aws

Laporan Resmi AWS

Arsitektur Multi-Tingkat Nirserver AWS menggunakan Amazon API Gateway dan AWS Lambda

Arsitektur Multi-Tingkat Nirserver AWS menggunakan Amazon API Gateway dan AWS Lambda : Laporan Resmi AWS

Table of Contents

Abstrak 1
Abstrak 1
Pengantar 2
Gambaran umum arsitektur tiga tingkat 4
Tingkat logika nirserver 5
AWS Lambda 5
Logika bisnis Anda ada di sini, server tidak diperlukan 6
Keamanan Lambda 6
Performa dalam skala besar 7
Deployment dan manajemen nirserver 7
Amazon API Gateway 9
Integrasi dengan AWS Lambda 9
Performa API yang stabil di Seluruh Wilayah 10
Dorong inovasi dan kurangi overhead dengan fitur bawaan 10
Iterasi cepat, tetap tangkas 11
Tingkat data 14
Opsi penyimpanan data nirserver 14
Opsi penyimpanan data non-nirserver 15
Tingkat presentasi 17
Sampel pola arsitektur 19
Backend seluler 20
Aplikasi halaman tunggal 21
Aplikasi web 23
Layanan Mikro dengan Lambda 25
Kesimpulan 27
Kontributor 28
Bacaan Lebih Lanjut 29
Revisi dokumen 30
Pemberitahuan 31

Arsitektur Multi-Tingkat Nirserver AWS menggunakan Amazon API Gateway dan AWS Lambda

Tanggal publikasi: 20 Oktober 2021 (Revisi dokumen)

Abstrak

Laporan resmi ini menggambarkan bagaimana inovasi dari Amazon Web Services (AWS) dapat digunakan untuk mengubah cara Anda merancang arsitektur multi-tingkat serta menerapkan pola populer seperti layanan mikro, backend seluler, dan aplikasi halaman tunggal. Arsitek dan developer dapat menggunakan Amazon API Gateway, AWS Lambda, dan layanan lainnya untuk mengurangi siklus pengembangan dan operasi yang diperlukan untuk membuat dan mengelola aplikasi multi-tingkat.

Pengantar

Aplikasi multi-tingkat (tiga tingkat, n tingkat, dan seterusnya.) telah menjadi fondasi pola arsitektur selama beberapa dekade, dan tetap menjadi pola populer untuk aplikasi yang berorientasi pada pengguna. Meskipun bahasa yang digunakan untuk menggambarkan arsitektur multi-tingkat beragam, aplikasi multi-tingkat umumnya terdiri dari komponen-komponen berikut:

- Tingkat presentasi: Komponen yang berinteraksi langsung dengan pengguna (misalnya, halaman web dan UI aplikasi seluler).
- Tingkat logika: Kode yang diperlukan untuk menerjemahkan tindakan pengguna ke fungsionalitas aplikasi (misalnya, operasi basis data CRUD dan pemrosesan data).
- Tingkat data: Media penyimpanan (misalnya, basis data, penyimpanan objek, cache, dan sistem file) yang menyimpan data yang relevan ke aplikasi tersebut.

Pola arsitektur multi-tingkat menyediakan kerangka kerja umum untuk memastikan komponen aplikasi dapat di-decouple dan dapat diskalakan secara independen, dapat dikembangkan, dikelola, serta dipelihara secara terpisah (sering kali oleh tim yang berbeda-beda).

Sebagai konsekuensi dari pola ini yang memerlukan jaringan (sebuah tingkat harus membuat panggilan jaringan untuk berinteraksi dengan tingkat lain) untuk bertindak sebagai batas di antara tingkatan, mengembangkan aplikasi multi-tingkat sering membutuhkan pembuatan banyak komponen aplikasi yang tidak terdiferensiasi. Beberapa komponen ini meliputi:

- Kode yang mendefinisikan antrean pesan untuk komunikasi di antara tingkat
- Kode yang mendefinisikan antarmuka pemrograman aplikasi (API) dan model data
- Kode terkait keamanan yang memastikan akses yang tepat ke aplikasi tersebut

Semua contoh ini dapat dianggap komponen “boilerplate” yang, meskipun diperlukan dalam aplikasi multi-tingkat, tetapi tidak sangat bervariasi dalam penerapannya dari satu aplikasi ke aplikasi berikutnya.

AWS menawarkan sejumlah layanan yang memungkinkan pembuatan aplikasi multi-tingkat nirserver – yang sangat menyederhanakan proses deployment aplikasi tersebut ke produksi dan meniadakan overhead terkait manajemen server tradisional. Amazon API Gateway, layanan untuk membuat dan mengelola API, dan AWS Lambda, layanan untuk menjalankan fungsi kode arbitrer, dapat digunakan bersama untuk menyederhanakan pembuatan aplikasi multi-tingkat yang canggih.

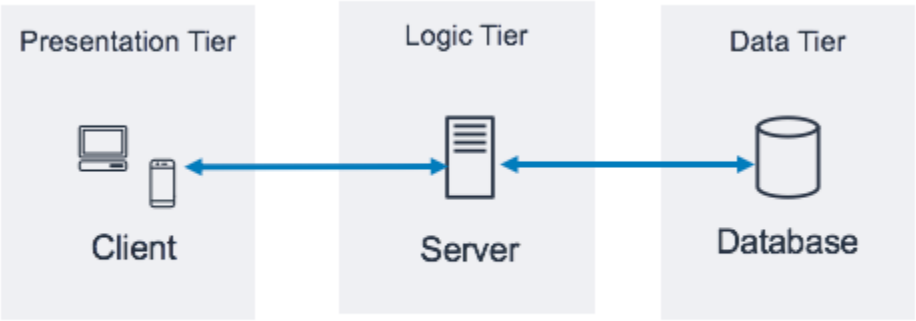
Integrasi Amazon API Gateway dengan AWS Lambda memungkinkan fungsi kode yang ditetapkan pengguna diinisiasi secara langsung melalui permintaan HTTPS. Terlepas dari volume permintaan, baik API Gateway dan Lambda menskalakan secara otomatis untuk mendukung kebutuhan aplikasi Anda secara tepat (lihat kuota dan catatan penting Amazon API Gateway untuk informasi skalabilitas). Dengan menggabungkan kedua layanan ini, Anda dapat membuat tingkat yang memungkinkan Anda menulis hanya kode yang penting untuk aplikasi Anda dan tidak fokus pada berbagai aspek lain yang tidak terdiferensiasi dalam menerapkan arsitektur multi-tingkat seperti arsitek untuk ketersediaan tinggi, menulis SDK klien, manajemen server dan sistem operasi (OS), penskalaan, serta menerapkan mekanisme otorisasi klien.

API Gateway dan Lambda memungkinkan pembuatan tingkat logika nirserver. Tergantung pada persyaratan aplikasi Anda, AWS juga menyediakan opsi untuk membuat tingkat presentasi nirserver (misalnya, dengan Amazon CloudFront dan Amazon Simple Storage Service) dan tingkat data (misalnya, Amazon Aurora, Amazon DynamoDB).

Laporan resmi ini berfokus pada contoh arsitektur multi-tingkat, aplikasi web tiga tingkat. Namun, Anda dapat menerapkan pola multi-tingkat ini di luar aplikasi web tiga tingkat yang standar.

Gambaran umum arsitektur tiga tingkat

Arsitektur tiga tingkat adalah implementasi paling populer dari arsitektur multi-tingkat dan terdiri dari satu tingkat presentasi, tingkat logika, dan tingkat data. Ilustrasi berikut menunjukkan contoh aplikasi tiga tingkat sederhana dan umum.



Pola arsitektur untuk aplikasi tiga tingkat

Ada banyak sumber daya online yang tepat yang memungkinkan Anda mempelajari lebih lanjut tentang pola arsitektur tiga tingkat umum . Laporan resmi ini berfokus pada pola implementasi spesifik untuk arsitektur ini menggunakan Amazon API Gateway dan AWS Lambda.

