From “How” to “What” – APIs as the Central Architectural Feature in an eCommerce Architecture

Chuck Rhoades, Pizza Hut
What is a “Microservice”? 

- General Concepts
  - “Small” independent services
  - Automated deployed via CI/CD
  - Automatic service discovery
What is a “Microservice”? 

• General Concepts 
  – “Small” independent services 
  – Automated deployment via CI/CD 
  – Automatic service discovery 
  – Containerized (Docker) 
  – Managed via container orchestration (Kubernetes)
But what about service boundaries?

What exactly does “small” mean?
Use Case: Find me a store that can serve me how and what I want.

Store Service

- **FindNearByStores**: GET /stores/near?lat={lat}&lng={lng}
- **FindDeliveryStore**: GET /stores/deliversTo?lat={lat}&lng={lng}
- **GetStoreDetails**: GET /stores/{storeId}
Use Case: Find me a store that can serve me how and what I want.

Store Service

- **FindNearByStores**: GET /stores/near?lat={lat}&lng={lng}
- **FindDeliveryStore**: GET /stores/deliversTo?lat={lat}&lng={lng}
- **GetStoreDetails**: GET /stores/{storeId}
- **UpdateStoreDetails**
- **AddStores**
- **RemoveStores**
- And Many More...

~20 Endpoints
Service Boundaries

• As small as possible while providing meaningful business function
• Must fully encapsulate the business function and data it provides
Use Case: Find me a store that can serve me how and what I want.
Directly Expose Microservices?

- Too many services = cognitive load
- Exposing sensitive endpoints
- Poor fit for some consumers (i.e. mobile apps)
- Designed for the consumer
- Generated from OpenAPI Spec
- Minimal business logic
- Google Cloud Endpoint Proxy

- Designed for developers
- Contain business logic
- Adhere to service boundary policy
What did we learn?

- Our REST layer works, but gets a bit boilerplate-ish
  - Lots of duplication from protobuf to OpenAPI Specs
What did we learn?

• Our REST layer works, but gets a bit boilerplate-ish
  – Lots of duplication from protobuf to OpenAPI Specs
  – gRPC Gateway per RPC, Apigee for composing API Products
What did we learn?

• Our REST layer works, but gets a bit boilerplate-ish
  – Lots of duplication from protobuf to OpenAPI Specs
  – gRPC Gateway per RPC, Apigee for composing API Products

• Well defined gRPC services in k8 clusters works really well
  – But, Kubernetes has a gap in service to service communication
What did we learn?

• Our REST layer works, but gets a bit boilerplate-ish
  – Lots of duplication from protobuf to OpenAPI Specs
  – gRPC Gateway per RPC, Apigee for composing API Products

• Well defined gRPC services in k8 clusters works really well
  – But, Kubernetes has a gap in service to service communication
  – Istio to establish a single service mesh across clusters