**Introduction to Microservices**

**I. Principles of Microservices**

* Decentralization: Services are developed, deployed, and managed independently.
  + Instead of using Monolith where everything is implemented in a large system, can evolve in a big ball of mud. We use microservice to break down the business into smaller services. Each service will be isolated to domains closely aligned with business capabilities, it can develop and maintain by different team. Exp: we can break the Shopping App into some small services such as ProductService, ShoppingCartService, OrderService, UserService…
* Componentization: Services are treated as independent components that can be easily replaced and upgraded.
  + Since each service is managed by different team, it can be implemented by different technology which most suitable for the business.
  + In the microservice, services are loosely couple so if 1 service is down, it don’t affect the others.
* Autonomy: Teams work independently on each service, reducing the coordination overhead.
  + Each service is managed by different team, they’re loosely coupling and contact with each other via the Protocol. Therefore, Services don’t need to wait for other to be done before implementing.
* Technology Diversity: Teams can choose the best tool for their specific needs, fostering innovation.
  + Based on the business logic, the client traffic … Team can choose the best technology for it’s service. For example: ProductService will receive a lot of traffic from user searching, and don’t need Transaction. OrderService will need Transaction.

**II. Advantages of Microservices**

* Scalability: Services can be scaled independently, allowing for more efficient use of resources.
  + Due to the requirement and use case, we can replica the service to make sure it can handle enough use traffic. Exp : ProductService need handle a lot of traffic so we need to scale up it by replica more instances of this service. OrderService don’t have too much traffic. Contrast to Monolith when we have scale up both the whole system.
* Resilience: Faults in one service do not impact others, improving overall system robustness.
  + Each service run in an independent process. Therefore when 1 service down, the others still running. Microservice also have tool for us to call backup for a service when it’s down.
* Technological Agility: Allows the adoption of new technologies and processes without overhauling the entire system.
  + Services are treated as independent components that can be easily replaced and upgraded with the technology which good fit for business
  + Exp: ProductService can use mongodb for easy searching while OrderService should use mysql with Transaction for saving data.

**III. Challenges of Microservices**

* Complexity: Increased operational and management complexity.
  + Each service run on independent process and managed by different team, therefore we need person have big picture/devop to manage the whole system.
  + Each service can have different technology so each team when communication with each other should have common language.
* Data Integrity: Ensuring data consistency across services can be challenging.
  + Each service has it’s own database, when we save data which need involving some services we need to make sure all transactions are treated as one.
* Network Issues: Dependency on network latency and load balancing.
  + Each service communication with each other via Rest or Messaging, it will increase the request time. We need to apply some tool to monitor the latency between services to improve if need.
* Skill Set: Requires a broad set of skills from development teams, including DevOps capabilities.
  + Each service is managed by different team with different technology such as PHP, Java, C#, .Net…
  + We also need the DevOps team to configure, managed the Services, monitoring them for any issue. DevOps team also need to build the CI/CD tool for build, test and deploy the service.