AWSPanduan Preskriptif

Membusuk monolit menjadi layanan mikro

AWSPanduan Preskriptif: Membusuk monolit menjadi layanan mikro

Table of Contents

Membusuk monolit menjadi layanan mikro

Hari Ohm Prasath Rajagopal, Tabby Ward, dan Dmitry Gulin, Amazon Web Services (AWS)

Januari 2021(Riwayat dokumen (p. 11))

Migrasi ke Amazon Web Services (AWSCloud memilikibanyak keuntungan, termasuk kelincahan teknis dan bisnis, peluang pendapatan baru, dan pengurangan biaya. Untuk sepenuhnya mendapatkan keuntungan dari keuntungan ini, Anda harus terus memodernisasi perangkat lunak organisasi Anda dengan refactoring aplikasi monolitik Anda menjadi microservices. Proses ini terdiri dari tiga langkah utama:

- Terurai monolit menjadi layanan mikro- Gunakan pola dekomposisi yang disediakan oleh panduan ini untuk memecah aplikasi monolitik menjadi layanan mikro.
- Integrasikan layanan mikro- Integrasikan layanan mikro yang baru dibuat menjadiarsitektur mikrodengan menggunakanAWSlayanan irserver.
- Aktifkan persistensi data untuk layanan mikro- Promosikanketekunan poliglotdi antara layanan mikro Anda dengan mendesentralisasi penyimpanan data.

Salah satu langkah pertama dalam perjalanan modernisasi aplikasi Anda adalah menguraikan monolit dalam portofolio Anda menjadi layanan mikro. Sebagian besar aplikasi dimulai sebagai monolit yang mewakili kasus penggunaan bisnis tertentu, dan jika arsitektur monolit tidak menegakkan desain modular, monolit dapat tetap menjadi pilihan yang valid untuk aplikasi yang tidak memiliki pengetahuan domain yang mapan. Karakteristik sentral monolit sebagai satu unit penyebaran juga dapat membantu mengurangi kekurangan desain, seperti kopling ketat atau kurangnya struktur internal.

Meskipun monolit dapat menjadi pilihan yang valid untuk beberapa kasus penggunaan, biasanya tidak cocok untuk aplikasi modern. Struktur internal monolit yang didefinisikan dengan buruk dapat menyulitkan pemeliharaan kode, yang menciptakan kurva belajar yang curam untuk pengembang baru dan menyebabkan biaya dukungan tambahan. Kopling tinggi dan kohesi rendah dapat secara signifikan meningkatkan waktu yang diperlukan untuk menambahkan fitur baru, dan Anda mungkin tidak dapat menskalakan komponen individual berdasarkan pola lalu lintas. Monolit juga membutuhkan banyak tim untuk berkoordinasi untuk satu rilis besar, yang meningkatkan kolaborasi dan beban transfer pengetahuan. Akhirnya, Anda dapat menemukan bahwa menambahkan fitur baru atau membangun pengalaman pengguna baru menjadi sulit ketika bisnis Anda tumbuh atau jumlah pengguna meningkat.

Untuk menghindari hal ini, Anda dapat menggunakan pola dekomposisi untuk memecah aplikasi monolitik, mengubahnya menjadi beberapa layanan mikro, dan memigrasinya ke arsitektur layanan mikro. Sebelum memulai proses dekomposisi, Anda harus mengevaluasi monolit mana yang akan terurai, dan pastikan untuk menyertakan yang memiliki masalah keandalan atau kinerja, atau yang menyertakan beberapa komponen dalam arsitektur yang digabungkan erat. Kami juga menyarankan agar Anda memahami sepenuhnya kasus penggunaan bisnis monolit, teknologi, dan saling ketergantungannya dengan aplikasi lain.

