Taking Ops & Infrastructure From Iterative To Functional

(Just Like Dev)

Cornelia Davis • @cdavisafc
Sr. Director of Technology
Pivotal

Me?

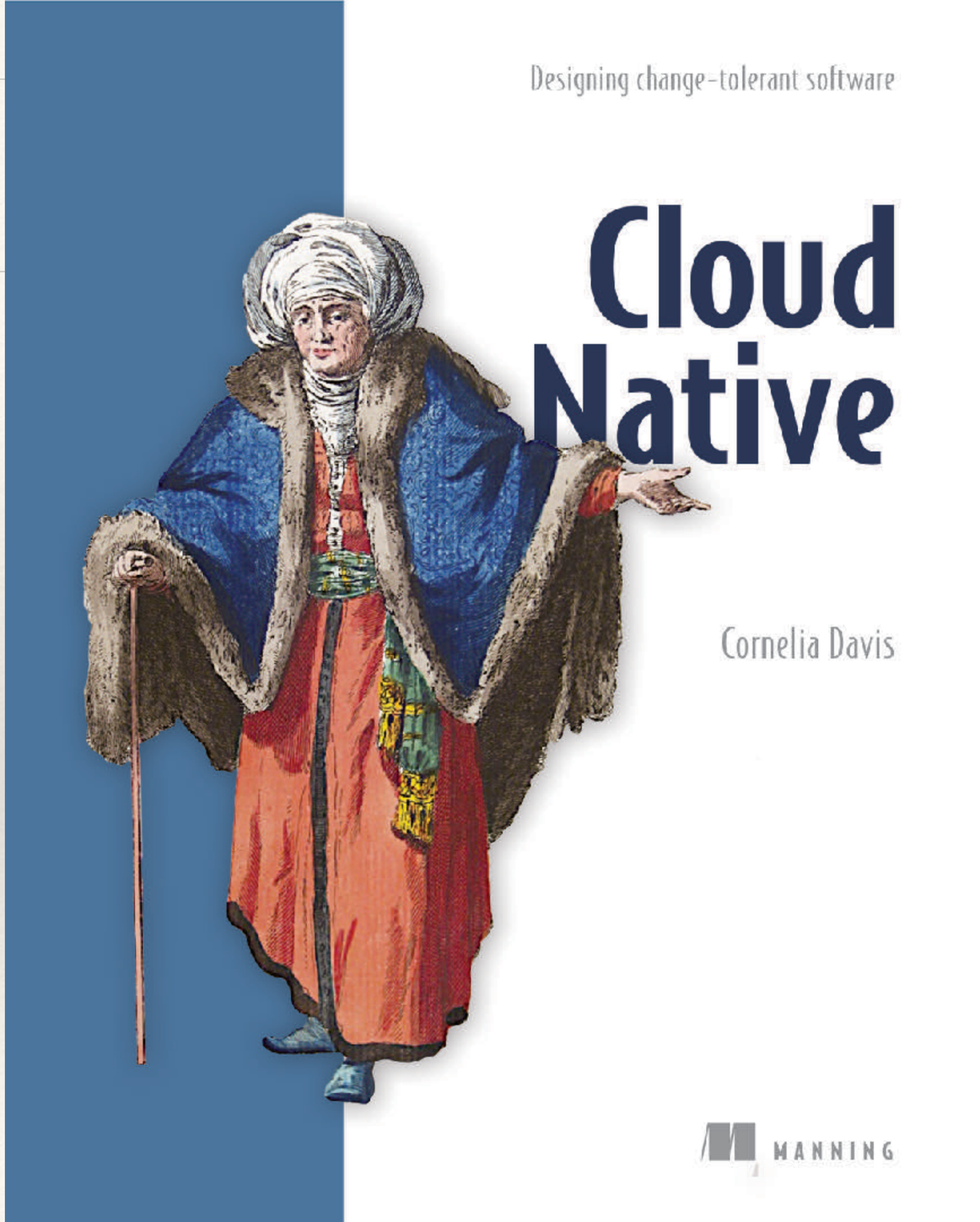
- Cal State Northridge: Computer Science
- Indiana University: Computer Science (ABD)
- Pivotal: Product Strategy (PCF, PCC, PKS)

Ops
as much as
Dev
now

- And …

Discount code 40% off!: 40cloudnat

https://www.manning.com/books/cloud-native



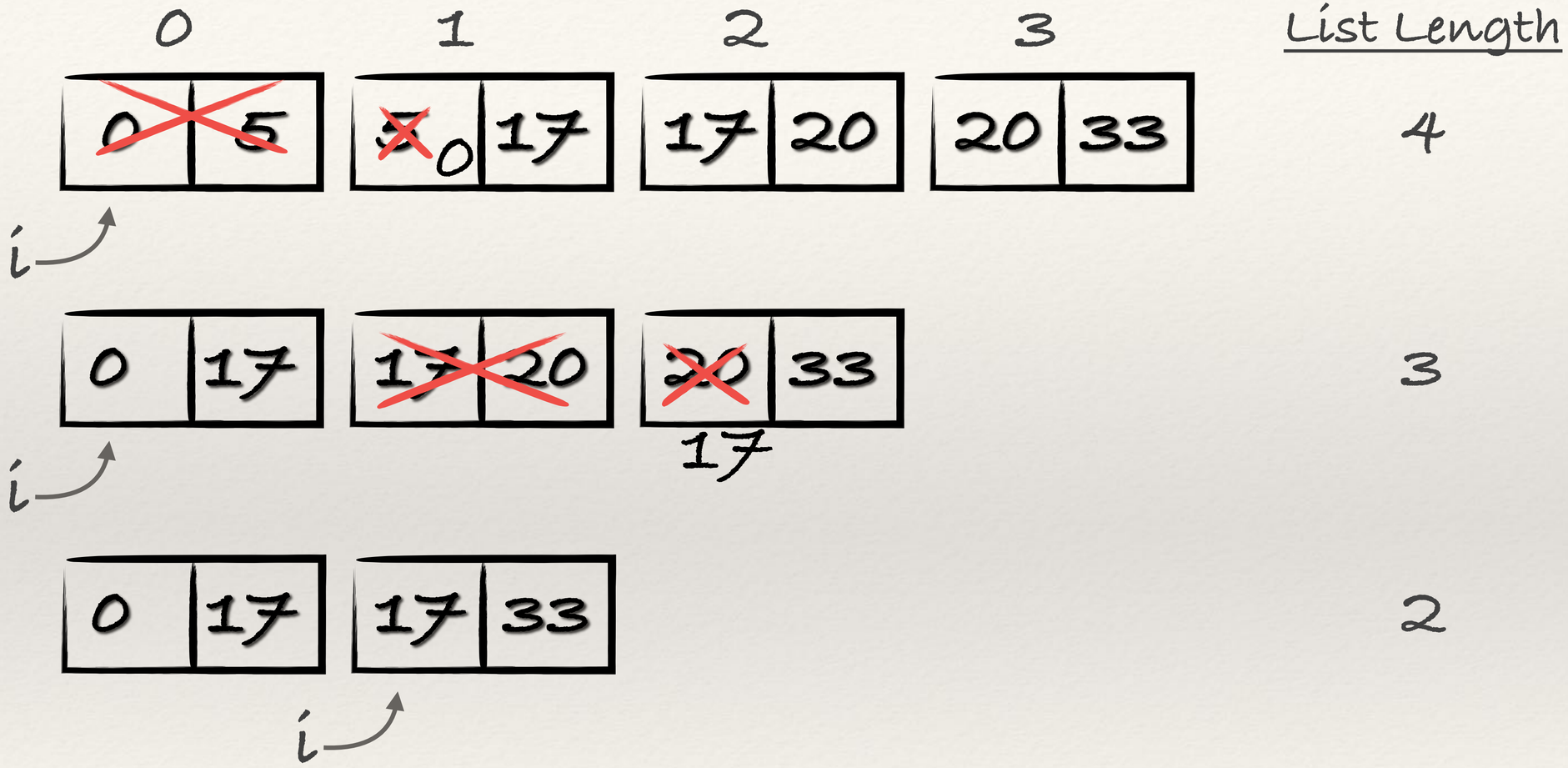
Let's cut some code…

```
[(0, 5), (5, 17), (17, 20), (20, 33)]
```

Merge short Segments together

```
[(0, 17), (17, 33)]
```

```
def mergeSegments(s_list, threshold):
  list_len = len(s_list)
  i = 0
  while i < list_len:
    seg_length = s_list[i][1]-s_list[i][0]
    if seg_length < threshold:
      s_list[i+1] =
            (s_list[i][0], s_list[i+1][1])
      del s_list[i]
      list_len -= 1
    else:
      i += 1
```

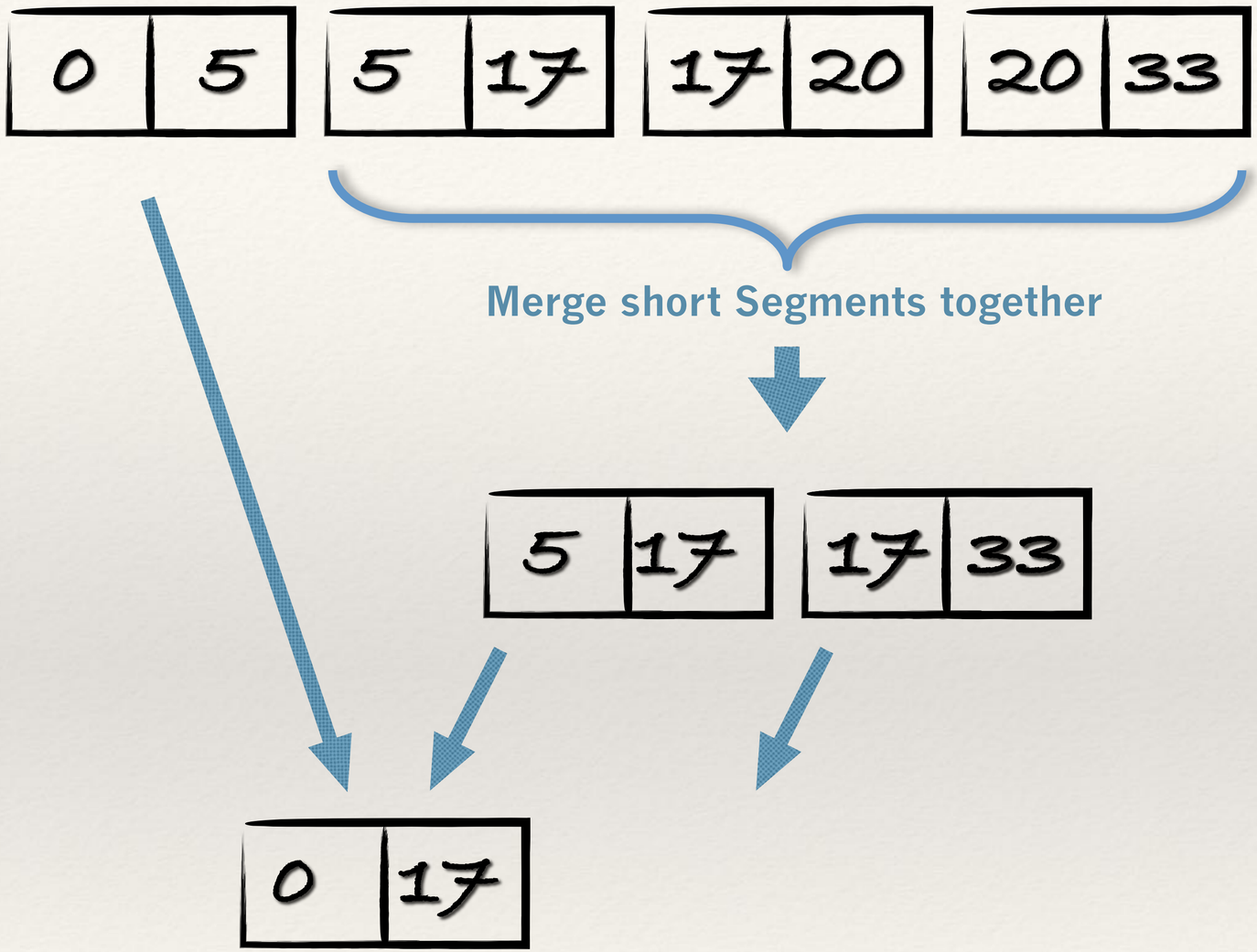

0
1
2
3
List Length
0 5
5 0 17
17 20
20 33
4
i
0 17
17 20
20 33
17
3
i
0 17
17 33
2
i

Hallmarks of Iterative Programs

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

- Difficult to parallelize
- Hairy edge cases

```
(define (mergeSegments segs threshold)
  (if
    (or (null? segs) (null? (cdr segs)))
    segs
    (let
      ((rest (mergeSegments (cdr segs) threshold)))
      (if
        (< (currSegLen segs) threshold)
        (cons (cons (start segs) (end rest))
              (cdr rest))
        (cons (car segs) rest)))))
```


0
5
5
17
17
20
20
33
Merge short Segments together
5
17
17
33
0
17

Hallmarks of Functional Programs

- Declarative
- NO side effects
- Tail recursion (continuations!!!)

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

3000 Lines of Objective C

To

1500 Lines of Typescript & React

To

500 Lines of Clojure

“One of the hardest things I’ve ever learned, but one of the most rewarding. It’s the first time I’ve been able to write **code that doesn’t collapse on itself**.

It makes programming fun again!”

