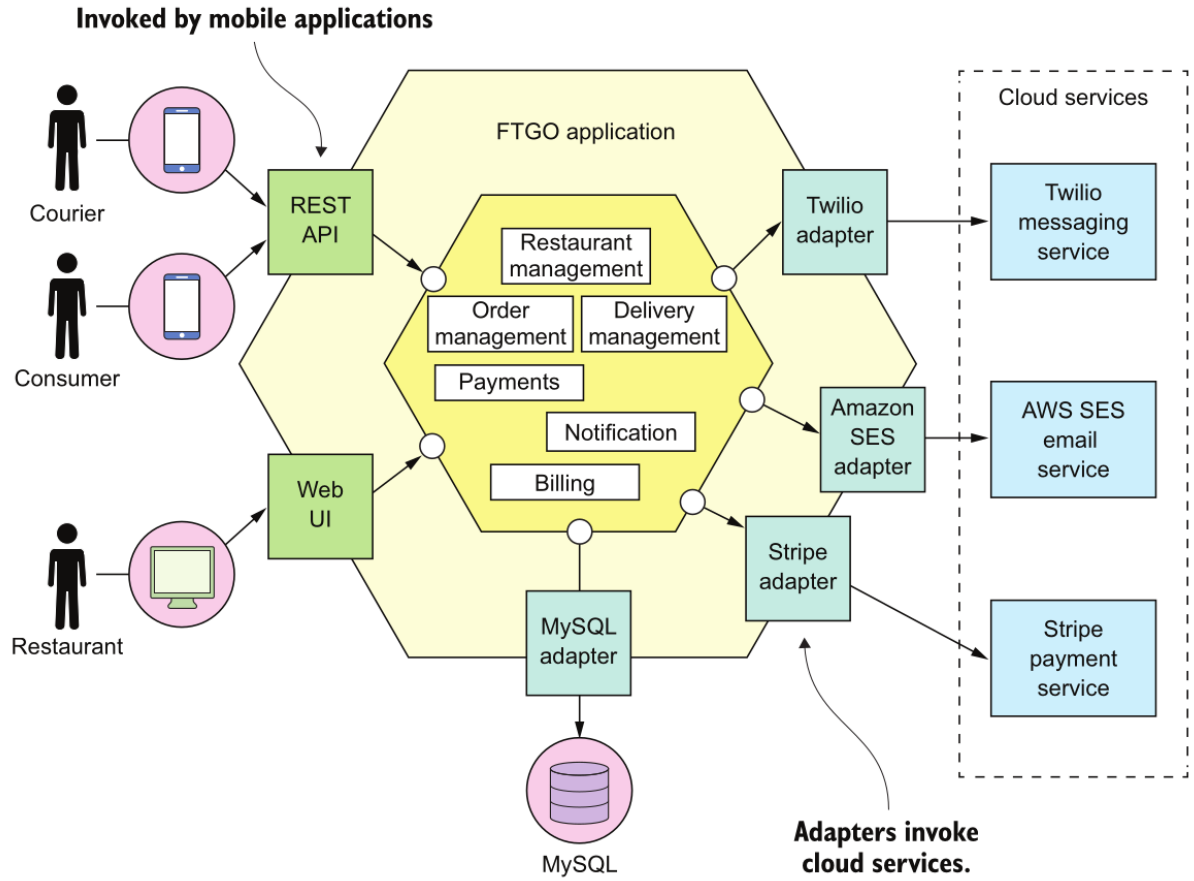
Event-Driven Arch

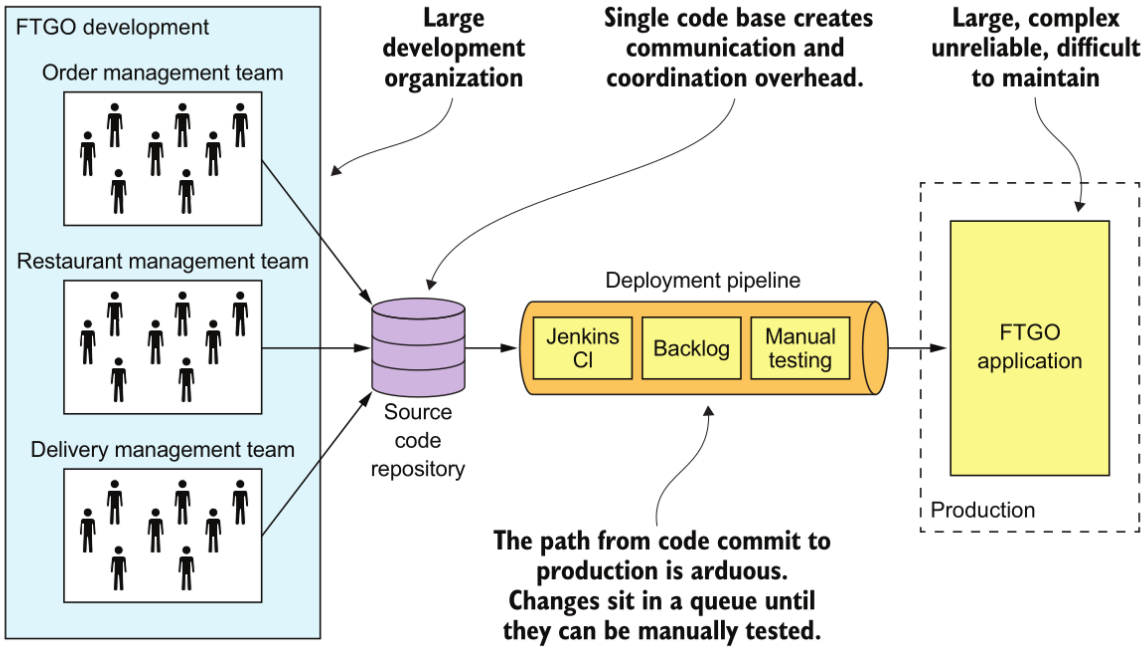
Evolving your business will require change

Blocker: `The Monolith Arch`

The `Monolith` architecture of the Food to Go, Inc. (`FTGO`) application



Living in monolithic hell



1. COMPLEXITY INTIMIDATES DEVELOPERS
2. DEVELOPMENT IS SLOW
3. PATH FROM COMMIT TO DEPLOYMENT IS LONG AND ARDUOUS
4. DELIVERING A RELIABLE MONOLITH IS CHALLENGING
