Unable to locate subtitle

AWS Well-Architected Framework

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Well-Architected Framework: ***Unable to locate subtitle***

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Abstrak dan pengantar 1
Pengantar 1
Definisi 2
Pada arsitektur 5
Prinsip desain umum 6
Pilar kerangka kerja 8
Keunggulan operasional 8
Prinsip desain 9
Definisi 10
Praktik terbaik 10
Sumber daya 20
Keamanan 21
Prinsip desain 21
Definisi 22
Praktik terbaik 23
Sumber daya 29
Keandalan 30
Prinsip desain 31
Definisi 31
Praktik terbaik 32
Sumber daya 38
Efisiensi kinerja 38
Prinsip desain 38
Definisi 39
Praktik terbaik 40
Sumber daya 45
Optimasi biaya 46
Prinsip desain 46
Definisi 47
Praktik terbaik 48
Sumber daya 54
Pelestarian lingkungan 55
Prinsip desain 55
Definisi 56

Praktik terbaik 57
Proses peninjauan 65
Kesimpulan 68
Kontributor 69
Sumber Bacaan Lebih Lanjut 70
Revisi Dokumen 71
Lampiran: Pertanyaan dan praktik terbaik 75
Keunggulan operasional 75
Organisasi 75
Persiapan 112
Jalankan 182
Kembangkan 217
Keamanan 234
Fondasi keamanan 234
Manajemen identitas dan akses 253
Deteksi 305
Perlindungan infrastruktur 315
Perlindungan data 334
Respons insiden 365
Keamanan aplikasi 388
Keandalan 408
Fondasi 408
Arsitektur beban kerja 448
Manajemen perubahan 493
Manajemen kegagalan 532
Efisiensi kinerja 635
Pemilihan arsitektur 635
Komputasi dan perangkat keras 649
Manajemen data 666
Jaringan dan pengiriman konten 691
Proses dan budaya 721
Optimisasi biaya 736
Mempraktikkan Manajemen Keuangan Cloud 736
Kesadaran akan penggunaan dan pengeluaran 761
Sumber daya yang hemat 803
Kelola sumber daya pasokan dan permintaan 843

Pengoptimalan dari waktu ke waktu 856
Pelestarian Lingkungan 865
Pemilihan wilayah 865
Penyelarasan dengan permintaan 867
Perangkat lunak dan arsitektur 881
Data 892
Perangkat keras dan layanan 912
Proses dan budaya 922
Pemberitahuan 930

Kerangka Kerja AWS Well-Architected

Tanggal publikasi: 3 Oktober 2023 (Revisi Dokumen)

AWS Well-Architected Framework membantu Anda mengetahui kelebihan dan kekurangan keputusan yang Anda ambil saat membangun sistem di AWS. Dengan menggunakan Kerangka Kerja ini, Anda akan mengetahui praktik terbaik arsitektur untuk mendesain dan mengoperasikan sistem yang andal, aman, efisien, hemat biaya, dan ramah lingkungan di cloud.

Pengantar

AWS Well-Architected Framework membantu Anda mengetahui kelebihan dan kekurangan keputusan yang Anda ambil saat membangun sistem di AWS. Kerangka Kerja membantu Anda mempelajari praktik terbaik arsitektur untuk mendesain dan mengoperasikan beban kerja yang aman, andal, efisien, hemat biaya, dan ramah lingkungan di AWS Cloud. Layanan ini menyediakan cara untuk secara terus menerus menilai arsitektur Anda berdasarkan praktik terbaik dan mengidentifikasi area yang perlu diperbaiki. Proses peninjauan arsitektur adalah percakapan konstruktif tentang keputusan arsitektur, dan tidak dilakukan melalui mekanisme audit. Kami percaya bahwa memiliki sistem yang didesain dengan baik akan meningkatkan peluang keberhasilan bisnis.

Arsitek Solusi AWS telah berpengalaman selama bertahun-tahun dalam merancang solusi di berbagai vertikal bisnis dan kasus penggunaan. Kami telah membantu merancang dan meninjau ribuan arsitektur pelanggan di AWS. Dari pengalaman ini, kami mengidentifikasi praktik terbaik dan strategi inti dalam merancang sistem di cloud.

Kerangka Kerja AWS Well-Architected berisi serangkaian pertanyaan mendasar yang dapat Anda gunakan untuk memahami apakah arsitektur tertentu selaras dengan praktik terbaik cloud. Kerangka kerja ini memberikan pendekatan yang konsisten untuk mengevaluasi sistem berdasarkan kualitas sistem berbasis cloud yang Anda harapkan, serta perbaikan yang diperlukan untuk mencapai kualitas tersebut. Seiring perkembangan AWS dan pengalaman yang kami dapat dari bekerja sama dengan pelanggan, kami akan terus menyesuaikan definisi well-architected (dirancang dengan baik).

Kerangka kerja ini ditujukan bagi orang yang memiliki peran di bidang teknologi, seperti kepala pejabat teknologi (chief technology officer/CTO), arsitek, developer, dan anggota tim operasi. Kerangka ini menjabarkan praktik terbaik dan strategi AWS yang akan digunakan ketika merancang dan mengoperasikan beban kerja cloud, serta memberikan tautan ke pola arsitektur dan detail implementasi selengkapnya. Untuk informasi selengkapnya, lihat Halaman beranda AWS Well-Architected.

AWS juga menyediakan layanan peninjauan beban kerja gratis. Fitur Alat AWS Well-Architected (Alat AWS WA) adalah layanan di cloud yang menyediakan proses yang konsisten untuk meninjau dan mengukur arsitektur Anda menggunakan Kerangka Kerja AWS Well-Architected. Alat AWS WA memberikan rekomendasi untuk membuat beban kerja Anda lebih andal, aman, efisien, dan hemat biaya.

Untuk membantu Anda menerapkan praktik terbaik, kami telah menciptakan Lab AWS Well-Architected, yang menyediakan repositori kode dan dokumentasi agar dapat mencoba pengalaman praktik terbaik secara langsung. Kami juga telah bekerja sama dengan Partner Jaringan Partner AWS (APN) pilihan, yang merupakan anggota dari Program Partner AWS Well-Architected. Partner AWS ini memiliki pengetahuan mendalam tentang AWS, dan dapat membantu Anda untuk meninjau dan meningkatkan beban kerja Anda.

Definisi

Setiap hari, para ahli di AWS membantu pelanggan dalam merancang sistem untuk memanfaatkan praktik terbaik di cloud. Kami bekerja sama dengan Anda untuk membuat perubahan arsitektur seiring perkembangan desain Anda. Saat Anda melakukan deployment sistem ini ke lingkungan penggunaannya, kami mempelajari seberapa baik kinerja sistem ini dan konsekuensi dari perubahan tersebut.

Berdasarkan pembelajaran ini, kami menciptakan Kerangka Kerja AWS Well-Architected, yang menyediakan serangkaian praktik terbaik yang konsisten bagi pelanggan dan partner untuk mengevaluasi arsitektur, dan menyediakan serangkaian pertanyaan yang dapat Anda gunakan untuk mengevaluasi keselarasan arsitektur dengan praktik terbaik AWS.

Kerangka Kerja AWS Well-Architected mengacu pada enam pilar — keunggulan operasional, keamanan, keandalan, efisiensi kinerja, optimisasi biaya, dan pelestarian lingkungan.

Tabel 1. Pilar Kerangka Kerja AWS Well-Architected

Nama	Deskripsi
Keunggulan operasional	Kemampuan untuk mendukung pengembangan dan menjalankan beban kerja dengan efektif, mendapatkan wawasan tentang operasi mereka, serta untuk meningkatkan proses dan

Nama	Deskripsi
	prosedur pendukung secara terus menerus untuk memberikan nilai bisnis.
Keamanan	Pilar keamanan menjelaskan cara memanfaat kan teknologi cloud untuk melindungi data, sistem, dan aset guna meningkatkan postur keamanan Anda.
Keandalan	Pilar keandalan berkenaan dengan kemampuan beban kerja untuk menjalankan fungsinya dengan benar dan konsisten sesuai harapan. Ini termasuk kemampuan untuk mengoperasikan dan menguji beban kerja di seluruh siklus hidupnya. Laporan resmi ini berisi panduan praktik terbaik yang mendalam untuk mengimplementasikan beban kerja yang andal di AWS.
Efisiensi kinerja	Kemampuan untuk menggunakan sumber daya komputasi secara efisien untuk memenuhi persyaratan sistem, dan untuk memelihara efisiensi tersebut seiring dengan perubahan permintaan dan perkembangan teknologi.
Optimisasi biaya	Kemampuan untuk menjalankan sistem guna mendapatkan nilai bisnis dengan harga yang paling rendah.
Pelestarian Lingkungan	Kemampuan untuk terus meningkatkan dampak pada pelestarian lingkungan dengan mengurangi konsumsi energi dan meningkat kan efisiensi di semua komponen beban kerja dengan memaksimalkan manfaat dari sumber daya yang disediakan dan meminimalkan total sumber daya yang dibutuhkan.

Istilah yang kami gunakan di Kerangka Kerja AWS Well-Architected:

- A komponen adalah kode, konfigurasi, dan Sumber Daya AWS yang dikombinasikan untuk memenuhi persyaratan. Komponen biasanya berupa unit kepemilikan teknis, dan terpisah dari komponen lainnya.
- Istilah beban kerja digunakan untuk mengidentifikasi serangkaian komponen yang dikombinasikan untuk memberikan nilai bisnis. Beban kerja biasanya merupakan tingkat detail yang dibicarakan oleh pimpinan bisnis dan teknologi.
- Kami menganggap arsitektur sebagai cara penggabungan komponen dalam beban kerja. Cara komunikasi dan interaksi antarkomponen sering menjadi fokus diagram arsitektur.
- Pencapaian menandai perubahan kunci di arsitektur Anda seiring dengan perkembangannya di siklus hidup produksi (desain, implementasi, pengujian, peluncuran, dan proses produksi).
- Dalam suatu organisasi, portofolio teknologi adalah sekumpulan beban kerja yang diperlukan agar bisnis dapat beroperasi.
- Fitur tingkat usaha mengategorikan banyaknya waktu, upaya, dan kesulitan untuk mengimplementasikan suatu tugas. Setiap organisasi harus mempertimbangkan ukuran dan keahlian tim serta kompleksitas beban kerja untuk konteks tambahan agar dapat mengategorikan tingkat usaha organisasi dengan tepat.
  - Tinggi: Pekerjaan dapat berlangsung selama beberapa minggu atau bulan. Upaya ini dapat dibagi menjadi beberapa kisah, rilis, dan tugas.
  - Sedang: Pekerjaan dapat berlangsung selama beberapa hari atau minggu. Upaya ini dapat dibagi menjadi beberapa rilis, dan tugas.
  - Rendah: Pekerjaan dapat berlangsung selama beberapa jam atau hari. Upaya ini dapat dibagi menjadi beberapa tugas.

Saat merancang beban kerja, Anda memilah pilar sesuai dengan konteks bisnis Anda. Keputusan bisnis ini dapat mendorong prioritas rekayasa Anda. Anda dapat mengoptimalkan pengurangan dampak terhadap pelestarian lingkungan dan memperkecil biaya dengan mengorbankan keandalan dalam lingkungan pengembangan, atau, untuk solusi yang sangat penting, Anda dapat mengoptimalkan keandalan dengan biaya dan dampak terhadap pelestarian lingkungan yang lebih besar. Dalam solusi e-commerce, kinerja dapat memengaruhi pendapatan dan minat beli pelanggan. Keamanan dan keunggulan operasi umumnya menjadi pilar yang tidak dapat dikorbankan.

Pada arsitektur

Di lingkungan on-premise, pelanggan sering kali memiliki tim pusat untuk arsitektur teknologi yang bertindak sebagai lapisan atas bagi tim produk lain untuk memastikan mereka mengikuti praktik terbaik. Tim arsitektur teknologi biasanya berisi serangkaian peran, seperti Arsitek Teknis (infrastruktur), Arsitek Solusi (perangkat lunak), Arsitek Data, Arsitek Jaringan, dan Arsitek Keamanan. Tim ini sering menggunakan Kerangka Kerja TOGAF atau Zachman sebagai bagian dari kemampuan arsitektur perusahaan.

Di AWS, kami lebih suka mendistribusikan kemampuan ke beberapa tim daripada membentuk satu tim inti dengan kemampuan tersebut. Tentunya akan ada risiko ketika Anda memilih untuk mendistribusikan wewenang pengambilan keputusan, misalnya, memastikan semua tim memenuhi standar internal. Kami meminimalkan risiko ini melalui dua cara. Pertama, kami memiliki praktik (panduan, proses, standar, dan norma yang berlaku) yang fokus untuk memberikan kemampuan tersebut kepada setiap tim, dan kami memiliki ahli yang memastikan bahwa tim kami telah memenuhi standar yang ditetapkan. Kedua, kami mengimplementasikan mekanisme yang menjalankan pemeriksaan otomatis untuk menjamin pemenuhan standar.

> “Niat baik saja tidak cukup, harus diikuti dengan mekanisme yang bagus untuk mewujudkan sesuatu” — Jeff Bezos.

Artinya, upaya terbaik manusia digantikan dengan mekanisme (seringnya otomatis) yang memeriksa kepatuhan pada aturan atau proses. Pendekatan terdistribusi ini didukung oleh prinsip kepemimpinan Amazon, dan membangun budaya di semua peran yang dimulai dari pelanggan. Penelusuran mundur adalah bagian mendasar dalam proses inovasi kami. Kami memulai dari pelanggan dan keinginan mereka, kemudian menggunakan informasi ini sebagai penentu dan pedoman upaya kami. Tim khusus pelanggan membuat produk sesuai kebutuhan pelanggan.

Untuk arsitektur, artinya kami mengharapkan setiap tim dapat membuat arsitektur dan mengikuti praktik terbaik. Untuk membantu tim baru menguasai kemampuan ini atau agar tim yang ada dapat bekerja lebih baik lagi, kami menyediakan akses ke komunitas virtual yang berisi kepala rekayasawan yang dapat meninjau desain tim Anda serta membantu tim untuk memahami apa saja praktik terbaik AWS. Komunitas kepala rekayasa berupaya agar praktik terbaik dapat terlihat dan diterapkan. Salah satu contohnya melalui obrolan makan siang yang berfokus pada penerapan praktik terbaik dengan contoh nyata. Obrolan ini direkam dan dapat digunakan sebagai materi orientasi untuk anggota tim baru.

Praktik terbaik AWS muncul dari pengalaman kami menjalankan ribuan sistem pada skala internet. Kami cenderung menggunakan data untuk menentukan praktik terbaik, tetapi selain itu kami juga menggunakan ahli pokok bahasan, seperti kepala rekayasawan. Saat kepala rekayasawan membentuk praktik terbaik baru, mereka bekerja sebagai komunitas untuk memastikan tim mengikuti mereka. Nantinya, praktik terbaik ini diformalkan ke dalam proses peninjauan internal kami, serta ke dalam mekanisme yang menegakkan kepatuhan. Kerangka Kerja Well-Architected adalah implementasi proses tinjauan internal kami yang digunakan oleh pelanggan, yang telah kami kodifikasi dengan pemikiran kepala rekayasa di berbagai peran, seperti Arsitektur Solusi dan tim rekayasa internal. Kerangka Kerja Well-Architected adalah mekanisme yang dapat diskalakan yang dapat Anda gunakan untuk memanfaatkan pembelajaran ini.

Dengan mengikuti pendekatan komunitas kepala rekayasa dengan kepemilikan arsitektur terdistribusi, kami percaya bahwa arsitektur korporasi Well-Architected dapat dibuat sesuai kebutuhan pelanggan. Pemimpin teknologi (seperti CTO atau manajer pengembangan), melakukan peninjauan Well-Architected di semua beban kerja Anda agar Anda dapat lebih memahami risiko dalam portofolio teknologi Anda. Dengan pendekatan ini, Anda dapat mengidentifikasi tema di seluruh tim yang dapat ditangani oleh organisasi Anda melalui mekanisme, pelatihan, atau obrolan makan siang di mana kepala rekayasawan dapat membagikan pemikirannya tentang area tertentu kepada banyak tim.

Prinsip desain umum

Kerangka Kerja Well-Architected mengidentifikasi seperangkat prinsip desain umum untuk mendukung desain yang baik di cloud:

- Berhenti menebak kebutuhan kapasitas: Jika Anda tidak menentukan kapasitas dengan baik selama deployment beban kerja, sumber daya Anda yang mahal mungkin tidak akan terpakai atau ada banyak kendala kinerja karena keterbatasan kapasitas. Dengan komputasi cloud, permasalahan ini akan sirna. Anda dapat menggunakan jumlah kapasitas sesuai kebutuhan, dan menaikkan atau menurunkan skalanya secara otomatis.
- Uji sistem pada skala produksi: Di cloud, Anda dapat membuat lingkungan pengujian berskala produksi sesuai permintaan, menyelesaikan pengujian, kemudian menonaktifkan sumber dayanya. Karena Anda hanya membayar lingkungan pengujian yang dijalankan, Anda dapat menyimulasikan lingkungan langsung Anda dengan biaya yang lebih murah daripada pengujian on-premise.
- Mengotomatisasi dengan mempertimbangkan eksperimen arsitektur: Dengan otomatisasi, Anda dapat membuat dan mereplikasi beban kerja dengan biaya rendah dan menghindari biaya untuk

upaya manual. Anda dapat melacak perubahan pada otomatisasi, mengaudit dampaknya, dan mengembalikan ke parameter sebelumnya saat dibutuhkan.

- Pertimbangkan arsitektur evolusioner: Di lingkungan tradisional, keputusan arsitektur sering diimplementasikan sebagai peristiwa statis sekali tempo, dengan beberapa versi utama sistem selama masa pakainya. Seiring dengan perkembangan bisnis dan konteksnya, keputusan awal ini dapat menghambat kemampuan sistem untuk memenuhi kebutuhan bisnis yang terus berubah. Di cloud, kemampuan untuk mengotomatiskan dan menguji sesuai permintaan menurunkan risiko dampak perubahan desain. Hal ini memungkinkan sistem berkembang seiring waktu sehingga bisnis dapat memanfaatkan inovasi sebagai praktik standar.
- Bentuk arsitektur menggunakan data: Di cloud, Anda dapat mengumpulkan data tentang pengaruh pilihan arsitektur terhadap perilaku beban kerja Anda. Dengan demikian, Anda dapat membuat keputusan sesuai fakta terkait cara memperbaiki beban kerja. Infrastruktur cloud Anda berupa kode, sehingga Anda dapat menggunakan data tersebut untuk menginformasikan pilihan dan peningkatan arsitektur Anda dari waktu ke waktu.
- Tingkatkan melalui game day: Uji kinerja proses dan arsitektur Anda dengan rutin mengadakan game day untuk menyimulasikan peristiwa di produksi. Hal ini akan membantu Anda memahami sisi mana yang perlu ditingkatkan dan membantu mengembangkan pengalaman organisasi dalam menangani peristiwa.

Pilar kerangka kerja

Membuat sistem perangkat lunak sangat mirip dengan membangun sebuah bangunan. Jika fondasinya tidak kokoh, masalah struktur dapat mengganggu kesatuan dan fungsi bangunan. Saat merancang solusi teknologi, jika Anda mengabaikan enam pilar keunggulan operasional, keamanan, keandalan, efisiensi kinerja, optimasi biaya, dan pelestarian lingkungan, membangun sistem yang sesuai dengan harapan dan persyaratan Anda bisa jadi sulit dilakukan. Dengan memasukkan pilar-pilar ini ke dalam arsitektur, Anda lebih mudah dalam memproduksi sistem yang stabil dan efisien. Ini akan memungkinkan Anda untuk berfokus pada aspek-aspek desain lain, seperti persyaratan fungsional.

Pilar

- Keunggulan operasional
- Keamanan
- Keandalan
- Efisiensi kinerja
- Optimasi biaya
- Pelestarian lingkungan

Keunggulan operasional

Pilar Keunggulan Operasional mencakup kemampuan untuk mendukung pengembangan dan menjalankan beban kerja dengan efektif, mendapatkan wawasan tentang operasi mereka, serta untuk meningkatkan proses dan prosedur pendukung secara terus menerus untuk memberikan nilai bisnis.

Pilar keunggulan operasional menyediakan gambaran umum tentang prinsip, praktik terbaik, dan pertanyaan desain. Anda dapat menemukan panduan preskriptif tentang implementasi dalam whitepaper Pilar Keunggulan Operasional.

Topik

- Prinsip desain
- Definisi
- Praktik terbaik
- Sumber daya

Prinsip desain

Berikut prinsip desain untuk keunggulan operasional di cloud:

- Jalankan operasi sebagai kode: Di cloud, Anda dapat menerapkan teknik rekayasa yang sama yang Anda gunakan untuk kode aplikasi ke lingkungan Anda secara keseluruhan. Anda dapat menentukan seluruh beban kerja Anda (aplikasi, infrastruktur, dst.) sebagai kode dan memperbaruinya dengan kode. Anda dapat merencanakan prosedur operasi Anda dan mengotomatiskan prosesnya dengan meluncurkannya saat peristiwa terjadi. Dengan melakukan operasi sebagai kode, Anda membatasi kesalahan manusia dan membuat respons yang sesuai terhadap peristiwa.
- Buat perubahan yang sering, kecil, dan dapat dibalik: Rancang beban kerja yang dapat diskalakan dan digabungkan secara longgar untuk memungkinkan komponen diperbarui secara teratur. Teknik deployment otomatis bersama dengan perubahan yang lebih kecil dan bertahap mengurangi radius ledakan dan memungkinkan pembalikan lebih cepat ketika terjadi kegagalan. Hal ini meningkatkan kepercayaan diri untuk memberikan perubahan yang menguntungkan pada beban kerja Anda sekaligus mempertahankan kualitas dan beradaptasi dengan cepat terhadap perubahan kondisi pasar.
- Sering-seringlah menyempurnakan prosedur operasi: Seiring dengan perkembangan beban kerja Anda, kembangkan operasi Anda dengan semestinya. Saat Anda menggunakan prosedur operasi, carilah peluang untuk meningkatkannya. Lakukan peninjauan rutin dan validasikan bahwa semua prosedur sudah efektif dan dipahami dengan baik oleh tim. Jika kesenjangan diidentifikasi, perbarui prosedur yang sesuai. Komunikasikan pembaruan prosedural kepada semua pemangku kepentingan dan tim. Ciptakan mekanisme yang menyenangkan dalam operasi Anda untuk berbagi praktik terbaik dan mengedukasi tim.
- Antisipasi kegagalan: Lakukan uji pre-mortem untuk mengidentifikasi kemungkinan sumber kegagalan agar sumber tersebut dapat dihapus atau dimitigasi. Uji skenario kegagalan Anda dan validasi pemahaman Anda tentang dampaknya. Uji prosedur respons Anda untuk memastikan prosedur sudah efektif dan tim sudah memahami prosesnya. Atur game day secara rutin untuk menguji beban kerja dan respons tim terhadap simulasi peristiwa.
- Belajar dari semua kegagalan operasional: Dorong peningkatan dengan belajar dari semua peristiwa dan kegagalan operasional yang telah terjadi. Bagikan materi yang telah dipelajari kepada seluruh tim dan organisasi.
- Gunakan layanan terkelola: Kurangi beban operasional menggunakan layanan terkelola AWS jika memungkinkan. Bangun prosedur operasional seputar interaksi dengan layanan tersebut.

- Terapkan observabilitas untuk wawasan yang dapat ditindaklanjuti: Dapatkan pemahaman komprehensif tentang perilaku beban kerja, performa, keandalan, biaya, dan kesehatan. Tetapkan indikator kinerja utama (KPI) dan manfaatkan telemetri observabilitas untuk membuat keputusan yang lebih tepat dan mengambil tindakan cepat ketika hasil bisnis berisiko. Tingkatkan performa, keandalan, dan biaya secara proaktif berdasarkan data observabilitas yang dapat ditindaklanjuti.

Definisi

Berikut empat area praktik terbaik untuk keunggulan operasional di cloud:

- Pengaturan
- Persiapan
- Pengoperasian
- Evolusi

Kepemimpinan organisasi Anda menentukan tujuan bisnis. Organisasi Anda harus memahami kebutuhan dan prioritas serta menggunakannya untuk mengatur dan melakukan pekerjaan guna mendukung pencapaian hasil bisnis. Beban kerja Anda harus memberikan informasi yang diperlukan untuk mendukungnya. Mengimplementasikan layanan untuk mencapai integrasi, deployment, dan penyediaan beban kerja Anda akan menciptakan peningkatan alur perubahan yang menguntungkan menuju produksi dengan mengotomatiskan proses yang berulang.

Mungkin ada risiko yang melekat dalam pengoperasian beban kerja Anda. Anda harus memahami risiko tersebut dan mengambil keputusan yang matang untuk beralih ke produksi. Tim Anda harus mampu mendukung beban kerja Anda. Dengan metrik bisnis dan operasional yang diperoleh dari hasil bisnis yang diinginkan, Anda dapat memahami kondisi beban kerja, aktivitas operasi, serta respons Anda terhadap insiden. Prioritas Anda akan berubah sesuai kebutuhan bisnis dan perubahan lingkungan bisnis Anda. Gunakan ini sebagai loop umpan balik untuk mendorong peningkatan secara berkelanjutan bagi organisasi Anda dan operasi beban kerja Anda.

Praktik terbaik

Topik

- Pengaturan
- Persiapan
- Pengoperasian

- Evolusi

Pengaturan

Tim Anda harus memiliki pemahaman bersama tentang seluruh beban kerja Anda, peran mereka di dalamnya, dan sasaran bisnis bersama untuk menetapkan prioritas yang akan mendukung kesuksesan bisnis. Prioritas yang terdefinisi dengan baik akan memaksimalkan manfaat dari upaya Anda. Evaluasi kebutuhan internal dan eksternal pelanggan dengan melibatkan pemangku kepentingan utama, termasuk tim bisnis, pengembangan, dan operasional, untuk menentukan arah fokus upaya. Mengevaluasi kebutuhan pelanggan akan memastikan Anda memiliki pemahaman menyeluruh mengenai dukungan yang dibutuhkan untuk mencapai hasil bisnis. Pastikan Anda mengetahui pedoman atau kewajiban yang ditetapkan oleh tata kelola organisasi Anda serta faktor eksternal, seperti persyaratan kepatuhan peraturan dan standar industri, yang mungkin mewajibkan atau menekankan fokus tertentu. Validasikan bahwa Anda memiliki mekanisme untuk mengidentifikasi perubahan pada tata kelola internal dan persyaratan kepatuhan eksternal. Jika persyaratan ini belum teridentifikasi, pastikan Anda telah menerapkan uji kelayakan untuk penetapan tersebut. Tinjau prioritas Anda secara berkala agar dapat diperbarui sesuai perubahan kebutuhan.

Evaluasi ancaman pada bisnis (misalnya, risiko dan kewajiban hukum bisnis, serta ancaman keamanan informasi) dan pelihara informasi ini pada registri risiko. Evaluasi dampak risiko, serta kompromi di antara kepentingan yang bertentangan atau pendekatan alternatif. Misalnya, meningkatkan kecepatan fitur baru ke pasar dapat lebih diprioritaskan daripada optimisasi biaya, atau Anda dapat memilih basis data relasional untuk data non-relasional guna menyederhanakan upaya migrasi sistem tanpa pemfaktoran ulang. Kelola manfaat dan risiko untuk membuat keputusan yang tepat ketika menentukan arah fokus upaya. Risiko atau pilihan tertentu mungkin dapat diterima sesaat, risiko terkait mungkin dapat dimitigasi, atau membiarkan risiko tetap ada mungkin menjadi tidak dapat diterima. Jika demikian, Anda akan mengambil tindakan untuk mengatasi risiko tersebut.

Tim Anda harus memahami peran mereka dalam mencapai hasil bisnis. Tim harus memahami peran mereka dalam kesuksesan tim lain, peran tim lain dalam kesuksesan mereka, dan memiliki sasaran bersama. Memahami tanggung jawab, kepemilikan, bagaimana keputusan diambil, dan siapa yang memiliki otoritas untuk mengambil keputusan akan membantu memfokuskan upaya dan memaksimalkan manfaat dari tim Anda. Kebutuhan sebuah tim akan dibentuk oleh pelanggan yang mereka dukung, organisasi mereka, formasi tim, dan karakteristik beban kerja mereka. Tidak realistis berharap pada satu model operasional untuk mampu mendukung semua tim dan beban kerja mereka di organisasi Anda.

Pastikan ada pemilik yang teridentifikasi untuk setiap aplikasi, beban kerja, platform, dan komponen infrastruktur, serta ada pemilik yang teridentifikasi untuk setiap proses dan prosedur yang bertanggung jawab atas definisinya, dan pemilik yang bertanggung jawab atas kinerja mereka.

