**مقایسه معماری نرم‌افزاری: Monolithic vs Microservices**

**۱. تعریف و تفاوت ساختاری**

**معماری مونولیتیک (Monolithic)**

* کل سیستم به صورت یک واحد اجرا می‌شود.
* اجزا شامل رابط کاربری، منطق کسب‌وکار، دسترسی به داده‌ها و ماژول‌ها در یک کدبیس مشترک هستند.
* تعامل بین اجزا داخل همان فرآیند است و معمولاً از یک دیتابیس مشترک استفاده می‌شود.

**مزایا:**

* ساده‌تر در طراحی اولیه.
* مدیریت وابستگی‌ها و مسیر اجرا راحت‌تر است.
* استقرار (Deployment) ساده: فقط یک برنامه را مستقر می‌کنیم.
* تست end-to-end ساده‌تر.

**معایب:**

* با رشد سیستم، افزودن قابلیت جدید دشوار می‌شود.
* مقیاس‌پذیری جزئی محدود است.
* تغییر فناوری یا بازنویسی بخشی از سیستم سخت است.
* زمان استقرار و آپدیت کل سیستم ممکن است طولانی باشد.

**معماری میکروسرویس (Microservices)**

* سیستم به مجموعه‌ای از سرویس‌های مستقل تقسیم می‌شود، که هر سرویس مسئول یک قابلیت خاص است.
* هر سرویس ممکن است دیتابیس یا منابع مخصوص خود را داشته باشد.
* سرویس‌ها از طریق شبکه با هم ارتباط برقرار می‌کنند.

**مزایا:**

* استقلال بین سرویس‌ها.
* امکان استفاده از فناوری‌های متفاوت در هر سرویس.
* مقیاس‌پذیری دقیق‌تر: فقط سرویس‌های پر بار مقیاس می‌شوند.
* تیم‌ها می‌توانند به سرویس‌های جداگانه اختصاص یابند.

**معایب:**

* پیچیدگی ارتباطات شبکه‌ای بین سرویس‌ها.
* نیاز به زیرساخت DevOps قوی.
* مدیریت تراکنش‌ها و سازگاری داده‌ها پیچیده‌تر است.
* استقرار و هماهنگی بین سرویس‌ها دشوارتر است.

**احراز هویت و امنیت**

**مونولیتیک**

* یک نقطه مرکزی برای احراز هویت و مجوزها وجود دارد.
* Session-based Authentication یا JWT معمول است.
* کنترل دسترسی راحت‌تر است.
* نقطه ضعف: در صورت نفوذ، کل سیستم در خطر است.

**میکروسرویس**

* یک سرویس مرکزی (Auth Service / Identity Provider) وظیفه احراز هویت را دارد.
* احراز هویت بین سرویس‌ها از طریق JWT، Mutual TLS یا API Gateway انجام می‌شود.
* کنترل مجوز به صورت fine-grained در هر سرویس اعمال می‌شود. (کنترل دقیق و سطح پایین، به صورت خیلی مشخص روی اجزا یا عملیات کوچک)
* نیاز به مانیتورینگ و لاگینگ متمرکز دارد.
* در صورت نفوذ در یک سرویس، امکان ایزوله کردن آن وجود دارد.

**مقایسه قابلیت توسعه‌پذیری (Scalability)**

|  |  |  |
| --- | --- | --- |
| **مونولیتیک** | **میکروسرویس** | **مورد** |
| عمودی (افزایش منابع سرور) | افقی دقیق‌تر (فقط سرویس‌های پر بار) | نوع مقیاس‌پذیری |
| سخت | راحت | مقیاس بخش خاص |
| کمتر بهینه | بهینه‌تر، مصرف منابع بهینه‌تر | منابع بهینه |
| ساده | نیاز به orchestration، load balancing، auto-scaling | نیاز زیرساخت |

**هزینه‌ها:**

**مونولیتیک**

**مزایا:**

* هزینه اولیه کمتر.
* نگهداری ابتدایی ساده‌تر.
* استقرار و ابزارهای CI/CD ساده.

**معایب:**

* هزینه نگهداری بلندمدت بالا.
* مقیاس‌پذیری محدود.
* زمان downtime بیشتر برای به‌روزرسانی کل سیستم.

**میکروسرویس**

**مزایا:**

* بهینه‌سازی منابع در بلندمدت.
* توسعه موازی تیمی.
* امکان به‌روزرسانی سرویس‌های کوچک بدون توقف کل سیستم.

**معایب:**

* هزینه اولیه و زیرساختی بیشتر.
* پیچیدگی عملیاتی و مدیریت داده توزیع‌شده.
* نیاز به DevOps قوی و مانیتورینگ پیشرفته.

|  |  |  |
| --- | --- | --- |
| **مونولیتیک** | **میکروسرویس** | **موضوع** |
| ساده‌تر در شروع | پیچیدگی توزیع‌شده از ابتدا | پیچیدگی |
| تست یکپارچه آسان | تست واحد + تعامل بین سرویس‌ها + تست end-to-end | تست |
| کل برنامه یکجا مستقر می‌شود | هر سرویس مستقل مستقر می‌شود | استقرار |
| ساده‌تر | نیاز به Distributed Tracing و logging متمرکز | خطایابی |
| coupling بالا، تغییر یک بخش روی کل سیستم اثر می‌گذارد | coupling کمتر، سرویس‌ها قرارداد API دارند | وابستگی‌ها |
| سخت | آسان، سرویس‌های جدید می‌توانند فناوری متفاوت داشته باشند | تغییر فناوری |
| همگام‌سازی سخت | تیم‌ها مستقل روی سرویس خود کار می‌کنند | توسعه تیمی |
| خطا یک بخش، کل سیستم ممکن است متوقف شود | خطا یک سرویس، سایر سرویس‌ها ادامه می‌دهند | مقاومت |
| کم | بالاتر به دلیل تماس‌های شبکه‌ای بین سرویس‌ها | تاخیر |

**. جمع‌ بندی و توصیه‌ها**

* **مونولیتیک:** مناسب پروژه‌های کوچک، MVP، تیم‌های کوچک، راه‌اندازی سریع.
* **میکروسرویس:** مناسب سیستم‌های بزرگ، نیاز به مقیاس‌پذیری دقیق، توسعه موازی تیم‌ها، انعطاف در فناوری.
* **توصیه عملی:**
  + پروژه کوچک → شروع با مونولیتیک ماژولار.
  + پروژه بزرگ یا در حال رشد → میکروسرویس یا مهاجرت تدریجی از مونولیتیک.
  + رعایت اصول امنیت، پایش، مدیریت نسخه و طراحی سرویس‌ها در میکروسرویس ضروری است.

به صورت کلی بستگی به نیاز پروژه دارد.