Gene Kim

The

Programming Primitives

have a

Marked impact

on the

Maintainability and Stability

of the

Application

Platform The

~~Programming~~ Primitives

have a

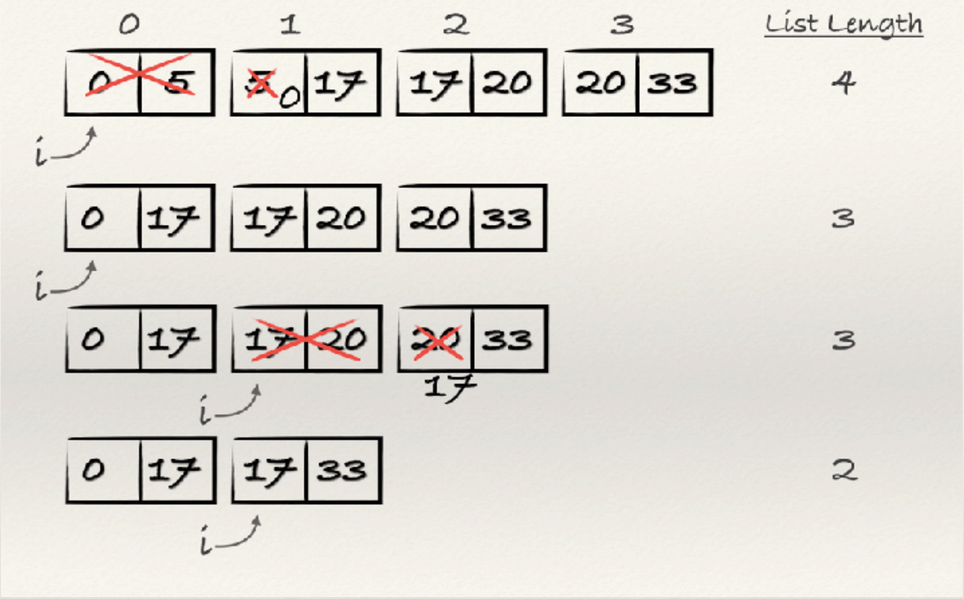
Marked impact

on the

Maintainability and Stability

of the

Application



What is our

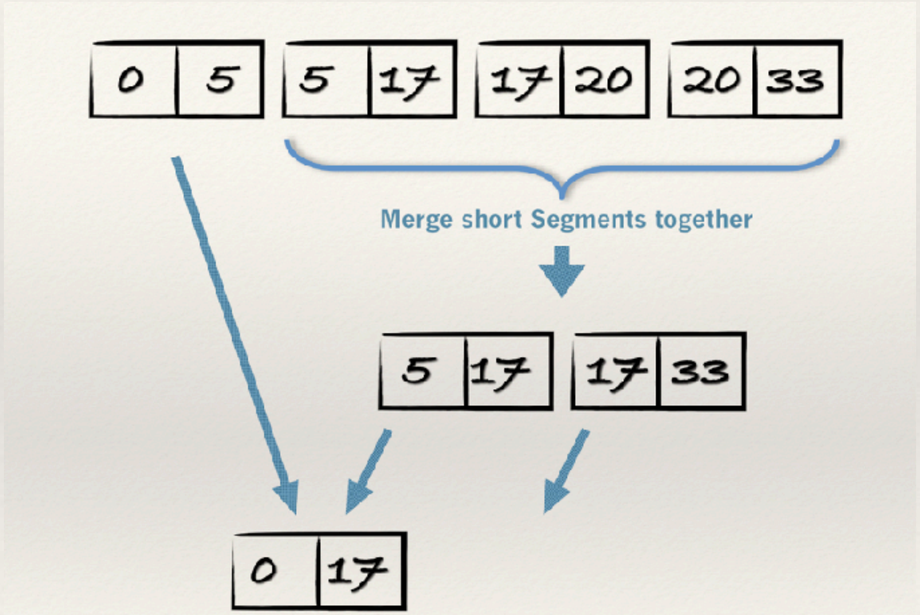
Iterative

For

Ops?

What could be our

Functional?



Iterative

```
function SumToN(n : INTEGER): INTEGER;
var
  i : INTEGER;
  s : INTEGER;
begin
  i := 0;
  s := 0;
  while (i <= n) do
    begin
      s := s + i;
      i := i + 1;
    end;
  SumToN := s;
end;
```

Functional

```
(define sumToN
  (lambda (n)
    (if (= n 1)
        1
        (+ n (sumToN (- n 1))))))
```

Base Case

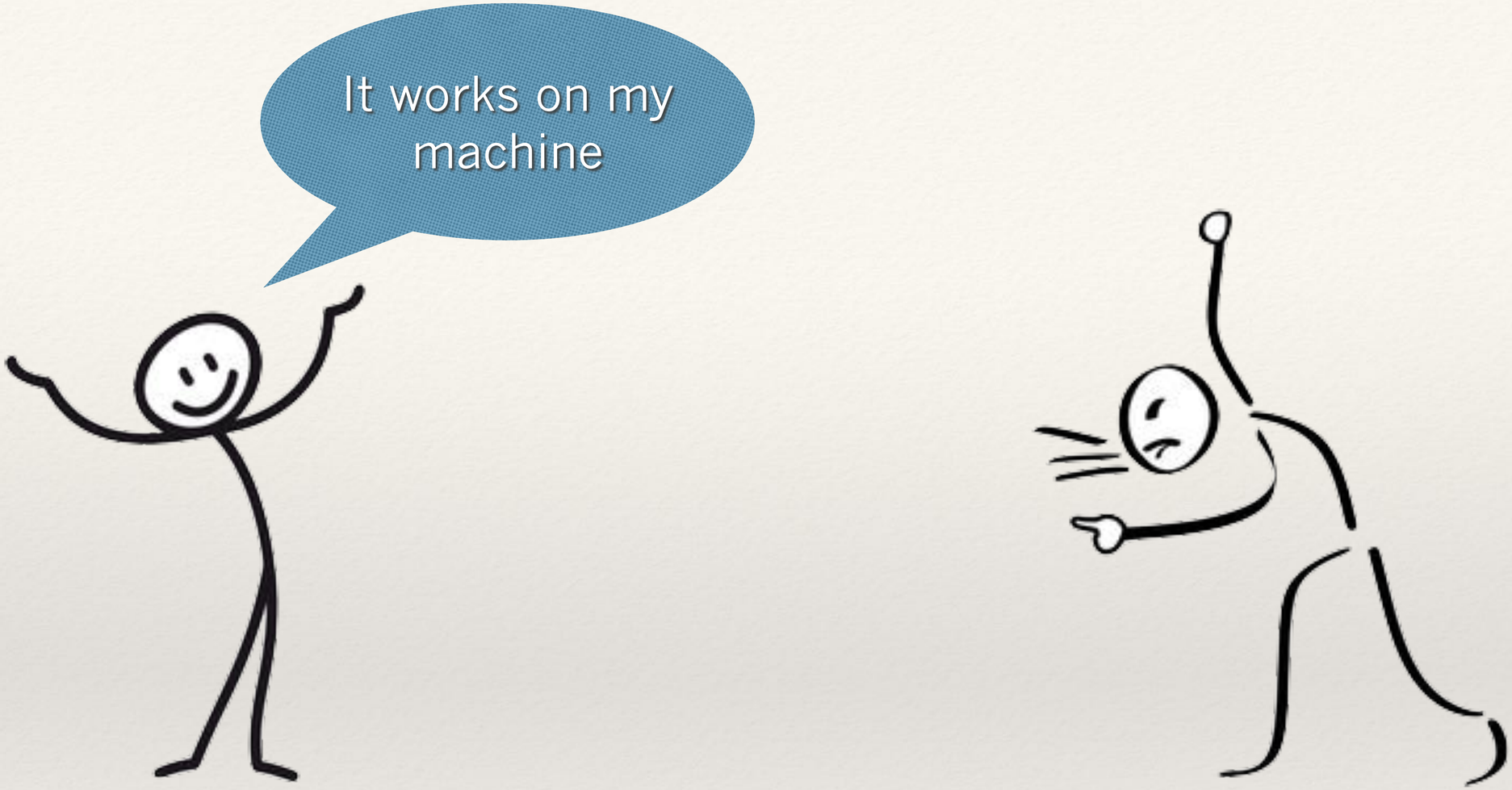
General Case
(Bit of a leap of faith)

Let something else do some of the work!