Panduan ini ditujukan untuk pemilik aplikasi, pemilik bisnis, arsitek, prospek teknis, dan manajer proyek. Ini membahas empat pola cloud-native berikut yang digunakan untuk menguraikan monolit, dan menjelaskan kelebihan dan kekurangan masing-masing:

- Terurai oleh kemampuan bisnis (p. 3)
- Terurai oleh subdomain (p. 4)
- Terurai oleh transaksi (p. 5)

- Layanan per pola tim (p. 7)

Panduan ini merupakan bagian dari rangkaian konten yang mencakup pendekatan modernisasi aplikasi yang direkomendasikan oleh AWS. Serial ini juga mencakup:

- Strategi untuk memodernisasi aplikasi diAWSCloud
- Pendekatan bertahap untuk memodernisasi aplikasi diAWSCloud
- Mengevaluasi kesiapan modernisasi untuk aplikasi diAWSCloud
- Mengintegrasikan layanan mikro dengan menggunakanAWSlayanan irserver
- Mengaktifkan persistensi data untuk layanan mikro

Hasil bisnis yang ditargetkan

Anda harus mengharapkan hasil berikut setelah Anda menguraikan monolit Anda menjadi layanan mikro:

- Transisi yang efisien dari aplikasi monolitik Anda ke dalam arsitektur layanan mikro.
- Penyesuaian cepat terhadap permintaan bisnis yang berfluktuasi tetapi tanpa mengganggu aktivitas inti, seperti skalabilitas tinggi, peningkatan ketahanan, pengiriman berkelanjutan, dan isolasi kegagalan.
- Inovasi lebih cepat karena setiap layanan mikro dapat diuji dan digunakan secara individual.

Pola untuk membusuk monolit

Setelah Anda memutuskan untuk menguraikan monolit dalam portofolio aplikasi Anda, Anda harus memilih pola dekomposisi yang sesuai dan mengenalkannya ke dalam organisasi Anda.

> Note
>
> Anda dapat menggunakan beberapa pola untuk menguraikan monolit. Misalnya, Anda dapat menguraikan monolit dengankemampuan bisnis (p. 3)dan kemudian menggunakanpola subdomain (p. 4)untuk memecahnya lebih.

