🌈WebNative🚀

How to put a full stack directly in the browser

✨Speakeasy JS🍻

Or:

Or:

The Beginnings of a WebOS

Or:

The Beginnings of a ~~WebOS~~

Wait, no

Or:

Or:

A Browser-Based File System, Location Independence, User Controlled Data, Self-Modifying Apps, & Serverless Auth

...plus some surprising things we've learned along the way

Brooklyn Zelenka

@expede

fission

Brooklyn Zelenka

@expede

- CTO at Fission
  - https://fission.codes
  - 100% FOSS
  - Obsoleting backends one function at a time
- PLT, VMs, Distributed Systems, Prev. ETH Core
- Founded Vancouver FP, Code & Coffee YVR
- FOSS — Witchcraft, Exceptional, Rescue, &c

fission

Speakeasy JS

shop.fission.codes
Code: SPEAKEASYJS

Stickers!

shop.fission.codes

Code: SPEAKEASYJS

This is the JavaScript meetup for
🥼 ***mad science***,
🧙‍♂️ ***hacking***, and
🧪 ***experiments***

SpeakeasyJS Homepage

This is the JavaScript meetup for

🥼 ***mad science***,

🧙‍♂️ ***hacking***, and

🧪 ***experiments***

SpeakeasyJS Homepage

This is the JavaScript meetup for

✅ 🥼 ***mad science***,

✅ 🧙‍♂️ ***hacking***, and

✅ 🧪 ***experiments***

👍 SpeakeasyJS Homepage

The Problem(s)

Starting Conditions

WebNative 🚀

The Web Today

WebNative 🚀

The Web Today

WebNative 🚀

The Web Today

WebNative 🚀

The Web Today

WebNative 🚀

The Web Today

WebNative 🚀

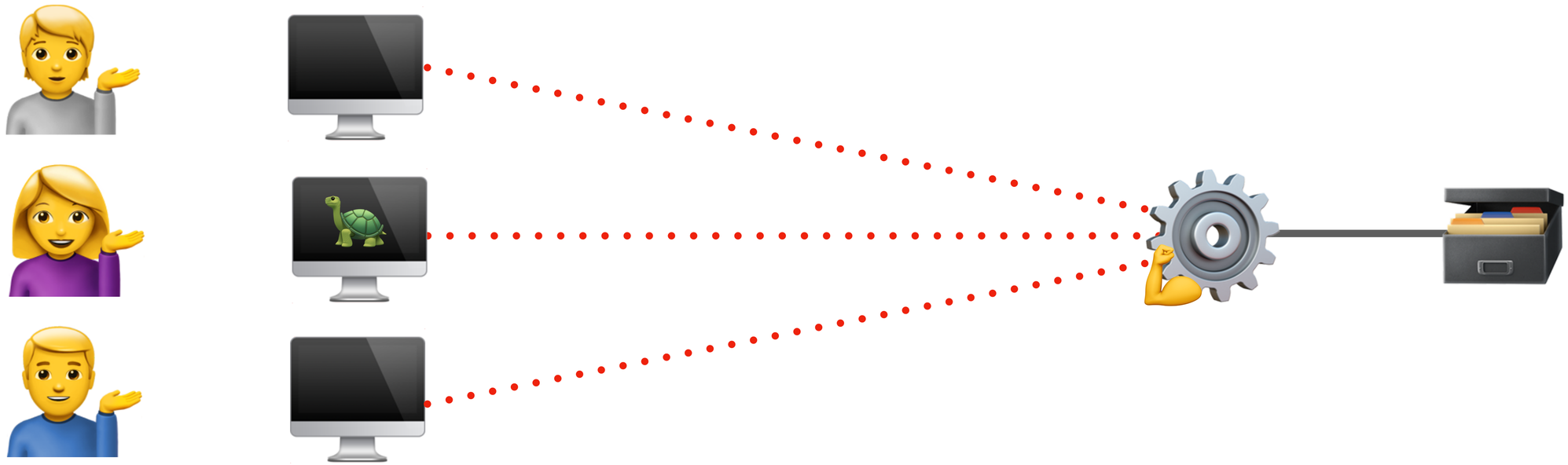
The Web Today

WebNative 🚀

The Web Today

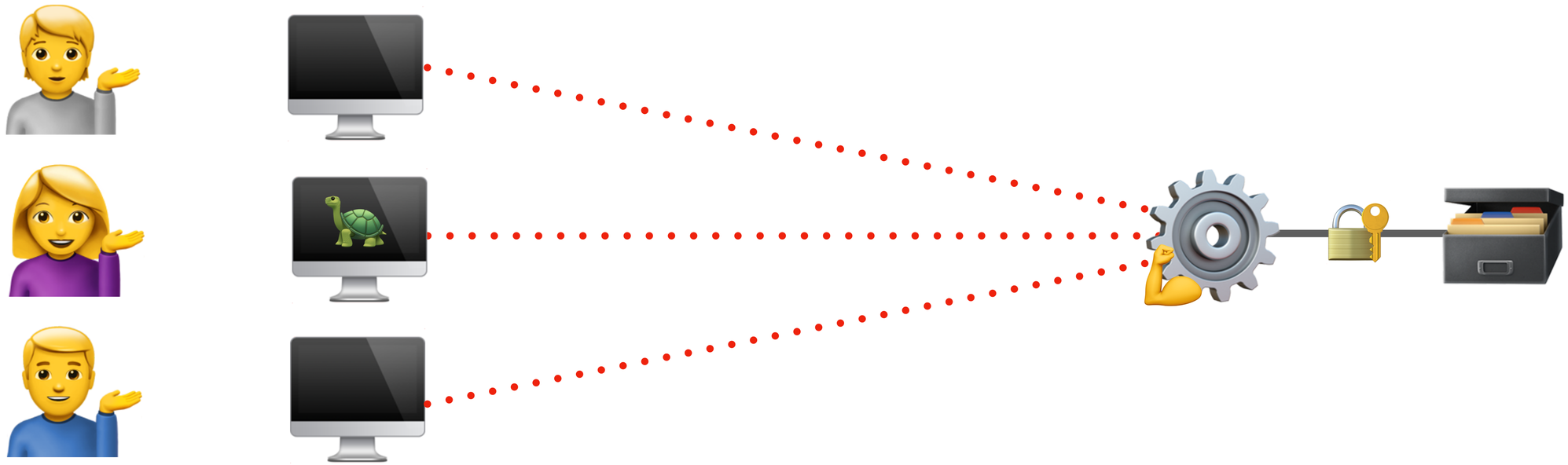
WebNative 🚀

The Web Today



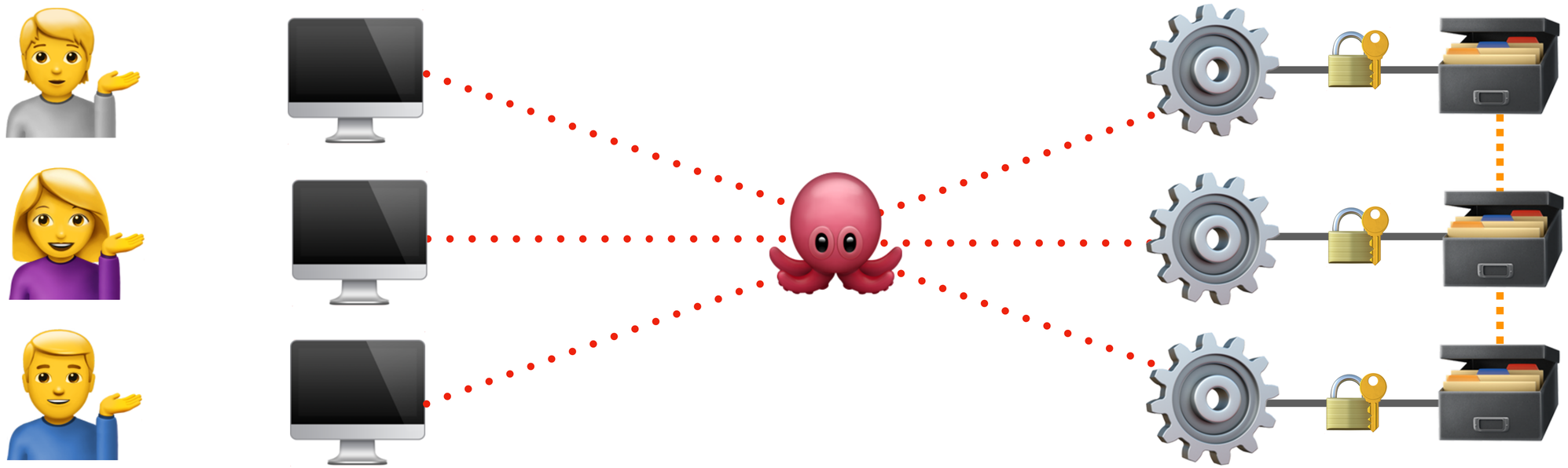
WebNative 🚀

The Web Today



WebNative 🚀

The Web Today



WebNative 🚀

*What We **Actually** Want*

WebNative 🚀

What We Act

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

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

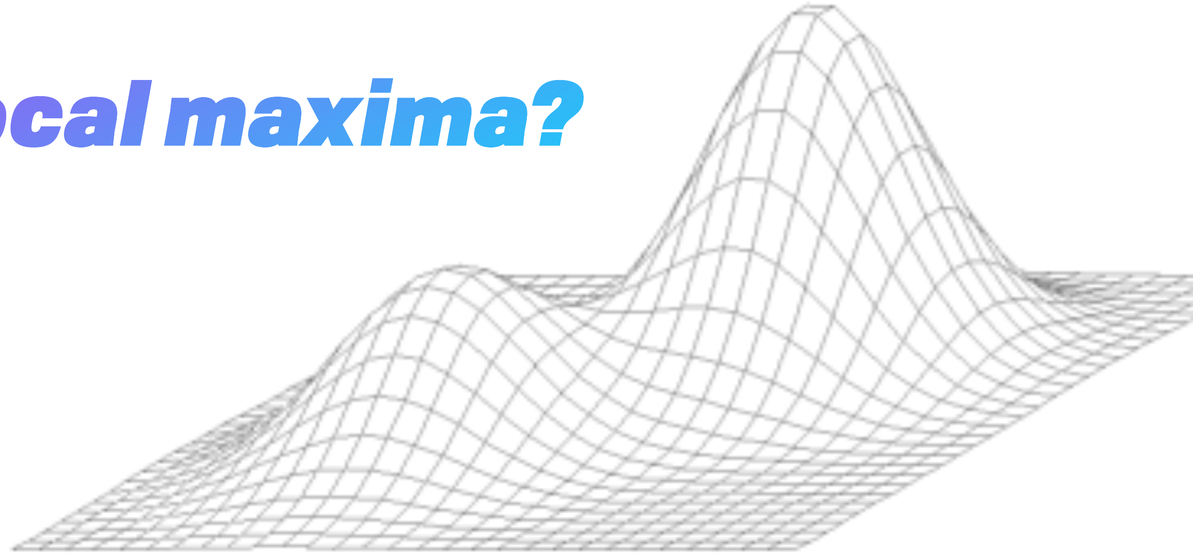
🦖 ☄️ 🌋 🌾 🏰 🏢 🚀

Is the way we do things ***today*** the “one true way”?

Will we be doing things this way in ***2025? 2050? 2100?***

Does knowledge ***always progress*** from good to better?

Are we stuck in a ***local maxima?***



WebNative 🚀

Natural Consequences

- Server-focus
  - Must learn more of stack
  - Single source of truth
  - DevOps, Docker, k8s
- Latency assumption
- FE deeply concerned with data sync

WebNative 🚀

What Even is a "Server"?

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

WebNative 🚀

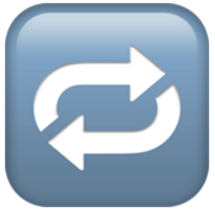
What Even is a "Server"?

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

Remember this list!

What if we turn the web architecture

Inside Out?



WebNative 🚀

Like Native... but for the Web 🧐

WebNative 🚀

Like Native... but for the Web 🧐

📱

WebNative 🚀

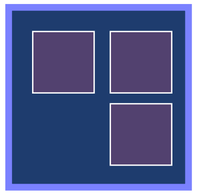
Like Native... but for the Web 🧐

WebNative 🚀

Like Native... but for the Web 🧐

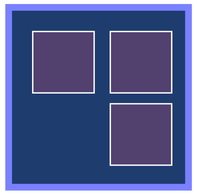
WebNative 🚀

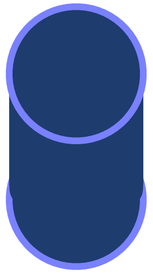
Like Native... but for the Web 🧐



WebNative 🚀

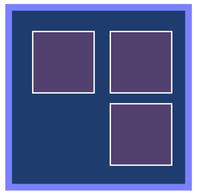
Like Native... but for the Web 🧐

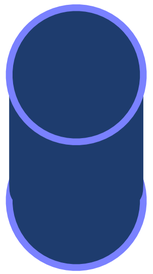




WebNative 🚀

Like Native... but for the Web 🧐





WebNative 🚀

New Assumptions, New Approach

- 2021 != 1991
  - Don't need to rely on client/server
  - Browsers are super powerful
  - UI & data = only essential parts
- Post-serverless, edge++
- New primitives ("game changers")
  - Location independent data 🙌
  - Browser-based encryption 💪
  - Consistency models (OT, CRDTs, RAFT) 🤝
    - i.e. State transfer -> state synchronization
- New features naturally fall out of the architecture
- Recognize that we're increasingly connected/networked
- Local-first means network efficient (in the normal case)

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

Local-first software

You own your data, in spite of the cloud

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

Ink & Switch

Martin Kleppmann
Adam Wiggins
Peter van Hardenberg
Mark McGranaghan
April 2019

WebNative 🚀

Bootstrapping from Browsers APIs

- WebCrypto API
- Web Workers
- Service Workers
- IndexedDB
- PWA & Web App Manifest

WebNative 🚀

How Many Steps Can We Skip?

WebNative 🚀

How Many Steps Can We Skip?

Users 👩🏿‍💻👩🏻‍🔬👨🏼‍🔬👷🏾‍♀️

Developer 👩🏾‍💻

WebNative 🚀

How Many Steps Can We Skip?

Users 👩🏿‍💻 👩🏻‍🔬 👨🏼‍🏫 👷🏾‍♂️

Browser 🖥️

REST / JSON-RPC / GraphQL ⬆️

Server ⚙️

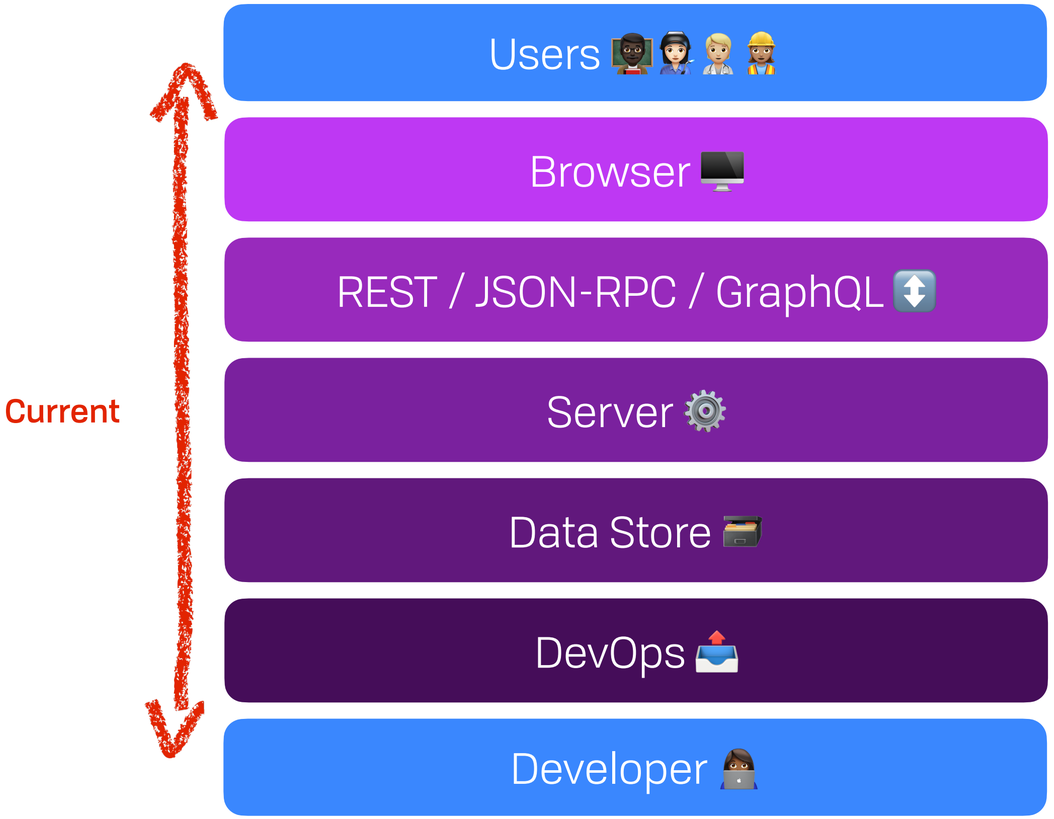
Data Store 🗃️

DevOps 📤

Developer 👩🏾‍💻

WebNative 🚀

How Many Steps Can We Skip?

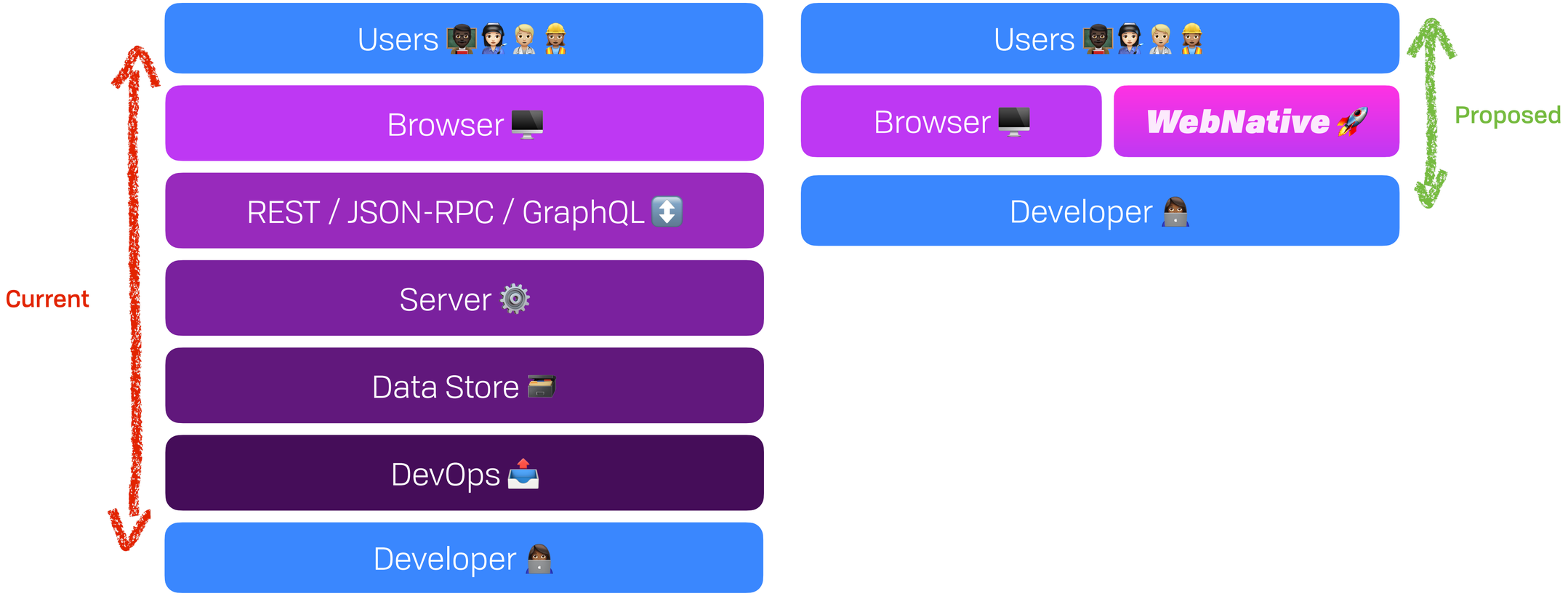


WebNative 🚀

How Many Steps Can We Skip?

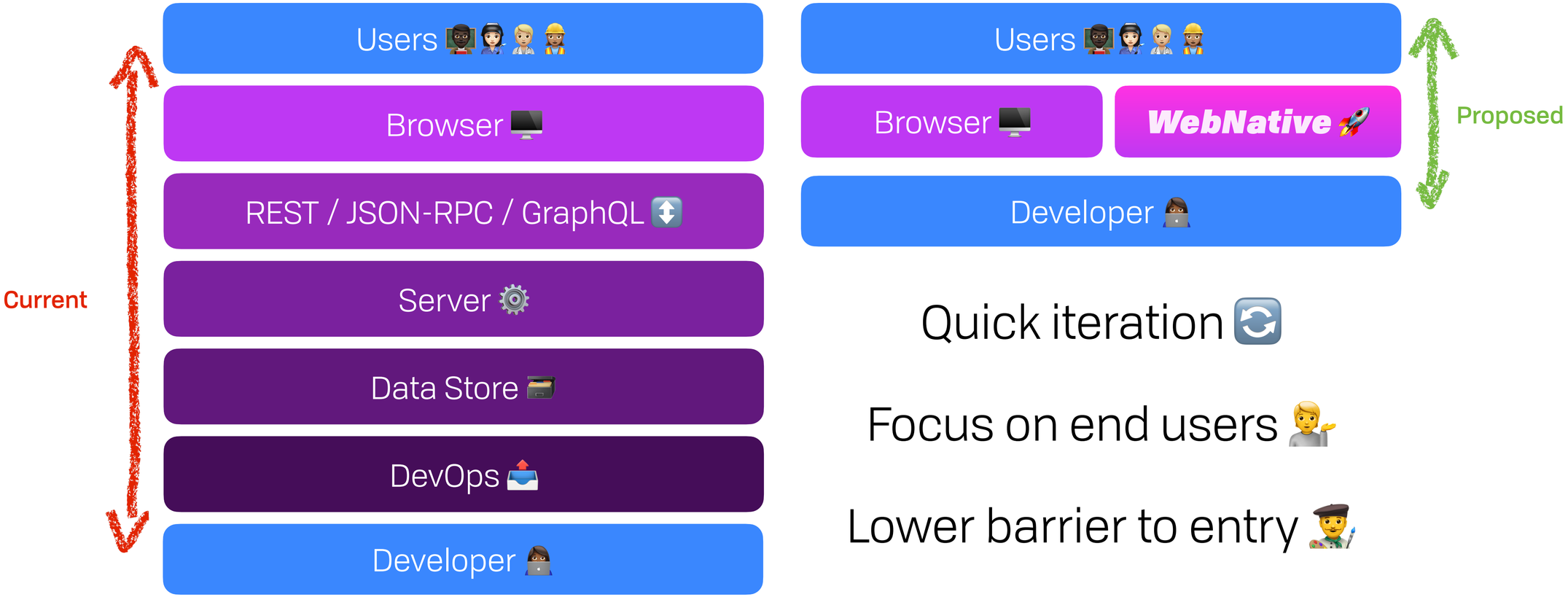
WebNative 🚀

How Many Steps Can We Skip?



WebNative 🚀

How Many Steps Can We Skip?



Quick iteration 🔄

Focus on end users 💁‍♀️

Lower barrier to entry 👨‍🎨

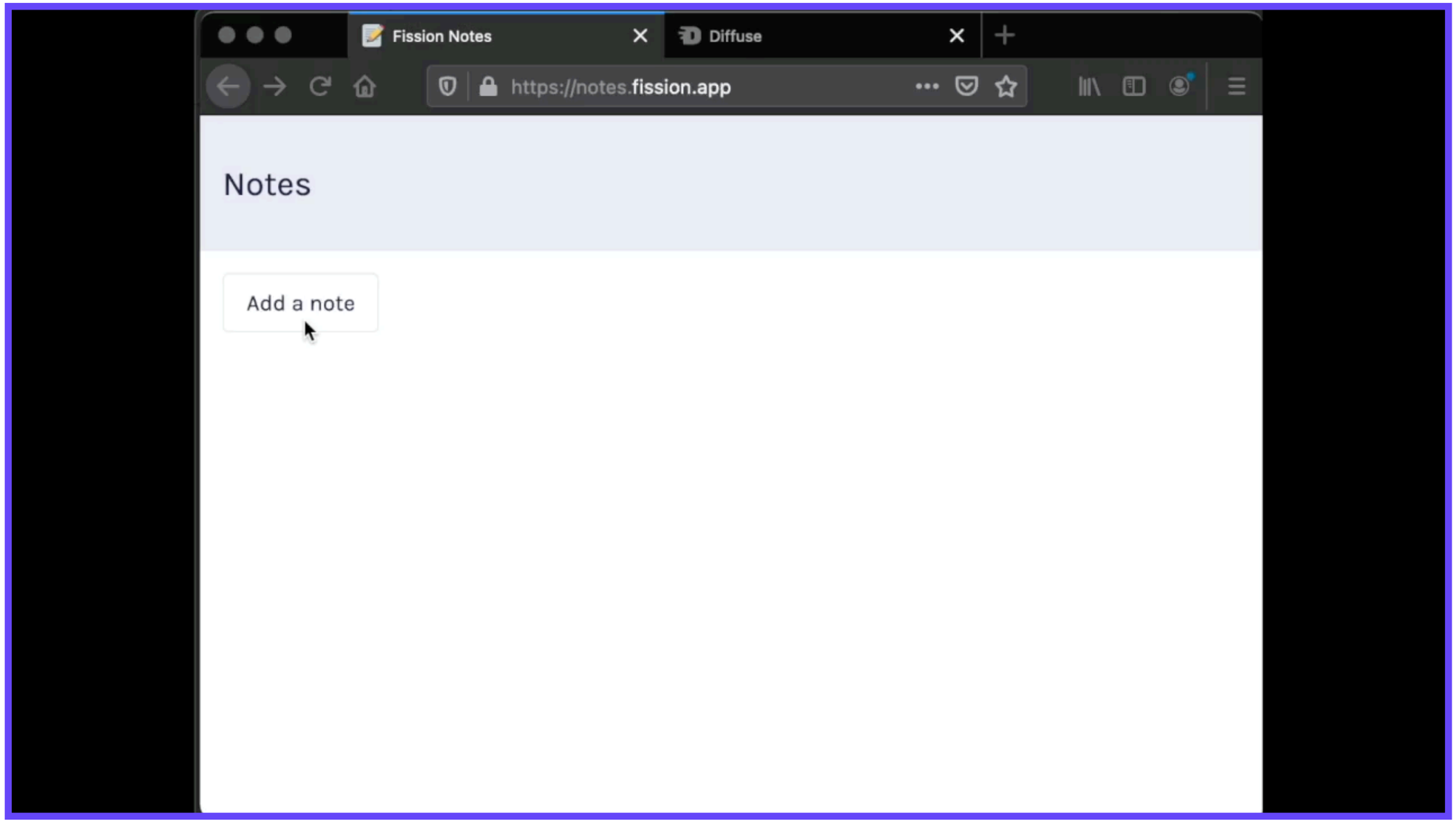
WebNative 🚀

Constraints

- Vanilla browser, no plugins
- UX as good or better than existing
- Literally no distinction between local and production
  - No server required, put it in the browser
- User controlled identity & data
- Open to participation
- Accessible offline
- At least as secure as existing apps

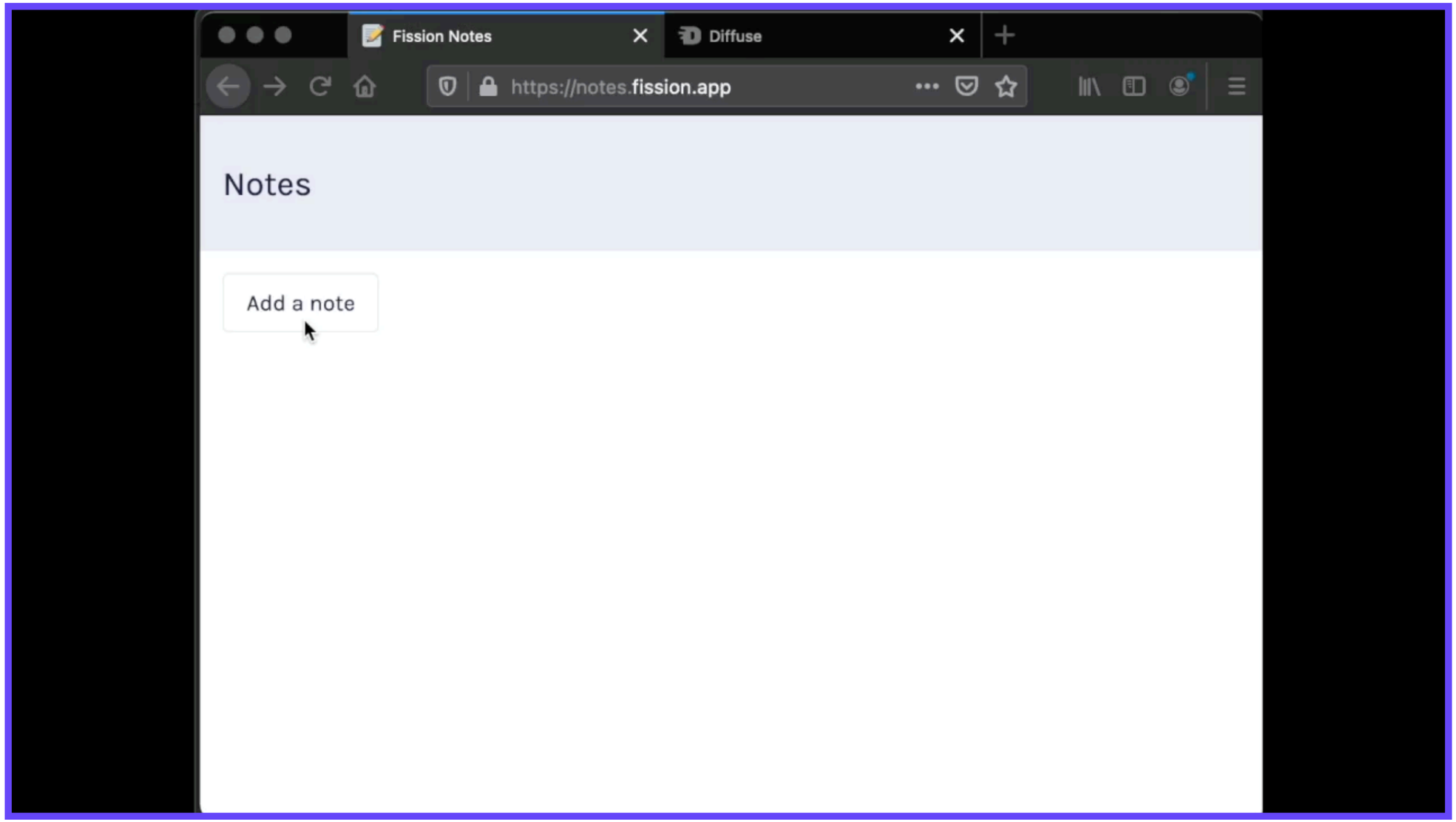
WebNative 🚀

Mini-Demo



WebNative 🚀

Mini-Demo



WebNative 🚀

Code

```
1  import { useEffect, useState } from 'react'
2  import * as wn from 'webnative'
3  import FileSystem from 'webnative/fs'
4
5  wn.setup.debug({ enabled: true })
6
7  export function useAuth() {
8    const [state, setState] = useState<wn.State>()
9    let fs: FileSystem | undefined
10
11   const authorise = () => {
12     if (state) {
13       wn.redirectToLobby(state.permissions)
14     }
15   }
```

```
17   useEffect(() => {
18     async function getState() {
19       const result = await wn.initialise({
20         permissions: {
21           app: {
22             name: 'Notes',
23             creator: 'walkah',
24           },
25         },
26       })
27       setState(result)
28     }
29
30     getState()
31   }, [])
32
33   switch (state?.scenario) {
34     case wn.Scenario.AuthSucceeded:
35     case wn.Scenario.Continuation:
36       fs = state.fs
37       break
38   }
```

WebNative 🚀

Code

```
1   import { useEffect, useState } from 'react'
2   import * as wn from 'webnative'
3   import FileSystem from 'webnative/fs'
4
5   wn.setup.debug({ enabled: true })
6
7   export function useAuth() {
8     const [state, setState] = useState<wn.State>()
9     let fs: FileSystem | undefined
10
11    const authorise = () => {
12      if (state) {
13        wn.redirectToLobby(state.permissions)
14      }
15    }
```

Auth doesn't even leave your ***browser*** 👆

```
17    useEffect(() => {
18      async function getState() {
19        const result = await wn.initialise({
20          permissions: {
21            app: {
22              name: 'Notes',
23              creator: 'walkah',
24            },
25          },
26        })
27        setState(result)
28      }
29
30      getState()
31    }, [])
32
33    switch (state?.scenario) {
34      case wn.Scenario.AuthSucceeded:
35      case wn.Scenario.Continuation:
36        fs = state.fs
37        break
38    }
```

WebNative 🚀

Code

```
const createNote = async () => {
  if (!fs || !fs.appPath) return

  console.log(`📝 Creating new note`)
  let fileName = 'Untitled'
  let num = 0
  while (await fs.exists(fs.appPath(`${fileName}.md`))) {
    num++
    fileName = `Untitled ${num}`
  }

  try {
    const encoder = new TextEncoder()
    await fs.add(fs.appPath(`${fileName}.md`), encoder.encode('') as Buffer)
    await fs.publish()
    await listNotes()
    setCurrentNote(notes.find((note) => note.name === `${fileName}.md`))
    setContent('')
  } catch (e) {
    console.error(e)
  }
}
```

If React is “just the view layer”,
then ***WebNative is “just the data layer”***

It turns out the data layer
touches lots of other things

WebNative🚀

Stack

WebNative 🚀

Stack

1st & 3rd Party

Dev's App

Business Logic & View

WebNative 🚀

Stack

1st & 3rd Party

Dev's App

Business Logic & View

API

Platform Abstractions

WebNative SDK

WebNative 🚀

Stack

1st & 3rd Party

Dev's App

Business Logic & View

API

Platform Abstractions

WebNative SDK

⬆️ Apps

⬇️ Core Technology

WebNative 🚀 Stack

1st & 3rd Party

Dev's App
Business Logic & View

API

Platform Abstractions
WebNative SDK

⬆️ Apps

⬇️ Core Technology

Auth & ID

Command/Mutation
UCAN

Read/Query
Cryptree

Identity
did:key

WebNative 🚀 Stack

1st & 3rd Party

Dev's App
Business Logic & View

API

Platform Abstractions
WebNative SDK

⬆️ Apps

⬇️ Core Technology

Command/Mutation
UCAN

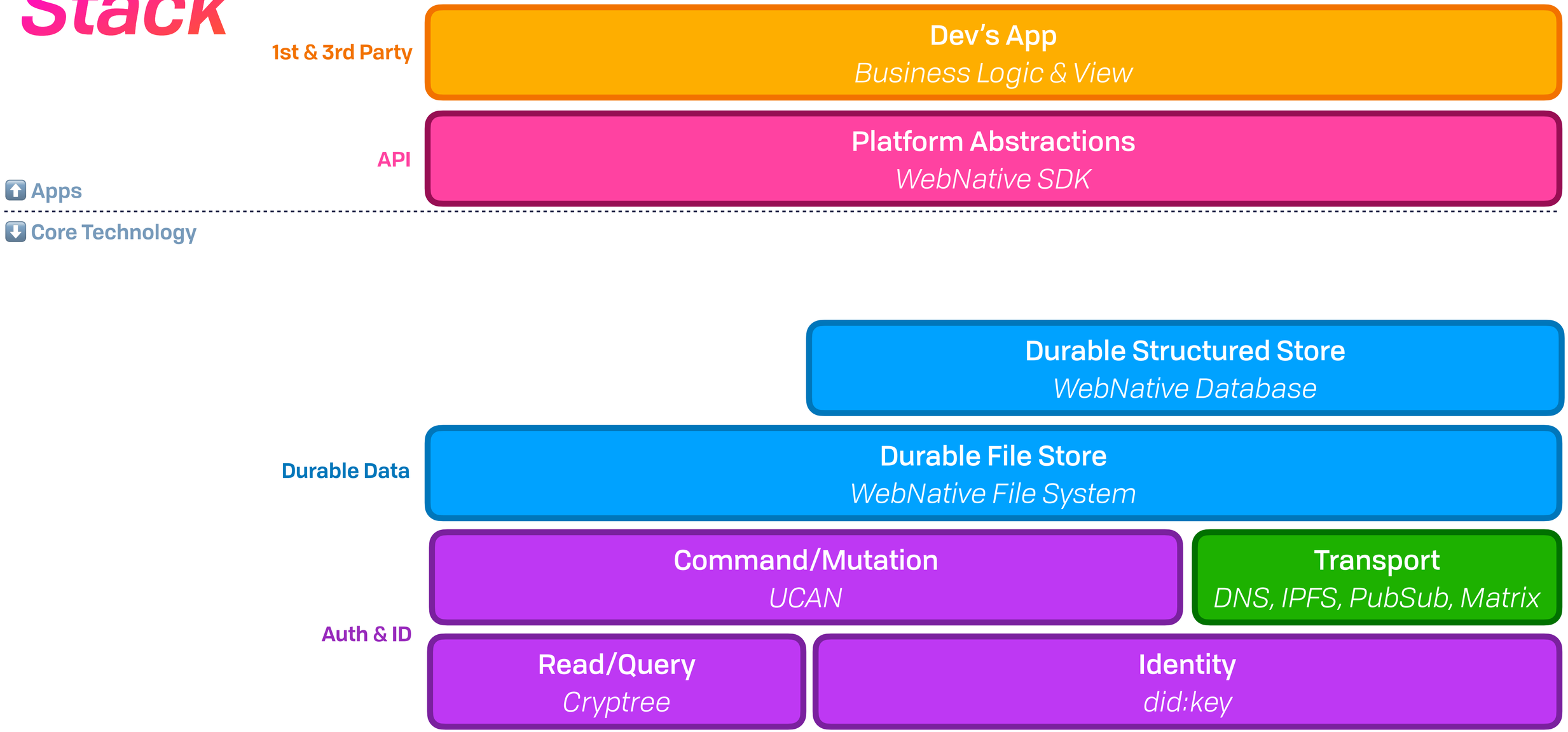
Transport
DNS, IPFS, PubSub, Matrix

Auth & ID

Read/Query
Cryptree

Identity
did:key

WebNative 🚀 Stack



WebNative 🚀 Stack

⬆️ Apps

⬇️ Core Technology

1st & 3rd Party

Dev's App
Business Logic & View

API

Platform Abstractions
WebNative SDK

Broadcast

Collaboration, Chat, Instant Sync
Soft Realtime Store

Global: Aggregation, Forms, Feeds
Gossip Broadcast

Offline & Async Sharing
Exchange Store

Durable Structured Store
WebNative Database

Durable Data

Durable File Store
WebNative File System

Auth & ID

Command/Mutation
UCAN

Transport
DNS, IPFS, PubSub, Matrix

Read/Query
Cryptree

Identity
did:key

WebNative 🚀

Painting a Picture 🖼️

- Go from zero to production on a plane ✈️
- Move data to compute and vice versa 🔄
- Publish updates from inside the browser 🚀
  - Code is data = self modifying apps 🐣
- Anyone can be a service provider (lower bar to entry) 💁‍♀️💁‍♂️
  - Including adversarial cooperation

Content Addressed Data

Content Addressed Data

It works offline and online, totally distributed & concurrent,
anyone can create or request data,
& data is always changing.

Content Addressed Data

It works offline and online, totally distributed & concurrent,
anyone can create or request data,
& data is always changing.

Content Addressed Data

It works offline and online, totally distributed & concurrent,
anyone can create or request data,
& data is always changing.

Great!

Content Addressed Data

It works offline and online, totally distributed & concurrent,
anyone can create or request data,
& data is always changing.

Great!

Content Addressed Data

It works offline and online, totally distributed & concurrent,
anyone can create or request data,
& data is always changing.

Great!

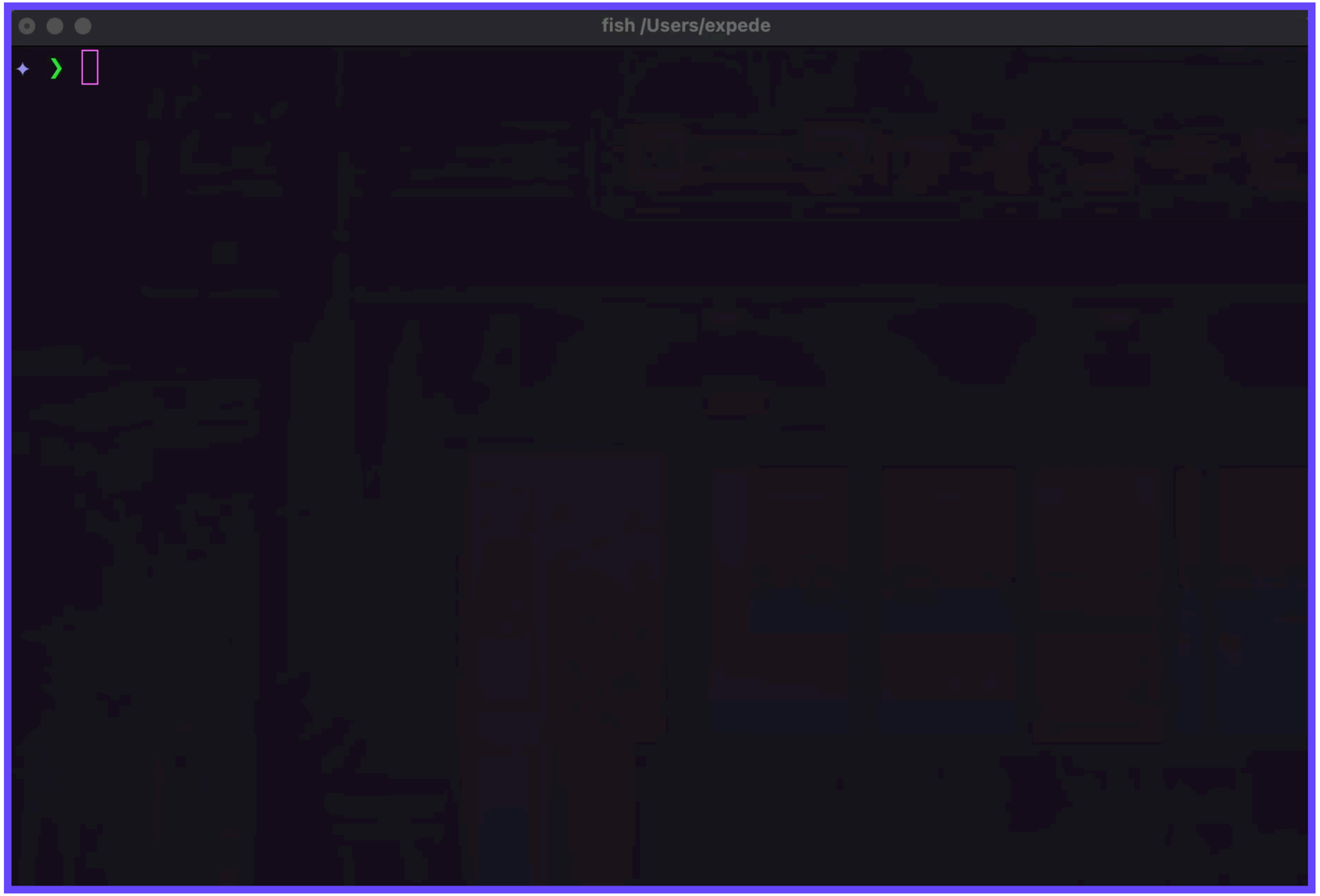
How do you even get a consistent pointer?

