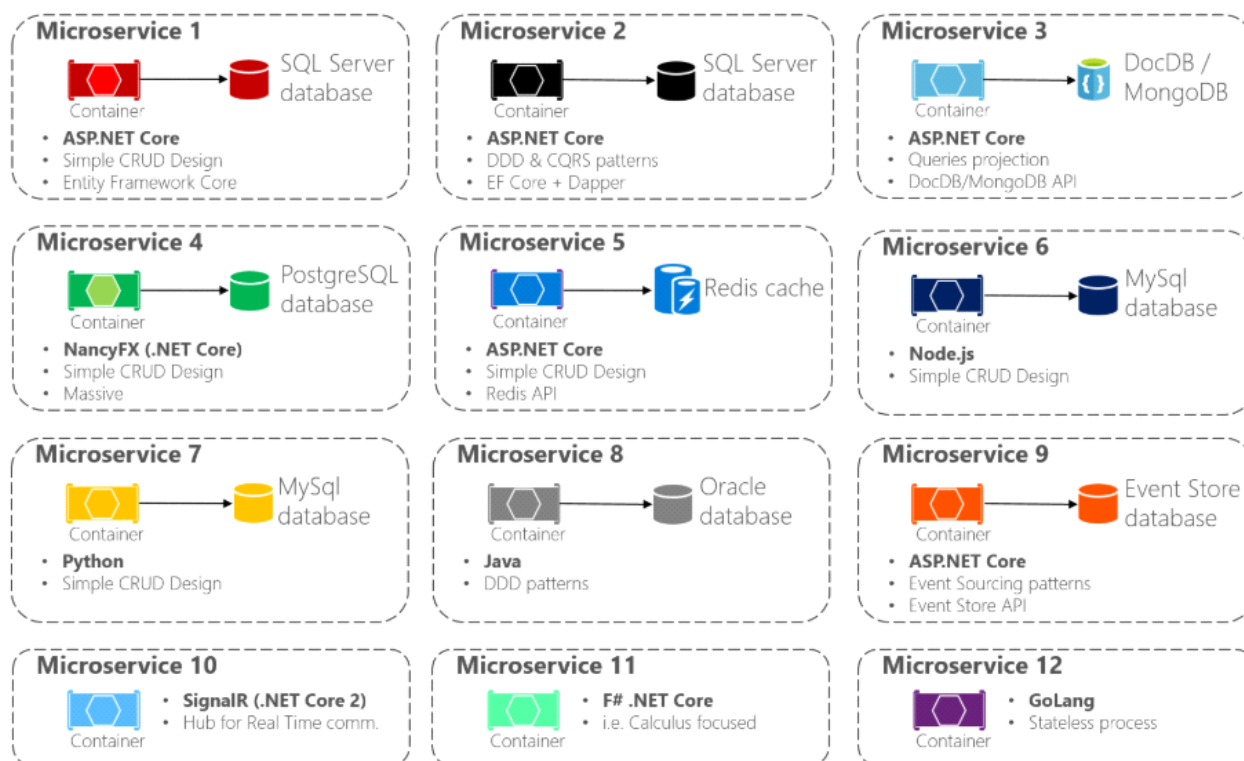


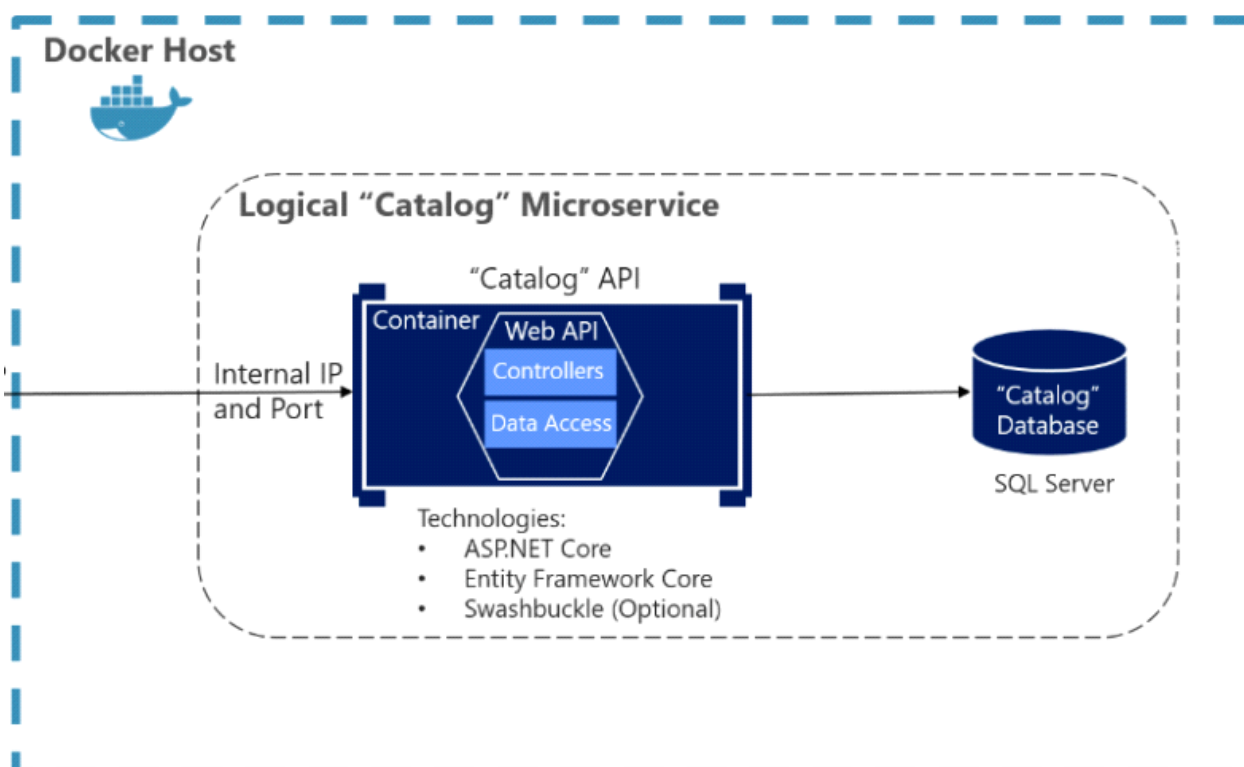
第25期-设计和开发多容器微服务

2020年4月28日 13:19

多个体系结构模式



设计简单的 CRUD 微服务

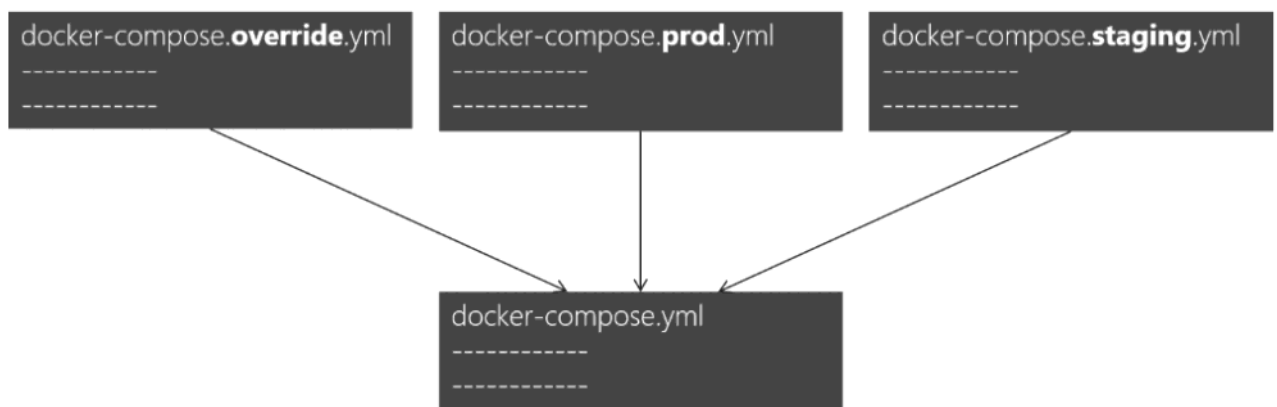


不建议将应用程序和数据库部署在同一台机器，不建议容器化数据库，无法发挥数据库高可用性。

Web API + Entity Framework Core + Swagger + API 版本控制

数据库连接字符串和环境变量从 Docker 环境中获取（生产或开发环境），在 docker-compose.yml 或 docker-compose.override.yml 文件中，可初始化这些环境变量。

定义多容器应用程序



命令：`docker-compose -f docker-compose.yml -f docker-compose.prod.yml up -d`

在容器中运行数据库

主流的数据库都支持在容器中运行，开发和测试环境来说，将数据库作为容器运行很方便，因为没有任何外部依赖项，随时可通过镜像构建数据库。

```
docker run -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=Pass@word' -p 5433:1433 -d  
mcr.microsoft.com/mssql/server:2017-latest
```

数据库迁移与演示数据生成

```
Log.Information("Applying migrations ({ApplicationContext})...", AppName);  
host.MigrateDbContext<CatalogContext>((context, services) =>  
{  
    var env = services.GetService<IWebHostEnvironment>();  
    var settings = services.GetService<IOptions<CatalogSettings>>();  
    var logger = services.GetService<ILogger<CatalogContextSeed>>();  
  
    new CatalogContextSeed()
```

```
        .SeedAsync(context, env, settings, logger)
        .Wait();
    })
    .MigrateDbContext<IntegrationEventLogContext>((_, __) => { });
```

使用内存数据库提高开发性能

```
services.AddDbContext<CatalogContext>(opt => opt.UseInMemoryDatabase());
```

不过，存在一个重要的问题。内存数据库不支持指定于特定数据库的许多约束。即便如此，内存数据库仍然可用于测试和原型设计。

使用在容器中运行的 Redis 缓存服务

```
docker run --name some-redis -d redis
```

使用 IHostedService 和 BackgroundService 类在微服务中实现后台任务

使用后台任务查询数据库表以查找具有特定状态的订单，并且在应用更改时，它会通过事件总线（其下可以使用 RabbitMQ 或 Azure 服务总线）发布集成事件。

