Living On the Edge

🌐⚡A Brave New (Post-Cloud) World 🛰️✨

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

~ IBM (paraphrasing a Gartner study)

Brooklyn Zelenka

@expede

Brooklyn Zelenka

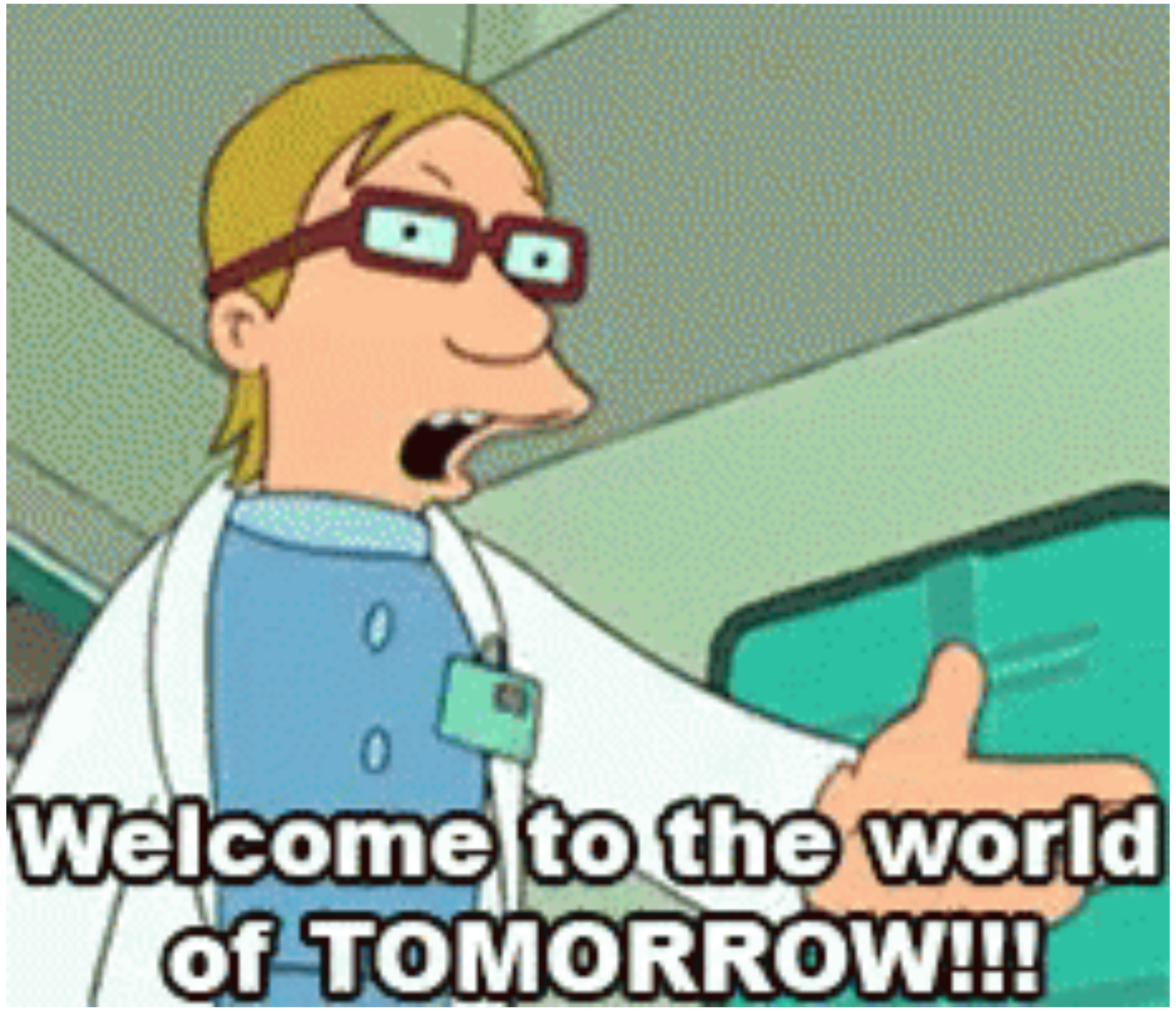
@expede

- CTO at Fission
  - https://fission.codes
  - Infra & browser SDK for edge apps
- PLT, distributed systems
- Specs: DIF, ETH Core
- Meetups: Vancouver FP, Code & Coffee YVR
- Libs: Witchcraft, Exceptional, Rescue, &c

WebNative 🚀

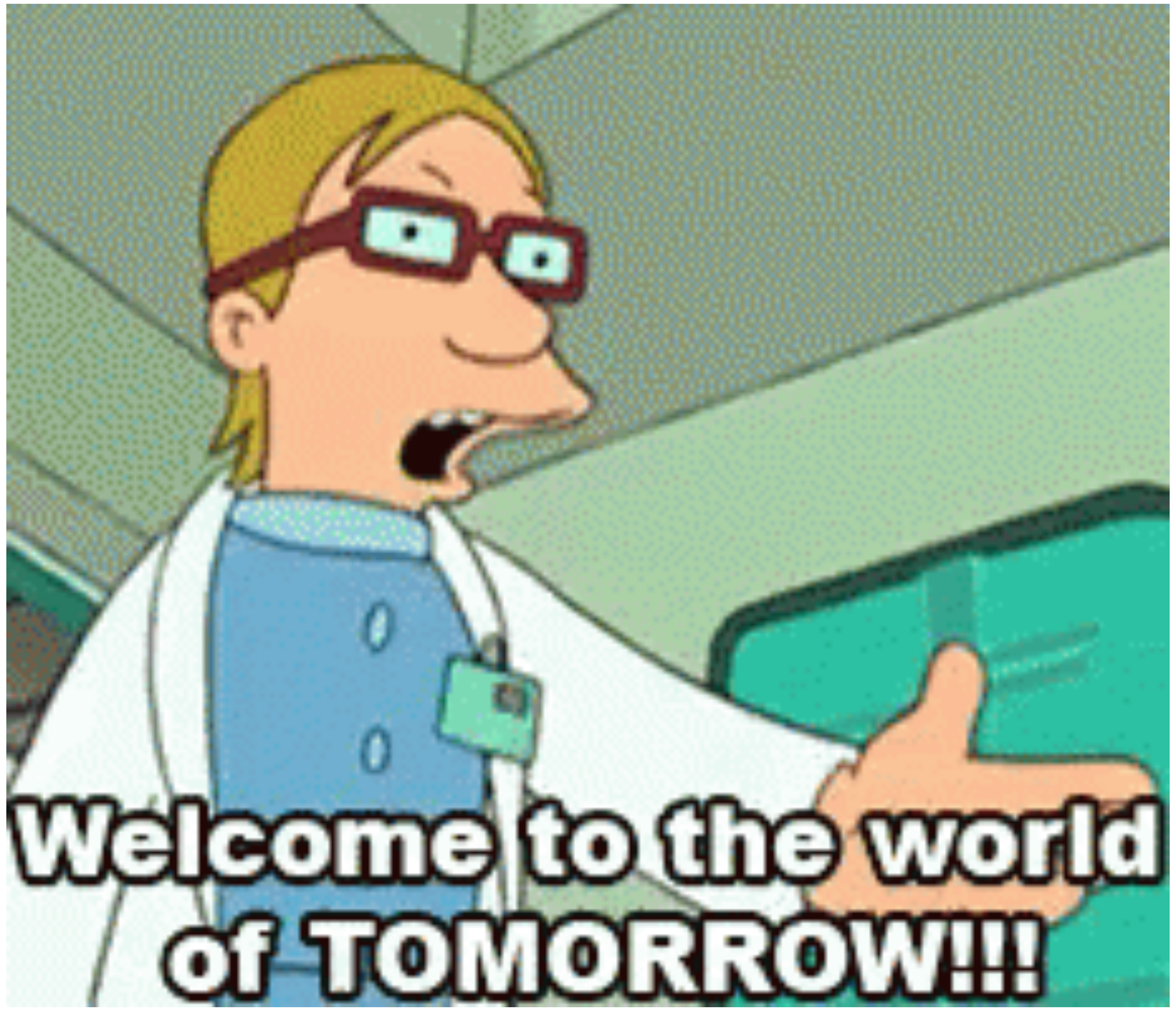
Meta 🔮

- R&D from Fission & others
- Future looking / an emerging area
- Interesting tech, very exciting
  - …but not all problems solved today
- Some advantages to flexible tech even before the network changes
- Universal Hostless Substrate (2019)



WebNative 🚀

Meta 🔮



- R&D from Fission & others
- Future looking / an emerging area
- Interesting tech, very exciting
  - …but not all problems solved today
- Some advantages to flexible tech even before the network changes
- Universal Hostless Substrate (2019)

WebNative 🚀

Fission R&D ⌘ @FISSIONCodes

- Local first
- Edge only
- No servers
- Fully distributed
- Encrypted at Rest, E2EE
- User owned data

WebNative 🚀

Overview

Part I: Motivation

How we got here

What changed?

Part II: On the Edge

Why BEAM

Primer

All About Data

A Few Techniques

Part I

Motivation

🎭

Motivation 🎭

90s Web

Motivation 🎭

90s Web

💁‍♀️ 🖥️

Motivation 🎭

90s Web

Motivation 🎭

90s Web

💁‍♀️ 🖥️🐢 🗃️

Motivation 🎭

90s Web

Motivation 🎭

90s Web

Motivation 🎭

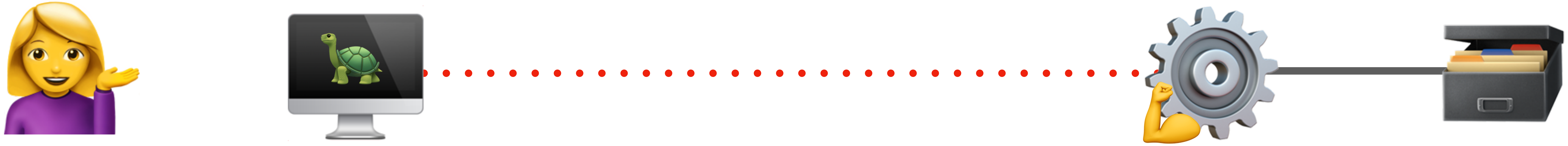
90s Web

Motivation 🎭

90s Web

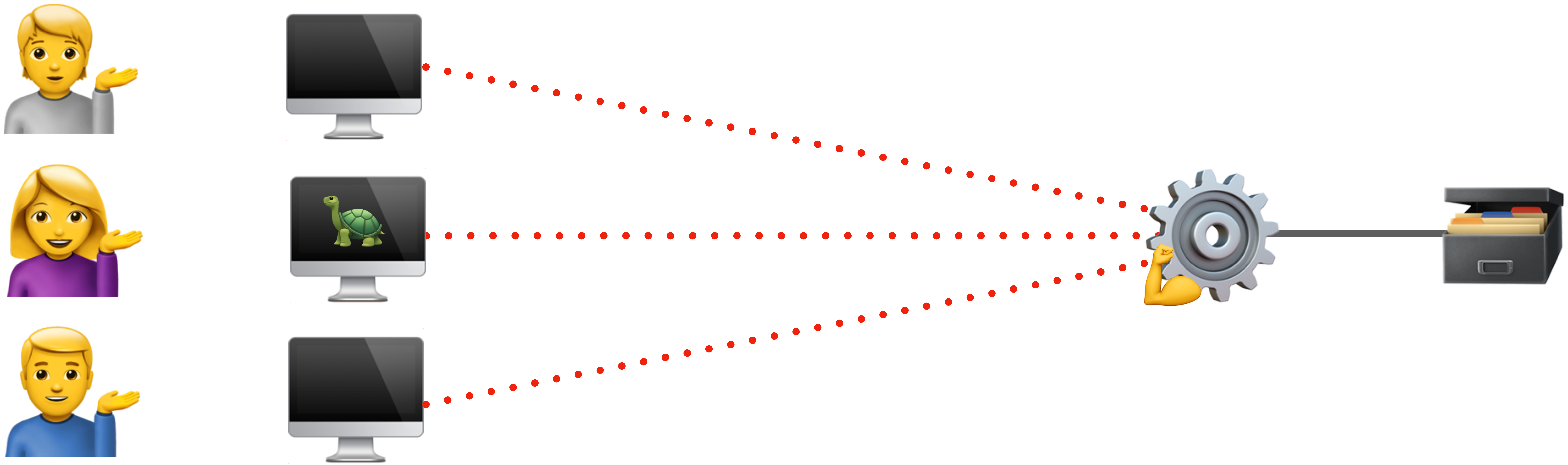
Motivation 🎭

90s Web



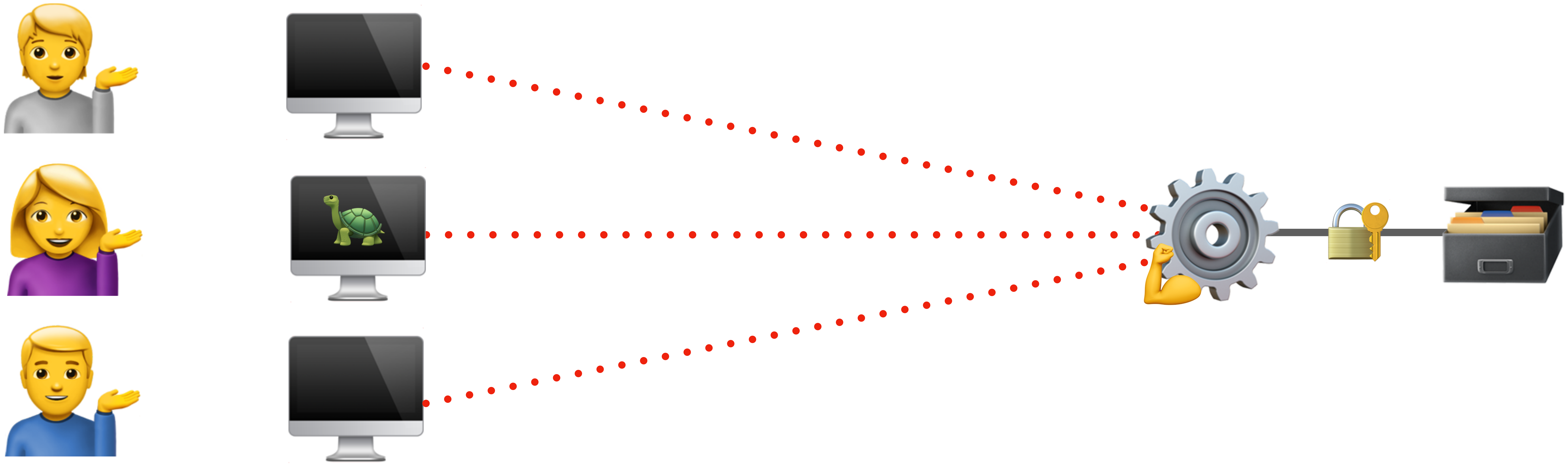
Motivation 🎭

90s Web



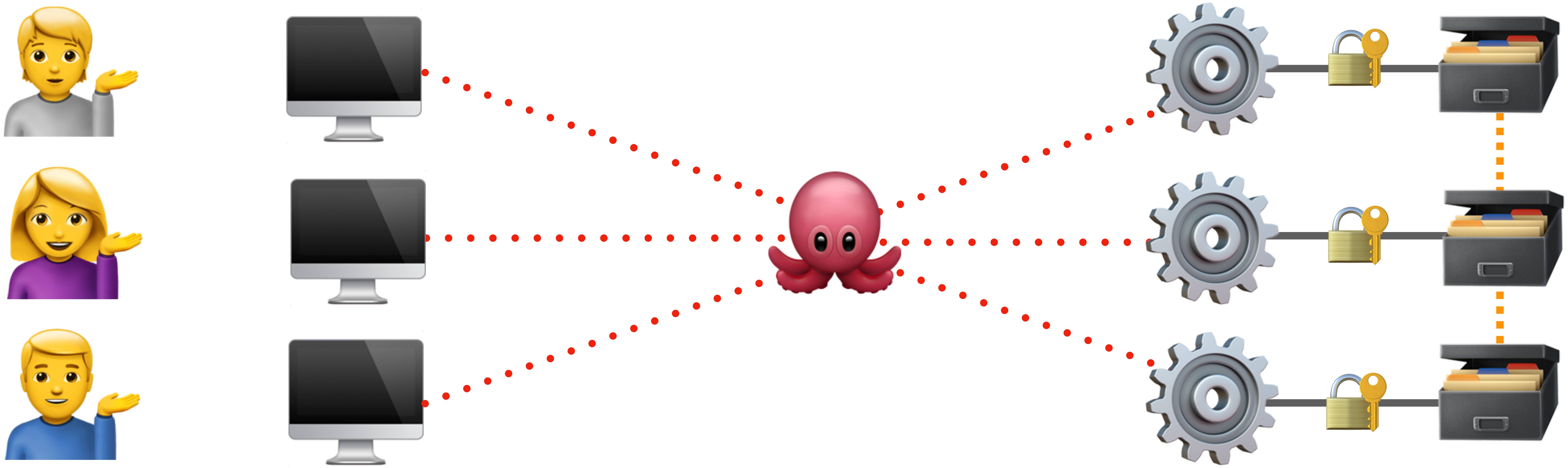
Motivation 🎭

90s Web



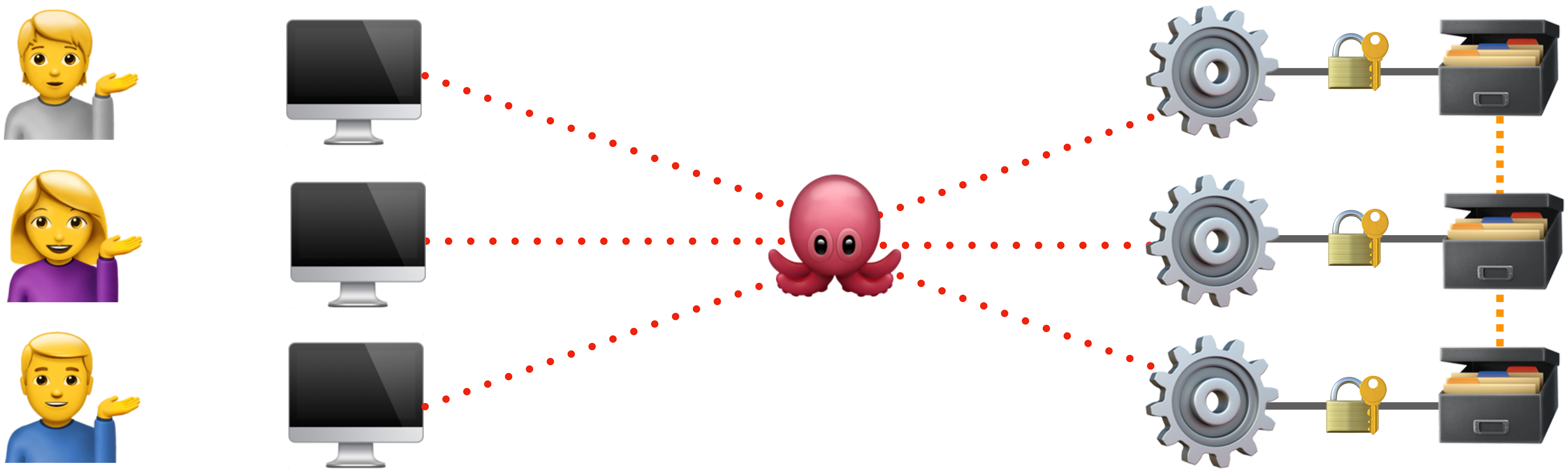
Motivation 🎭

Scaling Up



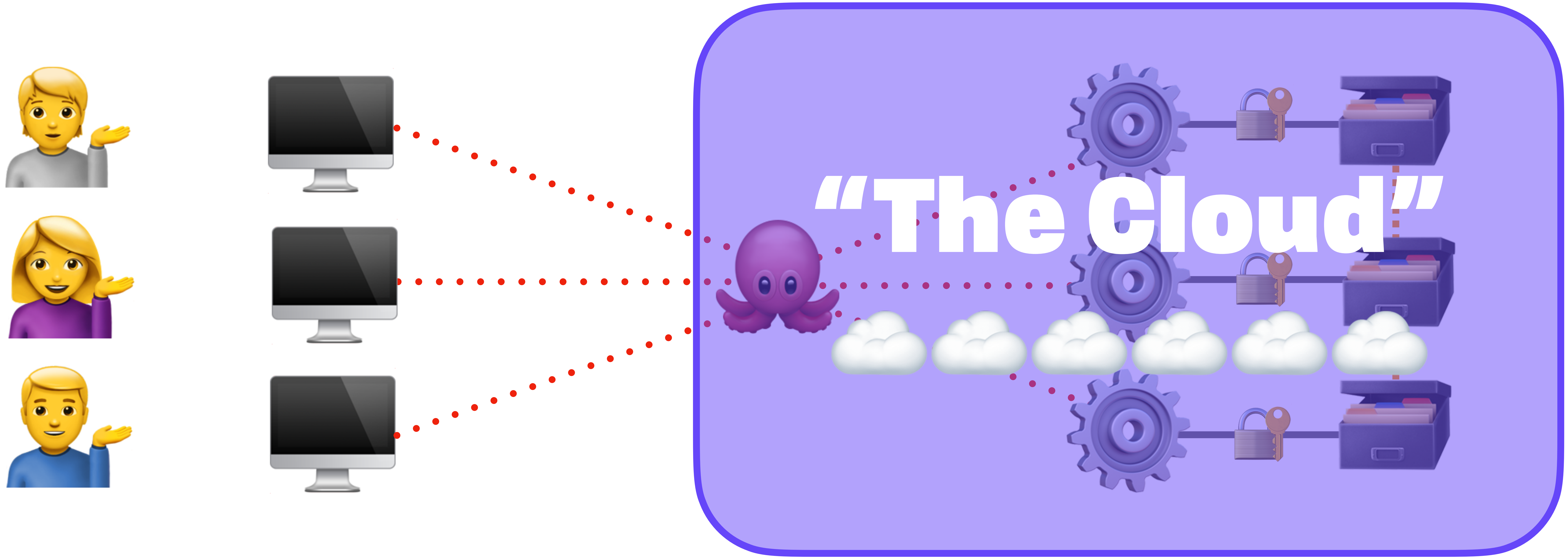
Motivation 🎭

Scaling Up



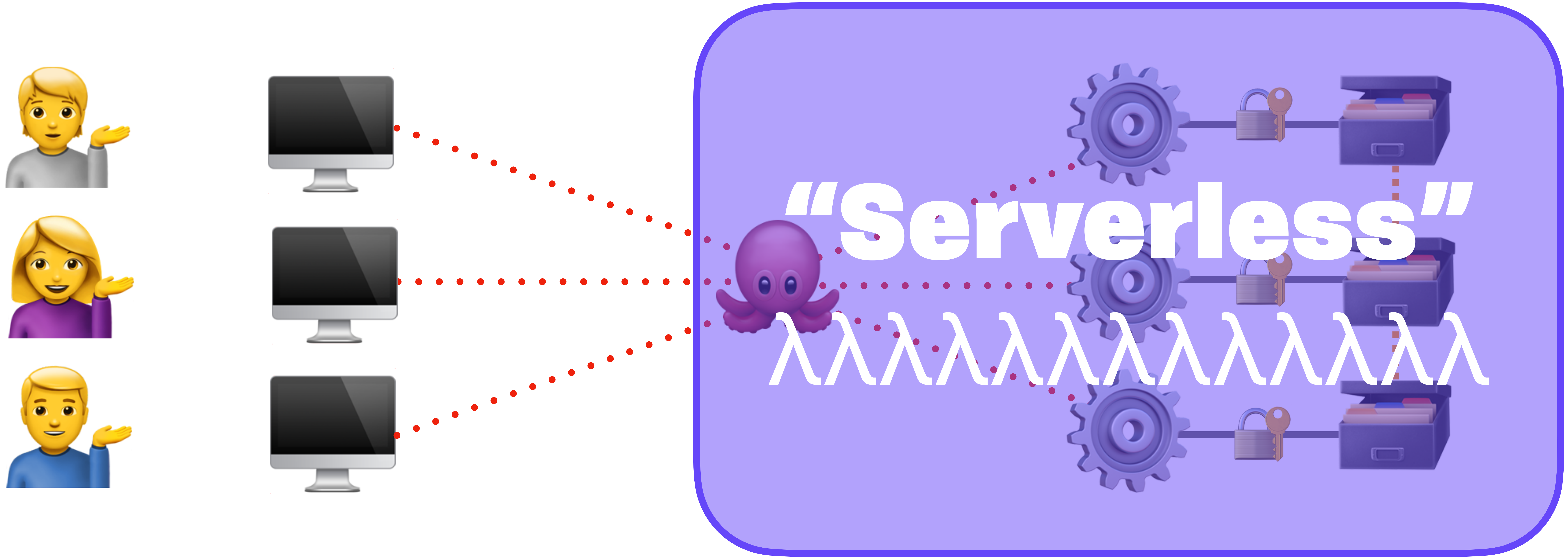
Motivation 🎭

Scaling Up



Motivation 🎭

Abstracting



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

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

🦖 ☄️ 🌋 🌾 🏰 🏢 🚀

Motivation 🎭

Natural Consequences 🍃

Motivation 🎭

Natural Consequences 🍃

- Server-focus
  - More stack to learn
  - DevOps, Docker, k8s

Motivation 🎭

Natural Consequences 🍃

- Server-focus
  - More stack to learn
  - DevOps, Docker, k8s
- Single source of truth
  - i.e. “the database”

Motivation 🎭

Natural Consequences 🍃

- Server-focus
  - More stack to learn
  - DevOps, Docker, k8s
- Single source of truth
  - i.e. “the database”
- Client concerned with data sync

Motivation 🎭

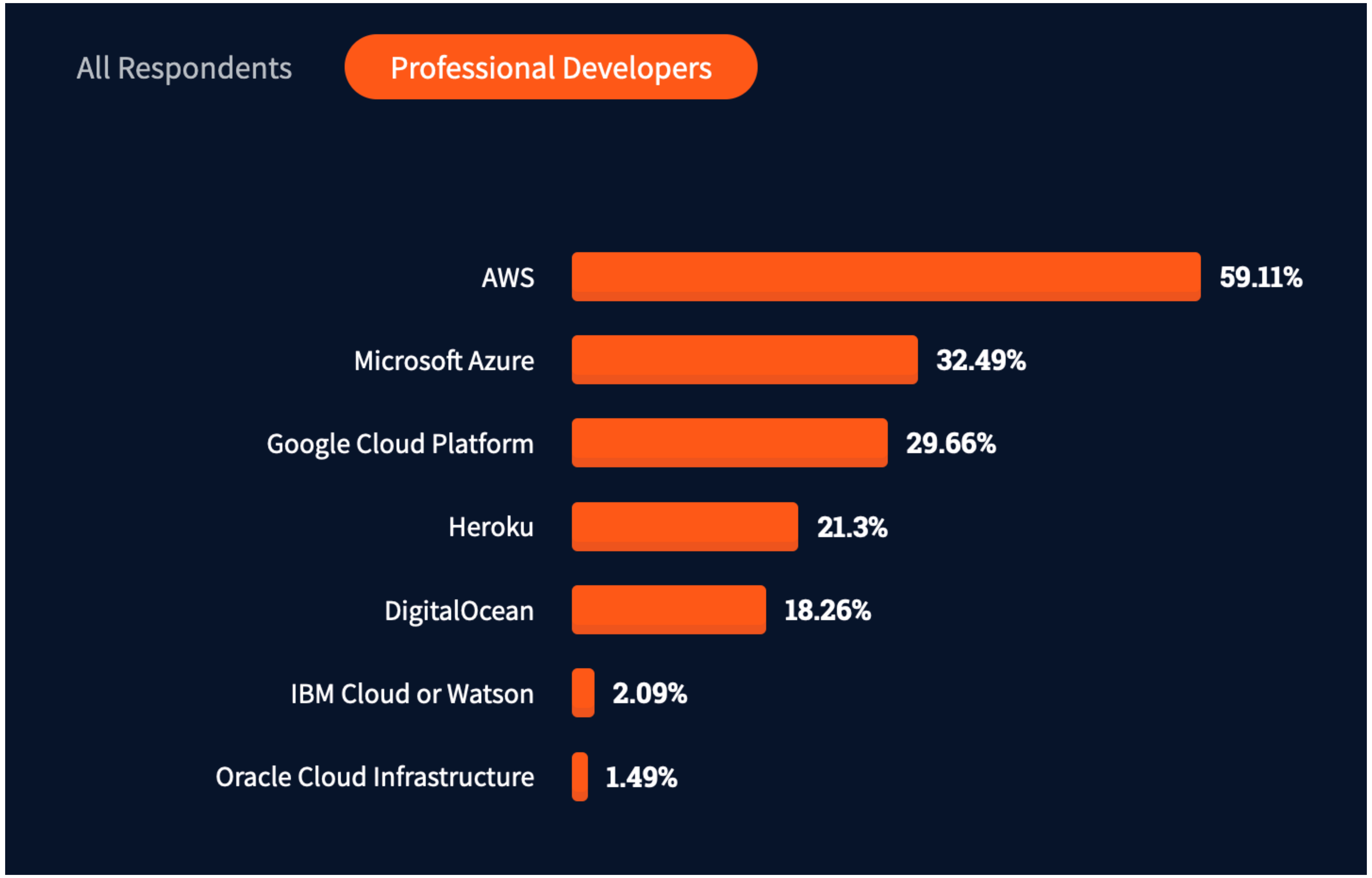
Natural Consequences 🍃

- Server-focus
  - More stack to learn
  - DevOps, Docker, k8s
- Single source of truth
  - i.e. “the database”
- Client concerned with data sync
- AWS, Azure, GCP

Motivation 🎭

Natural Consequences 🍃

- Server-focus
  - More stack to learn
  - DevOps, Docker, k8s
- Single source of truth
  - i.e. “the database”
- Client concerned with data sync
- AWS, Azure, GCP



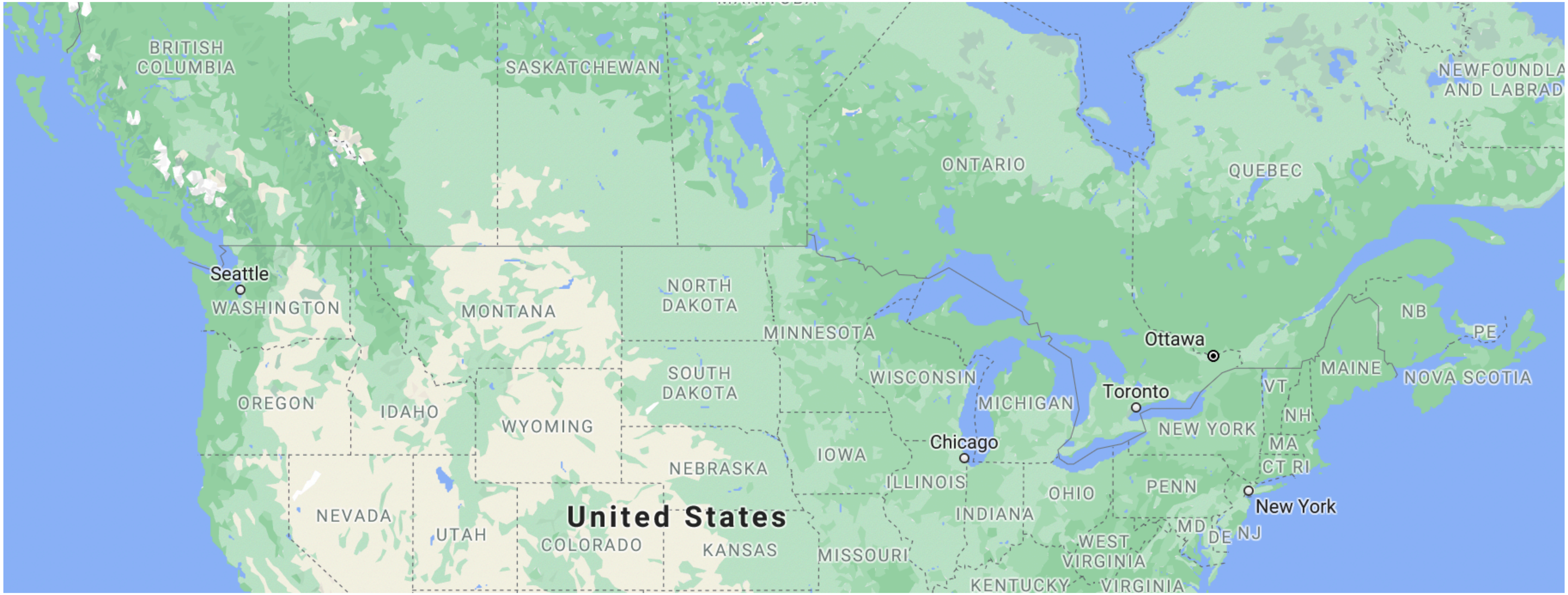
Source: 2021 Stack Overflow Developer Survey

Motivation 🎭

Sending a “Direct” Message

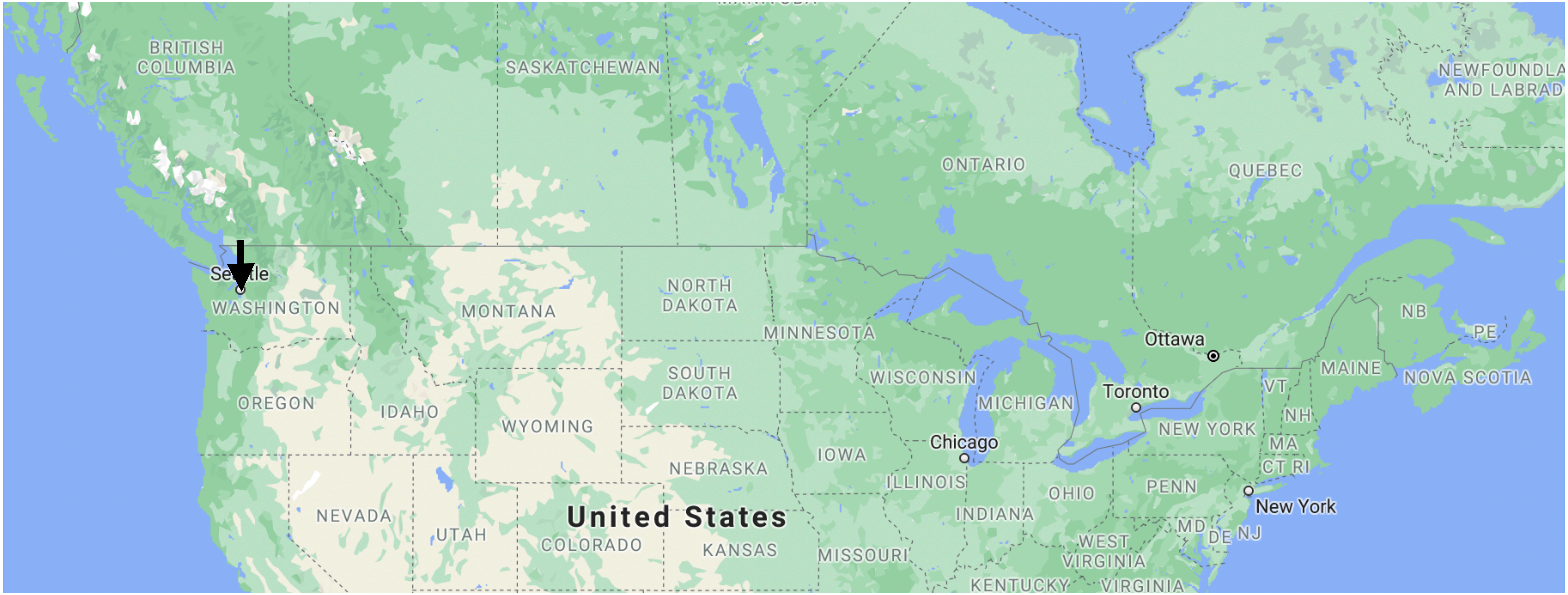
Motivation 🎭

Sending a "Direct" Message



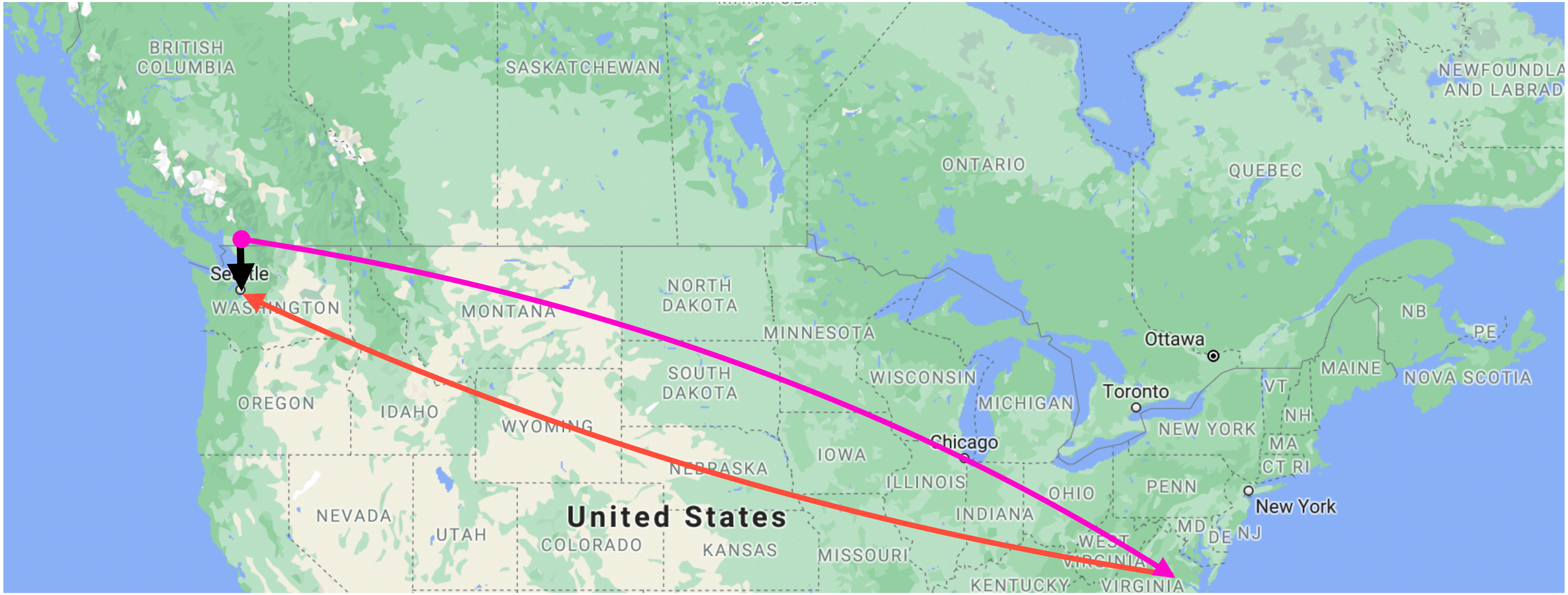
Motivation 🎭

Sending a “Direct” Message



Motivation 🎭

Sending a "Direct" Message



Motivation 🎭

What Even is a “Server”? 🧐

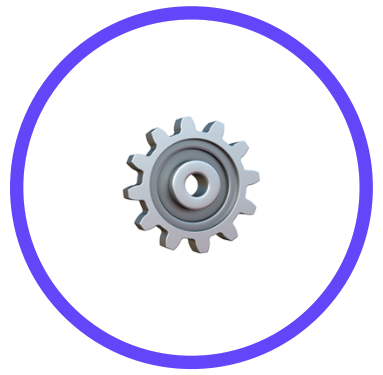
1. Auth gatekeeper (because multi-tenant data)
2. Resource availability
3. Out-of-band compute (e.g. batch tasks, cron, OLAP)

Motivation 🎭

Network Topology 🧠

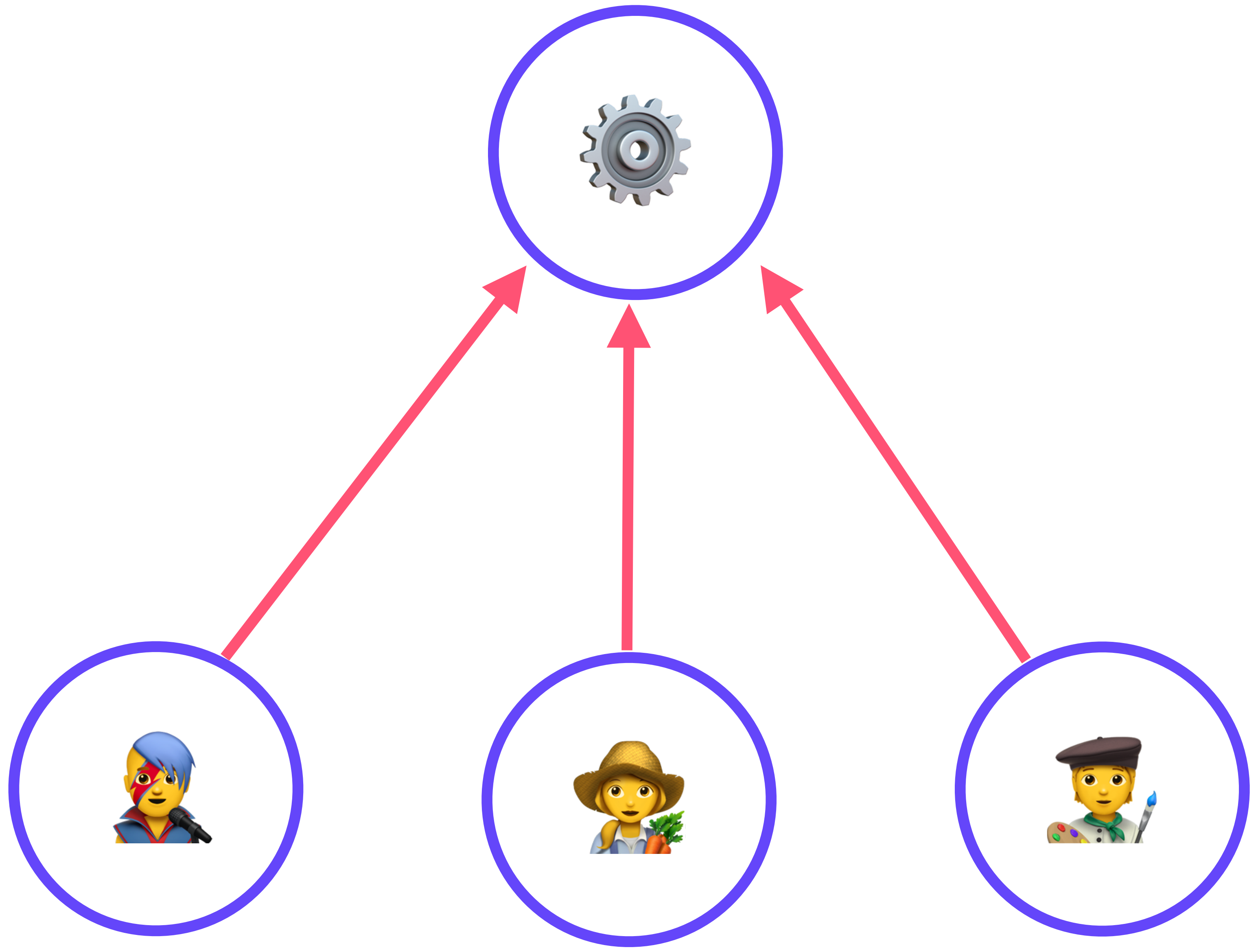
Motivation 🎭

Network Topology 🧠



Motivation 🎭

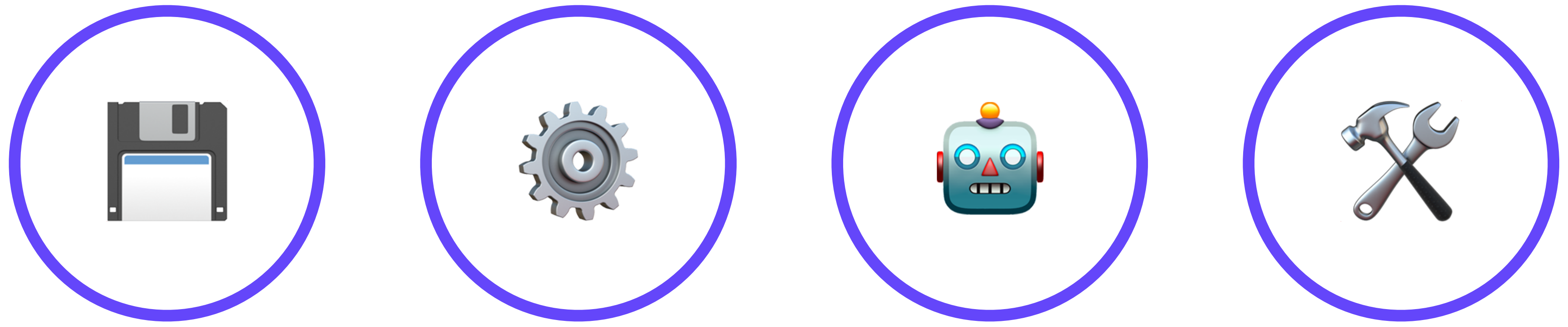
Network Topology 🧠

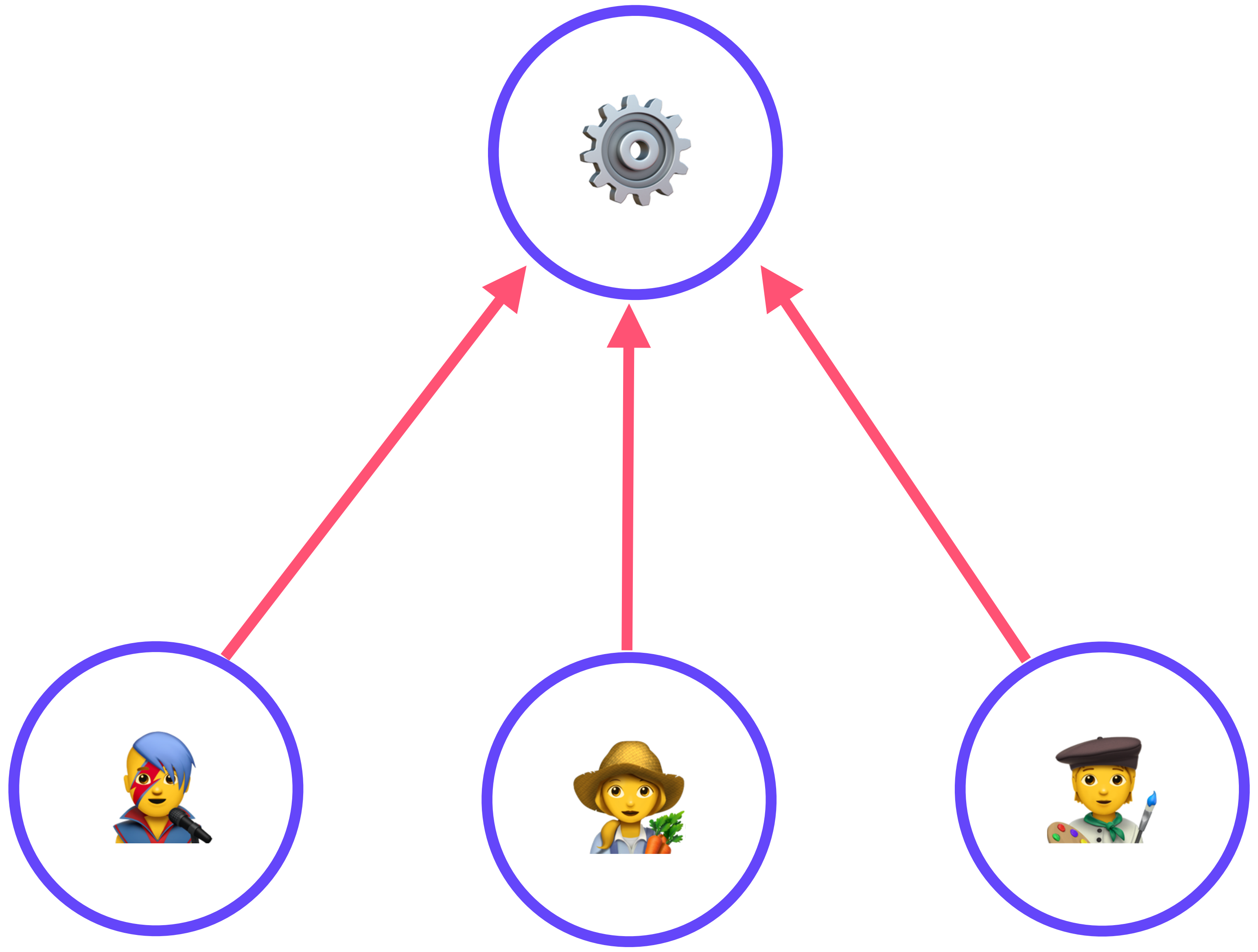


Centralized

Motivation 🎭

Network Topology 🧠

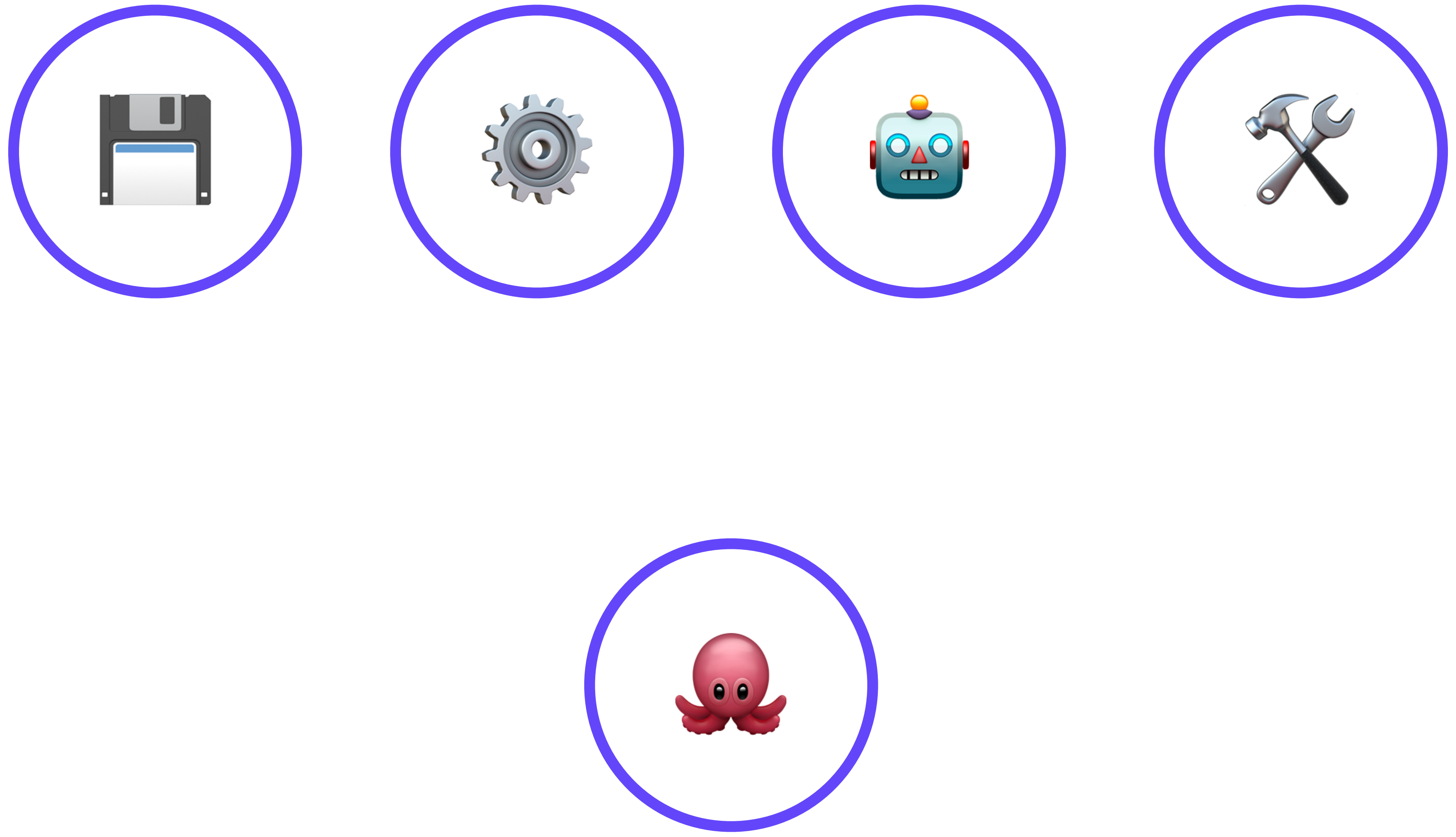


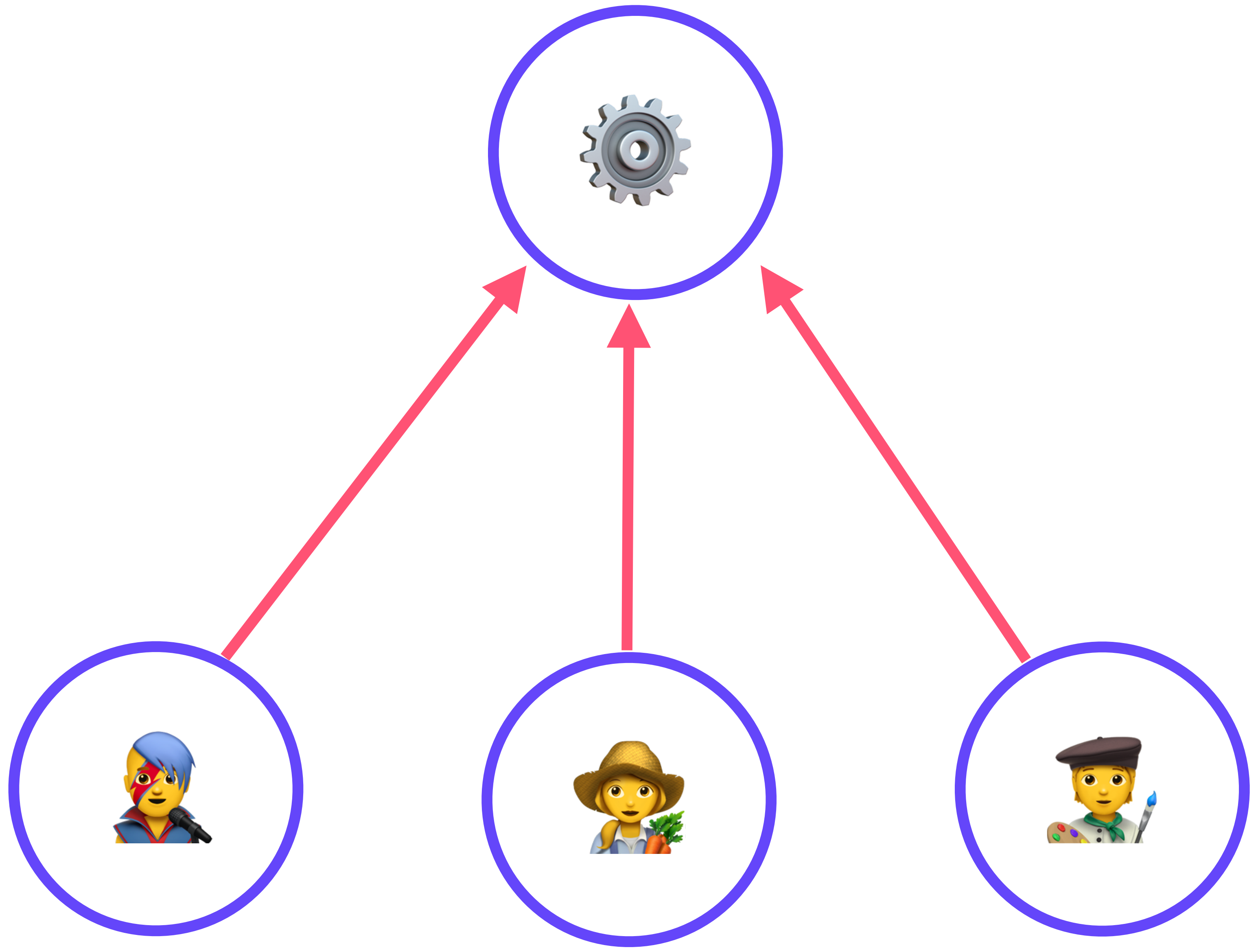


Centralized

Motivation 🎭

Network Topology 🧠

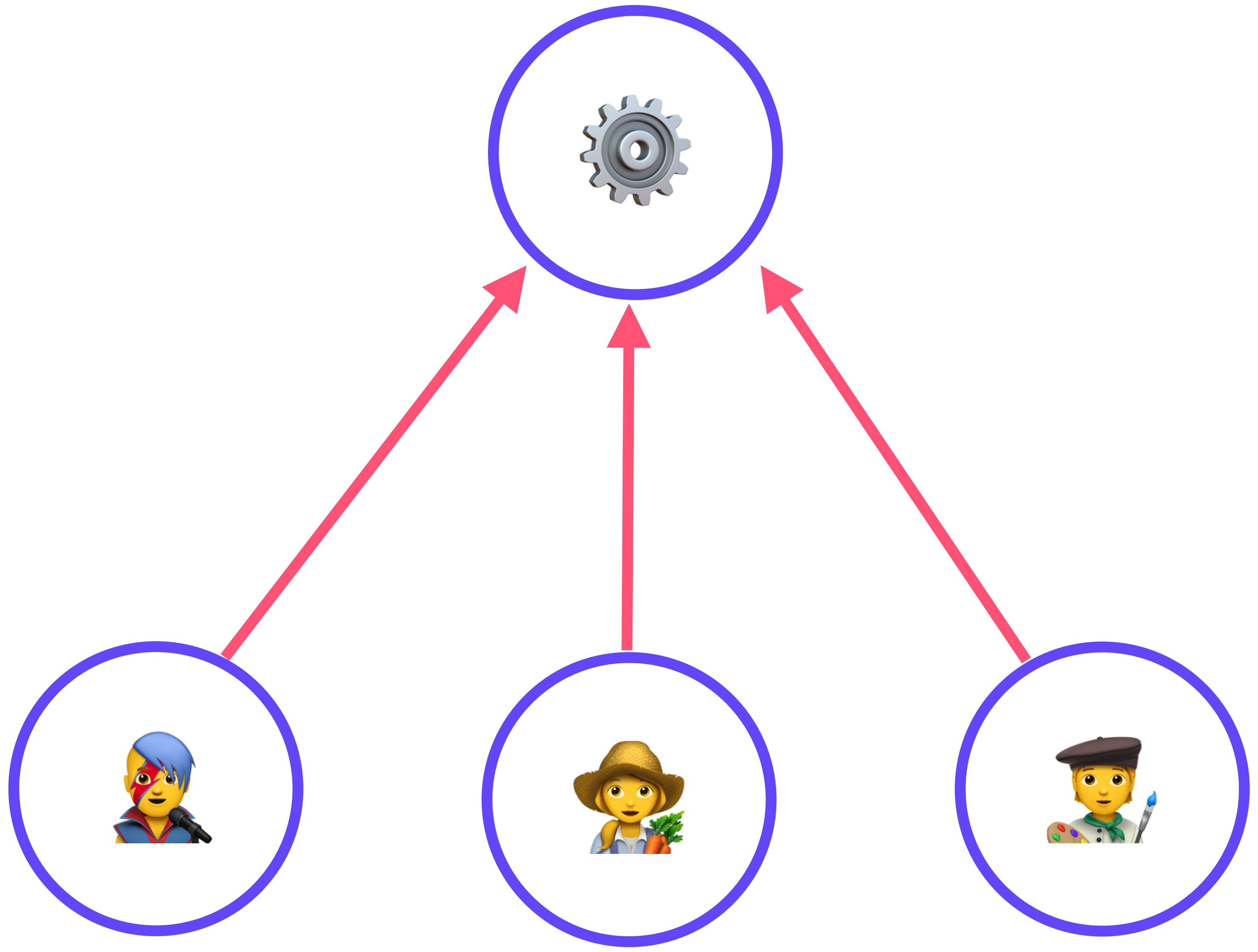




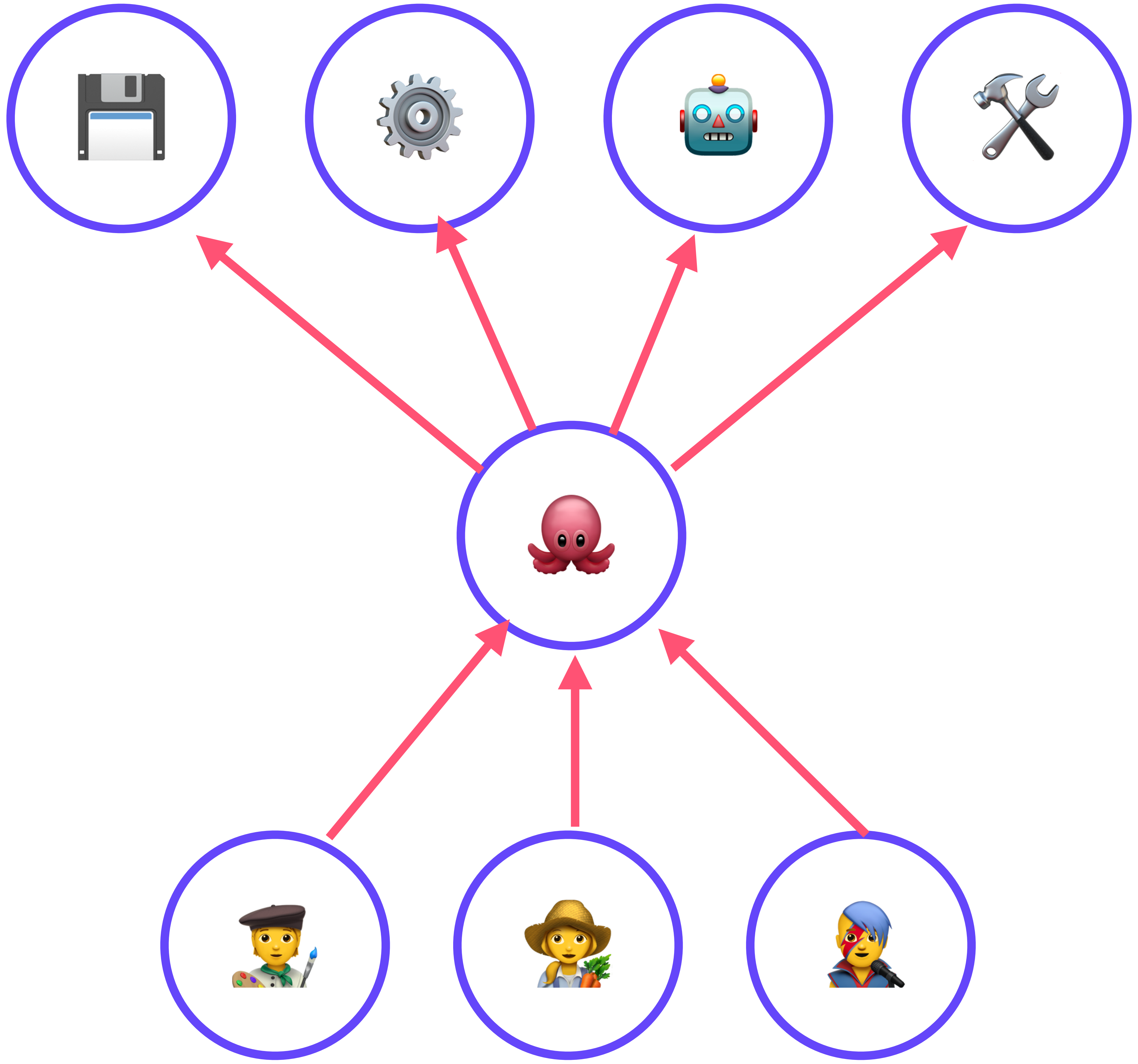
Centralized

Motivation 🎭

Network Topology 🧠



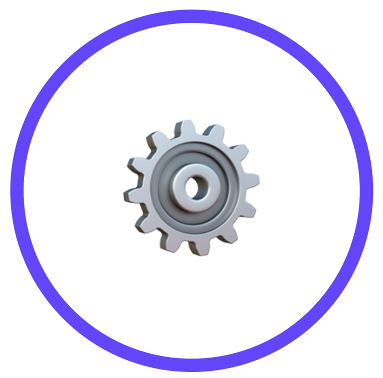
Centralized

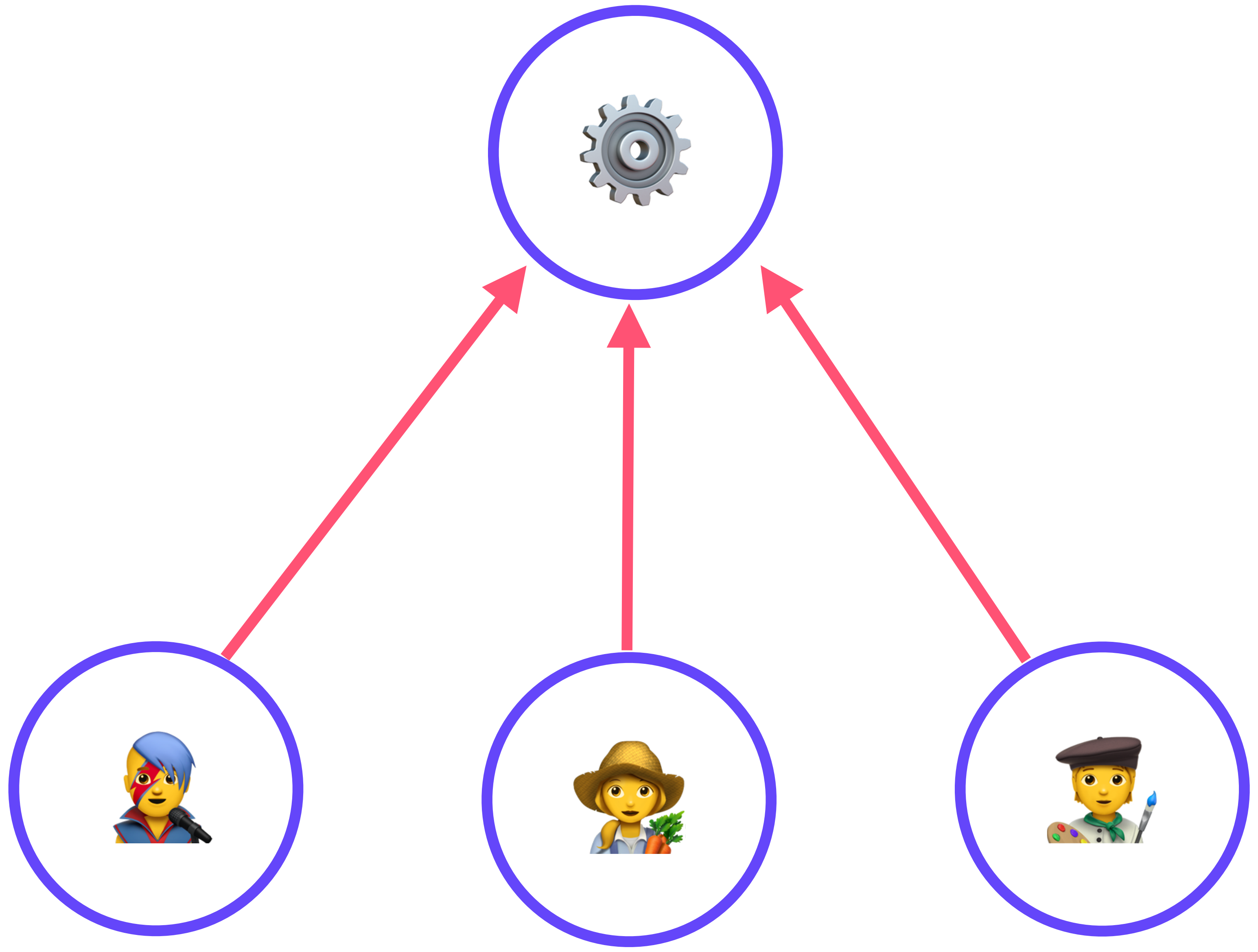


Hub (e.g. gateway or load balanced)

