Microservices

Group 6

- Eric Han
- Mayuri Wadkar
- Sonali Mishra

Microservices

- Also known as Microservice architecture.
- Is an architectural style that structures an application as a collection of loosely coupled services, that implement business capabilities.
- The microservice architecture enables the continuous delivery/deployment of large, complex applications. It also enables an organization to evolve its technology stack.

Limitations of Monolithic Architectural style:

- Requires a long-term commitment to a technology stack
- Obstacle to scaling development
- Scaling the application can be difficult
- Continuous deployment is difficult
- Overloaded web container
- Overloaded IDE
- Large monolithic code base

Benefits of Microservices

- Each microservice is relatively small
  - Easier for a developer to understand
  - The IDE is faster making developers more productive
- Each service can be deployed independently of other services - easier to deploy new versions of services frequently
- Easier to scale development.
- Improved fault isolation.
- Each service can be developed and deployed independently.
- Eliminates any long-term commitment to a technology stack.

Drawbacks of Microservices

- Developers must deal with the additional complexity of creating a distributed system.
    - Developer tools/IDEs are oriented on building monolithic applications and don’t provide explicit support for developing distributed applications.
    - Testing is more difficult
    - Developers must implement the inter-service communication mechanism.
- Deployment complexity. In production, there is also the operational complexity of deploying and managing a system comprised of many different service types.
- Increased memory consumption.

Synchronous Model:

- Behaves almost like a monolith. Every service is dependent on another via RESTful calls, which are synchronous in nature.

Asynchronous Model

- Much better than synchronous. You don’t have microservices waiting for other microservices and so on down the chain to return a value to the end user.

Communication Methods in Microservice Architecture:

1. Apache Kafka- message passing
2. Events

- There are lots of different Inter-Process Communication technologies to choose from.
- Services can use synchronous request/response-based communication mechanisms such as HTTP-based REST or Thrift.
- Alternatively, they can use asynchronous, message-based communication mechanisms such as AMQP or STOMP.

Messaging Systems

- A message consists of headers (metadata such as the sender) and a message body.
- Messages are exchanged over channels.
- Any number of producers can send messages to a channel.
- Similarly, any number of consumers can receive messages from a channel.
- There are two kinds of channels:
  - Point-to-point
  - Publish-subscribe.

Messaging Systems - Point to Point

- Point-to-point channel delivers a message to exactly one of the consumers that is reading from the channel.
- Services use point-to-point channels for the one-to-one interaction styles described earlier.

Messaging Systems - Publish Subscribe

- A publish-subscribe channel delivers each message to all of the attached consumers.
- Services use publish-subscribe channels for the one-to-many interaction styles described above.

Messaging Systems - Advantages

- Decouples the client from the service.
- Message buffering.
- Flexible client-service interactions.
- Explicit inter-process communication.

Messaging Systems - Disadvantages

- Additional operational complexity.
- Complexity of implementing request/response-based interaction – Request/response - style interaction requires some work to implement.

Examples:

HTTP REST and Protobuf are used to communicate.