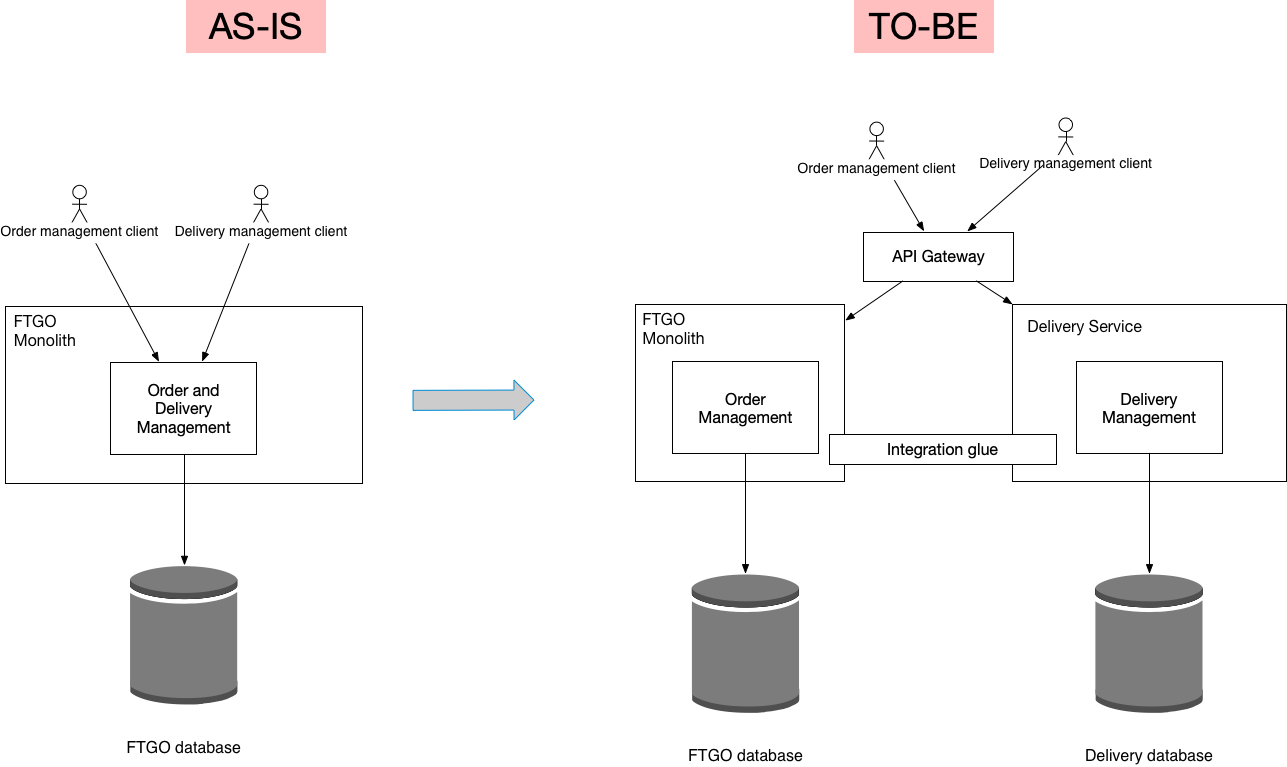
Extract services from the monolith

While implementing new functionality as services is extremely useful, the only way of eliminating the monolith is to incrementally extract modules out of the monolith and convert them into services. For example, let’s imagine that the FTGO team wants to improve the efficiency of the business and customer satisfaction by rapidly iterating on the courier scheduling algorithm. It will be a lot easier for them to work on the delivery management logic if it’s a separate Delivery Service. To do that, the FTGO team must separate delivery management from order management and convert it into service.