Content Addressed Data

Pushing Bytes Around

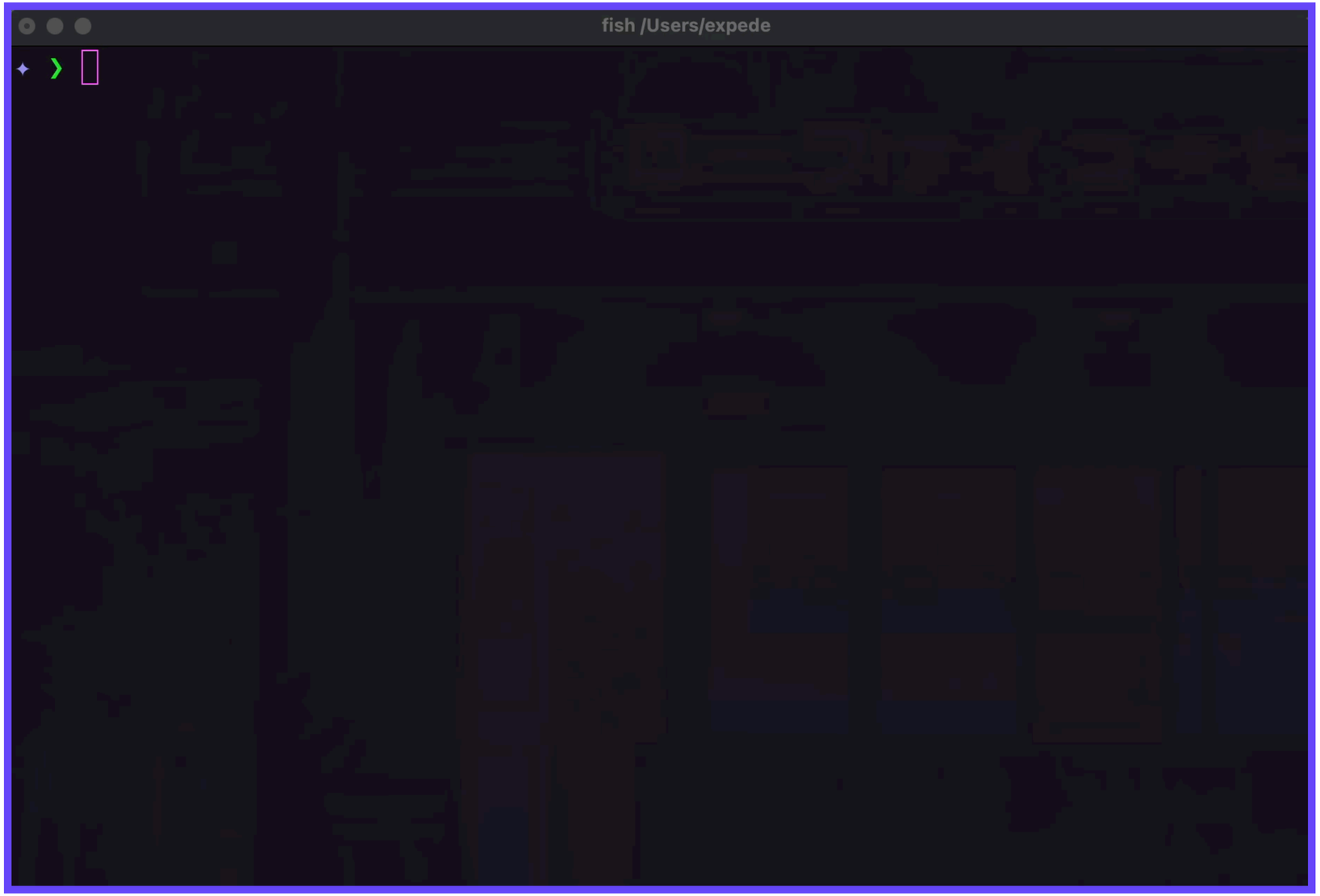
Content Addressed Data

Unique Hash ~ UUID++



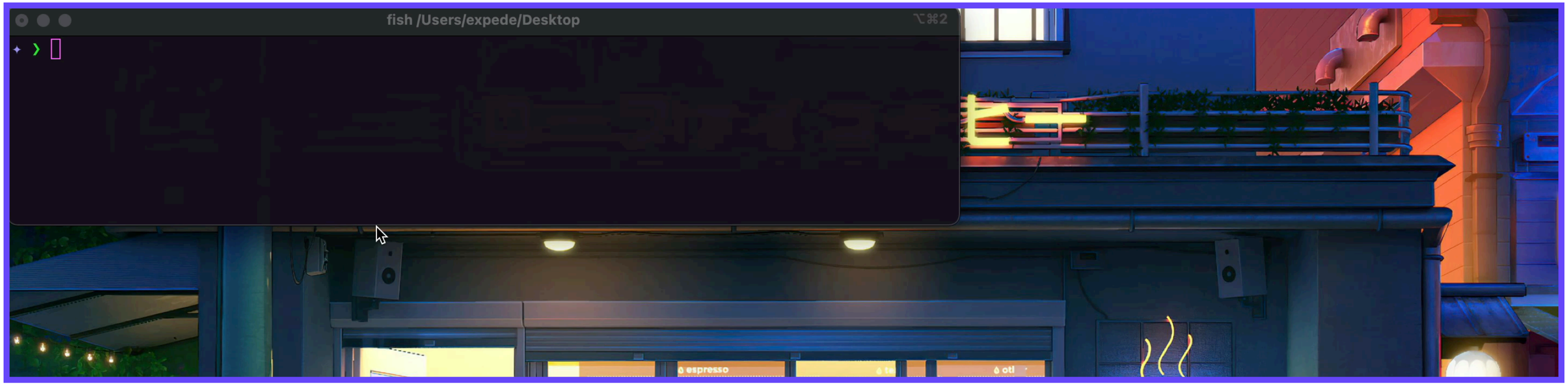
Content Addressed Data

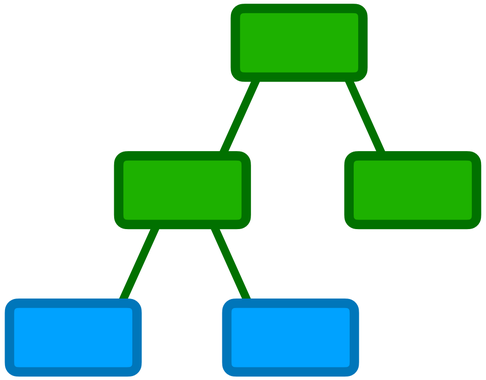
Unique Hash ~ UUID++



Content Addressed Data

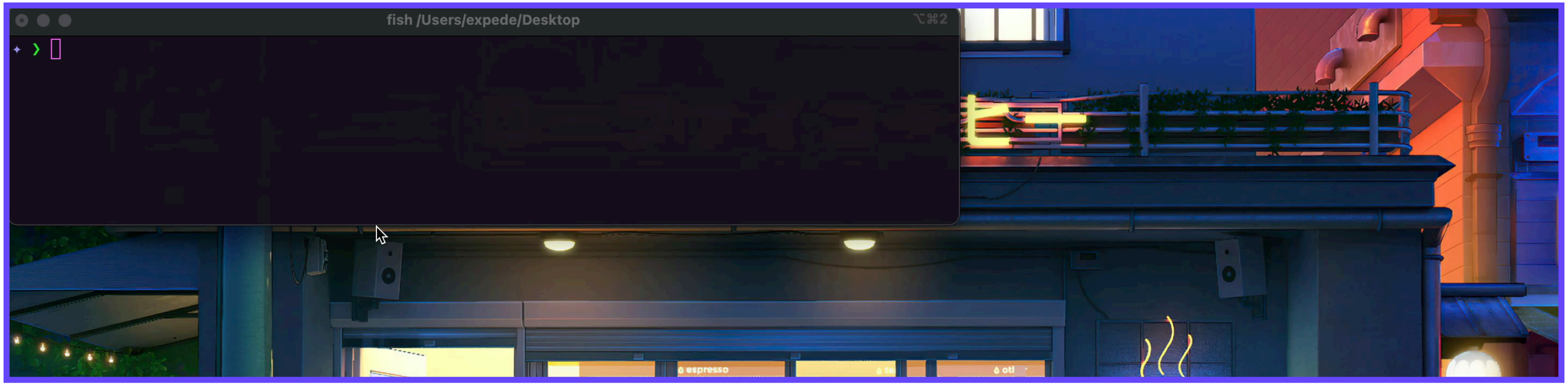
Arbitrary Content

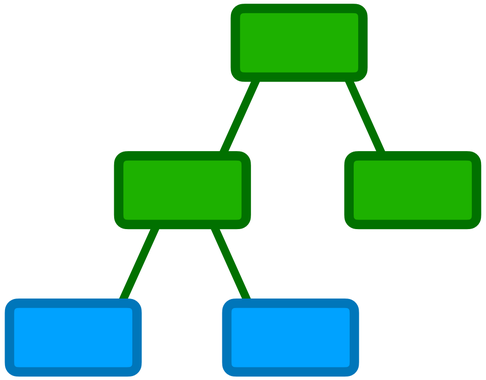




Content Addressed Data

Arbitrary Content





Content Addressed Data

Location Addressing

- Predominantly single-source (per file) server/client
- Like a key/value store {ip => {path => content}}
  - DNS maps names to IP addresses
- Focused on the physical network
- Mutable addressing
  - www.foo.com/baz may be JSON today, but a video tomorrow
  - ...or altered content

Content Addressed Data

Location Addressing

- Predominantly single-source (per file) server/client
- Like a key/value store {ip => {path => content}}
  - DNS maps names to IP addresses
- Focused on the physical network
- Mutable addressing
  - www.foo.com/baz may be JSON today, but a video tomorrow
  - ...or altered content

VIRTUAL ADDRESS

PHYSICAL LOCATION

Content Addressed Data

Universal / Content-Based Routing

- A layer of abstraction above location
- Like a key/value store {hash(content) => content}
  - Content hash AKA “content identifier” or CID
  - Special “universal” relationship to content
- Focused on the ***data***
  - Who cares where it’s stored?
  - Efficient auto-caching
- Still have paths
  - Immutable DAG
  - No loops

VIRTUAL ADDRESS

PHYSICAL LOCATION

Content Addressed Data

Universal / Content-Based Routing

- A layer of abstraction above location
- Like a key/value store {hash(content) => content}
  - Content hash AKA “content identifier” or CID
  - Special “universal” relationship to content
- Focused on the ***data***
  - Who cares where it’s stored?
  - Efficient auto-caching
- Still have paths
  - Immutable DAG
  - No loops

CONTENT ID

VIRTUAL ADDRESS

PHYSICAL LOCATION

Content Addressed Data

Hash-Linked Data

Content Addressed Data

Hash-Linked Data

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

Content Addressed Data

Hash-Linked Data

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

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

Content Addressed Data

Hash-Linked Data

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

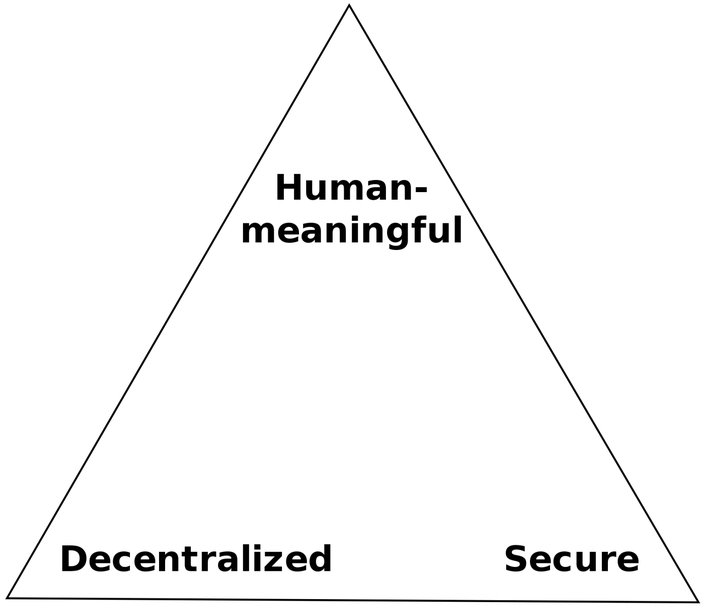
```
{
  Qmabcdef…: {
    data: “FISSION”,
    links: [
      {name: “city”,  hash: Qm1gb5sn…},
      {name: “about”, hash: Qmzyxwvu…}
    ]
  }
}
```

ipfs cat /ipfs/Qm123456…/company/about/founder
=> “Brooke”

Content Addressed Data

Tradeoffs

- Equality vs identity
  - Recovering identity from structural equality, but not vice-versa
- Caching is trivial
  - Data fetches
  - Artifacts
  - Results of computation
- Zooko's Triangle



