AWSWhitepaper

Dasar-dasar Arsitektur SaaS

Dasar-dasar Arsitektur SaaS: AWSWhitepaper

Table of Contents

Dasar-dasar Arsitektur SaaS

Tanggal penerbitan: 3 Agustus 2022 () Revisi dokumen

Ruang lingkup, tujuan, dan sifat menjalankan bisnis dalam model perangkat lunak sebagai layanan (SaaS) bisa sulit untuk didefinisikan. Terminologi dan pola yang digunakan untuk mengkarakterisasi SaaS bervariasi berdasarkan asalnya. Tujuan dari dokumen ini adalah untuk lebih menentukan elemen-elemen dasar SaaS dan menciptakan gambaran yang lebih jelas tentang pola, istilah, dan sistem nilai yang diterapkan saat merancang dan memberikan sistem SaaS pada. AWS Tujuan yang lebih luas adalah untuk menyediakan kumpulan wawasan dasar yang memberi pelanggan pandangan yang lebih jelas tentang opsi yang harus mereka pertimbangkan saat mereka ingin mengadopsi model pengiriman SaaS.

Makalah ini ditargetkan untuk pembangun SaaS dan arsitek yang berada di awal perjalanan SaaS mereka, serta pembangun yang lebih berpengalaman yang ingin menyempurnakan pemahaman mereka tentang konsep SaaS inti. Beberapa informasi ini juga dapat bermanfaat bagi pemilik produk SaaS dan ahli strategi yang ingin lebih mengenal lanskap SaaS.

Pengantar

Istilah perangkat lunak sebagai layanan (SaaS) digunakan untuk menggambarkan model bisnis dan pengiriman. Tantangannya, bagaimanapun, adalah bahwa apa artinya menjadi SaaS tidak dipahami secara universal.

Meskipun ada beberapa kesepakatan tentang beberapa pilar inti SaaS, masih ada beberapa kebingungan di sekitar apa artinya menjadi SaaS. Itu wajar untuk memiliki beberapa variasi dalam bagaimana tim mungkin melihat SaaS. Pada saat yang sama, kurangnya kejelasan seputar konsep dan istilah SaaS dapat menimbulkan kebingungan bagi mereka yang menjelajahi model pengiriman SaaS.

Dokumen ini difokuskan pada menguraikan terminologi yang digunakan untuk menggambarkan konsep SaaS inti. Memiliki pola pikir bersama di sekitar konsep-konsep ini menciptakan gambaran yang jelas tentang elemen dasar arsitektur SaaS, melengkapi Anda dengan kosakata bersama untuk menggambarkan konstruksi arsitektur SaaS. Ini sangat berguna saat Anda menggali konten tambahan yang dibangun di atas tema-tema ini.

Whitepaper ini mundur dari detail arsitektur multi-tenancy, dan mengeksplorasi bagaimana kita mendefinisikan dasar-dasar dari apa artinya menjadi SaaS. Idealnya, ini juga akan memberikan

serangkaian terminologi yang lebih jelas yang memungkinkan organisasi untuk lebih cepat menyelaraskan rasa dan sifat solusi SaaS mereka.

Apakah Anda Well-Architected?

Kerangka Kerja AWS Well-Architected membantu Anda memahami pro dan kontra dari keputusan yang Anda buat saat membangun sistem di cloud. Enam pilar Kerangka ini memungkinkan Anda mempelajari praktik terbaik arsitektur untuk merancang dan mengoperasikan sistem yang andal, aman, efisien, hemat biaya, dan berkelanjutan. Dengan menggunakan AWS Well-Architected Tool, tersedia tanpa biaya di AWS Management Console, Anda dapat meninjau beban kerja Anda terhadap praktik terbaik ini dengan menjawab serangkaian pertanyaan untuk setiap pilar.

Dalam Lensa SaaS, kami fokus pada praktik terbaik untuk merancang perangkat lunak Anda sebagai beban kerja layanan (SaaS). AWS

Untuk panduan ahli dan praktik terbaik lainnya untuk arsitektur cloud Anda—referensi penerapan arsitektur, diagram, dan whitepaper—lihat Architecture Center. AWS

SaaS adalah model bisnis

Mendefinisikan apa artinya menjadi SaaS dimulai dengan menyetujui satu prinsip utama: SaaS adalah model bisnis. Ini berarti bahwa - di atas segalanya - adopsi model pengiriman SaaS secara langsung didorong oleh serangkaian tujuan bisnis. Ya, teknologi akan digunakan untuk mewujudkan beberapa tujuan tersebut, tetapi SaaS adalah tentang menempatkan pola pikir dan model yang menargetkan serangkaian tujuan bisnis tertentu.

Mari kita lihat lebih dekat beberapa tujuan bisnis utama yang terkait dengan mengadopsi model pengiriman SaaS.

- Agility - Istilah ini merangkum tujuan SaaS yang lebih luas. Perusahaan SaaS yang sukses dibangun dengan gagasan bahwa mereka harus siap untuk terus beradaptasi dengan pasar, pelanggan, dan dinamika kompetitif. Perusahaan SaaS yang hebat disusun untuk terus merangkul model harga baru, segmen pasar baru, dan kebutuhan pelanggan baru.
- Efisiensi operasional — Perusahaan SaaS mengandalkan efisiensi operasional untuk meningkatkan skala dan kelincahan. Ini berarti menempatkan budaya dan perkakas yang difokuskan pada menciptakan jejak operasional yang mempromosikan rilis fitur baru yang sering dan cepat. Ini juga berarti memiliki satu pengalaman terpadu yang memungkinkan Anda mengelola, mengoperasikan, dan menyebarkan semua lingkungan pelanggan secara kolektif. Lewatlah sudah ide mendukung versi dan kustomisasi satu kali. Bisnis SaaS menempatkan premi pada efisiensi operasional sebagai pilar inti dari kemampuan mereka untuk berhasil tumbuh dan skala bisnis.
- Orientasi tanpa gesekan - Sebagai bagian dari pertumbuhan yang lebih gesit dan merangkul, Anda juga harus menempatkan premi untuk mengurangi gesekan apa pun dalam proses orientasi pelanggan penyewa. Ini berlaku secara universal untuk pelanggan bisnis ke bisnis (B2B) dan business-to-customer (B2C). Apa pun segmen atau jenis pelanggan yang Anda dukung, Anda tetap perlu fokus pada waktu untuk menghargai pelanggan Anda. Pergeseran ke model yang berpusat pada layanan mengharuskan bisnis SaaS untuk fokus pada semua aspek pengalaman pelanggan, dengan penekanan khusus pada pengulangan dan efisiensi siklus hidup orientasi keseluruhan.
- Inovasi - Pindah ke SaaS bukan hanya tentang mengatasi kebutuhan pelanggan saat ini; ini tentang menempatkan elemen dasar yang memungkinkan Anda untuk berinovasi. Anda ingin bereaksi dan menanggapi kebutuhan pelanggan dalam model SaaS Anda. Namun, Anda juga ingin menggunakan kelincahan ini untuk mendorong inovasi future yang memungkinkan Anda membuka pasar, peluang, dan efisiensi baru bagi pelanggan Anda.

- Respons pasar - SaaS menjauh dari gagasan tradisional rilis triwulanan dan rencana dua tahun. Ini bergantung pada kelincahannya untuk memberi organisasi kemampuan untuk bereaksi dan merespons dinamika pasar dalam waktu dekat waktu nyata. Investasi dalam elemen organisasi, teknis, dan budaya SaaS menciptakan peluang untuk memutar strategi bisnis berdasarkan dinamika pelanggan dan pasar yang sedang berkembang.
- Pertumbuhan - SaaS adalah strategi bisnis yang berpusat pada pertumbuhan. Menyelaraskan semua bagian yang bergerak dari organisasi di sekitar kelincahan dan efisiensi memberikan organisasi SaaS kemampuan untuk menargetkan model pertumbuhan. Ini berarti menempatkan mekanisme yang merangkul dan menyambut adopsi cepat dari penawaran SaaS Anda.

Anda akan melihat bahwa masing-masing item ini difokuskan pada hasil bisnis. Ada berbagai strategi teknis dan pola yang dapat digunakan untuk membangun sistem SaaS. Namun, tidak ada tentang strategi teknis ini yang mengubah kisah bisnis yang lebih luas.

Saat kami duduk bersama organisasi dan bertanya kepada mereka apa yang ingin mereka capai sebagai bagian dari adopsi SaaS, kami selalu mulai dengan diskusi yang berfokus pada bisnis ini. Pilihan teknologi itu penting, tetapi harus diwujudkan dalam konteks tujuan bisnis ini. Menjadi multi-penyewa tanpa mencapai kelincahan, efisiensi operasional, atau orientasi tanpa gesekan, misalnya, akan merusak kesuksesan bisnis SaaS Anda.

Dengan ini sebagai latar belakang kita, mari kita coba meresmikan ini menjadi definisi SaaS yang lebih ringkas yang sesuai dengan prinsip-prinsip yang diuraikan sebelumnya:

SaaS adalah model pengiriman bisnis dan perangkat lunak yang memberi organisasi kemampuan untuk menawarkan solusi mereka dalam model gesekan rendah dan berpusat pada layanan yang memaksimalkan nilai bagi pelanggan dan penyedia. Ini bergantung pada kelincahan dan efisiensi operasional sebagai pilar strategi bisnis yang mendorong pertumbuhan, jangkauan, dan inovasi.

Anda harus melihat keselarasan antara tujuan bisnis dan bagaimana mereka mengandalkan memiliki pengalaman bersama untuk semua pelanggan. Sebagian besar pindah ke SaaS berarti menjauh dari penyesuaian satu kali yang mungkin menjadi bagian dari model perangkat lunak tradisional. Setiap upaya untuk menawarkan spesialisasi bagi pelanggan umumnya membawa kita menjauh dari nilai-nilai inti yang ingin kita capai dengan SaaS.

Anda adalah layanan, bukan produk

Mengadopsi model “layanan” lebih dari sekedar pemasaran atau terminologi. Dalam pola pikir layanan, Anda akan menemukan diri Anda bergerak menjauh dari aspek pendekatan berbasis produk tradisional untuk pengembangan. Meskipun fitur dan fungsionalitas tentu penting untuk setiap produk, SaaS lebih menekankan pada pengalaman yang akan dimiliki pelanggan dengan layanan Anda.

Apa artinya ini? Dalam model layanan-sentris, Anda berpikir lebih banyak tentang bagaimana pelanggan onboarded ke layanan Anda, seberapa cepat mereka mencapai nilai, dan seberapa cepat Anda dapat memperkenalkan fitur yang memenuhi kebutuhan pelanggan. Detail yang terkait dengan bagaimana layanan Anda dibangun, dioperasikan, dan dikelola berada di luar pandangan pelanggan Anda.

