**Glosarium**

**Artificial Intelligence**

Kemampuan komputer untuk bertindak seperti manusia.

**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.

**Cache**

Cache (diucapkan "cash"), menyimpan informasi yang baru saja digunakan sehingga dapat diakses dengan cepat di lain waktu.

**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.

**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.

**Entitas**

Satu objek unik di dunia nyata. Ia mengacu pada individu, organisasi, produk, atau komponen sistem.

**Exabyte**

Exabyte adalah 1018 atau 1.000.000.000.000.000.000 byte. 1 exabyte sama dengan 1.000 petabyte.

**Hard Drive**

Hard drive adalah tempat menyimpan semua data Anda. Data tersebut disimpan 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.

**Instance**

Instance adalah server virtual di AWS Cloud.

**Latensi**

Waktu yang diperlukan untuk mengirim dan menerima data.

**Machine Learning**

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

**On-premise**

Penyimpanan dan pemeliharaan data di server lokal atau pribadi.

**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 1015 atau 1.000.000.000.000.000 byte. 1 petabyte sama dengan 1.000 terabyte.

**Sumber Daya**

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

**Terabyte**

Terabyte adalah 1012 atau 1.000.000.000.000 byte. 1 terabyte sama dengan 1.000 gigabyte.

**Throughput**

Jumlah data yang dapat dikirim dalam waktu tertentu.

**Web Hosting**

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

**Daftar Referensi**

[1] Cisco. “What Is a Data Center.” Cisco.com. <https://www.cisco.com/c/en/us/solutions/data-center-virtualization/what-is-a-data-center.html> (diakses pada 1 November 2020)

[2] H. Hsu. “How the Metaphor of “the Cloud” Changed Our Attitude Toward the Internet.” Newyorker.com. <https://www.newyorker.com/books/page-turner/how-the-metaphor-of-the-cloud-changed-our-attitude-toward-the-internet> (diakses pada 30 Oktober 2020)

[3] AWS Training & Certification. (2020). AWS Cloud Practitioner Essentials. [Online]. Tersedia: <https://www.aws.training/Details/eLearning?id=60697>

**Pengantar ke Amazon Web Services**

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.

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

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?**

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.
* Menjelaskan 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.

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]](https://www.cisco.com/c/en/us/solutions/data-center-virtualization/what-is-a-data-center.html).

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.

**Komputasi Cloud**

* 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.

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.

**Model Penerapan untuk Komputasi Cloud**

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.

**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.

**Ikhtisar**

Anggaplah modul ini sebagai kilas balik untuk mempertajam pemahaman Anda. Mari kita uraikan:

* Pertama, kita telah berkenalan dengan skenario kedai kopi yang akan terus dipakai di setiap modul pada kelas ini.
* Kita juga sudah belajar tentang komputasi cloud.
* Lalu, kita telah menelaah setiap model penerapan untuk komputasi cloud.
* Terakhir, kita telah menilik manfaat dari komputasi cloud.

Dengan ringkasan tersebut, diharapkan Anda dapat memahami semua materi yang telah disampaikan. Yuk lanjut ke modul berikutnya!

**Materi Pendukung**

Untuk mempelajari lebih lanjut tentang konsep yang dipaparkan sedari tadi, Anda dapat meninjau beberapa sumber berikut:

* [AWS glossary](https://docs.aws.amazon.com/general/latest/gr/glos-chap.html)
* [Whitepaper: Overview of Amazon Web Services](https://d0.awsstatic.com/whitepapers/aws-overview.pdf)
* [AWS Fundamentals: Overview](https://aws.amazon.com/getting-started/fundamentals-overview/)
* [Apa itu Komputasi Cloud](https://aws.amazon.com/id/what-is-cloud-computing/)
* [Jenis-Jenis Komputasi Cloud](https://aws.amazon.com/id/types-of-cloud-computing/)
* [Apa itu AWS](https://aws.amazon.com/id/what-is-aws/)

**Daftar Referensi**

[1] Cisco. “What Is a Data Center.” Cisco.com. <https://www.cisco.com/c/en/us/solutions/data-center-virtualization/what-is-a-data-center.html> (diakses pada 1 November 2020)