Ongoing Ops Challenges

- Snowflakes!!
- Security …
- … and Compliance
- Resilience

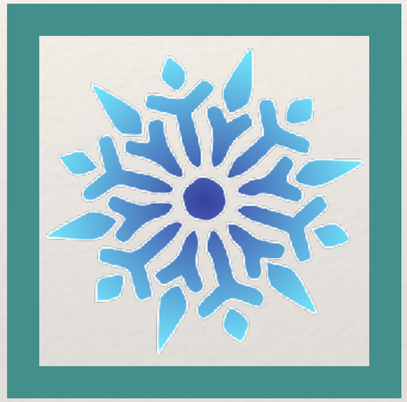
Snowflakes


It works on my machine

Dev

Test

UAT

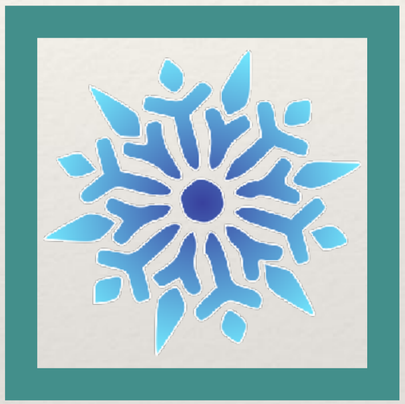


Prod

Dev

Test

UAT

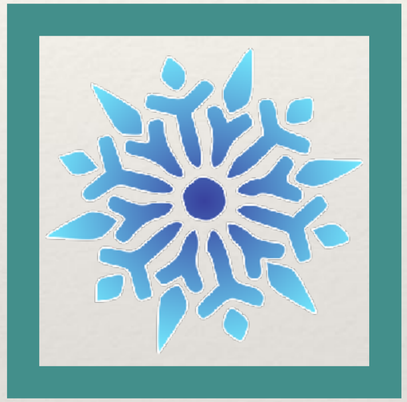


Prod

Dev

Test

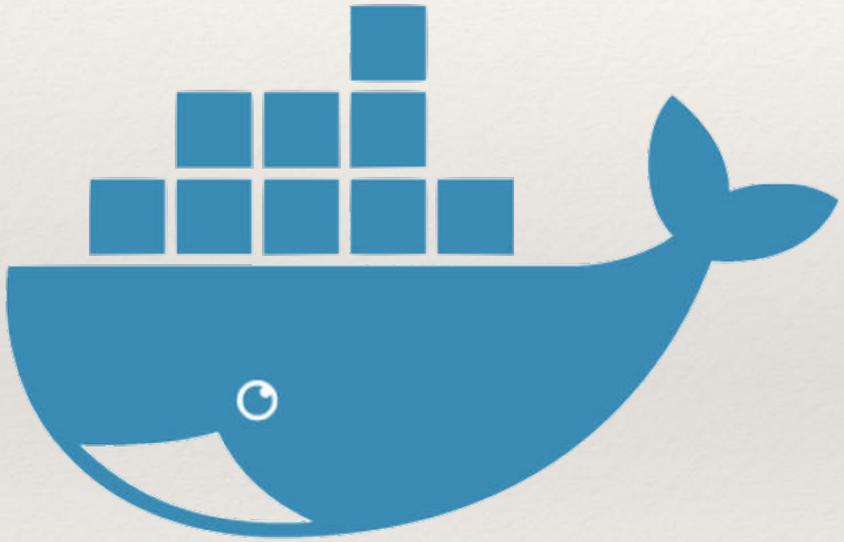
UAT



Prod

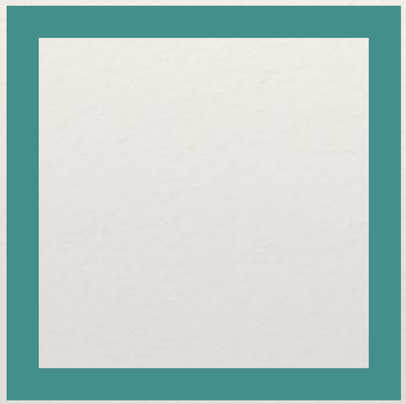
Single Deployable Artifact

Common theme: Immutability

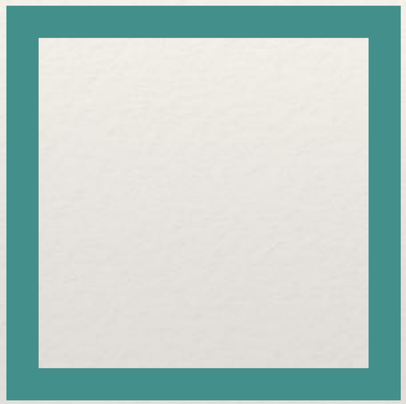


- Bundles all dependencies
- Properly parameterized

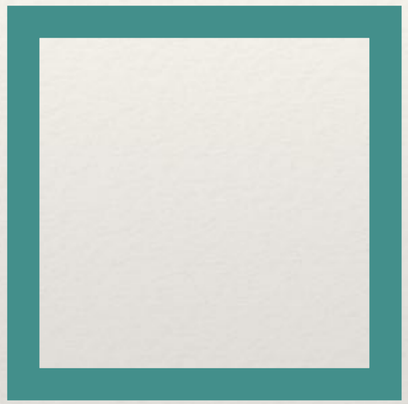
Security …



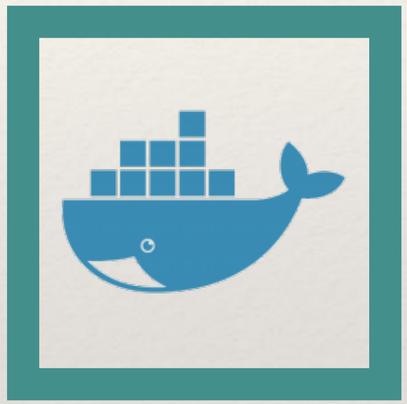
Dev



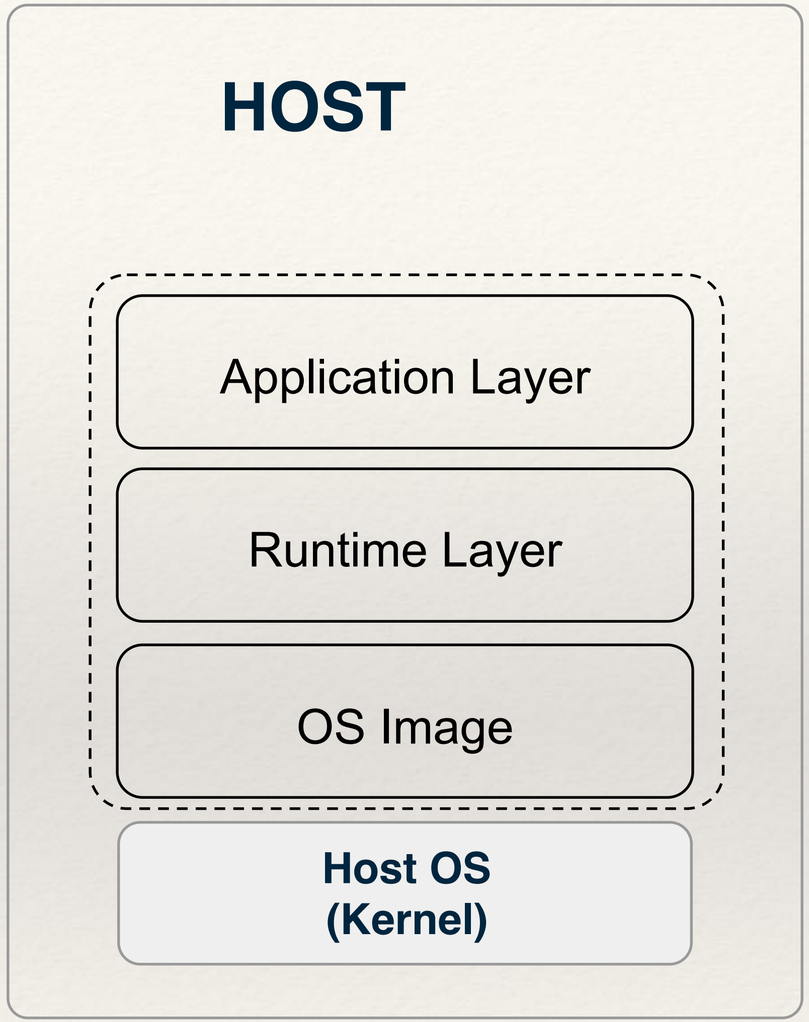