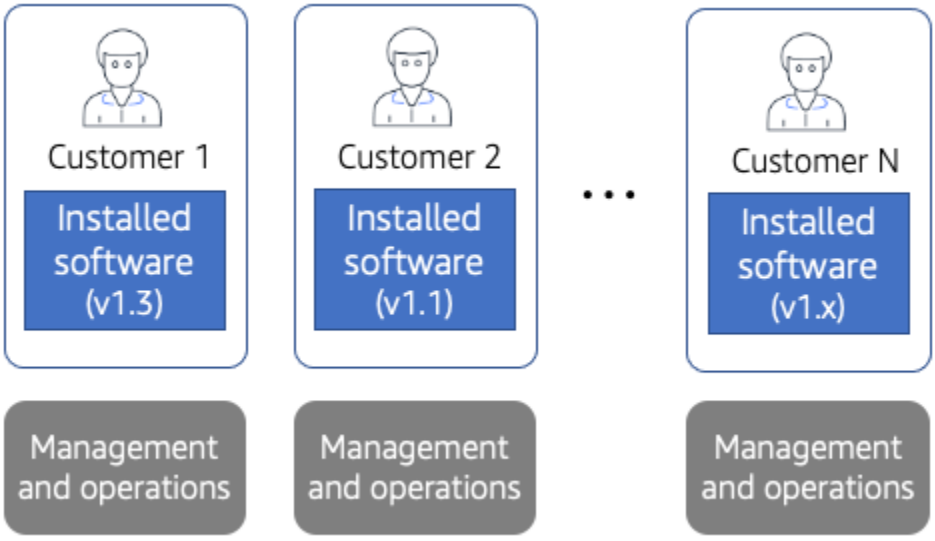
Dalam mode ini, kami memikirkan layanan SaaS ini seperti yang kami lakukan dengan layanan lain yang mungkin kami konsumsi. Jika kita berada di restoran, kita pasti peduli dengan makanannya, tapi kita juga peduli dengan layanannya. Seberapa cepat server Anda datang ke meja Anda, seberapa sering mereka mengisi ulang air Anda, seberapa cepat makanan datang — ini semua adalah ukuran pengalaman layanan. Ini adalah sistem pola pikir dan nilai yang sama yang seharusnya membentuk bagaimana kita berpikir tentang membangun layanan SaaS.

as-a-serviceModel ini harus memiliki pengaruh besar pada bagaimana Anda membangun tim dan layanan Anda. Backlog pekerjaan Anda sekarang akan menempatkan atribut pengalaman ini pada pijakan yang sama atau lebih tinggi daripada fitur dan fungsi. Bisnis juga akan melihat ini sebagai dasar untuk pertumbuhan jangka panjang dan keberhasilan penawaran SaaS Anda.

Motivasi awal motivasi

Untuk memahami SaaS, mari kita mulai dengan gagasan yang cukup sederhana tentang apa yang ingin kita capai saat membuat bisnis SaaS. Tempat terbaik untuk memulai adalah dengan melihat bagaimana perangkat lunak tradisional (non-SaaS) telah dibuat, dikelola, dan dioperasikan.

Diagram berikut memberikan pandangan konseptual tentang bagaimana beberapa vendor telah mengemas dan menyampaikan solusi mereka.



Model klasik untuk pengemasan dan memberikan solusi perangkat lunak

Dalam diagram ini, kami telah menjelaskan kumpulan lingkungan pelanggan. Pelanggan ini mewakili berbagai perusahaan atau entitas yang telah membeli perangkat lunak vendor. Masing-masing pelanggan ini pada dasarnya berjalan di lingkungan mandiri di mana mereka telah menginstal produk penyedia perangkat lunak.

Dalam mode ini, instalasi setiap pelanggan diperlakukan sebagai lingkungan mandiri yang didedikasikan untuk pelanggan itu. Ini berarti bahwa pelanggan melihat diri mereka sebagai pemilik lingkungan ini, berpotensi meminta penyesuaian satu kali atau konfigurasi unik yang mendukung kebutuhan mereka.

Salah satu efek samping yang umum dari pola pikir ini adalah bahwa pelanggan akan mengontrol versi produk yang mereka jalankan. Ini dilakukan karena berbagai alasan. Pelanggan mungkin

khawatir tentang fitur baru, atau khawatir tentang gangguan yang terkait dengan mengadopsi versi baru.

Anda dapat membayangkan bagaimana dinamika ini dapat memengaruhi jejak operasional penyedia perangkat lunak. Semakin Anda mengizinkan pelanggan untuk memiliki lingkungan satu kali, semakin menantang untuk mengelola, memperbarui, dan mendukung berbagai konfigurasi setiap pelanggan.

Kebutuhan akan lingkungan satu kali ini seringkali mengharuskan organisasi untuk membuat tim khusus yang menyediakan pengalaman manajemen dan operasi terpisah untuk setiap pelanggan. Sementara beberapa sumber daya ini mungkin dibagi di seluruh pelanggan, model ini umumnya memperkenalkan biaya tambahan untuk setiap pelanggan baru yang onboarded.

Memiliki setiap pelanggan menjalankan solusi mereka di lingkungan mereka sendiri (di cloud atau di tempat) juga berdampak biaya. Meskipun Anda dapat mencoba menskalakan lingkungan ini, penskalaan akan terbatas pada aktivitas satu pelanggan. Pada dasarnya, pengoptimalan biaya Anda terbatas pada apa yang dapat Anda capai dalam lingkup lingkungan pelanggan individu. Ini juga berarti bahwa Anda mungkin memerlukan strategi penskalaan terpisah untuk mengakomodasi variasi aktivitas antar pelanggan.

Awalnya, beberapa bisnis perangkat lunak akan melihat model ini sebagai konstruksi yang kuat. Mereka menggunakan kemampuan untuk menyediakan kustomisasi satu kali sebagai alat penjualan, memungkinkan pelanggan baru untuk memaksakan persyaratan yang unik untuk lingkungan mereka. Sementara jumlah pelanggan dan pertumbuhan bisnis tetap sederhana, model ini tampaknya sangat berkelanjutan.

Namun, ketika perusahaan mulai memiliki kesuksesan yang lebih luas, kendala model ini mulai menciptakan tantangan nyata. Bayangkan, misalnya, skenario di mana bisnis Anda mencapai lonjakan pertumbuhan yang signifikan yang membuat Anda menambahkan banyak pelanggan baru dengan cepat. Pertumbuhan ini akan mulai menambah overhead operasional, kompleksitas manajemen, biaya, dan sejumlah masalah lainnya.

Pada akhirnya, overhead kolektif dan dampak dari model ini dapat mulai secara fundamental merusak keberhasilan bisnis perangkat lunak. Titik nyeri pertama mungkin efisiensi operasional. Kepegawaian tambahan dan biaya yang terkait dengan membawa pelanggan mulai mengikis margin bisnis.

Namun, masalah operasional hanyalah bagian dari tantangan. Masalah sebenarnya adalah bahwa model ini, sesuai skala, secara langsung mulai memengaruhi kemampuan bisnis untuk merilis fitur-fitur baru dan mengimbangi pasar. Ketika setiap pelanggan memiliki lingkungan mereka sendiri,

penyedia harus menyeimbangkan sejumlah pembaruan, migrasi, dan persyaratan pelanggan saat mereka mencoba untuk memperkenalkan kemampuan baru ke dalam sistem mereka.

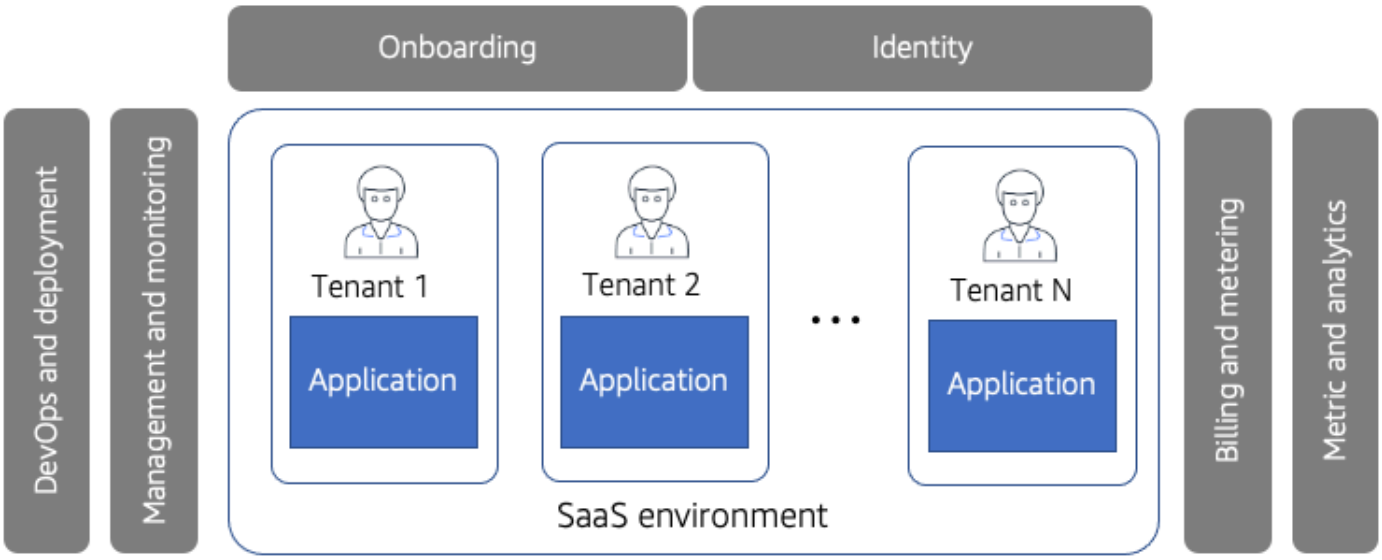
Ini biasanya mengarah ke siklus rilis yang lebih lama dan lebih kompleks, yang cenderung mengurangi jumlah rilis yang dibuat setiap tahun. Lebih penting lagi, kompleksitas ini memiliki tim yang mencurahkan lebih banyak waktu untuk menganalisis setiap fitur baru jauh sebelum dirilis ke pelanggan. Tim mulai lebih fokus pada memvalidasi fitur baru dan lebih sedikit pada kecepatan pengiriman. Overhead merilis fitur baru menjadi sangat signifikan sehingga tim menjadi lebih fokus pada mekanisme pengujian, dan lebih sedikit pada fungsionalitas baru yang mendorong inovasi penawaran mereka.

Dalam mode yang lebih lambat dan lebih berhati-hati ini, tim cenderung memiliki waktu siklus yang panjang yang menempatkan kesenjangan yang lebih besar dan lebih besar antara awal ide dan ketika mendarat di tangan pelanggan. Secara keseluruhan, ini dapat menghambat kemampuan Anda untuk bereaksi terhadap dinamika pasar dan tekanan kompetitif.

Pindah ke pengalaman terpadu

Untuk mengatasi kebutuhan dilema perangkat lunak klasik ini, organisasi beralih ke model yang memungkinkan mereka menciptakan pengalaman tunggal dan terpadu yang memungkinkan pelanggan dikelola dan dioperasikan secara kolektif.

Diagram berikut memberikan pandangan konseptual dari lingkungan di mana semua pelanggan dikelola, onboarded, ditagih, dan dioperasikan melalui model bersama.



Pandangan konseptual tentang lingkungan di mana semua pelanggan dikelola, di-onboard, ditagih, dan dioperasikan melalui model bersama

Pada pandangan pertama, ini mungkin tidak tampak semua yang berbeda dari model sebelumnya. Namun, saat kita menggali sedikit lebih jauh, Anda akan melihat bahwa ada perbedaan mendasar dan signifikan dalam dua pendekatan ini.