[2] H. Hsu. “How the Metaphor of “the Cloud” Changed Our Attitude Toward the Internet.” Newyorker.com. <https://www.newyorker.com/books/page-turner/how-the-metaphor-of-the-cloud-changed-our-attitude-toward-the-internet> (diakses pada 30 Oktober 2020)

[3] AWS Training & Certification. (2020). AWS Cloud Practitioner Essentials. [Online]. Tersedia: <https://www.aws.training/Details/eLearning?id=60697>

[**https://youtu.be/Za4zQnFKHiA?si=ZY03NqQ5PkmPH6MF&embeds\_referring\_euri=https%3A%2F%2Fwww.dicoding.com%2F&source\_ve\_path=MTY0NTA2**](https://youtu.be/Za4zQnFKHiA?si=ZY03NqQ5PkmPH6MF&embeds_referring_euri=https%3A%2F%2Fwww.dicoding.com%2F&source_ve_path=MTY0NTA2)

Tujuan Pembelajaran

Di modul ini, kita akan belajar bagaimana cara:

* Menjelaskan manfaat Amazon EC2 secara dasar.
* Mengidentifikasi perbedaan dari tipe Amazon EC2 instance.
* Membedakan berbagai opsi penagihan untuk Amazon EC2.
* Merangkum manfaat dari Amazon EC2 Auto Scaling.
* Meringkas manfaat dari Elastic Load Balancing.
* Memberikan contoh penggunaan Elastic Load Balancing.
* Menjelaskan perbedaan antara Amazon Simple Notification Service (Amazon SNS) dan Amazon Simple Queue Service (Amazon SQS).
* Memaparkan opsi komputasi tambahan di AWS.

**Pengenalan ke Amazon Elastic Compute Cloud (Amazon EC2)**

**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.

**Tipe Instance Amazon EC2**

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.

**Harga Amazon EC2**

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.
* 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 atau3 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*.

**Penyesuaian Kapasitas Amazon EC2**

**Amazon EC2 Auto Scaling**

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.

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.

