# Functional Programming for (Dev)Ops and Infra

## … What is the Business Value?

Cornelia Davis
VP of Technology
Pivotal

@cdavisafc

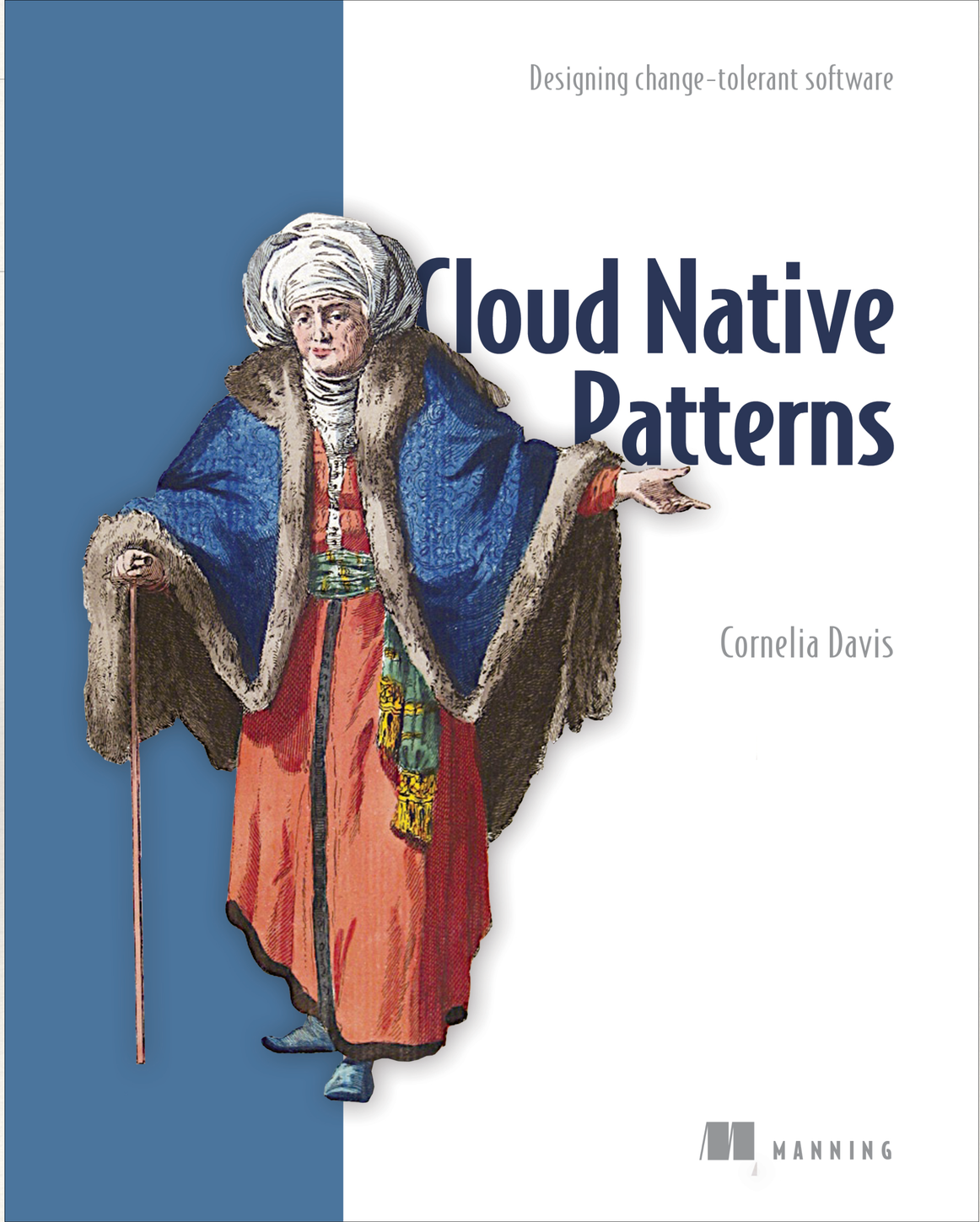
# Me?

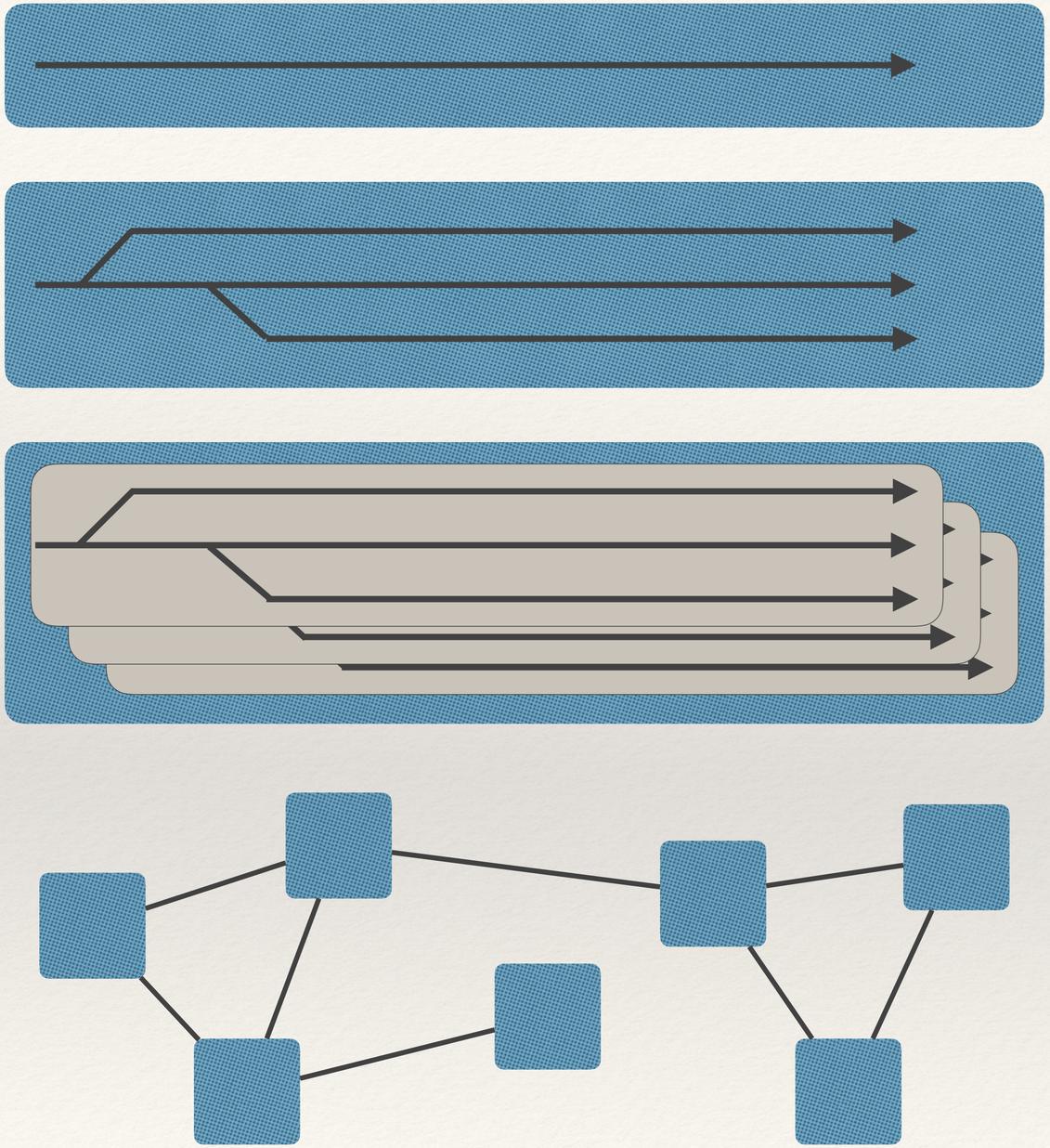
- Cal State Northridge: Computer Science
- Indiana University: Computer Science (ABD)
- Pivotal: Product Strategy (PCF, PCC, PKS)

**Ops**
as much as
**Dev**
now

- And now…

Discount code 40% off!: ctwdevopsent19





Assembly Language

Fortran

C

C++

Java

Golang

Imperative Programming

Clojure

Scala

Kotlin

F#

Functional Programming

# Hallmarks of Imperative Programs

- Sequential - we control every step in the process.
- Variables … with side effects

- Hairy edge cases
- Difficult to parallelize

# Hallmarks of Functional Programs

- Declarative
- NO side effects
- Recursive

- Easier to program in a way that allows for parallelization
- Far fewer edge cases
- Provably correct

