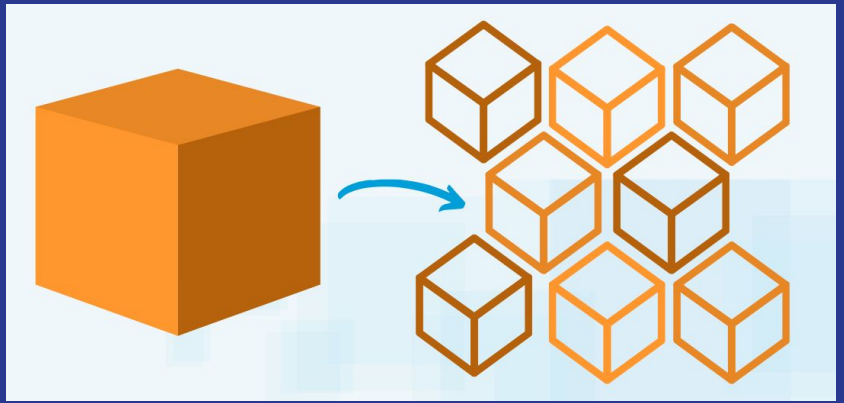
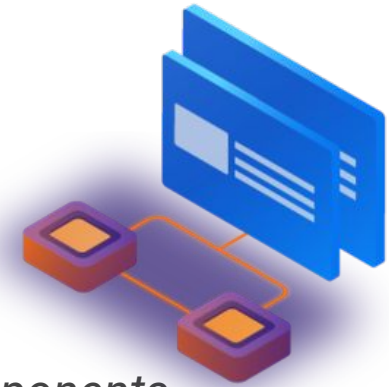
Microservice Architecture



By: Alejandro Zamora, Eduardo Todd, Jacob Torres, Jorge Felix, Matthew Montoya

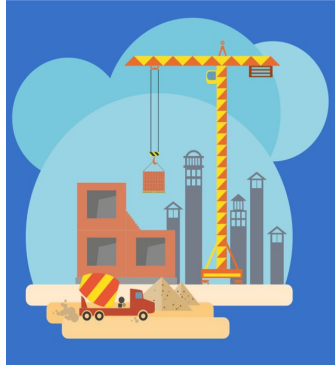
A Team 6 (Team404: Good Times, Inc.) Presentation.

Architecture Overview



- Modularized Architecture
  - Breaks up project into independently-replaceable *components*
    - Similar to replacing independent hardware components in a PC
  - Modularization removes the need for in-process calls (i.e. libraries)
    - Each component makes service calls to other remote processes
- Decentralized Data Storage
  - Each service has independent data storage
    - No direct data share
  - Sharing done through services

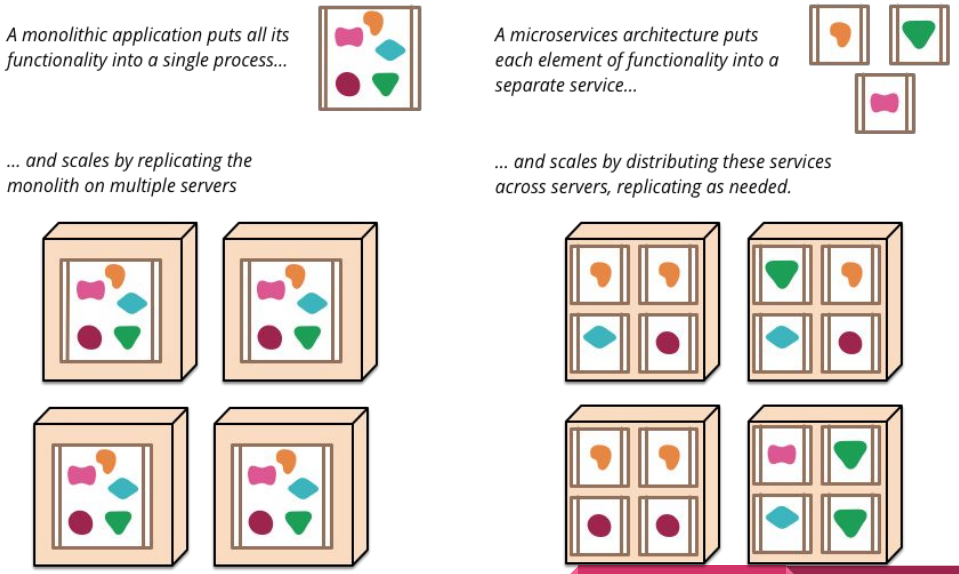
What problem does it solve?



