DEVOPS ENTERPRISE SUMMIT

AN IT REVOLUTION EVENT

GET TOGETHER GO FASTER

October 22-24, 2018
The Cosmopolitan of Las Vegas
Las Vegas, Nevada

Electric Cloud IT REVOLUTION #DOES18

Taking Ops & Infrastructure from Iterative to Functional, Just like Dev

Cornelia Davis Sr. Director of Technology, Pivotal

DEVOPS ENTERPRISE SUMMIT
AN IT REVOLUTION EVENT

Taking Ops & Infrastructure From Iterative To Functional

(Just Like Dev)

Cornelia Davis
Sr. Director of Technology
Pivotal

@cdavisafc

Me?

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

Ops

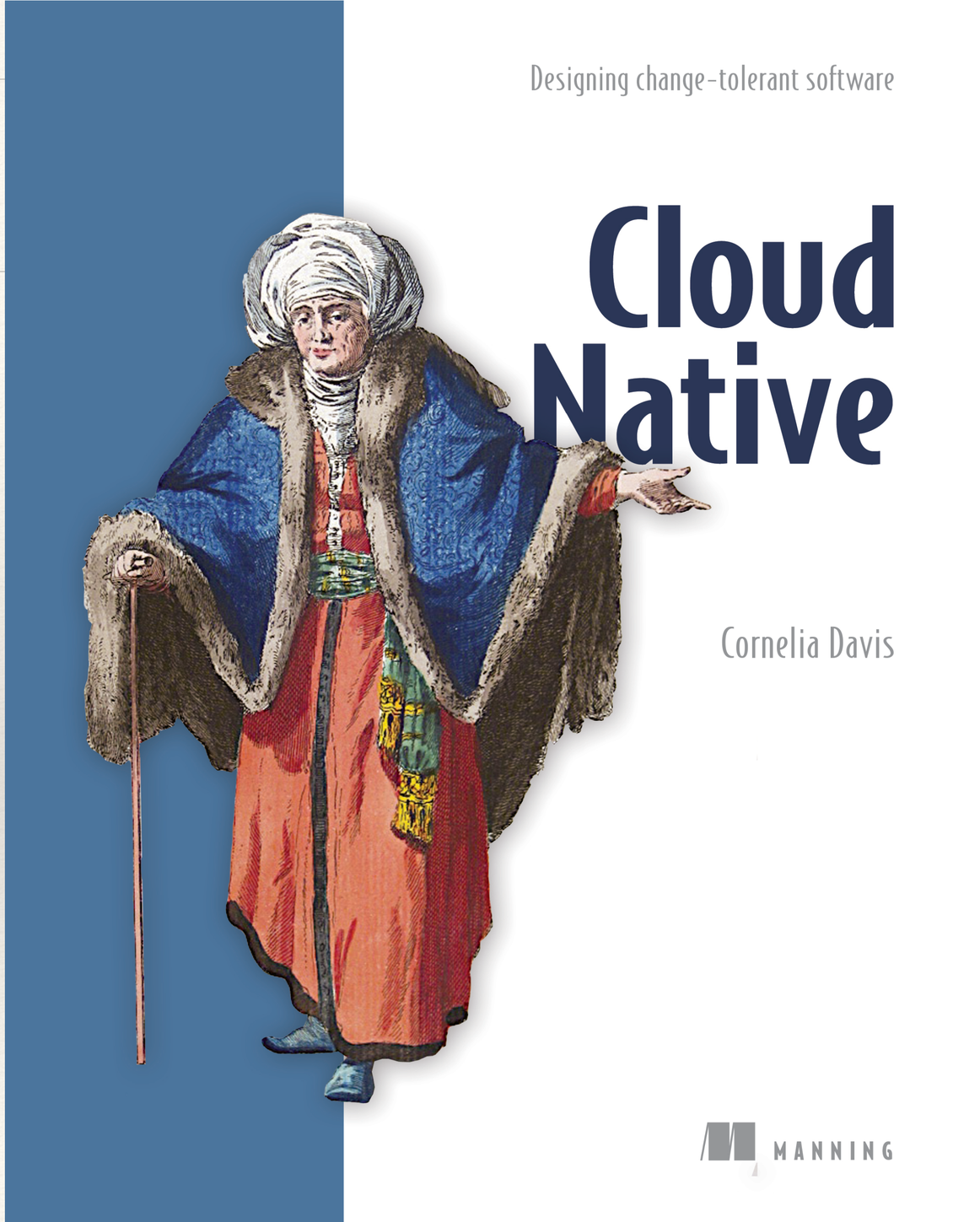
as much as

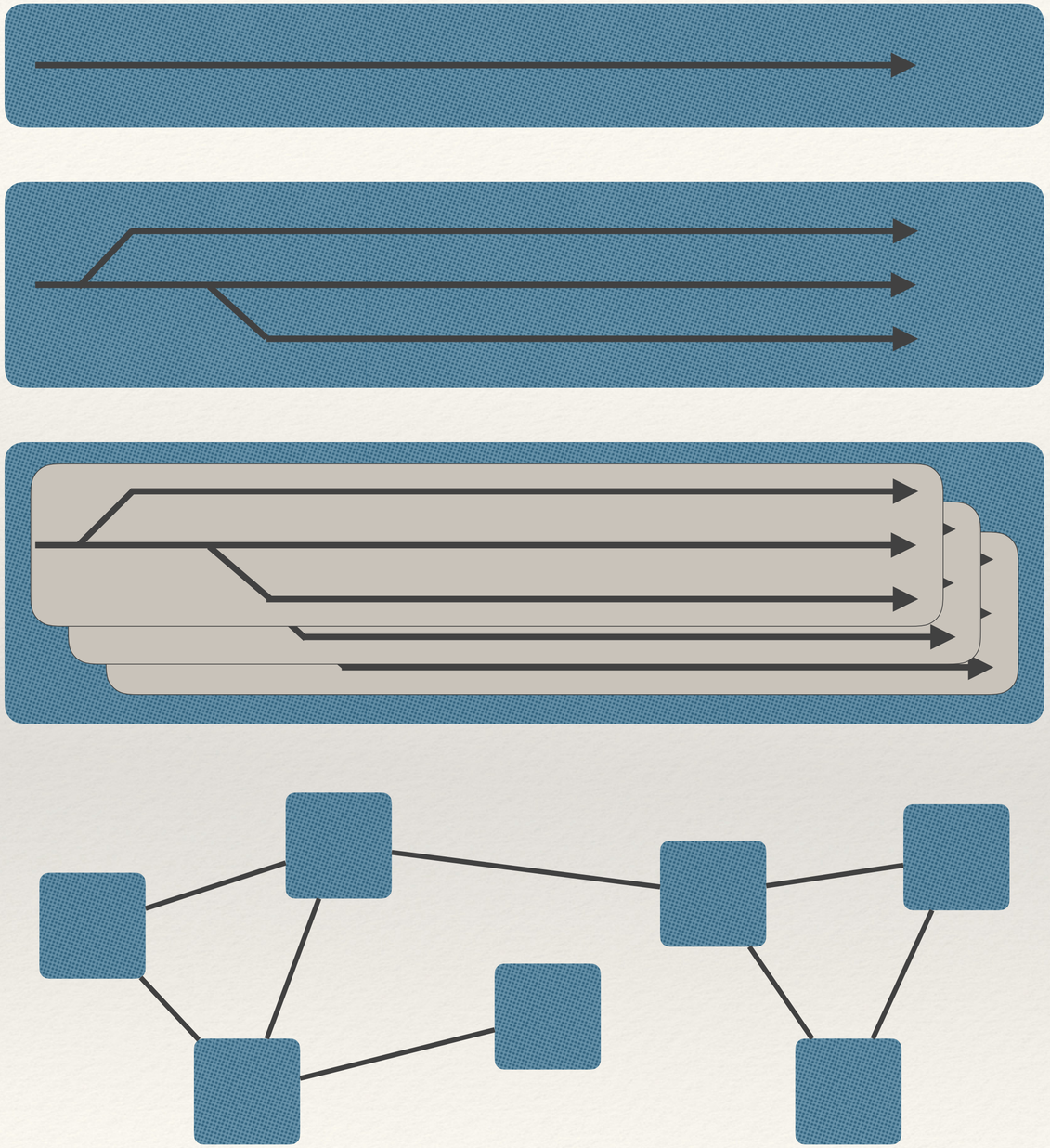
Dev

now

- And now…

Discount code 40% off!: 40cloudnat





Assembly Language

Fortran

C

C++

Java

Golang

Imperative Programming

Clojure

Scala

Kotlin

F#

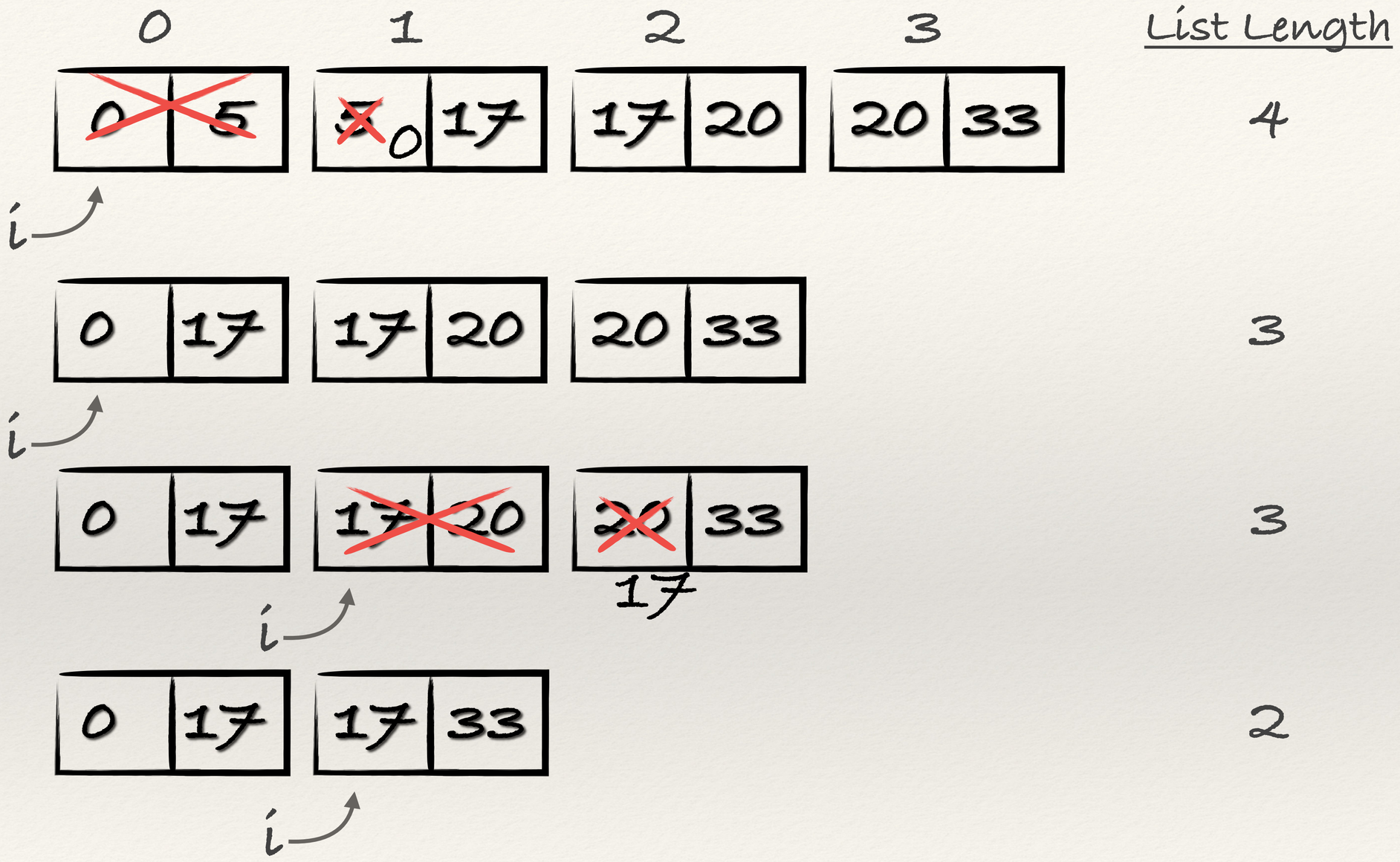
Functional Programming

```
[(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
```

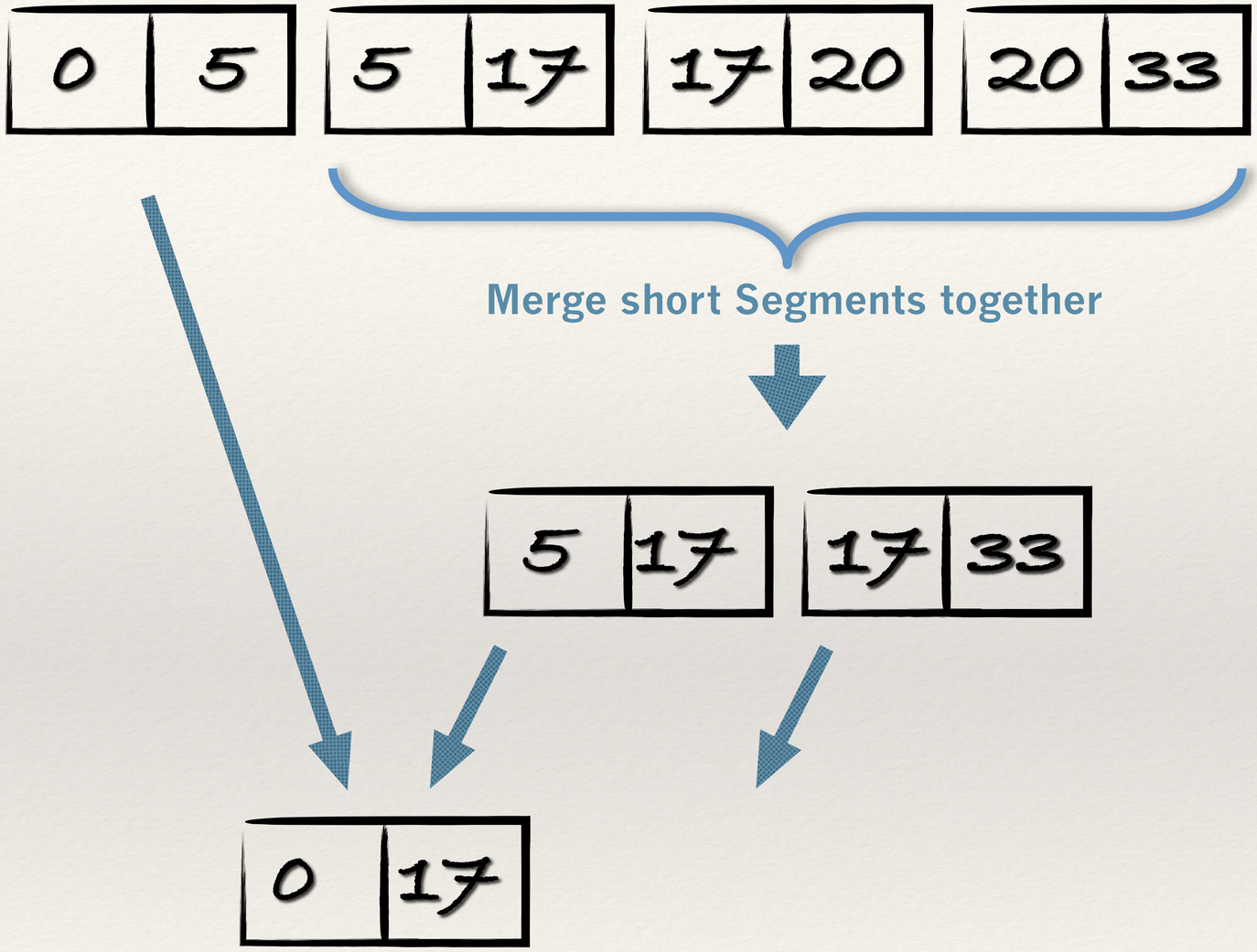

0
1
2
3
List Length
0 5
5 0 17
17 20
20 33
4
i
0 17
17 20
20 33
3
i
0 17
17 20
20 33
17
3
i
0 17
17 33
2
i

