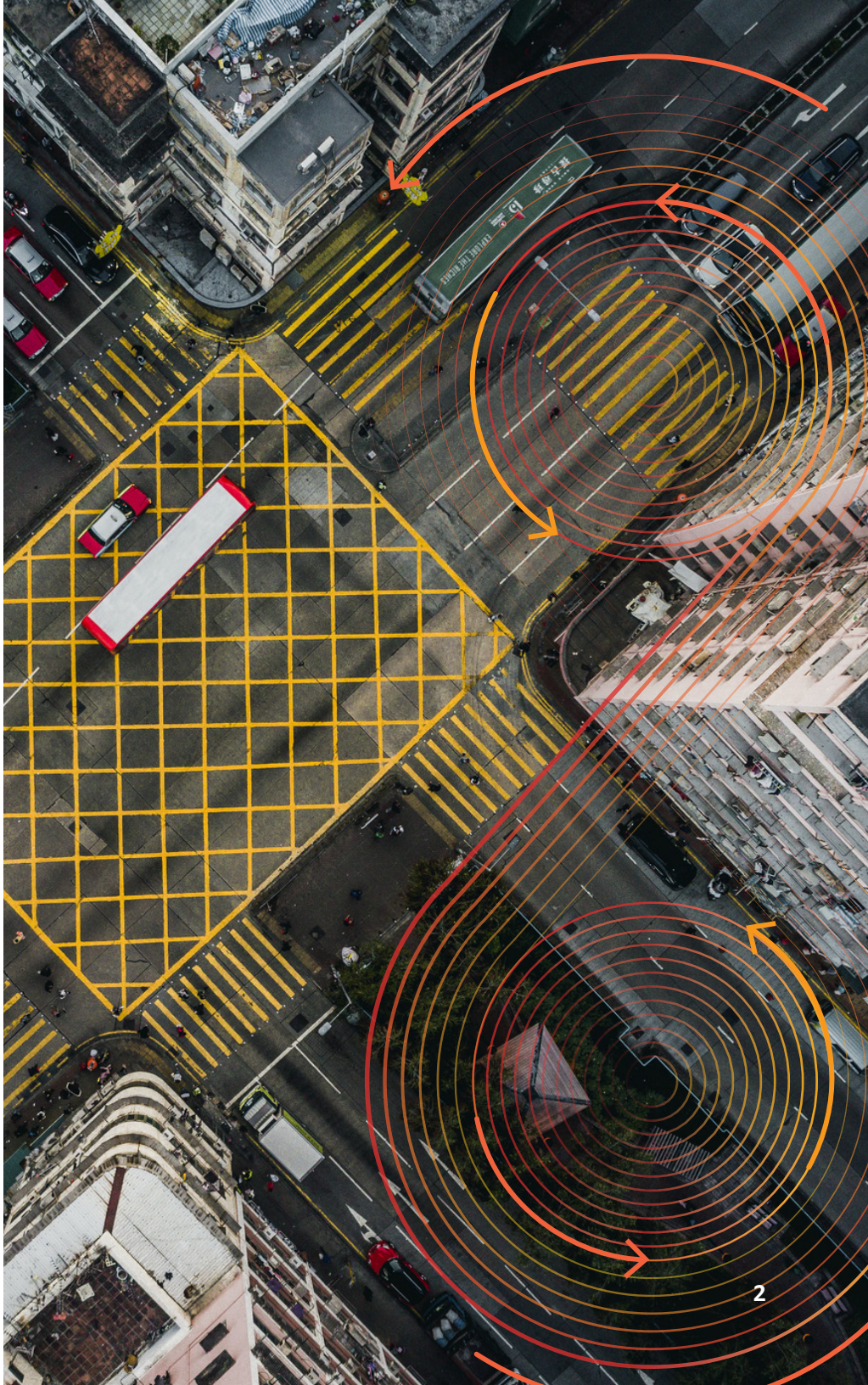

aws modern apps
Bangun aplikasi modern di AWS
Kelola lebih sedikit. Bangun dengan cepat. Inovasi lebih banyak.

Aplikasi modern mengubah cara Anda menghasilkan nilai pelanggan

Dalam beberapa tahun ke depan, organisasi-organisasi akan membangun lebih dari 500 juta aplikasi baru, lebih banyak dari jumlah yang dikembangkan dalam 40 tahun sebelumnya secara keseluruhan.[1] Banyak yang mengalami kesulitan untuk membangunnya karena mereka tidak dapat menemukan keseimbangan antara mengelola teknologi dan menghadirkan fitur baru.

Meskipun ***cloud*** menjanjikan ketangkasan, hal ini tidak terjadi secara otomatis. Saat organisasi ingin mempercepat inovasi, mendapatkan hasil maksimal dari data mereka, dan menciptakan pengalaman pelanggan baru, mereka perlu memodernisasikan cara mereka dalam membangun dan mengoperasikan aplikasi. Aplikasi modern dibangun dengan kombinasi pola arsitektur modular, model operasional nirserver, dan proses developer yang tangkas.

Dalam ***eBook*** ini, kami akan memandu Anda dalam tiga jalur yang akan membantu menetapkan landasan untuk pengembangan aplikasi modern di organisasi Anda sendiri. Kami juga akan menelusuri bagaimana pengembangan aplikasi modern dengan AWS dapat membantu organisasi Anda berinovasi, mengurangi biaya, mempercepat kecepatan masuk pasar (TTM), dan meningkatkan keandalan.



aws modern apps

[1] *IDC PlanScape: Connected Cloud Architectures to Enable the Future of Digital Infrastructure, Oktober 2020*