What is the

**model**

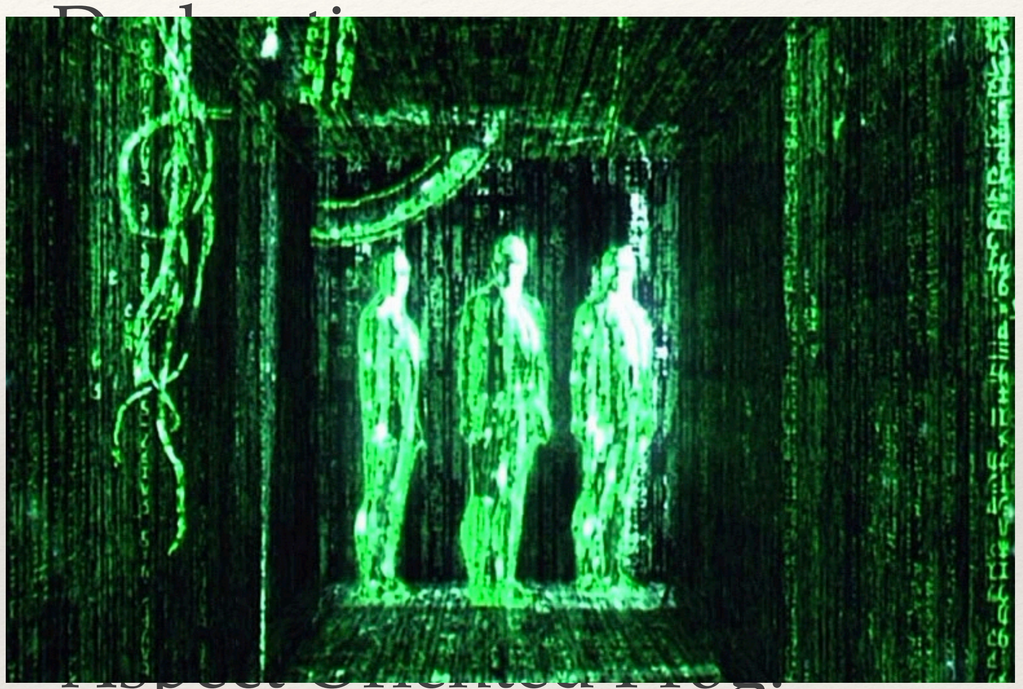
we use to reason with?

# Imperative → Functional Programming

| | | |
|---|---|---|
| Iterative | → | Declarative |
| Mutable State | → | Immutability |
| Looping | → | Recursive |
| Libraries (compiled in) | → | Aspect Oriented Prog.* |

* Maybe not functional, but modern programming practice

# Imperative → Functional Programming



We cannot reason about every detail so we need a model
that allows machines to reason for us.

What do I mean by
machines doing reasoning
for us?

```
val a = List(23, 27, 1, 25, 40, 5);

val q0 = a.filter(lessThan30);
```

(23, 27, 1, 25, 5)

```
val q1 = q0.filter(moreThan20);
```

(23, 27, 25)

```
q1(2);
```

27

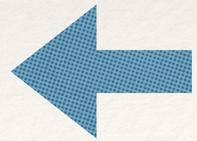
```
val a = List(23, 27, 1, 25, 40, 5);
val q0 = a.filter(lessThan30);
val q1 = q0.filter(moreThan20);

q1(2);
```

