Microservices

K V SUBRAMANIAM

Why Microservices?

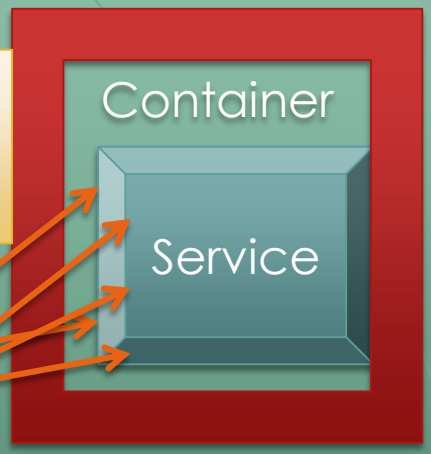
Monolithic Web applications

- Difficult to modify as code becomes larger

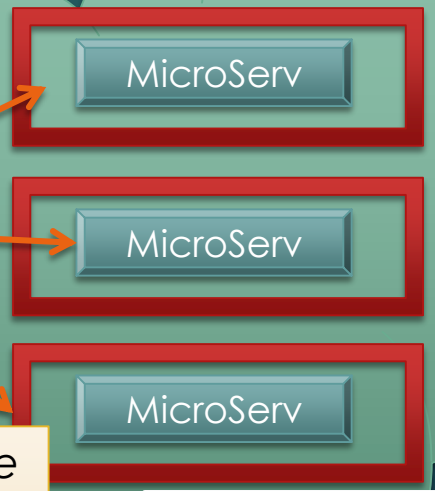
Microservices

- Small and easy to code and deploy
- Used by companies such as Netflix, eBay, Amazon, Twitter, PayPal, to add features quickly

All functionality in a single monolithic app



App typically layered – web tier, DB tier, and so on



- Each *microservice* supports only one functionality
- Storage for shared state

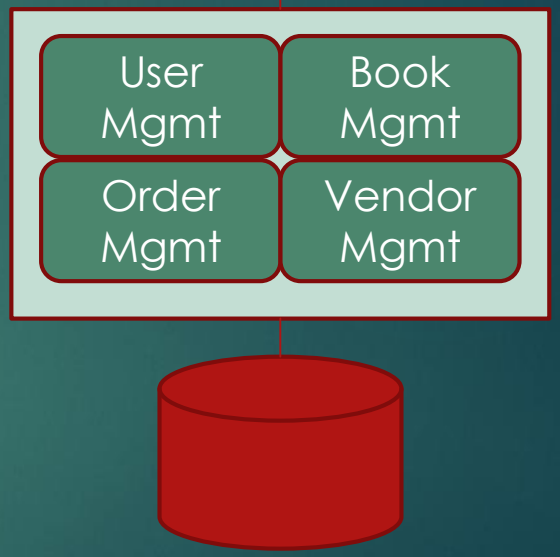
Each *microservice* will have multiple tiers

Exercise 2 (10 minutes)

- Consider the Bookkart application with the following functions
  - User Management : Profile and Login
  - Book Inventory: List books in a category
  - Order Placement: placing orders
  - External Vendor management: System orders book if not available in warehouse
- What will a monolithic software architecture for a such an application look like

Solution

- All components are compiled and packaged together
  - Even if there is a defect in one, you have to compile/install the entire software
  - Increases the software development lifecycle time.
  - Schemas are linked with each other.
  - Components will interact through mostly direct function calls.



UserTable InventoryTable

OrdersTable VendorTable

What is the microservice model?

- Package each sub-application such that it has
  - It's own GUI screens
  - Its business logic
  - Its own database
  - Definition of a API for all endpoints within the service.
    - All communication happens through API REST/RPC

