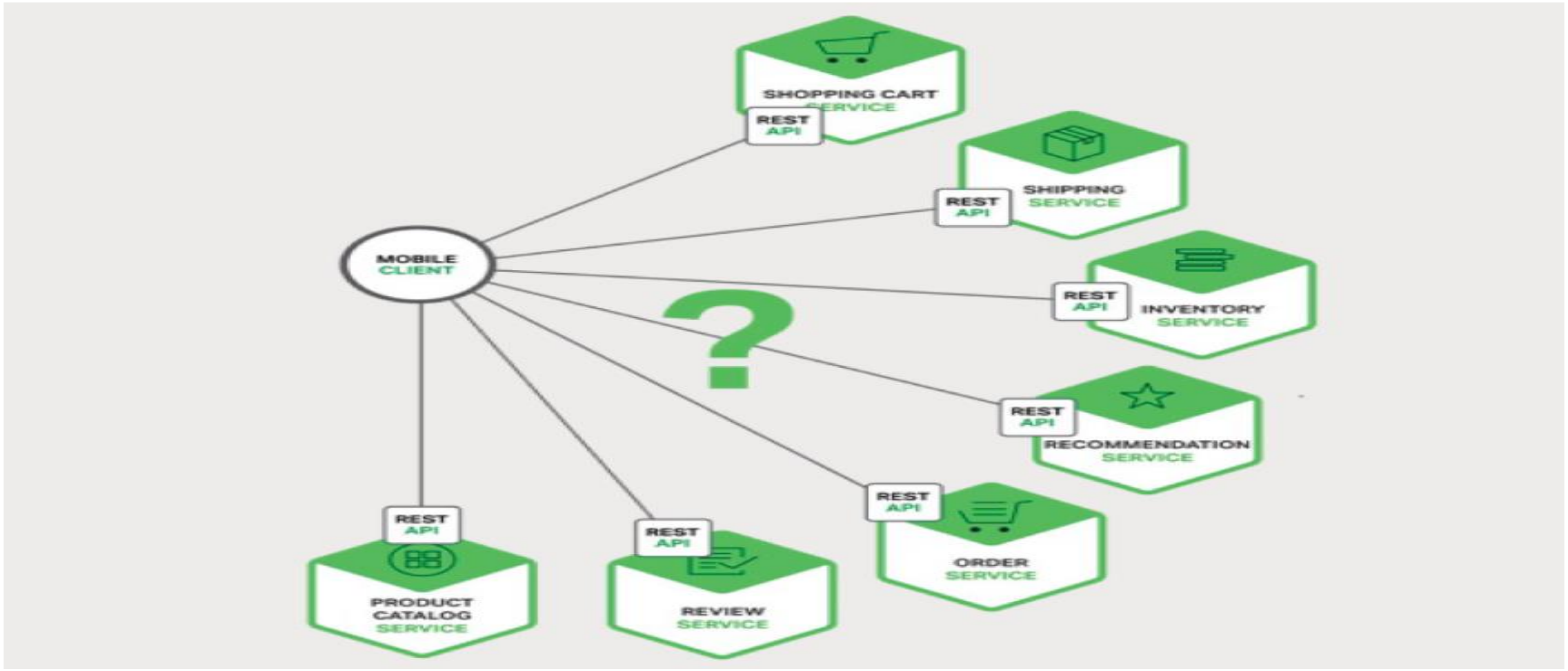
Using an API Gateway

Vikash Verma

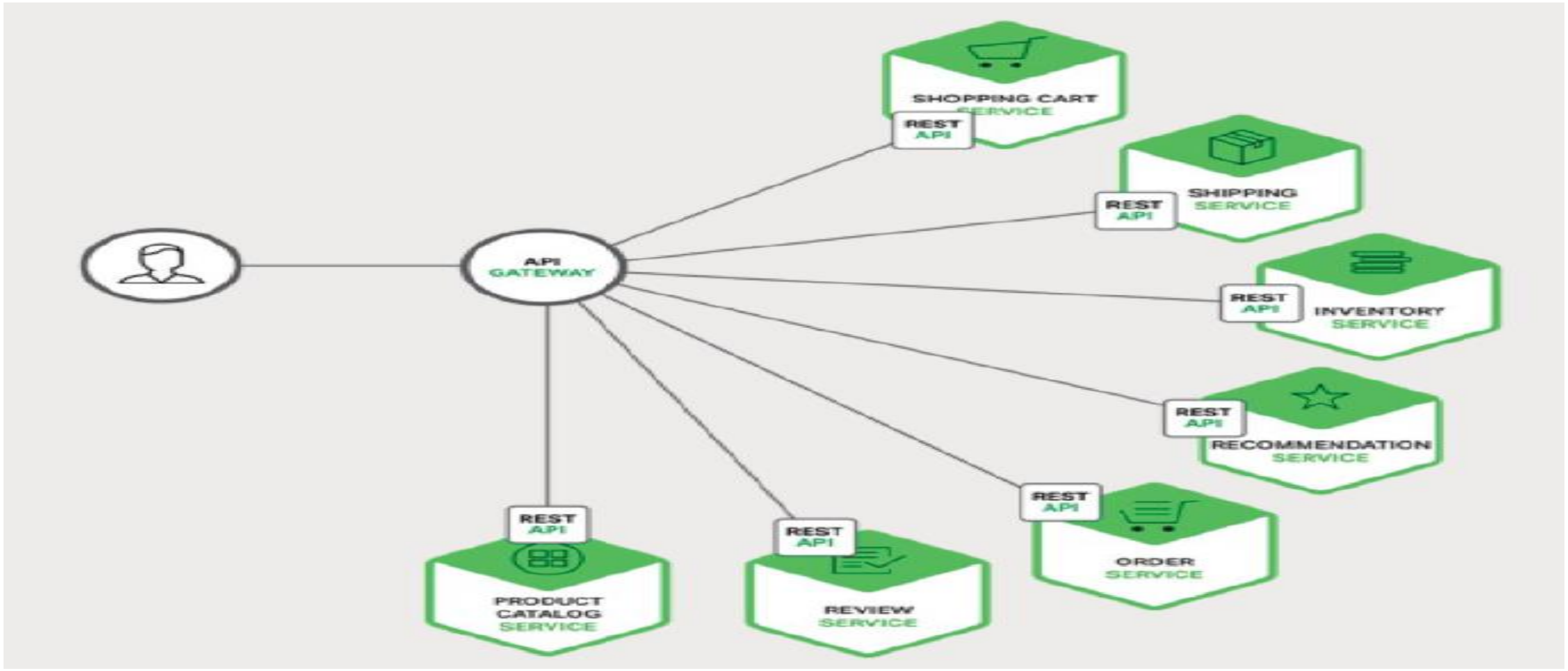
Introduction



Using an API Gateway

- An API Gateway is a server that is the single entry point into the system.
- It is similar to the Façade pattern from object-oriented design.
- 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.
- The API Gateway is responsible for request routing, composition, and protocol translation.
- All requests from clients first go through the API Gateway. It then routes requests to the
- appropriate microservice.
- The API Gateway will often handle a request by invoking multiple microservices and aggregating the results.
- It can translate between web protocols such as HTTP and WebSocket and web-unfriendly protocols that are used internally.

Using an API Gateway



Benefits and Drawbacks of an API Gateway

- Advantages
  - API Gateway is that it encapsulates the internal structure of the application.
  - Rather than having to invoke specific services, clients simply talk to the gateway.
  - The API Gateway provides each kind of client with a specific API. This reduces the number of round trips between the client and application. It also simplifies the client code.
- Disadvantages
  - It is yet another highly available component that must be developed, deployed, and managed.
  - There is also a risk that the API Gateway becomes a development bottleneck. Developers must update the API Gateway in order to expose each microservice’s endpoints.
  - It is important that the process for updating the API Gateway be as lightweight as possible. Otherwise, developers will be forced to wait in line in order to update the gateway.

Implementing an API Gateway

- API Gateway should be built on a platform that supports asynchronous, non-blocking I/O.
- There are a variety of different technologies that can be used to implement a scalable API Gateway.
  - JVM
    - Netty, Vertx, Spring Reactor
  - Node.Js
  - NGINX Plus
  - Asp.net Core

Using a Reactive Programming Model

- Avoid callback hell using traditional event based asynchronous patterns
- Use async and await

Thank you