Hallmarks of Imperative Programs

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

- Hairy edge cases
- Difficult to parallelize

```
(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
- Recursive

- 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

Gene Kim

What is the

model

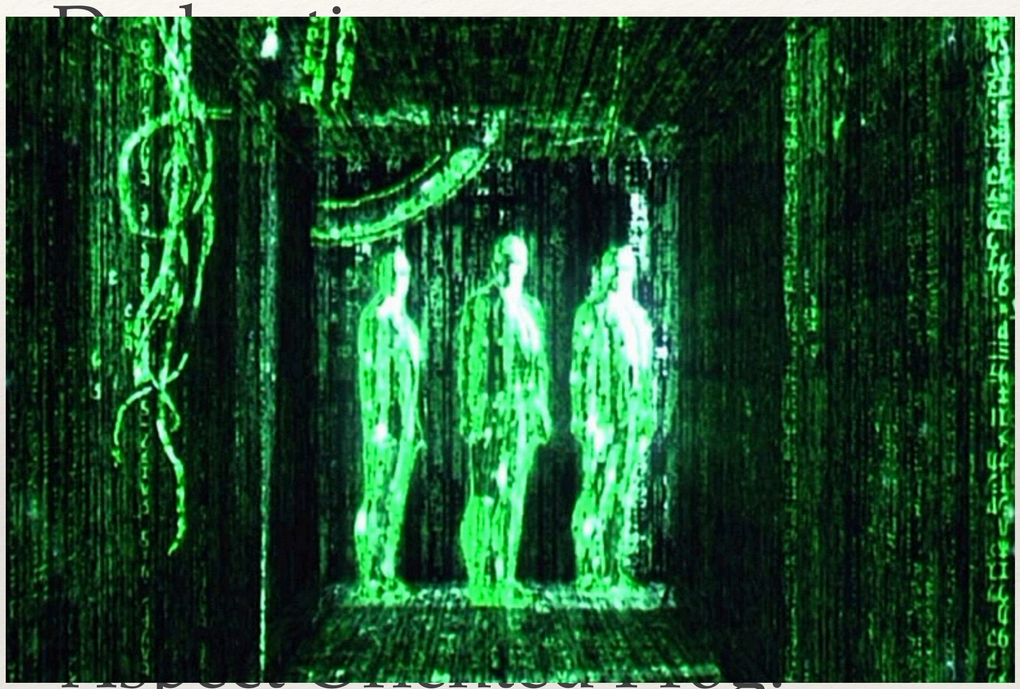
we use to reason with?

Imperative → Functional Programming

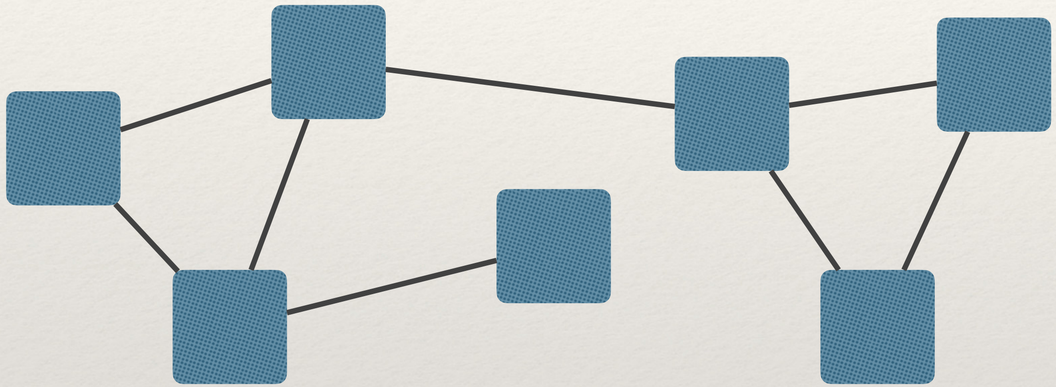
Iterative	→	Declarative
Mutable State	→	Immutability
Looping	→	Recursive
Libraries (compiled in)	→	Aspect Oriented Prog.*

* 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.



Imperative Systems Programming

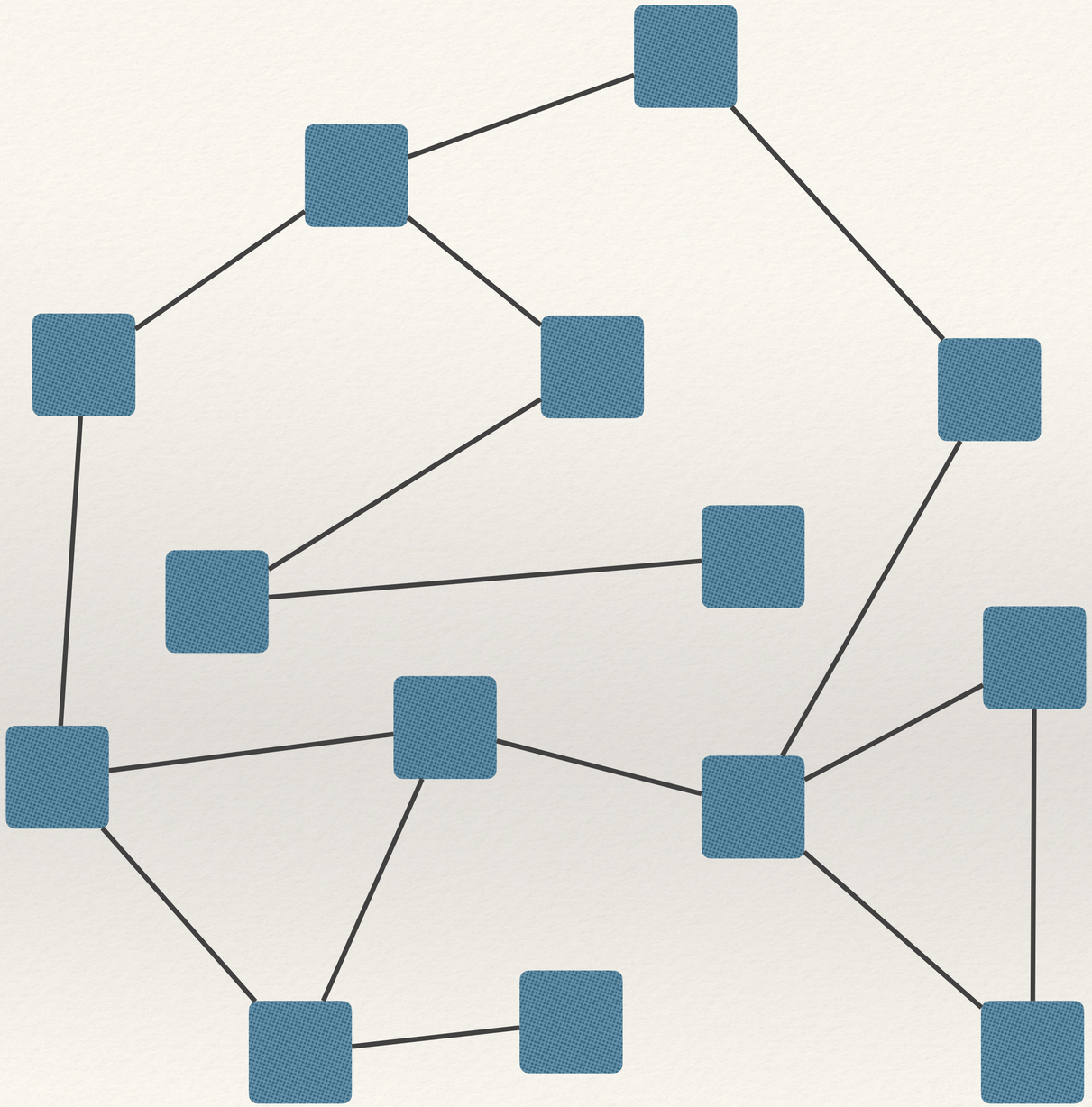
Bash

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

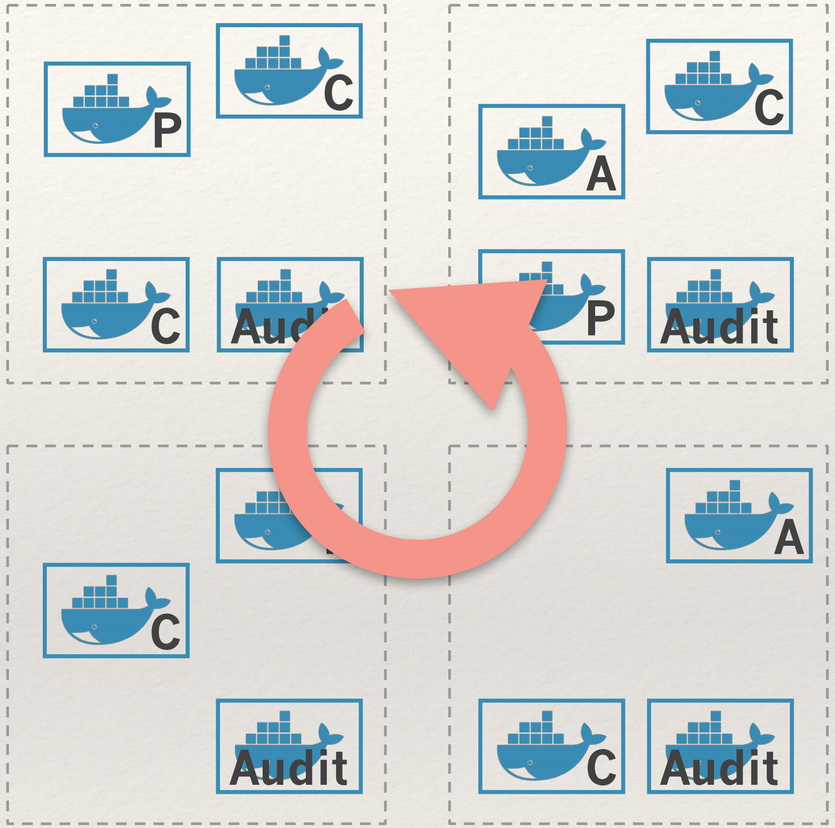
a model

that allows machines to reason for us.

Declarative Deployments

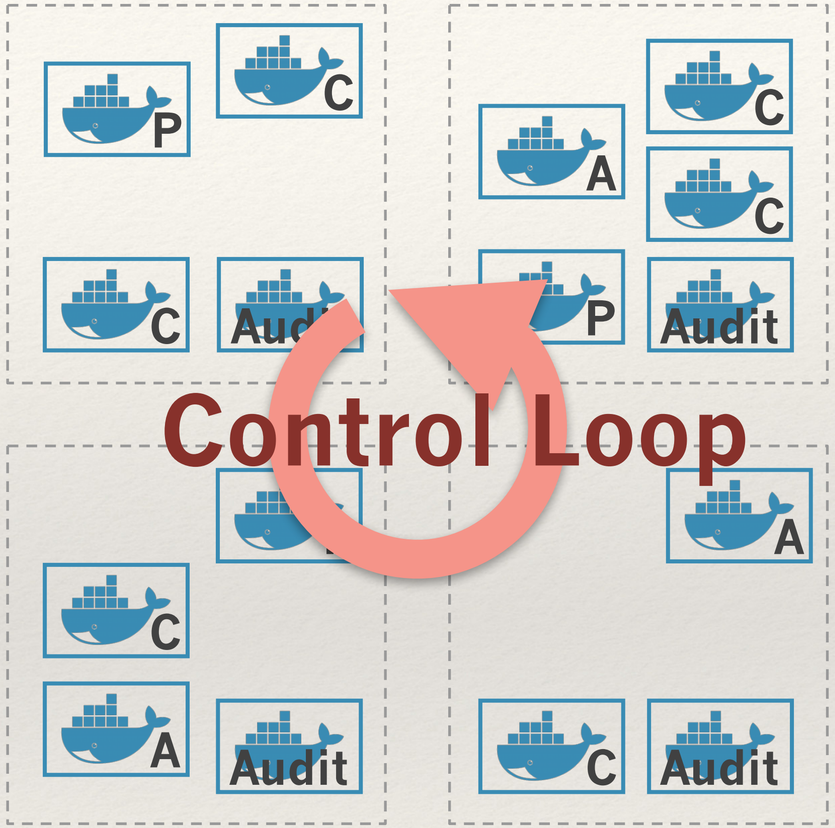
```
My app:
- image: posts
  type: pod
  replicas: 3
- image: connections
  type: pod
  replicas: 5
- image: aggregator
  type: pod
  replicas: 2
```

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



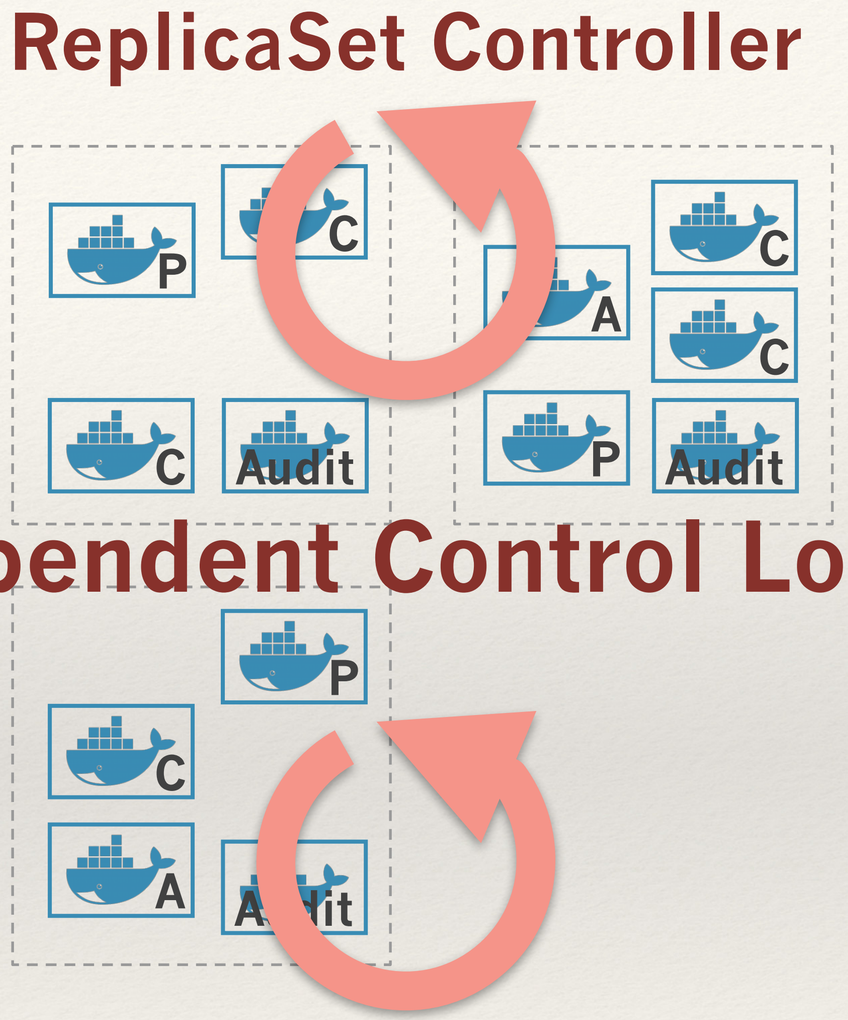
```
My app:
- image: posts
  type: pod
  replicas: 3
- image: connections
  type: pod
  replicas: 5
- image: aggregator
  type: pod
  replicas: 2
```

```
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!!!