Test



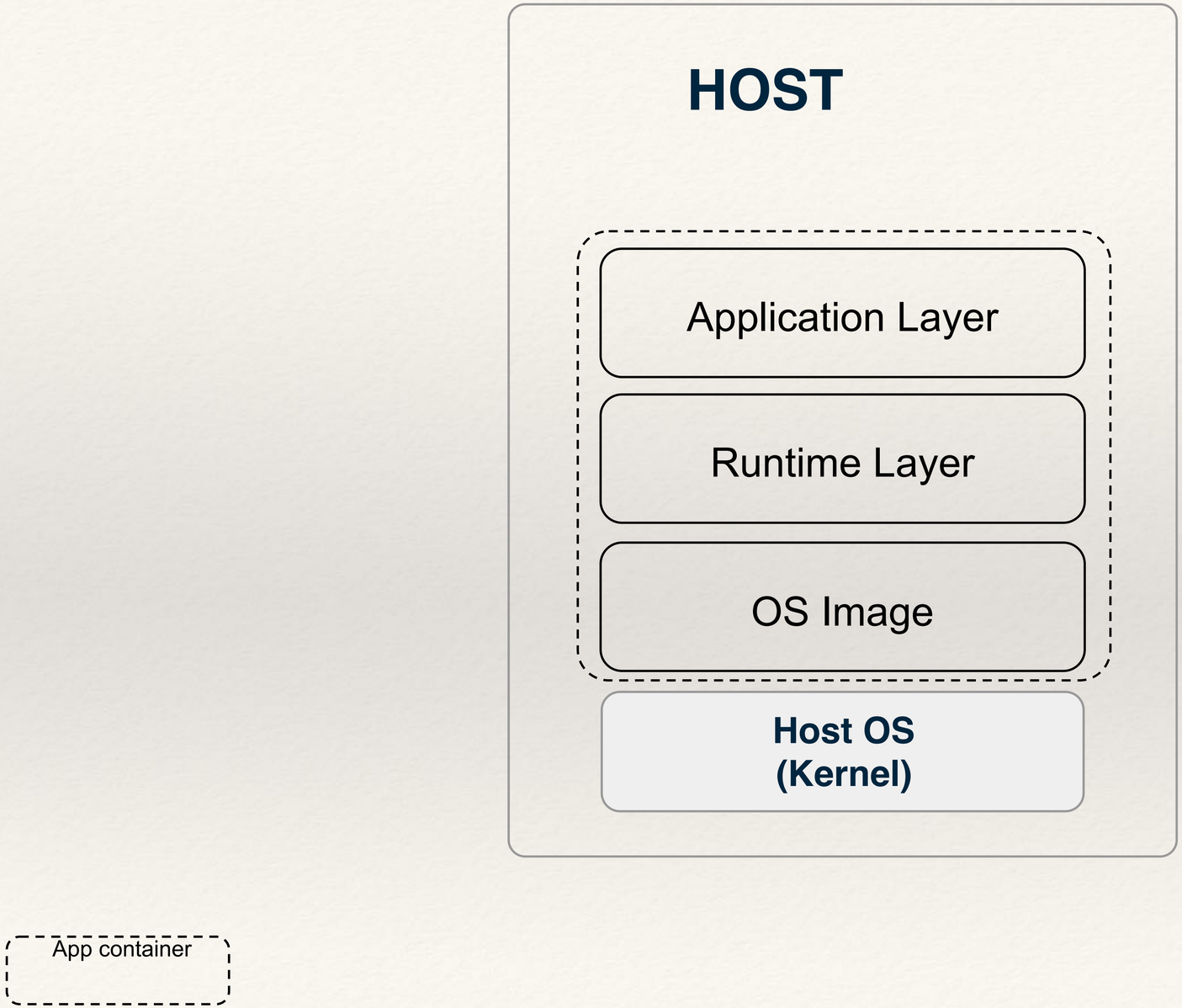
UAT

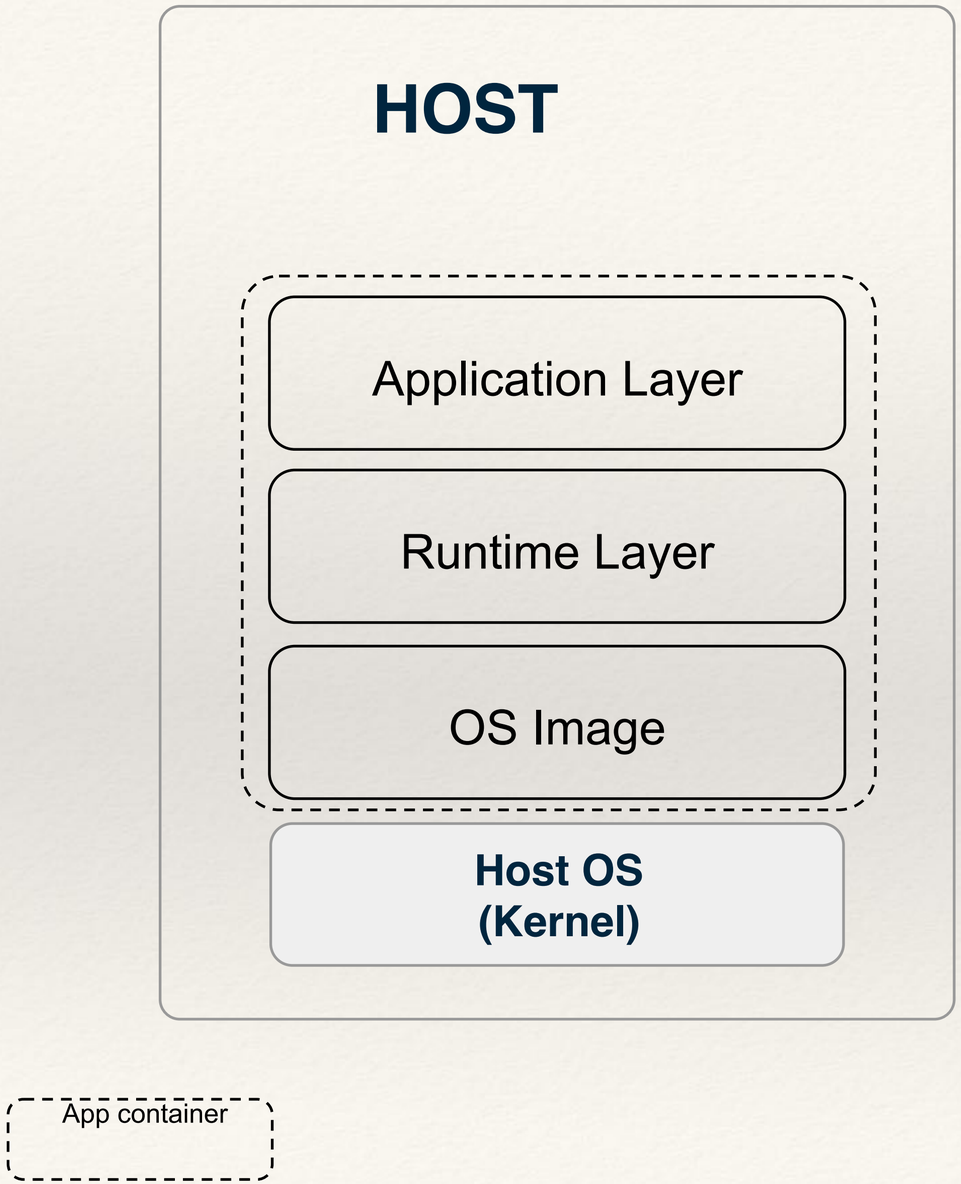


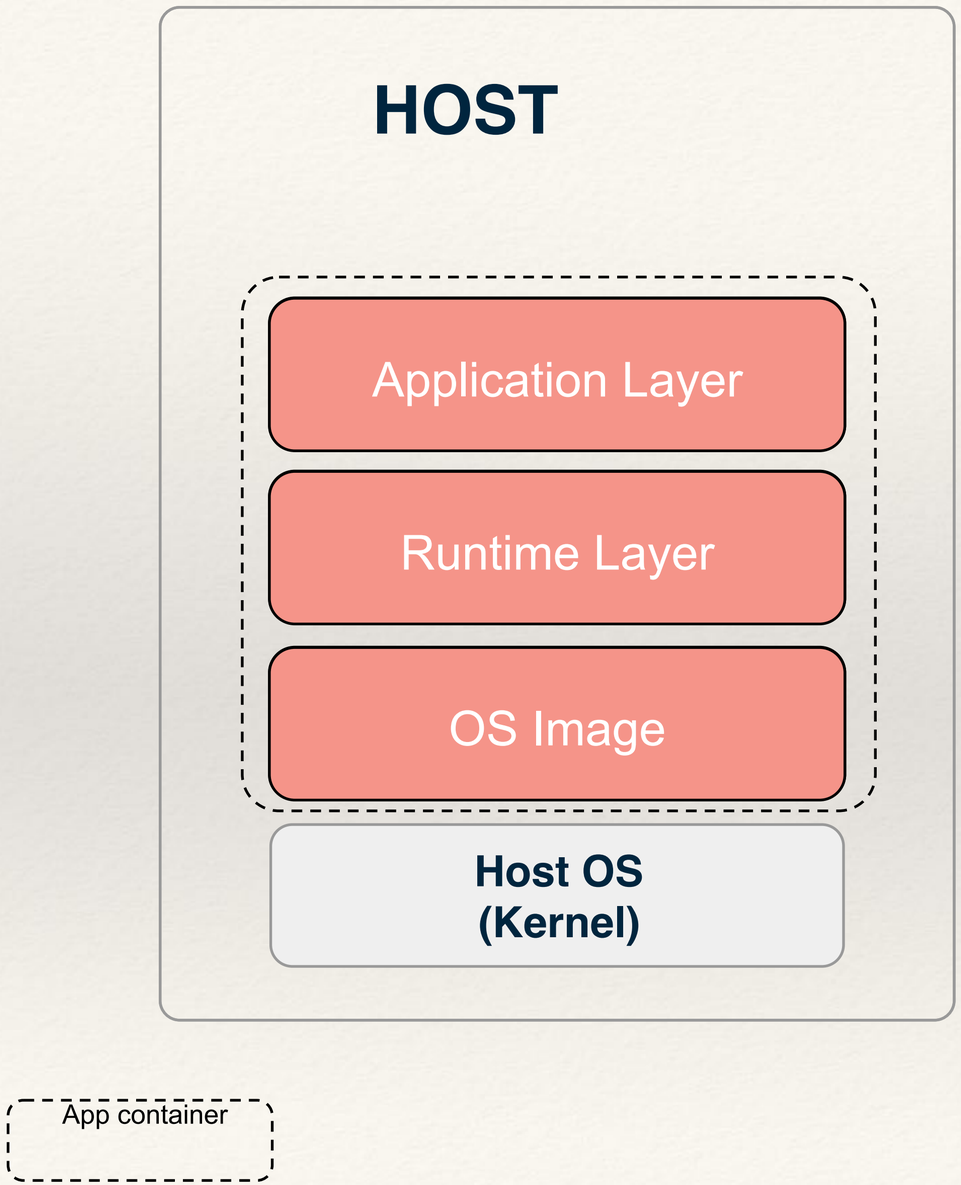
Prod

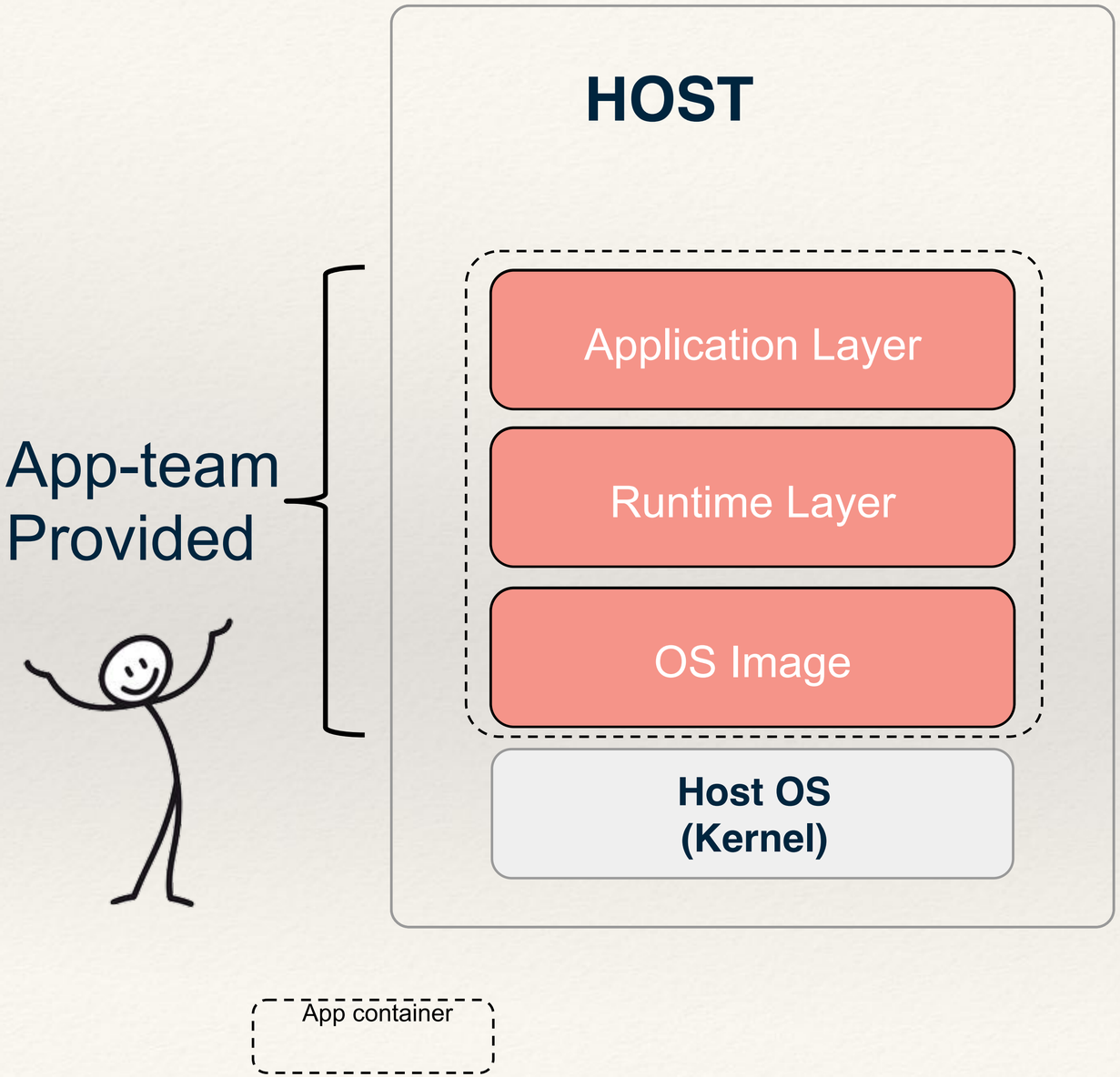


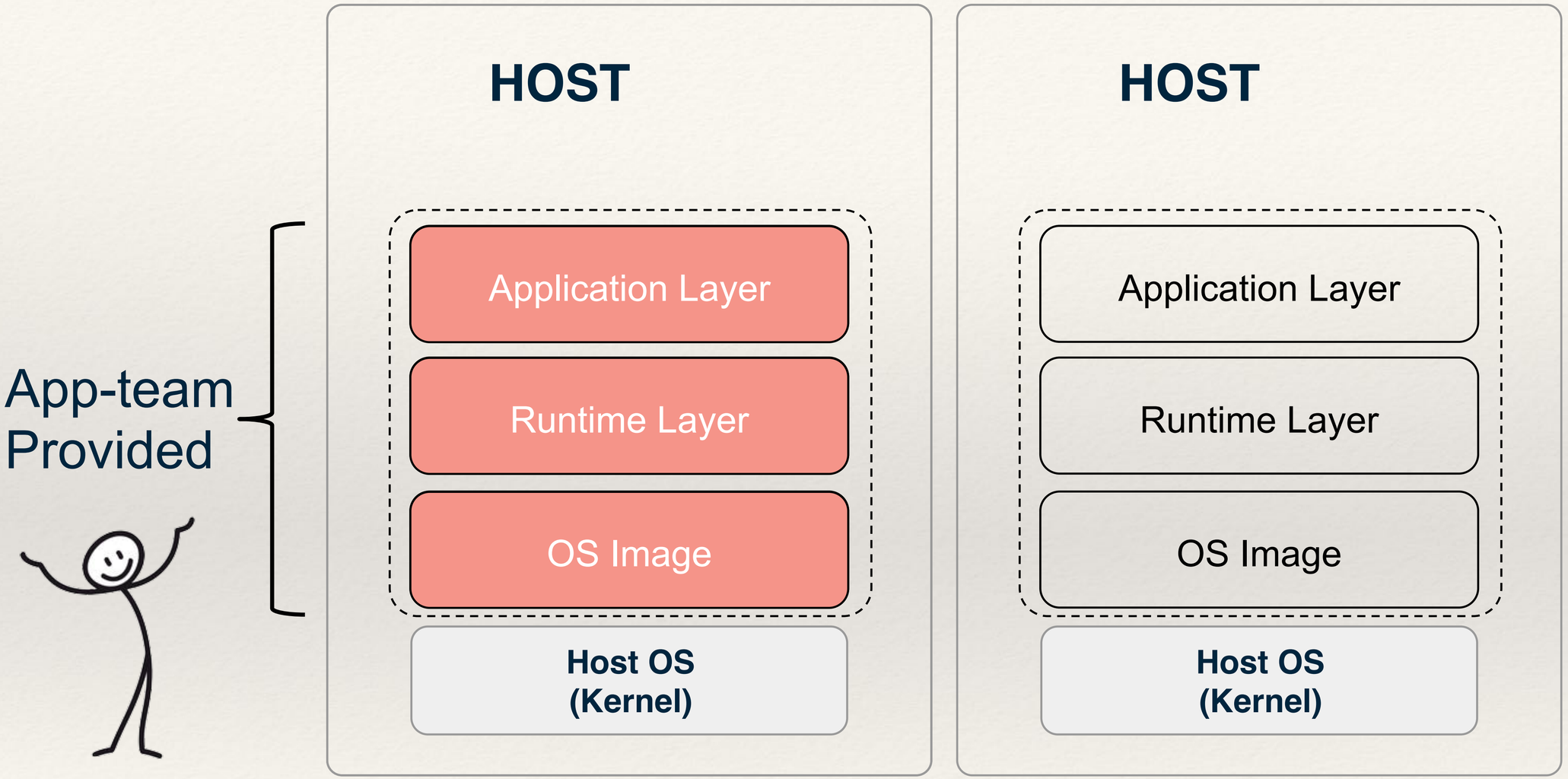
App container

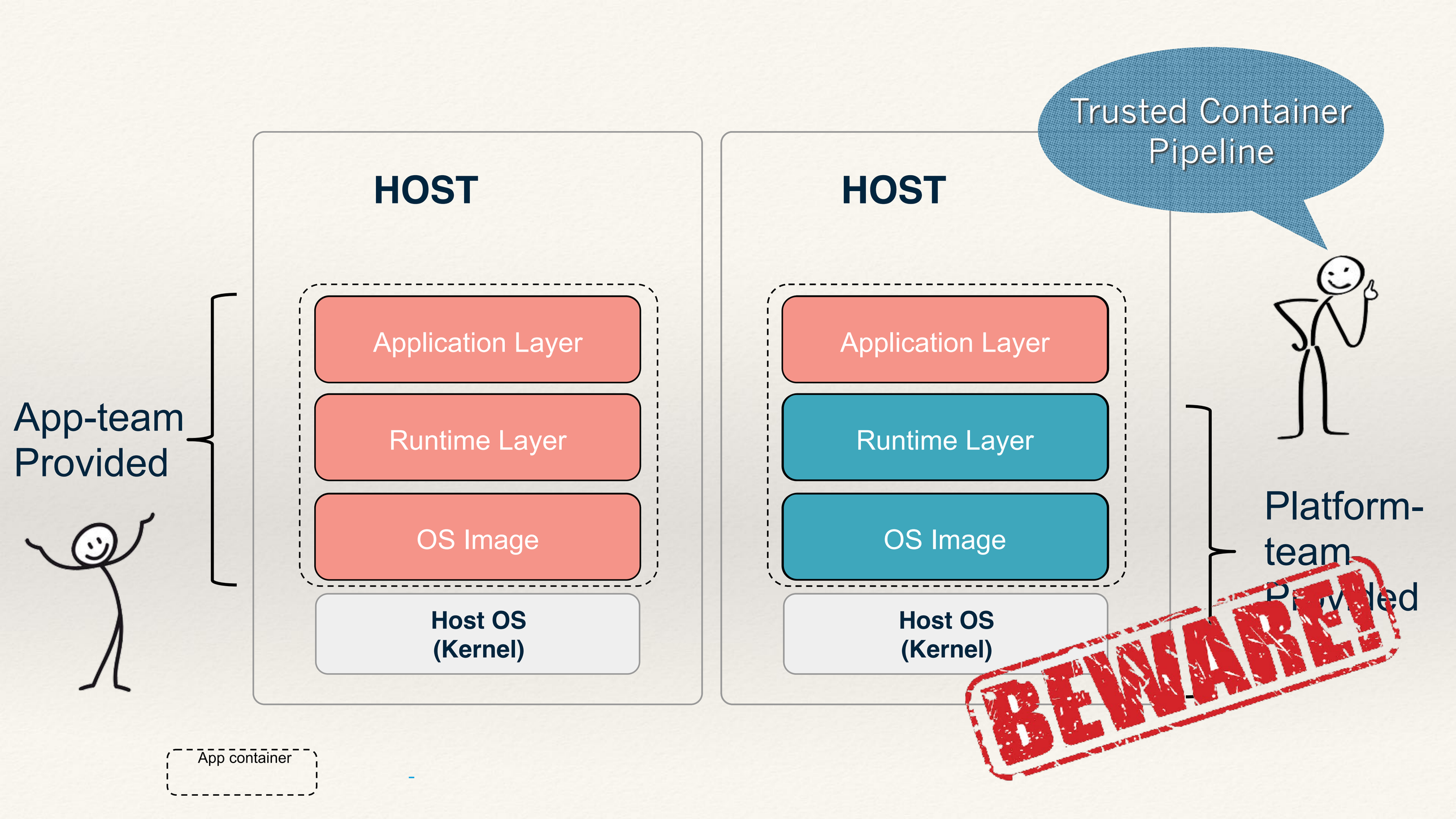

HOST
Application Layer
Runtime Layer
OS Image
Host OS
(Kernel)
App container

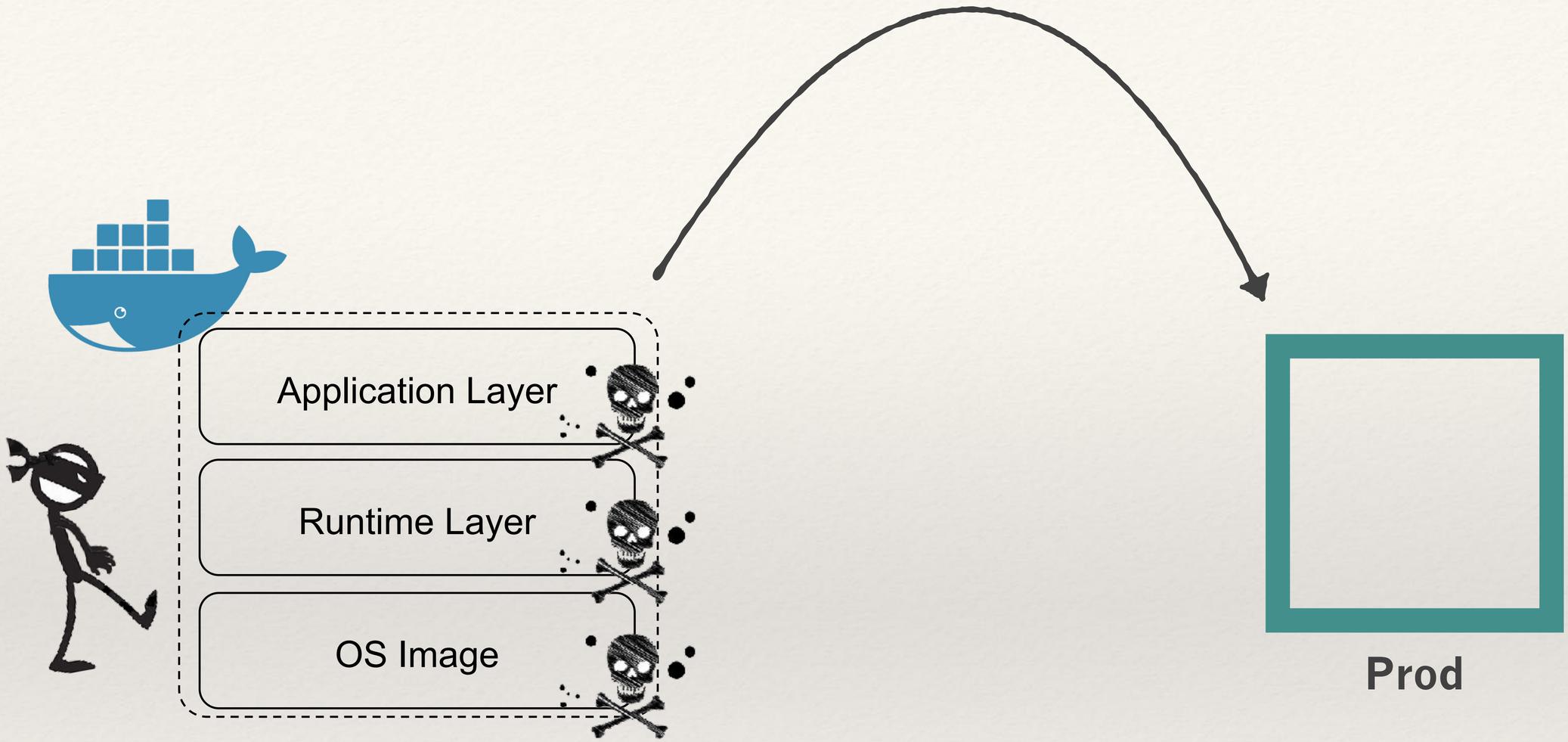

HOST
Application Layer
Runtime Layer
OS Image
Host OS
(Kernel)
App container

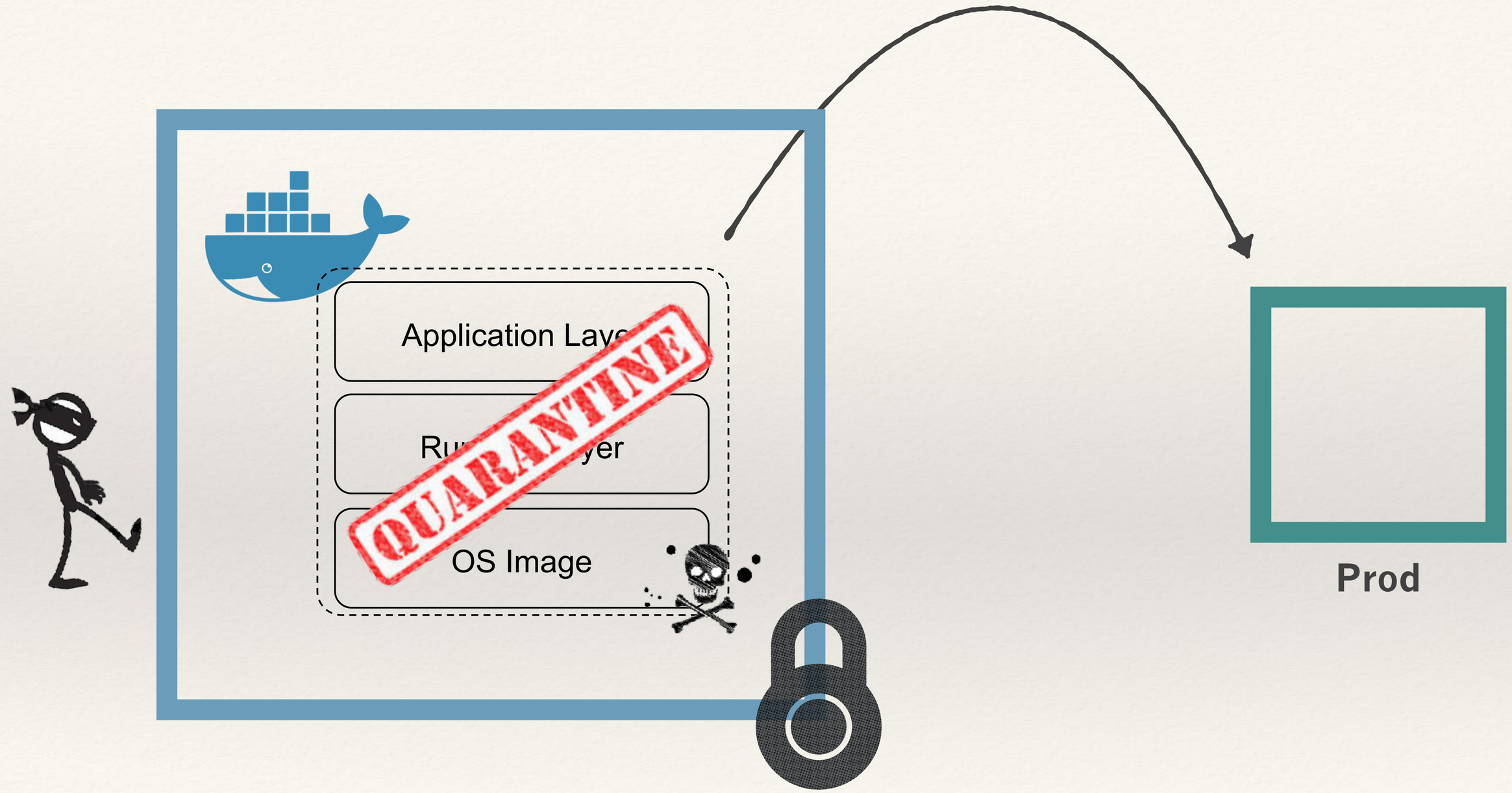

HOST
Application Layer
Runtime Layer
OS Image
Host OS
(Kernel)
App container


HOST
Application Layer
Runtime Layer
OS Image
Host OS
(Kernel)
App-team
Provided
App container