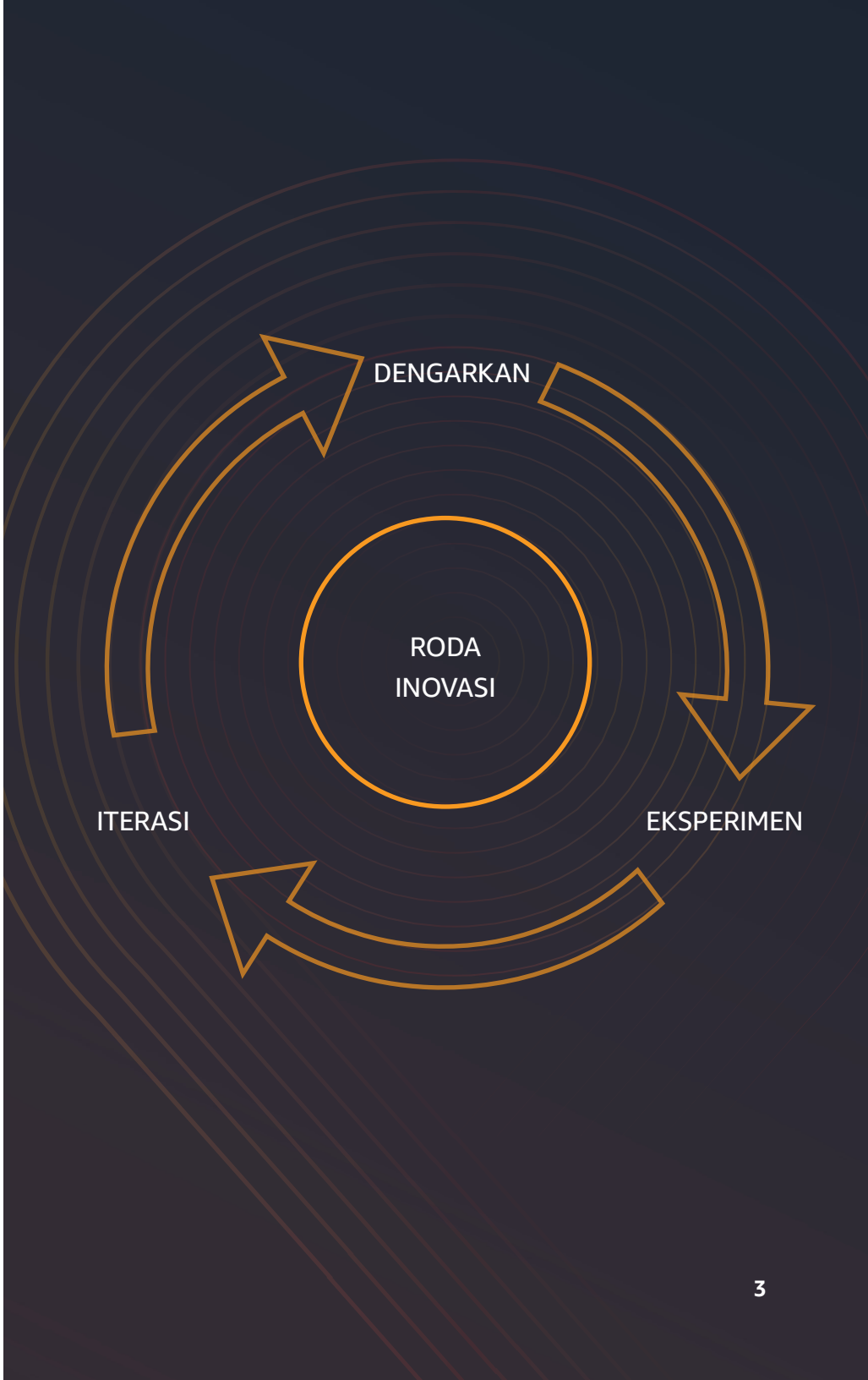
Inovasi berarti mendengarkan pelanggan

Dalam Vision Report terbaru, *Digital Rewrites the Rules of Business*, Forrester Research mendefinisikan pola pikir yang berorientasi ke pelanggan yang dimiliki oleh inovator digital. Misi inti dari ***disruptor*** modern ini adalah untuk:

"…Memanfaatkan aset dan ekosistem digital untuk meningkatkan hasil pelanggan secara berkelanjutan dan meningkatkan keunggulan operasional secara berkesinambungan…dengan menerapkan pemikiran digital terhadap pengalaman pelanggan, operasi, ekosistem, dan inovasi."

Berfokus pada pelanggan berarti mengambil keputusan bisnis dengan melakukan penelusuran mundur dari sudut pandang pelanggan. Hal tersebut berarti terus mengembangkan produk dan layanan untuk memberikan hasil lebih baik yang memuaskan pelanggan. Hal tersebut juga berarti mendengarkan hal yang benar-benar penting bagi pelanggan sehingga Anda dapat terus melakukan inovasi dan iterasi untuk mereka. Hal ini disebut "roda inovasi."

Gagasan dasarnya adalah bahwa penggerak segala inovasi dimulai dengan permintaan pelanggan, melakukan peningkatan dengan menerapkan umpan balik pelanggan, dan berulang secara konstan (dan menguntungkan) hingga permintaan tersebut berubah dan seluruh siklusnya dimulai kembali. Semakin cepat tim Anda memutar roda inovasi, semakin baik Anda dapat membangun aplikasi modern dan semakin unggul Anda dari pesaing.



Dengan membangun aplikasi modern di AWS, Anda akan memasuki pasar lebih cepat. Dengan mempercepat siklus bangun dan rilis serta melepaskan beban operasional, developer dapat membangun fitur baru dengan cepat. Anda akan meningkatkan inovasi dengan arsitektur modular yang memungkinkan tim bereksperimen dengan setiap komponen aplikasi tanpa menimbulkan risiko untuk aplikasi secara keseluruhan. Dengan mengotomatiskan prosedur pengujian dan pemantauan di setiap tahap siklus hidup pengembangan, Anda akan meningkatkan keandalan. Selain itu, Anda akan meningkatkan total biaya kepemilikan (TCO) dengan model harga bayar sesuai nilai yang mengurangi biaya penyediaan berlebih atau membayar sumber daya yang tidak terpakai.

Untuk membangun aplikasi modern, Anda mungkin perlu mempertimbangkan kembali landasan dalam membangunnya. Meskipun pergeseran arsitektur ini mungkin akan tampak drastis pada level organisasi, prosesnya tidak akan seburuk yang dibayangkan. Banyak organisasi memutuskan untuk membangun aplikasi modern baru di ***cloud*** hanya bermodalkan keyakinan, tetapi banyak lainnya mengambil pendekatan campuran, sering kali menjalankan proses tim demi tim dan beban kerja demi beban kerja, serta mengambil langkah satu per satu berdasarkan peluang.

50%

teknologi informasi dan komunikasi diprediksi akan secara langsung dialokasikan untuk transformasi digital paling lambat tahun 2023

67%

eksekutif percaya bahwa mereka harus menambah kecepatan agar tetap kompetitif

90%

aplikasi baru diprediksi menjadi *cloud native* paling lambat tahun 2025

Melalui pengalaman kami membangun aplikasi untuk Amazon.com dan melayani jutaan pelanggan AWS, kami telah mengamati tiga jalur yang dapat diambil organisasi untuk mewujudkan visi modernisasi aplikasi mereka, sehingga menghasilkan nilai bagi bisnis dalam prosesnya.

1. Platform ulang ke layanan kontainer terkelola. Organisasi yang sudah menjalankan kontainer ***on-premise*** atau mempertimbangkan untuk memindahkan aplikasi ke kontainer dapat memplatform ulang beban kerja tersebut ke layanan kontainer di AWS untuk menyederhanakan operasi dan mengurangi biaya tambahan (***overhead***) manajemen, seperti orkestrasi dan penyediaan infrastruktur.

2. Bangun aplikasi baru di arsitektur nirserver. Saat organisasi membangun aplikasi atau fitur baru, kami menyarankan agar mereka menggunakan teknologi nirserver dan basis data yang dibuat khusus untuk memaksimalkan ketangkasan, serta alat pengembangan canggih untuk mempercepat pengembangan.

3. Bertransformasi ke model Dev+Ops Modern. Untuk menciptakan perubahan budaya guna membangun aplikasi modern dalam skala besar, organisasi dapat memanfaatkan layanan dan alat DevOps sambil mempertahankan standar keamanan dan tata kelola yang tinggi.