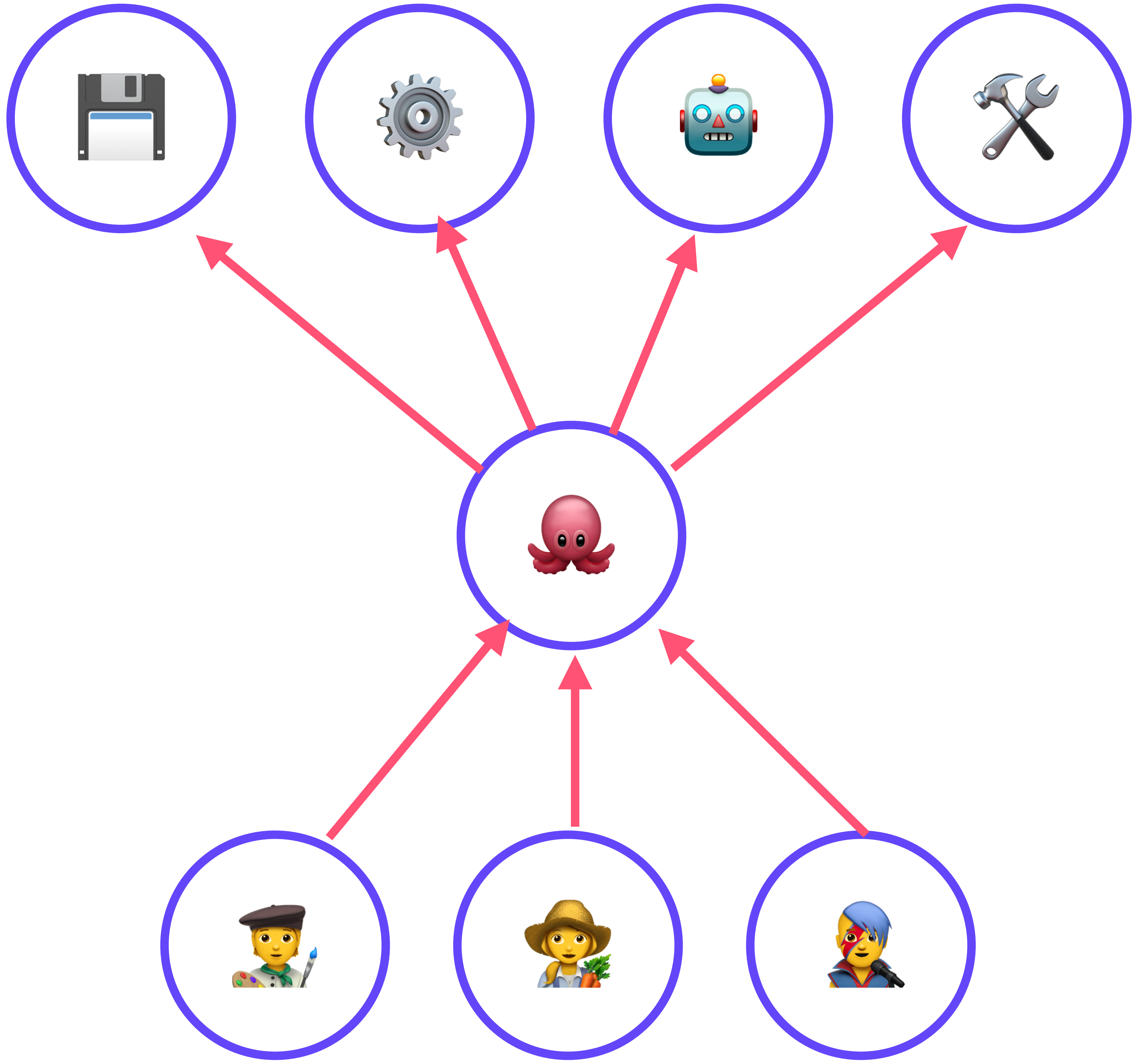
Motivation 🎭

Network Topology 🧠





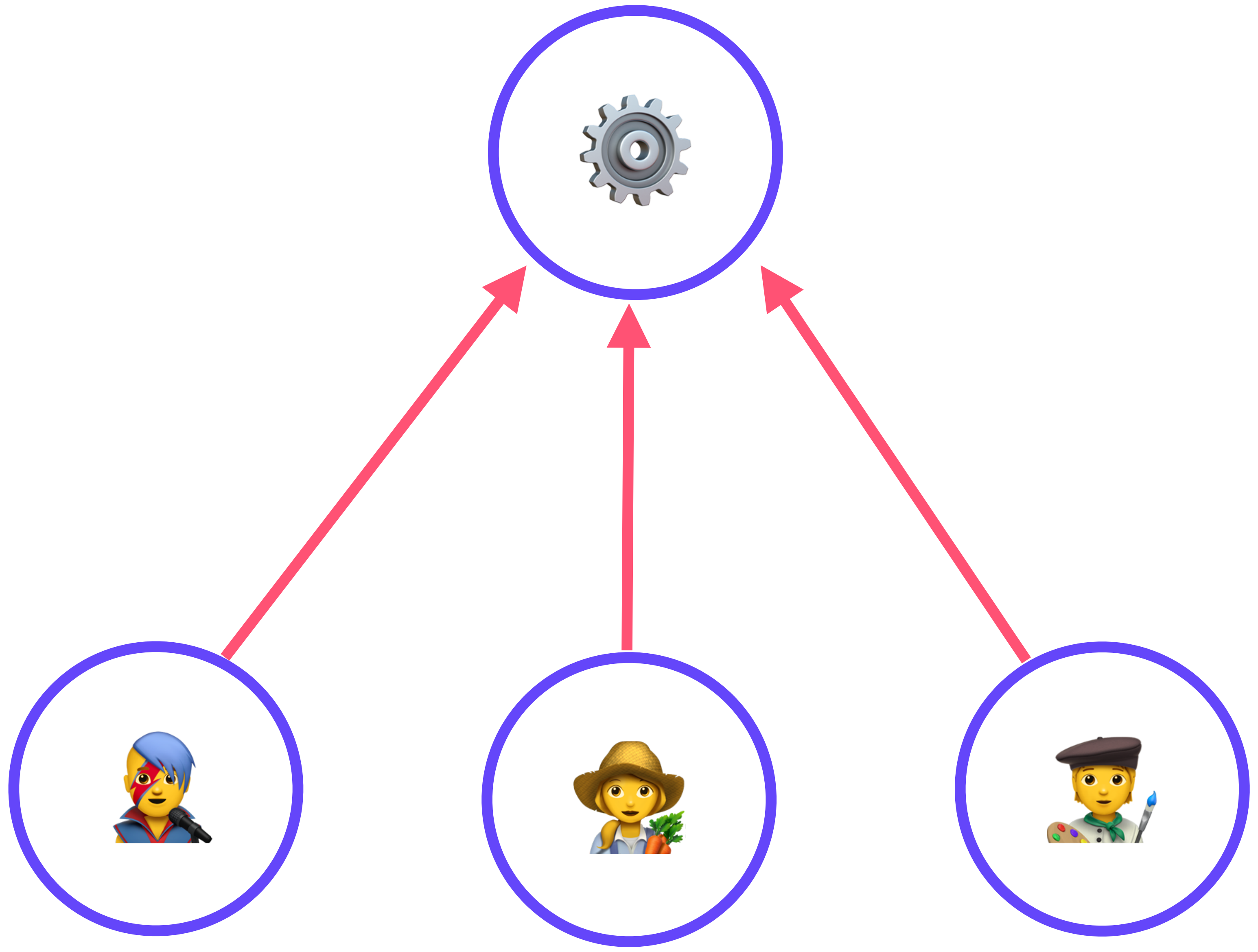
Centralized



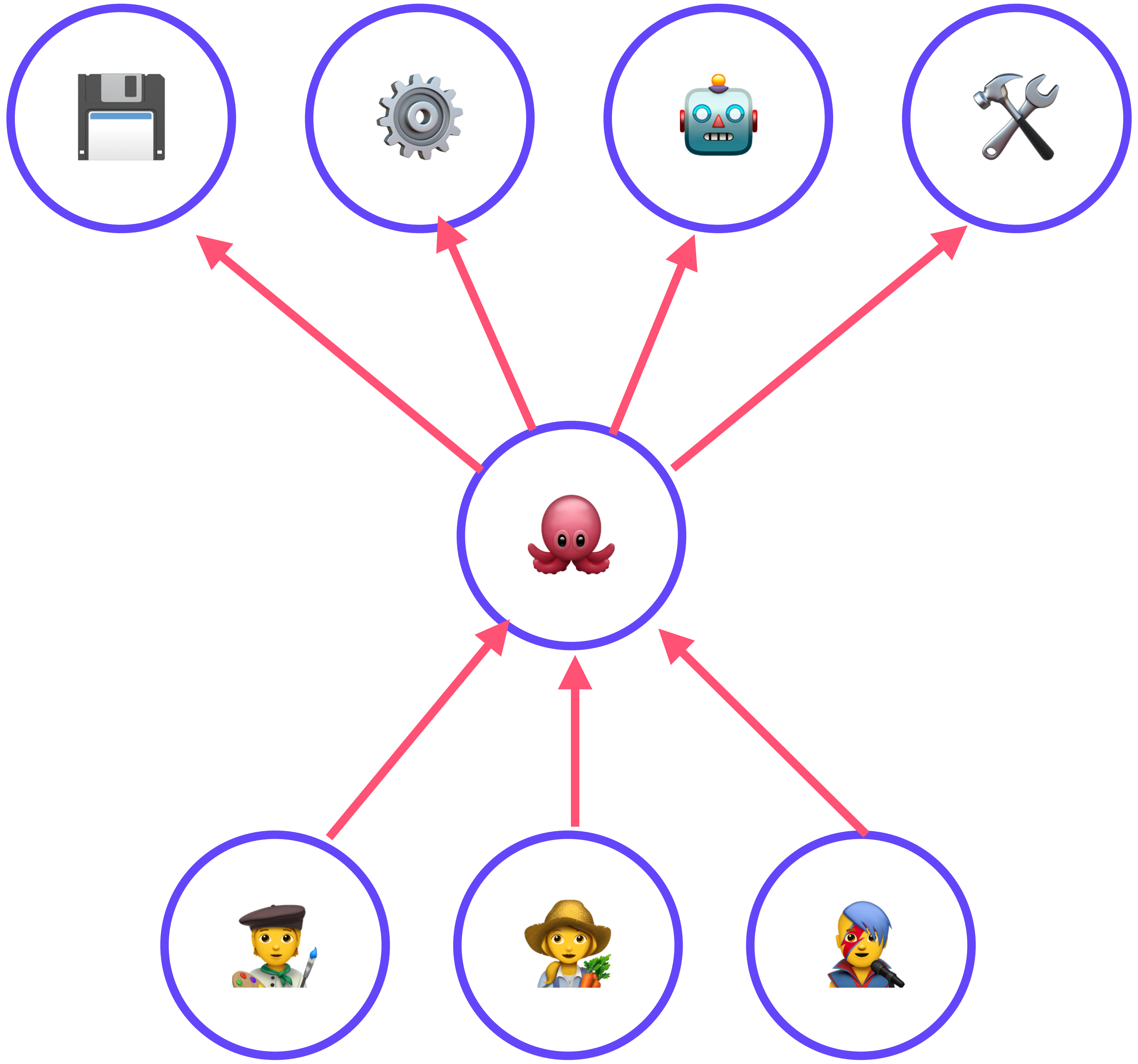
Hub (e.g. gateway or load balanced)

Motivation 🎭

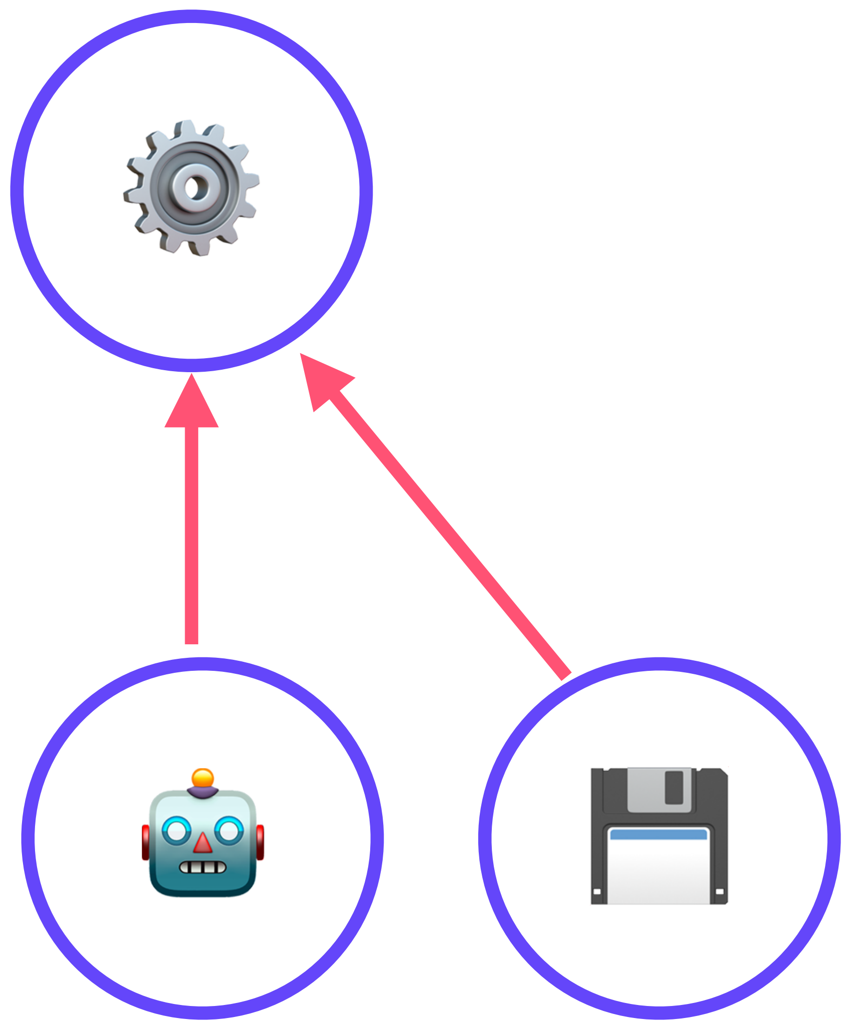
Network Topology 🧠



Centralized

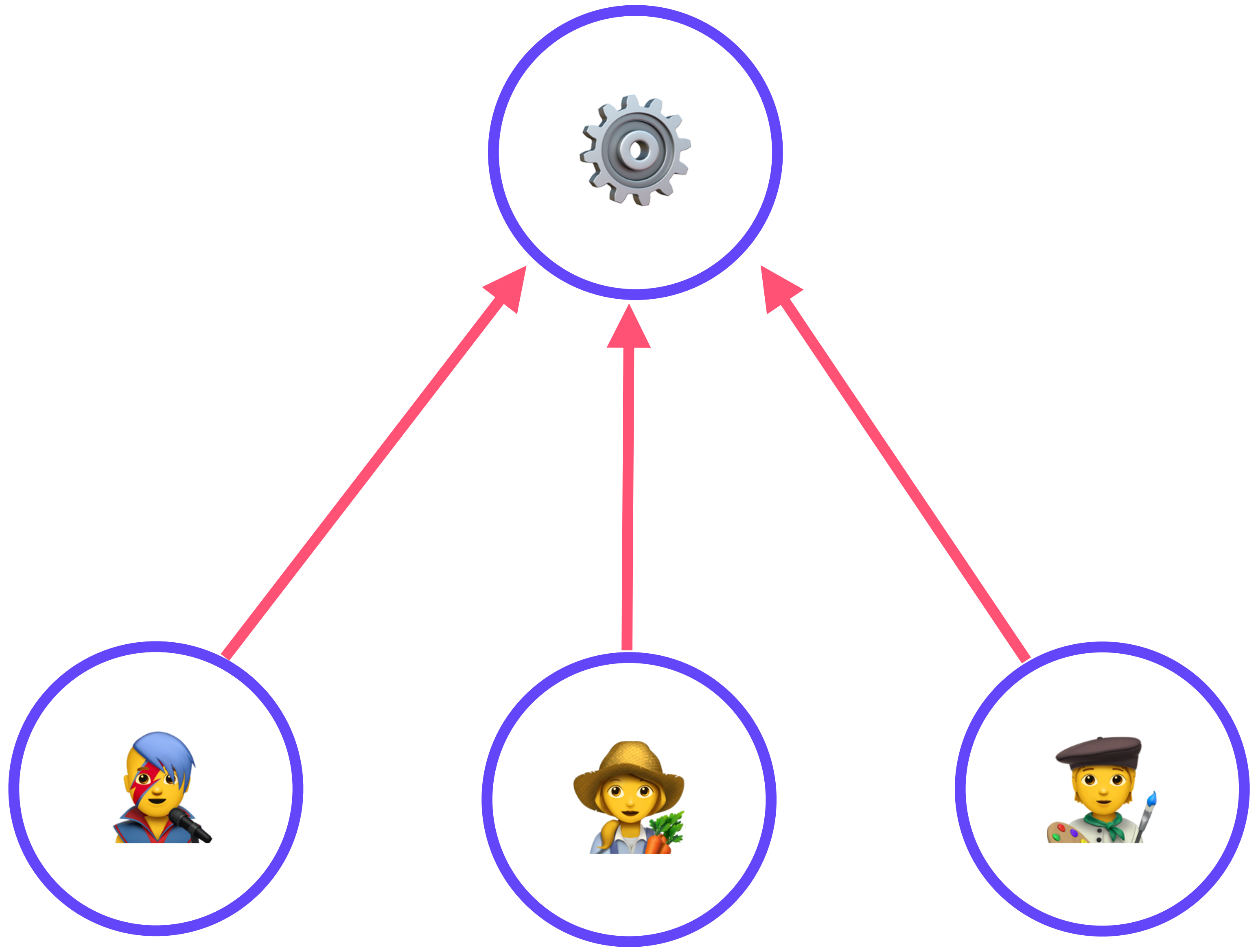


Hub (e.g. gateway or load balanced)

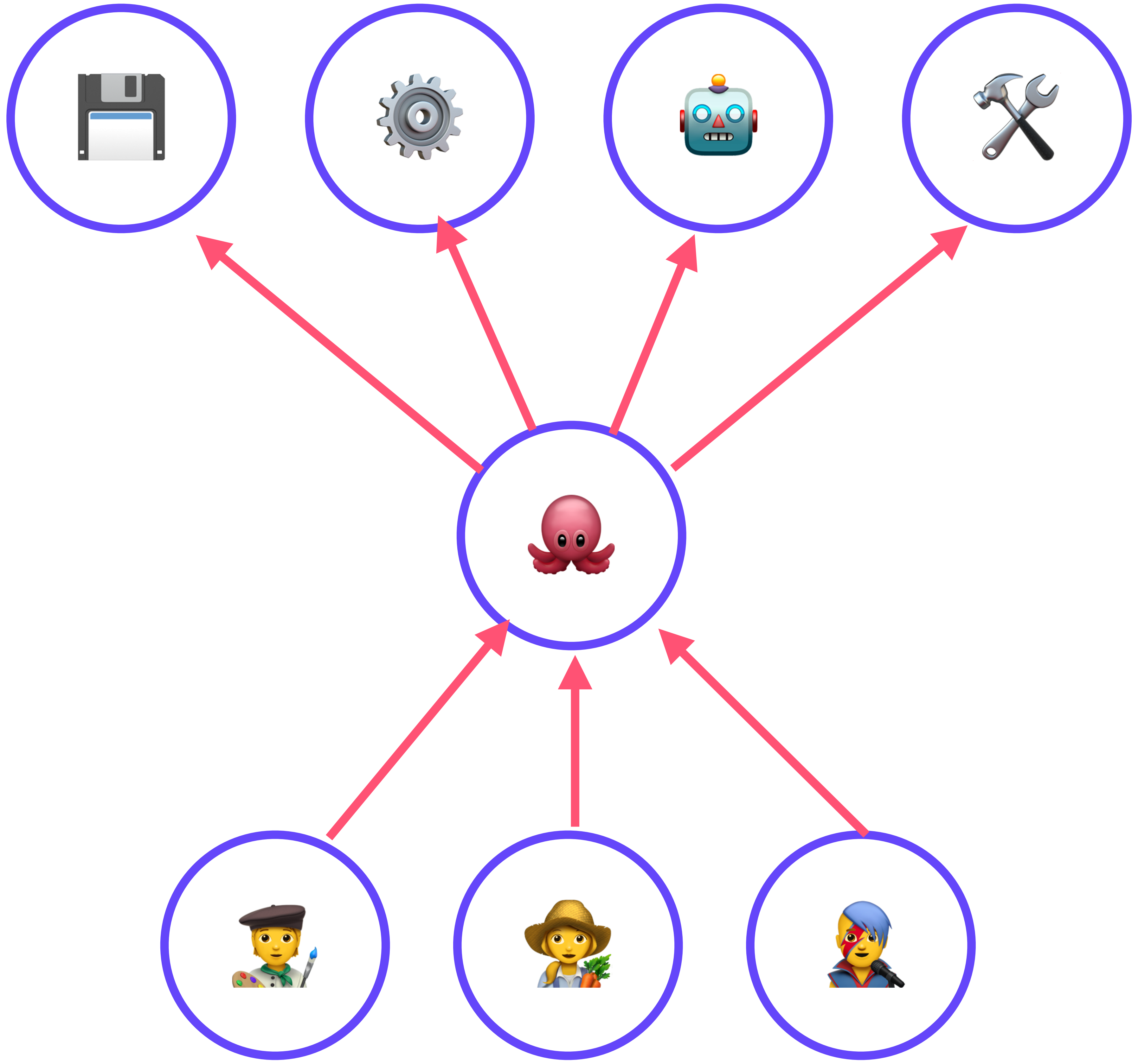


Motivation 🎭

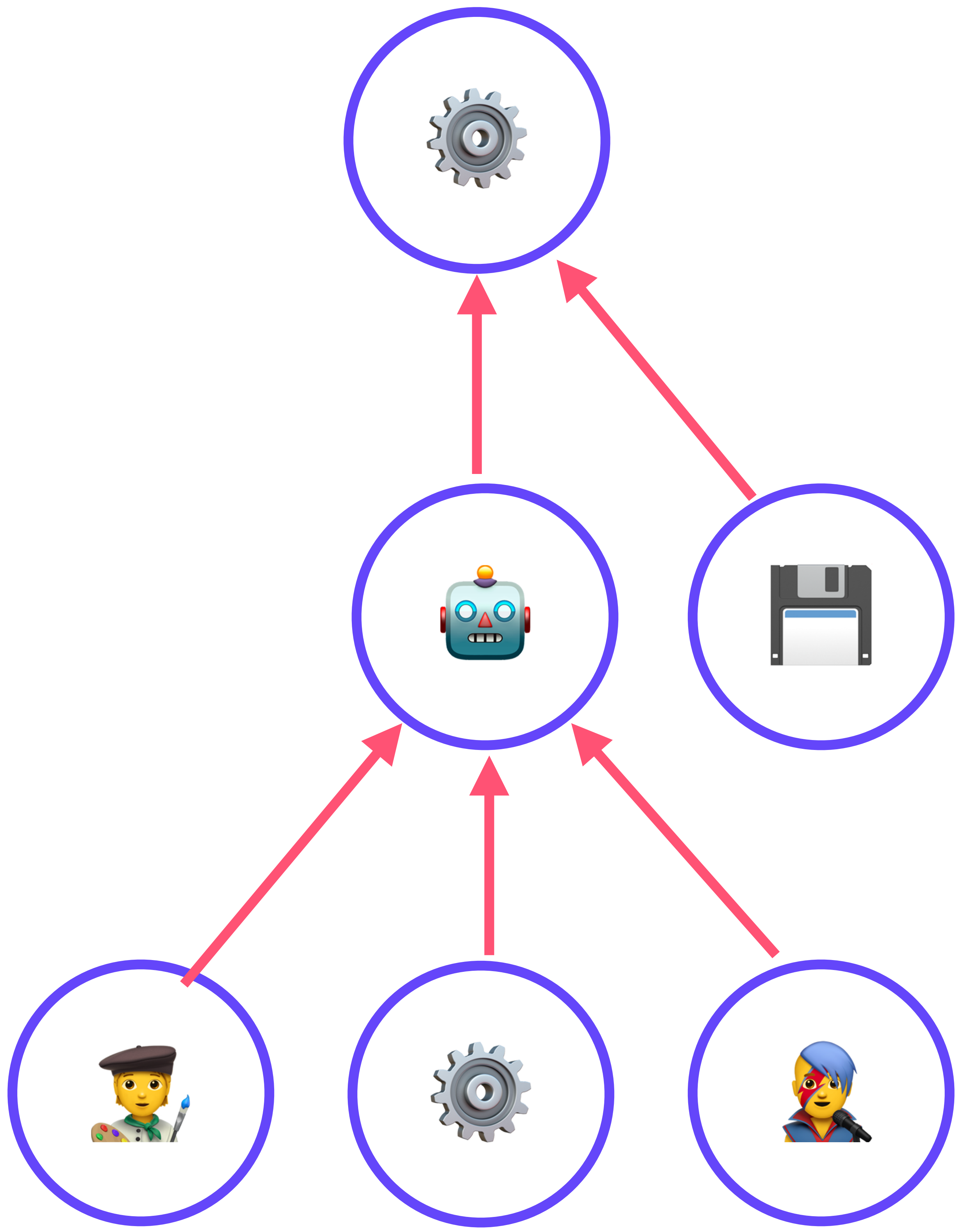
Network Topology 🧠



Centralized

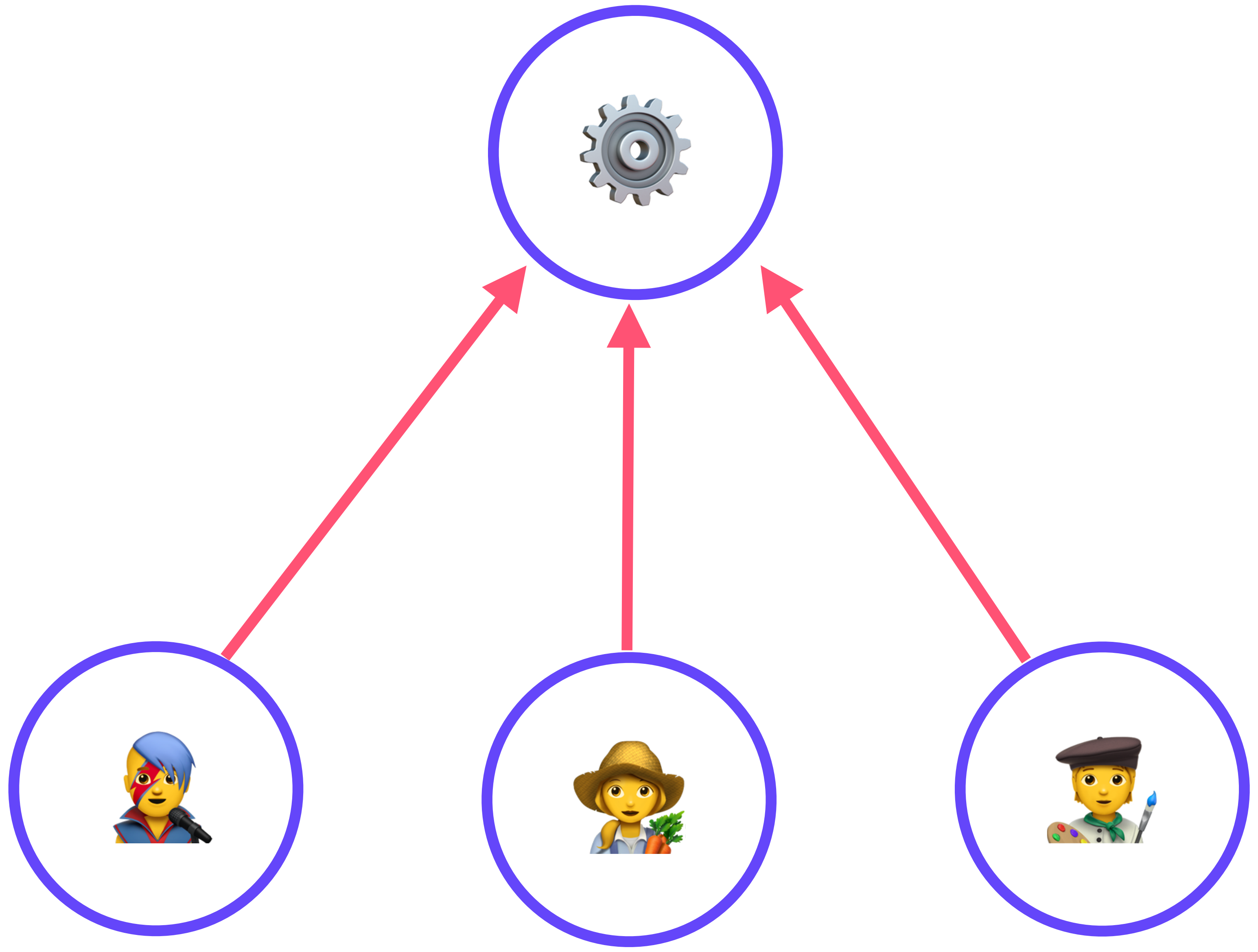


Hub (e.g. gateway or load balanced)

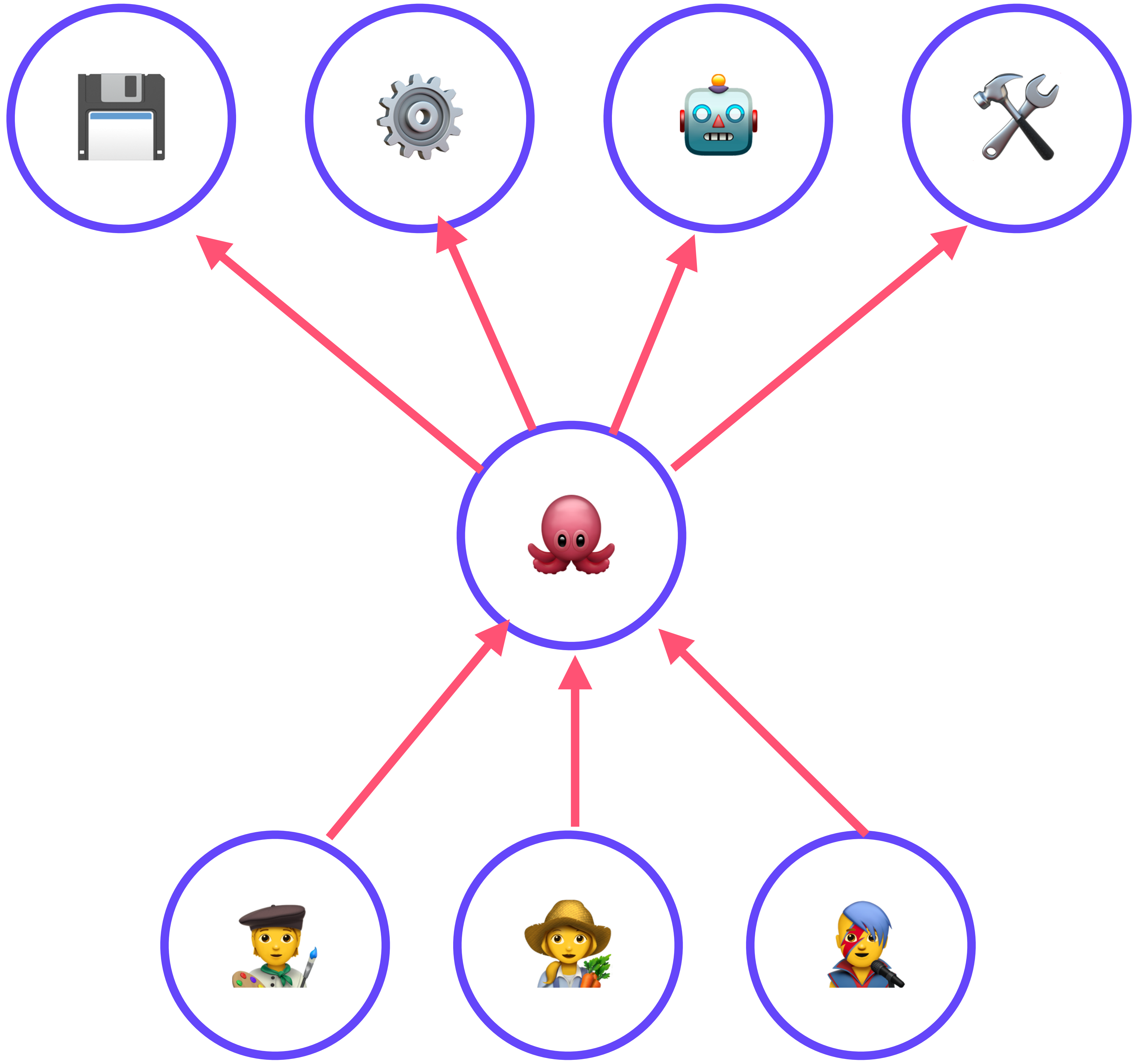


Motivation 🎭

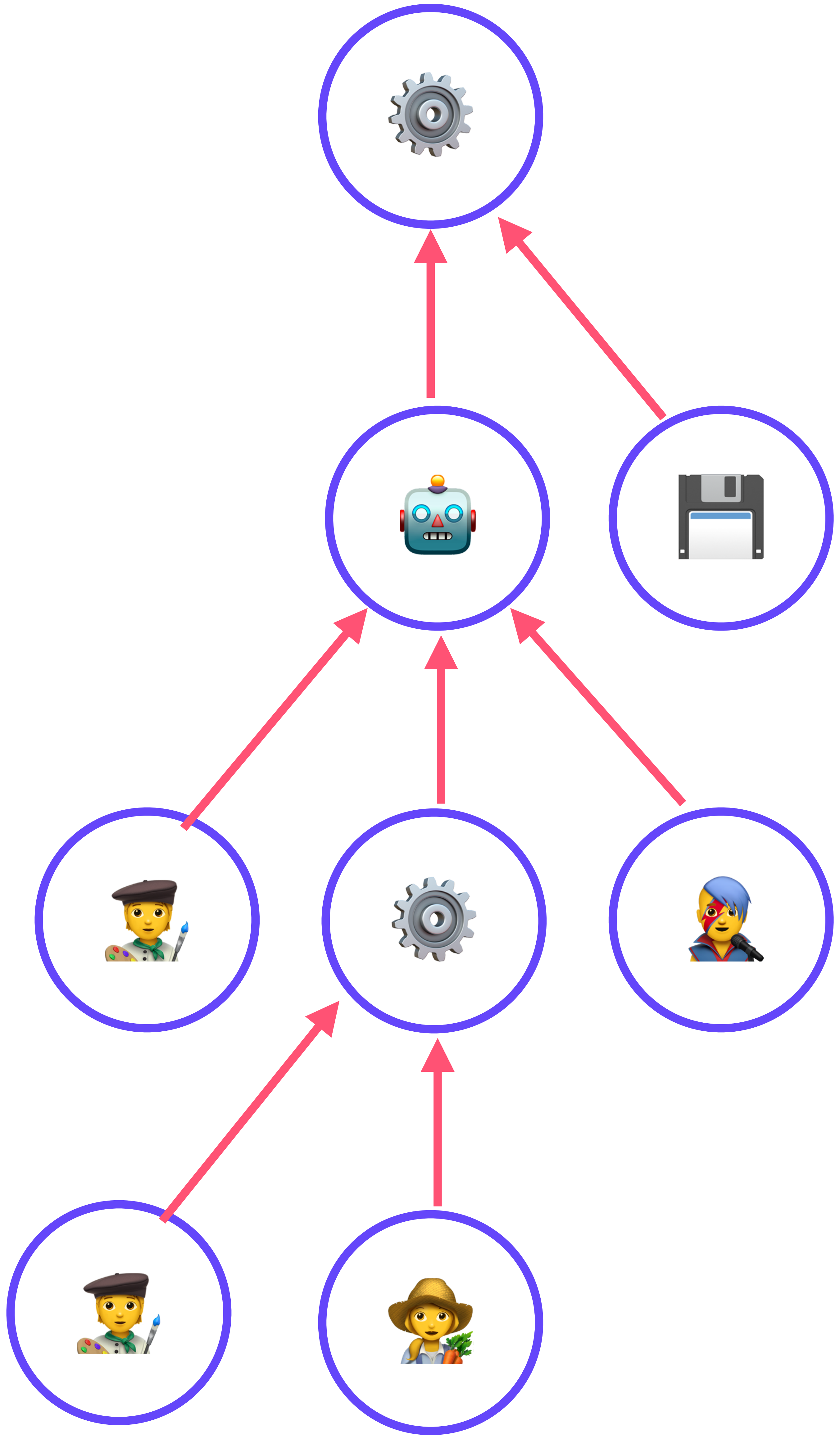
Network Topology 🧠



Centralized



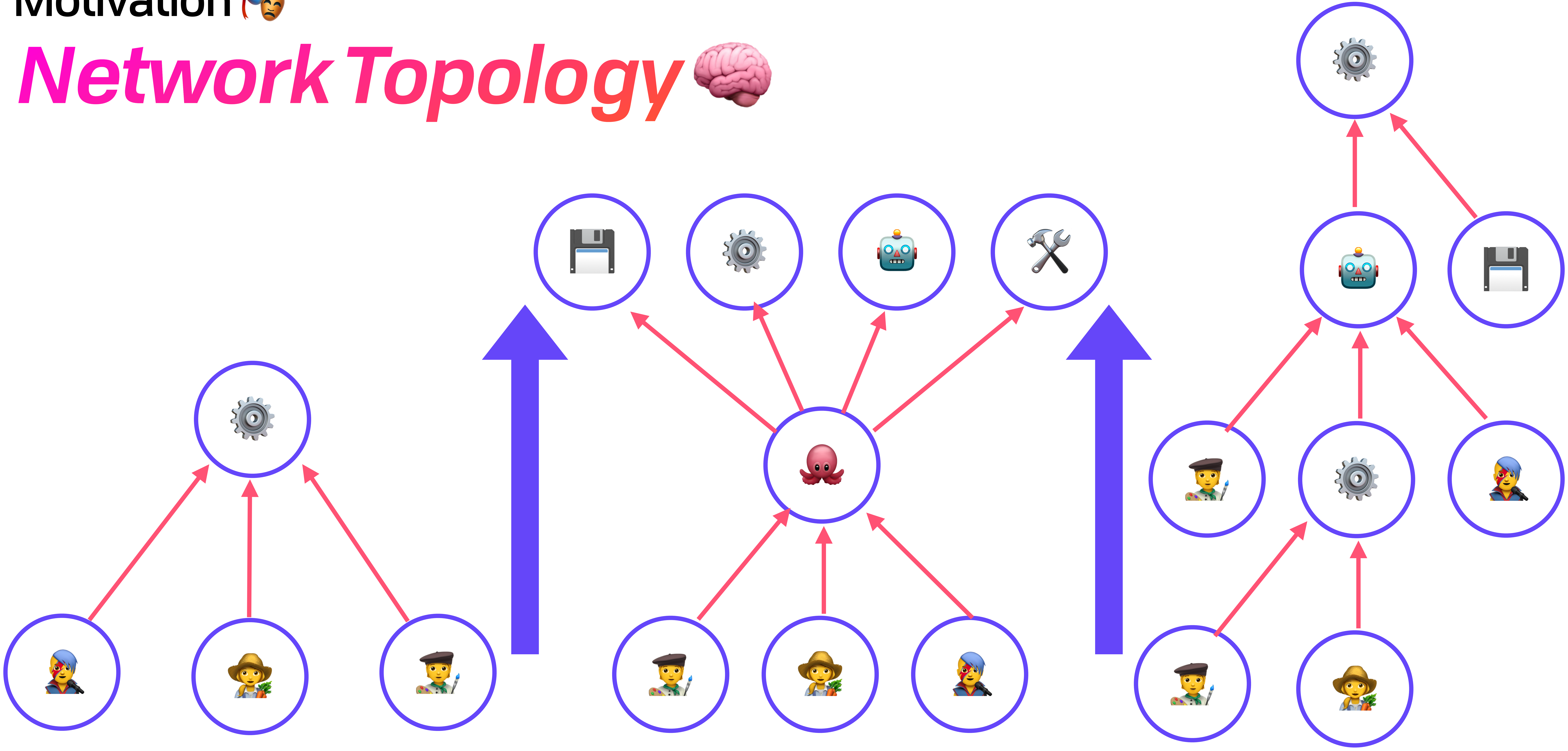
Hub (e.g. gateway or load balanced)



Hierarchical or pipelined

Motivation 🎭

Network Topology 🧠



Centralized

Hub (e.g. gateway or load balanced)

Hierarchical or pipelined

A Challenger Emerges

A New Environment

New Environment 🛰️

New Assumptions

- Powerful client devices (e.g. M1 chips, smartphones, IoT)
- Latency is the bottleneck
- Mobile (i.e. smartphone) use only growing
  - Lose connection, drop when switching towers
- Do more with the existing physical network
  - Not unlike how Moore's Law lead to more parallelism

New Environment 🛰️

New Biz Who Dis?

New Environment 🛰️

New Biz Who Dis?

- Paradigm shift means new opportunities

New Environment 🛰️

New Biz Who Dis?

- Paradigm shift means new opportunities
- 5G networks & Starlink
  - Put an edge PoP right on the base station
  - Low-latency compute across the street

STARLINK

New Environment 🛰️

New Biz Who Dis?

- Paradigm shift means new opportunities
- 5G networks & Starlink
  - Put an edge PoP right on the base station
  - Low-latency compute across the street
- Edge PoPs in retail stores (yes really)
  - 90% of Americans live <16km from a Walmart
  - Walmart has lots of floor space
  - Add servers to Walmart = Walmart Edge

STARLINK

A New Environment

Low Latency

🐇

Low Latency 🐇

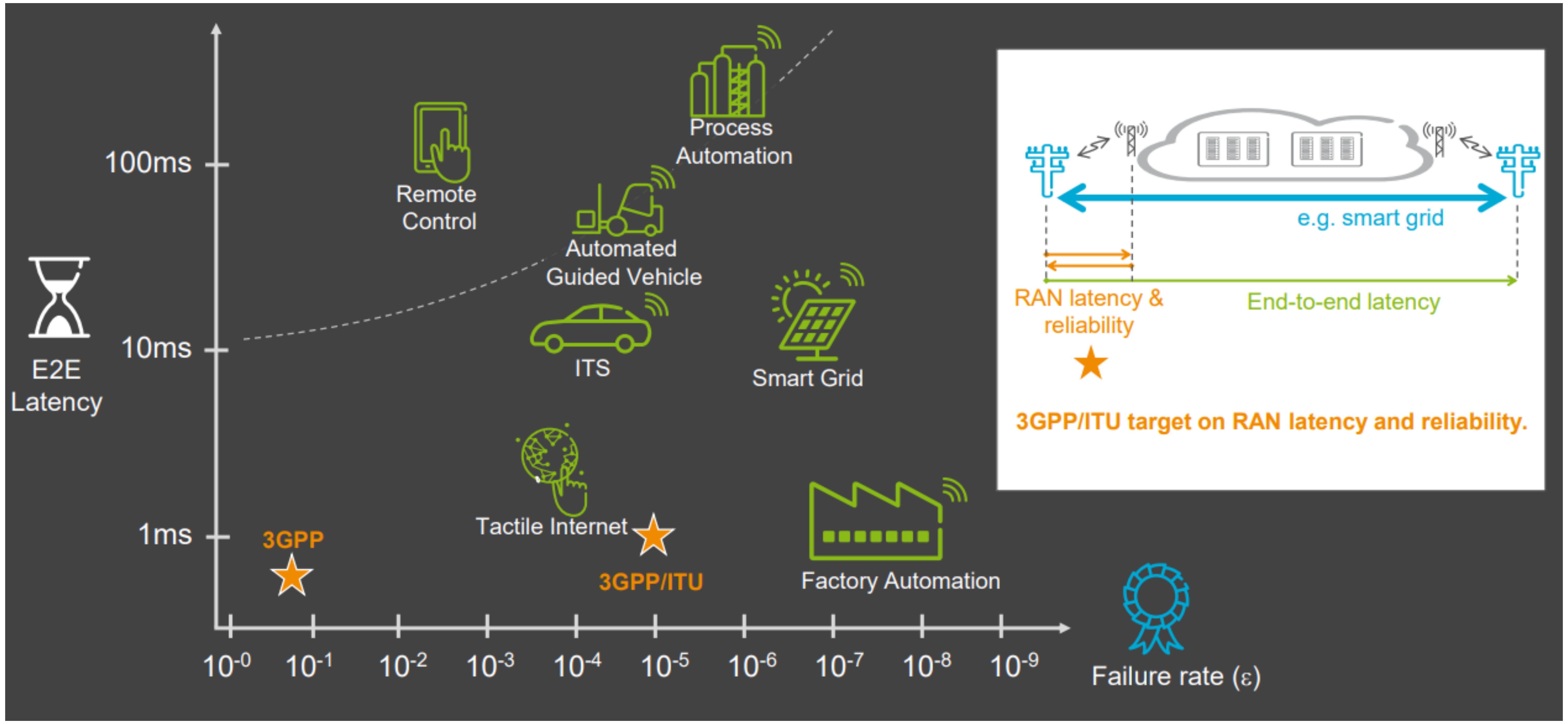
Latency is a Physical Barrier 🚧

- Speed of light / speed of *causality*
- <40ms = edge dominates
- 8ms is ideal
- Ultra Reliable Low Latency (URLLC)

Low Latency 🐇

Latency is a Physical Barrier 🚧

- Speed of light / speed of *causality*
- <40ms = edge dominates
- 8ms is ideal
- Ultra Reliable Low Latency (URLLC)



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

Low Latency 🐇

Spherical Cow Assumption 🐮

- No compute, straight line, in a vacuum, guaranteed delivery, etc
- 40ms
  - São Paulo ➡️ NYC, Vancouver, Stockholm
  - São Paulo ❌ Sidney, Tokyo, Seoul

Credit: Keenan Crane
http://www.cs.cmu.edu/~kmcrane/Projects/ModelRepository/

Low Latency 🐇

Spherical Cow Assumption 🐮

- No compute, straight line, in a vacuum, guaranteed delivery, etc
- 40ms
  - São Paulo ➡️ NYC, Vancouver, Stockholm
  - São Paulo ❌ Sidney, Tokyo, Seoul

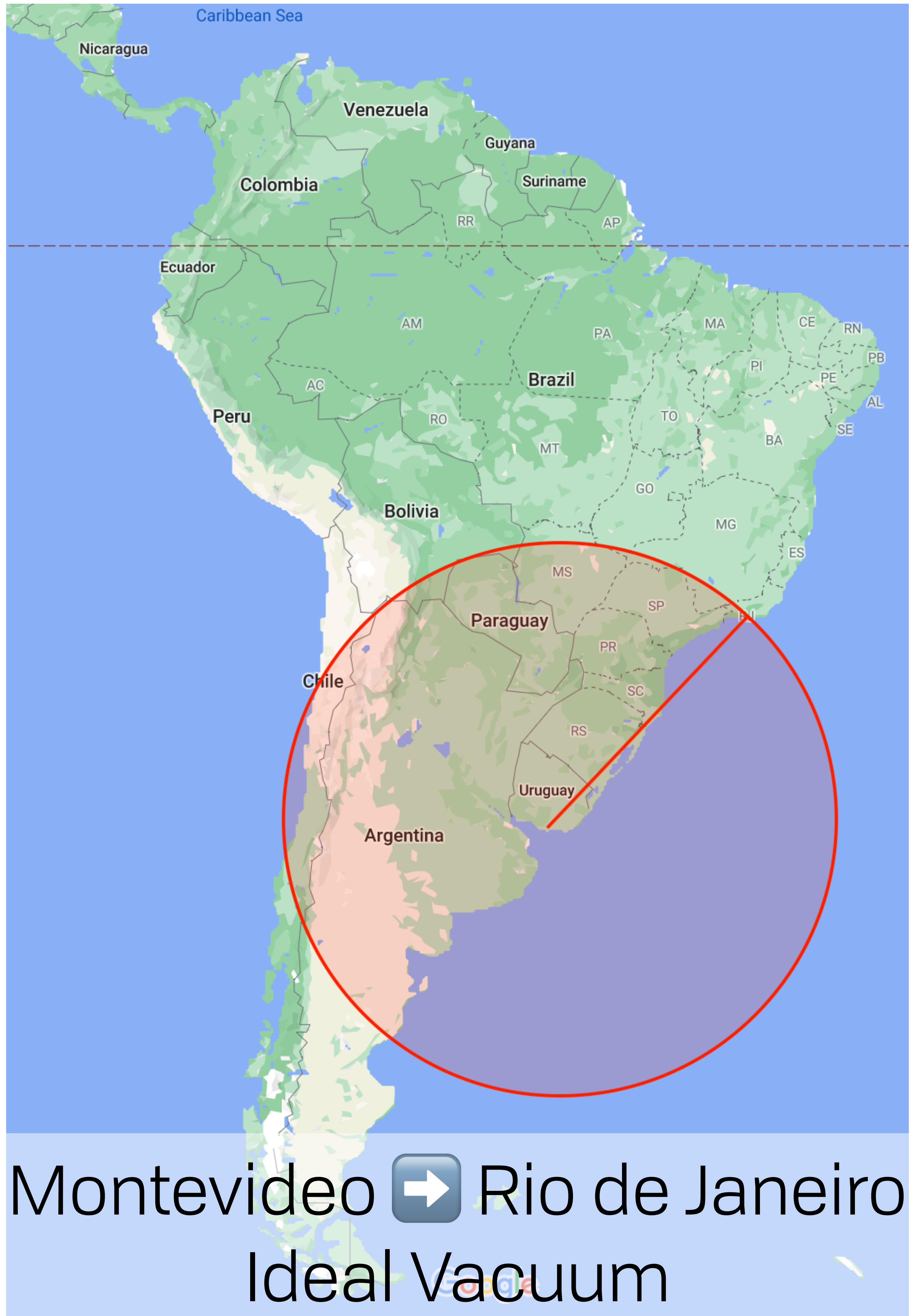
Credit: Keenan Crane
http://www.cs.cmu.edu/~kmcrane/Projects/ModelRepository/

Low Latency 🐇

What 8ms Looks Like

Low Latency 🐇

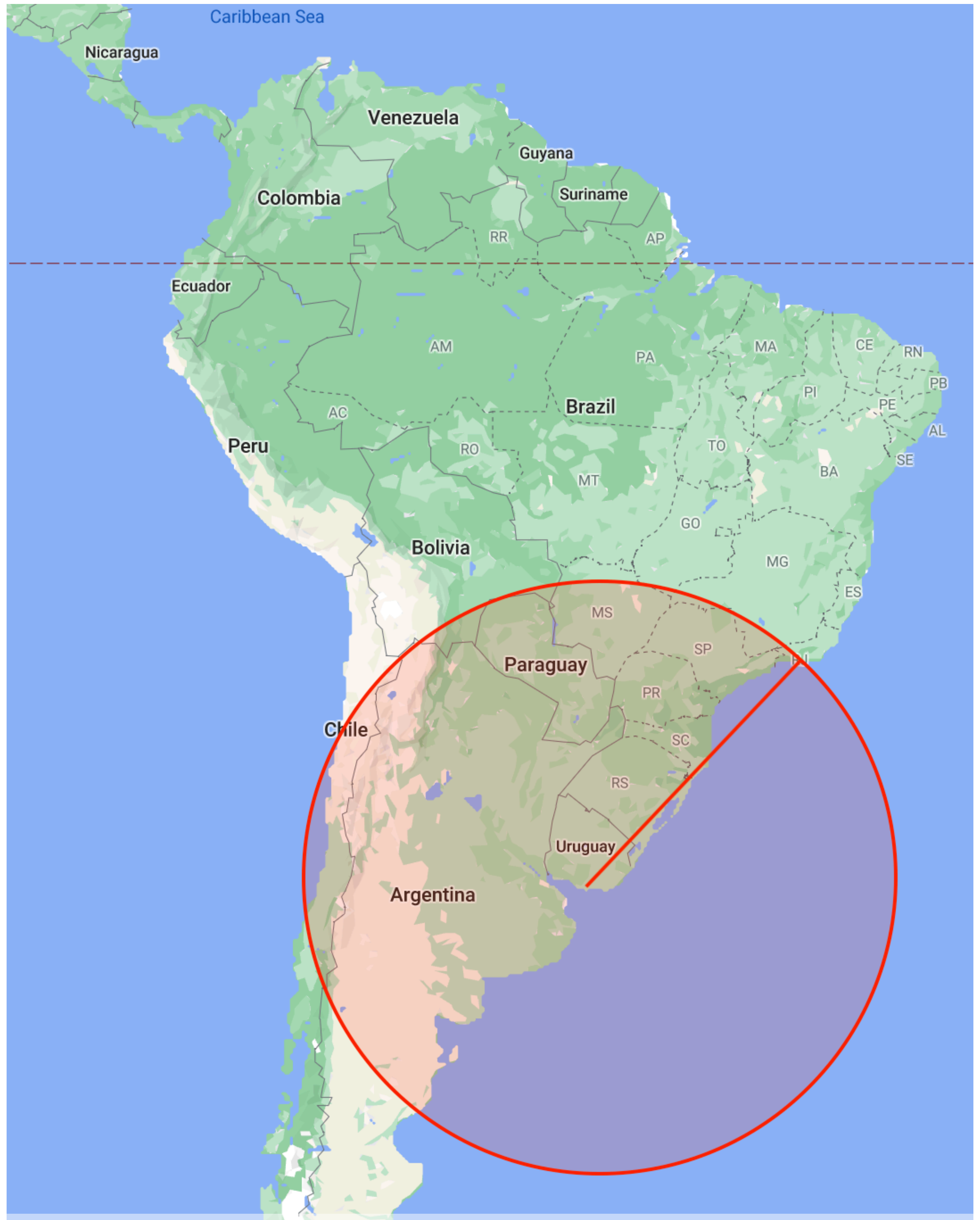
What 8ms Looks Like



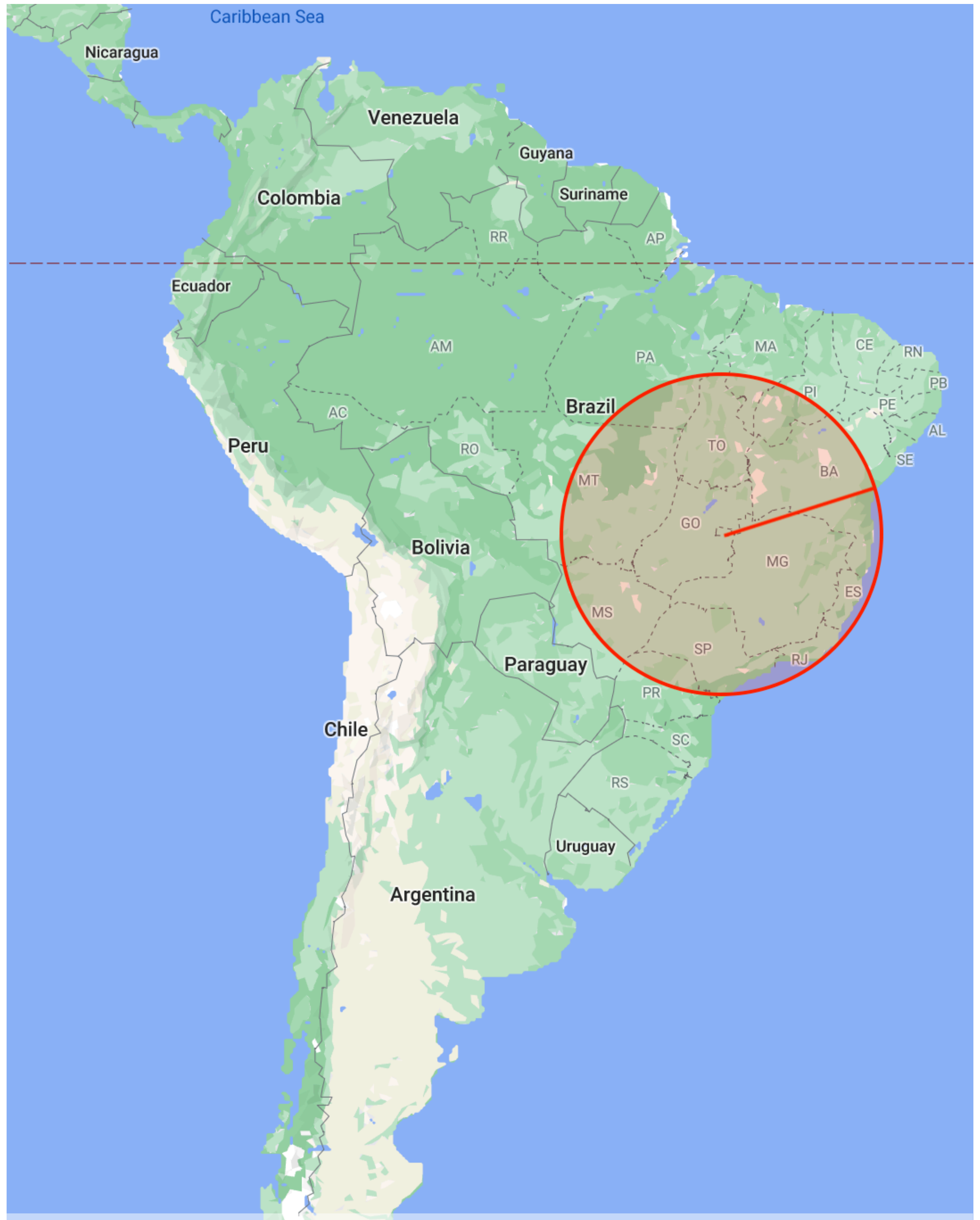
Montevideo ➡️ Rio de Janeiro
Ideal Vacuum

Low Latency 🐇

What 8ms Looks Like



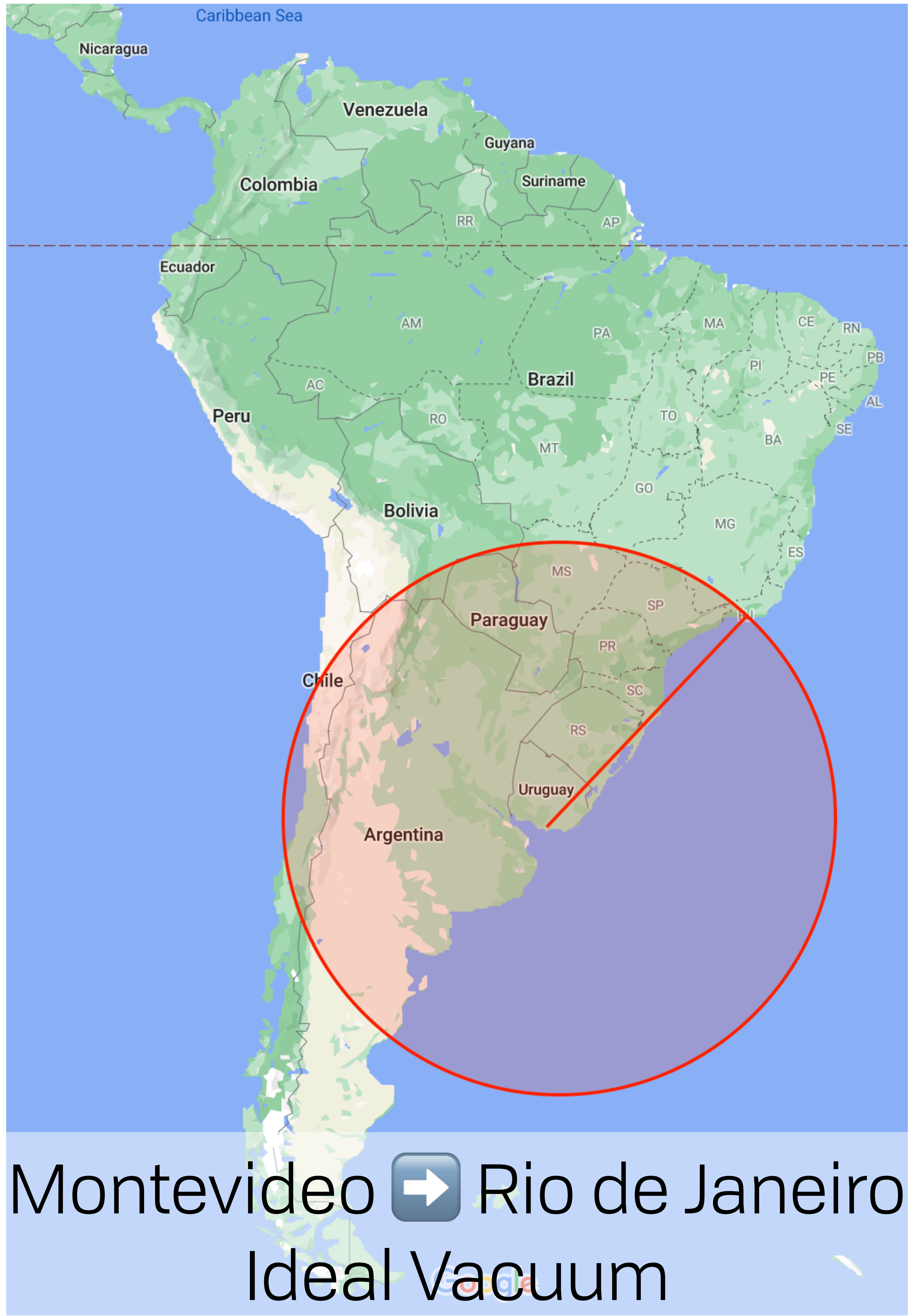
Montevideo ➡️ Rio de Janeiro
Ideal Vacuum