23 < 30? ✓

23 > 20? ✓ → q1(1) = 23

27 < 30? ✓

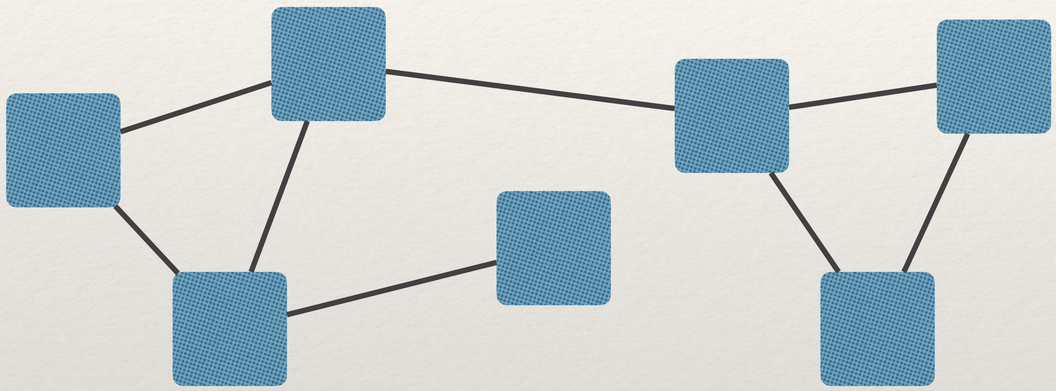
27 > 20? ✓ → q1(2) = 27



# Business Value?

Process Big Data Faster ⟶ First to Market

# Okay, so how does this relate to Ops and Infra?



# Imperative Systems Programming

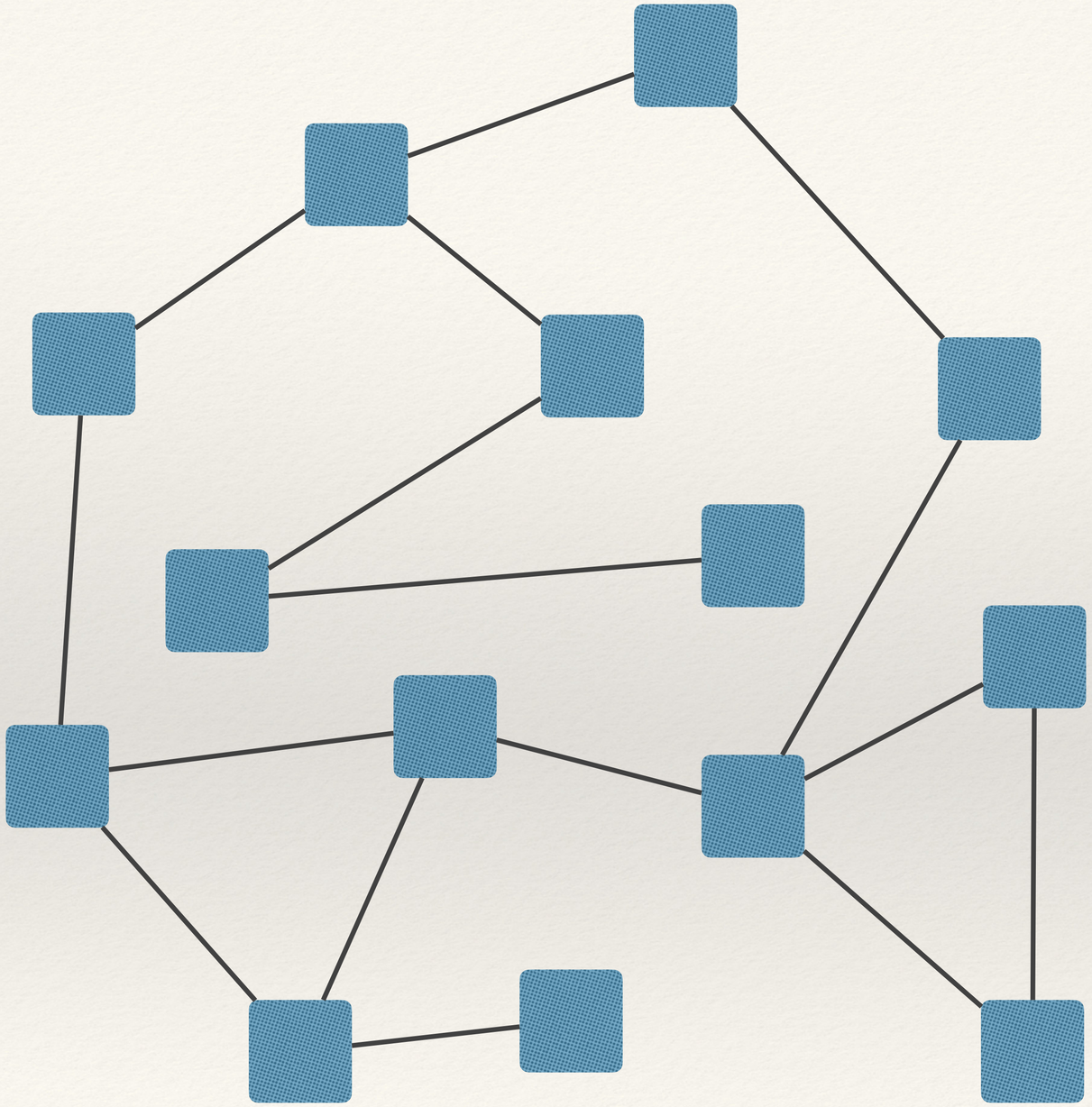
Bash

Puppet

Chef

Ansible

Salt



# Functional Systems Programming

CF/PAS

Kubernetes

BOSH

# Imperative → Functional

Systems Programming

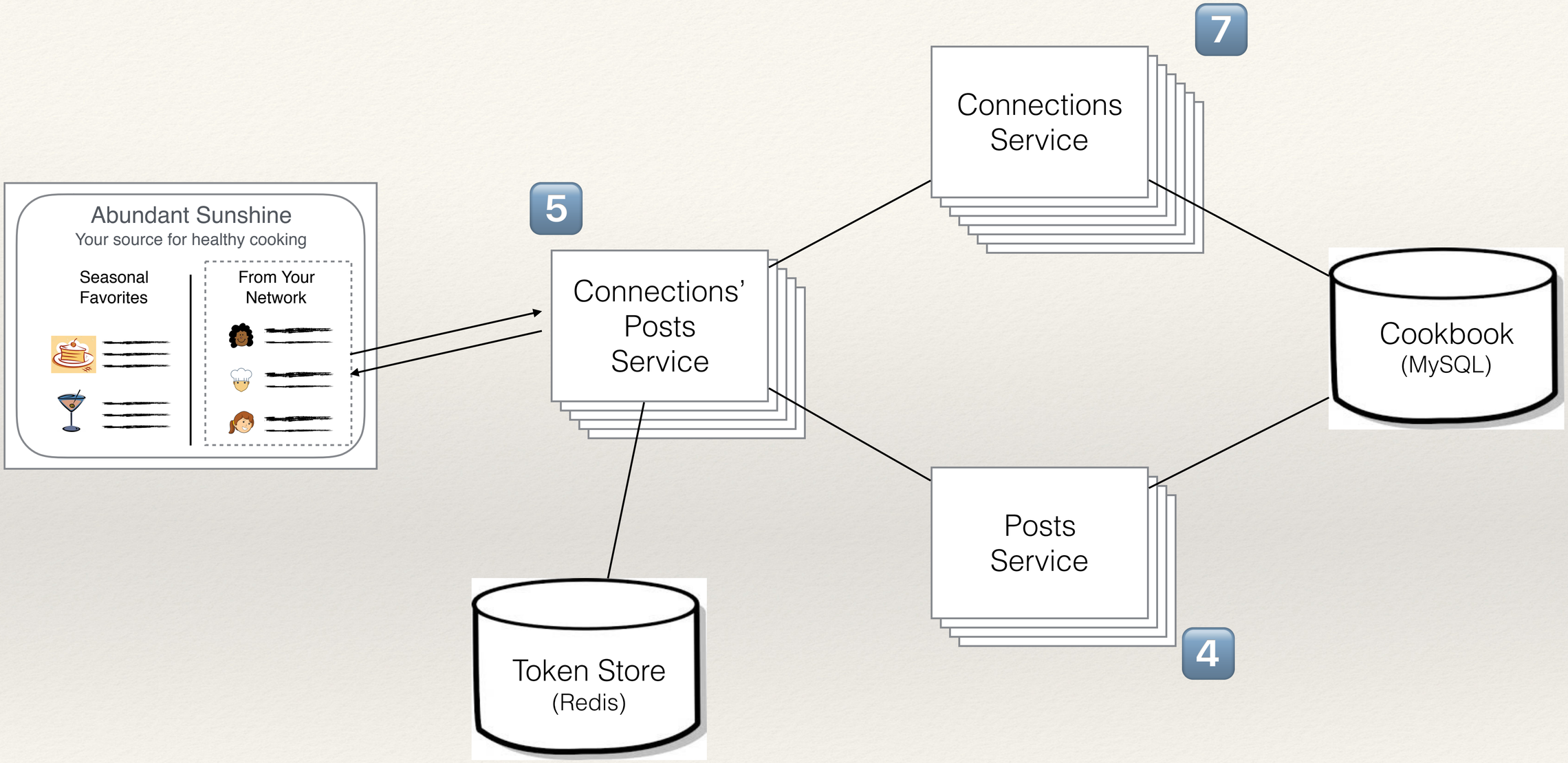
| | | |
|---|---|---|
| Scripted Deployments | → | Declarative Deployments |
| ssh | → | Immutable Infrastructure |
| Long Running Context | → | (Ephemeral) Containers |
| Middleware | → | Sidecars |

We cannot reason about every detail so we need

**a model**

that allows machines to reason for us.