Memahami nilai bisnis setiap komponen, proses, dan prosedur, tentang alasan penyediaan sumber daya atau alasan dilakukannya aktivitas, serta alasan adanya kepemilikan tersebut akan menjadi dasar tindakan anggota tim Anda. Tentukan dengan jelas tanggung jawab anggota tim sehingga mereka bisa bertindak dengan benar dan memiliki mekanisme untuk mengidentifikasi tanggung jawab dan kepemilikan. Miliki mekanisme untuk meminta penambahan, perubahan, dan pengecualian sehingga Anda tidak membatasi inovasi. Tetapkan perjanjian antartim yang menjelaskan tentang bagaimana mereka bekerja sama untuk mendukung satu sama lain dan mendukung hasil bisnis Anda.

Sediakan dukungan untuk anggota tim Anda agar mereka bisa menjadi lebih efektif dalam mengambil tindakan dan mendukung hasil bisnis Anda. Kepemimpinan senior yang terlibat harus menetapkan ekspektasi dan mengukur kesuksesan. Pimpinan senior harus menjadi pendukung, penasihat, dan pendorong untuk mengadopsi praktik terbaik maupun perkembangan organisasi. Biarkan anggota tim mengambil tindakan ketika terdapat risiko pada hasil agar dampak dapat diminimalkan, serta dorong mereka untuk melakukan eskalasi kepada pengambil keputusan dan pemangku kepentingan ketika mereka yakin terdapat risiko agar dapat segera ditangani dan insiden dapat dicegah. Sediakan komunikasi yang tepat waktu, jelas, dan dapat ditindaklanjuti tentang risiko yang diketahui serta peristiwa yang direncanakan agar anggota tim dapat mengambil tindakan yang tepat waktu dan sesuai.

Dorong pelaksanaan eksperimen untuk meningkatkan proses pembelajaran dan menjaga minat serta keterlibatan anggota tim. Tim harus mengembangkan set keterampilan mereka untuk mengadopsi perkembangan teknologi, serta untuk mengimbangi perubahan permintaan dan tanggung jawab. Dukung dan dorong hal ini dengan menyediakan waktu terstruktur khusus untuk pembelajaran. Pastikan anggota tim Anda memiliki sumber daya, berupa alat bantu dan anggota tim, untuk meraih keberhasilan serta skala untuk mendukung hasil bisnis Anda. Manfaatkan keragaman lintas organisasi untuk mencari berbagai perspektif unik. Gunakan perspektif ini untuk meningkatkan inovasi, menantang asumsi Anda, dan mengurangi risiko bias konfirmasi. Kembangkan inklusi, keragaman, dan kemudahan akses dalam tim Anda untuk mendapatkan perspektif yang bermanfaat.

Jika ada peraturan eksternal atau persyaratan kepatuhan yang berlaku untuk organisasi Anda, Anda harus menggunakan sumber daya yang disediakan oleh AWS Cloud Compliance untuk membantu mengedukasi tim Anda agar mereka dapat menentukan dampak pada prioritas Anda. Kerangka Kerja Well-Architected menekankan pembelajaran, pengukuran, dan peningkatan. Kerangka

kerja ini menyediakan pendekatan yang konsisten bagi Anda untuk mengevaluasi arsitektur, dan mengimplementasikan desain yang akan meningkat seiring waktu. AWS menyediakan AWS Well-Architected Tool untuk membantu Anda meninjau pendekatan Anda sebelum pengembangan, status beban kerja Anda sebelum produksi, serta status beban kerja Anda dalam produksi. Anda dapat membandingkan beban kerja dengan praktik terbaik arsitektur AWS terbaru, memantau statusnya secara keseluruhan, dan mendapatkan wawasan tentang potensi risiko. AWS Trusted Advisor adalah alat bantu yang menyediakan akses ke rangkaian inti pemeriksaan yang merekomendasikan optimalisasi yang dapat membantu membentuk prioritas Anda. Pelanggan Dukungan Bisnis dan Korporasi menerima akses ke pemeriksaan tambahan yang berfokus pada keamanan, keandalan, kinerja, optimisasi biaya, dan keberlanjutan yang dapat membantu membentuk prioritas mereka lebih lanjut.

AWS dapat membantu mengedukasi tim Anda tentang AWS beserta layanannya untuk meningkatkan pemahaman mereka tentang bagaimana pilihan mereka dapat menimbulkan dampak pada beban kerja. Gunakan sumber daya yang disediakan oleh AWS Dukungan (AWS Pusat Pengetahuan, AWSForum Diskusi, dan Pusat AWS Dukungan) serta Dokumentasi AWS untuk mengedukasi tim Anda. Hubungi AWS Dukungan melalui Pusat AWS Dukungan untuk mengajukan pertanyaan AWS Anda. AWS juga membagikan praktik terbaik serta pola yang telah kami pelajari melalui operasi AWS di Amazon Builders' Library. Beragam informasi berguna lainnya dapat diakses melalui Blog AWS dan Podcast AWS Resmi. AWS Pelatihan dan Sertifikasi menyediakan beberapa pelatihan melalui kursus digital mandiri tentang dasar-dasar AWS. Anda juga dapat mengikuti pelatihan yang dibimbing instruktur untuk mendukung perkembangan keterampilan AWS tim Anda.

Gunakan alat bantu atau layanan yang memungkinkan Anda mengelola lingkungan di seluruh akun secara terpusat, seperti AWS Organizations, untuk membantu mengelola model operasi Anda. Layanan seperti AWS Control Tower akan memperluas kemampuan manajemen ini dengan memudahkan Anda menentukan cetak biru (yang mendukung model operasi Anda) untuk penyiapan akun, menerapkan tata kelola berkelanjutan menggunakan AWS Organizations, dan mengotomatiskan penyediaan akun baru. Penyedia Layanan Terkelola seperti AWS Managed Services, Partner AWS Managed Services, atau Penyedia Layanan Terkelola di Jaringan Partner AWS, menyediakan keahlian implementasi lingkungan cloud, dan mendukung persyaratan keamanan dan kepatuhan serta tujuan bisnis Anda. Menambahkan Layanan Terkelola ke model operasi Anda dapat menghemat waktu dan sumber daya Anda, serta memungkinkan Anda menjaga tim internal Anda untuk tetap belajar dan fokus pada hasil strategis yang akan membedakan bisnis Anda, dan bukan mengembangkan keterampilan dan kemampuan baru.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk keunggulan operasional. (Untuk melihat daftar pertanyaan dan praktik terbaik keunggulan operasional, buka Appendix.)

OPS 1: How do you determine what your priorities are?

Everyone must understand their part in achieving business success. Have shared goals in order to set priorities for resources. This will maximize the benefits of your efforts.

OPS 2: How do you structure your organization to support your business outcomes?

Your teams must understand their part in achieving business outcomes. Teams must understand their roles in the success of other teams, the role of other teams in their success, and have shared goals. Understanding responsibility, ownership, how decisions are made, and who has authority to make decisions will help focus efforts and maximize the benefits from your teams.

OPS 3: How does your organizational culture support your business outcomes?

Provide support for your team members so that they can be more effective in taking action and supporting your business outcome.

Anda mungkin mendapati bahwa Anda ingin mengutamakan sebagian kecil dari prioritas Anda pada titik waktu tertentu. Gunakan pendekatan yang seimbang dalam jangka panjang untuk memastikan pengembangan kemampuan yang diperlukan dan manajemen risiko. Tinjau prioritas Anda secara rutin dan perbarui sesuai perubahan kebutuhan. Ketika tanggung jawab dan kepemilikan tidak ditetapkan atau tidak diketahui, Anda menanggung risiko karena tidak melakukan tindakan yang diperlukan secara tepat waktu serta risiko munculnya upaya yang berulang dan berpotensi bertentangan untuk menangani kebutuhan-kebutuhan tersebut. Budaya organisasi berdampak langsung pada retensi dan kepuasan kerja anggota tim. Dukung keterlibatan dan kemampuan anggota tim Anda untuk mencapai keberhasilan bisnis Anda. Eksperimen diperlukan untuk menciptakan inovasi dan mewujudkan ide. Ketahui bahwa hasil yang tidak sesuai harapan merupakan eksperimen yang berhasil, karena dengan begitu jalur yang tidak mengarahkan kepada keberhasilan dapat diidentifikasi.

Persiapan

Untuk menyiapkan keunggulan operasional, Anda harus memahami beban kerja Anda serta perkiraan perilakunya. Dengan begitu Anda akan mampu merancangnya agar dapat menyediakan wawasan tentang statusnya dan membangun prosedur untuk mendukungnya.

Desain beban kerja Anda sedemikian rupa sehingga memberikan informasi yang Anda perlukan untuk memahami status internalnya (seperti metrik, log, dan jejak) di semua komponen untuk mendukung observabilitas dan investigasi masalah. Observabilitas lebih dari sekadar pemantauan sederhana, yang memberikan pemahaman yang komprehensif tentang cara kerja internal sistem berdasarkan output eksternalnya. Berakar pada metrik, log, dan jejak, observabilitas menawarkan wawasan mendalam tentang perilaku dan dinamika sistem. Dengan observabilitas yang efektif, tim dapat membedakan pola, anomali, dan tren, memungkinkan mereka untuk secara proaktif mengatasi masalah potensial dan menjaga kesehatan sistem yang optimal. Mengidentifikasi indikator kinerja utama (KPI) sangat penting untuk memastikan keselarasan antara kegiatan pemantauan dan tujuan bisnis. Penyelarasan ini memastikan bahwa tim membuat keputusan berbasis data menggunakan metrik yang benar-benar penting, mengoptimalkan kinerja sistem dan hasil bisnis. Selain itu, observabilitas memberdayakan bisnis untuk menjadi proaktif, bukan reaktif. Tim dapat memahami hubungan sebab-akibat dalam sistem mereka, memprediksi dan mencegah masalah, bukan hanya bereaksi terhadapnya. Seiring berkembangnya beban kerja, penting untuk meninjau kembali dan menyempurnakan strategi observabilitas, guna memastikannya tetap relevan dan efektif.

Adopsi pendekatan yang meningkatkan aliran perubahan ke dalam produksi dan yang mencapai pemfaktoran ulang, umpan balik cepat atas kualitas, dan perbaikan bug. Hal-hal ini mempercepat perubahan positif yang memasuki tahap produksi, membatasi masalah yang diterapkan, dan memungkinkan identifikasi serta perbaikan yang cepat terhadap masalah yang muncul dari aktivitas deployment atau yang ditemukan di lingkungan Anda.

Adopsi pendekatan yang memberikan umpan balik cepat atas kualitas dan mencapai pemulihan cepat dari perubahan yang tidak memiliki hasil yang tidak diinginkan. Menggunakan praktik tersebut akan memitigasi dampak masalah akibat deployment perubahan. Antisipasikan perubahan yang tidak berhasil sehingga Anda mampu merespons lebih cepat jika dibutuhkan serta menguji dan memvalidasi perubahan yang Anda buat. Perhatikan aktivitas terencana di lingkungan Anda sehingga Anda dapat mengelola risiko perubahan yang mempengaruhi aktivitas terencana. Prioritaskan perubahan yang sering, kecil, dan dapat dikembalikan untuk membatasi cakupan perubahan. Hal ini menghasilkan pemecahan masalah dan perbaikan yang lebih cepat dengan opsi membatalkan perubahan. Dengan begitu Anda juga dapat memperoleh manfaat dari perubahan yang berharga secara lebih sering.

Evaluasi kesiapan operasional beban kerja, proses, prosedur, dan personel Anda untuk memahami risiko operasional terkait beban kerja Anda. Gunakan proses yang konsisten (termasuk daftar periksa manual dan otomatis) untuk mengetahui saat Anda siap untuk mengoperasikan beban kerja Anda atau untuk melakukan perubahan. Hal ini juga akan membantu Anda menemukan area mana pun yang harus Anda buatkan rencana untuk ditangani. Miliki runbook yang mendokumentasikan aktivitas rutin serta buku pedoman yang memandu proses penyelesaian masalah Anda. Pahami manfaat dan risiko untuk membuat keputusan yang tepat agar perubahan dapat diterapkan dalam produksi.

AWS memungkinkan Anda menampilkan keseluruhan beban kerja (aplikasi, infrastruktur, kebijakan, tata kelola, dan operasi) sebagai kode. Hal ini berarti Anda dapat menerapkan disiplin rekayasa yang sama yang Anda gunakan untuk kode aplikasi ke setiap elemen tumpukan Anda dan membagikan semuanya ke seluruh tim atau organisasi untuk memperbesar manfaat upaya pengembangan. Gunakan operasi sebagai kode di cloud dan kemampuan untuk bereksperimen dengan aman guna mengembangkan beban kerja Anda, prosedur operasi Anda, serta antisipasi kegagalan. Menggunakan CloudFormation memungkinkan Anda memiliki lingkungan pengembangan, pengujian, dan produksi sandbox yang konsisten, bertemplat dengan tingkat kontrol operasi yang makin meningkat.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk keunggulan operasional.

OPS 4: How do you implement observability in your workload?

Implement observability in your workload so that you can understand its state and make data-driv en decisions based on business requirements.

OPS 5: How do you reduce defects, ease remediation, and improve flow into production?

Adopt approaches that improve flow of changes into production that achieve refactoring fast feedback on quality, and bug fixing. These accelerate beneficial changes entering production, limit issues deployed, and achieve rapid identification and remediation of issues introduced through deployment activities.

OPS 6: How do you mitigate deployment risks?

Adopt approaches that provide fast feedback on quality and achieve rapid recovery from changes that do not have desired outcomes. Using these practices mitigates the impact of issues introduce d through the deployment of changes.

OPS 7: How do you know that you are ready to support a workload?

Evaluate the operational readiness of your workload, processes and procedures, and personnel to understand the operational risks related to your workload.

Berinvestasi dalam implementasi aktivitas operasi sebagai kode untuk memaksimalkan produktivitas personel operasi, meminimalkan tingkat kesalahan, dan mencapai respons otomatis. Gunakan “pre-mortem” untuk mengantisipasi kegagalan dan membuat prosedur ketika diperlukan. Terapkan metadata menggunakan Tag Sumber Daya dan AWS Resource Groups sesuai strategi penandaan yang konsisten untuk mencapai identifikasi sumber daya Anda. Tandai sumber daya Anda untuk pengaturan, akuntansi biaya, kontrol akses, dan penargetan pelaksanaan aktivitas operasi otomatis. Adopsi praktik deployment yang memanfaatkan elastisitas cloud untuk memfasilitasi aktivitas pengembangan, dan pra-deployment sistem untuk implementasi yang lebih cepat. Ketika Anda membuat perubahan pada daftar periksa yang Anda gunakan untuk mengevaluasi beban kerja Anda, rencanakan apa yang akan Anda lakukan dengan sistem langsung yang tidak lagi patuh.

Pengoperasian

Observabilitas memungkinkan Anda fokus pada data yang bermakna serta memahami interaksi dan output beban kerja Anda. Dengan berkonsentrasi pada wawasan penting dan menghilangkan data yang tidak perlu, Anda mempertahankan pendekatan langsung untuk memahami kinerja beban kerja. Hal ini sangat penting tidak hanya untuk mengumpulkan data tetapi juga untuk menafsirkannya dengan benar. Menentukan garis acuan yang jelas, menetapkan ambang batas peringatan yang sesuai, dan memantau secara aktif setiap penyimpangan. Pergeseran metrik kunci, terutama ketika berkorelasi dengan data lain, dapat menunjukkan dengan tepat area masalah tertentu. Dengan observabilitas, Anda lebih siap untuk memperkirakan dan mengatasi tantangan potensial, memastikan bahwa beban kerja Anda beroperasi dengan lancar dan memenuhi kebutuhan bisnis.

Keberhasilan operasi beban kerja diukur dengan pencapaian hasil bisnis dan pelanggan. Tetapkan hasil yang diharapkan, tentukan bagaimana keberhasilan akan diukur, dan identifikasi metrik yang

akan digunakan pada perhitungan tersebut untuk menentukan apakah beban kerja dan operasi Anda berhasil. Kondisi operasional meliputi kondisi beban kerja serta kondisi dan keberhasilan aktivitas operasi yang dilakukan dalam dukungan beban kerja (misalnya, deployment dan respons insiden). Tetapkan baris acuan metrik untuk peningkatan, investigasi, serta intervensi, kumpulkan dan analisis metrik Anda, kemudian validasi pemahaman Anda tentang keberhasilan operasi dan bagaimana hal tersebut berubah seiring waktu. Gunakan metrik yang dikumpulkan untuk menentukan apakah Anda memenuhi kebutuhan pelanggan dan bisnis, serta identifikasi area peningkatan.

Manajemen peristiwa operasional yang efektif dan efisien diperlukan untuk mencapai keunggulan operasional. Hal ini berlaku untuk peristiwa operasional baik yang terencana maupun tidak terencana. Gunakan runbook yang telah dibuat untuk peristiwa yang dipahami dengan baik, dan gunakan buku panduan untuk membantu investigasi dan resolusi masalah. Prioritaskan respons terhadap peristiwa berdasarkan dampaknya pada bisnis dan pelanggan. Pastikan bahwa jika muncul peringatan sebagai respons terhadap suatu peristiwa, ada proses terkait untuk dijalankan, dengan pemilik yang diidentifikasi secara spesifik. Tentukan terlebih dulu personel yang dibutuhkan untuk menyelesaikan suatu peristiwa dan sertakan proses eskalasi agar dapat melibatkan personel tambahan, jika diperlukan, berdasarkan urgensi dan dampaknya. Identifikasi dan libatkan individu yang memiliki wewenang untuk membuat keputusan mengenai tindakan yang akan menimbulkan dampak bisnis dari respons peristiwa yang belum ditangani sebelumnya.

Komunikasikan status operasional beban kerja melalui dasbor dan pemberitahuan yang disesuaikan dengan audiens target (misalnya, pelanggan, bisnis, pengembang, operasi) sehingga mereka bisa mengambil tindakan yang sesuai, ekspektasi mereka terkelola, serta mereka mendapatkan informasi ketika operasi kembali normal.

Di AWS, Anda dapat membuat tampilan dasbor metrik Anda yang dikumpulkan dari beban kerja dan secara native dari AWS. Anda dapat memanfaatkan CloudWatch atau aplikasi pihak ketiga untuk menggabungkan dan mempresentasikan tampilan tingkat bisnis, beban kerja, dan operasi terkait aktivitas operasi. AWS menyediakan wawasan beban kerja melalui kemampuan pencatatan yang mencakup AWS X-Ray, CloudWatch, CloudTrail, dan Log Alur VPC untuk mengidentifikasi masalah beban kerja dalam mendukung analisis akar masalah dan perbaikan.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk keunggulan operasional.

OPS 8: How do you utilize workload observability in your organization?

Ensure optimal workload health by leveraging observability. Utilize relevant metrics, logs, and traces to gain a comprehensive view of your workload's performance and address issues efficient ly.

OPS 9: How do you understand the health of your operations?

Define, capture, and analyze operations metrics to gain visibility to operations events so that you can take appropriate action.

OPS 10: How do you manage workload and operations events?

Prepare and validate procedures for responding to events to minimize their disruption to your workload.

Semua metrik yang Anda kumpulkan harus selaras dengan kebutuhan bisnis dan hasil yang didukung. Kembangkan respons dalam skrip untuk memahami peristiwa dengan baik dan otomatiskan respons tersebut saat ada peristiwa yang dikenali.

Evolusi

Belajar, berbagi, dan terus melakukan peningkatan untuk mempertahankan keunggulan operasional. Dedikasikan siklus kerja untuk melakukan perbaikan bertahap yang hampir terus-menerus. Lakukan analisis pasca-insiden tentang semua peristiwa yang memengaruhi pelanggan. Identifikasi faktor kontribusi dan tindakan preventif untuk meminimalkan atau mencegah kemungkinan terjadi lagi. Komunikasikan faktor penyebab kepada komunitas yang terpengaruh jika perlu. Evaluasi secara teratur dan prioritaskan peluang untuk peningkatan (misalnya, permintaan fitur, perbaikan masalah, serta persyaratan kepatuhan), termasuk beban kerja dan prosedur operasi.

Sertakan loop umpan balik dalam prosedur Anda untuk mengidentifikasi dengan cepat area yang perlu ditingkatkan serta menangkap pembelajaran dari operasi yang berjalan.

Bagikan pelajaran yang didapatkan kepada seluruh tim untuk membagikan manfaat dari pelajaran tersebut. Analisis tren dalam pelajaran yang didapatkan dan lakukan analisis metrik operasi

secara retrospektif lintas tim untuk mengidentifikasi peluang dan metode untuk peningkatan. Implementasikan perubahan yang ditujukan untuk menghadirkan peningkatan dan evaluasi hasil untuk menentukan keberhasilan.

Di AWS, Anda dapat mengekspor data log Anda ke Amazon S3 atau mengirim log secara langsung ke Amazon S3 untuk penyimpanan jangka panjang. Menggunakan AWS Glue, Anda dapat menelusuri dan menyiapkan data log Anda di Amazon S3 untuk analitik, serta menyimpan metadata terkait di AWS Glue Data Catalog. Amazon Athena, melalui integrasi native-nya dengan AWS Glue, dapat digunakan untuk menganalisis data log Anda, mengkuerinya menggunakan SQL standar. Menggunakan alat kecerdasan bisnis seperti Amazon Quick, Anda dapat memvisualisasi, menelusuri, dan menganalisis data Anda. Menemukan tren dan peristiwa ketertarikan yang bisa mendorong peningkatan.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk keunggulan operasional.

OPS 11: How do you evolve operations?

Dedicate time and resources for nearly continuous incremental improvement to evolve the effectiveness and efficiency of your operations.

Keberhasilan pengembangan operasi terbentuk dari: peningkatan kecil yang sering, penyediaan lingkungan yang aman dan waktu untuk bereksperimen, mengembangkan, serta menguji peningkatan, serta lingkungan yang mendukung untuk belajar dari kegagalan. Dukungan operasi untuk lingkungan sandbox, pengembangan, pengujian, dan produksi, dengan meningkatkan tingkat kontrol operasi, memfasilitasi pengembangan dan meningkatkan prediktabilitas hasil yang sukses dari perubahan yang diterapkan ke produksi.

Sumber daya

Buka sumber daya berikut untuk mempelajari selengkapnya tentang praktik terbaik kami untuk Keunggulan Operasional.

Dokumentasi

- DevOps dan AWS

Laporan resmi

- Pilar Keunggulan Operasional

Video:

- DevOps di Amazon

Keamanan

Pilar Keamanan berkenaan dengan kemampuan untuk melindungi data, sistem, dan aset untuk memanfaatkan teknologi cloud guna meningkatkan keamanan Anda.

Pilar keamanan memberikan ikhtisar prinsip desain, praktik terbaik, dan pertanyaan. Anda dapat menemukan panduan preskriptif tentang implementasi di laporan resmi Pilar Keamanan.

Topik

- Prinsip desain
- Definisi
- Praktik terbaik
- Sumber daya

Prinsip desain

Terdapat tujuh prinsip desain untuk keamanan di cloud:

- Mengimplementasikan landasan identitas yang kuat: Implementasikan prinsip hak akses paling rendah dan berlakukan pemisahan tugas dengan otorisasi yang sesuai untuk setiap interaksi dengan sumber daya AWS Anda. Pusatkan manajemen identitas, dan targetkan untuk tidak bergantung pada kredensial statis jangka panjang.
- Aktifkan keterlacakan: Pantau, munculkan peringatan, dan audit tindakan serta perubahan dalam lingkungan Anda secara waktu nyata. Integrasikan pengumpulan log dan metrik dengan sistem agar dapat bertindak berdasarkan investigasi yang berjalan otomatis.
- Menerapkan keamanan di semua lapisan: Terapkan pertahanan secara mendalam dengan banyak kontrol keamanan. Terapkan ke semua lapisan (misalnya, edge jaringan, VPC, penyeimbangan beban, setiap layanan komputasi dan instans, sistem operasi, aplikasi, dan kode).

- Mengotomatiskan praktik terbaik keamanan: Mekanisme keamanan berbasis perangkat lunak otomatis meningkatkan kemampuan Anda untuk meningkatkan skala dengan lebih cepat, hemat biaya, dan aman. Ciptakan arsitektur yang aman, termasuk implementasi kontrol yang ditentukan dan dikelola sebagai kode dalam templat yang dikontrol versi.
- Melindungi data bergerak dan data diam: Klasifikasikan data sesuai tingkatan sensitivitasnya dan gunakan mekanisme, seperti enkripsi, tokenisasi, dan kontrol akses jika sesuai.
- Minimalkan campur tangan manusia dari data: Gunakan mekanisme dan alat untuk mengurangi atau meniadakan akses langsung atau pemrosesan data secara manual. Ini akan mengurangi risiko kekeliruan atau perubahan dan kesalahan manusia dalam penanganan data sensitif.
- Bersiap untuk peristiwa keamanan: Bersiaplah menghadapi insiden dengan membentuk manajemen insiden serta proses dan kebijakan investigasi yang selaras dengan kebutuhan organisasi Anda. Jalankan simulasi respons insiden dan gunakan alat dengan otomatisasi untuk mempercepat deteksi, investigasi, dan pemulihan.

Definisi

Terdapat enam area praktik terbaik untuk keamanan di cloud:

- Keamanan
- Identity and Access Management
- Deteksi
- Perlindungan Infrastruktur
- Perlindungan Data
- Respons Insiden

Sebelum merancang beban kerja, Anda harus menerapkan praktik yang berpengaruh terhadap keamanan. Tentukan peran untuk setiap orang. Selain itu, Anda perlu mengidentifikasi insiden keamanan, melindungi sistem dan layanan Anda, serta menjaga kerahasiaan dan integritas data melalui perlindungan data. Anda harus memiliki proses yang terdefinisi dengan baik dan dapat dipraktikkan untuk merespons insiden keamanan. Alat dan teknik ini penting karena dapat mendukung tujuan seperti mencegah kerugian finansial atau mematuhi peraturan yang berlaku.

Model Tanggung Jawab Bersama AWS memungkinkan organisasi yang mengadopsi cloud untuk mencapai tujuan keamanan dan kepatuhan. AWS mengamankan fisik infrastruktur yang mendukung layanan cloud kami, sehingga sebagai pelanggan AWS Anda dapat berfokus pada penggunaan

layanan untuk mencapai tujuan Anda. AWS Cloud juga menyediakan akses yang lebih luas ke data keamanan serta pendekatan otomatis untuk merespons peristiwa keamanan.

Praktik terbaik

Topik

- Keamanan
- Manajemen identitas dan akses
- Deteksi
- Perlindungan infrastruktur
- Perlindungan data
- Respons insiden

Keamanan

Untuk mengoperasikan beban kerja dengan aman, Anda harus menerapkan praktik terbaik yang menyeluruh ke setiap area keamanan. Pilih persyaratan dan proses yang telah Anda tetapkan dalam keunggulan operasional pada tingkat organisasi dan beban kerja, lalu terapkan ke semua area.

Dengan terus mengikuti rekomendasi terbaru dari AWS dan industri serta kecerdasan ancaman, Anda dapat mengembangkan model ancaman dan tujuan kontrol. Mengotomatiskan proses keamanan, pengujian, dan validasi dapat membantu menskalakan operasi keamanan.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk keamanan ini. (Untuk melihat daftar pertanyaan dan praktik terbaik keamanan, buka Lampiran.).

BAG 1: Bagaimana cara mengoperasikan beban kerja dengan aman?

Untuk mengoperasikan beban kerja dengan aman, Anda harus menerapkan praktik terbaik yang menyeluruh ke setiap area keamanan. Pilih persyaratan dan proses yang telah Anda tetapkan dalam keunggulan operasional pada tingkat organisasi dan beban kerja, lalu terapkan ke semua area. Dengan terus mengikuti rekomendasi terbaru dari AWS, sumber industri, serta kecerdasa n ancaman, Anda dapat mengembangkan model ancaman dan tujuan kontrol. Mengotomatiskan proses keamanan, pengujian, dan validasi dapat membantu menskalakan operasi keamanan.

Di AWS, kami menyarankan untuk memisahkan beban kerja per akun berdasarkan fungsi dan persyaratan kepatuhan atau sensitivitas data.

Manajemen identitas dan akses

Manajemen identitas dan akses adalah bagian penting dari program keamanan informasi, memastikan bahwa hanya pengguna dan komponen yang sah dan diautentikasi yang dapat mengakses sumber daya Anda, dan hanya melalui cara yang Anda izinkan. Misalnya, Anda harus menentukan prinsipal (yaitu, akun, pengguna, peran, dan layanan yang dapat melakukan tindakan di akun Anda), membuat kebijakan yang selaras dengan prinsipal ini, dan menerapkan manajemen kredensial yang kuat. Elemen manajemen hak istimewa ini membentuk inti autentikasi dan otorisasi.