Microservice architecture

API Gatewa y

RPC/REST API

RPC/REST API

RPC/REST API

uServic e1

uServic e1

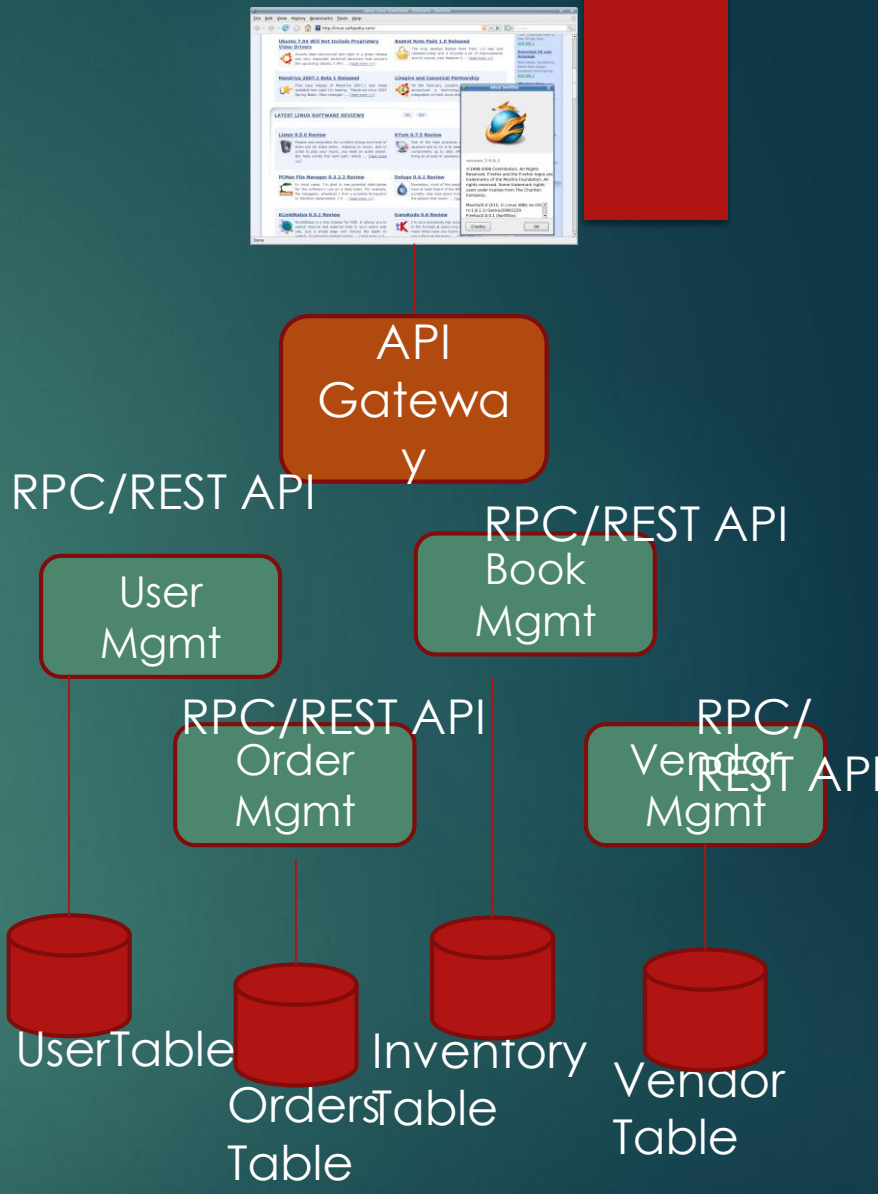
uServic e1

Exercise 3 (10 minutes)

- Consider the Bookkart application with the following functions
  - User Management : Profile and Login
  - Book Inventory: List books in a category
  - Order Placement: placing orders
  - External Vendor management: System orders book if not available in warehouse
- Design a microservices architecture for this application