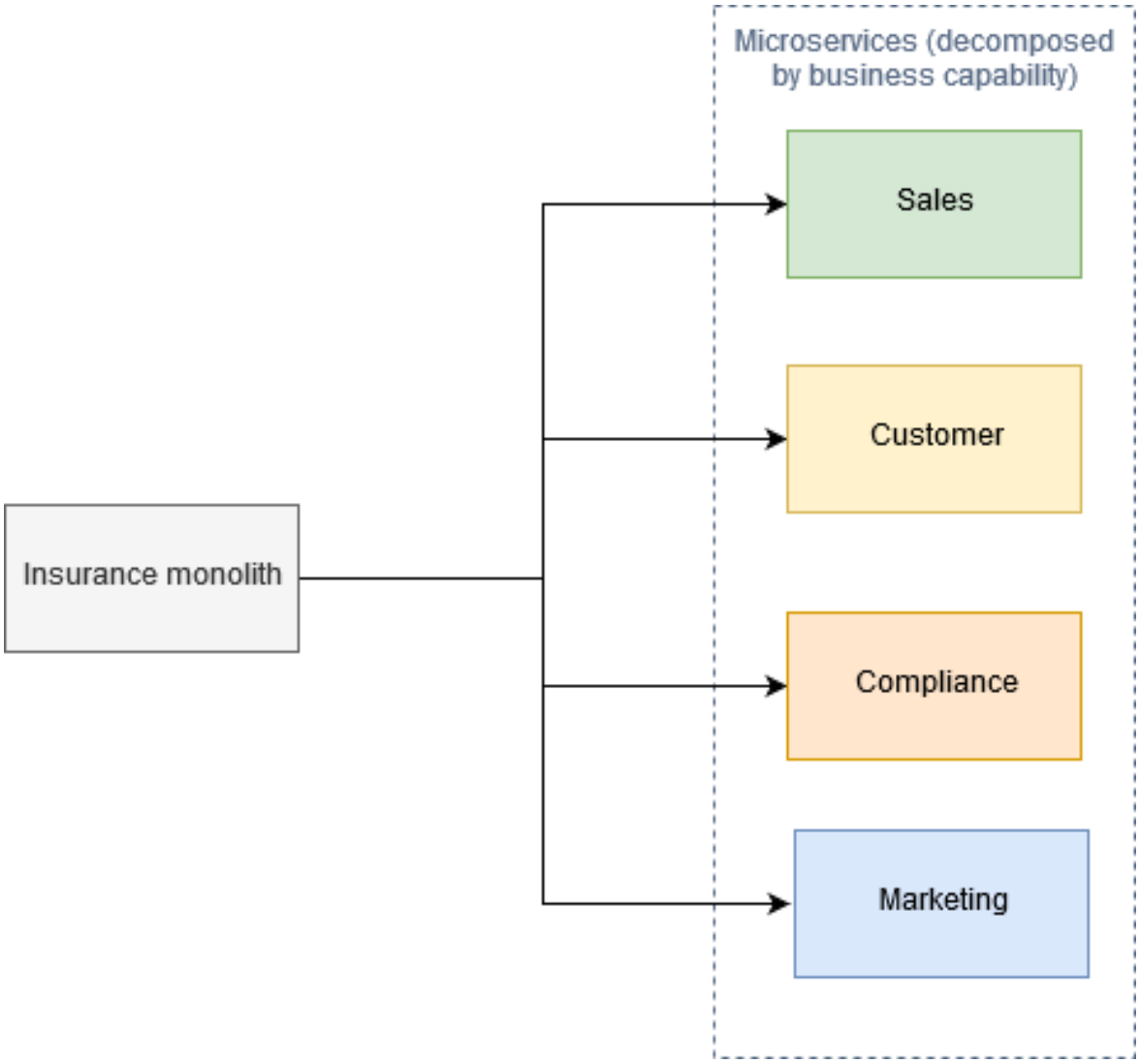
Topik

Terurai oleh kemampuan bisnis

Monolit dapat didekomposisi dengan menggunakan kemampuan bisnis organisasi Anda. SEBUAHkemampuan bisnisadalah apa yang dilakukan bisnis untuk menghasilkan nilai (misalnya, penjualan, layanan pelanggan, atau pemasaran). Biasanya, sebuah organisasi memiliki beberapa kemampuan bisnis dan ini bervariasi menurut sektor atau industri. Gunakan pola ini jika tim Anda memiliki wawasan yang cukup tentang unit bisnis organisasi Anda dan Anda memiliki ahli materi pelajaran (UKM) untuk setiap unit bisnis. Tabel berikut memberikan keuntungan dan kerugian menggunakan pola ini.

Keuntungan	Kekurangan
• Menghasilkan arsitektur layanan mikro yang stabil jika kemampuan bisnis relatif stabil. • Tim pengembangan bersifat cross-functional dan terorganisir di sekitar memberikan nilai bisnis, bukan fitur teknis. • Layanan yang longgar digabungkan.	• Desain aplikasi erat ditambah dengan model bisnis. • Membutuhkan pemahaman mendalam tentang bisnis secara keseluruhan, karena sulit untuk mengidentifikasi kemampuan dan layanan bisnis.

Dalam diagram berikut, monolit asuransi didekomposisi menjadi empat layanan mikro yang berbeda berdasarkan kemampuan bisnis.