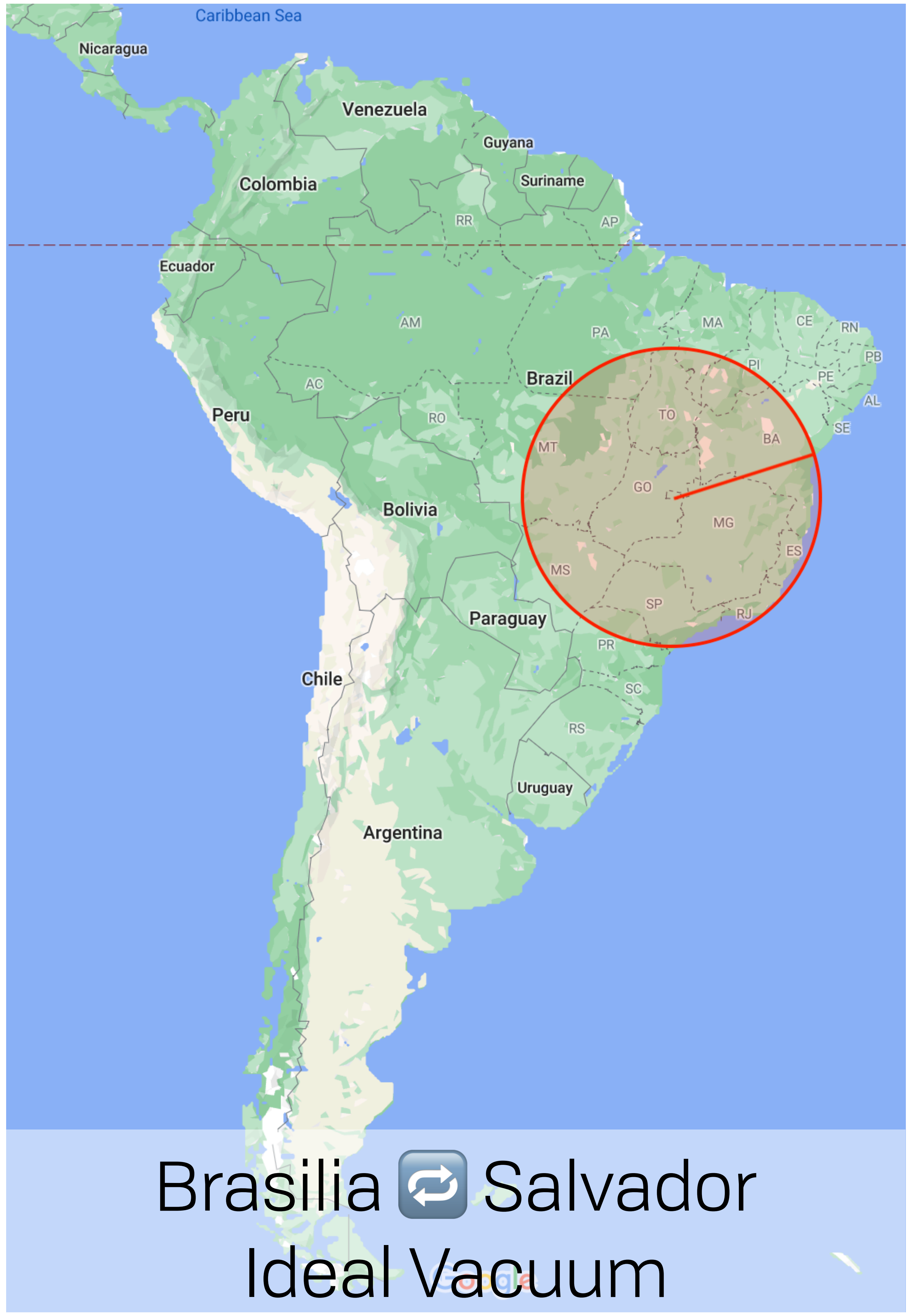
Brasilia 🔁 Salvador
Ideal Vacuum

Low Latency 🐇

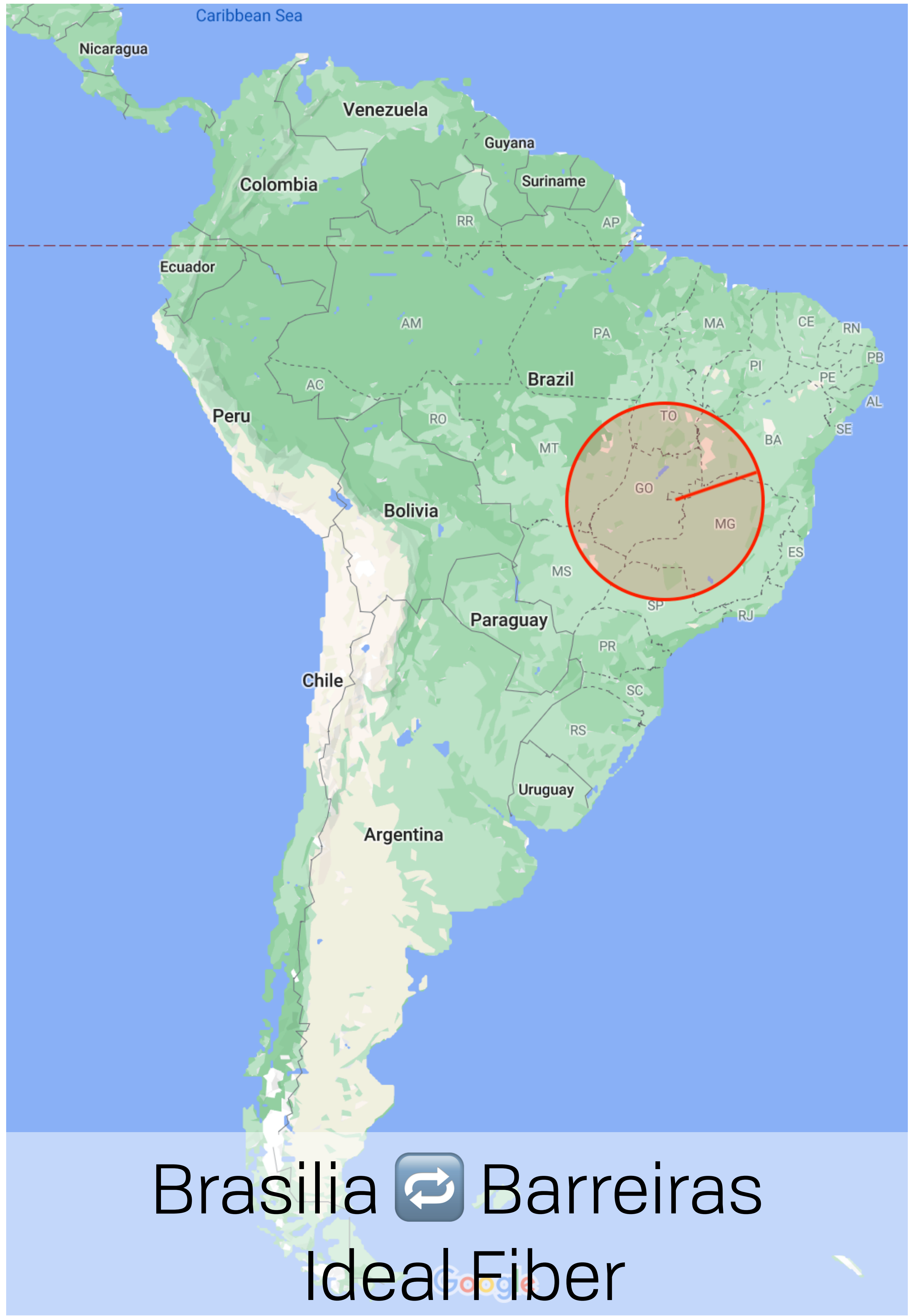
What 8ms Looks Like



Montevideo ➡️ Rio de Janeiro
Ideal Vacuum



Brasilia 🔁 Salvador
Ideal Vacuum



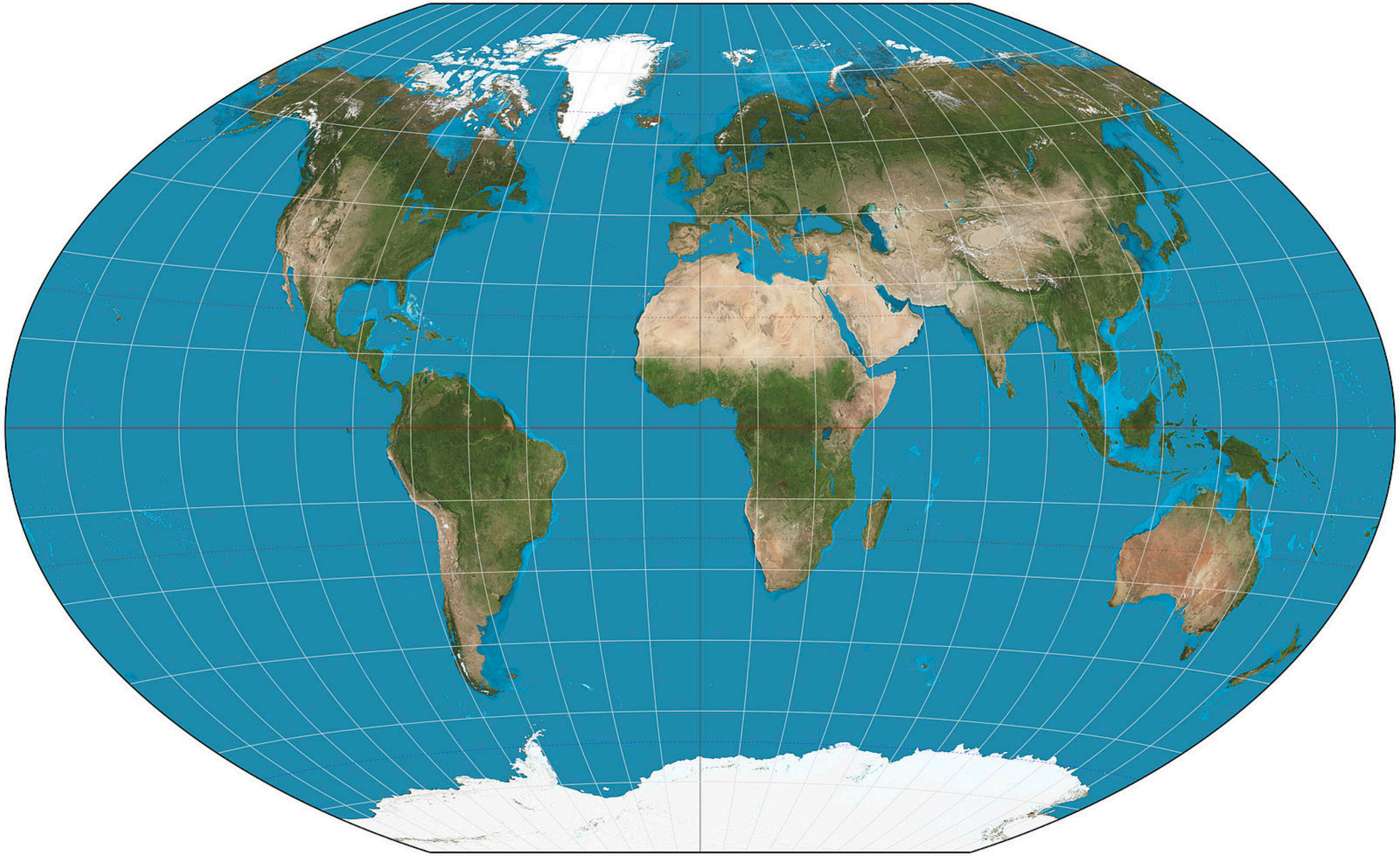
Brasilia 🔁 Barreiras
Ideal Fiber

Low Latency 🐇

Causal Islands 🏖️🏝️

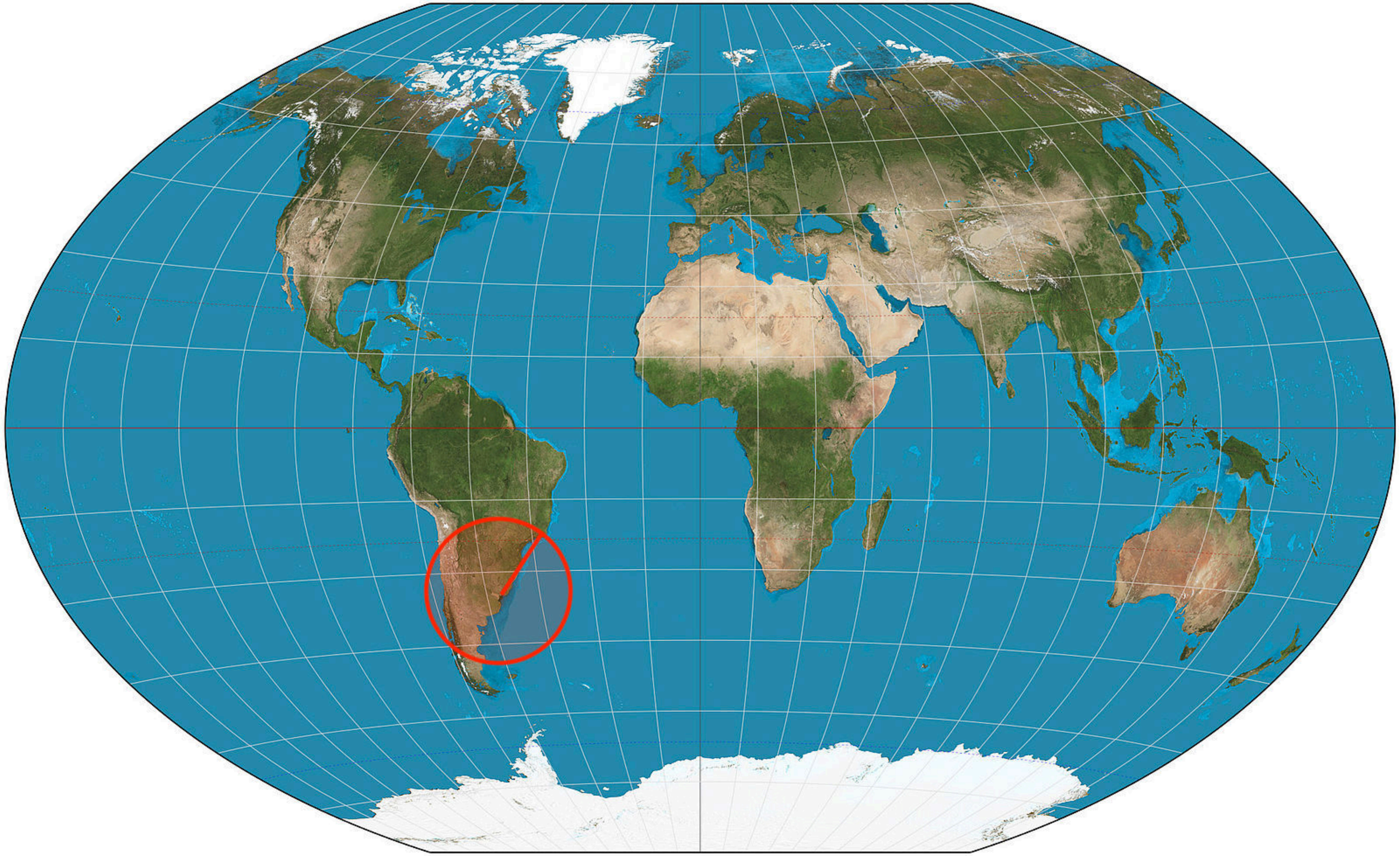
Low Latency 🐇

Causal Islands 🏖️🏝️



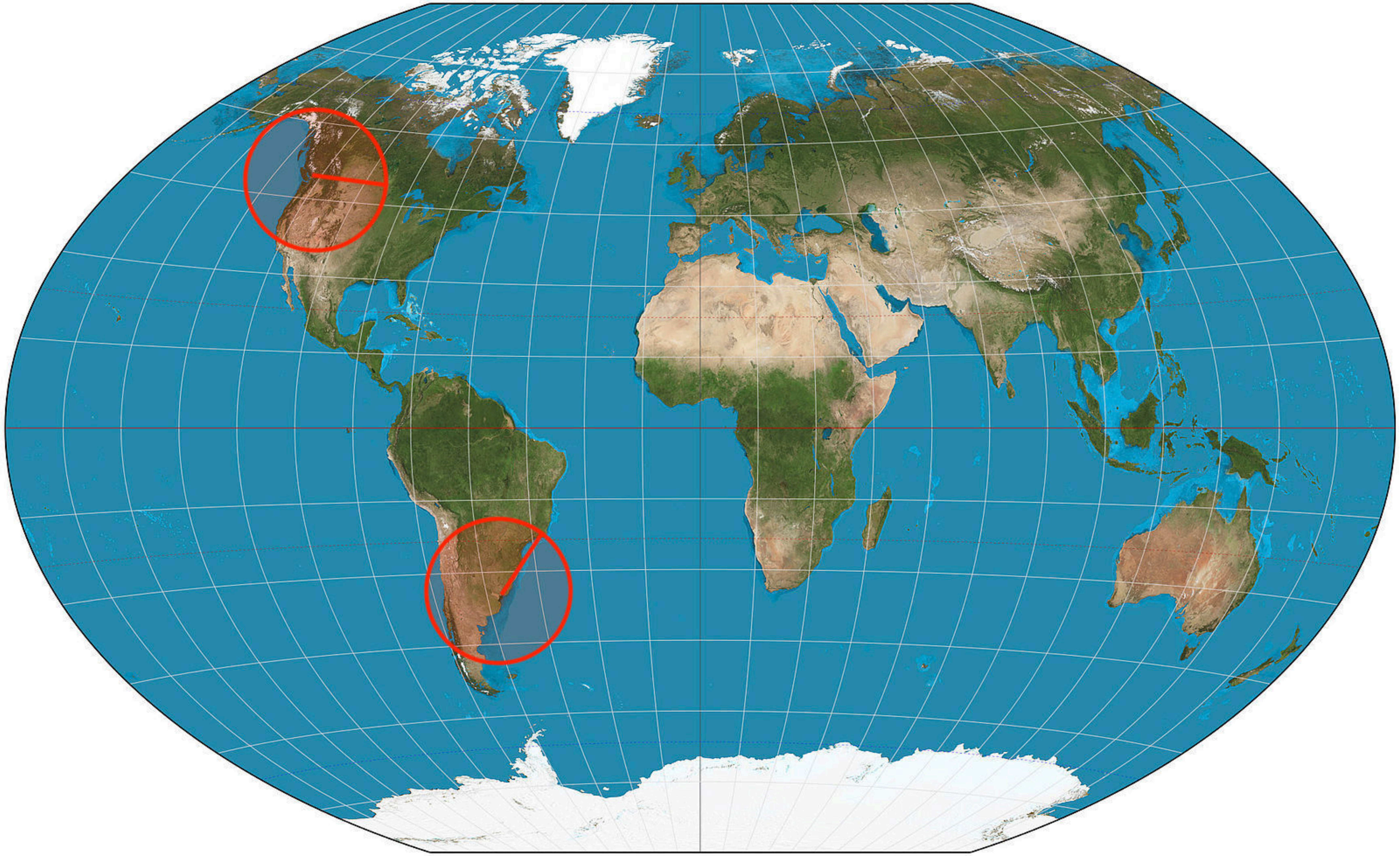
Low Latency 🐇

Causal Islands 🏖🏝



Low Latency 🐇

Causal Islands 🏖️🏝️

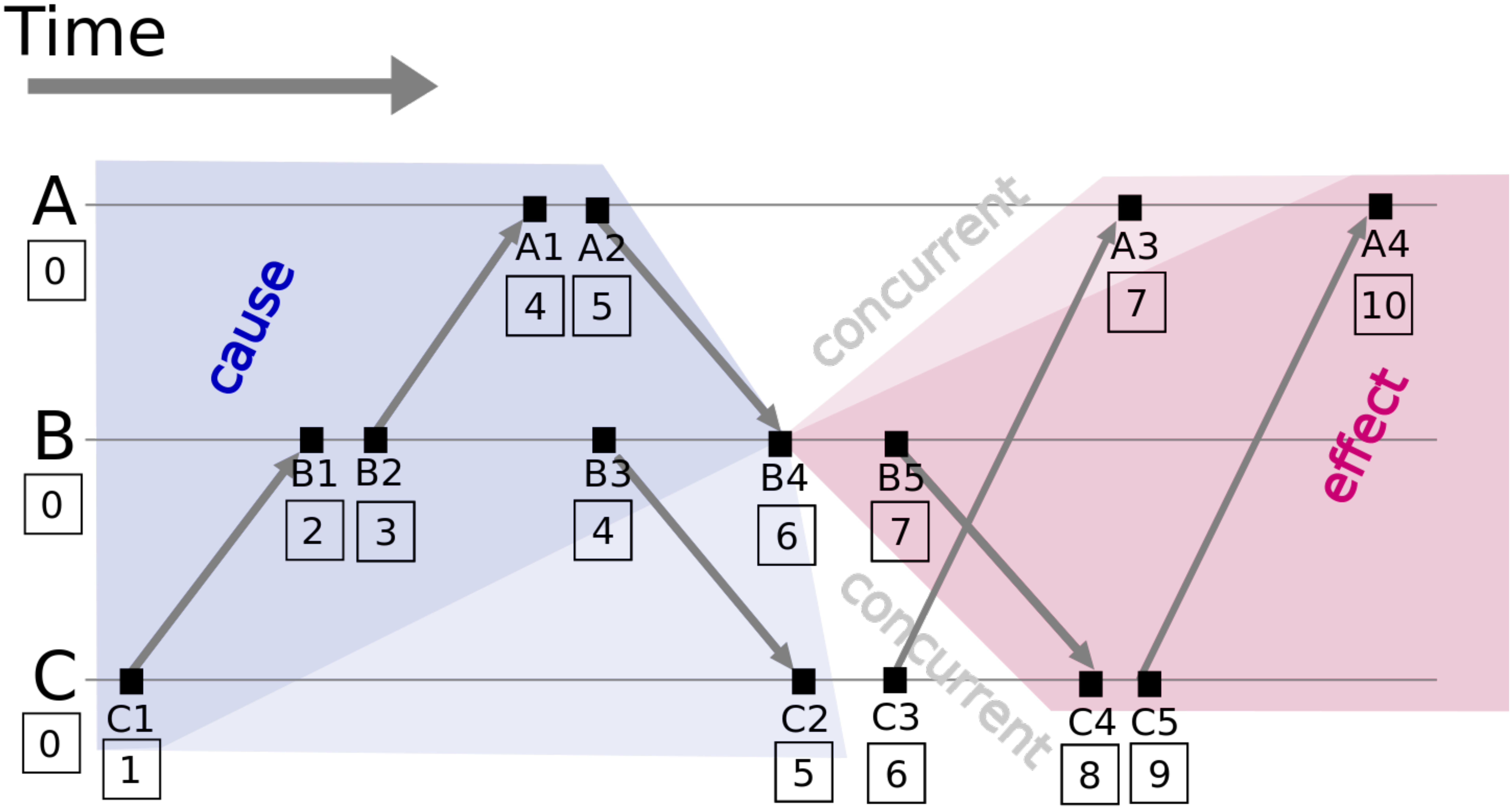


Low Latency 🐇

Light Cone & Relativistic Ordering

Low Latency 🐇

Light Cone & Relativistic Ordering



Source: Duesentrieb via Wikimedia Commons

Turning Up

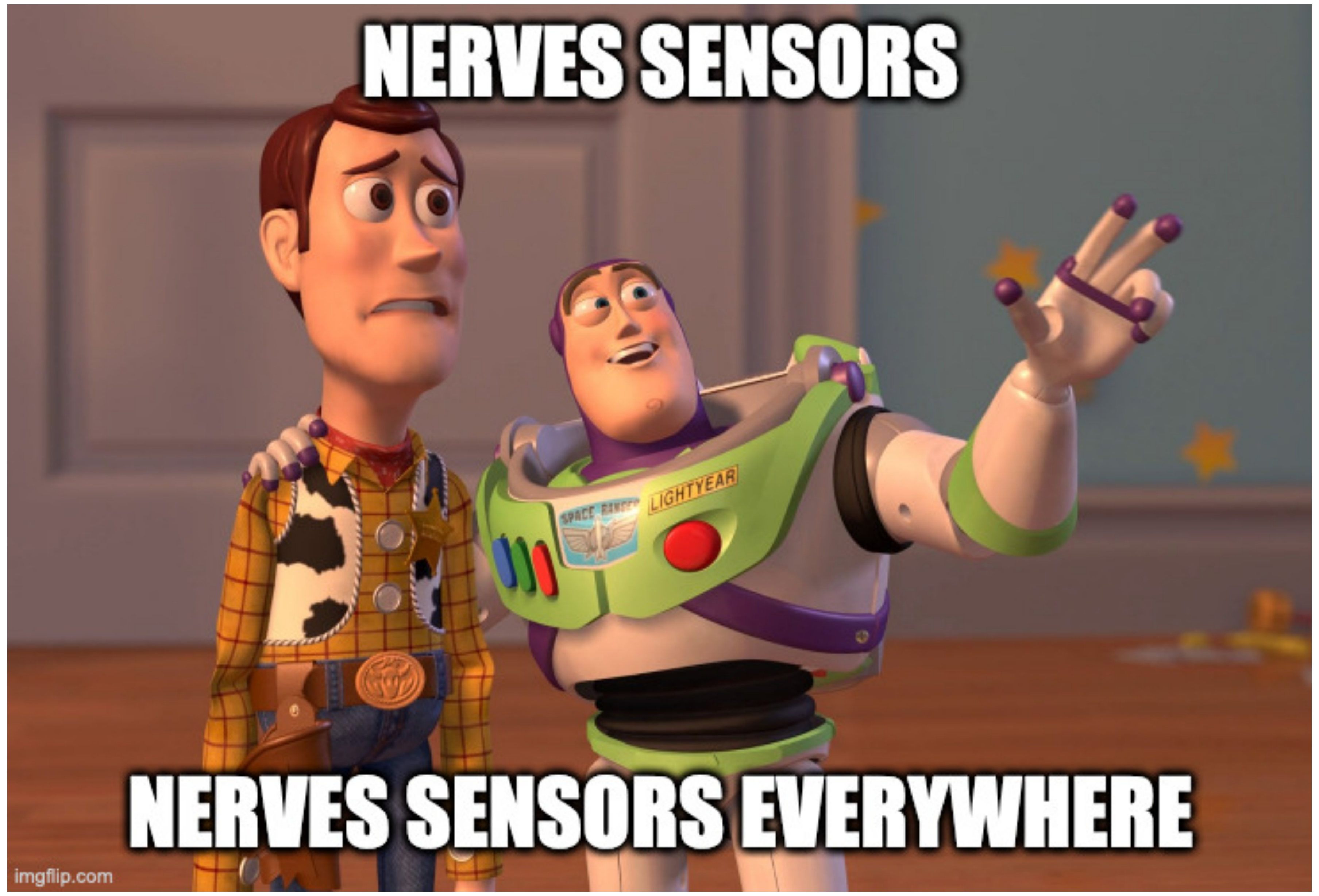
High Volume

🌊

High Volume 🌊

Unprecedented Volume 🦖

- We have high scale NOW? Only more devices & usage in the future!
- Sensors everywhere: IoT devices, continuous health data
- Geospatial data (e.g. autonomous vehicles, XR)



High Volume 🌊

Feedback Cycle

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

Source: Microsoft

China completes world's first 5G remote surgery

Source: YouTube, South China Morning Post

Source: Google & Bungie

Sensor data explosion will kill the cloud.
Sensors will produce massive amounts of data, but the 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 model [...] not unlike the **distributed** computing model

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

~ a16z, “The End of Cloud Computing”

High Volume 🌊

Edge Absorbs Cloud (and MEC)

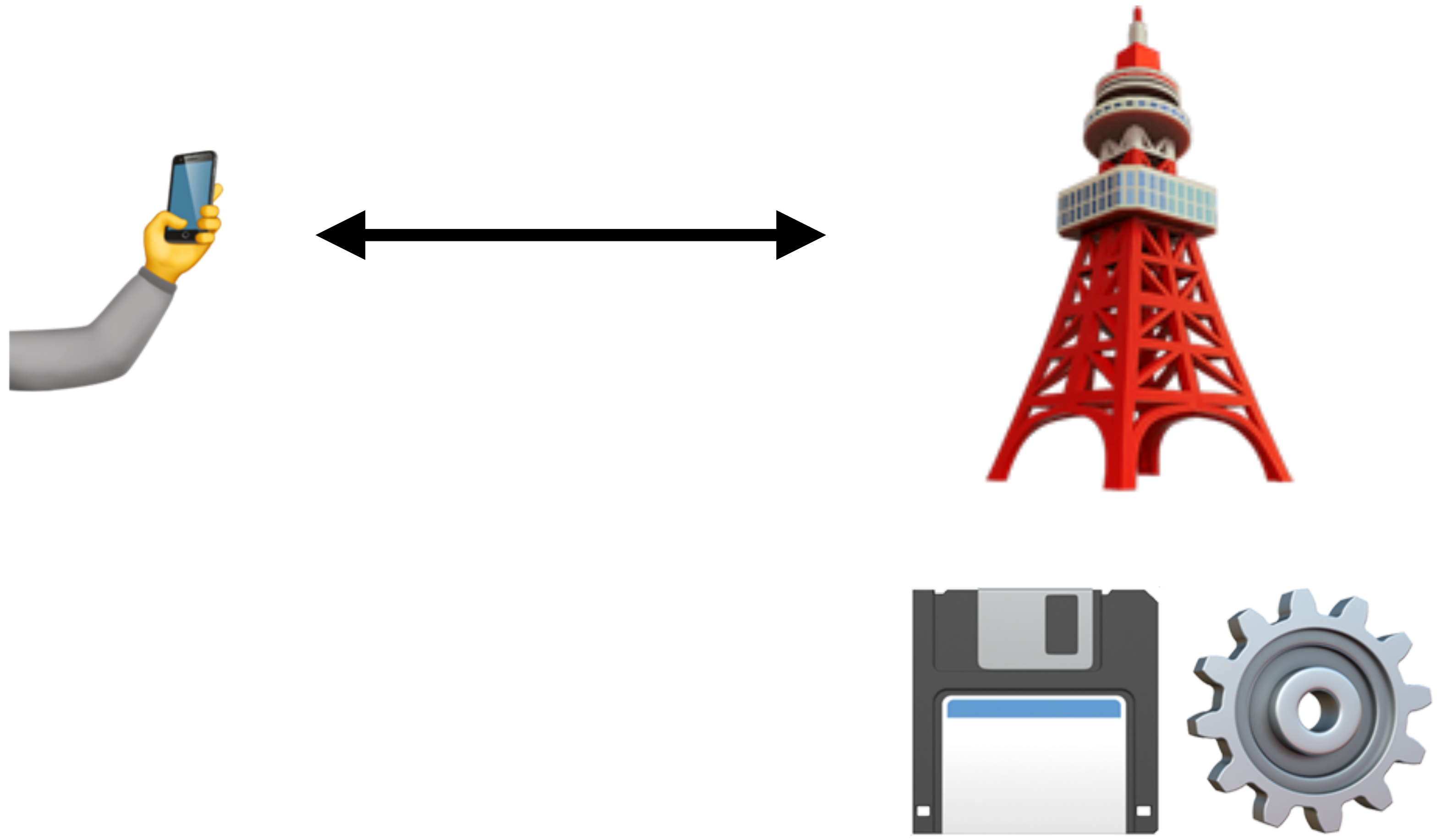
High Volume 🌊

Edge Absorbs Cloud (and MEC)

🤳

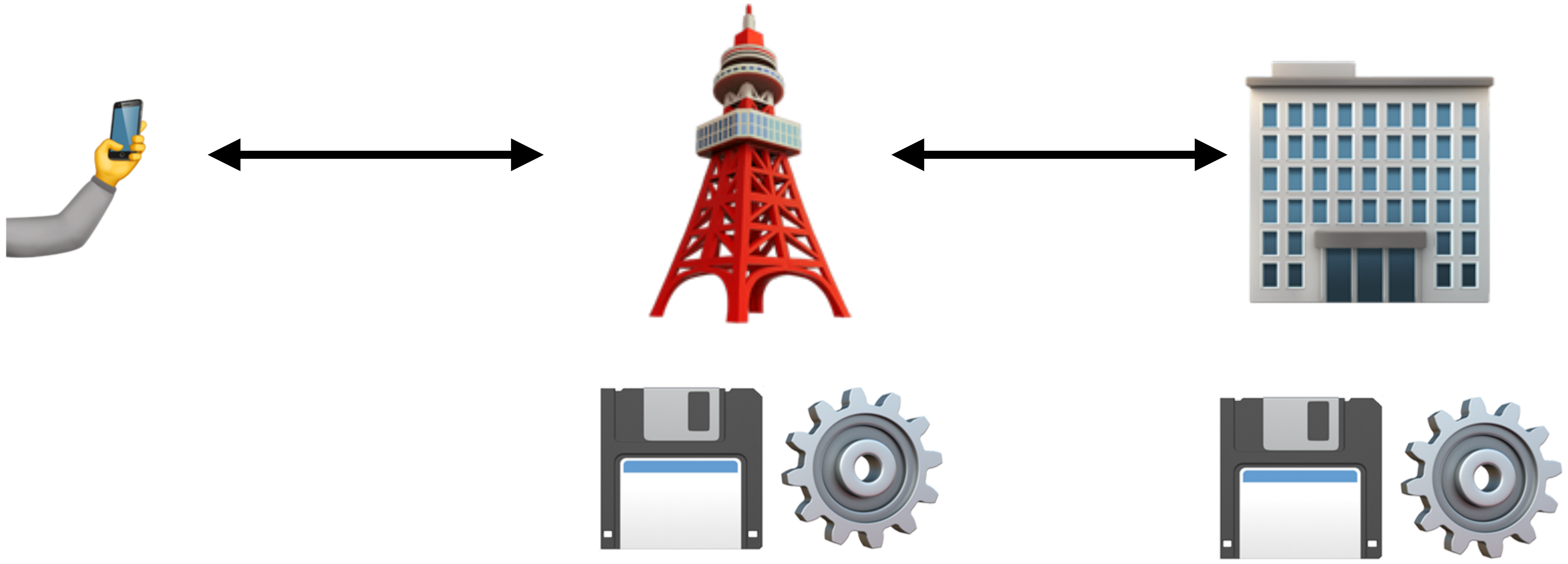
High Volume 🌊

Edge Absorbs Cloud (and MEC)



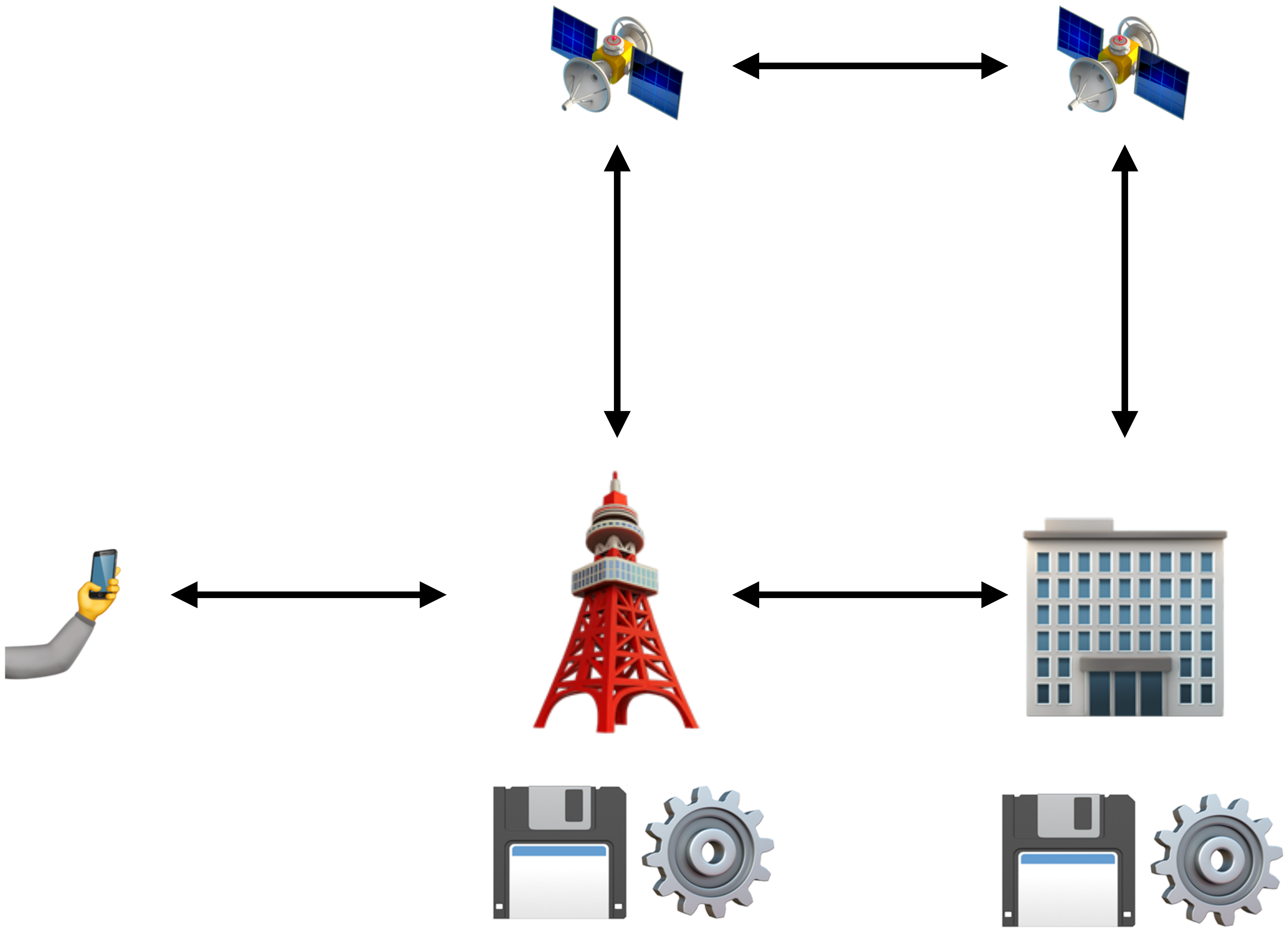
High Volume 🌊

Edge Absorbs Cloud (and MEC)



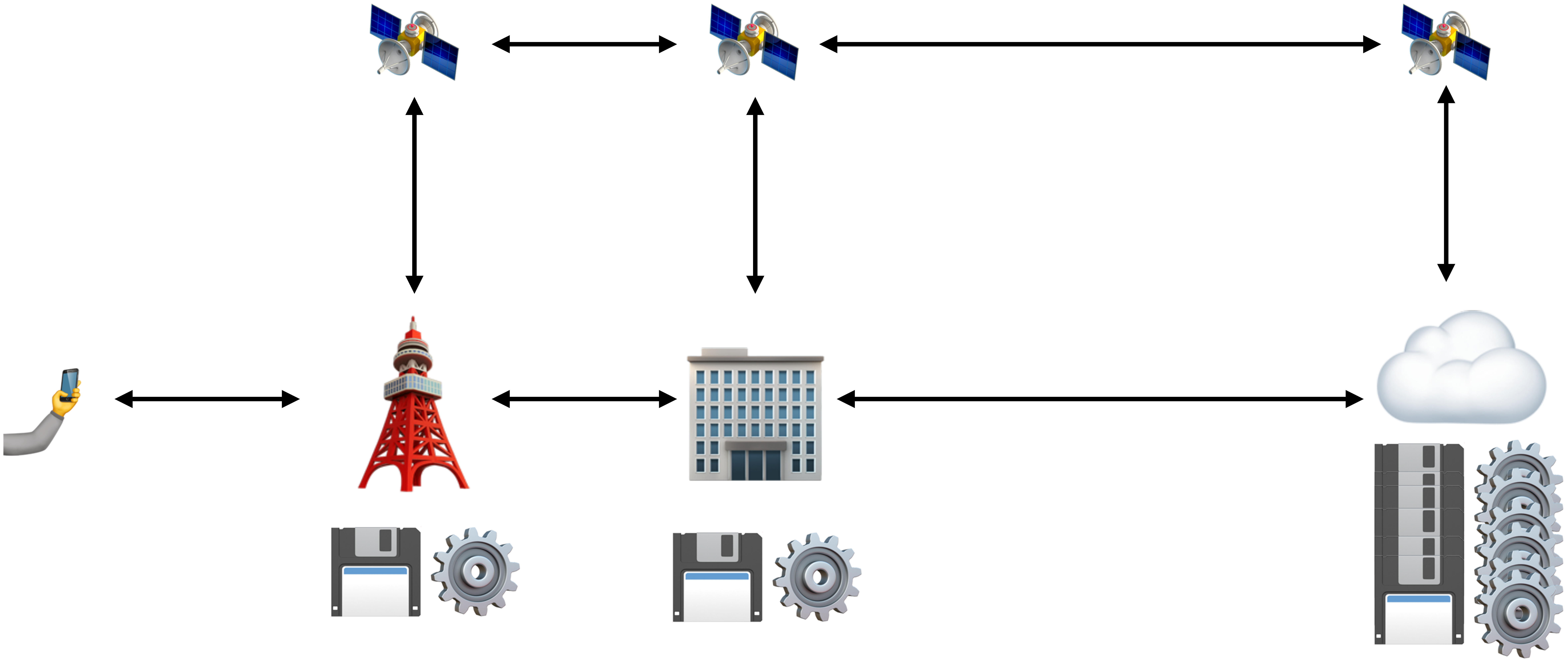
High Volume 🌊

Edge Absorbs Cloud (and MEC)



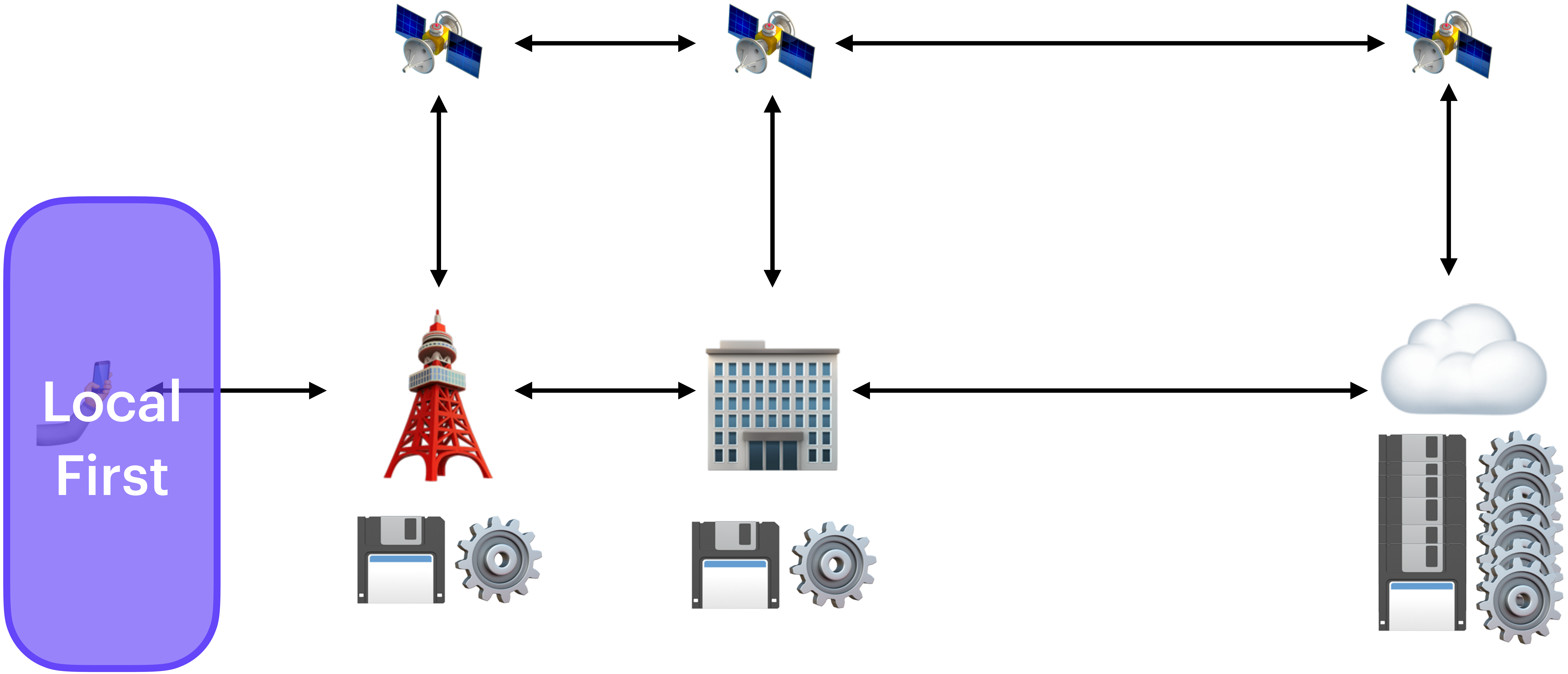
High Volume 🌊

Edge Absorbs Cloud (and MEC)



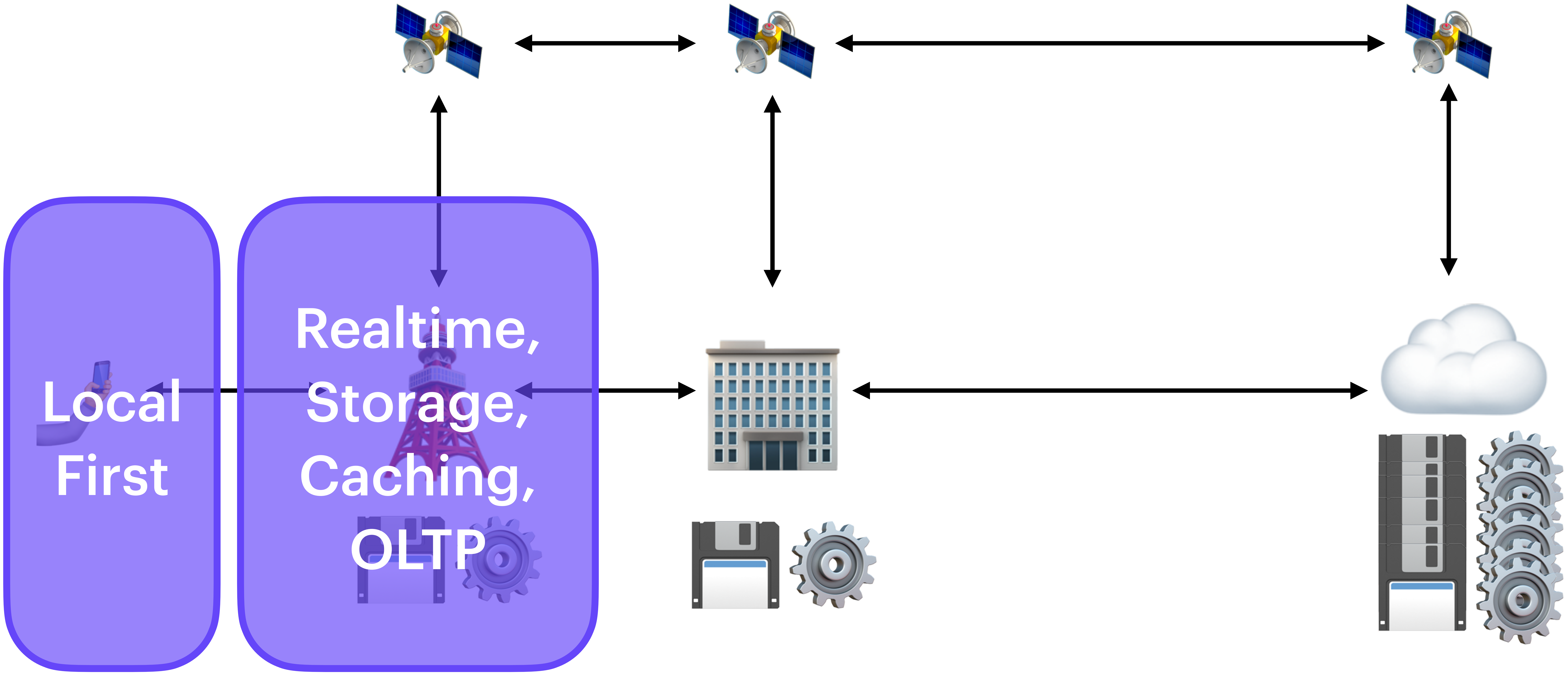
High Volume 🌊

Edge Absorbs Cloud (and MEC)



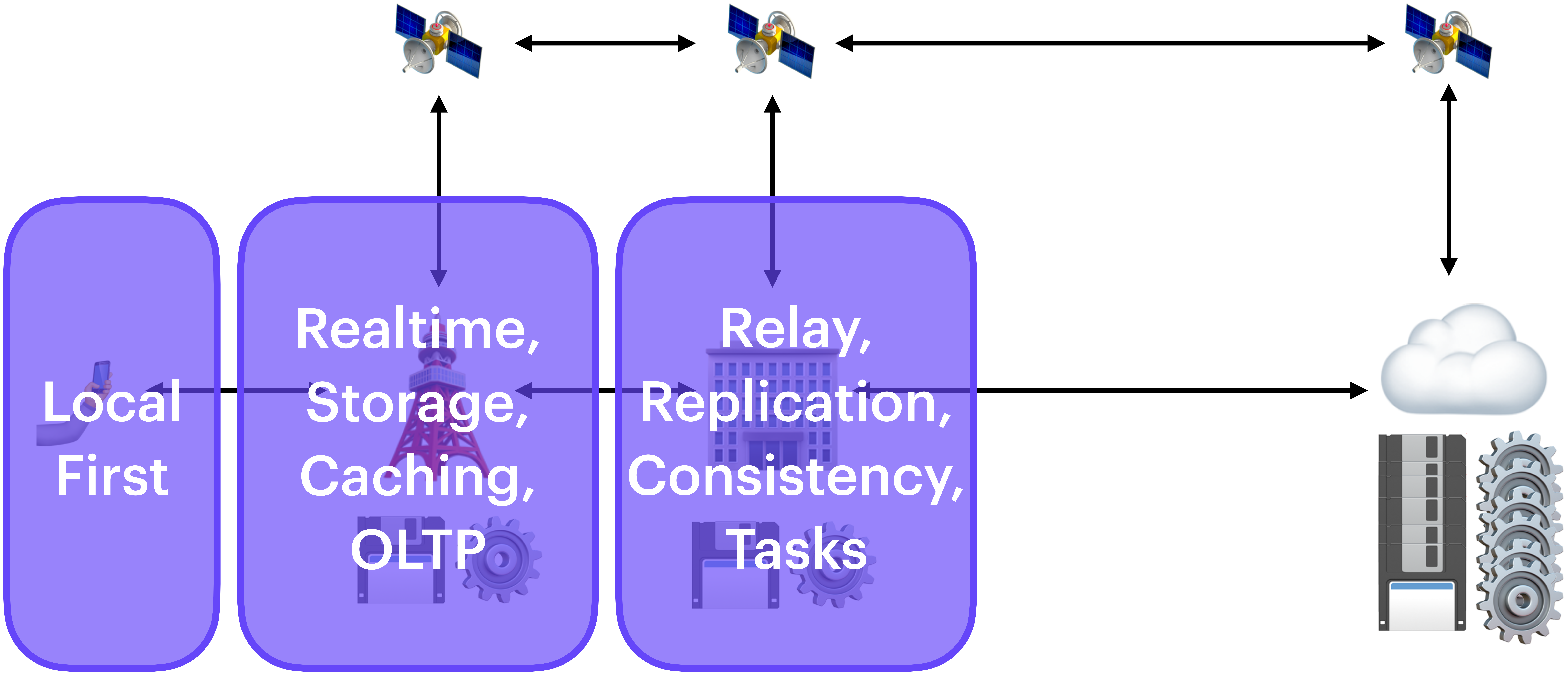
High Volume 🌊

Edge Absorbs Cloud (and MEC)



High Volume 🌊

Edge Absorbs Cloud (and MEC)



Local First
Realtime, Storage, Caching, OLTP
Relay, Replication, Consistency, Tasks

High Volume 🌊

Edge Absorbs Cloud (and MEC)

Local First

Realtime, Storage, Caching, OLTP

Relay, Replication, Consistency, Tasks

Aggregation, Batching, Training, OLAP

What does this all mean?

Consequence

🛸

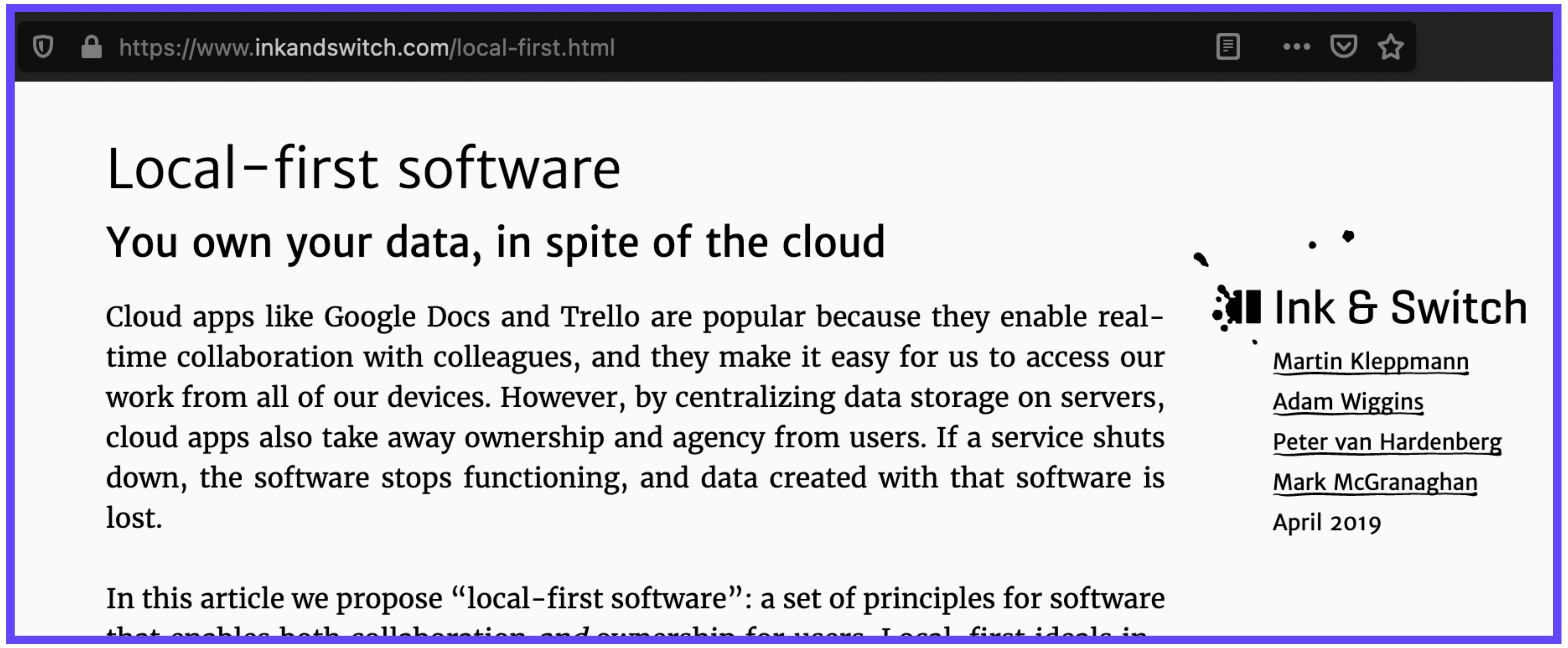
Consequence 🛸

New Assumptions, New Approach

Consequence 🛸

New Assumptions, New Approach

- New features naturally fall out of the architecture
- Recognize that we're increasingly connected/networked
- Local-first means network efficient (in the normal case)
- Data can run anywhere = commons networks



https://www.inkandswitch.com/local-first.html

Local-first software

You own your data, in spite of the cloud

Ink & Switch

Martin Kleppmann
Adam Wiggins
Peter van Hardenberg
Mark McGranaghan
April 2019

Cloud apps like Google Docs and Trello are popular because they enable real-time collaboration with colleagues, and they make it easy for us to access our work from all of our devices. However, by centralizing data storage on servers, cloud apps also take away ownership and agency from users. If a service shuts down, the software stops functioning, and data created with that software is lost.

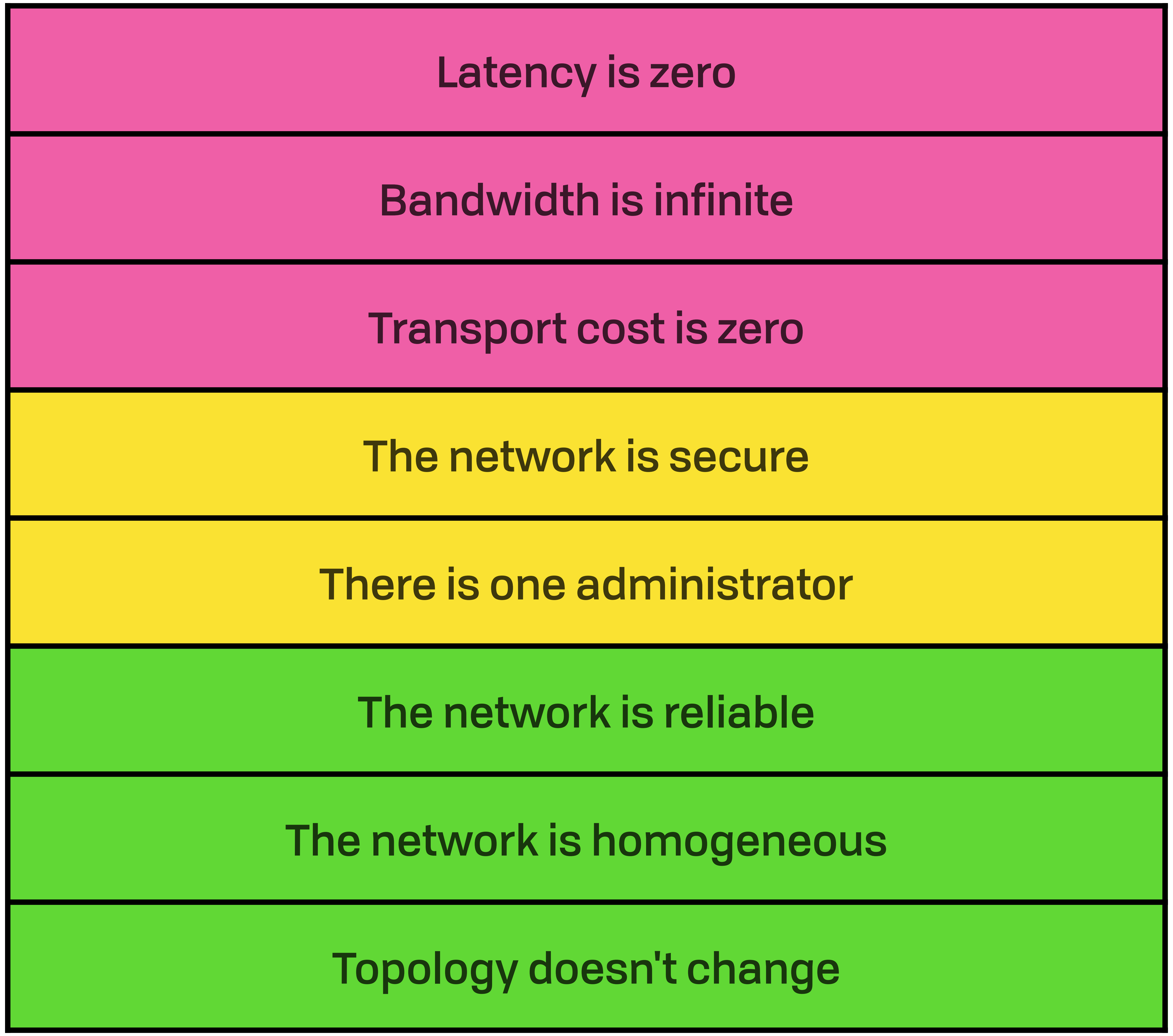
In this article we propose "local-first software": a set of principles for software that enables both collaboration *and* ownership for users. Local-first ideals in

Consequence 🛸

Tackling the Fallacies

Consequence 🛸

Tackling the Fallacies



- Latency is zero
- Bandwidth is infinite
- Transport cost is zero
- The network is secure
- There is one administrator
- The network is reliable
- The network is homogeneous
- Topology doesn't change

Consequence 🛸

Tackling the Fallacies

Latency is zero

Bandwidth is infinite

Transport cost is zero

The network is secure

There is one administrator

The network is reliable

The network is homogeneous

Topology doesn't change

We need to handle 100% of these up front

Consequence 🛸

Tackling the Fallacies

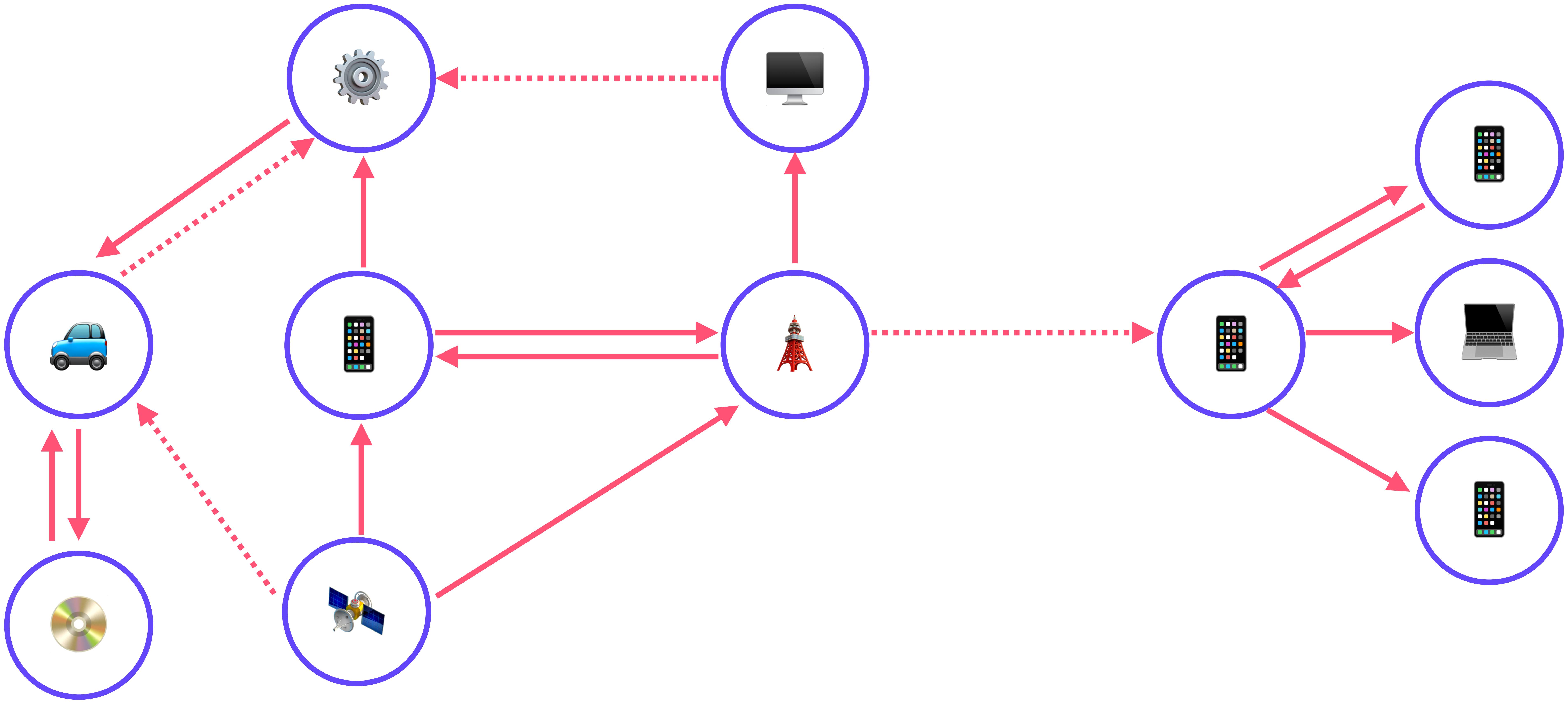
Latency is zero	Treat latency directly (speed of causality) Treat (order of causality / relativistic)
Bandwidth is infinite	Apps continue to work with zero bandwidth Only push when & what needed
Transport cost is zero	Minimize network use
The network is secure	Assume that the pipes are broken Direct access control
There is one administrator	Fine grained, delegate capabilities (OCAP)
The network is reliable	Time, delivery, & order independence
The network is homogeneous	Device agnostic
Topology doesn't change	atomic unit is the edge device (same like the atomic unit is the actor)

Consequence 🛸

Giving Up Topological Control

Consequence 🛸

Giving Up Topological Control



Consequence 🛸

Data, Data, Data 💾

Consequence 🛸

Data, Data, Data 💾

- Only UI & data are essential

Consequence 🛸

Data, Data, Data 💾

- Only UI & data are essential
- New primitives
  - Consistency (CRDTs, STM, Distributed Datalog)
  - State transfer ➡️ state synchronization ➡️ state views

Consequence 🛸

Data, Data, Data 💾

- Only UI & data are essential
- New primitives
  - Consistency (CRDTs, STM, Distributed Datalog)
  - State transfer ➡️ state synchronization ➡️ state views
- Access control needs to be inherent
  - OCAP & CBC methods (AKA cryptography)

Part II

On the Edge

🧗‍♀️

On the Edge 🧗‍♀️

Why Functional Programming

- Data-oriented
- Pure functions on data is just data
- Shared nothing architectures
- Immutability, easy concurrency
- Manage complexity by being declarative
    - What > how
    - Data > process

On the Edge 🧗‍♀️

Why the BEAM Specifically

- Low conceptual distance from actor model to OCAP
- Community experience with distributed systems
- Used to building up complexity from simple parts
- We're already using a bunch of this!
  - e.g. Phoenix Presence 👉 👉 👉

What's special about Phoenix's implementation is 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**
- relies entirely on the standard library with no operational dependencies
- **self heals**

~ Chris McCord, "What Makes Phoenix Presence Special"

What if we turn Phoenix Live View

Upside Down?

