Microservices

Microservice Trade-Offs

Strong Modular Boundaries (Pro):

- Within monolithic application, it is possible to build an architecture with strong modular boundaries (i.e., modules that are highly cohesive and loosely coupled).
- Within monolithic applications, it is possible to sneak around such principles, whereas with microservice architectural styles it is not possible (why?)

Microservice Trade-Offs

Distribution (Con)

- Microservices use a distributed system to improve modularity.
- Any limitations of such distributed nature?

1. Performance (how?)
    - Mitigation?
        - Coarse-grained calls
        - Asynchrony
            - Debugging?
2. Reliability (how?)

Microservice Trade-Offs

Eventual Consistency (Con)

- With a monolith, you can update a bunch of things together in a single transaction. Microservices require multiple resources to update. (decentralized data management)
- Example: *You make an update to something, it refreshes your screen and the update is missing. You wait a minute or two, hit refresh, and there it is.*
  - How did this happen?
    - Your update was received by the pink node, but your get request was handled by the green node.
    - Until the green node gets its update from pink, you're stuck in an inconsistency window.

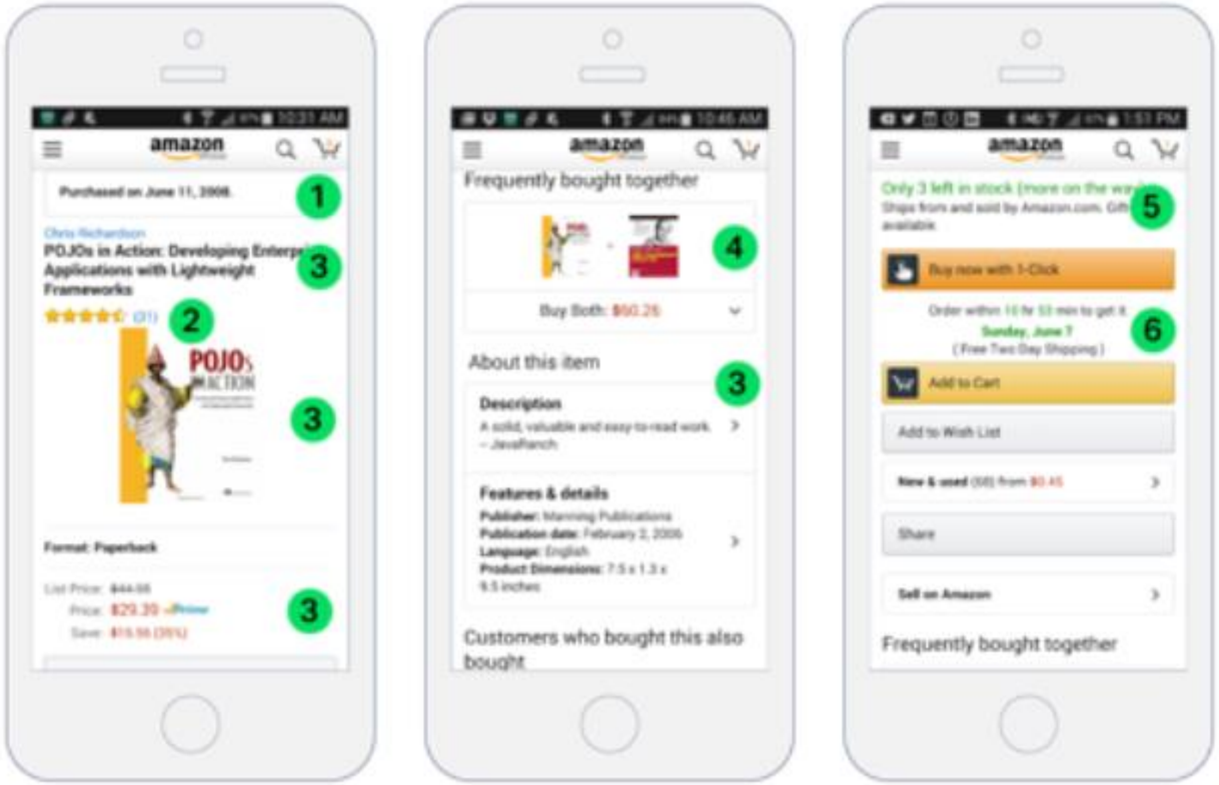
Microservice Trade-Offs

- Independent Deployment (Pro)
- Technology Diversity (Pro)