Tingkat logika nirserver

Tingkat logika untuk arsitektur tiga tingkat mewakili inti aplikasi. Oleh karena itu, menggunakan Amazon API Gateway dan AWS Lambda dapat memberikan dampak paling besar dibandingkan dengan implementasi berbasis server yang tradisional. Fitur dari kedua layanan ini memungkinkan Anda membangun aplikasi nirserver yang sangat tersedia, dapat diskalakan, dan aman. Dalam model tradisional, aplikasi Anda dapat memerlukan ribuan server; namun, menggunakan Amazon API Gateway dan AWS Lambda, Anda tidak bertanggung jawab atas manajemen server dalam kapasitas apa pun. Selain itu, menggunakan layanan terkelola ini bersama-sama, Anda mendapatkan manfaat sebagai berikut:

- AWS Lambda:
  - Tidak ada OS yang perlu dipilih, diamankan, di-patch, atau dikelola
  - Tidak ada server yang perlu disesuaikan ukurannya, dipantau, atau diskalakan secara tepat
  - Mengurangi risiko biaya Anda dari penyediaan yang berlebih
  - Mengurangi risiko terhadap performa Anda karena penyediaan yang tidak memadai

- Amazon API Gateway:
  - Mekanisme yang disederhanakan untuk men-deploy, memantau, dan mengamankan API
  - Peningkatan performa API melalui caching dan pengiriman konten

AWS Lambda

AWS Lambda adalah layanan komputasi yang memungkinkan Anda menjalankan fungsi kode arbitrer dalam bahasa yang didukung (Node.js, Python, Ruby, Java, Go, NET, untuk informasi lebih lanjut, lihat Pertanyaan yang Sering Diajukan terkait Lambda) tanpa penyediaan, pengelolaan, atau penskalaan server. Fungsi Lambda dijalankan dalam kontainer yang dikelola dan terisolasi, dan diluncurkan sebagai respons terhadap suatu peristiwa yang dapat menjadi salah satu dari beberapa pemicu terprogram yang disediakan AWS, yang disebut sumber peristiwa. Lihat Pertanyaan yang Sering Diajukan terkait Lambda untuk mengetahui semua sumber peristiwa.

Banyak kasus penggunaan populer untuk Lambda berkisar seputar alur kerja pemrosesan data berbasis peristiwa, seperti file pemrosesan yang disimpan di Amazon S3 atau catatan data streaming dari Amazon Kinesis. Saat digunakan bersama dengan Amazon API Gateway, fungsi Lambda melakukan fungsionalitas layanan web standar: fungsi tersebut menginisiasi kode sebagai respons

terhadap permintaan HTTPS klien; API Gateway akan berfungsi sebagai pintu depan untuk tingkat logika Anda, dan AWS Lambda akan memanggil kode aplikasi.

Logika bisnis Anda ada di sini, server tidak diperlukan

Lambda mengharuskan Anda untuk menulis fungsi kode, yang disebut handler, yang akan berjalan saat diinisiasi oleh suatu peristiwa. Untuk menggunakan Lambda dengan API Gateway, Anda dapat mengonfigurasi API Gateway untuk meluncurkan fungsi handler saat permintaan HTTPS ke API Anda terjadi. Dalam arsitektur multi-tingkat nirserver, masing-masing API yang Anda buat di API Gateway akan berintegrasi dengan fungsi Lambda (dan handler di dalamnya) yang memanggil logika bisnis yang diperlukan.

Menggunakan fungsi AWS Lambda untuk menyusun tingkat logika memungkinkan Anda menentukan tingkat granularitas yang diinginkan untuk mengekspos fungsionalitas aplikasi (satu fungsi Lambda per API atau satu fungsi Lambda per metode API). Di dalam fungsi Lambda, handler dapat menjangkau dependensi lain (misalnya, metode lain yang telah Anda unggah dengan kode, pustaka, biner native, dan layanan web eksternal), atau bahkan fungsi Lambda lainnya.

Membuat atau memperbarui fungsi Lambda memerlukan pengunggahan kode sebagai paket deployment Lambda dalam file zip ke bucket Amazon S3, atau kode paket sebagai image kontainer bersama dengan semua dependensi. Fungsi ini dapat menggunakan metode deployment yang berbeda, seperti AWS Management Console, menjalankan AWS Command Line Interface (AWS CLI), atau menjalankan infrastruktur sebagai templat kode atau kerangka kerja seperti AWS CloudFormation, AWS Serverless Application Model (AWS SAM), atau AWS Cloud Development Kit (AWS CDK). Saat Anda membuat fungsi menggunakan salah satu metode ini, Anda menentukan metode mana di dalam paket deployment Anda yang akan berfungsi sebagai handler permintaan. Anda dapat menggunakan kembali paket deployment yang sama untuk beberapa definisi fungsi Lambda, yang memungkinkan setiap fungsi Lambda memiliki handler unik dalam paket deployment yang sama.

Keamanan Lambda

Untuk menjalankan fungsi Lambda, fungsi tersebut harus dipanggil oleh peristiwa atau layanan yang diizinkan oleh kebijakan AWS Identity and Access Management (IAM). Menggunakan kebijakan IAM, Anda dapat membuat fungsi Lambda yang tidak dapat diinisiasi sama sekali kecuali jika dipanggil oleh sumber daya API Gateway yang Anda tetapkan. Kebijakan tersebut dapat didefinisikan menggunakan kebijakan berbasis sumber daya di berbagai layanan AWS.

Setiap fungsi Lambda mengambil IAM role yang ditetapkan saat fungsi Lambda di-deploy. IAM role ini mendefinisikan layanan dan sumber daya AWS lain yang dapat berinteraksi dengan fungsi Lambda Anda (misalnya, Amazon DynamoDB Amazon S3). Dalam konteks fungsi Lambda, hal ini disebut peran eksekusi.

Jangan menyimpan informasi yang sensitif di dalam fungsi Lambda. IAM menangani akses ke layanan AWS melalui peran eksekusi Lambda; jika Anda perlu mengakses kredensial lainnya (misalnya, kredensial basis data dan Kunci API) dari dalam fungsi Lambda Anda, Anda dapat menggunakan AWS Key Management Service (AWS KMS) dengan variabel lingkungan, atau menggunakan layanan seperti AWS Secrets Manager untuk menjaga informasi ini aman saat tidak digunakan.

Performa dalam skala besar

Kode ditarik sebagai image kontainer dari Amazon Elastic Container Registry (Amazon ECR), atau dari file zip yang diunggah ke Amazon S3, yang berjalan di lingkungan terisolasi yang dikelola AWS. Anda tidak perlu menskalakan fungsi Lambda Anda—setiap kali notifikasi peristiwa diterima oleh fungsi Anda, AWS Lambda mencari kapasitas yang tersedia dalam armada komputasi dan menjalankan kode Anda dengan konfigurasi runtime, memori, disk, dan batas waktu yang Anda tetapkan. Dengan pola ini, AWS dapat memulai sebanyak mungkin salinan fungsi Anda sesuai kebutuhan.

Tingkat logika berbasis Lambda selalu disesuaikan ukurannya secara tepat untuk kebutuhan pelanggan Anda. Kemampuan untuk menyerap lonjakan lalu lintas dengan cepat melalui penskalaan terkelola dan inisiasi kode konkuren, yang dikombinasikan dengan harga bayar sesuai penggunaan Lambda, memungkinkan Anda selalu memenuhi permintaan pelanggan tanpa perlu membayar kapasitas komputasi yang tidak terpakai.