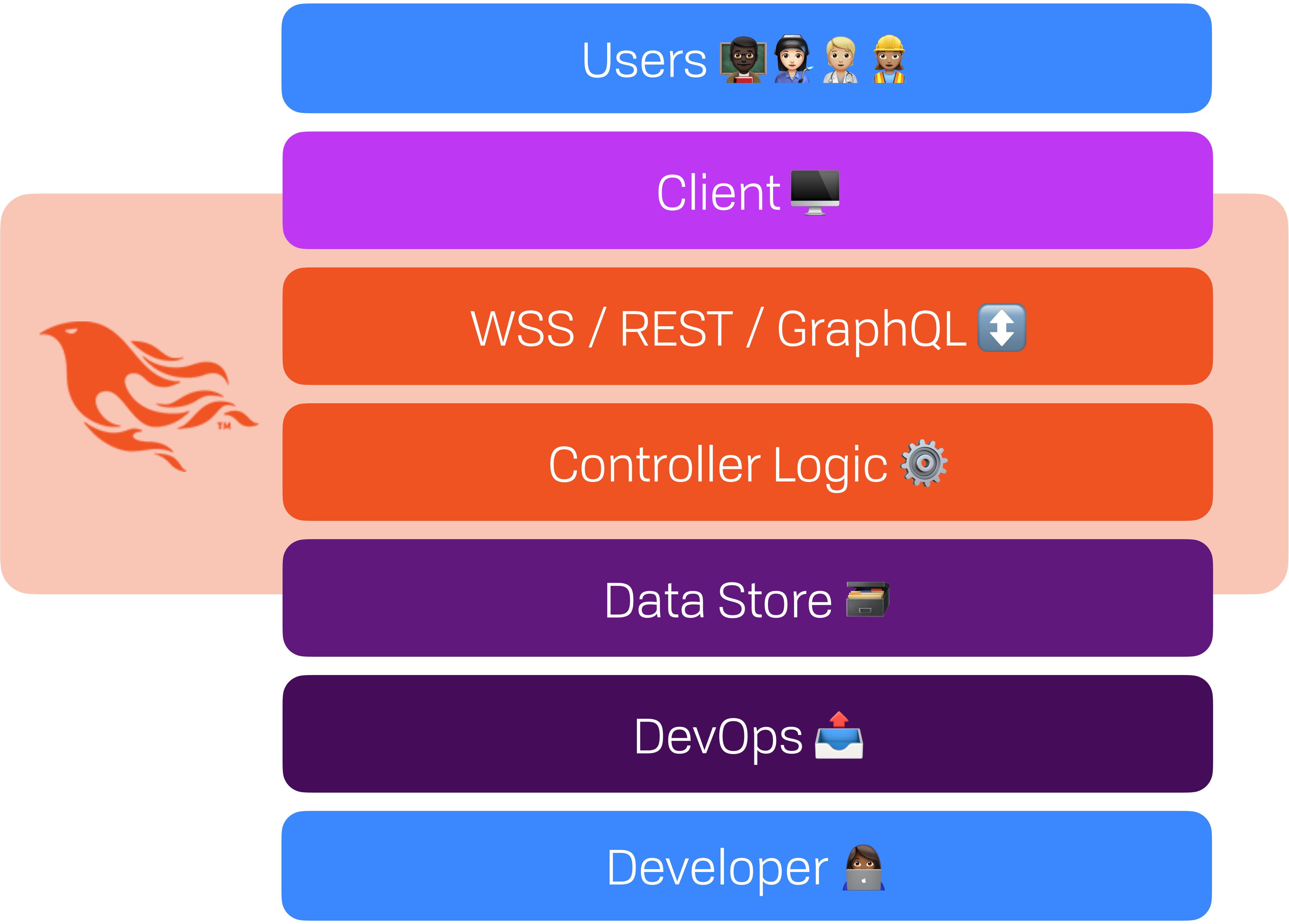
🔁

On the Edge 🧗‍♀️

Phoenix LiveView

On the Edge 🧗‍♀️

Phoenix LiveView



Users 👩🏿‍🏫👩🏻‍✈️👨🏼‍⚕️👷🏽‍♀️

Client 🖥️

WSS / REST / GraphQL ↕️

Controller Logic ⚙️

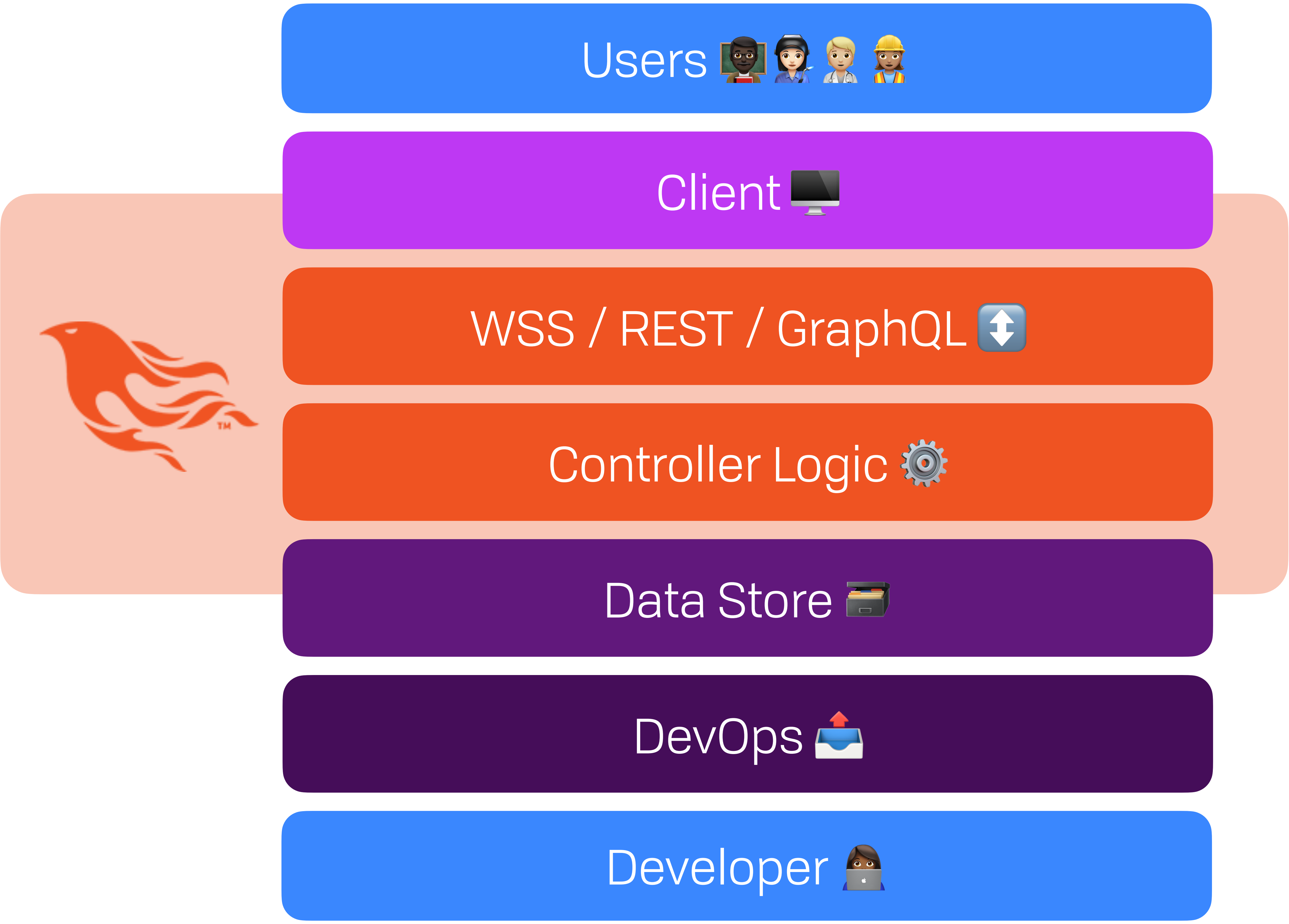
Data Store 🗃️

DevOps 📤

Developer 👩🏾‍💻

On the Edge 🧗‍♀️

Phoenix LiveView



Users 👩🏿‍🏫👩🏻‍✈️👨🏼‍⚕️👷🏽‍♀️

Client 🖥️

WSS / REST / GraphQL ↕️

Controller Logic ⚙️

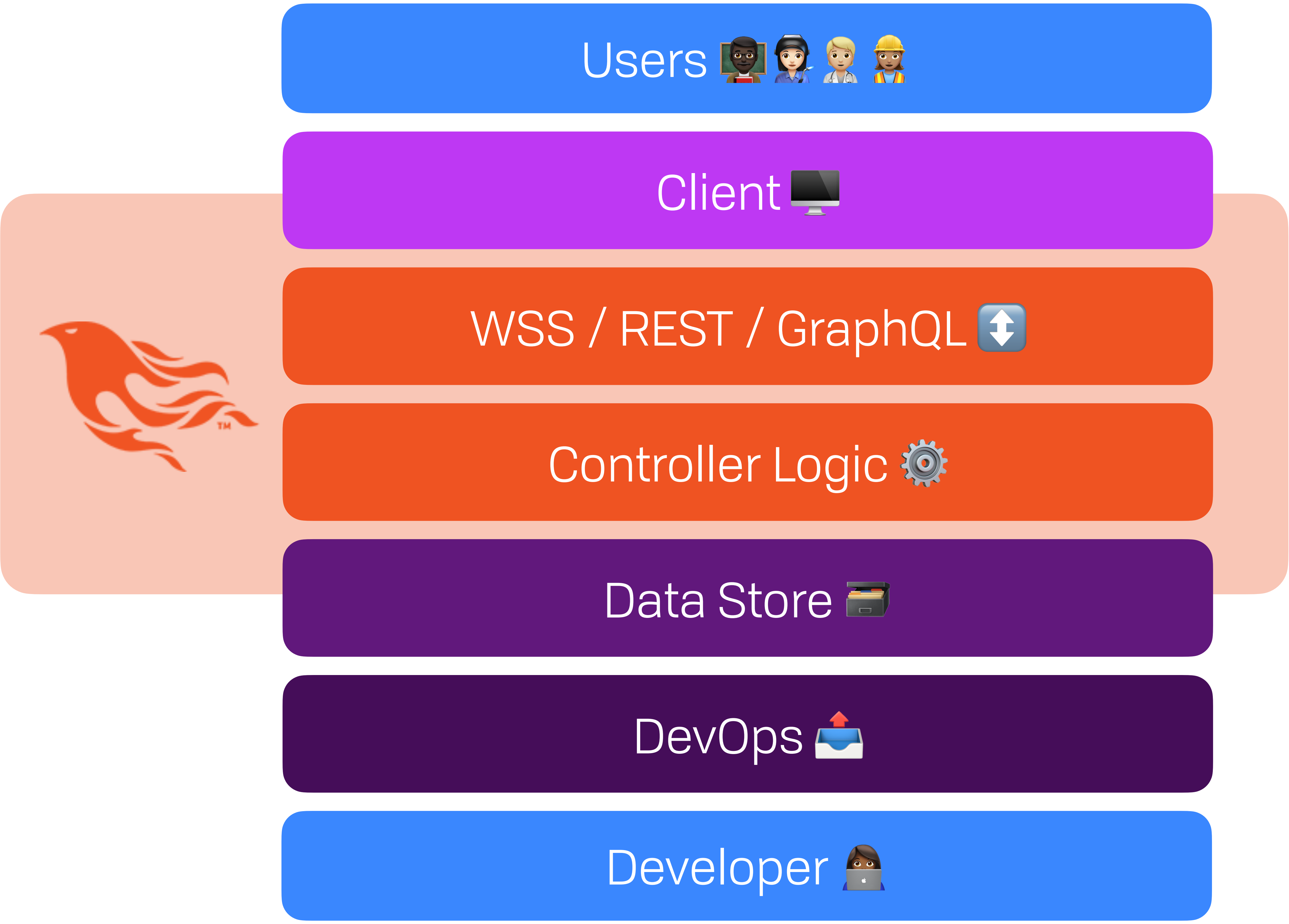
Data Store 🗃️

DevOps 📤

Developer 👩🏾‍💻

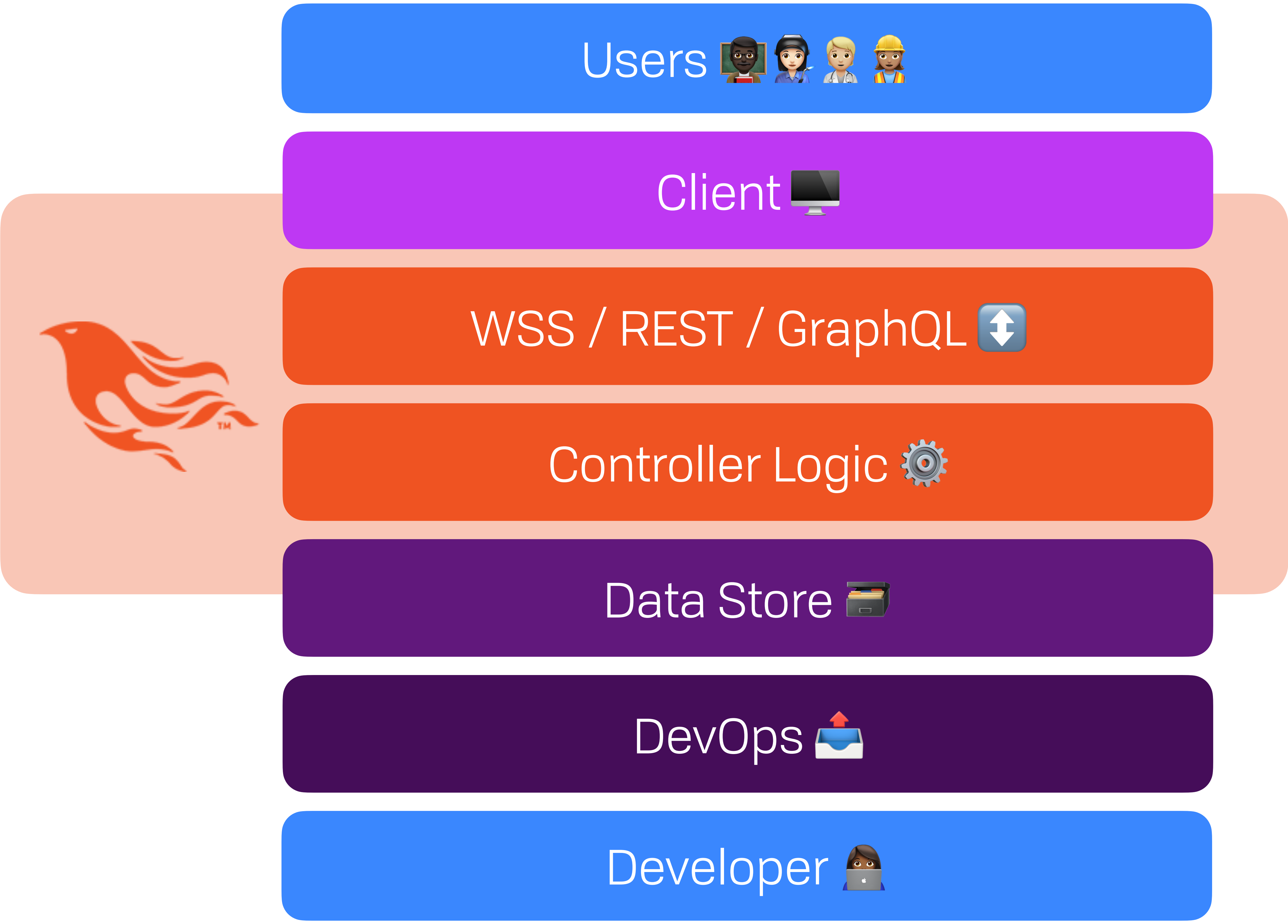
On the Edge 🧗‍♀️

Phoenix LiveView



On the Edge 🧗

Phoenix LiveView



Users 🧑🏿‍🏫👩🏻‍✈️🧑🏼‍⚕️👷🏽

Client 🖥️

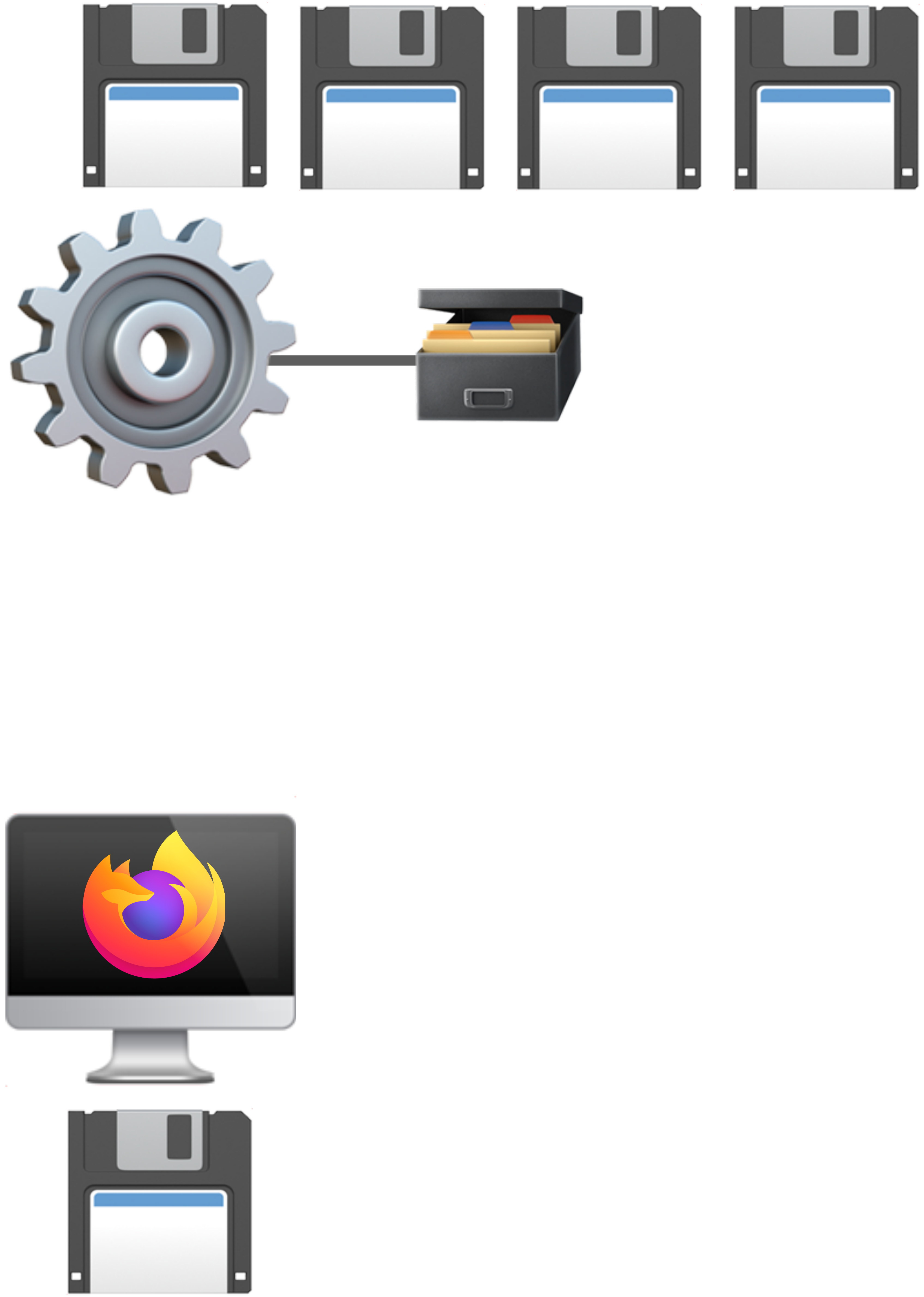
WSS / REST / GraphQL ↕️

Controller Logic ⚙️

Data Store 🗃️

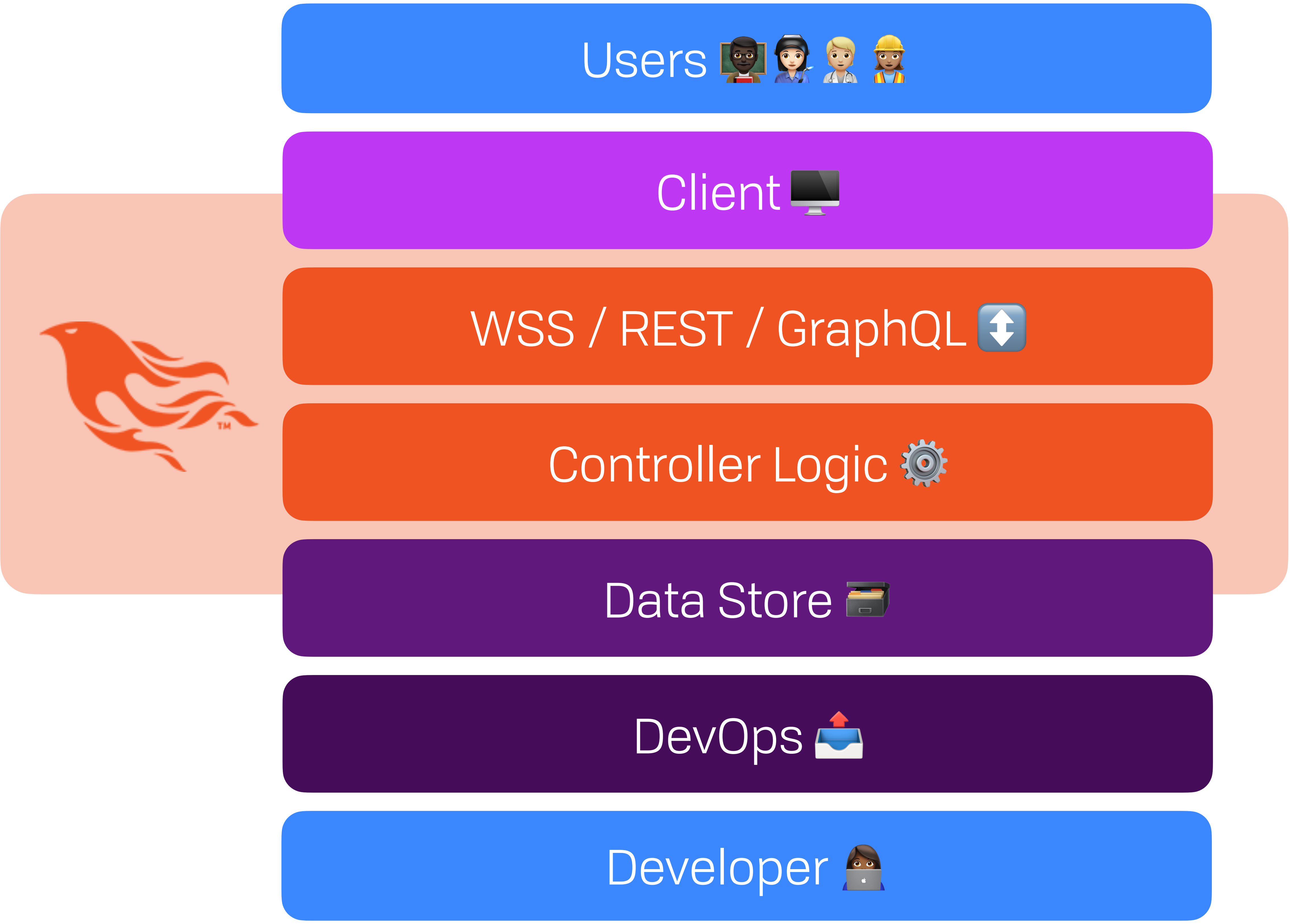
DevOps 📤

Developer 🧑🏽‍💻



On the Edge 🧗‍♀️

Phoenix LiveView



Users 👩🏿‍🏫 👩🏻‍✈️ 👨🏼‍⚕️ 👷🏽‍♀️

Client 🖥️

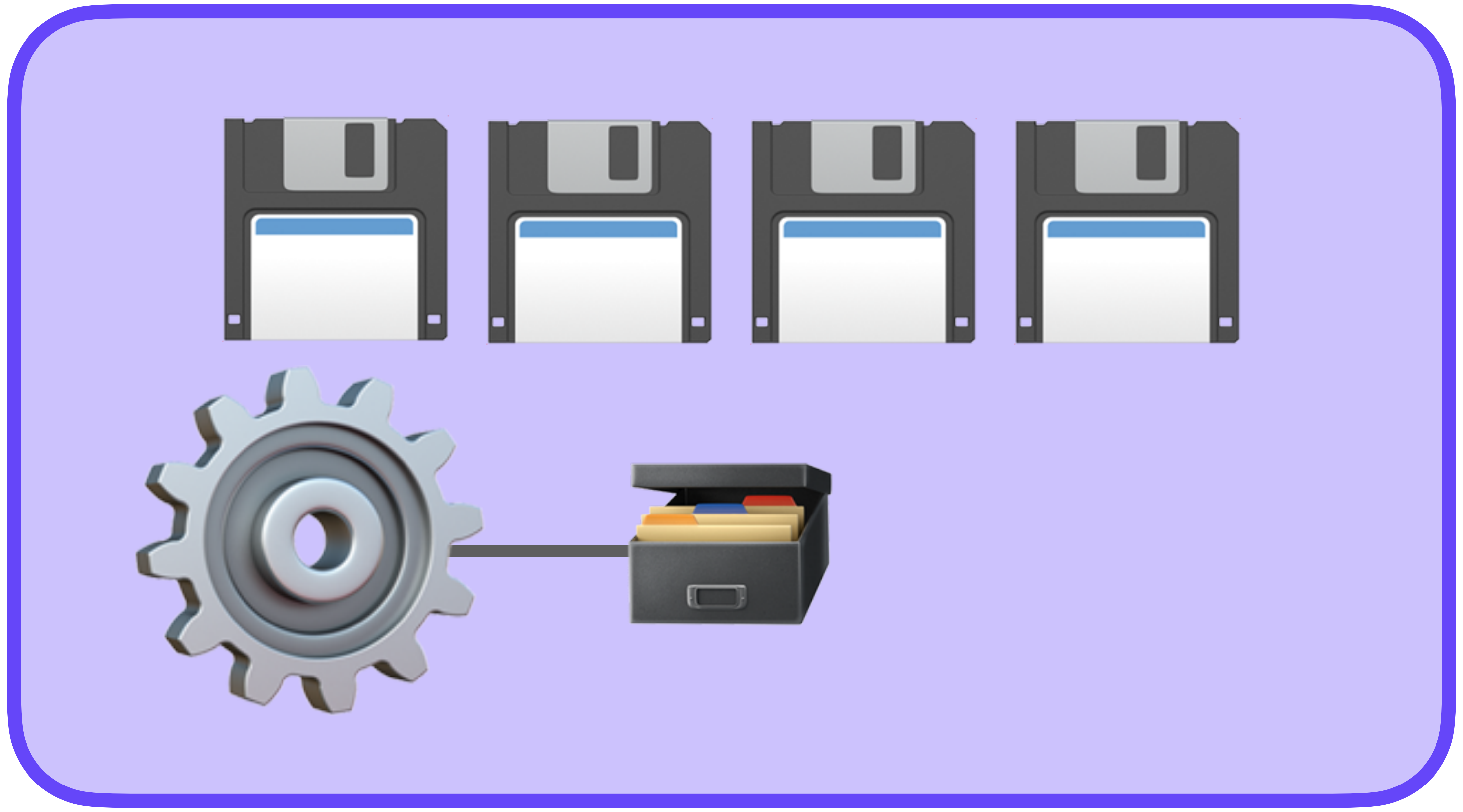
WSS / REST / GraphQL ↕️

Controller Logic ⚙️

Data Store 🗃️

DevOps 📤

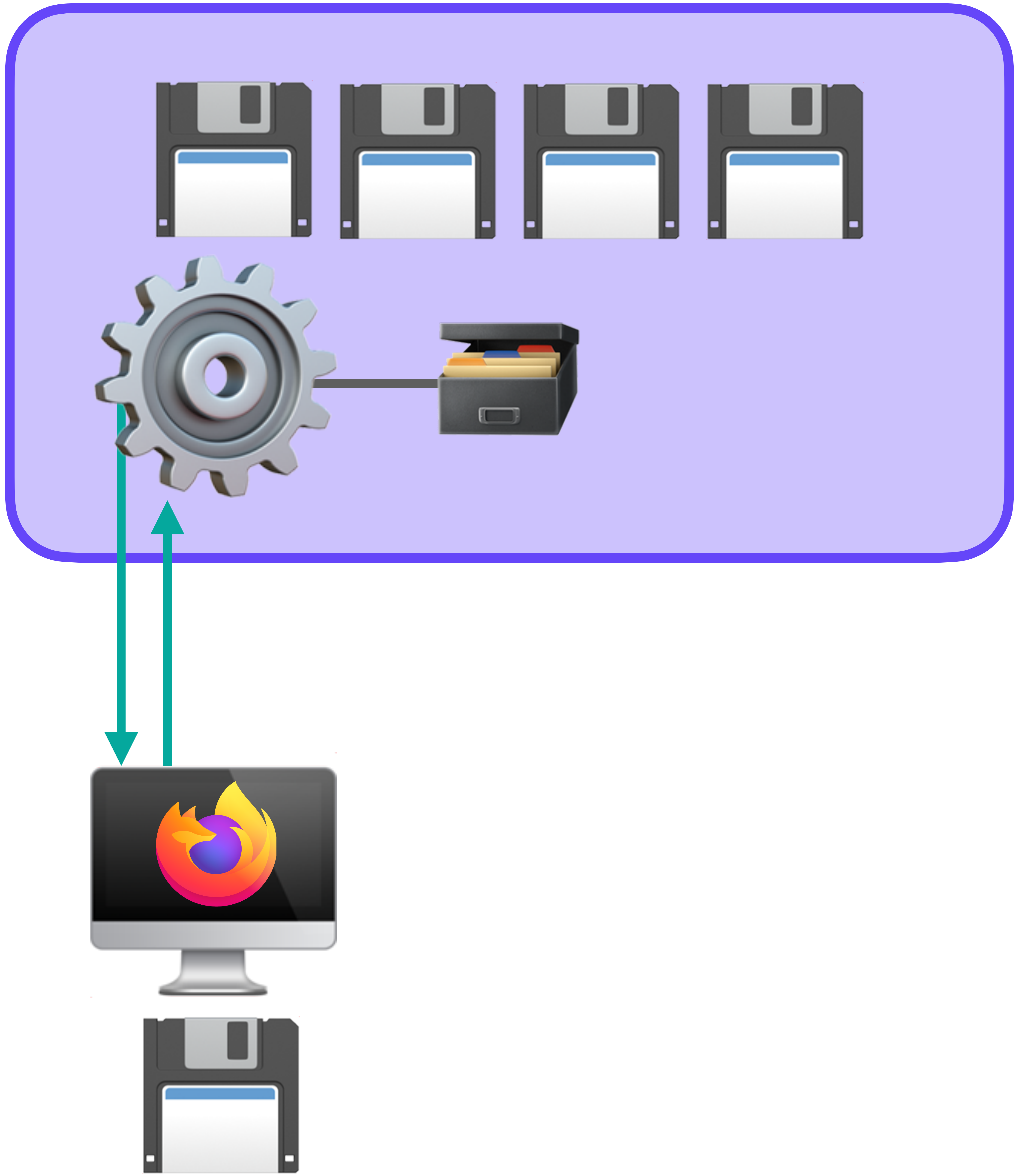
Developer 👩🏾‍💻



On the Edge 🧗‍♀️

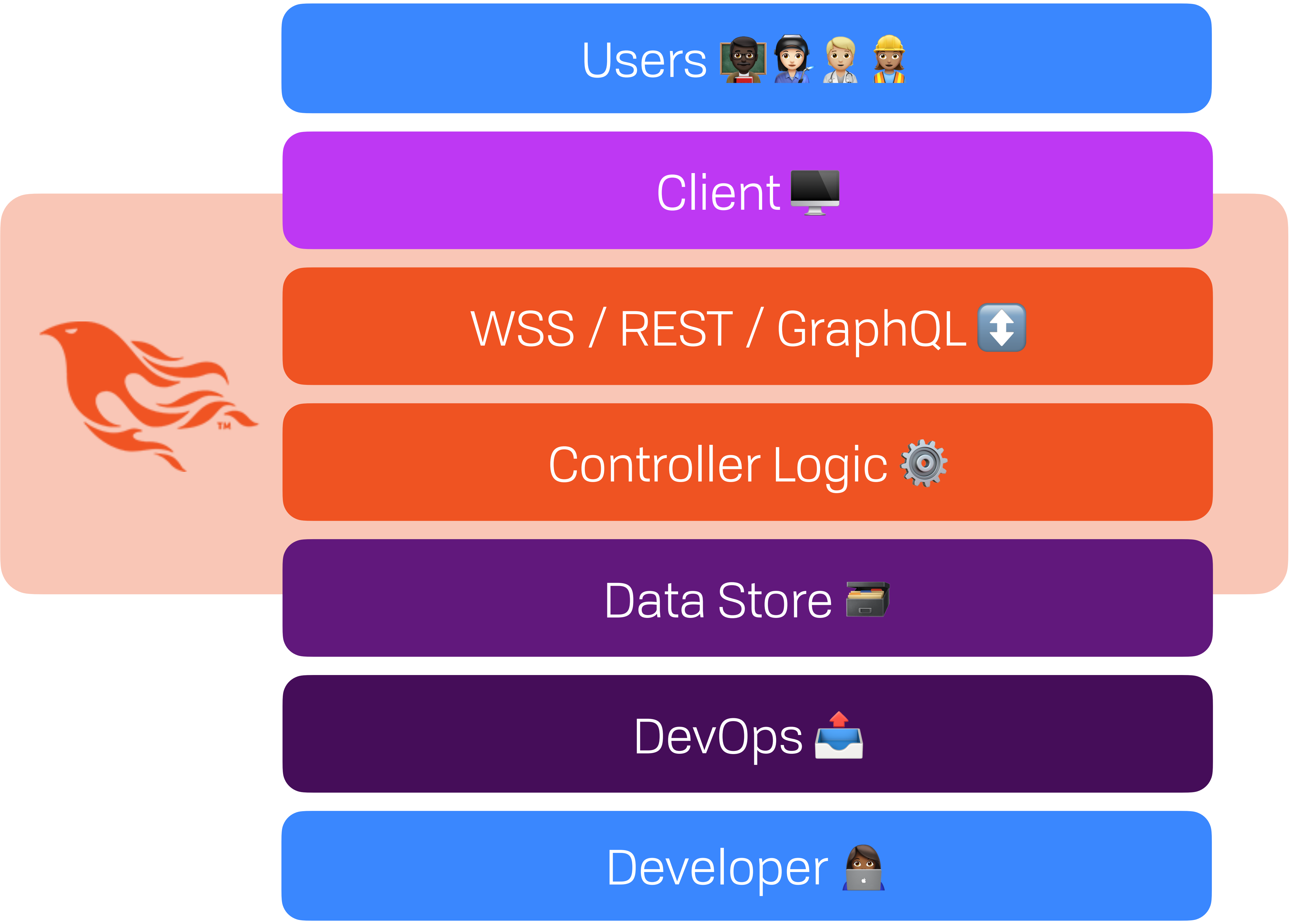
Phoenix LiveView

- Users 👩🏿‍🏫👩🏻‍✈️👨🏼‍⚕️👷🏽‍♀️
- Client 🖥️
- WSS / REST / GraphQL ↕️
- Controller Logic ⚙️
- Data Store 🗃️
- DevOps 📤
- Developer 👩🏾‍💻



On the Edge 🧗‍♀️

Phoenix LiveView



Users 👩🏿‍🏫👩🏻‍✈️👨🏼‍⚕️👷🏽‍♀️

Client 🖥️

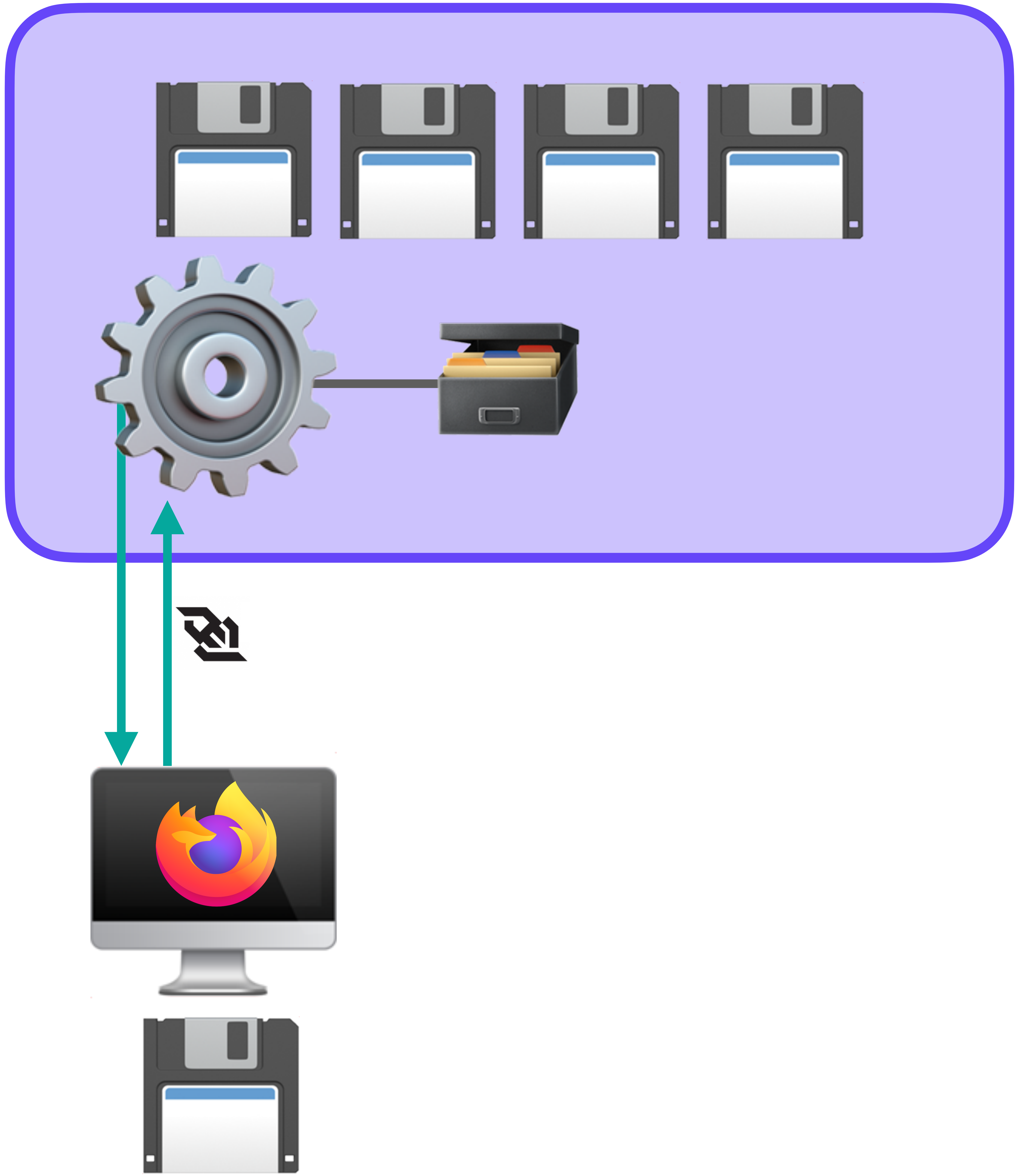
WSS / REST / GraphQL ↕️

Controller Logic ⚙️

Data Store 🗃️

DevOps 📤

Developer 👩🏽‍💻



On the Edge 🧗‍♀️

Phoenix LiveView

Users 👩🏿‍🏫👩🏻‍✈️👨🏼‍⚕️👷🏽‍♀️

Client 🖥️

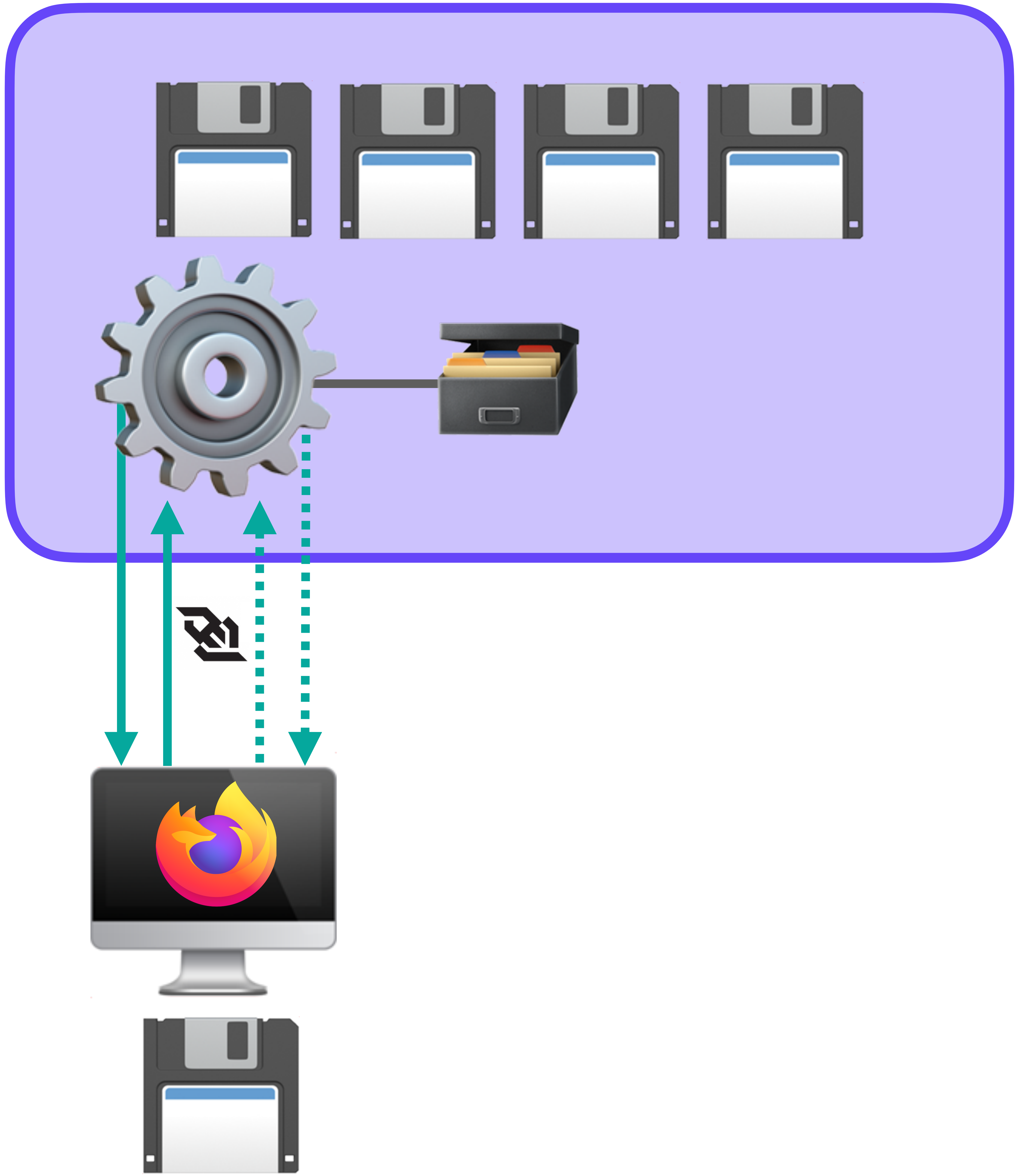
WSS / REST / GraphQL ↕️

Controller Logic ⚙️

Data Store 🗃️

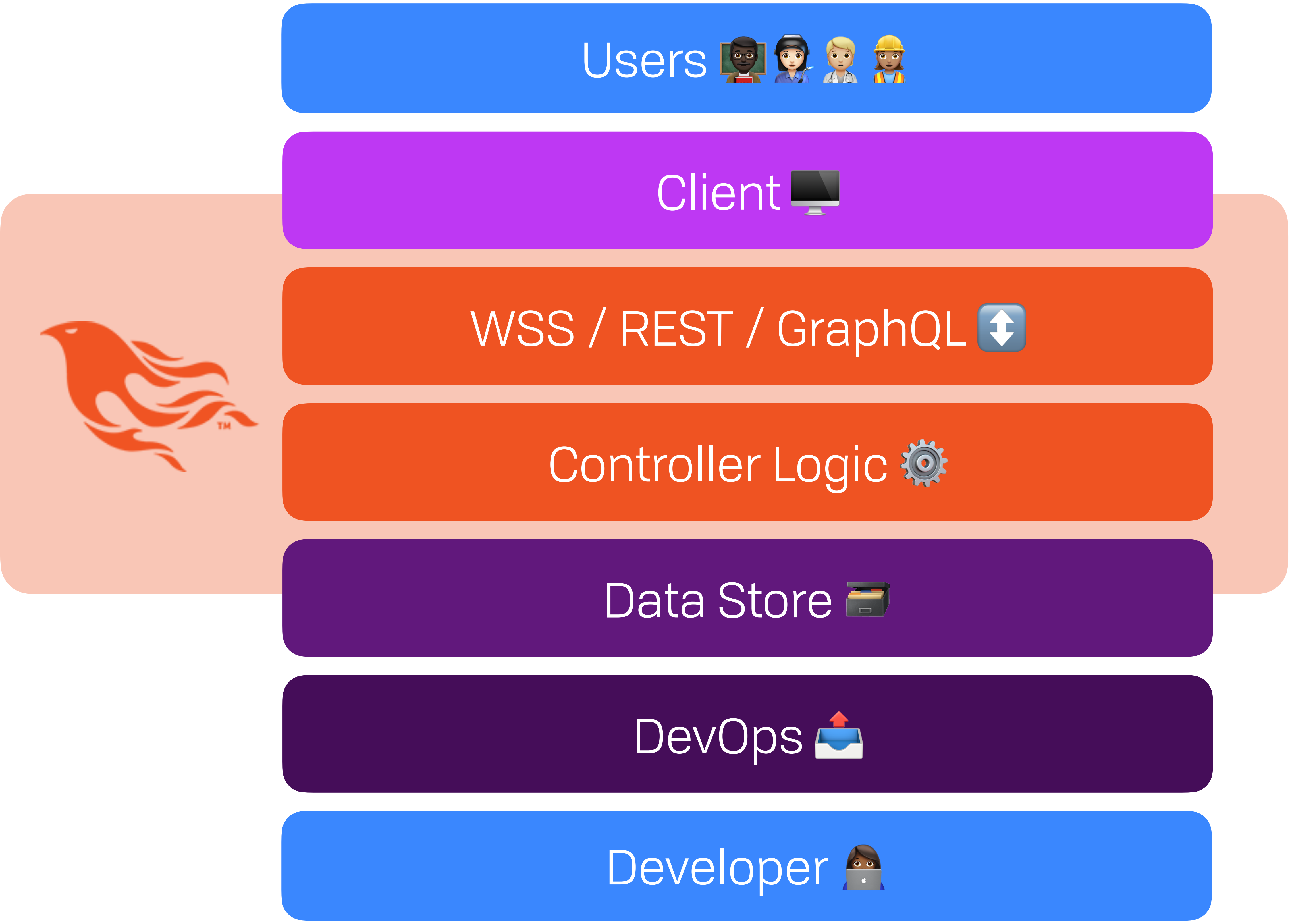
DevOps 📤

Developer 👩🏾‍💻



On the Edge 🧗

Phoenix LiveView



Users 👩🏿‍🏫👩🏻‍✈️👨🏼‍⚕️👷🏽‍♀️

Client 🖥️

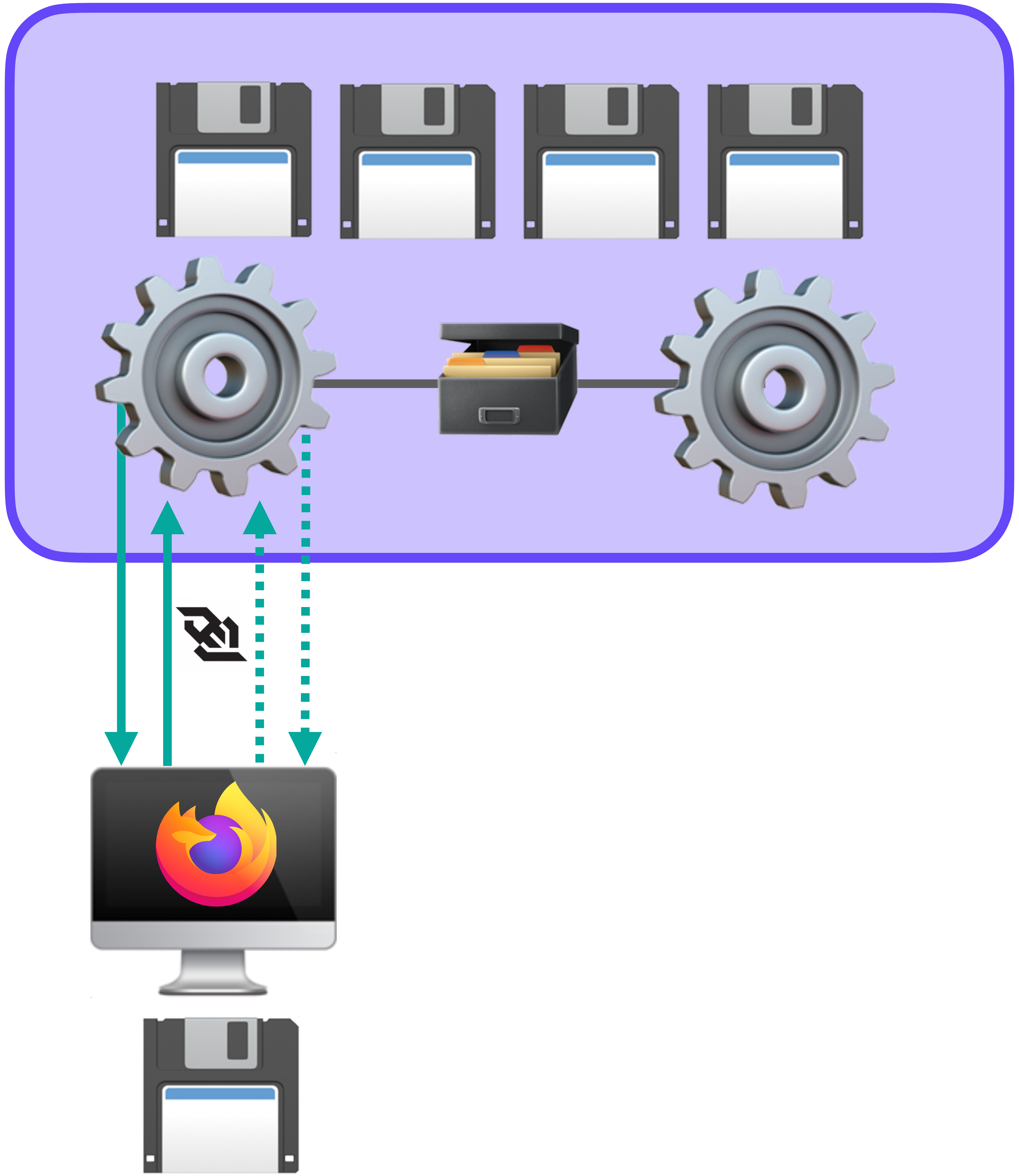
WSS / REST / GraphQL ↕️

Controller Logic ⚙️

Data Store 🗃️

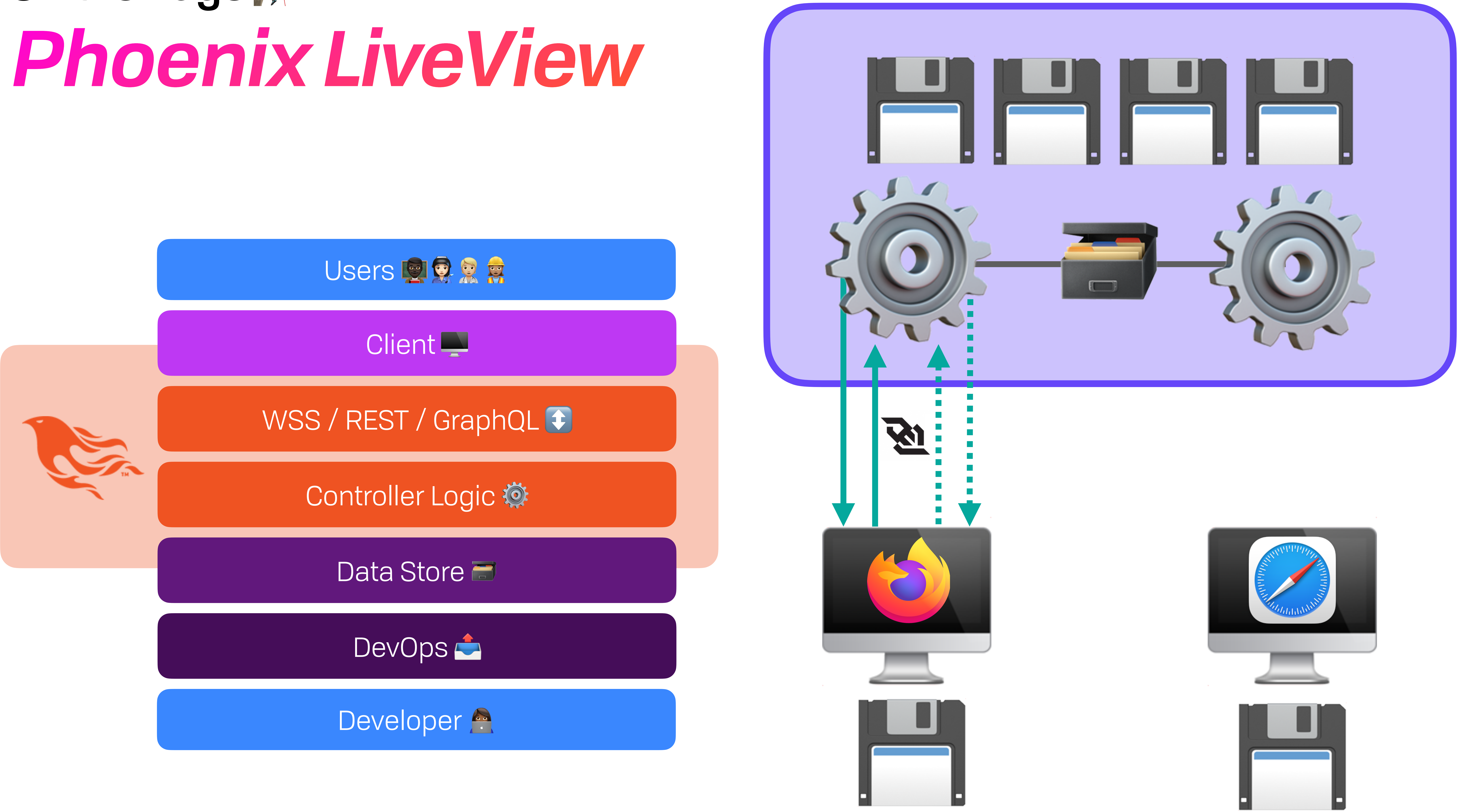
DevOps 📤

Developer 👩🏾‍💻



On the Edge 🧗

Phoenix LiveView



Users 👩🏿‍🏫👩🏻‍✈️👨🏼‍⚕️👷🏽‍♀️

Client 🖥️

WSS / REST / GraphQL ↕️

Controller Logic ⚙️

Data Store 🗃️

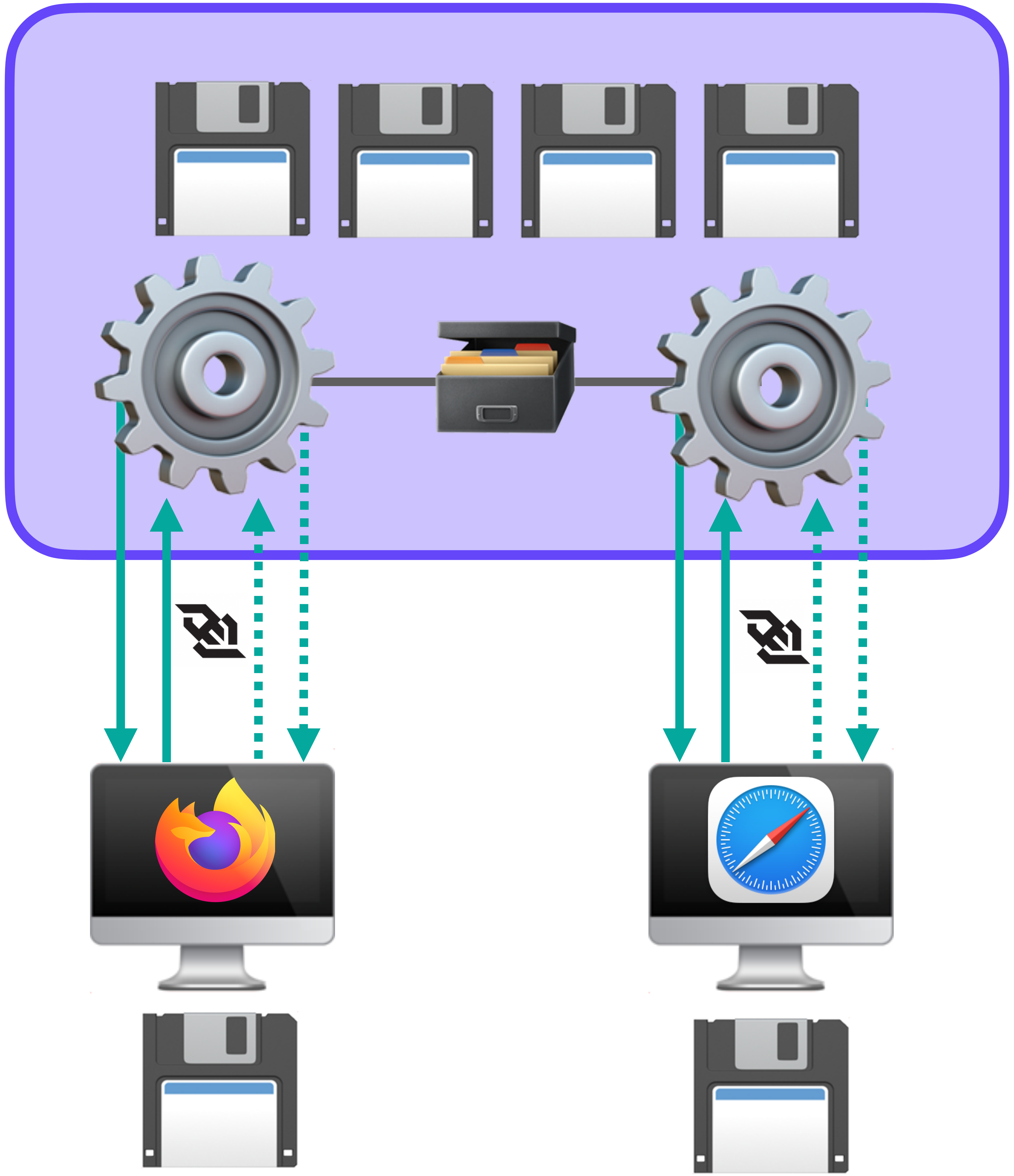
DevOps 📤

Developer 🧑🏽‍💻

On the Edge 🧗

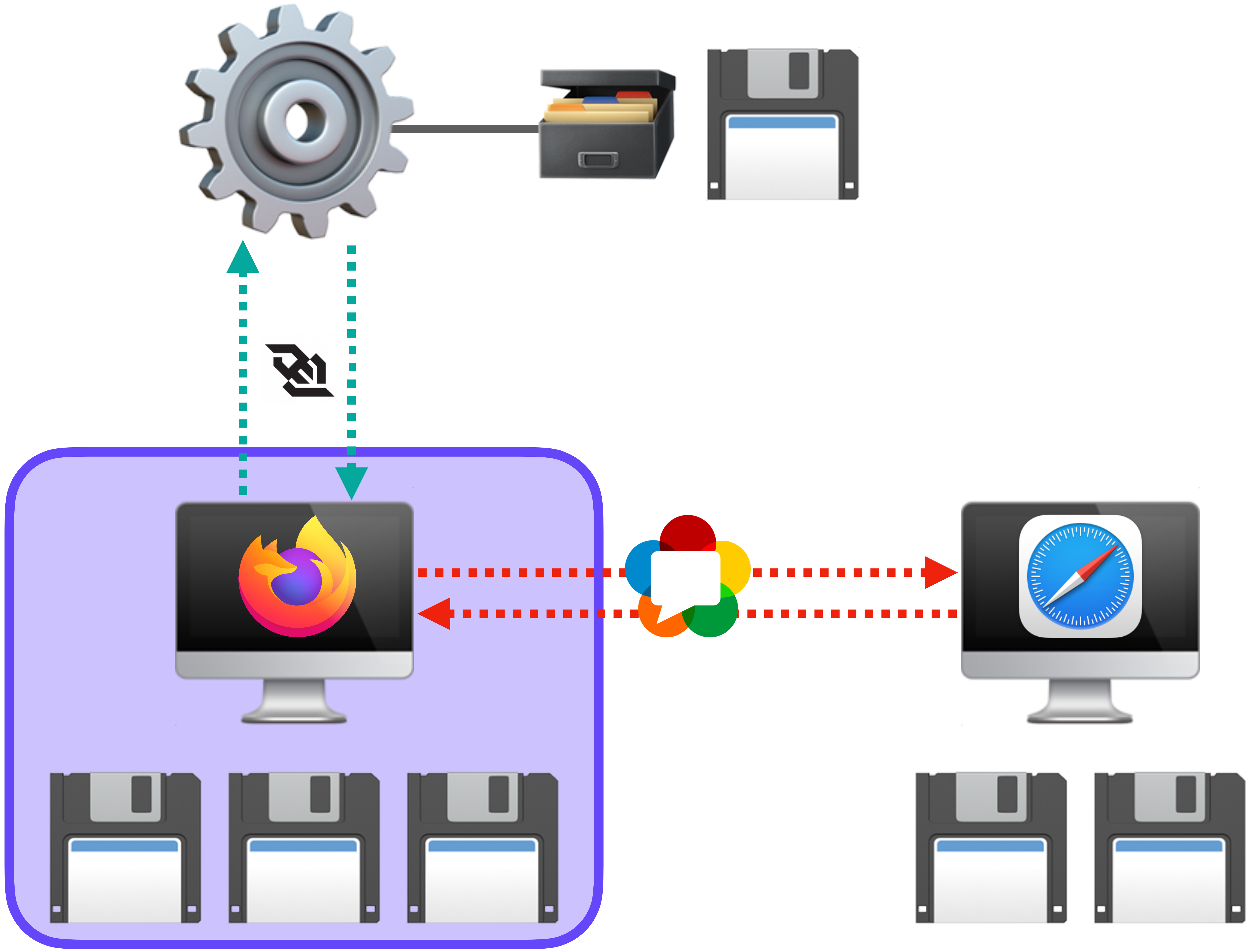
Phoenix LiveView

- Users 👩🏿‍🏫👩🏻‍✈️👨🏼‍⚕️👷🏽
- Client 🖥️
- WSS / REST / GraphQL ↕️
- Controller Logic ⚙️
- Data Store 🗃️
- DevOps 📤
- Developer 👩🏾‍💻



On the Edge 🧗‍♀️

Upside Down



It’s all about the

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 the Data 📊

Problems!

Property	Consequence
Run anywhere	No process in charge of access control
Casual islands	Inconsistent views of data (or downtime)
Unstable topology	No consistent connections
Local first	In accessible, no replicas

It’s All About the Data 📊

CAP ➡️ PACELC 📦🦌

It’s All About the Data 📊

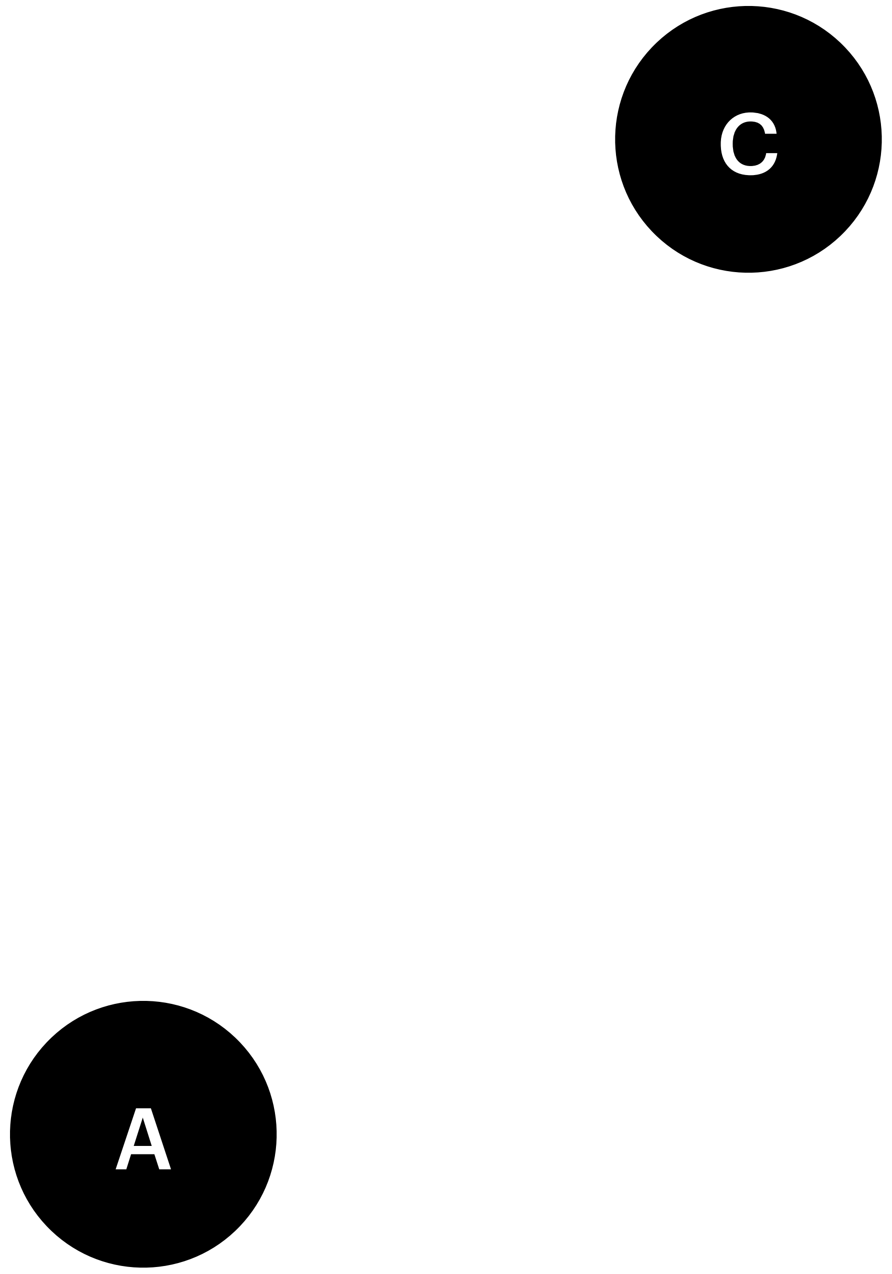
CAP ➡️ PACELC 📦🦌

- If network partition (P)
  - Choose between:
    - Availability (A) ✅ Local-first & uptime
    - Consistency (C)

It’s All About the Data 📊

CAP ➡️ PACELC 📦🦌

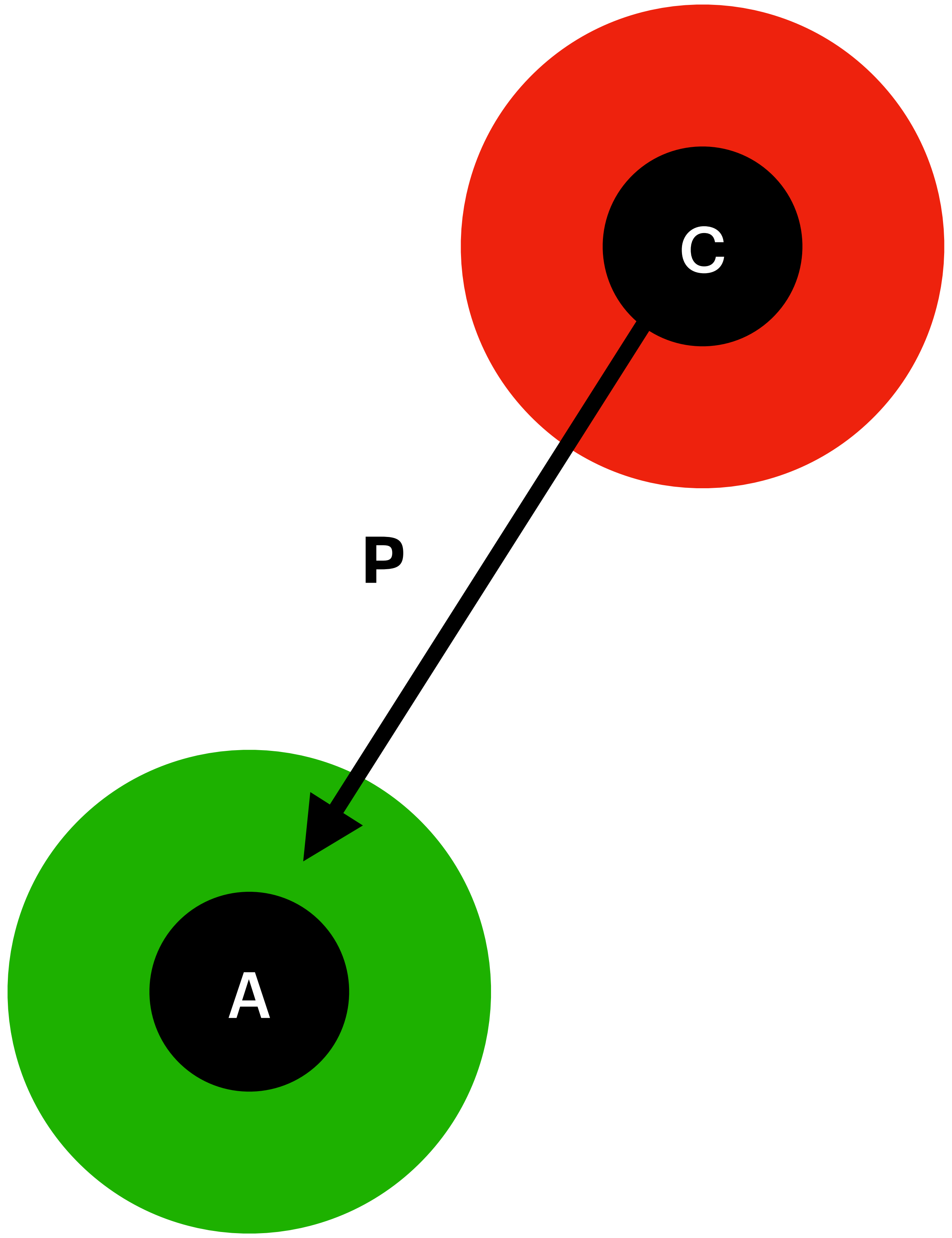
- If network partition (P)
  - Choose between:
    - Availability (A) ✅ Local-first & uptime
    - Consistency (C)



It’s All About the Data 📊

CAP ➡️ PACELC 📦🦌

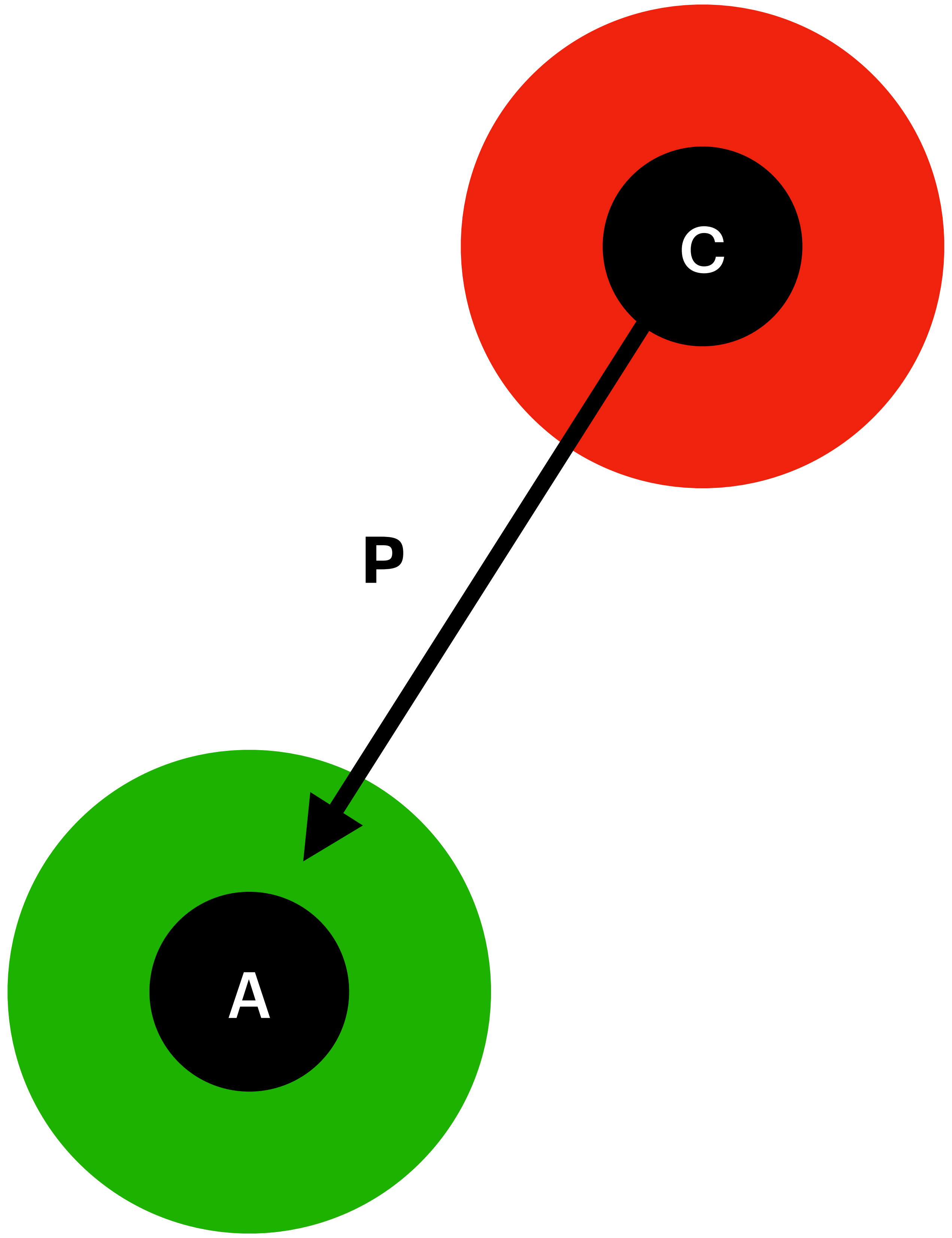
- If network partition (P)
    - Choose between:
        - Availability (A) ✅ Local-first & uptime
        - Consistency (C)



It’s All About the Data 📊

CAP ➡️ PACELC 📦🦌

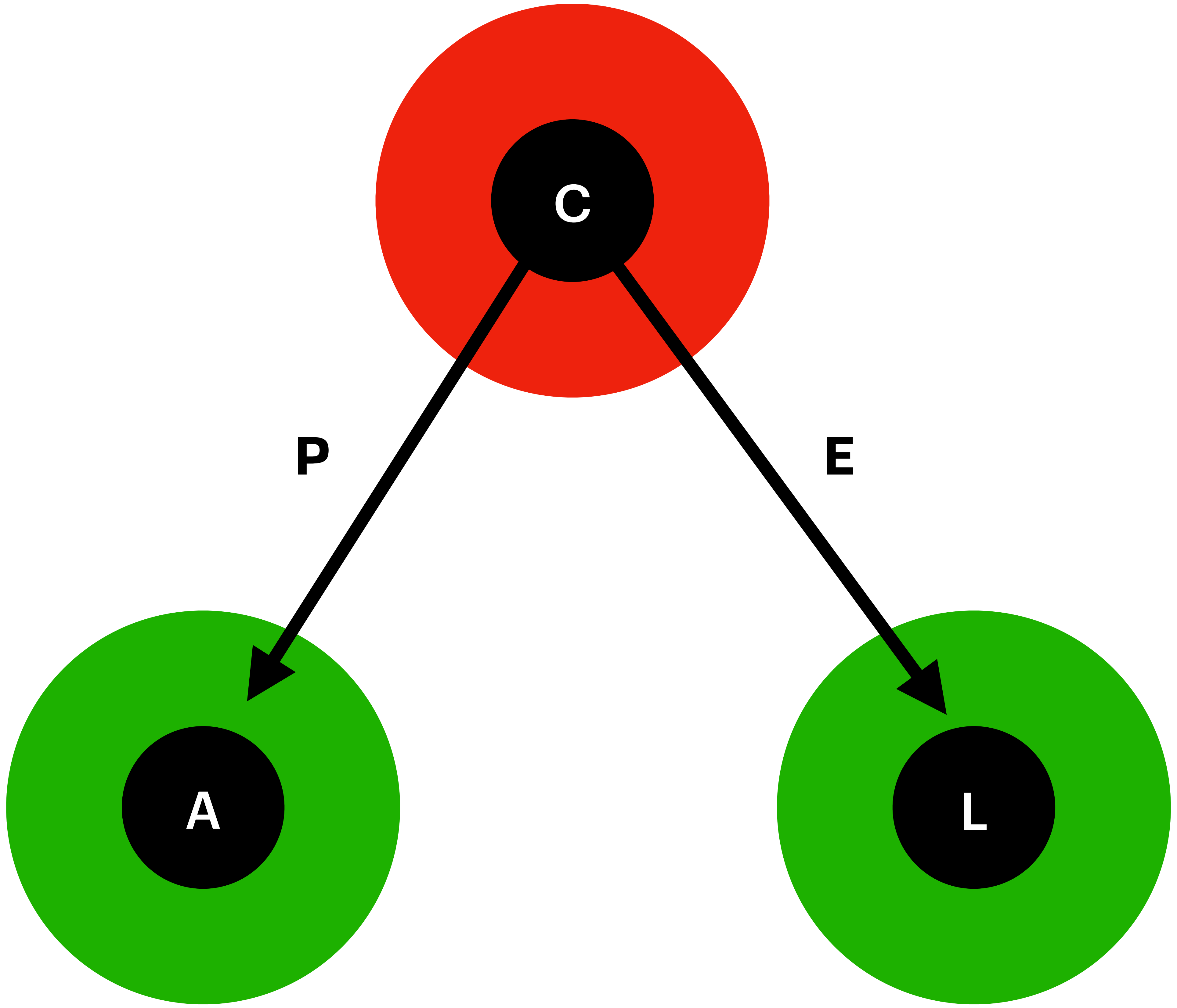
- If network partition (P)
  - Choose between:
    - Availability (A) ✅ Local-first & uptime
    - Consistency (C)
- Else (E) when running normally:
  - Choose between:
    - Latency (L) ✅
    - Consistency (C)



It’s All About the Data 📊

CAP ➡️ PACELC 📦🦌

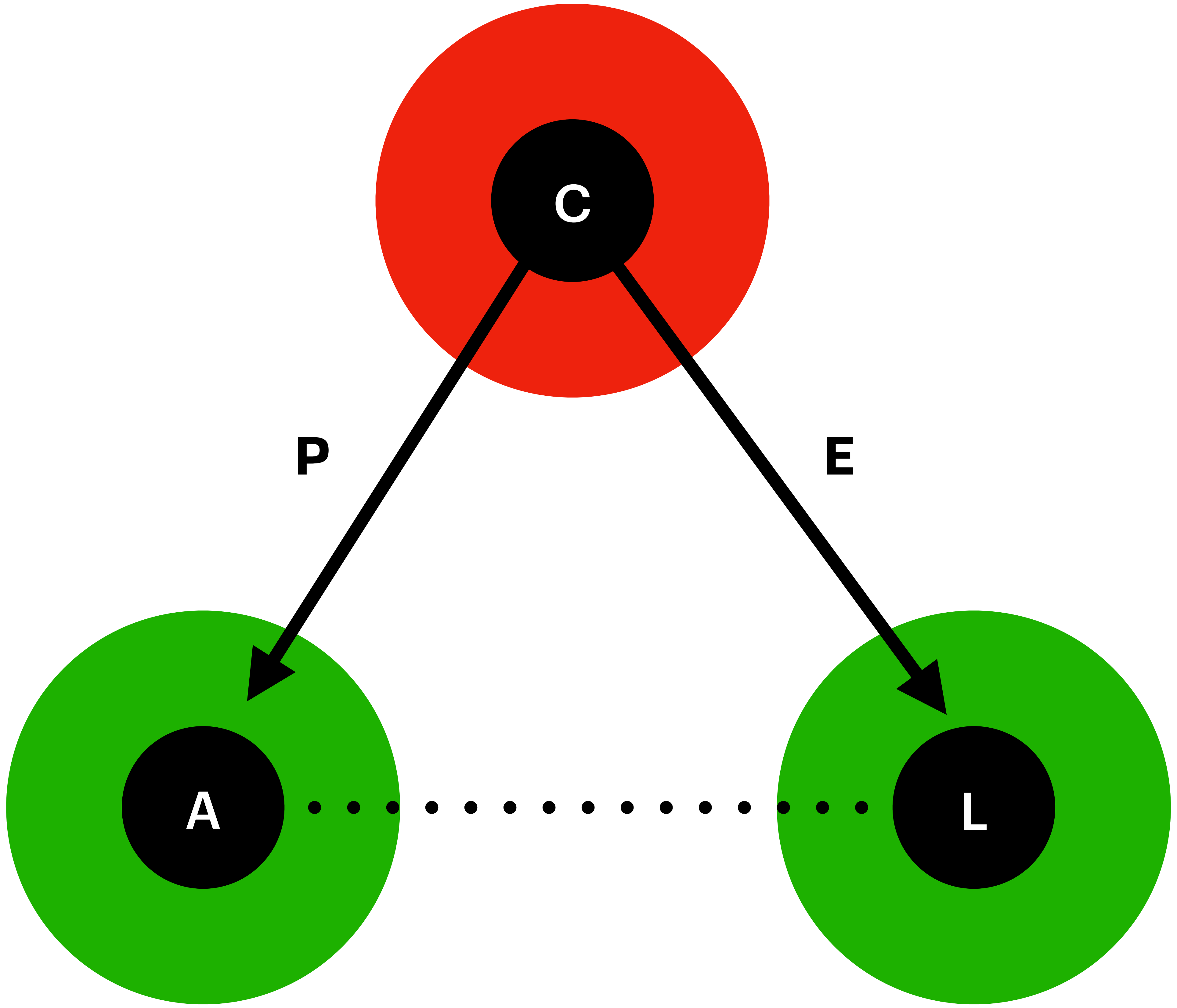
- If network partition (P)
  - Choose between:
    - Availability (A) ✅ Local-first & uptime
    - Consistency (C)
- Else (E) when running normally:
  - Choose between:
    - Latency (L) ✅
    - Consistency (C)



It’s All About the Data 📊

CAP ➡️ PACELC 📦🦌

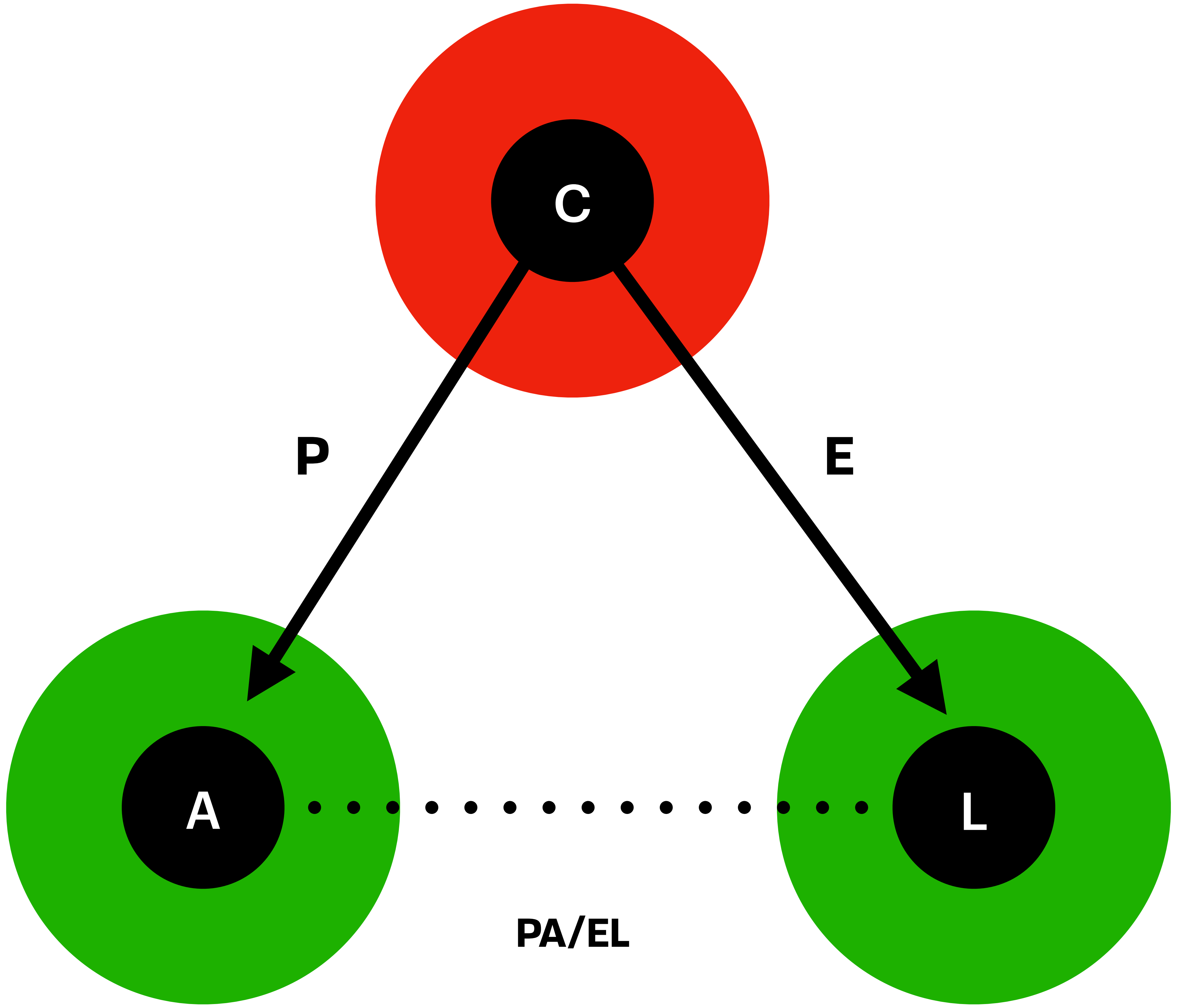
- If network partition (P)
  - Choose between:
    - Availability (A) ✅ Local-first & uptime
    - Consistency (C)
- Else (E) when running normally:
  - Choose between:
    - Latency (L) ✅
    - Consistency (C)



It's All About the Data 📊

CAP ➡️ PACELC 📦🦌

- If network partition (P)
  - Choose between:
    - Availability (A) ✅ Local-first & uptime
    - Consistency (C)
- Else (E) when running normally:
  - Choose between:
    - Latency (L) ✅
    - Consistency (C)



It’s All About the Data 📊

Mutable Content

- Predominantly single-source (per file) server/client
- %{node_id => %{path => content}}
  - DNS maps names to IP addresses
  - PIDs associate processes with numbers
  - e.g. `send(:example@42.123.45.6, :ping)`
- Focused on the physical network
- Referential opacity
  - Calling same PID often will return different data