A bit like Functional Programming

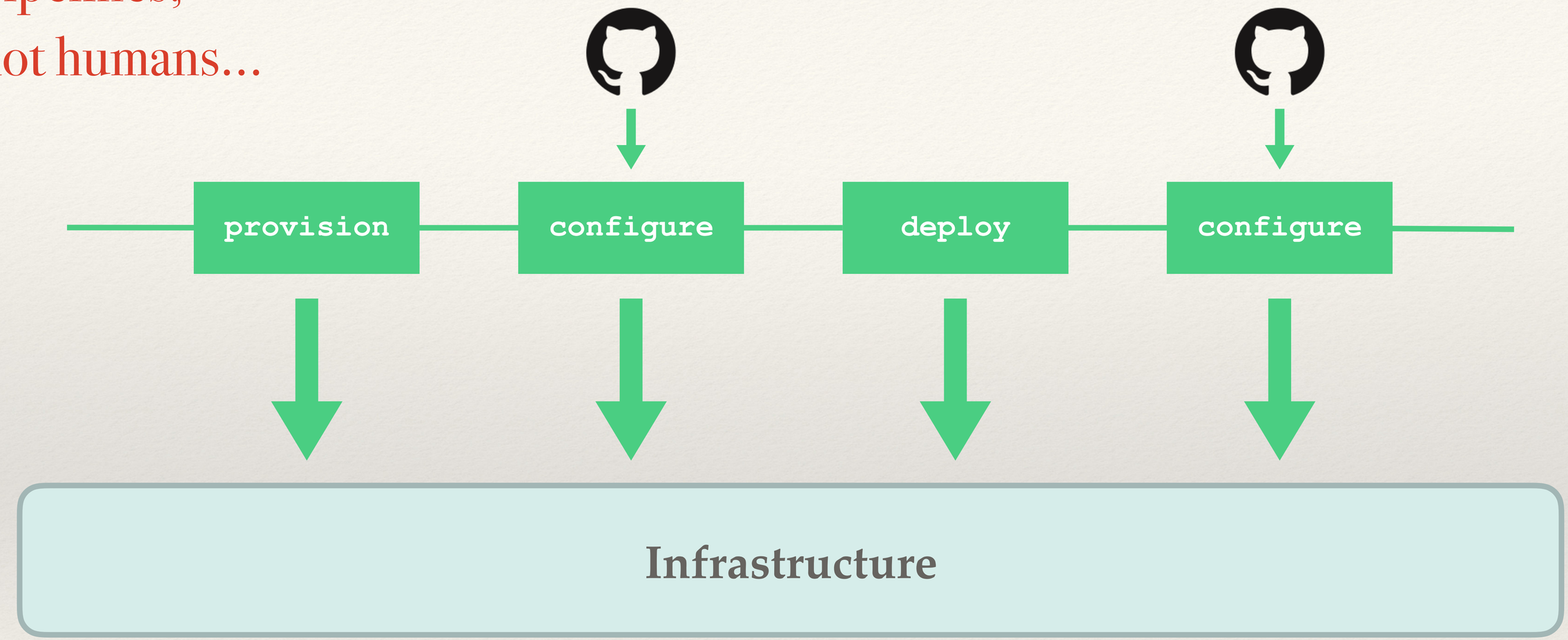
```
(define sumToN
  (lambda (n)
    (if (= n 1)
        1
        (+ n (sumToN (- n 1)))
    )))
```

Base Case

General Case
(Bit of a leap of faith)

Immutable Infrastructure

Pipelines,
not humans…

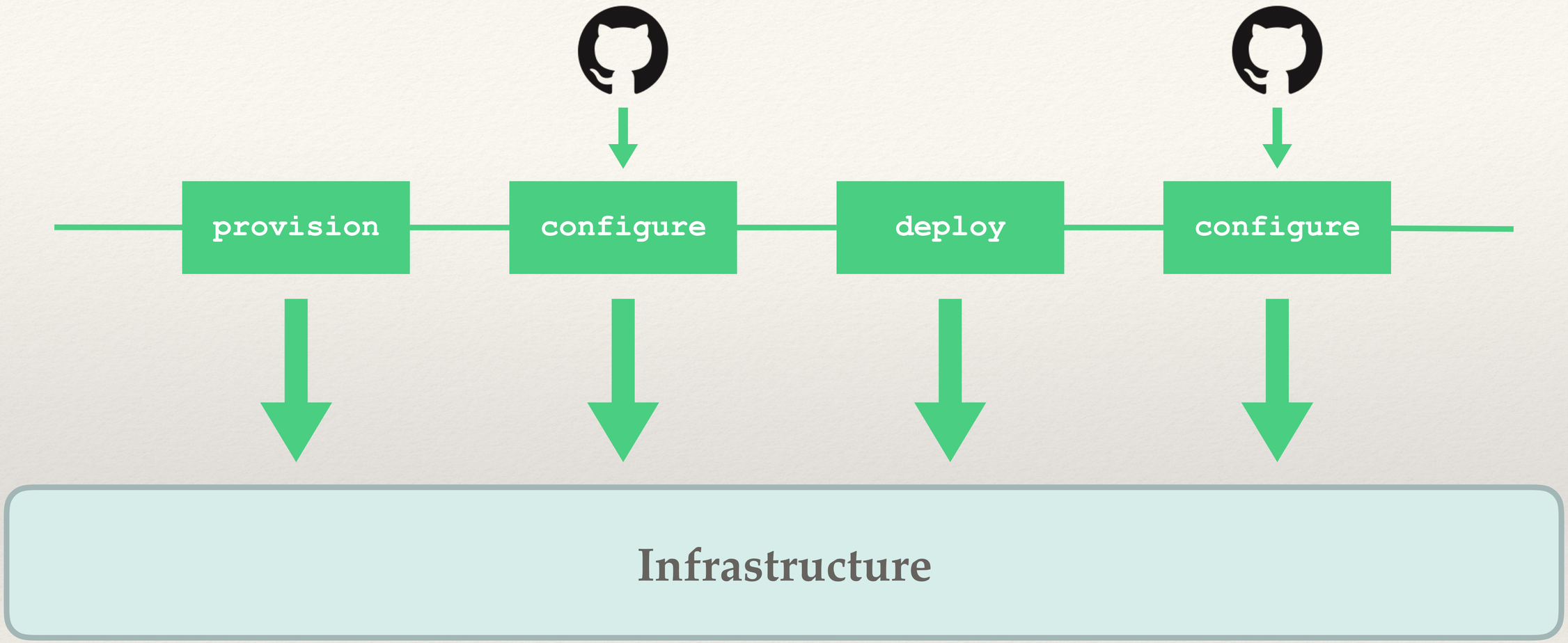


Challenges Remain

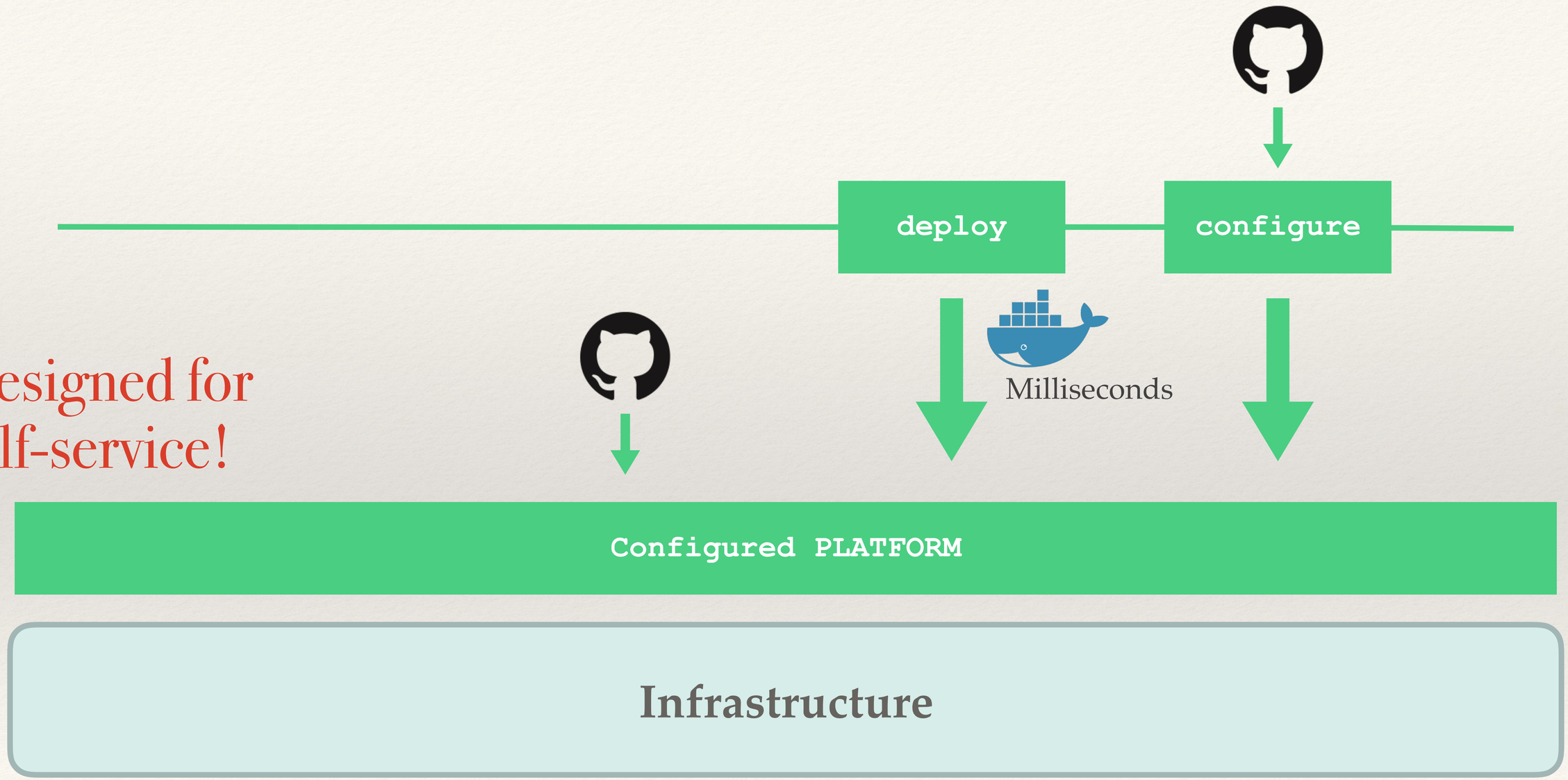
Slow → **Shortcuts** → ❄

ssh (eeghads!)