Pertama, Anda akan melihat bahwa lingkungan pelanggan telah diganti namanya menjadi penyewa. Gagasan penyewa ini mendasar bagi SaaS. Ide dasarnya adalah bahwa Anda memiliki lingkungan SaaS tunggal, dan masing-masing pelanggan Anda dipandang sebagai penyewa lingkungan itu, mengkonsumsi sumber daya yang mereka butuhkan. Penyewa bisa menjadi perusahaan dengan banyak pengguna, atau bisa berkorelasi langsung dengan pengguna individu.

Untuk lebih memahami gagasan penyewa, pertimbangkan gagasan apartemen atau bangunan komersial. Ruang di masing-masing bangunan ini disewakan untuk penyewa individu. Penyewa

mengandalkan beberapa sumber daya bersama bangunan (air, listrik, dan sebagainya), membayar apa yang mereka konsumsi.

Penyewa SaaS mengikuti pola yang sama. Anda memiliki infrastruktur lingkungan SaaS Anda, dan penyewa yang mengkonsumsi infrastruktur lingkungan itu. Jumlah sumber daya yang dikonsumsi oleh masing-masing penyewa dapat bervariasi. Penyewa ini juga dikelola, ditagih, dan dioperasikan secara kolektif.

Jika Anda kembali ke diagram, Anda akan melihat gagasan penyewaan dibawa ke kehidupan. Di sini, penyewa tidak lagi memiliki lingkungannya sendiri. Sebaliknya, semua penyewa ditempatkan dan dikelola dalam dinding satu lingkungan SaaS kolektif.

Diagram ini juga mencakup berbagai layanan bersama yang mengelilingi lingkungan SaaS Anda. Layanan ini bersifat global untuk semua penyewa lingkungan SaaS Anda. Ini berarti bahwa orientasi dan identitas, misalnya, dibagikan oleh semua penyewa lingkungan ini. Hal yang sama berlaku untuk manajemen, operasi, penyebaran, penagihan, dan metrik.

Gagasan tentang serangkaian layanan terpadu yang diterapkan secara universal untuk semua penyewa Anda adalah dasar bagi SaaS. Dengan membagikan konsep-konsep ini, Anda dapat mengatasi sejumlah tantangan yang terkait dengan model klasik yang dijelaskan di atas.

Kunci lain, elemen agak halus dari diagram ini adalah bahwa semua penyewa di lingkungan ini menjalankan versi yang sama dari aplikasi Anda. Lewatlah sudah ide untuk memiliki versi satu kali terpisah yang berjalan untuk setiap pelanggan. Memiliki semua penyewa yang menjalankan versi yang sama mewakili salah satu atribut pembeda mendasar dari lingkungan SaaS.

Dengan memiliki semua pelanggan menjalankan versi yang sama dari produk Anda, Anda tidak lagi menghadapi banyak tantangan dari model perangkat lunak yang diinstal klasik. Dalam model terpadu, fitur baru dapat digunakan untuk semua penyewa dengan satu proses bersama.

Pendekatan ini memberi Anda kemampuan untuk menggunakan satu panel kaca operasional yang dapat mengelola dan mengoperasikan semua penyewa. Hal ini memungkinkan Anda mengelola dan memantau penyewa Anda melalui pengalaman operasional umum, memungkinkan penyewa baru untuk ditambahkan tanpa menambahkan overhead operasional tambahan. Ini adalah bagian inti dari proposisi nilai SaaS yang memberi tim kemampuan untuk mengurangi biaya operasional, dan meningkatkan kelincahan organisasi secara keseluruhan.

Bayangkan apa artinya menambahkan 100 atau 1.000 pelanggan baru dalam model ini. Alih-alih khawatir tentang bagaimana pelanggan baru ini mungkin mengikis margin dan menambah kompleksitas, Anda dapat melihat pertumbuhan ini sebagai peluang yang diwakilinya.

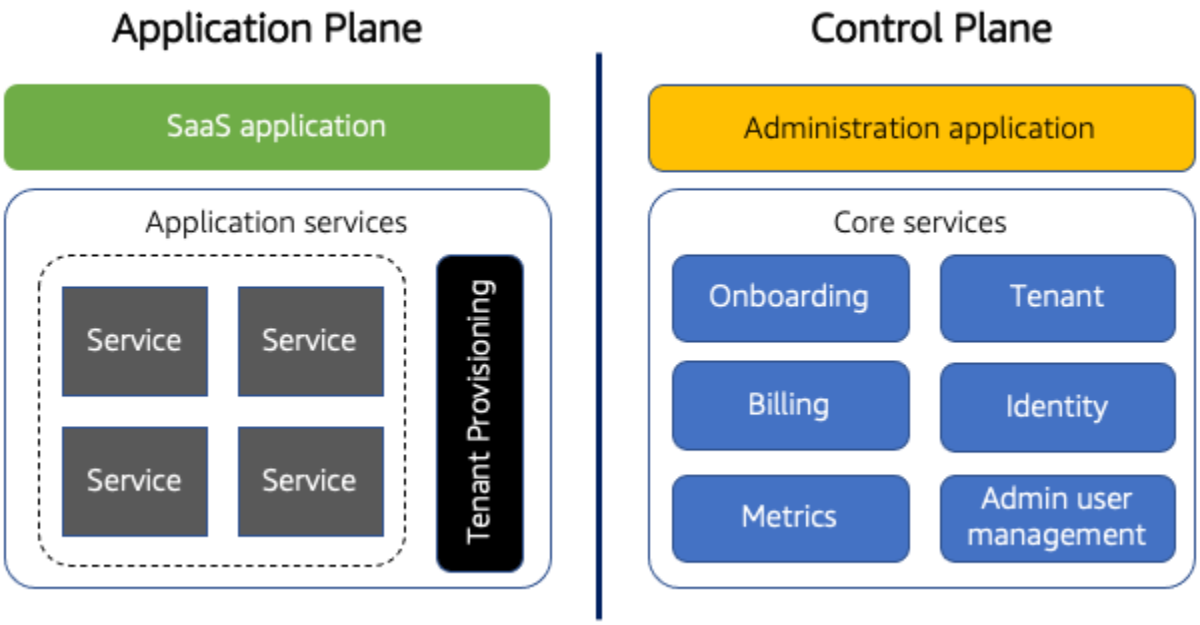
Umumnya, fokus SaaS ditempatkan pada bagaimana aplikasi di tengah model ini diimplementasikan. Bisnis ingin fokus pada bagaimana data disimpan, bagaimana sumber daya dibagikan, dan sebagainya. Namun, kenyataannya adalah bahwa, sementara rincian ini pasti penting, ada banyak cara aplikasi Anda dapat dibangun dan masih menampilkan dirinya sebagai solusi SaaS untuk pelanggan Anda.

Yang penting adalah tujuan yang lebih luas untuk memiliki satu pengalaman terpadu yang mengelilingi lingkungan penyewa Anda. Memiliki pengalaman bersama inilah yang memungkinkan Anda untuk mendorong pertumbuhan, kelincahan, dan efisiensi operasional yang terhubung ke tujuan keseluruhan bisnis SaaS.

Pesawat kontrol vs. bidang aplikasi

Diagram sebelumnya memberikan pandangan konseptual dari konsep arsitektur SaaS inti. Sekarang mari kita menelusuri lebih dalam, dan lebih baik mendefinisikan bagaimana lingkungan SaaS Anda terurai menjadi lapisan yang berbeda. Memiliki gambaran yang lebih jelas tentang batas-batas antara konsep SaaS akan membuatnya lebih mudah untuk menggambarkan bagian yang bergerak dari solusi SaaS.

Diagram berikut membagi lingkungan SaaS Anda menjadi dua bidang yang berbeda. Di sisi kanan adalah bidang kontrol. Sisi diagram ini mencakup semua fungsi dan layanan yang digunakan untuk onboard, mengotentikasi, mengelola, mengoperasikan, dan menganalisis lingkungan multi-penyewa.



Pesawat kontrol vs. bidang aplikasi

Pesawat kontrol ini adalah dasar untuk model SaaS multi-penyewa. Setiap solusi SaaS — terlepas dari penerapan aplikasi dan skema isolasi—harus mencakup layanan yang memberi Anda kemampuan untuk mengelola dan mengoperasikan penyewa Anda melalui satu pengalaman terpadu.

Dalam bidang kontrol, kita telah lebih lanjut dipecah ini menjadi dua elemen yang berbeda. Layanan inti di sini mewakili koleksi layanan yang digunakan untuk mengatur pengalaman multi-penyewa Anda. Kami telah menyertakan beberapa contoh umum layanan yang biasanya merupakan bagian dari inti, mengakui bahwa layanan inti dapat bervariasi untuk setiap solusi SaaS.

Anda juga akan melihat bahwa kami menampilkan aplikasi administrasi terpisah. Ini mewakili aplikasi (aplikasi web, antarmuka baris perintah, atau API) yang mungkin digunakan oleh penyedia SaaS untuk mengelola lingkungan multi-penyewa mereka.

Peringatan penting adalah bahwa bidang kontrol dan layanannya sebenarnya bukan multi-penyewa. Fungsionalitas ini tidak menyediakan atribut fungsional aktual dari aplikasi SaaS Anda (yang tidak perlu multi-penyewa). Jika Anda melihat ke dalam salah satu layanan inti, misalnya, Anda tidak akan menemukan isolasi penyewa dan konstruksi lain yang merupakan bagian dari fungsionalitas aplikasi multi-penyewa Anda. Layanan ini bersifat global untuk semua penyewa.

Sisi kiri diagram referensi bidang aplikasi dari lingkungan SaaS. Di sinilah fungsionalitas multi-penyewa aplikasi Anda berada. Apa yang muncul dalam diagram perlu tetap agak kabur, karena setiap solusi dapat digunakan dan didekomposisi secara berbeda berdasarkan kebutuhan domain Anda, jejak teknologi Anda, dan sebagainya.

Domain aplikasi dipisahkan menjadi dua elemen. Ada aplikasi SaaS yang mewakili pengalaman/aplikasi penyewa untuk solusi Anda. Ini adalah permukaan yang disentuh penyewa untuk berinteraksi dengan aplikasi SaaS Anda. Lalu ada layanan backend yang mewakili logika bisnis dan elemen fungsional dari solusi SaaS. Ini bisa berupa layanan mikro, atau kemasan lain dari layanan aplikasi Anda.

Anda juga akan mencatat bahwa kami telah melanggar ketentuan. Hal ini dilakukan untuk menyoroti fakta bahwa setiap penyediaan sumber daya untuk penyewa selama orientasi akan menjadi bagian dari domain aplikasi ini. Beberapa orang bisa berpendapat bahwa ini termasuk di bidang kendali. Namun, kami telah menempatkannya di domain aplikasi, karena sumber daya yang harus disediakan dan dikonfigurasi lebih terhubung langsung ke layanan yang dibuat dan dikonfigurasi di bidang aplikasi.

Memecah ini menjadi pesawat yang berbeda membuatnya lebih mudah untuk berpikir tentang lanskap keseluruhan arsitektur SaaS. Lebih penting lagi, ini menyoroti perlunya serangkaian layanan yang sepenuhnya berada di luar cakupan fungsionalitas aplikasi Anda.

