Introduction to Scalable Applications and DevOps

Waheed Iqbal

DevOps (Fall 2023)
Department of Data Science, FCIT, University of the Punjab
Lahore, Pakistan

About this lecture!

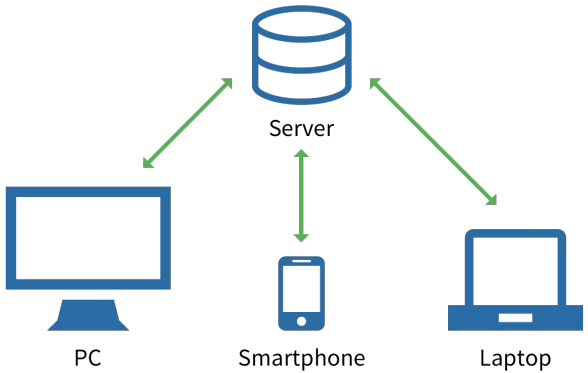
In this lecture, we are going to discuss:

- Different application architectures.
- Why microservices are gaining traction?
- Some examples of migrating from traditional application architecture to microservices.
- What are different patterns to deploy those applications for scalability?
- 12 factor app development methodology

Internet Applications

- An Internet application is a client/server application that uses standard Internet protocols for connecting the client to the server
- These can be Desktop, Web, or Mobile applications

Client-Server Model



Internet Applications: Examples

- Communication
- Job Search
- Online Shopping
- Web Browsing
- Stock Market Updates
- Travel
- Research
- E-Commerce
- Online Payments
- Social Networking
- E-banking
- Education
- Entertainment
- You may name more!

Application Architectures

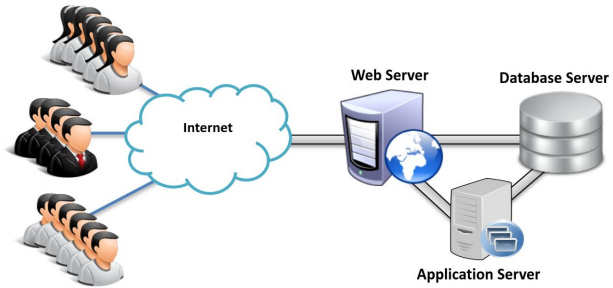
- Application architecture describes the patterns and techniques used to design and build an application.
- Different types of application architectures:
  1. Monolithic
  2. Multi-tier
  3. Microservices

Application Architectures: Monolithic

- It is a software design architecture in which all components of an application are composed as a single unit.
- This is tightly coupled, both in the interaction between the components and how they are developed and delivered.
- Updating or scaling a single aspect of a monolithic application has implications for the entire application and its underlying infrastructure.
- A single change to the application code requires the whole application to be re-released and redeployed.

Application Architectures: Multi-tier

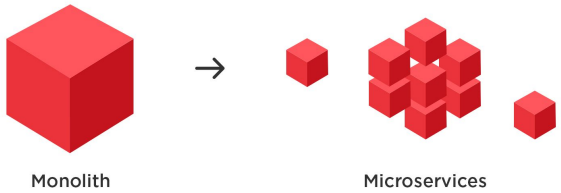
- Multi-tier application consists on multiple layers/tiers which can be deployed on different servers. It is also known as multi-layer architecture.
- Typically, multi-tier applications contain:
  - Web server
  - Database server
  - Application server
  - Batch job processors, etc.
- Scaling multi-tier application is challenging, and also have some issues like monolithic applications!



Application Architectures: Microservices

In Microservices architecture, applications are broken down into their smallest components, independent from each other.

- Microservices are so small and specific to one task
- They can easily be brought down, redeployed, and easily developed.
- The cost of error is small and allows for lots of experimentation and ultimately faster time to market
- Communication between microservices can be either synchronous or asynchronous



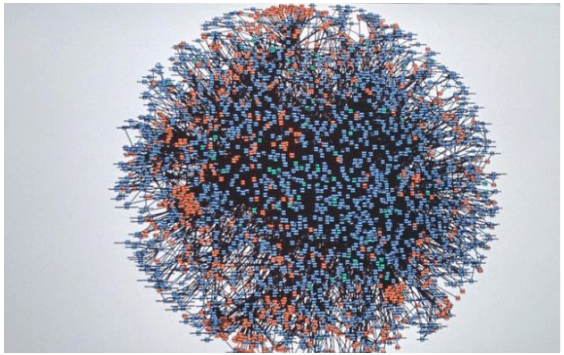
Let's discuss a simple blog application for different architectures!

Microservices Example: Amazon

- Amazon is an Internet retail giant!
- In the early 2000s, Amazon's retail website behaved like a single monolithic application.
- Developers had to carefully untangle dependencies every time they wanted to upgrade or scale Amazon's systems.
- Amazon found that the monolith structure worked very well. However, the code base quickly expanded as more developers joined the team.
- In 2001, development delays, coding challenges, and service interdependencies inhibited Amazon's ability to meet the scaling requirements of its rapidly growing customer base.
- Amazon broke its monolithic applications into small, independently running, service-specific applications.

Microservices Example: Amazon (Cont.)

- Developers analyzed the source code and pulled out units of code that served a single functional purpose
- Wrapped these units in a web service interface. For example, a single service for the Buy button on a product page, a single service for the tax calculator function, and so on. Each function had its own section.
- Amazon assigned ownership of each independent service to a team of developers
- Amazon's "service-oriented architecture" was largely the beginning of what we now call microservices.



