Glosarium

A

Artificial Intelligence
Kemampuan komputer untuk bertindak seperti manusia.

B

Blockchain
Blockchain adalah catatan transaksi digital. Nama “blockchain” berasal dari strukturnya, di mana catatan individu (disebut sebagai *block*) dihubungkan bersama dalam daftar tunggal (disebut sebagai *chain*). Blockchain digunakan untuk mencatat transaksi yang dilakukan dengan cryptocurrency, seperti Bitcoin, dll.

C

Cloud
Dalam dunia IT, cloud mengacu pada server yang diakses melalui Internet, serta perangkat lunak dan database yang berjalan di server tersebut.
Compliance
Dalam tata kelola perusahaan, compliance berarti mengikuti suatu spesifikasi, standar, atau hukum yang telah diatur dengan jelas yang biasanya diterbitkan oleh lembaga atau organisasi yang berwenang dalam suatu bidang tertentu.

D

Database
Struktur data yang menyimpan informasi yang terorganisir.
Deploy
Dalam konteks IT, deploy merujuk pada semua proses yang terlibat dalam mendapatkan software atau hardware rilis dan berjalan dengan baik, termasuk instalasi, konfigurasi, pengoperasian, pengujian, dan membuat perubahan yang diperlukan.

E

Entitas
Satu objek unik di dunia nyata. Ia mengacu pada individu, organisasi, produk, atau komponen sistem.
Exabyte
Exabyte adalah 10^{18} atau 1.000.000.000.000.000.000.000 byte. 1 exabyte sama dengan 1.000 petabyte.

H

Hard Drive
Hard drive adalah tempat menyimpan semua data Anda. Data tersebut disimpan di secara magnetis, sehingga tetap berada di drive bahkan setelah daya dimatikan. Istilah "hard drive" sebenarnya adalah singkatan dari "hard disk drive", keduanya merujuk pada arti yang sama.

Hypervisor
Program perangkat lunak yang mengelola satu atau lebih mesin virtual. Hypervisor digunakan untuk membuat, memulai, menghentikan, dan mengatur ulang VM.

I

Instance
Instance adalah server virtual di AWS Cloud.

L

Latensi
Waktu yang diperlukan untuk mengirim dan menerima data.

M

Machine Learning
Sebuah jenis dari Artificial Intelligence yang dapat "belajar" atau beradaptasi dari waktu ke waktu.

O

On-premise
Penyimpanan dan pemeliharaan data di server lokal atau pribadi.

P

Permission
Permission diartikan sebagai memberikan persetujuan atau otorisasi atau mengizinkan seseorang untuk melakukan sesuatu. Pada konteks AWS, permission memungkinkan Anda untuk menentukan akses ke sumber daya AWS.
Petabyte
Petabyte adalah 10^{15} atau 1.000.000.000.000.000 byte. 1 petabyte sama dengan 1.000 terabyte.

S

Sumber Daya
Di AWS, sumber daya (resource) berarti entitas yang dapat Anda pakai, contohnya Amazon EC2 instance atau Amazon S3 bucket.

T

Terabyte
Terabyte adalah 10^{12} atau 1.000.000.000.000 byte. 1 terabyte sama dengan 1.000 gigabyte.
Throughput
Jumlah data yang dapat dikirim dalam waktu tertentu.

W

Web Hosting

Web Hosting menyimpan semua halaman website Anda dan membuatnya tersedia untuk komputer yang terhubung ke Internet.

Pengantar ke Amazon Web Services

Selamat datang di Kelas Cloud Practitioner Essentials! Kelas ini adalah kelas dasar yang hadir untuk mengenalkan Anda pada layanan komputasi *cloud* dengan Amazon Web Services atau biasa disingkat AWS.

Amazon Web Services atau AWS adalah salah satu layanan penyedia komputasi cloud yang telah hadir di seluruh dunia. AWS merupakan platform cloud yang paling komprehensif dan digunakan secara luas. Faktanya, jumlah layanan di AWS mencapai lebih dari ratusan layanan unggulan dengan jutaan pelanggan.

Dengan cloud seperti AWS ini, pengguna dari berbagai kalangan perusahaan IT, pada umumnya, menjadi lebih tangkas dalam menjalankan aktivitas operasional sehari-hari dan lebih cepat dalam berinovasi.

Di kelas ini kita akan mempelajari setiap materi secara terstruktur. Mari awali langkah perjalanan kita dengan beberapa pertanyaan terkait Kelas Cloud Practitioner Essentials.

Siapa yang dapat mengambil kelas ini?

Kelas Cloud Practitioner Essentials ini ditujukan bagi Anda yang mencari pemahaman secara keseluruhan tentang AWS Cloud, terlepas dari latar belakang pekerjaan tertentu. Kelas ini pun ditujukan bagi Anda yang bekerja di bidang berikut:

- Sales
- Hukum
- Pemasaran
- Analis bisnis
- Manajer proyek
- Pelajar AWS Academy
- Profesi lain terkait IT

Apa saja prasyarat untuk mengambil kelas ini?

Untuk mulai mempelajari kelas ini, setidaknya Anda harus memahami beberapa pengetahuan dasar seperti berikut:

- Teknis IT secara umum. Minimal Anda mengenal istilah komputer, jaringan, perangkat keras, perangkat lunak, web, dan internet.
- Bisnis IT secara umum. Anda mengetahui perusahaan-perusahaan yang bergerak di dunia IT seperti Amazon.

Apakah Anda sudah punya bekal yang cukup untuk mengikuti kelas ini? Jika jawabannya adalah “ya”, mari kita lanjut ke pertanyaan berikutnya!

Apa saja yang akan kita pelajari?

Jangan khawatir jika Anda tidak memiliki pengetahuan sama sekali tentang komputasi cloud atau bahkan Amazon Web Services, karena materi yang dibahas pada kelas ini tergolong ringan dan disajikan dengan bahasa yang mudah dipahami. Sebelum mengikuti kelas ini, Anda perlu tahu terlebih dahulu modul apa saja yang ada di dalamnya.

Kelas ini terdiri dari 11 modul yang akan membahas tentang pengenalan konsep AWS Cloud, layanan-layanannya, keamanan, arsitektur, harga dan dukungan, kemudian ditutup dengan penilaian akhir berupa ujian sehingga menambah pengetahuan dan wawasan Anda mengenai AWS Cloud menjadi lebih baik.

Berikut adalah materi-materi yang akan Anda perdalam / pelajari secara komprehensif:

- **Modul 1: Pengantar ke Amazon Web Services**
  Menjelaskan tentang materi pengenalan, seperti apa saja yang harus Anda siapkan sebelum mengikuti kelas; manfaat dari AWS; perbedaan antara penyajian *on-demand* (sesuai permintaan) dan model penerapan cloud; serta model biaya dengan skema *pay-as-you-go*.
- **Modul 2: Komputasi di Cloud**
  Membahas materi komputasi di cloud, yakni manfaat dari Amazon Elastic Compute Cloud (Amazon EC2) di level dasar; perbedaan tipe dari Amazon EC2 instance; perbedaan antara variasi pilihan penagihan untuk Amazon EC2; manfaat Amazon EC2 Auto Scaling; manfaat Elastic Load Balancing, contoh penggunaan Elastic Load Balancing; perbedaan antara Amazon Simple Notification Service (Amazon SNS) dan Amazon Simple Queue Services (Amazon SQS); serta layanan komputasi lain di AWS.
- **Modul 3: Infrastruktur Global dan Keandalan**
  Menelaah materi terkait infrastruktur global AWS; konsep dasar Availability Zone; manfaat Amazon CloudFront dan Edge locations; serta membandingkan perbedaan metode untuk penyajian layanan AWS.

- **Modul 4: Jaringan**
  Mengupas tuntas materi jaringan, seperti konsep dasarnya; perbedaan antara sumber daya jaringan publik dan privat; virtual private gateway dan virtual private network (VPN) untuk menghubungkan AWS Cloud dengan jaringan lain; AWS Direct Connect; manfaat penerapan arsitektur *hybrid;* lapisan keamanan yang digunakan dalam strategi IT; dan layanan yang digunakan untuk berinteraksi dengan jaringan global AWS.
- **Modul 5: Penyimpanan dan Database**
  Mengulas konsep dasar penyimpanan dan *databases* (basis data); manfaat Amazon Elastic Block Store (Amazon EBS); Amazon Simple Storage Service (Amazon S3); Amazon Elastic File System (Amazon EFS); variasi solusi penyimpanan; Amazon DynamoDB; dan terakhir ragam layanan database.
- **Modul 6: Keamanan**
  Mendeskripsikan materi keamanan, yakni manfaat *shared responsibility model* (model tanggung jawab bersama); *multi-factor authentication* (autentikasi multifaktor) atau MFA; tingkat keamanan AWS Identity and Access Management (IAM); dasar-dasar kebijakan keamanan; AWS Organizations; *compliance* (kepatuhan) dengan AWS; dan layanan keamanan utama AWS yang mudah.
- **Modul 7: Pemantauan dan Analitik**
  Menelaah pendekatan untuk memantau *environment* (lingkungan) AWS Anda, manfaat Amazon CloudWatch, AWS CloudTrail, dan AWS Trusted Advisor.
- **Modul 8: Harga dan Dukungan**
  Menguraikan materi terkait model harga dan dukungan, seperti AWS Free Tier (Tingkat Gratis); AWS Organizations dan consolidated billing (tagihan terkonsolidasi); AWS Budgets; AWS Cost Explorer; AWS Pricing Calculator; membedakan setiap AWS Support Plans; dan terakhir AWS Marketplace.
- **Modul 9: Migrasi dan Inovasi**
  Mengkaji materi terkait migrasi dan inovasi di AWS Cloud, yaitu AWS Cloud Adoption Framework (AWS CAF); enam faktor utama dari strategi migrasi cloud; manfaat beragam solusi migrasi data: AWS Snowcone, AWS Snowball, dan AWS Snowmobile; dan terakhir, meringkas cakupan luas dari solusi inovatif yang ditawarkan AWS.
- **Modul 10: Perjalanan Cloud**
  Menjelaskan lima pilar dari AWS Well-Architected Framework dan enam manfaat dari komputasi cloud.
- **Modul 11: Dasar-Dasar AWS Certified Cloud Practitioner**
  Mengulik sumber daya untuk persiapan ujian AWS Certified Cloud Practitioner sekaligus manfaat menjadi seseorang yang bersertifikat AWS.
- **Penilaian Akhir Kelas**
  Penilaian akhir ini berisi soal-soal yang mendekati ujian AWS Certified Cloud Practitioner.

Apa Yang Diharapkan Setelah Mengikuti Kelas Ini?

Agar memahami kelas ini dengan baik, Anda diharapkan untuk belajar secara perlahan. Tidak perlu terburu-buru dalam studi Anda. Karena yang terpenting adalah peningkatan terhadap pemahaman dan kecakapan Anda.

Setelah mempelajari kelas ini, Anda diharapkan mampu untuk:

- Merangkum definisi kerja AWS.
- Memaparkan perbedaan antara *on-premise* (lokal), *hybrid*, dan *cloud* sepenuhnya.
- Menjelaskan dasar-dasar infrastruktur global AWS Cloud.
- Menguraikan 6 (enam) manfaat AWS Cloud.
- Mendeskripsikan dan memberikan contoh layanan utama AWS, termasuk komputasi, jaringan, database, dan penyimpanan.
- Mengidentifikasi solusi yang sesuai menggunakan layanan AWS Cloud dengan berbagai kasus penggunaan.
- Menerangkan AWS Well-Architected Framework.
- Menguraikan dan menjelaskan shared responsibility model (model tanggung jawab bersama).
- Menggambarkan layanan keamanan utama dalam AWS Cloud.
- Menjelaskan dasar-dasar migrasi AWS Cloud.
- Mengartikulasikan manfaat finansial dari AWS Cloud untuk manajemen biaya organisasi/perusahaan.
- Menentukan penagihan utama, pengelolaan akun, dan model harga.
- Memaparkan cara penggunaan *pricing tools* (alat penetapan harga) yang dapat menghemat layanan AWS.

Metode Apa Saja Yang Akan Kita Gunakan?

Ada berbagai metode pembelajaran yang digunakan dalam kelas ini, di antaranya:

- Teori, berisi materi pembahasan yang mudah dipahami tentang AWS.
- Quiz, berupa soal yang bertujuan untuk menguji pemahaman Anda.
- Exam, mencakup soal-soal yang bertujuan untuk menguji pemahaman Anda setelah mempelajari keseluruhan kelas. Ini merupakan syarat kelulusan kelas.

Anda diharapkan dengan tekun menjalani semua metode tersebut agar paham AWS secara mendalam. Sudah tidak sabar ingin segera menyelami layanan cloud dengan Amazon Web Services? Mari kita mulai.

Tujuan Pembelajaran
Di modul ini, Anda akan mempelajari bagaimana cara:

- Merangkum manfaat menggunakan AWS.
- Menjelaskan perbedaan antara penyajian *on-demand* dan penerapan cloud.
- Merangkum model harga *pay-as-you-go*.

Oke, mari kita lanjutkan ke materi berikutnya!

Selamat Datang di Kedai Kopi
Kelas ini akan mencakup semua informasi penting yang perlu Anda pahami seputar AWS. Dengan begitu, Anda dapat mengetahui mengapa AWS itu bermanfaat bagi kebutuhan perusahaan dan bisnis saat ini.

AWS menawarkan berbagai macam layanan untuk setiap kegunaan. Dimulai dengan elemen dasar, seperti komputasi, penyimpanan, dan keamanan jaringan, hingga solusi kompleks seperti *blockchain*, *machine learning*, atau *artificial intelligence* (kecerdasan buatan), serta platform pengembangan robot.

Bahkan termasuk juga layanan yang sangat terspesialisasi, seperti sistem manajemen produksi video dan satelit orbital yang dapat Anda sewa setiap menit.

Namun semua hal tersebut nampaknya terlalu kompleks dan perlu lebih banyak waktu untuk kita bahas di kelas dasar seperti ini. Jadi, mari kita sederhanakan pembahasan kita dengan memulai dari model komputasi *cloud* dasar.

Tahukah Anda? Hampir semua model komputasi modern adalah berbentuk *client-server*.

Dalam dunia komputasi, client dapat berupa web browser atau aplikasi yang dapat membuat permintaan ke server. Sebuah server dapat berupa layanan seperti Amazon Elastic Compute Cloud (Amazon EC2).

Contoh interaksinya adalah client membuat permintaan untuk mengakses sebuah artikel berita, skor dalam game online, atau video lucu lalu server mengevaluasi detail permintaan tersebut dan memenuhinya dengan mengembalikan informasi ke client.

Oke, mungkin pembahasan di atas terlalu teknis ya. Bagaimana kalau kita membuat suatu perumpamaan yang dapat digunakan secara berkelanjutan di setiap modulnya? Tapi perumpamaan seperti apa ya yang dapat mencakup setiap pembahasan di kelas ini?

Bagaimana dengan skenario kedai kopi? Sepertinya menarik. Kedai kopi ini akan memberi kita beberapa metafora dunia nyata untuk membantu Anda memahami mengapa AWS dapat mengubah cara pengoperasian IT di seluruh dunia.

Kita mulai dari sebuah pertanyaan dasar. Apa saja elemen yang ada di sebuah kedai kopi? Kasir dan pelanggan, tentu.

Dalam model client-server. Kasir berperan sebagai server sedangkan pelanggan adalah client. Di kedai kopi pelanggan membuat suatu permintaan berupa segelas kopi. Namun di dunia komputasi, permintaan dapat berbentuk apa pun: analisis pola hujan di negara Afrika Selatan, rontgen terbaru dari lutut Anda, atau mungkin video anak kucing yang menggemaskan.

Apa pun bisnisnya, pada dasarnya client membuat suatu permintaan--tentu dengan telah memiliki izin akses--kemudian server menanggapi permintaan tersebut.

Kembali ke kedai kopi. Kasir adalah *server*-nya. Di AWS, kasir tersebut diberi nama Amazon Elastic Compute Cloud (EC2), sebuah server virtual dan kita akan memanggilnya instance.

Mari kita lihat proses transaksi yang terjadi antara kasir dan pelanggan ini dari sudut pandang arsitektural.

1. Pelanggan (client) membuat permintaan ke kasir (server).
2. Kasir memvalidasi bahwa permintaan tersebut sah, dalam hal ini apakah pelanggan telah membayar atau belum.
3. Jika ya, maka kasir akan ke belakang untuk membuat kopi sesuai permintaan.
4. Setelah selesai, kasir tersebut akan kembali kepada pelanggan dengan membawa kopinya, dalam hal ini adalah kapucino dengan ekstra karamel. Yummy!

Di dunia nyata, aplikasi bisa lebih rumit dari sekadar satu transaksi dengan satu server, bahkan bisa menjadi sangat kompleks ketika diterapkan ke dalam solusi bisnis yang mapan.

Nah, untuk menghindari kompleksitas ini, mari kita mulai dengan yang simpel, seperti konsep utama di AWS, yakni *pay for what you use* (bayar untuk apa yang Anda gunakan).

Prinsip ini sangat tepat dan masuk akal dengan skenario kedai kopi kita. Pegawai hanya dibayar saat mereka bekerja di toko. Jika mereka tidak bekerja, maka tidak ada gaji. Pemilik kedai dapat memutuskan berapa banyak pegawai yang dia butuhkan lalu memberikan mereka upah sesuai jam kerja.

Sebagai contoh, kedai kopi tersebut akan merilis minuman baru, Robusta. *Delicioso!*

Untuk mengantisipasi peluncuran ini, Anda bisa mempekerjakan selusin pegawai sepanjang hari guna berjaga-jaga jika pelanggan membludak berdatangan secara tak terduga di hari spesial tersebut. Hanya saja, pelanggan tidak selalu akan membludak setiap saat, bukan?

Tapi tahukah Anda? Inilah yang sebenarnya terjadi di data center on-premise (lokal). Anda tidak bisa hanya sekadar menjentikkan jari lalu *voila!* Kapasitas Anda berlipat ganda dengan sendirinya. *Nope*. Banyak proses administratif yang perlu Anda lakukan dan berujung pada mahalnya biaya yang perlu Anda keluarkan.

Dengan Amazon Web Service, Anda tidak perlu membayar uang muka untuk apa pun dan tidak perlu khawatir tentang kendala kapasitas.

Oke. Sekarang kita menemukan istilah baru, data center on-premise. Apa itu? Mari kita kupas.

Pertama, *data center*. Berdasarkan *website* Cisco--salah satu perusahaan telekomunikasi global--data center adalah fasilitas yang digunakan perusahaan untuk menempatkan aplikasi dan data penting mereka. Komponen utama dari data center adalah router, switch, firewall, sistem penyimpanan, dan juga server [1].

Sementara *on-premise* mengacu pada penyimpanan dan pemeliharaan data di server lokal atau pribadi.

Lanjut ke prinsip berikutnya, yaitu *pay for what you need* (bayar untuk apa yang Anda butuhkan). Misal ketika Anda membutuhkan sebuah instance atau mungkin barista, cukup dengan klik sebuah tombol ajaib segera mereka pun seketika tersedia untuk Anda. Dan ketika tak membutuhkannya, klik tombol lagi kemudian mereka akan pergi sesaat kemudian sehingga Anda tak perlu membayarnya lagi.

Prinsip ini menjadi nilai utama di AWS. Itulah alasan sebenarnya kelas ini dihadirkan, yakni untuk membantu Anda memahami bagaimana AWS dibangun untuk membantu Anda menjalankan bisnis dengan lebih baik.

Tetaplah pahami dan ikuti kelas ini dengan saksama karena kita akan segera menyelami konsep-konsep tersebut lebih dalam serta membantu Anda melangkah menuju Cloud Practitioner. Semangat!

MODUL 1 CLOUD

me

Sebelum kita melangkah ke bagian AWS lebih dalam, mari kita persempit lingkupan pembahasannya, yaitu mengenai definisi kerja dari *cloud*.

Apa yang tebersit di pikiran Anda saat mendengar kata *cloud*? Jika Anda membayangkan gumpalan awan, Anda tidak salah karena memang kebanyakan ilustrasi jaringan komputer memuat simbol cloud atau awan yang merupakan perumpamaan dari internet.

Tapi, kenapa internet disimbolkan dengan awan? Nah, menurut newyorker.com, simbol awan dipakai karena merepresentasikan keberadaan data kita ada di suatu tempat di luar sana, mengambang, melayang, nirkabel, tersedia di mana saja dan kapan saja ketika kita membutuhkannya. Istilah awan juga menarik karena merupakan kebalikan dari dunia material yang riil seperti colokan, kabel, hard disk, dsb [2].

Sekarang, apa yang dimaksud dengan *cloud computing*? Menurut AWS, cloud computing--di kelas ini kita menyebutnya dengan komputasi cloud--adalah penggunaan sesuai kebutuhan (*on-demand*) sumber daya IT melalui internet dengan harga sesuai pemakaian (*pay-as-you-go*) [3]. Mari kita uraikan definisi tersebut.

- **Penggunaan sesuai kebutuhan**, ini menunjukkan bahwa AWS memiliki sumber daya yang Anda butuhkan kapan pun dan di mana pun. Tak perlu memberi tahu AWS kapan Anda akan membutuhkannya. Jika tiba-tiba Anda membutuhkan 300 server virtual, lakukan beberapa klik saja dan mereka pun akan langsung tersedia.

  Demikian pula ketika Anda membutuhkan penyimpanan sebesar 2000 terabyte misalnya, silakan gunakan penyimpanan tersebut sesuai kebutuhan. Saat tidak

membutuhkannya lagi, klik untuk melepasnya. Semudah itu. Fleksibilitas semacam ini tak mungkin Anda dapatkan di data center on-premise (lokal).

- **Sumber daya IT** sebenarnya adalah bagian besar dari filosofi AWS. AWS memiliki ratusan layanan unggulan. Mengapa begitu banyak? Jawabannya sangat sederhana: karena bisnis membutuhkannya. Di AWS pun ada beberapa sumber daya IT yang sudah umum digunakan di sejumlah perusahaan. Contohnya begini. Anggaplah Anda menggunakan database (basis data) MySQL di perusahaan tempat Anda bekerja.

  Apakah kemampuan untuk menginstal mesin MySQL membuat perusahaan Anda menjadi lebih bersaing daripada kompetitor? Mungkin tidak.

  Apakah dengan menyimpan *backup* (cadangan) membuat perusahaan Anda lebih unggul dari perusahaan lain? Sekali lagi, diragukan.

  Data di dalam database, cara membangun tabel, dan mengelola strukturnya itulah yang membuat perusahaan Anda menjadi pembeda di antara pesaing yang lain.

  Di AWS, hal semacam itu disebut *undifferentiated heavy lifting* (semua proses kerja yang tidak menambah nilai bagi perusahaan) dari IT. Proses kerja IT yang umum--seperti instalasi OS, pembaruan perangkat lunak, dll--seringkali dilakukan secara repetitif dan akhirnya memakan waktu. Nah, hadirlah AWS untuk membantu Anda menangani hal-hal semacam itu. Jadi, Anda bisa fokus pada bisnis perusahaan Anda.

- **Melalui internet**, ini menyiratkan bahwa Anda dapat mengakses sumber daya tersebut--yang telah dipaparkan sebelumnya--menggunakan web browser atau secara terprogram.

Tak perlu kontrak apa pun, cukup bayar dengan mekanisme *pay-as-you-go* (sesuai pemakaian). Sama halnya seperti skenario kedai kopi kita. Anda tak perlu mempekerjakan banyak pegawai terus-menerus, cukup di waktu jam sibuk saja. Bahkan di waktu malam hingga pagi, Anda tak butuh pegawai sama sekali karena kedai kopi tutup. Itulah *cloud computing* alias komputasi cloud.

Sebelum datangnya komputasi cloud, perusahaan yang ingin membuat data center harus membangunnya sendiri dan harus memprediksikan beban kerja layanan. Ini tentu akan sangat merepotkan karena perlu biaya yang cukup besar.

Sebuah data center biasanya terdiri dari rak-rak komputer yang berjajar; jaringan yang kompleks; dan juga sistem penyimpanan yang terkelola. Selain itu, perusahaan juga harus mengeluarkan dana untuk membayar sewa bangunan, kebersihan, listrik, pendingin, dan keamanan.

Dan meskipun data center telah siap digunakan, perusahaan wajib memastikan bahwa data center tersebut mampu melayani kebutuhan beban kerja. Kenapa begitu?

Anggaplah begini. Suatu perusahaan telah berhasil membangun data center untuk menopang kebutuhan *website*-nya. Setiap hari website tersebut rata-rata diakses oleh 10.000 pengunjung, namun umumnya pada hari minggu jumlahnya hanya 5.000 saja. Angka ini berubah lagi di hari-hari spesial seperti malam tahun baru ketika tiba-tiba 30.000 pengunjung membanjirinya. Lantas website tersebut pun *down* beberapa kali.

Nah, bagaimana cara mengatasi masalah ini? Tenang, dengan komputasi cloud semua akan terselesaikan. Anda bisa menyesuaikan kemampuan data center sesuai kebutuhan beban kerja. Tidak akan kekurangan atau berlebihan. Nanti kita akan membahas ini lebih detail. Sekarang, mari kita menyimak materi berikutnya!

Model Penerapan untuk Komputasi Cloud

Saat memilih strategi untuk menerapkan cloud, Anda harus mempertimbangkan beberapa faktor, seperti komponen aplikasi cloud yang diperlukan, layanan manajemen sumber daya yang dibutuhkan, dan setiap persyaratan infrastruktur IT.

Tiga model penerapan komputasi cloud adalah cloud-based, on-premises (lokal), dan hybrid. Mari kita uraikan masing-masing model tersebut:

- **Cloud-based Deployment**
  Dalam model penerapan *cloud-based*, Anda dapat merancang, membangun, dan menjalankan aplikasi baru di cloud. Anda pun dapat memigrasikan aplikasi yang telah ada ke cloud.

  Anda dapat membangun aplikasi tersebut pada *low-level infrastructure* (infrastruktur tingkat rendah) yang mana memerlukan staf IT Anda untuk mengelolanya. Atau dengan alternatif lain, yaitu

menggunakan *higher-level services* (layanan dengan tingkat lebih tinggi) sehingga mengurangi kebutuhan pengelolaan, arsitektur, dan *scaling* (penyesuaian kapasitas) pada infrastruktur Anda.

Misalnya, Anda dapat membuat aplikasi yang terdiri dari server virtual, database, dan komponen jaringan yang sepenuhnya berbasis di cloud.

- **On-premises Deployment**
  On-premises juga dikenal sebagai *private cloud* (cloud privat). Dalam model ini, sumber daya di-*deploy* (diterapkan) menggunakan layanan manajemen aplikasi dan teknologi virtualisasi pada data center pribadi sehingga penggunaan dan pemanfaatannya dapat meningkat.

- **Hybrid Deployment**
  Dalam penerapan hybrid, sumber daya berbasis cloud terhubung ke data center on-premises (lokal). Anda bisa gunakan pendekatan ini untuk beberapa situasi, seperti aplikasi lama yang memang lebih baik dikelola di on-premises atau mungkin karena peraturan pemerintah yang mengharuskan Anda menyimpan data tertentu di data center lokal.

  Nah, dengan skenario seperti ini, Anda dapat menyimpan aplikasi lama di on-premise selagi memanfaatkan data dan layanan yang berjalan di cloud.

Manfaat dari Komputasi Cloud
Ada beberapa hal yang perlu Anda pertimbangkan agar semakin yakin untuk memilih komputasi cloud sebagai solusi yang dapat menangani kebutuhan Anda dibandingkan dengan data center on-premise. Mari kita uraikan:

- **Ubah pengeluaran di muka menjadi pengeluaran variabel**
  Pengeluaran di muka (*upfront expense*) mengacu pada data center, server fisik, dan sumber daya lain yang perlu Anda investasikan sebelum Anda menggunakannya. Sedangkan pengeluaran variabel (*variable expense*) berarti Anda hanya membayar untuk sumber daya komputasi yang Anda konsumsi.

  Dengan mengambil pendekatan komputasi cloud yang menawarkan keuntungan biaya variabel, perusahaan dapat mengimplementasikan solusi inovatif sekaligus menghemat biaya.

- **Hentikan biaya pengelolaan dan pemeliharaan data center**
  Komputasi di data center sering kali mengharuskan Anda untuk mengeluarkan lebih banyak biaya dan waktu untuk mengelola infrastruktur dan server.

  Nah, dengan komputasi cloud, Anda tak perlu lagi khawatir akan tugas-tugas

ini. Dengan begitu, Anda dapat lebih fokus pada aplikasi dan pelanggan Anda.

- **Berhenti menebak kapasitas**
  Dengan komputasi cloud, Anda tak perlu memprediksi berapa banyak kapasitas infrastruktur yang Anda perlukan sebelum men-*deploy* aplikasi.

  Misalnya, Anda dapat meluncurkan Amazon EC2 instance dan cukup membayar untuk waktu komputasi yang digunakan. Daripada harus membayar sumber daya yang tak terpakai atau berurusan dengan kapasitas yang terbatas, dengan komputasi cloud, Anda dapat menggunakan kapasitas sesuai keinginan.

  Bahkan Anda juga dapat melakukan proses *scale in* (mengurangi) atau *scale out* (memperbanyak) kapasitas sesuai permintaan.

- **Manfaatkan skala ekonomi yang masif**
  Dengan menggunakan komputasi cloud, Anda dapat mewujudkan biaya variabel yang lebih rendah daripada yang dapat Anda peroleh dari data center on-premise.

  Penggunaan dari ratusan ribu pelangganlah yang memungkinkan AWS dapat mencapai skala ekonomi (economies of scale) yang lebih tinggi. Kemudian skala ekonomi ini diterjemahkan ke dalam harga pay-as-you-go yang lebih murah.

- **Tingkatkan kecepatan dan ketangkasan**
  Fleksibilitas dari penggunaan komputasi cloud memudahkan Anda untuk mengembangkan dan men-*deploy* aplikasi.

  Dengan komputasi cloud, Anda memiliki lebih banyak waktu untuk bereksperimen dan berinovasi. Tentu ini tak bisa Anda lakukan jika menggunakan data center on-premise. Misal untuk mendapatkan sumber daya baru, mungkin Anda memerlukan waktu berminggu-minggu. Sedangkan dengan AWS, sumber daya baru akan langsung siap diakses dalam hitungan menit.

- **Mendunia dalam hitungan menit**
  AWS Cloud memungkinkan Anda dapat meluncurkan aplikasi ke pelanggan di seluruh dunia dengan cepat sekaligus memberikan latensi yang rendah. Ini berarti meskipun Anda berada di belahan dunia yang berbeda dengan pelanggan, mereka tetap dapat mengakses aplikasi dengan waktu tunda (delay) yang minimal.

Nantikan ya. Anda pasti sudah tak sabar menjelajahi materi infrastruktur global AWS dengan lebih detail! Mari temukan beberapa layanan yang dapat digunakan untuk mengirim konten ke pelanggan di seluruh dunia. Penasaran? Mari kita lihat materi berikutnya!

Pengenalan ke Amazon Elastic Compute Cloud (Amazon EC2)

Di modul ini kita akan membahas secara mendalam tentang layanan yang disebut Amazon Elastic Compute Cloud (Amazon EC2).

Jika Anda ingat skenario kedai kopi kita, pegawai dan pelanggan adalah metafora untuk model *client-server*. Client mengirimkan permintaan ke server, server melakukan beberapa pekerjaan, dan kemudian mengirimkan tanggapan. Contoh tersebut sangat pas untuk kedai kopi.

Ide yang sama pun berlaku jika Anda memiliki bisnis lain. Baik itu bisnis perawatan kesehatan, manufaktur, asuransi, ataupun pengiriman konten video. Itu semua menggunakan model client-server untuk menyajikan produk, sumber daya, atau data ke pelanggan.

Maka dari itu, Anda membutuhkan server yang dapat memberikan kapasitas komputasi untuk menjalankan aplikasi dan menyediakan daya komputasi sesuai kebutuhan bisnis Anda. Di AWS server tersebut berbentuk virtual. Dan layanan yang dapat Anda gunakan untuk mendapatkan akses ke server virtual tersebut disebut dengan Amazon EC2.

Dengan menggunakan layanan EC2, Anda memiliki kapasitas komputasi yang fleksibel, hemat biaya, dan cepat dibandingkan dengan menjalankan server sendiri di data center *on-premise*.

Bayangkan, untuk mengaktifkan dan menjalankan sumber daya di *on-premise*, Anda memerlukan banyak waktu dan biaya. Coba kita uraikan bagaimana prosesnya.

1. Pertama, Anda harus melakukan banyak riset untuk mengetahui jenis server apa yang ingin dibeli dan berapa banyak yang diperlukan.
2. Setelah itu, Anda membelinya dengan biaya di muka yang cukup menguras kantong. Lalu masuklah ke proses yang memakan waktu, yaitu Anda mesti

menunggu beberapa minggu atau bahkan berbulan-bulan sampai server tersebut tersedia untuk Anda.
3. Anggaplah server tersebut sudah tiba di bangunan data center yang Anda miliki atau sewa.
4. Langkah selanjutnya, Anda perlu memasang, menyusun, dan menghubungkan semuanya.
5. Kemudian, pastikan server-server tersebut aman dan menyala dengan baik, barulah mereka siap untuk digunakan.

Hanya dengan cara itulah Anda bisa mulai menjalankan aplikasi di server ini. Satu kata: *"huft*!*"* Sangat melelahkan. Eh tapi, tunggu! Masih ada bagian terburuknya. Setelah membeli server-server ini, Anda terjebak dengan mereka, entah Anda menggunakannya secara maksimal atau tidak. Nah, tentu ini akan jauh berbeda jika Anda menggunakan AWS.

Amazon EC2 memberikan kapasitas komputasi yang aman dan dapat Anda ubah-ubah ukurannya di cloud. Masih ingat persoalan sulitnya mengaktifkan dan menjalankan sumber daya di *on-premise* pada materi sebelumnya?

Nah, dengan Amazon EC2, proses tersebut akan jauh lebih mudah. Jangan khawatir! AWS sudah menangani bagian-bagian yang sulit untuk Anda. AWS telah membangun dan mengamankan data center, membeli, menyusun, dan memasang server sehingga siap untuk Anda gunakan.

AWS terus mengoperasikan kapasitas komputasi dalam jumlah besar sehingga Anda dapat menggunakannya kapan pun dan berapa pun sesuai dengan porsi kapasitas yang Anda butuhkan. Anda hanya perlu membuat permintaan untuk EC2 instance sesuai keinginan dan saat itu juga mereka pun tersaji dalam hitungan menit. Di AWS, server virtual disebut sebagai *instance*.

Nah, jika telah selesai menggunakannya, Anda dapat menghentikan atau mengakhiri instance tersebut dengan mudah. Anda tidak perlu lagi khawatir akan terjebak dengan server yang tidak digunakan. Anda hanya harus membayar sesuai dengan apa yang

Anda gunakan saja (*pay for what you use*), bukannya saat instance berhenti atau berakhir.

Amazon EC2 berjalan di atas *host* (mesin fisik) yang dikelola oleh AWS menggunakan teknologi virtualisasi. Saat menjalankan instance, Anda tidak menggunakan keseluruhan mesin host untuk sendiri melainkan Anda akan berbagi mesin host dengan beberapa instance lainnya. Ini dikenal dengan nama *virtual machines* alias mesin virtual.

Hypervisor-lah yang bertanggung jawab untuk membagi sumber daya fisik yang mendasarinya di antara mesin virtual tersebut. Ini sepenuhnya dikelola oleh AWS. Ide berbagi perangkat keras yang mendasarinya ini disebut *multitenancy*. Hypervisor juga bertanggung jawab untuk mengisolasi mesin virtual satu sama lain saat mereka berbagi sumber daya dari host. Ini berarti EC2 instance tetap aman meskipun mereka berbagi sumber daya. Satu instance tidak akan mengetahui keberadaan instance lainnya walau mereka ada di host yang sama. Mereka tetap aman dan terpisah satu sama lain.

Amazon EC2 memberikan Anda banyak fleksibilitas dan kontrol. Tak hanya dapat menjalankan server baru atau menghentikannya sesuka hati, Anda juga memiliki kuasa atas konfigurasinya.

Misal pada saat Anda membuat EC2 instance. Anda dapat memilih OS (operating system/sistem operasi) yang Anda inginkan, baik itu Windows atau Linux. Anda juga dapat membuat ribuan instance EC2 sekaligus dengan perpaduan sistem operasi dan konfigurasi sehingga dapat mendukung berbagai aplikasi bisnis Anda.

Selain OS, Anda juga dapat melakukan instalasi perangkat lunak apa yang ingin dijalankan pada instance. Baik itu aplikasi bisnis internal, web sederhana, web yang kompleks, database (basis data), hingga perangkat lunak pihak ketiga seperti paket perangkat lunak perusahaan. Anda memiliki kendali penuh atas apa yang ada di instance tersebut.

Instance EC2 juga dapat diubah-ubah ukurannya. Anda dapat mulai dengan menggunakan *small instance* (instance dengan tipe small).

Ketika aplikasi yang Anda jalankan mulai membutuhkan kapasitas yang lebih besar, Anda dapat menambahkan lebih banyak memori dan CPU. Itulah yang dinamakan *vertical scaling* atau mengatur skala instance secara vertikal. Intinya, Anda dapat membuat instance lebih besar atau lebih kecil kapan pun Anda mau.

Bahkan tak hanya itu. Anda juga dapat mengontrol aspek jaringan dari EC2, seperti jenis permintaan apa yang diizinkan atau bagaimana instance dapat diakses (publik atau privat). Di modul berikutnya kita akan membahas lebih lanjut berkenaan jaringan.

Sekali lagi, Amazon EC2 berjalan dengan bantuan teknologi virtualisasi. Mungkin Anda sudah tak asing ya dengan istilah mesin virtual. *Yup!* Karena ini bukanlah sesuatu yang baru.

Namun AWS membuat proses penyediaan server menjadi lebih mudah dan lebih hemat melalui model *Compute as a Service* (CaaS) seperti Amazon EC2 ini. Dengan semua keuntungan tersebut, *programmer* dan bisnis dapat berinovasi lebih cepat.

Cara Kerja Amazon EC2

Mungkin kening Anda sempat sedikit mengerut, “Bagaimana cara kerja Amazon EC2?” Tak seperti server di data center yang memerlukan proses panjang, Amazon EC2 dapat digunakan dengan mudah dengan beberapa langkah saja.

1. **Luncurkan**
   Mulailah dengan memilih sebuah *template* dengan konfigurasi dasar untuk instance Anda. Konfigurasi dasar ini termasuk sistem operasi, server aplikasi, atau aplikasi lainnya. Anda juga dapat memilih tipe instance, yaitu konfigurasi perangkat keras tertentu dari instance Anda.

   Selagi menyiapkan peluncuran instance, tentukanlah pengaturan keamanan untuk mengontrol lalu lintas jaringan yang dapat mengalir masuk dan keluar instance Anda. Nanti kita akan menjelajahi fitur keamanan Amazon EC2 secara lebih detail di materi selanjutnya.

2. **Hubungkan**
   Anda dapat terhubung ke instance dengan beberapa cara. Program dan aplikasi Anda memiliki beberapa metode berbeda untuk terhubung dan bertukar data langsung ke instance. Anda dapat terhubung juga ke instance dengan mengaksesnya dari desktop.

3. **Gunakan**
   Setelah terhubung ke instance, Anda dapat mulai menggunakannya. Ada banyak hal yang bisa dilakukan dengan Amazon EC2 instance, seperti menginstal perangkat lunak, menambah penyimpanan, menyalin dan mengatur file, dll.

Masih banyak hal yang harus kita pelajari seputar Amazon EC2. Mari kita bahas di materi berikutnya!

Tipe Instance Amazon EC2

Setelah mempelajari tentang EC2 instance dan peran pentingnya di AWS, mari kita bahas perkara berbagai tipe instance yang tersedia. Pikirkan kembali analogi kita soal kedai kopi, Anda mungkin ingat bahwa EC2 instance itu seperti pegawai di kedai kopi. Mereka melayani permintaan *client*.

Jika ingin memiliki kedai kopi yang mampu melayani banyak pelanggan, maka kita membutuhkan banyak pegawai, bukan? Tentunya para pegawai tersebut tak bisa semuanya berperan sebagai kasir. Harus ada seseorang yang membuat minuman, mengurusi makanan, dan mungkin yang dapat membuat seni *latte* keren agar pelanggan suka.
Seperti bisnis yang lain, ada berbagai tugas khusus yang perlu diselesaikan dan kerap kali membutuhkan keahlian yang berbeda-beda. Jika ingin bisnis kita beroperasi seefisien mungkin, maka pastikan karyawan memiliki keahlian yang sesuai dengan peran mereka.
Di kedai kopi kita memiliki berbagai jenis karyawan beserta perannya. Sama halnya dengan itu, AWS pun memiliki berbagai tipe EC2 instance yang dapat Anda jalankan dan terapkan ke dalam lingkungan AWS Anda.
Setiap tipe instance dikelompokkan dalam satu *instance family* (keluarga instance) dan dioptimalkan untuk jenis tugas tertentu. Tipe instance menawarkan berbagai kombinasi dari kapasitas CPU, memori, penyimpanan, jaringan, serta memberi Anda fleksibilitas untuk memilih kombinasi sumber daya yang sesuai untuk aplikasi Anda.
Instance family di Amazon EC2 memiliki fungsi yang berbeda-beda. Di antaranya ada *general purpose*, *compute optimized*, *memory optimized*, *accelerated computing* (komputasi terakselerasi), dan *storage optimized*. Berikut uraiannya:

- **General purpose instances** (Instance tujuan umum)
  Tipe ini memberikan keseimbangan yang baik dari segi sumber daya komputasi, memori, dan jaringan. Selain itu, opsi ini juga dapat digunakan untuk berbagai beban kerja yang beragam seperti server aplikasi web atau repositori kode.

- **Compute optimized instances** (Instance teroptimasi untuk komputasi)
  Tipe yang satu ini ideal untuk tugas komputasi yang intensif dan berpusat pada prosesor dengan performa tinggi, seperti *server game*, HPC (high-performance computing/komputasi dengan performa tinggi), atau bahkan pemodelan ilmiah.

  Anda juga bisa menggunakan tipe compute optimized instances untuk beban kerja *batch processing* yang membutuhkan banyak proses transaksi di satu grup.

- **Memory optimized instances** (Instance teroptimasi untuk memori)
  Opsi ini didesain untuk memberikan performa tinggi untuk beban kerja yang memproses kumpulan data besar di dalam memori, seperti relasional dan nonrelasional database atau HPC (high-performance computing).

- **Accelerated computing instances** (Instance terakselerasi untuk komputasi)
  Tipe ini menggunakan perangkat keras akselerator untuk menjalankan beberapa fungsi secara lebih efisien dibandingkan dengan perangkat lunak yang berjalan pada CPU. Contohnya adalah penghitungan bilangan floating-

point, pemrosesan grafik, dan *data pattern matching* (pencocokan pola data).

- **Storage optimized instance** (Instance teroptimasi untuk penyimpanan)
  Opsi ini didesain untuk beban kerja yang membutuhkan akses *read* (baca) dan *write* (tulis) yang tinggi dan berurutan untuk kumpulan data yang besar di penyimpanan lokal.

  Contoh beban kerja yang sesuai untuk tipe ini mencakup sistem file terdistribusi, aplikasi data warehousing (gudang data), dan sistem *online transaction processing* (OLTP) berfrekuensi tinggi.

  Dalam komputasi, istilah *input/output operation per second* (IOPS) adalah metrik yang mengukur kinerja perangkat penyimpanan. Ini menunjukkan berapa banyak operasi *input* atau *output* yang dapat dilakukan oleh perangkat dalam satu detik.

  Singkatnya, Anda dapat menganggap operasi input sebagai data yang dimasukkan ke dalam sistem, seperti data yang dimasukkan ke dalam database. Sedangkan operasi output adalah data yang dihasilkan oleh sistem. Contoh output adalah hasil analitik yang dilakukan pada data dalam database.

  Jika Anda memiliki aplikasi yang memerlukan IOPS tinggi, *storage optimized instance* dapat memberikan kinerja yang lebih baik dibandingkan dengan tipe lain yang tak teroptimasi untuk jenis kasus penggunaan ini.

Jika dianalogikan ke dalam skenario kedai kopi, kasir itu akan menjadi memory optimized instance, barista menjadi compute optimized instance, dan si pembuat seni pada *latte* adalah accelerated computing instance. Itulah tipe instance pada Amazon EC2. Yuk kita lanjut ke materi berikutnya!

Harga Amazon EC2

Di materi sebelumnya kita sudah menelaah Amazon EC2 beserta tipe instance-nya. Namun masih ada persoalan lain yang belum kita bahas, yaitu harga. Sebelum dahi Anda berkerut dan bertanya, “Berapa biaya yang harus saya keluarkan?” Yuk langsung saja kita bahas.

AWS memiliki beberapa pilihan penagihan terkait Amazon EC2. Di antaranya adalah:

- **On-Demand** (Sesuai Permintaan)
  Opsi ini adalah yang paling dikenal, yaitu *On-Demand*. Anda hanya membayar selama instance berjalan--bisa per jam atau per detik--tergantung pada tipe instance dan sistem operasi yang Anda pilih.

  On-Demand sangat ideal untuk penggunaan jangka pendek, pengembangan dan pengujian aplikasi, serta beban kerja yang tidak dapat diprediksi dan diinterupsi. Selain itu, model harga ini juga biasa digunakan untuk yang baru memulai, menguji beban kerja, sekadar bereksperimen, atau mendapatkan rata-rata dasar pemakaian instance.

  Tak perlu kontrak, komitmen jangka panjang, pembayaran di muka, atau komunikasi dengan AWS sebelumnya untuk menggunakan pilihan penagihan yang satu ini.

- **Savings Plans** (Rencana Tabungan)
  Savings Plans memungkinkan Anda mengurangi biaya komputasi dengan berkomitmen terhadap *jumlah dolar per jam yang keluar* dan penggunaan komputasi yang konsisten untuk jangka waktu 1 *atau* 3 tahun. Setiap penggunaan di luar itu akan dikenakan tarif On-Demand biasa.

  Oleh karena itu, model penetapan harga ini dapat memberikan penghematan hingga 72% pada penggunaan komputasi AWS Anda terlepas dari *instance family* (keluarga instance), ukuran, OS, *tenancy* (penyewaan), atau region AWS.

  Model Ini juga berlaku untuk penggunaan AWS Fargate dan AWS Lambda yang merupakan opsi komputasi tanpa server yang akan kita bahas nanti.

  Nanti di kelas ini kita akan meninjau tentang AWS Cost Explorer, yaitu layanan yang memungkinkan Anda untuk memvisualisasikan, memahami, serta mengelola biaya dan penggunaan AWS Anda dari waktu ke waktu.

  Jika Anda sedang mempertimbangkan opsi Savings Plans, AWS Cost Explorer dapat menganalisis penggunaan Amazon EC2 Anda selama 7, 30, atau 60 hari terakhir. AWS Cost Explorer juga memberikan rekomendasi yang disesuaikan untuk Savings Plans.

  Rekomendasi ini dapat memperkirakan seberapa banyak Anda dapat menghemat biaya bulanan berdasarkan penggunaan Amazon EC2 sebelumnya dan jumlah komitmen per jam dalam 1 atau 3 tahun.

- **Reserved Instances** (Instance Terpesan)
  Reserved Instances menawarkan diskon penagihan yang diterapkan untuk instance On-Demand dengan berkomitmen terhadap *tingkat penggunaan* untuk jangka waktu 1 atau 3 tahun.

  Ada beberapa opsi yang tersedia: Standard Reserved dan Convertible Reserved Instances (Instance Terpesan Standar dan Terpesan Konvertibel) untuk jangka waktu 1 atau 3 tahun. Dan juga tersedia Scheduled Reserved Instance (Instance Terpesan Terjadwal) untuk jangka waktu 1 tahun saja.

  Opsi ini cocok untuk beban kerja dengan kondisi yang stabil atau dapat diprediksi. *Reserved Instance* menawarkan diskon hingga 75% dibandingkan dengan opsi On-Demand.

  Terdapat tiga opsi pembayaran pada Reserved Instances:
  1. *All upfront* (semua di muka), yaitu Anda membayarnya secara penuh saat Anda berkomitmen.
  2. *Partial upfront* (sebagian di muka), di mana Anda membayar sebagian di awal.
  3. *No upfront* (tanpa uang muka), di mana Anda tak membayar apa pun di muka.

  Ketika Reserved Instance berakhir, Anda tetap bisa menggunakan Amazon EC2 instance tanpa gangguan. Namun akan dikenai tarif On-Demand hingga Anda menghentikannya atau membeli Reserved Instance baru yang sesuai dengan atribut instance (tipe instance, region, tenancy (penyewaan), dan platform).