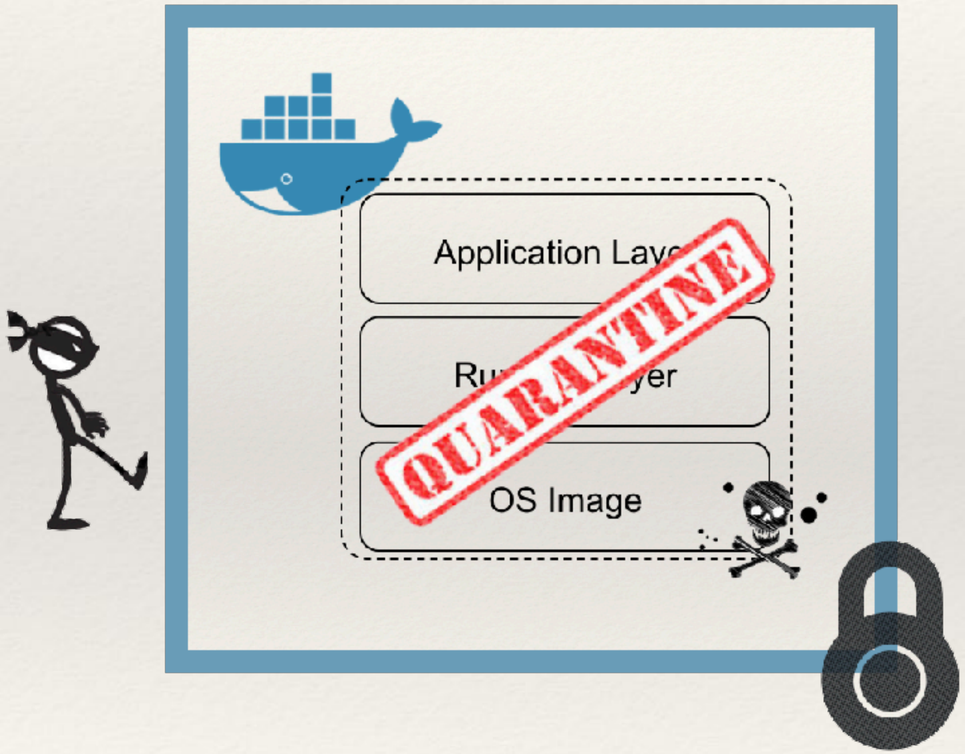

HOST
HOST
Application Layer
Runtime Layer
OS Image
Application Layer
Runtime Layer
OS Image
Host OS
(Kernel)
Host OS
(Kernel)
App-team
Provided
App container


Trusted Container Pipeline
HOST
HOST
Application Layer
Runtime Layer
OS Image
Application Layer
Runtime Layer
OS Image
Host OS (Kernel)
Host OS (Kernel)
App-team Provided
Platform-team Provided
BEWARE!
App container


Application Layer
Runtime Layer
OS Image
Prod


Application Lay
yer
QUARANTINE
OS Image
Prod

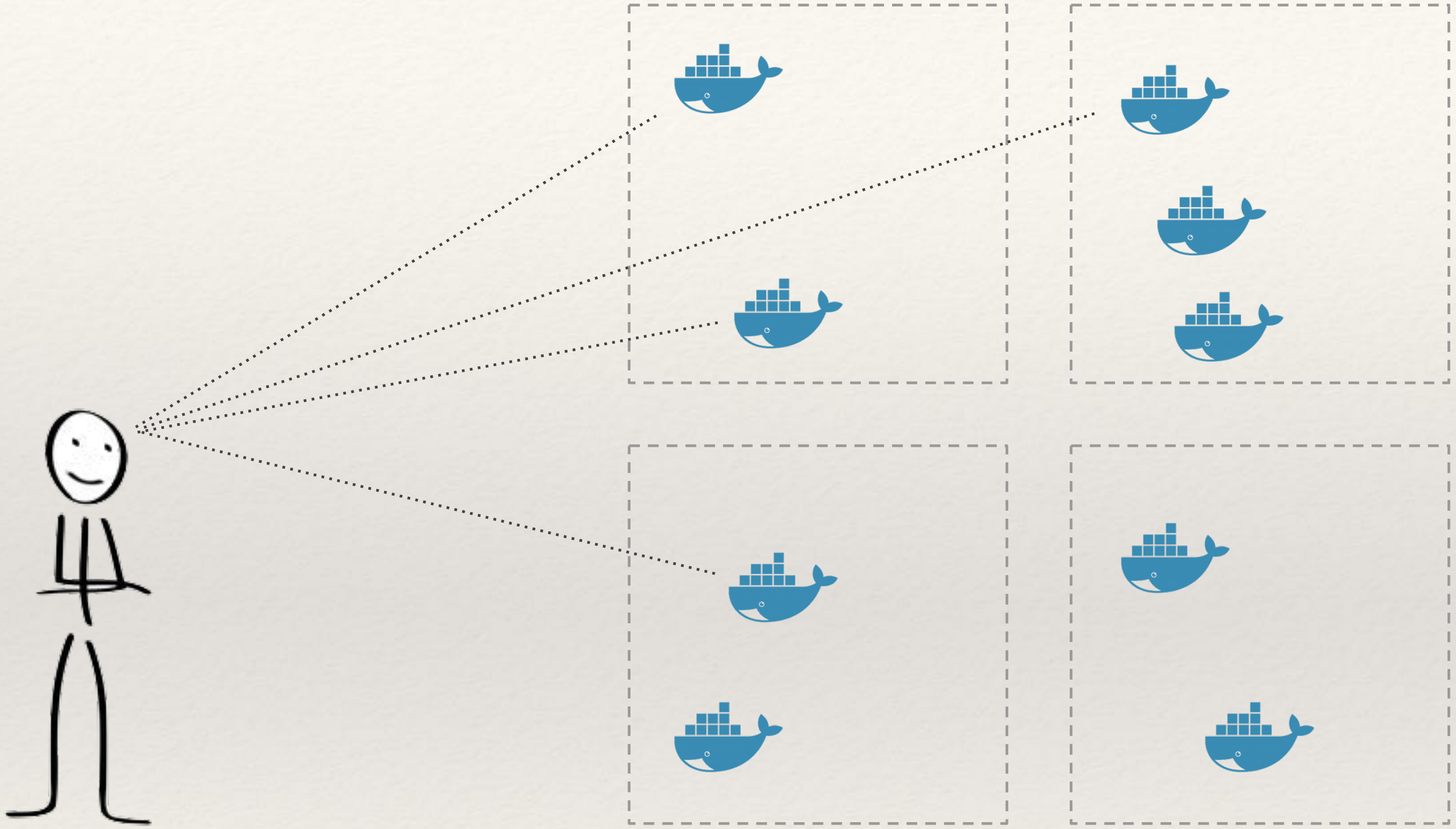
Artifact Management

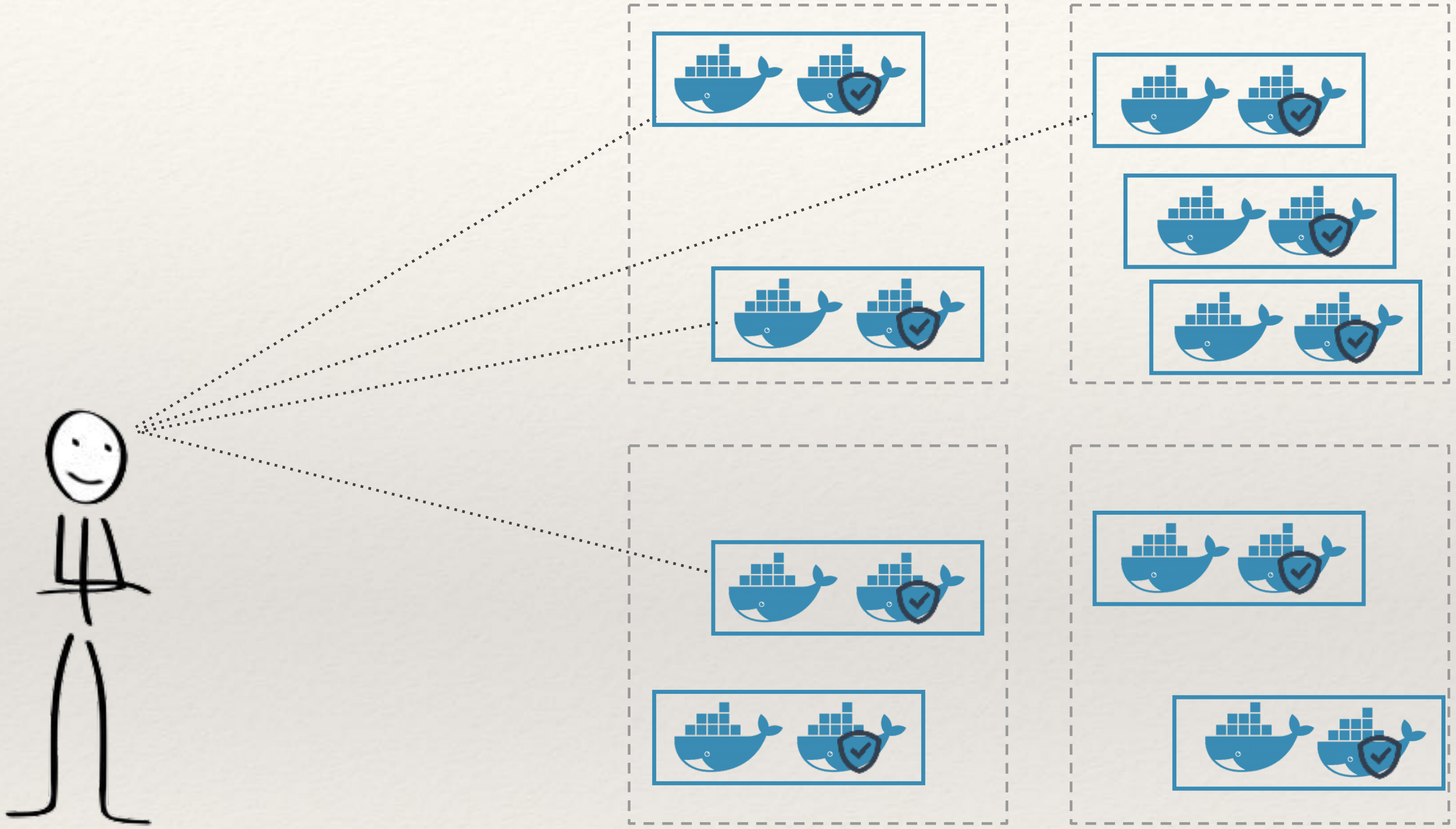


Common theme: Pipelines & SDLC

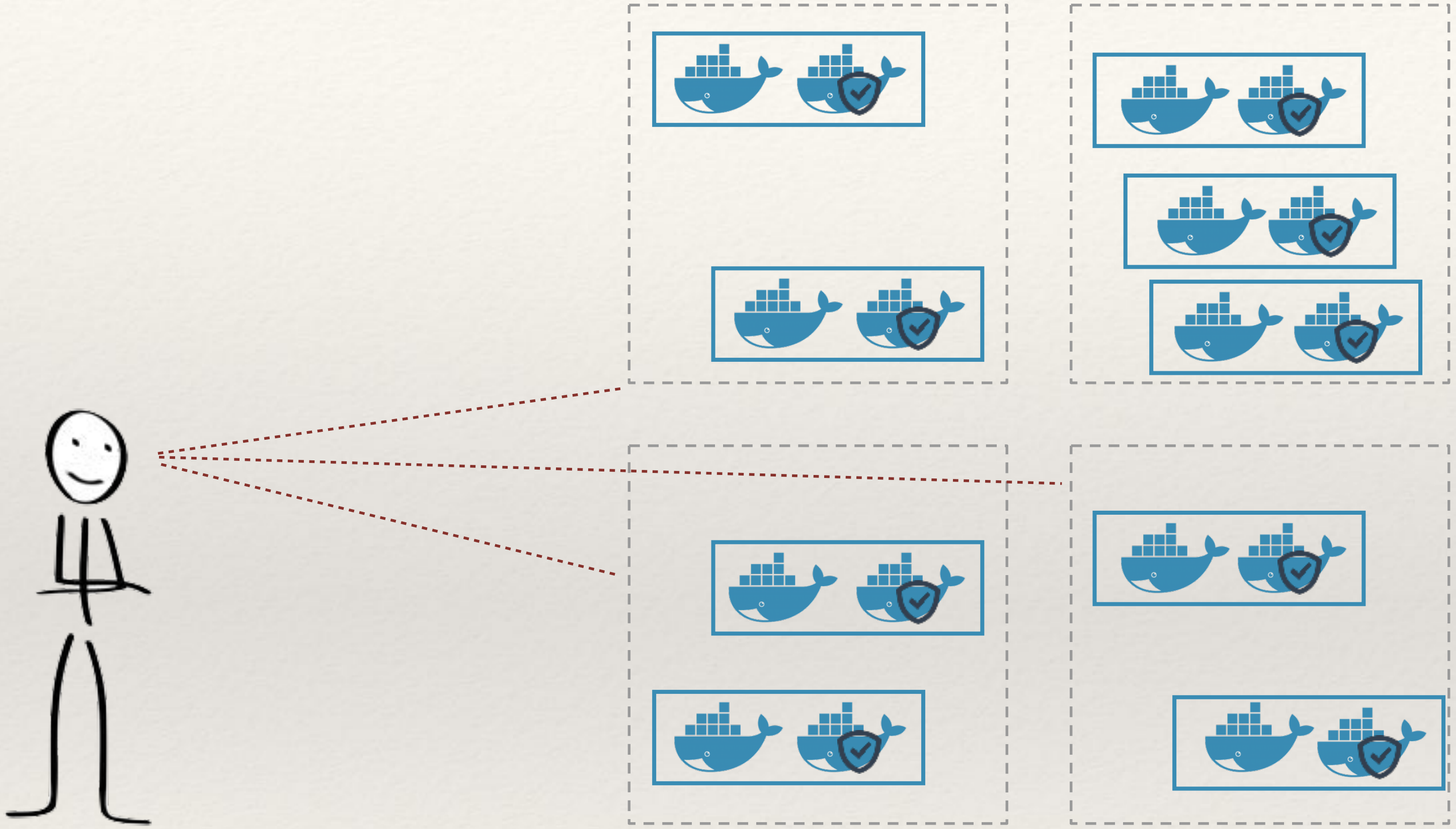
- Standard base images
- Approved runtimes
- Approved image build files
- Automated builds
- Image signing, scanning and quarantine
- ACLs