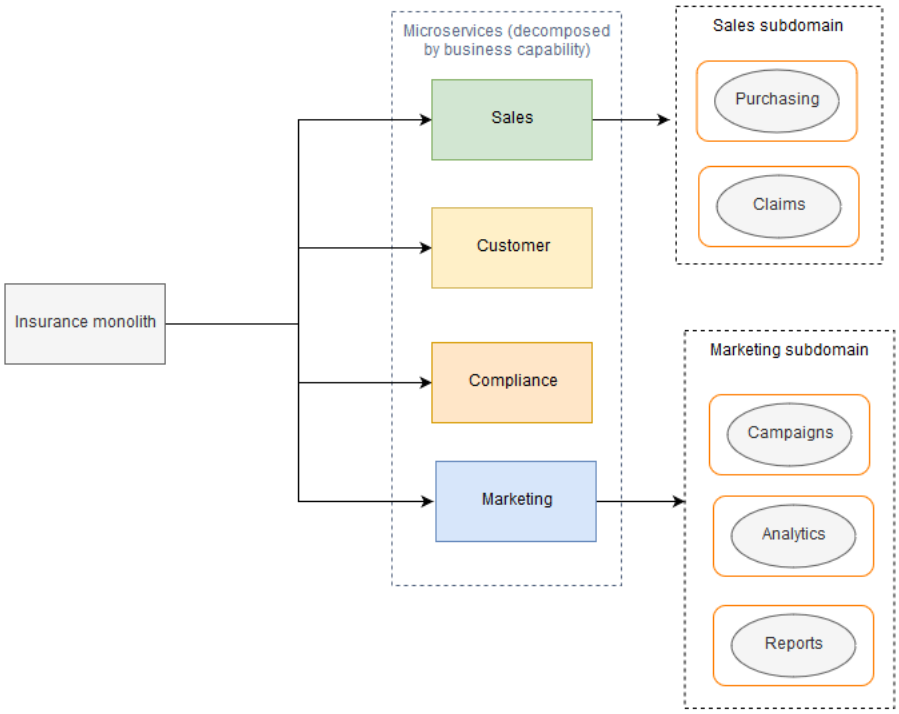
Terurai oleh subdomain

Pola ini menggunakan subdomain domain-driven design (DDD) untuk menguraikan monolit. Pendekatan ini memecah model domain organisasi menjadi subdomain terpisah yang diberi label sebagaiinti(pembeda kunci untuk bisnis),mendukung(mungkin terkait dengan bisnis tetapi bukan pembeda), ataugenerik(tidak spesifik bisnis). Pola ini sesuai untuk sistem monolitik yang ada yang memiliki batas yang terdefinisi dengan baik antara modul terkait subdomain. Ini berarti Anda dapat menguraikan monolit dengan mengemas ulang modul yang ada sebagai layanan mikro tetapi tanpa menulis ulang kode yang ada secara signifikan. Setiap subdomain memiliki model dan ruang lingkup model yang disebutkonteks terbatas; layanan mikro dikembangkan di sekitar konteks terbatas ini.

Tabel berikut memberikan keuntungan dan kerugian menggunakan pola ini.

Keuntungan	Kekurangan
• Arsitektur yang digabungkan secara longgar memberikan skalabilitas, ketahanan, pemeliharaan, perluasan, transparansi lokasi, independensi protokol, dan kemandirian waktu. • Sistem menjadi lebih terukur dan dapat diprediksi.	• Dapat membuat terlalu banyak layanan mikro, yang membuat penemuan layanan dan integrasi sulit. • Subdomain bisnis sulit untuk mengidentifikasi karena mereka memerlukan pemahaman mendalam tentang bisnis secara keseluruhan.

Ilustrasi berikut menunjukkan bagaimana monolit asuransi didekomposisi menjadi subdomain setelah didekomposisi oleh kemampuan bisnis.



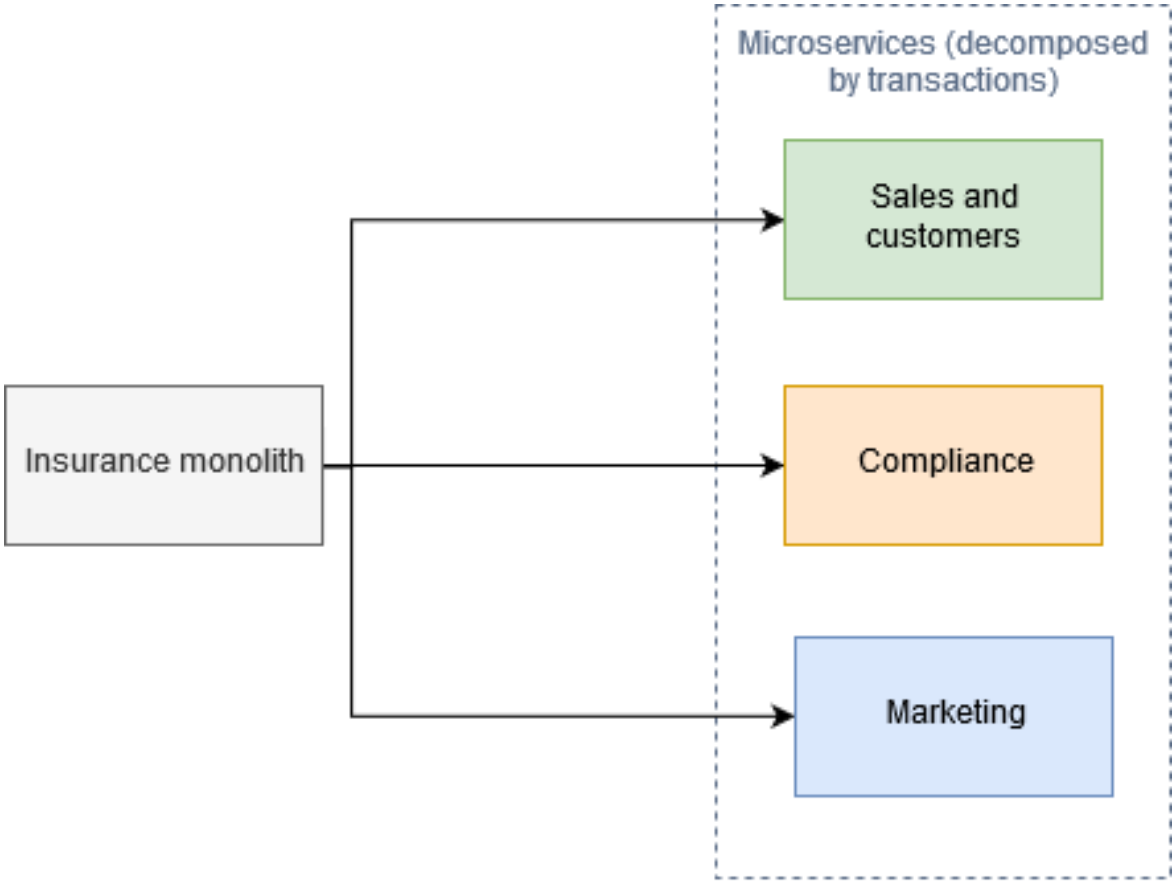
Ilustrasi menunjukkan bahwa layanan “Penjualan” dan “Pemasaran” dipecah menjadi layanan mikro yang lebih kecil. Model “Pembelian” dan “Klaim” adalah pembeda bisnis penting untuk “Penjualan,” dan dibagi menjadi dua layanan mikro terpisah. “Pemasaran” didekomposisi dengan menggunakan fungsi bisnis pendukung, seperti “Kampanye,” “Analytics,” dan “Laporan.”