5. SCALING IS DIFFICULT
6. HIGH RISK OF FAILURES
7. LOCKED INTO INCREASINGLY OBSOLETE TECHNOLOGY STACK

Tenets for faster application development

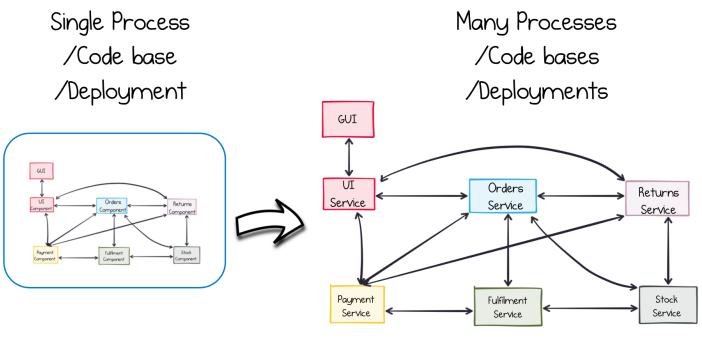
1. Enable teams to build independently
2. Eliminate dependencies in feature delivery

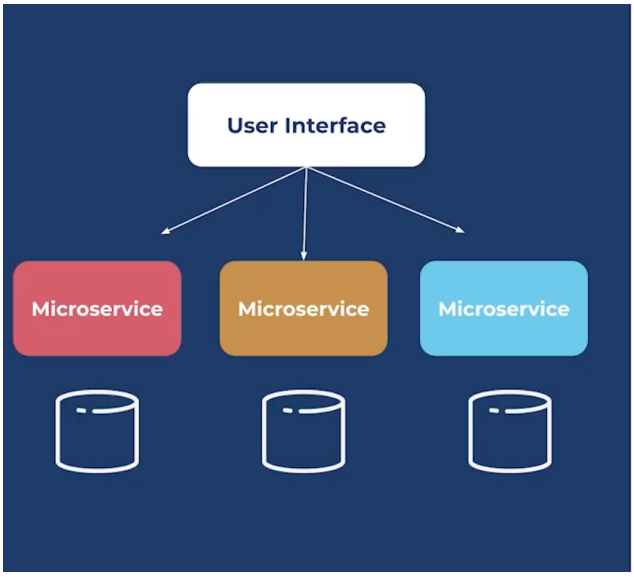
3. Scale services with smaller & Agile teams
4. Reduce infrastructure cost & complexity