- **Spot Instances** (Instance Spot)
  Spot Instances menggunakan kapasitas komputasi Amazon EC2 yang tak terpakai dan menawarkan penghematan biaya hingga 90% dari harga On-Demand. Opsi ini sangat ideal untuk beban kerja dengan waktu mulai dan akhir yang fleksibel dan tak masalah dengan interupsi.

  Jika Anda mengajukan Spot Instances dan kapasitas Amazon EC2 sedang tersedia, maka instance akan diluncurkan. Namun jika tidak, permintaan akan gagal sampai kapasitas tersedia kembali.

  Setelah Anda meluncurkan Spot Instances, AWS dapat mengklaim kembali instance tersebut kapan pun ketika mereka membutuhkannya.

  AWS akan memberikan waktu peringatan dua menit sebelumnya untuk Anda menyelesaikan pekerjaan. Anda selalu dapat melanjutkannya nanti jika perlu. Jadi, saat memilih opsi ini, pastikan beban kerja Anda dapat menerima

interupsi.

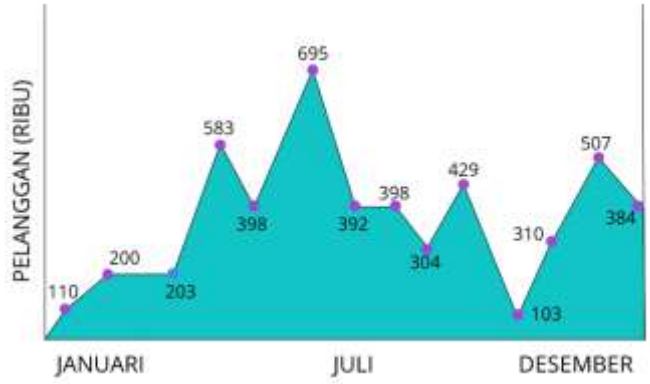
- **Dedicated Hosts** (*Host* Khusus)
  Dedicated Hosts merupakan server fisik dari kapasitas Amazon EC2 instance yang didedikasikan sepenuhnya untuk Anda gunakan.

  Opsi ini biasanya digunakan untuk memenuhi persyaratan *compliance* (kepatuhan) tertentu dan tidak ada orang lain yang akan berbagi sewa dari server fisik tersebut.

  Pada opsi ini Anda dapat menggunakan lisensi perangkat lunak per-*socket*, per-*core*, atau per-VM yang Anda punya untuk membantu menjaga persyaratan lisensi yang terikat dengan server.

Itulah mengenai opsi harga pada Amazon EC2. Anda bisa memilih opsi apa pun tergantung dengan kasus penggunaannya. Jika Anda memiliki beban kerja yang tak masalah dengan interupsi, pilihlah *Spot Instances*. Atau Anda dapat menghemat dengan melakukan pembayaran lebih awal dan mengunci minimum tingkat penggunaan dengan *Reserved Instance*.
Dari semua opsi harga Amazon EC2 yang telah dibahas, opsi *Dedicated Hosts* adalah yang paling mahal.

Penyesuaian Kapasitas Amazon EC2
Sampai sini, kita sudah memiliki pemahaman tentang dasar-dasar Amazon EC2 dan bagaimana itu dapat membantu kita dalam menangani kebutuhan komputasi apa pun, seperti membuat kopi di skenario kedai kopi kita. Minuman kopi mewakili apa pun yang dapat dihasilkan oleh *instance* Anda.
Di modul ini kita akan berbincang mengenai manfaat utama lainnya dari AWS, yaitu skalabilitas dan elastisitas. Dua hal ini mengacu pada bagaimana kapasitas dapat bertambah dan berkurang sesuai kebutuhan. Itulah yang menjadi dilema jika menggunakan data center *on-premise*.
Jika Anda memiliki bisnis seperti 99% dari semua bisnis di dunia, maka pastinya beban kerja aplikasi Anda bervariasi dari waktu ke waktu. Ada kalanya Anda mengalami masa-masa sibuk, ada juga di mana aplikasi Anda sepi.



Jika Anda menggunakan data center *on-premise*, maka akan muncul satu pertanyaan yang selalu mencemaskan, “Berapa tepatnya jumlah perangkat keras yang harus dibeli?”

Kalau Anda membelinya sesuai dengan jumlah penggunaan rata-rata, maka tidak akan terjadi pemborosan biaya. Namun sayangnya, ketika beban kerja melonjak, Anda tak akan memiliki perangkat keras yang cukup untuk melayani pelanggan. Akan banyak keluhan yang datang dari pelanggan karena mereka kesulitan mengakses aplikasi Anda.

Nah, bagaimana jika Anda membeli perangkat keras melebihi beban kerja tertinggi dan berharap semua berjalan dengan lancar? *Hmm.* Mungkin pelanggan Anda akan senang. Tapi Anda akan memiliki sumber daya yang menganggur hampir sepanjang tahun, tentu ini akan terbuang sia-sia. Anda tak ingin terjebak dengan mereka, bukan? Jadi, bagaimana menyelesaikan masalah ini di *on-premise*? *Ups.* Tidak bisa. Namun jangan khawatir! Itulah kenapa kita belajar mengenai AWS di kelas ini.

Sebenarnya dengan AWS, Anda dapat mengatur beban kerja berdasarkan kondisi yang Anda tentukan agar sesuai dengan permintaan. Nah, sekarang pelanggan Anda akan senang karena mereka selalu bisa mengakses aplikasi Anda. Anda pun akan bahagia karena tak lagi harus berkutat dengan prediksi dan biaya yang besar di awal. Itulah sedikit gambaran mengenai penyesuaian kapasitas alias *scaling*--selebihnya kita akan sering menggunakan kata *scaling*. Lalu, bagaimana cara kerjanya di AWS? Mari kita sangkut pautkan dengan skenario kedai kopi.

Di kedai kopi kita menggunakan *decoupled system* (sistem yang terpisah). Artinya, setiap pegawai tidak melakukan semua pekerjaan sendiri. Seorang kasir hanya bertugas untuk menerima pesanan. Dan barista bertanggung jawab untuk membuat minuman.

Nah, masalah pertama yang harus kita pecahkan adalah membuat rencana jika suatu saat terjadi bencana. Ada kutipan hebat dari Werner Vogels--VP dan CTO di Amazon--yang mengatakan, "Semuanya gagal setiap saat, jadi rencanakan kegagalan sehingga tidak ada yang gagal."

Dengan kata lain, tanyakan pada diri sendiri apa yang akan terjadi jika kita kehilangan *instance* kasir di kedai kopi? Mungkin, kita tidak dapat melayani pelanggan hingga *instance* lain aktif dan lanjut bekerja.

AWS membuatnya sangat sederhana. Dengan menggunakan metode terprogram yang sama seperti instance kasir yang asli, kita dapat membuat kasir kedua. Sehingga, jika salah satu instance tersebut mengalami kegagalan, kita memiliki instance lain yang sudah berada di garis depan dan siap menerima pesanan.

Dengan begitu, pelanggan tak akan pernah kehilangan kasir. Hal yang sama pun bisa Anda lakukan terhadap instance barista jika Anda mau.

Nah, sekarang kita memiliki sistem yang *highly available* (sangat tersedia) tanpa satu pun titik celah kegagalan. Selama jumlah pelanggan yang ada di antrean selaras dengan kapasitas instance, kita baik-baik saja.

Tapi, kita semua tahu bahwa jumlah pelanggan yang sedang mengantre tak dapat diprediksi, bukan? Jadi, mari kita lihat apa yang akan terjadi ketika kita memiliki banyak pelanggan, yaitu dengan peningkatan kapasitas berdasarkan permintaan.

Mari kita bahas apa yang sudah kita singgung di awal modul, tentang skalabilitas. Skalabilitas berarti kapasitas dari arsitektur Anda dapat merespons terhadap perubahan permintaan dengan melakukan *scaling out* atau *scaling in*--keduanya akan kita bahas nanti.

Anda cukup membayar sumber daya yang Anda gunakan dan tak perlu lagi khawatir akan kekurangan kapasitas komputasi untuk memenuhi kebutuhan Anda.

Jika ingin proses *scaling* terjadi secara otomatis untuk instance EC2, maka layanan AWS yang tepat adalah Amazon EC2 Auto Scaling.

Amazon EC2 Auto Scaling

Pernahkah Anda mencoba mengakses sebuah website namun halaman tersebut tak dapat memuat info dan malah sering kali menunjukkan eror seperti *timeout* (kehabisan waktu). Itu artinya, website tersebut terlalu banyak menerima permintaan masuk sehingga tak dapat menanganinya lagi. Maka dari itu, hadirlah solusi Amazon EC2 Auto Scaling.

Amazon EC2 Auto Scaling memudahkan Anda untuk menambah atau menghapus Amazon EC2 instances secara otomatis sesuai kebutuhan. Dengan begitu, Anda dapat membuat aplikasi selalu tersedia.

Dengan menggunakan Amazon EC2 Auto Scaling, Anda dapat menggunakan dua pendekatan:

- *Dynamic scaling*, yaitu merespons terhadap perubahan permintaan.
- *Predictive scaling*, yaitu secara otomatis menjadwalkan jumlah Amazon EC2 instances yang tepat berdasarkan prediksi permintaan.

Catatan: Anda pun dapat menggunakan dynamic scaling dan predictive scaling secara bersamaan agar dapat melakukan scaling arsitektur dengan lebih cepat.

Sekarang, mari kita belajar tentang beberapa cara untuk menangani permintaan yang melonjak. Anda dapat melakukan *scaling up/vertical scaling* atau *scaling out/horizontal scaling*.

- **Scaling up**

Scaling up artinya menambahkan lebih banyak daya pada mesin yang sedang berjalan. Saat pelanggan kedai kopi Anda semakin banyak, *instance* kasir yang menjadi lebih besar bukanlah solusinya karena kasir tetap tidak dapat menerima pesanan pelanggan dengan lebih cepat. Karena terkadang, kecepatan menerima pesanan itu tergantung pada pelanggan, bukan kasir.

Lantas apa solusinya? Tentu dengan memperbanyak pegawai!

- **Scaling Out**

Sederhananya, *scaling out* artinya menambahkan lebih banyak instance agar dapat menangani permintaan.
Coba perhatikan gambar di atas. Kenapa terdapat lebih banyak instance kasir daripada instance barista? Nah, dalam kasus ini, jumlah tugas yang dapat diselesaikan oleh instance barista masih dapat ditangani dengan baik daripada instance kasir.

Salah satu keunggulan dengan *decoupling the system* (memisahkan sistem) adalah Anda bisa mendapatkan jumlah daya yang tepat untuk setiap bagian dari proses daripada harus menyediakan terlalu banyak instance.

Oke, sepertinya kita baru saja membereskan antrean tersebut. Saat kedai kopi

Anda sudah sepi pelanggan, Anda dapat menyuruh para pegawai tambahan tersebut pulang atau menghentikan instance-nya.

Catatan: Selain *scaling up* dan *scaling out*, ada juga istilah *scaling down* dan *scaling in*. Dua hal ini adalah kebalikan dari yang telah kita bahas. Scaling down berarti Anda membuat daya komputasi menjadi lebih kecil, sementara scaling in berarti Anda mengurangi jumlah instance.

Kesimpulannya, dengan Amazon EC2 Auto Scaling Anda dapat menambahkan instance sesuai permintaan kemudian menonaktifkannya saat tak memerlukannya lagi. Ini berarti Anda akan selalu memiliki jumlah instance yang tepat setiap saat.

Auto Scaling Group

Di cloud, komputasi adalah sumber daya yang terprogram sehingga Anda dapat mengambil pendekatan yang lebih fleksibel untuk masalah *scaling* (penyesuaian kapasitas).

Nah, untuk melakukan *scaling*, kita perlu mengonfigurasi ukuran dari Auto Scaling group (grup Auto Scaling)--kumpulan Amazon EC2 instance untuk tujuan scaling dan manajemen secara otomatis.

Untuk mengaturnya, Anda perlu menentukan berbagai jenis konfigurasi, seperti *minimum capacity* (kapasitas minimum), *desired capacity* (kapasitas yang diinginkan), dan *maximum capacity* (kapasitas maksimum).

1. **Minimum capacity**
   Minimum capacity alias kapasitas minimum adalah jumlah Amazon EC2 instance yang diluncurkan segera setelah Anda membuat Auto Scaling group. Ambil contoh kita menentukan minimumnya 1. Ini berarti setidaknya harus ada 1 Amazon EC2 instance yang berjalan setiap saat.

2. **Desired capacity**
   Selain itu, Anda dapat mengisi *desired capacity* dengan 2 Amazon EC2 instance meskipun aplikasi Anda hanya memerlukan minimal 1 instance untuk dijalankan.

   Catatan: Jika Anda tidak menentukan jumlah *desired capacity* dalam Auto Scaling group, maka otomatis akan diatur menjadi *default* ke *minimum capacity* Anda.

3. **Maximum capacity**
   Konfigurasi ketiga yang dapat Anda atur adalah maximum capacity. Misalnya, Anda dapat mengonfigurasi Auto Scaling group untuk menyesuaikan dengan permintaan yang melonjak namun maksimum hanya untuk 4 Amazon EC2 instance.

Karena Amazon EC2 Auto Scaling menggunakan Amazon EC2 instance, Anda hanya membayar sesuai yang Anda gunakan. Tak hanya itu, Anda pun akan memiliki

arsitektur yang hemat biaya dan dapat memberikan pengalaman terbaik kepada pelanggan.

Mengarahkan Traffic dengan Balancing

Di modul sebelumnya kita telah berhasil memecahkan masalah *scaling* (penyesuaian kapasitas) dengan Amazon EC2 Auto Scaling. Tapi kita masih punya satu masalah lainnya terkait *traffic* (lalu lintas).

Mari lihat situasinya di skenario kedai kopi. Sekarang kita memiliki 3 *instance* kasir yang disiapkan untuk menangani masalah ramainya pelanggan.

Namun anehnya, kebanyakan dari mereka malah mengantre di satu instance kasir saja sehingga menyebabkan distribusi pelanggan yang tidak merata. Ini membuat instance kasir yang lain hanya terdiam dan tak melakukan apa pun sambil terus menunggu pesanan.

Masalah ini bisa terjadi karena saat pelanggan tersebut datang, mereka tak yakin harus menuju ke kasir yang mana.

Lantas apa solusinya?

Akan sangat membantu jika kita mempekerjakan satu pegawai yang bertugas untuk menerima dan mengonfirmasi reservasi dari para pelanggan saat masuk ke kedai kopi. Peran semacam ini biasa disebut dengan nama *host*--bukan mesin fisik yang kita bahas sebelumnya ya--dan biasanya ditempatkan di depan pintu kedai kopi.

Host akan senantiasa mengarahkan setiap pelanggan yang baru masuk untuk berbaris di kasir dengan antrean terpendek. Dengan demikian, antrean pun akan merata di seluruh kasir sehingga pelanggan dapat terlayani dengan efisien.

Ide yang sama pun berlaku di lingkungan AWS. Katakanlah Anda memiliki beberapa EC2 instance yang menjalankan program serupa. Anda perlu mengarahkan setiap permintaan yang masuk untuk menuju ke EC2 instance tertentu. Anda juga harus memastikan bahwa distribusi beban kerja merata di seluruh EC2 instance sehingga tak ada satu instance pun yang menganggur.

Proses dari apa yang sejak tadi kita bincangkan ini disebut dengan *load balancing* (menyeimbangkan beban). Sedangkan aplikasi yang dapat menerima permintaan lalu mengarahkannya ke instance untuk diproses disebut dengan *load balancer* (penyeimbang beban).

Load balancer bertindak sebagai satu titik kontak untuk semua *traffic* web yang masuk ke Auto Scaling group Anda. Ini berarti saat Anda menambah atau menghapus Amazon EC2 instance sebagai respons terhadap jumlah traffic yang masuk,

permintaan ini diarahkan ke load balancer terlebih dahulu. Barulah kemudian permintaan tersebut disebar ke berbagai sumber daya yang akan menanganinya.

Elastic Load Balancing

AWS memiliki layanan *load balancer* yang berkinerja tinggi, hemat biaya, *highly available* (sangat tersedia), dan dapat diskalakan secara otomatis. Tak usah Anda menginstal, mengelola, memperbarui, melakukan *scaling*, menangani kegagalan, dan ketersediaan layanannya. AWS yang mengurus itu semua.

Perkenalkan Elastic Load Balancing (ELB), yaitu layanan AWS yang secara otomatis mendistribusikan traffic aplikasi yang masuk ke berbagai sumber daya, seperti Amazon EC2 instance.

Elastic Load Balancing merupakan salah satu layanan terkelola pertama yang akan kita telaah dalam kelas ini. Layanan ini dirancang untuk mengatasi undifferentiated heavy lifting--telah kita bahas di modul 1--dari load balancing.

Sebagai permulaan, Elastic Load Balancing adalah *regional construct* (konstruksi regional). Ini berarti ELB berjalan di tingkat Region, bukan pada individu EC2 instance sehingga membuatnya *highly available* secara otomatis.

ELB dapat diskalakan secara otomatis sehingga mampu menangani kepadatan traffic tanpa berdampak pada biaya per jamnya. Elastic Load Balancing dapat bekerja sama dengan Amazon EC2 Auto Scaling untuk membantu memastikan aplikasi yang berjalan di Amazon EC2 dapat memberikan kinerja dan ketersediaan tinggi.

Mari kita ilustrasikan penggunaan ELB yang berkolaborasi bersama layanan Amazon EC2 Auto Scaling dalam menangani traffic.

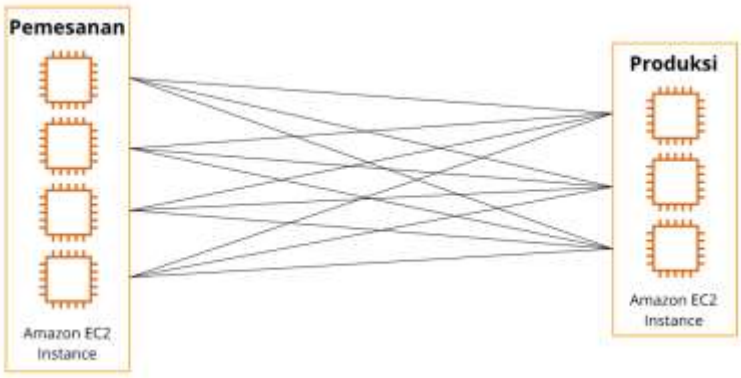
Anggaplah di suatu pagi aplikasi Anda memiliki *traffic* yang normal. Lalu di siang hari, Anda mengadakan *promo flash sale* secara besar-besaran di aplikasi bisnis Anda, tak lama kemudian lalu lintas pun semakin meningkat.

Saat *traffic* membanjiri aplikasi Anda, EC2 instance akan melakukan *scaling out*. Saat instance siap, Amazon EC2 Auto Scaling akan memberi tahu Elastic Load Balancing bahwa ia siap untuk menangani traffic.

Katakanlah malam tiba dan *promo flash sale* pun berakhir. Ini membuat traffic pada aplikasi Anda semakin berkurang sehingga Amazon EC2 Auto Scaling harus melakukan *scaling in.* Artinya, ada beberapa EC2 instance yang akan diakhiri.

Tapi sebelum itu, ELB akan berhenti mengirimkan traffic kepada instance yang akan diakhiri tersebut dan menunggu hingga permintaan selesai ditangani. Setelah selesai, barulah Amazon EC2 Auto Scaling bisa mengakhiri instance tanpa mengganggu aktivitas pelanggan yang ada.

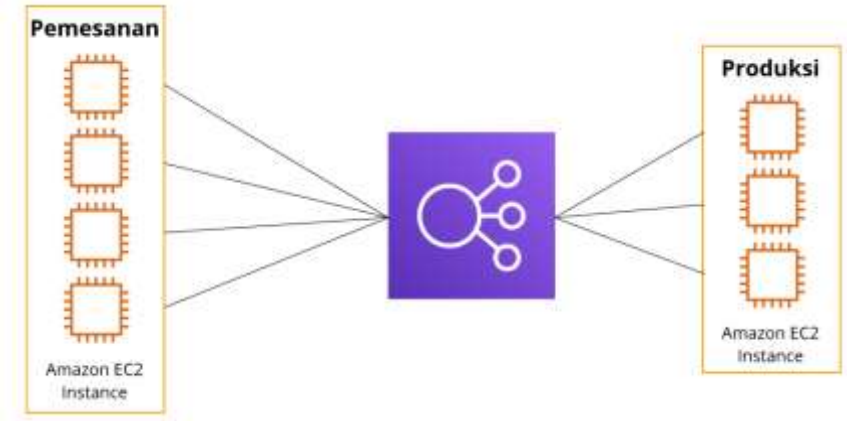
Selain untuk lalu lintas eksternal, Anda juga bisa menggunakan ELB untuk traffic di dalam arsitektur AWS. Mari kita lihat ilustrasikan bagaimana ELB berperan menangani komunikasi untuk setiap instance di antara bidang pemesanan dan produksi.



Sebelum menggunakan ELB, setiap instance di bidang pemesanan mengetahui seluruh instance produksi. Jadi, jika ada instance baru di bidang produksi, dia harus memberi tahu semua instance pemesanan bahwa sekarang dirinya dapat menerima traffic. Huh! Ini cukup rumit ya walau hanya ada 4 instance.

Sekarang bayangkan jika Anda memiliki ratusan instance di kedua bidang tersebut. Ampun! Tak sanggup lagi Anda bayangkan akan betapa kacaunya. Dengan kompleksitas seperti itu, mustahil rasanya membuat mereka tetap terhubung secara efisien.

Nah, di momen inilah ELB hadir memberikan solusi terbaik sehingga sekarang kita bisa menuntaskan kacau balau traffic pada bidang produksi. Mari kita pecahkan!



Kita telah menyinggung di awal bahwa ELB bersifat regional. Ini membuat setiap instance di bidang pemesanan dapat menggunakan satu URL saja dan ELB pun akan mengarahkannya ke instance produksi yang memiliki permintaan paling sedikit.

Lantas, bagaimana jika ada instance baru di bidang produksi? Mudah. Instance tersebut cukup memberi tahu ELB bahwa dirinya siap menerima traffic. Instance di bidang pemesanan tak perlu tahu dan tak akan peduli ada berapa banyak instance yang berjalan di bidang produksi. Sekali lagi, inilah yang dinamakan *decoupled architecture* (arsitektur yang terpisah).

Ada lebih banyak lagi hal yang dapat dilakukan oleh ELB yang nanti akan kita pelajari. Kesimpulannya, pilihlah layanan yang tepat untuk tugas yang tepat. Itulah salah satu alasan mengapa AWS menawarkan begitu banyak layanan yang beragam.

Selanjutnya, kita akan mengupas beberapa layanan lain yang mungkin bekerja lebih baik untuk beberapa arsitektur.

Messaging dan Queueing

Mari kita membahas tentang perpesanan dan antrean. Di skenario kedai kopi, ada dua jenis peran: kasir--yang menerima pesanan dari pelanggan--dan barista--yang membuat pesanan.

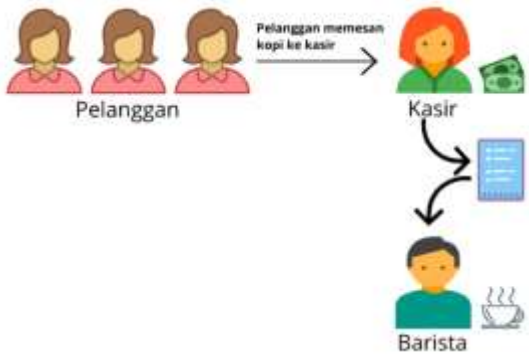
Proses interaksi di antara keduanya adalah seperti ini: kasir mengambil pesanan dari pelanggan, menuliskannya dengan pena dan kertas, dan mengirimkannya ke barista. Kemudian, barista mengambil kertas tersebut dan membuat pesanan.

Saat pesanan berikutnya masuk, prosesnya berulang. Proses ini akan bekerja dengan baik selama kasir dan barista selaras. Tetapi, apa yang akan terjadi jika kasir ingin menyerahkan pesanan pelanggan namun barista sedang istirahat atau sibuk dengan pesanan lain?

Kasir tersebut akan berhenti melayani sampai barista siap mengambil pesanan. Pada titik tertentu pesanan mungkin akan dibatalkan dan kasir pun melayani pelanggan berikutnya.

Coba amati! Kasus tersebut adalah proses yang tak sempurna. Ini karena jika kasir atau barista tidak sinkron, maka keseluruhan prosesnya akan terganggu sehingga menyebabkan lambannya penerimaan pesanan. Bahkan bisa sampai mengakibatkan kegagalan penyelesaian pesanan. Ah! Tentu Anda tak ingin ini terjadi, bukan?

Jadi, bagaimana jalan keluar yang paling efektif untuk kasus ini?



Solusi terbaik untuk menangani masalah ini adalah dengan menyediakan semacam *buffer* (antrean pesanan) ke dalam sistem. Daripada menyerahkan pesanan langsung ke barista, kasir akan menaruhnya ke semacam papan pesanan. Barista akan memeriksa buffer tersebut dan membuat minuman sesuai pesanan.

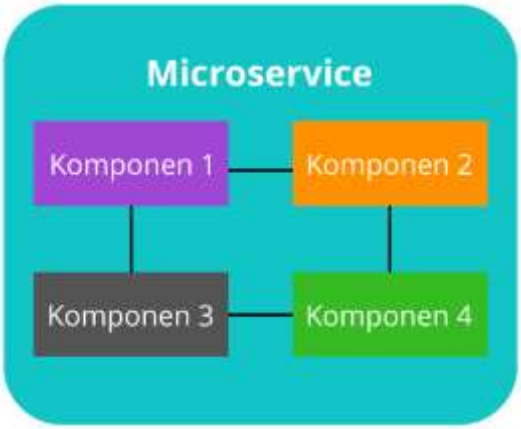
Setelah minuman tersaji dan memberikannya kepada pelanggan, barista akan menghapus pesanan yang sudah selesai tersebut dari buffer. Dengan begitu, selagi barista menyiapkan minuman, kasir dapat terus menerima pesanan baru dan menambahkannya ke buffer.

Ide dari menempatkan pesan ke dalam buffer disebut *messaging* dan *queueing*. Sama seperti kasir yang mengirimkan pesanan ke barista, aplikasi saling mengirim pesan untuk berkomunikasi. Ketika aplikasi berkomunikasi secara langsung seperti kasus kasir dan barista kita sebelumnya, maka itu disebut dengan *tightly coupled architecture*.

Ciri khas dari arsitektur yang *tightly coupled* adalah jika ada satu komponen yang gagal atau berubah, maka kegagalan ini memicu masalah untuk komponen lain atau bahkan keseluruhan sistem.

Misalnya kita punya aplikasi A yang mengirimkan pesan langsung ke aplikasi B. Jika aplikasi B mengalami kegagalan dan tak dapat menerima pesan tersebut, maka aplikasi A pun akan terkena eror juga.
Desain aplikasi seperti ini dapat dianggap sebagai pendekatan *monolithic application* alias aplikasi monolitik, yaitu saat berbagai komponen digabungkan menjadi satu kesatuan.

Nah, untuk mengatasi masalah ini kita harus membuat arsitektur yang lebih andal dengan *loosely coupled architecture*. Karakter dari arsitektur ini adalah jika satu komponen gagal, maka komponen tersebut akan diisolasi sehingga tak akan menyebabkan kegagalan beruntun ke seluruh sistem. Lebih baik yang ini, bukan?
Sama seperti di kedai kopi yang menyertakan *buffer* di antara kasir dan barista, kita juga dapat menggunakan komponen yang serupa, yaitu *message queue* (antrean pesan).
Pesan dikirim ke antrean oleh aplikasi A dan diproses oleh aplikasi B. Jika aplikasi B gagal, aplikasi A tidak mengalami gangguan apa pun. Pesan yang dikirim masih dapat dikirim ke antrean dan akan tetap berada di sana sampai akhirnya diproses.
Desain aplikasi semacam ini merupakan pendekatan dari *microservice* (layanan mikro), yaitu saat komponen dibuat menjadi *loosely coupled* sehingga dapat dikembangkan, di-*deploy* (diterapkan), dan dikelola secara independen. Setiap komponen mempunyai tugasnya masing-masing dan juga dapat berkomunikasi satu sama lain.



Oke, jadi layanan apa yang dapat kita gunakan di AWS?

Perkenalkan dua layanan AWS yang dapat membantu Anda dalam meraih arsitektur yang *loosely coupled*: Amazon Simple Queue Service (Amazon SQS) dan Amazon Simple Notification Service (Amazon SNS).
Tetapi sebelum kita menyelami keduanya, marilah memesan sebuah kopi terlebih dahulu di website kedai kopi kita. Sudah? Oke, kita tunggu saja. Seharusnya kita akan mendapatkan notifikasi saat pesanan itu siap. Lanjut!

Amazon Simple Queue Service (Amazon SQS)
Amazon Simple Queue Service (Amazon SQS) memungkinkan Anda untuk mengirim, menyimpan, dan menerima pesan antar komponen perangkat lunak dengan volume berapa pun tanpa perlu khawatir akan kehilangan pesan tersebut atau membutuhkan layanan lain untuk menyediakan pesan. Data yang terkandung di dalam pesan disebut *payload* dan itu dilindungi hingga terkirim.
Amazon SQS queue adalah tempat di mana pesan ditaruh sampai diproses. Cara kerjanya adalah aplikasi A akan mengirim sebuah pesan ke dalam *queue* lalu aplikasi B akan mengambilnya, memprosesnya, dan kemudian menghapusnya dari antrean.
AWS mengelola infrastruktur yang mendasarinya sehingga layanan ini dapat otomatis diskalakan, andal, serta mudah dikonfigurasi dan digunakan.
Jika Anda sukar memahaminya, bayangkan saja sebuah pesan sebagai sebuah pesanan kopi dan SQS queue adalah *buffer*, sebagaimana yang terdapat di skenario kedai kopi kita.

Amazon Simple Notification Service (Amazon SNS)
Amazon Simple Notification Service (Amazon SNS) juga digunakan untuk mengirimkan pesan ke layanan. Bedanya, ia juga dapat mengirimkan pemberitahuan ke pelanggan.
Proses tersebut dilakukan dengan cara yang berbeda, yaitu menggunakan model *publish*/*subscribe* alias pub/sub. Itu artinya Anda dapat membuat suatu saluran untuk menyampaikan pesan yang disebut dengan SNS topic. Jika ingin mempublikasikan pesan (*publish*), Anda bisa mengatur pelanggan (*subscribers*) yang akan menerima topik tersebut.
Dalam praktiknya, Anda dapat mengirim satu pesan ke SNS topic yang kemudian akan menyebar ke semua *subscribers* dalam sekali jalan. Subscribers dapat berupa *endpoint* (titik akhir) layanan lain, seperti SQS queue, fungsi AWS Lambda--akan kita bahas nanti, dan juga server web.
Selain itu, Amazon SNS dapat digunakan untuk menyebarkan notifikasi kepada pelanggan menggunakan *push notification* (pesan yang muncul di perangkat seluler), SMS, dan email.
Nah, begitu juga dengan skenario kedai kopi. Kita dapat mengirimkan pemberitahuan kepada pelanggan ketika pesanan mereka sudah siap untuk diambil, bisa berupa SMS atau *push notification*.
Wah! Sepertinya kopi yang telah kita pesan tadi sudah siap diambil. Apakah Anda menerima pemberitahuannya juga?

Studi Kasus: Amazon SNS

Katakanlah Anda membuat suatu buletin di kedai kopi berupa pembaruan yang mencakup informasi kupon, trivia kopi, dan produk baru. Semua informasi ini dikelompokkan menjadi satu topik karena ini adalah buletin tunggal. Semua pelanggan yang berlangganan buletin menerima pembaruan tentang topik-topik tersebut.

Tak lama kemudian, beberapa pelanggan Anda memberikan umpan balik bahwa mereka lebih suka menerima buletin terpisah hanya untuk topik tertentu saja, sesuai ketertarikan mereka. Anda pun mengabulkannya.

Sekarang buletin di kedai kopi telah terbagi menjadi tiga: kupon, trivia kopi, dan produk baru. Pelanggan pun akan menerima buletin sesuai dengan topik tertentu yang mereka inginkan. Mereka dapat berlangganan satu topik atau beberapa topik sekaligus.

Misalnya, pelanggan pertama hanya berlangganan topik kupon; pelanggan kedua hanya berlangganan topik trivia kopi; dan pelanggan ketiga berlangganan topik kopi trivia dan produk baru.

Meskipun contoh dari kedai kopi ini melibatkan pelanggan yang merupakan manusia, di Amazon SNS, pelanggan/subscribers dapat berupa server web, alamat email, AWS Lambda function, atau beberapa opsi lainnya.

Layanan Komputasi Tambahan

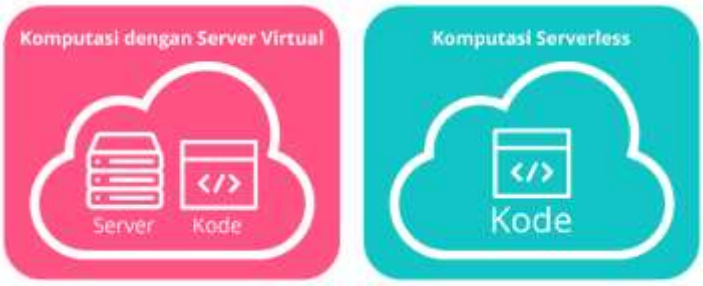
EC2 instance adalah mesin virtual yang dapat Anda gunakan di AWS. EC2 sangat ideal untuk semua jenis kasus penggunaan seperti menjalankan server web sederhana hingga menjalankan *high performance computing clusters* (klaster komputasi berkinerja tinggi).

- EC2 mengharuskan Anda untuk mengatur dan mengelola *instance* dari waktu ke waktu. Saat Anda menggunakan EC2, Anda bertanggung jawab untuk:
- Melakukan *patching* (memperbaiki masalah dengan memperbarui program komputer) saat *software package* (paket perangkat lunak) yang baru tersedia.
- Menyiapkan *scaling* (penyesuaian kapasitas).
- Merancang aplikasi untuk dijalankan dengan cara yang *highly available* (sangat tersedia).

Bahkan, jika Anda menggunakan data center *on-premise*, masih ada banyak hal lain yang harus Anda kelola.

Lalu, bagaimana solusinya? Mari kita melangkah ke materi berikutnya.

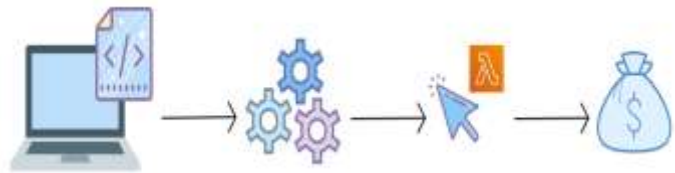
Komputasi Serverless



Anda mungkin akan bertanya-tanya, “Apa ada layanan komputasi lain di AWS yang tak perlu berkutat dengan pengelolaan?”
Di sinilah istilah *serverless* (tanpa server) hadir. Serverless berarti Anda tidak dapat melihat dan mengakses infrastruktur dasar yang menjalankan aplikasi Anda. Semua pengelolaan lingkungan yang mendasari penyediaan, *scaling*, *high availability* (ketersediaan tinggi), dan pemeliharaan sudah ditangani sehingga Anda bisa fokus pada aplikasi yang akan dijalankan.

AWS Lambda
AWS menawarkan beberapa opsi komputasi *serverless*, salah satunya adalah AWS Lambda. AWS Lambda adalah layanan yang memungkinkan Anda untuk menjalankan kode tanpa harus membuat atau mengelola server.
AWS Lambda dikelola sepenuhnya, dapat diskalakan secara otomatis, *highly available* (sangat tersedia), dan semua pemeliharaan dilakukan oleh AWS. Jika Anda memiliki 1 atau bahkan 1000 *trigger* (pemicu) yang masuk untuk memanggil *function* (fungsi), Lambda akan melakukan scaling terhadap function tersebut guna memenuhi permintaan.
AWS Lambda dirancang untuk menjalankan kode di bawah 15 menit sehingga layanan ini tak cocok untuk proses yang berjalan lama seperti *deep learning* misalnya. Layanan Ini lebih ideal untuk pemrosesan cepat seperti *web backend*, penanganan permintaan, atau pemrosesan laporan pengeluaran yang mana hanya membutuhkan waktu kurang dari 15 menit.

Cara Kerja AWS Lambda



Mungkin sempat terbayangkan oleh Anda, bagaimana AWS Lambda ini bekerja. Mari kita uraikan yuk.

1. Unggah kode Anda ke AWS Lambda.
2. Konfigurasikan kode Anda agar terpicu (*trigger*) dari sumber kejadian, seperti layanan AWS, aplikasi seluler, atau HTTP *endpoint* (titik akhir HTTP).
3. Kode berjalan hanya ketika mendapat *trigger*.
4. Cukup bayar sesuai waktu komputasi yang Anda gunakan. Misalnya, Anda mempunyai kode yang dapat mengubah ukuran gambar. Nah, Anda hanya akan membayar waktu komputasi yang digunakan untuk menjalankan fungsi pengubahan ukuran gambar saat ada yang mengunggah sebuah gambar baru.

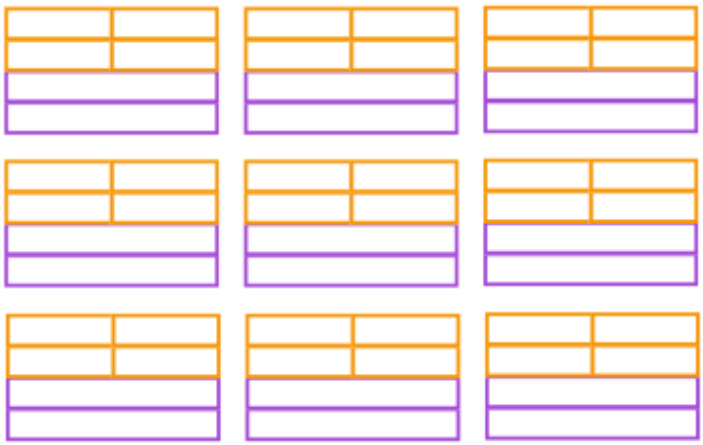
Jadi begitulah cara kerja AWS Lambda. Mari kita lanjutkan pembahasannya ke materi container.

Container

Jika Anda belum cukup siap untuk menggunakan *serverless* atau memerlukan akses ke infrastrukturnya namun tetap menginginkan efisiensi dan portabilitas, Anda bisa mencoba layanan *container* (kontainer) seperti Amazon Elastic Container Service (Amazon ECS) dan Amazon Elastic Kubernetes Service (Amazon EKS). Kita akan menjabarkan ini nanti ya.

Keduanya merupakan layanan *container orchestration* alias orkestrasi kontainer. Container dalam hal ini adalah *Docker container*. Apa itu?

Docker adalah platform perangkat lunak populer yang menggunakan virtualisasi sistem operasi untuk memudahkan Anda dalam membangun, menguji, dan men-*deploy* (menerapkan) aplikasi dengan cepat. Sementara container menyediakan cara untuk mengemas kode, konfigurasi, dan dependensi aplikasi Anda ke dalam satu objek.



Container bekerja di atas EC2 instance dan berjalan secara terpisah satu sama lain. Cara kerja container serupa dengan mesin virtual, namun dalam kasus ini, *host*-nya (server) adalah EC2 instance.

Saat menggunakan Docker container di AWS, Anda memerlukan proses untuk memulai, menyetop, memulai ulang, dan memantau container yang berjalan tidak hanya di 1 EC2 instance, melainkan beberapa yang disebut dengan cluster (klaster).

Proses menggarap tugas-tugas inilah yang disebut dengan *container orchestration* dan tentu akan sangat sulit jika melakukannya sendiri. Layanan orkestrasi dibuat untuk membantu mengelola container Anda.

Studi Kasus: Container

Misal *developer* aplikasi di suatu perusahaan memiliki infrastruktur komputer yang berbeda dengan staf operasi IT. Developer tersebut ingin memastikan bahwa lingkungan aplikasi tetap konsisten terlepas dari *deployment*-nya (penerapannya) sehingga dia pun menggunakan pendekatan *container*.

Container membantu developer tersebut mengurangi waktu yang dihabiskan untuk *debugging* (proses mengidentifikasi dan memperbaiki eror) aplikasi dan mendiagnosis perbedaan dalam lingkungan komputasi.

Saat menjalankan *containerized application* (aplikasi dalam container), penting untuk mempertimbangkan skalabilitas. Ini tergantung kepada setiap kasus penggunaan, Anda bisa saja:

- Menggunakan satu *host* dengan banyak container.
- Mengelola puluhan *host* dengan ratusan container.
- Mengurus mungkin ratusan *host* dengan ribuan container.

Dalam skala besar, bayangkan berapa lama waktu yang Anda butuhkan untuk memantau penggunaan memori, keamanan, *logging* (tindakan menyimpan log), dsb.
Untuk itulah hadir layanan *container orchestration* (orkestrasi container) yang membantu Anda men-*deploy* (menerapkan), mengelola, dan men-*scaling* aplikasi dalam container. Selanjutnya, kita akan mempelajari tentang dua layanan yang menyediakan container orchestration: Amazon Elastic Container Service dan Amazon Elastic Kubernetes Service

Amazon Elastic Container Service (Amazon ECS)
Amazon Elastic Container Service (Amazon ECS) adalah sistem manajemen container berkinerja tinggi yang dapat memungkinkan Anda untuk menjalankan dan melakukan scaling terhadap *containerized application* (aplikasi dalam container) di AWS.
Amazon ECS mendukung Docker container. AWS mendukung penggunaan *open-source Docker Community Edition* and *subscription-based Docker Enterprise Edition.* Dan juga, dengan Amazon ECS, Anda dapat menggunakan panggilan API untuk meluncurkan dan menghentikan aplikasi yang mendukung Docker.
API atau *Application Programming Interface* adalah perantara perangkat lunak yang memungkinkan dua aplikasi untuk berinteraksi satu sama lain. Kita tak akan membahas detailnya di sini. Jadi, mari lanjut ke materi berikutnya!

Amazon Elastic Kubernetes Service (Amazon EKS)
Amazon Elastic Kubernetes Service (Amazon EKS) adalah layanan terkelola sepenuhnya yang dapat Anda gunakan untuk menjalankan Kubernetes di AWS.
Kubernetes adalah perangkat lunak *open-source* (sumber terbuka) yang memungkinkan Anda untuk men-*deploy* (menerapkan) dan mengelola *containerized application* (aplikasi dalam container) dalam skala besar.
AWS secara aktif bekerja sama dengan komunitas Kubernetes--yang mengelola Kubernetes. Saat fitur dan fungsionalitas baru dirilis untuk aplikasi Kubernetes, Anda dapat dengan mudah menerapkan pembaruan tersebut ke aplikasi Anda yang dikelola oleh Amazon EKS.

Amazon Fargate
Baik Amazon ECS dan Amazon EKS, keduanya berjalan di atas EC2. Tetapi jika Anda tak ingin sibuk mengurusi EC2, Anda dapat menggunakan platform komputasi lainnya yang disebut dengan AWS Fargate.

AWS Fargate adalah platform komputasi serverless untuk Amazon ECS dan Amazon EKS. Saat menggunakan layanan ini, Anda tak perlu menyediakan atau mengelola server karena AWS Fargate yang mengelolanya untuk Anda.
Dengan begitu, Anda dapat lebih fokus pada inovasi dan pengembangan aplikasi. Bahkan Anda membayar hanya untuk sumber daya yang diperlukan dalam menjalankan container.
Masih bingung? Mari kita perjelas. Setiap layanan dapat Anda gunakan sesuai dengan kebutuhan.

- Jika Anda ingin menjalankan aplikasi dan menginginkan akses penuh ke sistem operasinya seperti Linux atau Windows, Anda bisa menggunakan **Amazon EC2**.
- Jika Anda ingin menjalankan fungsi yang berjalan singkat, aplikasi berbasis kejadian, dan Anda tak ingin mengelola infrastrukturnya sama sekali, gunakanlah layanan **AWS Lambda**.
- Jika Anda ingin menjalankan beban kerja berbasis Docker container di AWS, langkah yang perlu Anda lalui adalah:
    - Anda harus memilih layanan orkestrasinya terlebih dahulu. Anda bisa menggunakan **Amazon ECS** atau **Amazon EKS**.
    - Setelah memilih alat orkestrasinya, kemudian Anda perlu menentukan platformnya. Anda dapat menjalankan container pada **EC2 instance** yang Anda kelola sendiri atau dalam lingkungan *serverless* seperti **AWS Fargate** yang dikelola oleh AWS.

Itulah tadi beberapa opsi komputasi di AWS. Silakan lanjutkan ke modul berikutnya, yaitu berupa kesimpulan dari materi-materi yang telah kita pelajari sejauh ini. Semangat!

Ikhtisar

Tibalah kita di penghujung modul. Mari kita uraikan apa yang telah kita pelajari di modul ini:

- Hal pertama yang kita pelajari di modul ini berkenaan dengan komputasi cloud dan apa saja yang ditawarkan AWS.

  AWS mendefinisikan komputasi cloud sebagai penyajian sesuai permintaan (*on-demand*) sumber daya IT melalui internet dengan harga sesuai pemakaian (*pay-as-you-go*).

  Ini berarti Anda dapat membuat permintaan untuk sumber daya IT seperti komputasi, jaringan, penyimpanan, analitik, atau jenis sumber daya lainnya. Alih-alih membayar sumber daya tersebut di muka, Anda cukup membayarnya pada setiap akhir bulan.

- AWS menawarkan banyak layanan dan kategori yang dapat Anda gunakan. Sejauh ini kita telah membahas beberapa layanan, salah satunya adalah

tentang Amazon EC2. Dengan EC2, Anda dapat membuat dan menghapus server virtual secara dinamis yang disebut dengan EC2 instance.

- Saat Anda meluncurkan EC2 instance, Anda dapat memilih *instance family*(keluarga instance) yang menentukan perangkat keras tempat instance tersebut berjalan. Sehingga, Anda dapat memiliki instance yang dibuat untuk tujuan tertentu. Kategorinya adalah:
    - ***General purpose*** (tujuan umum)
    - ***Compute optimized*** (teroptimasi untuk komputasi)
    - **M*emory optimized*** (teroptimasi untuk memori)
    - **A*ccelerated computing*** (terakselerasi untuk komputasi)
    - ***Storage optimized*** (teroptimasi untuk penyimpanan).

- Selanjutnya, Anda dapat melakukan scaling EC2 instance baik secara vertikal--dengan mengubah ukuran instance--atau secara horizontal--dengan meluncurkan instance baru. Anda dapat mengatur *horizontal scaling* secara otomatis menggunakan Amazon EC2 Auto Scaling.

- Setelah Anda melakukan *scaling* EC2 instance secara horizontal, Anda memerlukan sesuatu untuk mendistribusikan *traffic* yang masuk. Layanan yang dapat Anda gunakan adalah Elastic Load Balancer.

- Kemudian, EC2 instance memiliki model harga yang berbeda, di antaranya:
    - ***On-Demand*** adalah yang paling fleksibel dan tidak memiliki kontrak.
    - ***Spot Instances*** memungkinkan Anda untuk menggunakan kapasitas yang tak terpakai dengan tarif diskon.
    - ***Reserved Instances*** dapat memberikan diskon ketika Anda berkomitmen pada tingkat penggunaan tertentu.
    - ***Savings Plans*** juga akan memberikan Anda diskon saat berkomitmen pada tingkat penggunaan tertentu dan dapat diterapkan untuk EC2 instance, AWS Lambda dan AWS Fargate.

- Lalu, kita juga telah membahas layanan perpesanan. Ada dua layanan yang kita pelajari:
    - **Amazon Simple Queue Service** (Amazon SQS)
      SQS memungkinkan Anda untuk melakukan *decouple system components* (memisahkan komponen sistem). Pesan tetap berada dalam antrean sampai diproses atau dihapus.
    - **Amazon Simple Notification Service** (Amazon SNS)
      SNS digunakan untuk mengirim pesan seperti email, pesan teks, *push notification*, atau bahkan permintaan HTTP. Setelah di-*publish* (diterbitkan), pesan akan terkirim ke semua *subscribers* (pelanggan).

- Berikutnya, kita telah mengetahui bahwa AWS memiliki jenis layanan komputasi di luar server virtual seperti EC2. Ada layanan *container* (kontainer) seperti Amazon Elastic Container Service (Amazon ECS) dan Amazon Elastic Kubernetes Service (Amazon EKS).

  Keduanya merupakan layanan *container orchestration* (orkestrasi kontainer). Anda dapat menggunakan layanan tersebut dengan EC2 instance.

