Spring Cloud

Milad Barzideh

MB

فهرست مطالب

فصل ۱

میکروسرویس‌ها، ابر و چیزهای دیگر

میکروسرویس یا ریزخدمت[۱] یکی از اصطلاحاتی‌ست که این روزها در دنیای نرم‌افزار رایج شده است. میکروسرویس‌ها، سرویس‌هایی توزیع‌شده و مجزا هستند که هر کدام از آن‌ها وظیفه‌ی کوچک و مشخصی را انجام می‌دهند. در این نوشتار، پس از معرفی میکرسرویس‌ها و ذکر مشخصه‌های آن، با استفاده از فریمورک‌های Spring Boot و Spring Cloud به صورت عملی، معماری میکروسرویس‌ها را بررسی خواهیم کرد.

۱.۱ میکروسرویس چیست؟

قبل از مطرح شدن میکروسرویس‌ها بیشتر برنامه‌های تحت‌وب با استفاده از معماری یکپارچه[۲] ساخته می‌شدند. در معماری یکپارچه، هر برنامه به عنوان یک محصول نرم‌افزاری قابل اجرا در نظر گرفته می‌شود و رابط کابری، منطق برنامه و دسترسی به پایگاه داده، همه در یک برنامه‌ی کاربردی قرار گرفته می‌شود.

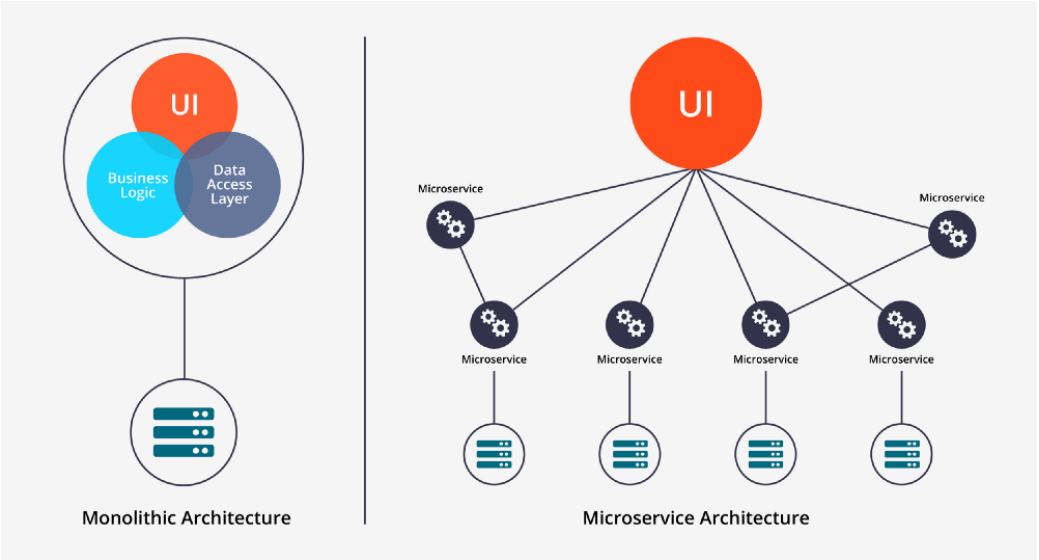
هر برنامه به عنوان یک واحد کاری در نظر گرفته می‌شود که معمولا چندین تیم توسعه روی بخش‌های مختلف آن کار می‌کنند. یکی از مشکلاتی که در این حالت پیش می‌آید این است که با افزایش اندازه و پیچیدگی برنامه، هزینه‌های ارتباطی و هماهنگی تیم‌ها، بیشتر و سخت‌تر می‌شود. برای مثال با هر تغییری که تیم‌ها ایجاد می‌کنند، کل برنامه دوباره باید تست و مستقر[۳] شود.

مفهوم میکروسرویس تقریبا از سال ۲۰۱۴ و در پاسخ به چالش‌هایی که برای مقیاس‌پذیر کردن برنامه‌های یکپارچه وجود داشت، وارد جامعه‌ی نرم‌افزار شد. همان‌طور که گفته شد یک میکروسرویس، کوچک، مجزا و توزیع‌شده است. بنابراین این امکان را فراهم می‌کند که برنامه‌های بزرگ را به بخش‌های کوچکتری تقسیم کرد تا مدیریت و توسعه‌ی آن‌ها آسان‌تر داده شوند. مهم‌ترین نکته‌ای که باید به آن توجه داشته باشید، تجزیه و جداسازی عملکردهای برنامه است به صورتی که کاملا مستقل از همدیگر باشند.

همان‌طور که شکل ۱.۱ نشان می‌دهد هر تیم می‌تواند کدها و زیرساخت مربوط به سرویس خود را داشته باشد و آن‌ها را به صورت مستقل نسبت به سرویس‌های دیگر توسعه و ارزیابی کند.