Di AWS, manajemen hak istimewa utamanya didukung oleh layanan AWS Manajemen Identitas dan Akses (IAM), yang dapat Anda gunakan untuk mengontrol akses program dan pengguna ke layanan dan sumber daya AWS. Anda harus menerapkan kebijakan yang terperinci, yang memberikan izin kepada pengguna, grup, peran, atau sumber daya. Anda juga dapat mewajibkan penggunaan kata sandi yang kuat, seperti tingkat kesulitan, agar kata sandi tidak digunakan kembali, dan menerapkan autentikasi multi-faktor (MFA). Anda dapat menggunakan federasi dengan layanan direktori yang ada. Untuk beban kerja yang mengharuskan sistem untuk mengakses AWS, IAM memberikan akses aman melalui peran, profil instans, federasi identitas, dan kredensial sementara.

Pertanyaan berikut ini berfokus pada pertimbangan untuk keamanan ini.

BAG 2: Bagaimana cara mengelola identitas untuk orang dan mesin?

Ada dua jenis identitas yang harus Anda kelola ketika mengoperasikan beban kerja AWS yang aman. Memahami jenis identitas yang perlu Anda kelola dan Anda beri akses akan membantu Anda memastikan bahwa suatu identitas dapat mengakses sumber daya yang tepat dalam kondisi yang tepat.

Identitas Manusia: Administrator, developer, operator, serta pengguna akhir Anda memerlukan identitas untuk mengakses lingkungan dan aplikasi AWS. Ini adalah anggota organisasi Anda, atau pengguna eksternal yang berkolaborasi dengan Anda, dan yang berinteraksi dengan sumber daya AWS Anda melalui browser web, aplikasi klien, atau alat baris perintah interaktif.

Identitas Mesin: Aplikasi layanan, alat operasional, dan beban kerja Anda memerlukan identitas untuk membuat permintaan ke layanan AWS, misalnya, untuk membaca data. Identitas ini mencakup mesin yang berjalan di lingkungan AWS Anda, seperti instans Amazon EC2 atau fungsi AWS Lambda. Anda juga dapat mengelola identitas mesin untuk pihak eksternal yang membutuhk

BAG 2: Bagaimana cara mengelola identitas untuk orang dan mesin?

an akses. Selain itu, Anda mungkin juga memiliki mesin di luar AWS yang memerlukan akses ke lingkungan AWS Anda.

BAG 3: Bagaimana cara mengelola izin untuk orang dan mesin?

Kelola izin untuk mengontrol akses untuk identitas orang dan mesin yang memerlukan akses ke AWS dan beban kerja Anda. Izin mengontrol cakupan dan ketentuan akses seseorang.

Kredensial tidak boleh dibagikan ke pengguna atau sistem lain. Akses pengguna harus diberikan menggunakan pendekatan hak akses paling rendah dan praktik terbaik, termasuk wajib menggunakan kata sandi dan MFA. Akses program, termasuk panggilan API ke layanan AWS harus dilakukan menggunakan kredensial sementara dengan hak istimewa terbatas seperti yang dikeluarkan oleh AWS Security Token Service.

AWS menyediakan sumber daya yang dapat membantu mengelola manajemen identitas dan akses. Untuk membantu mempelajari praktik terbaik, jelajahi lab langsung kami tentang mengelola kredensial & autentikasi, mengontrol akses manusia, dan mengontrol akses terprogram.

Deteksi

Anda dapat menggunakan kontrol deteksi untuk mengidentifikasi potensi ancaman atau insiden keamanan. Ini merupakan bagian yang sangat penting dalam kerangka kerja tata kelola dan dapat digunakan untuk mendukung proses yang berkualitas, kewajiban kepatuhan atau hukum, serta upaya identifikasi dan respons terhadap ancaman. Ada beberapa jenis kontrol deteksi. Misalnya, melakukan inventarisasi aset dan detail atributnya mendorong pengambilan keputusan (dan kontrol siklus hidup) yang lebih efektif untuk membantu menetapkan dasar operasional. Anda juga dapat menggunakan audit internal, pemeriksaan kontrol yang terkait dengan sistem informasi, untuk memastikan bahwa praktik telah memenuhi kebijakan serta persyaratan dan bahwa Anda telah menetapkan notifikasi peringatan otomatis sesuai kondisi yang ditentukan. Kontrol ini merupakan faktor reaktif penting yang dapat membantu organisasi Anda mengidentifikasi dan memahami cakupan aktivitas anomali.

Di AWS, Anda dapat menerapkan kontrol deteksi dengan memproses log, peristiwa, dan pemantauan yang memungkinkan audit, analisis otomatis, dan peringatan. Log CloudTrail, panggilan API AWS, serta CloudWatch menyediakan pemantauan metrik dengan memberikan peringatan, dan AWS

Config menyediakan riwayat konfigurasi. Amazon GuardDuty adalah layanan deteksi ancaman terkelola yang terus memantau aktivitas berbahaya atau tidak sah untuk membantu melindungi akun dan beban kerja AWS Anda. Log tingkat layanan juga tersedia, misalnya, Anda dapat menggunakan Amazon Simple Storage Service (Amazon S3) untuk membuat log permintaan akses.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk keamanan ini.

BAG 4: Bagaimana cara mendeteksi dan menyelidiki peristiwa keamanan?

Catat dan analisis peristiwa dari log dan metrik untuk mendapatkan visibilitas. Ambil tindakan atas peristiwa keamanan dan potensi ancaman untuk membantu mengamankan beban kerja Anda.

Manajemen log sangat penting dalam beban kerja Well-Architected. Mulai dari untuk alasan keamanan atau forensik, hingga persyaratan hukum atau regulasi. Anda harus menganalisis log dan meresponsnya agar Anda dapat mengidentifikasi potensi insiden keamanan. AWS menyediakan fungsionalitas yang memudahkan manajemen log dengan memberikan kemampuan untuk menentukan siklus hidup retensi data atau menentukan tempat penyimpanan, pengarsipan, dan penghapusan data. Hal ini memudahkan penanganan data yang dapat diprediksi, andal, dan lebih hemat biaya.

Perlindungan infrastruktur

Perlindungan infrastruktur berkenaan dengan metodologi kontrol, seperti pertahanan mendalam, yang diperlukan untuk memenuhi praktik terbaik dan kewajiban organisasi atau peraturan. Penggunaan metodologi ini vital untuk keberhasilan dan keberlangsungan operasi, baik di cloud maupun on-premise.

Di AWS, Anda dapat menerapkan stateful dan stateless packet inspection, baik menggunakan teknologi AWS maupun produk dan layanan partner yang tersedia melalui AWS Marketplace. Anda harus menggunakan Amazon Virtual Private Cloud (Amazon VPC) untuk membuat lingkungan privat yang aman dan dapat diskalakan guna menentukan topologi—termasuk gateway, tabel perutean, serta subnet publik dan privat.

Pertanyaan berikut ini berfokus pada pertimbangan untuk keamanan ini.

BAG 5: Bagaimana cara melindungi sumber daya jaringan?

Setiap beban kerja yang memiliki konektivitas jaringan, baik internet maupun jaringan privat, memerlukan beberapa lapisan pertahanan untuk membantu melindungi dari ancaman eksternal dan internal di jaringan.

BAG 6: Bagaimana cara melindungi sumber daya komputasi?

Sumber daya komputasi di beban kerja Anda memerlukan beberapa lapisan pertahanan untuk membantu melindungi dari ancaman eksternal dan internal. Sumber daya komputasi meliputi instans EC2, kontainer, fungsi AWS Lambda, layanan basis data, perangkat IoT, dan banyak lagi.

Sangat disarankan untuk memberikan banyak lapisan keamanan di semua jenis lingkungan. Untuk perlindungan infrastruktur, banyak konsep dan metode yang dapat digunakan untuk model cloud dan on-premise. Menerapkan perlindungan batas, memantau titik masuk dan keluar, serta pencatatan log, pemantauan, dan peringatan yang komprehensif sangat penting dalam rencana keamanan informasi yang efektif.

Pelanggan AWS dapat menyesuaikan atau memperkuat konfigurasi Amazon Elastic Compute Cloud (Amazon EC2), kontainer Amazon Elastic Container Service (Amazon ECS), atau instans AWS Elastic Beanstalk, dan menyimpan konfigurasi ini ke Amazon Machine Image (AMI) tetap. Kemudian, saat dipicu oleh Auto Scaling atau diluncurkan secara manual, semua server virtual (instans) baru yang diluncurkan dengan AMI ini menerima konfigurasi yang diperkuat ini.

Perlindungan data

Sebelum merancang sistem apa pun, praktik dasar yang memengaruhi keamanan harus diterapkan. Misalnya, klasifikasi data menjadi cara untuk mengategorikan data organisasi berdasarkan tingkat sensitivitas, dan enkripsi melindungi data dengan membuatnya tidak dapat dikenali oleh akses tidak sah. Alat dan teknik ini penting karena dapat mendukung tujuan seperti mencegah kerugian finansial atau mematuhi peraturan yang berlaku.

Berikut adalah praktik yang mendukung perlindungan data di AWS:

- Sebagai pelanggan AWS, kontrol data sepenuhnya berada di tangan Anda.

- AWS memudahkan enkripsi data dan pengelolaan kunci, termasuk rotasi kunci rutin yang dapat dengan mudah diotomatiskan oleh AWS atau Anda kelola sendiri.
- Tersedia log mendetail yang berisi konten penting seperti perubahan dan akses file.
- AWS telah mendesain sistem penyimpanan yang sangat tangguh. Contohnya, Amazon S3 Standard, S3 Standard–IA, S3 One Zone-IA, dan Amazon Glacier didesain untuk memberikan 99,999999999% ketahanan objek dalam setahun. Tingkat ketahanan ini sesuai dengan perkiraan rata-rata penurunan tahunan pada objek sebesar 0,000000001%.
- Versioning, yang dapat menjadi bagian dari proses pengelolaan siklus hidup data yang lebih besar, dapat melindungi dari penimpaan, penghapusan, dan kerusakan serupa yang tidak disengaja.
- AWS tidak pernah memindahkan data dari satu Wilayah ke Wilayah lain. Konten yang ditempatkan pada suatu Wilayah tidak akan berpindah kecuali Anda mengaktifkan fitur atau menggunakan layanan untuk memindahkannya.

Pertanyaan berikut ini berfokus pada pertimbangan untuk keamanan ini.

BAG 7: Bagaimana cara mengklasifikasikan data?

Klasifikasi menjadi cara untuk mengategorikan data berdasarkan tingkat kepentingan dan sensitivi tas untuk membantu Anda menentukan kontrol retensi dan perlindungan yang sesuai.

BAG 8: Bagaimana cara melindungi data diam?

Lindungi data diam dengan mengimplementasikan beberapa kontrol untuk mengurangi risiko akses tidak sah atau kesalahan penanganan.

BAG 9: Bagaimana cara melindungi data bergerak?

Lindungi data bergerak dengan mengimplementasikan beberapa kontrol untuk mengurangi risiko akses tidak sah atau kehilangan data.

AWS menyediakan banyak alat untuk mengenkripsi data diam dan bergerak. Kami menyertakan fitur-fitur ke layanan kami yang mampu memudahkan enkripsi data. Sebagai contoh, kami telah

menerapkan enkripsi di sisi server (SSE) untuk Amazon S3 guna memudahkan penyimpanan data yang dienkripsi. Anda juga dapat menyerahkan semua proses enkripsi dan pembatalan enkripsi HTTPS (dikenal sebagai penghentian SSL) kepada Elastic Load Balancing (ELB).

Respons insiden

Dengan kontrol deteksi dan preventif yang sangat matang sekalipun, organisasi Anda harus tetap menyiapkan proses untuk merespons dan memitigasi potensi dampak insiden keamanan. Arsitektur beban kerja sangat berpengaruh pada kemampuan tim Anda untuk beroperasi secara efektif selama insiden, untuk mengisolasi atau membatasi sistem, serta untuk memulihkan operasi ke kondisi yang baik. Menetapkan alat dan akses sebelum terjadi insiden keamanan, lalu secara rutin melatih respons insiden melalui game day akan membantu memastikan bahwa arsitektur Anda dapat melakukan penyelidikan dan pemulihan tepat waktu.

Berikut praktik yang mendukung respons insiden yang efektif di AWS:

- Tersedia log mendetail yang berisi konten penting seperti perubahan dan akses file.
- Peristiwa dapat diproses secara otomatis dan memicu alat yang mengotomatiskan respons melalui penggunaan API AWS.
- Anda dapat membuat “clean room” dan alat terlebih dahulu menggunakan AWS CloudFormation. Dengan demikian, Anda dapat melakukan forensik di lingkungan yang aman dan terisolasi.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk keamanan ini.

BAG 10: Bagaimana cara mengantisipasi, merespons, dan pulih dari insiden?

Persiapan sangat penting dalam penyelidikan, respons, dan pemulihan peristiwa keamanan yang tepat waktu dan efektif guna membantu meminimalkan gangguan terhadap organisasi Anda.

Pastikan Anda dapat dengan cepat memberikan akses untuk tim keamanan, dan mengotomatiskan isolasi instans serta pencatatan data status untuk forensik.

Sumber daya

Lihat sumber daya berikut untuk mempelajari selengkapnya tentang praktik terbaik kami untuk Keamanan.

Dokumentasi

- Keamanan AWS Cloud
- Kepatuhan AWS
- Blog Keamanan AWS

Laporan Resmi

- Pilar Keamanan
- Ikhtisar Keamanan AWS
- Risiko dan Kepatuhan AWS

Video

- AWS Security State of the Union
- Ikhtisar Tanggung Jawab Bersama

Keandalan

Pilar keandalan berkenaan dengan kemampuan beban kerja untuk menjalankan fungsinya dengan benar dan konsisten sesuai ekspektasi. Ini termasuk kemampuan untuk mengoperasikan dan menguji beban kerja di seluruh siklus hidupnya. Laporan resmi ini berisi panduan praktik terbaik yang mendalam untuk mengimplementasikan beban kerja yang andal di AWS.

Pilar keandalan memberikan ikhtisar prinsip desain, praktik terbaik, dan pertanyaan. Anda dapat menemukan panduan preskriptif tentang implementasi di Laporan Resmi Pilar Keandalan.

Topik

- Prinsip desain
- Definisi
- Praktik terbaik
- Sumber daya

Prinsip desain

Terdapat lima prinsip desain untuk keandalan di cloud:

- Pulihkan secara otomatis dari kegagalan: Dengan memantau beban kerja untuk indikator kinerja utama (KPI), Anda dapat memicu otomatisasi ketika ambang batas dilanggar. KPI ini harus menjadi ukuran dari nilai bisnis, bukan menjadi ukuran aspek teknis operasi layanan. Hal ini memungkinkan notifikasi otomatis dan pelacakan kegagalan, serta proses pemulihan otomatis yang menangani atau memperbaiki kegagalan. Dengan otomatisasi yang lebih canggih, antisipasi dan perbaikan kegagalan dapat dilakukan sebelum kegagalan terjadi.
- Uji prosedur pemulihan: Dalam lingkungan on-premise, pengujian biasanya dijalankan untuk membuktikan bahwa beban kerja bekerja dalam skenario tertentu. Pengujian tidak digunakan untuk memvalidasi strategi pemulihan. Di cloud, Anda dapat menguji bagaimana beban kerja gagal, dan dapat memvalidasi prosedur pemulihan. Gunakan otomatisasi untuk memicu berbagai kegagalan atau untuk membuat ulang skenario yang mengarah pada kegagalan sebelumnya. Pendekatan ini memperlihatkan jalur kegagalan yang dapat diuji dan diperbaiki sebelum skenario kegagalan benar-benar terjadi, dan juga mengurangi risiko.
- Skalakan secara horizontal untuk meningkatkan ketersediaan beban kerja agregat: Ganti satu sumber daya besar dengan beberapa sumber daya kecil untuk mengurangi dampak kegagalan tunggal terhadap beban kerja keseluruhan. Distribusikan permintaan ke beberapa sumber daya yang lebih kecil untuk memastikan tidak adanya kesamaan titik kegagalan.
- Jangan menebak kapasitas: Penyebab umum kegagalan dalam beban kerja on-premise adalah saturasi sumber daya, yaitu ketika permintaan yang ditempatkan di beban kerja melebihi kapasitas beban kerjanya (ini sering kali menjadi sasaran dari serangan denial of service). Di cloud, Anda dapat memantau pemanfaatan beban kerja dan permintaan, serta mengotomatiskan penambahan atau penghapusan sumber daya untuk mempertahankan tingkat keoptimalan guna memenuhi permintaan tanpa kekurangan atau kelebihan penyediaan. Batasan tentunya masih ada, tetapi sebagian kuota dapat dikontrol dan sebagian lainnya dapat dikelola (lihat Mengelola Service Quotas dan Pembatasan).
- Kelola perubahan dalam otomatisasi: Perubahan untuk infrastruktur Anda harus dibuat menggunakan otomatisasi. Perubahan yang harus dikelola mencakup perubahan untuk otomatisasi, yang kemudian dapat dilacak dan ditinjau.

Definisi

Terdapat empat area praktik terbaik untuk keandalan di cloud:

- Fondasi
- Arsitektur Beban Kerja
- Manajemen Perubahan
- Manajemen Kegagalan

Untuk mencapai keandalan, Anda harus mengawalinya dengan fondasi — sebuah lingkungan dengan kuota layanan dan topologi jaringan yang mengakomodasi beban kerja. Arsitektur beban kerja sistem terdistribusi harus didesain untuk mencegah dan memitigasi kegagalan. Beban kerja harus menangani perubahan dalam permintaan dan persyaratan, serta harus didesain untuk mendeteksi kegagalan dan melakukan pemulihan mandiri secara otomatis.

Praktik terbaik

Topik

- Fondasi
- Arsitektur beban kerja
- Manajemen perubahan
- Manajemen kegagalan

Fondasi

Persyaratan mendasar memiliki cakupan yang lebih luas dari satu beban kerja atau proyek. Sebelum merancang sistem apa pun, persyaratan mendasar yang memengaruhi keandalan harus diterapkan. Misalnya, Anda harus memiliki bandwith jaringan yang memadai untuk pusat data Anda.

Dengan AWS, sebagian besar persyaratan mendasar ini sudah digabungkan atau ditangani sesuai kebutuhan. Cloud didesain agar menjadi hampir tidak terbatas, ini adalah tanggung jawab AWS untuk memenuhi persyaratan jaringan dan kapasitas komputasi yang memadai, agar Anda dapat dengan leluasa mengubah alokasi dan ukuran sumber daya sesuai permintaan.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk keandalan. (Untuk melihat daftar pertanyaan keandalan dan praktik terbaik, buka Lampiran.).

REL 1: Bagaimana cara mengelola pembatasan dan kuota layanan?

Untuk arsitektur beban kerja berbasis cloud, terdapat kuota layanan (yang juga disebut sebagai batas layanan). Kuota ini ada agar penyediaan sumber daya tidak melebihi yang Anda butuhkan dan untuk membatasi tingkat permintaan di operasi API serta melindungi layanan dari penyalahg unaan. Selain itu, terdapat beberapa hal yang membatasi sumber daya, misalnya, rasio yang dapat menurunkan bit di kabel serat optik, atau jumlah penyimpanan di dalam disk fisik.

REL 2: Bagaimana cara merencanakan topologi jaringan Anda?

Beban kerja sering kali ada di beberapa lingkungan. Termasuk beberapa lingkungan cloud (baik yang dapat diakses publik maupun privat) dan kemungkinan juga di infrastruktur pusat data Anda yang ada. Rencana harus mencakup pertimbangan jaringan seperti konektivitas di dalam dan antarsistem, manajemen alamat IP publik, manajemen alamat IP privat, dan resolusi nama domain.

Untuk arsitektur beban kerja berbasis cloud, terdapat kuota layanan (yang juga disebut sebagai batas layanan). Kuota ini ada untuk agar penyediaan sumber daya tidak melebihi yang Anda butuhkan dan untuk membatasi tingkat permintaan di operasi API serta melindungi layanan dari penyalahgunaan. Beban kerja sering kali ada di beberapa lingkungan. Anda harus memantau dan mengelola kuota tersebut untuk semua lingkungan beban kerja. Termasuk beberapa lingkungan cloud (baik yang dapat diakses secara publik maupun privat) dan bisa juga termasuk infrastruktur pusat data Anda yang ada. Rencana harus mencakup pertimbangan jaringan, seperti konektivitas di dalam sistem dan antarsistem, manajemen alamat IP publik, manajemen alamat IP privat, dan resolusi nama domain.

Arsitektur beban kerja

Beban kerja yang andal dimulai dengan desain perangkat lunak dan infrastruktur yang diputuskan sejak awal. Pilihan arsitektur Anda akan memengaruhi perilaku beban kerja Anda di semua pilar Well-Architected. Untuk keandalan, terdapat beberapa pola tertentu yang harus diikuti.

Dengan AWS, developer beban kerja dapat memilih bahasa dan teknologi yang akan mereka digunakan. SDK AWS menyediakan API dengan bahasa khusus untuk layanan AWS, sehingga pengodean menjadi lebih mudah. SDK ini, dengan pilihan banyak bahasa, memungkinkan developer untuk mengimplementasikan praktik terbaik keandalan yang tercantum di sini. Para developer juga

dapat membaca dan belajar dari cara Amazon membangun dan mengoperasikan perangkat lunak, di dalam Amazon Builders' Library.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk keandalan.

REL 3: Bagaimana cara mendesain arsitektur layanan beban kerja Anda?

Bangun beban kerja yang mudah diskalakan dan andal menggunakan arsitektur berorientasi layanan (SOA) atau arsitektur layanan mikro. Arsitektur berorientasi layanan (SOA) merupakan praktik pembuatan komponen perangkat lunak yang dapat digunakan ulang lewat antarmuka layanan. Arsitektur layanan mikro melakukan hal yang lebih dengan membuat komponen menjadi lebih kecil dan lebih sederhana.

REL 4: Bagaimana cara mendesain interaksi di dalam sistem terdistribusi untuk mencegah kegagalan?

Sistem terdistribusi mengandalkan jaringan komunikasi untuk membuat interkoneksi komponen, seperti server atau layanan. Beban kerja Anda harus beroperasi secara andal walaupun terdapat latensi atau kehilangan data di jaringannya. Komponen sistem terdistribusi harus beroperasi tanpa memberikan dampak secara negatif kepada komponen dan beban kerja yang lain. Praktik terbaik ini mencegah kegagalan dan meningkatkan waktu rata-rata antara kegagalan (MTBF).

REL 5: Bagaimana cara mendesain interaksi dalam sistem terdistribusi untuk mitigasi atau bertahan dari kegagalan?

Sistem terdistribusi mengandalkan jaringan komunikasi untuk membuat interkoneksi komponen (seperti server atau layanan). Beban kerja Anda harus beroperasi secara andal walaupun terdapat latensi atau kehilangan data di jaringannya. Komponen sistem terdistribusi harus beroperasi tanpa memberikan dampak secara negatif kepada komponen dan beban kerja yang lain. Berbagai praktik terbaik ini memungkinkan beban kerja bertahan dari tekanan atau kegagalan, pulih dengan lebih cepat, serta memitigasi dampak gangguan tersebut. Hasilnya adalah peningkatan waktu rata-rata untuk pemulihan (MTTR).

Manajemen perubahan

Perubahan beban kerja atau lingkungannya harus diantisipasi dan diakomodasi guna mencapai operasi beban kerja yang andal. Perubahan mencakup semua yang terjadi ke beban kerja seperti lonjakan permintaan, serta perubahan dari dalam, seperti deployment fitur dan patch keamanan.

Menggunakan AWS, Anda dapat memantau perilaku beban kerja dan mengotomatiskan respons untuk KPI. Misalnya, beban kerja dapat menambahkan server tambahan seiring dengan bertambahnya pengguna dalam beban kerja. Anda dapat mengatur pemberian izin untuk siapa saja yang dapat membuat perubahan beban kerja dan mengaudit riwayat perubahan ini.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk keandalan.

REL 6: Bagaimana cara memantau sumber daya beban kerja?

Log dan metrik merupakan alat yang sangat andal untuk mendapatkan wawasan tentang kondisi beban kerja. Anda dapat mengonfigurasikan beban kerja untuk memantau log dan metrik serta mengirimkan notifikasi ketika ambang batas terlampaui atau terjadi peristiwa yang signifikan. Dengan pemantauan, beban kerja Anda dapat mengidentifikasi saat ambang batas kinerja rendah terlampaui atau kegagalan terjadi, sehingga dapat bereaksi dengan melakukan pemulihan otomatis.

REL 7: Bagaimana cara merancang beban kerja Anda agar dapat beradaptasi dengan perubahan dalam permintaan?

Beban kerja yang dapat diskalakan memberikan elastisitas untuk menambahkan atau menghapus sumber daya secara otomatis sehingga sangat sesuai dengan permintaan yang sedang berjalan pada titik waktu tertentu.

REL 8: Bagaimana cara mengimplementasikan perubahan?

Perubahan terkontrol diperlukan untuk melakukan deployment fungsionalitas baru, serta memastikan bahwa beban kerja dan lingkungan operasi menjalankan perangkat lunak yang dikenal dan dapat di-patch atau diganti dengan cara yang dapat diprediksi. Jika perubahan-perubahan ini tidak terkontrol, akan sulit untuk memprediksi efek dari perubahan-perubahan tersebut, atau untuk mengatasi masalah yang ditimbulkannya.

Ketika Anda merancang beban kerja agar otomatis menambahkan dan menghapus sumber daya sebagai respons atas perubahan permintaan, ini tidak hanya meningkatkan keandalan tetapi juga memastikan bahwa keberhasilan bisnis tidak menambah beban. Dengan menerapkan pemantauan, tim Anda secara otomatis dapat mengetahui ketika KPI menyimpang dari perilaku yang diharapkan. Pencatatan otomatis log perubahan lingkungan memungkinkan Anda untuk mengaudit dan dengan cepat mengidentifikasi tindakan yang mungkin berdampak terhadap keandalan. Kontrol dalam manajemen perubahan memastikan bahwa Anda dapat menerapkan aturan untuk memberikan keandalan yang Anda butuhkan.

Manajemen kegagalan

Dalam sistem apa pun yang memiliki kompleksitas wajar, kegagalan diperkirakan akan terjadi. Keandalan hanya dapat terwujud jika beban kerja Anda dapat mengidentifikasi kegagalan yang terjadi dan mengambil tindakan untuk menghindari dampaknya terhadap ketersediaan. Beban kerja harus mampu bertahan dari kegagalan serta secara otomatis memperbaiki masalah.

Dengan AWS, Anda dapat memanfaatkan otomatisasi untuk memberikan reaksi terhadap data pemantauan. Misalnya, ketika metrik tertentu melewati ambang batas, Anda dapat memicu tindakan otomatis untuk memperbaiki masalah. Selain itu, daripada berupaya untuk mendiagnosis dan memperbaiki sumber daya gagal yang merupakan bagian dari lingkungan produksi, Anda dapat menggantinya dengan yang baru dan melakukan analisis terhadap sumber daya yang gagal tersebut di luar jaringan. Karena cloud memungkinkan Anda untuk menggunakan versi sementara dengan harga yang rendah, Anda dapat menggunakan pengujian otomatis untuk memverifikasi proses pemulihan penuh.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk keandalan.

REL 9: Bagaimana cara mencadangkan data?

Cadangkan data, aplikasi, dan konfigurasi untuk memenuhi persyaratan untuk sasaran waktu pemulihan (RTO) dan sasaran titik pemulihan (RPO).

REL 10: Bagaimana cara menggunakan isolasi kesalahan untuk melindungi beban kerja Anda?

Batas isolasi kesalahan membatasi efek kegagalan di dalam beban kerja untuk jumlah komponen yang terbatas. Komponen di luar batas ini tidak terpengaruh oleh kegagalan tersebut. Dengan beberapa batas isolasi kesalahan, Anda dapat membatasi dampak pada beban kerja Anda.

REL 11: Bagaimana cara mendesain beban kerja Anda agar bertahan dalam kegagalan komponen?

Beban kerja dengan persyaratan ketersediaan tinggi dan waktu rata-rata untuk pemulihan (MTTR) rendah harus dirancang agar tangguh.

REL 12: Bagaimana cara menguji keandalan?

Setelah Anda merancang beban kerja agar tangguh terhadap tekanan produksi, pengujian adalah satu-satunya cara untuk memastikan bahwa beban kerja akan beroperasi sesuai desain, dengan ketangguhan yang diharapkan.