- Namun jika tidak ingin repot-repot mengelolanya, Anda bisa menggunakan AWS Fargate. Layanan ini memungkinkan Anda untuk menjalankan container di atas platform *serverless compute* (komputasi tanpa server).

  Terakhir, ada AWS Lambda. Layanan ini memungkinkan Anda untuk mengunggah kode dan mengonfigurasinya untuk berjalan berdasarkan *triggers* (pemicu). Anda akan dikenakan biaya hanya pada saat kode berjalan. Tak perlu container atau mesin virtual. Hanya kode dan konfigurasi.

Itulah sedikit rangkuman dari modul Komputasi di Cloud. Semoga dapat mencakup semua materi yang telah disampaikan. Sampai jumpa di modul berikutnya!

Materi Pendukung

Untuk mempelajari lebih lanjut tentang konsep yang kita eksplor di modul ini, silakan tinjau beberapa sumber berikut:

- Compute on AWS
- AWS Compute Blog
- AWS Compute Services
- Hands-On Tutorials: Compute
- Category Deep Dive: Serverless
- AWS Customer Stories: Serverless

MODUL 2

Pengenalan ke Infrastruktur Global dan Keandalan

Selamat datang! Untuk memulai modul ini, mari kita sedikit berbincang tentang *high availability* (ketersediaan tinggi).

Ceritanya begini. Katakanlah pelanggan Anda ingin menikmati secangkir *latte* hangat di kedai kopi kita. Namun sayangnya, hari ini tak berjalan seperti biasa. Ada parade perayaan keberhasilan migrasi *cloud* yang menghalangi jalan menuju ke sana dan akan berbaris tepat di depan kedai kopi tersebut.

Sebenarnya ini bagus karena siapa yang tak suka melihat balon dan bermandi hujan permen? Akan tetapi, hal ini juga dapat berdampak buruk terhadap bisnis kita karena saat parade berlangsung, pelanggan yang tadinya ingin datang ke kedai kopi akan dialihkan jalannya sehingga tidak bisa mampir. Akibatnya, pelanggan akan kecewa dan penjualan pun menurun.

Untungnya, kita telah berpikir jauh ke depan untuk menyiasati gangguan semacam ini. Tahukah Anda? Sebenarnya kedai kopi kita tidak hanya berada di satu tempat saja *loh*, melainkan terdapat juga di beberapa lokasi lain di seluruh kota.

Dengan demikian, Anda tak perlu khawatir lagi jika ada parade di sejumlah ruas jalan. Bahkan tak hanya untuk parade, halangan lain seperti banjir, bencana, atau apa pun yang bisa mencegah pelanggan ke kedai kopi, telah kita atasi.

Sekarang kedai kopi kita dapat selalu tersedia bagi pelanggan. Mereka tetap bisa mendapatkan secangkir *latte* dengan mengunjungi kedai kopi kita lainnya yang berada tidak terlalu jauh. Semuanya akan baik-baik saja, bukan? Kita tetap bisa menjalankan bisnis dan pelanggan tetap bisa mendapatkan kopi.

Nah, AWS pun telah melakukan hal yang serupa dengan itu, yaitu dengan menyiapkan infrastruktur global AWS.

Tahukah Anda hikmah cerita di atas? Janganlah kita menempatkan semua sumber daya hanya di satu data center saja. Karena jika terjadi hal yang tidak diinginkan pada data center tersebut--seperti pemadaman listrik atau bencana alam--semua aplikasi akan *down* (mati) sekaligus. Bahkan, memiliki dua data center pun tetap tidak cukup baik *loh*.

Solusinya adalah Anda membutuhkan *high availability* (ketersediaan tinggi) dan *fault tolerance* (toleransi terhadap kesalahan). High availability adalah kemampuan untuk memastikan bahwa sistem selalu bekerja dan dapat diakses dengan waktu henti yang minimal tanpa memerlukan intervensi manusia. Sedangkan fault tolerance berarti sistem masih mampu beroperasi meskipun beberapa komponen mengalami kegagalan.

Faktanya, AWS beroperasi di lokasi yang berbeda-beda di seluruh dunia yang disebut Region. Tapi sabar ya. Kita baru akan membicarakan hal ini secara mendalam di modul mendatang. Jadi, mari kita lanjutkan!

Infrastruktur Global AWS

Untuk memahami infrastruktur global AWS, yuk kita mulai dengan menguraikan kebutuhan dasar untuk memulai bisnis di masa sekarang ini. Apa saja? Tentu kita membutuhkan:

- Aplikasi yang harus berjalan.
- Konten yang perlu disimpan.
- Data yang perlu dianalisis.

Intinya, kita memiliki sesuatu yang harus berjalan dan beroperasi di suatu tempat. Sebelum adanya teknologi *Cloud*, perusahaan harus menjalankan aplikasi mereka di *on-premise* karena mereka tak punya pilihan lain. Namun setelah AWS hadir, mereka dapat menjalankannya di data center lain tanpa harus memilikinya.

Tetapi pembahasan kita di modul ini akan jauh lebih dari itu. Kita harus memahami masalah yang fundamental terkait data center.

Kejadian tak terduga seperti terputusnya koneksi ke data center dapat terjadi kapan pun. Ada banyak faktor yang dapat memengaruhi problem seperti ini, salah satunya adalah bencana.

Jika Anda menjalankan *on-premise*, apa yang akan Anda lakukan saat bencana melanda data center tersebut? Mungkin solusinya adalah membangun data center kedua. Tapi ini bukan solusi yang terbaik.

Harga gedung atau bangunan untuk membangun data center akan terlalu mahal bahkan bisa jadi menghentikan bisnis Anda. Belum lagi biaya untuk perangkat keras, karyawan, listrik, pemanas dan pendingin, serta keamanan.

Oleh sebab itu, sebagian besar perusahaan pada akhirnya hanya menyimpan data cadangan mereka di suatu tempat dan berharap bencana tidak akan pernah datang. Kita semua tahu, harapan bukanlah rencana bisnis yang baik.
Tapi, tenang! AWS dapat membantu Anda mengatasi persoalan tersebut. Solusinya adalah dengan membangun data center dalam kelompok besar yang disebut dengan **AWS Regions** (Wilayah/Region AWS).
Mari kita bahas bagaimana AWS Regions didesain.

AWS Regions

Diambil dari Jaringan Infrastruktur Global AWS.
AWS telah membangun Region di seluruh dunia agar permintaan traffic bisnis dapat dilayani oleh Region terdekat. Beberapa contoh lokasi Region adalah kota Paris, Tokyo, Sao Paulo, Dublin, Ohio, dll.
Di dalam Region, AWS memiliki beberapa data center yang berisi semua sumber daya yang dibutuhkan seperti komputasi, penyimpanan, jaringan, dll.
Setiap Region tersebut dapat terkoneksi ke Region lain melalui *high speed fiber network* (jaringan fiber berkecepatan tinggi) yang dikontrol oleh AWS--kita membutuhkannya untuk operasi yang benar-benar global dari satu lokasi ke lokasi lainnya di seluruh dunia.
Ketahuilah! AWS memungkinkan Anda untuk dapat memilih Region mana yang ingin dijalankan. Selain itu, setiap Region juga terisolasi dari Region lainnya. Ini berarti tidak akan ada data yang masuk atau keluar dari Region tersebut kecuali Anda secara eksplisit mengizinkan data itu untuk berpindah.
Misalnya begini. Anda memiliki suatu persyaratan *compliance* (kepatuhan) dari pemerintah yang menyatakan bahwa informasi keuangan Anda yang berada di kota Frankfurt tidak dapat meninggalkan negara Jerman.
Di sinilah bagaimana AWS benar-benar beroperasi secara *out of the box* alias unik. Di AWS:

- Setiap data yang disimpan di Region Frankfurt tidak pernah meninggalkan Region Frankfurt.
- Setiap data di Region London tidak pernah meninggalkan Region London.
- Setiap data yang berada di Sydney tidak pernah meninggalkan Region Sydney.

Intinya, semua data yang disimpan di dalam Region, tak akan pernah meninggalkan Region tersebut, *kecuali* Anda secara eksplisit--dengan kredensial dan izin yang tepat--meminta data tersebut untuk diekspor.
Regional data sovereignty atau kedaulatan data regional adalah bagian dari desain penting AWS Regions, di mana data mengikuti hukum dan undang-undang lokal negara tempat Region berada.
Nah, setelah kita tahu bahwa data dan aplikasi akan tinggal dan berjalan di suatu Region, Anda harus memilih Region mana yang tepat dan sesuai dengan kebutuhan. Ada 4 faktor bisnis yang menentukan pemilihan suatu Region.

1. **Compliance** (Kepatuhan)
   Sebelum faktor lainnya, Anda harus terlebih dahulu melihat *compliance requirement* (persyaratan kepatuhan) Anda. Titik. Opsi lainnya menjadi tidak penting saat Anda memiliki persyaratan kepatuhan.

   Misalnya, jika Anda memiliki persyaratan bahwa data Anda harus tinggal di perbatasan negara Inggris, maka pilihlah Region London. Atau jika Anda harus masuk ke dalam perbatasan negara Cina, maka Anda bisa memilih salah satu Region Cina.

   Namun, sebagian besar perusahaan tidak diatur oleh regulasi yang ketat semacam itu. Jadi, jika Anda tidak memiliki kontrol kepatuhan atau regulasi yang mewajibkan penentuan Region, maka Anda dapat mempertimbangkan faktor lain.

2. **Proximity** (Kedekatan)
   Memilih region yang paling dekat dengan basis pelanggan akan membantu Anda untuk mengirimkan konten lebih cepat kepada mereka .

   Katakanlah suatu perusahaan berlokasi di Washington, DC, namun kebanyakan pelanggannya tinggal di negara Singapura. Nah, untuk kasus ini, solusi yang tepat adalah dengan menjalankan infrastruktur di Region Northern Virginia agar dekat dengan lokasi perusahaan dan menerapkan aplikasi dari Region Singapura.

   Dengan begitu, **latensi** (waktu yang diperlukan untuk mengirim dan menerima data) bisa semakin kecil dan pengiriman konten ke pelanggan menjadi lebih cepat.

3. **Feature Availability** (Ketersediaan Fitur)
   Adakalanya, Region terdekat mungkin tidak memiliki semua fitur AWS yang Anda inginkan. Namun, AWS terus berinovasi untuk pelanggan. Setiap tahun, AWS merilis ribuan fitur dan produk baru secara spesifik untuk menjawab permintaan dan kebutuhan pelanggan.

Tapi, terkadang layanan baru tersebut membutuhkan banyak perangkat keras fisik baru yang harus AWS bangun agar dapat beroperasi. Dan terkadang, itu berarti AWS harus membangun layanan di satu Region pada satu waktu.

Misal Anda ingin membangun sebuah aplikasi yang menggunakan Amazon Braket--platform komputasi kuantum baru AWS. Faktanya, layanan ini belum tersedia di seluruh AWS Regions sehingga Anda harus menjalankannya di Region yang menyediakannya.

Kalau begitu, dapatkah kita mengharapkan fitur tersebut tersedia di semua Region? Ya, itu ekspektasi yang bagus. Tapi, jika Anda ingin menggunakannya hari ini juga, maka Anda harus mempertimbangkan untuk memilih Region lain yang memiliki fitur tersebut.

4. **Pricing** (Harga)
   Meskipun spesifikasi perangkat keras pada suatu Region setara dengan Region lain, namun beberapa lokasi bisa lebih mahal pengoperasiannya.

   Misal negara Brasil, struktur pajak di sana memiliki tatanan sedemikian rupa sehingga biaya AWS bisa jauh lebih mahal untuk mengoperasikan layanan yang sama persis dibandingkan dengan negara lain. Pengoperasian beban kerja yang sama persis di kota Sao Paulo mungkin bisa 50% lebih mahal daripada di kota Oregon di negara Amerika Serikat.

   Harga dapat ditentukan oleh banyak faktor dan AWS akan transparan mengenai hal tersebut.

   Ingat! Setiap Region memiliki harga yang berbeda-beda. Jadi, jika bujet adalah perhatian utama Anda, mungkin akan lebih baik untuk mengoperasikan lingkungan AWS Anda di negara lain meskipun basis pelanggan Anda tinggal di Brasil. Sekali lagi, ini berlaku jika anggaran adalah motivasi utama Anda.

Jadi, itulah 4 faktor utama untuk memilih suatu Region: Compliance, Proximity, Feature availability, dan Pricing. Selanjutnya, kita akan menelaah bagian yang lebih kompleks di dalam Region.

Availability Zone

Seperti yang telah kita singgung di materi sebelumnya, menjalankan aplikasi di satu bangunan data center saja bukanlah solusi yang baik karena gedung tersebut bisa mengalami kegagalan untuk sejumlah alasan yang tak dapat dihindari.

Itulah mengapa AWS Regions tidak hanya terdiri dari satu lokasi saja. AWS memiliki banyak data center di seluruh dunia dan setiap Region terdiri dari beberapa data center.

AWS memberi nama satu atau sekelompok data center tersebut dengan sebutan Availability Zone (AZ). Setiap Availability Zone adalah satu atau beberapa data center terpisah dengan daya, jaringan, dan konektivitasnya sendiri-sendiri.

Saat meluncurkan Amazon EC2 instance, sebenarnya Anda menjalankan mesin virtual pada perangkat keras fisik yang diinstal di Availability Zone.

Ketahuilah! Setiap AWS Regions terdiri dari beberapa Availability Zone yang terisolasi dan secara fisik terpisah di dalam Region geografis. AWS tidak membangun Availability Zone bersebelahan satu sama lain. Kenapa?

Karena, jika insiden dengan skala besar terjadi--misal bencana alam--Anda dapat kehilangan konektivitas ke semua sumber daya yang ada di dalam Availability Zone tersebut.

Sekarang begini, saat Anda meluncurkan 1 EC2 instance, itu berarti instance tersebut hanya berjalan di 1 gedung atau 1 Availability Zone saja. Nah, persoalannya, jika terjadi bencana berskala besar, maka aplikasi tersebut tentunya tak lagi dapat melayani bisnis Anda.

Solusi terbaik untuk masalah ini adalah dengan menjalankan beberapa EC2 instance--seperti yang kita pelajari dari contoh *scaling* sebelumnya. Tetapi, jangan menjalankannya di gedung yang sama.

AWS memberikan jarak Availability Zone sejauh puluhan mil untuk satu sama lainnya namun tetap mempertahankan *single-digit millisecond latency* (latensi milidetik satu digit).

Sebagai praktik terbaik, AWS selalu menyarankan untuk menjalankan setidaknya 2 Availability Zone dalam satu Region. Itu artinya, Anda perlu menerapkan infrastruktur Anda di 2 AZ yang berbeda.

Dengan demikian, sekarang jika bencana melanda, aplikasi Anda akan tetap beroperasi dengan baik karena momen tersebut hanya melumpuhkan sebagian dari

kapasitas Anda, tidak semua. Dan Anda dapat dengan cepat menambah kapasitas di Availability Zone yang tersisa sehingga memungkinkan bisnis Anda terus beroperasi tanpa interupsi.
Tapi Region bukan sekadar tempat untuk mengelola EC2, banyak layanan AWS yang berjalan di level Region. Itu artinya, layanan tersebut sudah beroperasi secara sinkron di beberapa AZ tanpa upaya tambahan.
Ambil contoh layanan Elastic Load Balancing yang telah kita bicarakan sebelumnya. Ia adalah *regional construct* (konstruksi regional) yang berjalan di semua Availability Zone. Ia dapat pula berkomunikasi dengan EC2 instance yang beroperasi di AZ tertentu.
Layanan regional menurut definisi sudah *highly available* (sangat tersedia) tanpa biaya tambahan. Jadi, saat merencanakan infrastruktur dengan *high availability* (ketersediaan tinggi), Anda bisa menggunakan layanan apa pun yang terdaftar sebagai *regional scoped service* (layanan dengan cakupan regional).
Oke, itulah tadi seputar infrastruktur global AWS. Sekarang, mari kita masuk ke pembahasan materi berikutnya.

Edge Locations

Diambil dari AWS News Blog: 98, 99, 100 CloudFront Points of Presence!
Salah satu hal hebat tentang infrastruktur global AWS adalah caranya didesain untuk membantu Anda melayani pelanggan dengan lebih baik. Ingat kembali materi kita sebelumnya yang berkenaan dengan memilih Region. Salah satu kriteria utamanya adalah proximity alias kedekatan dengan pelanggan.
Tapi, masih ada permasalahan lainnya. Coba pikirkan kembali tentang skenario kedai kopi kita. Anggaplah Anda ingin membangun kedai kopi di suatu kota yang memiliki basis pelanggan besar. Namun sayangnya, kota tersebut tidak berada dekat dengan salah satu AWS Regions. Apa yang harus Anda lakukan?
AWS menyediakan solusinya. Daripada harus membangun kedai kopi baru, Anda dapat membangun kedai satelit (kedai kopi kecil di bawah pelayanan pusat distribusi yang sama) untuk melayani pelanggan Anda di kota tersebut.
Mari kita lihat dari perspektif IT. Misalnya Anda memiliki pelanggan di kota Jakarta yang membutuhkan akses ke data aplikasi. Akan tetapi, data tersebut disimpan di Region Tokyo, tentu dengan jarak yang jauh ini akan menyebabkan latensi yang besar.

Lalu bagaimana solusinya? Begini, daripada semua pelanggan yang berada di kota Jakarta itu mengakses data ke Tokyo, akan lebih baik jika Anda cukup menempatkan salinan data secara lokal atau simpan salinannya ke dalam *cache* (penyimpanan data sementara agar dapat diakses lebih cepat) di Singapura.

Nah dengan begitu, saat pelanggan dari kota Jakarta mengakses salah satu data Anda, Amazon CloudFront--nanti kita bahas--akan mengambil data dari *cache* di Edge locations lalu mengirimkannya ke pelanggan tersebut. Data tadi terkirim lebih cepat karena berasal dari Edge locations Singapura--yang jaraknya lebih dekat dengan kota Jakarta--bukan dari Tokyo.

Teknik menyimpan salinan data di *cache* dengan lokasi yang lebih dekat dengan pelanggan di seluruh dunia adalah konsep yang digunakan oleh jaringan pengiriman konten alias *content delivery network* (CDN). Di AWS, CDN disebut dengan Amazon CloudFront.

Amazon CloudFront adalah layanan yang dapat membantu Anda untuk mengirimkan data, video, aplikasi, dan API ke pelanggan di seluruh dunia dengan latensi rendah dan kecepatan transfer yang tinggi. Amazon CloudFront menggunakan Edge locations di seluruh dunia untuk membantu mempercepat komunikasi dengan pengguna--di mana pun mereka berada.

Jadi, Edge locations adalah lokasi yang digunakan Amazon CloudFront untuk menyimpan salinan *cache* dengan jarak yang dekat dengan pelanggan sehingga konten dapat terkirim lebih cepat.

Ketahuilah! Edge locations itu terpisah dari AWS Regions. Sehingga, Anda dapat mengirim konten dari dalam Region ke kumpulan Edge locations di seluruh dunia.

AWS Edge locations tak hanya berguna untuk menjalankan CloudFront, melainkan juga layanan domain name system (DNS) atau sistem nama domain yang dikenal sebagai Amazon Route 53. Layanan ini dapat membantu mengarahkan pelanggan ke lokasi web yang tepat dengan latensi rendah yang andal.

Lalu, mungkin Anda akan bergumam, “Bisa *nggak* ya kalau kita ingin menggunakan layanan AWS langsung di dalam gedung milik sendiri?” Coba tebak.

Tentu bisa. AWS dapat melakukannya untuk Anda.

Perkenalkan, AWS Outposts. Dengan layanan ini, pada dasarnya, AWS akan menginstal Region mini yang beroperasi penuh, tepat di dalam data center Anda sendiri. Semua infrastruktur dan layanan tersebut dimiliki serta dioperasikan oleh AWS menggunakan 100% kegunaan AWS namun tetap terisolasi di dalam data center Anda.

AWS Outposts bukanlah solusi yang dibutuhkan bagi sebagian besar pelanggan AWS. Tetapi, jika Anda memiliki masalah tertentu yang hanya dapat diselesaikan dengan tetap berada di dalam data center Anda sendiri, AWS Outposts dapat menjadi jalan keluarnya.

Oke, sebenarnya masih banyak yang dapat kita paparkan tentang infrastruktur global AWS. Meskipun demikian, mari kita buat bahasan ini tetap sederhana dan berakhir di sini saja. Jadi, berikut adalah poin kuncinya:

1. Region adalah wilayah yang terisolasi secara geografis di mana Anda dapat mengakses layanan yang diperlukan untuk menjalankan segala macam kebutuhan.
2. Region terdiri dari Availability Zone yang memungkinkan Anda untuk menjalankan seluruh bangunan data center yang terpisah secara fisik dengan jarak puluhan mil sambil menjaga aplikasi Anda tetap bersatu secara logis. Availability Zone membantu Anda untuk dapat mencapai *high availability* (ketersediaan tinggi) dan *disaster recovery* (pemulihan bencana) tanpa upaya apa pun dari Anda.
3. AWS Edge locations digunakan untuk menjalankan Amazon CloudFront sehingga dapat memperdekat konten kepada pelanggan Anda di mana pun mereka berada.

Cara Menyediakan Sumber Daya AWS

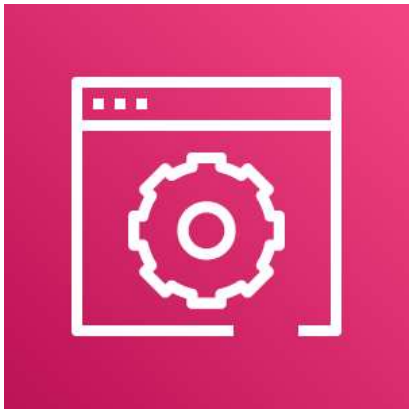
Sampai di sini, kita telah membicarakan tentang beberapa sumber daya serta infrastruktur global AWS. Melihat materi-materi sebelumnya, muncullah sebuah pertanyaan, bagaimana sebenarnya kita dapat berinteraksi dengan layanan-layanan tersebut?

Jawabannya adalah API--sudah kita singgung sedikit di modul sebelumnya. Di AWS semua aktivitas adalah panggilan API. API adalah application programming interface atau antarmuka pemrograman aplikasi. Dengan kata lain, ada cara yang telah ditentukan sebelumnya untuk Anda sehingga dapat berinteraksi dengan layanan AWS.

Anda dapat memanggil API ini untuk menyediakan, mengonfigurasi, dan mengelola sumber daya AWS. Misal untuk meluncurkan EC2 instance atau membuat AWS Lambda *function*. Masing-masing akan menjadi permintaan dan panggilan API yang berbeda ke AWS.

Jadi, untuk berinteraksi dengan layanan AWS, Anda dapat menggunakan AWS Management Console, AWS Command Line Interface (CLI), dan AWS Software Development Kit (SDK). Atau berbagai alat lain seperti AWS Elastic Beanstalk dan AWS CloudFormation (layanan yang dapat membuat permintaan untuk dikirim ke AWS API guna membuat dan mengelola sumber daya AWS).

AWS Management Console

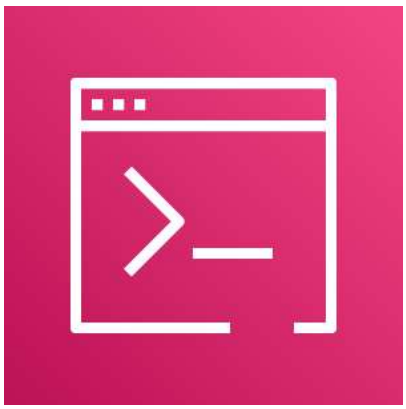


AWS Management Console adalah antarmuka berbasis browser yang dapat digunakan untuk mengakses dan mengelola layanan AWS. Melalui *console* (konsol), Anda dapat mengelola sumber daya AWS secara visual dan dengan cara yang mudah dipahami. Tentu ini adalah cara yang ideal untuk memulai dan membangun pengetahuan Anda tentang layanan AWS.
Dengan AWS Management Console, Anda dapat

- mencari layanan AWS dari nama, kata kunci, atau akronim;
- membangun lingkungan pengujian;
- melihat tagihan AWS;
- melakukan pemantauan; atau
- bekerja dengan sumber daya nonteknis lainnya.

AWS Console versi aplikasi seluler juga tersedia dan dapat Anda gunakan untuk melakukan tugas seperti memantau sumber daya, melihat alarm, dan mengakses informasi penagihan.
AWS Management Console adalah tempat pertama terbaik yang perlu Anda tuju ketika ingin mempelajari tentang AWS.

AWS Command Line Interface



Jika Anda akan menjalankan sumber daya di lingkungan produksi, tentu Anda tak ingin bergantung dengan cara *point and click* (tunjuk dan klik) yang diberikan console untuk membuat dan mengelola sumber daya AWS.
Ambil contoh pembuatan Amazon EC2 instance. Dengan AWS Management Console, Anda perlu klik berkali-kali, mengatur semua konfigurasi, barulah instance Anda dapat diluncurkan. Jika nantinya ingin meluncurkan EC2 instance lain, Anda harus kembali ke console dan melakukan proses klik yang sama terhadap layar-layar itu lagi. Ini akan membuang banyak waktu.
Jika seorang manusia terus-menerus melakukan penyediaan manual semacam ini, artinya Anda membuka peluang terhadap potensi kesalahan karena manusia sangat mudah untuk lupa mengeklik kotak centang atau salah mengeja sesuatu.
Jawaban untuk masalah ini adalah dengan menggunakan sarana yang memungkinkan Anda membuat skrip atau memprogram panggilan API, yaitu dengan AWS Command Line Interface atau CLI.

CLI memungkinkan Anda untuk mengendalikan layanan AWS dengan baris perintah melalui satu alat. Jelas ini berbeda dengan gaya navigasi visual dari AWS Management Console. AWS CLI tersedia untuk pengguna Windows, macOS, dan Linux.
Dengan menulis perintah menggunakan CLI, Anda dapat membuat tindakan yang dapat ditulis berkali-kali. Misal Anda menulis dan menjalankan perintah untuk meluncurkan EC2 instance. Nah, jika Anda ingin meluncurkan instance lain, cukup jalankan kembali perintah tersebut. Dengan demikian, Anda akan terhindar dari *human error* alias kesalahan manusia.
Selain itu, Anda juga dapat menjalankan skrip tersebut secara otomatis, contohnya dengan berdasarkan jadwal atau dipicu oleh proses lain.

AWS Software Development Kit

Cara lain untuk berkomunikasi dengan AWS adalah melalui AWS Software Development Kit atau SDK. SDK memudahkan Anda untuk berinteraksi dengan sumber daya AWS melalui berbagai bahasa pemrograman.
Hal ini memudahkan *developer* (pengembang) untuk membuat program di AWS *tanpa* menggunakan *low-level* API serta menghindari pembuatan sumber daya secara manual yang dari tadi kita bincangkan.
Sederhananya, *low-level* API memungkinkan Anda untuk memanipulasi fungsi di dalam API secara terperinci sesuai dengan kebutuhan. Lawannya adalah *high-level* API, yang memberikan lebih banyak fungsi dalam satu perintah dan lebih mudah digunakan sehingga Anda tak perlu mendalami detail teknis dan struktur API-nya.
Untuk membantu Anda memulai menggunakan SDK, AWS menyediakan dokumentasi dan sampel kode untuk setiap bahasa pemrograman yang didukung, yaitu mencakup C++, Go, Java, JavaScript, .NET, Node.js, PHP, Python, dan Ruby.

AWS Elastic Beanstalk
Opsi-opsi sebelumnya, yaitu AWS Management Console, CLI, dan SDK adalah cara penyediaan dan pengelolaan lingkungan AWS yang harus *Anda lakukan sendiri*.
Kita telah belajar, bahwa jika ingin menyediakan sumber daya, kita dapat melakukan salah satu cara di bawah ini:

- Masuk ke konsol, arahkan dan klik.
- Menulis beberapa perintah.

- Menulis beberapa program untuk melakukannya.

Nah, selain yang disebutkan di atas, ada juga cara lain untuk mengelola lingkungan AWS Anda menggunakan *managed service* (layanan terkelola) seperti AWS Elastic Beanstalk.

AWS Elastic Beanstalk adalah layanan yang dapat membantu Anda menyediakan lingkungan berbasis Amazon EC2. Tak perlu lagi harus klik sana sini di console atau menulis beberapa baris perintah untuk membangun jaringan, EC2 instance, *scaling* (penyesuaian kapasitas), dan Elastic Load Balancer.

Ucapkan selamat tinggal kepada semua itu. AWS Elastic Beanstalk dapat menyediakan dan mengelola semua infrastruktur tersebut sembari tetap memberi visibilitas dan kendali atas sumber daya yang mendasarinya. Dengan begitu, Anda bisa fokus pada aplikasi bisnis bukan infrastrukturnya.

Cukup unggah kode dan tentukan konfigurasi yang Anda inginkan, maka AWS Elastic Beanstalk pun akan mengolah informasi tersebut dan membangun lingkungan AWS-nya untuk Anda.

Lingkungan yang dimaksud adalah:

- Penyesuaian Kapasitas
- *Load balancing* (Penyeimbang beban)
- *Auto-scaling* (Penskalaan otomatis)
- Pemantauan kesehatan aplikasi

Simpan konfigurasi lingkungan tersebut sehingga nantinya akan memudahkan Anda saat menerapkannya kembali.

AWS CloudFormation

Layanan lain yang dapat Anda gunakan untuk membantu membuat penerapan otomatis dan berulang adalah AWS CloudFormation.

AWS CloudFormation adalah layanan *infrastructure as code* (infrastruktur sebagai kode) yang memungkinkan Anda untuk menentukan berbagai sumber daya AWS dengan cara deklaratif menggunakan dokumen berbasis teks JSON atau YAML yang disebut sebagai CloudFormation template.

Format deklaratif memudahkan Anda untuk memberikan spesifikasi apa yang ingin dibangun tanpa harus mendeskripsikan detail bagaimana caranya karena mesin CloudFormation yang akan mengurusnya. Anda hanya perlu menulis baris kode untuk membangun lingkungan ketimbang menggunakan AWS Management Console.

AWS CloudFormation tak hanya terbatas pada solusi berbasis EC2, melainkan mendukung banyak sumber daya AWS, seperti penyimpanan, database, analitik, *machine learning*, dan banyak lagi.

Oke, setelah Anda menentukan sumber daya dalam CloudFormation template, AWS CloudFormation akan menguraikannya dan mulai menyediakan semua sumber daya tersebut secara paralel.

AWS CloudFormation mengelola semua panggilan API. Anda dapat menjalankan CloudFormation template yang sama di beberapa akun atau Region. Dengan begitu,

layanan ini akan membuat lingkungan yang identik di dalamnya. Tak akan ada lagi *human error* karena semua proses berjalan secara otomatis dengan sepenuhnya.

Ikhtisar

Luar biasa! Kita sudah berbincang banyak hal di modul ini. Kita telah:

- Membahas bagaimana klaster logis dari data center membentuk Availability Zone. Lalu, Availability Zone membentuk Region dan tersebar secara global.
- Menelaah cara memilih Region dan Availability Zone yang ingin dioperasikan. Sebagai praktik terbaik, Anda harus selalu menerapkan infrastruktur minimal di 2 Availability Zone.
- Menilik beberapa layanan AWS seperti Elastic Load Balancing, Amazon SQS, dan Amazon SNS.
- Membicarakan tentang Edge locations dan bagaimana Anda dapat menyebarkan konten untuk mempercepat pengiriman ke pelanggan.
- Membahas perangkat edge seperti AWS Outposts yang memungkinkan Anda menjalankan infrastruktur AWS langsung di data center Anda sendiri.
- Mendiskusikan bagaimana cara menyediakan sumber daya AWS melalui berbagai opsi: AWS Management Console, SDK, CLI, AWS Elastic Beanstalk, dan AWS CloudFormation--di mana Anda mempelajari cara menyiapkan *infrastructure as code* (infrastruktur sebagai kode).

Dengan mempelajari modul ini, semoga Anda Anda dapat memahami bahwa AWS tersedia secara global dan betapa mudahnya memulai dengan penyediaan sumber daya.

Materi Pendukung

Tinjau tautan berikut untuk mempelajari lebih lanjut tentang konsep yang telah kita jelajahi di modul ini:

- Global Infrastructure
- Interactive map of the AWS Global Infrastructure
- Regions and Availability Zones
- AWS Networking and Content Delivery Blog
- Tools to Build on AWS
- AWS Customer Stories: Content Delivery

MODUL 3

Pengenalan ke Jaringan

Pikirkan kembali skenario kedai kopi atau lingkungan AWS kita. Proses pemesanan pada kedai kopi adalah pelanggan menyampaikan ordernya kepada kasir yang lantas meneruskannya ke barista. Proses ini seharusnya sudah berjalan lancar tanpa kendala ya.

Meskipun demikian, bagaimana jika ada beberapa pelanggan tak sabar yang coba mengabaikan kasir dan ingin memberikan pesanannya langsung ke barista? Para pelanggan penerobos antrean ini tentu akan merusak alur kedai kopi.

Yah, tak mungkin kita biarkan pelanggan terus berinteraksi dengan barista. Alih-alih menerima pesanan, sang barista harus fokus membuat kopi saja. Jadi, apa yang harus kita perbuat?

Di AWS Anda bisa menggunakan Amazon Virtual Private Cloud (Amazon VPC) untuk menuntaskan dilema ini. VPC memungkinkan Anda untuk menyediakan bagian yang terisolasi secara logis dari AWS Cloud di mana Anda dapat meluncurkan sumber daya AWS di jaringan virtual sesuai kebutuhan.

Sumber daya tersebut dapat menjadi *public-facing* yang berarti memiliki akses ke internet ataupun *private* alias tanpa akses internet. Jenis kedua biasanya dipakai untuk layanan *backend* seperti database atau server aplikasi.

Nah, pengelompokan sumber daya menjadi publik dan privat ini disebut dengan *subnet* yang juga merupakan rentang alamat IP di VPC Anda.

Sekarang mari kembali ke kedai kopi dan mengimplementasikan apa yang telah kita pelajari di atas. Anda dapat menempatkan kasir dan barista di area kerja yang terpisah.

Karena kasir bertanggung jawab untuk menerima pesanan pelanggan, maka kita tempatkan ia di *subnet publik*. Sehingga, kasir dapat berkomunikasi langsung dengan pelanggan atau internet--jika kasusnya adalah *instance*.

Berbeda kasusnya buat barista. Karena kita ingin ia fokus untuk membuat kopi dan tidak berinteraksi dengan pelanggan secara langsung, maka kita tempatkan ia di *subnet privat*. Dengan begitu, barista tetap dapat menerima pesanan dari kasir tetapi tidak langsung dari pelanggan.

Semoga modul pengantar ini dapat membuka gerbang pemahaman dasar kita menuju materi yang lebih detail di modul mendatang. Mari kita masuki gerbang tersebut sekarang! *Are you ready?*

Konektivitas ke AWS

Ada jutaan pelanggan yang menggunakan layanan AWS. Lalu, bayangkan terdapat pula jutaan sumber daya yang telah dibuat oleh pelanggan tersebut, seperti Amazon EC2 instance. Tanpa adanya batasan di sekitar semua sumber daya itu, *traffic* jaringan akan mengalir di antaranya tanpa restriksi.

Layanan jaringan yang dapat Anda gunakan untuk menetapkan batasan di sekitar sumber daya AWS adalah Amazon Virtual Private Cloud (Amazon VPC).

Amazon Virtual Private Cloud (Amazon VPC)

Selamat datang di Amazon Virtual Private Cloud (Amazon VPC). Anggaplah ia sebagai benteng kukuh di mana tak akan ada yang dapat masuk atau keluar tanpa izin secara eksplisit.

Amazon VPC pada dasarnya adalah jaringan pribadi Anda di AWS. VPC memungkinkan Anda untuk membuat bagian terisolasi dari AWS Cloud dan meluncurkan sumber daya seperti EC2 instance dan ELB di dalamnya.

Anda tak bisa menaruh sumber daya ke VPC begitu saja, melainkan harus mengelolanya di dalam subnet yang berbeda. Subnet adalah bagian dari VPC yang dapat mengelompokkan sumber daya. Subnet bersama dengan aturan jaringan--akan kita bahas nanti--dapat mengontrol apakah sumber daya tersedia untuk publik atau privat.

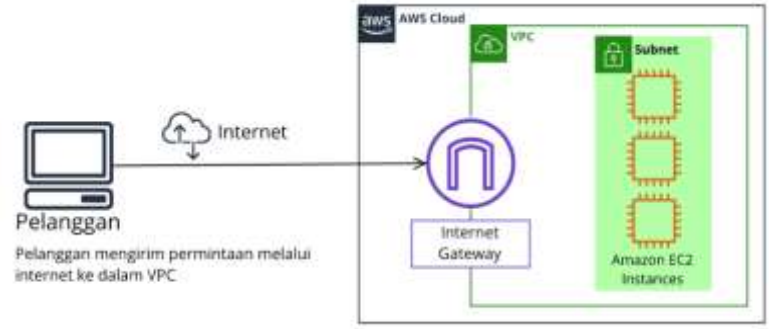
Anda bisa saja memiliki sumber daya yang *internet-facing* (terhubung ke internet) sehingga dapat dijangkau oleh umum, seperti website publik.

Namun, dalam skenario lain, Anda mungkin ingin memiliki sumber daya yang hanya Anda saja yang dapat menjangkaunya. Ini mungkin pas untuk layanan internal, seperti aplikasi HRD atau database.

Internet Gateway

Berkaca dari materi sebelumnya. Mari kita telaah tentang sumber daya yang *internet-facing* atau *public-facing* (berhubungan dengan internet/publik).

Untuk mengizinkan *traffic* dari internet publik mengalir masuk dan keluar dari VPC, Anda harus melampirkan apa yang disebut dengan Internet Gateway (IGW). Di bawah ini adalah contoh arsitektur yang menggunakan Internet Gateway.



Jika Anda tak kunjung paham apa fungsi dari Internet Gateway, bayangkanlah ia seperti pintu depan yang terbuka untuk publik. Di kedai kopi, pintu depan berguna supaya orang-orang dapat keluar masuk dengan leluasa. Jika tak ada, bagaimana pelanggan ingin memesan kopi?

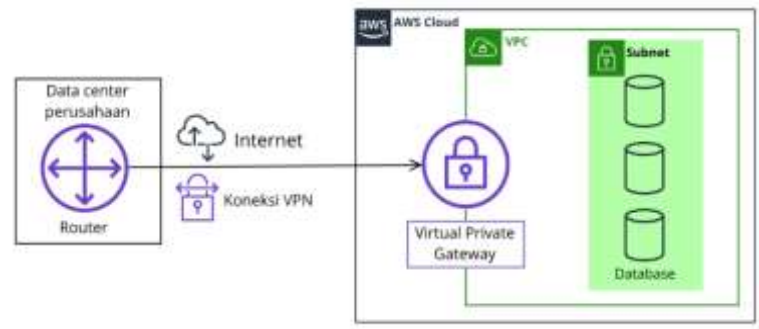
Pintu depan ini adalah perumpamaan yang ideal untuk Internet Gateway. Tanpanya, tidak ada yang dapat menjangkau sumber daya di dalam VPC Anda.

Selanjutnya, timbul pertanyaan seperti ini, “Bagaimana jika kita memiliki sumber daya pribadi di VPC dan tidak ingin sembarang orang bisa menjangkaunya?” Mari kita lihat di materi berikutnya!

Virtual Private Gateway

Anda bisa memasang gateway privat yang hanya mengizinkan masuk suatu permintaan jika ia berasal dari jaringan yang disetujui, bukan internet publik. Gateway privat ini disebut juga dengan *virtual private gateway*. Ia memudahkan Anda untuk membuat koneksi VPN (virtual private network) terenkripsi antara jaringan privat--seperti data center on-premise atau jaringan perusahaan internal--ke VPC Anda.

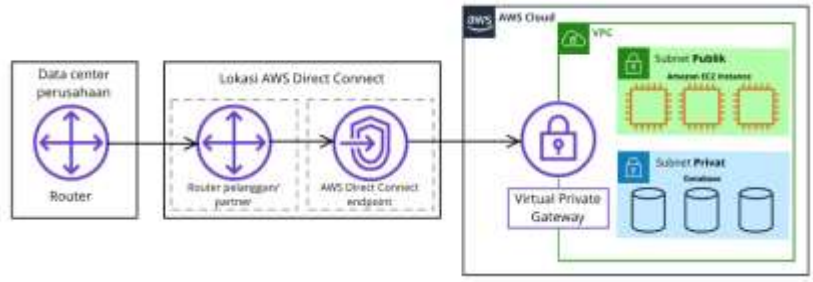
Jadi, dapat disederhanakan bahwa *virtual private gateway* adalah komponen yang memungkinkan traffic internet yang terlindungi masuk ke dalam VPC. Silakan amati contoh arsitektur berikut:



Bingung? Nah, untuk mempermudah penjelasan bagaimana virtual private gateway bekerja, mari kita cari tahu dengan mengaitkannya ke skenario kedai kopi. Anda bisa mengibaratkan internet itu sebagai jalan raya antara rumah Anda dan kedai kopi.
Katakanlah Anda bepergian melalui jalan tersebut dengan seorang pengawal untuk melindungi Anda. Tentu, sebenarnya Anda masih menggunakan jalan yang sama dengan pelanggan lain, bedanya, Anda memiliki lapisan perlindungan ekstra.
Nah, pengawal tersebut bisa Anda anggap sebagai koneksi VPN yang mengenkripsi (atau melindungi) traffic internet dari semua permintaan lain di sekitarnya.
Oh, tidak! Sayangnya, sekarang timbul masalah baru terhadap kasus di atas, meskipun Anda memiliki perlindungan yang ekstra. Apa itu? Anda masih menggunakan jalanan yang sama dengan pelanggan lain. Walhasil, sudah barang tentu ia akan rentan terhadap kemacetan atau perlambatan lalu lintas.
Hal yang sama pun berlaku untuk koneksi VPN. Memang betul koneksi VPN bersifat pribadi dan dienkripsi, tetapi faktanya ia masih menggunakan koneksi internet reguler dengan *bandwidth* (jumlah maksimum data yang dapat dikirim) yang terbagi kepada banyak pengguna internet lainnya.
Lalu, bagaimana jika kita ingin memiliki koneksi pribadi yang mengarah langsung ke VPC?

AWS Direct Connect

Untuk memulai, bayangkanlah sebuah apartemen dengan lorong pribadi yang langsung terhubung ke kedai kopi. Hanya penghuni apartemen saja yang dapat melewati lorong ini.
Lorong ini menyediakan jenis koneksi khusus/terdedikasi di mana penghuni apartemen dapat masuk ke kedai kopi tanpa perlu menggunakan jalan raya bersama para pelanggan lain.
Intinya, jika Anda menginginkan koneksi privat, koneksi terdedikasi, jumlah latensi yang rendah, dan tingkat keamanan yang tinggi, maka Anda bisa mewujudkannya di AWS dengan menggunakan AWS Direct Connect.



AWS Direct Connect memungkinkan Anda untuk membuat koneksi fiber yang privat nan terdedikasi sepenuhnya antara data center Anda dan VPC. Untuk membangun koneksi tersebut, Anda perlu berpartner dengan mitra Direct Connect yang tersedia di wilayah Anda.
Dengan demikian, layanan ini dapat membantu Anda memenuhi kebutuhan regulasi dan kepatuhan yang tinggi serta menghindari potensi masalah pada *bandwidth*.
Catatan: Satu VPC mungkin memiliki beberapa tipe gateway yang terpasang untuk berbagai jenis sumber daya di VPC yang sama namun dengan subnet yang berbeda.

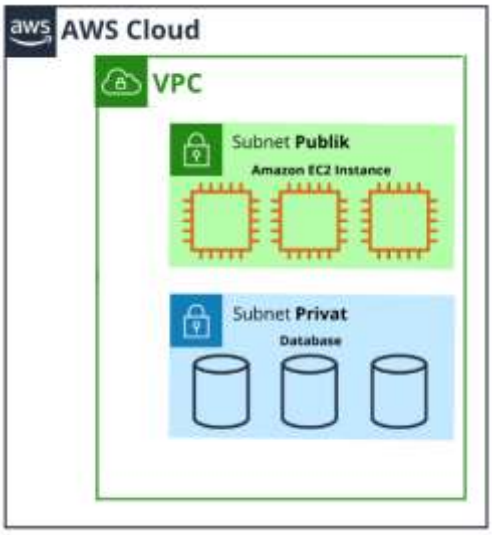
Subnet dan Network Access Control List

Pada materi pengantar sebenarnya kita telah mengupas sedikit tentang peran dari subnet. Bahkan kita juga telah mengilustrasikannya di kedai kopi. Tapi rasanya masih belum dalam ya. Begitu juga dengan network access control list, sepertinya kita belum menyinggungnya sama sekali. Tak apa. Di pembahasan kali ini, kita akan belajar lebih lanjut.

Subnet dan network access control list adalah dua hal yang penting untuk dipahami saat Anda belajar jaringan AWS. Jadi, tak perlu menunggu waktu lama. Mari kita lanjutkan ke materi berikutnya!

Subnet

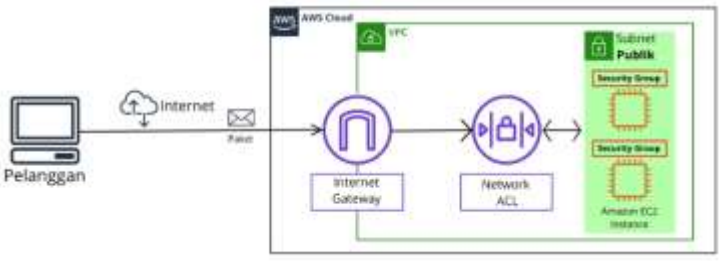
Subnet adalah sebuah bagian dari VPC di mana Anda dapat mengelompokkan sumber daya berdasarkan keamanan atau kebutuhan operasional. Subnet bisa menjadi publik maupun privat.



Subnet publik biasanya berisi sumber daya yang perlu diakses oleh publik, seperti website toko online. Sedangkan subnet privat memuat sumber daya yang seharusnya hanya dapat diakses melalui jaringan privat, seperti database yang berisi informasi pribadi pelanggan dan riwayat pesanan.

Di VPC subnet dapat berkomunikasi satu sama lain. Misalnya, Anda dapat memiliki aplikasi pada Amazon EC2 instance di subnet publik yang berkomunikasi dengan database di subnet pribadi.

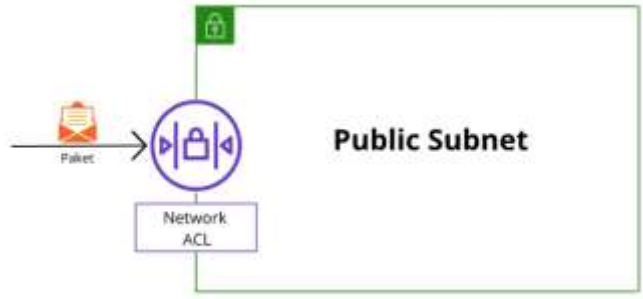
Network Access Control List (Network ACL)



Sebelumnya, kita telah belajar seputar Internet Gateway (IGW) yang dapat mengizinkan traffic masuk atau keluar dari VPC. Tetapi, layanan ini hanya meliputi satu bagian saja dari keamanan jaringan--yang harus Anda fokuskan sebagai bagian dari strategi IT.
Ketahuilah, AWS memiliki berbagai layanan yang dapat mencakup setiap lapisan keamanan:

- *Network hardening* (Penguatan jaringan).
- Keamanan aplikasi.
- Identitas pengguna.
- Autentikasi dan otorisasi.
- Pencegahan *distributed denial-of-service* (DDoS).
- Integritas data.
- Enkripsi.
- dan masih banyak lainnya.

Tenang, Anda tak harus memahami semuanya. Di kelas ini kita hanya akan membahas beberapa bagian saja.
Sekarang mari kita berbincang mengenai beberapa aspek dari network hardening (penguatan jaringan) dengan melihat praktiknya di dalam VPC.
Satu-satunya alasan teknis untuk menggunakan subnet di VPC adalah untuk mengontrol akses ke gateway. Subnet publik memiliki akses ke Internet Gateway, sementara Subnet privat tidak. Tapi walaupun begitu, tahukah Anda? Subnet juga bisa mengontrol perizinan traffic, *loh*.
Bagaimana caranya? Simak paparan berikut.
Ketika pelanggan meminta data dari aplikasi yang berjalan di AWS Cloud, maka permintaan ini dikirim sebagai paket. Paket adalah sebuah unit data yang dikirim melalui internet atau jaringan.