Dekomposisi oleh transaksi

Dalam sistem terdistribusi, aplikasi biasanya perlu memanggil beberapa layanan mikro untuk menyelesaikan satu transaksi bisnis. Untuk menghindari masalah latensi atau masalah komit dua fase, Anda dapat mengelompokkan layanan mikro berdasarkan transaksi. Pola ini sesuai jika Anda mempertimbangkan waktu respons penting dan modul yang berbeda Anda tidak membuat monolit setelah Anda mengemas mereka. Tabel berikut memberikan keuntungan dan kerugian menggunakan pola ini.

Keuntungan	Kekurangan
• Waktu respon yang lebih cepat. • Anda tidak perlu khawatir tentang konsistensi data. • Peningkatan ketersediaan.	• Beberapa modul dapat dikemas bersama-sama, dan ini dapat membuat monolit. • Peningkatan biaya dan kompleksitas karena beberapa fungsi yang diimplementasikan dalam microservice, bukan sebagai microservice terpisah. • Layanan mikro berorientasi transaksi dapat tumbuh jika jumlah domain bisnis dan dependensi di antara mereka tinggi. • Versi yang tidak konsisten mungkin digunakan pada saat yang sama untuk domain bisnis yang sama.

Dalam ilustrasi berikut, monolit asuransi dipecah menjadi beberapa layanan mikro berdasarkan transaksi.