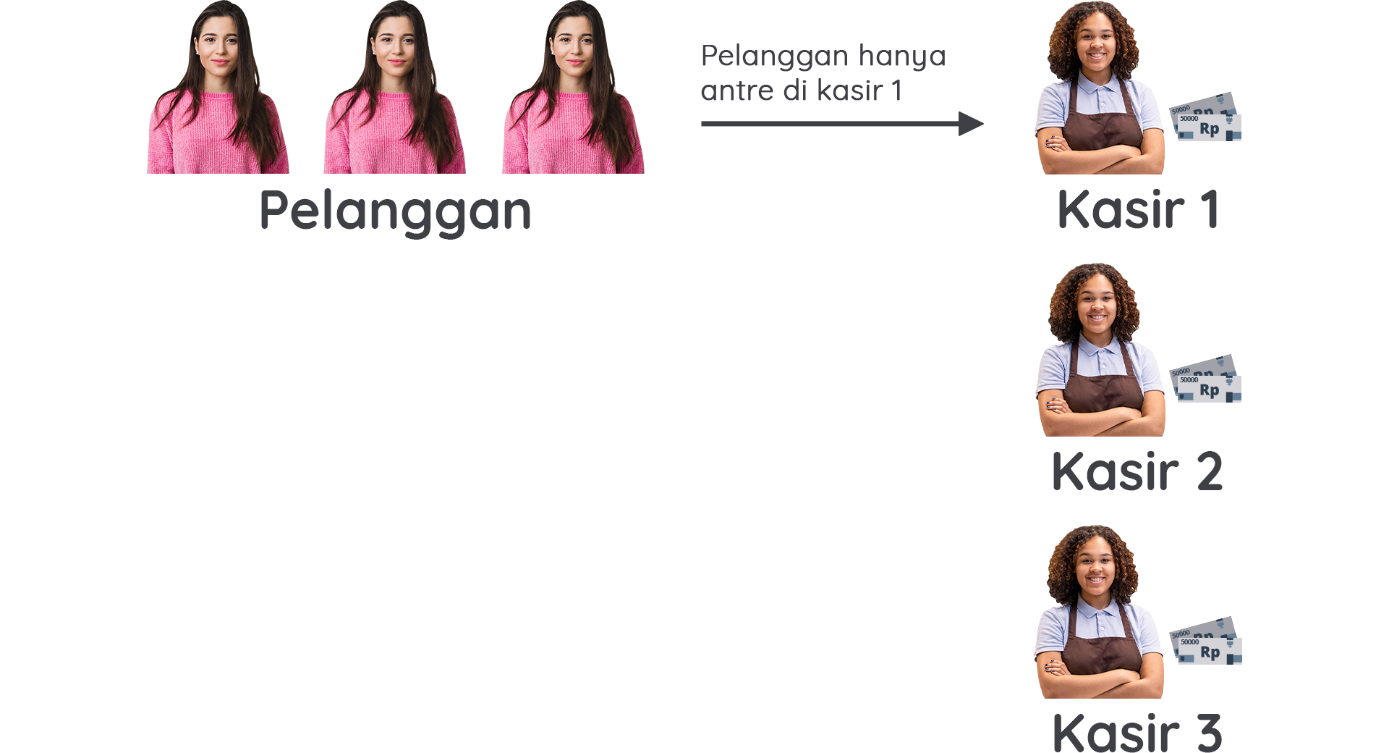
**Auto Scaling Group**

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

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.

**Mengarahkan Traffic dengan Elastic Load Balancing**



*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**

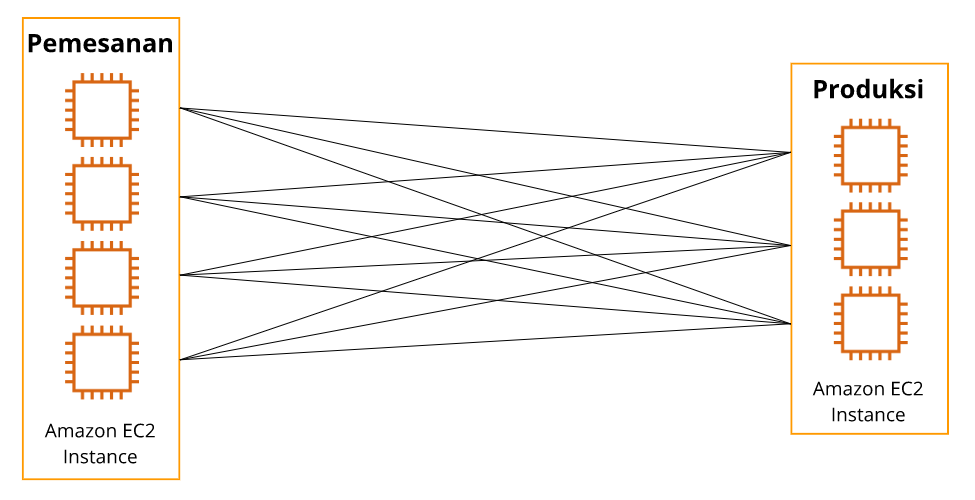
AWS memiliki layanan *load balancer* yang berkinerja tinggi, hemat biaya, *highly available* (sangat tersedia), dan dapat diskalakan secara otomatis. Tak usah Anda menginstal, mengelola, memperbarui, melakukan *scaling*, menangani kegagalan, dan ketersediaan layanannya. AWS yang mengurus itu semua.

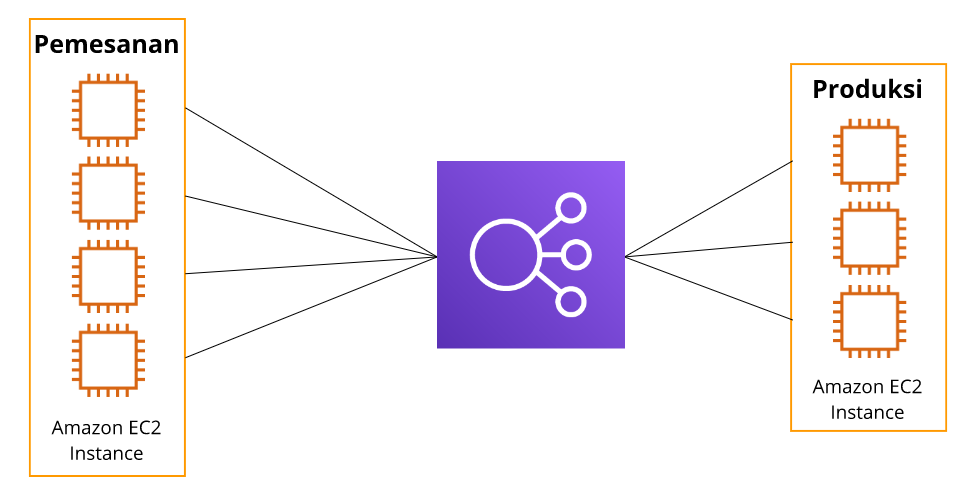
Perkenalkan Elastic Load Balancing (ELB), yaitu layanan AWS yang secara otomatis mendistribusikan traffic aplikasi yang masuk ke berbagai sumber daya, seperti Amazon EC2 instance.