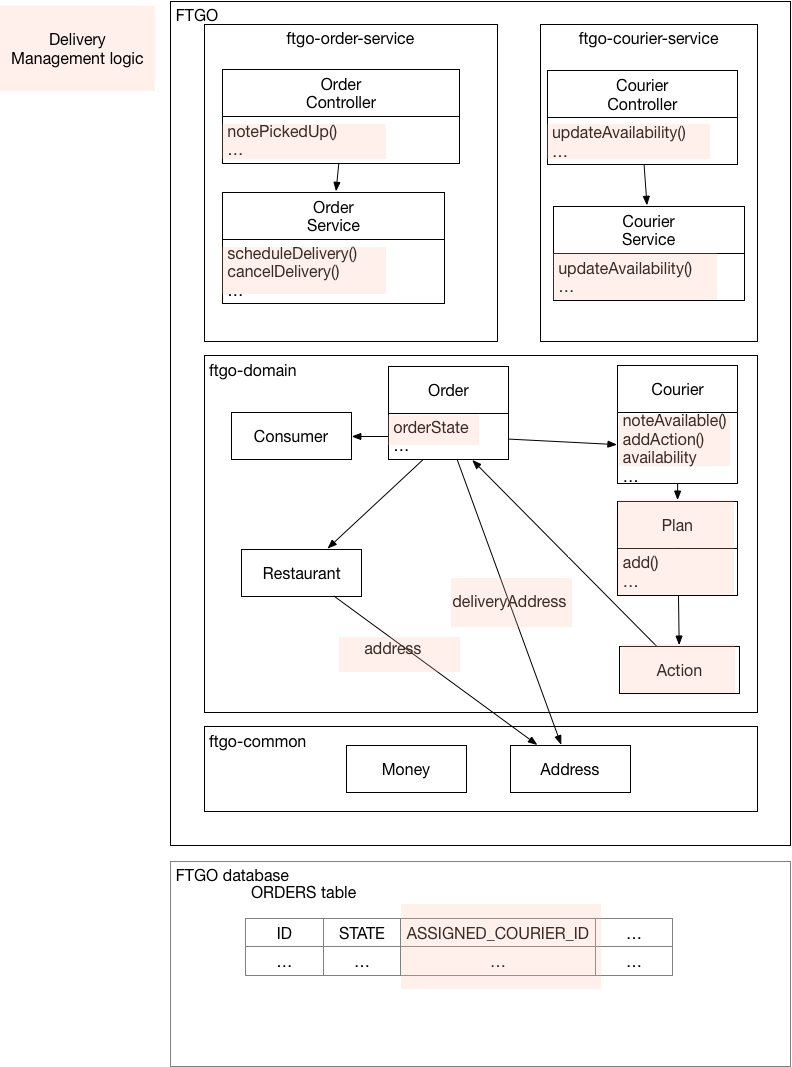


Extracting the Delivery Service consists of the following the steps:

1. Split the code and convert delivery management into a separate, loosely coupled module within the monolith
2. Split the database and define a separate schema for delivery management.
3. Define a standalone Delivery Service
4. Use the standalone Delivery Service
5. Remove the old and now unused delivery management functionality from the FTGO monolith

Before looking at each of these steps, let’s first review the AS-IS code.