Layanan inti

Pesawat kontrol yang direferensikan sebelumnya menyebutkan serangkaian layanan inti yang mewakili layanan khas yang digunakan untuk onboard, mengelola, dan mengoperasikan lingkungan SaaS. Mungkin akan membantu untuk lebih menyoroti peran beberapa layanan ini untuk menyoroti ruang lingkup dan tujuan mereka dalam lingkungan SaaS. Persingkat dari masing-masing layanan ini menyediakan ringkasan singkat dari masing-masing layanan ini:

- Orientasi - Setiap solusi SaaS harus menyediakan mekanisme tanpa gesekan untuk memperkenalkan penyewa baru ke lingkungan SaaS Anda. Ini bisa berupa halaman pendaftaran swalayan atau pengalaman yang dikelola secara internal. Bagaimanapun, solusi SaaS harus melakukan semua yang dapat menghilangkan gesekan internal dan eksternal dari pengalaman ini dan memastikan stabilitas, efisiensi, dan pengulangan untuk proses ini. Ini memainkan peran penting dalam mendukung pertumbuhan dan skala bisnis SaaS. Secara umum, layanan ini mengatur layanan lain untuk membuat pengguna, penyewa, kebijakan isolasi, penyediaan, dan sumber daya per-penyewa.
- Penyewa - Layanan penyewa menyediakan cara untuk memusatkan kebijakan, atribut, dan keadaan penyewa. Kuncinya adalah penyewa bukan pengguna perorangan. Bahkan, penyewa kemungkinan terkait dengan banyak pengguna.
- Identitas - Sistem SaaS membutuhkan cara yang jelas untuk menghubungkan pengguna ke penyewa yang akan membawa konteks penyewa ke pengalaman otentikasi dan otorisasi solusi mereka. Ini memengaruhi pengalaman orientasi dan pengelolaan profil pengguna secara keseluruhan.
- Penagihan - Sebagai bagian dari mengadopsi SaaS, organisasi sering merangkul model penagihan baru. Mereka juga dapat mengeksplorasi integrasi dengan penyedia penagihan pihak ketiga. Layanan inti ini sebagian besar difokuskan pada mendukung orientasi penyewa baru, dan mengumpulkan data konsumsi dan aktivitas yang digunakan untuk menghasilkan tagihan bagi penyewa.
- Metrik — Tim SaaS sangat bergantung pada kemampuan mereka untuk menangkap dan menganalisis data metrik kaya yang membawa lebih banyak visibilitas tentang bagaimana penyewa menggunakan sistem mereka, cara mereka mengkonsumsi sumber daya, dan bagaimana penyewa mereka melibatkan sistem mereka. Data ini digunakan untuk membentuk strategi operasional, produk, dan bisnis.

- Manajemen pengguna admin - Sistem SaaS harus mendukung pengguna penyewa dan pengguna admin. Pengguna admin mewakili administrator penyedia SaaS. Mereka akan masuk ke pengalaman operasional Anda untuk memantau dan mengelola lingkungan SaaS Anda.

Mendefinisikan ulang multi-tenancy

Istilah multi-tenancy dan SaaS sering terhubung erat. Dalam beberapa kasus, organisasi menggambarkan SaaS dan multi-tenancy sebagai hal yang sama. Meskipun ini mungkin tampak alami, menyamakan SaaS dan multi-tenancy cenderung memimpin tim untuk mengambil pandangan teknis murni SaaS ketika, pada kenyataannya, SaaS lebih merupakan model bisnis daripada strategi arsitektur.

Untuk lebih memahami konsep ini, mari kita mulai dengan pandangan klasik multi-tenancy. Dalam pandangan yang murni berfokus pada infrastruktur ini, multi-tenancy digunakan untuk menggambarkan bagaimana sumber daya dibagi oleh penyewa untuk mempromosikan kelincahan dan efisiensi biaya.

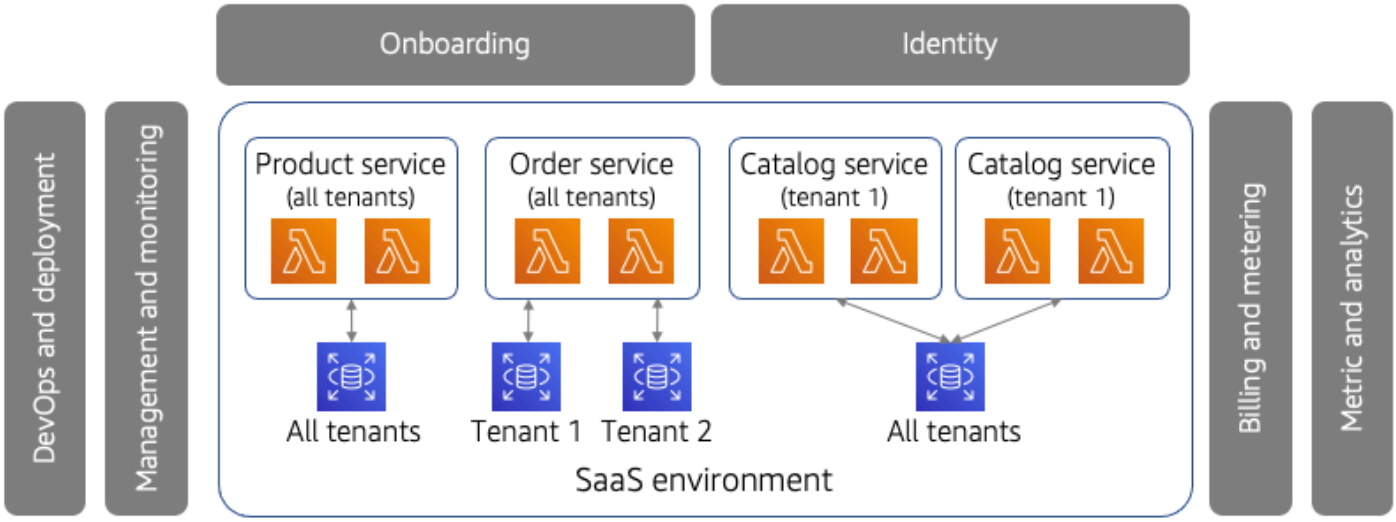
Misalkan, misalnya, Anda memiliki layanan mikro atau instans Amazon Elastic Compute Cloud (Amazon EC2) yang dikonsumsi oleh beberapa penyewa sistem SaaS Anda. Layanan ini akan dipandang berjalan dalam model multi-tenant, karena penyewa berbagi penggunaan infrastruktur yang menjalankan layanan ini.

Tantangan dari definisi ini adalah bahwa ia juga secara langsung melampirkan gagasan teknis multi-tenancy ke SaaS. Diasumsikan bahwa karakteristik SaaS yang menentukan adalah bahwa ia harus memiliki infrastruktur multi-penyewa bersama. Pandangan SaaS ini mulai berantakan ketika kita melihat berbagai cara SaaS diwujudkan di lingkungan yang berbeda.

Diagram berikut memberikan tampilan sistem SaaS yang mengekspos beberapa tantangan yang kami miliki dengan mendefinisikan multi-tenancy.

Di sini Anda akan melihat model SaaS klasik yang dijelaskan sebelumnya, dengan serangkaian layanan aplikasi yang dikelilingi oleh layanan bersama yang memungkinkan Anda mengelola dan mengoperasikan penyewa secara kolektif.

Yang baru adalah layanan mikro yang kami sertakan. Diagram mencakup tiga contoh layanan mikro: produk, pesanan, dan katalog. Jika Anda melihat lebih dekat pada model penyewaan masing-masing layanan ini, Anda akan melihat mereka semua menggunakan pola sewa yang sedikit berbeda.



SaaS dan multi-tenancy

Layanan produk berbagi semua sumber dayanya (komputasi dan penyimpanan) dengan semua penyewa. Ini sejalan dengan definisi klasik multi-tenancy. Namun, jika Anda melihat layanan pesanan, Anda akan melihat bahwa itu telah berbagi komputasi, tetapi penyimpanan terpisah untuk setiap penyewa.

Layanan katalog menambahkan variasi lain di mana komputasi terpisah untuk setiap penyewa (penyebaran layanan mikro terpisah untuk setiap penyewa), tetapi berbagi penyimpanan untuk semua penyewa.

Variasi sifat ini umum terjadi di lingkungan SaaS. Tetangga yang bising, model berjenjang, kebutuhan isolasi — ini adalah di antara daftar alasan yang mungkin Anda bagikan secara selektif atau silo bagian dari solusi SaaS Anda.

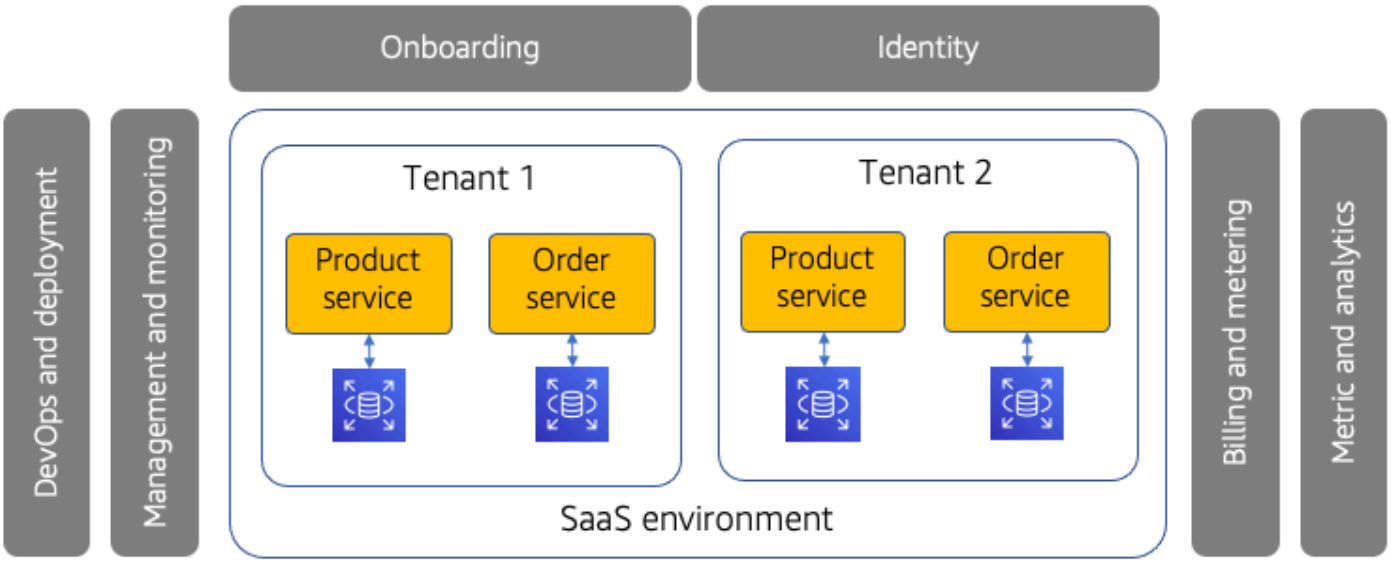
Mengingat variasi-variasi ini—dan banyak kemungkinan lainnya—menjadi lebih menantang untuk mengetahui bagaimana Anda harus menggunakan istilah multi-tenant untuk mengkarakterisasi lingkungan ini. Secara keseluruhan, sejauh menyangkut pelanggan, ini adalah lingkungan multi-penyewa. Namun, jika kita menggunakan definisi paling teknis, beberapa bagian dari lingkungan ini multi-penyewa dan beberapa tidak.