Content Addressed Data

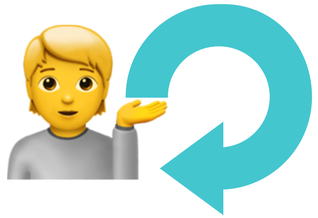
P2P Discovery, Lookup, Transport

Content Addressed Data

P2P Discovery, Lookup, Transport

Content Addressed Data

P2P Discovery, Lookup, Transport

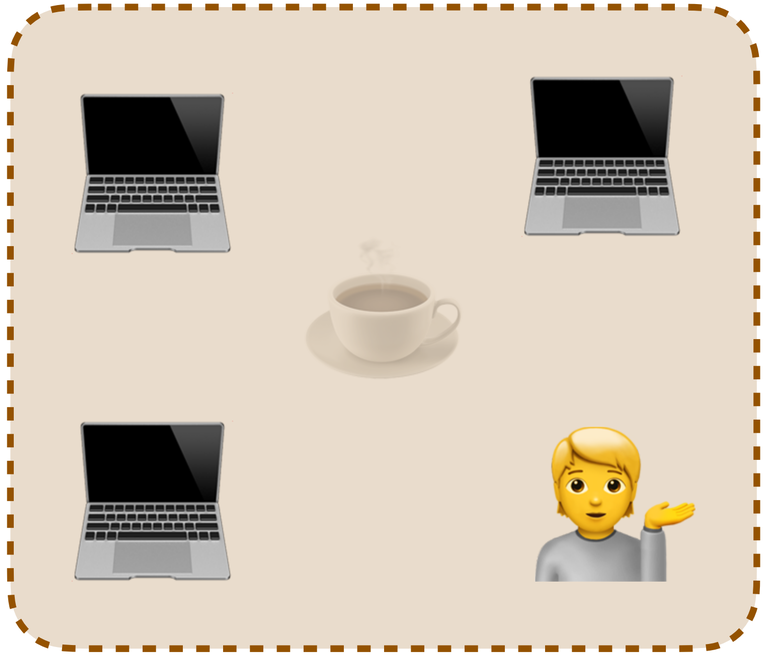


Content Addressed Data

P2P Discovery, Lookup, Transport

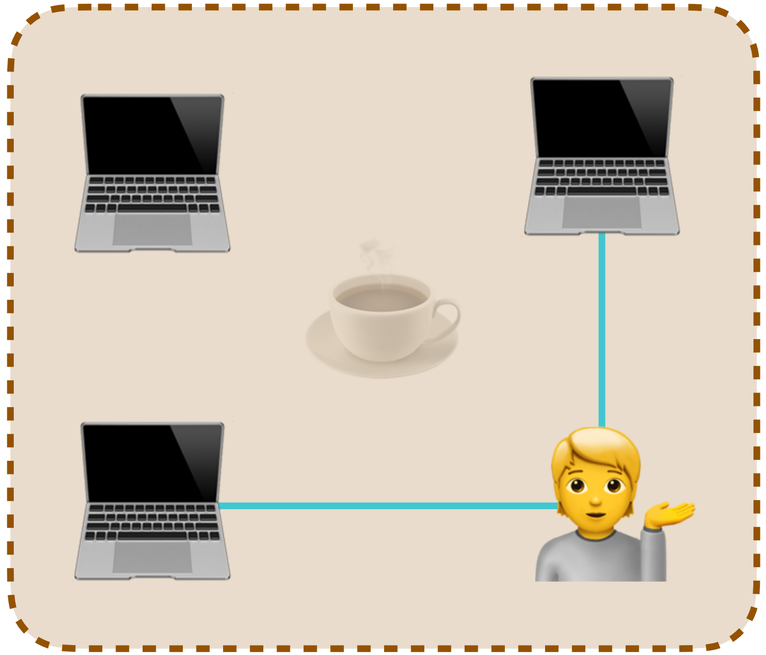
Content Addressed Data

P2P Discovery, Lookup, Transport



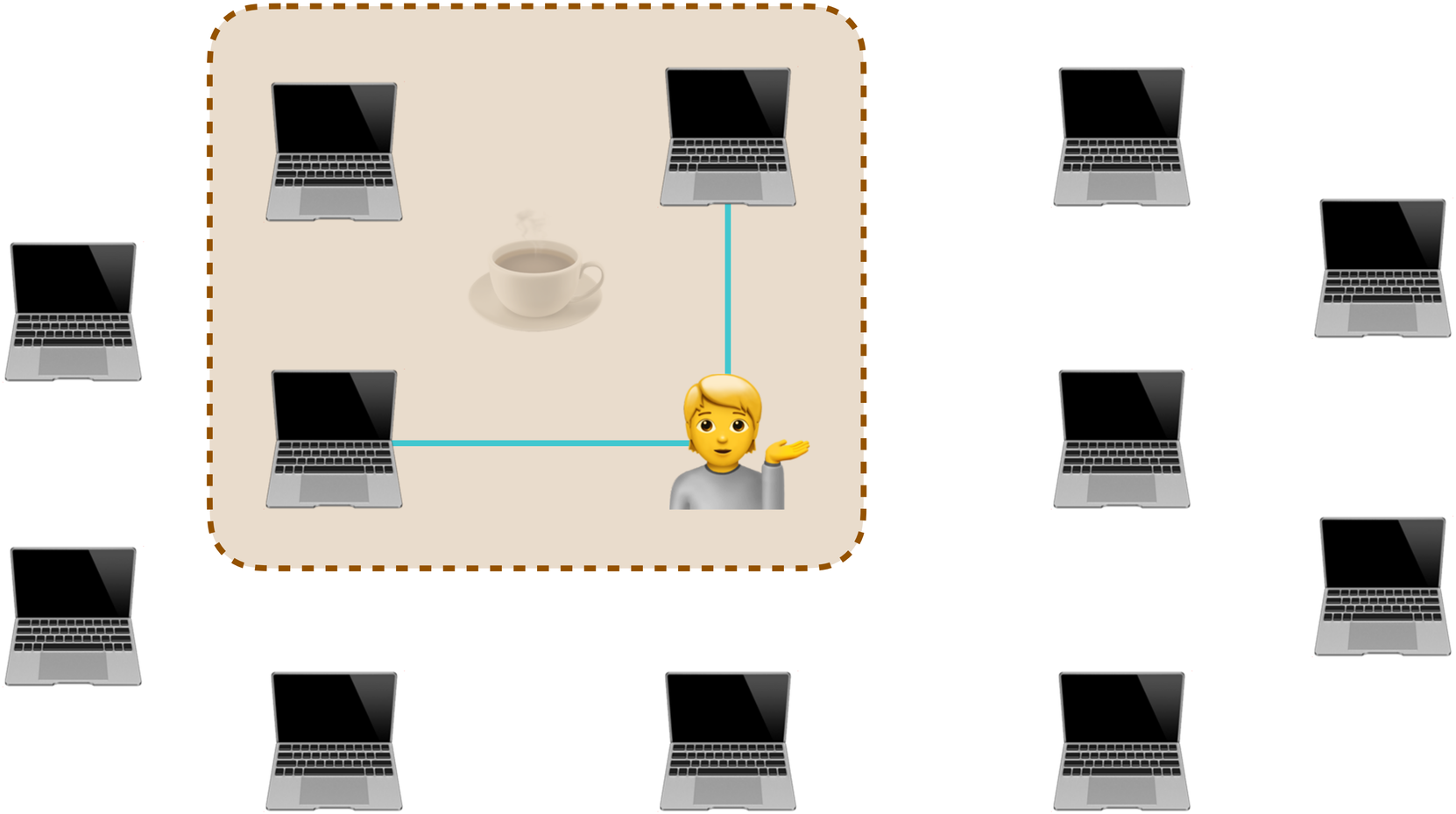
Content Addressed Data

P2P Discovery, Lookup, Transport



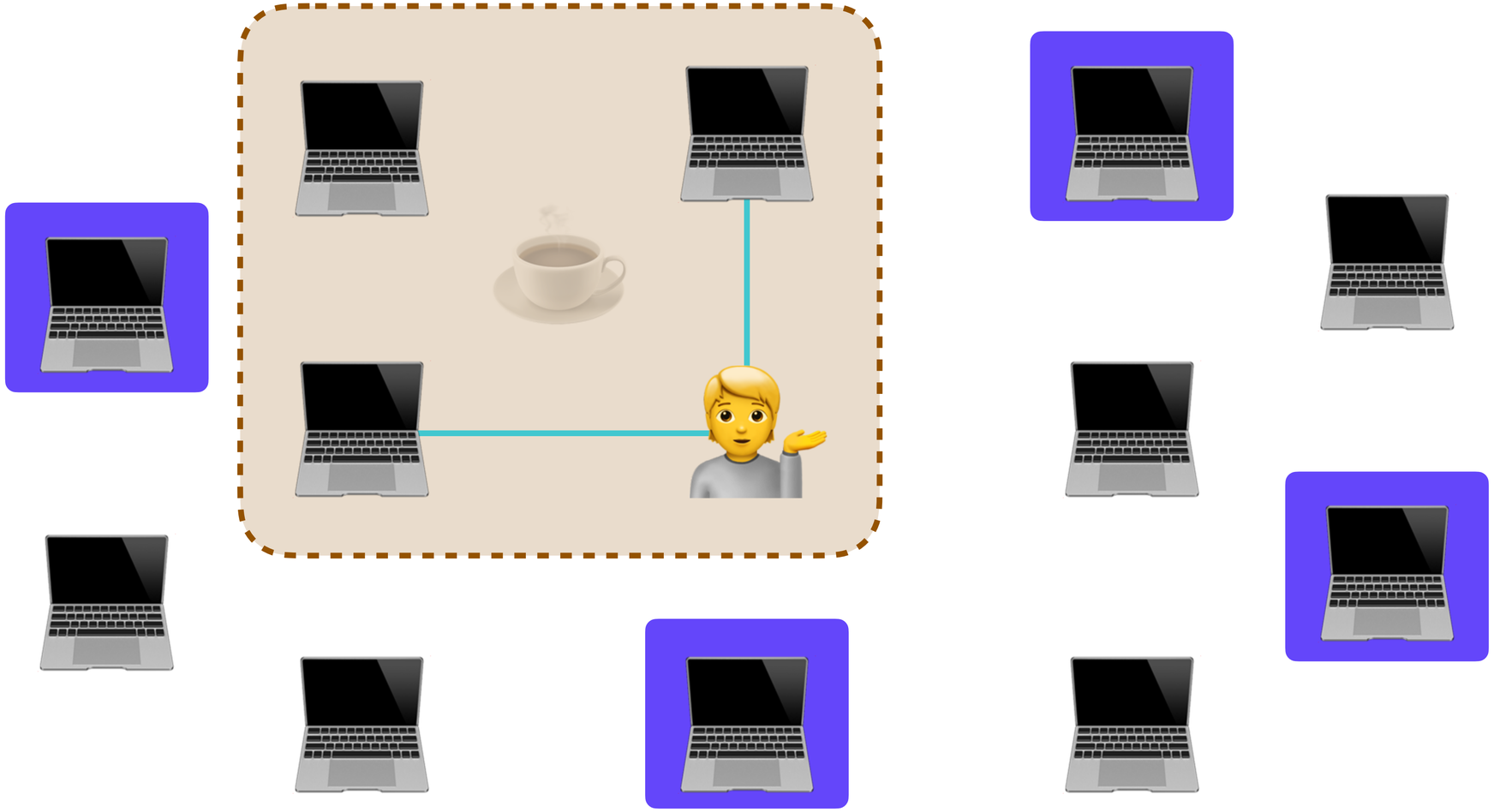
Content Addressed Data

P2P Discovery, Lookup, Transport



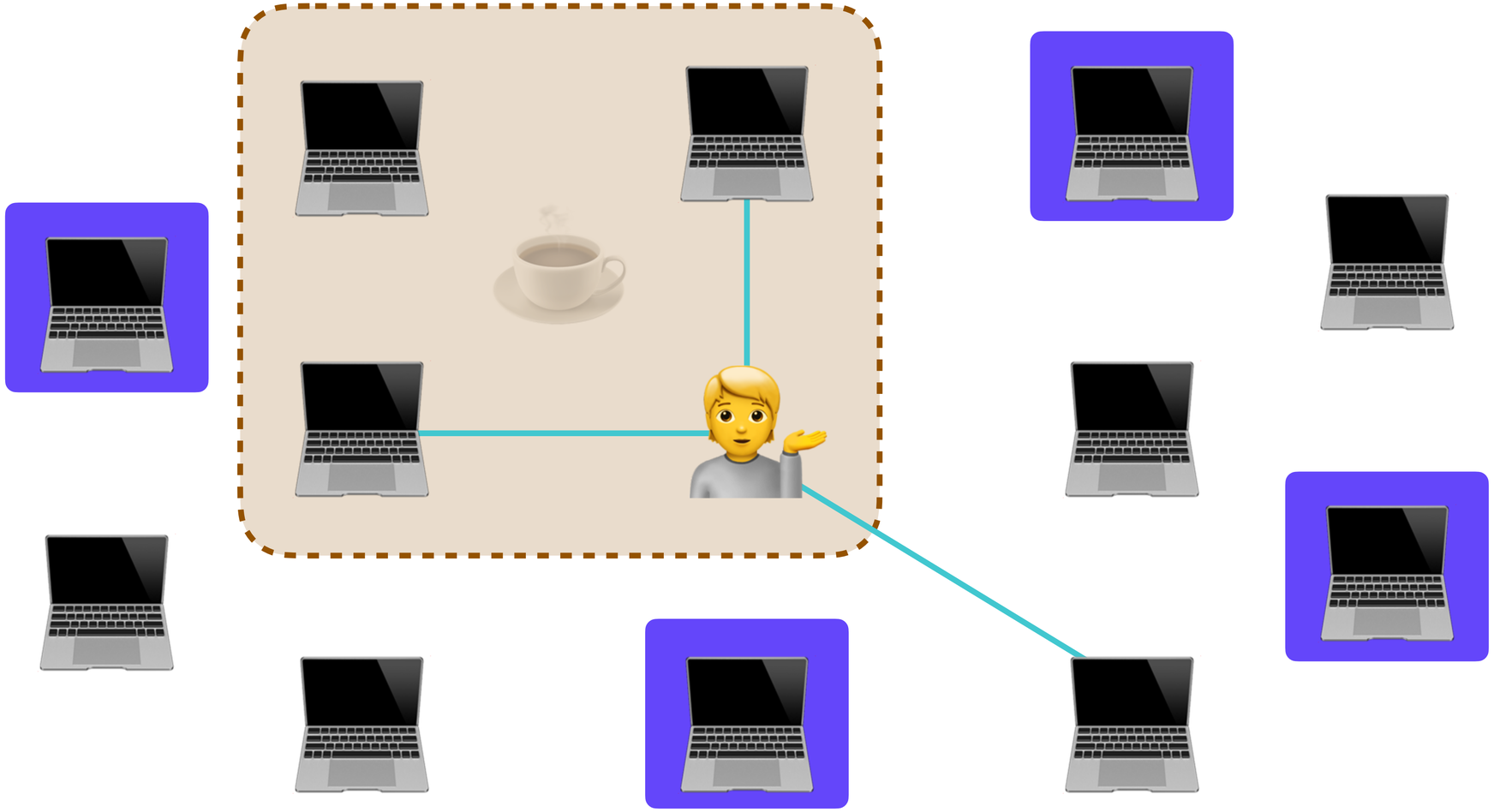
Content Addressed Data

P2P Discovery, Lookup, Transport



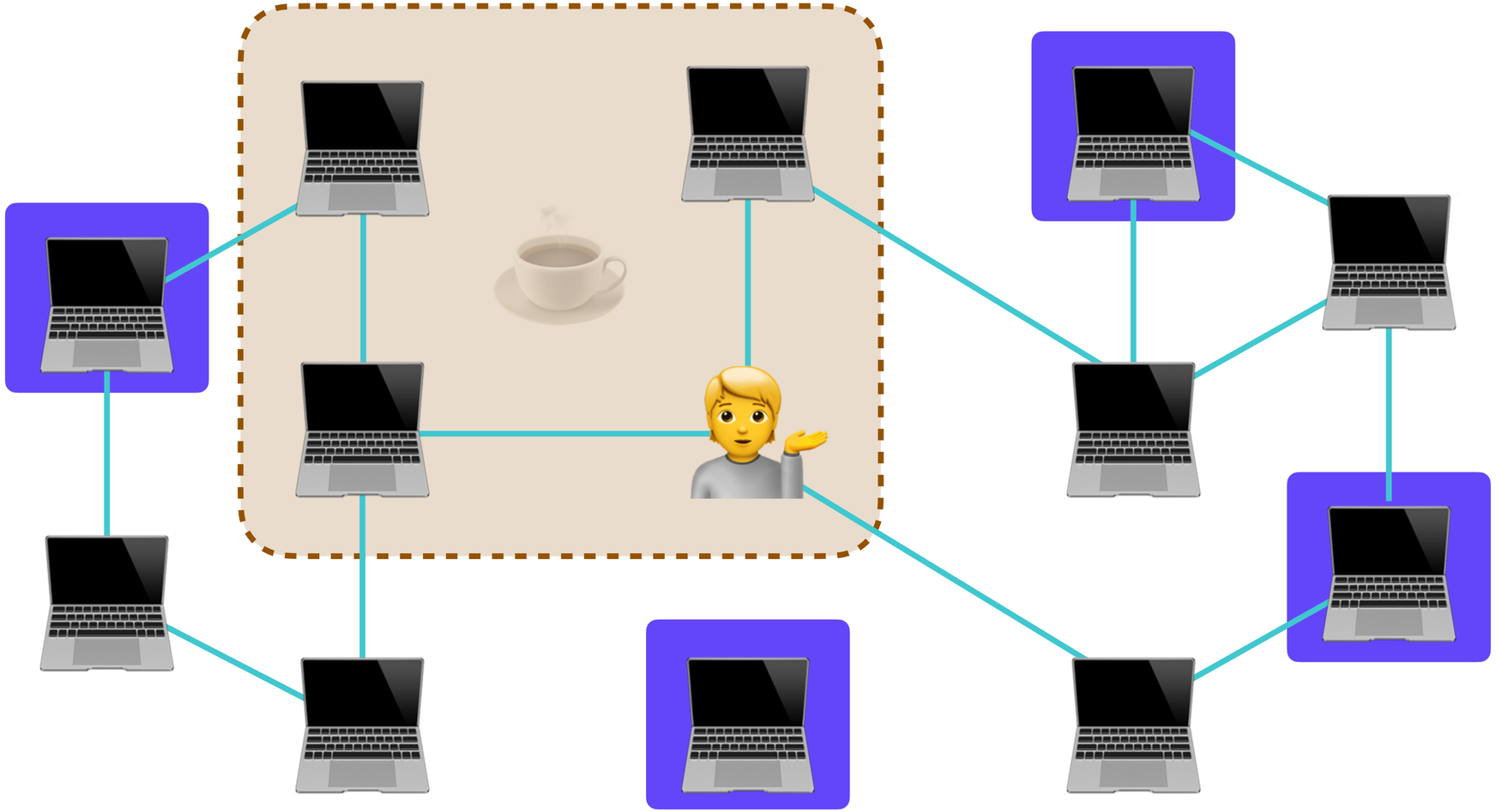
Content Addressed Data

P2P Discovery, Lookup, Transport



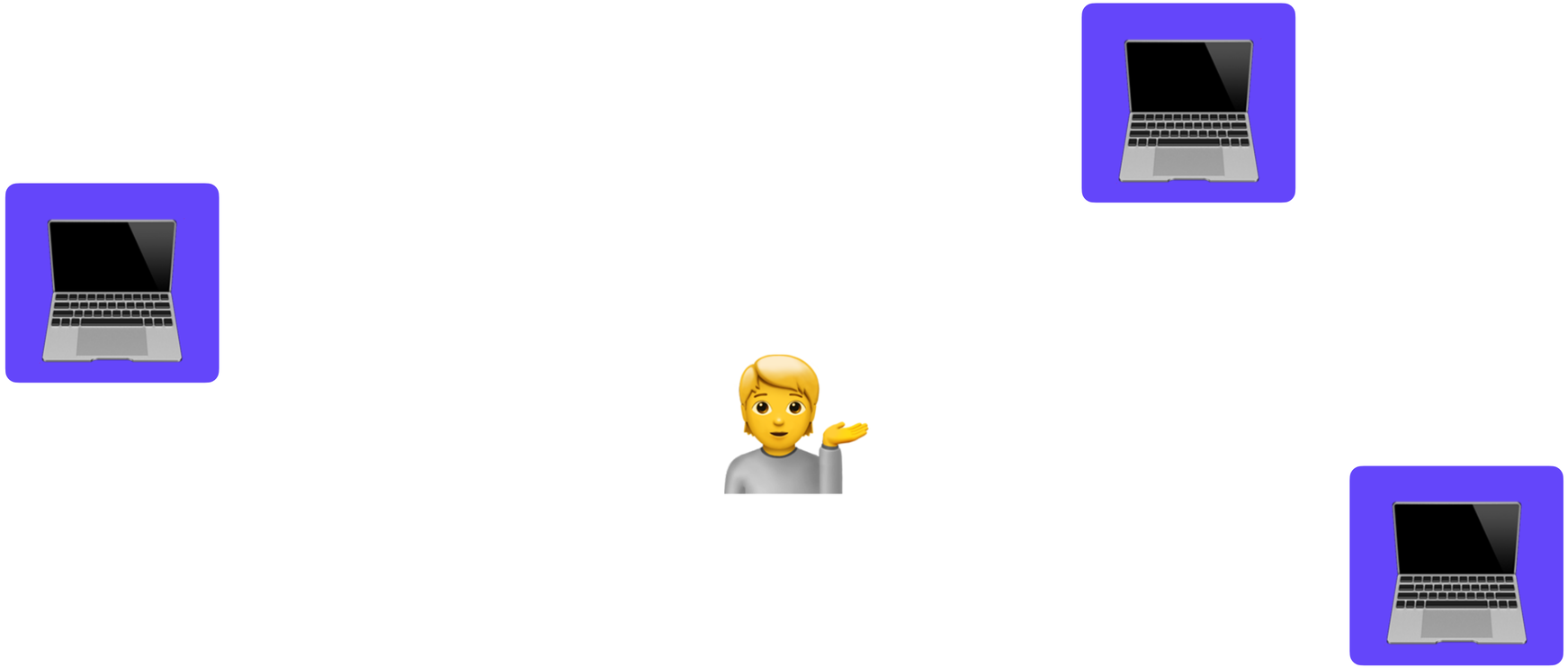
Content Addressed Data

P2P Discovery, Lookup, Transport



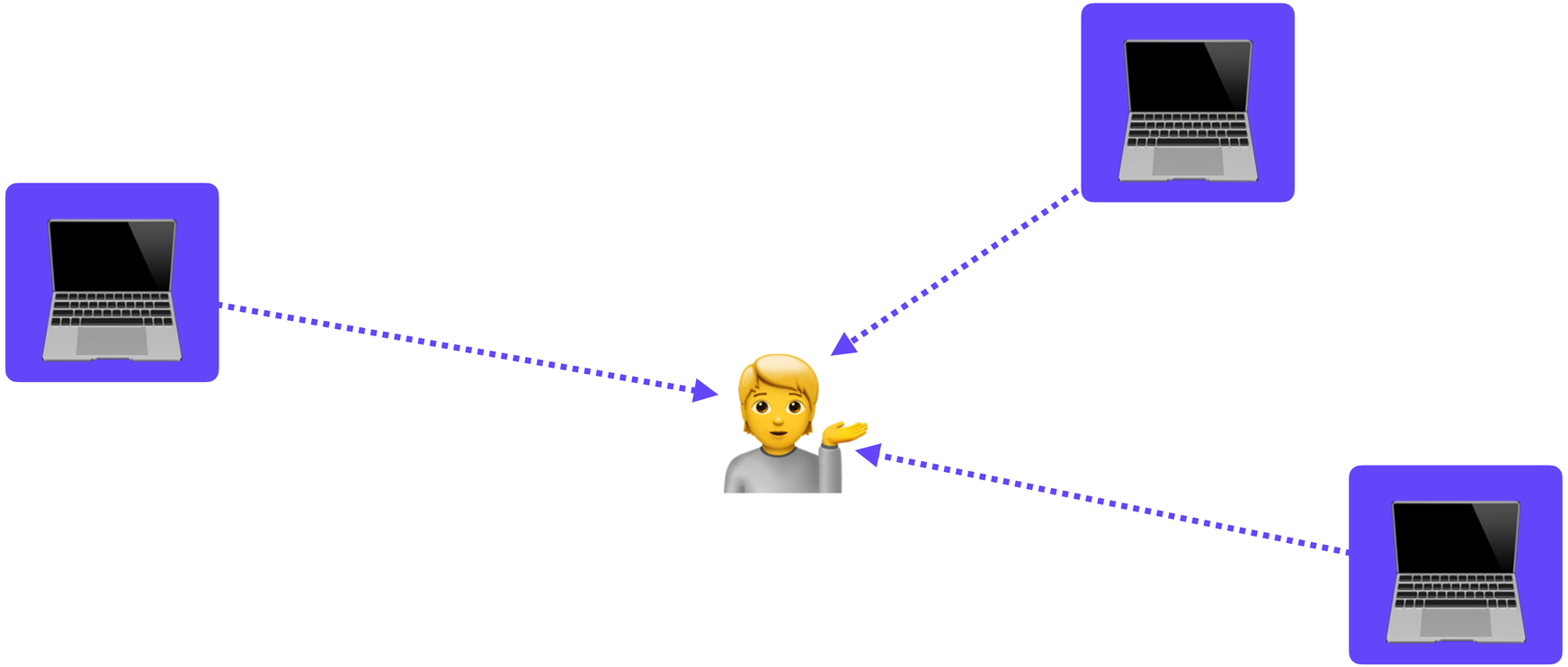
Content Addressed Data

P2P Discovery, Lookup, Transport



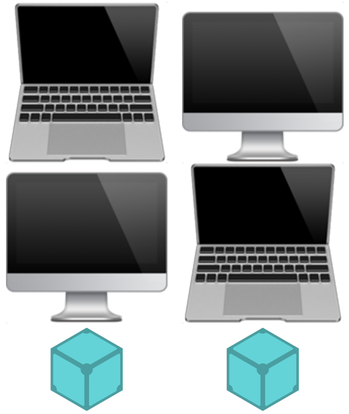
Content Addressed Data

P2P Discovery, Lookup, Transport

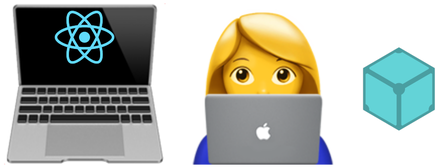


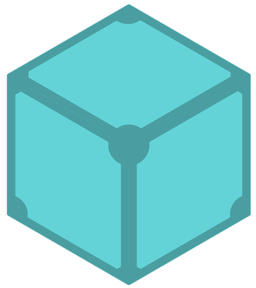
Content Addressed Data

Mutable Pointer Broadcast: DNSLink



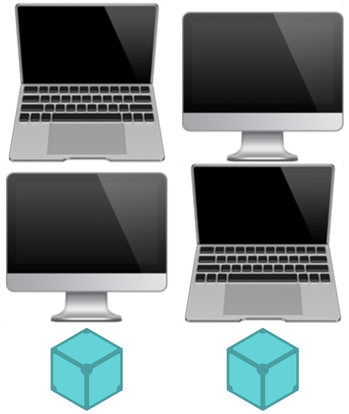
https://yourname.example.com



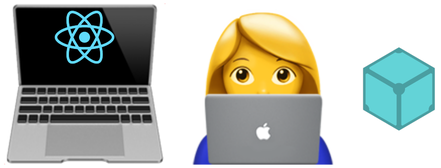


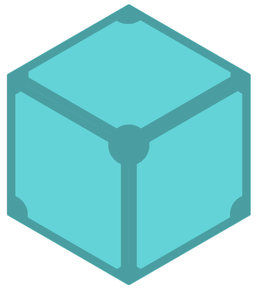
Content Addressed Data

Mutable Pointer Broadcast: DNSLink



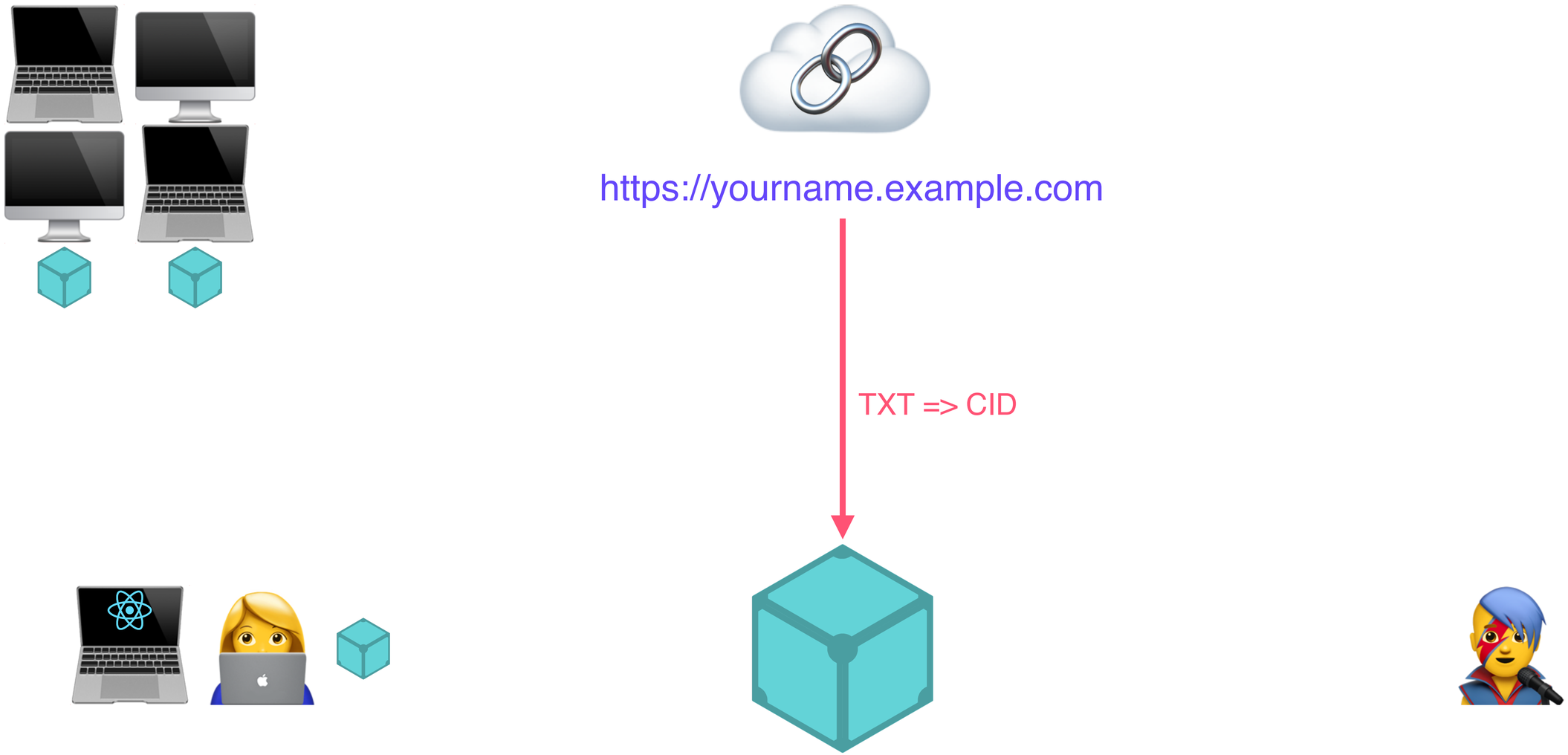
https://yourname.example.com





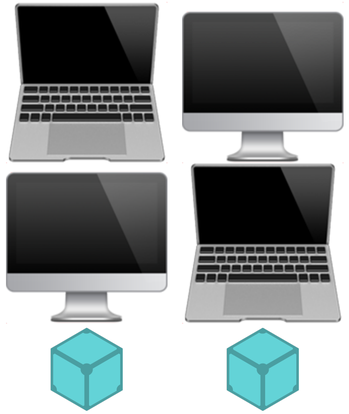
Content Addressed Data

Mutable Pointer Broadcast: DNSLink

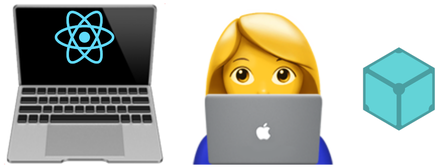


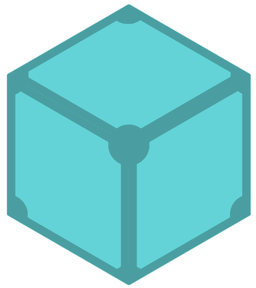
Content Addressed Data

Mutable Pointer Broadcast: DNSLink



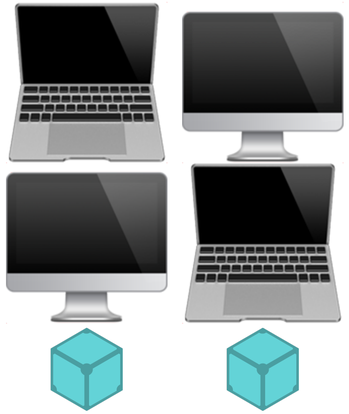
https://yourname.example.com



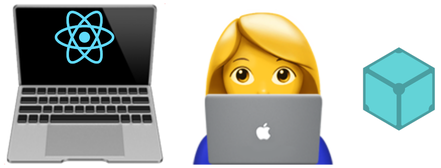


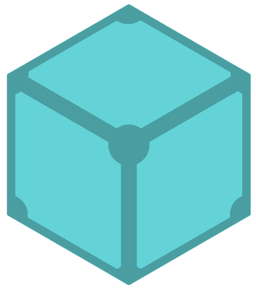
Content Addressed Data

Mutable Pointer Broadcast: DNSLink



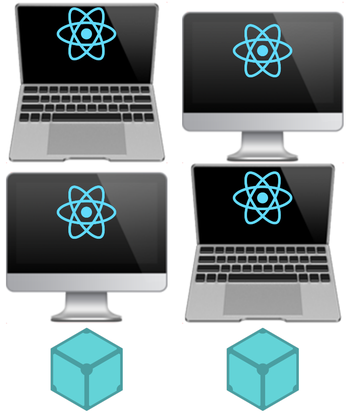
https://yourname.example.com



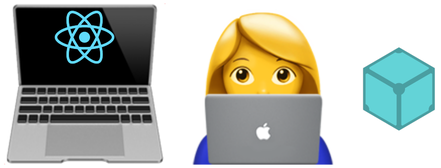


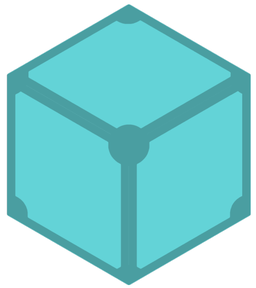
Content Addressed Data

Mutable Pointer Broadcast: DNSLink



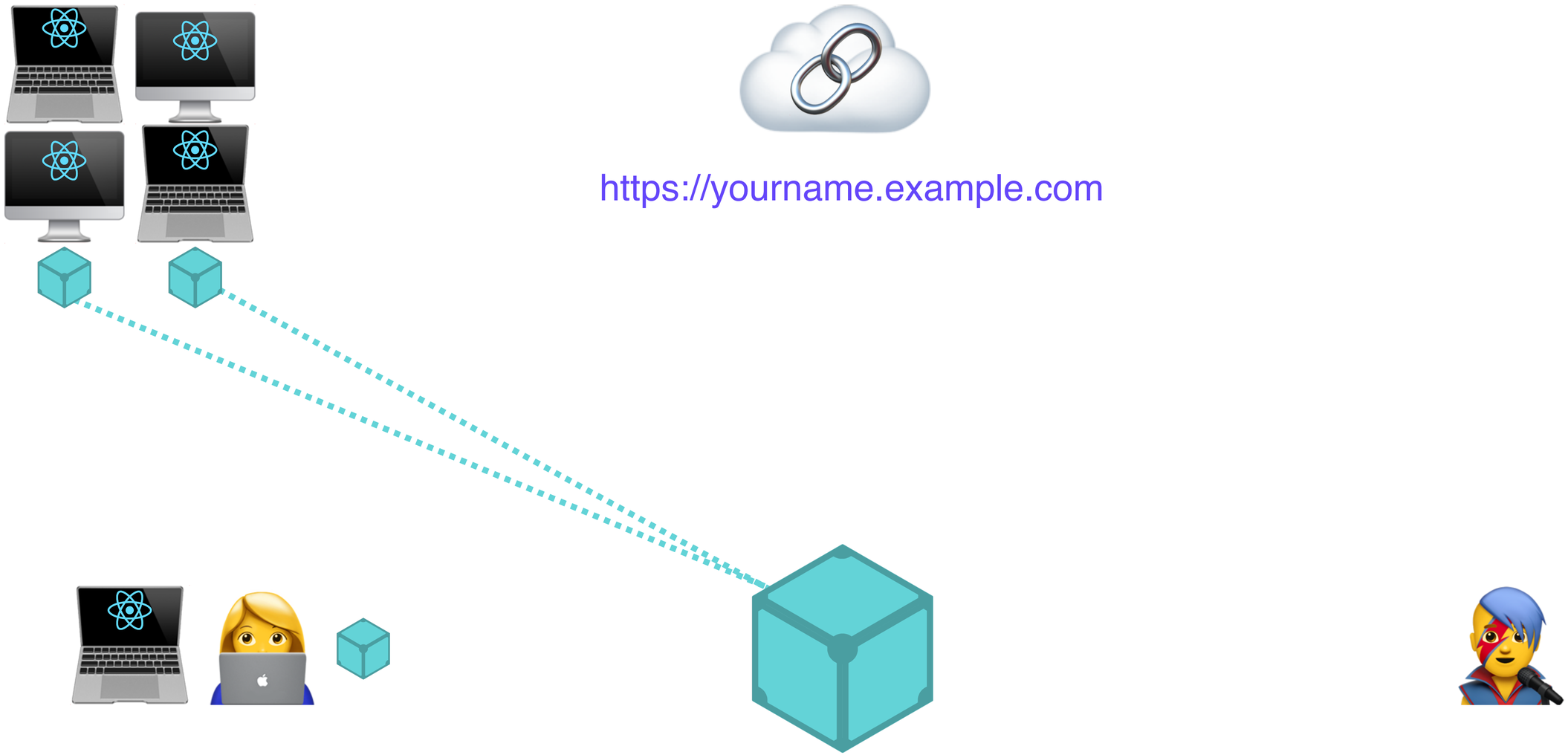
https://yourname.example.com





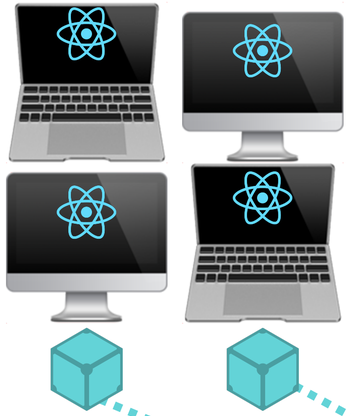
Content Addressed Data

Mutable Pointer Broadcast: DNSLink

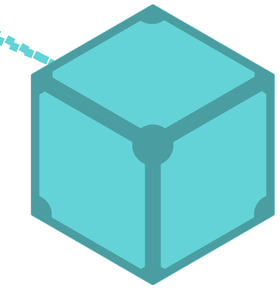


Content Addressed Data

Mutable Pointer Broadcast: DNSLink

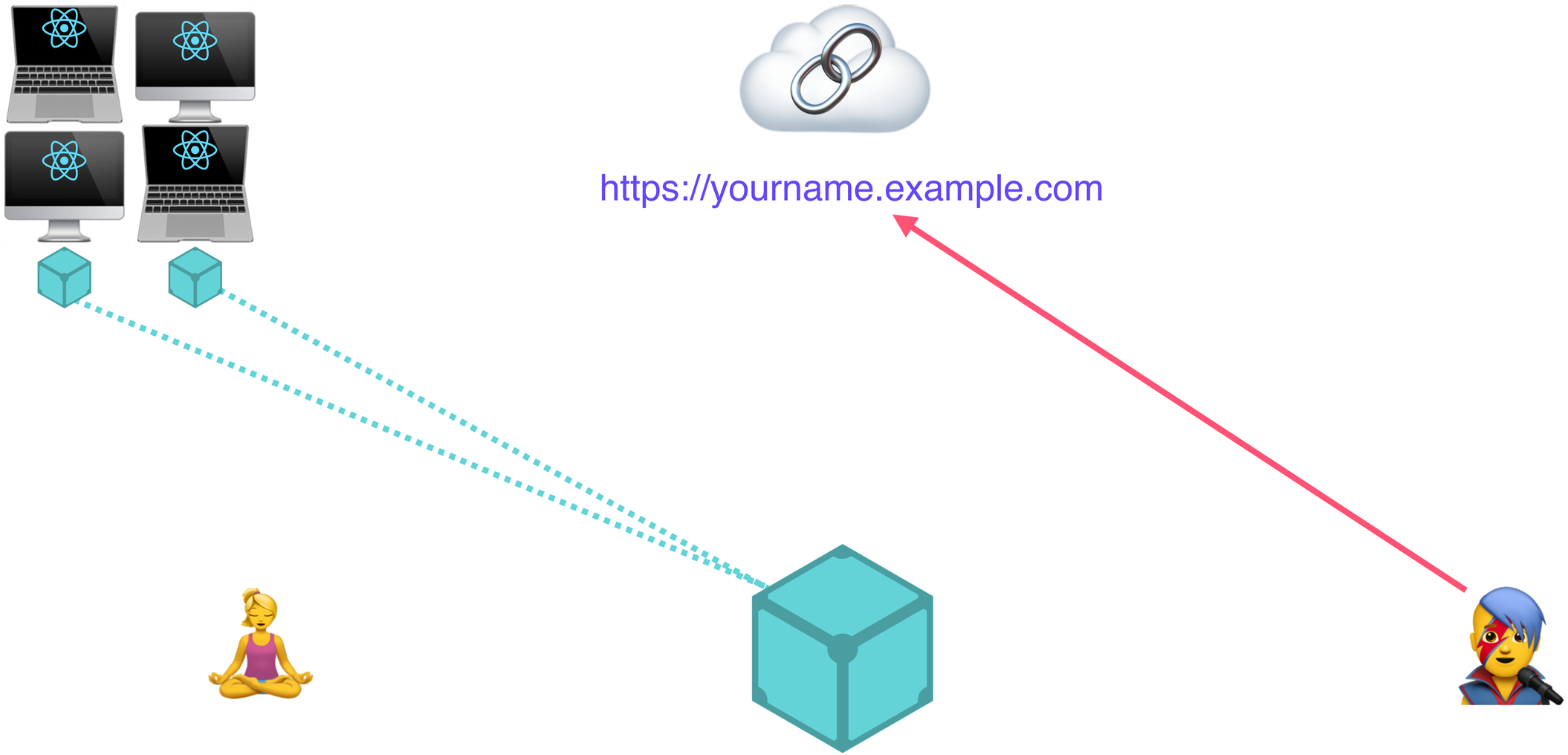


https://yourname.example.com



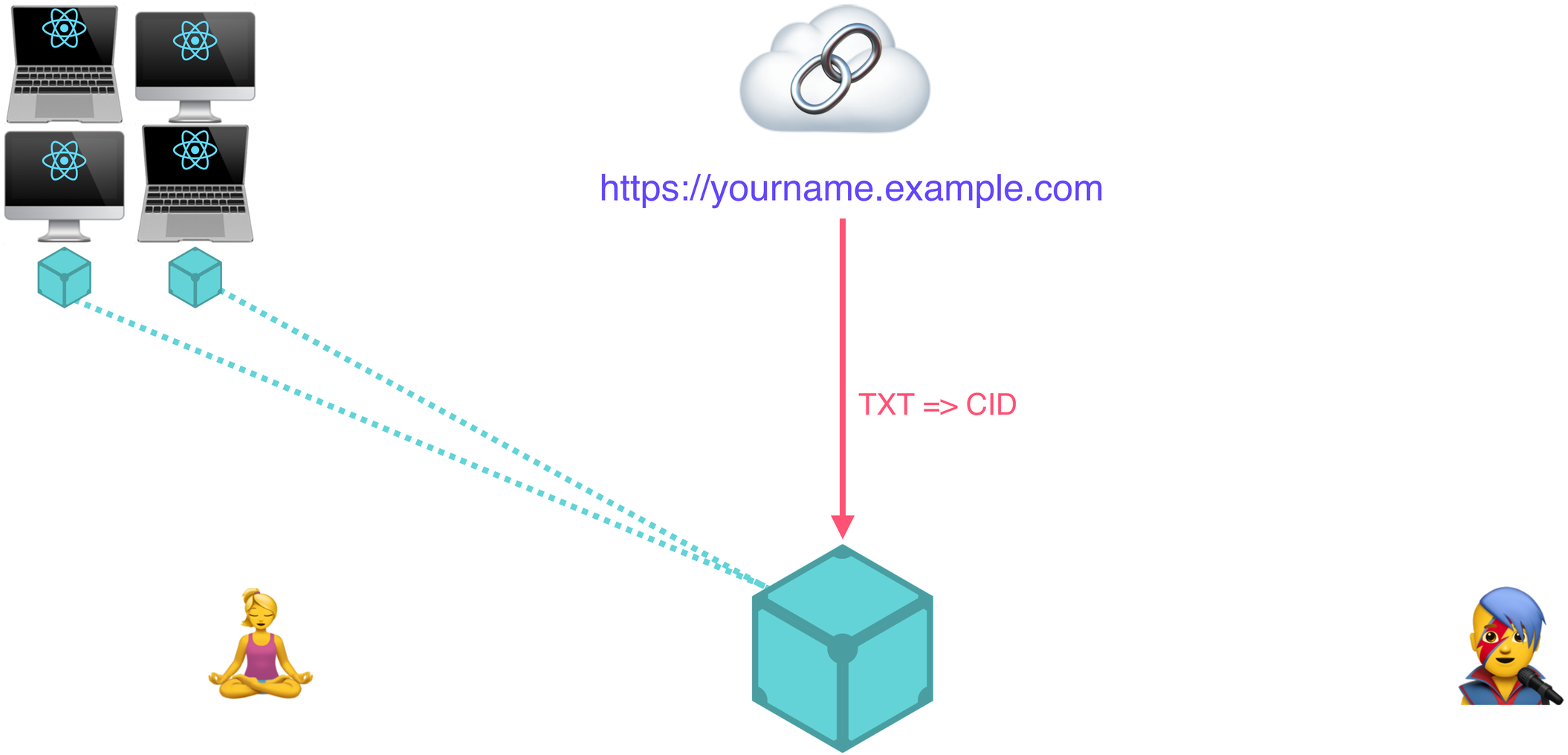
Content Addressed Data

Mutable Pointer Broadcast: DNSLink



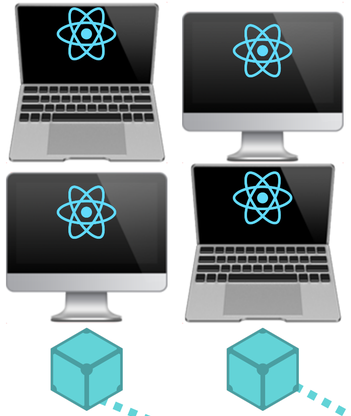
Content Addressed Data

Mutable Pointer Broadcast: DNSLink

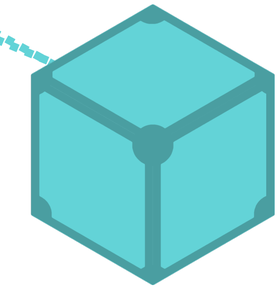


Content Addressed Data

Mutable Pointer Broadcast: DNSLink



https://yourname.example.com



Content Addressed Data

So we have a universal namespace.

So we have a universal namespace.

So we have a universal namespace.

Great!

So we have a universal namespace.

Great!

So we have a universal namespace.

Great!

Well that seems pretty insecure...

