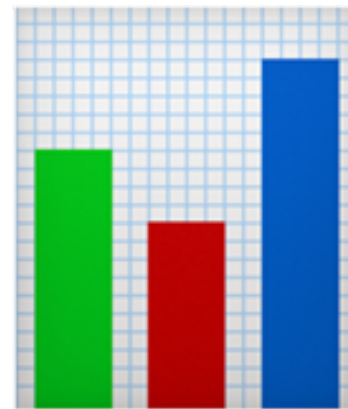
Fixing the Leaky Pipes 🚿

Reading the Universal Dataspace



It's all about that

Data, Data, Data



Data dominates. If you've chosen the right data structures and organized things well, the algorithms will almost always be self-evident.

***Data structures, not algorithms,
are central to programming.***

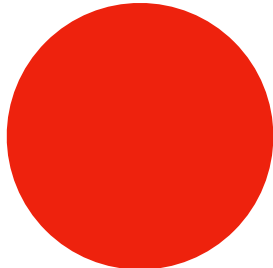
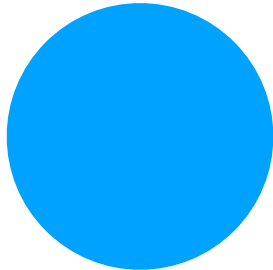
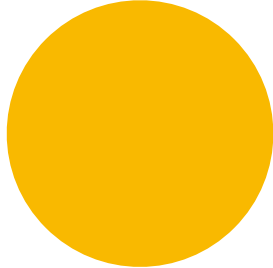
Rob Pike, 5 Rules of Programming

It's All About that Data 

Gossiping Out of Order 

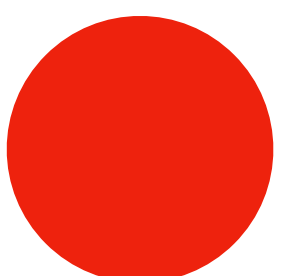
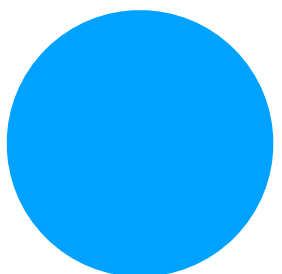
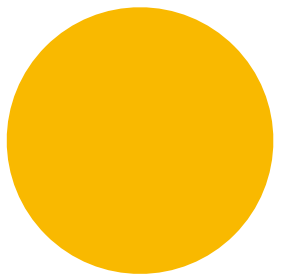
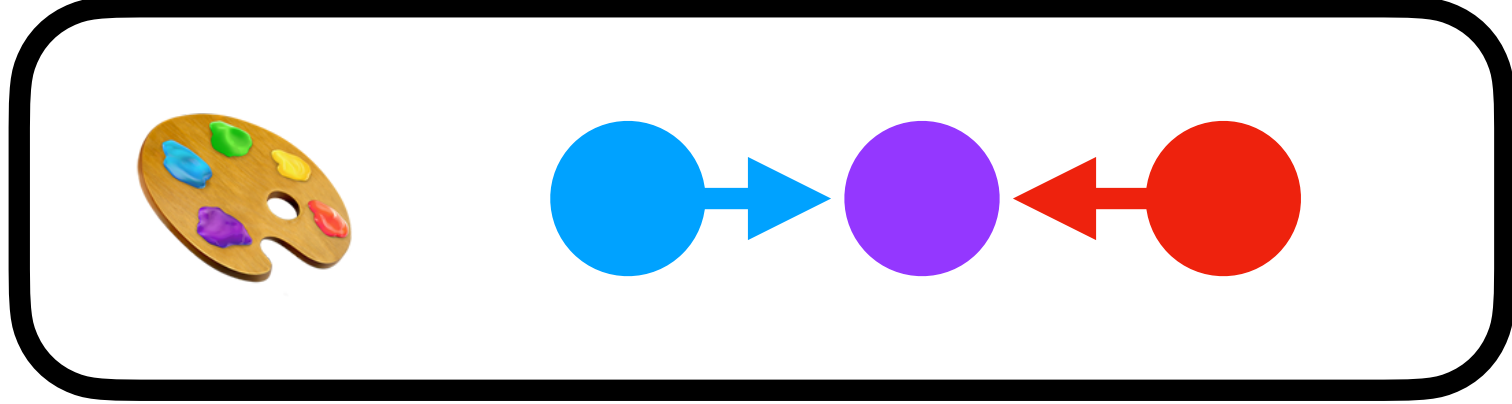
It's All About that Data 

Gossiping Out of Order



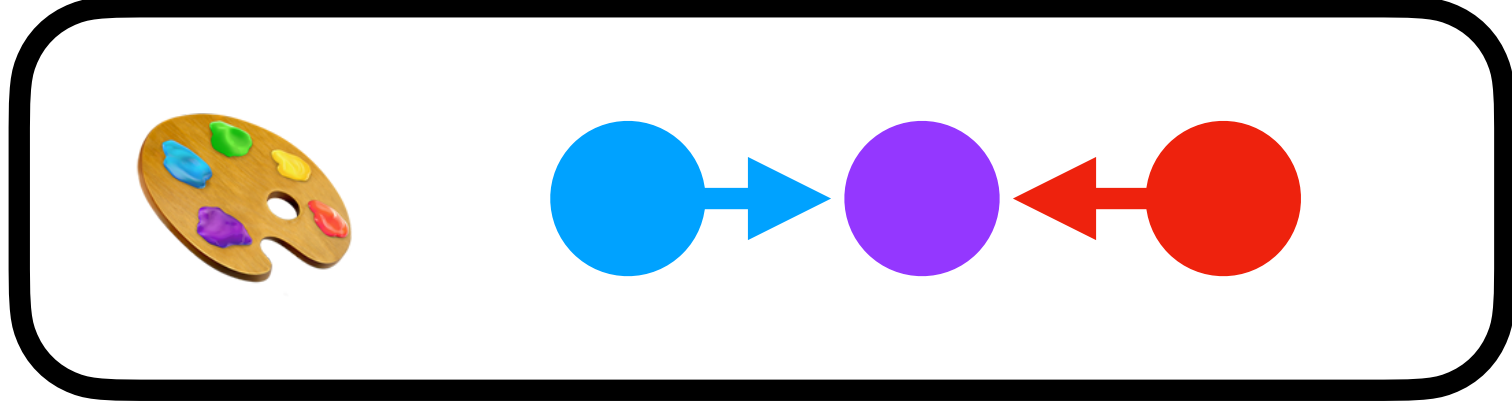
It's All About that Data 

Gossiping Out of Order



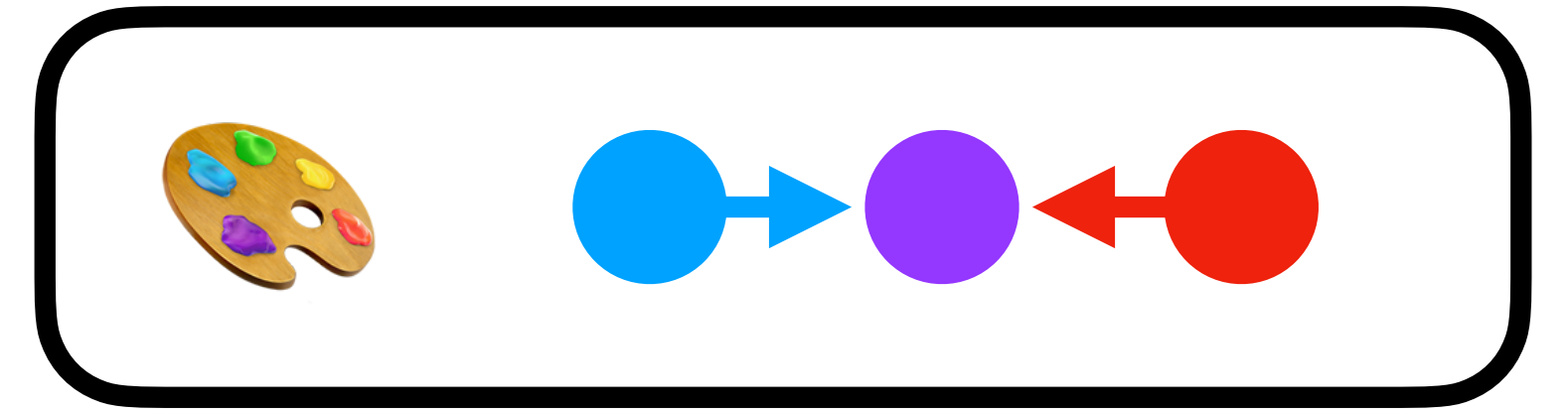
It's All About that Data 

Gossiping Out of Order



It's All About that Data 📊

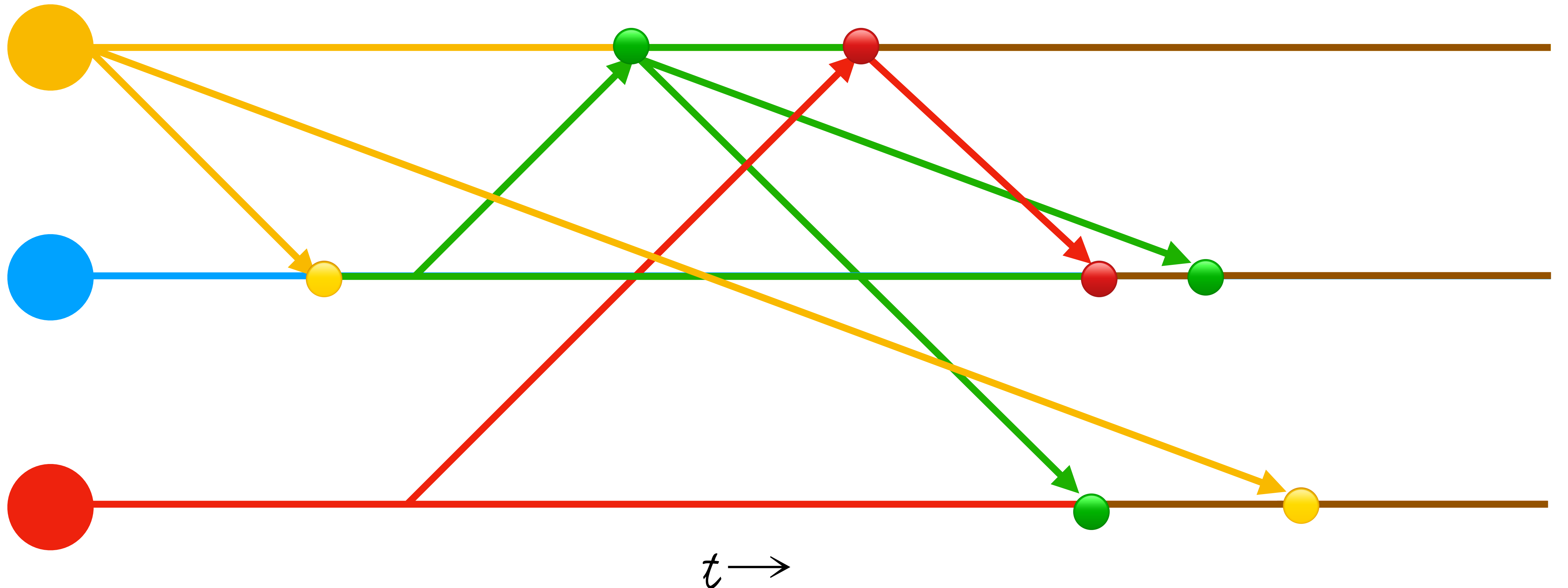
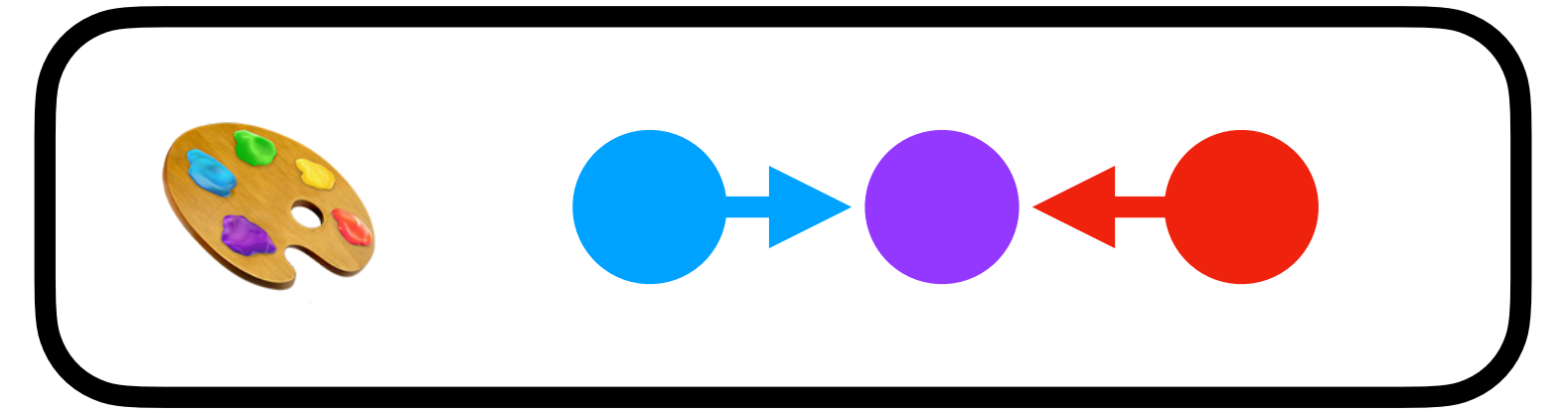
Gossiping Out of Order 🙊



$t \rightarrow$

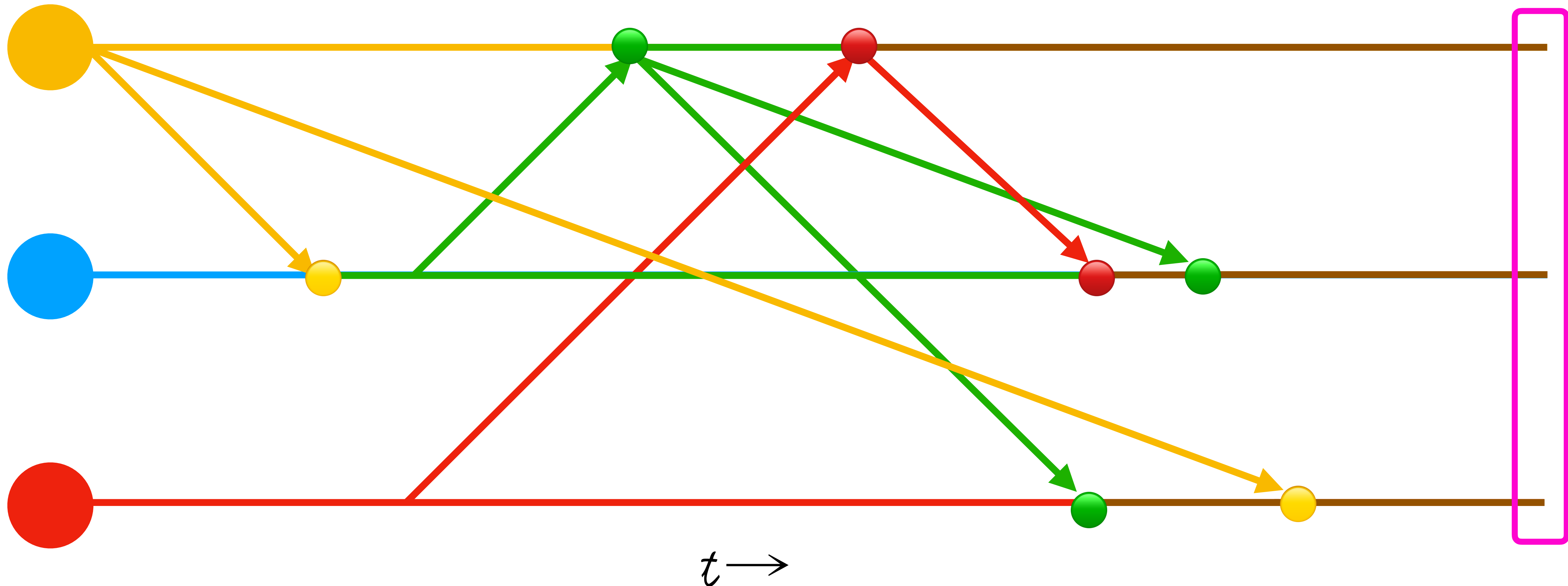
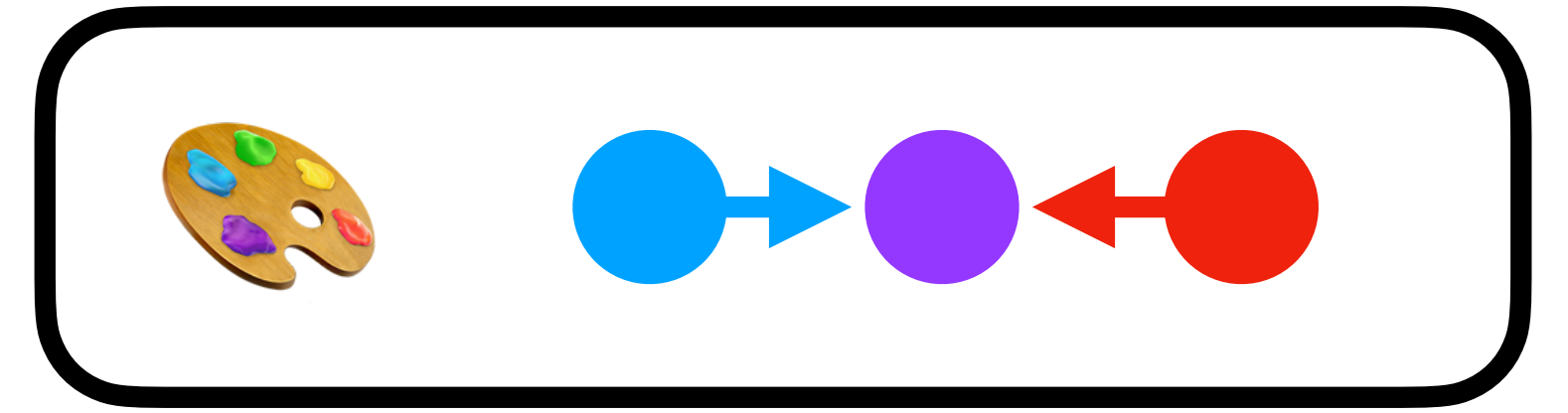
It's All About that Data 

Gossiping Out of Order



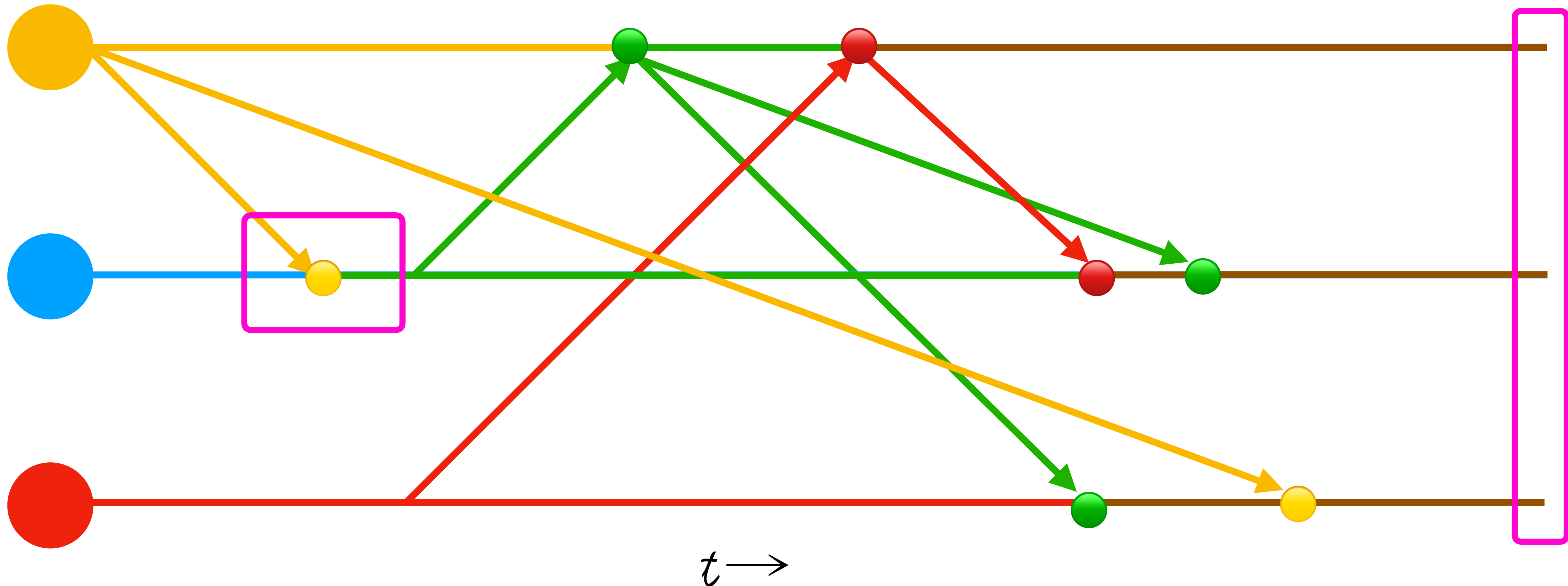
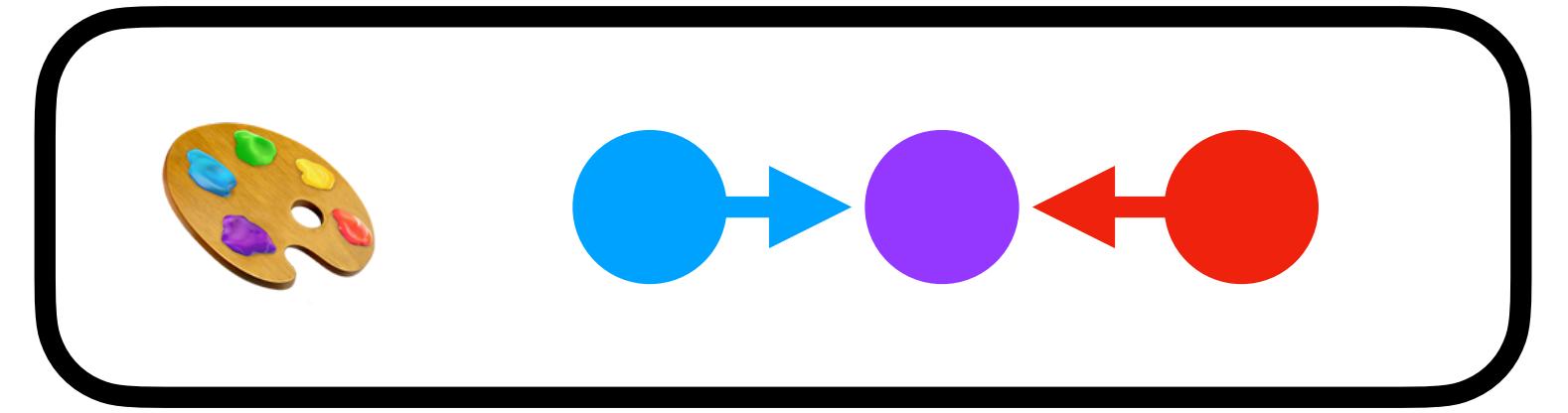
It's All About that Data 📊

Gossiping Out of Order 🙈



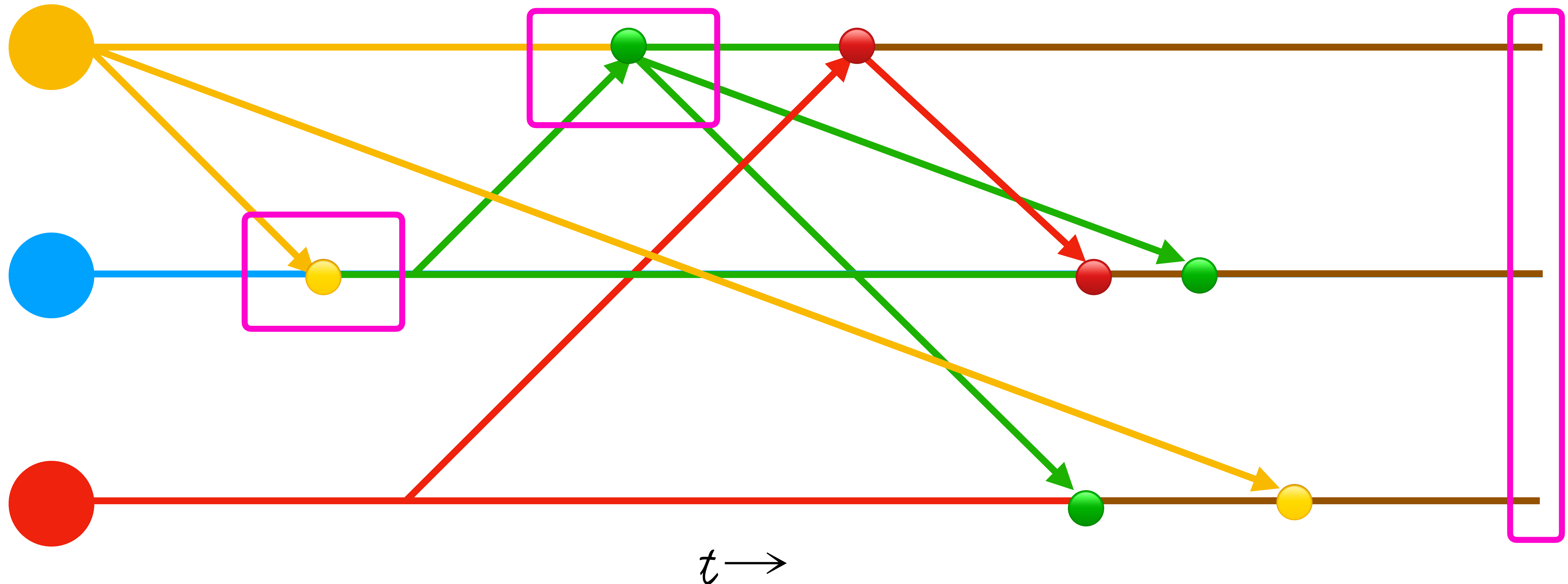
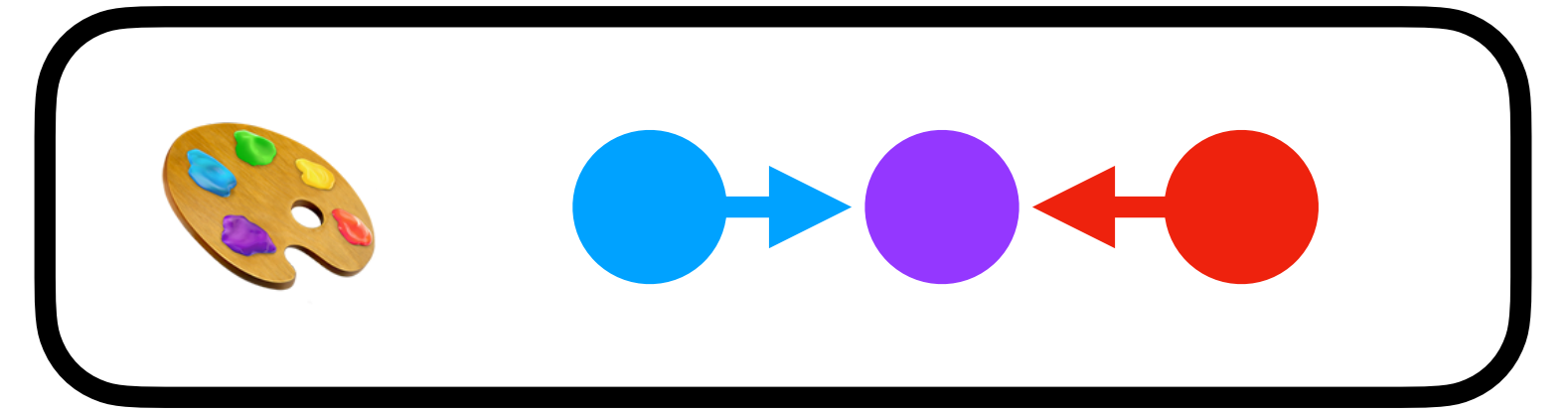
It's All About that Data 📊

Gossiping Out of Order 🙈



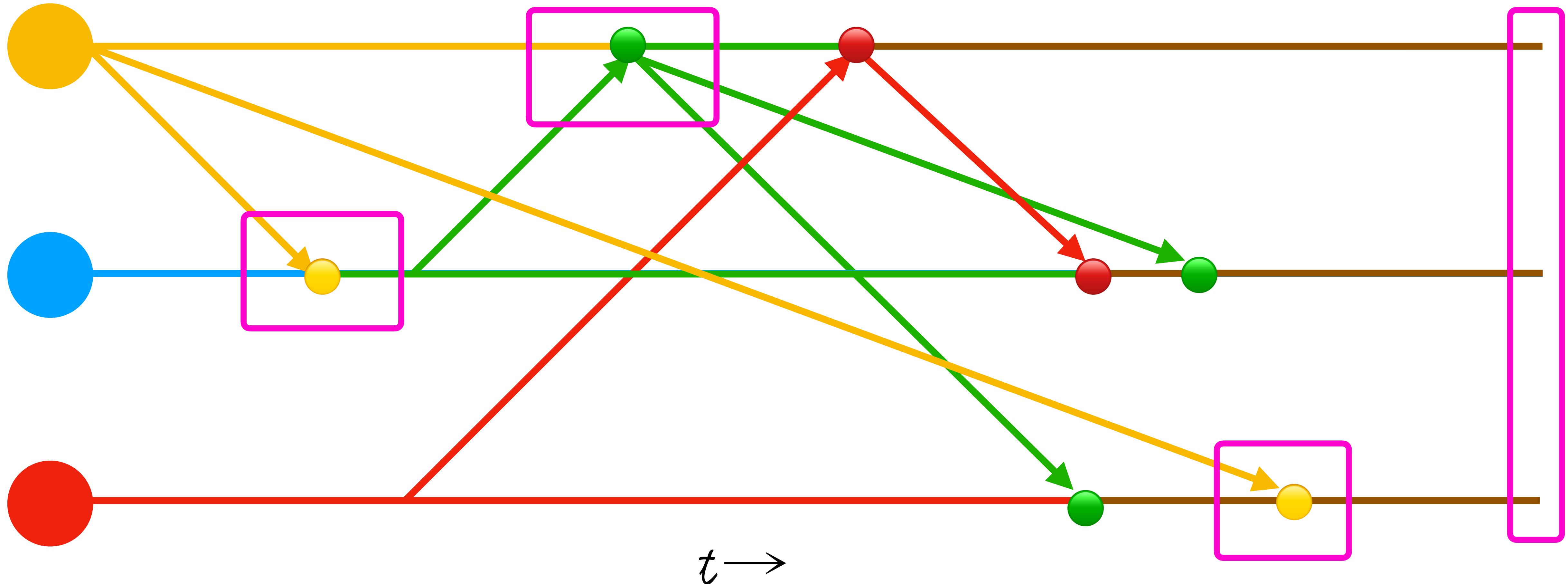
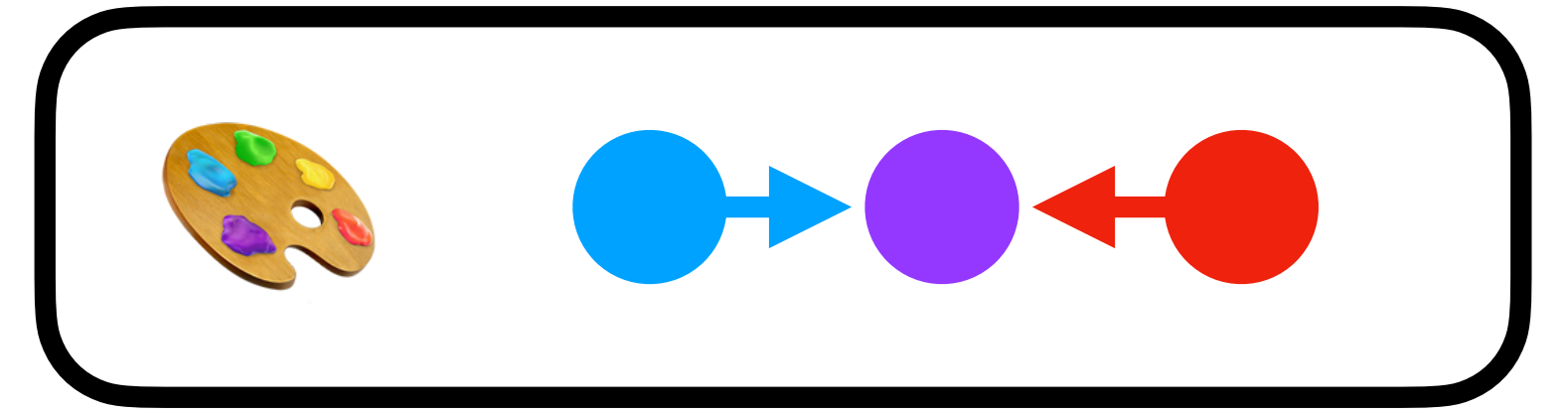
It's All About that Data 📊

Gossiping Out of Order 🙈



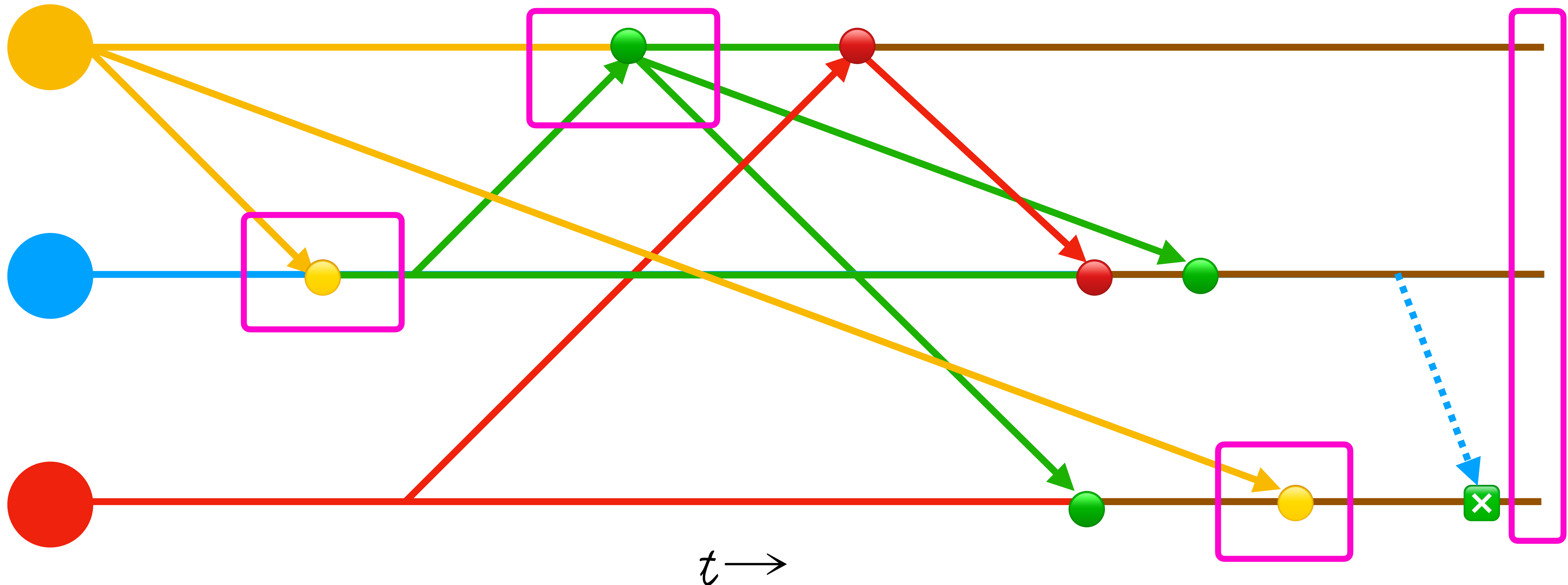
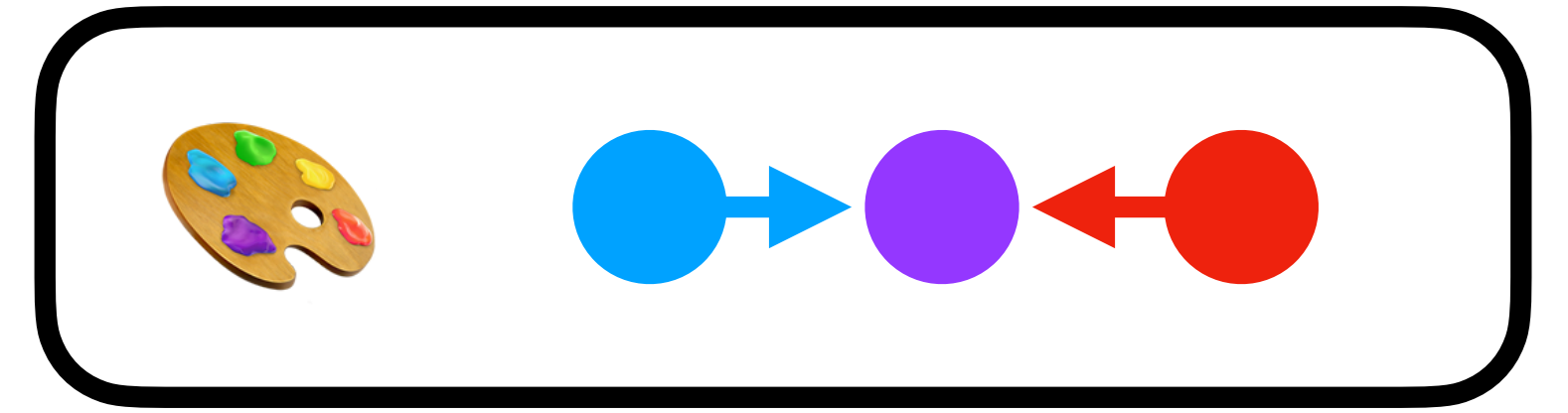
It's All About that Data 📊

Gossiping Out of Order 🙈



It's All About that Data 📊

Gossiping Out of Order 🙈



It's All About that Data 

Entropy Isn't What It Used To Be

It's All About that Data 📊

Entropy Isn't What It Used To Be

B E C A U S E

$\phi \Delta I_c$
 $t \frac{\Delta I_B}{\Delta I_A}$
 $\Sigma = n^{-1} f_n$
 $x \geq 8$
 ∞
 $P = 4a$
 $\left(\frac{x+2}{x-2}\right)$

$\frac{AB}{N \rightarrow R}$

$[1]^2$

$\sqrt{25}$
 \sin
 $P(x)$
 $\frac{P(x)}{Q(x)}$

A
 $v=2$
 $a^2+b^2=c^2$
 $X_{n+1} \dots$

$\int \frac{a^x}{b^x} dx$
 $y = mx$
 $\Delta = 4ac - b^2$
 x^n
 $E = mc^2$
 ϵtg
 3.14

$\log(x)$

\lim

π
 $x - \sqrt{x}$

ϕ
 $\sqrt{2}$

$\frac{m}{161\%}$

$\begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix}$

$e^x = 1$
 $\pm \alpha n!$

1001
0101

$a \neq 0$

$\sqrt{x^2 + 9}$

$\{m, n, j\}$

$I_0 \sqrt{2}$

$h \text{ctg}$
 $\cos(-x)$
 $\frac{dF}{dx}$
 $\frac{d}{dx} x^{-1}$
 $A \text{kat}$

$X_m \phi_1$

$\frac{2y}{x}$
 $\frac{1}{x}$
 $\frac{1}{x^2}$
 $\frac{1}{x^3}$
 $\frac{1}{x^4}$
 $\frac{1}{x^5}$
 $\frac{1}{x^6}$
 $\frac{1}{x^7}$
 $\frac{1}{x^8}$
 $\frac{1}{x^9}$
 $\frac{1}{x^{10}}$
 $\frac{1}{x^{11}}$
 $\frac{1}{x^{12}}$
 $\frac{1}{x^{13}}$
 $\frac{1}{x^{14}}$
 $\frac{1}{x^{15}}$
 $\frac{1}{x^{16}}$
 $\frac{1}{x^{17}}$
 $\frac{1}{x^{18}}$
 $\frac{1}{x^{19}}$
 $\frac{1}{x^{20}}$