Kami akan menelusuri setiap jalur secara lebih mendetail, dengan menunjukkan bagaimana masing-masing jalur ini dapat membantu meningkatkan ketangkasan, menurunkan biaya, dan membangun aplikasi yang lebih baik dalam mendukung kesuksesan bisnis. Meskipun Anda dapat memodernisasikan aplikasi dari titik awal mana pun, hasilnya harus tetap sama: aplikasi yang aman, andal, dapat diskalakan, dan tersedia secara cepat bagi pelanggan dan partner sejak awal.

Tiga jalur menuju pengembangan aplikasi modern

Pengembangan aplikasi modern adalah pendekatan yang kuat untuk merancang, membangun, dan mengelola perangkat lunak di ***cloud***. Pendekatan yang sudah terbukti ini meningkatkan ketangkasan tim pengembangan Anda serta keandalan dan keamanan aplikasi Anda, sehingga memungkinkan Anda untuk membangun dan merilis produk yang lebih baik secara lebih cepat. Dari pengalaman kami membantu segala jenis organisasi membangun aplikasi, kami telah mengidentifikasi tiga pilar solusi pengembangan aplikasi modern untuk membantu Anda dalam perjalanan menuju modernisasi.

Jalur menuju aplikasi modern

1. **Platform ulang ke layanan kontainer terkelola**
2. **Bangun aplikasi baru yang aman di arsitektur nirserver**
3. **Bertransformasi ke model Dev+Ops Modern**

Platform ulang ke layanan kontainer terkelola

Kontainer adalah cara yang ringan dan portabel untuk menjalankan dan men-***deploy*** aplikasi. Kontainerisasi terhadap aplikasi yang ada sering kali merupakan langkah pertama dalam perjalanan modernisasi organisasi. Jika Anda mempertimbangkan untuk memindahkan aplikasi Anda ke kontainer, Anda akan mendapatkan manfaat dari platform ulang (***re-platform***) beban kerja tersebut ke layanan terkelola AWS seperti Amazon Elastic Kubernetes Service (Amazon EKS) atau Amazon Elastic Container Service (Amazon ECS) dengan AWS Fargate. Layanan kontainer terkelola akan membantu mengurangi beban operasional sambil meningkatkan skalabilitas, keandalan, keamanan, dan ketersediaan. Dengan layanan kontainer terkelola di AWS, Anda tidak perlu lagi mengurus pengelolaan kontainer. Sebagai gantinya, Anda dapat memfokuskan sumber daya pada edukasi untuk meningkatkan keterampilan yang Anda perlukan untuk mengembangkan aplikasi modern dengan komputasi nirserver.

Vanguard®

> ***"AWS Fargate Spot telah mengurangi biaya unit kami dan memperkuat kasus bisnis kami untuk bermigrasi ke AWS. Kami menghasilkan lebih banyak nilai untuk pengeluaran kami setiap bulannya dengan optimasi ini. Pengembalian nilai kepada pemegang saham kami melalui peningkatan efisiensi adalah inti dari misi kami di seluruh perusahaan."***
>
> – Tim Treston, Manajer Senior (***Senior Manager***), Kantor Bisnis ***Cloud***, Vanguard

Sejak memulai migrasi ke AWS pada tahun 2015, Vanguard Group telah memperoleh manfaat yang signifikan. Misalnya, penggunaan layanan AWS membuat departemen IT Vanguard tidak perlu lagi mengelola server. Hasilnya, developer memiliki lebih banyak waktu untuk membangun layanan mikro baru yang inovatif dan menyempurnakan aplikasi saat ini, sehingga meningkatkan kecepatan masuk pasar (TTM) Vanguard dari tiga bulan menjadi 24 jam.

Namun, kecepatan masuk pasar (TTM) bukan satu-satunya keuntungan yang telah diperoleh perusahaan ini. Vanguard memilih untuk menggunakan Amazon Elastic Container Service (Amazon ECS), layanan orkestrasi kontainer yang terkelola penuh, bersama AWS Fargate. AWS Fargate adalah layanan komputasi nirserver yang menghilangkan kebutuhan untuk menyediakan dan mengelola Amazon Elastic Compute Cloud (Amazon EC2), sebuah layanan web yang menyediakan kapasitas komputasi yang aman dan dapat diubah ukurannya di ***cloud***. Selain itu, dengan memanfaatkan opsi pembelian baru untuk AWS Fargate, perusahaan layanan keuangan tersebut mengurangi biaya unitnya sebesar 50 persen.

Baca kisah lengkapnya »

Modernisasi pengembangan aplikasi adalah adopsi layanan, praktik, dan strategi yang memungkinkan developer membangun aplikasi yang lebih tangkas. Dengan kecepatan dan keandalan infrastruktur modern, developer juga dapat menghadirkan aplikasi aman yang dapat diskalakan dari prototipe ke jutaan pengguna secara otomatis, sehingga mereka dapat berinovasi dan merespons perubahan dengan lebih cepat.

Banyak aplikasi modern dibangun dengan mengutamakan pendekatan nirserver (***serverless-first***), sebuah strategi yang memprioritaskan adopsi layanan nirserver agar pelanggan dapat meningkatkan ketangkasan di seluruh tumpukan (***stack***) aplikasi. Dengan teknologi nirserver, Anda tidak harus lagi mengelola server fisik dan Anda akan mendapatkan manfaat dari penskalaan otomatis, ketersediaan tinggi bawaan, dan model penagihan bayar sesuai nilai. Daripada mengurus pengelolaan dan pengoperasian server atau sistem ***runtime***, Anda dapat berfokus pada inovasi produk sambil meningkatkan kecepatan masuk pasar (TTM).

Selain itu, alat dan layanan web dan seluler ***front-end*** dapat dibangun di atas AWS. Keandalan infrastruktur ini membantu banyak merek menghadirkan aplikasi yang aman dan memiliki ketersediaan tinggi yang dapat diskalakan secara otomatis di seluruh dunia.

Pertimbangan utama untuk membuat aplikasi modern yang dapat diskalakan

Pola arsitektur: layanan mikro

Meskipun aplikasi monolitik Anda mungkin mudah dikelola sekarang, tantangan sering muncul seiring Anda bertumbuh, termasuk cara mendistribusikan kepemilikan aplikasi ke seluruh tim Anda. Anda dapat membangun budaya kepemilikan yang kuat tetapi kesulitan untuk menaikkan skala jika arsitektur aplikasi Anda memiliki dependensi yang erat sehingga menghambat tim untuk mengambil kepemilikan atas produk akhir. Itulah alasan kami merekomendasikan untuk membangun arsitektur layanan mikro untuk aplikasi yang tumbuh dan berubah secara cepat. Layanan mikro adalah ekspresi arsitektural budaya kepemilikan—Layanan ini membagi aplikasi yang kompleks secara apik menjadi beberapa komponen yang dapat dimiliki dan dijalankan oleh satu tim secara terpisah.