Building Microservices: Using an API Gateway

- Imagine that you are developing a native mobile client for a shopping application. It’s likely that you need to implement a product details page, which displays information about any given product.
- Let us check Amazon’s android application …

Building Microservices: Using an API Gateway

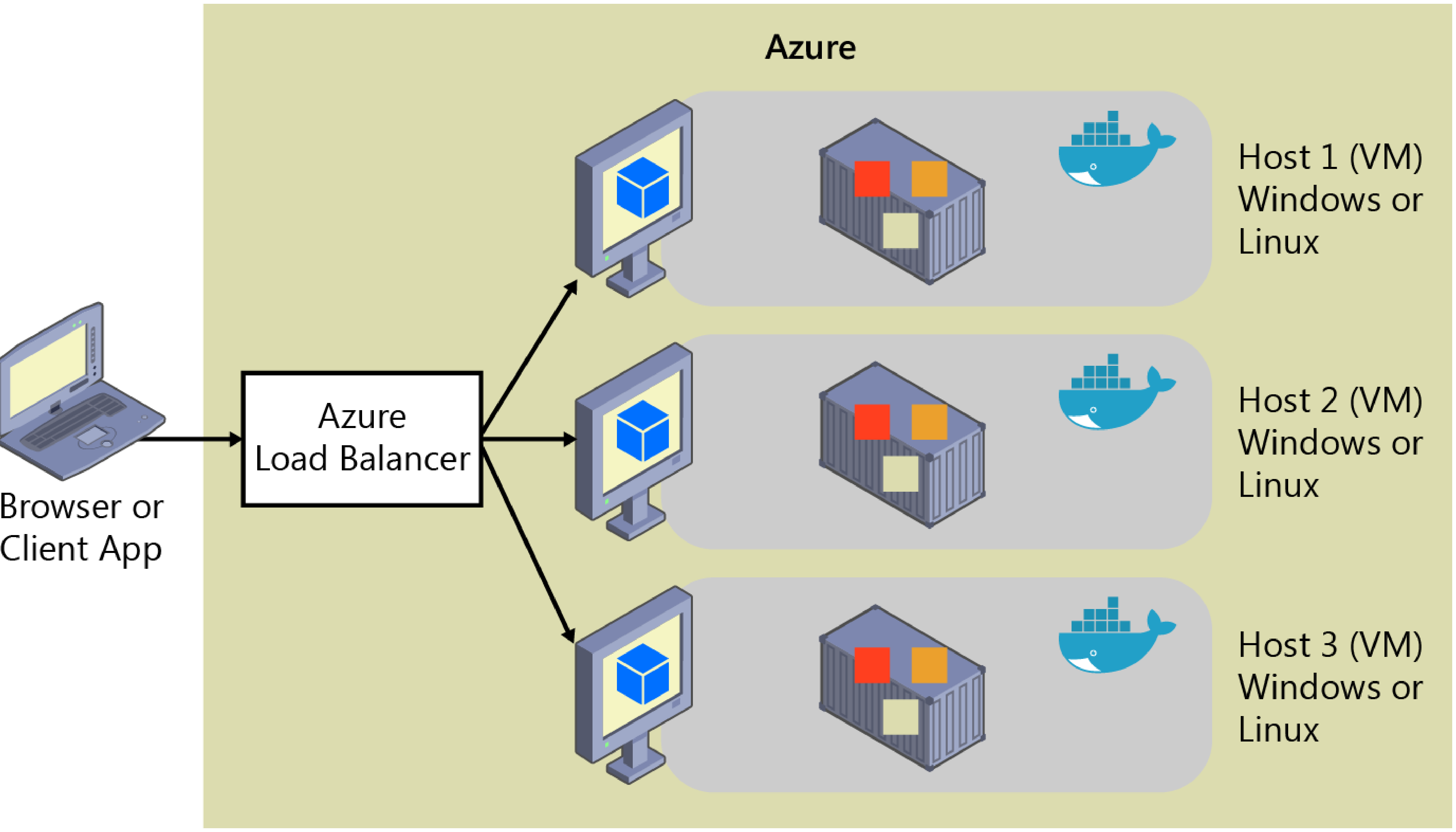

amazon
Purchased on June 11, 2008.
1
Chris Richardson
POJOs in Action: Developing Enterprise Applications with Lightweight Frameworks
3
(21)
2
3
Format: Paperback
List Price: $44.99
Price: $29.39 Prime
Save: $15.56 (35%)
3
Frequently bought together
4
Buy Both: $60.26
About this item
3
Description
A solid, valuable and easy-to-read work. – JavaRanch
Features & details
Publisher: Manning Publications
Publication date: February 2, 2006
Language: English
Product Dimensions: 7.5 x 1.3 x 9.5 inches
Customers who bought this also bought
Only 3 left in stock (more on the way).
Ships from and sold by Amazon.com. Gift-wrap available.
5
Buy now with 1-Click
Order within 10 hr 53 min to get it Sunday, June 7 (Free Two-Day Shipping)
6
Add to Cart
Add to Wish List
New & used (58) from $0.45
Share
Sell on Amazon
Frequently bought together

1. ORDER HISTORY
2. REVIEWS
3. BASIC PRODUCT INFO
4. RECOMMENDATION
5. INVENTORY
6. SHIPPING

Building Microservices: Using an API Gateway

- How can a mobile client retrieve such data in a monolithic application architecture?
- A mobile client would retrieve this data by making a single REST call (GET api.company.com/productdetails/productId) to the application.
- A load balancer routes the request to one of N identical application instances. The application would then query various database tables and return the response to the client.