4
 $f(x)$
 $2\pi r$

It's All About that Data 

Abstract

- Network agnostic
- Any number of replicas

It's All About that Data 

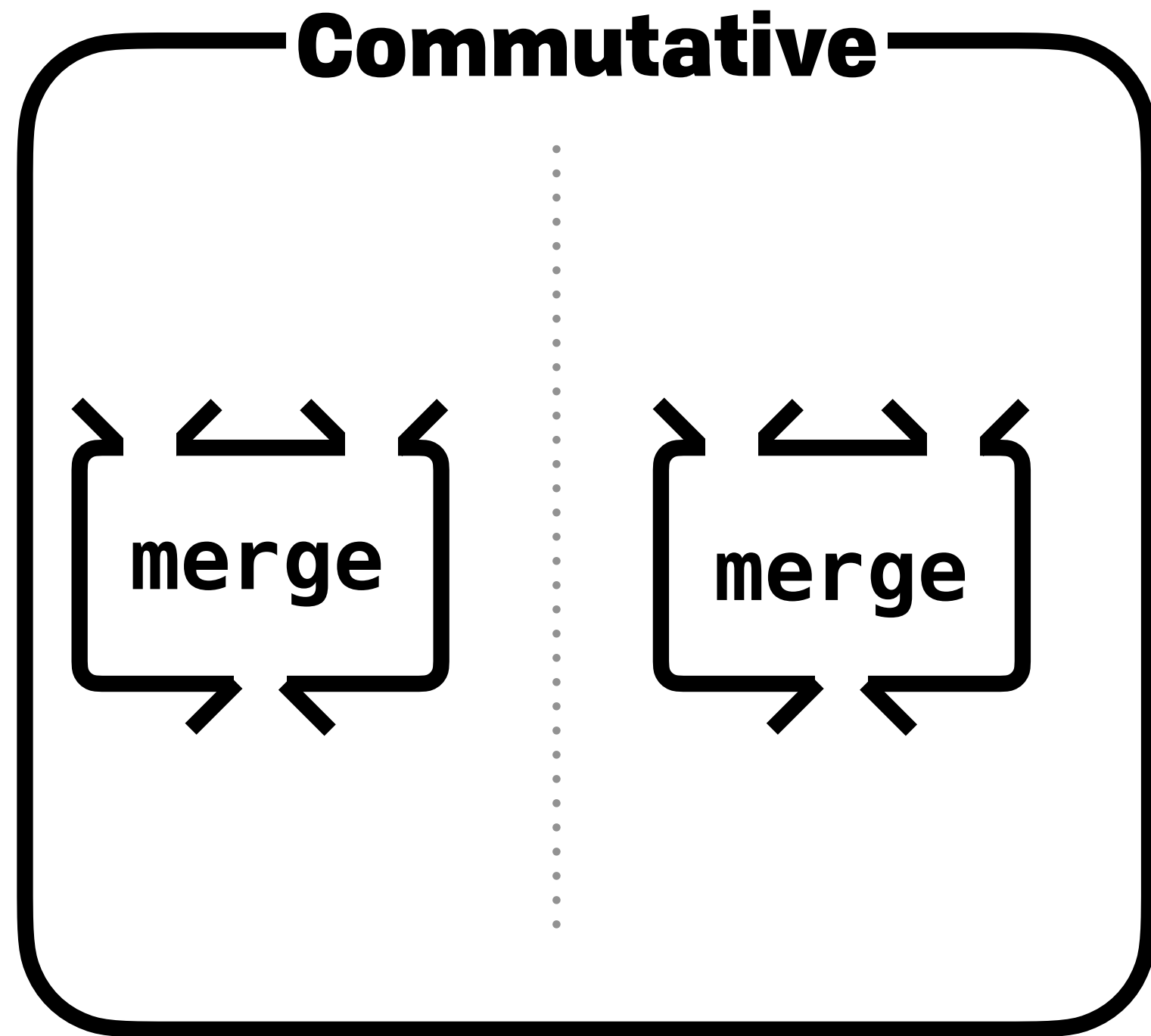
Abstract

Commutative

- Network agnostic
- Any number of replicas

It's All About that Data 📊

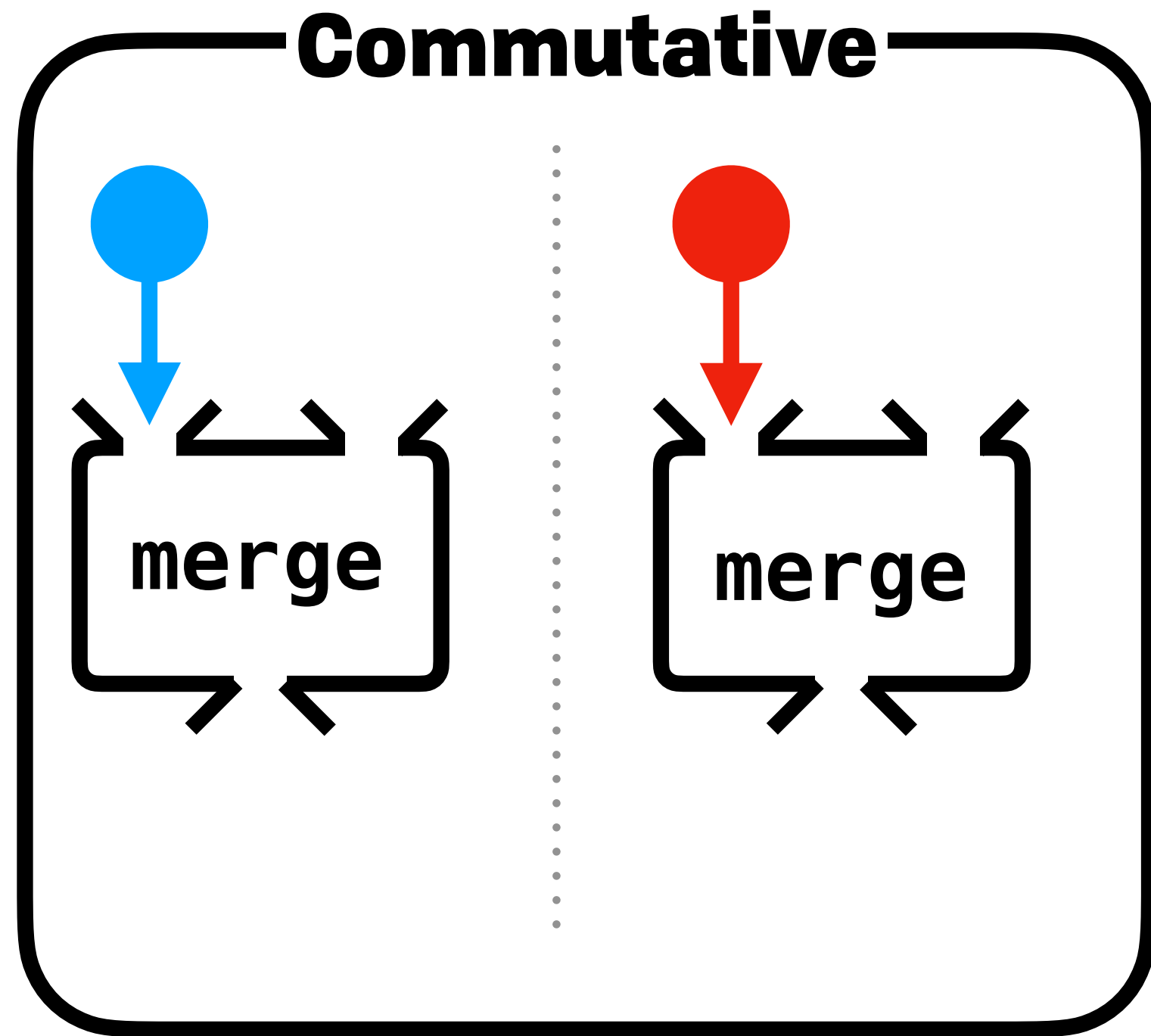
Abstract



- Network agnostic
- Any number of replicas

It's All About that Data 📊

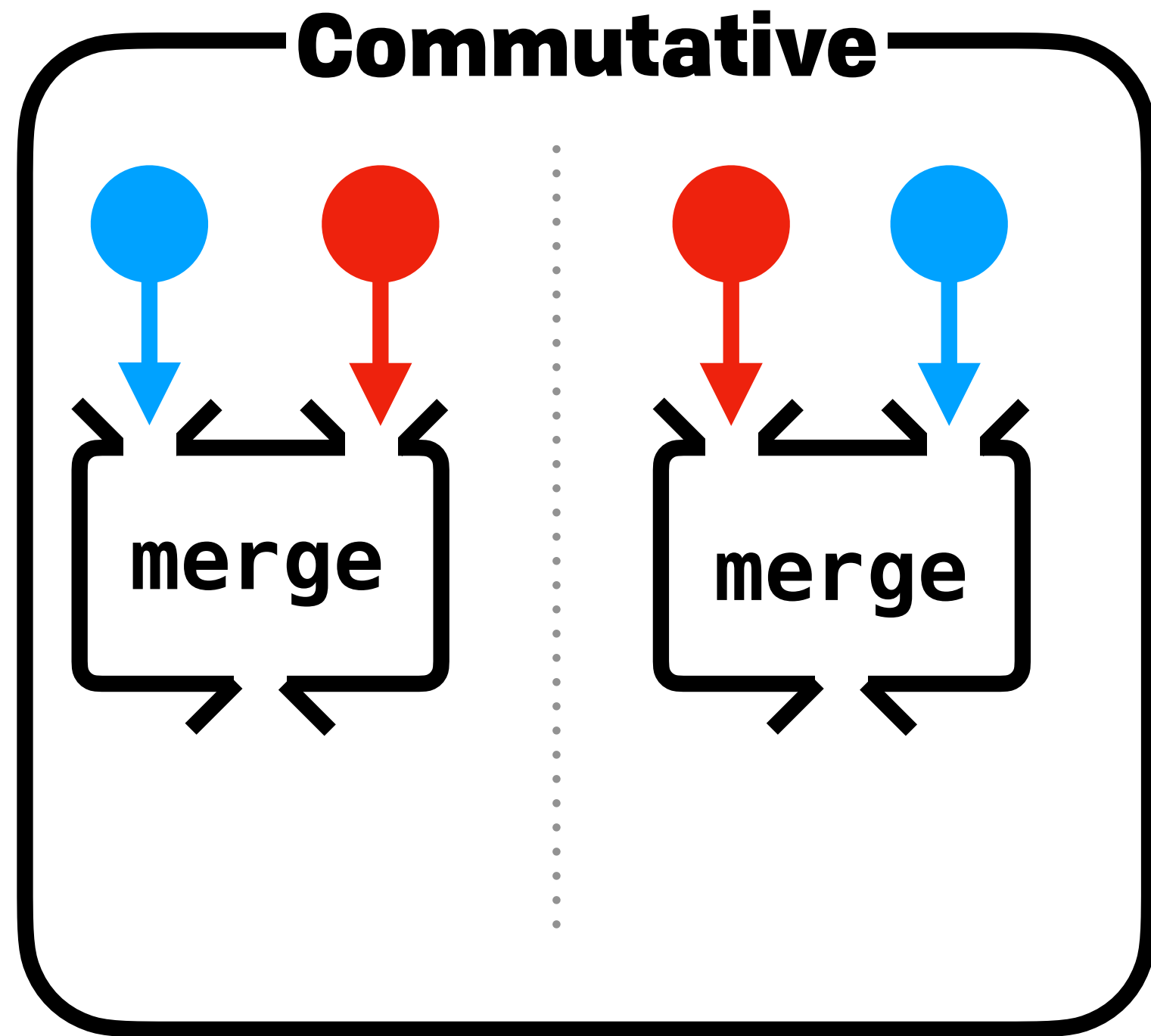
Abstract



- Network agnostic
- Any number of replicas

It's All About that Data 📊

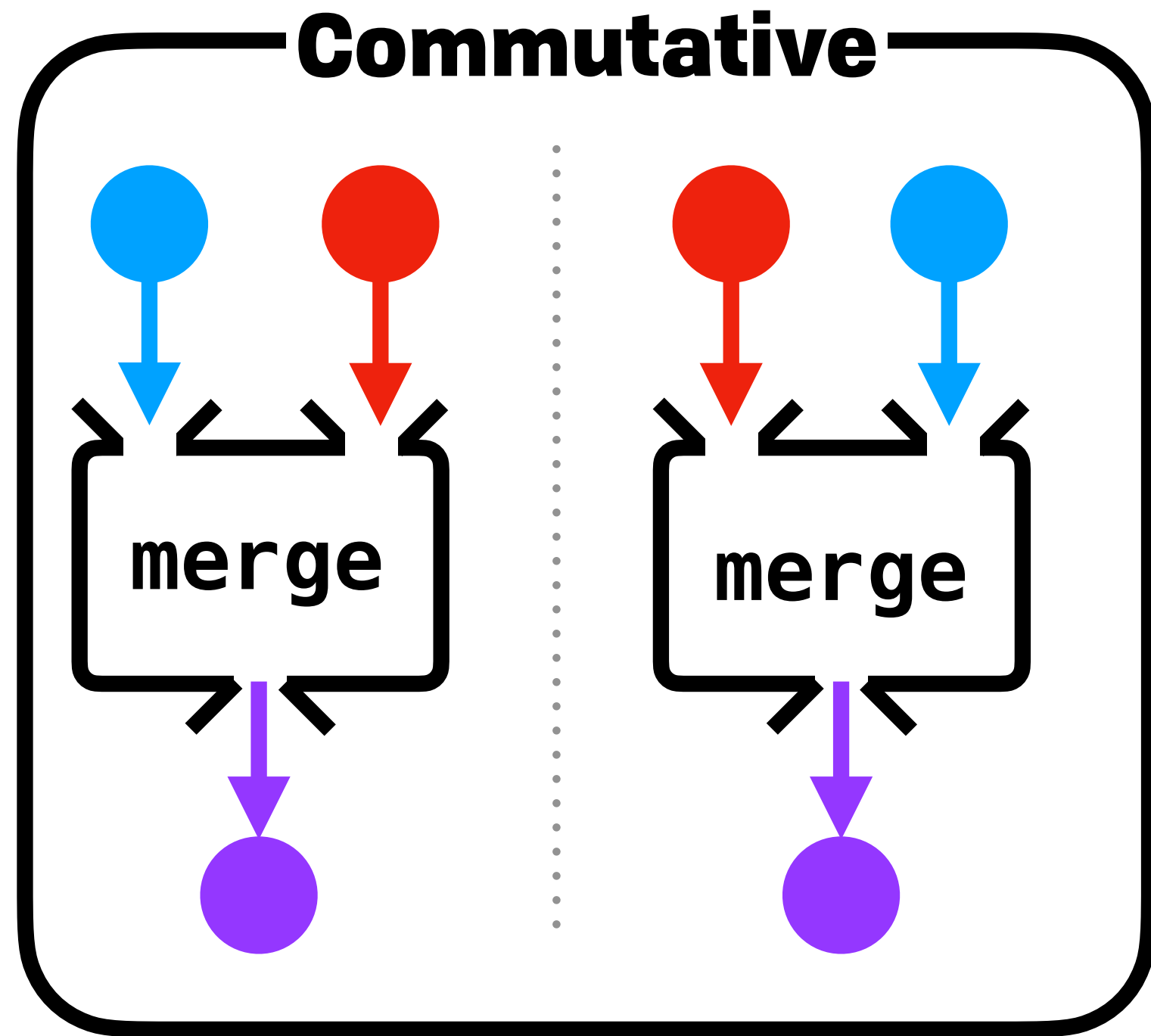
Abstract



- Network agnostic
- Any number of replicas

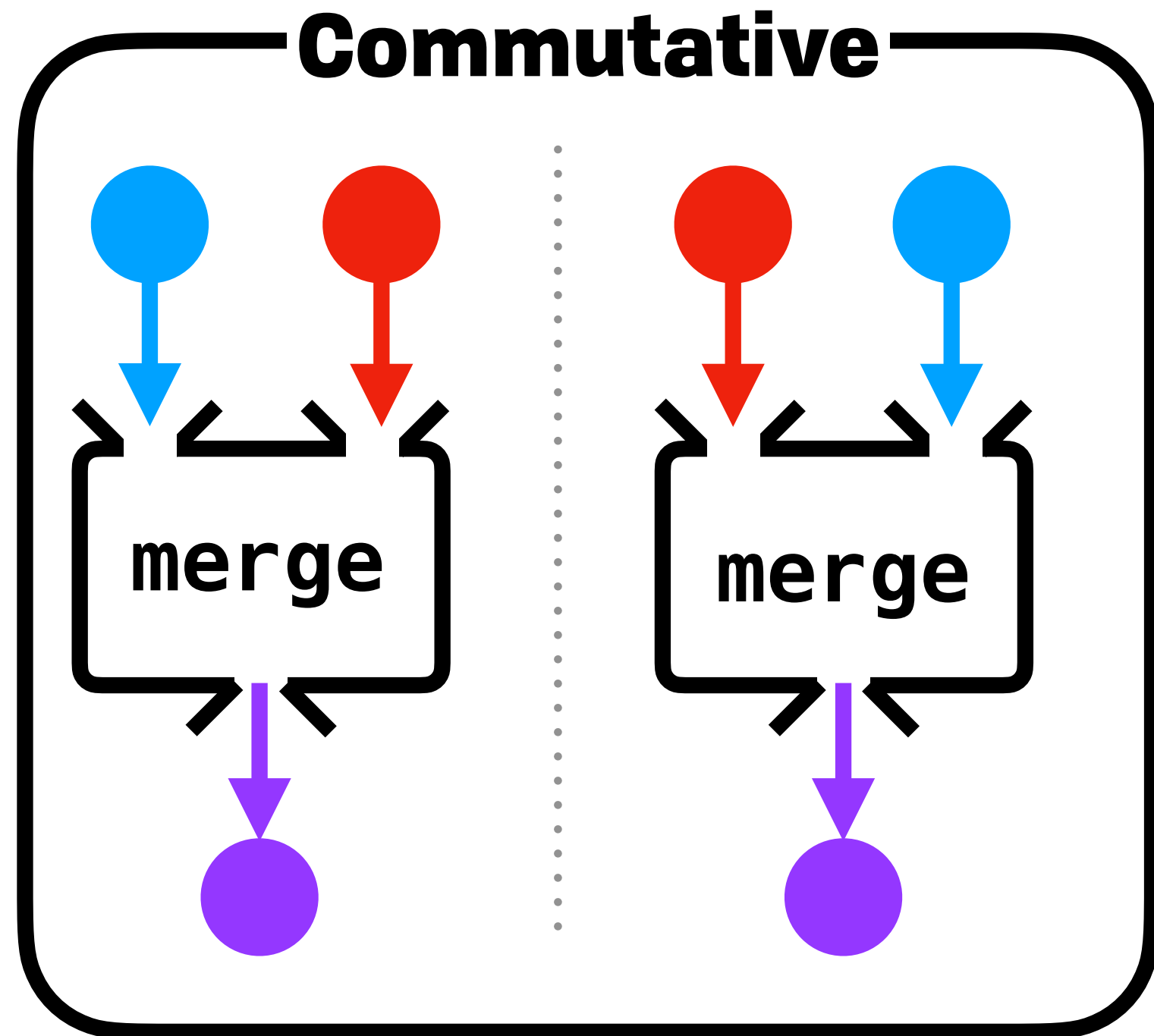
It's All About that Data 📊

Abstract

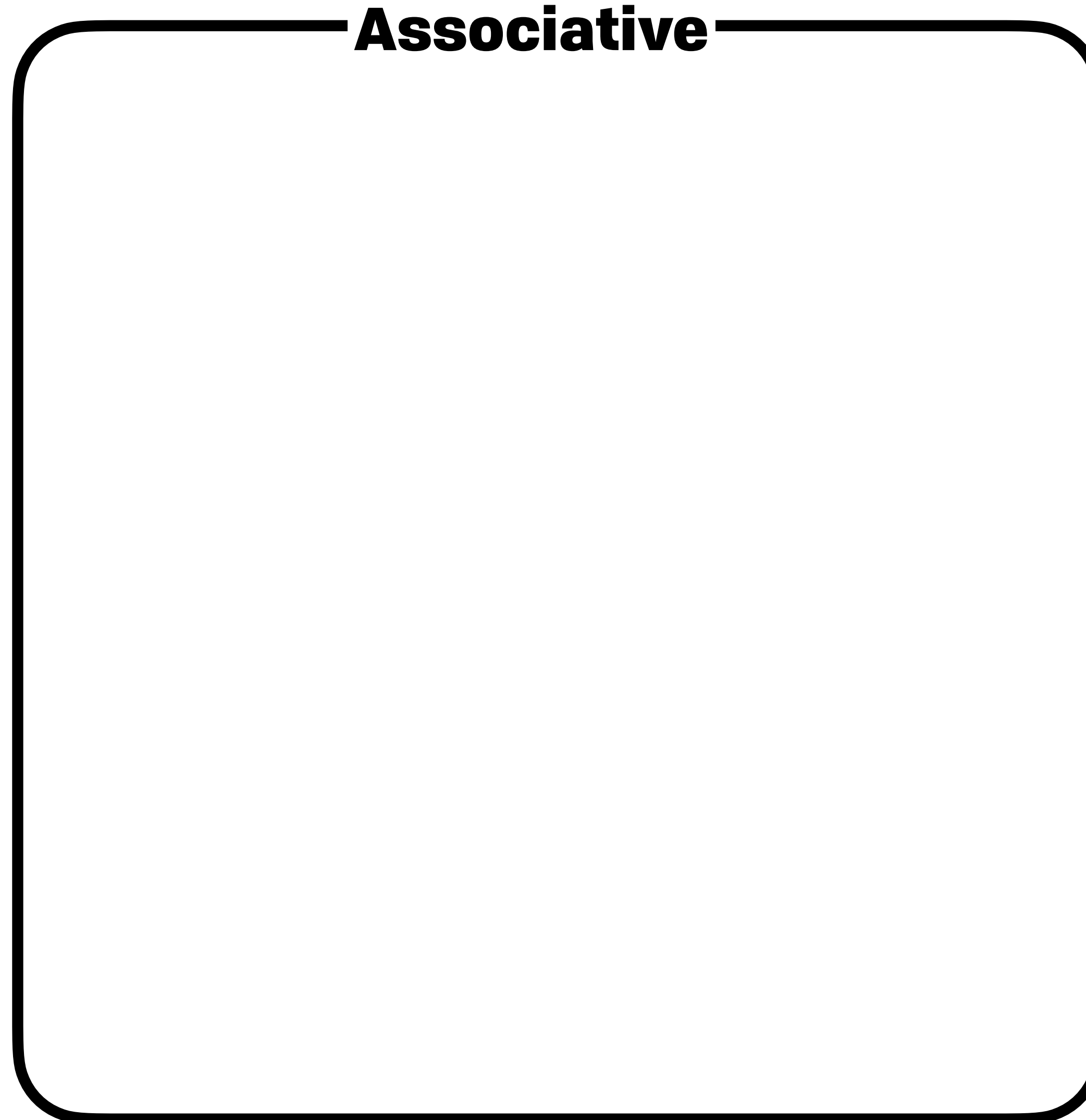


- Network agnostic
- Any number of replicas

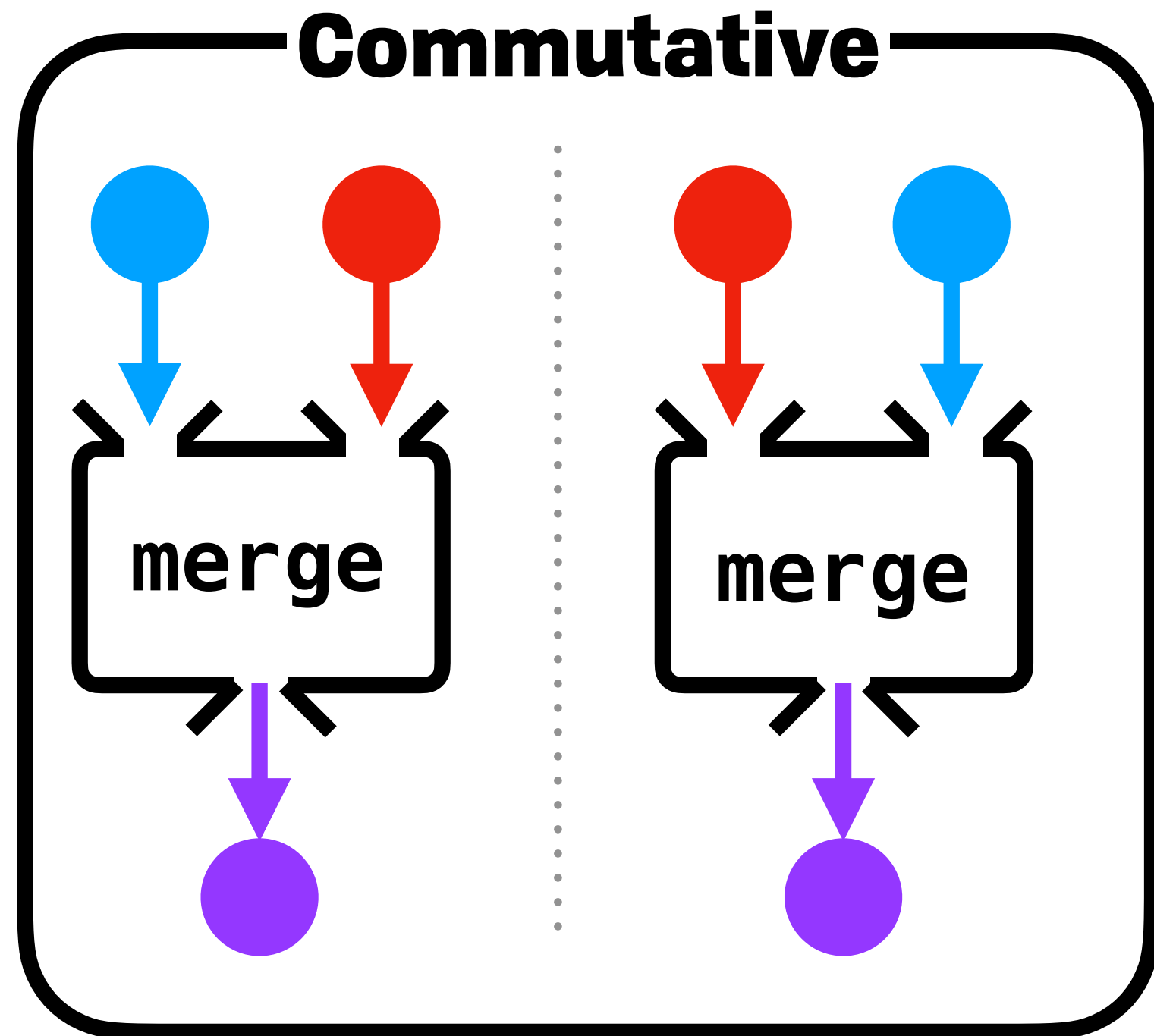
Abstract



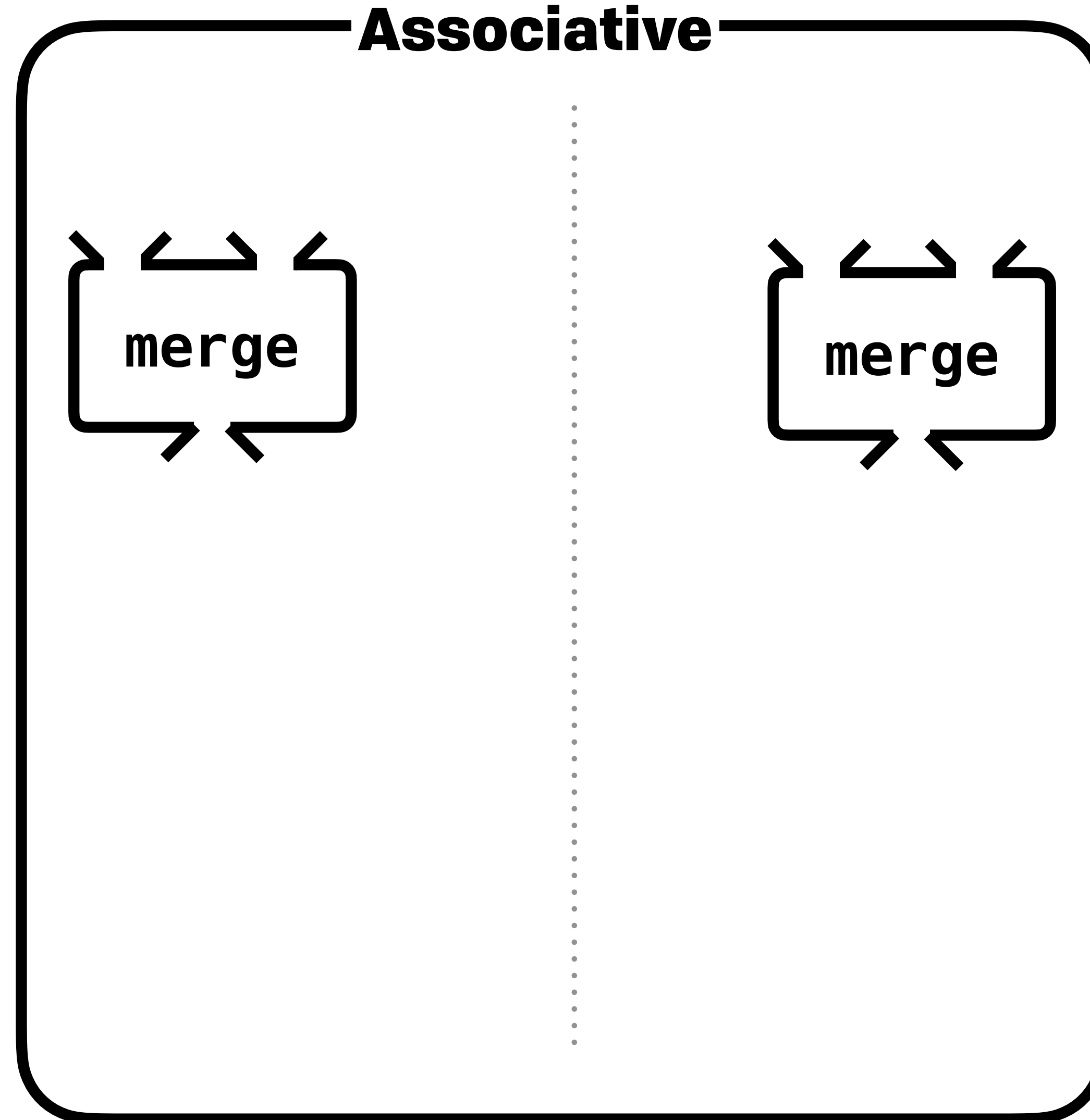
- Network agnostic
- Any number of replicas



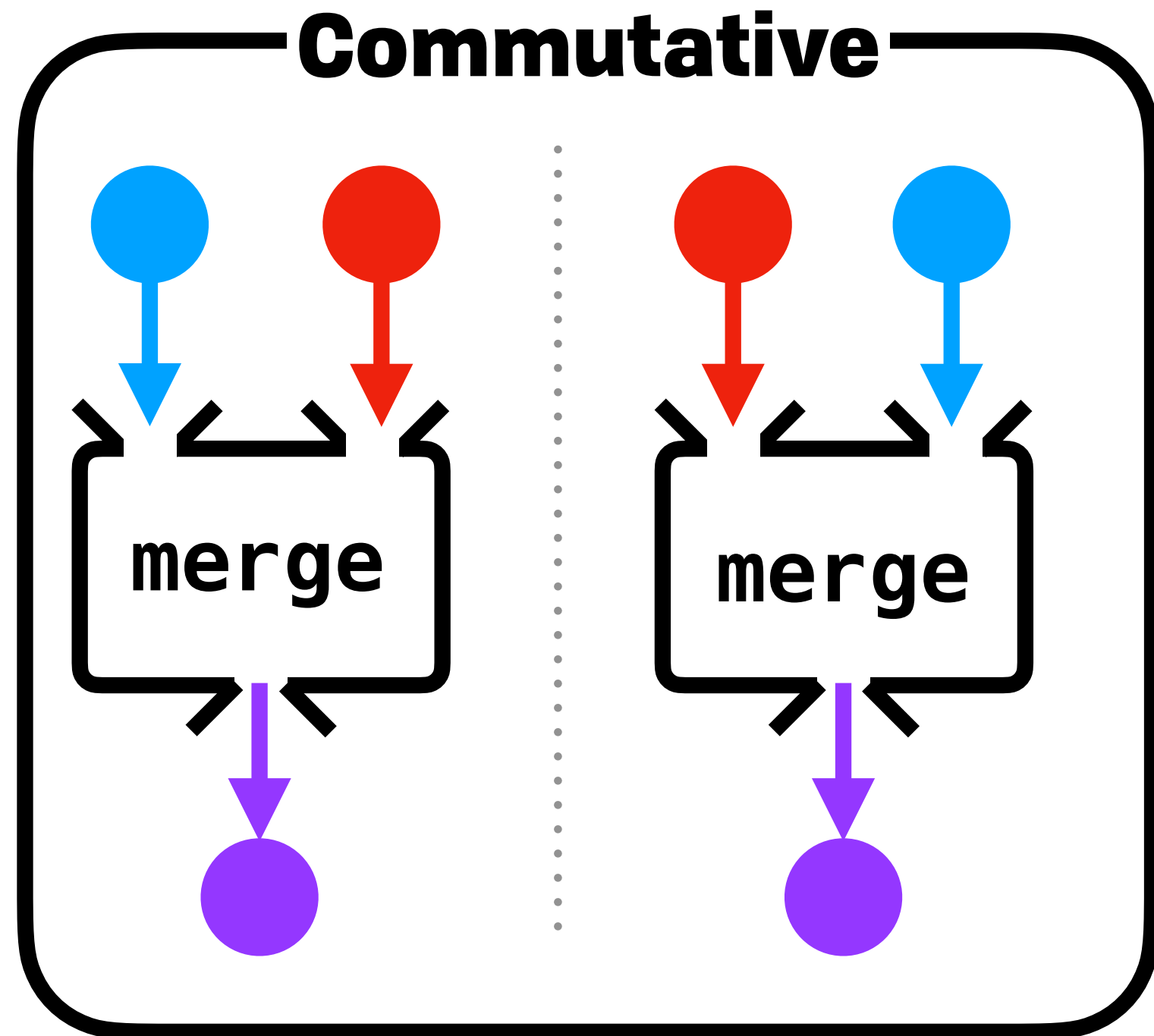
Abstract



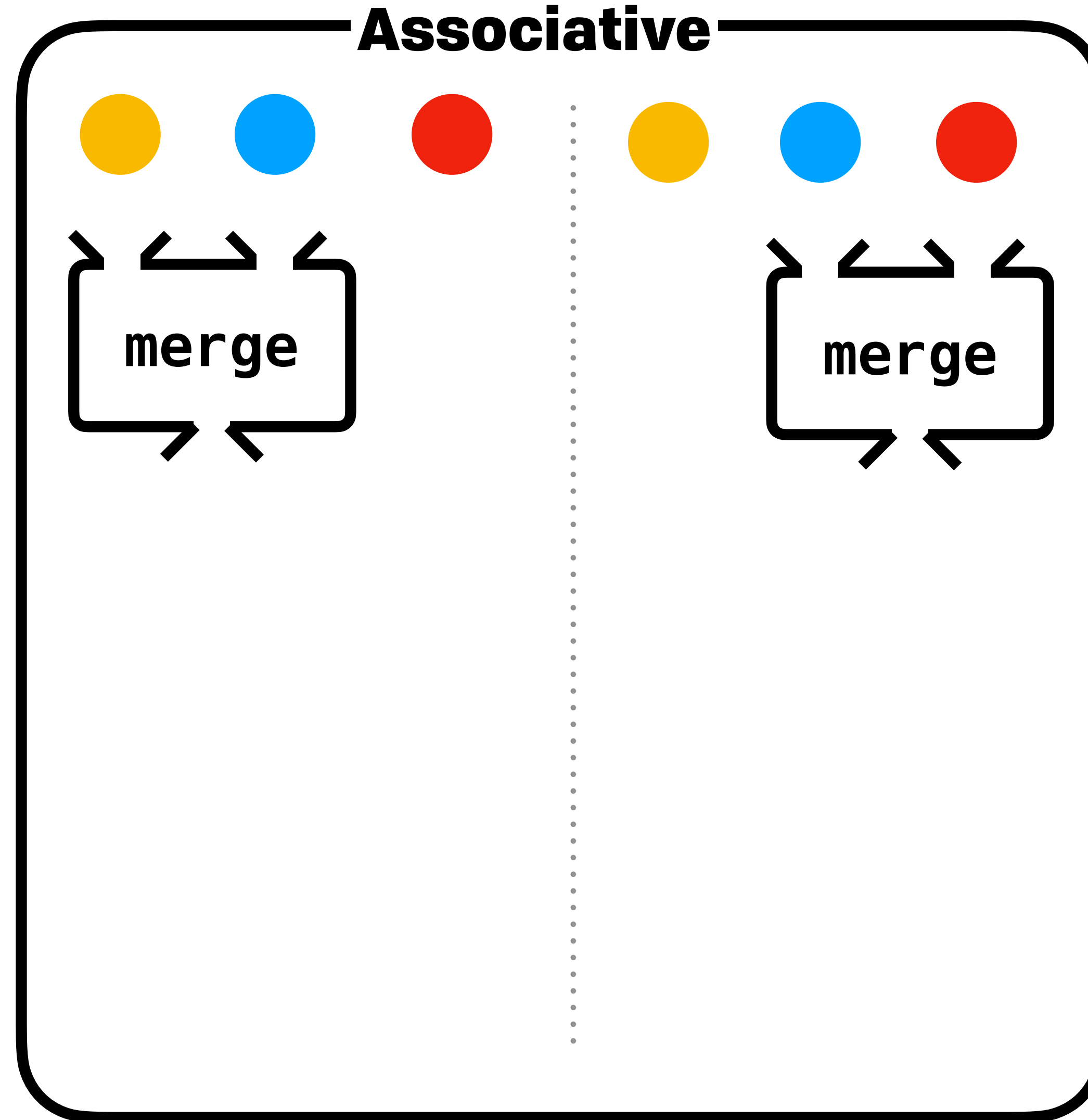
- Network agnostic
- Any number of replicas



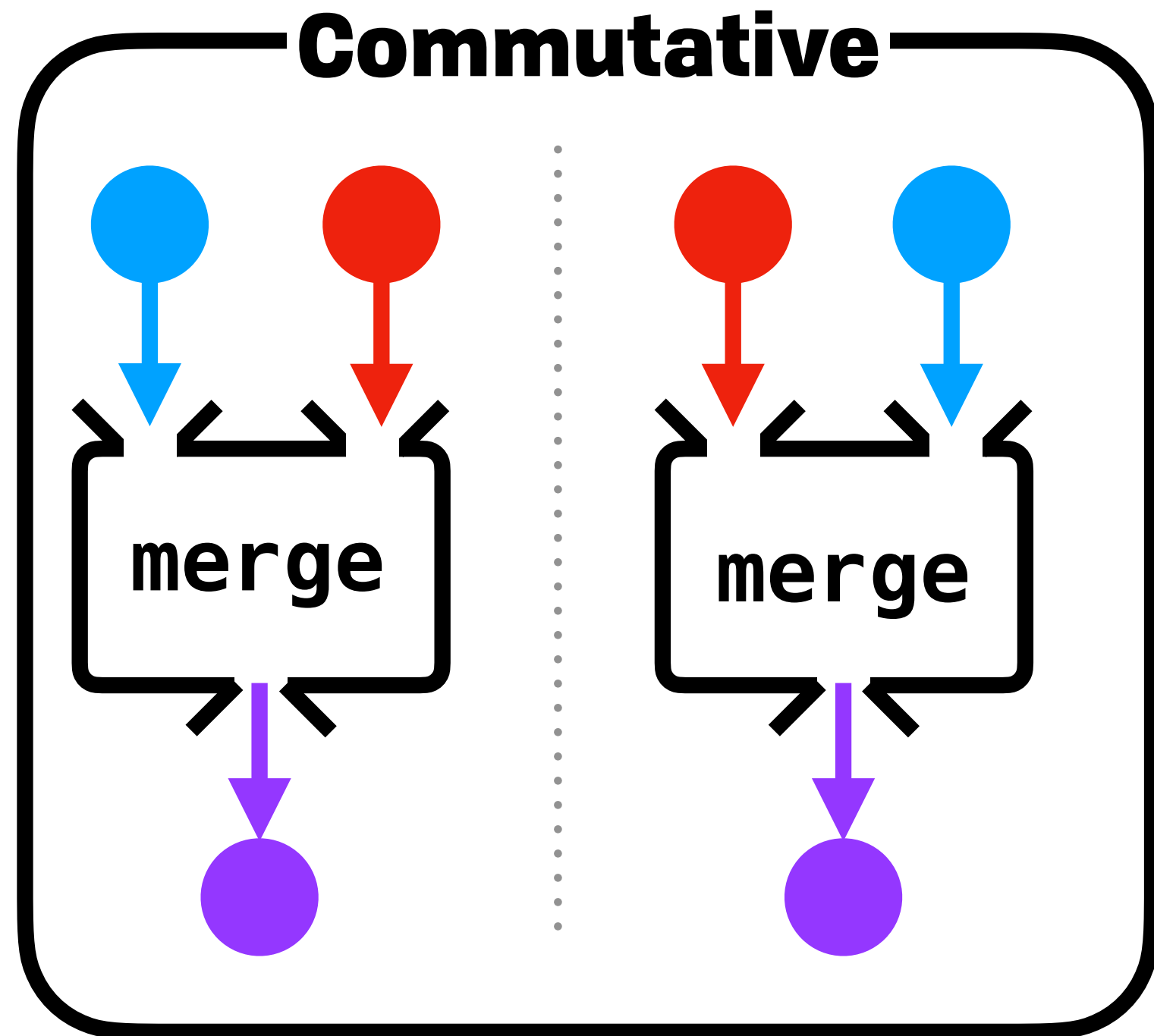
Abstract



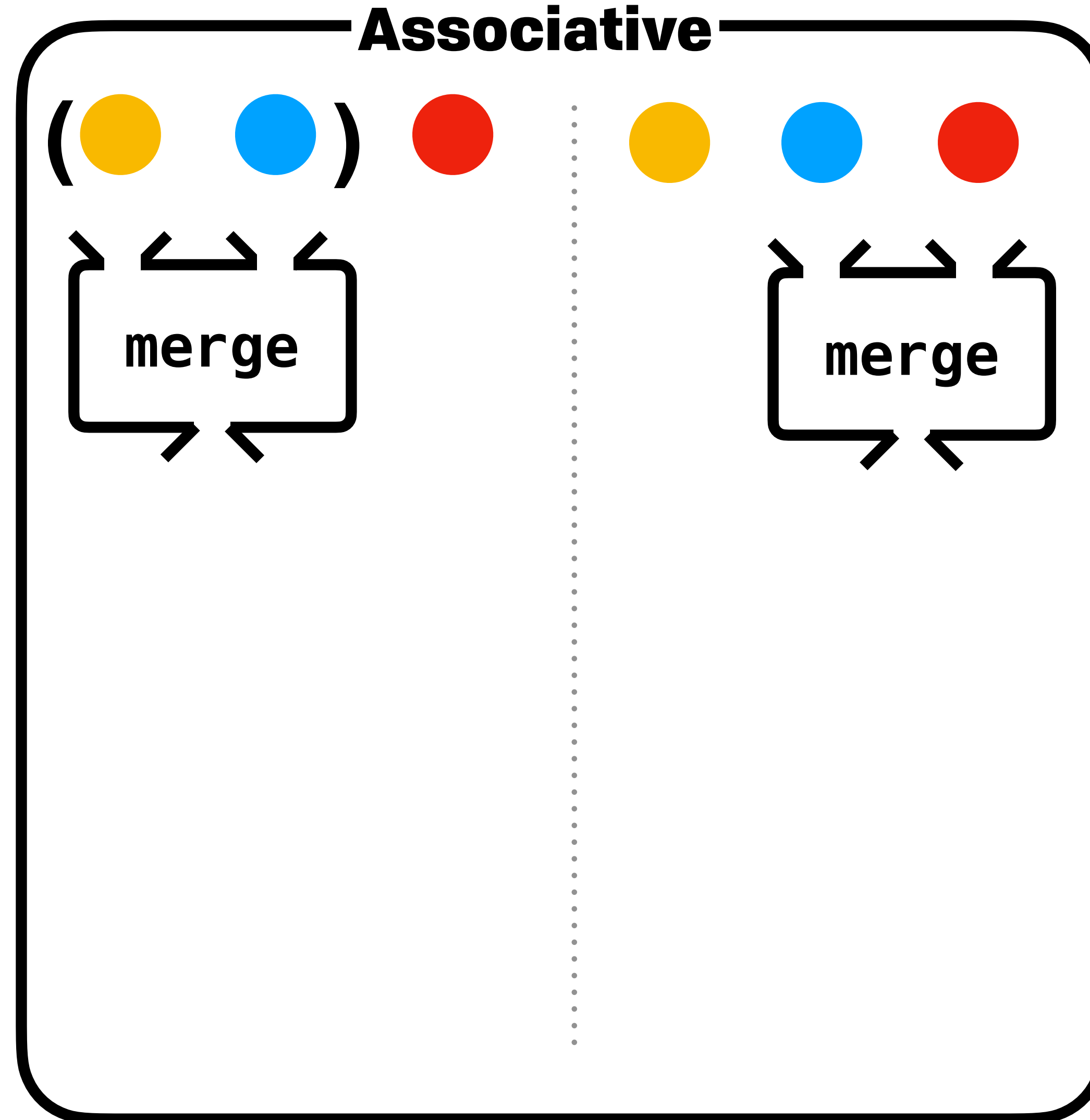
- Network agnostic
- Any number of replicas



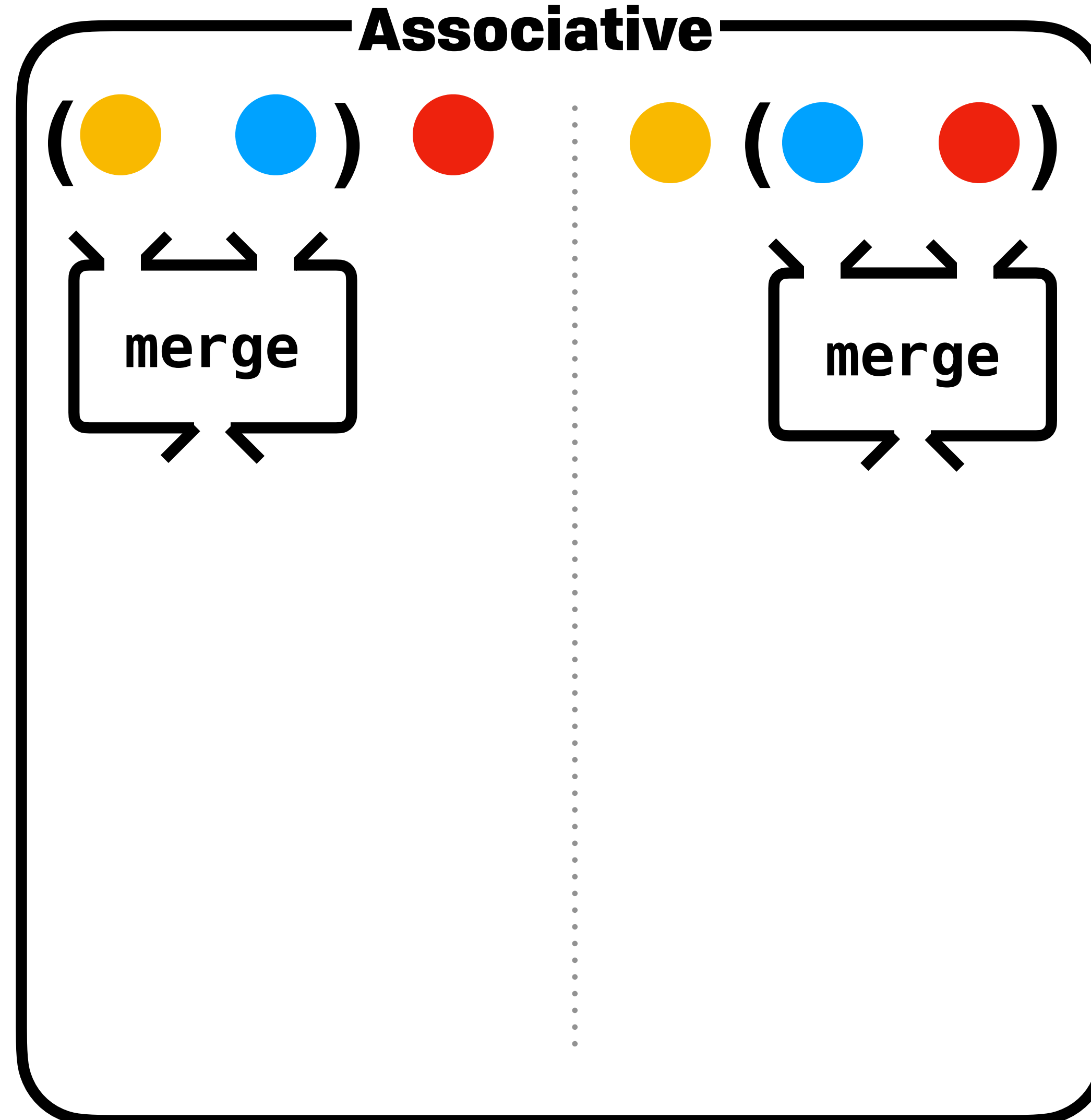
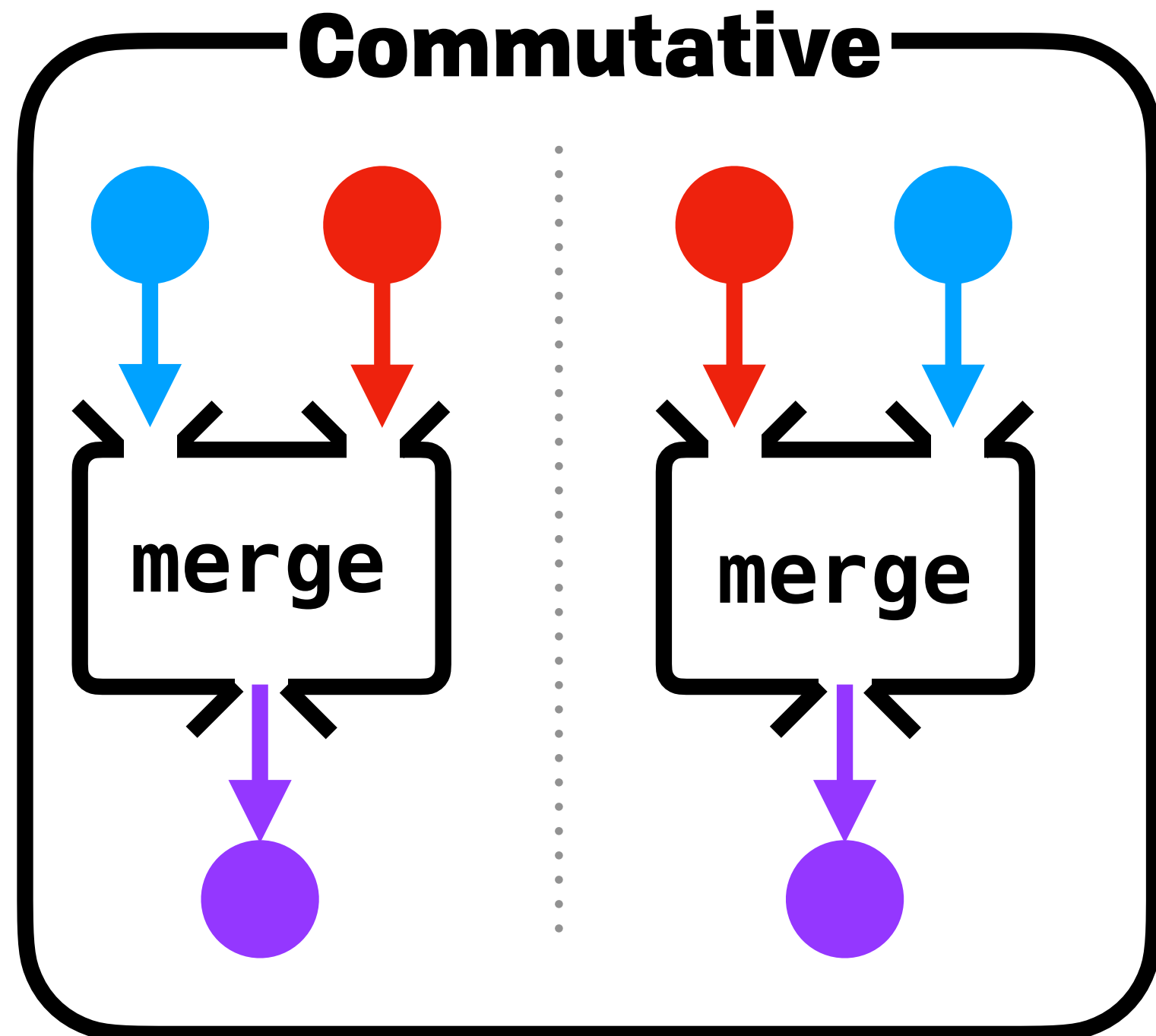
Abstract