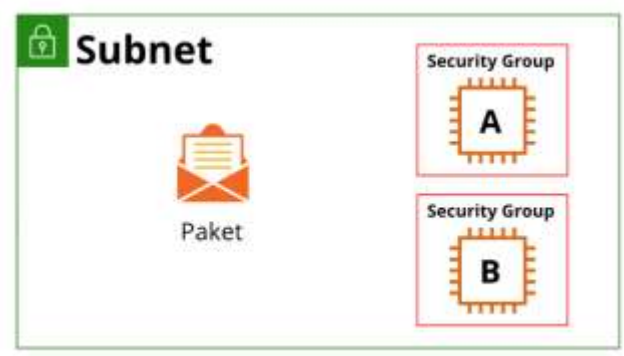
Paket masuk ke VPC melalui Internet Gateway. Sebelum paket dapat masuk atau keluar dari subnet, ia akan diperiksa terkait perizinannya. Pemeriksaan ini dilakukan untuk melihat apakah paket memiliki izin untuk masuk ke subnet berdasarkan *siapa* pengirimnya dan *bagaimana* ia mencoba berkomunikasi dengan sumber daya yang berada di subnet.
Komponen VPC yang memeriksa izin paket untuk subnet adalah network access control list alias network ACL. Network ACL adalah firewall virtual yang mengontrol traffic masuk dan keluar di tingkat subnet. Tentu ini berbeda dengan Internet Gateway yang cakupannya di tingkat VPC.

Jika paket memiliki potensi yang dapat membahayakan sumber daya di dalam subnet--seperti upaya untuk menguasai sistem melalui permintaan administratif--maka ia akan diblokir sebelum dapat menyentuh target.
Jika masih sukar memahaminya, Anda bisa menganggap network ACL sebagai petugas pengawas paspor. Misalnya begini.
Mari kita keluar dari kedai kopi dan bayangkan Anda sedang berada di bandara. Di sana ada banyak turis yang mencoba masuk ke negara lain. Anda dapat menganggap para turis itu sebagai paket dan petugas pengawas paspor sebagai network ACL.
Petugas pengawas paspor memeriksa kredensial setiap turis yang masuk ke suatu negara. Jika nama turis tertera di dalam daftar yang disetujui, maka ia diizinkan untuk masuk. Sebaliknya, jika namanya tak terdaftar atau bahkan secara eksplisit tercatat di dalam daftar turis yang diblokir, maka tentu ia dilarang masuk.
Selain memeriksa traffic yang masuk, network ACL pun akan mengecek setiap traffic yang keluar dari subnet. Ini serupa pula dengan petugas pengawas paspor. Hanya karena Anda diizinkan masuk, bukan berarti petugas akan membiarkan Anda keluar dengan leluasa.
Setiap akun AWS menyertakan network ACL secara default (bawaan). Saat mengonfigurasi VPC, Anda dapat menggunakan default network ACL (mengizinkan semua traffic masuk dan keluar) atau custom network ACL (menolak semua traffic masuk dan keluar hingga Anda secara eksplisit mengizinkannya).
Selain itu, network ACL memiliki aturan penolakan secara eksplisit. Aturan ini berguna untuk memastikan jika sebuah paket tidak cocok dengan salah satu aturan lain di daftar, paket tersebut akan ditolak.
Mungkin terdengar seperti keamanan yang hebat ya? Tapi nyatanya, network ACL tidak bisa menjawab semua masalah terkait kontrol jaringan. Sebab, ia hanya dapat mengevaluasi paket jika melintasi batas subnet--baik masuk atau keluar namun tidak tahu-menahu apakah paket tersebut dapat mencapai EC2 instance tertentu atau tidak.

Boleh jadi Anda memiliki beberapa EC2 instance di subnet yang sama. Namun pada praktiknya, mungkin tiap-tiapnya akan memiliki aturan yang berbeda tentang

- *siapa* yang dapat mengiriminya pesan; atau
- port mana yang diizinkan untuk menerima pesan.

Jadi, Anda juga memerlukan keamanan jaringan pada tingkat instance. Nah, untuk menjawabnya, AWS memperkenalkan *security group*.
Security group adalah firewall virtual yang mengontrol traffic masuk dan keluar untuk Amazon EC2 instance. Terlihat berbeda ya dengan network ACL yang cakupannya di tingkat subnet.



Saat EC2 instance diluncurkan, ia secara otomatis dilengkapi dengan security group. Jika Anda memiliki beberapa Amazon EC2 instance di dalam subnet yang sama, Anda dapat mengaitkannya dengan security group yang sama maupun berbeda untuk setiap instance.

Ingat! secara default (bawaan), security group menolak semua traffic masuk namun mengizinkan semua lalu lintas yang keluar dari instance.

Dengan security group default, semua port dan alamat IP yang mengirimkan paket akan diblokir. Tentu ini sangat aman, tapi mungkin membuat instance tidak berguna. Maka dari itu, tentu Anda bisa mengonfigurasinya dengan menambah aturan sendiri yang mengizinkan atau menolak traffic sesuai kebutuhan.

Misalnya, dalam kasus website, Anda bisa mengatur security group untuk menerima traffic berbasis web (HTTPS) dan tidak untuk jenis lalu lintas lain (sistem operasi atau permintaan administrasi).

Jika sebelumnya kita mengibaratkan network ACL sebagai petugas pengawas paspor, nah, anggaplah security group itu seperti penjaga pintu di gedung apartemen Anda. Dalam hal ini, gedung tersebut adalah EC2 instance.

Penjaga pintu akan memeriksa setiap orang yang ingin memasuki gedung untuk memastikan apakah mereka memiliki izin atau tidak. Namun, bagi setiap orang yang akan keluar dari gedung tersebut tak akan diperiksa olehnya.

Serupa dengan itu, *security group* mengizinkan traffic tertentu untuk masuk dan--secara default--membolehkan semua lalu lintas keluar.

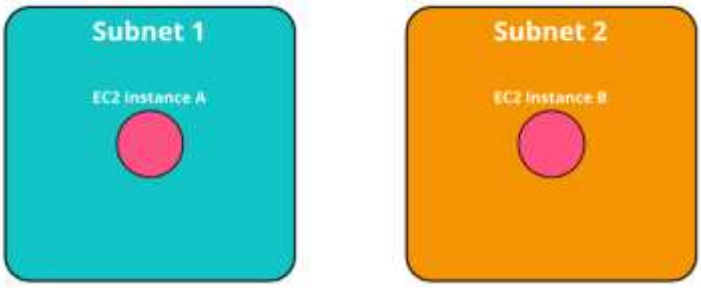
Mungkin dahi Anda akan mulai berkerut, “Tunggu sebentar. Kita baru saja belajar dua mesin berbeda namun melakukan pekerjaan yang sama, yaitu membiarkan paket dari alamat yang kita izinkan untuk masuk dan menolak paket dari alamat yang tidak kita izinkan. Lalu, apa bedanya?”

Oke, Oke. Tenang! Mari kita tilik perbedaannya.

Perbedaan utama antara security group dan network ACL adalah:

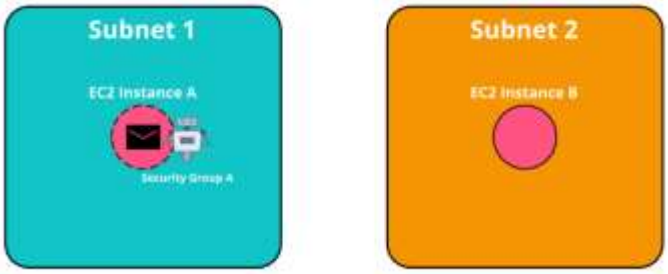
- Security group bersifat *stateful*, yang berarti ia memiliki semacam memori untuk mengingat siapa yang diizinkan masuk atau keluar.
- Network ACL bersifat *stateless*, artinya ia tidak mengingat apa pun. Layanan ini akan memeriksa setiap paket yang melintasi perbatasannya terlepas dari keadaan apa pun.

Oke, sekarang kita akan mengilustrasikan suatu perjalanan bolak-balik sebuah paket saat berpindah dari satu instance ke instance lain di subnet yang berbeda. Pahami metafora ini dengan baik ya.



Untuk mengawali, ketahuilah bahwa layanan manajemen traffic ini tak peduli dengan isi dari paket itu sendiri. Ia hanya memeriksa apakah pengirimnya tertera di dalam daftar yang disetujui atau tidak.
Baiklah. Mari kita mulai ilustrasinya. Katakanlah Anda ingin mengirim paket dari instance A ke instance B di subnet yang berbeda di VPC yang sama.

1. Paket dikirim dari instance A. Hal pertama yang akan terjadi adalah paket tersebut akan bertemu dengan batas security group dari instance A. Ingat! Secara default, security group akan mengizinkan semua traffic keluar. Jadi, paket bisa melanjutkan perjalanannya dan sukses melewati security group dari instance A.



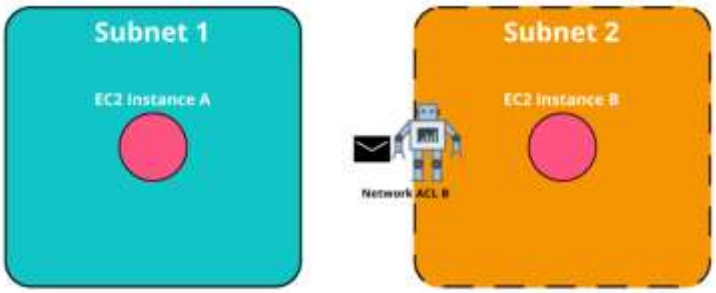
2. Selanjutnya, paket berhadapan dengan perbatasan subnet 1. Di sana ada petugas pengawas paspor, yakni network ACL. Network ACL tetap akan memeriksa paket walaupun security group telah mengizinkannya karena ia memiliki daftarnya sendiri atas siapa yang bisa dan tidak bisa lewat. Jika diperbolehkan, paket dapat melanjutkan perjalanannya.

3. Oke, sekarang paket telah keluar dari subnet asal dan menuju ke subnet target di mana instance B berada. Untuk memasuki subnet 2, paket kembali

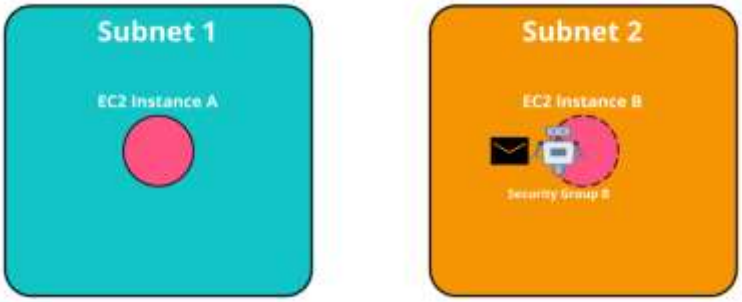
berhadapan dengan petugas pengawas paspor, yaitu Network ACL.

Hanya karena paket diizinkan keluar dari wilayah asalnya, bukan berarti ia dapat bebas masuk ke wilayah atau subnet target. Masing-masing subnet memiliki petugas pengawas paspornya sendiri. Walhasil, paket harus mendapatkan izin dari keduanya, jika tidak maka paket bisa ditolak di perbatasan.

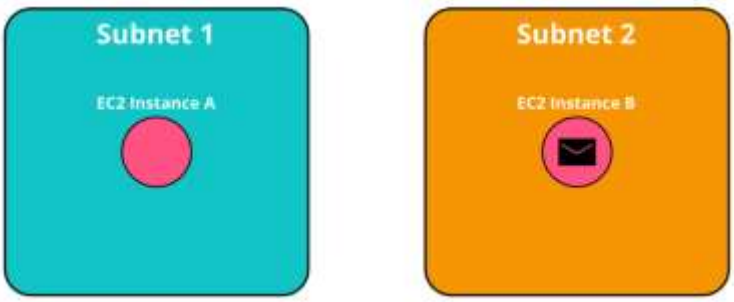
Nah, ternyata paket Anda tertera di dalam daftar yang disetujui untuk masuk ke subnet 2. Ayo, hampir sampai.



4. Sekarang paket semakin dekat dengan instance target, yakni instance B. Setiap EC2 instance memiliki security group-nya sendiri. Jika paket ingin masuk ke instance B, maka penjaga pintu alias security group perlu melakukan pemeriksaan terlebih dahulu untuk memastikan apakah paket diizinkan masuk atau tidak.



5. Apabila terdaftar, maka paket Anda pun dapat masuk dan akhirnya sampai ke instance target.

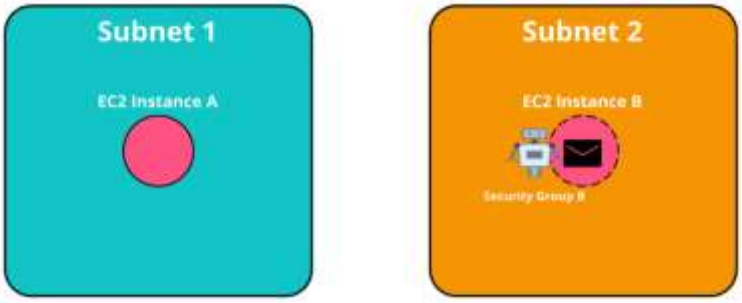


Wah cukup meletihkan ya. Setelah paket sampai ke tujuan, ia akan melakukan apa yang harus dilakukan. Nah, setelah proses transaksi selesai, sekarang saatnya

pulang. Kita akan melihat bagaimana *return traffic pattern* alias pola lalu lintas kembali terjadi.

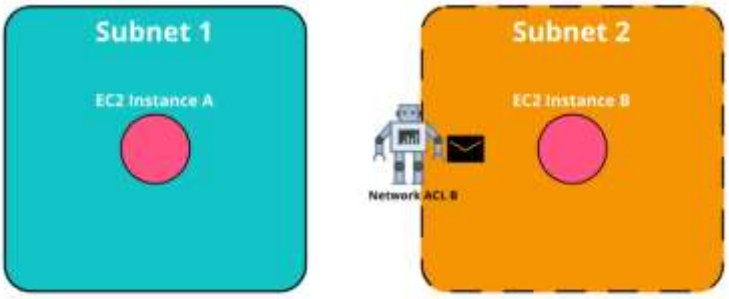
Ini adalah bagian yang paling menarik karena di sinilah sifat *stateful* versus *stateless* dari mesin yang berbeda berperan. Pasalnya, paket masih harus dievaluasi pada setiap pos pemeriksaan. Agar tak penasaran, silakan simak uraian perjalanan pulang dari paket berikut:

1. Seperti yang telah kita pelajari, security group secara default mengizinkan semua lalu lintas keluar. Jadi, ia tak perlu lagi memeriksa apakah paket diizinkan keluar atau tidak. Tanpa kendala paket pun berhasil meninggalkan instance B dan menuju perbatasan subnet 2.



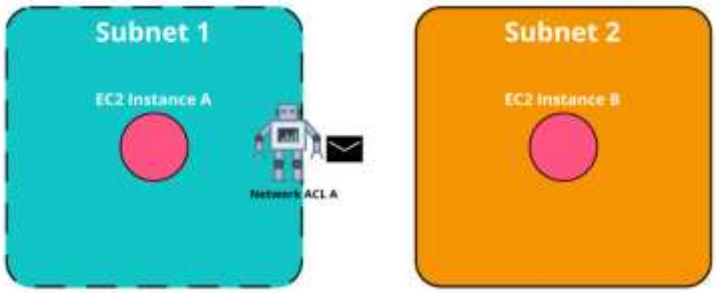
2. Nah, di perbatasan subnet berdirilah seorang petugas pengawas paspor. Tentu Anda sudah tak asing ya, dialah network ACL yang *stateless* alias tidak dapat mengingat status. Ia tidak peduli bahwasanya paket Anda telah melewatinya. Ini karena mungkin saja paket Anda tercantum di dalam daftar tidak-bisa-keluar.

   Setiap jalan masuk maupun keluar tetap diperiksa sesuai dengan daftar yang ada. Alamat pengembalian paket harus tercatat di dalam daftar yang disetujui agar berhasil melintasi perbatasan. Tenang, paket Anda diperbolehkan keluar.

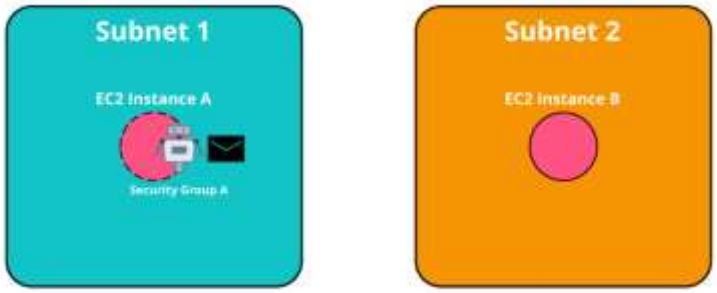


3. Oke, setelah keluar dari perbatasan subnet 2, paket pun tiba di perbatasan subnet asal, yakni subnet 1. Akan tetapi, paket harus berhadapan lagi dengan network ACL. Karena network ACL bersifat *stateless*, maka ia akan selalu memeriksa daftarnya. Untungnya, paket Anda diberikan izin untuk masuk ke

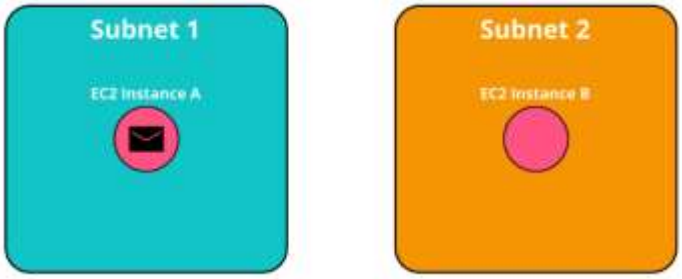
subnet asal.



4. Satu langkah lagi untuk kembali ke rumah. Tapi, security group--sang penjaga pintu--masih bertanggung jawab dan berdiri di sana. Nah, perbedaan utamanya terlihat di sini. Karena security group bersifat stateful, ia dapat mengenali sumber paket. Jadi, paket Anda tak akan diperiksa lagi saat hendak memasuki instance A.



5. *Welcome home!*



Huh! Mungkin tampak sedikit melelahkan ya hanya untuk mendapatkan paket dari satu instance ke instance lainnya dan kembali lagi. Jangan khawatirkan semua proses panjangnya. Faktanya, operasi pertukaran tersebut terjadi secara instan sebagai bagian dari cara kerja AWS Networking.

Manfaatkanlah network ACL dan security group ini guna mencapai keamanan jaringan yang komprehensif. Mengapa? Karena keamanan yang mendalam merupakan hal yang sangat penting untuk sebuah arsitektur modern.

Jaringan Global

Sebelumnya kita telah banyak belajar mengenai bagaimana *Anda* berinteraksi dengan infrastruktur AWS. Sekarang pertanyaannya adalah bagaimana *pelanggan Anda* berinteraksi dengan infrastruktur AWS tersebut?

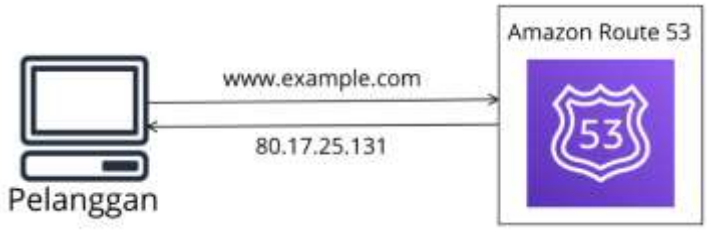
Tak perlu tergesa-gesa! Di modul ini akan dijelaskan tentangnya. Kita mulai dari DNS terlebih dahulu. Apa itu? Yuk kita lihat di materi berikutnya!

Domain Name System (DNS)

Untuk mengawali, yuk kita buat suatu cerita. Misalnya, Anda memiliki website yang berjalan di AWS Cloud. Silakan jawab, bagaimana pelanggan Anda dapat mengaksesnya? Mudah saja. tentu kita semua sudah tahu. Caranya adalah dengan memasukkan nama website Anda di browser, tekan Enter, dan *boom*! Website pun termuat.

Proses tersebut tentu saja tidak sekonyong-konyong terjadi begitu saja. Ada peran domain name system (DNS) di sana. Di AWS tersedia layanan DNS yang dapat Anda gunakan, yakni Amazon Route 53--akan dijelaskan nanti.

Tapi, tunggu! Sebelum melangkah lebih jauh, apa *sih* yang dimaksud dengan DNS? Begini, anggaplah DNS itu sebagai buku telepon bagi internet. DNS dapat menerjemahkan sebuah nama domain ke dalam alamat IP (Internet Protocol) yang dapat dibaca komputer.



Mari kita ambil contoh bagaimana DNS bekerja. Katakanlah Anda ingin membuka halaman www.example.com.

1. Masukkan nama domain tersebut ke browser Anda.
2. Lalu, permintaan tersebut akan dikirimkan ke Amazon Route 53 guna memperoleh alamat IP yang sesuai dengan website tersebut.
3. Route 53 merespons dengan memberikan alamat IP. Misalkan 80.17.25.131.
4. Kemudian, komputer atau browser Anda pun akan dirutekan ke alamat tadi.
5. Tada! Website pun termuat.

Tidak sulit kan untuk memahami DNS dan cara kerjanya? Nah, seperti yang telah dijanjikan di awal, sekarang mari kita tilik layanan Amazon Route 53 yang tercantum pada materi berikutnya.

Amazon Route 53

Amazon Route 53 adalah layanan domain name system (DNS) atau sistem nama domain di AWS yang *highly available* (sangat tersedia) dan *scalable* (dapat diskalakan). Layanan ini dapat memberikan Anda cara yang andal untuk merutekan pelanggan ke aplikasi internet yang berjalan di AWS.

Amazon Route 53 bertugas untuk menghubungkan permintaan pelanggan ke infrastruktur yang berjalan di AWS (seperti Amazon EC2 instance dan load balancers).

Bahkan, ia bisa pula mengarahkan pelanggan ke infrastruktur yang berada di luar AWS.
Jika kita melangkah lebih jauh, Amazon Route 53 itu dapat pula mengarahkan traffic ke *endpoints* (titik akhir) yang berbeda menggunakan beberapa *routing policies* (kebijakan perutean) yang berbeda, seperti:

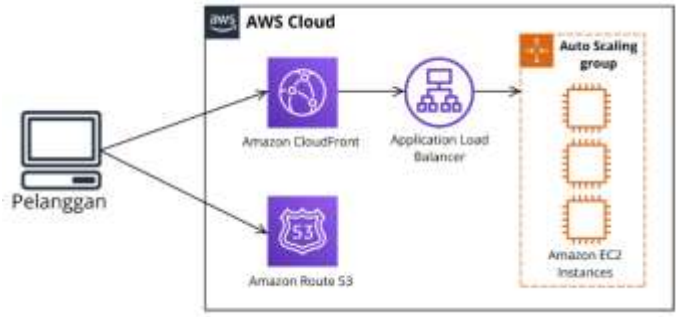
- Latency-based routing (Perutean berbasis latensi)
- Geolocation DNS
- Geoproximity routing
- Weighted round robin

Kita tidak akan memaparkan semuanya di sini. Tapi, mari ambil contoh dari salah satunya, yaitu Geolocation DNS. Dengan opsi tersebut, kita mengarahkan traffic berdasarkan lokasi pelanggan. Contohnya, lalu lintas yang datang dari Indonesia akan dialihkan ke Region Singapura atau jika berasal dari Jepang akan dialihkan ke Region Tokyo.
Selain mengarahkan traffic, Route 53 dapat digunakan untuk mendaftarkan nama domain baru atau menggunakan nama domain yang Anda miliki. Sehingga, ini memudahkan Anda untuk mengelola semua nama domain dalam satu lokasi.

Amazon CloudFront
Berkaca dari kasus website yang telah kita bicarakan dari awal, ada layanan yang bisa mempercepat pengiriman aset website, yaitu Amazon CloudFront--telah kita bahas di modul sebelumnya yah.
Amazon CloudFront adalah layanan yang dapat membantu Anda untuk mengirimkan konten (data, aplikasi, maupun API) ke pelanggan di seluruh dunia dengan aman dan latensi rendah. Konten yang dimaksud ini bisa berbagai hal, seperti data, video, aplikasi, dan API.
Jika Anda ingat, kita telah belajar tentang Edge locations sebelumnya. Edge locations menyajikan konten sedekat mungkin dengan pelanggan, salah satu bagiannya adalah *content delivery network* (CDN) atau jaringan pengiriman konten. Sebagai pengingat, CDN adalah jaringan yang membantu Anda untuk memberikan konten kepada pelanggan berdasarkan lokasi geografis mereka.
CloudFront sangat terintegrasi dengan layanan lainnya seperti AWS Web Application Firewall (WAF), AWS Certificate Manager, Amazon Route 53, Amazon S3, dan lainnya.
Anda dapat memulai menggunakan Amazon CloudFront hanya dalam hitungan menit menggunakan layanan AWS yang sudah Anda kenal: API, AWS Management Console, Command Line Interface (CLI), dan SDK.

Studi Kasus: Bagaimana Amazon Route 53 dan Amazon CloudFront Mengirimkan Konten
Di modul sebelumnya kita telah belajar tentang Amazon Route 53 dan Amazon CloudFront. Kali ini kita akan menelaah bagaimana kedua layanan tersebut berkolaborasi untuk mengirimkan konten kepada pelanggan.

Misalnya, Anda memiliki aplikasi yang berjalan di beberapa Amazon EC2 instance. Instance ini berada di dalam Auto Scaling group (grup Auto Scaling) yang dilampirkan ke Application Load Balancer. Perhatikan gambar berikut:



Dengan arsitektur di atas, sekarang proses memuat website pun bisa semakin lebih cepat. Mari kita uraikan arsitektur tersebut:

1. Pelanggan memasukkan alamat website di browser mereka.
2. Permintaan tersebut akan dikirimkan ke Amazon Route 53 untuk diidentifikasi alamat IP-nya.
3. Informasi tersebut kemudian dikirim kembali ke pelanggan.
4. Selanjutnya, permintaan dari pelanggan tersebut ditransfer ke Edge locations terdekat melalui Amazon CloudFront.
5. Setelah itu, Amazon CloudFront yang terhubung ke Application Load Balancer akan mengirimkannya ke Amazon EC2 instance.
6. Dan akhirnya, website pun termuat.

Selamat! Sekarang Anda sudah mengerti bagaimana proses dibalik pemuatan suatu website yang menggunakan penggabungan layanan Amazon CloudFront dan Amazon Route 53.

Ikhtisar

Dulunya, jaringan adalah ranah eksklusif para ahli topologi saja. Tapi kini dengan AWS, jaringan dapat disimplifikasi dan diabstraksi. Selama Anda dapat menjawab pertanyaan tentang siapa saja yang diizinkan untuk berkomunikasi satu sama lain, maka Anda dapat mengatur jaringan di AWS.

Sejauh ini, kita telah menyelami:

- Dasar-dasar Amazon Virtual Private Cloud dan cara mengisolasi beban kerja di AWS.
- Fundamental keamanan jaringan, seperti Internet Gateway, network ACL, dan security group. Semua hal tersebut adalah metode yang memudahkan Anda untuk membuat jaringan yang dapat mengizinkan akses traffic sehat sembari menghentikan serangan subversif sebelum mencapai instance Anda.
- Cara untuk terhubung ke AWS melalui VPN dan bahkan Direct Connect, yaitu jalur aman yang dienkripsi melalui internet umum atau fiber eksklusif yang hanya digunakan oleh Anda sendiri.
- Jaringan global yang disediakan AWS menggunakan Edge locations.

- Bagaimana menggunakan Amazon Route 53 untuk DNS.
- Menggunakan Amazon CloudFront untuk menyimpan konten ke dalam cache yang lebih dekat dengan pelanggan.

Meskipun apa yang kita pelajari ini hanya menggores permukaan saja dari banyak hal lain yang dapat dilakukan dengan jaringan AWS, tapi semoga Anda paham akan hal-hal penting yang diperlukan untuk memulai menggunakan AWS.

Semoga Anda makin semangat ya untuk melaju ke modul berikutnya!

Materi Pendukung

Untuk mempelajari lebih lanjut tentang konsep yang sudah kita telaah di modul ini, tinjau sumber berikut:

- Networking and Content Delivery on AWS
- AWS Networking & Content Delivery Blog
- Amazon Virtual Private Cloud
- What is Amazon VPC?
- How Amazon VPC works

MODUL 4

Pengenalan ke Penyimpanan dan Database

Oke, kembali ke skenario. Sekarang kedai kopi kita telah beroperasi dengan cukup baik. Kita punya banyak pelanggan yang puas.

Faktanya, kita juga telah mempunyai arsitektur yang elastis, dapat diskalakan, tahan akan bencana, dan secara biaya pun telah optimal. Bahkan, sekarang kita memiliki jaringan global yang sangat aman dan dapat diterapkan sepenuhnya secara terprogram.

Melihat semakin banyaknya pelanggan setia yang hadir ke kedai kopi, maka kita harus memberikan apresiasi kepada mereka. Tapi, berbentuk apa ya?

Hmm. Bagaimana dengan program loyalitas pelanggan? Kita bisa membagikan kartu stempel kepada mereka yang sering memesan kopi di tempat kita. Tetapi, jujur saja, kita tak akan dapat melacak kartu tersebut dan mengenal pelanggan kita dengan baik.

Jadi, sepertinya kita membutuhkan kartu digital yang dapat melacak riwayat pemesanan pelanggan (apa yang mereka pesan atau berapa banyak yang mereka beli). Dengan demikian, kartu ini akan membantu c*ustomer* kita mendapat apresiasi terbaik atas loyalitas mereka. Kita pun jadi bisa mengenal basis pelanggan dengan lebih baik dan lebih mudah.

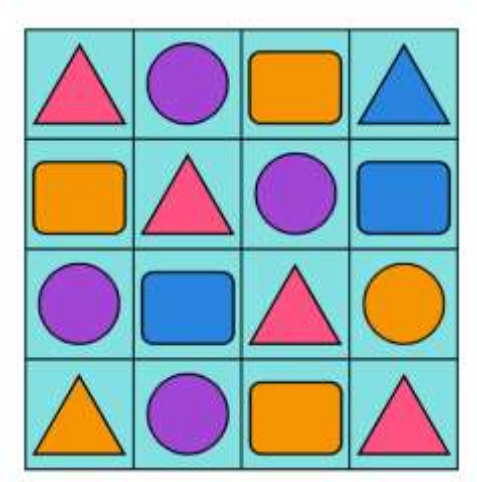
Nah, itu berarti kita akan membutuhkan penyimpanan dan database (basis data). Ingat, bukan sembarang database. Pilihlah penyimpanan dan database yang tepat sesuai dengan masing-masing kebutuhan Anda.

Mari kita pelajari tentang berbagai layanan AWS yang dapat membantu Anda membangun solusi data yang sempurna.

Instance Store dan Amazon Elastic Block Store (Amazon EBS)
Saat Anda menjalankan aplikasi di AWS, tentunya aplikasi tersebut memerlukan akses ke CPU, memori, jaringan, dan penyimpanan. Nah, untungnya, EC2 instance dapat memberikan akses ke semua komponennya. Untuk saat ini, mari kita fokus pada penyimpanan.

Ketika aplikasi berjalan di EC2 instance, mereka kerap kali membutuhkan akses ke *block-level storage* (penyimpanan tingkat blok).

Jika Anda kurang kenal dengan istilah block-level storage, maka anggaplah ia sebagai tempat menyimpan file. File adalah serangkaian *byte* (bita) yang disimpan di dalam blok pada disk.



Pada saat file pada disk tersebut diperbarui, ia tak akan menimpa seluruh rangkaian blok, melainkan memperbarui bagian yang berubah saja. Dengan sistem seperti ini, penyimpanan untuk aplikasi (database, perangkat lunak perusahaan, atau sistem file) jadi lebih efisien.

Hmm. Mungkin penjelasan di atas sangat teknis ya. Oke, mari kita sederhanakan. Apakah sekarang Anda sedang menggunakan laptop atau komputer pribadi?

Nah! Itu berarti Anda sedang mengakses block-level storage alias penyimpanan tingkat blok. Block-level storage dalam kasus ini adalah hard drive (cakram keras) di komputer Anda.

EC2 instance juga memiliki hard drive dengan beberapa tipe yang berbeda.

Instance Store

Instance store (tempat penyimpanan instance) adalah penyimpanan block-level storage, sementara untuk Amazon EC2 instance. Saat Anda meluncurkan EC2 instance--tergantung tipe EC2 instance yang Anda pilih--biasanya sudah tersedia penyimpanan lokal alias instance store volume di dalamnya.

Volume ini secara fisik terpasang ke *host* (mesin fisik), yaitu tempat di mana EC2 instance Anda berjalan. Anda dapat melakukan proses *write* (menulis) data padanya seperti *hard drive* pada umumnya.

Namun masalahnya, jika Anda menghentikan atau mengakhiri EC2 instance tersebut, maka semua data di sana akan terhapus. Ini terjadi karena ketika Anda memulai instance dari status *stop* alias berhenti kemungkinan EC2 instance akan berjalan di host lain, yang mana instance store volume tersebut tidak berada di sana.

Ingat! EC2 instance adalah mesin virtual. Oleh karena itu, host yang mendasarinya dapat berubah pada saat instance berhenti dan memulai.

Karena sifatnya yang sementara inilah biasanya instance store volume digunakan untuk penyimpanan data yang sering berubah, seperti *cache, temporary file* (file sementara), data yang dapat dibuat ulang dengan mudah, dan konten sementara lainnya. Tapi, ingat! Jangan simpan data penting Anda ke dalam instance store volume.

Lantas, bagaimana solusinya jika kita ingin menyimpan data secara persisten dan berada di luar siklus hidup EC2 instance? Atau dengan kata lain, bagaimana kita ingin menyimpan data yang takkan terhapus walau EC2 instance berhenti? Nah, jangan khawatir, di sinilah Anda perlu mengenal layanan Amazon Elastic Block Store (Amazon EBS).

Amazon Elastic Block Store (Amazon EBS)
Amazon Elastic Block Store (Amazon EBS) adalah layanan yang menyediakan *block-level storage* (penyimpanan tingkat blok) yang dapat Anda gunakan bersama dengan Amazon EC2 instance.

Amazon EBS memungkinkan Anda untuk membuat *hard drive* virtual (EBS volume) yang kemudian bisa di-*attach* (dipasang) ke EC2 instance. EBS volume ini merupakan penyimpanan yang terpisah dari *instance store volume*. Ia pun tak terikat langsung ke *host* yang menjalankan EC2 instance Anda.

Lalu, bagaimana cara membuat EBS volume? Sebenarnya, mudah saja. Anda hanya perlu menentukan konfigurasinya (seperti ukuran dan tipe) sesuai dengan kebutuhan. Jika sudah, Anda bisa meluncurkannya dan memasangkannya ke Amazon EC2 instance.

Sekarang, jika Anda menghentikan lalu memulai EC2 instance, data yang Anda simpan di EBS volume akan tetap ada.

Karena EBS volume digunakan untuk kebutuhan data yang persisten, maka penting untuk Anda melakukan *backup* (pencadangan) data. Anda dapat menjalankan *incremental backup* (pencadangan secara inkremental) dari EBS volume dengan membuat Amazon EBS snapshot.

Amazon EBS snapshot disimpan secara bertahap/inkremental. Itu berarti pada saat pertama kali proses pencadangan dilakukan, ia akan menyalin semua data yang ada di EBS volume. Namun, untuk pencadangan berikutnya, ia hanya menyimpan blok data yang berubah dari snapshot terakhir.

Tentu, incremental backup ini sesungguhnya berbeda ya dengan full backup (pencadangan penuh). Full backup itu akan menyalin semua data yang ada di dalam volume setiap pencadangan dilakukan, sementara incremental backup hanya mencadangkan data yang berubah (delta) dari pencadangan sebelumnya.

Lakukanlah snapshot untuk EBS volume secara teratur. Dengan begitu, jika sebuah drive *corrupted* atau rusak, maka Anda tidak akan kehilangan data, melainkan Anda dapat memulihkannya dari *snapshot*.

Amazon Simple Storage Service (Amazon S3)

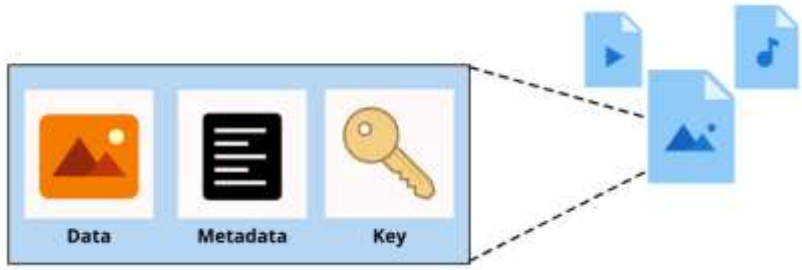
Sekarang kita masuk ke materi yang berkenaan tentang Amazon Simple Storage Service (Amazon S3). Dari namanya, mungkin Anda sudah menduga bahwa ini adalah layanan penyimpanan yang sederhana.

Tahukah Anda? Sebagian besar bisnis memiliki data yang perlu disimpan di suatu tempat. Misalnya untuk kedai kopi kita, ini bisa berupa struk, gambar, spreadsheet Excel, video pelatihan karyawan, bahkan file teks.

Nah, Anda dapat menyimpan file-file tersebut di Amazon S3 karena ia merupakan layanan yang dapat menyimpan dan mengambil data dalam jumlah tak terbatas pada skala apa pun.

Dengan Amazon S3, data disimpan sebagai objek. Objek tersebut tidak akan ditaruh di direktori file, melainkan data Anda akan disimpan di dalam *bucket*. Sederhananya begini. Anggaplah file yang ada di hard drive Anda sebagai objek dan direktori file adalah *bucket*.

Amazon S3 juga merupakan *object-level storage* (penyimpanan tingkat objek). Setiap objek terdiri dari data, metadata, dan kunci.



Mari kita lihat lebih lanjut. Data yang dimaksud itu bisa bermacam-macam, seperti gambar, video, dokumen teks, atau jenis file lainnya. Lalu, metadata adalah informasi yang berisi tentang apa itu data, cara penggunaannya, ukuran objeknya, dan sebagainya. Nah, *key* (kunci) pada suatu objek adalah identifier/pengenal yang unik.

Ukuran maksimum dari setiap objek yang dapat Anda unggah adalah 5 terabyte. S3 juga memiliki fitur *versioning* dengan membuat *object version* (versi objek). Maksudnya, Anda akan tetap dapat memiliki versi sebelumnya dari objek tersebut walaupun secara tidak sengaja menimpa objek dengan isi yang berbeda.

Selain itu, Anda juga dapat membuat beberapa bucket lalu menentukan *permission* (izin) untuk membatasi siapa yang dapat melihat atau mengakses objek di dalamnya.

Hal lain yang perlu diingat adalah ketika Anda mengubah file di *block-level storage* (penyimpanan tingkat blok), hanya bagian yang diubah saja yang akan diperbarui. Sebaliknya, saat Anda mengubah file di *object-level storage* (penyimpanan tingkat objek), maka keseluruhan objek yang akan diperbarui.

Oke, sekarang mari kita membahas *storage class* atau kelas penyimpanan yang ada pada Amazon S3. Maksudnya, ia menawarkan mekanisme untuk kasus penggunaan penyimpanan yang berbeda-beda. Misalnya untuk data yang sering diakses atau bahkan data audit yang perlu disimpan selama beberapa tahun. Mari kita uraikan.

- **S3 Standard**
  S3 Standard hadir dengan daya tahan 11 sembilan. Artinya, objek yang disimpan akan memiliki 99,999999999% probabilitas tetap utuh setelah jangka waktu satu tahun. Waw, Itu cukup tinggi, bukan?

  Selain itu, data disimpan setidaknya di tiga data center. Sehingga, ini membuatnya dapat menawarkan *high availability* (ketersediaan tinggi) bagi objek. S3 Standard menjadi pilihan yang ideal untuk berbagai kasus penggunaan, seperti website, distribusi konten, dan analitik data.

  Anda juga dapat menggunakan Amazon S3 untuk meng-*hosting*website statis, yaitu jenis website yang paling dasar dan berisi halaman web dengan konten statis. Caranya cukup mudah, Anda hanya perlu:
    - Unggah semua file HTML, aset web statis, dan sebagainya ke dalam bucket.
    - Centang opsi untuk meng-*hosting* website statis.
    - Lalu buka website tersebut dengan memasukkan URL *bucket* dan ta-da! Website instan.

  Cara yang cukup mengagumkan ya untuk memulai blog tentang kedai kopi kita.

- **S3 Standard-Infrequent Access (S3 Standard-IA)**
  Kelas penyimpanan jenis ini digunakan untuk data yang jarang diakses tetapi membutuhkan proses cepat saat dibutuhkan. Artinya, opsi ini adalah tempat yang ideal untuk menyimpan *backup* (cadangan), *disaster recovery file* (file pemulihan bencana), atau objek apa pun yang memerlukan penyimpanan jangka panjang.

- **S3 One Zone-Infrequent Access (S3 One Zone-IA)**
  Berbeda dengan S3 Standard dan S3 Standard-IA yang menyimpan data minimal di tiga Availability Zone, kelas penyimpanan S3 One Zone-IA menyimpan data hanya di satu Availability Zone.

  Nah, ini menjadikannya kelas penyimpanan yang perlu Anda pertimbangkan jika memiliki kondisi seperti berikut:
    - Ingin menghemat biaya penyimpanan.
    - Data dapat diproduksi ulang dengan mudah jika terjadi kegagalan di Availability Zone.

- **S3 Intelligent-Tiering**
  Pada kelas penyimpanan S3 Intelligent-Tiering, Amazon S3 memantau pola akses objek. Jika Anda tidak pernah mengakses objek selama 30 hari berturut-turut, Amazon S3 akan memindahkannya secara otomatis ke S3 Standard-IA.

Atau sebaliknya, jika Anda mengakses kembali objek di S3 Standard-IA, Amazon S3 memindahkannya secara otomatis ke S3 Standard.

- **S3 Glacier**
  Opsi kelas penyimpanan ini ideal untuk data audit. Katakanlah Anda perlu menyimpan data selama beberapa tahun untuk tujuan audit. Sehingga, tidak memerlukan proses akses yang langsung pada saat itu juga. Maka dari itu, Anda dapat menggunakan Amazon S3 Glacier untuk mengarsipkan data tersebut.

  Perlu diingat bahwa untuk mengakses objek yang disimpan di S3 Glacier, Anda memerlukan waktu beberapa menit hingga beberapa jam.

  Lalu bagaimana cara menggunakan Glacier? Mudah saja, Anda hanya perlu memindahkan data ke sana atau dengan membuat *vault* (brankas) lalu mengisinya dengan arsip.

  Jika Anda memiliki *compliance requirement* (persyaratan kepatuhan) tentang penyimpanan data untuk periode waktu tertentu, Anda dapat menerapkan *S3 Glacier vault lock policy* untuk mengunci *vault* Anda.

  Kontrol yang dapat Anda tentukan pada *vault lock policy* adalah *write once/read many* (WORM) alias cukup menulis data sekali lalu membacanya berkali-kali. Selain itu, Anda juga dapat mengunci kebijakan dari pengeditan di masa mendatang sehingga setelah terkunci, kebijakan tersebut tidak dapat lagi diubah.

  Anda juga memiliki tiga opsi untuk pengambilan data yang berkisar dari hitungan menit hingga jam. Bahkan, Anda memiliki pilihan untuk mengunggah langsung ke kelas Glacier atau menggunakan S3 Lifecycle policies.

  S3 Lifecycle policies adalah kebijakan yang bisa Anda buat untuk memindahkan data secara otomatis antar *storage class* (kelas penyimpanan).

  Misalnya, Anda perlu menyimpan objek dalam S3 Standard selama 90 hari. Lalu, Anda ingin memindahkannya ke S3-IA selama 30 hari ke depan. Kemudian setelah total 120 hari, Anda ingin memindahkannya ke S3 Glacier.

  Nah, kebutuhan semacam itu dapat Anda capai dengan S3 Lifecycle policies yang juga merupakan contoh lain dari layanan AWS terkelola.

- **S3 Glacier Deep Archive**
  Opsi ini merupakan kelas penyimpanan objek yang memiliki biaya terendah dan ideal untuk pengarsipan.

> Saat Anda ingin memilih antara menggunakan Amazon S3 Glacier atau Amazon S3 Glacier Deep Archive, coba pertimbangkan seberapa cepat Anda perlu mengakses objek yang diarsipkan.
>
> Di S3 Glacier waktu pengaksesan suatu objek berlangsung beberapa menit hingga jam saja, sementara dengan S3 Glacier Deep Archive Anda memerlukan waktu 12 hingga 48 jam.

Oke, sampai tahap ini, Anda sudah belajar tentang Amazon EBS dan juga Amazon S3. Mungkin sekarang Anda akan mengerutkan dahi, “Kapan kita harus menggunakan Amazon EBS dan Amazon S3?”

Baiklah, mari kita bandingkan dua layanan tersebut secara lebih mendalam.

Amazon EBS	**Amazon S3**
Block-level storage (penyimpanan tingkat blok), dapat berukuran hingga 16 terabyte, dan tetap tersedia walau Amazon EC2 instance dihentikan.	*Object-level storage* (penyimpanan tingkat objek) yang memiliki ukuran tak terbatas dan mampu menampung setiap objek hingga 5.000 gigabyte. Setiap objek di S3 memiliki URL yang dapat Anda kontrol hak aksesnya.
Hadir dalam bentuk hard disk drive (HDD) dan solid state drive (SSD).	*Write once/read many* (cukup menulis data sekali lalu membacanya berkali-kali) dan menawarkan 99,999999999% ketahanan.
Cocok untuk mengedit video berukuran 80 gigabyte.	Ideal untuk website yang menjalankan analisis foto.
Memecah file menjadi bagian atau blok komponen kecil. Sehingga, saat Anda melakukan perubahan dan menyimpannya, sistem hanya memperbarui blok tempat bit tersebut berada.	Memperlakukan file apa pun sebagai objek yang lengkap dan terpisah sehingga ia cocok digunakan sebagai dokumen, gambar, dan video yang diunggah dan sebagai objek keseluruhan.
	Termasuk layanan serverless (tanpa server) sehingga tidak memerlukan Amazon EC2 instance. Tak perlu khawatir akan strategi *backup* (pencadangan) karena ia merupakan layanan *backup* di AWS.
	Penghematan biaya yang substansial melebihi EBS walau dengan beban penyimpanan yang sama.

Kesimpulannya, jika Anda memiliki objek atau file yang lengkap dan hanya membutuhkan sesekali perubahan, maka pilihlah Amazon S3. Namun, jika Anda

membutuhkan proses *read* (baca) data yang kompleks, maka tentu saja Anda perlu memilih Amazon EBS.

Jadi, memilih penyimpanan yang tepat itu tergantung pada kebutuhan beban kerja Anda. Setiap layanan sebenarnya merupakan solusi yang tepat untuk kebutuhan tertentu. Pahamilah apa yang Anda butuhkan, maka Anda akan tahu layanan mana yang ideal.

Amazon Elastic File System (Amazon EFS)

Di modul ini, kita akan membahas tentang layanan *file storage* alias penyimpanan file, yang berarti beberapa *client*--pengguna, aplikasi, server, dsb--dapat mengakses data yang disimpan di folder file secara bersamaan.

Dalam pendekatan ini, file server menggunakan *block storage* (penyimpanan blok) dengan *local file system* (sistem file lokal) untuk mengatur file. Nah, client dapat mengakses data di dalamnya melalui *file path* (jalur file).

Dibandingkan dengan block storage dan object storage, file storage ini sangat ideal untuk kasus penggunaan di mana beberapa layanan dan sumber daya perlu mengakses data yang sama pada waktu yang sama.

Layanan AWS yang termasuk ke dalamnya adalah Amazon Elastic File System, atau juga disebut dengan EFS. Amazon EFS adalah sistem file terkelola yang bisa diskalakan dan dapat digunakan oleh layanan AWS Cloud dan sumber daya di data center on-premise.

Sudah sangat umum bagi perusahaan untuk berbagi sistem file di seluruh aplikasi mereka. Mari kita ambil contoh suatu kasus. Misalnya Anda memiliki beberapa server yang menjalankan analitik pada sejumlah besar data yang disimpan dalam sistem file bersama di data center on-premise.

Karena berjalan di on-premise, tentu saja Anda harus memastikan bahwa kapasitas penyimpanan di sana dapat menyesuaikan dengan jumlah data yang Anda simpan. Anda juga harus memastikan data tersebut telah dicadangkan dan disimpan secara redundan (di beberapa tempat). Satu lagi, Anda pun harus mengelola semua servernya. Repot ya?