2008 graphic of Amazon's microservices infrastructure

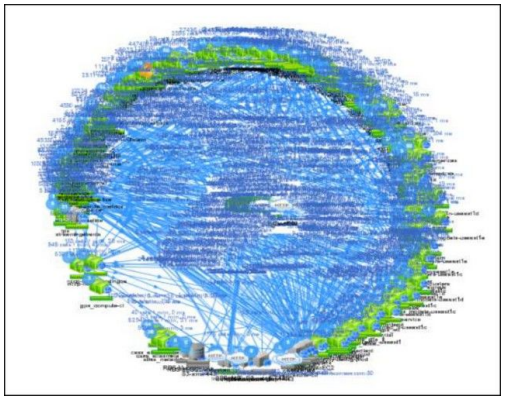
Microservices Example: Netflix

- Neflix founded in 1997 as video rental company.
- Netflix started its movie-streaming service in 2007.
- By 2008, it was suffering from service outages and scaling challenges; for three days!
- In 2019, Netflix, a video streaming app, consumed 15% of the Internet's bandwidth, all across the world.
- As of September 2022, Netflix had 222 million subscribers worldwide.

"Our journey to the cloud at Netflix began in August of 2008, when we experienced a major database corruption and for three days could not ship DVDs to our members. That is when we realized that we had to move away from vertically scaled single points of failure, like relational databases in our datacenter, towards highly reliable, horizontally scalable, distributed systems in the cloud. We chose Amazon Web Services (AWS) as our cloud provider because it provided us with the greatest scale and the broadest set of services and features." (*source*)

Microservices Example: Netflix (Cont.)

- In 2009, Netflix began the gradual process of refactoring its monolithic architecture, service by service, into microservices.
- The first step was to migrate its non-customer-facing, movie-coding platform to run on Amazon AWS cloud servers as an independent microservice.
- Netflix spent the following two years converting its customer-facing systems to microservices, finalizing the process in 2012.



Monolithic to Microservices Decomposition

- Mostly it is a manual engineering effort
- However, there are some efforts to use ML for decomposing monolithic application to microservices:
  - Unsupervised learning approach for web application auto-decomposition into microservices, Abdullah et al., Journal of Systems and Software, 2019.
  - Mono2Micro: IBM tool based on AI, 2020.

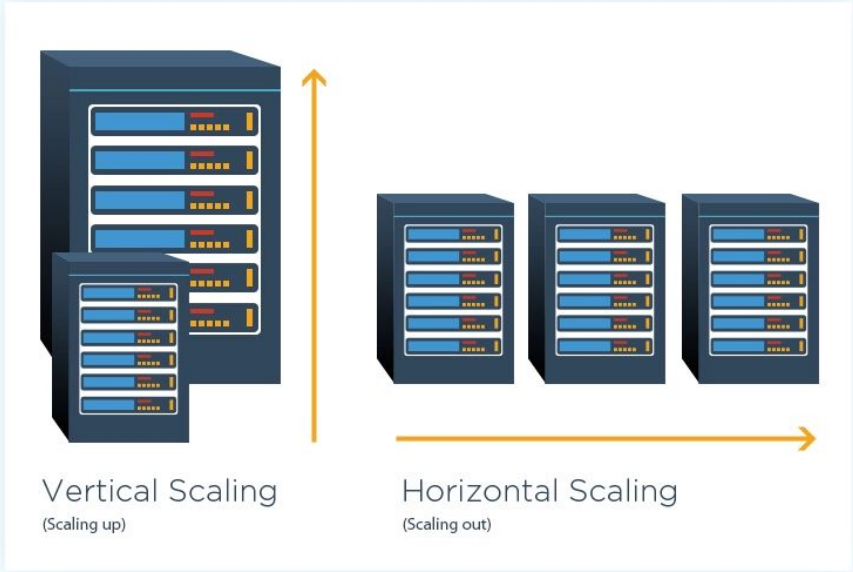
We will discuss more about those later!

Scalable Applications

Building scalable applications require skills to write optimized code and deployment on proper infrastructure.

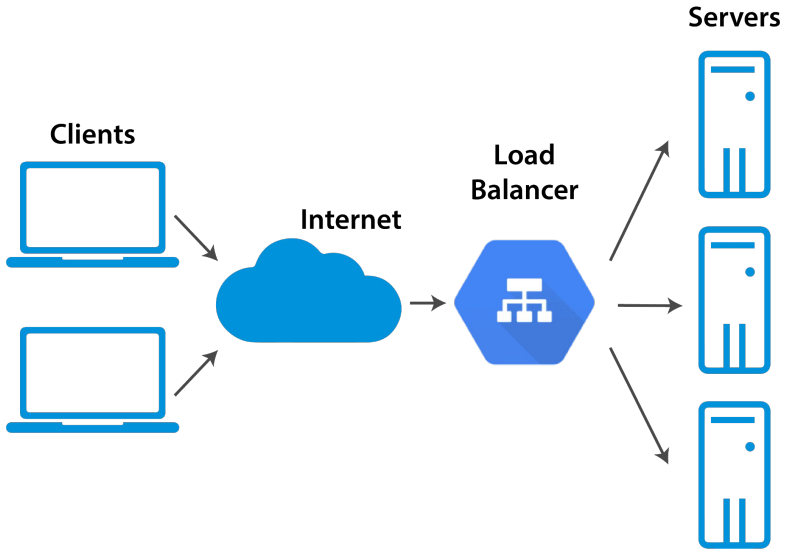
We can scale application infrastructure through:

- Vertical Scaling (Scaling up)
  - Increase the specs/capacity of the machines hosting application
- Horizontal Scaling (Scaling out)
  - Add application replicas to new machines
  - Requires load balancing



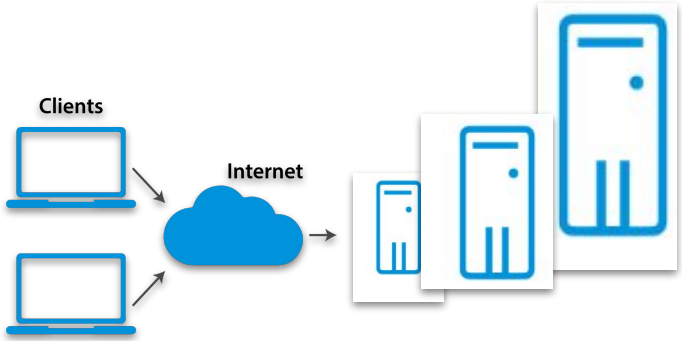
Load Balancing

- Load balancers distribute incoming traffic across your infrastructure to increase application's:
  - Availability
  - Performance
- Load balancers are available as hardware-based and software-based solutions.
- Load balancer use strategies like Round Robin, Fair-share, etc.