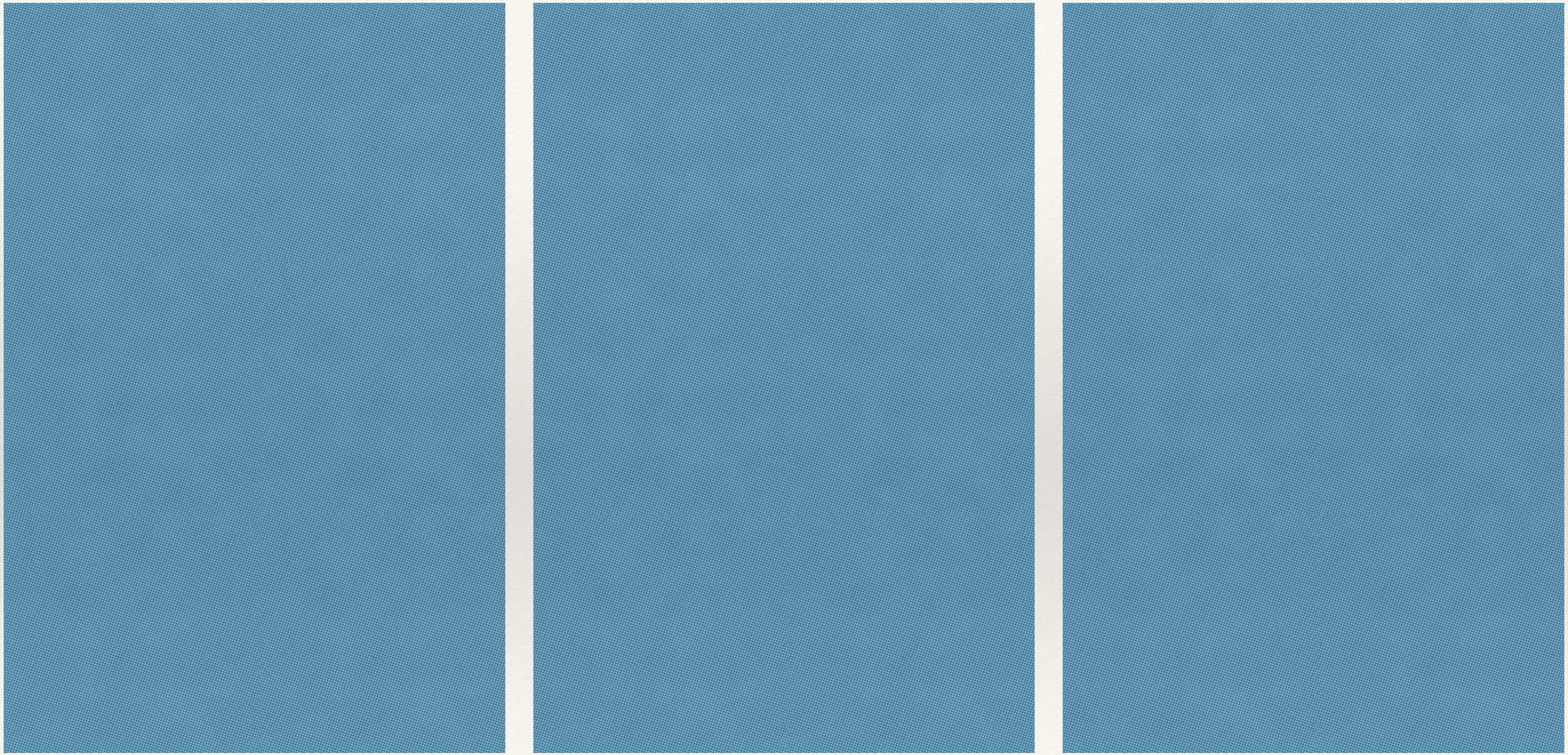
# Declarative Deployments

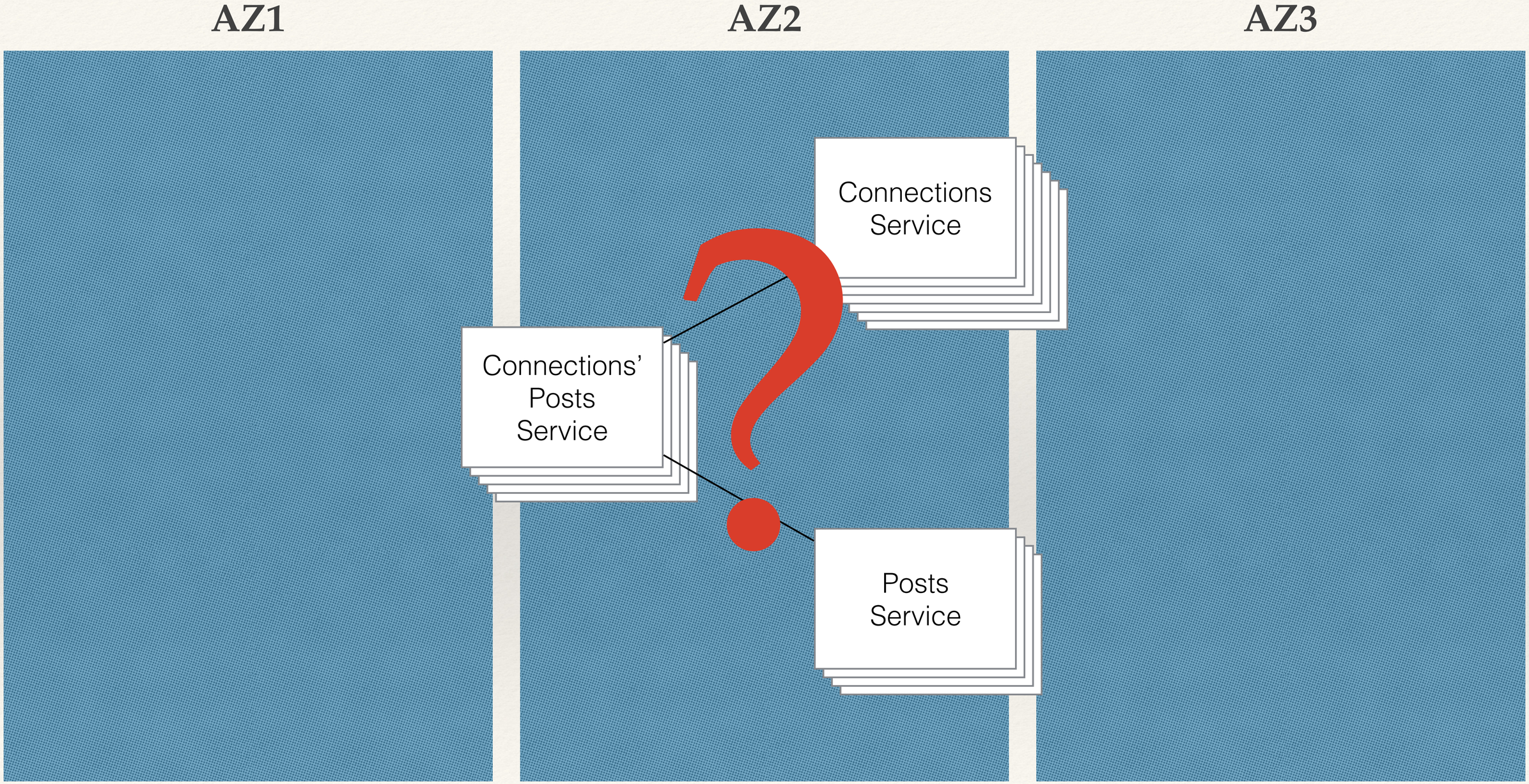

Abundant Sunshine
Your source for healthy cooking
Seasonal Favorites
From Your Network
5
Connections' Posts Service
Connections Service
7
Cookbook (MySQL)
Posts Service
4
Token Store (Redis)