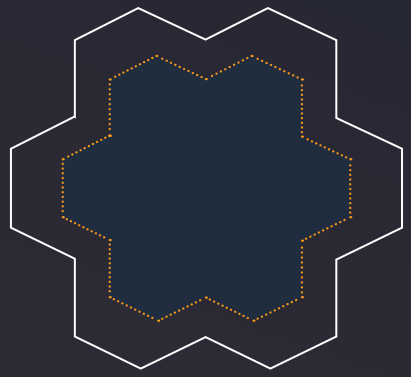
Pada aplikasi monolit, banyak developer mendorong perubahan melalui alur rilis bersama, yang menimbulkan hambatan di banyak titik siklus hidup. Selama pengembangan, rekayasawan perlu mengoordinasikan perubahan yang mereka lakukan agar memastikan mereka tidak merusak kode rekayasawan lain. Untuk memutakhirkan (***upgrade***) pustaka bersama guna memanfaatkan fitur baru, Anda perlu meyakinkan semua orang untuk memutakhirkan pada saat yang sama—Itu pekerjaan yang sulit! Selain itu, jika Anda ingin segera melakukan perbaikan penting pada fitur, Anda masih harus menggabungkannya dengan perubahan yang masih di tengah proses.

Setelah pengembangan, Anda juga menghadapi beban tambahan ketika Anda mendorong perubahan melalui alur pengiriman. Bahkan ketika membuat perubahan satu baris di satu bagian kecil kode, rekayasawan perlu mengoordinasikan terlebih dahulu perubahan yang mereka lakukan, menggabungkan kode, menyelesaikan konflik dalam beberapa rilis, membangun kembali keseluruhan aplikasi, menjalankan semua rangkaian pengujian, dan men-***deploy*** ulang sekali lagi.

Dengan arsitektur layanan mikro, aplikasi dibentuk dari komponen-komponen independen yang menjalankan setiap proses aplikasi sebagai suatu layanan. Layanan dibangun untuk kemampuan bisnis, dan setiap layanan melakukan satu fungsi. Karena dijalankan secara independen dan dikelola oleh satu tim pengembangan, setiap layanan dapat diperbarui, di-***deploy***, dan diskalakan untuk memenuhi permintaan terhadap fungsi spesifik suatu aplikasi. Sebagai contoh, keranjang belanja ***online*** dapat digunakan oleh lebih banyak pengguna saat ada obral. Layanan mikro mengomunikasikan data satu sama lain melalui antarmuka yang jelas, menggunakan ***stream***, peristiwa, atau API yang ringan. Pelanggan kami semakin mengandalkan arsitektur yang didorong peristiwa—yang di dalamnya, tindakan dipicu sebagai respons terhadap perubahan data—untuk meningkatkan skalabilitas dan ketangguhan serta mengurangi biaya.

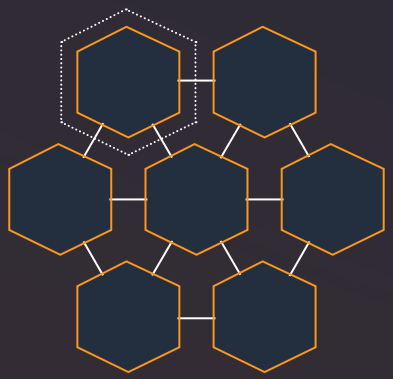
SEGALA HAL VS. SATU HAL: DUA JENIS APLIKASI

Aplikasi monolit



- Melakukan segalanya
- Satu aplikasi
- Harus men-*deploy* keseluruhan aplikasi
- Satu basis data
- Disusun berdasarkan lapisan teknologi
- Status di setiap instans sistem *runtime*
- Satu tumpukan teknologi untuk keseluruhan aplikasi

Layanan mikro



- Melakukan satu hal
- Layanan fungsi minimal
- Di-*deploy* secara terpisah, berinteraksi secara bersama
- Masing-masing memiliki penyimpanan data sendiri
- Disusun berdasarkan kemampuan bisnis
- Status dieksternalisasi
- Pilihan teknologi untuk setiap layanan mikro

Centrica adalah perusahaan energi dan layanan multinasional asal Inggris yang ingin menurunkan biaya dan meningkatkan ketangkasan dengan mengubah arsitektur aplikasinya. Perusahaan ini memilih untuk memfaktor ulang (***refactor***) arsitektur layanan mikro dan mengadopsi strategi nirserver untuk mencapai sasaran tersebut.

Untuk mengubah organisasinya, Centrica menyiapkan grup kerja nirserver bersama beberapa tim perwakilan dan mulai membangun proyek percontohan bersama-sama. Berdasarkan keberhasilan tersebut, tim lain di organisasi mereka sekarang juga telah mengadopsi pendekatan tersebut—teknologi nirserver kini menjadi hal yang umum di seluruh organisasi.

Dengan pendekatan nirserver, Centrica kini dapat melihat dan menanggapi masalah pelanggan secara waktu nyata, sesuatu yang belum pernah dapat mereka lakukan sebelumnya.

Tonton video »

Model operasional nirserver

Sebisa mungkin nirserver

Saat pola arsitektur dan proses pengiriman perangkat lunak Anda berubah, Anda mungkin ingin mengadopsi model operasional yang memungkinkan Anda untuk mengalihkan aktivitas yang bukan merupakan kompetensi inti bisnis Anda. Untuk memperoleh ketangkasan yang dapat mewujudkan inovasi cepat, sebaiknya bangun arsitektur layanan mikro, dengan mengoperasikan dan men-***deploy*** perangkat lunak menggunakan otomatisasi untuk hal-hal seperti pemantauan, penyediaan, manajemen biaya, ***deployment***, serta keamanan dan tata kelola aplikasi. Dengan memilih strategi pengutamaan nirserver (***serverless-first***)—selalu memilih teknologi nirserver jika memungkinkan—Anda dapat memaksimalkan manfaat operasional dari AWS.

Dengan model operasional nirserver, Anda dapat membangun dan menjalankan aplikasi dan layanan tanpa menyediakan dan mengelola server. Hal ini menghapuskan pengelolaan server, menyediakan penskalaan yang fleksibel, memungkinkan Anda untuk membayar hanya untuk hal yang Anda anggap penting, dan mengotomatiskan ketersediaan tinggi. Model ini memungkinkan Anda membangun dan mengelola aspek-aspek aplikasi Anda yang mewujudkan nilai pelanggan tanpa harus khawatir tentang perincian yang mendasarinya.

Baik Anda membangun aplikasi yang benar-benar baru maupun memigrasikan aplikasi warisan, membangun dengan ***primitive*** nirserver untuk komputasi, data, dan integrasi akan memungkinkan Anda mendapatkan manfaat dari ketangkasan terbaik yang ditawarkan ***cloud***.

Bagaimana kami mendefinisikan nirserver di AWS?

Saat kami menyebut "nirserver", maksud kami adalah penghapusan beban tidak terdiferensasi, yaitu operasi server. Ini adalah perbedaan yang penting karena hal ini memungkinkan Anda untuk fokus pada pembangunan aplikasi alih-alih pengelolaan dan penskalaan infrastruktur untuk mendukung aplikasi. Empat prinsip model operasional nirserver adalah:

1. **Tidak ada pengelolaan server** – Tidak perlu menyediakan atau memelihara server apa pun. Tidak ada perangkat lunak atau sistem *runtime* yang harus diinstal, dipelihara, atau dikelola.

2. **Penskalaan yang fleksibel** – Aplikasi Anda dapat diskalakan secara otomatis atau disesuaikan kapasitasnya dengan mengubah unit konsumsi (misalnya *throughput*, memori) dan bukan unit masing-masing server.