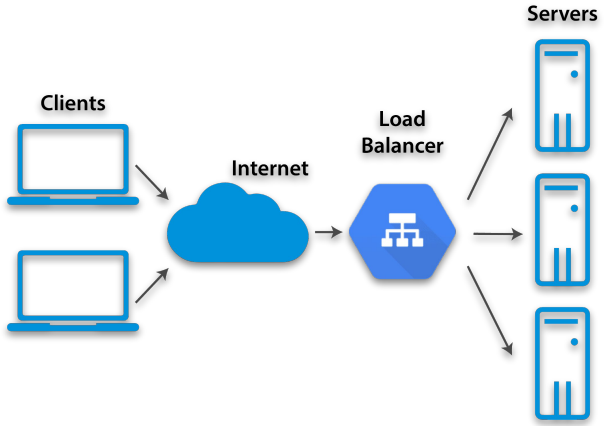


Scaling Monolithic Applications

Scaling monolithic application is simple, adding more replicas and serve the requests through load balancer or simply scale vertically.



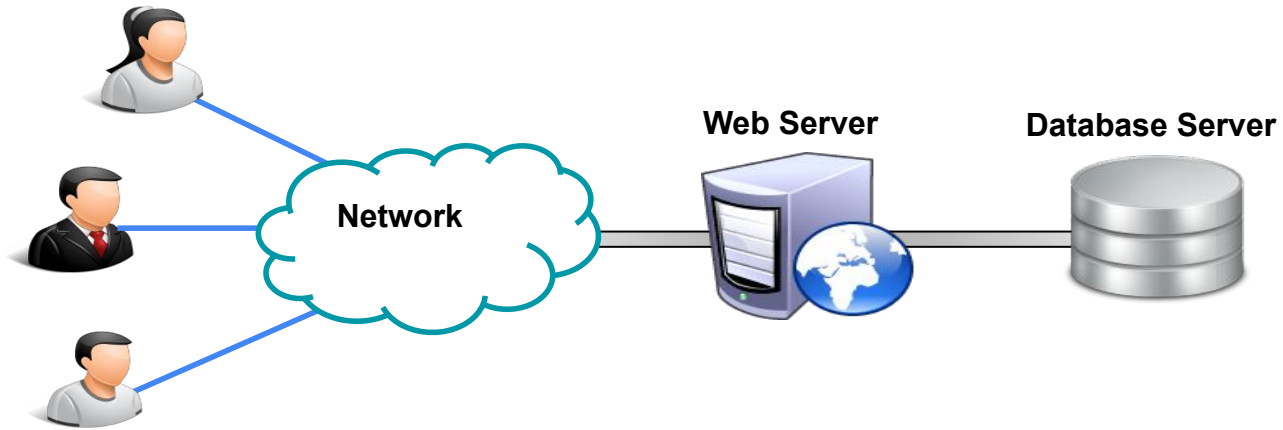
Vertical Scaling: Only one instance is active



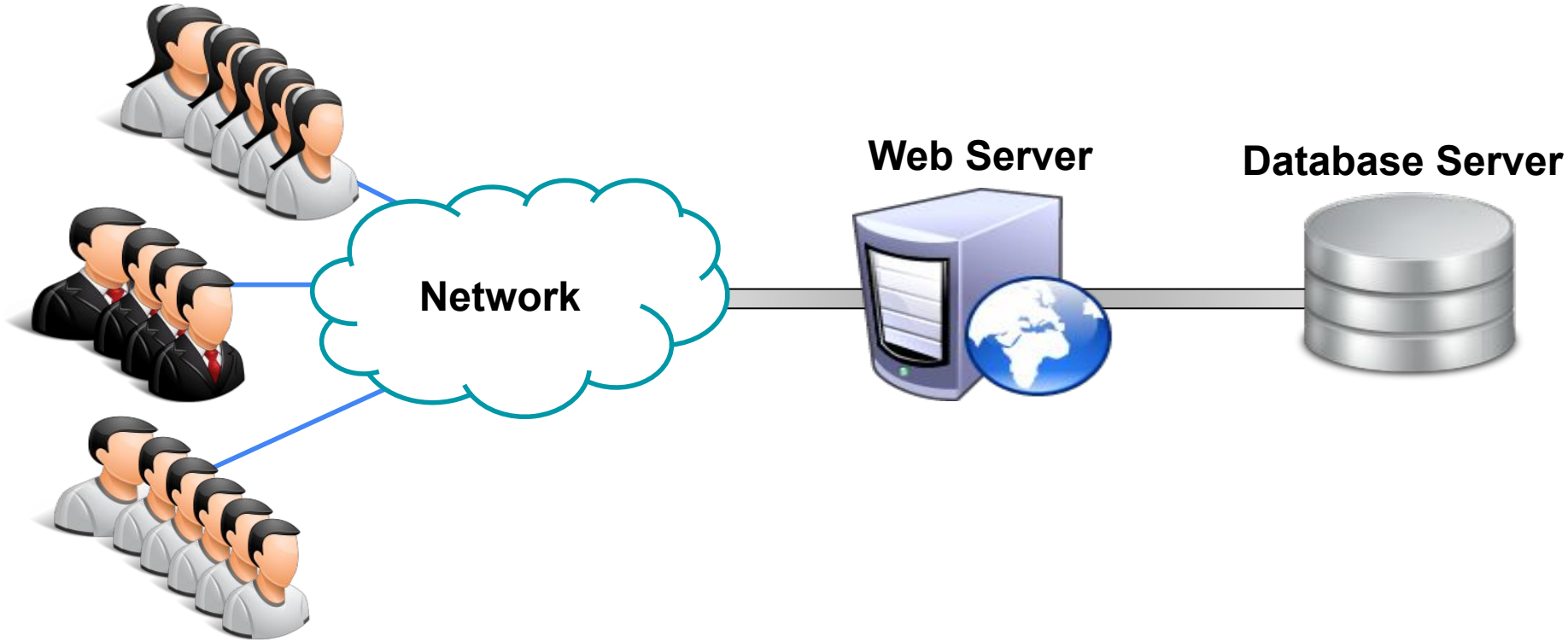
Horizontal Scaling: Multiple replica instances are active