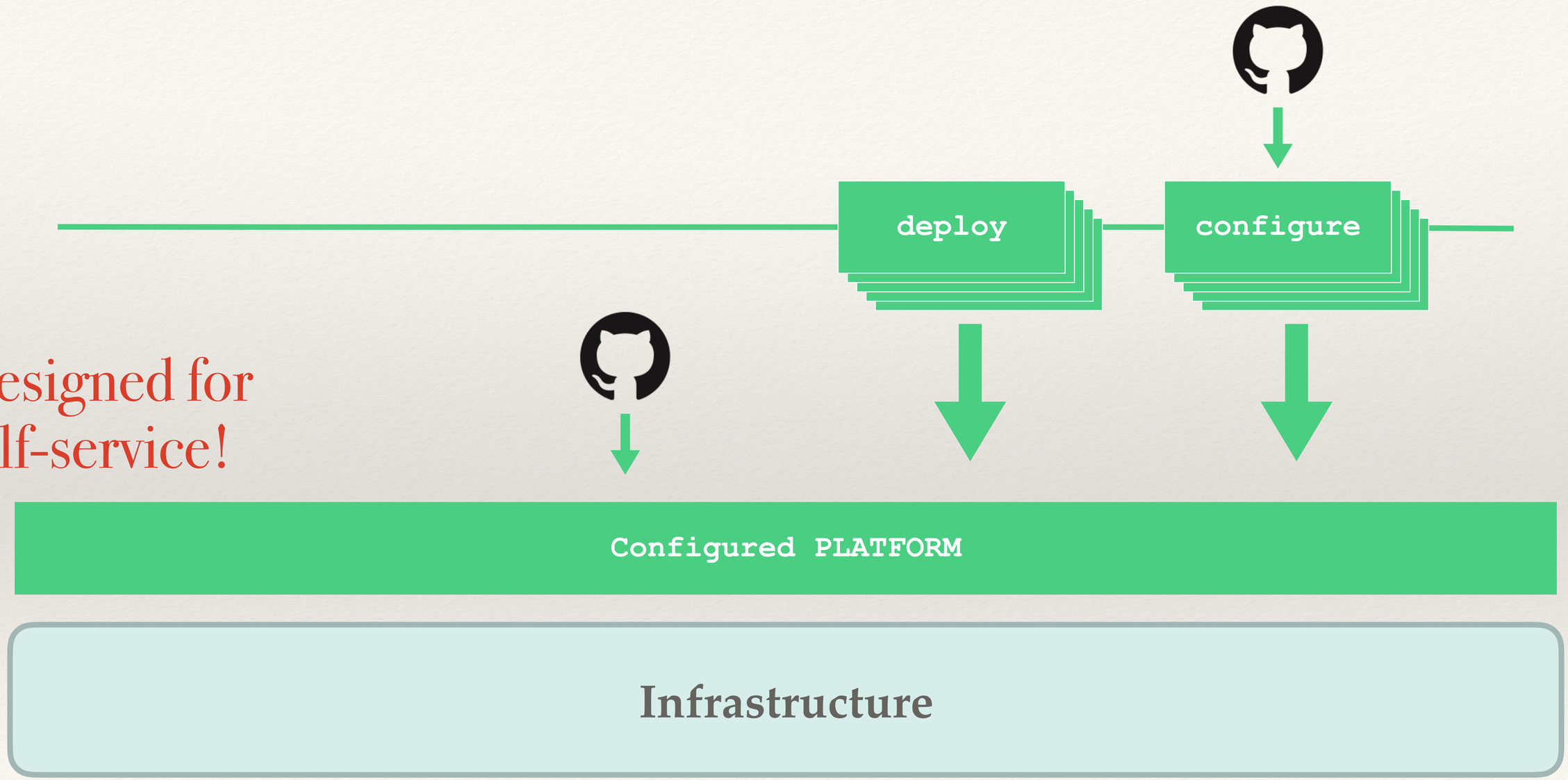
Complex Dependencies → ❄

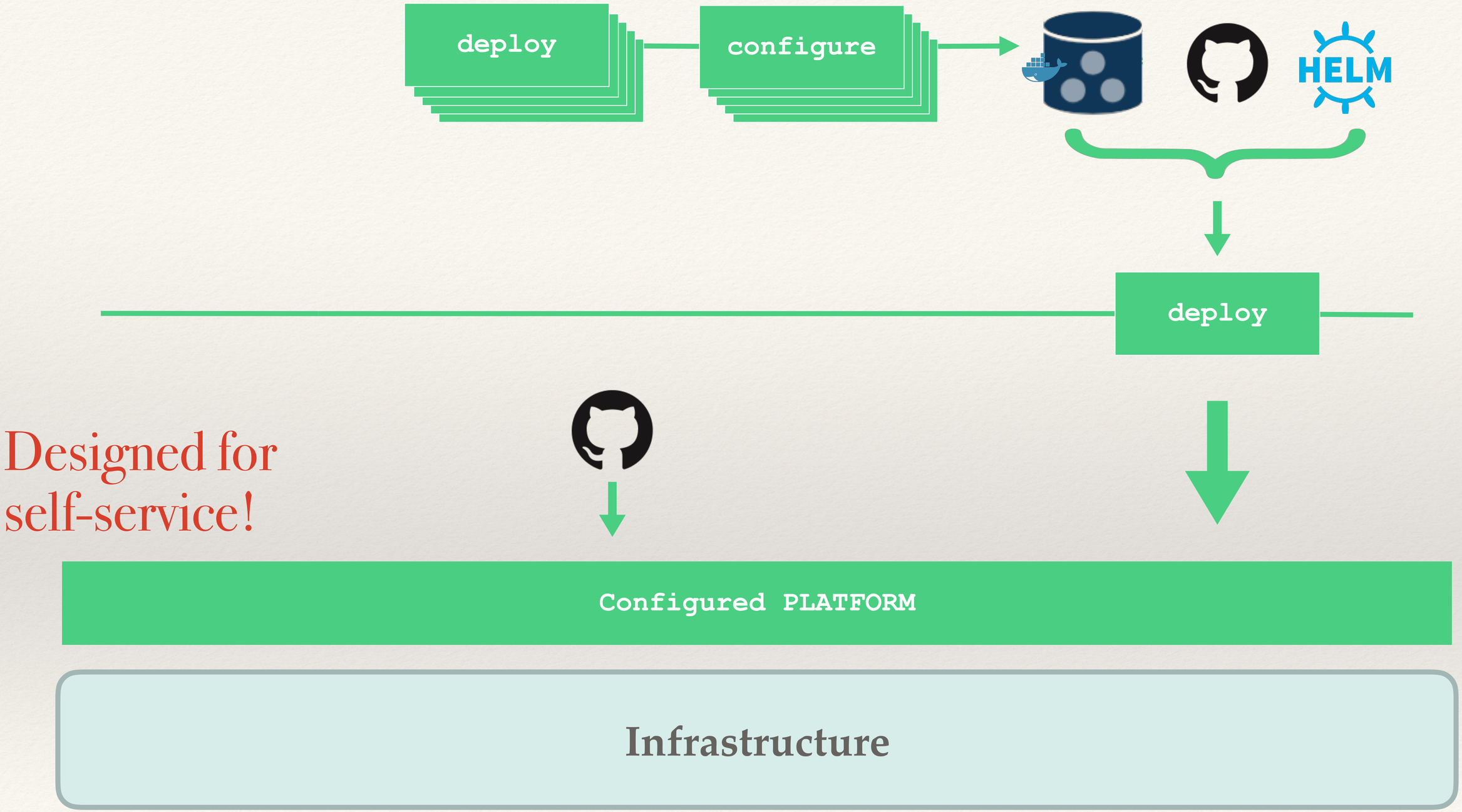

provision
configure
deploy
configure
Infrastructure

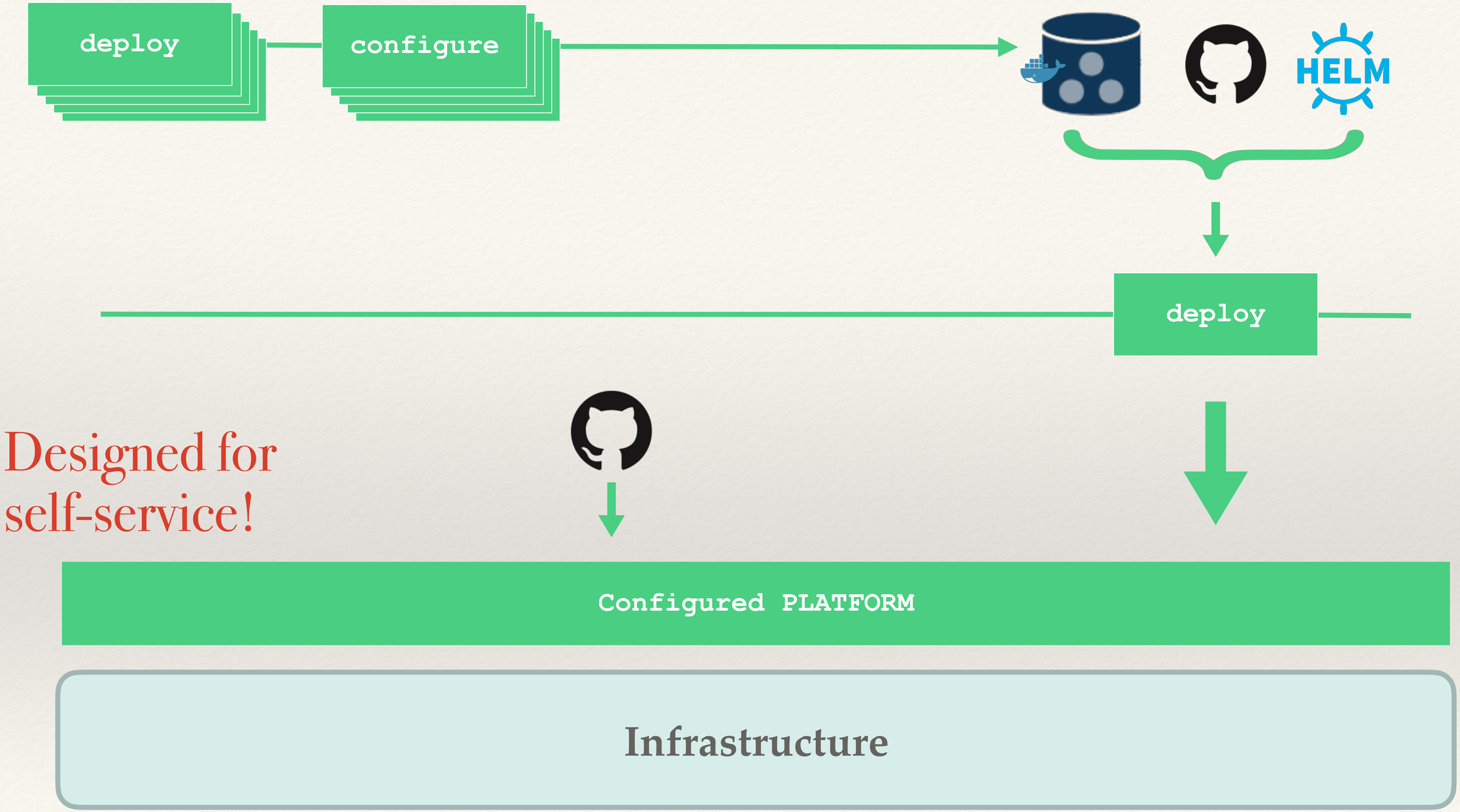
Designed for self-service!



Designed for self-service!



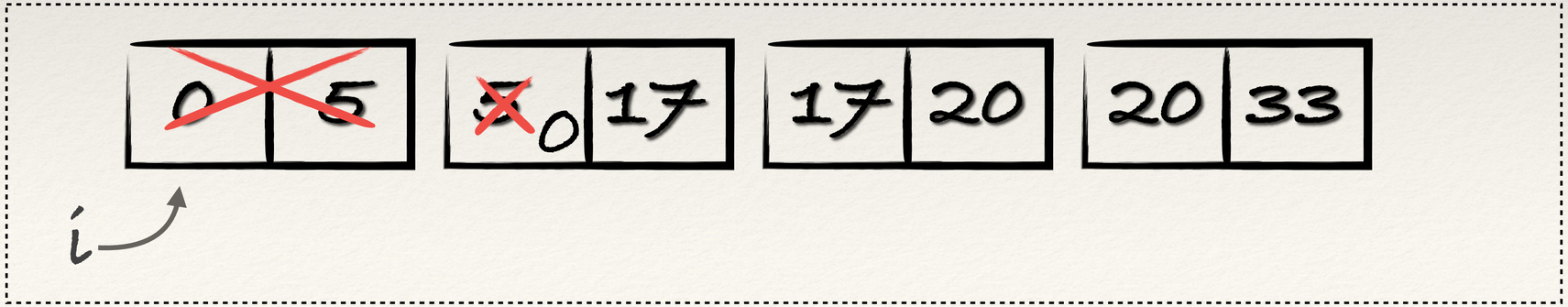

deploy
configure
HELM
deploy
Designed for self-service!
Configured PLATFORM
Infrastructure


deploy
configure
HELM
deploy
Designed for self-service!
Configured PLATFORM
Infrastructure

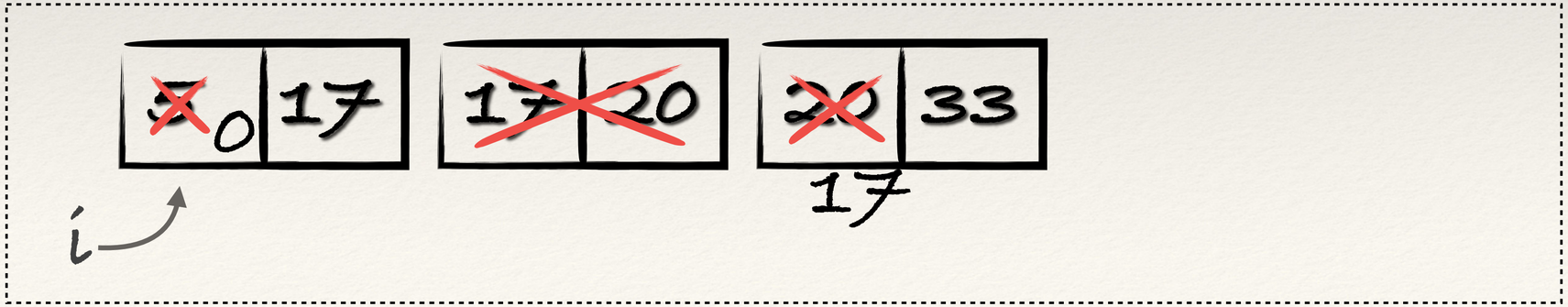
But what does all of this have to do with

Imperative and Functional Models?

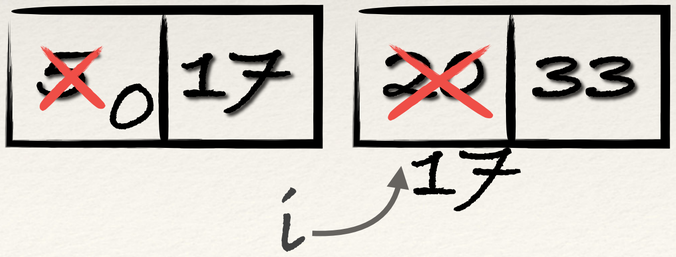
State:



State:



State:



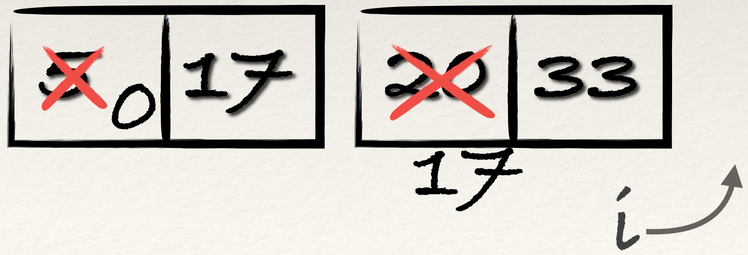
Correctness?

Other Influences?