Deployment dan manajemen nirserver

Untuk membantu Anda men-deploy dan mengelola fungsi Lambda, gunakan AWS Serverless Application Model (AWS SAM), kerangka kerja sumber terbuka yang mencakup:

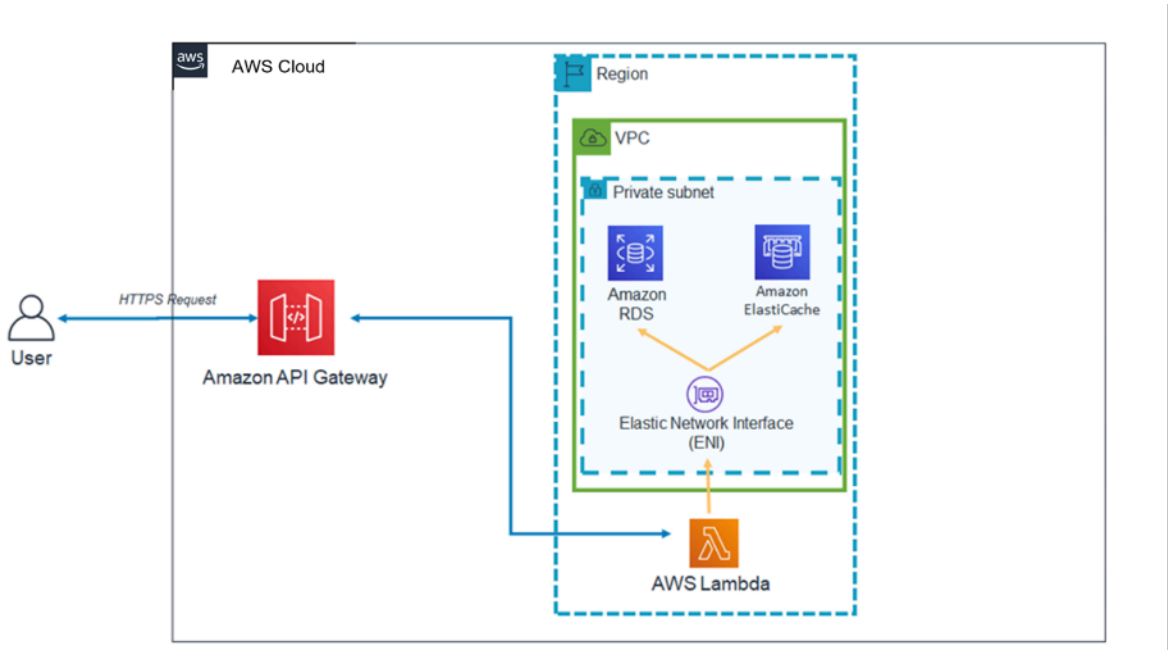
- Spesifikasi templat AWS SAM - Sintaks yang digunakan untuk menentukan fungsi Anda dan menggambarkan lingkungan, izin, konfigurasi, dan peristiwa untuk pengunggahan dan deployment yang disederhanakan.

- AWS SAM CLI - Perintah yang memungkinkan Anda memverifikasi sintaks templat SAM, memanggil fungsi secara lokal, melakukan debug pada fungsi Lambda, dan fungsi paket deployment.

Anda juga dapat menggunakan AWS CDK, yang merupakan kerangka kerja pengembangan perangkat lunak untuk mendefinisikan infrastruktur cloud menggunakan bahasa pemrograman dan menyediakannya melalui CloudFormation. CDK menyediakan cara imperatif untuk menentukan sumber daya AWS, sedangkan AWS SAM menyediakan cara deklaratif.

Biasanya, saat Anda menerapkan fungsi Lambda, fungsi tersebut dipanggil dengan izin yang ditentukan oleh IAM role yang ditetapkan, dan mampu mencapai titik akhir yang terhubung ke internet. Sebagai inti dari tingkat logika Anda, AWS Lambda adalah komponen yang secara langsung berintegrasi dengan tingkat data. Jika tingkat data Anda berisi informasi bisnis atau pengguna yang sensitif, penting untuk memastikan bahwa tingkat data ini diisolasi dengan tepat (dalam subnet privat).

Anda dapat mengonfigurasi fungsi Lambda untuk terhubung ke subnet privat di virtual private cloud (VPC) di akun AWS Anda jika Anda ingin fungsi Lambda mengakses sumber daya yang tidak dapat Anda ekspos secara publik, seperti instans basis data privat. Saat Anda menghubungkan fungsi ke VPC, layanan ini membuat antarmuka jaringan elastis untuk setiap subnet dalam konfigurasi VPC fungsi Anda dan antarmuka jaringan elastis digunakan untuk mengakses sumber daya internal Anda secara privat.



Pola arsitektur Lambda di dalam VPC

Penggunaan Lambda dengan VPC berarti bahwa basis data dan media penyimpanan lain yang diandalkan oleh logika bisnis Anda dapat diatur agar tidak dapat diakses melalui internet. VPC juga memastikan bahwa satu-satunya cara untuk berinteraksi dengan data Anda dari internet adalah melalui API yang telah Anda tetapkan dan fungsi kode Lambda yang telah Anda tulis.

Amazon API Gateway

Amazon API Gateway adalah layanan terkelola penuh yang memungkinkan developer membuat, memublikasikan, memelihara, memantau, dan mengamankan API pada skala apa pun.

Klien (yaitu, tingkat presentasi) berintegrasi dengan API yang diekspos melalui API Gateway menggunakan permintaan HTTPS standar. Kemampuan penerapan API yang diekspos melalui API Gateway ke arsitektur multi-tingkat yang berorientasi layanan adalah kemampuan untuk memisahkan bagian-bagian fungsionalitas aplikasi individu dan mengekspos fungsionalitas ini melalui titik akhir REST. Amazon API Gateway memiliki fitur dan kualitas khusus yang dapat menambahkan kemampuan yang canggih ke tingkat logika Anda.

Integrasi dengan AWS Lambda

Amazon API Gateway mendukung jenis API REST dan HTTP. API untuk layanan API Gateway terdiri dari sumber daya dan metode. Sumber daya adalah entitas logika yang dapat diakses oleh aplikasi melalui jalur sumber daya (misalnya, `/tickets`). Sebuah metode akan sesuai dengan permintaan API yang dikirimkan ke sumber daya API (misalnya, `GET /tickets`). API Gateway memungkinkan Anda mendukung setiap metode dengan fungsi Lambda, yaitu, saat Anda memanggil API melalui titik akhir HTTPS yang terekspos di API Gateway, API Gateway akan memanggil fungsi Lambda.

Anda dapat menghubungkan fungsi API Gateway dan Lambda menggunakan integrasi proksi dan integrasi non-proksi.

Integrasi proksi

Dalam integrasi proksi, seluruh permintaan HTTPS klien dikirim apa adanya ke fungsi Lambda. API Gateway meneruskan seluruh permintaan klien sebagai parameter peristiwa untuk fungsi handler Lambda, dan output dari fungsi Lambda dikembalikan langsung ke klien (termasuk kode status, header, dan sebagainya.).

Integrasi non-proksi

Dalam integrasi non-proksi, Anda mengonfigurasi bagaimana parameter, header, dan isi permintaan klien diteruskan ke parameter peristiwa fungsi handler Lambda. Selain itu, Anda mengonfigurasi bagaimana output Lambda diterjemahkan balik ke pengguna.

> **Note**
>
> API Gateway juga dapat mengirimkan proksi ke sumber daya nirserver tambahan di luar AWS Lambda, seperti integrasi sementara (berguna untuk pengembangan aplikasi awal), dan proksi langsung ke objek S3.

Performa API yang stabil di seluruh wilayah

Setiap deployment Amazon API Gateway mencakup distribusi Amazon CloudFront yang dilakukan di latar belakang. CloudFront adalah layanan pengiriman konten yang menggunakan jaringan global lokasi edge Amazon sebagai titik koneksi untuk klien yang menggunakan API Anda. CloudFront membantu mengurangi latensi respons untuk API Anda. Menggunakan beberapa lokasi edge di seluruh dunia, Amazon CloudFront juga menyediakan kemampuan untuk memerangi skenario serangan Distributed Denial of Service (DDoS). Untuk informasi selengkapnya, lihat laporan resmi Praktik Terbaik AWS untuk Ketahanan DDoS.

Anda dapat meningkatkan performa permintaan API tertentu menggunakan API Gateway untuk menyimpan respons di cache dalam memori opsional. Pendekatan ini tidak hanya memberikan manfaat performa untuk permintaan API berulang, tetapi juga mengurangi berapa kali fungsi Lambda Anda dipanggil, yang dapat mengurangi biaya keseluruhan Anda.