Scaling Multi-tier Application

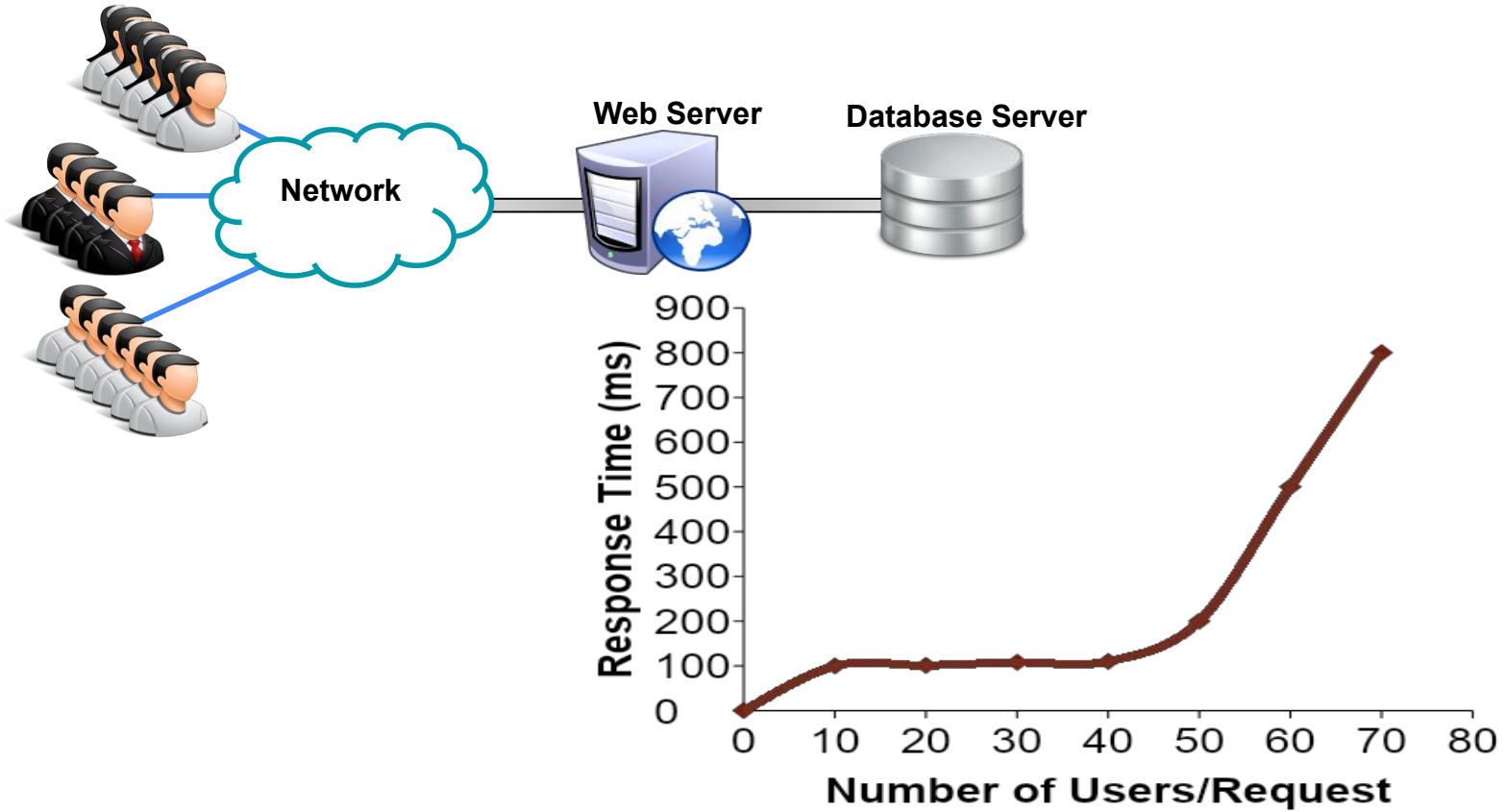
Let's consider a very simple multi-tier application for understanding scaling issues!