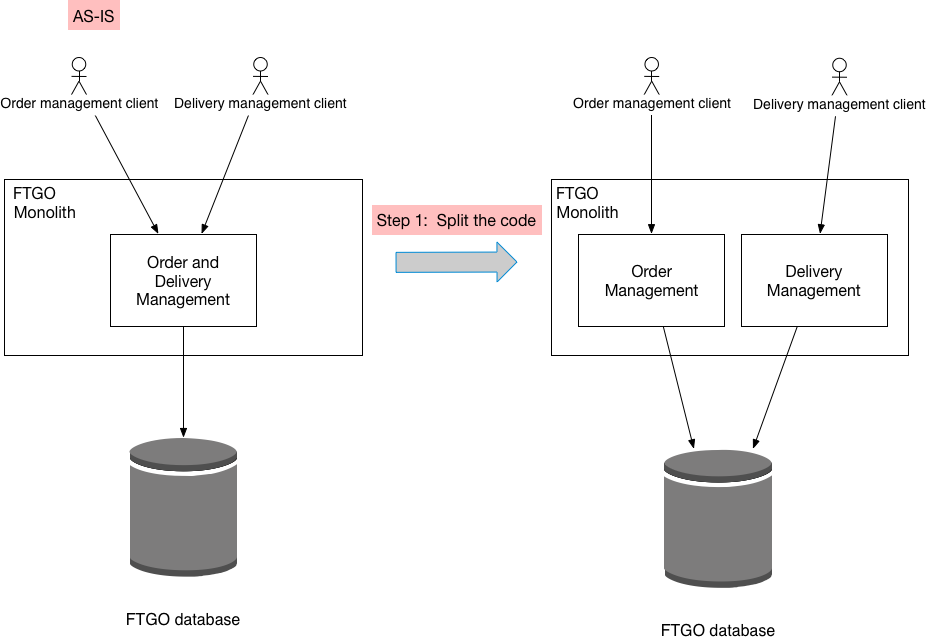
Step 0: Review AS-IS code



In the [FTGO monolithic application](https://github.com/microservices-patterns/ftgo-monolith), Order management and Delivery management are intertwined as shown in this diagram.

[Read more](https://microservices.io/refactoring/extract-service-delivery-service-step-0.html) about the AS-IS architecture.

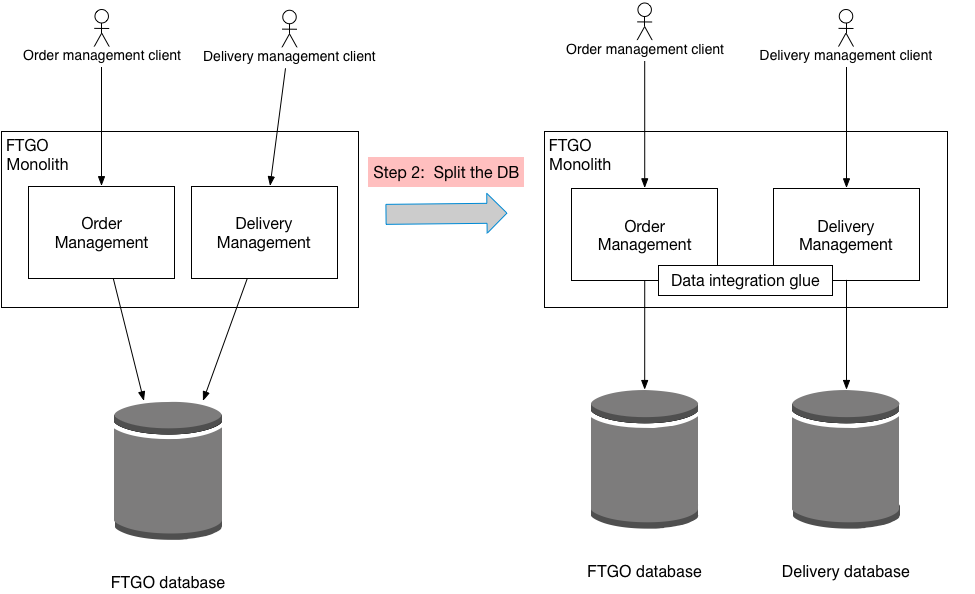
Step 1: Split the code



The first step is to split the code and convert delivery management into a separate, loosely coupled module within the monolith.

[Read more](https://microservices.io/refactoring/extract-service-delivery-service-step-1.html) about how the code is split.