It’s All About the Data 📊

Mutable Content

- Predominantly single-source (per file) server/client
- %{node_id => %{path => content}}
  - DNS maps names to IP addresses
  - PIDs associate processes with numbers
  - e.g. `send(:example@42.123.45.6, :ping)`
- Focused on the physical network
- Referential opacity
  - Calling same PID often will return different data

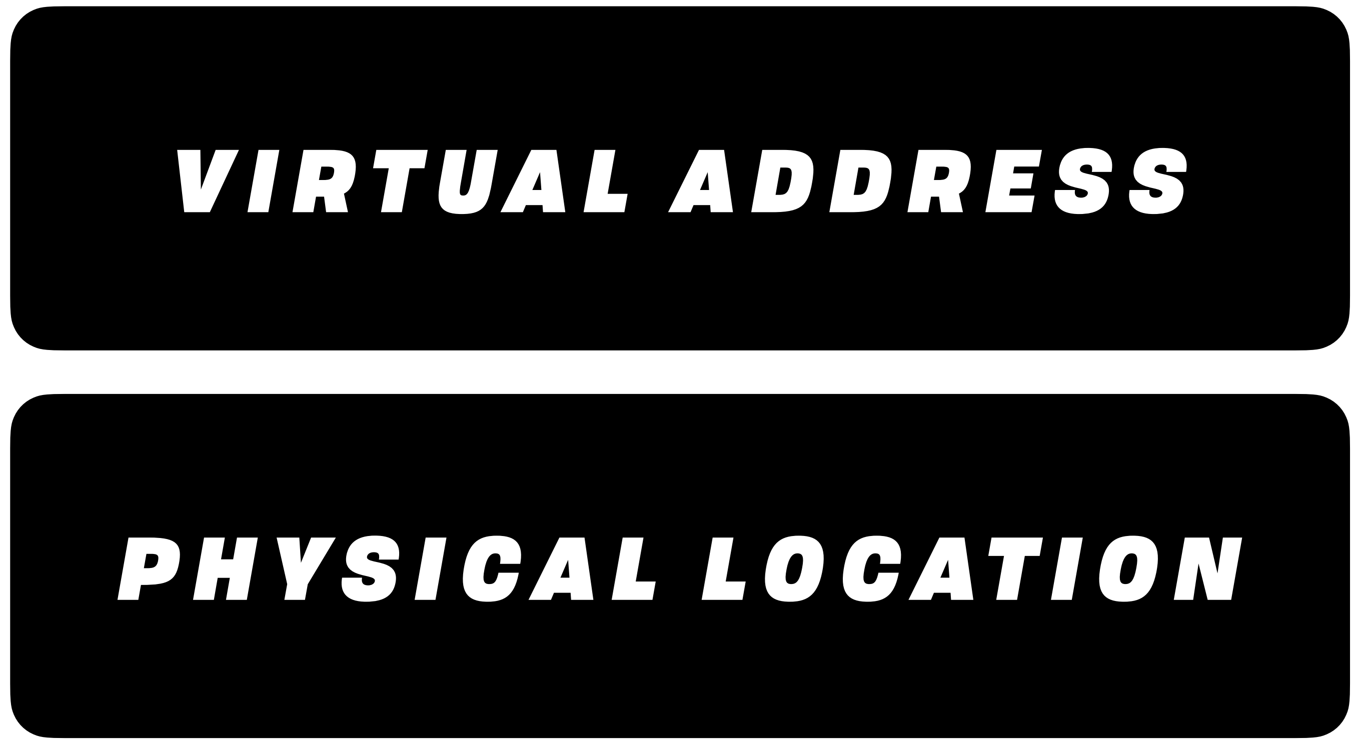
VIRTUAL ADDRESS

PHYSICAL LOCATION

It’s All About the Data 📊

Consistent Keys

- A layer of abstraction above location
- %{hash(content) => content}
  - Hash AKA “content identifier” or CID
  - Special “universal” relationship to content
- Focused on the **data**
  - Stored anywhere, same ID
  - Efficient caching
- Immutable data++
  - Not just consistent pointers; consistent data



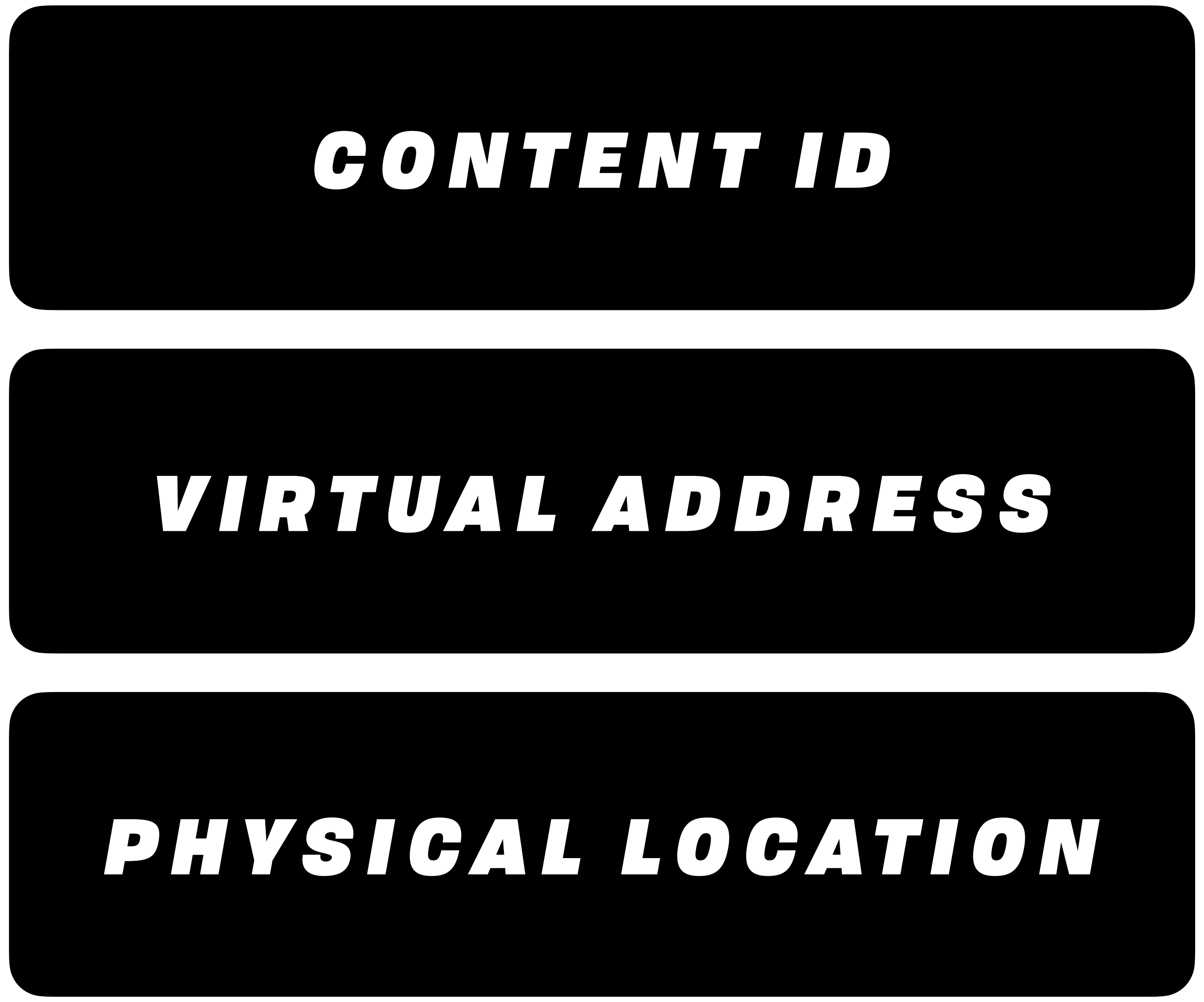
VIRTUAL ADDRESS

PHYSICAL LOCATION

It’s All About the Data 📊

Consistent Keys

- A layer of abstraction above location
- %{hash(content) => content}
  - Hash AKA “content identifier” or CID
  - Special “universal” relationship to content
- Focused on the **data**
  - Stored anywhere, same ID
  - Efficient caching
- Immutable data++
  - Not just consistent pointers; consistent data



CONTENT ID

VIRTUAL ADDRESS

PHYSICAL LOCATION

It’s All About the Data 📊

Hash-Based Relationships

It’s All About the Data 📊

Hash-Based Relationships

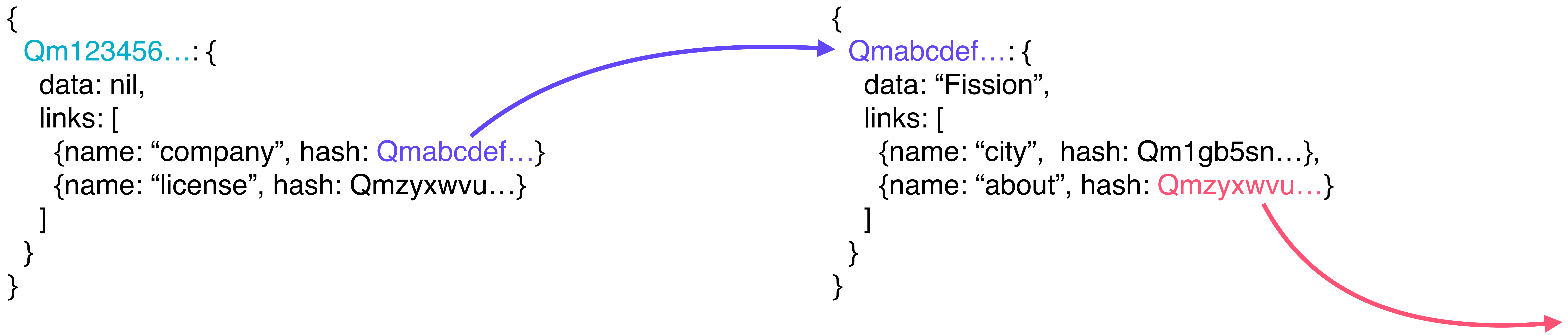
(CID ~ Data PID)

```
{
  Qm123456…: {
    data: nil,
    links: [
      {name: “company”, hash: Qmabcdef…}
      {name: “license”, hash: Qmzyxwvu…}
    ]
  }
}
```

It’s All About the Data 📊

Hash-Based Relationships

(CID ~ Data PID)



```
{
  Qm123456…: {
    data: nil,
    links: [
      {name: “company”, hash: Qmabcdef…}
      {name: “license”, hash: Qmzyxwvu…}
    ]
  }
}
```

```
{
  Qmabcdef…: {
    data: “Fission”,
    links: [
      {name: “city”,  hash: Qm1gb5sn…},
      {name: “about”, hash: Qmzyxwvu…}
    ]
  }
}
```

It's All About the Data 📊

Hash-Based Relationships

(CID ~ Data PID)

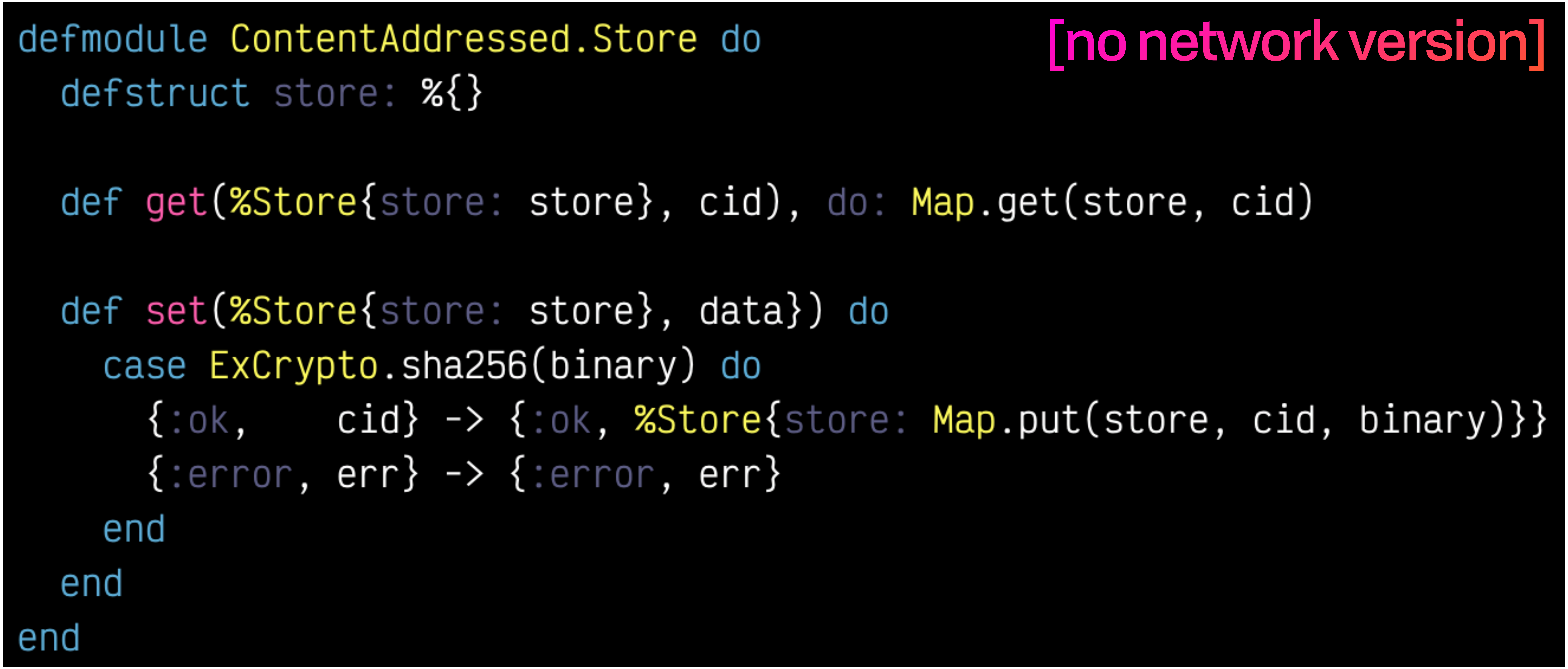
```
{
  Qm123456…: {
    data: nil,
    links: [
      {name: “company”, hash: Qmabcdef…}
      {name: “license”, hash: Qmzyxwvu…}
    ]
  }
}
```

```
{
  Qmabcdef…: {
    data: “Fission”,
    links: [
      {name: “city”,  hash: Qm1gb5sn…},
      {name: “about”, hash: Qmzyxwvu…}
    ]
  }
}
```

```
Qm123456…/company/about/ceo
=> “Boris Mann”
```

It’s All About the Data 📊

Content IDs Are Easy



[no network version]

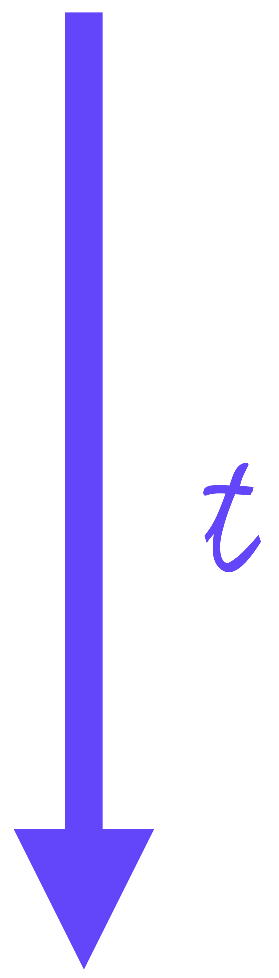
```
defmodule ContentAddressed.Store do
  defstruct store: %{}

  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
```

It’s All About the Data 📊

Partial Dependencies

It’s All About the Data 📊

Partial Dependencies

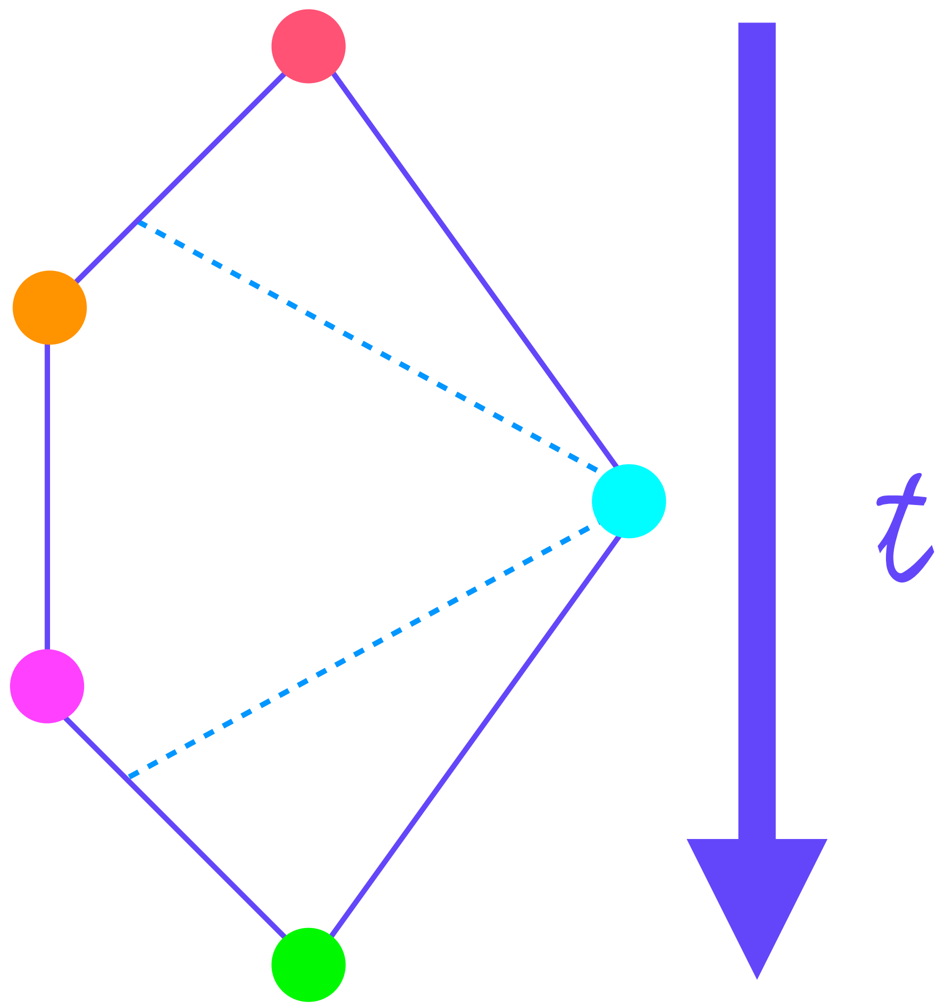


It’s All About the Data 📊

Partial Dependencies

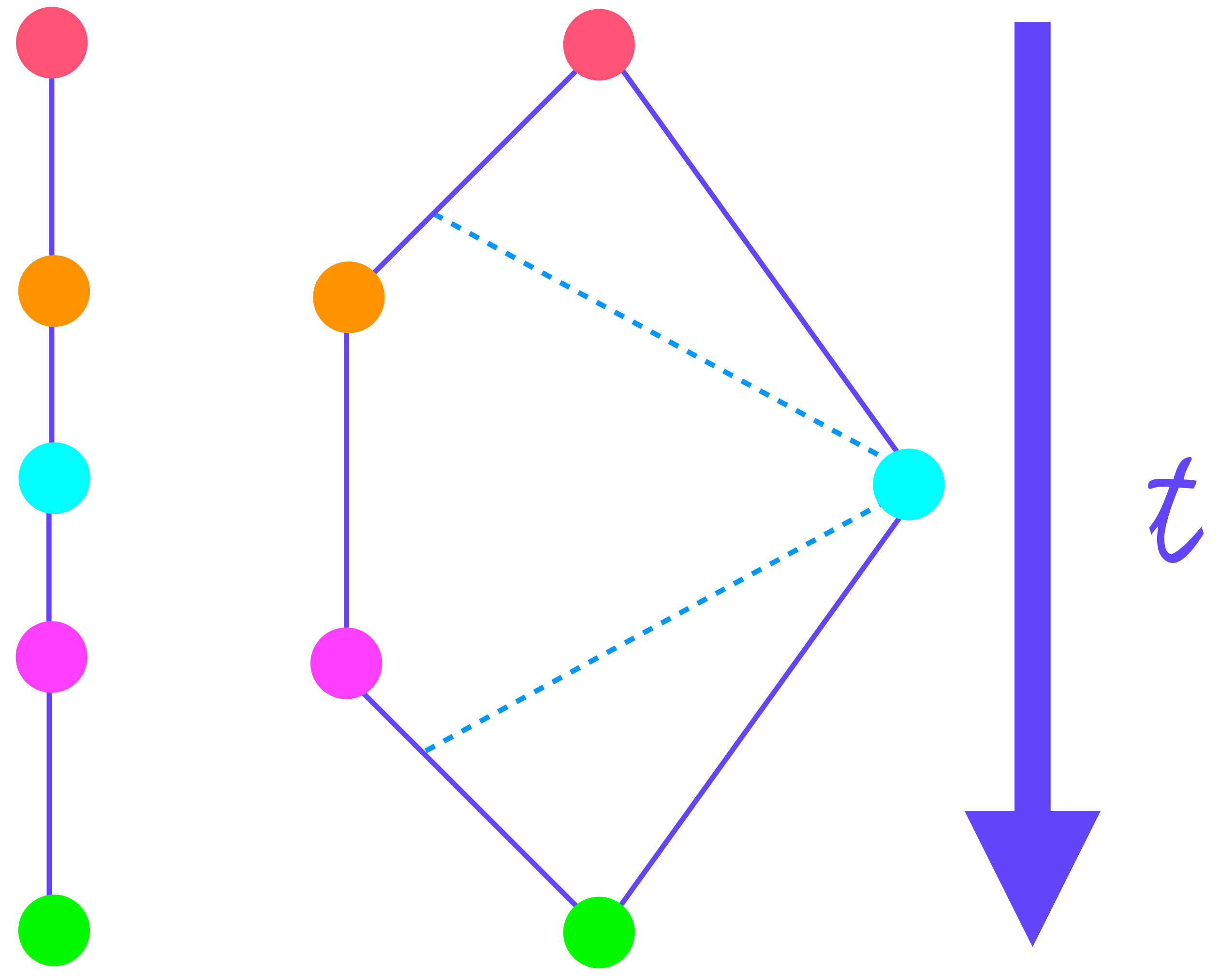
It’s All About the Data 📊

Partial Dependencies



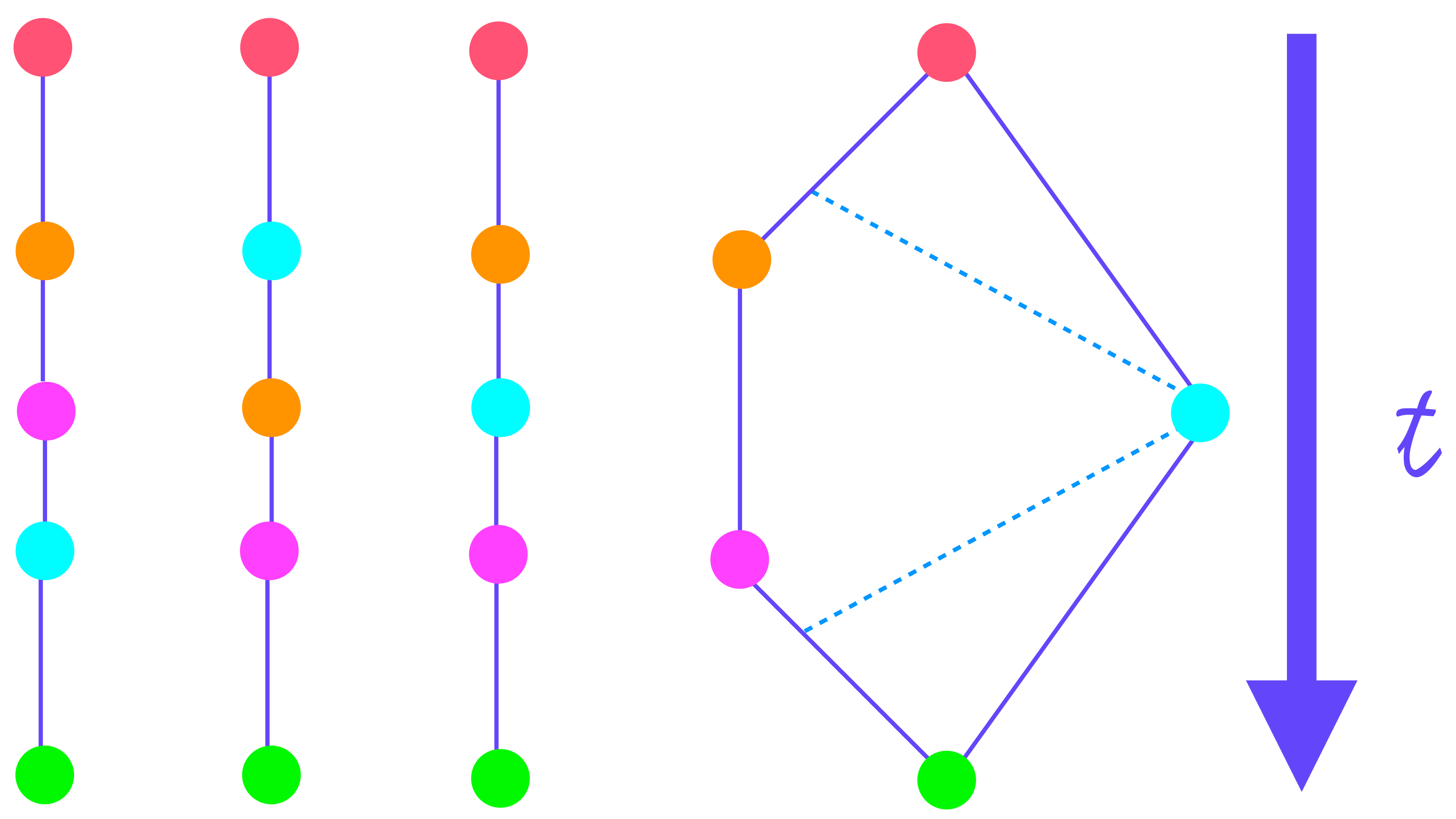
It’s All About the Data 📊

Partial Dependencies



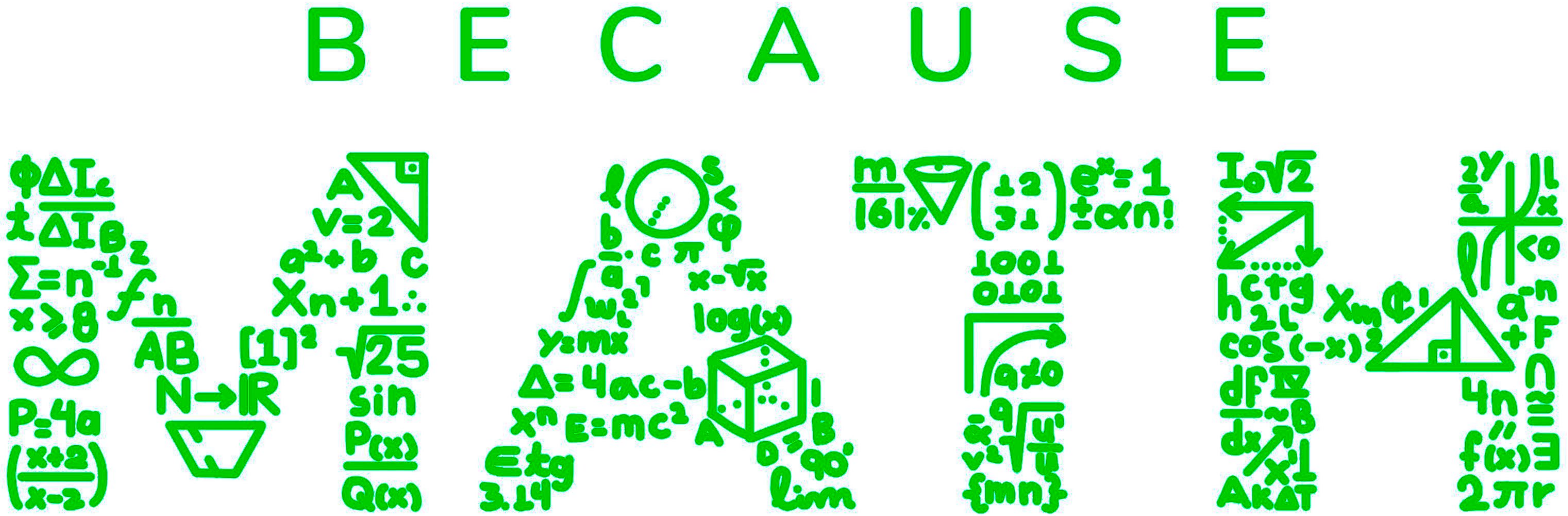
It’s All About the Data 📊

Partial Dependencies



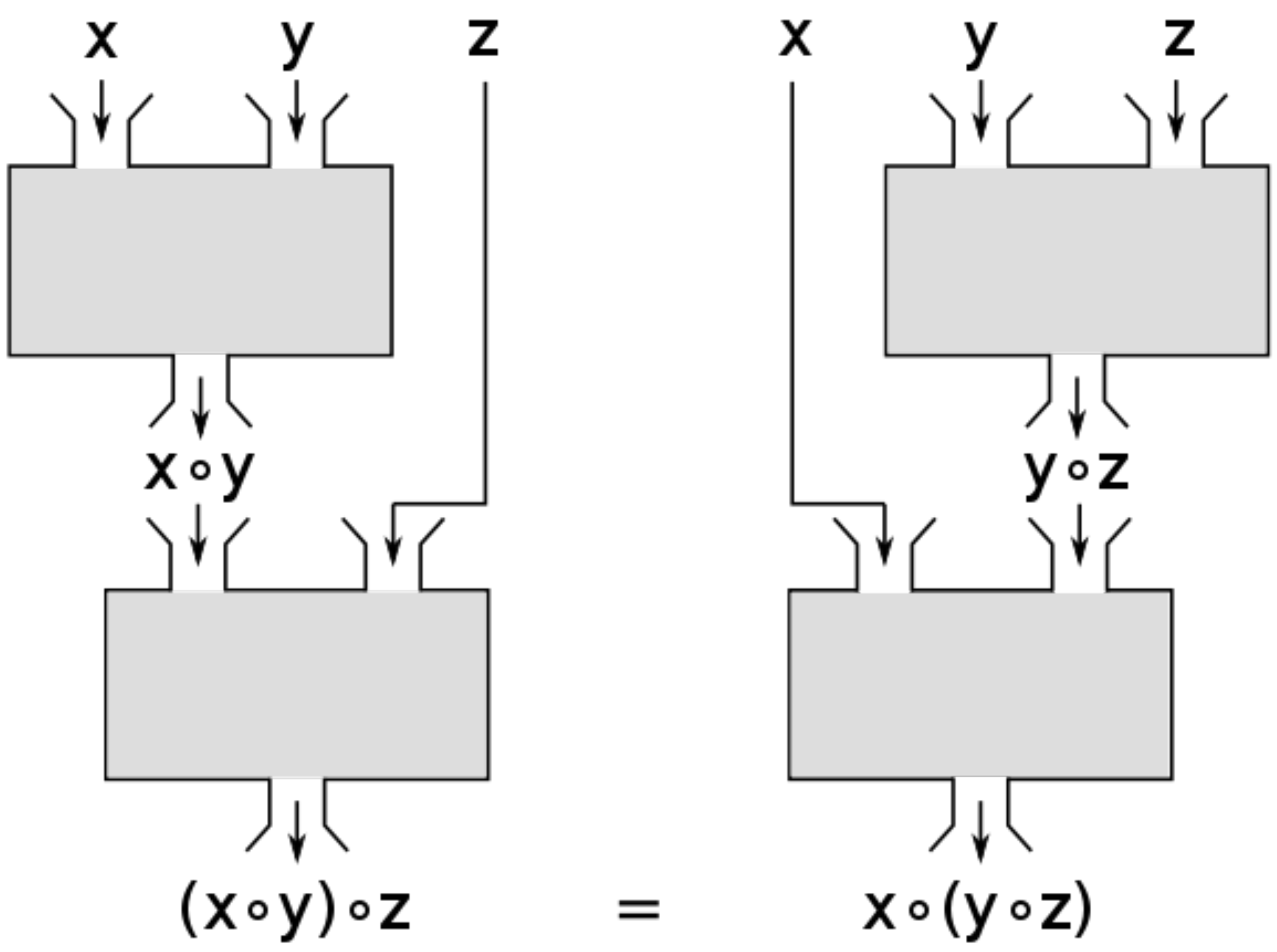
It's All About the Data 📊

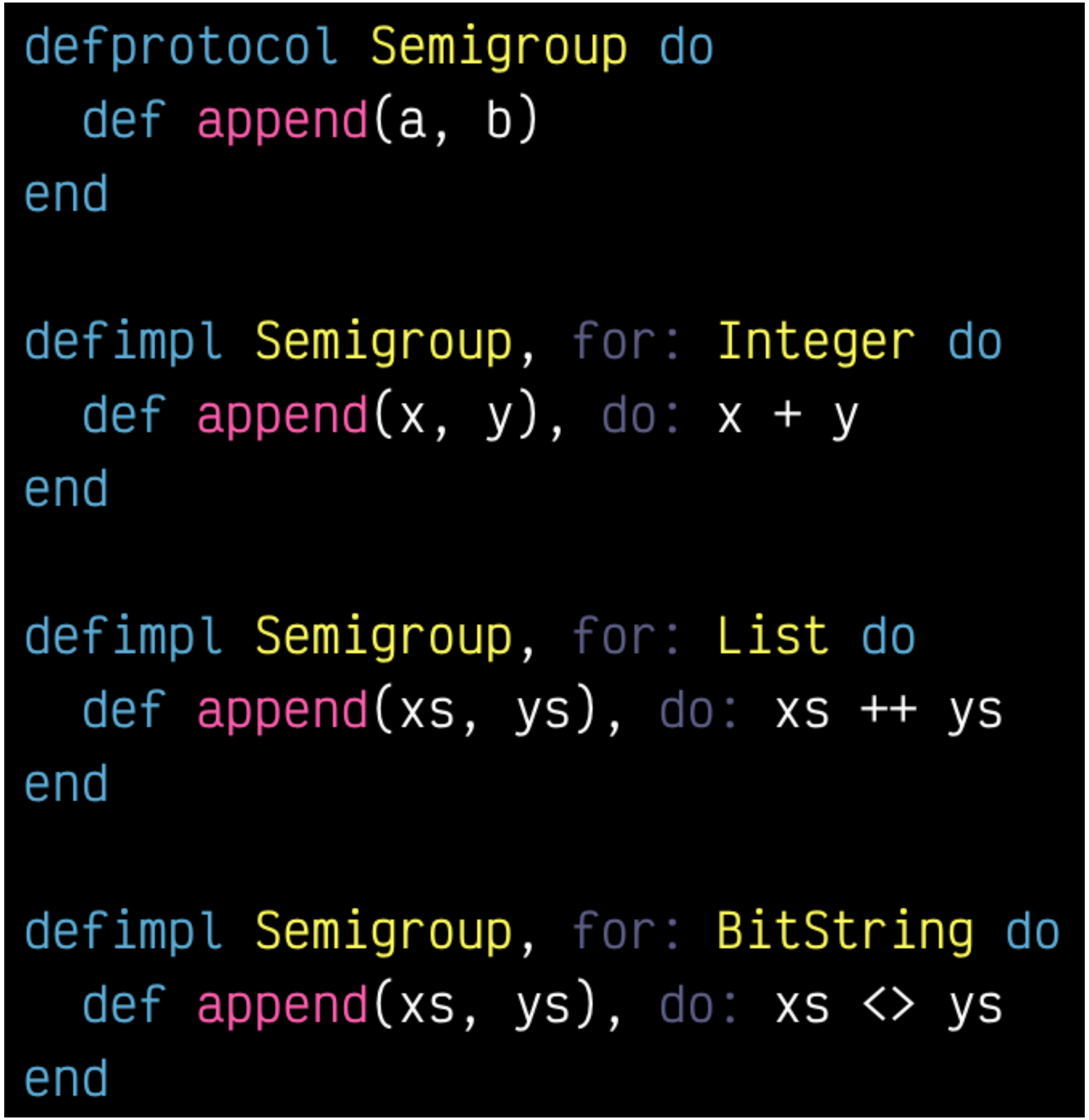
This all works...



It’s All About the Data 📊

Associative

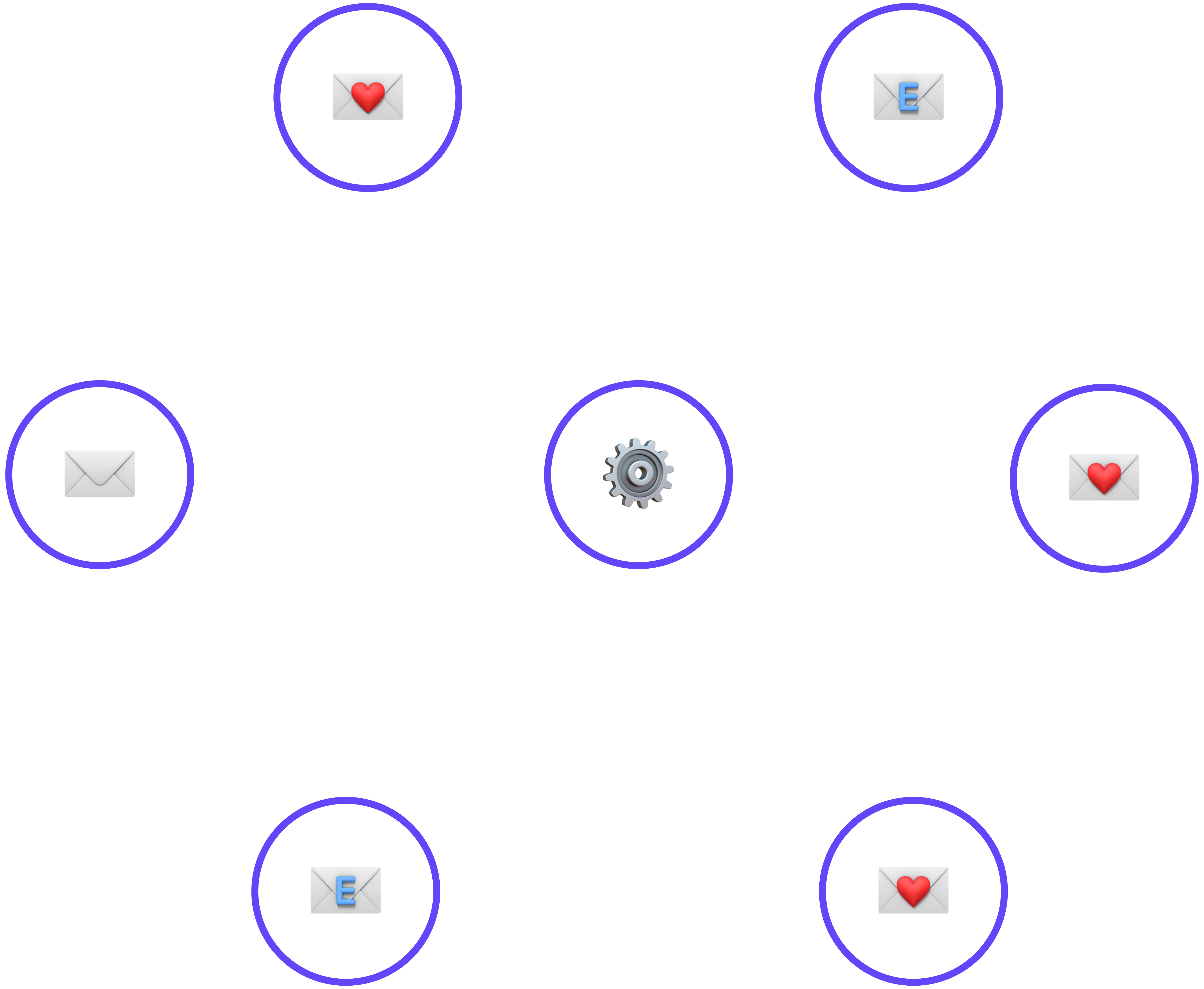




```
defprotocol Semigroup do
  def append(a, b)
end

defimpl Semigroup, for: Integer do
  def append(x, y), do: x + y
end

defimpl Semigroup, for: List do
  def append(xs, ys), do: xs ++ ys
end

defimpl Semigroup, for: BitString do
  def append(xs, ys), do: xs <> ys
end
```

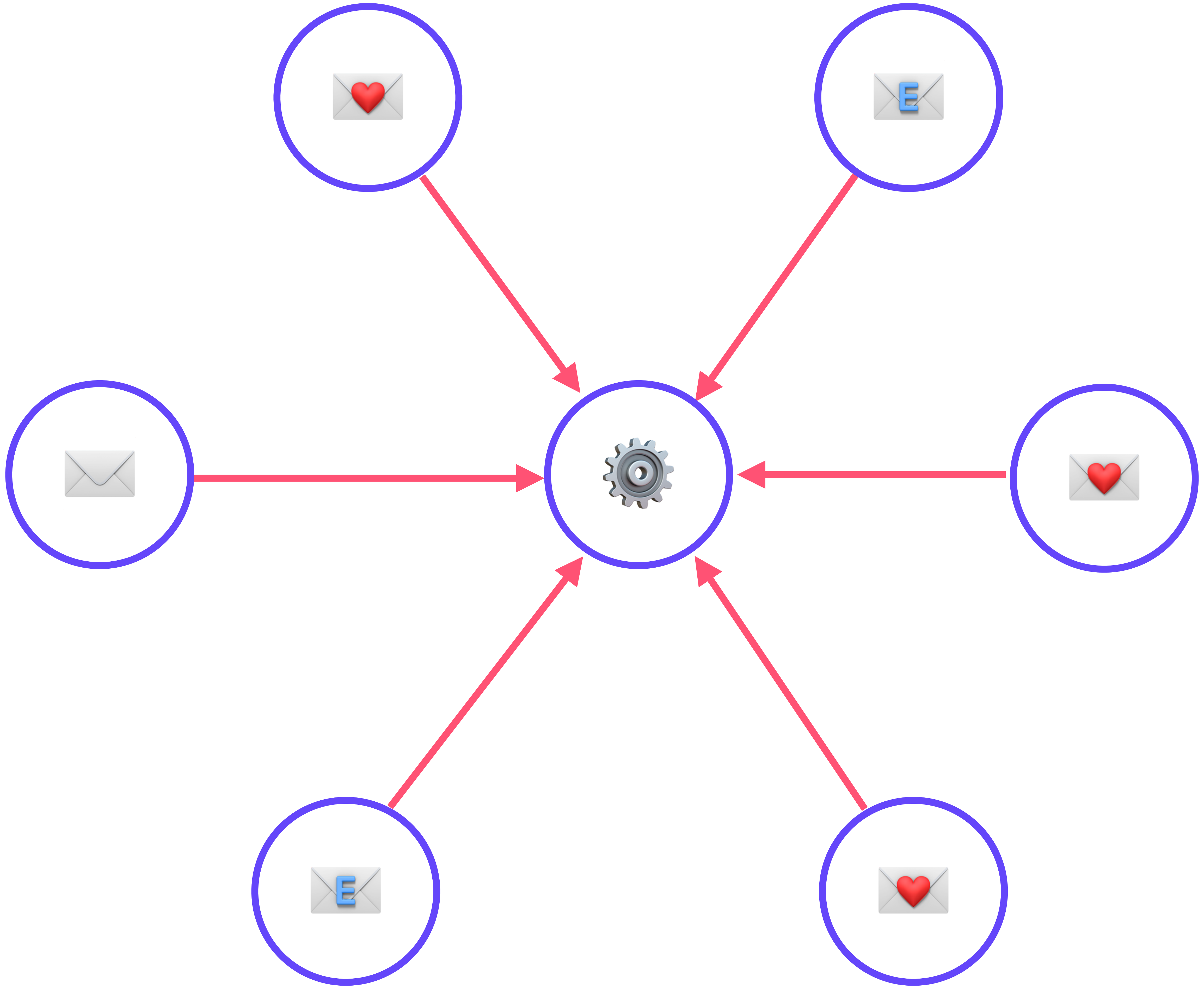
It’s All About the Data 📊

Out of Order Delivery



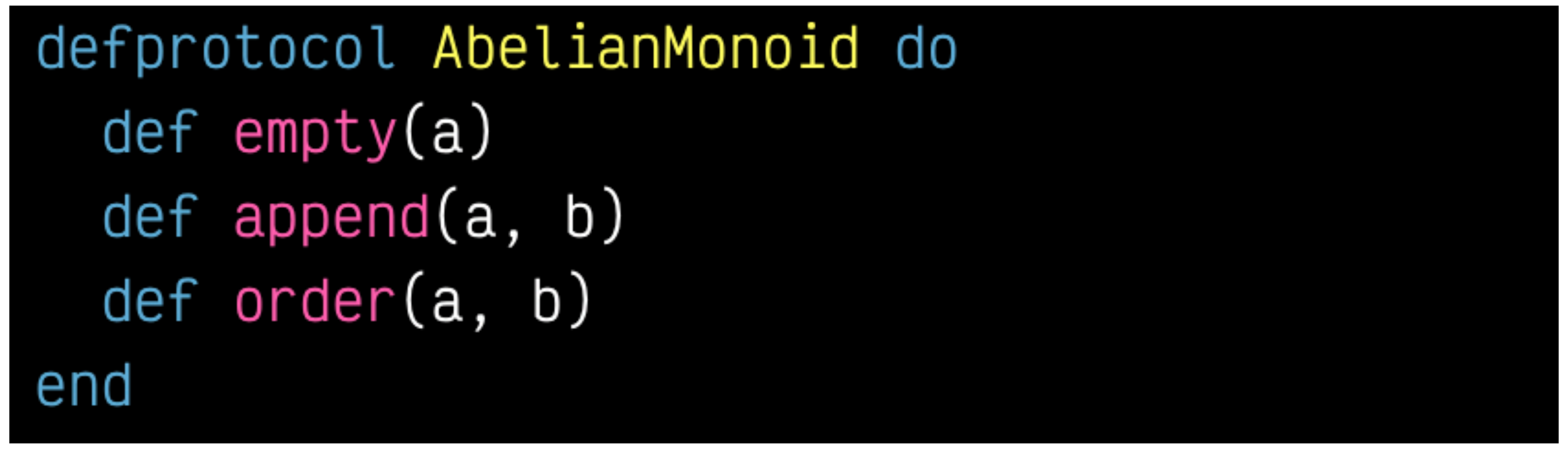
It's All About the Data 📊

Out of Order Delivery

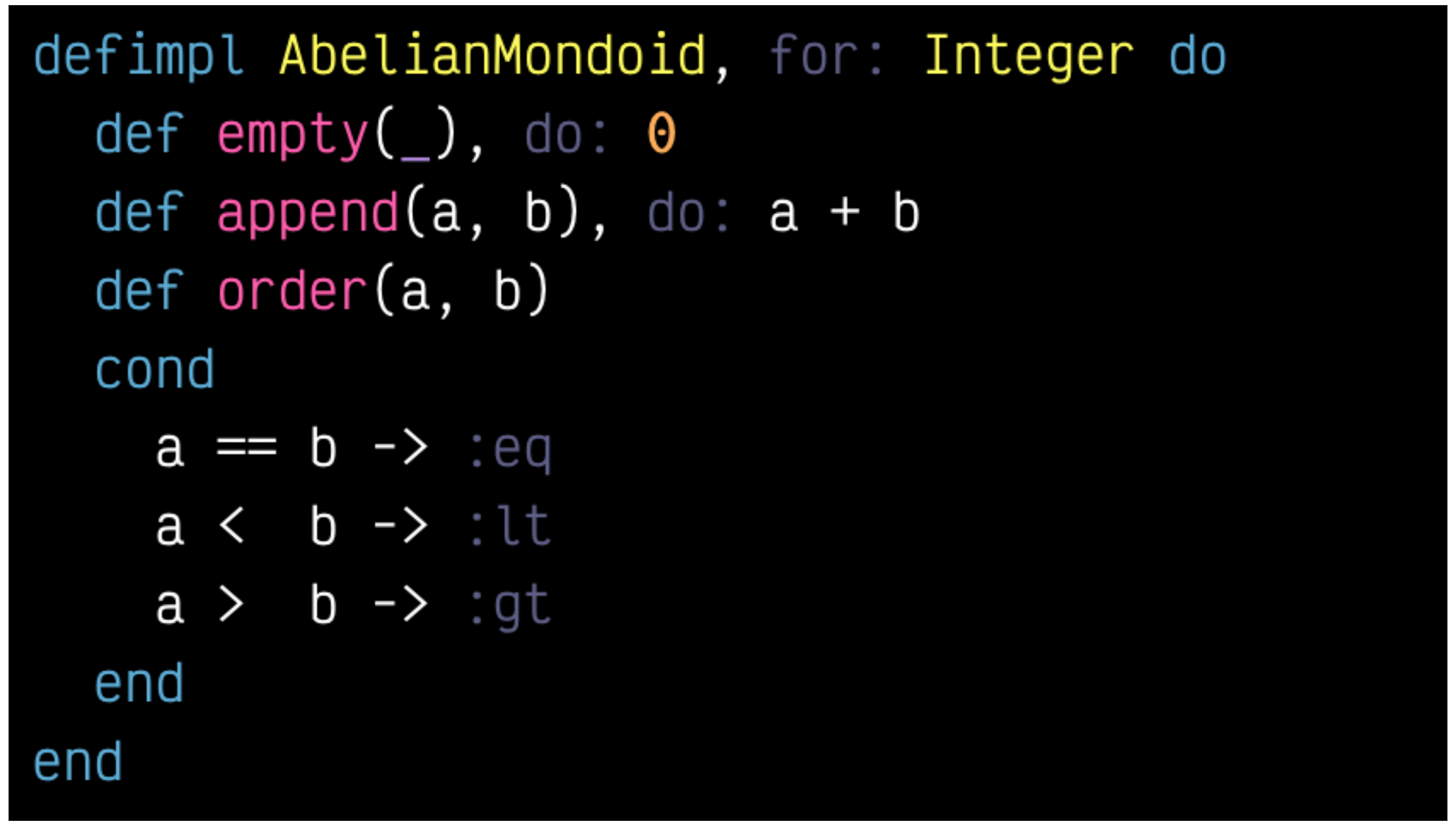


It's All About the Data 📊

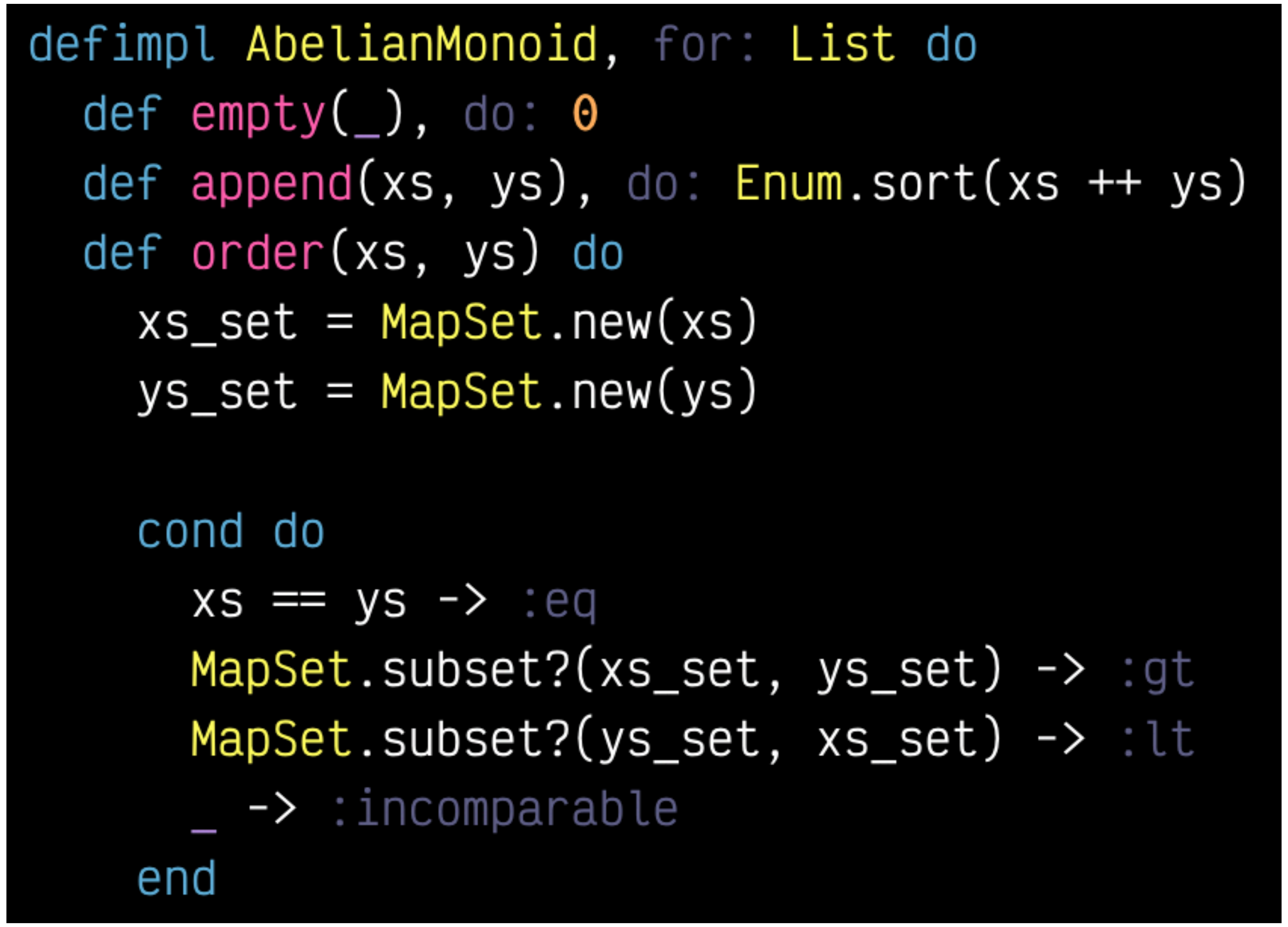
Commutative Monoid (AKA Minimal CRDT)



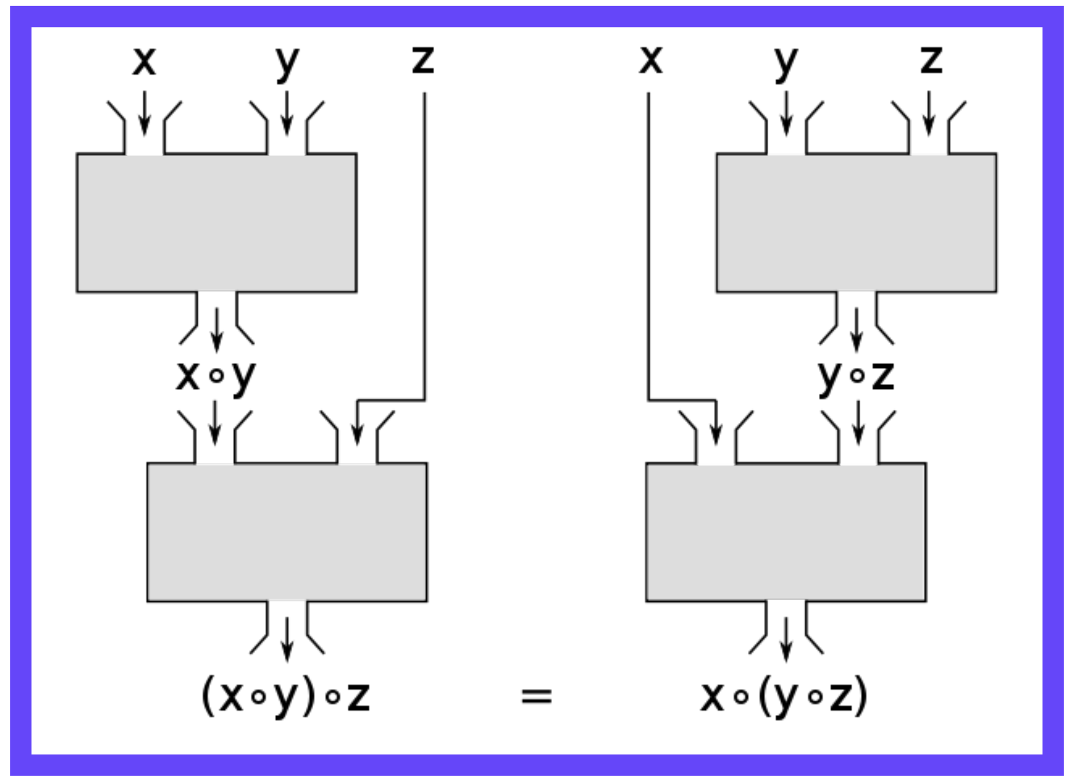
```
defprotocol AbelianMonoid do
  def empty(a)
  def append(a, b)
  def order(a, b)
end
```

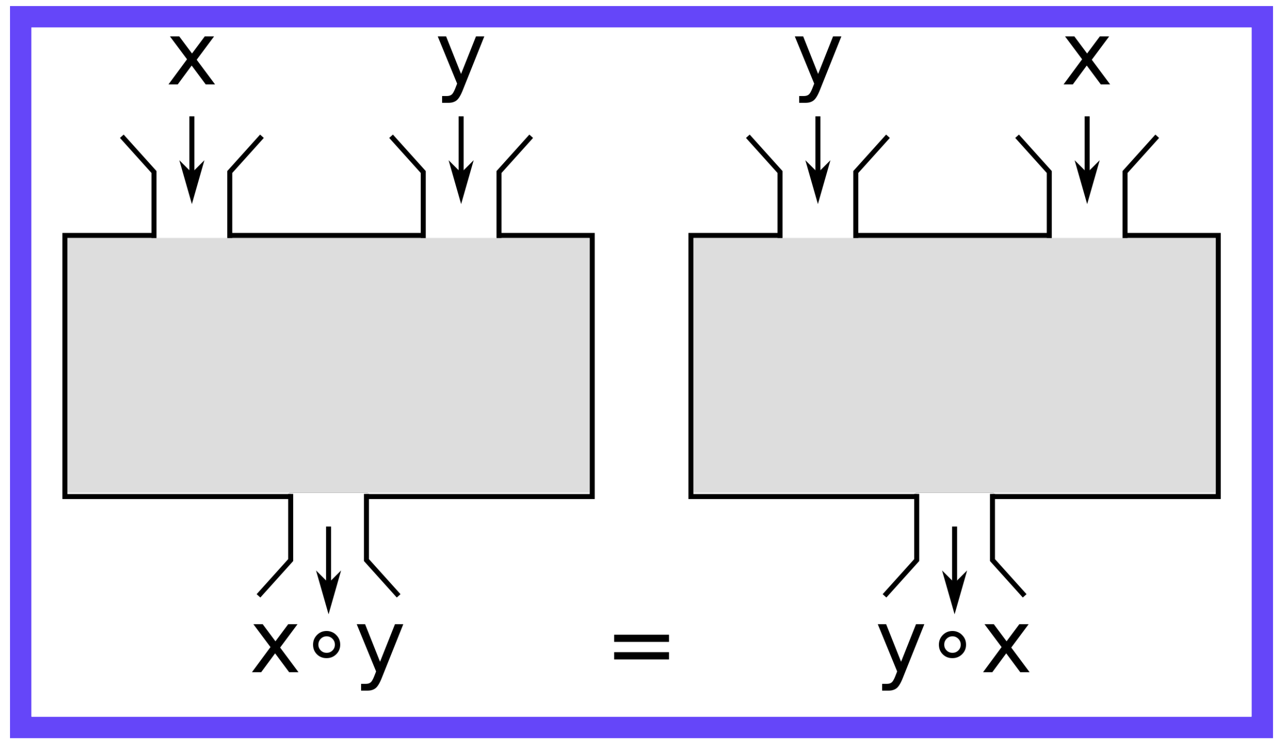


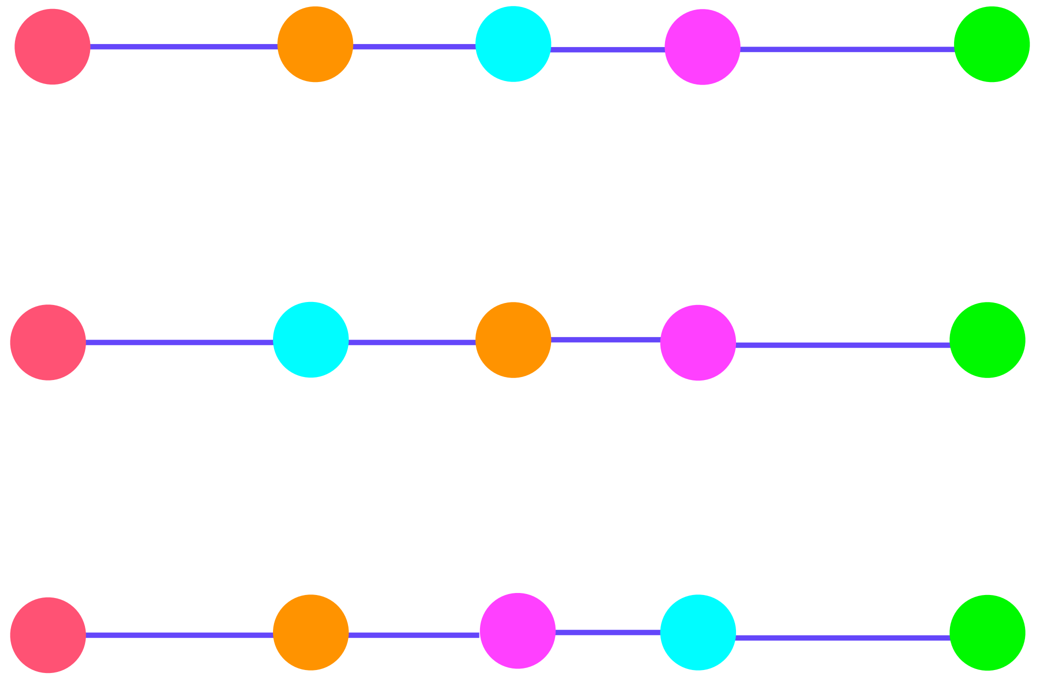
```
defimpl AbelianMondoid, for: Integer do
  def empty(_), do: 0
  def append(a, b), do: a + b
  def order(a, b)
  cond
    a == b -> :eq
    a <  b -> :lt
    a >  b -> :gt
  end
end
```



```
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
```

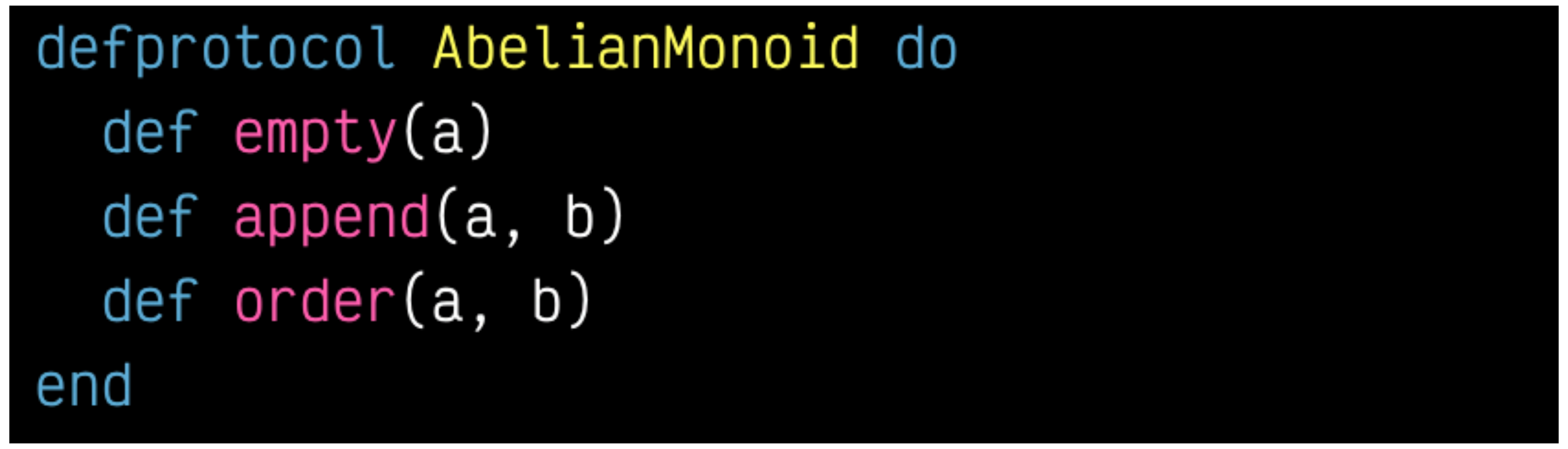




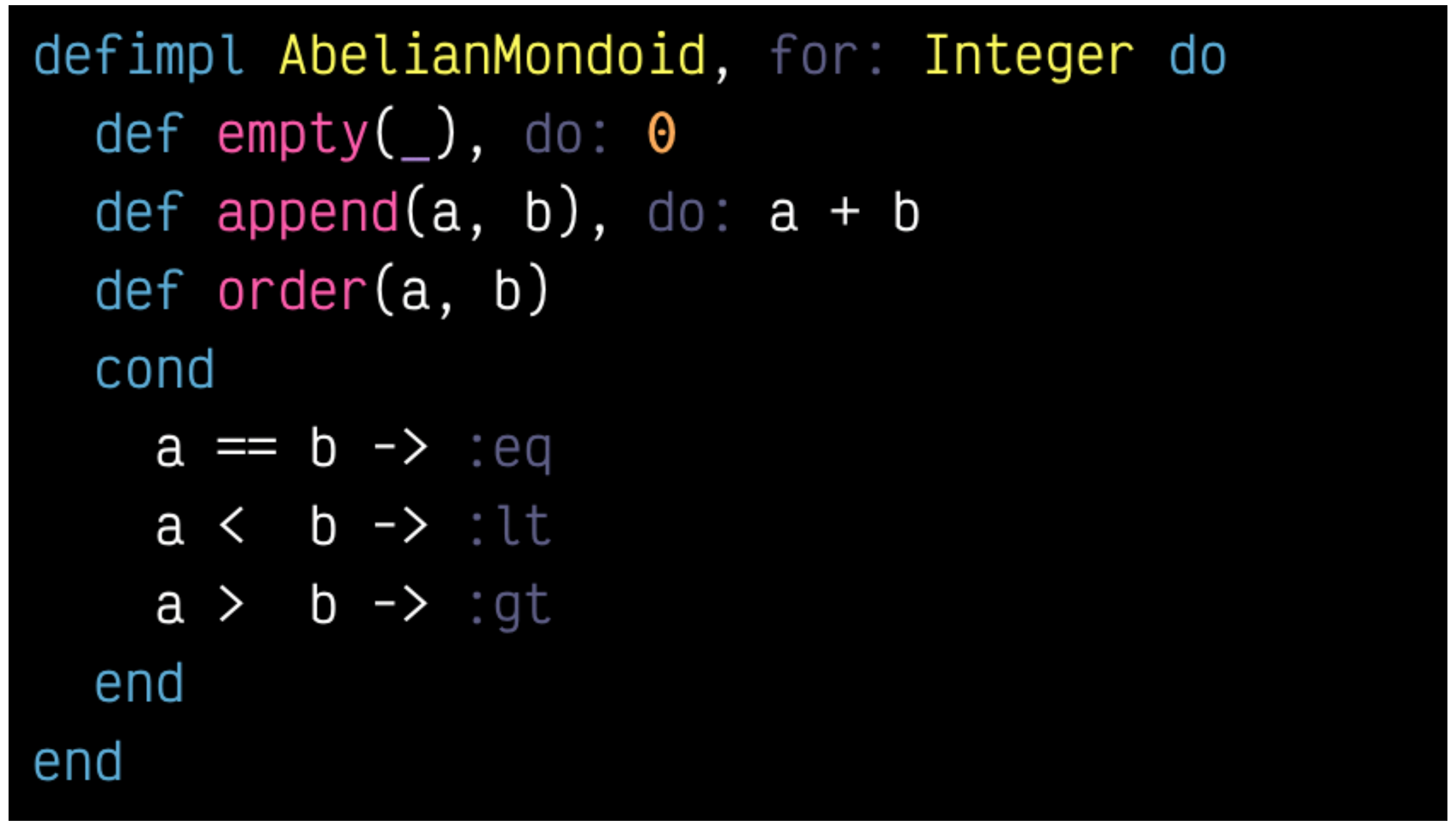


It's All About the Data 📊

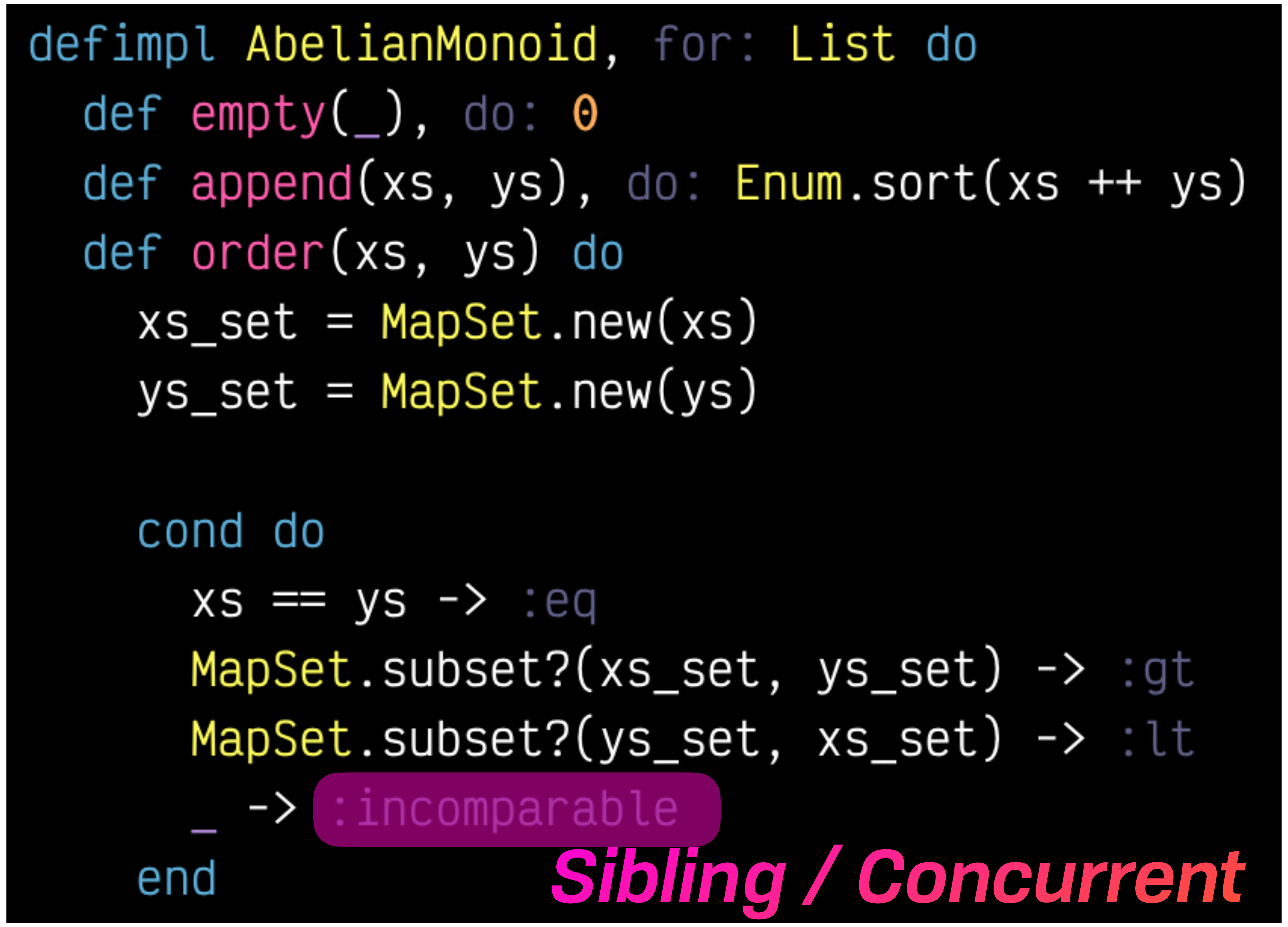
Commutative Monoid (AKA Minimal CRDT)