3. **Bayar sesuai nilai** – Alih-alih membayar unit server, bayar untuk hal yang bernilai bagi Anda—*throughput* yang konsisten atau durasi pelaksanaan.

4. **Ketersediaan tinggi yang terotomatisasi** – Teknologi nirserver menyediakan ketersediaan dan toleransi kesalahan bawaan. Anda tidak perlu menentukan arsitektur untuk kemampuan tersebut karena layanan yang menjalankan aplikasi akan menyediakannya secara *default*.

Model operasional nirserver memiliki fungsi ideal untuk perusahaan dengan pertumbuhan tinggi yang ingin berinovasi dengan cepat. Teknologi nirserver memungkinkan tim untuk bergerak lebih cepat lagi dan mempertahankan fokus pada aktivitas yang membedakan bisnis Anda, sehingga Anda dapat mempercepat roda inovasi Anda.

Memanfaatkan AWS Lambda dan layanan kontainer terkelola di AWS

Dengan munculnya kontainer dan komputasi nirserver, instans bukan lagi satu-satunya pilihan komputasi ***cloud*** Anda. Pemilihan komputasi yang optimal untuk aplikasi modern Anda dimulai dengan menjawab beberapa pertanyaan. Apakah infrastruktur yang dikelola sendiri meningkatkan hasil bisnis Anda? Apakah Anda memiliki keahlian untuk melakukannya? Dan apakah usaha ekstra pada akhirnya akan mendorong nilai?

Semakin banyak pelanggan memilih untuk melepaskan beban manajemen server dengan mengadopsi layanan kontainer seperti Amazon ECS dan Amazon EKS atau layanan komputasi nirserver yang didorong peristiwa seperti AWS Lambda.

Pada kenyataannya, sebagian besar pelanggan menggunakan kombinasi keduanya. Sekitar 80 persen pelanggan kontainer AWS juga telah mengadopsi AWS Lambda.[2] Penggunaan kedua opsi ini dapat memberikan manfaat, termasuk layanan terkelola penuh yang memiliki integrasi mendalam dengan infrastruktur AWS, dukungan untuk berbagai kasus penggunaan, abstraksi dari kompleksitas, dan ekosistem partner yang luas.

[2] *Datadog State of Serverless, 2020*

Jadi bagaimana cara Anda menentukan pilihan?

Pelanggan memilih AWS Lambda saat mereka memiliki tim yang terutama berfokus pada penulisan kode dan tidak memiliki batasan terkait instans atau platform kontainer yang ada. AWS Lambda menawarkan abstraksi maksimum dari infrastruktur agar pelanggan dapat merilis dengan sangat cepat, dan itulah sebabnya aplikasi baru sangat cocok untuk AWS Lambda.

Pelanggan sering memilih kontainer saat mereka memiliki investasi kontainer yang sudah ada, preferensi Kubernetes sumber terbuka, atau persyaratan khusus untuk mengelola atau mengonfigurasi infrastruktur. Kontainer adalah cara paling populer untuk memaketkan kode dan merupakan pilihan tepat untuk memodernisasikan aplikasi warisan.

	AWS Mengelola	Pelanggan Mengelola	
AWS Lambda Fungsi nirserver	Integrasi sumber data Perangkat keras fisik, perangkat lunak, jaringan, fasilitas Penyediaan	Kode aplikasi	
AWS Fargate Kontainer nirserver	Orkestrasi kontainer, penyediaan Penskalaan klaster Perangkat keras fisik, OS/***kernel host***, jaringan, fasilitas	Kode aplikasi Integrasi sumber data Konfigurasi dan pembaruan keamanan, konfigurasi jaringan	Tugas manajemen
Amazon ECS/ Amazon EKS Pengelolaan kontainer sebagai layanan	Aspek kontrol orkestrasi kontainer Perangkat keras fisik, jaringan, fasilitas	Kode aplikasi Integrasi sumber data Konfigurasi dan pembaruan keamanan, konfigurasi jaringan, ***firewall***	Klaster kerja Tugas manajemen
Amazon EC2 Infrastruktur sebagai layanan	Perangkat keras fisik, jaringan, fasilitas	Kode aplikasi Integrasi sumber data Penskalaan Tugas manajemen	Konfigurasi dan pembaruan keamanan, konfigurasi jaringan Penyediaan, pengelolaan penskalaan dan ***patching*** server

AbiBird®

> *"Kami percaya teknologi kami yang berjalan di AWS memungkinkan semakin banyak lansia untuk hidup mandiri di rumah—tempat mereka merasa lebih bahagia."*
>
> – Robin Mysell, CEO, ATF Services dan AbiBird

AbiBird adalah divisi yang sepenuhnya dimiliki oleh ATF Services, sebuah grup perusahaan asal Australia dengan 350 karyawan dan kontraktor, serta 60 cabang di Australia dan Selandia Baru. AbiBird menawarkan layanan yang terdiri atas sensor inframerah untuk rumah, yang membantu memantau aktivitas penghuni lansia melalui aplikasi berbasis ponsel pintar.

AbiBird sebelumnya berjalan di Microsoft Azure dan mendapati bahwa mereka terlalu sering mengajukan tiket dukungan ke penyedia ***cloud***-nya untuk menjaga layanannya tetap berjalan. Perusahaan ini membutuhkan stabilitas dan skalabilitas yang lebih tinggi. AbiBird beralih ke AWS dan sekarang menggunakan kombinasi layanan komputasi berdasarkan kebutuhannya. Mereka menggunakan AWS Lambda di ***backend*** karena kemudahan penggunaan dan skalabilitasnya, serta karena sifat layanannya yang terkelola.