Solution

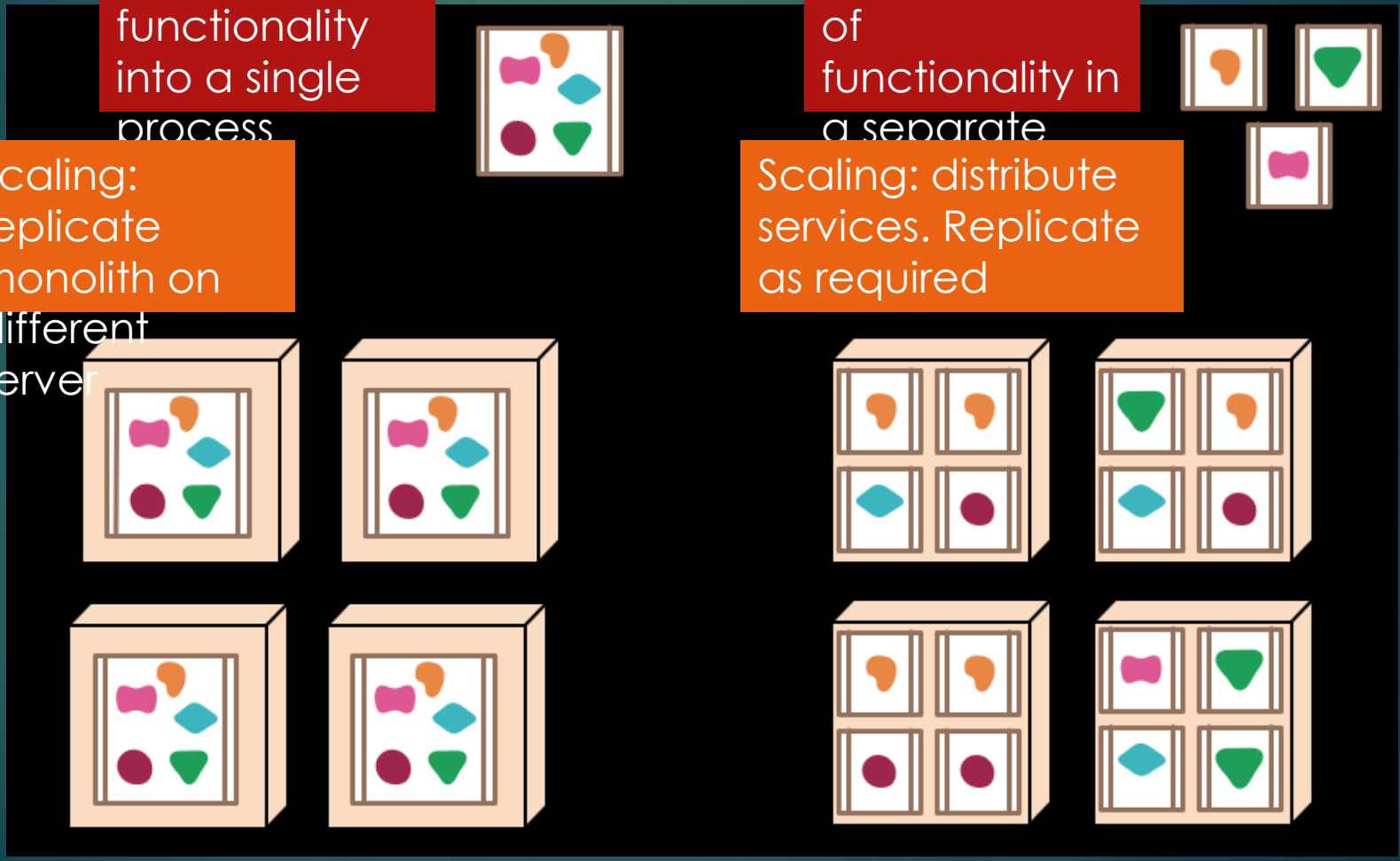
- Each component is separately compiled and deployed onto a container
- Will communicate with other microservices using REST APIs



Monolithic v/s microservices…

All functionality into a single process

Scaling: replicate monolith on different server



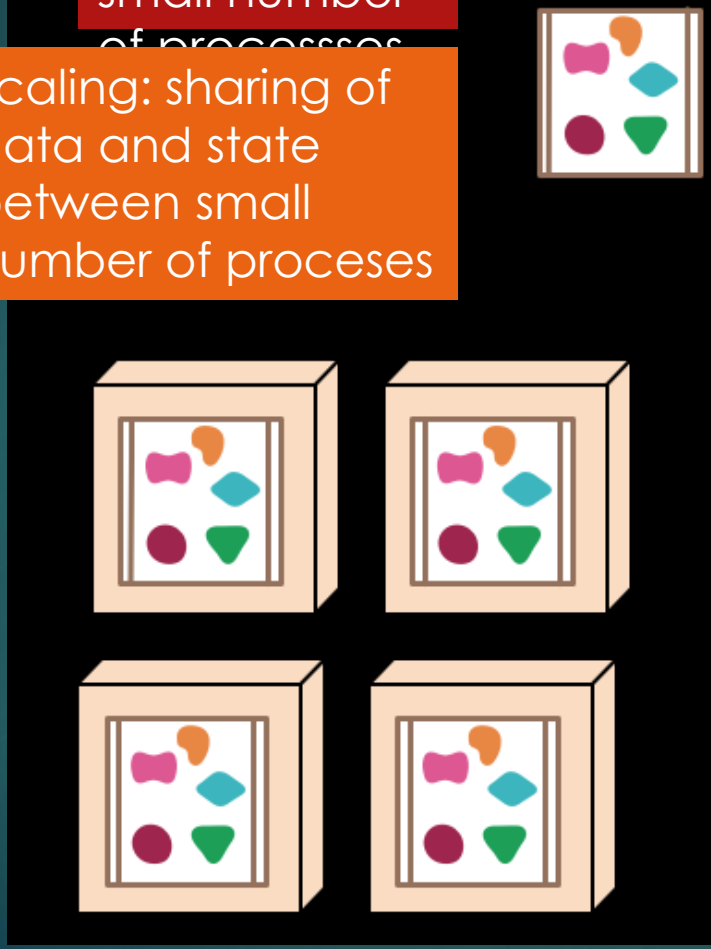
Each element of functionality in a separate

