<u>Microservices</u>
<u>UNIT I (chapters 1 and 2)</u>

Chapter 1: Microservices

Q1) Balancing Speed and Safety:

- Sweden is a remarkably safer place for road users than the rest of the world.
- Are they better drivers? Are the traffic laws in Sweden stricter than other countries? Are their roads just better designed?
- It turns out that the recipe for traffic safety is a combination of all of these things, delivered by an innovative program called Vision Zero.
- Vision Zero has a laudable goal—reducing all road accident–related deaths to zero.
- It aims to achieve this by designing road systems that prioritize safety above all other factors, while still recognizing the importance of keeping traffic moving.
- In other words, a road system that is designed first and foremost with safety in mind.
- Policymakers, traffic system designers, and citizens have a shared belief that the safety of pedestrians and drivers is more valuable than the need to move from place to place as quickly as possible.
- In addition, the road system itself is designed to be safer.
- Traffic designers apply speed limits, road signs, and traffic movement patterns in a way that benefits the overall safety of the system.
- Vision Zero adopts a systematic approach to design in a safety-first manner.
- Just like traffic systems, software systems become more complex as their scale—in the form of scope, volume, and user interactions—increases.
- And like road designers, software architects and engineers must maintain a balance of speed and safety in their software systems.
- Software development organizations have used microservice architecture to achieve faster delivery and greater safety as the scale of their systems increase.

Q2) Understanding Microservices.

- Three concepts that are principal to the style.
- **1) Microservices are ideal for big systems:**
- The common theme among the problems that people were facing was related to size.
- Microservices style is designed to solve problems for systems that are big.
- But size is a relative measure, and it is difficult to quantify the difference between small, normal, and big.
- **2) Microservice architecture is goal-oriented:**
- Microservice architecture isn't about identifying a specific collection of practices, rather it's an acknowledgment that software professionals are trying to solve a similar goal using a particular approach.
- There may be a set of common characteristics that arise from this style of software development, but the focus is meant to be on solving the initial problem of systems that are too big.
- **3) Microservices are focused on replaceability:**
- This idea that driving toward replacement of components rather than maintaining existing components get to the very heart of what makes the microservices approach special.
- The methods that have helped companies improve changeability the most are primarily rooted in improving the replaceability of components.

Q3)Adopting Microservices:

- The microservice focus on building replaceable components.
- In particular, after learning more about microservices methods, potential adopters frequently identify the following issues:

- 1. They have already built a microservice architecture, but they didn't know it had a name.
- 2. The management, coordination, and control of a microservices system would be too difficult.
- 3. The microservices style doesn't account for their unique context, environment, and requirements.
- **"What are microservices? Don't I already have them?"**
- There is not one single definition for the term "microservice," there are two:
- 1. Microservices are small, autonomous services that work together.
- 2. Loosely coupled service-oriented architecture with bounded contexts.
- They both emphasize some level of independence, limited scope, and interoperability.
- A microservice is an independently deployable component of bounded scope that supports interoperability through message-based communication.
- Microservice architecture is a style of engineering highly automated, evolvable software systems made up of capability-aligned microservices.
- If you are considering adopting a microservice architecture for your organization, consider how effective the existing architecture is in terms of changeability and more specifically replaceability.
- Are their opportunities to improve?
- Ex: If you've implemented DevOps practices you've already invested in automated deployment.
-
- **"How could this work here?"**
- Microservice applications share some important characteristics:
- • Small in size
- • Messaging enabled
- • Bounded by contexts
- • Autonomously developed
- • Independently deployable
- • Decentralized
- • Built and released with automated processes
- The microservices stories we hear the most about are from companies that provide streamed content.
- While this is a domain with incredible pressure to remain resilient and perform at great scale, the business impact of an individual stream failing is simply incomparable to a hotel losing a reservation.
-
- **"How would we deal with all the parts? Who is in charge?"**
- Two microservices characteristics that you might find especially concerning are decentralization and autonomy.
- Decentralization means that the bulk of the work done within your system will no longer be managed and controlled by a central body.
- Embracing team autonomy means trusting your development teams to make their own decisions about the software they produce.
- The key benefit to both of these approaches is that software changes become both easier and faster—less centralization results in fewer bottlenecks and less resistance to change, while more autonomy means decisions can be made much quicker.
- In a microservice architecture, the services tend to get simpler, but the architecture tends to get more complex.
- Asserting control and management of a microservice system is more expensive than in other architectural styles.