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

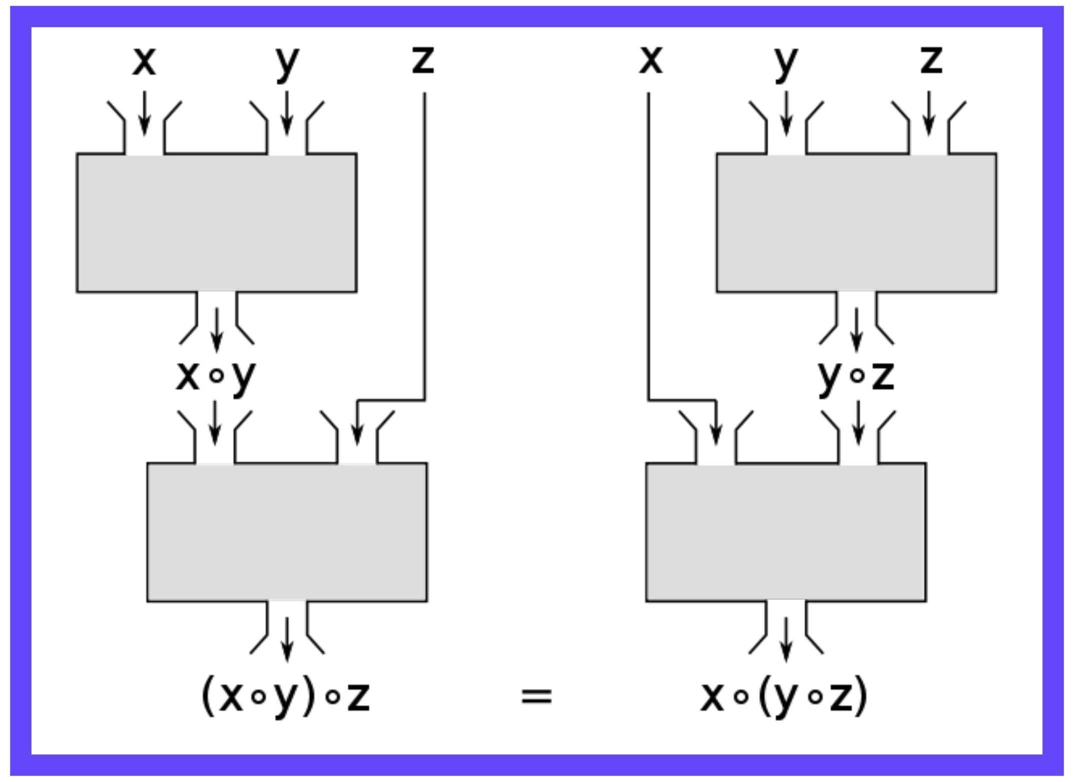


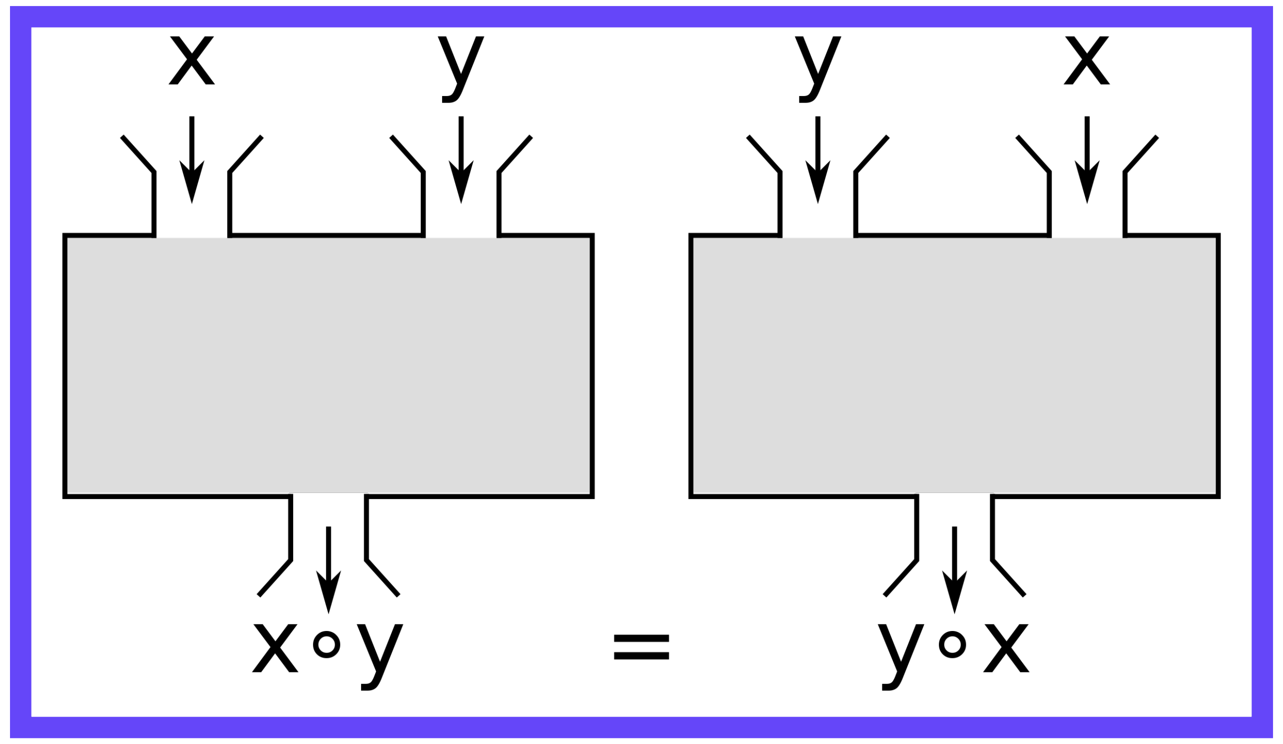
```elixir
defimpl AbelianMondoid, for: Integer do
  def empty(_), do: 0
  def append(a, b), do: a + b
  def order(a, b)
  cond
    a == b -> :eq
    a <  b -> :lt
    a >  b -> :gt
  end
end
```

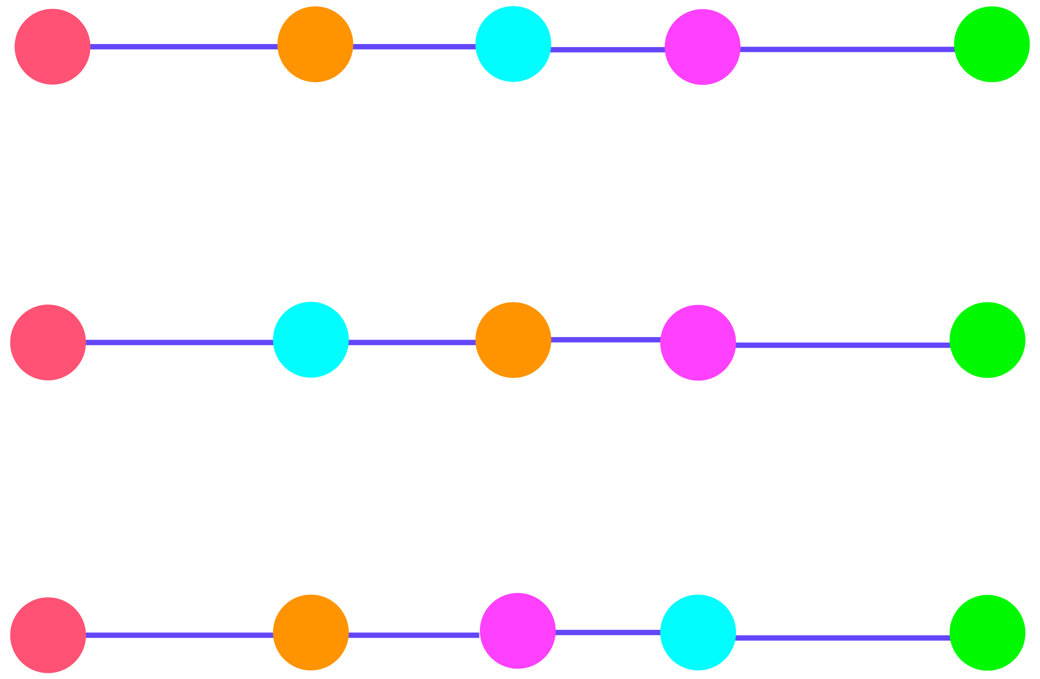


```elixir
defimpl AbelianMonoid, for: List do
  def empty(_), do: 0
  def append(xs, ys), do: Enum.sort(xs ++ ys)
  def order(xs, ys) do
    xs_set = MapSet.new(xs)
    ys_set = MapSet.new(ys)

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

Sibling / Concurrent





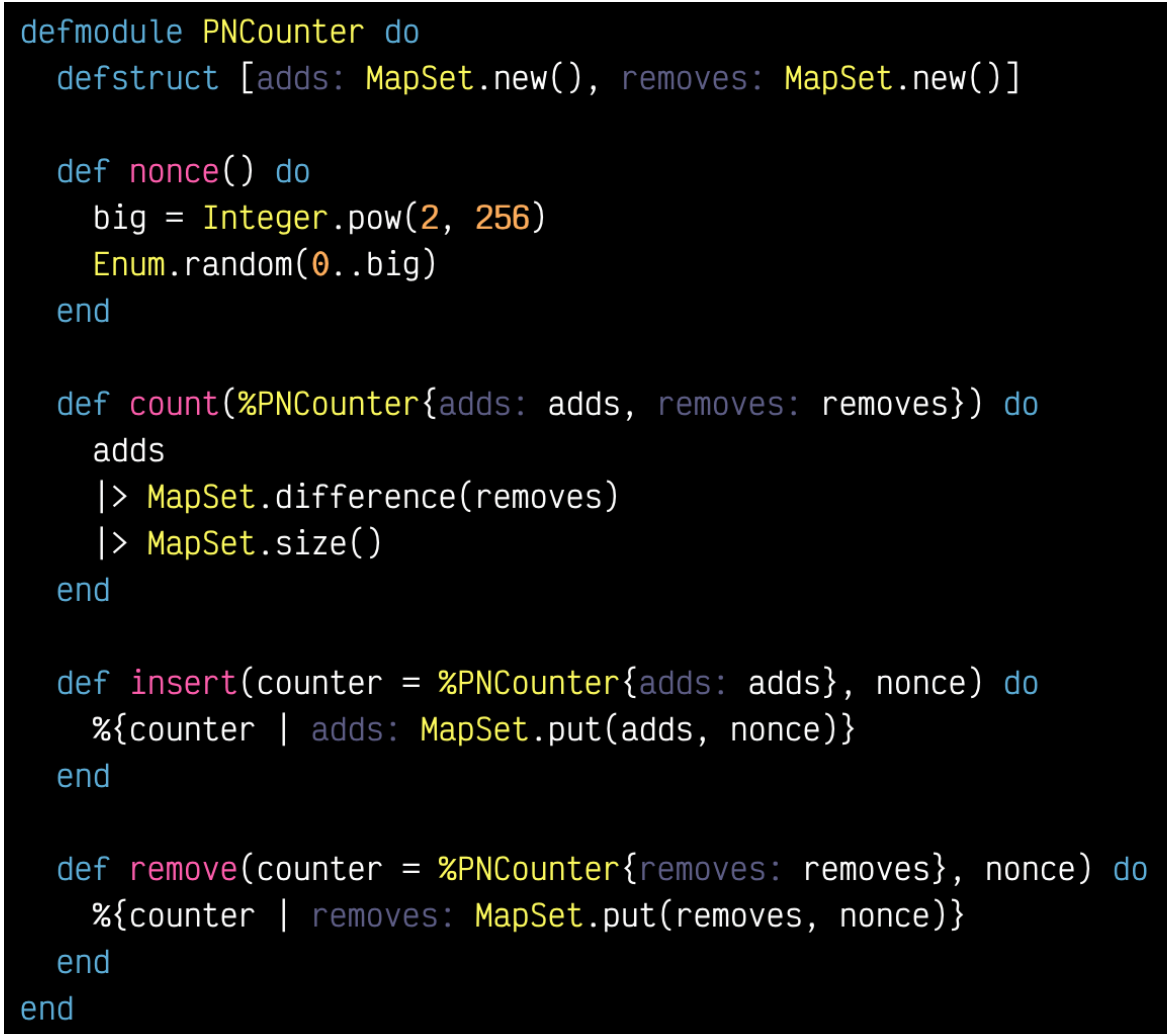


It’s All About the Data 📊

PNCounter

It’s All About the Data 📊

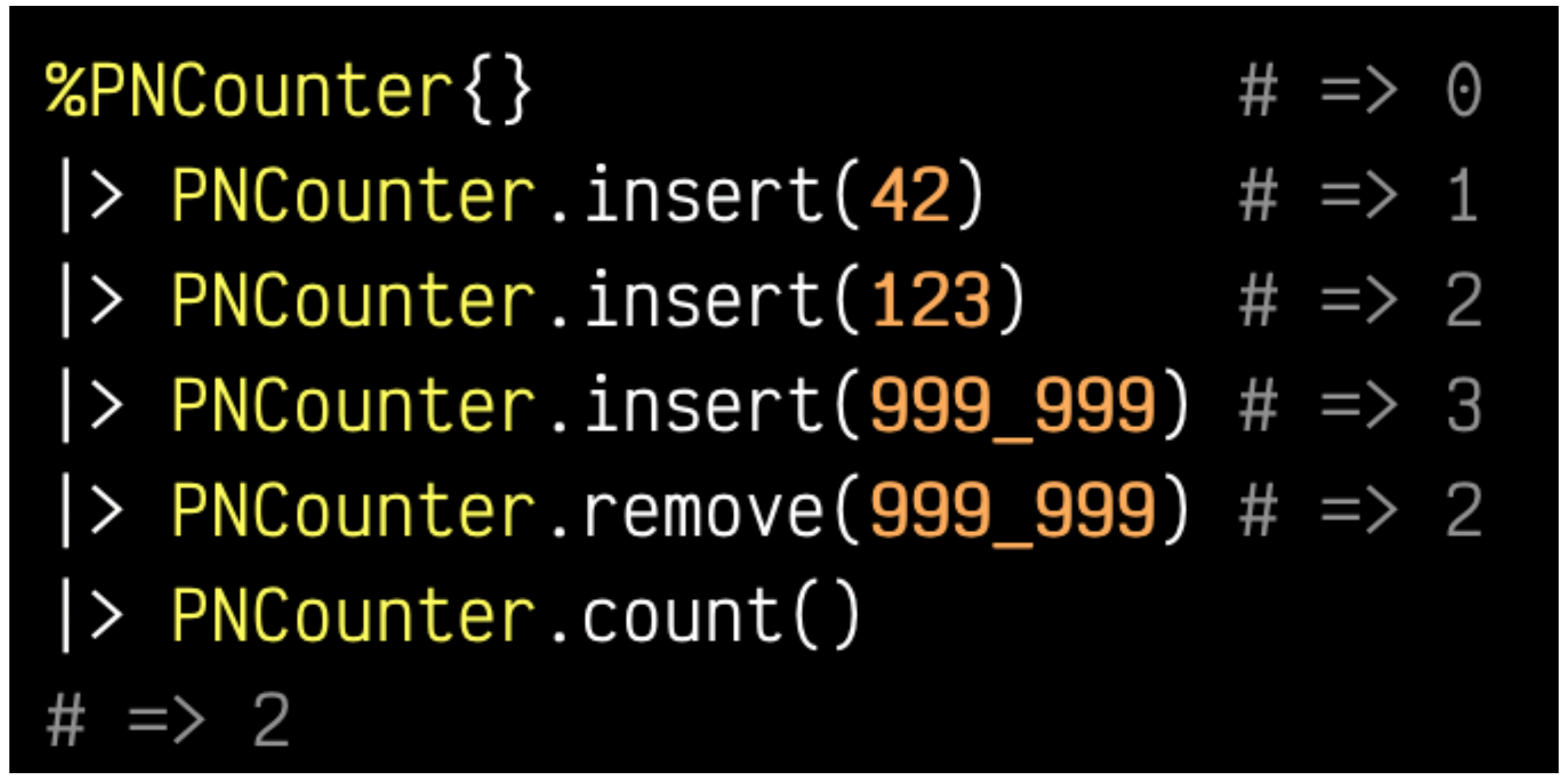
PNCounter



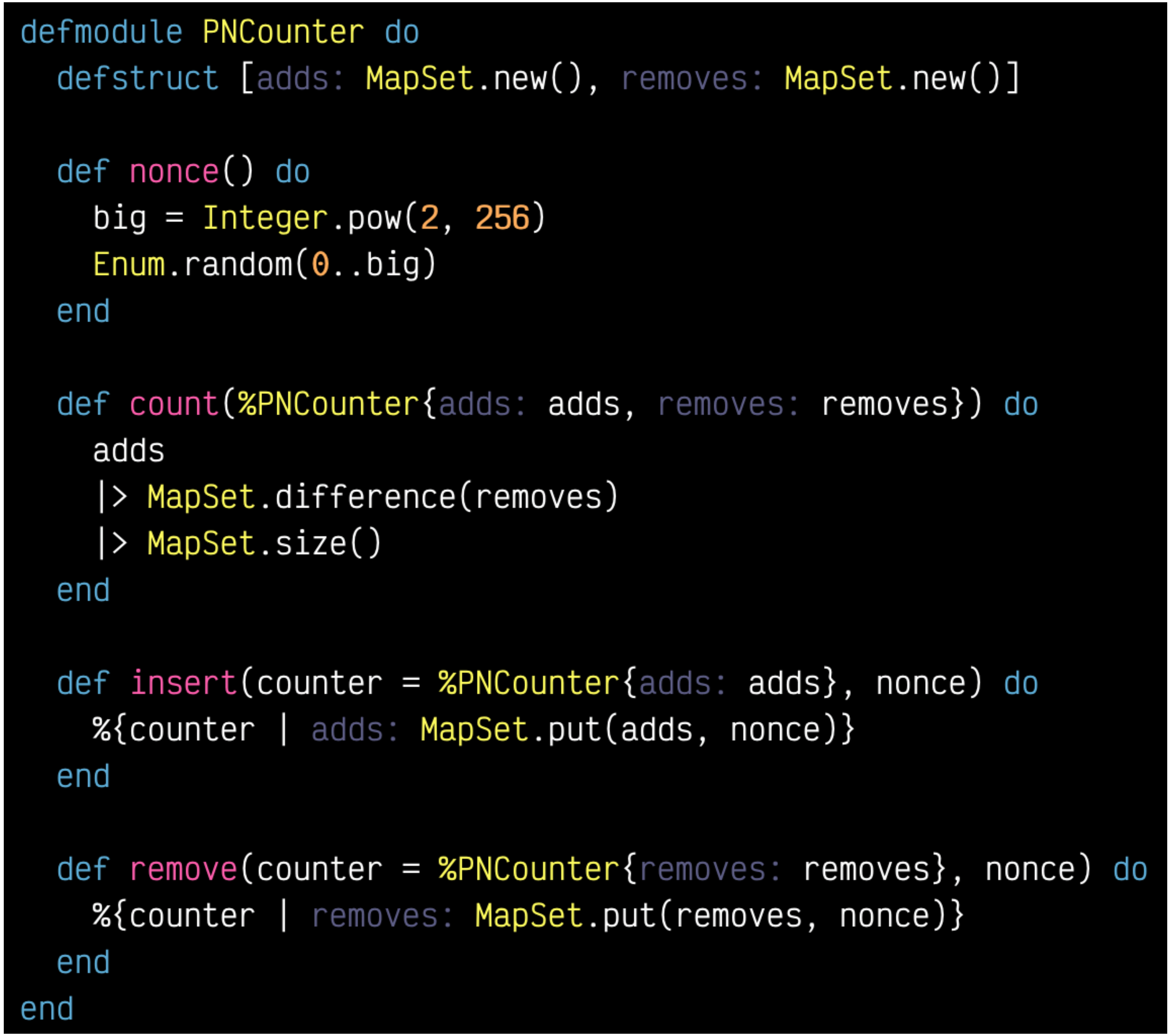
```
defmodule PNCounter do
  defstruct [adds: MapSet.new(), removes: MapSet.new()]

  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 the Data 📊

PNCounter



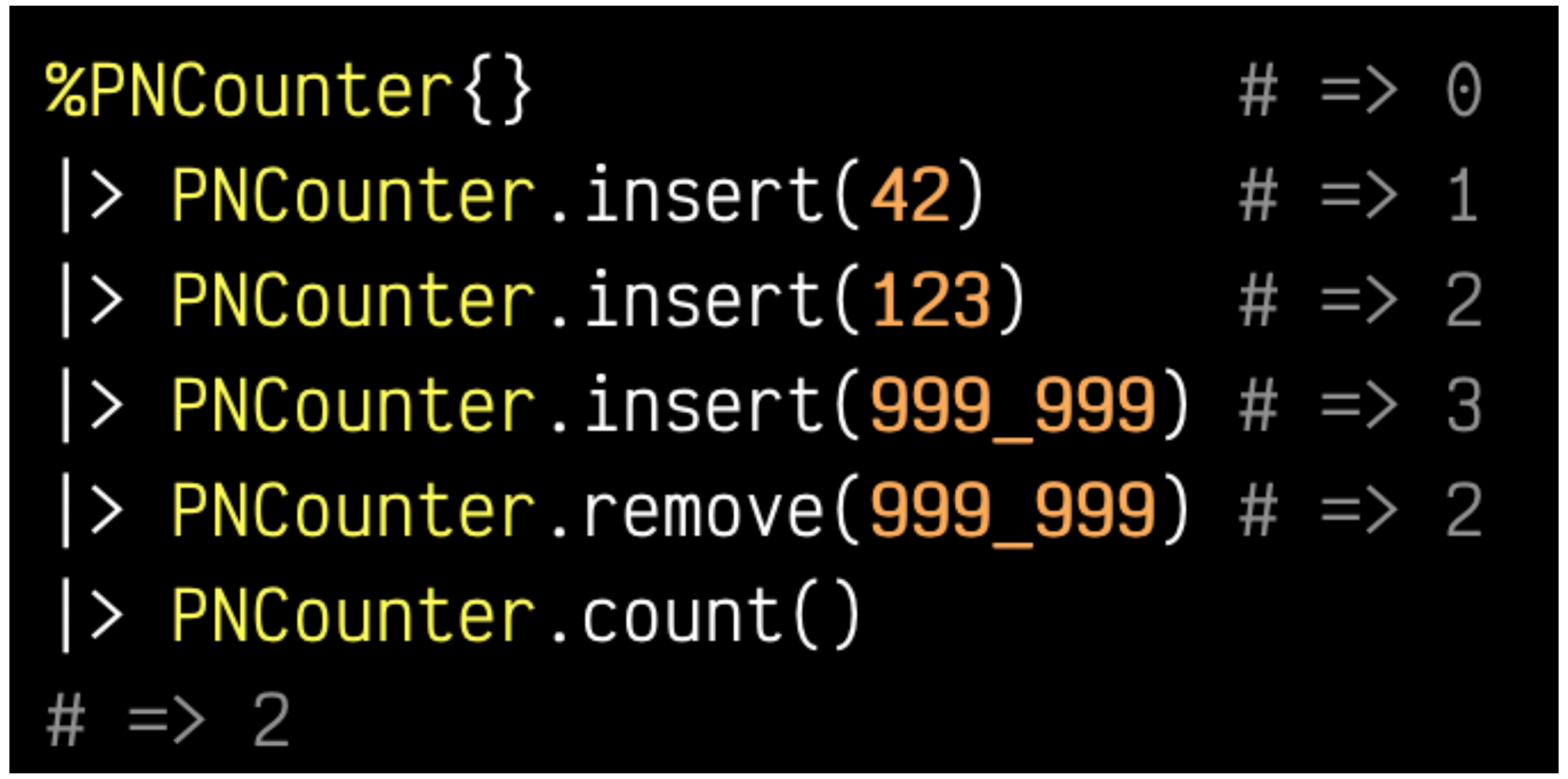
```
%PNCounter{}                 # => 0
|> PNCounter.insert(42)      # => 1
|> PNCounter.insert(123)     # => 2
|> PNCounter.insert(999_999) # => 3
|> PNCounter.remove(999_999) # => 2
|> PNCounter.count()
# => 2
```



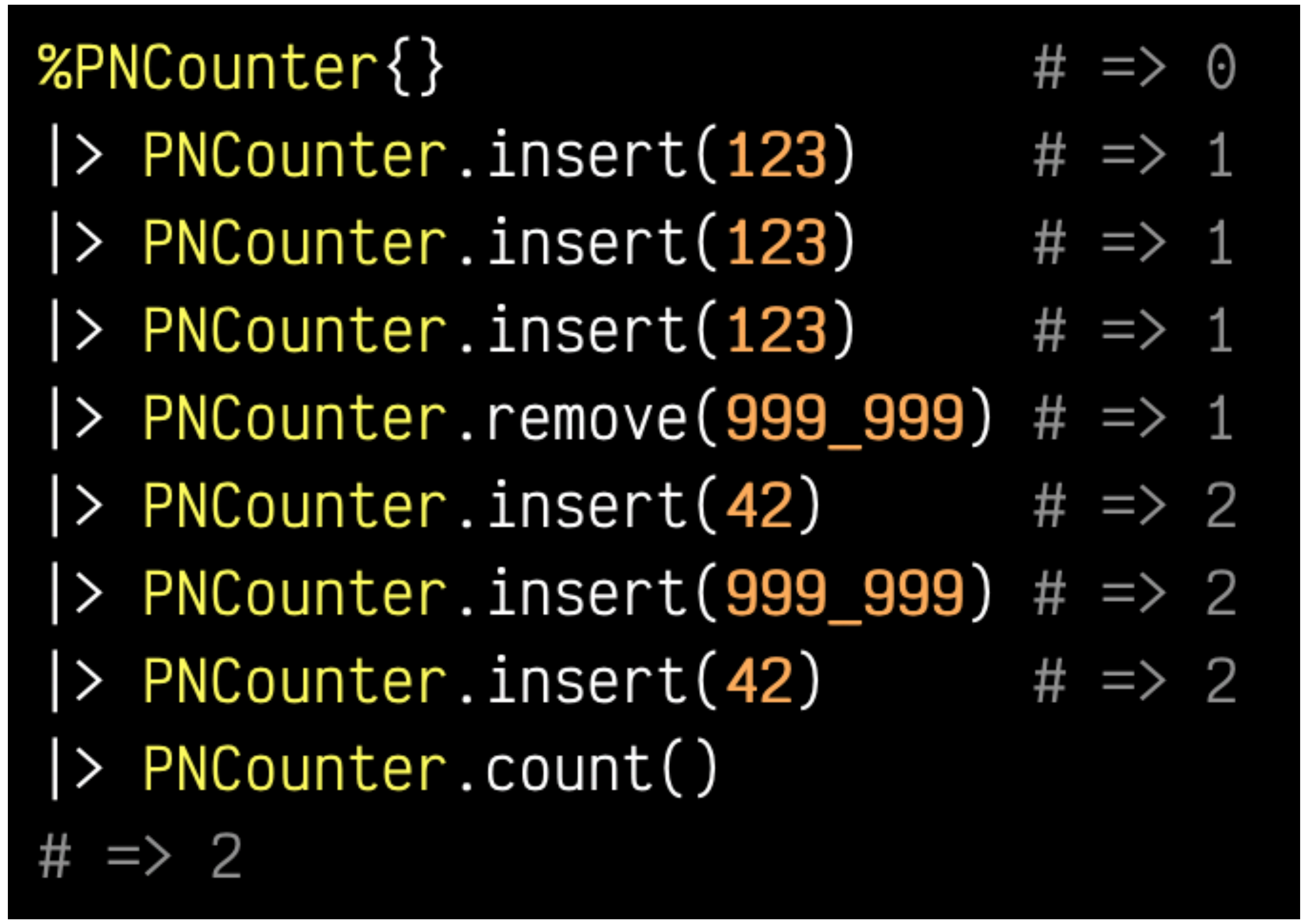
```
defmodule PNCounter do
  defstruct [adds: MapSet.new(), removes: MapSet.new()]

  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 the Data 📊

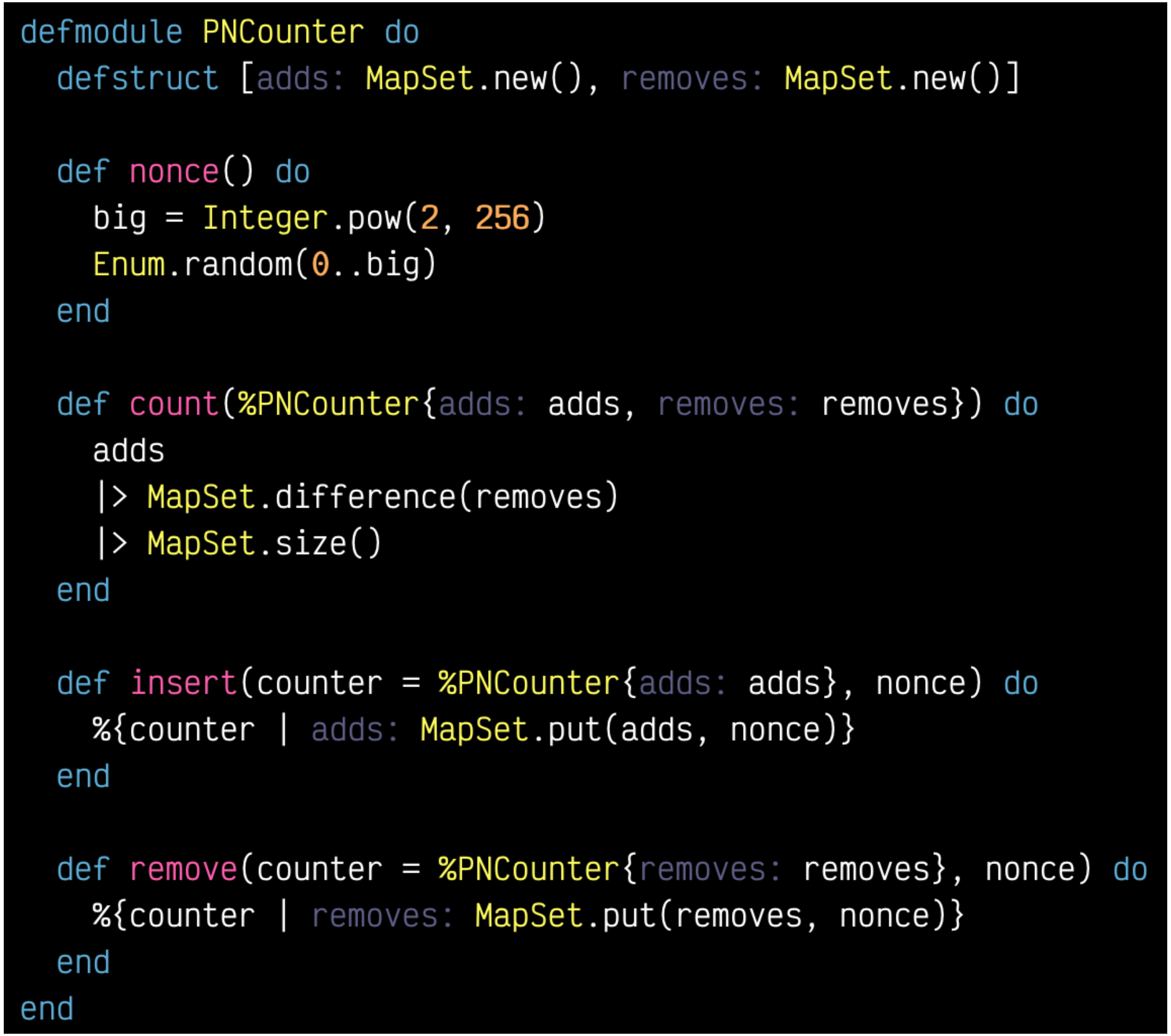
PNCounter



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



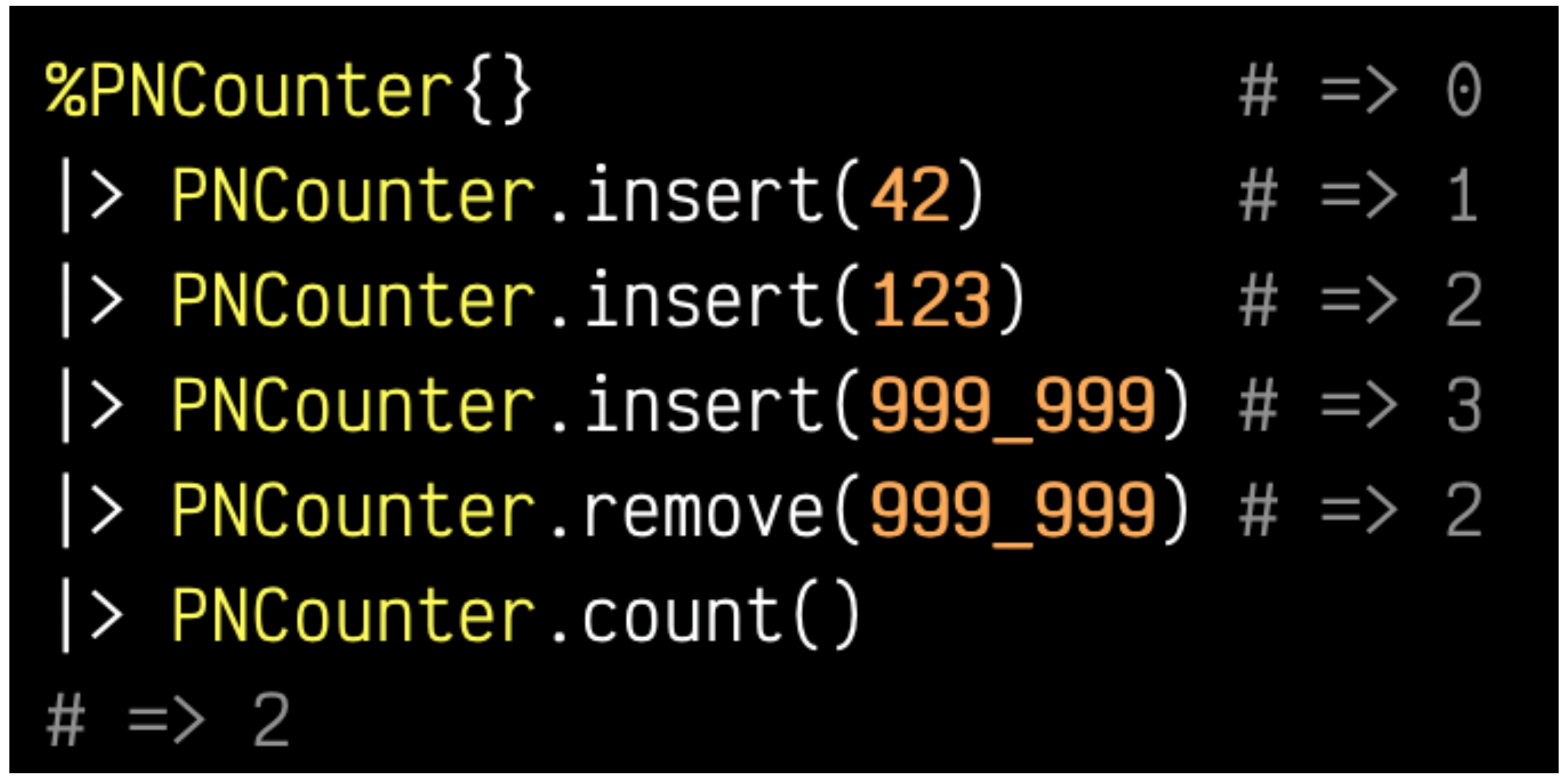
```elixir
%PNCounter{}                # => 0
|> PNCounter.insert(123)    # => 1
|> PNCounter.insert(123)    # => 1
|> PNCounter.insert(123)    # => 1
|> PNCounter.remove(999_999) # => 1
|> PNCounter.insert(42)     # => 2
|> PNCounter.insert(999_999) # => 2
|> PNCounter.insert(42)     # => 2
|> PNCounter.count()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



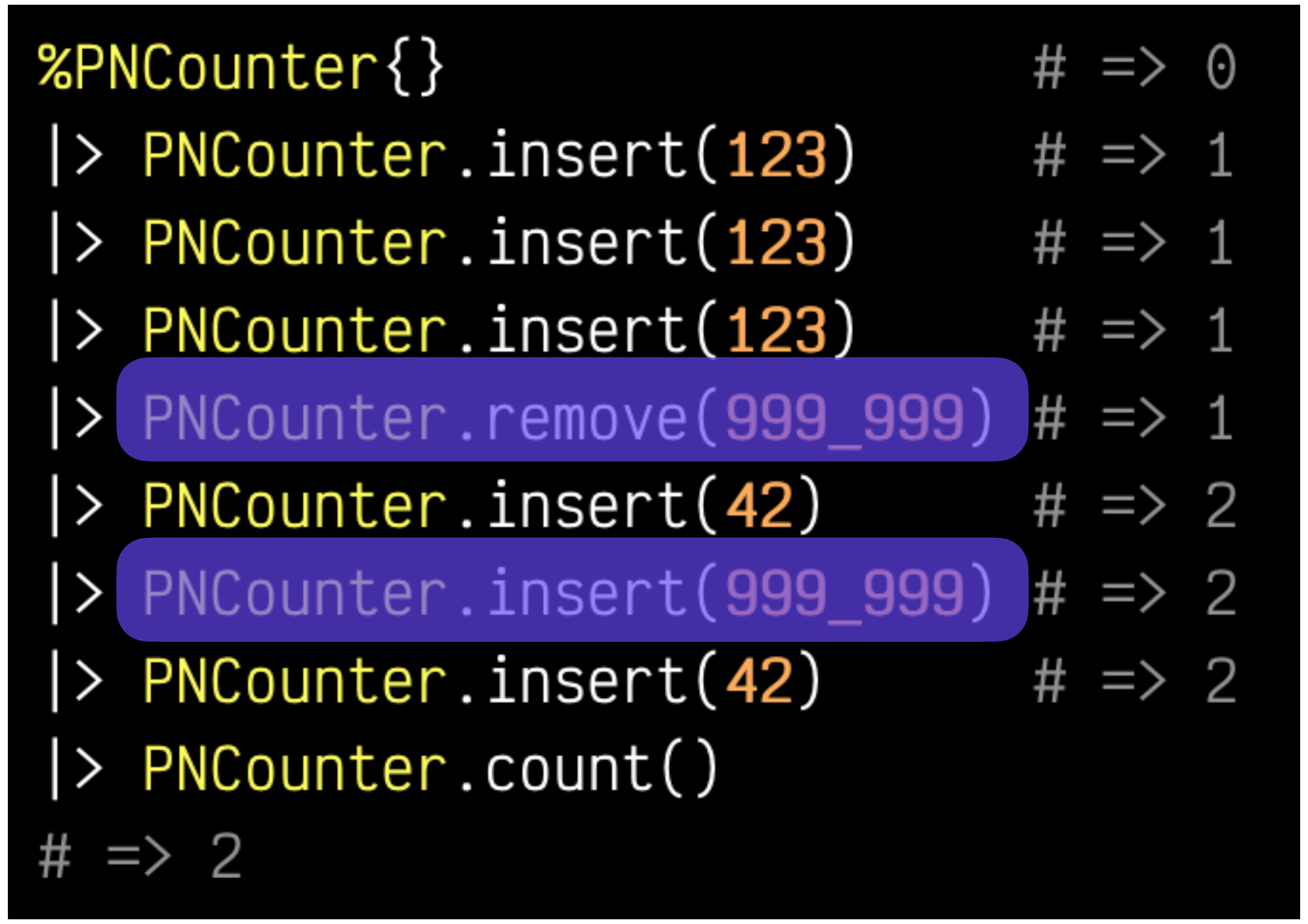
```elixir
defmodule PNCounter do
  defstruct [adds: MapSet.new(), removes: MapSet.new()]

  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 the Data 📊

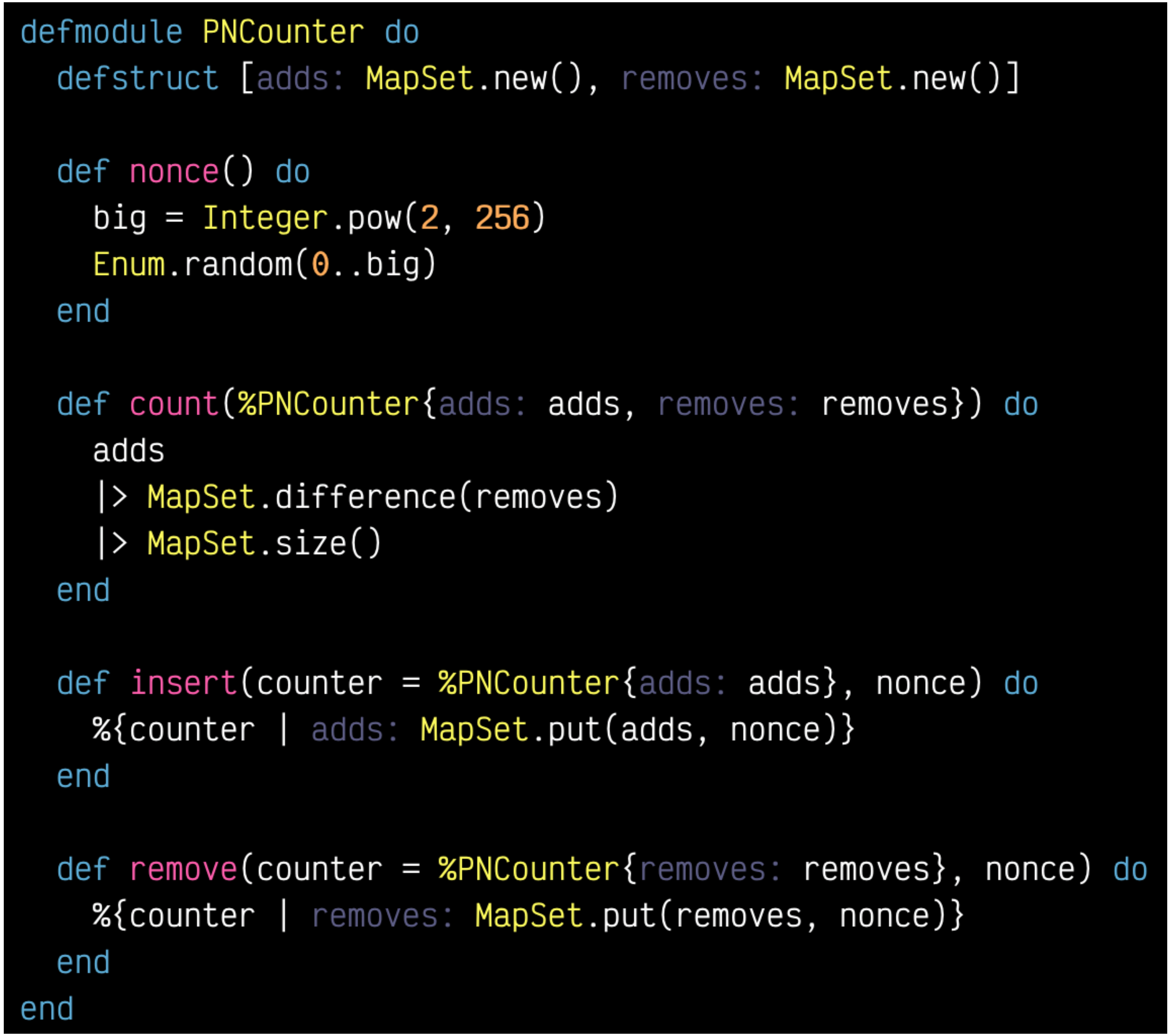
PNCounter



```
%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
```



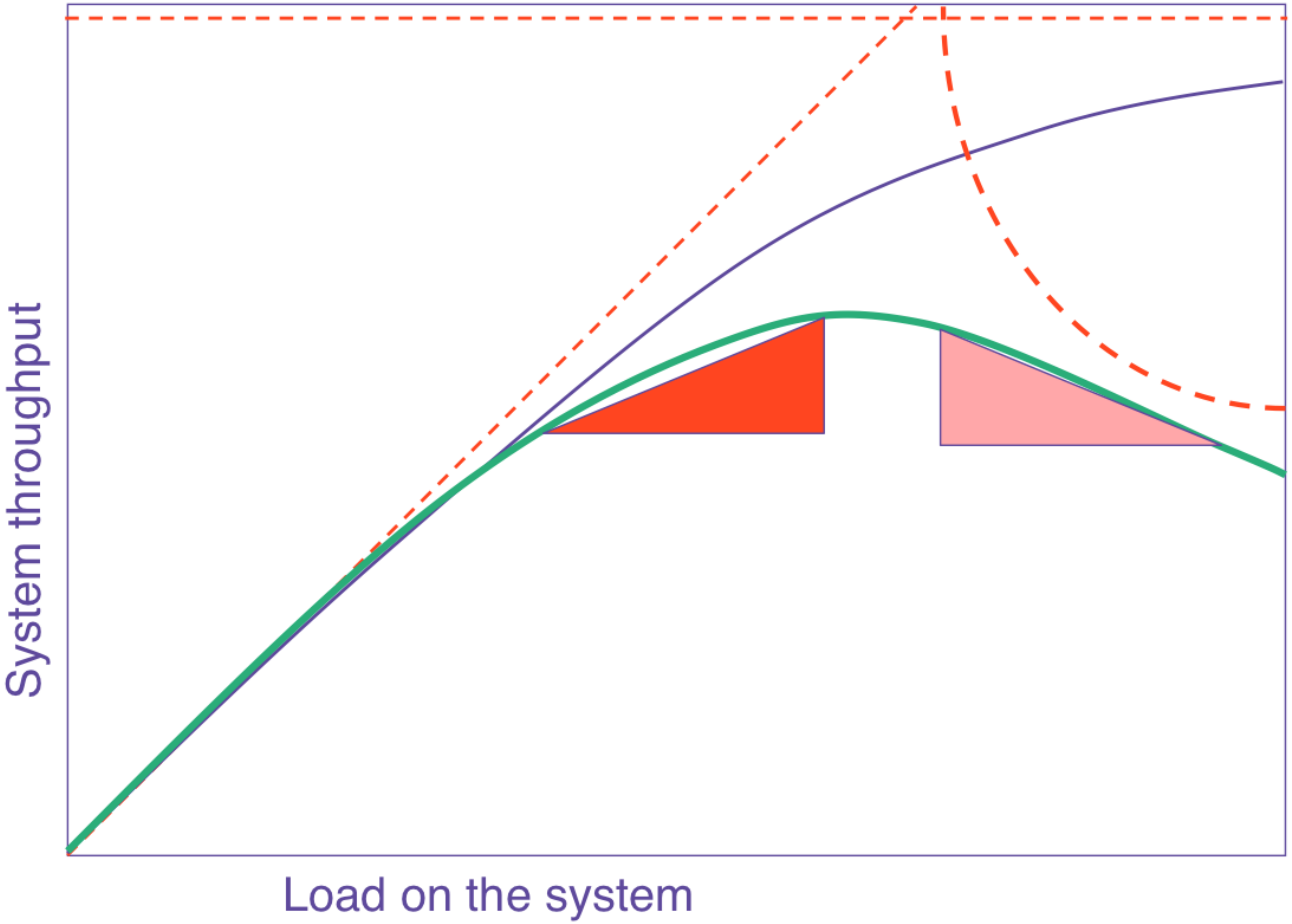
```
defmodule PNCounter do
  defstruct [adds: MapSet.new(), removes: MapSet.new()]

  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
```

The Age of
Decentralized Systems

🌈

Decentralized Systems 🌈

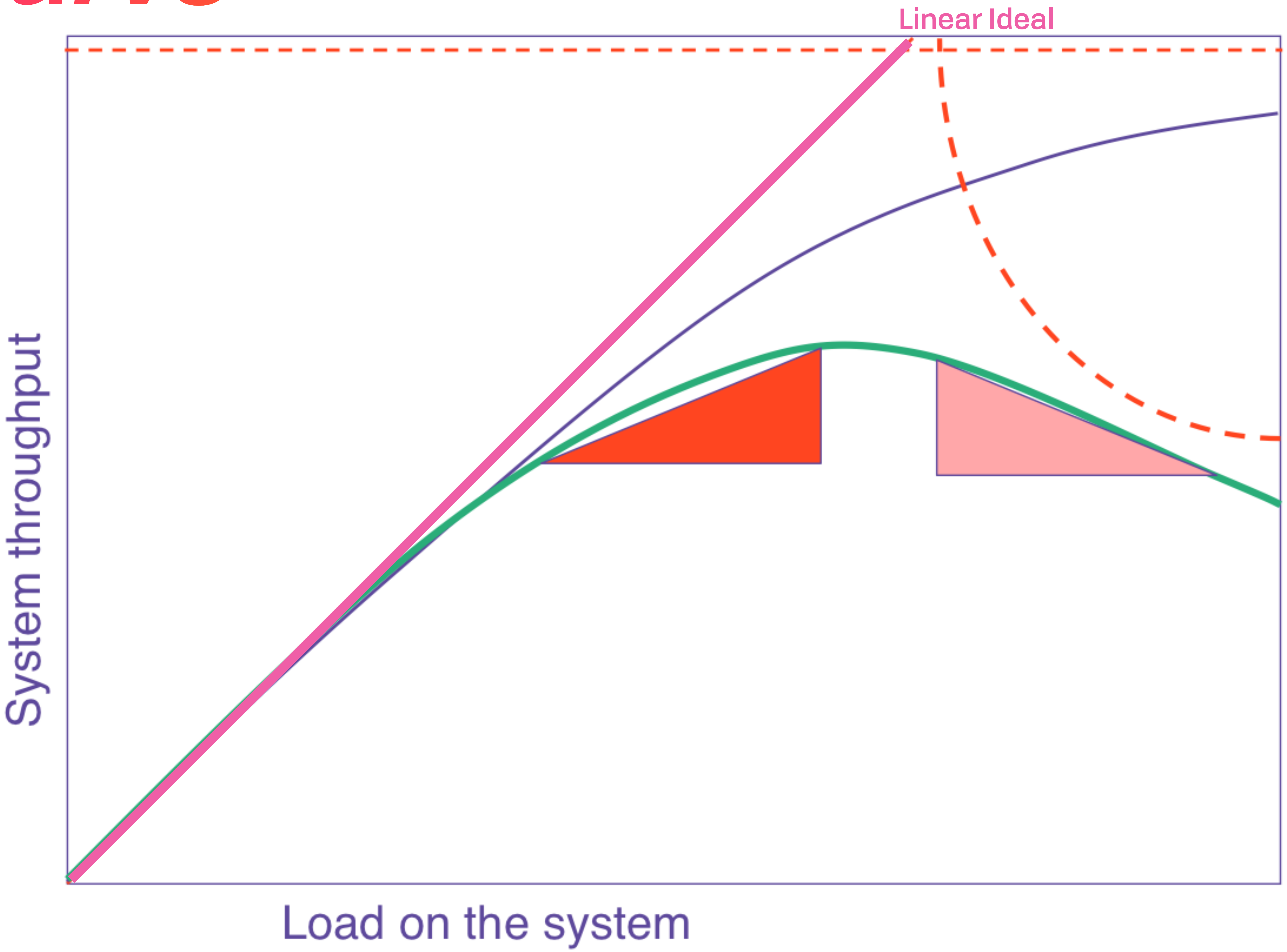
Scale Curve



Adapted from http://www.perfdynamics.com/Manifesto/USLscalability.html

Decentralized Systems 🌈

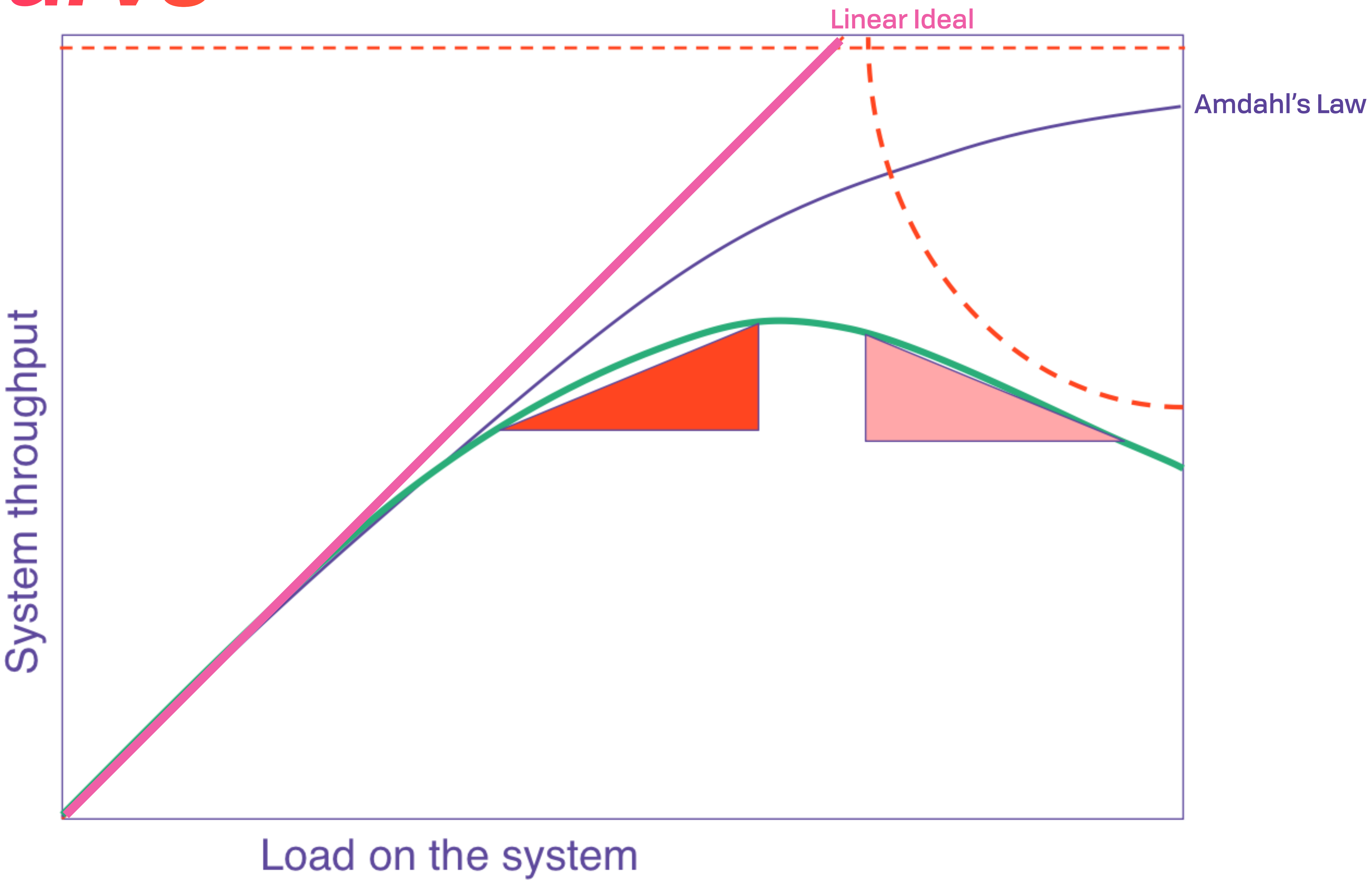
Scale Curve



Adapted from http://www.perfdynamics.com/Manifesto/USLscalability.html

Decentralized Systems 🌈

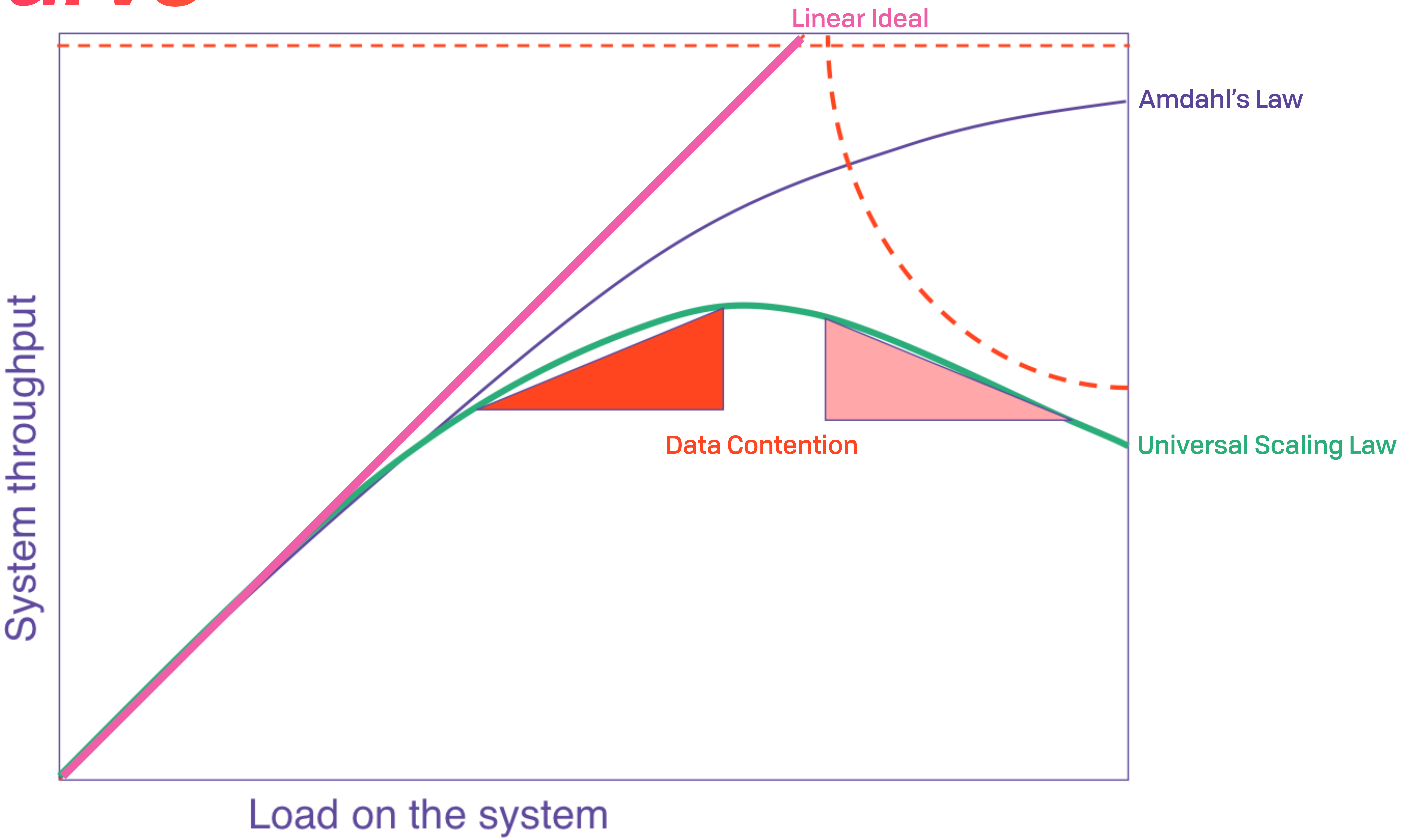
Scale Curve



Adapted from http://www.perfdynamics.com/Manifesto/USLscalability.html

Decentralized Systems 🌈

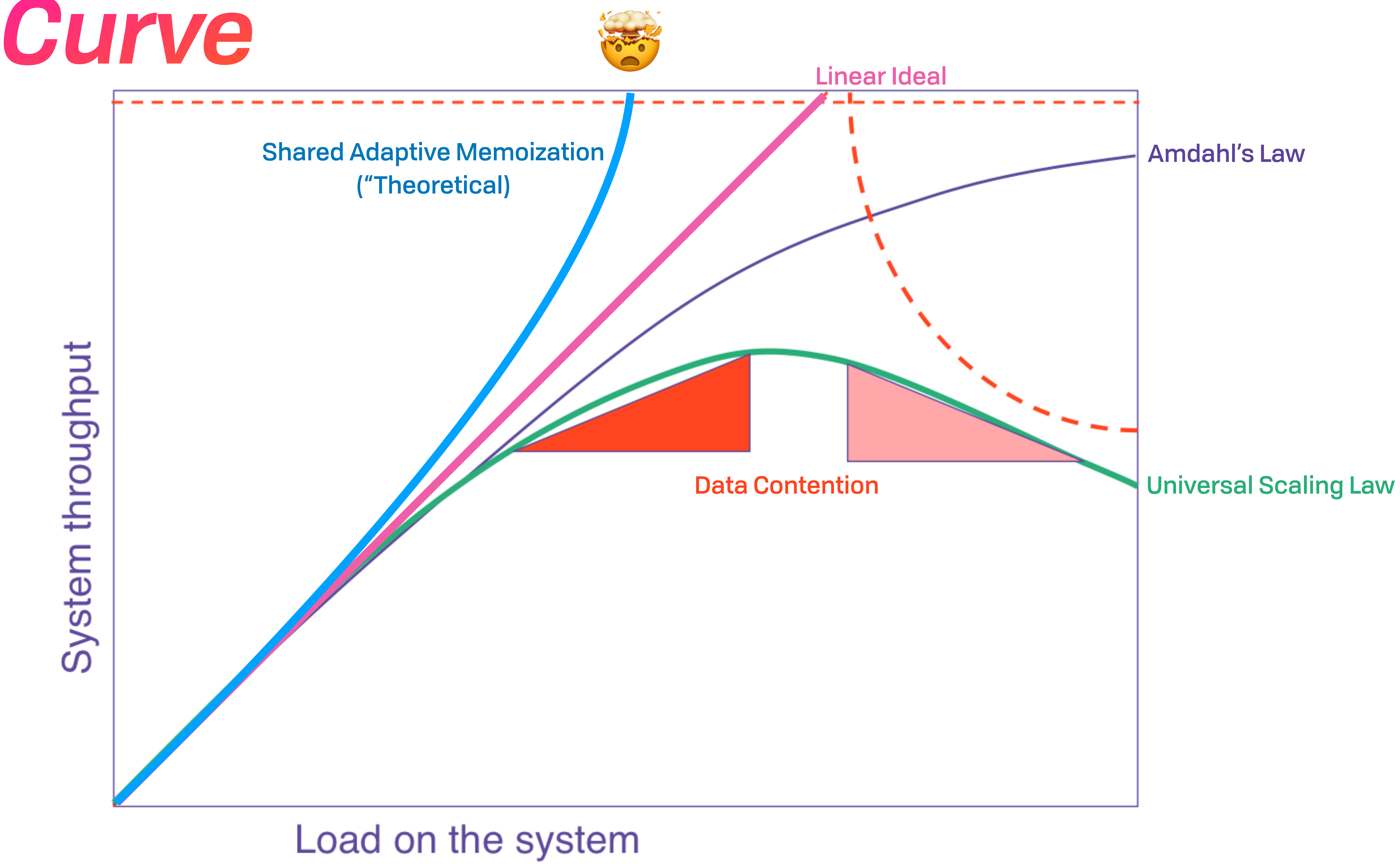
Scale Curve



Adapted from http://www.perfdynamics.com/Manifesto/USLscalability.html

Decentralized Systems 🌈

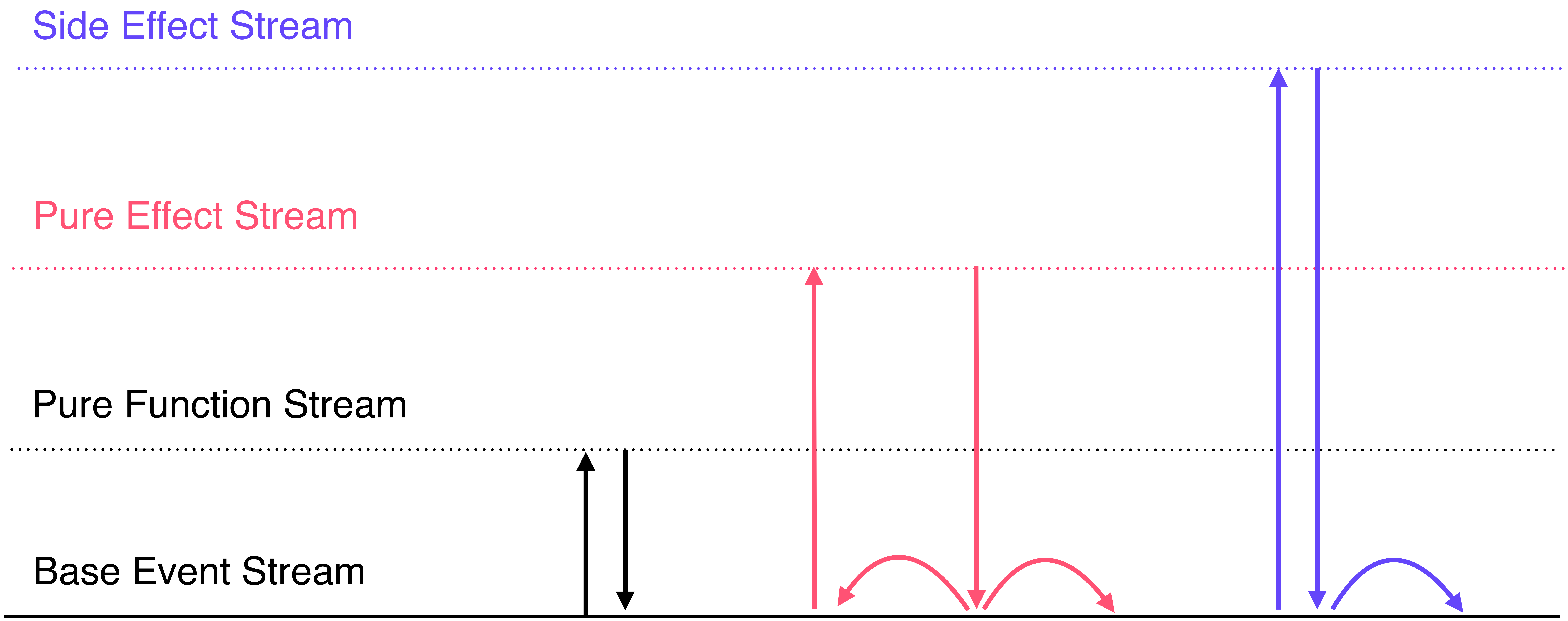
Scale Curve



Adapted from http://www.perfdynamics.com/Manifesto/USLscalability.html

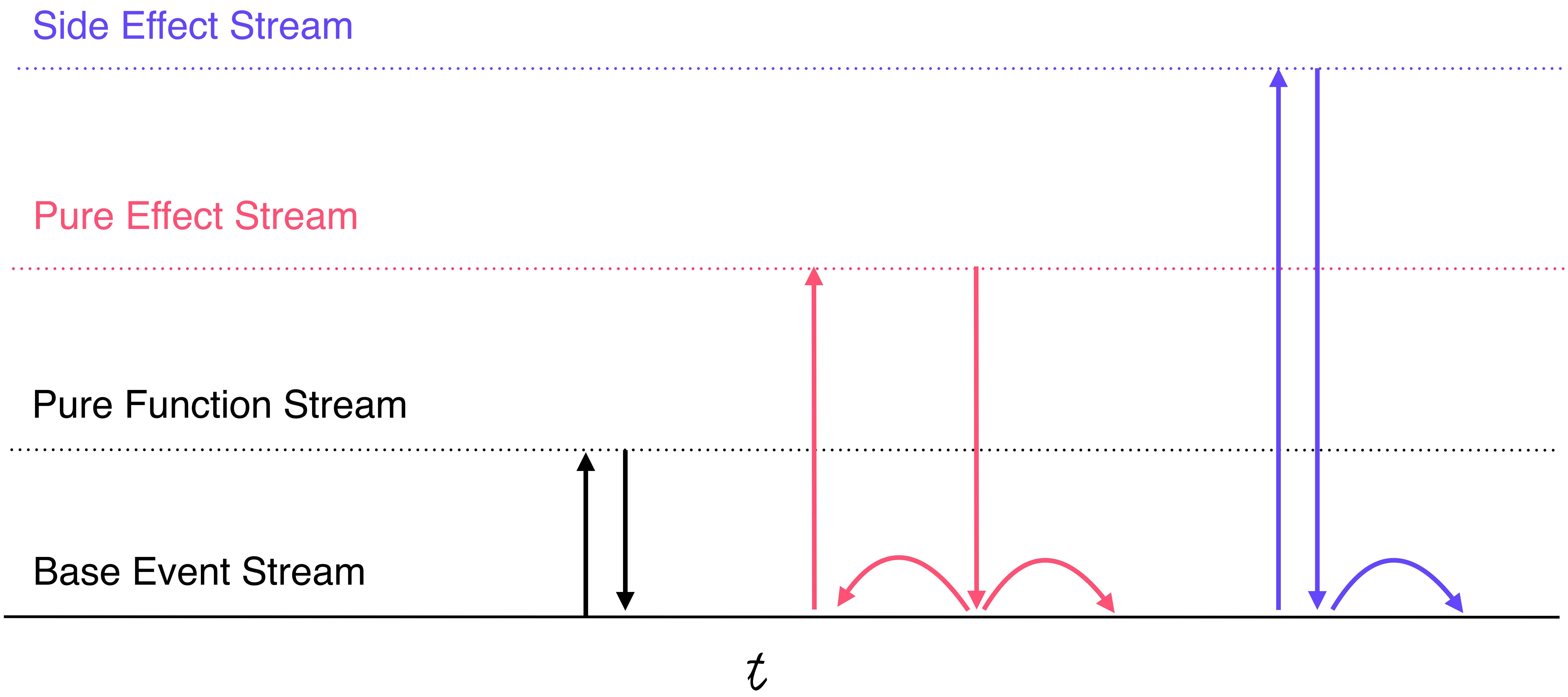
Decentralized Systems 🌈

Conflict Free Effects 🕊️🧱



Decentralized Systems 🌈

Conflict Free Effects 🕊️🧱



Side Effect Stream

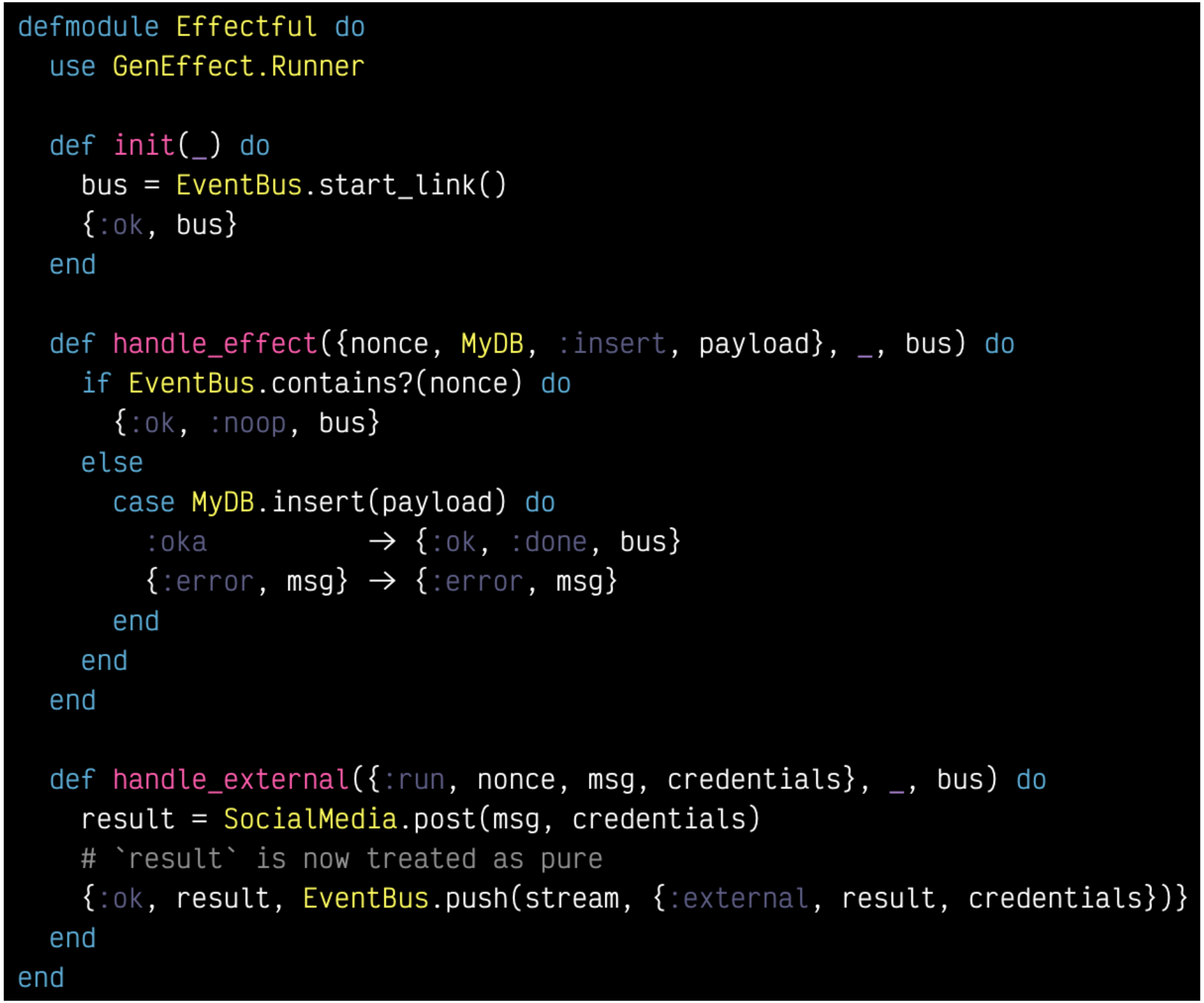
Pure Effect Stream

Pure Function Stream

Base Event Stream

Decentralized Systems 🌈

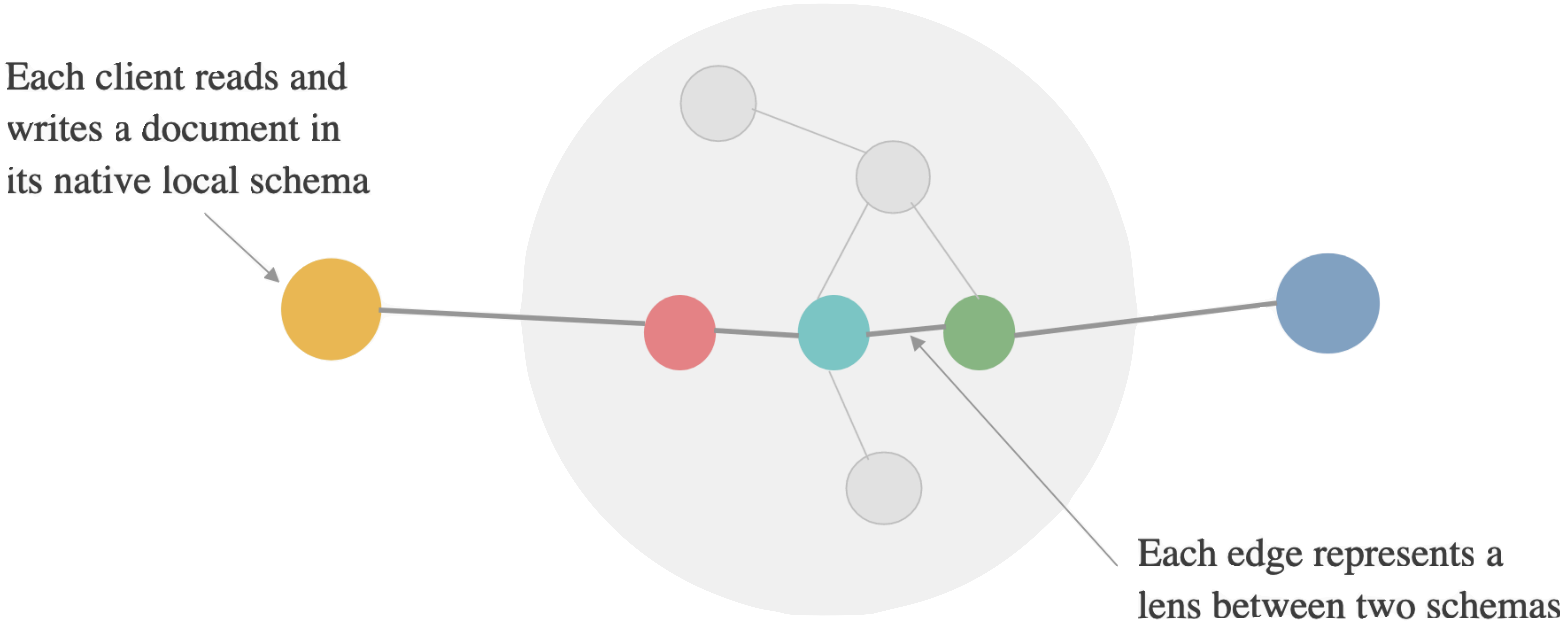
GenEffect 🚀



