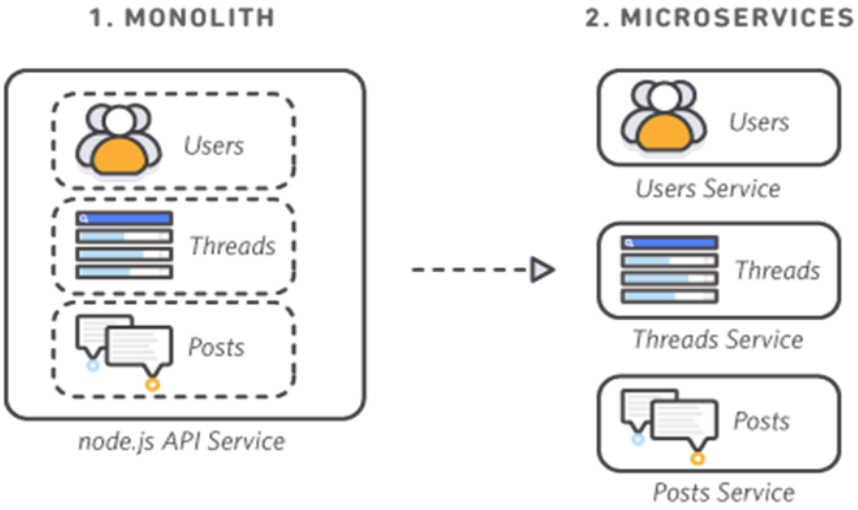
6) Monolith vs Microservice Architecture



Monolithic Architecture	Microservice Architecture
Advantages of Monolithic Architecture: 1. **Unified Codebase**: All modules such as the database, server-side, and client-side applications are built as a single code component. This simplicity makes it easy to develop and debug since all the necessary code is in one place. 2. **Ease of Development for Small Applications**: For relatively small applications, monolithic architecture is easier to develop and faster to get up and running. 3. **Simplified Deployment**: Deployment is easier with monolithic architectures as they require only a single Continuous Integration/Continuous Deployment (CI/CD) pipeline and infrastructure.	**Advantages of Microservice Architecture:** 1. **Distributed Services with Individual Business Logic**: Microservices architecture involves distributed services, each implementing specific business logic and features independently. 2. **Ease of Maintenance**: Each microservice is responsible for a specific task, making them easy to maintain and update without affecting other services. 3. **Isolation and Reliability**: Changes made to one microservice do not impact others, leading to a more reliable architecture. If a microservice fails, it can be updated or replaced without affecting the entire application.

4. **Ease of Local Deployment**: Since the entire application is contained within a single component, it is easier to run the application locally for testing and development purposes.

Disadvantages of Monolithic Architecture:

1. **Scaling Challenges**: Monolithic architectures are difficult to scale as the application grows. Scaling up requires scaling the entire application, leading to increased costs.
2. **Codebase Complexity Over Time**: As the application grows and new features are added, monolithic architectures can become complex and prone to "spaghetti code," making maintenance and updates more challenging.
3. **High Dependency**: Monolithic architectures do not easily allow for isolating features or business logic, leading to highly dependent code that can be hard to modify without affecting other parts of the application.
4. **Limited Flexibility**: Any change in code can have a cascading effect, requiring changes across the entire codebase, which can slow down development and testing processes.

4. **Technology Flexibility**: Different technology stacks can be used for specific purposes in each microservice, providing flexibility and optimization opportunities.
5. **Team Independence**: Microservices enable different teams to work independently on parts of the application, promoting faster development cycles and specialization.
6. **Scalability**: Microservices architecture allows for easy scaling of specific services, improving scalability and resource utilization.
7. **Continuous Deployment**: Changes can be tested and deployed independently for each microservice, facilitating continuous deployment practices.

Disadvantages of Microservice Architecture:

1. **Increased Complexity**: Managing communication between microservices, multiple CI/CD pipelines, and monitoring services can increase system complexity.
2. **Dependency Challenges**: Highly dependent microservices can make it harder to run and test the application locally.
3. **Monitoring and Debugging**: Due to the distributed nature, good monitoring practices are

5. **Single Server Dependency**: Monolithic applications typically run on a single server, which can be a limitation in terms of scalability and high availability requirements.	crucial for identifying and debugging issues in production environments. 4. **Communication Management**: Effective communication management between services is essential for the overall success of the architecture. 5. **System Design Challenges**: Building cohesive and well-defined microservices requires careful consideration of functional groups and domain-oriented design principles. 6. **Cloud Environment Advantages**: Microservices architecture can leverage the benefits of cloud environments for scalability, flexibility, and resource management.