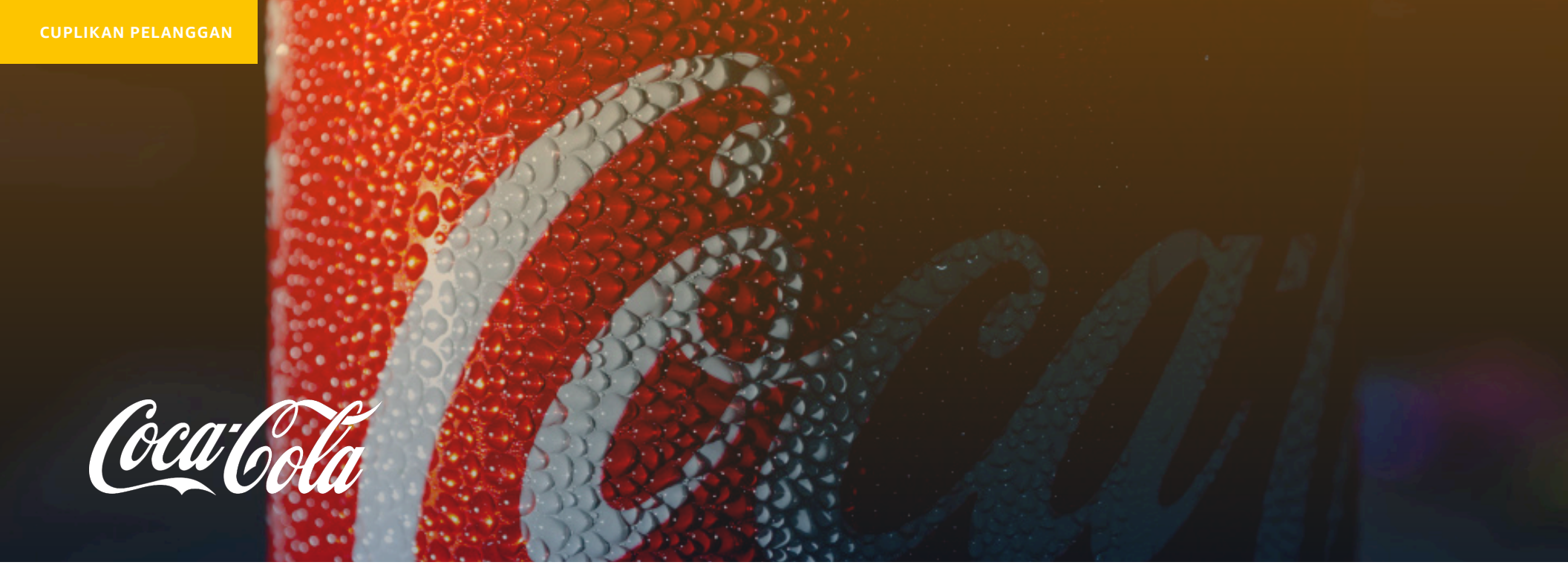
AbiBird juga menggunakan layanan kontainer di AWS, yang meng-***hosting*** API publik di Amazon ECS dan memanfaatkan AWS Fargate untuk menjalankan kontainernya tanpa perlu repot mengelola armada mesin virtualnya sendiri.

Sejak meluncurkan sistem ini di AWS pada tahun 2019, AbiBird belum pernah perlu mengajukan satu pun tiket dukungan, yang memungkinkan perusahaan ini berjalan secara efektif dengan jumlah dukungan beban tambahan (***overhead***) yang minimal.

Baca kisah lengkapnya »

Salah satu merek makanan paling ikonik di Amerika, Taco Bell memiliki lebih dari 7.000 restoran di AS. Selama pandemi COVID-19, Taco Bell perlu bergerak cepat untuk memenuhi permintaan layanan pesan antar dari konsumen. Menurut Vadim Parizher, wakil presiden (***vice president***) rekayasa dan analitik, Taco Bell menjalankan hampir semua infrastrukturnya di Amazon Web Services (AWS) dan menggunakan teknologi nirserver di AWS agar lebih sedikit berfokus dalam mengelola server dan lebih banyak berfokus dalam membangun logika bisnis serta transformasi data untuk mengirimkan menu dan informasi restoran secara waktu nyata kepada partner layanan pesan antarnya. "Kami memiliki menu yang sangat rumit dan harus dibagikan ke berbagai saluran digital. Teknologi nirserver sangat cocok dengan model tersebut," kata Parizher. Dengan menggunakan layanan nirserver, Taco Bell dapat menghemat biaya di muka, memulai dari skala kecil, dan hanya membayar sesuai kebutuhan, lalu menskalakan secara otomatis saat perusahaan ini menggunakan lebih banyak layanan.

Tonton video »



Ketika COVID-19 merebak, kebiasaan konsumen benar-benar berubah dengan cepat. Coca-Cola memberikan respons cepat dengan menyediakan pengalaman mengeluarkan minuman tanpa sentuhan untuk melengkapi dispenser minuman Freestyle-nya yang inovatif. Coca-Cola memilih untuk membangun aplikasi dengan AWS Lambda, dan sebagai hasilnya, timnya dapat berfokus pada aplikasi tanpa perlu mengurus keamanan, latensi, atau skalabilitas. Karena dengan AWS Lambda, semua hal tersebut merupakan fungsi bawaan. Aplikasi baru tersebut diluncurkan hanya dalam 100 hari, dan sekarang lebih dari 52.000 mesin memiliki kemampuan nirsentuh.

Tonton kisah lengkapnya »

"Latensi rendah sangat penting untuk pengalaman pengguna, itulah sebabnya kami menggunakan solusi nirserver di AWS."

– Michael Connor, Kepala Arsitek (***Chief Architect***), Pusat Inovasi Peralatan Freestyle Coca-Cola

Dev+Ops Modern adalah kombinasi filosofi budaya kerja, praktik, dan alat yang memungkinkan organisasi mengembangkan perangkat lunak dengan cepat dan aman, merilisnya ke tahap produksi, serta mempertahankan ketersediaan dan performa targetnya.

AWS telah mengidentifikasi kumpulan praktik umum yang diterima secara luas yang, jika diadopsi, akan memberikan mekanisme untuk membangun organisasi DevOps berperforma tinggi. Pendekatan ini menggunakan ide sederhana—peningkatan berkelanjutan—dan menerapkannya ke semua hal dalam siklus hidup DevOps, mulai dari perencanaan dan penulisan kode hingga ***deployment*** dan pemantauan.

Kami menyebut pendekatan ini sebagai Dev+Ops Modern, dan pendekatan ini berpusat pada upaya mendekatkan developer dan operasi dengan membagikan tugas operasional seperti kepatuhan, kemampuan pengamatan, ketahanan, dan infrastruktur secara lebih awal ke dalam proses pengembangan dan menyempurnakannya dengan kecerdasan buatan dan ***machine learning*** (AI/ML).

Ketangkasan developer: abstraksi, otomatisasi, dan standardisasi

Arsitektur layanan mikro membuat tim pengembangan menjadi tangkas dan memungkinkan mereka bergerak lebih cepat, yang berarti Anda membangun lebih banyak hal yang perlu dirilis — hebat! Namun, Anda tidak akan memberikan fitur baru kepada pelanggan dengan lebih cepat jika proses bangun dan rilis Anda tidak mengikuti kecepatan tim Anda. Proses pengembangan dan alur rilis tradisional terutama diperlambat oleh proses manual dan kode khusus. Kode khusus pada akhirnya merupakan beban jangka panjang karena hal ini menghadirkan kemungkinan kesalahan dan memerlukan pemeliharaan jangka panjang. Langkah manual—dari perubahan kode dan permintaan pembangunan hingga pengujian dan ***deployment***—adalah hambatan terbesar untuk kecepatan rilis. Solusinya membutuhkan abstraksi, otomatisasi, dan standardisasi.

Untuk mempercepat proses pengembangan, lakukan abstraksi pada sebanyak mungkin kode, khususnya baris kode logika non-bisnis, yang diperlukan untuk mengembangkan dan menghasilkan aplikasi yang siap diproduksi. Salah satu cara untuk melakukannya adalah dengan menggunakan kerangka kerja dan alat yang mengurangi kompleksitas penyediaan dan konfigurasi sumber daya. Hal ini memberi developer kemampuan untuk bergerak dengan cepat, sambil juga menerapkan praktik terbaik untuk keamanan, privasi, keandalan, performa, kemampuan pengamatan, dan kemampuan perluasan di seluruh proses pengembangan. Kerangka kerja pengembangan memberi Anda kepercayaan diri bahwa arsitektur Anda akan mendukung pertumbuhan bisnis Anda dalam jangka panjang.