Untungnya, AWS hadir memberikan solusi agar Anda tak perlu lagi khawatir untuk mengurus semuanya itu. AWS akan mengelola semua pekerjaan terkait *scaling* (penyesuaian kapasitas) dan replikasinya untuk Anda.

Dengan EFS, Anda dapat memiliki beberapa *instance* yang mengakses data secara bersamaan. Ia akan melakukan *scaling up* dan *scaling down*--keduanya telah kita

bahas di modul tentang penyesuaian kapasitas--sesuai kebutuhan secara otomatis. Sangat keren, bukan?

Oke, mungkin Anda akan berpikir, "*Loh*, Amazon EBS juga bisa menyimpan file dan dapat diakses dari EC2 instance. Jadi, apa perbedaan sebenarnya?"

Jawabannya sangat sederhana. Amazon EBS volume dilampirkan ke EC2 instance dan merupakan *Availability Zone-level resource* atau sumber daya tingkat Availability Zone. Itu artinya, EBS akan menyimpan data hanya di satu Availability Zone (AZ). Terlebih lagi, jika Anda ingin memasang EC2 ke EBS, maka Anda harus berada di AZ yang sama.

Dengan Amazon EBS, Anda dapat menyimpan file, menjalankan database, atau menyimpan aplikasi di dalamnya. Ia adalah *hard drive* (cakram keras). Namun, jika Anda membuat EBS volume sebesar 2 terabyte lalu mengisinya hingga penuh, ia tidak akan serta-merta melakukan proses scaling dengan sendirinya. Itulah EBS.

Lalu bagaimana dengan EFS? Amazon EFS memungkinkan beberapa instance untuk melakukan proses *read* (membaca) dan *write* (menulis) data darinya pada saat bersamaan.

Tetapi, ia bukan sekadar hard drive kosong yang dapat Anda gunakan untuk menyimpan data. Amazon EFS adalah sistem file untuk Linux dan merupakan Regional resource (sumber daya regional). Itu berarti data akan disimpan di beberapa AZ. Dengan demikian, setiap EC2 instance yang berada di Region yang sama dapat menyimpan data ke sistem file Amazon EFS.

Nah, jika Anda menyimpan banyak data ke EFS, ia secara otomatis akan melakukan scaling bahkan hingga petabyte tanpa mengganggu aplikasi.

Amazon Relational Database Service (Amazon RDS)

Sampai sini, kita sudah banyak belajar tentang berbagai sistem penyimpanan di AWS yang dapat membantu Anda untuk menyelesaikan persoalan terkait kartu digital untuk pelanggan setia di kedai kopi, sebagaimana yang telah kita paparkan di awal modul.

Tetapi, ketahuilah! Anda juga perlu menjaga relasi antara berbagai tipe data. Tunggu, apa maksudnya?

Begini. Misalnya, seorang pelanggan di kedai kopi telah memesan minuman yang sama beberapa kali. Nah, melihat hal ini, mungkin Anda sebagai pemilik kedai kopi ingin menawarkan diskon promosi untuk pembelian berikutnya.

Tentu, Anda membutuhkan suatu cara untuk melacak relasi/hubungan semacam ini, bukan?

Solusi terbaik untuk masalah tersebut adalah dengan menggunakan *relational database management system* (RDBMS) alias sistem manajemen database/basis data relasional. Artinya, data yang kita simpan dapat memiliki relasi dengan bagian data lainnya.

Dalam dunia database, Anda akan sering mendengar kata query atau kueri. Itu adalah sekumpulan instruksi khusus untuk mengekstraksi data.

Nah, database relasional menggunakan *structured query language* (SQL) alias bahasa kueri terstruktur untuk menyimpan dan membuat kueri data. Pendekatan semacam ini memungkinkan data disimpan dengan cara yang mudah dimengerti, konsisten, dan dapat diskalakan.

Inilah solusi dari persoalan di awal tadi. Dengan database relasional, sekarang Anda dapat menulis kueri SQL untuk mengidentifikasi minuman apa yang sering dibeli oleh masing-masing pelanggan.

Contoh sederhana dari database relasional adalah sistem manajemen inventaris di skenario kedai kopi kita. Setiap *record* (kumpulan data) di database mencakup data untuk satu item, seperti nama produk, ukuran, harga, dsb.

Berikut adalah contoh sederhana dari tabel database relasional:

ID	Product name	Size	Price
1	Kopi gula aren	Besar	Rp25.000
2	Kopi susu	Sedang	Rp15.000

Sekarang pertanyaannya adalah, layanan AWS apa yang mendukung database relasional?

Sambutlah, Amazon Relational Database Service (Amazon RDS). Ia adalah layanan yang memungkinkan Anda untuk menjalankan database relasional di AWS Cloud.

Amazon RDS adalah layanan yang terkelola dan mendukung 6 (enam) mesin database, di antaranya:

- Amazon Aurora
- PostgreSQL
- MySQL
- MariaDB
- Oracle Database
- Microsoft SQL Server

Ketahuilah! Jika Anda memiliki data center on-premise yang menjalankan salah satu mesin database di atas, Anda bisa memindahkannya ke cloud dengan mudah. Bagaimana caranya?

AWS memungkinkan Anda untuk melakukan *Lift-and-Shift*, yaitu proses memigrasikan beban kerja dari on-premise ke AWS dengan sedikit atau bahkan tanpa modifikasi.

Contohnya, Anda bisa memindahkan database on-premise lalu menjalankannya di Amazon EC2. Dengan begitu, Anda mempunyai kendali atas variabel yang sama dengan keadaan di on-premise, seperti OS, memori, CPU, kapasitas penyimpanan, dsb. Ini merupakan entri yang cepat untuk menuju cloud, bukan?

Salah satu cara yang dapat Anda lakukan untuk mewujudkan proses migrasi ini adalah dengan menggunakan layanan Database Migration Service, yang nanti akan kita bahas.

Sekarang, mari kita kembali ke pembahasan mengenai Amazon RDS. Layanan Amazon RDS hadir dengan berbagai fitur, termasuk:

- *Automated patching* (memperbaiki masalah dengan memperbarui program).
- *Backup* (pencadangan).
- *Redundancy* (memiliki lebih dari satu instance untuk berjaga-jaga jika instance utama gagal beroperasi).
- *Failover* (instance lain akan mengambil alih saat instance utama mengalami kegagalan).
- *Disaster recovery* (memulihkan pascabencana).
- *Encryption at rest* (enkripsi data saat disimpan).
- *Encryption in-transit* (enkripsi data saat sedang dikirim dan diterima).

Semua hal di atas adalah proses yang biasanya harus Anda kelola sendiri jika menggunakan data center on-premise. Tentu ini menjadikannya pilihan menarik karena Anda tak pusing dengan pemeliharaan dan pengelolaan database. Karena faktanya, semua proses tersebut sangat sulit dan memakan waktu yang lama.

Pertanyaan selanjutnya muncul, “Adakah cara yang bahkan bisa lebih mudah lagi untuk menjalankan beban kerja database di cloud?”

Nah, mari kita sedikit membahas tentang layanan Amazon Aurora. Ia adalah opsi database relasional kelas enterprise/perusahaan yang terkelola oleh AWS. Layanan ini memiliki banyak fitur, seperti:

- Bisa bekerja secara kompatibel dengan MySQL dan PostgreSQL. Bahkan, dapat 5 kali lebih cepat dari database MySQL standar dan bisa 3 kali lebih cepat dari database PostgreSQL standar.
- Memberikan performa yang setara dengan database komersial dengan perbandingan biaya 1/10.
- Mampu memastikan replikasi data di seluruh fasilitas.
- Menerapkan hingga 15 *read replica* (replika baca).

- Mencadangkan secara kontinu ke Amazon S3.
- Menerapkan *point-in-time recovery* (Pemulihan data dari periode tertentu).

Oke, itulah dia penjelasan tentang database relasional. Masih ada *loh* materi menarik lainnya yang terkait dengan database, nantikan di modul berikutnya ya.

Amazon DynamoDB

Sebelumnya, kita telah belajar tentang database relasional. Database relasional--misalnya MySQL standar--mengharuskan Anda untuk menentukan skema dengan baik. Ia bisa jadi terdiri dari satu atau banyak tabel yang saling berhubungan. Barulah Anda bisa menggunakan SQL untuk membuat kueri data.

Jenis ini sering dipakai untuk banyak kasus penggunaan dan telah menjadi tipe database standar secara historis. Bagaimanapun, jenis database SQL yang kaku ini dapat memiliki masalah *scaling* dan kinerja saat berada di bawah tekanan.

Skema yang tetap (*fixed schema*) juga membuatnya tidak dapat memiliki variasi jenis data di dalam tabel. Jadi, database relasional bukanlah solusi terbaik untuk kumpulan data yang fleksibel dan membutuhkan akses kilat.

Selain itu, pada praktiknya, banyak juga yang menjalankan database SQL bukan untuk penggunaan relasional yang kompleks, melainkan hanya untuk tabel pencarian.

Nah, di sinilah Amazon DynamoDB hadir. Ia merupakan database nonrelasional (NoSQL) dan menggunakan jenis pendekatan pasangan *key-value* (kunci-nilai).

Dengan Amazon DynamoDB, Anda dapat membuat tabel, yakni tempat menyimpan dan membuat kueri data. Data diatur menjadi item/key dan item memiliki atribut/value.

Anda dapat menambah dan menghapus atribut dari item di dalam tabel kapan pun. Setiap item tidak harus memiliki atribut yang sama. Sehingga, ini akan sangat baik untuk kumpulan data yang memiliki beberapa variasi antara satu item dengan item lainnya.

Karena kueri pada database nonrelasional itu cenderung lebih sederhana, ini membuat Anda bisa fokus pada kumpulan item dari satu tabel, bukan kueri dari rentang beberapa tabel.

Berikut adalah contoh sederhana dari tabel database nonrelasional.

Key	Value
1	**Nama**: Budi
	Alamat: Kenanga 123

Key	Value
	Minuman favorit: Kopi gula aren
2	**Nama**: Siska
	Alamat: Mawar 321
	Tanggal Lahir: 5 Juli 1994

Mudah, bukan? Seperti yang telah kita pelajari sebelumnya, setiap item pada tabel tidak harus memiliki atribut yang sama.

Oke, mari kita lanjutkan pembahasannya.

Ketahuilah! Amazon DynamoDB ini adalah layanan yang terkelola sepenuhnya dan merupakan database *serverless* (tanpa server). Itu artinya Anda tak perlu mengelola instance atau infrastruktur dasarnya. Bahkan, Anda tidak perlu khawatir akan proses *scaling* (penyesuaian kapasitas) yang terjadi pada sistem.

Selain itu, Amazon DynamoDB juga menyimpan data di beberapa perangkat di seluruh availability zone. Sehingga, ini menjadikannya database yang *highly available* (sangat tersedia). Layanan ini memiliki kinerja yang sangat tinggi. Ia punya *response time* (waktu respons) kilat, yakni milidetik, yang akan sangat bermanfaat untuk aplikasi dengan potensi jutaan pengguna.

Ingat! Karena DynamoDB adalah database NoSQL alias nonrelasional, ia sangat ideal untuk kasus penggunaan tertentu, bukan untuk semua beban kerja.

Salah satu contoh penggunaan Amazon DynamoDB yang luar biasa adalah pada saat Amazon Prime Day pada tahun 2019. Prime Day adalah acara belanja tahunan eksklusif untuk anggota Amazon Prime.

Selama 48 jam Prime Day, ada 7,11 triliun panggilan ke API DynamoDB. Puncaknya bahkan ada pada 45,4 juta permintaan per detik. Waw, sangat andal, bukan?

Sekarang, mungkin Anda akan bertanya-tanya, “Kapan kita harus menggunakan Amazon RDS dan Amazon DynamoDB?”

Oke, mari kita bandingkan dua layanan tersebut secara lebih detail.

Amazon RDS	Amazon DynamoDB
Dirancang untuk mengurangi kerumitan administrator sekaligus memberikan *high availability* (ketersediaan tinggi) pada *recovery* (pemulihan) database Anda.	Menggunakan pasangan *key-value* tanpa memerlukan skema yang rumit serta dapat beroperasi sebagai database global hanya dengan satu klik.

Amazon RDS	Amazon DynamoDB
Anda dapat mengontrol data, skema, dan jaringan.	Memiliki *throughput* (jumlah data yang dapat dikirim dalam waktu tertentu) yang besar, *scaling* hingga petabyte, dan akses API secara detail.
Mampu membangun sistem analisis data yang kompleks.	Memungkinkan Anda membangun database yang kuat dan sangat cepat tanpa perlu fungsionalitas yang rumit.
Ideal untuk analisis sistem manajemen *supply chain* (rantai pasokan).	Cocok untuk penggunaan data yang fleksibel dan sederhana seperti daftar kontak karyawan yang berisikan nama, nomor telepon, email, ID karyawan, dsb.

Jadi, sebenarnya ini tergantung pada kebutuhan beban kerja Anda. Setiap layanan akan menjadi solusi yang tepat untuk kebutuhan tertentu. Maka pahamilah apa yang Anda butuhkan, dengan begitu Anda akan dapat memilih layanan mana yang ideal.

Amazon Redshift

Sedari tadi kita telah banyak membahas tentang jenis alur kerja dengan kebutuhan proses data yang cepat, andal, dan terkini. Yup! Database bisa menjadi solusinya karena ia dapat menangani 1.000 transaksi per detik, sangat tersedia, dan berdaya tahan.

Tapi terkadang, mungkin ada kebutuhan bisnis yang melebihi dari apa yang pernah kita alami. Maka dari itu, kita membutuhkan layanan yang dapat menganalisis data.

Tentu Anda dapat menggunakan satu layanan database untuk semua kebutuhan. Namun, database modern yang dirancang untuk penggunaan kueri berkinerja tinggi dan *real time*, bukanlah yang paling ideal. Simak penjelasan berikut.

Database relasional akan sangat ideal untuk jenis pekerjaan yang membutuhkan fungsionalitas *read/write* (membaca/menulis) data karena ia dapat menanganinya secara real time.

Namun, masalah akan muncul saat Anda menggunakannya untuk jenis pekerjaan yang lain. Misalnya, jika Anda memakai database relasional untuk kebutuhan analisis historis, maka operasi pengumpulan data akan terus memprosesnya dan tak akan pernah berhenti. Dengan telemetri modern, volume data bisa membanjiri database. Bahkan, database relasional tradisional yang terkuat sekalipun tak bisa menanganinya.

Selain itu, data yang bervariasi juga akan menjadi masalah. Misalkan Anda ingin menjalankan proyek *business intelligence* (inteligensi bisnis) untuk data dari penyimpanan yang berbeda, seperti inventaris, keuangan, dan sistem penjualan ritel.

Mungkin, solusinya adalah dengan menggunakan satu kueri untuk beberapa database sekaligus. Tetapi faktanya, database tradisional tidak bisa menanganinya dengan mudah. Nah, ketika data menjadi semakin kompleks untuk ditangani oleh database relasional tradisional, maka *data warehouse* (gudang data) adalah solusinya. Data warehouse dirancang secara spesifik untuk jenis *big data* semacam ini, yaitu analitik historis, bukan analisis operasional.

Mari kita perjelas. Analitik historis itu seperti pertanyaan, “Tunjukkan angka penjualan satu jam terakhir di semua toko.” Intinya, data sudah siap pada saat diproses. Data tidak akan berubah lagi karena data ini adalah data historis yang sudah terjadi sebelumnya.

Bandingkan pertanyaan itu dengan, "Berapa kantong kopi yang masih ada di inventaris kita sekarang?" Yang mana data tersebut bisa berubah bahkan pada saat kita bertanya. Selama pertanyaan Anda melihat ke belakang alias lampau, maka data warehouse adalah solusi yang tepat untuk lini *business intelligence* tersebut.

Ada banyak solusi data warehouse yang beredar di pasaran. Namun, dengan itu semua, Anda mungkin tetap harus melakukan banyak *undifferentiated heavy lifting* (proses kerja yang tidak menambah nilai bagi perusahaan) guna menjaga data warehouse agar tetap dikonfigurasi, senantiasa tangguh, dan *scaling* secara berkelanjutan.

Tetapi, bukankah lebih baik jika Anda fokus pada data daripada perawatan dan pengelolaan mesin yang tak terhindarkan?

Anda mungkin perlu berkenalan dengan layanan yang satu ini, yaitu Amazon Redshift. Amazon Redshift adalah layanan data warehousing yang dapat Anda gunakan untuk analitik big data. Layanan ini menawarkan kemampuan untuk mengumpulkan data dari banyak sumber dan membantu Anda memahami hubungan dan tren di seluruh data. Selain itu, ia juga dapat diskalakan secara masif.

Adalah hal yang wajar untuk Redshift nodes (kumpulan sumber daya komputasi) tersedia dalam berbagai ukuran *petabyte*. Kolaborasikan Redshift nodes dengan Amazon Redshift Spectrum sehingga Anda pun dapat langsung menjalankan satu kueri SQL untuk *exabyte* data tidak terstruktur yang berjalan di *data lake* (penyimpanan untuk data terstruktur dan tidak terstruktur pada skala apa pun).

Selebihnya lagi, Amazon Redshift ini bukan hanya sekadar mampu menangani *data set* (kumpulan data) yang sangat besar, melainkan juga dapat mencapai kinerja hingga 10 kali lebih tinggi--dengan menggunakan berbagai inovasi--daripada database tradisional dalam hal beban kerja business intelligence.

Kita tidak akan memaparkan terperinci tentang cara kerja Amazon Redshift ya. Intinya, pahami bahwa saat Anda perlu solusi business intelligence big data, ia hadir

memudahkan Anda untuk memulainya hanya dengan satu panggilan API. Dengan begitu, akan lebih sedikit waktu buat menunggu hasil dan lebih banyak waktu untuk mendapatkan jawaban.

AWS Database Migration Service

Wah, tak terasa ya kita telah banyak membahas berbagai opsi database di AWS. Nah, sekarang mungkin Anda akan mengerutkan dahi, “Bagaimana kalau kita sudah punya database di on-premise atau mungkin di platform lain? Apa itu berarti kita harus memulai dari awal?”

Oh, tak perlu khawatir akan itu! AWS punya cara yang ajaib.

Mari berkenalan dengan AWS Database Migration Service (AWS DMS). Ia dapat memigrasikan database yang Anda miliki--baik relasional, nonrelasional (NoSQL), atau tipe penyimpanan data lain--ke AWS dengan mudah dan aman.

Pada dasarnya, proses migrasi itu membutuhkan sumber dan target database. Nah, dengan AWS DMS:

- Database sumber tetap beroperasi penuh selama proses migrasi.
- *Downtime* (waktu henti) diminimalkan untuk aplikasi yang bergantung pada database tersebut.
- Database sumber dan target *tidak* harus bertipe sama.

Oke. Meski tak harus bertipe sama, mari mulai pembahasan kita dari proses migrasi database yang bertipe sama (homogen) terlebih dahulu, atau dikenal sebagai *homogenous database migration*. Proses semacam ini dapat memigrasikan database dari:

- MySQL ke Amazon RDS for MySQL.
- Microsoft SQL Server ke Amazon RDS for SQL Server.
- Bahkan, Oracle ke Amazon RDS for Oracle.

Karena sumber database dan target memiliki struktur skema, tipe data, dan kode database yang sama, hasilnya proses migrasi akan berlangsung mudah.

Nah, contoh dari sumber database ini dapat berupa database yang berjalan di on-premise, Amazon EC2 instance, ataupun Amazon RDS. Sementara untuk database target, ia bisa saja database yang berada di Amazon EC2 maupun Amazon RDS.

Cara kerja migrasi homogen ini pun cukup sederhana. Anda hanya perlu membuat tugas migrasi, mengoneksikannya ke database sumber dan target, lalu mulai prosesnya dengan mengeklik tombol. Sisanya, akan ditangani oleh AWS DMS. Keren, bukan?

Sekarang, mari kita beralih ke jenis kedua, yakni migrasi database heterogen alias *heterogeneous database migration*. Berbeda dengan jenis pertama, migrasi heterogen ini terjadi saat database sumber dan target memiliki tipe yang berbeda. Ia juga merupakan proses dua langkah.

Maksudnya, karena struktur skema, tipe data, dan kode database asal berbeda dengan database target, maka kita perlu mengubahnya terlebih dahulu menggunakan AWS Schema Conversion Tool. Setelah itu, barulah kita gunakan AWS DMS untuk memigrasikannya.

Nah, selain dari yang dijelaskan di atas, AWS DMS juga dapat digunakan untuk:

- **Migrasi ke database pengembangan dan pengujian**
  Contoh penggunaannya adalah ketika Anda ingin melakukan pengujian aplikasi pada database produksi tanpa memengaruhi pengguna. Nah, dengan AWS DMS, Anda dapat memigrasikan salinan dari database produksi tersebut ke lingkungan pengembangan (*development*) atau pengujian (*testing*).
- **Konsolidasi database**
  Dengan AWS Database Migration Service, Anda dapat menggabungkan beberapa database menjadi satu database pusat.
- **Replikasi database yang kontinu**
  AWS DMS memungkinkan Anda untuk melakukan replikasi data yang berkelanjutan. Ini berguna untuk *disaster recovery* (pemulihan bencana) atau pemisahan geografis.

Oke, itulah dia materi tentang AWS Database Migration Service. Perjalanan kita belum berakhir ya, masih ada materi menarik lain di modul berikutnya. *Ganbatte!*

Layanan Database Tambahan

Masih ingatkah Anda dengan kalimat yang ada di materi pengantar? “Pilihlah penyimpanan dan database yang tepat untuk masing-masing kebutuhan Anda.”

Yap, itu benar! Memang akan lebih baik jika kita menjalankan database yang sesuai dengan data daripada memaksakan data agar kompatibel dengan database.

Ketahuilah! Tak ada satu pun database yang mampu menangani semua kebutuhan sekaligus.

Sejauh ini, kita telah membahas beberapa jenis database. Sebenarnya, masih ada banyak *loh* layanan database lain yang ditawarkan oleh AWS guna memenuhi kebutuhan-kebutuhan tertentu.

Tentu kita tidak akan membahas semuanya secara detail di modul ini, tapi tak ada salahnya juga untuk sedikit menyentuhnya agar Anda mengetahui bahwa layanan-layanan tersebut ada.

- **Amazon DocumentDB**
  Sebelumnya, kita telah menilik tentang layanan DynamoDB yang dapat digunakan untuk database pasangan *key-value* (kunci-nilai). Tetapi, bagaimana jika Anda memiliki kebutuhan yang lebih dari sekadar atribut sederhana? Seperti sistem manajemen konten yang lengkap misalnya.

  Nah, berarti Anda harus berkenalan dengan Amazon DocumentDB. Ia adalah layanan yang dapat membantu Anda untuk menangani manajemen konten, katalog, ataupun profil pengguna.

  Amazon DocumentDB merupakan layanan database dokumen yang mendukung beban kerja MongoDB (program database dokumen).

- **Amazon Neptune**
  Katakanlah Anda memiliki aplikasi web jejaring sosial dan ingin melacak: siapa terhubung dengan siapa. Pastinya akan sangat kaku dan sulit jika Anda mengelolanya di database relasional tradisional.

  Nah, Amazon Neptune dapat membantu Anda dalam hal tersebut. Ia adalah layanan *graph database* (database grafik) yang berguna untuk membuat dan menjalankan aplikasi dengan kumpulan data yang sangat terhubung, seperti *social networking* (jejaring sosial), *recommendation engines* (mesin pemberi rekomendasi), *fraud detection* (sistem pendeteksi penipuan), dan *knowledge graph* (grafik pengetahuan: kumpulan deskripsi yang saling terkait dari entitas).

- **Amazon Managed Blockchain**
  Jika Anda memiliki sistem *supply chain* (rantai pasokan) yang harus Anda lacak untuk menjamin tak akan ada suplai yang hilang, atau mungkin jika Anda mempunyai catatan perbankan/finansial yang aman, atau dalam kata lain adalah *blockchain* maka Anda dapat menggunakan Amazon Managed Blockchain.

Dengannya, Anda dapat membuat dan mengelola jaringan blockchain dengan *framework* (kerangka kerja) yang *open-source*. Barangkali Anda kurang familier dengan *blockchain*? Yup, ia adalah sistem *ledger* (kumpulan catatan riwayat aktivitas) terdistribusi yang memungkinkan banyak pihak menjalankan transaksi dan berbagi data tanpa otoritas pusat.

Amazon Managed Blockchain menggunakan desain *decentralized ownership* (kepemilikan yang terdesentralisasi). Artinya, beberapa pihak dapat bertransaksi tanpa harus saling mengenal atau mempercayai satu sama lain.

- **Amazon Quantum Ledger Database (Amazon QLDB)**
  Kalau memang Amazon Managed Blockchain tak dapat memenuhi kebutuhan Anda, gunakanlah Amazon Quantum Ledger Database (Amazon QLDB).

  Amazon QLDB adalah sistem pencatatan yang *immutable* di mana entri apa pun tidak akan pernah bisa dihapus dari audit. Layanan ini menyediakan log transaksi yang terpusat, tidak dapat diubah, dan dapat diverifikasi secara kriptografi.

  Amazon QLDB menggunakan desain *centralized ownership* (kepemilikan yang tersentralisasi). Maksudnya, otoritas pusat nan tepercaya memiliki, mengelola, dan membagikan ledger dengan sejumlah pihak tertentu.

Oke, kita berhenti sejenak terlebih dahulu. Seperti yang kita tahu bersama, database itu sesungguhnya adalah sistem yang hebat, bukan? Tetapi, bukankah akan jauh lebih baik jika ada cara yang dapat membuatnya lebih cepat?

Nah, AWS menawarkan beberapa opsi akselerator database yang dapat Anda gunakan untuk sejumlah skenario tertentu.

- **Amazon ElastiCache**
  AWS memungkinkan Anda untuk menambahkan lapisan *cache* pada database yang dapat meningkatkan *read time* (waktu baca) untuk permintaan umum dengan menggunakan Amazon ElastiCache. Ia mendukung dua jenis penyimpanan data: Redis dan Memcached.

- **Amazon DynamoDB Accelerator (DAX)**
  Jika Anda menggunakan Amazon DynamoDB, maka Anda harus mencoba layanan Amazon DynamoDB Accelerator (DAX), yakni *native caching layer* yang dirancang untuk meningkatkan waktu *read* (baca) untuk data nonrelasional.

Oke, itulah beberapa layanan database tambahan yang AWS miliki. Gunakanlah setiap layanan dengan tepat untuk masing-masing kebutuhan Anda.

Ikhtisar

Tibalah kita di penghujung dari modul ini. Banyak hal mengagumkan yang telah kita pelajari di modul ini. Kita telah memahami tentang semua jenis mekanisme penyimpanan di AWS. Mari kita rekap ulang, oke?

- Hal pertama yang telah kita pelajari adalah Amazon Elastic Block Store (Amazon EBS) volume. Ia dapat dilampirkan ke EC2 instance sehingga Anda bisa memiliki penyimpanan lokal yang bersifat persisten.
- Kita juga telah belajar tentang Amazon S3 yang dapat menyimpan objek di AWS hanya dengan mengeklik tombol atau panggilan API.
- Bahkan kita telah menilik berbagai opsi database relasional yang tersedia di AWS, yaitu Amazon RDS dan Amazon Aurora.
- Untuk beban kerja yang hanya membutuhkan pasangan *key-value* (kunci-nilai), AWS menawarkan layanan database nonrelasional yang disebut Amazon DynamoDB.
- Selanjutnya, ada Amazon EFS yang dapat Anda gunakan untuk kasus penyimpanan file secara bersamaan oleh Amazon Linux instance.
- Kita juga telah menelaah layanan Amazon Redshift yang berguna untuk semua kebutuhan *data warehouse* (gudang data).
- Guna membantu proses migrasi database yang Anda miliki, AWS menawarkan Amazon Database Migration Service (Amazon DMS).
- Kita telah meninjau seputar layanan penyimpanan lainnya, seperti Amazon DocumentDB, Amazon Neptune, Amazon QLDB, dan Amazon Managed Blockchain.
- Terakhir, kita juga telah mengulas tentang bagaimana solusi *caching* menggunakan Amazon ElastiCache dan Amazon DynamoDB Accelerator.

Ada berbagai pilihan layanan untuk menyimpan berbagai jenis data di AWS. Dengan sampainya kita di modul ini, ini tandanya Anda telah paham akan layanan apa yang tepat untuk menyimpan setiap jenis data. Bukan begitu?

Studi kita belum usai di sini. Jaga spirit Anda untuk sampai di garis finish ya. *Bon courage!*

Materi Pendukung

Silakan kunjungi tautan berikut untuk mempelajari lebih lanjut tentang konsep yang kita telah pelajari di modul ini:

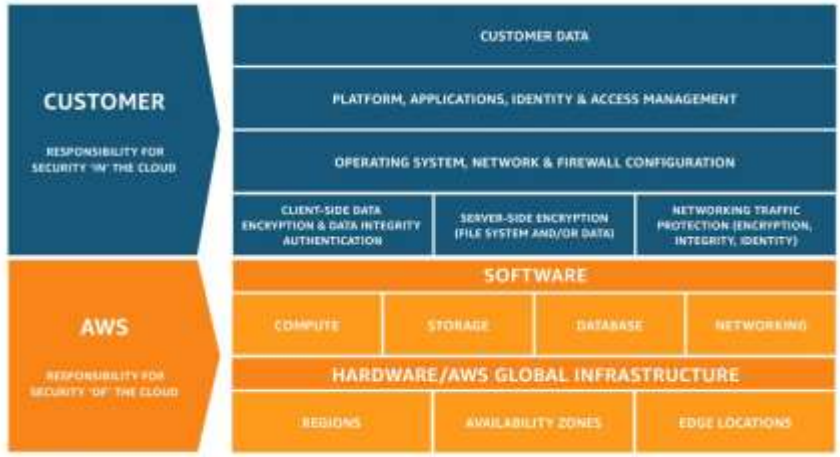
- Cloud Storage on AWS
- AWS Storage Blog
- Hands-On Tutorials: Storage

- AWS Customer Stories: Storage
- AWS Database Migration Service
- Databases on AWS
- Category Deep Dive: Databases
- AWS Database Blog
- AWS Customer Stories: Databases

MODUL 5

Pengenalan ke Keamanan

Di modul ini kita akan menyelami materi AWS semakin dalam. Kita akan belajar tentang pembagian kontrol terhadap lingkungan AWS platform, melalui konsep *shared responsibility model* alias model tanggung jawab bersama. Silakan amati gambar di bawah ini.



Diambil dari Model Tanggung Jawab Bersama AWS.
Ada dua hal utama yang harus Anda perhatikan di shared responsibility model ini, yaitu:

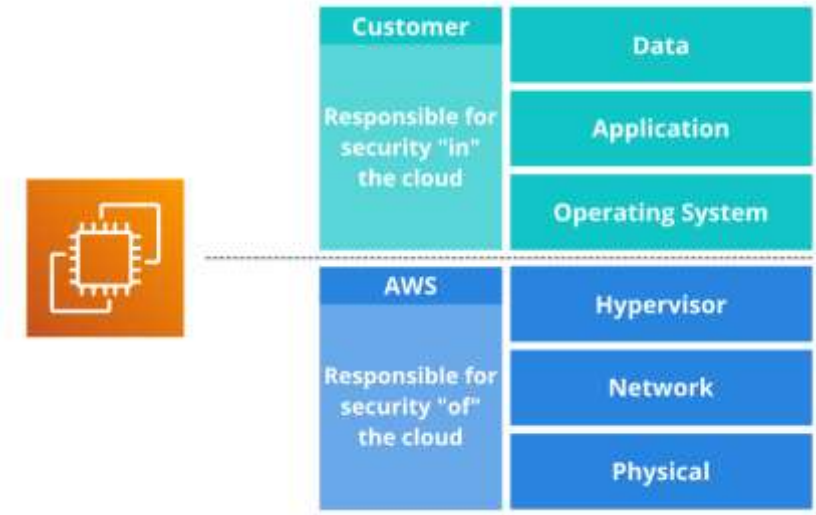
- **AWS** mengontrol *security of the cloud* (keamanan dari cloud).
- **Pelanggan** mengontrol *security in the cloud* (keamanan di cloud).

AWS bertanggung jawab untuk mengontrol data center, keamanan setiap layanan, dan lain sebagainya--dapat Anda lihat di gambar tersebut.
Lalu, bagian pelanggan adalah mengamankan beban kerja yang mereka jalankan di cloud. Ini merupakan bagian tanggung jawab yang AWS bagikan dengan pelanggan guna memastikan keamanan di cloud.
Kita akan membahas materi ini sekaligus menguak berbagai layanan, mekanisme, dan fitur keamanan lain di AWS secara lebih detail di modul berikutnya. Jadi, sudah siapkah Anda? *Stay tuned!*

Shared Responsibility Model

Sampai tahap ini, kita telah belajar tentang berbagai sumber daya yang dapat Anda buat di AWS Cloud, termasuk Amazon EC2 instance, Amazon S3 bucket, dan Amazon RDS database.
Nah, karena modul ini berbicara tentang keamanan, mari memulai materi kali ini dengan sebuah pertanyaan. Siapakah yang bertanggung jawab atas keamanan?
A. Anda sebagai pelanggan
B. AWS
Jawaban yang tepat adalah: Keduanya. Baik Anda maupun AWS bertanggung jawab untuk memastikan lingkungan cloud Anda aman.

Wah! Mungkin jika ada seorang pakar keamanan yang membaca tulisan ini, ia akan menggelengkan kepala dan berkata, “Tunggu! Dua entitas yang berbeda tidak boleh bertanggung jawab atas objek yang sama. Itu tak akan aman!”
Yup! AWS setuju dengan itu. Tetapi, ketahuilah! AWS tidak melihat lingkungan cloud Anda sebagai satu objek, melainkan kumpulan banyak bagian yang saling membangun. AWS bertanggung jawab 100% atas keamanan sebagian objek dan Anda bertanggung jawab untuk bagian lainnya. Inilah yang dikenal sebagai *shared responsibility model* atau model tanggung jawab bersama.
Bingung? Oke, anggaplah model ini sebagai pembagian tanggung jawab antara Anda--sang pemilik rumah--dan arsitek--yang membangun rumah. Arsitek (AWS) bertanggung jawab untuk memastikan rumah Anda dibangun dengan kokoh, sementara Anda (pelanggan AWS) bertanggung jawab untuk mengamankan seisi rumah dan mengunci pintu dengan benar. Mudah, ‘kan?
Begitu juga di AWS. Sebagai contoh, mari kita kupas setiap bagian Amazon EC2 berdasarkan shared responsibility model.



Amazon EC2 berjalan di data center AWS yang sangat aman. Ia memiliki jaringan dan hypervisor yang mendukung instance di atasnya beserta sistem operasinya. Di atas sistem operasi, Anda bisa menjalankan aplikasi dan mengelola data.
Bahkan tidak hanya Amazon EC2, setiap layanan AWS memiliki *layering* yang dibangun di atas satu sama lain. AWS 100% bertanggung jawab untuk bagian tertentu dan Anda bertanggung jawab untuk bagian lainnya.
Oke, supaya lebih jelas, mari kita uraikan setiap bagiannya:

- **Physical**
  Bagian ini terdiri dari berbagai komponen keamanan fisik, seperti gedung, sumber daya listrik, instalasi jaringan, sistem pendingin, penjagaan keamanan, dan lain sebagainya. Ini semua adalah tanggung jawab AWS.

- **Network & Hypervisor**
  Kita tidak akan membahas secara mendalam tentang bagaimana area ini diamankan. Tetapi pada dasarnya, AWS telah mempersiapkan teknologi

tersebut dan membuatnya menjadi lebih cepat, lebih baik, lebih kuat, dan tahan kerusakan.

AWS memiliki banyak auditor pihak ketiga yang memperhatikan dan mengawasi bagaimana infrastruktur AWS dibangun. Dan sehubungan dengan aspek tersebut, AWS dapat memberi Anda dokumentasi sesuai kebutuhan untuk struktur *security compliance* (kepatuhan keamanan). Ini juga adalah tanggung jawab AWS.

Oke, kita berhenti sejenak. Di sinilah garis pemisahnya. Sebelumnya kita telah menilik bagian yang menjadi tanggung jawab AWS, sekarang mari kita telaah bagian yang menjadi tanggung jawab Anda sebagai pelanggan.

- **Operating System**
  Dengan Amazon EC2, Anda bisa memilih sistem operasi yang ingin dijalankan. Ingat! AWS tidak memiliki akses sama sekali ke sistem Anda. Hanya Anda yang memiliki akses untuk masuk ke sistem operasi yang Anda jalankan.

  Lebih lanjut, Anda juga bertanggung jawab untuk melakukan *patching* (memperbaiki masalah dengan memperbarui program) terhadap sistem operasi tersebut. Jika ditemukan beberapa kerentanan baru di versi Windows, Anda yang perlu mencari solusinya dengan menerapkan patch terbaru.

  Ini adalah hal yang sangat bagus untuk keamanan. Tak akan ada yang dapat men-*deploy* (menerapkan) sesuatu sehingga mengganggu atau merusak sistem Anda.

- **Application**
  Di atas sistem operasi, Anda dapat menjalankan aplikasi apa pun yang diinginkan. Anda bertanggung jawab 100% untuk mengelolanya.

- **Data**
  Data adalah bagian terpenting dan sepenuhnya menjadi ranah Anda untuk mengontrolnya. Anda bisa membuat data dapat diakses oleh semua orang, beberapa orang, satu orang dengan kondisi tertentu, atau bahkan benar-benar menguncinya sehingga tidak ada yang dapat mengakses data tersebut. Plus, Anda juga dapat melakukan enkripsi pada data tersebut.

Shared responsibility model berguna untuk memastikan, baik AWS ataupun Anda--sebagai pelanggan--memahami tugasnya masing-masing dengan tepat. Pada dasarnya, AWS bertanggung jawab atas *security of the cloud* (keamanan dari cloud) dan Anda bertanggung jawab atas *security in the cloud* (keamanan di cloud).

Perizinan dan Hak Akses Pengguna

Pada skenario kedai kopi kita, setiap pegawai memiliki identitas dan akses sesuai perannnya masing-masing. Misalnya kita ambil contoh seorang kasir dan petugas gudang seperti berikut:

- Seorang kasir bertugas untuk menerima pesanan sehingga ia memiliki akses ke mesin kasir.
- Petugas gudang bertanggung jawab untuk memeriksa inventaris sehingga ia memiliki akses ke komputer gudang.

Mereka memiliki dua akses login dan dua set permission (izin) yang berbeda. Sehingga seorang kasir tidak akan diizinkan masuk ke sistem inventaris dan begitu juga sebaliknya.
Nah, mekanisme seperti itu juga bisa Anda implementasikan di AWS. Saat pertama kali membuat akun, Anda memulai dengan identitas sebagai AWS account root user atau bisa disederhanakan menjadi root user.
Root user adalah pemilik akun AWS. Ia memiliki permission untuk mengakses dan mengontrol seluruh sumber daya apa pun dalam akun tersebut, seperti menjalankan database, membuat EC2 instance, layanan blockchain, dan lain-lain.
Mudahnya, anggap saja root user sebagai pemilik kedai kopi. Ia bisa datang ke kedai dan melakukan apa pun, seperti mengoperasikan mesin kasir, menggunakan komputer gudang, atau hal-hal lainnya. Ia tidak akan mengalami pembatasan.
Nah, karena root user ini sangat berkuasa, AWS menyarankan Anda untuk mengaktifkan *multi-factor authentication* (MFA) guna memastikan agar akun tersebut aman. Apa itu MFA?
Begini. Pernahkah Anda login ke suatu website yang tak hanya meminta email dan password, melainkan juga melakukan verifikasi dua langkah dengan mengirimkan kode acak ke ponsel Anda? Nah, itulah contoh MFA. Ia berguna untuk memberikan lapisan keamanan tambahan untuk akun AWS Anda.
Tetapi, walaupun sudah mengaktifkan MFA, tentu Anda tak ingin memberikan akses root user ini ke semua pegawai di kedai kopi tersebut.
Masih ingat persoalan kita di awal? Kita tak ingin seorang kasir dapat mengakses komputer gudang. Lantas bagaimana solusinya?
Tenang, AWS memungkinkan Anda untuk dapat mengontrol akses secara terperinci dengan menggunakan layanan AWS Identity and Access Management (AWS IAM). Penasaran, apa saja fitur yang ditawarkan oleh AWS IAM? Mari kita bahas di materi berikutnya!

AWS Identity and Access Management (AWS IAM)
AWS Identity and Access Management (AWS IAM) dapat membantu Anda untuk mengelola akses ke layanan dan sumber daya AWS dengan aman.
IAM memberi Anda fleksibilitas untuk mengonfigurasi akses berdasarkan kebutuhan operasional dan keamanan yang spesifik. Di modul ini, kita akan membahas fitur-fitur IAM, seperti:

- IAM users

- IAM policies
- IAM groups
- IAM roles

Penasaran seperti apa? Mari kita bahas masing-masing fiturnya.

IAM Users

Di AWS Identity and Access Management (AWS IAM) Anda dapat membuat IAM users. Ia mewakili orang (personal) yang berinteraksi dengan layanan dan sumber daya AWS.

IAM users secara default belum memiliki permission sama sekali. Ia tidak bisa masuk ke akun AWS, meluncurkan EC2 instance, atau bahkan membuat S3 bucket. Intinya, secara default semua tindakan yang dilakukan oleh IAM users akan ditolak.

Jika ingin membuat IAM users bisa melakukan sesuatu, maka Anda harus memberikan permission secara eksplisit. Lalu, bagaimana cara memberikan atau menolak permission? Jawabannya, Anda bisa mengaitkan IAM policies ke IAM users.

IAM Policies

IAM policies adalah dokumen JSON yang mengizinkan atau menolak aktivitas tertentu terhadap layanan dan sumber daya AWS.

Tunggu, apa itu JSON? Sederhananya, JSON (JavaScript Object Notation) adalah format dokumen pertukaran data yang mudah dimengerti, baik oleh manusia maupun mesin.

