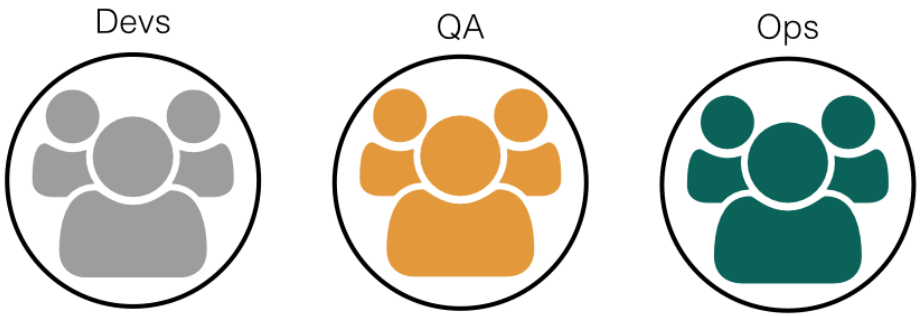
CNA Changes

CNA Changes

- Culture Change
- Organizational Change
- Technical Change

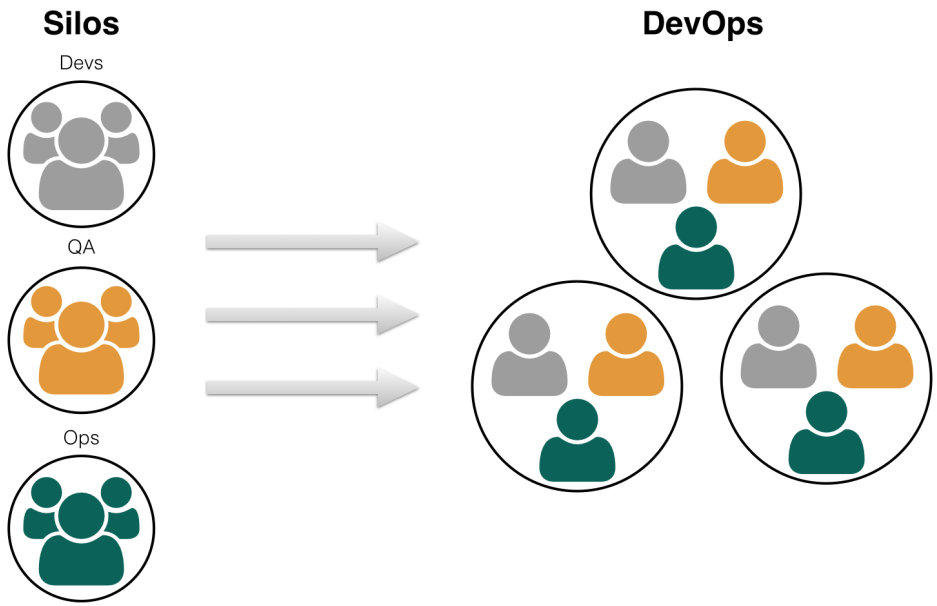
Traditional IT Teams

- Grouped by speciality
- Different vocabulary, tools, mgmt, incentive structures
- Different views on the role of IT
- Heavy weight processes to bridge the gap
- Opposed to moving fast



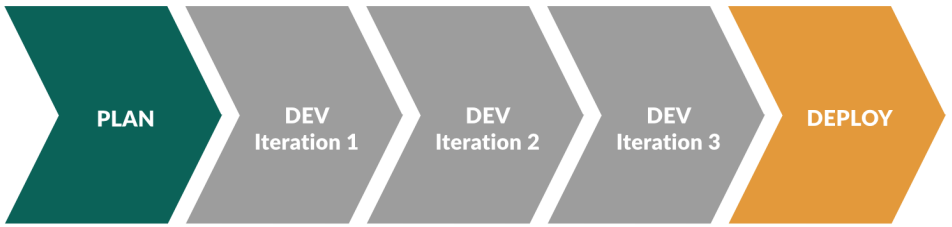
Silos to DevOps

- Goal: Deliver value rapidly and safely
- Shared vocabulary, tools, and incentive structures
- Bureaucratic processes replaced with trust and accountability
- Common leadership