REL 13: Bagaimana cara merencanakan pemulihan bencana (DR)?

Memiliki cadangan dan komponen beban kerja berlebih adalah awal strategi DR Anda. RTO dan RPO adalah sasaran untuk pemulihan beban kerja Anda. Atur hal ini berdasarkan kebutuhan bisnis. Implementasikan strategi sesuai sasaran ini, dengan mempertimbangkan lokasi dan fungsi data serta sumber daya beban kerja. Probabilitas gangguan dan biaya pemulihan juga merupakan faktor penting yang membantu memahami nilai bisnis dari penyediaan pemulihan bencana untuk beban kerja.

Cadangkan data dan uji file cadangan Anda secara rutin untuk memastikan bahwa Anda dapat memulihkan kesalahan fisik dan logisnya. Kunci untuk mengelola kegagalan adalah pengujian beban kerja secara rutin dan otomatis dengan cara menyebabkan kegagalan, kemudian mengamati bagaimana pemulihan dilakukan. Lakukan hal ini secara terjadwal serta pastikan bahwa pengujian serupa juga dilakukan setelah perubahan beban kerja yang signifikan. Lacak KPI secara aktif, serta sasaran waktu pemulihan (RTO) dan sasaran titik pemulihan (RPO), untuk mengukur ketangguhan beban kerja (terutama dalam skenario uji kegagalan). Pelacakan KPI akan membantu Anda mengidentifikasi dan memitigasi titik kegagalan tunggal. Sasarannya adalah untuk menguji secara keseluruhan proses pemulihan beban kerja Anda sehingga Anda yakin bahwa Anda dapat memulihkan semua data dan terus melayani pelanggan, bahkan saat menghadapi masalah yang berlanjut. Proses pemulihan Anda harus terlatih dengan baik sebagaimana proses produksi normal Anda.

Sumber daya

Lihat referensi berikut untuk mempelajari selengkapnya tentang praktik terbaik kami untuk Keandalan.

Dokumentasi

- Dokumentasi AWS
- Infrastruktur Global AWS
- AWS Auto Scaling: Cara Kerja Rencana Penskalaan
- Apa Itu AWS Backup?

Laporan Resmi

- Pilar Keandalan: AWS Well-Architected
- Mengimplementasikan Layanan Mikro di AWS

Efisiensi kinerja

Pilar Efisiensi Kinerja menyertakan kemampuan untuk menggunakan sumber daya komputasi dengan efisien agar memenuhi persyaratan sistem, dan untuk memelihara efisiensi tersebut seiring dengan perubahan permintaan dan perkembangan teknologi.

Pilar efisiensi kinerja memberikan gambaran umum tentang prinsip, praktik terbaik, dan pertanyaan desain. Anda dapat menemukan panduan preskriptif tentang implementasi di dokumen resmi Pilar Efisiensi Kinerja.

Topik

- Prinsip desain
- Definisi
- Praktik terbaik
- Sumber daya

Prinsip desain

Ada lima prinsip desain untuk efisiensi kinerja di cloud:

- Demokratisasikan teknologi canggih: Jadikan implementasi teknologi canggih lebih lancar untuk tim Anda dengan mendelegasikan tugas kompleks ke vendor cloud Anda. Daripada bertanya kepada tim IT Anda tentang hosting dan menjalankan teknologi baru, pertimbangkan untuk memanfaatkan teknologi sebagai layanan. Misalnya, basis data NoSQL, transkode media, dan machine learning merupakan teknologi yang memerlukan keahlian khusus. Di cloud, teknologi ini menjadi layanan yang dapat digunakan tim Anda, sehingga mereka dapat berfokus pada pengembangan produk, bukan penyediaan dan manajemen sumber daya.
- Menjangkau dunia dalam hitungan menit: Men-deploy beban kerja ke beberapa Wilayah AWS di seluruh dunia untuk menyediakan latensi yang lebih rendah dan pengalaman yang lebih baik bagi pelanggan dengan biaya minimal.
- Menggunakan arsitektur nirserver: Arsitektur nirserver akan meniadakan perlunya menjalankan dan memelihara server fisik untuk aktivitas komputasi tradisional. Misalnya, layanan penyimpanan nirserver dapat bertindak sebagai situs web statis (tanpa memerlukan server web) dan layanan peristiwa dapat melakukan hosting kode. Dengan demikian, beban operasional untuk mengelola server fisik tidak lagi ada, dan biaya transaksional berkurang karena layanan terkelola dioperasikan pada skala cloud.
- Bereksperimen lebih sering: Dengan sumber daya virtual dan otomatis, Anda dapat melakukan pengujian komparatif dengan cepat menggunakan berbagai tipe instans, penyimpanan, atau konfigurasi.
- Mempertimbangkan simpati mekanis: Pahami cara layanan cloud digunakan dan selalu gunakan pendekatan teknologi yang selaras dengan tujuan beban kerja Anda. Misalnya, pertimbangkan pola akses data saat Anda memilih pendekatan basis data atau penyimpanan.

Definisi

Berikut lima area praktik terbaik untuk efisiensi kinerja di cloud:

- Pemilihan arsitektur
- Komputasi dan perangkat keras
- Manajemen data
- Jaringan dan pengiriman konten
- Proses dan budaya

Gunakan pendekatan yang didorong data untuk membangun arsitektur dengan kinerja tinggi. Kumpulkan data tentang semua aspek arsitektur, dari desain tingkat tinggi hingga pemilihan dan konfigurasi jenis sumber daya.

Meninjau pilihan Anda secara rutin akan memastikan bahwa Anda memanfaatkan Cloud AWS yang terus berkembang. Pemantauan akan memastikan Anda mengetahui adanya penyimpangan dari kinerja yang diharapkan. Buat kompensasi dalam arsitektur untuk meningkatkan kinerja, seperti menggunakan kompresi atau caching, atau persyaratan konsistensi yang lebih fleksibel.

Praktik terbaik

Topik

- Pemilihan arsitektur
- Komputasi dan perangkat keras
- Manajemen data
- Jaringan dan pengiriman konten
- Proses dan budaya

Pemilihan arsitektur

Solusi yang optimal bervariasi untuk beban kerja tertentu, dan solusi sering kali menggabungkan beberapa pendekatan. Beban kerja Well-Architected menggunakan beberapa solusi dan memungkinkan berbagai fitur guna meningkatkan kinerja.

Sumber daya AWS tersedia dalam berbagai jenis dan konfigurasi, sehingga memudahkan Anda menemukan pendekatan yang sesuai kebutuhan. Anda juga dapat menemukan opsi yang tidak mudah dicapai dengan infrastruktur on-premise. Misalnya, layanan terkelola seperti Amazon DynamoDB menyediakan basis data NoSQL terkelola penuh dengan latensi satu digit milidetik pada skala berapa pun.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk efisiensi kinerja. (Untuk melihat daftar pertanyaan dan praktik terbaik efisiensi kinerja, buka Appendix.).

PERF 1: How do you select appropriate cloud resources and architecture patterns for your workload?

Often, multiple approaches are required for more effective performance across a workload. Well-Architected systems use multiple solutions and features to improve performance.

Komputasi dan perangkat keras

Pilihan komputasi yang optimal untuk beban kerja tertentu bervariasi berdasarkan desain aplikasi, pola penggunaan, dan pengaturan konfigurasi. Arsitektur dapat menggunakan pilihan komputasi yang berbeda untuk berbagai komponen, dan memungkinkan fitur yang berbeda untuk meningkatkan kinerja. Memilih pilihan komputasi yang salah untuk arsitektur dapat menyebabkan efisiensi kinerja menjadi lebih rendah.

Di AWS, komputasi tersedia dalam tiga bentuk: instans, kontainer, dan fungsi:

- Instans adalah server tervirtualisasi, yang memungkinkan Anda mengubah kemampuannya hanya dengan menekan tombol atau melakukan panggilan API. Karena keputusan sumber daya di cloud tidaklah tetap, Anda dapat bereksperimen dengan jenis server yang berbeda-beda. Di AWS, instans server virtual ini berasal dari kelompok dan ukuran yang berbeda, dan menawarkan berbagai kemampuan, termasuk solid-state drive (SSD) dan unit pemrosesan grafis (GPU).
- Kontainer adalah metode virtualisasi sistem operasi yang memungkinkan Anda menjalankan aplikasi dan dependensinya dalam proses yang terisolasi sumber daya. AWS Fargate adalah komputasi nirserver untuk kontainer atau Amazon EC2 yang dapat digunakan jika Anda memerlukan kontrol atas penginstalan, konfigurasi, dan manajemen lingkungan komputasi. Anda juga dapat memilih dari berbagai platform pengaturan kontainer: Amazon Elastic Container Service (ECS) atau Amazon Elastic Kubernetes Service (EKS).
- Fungsi mengabstraksi lingkungan yang dijalankan dari kode yang ingin Anda terapkan. Misalnya, AWS Lambda memungkinkan Anda menjalankan kode tanpa menjalankan instans.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk efisiensi kinerja.

PERF 2: How do you select and use compute resources in your workload?

The more efficient compute solution for a workload varies based on application design, usage patterns, and configuration settings. Architectures can use different compute solutions for various

PERF 2: How do you select and use compute resources in your workload?

components and turn on different features to improve performance. Selecting the wrong compute solution for an architecture can lead to lower performance efficiency.

Manajemen data

Solusi manajemen data yang optimal untuk sistem tertentu bervariasi berdasarkan jenis data (blok, file, atau objek), pola akses (acak atau berurutan), throughput yang diperlukan, frekuensi akses (online, offline, arsip), frekuensi pembaruan (WORM, dinamis), dan ketersediaan serta batasan daya tahan. Beban kerja Well-Architected menggunakan penyimpanan data yang dibuat khusus yang memungkinkan berbagai fitur untuk meningkatkan kinerja.

Di AWS, penyimpanan tersedia dalam tiga bentuk: objek, blok, dan file:

- Penyimpanan objek menyediakan platform yang dapat diskalakan dan berdaya tahan agar data dapat diakses dari lokasi internet mana pun untuk konten buatan pengguna, arsip aktif, komputasi nirserver, penyimpanan Big Data, atau pencadangan dan pemulihan. Amazon Simple Storage Service (Amazon S3) adalah layanan penyimpanan objek yang menawarkan skalabilitas, ketersediaan data, keamanan, dan kinerja yang terdepan dalam industri. Amazon S3 didesain untuk ketahanan hingga 99,999999999% (11 angka 9), dan menyimpan data jutaan aplikasi dari berbagai perusahaan di seluruh dunia.
- Penyimpanan blok menyediakan penyimpanan blok dengan ketersediaan tinggi, konsisten, dan berlatensi rendah untuk setiap host virtual dan serupa dengan penyimpanan terpasang langsung (DAS) atau Storage Area Network (SAN). Amazon Elastic Block Store (Amazon EBS) didesain untuk beban kerja yang memerlukan penyimpanan persisten yang dapat diakses oleh instans EC2 yang membantu Anda menyesuaikan aplikasi dengan kapasitas, kinerja, dan biaya penyimpanan yang tepat.
- Penyimpanan file menyediakan akses ke sistem file bersama di berbagai sistem. Solusi penyimpanan file seperti Amazon Elastic File System (Amazon EFS) ideal untuk kasus penggunaan seperti repositori konten besar, lingkungan pengembangan, penyimpanan media, atau direktori beranda pengguna. Amazon FSx menjadikannya efisien dan hemat biaya untuk meluncurkan dan menjalankan sistem file populer sehingga Anda dapat memanfaatkan rangkaian fitur yang kaya dan kinerja cepat dari sistem file open source dan berlisensi komersial yang banyak digunakan.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk efisiensi kinerja.

PERF 3: How do you store, manage, and access data in your workload?

The more efficient storage solution for a system varies based on the kind of access operation (block, file, or object), patterns of access (random or sequential), required throughput, frequency of access (online, offline, archival), frequency of update (WORM, dynamic), and availability and durability constraints. Well-architected systems use multiple storage solutions and turn on different features to improve performance and use resources efficiently.

Jaringan dan pengiriman konten

Solusi jaringan optimal untuk beban kerja bervariasi berdasarkan latensi, persyaratan throughput, jitter, dan bandwith. Batas fisik, seperti sumber daya on-premise atau pengguna, menentukan opsi lokasi. Batas-batas ini dapat diimbangi dengan penempatan sumber daya atau lokasi edge.

Di AWS, jaringan dibuat menjadi virtual dan tersedia dalam berbagai jenis dan konfigurasi yang berbeda-beda. Hal ini membuatnya lebih mudah untuk disesuaikan dengan kebutuhan jaringan Anda. AWS menawarkan fitur produk (misalnya, Enhanced Networking, instans yang dioptimalkan Amazon EC2, akselerasi transfer Amazon S3, dan Amazon CloudFront yang dinamis) untuk mengoptimalkan lalu lintas jaringan. AWS juga menawarkan fitur jaringan (misalnya perutean latensi Amazon Route 53, endpoint Amazon VPC, AWS Direct Connect, dan AWS Global Accelerator) untuk mengurangi jarak jaringan atau jitter.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk efisiensi kinerja.

PERF 4: How do you select and configure networking resources in your workload?

This question includes guidance and best practices to design, configure, and operate efficient networking and content delivery solutions in the cloud.

Proses dan budaya

Saat merancang beban kerja, ada prinsip dan praktik yang dapat Anda adopsi untuk membantu Anda menjalankan beban kerja cloud berkinerja tinggi yang efisien dengan lebih baik. Untuk mengadopsi budaya yang mendorong efisiensi kinerja beban kerja cloud, pertimbangkan prinsip dan praktik utama berikut.

Pertimbangkan prinsip-prinsip utama berikut untuk membangun budaya ini:

- Infrastruktur sebagai kode: Tentukan infrastruktur Anda sebagai kode menggunakan pendekatan seperti templat AWS CloudFormation. Penggunaan templat memungkinkan Anda untuk menempatkan infrastruktur di kontrol sumber bersama dengan konfigurasi dan kode aplikasi Anda. Ini memungkinkan Anda untuk menerapkan praktik yang sama yang Anda gunakan untuk mengembangkan perangkat lunak di infrastruktur Anda sehingga Anda dapat mengulang dengan cepat.
- Pipeline deployment: Gunakan alur integrasi berkelanjutan/deployment berkelanjutan (CI/CD) (misalnya, repositori kode sumber, sistem pembangunan, deployment, dan otomatisasi pengujian) untuk men-deploy infrastruktur Anda. Ini memungkinkan Anda untuk melakukan deployment dengan cara yang dapat diulang, konsisten, dan murah saat Anda melakukan pengulangan.
- Metrik yang terdefinisi dengan baik: Siapkan dan pantau metrik untuk menangkap indikator kinerja utama (KPI). Kami menyarankan Anda menggunakan metrik teknis dan metrik bisnis. Untuk situs web atau aplikasi seluler, metrik utama menangkap waktu ke bita pertama atau rendering. Metrik lain yang umumnya berlaku antara lain, hitungan thread, laju pengumpulan sampah, dan keadaan tunggu. Metrik bisnis, seperti biaya kumulatif agregat per permintaan, dapat memberikan peringatan kepada Anda tentang berbagai cara untuk menghemat biaya. Pertimbangkan dengan hati-hati bagaimana Anda akan menafsirkan metrik. Misalnya, Anda dapat memilih nilai maksimum atau persentil 99 dan bukannya nilai rata-rata.
- Pengujian kinerja otomatis: Sebagai bagian dari proses deployment Anda, mulai uji kinerja secara otomatis setelah pengujian yang berjalan lebih cepat berhasil dilewati. Otomatisasi harus menciptakan lingkungan baru, menyiapkan kondisi awal seperti data uji, kemudian jalankan serangkaian uji beban dan tolok ukur. Hasil dari pengujian-pengujian ini harus dikaitkan kembali dengan pembangunan sehingga Anda dapat melacak perubahan performa seiring waktu. Untuk pengujian yang lama, Anda dapat membuat ini sebagai bagian dari pipeline yang asinkron dari sisa pembangunan. Atau, Anda dapat menjalankan uji kinerja semalaman menggunakan Instans Spot Amazon EC2.
- Pembuatan beban: Anda harus membuat serangkaian skrip pengujian yang mereplikasi perjalanan pengguna sintetis atau yang dicatat sebelumnya. Skrip ini harus idempoten dan tidak dipasangkan, dan Anda mungkin harus menyertakan skrip prapemanasan untuk mendapatkan hasil yang valid. Sejauh dapat dilakukan, skrip pengujian Anda harus mereplikasi perilaku penggunaan dalam produksi. Anda dapat menggunakan solusi perangkat lunak sebagai layanan (SaaS) atau perangkat lunak untuk membuat beban. Pertimbangkan untuk menggunakan solusi AWS Marketplace dan Instans Spot— solusi ini dapat menjadi cara yang hemat biaya untuk membuat beban.

- Visibilitas kinerja: Metrik utama harus terlihat oleh tim Anda, terutama metrik terhadap setiap versi build. Tindakan ini memungkinkan Anda melihat setiap tren positif atau negatif yang signifikan seiring waktu. Anda juga harus menampilkan metrik atas jumlah kesalahan atau pengecualian untuk memastikan Anda menguji sistem yang berfungsi.
- Visualisasi: Gunakan teknik visualisasi yang memperjelas di mana masalah kinerja, hot spot, status tunggu, atau pemanfaatan rendah terjadi. Lapisi diagram arsitektur dengan metrik kinerja — kode atau grafik panggilan dapat membantu mengidentifikasi masalah dengan cepat.
- Proses peninjauan rutin: Arsitektur yang berkinerja buruk biasanya disebabkan oleh tidak adanya atau rusaknya proses peninjauan kinerja. Jika arsitektur Anda memiliki kinerja buruk, implementasi proses peninjauan kinerja memungkinkan Anda mendorong peningkatan berulang.
- Optimisasi berkelanjutan: Adopsi budaya untuk terus mengoptimalkan efisiensi kinerja beban kerja cloud Anda.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk efisiensi kinerja.

PERF 5: What process do you use to support more performance efficiency for your workload?

When architecting workloads, there are principles and practices that you can adopt to help you better run efficient high-performing cloud workloads. To adopt a culture that fosters performance efficiency of cloud workloads, consider these key principles and practices.

Sumber daya

Lihat referensi berikut untuk mempelajari selengkapnya tentang praktik terbaik kami untuk Efisiensi Kinerja.

Dokumentasi

- Amazon S3 Optimisasi Kinerja
- Amazon EBS Kinerja Volume

Laporan resmi

- Pilar Efisiensi Kinerja

Video:

- AWS re:Invent 2019: Landasan Amazon EC2 (CMP211-R2)
- AWS re:Invent 2019: Sesi kepemimpinan: Status penyatuan penyimpanan (STG201-L)
- AWS re:Invent 2019: Sesi kepemimpinan: Basis data yang dibuat khusus AWS (DAT209-L)
- AWS re:Invent 2019: Konektivitas ke AWS dan arsitektur jaringan AWS hibrid (NET317-R1)
- AWS re:Invent 2019: Memberdayakan Amazon EC2 generasi berikutnya: Mendalami sistem Nitro (CMP303-R2)
- AWS re:Invent 2019: Menaikkan skala ke 10 juta pengguna pertama Anda (ARC211-R)

Optimasi biaya

Pilar Optimasi Biaya mencakup kemampuan untuk menjalankan sistem guna menghadirkan nilai bisnis dengan harga yang paling rendah.

Pilar optimasi biaya menyediakan gambaran umum tentang prinsip, praktik terbaik, dan pertanyaan desain. Anda dapat menemukan panduan preskriptif tentang implementasi di laporan resmi Pilar Optimasi Biaya.

Topik

- Prinsip desain
- Definisi
- Praktik terbaik
- Sumber daya

Prinsip desain

Terdapat lima prinsip desain untuk optimasi biaya di cloud:

- Implementasikan Manajemen Keuangan Cloud: Untuk mencapai keberhasilan keuangan dan mempercepat realisasi nilai bisnis di cloud, Anda perlu berinvestasi dalam Manajemen Keuangan Cloud/Optimasi Biaya Organisasi Anda perlu mendedikasikan waktu dan sumber daya untuk membangun kemampuan dalam domain teknologi baru ini dan dalam manajemen penggunaan. Serupa dengan kemampuan Keamanan atau Keunggulan Operasi Anda, Anda perlu membangun

kemampuan melalui pembangunan pengetahuan, program, sumber daya, dan proses guna menjadi organisasi yang hemat biaya.

- Adopsi model pemakaian: Bayar hanya untuk sumber daya komputasi yang Anda perlukan dan tingkatkan atau turunkan penggunaan tergantung kebutuhan bisnis Anda, bukan menggunakan prakiraan rumit. Misalnya, lingkungan pengembangan dan pengujian umumnya hanya digunakan selama delapan jam sehari selama minggu operasional. Anda dapat menghentikan sumber daya ini ketika tidak digunakan untuk mendapatkan potensi penghematan biaya sebesar 75% (40 jam dibandingkan 168 jam).
- Ukur keseluruhan efisiensi: Ukur output bisnis beban kerja serta biaya terkait pengirimannya. Gunakan pengukuran ini untuk mengetahui keuntungan yang Anda dapatkan dari peningkatan output dan pengurangan biaya.
- Hentikan pembelanjaan untuk pekerjaan berat yang tidak terdiferensiasi: AWS melakukan pekerjaan berat operasi pusat data seperti pembuatan rak, penumpukan, dan pemberian daya pada server. AWS juga menyingkirkan beban operasional berupa pengelolaan sistem operasi dan aplikasi dengan layanan terkelola. Dengan demikian Anda dapat berkonsentrasi pada pelanggan dan proyek bisnis Anda dan mengalihkan fokus dari infrastruktur IT.
- Analisis dan pengeluaran atribut: Cloud memudahkan identifikasi biaya dan penggunaan sistem secara akurat, yang kemudian memungkinkan atribusi biaya IT yang transparan ke setiap pemilik beban kerja. Hal ini membantu mengukur laba atas investasi (ROI) dan memberi pemilik beban kerja sebuah peluang untuk mengoptimalkan sumber daya mereka dan memangkas biaya.

Definisi

Terdapat lima area praktik terbaik untuk optimasi biaya di cloud:

- Mempraktikkan Manajemen Keuangan Cloud
- Kesadaran akan penggunaan dan pengeluaran
- Sumber daya yang hemat biaya
- Kelola sumber daya pasokan dan permintaan
- Pengoptimalan dari waktu ke waktu

Seperti halnya pilar lain di dalam Well-Architected Framework, terdapat kompromi yang perlu dipertimbangkan, misalnya, pilihan untuk mengoptimalkan kecepatan masuk pasar atau biaya. Di beberapa kasus, pilihan terbaik adalah mengoptimalkan untuk kecepatan—memasuki pasar dengan cepat, mengirimkan fitur baru, atau hanya memenuhi tenggat—daripada berinvestasi untuk

optimasi biaya di awal. Keputusan desain terkadang disetir oleh sikap terburu-buru, bukan oleh data, dan selalu ada godaan untuk melakukan kompensasi berlebihan “untuk jaga-jaga”, alih-alih meluangkan waktu untuk membandingkan opsi-opsi deployment dengan biaya paling optimal. Hal ini dapat berakibat pada deployment dengan pengadaan yang berlebihan dan kurang optimal. Namun, ini adalah pilihan yang wajar ketika Anda perlu melakukan “angkat dan geser” sumber daya dari lingkungan on-premise ke cloud lalu melakukan optimasi setelahnya. Menginvestasikan energi yang cukup dalam strategi optimasi biaya di awal memungkinkan Anda untuk lebih mudah mewujudkan manfaat cloud pada aspek ekonomi dengan memastikan kepatuhan yang konsisten terhadap praktik terbaik dan menghindari pengadaan berlebihan yang tidak diperlukan: Bagian berikutnya menyediakan teknik dan praktik terbaik untuk implementasi Manajemen Keuangan Cloud serta optimasi biaya awal dan yang sedang berjalan untuk beban kerja Anda.

Praktik terbaik

Topik

- Mempraktikkan Manajemen Keuangan Cloud
- Kesadaran pengeluaran dan penggunaan
- Sumber daya yang hemat biaya
- Kelola sumber daya pasokan dan permintaan
- Pengoptimalan dari waktu ke waktu

Mempraktikkan Manajemen Keuangan Cloud

Dengan adopsi cloud, tim teknologi berinovasi lebih cepat dikarenakan siklus deployment infrastruktur, pengadaan, dan persetujuan yang lebih pendek. Pendekatan baru ke manajemen keuangan di cloud diperlukan untuk merealisasikan nilai bisnis dan keberhasilan keuangan. Pendekatan ini adalah Manajemen Keuangan Cloud, dan membangun kemampuan di organisasi Anda dengan mengimplementasikan pembangunan pengetahuan, program, sumber daya, dan proses di seluruh organisasi.

Banyak organisasi terdiri dari banyak unit yang berbeda dengan prioritas yang berbeda-beda. Kemampuan untuk menyelaraskan organisasi Anda dengan rangkaian tujuan keuangan yang telah disepakati, dan untuk membekali organisasi Anda dengan mekanisme untuk memenuhi tujuan tersebut, akan menciptakan organisasi yang lebih efisien. Organisasi yang mumpuni akan berinovasi dan membangun lebih cepat, menjadi lebih tangkas, dan selaras dengan faktor-faktor internal atau eksternal apa pun.

Di AWS, Anda dapat menggunakan Cost Explorer, dan Amazon Athena serta Amazon QuickSight dengan Laporan Biaya dan Penggunaan (CUR) yang bersifat opsional, untuk menyediakan kesadaran biaya dan penggunaan di seluruh organisasi Anda. AWS Budgets menyediakan notifikasi proaktif untuk biaya dan penggunaan. Blog AWS menyediakan informasi tentang layanan dan fitur baru untuk memastikan Anda tidak melewatkan perilisan layanan baru.

Pertanyaan berikut berfokus pada semua pertimbangan untuk optimasi biaya ini. (Untuk melihat daftar pertanyaan dan praktik terbaik optimasi biaya, buka Lampiran.).

BIAYA 1: Bagaimana cara mengimplementasikan manajemen keuangan cloud?

Dengan mengimplementasikan Manajemen Keuangan Cloud, organisasi dapat mewujudkan nilai bisnis dan keberhasilan finansial dengan mengoptimalkan biaya, penggunaan, dan skala di AWS.

Ketika membangun fungsi optimasi biaya, gunakan anggota dan lengkapi tim dengan ahli di bidang CFM dan optimasi biaya. Anggota tim yang ada akan memahami bagaimana fungsi organisasi saat ini dan cara mengimplementasikan perbaikan dengan cepat. Pertimbangkan juga untuk menyertakan orang-orang dengan set keterampilan tambahan atau khusus, seperti analitik dan manajemen proyek.

Ketika mengimplementasikan kesadaran biaya di organisasi Anda, tingkatkan atau kembangkan program dan proses yang sudah ada. Jauh lebih cepat melakukan penambahan ke yang sudah ada daripada membangun proses dan program baru. Dengan begitu, hasil akan dicapai dengan jauh lebih cepat.

Kesadaran pengeluaran dan penggunaan

Peningkatan fleksibilitas dan ketangkasan yang dihadirkan oleh cloud mendorong inovasi serta pengembangan dan deployment dengan laju cepat. Hal ini menyingkirkan proses manual serta waktu terkait pengadaan infrastruktur on-premise, termasuk identifikasi spesifikasi perangkat keras, negosiasi pengajuan harga, pengelolaan pesanan pembelian, penjadwalan pengiriman, lalu deployment sumber daya. Namun, kemudahan penggunaan dan kapasitas sesuai permintaan yang hampir tanpa batas ini memerlukan cara berpikir baru tentang pengeluaran.

Banyak bisnis terdiri dari beberapa sistem yang dijalankan oleh berbagai tim. Kemampuan untuk mengaitkan biaya sumber daya dengan tiap-tiap organisasi atau pemilik produk mendorong perilaku penggunaan yang efisien dan membantu mengurangi pemborosan. Pengaitan biaya yang

akurat memungkinkan Anda mengetahui produk mana yang benar-benar menguntungkan, dan memungkinkan Anda untuk mengambil keputusan yang lebih matang tentang target-target alokasi anggaran.

Di AWS, Anda membuat struktur akun dengan AWS Organizations atau AWS Control Tower, yang menyediakan pemisahan dan membantu dalam hal alokasi biaya dan penggunaan Anda. Anda juga dapat menggunakan tag sumber daya untuk menerapkan informasi bisnis dan organisasi ke penggunaan dan biaya Anda. Gunakan AWS Cost Explorer untuk mendapatkan visibilitas biaya dan penggunaan Anda, atau buat dasbor dan analitik khusus dengan Amazon Athena dan Amazon QuickSight. Kontrol atas biaya dan penggunaan Anda dilakukan dengan notifikasi melalui AWS Budgets, dan kontrol menggunakan AWS Identity and Access Management (IAM), dan Service Quotas.