Oke, kembali ke topik. Anda dapat menggunakan IAM policies untuk mengatur akses user ke sumber daya AWS. Misalnya, untuk mengizinkan user untuk mengakses beberapa atau spesifik salah satu Amazon S3 bucket dalam akun AWS Anda. Bingung? Tenang, mari kita lihat contoh singkat berikut:

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListObject",
    "Resource": "arn:aws:s3:::Example-Bucket"
  }
}
```

Mungkin sekarang Anda sedang bergumam dan berkata, “Apa maksud dari pernyataan-pernyataan tersebut?” Oke, mari kita bedah.

- Pada bagian “Effect”, Anda hanya bisa mengisinya dengan dua opsi: **Allow** (izinkan) atau **Deny** (tolak). Dalam kasus ini, kita memberikan izin kepada user untuk melakukan sesuatu.

- Untuk “Action”, Anda dapat mengisinya dengan panggilan API apa pun. Di sini kita menuliskan **s3:ListObject**. Artinya, user dapat mengetahui objek-objek apa saja yang berada di S3 bucket tertentu.
- Untuk bagian “Resource”, Anda bisa mengisinya dengan alamat sumber daya yang dimaksud. Di pernyataan tersebut kita bisa mengisinya dengan **arn:aws:s3:::EXAMPLE-BUCKET**, yaitu alamat ID unik dari S3 bucket tertentu.

Jadi, jika Anda melampirkan IAM policies tersebut ke IAM users, maka user tersebut dapat melihat daftar seluruh objek yang ada pada bucket yang bernama “EXAMPLE-BUCKET”.

Ingat! Saat memberikan permission, pastikan Anda mengikuti *“principle of least privilege”*. Maksudnya, berikanlah akses sesuai dengan kebutuhan saat itu saja.

Misalnya, jika seorang user hanya memerlukan akses ke bucket tertentu, maka berikanlah akses hanya untuk bucket tersebut di IAM policies, jangan ke semua bucket.

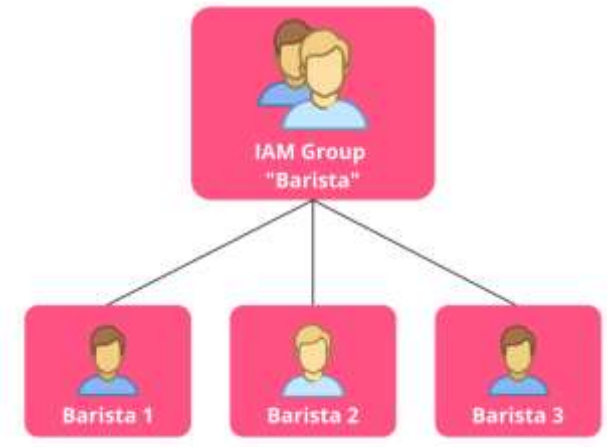
IAM Groups

Salah satu cara yang dapat mempermudah pengelolaan user dan permission adalah dengan mengelompokkannya ke dalam IAM groups.

IAM groups adalah grup/kelompok yang berisi kumpulan dari user. Menariknya, Anda bisa melampirkan policy ke group sehingga semua user yang berada di group tersebut akan memiliki permission yang sama.

Mari kita ambil contoh kedai kopi. Katakanlah Anda memiliki banyak pegawai barista baru dan ingin memberikan permission kepada mereka. Nah, daripada memberikannya satu per satu, Anda bisa melakukan hal berikut:

- Buat IAM groups bernama “Barista”.
- Tambahkan semua user barista baru ke dalam group.
- Lampirkan permission ke group tersebut.



Oke, sampai sini, kita sudah mengenal beberapa hal: IAM users, IAM groups, dan IAM policies. Tunggu, masih ada satu identitas utama lainnya, yaitu IAM roles.

IAM Roles

Untuk memahami apa itu IAM roles, mari kita analogikan dengan kedai kopi. Katakanlah Anda memiliki beberapa kasir. Seperti yang kita tahu, tak setiap saat keadaan kedai itu ramai pengunjung, pada waktu-waktu tertentu justru cenderung sepi.

Oleh karena itu, saat kedai sedang sepi, Anda ingin menugaskan beberapa kasir tersebut untuk melakukan pekerjaan yang berbeda, seperti bersih-bersih, mengecek inventaris, atau menyambut pelanggan yang datang. Namun ini hanya sementara, mereka harus kembali ke mesin kasir saat kedai kembali ramai.

Nah, Anda--sebagai pemilik kedai kopi--memiliki wewenang untuk memberikan peran sementara ini kepada beberapa kasir tersebut. Hal seperti ini dapat Anda implementasikan di AWS dengan IAM roles.

IAM roles memiliki permission yang dapat mengizinkan tindakan tertentu yang dibutuhkan secara temporer atau sementara. Role ini juga sebenarnya mirip dengan user, bedanya, ia tak memiliki *credential* (username dan password).

IAM roles dapat Anda gunakan untuk memberikan akses sementara ke beberapa hal, seperti sumber daya AWS, user, eksternal user, aplikasi, bahkan layanan AWS lainnya.

Ketika sebuah identitas menggunakan IAM roles, identitas tersebut menanggalkan semua permission sebelumnya yang dimiliki dan mengambil permission dari role tersebut.

Tak hanya itu, Anda dapat memfederasikan/menggabungkan eksternal user ke akun Anda. Maksudnya, daripada terus membuat IAM users untuk setiap orang di organisasi, Anda dapat menggunakan *regular corporate credential* (kredensial perusahaan reguler) untuk login ke AWS dengan memetakan identitas perusahaan ke IAM roles.

AWS Organizations

Saat Anda terjun pertama kali ke AWS Cloud, kemungkinan besar Anda akan memulai dengan satu akun AWS, kebanyakan orang pun akan seperti itu. Tetapi, seiring dengan pertumbuhan bisnis atau perjalanan cloud, Anda perlu memisahkan tugas dengan akun yang berbeda.

Misalnya begini. Katakanlah Anda memiliki beberapa tim yang menjalankan bisnis kedai kopi. Anda ingin

- tim developer dapat memiliki akses ke sumber daya pengembangan;
- tim akuntansi bisa mengakses informasi penagihan; atau bahkan
- memisahkan tim bisnis agar mereka dapat bereksperimen dengan layanan AWS tanpa mempengaruhi satu sama lain.

Semakin banyak tugas yang dilakukan setiap tim, maka lama-kelamaan akun AWS Anda akan makin kusut karena tak terkelola dengan baik.

Dengan kondisi tersebut, Anda perlu mengenal layanan yang satu ini, yaitu AWS Organizations. Sederhananya, ia adalah lokasi sentral yang dapat mengelola beberapa akun AWS. Dengannya, Anda dapat mengelola biaya, kontrol

akses, *compliance* (kepatuhan), keamanan, dan berbagi sumber daya dengan seluruh akun-akun AWS.
Saat Anda membuat organisasi, AWS Organizations secara otomatis membuat *root* (wadah induk yang terdiri dari OU--nanti kita bahas--dan akun AWS di organisasi Anda).
Lalu, apa saja fitur-fitur yang ditawarkan oleh AWS Organizations? Oke, mari kita uraikan.

1. **Manajemen terpusat**
   AWS Organizations dapat menjadi alat manajemen terpusat dari semua akun AWS Anda. Misal jika Anda memiliki beberapa akun (A, B, C, D, E), maka Anda dapat menggabungkannya menjadi sebuah organisasi sehingga memungkinkan akun terkelola secara terpusat.

2. **Consolidated billing (Tagihan terkonsolidasi)**
   Anda dapat menggunakan akun utama dari organisasi untuk menggabungkan dan mengatur pembayaran biaya penggunaan semua akun anggota. Bahkan, keuntungan lain dari consolidated billing adalah diskon massal. Kita akan mempelajari secara detail tentang consolidated billing ini di modul yang akan datang.

3. **Pengelompokan hierarki akun**
   Anda dapat mengimplementasikan fitur ini untuk memenuhi kebutuhan keamanan, *compliance*, atau anggaran. Kelompokkan akun ke dalam organizational unit (OU) untuk mempermudah pengelolaan akun-akun yang memiliki tujuan serupa atau kepentingan persyaratan keamanan.

   Saat Anda menerapkan *policy* (kebijakan) ke OU, semua akun otomatis mewarisi permission yang ada di policy tersebut. Fitur ini juga memudahkan Anda untuk mengisolasi beban kerja atau aplikasi yang memiliki persyaratan keamanan tertentu.

   Misalnya, jika Anda memiliki akun yang hanya dikhususkan untuk mengakses layanan AWS tertentu, maka Anda dapat memasukkannya ke dalam suatu OU. Kemudian lampirkan policy yang mengatur akses ke layanan AWS tersebut.

4. **Kontrol atas layanan AWS dan tindakan API**
   Dengan AWS Organizations, Anda bisa mengontrol layanan AWS dan layanan API yang dapat diakses oleh setiap akun administrator dari akun utama organisasi.

   Anda juga dapat menggunakan *service control policies* (SCP) untuk menentukan *permission* alias izin maksimum untuk akun anggota di organisasi. Maksudnya, Anda bisa membatasi layanan AWS, sumber daya, dan layanan

API individual yang mana dapat diakses oleh *user* dan *role* di setiap akun anggota.

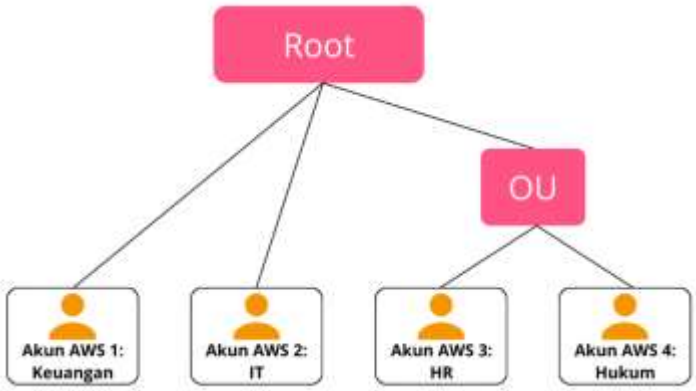
Studi Kasus: AWS Organizations

Katakanlah Anda memiliki bisnis dengan akun AWS terpisah untuk setiap departemen: Keuangan, IT, HR (Human Resource/Sumber Daya Manusia), dan Hukum. Anda memutuskan untuk menggabungkan akun ini ke dalam satu organisasi sehingga dapat dikelola dari satu tempat.

Tentu, kebutuhan semacam ini dapat diwujudkan dengan AWS Organizations. Dalam mendesain organisasi, Anda mempertimbangkan kebutuhan bisnis, keamanan, dan peraturan dari setiap departemen. Informasi ini Anda gunakan untuk memutuskan departemen mana yang akan dikelompokkan ke dalam sebuah OU.

Karena departemen Keuangan dan IT memiliki persyaratan yang tidak tumpang tindih dengan departemen lain, Anda memutuskan untuk memasukkannya ke dalam organisasi untuk berbagai keuntungan seperti *consolidated billing* dan tidak menempatkannya ke dalam OU mana pun.

Nah selanjutnya, karena departemen HR dan Hukum perlu mengakses layanan dan sumber daya AWS yang sama, Anda pun menempatkannya ke dalam satu OU.



Dengan menempatkan akun departemen HR dan Hukum ke dalam OU yang sama, Anda dapat melampirkan *policy* yang berlaku untuk keduanya. Selain itu, Anda juga dapat lebih mudah memberikan akses ke layanan dan sumber daya yang dibutuhkan. Walaupun telah menempatkan akun-akun tersebut ke dalam satu OU, Anda tetap dapat memberikan akses untuk *user*, *group*, dan *role* melalui AWS IAM.

Compliance (Kepatuhan)

Untuk mengawali modul, mari kita ungkap suatu fakta menarik. Tahukah Anda? Di setiap industri ada standar tertentu yang perlu ditegakkan. Anda akan diaudit atau diperiksa untuk memastikan bahwa standar tersebut telah dipenuhi.

Misalnya di kedai kopi. Suatu saat akan ada inspektur kesehatan yang datang untuk memeriksa apakah semuanya sesuai dengan regulasi dan sanitasi. Selain itu, akan ada petugas pajak yang datang mengaudit untuk memastikan bahwa Anda telah menjalankan administrasi dengan benar dan mematuhi hukum. Agar bisa lulus audit dan pemeriksaan *compliance*, Anda perlu mengandalkan dokumentasi, catatan, dan inspeksi.

Nah, begitu juga saat menggunakan AWS. Anda harus memenuhi standar dan regulasi tertentu sesuai jenis aplikasi yang dijalankan. Misal:

- Anda harus mematuhi GDPR (General Data Protection Regulation) saat menjalankan software yang menangani data konsumen di Uni Eropa; atau
- Anda perlu merancang arsitektur yang dapat memenuhi persyaratan compliance HIPAA (Health Insurance Portability and Accountability Act) ketika menjalankan aplikasi perawatan kesehatan di Amerika Serikat.

Maka dari itu, tentu Anda memerlukan suatu layanan yang dapat mengumpulkan dokumen, menyimpan catatan, dan memeriksa lingkungan AWS, dengan tujuan untuk memastikan regulasi compliance yang Anda jalankan telah terpenuhi dengan baik.
Ketahuilah, AWS telah membangun infrastruktur dan jaringan data center sesuai dengan standar praktik terbaik industri untuk keamanan. Nah, Anda--sebagai pelanggan AWS--seharusnya ikut juga mewarisi semua praktik terbaik tersebut baik dari segi kebijakan, arsitektur, dan juga proses operasional di atas AWS platform.
AWS telah memenuhi daftar panjang dari program *compliance*--dapat Anda temukan secara online di halaman AWS Compliance Programs. Ini berarti, sebagian dari aspek compliance Anda telah terpenuhi. Dengan begitu, Anda pun dapat fokus untuk memenuhi kebutuhan compliance di sisi arsitektur yang Anda bangun.
Oke, hal berikutnya yang perlu Anda ketahui terkait compliance di lingkungan AWS adalah Region. Region yang Anda pilih dapat juga membantu memenuhi regulasi compliance. Ibaratnya, jika Anda hanya dapat menyimpan data secara legal di negara tertentu, maka pilihlah Region yang sesuai dengan tujuan agar bisnis operasional Anda tunduk kepada aturan yang berlaku di negara tertentu.
Catat! Anda memiliki kontrol penuh atas keseluruhan data yang tersimpan di AWS. Anda bisa menggunakan beberapa mekanisme enkripsi tambahan untuk menjaga agar data tetap aman. Selain itu, jika Anda memiliki standar khusus terkait penyimpanan data, maka Anda dapat membuat mekanisme sendiri di AWS atau menggunakan fitur yang tersedia pada layanan AWS.

AWS Artifact

Berbicara tentang compliance, tahukah Anda? AWS menyediakan *whitepaper* (panduan resmi AWS) dan dokumen yang dapat Anda gunakan untuk keperluan laporan compliance.
Bahkan, AWS dapat memberikan dokumentasi yang membuktikan dirinya mengikuti praktik terbaik untuk keamanan dan compliance. Wow, menarik ‘kan?
Nah, layanan yang dapat Anda gunakan untuk mengakses dokumentasi tersebut adalah AWS Artifact. Dengannya, Anda bisa mendapatkan akses on-demand ke laporan keamanan dan compliance AWS serta online agreements (perjanjian online) tertentu.
Berikut adalah beberapa laporan dan regulasi compliance yang dapat Anda temukan dalam AWS Artifact.

Diambil dari AWS Artifact.
AWS Artifact terdiri dari 2 bagian utama, yaitu AWS Artifact Agreements dan AWS Artifact Reports. Mari kita telaah:

- **AWS Artifact Agreements**
  Jika Anda perlu menandatangani perjanjian dengan AWS terkait penggunaan jenis informasi tertentu di seluruh layanan, Anda dapat melakukannya melalui AWS Artifact Agreements.

  Di AWS Artifact Agreements, Anda dapat meninjau, menerima, dan mengelola perjanjian, baik untuk akun individu maupun semua akun di AWS Organizations.

  AWS menyediakan berbagai jenis perjanjian untuk memenuhi kebutuhan pelanggan yang patuh pada peraturan tertentu, seperti Health Insurance Portability and Accountability Act (HIPAA)**.**

- **AWS Artifact Reports**
  Ketika Anda hendak membuat aplikasi dan membutuhkan informasi tentang tanggung jawab untuk mematuhi standar regulasi tertentu, Anda dapat mengakses AWS Artifact Reports.

  AWS Artifact Reports menyediakan laporan compliance dari auditor pihak ketiga. Auditor ini telah menguji dan memverifikasi bahwa AWS mematuhi berbagai standar dan regulasi keamanan global, regional, dan industri.

Customer Compliance Center
Customer Compliance Center menyediakan informasi yang dapat membantu Anda untuk mempelajari lebih lanjut tentang compliance.
Di sana, Anda dapat membaca beberapa contoh kasus yang berhubungan dengan compliance dari para pelanggan AWS untuk memberikan gambaran bagaimana perusahaan dalam *regulated industry* (industri teregulasi) menyelesaikan berbagai tantangan compliance, governance/tata kelola, dan audit.
Anda juga dapat mengakses whitepaper dan dokumentasi tentang:

- Jawaban untuk pertanyaan mengenai compliance.
- Tinjauan mengenai AWS Risk and Compliance.
- Checklist audit keamanan.

- Dan masih banyak lainnya.

Serangan Denial-of-Service

Oke, sekarang kita masuk ke modul yang serius, benar-benar serius, yakni seputar serangan yang dapat melumpuhkan infrastruktur Anda.

Tapi, untuk mempermudah penjelasan, mari kita kaitkan dengan skenario kedai kopi. Katakanlah kedai kopi Anda memiliki layanan pemesanan melalui telepon. Cara kerjanya, seorang kasir akan menulis pesanan dan memberikannya kepada barista. Setelah minuman tersaji, pelanggan bisa mengambilnya di kedai kopi.

Namun, anggaplah ada orang iseng yang menelepon beberapa kali untuk memesan kopi tetapi ia tak pernah mengambil minumannya. Karena terus-menerus menelepon, ia membuat kasir tak bisa menerima panggilan dari pelanggan lain.

Lalu, bagaimana cara mengatasi masalah ini? Solusi sederhananya adalah dengan memblokir nomor telepon tersebut. Nah, tindakan orang iseng tersebut mirip dengan serangan *denial-of-service*.

Serangan denial-of-service (DoS) adalah upaya yang dilakukan secara sengaja untuk membuat website atau aplikasi menjadi tidak bekerja dengan optimal bagi pengguna.

Salah satu contohnya adalah ketika penyerang membanjiri aplikasi Anda dengan traffic jaringan yang masif sehingga membuatnya kewalahan dan tak lagi dapat merespons permintaan pengguna.

Serangan Distributed Denial-of-Service

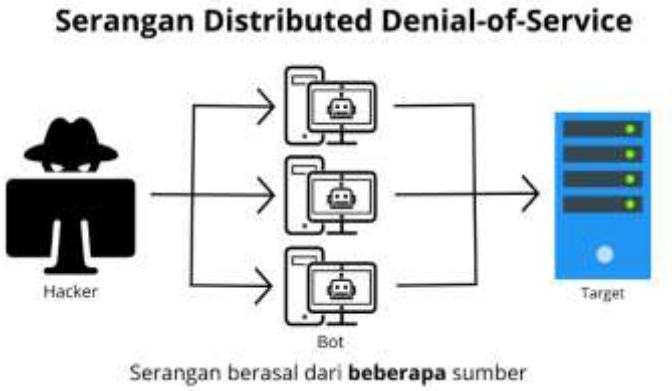
Distributed denial-of-service alias DDoS adalah salah satu serangan yang dapat menimpa infrastruktur atau aplikasi Anda. Serangan ini telah membuat banyak bisnis hancur.

Banyak tim keamanan di luar sana yang telah membangun rencananya yang kompleks untuk mencegah serangan ini. Tapi, apa sebenarnya DDoS itu? Bagaimana kita bisa bertahan melawannya?

Huh! Ini akan menjadi pembahasan yang panjang, bahkan bisa memakan waktu yang cukup lama untuk benar-benar memahami semuanya. Tapi tenang! Anda tak perlu tahu semuanya.

Pada modul kali ini, kita hanya akan membahas dasar-dasar bagaimana serangan DDoS terjadi dan bagaimana AWS dapat secara otomatis mempertahankan infrastruktur Anda darinya. Oke, mari kita mulai.

Guna mempermudah, seperti biasa, kita analogikan dengan kedai kopi dan menggunakan kasus yang sama seperti sebelumnya, yaitu orang iseng yang menelepon berulang kali.
Yup! Kita memang telah memblokir nomor tersebut. Tapi ternyata, orang ini meminta bantuan teman-temannya. Mereka terus-menerus menghubungi kedai kopi dengan nomor telepon yang berbeda. Tentu ini akan membuat pelanggan lain semakin kesulitan untuk menelepon kedai kopi Anda.
Nah, itulah konsep dari serangan DDoS. Berbeda dengan DoS yang hanya berasal dari satu sumber, serangan DDoS ini menggunakan beberapa sumber untuk melakukan serangan. Tujuannya untuk membuat aplikasi Anda kewalahan dan tak dapat beroperasi lagi.
Serangan ini bisa bisa berasal dari sekelompok orang atau bahkan individu. Cara kerjanya, penyerang menggunakan beberapa komputer yang terinfeksi (juga dikenal sebagai "bot") untuk mengirimkan traffic yang masif ke situs aplikasi Anda.



Oke, pembahasan terkait mekanisme DDoS ini akan sangat panjang. Maka dari itu, mari kita beralih dan menguraikan beberapa tipe dari serangan DDoS.

- **UDP flood**
  Seseorang dapat menggunakan jenis serangan ini untuk melakukan DDoS dengan mudah, salah satunya adalah memanfaatkan layanan cuaca nasional.

  Saat Anda mengirim permintaan informasi ke layanan cuaca nasional, ia akan mengirimkan telemetri cuaca, prakiraan, pembaruan, dan lain-lain dengan jumlah yang masif.

  Nah, di sini masalahnya. Penyerang akan melakukan permintaan seputar informasi cuaca, tapi ia mencantumkan alamat penerima yang palsu, yakni alamat infrastruktur Anda.

  Dengan demikian, layanan cuaca akan membanjiri server Anda dengan data berukuran megabyte yang dapat membuat sistem Anda kewalahan dalam memprosesnya. Hingga akhirnya, server Anda akan berhenti.

- **HTTP level attack**
  HTTP level attack merupakan serangan yang jauh lebih canggih. Penyerang akan terlihat seperti pelanggan normal yang mengakses aplikasi Anda dan melakukan permintaan umum, seperti pencarian produk yang rumit.

  Bedanya, penyerang melakukannya secara berulang kali dan terus-menerus sehingga server akan selalu memprosesnya. Ini mengakibatkan pelanggan yang sah menjadi tidak bisa mengakses aplikasi Anda.

- **Slowloris attack**
  Serangan ini bahkan jauh lebih mengerikan. Agar lebih mudah memahaminya, mari kita umpamakan.

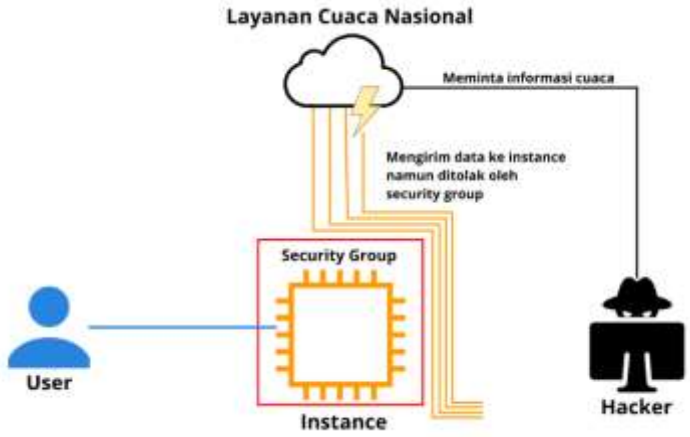
  Katakanlah ada seorang pelanggan yang memesan minuman di kedai kopi Anda. Pelanggan tersebut membutuhkan waktu hingga tujuh menit lamanya. Tentu ini akan membuat para pelanggan lain yang mengantre di belakangnya menjadi tak bisa memesan kopi. Nah, serangan Slowloris serupa dengan itu.

  Penyerang berpura-pura memiliki koneksi yang sangat lambat. Ini menyebabkan server Anda terus menunggu pelanggan a.k.a penyerang tersebut menyelesaikan permintaannya dan membuat pelanggan lain tak terlayani. Beberapa penyerang Slowloris bahkan dapat menghabiskan seluruh kapasitas server Anda hanya dengan sedikit usaha.

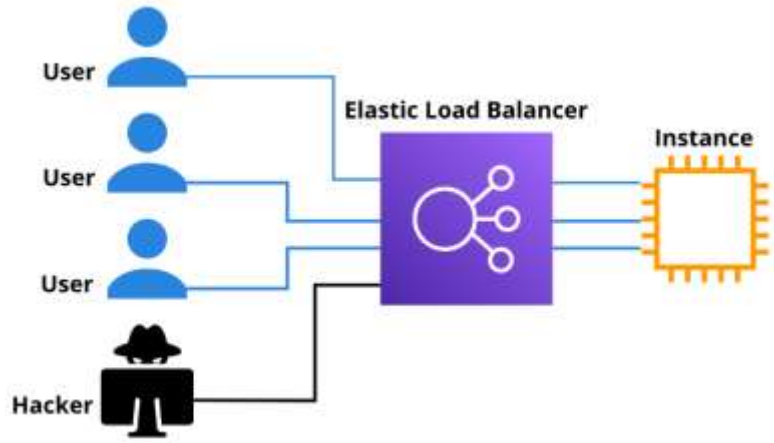
Oke. Sekarang saatnya kita hentikan serangan-serangan ini dengan solusi yang tepat. Sebetulnya, Anda sudah tahu caranya. Apa maksudnya?

Begini. Semua hal yang telah kita pelajari di kelas ini tak hanya berbicara tentang membangun arsitektur yang baik, melainkan juga membantu Anda menyelesaikan hampir semua jenis serangan DDoS, bahkan tanpa biaya tambahan. Mari kita kupas satu per satu.

Mulai dari serangan pertama, yakni UDP flood. Solusi untuk mencegah serangan ini adalah dengan menggunakan *security group*. Ia hanya dapat mengizinkan traffic permintaan yang tepat.



Tahukah Anda? Data laporan cuaca menggunakan protokol yang benar-benar berbeda dari pelanggan Anda. Artinya, security group bisa menolak permintaan tersebut jika memang tak ada di dalam daftar yang diizinkan.
Lalu, bagaimana untuk menghalau serangan Slowloris? Solusinya sederhana, Anda pun sudah mempelajarinya, yakni menggunakan Elastic Load Balancer (ELB) yang dapat mengarahkan traffic lalu lintas untuk Amazon EC2 instance.
Jadi, walaupun penyerang memiliki koneksi yang sangat lambat, pelanggan Anda yang sah tak perlu menunggunya hingga selesai, mereka tetap bisa mengakses server.



Serangan ini juga tak akan mengenai instance Anda karena sebelum diteruskan ke server, ELB akan menangani setiap permintaan hingga selesai terlebih dahulu, tak peduli ia memiliki koneksi yang cepat atau bahkan lambat sekali pun.
Ingat! ELB itu sangat kuat dan kapasitasnya dapat diskalakan. Ia juga berjalan di tingkat Region. Artinya, untuk bisa membanjiri ELB, Anda harus membanjiri keseluruhan AWS Regions. Bukannya mustahil, tetapi secara teoritis akan terlalu mahal bagi siapa pun yang melakukannya.
Nah, bagaimana untuk mengadang serangan yang paling tajam dan paling canggih di luar yang telah dijelaskan? Tenang, tenang! AWS juga menawarkan layanan pertahanan khusus yang disebut dengan AWS Shield.

AWS Shield
AWS Shield adalah layanan yang dapat melindungi aplikasi Anda dari serangan DDoS. Layanan ini memberikan dua tingkat perlindungan: Standard dan Advanced. Mari kita kupas keduanya.

- **AWS Shield Standard**
  AWS Shield Standard secara otomatis melindungi sumber daya AWS Anda dari jenis serangan DDoS yang paling umum tanpa biaya.

  Dengan menggunakan berbagai teknik analisis, AWS Shield Standard dapat mendeteksi dan memitigasi traffic berbahaya secara *real time* saat memasuki aplikasi Anda.

- **AWS Shield Advanced**
  AWS Shield Advanced adalah layanan berbayar yang menyediakan kemampuan untuk mendiagnostik, mendeteksi, dan memitigasi serangan DDoS yang canggih.

  AWS Shield Advanced terintegrasi dengan layanan lain seperti Amazon CloudFront, Amazon Route 53, dan Elastic Load Balancing.

  Selain itu, Anda juga dapat mengintegrasikan AWS Shield dengan AWS WAF. AWS WAF merupakan *web application firewall* untuk melindungi aplikasi web atau API Anda dari eksploitasi web umum yang dapat memengaruhi ketersediaan, mengganggu keamanan, atau memakai sumber daya secara berlebihan.

Layanan Keamanan Tambahan

Hari silih berganti dan kedai kopi Anda semakin ramai. Banyak pelanggan yang datang dan pergi. Karena hal ini, tentu Anda ingin meningkatkan keamanan di sana, bukan?

Salah satu hal yang harus Anda amankan adalah biji kopi. Anda harus memastikan mereka aman, baik pada saat di gudang penyimpanan atau ketika proses pengiriman antar toko. Intinya, Anda tak ingin satu pun pelanggan memiliki akses ke biji kopi. Cara sederhana untuk mengamankannya adalah dengan mengunci pintu saat Anda pergi di malam hari.

Nah, begitu juga dengan data. Anda perlu mengamankannya, baik saat keadaan *at rest* (diam) maupun *in-transit* (berpindah). Anda bisa melakukannya dengan enkripsi, yaitu mengamankan suatu pesan atau data yang hanya dapat diakses oleh pihak terotorisasi. Dengan melakukan enkripsi, pihak yang tak berwenang tidak akan bisa mengakses data Anda sama sekali.

Kalau Anda bingung, anggap saja enkripsi itu sebagai kunci dan pintu di kedai kopi. Jika Anda memiliki kuncinya, maka Anda dapat membuka pintu. Tetapi jika tidak, yup! Anda pasti sudah tahu jawabannya. Sederhana, ‘kan?

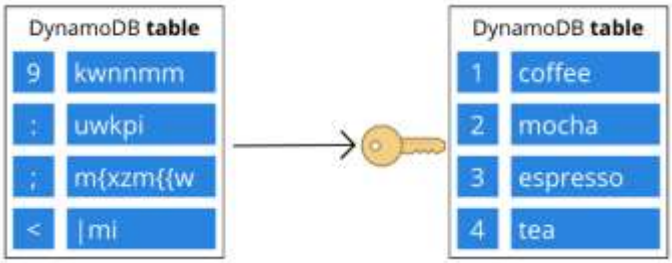
Di AWS, enkripsi hadir dalam dua varian: *at rest* (saat diam) dan *in-transit* (dalam perjalanan). Mari kita bedah.

- ***Encryption at rest***
  Pada jenis ini, proses enkripsi terjadi saat data Anda dalam keadaan tidak bergerak (tersimpan dan tidak berpindah). Contohnya adalah *server-side encryption at rest* (enkripsi at rest pada sisi server) yang secara bawaan telah aktif untuk semua data di tabel DynamoDB.

  Data yang tersimpan di sana akan terenkripsi alias berubah menjadi serangkaian kata yang tak terbaca. Ini berguna untuk mencegah data Anda diakses oleh pihak yang tidak berwenang.

  Enkripsi at rest pada DynamoDB juga terintegrasi dengan AWS Key

Management Service (AWS KMS) yang berguna mengelola *key*/kunci enkripsi untuk tabel Anda--nanti kita bahas. Kunci enkripsi ini berperan seperti kunci pintu, ingat? Tanpanya, Anda tak akan dapat mengakses data, jadi simpanlah dengan aman.



- ***Encryption in-transit***
  Proses enkripsi in-transit terjadi saat data Anda berpindah antara A dan B. A dan B ini bisa berupa apa pun, seperti layanan AWS dan klien yang mengakses layanan tersebut.

  Misalnya, Anda ingin menghubungkan Redshift instance dan SQL client. Untuk kasus tersebut, Anda bisa menggunakan koneksi SSL alias *secure sockets layer.* Dengan begitu, Anda dapat melakukan enkripsi data serta menggunakan sertifikat layanan untuk validasi dan otorisasi klien.

- Dengan demikian, data Anda akan terlindungi selama perjalanan antara Redshift dan klien. Ketahuilah! Fungsionalitas semacam ini tersedia di banyak layanan AWS lainnya seperti SQS, S3, RDS, dan banyak lagi.

AWS Key Management Service (AWS KMS)
Sebelumnya kita telah menyinggung sedikit tentang layanan ini, sekarang mari kita selami lebih dalam tentangnya.
AWS Key Management Service (AWS KMS) adalah layanan yang memungkinkan Anda untuk melakukan enkripsi menggunakan *cryptographic key* (kunci kriptografi). Kunci kriptografi adalah rangkaian angka acak yang digunakan untuk mengunci (mengenkripsi) dan membuka kunci (mendekripsi) data.
Dengan AWS KMS, Anda dapat mengontrol penggunaan kunci kriptografi di berbagai layanan ataupun di dalam aplikasi Anda.
Selain itu, Anda juga dapat memilih tingkat kontrol akses tertentu yang Anda perlukan untuk kunci tersebut. Misal:

- Anda bisa menentukan IAM users dan IAM roles mana yang dapat mengelola kunci; atau
- Anda dapat menonaktifkan kunci untuk sementara sehingga tidak ada yang bisa menggunakannya.

Kunci kriptografi yang Anda simpan di AWS KMS akan senantiasa aman dan terlindungi.

AWS Web Application Firewall (AWS WAF)

AWS Web Application Firewall alias AWS WAF memungkinkan Anda untuk dapat memantau request/permintaan jaringan yang masuk ke aplikasi web.

Masih ingatkah Anda tentang *network access control list* (network ACL) yang telah kita pelajari di modul sebelumnya? AWS WAF ini bekerja dengan cara yang mirip dengan network ACL, yaitu memblokir atau mengizinkan lalu lintas. Bedanya, ia menggunakan *web access control list* (web ACL) untuk melindungi sumber daya AWS Anda.

Berikut adalah contoh bagaimana Anda dapat menggunakan AWS WAF untuk mengizinkan dan memblokir request tertentu.

Anggaplah aplikasi Anda telah menerima request jaringan berbahaya dari beberapa alamat IP. Tentu, Anda ingin mencegah request ini, bukan? Tetapi, Anda juga ingin memastikan user sah masih dapat mengakses aplikasi.

Nah untuk masalah ini, Anda dapat mengonfigurasi web ACL yang dapat mengizinkan semua request kecuali dari alamat IP yang telah Anda tentukan.

Saat sebuah request hendak masuk ke aplikasi Anda, AWS WAF akan memeriksa daftar pengaturan yang telah Anda atur di web ACL.

Jika request tersebut bukan berasal dari salah satu alamat IP yang diblokir, maka ia diizinkan masuk ke aplikasi. Namun, jika sebaliknya, ia akan ditolak.

Amazon Inspector

Mari kita awali modul ini dengan sebuah analogi kedai kopi. Misalkan Anda sedang mengembangkan dan menguji aplikasi pemesanan baru. Anda ingin memastikan

rancangan aplikasi tersebut sesuai dengan praktik keamanan terbaik. Namun, Anda tidak memiliki banyak waktu untuk melakukan penilaian manual.
Nah, sekarang pertanyaannya, bagaimana cara menilai keamanan secara otomatis?
AWS memberikan solusi terbaik, yaitu dengan layanan Amazon Inspector. Ia dapat membantu Anda untuk melengkapi pemahaman kita untuk meningkatkan keamanan dan *compliance*/kepatuhan aplikasi dengan menjalankan penilaian keamanan secara otomatis terhadap infrastruktur Anda.
Amazon Inspector bekerja dengan cara memeriksa aplikasi terhadap kerentanan dan penyimpangan praktik terbaik keamanan. Contohnya seperti akses yang terbuka ke Amazon EC2 instance atau penginstalan versi software yang memiliki kerentanan keamanan.
Setelah melakukan penilaian, Amazon Inspector memberikan Anda daftar temuan keamanan yang diprioritaskan menurut tingkat keparahan. Daftar ini juga berisi deskripsi mendetail tentang setiap masalah keamanan dan rekomendasi cara memperbaikinya.
Namun ingat, AWS tidak menjamin bahwa dengan mengikuti rekomendasi tersebut dapat menyelesaikan setiap potensi masalah keamanan. Tentu Anda masih ingat, berdasarkan shared responsibility model, pelanggan AWS bertanggung jawab atas keamanan aplikasi, proses, dan tools yang berjalan di layanan AWS.

Amazon GuardDuty

Layanan keamanan lain nan tak kalah penting adalah Amazon GuardDuty. Ia adalah layanan yang menyediakan deteksi ancaman cerdas untuk infrastruktur dan sumber daya AWS Anda.
Layanan ini mengidentifikasi ancaman dengan senantiasa memantau aktivitas jaringan dan perilaku akun di dalam lingkungan AWS Anda.
Amazon GuardDuty menggunakan *integrated threat intelligence* (kecerdasan ancaman terintegrasi) seperti alamat IP berbahaya, deteksi anomali, dan machine learning untuk mengidentifikasi ancaman dengan lebih akurat.
Nah, saat GuardDuty mendeteksi suatu ancaman, Anda dapat meninjau temuan mendetail dari AWS Management Console. Temuan ini juga mencakup langkah-langkah rekomendasi yang bisa Anda tindak lanjuti.

Tunggu, masih ada bagian terbaiknya. Layanan ini berjalan secara independen dari layanan AWS yang lain. Jadi, ia tak akan memengaruhi kinerja, ketersediaan, dan beban kerja infrastruktur Anda.

Ikhtisar

Tak terasa ya sudah sejauh ini kita melangkah pada materi keamanan. Baiklah, sekarang saatnya mengurai apa yang telah kita pelajari.

- Pertama, AWS menyajikan *shared responsibility model* alias model tanggung jawab bersama. AWS bertanggung jawab atas keamanan dari cloud sementara Anda bertanggung jawab untuk keamanan di cloud.
- Kemudian, AWS IAM memungkinkan Anda untuk memiliki users, groups, roles, dan policies.
    - ***Users*** dapat Anda pakai untuk *login* atau masuk ke AWS dengan menggunakan nama pengguna dan kata sandi. Ia juga secara default tidak memiliki *permission* (izin) sama sekali.
    - ***Groups*** merupakan kumpulan dari beberapa pengguna.
    - ***Roles*** adalah identitas yang berguna untuk memberikan akses kredensial sementara dan permission untuk jangka waktu tertentu.
    - ***Policies*** berfungsi untuk memberikan permission ke sebuah identitas secara eksplisit, baik *allow* (mengizinkan) atau *deny* (menolak) suatu tindakan tertentu di AWS.
    - IAM juga menghadirkan ***identity federation*** (federasi identitas). Jika suatu perusahaan telah memiliki penyimpanan identitasnya sendiri, maka user di perusahaan tersebut dapat terkoneksi ke AWS menggunakan *role based access* (akses berbasis peran). Hal itu memungkinkan user melakukan sekali *login* untuk sistem perusahaan tersebut dan sekaligus juga lingkungan AWS.
    - Satu hal terakhir yang perlu diingat tentang IAM adalah ***multi-factor authentication*** (MFA). Pastikan Anda mengaktifkan MFA untuk setiap user, terutama *root user* yang memiliki semua permission secara default dan tak dapat direstriksi.
- Selanjutnya, kita juga telah membahas AWS Organizations. Saat menggunakan AWS, kemungkinan Anda akan memiliki beberapa akun. Biasanya, akun digunakan untuk mengisolasi beban kerja, lingkungan, tim, atau aplikasi. Nah, AWS Organizations dapat membantu Anda untuk mengelola beberapa akun secara hierarkis.
- Tak luput kita juga telah belajar mengenai *compliance* (kepatuhan). AWS menggunakan auditor pihak ketiga untuk membuktikan compliance-nya terhadap beragam program compliance. Anda dapat menggunakan:
    - AWS Compliance Center untuk menemukan informasi lebih lanjut tentang compliance.
    - AWS Artifact untuk mendapatkan akses ke dokumen compliance.

  Persyaratan compliance yang Anda miliki mungkin dapat bervariasi untuk setiap aplikasinya.

- Lalu, kita telah menelaah tentang serangan distributed denial-of-service (DDoS) dan cara menanganinya dengan menggunakan layanan seperti ELB, security group, AWS Shield, dan AWS WAF.
- Kemudian, kita juga telah menilik materi enkripsi. Di AWS, Anda sebagai pemilik data bertanggung jawab atas keamanannya. Itu berarti Anda perlu menerapkan enkripsi untuk data yang Anda miliki, baik *in-transit* (ketika dikirim) maupun *at rest* (saat disimpan).

Keamanan adalah prioritas utama AWS, dan akan terus demikian. Ada banyak pertimbangan saat menangani keamanan di AWS. Oleh sebab itu, pastikan Anda membaca dokumentasi tentang cara mengamankan sumber daya AWS Anda karena akan berbeda setiap layanannya.

Ingat! Gunakan *least privilege principle* (prinsip privilese paling rendah) saat memberikan *permission* (izin) kepada user dan role di IAM; enkripsi data di setiap lapisan; dan pastikan Anda menggunakan layanan AWS untuk melindungi lingkungan cloud Anda.

Materi Pendukung

Silakan review tautan berikut untuk mempelajari lebih lanjut tentang konsep yang telah kita bahas di modul ini:

- Security, Identity, and Compliance on AWS
- Whitepaper: Introduction to AWS Security
- Whitepaper: Amazon Web Services - Overview of Security Processes
- AWS Security Blog
- AWS Compliance
- AWS Customer Stories: Security, Identity, and Compliance

MODUL 6

Pengenalan ke Pemantauan dan Analitik

Ingat kembali skenario kedai kopi kita. Sebagai pemilik, tentu Anda ingin mengetahui apa yang terjadi di sana sepanjang hari guna memastikan semuanya berjalan lancar.

Tetapi, Anda juga tak ingin hanya berdiri dan diam di sana seharian. Alangkah lebih baiknya Anda bisa mengecek kembali kinerja toko ketika sudah menjelang petang dengan mengajukan beberapa pertanyaan, seperti:

- Berapa banyak kopi yang terjual?
- Berapa lama rata-rata waktu tunggu seseorang saat memesan kopi?
- Apakah persediaan hari ini habis?

Bahkan, akan sangat membantu jika Anda bisa mendapat notifikasi ketika waktu tunggu pemesanan terlalu lama. Sehingga Anda dapat terjun langsung ke sana atau menyuruh pegawai lain membantu pekerjaan tersebut.

Setiap bisnis--termasuk kedai kopi kita--dapat menggunakan metrik untuk mengukur seberapa baik sistem dan proses yang berjalan.

Nah, proses mengamati sistem; mengumpulkan metrik; dan mengevaluasinya dari waktu ke waktu untuk membuat keputusan atau mengambil tindakan, disebut dengan *monitoring* atau pemantauan.

Tahukah Anda pentingnya sebuah pemantauan? Kegiatan ini perlu Anda lakukan ketika menggunakan layanan berbasis cloud untuk memastikan sumber daya AWS berjalan sesuai dengan harapan.

Misalnya, Anda dapat melakukan proses *scaling* (penyesuaian kapasitas) secara otomatis jika sebuah EC2 instance dalam keadaan *over-utilized* (digunakan secara berlebihan). Atau, Anda juga bisa menerima pemberitahuan jika suatu aplikasi mulai mengirimkan respons kesalahan dengan kecepatan tinggi.

Oke, untuk beberapa modul berikutnya kita akan membahas berbagai layanan yang dapat membantu Anda memonitor lingkungan AWS. Pemantauan dapat digunakan untuk mengukur performa dari sistem, memberi peringatan jika ada yang tak beres, bahkan dapat membantu proses *debugging* (identifikasi dan perbaikan eror).

Amazon CloudWatch

Mari kita awali modul ini dengan mengingat kembali skenario kedai kopi yang telah beroperasi cukup lama. Tapi, tahukah Anda? Ada satu masalah baru yang muncul di sana.

Karena kita terus-menerus menjalankan mesin kopi; menggunakan mug; membuka dan menutup lemari es, tentu akan lebih baik jika muncul suatu pemberitahuan.

Notifikasi tersebut sebaiknya datang pada saat ada sesuatu yang memerlukan tindakan tertentu, seperti mesin kopi yang harus dibersihkan atau diperbaiki.

Intinya, Anda sebagai pemilik kedai kopi memerlukan visibilitas terhadap status dari sistem, seperti:

- Apakah semuanya berjalan dengan baik?
- Apakah pelanggan Anda semakin senang atau malah sebaliknya?
- Apakah Anda sering mengirimkan minuman yang salah kepada pelanggan?

Selain beberapa pertanyaan di atas, masih banyak lagi pertanyaan lainnya yang dapat menunjukkan keberhasilan operasional Anda.

Ide yang sama pun berlaku untuk sistem yang dibangun di AWS. Anda perlu satu cara yang dapat memantau kesehatan dan pengoperasian aplikasi. Tak perlu repot-repot untuk membangun platform pemantauan sendiri karena AWS telah menyediakannya untuk Anda.

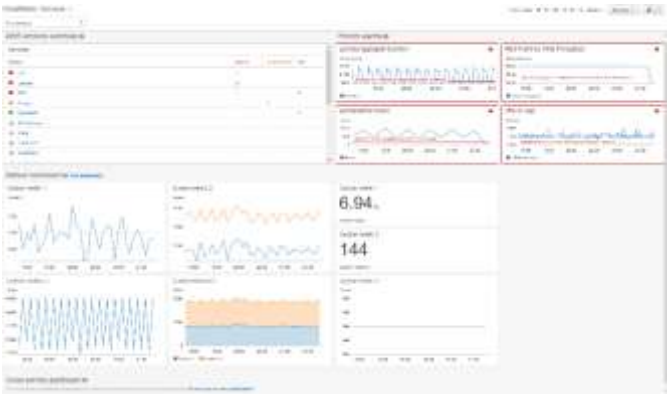
Sambutlah, Amazon CloudWatch. Ia dapat memantau infrastruktur dan aplikasi yang Anda jalankan di AWS secara *real time*. Layanan ini bekerja dengan cara melacak dan memantau metrik. Metrik adalah variabel yang terikat dengan sumber daya Anda, seperti penggunaan CPU dari EC2 instance.

Oke, ini akan menarik. Sesuai dengan persoalan kita di awal, Anda ingin memiliki suatu sistem yang dapat memberi tahu jika mesin kopi perlu dibersihkan, misalnya setiap kali selesai membuat 100 espreso.

Nah, hal ini bisa terlaksana dengan hadirnya Amazon CloudWatch alarm. Anda dapat membuat suatu metrik khusus dan menetapkan ambang batasnya adalah 100. Pada saat jumlah espreso yang dibuat telah mencapai angka tersebut, ia akan memperingatkan Anda untuk membersihkan mesin. Sederhana, bukan?

Oh, tidak hanya itu! Bahkan CloudWatch alarm dapat terintegrasi dengan layanan Amazon SNS. Jadi, Anda dapat mengirimkan SMS ke pegawai di kedai kopi supaya mereka membersihkan mesinnya.

Dengan Amazon CloudWatch, Anda dapat membuat alarm sendiri untuk metrik dari semua jenis sumber daya di AWS. Nah, bagaimana jika kita ingin menggabungkan semua metrik tersebut dalam satu panel?



Diambil dari Amazon CloudWatch: Getting Started with Amazon CloudWatch.

Tentu kita bisa menggunakan fitur CloudWatch dashboard. Ia adalah panel yang mencantumkan metrik hampir secara real time. Dengannya, Anda dapat memantau penggunaan CPU dari Amazon EC2 instance, jumlah total permintaan yang dibuat ke Amazon S3 bucket, dan masih banyak lainnya. Sehingga, Anda dapat memonitornya secara proaktif.

Dashboard ini akan me-*refresh* secara otomatis setiap kali terbuka sehingga akan selalu menunjukkan tampilan terkini dari sumber daya Anda.

Sekarang mungkin Anda akan bertanya-tanya, “Apa keuntungan menggunakan Amazon CloudWatch ini?” Yuk mari kita jabarkan.

- **Akses ke semua metrik dari satu lokasi**
  Anda dapat mengumpulkan metrik dan log dari semua sumber daya yang berjalan di AWS bahkan server yang berada di on-premise.

- **Visibilitas ke seluruh aplikasi, infrastruktur, dan layanan**
  Dengan visibilitas ke seluruh sistem, Anda dapat mengorelasikan bahkan memvisualisasikan metrik dan log untuk menunjukkan sekaligus menyelesaikan masalah dengan cepat.

- **Mengurangi waktu MTTR dan meningkatkan TCO**
  MTTR (mean time to resolution) adalah rata-rata waktu untuk menyelesaikan suatu masalah, sementara TCO (total cost of ownership) adalah biaya kepemilikan.

  Implementasi di kedai kopinya adalah, jika MTTR untuk jam pembersihan mesin lebih pendek, maka Anda dapat menghemat TCO. Dengan kata lain, Anda tak perlu repot-repot menghabiskan waktu untuk membuat sistem analitik sendiri. AWS telah menyediakan Amazon CloudWatch sehingga Anda dapat fokus pada peningkatan nilai bisnis.

