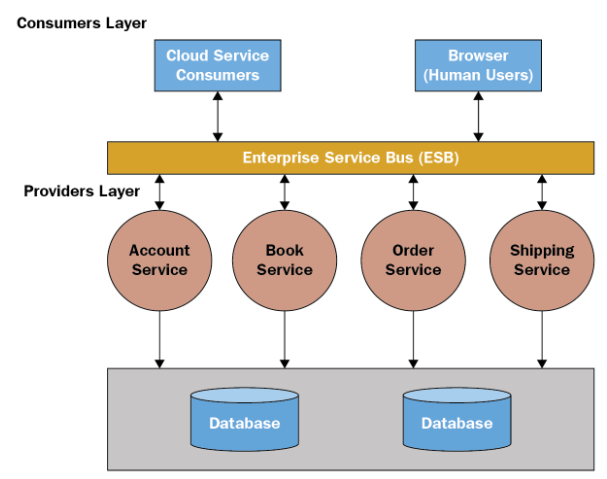
В монолите был 1 процесс, который предоставляет много функциональностей, в SOA же у нас несколько процессов, которые предоставляют функциональности

По сути у нас каждый отдельный сервис – это отдельное приложение, которое общается с другими сервисами по средством сети через API. Оно развернуто в отдельном docker контейнере

****

**Benefits**

* Так как функциональности разбиты на отдельные сервисы, то становится разработчикам менее соблазнительно объединять функциональности, что бы они зависили друг от друга
* Можно для каждой функциональности выкатывать отдельно свою версию (вносить изменения только в одну конкретную функциональность)
* Легче развертывание для каждого сервиса
* Можно развернуть только отдельные сервисы
* Если упадет какой то сервис, то перестанет работать только он, и может еще часть сервисов, которые зависят от него, но при этом остальная функциональность будет функционировать
* Разработчики менее вероятно будут имплементировать statefulness, из за того что база кода менее громоздкая (разбита на отдельные сервисы), то разработчки сможет подумать как имплементировать функциональность, которая не должна хранить состояние (из за этого легче горизонтально масштабировать чем монолит)

**Concerns**

* Коммуникация между серисами через сеть (сеть может упасть, появятся задержки между коммуникациями и тд). В монолите коммуникация между двумя различными функциональностями осуществлялась посредством вызова метода (функции). В SOA же коммуникация осуществляется посредством сети (REST, gRPC, MQ, …)
* Согласованность данных – так как у каждого сервиса может быть своя БД, данные могут быть не согласованными
* Развертывание всех сервисов сразу становится сложнее чем в Монолите (настраивание коммуникации между сервисами, …)