Pertanyaan berikut berfokus pada semua pertimbangan untuk optimasi biaya ini.

BIAYA 2: Bagaimana cara mengatur penggunaan?

Tetapkan kebijakan dan mekanisme untuk memastikan bahwa biaya yang dikenakan sudah sesuai dan tujuan tercapai. Dengan menerapkan pendekatan periksa dan timbang (check and balances), Anda dapat berinovasi tanpa mengeluarkan dana yang berlebihan.

BIAYA 3: Bagaimana cara memantau penggunaan dan biaya?

Tetapkan kebijakan dan prosedur untuk memantau dan mengalokasikan biaya Anda dengan tepat. Ini memungkinkan Anda untuk mengukur dan meningkatkan efisiensi biaya beban kerja ini.

BIAYA 4: Bagaimana cara menonaktifkan sumber daya?

Implementasikan kontrol perubahan dan manajemen sumber daya dari awal proyek hingga akhir masa pakai. Ini memastikan Anda akan mematikan atau mengakhiri sumber daya yang tidak digunakan agar tidak boros.

Anda dapat menggunakan tag alokasi biaya untuk mengelompokkan dan melacak penggunaan dan biaya AWS Anda. Ketika menerapkan tag ke sumber daya AWS Anda (seperti instans EC2 atau bucket S3), AWS menghasilkan laporan biaya dan penggunaan dengan penggunaan dan tag Anda.

Anda dapat menerapkan tag yang mewakili kategori organisasi (seperti pusat biaya, nama beban kerja, atau pemilik) untuk mengatur biaya Anda di beberapa layanan.

Pastikan Anda menggunakan tingkat detail yang tepat pada pelaporan dan pemantauan biaya dan penggunaan. Untuk wawasan dan tren tingkat tinggi, gunakan tingkat detail harian dengan AWS Cost Explorer. Dengan analisis dan penyelidikan yang lebih mendalam, gunakan tingkat detail per jam di AWS Cost Explorer, atau Amazon Athena dan Amazon Quick dengan Laporan Biaya dan Penggunaan (CUR) pada tingkat detail per jam.

Menggabungkan sumber daya ber-tag dengan pelacakan siklus hidup entitas (karyawan, proyek) memungkinkan identifikasi sumber daya atau proyek yang menganggur yang sudah tidak menghasilkan nilai untuk organisasi dan harus dinonaktifkan. Anda dapat mengatur pemberitahuan penagihan untuk memberi tahu Anda tentang prediksi pengeluaran yang berlebihan.

Sumber daya yang hemat biaya

Penggunaan instans dan sumber daya yang tepat untuk beban kerja Anda merupakan hal utama dalam penghematan biaya. Misalnya, proses laporan mungkin memerlukan lima jam untuk berjalan di server yang lebih kecil tetapi satu jam untuk berjalan di server yang lebih besar dengan harga lebih mahal dua kali lipat. Kedua server tersebut memberi Anda hasil yang sama, tetapi seiring waktu, server yang lebih kecil akan memakan biaya lebih besar.

Beban kerja yang dirancang dengan baik menggunakan sumber daya yang paling hemat biaya, yang dapat memberikan dampak ekonomi positif yang besar. Anda juga memiliki kesempatan untuk menggunakan layanan terkelola untuk memangkas biaya. Misalnya, alih-alih memelihara server untuk mengirimkan email, Anda dapat menggunakan layanan yang mengenakan biaya per pesan.

AWS menawarkan beragam opsi harga yang fleksibel dan paling hemat untuk mendapatkan instans dari Amazon EC2 dan layanan lain sesuai kebutuhan Anda. Instans Sesuai Permintaan memungkinkan Anda membayar kapasitas komputasi berdasarkan jam, tanpa memerlukan komitmen minimum. Savings Plans dan Instans Terpesan menawarkan penghematan hingga 75% dari harga Sesuai Permintaan. Dengan Instans Spot, Anda dapat memanfaatkan kapasitas Amazon EC2 yang tidak terpakai dan menawarkan penghematan hingga 90% dari harga Sesuai Permintaan. Instans Spot tepat ketika sistem dapat mentoleransi penggunaan armada server di mana tiap-tiap server dapat bergerak secara dinamis. seperti server web stateless, pemrosesan batch, atau saat menggunakan HPC dan big data.

Pilihan layanan yang tepat juga dapat mengurangi penggunaan dan biaya; seperti CloudFront untuk meminimalkan transfer data, atau menyingkirkan biaya sepenuhnya, seperti memanfaatkan Amazon Aurora di RDS untuk menghilangkan biaya lisensi basis data yang mahal.

Pertanyaan berikut berfokus pada semua pertimbangan untuk optimasi biaya ini.

BIAYA5: Bagaimana cara mengevaluasi biaya ketika Anda memilih layanan?

Amazon EC2, Amazon EBS, dan Amazon S3 adalah layanan blok penyusun AWS. Layanan terkelola, seperti Amazon RDS dan Amazon DynamoDB, adalah layanan AWS dengan tingkat lebih tinggi, atau tingkat aplikasi. Dengan memilih blok penyusun dan layanan terkelola yang sesuai, Anda dapat mengoptimalkan biaya beban kerja ini. Contohnya, dengan menggunak an layanan terkelola, Anda dapat mengurangi atau menghilangkan sebagian besar dari biaya tambahan untuk administrasi dan operasi, sehingga Anda bebas mengerjakan aplikasi dan aktivitas terkait bisnis.

BIAYA 6: Bagaimana cara memenuhi target biaya ketika Anda memilih jenis, ukuran, dan jumlah sumber daya?

Pastikan Anda memilih jumlah sumber daya dan ukuran sumber daya yang sesuai untuk tugas yang ada. Anda meminimalkan pemborosan dengan memilih jenis, ukuran, dan jumlah yang paling hemat.

BIAYA 7: Bagaimana cara menggunakan model harga untuk mengurangi biaya?

Gunakan model harga yang paling sesuai untuk sumber daya Anda untuk meminimalkan pengeluaran.

BIAYA 8: Bagaimana cara merencanakan biaya transfer data?

Pastikan Anda merencanakan dan memantau biaya transfer daya sehingga Anda dapat mengambil keputusan arsitektur untuk meminimalkan biaya. Perubahan arsitektur yang kecil namun efektif dapat secara drastis mengurangi biaya operasional Anda seiring waktu.

Dengan mempertimbangkan biaya selama pemilihan layanan, dan menggunakan alat-alat seperti Cost Explorer dan AWS Trusted Advisor untuk meninjau secara rutin penggunaan AWS Anda, Anda dapat secara aktif memantau pemanfaatan dan menyesuaikan deployment dengan semestinya.

Kelola sumber daya pasokan dan permintaan

Ketika Anda beralih ke cloud, Anda hanya perlu membayar sesuai dengan yang Anda butuhkan. Anda dapat memasok sumber daya sesuai dengan permintaan beban kerja pada saat dibutuhkan, sehingga menghilangkan pemborosan biaya dan penyediaan berlebih yang tidak terpakai. Anda juga dapat memodifikasi permintaan, menggunakan throttle, buffer, atau antrean untuk melancarkan permintaan dan melayaninya dengan sumber daya yang lebih sedikit sehingga biayanya juga menjadi lebih rendah, atau memprosesnya di lain waktu dengan layanan batch.

Di AWS, Anda dapat menyediakan sumber daya secara otomatis untuk disesuaikan dengan permintaan beban kerja. Penskalaan Otomatis yang menggunakan pendekatan berbasis permintaan atau waktu memungkinkan Anda untuk menambahkan dan menghapus sumber daya seperlunya. Jika Anda dapat mengantisipasi perubahan sesuai permintaan, Anda dapat menghemat lebih banyak dana dan memastikan sumber daya Anda sesuai dengan kebutuhan beban kerja. Anda dapat menggunakan Amazon API Gateway untuk mengimplementasikan throttling, atau Amazon SQS untuk mengimplementasikan antrean di beban kerja Anda. Keduanya akan membantu Anda memodifikasi permintaan pada komponen beban kerja Anda.

Pertanyaan berikut berfokus pada semua pertimbangan untuk optimasi biaya ini.

BIAYA 9: Bagaimana cara mengelola sumber daya pasokan dan permintaan?

Untuk beban kerja yang memiliki pengeluaran dan kinerja seimbang, pastikan semua yang Anda bayar benar-benar digunakan dan hindari tingkat penggunaan instans yang jauh terlalu rendah. Metrik penggunaan yang melenceng ke salah satu arah memiliki dampak buruk pada organisasi Anda, baik dalam hal biaya operasional (kinerja yang menurun akibat penggunaan yang berlebiha n), atau pemborosan pengeluaran AWS (akibat pengadaan yang berlebihan).

Ketika merancang untuk memodifikasi sumber daya pasokan dan permintaan, pikirkan secara aktif tentang pola-pola penggunaan, waktu yang diperlukan untuk menyediakan sumber daya baru, dan prediktabilitas pola permintaan. Ketika mengelola permintaan, pastikan Anda memiliki antrean atau buffer dengan ukuran yang tepat, dan Anda merespons permintaan beban kerja dalam waktu yang diperlukan.

Pengoptimalan dari waktu ke waktu

Saat AWS merilis fitur dan layanan baru, praktik yang terbaik adalah meninjau keputusan arsitektur yang ada untuk memastikan keputusan Anda tetap yang paling hemat. Seiring dengan perubahan

persyaratan, jangan ragu untuk menonaktifkan sumber daya, seluruh layanan, dan sistem yang sudah tidak diperlukan.

Implementasi fitur atau tipe sumber daya baru dapat mengoptimalkan beban kerja Anda secara bertahap, sambil meminimalkan upaya yang diperlukan untuk mengimplementasikan perubahan. Ini menyediakan peningkatan berkelanjutan dalam hal efisiensi seiring waktu dan memastikan Anda tetap berada di teknologi paling mutakhir untuk memangkas biaya operasi. Anda juga dapat mengganti atau menambahkan komponen baru ke beban kerja dengan layanan baru. Hal ini dapat menyediakan peningkatan yang signifikan dalam hal efisiensi, sehingga penting untuk secara rutin meninjau beban kerja, dan mengimplementasikan layanan dan fitur baru.

Pertanyaan berikut berfokus pada semua pertimbangan untuk optimasi biaya ini.

BIAYA 10: Bagaimana cara mengevaluasi layanan baru?

Saat AWS merilis fitur dan layanan baru, praktik yang terbaik adalah meninjau keputusan arsitektur yang ada untuk memastikan keputusan Anda tetap yang paling hemat.

Ketika meninjau deployment secara rutin, nilailah bagaimana layanan yang lebih baru dapat membantu menghemat dana Anda. Misalnya, Amazon Aurora di RDS dapat mengurangi biaya untuk basis data relasional. Penggunaan nirserver seperti Lambda dapat menghilangkan kebutuhan untuk mengoperasikan dan mengelola instans untuk menjalankan kode.

Sumber daya

Lihat sumber daya berikut untuk mempelajari selengkapnya tentang praktik terbaik kami untuk Optimasi Biaya.

Dokumentasi

- Dokumentasi AWS

Laporan Resmi

- Pilar Optimasi Biaya

Pelestarian lingkungan

Pilar Pelestarian Lingkungan berfokus pada dampak lingkungan, terutama konsumsi dan efisiensi energi, karena ini merupakan pendorong penting bagi arsitek untuk melandasi tindakan langsung untuk mengurangi penggunaan sumber daya. Anda dapat menemukan panduan preskriptif tentang implementasi di Laporan resmi Pilar Pelestarian Lingkungan.

Topik

- Prinsip desain
- Definisi
- Praktik terbaik

Prinsip desain

Terdapat enam prinsip desain untuk pelestarian lingkungan di cloud:

- Pahami dampak Anda: Ukur dampak beban kerja cloud Anda dan buat model dampak beban kerja Anda untuk masa mendatang. Sertakan semua sumber dampak, termasuk dampak akibat penggunaan produk Anda oleh pelanggan, serta dampak yang muncul dari penonaktifan dan penghentian produk. Bandingkan output produktif dengan total dampak beban kerja cloud Anda dengan meninjau sumber daya dan emisi yang diperlukan per unit kerja. Gunakan data ini untuk membuat indikator kinerja utama (KPI), evaluasi cara-cara untuk meningkatkan produktivitas sambil mengurangi dampak, dan perkirakan dampak perubahan yang diajukan seiring waktu.
- Tetapkan tujuan pelestarian lingkungan: Untuk masing-masing beban kerja cloud, tetapkan tujuan pelestarian lingkungan jangka panjang seperti mengurangi sumber daya komputasi dan penyimpanan yang diperlukan per transaksi. Modelkan laba atas investasi peningkatan pelestarian lingkungan untuk beban kerja yang ada, dan beri pemilik sumber daya yang mereka perlukan untuk berinvestasi dalam tujuan pelestarian lingkungan. Rencanakan pertumbuhan, dan rancang beban kerja Anda agar pertumbuhan menghasilkan penurunan intensitas dampak yang terukur berdasarkan unit yang tepat, seperti per pengguna atau per transaksi. Tujuan ini membantu Anda mendukung tujuan pelestarian lingkungan yang lebih luas untuk bisnis atau organisasi Anda, mengidentifikasi regresi, dan memprioritaskan area-area peningkatan potensial.
- Memaksimalkan pemanfaatan: Sesuaikan ukuran beban kerja dan implementasikan desain yang efisien untuk memastikan pemanfaatan yang tinggi dan memaksimalkan efisiensi energi untuk perangkat keras yang mendasari. Dua host yang berjalan dengan pemanfaatan 30% memiliki efisiensi yang lebih rendah daripada satu host dengan pemanfaatan 60% dikarenakan

konsumsi daya dasar per host. Pada saat yang sama, singkirkan atau minimalkan sumber daya, pemrosesan, dan penyimpanan yang tidak aktif untuk mengurangi total energi yang diperlukan untuk menjalankan beban kerja Anda.

- Antisipasi dan adopsi penawaran perangkat keras dan perangkat lunak baru yang lebih efisien: Dukung peningkatan hulu yang dibuat oleh partner dan pemasok Anda untuk membantu Anda mengurangi dampak beban kerja cloud Anda. Terus pantau dan evaluasi penawaran perangkat keras dan perangkat lunak baru yang lebih efisien. Rancang fleksibilitas untuk memungkinkan pengadopsian teknologi efisien baru secara cepat.
- Gunakan layanan terkelola: Berbagi layanan di seluruh basis pelanggan yang luas dapat membantu memaksimalkan pemanfaatan sumber daya, yang mengurangi jumlah infrastruktur yang diperlukan untuk mendukung beban kerja cloud. Sebagai contoh, pelanggan dapat berbagi dampak komponen pusat data yang sama seperti daya dan jaringan dengan memigrasikan beban kerja ke AWS Cloud dan mengadopsi layanan terkelola, seperti AWS Fargate untuk kontainer nirserver, tempat AWS beroperasi pada skala besar dan bertanggung jawab atas operasi efisien mereka. Gunakan layanan terkelola yang dapat membantu meminimalkan dampak Anda, seperti memindahkan data yang jarang diakses ke penyimpanan dingin secara otomatis dengan konfigurasi Amazon S3 Lifecycle atau Amazon EC2 Auto Scaling untuk menyesuaikan kapasitas guna memenuhi permintaan.
- Kurangi dampak hilir beban kerja cloud Anda: Kurangi jumlah energi atau sumber daya yang diperlukan untuk menggunakan layanan Anda. Kurangi atau singkirkan kebutuhan pelanggan untuk meningkatkan perangkat mereka untuk menggunakan layanan Anda. Uji menggunakan device farm untuk memahami dampak yang diperkirakan dan uji dengan pelanggan untuk memahami dampak riil dari penggunaan layanan Anda.

Definisi

Terdapat enam area praktik terbaik untuk pelestarian lingkungan di cloud:

- Pemilihan wilayah
- Pola perilaku pengguna
- Pola arsitektur dan perangkat lunak
- Pola data
- Pola perangkat keras
- Proses deployment dan pengembangan

Pelestarian lingkungan di cloud adalah sebuah upaya berkelanjutan yang difokuskan terutama pada pengurangan dan efisiensi energi di semua komponen beban kerja dengan mencapai manfaat maksimum dari sumber daya yang disediakan dan meminimalkan total sumber daya yang diperlukan. Upaya ini bisa mencakup pemilihan bahasa pemrograman yang efisien di awal, adopsi algoritme modern, penggunaan teknik penyimpanan data yang efisien, deployment ke infrastruktur komputasi yang efisien dan terukur dengan tepat, serta meminimalkan kebutuhan perangkat keras pengguna akhir berdaya tinggi.

Praktik terbaik

Topik

- Pemilihan wilayah
- Pola perilaku pengguna
- Pola arsitektur dan perangkat lunak
- Pola data
- Pola perangkat keras
- Pola pengembangan dan deployment
- Sumber daya

Pemilihan wilayah

Pilih Wilayah di mana Anda akan mengimplementasikan beban kerja Anda berdasarkan persyaratan bisnis dan tujuan pelestarian lingkungan Anda.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk pelestarian lingkungan ini. (Untuk melihat daftar pertanyaan dan praktik terbaik pelestarian lingkungan, buka Lampiran.)

SUS 1: Bagaimana cara memilih Wilayah untuk mendukung tujuan pelestarian lingkungan Anda?

Pilih Wilayah di dekat proyek-proyek energi terbarukan Amazon dan Wilayah dengan jaringan energi yang memiliki intensitas karbon terpublikasi yang lebih rendah daripada lokasi (atau Wilayah) lain.

Pola perilaku pengguna

Cara pengguna mengonsumsi beban kerja Anda dan sumber daya lainnya dapat membantu Anda mengidentifikasi peningkatan untuk memenuhi tujuan pelestarian lingkungan. Skalakan infrastruktur agar dapat terus sesuai dengan beban pengguna dan pastikan hanya melakukan deployment sumber daya minimum yang diperlukan untuk mendukung pengguna. Selaraskan tingkat layanan dengan kebutuhan pelanggan. Posisikan sumber daya untuk membatasi jaringan yang diperlukan pengguna untuk mengonsumsinya. Singkirkan aset yang ada tetapi tidak digunakan. Identifikasi aset yang telah dibuat dan tidak digunakan, dan berhenti membuat aset tersebut. Bekali anggota tim Anda dengan perangkat yang mendukung kebutuhan mereka dengan dampak minim terhadap pelestarian lingkungan.

Pertanyaan berikut ini berfokus pada pertimbangan untuk pelestarian lingkungan ini:

SUS 2: Bagaimana cara memanfaatkan pola perilaku pengguna untuk mendukung tujuan pelestari an lingkungan Anda?

Cara pengguna mengonsumsi beban kerja Anda dan sumber daya lainnya dapat membantu Anda mengidentifikasi peningkatan untuk memenuhi tujuan pelestarian lingkungan. Skalakan infrastru ktur agar dapat terus sesuai dengan beban pengguna dan pastikan hanya melakukan deploymen t sumber daya minimum yang diperlukan untuk mendukung pengguna. Selaraskan tingkat layanan dengan kebutuhan pelanggan. Posisikan sumber daya untuk membatasi jaringan yang diperluka n pengguna untuk mengonsumsinya. Singkirkan aset yang ada tetapi tidak digunakan. Identifikasi aset yang telah dibuat dan tidak digunakan, dan berhenti membuat aset tersebut. Bekali anggota tim Anda dengan perangkat yang mendukung kebutuhan mereka dengan dampak minim terhadap pelestarian lingkungan.

Skalakan infrastruktur dengan beban pengguna: Identifikasi periode pemanfaatan rendah atau nol dan skalakan sumber daya untuk menyingkirkan kapasitas berlebih dan meningkatkan efisiensi.

Selaraskan SLA dengan tujuan pelestarian lingkungan: Tetapkan dan perbarui kesepakatan tingkat layanan (SLA) seperti ketersediaan periode retensi data untuk meminimalkan jumlah sumber daya yang diperlukan untuk mendukung beban kerja Anda sambil terus memenuhi persyaratan bisnis.

Singkirkan pembuatan dan pemeliharaan aset tak terpakai: Analisis aset aplikasi (seperti laporan pra-kompilasi, set data, dan gambar statis) serta pola akses aset untuk mengidentifikasi redundansi, pemanfaatan yang terlalu rendah, dan potensi target penonaktifan. Gabungkan aset-aset yang

dihasilkan dengan konten yang redundan (misalnya laporan bulanan dengan set data dan output yang bertumpuk atau sama) untuk menyingkirkan sumber daya yang dipakai ketika menggandakan output. Nonaktifkan aset tidak terpakai (misalnya gambar-gambar produk yang sudah tidak dijual) untuk menghemat sumber daya yang dikonsumsi dan mengurangi jumlah sumber daya yang digunakan untuk mendukung beban kerja.

Optimalkan penempatan beban kerja secara geografis untuk lokasi pengguna: Analisis pola akses jaringan untuk mengidentifikasi lokasi geografis tempat pelanggan Anda terhubung. Pilih Wilayah dan layanan yang mengurangi jarak yang harus ditempuh oleh lalu lintas jaringan guna menurunkan total sumber daya jaringan yang diperlukan untuk mendukung beban kerja Anda.

Optimalkan sumber daya anggota tim untuk aktivitas yang dijalankan: Optimalkan sumber daya yang disediakan untuk anggota tim untuk meminimalkan dampak pelestarian lingkungan sambil mendukung kebutuhan mereka. Sebagai contoh, lakukan operasi yang kompleks, seperti rendering dan kompilasi, di desktop cloud bersama dengan tingkat pemanfaatan yang tinggi, bukan di sistem pengguna tunggal berdaya tinggi tetapi dengan pemanfaatan yang rendah.

Pola arsitektur dan perangkat lunak

Implementasikan pola untuk melancarkan beban dan mempertahankan penggunaan sumber daya yang diterapkan secara sangat konsisten untuk meminimalkan sumber daya yang dikonsumsi. Komponen mungkin akan menjadi tidak aktif dari kurangnya penggunaan dikarenakan perubahan perilaku pengguna seiring waktu. Revisi pola dan arsitektur untuk menggabungkan komponen dengan pemanfaatan rendah guna meningkatkan pemanfaatan secara keseluruhan. Pensiunkan komponen yang tidak lagi diperlukan. Pahami kinerja komponen beban kerja Anda, dan optimalkan komponen yang mengonsumsi sumber daya terbanyak. Ketahui perangkat yang digunakan pelanggan untuk mengakses layanan Anda, dan implementasikan pola untuk meminimalkan kebutuhan pemutakhiran perangkat.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk pelestarian lingkungan ini:

SUS 3: Bagaimana cara memanfaatkan pola arsitektur dan perangkat lunak untuk mendukung tujuan pelestarian lingkungan Anda?

Implementasikan pola untuk melancarkan beban dan mempertahankan penggunaan sumber daya yang diterapkan secara sangat konsisten untuk meminimalkan sumber daya yang dikonsumsi. Komponen mungkin akan menjadi tidak aktif dari kurangnya penggunaan dikarenakan perubahan perilaku pengguna seiring waktu. Revisi pola dan arsitektur untuk menggabungkan komponen

SUS 3: Bagaimana cara memanfaatkan pola arsitektur dan perangkat lunak untuk mendukung tujuan pelestarian lingkungan Anda?

dengan pemanfaatan rendah guna meningkatkan pemanfaatan secara keseluruhan. Pensiunkan komponen yang tidak lagi diperlukan. Pahami kinerja komponen beban kerja Anda, dan optimalkan komponen yang mengonsumsi sumber daya terbanyak. Ketahui perangkat yang digunakan pelanggan untuk mengakses layanan Anda, dan implementasikan pola untuk meminimalkan kebutuhan pemutakhiran perangkat.

Optimalkan perangkat lunak dan arsitektur untuk tugas-tugas asinkron dan terjadwal: Gunakan desain dan arsitektur perangkat lunak yang efisien untuk meminimalkan rata-rata sumber daya yang diperlukan per unit kerja. Implementasikan mekanisme yang menghasilkan pemanfaatan komponen yang merata untuk mengurangi sumber daya tidak aktif antartugas dan meminimalkan dampak lonjakan beban.

Singkirkan atau faktor ulang komponen beban kerja dengan penggunaan rendah atau nol: Pantau aktivitas beban kerja untuk mengidentifikasi perubahan dalam hal pemanfaatan setiap komponen seiring waktu. Singkirkan komponen yang tidak digunakan dan sudah tidak diperlukan, dan faktor ulang komponen dengan sedikit pemanfaatan, untuk membatasi sumber daya yang terbuang.

Optimalkan area-area kode yang memakai waktu atau sumber daya paling banyak: Pantau aktivitas beban kerja untuk mengidentifikasi komponen aplikasi yang memakai sumber daya paling banyak. Optimalkan kode yang berjalan di dalam komponen-komponen tersebut untuk meminimalkan penggunaan sumber daya sambil memaksimalkan kinerja.

Optimalkan dampak terhadap perangkat dan perlengkapan pelanggan: Pahami perangkat dan perlengkapan yang digunakan pelanggan untuk menggunakan layanan Anda, siklus hidup yang diharapkan, serta dampak penggantian komponen tersebut terhadap keuangan dan pelestarian lingkungan. Implementasikan pola dan arsitektur perangkat lunak guna meminimalkan kebutuhan pelanggan untuk mengganti perangkat dan memutakhirkan perlengkapan. Misalnya, implementasikan fitur baru menggunakan kode yang kompatibel dengan versi perangkat keras dan sistem operasi yang lebih lama, atau kelola ukuran payload agar tidak melebihi kapasitas penyimpanan perangkat target.

Gunakan pola dan arsitektur perangkat lunak yang paling mendukung pola akses dan penyimpanan data: Pahami bagaimana data digunakan di dalam beban kerja Anda, dipakai oleh pengguna Anda, ditransfer, dan disimpan. Seleksi teknologi untuk meminimalkan persyaratan pemrosesan data dan penyimpanan data.

Pola data

Implementasikan pola untuk melancarkan beban dan mempertahankan penggunaan sumber daya yang diterapkan secara sangat konsisten untuk meminimalkan sumber daya yang dikonsumsi. Komponen mungkin akan menjadi tidak aktif dari kurangnya penggunaan dikarenakan perubahan perilaku pengguna seiring waktu. Revisi pola dan arsitektur untuk menggabungkan komponen dengan pemanfaatan rendah guna meningkatkan pemanfaatan secara keseluruhan. Pensiunkan komponen yang tidak lagi diperlukan. Pahami kinerja komponen beban kerja Anda, dan optimalkan komponen yang mengonsumsi sumber daya terbanyak. Ketahui perangkat yang digunakan pelanggan untuk mengakses layanan Anda, dan implementasikan pola untuk meminimalkan kebutuhan pemutakhiran perangkat.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk pelestarian lingkungan ini:

SUS 4: Bagaimana cara memanfaatkan pola penggunaan dan akses data untuk mendukung tujuan pelestarian lingkungan Anda?

Implementasikan praktik manajemen data untuk mengurangi penyimpanan yang diberikan, yang diperlukan untuk mendukung beban kerja Anda, dan sumber daya yang diperlukan untuk menggunakannya. Pahami data Anda, dan gunakan konfigurasi dan teknologi penyimpanan penggunaan yang paling tepat untuk mendukung nilai bisnis data dan cara data digunakan. Buat siklus hidup data di penyimpanan yang lebih efisien dan berkinerja lebih sedikit ketika persyaratan menurun, dan hapus data yang tidak lagi diperlukan.

Implementasikan kebijakan klasifikasi data: Klasifikasikan data untuk memahami pentingnya setiap data bagi hasil bisnis. Gunakan informasi ini untuk menentukan kapan Anda dapat memindahkan data ke penyimpanan yang lebih hemat energi atau menghapusnya secara aman.

Gunakan teknologi yang mendukung pola akses dan penyimpanan data: Gunakan penyimpanan yang paling mendukung cara data Anda diakses dan disimpan untuk meminimalkan sumber daya yang disediakan sambil mendukung beban kerja Anda. Misalnya, perangkat solid state (SSD) memerlukan energi yang lebih besar daripada drive magnetik dan sebaiknya hanya digunakan untuk kasus penggunaan data aktif. Gunakan penyimpanan kelas arsip yang hemat energi untuk data yang jarang diakses.

Gunakan kebijakan siklus hidup untuk menghapus data yang tidak diperlukan: Kelola siklus hidup semua data Anda dan terapkan lini waktu penghapusan secara otomatis untuk meminimalkan total kebutuhan penyimpanan beban kerja Anda.