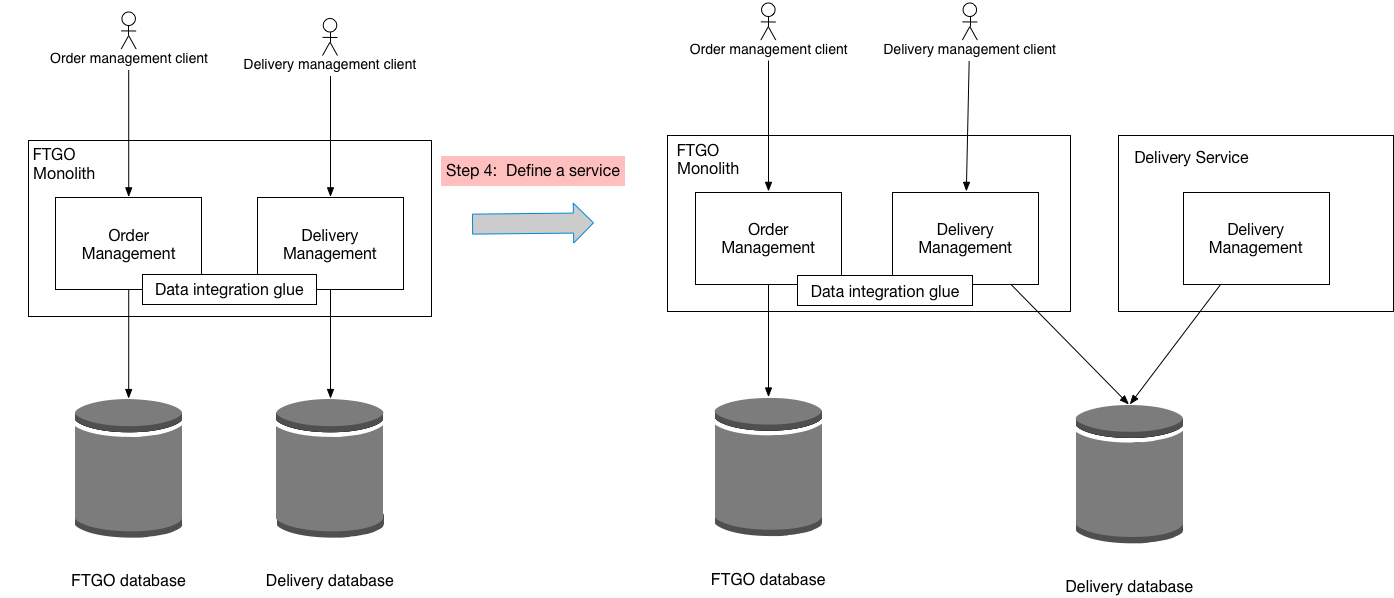
Step 2: Split the database



The second step is split the database and define a separate database schema for the ftgo-delivery-service module.

[Read more](https://microservices.io/refactoring/extract-service-delivery-service-step-2.html) about splitting the database.

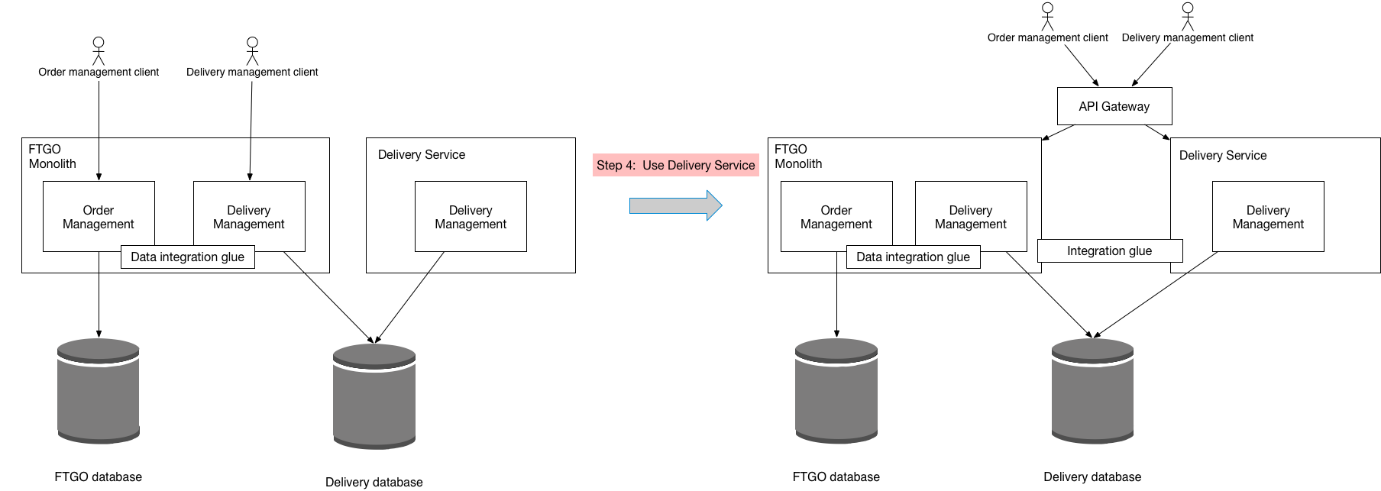
Step 3: Define a standalone Delivery Service