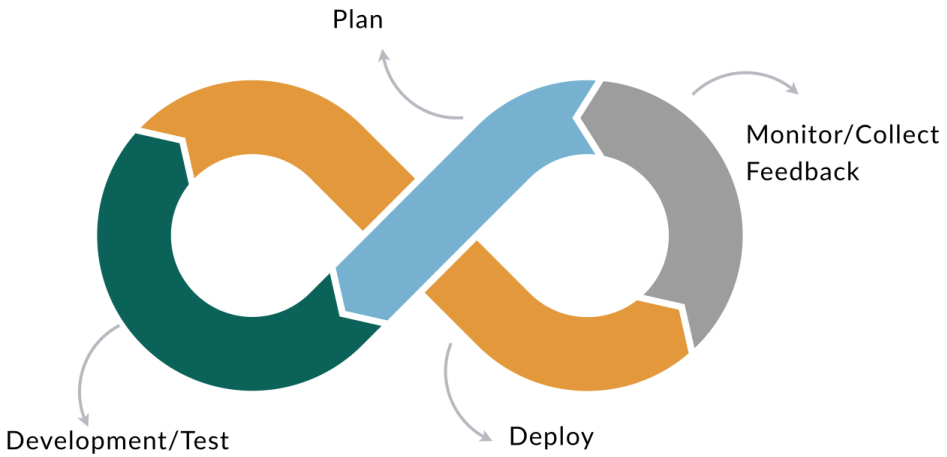
Waterscrumfall

- Agile practices adopted but only in development
- Waterscrumfall
  - Customers (Lines of Business) revert back waterfall tendencies
  - No feedback in to process



Continuous Delivery

- Embrace agile principles end to end
- Build out Continuous Delivery practice
- Every iteration is proven to be deployable in an automated fashion
- Leverage deployment pipelines to ensure testing and deployment validation and consistency



CNA Changes

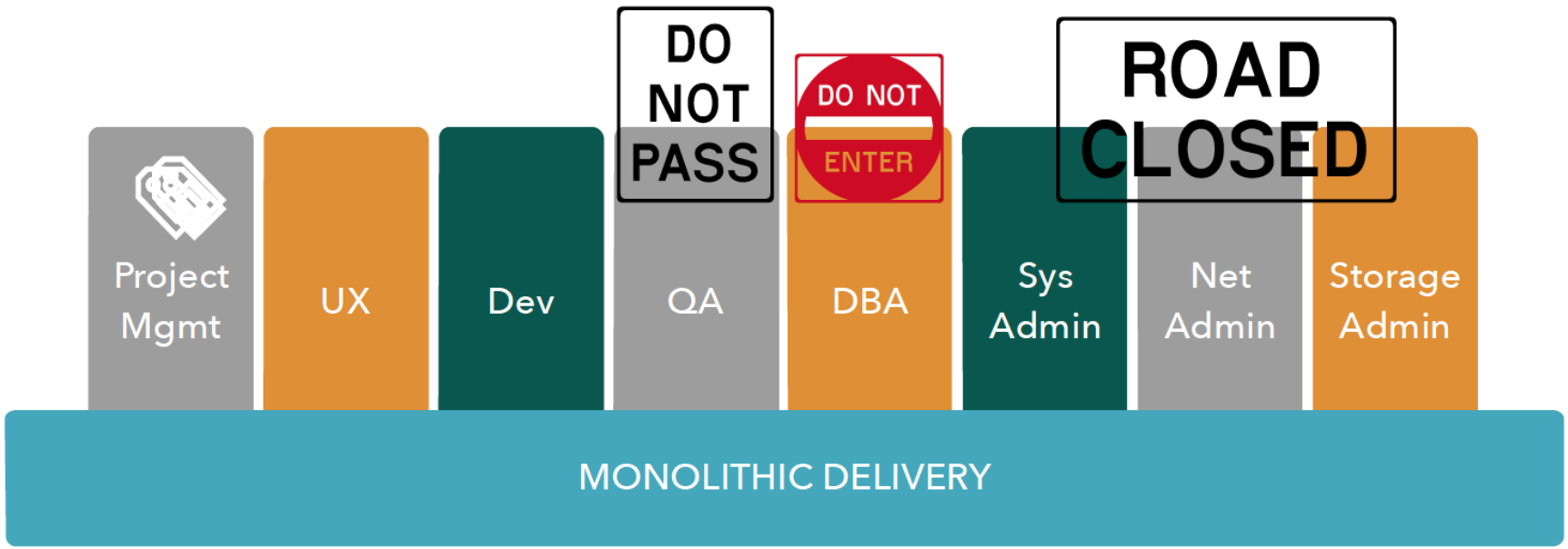
- Culture Change
- Organizational Change
- Technical Change