Dorong inovasi dan kurangi overhead dengan fitur bawaan

Biaya pengembangan untuk membangun aplikasi baru adalah investasi. Menggunakan API Gateway dapat mengurangi jumlah waktu yang diperlukan untuk tugas pengembangan tertentu dan menurunkan total biaya pengembangan, sehingga memungkinkan organisasi lebih bebas bereksperimen dan berinovasi.

Selama fase pengembangan aplikasi awal, implementasi pencatatan log dan pengumpulan metrik sering kali diabaikan untuk mengirimkan aplikasi baru secara lebih cepat. Hal ini dapat menyebabkan utang teknis dan risiko operasional saat men-deploy fitur ini ke aplikasi yang berjalan dalam produksi.

Amazon API Gateway terintegrasi secara lancar dengan Amazon CloudWatch, yang mengumpulkan dan memproses data mentah dari API Gateway menjadi metrik hampir waktu nyata yang dapat dibaca untuk memantau eksekusi API. API Gateway juga mendukung pencatatan log akses dengan laporan yang dapat dikonfigurasi, dan penelusuran AWS X-Ray untuk debugging. Masing-masing fitur ini tidak memerlukan penulisan kode, dan dapat disesuaikan dalam aplikasi yang berjalan dalam produksi tanpa risiko terhadap logika bisnis inti.

Masa pakai keseluruhan aplikasi mungkin tidak diketahui, atau mungkin masa pakainya singkat. Membuat kasus bisnis untuk membangun aplikasi semacam itu dapat dibuat lebih mudah jika titik awal Anda sudah menyertakan fitur terkelola yang disediakan API Gateway, dan jika Anda hanya menimbulkan biaya infrastruktur setelah API Anda mulai menerima permintaan. Untuk informasi lebih lanjut, lihat harga Amazon API Gateway.

Iterasi cepat, tetap tangkas

Menggunakan Amazon API Gateway dan AWS Lambda untuk membangun tingkat logika API Anda memungkinkan Anda beradaptasi secara cepat dengan perubahan permintaan dari basis pengguna Anda melalui penyederhanaan deloyment API dan manajemen versi.

Deployment tahap

Saat men-deploy API di API Gateway, Anda harus mengaitkan deployment dengan tahap API Gateway – setiap tahap adalah snapshot API dan tersedia untuk dipanggil aplikasi klien. Dengan menggunakan konvensi ini, Anda dapat dengan mudah men-deploy aplikasi ke tahap pengembangan, pengujian, penahapan, atau produksi, dan memindahkan deployment di antara tahap ini. Setiap kali Anda men-deploy API Anda ke suatu tahap, Anda membuat versi berbeda dari API yang dapat dikembalikan jika diperlukan. Fitur-fitur ini memungkinkan fungsionalitas dan dependensi klien yang ada untuk terus berjalan tanpa terganggu sementara fungsionalitas baru dirilis sebagai versi API terpisah.

Integrasi yang di-decouple dengan Lambda

Integrasi antara API di API Gateway dan fungsi Lambda dapat di-decouple menggunakan variabel tahap API Gateway dan alias fungsi Lambda. Hal ini menyederhanakan dan mempercepat deployment API. Dibandingkan dengan mengonfigurasi nama fungsi Lambda atau alias dalam API secara langsung, Anda dapat mengonfigurasi variabel tahap dalam API yang dapat menunjuk ke alias tertentu dalam fungsi Lambda. Selama deployment, ubah nilai variabel tahap agar menunjuk ke alias fungsi Lambda lalu API akan menjalankan versi fungsi Lambda di belakang alias Lambda untuk tahap tertentu.

Deployment peluncuran canary

Peluncuran canary adalah strategi pengembangan perangkat lunak yang menggunakan versi baru API untuk tujuan pengujian, dan versi dasar tetap digunakan sebagai rilis produksi untuk operasi normal pada tahap yang sama. Dalam deployment peluncuran canary, lalu lintas API total dipisahkan secara acak menjadi peluncuran produksi dan peluncuran canary dengan rasio yang telah dikonfigurasi sebelumnya. API di API Gateway dapat dikonfigurasi untuk penyebaran rilis canary untuk menguji fitur baru dengan seperangkat pengguna terbatas.

Nama domain kustom

Anda dapat memberikan nama URL intuitif yang mudah digunakan bisnis ke API, alih-alih URL yang disediakan oleh API Gateway. API Gateway menyediakan fitur untuk mengonfigurasi domain kustom untuk API. Dengan nama domain kustom, Anda dapat menyiapkan nama host API Anda, dan memilih jalur dasar multi-level (misalnya, `myservicemyservice/cat/v1`, atau `myservice/dog/v2`) untuk memetakan URL alternatif ke API Anda.

Prioritaskan keamanan API

Semua aplikasi harus memastikan bahwa hanya klien sah yang memiliki akses ke sumber daya API mereka. Saat merancang aplikasi multi-tingkat, Anda dapat memanfaatkan beberapa cara berbeda agar Amazon API Gateway berkontribusi untuk mengamankan tingkat logika Anda:

Keamanan transit

Semua permintaan ke API Anda dapat dilakukan melalui HTTPS untuk mengaktifkan enkripsi secara in transit.

API Gateway menyediakan Sertifikat SSL/TLS bawaan – jika menggunakan opsi nama domain kustom untuk API yang menghadap publik, Anda dapat memberikan sertifikat SSL/TLS Anda sendiri menggunakan AWS Certificate Manager. API Gateway juga mendukung autentikasi TLS (MTL) mutual. TLS mutual meningkatkan keamanan API Anda dan membantu melindungi data Anda dari serangan seperti spoofing klien atau serangan man-in-the middle.

Otorisasi AP

Setiap kombinasi sumber daya/metode yang Anda buat sebagai bagian dari API Anda akan diberikan Amazon Resource Name (ARN) unik yang dapat direferensikan dalam kebijakan AWS Identity and Access Management (IAM).

Ada tiga metode umum untuk menambahkan otorisasi ke API di API Gateway:

- IAM Role dan Kebijakan IAM: Klien menggunakan otorisasi AWS Signature Version 4 (Sigv4) dan kebijakan IAM untuk akses API. Kredensial yang sama dapat membatasi atau mengizinkan akses ke layanan dan sumber daya AWS lain sesuai kebutuhan (misalnya, bucket Amazon S3 atau tabel Amazon DynamoDB).
- Pool pengguna Amazon Cognito: Klien masuk melalui pool pengguna Amazon Cognito dan mendapatkan token, yang disertakan dalam header otorisasi permintaan.
- Pengotorisasi Lambda: Tentukan fungsi Lambda yang mengimplementasikan skema otorisasi kustom yang menggunakan strategi token bearer (misalnya, OAuth dan SAML) atau menggunakan parameter permintaan untuk mengidentifikasi pengguna.

Pembatasan akses

API Gateway mendukung pembuatan kunci API dan pengaitan kunci ini dengan rencana penggunaan yang dapat dikonfigurasi. Anda dapat memantau penggunaan kunci API dengan CloudWatch.

API Gateway mendukung throttling, batas laju, dan batas laju burst untuk setiap metode di API Anda.

API Privat

Menggunakan API Gateway, Anda dapat membuat REST API privat yang hanya dapat diakses dari virtual private cloud Anda di Amazon VPC dengan menggunakan titik akhir VPC antarmuka. Ini adalah antarmuka jaringan titik akhir yang Anda buat di VPC Anda.

Dengan menggunakan kebijakan sumber daya, Anda dapat mengizinkan atau menolak akses ke API Anda dari VPC dan titik akhir VPC yang dipilih, termasuk di seluruh akun AWS. Setiap titik akhir dapat digunakan untuk mengakses beberapa API privat. Anda juga dapat menggunakan AWS Direct Connect untuk membuat koneksi dari jaringan on-premise ke Amazon VPC dan mengakses API privat Anda melalui koneksi tersebut.

Dalam semua kasus, lalu lintas ke API privat Anda akan menggunakan koneksi aman dan tidak meninggalkan jaringan Amazon—lalu lintas tersebut diisolasi dari internet publik.