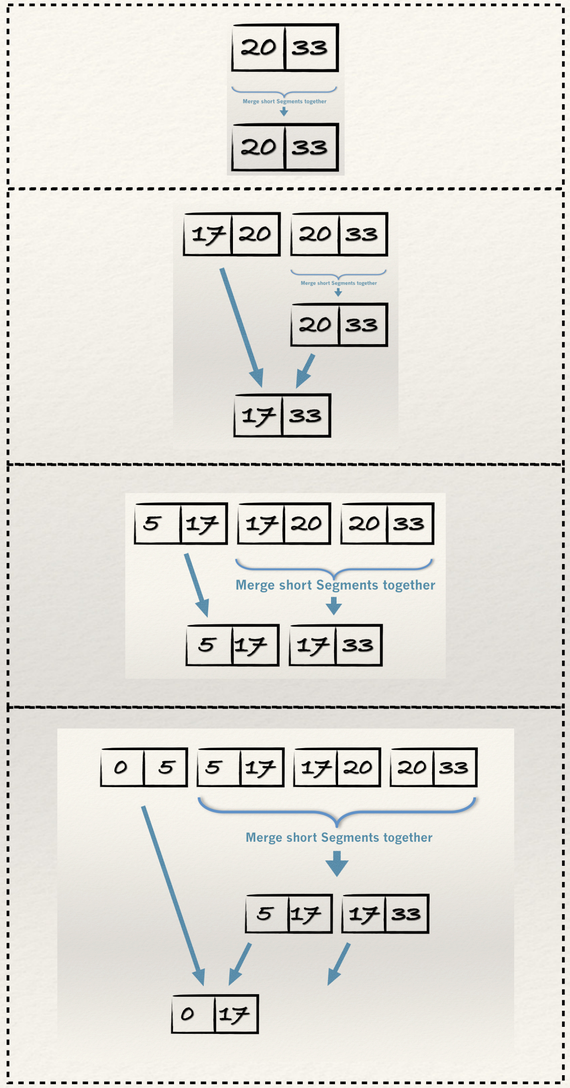
```
def mergeSegments(s_list, threshold):
  list_len = len(s_list)
  i = 0
  while i < list_len:
    seg_length = s_list[i][1]-s_list[i][0]
    if seg_length < threshold:
      s_list[i+1] =
            (s_list[i][0], s_list[i+1][1])
      del s_list[i]
      list_len -= 1
    else:
      i += 1
```

?

State:

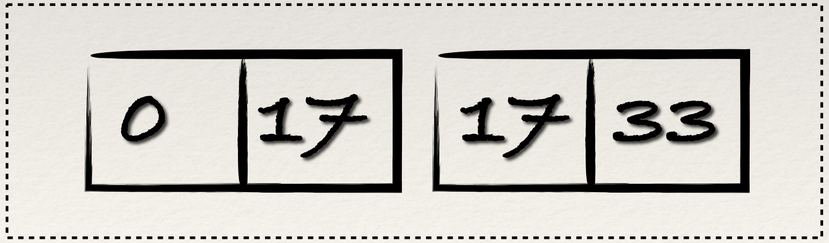


The call stack captures a series of events from which state can be derived

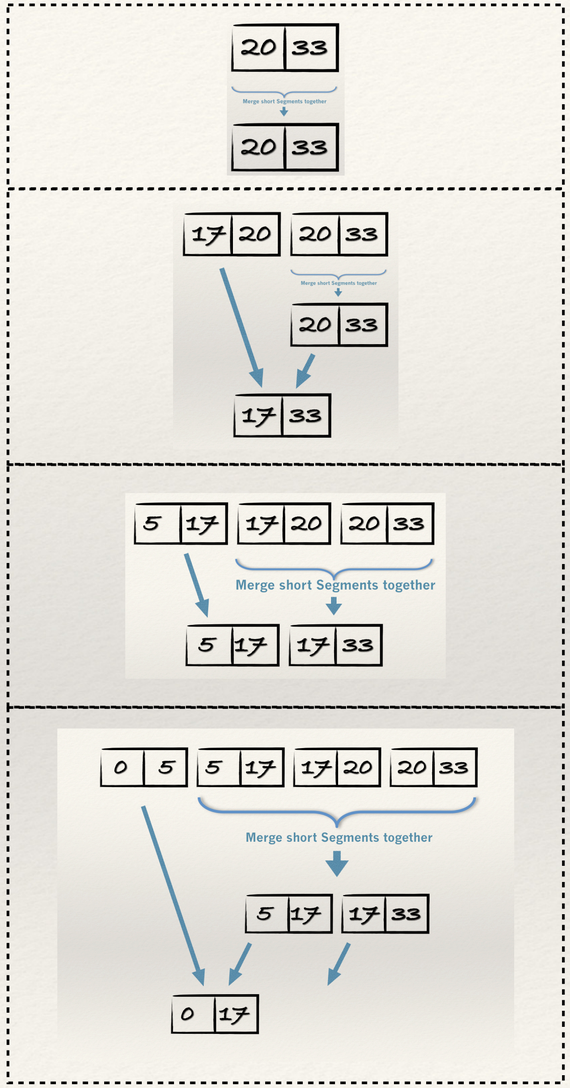


Frames are <u>immutable</u>!

Final State:

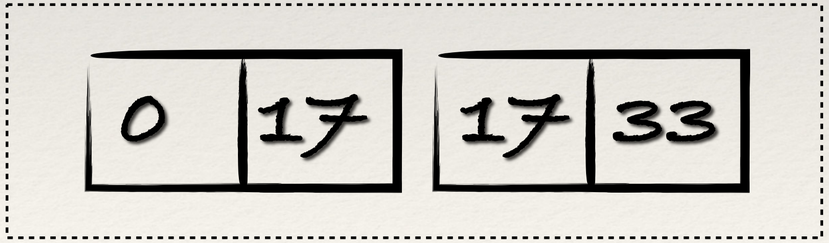


The call stack captures a series of events from which state can be derived

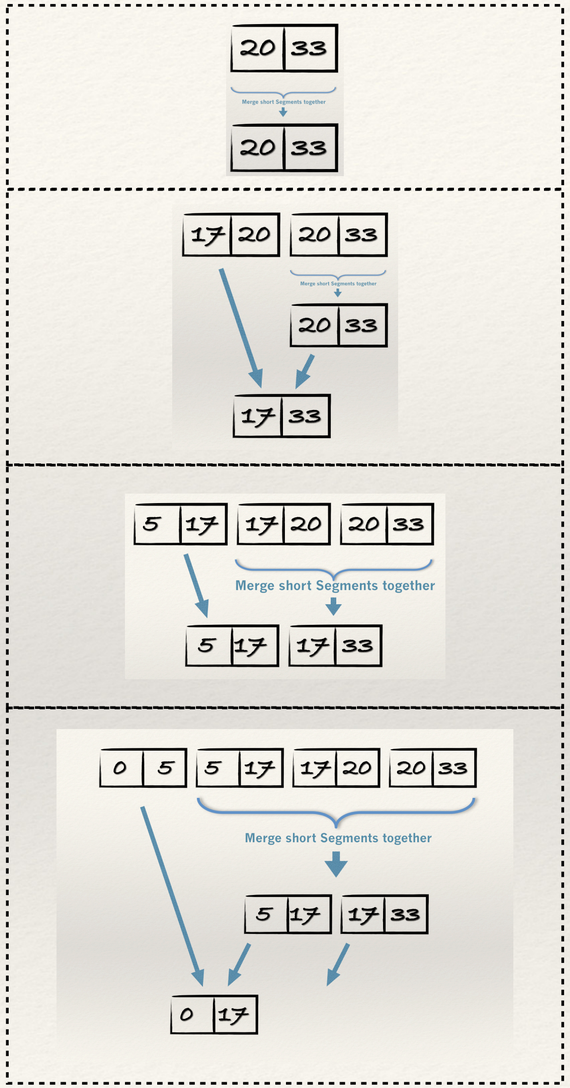


Frames are <u>immutable</u>!

Final State:



Container images are a bit like your stack frames…



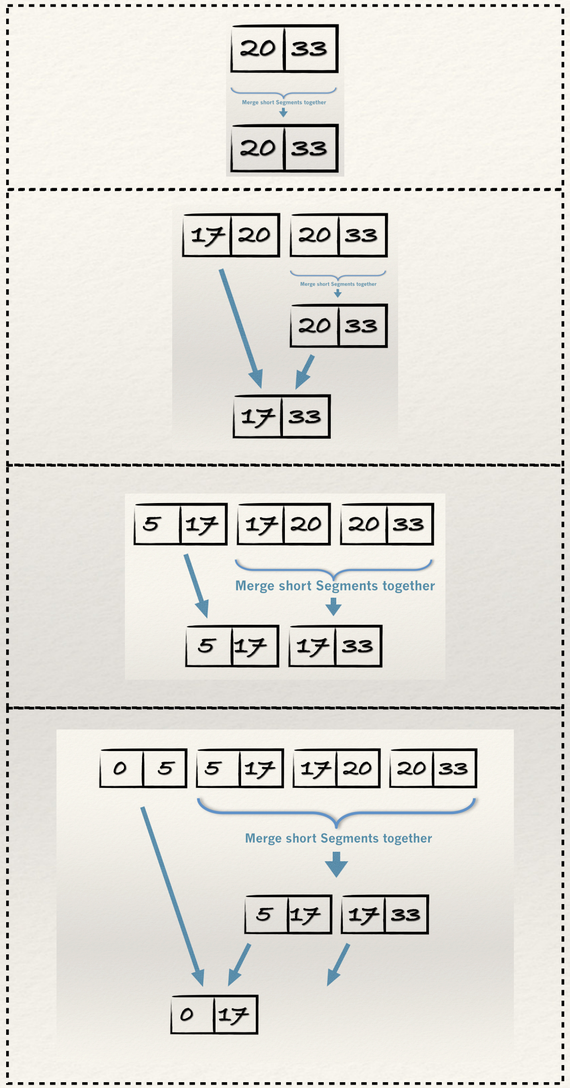
… with the controller as your interpreter

There is another place we are seeing this pattern...

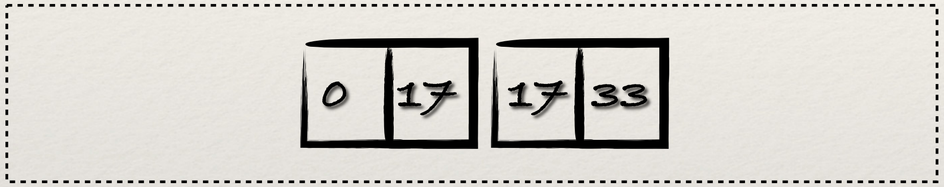
Events are <u>immutable</u> and <u>sequenced</u>!

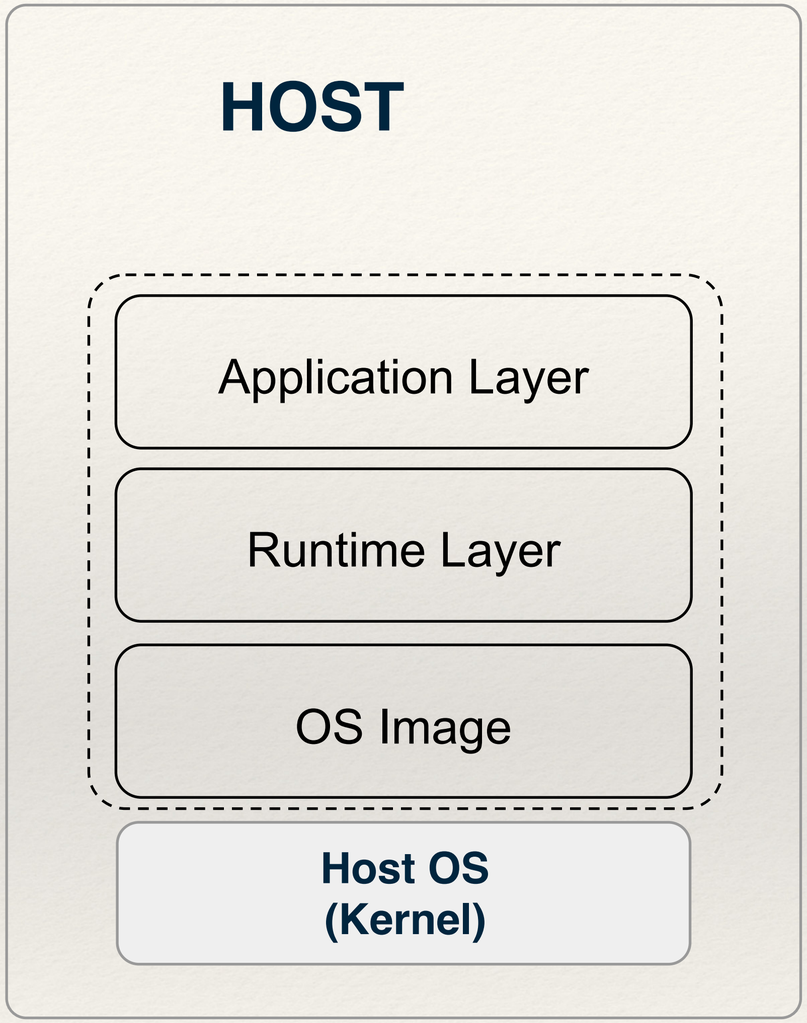
(Ephemeral) Containers

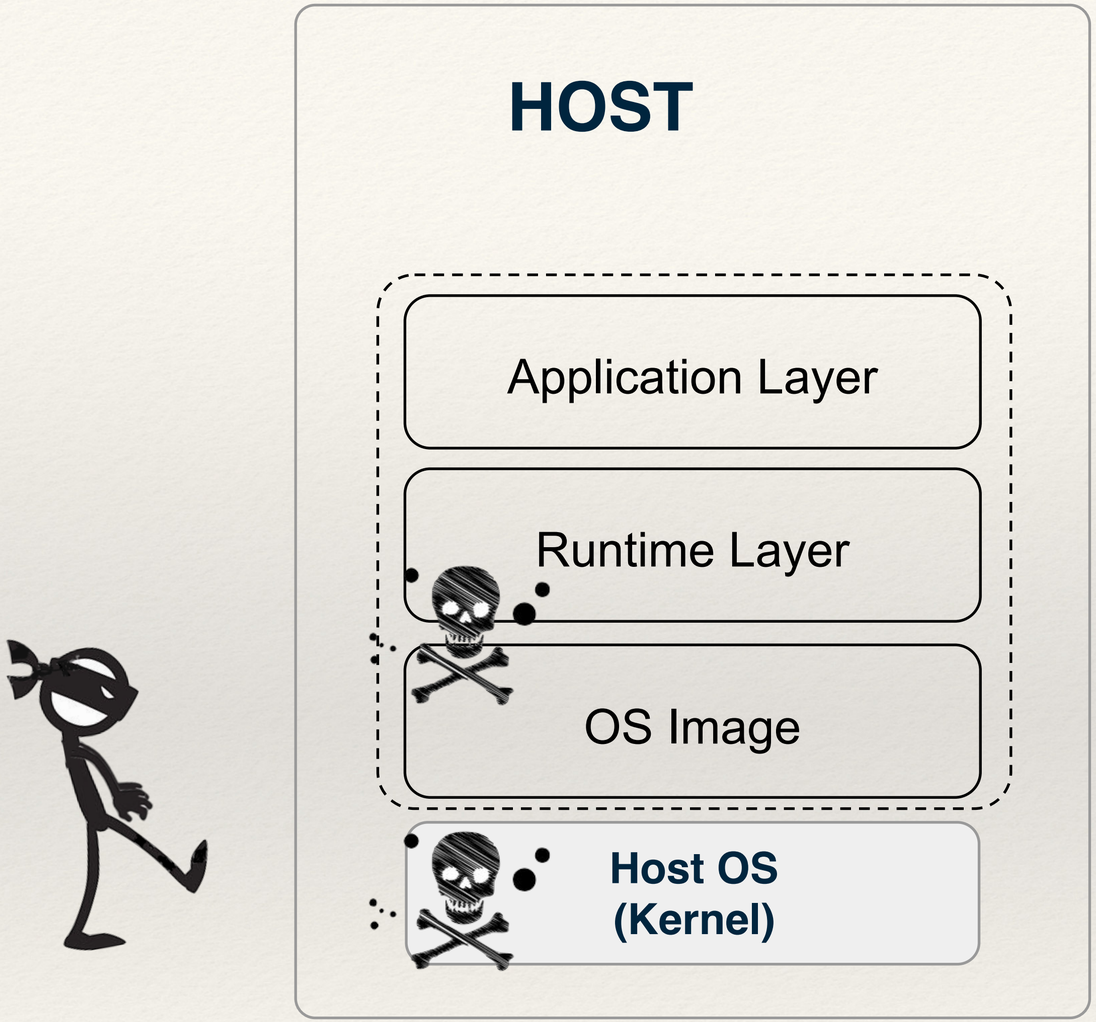
Many disposable stackframes:

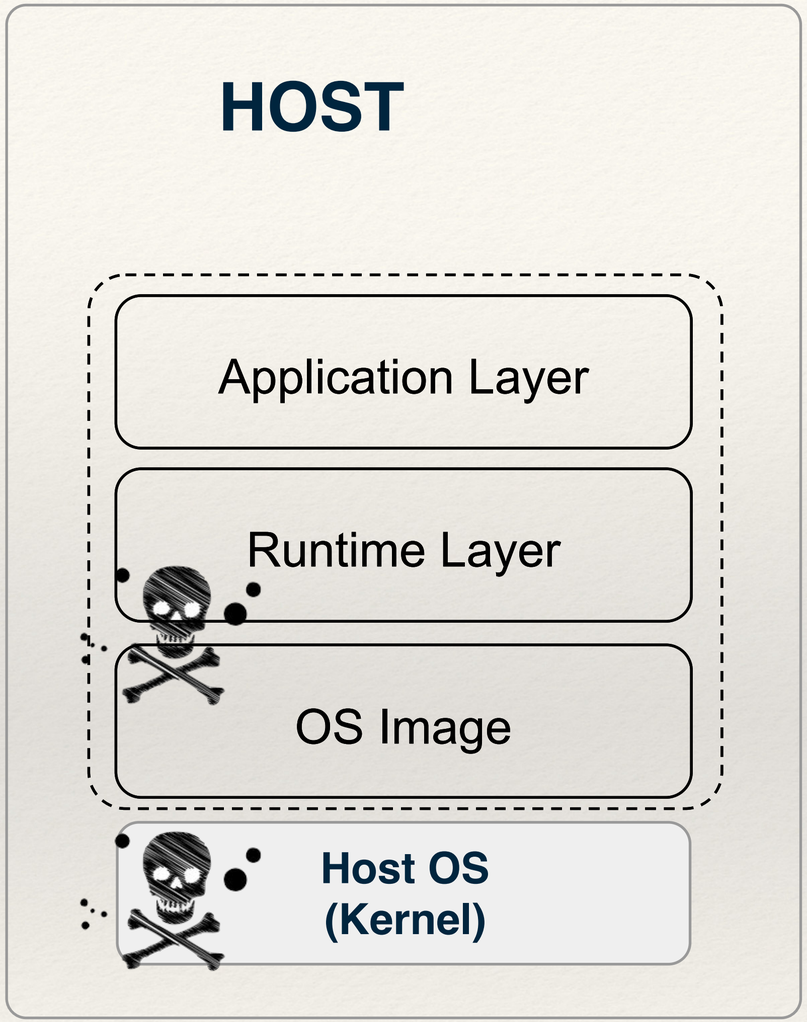


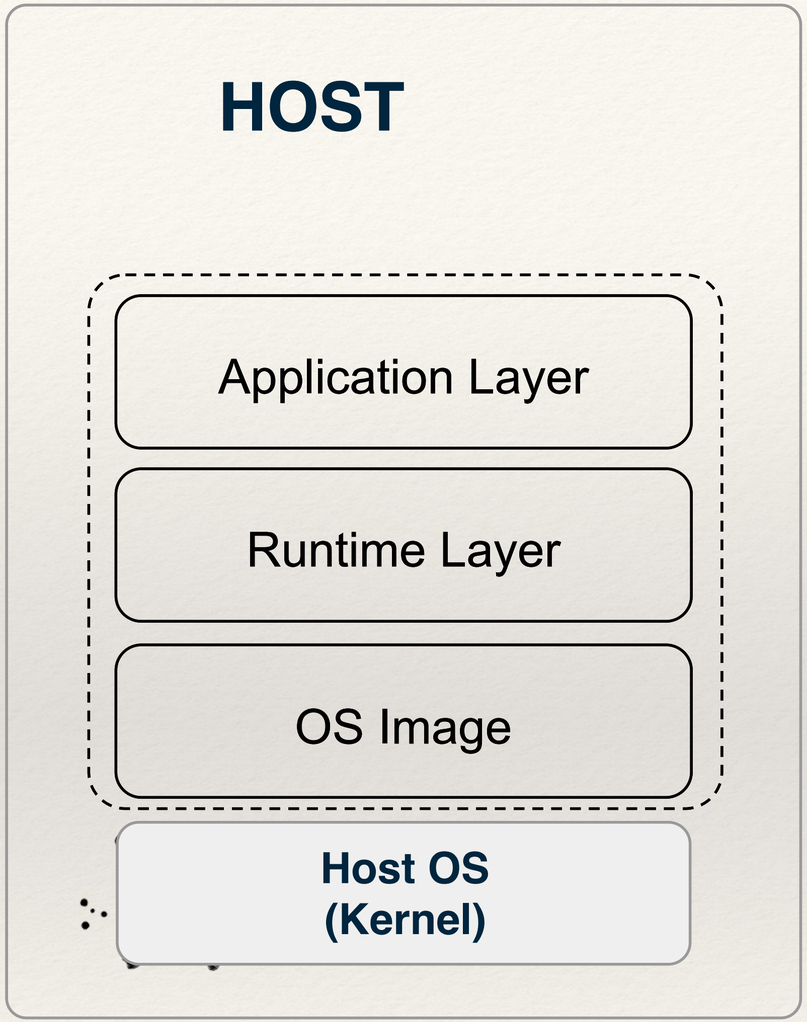
Long-running State:




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)

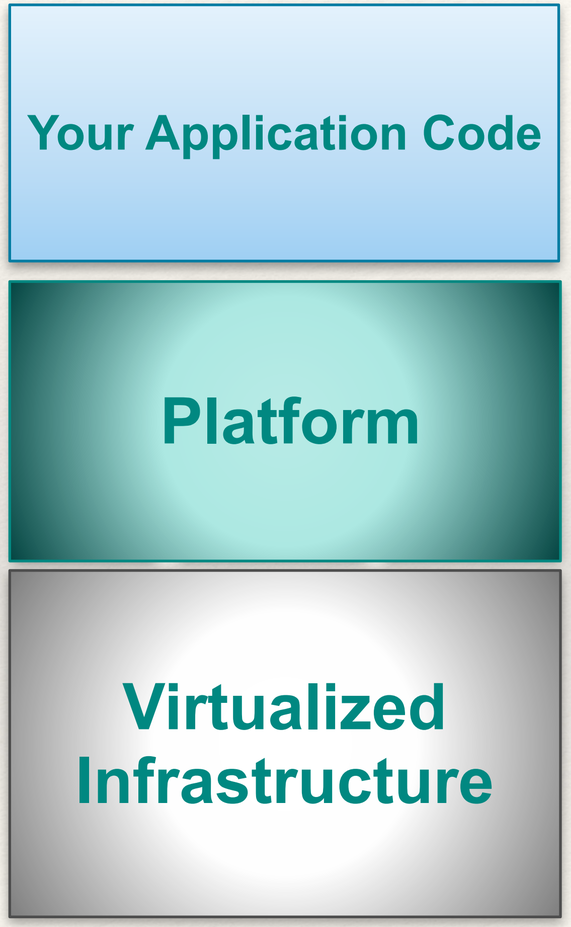


You can
Repave
the entire
environment

VERY Often!!!

Like several
<u>times per week</u>
“often”!