- Network agnostic
- Any number of replicas

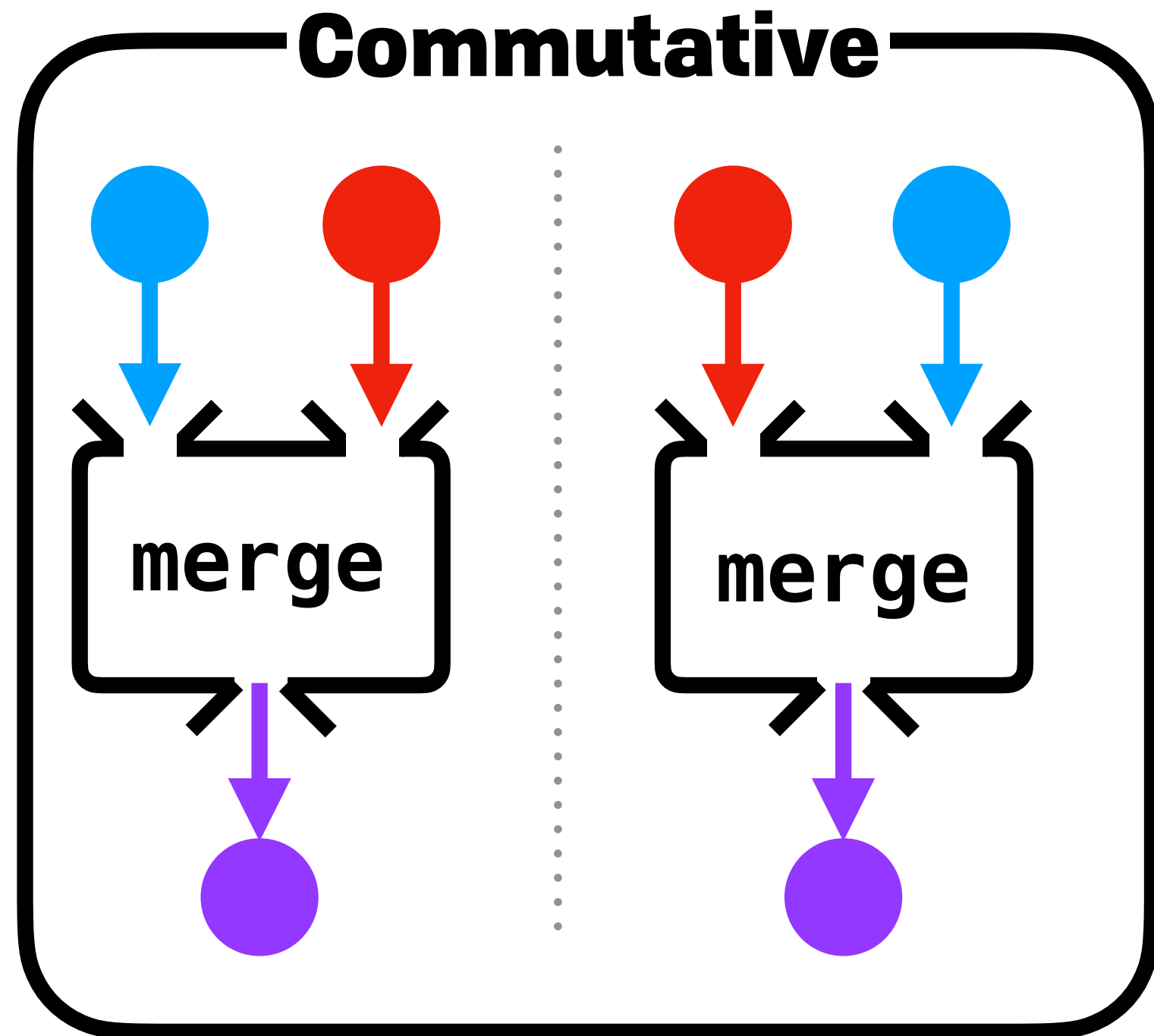


Abstract

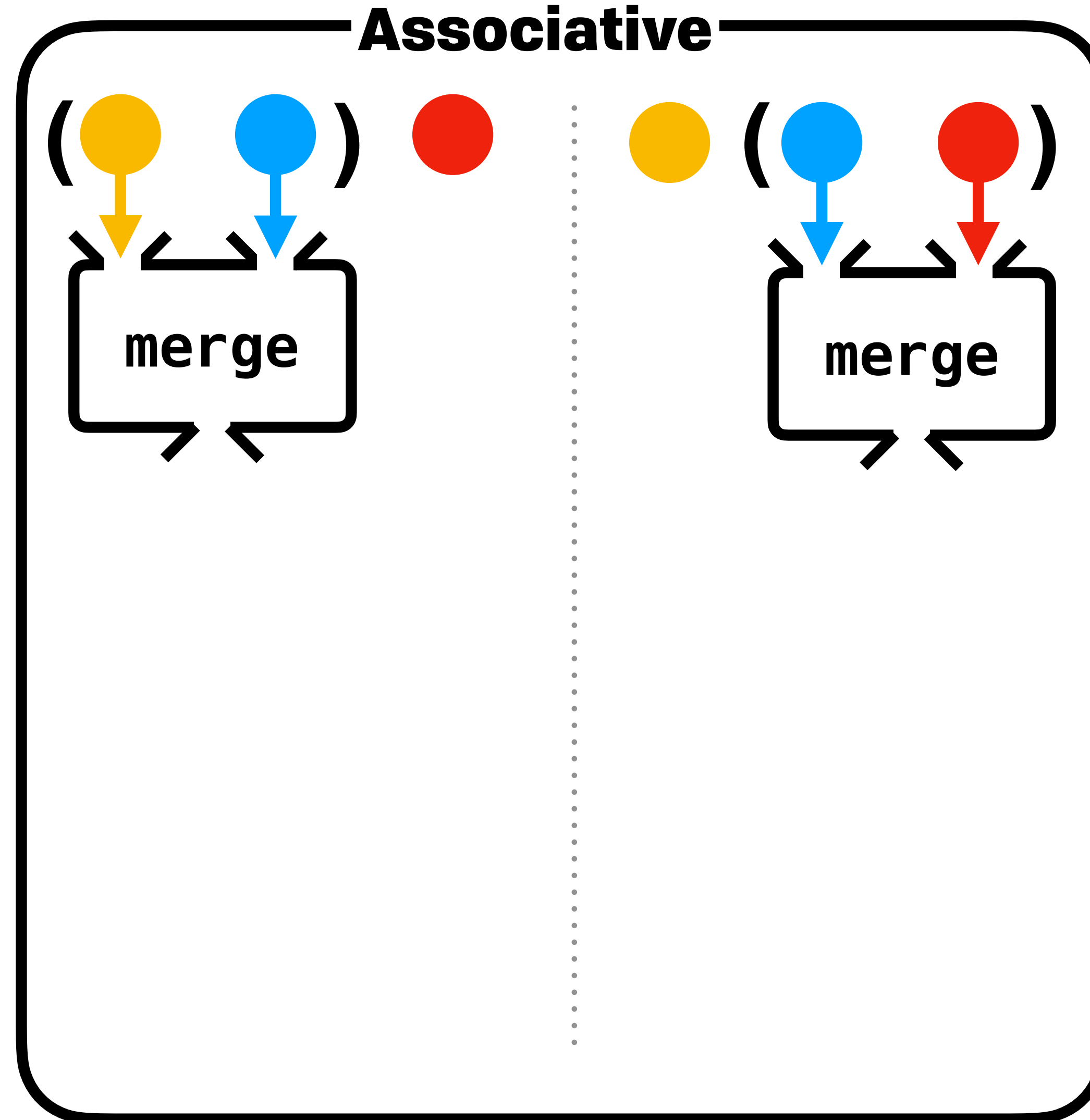


- Network agnostic
- Any number of replicas

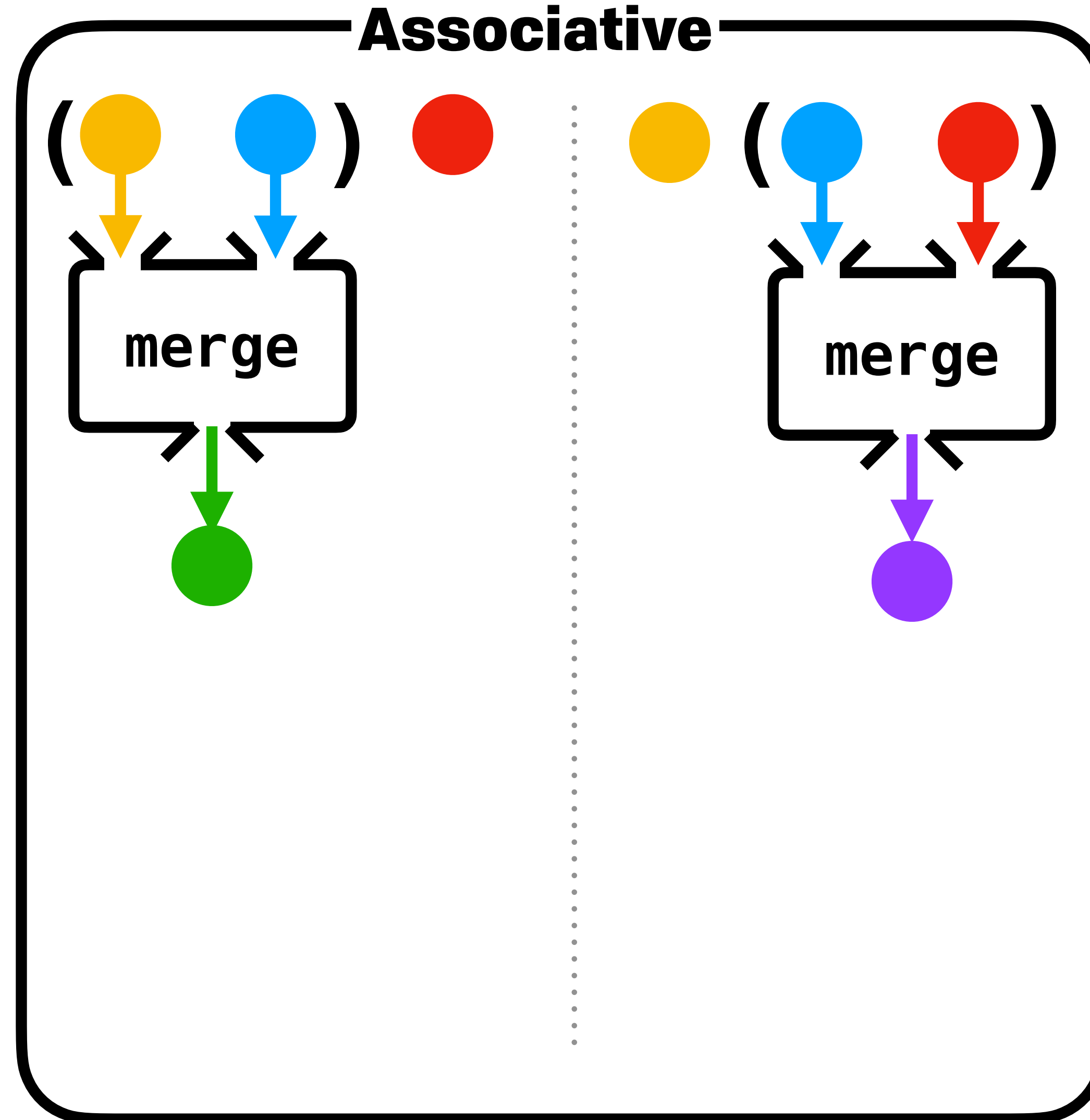
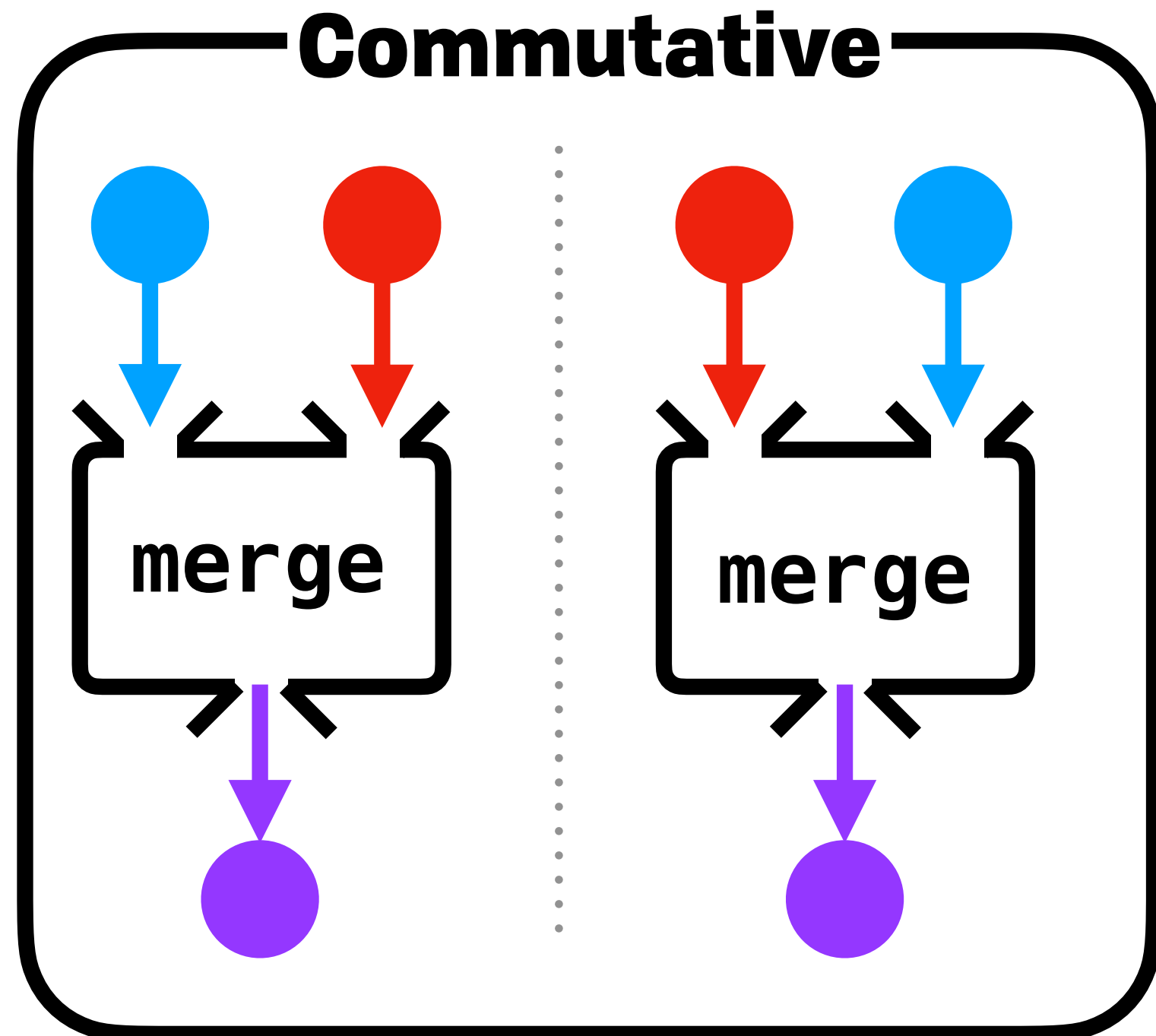
Abstract



- Network agnostic
- Any number of replicas

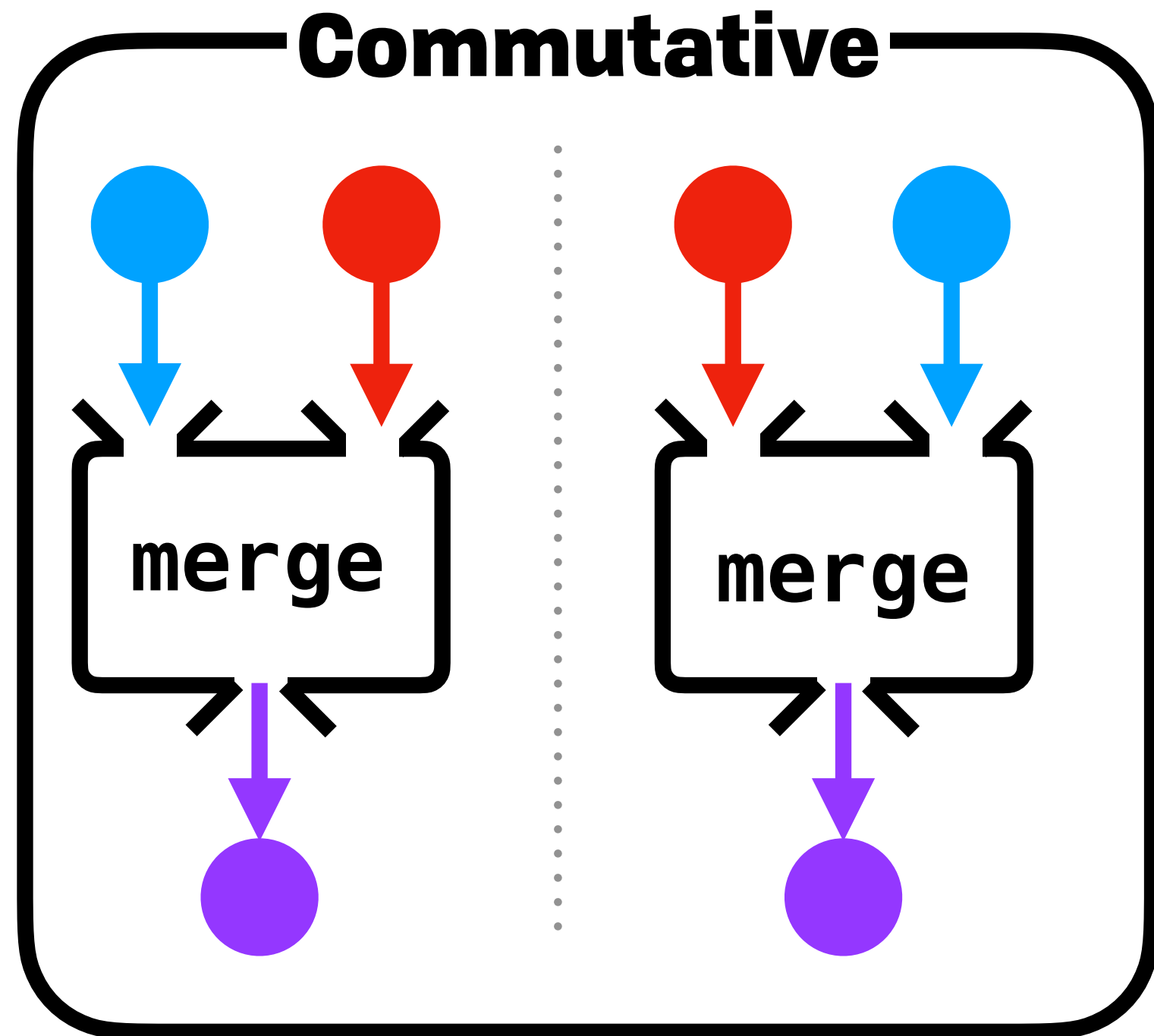


Abstract

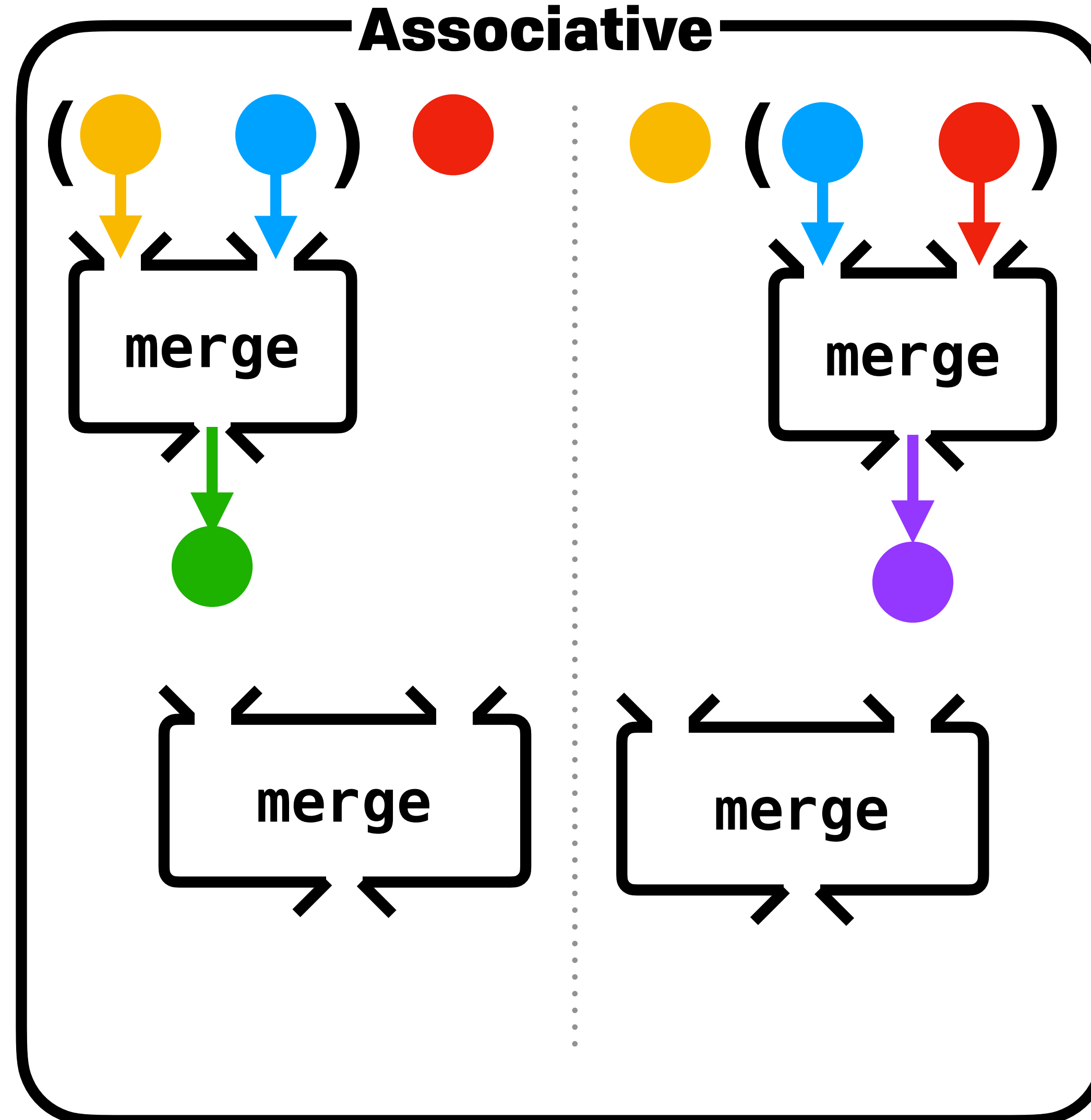


- Network agnostic
- Any number of replicas

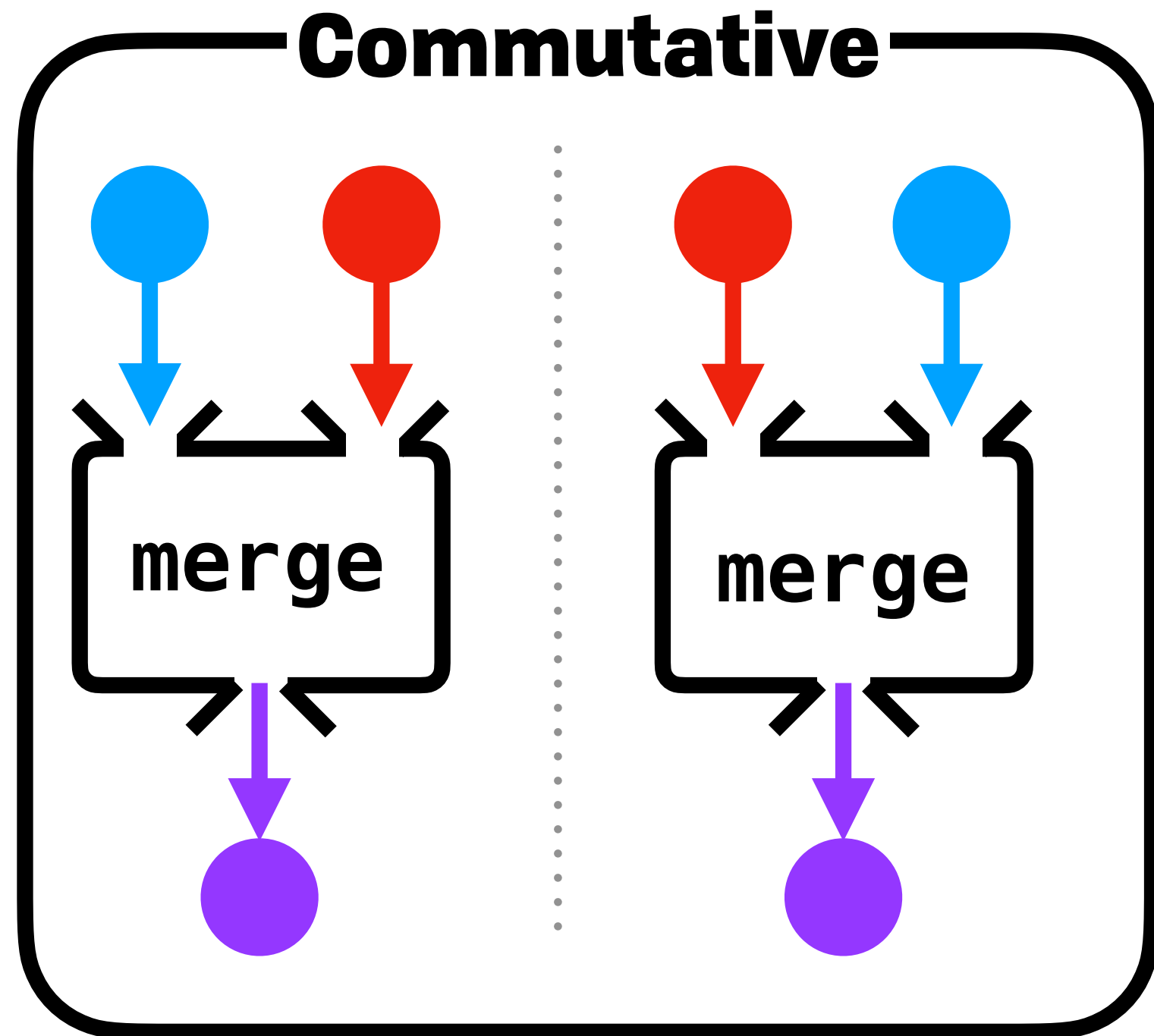
Abstract



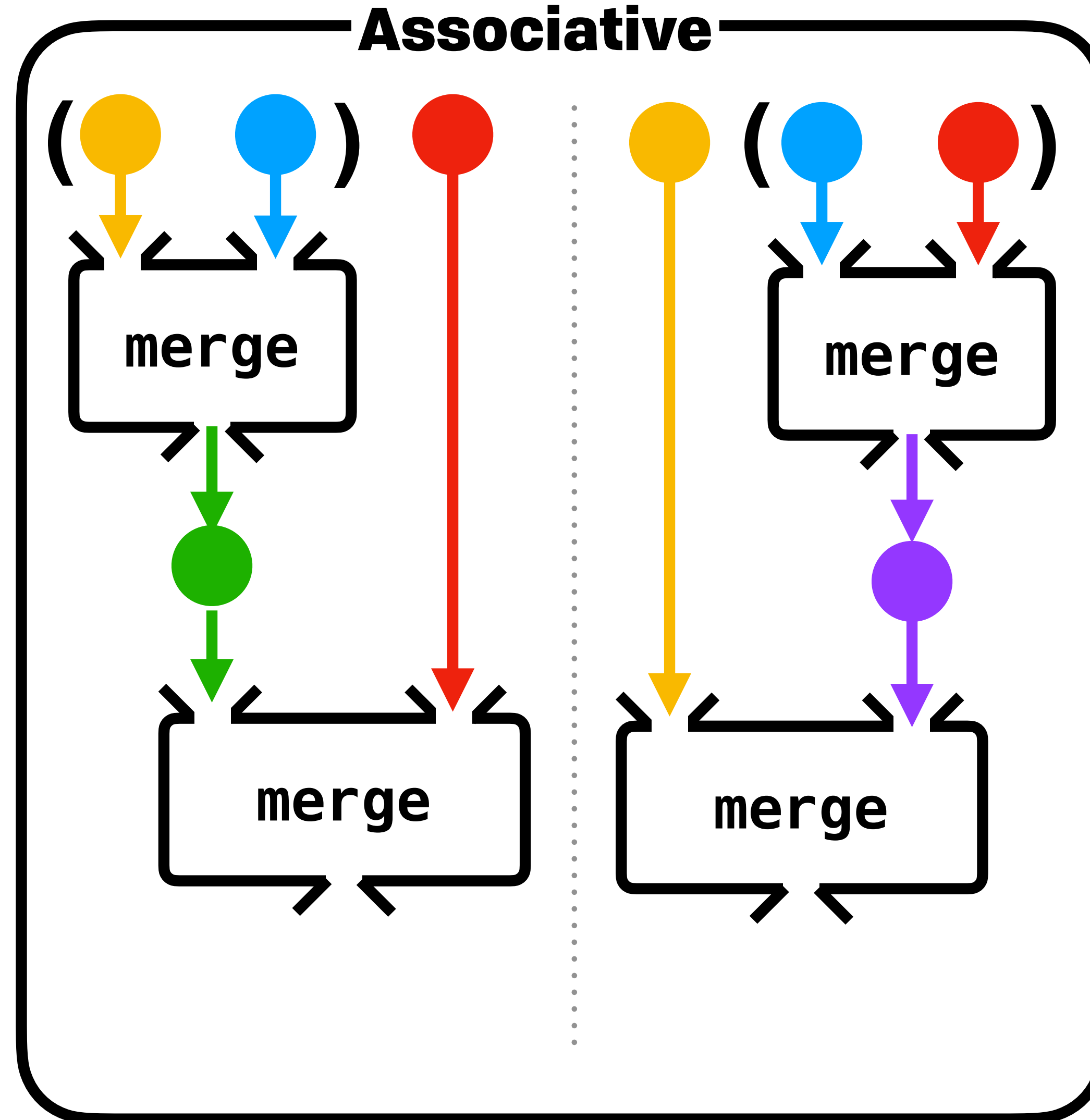
- Network agnostic
- Any number of replicas



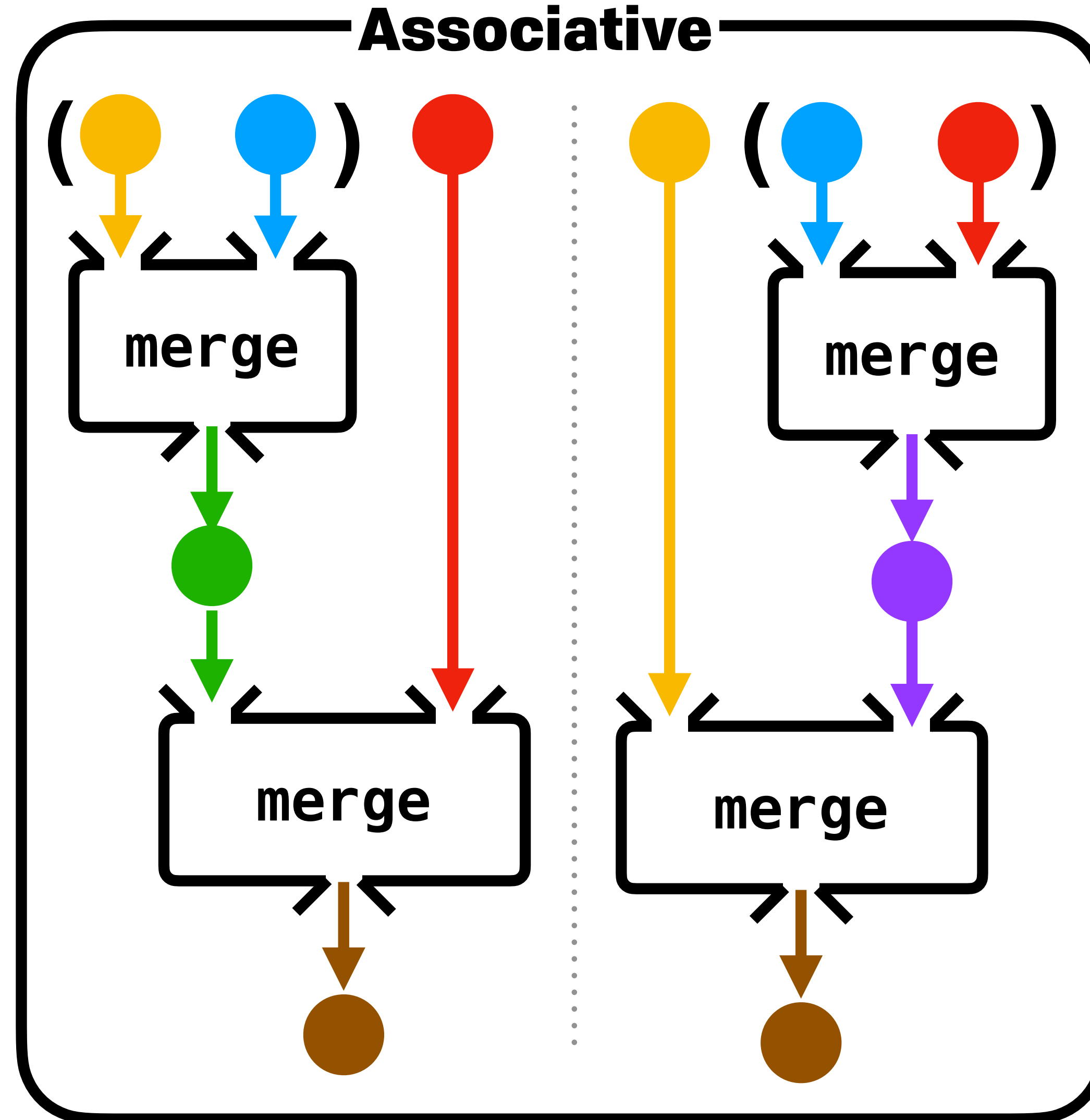
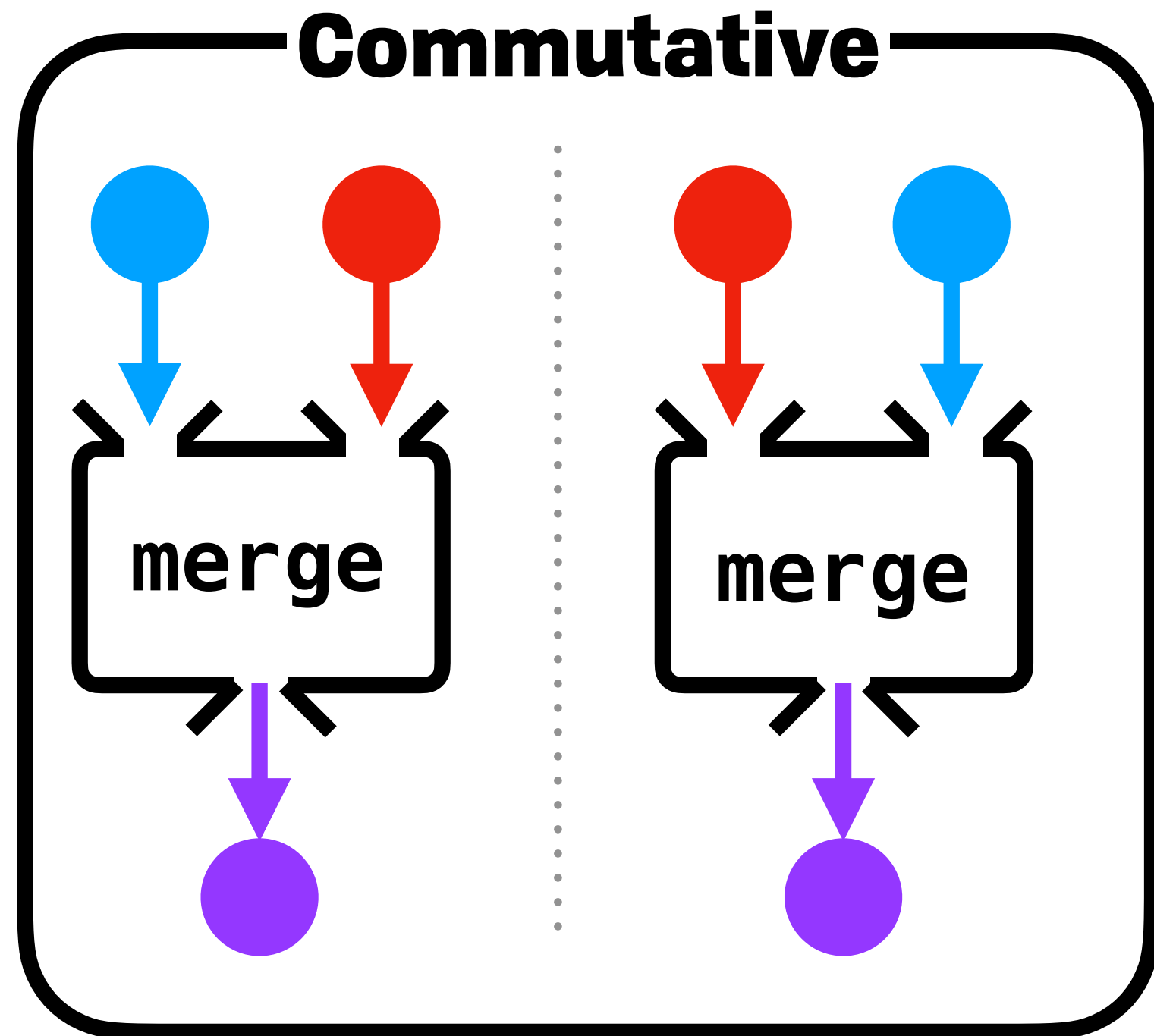
Abstract