Minimalkan pengadaan yang berlebihan dalam penyimpanan blok: Untuk meminimalkan total penyimpanan yang disediakan, buat penyimpanan blok dengan alokasi ukuran yang tepat untuk beban kerja. Gunakan volume elastis untuk memperluas penyimpanan seiring pertumbuhan data tanpa harus mengubah ukuran penyimpanan yang dilampirkan ke sumber daya komputasi. Secara rutin tinjau volume elastis dan kecilkan volume dengan penyediaan berlebih agar sesuai dengan ukuran data saat ini.

Singkirkan data yang tidak diperlukan atau redundan: Gandakan data hanya saat diperlukan untuk meminimalkan total penyimpanan yang digunakan. Gunakan teknologi pencadangan yang menghilangkan data ganda pada tingkat file dan blok. Batasi penggunaan konfigurasi Redundant Array of Independent Drives (RAID) kecuali diperlukan untuk memenuhi SLA.

Gunakan sistem file dan penyimpanan objek bersama untuk mengakses data umum: Adopsi penyimpanan bersama dan sumber kebenaran tunggal untuk menghindari data ganda dan mengurangi kebutuhan penyimpanan total untuk beban kerja Anda. Ambil data dari penyimpanan bersama hanya saat diperlukan. Lepaskan volume yang tidak digunakan untuk menghemat sumber daya. Minimalkan perpindahan data di jaringan: Gunakan penyimpanan bersama dan akses data dari penyimpanan data wilayah untuk meminimalkan total sumber daya jaringan yang diperlukan untuk mendukung perpindahan data untuk beban kerja Anda.

Cadangkan data hanya saat data sulit dibuat ulang: Untuk meminimalkan penggunaan penyimpanan, hanya cadangkan data yang memiliki nilai bisnis atau diperlukan untuk memenuhi persyaratan kepatuhan. Periksa kebijakan pencadangan dan jangan masukkan penyimpanan sementara yang tidak memberikan nilai dalam skenario pemulihan.

Pola perangkat keras

Cari peluang untuk mengurangi dampak beban kerja terhadap pelestarian lingkungan dengan membuat perubahan pada praktik manajemen perangkat keras Anda. Minimalkan jumlah perangkat keras yang diperlukan untuk diadakan dan diterapkan, serta pilih perangkat keras yang paling efisien untuk setiap beban kerja Anda.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk pelestarian lingkungan ini:

SUS 5: Bagaimana cara praktik penggunaan dan manajemen perangkat keras mendukung tujuan pelestarian lingkungan Anda?

Cari peluang untuk mengurangi dampak beban kerja terhadap pelestarian lingkungan dengan membuat perubahan pada praktik manajemen perangkat keras Anda. Minimalkan jumlah

SUS 5: Bagaimana cara praktik penggunaan dan manajemen perangkat keras mendukung tujuan pelestarian lingkungan Anda?

perangkat keras yang diperlukan untuk diadakan dan diterapkan, serta pilih perangkat keras yang paling efisien untuk setiap beban kerja Anda.

Gunakan perangkat keras dalam jumlah minim untuk memenuhi kebutuhan Anda: Menggunakan kemampuan cloud, Anda dapat membuat perubahan secara sering pada implementasi beban kerja Anda. Perbarui komponen yang diterapkan seiring perubahan kebutuhan Anda.

Gunakan tipe instans dengan dampak paling sedikit: Terus pantau perilisan tipe-tipe instans baru dan manfaatkan peningkatan efisiensi energi, termasuk tipe instans yang dirancang untuk mendukung beban kerja khusus seperti pelatihan dan inferensi machine learning, dan transkoding video.

Gunakan layanan terkelola: Dengan layanan terkelola, tanggung jawab untuk menjaga pemanfaatan rata-rata tetap tinggi, serta optimalisasi pelestarian lingkungan untuk perangkat keras yang diterapkan, dialihkan kepada AWS. Gunakan layanan terkelola untuk meratakan dampak layanan terhadap pelestarian lingkungan ke semua tenant layanan, sehingga kontribusi individu Anda dapat berkurang.

Optimalkan penggunaan GPU Anda: Unit pemrosesan grafis (GPU) bisa menjadi sumber konsumsi daya yang tinggi, dan beban kerja GPU sangat beragam, seperti rendering, transkoding, dan pelatihan serta pemodelan machine learning. Jalankan instans GPU hanya ketika diperlukan, dan nonaktifkan instans GPU secara otomatis saat tidak diperlukan, guna meminimalkan sumber daya yang digunakan.

Pola pengembangan dan deployment

Cari peluang untuk mengurangi dampak pelestarian lingkungan Anda dengan membuat perubahan pada praktik deployment, pengujian, dan pengembangan Anda.

Pertanyaan berikut ini berfokus pada semua pertimbangan untuk pelestarian lingkungan ini:

SUS 6: Bagaimana cara proses deployment dan pengembangan mendukung tujuan pelestarian lingkungan?

Cari peluang untuk mengurangi dampak pelestarian lingkungan Anda dengan membuat perubahan pada praktik deployment, pengujian, dan pengembangan Anda.

Adopsi metode yang dapat memulai peningkatan pelestarian lingkungan dengan cepat: Uji dan validasi potensi peningkatan sebelum diterapkan ke produksi. Pertimbangkan biaya pengujian saat menghitung potensi manfaat sebuah peningkatan untuk masa depan. Kembangkan metode pengujian berbiaya rendah untuk memungkinkan penerapan peningkatan-peningkatan kecil.

Jaga kemutakhiran beban kerja Anda: Sistem operasi, pustaka, dan aplikasi yang mutakhir dapat meningkatkan efisiensi beban kerja dan memungkinkan pengadopsian teknologi yang lebih efisien dengan lebih mudah. Perangkat lunak yang mutakhir juga dapat menyertakan fitur-fitur untuk mengukur dampak beban kerja terhadap pelestarian lingkungan secara lebih akurat, mengingat vendor juga menghadirkan fitur-fitur untuk memenuhi tujuan pelestarian lingkungan mereka sendiri.

Tingkatkan pemanfaatan lingkungan build: Gunakan otomatisasi dan infrastruktur sebagai kode untuk mengaktifkan lingkungan pra-produksi saat diperlukan dan menonaktifkannya saat tidak digunakan. Hal yang umum dilakukan adalah menjadwalkan periode ketersediaan yang bertepatan dengan jam kerja anggota tim pengembangan. Hibernasi adalah alat yang berguna untuk mempertahankan status dan mengaktifkan instans dengan cepat hanya pada saat dibutuhkan. Gunakan jenis instans dengan kapasitas lonjakan, instans spot, layanan basis data elastis, kontainer, dan teknologi lainnya untuk menyesuaikan pengembangan dan menguji kapasitas dengan penggunaan.

Gunakan device farm terkelola untuk pengujian: Device farm yang terkelola meratakan dampak manufaktur perangkat keras dan penggunaan sumber daya terhadap pelestarian lingkungan ke beberapa tenant. Device farm terkelola menawarkan tipe perangkat yang beragam sehingga Anda dapat mendukung perangkat keras yang lebih lama dan kurang populer, serta menghindari dampak pelestarian lingkungan pelanggan akibat pemutakhiran perangkat yang tidak perlu.

Sumber daya

Buka sumber daya berikut untuk mempelajari selengkapnya tentang praktik terbaik kami untuk pelestarian lingkungan.

Laporan resmi

- Pilar Pelestarian Lingkungan

Video

- The Climate Pledge (Sumpah Iklim)

Proses peninjauan

Peninjauan arsitektur harus dilakukan dengan cara yang konsisten, tanpa menyalahkan pihak mana pun untuk mendorong pengamatan yang mendalam. Ini harus menjadi proses yang ringan (hitungan jam, bukan hari) dengan berdiskusi, bukan proses audit. Tujuan dari peninjauan arsitektur adalah untuk mengidentifikasi masalah yang sangat penting dan perlu ditangani atau area yang dapat ditingkatkan. Hasil dari peninjauan adalah serangkaian tindakan yang dapat meningkatkan pengalaman pelanggan dalam menggunakan beban kerja.

Seperti yang telah dibahas di bagian “Tentang Arsitektur”, Anda ingin semua tim bertanggung jawab atas kualitas dari setiap arsitekturnya. Untuk anggota tim yang membangun arsitektur menggunakan Kerangka Kerja Well-Architected, sebaiknya tinjau arsitektur secara berkelanjutan, tanpa perlu terpaku pada pertemuan peninjauan yang formal. Pendekatan yang berkelanjutan memungkinkan anggota tim Anda untuk memperbarui jawaban seiring dengan berkembangnya arsitektur, serta meningkatkan arsitektur saat Anda memberikan fitur.

Kerangka Kerja AWS Well-Architected telah diselaraskan dengan cara AWS meninjau sistem dan layanan secara internal. Ini dilandaskan pada serangkaian prinsip desain yang memengaruhi arsitektur, serta pertanyaan-pertanyaan untuk memastikan bahwa orang-orang tidak mengabaikan area yang mengutamakan Analisis Akar Masalah (RCA). Kapan pun terjadi masalah yang signifikan terhadap sistem internal, layanan AWS, atau pelanggan, kami merujuk ke RCA untuk menentukan apakah kami perlu meningkatkan proses peninjauan yang digunakan.

Peninjauan harus diterapkan pada pencapaian utama dalam siklus hidup produk, sedini mungkin dalam tahap desain untuk menghindari keputusan kaku yang sulit untuk diubah, sebelum tanggal mulai pengaktifan. (Banyak keputusan yang dapat diubah dan memiliki alternatif. Keputusan tersebut dapat menggunakan proses yang ringan. Keputusan kaku akan sulit atau tidak mungkin dikembalikan serta menuntut inspeksi mendalam sebelum diputuskan.) Setelah Anda masuk ke dalam proses produksi, beban kerja Anda akan meningkat saat Anda menambahkan fitur baru dan mengubah implementasi teknologi. Arsitektur beban kerja berubah seiring waktu. Anda perlu menerapkan praktik pemeliharaan yang sesuai untuk mempertahankan karakteristik arsitektur agar tidak memburuk saat Anda mengembangkannya. Saat Anda membuat perubahan arsitektur yang signifikan, Anda harus melakukannya sesuai dengan aturan proses pemeliharaan termasuk peninjauan Well-Architected.

Jika Anda ingin menggunakan hasil tinjauan sebagai snapshot sekali pakai atau pengukuran independen, Anda perlu memastikan bahwa Anda berdiskusi dengan orang yang tepat. Kita sering kali mendapati bahwa tinjauan adalah hal pertama yang dapat membuat tim benar-benar paham tentang apa yang telah mereka implementasikan. Pendekatan yang sangat cocok digunakan saat

meninjau beban kerja tim lain adalah melakukan percakapan informal tentang arsitektur mereka dengan mencermati jawaban-jawaban dari pertanyaan. Kemudian Anda dapat menindaklanjutinya dengan satu atau dua pertemuan untuk mendapatkan kejelasan atau mendalami area yang masih ambigu atau berisiko.

Berikut adalah beberapa saran item untuk memfasilitasi rapat Anda:

- Ruang rapat yang dilengkapi papan tulis
- Cetakan diagram atau catatan desain
- Daftar pertanyaan yang memerlukan riset luar jaringan untuk menjawabnya (misalnya, “apakah kita sudah mengaktifkan enkripsi?”)

Setelah peninjauan selesai, Anda harus membuat daftar masalah yang dapat diprioritaskan berdasarkan konteks bisnis Anda. Anda juga perlu memperhitungkan dampak masalah-masalah tersebut terhadap kinerja harian tim Anda. Jika masalah tersebut segera diatasi, Anda memiliki lebih banyak waktu untuk meningkatkan nilai bisnis ketimbang sibuk dengan masalah yang berulang. Saat Anda mengatasi masalah, Anda dapat memperbarui tinjauan Anda untuk melihat perkembangan arsitektur.

Saat nilai dari tinjauan sudah diketahui dengan jelas, pada awalnya tim baru mungkin akan menafikannya. Berikut adalah beberapa kekurangan yang dapat ditangani dengan mengedukasi tim terkait manfaat tinjauan:

- “Kita sangat sibuk!” (Sering diucapkan ketika tim bersiap untuk peluncuran besar.)
  - Jika Anda bersiap untuk peluncuran besar, Anda menginginkan hal tersebut berjalan lancar. Tinjauan memungkinkan Anda memahami masalah apa pun yang mungkin terlewat.
  - Sebaiknya, lakukan peninjauan sedini mungkin dalam siklus hidup produk untuk mengetahui risiko dan menyiapkan rencana mitigasi yang selaras dengan peta strategi (roadmap) penyampaian fitur.
- “Kita tidak punya waktu untuk mengejar hasil itu!” (Sering diucapkan ketika ada acara dengan jadwal yang sudah tetap, seperti Super Bowl, yang ditargetkan.)
  - Acara-acara tersebut memiliki jadwal yang tetap. Apakah Anda benar-benar ingin melakukan sesuatu tanpa mengetahui risikonya terhadap arsitektur Anda? Bahkan jika Anda tidak mengatasi masalah-masalah ini, Anda masih dapat menggunakan buku panduan untuk menanganinya apabila hal itu terjadi.
- “Kita harus menjaga rahasia implementasi solusi kita dari pihak lain!”

- Jika Anda menyodorkan pertanyaan kepada tim terkait Kerangka Kerja Well-Architected, mereka akan melihat bahwa tidak ada satu pun dari pertanyaan tersebut yang mengarah pada informasi hak milik teknis atau komersial apa pun.

Saat Anda melakukan beberapa peninjauan terhadap beberapa tim dalam organisasi, Anda mungkin mengidentifikasi masalah-masalah yang mendasar. Misalnya, Anda mungkin mendapati grup dari beberapa tim memiliki klaster masalah di pilar atau topik tertentu. Anda perlu melihat semua tinjauan secara menyeluruh, kemudian mengidentifikasi mekanisme, pelatihan, atau pembicaraan teknis utama yang dapat membantu Anda mengetahui masalah mendasar tersebut.

Kesimpulan

AWS Kerangka Kerja Well-Architected memberikan praktik terbaik arsitektur di keenam pilar untuk mendesain dan mengoperasikan sistem yang andal, aman, efisien, hemat, dan berkelanjutan di cloud. Kerangka kerja ini memberikan serangkaian pertanyaan yang memungkinkan Anda untuk meninjau arsitektur yang ada atau yang diusulkan. Kerangka kerja ini juga memberikan serangkaian AWS praktik terbaik untuk setiap pilar. Menggunakan Kerangka kerja dalam arsitektur Anda akan membantu Anda menghasilkan sistem yang stabil dan efisien, yang memungkinkan Anda berfokus pada persyaratan fungsional Anda.

Kontributor

Individu dan organisasi berikut ini memiliki kontribusi dalam dokumen ini:

- Brian Carlson, Operations Lead Well-Architected, Amazon Web Services
- Ben Potter, Security Lead Well-Architected, Amazon Web Services
- Seth Eliot, Reliability Lead Well-Architected, Amazon Web Services
- Eric Pullen, Sr. Solutions Architect, Amazon Web Services
- Rodney Lester, Principal Solutions Architect, Amazon Web Services
- Jon Steele, Sr. Technical Account Manager, Amazon Web Services
- Max Ramsay, Principal Security Solutions Architect, Amazon Web Services
- Callum Hughes, Solutions Architect, Amazon Web Services
- Aden Leirer, Content Program Manager Well-Architected, Amazon Web Services

Sumber Bacaan Lebih Lanjut

Pusat Arsitektur AWS

Kepatuhan Cloud AWS

Program Partner AWS Well-Architected

AWS Well-Architected Tool

Halaman beranda AWS Well-Architected

Laporan resmi Pilar Keunggulan Operasional

Laporan resmi Pilar Keamanan

Laporan Resmi Pilar Keandalan

Laporan resmi Pilar Efisiensi Kinerja

Laporan resmi Pilar Optimasi Biaya

Laporan resmi Pilar Pelestarian Lingkungan

Amazon Builders' Library

Revisi Dokumen

Berlangganan umpan RSS untuk memperoleh pemberitahuan tentang pembaruan laporan resmi ini.

Perubahan	Deskripsi	Tanggal
Pembaruan besar	Restrukturisasi pilar kinerja besar untuk mengubah jumlah area praktik terbaik menjadi lima. Pembaruan besar pada praktik terbaik dan panduan dalam pilar keamanan di Respons insiden (SEC 10). Perubahan konten besar dan penggabungan di area keunggulan operasional OPS 04, 05, 06, 08, dan 09. Pembaruan panduan di seluruh pilar optimisasi biaya dan keandalan . Pembaruan kecil pada tingkat risiko pilar pelestarian lingkungan .	October 3, 2023
Pembaruan untuk Kerangka Kerja baru	Praktik terbaik diperbarui dengan panduan preskriptif dan praktik terbaik baru ditambahkan. Pertanyaan baru ditambahkan ke pilar Keamanan dan Optimasi Biaya.	April 10, 2023
Pembaruan kecil	Penambahan definisi untuk tingkat upaya dan pembaruan praktik terbaik di lampiran.	October 20, 2022

Laporan resmi diperbarui	Penambahan Pilar Pelestarian Lingkungan dan pembaruan tautan.	December 2, 2021
Pembaruan besar	Pilar Pelestarian Lingkungan ditambahkan ke kerangka kerja.	November 20, 2021
Pembaruan kecil	Bahasa noninklusif dihilangkan.	April 22, 2021
Pembaruan kecil	Perbaikan banyak tautan.	March 10, 2021
Pembaruan kecil	Perubahan editorial kecil di seluruh dokumen.	July 15, 2020
Pembaruan untuk Kerangka Kerja baru	Peninjauan dan penulisan ulang sebagian besar pertanyaan dan jawaban.	July 8, 2020
Laporan resmi diperbarui	Penambahan AWS Well-Architected Tool, tautan ke AWS Well-Architected Labs, dan AWS Well-Architected Partners, perbaikan kecil untuk memungkinkan kerangka kerja versi multibahasa.	July 1, 2019

Laporan resmi diperbarui	Peninjauan dan penulisan ulang sebagian besar pertanyaan dan jawaban, untuk memastikan pertanyaan berfokus pada satu topik pada satu waktu. Ini menyebabkan beberapa pertanyaan sebelumnya dipecah menjadi lebih dari satu pertanyaan. Penambahan istilah umum ke definisi (beban kerja, komponen, dll.). Perubahan penyajian pertanyaan di bagian utama agar menyertakan teks deskriptif.	November 1, 2018
Laporan resmi diperbarui	Pembaruan untuk menyederhanakan teks pertanyaan, menstandarkan jawaban, dan meningkatkan keterbacaan.	June 1, 2018
Laporan resmi diperbarui	Pilar Keunggulan Operasional dipindah ke depan dan ditulis ulang agar memayungi pilar-pilar lain. Penyegaran pilar-pilar lain agar mencerminkan perkembangan AWS.	November 1, 2017
Laporan resmi diperbarui	Pembaruan Framework agar menyertakan pilar keunggulan operasional, dan revisi serta pembaruan pilar-pilar lain untuk mengurangi duplikasi dan memasukkan pembelajaran dari pelaksanaan peninjauan dengan ribuan pelanggan.	November 1, 2016

Pembaruan kecil	Pembaruan Lampiran dengan informasi Amazon CloudWatch Logs terkini.	November 1, 2015
Publikasi awal	AWS Well-Architected Framework dipublikasikan.	October 1, 2015

Lampiran: Pertanyaan dan praktik terbaik

Lampiran ini merangkum semua pertanyaan dan praktik terbaik di Kerangka Kerja AWS Well-Architected.

Pilar

- Keunggulan operasional
- Keamanan
- Keandalan
- Efisiensi kinerja
- Optimisasi biaya
- Pelestarian Lingkungan

Keunggulan operasional

Pilar Keunggulan Operasional mencakup kemampuan untuk mendukung pengembangan dan menjalankan beban kerja dengan efektif, mendapatkan wawasan tentang operasi Anda, serta meningkatkan proses dan prosedur pendukung secara terus-menerus untuk memberikan nilai bisnis. Anda dapat menemukan panduan preskriptif tentang implementasi di Laporan resmi Pilar Keunggulan Operasional.

Area praktik terbaik

- Organisasi
- Persiapan
- Jalankan
- Kembangkan

Organisasi

Pertanyaan

- OPS 1. Bagaimana cara menentukan prioritas Anda?
- OPS 2. Bagaimana cara menyusun struktur organisasi untuk mendukung hasil bisnis Anda?

- OPS 3. Bagaimana budaya organisasi Anda mendukung hasil bisnis Anda?

OPS 1. Bagaimana cara menentukan prioritas Anda?

Setiap orang harus memahami peran mereka dalam mewujudkan kesuksesan bisnis. Miliki sasaran bersama guna menetapkan prioritas untuk sumber daya. Ini akan memaksimalkan manfaat dari upaya Anda.

Praktik terbaik

- OPS01-BP01 Mengevaluasi kebutuhan pelanggan eksternal
- OPS01-BP02 Mengevaluasi kebutuhan pelanggan internal
- OPS01-BP03 Evaluasi persyaratan tata kelola
- OPS01-BP04 Evaluasi persyaratan kepatuhan
- OPS01-BP05 Mengevaluasi lanskap ancaman
- OPS01-BP06 Mengevaluasi kompromi
- OPS01-BP07 Kelola manfaat dan risiko

OPS01-BP01 Mengevaluasi kebutuhan pelanggan eksternal

Libatkan pemangku kepentingan utama, termasuk tim bisnis, pengembangan, dan operasional, untuk menentukan ke mana harus memfokuskan usaha terkait kebutuhan pelanggan eksternal. Hal ini akan memastikan bahwa Anda memiliki pemahaman menyeluruh mengenai dukungan operasi yang dibutuhkan untuk mencapai hasil bisnis yang diinginkan.

Antipola umum:

- Anda memutuskan untuk tidak menyediakan dukungan pelanggan di luar jam kerja, tetapi Anda belum meninjau riwayat data permintaan dukungan. Anda tidak tahu apakah hal ini akan memengaruhi pelanggan Anda.
- Anda mengembangkan fitur baru tetapi belum melibatkan pelanggan untuk mencari tahu apakah hal tersebut diinginkan—jika diinginkan, dalam bentuk apa—dan belum menjalankan eksperimen untuk memvalidasi kebutuhan serta metode pengiriman.

Manfaat menerapkan praktik terbaik ini: Pelanggan yang kebutuhannya terpenuhi berpotensi menjadi pelanggan tetap. Mengevaluasi dan memahami kebutuhan pelanggan eksternal akan menginformasikan cara Anda memprioritaskan usaha untuk memberikan nilai bisnis.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Pahami kebutuhan bisnis: Kesuksesan bisnis dapat terwujud dengan adanya tujuan dan pemahaman bersama di seluruh pemangku kepentingan, termasuk tim bisnis, pengembangan, dan operasional.
  - Tinjau tujuan bisnis, kebutuhan, dan prioritas pelanggan eksternal: Libatkan pemangku kepentingan utama, termasuk tim bisnis, pengembangan, dan operasional, untuk mendiskusikan tujuan, kebutuhan, dan prioritas pelanggan eksternal. Hal ini memastikan bahwa Anda memiliki pemahaman menyeluruh mengenai dukungan operasional yang dibutuhkan untuk mencapai hasil bisnis dan pelanggan.
  - Tetapkan pemahaman bersama: Tetapkan pemahaman bersama terkait fungsi bisnis beban kerja, peran masing-masing tim dalam mengoperasikan beban kerja, dan bagaimana faktor-faktor ini mendukung tujuan bisnis bersama bagi seluruh pelanggan internal dan eksternal.

Sumber daya

Dokumen terkait:

- Konsep Kerangka Kerja AWS Well-Architected – Loop umpan balik

OPS01-BP02 Mengevaluasi kebutuhan pelanggan internal

Libatkan pemangku kepentingan utama, termasuk tim bisnis, pengembangan, dan operasional, untuk menentukan ke mana harus memfokuskan usaha terkait kebutuhan pelanggan internal. Hal ini akan memastikan bahwa Anda memiliki pemahaman menyeluruh mengenai dukungan operasi yang dibutuhkan untuk mencapai hasil bisnis yang diinginkan.

Gunakan prioritas yang ditetapkan untuk memfokuskan usaha peningkatan yang dapat memberikan dampak paling besar (misalnya, mengembangkan keterampilan tim, meningkatkan kinerja beban kerja, mengurangi biaya, mengotomatiskan runbook, atau meningkatkan pemantauan). Perbarui prioritas Anda sesuai perubahan kebutuhan.

Antipola umum:

- Anda memutuskan untuk mengubah alokasi alamat IP untuk tim produk tanpa berkonsultasi dengan mereka agar manajemen jaringan menjadi lebih mudah. Anda tidak tahu dampak yang akan ditimbulkan kepada tim produk.

- Anda mengimplementasikan alat pengembangan baru tetapi belum melibatkan pelanggan internal untuk mencari tahu apakah alat itu dibutuhkan atau kompatibel dengan praktik yang sudah ada.
- Anda mengimplementasikan sistem pemantauan baru tetapi belum menghubungi pelanggan internal untuk mencari tahu apakah mereka memiliki kebutuhan pemantauan atau pelaporan yang perlu dipertimbangkan.

Manfaat menerapkan praktik terbaik ini: Mengevaluasi dan memahami kebutuhan pelanggan internal akan menginformasikan cara Anda memprioritaskan usaha untuk memberikan nilai bisnis.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Pahami kebutuhan bisnis: Kesuksesan bisnis dapat terwujud dengan tujuan dan pemahaman bersama di seluruh pemangku kepentingan, termasuk tim bisnis, pengembangan, dan operasional.
  - Tinjau tujuan bisnis, kebutuhan, dan prioritas pelanggan internal: Libatkan pemangku kepentingan utama, termasuk tim bisnis, pengembangan, dan operasional, untuk mendiskusikan tujuan, kebutuhan, dan prioritas pelanggan internal. Hal ini memastikan bahwa Anda memiliki pemahaman menyeluruh mengenai dukungan operasional yang dibutuhkan untuk mencapai hasil bisnis dan pelanggan.
  - Tetapkan pemahaman bersama: Tetapkan pemahaman bersama terkait fungsi bisnis beban kerja, peran masing-masing tim dalam mengoperasikan beban kerja, dan bagaimana faktor-faktor ini mendukung tujuan bisnis bersama bagi seluruh pelanggan internal dan eksternal.

Sumber daya

Dokumen terkait:

- Konsep Kerangka Kerja AWS Well-Architected – Loop umpan balik

OPS01-BP03 Evaluasi persyaratan tata kelola

Tata kelola adalah serangkaian kebijakan, aturan, atau kerangka kerja yang digunakan perusahaan untuk mencapai tujuan bisnisnya. Persyaratan tata kelola dibuat dari dalam organisasi Anda. Persyaratan ini dapat memengaruhi jenis teknologi yang Anda pilih atau memengaruhi cara Anda mengoperasikan beban kerja Anda. Sertakan persyaratan tata kelola organisasi ke dalam

beban kerja Anda. Konformitas adalah kemampuan untuk menunjukkan bahwa Anda telah mengimplementasikan persyaratan tata kelola.

Hasil yang diinginkan:

- Persyaratan tata kelola disertakan ke dalam operasi dan desain arsitektur beban kerja Anda.
- Anda dapat memberikan bukti bahwa Anda telah mengikuti persyaratan tata kelola.
- Persyaratan tata kelola ditinjau dan diperbarui secara teratur.

Antipola umum:

- Organisasi Anda memerintahkan agar akun root memiliki autentikasi multi-faktor. Anda gagal mengimplementasikan persyaratan ini dan akun root terancam bahaya.
- Selama desain beban kerja Anda, Anda memilih jenis instans yang tidak disetujui oleh departemen IT. Anda tidak dapat meluncurkan beban kerja Anda dan harus mendesain ulang.
- Anda diwajibkan memiliki rencana pemulihan bencana. Anda tidak membuat rencana pemulihan bencana dan beban kerja Anda mengalami pemadaman yang berdurasi lama.
- Tim Anda ingin menggunakan instans baru tetapi persyaratan tata kelola Anda belum diperbarui untuk memungkinkannya.

Manfaat menjalankan praktik terbaik ini:

- Mengikuti persyaratan tata kelola akan menyelaraskan beban kerja Anda dengan kebijakan lebih besar dalam organisasi.
- Persyaratan tata kelola mencerminkan standar industri dan praktik terbaik untuk organisasi Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Identifikasi persyaratan tata kelola melalui kerja sama dengan pemangku kepentingan dan organisasi tata kelola. Sertakan persyaratan tata kelola ke dalam beban kerja Anda. Dapat menunjukkan bukti bahwa Anda telah mengikuti persyaratan tata kelola.