Perlindungan firewall menggunakan AWS WAF

API yang dapat diakses melalui internet rentan terhadap serangan berbahaya. AWS WAF adalah firewall aplikasi web yang membantu melindungi API dari serangan tersebut. Layanan ini melindungi API dari eksploitasi web umum seperti injeksi SQL dan serangan pembuatan skrip lintas situs. Anda dapat menggunakan AWS WAF API Gateway untuk membantu melindungi API.

Tingkat data

Menggunakan AWS Lambda sebagai tingkat logika Anda tidak membatasi opsi penyimpanan data yang tersedia di tingkat data Anda. Fungsi Lambda terhubung ke opsi penyimpanan data apa pun dengan menyertakan driver basis data yang sesuai dalam paket deployment Lambda, dan menggunakan akses berbasis IAM role atau kredensial terenkripsi (melalui AWS KMS atau AWS Secrets Manager).

Memilih penyimpanan data untuk aplikasi Anda sangat tergantung pada persyaratan aplikasi Anda. AWS menawarkan sejumlah penyimpanan data nirserver dan non-nirserver yang dapat Anda gunakan untuk menyusun tingkat data aplikasi Anda.

Opsi penyimpanan data nirserver

Amazon S3 adalah layanan penyimpanan objek yang menawarkan skalabilitas, ketersediaan data, keamanan, dan performa yang terdepan dalam industri.

Amazon Aurora adalah basis data relasional yang kompatibel dengan MySQL dan PostgreSQL yang dibangun untuk cloud, yang menggabungkan performa dan ketersediaan basis data korporasi tradisional dengan sistem yang sederhana dan basis data sumber terbuka yang hemat biaya. Aurora menawarkan model penggunaan nirserver dan tradisional.

Amazon DynamoDB adalah basis data kunci-nilai dan dokumen yang memberikan performa milidetik satu digit dalam skala apa pun. Ini adalah basis data terkelola penuh, nirserver, multi-wilayah, multi-master, berdaya tahan dengan keamanan, pencadangan, dan pemulihan bawaan, serta caching dalam memori untuk aplikasi skala internet.

Amazon Timestream adalah layanan basis data deret waktu yang cepat, dapat diskalakan, dan terkelola penuh untuk IoT dan aplikasi operasional yang memudahkan penyimpanan dan menganalisis triliunan peristiwa per hari dengan biaya sebesar 1/10 dari biaya basis data relasional. Didorong oleh perkembangan perangkat IoT, sistem IT, dan mesin industri cerdas, data deret waktu—data yang mengukur bagaimana segala sesuatu berubah seiring waktu—adalah salah satu jenis data yang paling cepat tumbuh.

Amazon Quantum Ledger basis data (Amazon QLDB) adalah basis data buku besar terkelola penuh yang menyediakan log transaksi yang transparan, tetap, dan dapat diverifikasi secara kriptografis milik otoritas pusat tepercaya. Amazon QLDB melacak setiap perubahan data aplikasi dan memelihara riwayat perubahan yang lengkap dan dapat diverifikasi dari waktu ke waktu.

Amazon Keyspaces (untuk Apache Cassandra) adalah layanan basis data yang kompatibel dengan Apache Cassandra yang dapat diskalakan, berketersediaan tinggi, dan terkelola. Dengan Amazon Keyspaces, Anda dapat menjalankan beban kerja Cassandra di AWS dengan kode aplikasi Cassandra dan alat developer yang sama dengan yang Anda gunakan saat ini. Anda tidak perlu menyediakan, melakukan patch, atau mengelola server, dan tidak perlu menginstal, mengelola, atau mengoperasikan perangkat lunak. Amazon Keyspaces ini nirserver, sehingga Anda hanya membayar sumber daya yang digunakan dan layanan ini secara otomatis akan menaikkan dan menurunkan skala tabel dalam merespons lalu lintas aplikasi.

Amazon Elastic File System (Amazon EFS) memberikan sistem file yang sederhana dan nirserver, tanpa pengaturan lebih lanjut, yang memungkinkan Anda berbagi data file tanpa menyediakan atau mengelola penyimpanan. Amazon EFS dapat digunakan dengan layanan AWS Cloud dan sumber daya on-premise, serta dibangun untuk menskalakan permintaan ke ukuran petabita tanpa mengganggu aplikasi. Dengan Amazon EFS, Anda dapat memperbesar dan memperkecil sistem file Anda secara otomatis saat Anda menambahkan dan menghapus file, yang mengurangi kebutuhan untuk menyediakan dan mengelola kapasitas guna mengakomodasi pertumbuhan. Amazon EFS dapat dipasangkan dengan fungsi Lambda yang menjadikannya opsi penyimpanan file yang layak untuk API.

Opsi penyimpanan data non-nirserver

Amazon Relational Database Service (Amazon RDS) adalah layanan web terkelola yang mempermudah Anda untuk menyiapkan, mengoperasikan, dan menskalakan basis data relasional menggunakan salah satu mesin yang tersedia (Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle, dan Microsoft SQL Server) dan berjalan pada beberapa jenis instans basis data yang berbeda yang dioptimalkan untuk memori, performa, atau I/O.

Amazon Redshift adalah layanan gudang data berskala petabita dan terkelola penuh di cloud.

Amazon ElastiCache adalah deployment Redis atau Memcached terkelola penuh. Deploy, jalankan, dan skalakan penyimpanan data dalam memori kompatibel sumber terbuka yang populer dengan lancar.

Amazon Neptune adalah layanan basis data grafik yang cepat, andal, dan terkelola penuh yang memudahkan untuk membuat dan menjalankan aplikasi yang bekerja dengan set data yang selalu terhubung. Neptune mendukung model grafik populer - grafik properti dan Resource Description Framework (RDF) W3C - serta bahasa kueri masing-masing, sehingga memungkinkan Anda dengan mudah membangun kueri yang secara efisien menavigasi set data yang sangat terhubung.

Amazon DocumentDB (dengan kompatibilitas MongoDB) adalah layanan basis data dokumen yang cepat, sangat tersedia, dapat diskalakan, dan terkelola penuh yang mendukung beban kerja MongoDB.

Terakhir, Anda juga dapat menggunakan penyimpanan data yang berjalan secara independen di Amazon EC2 sebagai tingkat data untuk aplikasi multi-tingkat

Tingkat presentasi

Tingkat presentasi bertanggung jawab untuk berinteraksi dengan tingkat logika melalui titik akhir API Gateway REST yang terekspos melalui internet. Setiap klien atau perangkat berkemampuan HTTPS dapat berkomunikasi dengan titik akhir ini, memberi tingkat presentasi Anda fleksibilitas untuk mengambil banyak bentuk (aplikasi desktop, aplikasi seluler, halaman web, perangkat IoT, dan sebagainya). Tergantung pada persyaratan Anda, tingkat presentasi Anda dapat menggunakan penawaran nirserver AWS berikut: Setiap klien atau perangkat berkemampuan HTTPS dapat berkomunikasi dengan titik akhir ini, memberi tingkat presentasi Anda fleksibilitas untuk mengambil banyak bentuk (aplikasi desktop, aplikasi seluler, halaman web, perangkat IoT, dan sebagainya). Tergantung pada persyaratan Anda, tingkat presentasi Anda dapat menggunakan penawaran nirserver AWS berikut:

- Amazon Cognito - Layanan identitas pengguna dan sinkronisasi data nirserver yang memungkinkan Anda menambahkan pendaftaran pengguna, proses masuk, dan kontrol akses ke web dan aplikasi seluler Anda dengan cepat dan efisien. Amazon Cognito menskalakan jutaan pengguna dan mendukung fitur masuk dengan penyedia identitas sosial seperti Facebook, Google, dan Amazon, serta penyedia identitas korporasi via SAML 2.0.
- Amazon S3 with CloudFront - Memungkinkan Anda menyajikan situs web statis, seperti aplikasi halaman tunggal, secara langsung dari bucket S3 tanpa memerlukan penyediaan server web. Anda dapat menggunakan CloudFront sebagai jaringan pengiriman konten terkelola (CDN) untuk meningkatkan performa dan mengaktifkan SSL/TL menggunakan sertifikat terkelola atau kustom.