- Network agnostic
- Any number of replicas

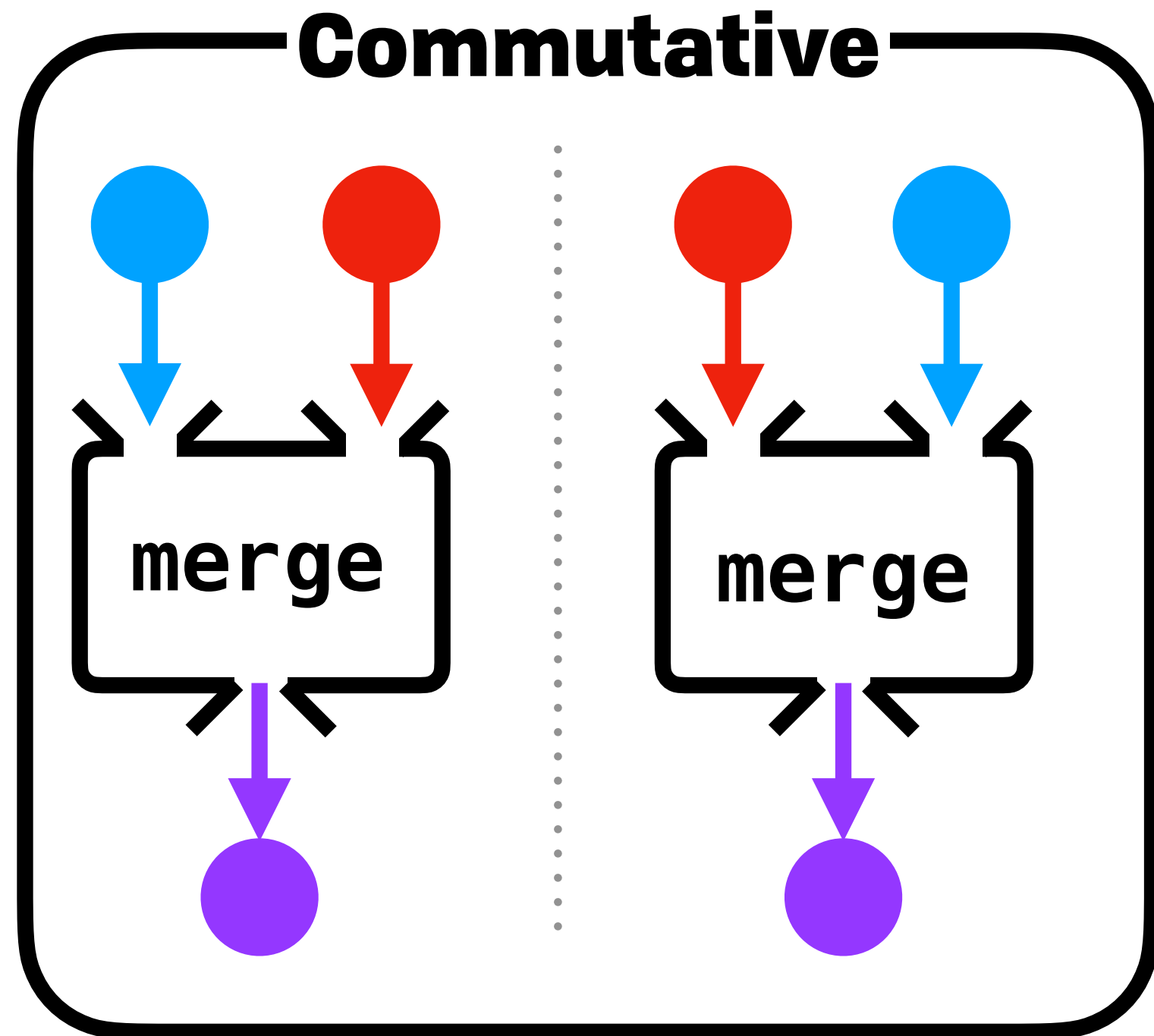


Abstract

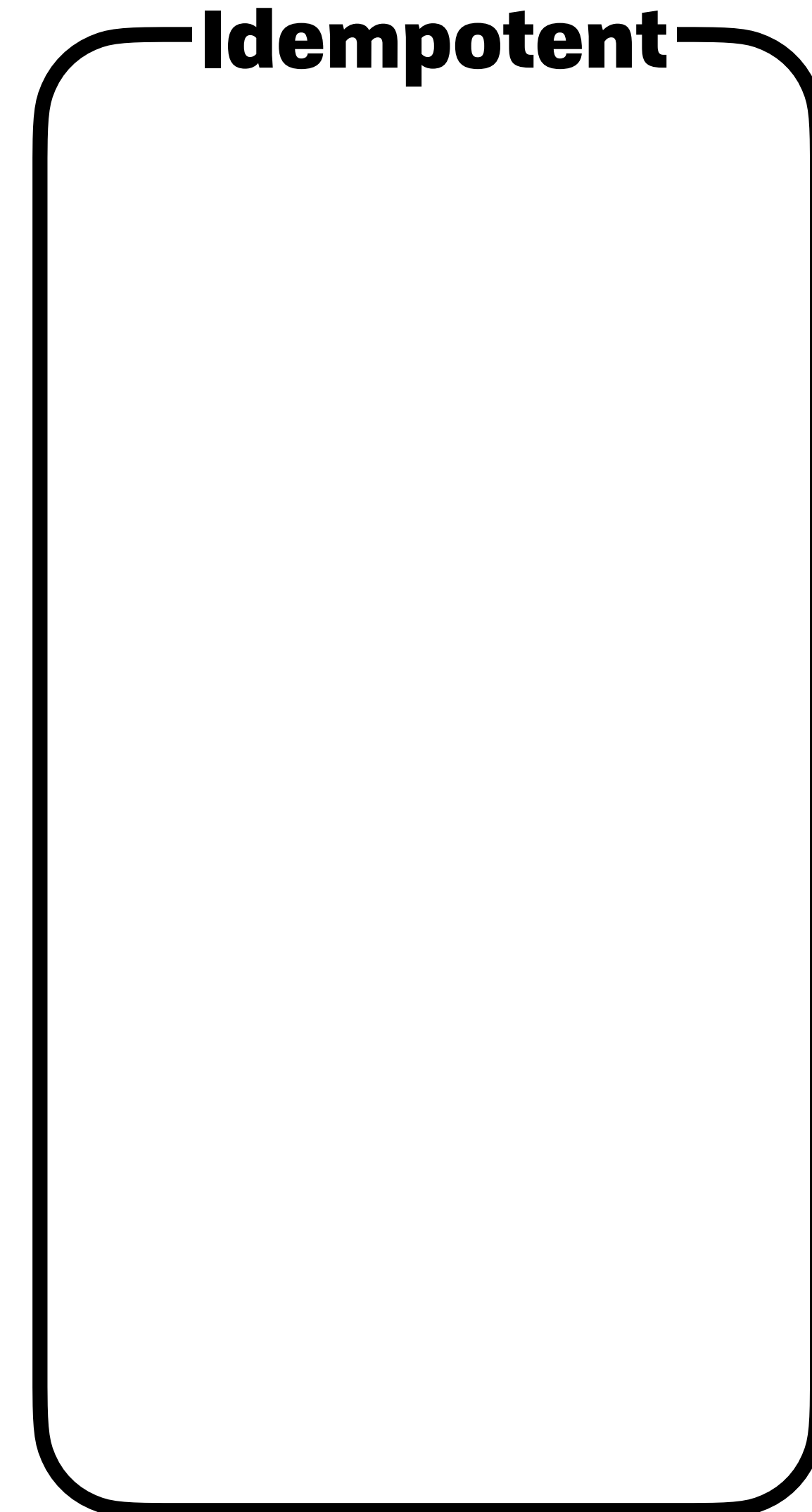
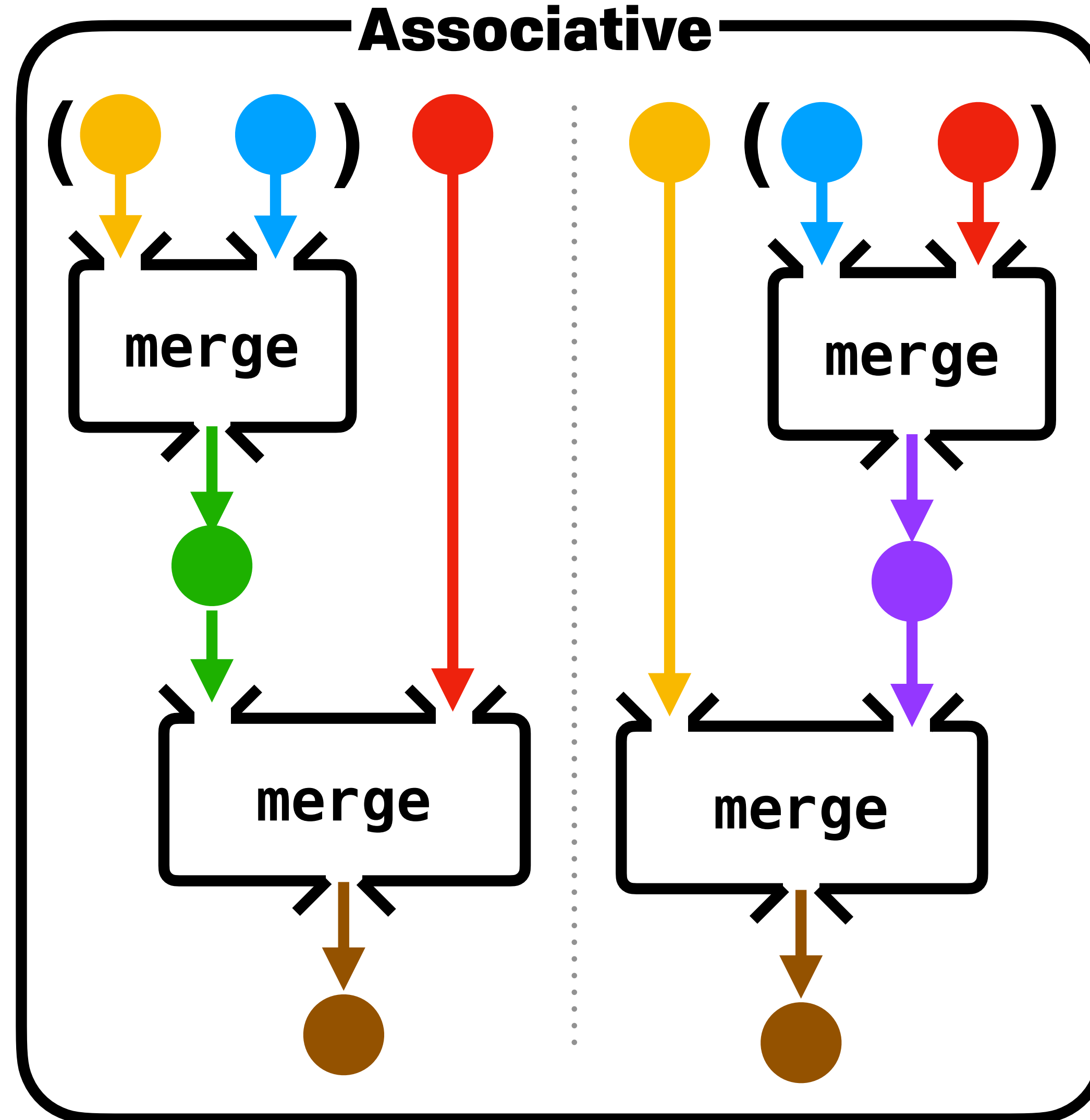


- Network agnostic
- Any number of replicas

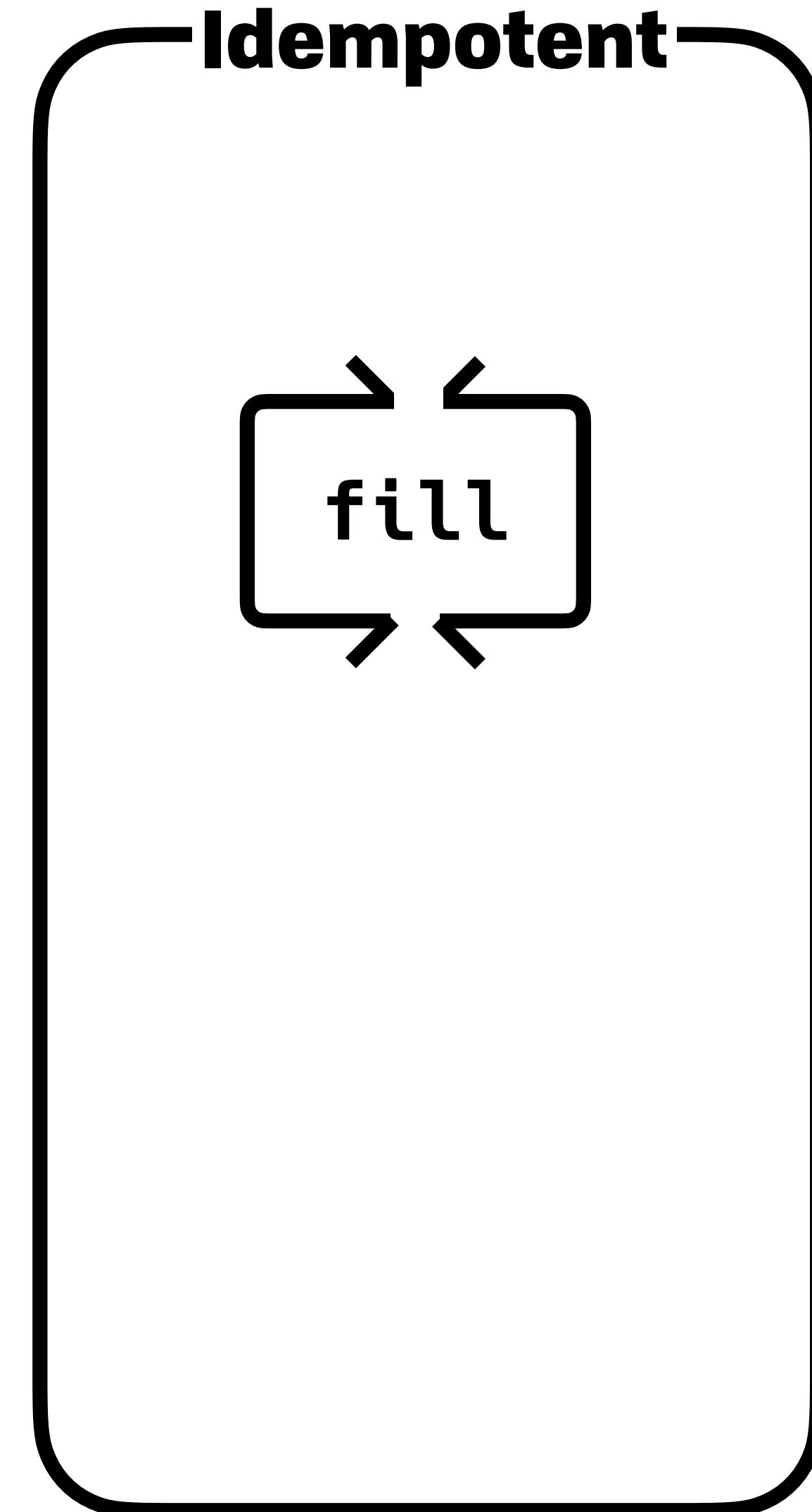
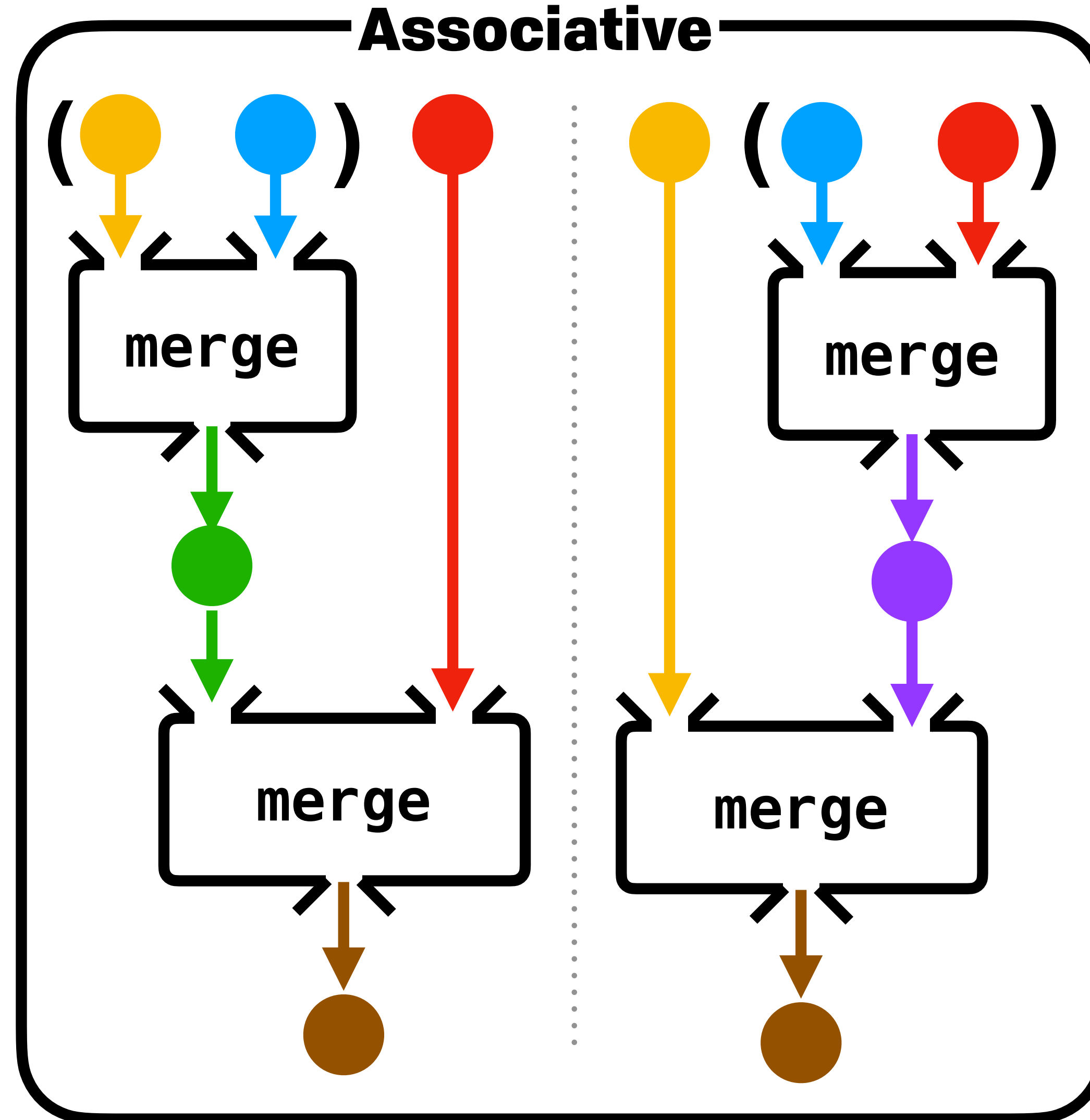
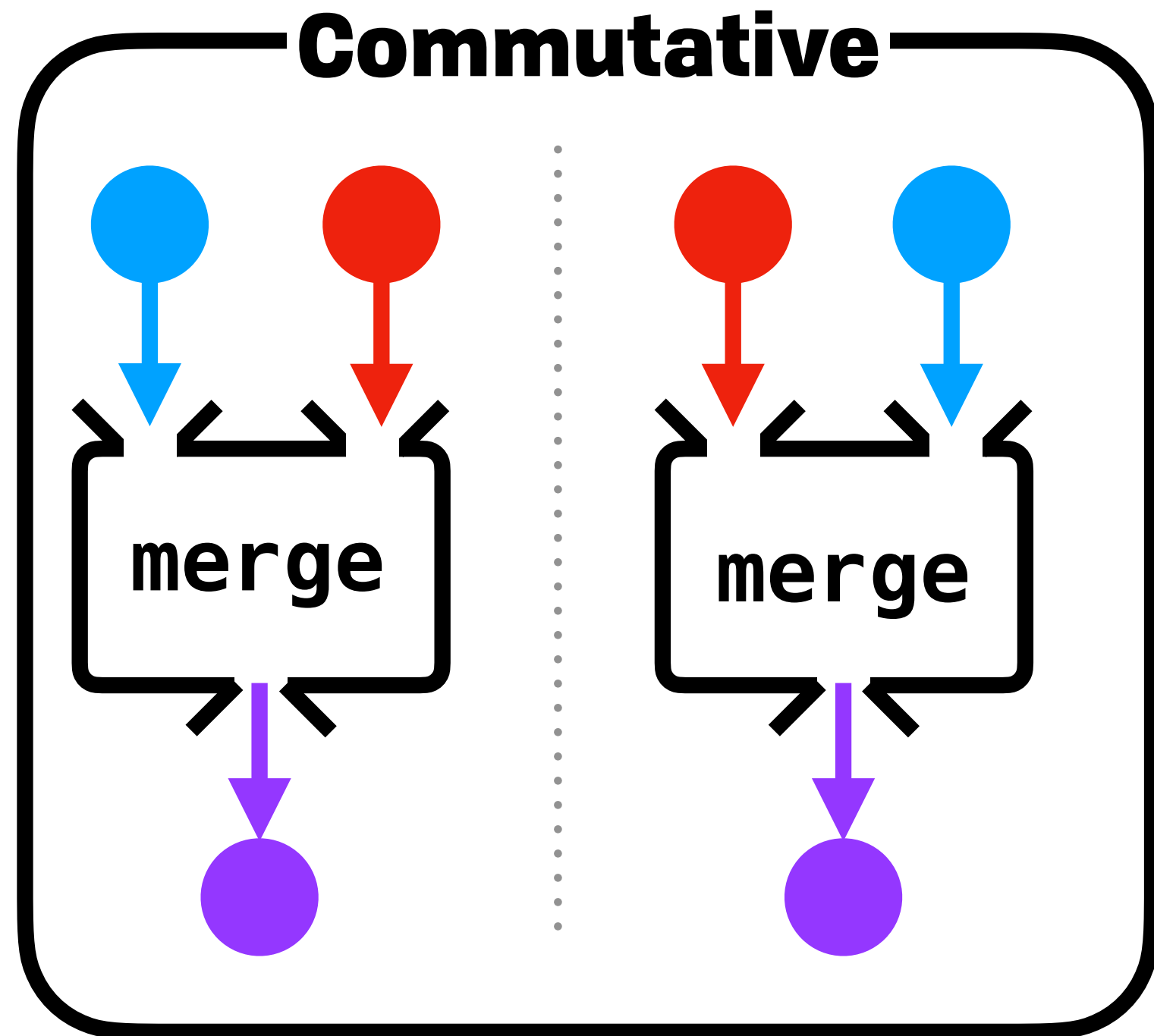
Abstract



- Network agnostic
- Any number of replicas

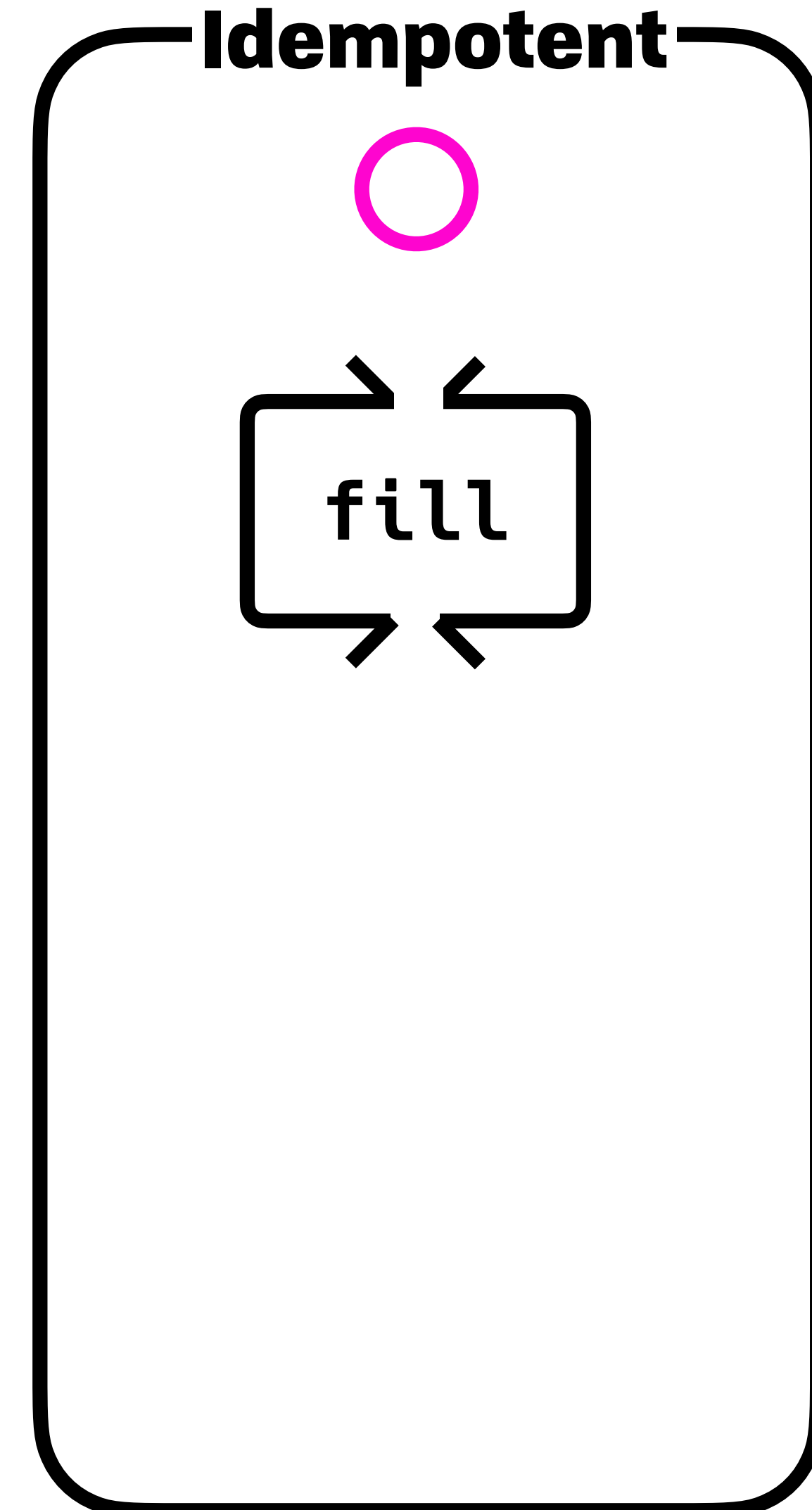
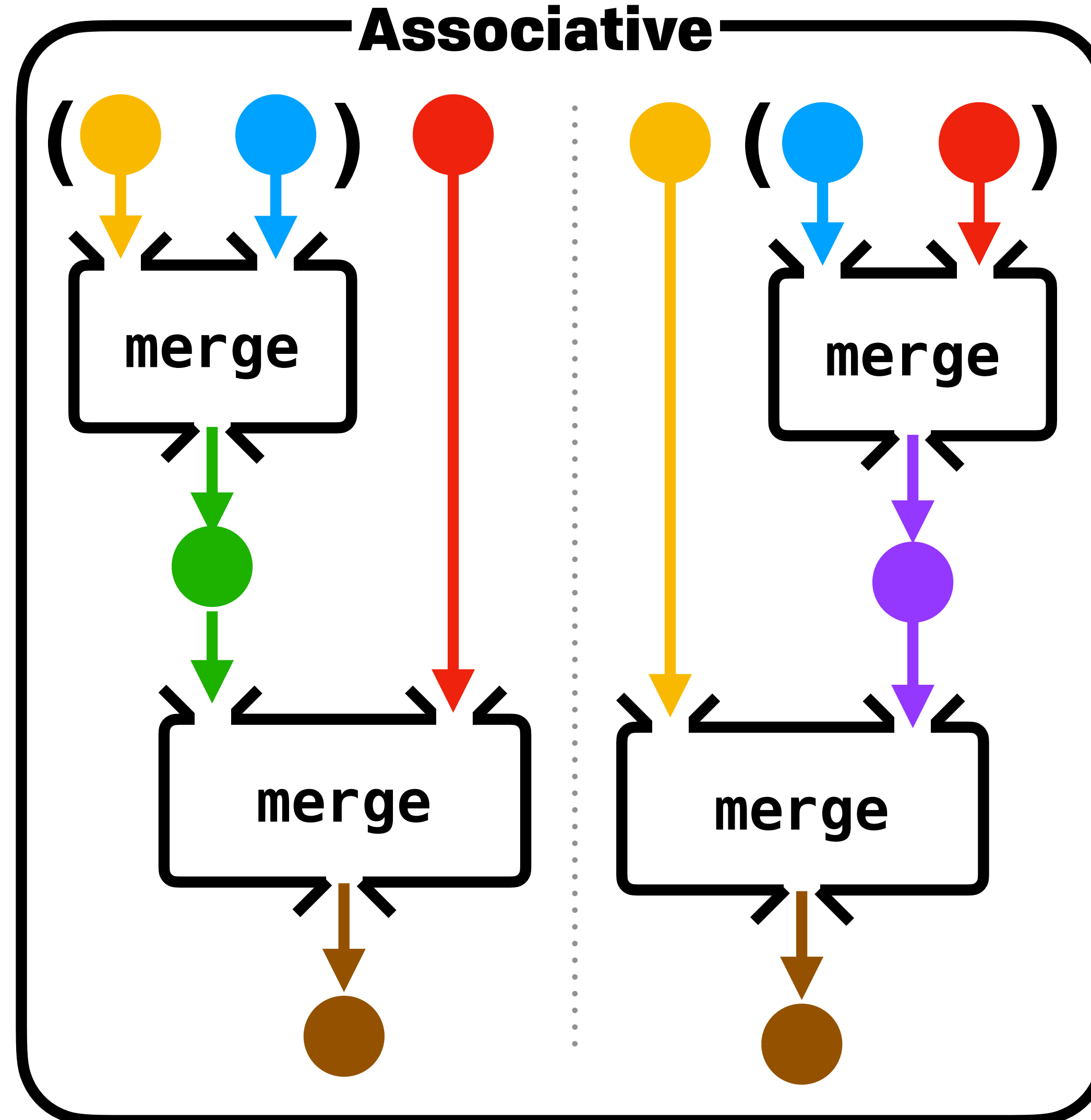
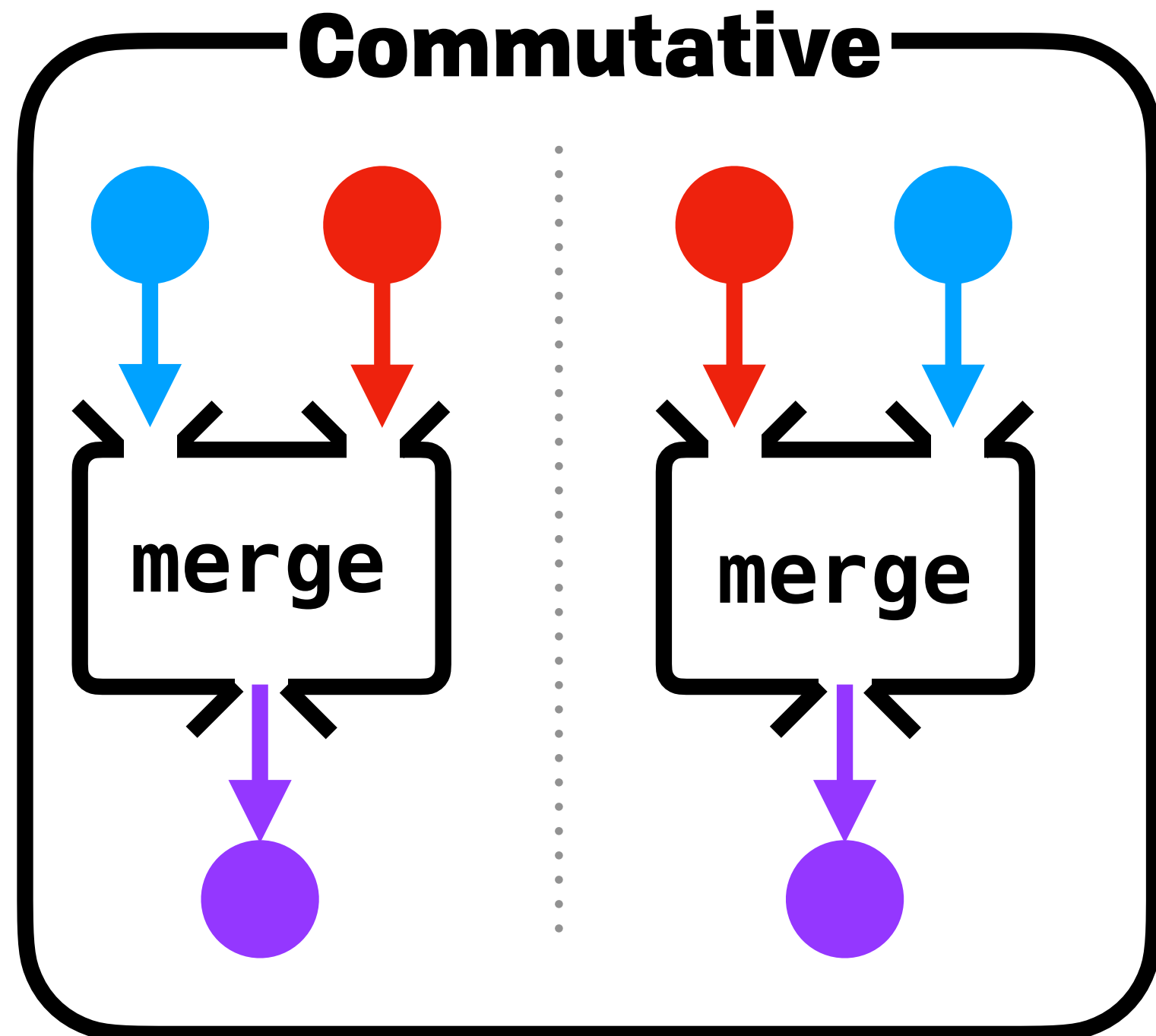


Abstract



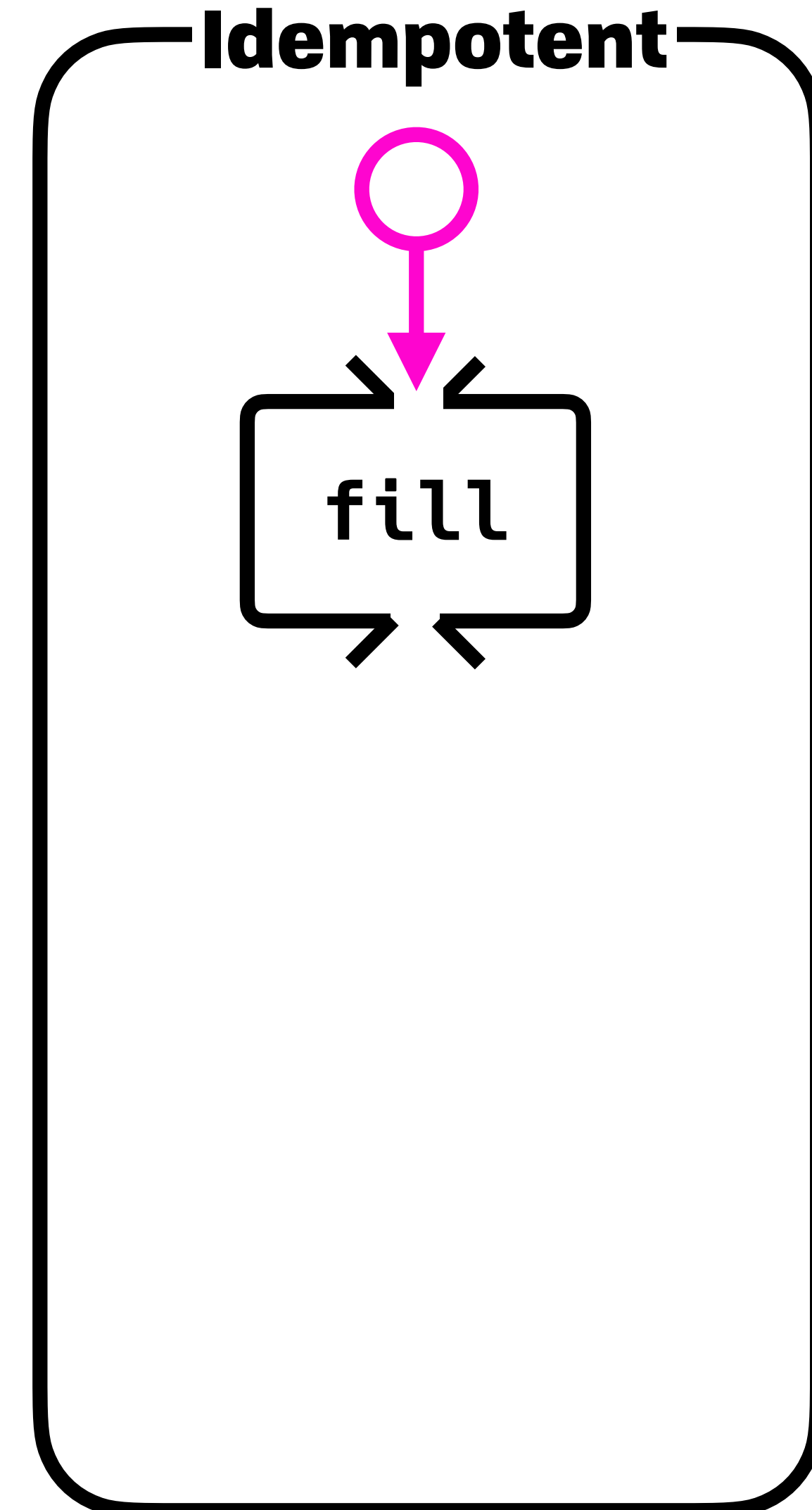
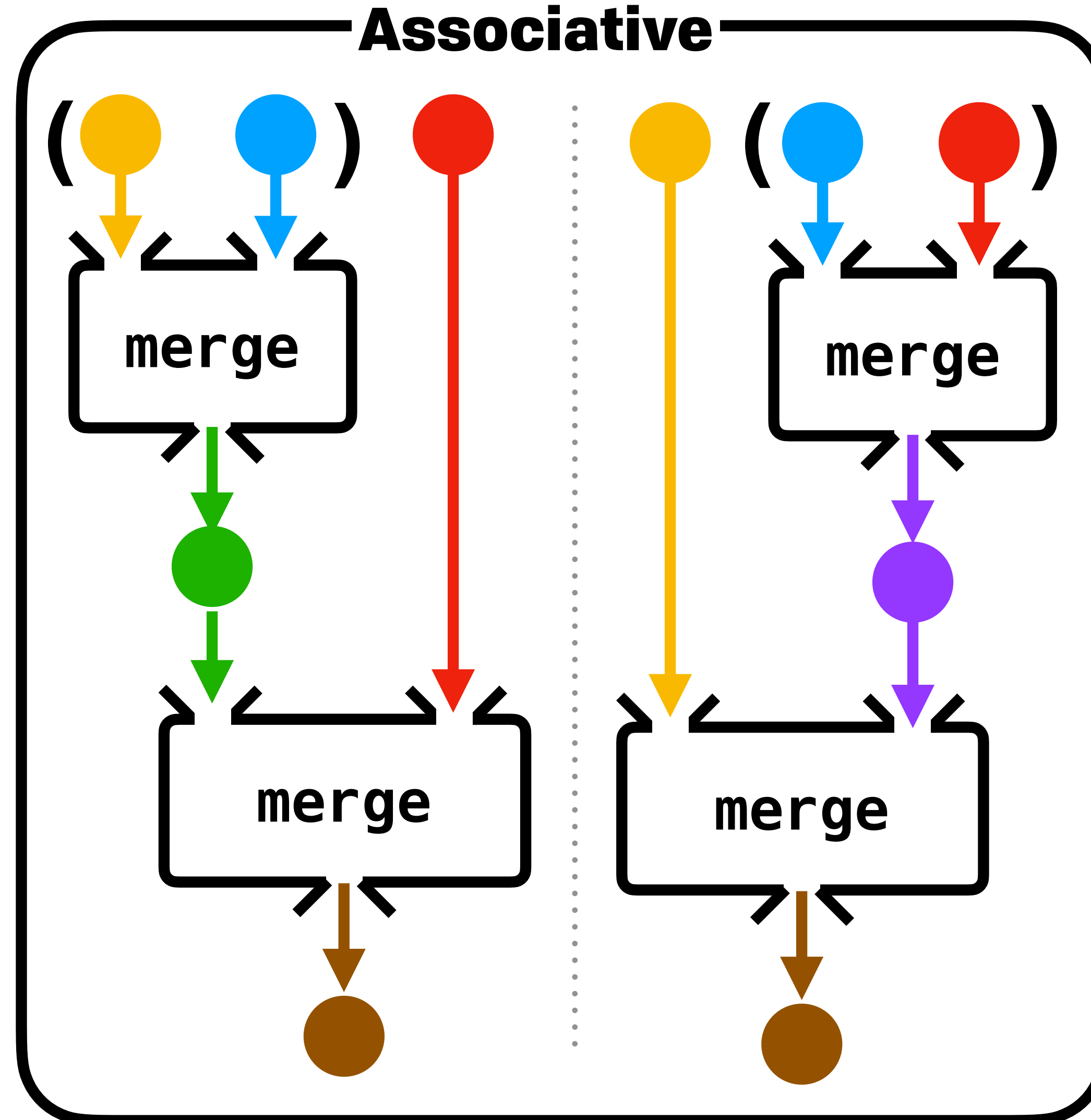
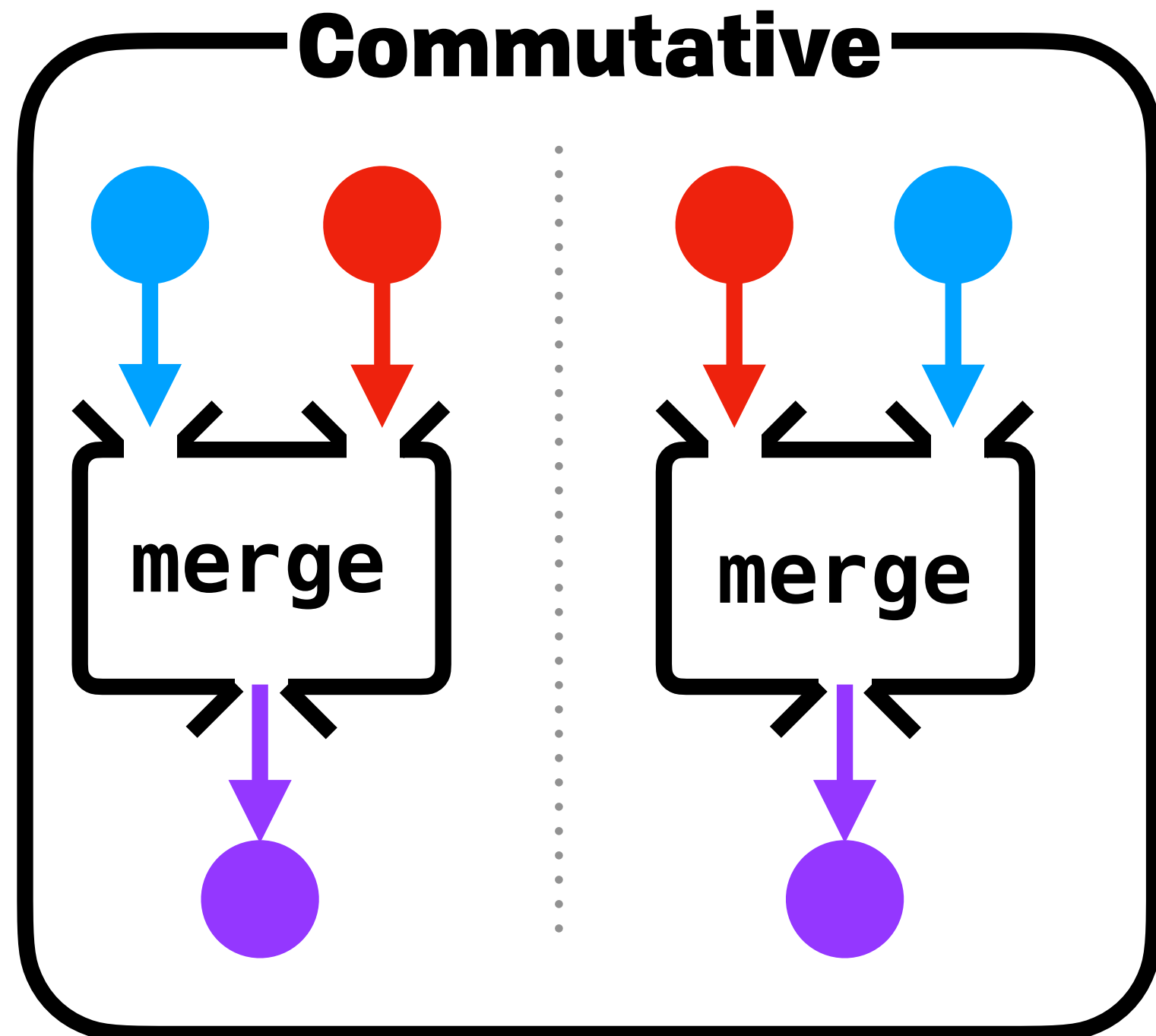
- Network agnostic
- Any number of replicas