Conway's Law

Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.

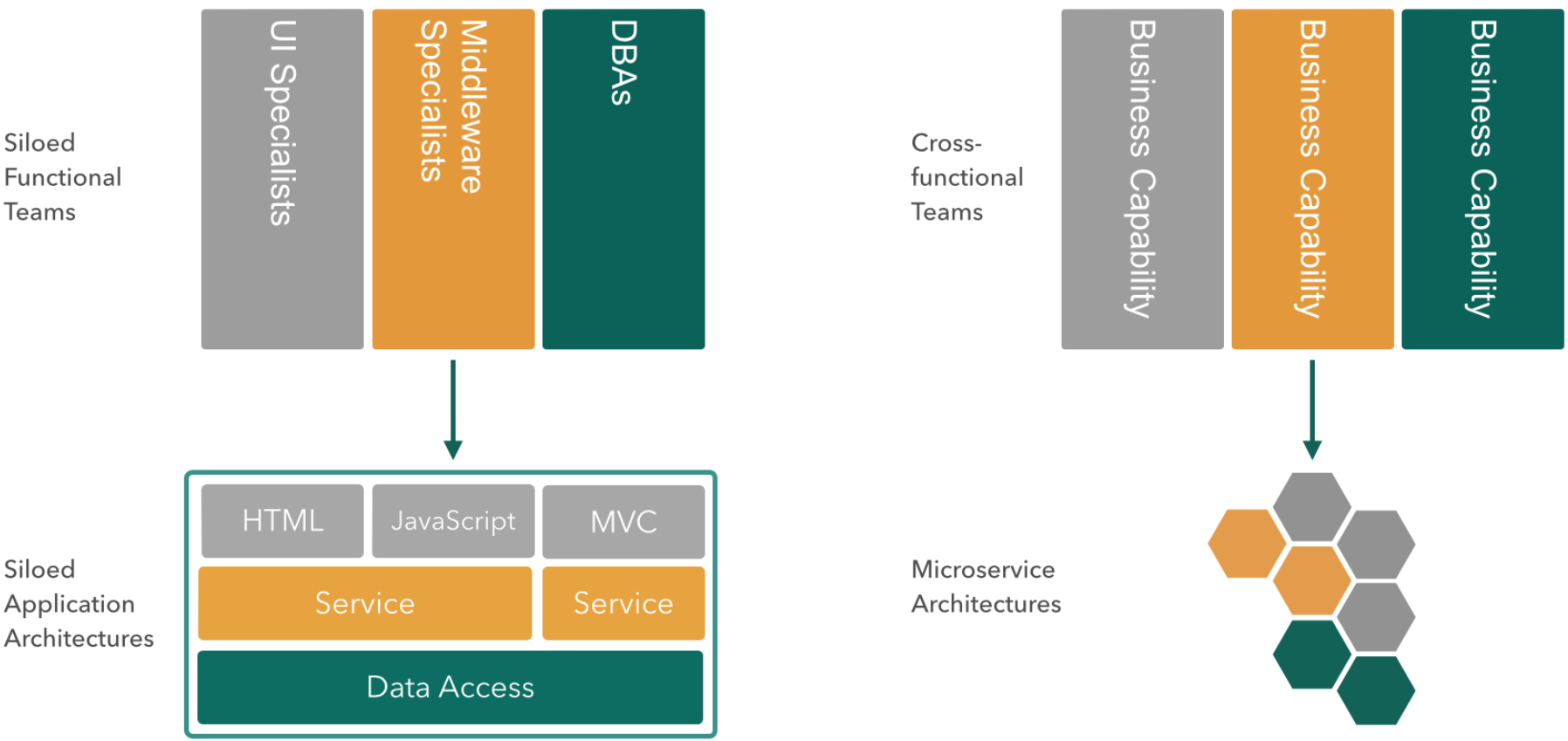
Melvyn Conway