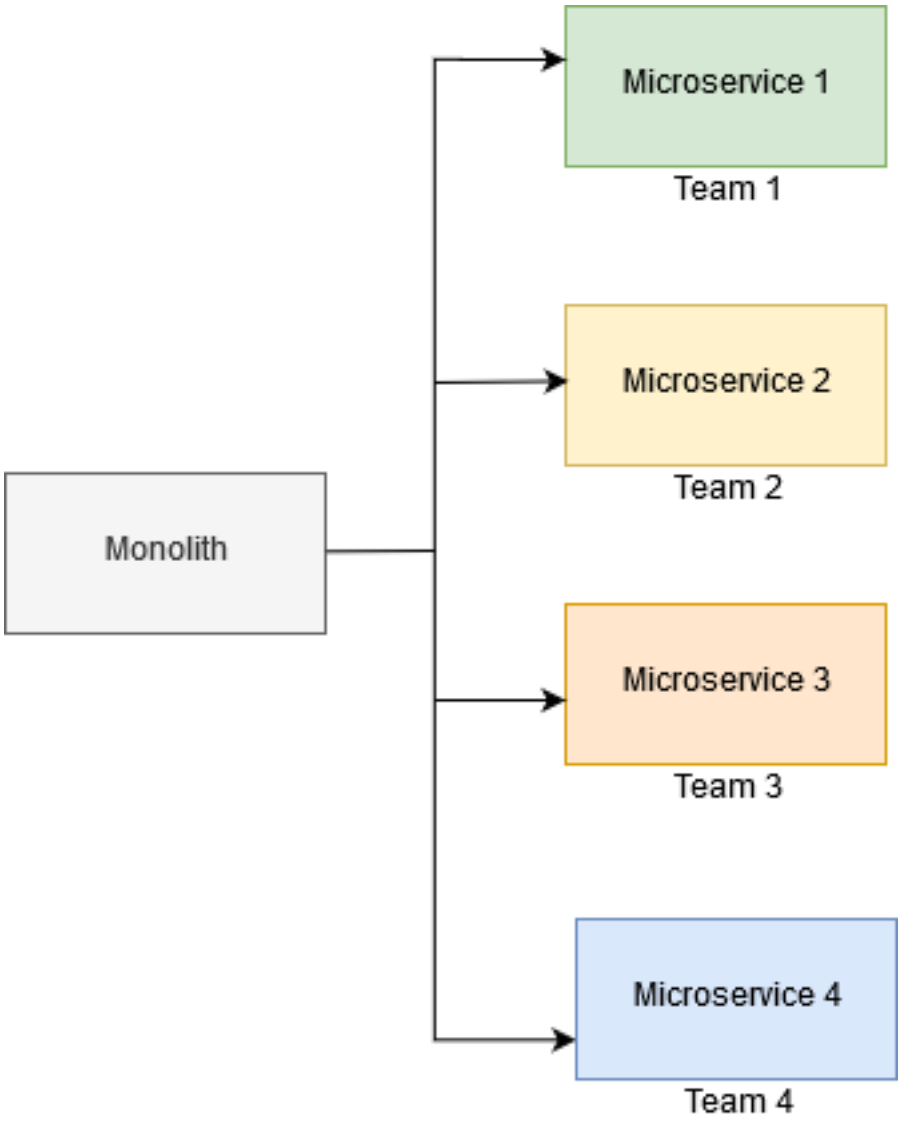
Dalam sistem asuransi, permintaan klaim biasanya ditandai kepada pelanggan setelah dikirimkan. Ini berarti bahwa layanan klaim tidak dapat ada tanpa layanan mikro “Pelanggan”. “Penjualan” dan “Pelanggan” dikemas bersama dalam satu paket microservice, dan transaksi bisnis memerlukan koordinasi dengan keduanya.

Layanan per pola tim

Alih-alih membusuk monolit oleh kemampuan atau layanan bisnis, pola layanan per tim memecahnya menjadi layanan mikro yang dikelola oleh tim individual. Setiap tim bertanggung jawab atas kemampuan bisnis dan memiliki basis kode kemampuan. Tim secara independen mengembangkan, menguji, menyebarkan, atau menskalakan layanannya, dan terutama berinteraksi dengan tim lain untuk menegosiasikan API. Kami merekomendasikan bahwa setiap microservice individu hanya dimiliki oleh satu tim. Namun, jika tim cukup besar, ada kemungkinan bahwa beberapa subtim dapat memiliki microservices terpisah dalam struktur tim yang sama. Tabel berikut memberikan keuntungan dan kerugian menggunakan pola ini.

Keuntungan	Kekurangan
• Tim bertindak mandiri dengan koordinasi minimal. • Basis kode dan layanan mikro tidak dibagikan oleh beberapa tim. • Tim dapat dengan cepat berinovasi dan mengulangi fitur produk. • Tim yang berbeda dapat menggunakan berbagai teknologi, kerangka kerja, atau bahasa pemrograman. Penting: Ini harus disembunyikan di balik API publik yang terdefinisi dengan baik dan stabil.	• Sulit untuk menyelaraskan tim dengan fungsionalitas pengguna akhir atau kemampuan bisnis. • Upaya tambahan diperlukan untuk memberikan peningkatan aplikasi yang lebih besar dan terkoordinasi, terutama jika ada dependensi melingkar antara tim.

Ilustrasi berikut menunjukkan bagaimana monolit dibagi menjadi layanan mikro yang dikelola, dipelihara, dan disampaikan oleh tim individual.