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.





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.

**Messaging dan Queueing**

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.

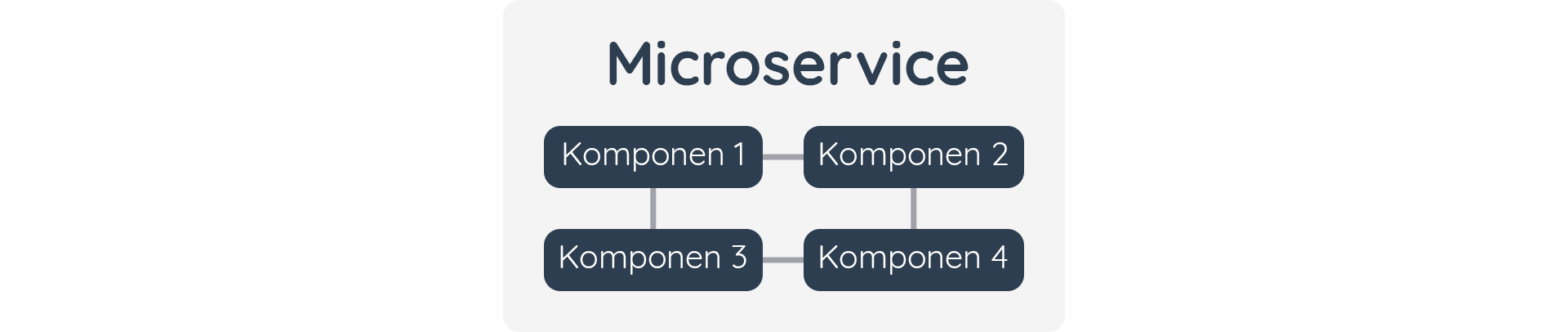
Saat pesanan berikutnya masuk, prosesnya berulang. Proses ini akan bekerja dengan baik selama kasir dan barista selaras. Tetapi, apa yang akan terjadi jika kasir ingin menyerahkan pesanan pelanggan namun barista sedang istirahat atau sibuk dengan pesanan lain?





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.



**Amazon Simple Queue Service (Amazon SQS)**

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 sulit memahaminya, bayangkan saja sebuah pesansebagai 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*.