Sidecars

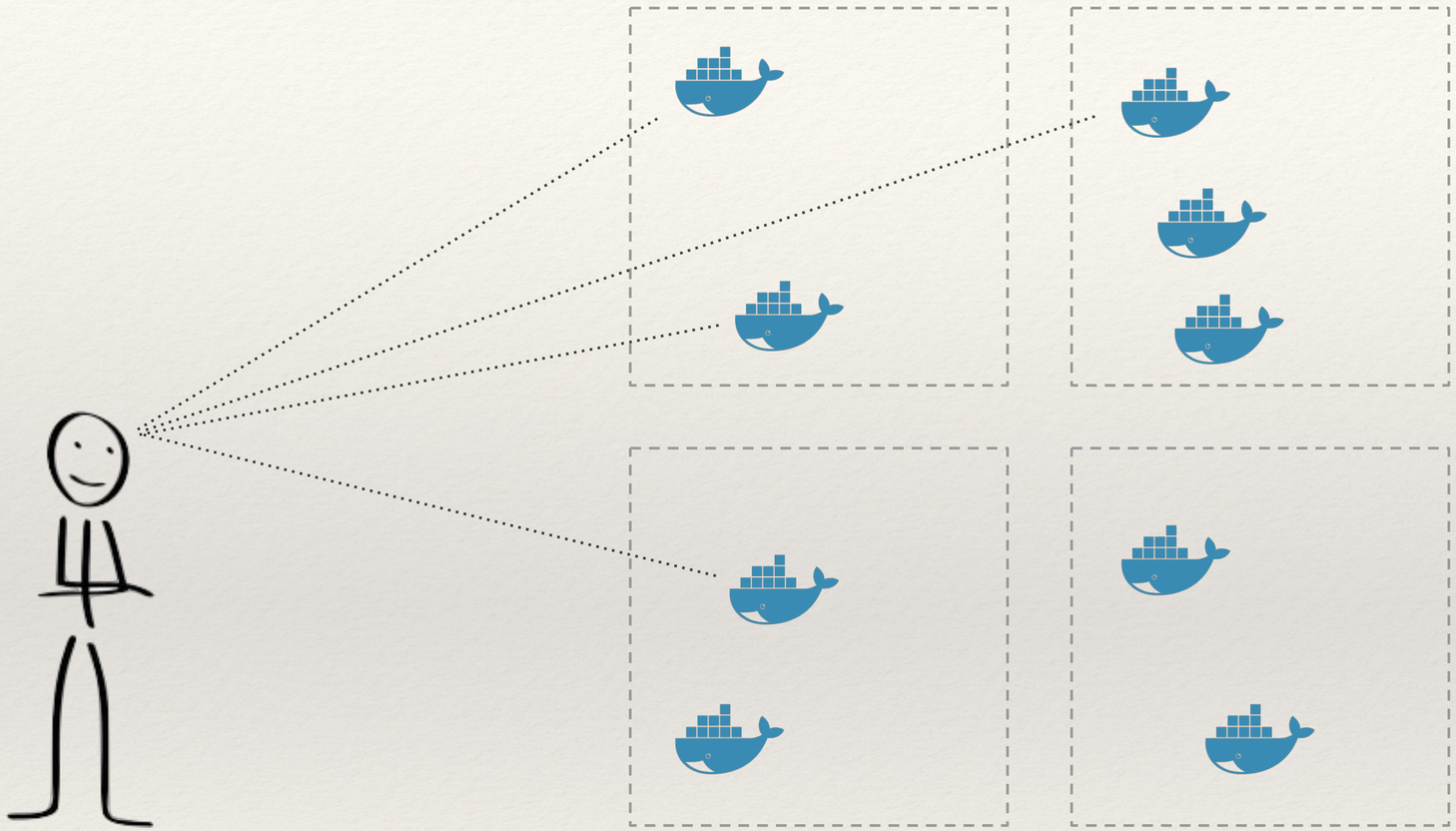


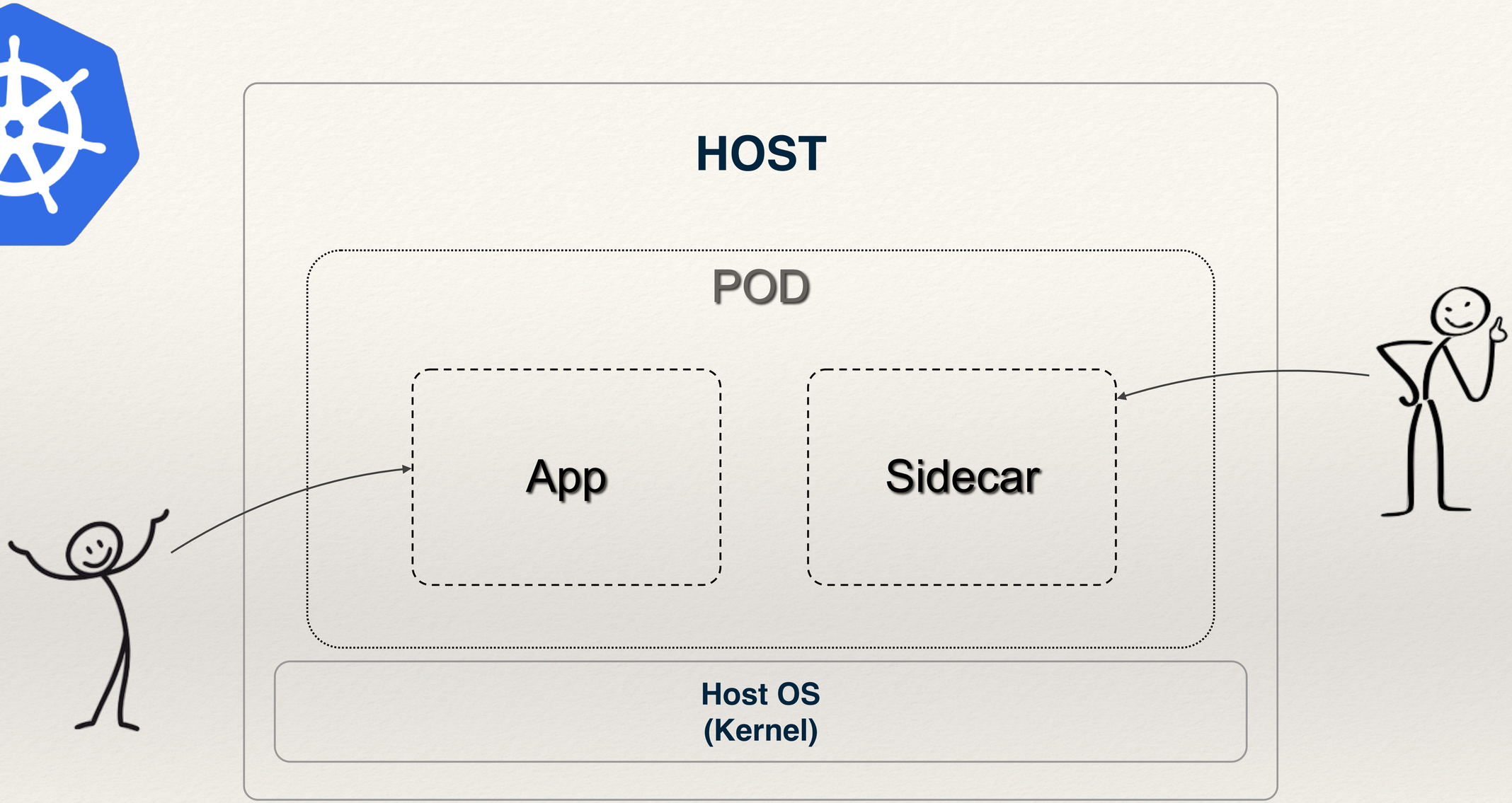
Application Team

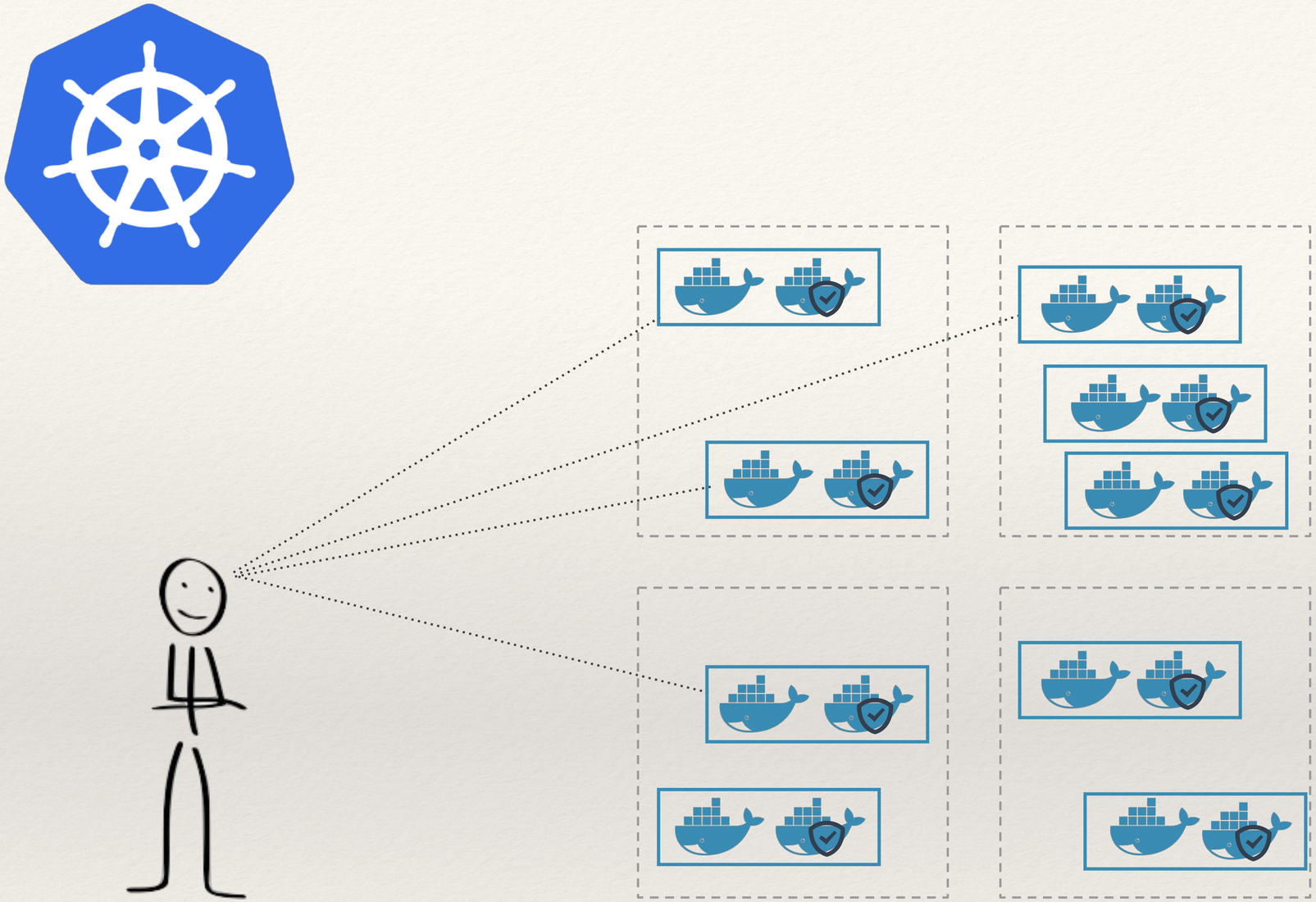
Platform Team

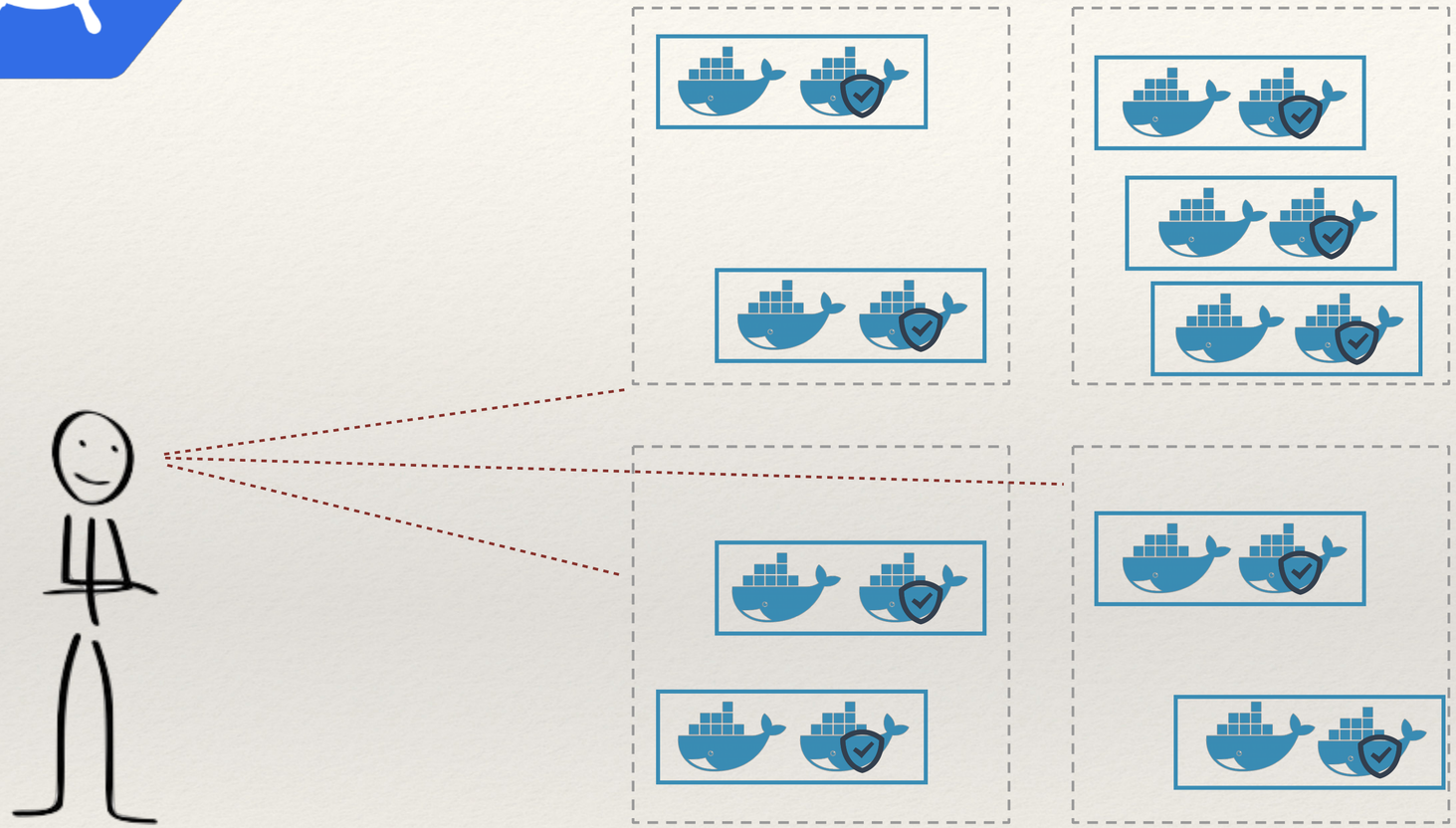
Iteratively building and delivering digital offerings to the consumer

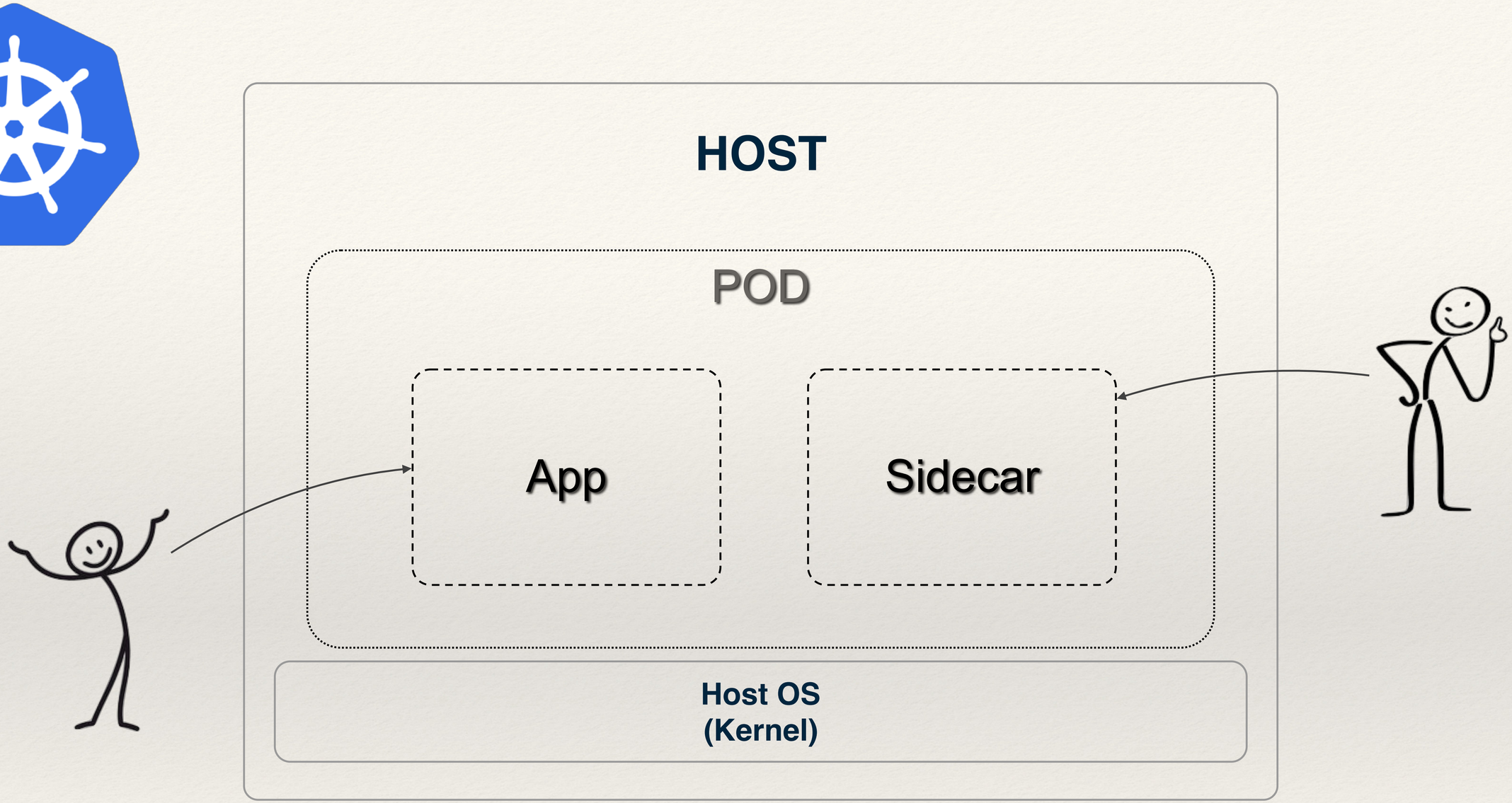
Enabling the app teams all while maintaining
Security
Compliance
Resilience
Cost Efficiency