Securing Data Access

Fixing the Leaky Pipes

Securing Data Access

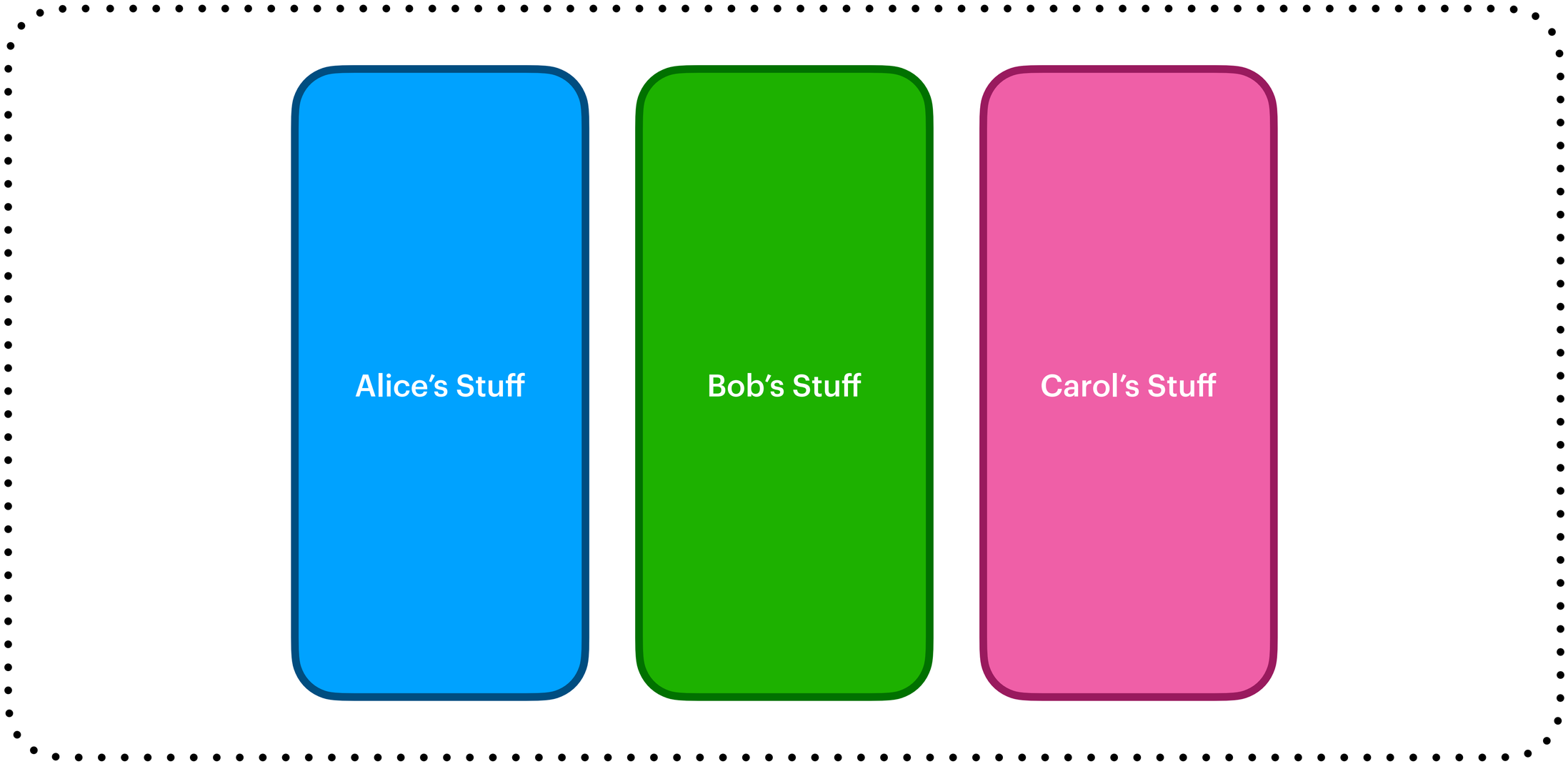
Grouped by User, Not by App

Securing Data Access

Grouped by User, Not by App

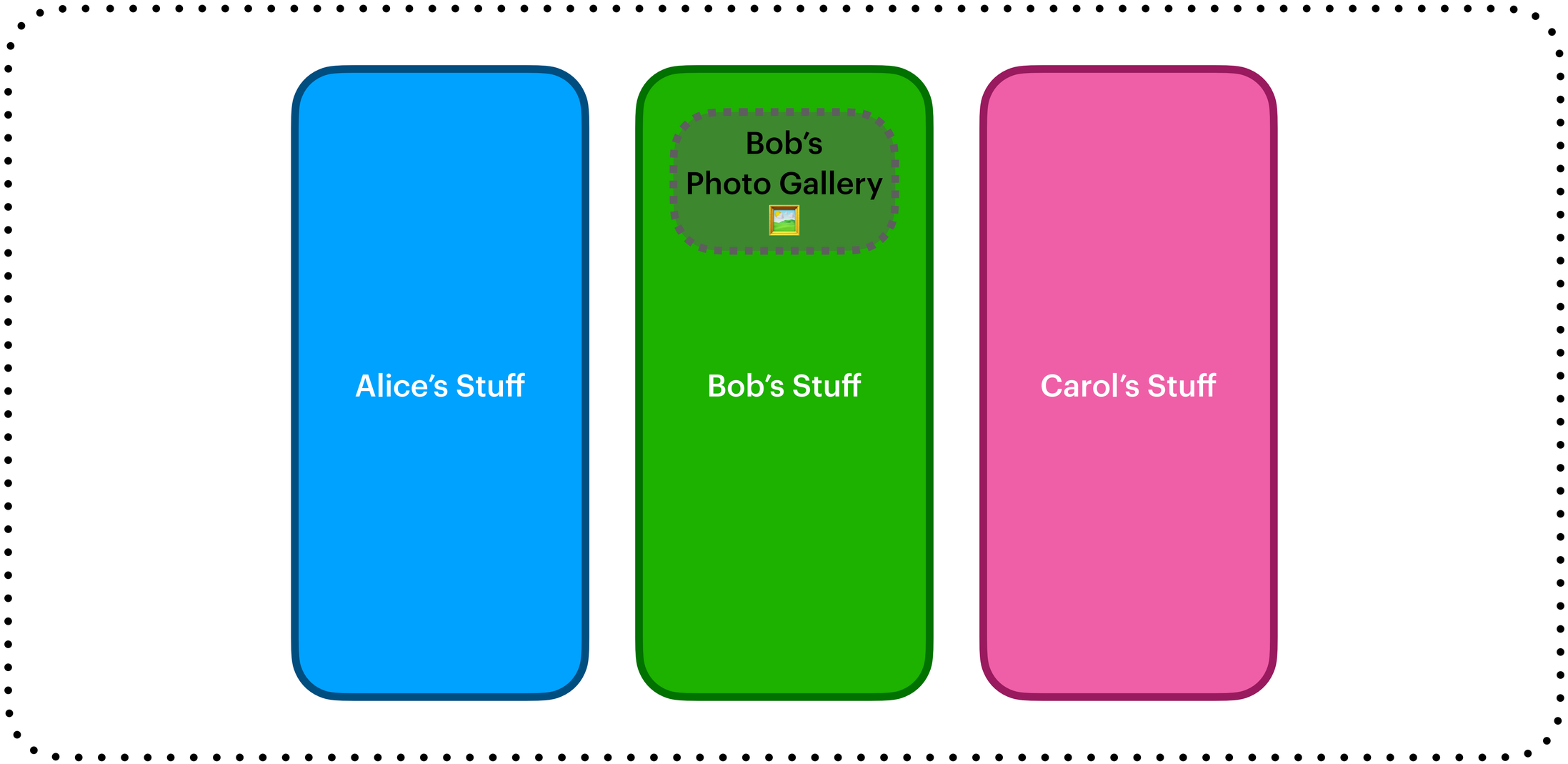
Securing Data Access

Grouped by User, Not by App



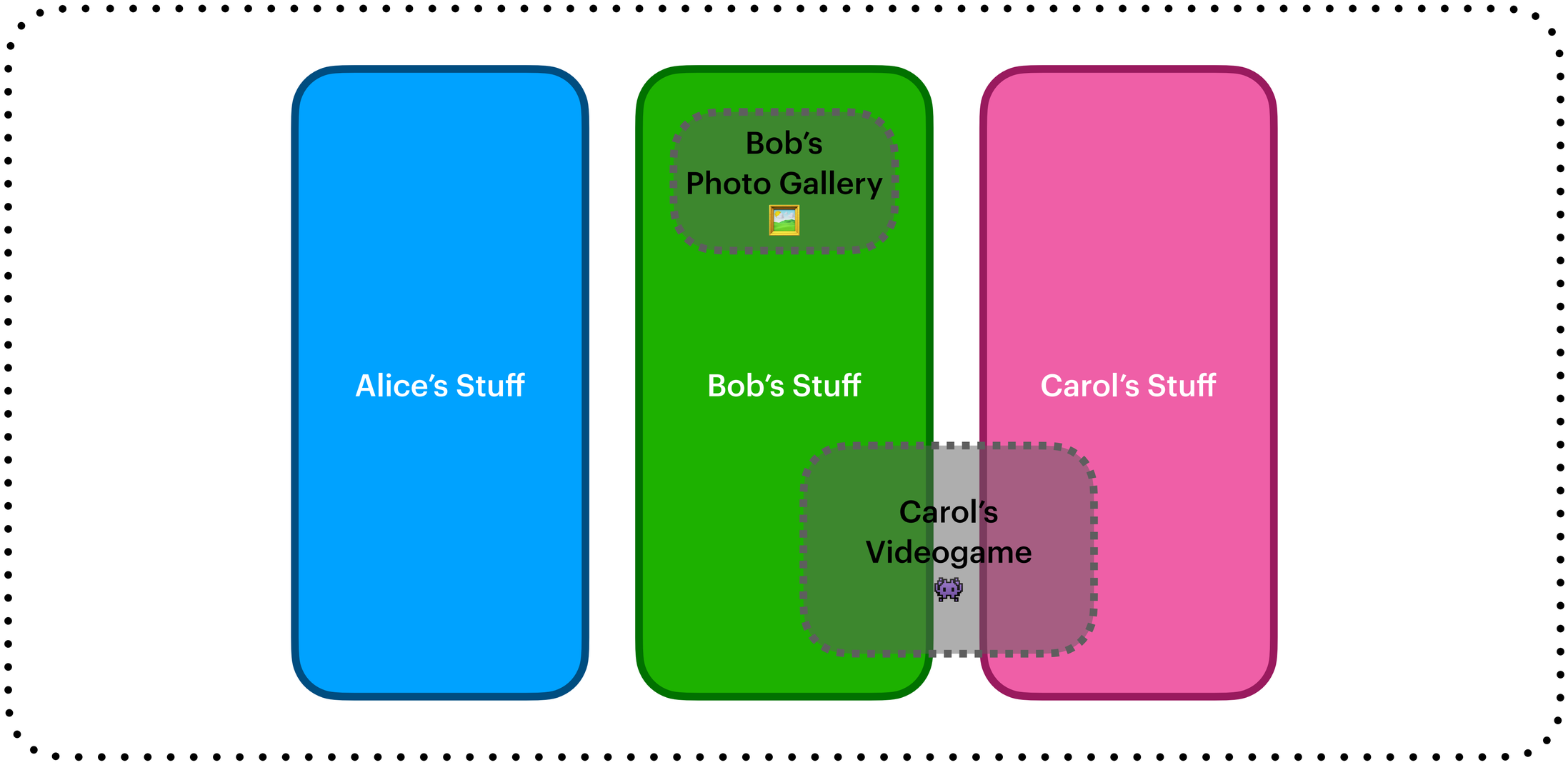
Securing Data Access

Grouped by User, Not by App



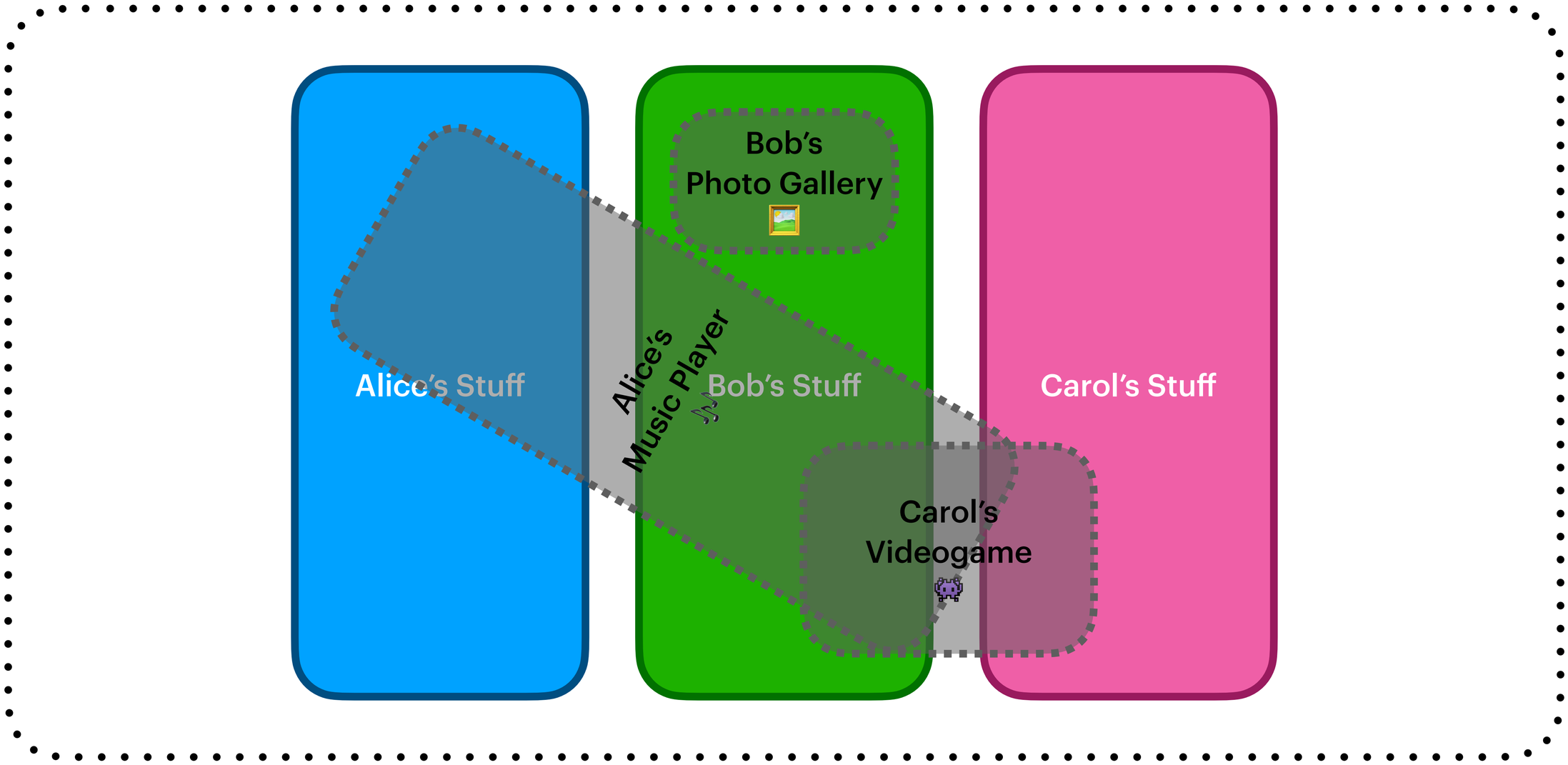
Securing Data Access

Grouped by User, Not by App



Securing Data Access

Grouped by User, Not by App



Securing Data Access

WNFS Layout

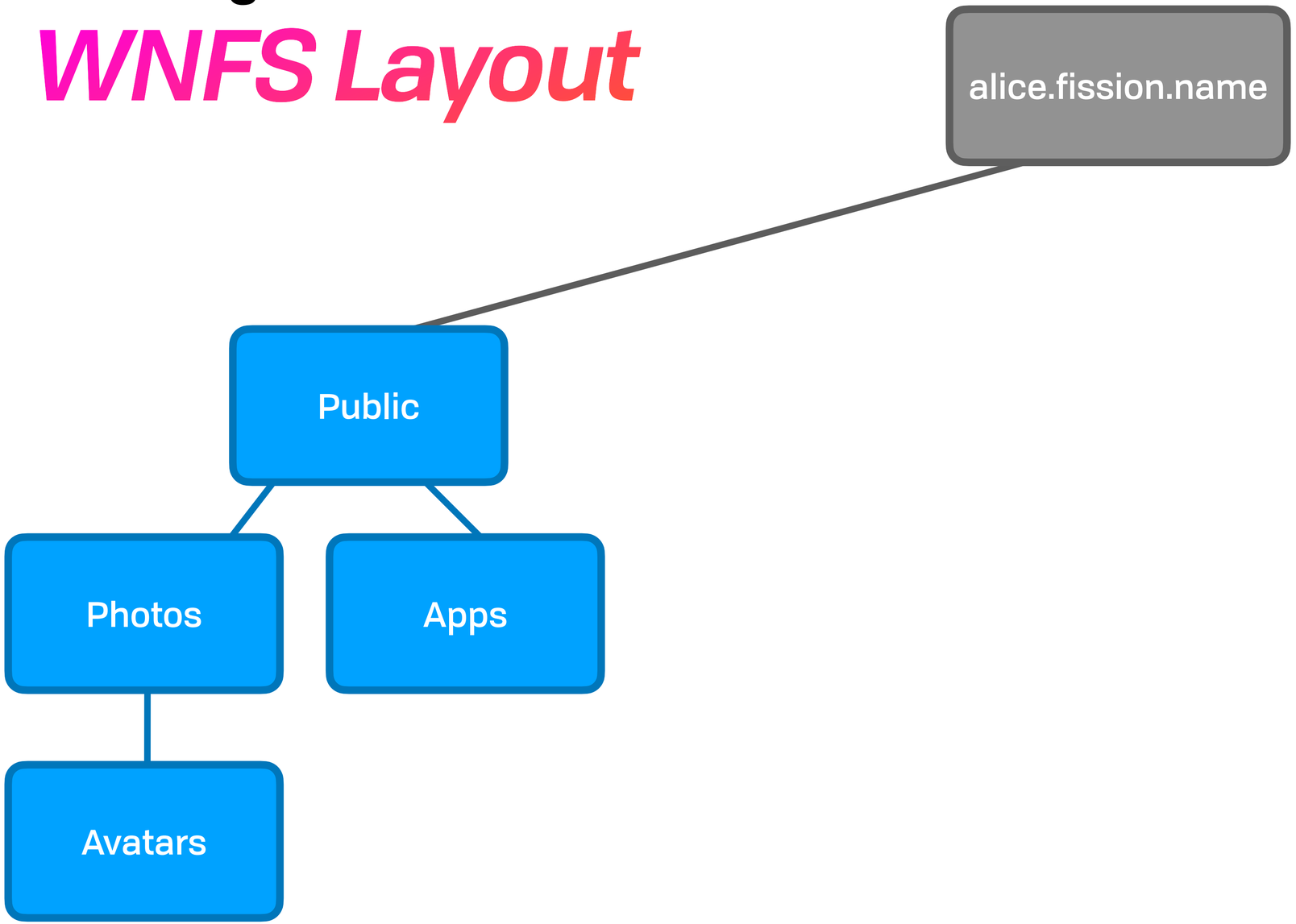
Securing Data Access

WNFS Layout

alice.fission.name

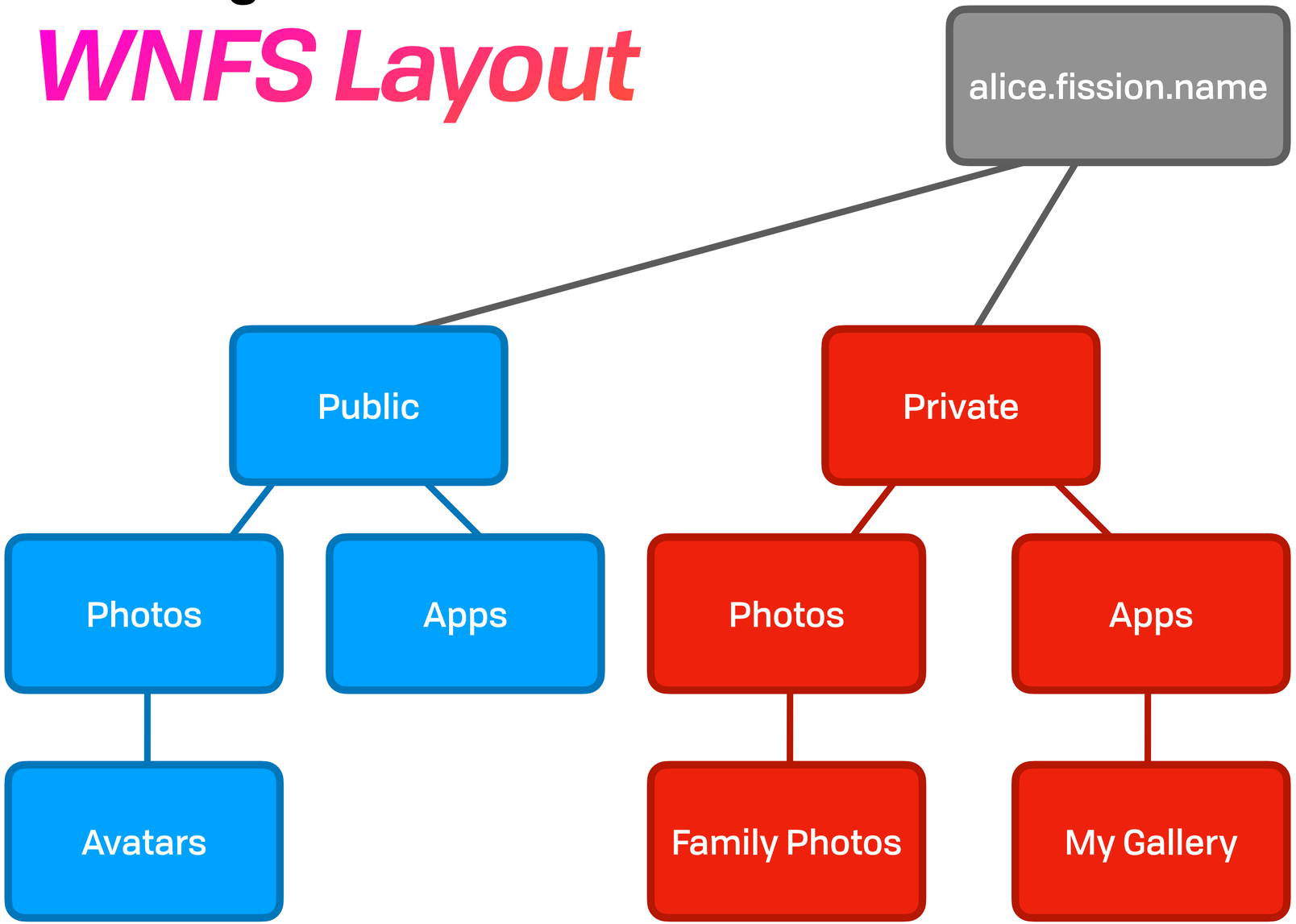
Securing Data Access

WNFS Layout



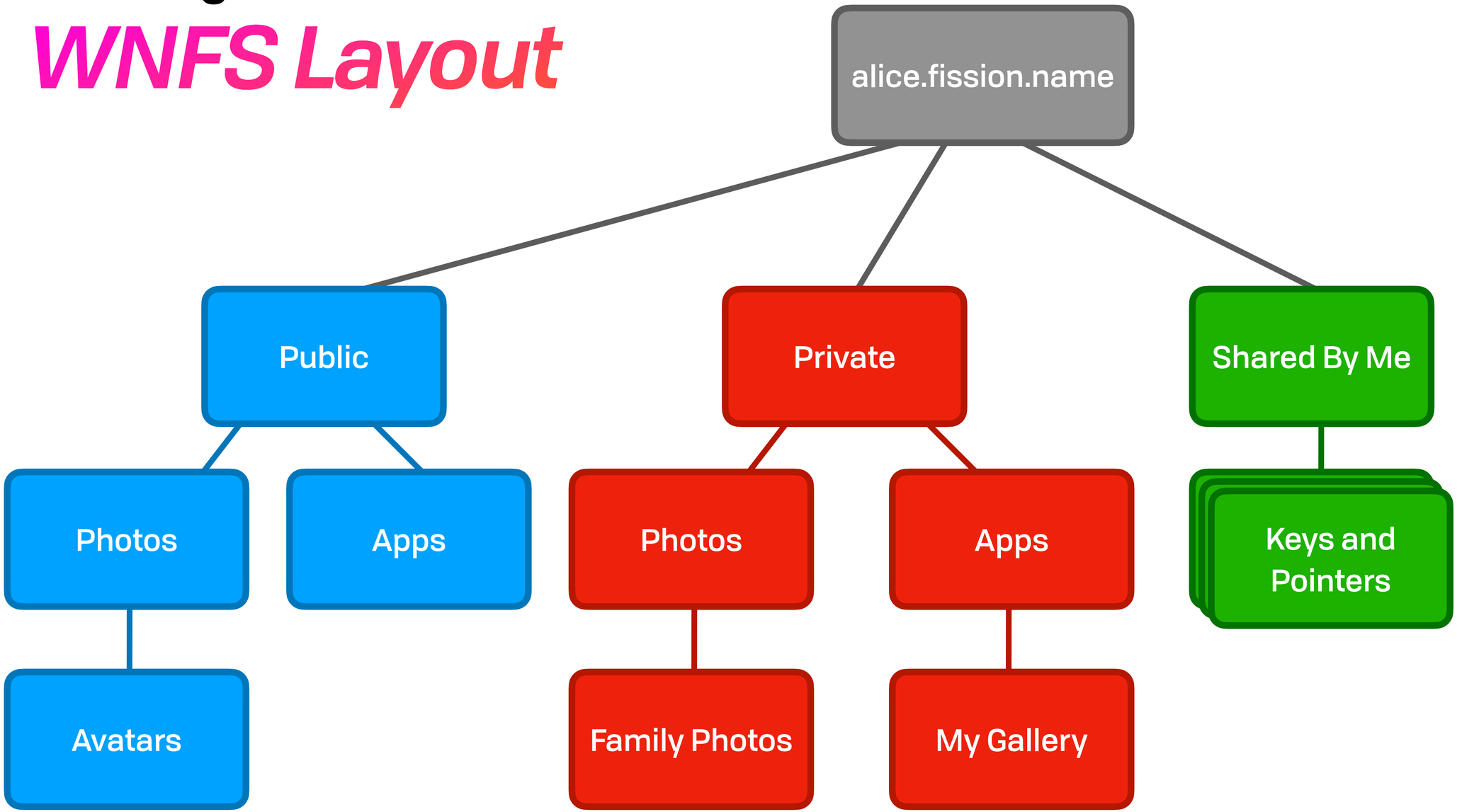
Securing Data Access

WNFS Layout



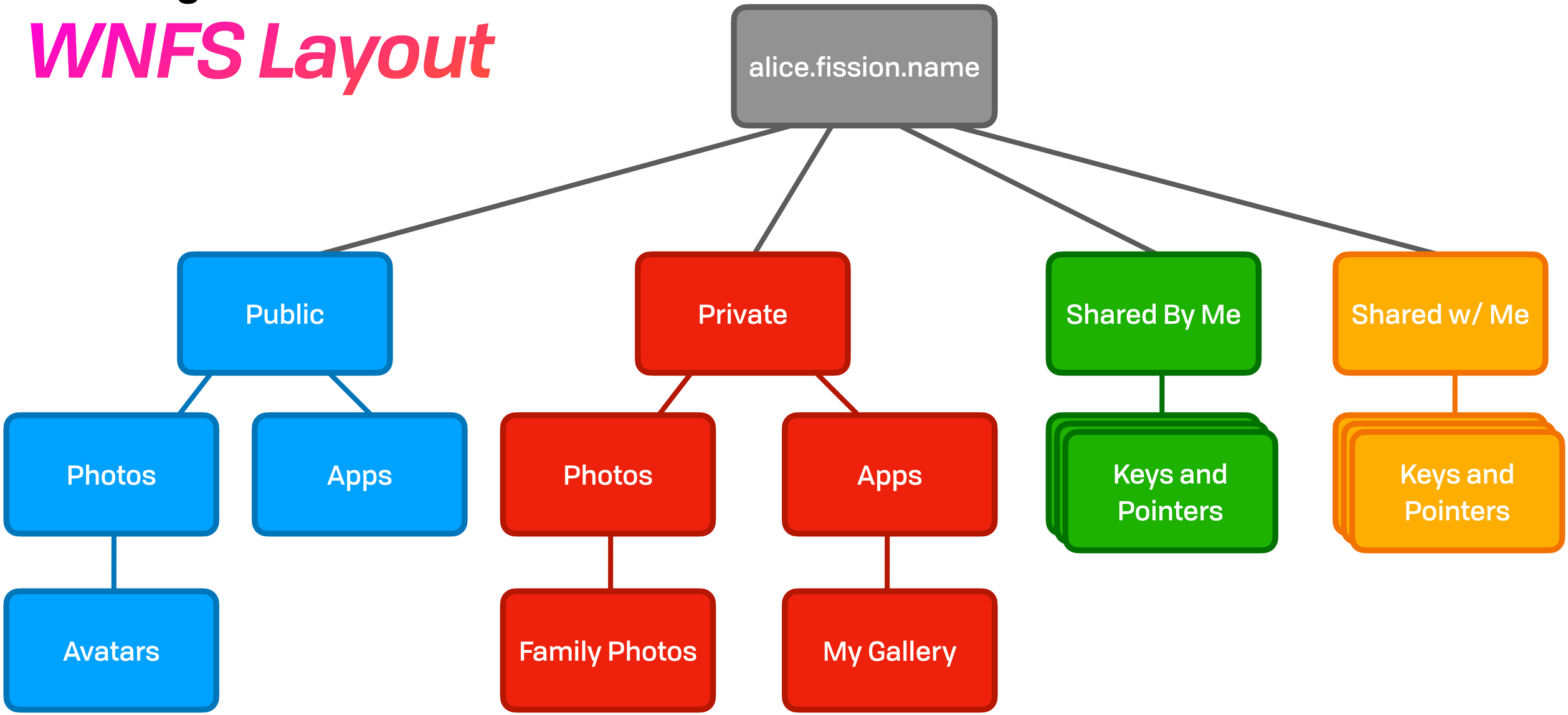
Securing Data Access

WNFS Layout



Securing Data Access

WNFS Layout

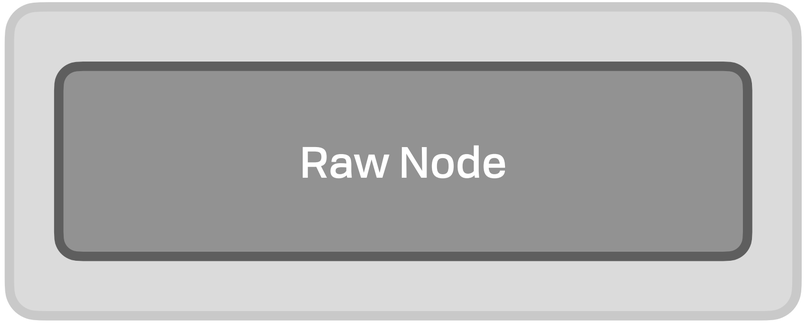


Securing Data Access

Virtual Nodes

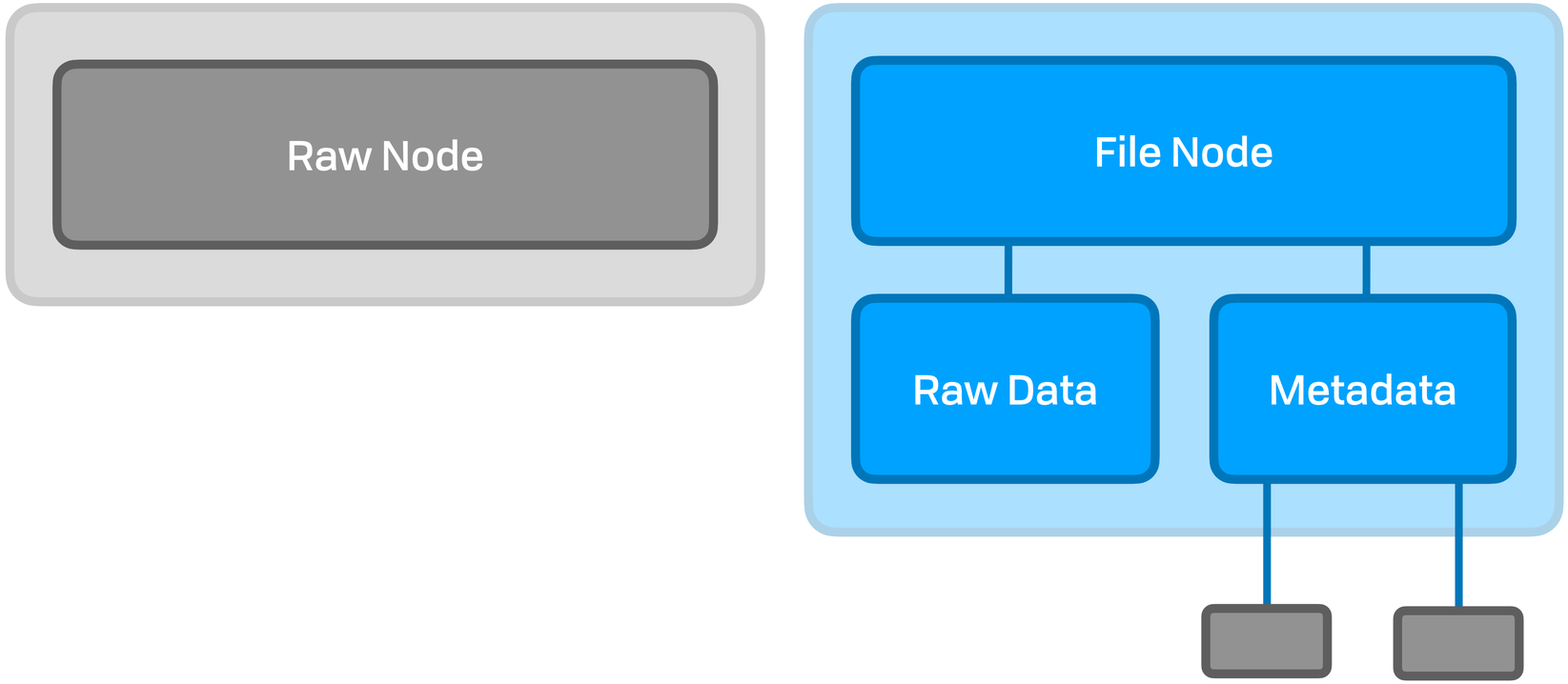
Securing Data Access

Virtual Nodes



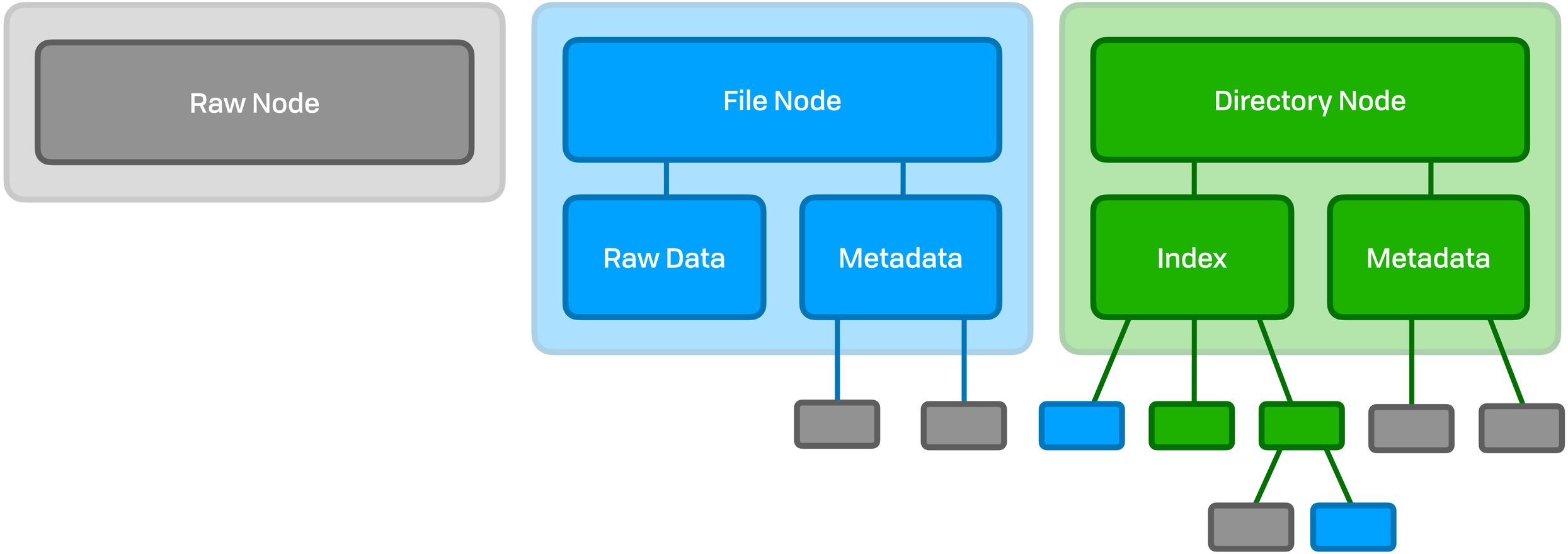
Securing Data Access

Virtual Nodes



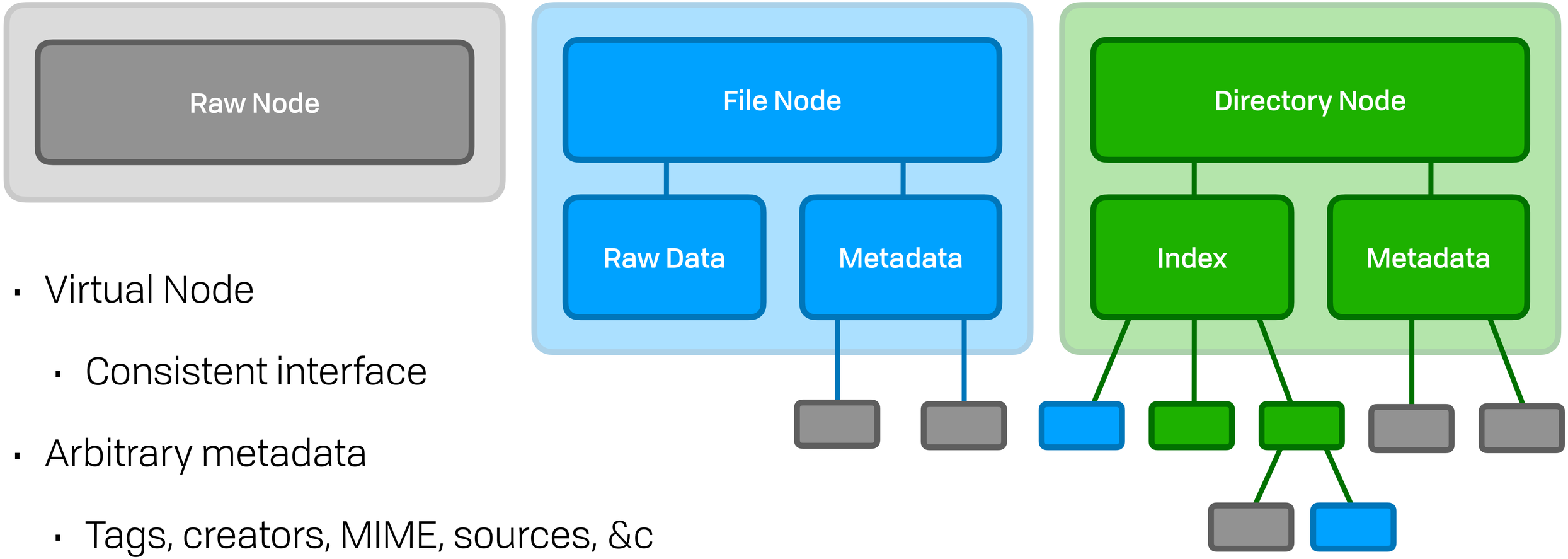
Securing Data Access

Virtual Nodes



Securing Data Access

Virtual Nodes



- Virtual Node
  - Consistent interface
- Arbitrary metadata
  - Tags, creators, MIME, sources, &c

Securing Data Access

Hard & Soft Links

Securing Data Access

Hard & Soft Links

- Hard links
  - New for the web!
  - Direct reference
  - 2 pointers ~ duplicate

Securing Data Access

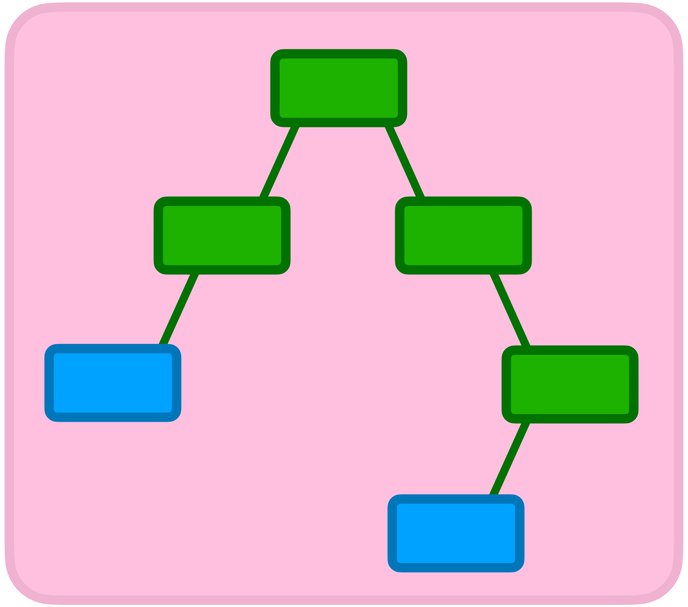
Hard & Soft Links

- Hard links
  - New for the web!
  - Direct reference
  - 2 pointers ~ duplicate
- Soft links
  - Like a symlink or web link
  - 2 pointers ~ latest
  - May break
    - Always some version available

Securing Data Access

Hard & Soft Links

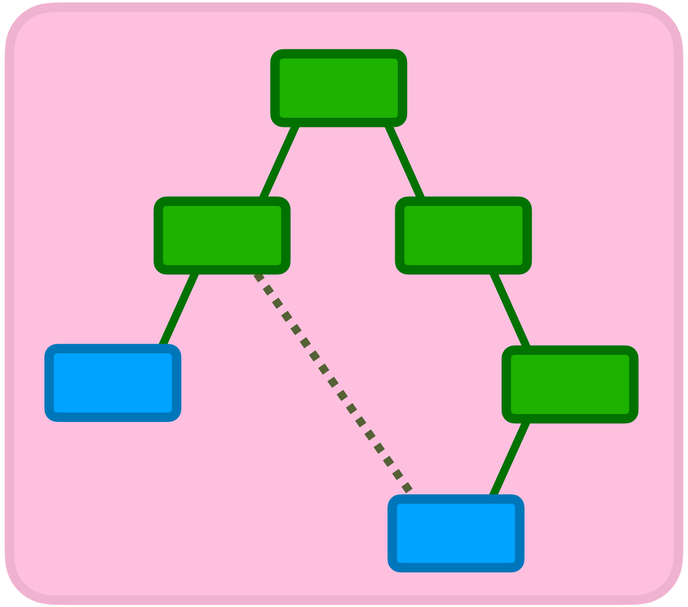
- Hard links
  - New for the web!
  - Direct reference
  - 2 pointers ~ duplicate
- Soft links
  - Like a symlink or web link
  - 2 pointers ~ latest
  - May break
    - Always some version available



Securing Data Access

Hard & Soft Links

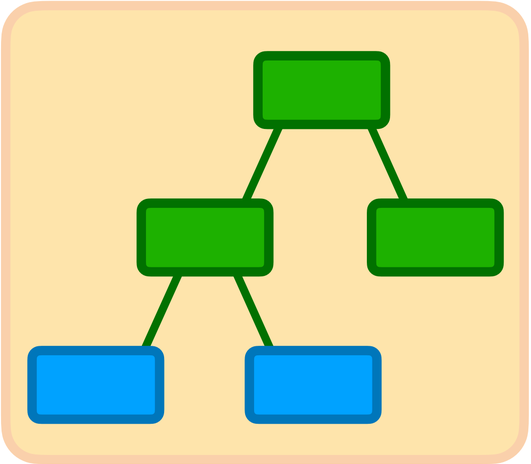
- Hard links
  - New for the web!
  - Direct reference
  - 2 pointers ~ duplicate
- Soft links
  - Like a symlink or web link
  - 2 pointers ~ latest
  - May break
    - Always some version available

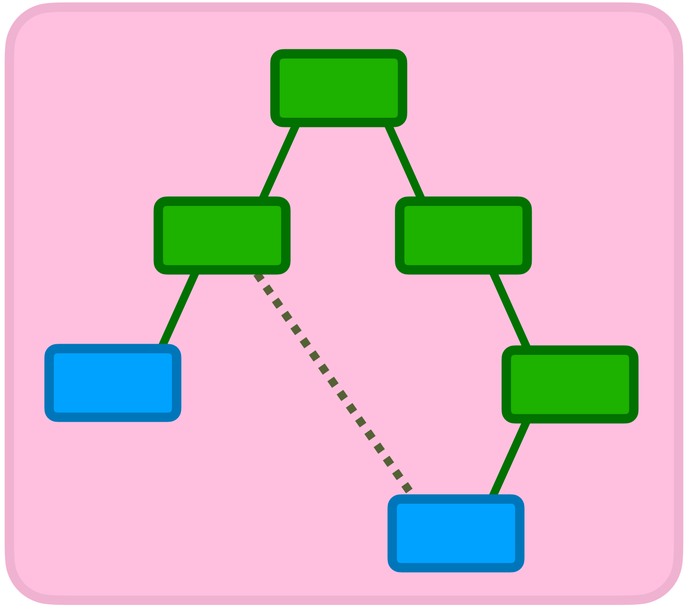


Securing Data Access

Hard & Soft Links

- Hard links
  - New for the web!
  - Direct reference
  - 2 pointers ~ duplicate
- Soft links
  - Like a symlink or web link
  - 2 pointers ~ latest
  - May break
    - Always some version available

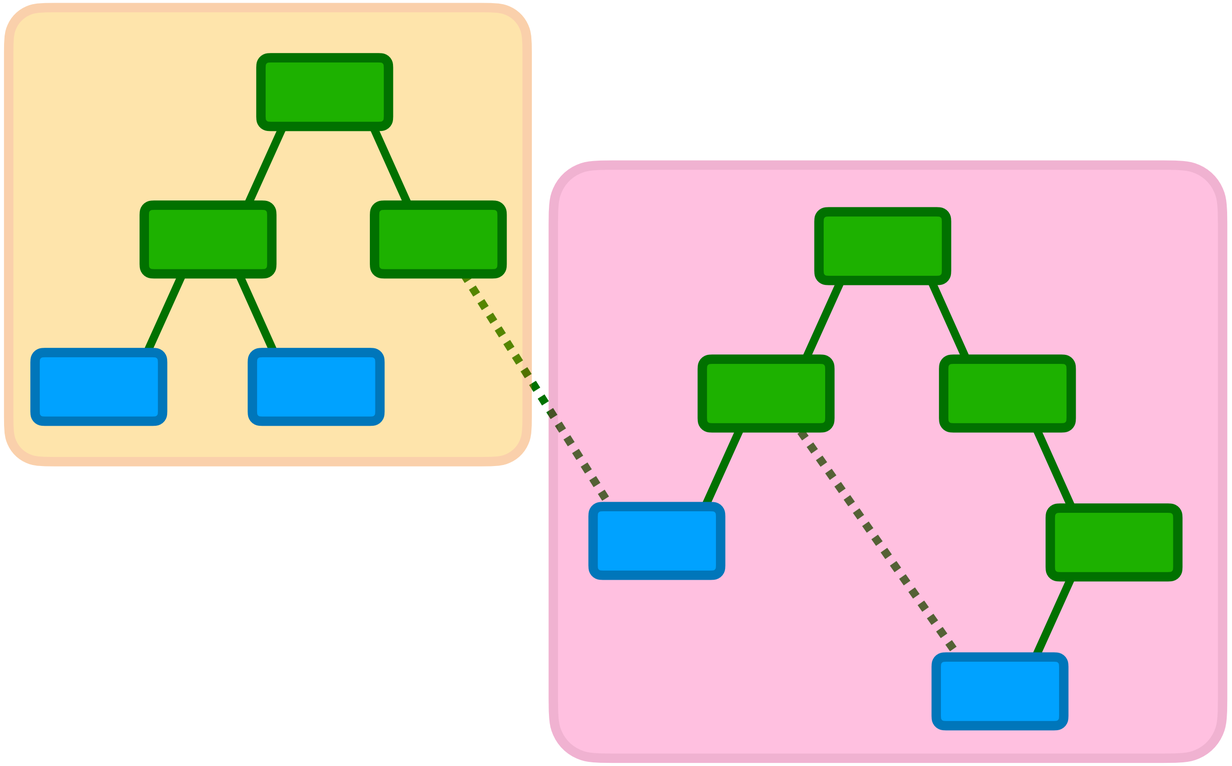




Securing Data Access

Hard & Soft Links

- Hard links
  - New for the web!
  - Direct reference
  - 2 pointers ~ duplicate
- Soft links
  - Like a symlink or web link
  - 2 pointers ~ latest
  - May break
    - Always some version available