Monolithic Approach To Delivery



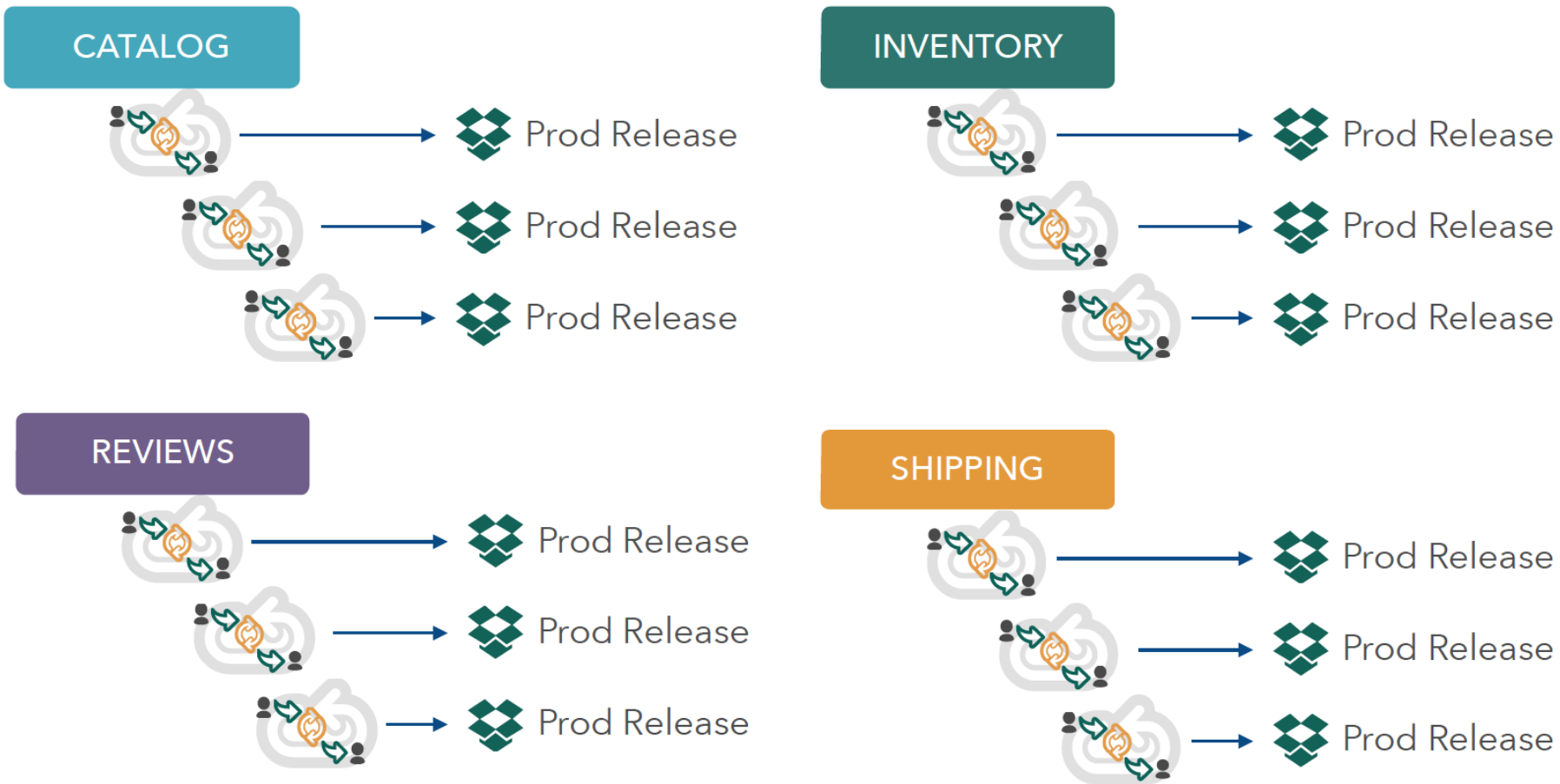
Adapted from:http://www.slideshare.net/adriancockcroft/goto-berlin