Building Microservices: Using an API Gateway

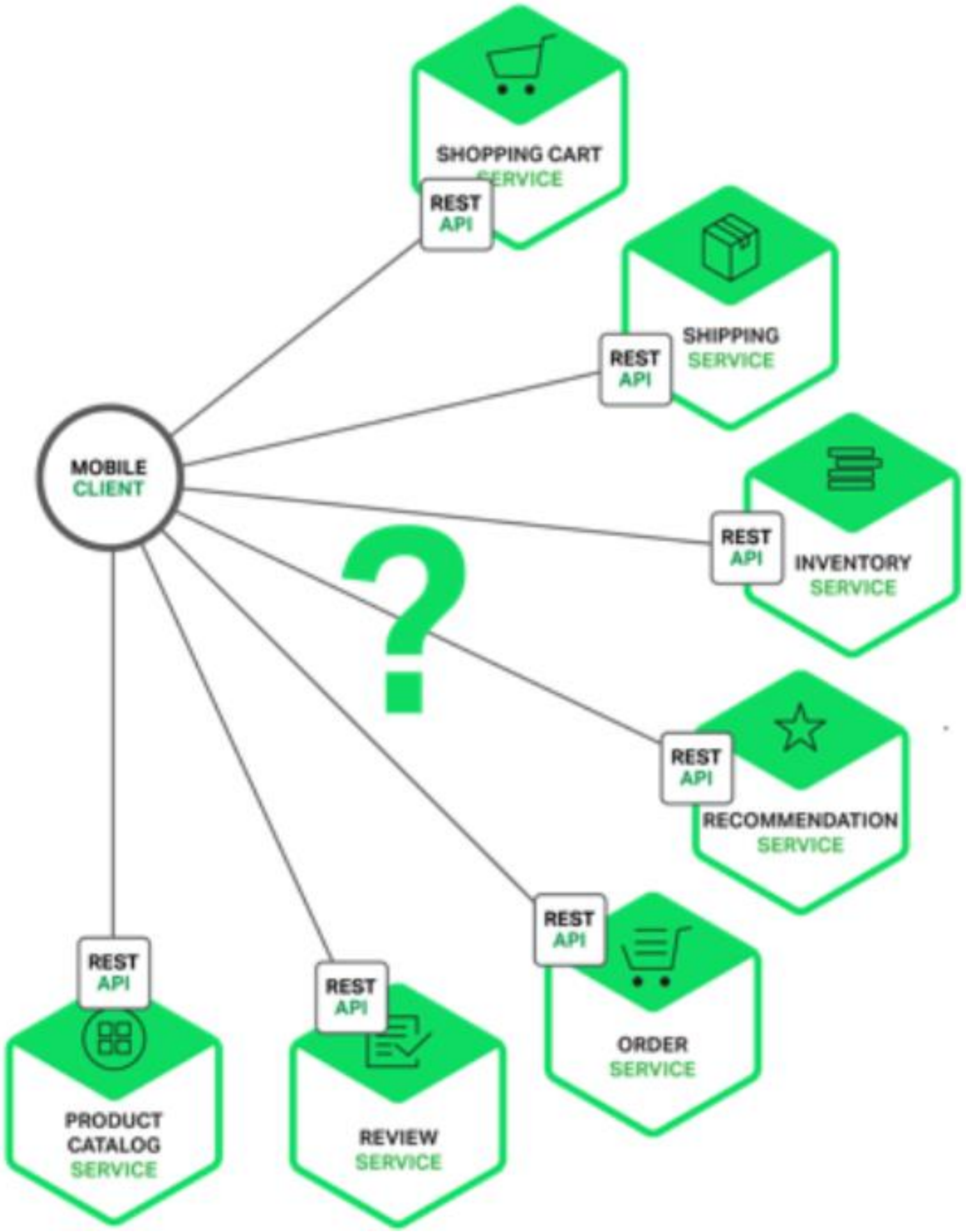

Azure
Browser or Client App
Azure Load Balancer
Host 1 (VM) Windows or Linux
Host 2 (VM) Windows or Linux
Host 3 (VM) Windows or Linux

Building Microservices: Using an API Gateway

- Consider of an alternate microservice architecture…
- What services could be present to provide the same data?
  - Shopping Cart Service – Number of items in the shopping cart
  - Order Service – Order history
  - Catalog Service – Basic product information, such as its name, image, and price
  - Review Service – Customer reviews
  - Inventory Service – Low inventory warning