AWS Amplify adalah serangkaian alat dan layanan yang dapat digunakan bersama atau terpisah, untuk membantu developer web dan seluler front-end membangun aplikasi tumpukan penuh yang dapat diskalakan dan didukung oleh AWS. Amplify menawarkan layanan terkelola penuh untuk men-deploy dan meng-host aplikasi web statis secara global, yang dilayani oleh CDN Amazon yang andal dengan ratusan titik kehadiran secara global dan dengan alur kerja CI/CD bawaan yang mempercepat siklus rilis aplikasi Anda. Amplify mendukung kerangka kerja web populer termasuk JavaScript, React, Angular, Vue, Next.js, dan platform seluler termasuk Android, iOS, React Native, Ionic, dan Flutter. Bergantung pada konfigurasi jaringan dan persyaratan aplikasi, Anda mungkin perlu mengaktifkan API Gateway API agar sesuai dengan cross-origin resource sharing (CORS). Kesesuaian CORS memungkinkan browser web langsung memanggil API Anda dari dalam halaman web statis.

Saat Anda menerapkan situs web dengan CloudFront, Anda diberi nama domain CloudFront untuk menjangkau aplikasi Anda (misalnya, `d2d47p2vcczkh2.cloudfront.net`). Anda dapat menggunakan Amazon Route 53 untuk mendaftarkan nama domain dan mengarahkannya ke distribusi CloudFront Anda, atau mengarahkan nama domain yang sudah dimiliki ke distribusi CloudFront Anda. Hal ini memungkinkan pengguna mengakses situs Anda menggunakan nama domain yang sudah dikenal. Perhatikan bahwa Anda juga dapat menetapkan nama domain kustom menggunakan Route 53 ke distribusi API Gateway Anda, yang memungkinkan pengguna memanggil API menggunakan nama domain yang sudah dikenal.

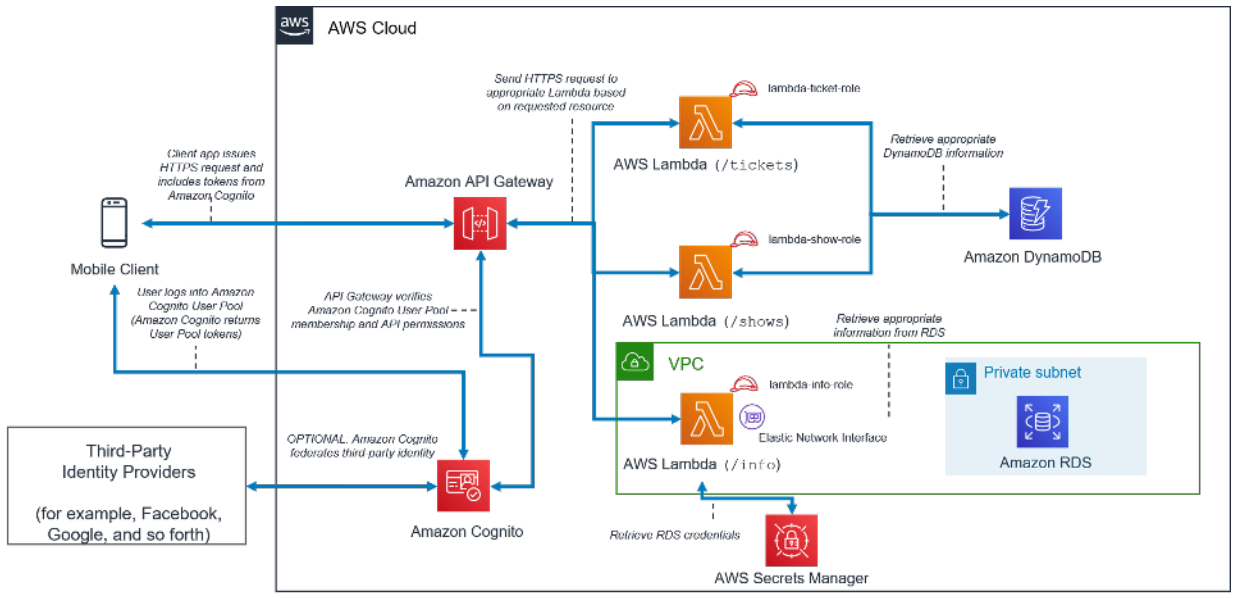
Sampel pola arsitektur

Anda dapat menerapkan pola arsitektur populer menggunakan API Gateway dan AWS Lambda sebagai tingkat logika Anda. Laporan resmi ini mencakup pola arsitektur paling populer yang memanfaatkan tingkat logika berbasis AWS Lambda:

- Backend seluler - Sebuah aplikasi seluler yang berkomunikasi dengan API Gateway dan Lambda untuk mengakses data aplikasi. Pola ini dapat diperluas ke klien HTTPS umum yang tidak menggunakan sumber daya AWS nirserver untuk meng-host sumber daya tingkat presentasi (seperti klien desktop, server web yang berjalan di EC2, dan lain sebagainya).
- Aplikasi halaman tunggal - Aplikasi halaman tunggal yang di-host di Amazon S3 dan CloudFront berkomunikasi dengan API Gateway dan AWS Lambda untuk mengakses data aplikasi.
- Aplikasi web – Aplikasi web adalah back-end aplikasi web tujuan umum dan berbasis peristiwa, yang menggunakan AWS Lambda dengan API Gateway untuk logika bisnisnya. Aplikasi ini juga menggunakan Amazon DynamoDB sebagai basis datanya dan Amazon Cognito untuk manajemen pengguna. Semua konten statis di-host menggunakan Amplify.

Selain dua pola ini, laporan resmi ini membahas penerapan Lambda dan API Gateway ke arsitektur layanan mikro umum. Arsitektur layanan mikro adalah pola populer yang, meski pun bukan arsitektur tiga tingkat standar, namun memungkinkan penerapan decouple pada komponen aplikasi dan men-deploy komponen aplikasi tersebut sebagai unit fungsionalitas individu dan stateless yang berkomunikasi satu sama lain.

Backend seluler



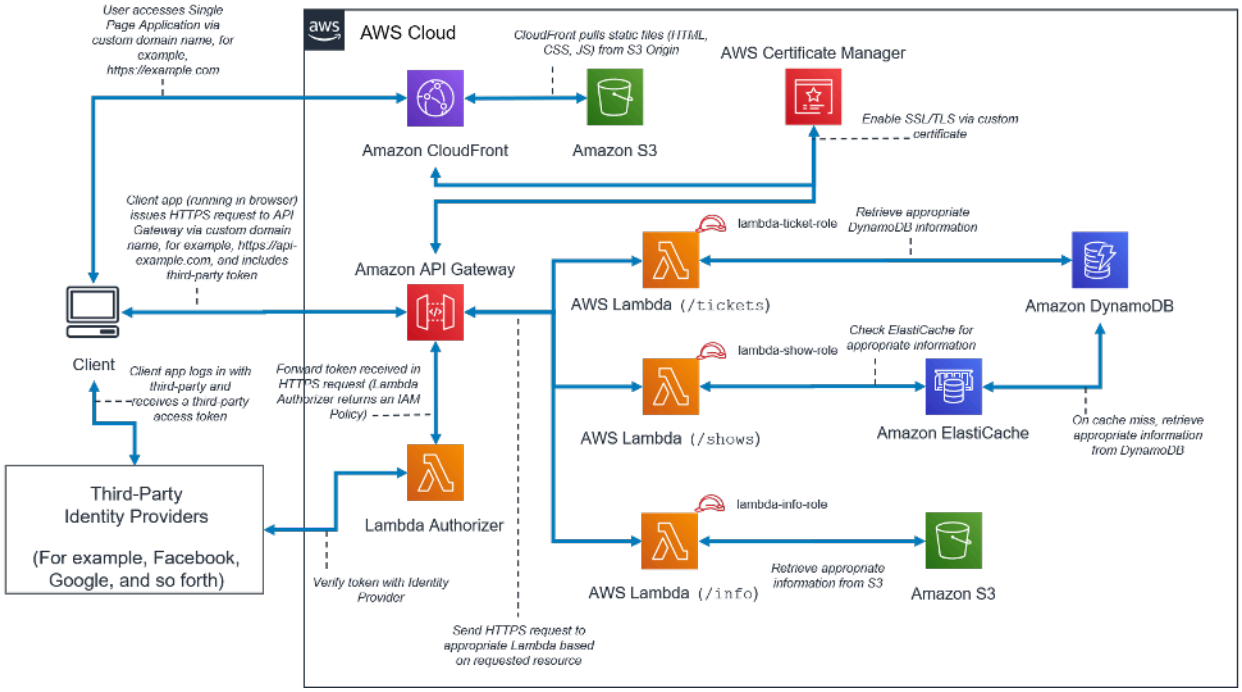
Pola arsitektur untuk backend seluler nirserver