```elixir
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
```

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

Source: Project Cambria, Ink & Switch
https://www.inkandswitch.com/cambria.html

Secure Decentralized Data Access

Fixing the Leaky Pipes

🚿

Fixing the Leaky Pipes 🚿

Object Capability Model (OCAP)

Fixing the Leaky Pipes 🚿

Object Capability Model (OCAP)

- ACL is “reactive auth” / OCAP is “proactive auth”

Fixing the Leaky Pipes 🚿

Object Capability Model (OCAP)

- ACL is “reactive auth” / OCAP is “proactive auth”
- OCAP contains all the info about access

Object Capability Model (OCAP)

- ACL is “reactive auth” / OCAP is “proactive auth”
- OCAP contains all the info about access
- Generally some reference, proof, or key
    - …not unlike having a PID
    - Rights to anything directly created (parenthood)
    - The right to delegate subset of access to another (introduction)

Fixing the Leaky Pipes 🚿

Object Capability Model (OCAP)

- ACL is “reactive auth” / OCAP is “proactive auth”
- OCAP contains all the info about access
- Generally some reference, proof, or key
    - …not unlike having a PID
    - Rights to anything directly created (parenthood)
    - The right to delegate subset of access to another (introduction)
- Long history (e.g. X.509, SDSI, SPKI, Macaroons)

Fixing the Leaky Pipes 🚿

3rd-Party Subdelegation & Attenuation

Fixing the Leaky Pipes 🚿

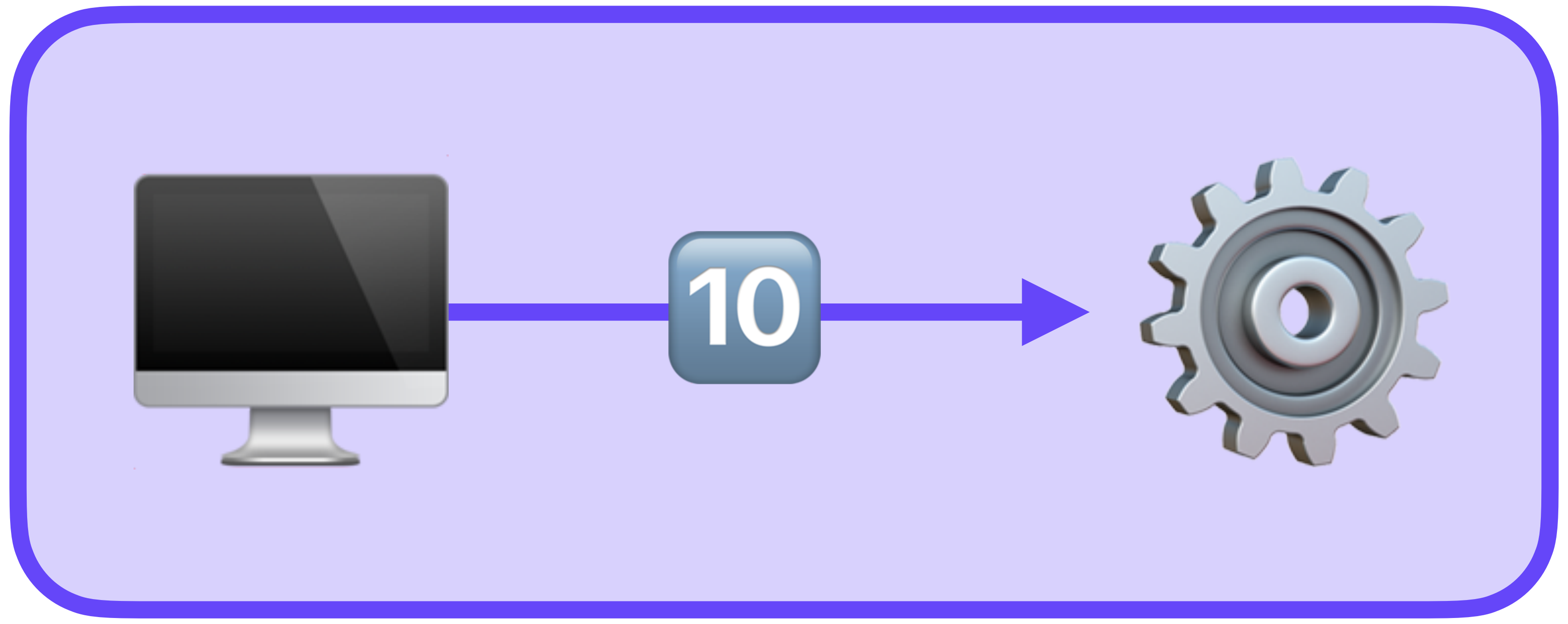
3rd-Party Subdelegation & Attenuation

Fixing the Leaky Pipes 🚿

3rd-Party Subdelegation & Attenuation

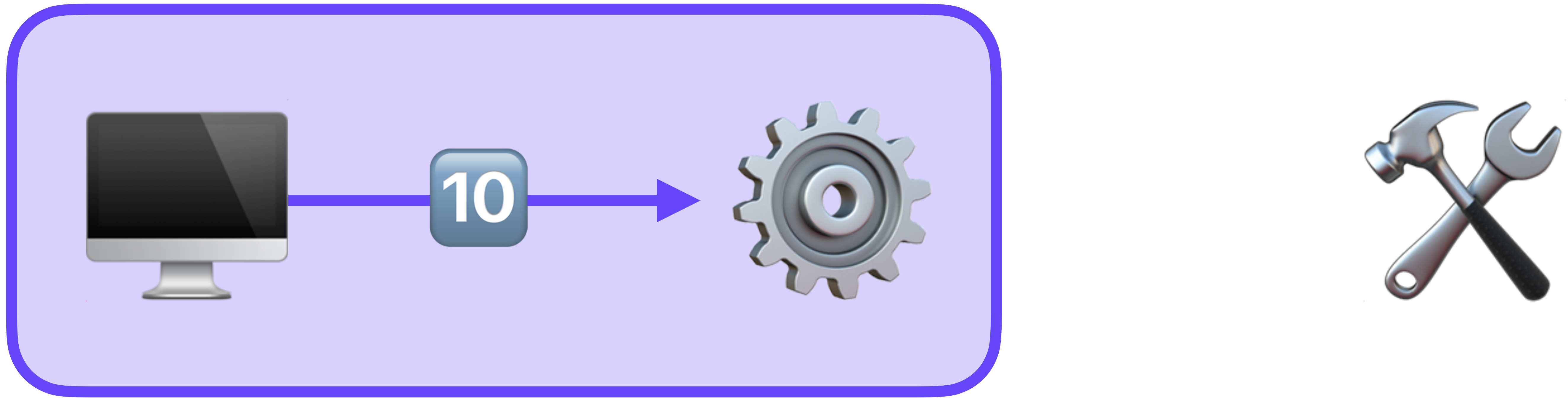
Fixing the Leaky Pipes 🚿

3rd-Party Subdelegation & Attenuation



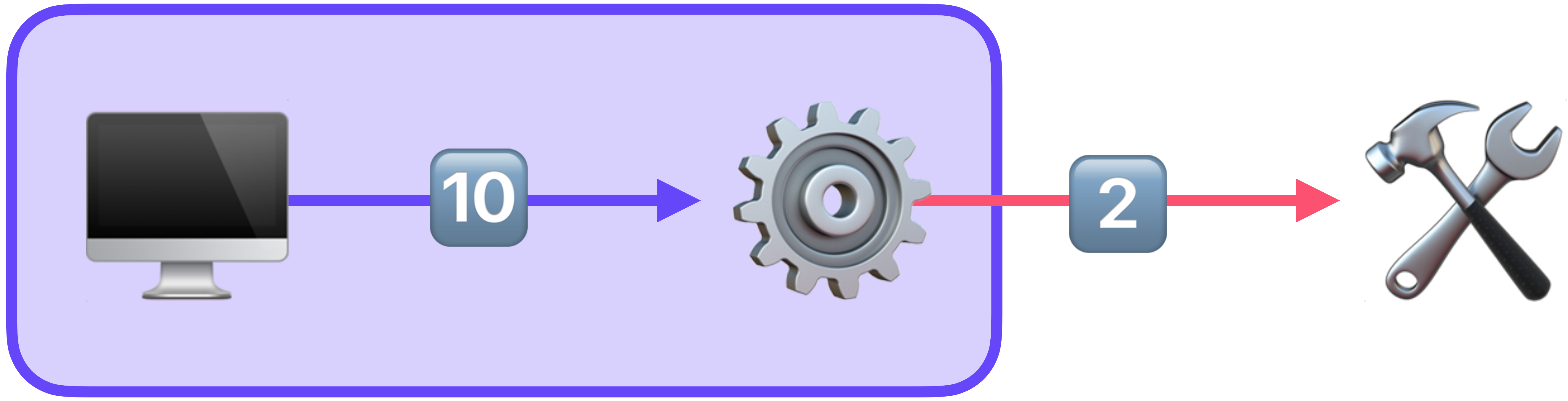
Fixing the Leaky Pipes 🚿

3rd-Party Subdelegation & Attenuation



Fixing the Leaky Pipes 🚿

3rd-Party Subdelegation & Attenuation



Fixing the Leaky Pipes 🚿

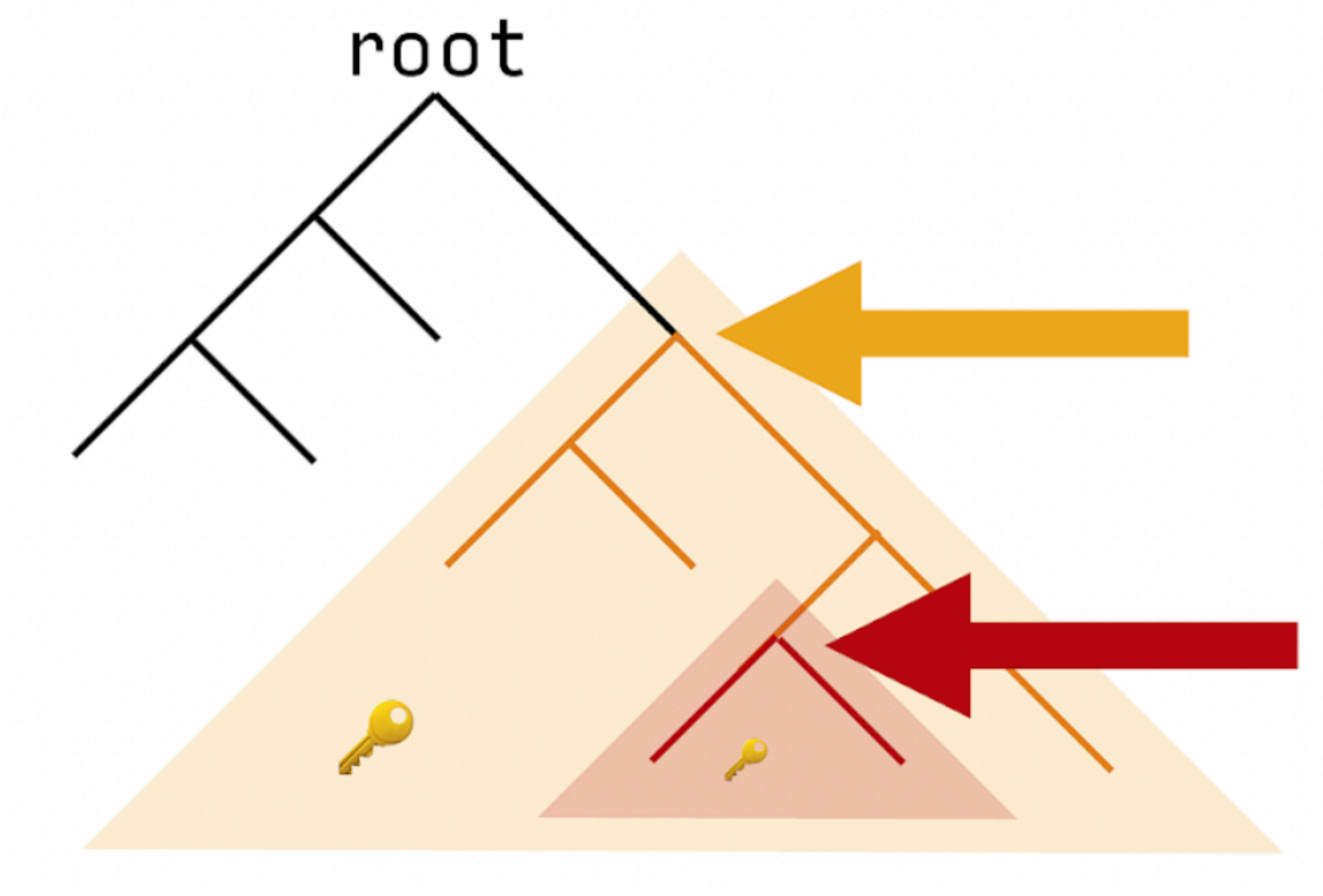
Direct Access Control

- Advantages
  - Proactive
  - Works offline
  - Attenuation
  - Easy to understand rules
  - User control (GDPR, CCPA)
  - Interoperable

- Challenges
  - Proactive
  - Revocation
  - Give up (more) access stats

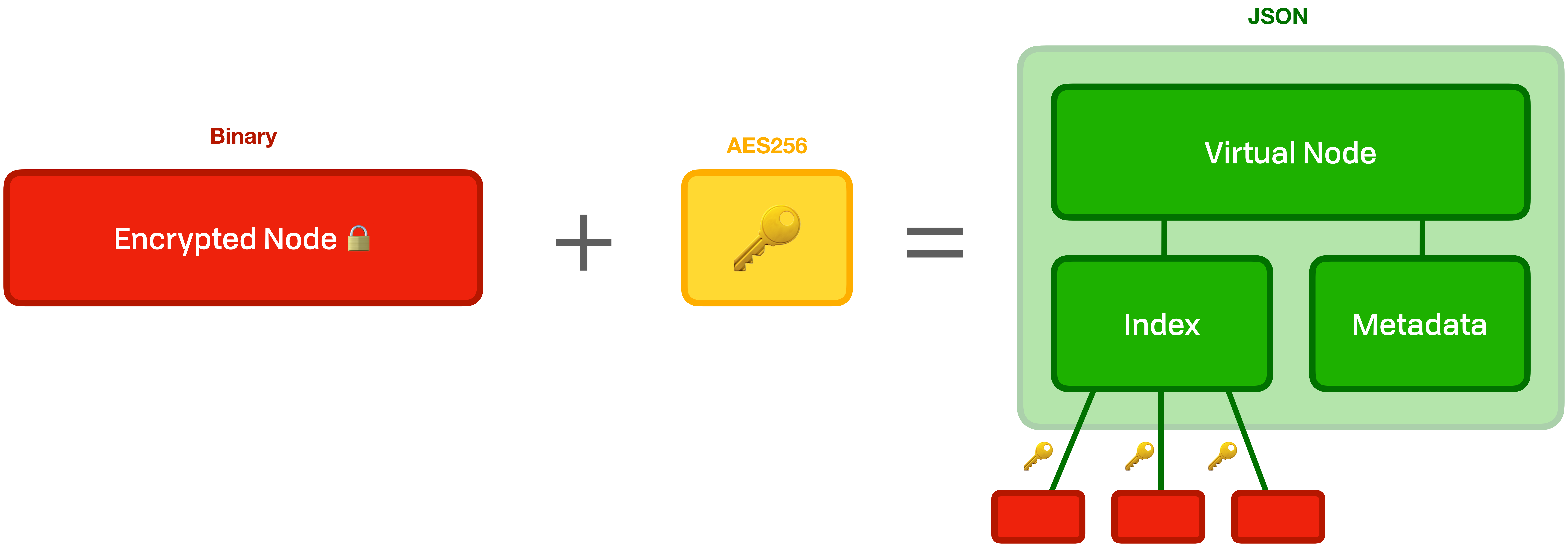
Fixing the Leaky Pipes 🚿

Hierarchal Read Access



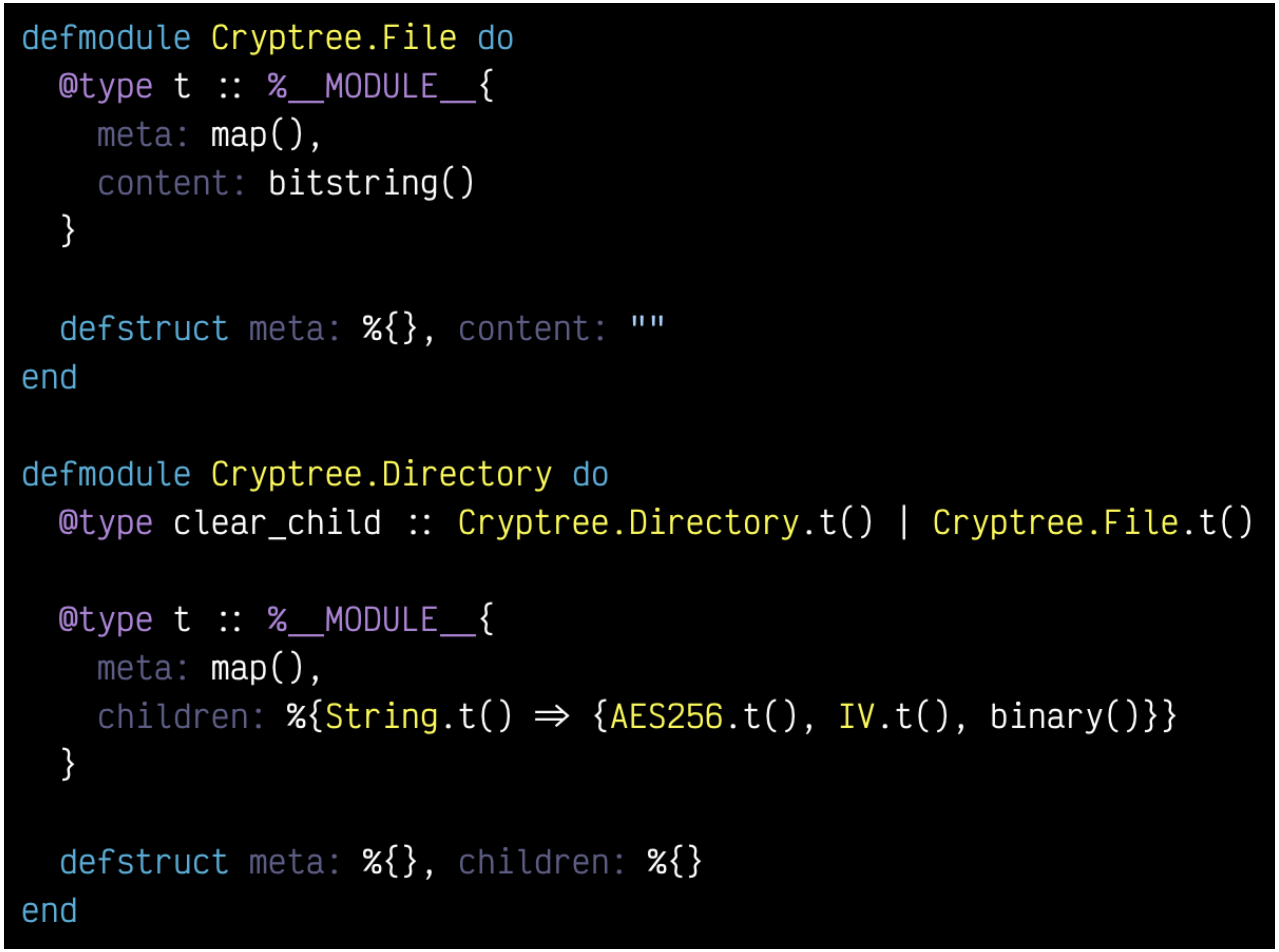
Fixing the Leaky Pipes 🚿

Cryptree 🎄



Fixing the Leaky Pipes 🚿

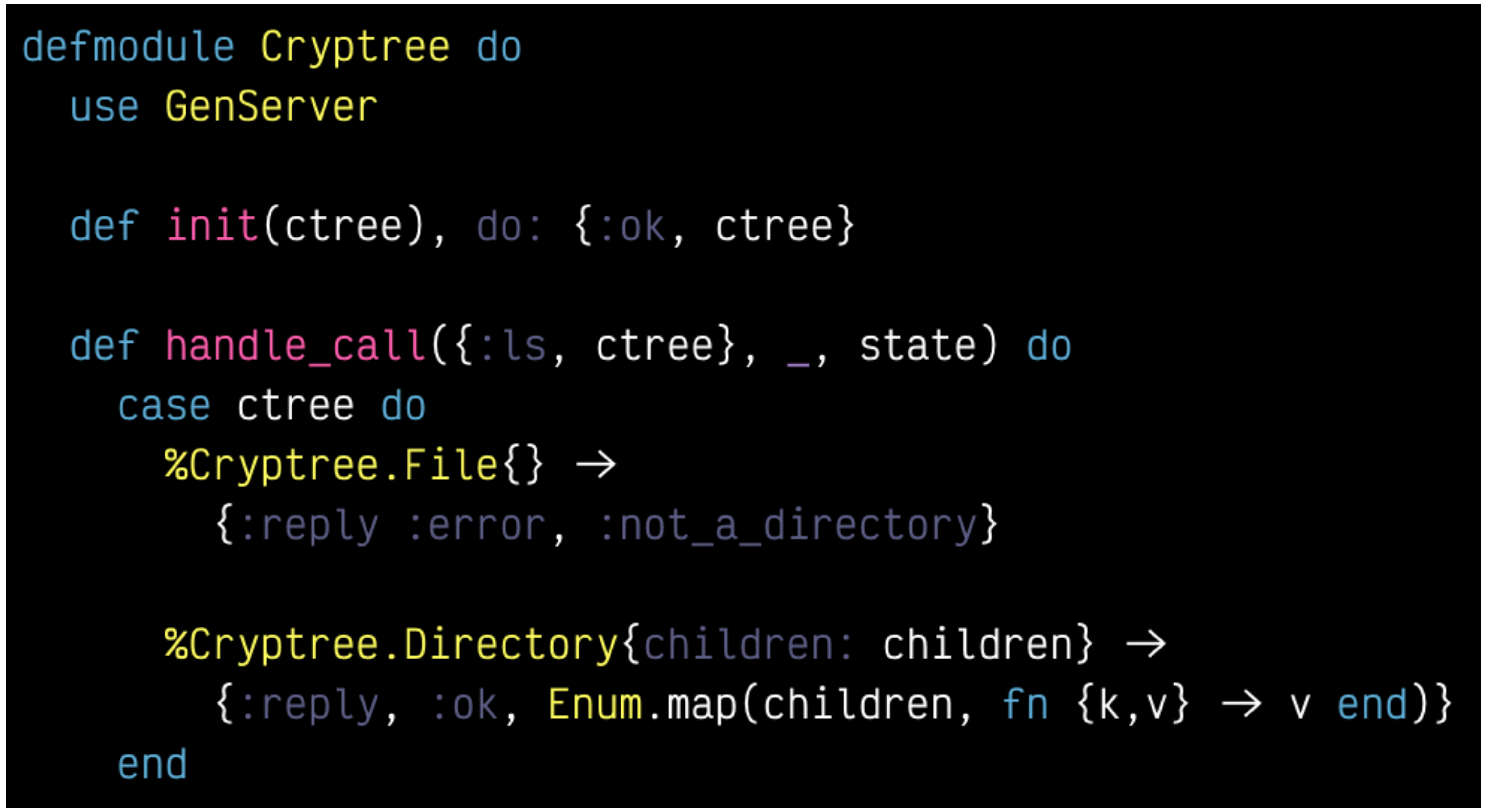
Cryptree Sketch ✍️



```elixir
defmodule Cryptree.File do
  @type t :: %__MODULE__{
    meta: map(),
    content: bitstring()
  }

  defstruct meta: %{}, content: ""
end

defmodule Cryptree.Directory do
  @type clear_child :: Cryptree.Directory.t() | Cryptree.File.t()

  @type t :: %__MODULE__{
    meta: map(),
    children: %{String.t() => {AES256.t(), IV.t(), binary()}}
  }

  defstruct meta: %{}, children: %{}
end
```

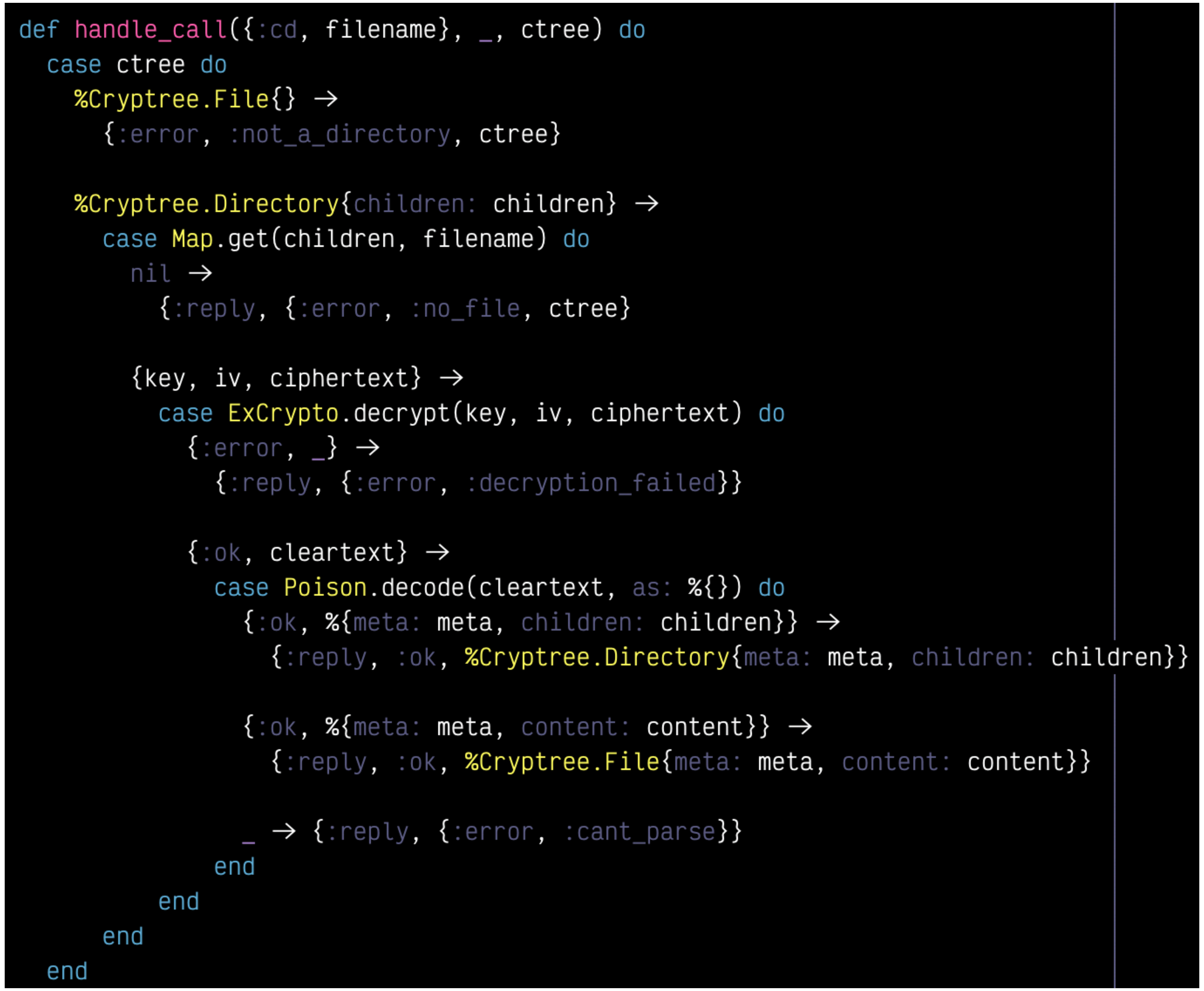
Fixing the Leaky Pipes 🚿

Cryptree Sketch ✍️



```elixir
defmodule Cryptree do
  use GenServer

  def init(ctree), do: {:ok, ctree}

  def handle_call({:ls, ctree}, _, state) do
    case ctree do
      %Cryptree.File{} ->
        {:reply :error, :not_a_directory}

      %Cryptree.Directory{children: children} ->
        {:reply, :ok, Enum.map(children, fn {k,v} -> v end)}
    end
```

Local stateful, remote stateless



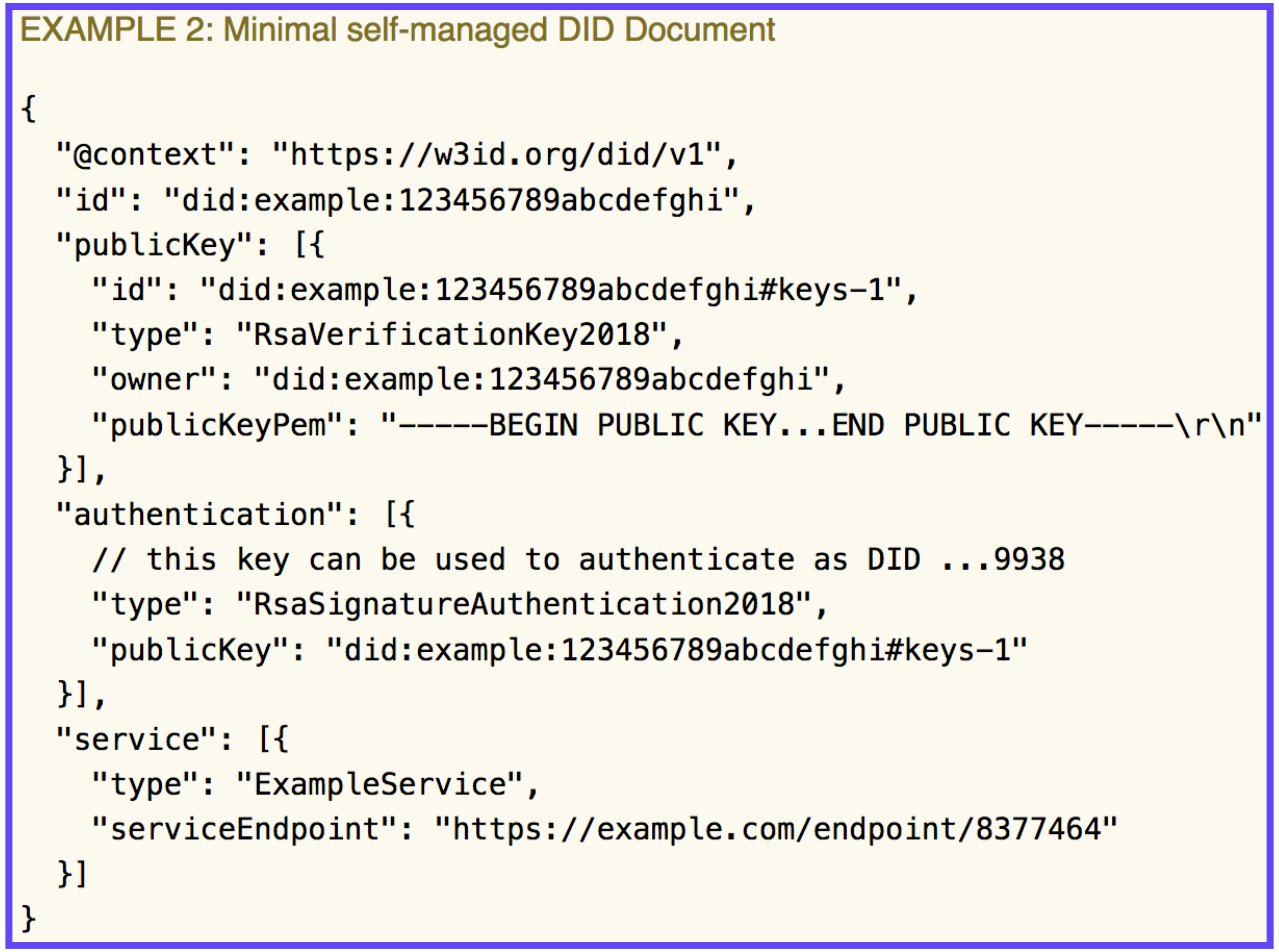
```elixir
def handle_call({:cd, filename}, _, ctree) do
  case ctree do
    %Cryptree.File{} ->
      {:error, :not_a_directory, ctree}

    %Cryptree.Directory{children: children} ->
      case Map.get(children, filename) do
        nil ->
          {:reply, {:error, :no_file, ctree}

        {key, iv, ciphertext} ->
          case ExCrypto.decrypt(key, iv, ciphertext) do
            {:error, _} ->
              {:reply, {:error, :decryption_failed}}

            {:ok, cleartext} ->
              case Poison.decode(cleartext, as: %{}) do
                {:ok, %{meta: meta, children: children}} ->
                  {:reply, :ok, %Cryptree.Directory{meta: meta, children: children}}

                {:ok, %{meta: meta, content: content}} ->
                  {:reply, :ok, %Cryptree.File{meta: meta, content: content}}

                _ -> {:reply, {:error, :cant_parse}}
              end
          end
      end
  end
```

How to Do Offline & Distributed Auth

Universal Auth & ID

Universal Auth & ID 🗝️

Universal IDs



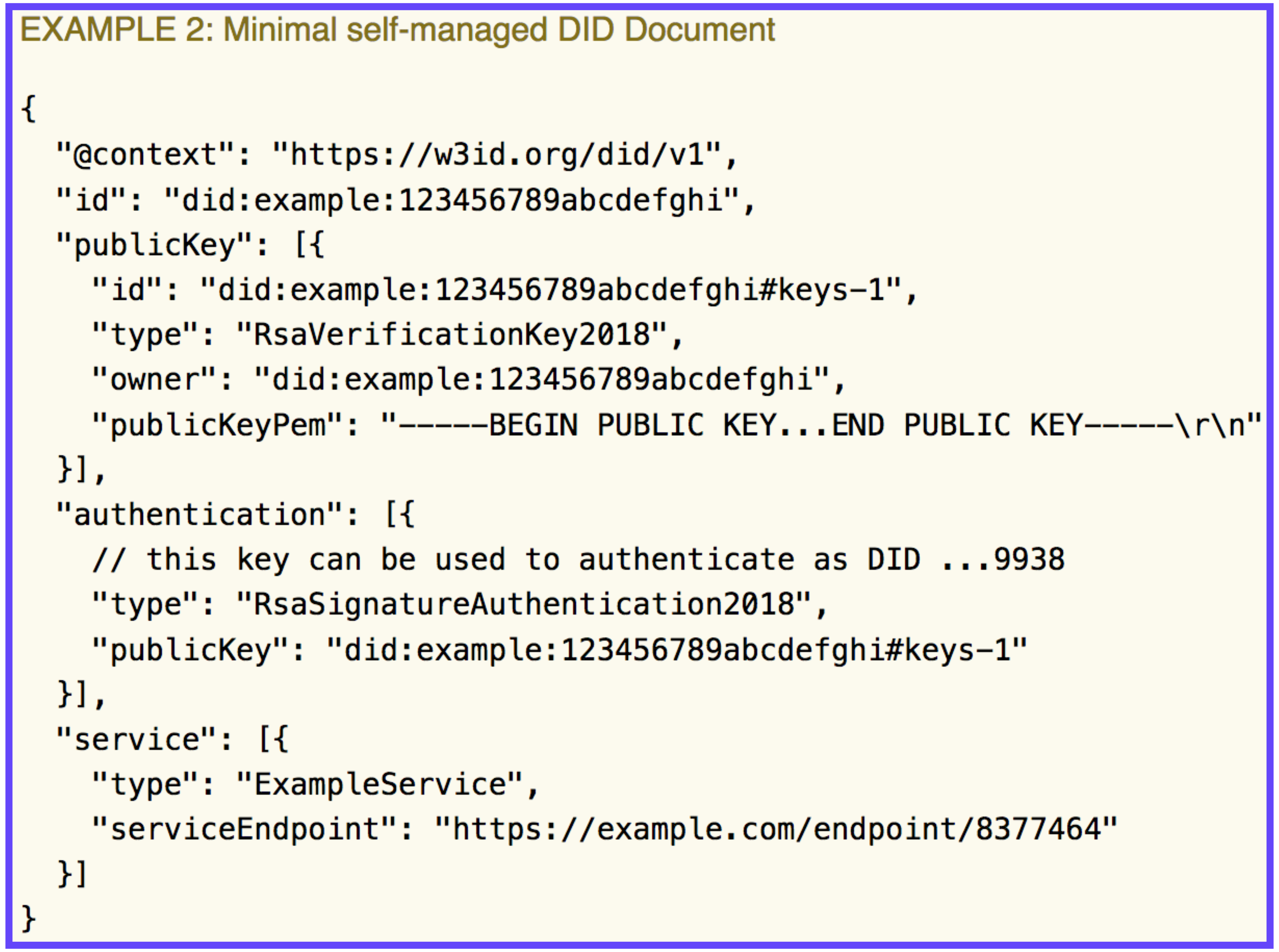
EXAMPLE 2: Minimal self-managed DID Document

```
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:example:123456789abcdefghi",
  "publicKey": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "RsaVerificationKey2018",
    "owner": "did:example:123456789abcdefghi",
    "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n"
  }],
  "authentication": [{
    // this key can be used to authenticate as DID ...9938
    "type": "RsaSignatureAuthentication2018",
    "publicKey": "did:example:123456789abcdefghi#keys-1"
  }],
  "service": [{
    "type": "ExampleService",
    "serviceEndpoint": "https://example.com/endpoint/8377464"
  }]
}
```

Universal Auth & ID 🗝️

Universal IDs

- W3C, DIF, Microsoft



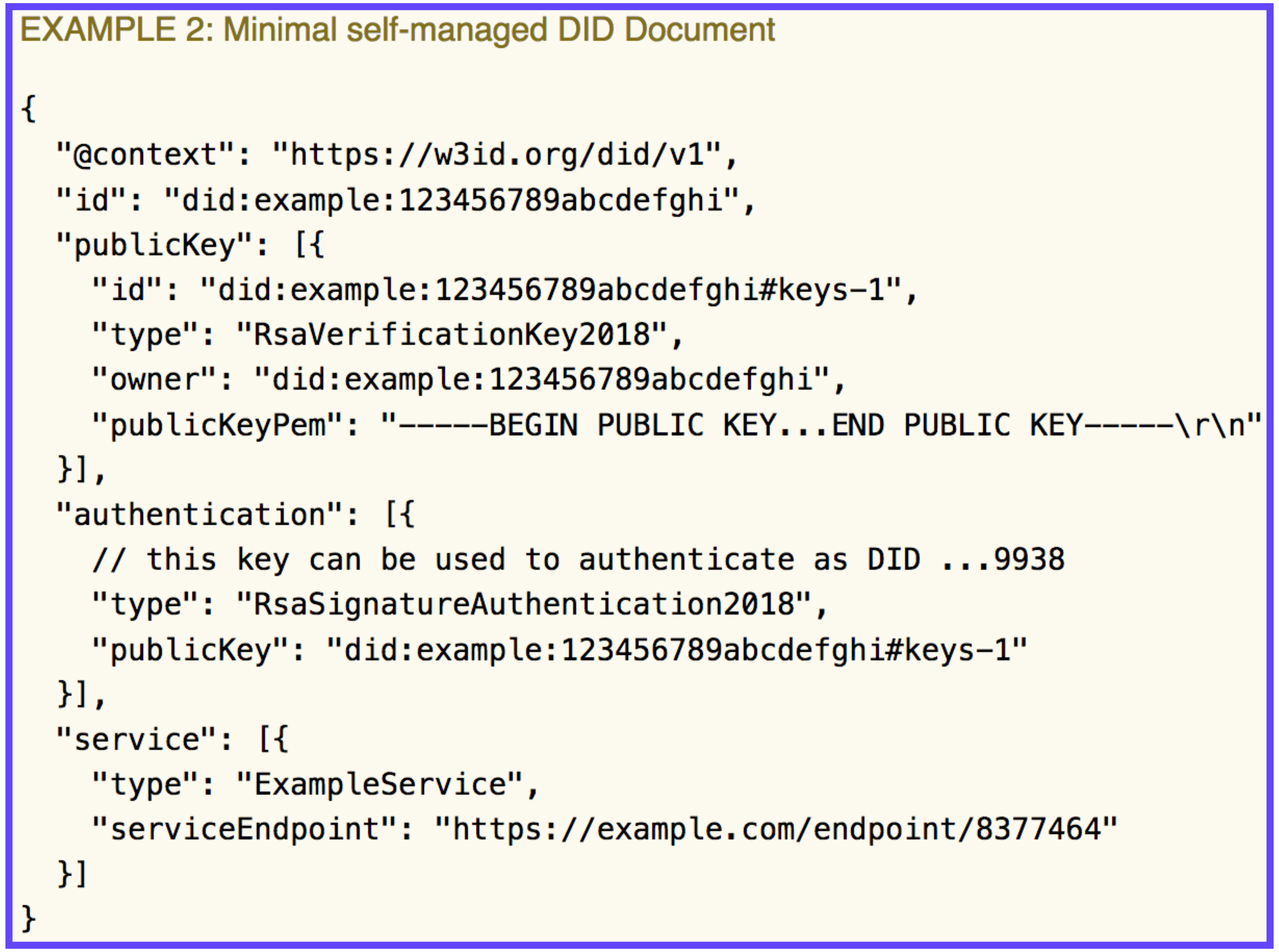
EXAMPLE 2: Minimal self-managed DID Document

```
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:example:123456789abcdefghi",
  "publicKey": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "RsaVerificationKey2018",
    "owner": "did:example:123456789abcdefghi",
    "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n"
  }],
  "authentication": [{
    // this key can be used to authenticate as DID ...9938
    "type": "RsaSignatureAuthentication2018",
    "publicKey": "did:example:123456789abcdefghi#keys-1"
  }],
  "service": [{
    "type": "ExampleService",
    "serviceEndpoint": "https://example.com/endpoint/8377464"
  }]
}
```

Universal Auth & ID 🗝️

Universal IDs

- W3C, DIF, Microsoft
- Based on public-key cryptography



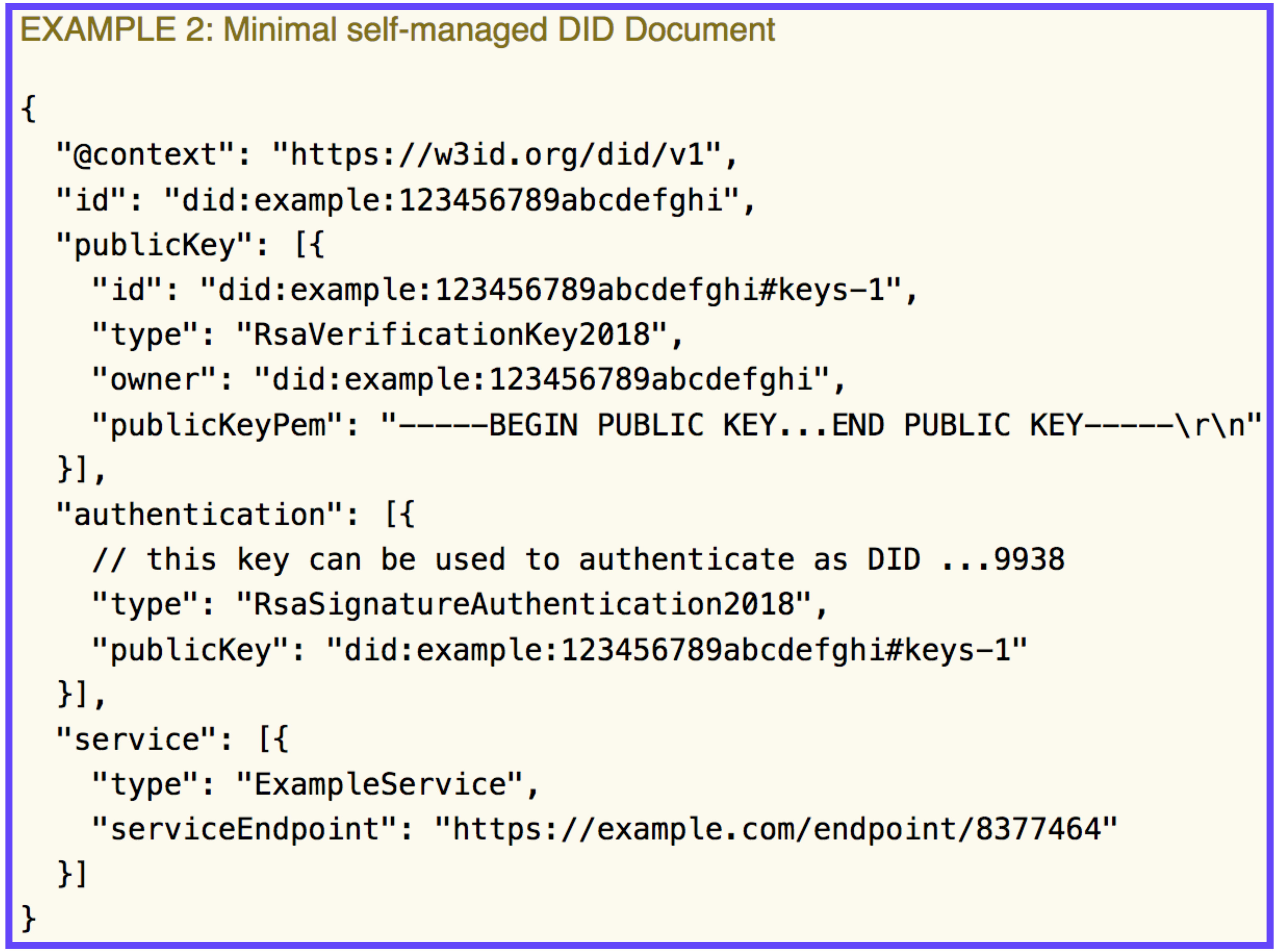
EXAMPLE 2: Minimal self-managed DID Document

```
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:example:123456789abcdefghi",
  "publicKey": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "RsaVerificationKey2018",
    "owner": "did:example:123456789abcdefghi",
    "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n"
  }],
  "authentication": [{
    // this key can be used to authenticate as DID ...9938
    "type": "RsaSignatureAuthentication2018",
    "publicKey": "did:example:123456789abcdefghi#keys-1"
  }],
  "service": [{
    "type": "ExampleService",
    "serviceEndpoint": "https://example.com/endpoint/8377464"
  }]
}
```

Universal Auth & ID 🗝️

Universal IDs

- W3C, DIF, Microsoft
- Based on public-key cryptography
- Truly “universal” user IDs



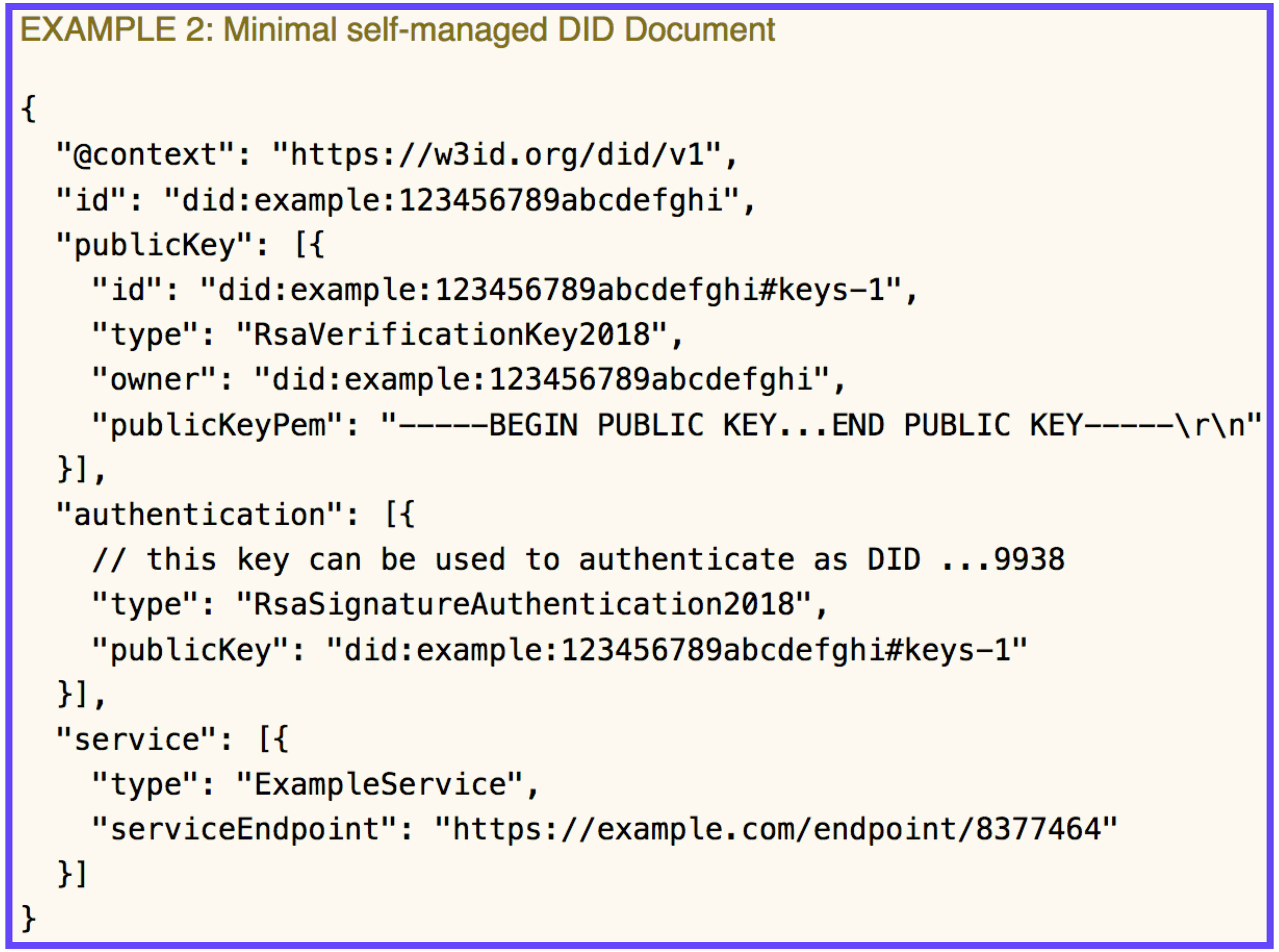
EXAMPLE 2: Minimal self-managed DID Document

```
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:example:123456789abcdefghi",
  "publicKey": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "RsaVerificationKey2018",
    "owner": "did:example:123456789abcdefghi",
    "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n"
  }],
  "authentication": [{
    // this key can be used to authenticate as DID ...9938
    "type": "RsaSignatureAuthentication2018",
    "publicKey": "did:example:123456789abcdefghi#keys-1"
  }],
  "service": [{
    "type": "ExampleService",
    "serviceEndpoint": "https://example.com/endpoint/8377464"
  }]
}
```

Universal Auth & ID 🗝️

Universal IDs

- W3C, DIF, Microsoft
- Based on public-key cryptography
- Truly “universal” user IDs
- Agnostic about backing



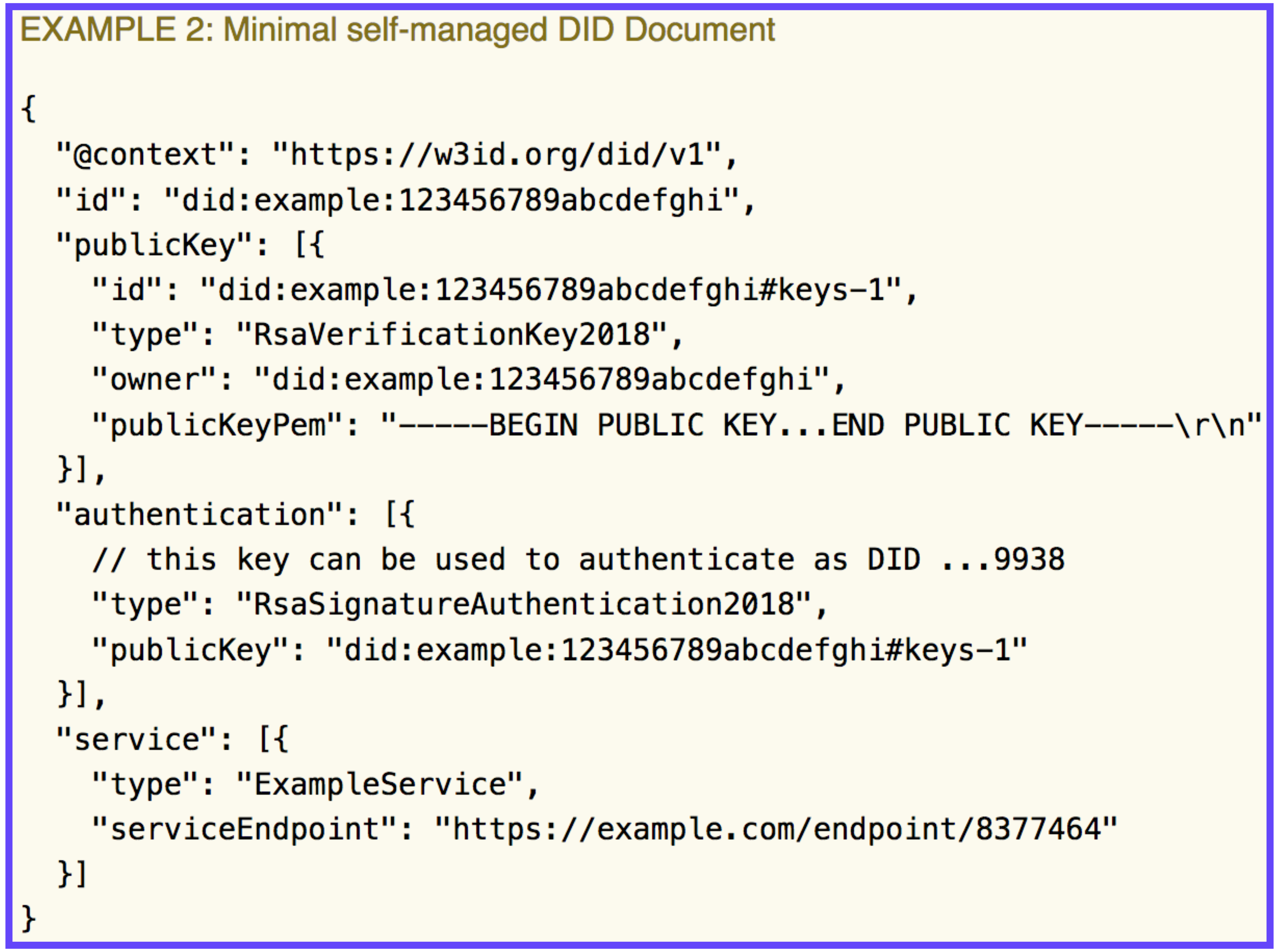
EXAMPLE 2: Minimal self-managed DID Document

```
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:example:123456789abcdefghi",
  "publicKey": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "RsaVerificationKey2018",
    "owner": "did:example:123456789abcdefghi",
    "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n"
  }],
  "authentication": [{
    // this key can be used to authenticate as DID ...9938
    "type": "RsaSignatureAuthentication2018",
    "publicKey": "did:example:123456789abcdefghi#keys-1"
  }],
  "service": [{
    "type": "ExampleService",
    "serviceEndpoint": "https://example.com/endpoint/8377464"
  }]
}
```

Universal Auth & ID 🗝️

Universal IDs

- W3C, DIF, Microsoft
- Based on public-key cryptography
- Truly “universal” user IDs
- Agnostic about backing
- For users, devices, and more

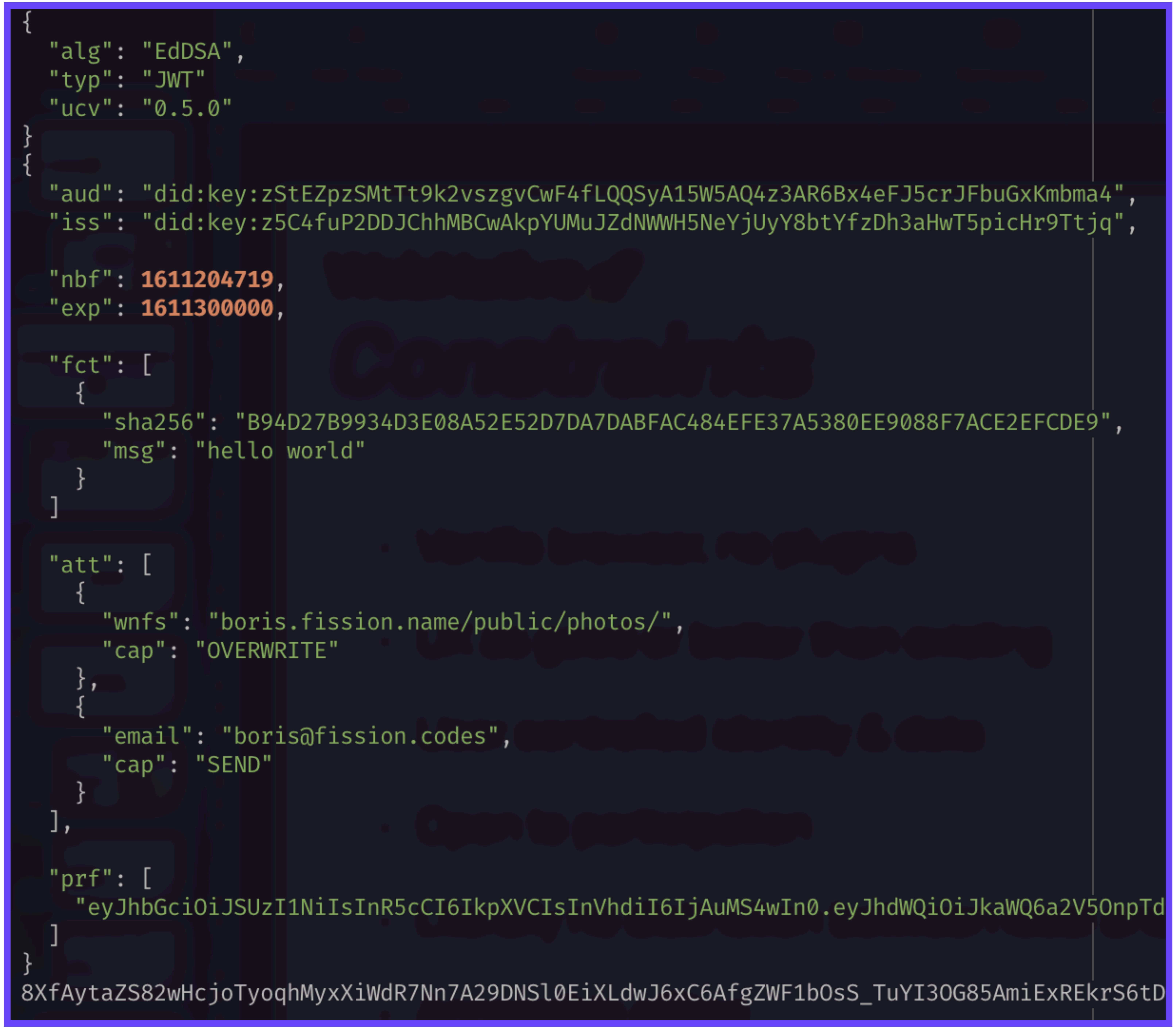


EXAMPLE 2: Minimal self-managed DID Document

```
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:example:123456789abcdefghi",
  "publicKey": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "RsaVerificationKey2018",
    "owner": "did:example:123456789abcdefghi",
    "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n"
  }],
  "authentication": [{
    // this key can be used to authenticate as DID ...9938
    "type": "RsaSignatureAuthentication2018",
    "publicKey": "did:example:123456789abcdefghi#keys-1"
  }],
  "service": [{
    "type": "ExampleService",
    "serviceEndpoint": "https://example.com/endpoint/8377464"
  }]
}
```

Universal Auth & ID 🗝️

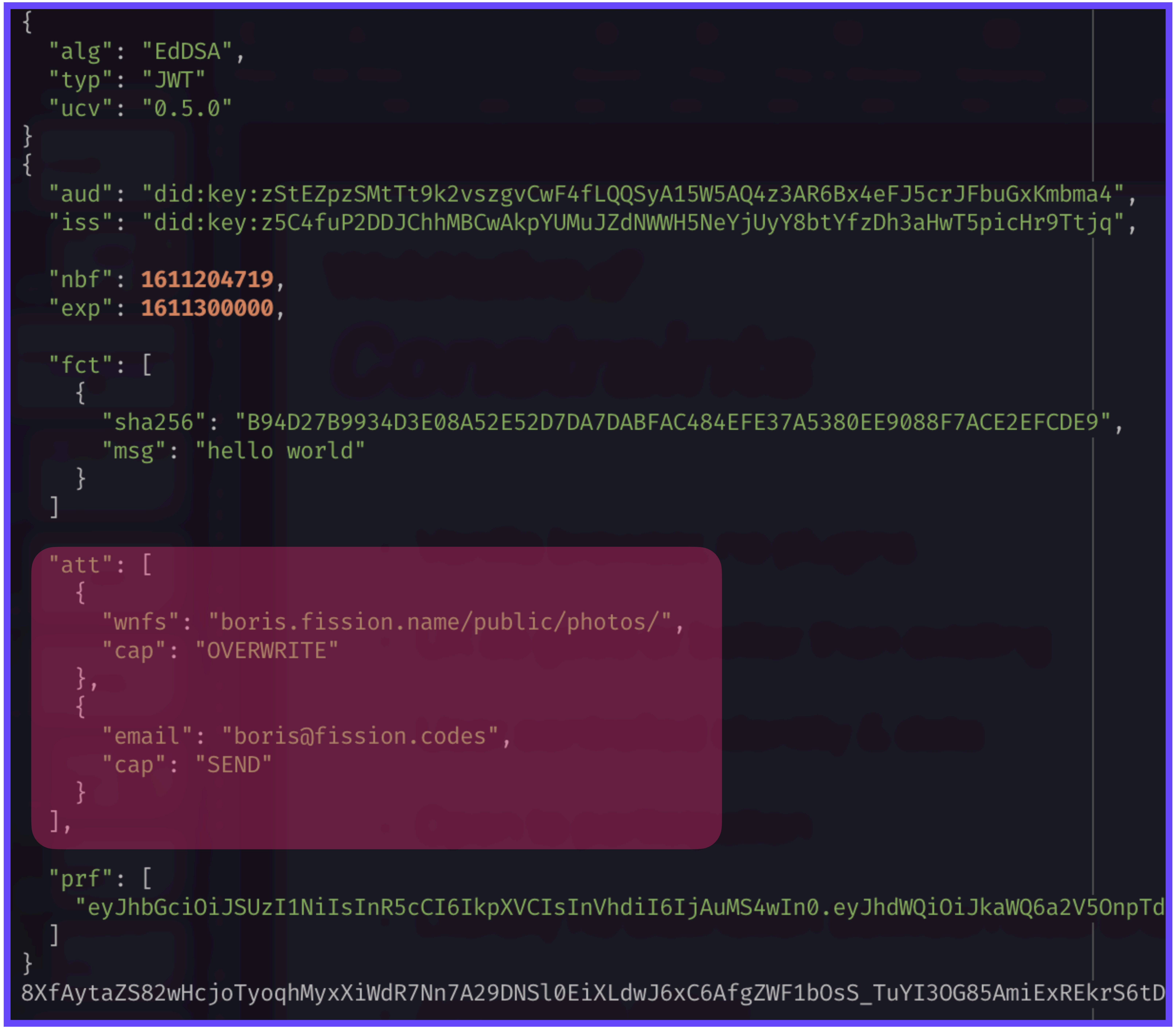
JWT Encoded