… and Compliance



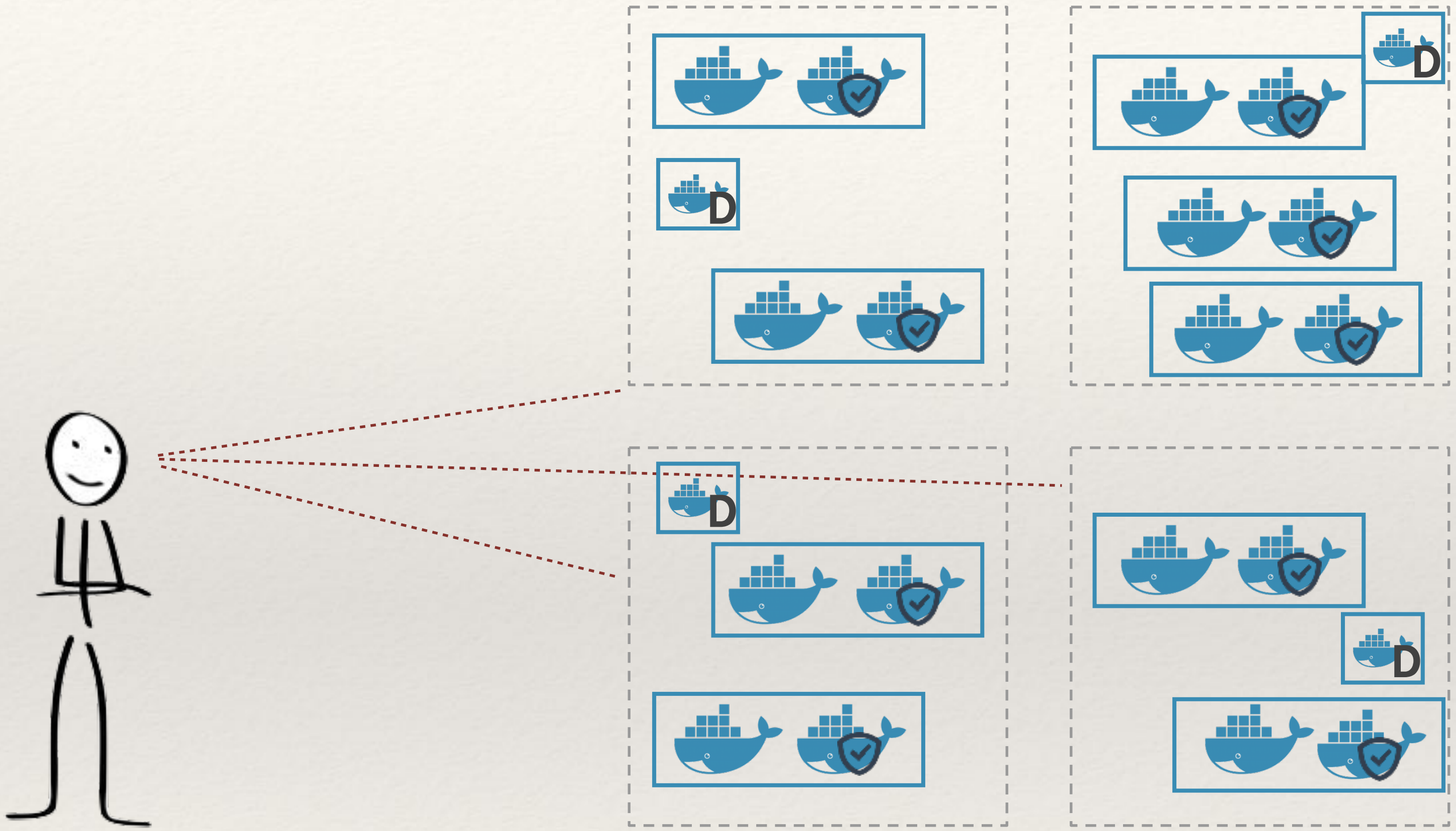


Cross-cutting Concerns…



Cross-cutting Concerns…

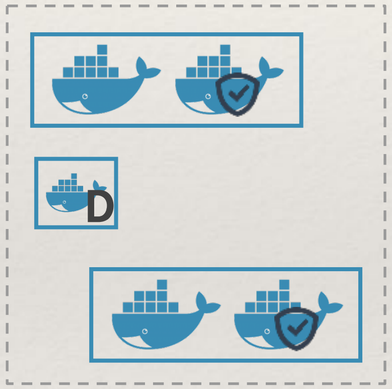
… both app and infrastructure



Cross-cutting Concerns…

… both app and infrastructure

Cross-cutting Concerns

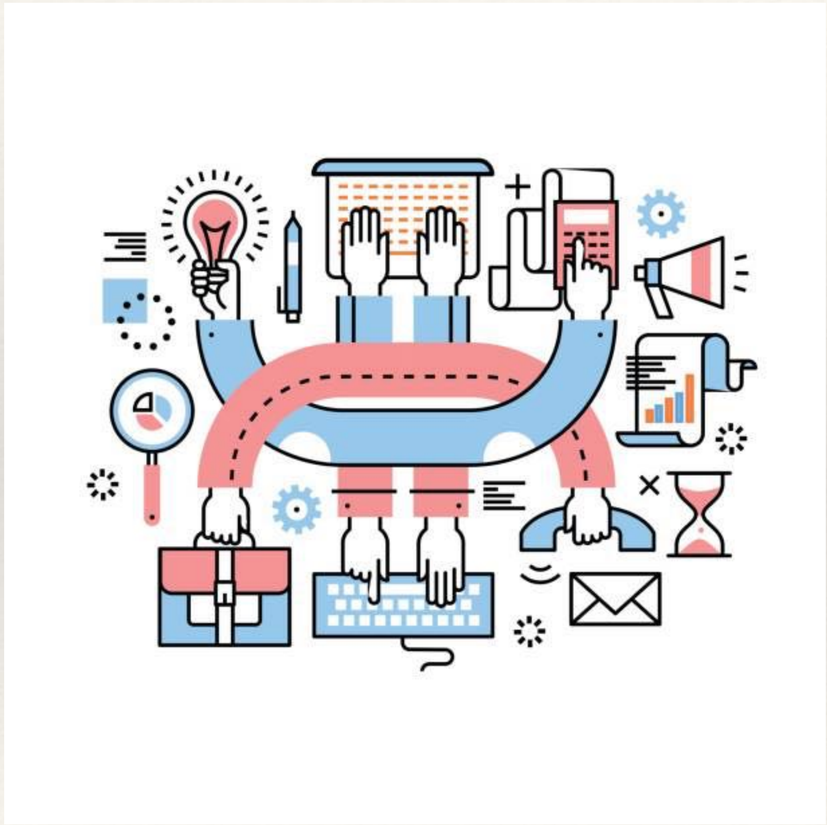


Common theme: Injection & Composition

- More than a container
- Aspect-oriented Programming
- App and Infra

Resilience

Imperative Deployments

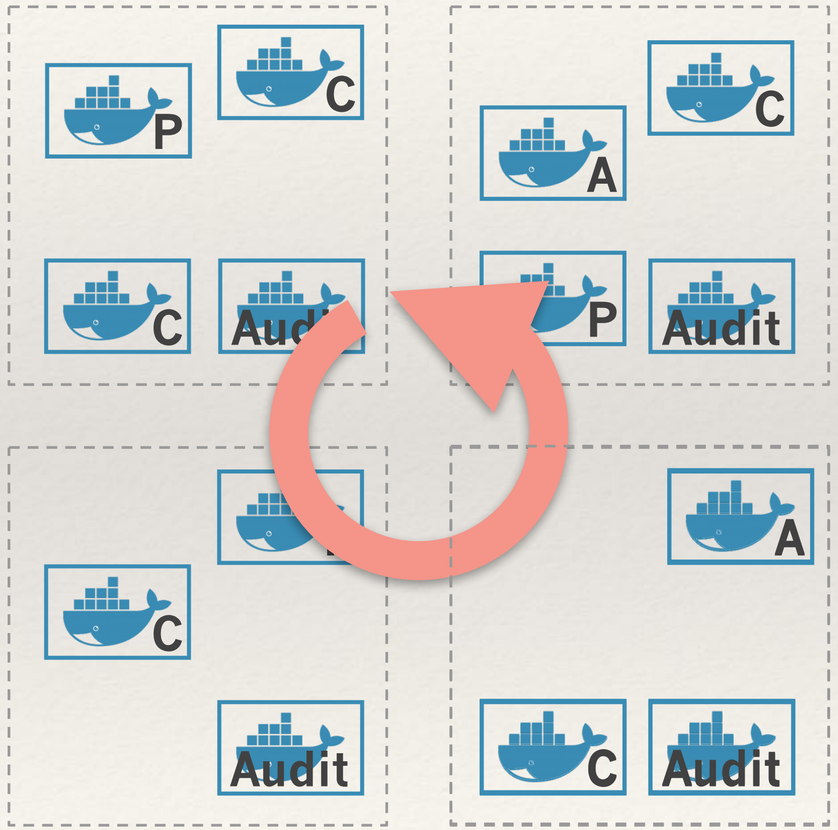


- Workflows:
  - Provision Machine
  - Install Operating System & Middleware
  - Install App
  - Configure Firewall
  - …
  - **Done!!!**

Functional Deployments

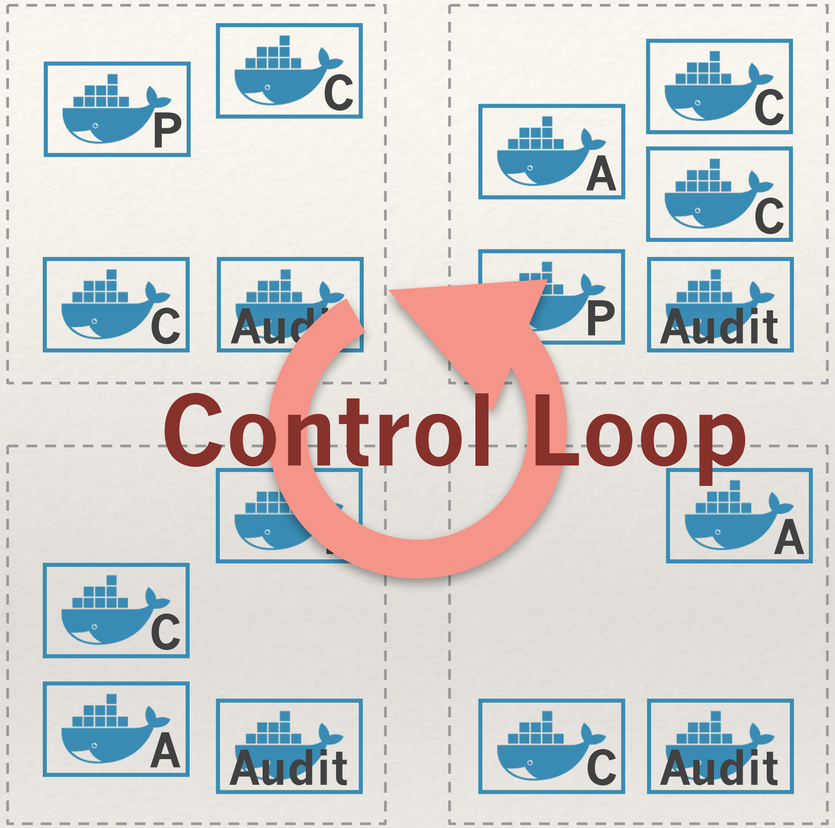
```
My app:
- image: posts
  type: pod
  replicas: 3
- image: connections
  type: pod
  replicas: 5
- image: aggregator
  type: pod
  replicas: 2
```

```
Compliance:
- image: audit
  type: daemonset
```



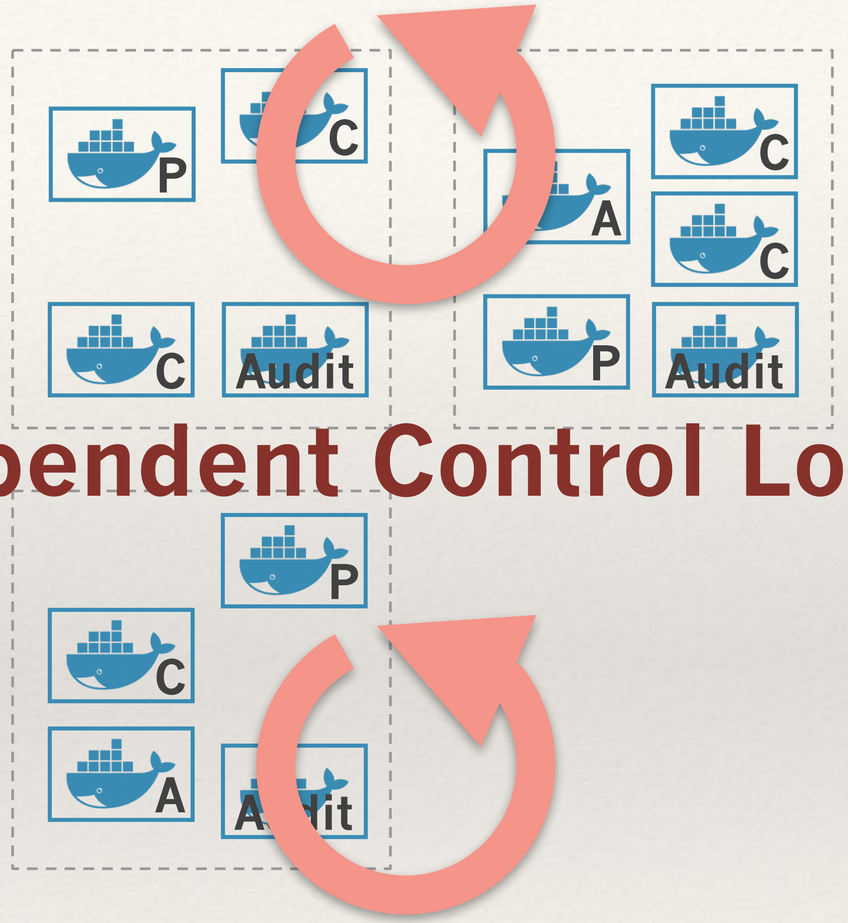
```
My app:
- image: posts
  type: pod
  replicas: 3
- image: connections
  type: pod
  replicas: 5
- image: aggregator
  type: pod
  replicas: 2
```

```
Compliance:
- image: audit
  type: daemonset
```