Business Capability Teams



http://martinfowler.com/articles/microservices.html#OrganizedAroundBusinessCapabilities

Business Capability Teams

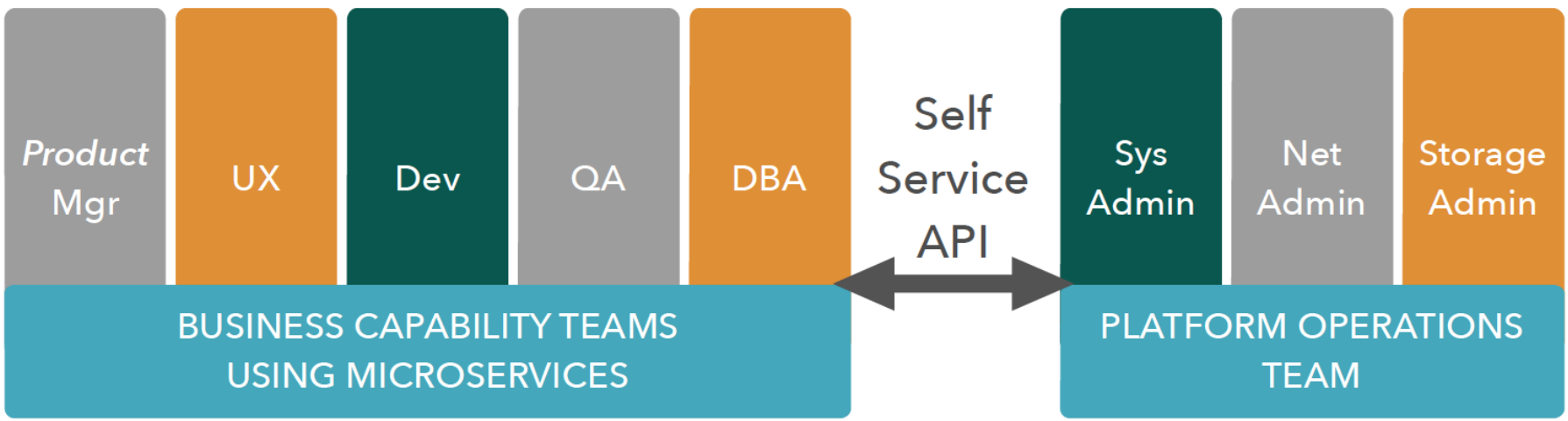


Platform Operations Teams

Adapted from:http://www.slideshare.net/adriancockcroft/goto-berlin

Self Service API

Decoupling - it actually works this time!