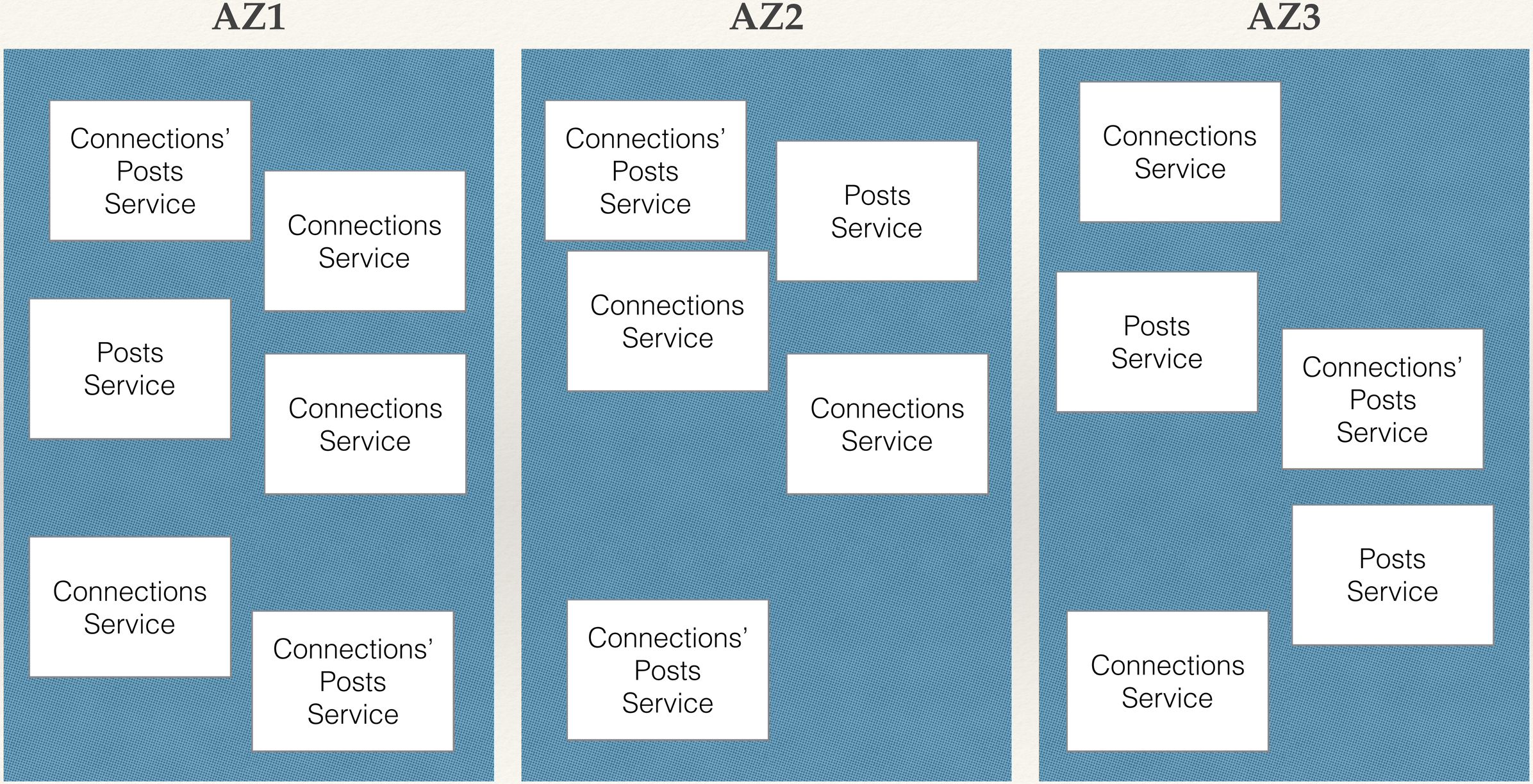
AZ1

AZ2

AZ3



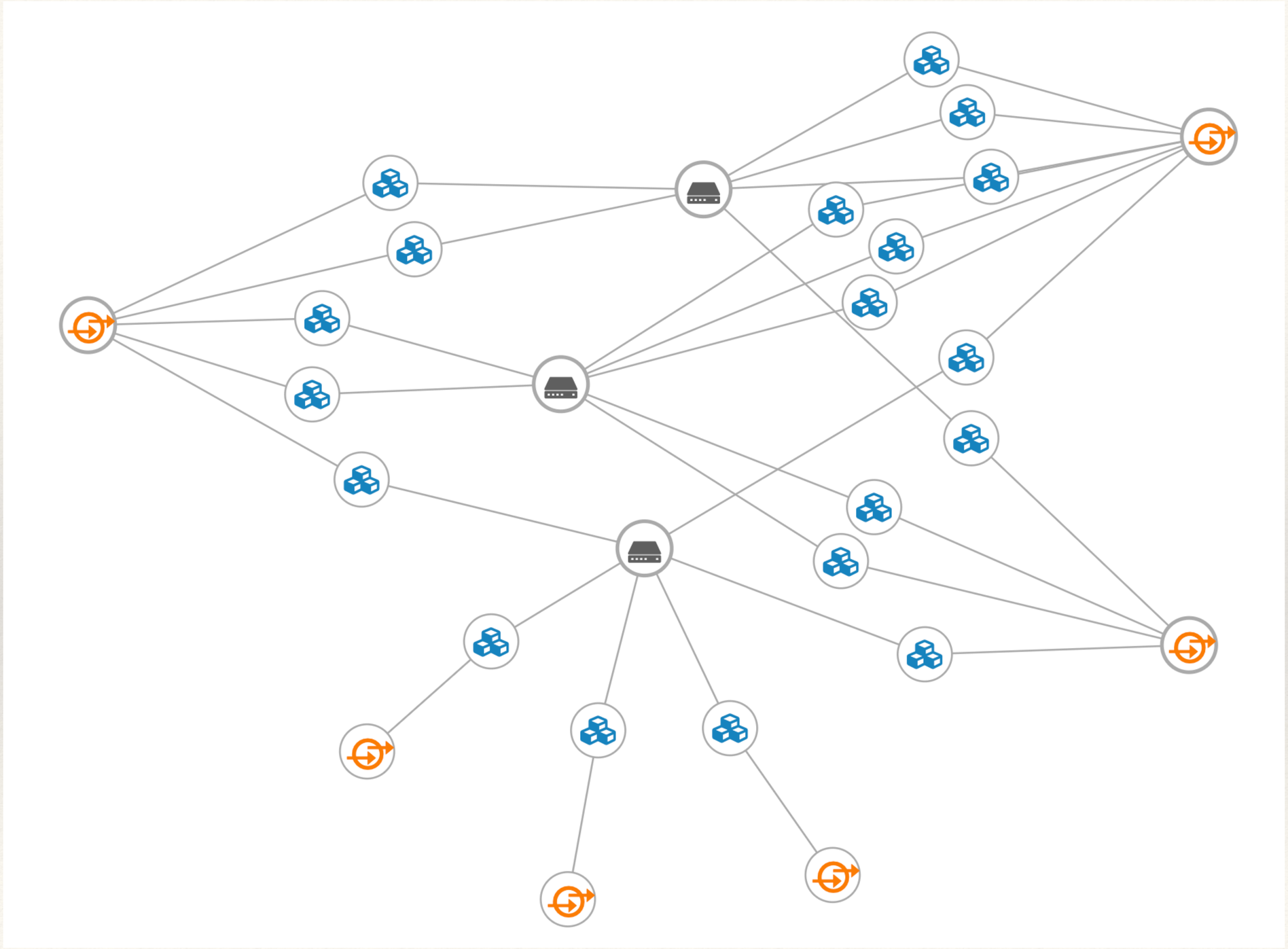

AZ1
AZ2
AZ3
Connections Service
Connections' Posts Service
Posts Service

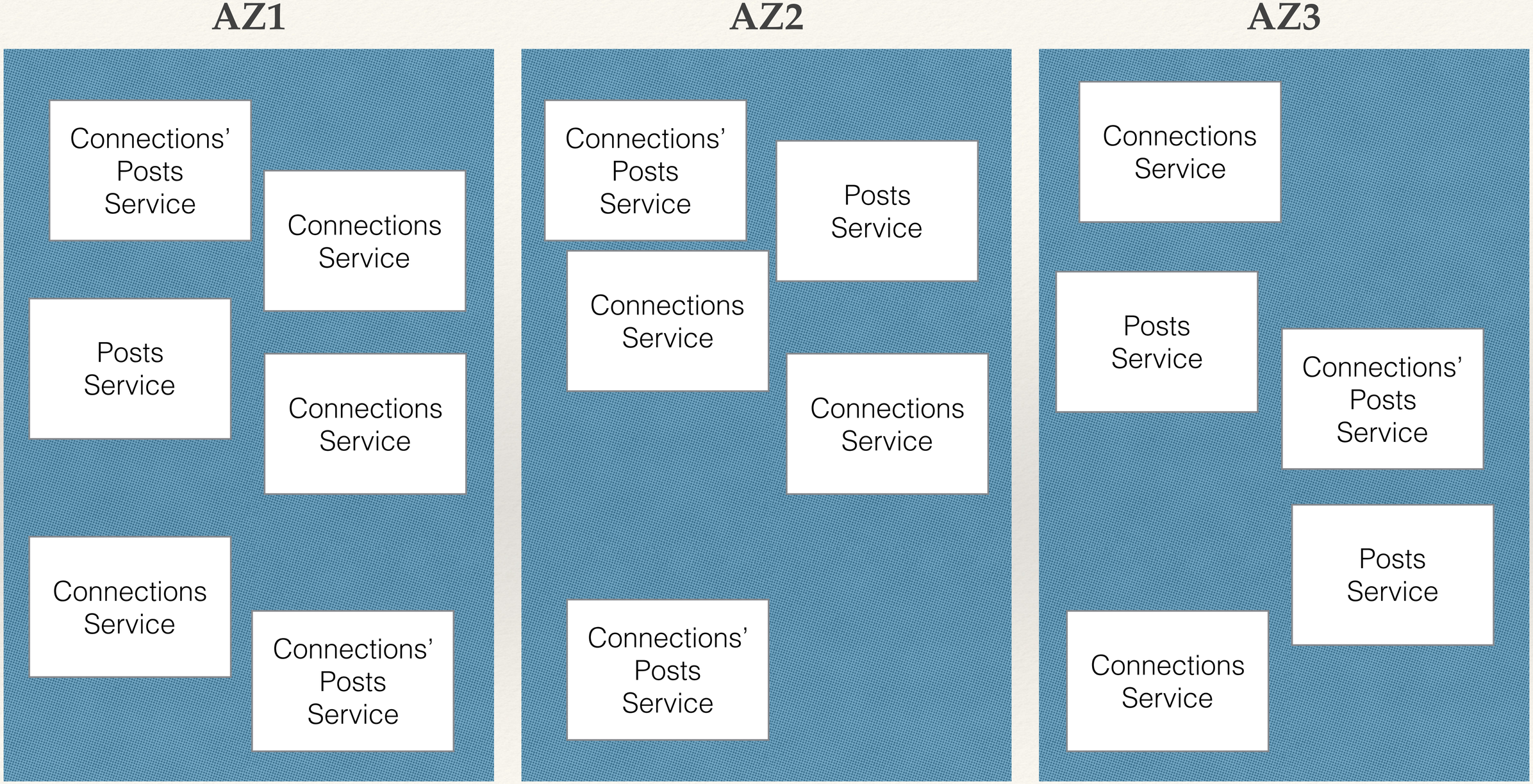

AZ1
Connections' Posts Service
Connections Service
Posts Service
Connections Service
Connections Service
Connections' Posts Service
AZ2
Connections' Posts Service
Posts Service
Connections Service
Connections Service
Connections' Posts Service
AZ3
Connections Service
Posts Service
Connections' Posts Service
Posts Service
Connections Service

# Are humans best-suited for this task?

(I say no)

Let’s let a machine
do that for us




AZ1
Connections' Posts Service
Connections Service
Posts Service
Connections Service
Connections Service
Connections' Posts Service
AZ2
Connections' Posts Service
Posts Service
Connections Service
Connections Service
Connections' Posts Service
AZ3
Connections Service
Posts Service
Connections' Posts Service
Posts Service
Connections Service

# Do humans get paged?

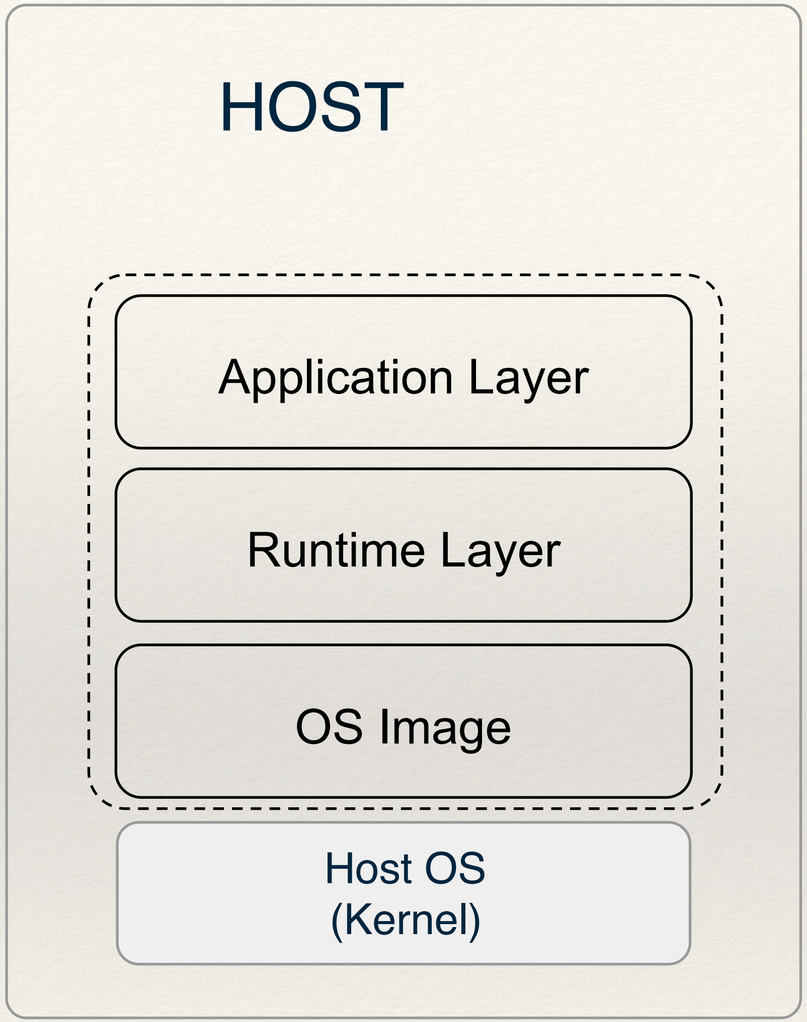
(maybe)