Scaling: distribute services. Replicate as required

…Monolithic v/s microservices

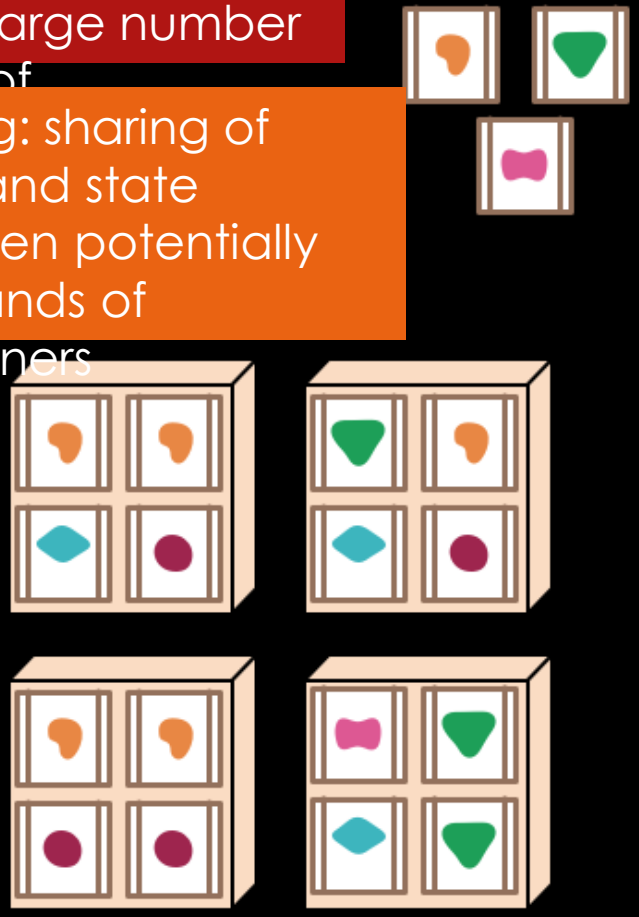
All data accessed by small number of processes