---

[۱] Microservice
[۲] Monolithic
[۳] Redeployed



شکل ۱.۱: معماری یکپارچه در مقابل معماری میکروسرویس

معماری میکروسرویس ویژگی‌های زیر را دارد:

* منطق برنامه به قسمت‌های کوچک‌تر شکسته می‌شود که هر قسمت مسئله‌ی مشخص و تعریف شده‌ای را حل می‌کند.

* هر قسمت محدوده‌ی مسئولیت مشخصی دارد و مستقل از سایر بخش‌ها توسعه و استقرار می‌یابد. همچنین میکروسرویس‌ها باید قابلیت استفاده‌ی مجدد در برنامه‌های دیگر را داشته باشند.

* میکروسرویس‌ها با استفاده از قواعد مشخص و بکارگیری پروتکل‌های ارتباطی سبک‌وزن مانند HTTP و JSON داده‌ها را مبادله می‌کنند.

* پیاده‌سازی فنی هر میکروسرویس بی‌اهمیت است؛ زیرا برنامه‌ها با پروتکل‌های مستقل از تکنولوژی[۴] با هم ارتباط برقرار می‌کنند. این یعنی برنامه‌ای که با معماری میکروسرویس ساخته شده می‌تواند از تکنولوژی‌ها و زبان‌های مختلفی استفاده کند.

* میکروسرویس‌ها برای سازمان‌ها این امکان را فراهم می‌کنند که تیم‌های توسعه‌ی کوچک با وظایف کاری مشخص داشته باشند.

[۴] Technology-neutral protocol

۲.۱ محاسبات ابری چیست؟

۳.۱ چرا ابر؟

به عنوان یک توسعه‌دهنده‌ی میکروسرویس، دیر یا زود باید تصمیم بگیرید سرویس‌تان را در یکی از جاهای زیر مستقر کنید:

* سرور فیزیکی: سازمان‌های کمی این کار را انجام می‌دهند؛ زیرا سرورهای فیزیکی محدودیت ایجاد می‌کنند. برای مثال شما نمی‌توانید ظرفیت سرور فیزیکی را سریعا افزایش دهید یا میکروسرویس را به دلیل هزینه‌هایی زیادی که در پی دارد روی چندین سرور فیزیکی مستقر کنید.
* ماشین مجازی: یکی از مهم‌ترین فواید میکروسرویس‌ها، توانایی آن‌ها در افزودن یا کاهش نمونه‌های یک سرویس است (مقیاس‌پذیری). یک میکروسرویس را می‌توان در یک ماشین مجازی قرار داد و به آسانی چندین نمونه از آن را روی زیرساخت‌های موجود مستقر کرد.
* ظرف مجازی[۵]: به جای استقرار میکروسرویس روی یک ماشین مجازی کامل، بسیاری از توسعه‌دهندگان سرویس‌های خود را به صورت ظرف‌های داکر[۶] در محیط ابری مستقر می‌کنند.

تفاوت داکر با ماشین مجازی

مزیت مهم میکروسرویس‌های تحت ابر انعطاف‌پذیری بالای آن‌هاست. ارائه‌دهندگان محیط ابری اجازه می‌دهند در کم‌تر از چند دقیقه یک ماشین مجازی یا یک ظرف جدید را اضافه یا حذف کنید. همچنین برنامه‌ها مقاوم‌تر هستند، برای مثال اگر یکی از میکروسرویس‌ها از کار بیفتد می‌توان از نمونه‌های دیگر آن میکروسرویس استفاده کرد و برنامه را بالا نگه داشت.

[۵] Virtual container
[۶] Docker

فصل ۲

کنترل پیکربندی‌ها

به عنوان یک توسعه‌دهنده می‌دانید که hard-code کردن مقادیر در کدهای برنامه چندان منطقی نیست، به همین دلیل معمولا توسعه‌دهندگان از یک فایل ثابت استفاده می‌کنند و همه اطلاعات پیکربندی برنامه را در آن قرار می‌دهند. با این حال قرار دادن این اطلاعات در کدهای برنامه مشکل‌ساز است، زیرا با هر تغییر در پیکربندی، کل برنامه دوباره باید کامپایل شود. برای جلوگیری از این کار، توسعه‌دهندگان اطلاعات مربوط به پیکربندی را از کدهای برنامه جدا می‌کنند.

این مشکل در برنامه‌های تحت ابر بیشتر خود را نشان می‌دهد، از آن‌جا که ممکن است این برنامه‌ها از صدها میکروسرویس با چندین نمونه‌ی در حال اجرا تشکیل شده باشند، مدیریت پیکربندی برنامه بسیار مشکل است. برای همین توسعه‌ی میکروسرویس‌های تحت ابر روی موارد زیر تاکید دارد:

۱. جداسازی پیکربندی برنامه از کدهای اصلی برنامه که باید مستقر شوند.

۲. تزریق اطلاعات مربوط به پیکربندی برنامه در زمان راه‌اندازی سرور با استفاده از یک مخزن مرکزی به میکروسرویس‌های برنامه.

۱.۲ معماری مدیریت پیکربندی

۲.۲ روش‌های پیاده‌سازی