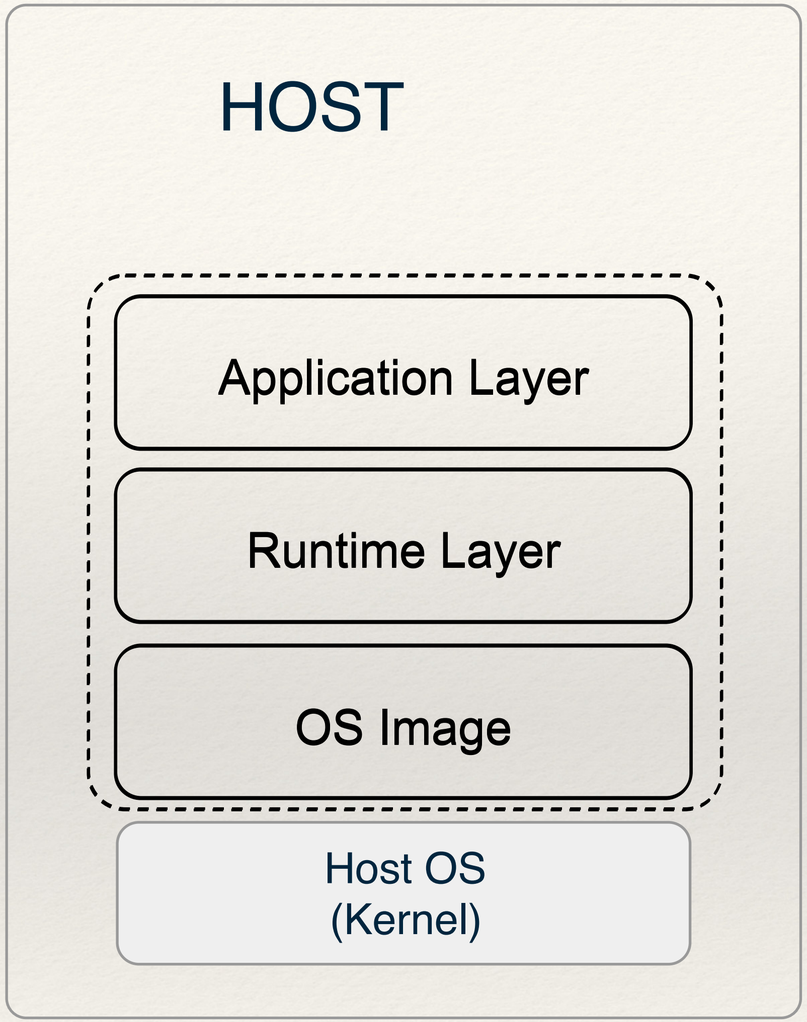
# Business Value

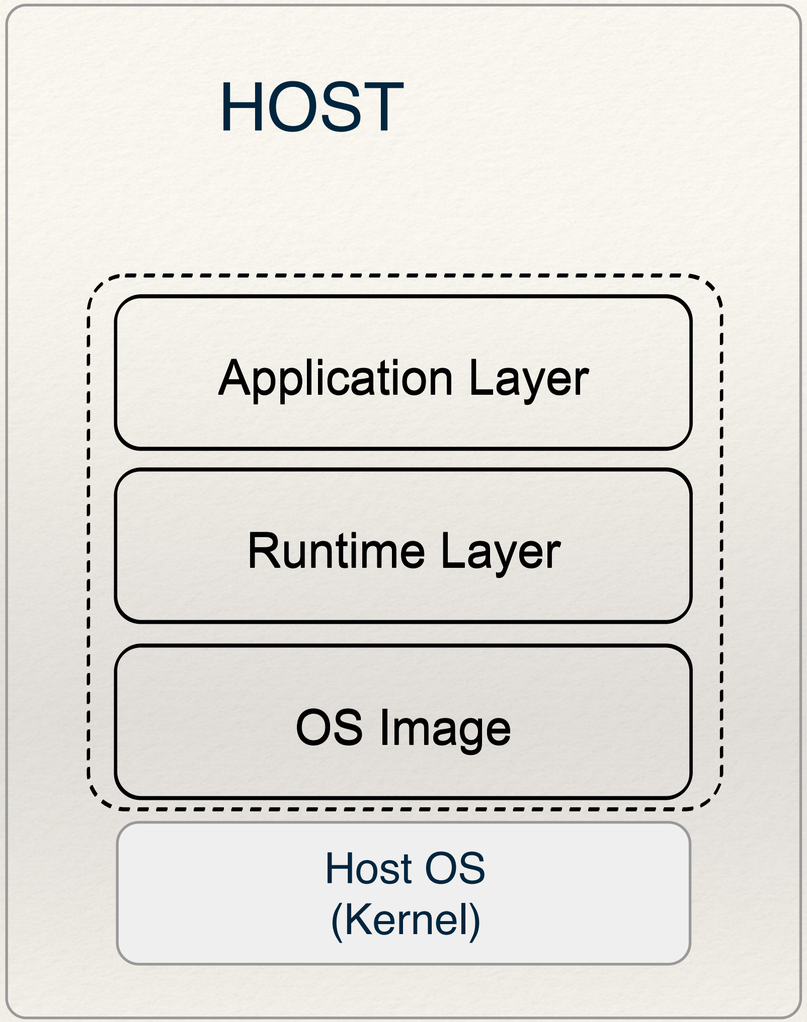
Resilient Systems → Happy Customers

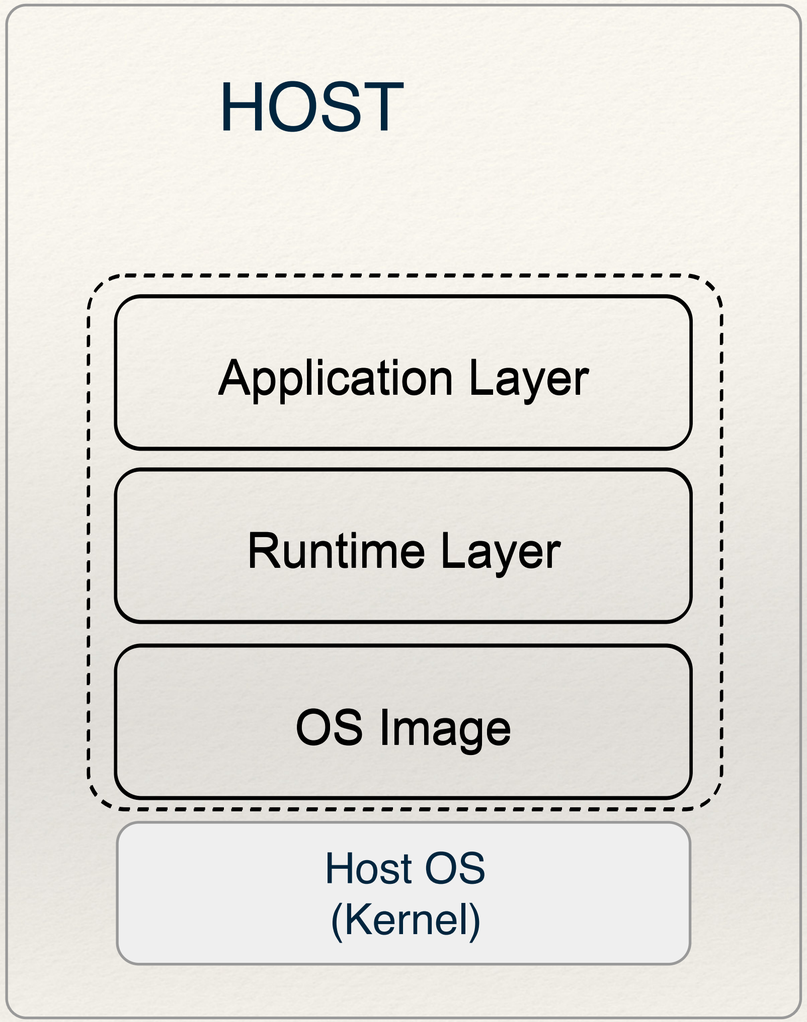
Less Stressed Staff → Engaged Employees

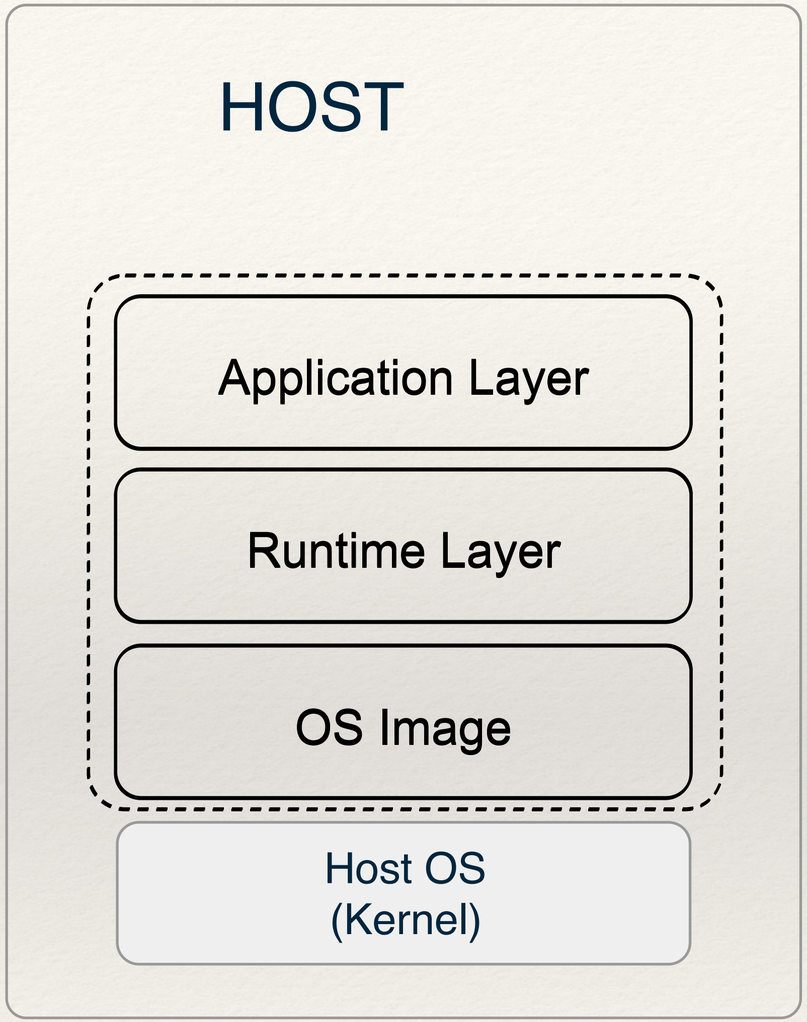