Solution: `Microservices Arch` | `Distributed Systems`

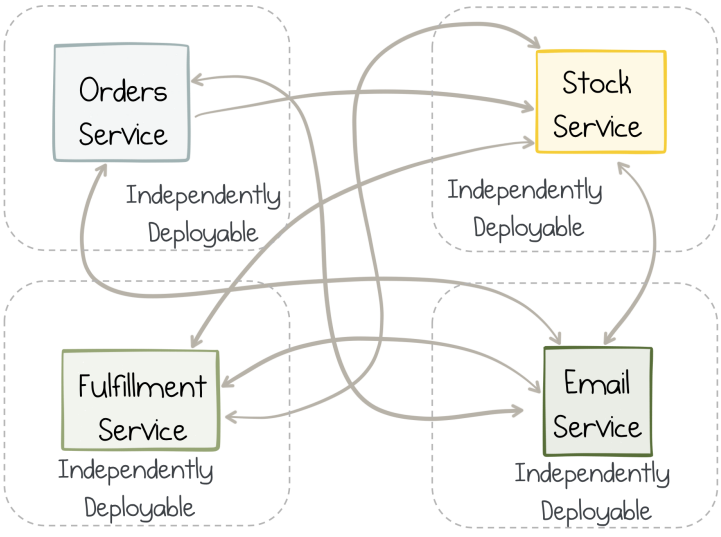
What are microservices really about?

Splitting the Monolith?



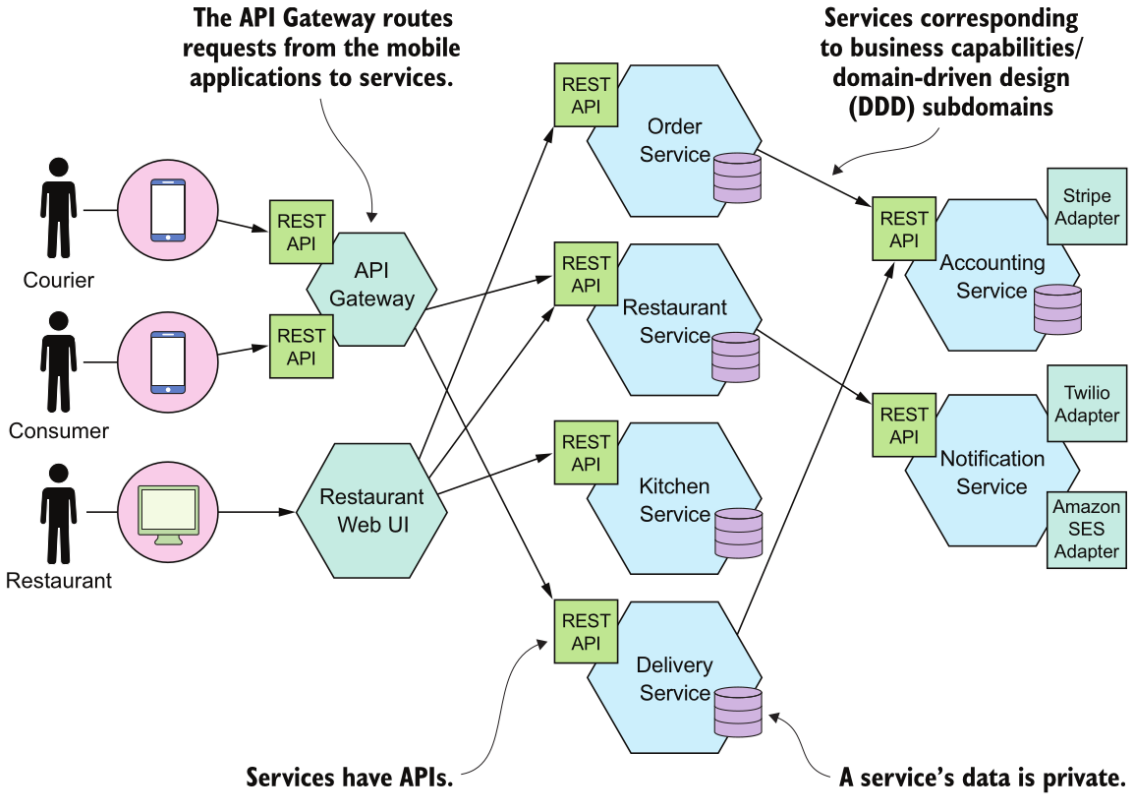


- Multiple smaller, single function apps
- Built around solving business problems
- Independently deployable and upgradable