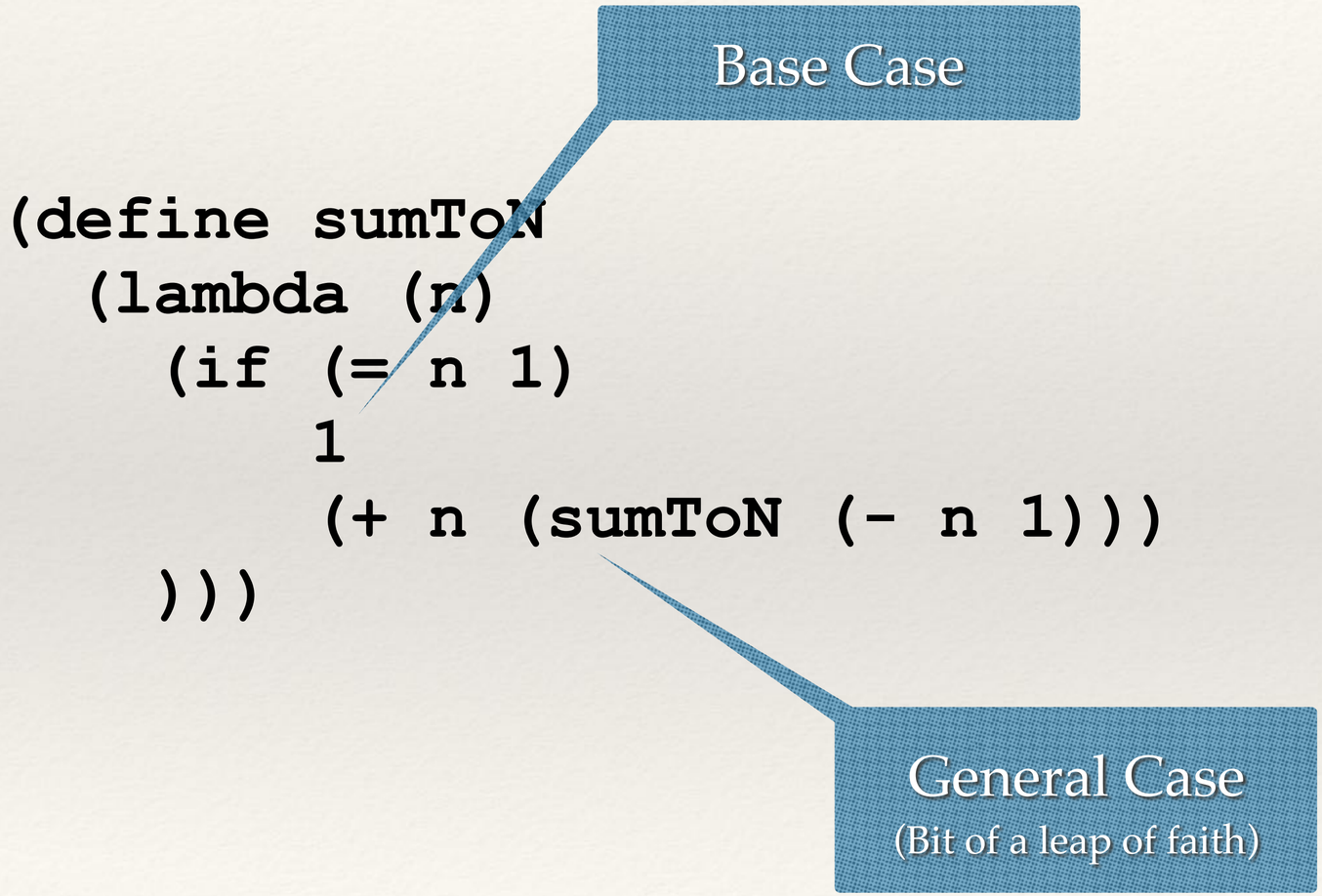
```
My app:
- image: posts
  type: pod
  replicas: 3
- image: connections
  type: pod
  replicas: 5
- image: aggregator
  type: pod
  replicas: 2
```

```
Compliance:
- image: audit
  type: daemonset
```



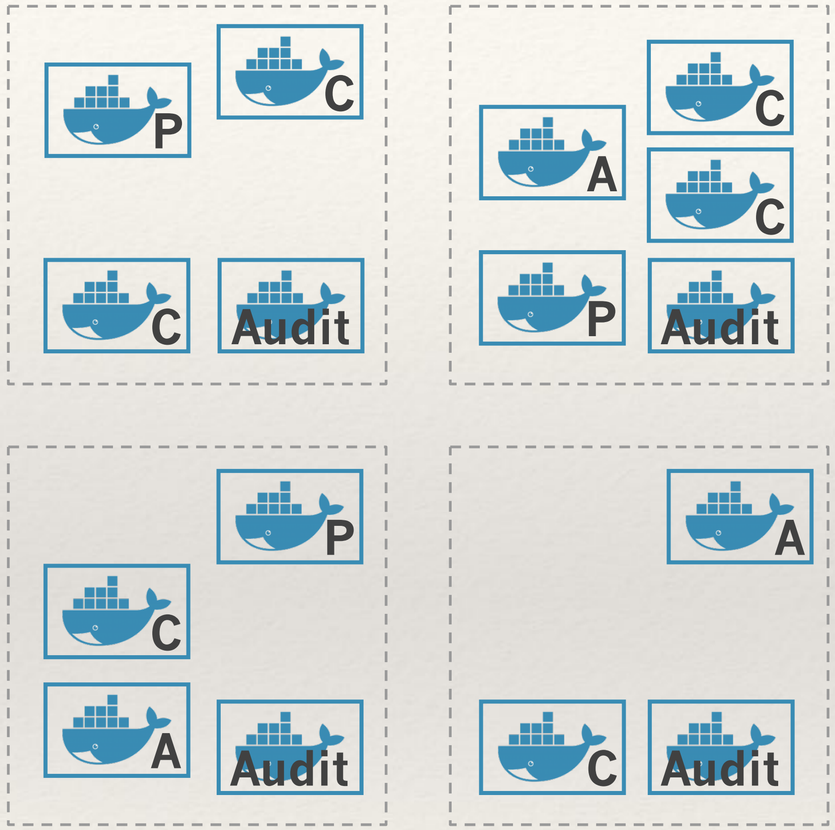
Independent Control Loops!!!

Functional Systems Management



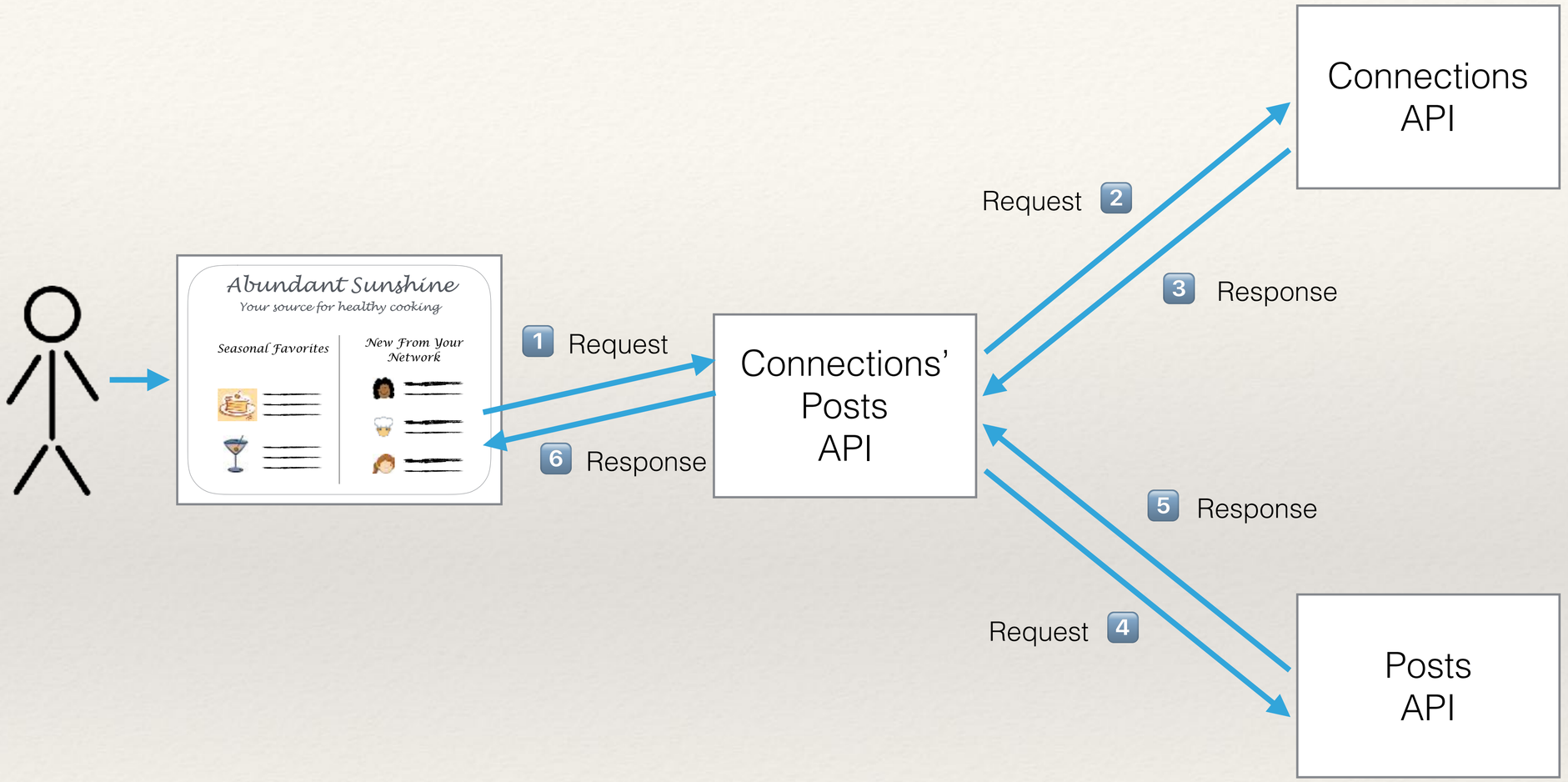
Common theme: Declarative & Strong Typing