- Scaling
  - In a monolithic application, an application that is built as a single logical executable, Scaling requires scaling the entire application rather than a single module.
  - Using Microservices, you can build your application as suites of services. So when you want to scale your application, you only have to worry about scaling the one module and not the entire application.

How does it solve the problem?

- Modularization
  - By taking components of functionality and putting them into separate services, you can achieve a Microservice Architecture.
  - These modules are independent of each other making them much easier to develop and deploy in your system.



Who are the Actors? (i.e. main elements)

- The services.
  - Services are small in size, messaging-enabled, bounded by contexts, autonomously developed, independently deployable, decentralized and built and released with automated processes. [1]
  - It's not "front-end" and "back-end"
  - A service is an independent piece of business functionality with well defined inputs and outputs

How do they relate to each other?

- Microservices interact using inter-process communication protocols
  - e.g. HTTP, AMQP, TCP, etc.
- Although microservices are very independent, they use their inputs and outputs to for a communication network, allowing the system to run successfully.

Architecture Advantages

- Independent upgrade of components
  - No need to upgrade the whole system
- Decentralized Data Management
  - Provides privacy pertaining to data sharing
  - Allows individual services to choose their own database and database architecture
- Faster Deployment
  - Smaller components

Microservice Architecture Disadvantages

- MicroServices are more complex
  - Requires more documentation which raises cost
  - Only necessary to use for big / monolithic projects
  - Hard to understand due to how big this architecture
- MicroServices are hard to test
  - Each microservice has to be individually tested.
  - Communication between services need to be individually verified.
- MicroServices require lots of resources
  - Processes can grow exponentially due to nature of protocols for services to talk to one another
  - Since most services are isolated which requires more CPU usage.

Real world Examples

- Netflix:
  - Was one of the earliest adopters of the Microservice Architecture.
  - Took their Monolith system and in the span of a few years split it into Microservices.
- Amazon:
  - Their components were highly coupled within the system and as they grew in size, Amazon couldn't deploy changes in a swift manner anymore
  - Started using microservices to clean up their codebase. But what started as cleanup, turned into an evolution for their system.
- Spotify
  - Wanting to be competitive in their market, Spotify started decoupling their system to build flexible structures that could be easily scalable, and less time consuming to develop.

Security Implications

- Microservices are usually isolated. The goal of this is to make sure that a microservice is deployed without impacting other microservices. This takes into account microservices communicating with one another.
- If any microservice impacts another microservice the effect cascades to other microservices.

Sources:

[1] Nadareishvili, I., Mitra, R., McLarty, M., Amundsen, M., Microservice Architecture: Aligning Principles, Practices, and Culture, O'Reilly 2016

[2] ThoughtWorks, "Martin Fowler – Microservices", Jan. 31, 2015. [Online]. Available: https://www.youtube.com/watch?v=2yko4TbC8cI&feature=emb_rel_pause. [Accessed Apr. 10, 2010].

[3] James Lewis, "Microservices: a definition of this new architectural term" Mar. 25, 2014. [Online]. Available: https://martinfowler.com/articles/microservices.html [Accessed Apr. 10, 2020]

[4] Tony Mauro, "Adopting Microservices at Netflix: Lessons for Architectural Design", Feb. 19, 2015. [Online]. Available: https://www.nginx.com/blog/microservices-at-netflix-architectural-best-practices/. [Accessed Apr. 10, 2010].

[5] Aleksandra Kwiecień, "10 companies that implemented the microservice architecture and paved the way for others", Feb. 8, 2019. [Online]. Available: https://divante.com/blog/10-companies-that-implemented-the-microservice-architecture-and-paved-the-way-for-others/ [Accessed Apr. 10, 2010].