Scaling: sharing of data and state between small number of proceses



Data accessed in parallel by large number of

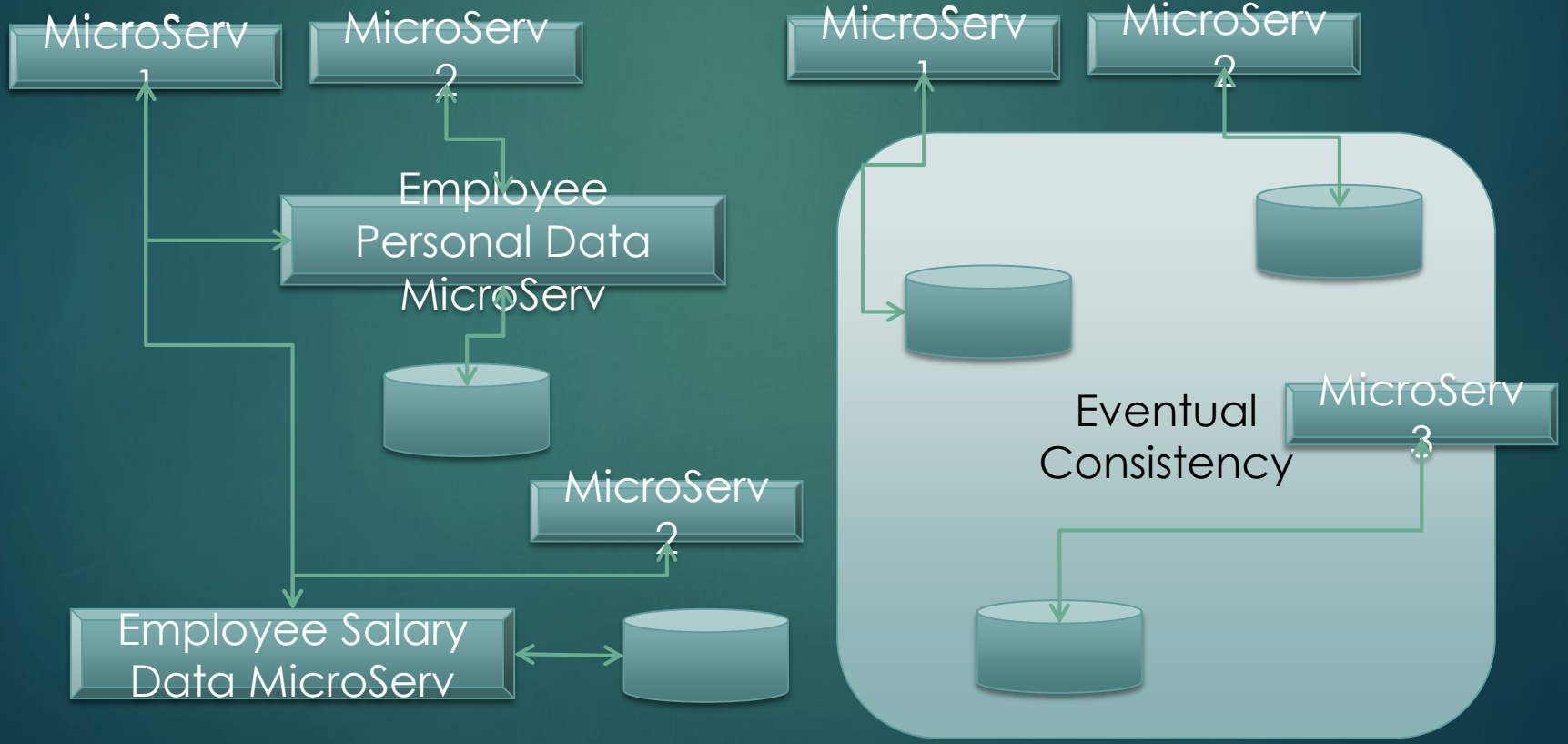
Scaling: sharing of data and state between potentially thousands of containers