Tabel 1 - Komponen tingkat backend seluler

Tingkat	Komponen
Presentasi	Aplikasi seluler yang berjalan pada perangkat pengguna.
Logika	Amazon API Gateway dengan AWS Lambda. Arsitektur ini menunjukkan tiga layanan terekspos (`/tickets`, `/shows`, dan `/info`). Titik akhir API Gateway diamankan oleh pool pengguna Amazon Cognito Dalam metode ini, pengguna masuk ke pool pengguna Amazon Cognito (menggunakan pihak ketiga terfedera si jika perlu), dan menerima token akses dan ID yang digunakan untuk mengotorisasi panggilan API Gateway. Setiap fungsi Lambda diberi peran Identity and Access Management (IAM) sendiri untuk

Tingkat	Komponen
	menyediakan akses ke sumber data yang sesuai.
Data	DynamoDB digunakan untuk layanan `/tickets` dan `/shows`. Amazon RDS digunakan untuk layanan `/info` ini. Fungsi Lambda ini mengambil kredensial Amazon RDS dari AWS Secrets Manager dan menggunakan antarmuka jaringan elastis untuk mengakses subnet privat.

Aplikasi halaman tunggal

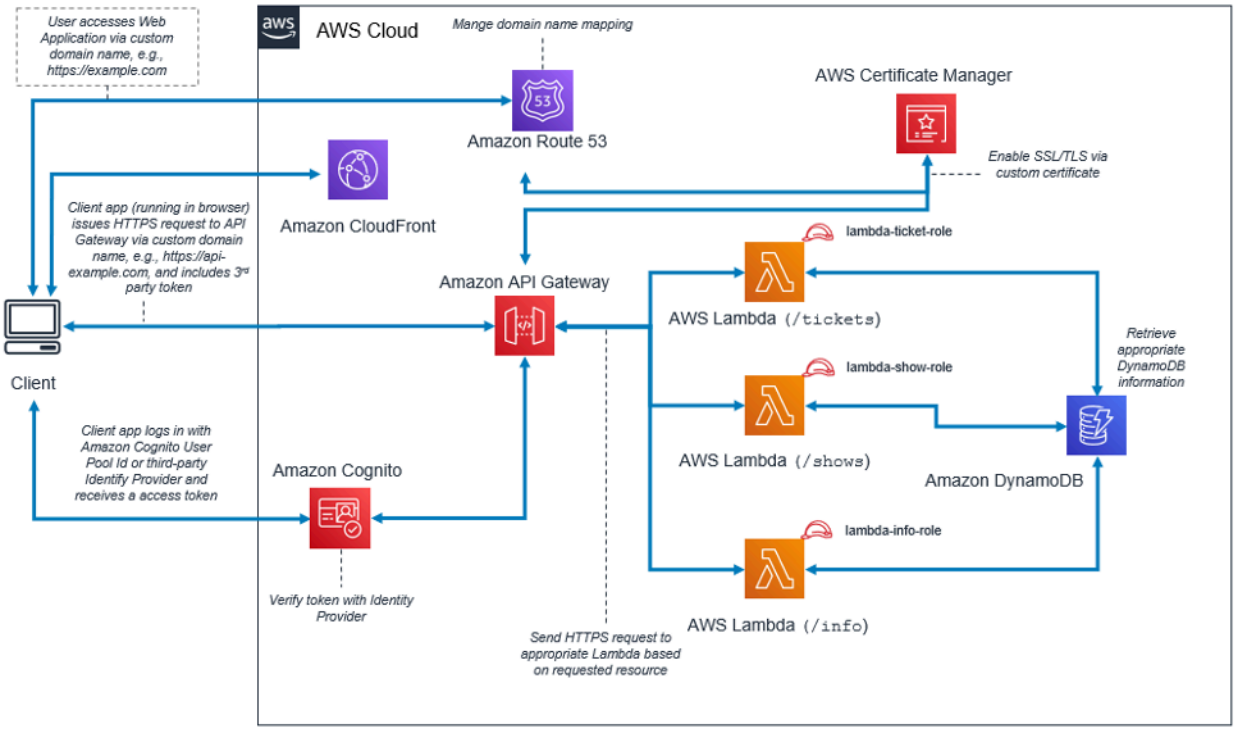


Pola arsitektur untuk aplikasi halaman tunggal nirserver

Tabel 2 - Komponen aplikasi halaman tunggal

Tingkat	Komponen
Presentasi	Konten situs web statis yang di-host di Amazon S3, didistribusikan oleh CloudFront. AWS Certificate Manager memungkinkan penggunaan sertifikat SSL/TLS kustom.
Logika	API Gateway dengan AWS Lambda. Arsitektur ini menunjukkan tiga layanan terekspos (`/tickets`, `/shows`, dan `/info`). Titik akhir API Gateway diamankan oleh otorisasi Lambda. Dalam metode ini, pengguna masuk melalui penyedia identitas pihak ketiga dan mendapatkan akses serta token ID. Token ini disertakan dalam panggilan API Gateway, dan pemberi otorisasi Lambda memvalidasi token ini dan menghasilkan kebijakan IAM yang berisi izin inisiasi API. Setiap fungsi Lambda diberi IAM role sendiri untuk menyediakan akses ke sumber data yang sesuai.
Data	Amazon DynamoDB digunakan untuk layanan `/tickets` dan `/shows`. Amazon ElastiCache digunakan oleh layanan `/shows` untuk meningkatkan performa basis data. Peristiwa cache miss dikirim ke DynamoDB. Amazon S3 digunakan untuk meng-host konten statis yang digunakan oleh `/info service`.