  - Shipping Service – Shipping options, deadlines, and costs drawn separately from the shipping provider's API
  - Recommendation Service(s) – Suggested items

Building Microservices: Using an API Gateway

- How should the mobile client access those services?



Building Microservices: Using an API Gateway

- How should the mobile client access those services?

1. Direct client to microservice communication (how?)
   - Limitations:
     1. The mismatch between the needs of the client and the fine-grained APIs exposed by each of the microservices.
     2. Enforcing specific protocols for communication
     3. Refactoring the webservices becomes difficult

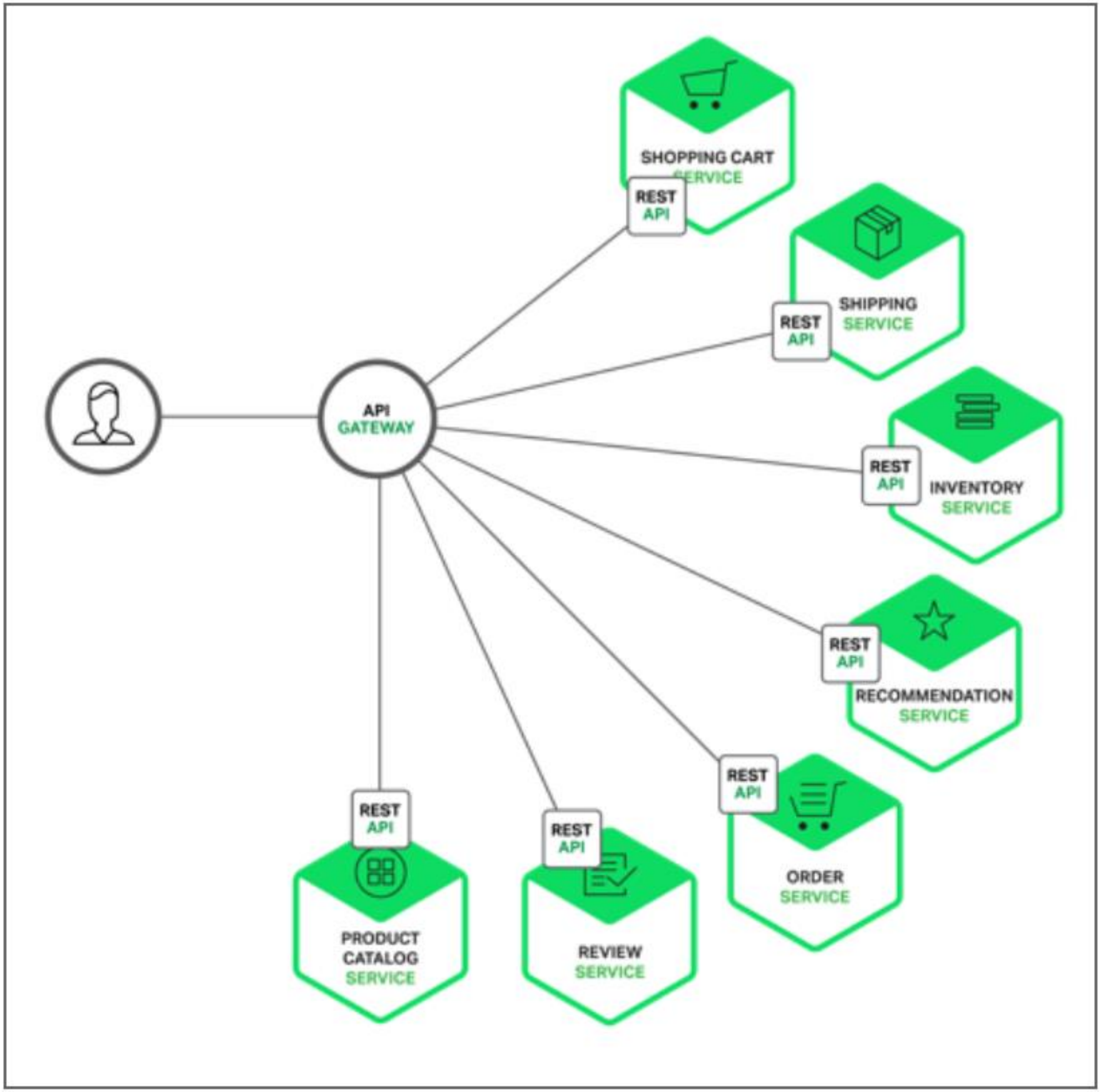
Building Microservices: Using an API Gateway