Options for Sharing Data and State

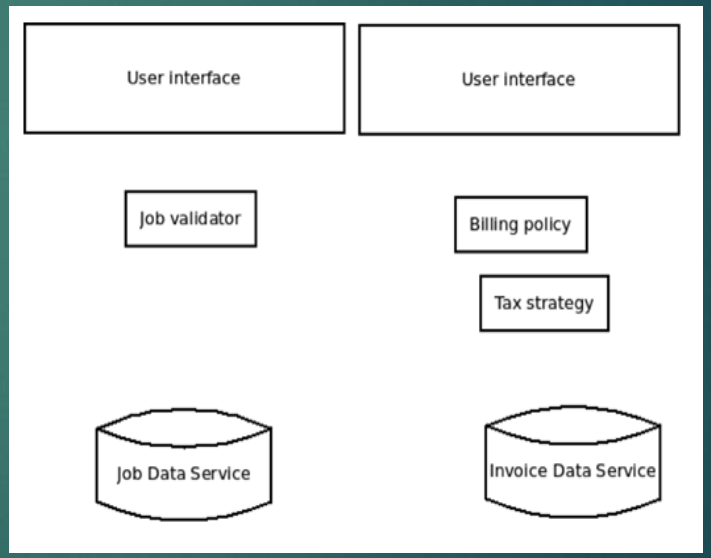
Data Service: simpler

Per Microservice DB: faster



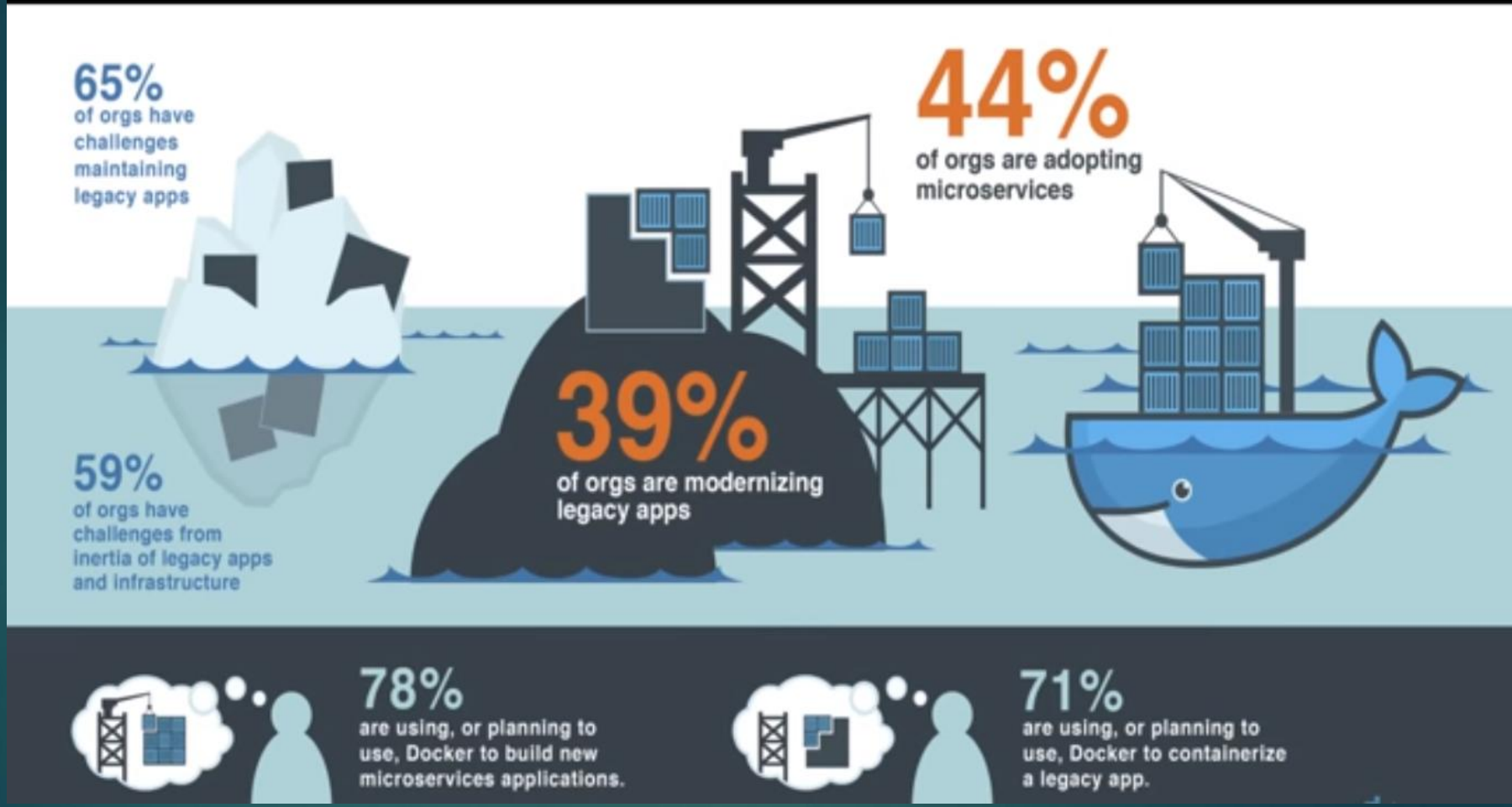
Microservices Example

- Uses data service
- Commands
  - Generate invoice
  - Complete job, generate invoice
- User creates a manual invoice
  - Adds data to invoice, status *created*
  - Invokes BillingPolicyService to determine when payable
  - Invoice is issued to customer
  - Persists to the invoice data service, status *sent*
- User finishes a job, creating an invoice
  - Validates job is complete
  - Adds data to invoice, status *created*
  - Invokes BillingPolicyService to determine wheninvoice is payable
  - Invoice is issued to customer
  - Persists to the invoice data service, status *sent*



https://smartbear.com/learn/api-design/what-are-microservices/
https://martinfowler.com/articles/microservices.html

Docker and Microservice adoption



Realizing the microservice model

- Your large application is now broken into
  - Small pieces with well defined APIs
  - Each will run in its own container(s)
- Question: starting one process on a single machine is quite challenging
  - Remember your assignment
- Now we want to start a host of microservices each in it's own container.