

PRODUCTIVITY

TECHNICAL
TIPS

10 MICROSERVICE PATTERNS EVERY DEVELOPER MUST KNOW!

@www.linkedin.com/in/pksingh90

Piyush Singh



INTRODUCTION

Microservices help in building scalable and flexible applications. But without the right design patterns, they can lead to complexity and failures.

Why Patterns Matter?

- Improve scalability & resilience
- Handle failures gracefully
- Optimize communication & data consistency
- In Simple words, It's makes developer's Life easy



PRODUCTIVITY

TECHNICAL
TIPS

ARCHITECTURE & DECOMPOSITION PATTERNS



① Service per Business Capability

📌 Each microservice should be designed around a specific business function.

✓ Keeps services modular and independent

✓ Easier to scale different parts of the system

💡 Example: An eCommerce system having separate services for Orders, Payments, and Inventory.



02 Strangler Fig Pattern

-  Gradually migrate from a monolith to microservices without downtime.
-  Incrementally replace monolith components
-  Reduces risk by keeping old & new systems running together
-  Example: Modernizing a legacy banking system by migrating services one by one.



PRODUCTIVITY

TECHNICAL
TIPS

COMMUNICATION PATTERNS



03 API Gateway

📌 A single entry point for all clients, handling authentication, routing, rate limiting and request transformation.

✓ Improves security and performance

✓ Reduces client-side complexity

💡 Example: Netflix API Gateway routes requests to multiple microservices.



④ Event-Driven Architecture

📌 Services communicate asynchronously using events instead of direct API calls.

✓ Reduces dependencies between services

✓ Improves scalability and fault tolerance

💡 Example: Order Service publishes an "Order Placed" event, Payment Service listens and processes it.



PRODUCTIVITY

TECHNICAL
TIPS

DATA MANAGEMENT PATTERNS



⑤ Database per Service

- 📌 Each microservice has its own database to ensure independence.
- ✓ Prevents data conflicts and scaling issues
- ✓ Avoids single points of failure
- 💡 Example: A banking app having separate databases for transactions, customer profiles, and reports.



⑥ Saga Pattern

- 📌 Manages distributed transactions in microservices by using compensating actions.
- ✓ Ensures data consistency without using distributed locks
- ✓ Two types: Choreography (events) & Orchestration (central controller)
- 💡 Example: If a hotel booking fails, cancel the flight reservation if customer booked package.



PRODUCTIVITY

TECHNICAL
TIPS

DEPLOYMENT & SCALING PATTERNS



⑦ Circuit Breaker

📌 Prevents cascading failures by stopping calls to failing services.

✓ Improves system stability

✓ Detects failures early and retries when the service is healthy

💡 Example: If the Payment Service fails, the system returns a fallback response instead of crashing.



⑧ Blue-Green Deployment

📌 Use two environments (Blue & Green) to ensure zero-downtime deployment.

✓ Instantly roll back if something goes wrong

✓ Ensures seamless updates for users

💡 Example: Deploying a new version of a mobile app backend without disrupting users.



PRODUCTIVITY

TECHNICAL
TIPS

OBSERVABILITY & SECURITY PATTERNS



⑨ Centralized Logging & Monitoring

📌 Collect logs, metrics, and traces from all microservices in one place.

✓ Debug and troubleshoot faster

✓ Helps in performance tuning

💡 Example: Using ELK Stack (Elasticsearch, Logstash, Kibana) for centralized logging.



⑩ JWT Authentication & OAuth 2.0

📌 Secure APIs using token-based authentication and authorization.

✓ Ensures secure communication between services

✓ Supports role-based access control

💡 Example: User logs in, gets a JWT token, and uses it to access multiple microservices securely.



CONCLUSION

- ✓ Using the right patterns makes microservices scalable, resilient, and secure.
 - ✓ Choose patterns based on your system needs.
- … Which pattern do you use the most? Drop a comment!
- 📌 Save this for future reference!





REPOST

AND

FOLLOW US TO GET
MORE INFORMATION AND
TIPS LIKE THIS.