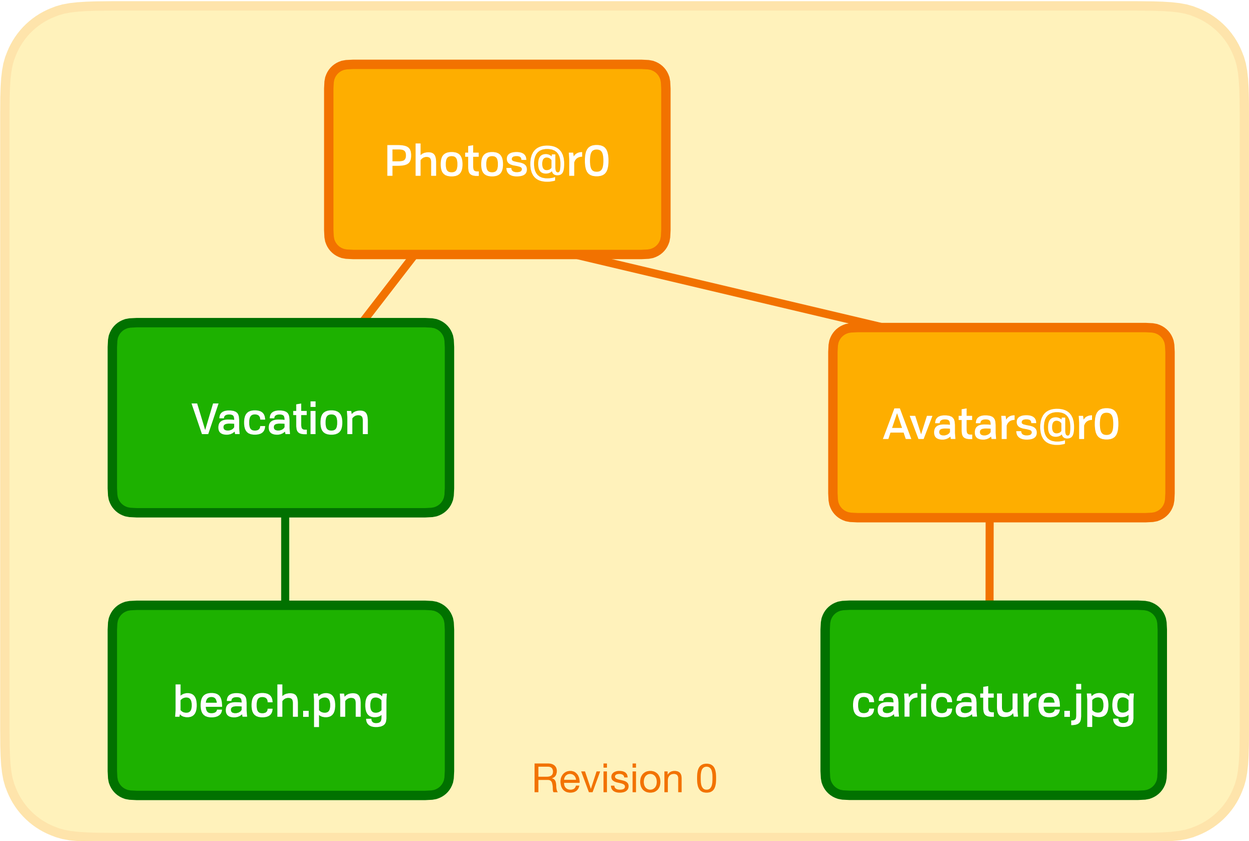


Securing Data Access

Persistent Versioning & Events

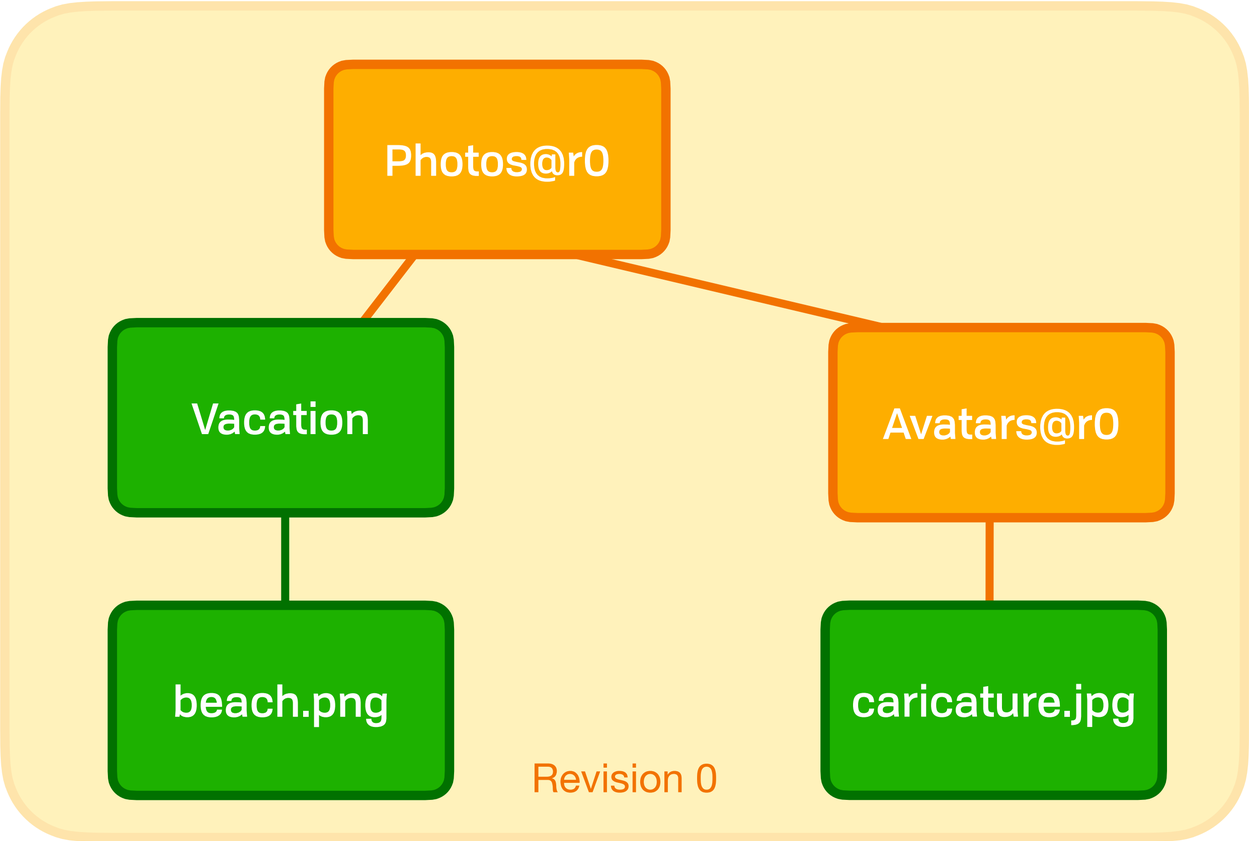
Securing Data Access

Persistent Versioning & Events



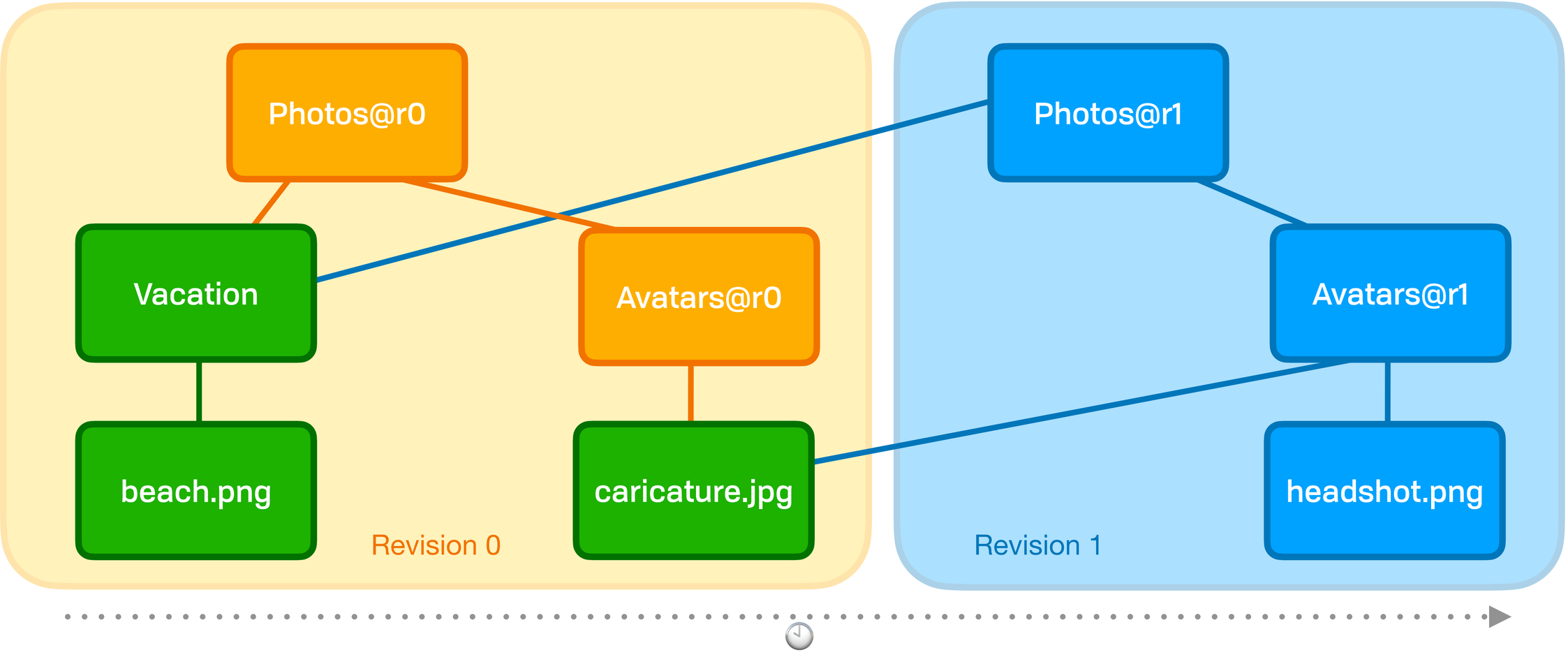
Securing Data Access

Persistent Versioning & Events



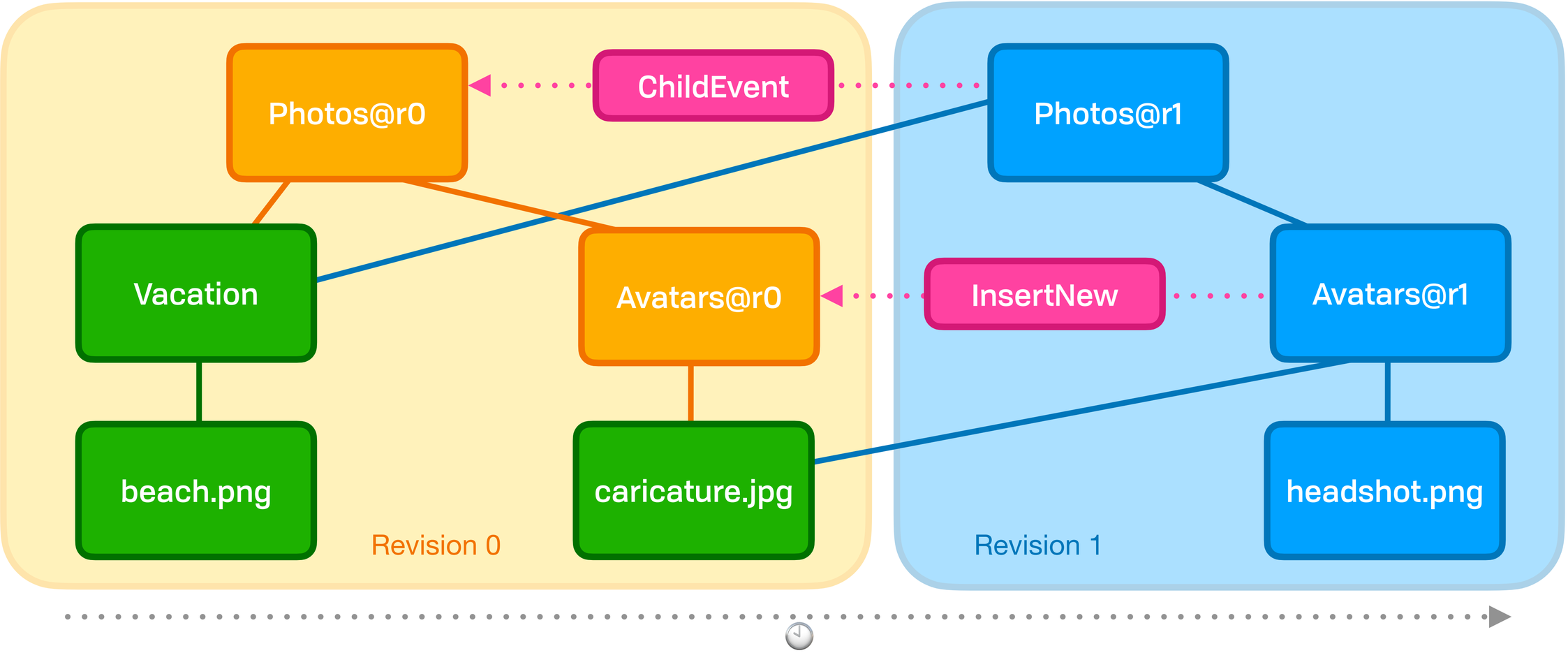
Securing Data Access

Persistent Versioning & Events



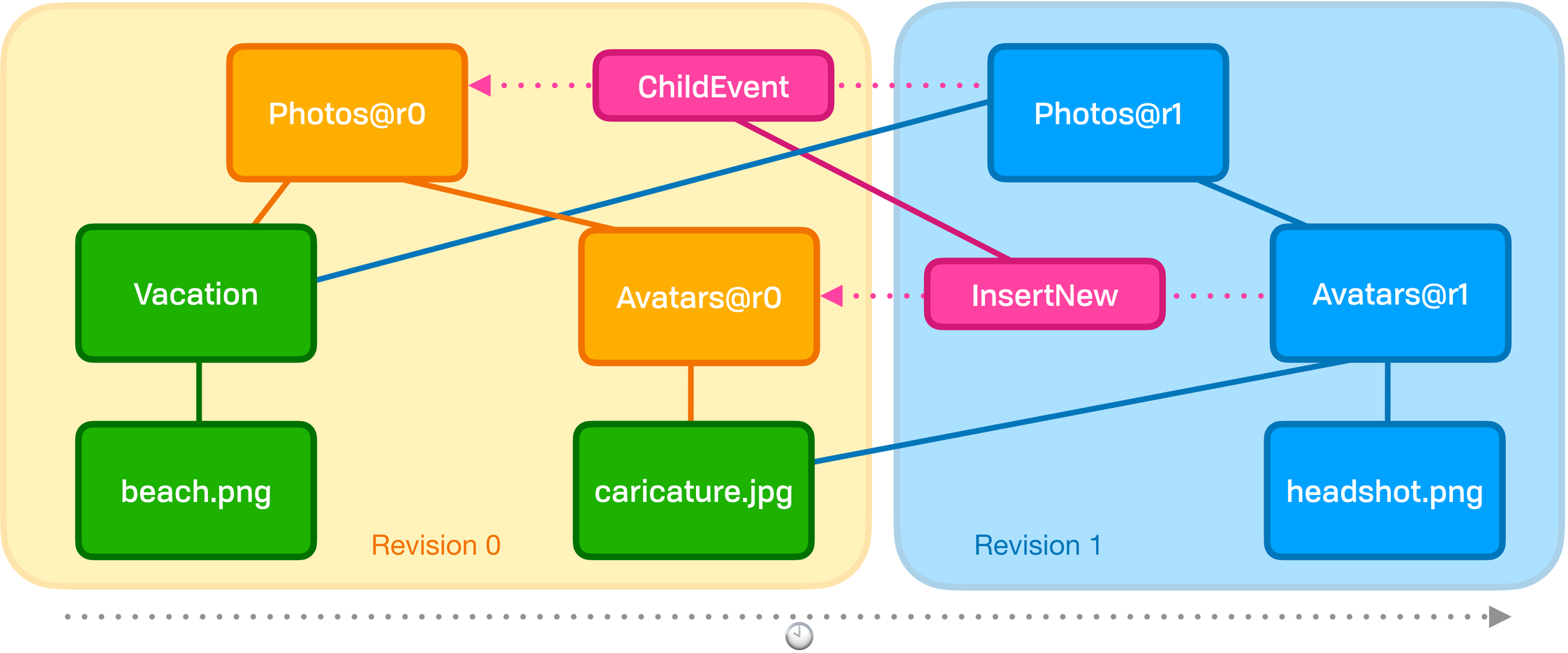
Securing Data Access

Persistent Versioning & Events



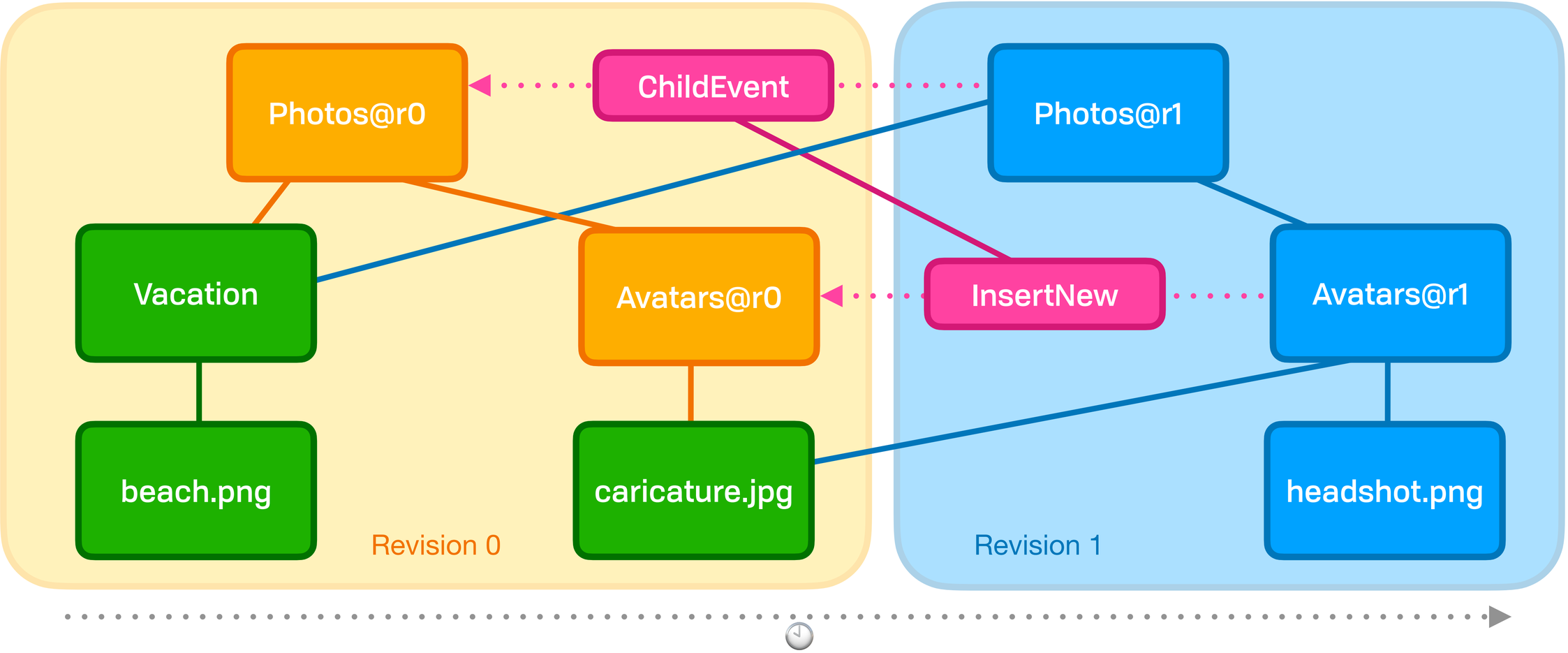
Securing Data Access

Persistent Versioning & Events



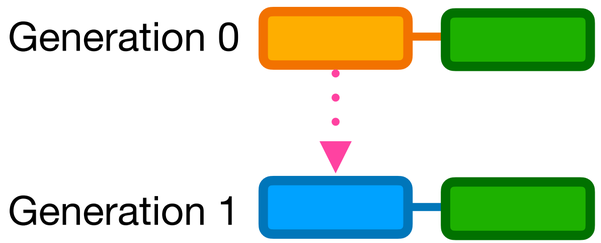
Securing Data Access

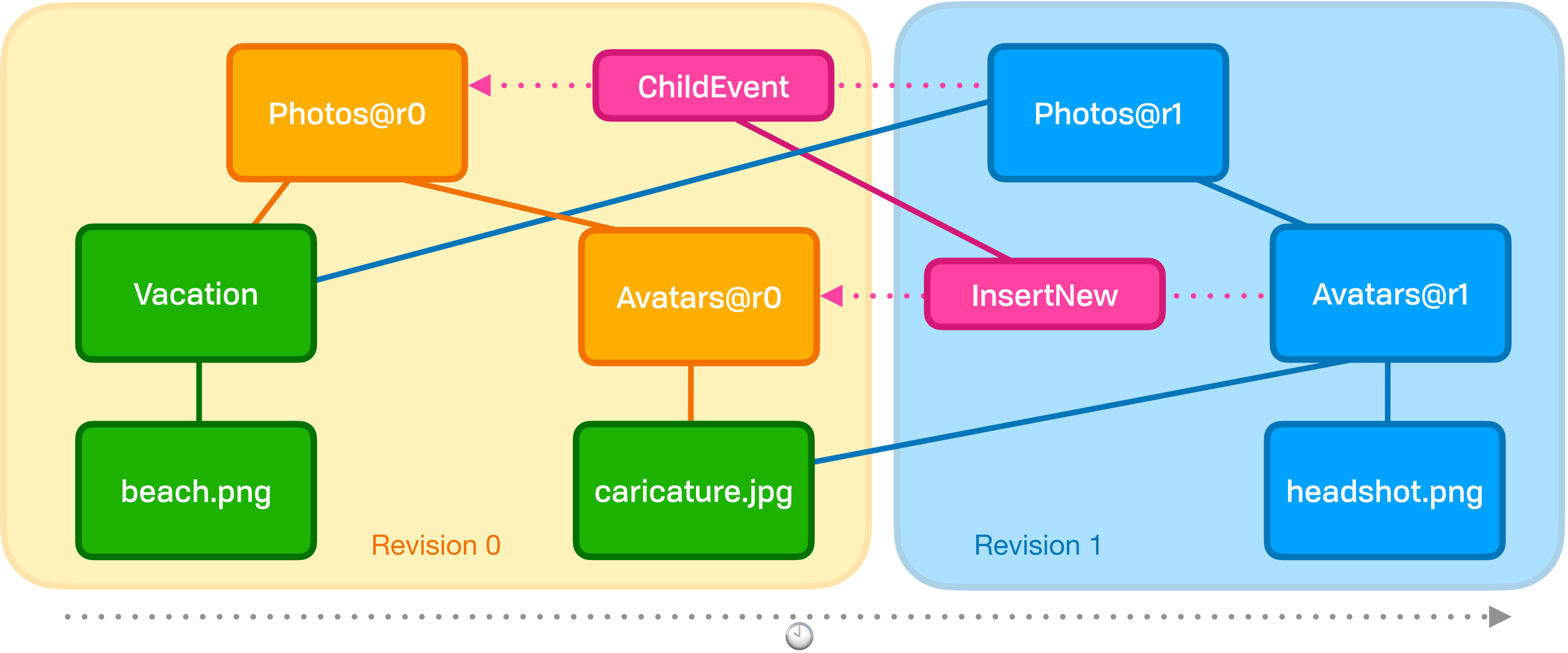
Persistent Versioning & Events



Securing Data Access

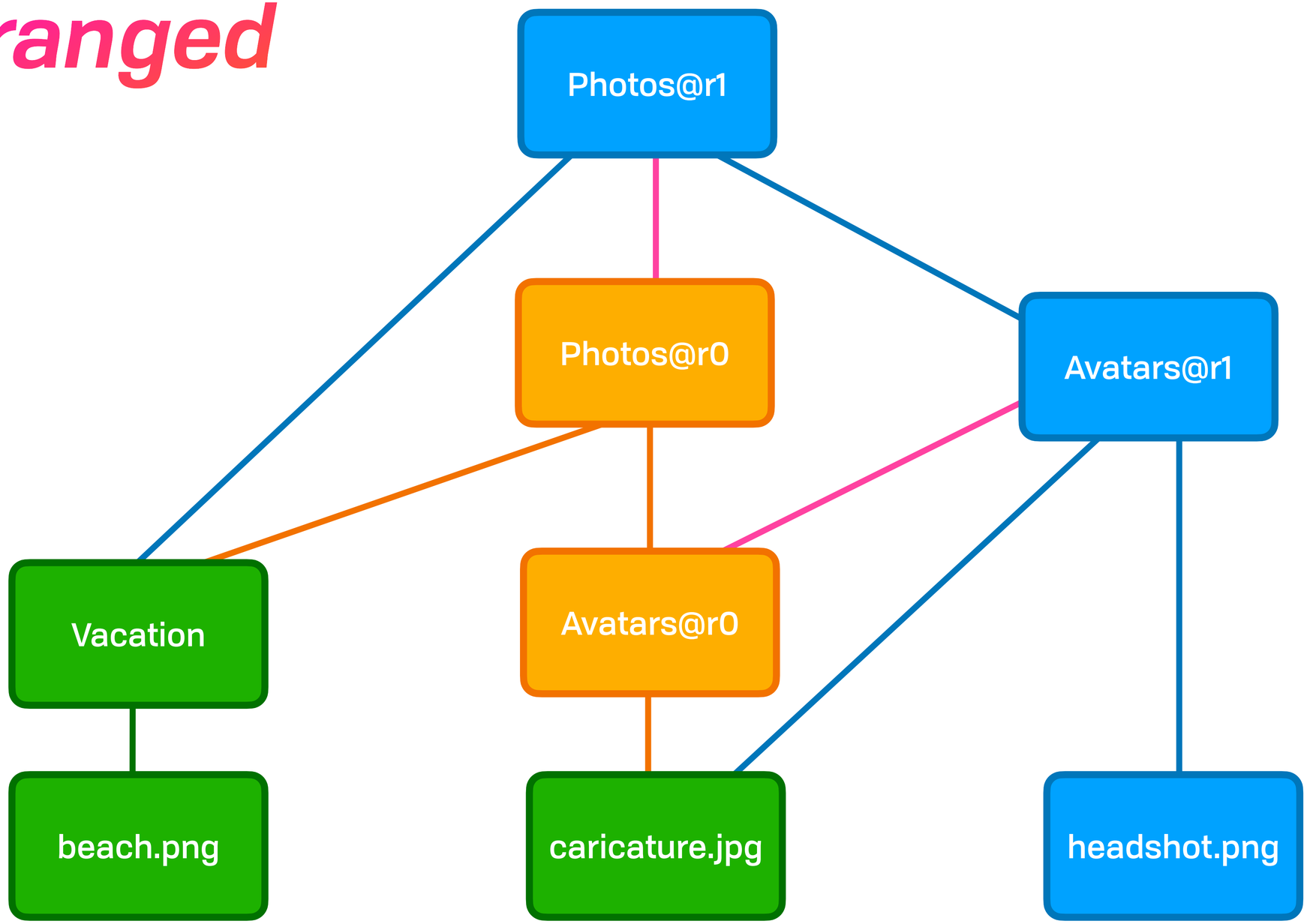
Persistent Versioning & Events





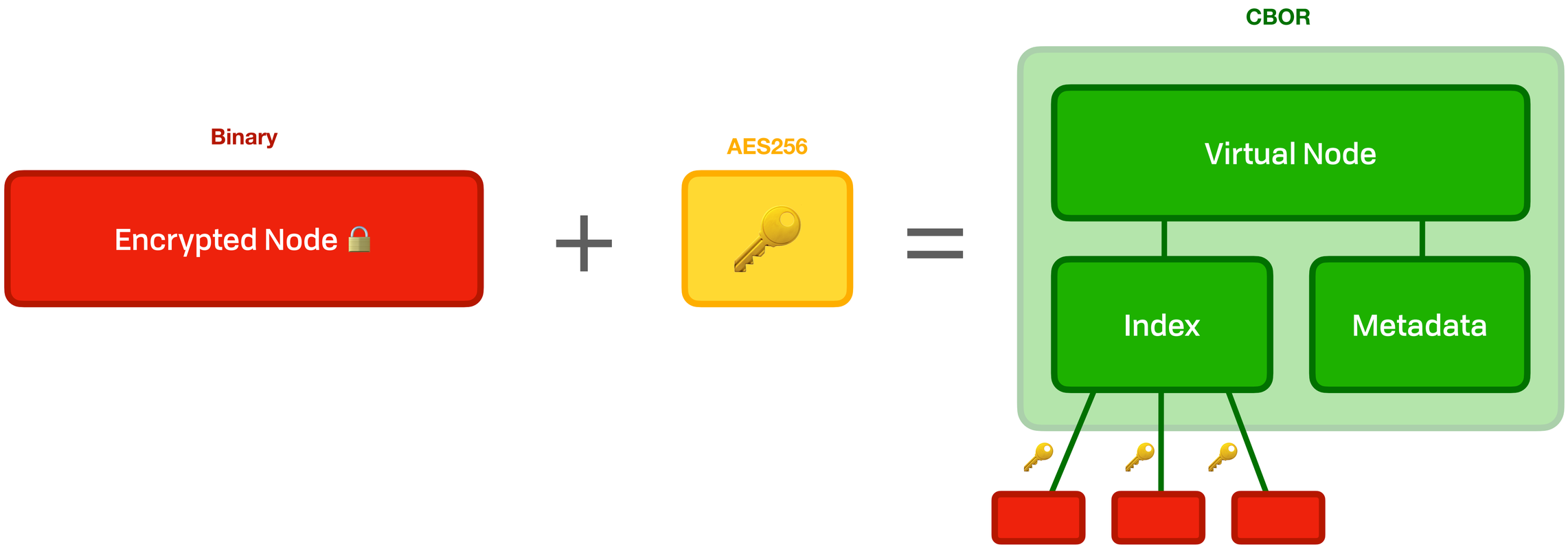
Securing Data Access

Rearranged



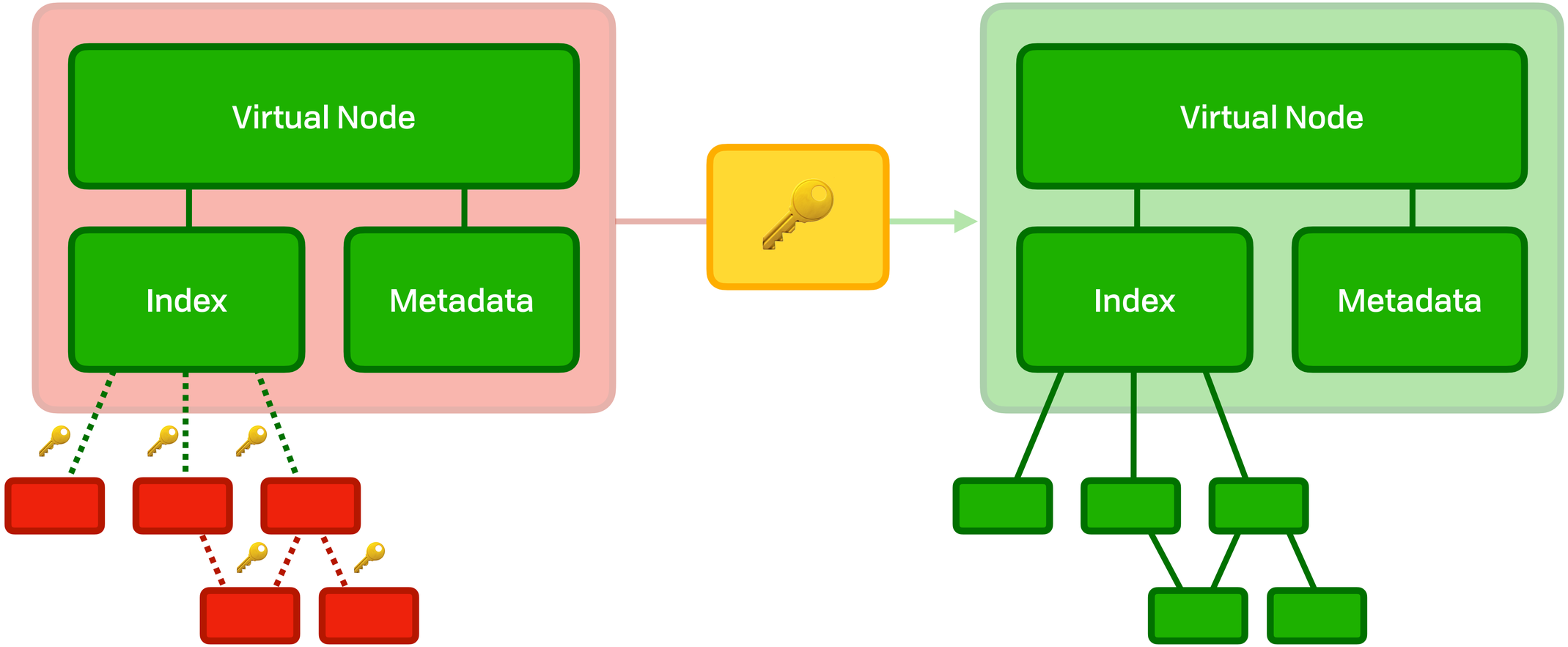
Securing Data Access

Private Nodes 🙈



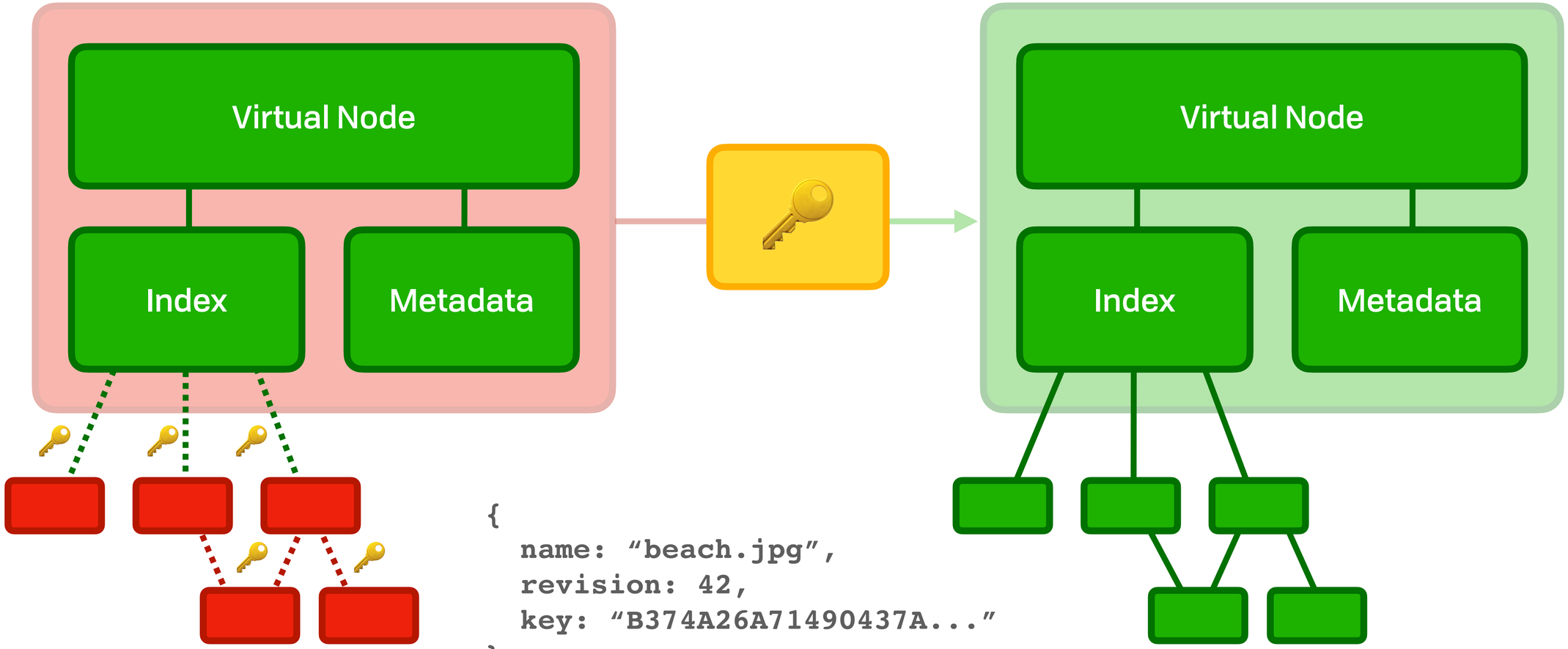
Securing Data Access

Cryptree 🎄



Securing Data Access

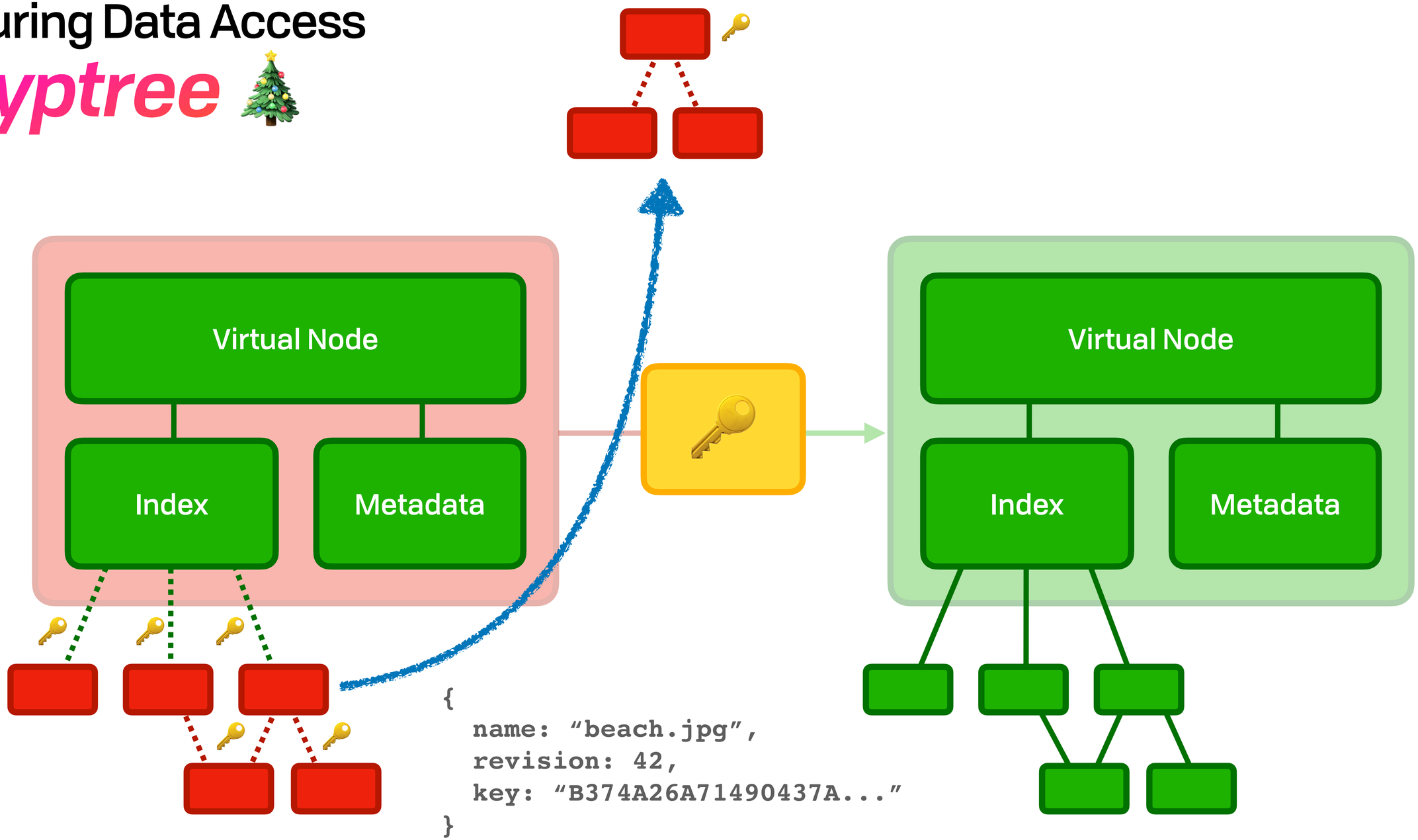
Cryptree 🎄




```
{
  name: “beach.jpg”,
  revision: 42,
  key: “B374A26A71490437A...”
}
```

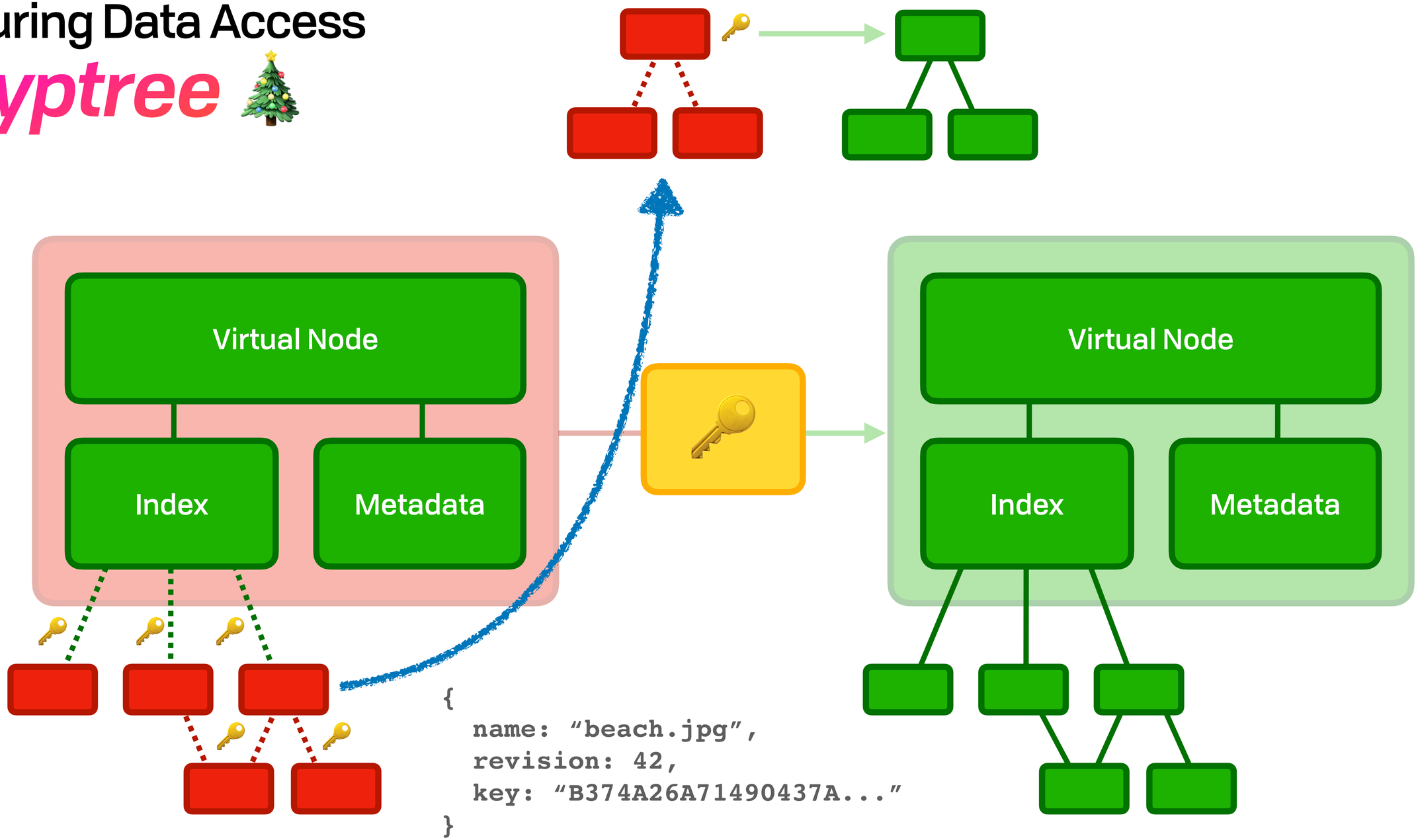
Securing Data Access

Cryptree 🎄



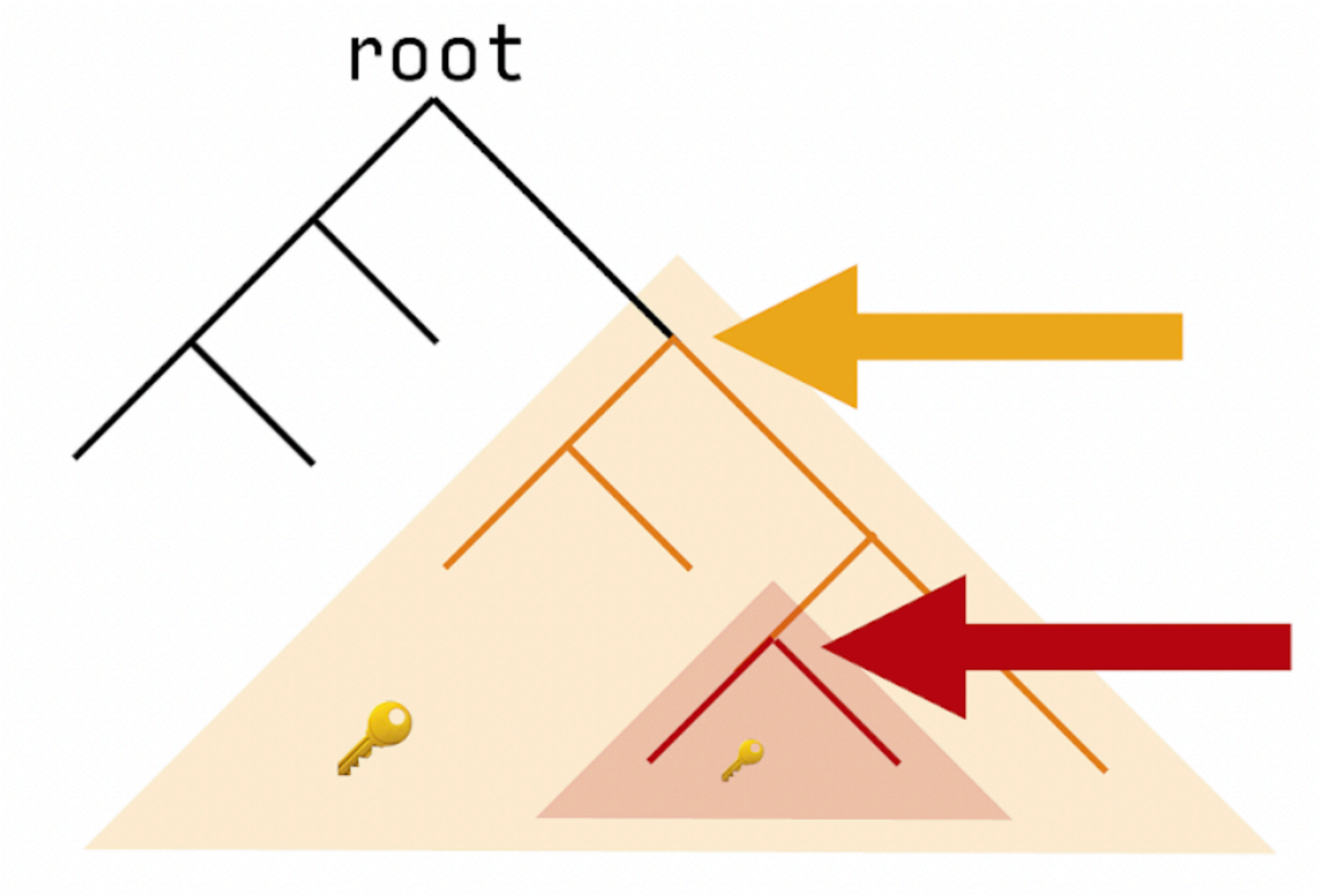
Securing Data Access

Cryptree 🎄



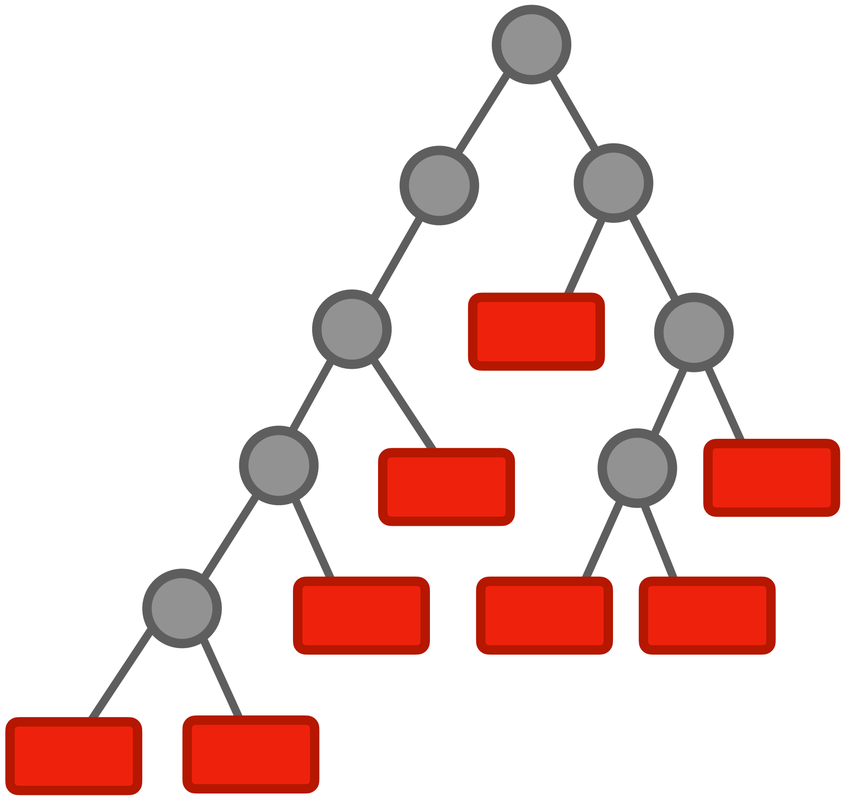
Securing Data Access

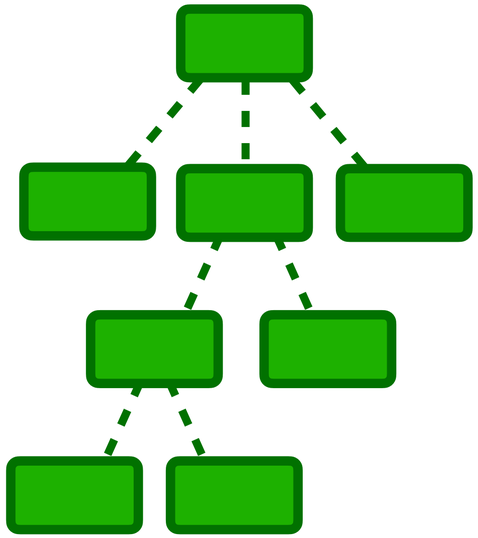
Subtree Read Access



Securing Data Access

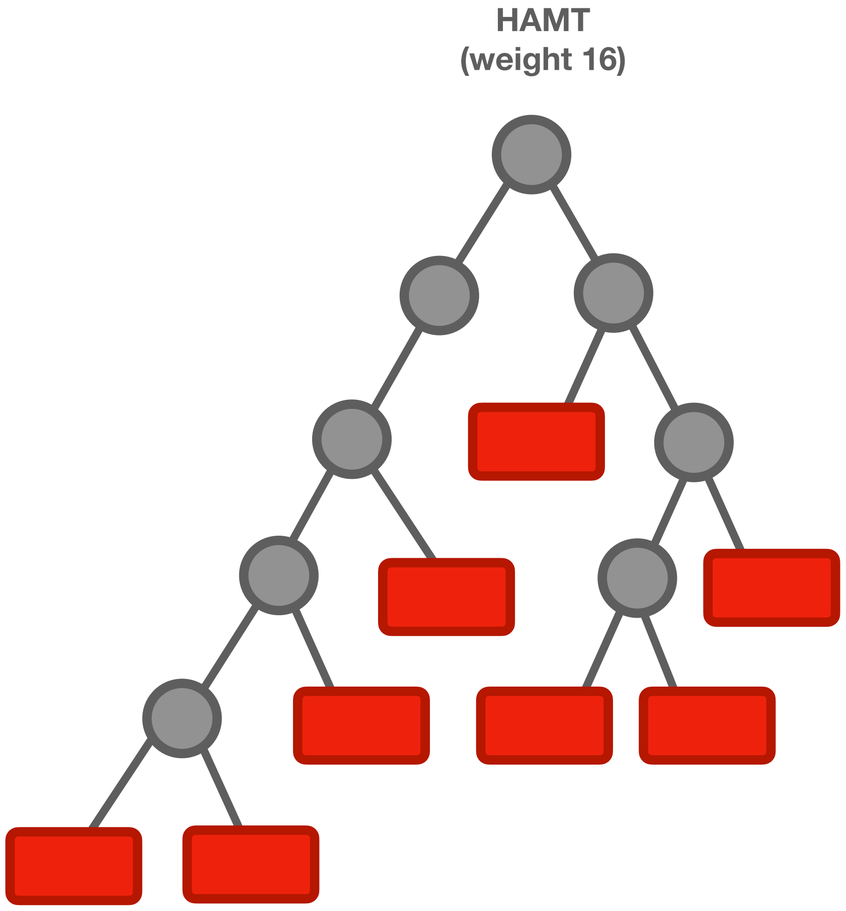
Encrypted Tree is Surprisingly Efficient

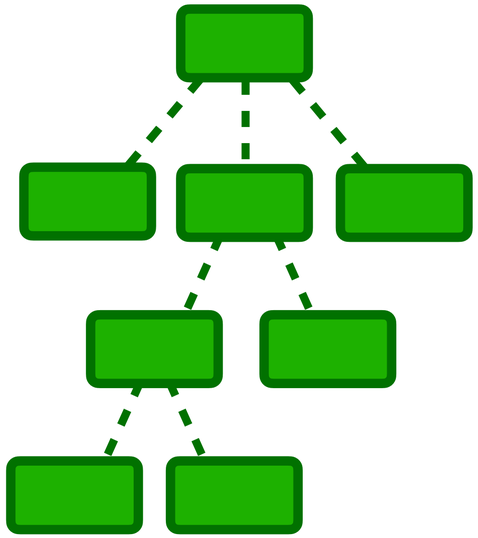




Securing Data Access

Encrypted Tree is Surprisingly Efficient

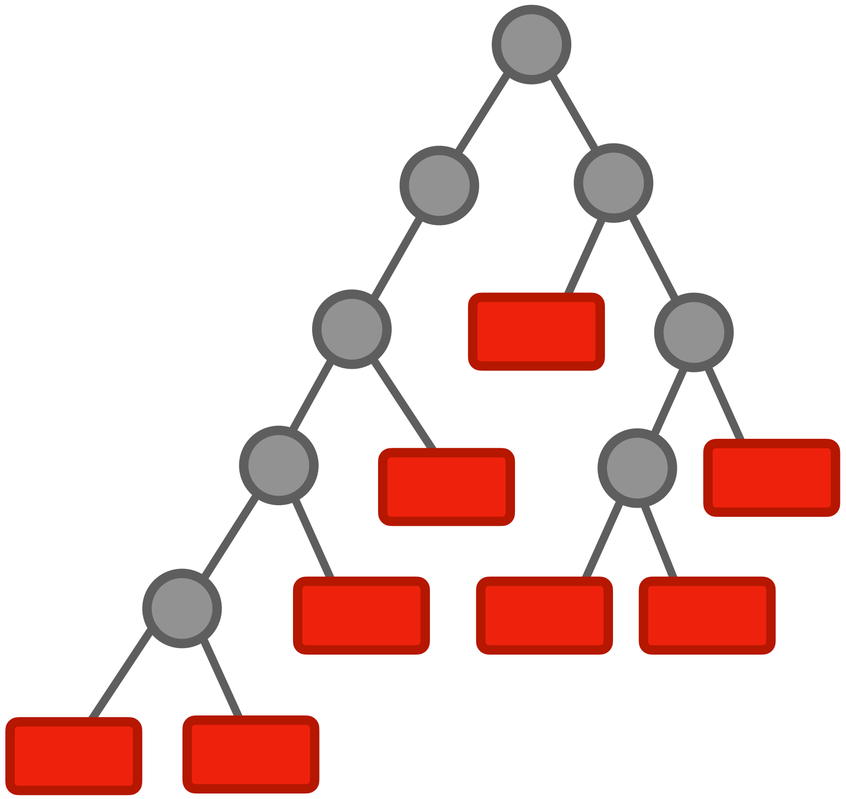


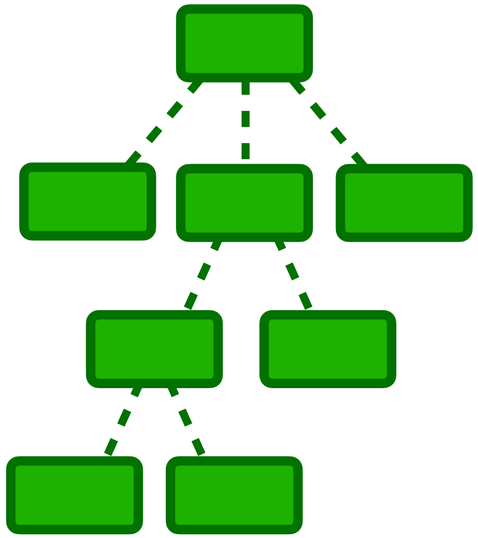


Securing Data Access

Encrypted Tree is Surprisingly Efficient

HAMT $16^3 = 4{,}096$ items
(weight 16) $16^4 = 65{,}536$ items



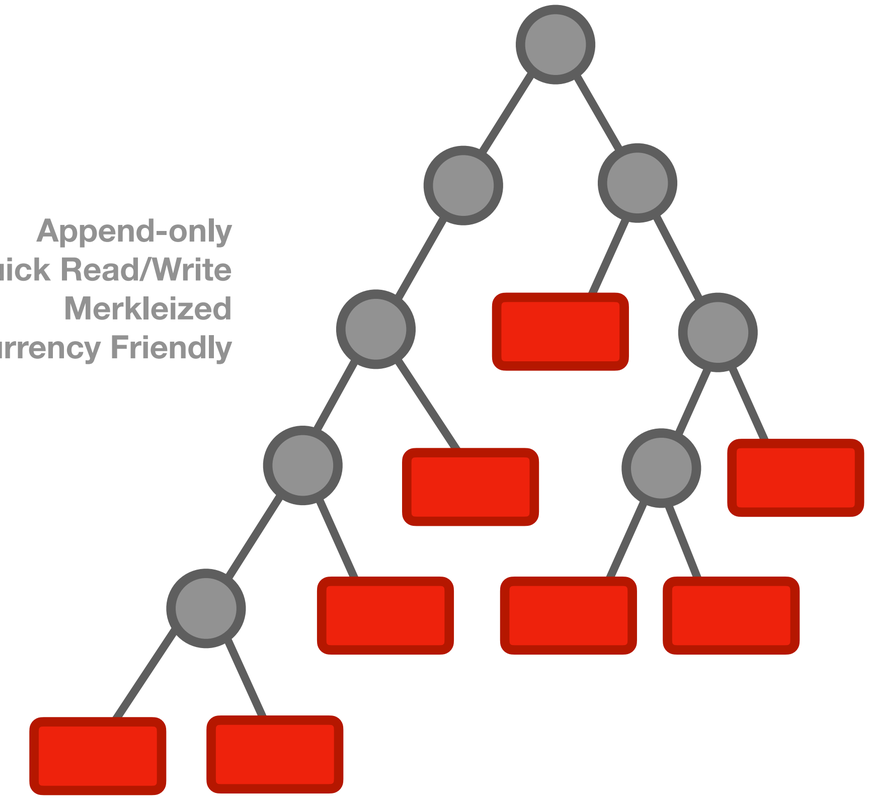


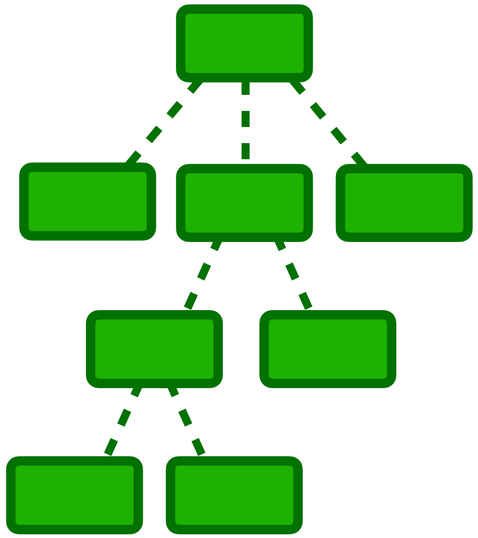
Securing Data Access

Encrypted Tree is Surprisingly Efficient

HAMT $16^3 = 4{,}096$ items
(weight 16) $16^4 = 65{,}536$ items

Append-only
Quick Read/Write
Merkleized
Concurrency Friendly



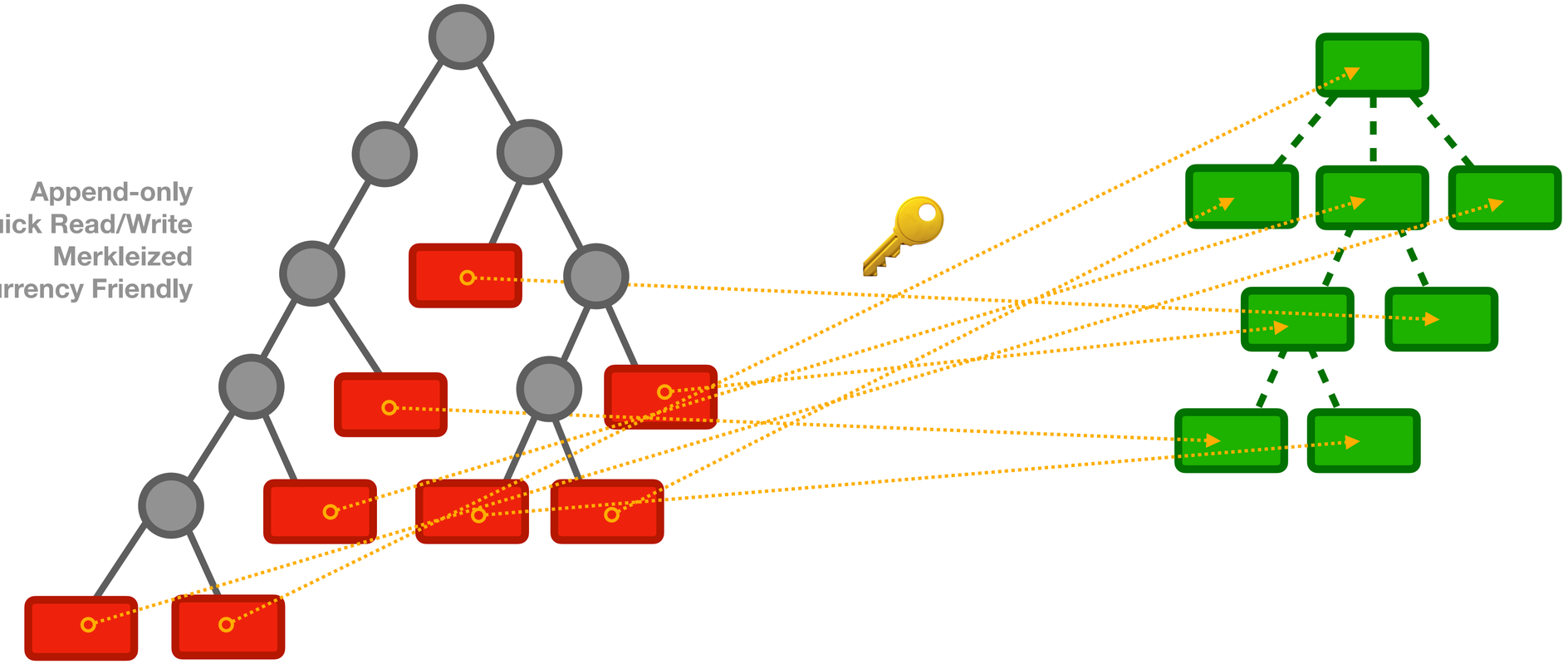


Securing Data Access

Encrypted Tree is Surprisingly Efficient

HAMT $16^3 = 4{,}096$ items
(weight 16) $16^4 = 65{,}536$ items

Append-only
Quick Read/Write
Merkleized
Concurrency Friendly



Securing Data Access

Namefilters & Hiding Paths

Securing Data Access

Namefilters & Hiding Paths

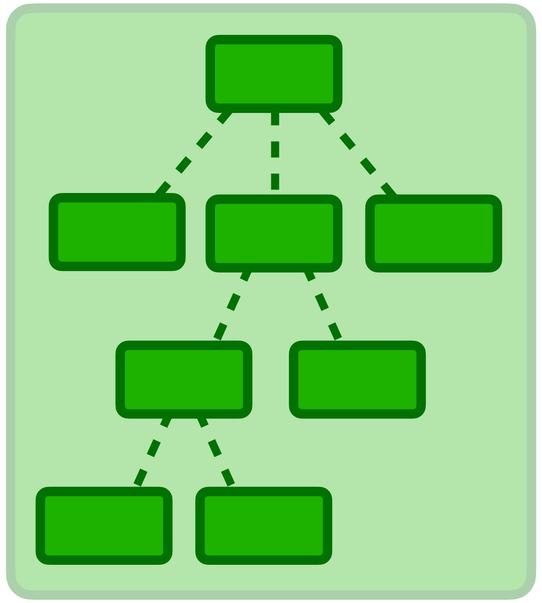
- Bare Filter
  - `parentFilter`
    - `AND bloom(SHA(aesKey))`
    - `AND bloom(SHA(aesKey ++ revision))`
- Saturation
  - `nameFilter AND bloom(SHA(nameFilter))`
  - Repeat until threshold bits flipped