Abstract



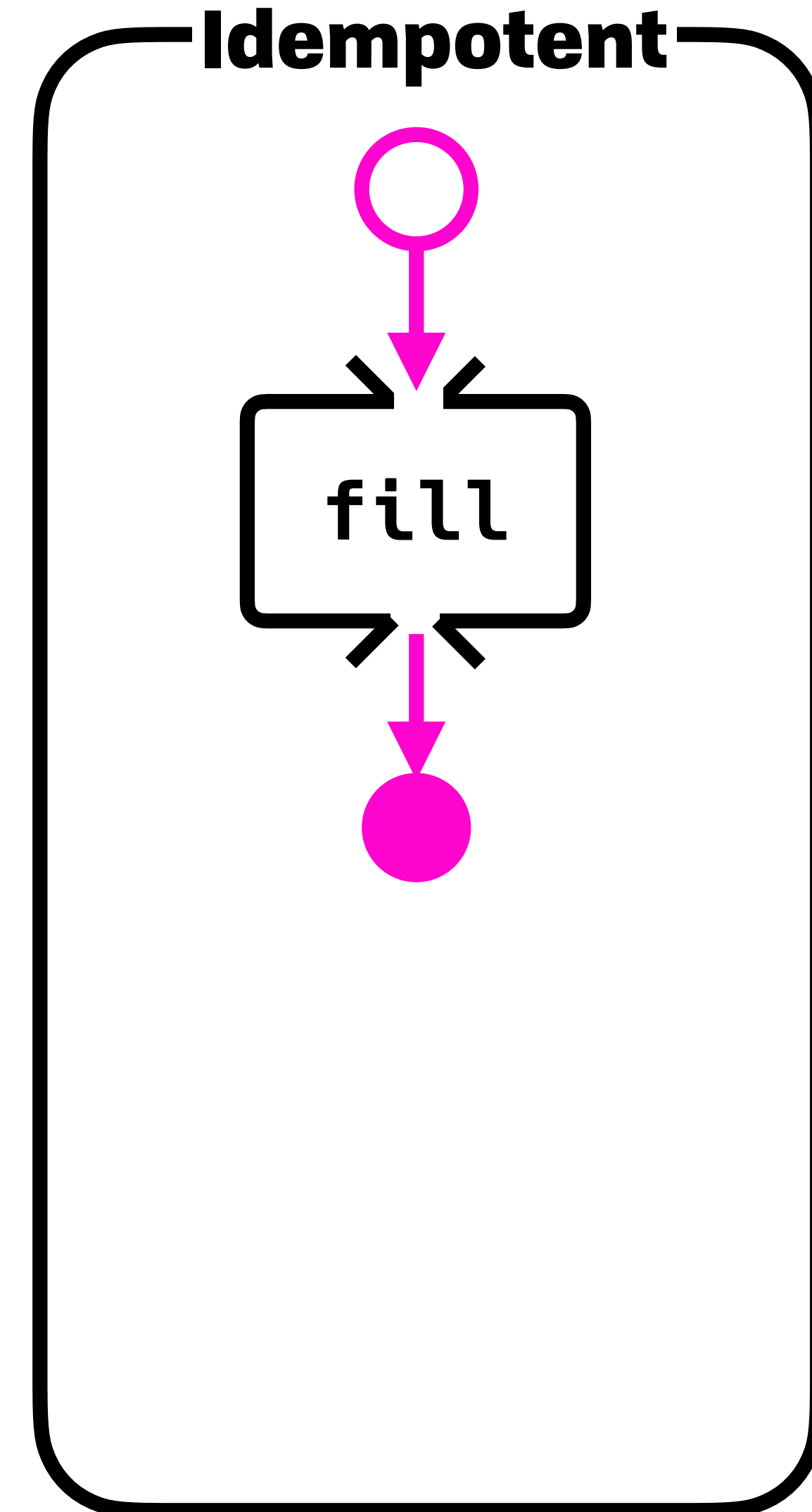
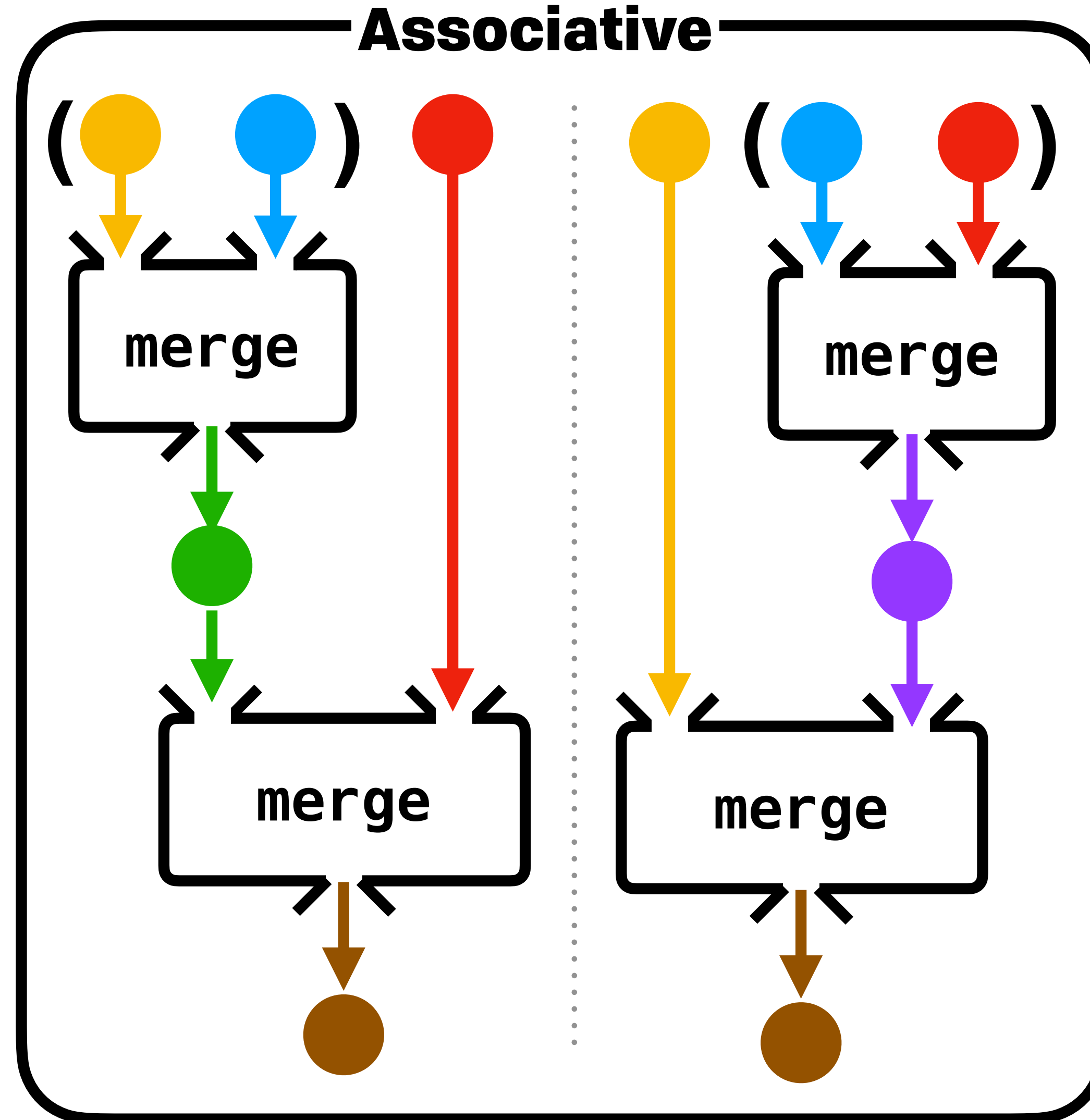
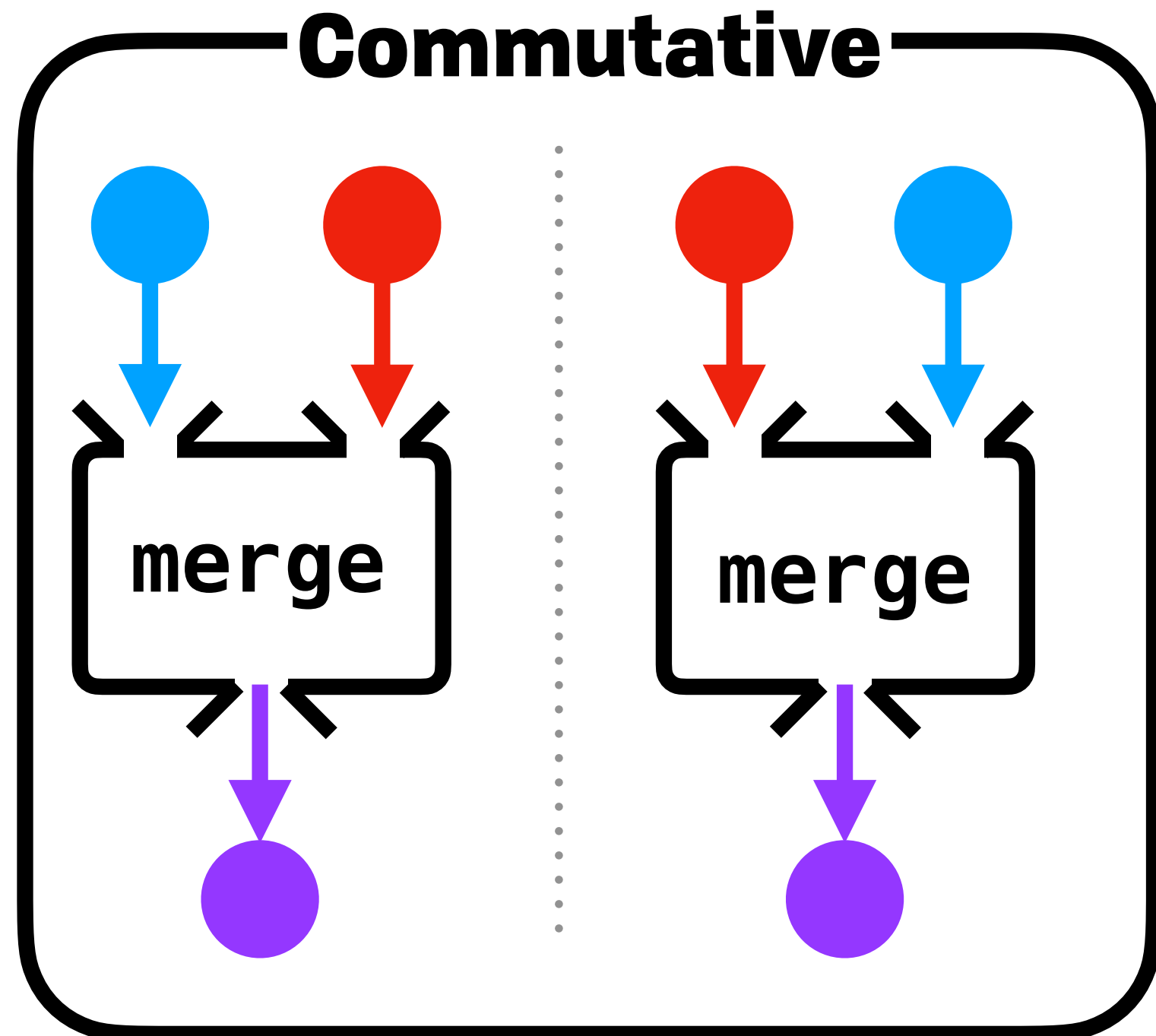
- Network agnostic
- Any number of replicas

Abstract



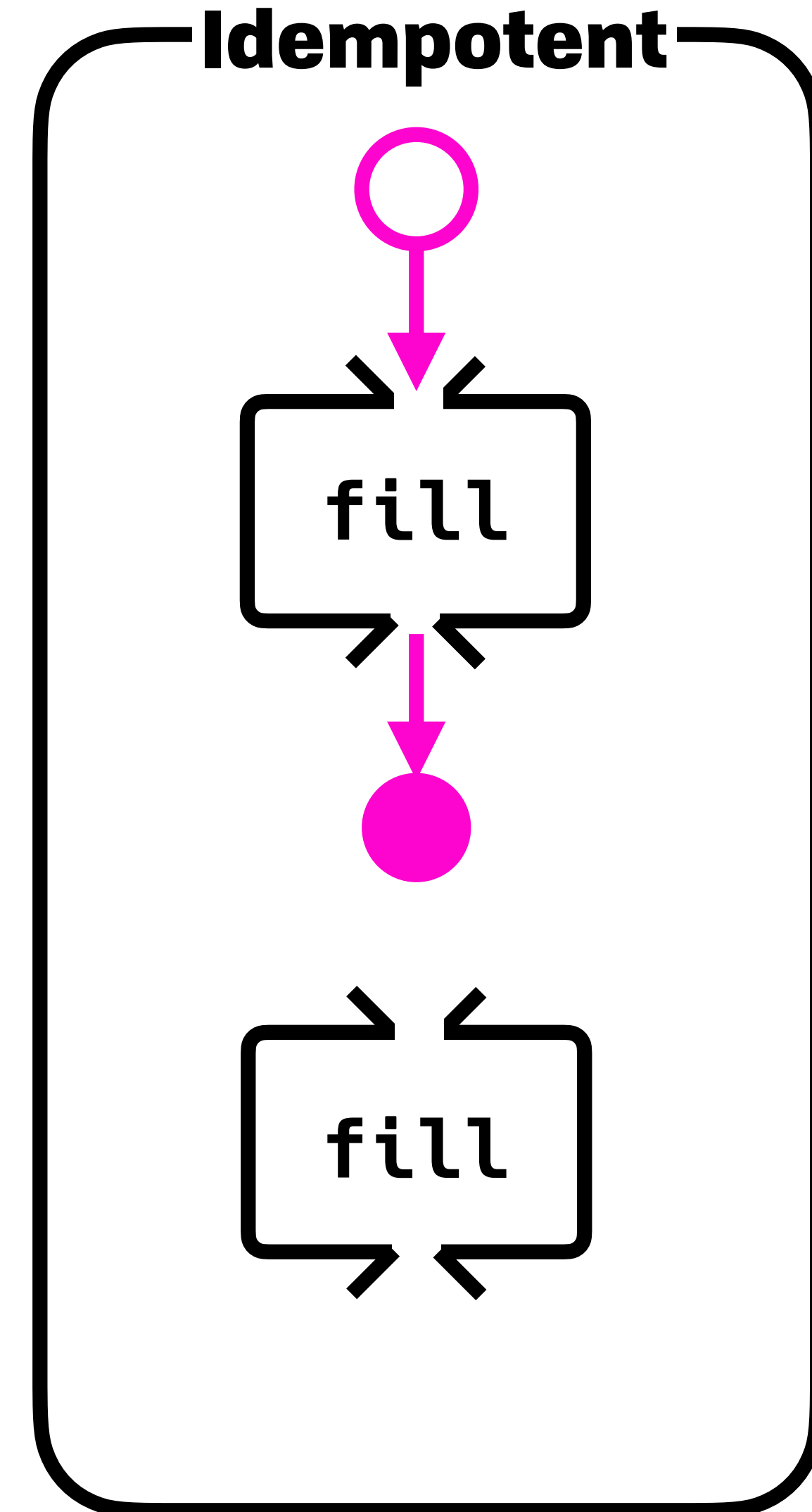
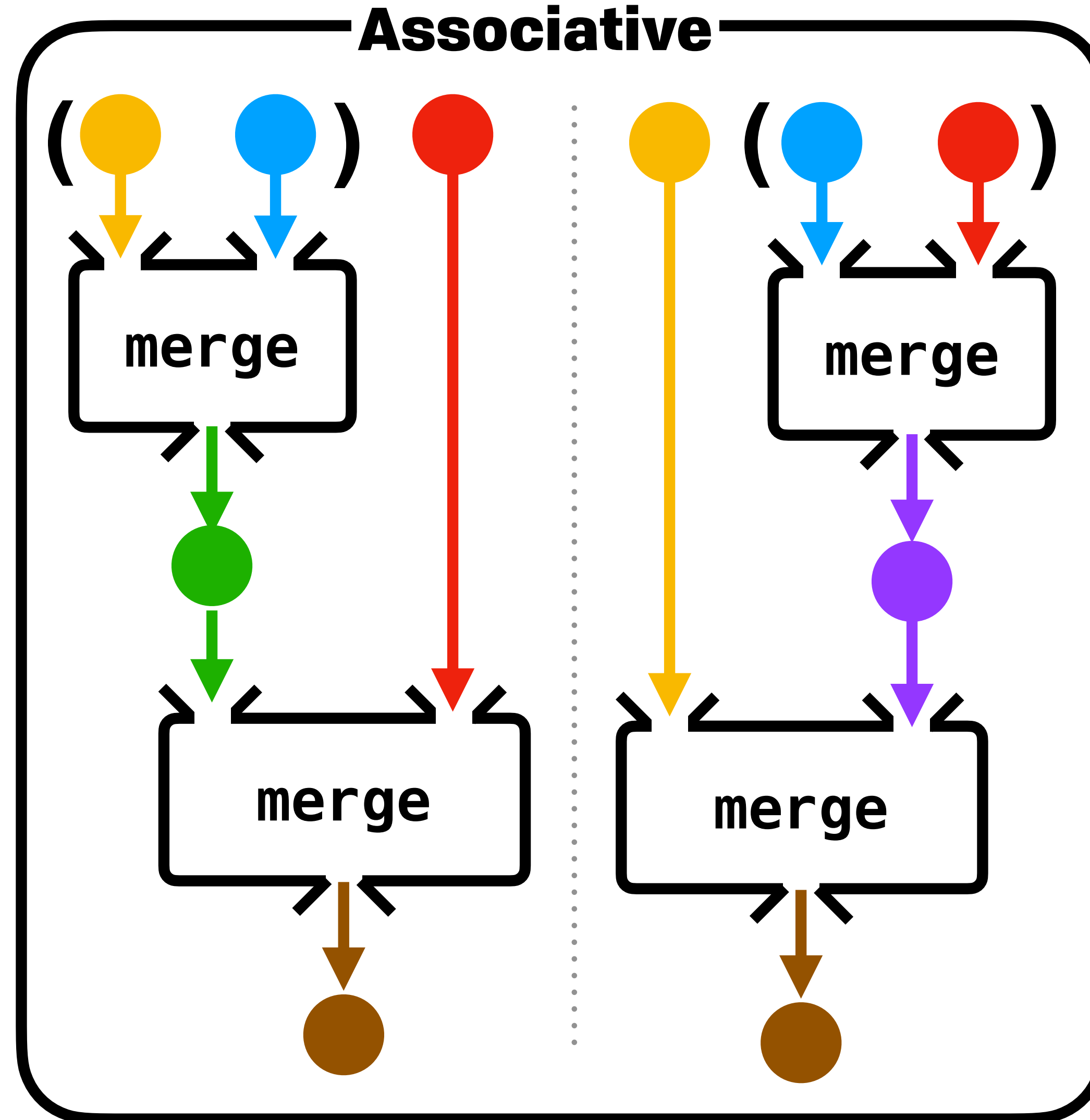
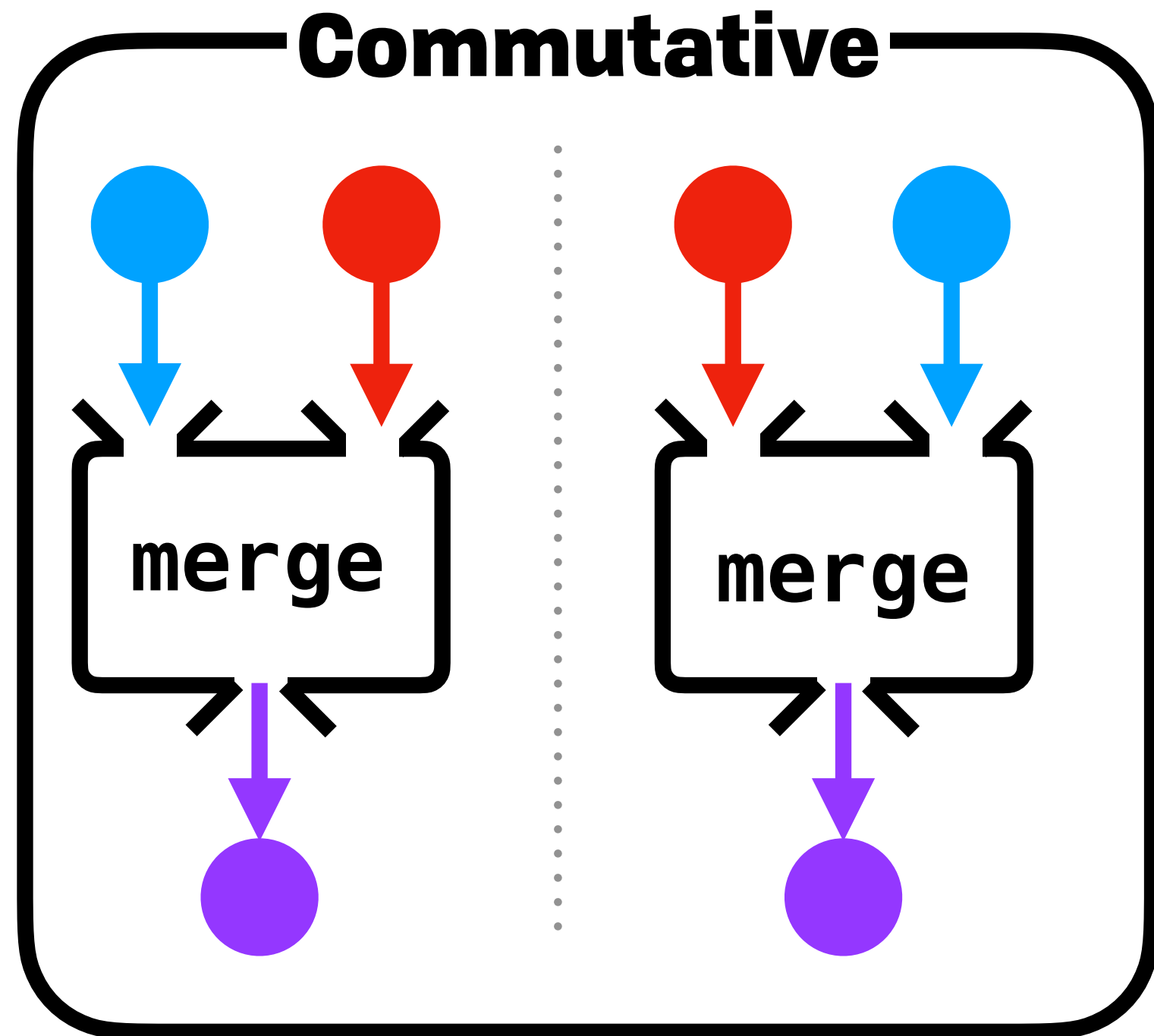
- Network agnostic
- Any number of replicas

Abstract



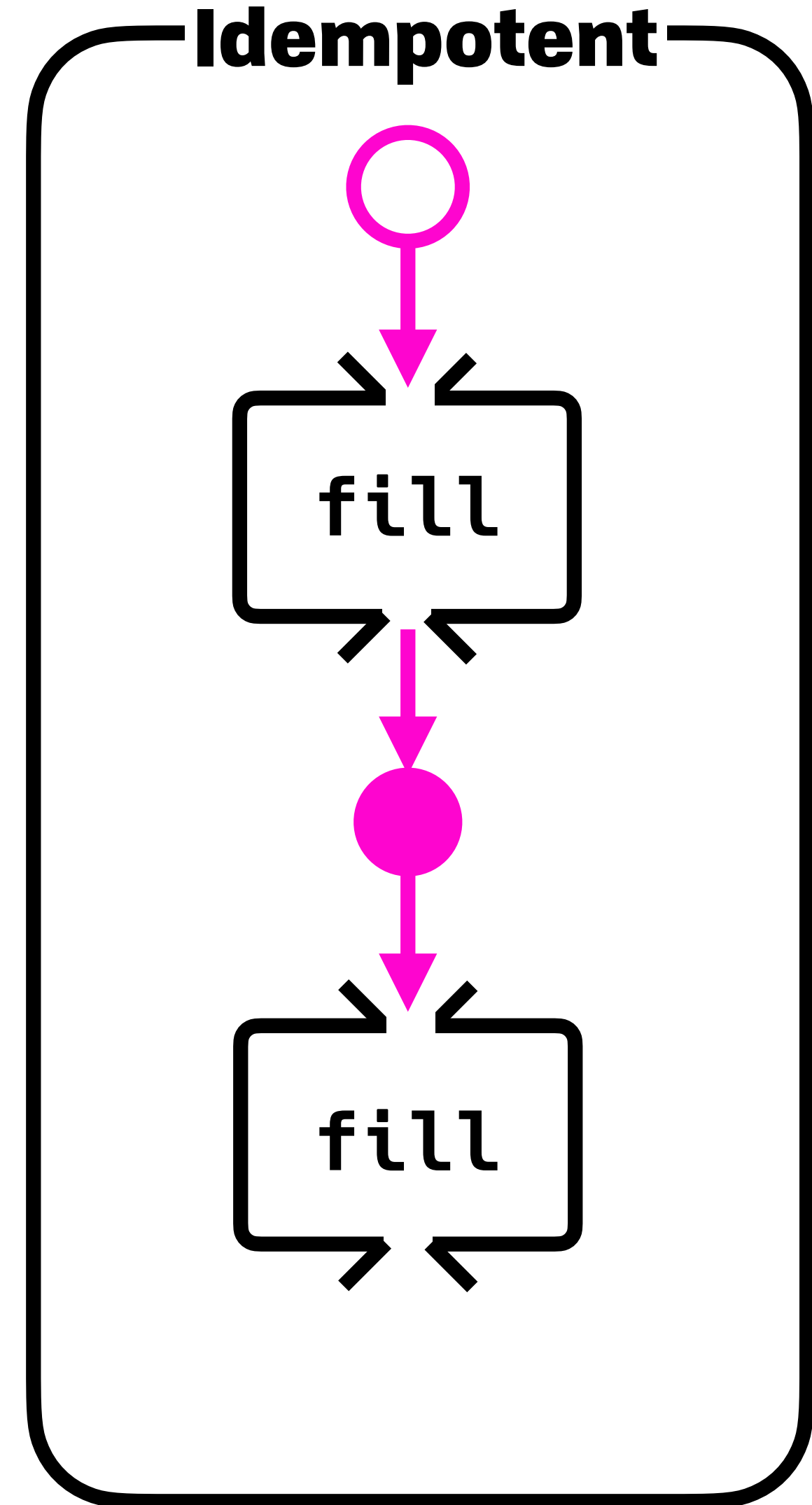
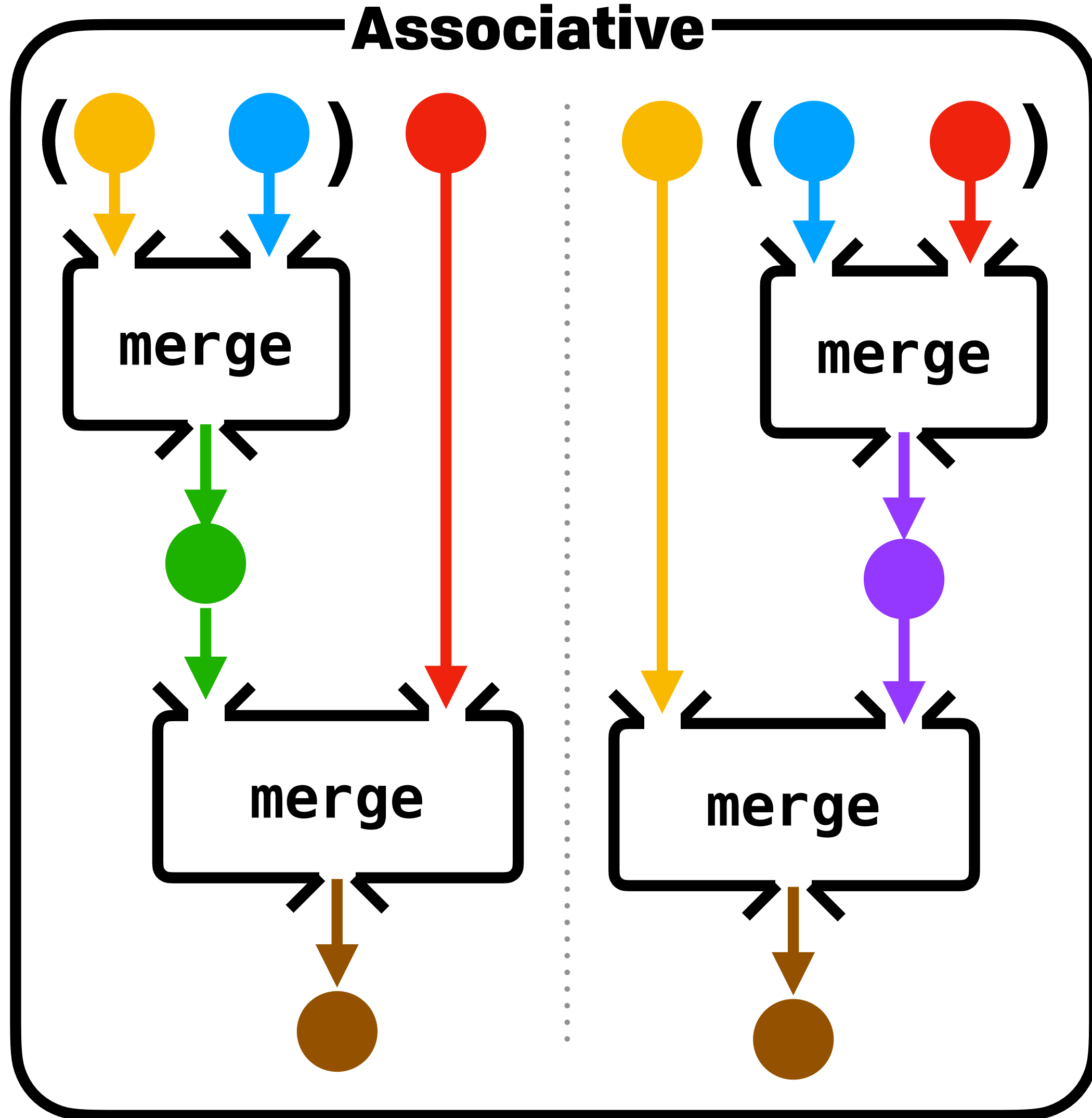
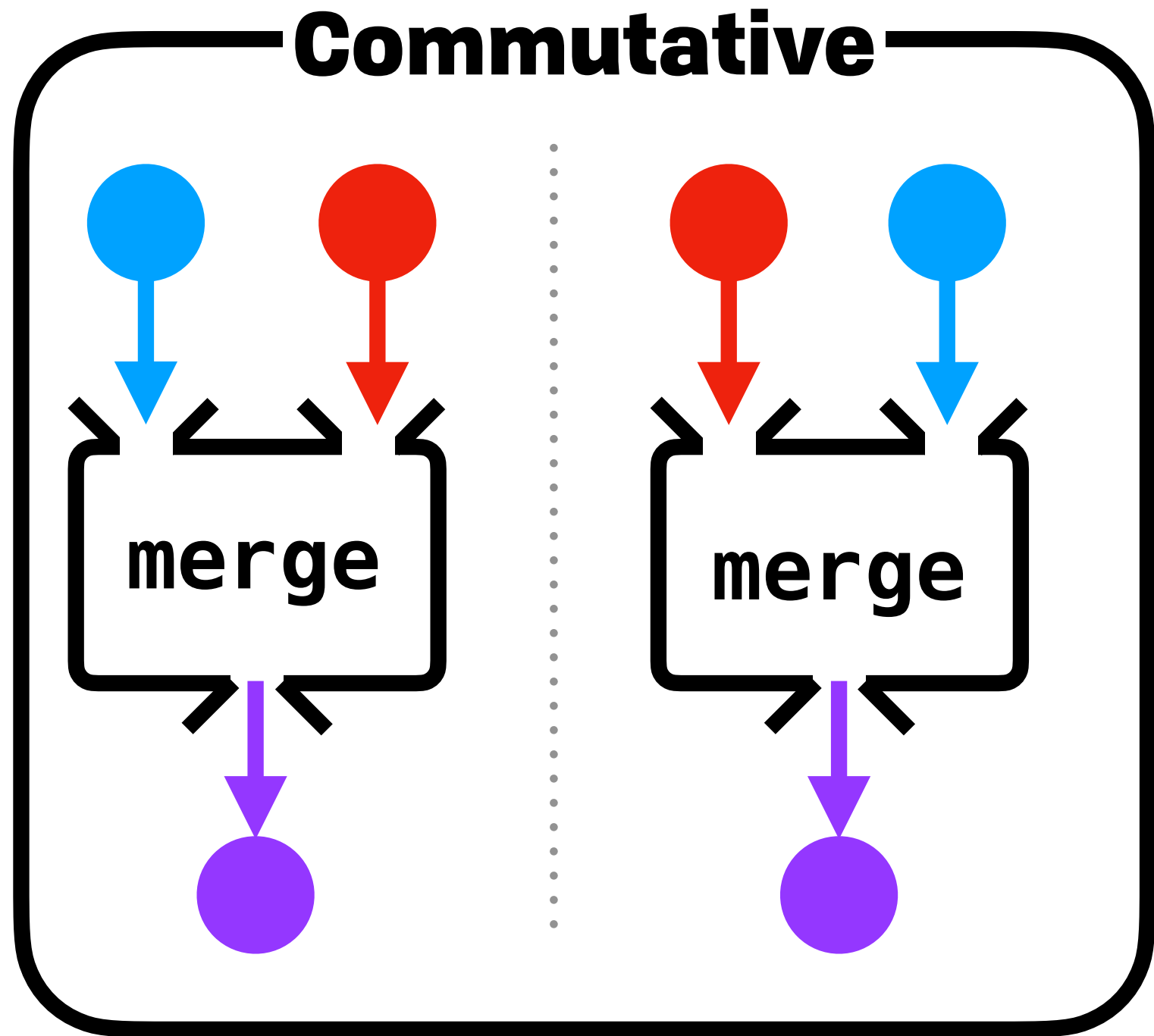
- Network agnostic
- Any number of replicas

Abstract



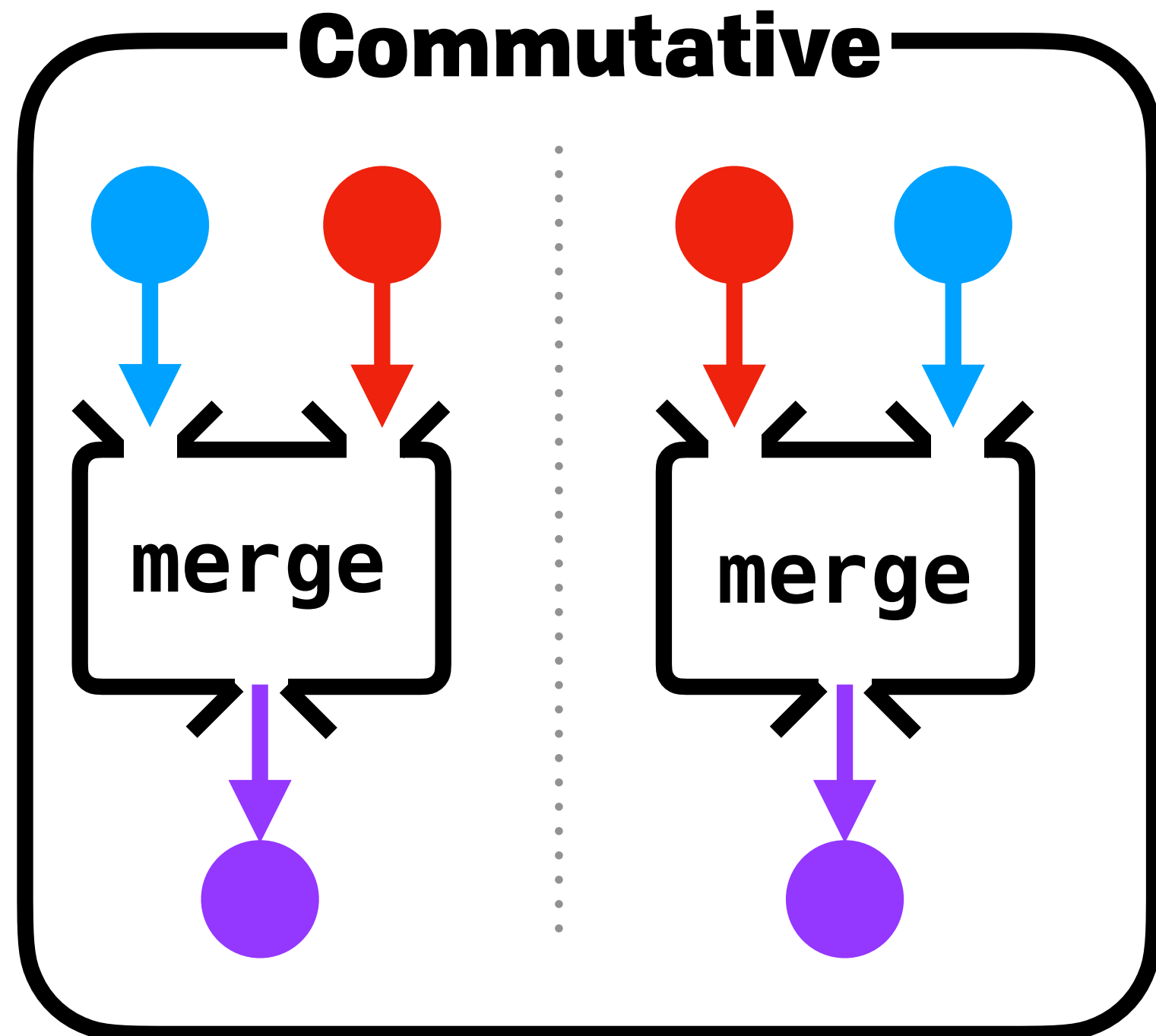
- Network agnostic
- Any number of replicas

Abstract

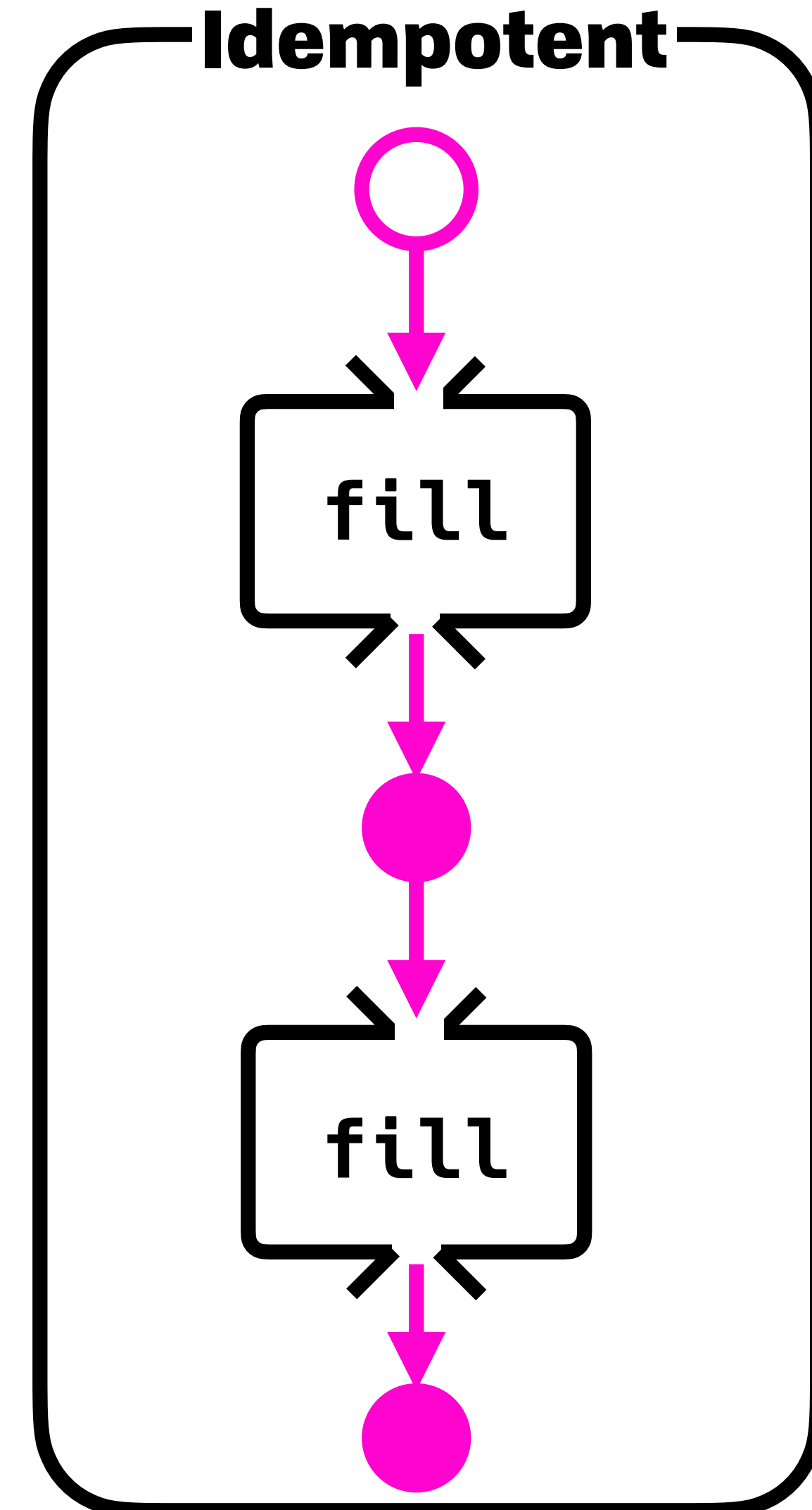
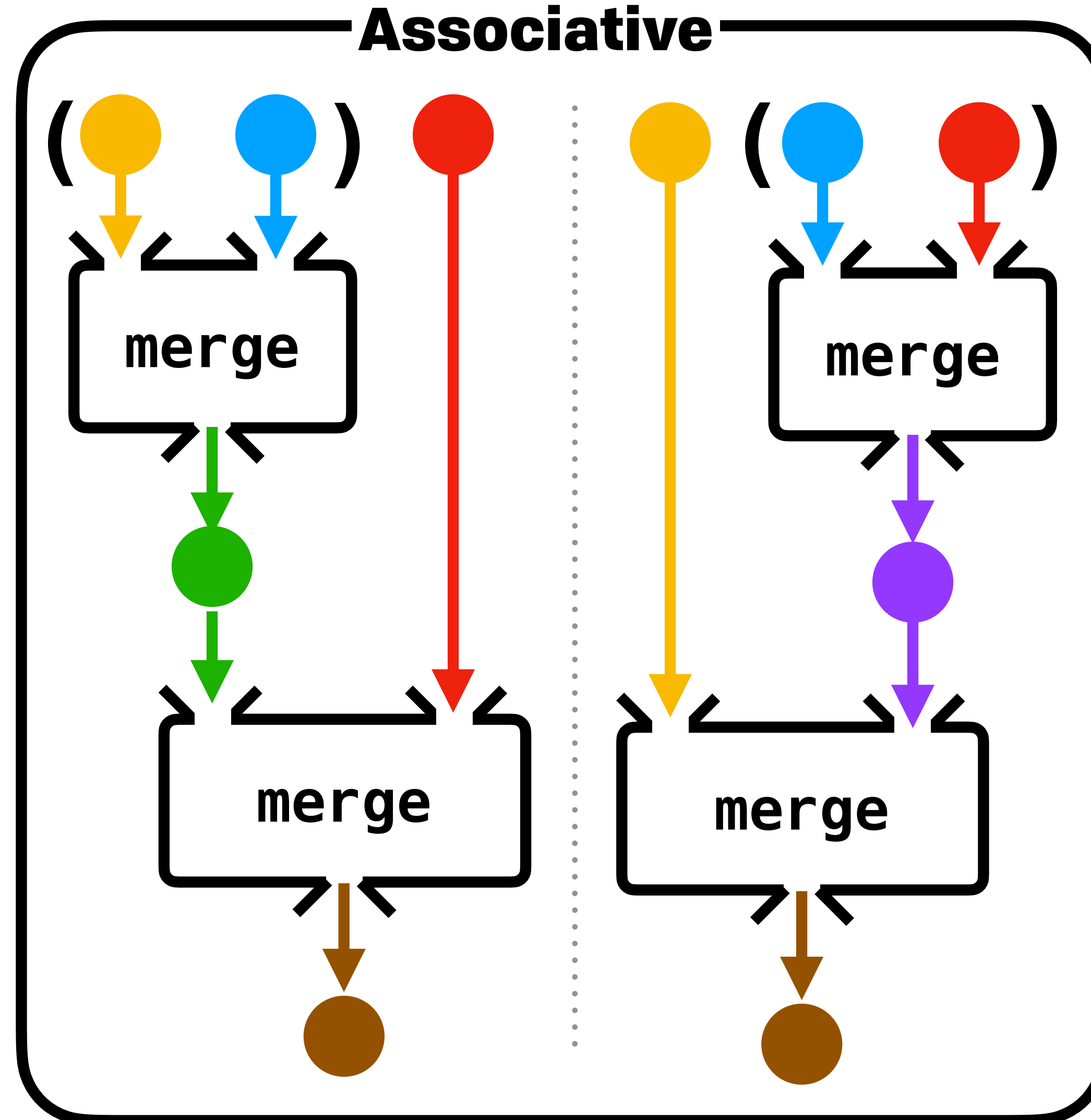