Dengan mendefinisikan proses pengiriman perangkat lunak melalui penggunaan templat praktik terbaik, Anda dapat menyediakan standar untuk pemodelan dan penyediaan seluruh sumber daya infrastruktur di lingkungan ***cloud***. Templat "infrastruktur sebagai kode" ini membantu tim untuk memulai dengan posisi terbaik karena templat tersebut menyediakan seluruh tumpukan teknologi untuk aplikasi melalui kode, dan bukan menggunakan proses manual.

Melalui otomatisasi, Anda dapat membuat gerakan yang dapat diulang yang mempercepat siklus pengiriman perangkat lunak Anda. Otomatisasi alur rilis melalui integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD) akan membantu tim merilis kode berkualitas tinggi dengan lebih cepat dan sering. Tim yang mempraktikkan CI/CD mengirimkan lebih banyak kode, melakukannya dengan lebih cepat, dan merespons masalah dengan lebih cepat. Bahkan, menurut Laporan State of DevOps 2020 dari Puppet, tim yang menerapkan praktik ini memiliki tingkat kegagalan yang lima kali lebih rendah, tingkat ***commit*** ke ***deploy*** yang 440 kali lebih cepat, dan tingkat ***deployment*** yang 46 kali lebih sering.[3] Terutamanya, tim yang mempraktikkan CI/CD menghabiskan 44 persen lebih banyak waktunya untuk membuat fitur dan kode baru daripada mengelola proses dan alat.

Alur CI/CD telah menjadi landasan baru untuk membangun aplikasi modern. Di Amazon, kami mulai menggunakan CI/CD untuk meningkatkan kecepatan rilis, dan hasilnya drastis—kami telah menyelesaikan jutaan ***deployment*** dalam setahun dan menjadi lebih cepat lagi setiap tahunnya. Untuk membantu perusahaan memperoleh manfaat dari pengalaman kami, kami membuat rangkaian alat developer berdasarkan alat yang kami gunakan secara internal, sehingga pelanggan kami dapat mengirimkan kode dengan lebih cepat.

Sedikit perincian selengkapnya:

Integrasi berkelanjutan – (CI) adalah praktik pengembangan perangkat lunak yang memungkinkan developer menggabungkan perubahan kode mereka ke dalam repositori pusat secara rutin, lalu menjalankan pembangunan dan pengujian terotomatisasi. Integrasi berkelanjutan paling sering merujuk pada tahap pembangunan atau integrasi dalam proses rilis perangkat lunak serta terdiri dari komponen otomatisasi (misalnya layanan CI atau pembangunan) dan komponen budaya kerja (misalnya belajar agar mampu melakukan integrasi dengan sering).

Pengiriman berkelanjutan – (CD) adalah praktik pengembangan perangkat lunak yang memungkinkan perubahan kode disiapkan secara otomatis untuk rilis ke tahap produksi. Pengiriman berkelanjutan mengembangkan integrasi berkelanjutan dengan men-***deploy*** semua perubahan kode ke lingkungan pengujian dan/atau lingkungan produksi setelah tahap pembangunan.

Pelajari cara Amazon mengotomatiskan hands-off deployment yang aman »

[3] *Puppet 2020 State of Devops Report, 2020*

Dengan AWS, Lululemon Athletica dapat membuat lingkungan pengembangan dalam hitungan menit, bukan hari, mengotomatiskan lingkungannya, serta mewujudkan integrasi dan ***deployment*** berkelanjutan. Perusahaan Kanada ini menjual pakaian yang terinspirasi yoga dan pakaian lainnya di lebih dari 350 lokasi di seluruh dunia. Lululemon menjalankan lingkungan pengembangan dan pengujian—serta aplikasi selulernya yang akan segera dirilis—di AWS Cloud.

Lululemon mengurangi waktu yang dibutuhkan untuk membangun akun produksi baru dari dua hari menjadi beberapa menit, menggunakan templat AWS CloudFormation dan AWS CodePipeline. Dengan peningkatan ketangkasan tersebut, tim pengembangan Lululemon kini dapat bereksperimen dan memperoleh solusi terbaik, berbeda dari sebelumnya, di mana mereka harus puas dengan hasil yang diperoleh sesuai dengan sumber daya yang mereka miliki.

Baca kisah lengkapnya »

"Setiap alur integrasi dan deployment berkelanjutan harus terotomatisasi, mudah dikelola, dan ditemukan, dan itulah yang kami dapatkan dengan menggunakan AWS. Kami meraih kesederhanaan dan transparansi dengan level yang tidak mungkin diperoleh di lingkungan on-premise kami sebelumnya."

– Sam Keen, Direktur Arsitektur Produk (***Director of Product Architecture***), Lululemon

HyperTrack adalah platform ***cloud*** layanan mandiri untuk pelacakan lokasi langsung melalui aplikasi. Ketika diluncurkan pada akhir tahun 2015, HyperTrack perlu membangun platform yang dapat menskalakan secara otomatis untuk memenuhi pertumbuhan yang diperkirakan tanpa mengurangi waktu yang digunakan developer mereka untuk membangun fitur baru.

HyperTrack memilih menggunakan AWS Amplify untuk kerangka kerja pengembangan seluler dan arsitektur nirserver agar dapat menambah dan mengurangi skala secara otomatis tanpa intervensi rekayasa.

Sebagai hasilnya, perusahaan ini telah mewujudkan penghematan biaya sebesar 30 persen dibandingkan dengan arsitektur yang mereka gunakan sebelum beralih ke teknologi nirserver. Sebagian besar penghematan itu berasal dari sumber daya operasional yang sekarang tidak perlu lagi difokuskan pada manajemen server. HyperTrack menghemat 40 jam kerja, setiap pekan, sambil mengelola jutaan peristiwa.

Baca kisah lengkapnya »

Menciptakan budaya kepemilikan: kelola lebih sedikit, inovasi lebih banyak dengan Dev+Ops Modern

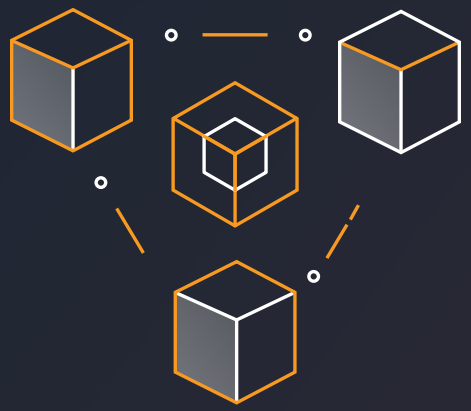
Inovasi pada dasarnya berasal dari manusia, sehingga pemberdayaan sumber daya manusia Anda untuk mewujudkan hasil pelanggan yang lebih baik merupakan langkah awal dari pengembangan aplikasi modern. Kami menggunakan konsep "produk, bukan proyek" untuk menguraikan bagaimana hal ini memengaruhi struktur tim. Secara sederhana, hal ini berarti bahwa tim yang membangun produk bertanggung jawab untuk menjalankan dan memelihara produk tersebut. Hal ini membuat tim produk bertanggung jawab untuk pengembangan keseluruhan produk, bukan hanya sebagian dari produk.