Securing Data Access

Access-Mediated Collaborative Rooting

Securing Data Access

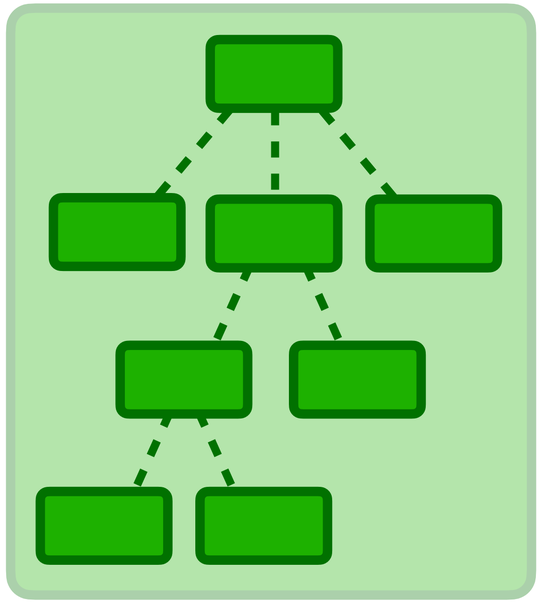
Access-Mediated Collaborative Rooting



Rev 0

Securing Data Access

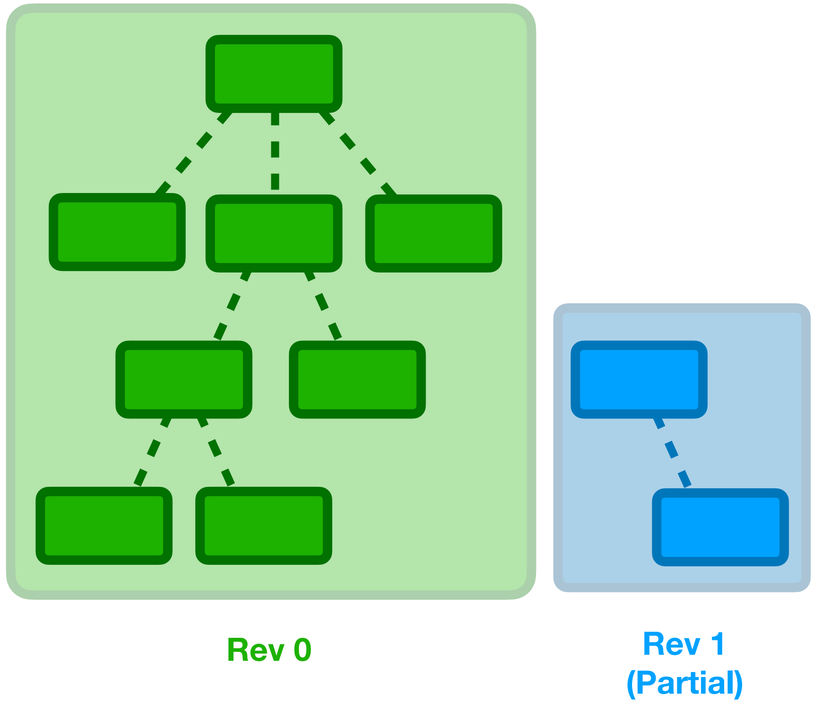
Access-Mediated Collaborative Rooting



Rev 0

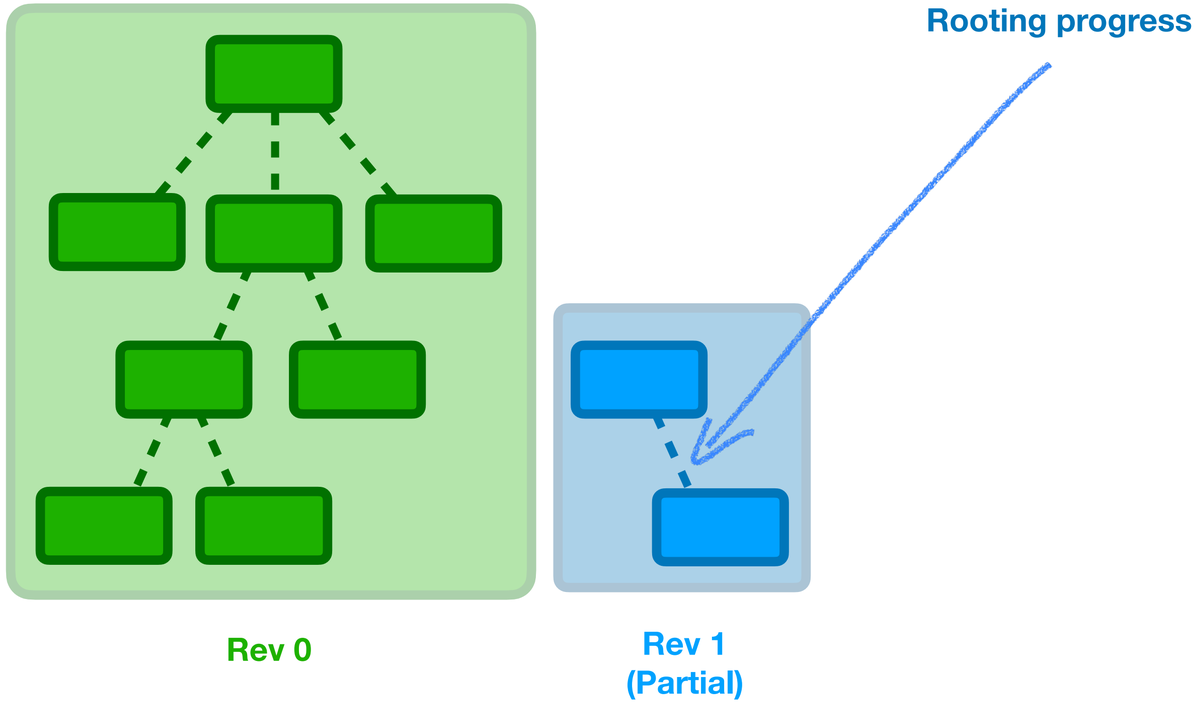
Securing Data Access

Access-Mediated Collaborative Rooting



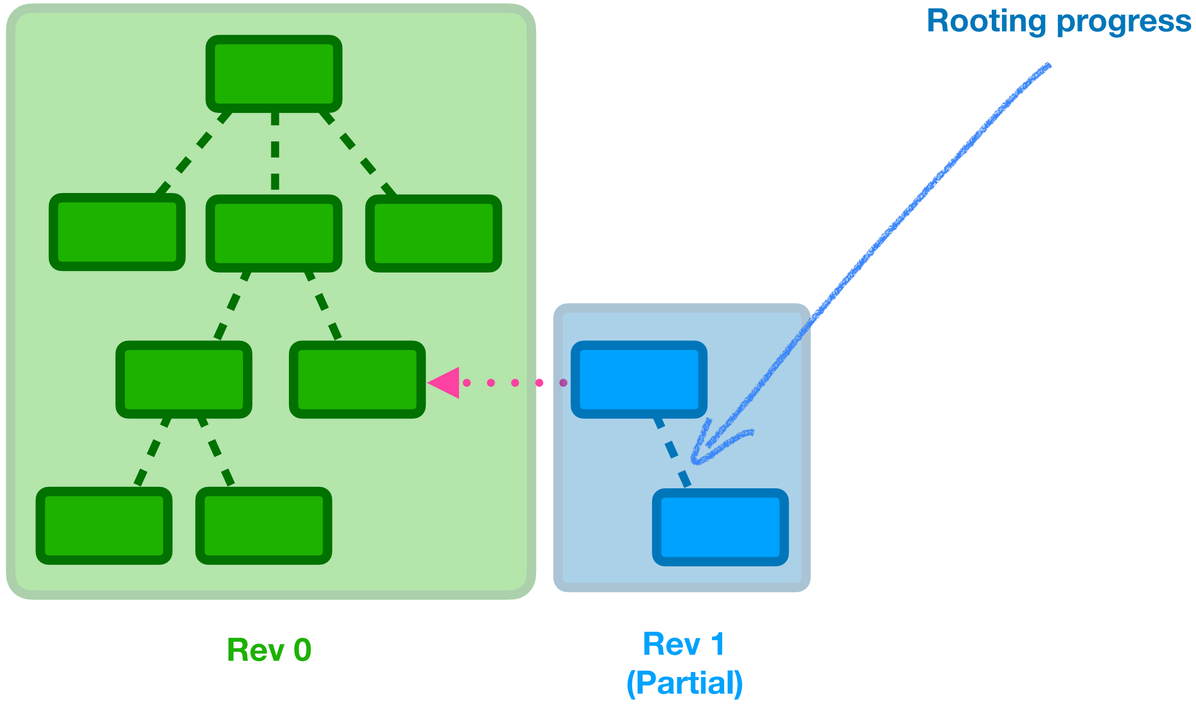
Securing Data Access

Access-Mediated Collaborative Rooting



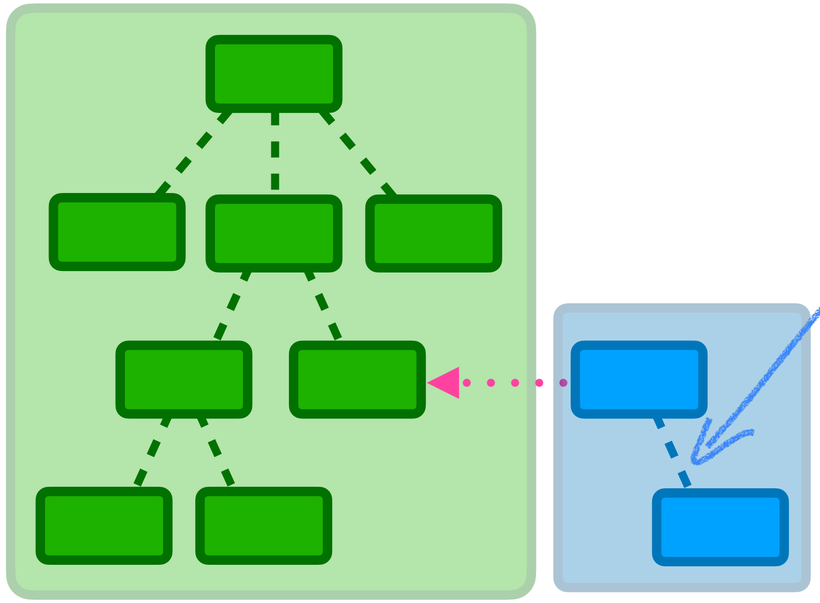
Securing Data Access

Access-Mediated Collaborative Rooting



Securing Data Access

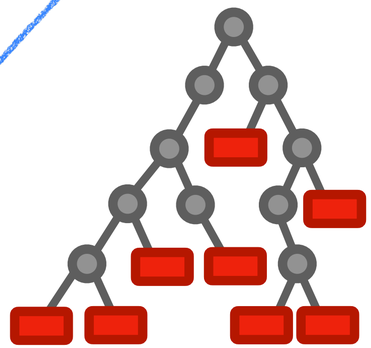
Access-Mediated Collaborative Rooting



Rev 0

Rev 1
(Partial)