- Network agnostic
- Any number of replicas

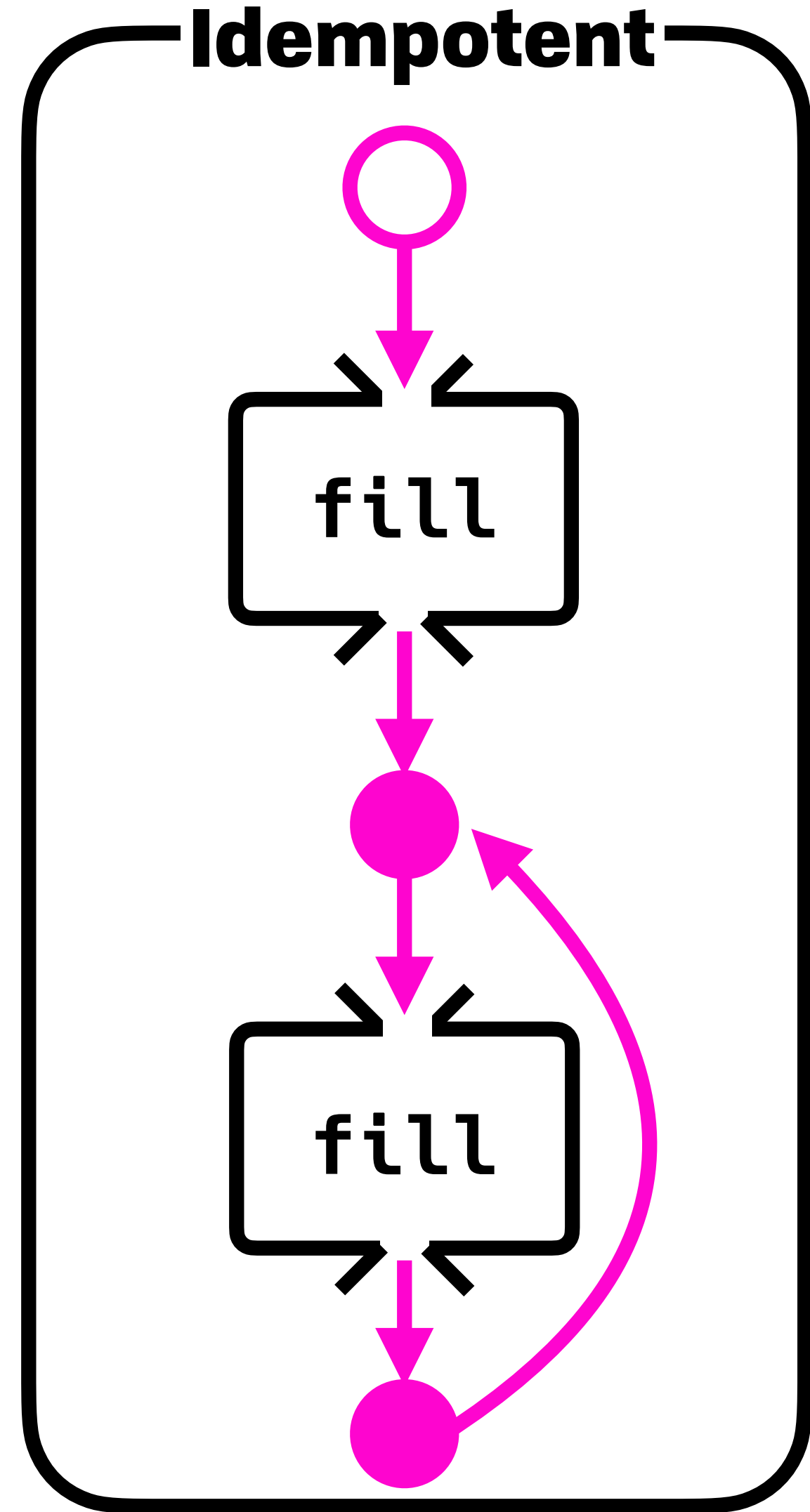
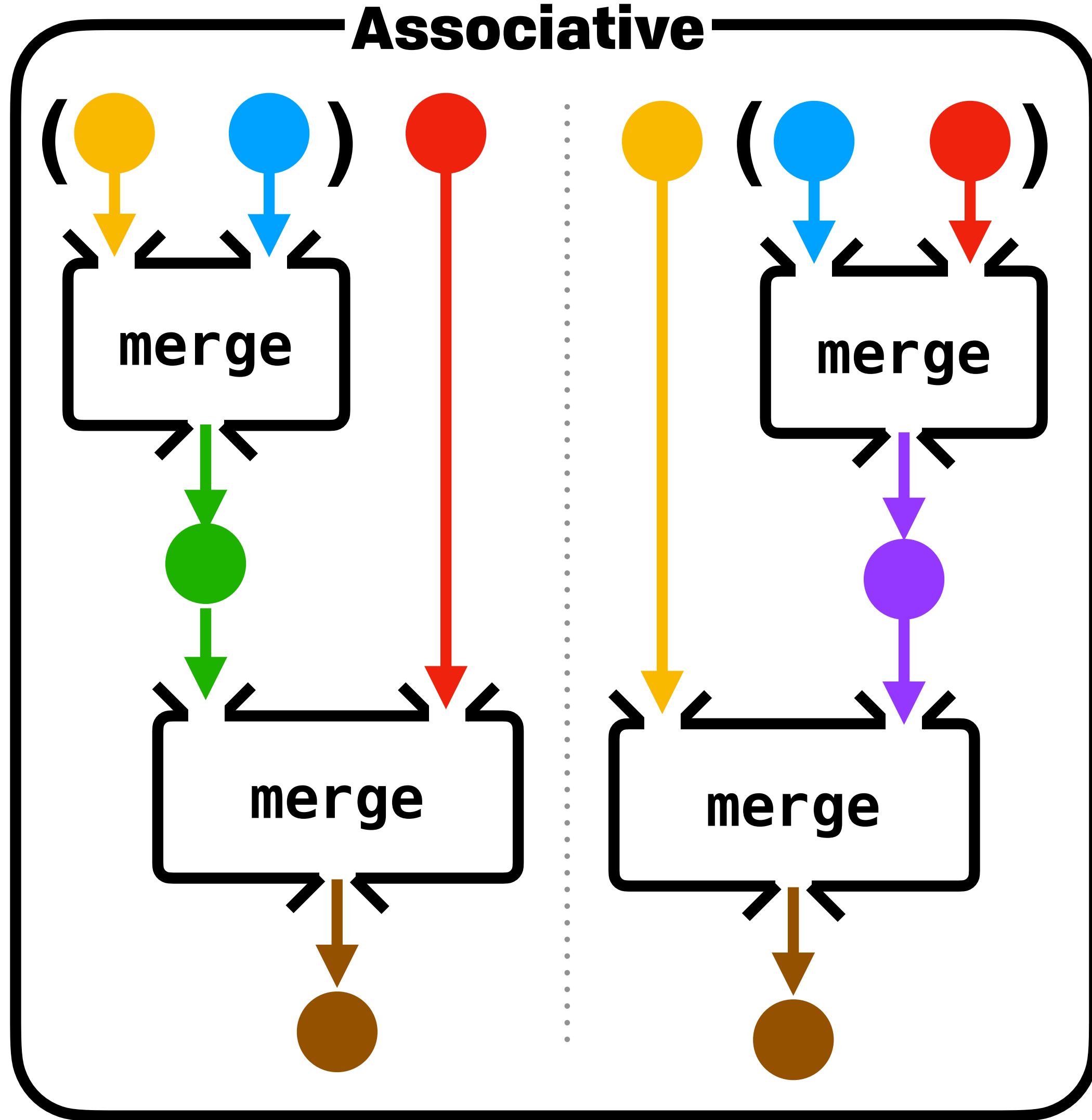
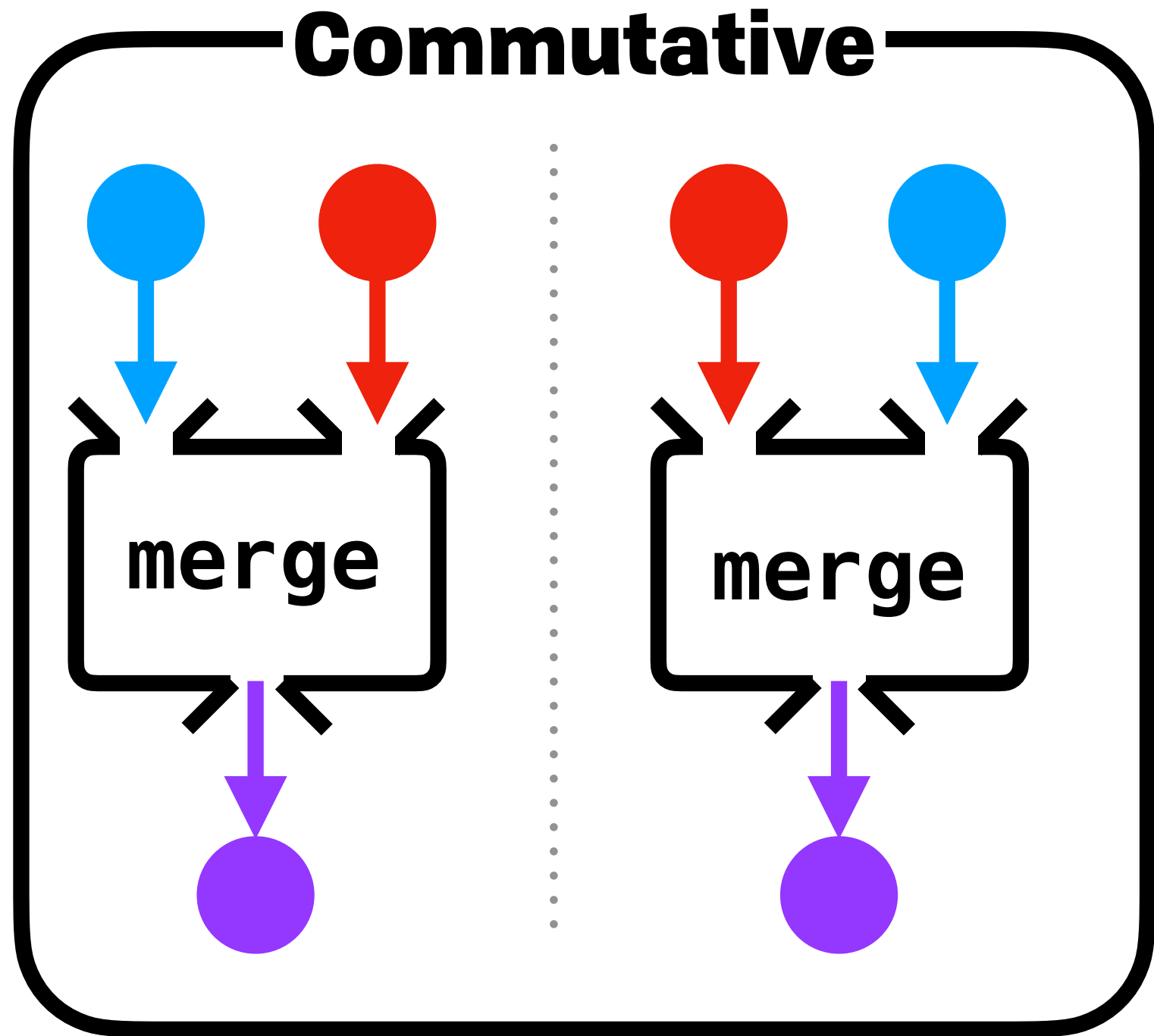
Abstract



- Network agnostic
- Any number of replicas



Abstract



- Network agnostic
- Any number of replicas

It's All About that Data 

PNCOUNTER

It's All About that Data 

PNCounter

```
defmodule PNCounter do
  defstruct [adds: MapSet.new(), removes: MapSet.new()]
```

It's All About that Data 

PNCounter

```
defmodule PNCounter do  
  defstruct [adds: MapSet.new(), removes: MapSet.new()]
```

It's All About that Data 

PNCounter

```
defmodule PNCounter do
  defstruct [adds: MapSet.new(), removes: MapSet.new()]

  def nonce() do
    big = Integer.pow(2, 256)
    Enum.random(0..big)
  end
end
```


It's All About that Data 

PNCounter

```
defmodule PNCounter do
  defstruct [adds: MapSet.new(), removes: MapSet.new()]

  def nonce() do
    big = Integer.pow(2, 256)
    Enum.random(0..big)
  end

  def count(%PNCounter{adds: adds, removes: removes}) do
    adds
    |> MapSet.difference(removes)
    |> MapSet.size()
  end
end
```

It's All About that Data 

PNCounter

```
defmodule PNCounter do
  defstruct [adds: MapSet.new(), removes: MapSet.new()]

  def nonce() do
    big = Integer.pow(2, 256)
    Enum.random(0..big)
  end

  def count(%PNCounter{adds: adds, removes: removes}) do
    adds
    |> MapSet.difference(removes)
    |> MapSet.size()
  end
end
```

It's All About that Data 

PNCounter

```
defmodule PNCounter do
  defstruct [adds: MapSet.new(), removes: MapSet.new()]

  def nonce() do
    big = Integer.pow(2, 256)
    Enum.random(0..big)
  end

  def count(%PNCounter{adds: adds, removes: removes}) do
    adds
    |> MapSet.difference(removes)
    |> MapSet.size()
  end

  def insert(counter = %PNCounter{adds: adds}, nonce) do
    %{counter | adds: MapSet.put(adds, nonce)}
  end
end
```

It's All About that Data 

PNCounter

```
defmodule PNCounter do
  defstruct [adds: MapSet.new(), removes: MapSet.new()]

  def nonce() do
    big = Integer.pow(2, 256)
    Enum.random(0..big)
  end

  def count(%PNCounter{adds: adds, removes: removes}) do
    adds
    |> MapSet.difference(removes)
    |> MapSet.size()
  end

  def insert(counter = %PNCounter{adds: adds}, nonce) do
    %{counter | adds: MapSet.put(adds, nonce)}
  end
end
```

It's All About that Data 

PNCounter

```
defmodule PNCounter do
  defstruct [adds: MapSet.new(), removes: MapSet.new()]

  def nonce() do
    big = Integer.pow(2, 256)
    Enum.random(0..big)
  end

  def count(%PNCounter{adds: adds, removes: removes}) do
    adds
    |> MapSet.difference(removes)
    |> MapSet.size()
  end

  def insert(counter = %PNCounter{adds: adds}, nonce) do
    %{counter | adds: MapSet.put(adds, nonce)}
  end

  def remove(counter = %PNCounter{removes: removes}, nonce) do
    %{counter | removes: MapSet.put(removes, nonce)}
  end
end
```

It's All About that Data 

PNCounter

```
defmodule PNCounter do
  defstruct [adds: MapSet.new(), removes: MapSet.new()]

  def nonce() do
    big = Integer.pow(2, 256)
    Enum.random(0..big)
  end

  def count(%PNCounter{adds: adds, removes: removes}) do
    adds
    |> MapSet.difference(removes)
    |> MapSet.size()
  end

  def insert(counter = %PNCounter{adds: adds}, nonce) do
    %{counter | adds: MapSet.put(adds, nonce)}
  end

  def remove(counter = %PNCounter{removes: removes}, nonce) do
    %{counter | removes: MapSet.put(removes, nonce)}
  end
end
```

It's All About that Data 

PNCounter

```
%PNCounter{} # => 0
|> PNCounter.insert(42) # => 1
|> PNCounter.insert(123) # => 2
|> PNCounter.insert(999_999) # => 3
|> PNCounter.remove(999_999) # => 2
|> PNCounter.count()
# => 2
```

```
defmodule PNCounter do
  defstruct [adds: MapSet.new(), removes: MapSet.new()]

  def nonce() do
    big = Integer.pow(2, 256)
    Enum.random(0..big)
  end

  def count(%PNCounter{adds: adds, removes: removes}) do
    adds
    |> MapSet.difference(removes)
    |> MapSet.size()
  end

  def insert(counter = %PNCounter{adds: adds}, nonce) do
    %{counter | adds: MapSet.put(adds, nonce)}
  end

  def remove(counter = %PNCounter{removes: removes}, nonce) do
    %{counter | removes: MapSet.put(removes, nonce)}
  end
end
```

It's All About that Data

PNCounter

```
%PNCounter{} # => 0
|> PNCounter.insert(42) # => 1
|> PNCounter.insert(123) # => 2
|> PNCounter.insert(999_999) # => 3
|> PNCounter.remove(999_999) # => 2
|> PNCounter.count()
# => 2
```

```
%PNCounter{} # => 0
|> PNCounter.insert(123) # => 1
|> PNCounter.insert(123) # => 1
|> PNCounter.insert(123) # => 1
|> PNCounter.remove(999_999) # => 1
|> PNCounter.insert(42) # => 2
|> PNCounter.insert(999_999) # => 2
|> PNCounter.insert(42) # => 2
|> PNCounter.count()
# => 2
```

```
defmodule PNCounter do
  defstruct [adds: MapSet.new(), removes: MapSet.new()]

  def nonce() do
    big = Integer.pow(2, 256)
    Enum.random(0..big)
  end

  def count(%PNCounter{adds: adds, removes: removes}) do
    adds
    |> MapSet.difference(removes)
    |> MapSet.size()
  end

  def insert(counter = %PNCounter{adds: adds}, nonce) do
    %{counter | adds: MapSet.put(adds, nonce)}
  end

  def remove(counter = %PNCounter{removes: removes}, nonce) do
    %{counter | removes: MapSet.put(removes, nonce)}
  end
end
```


It's All About that Data

PNCounter

```
%PNCounter{} # => 0
|> PNCounter.insert(42) # => 1
|> PNCounter.insert(123) # => 2
|> PNCounter.insert(999_999) # => 3
|> PNCounter.remove(999_999) # => 2
|> PNCounter.count()
# => 2
```

```
%PNCounter{} # => 0
|> PNCounter.insert(123) # => 1
|> PNCounter.insert(123) # => 1
|> PNCounter.insert(123) # => 1
|> PNCounter.remove(999_999) # => 1
|> PNCounter.insert(42) # => 2
|> PNCounter.insert(999_999) # => 2
|> PNCounter.insert(42) # => 2
|> PNCounter.count()
# => 2
```

```
defmodule PNCounter do
  defstruct [adds: MapSet.new(), removes: MapSet.new()]

  def nonce() do
    big = Integer.pow(2, 256)
    Enum.random(0..big)
  end

  def count(%PNCounter{adds: adds, removes: removes}) do
    adds
    |> MapSet.difference(removes)
    |> MapSet.size()
  end

  def insert(counter = %PNCounter{adds: adds}, nonce) do
    %{counter | adds: MapSet.put(adds, nonce)}
  end

  def remove(counter = %PNCounter{removes: removes}, nonce) do
    %{counter | removes: MapSet.put(removes, nonce)}
  end
end
```

It's All About that Data

PNCounter

```
%PNCounter{} # => 0
|> PNCounter.insert(42) # => 1
|> PNCounter.insert(123) # => 2
|> PNCounter.insert(999_999) # => 3
|> PNCounter.remove(999_999) # => 2
|> PNCounter.count()
# => 2
```

```
%PNCounter{} # => 0
|> PNCounter.insert(123) # => 1
|> PNCounter.insert(123) # => 1
|> PNCounter.insert(123) # => 1
|> PNCounter.remove(999_999) # => 1
|> PNCounter.insert(42) # => 2
|> PNCounter.insert(999_999) # => 2
|> PNCounter.insert(42) # => 2
|> PNCounter.count()
# => 2
```

```
defmodule PNCounter do
  defstruct [adds: MapSet.new(), removes: MapSet.new()]

  def nonce() do
    big = Integer.pow(2, 256)
    Enum.random(0..big)
  end

  def count(%PNCounter{adds: adds, removes: removes}) do
    adds
    |> MapSet.difference(removes)
    |> MapSet.size()
  end

  def insert(counter = %PNCounter{adds: adds}, nonce) do
    %{counter | adds: MapSet.put(adds, nonce)}
  end

  def remove(counter = %PNCounter{removes: removes}, nonce) do
    %{counter | removes: MapSet.put(removes, nonce)}
  end
end
```

It's All About that Data 

Syncing Tables Seems Inefficient

It's All About that Data 

Syncing Tables Seems Inefficient

| user_id | username | company | start_date | inserted_at |
|----------------|-----------------|----------------|-------------------|--------------------|
| 1 | expede | Fission | AUG-2019 | FEB-2020 |
| 2 | bmann | -- | -- | OCT-2020 |

It's All About that Data 

Syncing Tables Seems Inefficient

| user_id | username | company | start_date | inserted_at |
|----------------|-----------------|----------------|-------------------|--------------------|
| 1 | expede | Fission | AUG-2019 | FEB-2020 |
| 2 | bmann | -- | -- | OCT-2020 |

| kb_id | owner_id | mode | switches | inserted_at |
|--------------|-----------------|-----------------|-----------------|--------------------|
| 42 | 1 | Wireless | Blue | JAN-2020 |

It's All About that Data 

Syncing Tables Seems Inefficient

Who's clock?

Meaningful or coincidence?

| user_id | username | company | start_date | inserted_at |
|----------------|-----------------|----------------|-------------------|--------------------|
| 1 | expede | Fission | AUG-2019 | FEB-2020 |
| 2 | bmann | -- | -- | OCT-2020 |

| kb_id | owner_id | mode | switches | inserted_at |
|--------------|-----------------|-----------------|-----------------|--------------------|
| 42 | 1 | Wireless | Blue | JAN-2020 |

It's All About that Data 

Syncing Tables Seems Inefficient



Who's clock?

Meaningful or coincidence?

| user_id | username | company | start_date | inserted_at |
|----------------|-----------------|----------------|-------------------|--------------------|
| 1 | expede | Fission | AUG-2019 | FEB-2020 |
| 2 | bmann | -- | -- | OCT-2020 |

| kb_id | owner_id | mode | switches | inserted_at |
|--------------|-----------------|-----------------|-----------------|--------------------|
| 42 | 1 | Wireless | Blue | JAN-2020 |

It's All About that Data 

Syncing Tables Seems Inefficient



Who's clock?

Meaningful or coincidence?

| user_id | username | company | start_date | inserted_at |
|---------|----------|---------|------------|-------------|
| 1 | expede | Fission | AUG-2019 | FEB-2020 |
| 2 | bmann | -- | -- | OCT-2020 |

| kb_id | owner_id | mode | switches | inserted_at |
|-------|----------|----------|----------|-------------|
| 42 | 1 | Wireless | Blue | JAN-2020 |

It's All About that Data 

Syncing Tables Seems Inefficient



Who's clock?

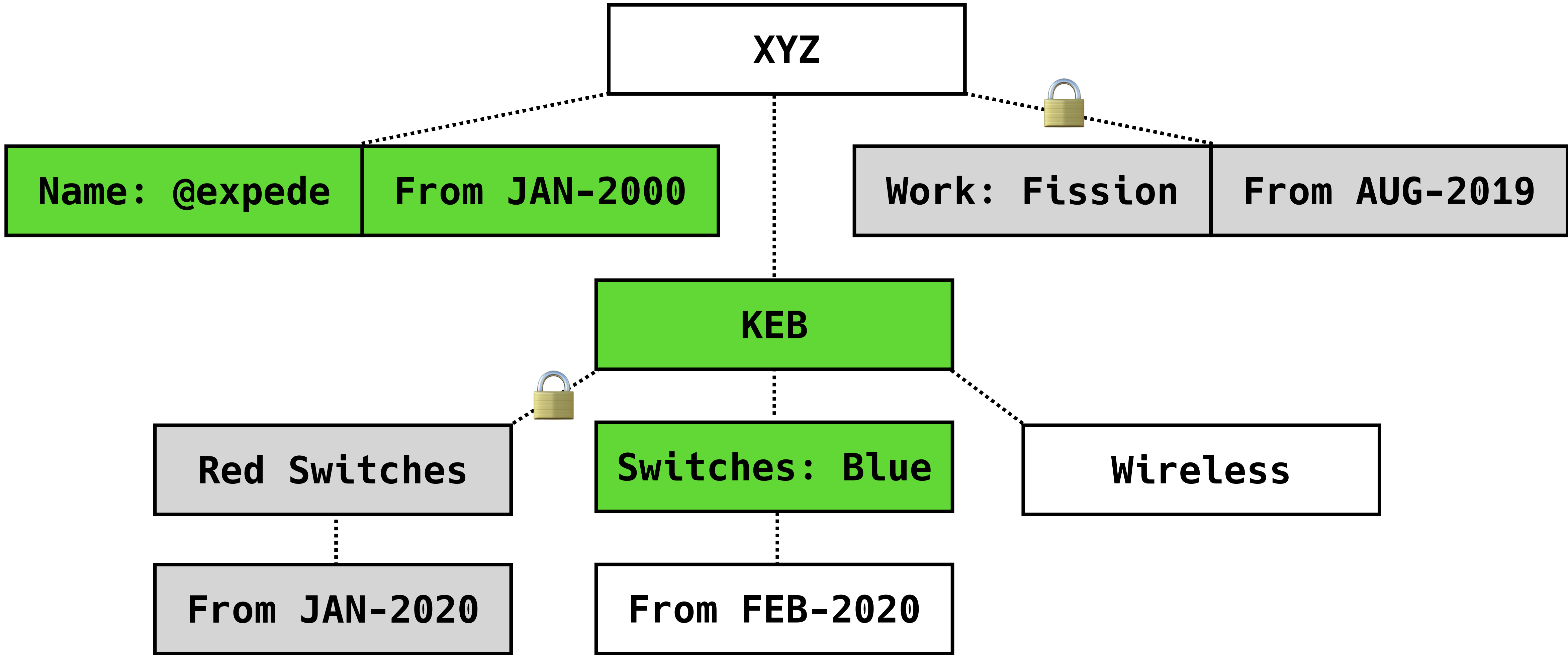
Meaningful or coincidence?

| user_id | username | company | start_date | inserted_at |
|---------|----------|---------|------------|-------------|
| 1 | expede | Fission | AUG-2019 | FEB-2020 |
| 2 | bmann | -- | -- | OCT-2020 |

| kb_id | owner_id | mode | switches | inserted_at |
|-------|----------|----------|----------|-------------|
| 42 | 1 | Wireless | Blue | JAN-2020 |

It's All About that Data 

Relationships



It's All About that Data 

Relationships

XYZ



Name: @expede From JAN-2000

Work: Fission From AUG-2019

KEB



Red Switches

Switches: Blue



Wireless

From JAN-2020

From FEB-2020



It's All About that Data 

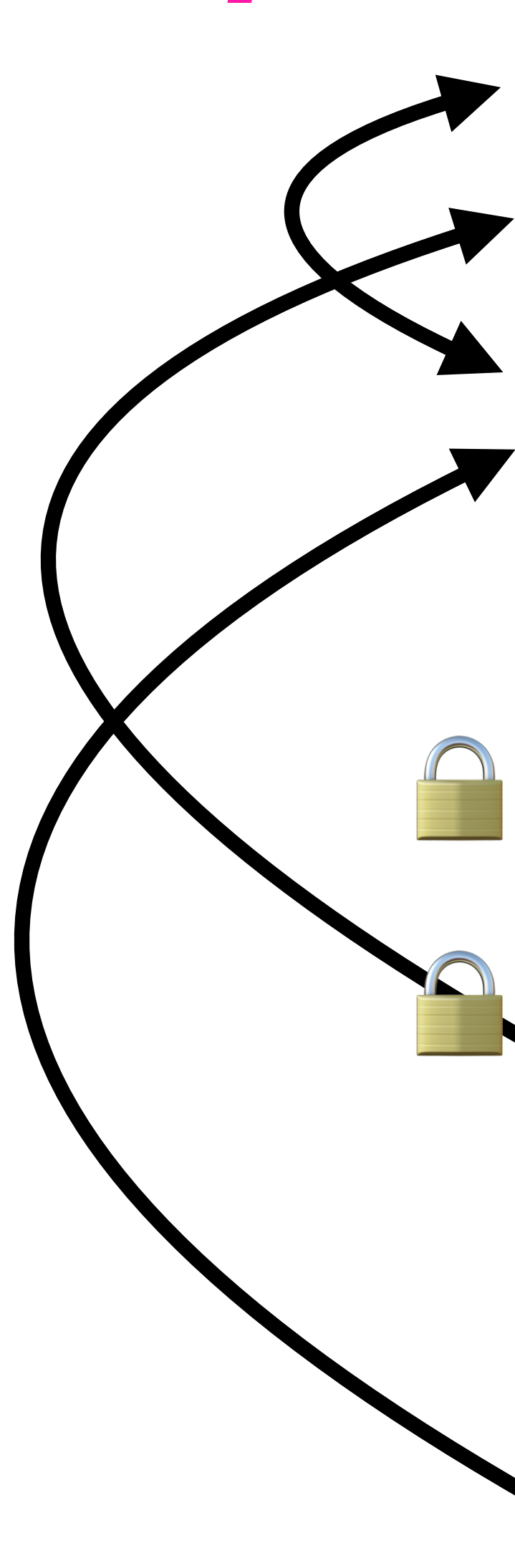
A Sequel to SQL: Nonlinear DBs

| | | |
|---|----------------|---------------|
| XYZ | Name: @expede | From JAN-2000 |
| ABC | Name: @bmann | From DEC-1999 |
| KEB | Type: Wireless | Always |
|  XYZ | Work: Fission | From AUG-2019 |
|  KEB | Switches: Red | From JAN-2020 |
| KEB | Owner: XYZ | From JAN-2020 |
| KEB | Switches: Blue | From FEB-2020 |

It's All About that Data 



A Sequel to SQL: Nonlinear DBs

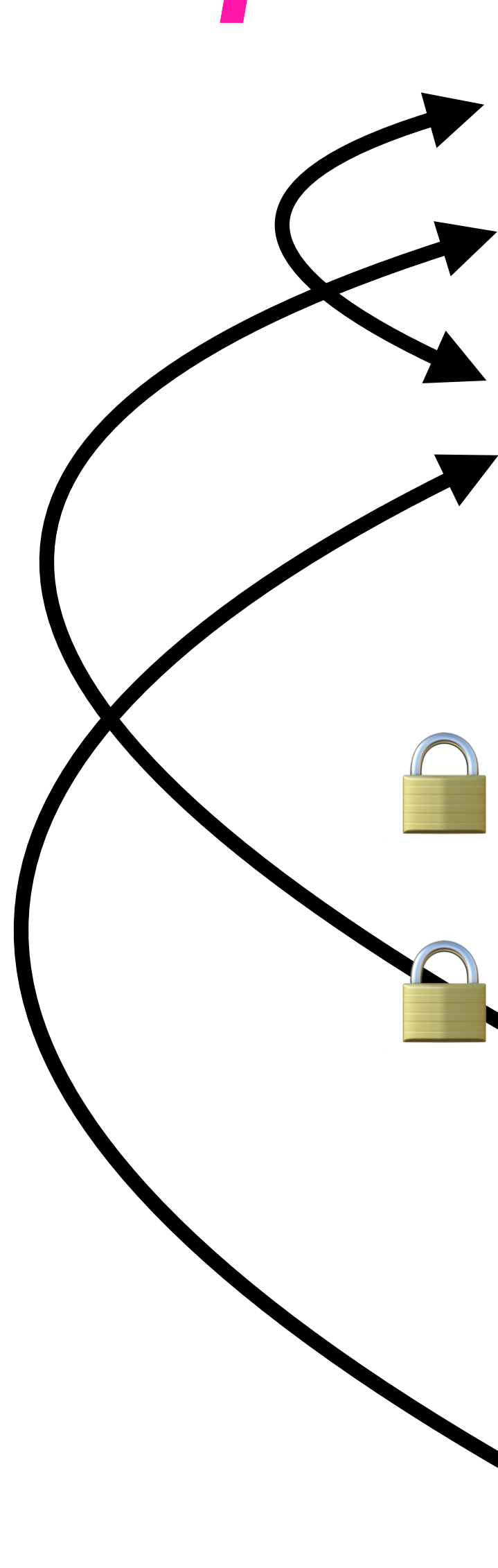
| | | |
|---|----------------|---------------|
| XYZ | Name: @expede | From JAN-2000 |
| ABC | Name: @bmann | From DEC-1999 |
| KEB | Type: Wireless | Always |
|  XYZ | Work: Fission | From AUG-2019 |
|  KEB | Switches: Red | From JAN-2020 |
| KEB | Owner: XYZ | From JAN-2020 |
| KEB | Switches: Blue | From FEB-2020 |



It's All About that Data 



A Sequel to SQL: Nonlinear DBs

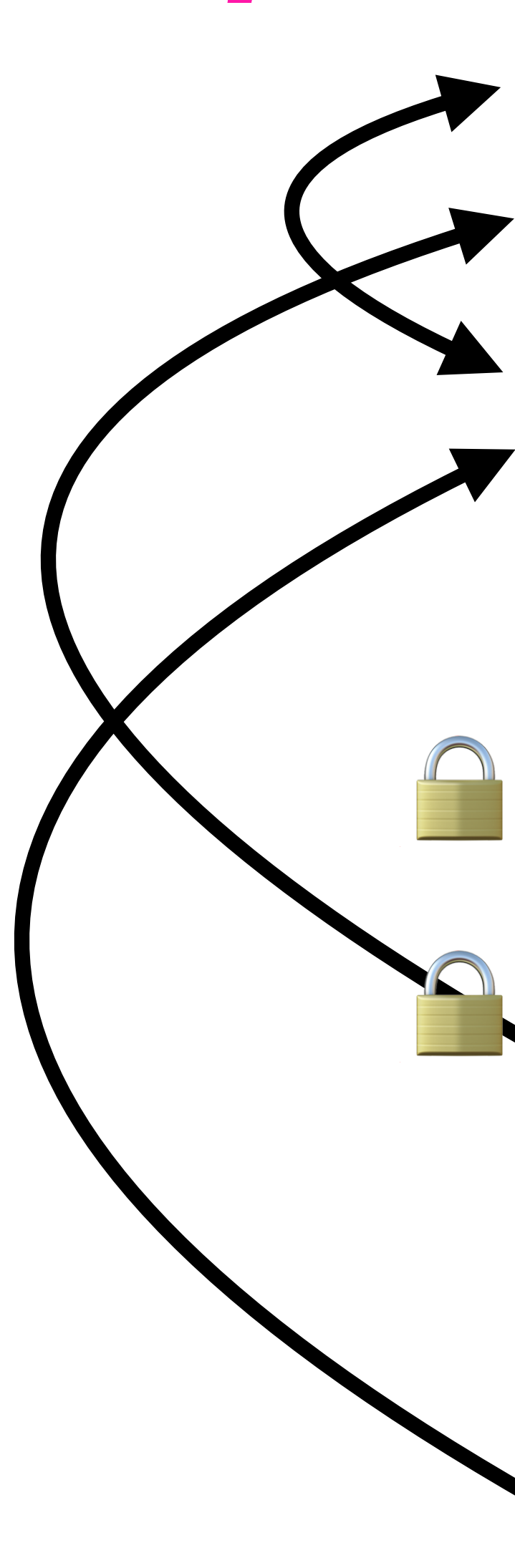
| | | |
|---|----------------|---------------|
| XYZ | Name: @expede | From JAN-2000 |
| ABC | Name: @bmann | From DEC-1999 |
| KEB | Type: Wireless | Always |
|  XYZ | Work: Fission | From AUG-2019 |
|  KEB | Switches: Red | From JAN-2020 |
| KEB | Owner: XYZ | From JAN-2020 |
| KEB | Switches: Blue | From FEB-2020 |



It's All About that Data 



A Sequel to SQL: Nonlinear DBs

| | | |
|---|----------------|---------------|
| XYZ | Name: @expede | From JAN-2000 |
| ABC | Name: @bmann | From DEC-1999 |
| KEB | Type: Wireless | Always |
|  XYZ | Work: Fission | From AUG-2019 |
|  KEB | Switches: Red | From JAN-2020 |
| KEB | Owner: XYZ | From JAN-2020 |
| KEB | Switches: Blue | From FEB-2020 |



It's All About that Data 

A Sequel to SQL: Nonlinear DBs

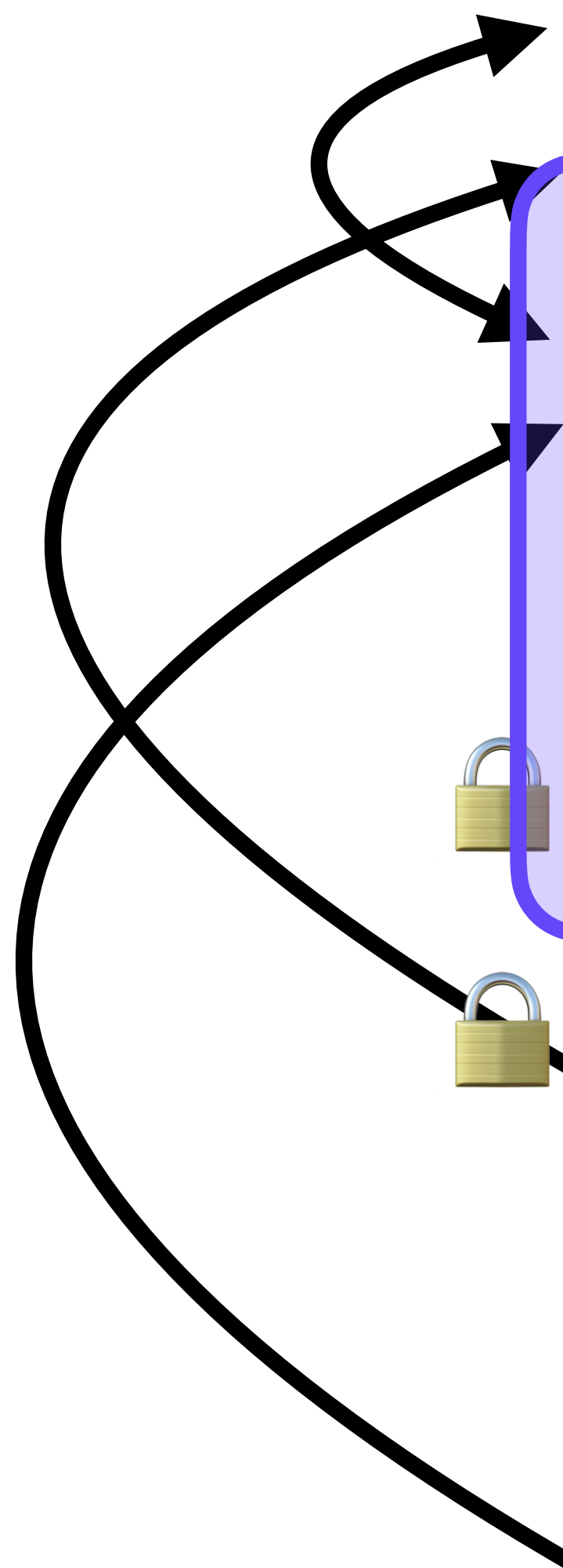
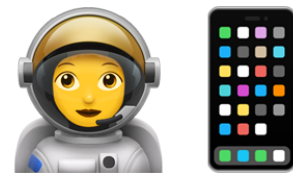
| | | |
|---|----------------|---------------|
| XYZ | Name: @expede | From JAN-2000 |
| ABC | Name: @bmann | From DEC-1999 |
| KEB | Type: Wireless | Always |
|  XYZ | Work: Fission | From AUG-2019 |
|  KEB | Switches: Red | From JAN-2020 |
| KEB | Owner: XYZ | From JAN-2020 |
| KEB | Switches: Blue | From FEB-2020 |



It's All About that Data 

A Sequel to SQL: Nonlinear DBs

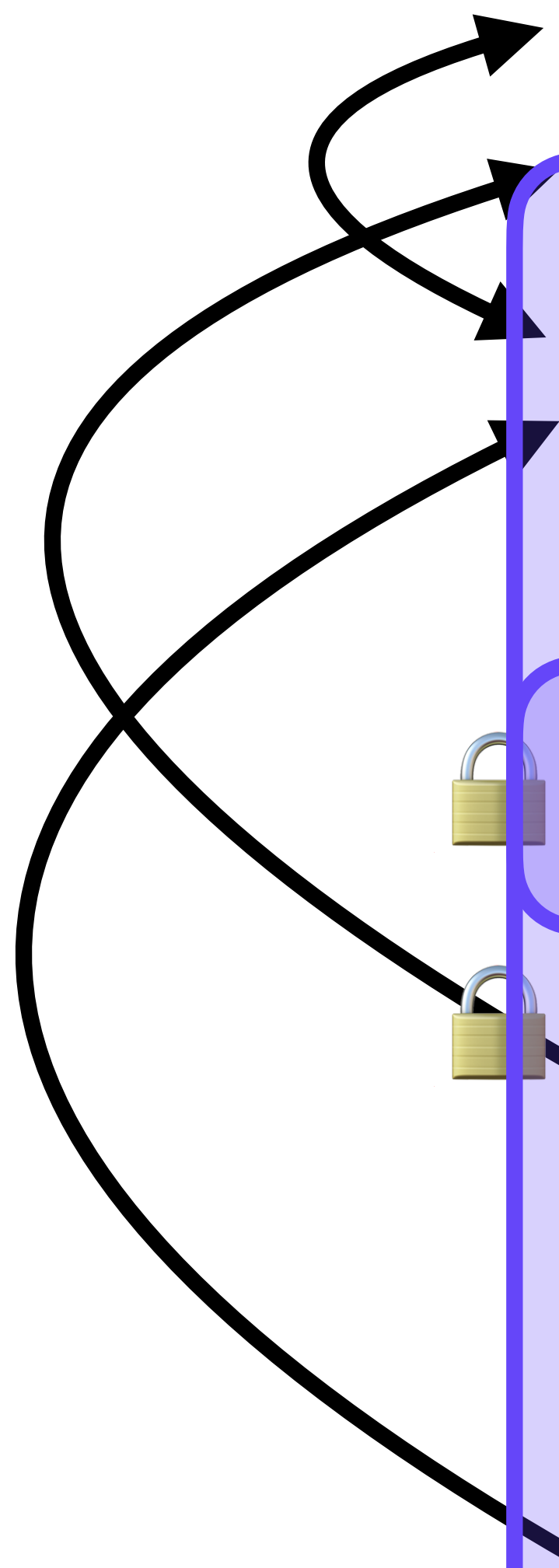
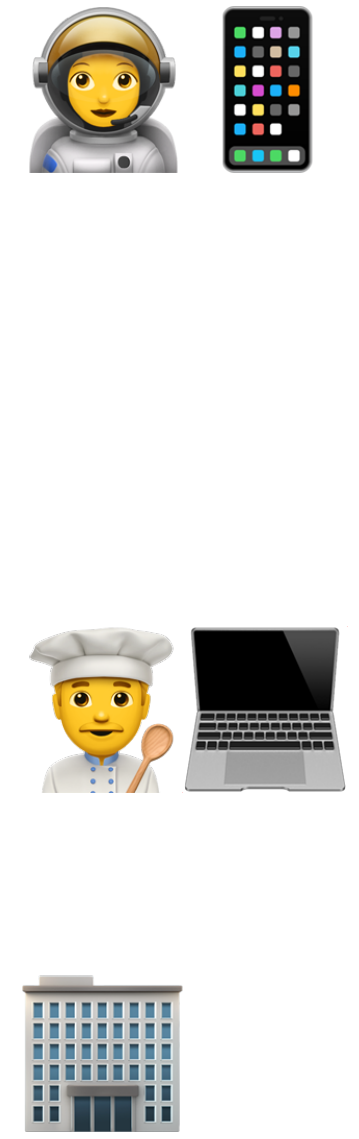
| | | |
|-----|----------------|---------------|
| XYZ | Name: @expede | From JAN-2000 |
| ABC | Name: @bmann | From DEC-1999 |
| KEB | Type: Wireless | Always |
| XYZ | Work: Fission | From AUG-2019 |
| KEB | Switches: Red | From JAN-2020 |
| KEB | Owner: XYZ | From JAN-2020 |
| KEB | Switches: Blue | From FEB-2020 |



It's All About that Data 

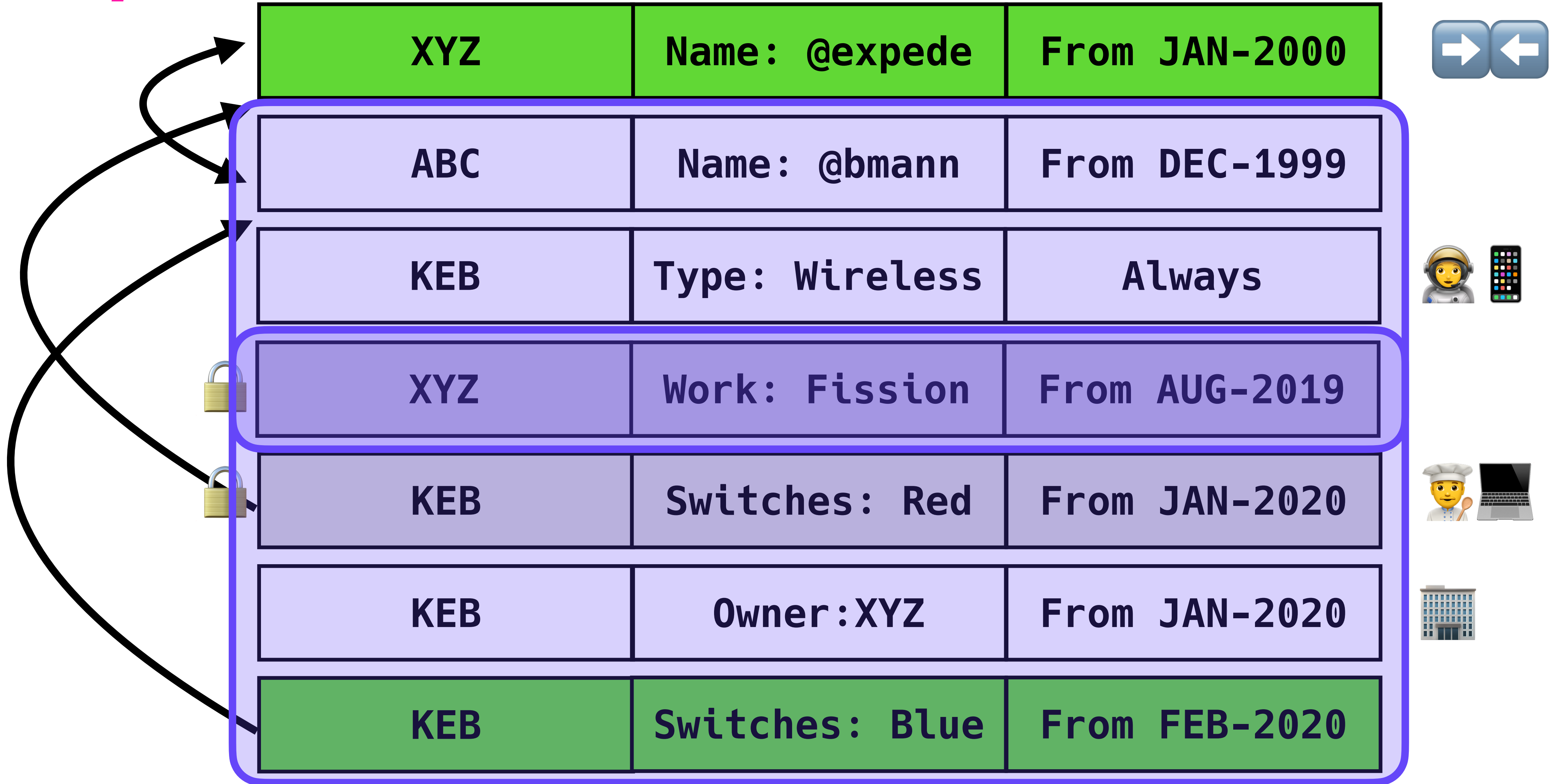
A Sequel to SQL: Nonlinear DBs

| | | |
|-----|----------------|---------------|
| XYZ | Name: @expede | From JAN-2000 |
| ABC | Name: @bmann | From DEC-1999 |
| KEB | Type: Wireless | Always |
| XYZ | Work: Fission | From AUG-2019 |
| KEB | Switches: Red | From JAN-2020 |
| KEB | Owner: XYZ | From JAN-2020 |
| KEB | Switches: Blue | From FEB-2020 |



It's All About that Data 

A Sequel to SQL: Nonlinear DBs



Collaboration & Memoization & Effects at Scale (oh my!)