Microservice 1
Team 1
Microservice 2
Team 2
Monolith
Microservice 3
Team 3
Microservice 4
Team 4

FAQ monolit

Bagian ini memberikan jawaban atas pertanyaan yang sering diajukan tentang monolit yang membusuk.

Dapatkah Anda menggunakan beberapa pola untuk memecah satu monolit?

Ya, Anda dapat menggunakan beberapa pola untuk menguraikan monolit. Cara yang paling umum adalah menguraikan monolit denganTerurai oleh kemampuan bisnis (p. 3)pola dan kemudian menggunakanTerurai oleh subdomain (p. 4)pola untuk memecahnya lebih.

Bagaimana cara membusuk monolit menjadi microservices mempengaruhi proses DevOps?

Karena Anda tidak perlu memindahkan semuanya setelah perubahan dilakukan pada aplikasi, Anda harus memiliki dukungan dan kepemilikan layanan mikro yang baru dibuat yang ditambahkan ke proses penyebaran. Ini bisa membuat proses DevOps lebih kompleks.

Sumber daya

Panduan terkait

- Strategi untuk memodernisasi aplikasi diAWSCloud
- Pendekatan bertahap untuk memodernisasi aplikasi diAWSCloud
- Mengevaluasi kesiapan modernisasi untuk aplikasi diAWSCloud
- Mengintegrasikan layanan mikro dengan menggunakanAWSlayanan nirserver
- Mengaktifkan ketekunan data dalam layanan mikro

Sumber daya lainnya

- Memecah aplikasi monolitik menjadi layanan mikro dengan Amazon ECS, Docker, dan Amazon EC2

Riwayat dokumen

Tabel berikut menjelaskan perubahan signifikan pada panduan ini. Jika Anda ingin diberi tahu tentang pembaruan di future, Anda dapat berlangganan keUmpan RSS.

Perubahan	Deskripsi	Tanggal
Publikasi awal (p. 11)	—	11 Januari 2021

AWSDaftar istilah Panduan Preskriptif

Berikut ini adalah istilah yang umum digunakan dalam strategi, panduan, dan pola yang disediakan oleh PanduanAWS Preskriptif. Untuk menyarankan entri, silakan gunakan tautan Berikan umpan balik di akhir glosarium.

Istilah modernisasi

kemampuan bisnis

Apa yang dilakukan bisnis untuk menghasilkan nilai (misalnya, penjualan, layanan pelanggan, atau pemasaran). Arsitektur layanan mikro dan keputusan pengembangan dapat didorong oleh kemampuan bisnis. Untuk informasi selengkapnya, lihat bagian Terorganisir di sekitar kemampuan bisnis pada layanan mikro kontainer yang sedang berjalan diAWS whitepaper.

desain berbasis domain

Pendekatan untuk mengembangkan sistem perangkat lunak yang kompleks dengan menghubungkan komponennya ke domain yang berkembang, atau tujuan bisnis inti, yang masing-masing komponen berfungsi. Konsep ini diperkenalkan oleh Eric Evans dalam bukunya, Domain-Driven Design: Mengatasi Kompleksitas di Jantung Perangkat Lunak (Boston: Addison-Wesley Professional, 2003). Untuk informasi tentang bagaimana Anda dapat menggunakan desain berbasis domain dengan pola ara pencekik, lihat Memodernisasi layanan web Microsoft ASP.NET (ASMX) lama secara bertahap dengan menggunakan kontainer dan Amazon API Gateway.

layanan mikro

Layanan kecil dan independen yang berkomunikasi melalui API yang terdefinisi dengan baik dan biasanya dimiliki oleh tim kecil dan mandiri. Misalnya, sistem asuransi mungkin mencakup layanan mikro yang memetakan kemampuan bisnis, seperti penjualan atau pemasaran, atau subdomain, seperti pembelian, klaim, atau analitik. Manfaat layanan mikro termasuk kelincahan, penskalaan fleksibel, penerapan yang mudah, kode yang dapat digunakan kembali, dan ketahanan. Untuk informasi selengkapnya, lihat Mengintegrasikan layanan mikro dengan menggunakan layananAWS tanpa server.