- How should the mobile client access those services?

2. Using an API gateway

- An API Gateway is a server that is the single entry point into the system (Façade pattern?)
- The API Gateway encapsulates the internal system architecture and provides an API that is tailored to each client.
- It might have other responsibilities such as authentication, monitoring, load balancing, caching, request shaping and management, and static response handling.

Buildi   way



SHOPPING CART SERVICE
REST API
SHIPPING SERVICE
REST API
API GATEWAY
REST API
INVENTORY SERVICE
REST API
RECOMMENDATION SERVICE
REST API
ORDER SERVICE
REST API
PRODUCT CATALOG SERVICE
REST API
REVIEW SERVICE

Building Microservices: Using an API Gateway

2. Using an API gateway

- Coarse-grained APIs are provided instead.
- Consider, for example, the product details scenario.
- The API Gateway can provide an endpoint (**/productdetails?productid=*xxx***) that enables a mobile client to retrieve all of the product details with a single request.
- The API Gateway handles the request by invoking the various services – product info, recommendations, reviews, etc. – and combining the results.

Case Study: Netflix API Gateway

- The Netflix streaming service is available on hundreds of different kinds of devices (e.g., televisions, smartphones, gaming systems, tablets, …etc)
- Netflix attempted to provide a one-size-fits-all API for their streaming service.
- The provided API didn’t work well because of the diverse range of devices and their unique needs.
- They reverted to using an API Gateway that provides an API tailored for each device. (how?)
- Pros/Cons of API gateways?