Universal Compute

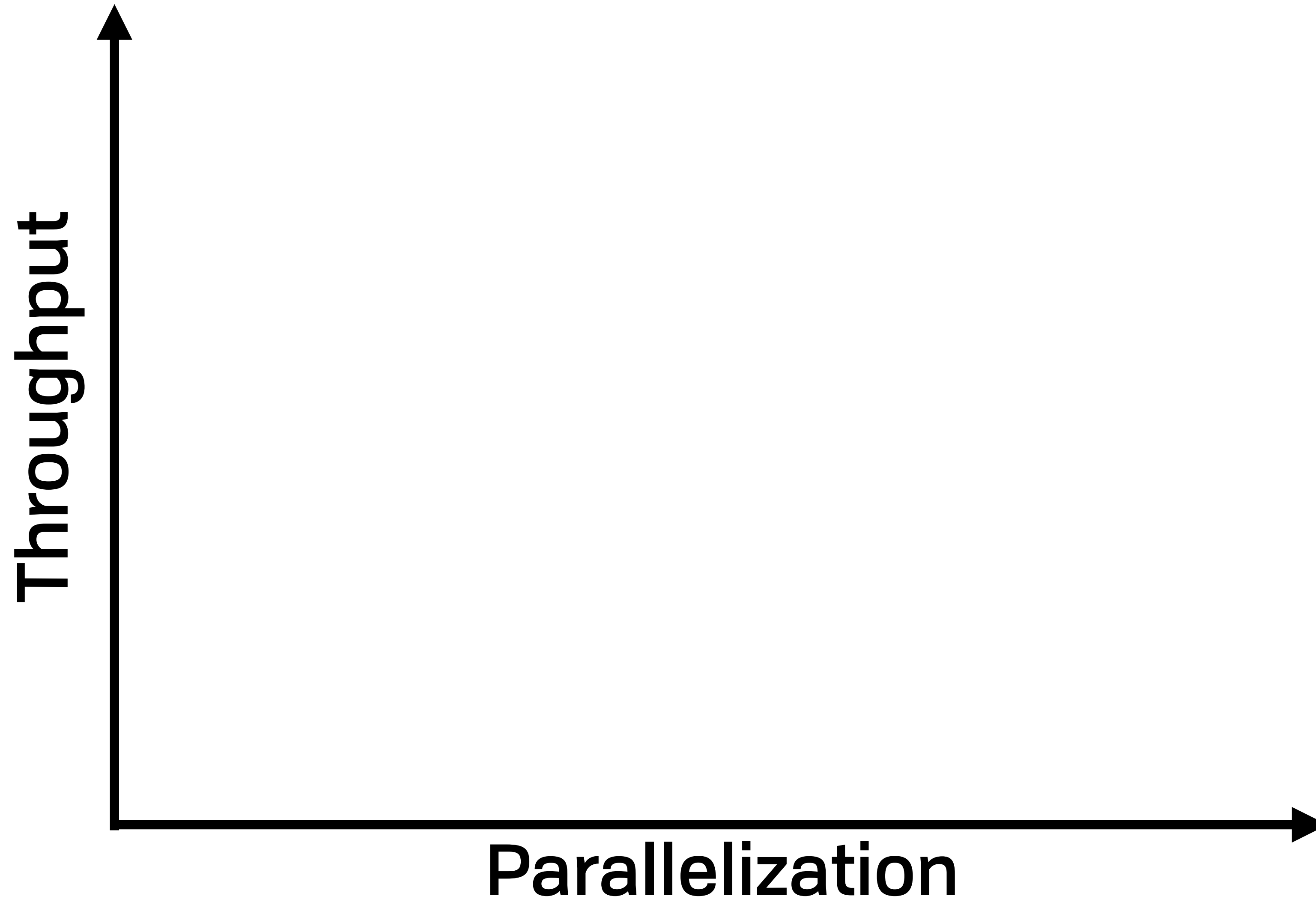


Universal Compute ✨

"With a Little Scale From My Friends"

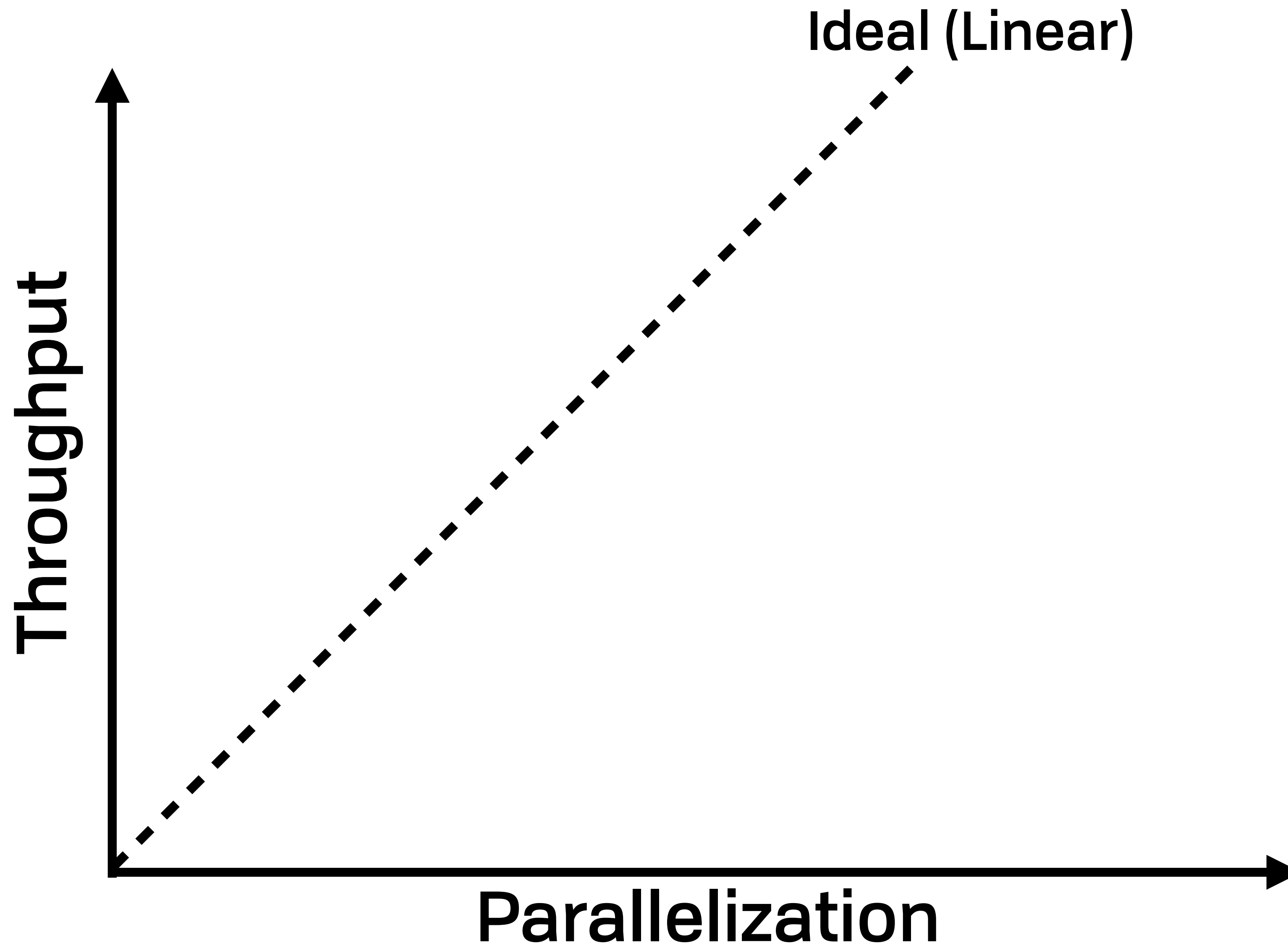
Universal Compute ✨

"With a Little Scale From My Friends"



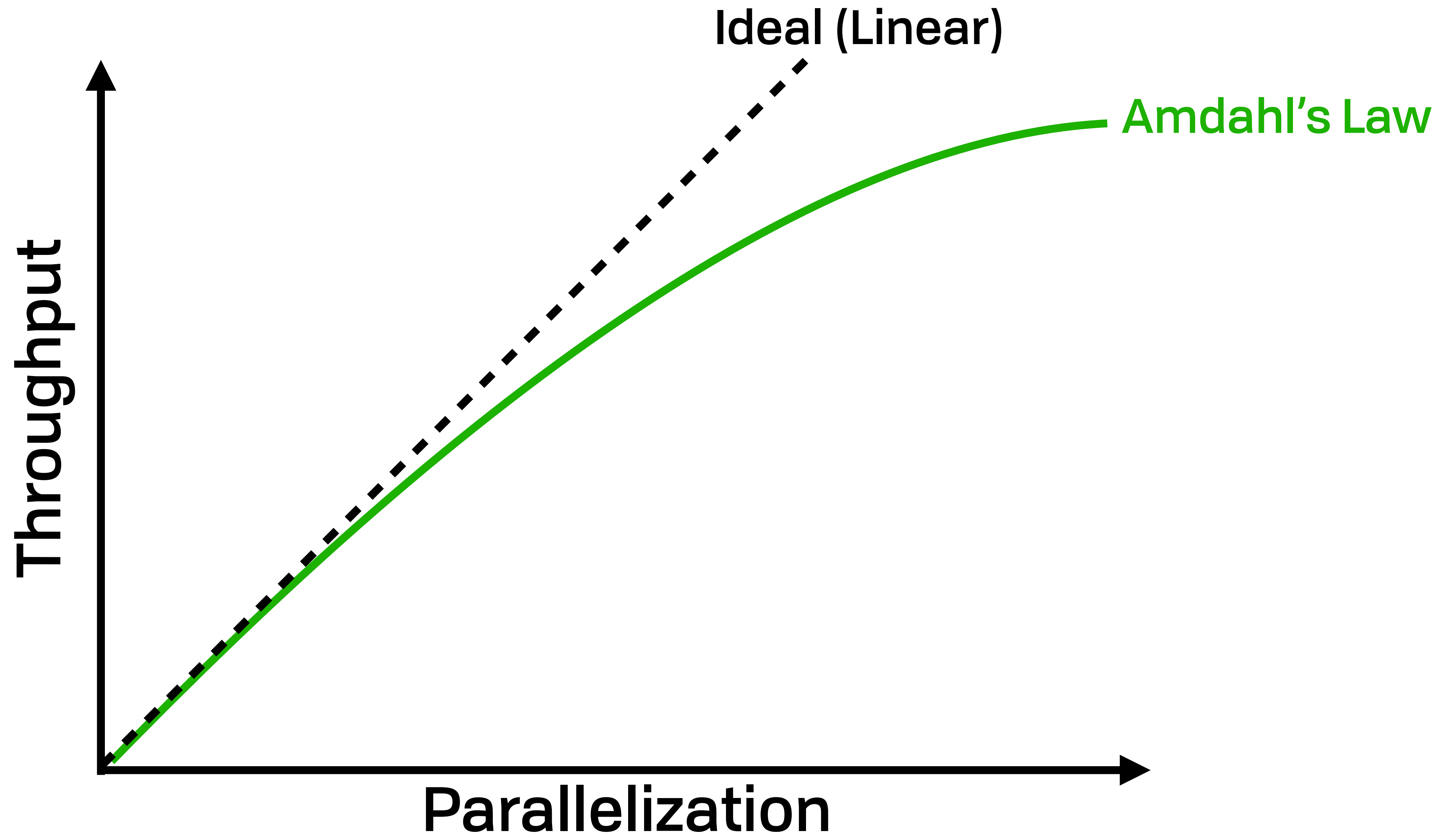
Universal Compute ✨

"With a Little Scale From My Friends"



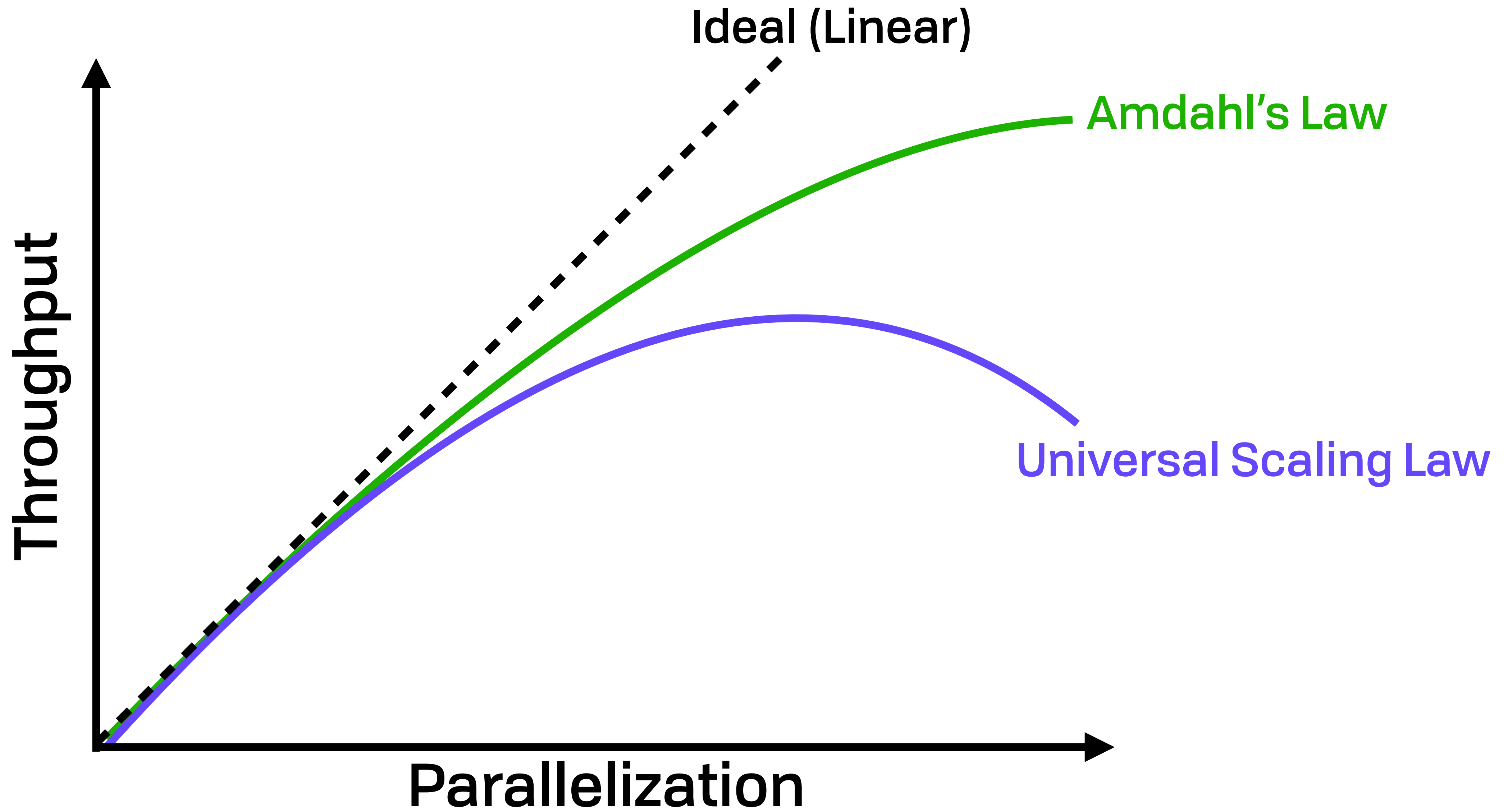
Universal Compute ✨

"With a Little Scale From My Friends"



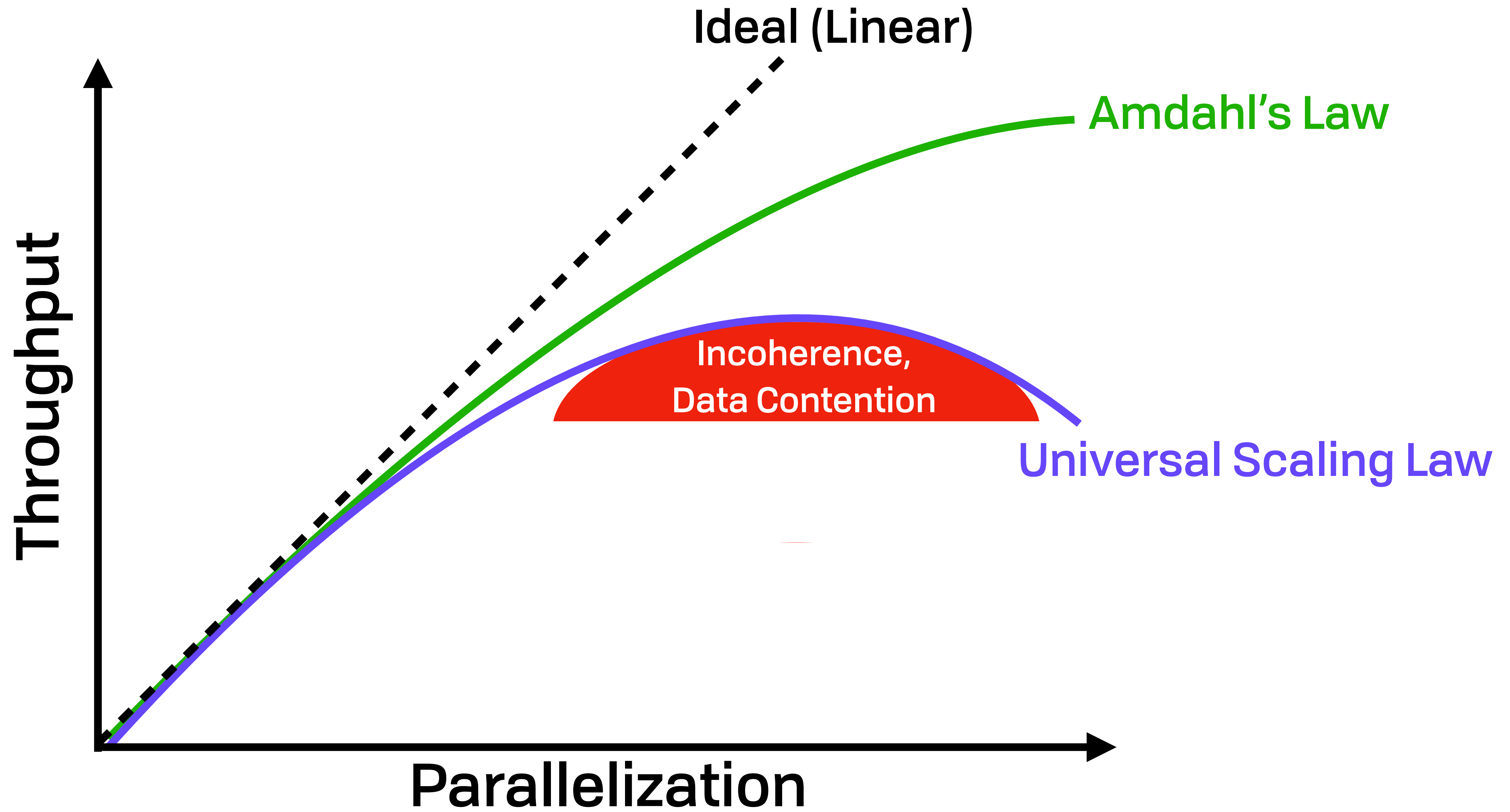
Universal Compute ✨

"With a Little Scale From My Friends"



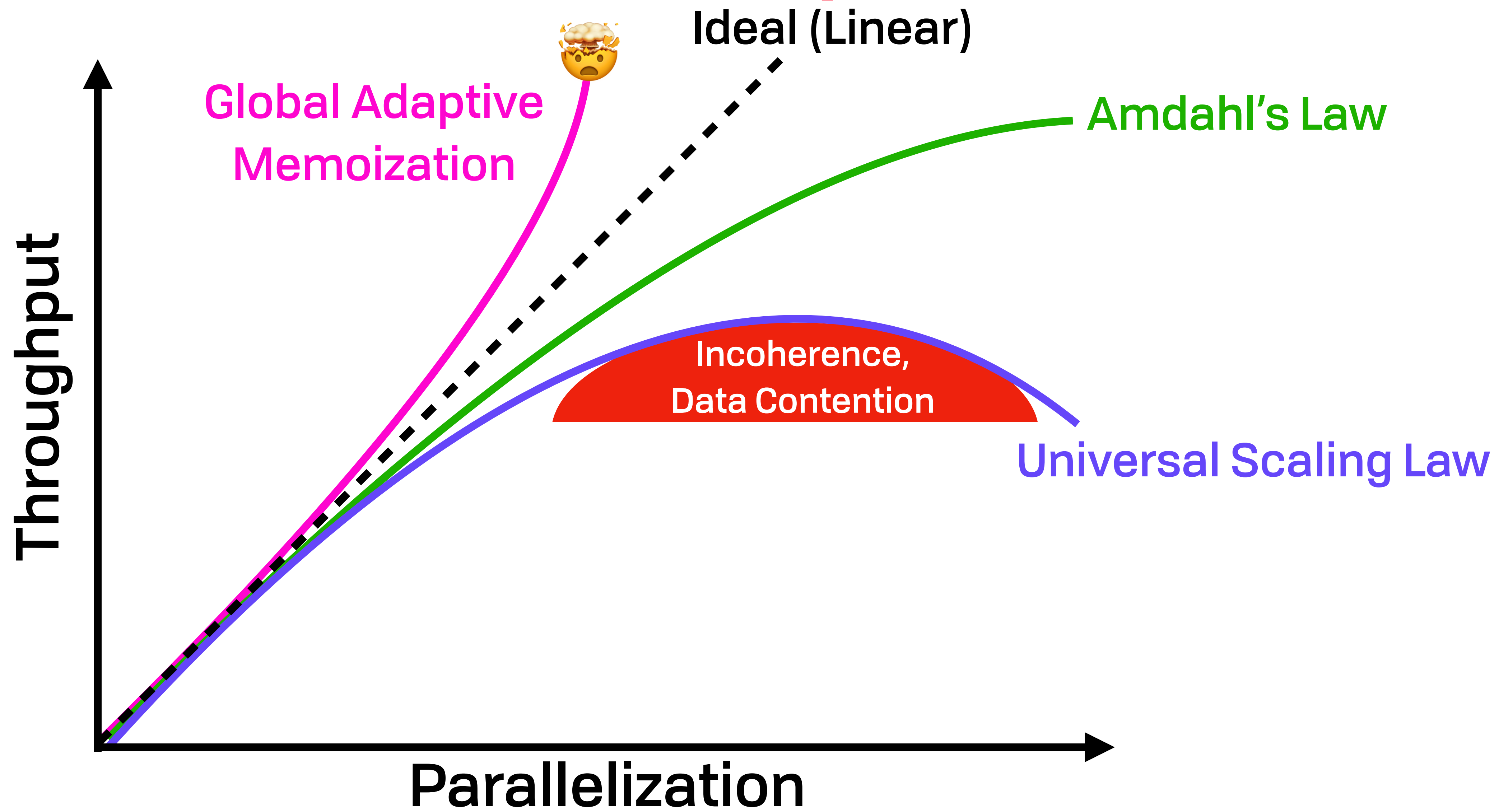
Universal Compute ✨

"With a Little Scale From My Friends"



Universal Compute ✨

"With a Little Scale From My Friends"



Universal Compute ✨

The Compute Commons 🕊️ 📧

Universal Compute ✨

The Compute Commons 🕊️ 📧

Side Effect Stream

Pure Effect Stream

Pure Function Stream

Base Event Stream

Universal Compute ✨

The Compute Commons 🕊️ 📧

Side Effect Stream - - - - -

Pure Effect Stream - - - - -

Pure Function Stream - - - - -

Base Event Stream _____

Universal Compute ✨

The Compute Commons 🕊️ 📧

Side Effect Stream - - - - -

Pure Effect Stream - - - - -

Pure Function Stream - - - - -

Base Event Stream _____

$t \rightarrow$

Universal Compute ✨

The Compute Commons 🕊️ 📧

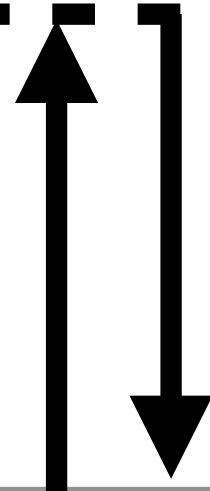
Side Effect Stream - - - - -

Pure Effect Stream - - - - -

Pure Function Stream - - - - -

Base Event Stream _____

$t \rightarrow$



Universal Compute ✨

The Compute Commons 🕊️ 📧

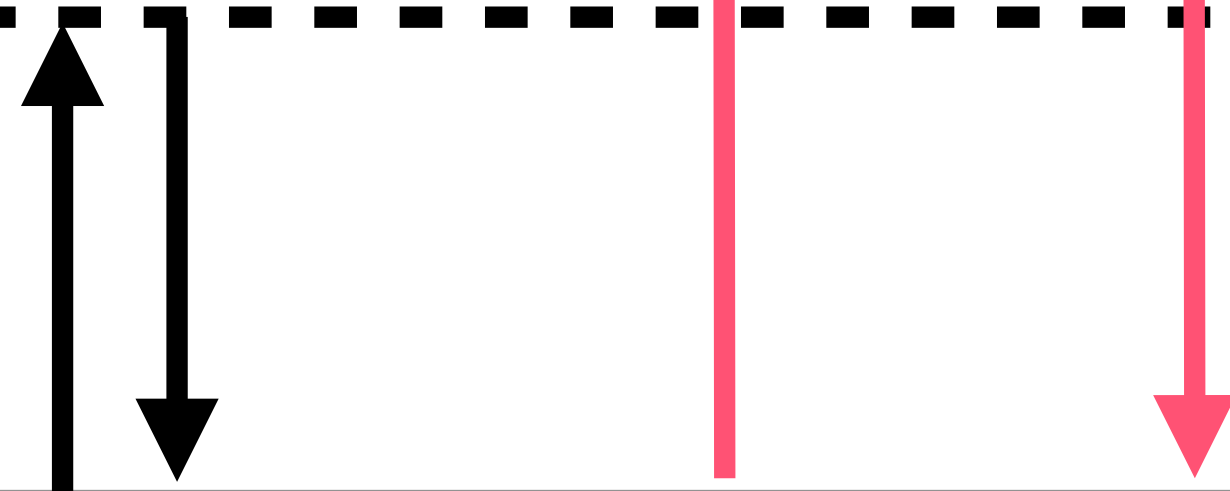
Side Effect Stream - - - - -

Pure Effect Stream - - - - -

Pure Function Stream - - - - -

Base Event Stream _____

$t \rightarrow$



Universal Compute ✨

The Compute Commons 🕊️ 📧

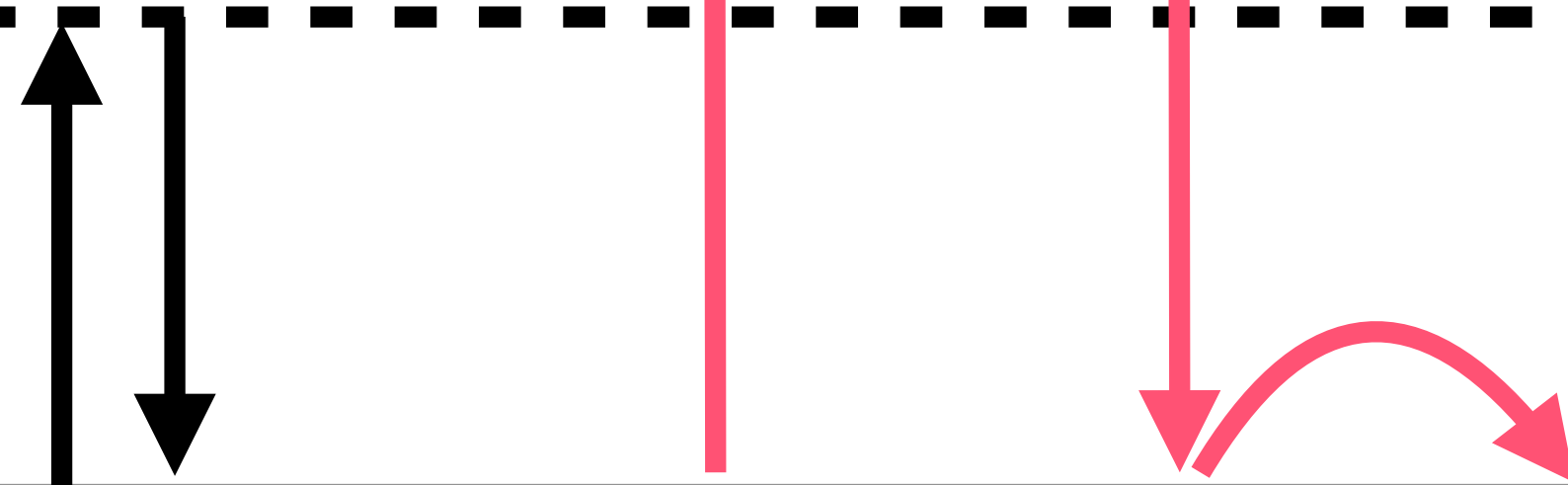
Side Effect Stream - - - - -

Pure Effect Stream - - - - -

Pure Function Stream - - - - -

Base Event Stream _____

$t \rightarrow$



Universal Compute ✨

The Compute Commons 🕊️ 📧

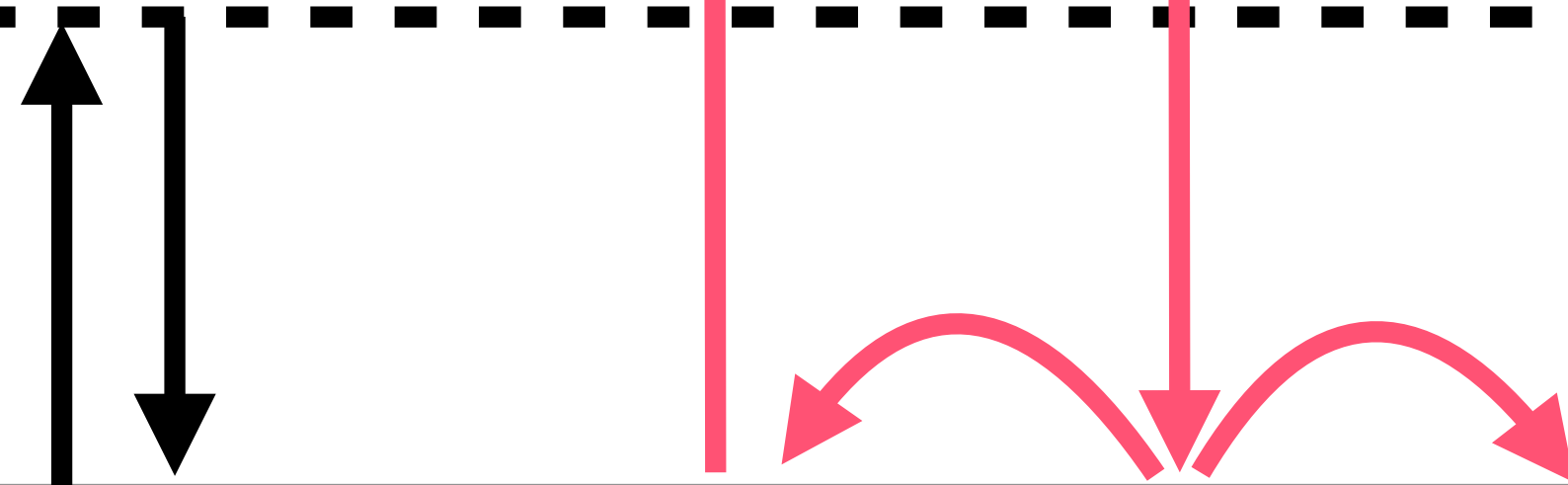
Side Effect Stream -----

Pure Effect Stream -----

Pure Function Stream -----

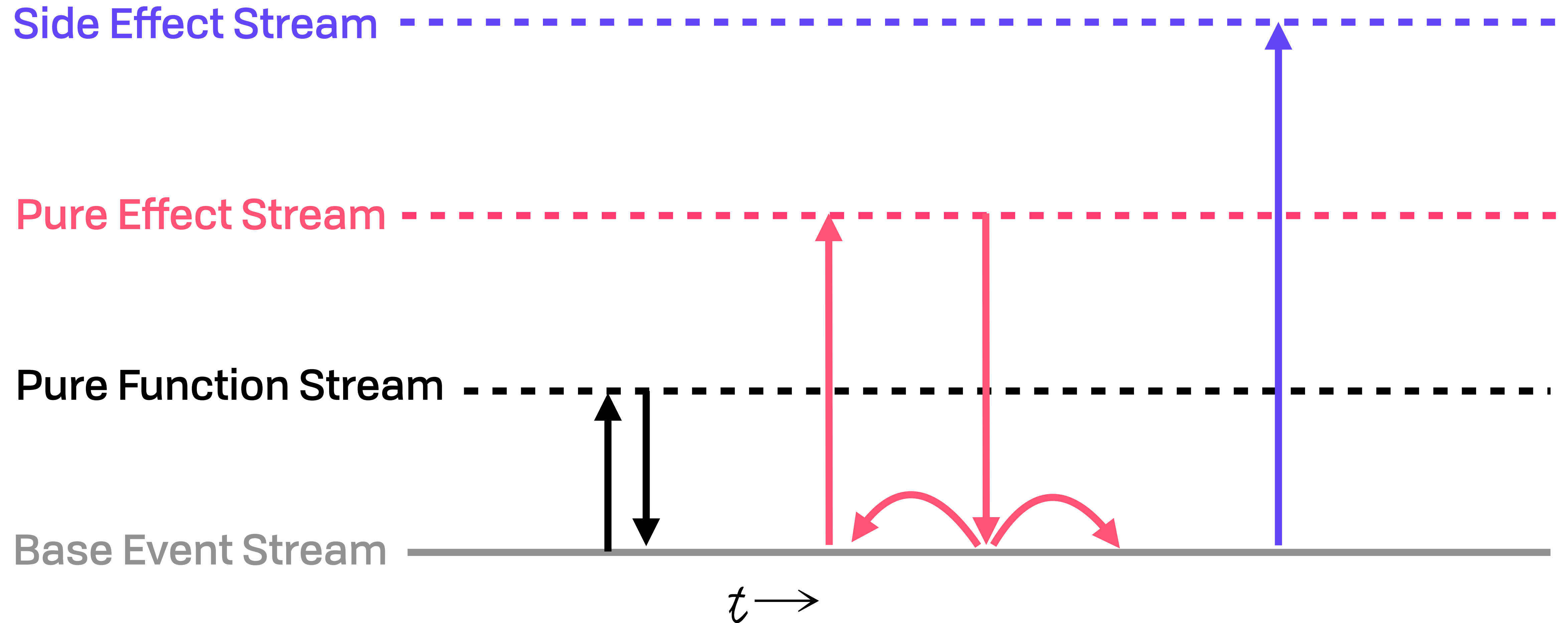
Base Event Stream _____

$t \rightarrow$



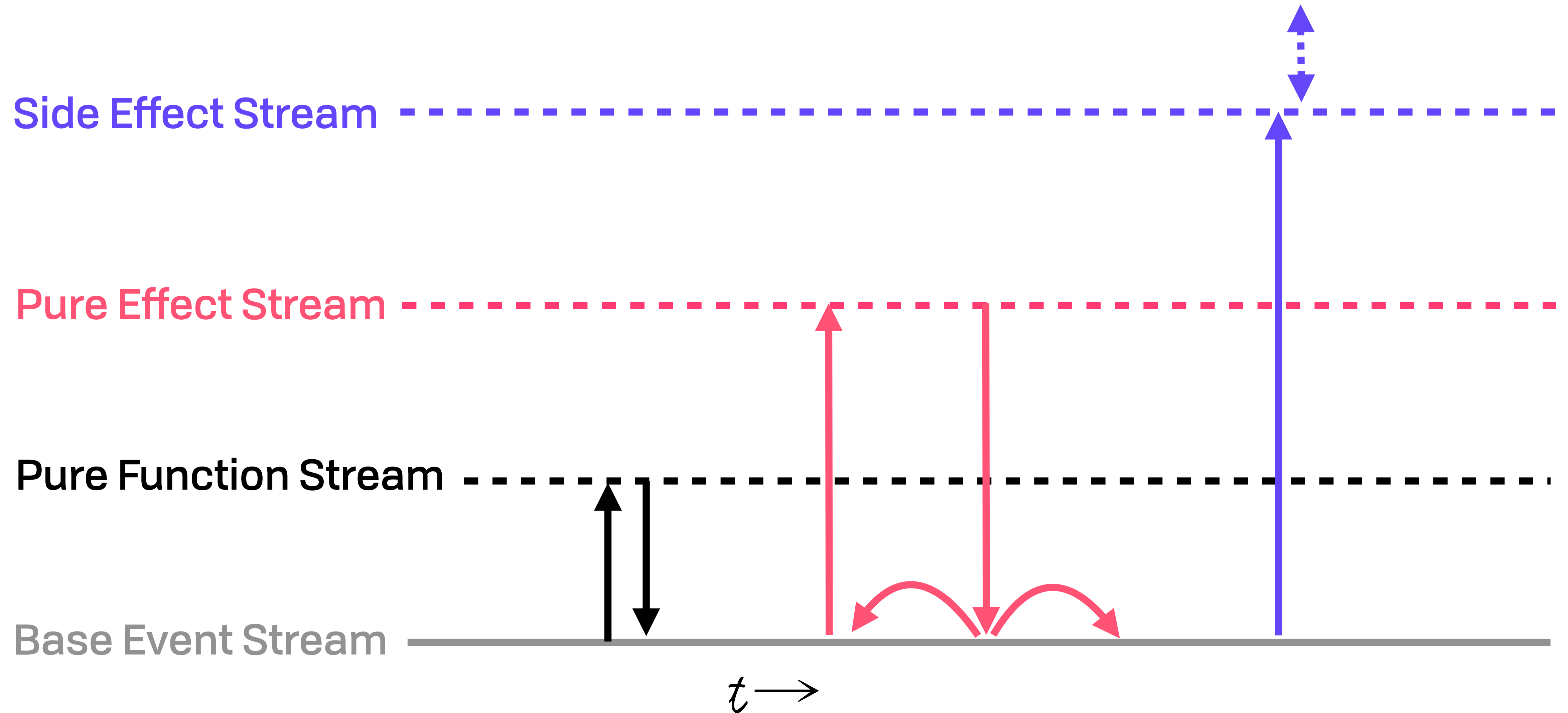
Universal Compute ✨

The Compute Commons 🕊️ 📧



Universal Compute ✨

The Compute Commons 🕊️ 📧



Universal Compute ✨

The Compute Commons 🕊️ 📧

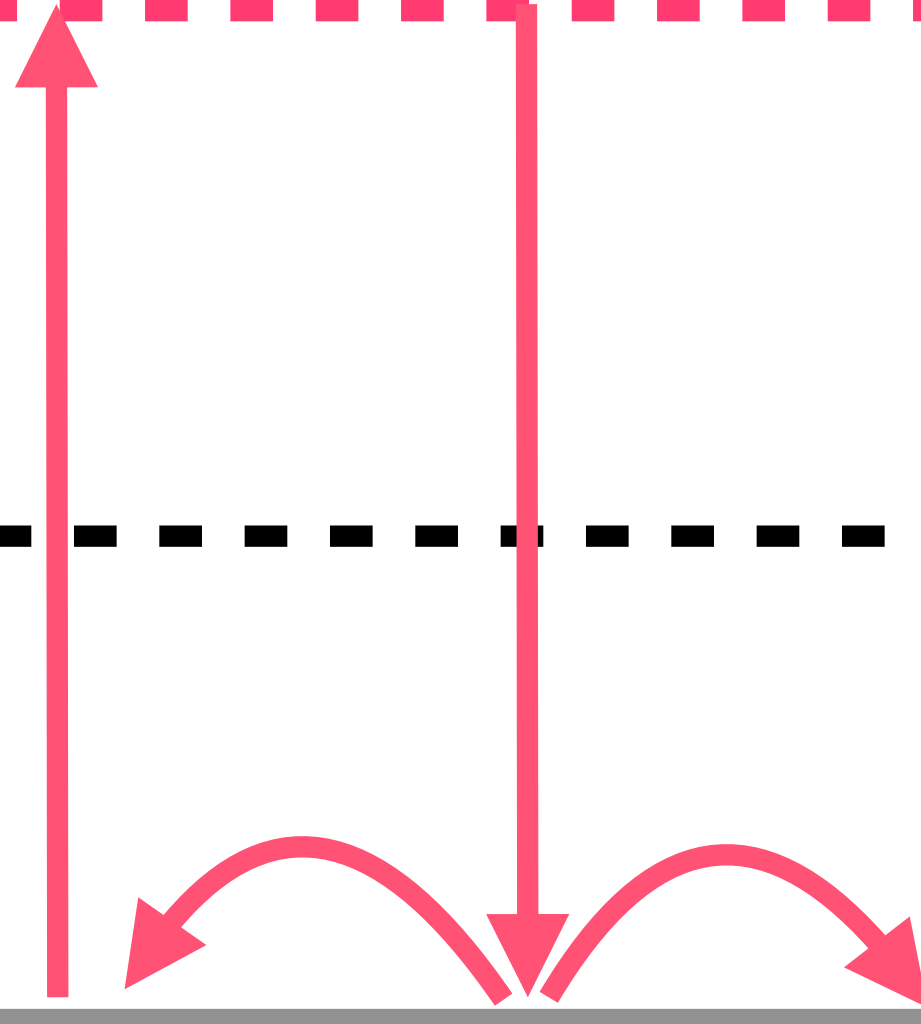
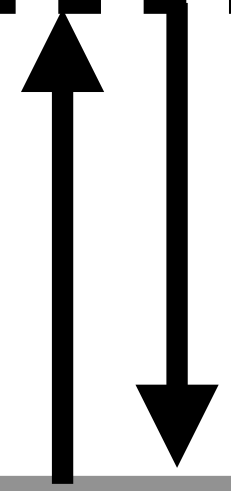
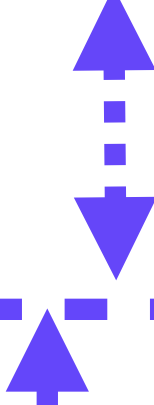
Side Effect Stream

Pure Effect Stream

Pure Function Stream

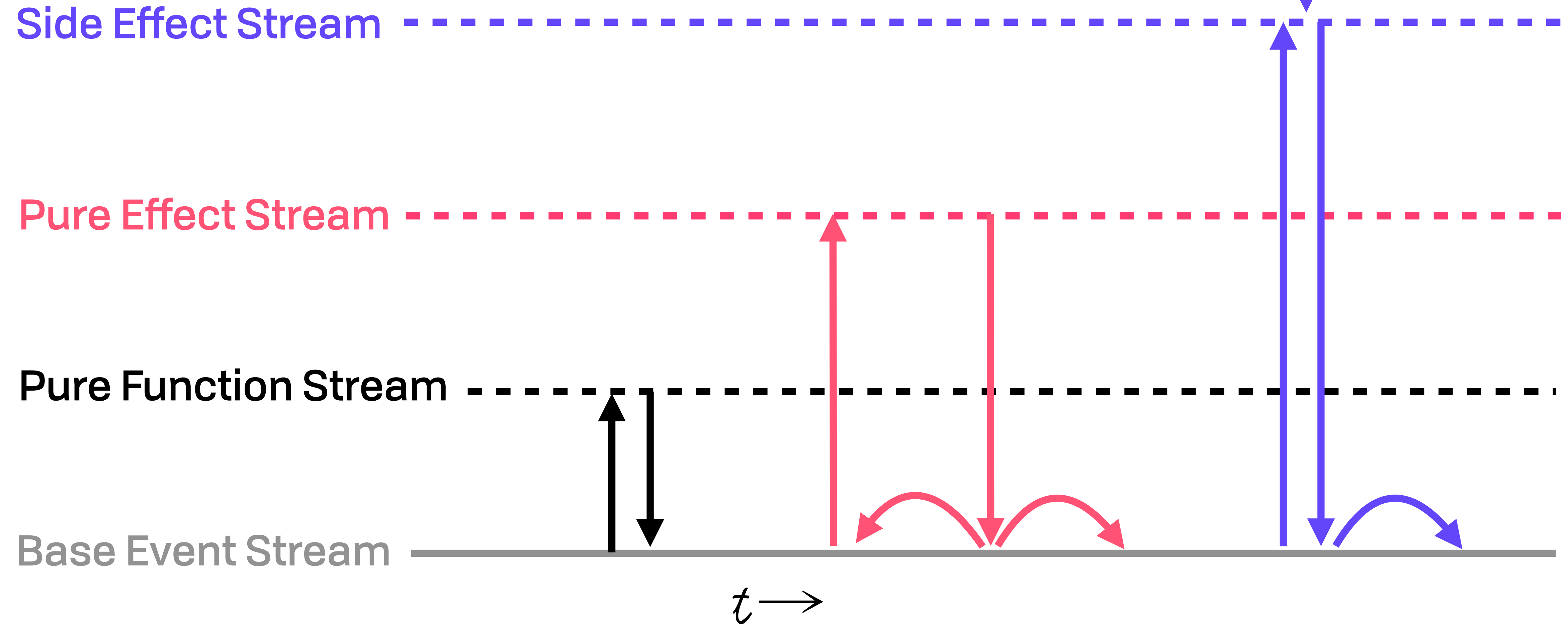
Base Event Stream

$t \rightarrow$



Universal Compute ✨

The Compute Commons 🕊️ 📧



Universal Compute ✨

GenEffect 🚀

```
defmodule Effectful do
  use GenEffect.Runner

  def init(_) do
    bus = EventBus.start_link()
    {:ok, bus}
  end

  def handle_effect({nonce, MyDB, :insert, payload}, _, bus) do
    if EventBus.contains?(nonce) do
      {:ok, :noop, bus}
    else
      case MyDB.insert(payload) do
        :oka          → {:ok, :done, bus}
        {:error, msg} → {:error, msg}
      end
    end
  end

  def handle_external({:run, nonce, msg, credentials}, _, bus) do
    result = SocialMedia.post(msg, credentials)
    # `result` is now treated as pure
    {:ok, result, EventBus.push(stream, {:external, result, credentials})}
  end
end
```


Universal Compute ✨

GenEffect 🚀

```
defmodule Effectful do
  use GenEffect.Runner

  def init(_) do
    bus = EventBus.start_link()
    {:ok, bus}
  end

  def handle_effect({nonce, MyDB, :insert, payload}, _, bus) do
    if EventBus.contains?(nonce) do
      {:ok, :noop, bus}
    else
      case MyDB.insert(payload) do
        :oka          → {:ok, :done, bus}
        {:error, msg} → {:error, msg}
      end
    end
  end

  def handle_external({:run, nonce, msg, credentials}, _, bus) do
    result = SocialMedia.post(msg, credentials)
    # `result` is now treated as pure
    {:ok, result, EventBus.push(stream, {:external, result, credentials})}
  end
end
```

Universal Compute ✨

GenEffect 🚀

```
defmodule Effectful do
  use GenEffect.Runner

  def init(_) do
    bus = EventBus.start_link()
    {:ok, bus}
  end

  def handle_effect({nonce, MyDB, :insert, payload}, _, bus) do
    if EventBus.contains?(nonce) do
      {:ok, :noop, bus}
    else
      case MyDB.insert(payload) do
        :oka          → {:ok, :done, bus}
        {:error, msg} → {:error, msg}
      end
    end
  end

  def handle_external({:run, nonce, msg, credentials}, _, bus) do
    result = SocialMedia.post(msg, credentials)
    # `result` is now treated as pure
    {:ok, result, EventBus.push(stream, {:external, result, credentials})}
  end
end
```

Universal Compute ✨

GenEffect 🚀

```
defmodule Effectful do
  use GenEffect.Runner

  def init(_) do
    bus = EventBus.start_link()
    {:ok, bus}
  end

  def handle_effect({nonce, MyDB, :insert, payload}, _, bus) do
    if EventBus.contains?(nonce) do
      {:ok, :noop, bus}
    else
      case MyDB.insert(payload) do
        :oka          → {:ok, :done, bus}
        {:error, msg} → {:error, msg}
      end
    end
  end

  def handle_external({:run, nonce, msg, credentials}, _, bus) do
    result = SocialMedia.post(msg, credentials)
    # `result` is now treated as pure
    {:ok, result, EventBus.push(stream, {:external, result, credentials})}
  end
end
```

Universal Compute ✨

GenEffect 🚀

```
defmodule Effectful do
  use GenEffect.Runner

  def init(_) do
    bus = EventBus.start_link()
    {:ok, bus}
  end

  def handle_effect({nonce, MyDB, :insert, payload}, _, bus) do
    if EventBus.contains?(nonce) do
      {:ok, :noop, bus}
    else
      case MyDB.insert(payload) do
        :oka          → {:ok, :done, bus}
        {:error, msg} → {:error, msg}
      end
    end
  end





  def handle_external({:run, nonce, msg, credentials}, _, bus) do
    result = SocialMedia.post(msg, credentials)
    # `result` is now treated as pure
    {:ok, result, EventBus.push(stream, {:external, result, credentials})}
  end
end
```

Where Do We Go From Here?