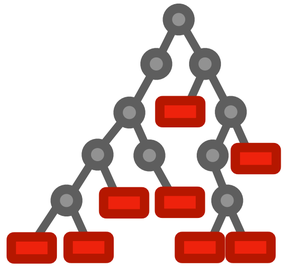
Rooting progress

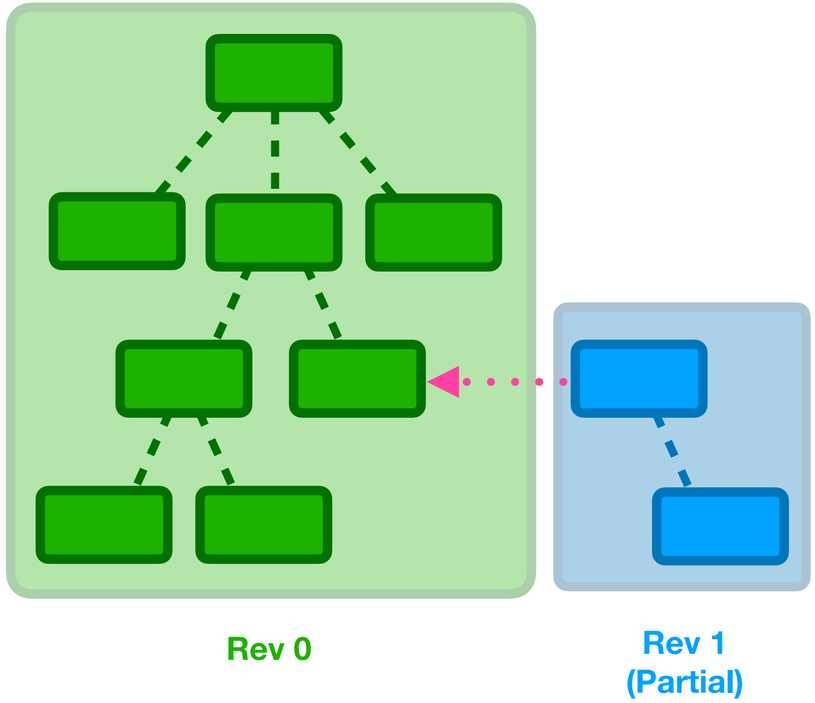


No common root at this layer!
Attached via HAMT

Securing Data Access

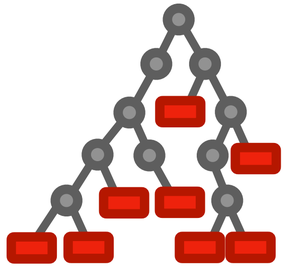
Progressive Fast Forward

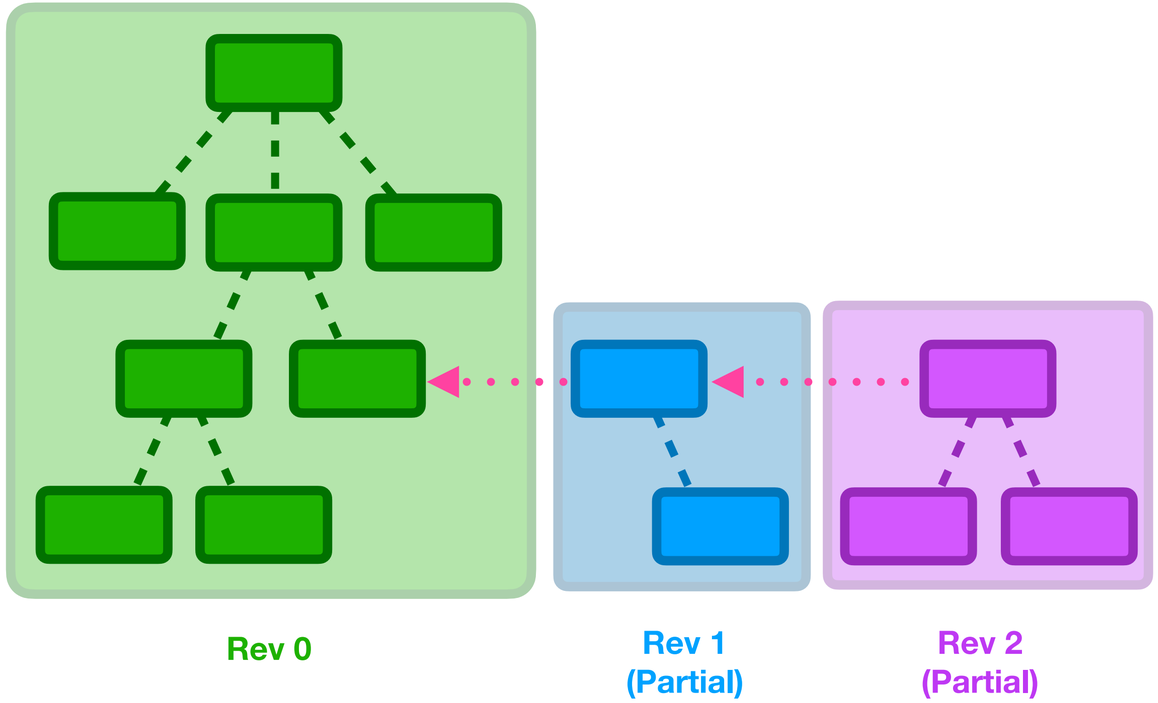




Securing Data Access

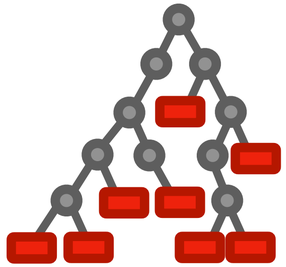
Progressive Fast Forward

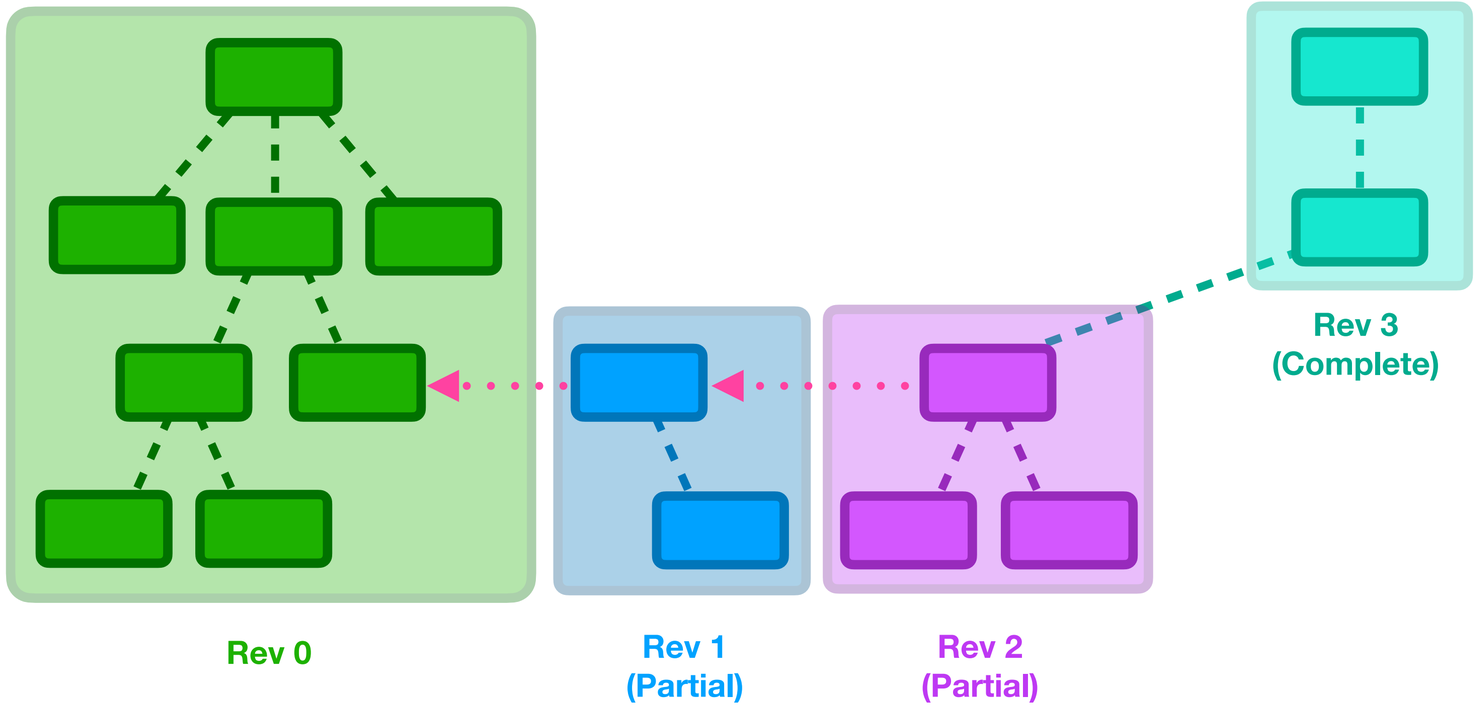




Securing Data Access

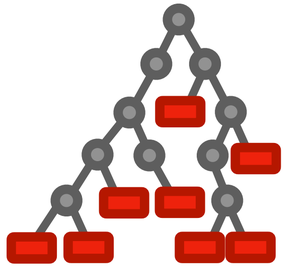
Progressive Fast Forward

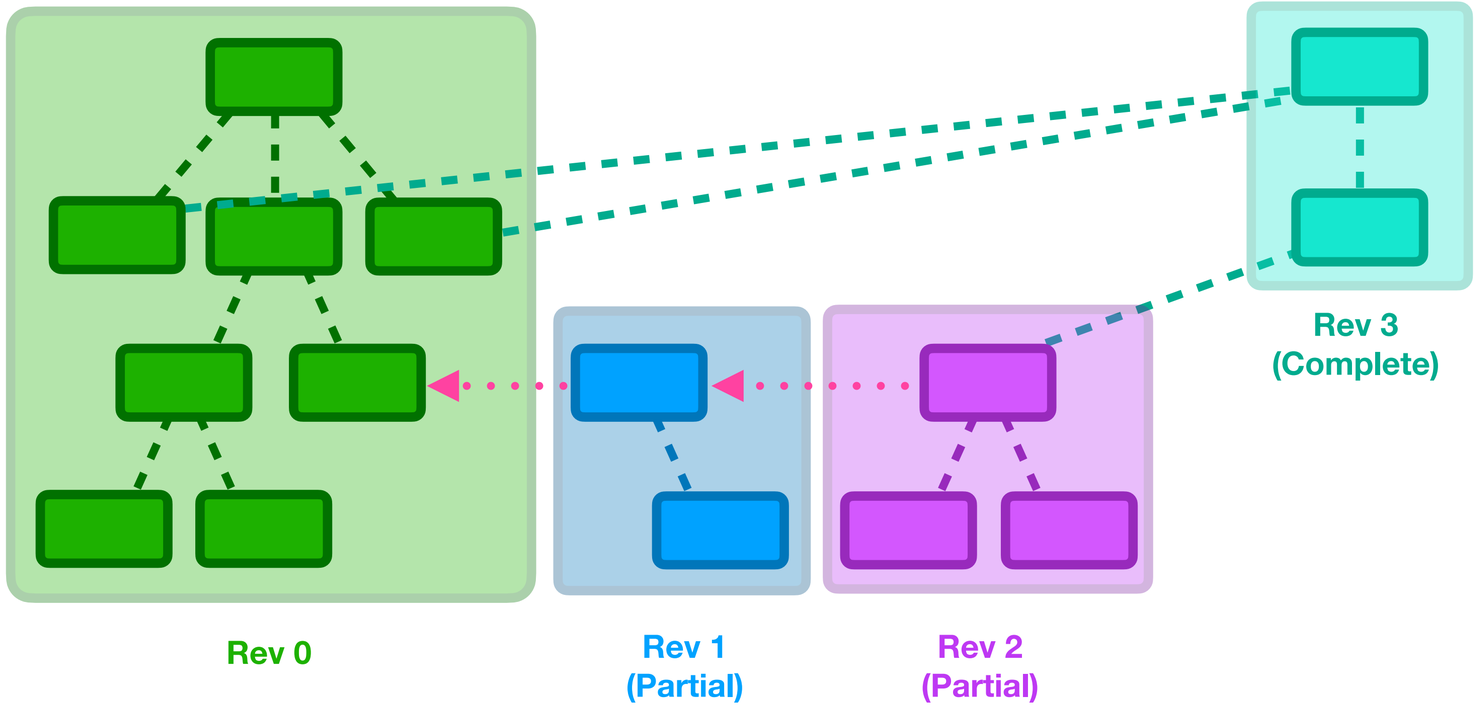




Securing Data Access

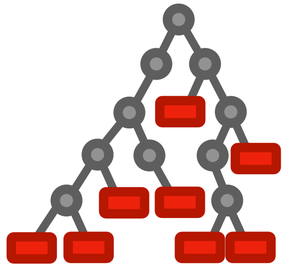
Progressive Fast Forward

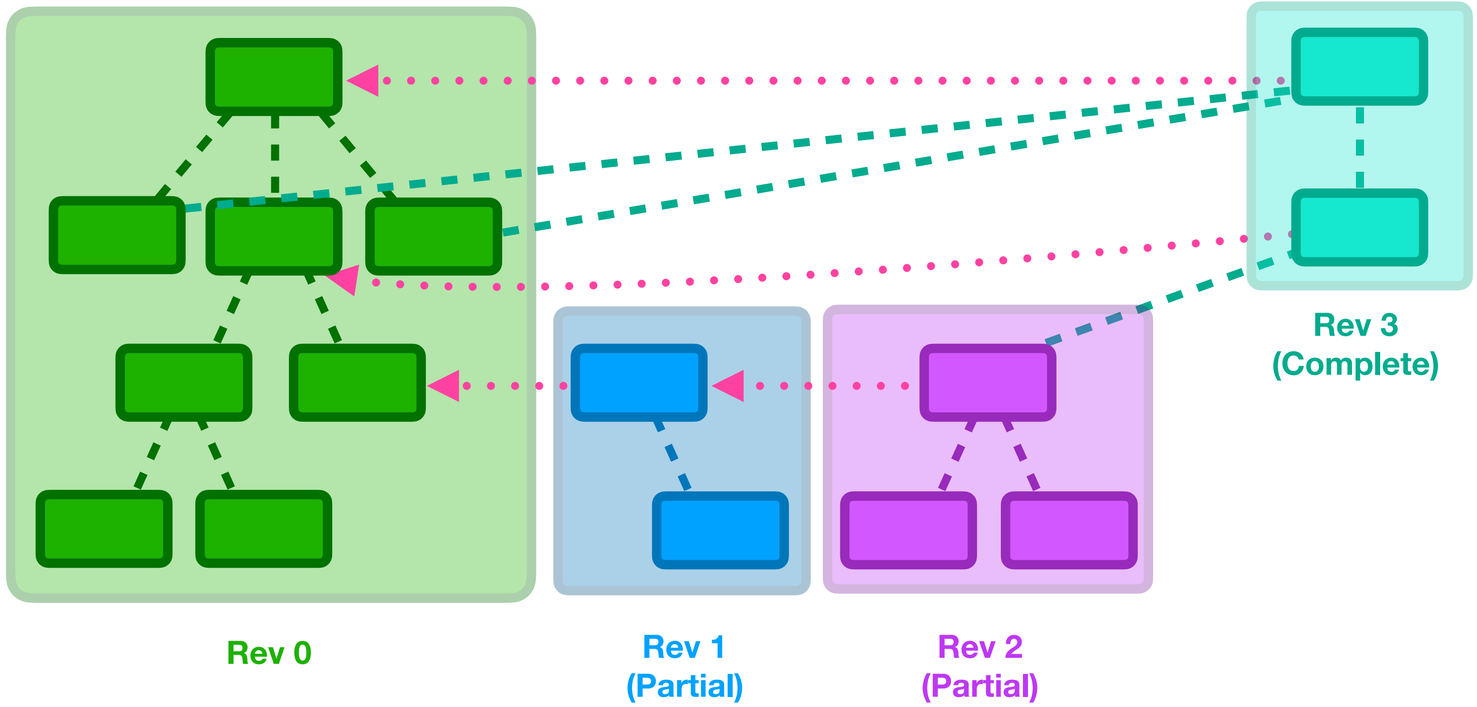




Securing Data Access

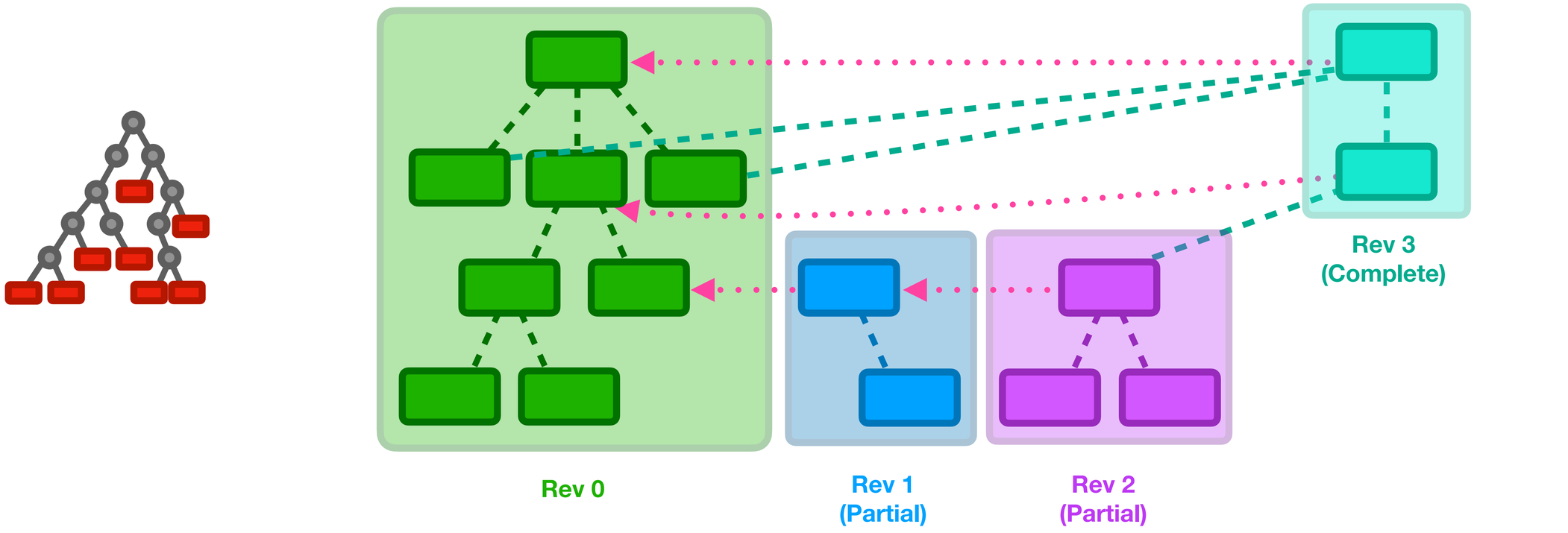
Progressive Fast Forward





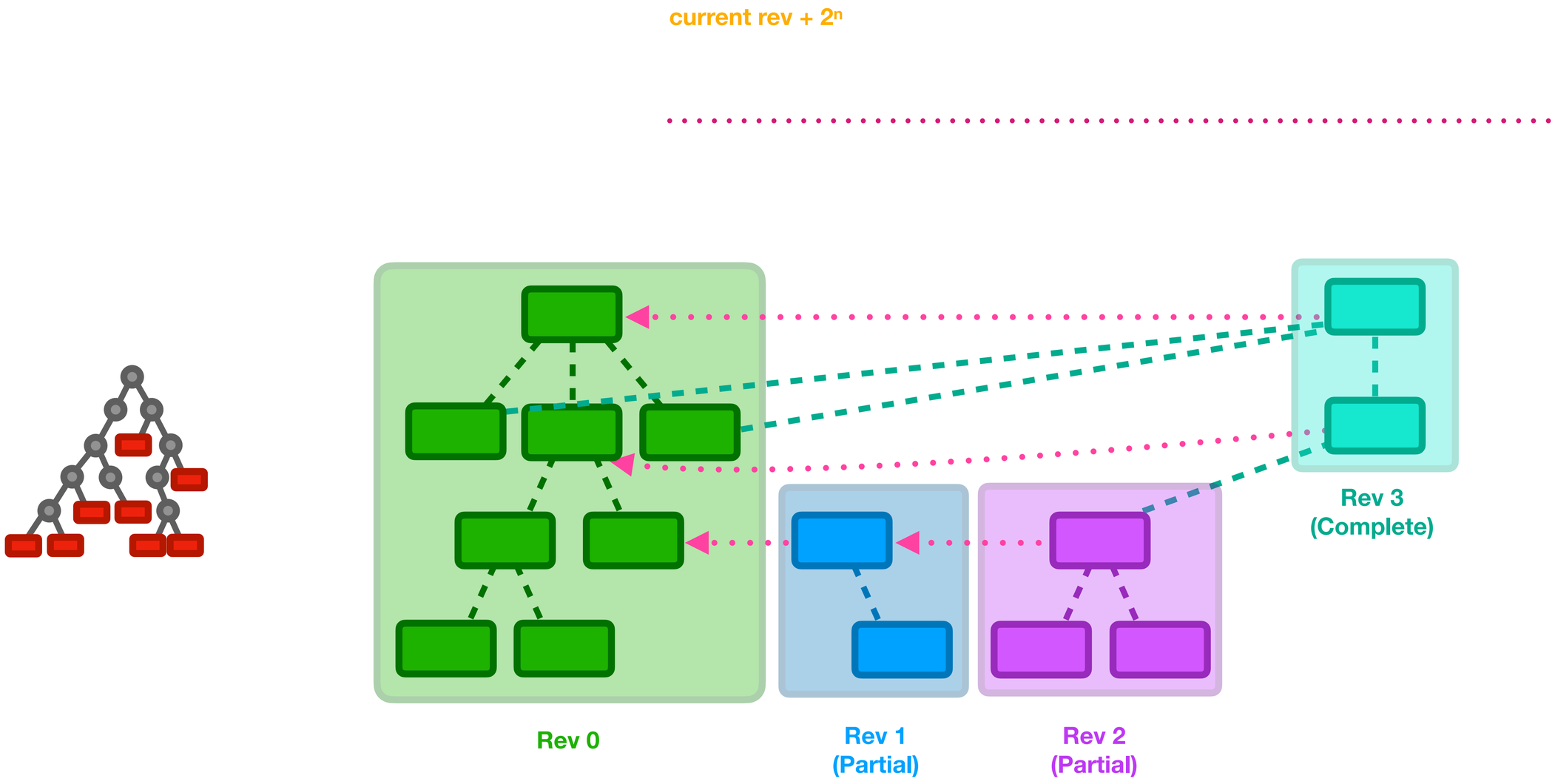
Securing Data Access

Progressive Fast Forward



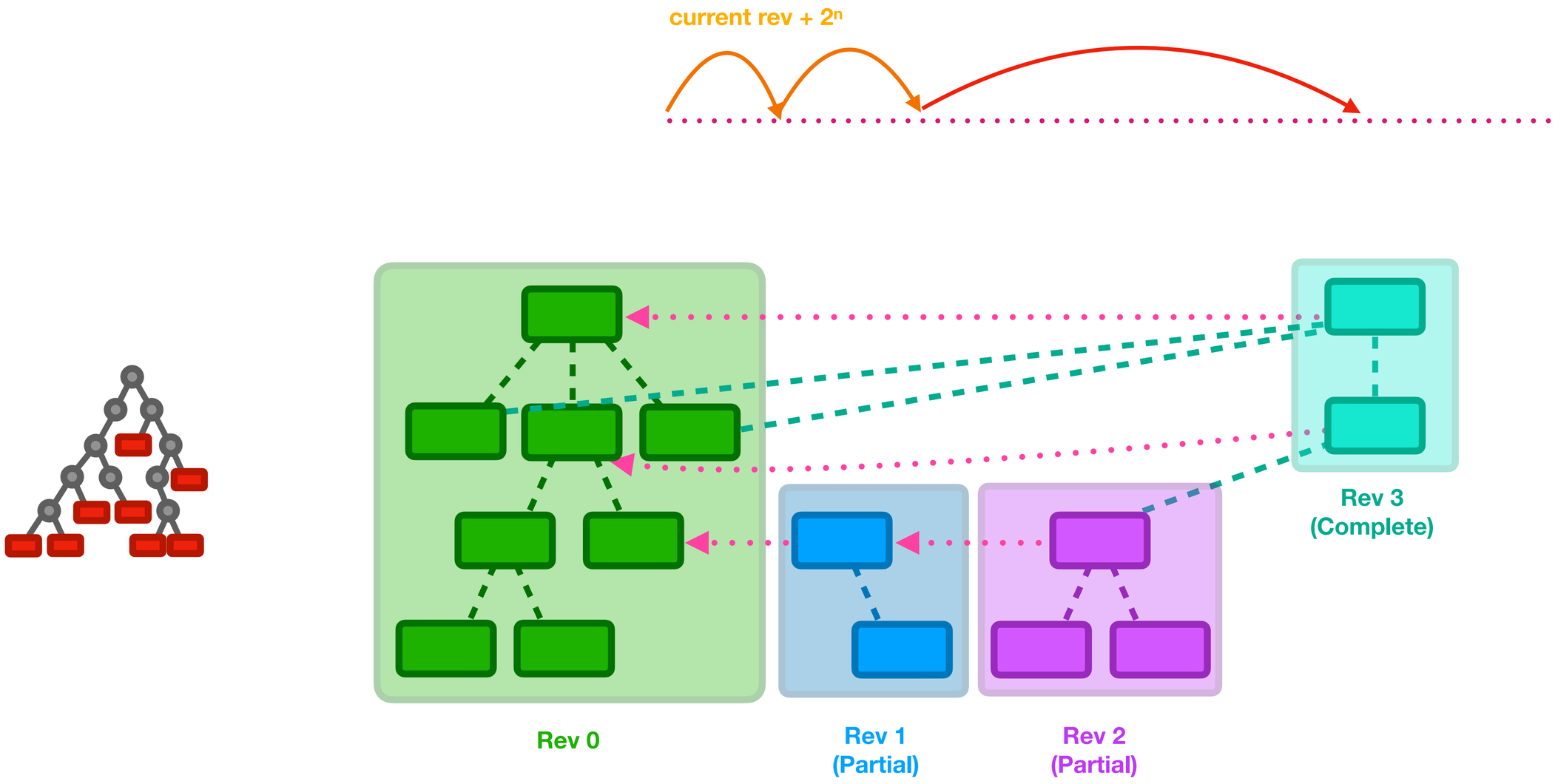
Securing Data Access

Progressive Fast Forward



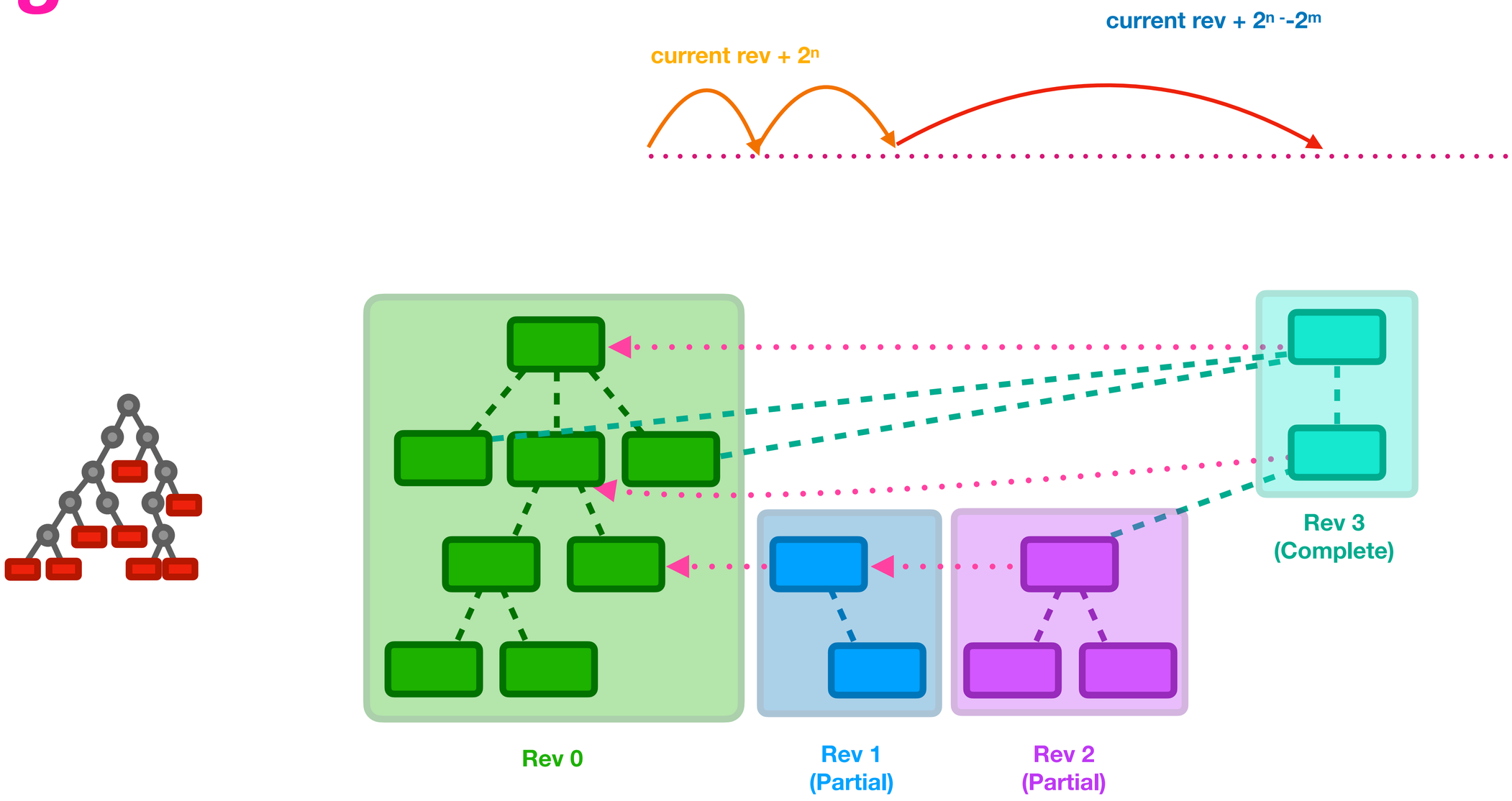
Securing Data Access

Progressive Fast Forward



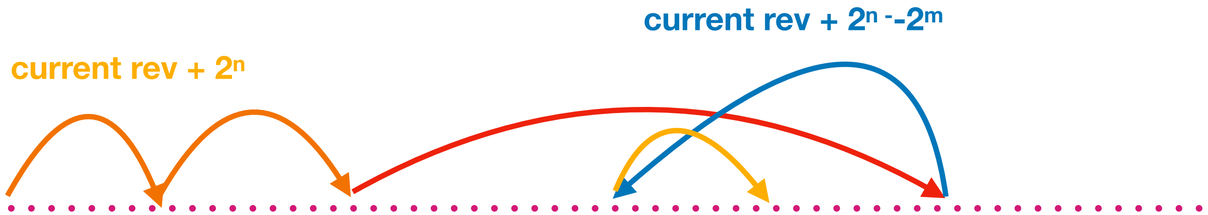
Securing Data Access

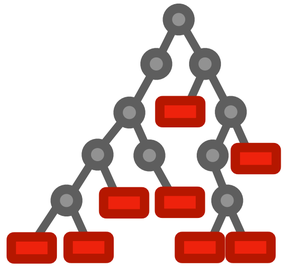
Progressive Fast Forward

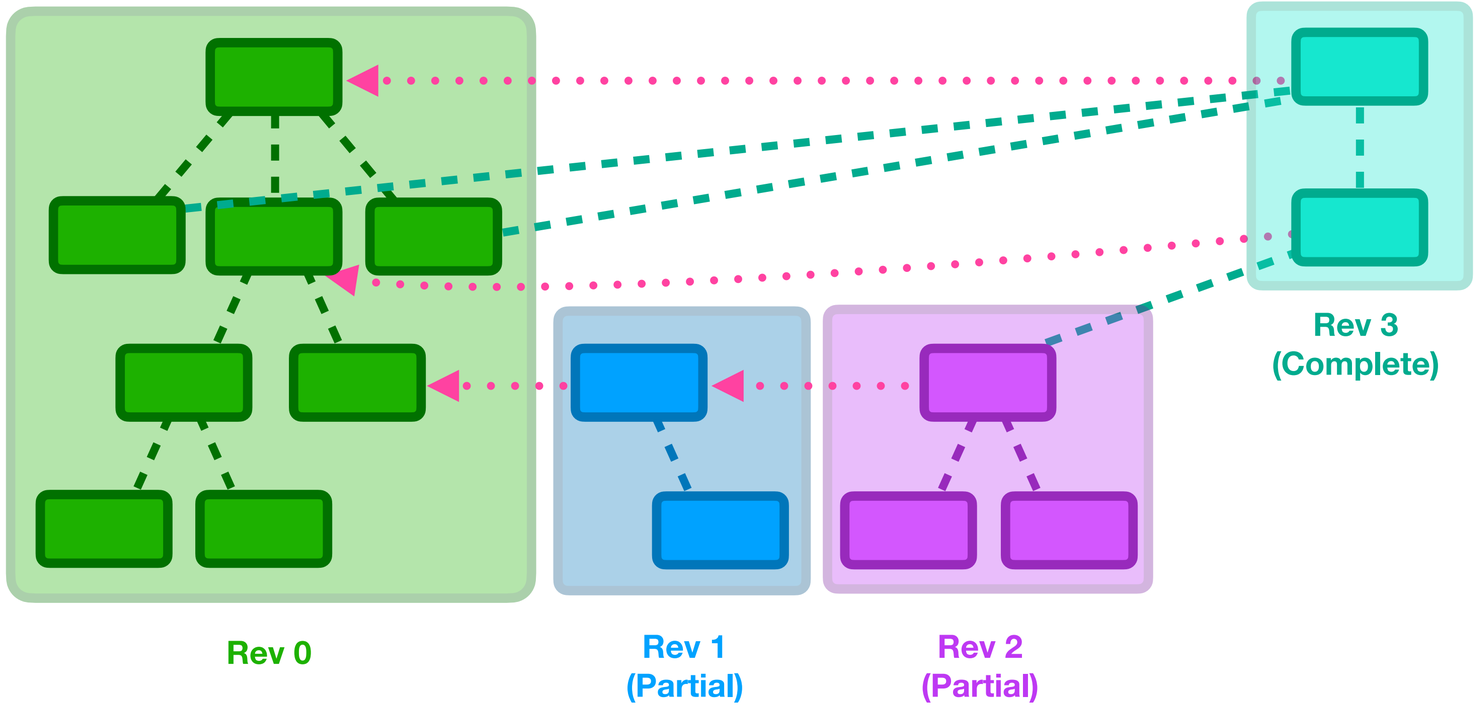


Securing Data Access

Progressive Fast Forward

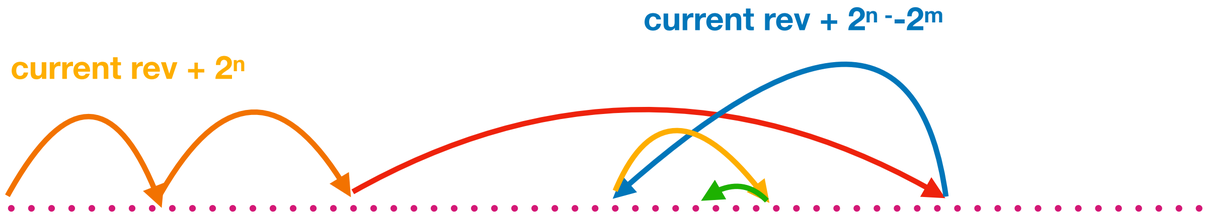


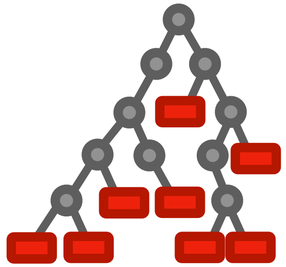


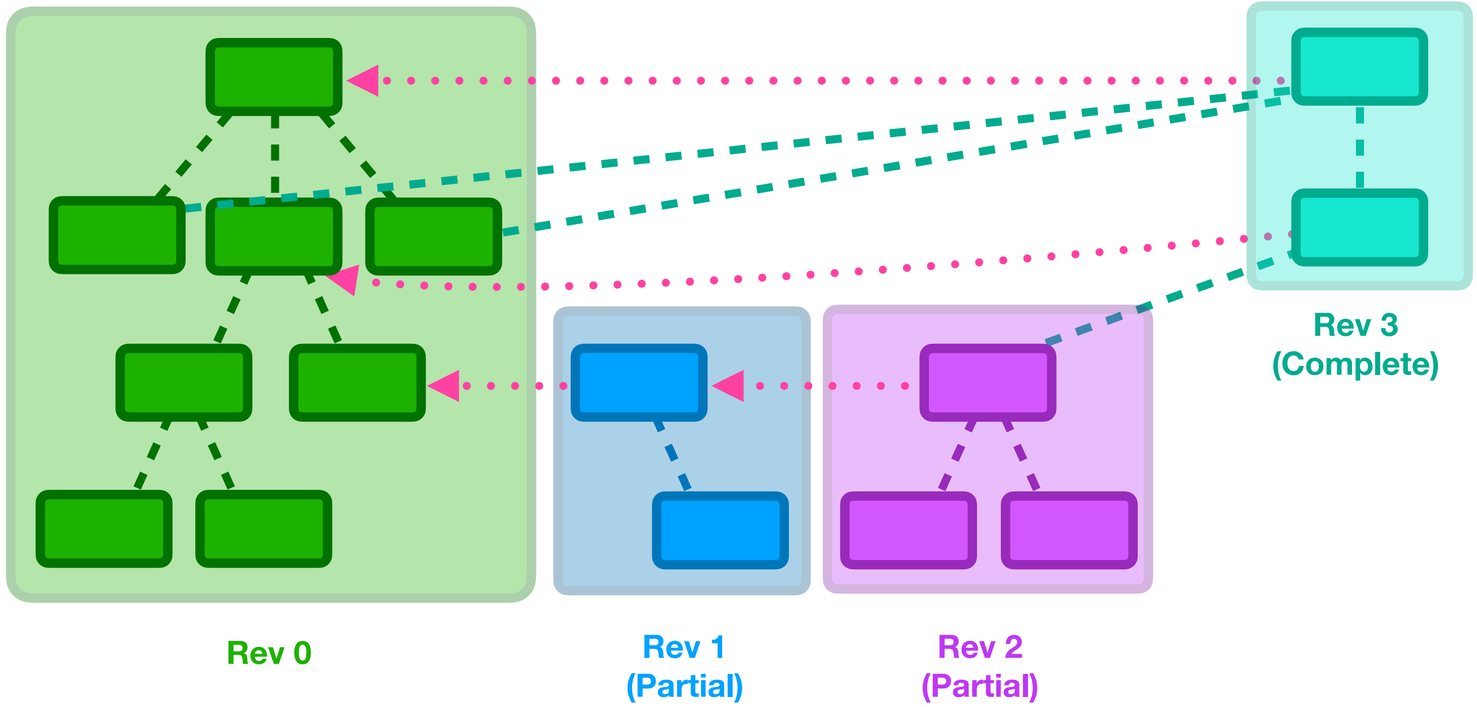


Securing Data Access

Progressive Fast Forward

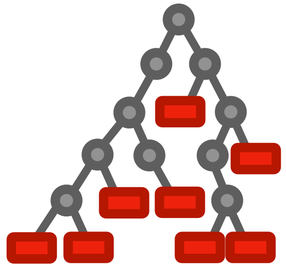


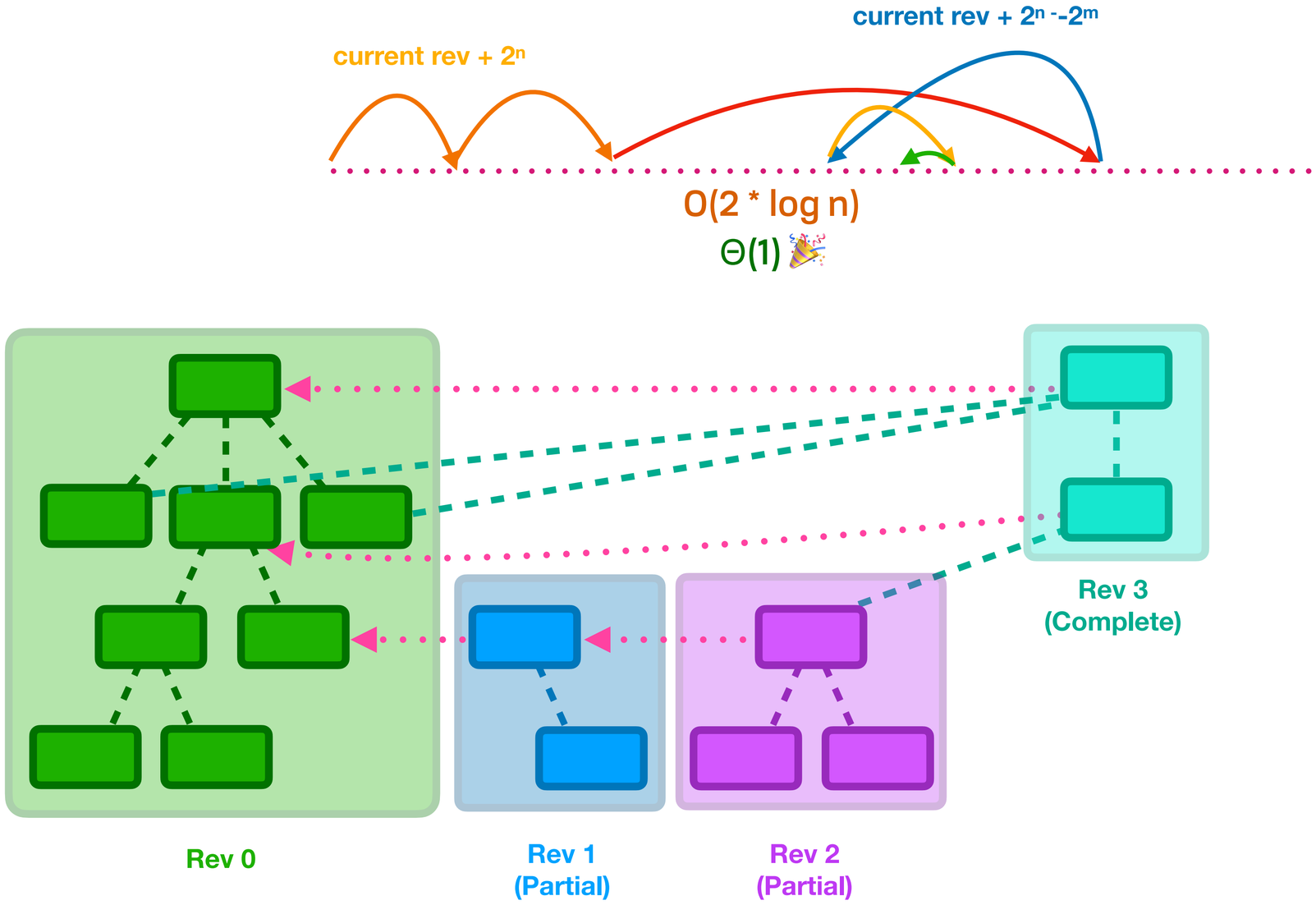




Securing Data Access

Progressive Fast Forward





Securing Data Access

File Sharing

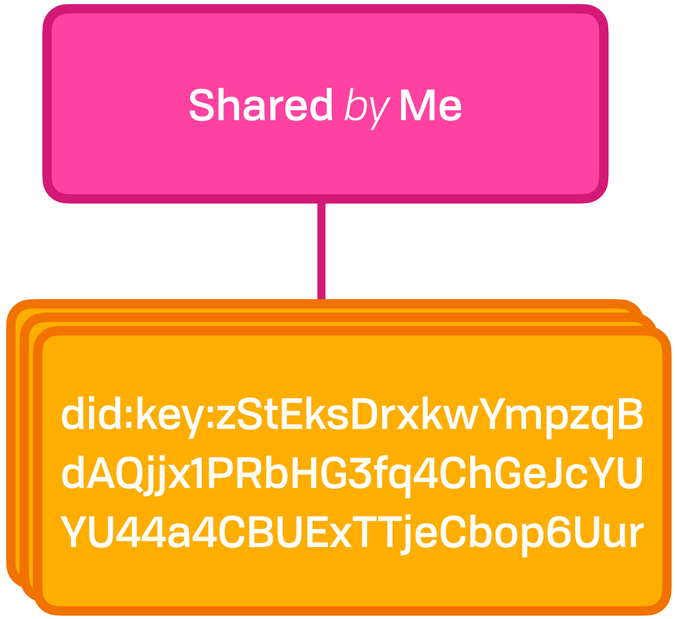
Securing Data Access

File Sharing

Shared *by* Me

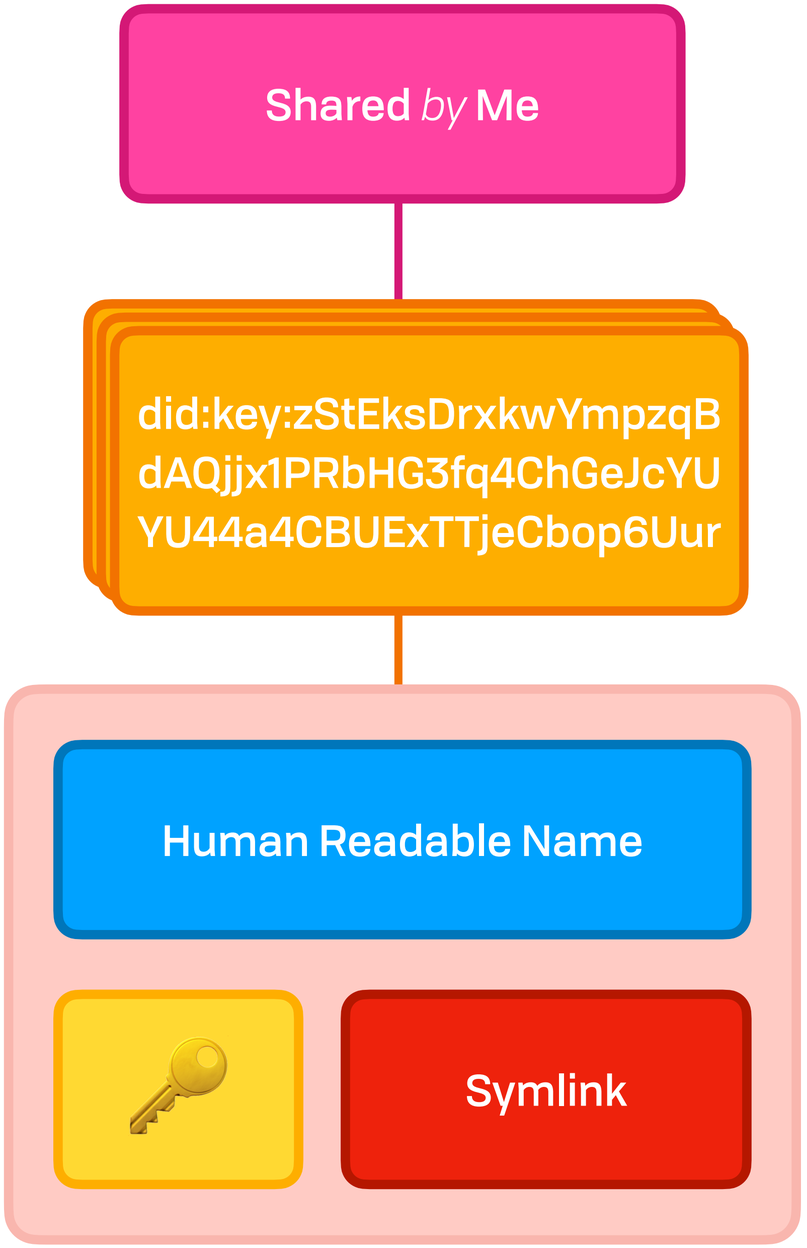
Securing Data Access

File Sharing



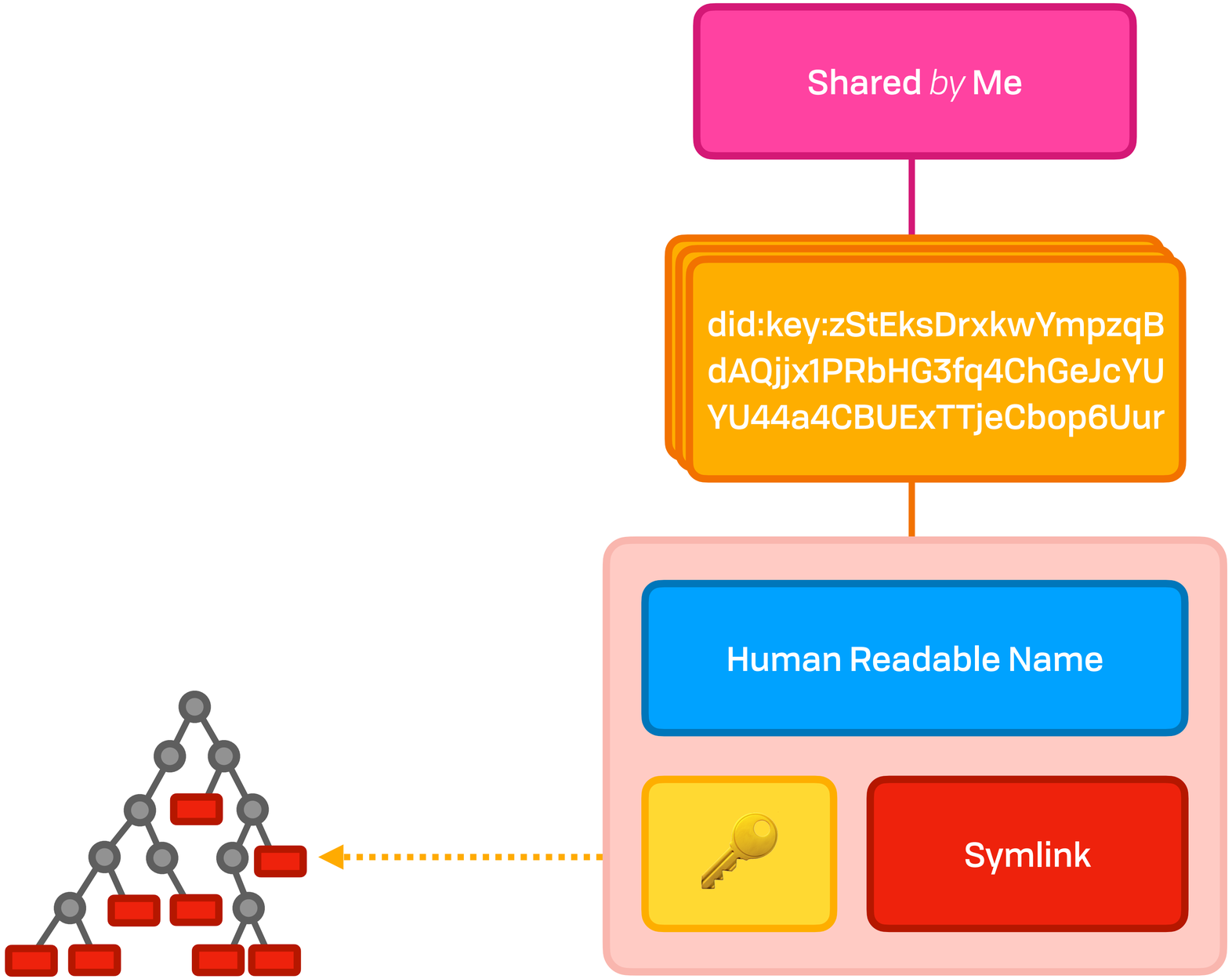
Securing Data Access

File Sharing



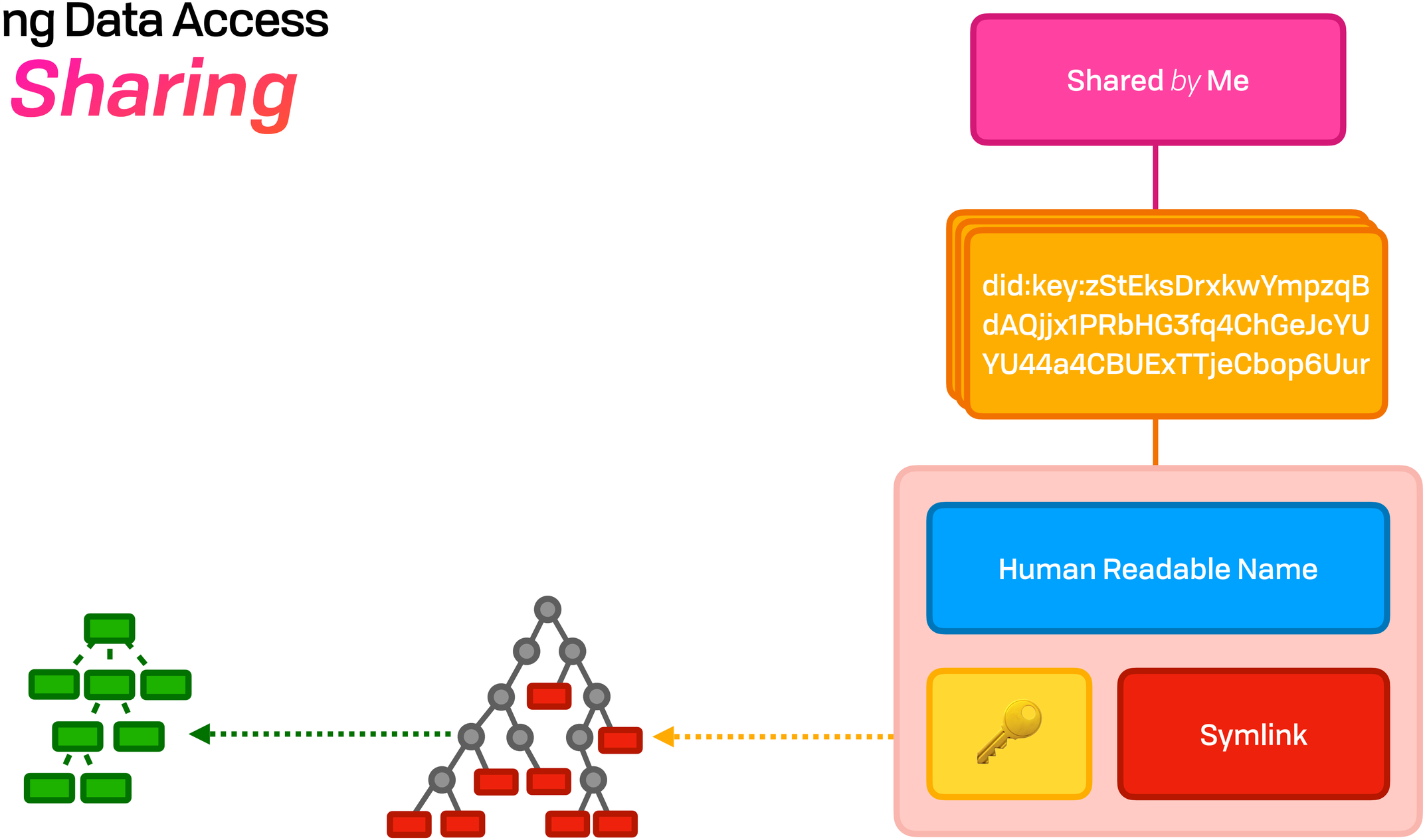
Securing Data Access

File Sharing



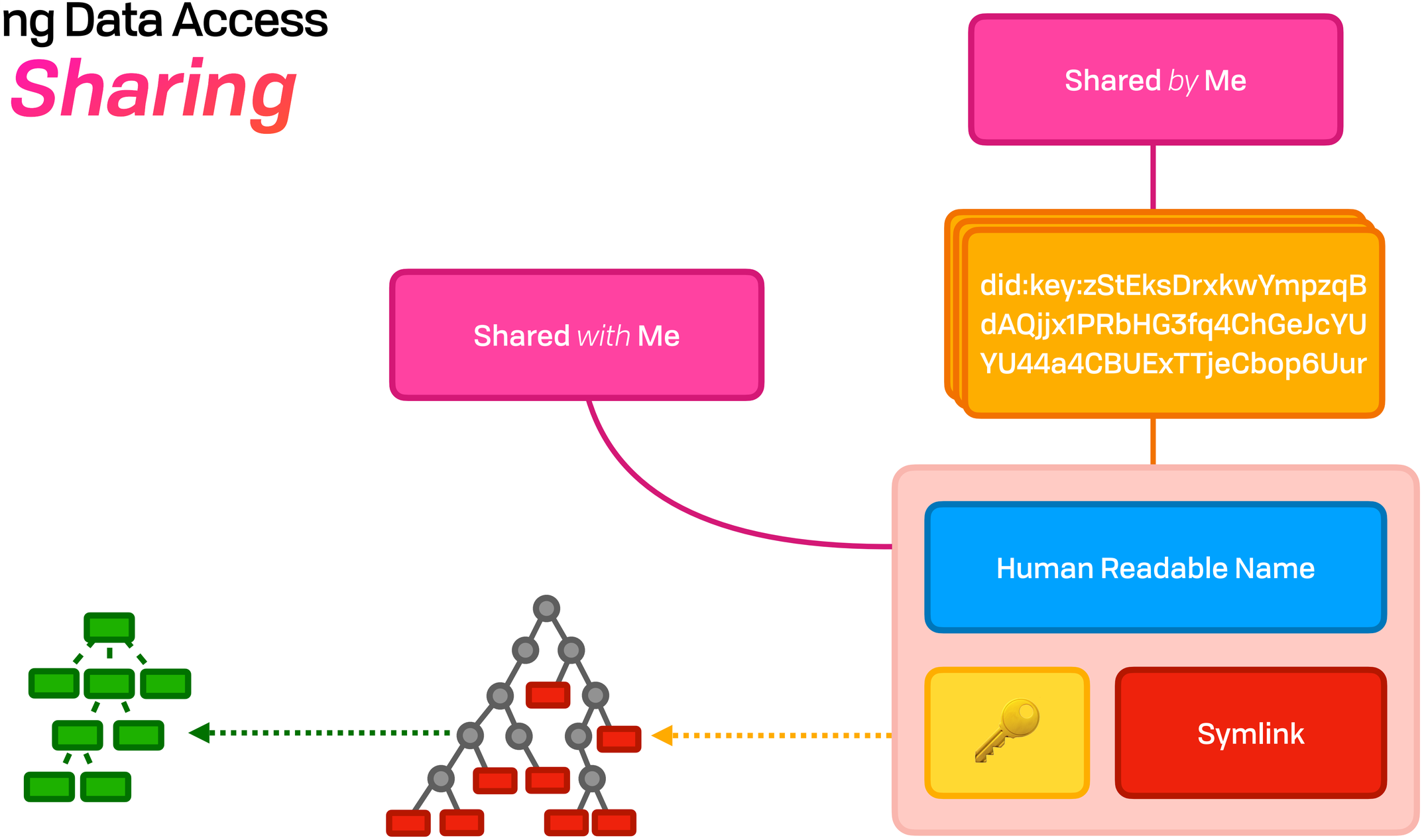
Securing Data Access

File Sharing



Securing Data Access

File Sharing



Securing Data Access

So we can read recursively
encrypted trees that live anywhere.

So we can read recursively
encrypted trees that live anywhere.

So we can read recursively
encrypted trees that live anywhere.

Great!

So we can read recursively
encrypted trees that live anywhere.

Great!

Securing Data Access

So we can read recursively
encrypted trees that live anywhere.

Great!

How do you do writes if a
a server can't check the content?

User Controlled, Serverless, Universal Auth & ID

...and UCAN Too

💪

UCAN

Self-Sovereign Identity

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"
  }]
}
```

UCAN

Self-Sovereign Identity

- W3C

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"
  }]
}
```

UCAN

Self-Sovereign Identity

- W3C
- 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"
  }]
}
```

UCAN

Self-Sovereign Identity

- W3C
- Microsoft
- Government of British Columbia

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"
  }]
}
```

UCAN

Self-Sovereign Identity

- W3C
- Microsoft
- Government of British Columbia
- 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"
  }]
}
```

UCAN

Self-Sovereign Identity

- W3C
- Microsoft
- Government of British Columbia
- 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"
  }]
}
```

UCAN

Self-Sovereign Identity

- W3C
- Microsoft
- Government of British Columbia
- 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"
  }]
}
```

UCAN

Self-Sovereign Identity

- W3C
- Microsoft
- Government of British Columbia
- 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"
  }]
}
```

UCAN

Object Capability Model (OCAP)

UCAN

Object Capability Model (OCAP)

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

UCAN

Object Capability Model (OCAP)

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

UCAN

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
  - Rights to anything directly created (parenthood)
  - The right to delegate subset of access to another (introduction)

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
  - 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)

UCAN

3rd-party Redelegation & Attenuation

UCAN

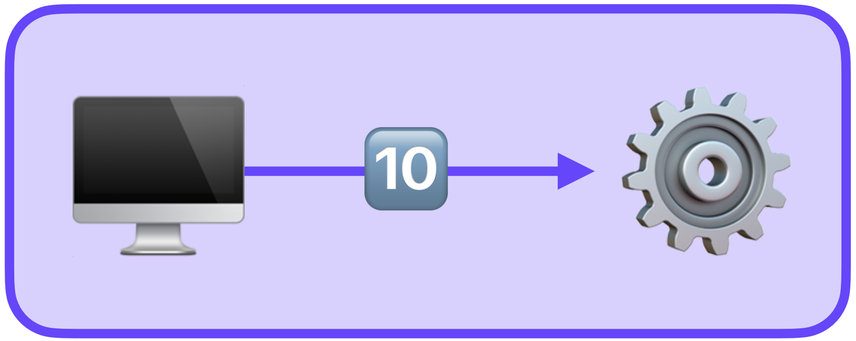
3rd-party Redelegation & Attenuation

UCAN

3rd-party Redelegation & Attenuation

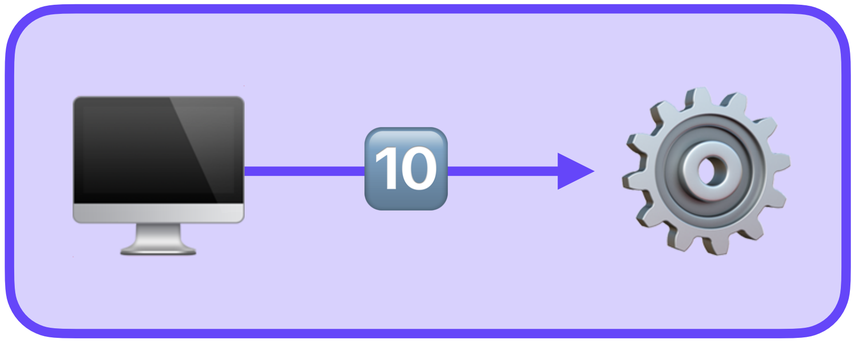
UCAN

3rd-party Redelegation & Attenuation



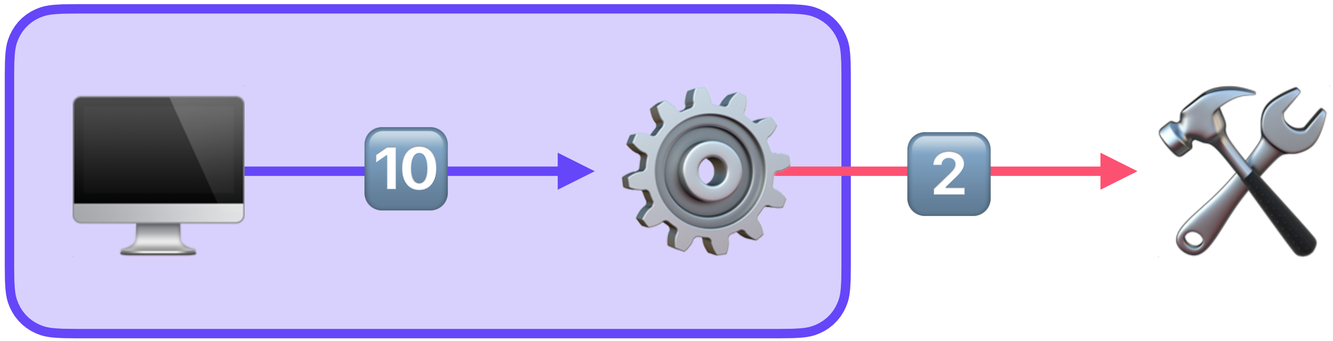
UCAN

3rd-party Redelegation & Attenuation



UCAN

3rd-party Redelegation & Attenuation



UCAN

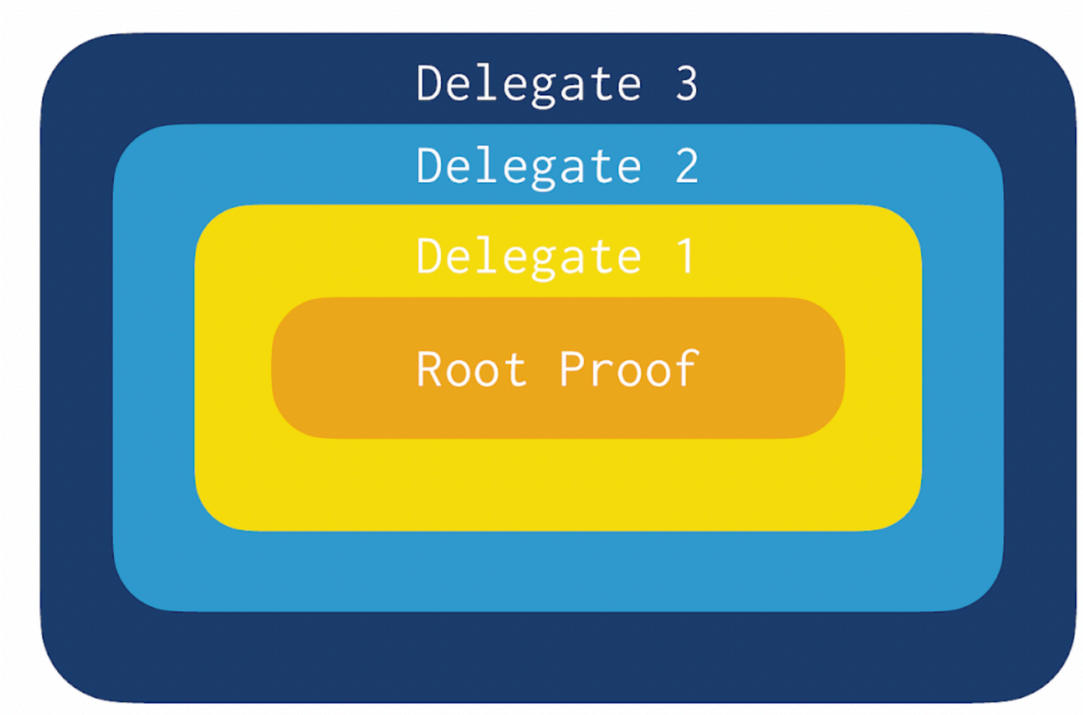
JWT

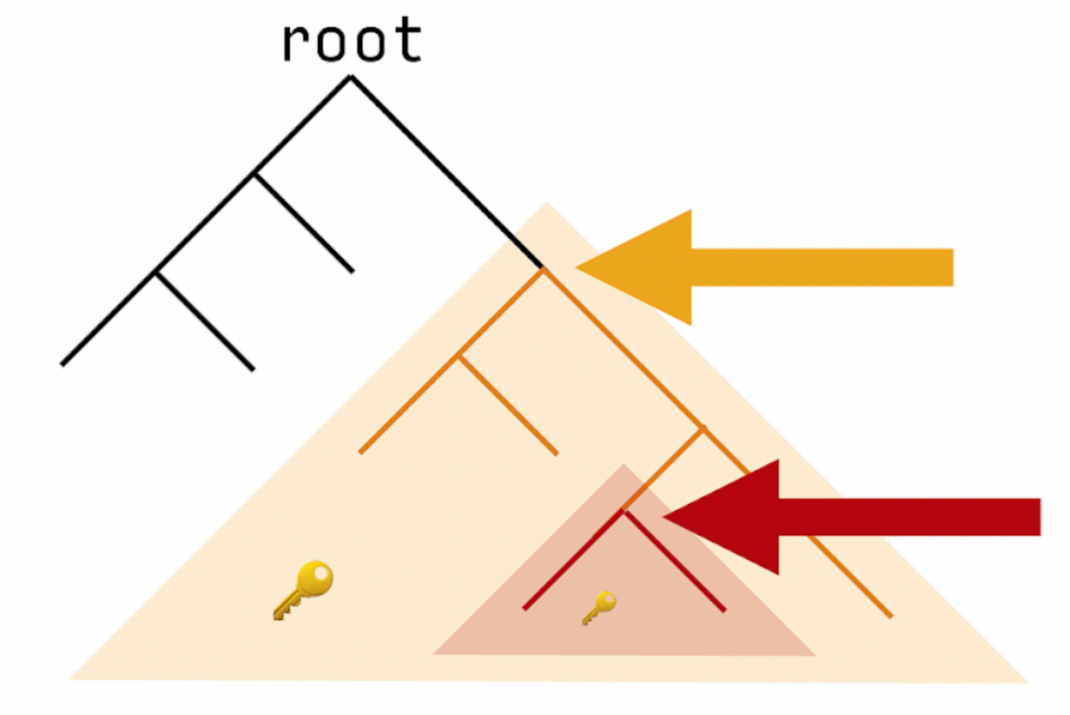

```
{
  "alg": "EdDSA",
  "typ": "JWT"
  "ucv": "0.5.0"
}
{
  "aud": "did:key:zStEZpzSMtTt9k2vszgvCwF4fLQQSyA15W5AQ4z3AR6Bx4eFJ5crJFbuGxKmbma4",
  "iss": "did:key:z5C4fuP2DDJChhMBCwAkpYUMuJZdNWWH5NeYjUyY8btYfzDh3aHwT5picHr9Ttjq",

  "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
```

UCAN

Auth Chaining

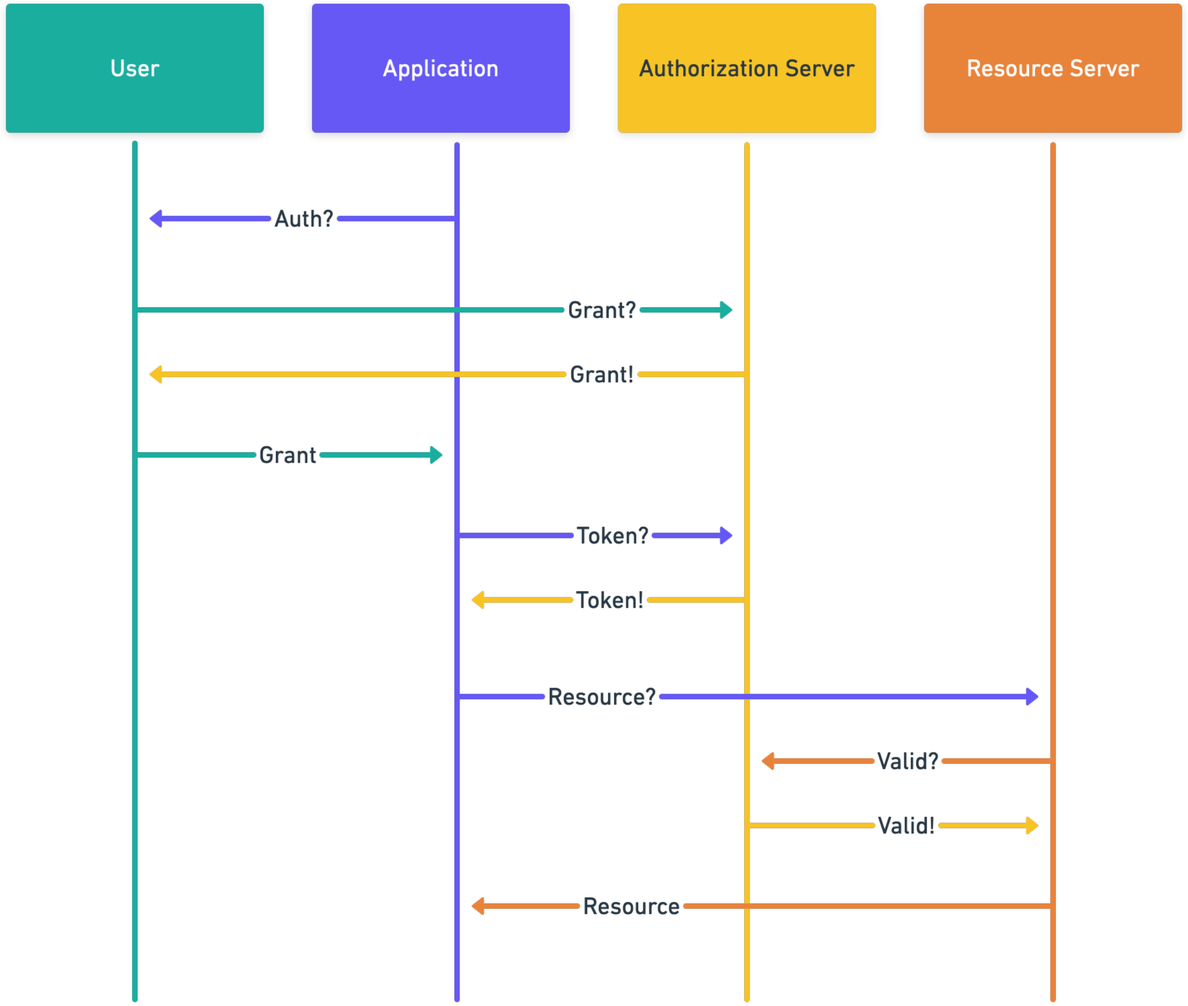
- OCAP, provable chains, revocable
- Non-exportable 2048-bit RSA (WebCrypto), Ed25519 & BLS everywhere else