Adapted from:http://www.slideshare.net/adriancockcroft/goto-berlin

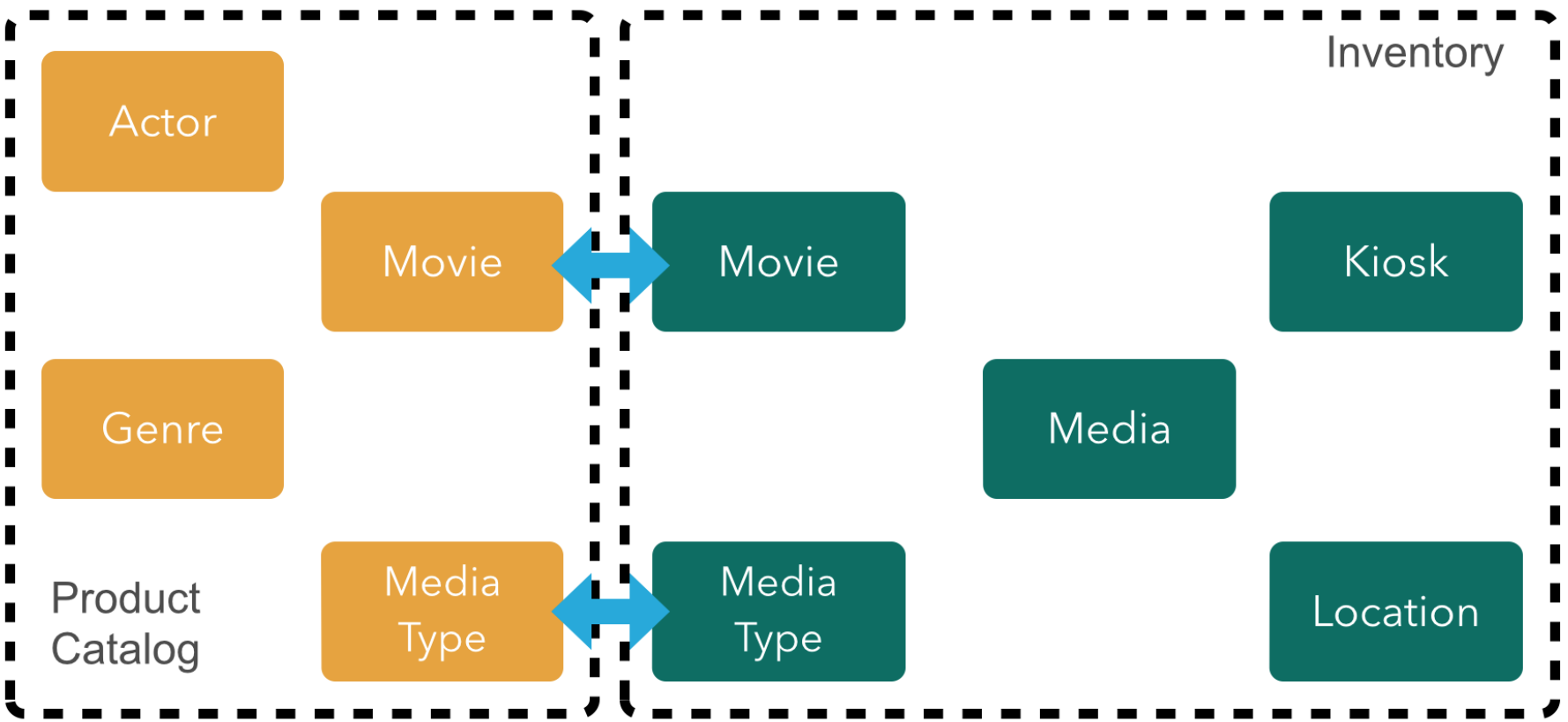
CNA Changes

- Culture Change
- Organizational Change
- Technical Change

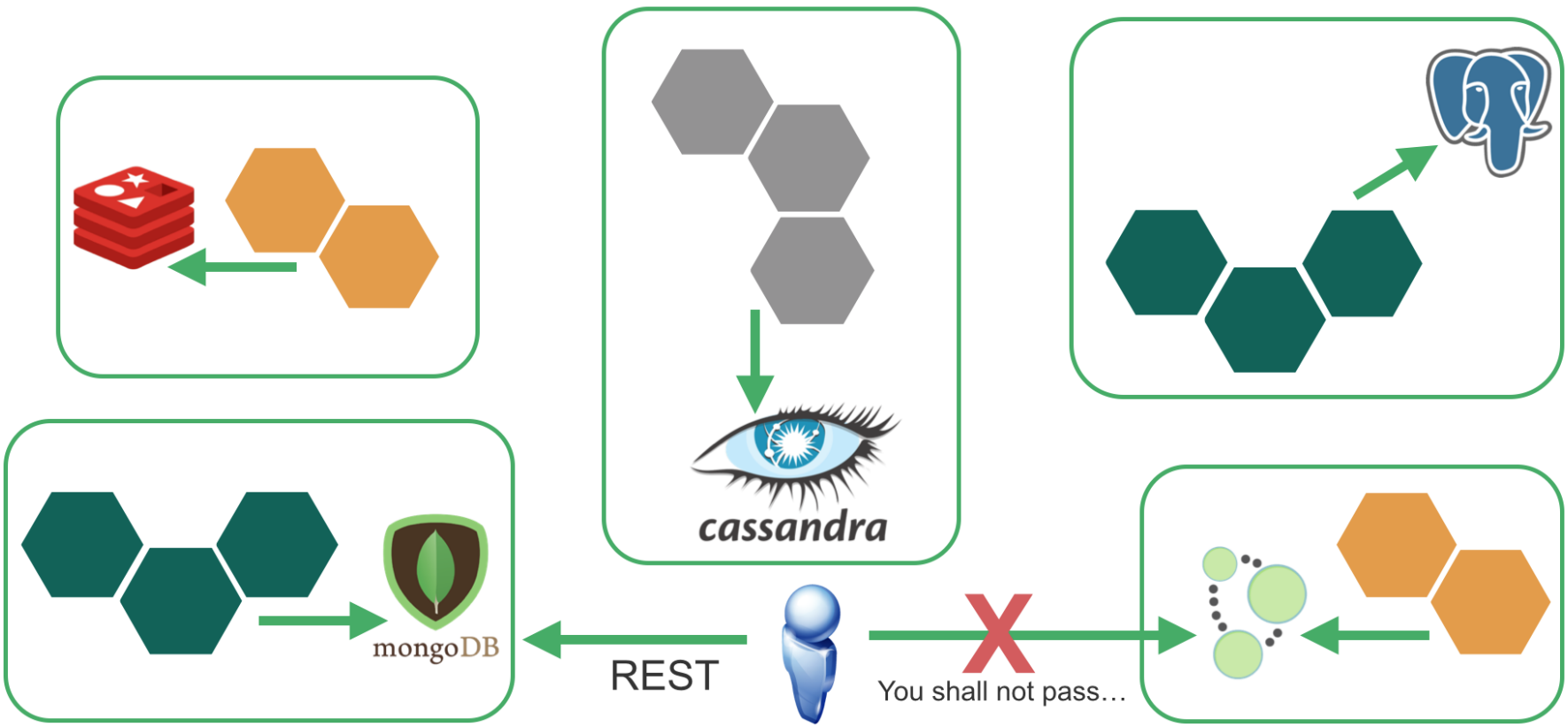
Why Decompose Monoliths into Microservices

- Monoliths design patterns not appropriate for the cloud
- Monoliths couple change cycles together
- Monolith services cannot be scaled independently
- Too many developers in one code base
- Developers struggle to understand the codebase
- Long term commitment to tech stack