**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.

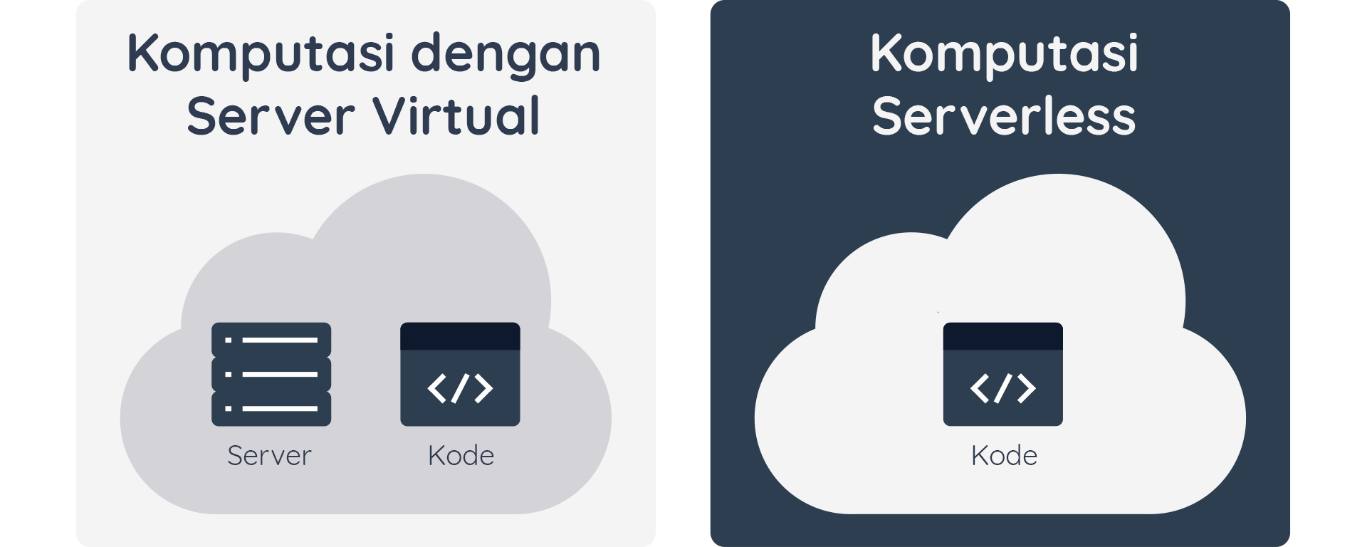
**Layanan Komputasi Tambahan**

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).

**Komputasi Serverless**



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 functiontersebut 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.

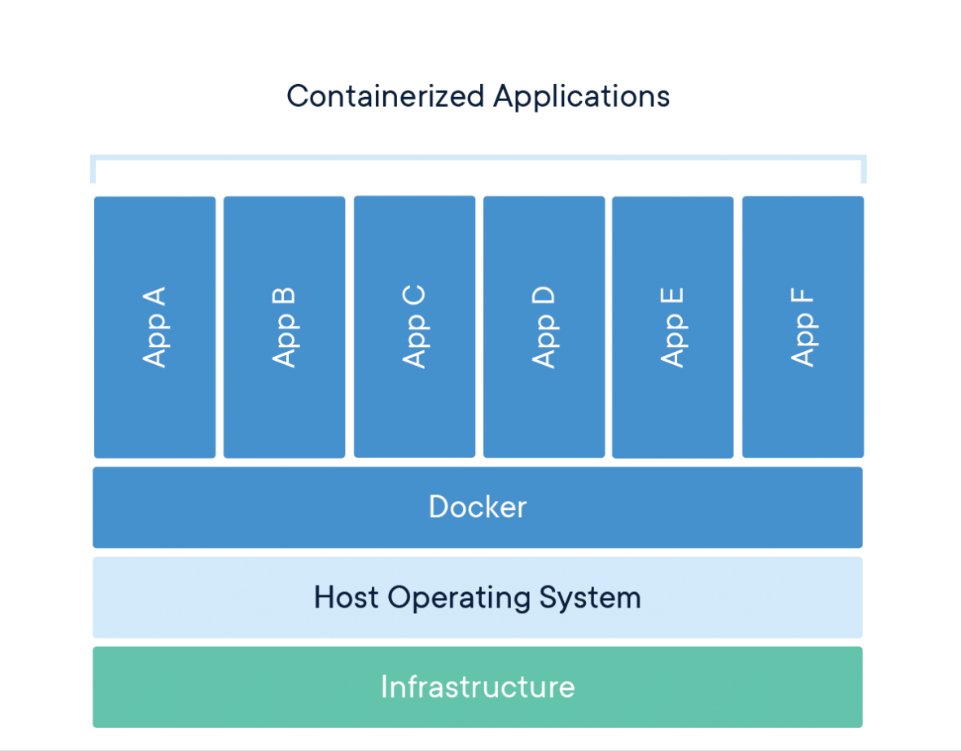
1. Unggah kode Anda ke AWS Lambda.
2. Konfigurasikan kode Anda agar terpicu (*trigger*) dari sumber kejadian, seperti layanan AWS, aplikasi seluler, atau HTTP *endpoint* (titik akhir HTTP).
3. Kode berjalan hanya ketika mendapat *trigger*.
4. Cukup bayar sesuai waktu komputasi yang Anda gunakan. Misalnya, 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.

**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.

**AWS 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!