CHAPTER 45

Containers and Google Kubernetes Engine

The microservice architecture is an approach for developing and deploying enterprise cloud-native software applications that involve separating the core business capabilities of the application into decoupled components. Each business capability represents some functionality that the application provides as services to the end user. The idea of microservices is in contrast to the monolithic architecture which involves building applications as a composite of its "individual" capabilities. See an illustration in Figure 45-1.

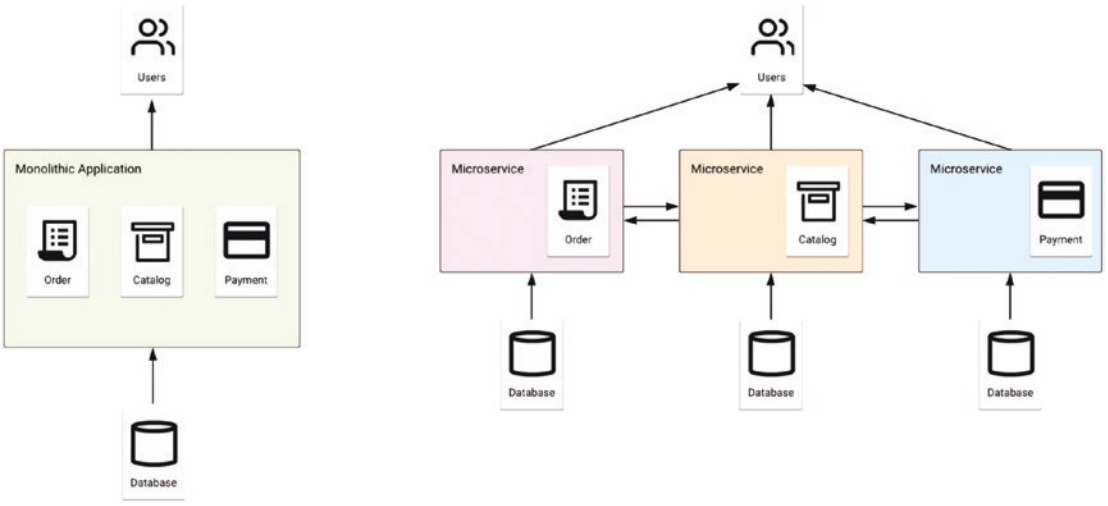

Figure 45-1. *Microservice applications (right) vs. monolithic applications (left)*

E. Bisong, *Building Machine Learning and Deep Learning Models on Google Cloud Platform*,
https://doi.org/10.1007/978-1-4842-4470-8_45

Microservices interact with each other using representational state transfer (REST) communications for stateless interoperability. By stateless, we mean that "the server does not store state about the client session." These protocols can be HTTP request/response APIs or an asynchronous messaging queue. This flexibility allows the microservice to easily scale and respond to request even if another microservice fails.

Advantages of Microservices

- Loosely coupled components make the application fault tolerant.
- Ability to scale out making each component highly available.
- The modularity of components makes it easier to extend existing capabilities.

Challenges with Microservices

- The software architecture increases in complexity.
- Overhead in management and orchestration of microservices. We will, however, see in the next sessions how Docker and Kubernetes work to mitigate this challenge.

Docker

Docker is a virtualization application that abstracts applications into isolated environments known as containers. The idea behind a container is to provide a unified platform that includes the software tools and dependencies for developing and deploying an application.

The traditional way of developing applications is where an application is designed and hosted on a single server. This is illustrated in Figure 45-2. This setup is prone to several problems including the famous "it works on my machine but not on yours". Also in this architecture, apps are difficult to scale and to migrate resulting in huge costs and slow deployment.