Start By Decomposing Data - Bounded Contexts

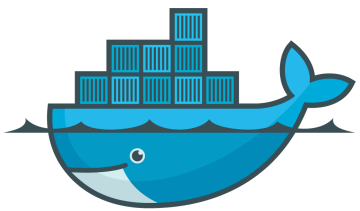


Polyglot Persistence



Containers - The New Unit of Deployment

LXC



docker

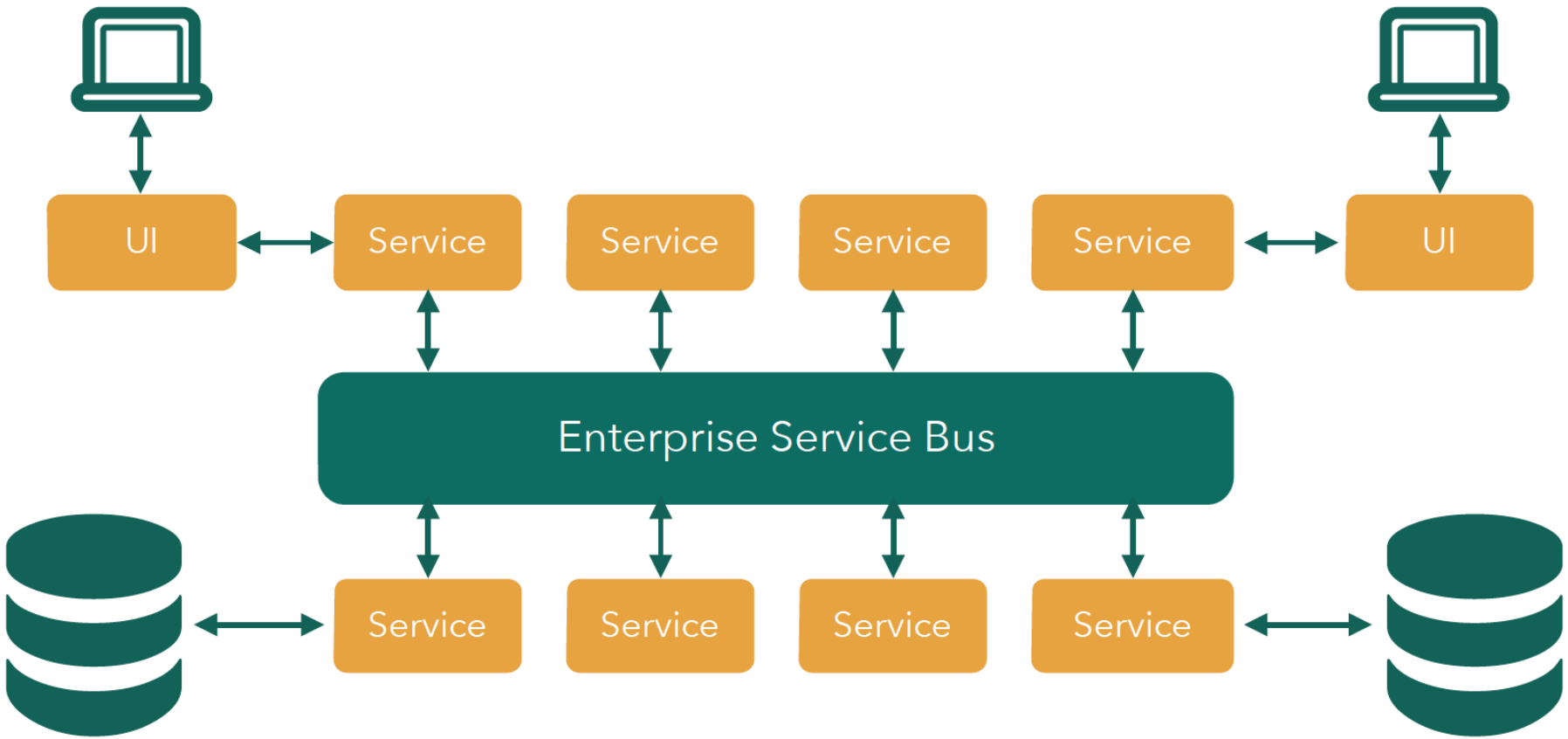
rkt

Integration - From Orchestration to Choreography

Microservices - Unix Pipes and Filters

```
cut -d" " -f1 < access.log | sort | uniq -c | sort -rn | less
```

Traditional ESB



Microservices