Independence is where services get their value

The `FTGO` `microservice` architecture



 Order Service —Manages orders

 Delivery Service —Manages delivery of orders from restaurants to consumers

 Restaurant Service —Maintains information about restaurants

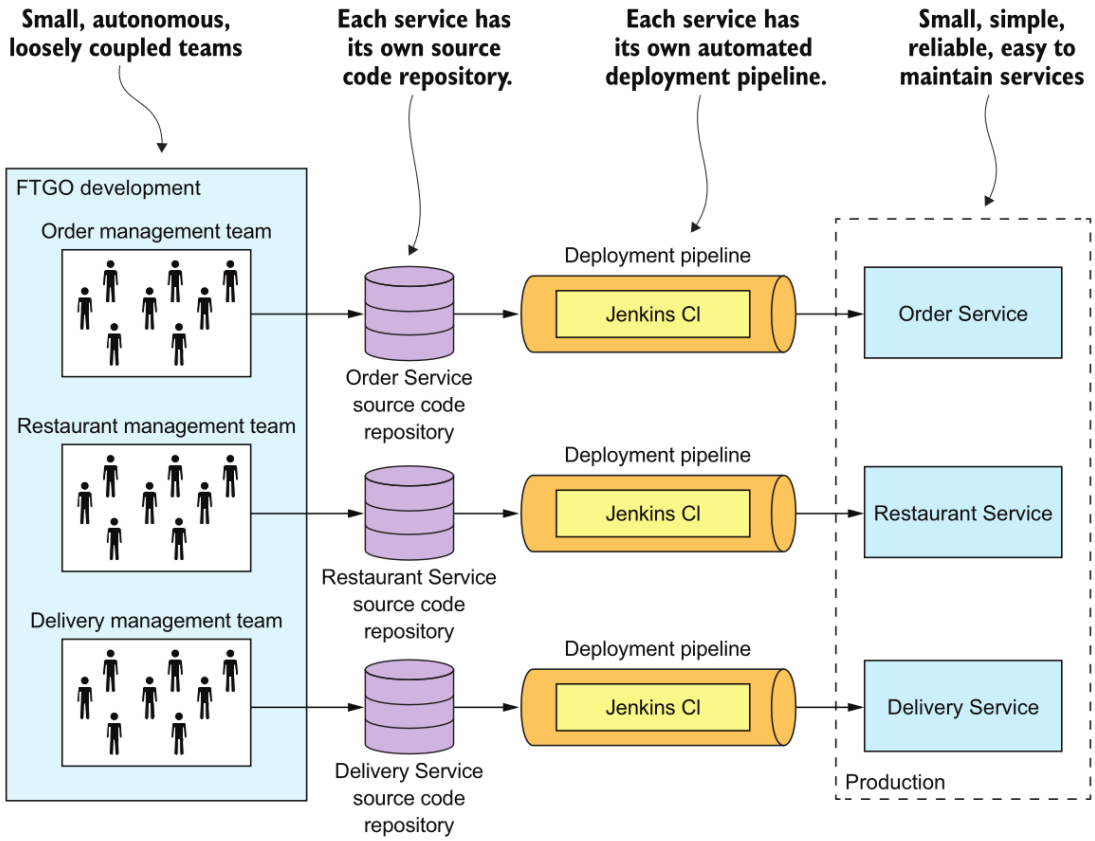
 Kitchen Service —Manages the preparation of orders

 Accounting Service —Handles billing and payments

---

1. It enables the continuous delivery and deployment of large, complex applications.

Benefits of the `microservice` architecture



2. Services are small and easily maintained.
3. Services are independently deployable.
4. Services are independently **scalable**.
5. Enables teams to be autonomous.
6. It allows easy experimenting and adoption of new technologies.
7. It has better fault isolation.