Contoh pelanggan

Di AnyCompany Retail, tim operasi cloud bekerja sama dengan pemangku kepentingan di seluruh organisasi untuk mengembangkan persyaratan tata kelola. Contohnya, mereka melarang akses SSH

ke instans Amazon EC2. Jika tim memerlukan akses ke sistem, mereka harus menggunakan AWS Systems Manager Session Manager. Tim operasi cloud secara teratur memperbarui persyaratan tata kelola saat layanan baru tersedia.

Langkah implementasi

1. Identifikasi pemangku kepentingan untuk beban kerja Anda, termasuk semua tim terpusat.
2. Bekerja sama dengan pemangku kepentingan untuk mengidentifikasi persyaratan tata kelola.
3. Setelah Anda membuat daftar, prioritaskan item untuk ditingkatkan, dan mulai implementasikan ke dalam beban kerja Anda.
   a. Gunakan layanan seperti AWS Config untuk membuat tata kelola sebagai kode dan validasikan bahwa persyaratan tata kelola telah diikuti.
   b. Jika Anda menggunakan AWS Organizations, Anda dapat memanfaatkan Kebijakan Kontrol Layanan untuk mengimplementasikan persyaratan tata kelola.
4. Berikan dokumentasi yang memvalidasi implementasinya.

Tingkat upaya untuk rencana implementasi: Sedang. Mengimplementasikan persyaratan tata kelola yang tidak ada dapat mengakibatkan beban kerja Anda harus dikerjakan ulang.

Sumber daya

Praktik terbaik terkait:

- OPS01-BP04 Evaluasi persyaratan kepatuhan - Kepatuhan sama seperti tata kelola tetapi berasal dari luar organisasi.

Dokumen terkait:

- Panduan untuk Manajemen dan Tata Kelola Lingkungan Cloud AWS
- Praktik Terbaik untuk Kebijakan Kontrol Layanan AWS Organizations dalam Lingkungan Multi-Akun
- Tata Kelola dalam AWS Cloud: Keseimbangan yang Tepat Antara Ketangkasan dan Keamanan
- Apa yang dimaksud Tata Kelola, Risiko, dan Kepatuhan (GRC)?

Video terkait:

- AWS Manajemen dan Tata Kelola: Konfigurasi, Kepatuhan, dan Audit - Online Tech Talks AWS

- AWS re:Inforce 2019: Tata Kelola untuk Era Cloud (DEM12-R1)
- AWS re:Invent 2020: Capai kepatuhan sebagai kode menggunakan AWS Config
- AWS re:Invent 2020: Tata kelola tangkas di AWS GovCloud (US)

Contoh terkait:

- Sampel Paket Kepatuhan AWS Config

Layanan terkait:

- AWS Config
- AWS Organizations - Kebijakan Kontrol Layanan

OPS01-BP04 Evaluasi persyaratan kepatuhan

Persyaratan kepatuhan peraturan, industri, dan internal merupakan pendorong penting dalam menentukan prioritas organisasi Anda. Kerangka kerja kepatuhan Anda dapat menghalangi Anda menggunakan teknologi atau lokasi geografi tertentu. Terapkan uji tuntas jika tidak ada kerangka kerja kepatuhan eksternal yang diidentifikasi. Buat audit atau laporan yang memvalidasi kepatuhan.

Jika Anda mengiklankan bahwa produk Anda memenuhi standar kepatuhan tertentu, Anda harus memiliki proses internal untuk memastikan kepatuhan yang berkelanjutan. Contoh standar kepatuhan antara lain PCI DSS, FedRAMP, dan HIPAA. Standar kepatuhan yang berlaku akan ditentukan oleh berbagai faktor, seperti jenis data yang disimpan atau dikirim oleh solusi dan wilayah geografis mana yang didukung oleh solusi.

Hasil yang diinginkan:

- Persyaratan kepatuhan peraturan, industri, dan internal disertakan ke dalam pemilihan arsitektur.
- Anda dapat memvalidasi kepatuhan dan membuat laporan audit.

Antipola umum:

- Bagian dari beban kerja Anda termasuk dalam kerangka kerja Standar Keamanan Data Industri Kartu Pembayaran (Payment Card Industry Data Security Standard, PCI-DSS) tetapi beban kerja Anda menyimpan data kartu kredit tanpa enkripsi.

- Arsitek dan developer perangkat lunak Anda tidak mengetahui kerangka kerja kepatuhan yang harus ditaati oleh organisasi Anda.
- Audit tahunan Kontrol Sistem dan Organisasi (SOC2) Tipe II akan segera diadakan dan Anda tidak dapat memverifikasi bahwa kontrol sudah ada.

Manfaat menjalankan praktik terbaik ini:

- Mengevaluasi dan memahami persyaratan kepatuhan yang berlaku untuk beban kerja Anda akan menginformasikan bagaimana Anda memprioritaskan usaha untuk memberikan nilai bisnis.
- Anda memilih teknologi dan lokasi yang tepat yang selaras dengan kerangka kerja kepatuhan Anda.
- Mendesain beban kerja Anda agar dapat diaudit memungkinkan Anda membuktikan bahwa Anda menaati kerangka kerja kepatuhan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Mengimplementasikan praktik terbaik ini berarti Anda menyertakan persyaratan kepatuhan ke dalam proses desain arsitektur. Anggota tim Anda mengetahui kerangka kerja kepatuhan yang diperlukan. Anda memvalidasi kepatuhan selaras dengan kerangka kerja.

Contoh pelanggan

AnyCompany Retail menyimpan informasi kartu kredit bagi pelanggan. Developer di tim penyimpanan kartu memahami bahwa mereka harus mematuhi kerangka kerja PCI-DSS. Mereka telah mengambil langkah untuk memverifikasi bahwa informasi kartu kredit disimpan dan diakses dengan aman sesuai dengan kerangka kerja PCI-DSS. Setiap tahun mereka bekerja sama dengan tim keamanan mereka untuk memvalidasi kepatuhan.

Langkah implementasi

1. Bekerjasamalah dengan tim tata kelola dan kepatuhan Anda untuk menentukan kerangka kerja kepatuhan industri, peraturan, atau internal apa yang harus ditaati beban kerja Anda. Sertakan kerangka kerja kepatuhan ke dalam beban kerja Anda.
   a. Validasi kepatuhan sumber daya AWS yang berkelanjutan dengan layanan seperti AWS Compute Optimizer dan AWS Security Hub CSPM.

2. Didik anggota tim Anda tentang persyaratan kepatuhan sehingga mereka dapat mengoperasikan dan mengubah beban kerja sesuai dengan persyaratan kepatuhan. Persyaratan kepatuhan harus disertakan dalam pilihan arsitektur dan teknologi.
3. Tergantung pada kerangka kerja kepatuhannya, Anda mungkin diharuskan untuk membuat laporan kepatuhan atau audit. Bekerjasamalah dengan organisasi Anda untuk mengotomatiskan proses ini sebanyak mungkin.
   a. Gunakan layanan seperti AWS Audit Manager untuk memvalidasi kepatuhan dan membuat laporan audit.
   b. Anda dapat mengunduh dokumen kepatuhan dan keamanan AWS dengan AWS Artifact.

Tingkat upaya untuk rencana implementasi: Sedang. Mengimplementasikan kerangka kerja kepatuhan bisa sulit dilakukan. Membuat laporan audit atau dokumen kepatuhan menambahkan kompleksitas tambahan.

Sumber daya

Praktik terbaik terkait:

- SEC01-BP03 Mengidentifikasi dan memvalidasi tujuan kontrol - Tujuan kontrol keamanan merupakan bagian penting dari kepatuhan secara keseluruhan.
- SEC01-BP06 Mengotomatiskan pengujian dan validasi kontrol keamanan di pipeline - Sebagai bagian dari pipeline Anda, validasikan kontrol keamanan. Anda juga dapat membuat dokumentasi kepatuhan untuk perubahan baru.
- SEC07-BP02 Menentukan kontrol perlindungan data - Sejumlah besar kerangka kerja kepatuhan didasarkan pada penanganan data dan kebijakan penyimpanan.
- SEC10-BP03 Menyiapkan kemampuan forensik - Kemampuan forensik terkadang dapat digunakan dalam mengaudit kepatuhan.

Dokumen terkait:

- Pusat Kepatuhan AWS
- Sumber Daya Kepatuhan AWS
- Laporan Resmi Kepatuhan dan Risiko AWS
- Model Tanggung Jawab Bersama AWS
- Layanan AWS dalam cakupan berdasarkan program kepatuhan

Video terkait:

- AWS re:Invent 2020: Capai kepatuhan sebagai kode menggunakan AWS Compute Optimizer
- AWS re:Invent 2021 - Kepatuhan cloud, jaminan, dan audit
- AWS Summit ATL 2022 - Mengimplementasikan kepatuhan, jaminan, dan audit di AWS (COP202)

Contoh terkait:

- PCI DSS dan Praktik Terbaik Keamanan Mendasar AWS di AWS

Layanan terkait:

- AWS Artifact
- AWS Audit Manager
- AWS Compute Optimizer
- AWS Security Hub CSPM

OPS01-BP05 Mengevaluasi lanskap ancaman

Evaluasi ancaman pada bisnis (misalnya, persaingan, risiko dan kewajiban bisnis, risiko operasional, serta ancaman keamanan informasi) dan pelihara informasi yang ada di registri risiko. Sertakan dampak risiko ketika menentukan ke mana upaya harus difokuskan.

Kerangka kerja Well-Architected Framework menekankan pembelajaran, pengukuran, dan peningkatan. Framework menyediakan pendekatan yang konsisten bagi Anda untuk mengevaluasi arsitektur, dan mengimplementasikan desain yang akan mengalami penskalaan seiring waktu. AWS menyediakan AWS Well-Architected Tool untuk membantu Anda meninjau pendekatan sebelum pengembangan, status beban kerja Anda sebelum produksi, dan status beban kerja Anda dalam produksi. Anda dapat membandingkannya dengan praktik terbaik arsitektur AWS terkini, memantau keseluruhan status beban kerja Anda, dan mendapatkan wawasan tentang potensi risiko.

Pelanggan AWS memenuhi syarat untuk Tinjauan Well-Architected terpandu tentang beban kerja misi penting mereka untuk mengukur arsitektur mereka berdasarkan praktik terbaik AWS. Pelanggan Dukungan Korporat memenuhi syarat untuk Tinjauan Operasi, yang dirancang untuk membantu mereka mengidentifikasi celah dalam pendekatan operasi di cloud mereka.

Interaksi lintas tim pada tinjauan ini membantu membangun pemahaman bersama tentang beban kerja Anda serta bagaimana peran tim membantu meraih keberhasilan. Kebutuhan yang diidentifikasi melalui tinjauan dapat membantu membentuk prioritas Anda.

AWS Trusted Advisor adalah alat yang menyediakan akses ke set inti pemeriksaan yang menyarankan optimalisasi yang dapat membantu membentuk prioritas Anda. Pelanggan Dukungan Bisnis dan Korporat menerima akses ke pemeriksaan tambahan yang berfokus pada keamanan, keandalan, kinerja, dan optimalisasi biaya yang dapat membantu membentuk prioritas mereka lebih lanjut.

Antipola umum:

- Anda menggunakan pustaka perangkat lunak versi lama dalam produk Anda. Anda tidak tahu bahwa ada pembaruan keamanan pustaka untuk masalah yang mungkin memiliki dampak yang tidak diinginkan pada beban kerja Anda.
- Kompetitor Anda baru saja merilis versi produk mereka yang mengatasi keluhan pelanggan Anda tentang produk Anda. Anda belum memprioritaskan penanganan masalah-masalah yang dikenal ini.
- Pembuat peraturan telah menyasar perusahaan yang tidak mematuhi persyaratan kepatuhan hukum seperti Anda. Anda belum memprioritaskan penanganan persyaratan kepatuhan Anda yang belum terpenuhi.

Manfaat menjalankan praktik terbaik ini: Identifikasi dan pemahaman tentang ancaman terhadap organisasi dan beban kerja Anda dapat membantu Anda menentukan ancaman mana yang harus ditangani, tingkat prioritasnya, serta sumber daya yang diperlukan untuk melakukannya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

- Evaluasi lanskap ancaman: Evaluasi ancaman terhadap bisnis (misalnya kompetisi, risiko dan kewajiban bisnis, risiko operasional, dan ancaman keamanan informasi), sehingga Anda dapat menyertakan dampaknya ketika menentukan ke mana upaya perlu difokuskan.
  - Buletin Keamanan Terkini AWS
  - AWS Trusted Advisor
  - Pelihara model ancaman: Buat dan pelihara model ancaman yang mengidentifikasi potensi ancaman, mitigasi terencana dan sedang diterapkan, serta prioritasnya. Tinjau kemungkinan ancaman yang berwujud insiden, biaya untuk melakukan pemulihan dari insiden tersebut serta

perkiraan bahaya yang ditimbulkan, dan biaya untuk mencegah insiden tersebut. Revisi proritas seiring perubahan konten model ancaman.

Sumber daya

Dokumen terkait:

- Kepatuhan AWS Cloud
- Buletin Keamanan Terkini AWS
- AWS Trusted Advisor

OPS01-BP06 Mengevaluasi kompromi

Evaluasi dampak kompromi antarkepentingan yang bertentangan atau pendekatan alternatif, untuk membantu mengambil keputusan yang matang saat menentukan ke mana upaya perlu difokuskan atau memiliki opsi tindakan. Misalnya, meningkatkan kecepatan masuk pasar untuk fitur baru dapat diprioritaskan daripada optimalisasi biaya, atau Anda bisa memilih basis data relasional untuk data non-relasional guna menyederhanakan upaya migrasi sistem, dibandingkan bermigrasi ke basis data yang dioptimalkan untuk tipe data Anda dan memperbarui aplikasi Anda.

AWS dapat membantu mengedukasi tim Anda tentang AWS dan layanannya untuk meningkatkan pemahaman mereka tentang bagaimana pilihan mereka dapat menimbulkan dampak pada beban kerja. Anda harus menggunakan sumber daya yang disediakan oleh AWS Dukungan (Pusat Pengetahuan AWS, Forum Diskusi AWS, dan Pusat AWS Dukungan) dan Dokumentasi AWS untuk mengedukasi tim Anda. Hubungi AWS Dukungan melalui Pusat AWS Dukungan jika Anda memiliki pertanyaan seputar AWS.

AWS juga membagikan pola dan praktik terbaik yang telah kami pelajari melalui operasi AWS di Amazon Builders' Library. Beragam jenis informasi berguna lainnya dapat diakses melalui Blog AWS dan Podcast AWS Resmi.

Antipola umum:

- Anda menggunakan basis data relasional untuk mengelola data seri waktu dan non-relasional. Terdapat opsi-opsi basis data yang dioptimalkan untuk mendukung tipe data yang Anda gunakan tetapi Anda tidak menyadari manfaatnya karena Anda belum mengevaluasi kompromi antarsolusi.
- Investor Anda meminta Anda mendemonstrasikan kepatuhan terhadap Standar Keamanan Data Industri Kartu Pembayaran (PCI DSS). Anda tidak mempertimbangkan kompromi antara

memenuhi permintaan mereka dan melanjutkan upaya pengembangan Anda saat ini. Alih-alih, Anda melanjutkan upaya pengembangan tanpa menunjukkan kepatuhan. Investor Anda menghentikan dukungan untuk perusahaan Anda karena mengkhawatirkan keamanan platform Anda serta investasi mereka.

Manfaat menjalankan praktik terbaik ini: Memahami implikasi dan konsekuensi pilihan yang Anda ambil dapat membantu Anda membuat prioritas opsi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

- Evaluasi kompromi: Evaluasi dampak kompromi antarkepentingan yang bertentangan untuk membantu mengambil keputusan yang matang saat menentukan ke mana upaya perlu difokuskan. Misalnya, mempercepat waktu masuk pasar untuk fitur baru dapat diprioritaskan daripada optimalisasi biaya.
- AWS dapat membantu mengedukasi tim Anda tentang AWS dan layanannya untuk meningkatkan pemahaman mereka tentang bagaimana pilihan mereka dapat menimbulkan dampak pada beban kerja. Anda harus menggunakan sumber daya yang disediakan oleh AWS Dukungan (Pusat Pengetahuan AWS, Forum DiskusiAWS, dan Pusat AWS Dukungan) serta Dokumentasi AWS untuk mengedukasi tim Anda. Hubungi AWS Dukungan melalui Pusat AWS Dukungan jika Anda memiliki pertanyaan seputar AWS.
- AWS juga membagikan pola dan praktik terbaik yang telah kami pelajari melalui operasi AWS di Amazon Builders' Library. Beragam jenis informasi berguna lainnya dapat diakses melalui Blog AWS dan Podcast AWS Resmi.

Sumber daya

Dokumen terkait:

- Blog AWS
- Kepatuhan AWS Cloud
- Forum Diskusi AWS
- Dokumentasi AWS
- Pusat Pengetahuan AWS
- AWS Dukungan

- Pusat AWS Dukungan
- Amazon Builders' Library
- Podcast AWS Resmi

OPS01-BP07 Kelola manfaat dan risiko

Kelola manfaat dan risiko untuk mengambil keputusan yang bijaksana ketika menentukan di mana akan memfokuskan upaya. Contohnya, mungkin akan bermanfaat untuk melakukan deploy beban kerja dengan masalah yang tak terselesaikan sehingga fitur baru yang signifikan dapat dibuat tersedia bagi pelanggan. Risiko terkait mungkin dapat dimitigasi, atau membiarkan risiko tetap ada mungkin menjadi tidak dapat diterima, jika demikian, Anda akan mengambil tindakan untuk mengatasi risiko tersebut.

Anda mungkin mendapatkan bahwa Anda ingin menekankan subset kecil prioritas pada titik waktu tertentu. Gunakan pendekatan yang seimbang dalam jangka panjang untuk memastikan pengembangan kemampuan yang diperlukan dan pengelolaan risiko. Perbarui prioritas Anda sesuai perubahan kebutuhan

Antipola umum:

- Anda telah memutuskan untuk menyertakan pustaka yang melakukan semua yang Anda perlukan yang ditemukan salah satu developer Anda di internet. Anda belum mengevaluasi risiko adopsi pustaka ini dari sumber tak dikenal dan Anda tidak tahu jika pustaka memiliki kelemahan atau kode jahat.
- Anda telah memutuskan untuk mengembangkan dan melakukan deploy fitur baru dan bukannya memperbaiki masalah yang ada. Anda belum mengevaluasi risiko meninggalkan masalah sampai fitur dilakukan deploy dan Anda tidak tahu apa saja dampaknya pada pelanggan Anda.
- Anda telah memutuskan untuk tidak melakukan deploy fitur yang sering diminta oleh pelanggan karena masalah yang tidak jelas dari tim kepatuhan Anda.

Manfaat menerapkan praktik terbaik ini: Mengidentifikasi manfaat yang tersedia dari pilihan Anda, dan menyadari risiko terhadap organisasi Anda, memungkinkan Anda untuk mengambil keputusan yang bijaksana.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Rendah

Panduan implementasi

- Kelola manfaat dan risiko: Seimbangkan manfaat dari keputusan terhadap risiko yang terlibat.
  - Identifikasi manfaat: Identifikasi manfaat berdasarkan tujuan, kebutuhan, dan prioritas bisnis. Contohnya antara lain waktu masuk pasar, keamanan, keandalan, performa, dan biaya.
  - Identifikasi risiko: Identifikasi risiko berdasarkan tujuan, kebutuhan, dan prioritas bisnis. Contohnya antara lain waktu masuk pasar, keamanan, keandalan, performa, dan biaya.
  - Evaluasi manfaat dibandingkan risiko dan ambil keputusan yang bijaksana: Tentukan dampak manfaat dan risiko berdasarkan tujuan, kebutuhan, dan prioritas pemangku kepentingan utama Anda, termasuk bagian bisnis, pengembangan, dan operasi. Evaluasi nilai manfaat dibandingkan dengan probabilitas terwujudnya risiko dan kerugian dampaknya. Contohnya, menekankan kecepatan masuk pasar dan bukannya keandalan dapat memberikan keunggulan yang bersaing. Tetapi, ini dapat mengakibatkan berkurangnya waktu aktif jika ada masalah keandalan.

OPS 2. Bagaimana cara menyusun struktur organisasi untuk mendukung hasil bisnis Anda?

Tim Anda harus memahami peran mereka dalam mencapai hasil bisnis. Tim harus memahami peran mereka dalam kesuksesan tim lain, peran tim lain dalam kesuksesan mereka, dan memiliki sasaran bersama. Memahami tanggung jawab, kepemilikan, bagaimana keputusan diambil, dan siapa yang memiliki otoritas untuk mengambil keputusan akan membantu memfokuskan upaya dan memaksimalkan manfaat dari tim Anda.

Praktik terbaik

- OPS02-BP01 Sumber daya memiliki pemilik teridentifikasi
- OPS02-BP02 Proses dan Prosedur memiliki pemilik teridentifikasi
- OPS02-BP03 Aktivitas operasi memiliki pemilik teridentifikasi yang bertanggung jawab atas kinerjanya
- OPS02-BP04 Anggota tim tahu tanggung jawab mereka
- OPS02-BP05 Mekanisme tersedia untuk mengidentifikasi tanggung jawab dan kepemilikan
- OPS02-BP06 Mekanisme tersedia untuk meminta penambahan, perubahan, dan pengecualian
- OPS02-BP07 Tanggung jawab antara tim telah dinegosiasi atau ditetapkan sebelumnya

OPS02-BP01 Sumber daya memiliki pemilik teridentifikasi

Sumber daya untuk beban kerja Anda harus memiliki pemilik yang teridentifikasi untuk pengontrolan perubahan, penyelesaian masalah, dan fungsi-fungsi lainnya. Pemilik ditetapkan untuk beban kerja, akun, infrastruktur, platform, dan aplikasi. Kepemilikan dicatat menggunakan alat seperti daftar sentral atau metadata yang dilampirkan ke sumber daya. Nilai bisnis komponen menginformasikan proses dan prosedur yang diterapkan kepadanya.

Hasil yang diinginkan:

- Sumber daya telah mengidentifikasi pemilik menggunakan metadata atau daftar sentral.
- Anggota tim dapat mengidentifikasi siapa pemilik sumber daya.
- Akun memiliki satu pemilik apabila mungkin.

Antipola umum:

- Kontak alternatif untuk Akun AWS Anda tidak diisi.
- Sumber daya tidak memiliki tag yang mengidentifikasi tim mana yang memilikinya.
- Anda memiliki antrean ITSM tanpa pemetaan email.
- Dua tim sama-sama merupakan pemilik bagian penting dari infrastruktur.

Manfaat menjalankan praktik terbaik ini:

- Kontrol perubahan untuk sumber daya mudah dilakukan dengan ditetapkannya kepemilikan.
- Anda dapat melibatkan pemilik yang benar ketika menyelesaikan masalah.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Tentukan pentingnya kepemilikan untuk kasus penggunaan sumber daya di lingkungan Anda. Kepemilikan dapat berarti siapa yang mengawasi perubahan pada sumber daya, mendukung sumber daya selama penyelesaian masalah, atau siapa yang bertanggung jawab secara finansial. Sebutkan dan catat pemilik untuk sumber daya, termasuk nama, informasi kontak, organisasi, serta tim.

Contoh pelanggan

AnyCompany Retail menetapkan kepemilikan sebagai tim atau individu yang memiliki perubahan dan dukungan untuk sumber daya. Mereka memanfaatkan AWS Organizations untuk mengelola Akun AWS mereka. Kontak akun alternatif dikonfigurasi menggunakan kotak masuk grup. Setiap antrean ITSM dipetakan ke alias email. Tag mengidentifikasi siapa yang memiliki sumber daya AWS. Untuk infrastruktur dan platform lainnya, mereka memiliki halaman wiki yang mengidentifikasi kepemilikan dan informasi kontak.

Langkah implementasi

1. Mulai dengan menetapkan kepemilikan untuk organisasi Anda. Kepemilikan dapat menyiratkan siapa yang memiliki risiko untuk sumber daya, siapa yang memiliki perubahan pada sumber daya, atau siapa yang mendukung sumber daya ketika menyelesaikan masalah. Kepemilikan juga dapat menyiratkan kepemilikan sumber daya secara finansial atau administratif.
2. Gunakan AWS Organizations untuk mengelola akun. Anda dapat mengelola kontak alternatif untuk akun Anda secara terpusat.
   a. Dengan menggunakan alamat email dan nomor telepon milik perusahaan untuk informasi kontak, Anda dapat tetap dapat mengaksesnya meskipun orang yang memilikinya tidak lagi bekerja di organisasi Anda. Misalnya, buat daftar distribusi email terpisah untuk penagihan, operasional, dan keamanan lalu konfigurasikan ketiganya sebagai kontak Penagihan, Keamanan, dan Operasional di setiap Akun AWS yang aktif. Banyak orang akan menerima notifikasi AWS dan dapat merespons, meskipun ada yang sedang berlibur, berganti posisi, atau meninggalkan perusahaan.
   b. Jika akun tidak dikelola oleh AWS Organizations, kontak akun alternatif dapat membantu AWS menghubungi personel yang tepat jika diperlukan. Konfigurasikan kontak alternatif akun sehingga menunjuk ke grup dan bukannya individu.
3. Gunakan tag untuk mengidentifikasi pemilik untuk sumber daya AWS. Anda dapat menentukan pemilik maupun informasi kontak mereka dalam tag terpisah.
   a. Anda dapat menggunakan aturan AWS Config untuk menegakkan bahwa sumber daya memiliki tag kepemilikan yang diperlukan.
   b. Untuk panduan mendalam tentang cara membangun strategi pemberian tag untuk organisasi Anda, lihat AWSlaporan resmi Praktik Terbaik Pemberian Tag.
4. Untuk sumber daya, platform, dan infrastruktur lainnya, buat dokumentasi yang mengidentifikasi kepemilikan. Dokumentasi ini harus dapat diakses oleh semua anggota tim.

Tingkat upaya untuk rencana implementasi: Rendah. Manfaatkan informasi kontak akun dan tag untuk menetapkan kepemilikan sumber daya AWS. Untuk sumber daya lainnya, Anda dapat

menggunakan sesuatu yang sederhana seperti tabel di wiki hingga catatan kepemilikan dan informasi kontak, atau gunakan alat ITSM untuk memetakan kepemilikan.

Sumber daya

Praktik terbaik terkait:

- OPS02-BP02 Proses dan Prosedur memiliki pemilik teridentifikasi - Proses dan prosedur untuk mendukung sumber daya bergantung pada kepemilikan sumber daya.
- OPS02-BP04 Anggota tim tahu tanggung jawab mereka - Anggota tim harus memahami sumber daya apa yang mereka miliki.
- OPS02-BP05 Mekanisme tersedia untuk mengidentifikasi tanggung jawab dan kepemilikan - Kepemilikan harus dapat ditemukan menggunakan mekanisme seperti tag atau kontak akun.

Dokumen terkait:

- Manajemen Akun AWS - Memperbarui informasi kontak
- Aturan AWS Config - tag yang diperlukan
- AWS Organizations - Memperbarui kontak alternatif dalam organisasi Anda
- Laporan resmi Praktik Terbaik Pemberian Tag AWS

Contoh terkait:

- Aturan AWS Config - Amazon EC2 dengan tag yang diperlukan dan nilai yang valid

Layanan terkait:

- AWS Config
- AWS Organizations

OPS02-BP02 Proses dan Prosedur memiliki pemilik teridentifikasi

Pahami siapa pemegang kepemilikan atas definisi dari masing-masing proses dan prosedur, alasan prosedur dan proses tertentu digunakan, serta alasan adanya kepemilikan tersebut. Dengan memahami alasan untuk menggunakan proses dan prosedur tertentu, identifikasi peluang pengembangan yang dapat dilakukan.