Setelah lebih dari satu dekade membangun dan menjalankan aplikasi web yang sangat dapat diskalakan, Amazon.com, kami mempelajari secara langsung pentingnya memberikan otonomi kepada tim kami. Ketika kami memberi tim kami kepemilikan atas siklus hidup aplikasi secara keseluruhan, termasuk menerima masukan pelanggan, merencanakan panduan (***roadmap***), serta mengembangkan dan mengoperasikan aplikasi, mereka menjadi pemilik dan merasa diberdayakan untuk mengembangkan dan mewujudkan hasil pelanggan yang baru.

Otonomi menciptakan motivasi, membuka pintu untuk kreativitas, dan mengembangkan budaya berani mengambil risiko dalam lingkungan yang penuh kepercayaan.

Meskipun merangkul budaya kepemilikan bukan hal yang pada dasarnya bersifat teknis, hal ini tetap menjadi salah satu aspek yang paling menantang dari pengembangan aplikasi modern. Pemberdayaan tim untuk menjadi pemilik produk membutuhkan perubahan pola pikir organisasi, struktur tim, dan pekerjaan yang menjadi tanggung jawab mereka.

Bagi sebagian besar organisasi, IT adalah salah satu dari dua aspek tersebut. IT dipandang sebagai senjata kompetitif strategis atau, lebih umumnya, sebagai pusat biaya yang diperlukan untuk mendukung pertumbuhan bisnis.



Membangun budaya inovasi

1. **Mulailah dengan pelanggan –** Setiap inovasi harus dimulai dengan kebutuhan pelanggan dan pada akhirnya inovasi ini harus dapat memuaskan pelanggan Anda. Selalu tetapkan prioritas untuk fokus pada permintaan pelanggan.

2. **Rekrut para pembangun dan biarkan mereka membangun –** Singkirkan hambatan apa pun yang memperlambat proses pembangunan serta rilis produk dan fitur untuk pelanggan. Semakin cepat Anda melakukan iterasi, semakin cepat roda Anda berputar.

3. **Dukung para pembangun dengan sistem kepercayaan –** Jangan hanya berbicara tentang inovasi—wujudkan dan lakukan inovasi di semua area bisnis, mulai dari kepemimpinan dan penjualan hingga dukungan.

Kelola lebih sedikit, inovasi lebih banyak

Aplikasi modern menciptakan diferensiasi kompetitif dengan memungkinkan inovasi yang cepat. Dengan mengadopsi layanan, praktik, dan strategi yang menekankan kecepatan dan ketangkasan, Anda dapat mengalihkan sumber daya dari operasi bisnis standar ke aktivitas diferensiasi dengan nilai pelanggan yang tinggi. Anda dapat bereksperimen lebih lanjut dan mewujudkan gagasan dengan lebih cepat. Anda dapat mengembangkan lingkungan tempat para pembangun menggunakan lebih banyak waktu untuk membangun dan lebih sedikit waktu untuk mengelola. Aplikasi modern adalah cara perusahaan, termasuk Amazon, berinovasi secara cepat dan tangkas.

Mengapa membangun aplikasi modern di AWS?

Waktu pemasaran yang lebih cepat	Dengan mempercepat siklus bangun dan rilis serta melepaskan beban operasional, developer dapat membangun fitur baru dengan cepat. Proses pengujian dan rilis otomatis mengurangi tingkat kesalahan, sehingga produk siap diluncurkan ke pasar dengan lebih cepat.	**Lihat buktinya:** **Urbanbase meluncurkan layanan 20x lebih cepat dengan AWS**
Tingkatkan Inovasi	Dengan arsitektur modular, perubahan terhadap masing-masing komponen aplikasi dapat dilakukan secara cepat dan dengan risiko lebih rendah terhadap seluruh aplikasi, sehingga tim dapat lebih sering bereksperimen dengan ide-ide baru.	**Lihat buktinya:** **iRobot menggunakan AWS Lambda dan platform AWS IoT untuk mengelola robot pengisap debunya, Roomba**
Tingkatkan Keandalan	Dengan mengotomatiskan prosedur pengujian dan pemantauan di setiap tahap siklus hidup pengembangan, aplikasi modern dapat diandalkan saat ***deployment***. Masalah apa pun dapat dievaluasi dan ditangani dalam waktu nyata.	**Lihat buktinya:** **Siemens mengurangi jumlah pemberitahuan sistem kontrol pelanggan sebesar 90 persen dan mengurangi biaya infrastruktur sebesar 85 persen**
Tingkatkan TCO	Dengan model harga bayar sesuai nilai, aplikasi modern mengurangi biaya penyediaan berlebih atau pembayaran sumber daya yang tidak terpakai. Dengan melepaskan beban manajemen infrastruktur, biaya pemeliharaan juga menjadi lebih rendah.	**Lihat buktinya:** **Hemat hingga 80 persen untuk pemeliharaan aplikasi dengan AWS Lambda**

Mulai perjalanan modernisasi aplikasi Anda

Platform Ulang ke Layanan Kontainer Terkelola

80% dari semua aplikasi dalam kontainer di ***cloud*** berjalan di AWS* 84% dari semua beban kerja Kubernetes di ***cloud*** berjalan di AWS*

Sumber Daya
Lokakarya Amazon ECS
Lokakarya Amazon EKS
Lokakarya AWS AppRunner

Pelatihan yang Direkomendasikan (Ruang Kelas)
Running Containers on Amazon Elastic Kubernetes Service (Amazon EKS)

Pelatihan yang Direkomendasikan (*Online*)
Amazon Elastic Container Service (ECS) Primer

Bangun Aplikasi Modern Baru dengan Teknologi dan Alat Nirserver

Hemat waktu hingga 80% untuk pemeliharaan dan sekitar 70% untuk pengembangan saat mengadopsi strategi pengutamaan nirserver (***serverless-first***) untuk membangun aplikasi modern**

Sumber Daya
Innovator Island - Lokakarya pengembangan aplikasi web nirserver. Tutorial video tentang membangun aplikasi web nirserver.

Pelatihan yang Direkomendasikan (Ruang Kelas)
Advanced Developing on AWS

Pelatihan yang Direkomendasikan (*Online*)
Architecting Serverless Solutions

Bertransformasi ke Model Dev+Ops Modern

60% tim dengan praktik DevOps yang lebih berkembang berhasil mengatasi kerentanan keamanan sepenuhnya dalam waktu kurang dari satu hari#

Sumber Daya
Amazon Builder's Library
Layanan AWS DevOps

Pelatihan yang Direkomendasikan (Ruang Kelas)
DevOps Engineering on AWS

Pelatihan yang Direkomendasikan (*Online*)
Getting started with DevOps on AWS

Pelajari lebih lanjut tentang cara membangun aplikasi modern di AWS →

Berdiskusi dengan seorang ahli untuk menerapkan praktik terbaik pengembangan aplikasi modern →

Terhubung dengan partner AWS untuk mempercepat proyek modernisasi Anda →

aws modern apps

* Sumber: Nucleus Research
** Sumber: Deloitte
# Sumber: 2020 DORA state of DevOps Report