# Immutable Infrastructure

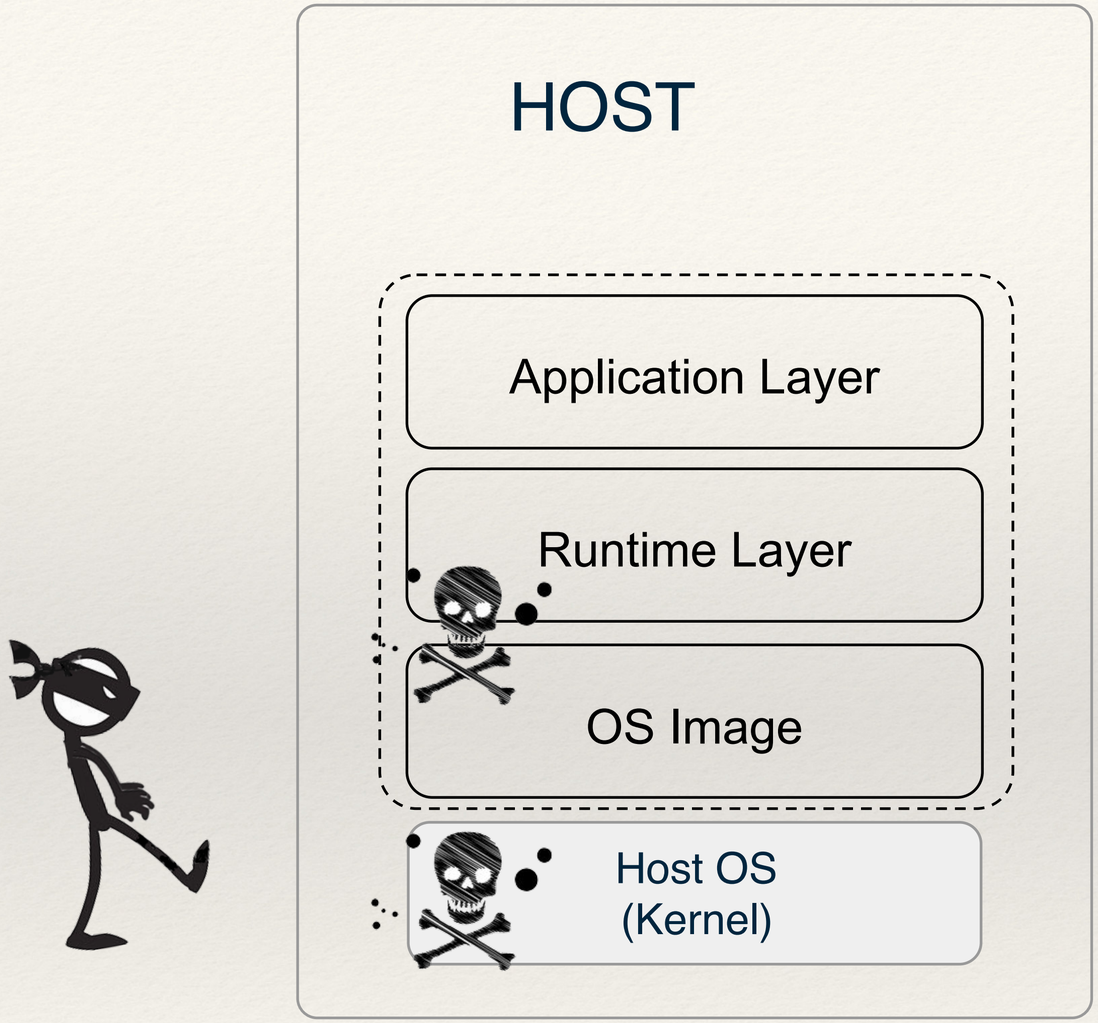

HOST
Application Layer
Runtime Layer
OS Image
Host OS
(Kernel)

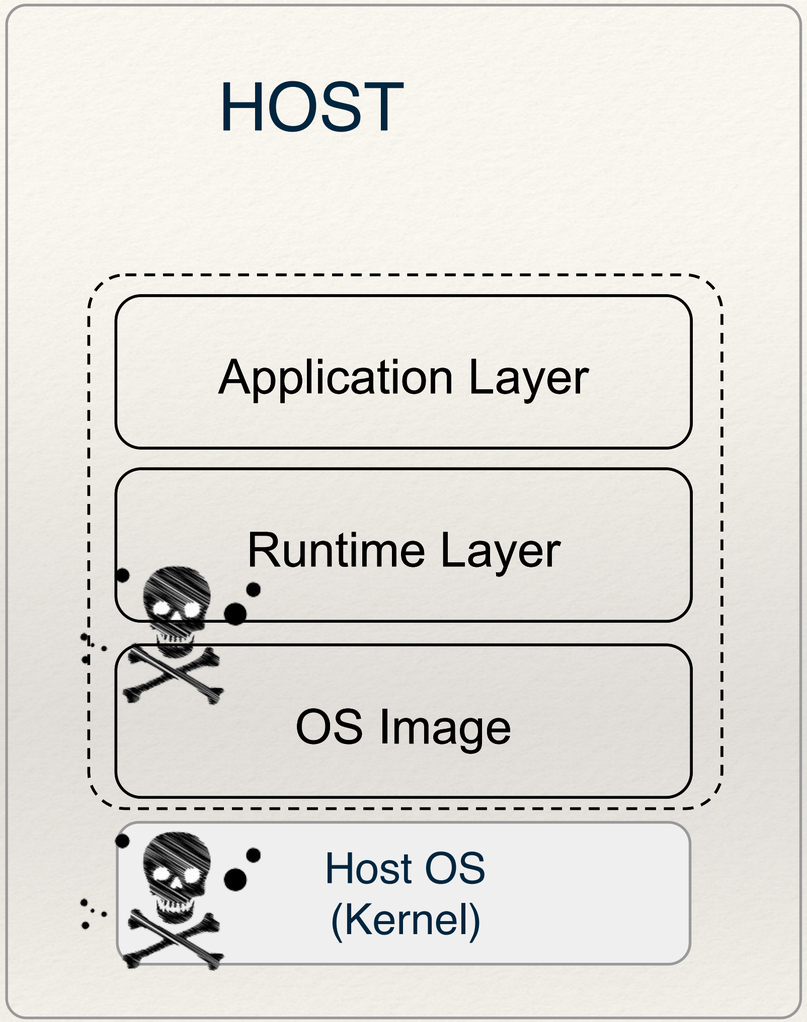

HOST
Application Layer
Runtime Layer
OS Image
Host OS
(Kernel)