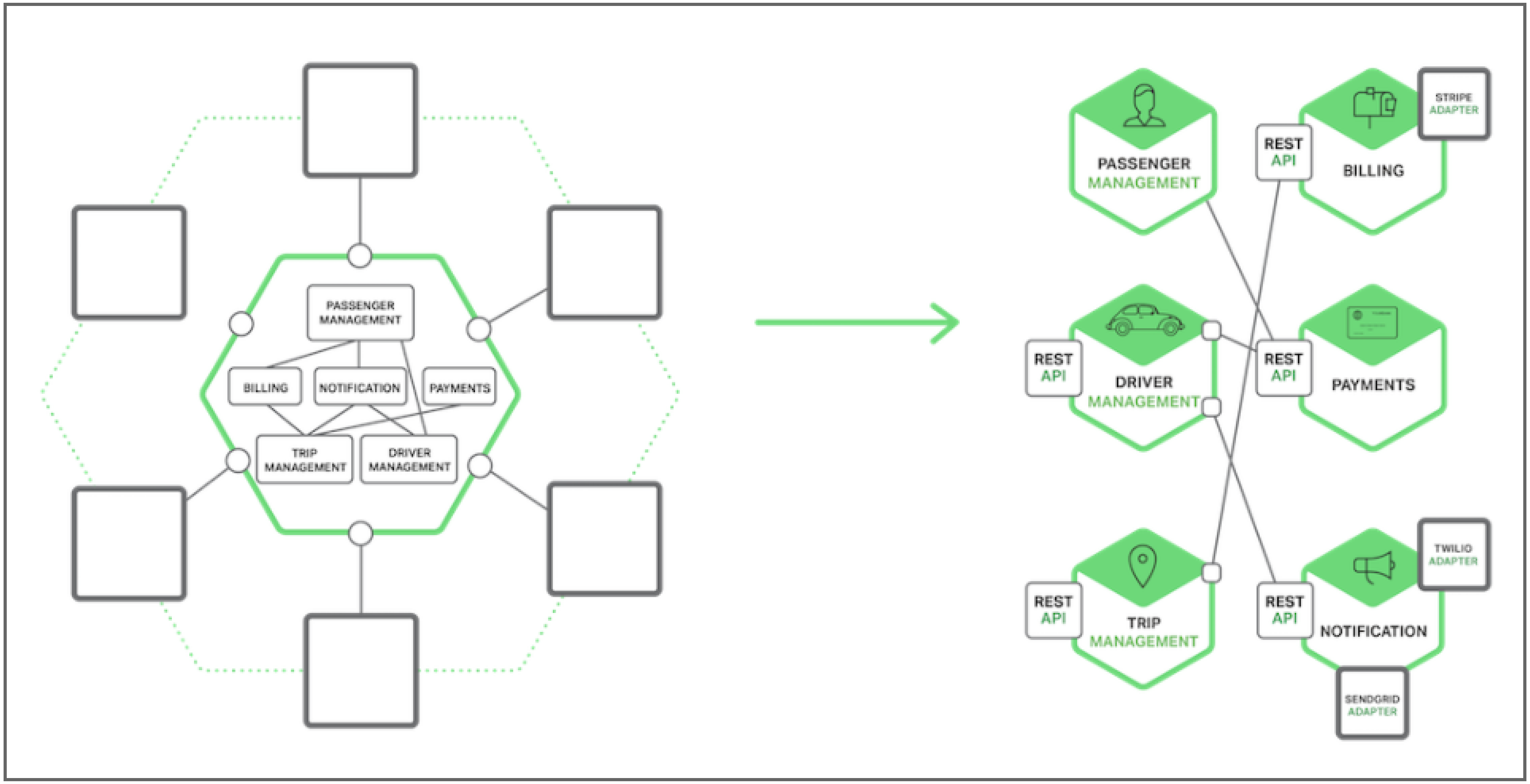
Using an API Gateway: Design Considerations

- Performance and scalability
- Service invocation
- Service discovery
- Handling partial failures

Inter-Process Communication in a Microservice Architecture

- In a monolithic application, components invoke one another via language-level method or function calls.
- In contrast, a microservices-based application is a distributed system running on multiple machines.
- Each service instance is typically a process.

