Understanding Cloud-Native Architecture

Cloud-native architecture is a modern approach to designing applications that fully leverage cloud computing environments. Unlike traditional monolithic architectures, which bundle all application components into a single codebase, cloud-native applications are built as independent, scalable, and resilient microservices. These applications take advantage of containerization, continuous integration/continuous deployment (CI/CD), and cloud orchestration tools to ensure flexibility and efficiency.

Key Benefits:

- **Scalability:** Individual services can be scaled independently, optimizing resource usage and performance.
- **Resilience:** If one service fails, it doesn't bring down the entire system, improving reliability.
- **Flexibility:** Frequent updates and deployments can be made without affecting the entire system, enabling faster innovation.

Application Architecture for a Flight Booking System

Scenario Overview

You are part of a team designing a centralized flight booking system for multiple airlines, each with unique payment requirements:

- **Airline A:** Only allows PayPal payments.
- **Airline B:** No online payments (bookings must be done in person or via phone).
- **Airline C:** Supports PayPal and debit card payments.

Choosing the Right Architecture: Microservices vs. Monolith

A **microservices architecture** is the best fit for this project. Unlike a monolithic approach, where all functionalities are tightly coupled, microservices allow each component to operate independently. Given the distinct payment needs of each airline, a modular design ensures that changes can be implemented seamlessly without affecting the entire system.

Proposed System Design

1. **Frontend Service:** A web-based UI that interacts with backend services via APIs.
2. **Booking Service:** Manages flight searches, seat selection, and reservations.
3. **Customer Service:** Handles user accounts, preferences, and booking history.
4. **Payment Services:**
    - **Airline A Payment Service:** PayPal integration.
    - **Airline B Payment Service:** No payment gateway (redirects users to offline booking instructions).
    - **Airline C Payment Service:** Supports both PayPal and debit card transactions.
5. **API Gateway:** Manages communication between microservices, ensuring security and seamless interaction.

Trade-offs of a Microservices Approach

- **Advantages:** Greater scalability, independent deployment, resilience, and better alignment with airline-specific requirements.
- **Challenges:** Increased system complexity, potential latency in inter-service communication, and higher infrastructure costs.

Conclusion

A cloud-native, microservices-based approach is the most effective solution for this flight booking system. It provides the flexibility to accommodate diverse airline needs while ensuring scalability and resilience in a cloud environment.

References

- Fowler, M. (2014). *Microservices: A Definition of This New Architectural Term*. Retrieved from https://martinfowler.com/articles/microservices.html
- Kratzke, N. & Quint, P. C. (2017). *Understanding Cloud-Native Applications*. *Journal of Cloud Computing: Advances, Systems and Applications,* 6(1), 3-15. https://doi.org/10.1186/s13677-017-0080-0
- Villamizar, M., Garcés, O., Castro, H., Verano, M., Salamanca, L., Casallas, R., & Gil, S. (2017). *Evaluating the Monolithic and the Microservice Architecture Pattern to Deploy Web Applications in the Cloud*. *Proceedings of the 10th International Conference on Cloud Computing and Services Science*. https://doi.org/10.5220/0006388301870198