```
{
  "alg": "EdDSA",
  "typ": "JWT"
  "ucv": "0.5.0"
}
{
  "aud": "did:key:zStEZpzSMtTt9k2vszgvCwF4fLQQSyA15W5AQ4z3AR6Bx4eFJ5crJFbuGxKmbma4",
  "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": [
    "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsInVhdiI6IjAuMS4wIn0.eyJhdWQiOiJkaWQ6a2V5OnpTd
  ]
}
8XfAytaZS82wHcjoTyoqhMyxXiWdR7Nn7A29DNSl0EiXLdwJ6xC6AfgZWF1bOsS_TuYI3OG85AmiExREkrS6tD
```

Universal Auth & ID 🗝️

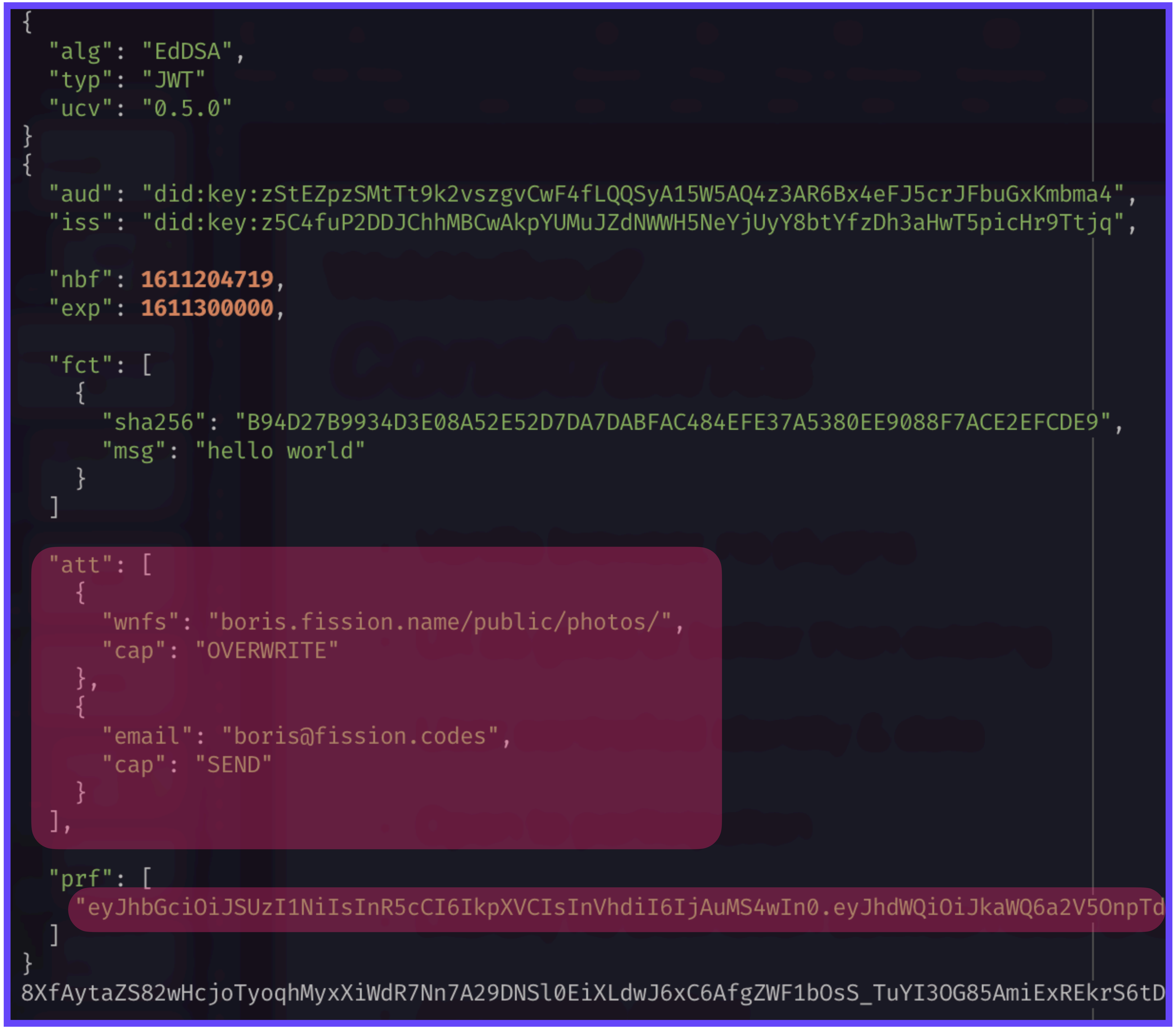
JWT Encoded



```
{
  "alg": "EdDSA",
  "typ": "JWT"
  "ucv": "0.5.0"
}
{
  "aud": "did:key:zStEZpzSMtTt9k2vszgvCwF4fLQQSyA15W5AQ4z3AR6Bx4eFJ5crJFbuGxKmbma4",
  "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": [
    "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsInVhdiI6IjAuMS4wIn0.eyJhdWQiOiJkaWQ6a2V5OnpTd
  ]
}
8XfAytaZS82wHcjoTyoqhMyxXiWdR7Nn7A29DNSl0EiXLdwJ6xC6AfgZWF1bOsS_TuYI3OG85AmiExREkrS6tD
```

Universal Auth & ID 🗝️

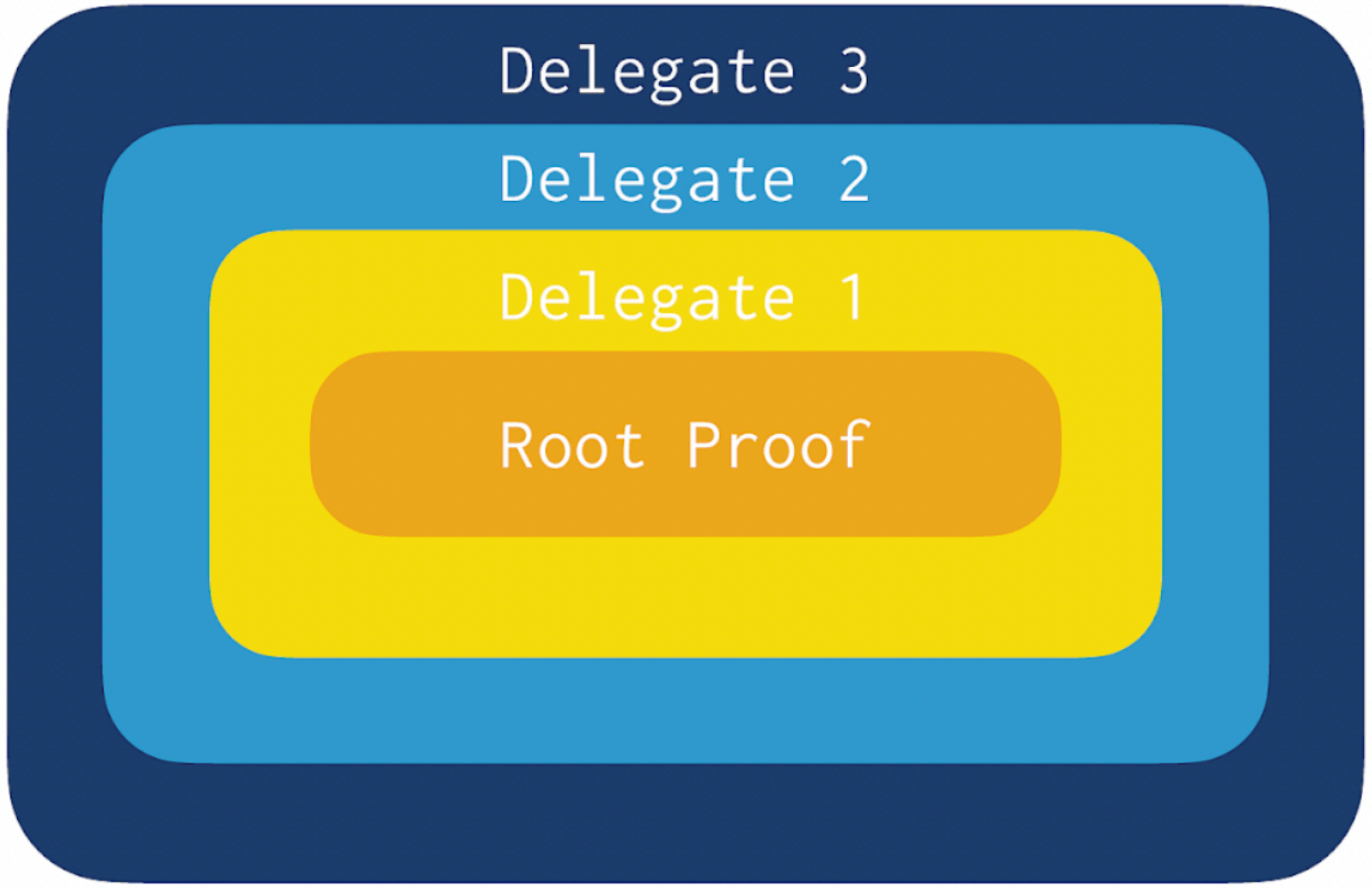
JWT Encoded



```
{
  "alg": "EdDSA",
  "typ": "JWT"
  "ucv": "0.5.0"
}
{
  "aud": "did:key:zStEZpzSMtTt9k2vszgvCwF4fLQQSyA15W5AQ4z3AR6Bx4eFJ5crJFbuGxKmbma4",
  "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": [
    "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsInVhdiI6IjAuMS4wIn0.eyJhdWQiOiJkaWQ6a2V5OnpTd
  ]
}
8XfAytaZS82wHcjoTyoqhMyxXiWdR7Nn7A29DNSl0EiXLdwJ6xC6AfgZWF1bOsS_TuYI3OG85AmiExREkrS6tD
```

Universal Auth & ID 🗝️

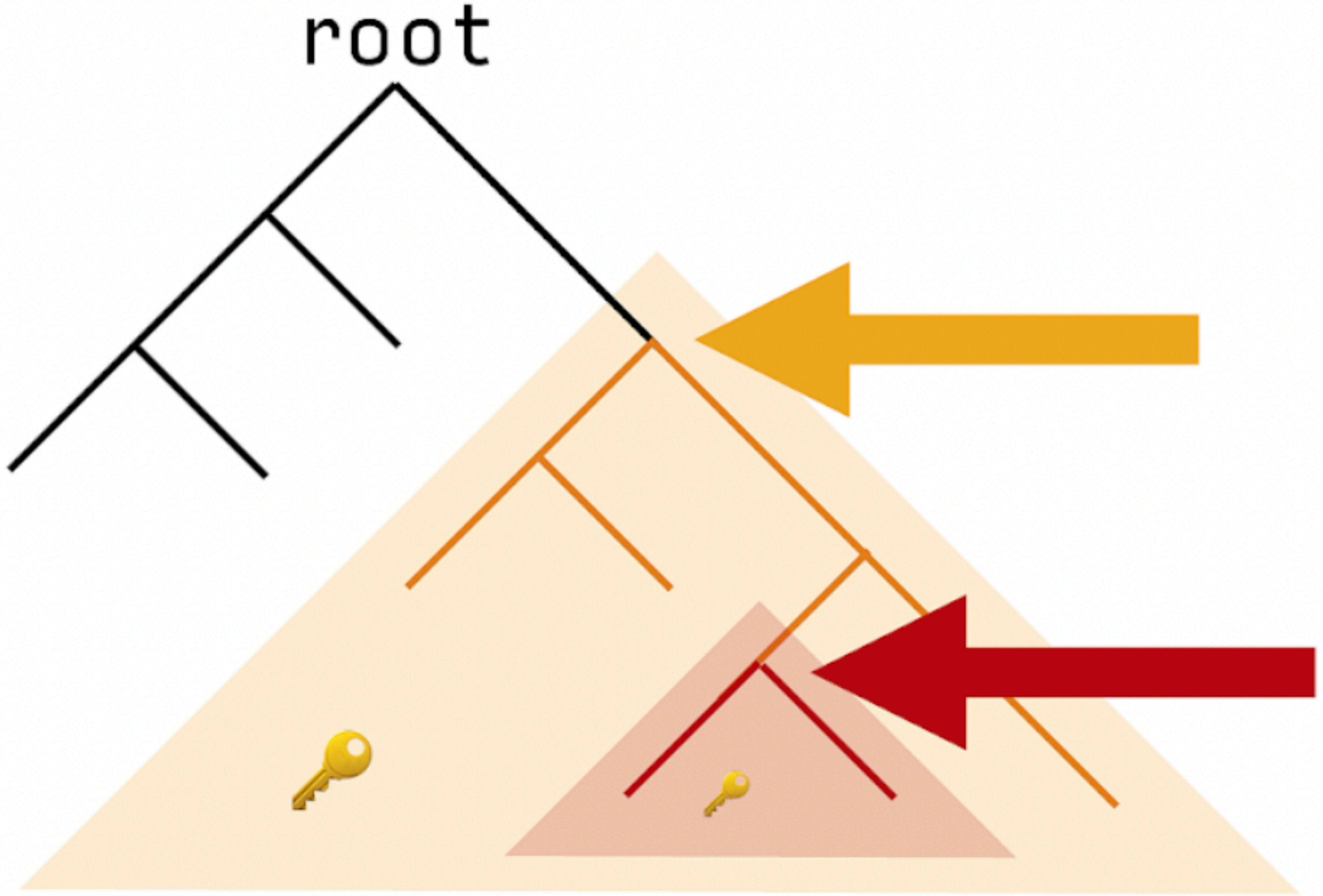
Auth Chaining


Delegate 3

Delegate 2

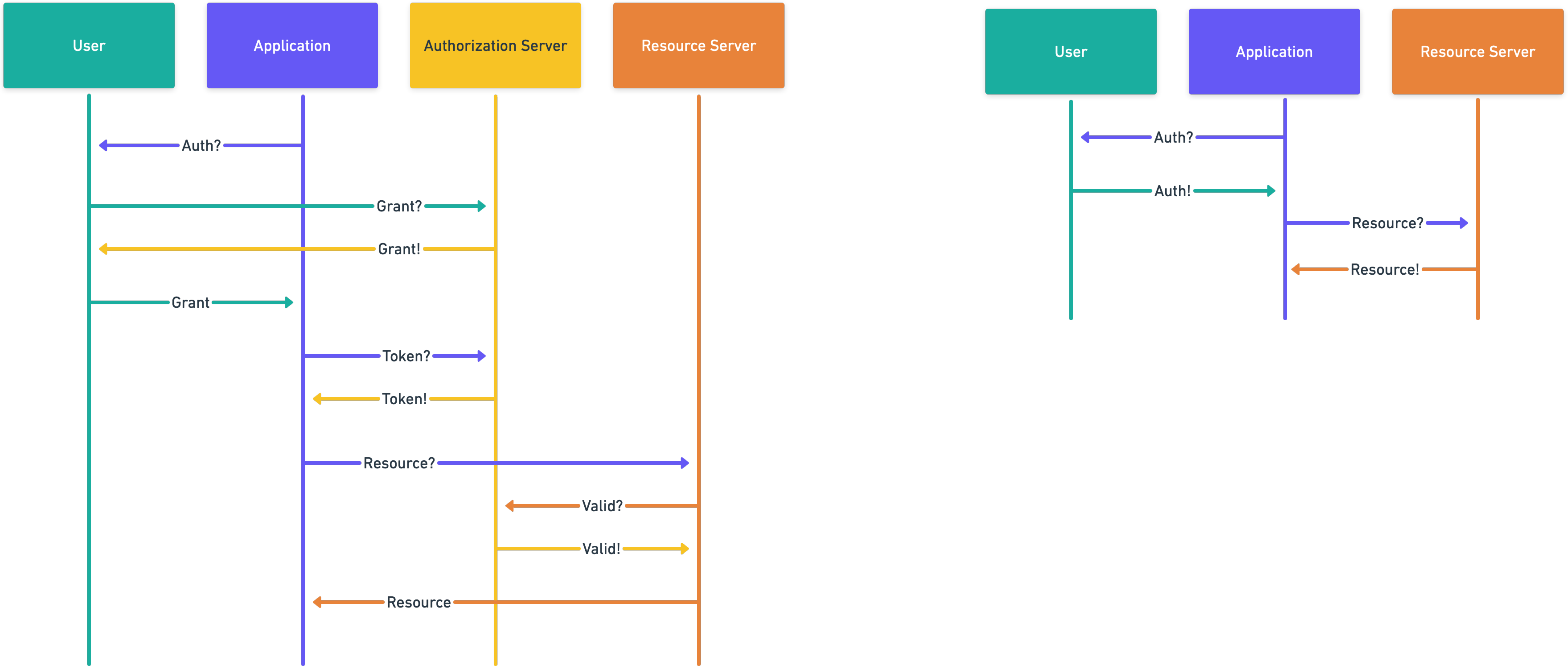
Delegate 1

Root Proof



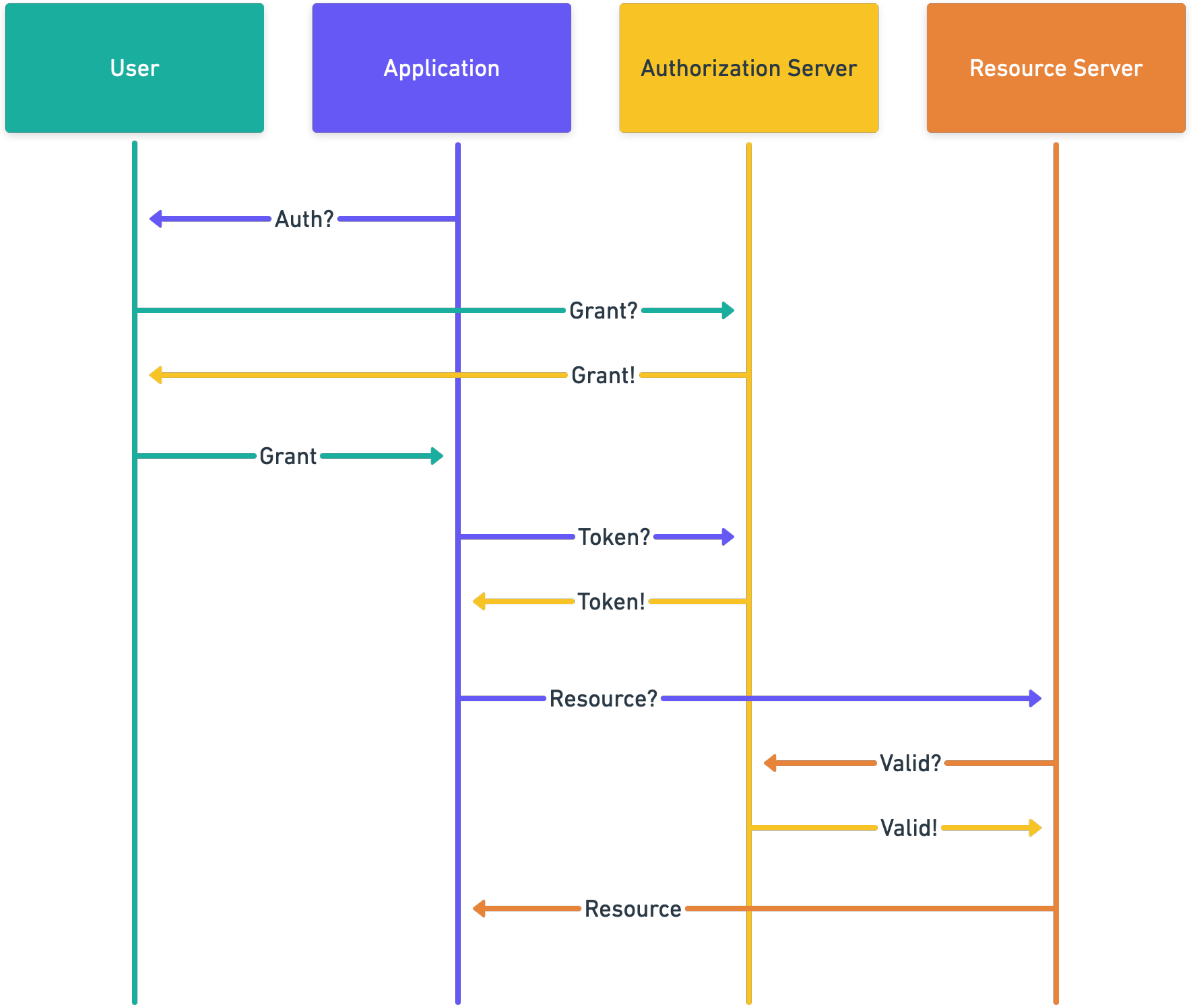
Universal Auth & ID 🗝️

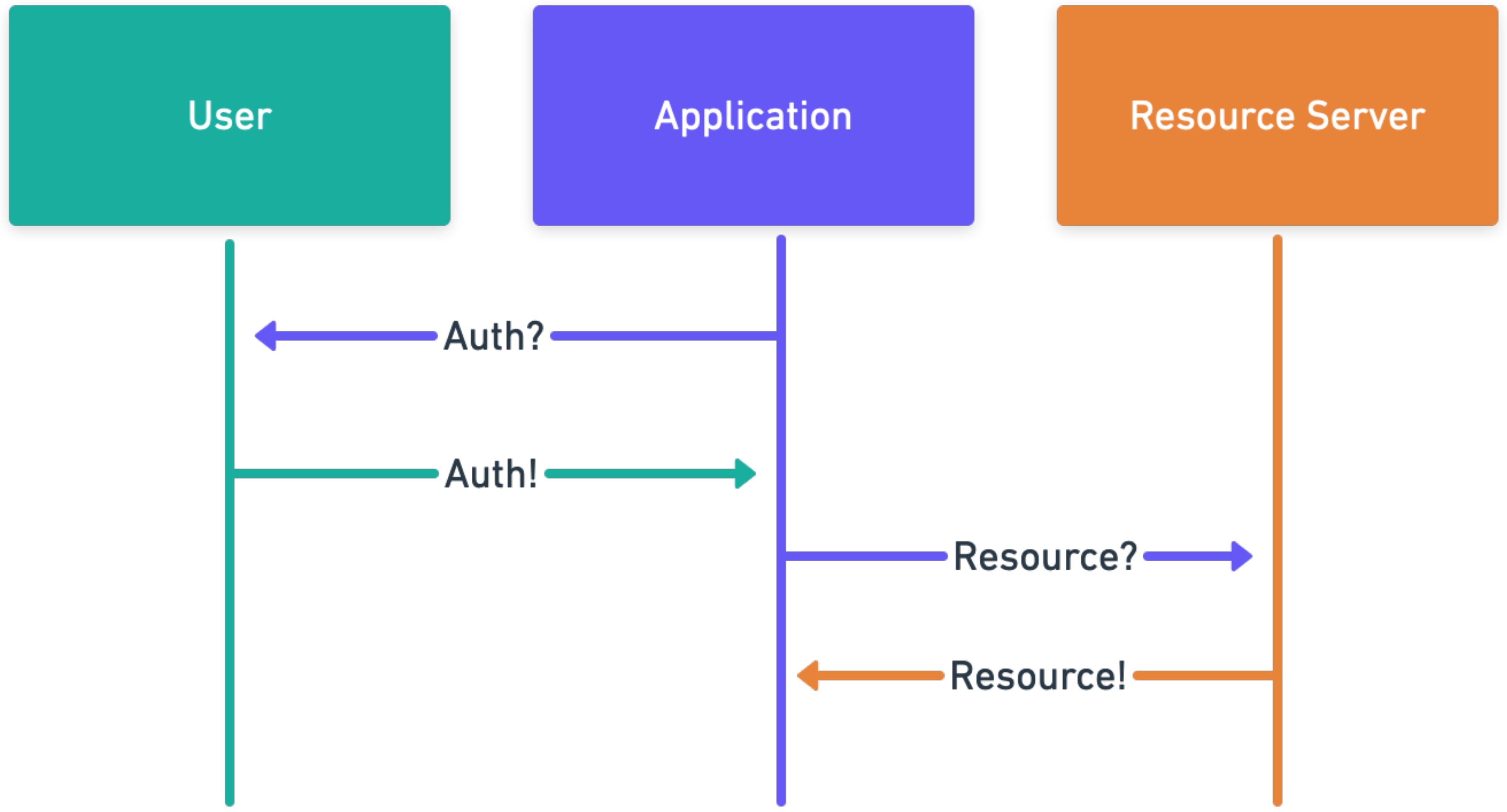
OAuth vs UCAN Sequence



Universal Auth & ID 🗝️

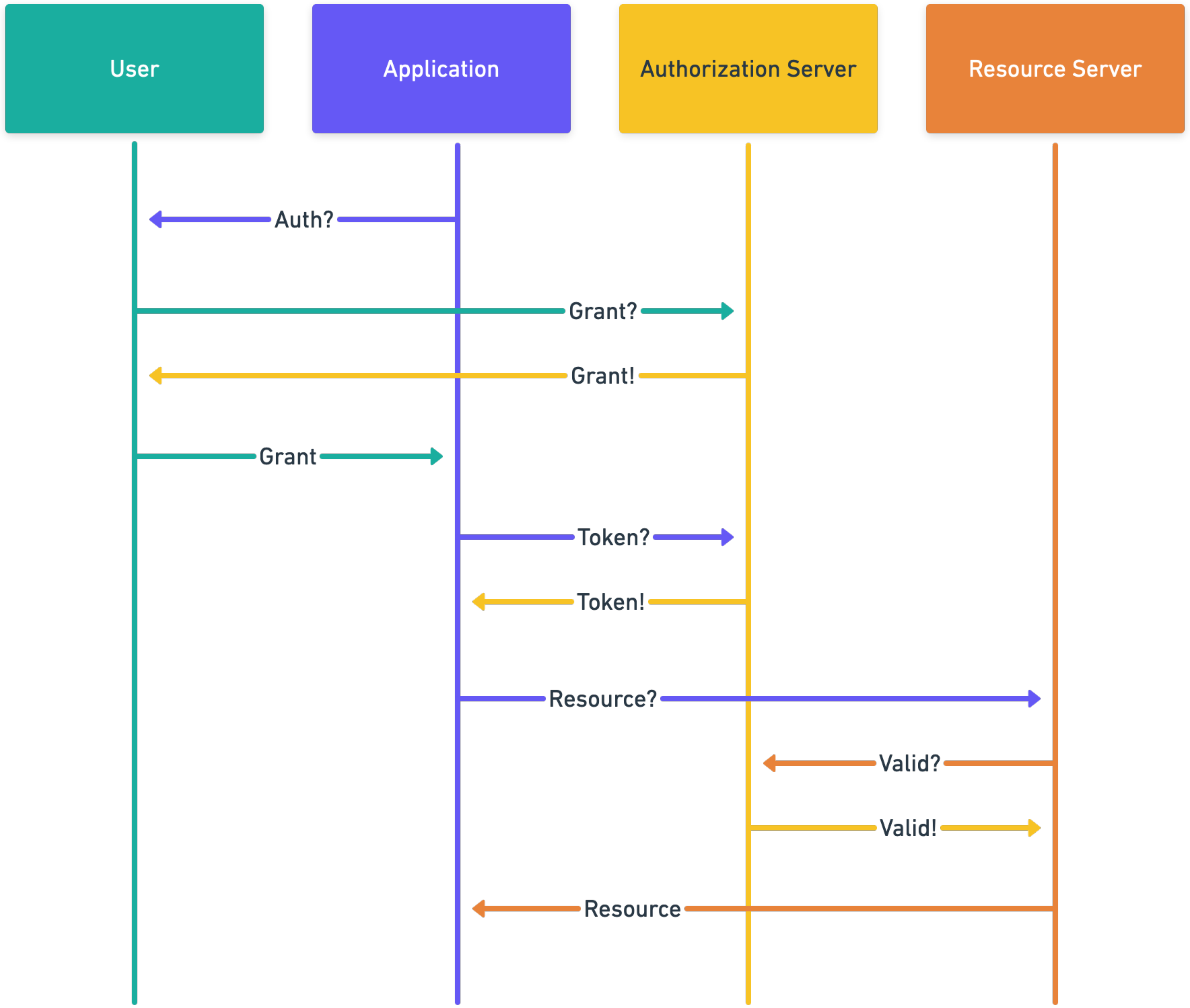
OAuth vs UCAN Sequence

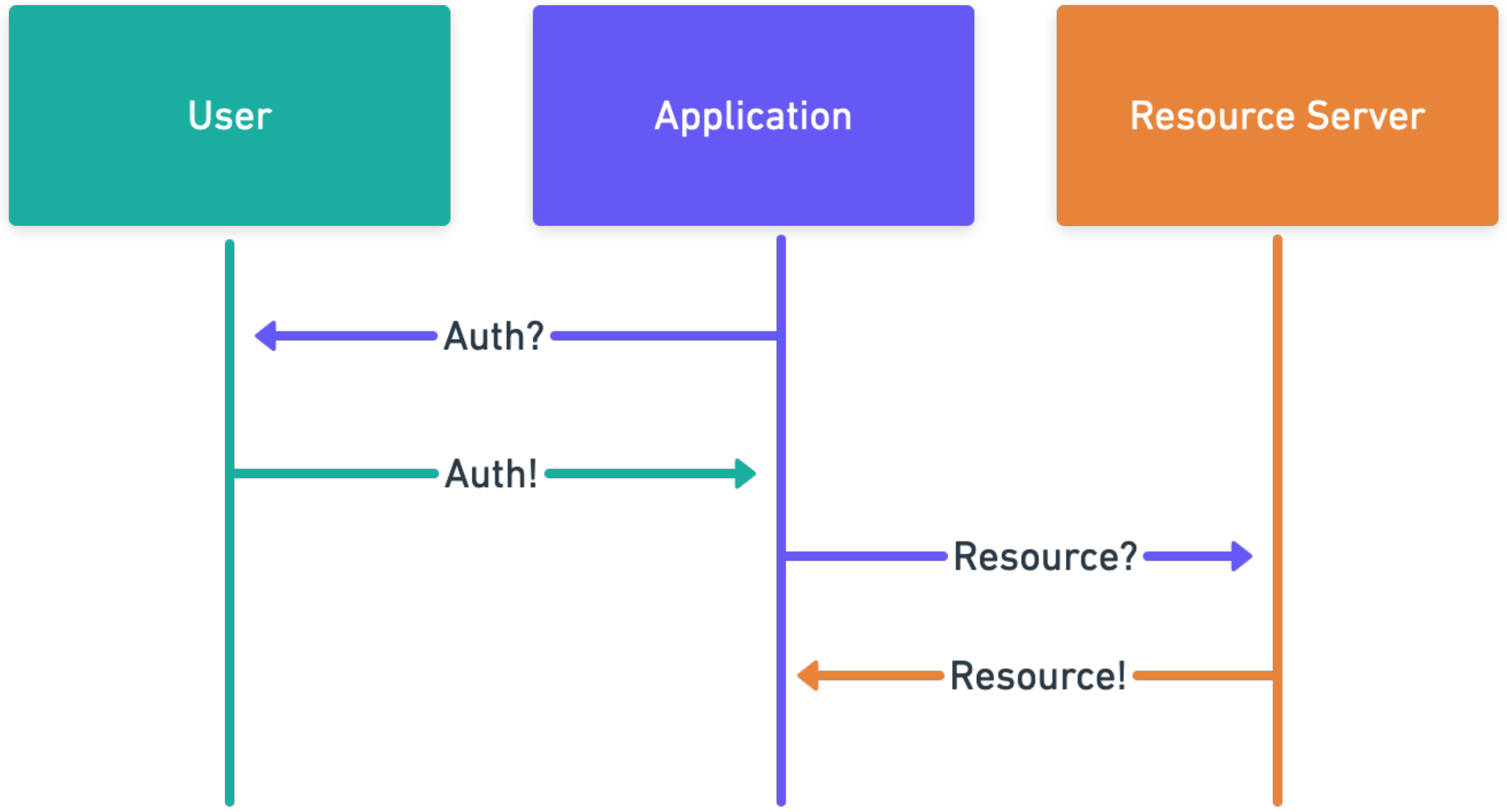




Universal Auth & ID 🗝️

OAuth vs UCAN Sequence

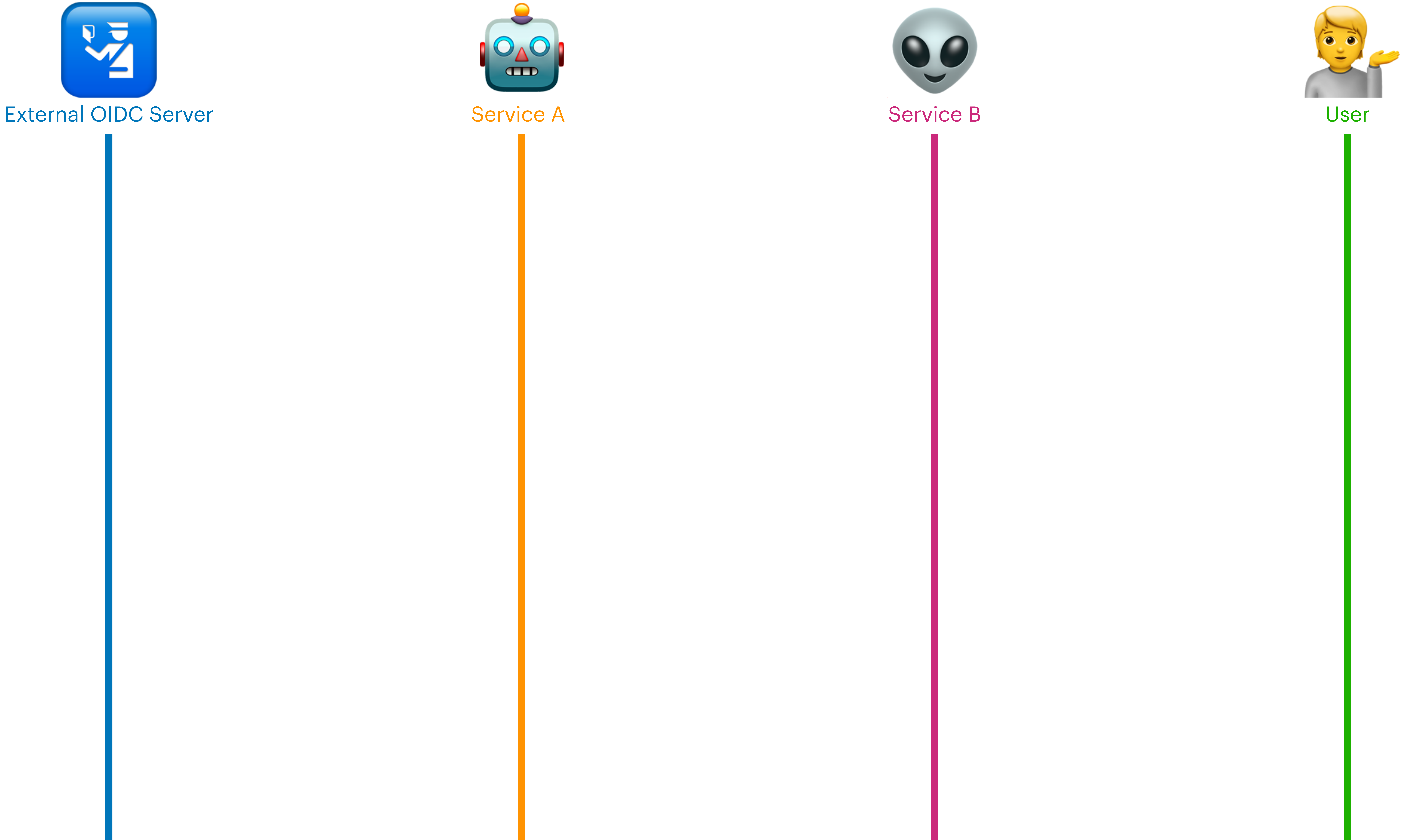




(Verifiable & user originated)

Universal Auth & ID 🗝️

Universal Auth & ID 🗝️



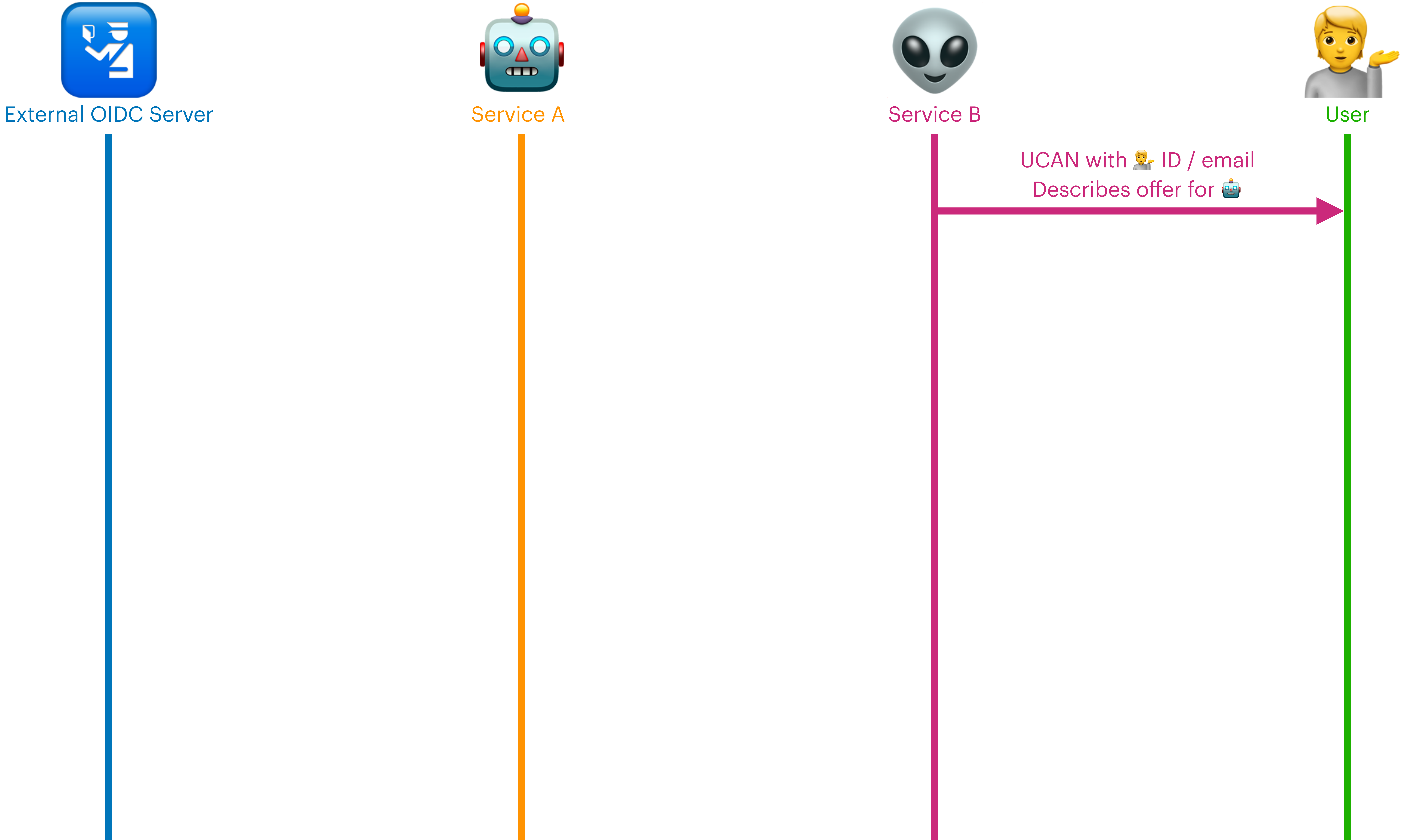
External OIDC Server

Service A

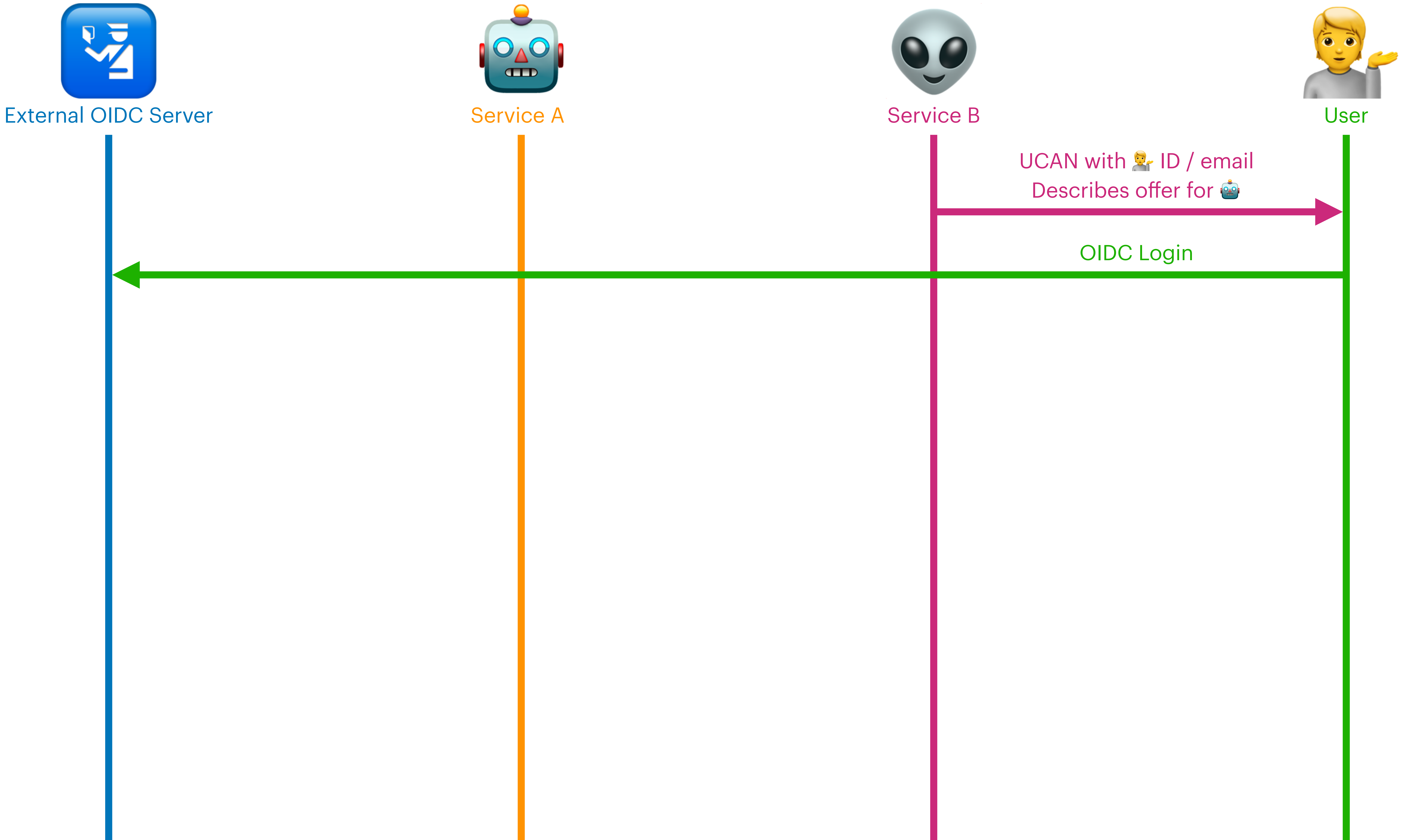
Service B

User

Universal Auth & ID 🗝️



Universal Auth & ID 🗝️



External OIDC Server

Service A

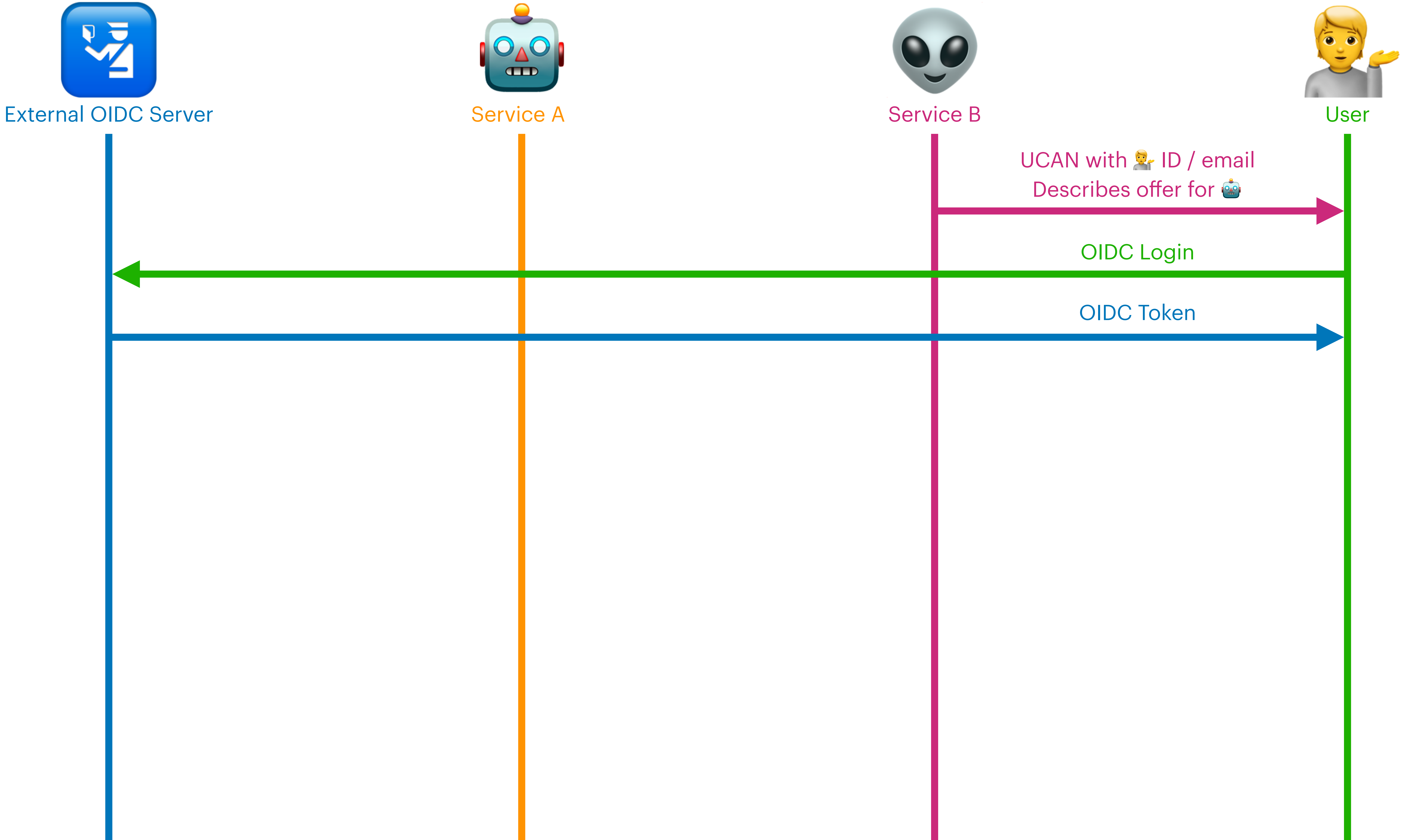
Service B

User

UCAN with 💁 ID / email
Describes offer for 🤖

OIDC Login

Universal Auth & ID 🗝️



External OIDC Server

Service A

Service B

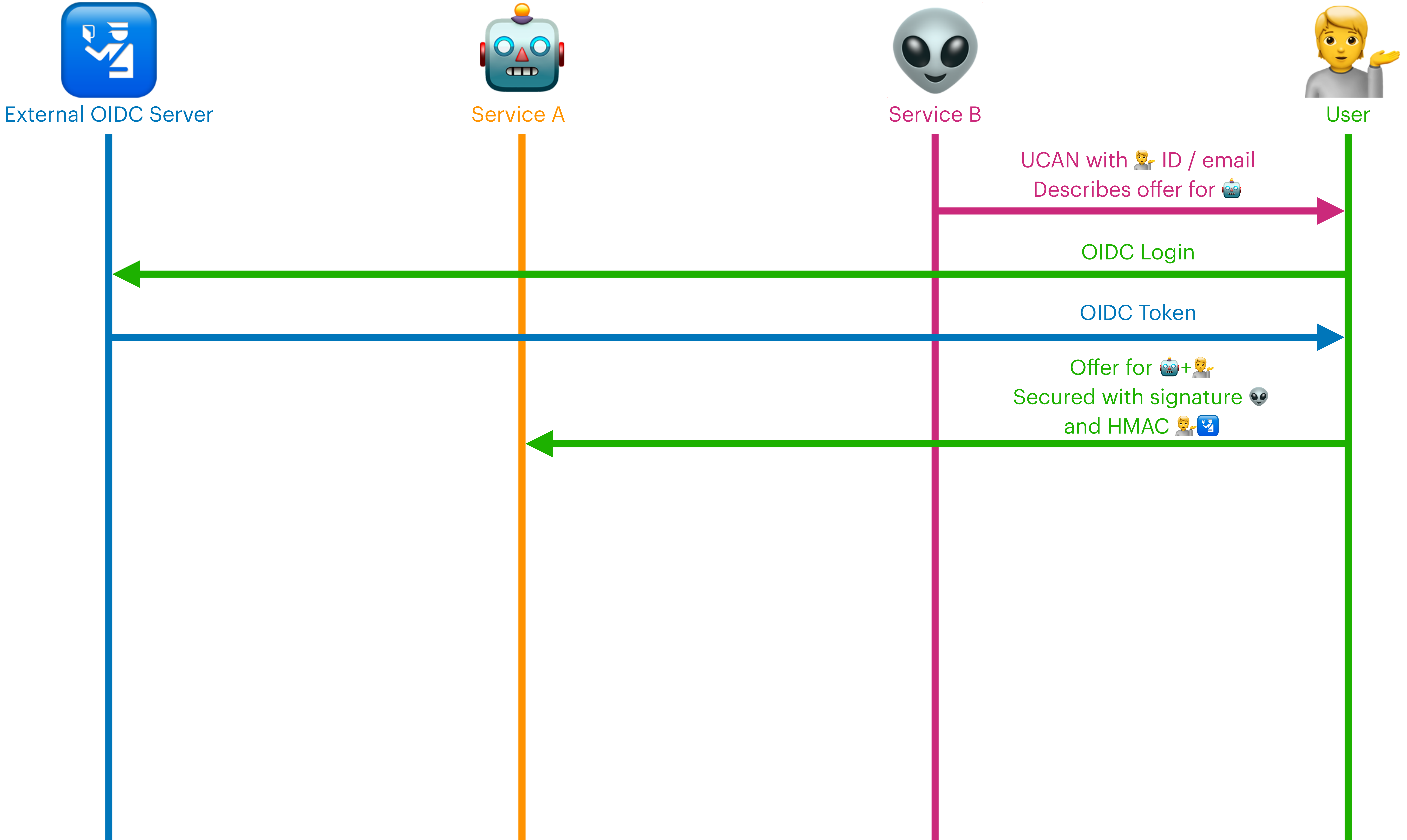
User

UCAN with 💁 ID / email
Describes offer for 🤖

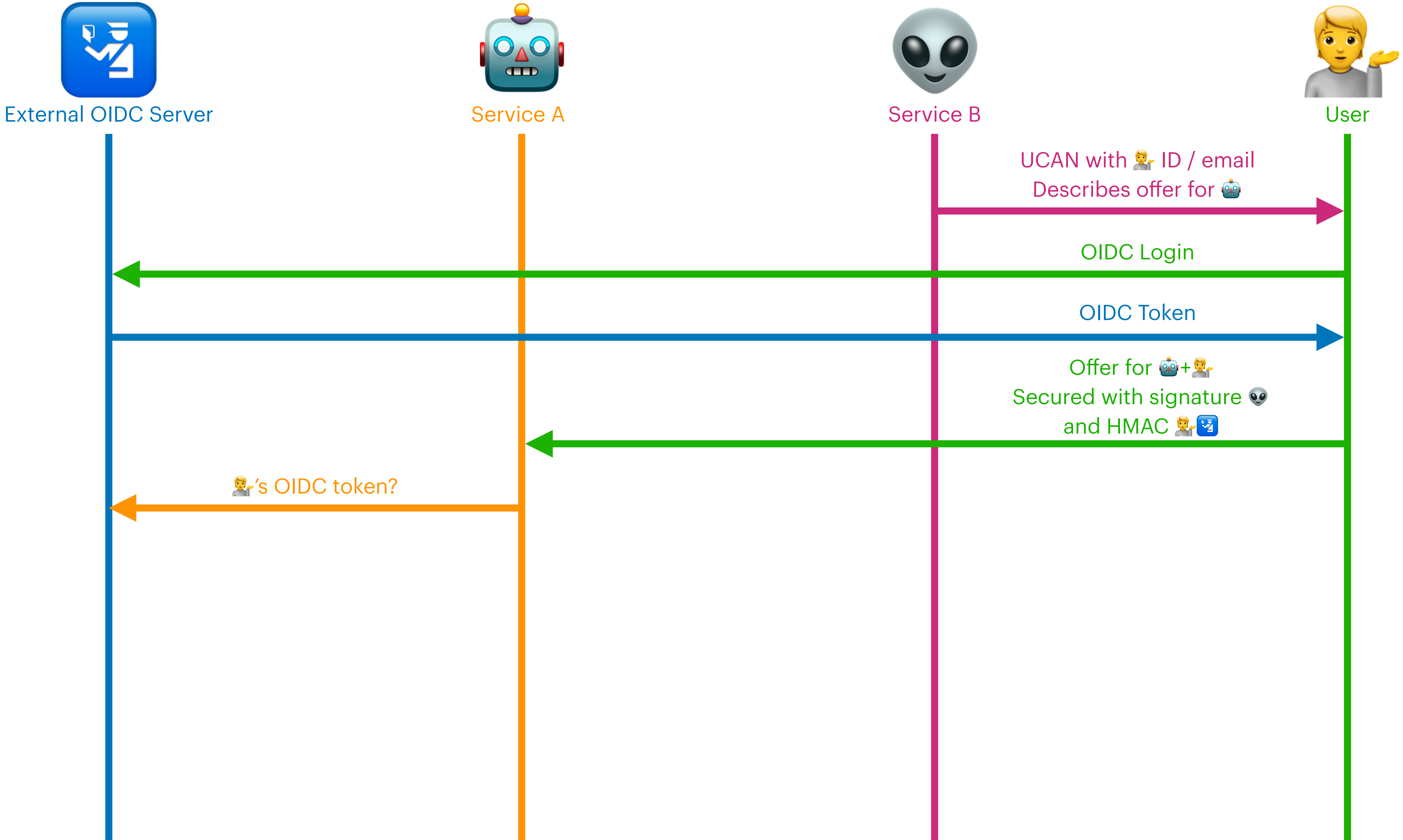
OIDC Login

OIDC Token

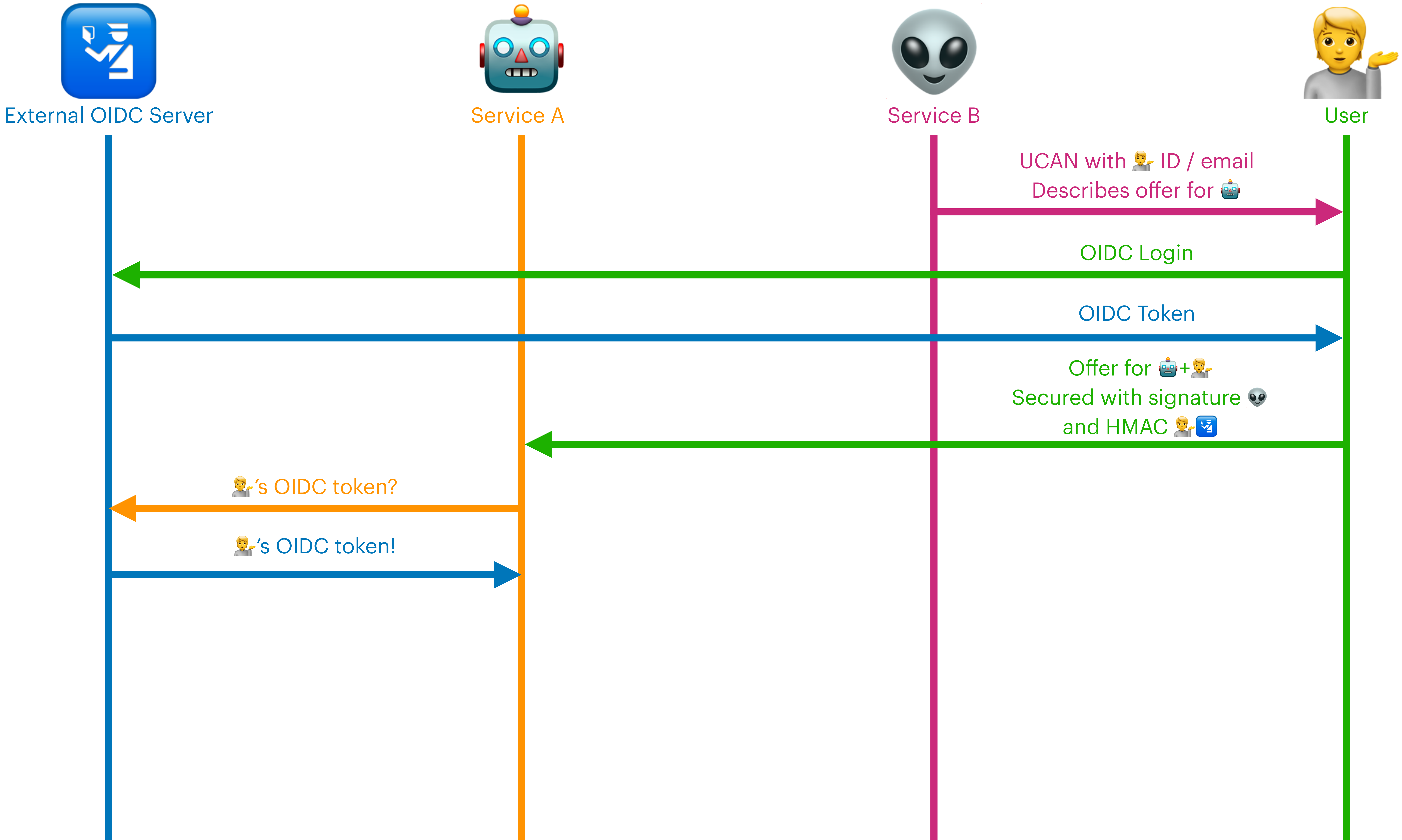
Universal Auth & ID 🗝️



Universal Auth & ID 🗝️



Universal Auth & ID 🗝️



🛂 External OIDC Server

🤖 Service A

👽 Service B

💁 User

UCAN with 💁 ID / email
Describes offer for 🤖

OIDC Login

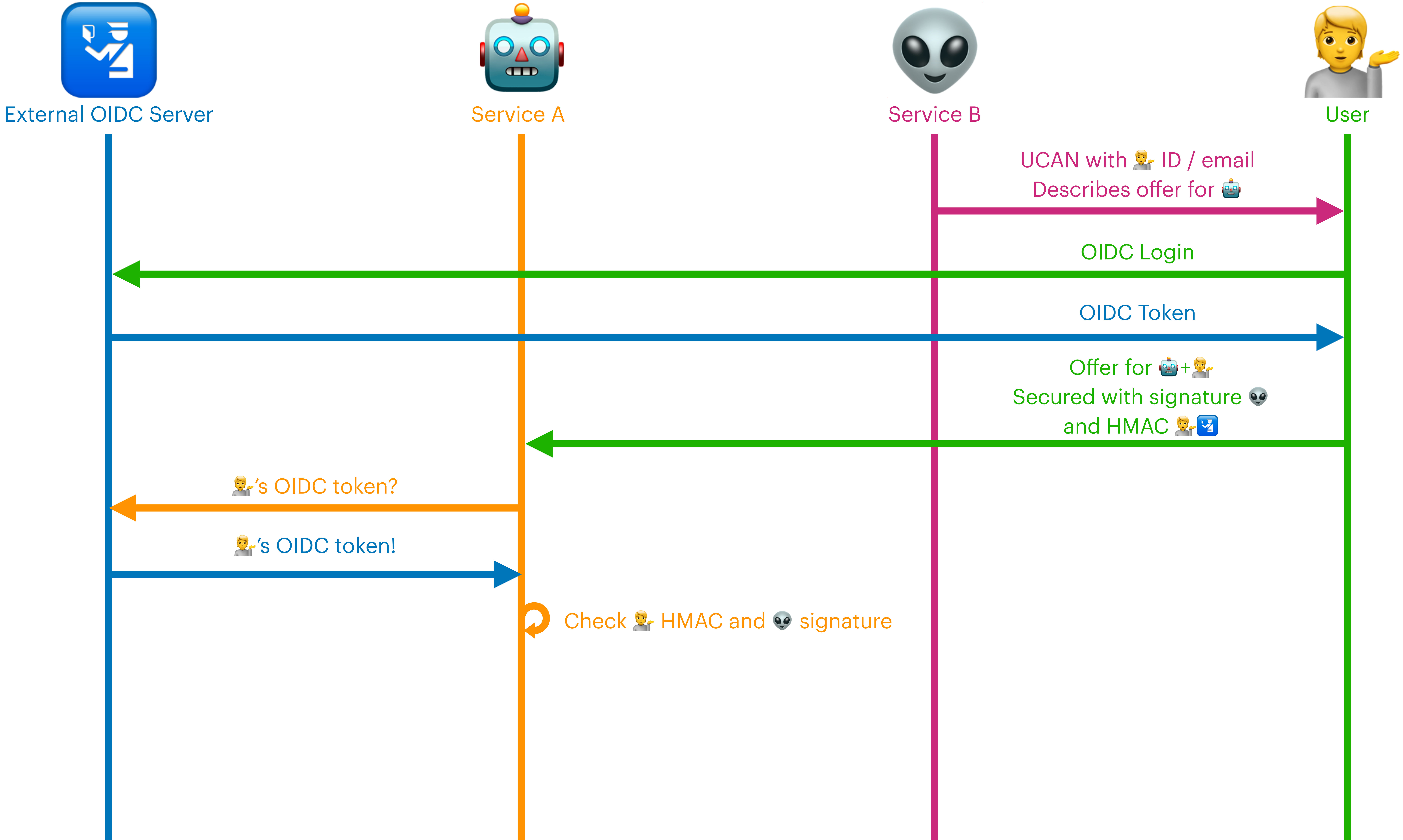
OIDC Token

Offer for 🤖+💁
Secured with signature 👽
and HMAC 💁🛂

💁's OIDC token?

💁's OIDC token!

Universal Auth & ID 🗝️



External OIDC Server

Service A

Service B

User

UCAN with 💁‍♂️ ID / email
Describes offer for 🤖

OIDC Login

OIDC Token

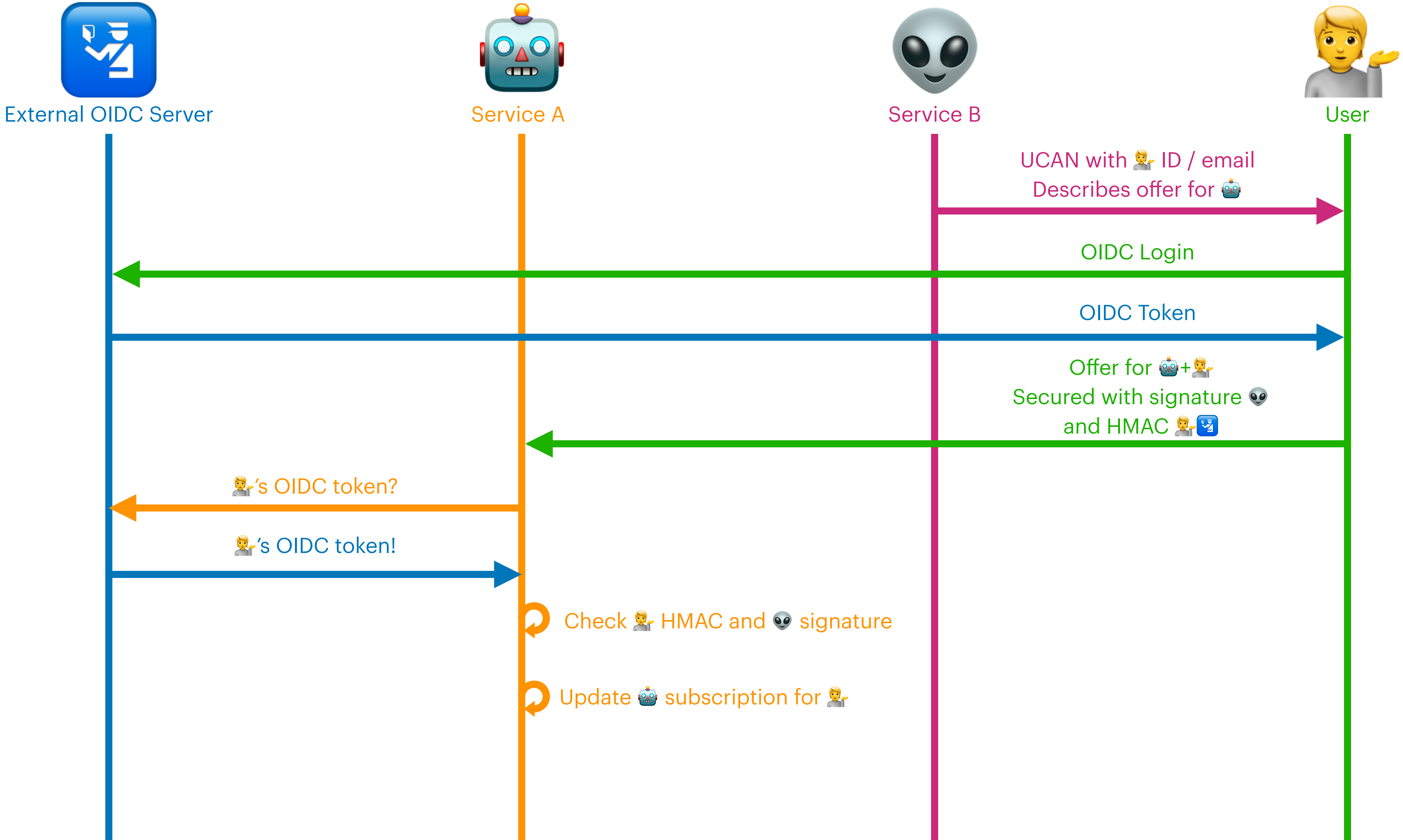
Offer for 🤖+💁‍♂️
Secured with signature 👽
and HMAC 💁‍♂️🛂

💁‍♂️'s OIDC token?

💁‍♂️'s OIDC token!

Check 💁‍♂️ HMAC and 👽 signature

Universal Auth & ID 🗝️



External OIDC Server

Service A

Service B

User

UCAN with 💁 ID / email
Describes offer for 🤖

OIDC Login

OIDC Token

Offer for 🤖+💁
Secured with signature 👽
and HMAC 💁🛂

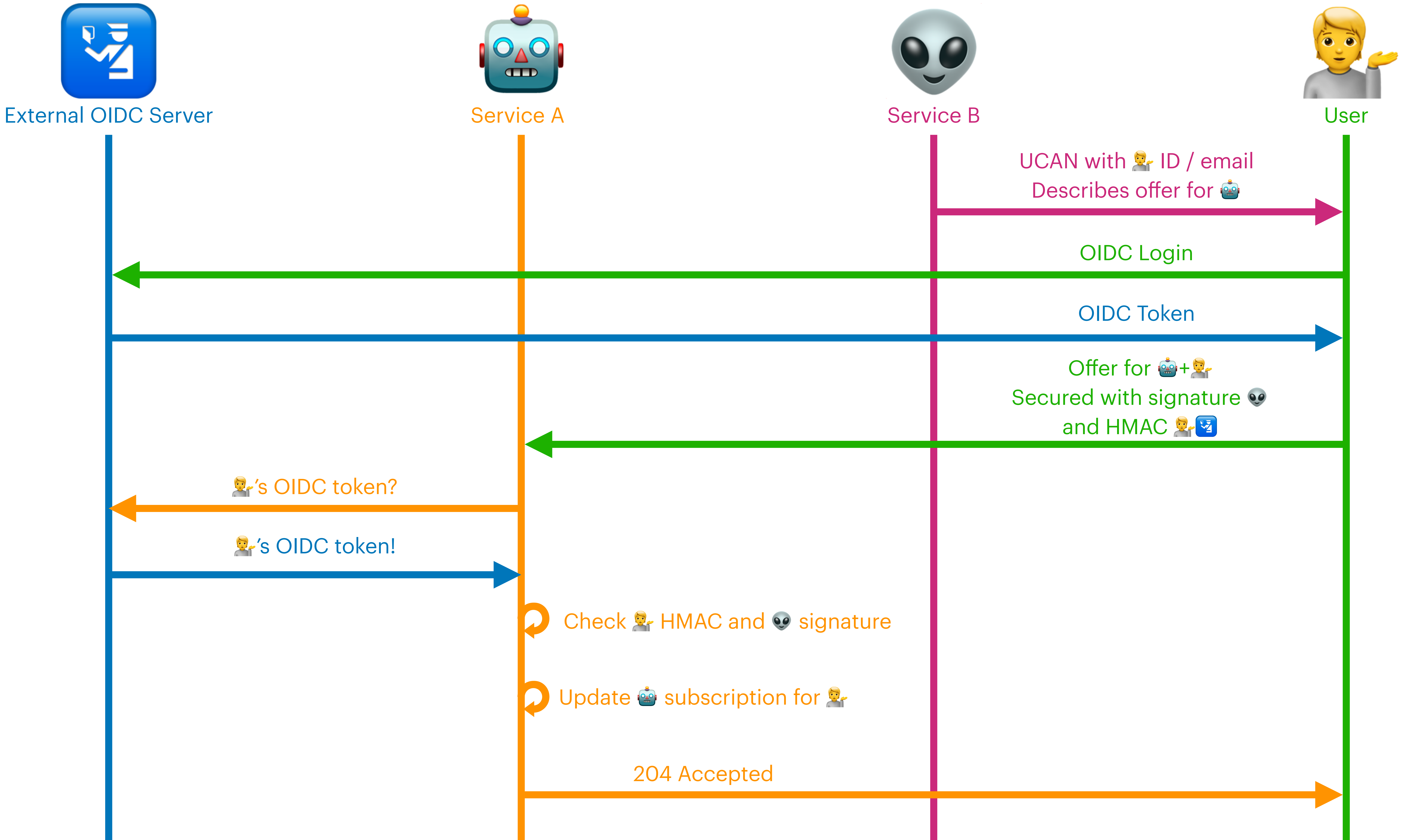
💁's OIDC token?

💁's OIDC token!

Check 💁 HMAC and 👽 signature

Update 🤖 subscription for 💁

Universal Auth & ID 🗝️



🛂 External OIDC Server

🤖 Service A

👽 Service B

💁 User

UCAN with 💁 ID / email
Describes offer for 🤖

OIDC Login

OIDC Token

Offer for 🤖+💁
Secured with signature 👽
and HMAC 💁🛂

💁's OIDC token?

💁's OIDC token!

Check 💁 HMAC and 👽 signature

Update 🤖 subscription for 💁

204 Accepted

Summary

🍱

Instead of immediately asking “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?**”.

The platforms you build on top of drive the design decisions you make in your products. With Elixir, **you are empowered** to tackle problems that in other platforms would feel impossible to solve without tradeoffs with heavy dependencies.

~ Chris McCord, What Makes Phoenix Presence Special

Getting Ready 🍱

Data > Compute

- Focus on data & structure
- Clarify “real” dependencies on data
- Start thinking about the properties in your code
- Adopt OCAP
- Use abstraction for declarative interfaces

🇧🇷 Thank You, CodeBEAM BR 🎉

brooklyn@fission.codes
https://fission.codes
github.com/expede
@expede