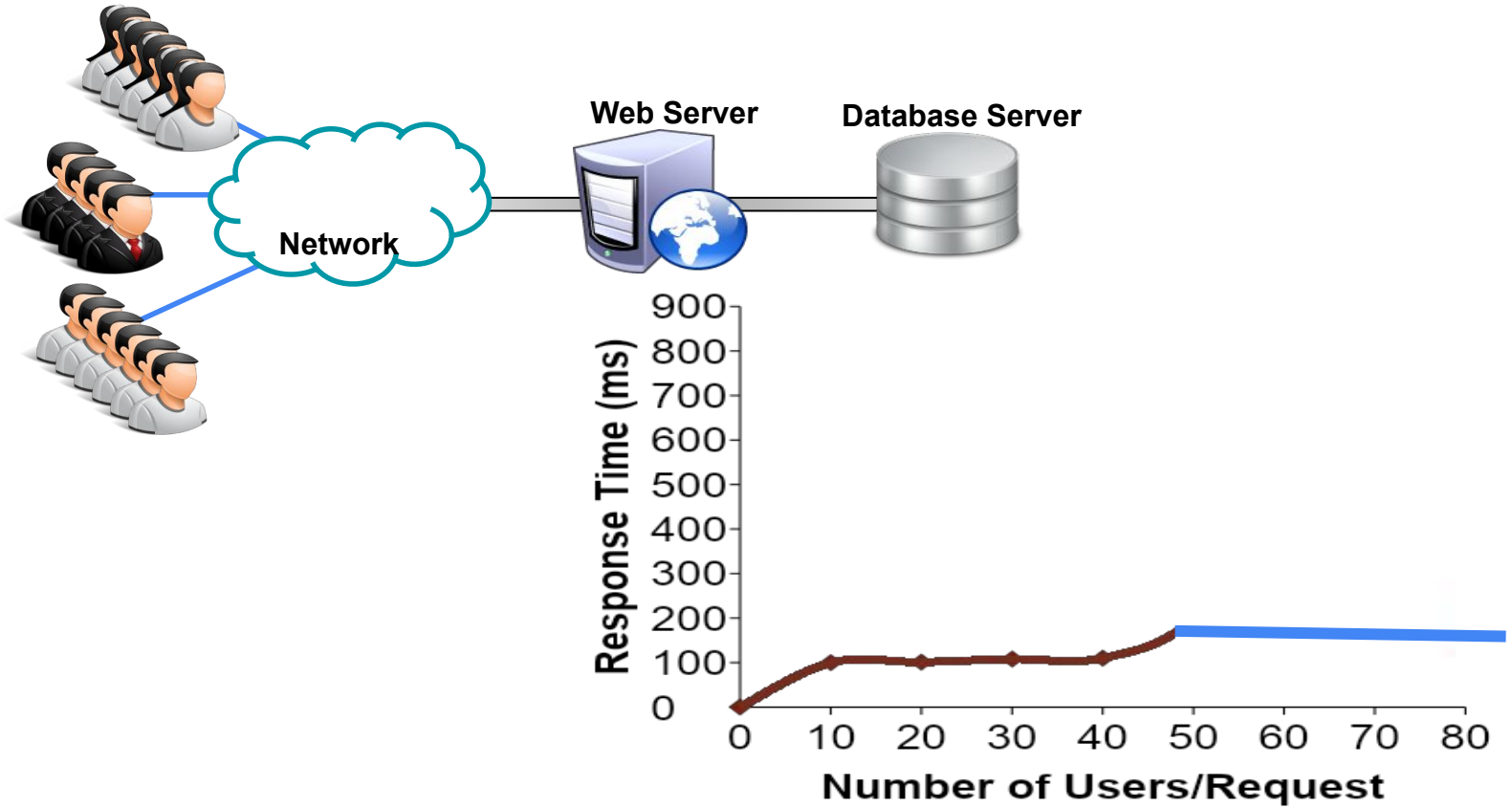
Scaling Multi-tier Application (Cont.)



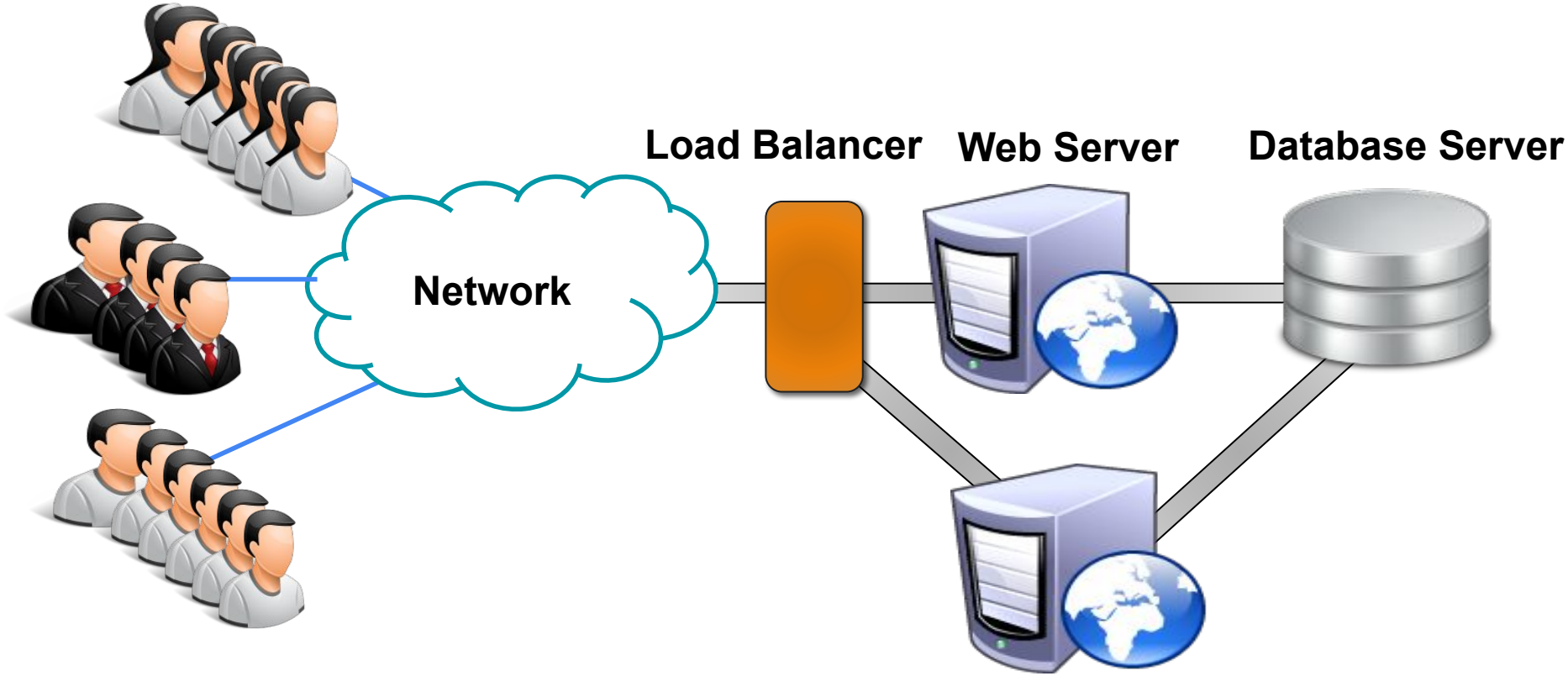
Scaling Multi-tier Application (Cont.)