- **Mengoptimalkan aplikasi dan sumber daya operasional**
  Anda dapat menggabungkan metrik dari seluruh EC2 instance untuk memperoleh wawasan akan operasional dan penggunaannya.

Itu dia materi kita tentang Amazon CloudWatch. Berikutnya, kita punya materi yang tak akan kalah seru. Jadi, tunggu apalagi? Mari kita masuk ke materi selanjutnya!

AWS CloudTrail

Masih ingatkah Anda dengan contoh kasus kedai kopi kita sebelumnya? Tentu kita semua tahu bahwa di sana terdapat sebuah mesin kasir.

Tapi tahukah Anda? Mesin kasir adalah salah satu perangkat audit mandiri pertama di dunia. Walaupun demikian, Anda tetap harus memiliki prinsip: *percaya tapi verifikasi*.

Maksudnya begini. Pegawai kasir yang bekerja di kedai kopi tentunya adalah seseorang yang telah Anda percayai. Namun, mungkin Anda perlu memastikan bahwa uang tunai di laci kas telah sesuai dengan penjualan yang terjadi sebenarnya. Bagaimana caranya?

Anda bisa mewujudkan hal itu dengan memeriksa mesin kasir tersebut karena ia dapat mencatat dan menyusun menjadi tabel semua transaksi yang terjadi. Inilah yang disebut dengan mengaudit transaksi. Kemampuan semacam itu di bidang IT merupakan elemen yang penting dalam sebagian besar struktur *compliance* (kepatuhan).

Coba kita bandingkan antara data center on-premise dengan AWS. Di on-premise, jika seseorang membuat suatu perubahan konfigurasi pada server, kita tak bisa mengetahui siapa pelakunya karena memang tidak ada yang dapat mencatat aksi modifikasi tersebut. Lantas, bagaimana dengan AWS? *Eits*, tentu masalah tersebut tak akan terjadi karena semuanya terprogram.

Perkenalkan, AWS CloudTrail. Ia adalah layanan audit API yang komprehensif. Dengan CloudTrail, Anda dapat melihat riwayat lengkap dari aktivitas pengguna dan panggilan API untuk aplikasi maupun sumber daya Anda.

Cara kerjanya sederhana, setiap permintaan yang dikirimkan ke AWS, seperti meluncurkan EC2 instance; menambahkan baris ke tabel DynamoDB; atau mengubah izin pengguna, semuanya akan tercatat di mesin CloudTrail.

Anda bisa menganggap AWS CloudTrail sebagai log tindakan atau “jejak” yang ditinggalkan seseorang. Mesin akan mencatat dengan tepat tentang identitas pemanggil API, waktu panggilan, alamat IP pemanggil, dan masih banyak lainnya. Dari perspektif audit, ini adalah hal yang luar biasa.

Mari kita buat perumpamaannya. Bayangkan Anda sedang berurusan dengan auditor. Ia ingin memeriksa dan memastikan bahwa tidak ada siapa pun dari internet yang dapat mengakses database Anda. Ini mudah, Anda telah membangun security group yang mengunci lalu lintas eksternal.

Tetapi, sang auditor pun bertanya, “Administrator masih memiliki *permission* (izin) untuk mengubah pengaturan tersebut, bukan?”

Nah, bagaimana membuktikan kepada auditor bahwa pengaturan security group Anda itu tidak pernah berubah?”

Jawabannya sudah jelas, AWS CloudTrail. Anda dapat menyimpan log tersebut tanpa batas waktu ke dalam S3 bucket yang aman. Bahkan, Anda dapat menyimpannya dengan menggunakan metode anti-gangguan seperti Vault Lock--telah kita pelajari di modul penyimpanan dan database.

Selain itu, Anda juga dapat mengaktifkan CloudTrail Insights. Ini adalah fitur opsional yang memungkinkan CloudTrail secara otomatis mendeteksi aktivitas API yang mencurigakan di akun AWS Anda.

Misalnya, baru-baru ini jumlah Amazon EC2 instance yang diluncurkan oleh akun Anda lebih banyak dari biasanya. Nah, Anda dapat meninjau detail kejadian tersebut secara lengkap dengan CloudTrail Insights guna menentukan tindakan apa yang perlu diambil selanjutnya.

Studi Kasus: AWS CloudTrail Event
Selamat datang kembali di modul yang mungkin sudah Anda rindukan ya pastinya, yakni Studi Kasus.

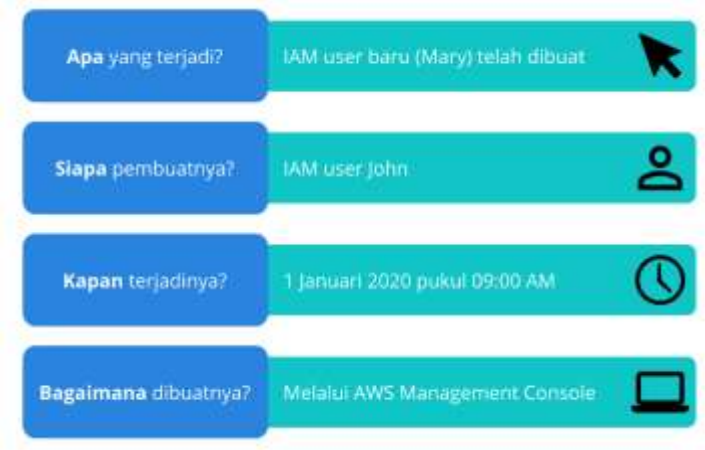
Oke. Di studi kasus kali ini kita akan menelaah tentang layanan AWS CloudTrail, tentunya dengan skenario kedai kopi.

Katakanlah Anda--sebagai pemilik kedai kopi--sedang menjelajahi bagian AWS Identity and Access Management (AWS IAM) dari AWS Management Console.

Anda menemukan adanya pembuatan IAM user baru bernama Mary. Namun, Anda tidak tahu siapa pembuatnya, kapan terjadinya, atau metode apa yang digunakan. Tahukah Anda cara untuk menyelesaikan permasalah di atas? Yup! Anda perlu menuju ke halaman AWS CloudTrail.

Di sana Anda bisa membuka CloudTrail Event History dan menerapkan filter agar hanya menampilkan peristiwa untuk tindakan API "CreateUser" di IAM.

Katakanlah Anda telah menemukan suatu catatan kejadian berupa panggilan API yang mengindikasikan pembuatan IAM user bernama Mary. Catatan ini memberikan detail yang begitu lengkap tentang apa yang terjadi. Tahukah Anda apa isi catatan tersebut?



Pada 1 Januari 2020 pukul 09.00 pagi, IAM user bernama John membuat pengguna baru (Mary) melalui AWS Management Console. Wah, menarik ‘kan?

Nah, dengan ini Anda sudah berhasil menyelesaikan studi kasus di atas menggunakan AWS CloudTrail. Selamat! Anda telah menjadi Sherlock Holmes yang baru.

AWS Trusted Advisor

Saat menjalankan kedai kopi, mungkin Anda memerlukan seorang penasihat yang datang menganalisis sistem dan berkata, “Hei, proses ini harus disederhanakan.”

Atau, “Saya punya beberapa tips bagus tentang cara menghemat uang untuk bisnis Anda.”

Atau bahkan, “Tahu, nggak? Saya bisa masuk ke area mesin kasir Anda dan membuka laci kasnya tanpa ada yang memperhatikan. Itu tidak baik!”

Poinnya, akan sangat sempurna jika Anda kenal seseorang yang mengerti praktik terbaik di industri, paham apa yang harus dianalisis, dan memberi tahu apa yang perlu dilakukan. Jika poin-poin di atas diterapkan, maka sistem dapat berjalan lebih efisien, lebih aman, dan lebih hemat.

Lalu, bagaimana cara untuk menerapkannya? AWS memiliki penasihat otomatis yang disebut dengan AWS Trusted Advisor. Ia adalah layanan web yang memeriksa lingkungan AWS Anda dan memberikan rekomendasi secara *real time* sesuai dengan praktik terbaik AWS.

Jika Anda penasaran, berikut adalah tampilan dari AWS Trusted Advisor dashboard:

Diambil dari AWS Trusted Advisor.

Ada 3 kategori yang terdapat pada dashboard tersebut, di antaranya:

- **Centang hijau**: menunjukkan jumlah item yang terdeteksi *tanpa masalah*.
- **Segitiga oranye**: mewakili jumlah saran yang mungkin perlu Anda *investigasi*.
- **Lingkaran merah**: mengindikasikan jumlah rekomendasi yang perlu Anda *tindak lanjuti*.

Dapat Anda lihat pada gambar tersebut, AWS Trusted Advisor mengevaluasi sumber daya Anda berdasarkan 5 pilar, di antaranya:

- *Cost optimization* (pengoptimalan biaya)
- *Performance* (kinerja)
- *Security* (keamanan)
- *Fault tolerance* (toleransi terhadap kesalahan)
- *Service limits* (batas layanan)

AWS Trusted Advisor juga menyertakan daftar rekomendasi tindakan dan materi pendukung di setiap pemeriksaan pilarnya agar Anda dapat mempelajari lebih lanjut tentang praktik terbaik AWS.

Setelah mengetahui apa itu AWS Trusted Advisor, sekarang mari kita uraikan contoh-contoh masalah yang dapat terdeteksi di setiap pilar dari layanan ini.

- **Cost optimization**
  Di pilar ini, contoh masalah yang bisa muncul adalah:
  - RDS instances yang tidak dipakai.
  - Beberapa EC2 instance yang jarang digunakan.
  - EBS volume yang tidak dimanfaatkan.

  Anda dapat melakukan *scaling down*--telah kita bahas di modul komputasi di cloud--terhadap instance yang jarang dipakai untuk menghemat biaya. Atau, Anda dapat menghapus sumber daya yang memang tidak digunakan sama sekali.

- **Performance**
  Salah satu contoh masalah yang muncul di pilar ini adalah pengiriman konten untuk Amazon CloudFront yang tidak teroptimasi.

- **Security**
  Beberapa contoh masalah yang muncul pada pilar security atau keamanan adalah:
  - *IAM password policy* atau kebijakan kata sandi IAM yang lemah untuk user.
  - MFA (multi-factor authentication) tidak diaktifkan untuk *root user*.
  - Security group yang mengizinkan akses publik ke EC2 instance.

  Semua hal ini membahayakan sumber daya di akun Anda dan harus ditangani secepat mungkin.

- **Fault tolerance**
  Berikut adalah beberapa contoh masalah yang dapat muncul:
  - EBS volume yang tidak memiliki snapshot. Ingat, snapshot adalah *backup* (cadangan). Tanpa backup, Anda akan kehilangan data di dalamnya jika volume EBS mengalami kegagalan.
  - Amazon EC2 yang tidak terdistribusi ke seluruh Availability Zone (AZ). Ini akan buruk jika salah satu AZ mengalami masalah, aplikasi Anda mungkin akan mengalami gangguan.

- **Service limits**
  Pilar ini akan memberi peringatan saat Anda mendekati atau mencapai batas layanan AWS. Contoh, limit dari kepemilikan VPC per Region adalah 5. Nah, jika telah mencapai batas tersebut, maka AWS Trusted Advisor akan segera memberi tahu Anda.

Oke. Jadi, AWS Trusted Advisor dapat memandu Anda ke arah yang lebih baik dalam hal 5 pilar yang telah kita bahas. Gunakan layanan ini dan mulailah mengambil tindakan. Semangat!

Ikhtisar

Kunci untuk menjaga aplikasi tetap efisien, aman, dan *compliant* (patuh) adalah dengan memahami apa yang terjadi di lingkungan cloud Anda.

Kita telah membahas beberapa hal di modul ini, di antaranya:

- Amazon CloudWatch dapat menyajikan informasi yang hampir *real-time* tentang bagaimana sistem Anda bekerja, termasuk memberi peringatan

ketika terjadi sesuatu. Bahkan, Anda juga dapat melihat metrik tersebut dari waktu ke waktu.

- AWS CloudTrail dapat membantu Anda untuk mengetahui dengan tepat tentang: *Siapa* melakukan *apa*, *kapan*, dan *dari mana*. Tentunya ia akan menjawab semua pertanyaan audit AWS Anda.
- Terakhir, AWS Trusted Advisor dapat menampilkan *dashboard* yang berisi lebih dari 40 masalah umum seputar biaya, kinerja, keamanan, dan ketahanan yang dapat ditindaklanjuti.

Semoga kesimpulan ini akan membantu Anda dalam memahami beberapa layanan pemantauan dan analitis yang AWS tawarkan.

Materi Pendukung
Ulas beberapa sumber di bawah ini untuk menelaah lebih lanjut tentang konsep yang sudah kita pelajari di modul ini:

- Management and Governance on AWS
- Monitoring and Observability
- Configuration, Compliance, and Auditing
- AWS Management & Governance Blog
- Whitepaper: AWS Governance at Scale

MODUL 7

Pengenalan ke Harga dan Dukungan

Pada skenario kedai kopi, Anda adalah pemiliknya. Tentu, Anda akan dibuat pusing dengan berbagai biaya yang timbul, seperti sewa bangunan, meja, gaji pegawai, pajak, listrik, pendingin udara, dan masih banyak lainnya.

Muncullah suatu pertanyaan, berapa biaya yang harus dibayar untuk bulan depan?

Solusi sederhananya adalah dengan menghitung biaya bulan ini lalu estimasikan untuk bulan berikutnya. Tapi itu hanya hitungan kasar, bagaimana untuk mengetahui detailnya?

Simpel, kita hanya perlu menghitung harga dari banyaknya item yang akan digunakan pada bulan depan. Beres, ‘kan? Lalu, berapa biaya yang harus dikeluarkan untuk pindah ke kota lain? Apa saja perbedaan biayanya? Tenang! Semua pertanyaan ini akan kita jawab di bagian selanjutnya.

Sst, *spoiler alert!* AWS menyediakan banyak layanan gratis untuk membantu Anda merencanakan dan menganalisis anggaran untuk lingkungan AWS. Penasaran? Mari selami lebih dalam!

AWS Free Tier

Saat membuat akun AWS baru ketika memulai perjalanan cloud, mungkin Anda merasa khawatir akan biaya yang harus dikeluarkan. Tenang saja, Anda tetap bisa mencoba beberapa layanan AWS secara gratis dengan AWS Free Tier (AWS Tingkat Gratis).

Tergantung pada layanannya, AWS Free Tier menyediakan tiga jenis penawaran:

- **Always free**
  Layanan yang termasuk ke dalam always free ini tersedia secara gratis untuk semua pelanggan AWS tanpa batas.

  Contohnya adalah AWS Lambda yang merupakan layanan komputasi serverless alias tanpa server. Layanan ini memungkinkan Anda untuk melakukan 1 juta request (permintaan) dan 3,2 juta detik waktu komputasi secara gratis setiap bulan.

  Layanan gratis lainnya adalah Amazon DynamoDB. Ia menawarkan 25 GB penyimpanan gratis per bulan.

- **12 months free**
  Anda memiliki waktu 12 bulan untuk mencoba beberapa layanan--yang mendukung 12 months free--secara gratis dimulai sejak tanggal pendaftaran akun AWS.

  Salah satu layanan yang masuk opsi ini adalah Amazon S3, yaitu layanan penyimpanan objek. Ia dapat digunakan secara gratis selama 12 bulan untuk kelas penyimpanan *S3 standard* hingga 5 GB. Layanan ini ideal jika Anda ingin menjalankan website statis di AWS.

  Selain Amazon S3, layanan lain yang masuk ke dalam opsi ini yaitu Amazon EC2 dan Amazon CloudFront. Amazon EC2 menawarkan 750 jam waktu komputasi per bulan, sementara Amazon CloudFront menyajikan 50 GB untuk transfer data keluar.

- **Trials**
  AWS memungkinkan Anda untuk mencoba beberapa layanan *trials* alias uji coba gratis jangka pendek terhitung sejak tanggal Anda mengaktifkan layanan tersebut.

  Lamanya waktu uji coba mungkin berbeda menurut jumlah hari atau jumlah penggunaan dalam layanan. Contoh layanannya adalah Amazon Inspector yang menawarkan 90 hari uji coba secara gratis.

  Selain itu, tersedia juga Amazon Lightsail, yaitu layanan yang memudahkan Anda untuk merancang aplikasi atau website dengan paket bulanan hemat biaya. Lightsail menawarkan uji coba gratis selama 30 hari dengan penggunaan hingga 750 jam. Layanan ini cocok Anda gunakan untuk menjalankan dan menguji blog WordPress.

  Tidak sampai di situ, masih ada beberapa layanan lain yang termasuk ke dalam AWS Free Tier, di antaranya Amazon SageMaker, Amazon Comprehend Medical, Amazon SNS, Amazon Cognito, dll. Jika Anda ingin melihat daftar lengkapnya, silakan kunjungi halaman website AWS Free Tier.

Konsep Harga AWS

Bagaimana Cara Kerja Harga AWS?
Dengan menggunakan AWS, Anda cukup membayar untuk setiap layanan dan tak perlu mendaftar dengan kontrak jangka panjang atau lisensi yang rumit. Sama halnya dengan membayar kebutuhan sehari-hari seperti air dan listrik, di AWS Anda hanya membayar layanan sesuai pemakaian. Setelah penggunaan berhenti atau tidak dilanjutkan, tak ada biaya tambahan atau biaya penghentian.

Ada beberapa keuntungan ketika Anda menggunakan AWS dalam hal biaya, mari kita uraikan.

- **Pay for what you use** (Bayar sesuai yang Anda gunakan)
  Saat menjalankan beban kerja di AWS, Anda tidak perlu repot-repot mengeluarkan banyak biaya, seperti membeli server atau menyewa gedung. Pengeluaran ini dapat Anda ganti dengan variabel lain yang lebih rendah, misal menggunakan beberapa layanan yang tepat sesuai kebutuhan.

  Untuk setiap layanan, Anda cukup membayar sesuai dengan jumlah sumber daya yang Anda gunakan, tanpa memerlukan komitmen jangka panjang.

- **Pay less when you reserve** (Berhemat saat Anda memesan di awal)
  Beberapa layanan menawarkan opsi reservasi yang memberikan diskon signifikan daripada harga instance dengan opsi penagihan On-Demand.

  Jika Anda memiliki kebutuhan beban kerja yang harus senantiasa berjalan, maka gunakanlah Amazon EC2 dengan opsi penagihan Savings Plans. Dengan Savings Plans, Anda dapat menghemat hingga 72% dibandingkan dengan kapasitas instance On-Demand.

  Anda tidak lupa ‘kan dengan opsi penagihan di atas? Kita telah membahasnya pada Modul Komputasi di Cloud ya.

- **Pay less with volume-based discounts when you use more** (Bayar lebih murah dengan diskon berbasis volume saat Anda menggunakan lebih sering)
  Beberapa layanan menawarkan harga yang berjenjang. Sehingga, biaya per unit akan semakin rendah jika Anda semakin sering menggunakannya.

  Salah satu jenis layanan yang dapat membantu Anda untuk menjaga pengeluaran tetap rendah adalah layanan penyimpanan AWS, misalnya Amazon S3. Semakin banyak ruang penyimpanan yang Anda gunakan, maka akan semakin sedikit pula harga per GB-nya.

AWS Pricing Calculator

AWS Pricing Calculator memungkinkan Anda untuk menjelajahi layanan AWS dan membuat estimasi biaya untuk kasus penggunaan di AWS. Layanan ini berguna baik untuk yang belum pernah menggunakan AWS maupun yang ingin mengatur ulang atau memperluas penggunaan AWS.

AWS Pricing Calculator memiliki banyak manfaat, di antaranya Anda dapat:

- Memodelkan arsitektur yang diinginkan sebelum membangunnya.
- Menelusuri setiap harga sekaligus membuat perhitungan.
- Menemukan tipe instance yang tersedia beserta persyaratan kontrak yang dapat memenuhi kebutuhan Anda.

Menarik, bukan? Dengan begitu, Anda dapat membuat keputusan tentang sumber daya apa saja yang akan Anda gunakan di AWS. Bahkan, Anda juga dapat membuat perencanaan biaya yang kelak Anda bayarkan.

Mari kita ambil contoh penggunaannya. Katakanlah Anda tertarik menggunakan Amazon EC2. Namun, Anda belum yakin AWS Regions atau tipe instance mana yang paling hemat biaya untuk kasus penggunaan Anda. Nah, dengan AWS Pricing Calculator, Anda dapat memasukkan detail seperti jenis sistem operasi, kebutuhan memori, dan input/output (I/O) yang Anda butuhkan.

Lebih menariknya lagi, AWS Pricing Calculator ini tidak menaruh biaya sama sekali alias gratis untuk Anda gunakan. Selamat mengestimasi!

Contoh Harga AWS
Di materi ini kita akan membahas lebih detail tentang harga di AWS. Terdapat tiga karakteristik mendasar yang Anda bayar: komputasi, penyimpanan, dan transfer data. Karakteristik tersebut beragam tergantung pada produk AWS yang Anda gunakan, namun ketiga hal ini memiliki pengaruh terbesar pada biaya.

Sekarang mari kita uraikan penetapan harga untuk beberapa produk AWS yang umum digunakan, seperti AWS Lambda, Amazon EC2, dan Amazon S3.

AWS Lambda
Saat menggunakan AWS Lambda, Anda dikenai biaya berdasarkan jumlah request (permintaan) untuk fungsi Anda dan waktu yang diperlukan untuk menjalankannya.

Layanan ini memungkinkan Anda untuk melakukan 1 juta request dan 3,2 juta detik waktu komputasi secara gratis setiap bulan.

Anda juga dapat menghemat biaya AWS Lambda dengan mendaftar Compute Savings Plan. Compute Savings Plan menawarkan biaya komputasi yang lebih rendah dengan berkomitmen pada jumlah penggunaan yang konsisten selama jangka waktu 1 atau 3 tahun. Ini adalah contoh mekanisme harga *pay less when you reserve*.

Sekarang mari kita lihat contoh tagihan untuk AWS Lambda. Perhatikan gambar berikut:

US West (Oregon)		**$0.08**
AWS Lambda USW2-Lambda-GB-Second		$0.00
AWS Lambda - Compute Free Tier - 400,000 GB-Seconds - US West (Oregon)	35,211.800 Second	$0.00
AWS Lambda USW2-Request		$0.08
AWS Lambda - Total Requests - US West (Oregon)	402,929 Requests	$0.08

Tagihan di AWS Lambda akan dikelompokkan menurut AWS Regions. Untuk contoh di atas, penggunaan AWS Lambda terjadi di Region US West (Oregon). Bisa Anda lihat, daftar tagihan untuk jumlah request dan total durasi dibuat terpisah.

Amazon EC2

Dengan Amazon EC2, Anda hanya membayar waktu komputasi sesuai yang digunakan. Bahkan, Anda juga dapat mengurangi biayanya secara signifikan dengan menggunakan Spot Instances--telah kita bahas pada Modul Komputasi di Cloud--untuk beberapa beban kerja.

Misalnya, Anda menjalankan pekerjaan *batch processing* yang tak masalah dengan interupsi. Nah, Spot Instances dapat memberi Anda penghematan biaya hingga 90% sekaligus tetap memenuhi kebutuhan atas ketersediaan beban kerja Anda.

Selain Spot Instances, Anda juga dapat melakukan penghematan tambahan untuk Amazon EC2 dengan Savings Plans dan Reserved Instances. Mari kita lihat contoh tagihan untuk Amazon EC2 berikut:

Diambil dari Back to Basics: Getting Started with AWS Billing Console.

Pada gambar di atas contoh tagihan mencakup

- tipe instance yang telah digunakan;

- jumlah ruang penyimpanan Amazon EBS yang telah digunakan; dan
- durasi penggunaan Elastic Load Balancing.

Amazon S3
Saat Anda menggunakan Amazon S3 pertimbangkanlah komponen biaya berikut:

- **Penyimpanan**
  Ingat! Di Amazon S3, Anda hanya membayar untuk penyimpanan yang digunakan. Tarif yang dikenakan akan dihitung berdasarkan ukuran, *storage class* (kelas penyimpanan), dan durasi penyimpanan setiap objek.

- **Permintaan dan pengambilan data**
  Anda akan dikenakan biaya untuk jumlah permintaan yang dibuat untuk objek dan Amazon S3 bucket.

  Misalnya, Anda menyimpan file foto di Amazon S3 bucket dan menggunakannya untuk website statis. Nah, setiap kali pengunjung membuka website, itu akan dihitung sebagai permintaan yang harus Anda bayar.

- **Transfer data**
  Di Amazon S3, Anda membayar untuk semua bandwidth yang masuk dan keluar dari Amazon S3, *kecuali* untuk:
  - Data yang ditransfer masuk dari internet.
  - Data yang ditransfer keluar ke Amazon EC2 instance di AWS Regions yang sama dengan S3 bucket (termasuk ke akun yang berbeda di Region yang sama).
  - Data yang ditransfer keluar ke Amazon CloudFront.
  - Data yang ditransfer antara S3 bucket yang berbeda atau dari Amazon S3 ke layanan lain dalam AWS Regions yang sama.

- **Manajemen dan replikasi**
  Anda akan dikenakan tarif atas fitur manajemen penyimpanan yang telah Anda aktifkan di Amazon S3 bucket. Fitur-fitur ini termasuk Amazon S3 Inventory, S3 Storage Class Analysis, dan S3 Object Tagging.

Nah, sekarang coba perhatikan gambar berikut:

Gambar di atas adalah contoh tagihan penggunaan Amazon S3 di dua Region: Asia Pacific (Singapore) dan US West (Oregon). Untuk setiap Region, rincian biaya didasarkan pada faktor-faktor berikut:

- Jumlah permintaan untuk menambah atau menyalin objek ke dalam bucket.
- Jumlah permintaan untuk mengambil/mengunduh objek dari bucket.
- Jumlah ruang penyimpanan yang digunakan.

Billing Dashboard

Halo, pecinta AWS! Saat memulai perjalanan cloud di AWS, tentu Anda ingin tahu tentang informasi penagihan pada akun AWS Anda, bukan?

Nah, di modul ini kita akan lihat seperti apa billing dashboard atau dashboard penagihan di AWS. Siap? Mari kita mulai!

Perkenalkan, AWS Billing and Cost Management dashboard. Ia adalah layanan yang dapat Anda gunakan untuk melihat informasi penagihan, membayar tagihan AWS, memantau penggunaan, menganalisis, dan mengontrol biaya.

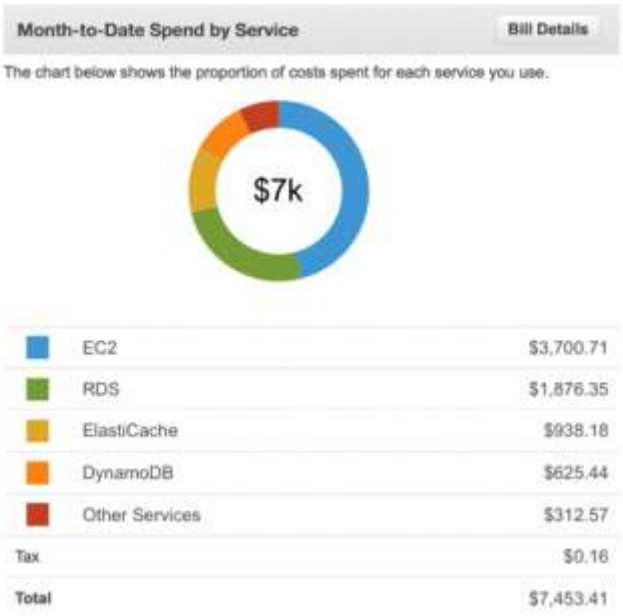
Supaya lebih jelas, silakan amati gambar berikut:

Catatan: Gambar-gambar yang ada di modul ini adalah tampilan demo agar mudah Anda bayangkan.

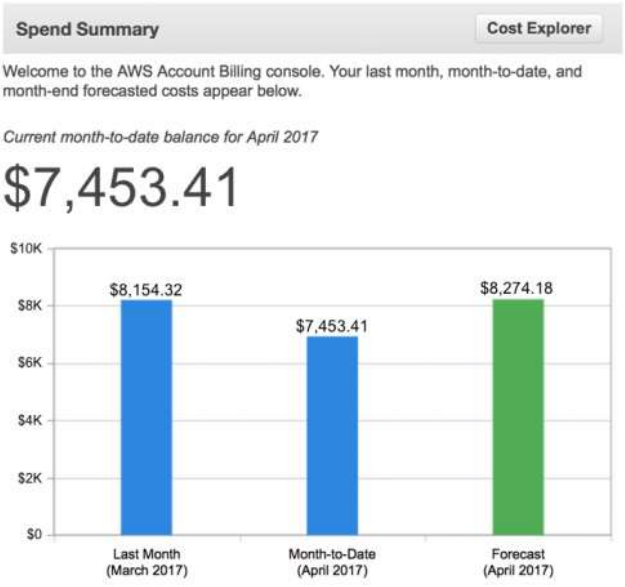
Diambil dari AWS Startups Blog: Getting Started with AWS Cost Management.

Gambar di atas adalah halaman utama dari layanan AWS Billing and Cost Management. Di sana, Anda memiliki beberapa informasi yang bermanfaat. Yuk kita telaah!

- Di sebelah kanan terdapat informasi pembelanjaan bulanan Anda sekaligus mengurutkan layanan yang memiliki pengeluaran terbesar.



- Di sebelahnya, Anda dapat membandingkan pengeluaran pada bulan sebelumnya, bulan ini, dan perkiraan bulan depan.



- Anda dapat melihat penggunaan AWS Free Tier berdasarkan layanan. Bahkan, Anda bisa melihat persentase penggunaan bulanan. Keren, ‘kan?

Top Free Tier Services by Usage — View all

Service	Free Tier usage limit	Month-to-date usage
Amazon Elastic Compute Cloud	30 GB of Amazon Elastic Block Storage in any combination of General Purpose (SSD) or Magnetic	57.93% (17.36/30 GB-Mo)
AmazonCloudWatch	5 GB of Log Data Archive for Amazon Cloudwatch	0.31% (0.02/5 GB-Mo)
Amazon Simple Storage Service	2,000 Put, Copy, Post or List Requests of Amazon S3	0.10% (2.00/2,000 Requests)
AWS Key Management Service	20,000 free requests per month for AWS Key Management Service	0.03% (5.00/20,000 Requests)
Amazon Simple Storage Service	5 GB of Amazon S3 standard storage	0.02% (0.00/5 GB-Mo)

- Anda juga memiliki akses ke layanan penagihan lain, seperti AWS Cost Explorer, AWS Budgets, dan sebagainya.

- Di AWS Billing and Cost Management, Anda juga dapat melihat tagihan secara lebih detail. Biaya yang harus Anda bayarkan telah dikelompokkan menurut Region.

Simple Storage Service		**$0.00**
US East (N. Virginia)		**$0.00**
Amazon Simple Storage Service Requests-Tier1		$0.00
$0.00 per request - PUT, COPY, POST, or LIST requests under the monthly global free tier	185,000 Requests	$0.00
Amazon Simple Storage Service Requests-Tier2		$0.00
$0.00 per request - GET and all other requests under the monthly global free tier	923,000 Requests	$0.00
Amazon Simple Storage Service TimedStorage-ByteHrs		$0.00
$0.000 per GB - storage under the monthly global free tier	0.159 GB-Mo	$0.00
US East (Ohio)		**$0.00**
Amazon Simple Storage Service USE2-Requests-Tier2		$0.00
$0.00 per request - GET and all other requests under the monthly global free tier	4,000 Requests	$0.00
Amazon Simple Storage Service USE2-TimedStorage-ByteHrs		$0.00
$0.000 per GB - storage under the monthly global free tier	0.000001 GB-Mo	$0.00

Diambil dari E-learning AWS Cloud Practitioner Essentials.

Nah, itulah cara sederhana yang dapat Anda lakukan untuk memeriksa pengeluaran dan tagihan di AWS.

Consolidated Billing

Masih ingatkah kedai kopi yang Anda miliki di seluruh kota? Kita telah membahasnya di Modul Infrastruktur Global dan Keandalan.

Setiap kedai kopi tersebut memiliki pengeluaran dan keuntungannya masing-masing. Oleh karena itu, Anda ingin uang tersebut mengalir ke dan dari satu rekening utama, yaitu rekening Anda selaku pemilik semua kedai kopi itu.

Misalnya, ada seorang petugas reparasi datang untuk memperbaiki mesin kopi yang rusak di kedai kopi A. Anda ingin orang itu menagih langsung kepada Anda, bukan ke pegawai kedai kopi A. Bagaimana caranya?

Nah, ide tersebut dapat Anda wujudkan dengan AWS Organizations. Seperti yang telah kita pelajari sebelumnya, layanan ini dapat mengelola beberapa akun dari satu lokasi terpusat. Ia juga memiliki sejumlah fitur menarik, salah satunya adalah *consolidated billing* (tagihan terkonsolidasi). Apa itu?

Consolidated billing adalah fitur yang memungkinkan Anda untuk mendapatkan satu tagihan untuk semua akun AWS yang ada di organisasi. Dengan demikian, Anda tak perlu membayar tagihan untuk setiap akun. Contoh, jika memiliki 100 akun AWS, Anda tak akan mendapatkan 100 tagihan, melainkan hanya satu tagihan terpusat. Keren, ‘kan? Tentunya ini akan mempermudah Anda dalam melacak dan mengelola tagihan.

Bahkan tak hanya itu, manfaat lain dari consolidated billing adalah Anda dapat mendistribusikan *bulk discount pricing* (harga diskon massal), Savings Plans, dan Reserved Instances di seluruh akun pada organisasi Anda.

Maksudnya begini. Satu akun AWS saja mungkin tidak akan memiliki penggunaan bulanan yang cukup untuk memenuhi syarat diskon. Namun, jika beberapa akun digabungkan, penggunaannya dapat menghasilkan manfaat yang berlaku untuk semua akun yang ada di dalam organisasi.

Tunggu, masih ada bagian terbaiknya. Fitur ini gratis dan mudah digunakan. *Nice!*

Studi Kasus: Consolidated Billing
Selamat datang kembali di Modul Studi Kasus. Kali ini kita akan membahas tentang penggunaan consolidated billing di kehidupan nyata.

Katakanlah Anda memiliki perusahaan yang menaungi semua kedai kopi di seluruh kota. Sebagai seorang pemimpin, Anda ingin mengawasi setiap tagihan yang terjadi pada masing-masing akun AWS perusahaan.

Anda memiliki 3 akun (di luar akun utama) untuk setiap departemen: Keuangan, IT, dan Operasional. Anda memutuskan untuk membuat organisasi di AWS Organizations dan menambahkan tiga akun tersebut.

Setiap bulan, AWS melakukan penagihan terhadap akun utama Anda untuk semua akun yang ada di organisasi dengan consolidated billing. Amati gambar berikut:

Perhatikan gambar di atas. Sebagai pemegang akun utama, Anda bisa mendapatkan laporan biaya terperinci untuk setiap akun, termasuk akun utama.

Bisa Anda lihat bahwa setiap akun memiliki pengeluaran yang bervariasi:

- **Akun Utama** : $14.14
- **Akun Keuangan** : $19.64
- **Akun IT** : $19.96
- **Akun Operasional** : $20.06

Jadi, total biaya yang harus Anda keluarkan pada akun utama adalah $73.80.

Selain itu, consolidated billing juga memungkinkan Anda untuk berbagi *volume pricing discounts* (diskon harga volume) di seluruh akun. Contohnya, beberapa layanan AWS (seperti Amazon S3) dapat memberikan harga yang lebih rendah saat Anda semakin sering menggunakannya.

Di Amazon S3 setelah menggunakan 10 TB data dalam sebulan, Anda bisa membayar harga transfer per GB lebih rendah untuk 40 TB data berikutnya.

Perhatikan gambar berikut:

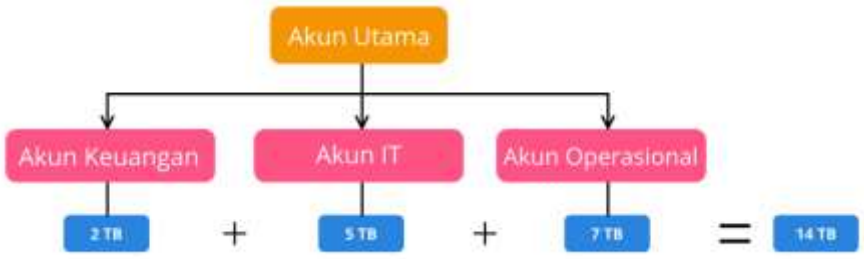
Gambar di atas adalah contoh dari proses transfer data pada akun departemen Anda ketika **tidak** menggunakan consolidated billing.

Selama bulan ini setiap akun mentransfer sejumlah data yang berbeda di Amazon S3:

- Keuangan mentransfer data sebesar 2 TB.
- IT mentransfer data sebesar 5 TB.
- Operasional mentransfer data sebesar 7 TB.

Nah, karena tak ada satu akun pun yang melewati jumlah transfer 10 TB, Anda tidak akan memperoleh harga transfer per GB yang lebih murah untuk 40 TB data berikutnya.

Sekarang mari kita contohkan proses transfer data pada akun departemen Anda ketika menggunakan consolidated billing.



Ketika penggunaan Amazon S3 untuk ketiga akun tersebut digabungkan, yakni (2 + 5 + 7), ia menghasilkan jumlah transfer data sebesar 14 TB. Artinya, angka ini telah melebihi ambang batas 10 TB sehingga Anda akan mendapat harga transfer per GB yang lebih rendah untuk data transfer 40 TB berikutnya.

Dengan consolidated billing, AWS akan menggabungkan penggunaan dari semua akun untuk menentukan tingkat harga volume mana yang akan diterapkan. Bahkan, ia juga dapat memberikan harga keseluruhan yang lebih rendah bila memungkinkan. Kemudian, AWS akan mengalokasikan sebagian dari keseluruhan diskon volume untuk setiap akun berdasarkan penggunaan.

Pada contoh di atas, akun Operasional akan menerima porsi yang lebih besar karena ia telah melakukan transfer data sebesar 7 TB. Jumlah ini lebih banyak dibandingkan dengan Keuangan (2 TB) dan IT (5 TB).

AWS Budgets

Semakin meningkatnya produktivitas di AWS, Anda mungkin ingin memastikan bahwa uang yang dikeluarkan tidak akan melebihi anggaran.

Misalnya Anda ingin membuat suatu anggaran untuk penggunaan Amazon EC2. Anda tak ingin biayanya melebihi 1 juta rupiah. Nah, bagaimana kita bisa mewujudkan kebutuhan semacam ini di AWS?

Perkenalkan, AWS Budgets. Ia sangat mudah digunakan, bahkan segampang membuat anggaran untuk pengeluaran pribadi Anda sendiri. Tentu Anda sudah merasa familier ‘kan?

AWS Budgets dapat membantu Anda untuk menetapkan anggaran pada berbagai skenario, seperti biaya atau penggunaan layanan. Bahkan, layanan ini bisa mengirimkan notifikasi saat penggunaan Anda sudah melebihi jumlah batas anggaran.

Sekarang silakan amati gambar berikut:

Diambil dari E-learning AWS Cloud Practitioner Essentials.

Gambar di atas adalah contoh anggaran yang dibuat dengan AWS Budgets. Terdapat beberapa informasi penting yang perlu Anda tinjau, perhatikan baris anggaran teratas bernama **Project Nemo Cost Budget**.

- Jumlah pengeluaran pada kolom *current* (terkini) adalah sebesar $43.90.
- Total pengeluaran pada kolom *forecasted* (perkiraan) untuk bulan ini adalah $56.33. Jumlah ini adalah hasil kalkulasi dari pola penggunaan sumber daya Anda.
- Anda juga bisa melihat perbandingan antara pengeluaran *current vs budgeted* (dianggarkan), dan *forecasted vs budgeted*. Pada gambar di atas, komparasi untuk total forecasted vs budgeted adalah 125.17%. Wah besar sekali ya. Ini terjadi karena jumlah forecasted ($56.33) melebihi jumlah yang telah dianggarkan alias budgeted ($45.00). Tentu, ini sudah melebihi anggaran yang direncanakan sebelumnya.

Nah, itulah materi kita kali ini tentang AWS Budgets. Semangat ya, kita sudah setengah jalan di modul ini. *Keep the spirit high!*

AWS Cost Explorer

Seperti yang telah kita pelajari, AWS memiliki model biaya yang variabel dan Anda hanya membayar sesuai pemakaian. Setiap akhir bulan, tagihan tak akan terus-menerus sama, melainkan berubah-ubah sesuai dengan sumber daya yang Anda gunakan. Oleh karena itu, Anda perlu menelusuri tagihan dan pengeluaran di AWS.

AWS memiliki layanan yang disebut dengan AWS Cost Explorer. Ia adalah layanan berbasis konsol yang dapat meninjau dan menganalisis secara visual pengeluaran Anda di AWS.

AWS Cost Explorer dapat menunjukkan layanan mana yang memiliki pengeluaran terbesar. Bahkan, ia akan memberikan 12 bulan data historis sehingga Anda bisa melacak pengeluaran dari waktu ke waktu.

Pengeluaran Anda dapat divisualisasikan dan dikelompokkan berdasarkan beberapa atribut: layanan, AWS Regions, tipe instance, tag, dll. Di antara atribut tersebut, tag adalah salah satu pengelompokan yang cukup penting untuk Anda terapkan.

Tag pada dasarnya adalah pasangan *key-value* (kunci-nilai). Dengan tag, Anda dapat memberi label pada beberapa sumber daya (misal EC2 instance) dengan nama tertentu. Lalu visualisasikan semua biaya yang terkait dengan tag tersebut.

Supaya lebih mudah, amati gambar berikut:

Diambil dari E-learning AWS Cloud Practitioner Essentials.

Gambar di atas adalah contoh dari AWS Cost Explorer dashboard yang menampilkan biaya bulanan untuk Amazon EC2 instance selama periode 6 bulan. Baris untuk setiap bulan menunjukkan pengeluaran dari berbagai tipe Amazon EC2 instance (seperti t2.micro atau m3.large).

Dengan menganalisis pengeluaran AWS dari waktu ke waktu, Anda dapat membuat keputusan yang tepat untuk biaya di masa mendatang. Selamat bereksplorasi!

AWS Support Plans

Tahukah Anda salah satu hal hebat tentang AWS? Tak peduli bisnis Anda besar atau kecil, sektor swasta atau publik, AWS akan senantiasa mendukung kebutuhan Anda.

AWS menawarkan 4 *support plan* alias paket dukungan berbeda untuk membantu Anda dalam memecahkan masalah, menghemat biaya, dan menggunakan layanan AWS secara efisien. Di antaranya:

- Basic
- Developer
- Business
- Enterprise

Mari kita kupas satu per satu.

AWS Basic Support

Secara otomatis, setiap pelanggan mendapatkan paket AWS Basic Support tanpa biaya sama sekali. Dengan paket ini, Anda akan dapat mengakses beberapa fungsi dukungan, seperti:

- Akses 24/7 ke customer service.
- Dokumentasi.
- Whitepaper.
- Forum dukungan.
- AWS Trusted Advisor.
- AWS Personal Health Dashboard (layanan yang dapat memberikan peringatan dan panduan remediasi saat AWS mengalami kejadian yang dapat memengaruhi sumber daya Anda).

Fungsi-fungsi tersebut gratis untuk semua pelanggan AWS. Tetapi, jika Anda mulai menjalankan beban kerja yang lebih kritis, AWS menawarkan tingkat dukungan yang lebih tinggi untuk menyesuaikan dengan kebutuhan Anda, yaitu AWS Developer Support.

AWS Developer Support

Di tingkat ini, Anda akan mendapatkan:

- Semua fitur yang ada di AWS Basic Support.
- Layanan diagnostik pada sisi klien.
- Panduan praktik terbaik.
- Dukungan arsitektur dasar yang terdiri dari panduan penggunaan penawaran, fitur, dan layanan AWS.
- Akses email ke customer support dengan waktu respons 24 jam untuk segala pertanyaan. Namun, jika sistem Anda mengalami gangguan, maka waktu respons menjadi kurang dari 12 jam.

Misalnya, Anda sedang menjelajahi layanan AWS. Anda sudah sedikit kenal dengan beberapa layanan, namun tak cukup yakin bagaimana cara penggunaannya untuk memenuhi kebutuhan. Nah, untuk masalah ini, AWS Developer Support bisa menjadi pilihan terbaik untuk Anda gunakan.

AWS Business Support

Selanjutnya, jika Anda mulai menerapkan beban kerja produksi, maka gunakanlah AWS Business Support. Tingkatan ini mendukung:

- Semua yang ada di AWS Basic Support dan AWS Developer Support.
- Seluruh rangkaian pemeriksaan AWS Trusted Advisor.
- Akses telepon langsung ke tim dukungan (cloud support engineer) yang memiliki SLA (service level agreement a.k.a jaminan) respons 4 jam jika sistem

produksi Anda mengalami gangguan dan 1 jam jika sistem Anda *down* atau tak bekerja.
- Akses ke Infrastructure Event Management dengan biaya tambahan. AWS dapat membantu Anda untuk merencanakan acara besar, seperti peluncuran produk baru atau periklanan global.
- Panduan kasus penggunaan untuk mengidentifikasi penawaran, fitur, dan layanan AWS yang paling mendukung kebutuhan spesifik Anda.
- Dukungan terbatas untuk perangkat lunak pihak ketiga, misalnya dalam proses instal, konfigurasi, dan pemecahan masalah pada sistem operasi pihak ketiga.

AWS Enterprise Support

Terakhir, jika Anda memiliki perusahaan yang menjalankan beban kerja penting, pilihlah AWS Enterprise Support. Ia memiliki:

- Segala fitur dari paket sebelumnya (Basic, Developer, Business).
- SLA 15 menit.
- Panduan arsitektur aplikasi, yaitu hubungan konsultatif yang dapat mendukung kasus penggunaan dan aplikasi spesifik Anda.
- Infrastructure Event Management.
- Technical Account Manager (TAM) yang akan mengoordinasikan akses ke program dan pakar AWS lainnya sesuai kebutuhan.

Mari kita bahas sedikit tentang TAM. Ia tak hanya memantau dan membantu pengoptimalan terhadap lingkungan AWS Anda, tetapi juga menyediakan infrastructure event management, tinjauan Well-Architected, dan reviu operasional.

Apa itu tinjauan Well-Architected? Sederhananya, TAM bekerja sama dengan pelanggan untuk meninjau arsitektur menggunakan Well-Architected Framework. Arsitektur AWS Anda akan diperiksa berdasarkan 5 pilar:

- Operational Excellence
- Security
- Reliability
- Performance Efficiency
- Cost Optimization

Materi tentang Well-Architected Framework ini akan kita bahas secara mendalam di modul mendatang.

Ketahuilah! AWS Support melihat pelanggan secara holistik, tidak hanya saat memiliki masalah, melainkan bagaimana dapat mendukung dan membantu pelanggan agar dapat sukses meraih tujuannya di AWS.

Untuk mempelajari selengkapnya tentang AWS Support, termasuk struktur harga dari paket tersebut, silakan kunjungi aws.amazon.com/premiumsupport.

AWS Marketplace

Cara lain yang dapat membantu Anda untuk menjalankan beban kerja di AWS adalah melalui AWS Marketplace. Ia adalah katalog digital pilihan yang memiliki ribuan perangkat lunak dari berbagai vendor.

Layanan ini dapat menyederhanakan langkah Anda guna menemukan, melakukan pengujian, dan membeli software pihak ketiga yang berjalan di arsitektur AWS.

Untuk setiap software yang ada di AWS Marketplace, Anda dapat mengakses rincian informasi terkait opsi harga, dukungan yang tersedia, dan ulasan dari pelanggan AWS lainnya.

Layanan ini dapat membantu Anda untuk meningkatkan kecepatan dan ketangkasan Anda dalam menggunakan AWS. Ketimbang harus membangun, menginstal, dan memelihara infrastruktur dasar yang diperlukan untuk menjalankan aplikasi pihak ketiga, Anda bisa melakukan *one-click deployment* (proses deploy sekali klik) untuk produk dari ribuan vendor.

Sebagian besar vendor di AWS Marketplace menawarkan mekanisme *pay-as-you-go* (bayar sesuai pemakaian). Bahkan, beberapa di antaranya juga menyediakan *free trials* (uji coba gratis) atau paket Quick Start (mulai dengan cepat) untuk membantu Anda bereksperimen dan mempelajari produk mereka.

AWS Marketplace menyajikan produk dalam beberapa kategori, seperti Infrastructure Software, Business Applications, Data & Analytics, DevOps, dll.

Diambil dari E-learning AWS Cloud Practitioner Essentials.

Selain itu, Anda juga dapat menjelajahi solusi perangkat lunak berdasarkan industri dan kasus penggunaan. Misalnya, jika Anda memiliki perusahaan yang bergerak di

industri perawatan kesehatan, Anda dapat meninjau beberapa kasus penggunaan, seperti:

- Menerapkan solusi untuk melindungi catatan pasien.