Inilah sebabnya mengapa menjadi perlu untuk menjauh dari menggunakan istilah multi-tenant untuk mengkarakterisasi lingkungan SaaS. Sebagai gantinya, kita dapat berbicara tentang bagaimana multi-tenancy diimplementasikan dalam aplikasi Anda, tetapi hindari menggunakannya untuk mengkarakterisasi solusi sebagai SaaS. Jika istilah multi-tenant akan digunakan, lebih masuk akal untuk menggunakannya untuk menggambarkan seluruh lingkungan SaaS sebagai multi-penyewa,

mengetahui bahwa beberapa bagian arsitektur dapat dibagi dan beberapa mungkin tidak. Secara keseluruhan, Anda masih mengoperasikan dan mengelola lingkungan ini dalam model multi-penyewa.

Kasus ekstrim

Untuk lebih menyoroti gagasan penyewaan ini, mari kita lihat model SaaS yang memiliki penyewa yang berbagi sumber daya nol. Diagram berikut memberikan contoh lingkungan SaaS yang digunakan oleh beberapa penyedia SaaS.



Tumpukan per penyewa

Dalam diagram ini, Anda akan melihat bahwa kita masih memiliki lingkungan bersama di sekitar penyewa ini. Namun, setiap penyewa dikerahkan dengan koleksi sumber daya khusus. Tidak ada yang dibagikan oleh penyewa dalam model ini.

Contoh ini menantang apa artinya menjadi multi-tenant. Apakah ini lingkungan multi-penyewa meskipun tidak ada sumber daya yang dibagikan? Penyewa yang menggunakan sistem ini memiliki semua harapan yang sama dengan yang Anda miliki tentang lingkungan SaaS dengan sumber daya bersama. Bahkan, mereka mungkin tidak memiliki kesadaran tentang bagaimana sumber daya mereka dikerahkan di bawah kap lingkungan SaaS.

Meskipun penyewa ini berjalan dalam infrastruktur siloed, mereka masih dikelola dan dioperasikan secara kolektif. Mereka berbagi satu orientasi terpadu, identitas, metrik, penagihan, dan pengalaman

operasional. Juga, ketika versi baru dirilis, itu digunakan untuk semua penyewa. Agar ini berfungsi, Anda tidak dapat mengizinkan penyesuaian satu kali untuk penyewa individu.

Contoh ekstrim ini memberikan model yang baik untuk menguji gagasan SaaS multi-penyewa. Meskipun mungkin tidak menyadari semua efisiensi infrastruktur bersama, ini adalah lingkungan SaaS multi-penyewa yang sepenuhnya valid. Untuk beberapa pelanggan, domain mereka mungkin mendikte bahwa beberapa atau semua pelanggan mereka menjalankan model ini. Itu tidak berarti mereka bukan SaaS. Jika mereka menggunakan layanan bersama ini dan semua penyewa menjalankan versi yang sama, ini masih sesuai dengan prinsip-prinsip dasar SaaS.

Mengingat parameter ini dan definisi SaaS yang lebih luas ini, Anda dapat melihat kebutuhan untuk mengembangkan penggunaan istilah multi-tenant. Lebih masuk akal untuk merujuk pada sistem SaaS apa pun yang dikelola dan dioperasikan secara kolektif sebagai multi-penyewa. Kemudian, Anda dapat tunduk pada terminologi yang lebih terperinci untuk menggambarkan bagaimana sumber daya dibagikan atau didedikasikan dalam implementasi solusi SaaS.

Menghapus istilah penyewa tunggal

Sebagai bagian dari penggunaan istilah multi-tenant, wajar bagi orang untuk ingin menggunakan istilah penyewa tunggal untuk menggambarkan lingkungan SaaS. Namun, mengingat latar belakang yang diuraikan sebelumnya, istilah penyewa tunggal menciptakan kebingungan.

Apakah diagram sebelumnya adalah lingkungan penyewa tunggal? Sementara setiap penyewa secara teknis memiliki tumpukannya sendiri, penyewa ini masih dioperasikan dan dikelola dalam model multi-tenant. Inilah sebabnya mengapa istilah penyewa tunggal umumnya dihindari. Sebaliknya, semua lingkungan dicirikan sebagai multi-penyewa, karena mereka hanya menerapkan beberapa variasi penyewaan di mana beberapa atau semua sumber daya dibagikan atau didedikasikan.

Memperkenalkan silo dan kolam

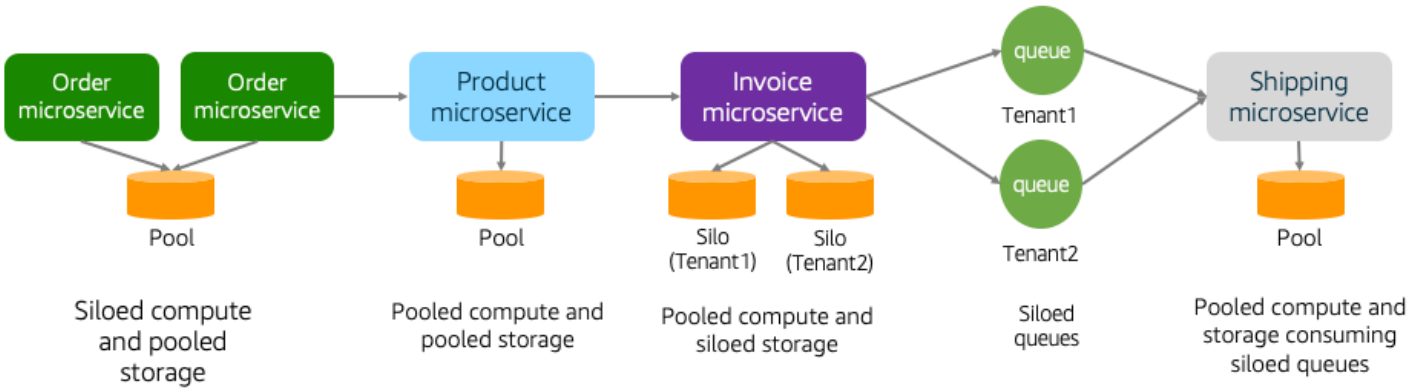
Dengan semua variasi model ini dan mengingat tantangan seputar istilah multi-tenancy, kami telah memperkenalkan beberapa terminologi yang memungkinkan kami menangkap dan menggambarkan model berbeda yang digunakan saat membangun SaaS dengan lebih akurat.

Dua istilah yang kita gunakan untuk mengkarakterisasi penggunaan sumber daya dalam lingkungan SaaS adalah silo dan pool. Istilah-istilah ini memungkinkan kami untuk memberi label sifat lingkungan SaaS, menggunakan multi-tenant sebagai deskripsi yang terlalu melengkung yang dapat diterapkan pada sejumlah model yang mendasarinya.

Pada tingkat yang paling dasar, istilah silo dimaksudkan untuk menggambarkan skenario di mana sumber daya didedikasikan untuk penyewa tertentu. Sebaliknya, model kolam digunakan untuk menggambarkan skenario di mana sumber daya dibagi oleh penyewa.

Saat kita melihat bagaimana istilah silo dan pool digunakan, penting untuk memperjelas bahwa silo dan pool bukanlah all-or-nothing konsep. Silo dan pool dapat diterapkan ke seluruh tumpukan sumber daya penyewa, atau dapat diterapkan secara selektif ke bagian dari lingkungan SaaS Anda secara keseluruhan. Jadi, jika kita mengatakan beberapa sumber daya menggunakan model silo, itu tidak berarti semua sumber daya di lingkungan itu dibungkam. Hal yang sama berlaku untuk bagaimana kita akan menggunakan istilah yang digabungkan.

Diagram berikut memberikan contoh bagaimana model siloed dan pooled digunakan lebih granular di lingkungan SaaS:



Model silo dan kolam renang

Diagram ini mencakup serangkaian sampel yang dimaksudkan untuk menggambarkan sifat model silo dan pool yang lebih bertarget. Jika Anda mengikuti ini dari kiri ke kanan, Anda akan melihat bahwa kita mulai dengan pesanan microservice. Microservice ini telah membungkam komputasi dan mengumpulkan penyimpanan. Ini berinteraksi dengan layanan produk yang telah mengumpulkan komputasi dan penyimpanan gabungan.

Layanan produk kemudian berinteraksi dengan layanan mikro faktur yang telah mengumpulkan komputasi dan penyimpanan silo. Layanan ini mengirimkan pesan melalui antrian ke layanan pengiriman. Antrian dikerahkan dalam model silo.

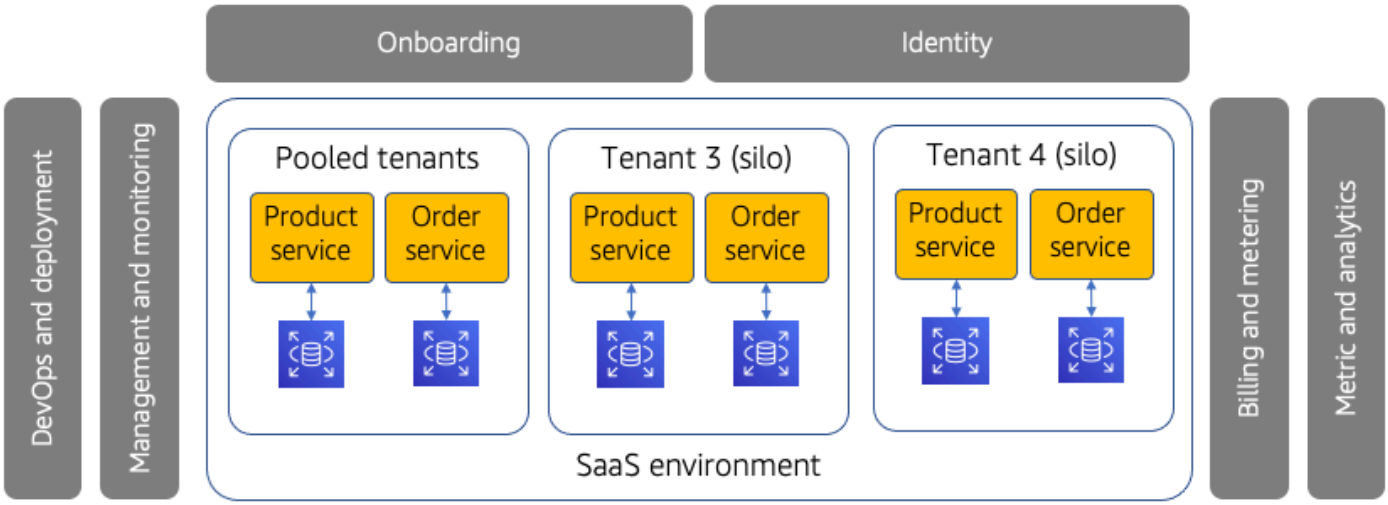
Akhirnya, microservice pengiriman memperoleh pesan dari antrian silo. Menggunakan komputasi dan penyimpanan yang terkumpul.