Inter-Process Communication in a Microservice Architecture

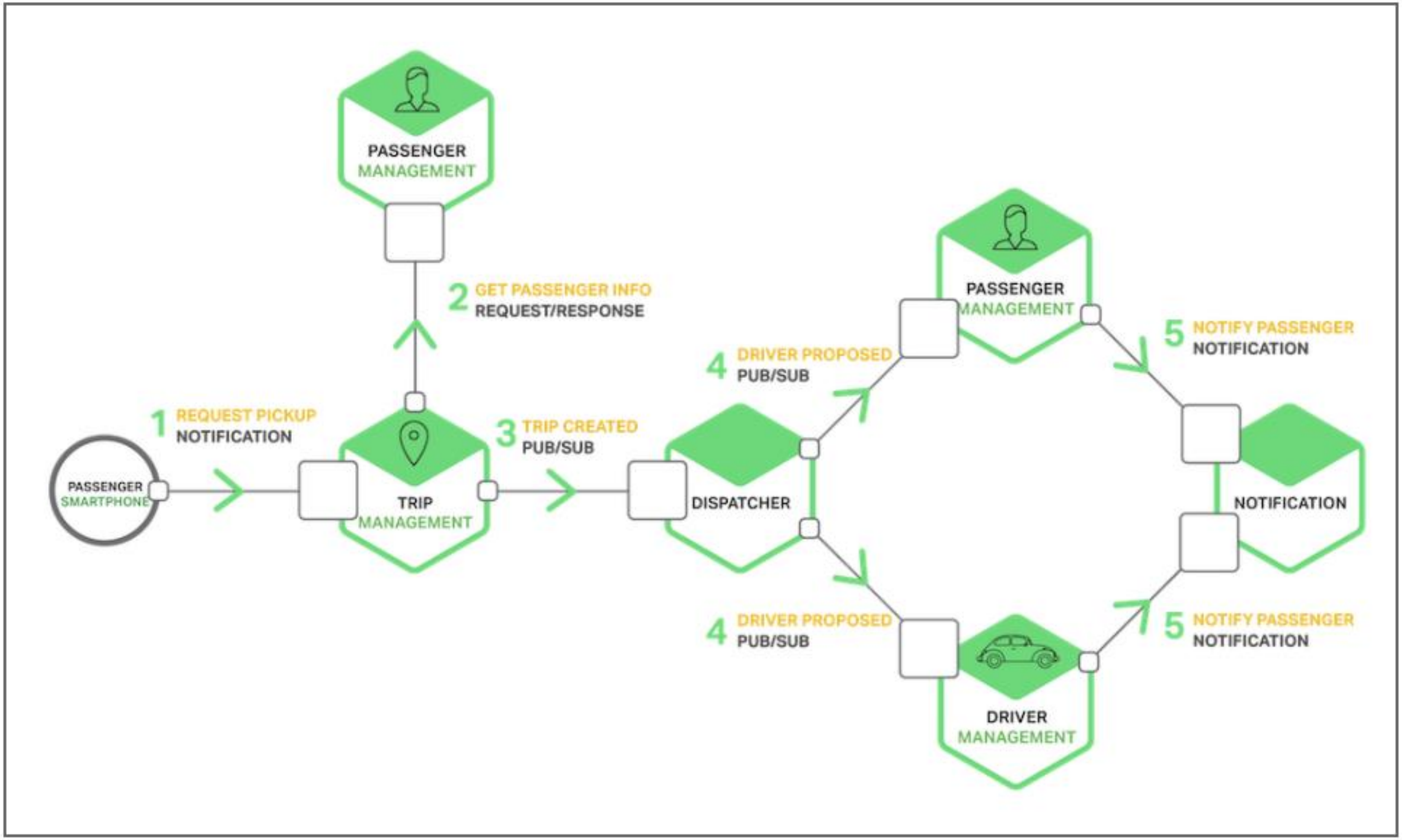


Inter-Process Communication in a Microservice Architecture

There are the following kinds of one-to-one interactions:

- Request/response – A client makes a request to a service and waits for a response.
- Notification (publish/subscribe) – A client sends a request to a service but no reply is expected or sent.
- Request/async response [Message queues] – A client sends a request to a service, which replies asynchronously.

Inter-Process Communication in a Microservice Architecture


PASSENGER MANAGEMENT
2 GET PASSENGER INFO REQUEST/RESPONSE
1 REQUEST PICKUP NOTIFICATION
PASSENGER SMARTPHONE
TRIP MANAGEMENT
3 TRIP CREATED PUB/SUB
DISPATCHER
4 DRIVER PROPOSED PUB/SUB
PASSENGER MANAGEMENT
5 NOTIFY PASSENGER NOTIFICATION
NOTIFICATION
4 DRIVER PROPOSED PUB/SUB
DRIVER MANAGEMENT
5 NOTIFY PASSENGER NOTIFICATION

Required Readings

- https://martinfowler.com/articles/microservice-trade-offs.html
- https://martinfowler.com/articles/microservices.html
- https://www.nginx.com/blog/introduction-to-microservices/
- https://www.nginx.com/blog/building-microservices-using-an-api-gateway/
- https://www.nginx.com/blog/building-microservices-inter-process-communication/