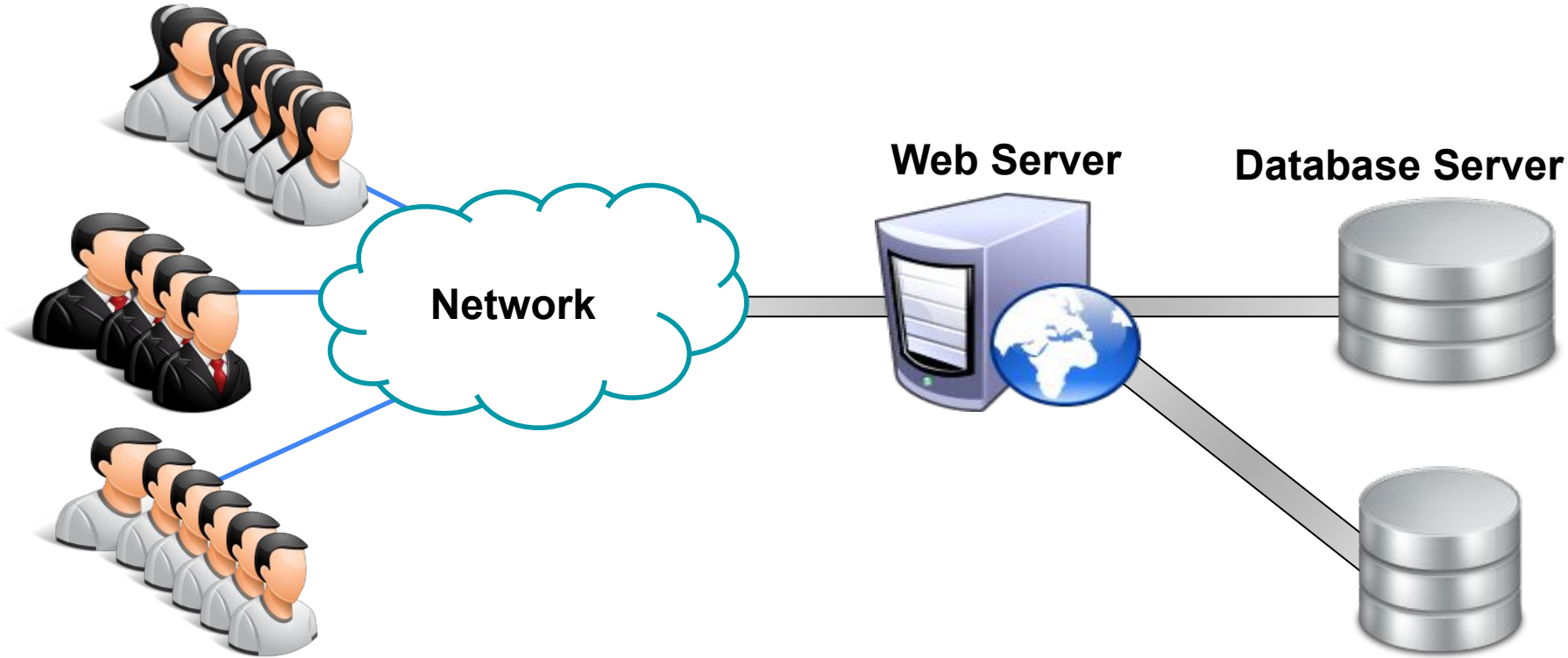
Scaling Multi-tier Application (Cont.)



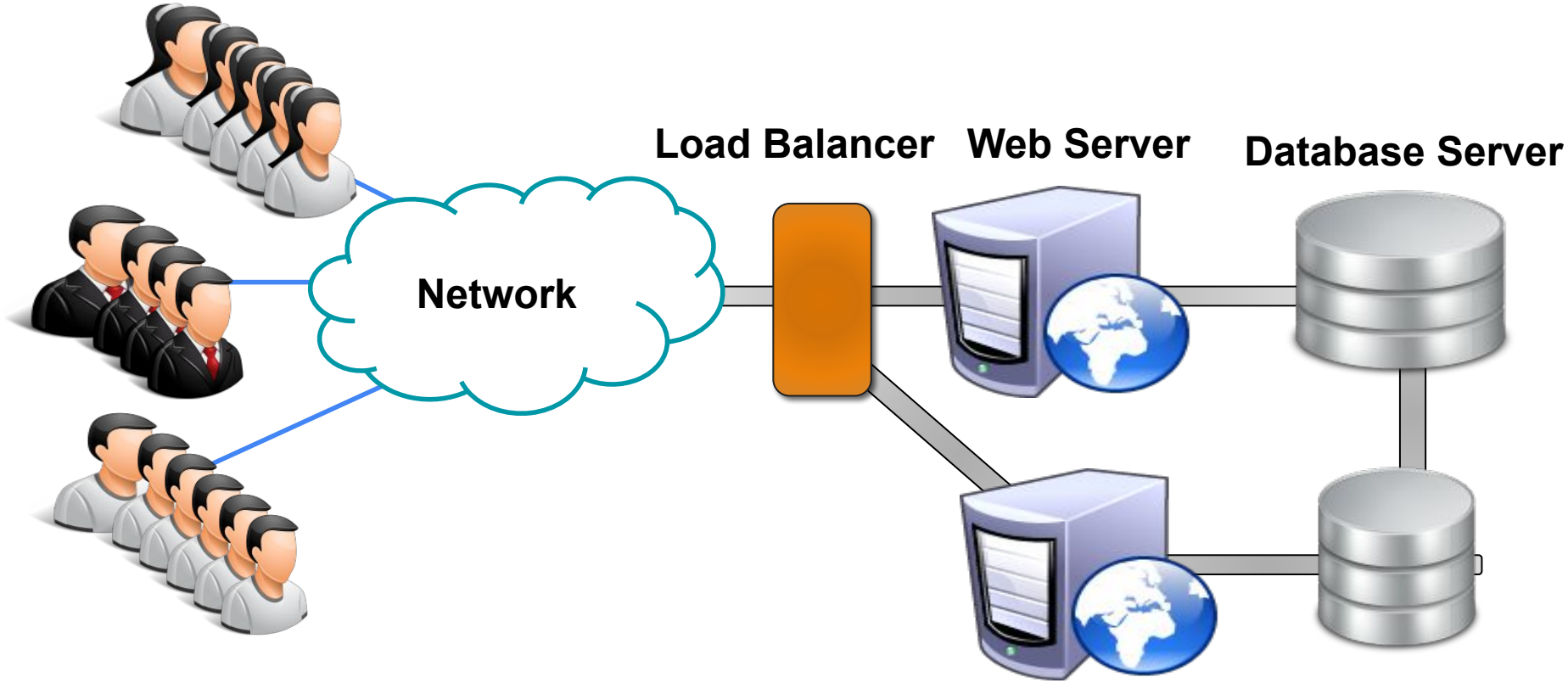
Scaling Multi-tier Application (Cont.)