arsitektur layanan mikro

Pendekatan untuk membangun aplikasi dengan komponen independen yang menjalankan setiap proses aplikasi sebagai microservice. Layanan mikro ini berkomunikasi melalui antarmuka yang terdefinisi dengan baik dengan menggunakan API ringan. Setiap layanan mikro dalam arsitektur ini dapat diperbarui, digunakan, dan diskalakan untuk memenuhi permintaan untuk fungsi tertentu dari suatu aplikasi. Untuk informasi selengkapnya, lihat Menerapkan Layanan Mikro diAWS.

modernisasi

Mengubah aplikasi usang (warisan atau monolitik) dan infrastrukturnya menjadi sistem yang gesit, elastis, dan sangat tersedia di cloud untuk mengurangi biaya, mendapatkan efisiensi, dan memanfaatkan inovasi. Untuk informasi selengkapnya, lihat Strategi untuk memodernisasi aplikasi diAWS Cloud.

penilaian kesiapan modernisasi

Evaluasi yang membantu menentukan kesiapan modernisasi aplikasi organisasi; mengidentifikasi manfaat, risiko, dan dependensi; dan menentukan seberapa baik organisasi dapat mendukung keadaan aplikasi tersebut di future. Hasil penilaian adalah cetak biru arsitektur target, peta jalan yang merinci fase pengembangan dan tonggak sejarah untuk proses modernisasi, dan rencana aksi

untuk mengatasi kesenjangan yang teridentifikasi. Untuk informasi selengkapnya, lihat Mengevaluasi kesiapan modernisasi untuk aplikasi diAWS Cloud.

aplikasi monolitik (monolit)

Aplikasi yang berjalan sebagai layanan tunggal dengan proses yang digabungkan erat. Aplikasi monolitik memiliki beberapa kelemahan. Jika salah satu fitur aplikasi mengalami lonjakan permintaan, seluruh arsitektur harus diskalakan. Menambahkan atau meningkatkan fitur aplikasi monolitik juga menjadi lebih kompleks ketika basis kode tumbuh. Untuk mengatasi masalah ini, Anda dapat menggunakan arsitektur layanan mikro. Untuk informasi lebih lanjut, lihat Mengurai monolit menjadi layanan mikro.

persistensi polyglot

Secara independen memilih teknologi penyimpanan data layanan mikro berdasarkan pola akses data dan persyaratan lainnya. Jika layanan mikro Anda memiliki teknologi penyimpanan data yang sama, mereka dapat menghadapi tantangan implementasi atau mengalami kinerja yang buruk. Layanan mikro lebih mudah diimplementasikan dan mencapai kinerja dan skalabilitas yang lebih baik jika mereka menggunakan penyimpanan data yang paling sesuai dengan kebutuhan mereka. Untuk informasi selengkapnya, lihat Mengaktifkan persistensi data di layanan mikro.

split-and-seed model

Sebuah pola untuk skala dan mempercepat proyek modernisasi. Karena fitur baru dan rilis produk didefinisikan, tim inti berpisah untuk membuat tim produk baru. Ini membantu meningkatkan kemampuan dan layanan organisasi Anda, meningkatkan produktivitas pengembang, dan mendukung inovasi yang cepat. Untuk informasi selengkapnya, lihat Pendekatan bertahap untuk memodernisasi aplikasi diAWS Cloud.

pola ara pencekik

Pendekatan untuk memodernisasi sistem monolitik dengan menulis ulang secara bertahap dan mengganti fungsionalitas sistem sampai sistem warisan dapat dinonaktifkan. Pola ini menggunakan analogi pohon ara yang tumbuh menjadi pohon yang mapan dan akhirnya mengatasi dan menggantikan inangnya. Pola ini diperkenalkan oleh Martin Fowler sebagai cara untuk mengelola risiko ketika menulis ulang sistem monolitik. Untuk contoh cara menerapkan pola ini, lihat Memodernisasi layanan web Microsoft ASP.NET (ASMX) lama secara bertahap dengan menggunakan kontainer dan Amazon API Gateway.

tim dua-pizza

Sebuah DevOps tim kecil yang dapat Anda makan dengan dua pizza. Ukuran tim dua pizza memastikan peluang terbaik untuk kolaborasi dalam pengembangan perangkat lunak. Untuk informasi lebih lanjut, lihat bagian tim dua-pizza dari Pengantar DevOps padaAWS whitepaper.

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.