Q4) The Microservices Way

- More specifically, the real value of microservices is realized when we focus on two key aspects—speed and safety.
- Finding an effective balance between them at scale is what we call the microservices way.
- Speed and Safety at Scale and in Harmony.
- **1. The Speed of Change:**
- The desire for speed is a desire for immediate change and ultimately a desire for adaptability.
- We could build software that is capable of changing itself.
- Shorten the time it takes for changes to move from individual workers to a production environment.
- After deliberate effort and careful quality control, our software was burned into a permanent state and delivered to users on tapes, CDs, DVDs, and diskettes.
- Of course, the popularity of the Web changed the nature of software delivery and the mechanics of releases have become much cheaper and easier.
- Ease of access combined with improved automation has drastically reduced the cost of a software change.
- **2. The Safety of Change:**
- Every change is potentially a breaking change and a system optimized purely for speed is only realistic if the cost of breaking the system is near zero.
- Most development environments are optimized for release speed, enabling the software developer to make multiple changes in as short a time as possible.
- On the other hand, most production environments are optimized for safety, restricting the rate of change to those releases that carry the minimum risk of damage.
- **3. At Scale:**
- To build at scale means to build software that can continue to work when demand grows beyond our initial expectations.
- Systems that can work at scale don't break when under pressure; instead they incorporate built-in mechanisms to increase capacity in a safe way.
- **4. In Harmony:**
- Your life is filled with decisions that impact speed and safety.
- How fast are you willing to drive a car to get where you need to be on time?
- How does that maximum speed change when there is someone else in the car with you?
- Is that number different if one of your passengers is a child?

Chapter 2: Microservices Value Proposition:

Q5) Microservice Architecture Benefits:

- We can scale our operation independently, maintain unparalleled system availability, and introduce new services quickly without the need for massive reconfiguration.
- Lessens dependencies between teams, resulting in faster code to production
- Allows lots of initiatives to run in parallel
- Supports multiple technologies/languages/frameworks
- Enables graceful degradation of service
- Promotes ease of innovation through disposable code—it is easy to fail and move on.
- With microservices, we can reduced the time it takes to deploy a useful piece of code and also reduced the frequency of deploying code that hasn't changed.
- We move quickly with good quality and flexible design.
- The goal of improving software delivery speed as functional scope grows is realized through greater agility (move quickly), higher composability, improved comprehensibility (understandable), independent service deployability etc.
- The goal of maintaining software system safety as scale increases is achieved through higher availability and resiliency, better efficiency, independent manageability and replaceability of components, increased runtime scalability, and more simplified testability.

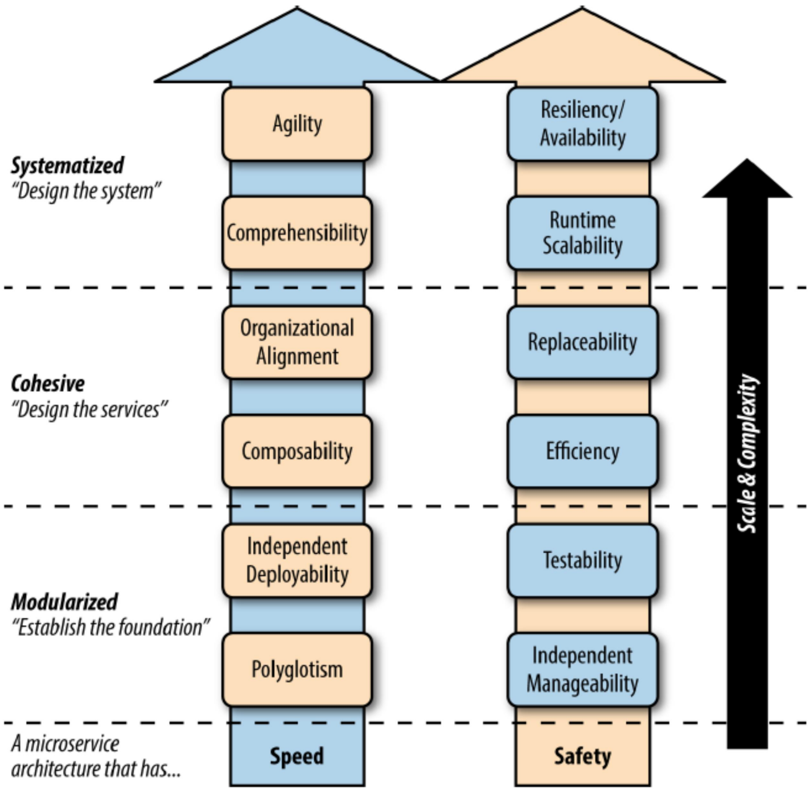
Q6) Deriving Business Value:

- Each of the microservice architecture benefits that drive delivery speed contribute real business value:
- • **Agility:** allows organizations to deliver new products, functions, and features more quickly and pivot more easily if needed.
- • **Composability**: reduces development time and provides a compound benefit through reusability over time.
- • **Comprehensibility**: of the software system simplifies development planning, increases accuracy, and allows new resources to come up to speed more quickly.
- • **Independent deployability of components**: gets new features into production more quickly and provides more flexible options for piloting and prototyping.
- • **Organizational alignment of services**: to teams reduces ramp-up (build up) time and encourages teams to build more complex products and features iteratively.
- • **Polyglotism**: permits the use of the right tools for the right task, thus accelerating technology introduction and increasing solution options.
- **The safety-aligned benefits also provide particular business value:**
- ● **Greater efficiency:** in the software system reduces infrastructure costs and reduces the risk of capacity-related service outages (not available).
- • **Independent manageability**: contributes to improved efficiency, and also reduces the need for scheduled downtime.
- • **Replaceability of components:** reduces the technical debt (short cuts) that can lead to unreliable environments.
- • **Stronger resilience and higher availability**: ensure a good customer experience.
- • **Better runtime scalability**: allows the software system to grow or shrink with the business.
- • **Improved testability** allows the business to mitigate (lessen) implementation risks.

Q7) Defining a Goal-Oriented Layered Approach:

- Set of layered characteristics to consider when adopting microservice architecture.
- **1. Modularized Microservice Architecture:**
- Modularity … is to a technological economy what the division of labor is to a manufacturing one.
- At its most basic level, microservice architecture is about breaking up an application or system into smaller parts.
- To help software delivery speed, modularized services are independently deployable.
- It is also possible to take a polyglot approach to tool and platform selection for individual services, regardless of what the service boundaries are.
- With respect to safety, services can be managed individually at this layer.
- Also, the abstracted service interfaces allow for more granular testing.
- **2. Cohesive Microservice Architecture:**
- The greater the cohesion of individual modules in the system, the lower the coupling between modules will be.
- In order to have a cohesive microservice architecture, it must already be modularized.
- A cohesive microservice architecture can enable software speed by aligning the system's services with the supporting organization's structure.
- service cohesion lessens the need for highly orchestrated message exchanges between components, thereby creating a more efficient system.
- **3. Systematized Microservice Architecture:**
- The key in making great and growable systems is much more to design how its modules communicate rather than what their internal properties and behaviors should be.
- The final and most advanced layer to consider in a microservice architecture is its system elements.

- After breaking the system into pieces through modularization, and addressing the services' contents through cohesion, it is time to examine the interrelationships between the services.
-
- **Maturity Model for Microservice Architecture Goals and Benefits:**



-

Q8)Applying the Goal-Oriented Layered Approach.

- To begin with, define the high-level business objectives you want to accomplish, and then weigh these against the dual goals of speed and safety.
- Within that context, consider the distinct benefits you are targeting.
- You can then use the maturity model to determine the complexity of the goal, and identify the best approach to achieve it.
- **An example of a goal-oriented approach in action(German digital media group):**
- Initial attempt at microservice architecture was on their monolithic service platform, which included functions such as user management and license management.
- The first attempt was explicitly focused on changing the architecture from monolith to service-enabled software system.
- The results were not positive.
- However, when they evaluated the main issues with the application—particularly the operational inefficiencies around it —they changed their approach from refactoring the existing architecture to automating the problematic deployment process.
- Through a small investment, they were able to take their service platform deployment downtime from 5 days to 30 minutes.
- Their next iteration will focus on reducing QA time through automation and a switch in methodology from white-box to black-box testing.
- Following these methodological changes, they will identify the domains in their monolithic application that require the greatest speed of innovation and unbundle those first.
- By taking an iterative approach tied to clear goals, they are able to measure success quickly and change course if needed.