Scaling Multi-tier Application (Cont.)



Scaling Multi-tier Application (Cont.)



Scaling Multi-tier Applications

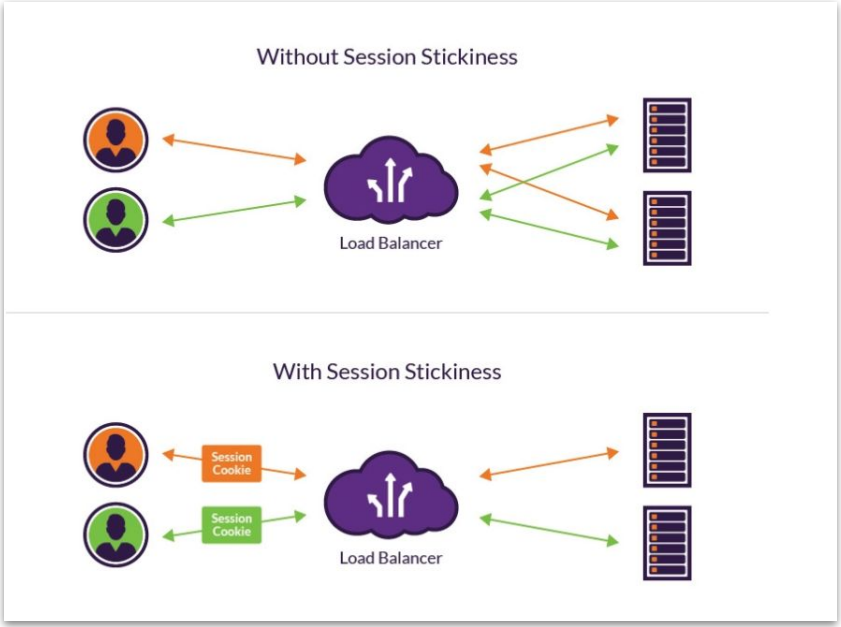
- When to Scale
  - Reactive: Based on metrics like CPU utilization, response time, bandwidth utilization etc
  - Proactive: Predict the possible performance issue may arises. For example fit a line on response time or arrival rate
- What to Scale
  - All tiers: Scale all tiers whenever there is a need of scaling
  - Predict appropriate tier: Identify the correct tier that is saturating and scale it
  - Policy-based or ML-based decisions: Make decision based on heuristics or using machine learning methods

We have done multiple contributions to scale multi-tier applications!

Session Management in Presence of Load Balancer

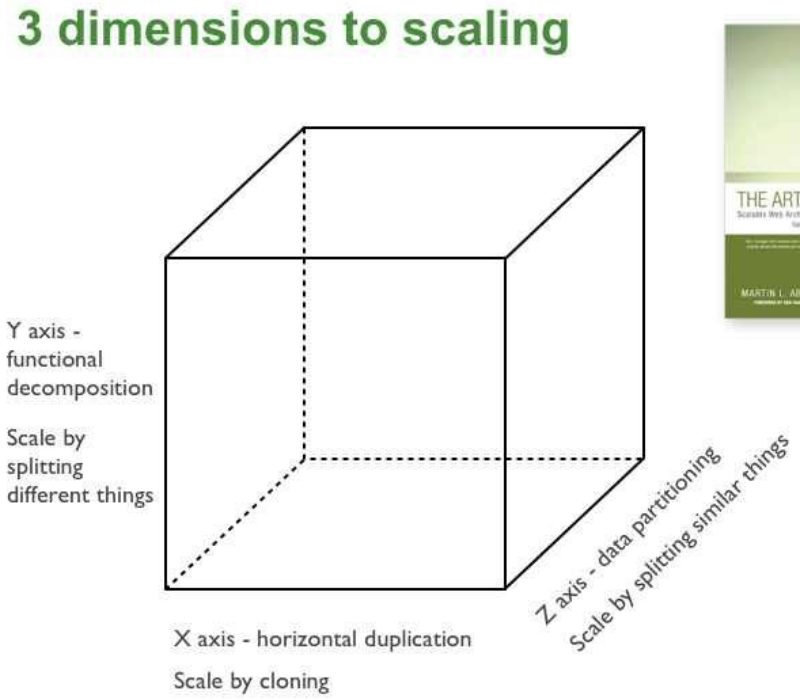
How to manage sessions in the presence of load balancer?

- Sticky Sessions (Session Stickiness)
- Session in Database



The Scale Cube

- X-axis is a typical scaling by increasing the replicas and serve requests through load balancer. (Horizontal scaling)
- Y-axis axis scaling splits the application into multiple, different services. Each service is responsible for one or more closely related functions. (Microservices)
- Z-axis is similar to X-axis scaling. The big difference is that each server is responsible for only a subset of the data.
    - Database partitioning
    - Consider a paid vs unpaid customer for better services, etc.