- Eventual consistency
- Decoupled control loops

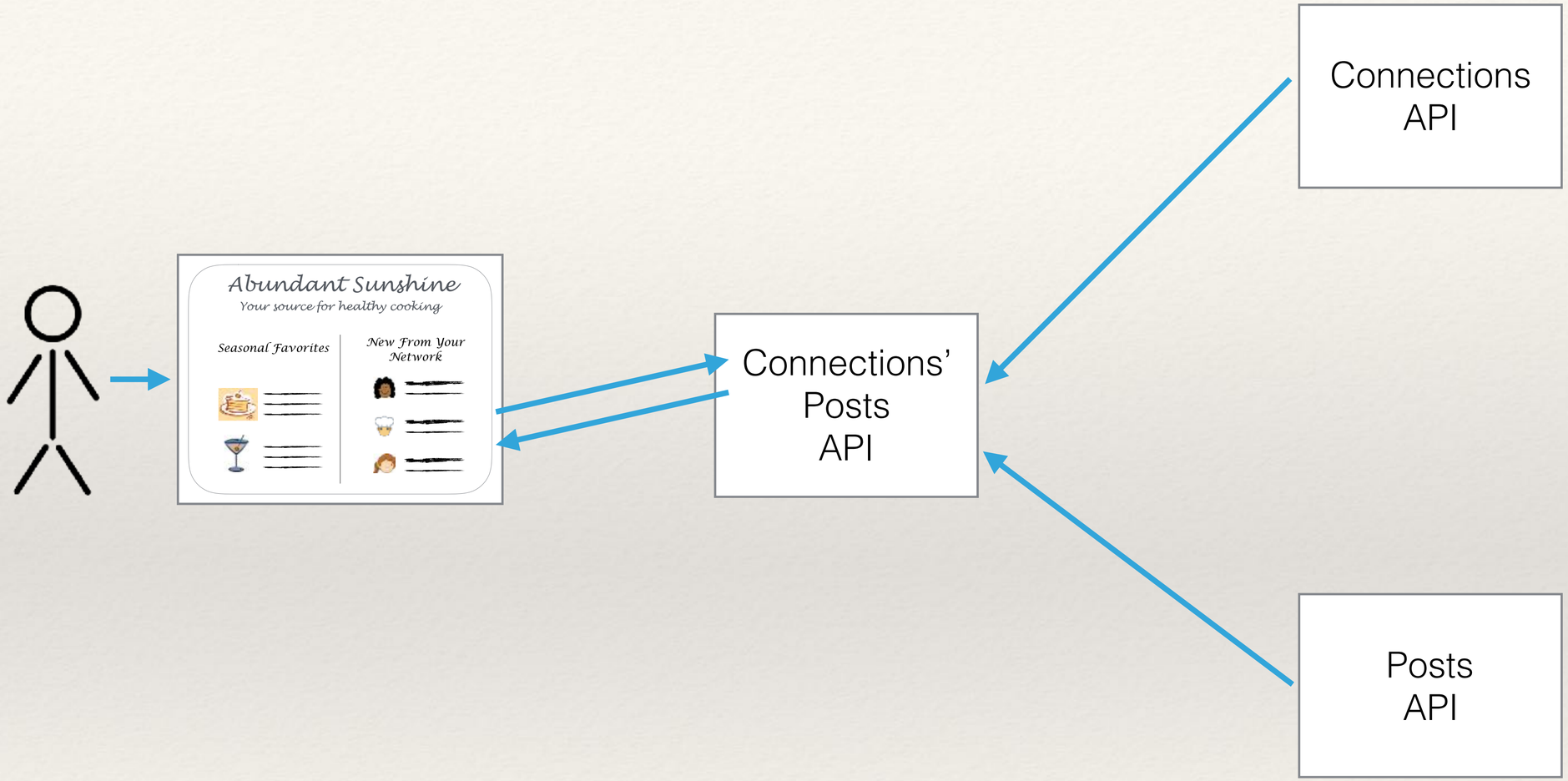


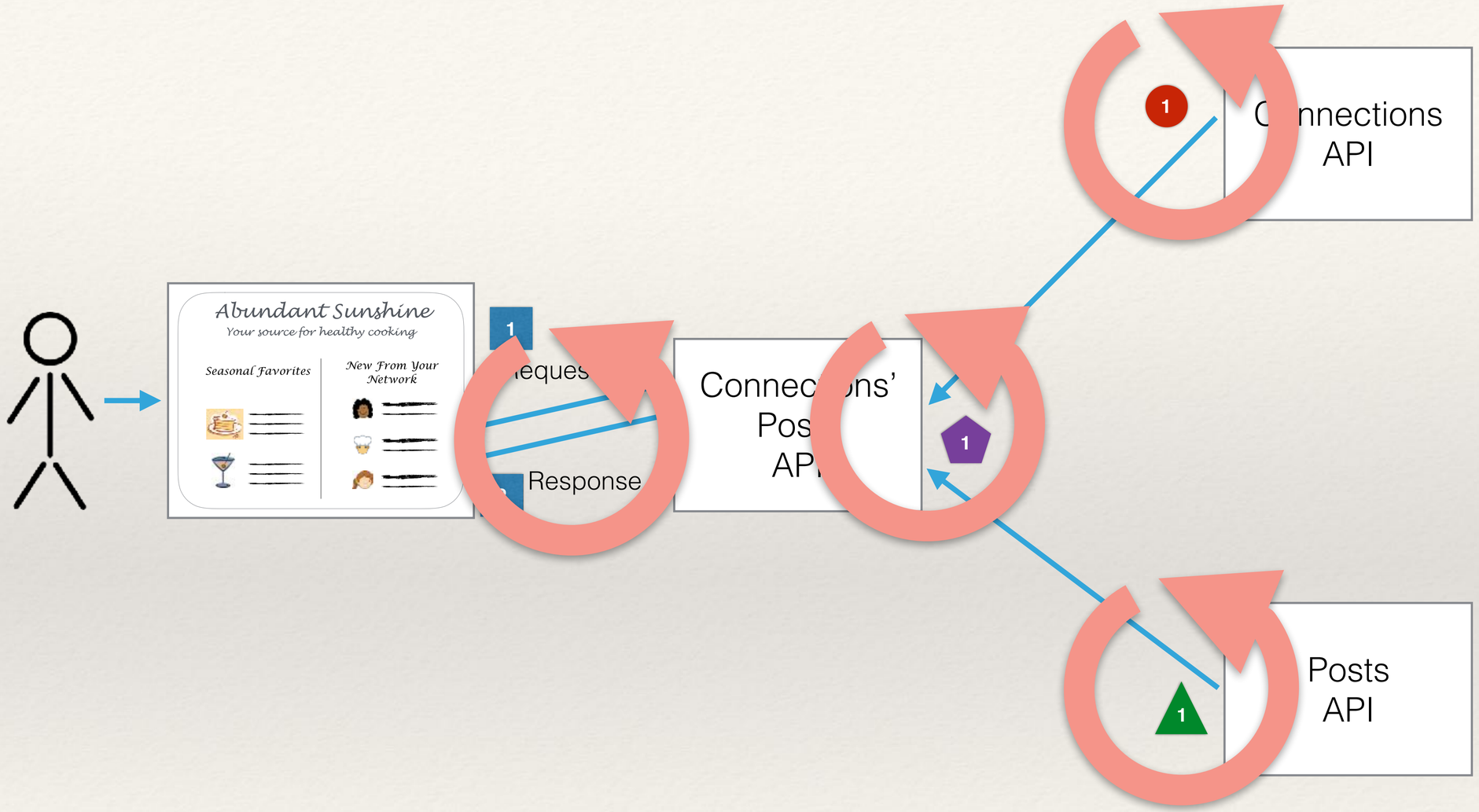
We're now operating

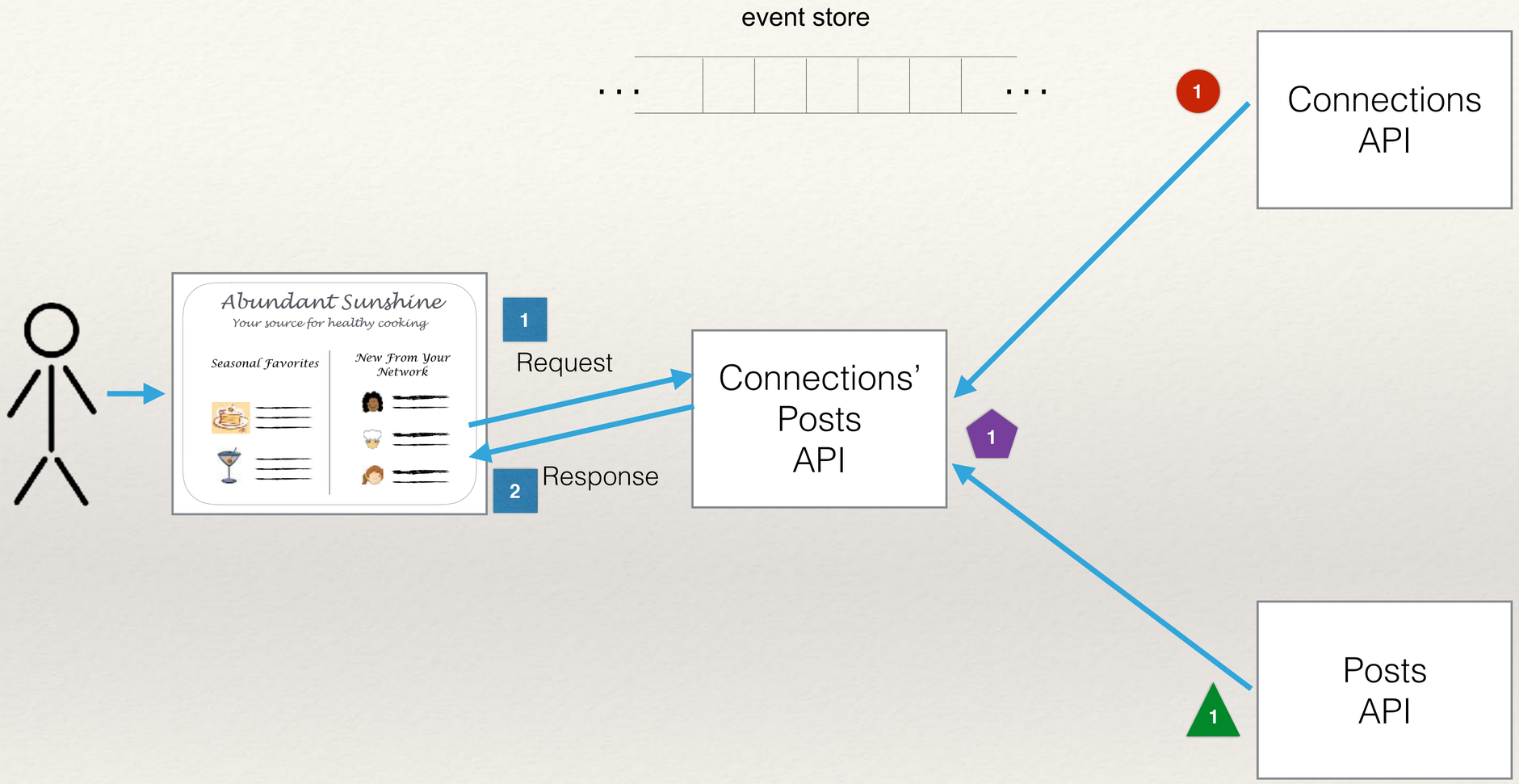
Distributed Systems

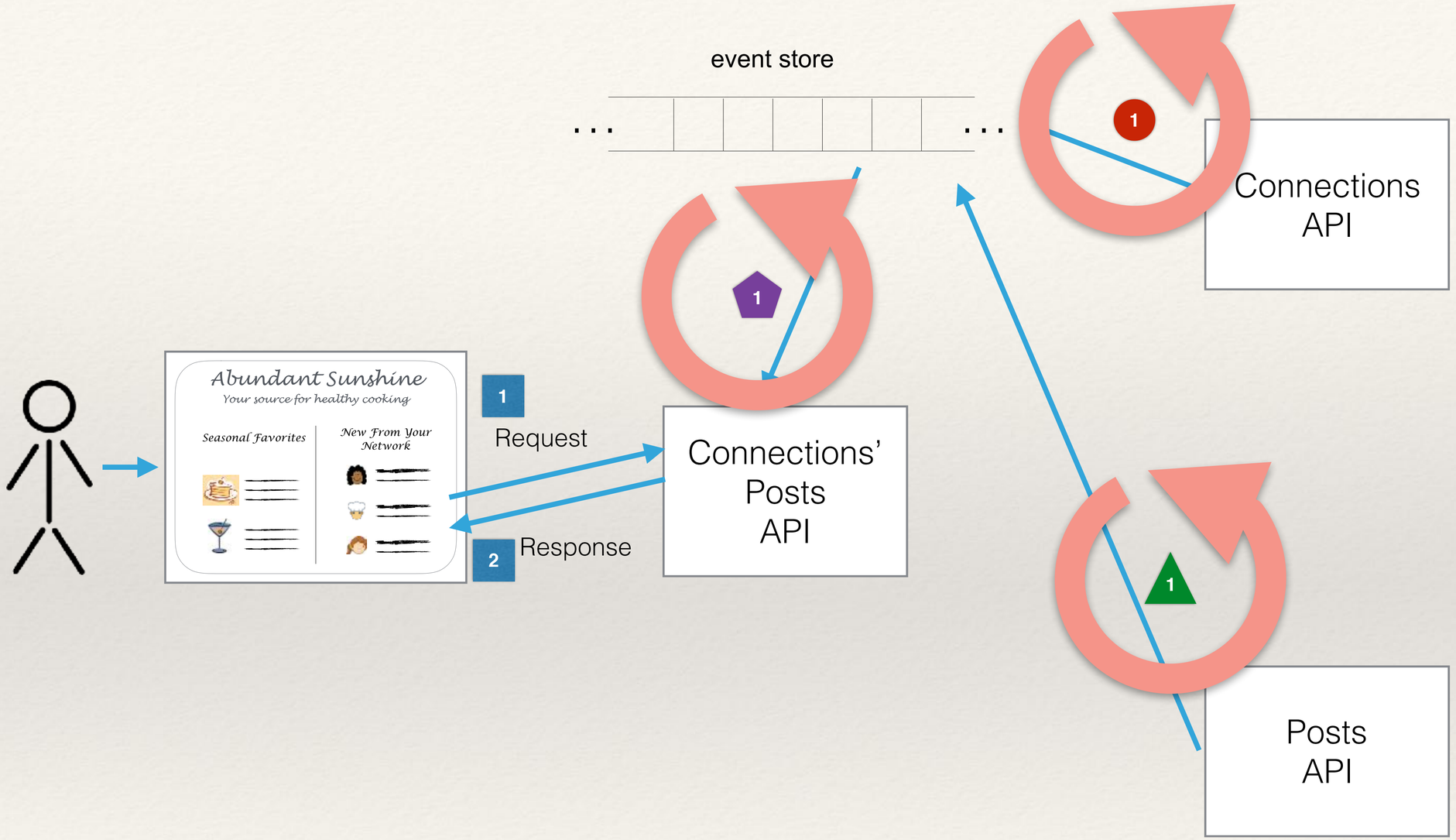

Abundant Sunshine
Your source for healthy cooking
Seasonal Favorites
New From Your Network
1 Request
6 Response
Connections' Posts API
Request 2
3 Response
Connections API
5 Response
Request 4
Posts API

Turning that on its head

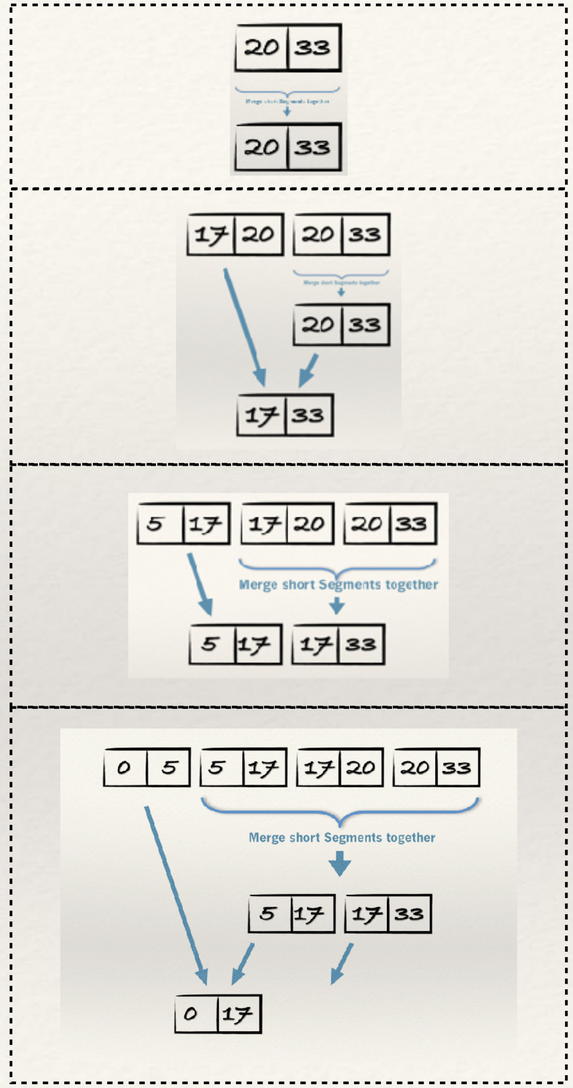



Abundant Sunshine
Your source for healthy cooking
Seasonal Favorites
New From Your Network
1
Response
Connections' Posts API
1
1
Connections API
1
Posts API


event store
Connections API
1
Abundant Sunshine
Your source for healthy cooking
Seasonal Favorites
New From Your Network
1
Request
2
Response
Connections' Posts API
1
Posts API
1


event store
Connections API
Posts API
Connections' Posts API
Abundant Sunshine
Your source for healthy cooking
Seasonal Favorites
New From Your Network
1
Request
2
Response
1
1
1

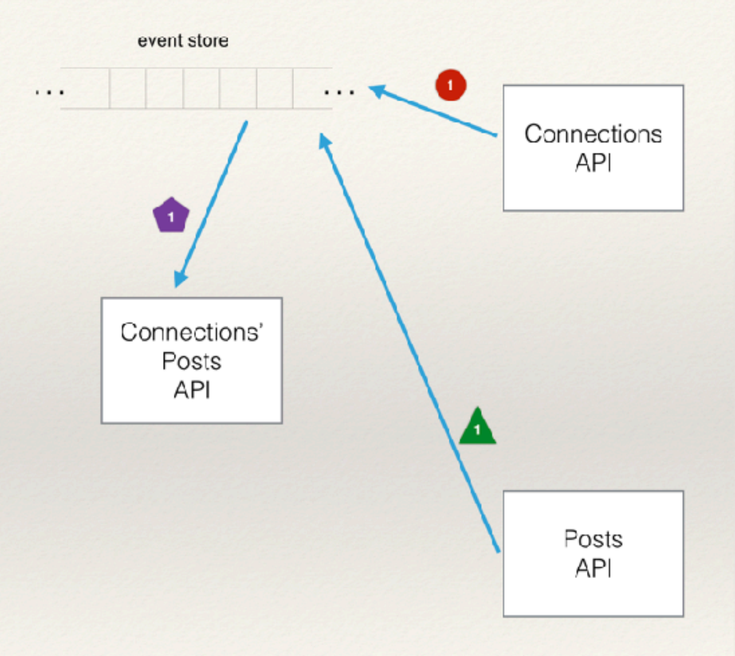
The call stack captures a series of events that result in state



Oh yeah!

Frames are <u>immutable</u>!

Event-driven, Distributed Systems



Common theme: Immutability & Continuations

- Independent Control loops
- Event log

Key Patterns

- Single deployable artifact **(immutability)**
- Trusted container pipeline and image management **(SDLC)**
- Aspect oriented programming **(composition)**
- Independent control loops **(declarative programming style & strong typing)**
- Event-driven architectures **(immutability & continuations)**


BEWARE!

```
(define (dontMergeSegmentsThisWayPlease segments threshold)
  (do ((listLen (vector-length segments))
       (i 0))
      ((>= i listLen))
      (if (< (segLen (vector-ref segments i)) threshold)
          (begin
            (vector-set! segments (+ i 1)
              (cons (car (vector-ref segments i))
                    (cdr (vector-ref segments (+ i 1)))))
            (set! segments (delVectorItem segments i))
            (set! listLen (- listLen 1)))
          (set! i (+ i 1))))
    segments)
```

What do I need help with?

Shifting from imperative to functional thinking can be hard.

Developers are getting it (or starting to).

How will we navigate this for ops-minded folks?

Thank you

@cdavisafc