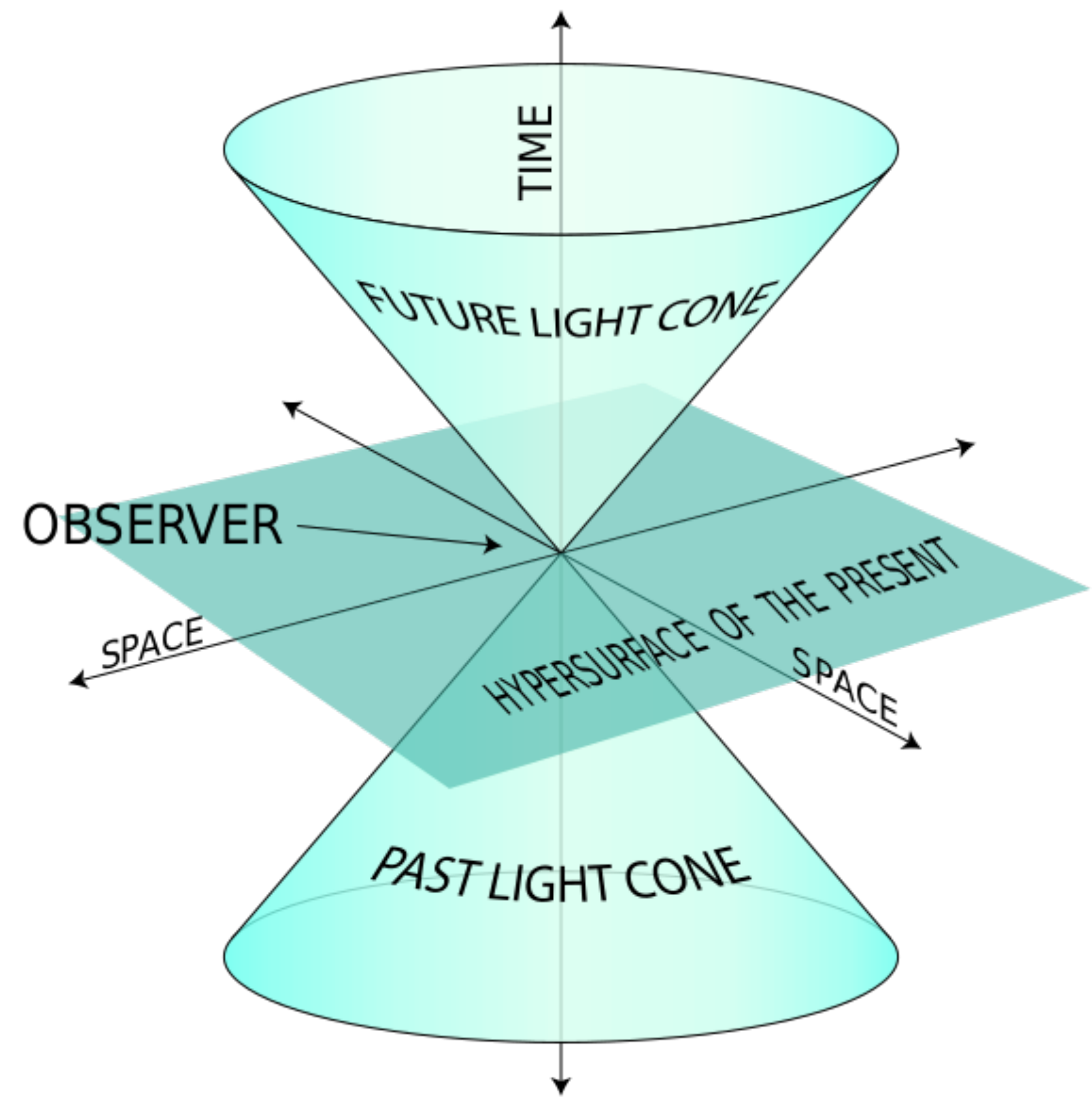
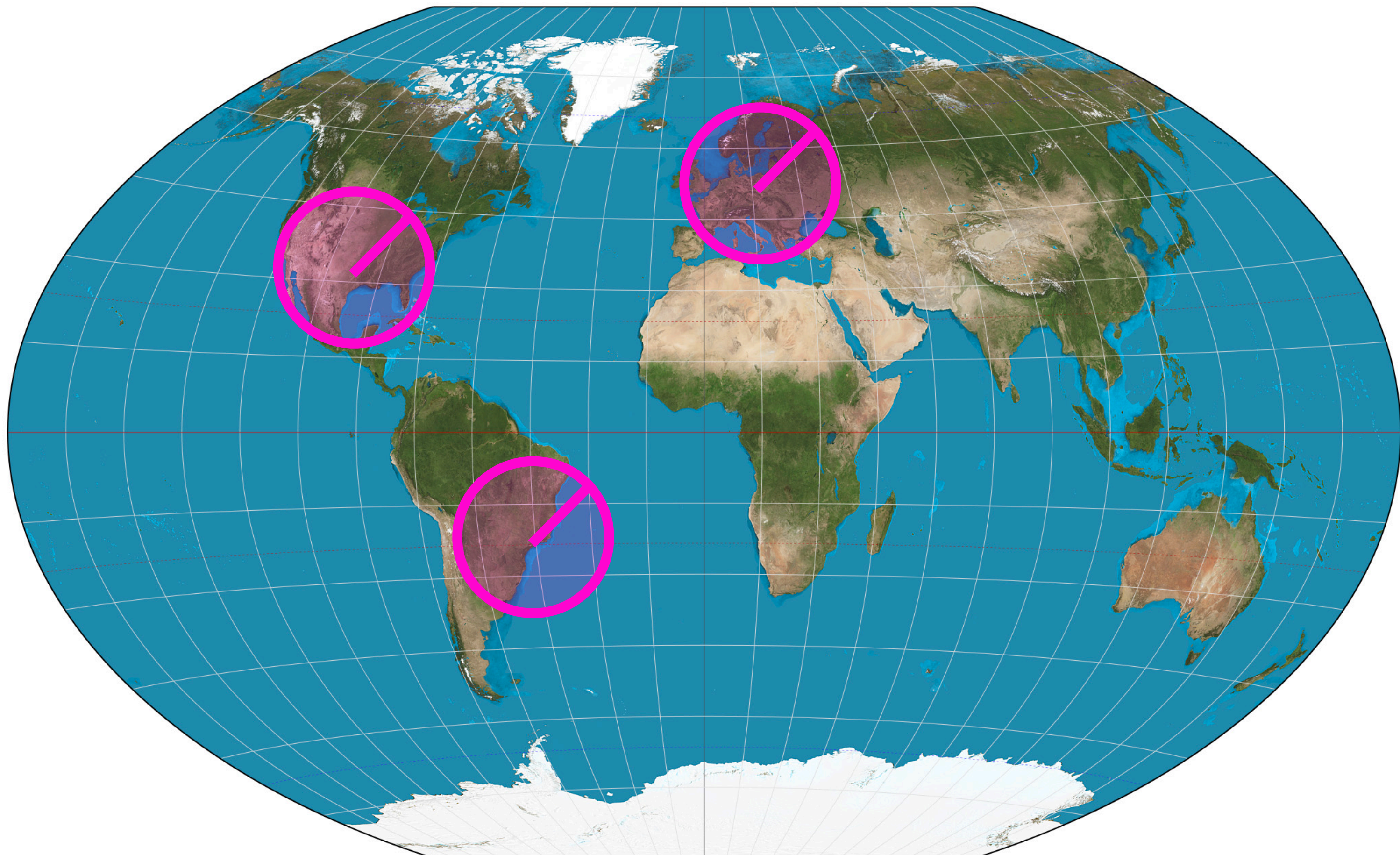
Where Do We Go From Here?

New Environment

| | Then 🖨️ | Now 🚀 |
|------------|---|---|
| Need | Convenient 🙋 | Critical 🚨 |
| Location | Data Centre 🏢 | Powerful Clients (M1, IoT) 🕒 🚗 👟 |
| Access |  |  |
| Bottleneck | Bandwidth 🚚 | Latency 🕒 |
| Market |  |  |

Where Do We Go From Here?

Relativistic Programming



Where Do We Go From Here?

A Better Future on the Edge

Where Do We Go From Here?

A Better Future on the Edge

Embrace the distributed nature of the internet:

Where Do We Go From Here?

A Better Future on the Edge

Embrace the distributed nature of the internet:

1. Only replicate what you need to

Where Do We Go From Here?

A Better Future on the Edge

Embrace the distributed nature of the internet:

1. Only replicate what you need to
2. Remember that data propagates relativistically

Where Do We Go From Here?

A Better Future on the Edge

Embrace the distributed nature of the internet:

1. Only replicate what you need to
2. Remember that data propagates relativistically
3. Free yourself from intrinsic time



Thank You, ElixirConf



brooklyn@fission.codes

<https://fission.codes>

github.com/expede

@expede



Thank You, ElixirConf



brooklyn@fission.codes

<https://fission.codes>

github.com/expede

@expede



Thank You, ElixirConf



brooklyn@fission.codes

<https://fission.codes>

github.com/expede

@expede

It's All About that Data 

Mutable Pointers

- Single-source server/client
 - DNS: hostname → IP address
 - PIDs: number → address
- Focused: **physical** network
- Referential **opacity** (same PID, different data)

```
send( :example@42.123.45.6, :ping )  
  %{node_id => %{path => content}}
```

It's All About that Data 

Mutable Pointers

- Single-source server/client
 - DNS: hostname → IP address
 - PIDs: number → address
- Focused: **physical** network
- Referential **opacity** (same PID, different data)

```
send( :example@42.123.45.6, :ping )  
  %{node_id => %{path => content}}
```

PHYSICAL LOCATION 

It's All About that Data 

Mutable Pointers

- Single-source server/client
 - DNS: hostname → IP address
 - PIDs: number → address
- Focused: **physical** network
- Referential **opacity** (same PID, different data)

```
send( :example@42.123.45.6, :ping )  
  %{node_id => %{path => content}}
```

VIRTUAL ADDRESS 

PHYSICAL LOCATION 

It's All About that Data 

Consistent Keys

`%{hash(content) => content}`

- Above virtual address
- Focused: **data itself**
 - Same for **everyone & everywhere**
 - Perfect for caching
- Immutable data++
 - Consistent pointers → consistent data

It's All About that Data 

Consistent Keys

- Above virtual address
- Focused: **data itself**
 - Same for **everyone & everywhere**
 - Perfect for caching
- Immutable data++
 - Consistent pointers → consistent data

$\%{\text{hash}(\text{content})} \Rightarrow \text{content}$

CONTENT ID 

PHYSICAL LOCATION 

It's All About that Data 

Consistent Keys

- Above virtual address
- Focused: **data itself**
 - Same for **everyone & everywhere**
 - Perfect for caching
- Immutable data++
 - Consistent pointers → consistent data

$\%{\text{hash}(\text{content})} \Rightarrow \text{content}$

CONTENT ID 

VIRTUAL ADDRESS 

PHYSICAL LOCATION 

It's All About that Data 

Hash-Based Relationships

It's All About that Data 

Hash-Based Relationships

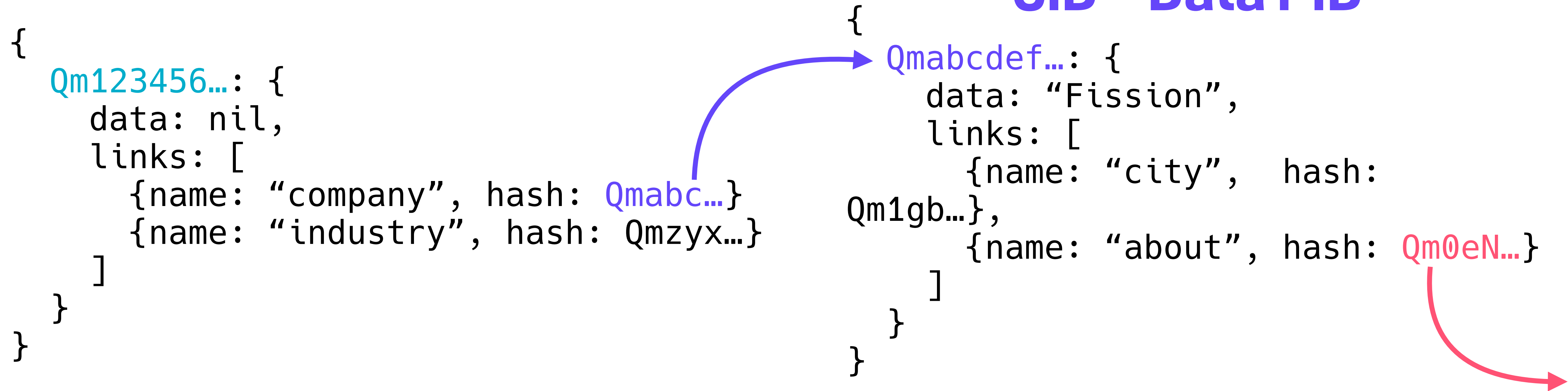
CID ~ Data PID

```
{
  Qm123456...: {
    data: nil,
    links: [
      {name: "company", hash: Qmabc...}
      {name: "industry", hash: Qmzyx...}
    ]
  }
}
```


It's All About that Data 📊

Hash-Based Relationships

CID ~ Data PID



It's All About that Data 📊

Hash-Based Relationships

CID ~ Data PID

```
{
  Qm123456...: {
    data: nil,
    links: [
      {name: "company", hash: Qmabc...}
      {name: "industry", hash: Qmzyx...}
    ]
  }
}
```

```
{
  Qmabcdef...: {
    data: "Fission",
    links: [
      {name: "city", hash: Qm1gb...},
      {name: "about", hash: Qm0eN...}
    ]
  }
}
```

Qm123456.../company/about/ceo
=> "Boris Mann"

It's All About that Data 

Content IDs Are Easy

```
defmodule ContentAddressed.Store do
  defstruct store: %{}

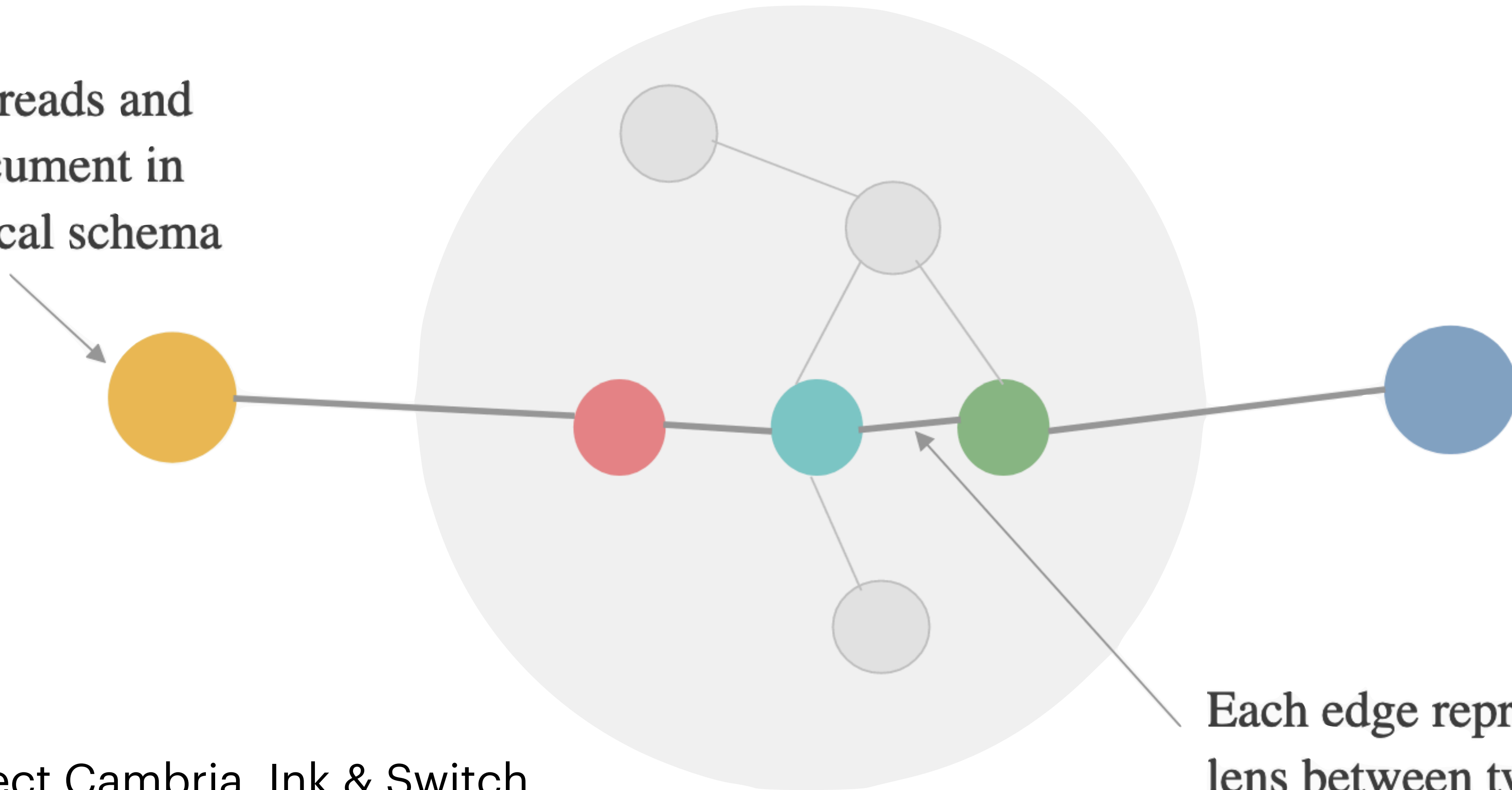
  def get(%Store{store: store}, cid), do: Map.get(store, cid)

  def set(%Store{store: store}, data) do
    case ExCrypto.sha256(binary) do
      {:ok, cid} -> {:ok, %Store{store: Map.put(store, cid, binary)}}
      {:error, err} -> {:error, err}
    end
  end
end
```


Decentralized Systems 🌈

Different Clients ~ Schema Drift 🔍

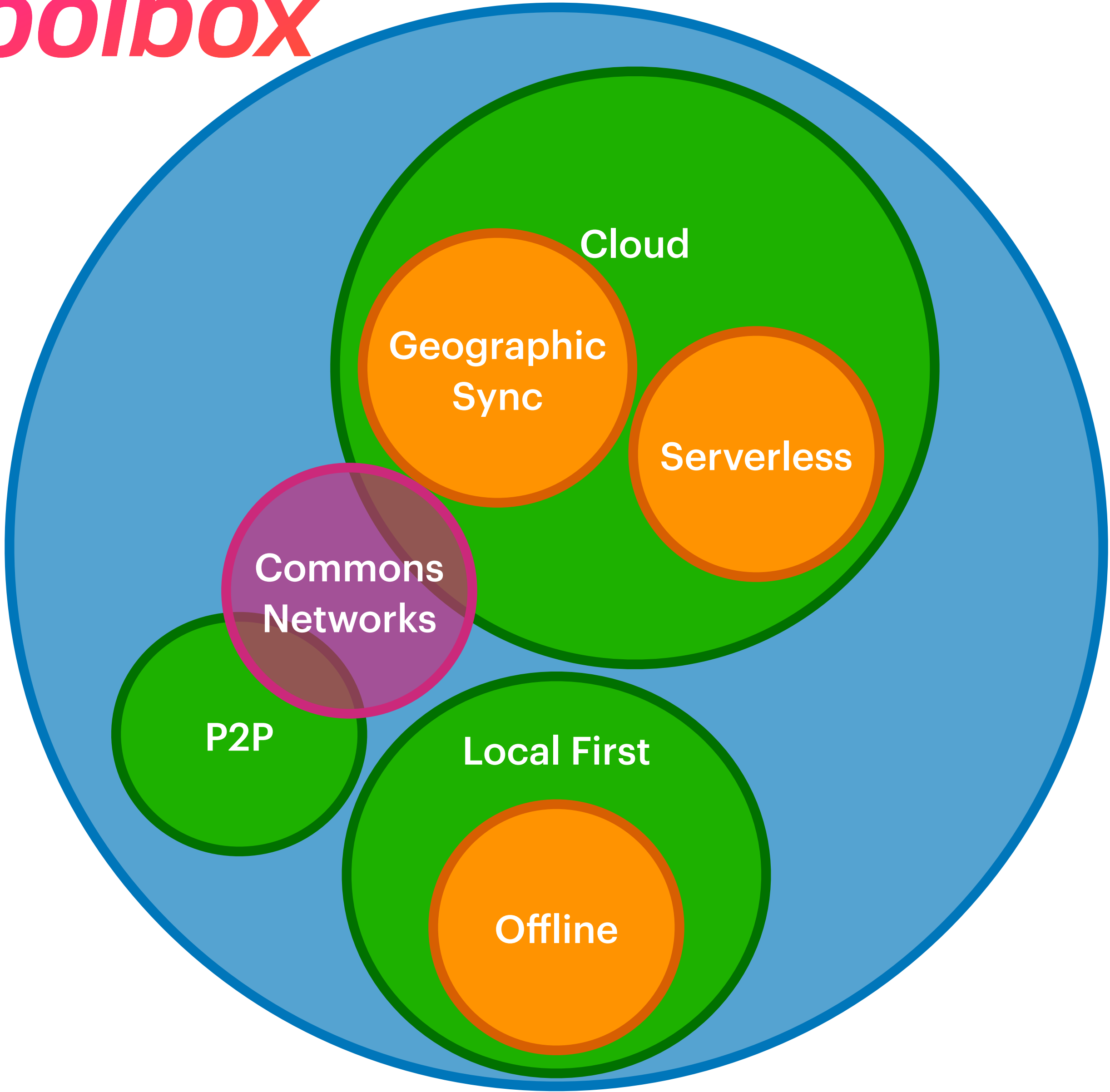
Each client reads and writes a document in its native local schema



Each edge represents a lens between two schemas


High Volume 

Evolving Toolbox



On the Edge 

Why Elixir?

- Data-oriented, immutable
- Code-as-data  data-as-code
- Shared-nothing architectures
- Easy concurrency, distributed systems
- Actor model → OCAP
- Build up complexity from simple parts

We have a system that applies **cutting edge** CS research to **tackle day-to-day problems** in the applications we all write.

Phoenix Presence

- has **no single point of failure**
- has **no single source of truth**
- [...]
- **self heals**

~ Chris McCord, "What Makes Phoenix Presence Special"

Low Latency 🐰

Earth is Just Too Big! 🌍

- Ideal 40ms one-way
- SF ➡ Austin, Vancouver, London, Tokyo
- SF ❌ Cape Town, Sidney, Rio de Janeiro



Credit: Keenan Crane

<http://www.cs.cmu.edu/~kmcrane/Projects/ModelRepository/>

Low Latency 🐰

Earth is Just Too Big! 🌍

- Ideal 40ms one-way
- SF ➡ Austin, Vancouver, London, Tokyo
- SF ❌ Cape Town, Sidney, Rio de Janeiro



Credit: Keenan Crane

<http://www.cs.cmu.edu/~kmcrane/Projects/ModelRepository/>

What does this all mean?

Consequence



Consequence 

Primary Progression

Consequence 🛸

Primary Progression

1. **Embrace** the distributed nature of the internet 🤗

Consequence 🛸

Primary Progression

1. **Embrace** the distributed nature of the internet 🤗
2. Universal IDs (without coordination)

Consequence 🛸

Primary Progression

1. **Embrace** the distributed nature of the internet 🤗
2. Universal IDs (without coordination)
3. Direct access control

Consequence 🛸

Primary Progression

1. **Embrace** the distributed nature of the internet 🤗
2. Universal IDs (without coordination)
3. Direct access control
4. Cache friendliness

Consequence 🛸

Primary Progression

1. **Embrace** the distributed nature of the internet 🤗
2. Universal IDs (without coordination)
3. Direct access control
4. Cache friendliness
5. Relativistic databases

Consequence 🛸

Primary Progression

1. **Embrace** the distributed nature of the internet 🤗
2. Universal IDs (without coordination)
3. Direct access control
4. Cache friendliness
5. Relativistic databases
6. Interoperable formats (across time & team)

Consequence 🛸

Primary Progression

1. **Embrace** the distributed nature of the internet 🤗
2. Universal IDs (without coordination)
3. Direct access control
4. Cache friendliness
5. Relativistic databases
6. Interoperable formats (across time & team)
7. Efficient replication

It's All About that Data 

Minimal CRDT From Smaller Blocks

It's All About that Data 

Minimal CRDT From Smaller Blocks

```
defprotocol AbelianMonoid do
  def empty(a)
  def append(a, b)
  def order(a, b)
end
```

It's All About that Data 

Minimal CRDT From Smaller Blocks

```
defprotocol AbelianMonoid do
  def empty(a)
  def append(a, b)
  def order(a, b)
end
```

```
defimpl AbelianMonoid, for: Integer do
  def empty(_), do: 0
  def append(a, b), do: a + b
  def order(a, b)
  cond
    a == b -> :eq
    a < b -> :lt
    a > b -> :gt
  end
end
```

It's All About that Data 

Minimal CRDT From Smaller Blocks

```
defprotocol AbelianMonoid do
  def empty(a)
  def append(a, b)
  def order(a, b)
end
```

```
defimpl AbelianMonoid, for: Integer do
  def empty(_), do: 0
  def append(a, b), do: a + b
  def order(a, b)
  cond
    a == b -> :eq
    a < b -> :lt
    a > b -> :gt
  end
end
```

```
defimpl AbelianMonoid, for: List do
  def empty(_), do: 0
  def append(xs, ys), do: Enum.sort(xs ++ ys)
  def order(xs, ys) do
    xs_set = MapSet.new(xs)
    ys_set = MapSet.new(ys)

    cond do
      xs == ys -> :eq
      MapSet.subset?(xs_set, ys_set) -> :gt
      MapSet.subset?(ys_set, xs_set) -> :lt
      _ -> :incomparable
    end
  end
end
```

It's All About that Data 

Minimal CRDT From Smaller Blocks

```
defprotocol AbelianMonoid do
  def empty(a)
  def append(a, b)
  def order(a, b)
end
```

```
defimpl AbelianMonoid, for: Integer do
  def empty(_), do: 0
  def append(a, b), do: a + b
  def order(a, b)
  cond
    a == b -> :eq
    a < b -> :lt
    a > b -> :gt
  end
end
```

```
defimpl AbelianMonoid, for: List do
  def empty(_), do: 0
  def append(xs, ys), do: Enum.sort(xs ++ ys)
  def order(xs, ys) do
    xs_set = MapSet.new(xs)
    ys_set = MapSet.new(ys)

    cond do
      xs == ys -> :eq
      MapSet.subset?(xs_set, ys_set) -> :gt
      MapSet.subset?(ys_set, xs_set) -> :lt
      _ -> :incomparable Sibling / Concurrent
    end
  end
end
```


Getting Ready 

Data > Compute

- Pure data-focus frees you!
- Clarify “real” dependencies on data
- Direct auth, OCAP

Motivation 🎭

What Even Is a "Server"?

1. Multi-tenant auth gatekeeper
2. Uptime / resource availability
3. Out-of-band compute (e.g. batch tasks, cron, OLAP)

Low Latency 🐰

Can't Go Faster, Make Shorter Trips

Low Latency 🐰

Can't Go Faster, Make Shorter Trips

1950s

Silicon
Transistor



1

Transistor

1960s

TTL
Quad Gate

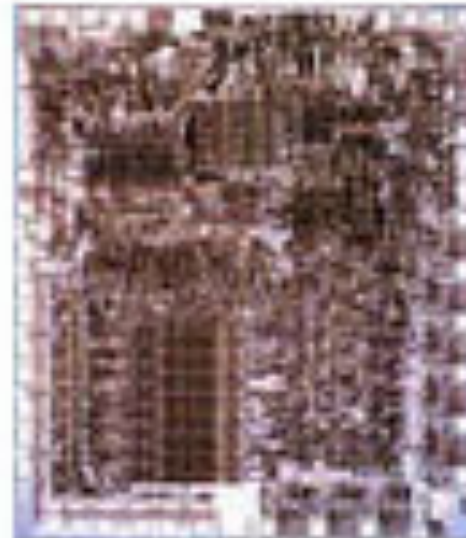


16

Transistors

1970s

8-bit
Microprocessor

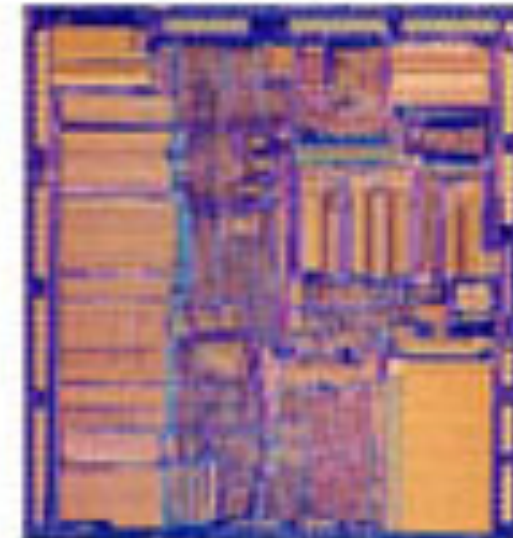


4500

Transistors

1980s

32-bit
Microprocessor

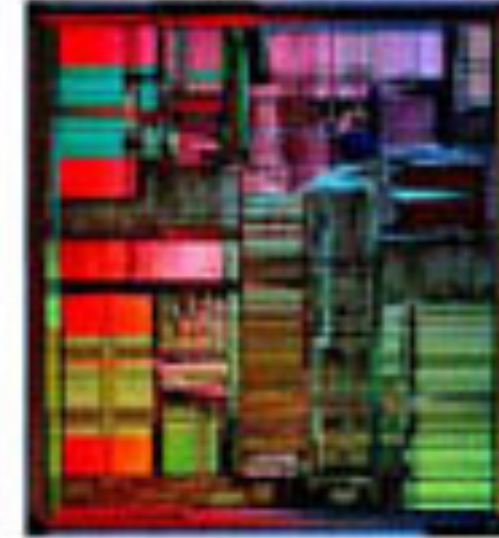


275,000

Transistors

1990s

32-bit
Microprocessor

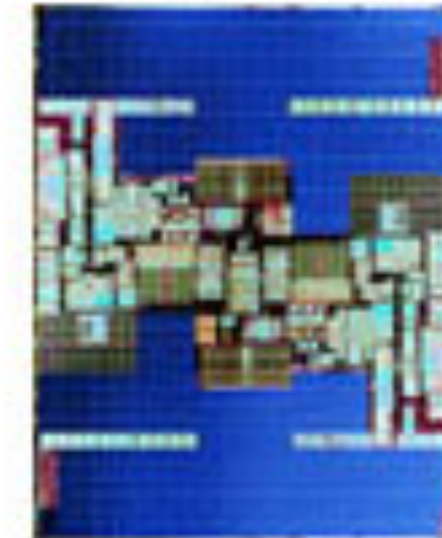


3,100,000

Transistors

2000s

64-bit
Microprocessor

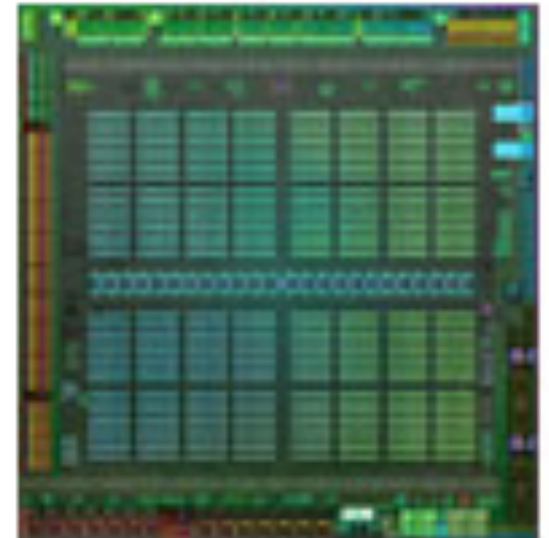


592,000,000

Transistors

2010s

3072-Core
GPU



8,000,000,000

Transistors

Source: Computer History Museum

Low Latency 🐰

Can't Go Faster, Make Shorter Trips

>10 μ m
>10,000nm

1950s

Silicon
Transistor



1

Transistor

1960s

TTL
Quad Gate

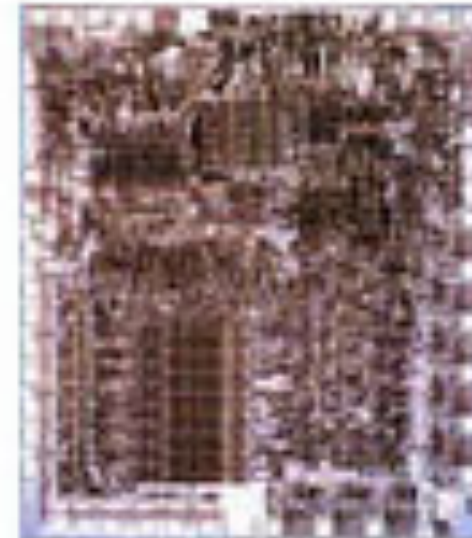


16

Transistors

1970s

8-bit
Microprocessor

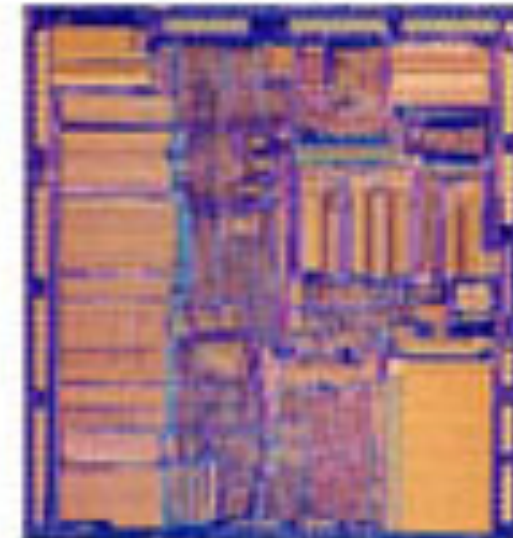


4500

Transistors

1980s

32-bit
Microprocessor

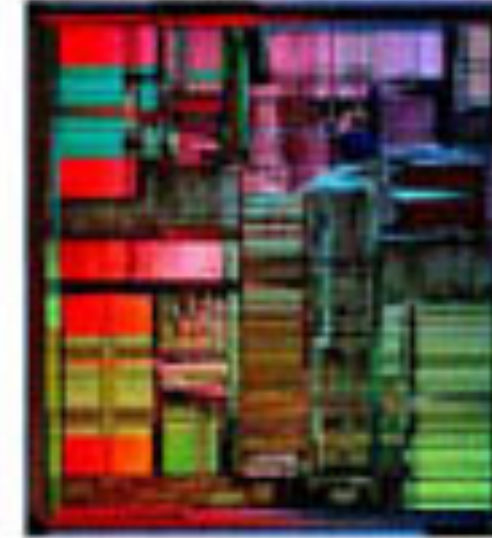


275,000

Transistors

1990s

32-bit
Microprocessor

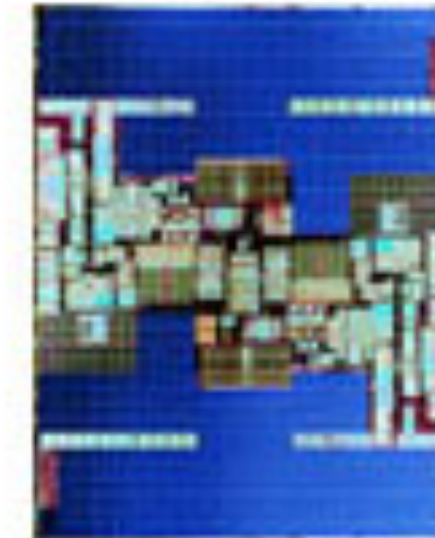


3,100,000

Transistors

2000s

64-bit
Microprocessor

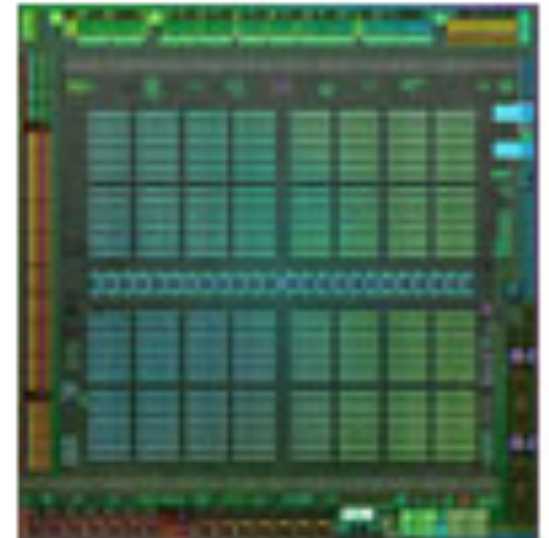


592,000,000

Transistors

2010s

3072-Core
GPU



8,000,000,000

Transistors

Source: Computer History Museum

Low Latency 🐰

Can't Go Faster, Make Shorter Trips

>10 μ m
>10,000nm **10nm**

1950s

Silicon
Transistor



1

Transistor

1960s

TTL
Quad Gate

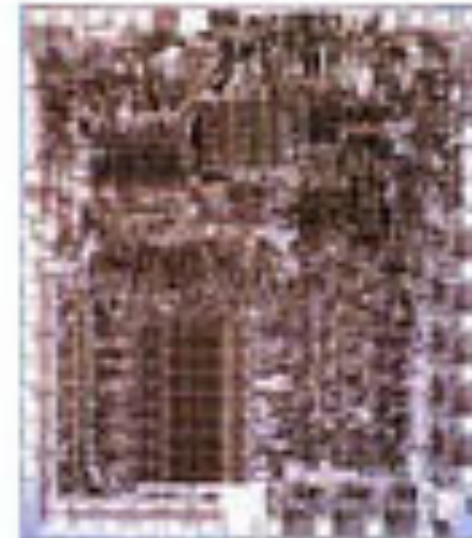


16

Transistors

1970s

8-bit
Microprocessor

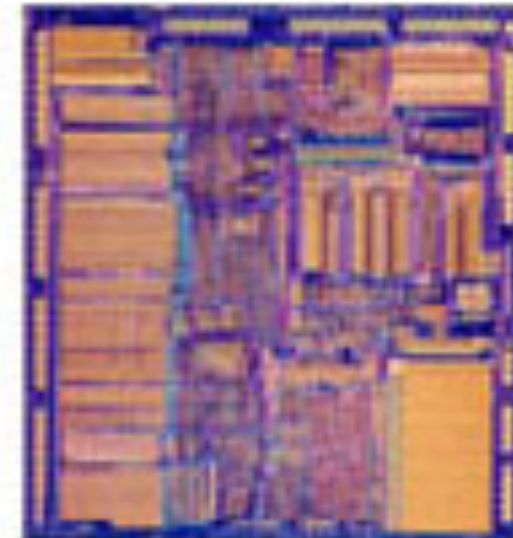


4500

Transistors

1980s

32-bit
Microprocessor

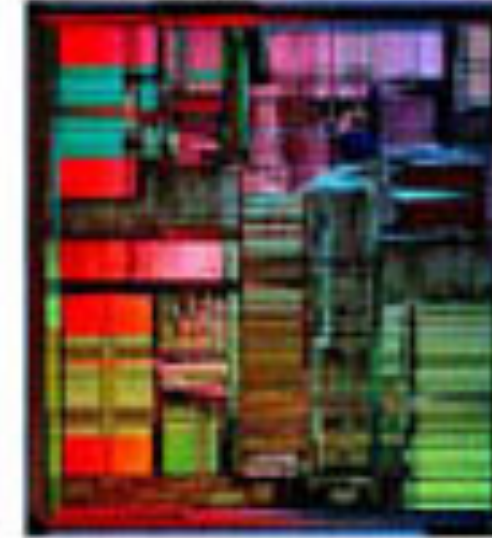


275,000

Transistors

1990s

32-bit
Microprocessor

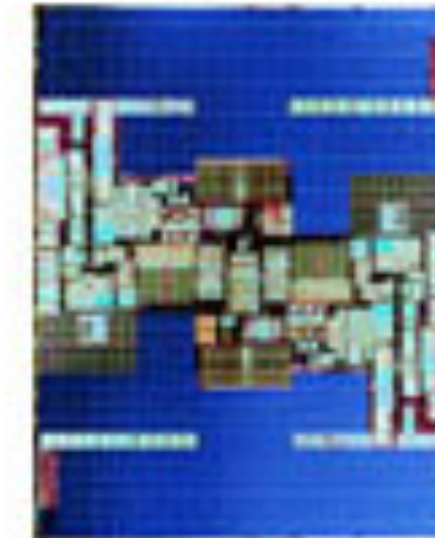


3,100,000

Transistors

2000s

64-bit
Microprocessor

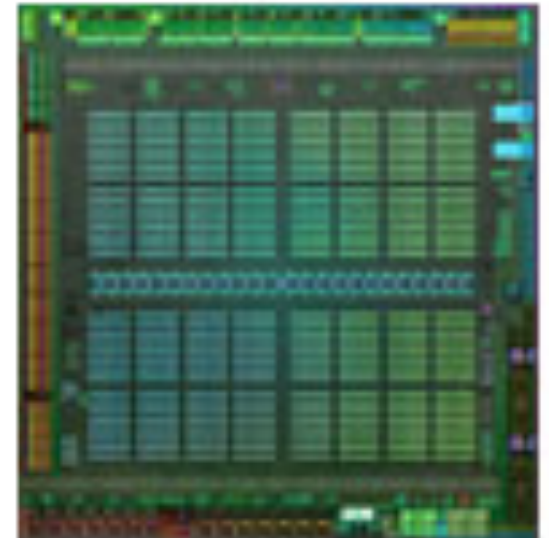


592,000,000

Transistors

2010s

3072-Core
GPU



8,000,000,000

Transistors

Source: Computer History Museum