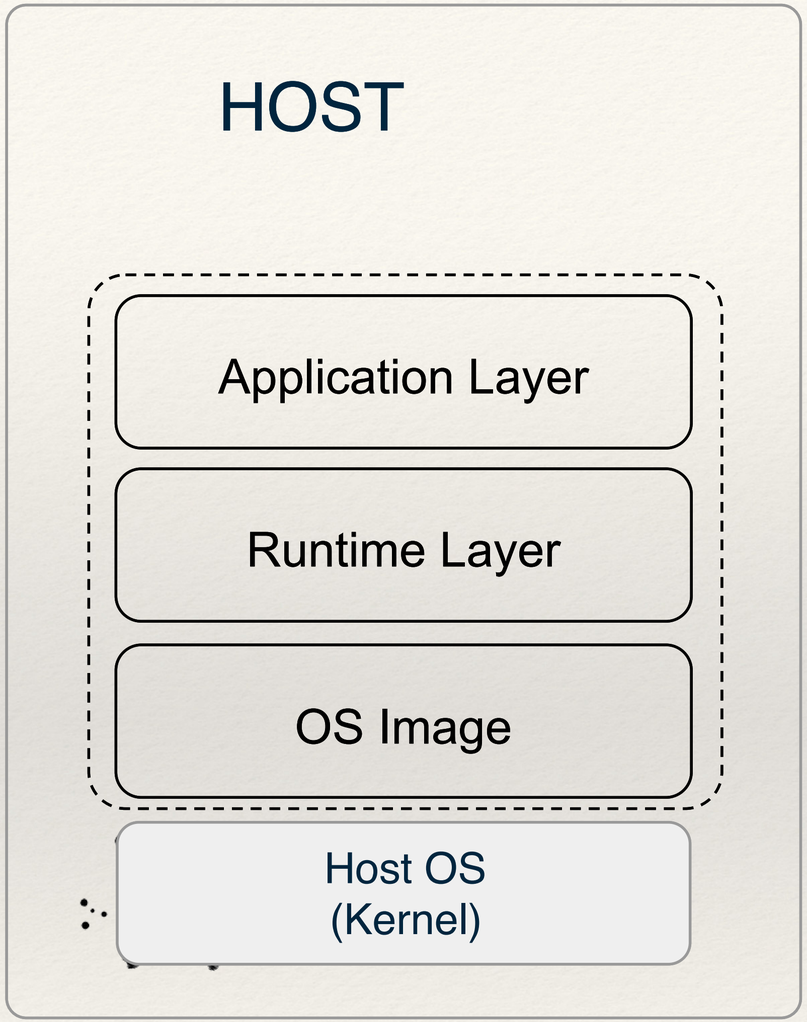

HOST
Application Layer
Runtime Layer
OS Image
Host OS
(Kernel)


HOST
Application Layer
Runtime Layer
OS Image
Host OS
(Kernel)


HOST
Application Layer
Runtime Layer
OS Image
Host OS
(Kernel)


HOST
Application Layer
Runtime Layer
OS Image
Host OS
(Kernel)


HOST
Application Layer
Runtime Layer
OS Image
Host OS
(Kernel)




You can
**Repave**
the entire
environment

# VERY Often!!!

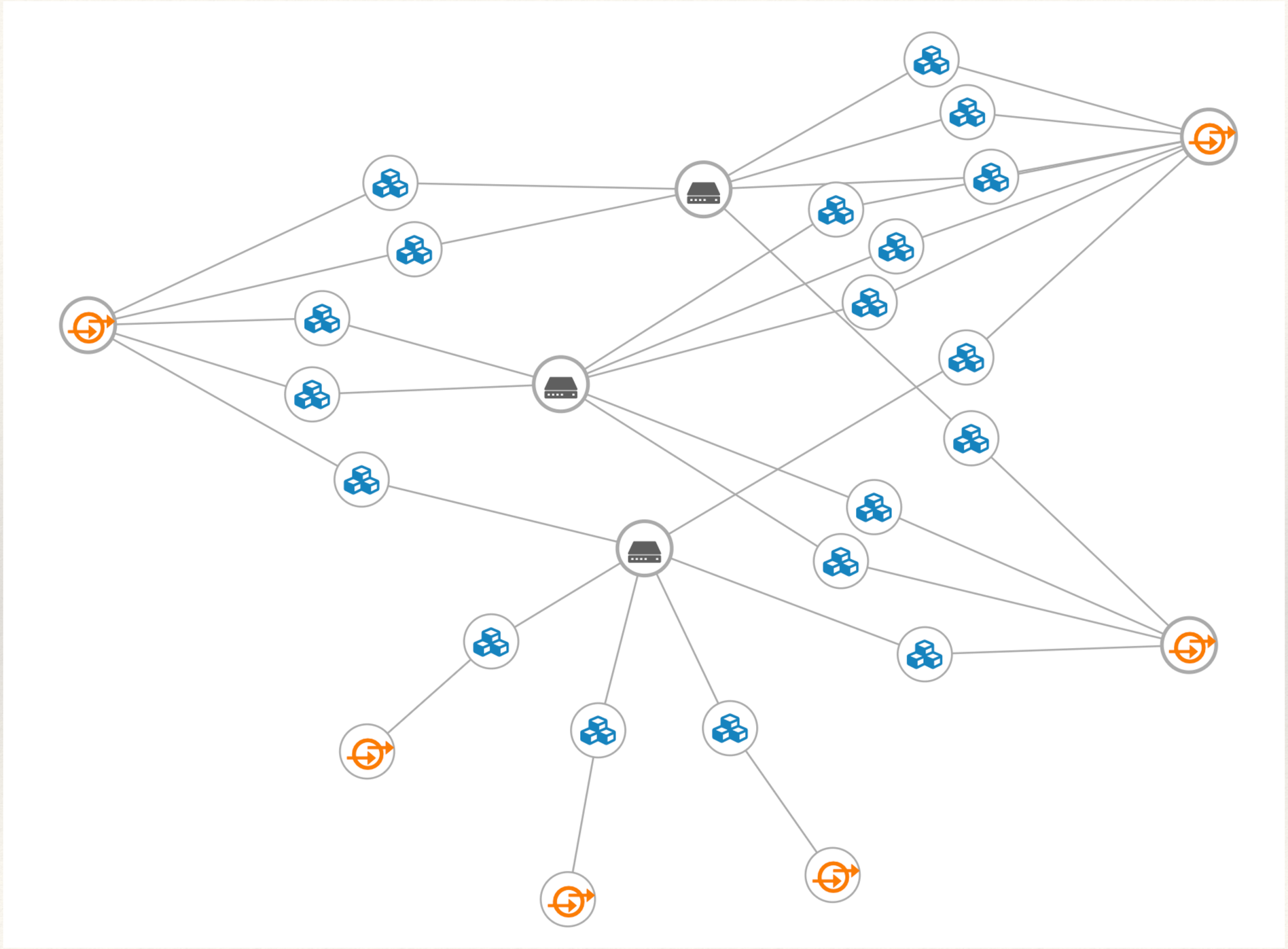
## Like several times per week “often”!

What is the mental model for humans to do this?

(How about: repave)

# How do we initiate a repave?

(How about we declare it?!)



# Business Value

Stronger Security Posture ⟶ Safe Customer Data

Stronger Security Posture ⟶ No Breaches

We cannot reason about every detail so we need

**a model**

that allows machines to reason for us.

# Functional Systems Programming

# Where to Begin

**For the app team…**

- Clojure
- Kotlin
- F#
- Scala

**For the platform team…**

- Kubernetes
  - Deploy a replica set, kill a node, watch the “magic”
  - Understand the Controller Pattern
- BOSH
- Kafka - Event Log

# What We Still Need Help With

- More people getting the “toe turn”
- Kubernetes as a programming model
- How to manage temporal dependencies in this distributed world.

# Thank you

@cdavisafc