- Menggunakan model machine learning untuk menganalisis riwayat medis pasien.
- Memprediksi kemungkinan risiko kesehatan pasien.

Ikhtisar

Oke, sampai tahap ini, Anda telah belajar banyak hal tentang harga dan dukungan, di antaranya:

- Menjelaskan model pay-as-you-go dalam menggunakan sumber daya AWS.
- Membedakan antara biaya di data center on-premise vs cloud.
- Menerangkan AWS Free Tier yang menawarkan sebagian besar layanan AWS.
- Menelaah penggunaan AWS Organizations dan bagaimana ia dapat membantu Anda dengan consolidated billing dari beberapa akun AWS.
- Membahas layanan AWS Budgets.
- Menilik AWS Cost Explorer.
- Menjelajahi AWS Console billing.
- Membicarakan tentang berbagai model dukungan yang ditawarkan di AWS alias AWS Support Plans. Ini sangat berguna jika Anda ingin mencari bantuan dalam perjalanan cloud.
- Terakhir, kita telah mempelajari AWS Marketplace yang dapat Anda gunakan untuk mencari layanan *click-and-go* (klik dan jalankan).

Materi Pendukung
Untuk mempelajari lebih lanjut tentang konsep yang dibahas di modul ini, silakan tinjau sumber daya berikut:

- AWS Pricing
- AWS Free Tier
- AWS Cost Management
- Whitepaper: How AWS Pricing Works
- Whitepaper: Introduction to AWS Economics
- AWS Support
- AWS Knowledge Center

MODUL 8

Pengenalan ke Migrasi dan Inovasi

Wah, tak terasa ya perjalanan cloud kita sudah cukup jauh. Sampai tahap ini, kita telah memahami bagaimana pemakaian dan arsitektur di AWS Cloud. Lalu, bagaimana jika Anda yang mempunyai *existing* beban kerja di data center on-premise atau memulai perjalanan cloud tanpa AWS?

Nah, AWS dapat membantu Anda untuk memigrasikan beban kerja dengan hemat. Ini membawa kita pada topik migrasi dan inovasi.

Kita akan mempelajari semua opsi layanan migrasi, AWS Cloud Adoption Framework, dan Snow Family yang merupakan perangkat fisik untuk memindahkan data, baik masuk atau keluar dari AWS.

Selain itu, kita akan membahas cara migrasi dari lingkungan on-premise atau cloud ke AWS dan juga strategi 6 R. Apa itu? Yuk kita singkap materi berikutnya!

AWS Cloud Adoption Framework (AWS CAF)

Proses migrasi ke cloud bukanlah suatu hal yang instan. Tak bisa hanya sekadar menjentikkan jari dan *boom!* Secara ajaib semuanya berjalan di AWS. Tentu, Anda memerlukan upaya dan keahlian ekstra untuk melakukan proses migrasi.

Untungnya, sudah banyak orang yang berhasil melakukan migrasi cloud dan membagikan kisahnya. Dengan demikian, Anda bisa mendapatkan banyak wawasan sehingga tidak akan salah langkah.

Mari kita buat skenario. Katakanlah kedai kopi Anda sudah memiliki banyak cabang yang tersebar di seluruh kota. Semua kedai tersebut berada di bawah naungan perusahaan Anda. Nah, setelah mempelajari tentang AWS, Anda pun semakin mantap untuk melakukan migrasi ke AWS.

Agar proses tersebut terlaksana dengan baik, Anda pun membentuk tim khusus dan melibatkan banyak pegawai dari departemen yang berbeda.

Oke, berangkat dari kasus tersebut, timbul satu pertanyaan di benak Anda, “Bagaimana agar proses migrasi cloud ini dapat berjalan sukses?”

Simak baik-baik! Setiap posisi pada suatu perusahaan akan memiliki peran dan perspektif yang berbeda terkait proses migrasi cloud. Misalnya, seorang *developer* tentu akan mempunyai sudut pandang yang berlainan dibandingkan dengan arsitek cloud, analis bisnis, dan analis keuangan.

Perbedaan perspektif ini bukanlah masalah. Justru yang terpenting adalah Anda perlu memastikan setiap orang di dalam tim memiliki tujuan yang sama, yakni keberhasilan migrasi cloud.

Jangan khawatir! Tim layanan profesional AWS telah menciptakan AWS Cloud Adoption Framework (AWS CAF). Fungsinya untuk memberikan Anda panduan agar proses migrasi ke AWS menjadi lebih cepat dan lancar.

Framework atau kerangka kerja tersebut membagi panduan menjadi 6 area yang disebut dengan **perspektif**. Masing-masing perspektif meliputi tanggung jawab dari kelompok yang berbeda.

Secara umum, ia mencakup perspektif Business, People, dan Governance berfokus pada kemampuan bisnis. Lalu, perspektif Platform, Security, dan Operations berfokus pada kemampuan teknis.

Diambil dari [AWS Migration Whitepaper: The AWS Cloud Adoption Framework (AWS CAF)].

Agar dapat memahami lebih jelas, mari kita uraikan setiap perspektif tersebut:

- **Business** (Bisnis)
  Perspektif ini membantu Anda untuk beralih dari strategi yang semula memisahkan antara bisnis dan IT menjadi model bisnis yang mengintegrasikan strategi IT.
  - Peran umum dalam perspektif Business meliputi:
  - Manajer bisnis
  - Manajer keuangan
  - *Budget owners* (pemilik anggaran)
  - *Strategy stakeholders*

- **People** (Orang)
  Perspektif ini dapat membantu Anda dalam mempersiapkan tim dengan memperbarui skill staf dan proses organisasi untuk migrasi cloud. Perspektif ini juga membantu Anda dalam memprioritaskan pelatihan, kepegawaian, dan

perubahan organisasi.

Berikut adalah peran umum yang ada di dalam perspektif People:

- HR (*Human Resource*/Sumber Daya Manusia)
- Staf
- Manajer personalia

- **Governance** (Tata Kelola)
  Perspektif Governance berfokus pada mengintegrasikan IT Governance (Tata Kelola IT) dengan Organizational Governance (Tata Kelola Organisasi). Perspektif ini juga memberikan panduan untuk mengidentifikasi dan menerapkan praktik terbaik untuk IT Governance serta mendukung proses bisnis dengan teknologi.

  Anda juga dapat menggunakan perspektif Governance untuk memahami cara pembaruan keterampilan staf dan proses yang diperlukan untuk memastikan Business Governance (Tata kelola Bisnis) di cloud.

  Peran umum perspektif Governance ini mencakup:
  - CIO (*Chief Information Officer*)
  - Manajer program
  - Manajer proyek
  - Analis bisnis
  - Manajer portofolio

- **Platform**
  Perspektif Platform dapat membantu Anda untuk merancang, menerapkan, dan mengoptimalkan arsitektur teknologi AWS berdasarkan sasaran bisnis. Ia juga bisa memberikan panduan strategis untuk desain, prinsip, layanan, dan kebijakan yang akan Anda gunakan untuk menentukan infrastruktur AWS.

  Selain itu, perspektif ini juga mencakup prinsip dan pola yang berguna untuk mengimplementasikan solusi baru atau migrasi beban kerja on-premise Anda ke cloud.

  Beberapa peran umum perspektif Platform meliputi:
  - CTO (*Chief Technology Officer*)
  - Manajer IT
  - Arsitek cloud

- **Security** (Keamanan)
  Perspektif Keamanan akan memastikan organisasi/perusahaan Anda memenuhi tujuan keamanan untuk visibilitas, kemampuan audit, kontrol, dan ketangkasan (*agility*).

Peran umum perspektif ini di antaranya:
    - CISO (*Chief Information Security Officer*)
    - Manajer keamanan IT
    - Analisis keamanan IT

- **Operations** (Operasi)
  Perspektif Operasi dapat membantu Anda dalam mengaktifkan, menjalankan, menggunakan, mengoperasikan, dan memulihkan beban kerja IT ke tingkat yang disepakati dengan *business stakeholder* (pemangku kepentingan bisnis) Anda.

  Wawasan yang diperoleh melalui perspektif ini akan menentukan prosedur operasi Anda serta mengidentifikasi perubahan proses dan pelatihan supaya terwujudlah adopsi cloud yang sukses.

  Peran umum perspektif Operasi meliputi:
    - IT Operations Manager
    - IT Support Manager

Setiap perspektif tersebut dapat Anda gunakan untuk memperlihatkan celah pada proses yang terjadi di perusahaan sebelum migrasi ke cloud. Hal tersebut nantinya dapat Anda catat sebagai masukan pada pembuatan AWS Cloud Adoption Framework Action Plan.

Diambil dari The AWS Cloud Adoption Framework: Creating an Action Plan.

Nah, AWS CAF Action Plan inilah yang berguna untuk memandu proses migrasi, seperti manajemen perubahan organisasi pada perjalanan cloud Anda.

Oke, migrasi ke cloud memang bisa menjadi hal yang rumit. Tetapi, tenang! Tersedia banyak sumber yang dapat membantu Anda untuk memulai. Salah satu solusi yang tepat adalah dengan menggunakan AWS Cloud Adoption Framework.

Strategi Migrasi

Setelah kita belajar beberapa kerangka kerja yang dapat membantu Anda dalam mempersiapkan proses migrasi ke AWS, sekarang mari kita lihat bagaimana cara implementasinya.

Ingatkah Anda? Proses migrasi tak terjadi hanya dengan menjentikkan jari lalu seketika semua elemen dari data center on-premise Anda berjalan di AWS.

Setiap aplikasi yang berarsitektur *tightly coupled*--telah kita bahas di Modul Komputasi di Cloud--akan memiliki 6 kemungkinan opsi terkait strategi migrasi, AWS menyebutnya dengan **6 R**.

Saat Anda sudah yakin untuk bermigrasi ke AWS, pilihlah opsi strategi migrasi di antara 6 R yang paling sesuai menurut waktu, biaya, prioritas, dan kekritisan beban kerja Anda. Mari kita jabarkan **6 R** satu per satu:

- **Rehosting**
  Rehosting juga dikenal sebagai *lift and shift* (angkat dan pindahkan). Opsi ini mudah dilakukan karena Anda tak perlu membuat perubahan apa pun. Cukup pindahkan aplikasi yang Anda miliki ke AWS.

  Dengan rehosting, Anda bisa menghemat hingga 30% dari total biaya walaupun tidak ada pengoptimalan apa pun pada aplikasi Anda.

  Aplikasi akan lebih mudah untuk dioptimalkan setelah ia berjalan di cloud. Ini karena kemampuan Anda dalam menggunakan AWS akan semakin berkembang seiring proses migrasi cloud dan bagian yang sulit--migrasi aplikasi, data, dan traffic--telah selesai sehingga Anda bisa fokus untuk mengoptimalkan aplikasi.

- **Replatforming**
  Strategi migrasi yang satu ini disebut juga dengan *lift, tinker, and shift* (angkat, perbaiki, dan pindahkan). Pada dasarnya, opsi ini masih berupa lift and shift namun Anda dapat melakukan beberapa pengoptimalan cloud. Dalam strategi ini, Anda tak akan mengubah arsitektur inti (*core architecture*) apa pun pada proses migrasi aplikasi.

  Sebagai contoh, Anda dapat mengurangi jumlah waktu yang dihabiskan untuk mengelola database instance dengan bermigrasi ke platform *database-as-a-service* seperti Amazon Relational Database Service (Amazon RDS). Atau, Anda dapat memigrasikan aplikasi Anda ke platform yang dikelola sepenuhnya seperti AWS Elastic Beanstalk.

- **Retiring**
  AWS menemukan bahwa sebanyak 10% hingga 20% dari portofolio IT perusahaan menyertakan aplikasi yang tak lagi digunakan dan bisa dimatikan. Maka dari itu, hapuslah aplikasi yang tidak lagi Anda butuhkan.

  Penghematan semacam ini dapat memberikan beberapa keuntungan, seperti meningkatkan *business case* (kasus bisnis), memfokuskan tim terhadap aplikasi yang digunakan, dan mengurangi jumlah aplikasi yang harus dikelola.

- **Retaining**
  Saat hendak melakukan migrasi ke AWS, mungkin Anda memiliki beberapa aplikasi yang akan segera deprecated (dihentikan). Tetapi, aplikasi tersebut masih perlu digunakan selama beberapa waktu.

  Nah, daripada memindahkan aplikasi tersebut ke AWS, sebaiknya Anda hanya memigrasikan aplikasi yang sekiranya masuk akal dan berguna untuk bisnis.

- **Repurchasing**
  Strategi migrasi ini umum terjadi pada perusahaan yang ingin meninggalkan vendor perangkat lunak lama dan memulai yang baru. Misalnya:
  - Mengakhiri kontrak dengan vendor CRM (*Customer Relationship Management*) lama dan pindah ke yang baru seperti Salesforce.com.
  - Beralih dari sistem HR (Human Resource/Sumber Daya Manusia) ke aplikasi Workday.
  - Berpindah dari perangkat lunak CMS (Content Management System) ke software Drupal.
  - Mengakhiri lisensi dengan vendor database yang usang dan migrasi ke database cloud.

  Oke, mungkin ini terdengar bagus. Tapi ingat! Anda akan berurusan dengan *software package* (paket perangkat lunak) yang baru. Beberapa di antaranya mudah diimplementasikan tetapi ada juga yang membutuhkan waktu.

  Oleh karena itu, total biaya di muka untuk strategi ini akan naik, tetapi potensi manfaatnya bisa sangat besar.

- **Refactoring/re-architecting**
  Pada opsi inilah Anda menulis kode yang baru. Hal ini didorong oleh kebutuhan bisnis yang kuat untuk menambahkan fitur, skala, atau kinerja yang mungkin sulit diraih pada data center on-premise.

  Perubahan dramatis pada arsitektur Anda bisa sangat bermanfaat. Namun, akan menimbulkan biaya awal yang paling tinggi dalam hal perencanaan dan usaha migrasi.

Oke. Itulah 6 strategi migrasi yang dapat Anda terapkan untuk migrasi ke AWS. Pilihlah opsi yang tepat demi suksesnya proses migrasi Anda.

AWS Snow Family

Beberapa pelanggan memerlukan suatu cara yang dapat mengirimkan data ke AWS dengan cara yang efisien dan cepat. Apakah Anda juga termasuk jenis pelanggan tersebut?

Metode yang biasanya dilakukan adalah melakukan transfer data yang diperlukan melalui internet atau AWS Direct Connect--telah kita bahas di Modul Jaringan. Namun, dengan keterbatasan *bandwidth* (jumlah maksimum data yang dapat dikirim), secara umum proses tersebut dapat memakan waktu berhari-hari, berminggu-minggu, atau bahkan berbulan-bulan.

Misalnya, untuk memindahkan 1 PB (*petabyte*) data, koneksi jaringan yang memiliki kecepatan 1 Gbps (*gigabit per second*) secara teori membutuhkan waktu sekitar 100 hari. Di dunia nyata, itu bisa memakan waktu lebih lama dan bahkan menguras biaya yang besar

Nah, maka dari itu, AWS memperkenalkan perangkat AWS Snow Family. Layanan ini adalah kumpulan perangkat fisik yang dapat membantu Anda untuk memindahkan data sampai dengan ukuran *exabyte* ke dalam dan keluar AWS.

AWS Snow Family terdiri dari AWS Snowcone, AWS Snowball, dan AWS Snowmobile.

Diambil dari AWS Snow Family.

Perangkat-perangkat tersebut menawarkan kapasitas yang berbeda dan sebagian besarnya menyertakan kemampuan komputasi *built-in*. Perangkat Snow Family juga terintegrasi dengan kemampuan keamanan, pemantauan, dan manajemen penyimpanan AWS.

Oke, sekarang mari kita bedah masing-masing perangkat tersebut.

AWS Snowcone

Diambil dari AWS Snow Family.

AWS Snowcone merupakan penawaran yang terbaru di AWS Snow Family. AWS Snowcone adalah perangkat yang kecil, kokoh, dan aman.

AWS Snowcone dapat menampung data sampai dengan ukuran 8 TB (terabyte) dan berisi *edge computing* (sistem komputasi yang dapat melakukan pemrosesan dan analisis data sedekat mungkin ke lokasi yang dibutuhkan).

Perangkat ini memiliki fitur sebagai berikut:

HDD (hard disk drive)	8 TB
vCPU (virtual CPU)	2 vCPU
Memory	4 GB

Lalu, bagaimana cara Anda untuk mendapatkan perangkat AWS Snowcone? Mudah saja, tahapannya adalah

1. pesan melalui AWS Management Console;
2. AWS mengirimkannya kepada Anda;
3. salin data ke perangkat tersebut; dan
4. kirimkan kembali kepada AWS.

Saat perangkat tiba di AWS Region, AWS akan menyalin data ke Amazon S3 bucket yang Anda miliki dan *boom!* Data Anda pun siap untuk digunakan.

Pelanggan AWS biasanya menggunakan perangkat ini untuk mengirimkan informasi yang besarnya dalam jumlah terabyte seperti data analitik, kumpulan video, koleksi gambar, dan bahkan *backups* (cadangan).

Tetapi, bagaimana jika 8 TB itu tidak cukup? Oh, tenang! AWS memiliki produk yang dapat memenuhi kebutuhan tersebut, yaitu AWS Snowball.

AWS Snowball

Diambil dari AWS Snowball Edge Developer Guide: Receiving the Snowball Edge.

AWS Snowball hadir dalam 2 versi: Snowball Edge Storage Optimized dan Snowball Edge Compute Optimized. Mari kita kupas keduanya.

- **Snowball Edge Storage Optimized**
  Selain berfungsi sebagai penyimpanan (storage) lokal dengan kebutuhan kapasitas yang lebih tinggi, perangkat ini sangat ideal untuk migrasi data berskala besar dan alur kerja transfer yang berulang.

  Berikut adalah fitur yang terdapat pada perangkat jenis ini:

HDD (Hard Disk Drive)	80 TB
SSD (Solid State Drive)	1 TB
vCPU (virtual CPU)	40 vCPU
Memory	80 GB

- **Snowball Edge Compute Optimized**
  Jenis ini menyediakan sumber daya komputasi yang kuat untuk kasus penggunaan seperti *machine learning*, analisis data, pemrosesan, dan penyimpanan lokal.

HDD (Hard Disk Drive)	42 TB
SSD (Solid State Drive)	7,68 TB
vCPU (virtual CPU)	52 vCPU
Memory	208 GB
GPU	NVIDIA V100 (opsional)

-

AWS Snowmobile

Diambil dari AWS Snow Family.

AWS Snowmobile adalah layanan transfer data dengan skala *exabyte* yang digunakan untuk memindahkan data dalam jumlah besar ke AWS. Seperti namanya, perangkat ini disimpan di dalam kontainer pengiriman yang kokoh sepanjang 45 kaki dan ditarik oleh truk semi-trailer.

AWS Snowmobile dapat menampung 100 PB (100.000 TB) data, ini membuatnya sangat ideal untuk migrasi secara masif dari data center Anda ke cloud.

Mungkin Anda akan bertanya-tanya, “Bagaimana cara kerjanya?” Oke, begini penjelasannya:

1. Pertama, AWS mengirimkan truk ke lokasi data center Anda.
2. Lalu, Personel AWS akan menghubungkannya ke jaringan lokal Anda.
3. Kemudian, mulailah mentransfer data Anda ke Snowmobile.
4. Selanjutnya, Snowmobile akan dibawa kembali ke AWS untuk diimpor ke Amazon S3.

AWS Snowmobile menggunakan beberapa lapisan keamanan guna melindungi data Anda, di antaranya termasuk personel keamanan khusus, pelacakan GPS, pemantauan alarm, pengawasan video 24/7, dan pengawalan (opsional) dengan kendaraan keamanan pada saat dalam perjalanan.

Catat! Semua perangkat AWS Snow Family dirancang agar aman dan tahan kerusakan saat berada di lokasi Anda atau dalam perjalanan. Semua data yang disimpan akan terenkripsi secara otomatis menggunakan *encryption key* (kunci enkripsi) 256-bit. Anda dapat mengontrol kunci enkripsi menggunakan layanan AWS Key Management Service (AWS KMS).

Bagaimana? Sudah siap migrasi?

Inovasi dengan AWS

Sejauh ini kita telah mempelajari beberapa layanan. Tapi, tahukah Anda? Masih ada banyak hal lain yang bisa Anda lakukan di AWS.

Nah, demi mendorong inovasi dengan AWS, Anda perlu mendeskripsikan kondisi-kondisi berikut terlebih dahulu:

- Keadaan saat ini.
- Keadaan yang diinginkan.
- Masalah yang sedang Anda coba pecahkan.

Oke, jika kondisi di atas sudah jelas, maka sekarang pertimbangkanlah beberapa jalur yang mungkin bisa Anda pilih dalam perjalanan cloud.

Agar lebih mudah memahaminya, simak beberapa contoh kategori berikut:

- ***Serverless applications*** (aplikasi tanpa server)
  Seperti yang sudah kita pelajari, serverless mengacu pada aplikasi yang *tidak* mengharuskan Anda untuk menyiapkan, mengurus, atau mengelola server.

  Anda tidak perlu khawatir akan *fault tolerance* (toleransi terhadap kesalahan) atau *availability* (ketersediaan). AWS akan menanganinya untuk Anda.

  AWS Lambda adalah salah satu contoh layanan yang dapat Anda gunakan untuk menjalankan aplikasi serverless. Dengan AWS Lambda, Anda dapat men-*trigger* (memicu) Lambda function sehingga kode dapat dijalankan tanpa perlu mengelola server. Keren, ‘kan?

  Nah, karena Anda tidak perlu mengelola dan mengoperasikan server, Anda bisa lebih fokus pada produk/solusi yang sedang dikembangkan.

- ***Artificial Intelligence*** (kecerdasan buatan)
  AWS menyajikan variasi layanan yang mendukung artificial intelligence (AI). Di bawah ini adalah beberapa layanan AWS yang termasuk dalam kategori AI:
  - **Amazon Transcribe** : Layanan *speech-to-text* (mengubah ucapan menjadi teks) secara otomatis.
  - **Amazon Comprehend** : Layanan NLP (Natural Language Processing/pemrosesan bahasa alami) untuk mencari *insight* (wawasan) dan hubungan dalam teks.
  - **Amazon Fraud Detector** : Layanan untuk mengidentifikasi aktivitas online yang berpotensi penipuan.
  - **Amazon Lex** : Layanan ini adalah solusi AI siap pakai yang dapat membantu Anda dalam membangun *chatbot* interaktif.

    - **Amazon Textract** : Layanan yang dapat membaca dan memproses teks, tulisan tangan, form, tabel, dan data lainnya secara otomatis dari jutaan halaman dokumen dalam hitungan jam.

- ***Machine learning*** (pemelajaran mesin)
  Anda dapat menggunakan machine learning untuk analisis data, pemecahan masalah yang kompleks, dan memprediksi hasil sebelum terjadi.

  AWS dibangun di atas platform cloud yang paling komprehensif, optimal untuk machine learning dengan komputasi berperforma tinggi, dan tanpa kompromi terhadap keamanan dan analitik.

  Mari kita uraikan beberapa layanan machine learning AWS:
    - **Amazon SageMaker**
      Perkenalkan, Amazon SageMaker. Dengan layanan ini, Anda dapat membangun, melatih, dan menerapkan model machine learning dalam skala besar secara cepat. Bahkan, Anda juga bisa membangun model khusus dengan dukungan semua *open-source framework* (kerangka kerja sumber terbuka) yang populer.
    - **AWS DeepRacer**
      Jika Anda adalah seorang developer yang ingin menangani machine learning sendiri, mengapa tidak mencoba AWS DeepRacer? Anda memiliki kesempatan untuk bereksperimen dengan *reinforcement learning*--salah satu cabang terbaru dari algoritma machine learning--melalui simulator balap 3D berbasis cloud. Wah serunya!

Selain yang sudah disebutkan di atas, masih ada berbagai kategori lainnya. Misal dalam perkara migrasi ke AWS, Anda bisa memiliki infrastruktur berbasis VMware layaknya yang Anda gunakan di on-premise dengan VMware Cloud on AWS.

AWS juga menawarkan teknologi baru, seperti IoT (Internet of Things) yang dapat mengaktifkan perangkat terhubung untuk berkomunikasi di seluruh dunia.

Oh iya, berbicara tentang komunikasi di seluruh dunia, pernahkah Anda ingin memiliki satelit sendiri? Tunggu, ini bukan gurauan. Impian tersebut bisa terwujud dengan menggunakan AWS Ground Station. Anda cukup membayar waktu satelit sesuai dengan kebutuhan.

Bagaimana? Keren-keren, ‘kan? Hal-hal itulah yang membuat AWS menjadi tempat kreasi dan inovasi bagi para pencipta solusi.

Huhh! Oke, oke. Kita bisa menghabiskan waktu berhari-hari hingga mulut berbusa hanya untuk membicarakan semua teknologi baru ini. Serius, *literally* berhari-hari!

Jadi, sepertinya kita sudahi sampai di sini saja pembahasan kali ini. Jika ingin belajar lebih lanjut, kunjungi AWS Training and Certification yang menawarkan kelas pelatihan seputar berbagai teknologi.

Ikhtisar

Di modul ini kita telah belajar tentang migrasi ke AWS dan beberapa layanan inovatif yang dapat digunakan untuk membantu meningkatkan eksplorasi Anda di AWS. Di antaranya:

- **AWS Cloud Adoption Framework (AWS CAF)**
  AWS Cloud Adoption Framework dapat memberikan panduan migrasi dari beberapa perspektif:
  - Business, People, dan Governance untuk perencanaan nonteknis.
  - Platform, Security, dan Operations untuk perencanaan teknis.
- **6 R migrasi**
  Setiap R mewakili berbagai strategi dalam proses migrasi ke cloud, yakni: Rehost, Replatform, Repurchase, Refactor, Retire, dan Retain.
- **AWS Snow Family**
  Untuk menjawab pertanyaan tentang cara memindahkan data dalam jumlah besar tanpa melalui jaringan, AWS memberikan solusi melalui AWS Snow Family yang terdiri dari AWS Snowcone, AWS Snowball, dan AWS Snowmobile.
  Dengan AWS Snow Family, Anda bisa memindahkan data ke perangkat fisik lalu mengirimkannya kembali ke AWS. Kemudian, AWS akan mengunggahnya untuk Anda. Layanan ini berguna untuk menghindari kemungkinan masalah jaringan yang Anda miliki. Bahkan, ia lebih aman daripada menggunakan internet berkecepatan tinggi.

Itulah pembahasan kita pada modul kali ini. Apakah Anda masih semangat? Ayo, bersiaplah untuk melangkah ke materi berikutnya.

Materi Pendukung
Untuk mempelajari lebih lanjut tentang materi yang dibahas pada modul ini, silakan tinjau tautan berikut:

- Migration & Transfer on AWS
- A Process for Mass Migrations to the Cloud
- 6 Strategies for Migrating Applications to the Cloud
- AWS Cloud Adoption Framework
- AWS Fundamentals: Core Concepts
- AWS Cloud Enterprise Strategy Blog
- Modernizing with AWS Blog
- AWS Customer Stories: Data Center Migration

MODUL 9

Pengenalan ke Perjalanan Cloud

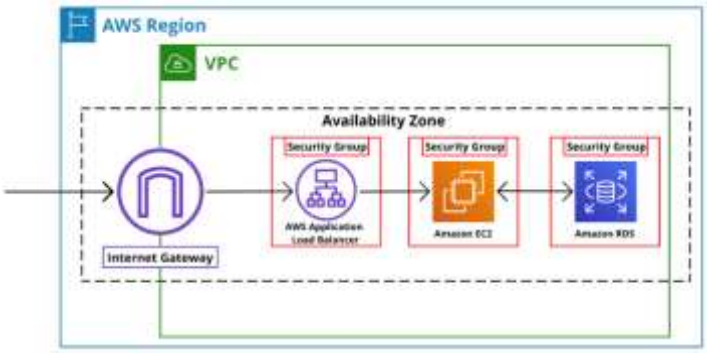
Bagaimana perjalanan Anda dalam menempuh materi-materi di kelas ini? Semoga aman dan mulus ya.

Sejauh ini, kita telah menelaah berbagai layanan AWS. Setiap layanan tersebut dapat menjadi esensi untuk solusi yang akan Anda bangun di AWS.

Selain itu, ada berbagai arsitektur yang bisa Anda buat untuk menyelesaikan beragam masalah. Anda bisa membuatnya dengan cara yang sederhana atau kompleks sekali pun.

Oke, memiliki banyak pilihan memang menarik, tetapi bagaimana cara mengetahui bahwa arsitektur yang dibuat itu bagus?

Untuk menjawabnya, silakan tinjau arsitektur sederhana berikut:

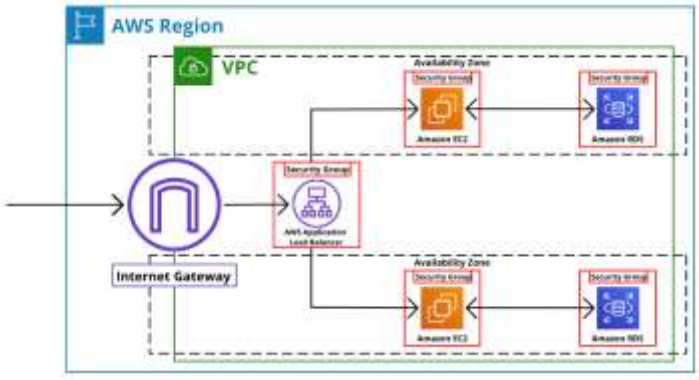


Apakah arsitektur di atas terlihat bagus? Mari kita bedah. Gambar di atas adalah contoh dari *three-tier architecture* atau arsitektur tiga tingkat. Kita memiliki load balancer, instance, dan database.

Apakah Anda bisa menemukan kekurangannya di mana? Oke, coba jawab pertanyaan berikut, “Bagaimana jika Availability Zone (AZ) yang menopang aplikasi Anda mengalami masalah?”

Yup! Mungkin sekarang Anda akan mengangguk-angguk. Dengan arsitektur tersebut, aplikasi Anda bisa berhenti atau tak tersedia jika AZ mengalami masalah.

Terus, bagaimana dong solusinya? Jawabannya ada di arsitektur berikut:



Berbeda dengan arsitektur sebelumnya, sekarang sumber daya telah tereplikasi di seluruh AZ yang mana ini sangat penting untuk menjaga keandalan. Kini, jika salah satu AZ mengalami masalah, aplikasi Anda akan tetap aktif dan berjalan di AZ kedua.

Nah, dari uraian di atas dapat kita ambil pelajaran bahwa menemukan kekurangan pada arsitektur adalah hal yang penting.

Tapi, ingat! Faktanya, Anda akan menghadapi kasus yang tak sesederhana contoh di atas. Untungnya, AWS menghadirkan alat yang dapat membantu Anda untuk membuat arsitektur terbaik, yakni AWS Well-Architected Framework.

AWS Well-Architected Framework adalah layanan untuk mengevaluasi arsitektur yang Anda bangun terhadap keunggulan pada beberapa kategori atau disebut dengan pilar, di antaranya:

- Operational Excellence (Keunggulan Operasional)
- Security (Keamanan)
- Reliability (Keandalan)
- Performance Efficiency (Efisiensi Kinerja)
- Cost Optimization (Pengoptimalan Biaya)

Simpan dulu kopi dan cemilan Anda karena pada materi berikutnya kita akan membahas sesuatu yang cukup serius.

AWS Well-Architected Framework

AWS Well-Architected Framework dirancang untuk membantu Anda memahami bagaimana cara merancang dan mengoperasikan sistem yang andal, aman, efisien, dan hemat biaya di AWS Cloud.

AWS Well-Architected Framework terdiri dari 5 pilar guna memastikan pendekatan yang konsisten untuk meninjau dan merancang arsitektur.

Diambil dari E-learning AWS Cloud Practitioner Essentials.

Mari kita uraikan 5 pilar tersebut:

- **Operational Excellence** (Keunggulan Operasional)
  Pilar ini berfokus untuk menjalankan dan memantau sistem guna memberikan nilai bisnis serta terus meningkatkan proses dan prosedur.

  Misalnya, melakukan *operation as code* (operasi sebagai kode), membuat anotasi dokumentasi, mengantisipasi kegagalan, dan sering memperbaiki prosedur operasi.

- **Security** (Keamanan)
  Seperti yang telah kita pelajari sebelumnya, keamanan adalah prioritas nomor 1 di AWS. Pilar ini akan melindungi informasi, sistem, dan aset Anda sekaligus memberikan nilai bisnis melalui *risk assessment* (penilaian risiko) dan strategi mitigasi.

  Saat mempertimbangkan keamanan arsitektur, implementasikan praktik terbaik berikut:
  - Terapkan keamanan di semua lapisan arsitektur Anda.
  - Lakukan automasi terhadap praktik terbaik keamanan.
  - Lindungi data in-transit dan at rest (sudah kita pelajari di Modul Keamanan).

- **Reliability** (Keandalan)
  Pilar ini mencakup kemampuan sistem untuk memastikan beban kerja melakukan fungsi yang diinginkan dengan benar dan konsisten sesuai harapan.

  Beberapa contohnya adalah seperti berikut:
  - Pemulihan otomatis dari kegagalan infrastruktur atau layanan.
  - *Horizontal scaling*--telah dibahas pada Modul Komputasi di Cloud--untuk meningkatkan ketersediaan beban kerja.

    - Pengujian prosedur pemulihan.

- **Performance Efficiency** (Efisiensi Kinerja)
  Pilar ini berfokus pada penggunaan sumber daya IT dan komputasi secara efisien untuk memenuhi kebutuhan.

  Misalnya, menggunakan arsitektur *serverless* (tanpa server), melakukan eksperimen lebih sering, dan merancang sistem agar dapat mendunia dalam hitungan menit.

- **Cost Optimization** (Pengoptimalan Biaya)
  Pilar ini berfokus untuk mengontrol ke mana uang dibelanjakan guna menghindari biaya yang tak perlu.

  Misalnya, menerapkan manajemen keuangan cloud, menganalisis pengeluaran, dan menggunakan *managed service* (layanan terkelola) untuk mengurangi biaya kepemilikan.

Itulah 5 pilar yang dimiliki oleh AWS Well-Architected Framework. Sebelumnya, untuk mengevaluasi pilar-pilar tersebut, Anda perlu meminta bantuan dari seorang Solutions Architect (perancang solusi teknis di AWS Cloud).

Namun, AWS mendengarkan feedback dari pelanggan dan memutuskan untuk merilis alat *Framework as a self-service* (Framework/Kerangka kerja sebagai layanan mandiri), yakni AWS Well-Architected Tool.

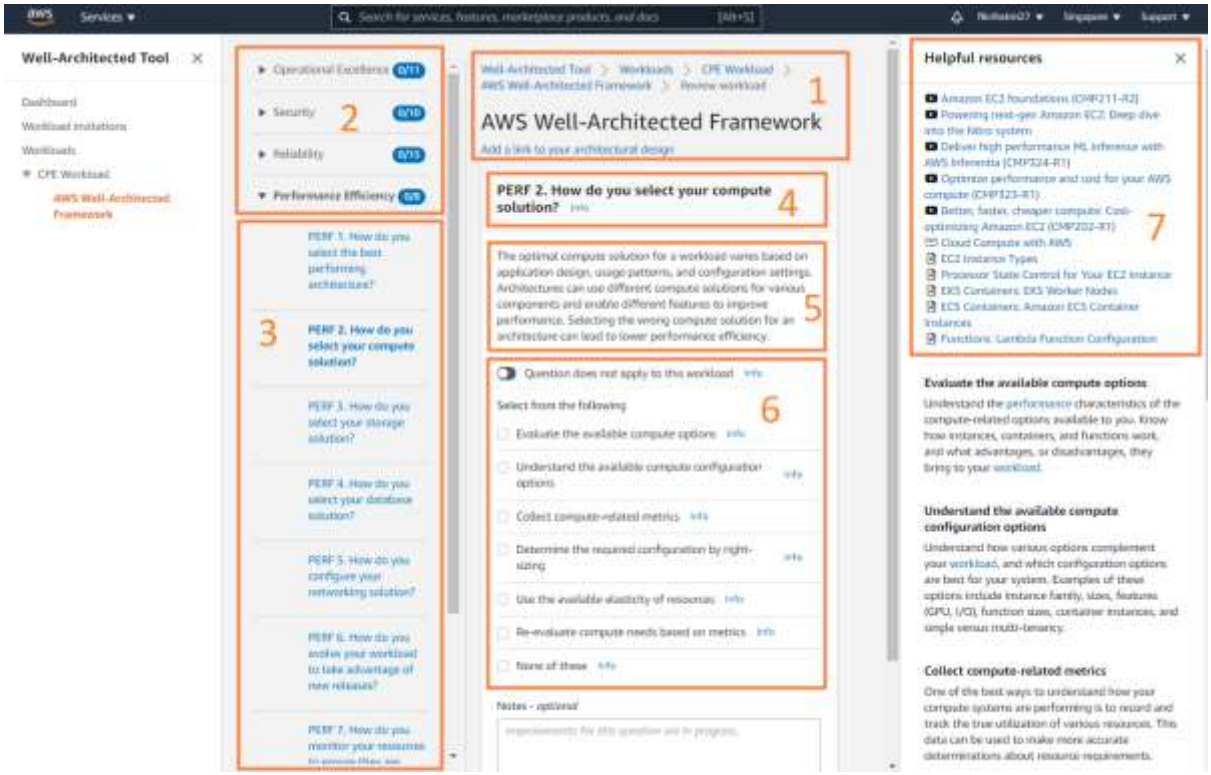
Anda dapat mengakses AWS Well-Architected Tool melalui AWS Management Console. Cara kerjanya pun sederhana, cukup buat *workload* (beban kerja) dan jalankan di akun AWS Anda. Lalu, layanan ini akan menghasilkan laporan dan menunjukkan area mana saja yang harus Anda tangani.

Sama seperti sistem lampu lalu lintas, Well-Architected Tool akan mengindikasikan hasil evaluasi dengan berbagai warna:

- **Hijau** berarti, “Bagus, pertahankan!”
- **Oranye** artinya, “Anda mungkin harus melihat ini karena masih ada yang harus diperbaiki.”
- **Merah** maksudnya, “Oke, Anda harus melihat ini segera karena ada sesuatu yang berisiko!”

Warna-warna di atas menunjukkan potensi masalah pada area yang dideteksi oleh AWS Well-Architected Tool. Ketiga warna tersebut berisikan rencana tentang cara merancang arsitektur menggunakan praktik terbaik yang telah tersedia.

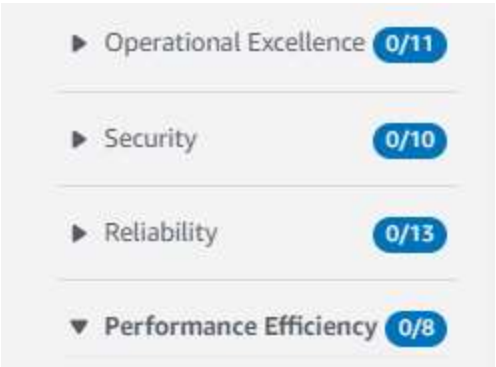
Oke, agar lebih memperjelas, mari kita tilik gambar berikut:



Gambar di atas adalah tampilan dari layanan Well-Architected Framework. Mari kita jabarkan menurut tiap bagiannya.

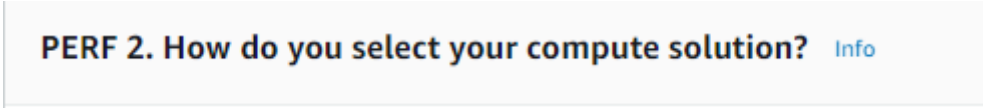
1. **Bagian 1** memperlihatkan nama dari workload (beban kerja) yang kita miliki.

2. **Bagian 2** menunjukkan 5 pilar dari Well-Architected Framework serta menu *drop-down* yang berisi berbagai pertanyaan--lihat bagian selanjutnya.

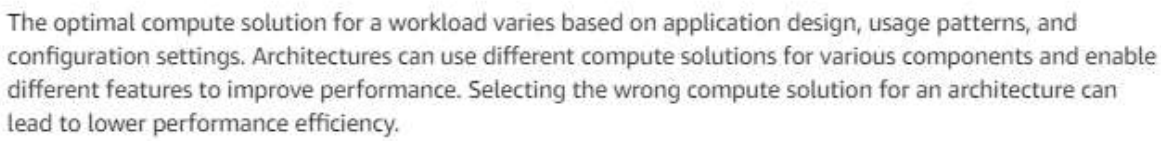


3. **Bagian 3** berisi pertanyaan dari setiap pilar Well-Architected Framework. Contoh di bawah ini adalah beberapa pertanyaan dari pilar *Performance Efficiency*.

4. **Bagian 4** adalah pilar dan inti pertanyaan.



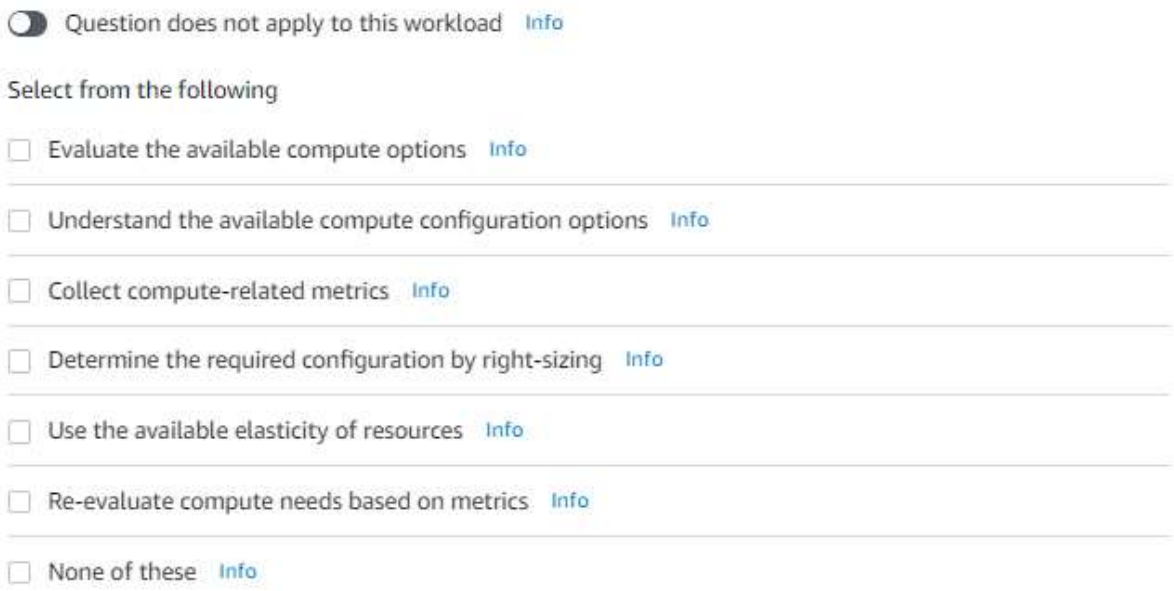
5. **Bagian 5** menunjukkan sedikit latar belakang dan rekomendasi.



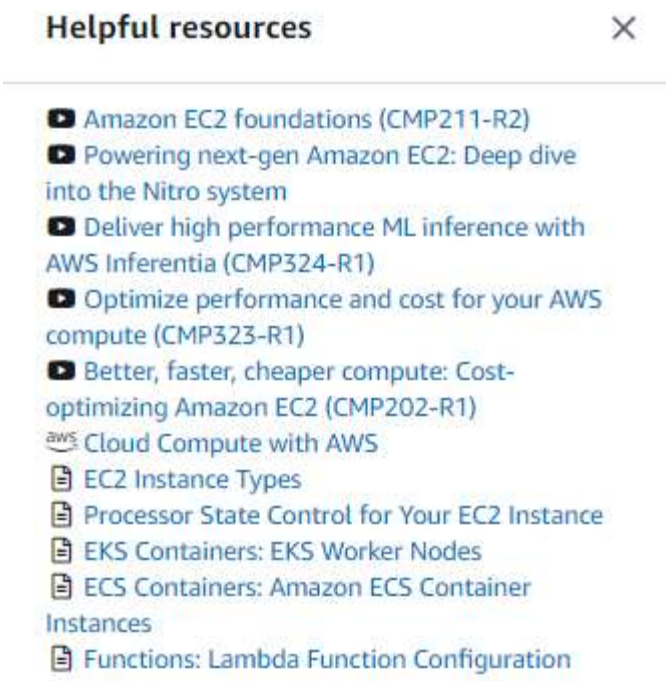
6. **Bagian 6** berisi pilihan yang bisa Anda centang/jawab terkait dengan pertanyaan. Bagian ini cukup penting karena dapat memengaruhi skor Anda secara keseluruhan.

   Di bagian ini hadir juga tombol yang dapat mengesampingkan pertanyaan jika

memang tidak berlaku atau tak bisa diterapkan untuk beban kerja Anda.



7. Terakhir, **Bagian 7** menyajikan video pendek seputar cara untuk menjawab pertanyaan tertentu.



Oke, itulah pembahasan kali ini tentang AWS Well-Architected Framework dan AWS Well-Architected Tool. Semoga Anda menikmati belajar bagaimana cara untuk mengevaluasi beban kerja. Sampai bertemu di modul berikutnya!

Manfaat dari AWS Cloud

Sampai di sini, Anda telah mempelajari semua hal yang perlu diketahui tentang AWS. Siap untuk membangun *startup*?

Oh, tidak, tidak. Perjalanan ke sana masih sangatlah panjang. Meskipun begitu, setidaknya sekarang Anda telah memiliki beberapa wawasan yang baik akan AWS, antara lain:

- **Layanan AWS**
  Anda seharusnya sekarang telah memiliki pemahaman tentang bermacam layanan yang ditawarkan AWS dan bagaimana menggunakannya secara bersamaan. Sehingga, Anda dapat membangun solusi yang fleksibel dan andal untuk bisnis.

- **Terminologi AWS**
  Kini, Anda sudah memahami beberapa terminologi yang digunakan oleh AWS. Tentu ini penting dan bermanfaat jika Anda melakukan perjalanan cloud dengan AWS.

  Misalnya, ketika membaca blog atau dokumentasi AWS, Anda akan mengerti berbagai macam akronim dan frasa yang tertulis di sana.

- **Enam manfaat utama menggunakan AWS Cloud**
  Hal yang harus Anda kuasai saat menggunakan AWS adalah 6 manfaat utama dari AWS Cloud.

  Dengan mengetahui manfaat-manfaat ini, Anda bisa membuat keputusan yang tepat saat mengevaluasi aplikasi dan solusi yang akan dibangun di AWS Cloud.

Oke. Agar lebih memperjelas pembahasan, mari kita uraikan apa saja enam manfaat utama menggunakan AWS Cloud itu.

- **Ubah pengeluaran di muka menjadi pengeluaran variabel**
  Saat membangun data center on-premise (lokal), Anda perlu menginvestasikan uang di awal dengan jumlah yang besar.

  Ada banyak hal yang perlu Anda siapkan, seperti ruangan, perangkat keras, staf untuk memasang dan menyusun perangkat keras tersebut, dan biaya untuk menjaga data center terus berjalan.

  Faktanya, untuk bisnis berskala menengah hingga besar, biaya yang harus keluar bisa mencapai ratusan ribu atau bahkan jutaan dolar. Wah! Jumlah yang fantastis, bukan?

  Apa pun penggunaannya, Anda harus mengeluarkan biaya yang tetap

alias *fixed cost* untuk data center tersebut.

Nah, tentu ini akan berbeda saat menggunakan AWS. Tagihan Anda akan bervariasi dari bulan ke bulan tergantung pada pemakaian sumber daya. Hal yang keren tentang ini adalah Anda tidak perlu mengeluarkan biaya hingga 1 juta dolar untuk mulai membangun lingkungan AWS.

Lalu, bagaimana jika tagihan yang muncul ternyata keluar dari anggaran yang telah dialokasikan?

Oh, tenang! Anda dapat menghemat dengan berbagai cara, seperti mematikan instance yang tidak digunakan, menghapus sumber daya lama yang tak lagi dipakai, mengoptimalkan aplikasi, atau menggunakan AWS Trusted Advisor untuk membantu Anda memberikan rekomendasi terkait penghematan biaya.

Jadi, walaupun tagihan Anda tinggi di bulan ini, Anda bisa melakukan penghematan agar tagihan bulan depan bisa turun.

- **Manfaatkan skala ekonomi yang masif**
  AWS telah membangun kapasitas data center dalam jumlah besar di seluruh dunia.

  Perlu Anda ketahui, AWS itu ahli dalam membangun data center yang efisien. AWS bisa membeli perangkat keras dengan harga yang lebih murah karena jumlahnya yang sangat banyak. Lalu, perangkat tersebut diinstal dan dijalankan dengan cara yang efisien.

  Karena faktor-faktor itulah, Anda mendapatkan biaya variabel yang lebih rendah dibandingkan dengan menjalankan data center sendiri.

- **Berhenti menebak kapasitas**
  Saat membangun data center on-premise, Anda mesti memperkirakan berapa banyak kapasitas yang diperlukan untuk menjalankan bisnis.

  Katakanlah Anda mengestimasi bahwa bisnis Anda akan memiliki basis pengguna sebanyak 10 juta orang selama tiga tahun ke depan. Anda pun membeli sejumlah perangkat keras yang mampu mendukung pertumbuhan tersebut dari waktu ke waktu.

  Namun seiring waktu berjalan, ternyata Anda hanya memiliki sekitar 500.000 pengguna. Waduh! Ini jumlah yang jauh lebih rendah dari perkiraan, bukan?

  Lebih buruknya lagi, Anda harus berkutat dengan biaya dan perangkat keras yang bukan main banyaknya. Estimasi Anda terlalu berlebihan.

Bagaimana kalau sebaliknya? Bagaimana jika Anda memperkirakan kapasitas terlalu rendah?

Oh, itu akan sama buruknya. Anda harus menambah kapasitas data center demi menangani beban kerja yang lebih tinggi sebelum pelanggan pergi. Tentu proses tersebut butuh waktu yang lama dan terlampau sulit untuk bisa ditangani.

Bagaimanapun juga, menebak-nebak kapasitas bisa menjadi masalah jika estimasi Anda tidak tepat.

Nah, dengan AWS, Anda tidak perlu menebak kapasitas. Melainkan, langsung saja buat sumber daya yang Anda butuhkan. Kemudian, gunakan mekanisme penyesuaian kapasitas dengan *scaling up* atau *scaling down* berdasarkan kebutuhan.

Scaling dengan AWS hanya memerlukan beberapa menit saja. Tak seperti di on-premise yang memakan waktu hingga berminggu-minggu bahkan berbulan-bulan.

- **Tingkatkan kecepatan dan ketangkasan**
  AWS memudahkan Anda untuk mencoba hal-hal baru. Anda dapat menjalankan *test environment* (lingkungan pengujian) dan bereksperimen untuk menemukan cara-cara baru dalam proses pemecahan masalah. Jika pendekatan tersebut tidak berhasil, Anda bisa menghapus sumber daya tersebut.

  Data center on-premise tak menawarkan fleksibilitas semacam ini. Jika ingin menjalankan eksperimen yang membutuhkan server baru, Anda harus membeli dan menginstalnya terlebih dahulu. Ribet, ‘kan?

  Nah, di AWS, hanya diperlukan beberapa menit saja untuk membuat server baru. Kemampuan semacam ini dapat membantu Anda untuk merilis aplikasi lebih cepat ke pasaran.

- **Hentikan biaya pengelolaan dan pemeliharaan data center**
  Di AWS, Anda tak perlu menghabiskan banyak uang dan waktu untuk menjalankan data center, AWS-lah yang menangani pekerjaan tersebut.

  Anda hanya perlu fokus terhadap apa yang membuat bisnis Anda berharga ketimbang berkutat menjalankan perangkat keras di data center.

- **Mendunia dalam hitungan menit**
  Mari kita buat cerita. Misalkan Anda memiliki perusahaan yang berbasis di Singapura dan ingin memperluas operasinya ke Jerman.

  Pertanyaannya, bagaimana cara agar Anda bisa melayani pelanggan di negara Jerman dengan baik?

  Tentu Anda memerlukan data center di negara itu. Tak hanya itu, Anda juga membutuhkan staf yang bertugas untuk menjalankan dan mengoperasikan data center tersebut. Tapi, itu akan rumit, repot, dan butuh waktu yang sangat lama.

  Nah, dengan AWS, Anda dapat mereplikasi arsitektur ke suatu wilayah di negara Jerman. Bahkan, Anda dapat melakukannya secara otomatis menggunakan layanan AWS CloudFormation--telah kita bahas sebelumnya.

  Lebih menariknya lagi, proses tersebut dapat selesai hanya dalam waktu beberapa menit saja.

Oke. Itulah 6 manfaat menggunakan AWS Cloud. Perjalanan cloud kita belum usai di sini. Mari melangkah ke modul berikutnya.

Ikhtisar

Sampailah kita di penghujung dari Modul Perjalanan Cloud ini. Eits! Dengan berakhirnya modul ini bukan berarti perjalanan cloud Anda selesai sampai di sini ya. Lanjutkan perjalanan Anda ke modul berikutnya!

Pada modul ini, kita telah belajar berbagai hal, mari kita uraikan.

- Lima pilar dari AWS Well-Architected Framework, di antaranya:
  - Operational excellence (Keunggulan Operasional)
  - Security (Keamanan)
  - Reliability (Keandalan)
  - Performance efficiency (Efisiensi Kinerja)
  - Cost optimization (Pengoptimalan Biaya)
- Enam manfaat dari komputasi cloud, antara lain:
  - Ubah pengeluaran di muka menjadi pengeluaran variabel.
  - Manfaatkan masifnya skala ekonomi.
  - Berhenti menebak kapasitas.
  - Tingkatkan kecepatan dan ketangkasan.
  - Hentikan biaya pengelolaan dan pemeliharaan data center.
  - Mendunia dalam hitungan menit.

Perjalanan kita di kelas ini hampir berakhir. Pastikan Anda masih semangat ya. Yuk kita meluncur ke modul berikutnya!

Materi Pendukung
Untuk mempelajari lebih lanjut tentang materi yang ada di modul ini, Anda bisa mengunjungi tautan berikut:

- AWS Well-Architected
- Whitepaper: AWS Well-Architected Framework
- AWS Architecture Center
- Six Advantages of Cloud Computing
- AWS Architecture Blog