Meskipun ini mungkin tampak sedikit berbelit-belit, tujuannya adalah untuk menyoroti sifat granular dari konsep silo dan kolam renang. Saat Anda ingin merancang dan membangun solusi SaaS, diharapkan Anda akan membuat keputusan silo dan pool ini berdasarkan kebutuhan domain dan pelanggan Anda.

Tetangga yang bising, isolasi, tingkatan, dan sejumlah alasan lain mungkin memengaruhi bagaimana dan kapan Anda memilih untuk menerapkan silo atau model gabungan.

Silo dan kolam tumpukan penuh

Silo dan kolam renang juga dapat digunakan untuk menggambarkan seluruh tumpukan SaaS. Dalam pendekatan ini, semua sumber daya untuk penyewa dikerahkan baik secara khusus atau bersama. Diagram berikut memberikan contoh bagaimana ini mungkin mendarat di lingkungan SaaS.



Model silo dan kolam renang penuh

Dalam diagram ini, Anda akan melihat bahwa ada tiga model berbeda untuk penyebaran penyewa tumpukan penuh Anda. Pertama, Anda akan melihat bahwa ada lingkungan kolam tumpukan penuh. Penyewa di kolam ini berbagi semua sumber daya (komputasi, penyimpanan, dan sebagainya).

Dua tumpukan lainnya yang ditampilkan dimaksudkan untuk mewakili lingkungan penyewa silo tumpukan penuh. Dalam hal ini, Tenant 3 dan Tenant 4 ditampilkan karena masing-masing memiliki tumpukan khusus mereka sendiri di mana tidak ada sumber daya yang dibagikan dengan penyewa lain.

Campuran model silo dan gabungan di lingkungan SaaS yang sama tidak terlalu atipikal. Bayangkan, misalnya, bahwa Anda memiliki satu set penyewa tingkat dasar yang membayar harga moderat untuk menggunakan sistem Anda. Penyewa ini ditempatkan di lingkungan yang terkumpulnya.

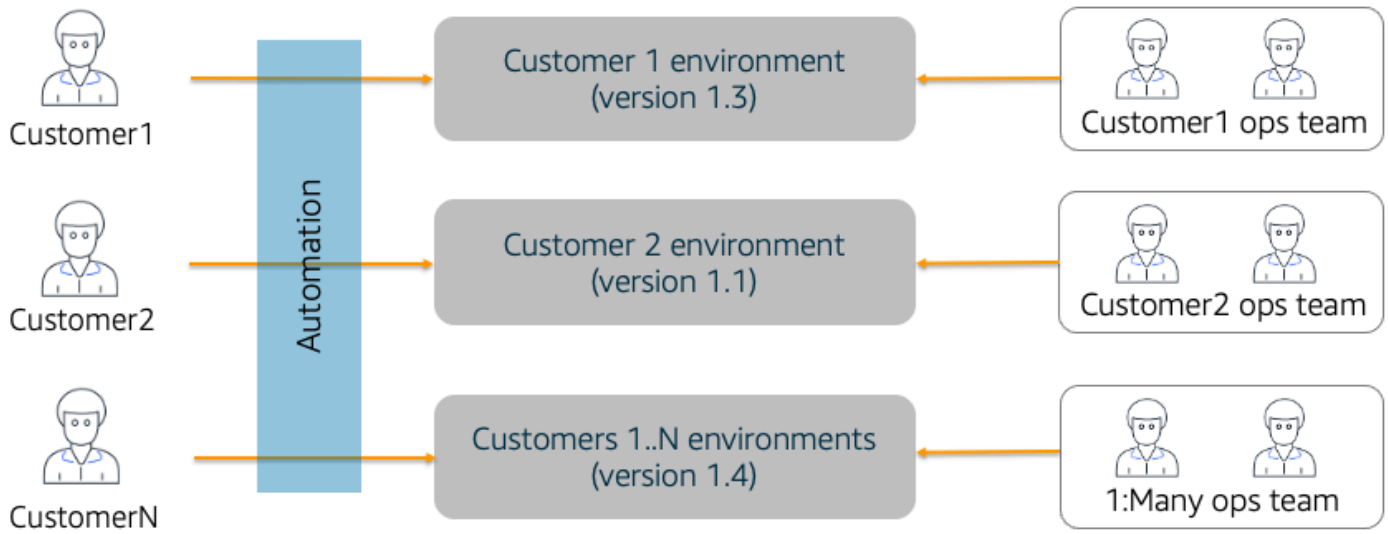
Sementara itu, Anda mungkin juga memiliki penyewa tingkat premium yang bersedia membayar lebih untuk hak istimewa berlari dalam silo. Pelanggan ini dikerahkan dengan tumpukan terpisah (seperti yang ditunjukkan pada diagram).

Bahkan dalam model ini, di mana Anda mungkin telah mengizinkan penyewa untuk menjalankan silo tumpukan penuh mereka sendiri, akan sangat penting bahwa silo ini tidak memungkinkan variasi atau penyesuaian satu kali untuk penyewa ini. Dalam segala hal, masing-masing tumpukan ini harus menjalankan konfigurasi tumpukan yang sama, dengan versi perangkat lunak yang sama. Ketika versi baru dirilis, itu diterapkan ke lingkungan penyewa yang digabungkan, dan masing-masing lingkungan yang dibungkam.

SaaS vs Penyedia Layanan Terkelola (MSP)

Ada juga beberapa kebingungan yang mengelilingi garis antara SaaS dan Managed Service Provider (MSP) model. Jika Anda melihat model MSP, model ini dapat tampak memiliki beberapa tujuan yang sama dengan model SaaS.

Namun, jika Anda menggali lebih banyak ke MSP, Anda akan menemukan bahwa MSP dan SaaS sebenarnya berbeda. Diagram berikut memberikan pandangan konseptual dari lingkungan MSP.



Model Penyedia Layanan Terkelola (MSP)

Diagram ini merupakan salah satu pendekatan untuk model MSP. Di sebelah kiri Anda akan melihat pelanggan yang berjalan dalam model MSP. Umumnya, pendekatan di sini adalah menggunakan otomatisasi apa pun yang tersedia untuk menyediakan setiap lingkungan pelanggan, dan menginstal perangkat lunak untuk pelanggan itu.

Di sebelah kanan adalah beberapa perkiraan jejak operasional yang akan diberikan MSP untuk mendukung lingkungan pelanggan ini.

Penting untuk dicatat bahwa MSP sering menginstal dan mengelola versi produk yang ingin dijalankan oleh pelanggan tertentu. Semua pelanggan dapat menjalankan versi yang sama, tetapi ini biasanya tidak diperlukan dalam model MSP.

Strategi umum adalah untuk menyederhanakan kehidupan penyedia perangkat lunak dengan memiliki instalasi dan pengelolaan lingkungan ini. Meskipun ini membuat hidup lebih sederhana bagi penyedia, ia tidak memetakan langsung ke nilai dan pola pikir yang penting untuk penawaran SaaS.

Fokusnya adalah pada pembongkaran tanggung jawab manajemen. Membuat langkah itu tidak setara dengan memiliki semua pelanggan berjalan pada versi yang sama dengan satu, manajemen terpadu dan pengalaman operasi. Sebaliknya, MSP sering memungkinkan untuk versi terpisah, dan sering memperlakukan masing-masing lingkungan ini sebagai operasional terpisah.

Tentu saja ada daerah di mana MSP mungkin mulai tumpang tindih dengan SaaS. Jika MSP pada dasarnya mengharuskan semua pelanggan untuk menjalankan versi yang sama dan MSP dapat secara terpusat onboard, mengelola, mengoperasikan, dan menagih semua penyewa melalui satu pengalaman, itu mungkin mulai lebih SaaS daripada MSP.

Tema yang lebih luas adalah bahwa mengotomatisasi pemasangan lingkungan tidak sama dengan memiliki lingkungan SaaS. Hanya ketika Anda menambahkan semua peringatan lain yang telah dibahas sebelumnya, ini akan mewakili lebih dari model SaaS sejati.

Jika kita mundur dari aspek teknologi dan operasi dari cerita ini, garis antara MSP dan SaaS menjadi lebih berbeda. Umumnya, sebagai bisnis SaaS, keberhasilan penawaran Anda bergantung pada kemampuan Anda untuk terlibat dalam semua bagian yang bergerak dari pengalaman.

Ini biasanya berarti memiliki jari Anda pada denyut nadi pengalaman orientasi, memahami bagaimana peristiwa operasional memengaruhi penyewa, melacak metrik dan analitik kunci, dan menjadi dekat dengan pelanggan Anda. Dalam model MSP di mana ini diserahkan kepada orang lain, Anda mungkin berakhir menjadi level yang dihapus dari detail utama yang merupakan inti untuk mengoperasikan bisnis SaaS.

migrasi SaaS

Banyak penyedia yang mengadopsi SaaS bermigrasi ke SaaS dari model perangkat lunak yang diinstal tradisional (dijelaskan sebelumnya). Untuk penyedia ini, sangat penting untuk memiliki keselarasan yang baik pada prinsip-prinsip inti SaaS.

Di sini, sekali lagi, adalah di mana ada beberapa kebingungan tentang apa artinya bermigrasi ke model SaaS. Beberapa, misalnya, melihat pindah ke cloud saat bermigrasi ke SaaS. Yang lain melihat penambahan otomatisasi ke proses instalasi dan penyediaan mereka sebagai pencapaian migrasi.

Adalah adil untuk mengatakan bahwa setiap organisasi dapat memulai di tempat yang berbeda, memiliki pertimbangan warisan yang berbeda, dan kemungkinan menghadapi pasar yang berbeda dan tekanan kompetitif. Ini berarti bahwa setiap migrasi akan terlihat berbeda.

Namun, sementara setiap jalur berbeda, ada beberapa area di mana ada terputus di sekitar prinsip-prinsip inti yang membentuk strategi migrasi. Memiliki keselarasan yang baik di sekitar konsep dan prinsip dapat berdampak signifikan pada keberhasilan keseluruhan migrasi SaaS Anda.

Berdasarkan konsep yang diuraikan sebelumnya, harus jelas bahwa perpindahan ke SaaS dimulai dengan strategi dan tujuan bisnis. Poin ini bisa hilang dalam pengaturan migrasi di mana ada tekanan untuk sampai ke SaaS secepat mungkin.

Dalam mode ini, organisasi sering melihat migrasi terutama sebagai latihan teknis. Kenyataannya adalah, setiap migrasi SaaS harus dimulai dengan pandangan yang jelas tentang target pelanggan, pengalaman layanan, tujuan operasional, dan sebagainya. Memiliki fokus yang lebih jelas pada apa yang dibutuhkan bisnis SaaS Anda akan memiliki dampak besar pada bentuk, prioritas, dan jalur yang Anda ambil untuk memigrasikan solusi Anda ke SaaS.

Memiliki visi yang jelas ini sejak awal migrasi Anda meletakkan dasar untuk bagaimana Anda memigrasikan teknologi dan bisnis sebagai bagian dari kepindahan Anda ke SaaS. Saat Anda berangkat di jalur ini, fokuslah pada pertanyaan-pertanyaan yang dapat memberi tahu Anda paling banyak tentang ke mana Anda menuju.

Tabel berikut memberikan pandangan tentang sifat kontras pola pikir migrasi teknis dan bisnis.