The third step of the refactoring process is to define a standalone Delivery Service and deploy it. The service does not, however, handle production traffic. Instead, it can, for example, be tested in production.

[Read more](https://microservices.io/refactoring/extract-service-delivery-service-step-3.html) about defining standalone Delivery Service.

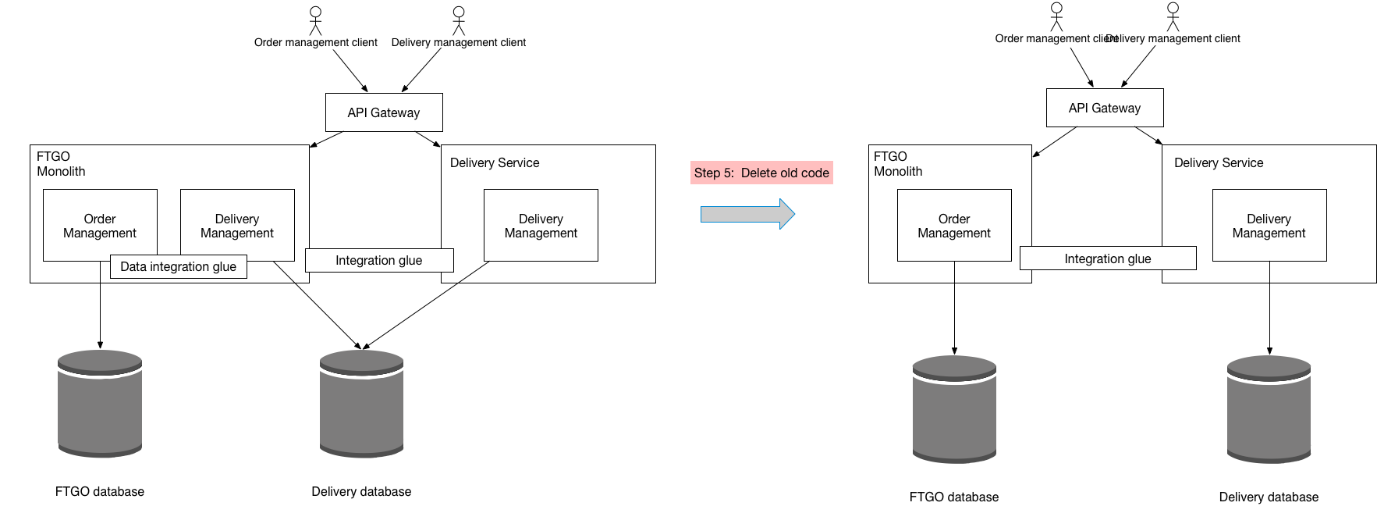
Step 4: Use the standalone Delivery Service



The fourth step of the refactoring process is to use the standalone Delivery Service.

[Read more](https://microservices.io/refactoring/extract-service-delivery-service-step-4.html) about how to route production traffic to the Delivery Service.

Step 5: Remove the delivery management functionality from the FTGO monolith



The fifth step of the refactoring process is to remove the now obsolete delivery management logic from the monolith.

[Read more](https://microservices.io/refactoring/extract-service-delivery-service-step-5.html) about removing the obsolete code.