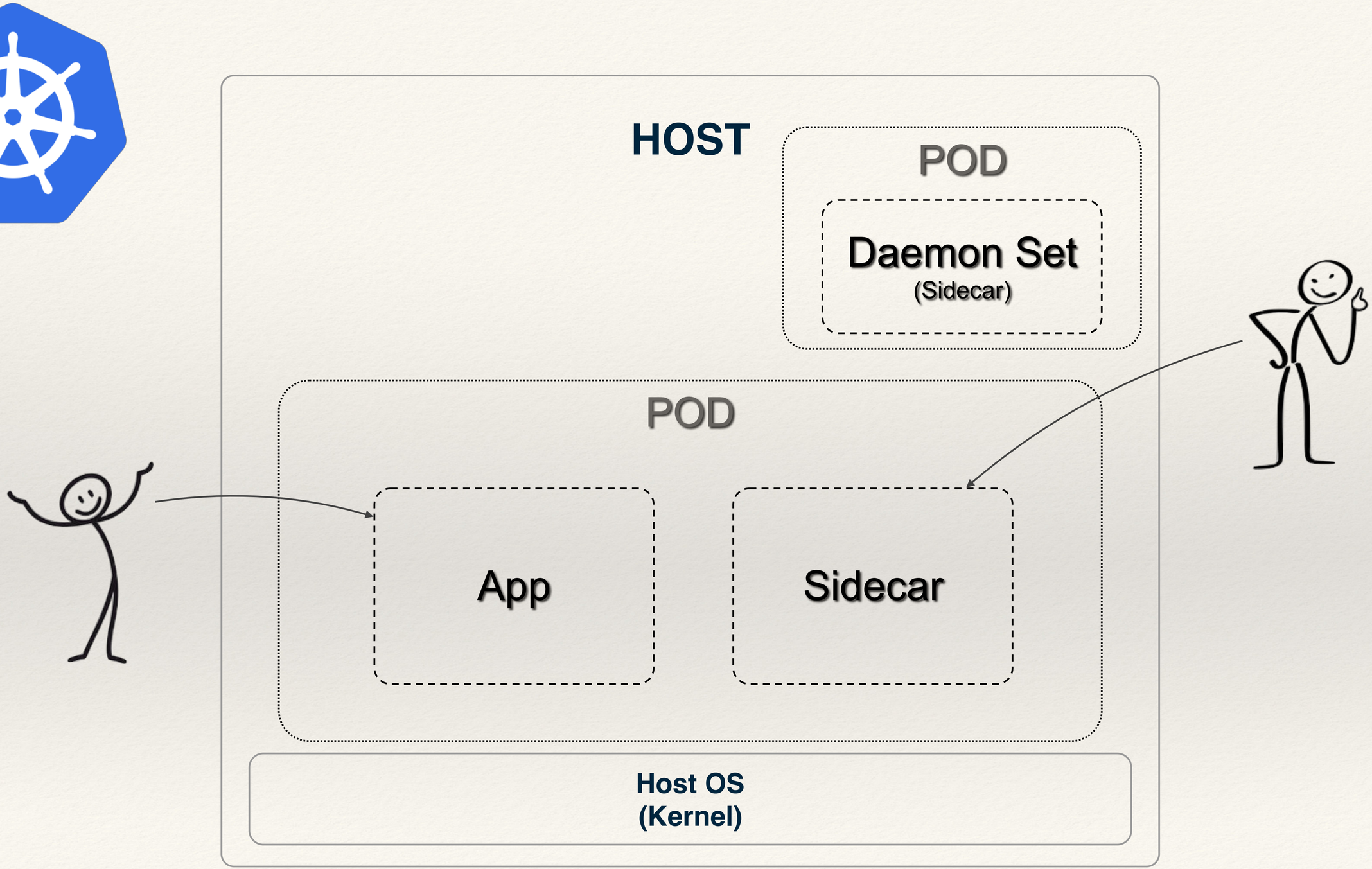


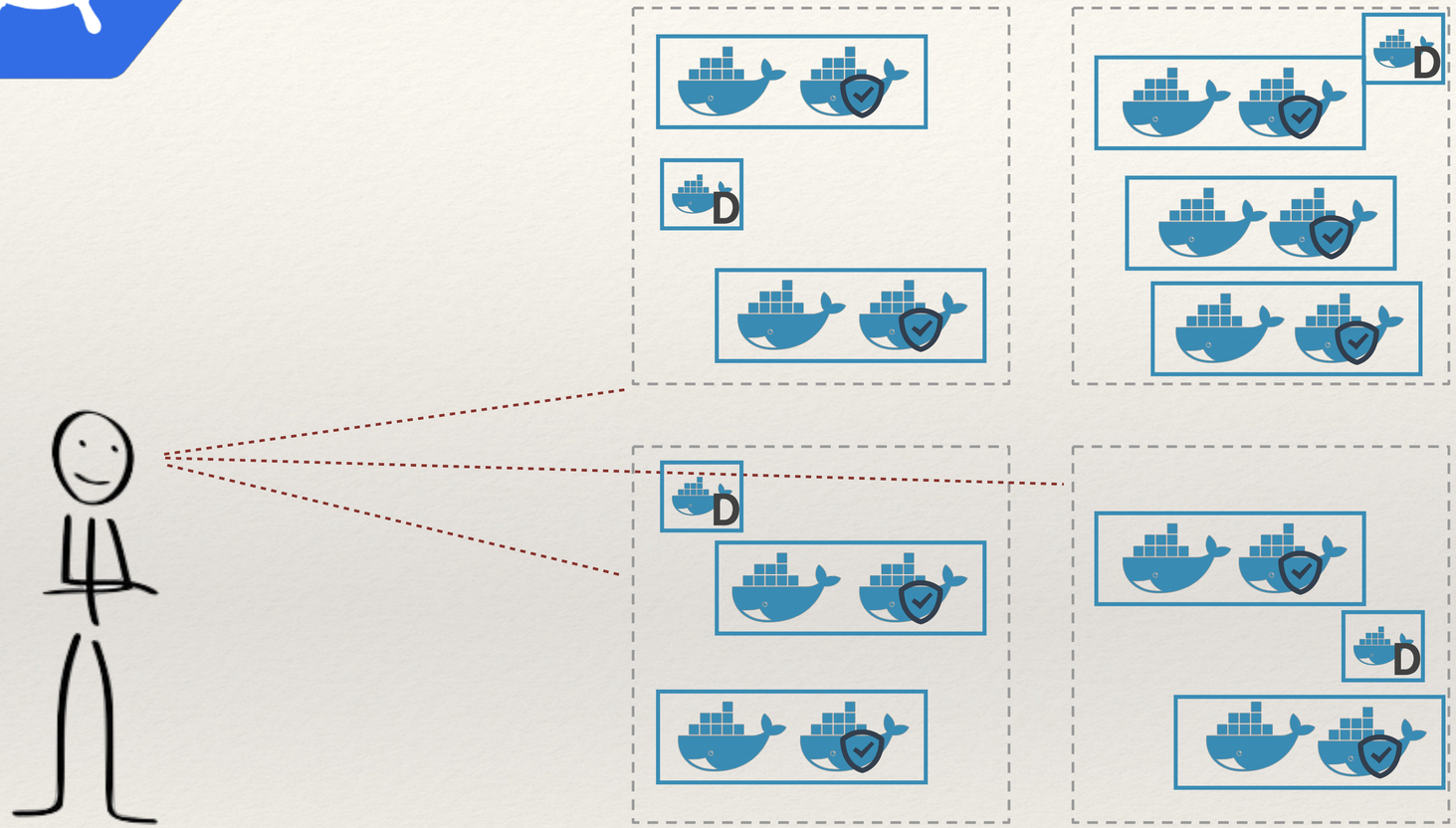

HOST
POD
App
Sidecar
Host OS
(Kernel)





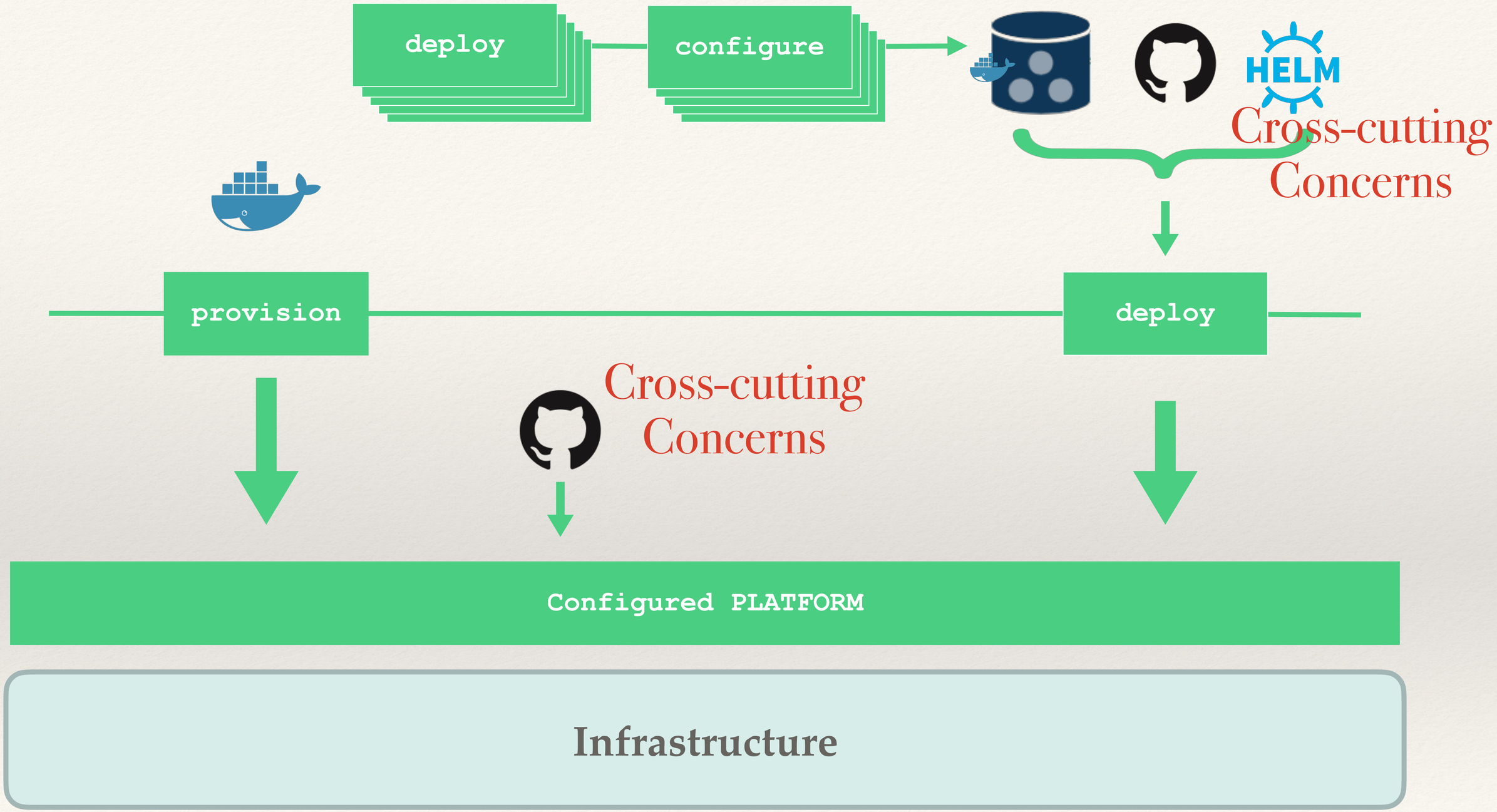

HOST
POD
App
Sidecar
Host OS
(Kernel)


HOST
POD
Daemon Set
(Sidecar)
POD
App
Sidecar
Host OS
(Kernel)


D
D
D
D

It’s about injecting cross-cutting concerns

(As it was for AOP)


deploy
configure
HELM
Cross-cutting Concerns
provision
deploy
Cross-cutting Concerns
Configured PLATFORM
Infrastructure

Systems Programming

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

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…

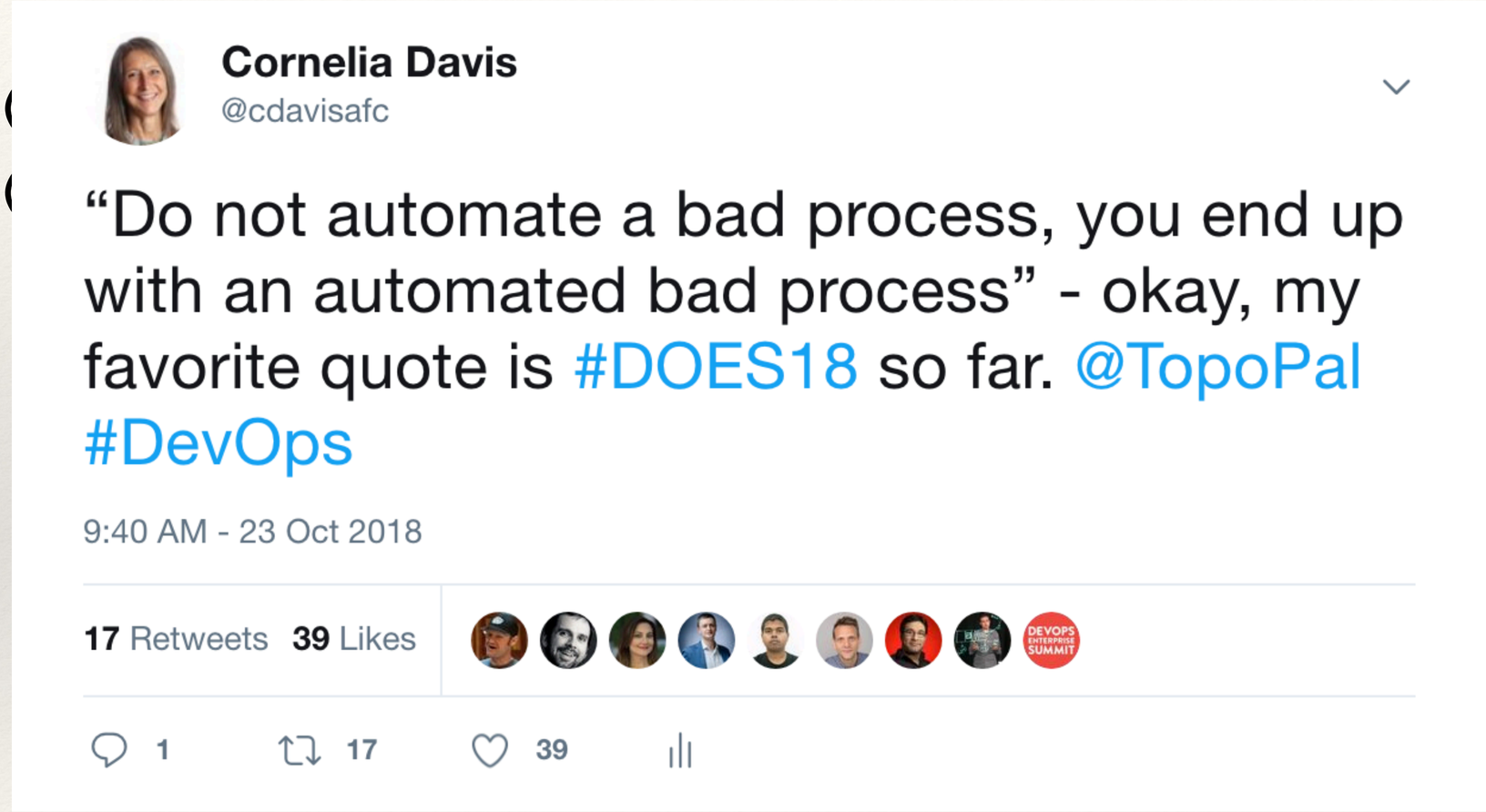
- Clojure
- Kotlin
- F#
- Scala

For the platform team…

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

Warning…

```
(define (dontMergeSegmentsThisWayPlease segments threshold)
  (do ((listLen (vector-length segments))
       (
       (                                                        )
                                                                ))
    segments)
```



What I Need Help With

- Stories, stories, stories!!
    - Is anyone doing this??
- How to achieve the “toe turn”

Thank you

DEVOPS ENTERPRISE SUMMIT

AN IT REVOLUTION EVENT

GET TOGETHER GO FASTER

October 22-24, 2018
The Cosmopolitan of Las Vegas
Las Vegas, Nevada

Electric Cloud IT REVOLUTION #DOES18