Tabel 1 - Migrasi pertama teknis vs. bisnis pertama

Pola pikir pertama teknologi	Pola pikir bisnis pertama
Bagaimana kita mengisolasi data penyewa?	Bagaimana SaaS dapat membantu kami mengembangkan bisnis kami?
Bagaimana kita menghubungkan pengguna dengan penyewa?	Segmen mana yang kita targetkan?
Bagaimana kita menghindari kondisi tetangga yang bising?	Berapa ukuran dan profil segmen ini?
Bagaimana kita melakukan pengujian A/B?	Tingkat apa yang perlu kita dukung?
Bagaimana kita menskalakan berdasarkan beban penyewa?	Pengalaman layanan apa yang kami targetkan?
Penyedia penagihan mana yang harus kita gunakan?	Apa strategi penetapan harga dan pengemasan kami?

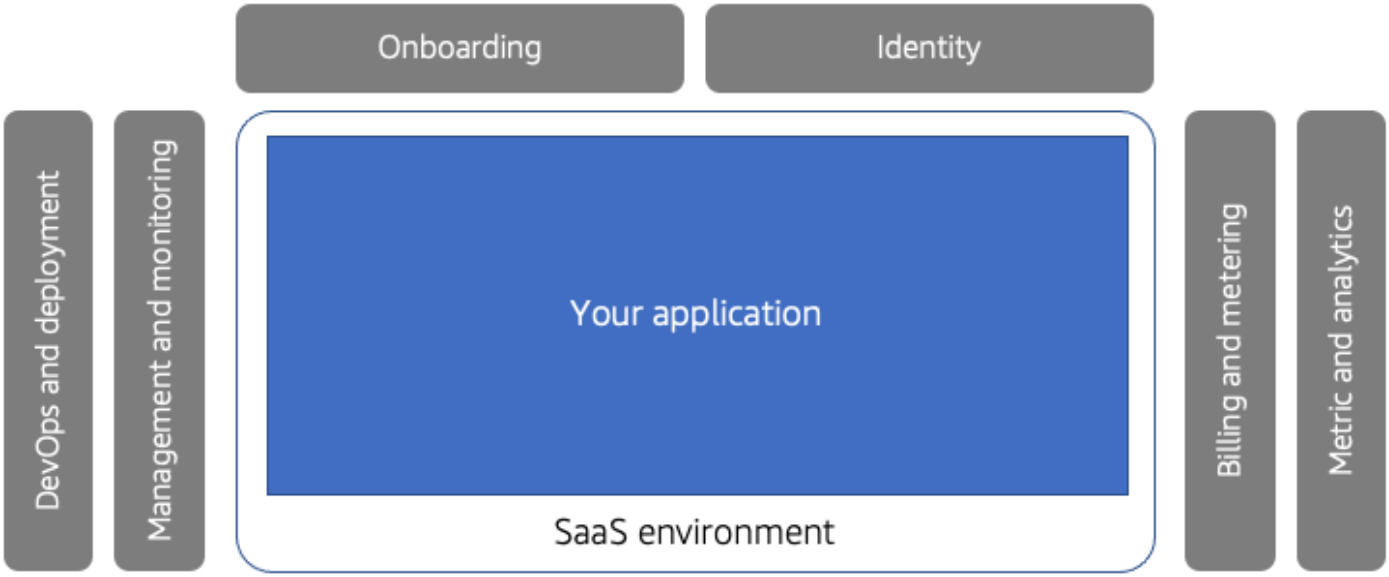
Di sebelah kiri adalah contoh seperti apa migrasi teknologi. Tim teknik sangat fokus pada mengejar topik multi-penyewa klasik yang tentunya penting bagi arsitektur SaaS mana pun.

Masalahnya adalah bahwa jawaban atas banyak pertanyaan di sebelah kiri sering dipengaruhi secara langsung oleh jawaban atas pertanyaan di sebelah kanan. Tidak mungkin titik ini baru bagi siapa pun yang melihat migrasi. Namun, kenyataannya adalah bahwa banyak organisasi memulai dengan mengejar efisiensi operasional dan biaya sebagai langkah pertama mereka, dengan asumsi bit bisnis akan bekerja sendiri.

Dalam strategi migrasi ini, ada juga kebingungan tentang bagaimana lingkungan warisan Anda mungkin berevolusi agar sesuai dengan model SaaS. Ini juga merupakan area di mana ada banyak pilihan untuk bermigrasi ke SaaS. Namun, ada sistem nilai umum yang umumnya kami anjurkan untuk migrasi apa pun.

Dalam diskusi kami sebelumnya tentang prinsip-prinsip SaaS, kami menguraikan berbagai pola dan terminologi yang digunakan untuk menggambarkan lingkungan SaaS. Salah satu tema umum yang mencakup semua solusi tersebut adalah gagasan memiliki layanan bersama yang mengelilingi aplikasi Anda. Identitas, orientasi, metrik, penagihan—ini semua disebut sebagai elemen umum yang merupakan inti dari lingkungan SaaS mana pun.

Sekarang, saat kita melihat migrasi, Anda akan melihat bahwa layanan bersama yang sama ini memainkan peran kunci dalam cerita migrasi apa pun. Berikut ini adalah diagram berikut ini.



Migrasi ke SaaS

Diagram ini merupakan pengalaman target untuk setiap jalur migrasi. Ini mencakup semua layanan bersama yang sama yang dijelaskan sebelumnya. Di tengah adalah placeholder untuk aplikasi Anda.

Ide utamanya adalah Anda dapat mendaratkan sejumlah model aplikasi di tengah lingkungan ini. Langkah pertama Anda dalam migrasi mungkin memiliki setiap penyewa berjalan di silo sendiri. Atau, Anda mungkin memiliki beberapa arsitektur hibrida di mana elemen dibungkam dan bit fungsionalitas lainnya ditangani melalui kumpulan layanan mikro yang dimodernisasi.

Seberapa banyak Anda awalnya memodernisasi aplikasi Anda akan bervariasi berdasarkan sifat lingkungan warisan Anda, kebutuhan pasar, pertimbangan biaya, dan sebagainya. Yang tidak berubah adalah pengenalan layanan bersama ini.

Setiap migrasi SaaS perlu mendukung layanan bersama dasar ini untuk memberi bisnis Anda kemampuan untuk beroperasi dalam model SaaS. Semua variasi arsitektur aplikasi membutuhkan identitas SaaS, misalnya. Anda memerlukan operasi yang sadar penyewa untuk mengelola dan memantau solusi SaaS Anda.

Menempatkan layanan bersama ini di depan migrasi memungkinkan Anda menyajikan pengalaman SaaS kepada pelanggan Anda—bahkan jika aplikasi yang mendasarinya masih berjalan dalam silo tumpukan penuh untuk setiap penyewa.

Tujuan umum adalah untuk mendapatkan aplikasi Anda berjalan dalam model SaaS. Kemudian, Anda dapat mengalihkan perhatian Anda ke modernisasi lebih lanjut dan penyempurnaan aplikasi Anda. Pendekatan ini juga memungkinkan Anda untuk memindahkan bagian lain dari bisnis Anda (pemasaran, penjualan, dukungan, dan sebagainya) dengan kecepatan yang lebih cepat. Lebih penting lagi, ini memungkinkan Anda untuk mulai terlibat dan mengumpulkan umpan balik pelanggan yang dapat digunakan untuk membentuk modernisasi lingkungan Anda yang sedang berlangsung.

Penting untuk dicatat bahwa layanan bersama yang Anda tempatkan mungkin tidak menyertakan setiap fitur atau mekanisme yang pada akhirnya Anda perlukan. Tujuan utamanya adalah menciptakan mekanisme bersama yang diperlukan pada awal migrasi Anda. Ini memungkinkan Anda untuk fokus pada elemen-elemen sistem yang penting untuk evolusi arsitektur aplikasi Anda dan evolusi operasional Anda.

Identitas SaaS

SaaS menambahkan pertimbangan baru ke model identitas aplikasi Anda. Karena setiap pengguna diautentikasi, mereka harus terhubung ke konteks penyewa tertentu. Konteks penyewa ini memberikan informasi penting tentang penyewa Anda yang digunakan di seluruh lingkungan SaaS Anda.

Pengikatan penyewa ke pengguna ini sering disebut sebagai identitas SaaS aplikasi Anda. Karena setiap pengguna mengautentikasi, penyedia identitas Anda biasanya akan menghasilkan token yang mencakup identitas pengguna dan identitas penyewa.

Menghubungkan penyewa ke pengguna merupakan aspek dasar dari arsitektur SaaS Anda yang memiliki banyak implikasi hilir. Token dari proses identitas ini mengalir ke layanan mikro aplikasi Anda dan digunakan untuk membuat log sadar penyewa, metrik catatan, penagihan meter, menegakkan isolasi penyewa, dan sebagainya.

Penting bagi Anda untuk menghindari skenario yang mengandalkan mekanisme terpisah dan mandiri yang memetakan pengguna ke penyewa. Hal ini dapat merusak keamanan sistem Anda, dan sering menciptakan hambatan dalam arsitektur Anda.

Isolasi penyewa

Semakin Anda memindahkan pelanggan ke model multi-penyewa, semakin mereka akan khawatir tentang potensi satu penyewa untuk mengakses sumber daya penyewa lain. Sistem SaaS mencakup mekanisme eksplisit yang memastikan bahwa sumber daya masing-masing penyewa — bahkan jika mereka berjalan pada infrastruktur bersama — terisolasi.

Inilah yang kita sebut sebagai isolasi penyewa. Ide di balik isolasi penyewa adalah bahwa arsitektur SaaS Anda memperkenalkan konstruksi yang mengontrol akses ke sumber daya secara ketat, dan memblokir upaya apa pun untuk mengakses sumber daya penyewa lain.

Perhatikan bahwa isolasi penyewa terpisah dari mekanisme keamanan umum. Sistem Anda akan mendukung otentikasi dan otorisasi; Namun, fakta bahwa pengguna penyewa diautentikasi tidak berarti bahwa sistem Anda telah mencapai isolasi. Isolasi diterapkan secara terpisah dari otentikasi dasar dan otorisasi yang mungkin menjadi bagian dari aplikasi Anda.

Untuk lebih memahami hal ini, bayangkan Anda telah menggunakan penyedia identitas untuk mengautentikasi akses ke sistem SaaS Anda. Token dari pengalaman otentikasi ini juga dapat mencakup informasi tentang peran pengguna yang dapat digunakan untuk mengontrol akses pengguna tersebut ke fungsionalitas aplikasi tertentu. Konstruksi ini memberikan keamanan, tetapi tidak isolasi. Bahkan, pengguna dapat diautentikasi dan diotorisasi, dan masih mengakses sumber daya penyewa lain. Tidak ada tentang otentikasi dan otorisasi tentu akan memblokir akses ini.

Isolasi penyewa berfokus secara eksklusif pada penggunaan konteks penyewa untuk membatasi akses ke sumber daya. Ini mengevaluasi konteks penyewa saat ini, dan menggunakan konteks itu untuk menentukan sumber daya mana yang dapat diakses oleh penyewa itu. Ini berlaku isolasi ini untuk semua pengguna dalam penyewa itu.