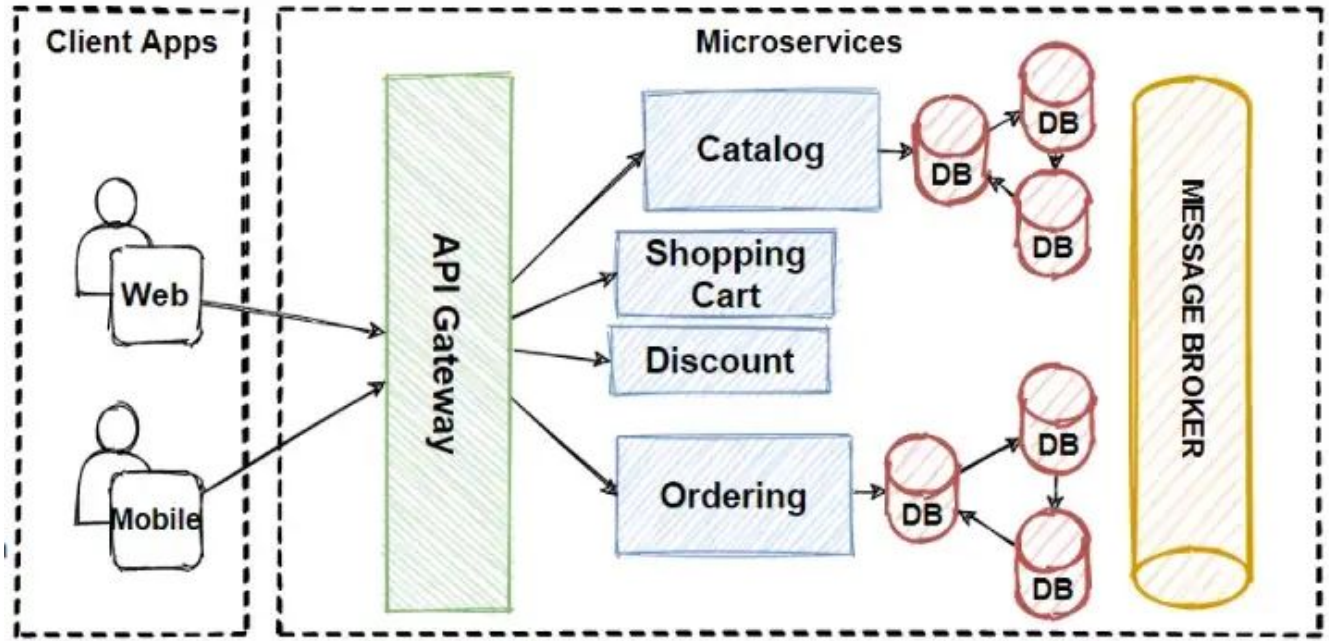


Scaling Microservices

For microservices, we can scale in all three axis.

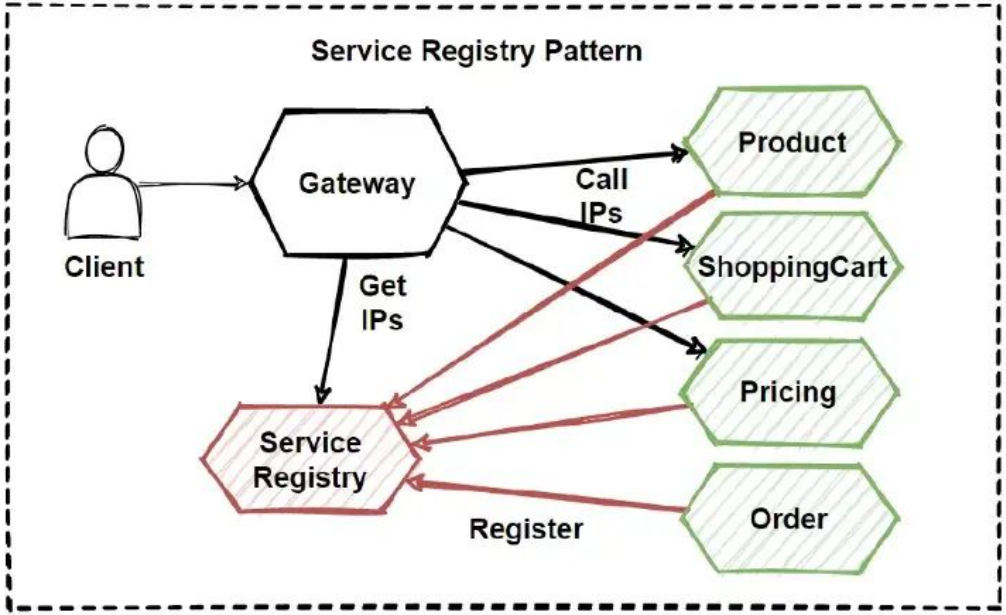
- Build small independent services (Y-axis)
- Enable scaling on each individual service (X-axis)
- Partition the database required for different services (Z-axis)

Example: E-commerce Microservices Scalability



Service Registry and Load Balancing for Microservices

1. Microservices register with the registry
2. Gateway keeps the updated endpoints by looking-up the registry
3. Gateway does load balancing and routing



12 Factor App Methodology

Best Practices for Building Scalable and Maintainable Applications

12-Factor App: Introduction

- The 12-factor app methodology is a set of best practices and principles for building modern, cloud-native applications that are scalable, maintainable, and easy to deploy.
- These principles were originally formulated by Heroku co-founder Adam Wiggins in 2011, and they have since become widely adopted in the world of cloud computing and application development.

12-Factor App: Introduction

1. Codebase
2. Dependencies
3. Backing Services
4. Config
5. Build, Release, Run
6. Processes
7. Port Binding
8. Concurrency
9. Disposability
10. Dev/Prod Parity
11. Logs
12. Admin Processes

Whats Next?

We will start learning Linux!

Credits

The slide uses material from the Internet. Some of the key reference are:

- https://www.redhat.com/en/topics/integration/what-is-event-driven-architecture
- https://www.redhat.com/en/topics/cloud-native-apps/what-is-an-application-architecture
- https://blog.dreamfactory.com/microservices-examples/
- https://www.techaheadcorp.com/blog/design-of-microservices-architecture-at-netflix/#:~:text=In%20the%20microservices%2Dbased%20architecture,has%20its%20own%20data%20encapsulation
- https://microservices.io/articles/scalecube.html
- https://dzone.com/articles/microservices-myths-and-misunderstanding