Aplikasi web



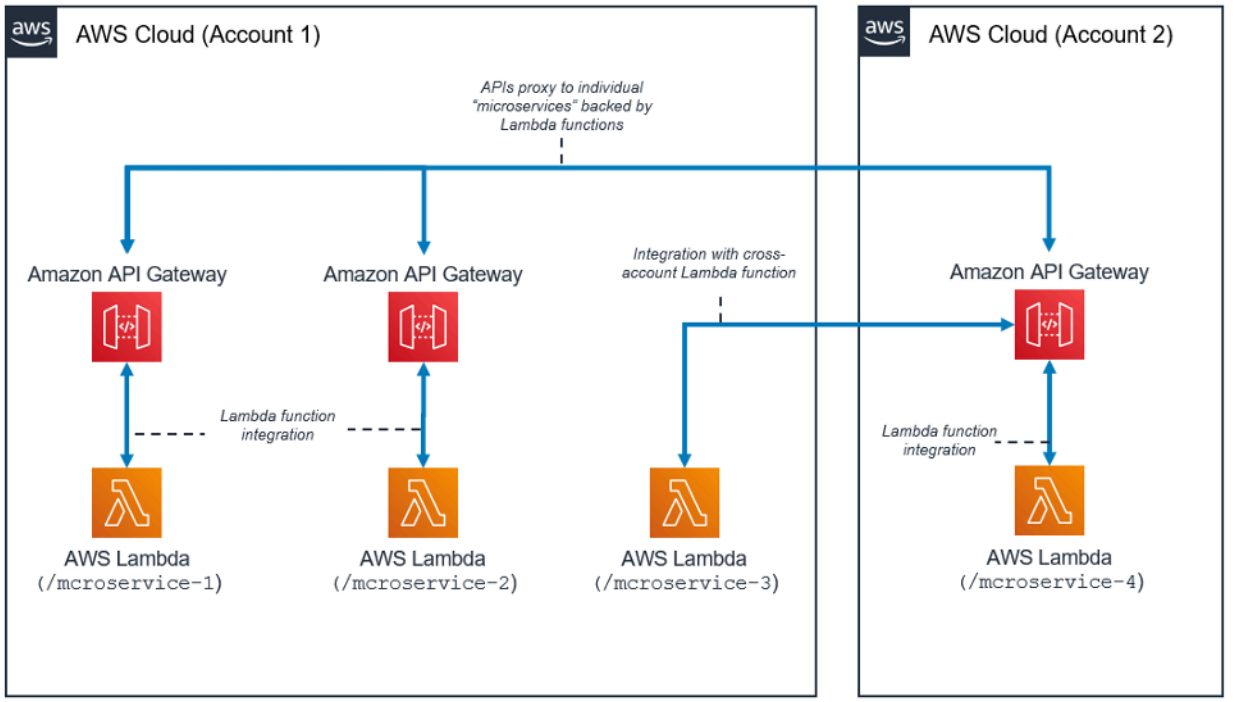
Pola arsitektur untuk aplikasi web

Tabel 3 - Komponen aplikasi web

Tingkat	Komponen
Presentasi	Aplikasi front-end adalah semua konten statis (HTML, CSS, JavaScript dan gambar) yang dihasilkan oleh utilitas React seperti create-react-app. Amazon CloudFront meng-host semua objek ini. Aplikasi web, jika digunakan , mengunduh semua sumber daya ke browser dan mulai berjalan dari sana. Aplikasi web yang terhubung ke backend memanggil API.
Logika	Lapisan logika dibangun menggunakan fungsi Lambda yang ada di belakang API REST API Gateway.

Tingkat	Komponen
	Arsitektur ini menunjukkan beberapa layanan terekspos. Ada berbagai fungsi Lambda yang masing-masing menangani aspek yang berbeda dari suatu aplikasi. Fungsi Lambda berada di belakang API Gateway dan dapat diakses menggunakan jalur URL API. Autentikasi pengguna ditangani menggunak an pool pengguna Amazon Cognito atau penyedia pengguna terfederasi. API Gateway menggunakan integrasi siap pakai dengan Amazon Cognito. Hanya setelah pengguna diautentikasi, klien akan menerima token JSON Web Token (JWT) yang kemudian harus digunakan saat melakukan panggilan API. Setiap fungsi Lambda diberi IAM role sendiri untuk menyediakan akses ke sumber data yang sesuai.
Data	Dalam contoh khusus ini, DynamoDB digunakan untuk penyimpanan data, tetapi basis data atau layanan penyimpanan Amazon yang dibuat khusus dapat digunakan tergantun g pada kasus penggunaan dan skenario penggunaan.

Layanan Mikro dengan Lambda



Pola arsitektur untuk layanan mikro dengan Lambda

Pola arsitektur layanan mikro tidak terikat pada arsitektur tiga tingkat yang standar; namun, pola populer ini dapat mewujudkan manfaat yang signifikan dari penggunaan sumber daya nirserver.

Dalam arsitektur ini, masing-masing komponen aplikasi di-decouple serta di-deploy dan dioperasikan secara independen. API yang dibuat dengan Amazon API Gateway beserta berbagai fungsi yang selanjutnya diluncurkan oleh AWS Lambda adalah semua yang Anda perlukan untuk membangun layanan mikro. Tim Anda dapat menggunakan layanan ini untuk men-decouple dan melakukan fragmentasi pada lingkungan Anda ke tingkat granularitas yang diinginkan.

Secara umum, lingkungan layanan mikro dapat memberikan kesulitan berikut: overhead berulang untuk membuat setiap layanan mikro baru, masalah dalam mengoptimalkan kepadatan dan pemanfaatan server, kompleksitas dalam menjalankan beberapa versi untuk beberapa layanan mikro secara bersamaan, dan melakukan proliferasi untuk persyaratan kode sisi klien guna mengintegrasikan dengan banyak layanan terpisah.

Jika Anda membuat layanan mikro menggunakan sumber daya nirserver, masalah ini menjadi tidak begitu sulit untuk diatasi dan, dalam beberapa kasus, akan teratasi dengan sendirinya. Pola layanan mikro nirserver mengurangi hambatan untuk pembuatan setiap layanan mikro berikutnya (API Gateway bahkan memungkinkan kloning untuk API yang ada, dan penggunaan fungsi Lambda di

akun lain). Mengoptimalkan pemanfaatan server tidak lagi relevan dengan pola ini. Terakhir, Amazon API Gateway menyediakan SDK klien yang dihasilkan secara terprogram dalam sejumlah bahasa populer untuk mengurangi overhead integrasi.

Kesimpulan

Pola arsitektur multi-tingkat mendorong praktik terbaik untuk menciptakan komponen aplikasi yang mudah dipelihara, di-decouple, dan diskalakan. Saat Anda membuat tingkat logika yang memungkinkan integrasi terjadi melalui API Gateway dan komputasi terjadi di dalam AWS Lambda, Anda akan mewujudkan sasaran ini sekaligus mengurangi jumlah percobaan untuk mencapainya. Bersama-sama, layanan ini menyediakan front end API HTTPS untuk klien Anda dan lingkungan yang aman untuk menerapkan logika bisnis Anda sambil meniadakan overhead yang diperlukan dengan mengelola infrastruktur berbasis server yang standar.

Kontributor

Kontributor dokumen ini meliputi:

- Andrew Baird, Arsitek Solusi (AWS Solutions Architect)
- Bryant Bost, Konsultan AWS ProServe (AWS ProServe Consultant)
- Stefano Buliani, Manajer Produk Senior (Senior Product Manager), Teknis, AWS Mobile
- Vyom Nagrani, Manajer Produk Senior (Senior Product Manager), AWS Mobile
- Ajay Nair, Manajer Produk Senior (Senior Product Manager), AWS Mobile
- Rahul Popat, Arsitek Solusi Global (Global Solutions Architect)
- Brajendra Singh, Arsitek Solusi Senior (Senior Solutions Architect)

Bacaan Lebih Lanjut

Untuk informasi tambahan, lihat:

- Laporan Resmi dan Panduan AWS

Revisi dokumen

Untuk mendapatkan notifikasi tentang pembaruan laporan resmi ini, sebaiknya berlangganan umpan RSS.

pembaruan-riwayat-perubahan	deskripsi-riwayat-pembaruan	tanggal-riwayat-pembaruan
Laporan resmi diperbarui	Memperbarui untuk fitur dan pola layanan baru.	20 Oktober 2021
Laporan resmi diperbarui	Memperbarui untuk fitur dan pola layanan baru.	1 Juni 2021
Laporan resmi diperbarui	Memperbarui untuk fitur layanan baru.	25 September 2019
Publikasi awal	Laporan resmi dipublikasikan.	1 November 2015

Pemberitahuan

Pelanggan bertanggung jawab untuk membuat penilaian independen mereka sendiri atas informasi dalam dokumen ini. Dokumen ini: (a) hanya disediakan sebagai informasi, (b) berisi penawaran produk dan praktik AWS saat ini, yang dapat berubah tanpa pemberitahuan, dan (c) tidak menjadi komitmen atau jaminan apa pun dari AWS dan afiliasi, pemasok, atau pemberi lisensinya. Produk atau layanan AWS disediakan “sebagaimana adanya” tanpa jaminan, representasi, atau ketentuan apa pun, baik tersurat maupun tersirat. Tanggung jawab dan kewajiban AWS kepada pelanggannya dikendalikan oleh perjanjian AWS, dan dokumen ini bukan bagian dari, juga tidak mengubah, perjanjian apa pun antara AWS dan pelanggannya.