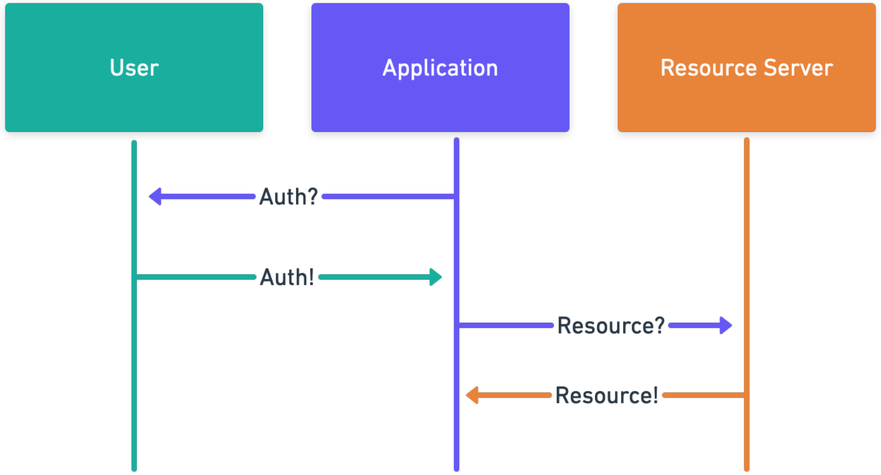




UCAN

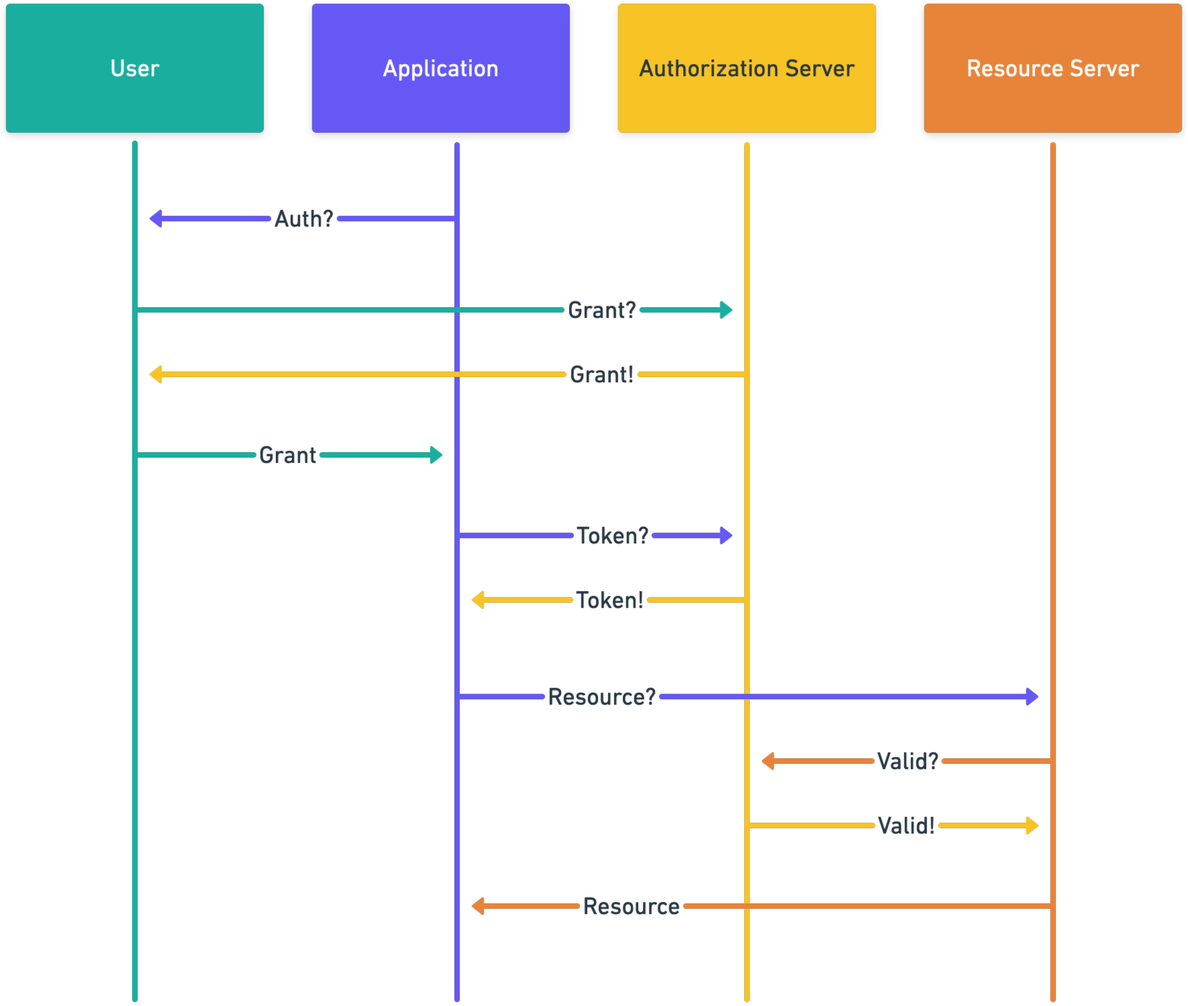
OAuth vs UCAN Sequence

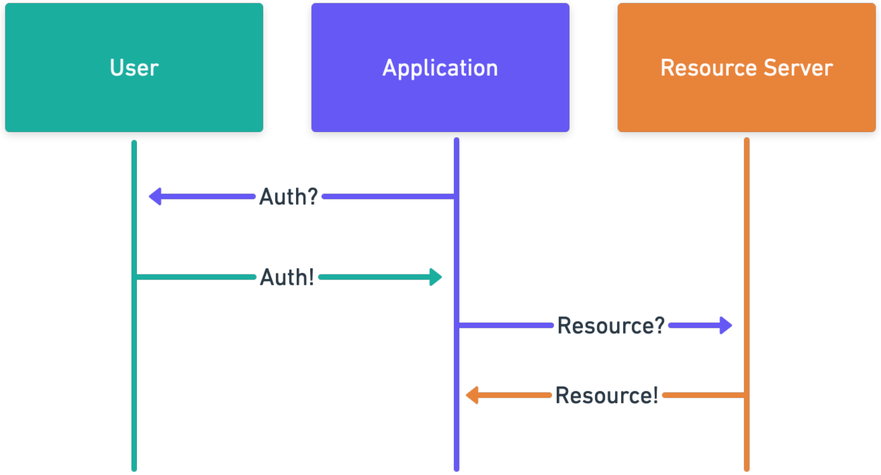




UCAN

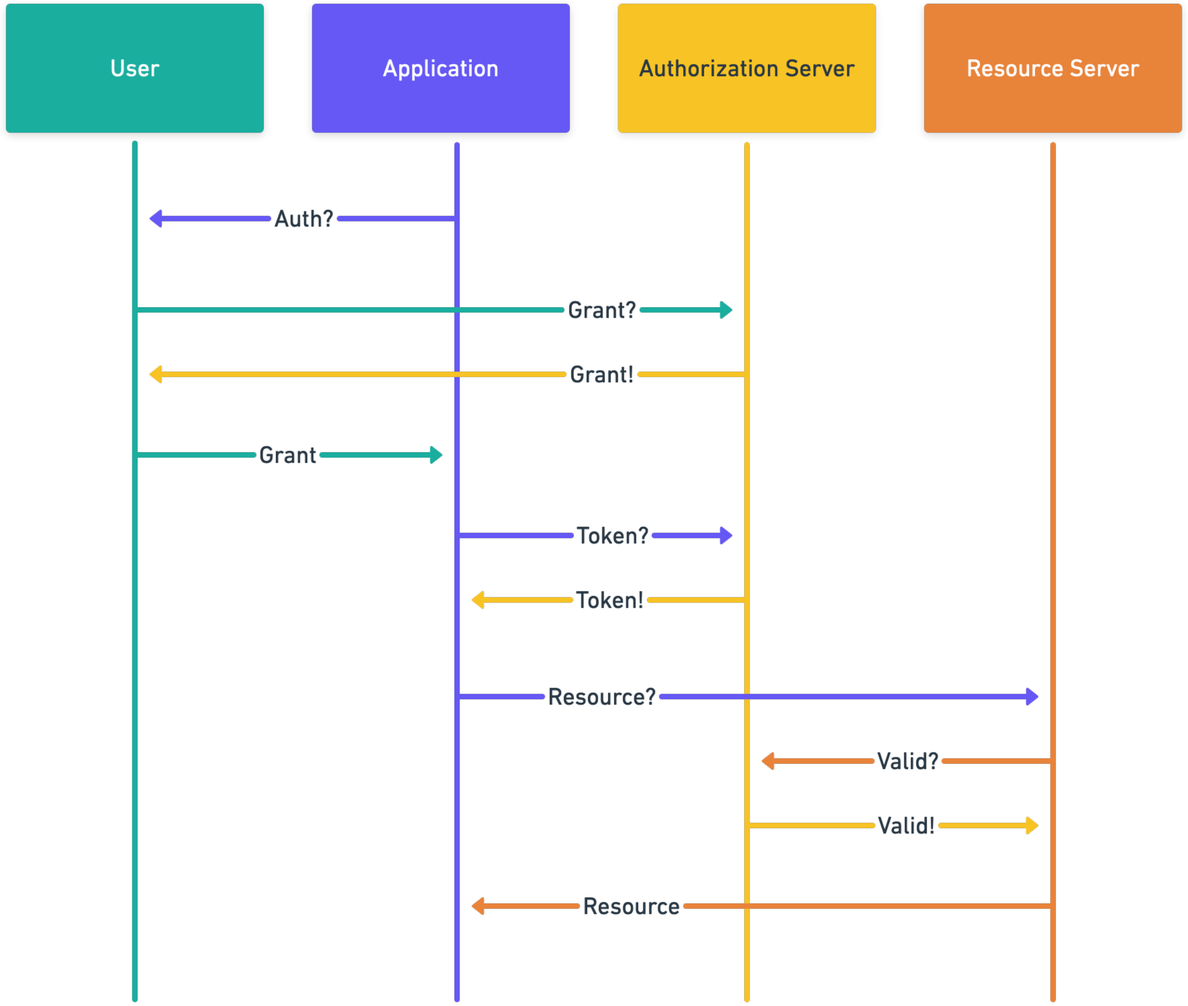
OAuth vs UCAN Sequence

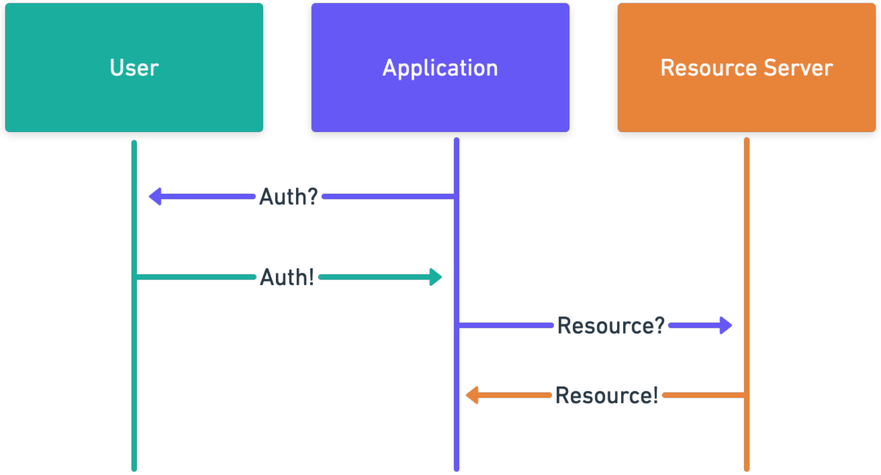




UCAN

OAuth vs UCAN Sequence





(Verifiable & user originated)

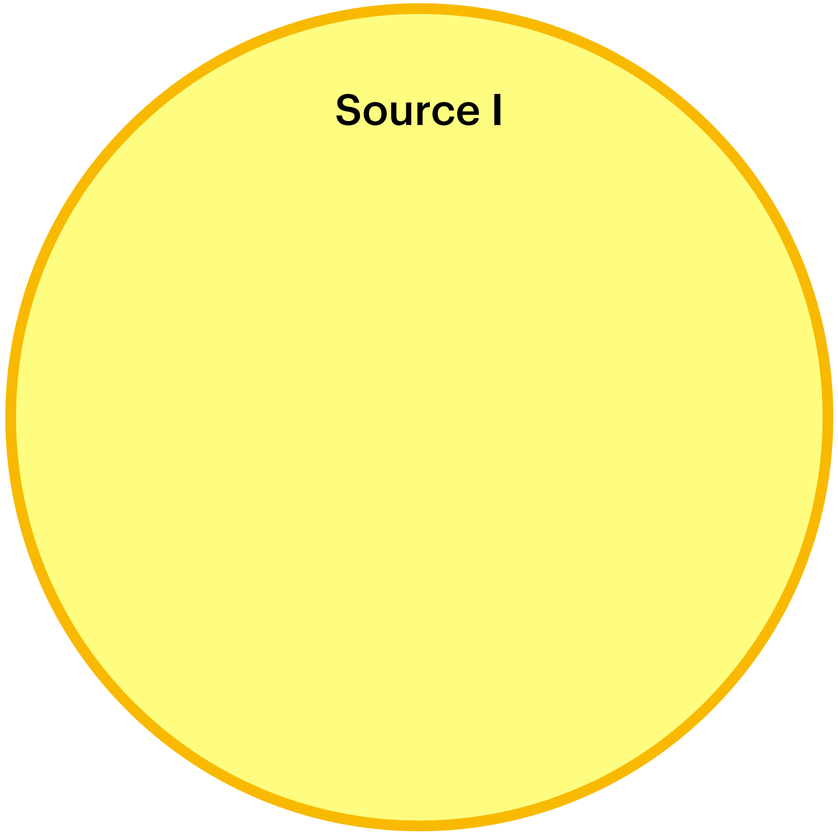
Final Thoughts

Final Thoughts

More Coming — Embarrassingly Distributed Deductive Database

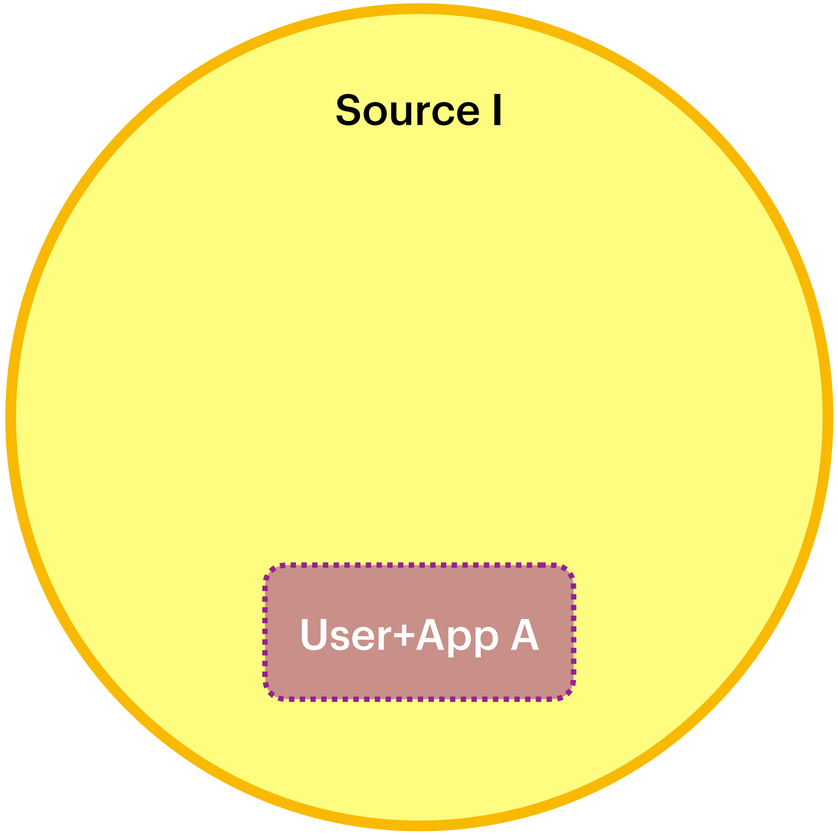
Final Thoughts

More Coming — Embarrassingly Distributed Deductive Database



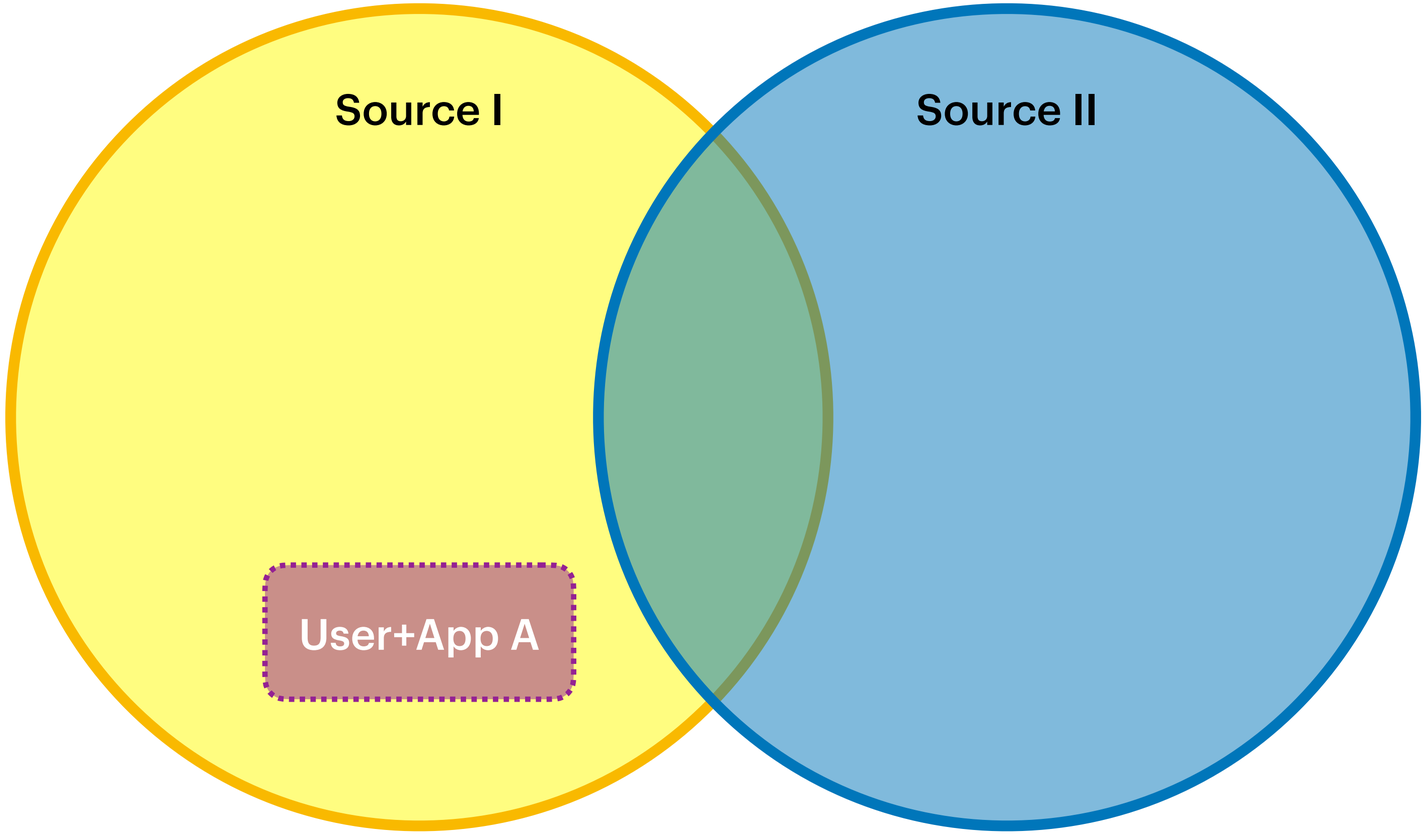
Final Thoughts

More Coming — Embarrassingly Distributed Deductive Database



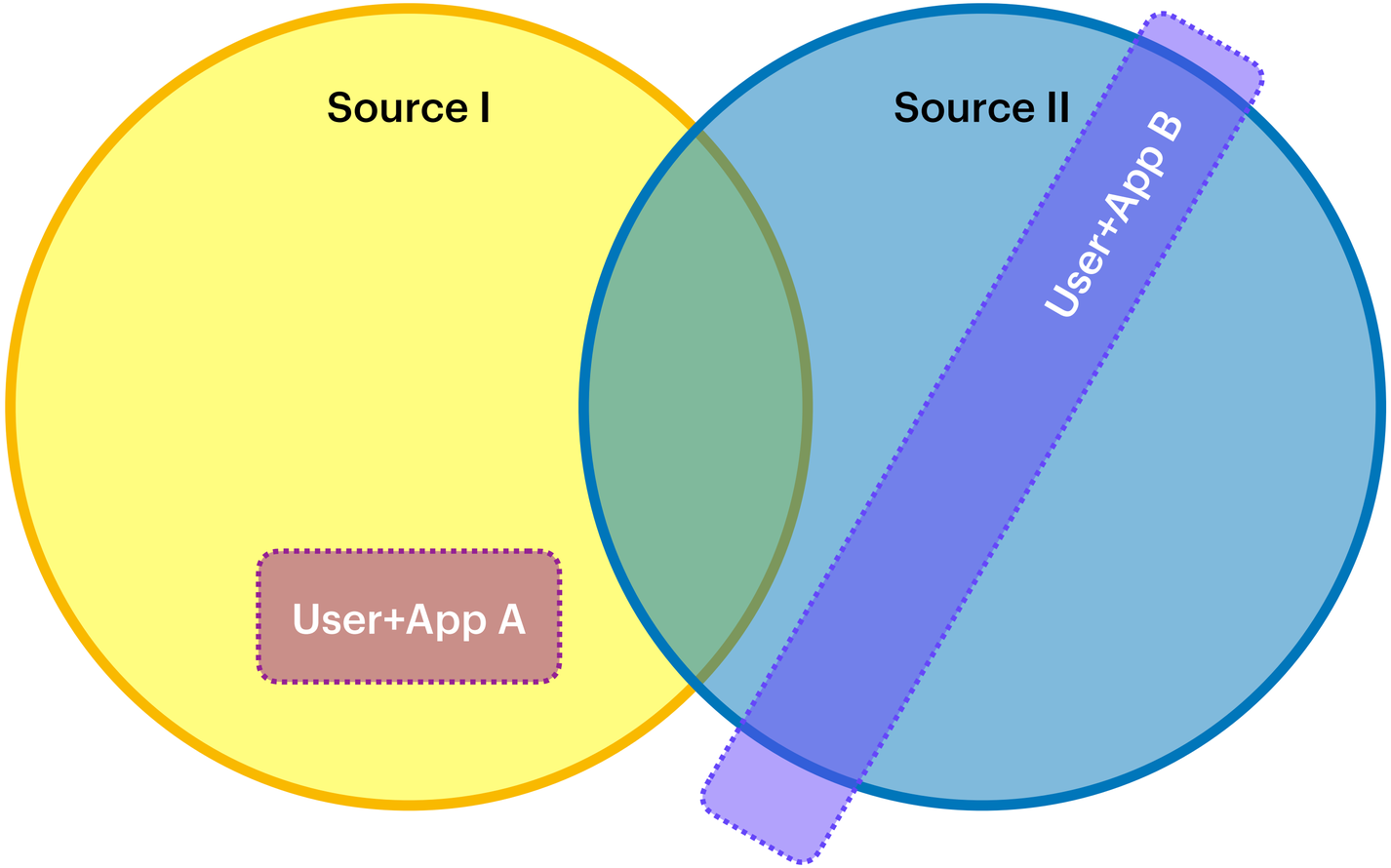
Final Thoughts

More Coming — Embarrassingly Distributed Deductive Database



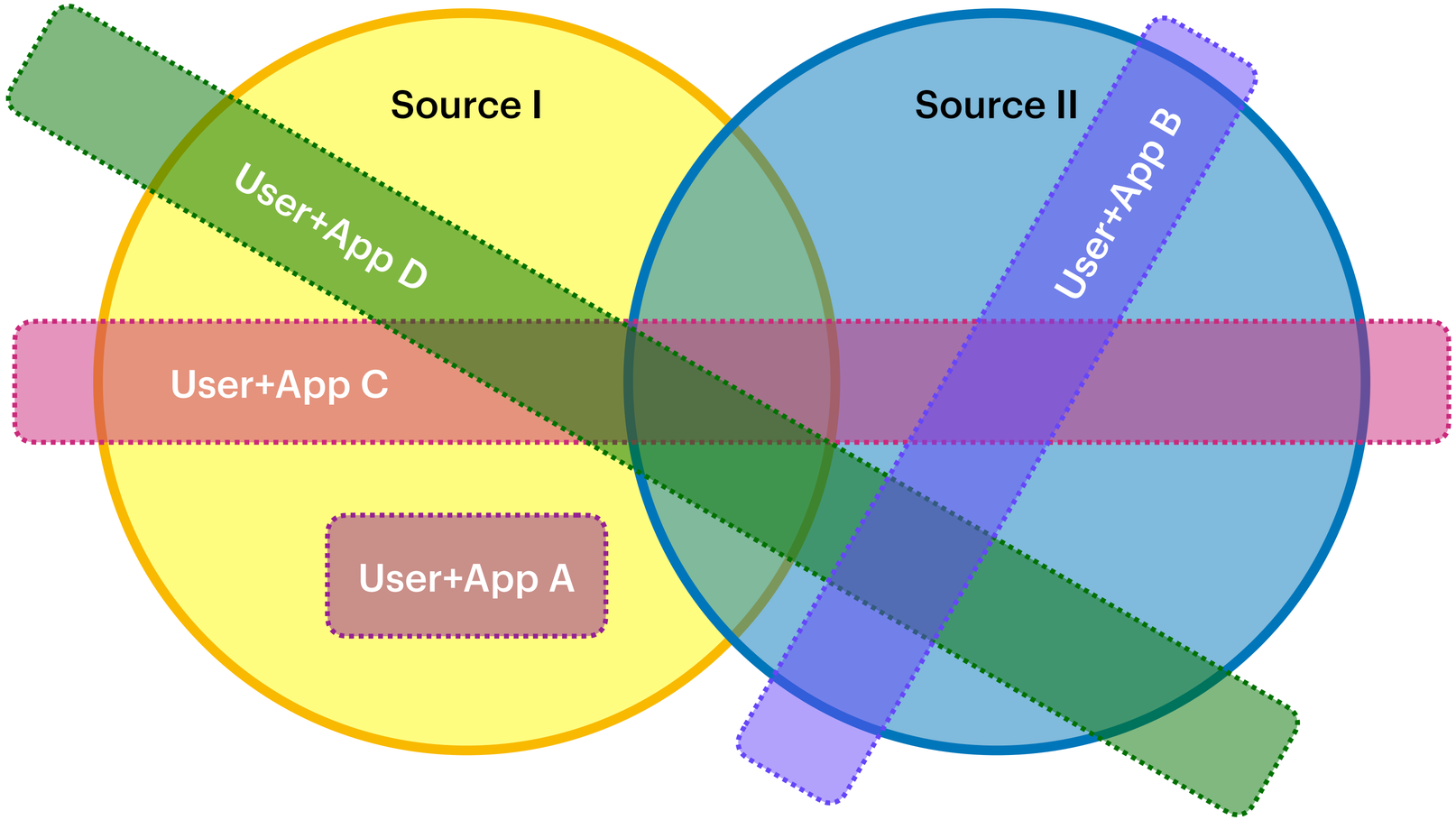
Final Thoughts

More Coming — Embarrassingly Distributed Deductive Database



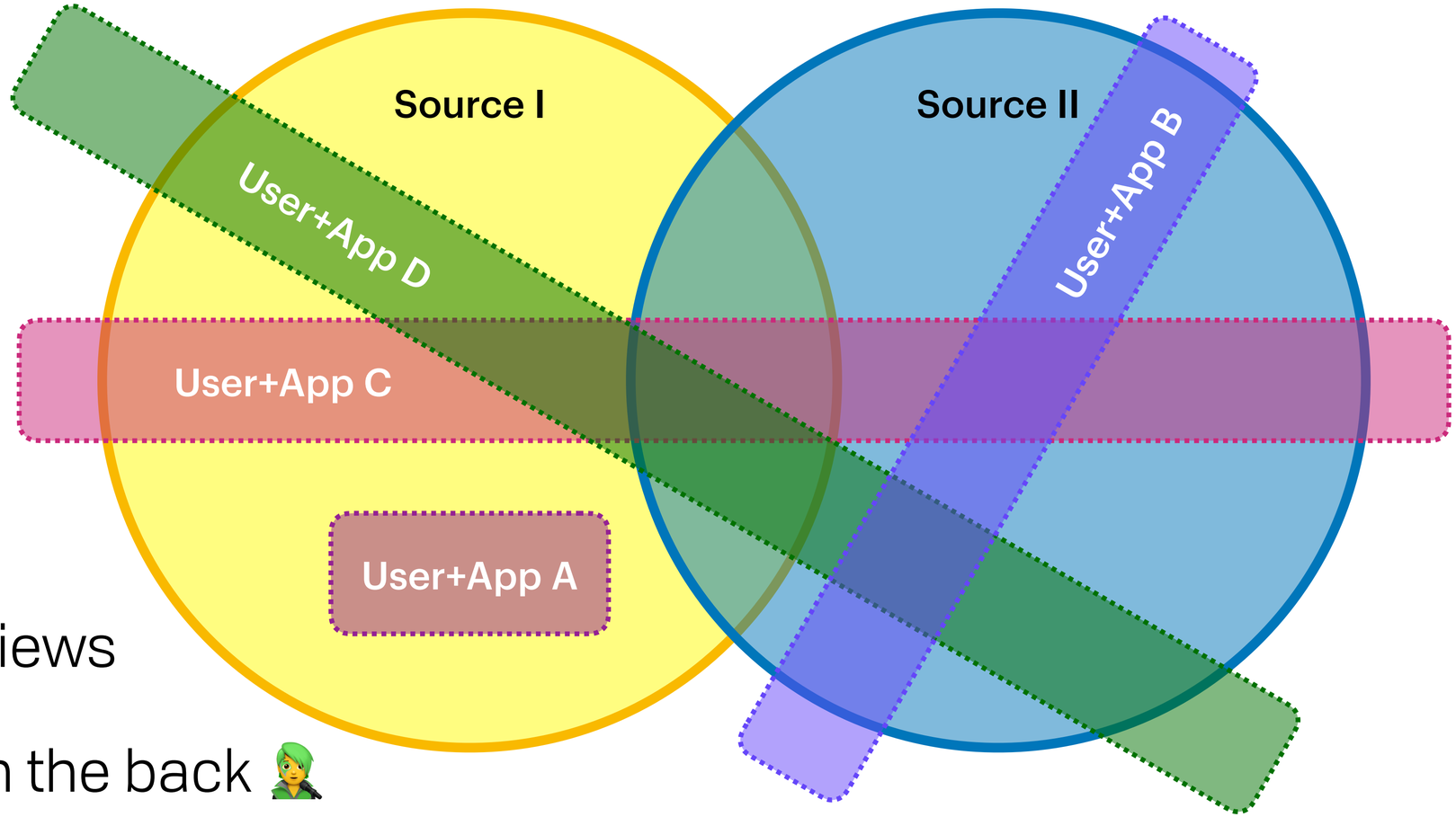
Final Thoughts

More Coming — Embarrassingly Distributed Deductive Database



Final Thoughts

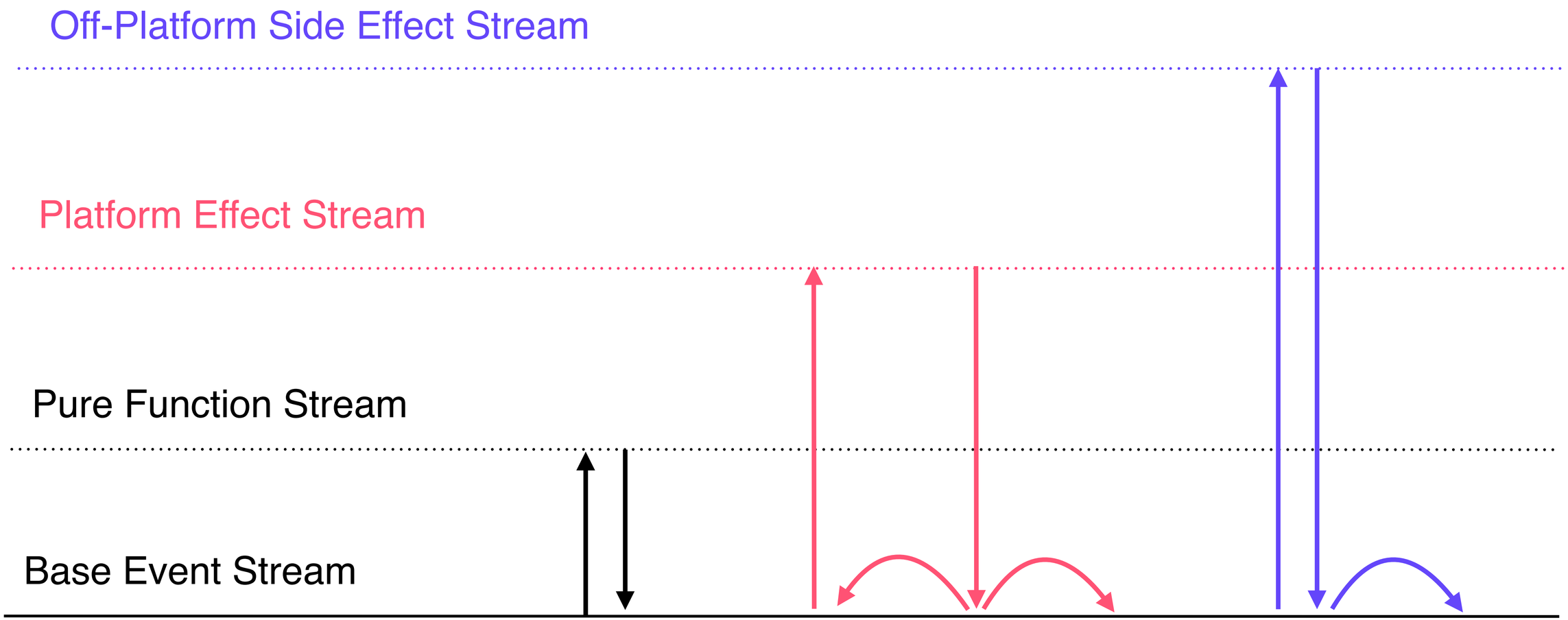
More Coming — Embarrassingly Distributed Deductive Database



- Assert, refute, time, source
- Merge / split easily
- Access control = different views
- JSON in the front, Datalog in the back 🧝‍♀️
- Help define API? **calendly.com/walkah**

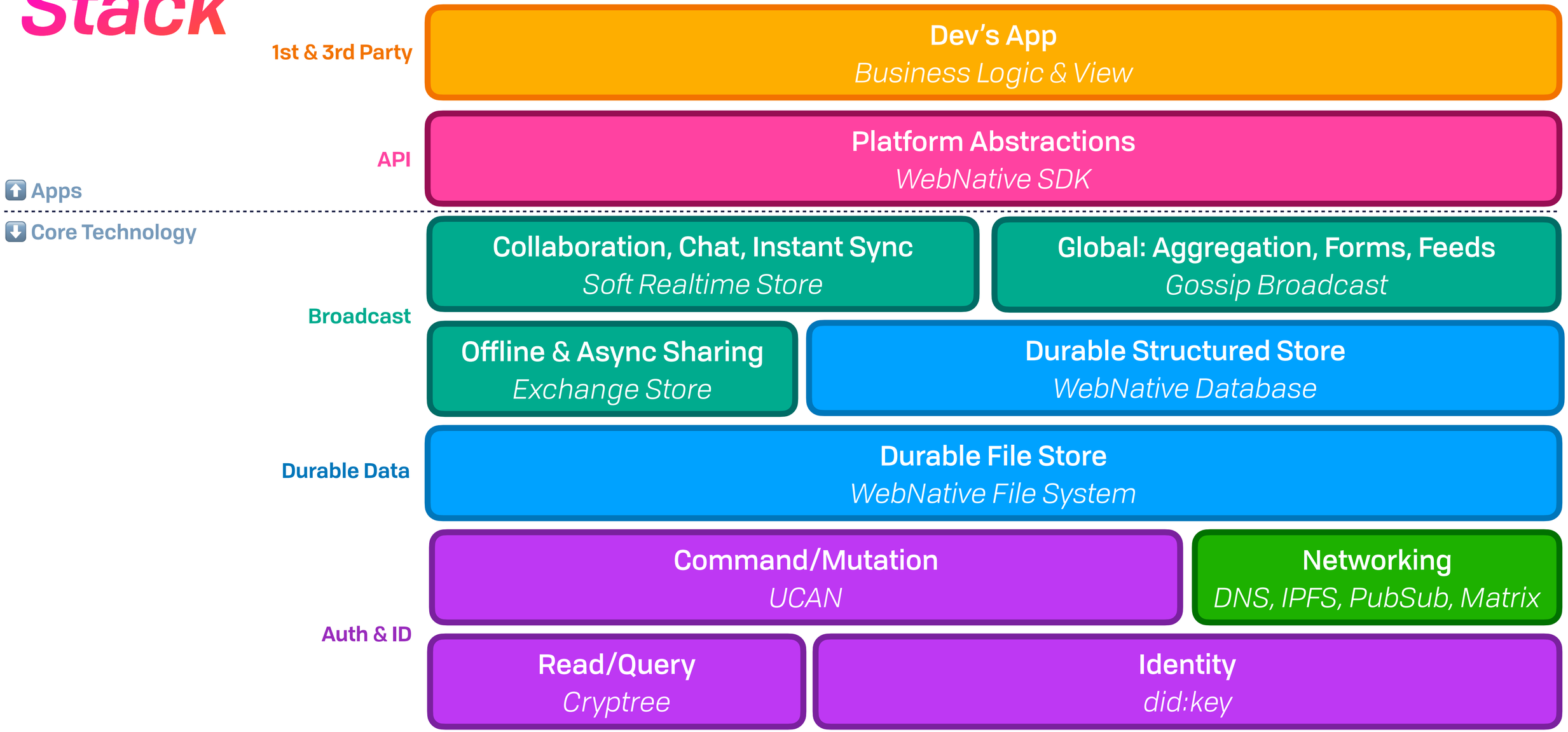
Final Thoughts

More Coming — Universal Distributed Compute



Final Thoughts

Stack

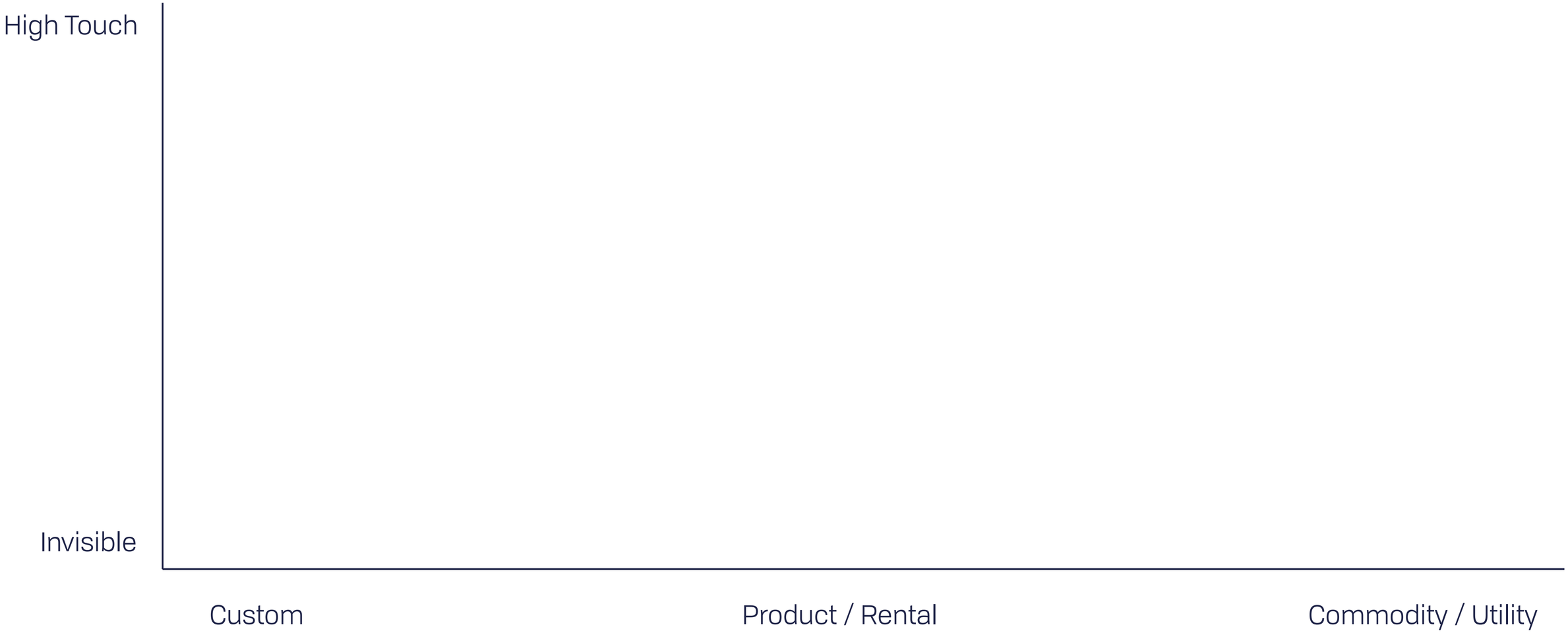


Final Thoughts

60+ Year Trend

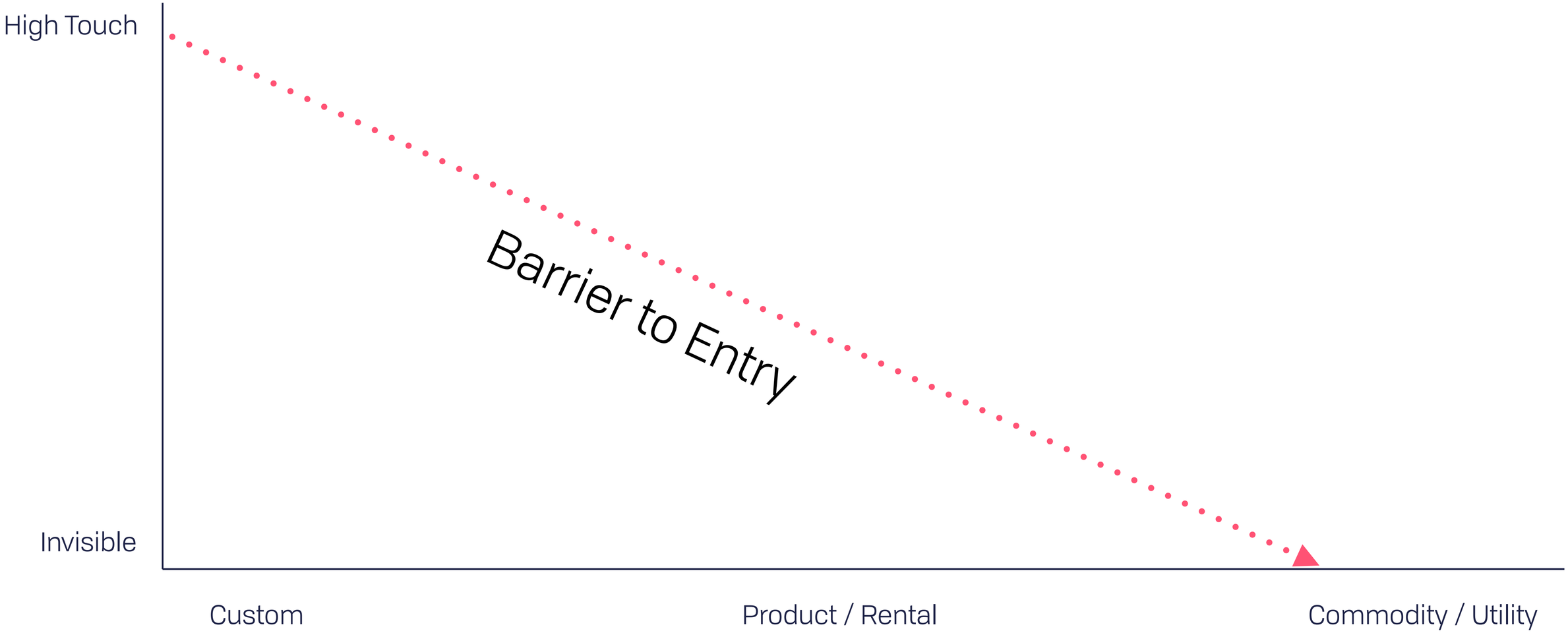
Final Thoughts

60+ Year Trend



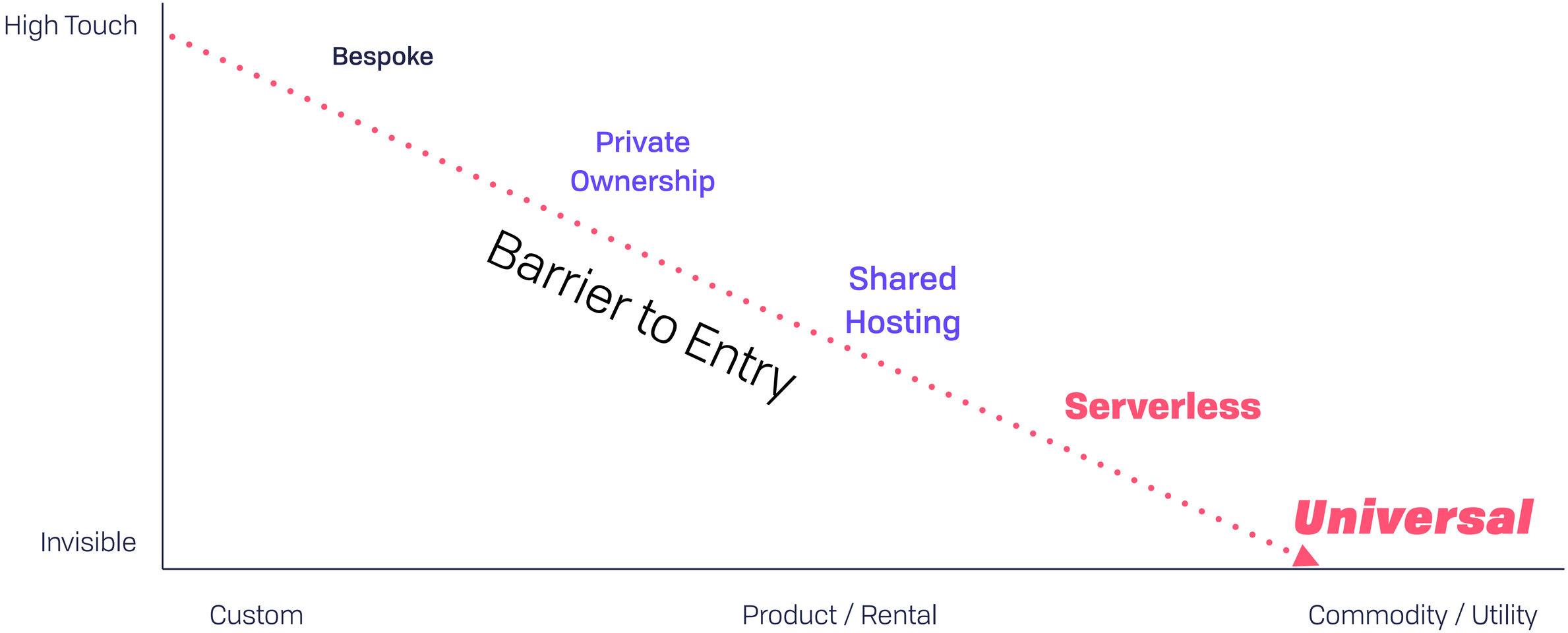
Final Thoughts

60+ Year Trend



Final Thoughts

60+ Year Trend



https://fission.codes
https://guide.fission.codes
https://discord.gg/zAQBDEq

😄 Thank You, Speakeasy JS 🎉

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

shop.fission.codes
Code: SPEAKEASYJS