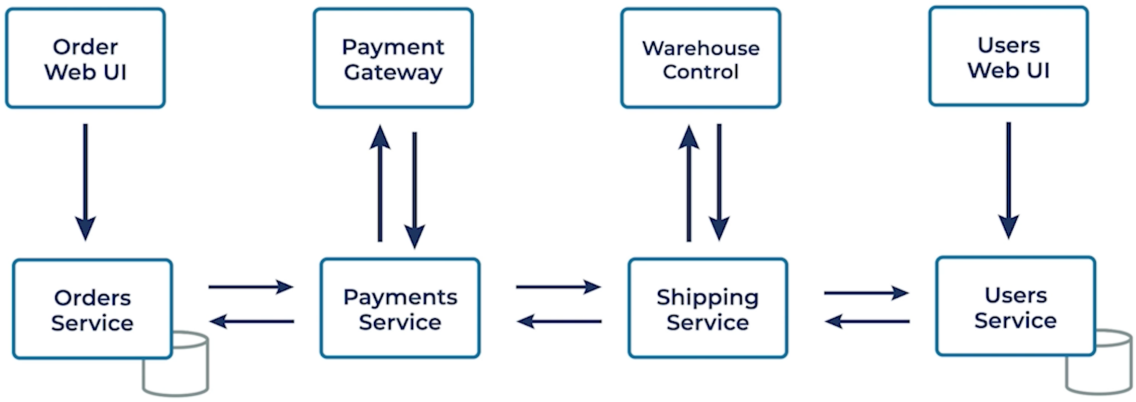
Approaches for developing Microservices

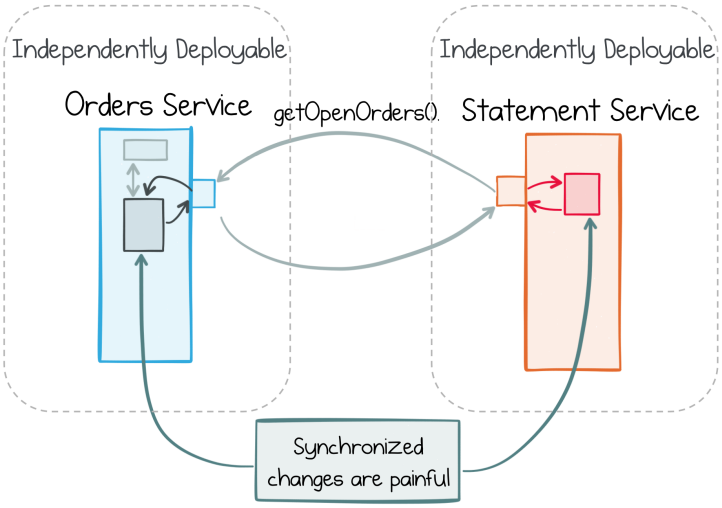
Remote Procedure calls (e.g HTTP REST API)	Messaging Queues

Simplified setup, relatively efficient delivery	Message brokers act as a centralized messaging service through which all Mcroservices communicate
Synchronous communications: client sends request and waits for response	Brokers handle messaging queueing, HA and reliable communication b/w services

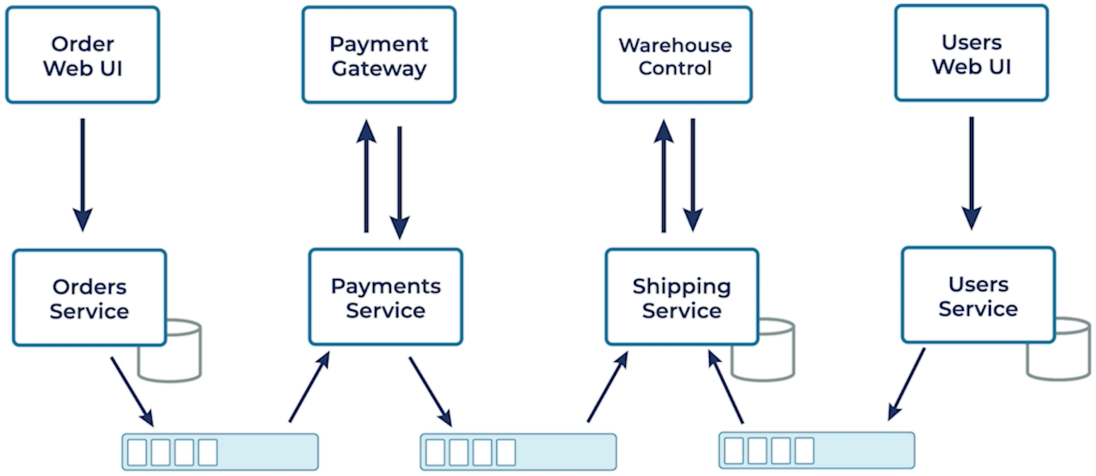
---

A Minimal microservice implementation (RPC)





A Minimal microservice implementation (Refactoring to Events)



https://docs.google.com/presentation/d/1Hdev_Feq7YF9mfnGJvgWLXse4_HAPhML1wDhBDNnVMo/edit?usp=sharing

https://www.reactivemanifesto.org/