Ini menjadi lebih menantang saat kita melihat bagaimana isolasi penyewa diwujudkan di semua pola arsitektur SaaS yang berbeda. Dalam beberapa kasus, isolasi dapat dicapai dengan memiliki seluruh tumpukan sumber daya yang didedikasikan untuk penyewa di mana jaringan (atau lebih kasar) kebijakan mencegah akses lintas penyewa. Dalam skenario lain, Anda mungkin memiliki sumber daya yang dikumpulkan (item dalam tabel Amazon DynamoDB) yang memerlukan kebijakan yang lebih halus untuk mengontrol akses ke sumber daya.

Setiap upaya untuk mengakses sumber daya penyewa harus dilingkupi hanya untuk sumber daya yang dimiliki penyewa itu. Adalah tugas pengembang dan arsitek SaaS untuk menentukan kombinasi alat dan teknologi mana yang akan mendukung persyaratan isolasi aplikasi spesifik Anda.

Pembuatan partisi data

Partisi data digunakan untuk menggambarkan strategi yang berbeda digunakan untuk mewakili data dalam lingkungan multi-penyewa. Istilah ini digunakan secara luas untuk mencakup berbagai pendekatan dan model yang berbeda yang dapat digunakan untuk mengaitkan konstruksi data yang berbeda dengan penyewa individu.

Perhatikan bahwa sering ada godaan untuk melihat partisi data dan isolasi penyewa sebagai dipertukarkan. Kedua konsep ini tidak dimaksudkan untuk setara. Ketika kita berbicara tentang partisi data, kita berbicara tentang bagaimana data penyewa disimpan untuk penyewa individu. Partisi data tidak memastikan bahwa data terisolasi. Isolasi masih harus diterapkan secara terpisah untuk memastikan bahwa satu penyewa tidak dapat mengakses sumber daya penyewa lain.

Setiap teknologi AWS penyimpanan membawa set sendiri pertimbangan untuk strategi partisi data. Misalnya, mengisolasi data di Amazon DynamoDB akan terlihat sangat berbeda dengan mengisolasi data dengan Amazon Relational Database Service (Amazon RDS).

Umumnya, ketika Anda berpikir tentang partisi data, Anda mulai dengan berpikir tentang apakah data akan dibungkam atau dikumpulkan. Dalam model silo, Anda memiliki konstruksi penyimpanan yang berbeda untuk setiap penyewa tanpa data yang digabungkan bersama. Untuk partisi yang dikumpulkan, data digabungkan bersama dan dipartisi berdasarkan pengenal penyewa yang menentukan data mana yang terkait dengan masing-masing penyewa.

Sebagai contoh, dengan Amazon DynamoDB, model silo menggunakan tabel terpisah untuk setiap penyewa. Mengumpulkan data di Amazon DynamoDB dicapai dengan menyimpan pengenal penyewa di kunci partisi setiap tabel Amazon DynamoDB yang mengelola data untuk semua penyewa.

Anda dapat membayangkan bagaimana hal ini dapat bervariasi di berbagai AWS layanan, dengan masing-masing memperkenalkan konstruksinya sendiri yang mungkin memerlukan pendekatan berbeda untuk mewujudkan model penyimpanan silo dan gabungan dengan setiap layanan.

Sementara partisi data dan isolasi penyewa adalah topik yang terpisah, strategi partisi data yang Anda pilih cenderung dipengaruhi oleh model isolasi data Anda. Misalnya, Anda mungkin silo beberapa penyimpanan karena pendekatan yang paling sesuai dengan persyaratan domain atau pelanggan Anda. Atau, Anda dapat memilih silo karena model kolam mungkin tidak memungkinkan Anda untuk menegakkan isolasi dengan tingkat granularitas yang diperlukan solusi Anda.

Tetangga yang bising juga dapat memengaruhi pendekatan Anda terhadap isolasi. Beberapa beban kerja atau kasus penggunaan dalam aplikasi Anda mungkin perlu dipisahkan untuk membatasi dampak dari penyewa lain atau untuk memenuhi perjanjian tingkat layanan (SLA).

Pengukuran, metrik, dan penagihan

Diskusi SaaS juga cenderung mencakup gagasan pengukuran, metrik, dan penagihan. Konsep-konsep ini sering dilipat menjadi satu konsep. Namun, penting untuk membedakan peran berbeda yang dimainkan pengukuran, metrik, dan penagihan di lingkungan SaaS.

Tantangan dari konsep-konsep ini adalah bahwa mereka sering memiliki tumpang tindih penggunaan kata yang sama. Misalnya, kita dapat berbicara tentang pengukuran yang digunakan untuk menghasilkan tagihan Anda. Pada saat yang sama, kita juga dapat berbicara tentang pengukuran yang digunakan untuk melacak konsumsi internal sumber daya yang tidak terhubung ke penagihan. Kami juga berbicara tentang metrik dan SaaS dalam banyak konteks yang dapat dicampur ke dalam diskusi ini.

Untuk membantu menyelesaikan masalah ini, mari kita kaitkan beberapa konsep spesifik dengan masing-masing istilah ini (mengetahui tidak ada yang absolut di sini).

- Pengukuran - Konsep ini, meskipun memiliki banyak definisi, paling cocok di domain penagihan SaaS. Idenya adalah bahwa Anda mengukur aktivitas penyewa atau konsumsi sumber daya untuk mengumpulkan data yang dibutuhkan untuk menghasilkan tagihan.
- Metrik — Metrik mewakili semua data yang Anda ambil untuk menganalisis tren di seluruh domain bisnis, operasi, dan teknologi Anda. Data ini digunakan untuk melintasi banyak konteks dan peran dalam tim SaaS.

Perbedaan ini tidak penting, tetapi membantu menyederhanakan cara kita berpikir tentang peran pengukuran dan metrik dalam lingkungan SaaS.

Sekarang, jika kita menghubungkan kedua konsep ini ke contoh, Anda dapat memikirkan instrumenting aplikasi Anda dengan peristiwa pengukuran tertentu yang digunakan untuk menampilkan data yang diperlukan untuk menghasilkan tagihan. Ini bisa berupa jumlah permintaan, jumlah pengguna aktif, atau bisa memetakan ke beberapa agregat konsumsi (permintaan, CPU, memori) yang berkorelasi dengan beberapa unit yang masuk akal bagi pelanggan Anda.

Di lingkungan SaaS Anda, Anda akan mempublikasikan peristiwa penagihan ini dari aplikasi Anda dan mereka akan dicerna dan diterapkan oleh konstruksi penagihan yang digunakan oleh sistem SaaS Anda. Ini bisa berupa sistem penagihan pihak ketiga atau sesuatu yang khusus.

Sebaliknya, pola pikir di balik metrik adalah menangkap tindakan, aktivitas, pola konsumsi, dan sebagainya yang penting untuk mengevaluasi jejak kesehatan dan operasional yang dikenakan

pada sistem Anda oleh penyewa yang berbeda. Metrik yang Anda publikasikan dan agregat di sini lebih ditentukan oleh kebutuhan persona yang berbeda (tim operasional, pemilik produk, arsitek, dan sebagainya). Di sini, data metrik ini dipublikasikan dan digabungkan ke dalam beberapa perkakas analitik yang memungkinkan pengguna berbeda ini membangun tampilan aktivitas sistem yang menganalisis aspek sistem yang paling sesuai dengan persona mereka. Seorang pemilik produk mungkin ingin memahami bagaimana penyewa yang berbeda mengkonsumsi fitur. Seorang arsitek mungkin membutuhkan pandangan yang membantu mereka memahami bagaimana penyewa mengkonsumsi sumber daya infrastruktur, dan sebagainya.

SaaS B2B dan B2C

Penawaran SaaS dibuat untuk pasar B2B dan B2C. Sementara pasar dan pelanggan tentu memiliki dinamika yang berbeda, prinsip keseluruhan SaaS entah bagaimana tidak mengubah masing-masing pasar ini.

Jika Anda melihat orientasi, misalnya, pelanggan B2B dan B2C mungkin memiliki pengalaman orientasi yang berbeda. Memang benar bahwa sistem B2C mungkin lebih fokus pada aliran orientasi swalayan (meskipun sistem B2B dapat mendukung ini juga).

Meskipun mungkin ada perbedaan dalam bagaimana orientasi ditawarkan kepada pelanggan, nilai dasar yang mendasari untuk orientasi sebagian besar sama. Bahkan jika solusi B2B Anda bergantung pada proses orientasi internal, kami masih berharap proses itu menjadi tanpa gesekan dan seotomatis mungkin. Menjadi B2B entah bagaimana tidak berarti bahwa kami mengubah harapan kami di sekitar waktu untuk menghargai pelanggan kami.

Kesimpulan

Tujuan dari dokumen ini adalah untuk menguraikan konsep arsitektur SaaS yang mendasar, memberikan beberapa klarifikasi seputar model dan terminologi yang digunakan untuk mengkarakterisasi pola dan strategi SaaS. Harapannya adalah bahwa ini akan melengkapi organisasi dengan pandangan yang lebih jelas tentang lanskap SaaS secara keseluruhan.

Sebagian besar dari apa yang dibahas di sini berfokus pada apa artinya menjadi SaaS, dengan penekanan signifikan pada menciptakan lingkungan yang memungkinkan Anda mengelola dan mengoperasikan semua penyewa SaaS Anda melalui pengalaman terpadu. Ini menghubungkan ke gagasan inti bahwa SaaS adalah model bisnis pertama dan terutama. Arsitektur SaaS yang Anda bangun dimaksudkan untuk mempromosikan tujuan bisnis dasar ini.

Bacaan lebih lanjut

Ada sejumlah sumber daya yang lebih detail seputar pola arsitektur SaaS yang sesuai dengan pola yang dijelaskan di sini.

Untuk informasi tambahan, lihat:

- Strategi Isolasi Penyewa SaaS (whitepaper) AWS
- Strategi Penyimpanan SaaS (AWSwhitepaper)
- Lensa SaaS yang Didesain dengan Baik (whitepaper) AWS

Kontributor

Individu dan organisasi berikut berkontribusi terhadap dokumen ini:

- Tod Golding, Arsitek Solusi Mitra Utama, Pabrik SaaS AWS

Revisi dokumen

Untuk pemberitahuan tentang pembaruan laporan ini, berlangganan ke umpan RSS.

Perubahan	Deskripsi	Tanggal
Publikasi awal	Whitepaper diterbitkan.	3

Pemberitahuan

Pelanggan bertanggung jawab untuk membuat penilaian independen mereka sendiri terhadap informasi dalam dokumen ini. Dokumen ini: (a) hanya untuk tujuan informasi, (b) mewakili penawaran dan praktik AWS produk saat ini, yang dapat berubah tanpa pemberitahuan, dan (c) tidak menciptakan komitmen atau jaminan apa pun dari AWS dan afiliasinya, pemasok, atau pemberi lisensinya. AWSproduk atau layanan disediakan “sebagaimana adanya” tanpa jaminan, pernyataan, atau kondisi apa pun, baik tersurat maupun tersirat. Tanggung jawab dan kewajiban AWS kepada pelanggannya dikendalikan oleh AWS perjanjian, dan dokumen ini bukan bagian dari, juga tidak memodifikasi, perjanjian apa pun antara AWS dan pelanggannya.

AWSGlosarium

Untuk AWS terminologi terbaru, lihat AWSglosarium di Referensi. Glosarium AWS

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