Manfaat menerapkan praktik terbaik ini: Dengan memahami kepemilikan, Anda dapat mengidentifikasi siapa yang dapat menyetujui pengembangan, mengimplementasikan pengembangan tersebut, atau melakukan keduanya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Proses dan prosedur memiliki pemilik teridentifikasi yang bertanggung jawab atas definisinya: Dokumentasikan proses dan prosedur yang digunakan di lingkungan Anda, serta individu atau tim yang bertanggung jawab atas definisinya.
  - Identifikasikan proses dan prosedur: Identifikasi aktivitas operasi yang dijalankan untuk mendukung beban kerja Anda. Dokumentasikan aktivitas ini di lokasi yang mudah ditemukan.
  - Tentukan siapa yang memiliki definisi proses atau prosedur: Identifikasi secara khusus individu atau tim yang bertanggung jawab atas spesifikasi aktivitas. Mereka bertanggung jawab untuk memastikan aktivitas dapat dijalankan dengan sukses oleh anggota tim yang memiliki keterampilan memadai dengan izin, akses, serta alat yang sesuai. Jika terdapat masalah saat menjalankan aktivitas tersebut, anggota tim yang menjalankannya bertanggung jawab untuk memberikan tanggapan mendetail yang diperlukan agar aktivitas tersebut dapat ditingkatkan.
  - Dokumentasikan kepemilikan di metadata artefak aktivitas: Prosedur yang diotomatiskan dalam layanan seperti AWS Systems Manager, melalui dokumen, dan AWS Lambda, sebagai fungsi, mendukung dokumentasi informasi metadata sebagai tanda. Dokumentasikan kepemilikan sumber daya menggunakan grup sumber daya atau tanda, yang menentukan informasi kontak dan kepemilikan. Gunakan AWS Organizations untuk membuat kebijakan penandaan serta memastikan dokumentasi informasi kontak serta kepemilikan.

OPS02-BP03 Aktivitas operasi memiliki pemilik teridentifikasi yang bertanggung jawab atas kinerjanya

Pahami siapa yang bertanggung jawab untuk menjalankan aktivitas tertentu terhadap beban kerja yang ditentukan serta alasan adanya tanggung jawab tersebut. Memahami siapa yang bertanggung jawab untuk menjalankan aktivitas dapat memberikan informasi tentang siapa yang akan melakukan aktivitas tersebut, memvalidasi hasilnya, serta memberikan umpan balik kepada pemilik aktivitas.

Manfaat menerapkan praktik terbaik ini: Memahami siapa yang bertanggung jawab untuk menjalankan sebuah aktivitas dapat memberikan informasi tentang siapa yang harus diberi tahu saat diperlukan tindakan dan siapa yang akan melakukan tindakan, memvalidasi hasilnya, serta memberikan umpan balik kepada pemilik aktivitas tersebut.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Aktivitas operasi memiliki pemilik teridentifikasi yang bertanggung jawab atas kinerjanya: Dokumentasikan tanggung jawab untuk menjalankan proses dan prosedur yang digunakan di lingkungan Anda.
  - Identifikasikan proses dan prosedur: Identifikasi aktivitas operasi yang dijalankan untuk mendukung beban kerja Anda. Dokumentasikan aktivitas ini di lokasi yang mudah ditemukan.
  - Tentukan siapa yang bertanggung jawab untuk menjalankan setiap aktivitas: Identifikasikan tim yang bertanggung jawab atas aktivitas. Pastikan mereka memiliki detail aktivitas, keterampilan yang diperlukan dan izin yang tepat, akses, dan alat yang sesuai untuk menjalankan aktivitas. Mereka harus memahami kapan aktivitas tersebut harus dijalankan (misalnya, sesuai peristiwa atau jadwal). Buat informasi ini dapat ditemukan sehingga para anggota organisasi Anda dapat mengidentifikasi siapa yang perlu mereka hubungi, tim atau individu, untuk kebutuhan tertentu.

OPS02-BP04 Anggota tim tahu tanggung jawab mereka

Memahami tanggung jawab peran Anda dan bagaimana Anda berkontribusi terhadap hasil bisnis memberitahukan penentuan prioritas tugas Anda dan mengapa peran Anda itu penting. Ini memampukan anggota tim untuk mengenali kebutuhan dan merespons dengan tepat.

Manfaat dari menjalankan praktik terbaik ini: Memahami tanggung jawab Anda memberikan informasi untuk keputusan yang Anda ambil, tindakan yang Anda ambil, dan penyerahan aktivitas Anda ke pemiliknya yang benar.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

- Pastikan anggota tim memahami peran dan tanggung jawab mereka: Identifikasi peran dan tanggung jawab anggota tim dan pastikan mereka memahami yang diharapkan dari peran mereka. Buat informasi ini dapat ditemukan sehingga para anggota organisasi Anda dapat mengidentifikasi siapa yang perlu mereka hubungi, tim atau individu, untuk kebutuhan tertentu.

OPS02-BP05 Mekanisme tersedia untuk mengidentifikasi tanggung jawab dan kepemilikan

Apabila tidak ada individu atau tim yang diidentifikasi, terdapat jalur eskalasi yang ditetapkan ke seseorang yang memiliki wewenang untuk menetapkan kepemilikan atau rencana untuk penanganan kebutuhan tersebut.

Manfaat menjalankan praktik terbaik ini: Dengan memahami siapa yang memiliki tanggung jawab atau kepemilikan, Anda dapat menghubungi tim atau anggota tim yang tepat untuk melakukan permintaan atau mengalihkan tugas. Dengan adanya orang yang diidentifikasi yang memiliki wewenang untuk menetapkan tanggung jawab atau kepemilikan atau rencana untuk menangani kebutuhan, risiko tidak adanya aksi dan tidak tertanganinya kebutuhan dapat diminimalkan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

- Mekanisme tersedia untuk mengidentifikasi tanggung jawab dan kepemilikan: Sediakan mekanisme yang dapat diakses bagi anggota organisasi untuk menemukan dan mengidentifikasi kepemilikan dan tanggung jawab. Mekanisme ini memungkinkan mereka untuk mengidentifikasi siapa yang harus dihubungi, baik tim maupun individu, untuk kebutuhan tertentu.

OPS02-BP06 Mekanisme tersedia untuk meminta penambahan, perubahan, dan pengecualian

Anda dapat mengajukan permintaan kepada pemilik proses, prosedur, dan sumber daya. Permintaan mencakup penambahan, perubahan, dan pengecualian. Permintaan ini diajukan melalui proses manajemen perubahan. Buat keputusan yang matang untuk menyetujui permintaan apabila memungkinkan dan dianggap tepat setelah dilakukan evaluasi manfaat dan risiko.

Hasil yang diinginkan:

- Anda dapat mengajukan permintaan untuk mengubah proses, prosedur, dan sumber daya berdasarkan kepemilikan yang ditetapkan.
- Perubahan dibuat dengan penuh pertimbangan, dengan memikirkan manfaat dan risikonya.

Antipola umum:

- Anda harus memperbarui cara Anda melakukan deployment aplikasi Anda, tetapi perubahan proses deployment tidak dapat diminta dari tim operasi.

- Rencana pemulihan bencana harus diperbarui, tetapi tidak ada pemilik yang teridentifikasi untuk dimintai perubahan.

Manfaat menjalankan praktik terbaik ini:

- Proses, prosedur, dan sumber daya dapat berubah seiring perubahan persyaratan.
- Pemilik dapat mengambil keputusan yang bijaksana ketika harus membuat perubahan.
- Perubahan dibuat dengan cara yang penuh pertimbangan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Untuk mengimplementasikan praktik terbaik ini, Anda harus dapat meminta perubahan proses, prosedur, dan sumber daya. Proses manajemen perubahan bisa ringan. Dokumentasikan proses manajemen perubahan.

Contoh pelanggan

AnyCompany Retail menggunakan matriks penetapan tanggung jawab (RACI) untuk mengidentifikasi siapa yang memiliki perubahan untuk proses, prosedur, dan sumber daya. Mereka memiliki proses manajemen perubahan terdokumentasi yang ringan dan mudah diikuti. Menggunakan matriks RACI dan proses, siapa pun dapat menyampaikan permintaan perubahan.

Langkah implementasi

1. Identifikasi proses, prosedur, dan sumber daya untuk beban kerja Anda dan pemilik untuk masing-masing. Dokumentasikan dalam sistem manajemen pengetahuan Anda.
   a. Jika Anda belum mengimplementasikan OPS02-BP01 Sumber daya memiliki pemilik teridentifikasi, OPS02-BP02 Proses dan Prosedur memiliki pemilik teridentifikasi, atau OPS02-BP03 Aktivitas operasi memiliki pemilik teridentifikasi yang bertanggung jawab atas kinerjanya, mulai dengan itu terlebih dahulu.
2. Bekerjasamalah dengan pemangku kepentingan di organisasi Anda untuk mengembangkan proses manajemen perubahan. Proses harus meliputi penambahan, perubahan, dan pengecualian untuk sumber daya, proses, dan prosedur.
   a. Anda dapat menggunakan AWS Systems Manager Manajer Perubahan sebagai platform manajemen perubahan untuk sumber daya beban kerja.
3. Dokumentasikan proses manajemen perubahan dalam sistem manajemen pengetahuan Anda.

Tingkat upaya untuk rencana implementasi: Sedang. Mengembangkan proses manajemen perubahan memerlukan penyelarasan dengan beberapa pemangku kepentingan di seluruh organisasi Anda.

Sumber daya

Praktik terbaik terkait:

- OPS02-BP01 Sumber daya memiliki pemilik teridentifikasi - Sumber daya memerlukan pengidentifikasian pemilik sebelum Anda membangun proses manajemen perubahan.
- OPS02-BP02 Proses dan Prosedur memiliki pemilik teridentifikasi - Proses memerlukan pengidentifikasian pemilik sebelum Anda membangun proses manajemen perubahan.
- OPS02-BP03 Aktivitas operasi memiliki pemilik teridentifikasi yang bertanggung jawab atas kinerjanya - Aktivitas operasi memerlukan pengidentifikasian pemilik sebelum Anda membangun proses manajemen perubahan.

Dokumen terkait:

- Panduan Preskriptif AWS - Pedoman mendasar untuk migrasi besar AWS: Membuat matriks RACI
- Laporan Resmi Manajemen Perubahan di Cloud

Layanan terkait:

- Manajer Perubahan AWS Systems Manager

OPS02-BP07 Tanggung jawab antara tim telah dinegosiasi atau ditetapkan sebelumnya

Miliki perjanjian yang telah ditetapkan atau dinegosiasi antara tim yang menjelaskan bagaimana mereka akan bekerja sama dan saling mendukung satu sama lain (contohnya, waktu respons, tujuan tingkat layanan, atau perjanjian tingkat layanan). Saluran komunikasi antar-tim didokumentasi. Memahami dampak dari pekerjaan tim atas hasil bisnis, dan hasil dari tim lain dan organisasi memberitahukan penentuan prioritas tugas mereka dan membantu mereka merespons dengan tepat.

Ketika tanggung jawab dan kepemilikan tidak ditetapkan atau tidak diketahui, Anda menanggung risiko tidak menangani aktivitas yang diperlukan secara tepat waktu serta risiko munculnya upaya yang berulang dan kemungkinan bertentangan untuk menangani kebutuhan-kebutuhan tersebut.

Hasil yang diinginkan:

- Perjanjian bekerja atau mendukung antar-tim disetujui dan didokumentasi.
- Tim yang mendukung atau bekerja dengan satu sama lain memiliki ekspektasi respons dan saluran komunikasi yang telah ditetapkan sebelumnya.

Antipola umum:

- Masalah terjadi dalam produksi dan dua tim terpisah mulai menyelesaikan masalahnya sendiri-sendiri. Upaya terpisah mereka memperpanjang masa penghentian produksi.
- Tim operasi membutuhkan bantuan dari tim pengembangan, tetapi tidak ada waktu respons yang disepakati. Permintaannya tetap tinggal di timbunan yang belum dikerjakan.

Manfaat menjalankan praktik terbaik ini:

- Tim mengetahui cara berinteraksi dan mendukung satu sama lain.
- Ekspektasi untuk tingkat responsivitas diketahui.
- Saluran komunikasi ditetapkan dengan jelas.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Mengimplementasikan praktik terbaik ini berarti tidak ada ambiguitas tentang bagaimana tim bekerja dengan satu sama lain. Perjanjian resmi mengodekan bagaimana tim bekerja sama atau mendukung satu sama lain. Saluran komunikasi antar-tim didokumentasi.

Contoh pelanggan

Tim SRE AnyCompany Retail memiliki perjanjian tingkat layanan dengan tim pengembangan mereka. Setiap kali tim pengembangan mengajukan permintaan dalam sistem tiket mereka, mereka dapat mengantisipasi bahwa respons akan diterima dalam waktu lima belas menit. Jika ada penghentian kerja di lokasi, tim SRE akan memimpin investigasinya dengan dukungan tim pengembangan.

Langkah implementasi

1. Melalui kerja sama dengan pemangku kepentingan di seluruh organisasi Anda, adakan perjanjian antara tim berdasarkan proses dan prosedur.
   a. Jika proses atau prosedur dibagi antara dua tim, kembangkan runbook tentang cara tim akan bekerja sama.

   b. Jika ada ketergantungan antara tim, setujui SLA respons untuk permintaan.
2. Dokumentasikan tanggung jawab dalam sistem manajemen pengetahuan Anda.

Tingkat upaya untuk rencana implementasi: Sedang. Jika belum ada perjanjian antara tim, mungkin akan diperlukan upaya agar para pemangku kepentingan di seluruh organisasi Anda bisa sepakat.

Sumber daya

Praktik terbaik terkait:

- OPS02-BP02 Proses dan Prosedur memiliki pemilik teridentifikasi - Kepemilikan proses harus diidentifikasi sebelum perjanjian diadakan antara tim.
- OPS02-BP03 Aktivitas operasi memiliki pemilik teridentifikasi yang bertanggung jawab atas kinerjanya - Kepemilikan aktivitas operasi harus diidentifikasi sebelum perjanjian diadakan antara tim.

Dokumen terkait:

- Wawasan Eksekutif AWS - Memberdayakan Inovasi dengan Tim Dua Piza
- Pengantar DevOps di AWS - Tim Dua Piza

OPS 3. Bagaimana budaya organisasi Anda mendukung hasil bisnis Anda?

Berikan dukungan kepada anggota tim Anda sehingga mereka dapat menjadi lebih efektif dalam mengambil tindakan dan mendukung hasil bisnis Anda.

Praktik terbaik

- OPS03-BP01 Sponsor Eksekutif
- OPS03-BP02 Anggota tim diberdayakan untuk bertindak ketika terdapat risiko pada hasil
- OPS03-BP03 Imbauan eskalasi
- OPS03-BP04 Komunikasi yang tepat waktu, jelas, dan dapat ditindaklanjuti
- OPS03-BP05 Mendorong eksperimen
- OPS03-BP06 Mendorong dan mendukung anggota tim untuk mempertahankan dan mengembangkan tingkat keterampilan mereka
- OPS03-BP07 Bekali tim dengan sumber daya dengan sesuai

- OPS03-BP08 Pendapat yang beragam didukung dan dicari di dalam dan lintas tim

OPS03-BP01 Sponsor Eksekutif

Pimpinan senior dengan jelas menetapkan ekspektasi untuk organisasi dan mengevaluasi kesuksesan. Pimpinan senior adalah sponsor, pendukung, dan pendorong untuk pengadopsian praktik terbaik serta perkembangan organisasi.

Manfaat menerapkan praktik terbaik ini: Pimpinan yang terlibat, ekspektasi yang dikomunikasikan dengan jelas, serta tujuan bersama, dapat memastikan anggota tim mengetahui apa yang diharapkan dari mereka. Dengan mengevaluasi kesuksesan, penghalang kesuksesan dapat diidentifikasi, sehingga dapat diatasi melalui intervensi oleh pendukung sponsor atau delegasinya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Sponsor Eksekutif: Pimpinan senior dengan jelas menetapkan ekspektasi untuk organisasi dan mengevaluasi kesuksesan. Pimpinan senior adalah sponsor, pendukung, dan pendorong untuk pengadopsian praktik terbaik serta perkembangan organisasi.
  - Tetapkan ekspektasi: Tentukan dan publikasikan tujuan untuk organisasi Anda, termasuk cara mengukur tujuan tersebut.
  - Lacak capaian tujuan: Ukur capaian bertahap dari tujuan secara rutin serta bagikan hasilnya, agar tindakan yang sesuai dapat segera dilakukan jika hasil sedang dipertaruhkan.
  - Sediakan sumber daya yang diperlukan untuk mencapai target Anda: Lakukan peninjauan secara rutin apakah sumber daya masih sesuai, atau berikan sumber daya tambahan jika diperlukan, berdasarkan pada: informasi baru, perubahan target, tanggung jawab, atau lingkungan bisnis Anda.
  - Dukung tim Anda: Tetap berinteraksi dengan tim Anda sehingga Anda memahami bagaimana kondisi mereka dan mengetahui jika ada faktor eksternal yang memengaruhi mereka. Ketika ada faktor eksternal yang memengaruhi kinerja mereka, evaluasi kembali tujuan dan sesuaikan target sebagaimana mestinya. Identifikasikan masalah yang menghambat kemajuan tim Anda. Bertindaklah atas nama tim Anda untuk membantu mengatasi masalah dan menghilangkan beban yang tidak perlu.
  - Jadilah penggerak untuk pengadopsian praktik terbaik: Identifikasikan praktik terbaik yang terbukti memberikan manfaat terukur serta beri pengakuan kepada pencipta dan penggunanya. Dukung adopsi lebih lanjut untuk memperbesar manfaat yang dapat dicapai.

- Jadilah penggerak perkembangan tim Anda: Ciptakan budaya peningkatan berkelanjutan. Dukung peningkatan dan perkembangan yang dicapai baik oleh perorangan maupun organisasi. Berikan target jangka panjang yang harus dikejar dan mengharuskan pencapaian bertahap dari waktu ke waktu. Sesuaikan visi ini untuk menyempurnakan kebutuhan, tujuan bisnis, serta lingkungan bisnis Anda seiring dengan perubahannya.

OPS03-BP02 Anggota tim diberdayakan untuk bertindak ketika terdapat risiko pada hasil

Pemilik beban kerja telah menetapkan panduan dan cakupan yang memberdayakan anggota tim untuk merespons ketika terdapat risiko pada hasil. Mekanisme eskalasi digunakan untuk mendapatkan petunjuk ketika peristiwa berada di luar cakupan yang ditetapkan.

Manfaat menerapkan praktik terbaik ini: Dengan menguji dan memvalidasi perubahan sejak dini, Anda dapat mengatasi masalah dengan biaya minim dan membatasi dampak terhadap pelanggan. Dengan menguji sebelum deployment, Anda meminimalkan munculnya kesalahan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

- Anggota tim diberdayakan untuk bertindak ketika terdapat risiko pada hasil: Bekali anggota tim Anda dengan izin, alat, dan peluang untuk mempraktikkan keterampilan yang diperlukan untuk merespons secara efektif.
  - Beri anggota tim Anda peluang untuk mempraktikkan keterampilan yang diperlukan untuk merespons: Sediakan alternatif lingkungan aman di mana proses dan prosedur dapat diuji dan digunakan untuk latihan dengan aman. Lakukan aktivitas permainan untuk memberi kesempatan pada anggota tim untuk mendapatkan pengalaman merespons insiden dunia nyata dalam lingkungan simulasi yang aman.
  - Tetapkan dan kenali wewenang anggota tim untuk bertindak: Tetapkan secara khusus wewenang anggota tim untuk bertindak dengan memberikan izin dan akses ke beban kerja dan komponen yang mereka dukung. Ketahui bahwa mereka diberdayakan untuk bertindak ketika terdapat risiko pada hasil.

OPS03-BP03 Imbauan eskalasi

Anggota tim memiliki mekanisme dan diimbau untuk mengeskalasikan masalah ke pengambil keputusan dan pemangku kepentingan jika mereka yakin terdapat risiko pada hasil. Eskalasi

harus dilakukan sejak dini dan secara sering agar risiko dapat diidentifikasi, dan dicegah sebelum menyebabkan insiden.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

- Dorong eskalasi sejak dini dan secara sering: Akui pada tingkat organisasi bahwa eskalasi yang dilakukan sejak dini dan secara sering merupakan praktik terbaik. Akui dan terima pada tingkat organisasi bahwa eskalasi mungkin saja terbukti tidak berdasar, tetapi lebih baik mengambil kesempatan untuk mencegah insiden daripada melewatkan kesempatan untuk melakukan eskalasi.
  - Miliki mekanisme untuk eskalasi: Miliki prosedur terdokumentasi yang menetapkan kapan dan bagaimana eskalasi harus dilakukan. Dokumentasikan sekelompok personel dengan wewenang berjenjang untuk mengambil tindakan atau menyetujui tindakan beserta informasi kontak mereka. Eskalasi harus berlanjut sampai anggota tim yakin bahwa risiko telah dialihkan ke personel yang mampu mengatasinya, atau mereka telah menghubungi personel yang memiliki hak atas risiko dan tanggung jawab atas operasi beban kerja. Personel tersebutlah yang memiliki semua keputusan akhir terkait beban kerja mereka. Eskalasi harus menyertakan sifat risiko, tingkat kekritisan beban kerja, orang yang terkena dampak, apa dampaknya, dan urgensinya, yakni kapan dampak diperkirakan akan dialami.
  - Lindungi karyawan yang melakukan eskalasi: Miliki kebijakan yang melindungi anggota tim dari tindakan pembalasan jika mereka melakukan eskalasi di sekitar pengambil keputusan atau pemangku kepentingan yang tidak responsif. Terapkan mekanisme untuk mengidentifikasi apakah hal ini terjadi dan beri respons yang tepat.

OPS03-BP04 Komunikasi yang tepat waktu, jelas, dan dapat ditindaklanjuti

Mekanisme dihadirkan dan digunakan untuk memberikan pengingat secara tepat waktu kepada anggota tim tentang risiko yang diketahui dan peristiwa yang direncanakan. Konteks, detail, dan waktu (ketika memungkinkan) yang diperlukan diberikan untuk membantu menentukan apakah memerlukan tindakan, tindakan apa yang diperlukan, serta untuk melakukan tindakan tepat waktu. Misalnya, memberikan peringatan kerentanan perangkat lunak agar patching dapat dipercepat, atau memberikan peringatan tentang promosi penjualan yang direncanakan sehingga pemberhentian perubahan dapat diimplementasikan untuk menghindari gangguan layanan. Peristiwa yang direncanakan dapat dicatat dalam kalender perubahan atau jadwal pemeliharaan sehingga anggota tim dapat mengidentifikasi aktivitas yang tertunda.

Hasil yang diinginkan:

- Komunikasi memberikan ekspektasi konteks, detail, dan waktu.
- Anggota tim memiliki pemahaman yang jelas tentang kapan dan bagaimana mereka harus bertindak untuk merespons komunikasi.
- Manfaatkan kalender perubahan untuk memberikan pemberitahuan ekspektasi perubahan.

Antipola umum:

- Peringatan positif palsu muncul beberapa kali seminggu. Anda mematikan suara pemberitahuan setiap kali peringatan muncul.
- Anda diminta untuk membuat perubahan pada grup keamanan Anda, tetapi tidak diberi ekspektasi kapan itu harus terjadi.
- Anda terus-menerus menerima pemberitahuan dalam obrolan ketika sistem menaikkan skala tetapi tidak diperlukan tindakan. Anda menghindari saluran obrolan dan melewatkan pemberitahuan penting.
- Perubahan dibuat pada produksi tanpa pemberitahuan kepada tim operasi. Perubahan tersebut memicu peringatan dan tim yang jaga diaktifkan.

Manfaat menjalankan praktik terbaik ini:

- Organisasi Anda menghindari kejemuan karena peringatan.
- Anggota tim dapat bertindak dengan ekspektasi dan konteks yang diperlukan.
- Perubahan dapat dibuat selama jendela perubahan, yang mengurangi risiko.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Untuk mengimplementasikan praktik terbaik ini, Anda harus bekerja sama dengan pemangku kepentingan di seluruh organisasi untuk menyepakati standar komunikasi. Publikasikan standar tersebut ke organisasi Anda. Identifikasi dan singkirkan peringatan yang positif palsu atau selalu muncul. Gunakan kalender perubahan sehingga anggota tim tahu kapan tindakan dapat diambil dan aktivitas apa yang masih perlu dikerjakan. Verifikasi apakah komunikasi menghasilkan tindakan yang jelas dengan konteks yang perlu.

Contoh pelanggan

AnyCompany Retail menggunakan obrolan sebagai media komunikasi utama mereka. Peringatan dan informasi lainnya memenuhi saluran tertentu. Ketika seseorang harus bertindak, hasil yang diinginkan dinyatakan dengan jelas, dan dalam banyak kasus, mereka diberi runbook atau playbook untuk digunakan. Mereka menggunakan kalender perubahan untuk menjadwalkan perubahan besar pada sistem produksi.

Langkah implementasi

1. Analisis peringatan Anda untuk mengetahui adanya peringatan positif palsu atau peringatan yang terus-menerus terpicu. Singkirkan atau ubah peringatan tersebut sehingga akan muncul apabila intervensi manusia diperlukan. Jika peringatan muncul, berikan runbook atau playbook.
    a. Anda dapat menggunakan Dokumen AWS Systems Manager guna membangun playbook dan runbook untuk peringatan.
2. 2Mekanisme diterapkan untuk memberikan pemberitahuan risiko atau peristiwa terencana dengan cara yang jelas dan dapat ditindaklanjuti, melalui peringatan yang memadai untuk memberi respons yang sesuai. Gunakan daftar email atau saluran obrolan untuk mengirimkan pemberitahuan sebelum acara yang sudah direncanakan.
    a. Amazon Q Developer in chat applications dapat digunakan untuk mengirimkan peringatan dan respons terhadap acara dalam platform pengiriman pesan organisasi Anda.
3. Berikan sumber informasi yang dapat diakses di mana acara yang sudah direncanakan dapat ditemukan. Beri pemberitahuan tentang peristiwa yang direncanakan dari sistem yang sama.
    a. Kalender Perubahan AWS Systems Manager dapat digunakan untuk membuat jendela perubahan ketika perubahan dapat terjadi. Hal ini memberi anggota tim pemberitahuan kapan mereka dapat membuat perubahan dengan aman.
4. Pantau pemberitahuan kerentanan dan informasi patch untuk memahami kerentanan risiko tinggi dan potensial yang berkaitan dengan komponen beban kerja Anda. Berikan pemberitahuan kepada anggota tim agar mereka dapat bertindak.
    a. Anda dapat berlangganan Buletin Keamanan AWS untuk menerima pemberitahuan tentang kerentanan di AWS.

Sumber daya

Praktik terbaik terkait:

- OPS07-BP03 Menggunakan runbook untuk menjalankan prosedur - Buat komunikasi dapat ditindaklanjuti dengan memberikan runbook ketika hasilnya diketahui.
- OPS07-BP04 Menggunakan buku panduan untuk menyelidiki masalah - Jika hasilnya tidak diketahui, playbook dapat membuat komunikasi dapat ditindaklanjuti.

Dokumen terkait:

- Buletin Keamanan AWS
- CVE Terbuka

Contoh terkait:

- Well-Architected Labs: Manajemen Inventaris dan Patch (Tingkat 100)

Layanan terkait:

- Amazon Q Developer in chat applications
- Kalender Perubahan AWS Systems Manager
- Dokumen AWS Systems Manager

OPS03-BP05 Mendorong eksperimen

Eksperimen adalah katalis untuk mengubah ide baru menjadi produk dan fitur. Eksperimen mempercepat proses pembelajaran dan membuat anggota tim terus tertarik dan terlibat. Anggota tim didorong untuk sering bereksperimen guna mendorong inovasi. Meskipun hasil yang tidak diinginkan terjadi, ada nilai dalam memiliki pengetahuan tentang apa yang sebaiknya tidak dilakukan. Anggota tim tidak dihukum untuk eksperimen yang berhasil dengan hasil yang tidak diinginkan.

Hasil yang diinginkan:

- Organisasi Anda mendorong eksperimen untuk mendukung inovasi.
- Eksperimen digunakan sebagai peluang untuk belajar.

Antipola umum:

- Anda ingin menjalankan pengujian A/B tetapi tidak ada mekanisme untuk menjalankan eksperimen tersebut. Anda melakukan deployment perubahan UI tanpa kemampuan untuk mengujinya. Tindakan tersebut mengakibatkan pengalaman pelanggan yang negatif.
- Perusahaan Anda hanya memiliki lingkungan produksi dan staging. Tidak ada lingkungan sandbox untuk bereksperimen dengan fitur atau produk baru sehingga Anda harus bereksperimen di dalam lingkungan produksi.

Manfaat menjalankan praktik terbaik ini:

- Eksperimen mendorong inovasi.
- Anda dapat bereaksi lebih cepat terhadap umpan balik dari pengguna melalui eksperimen.
- Organisasi Anda mengembangkan budaya belajar.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Eksperimen harus dijalankan dengan cara yang aman. Manfaatkan beberapa lingkungan untuk bereksperimen tanpa membahayakan sumber daya produksi. Gunakan pengujian A/B dan tanda fitur untuk menguji eksperimen. Berikan kepada anggota tim kemampuan untuk melakukan eksperimen dalam lingkungan sandbox.

Contoh pelanggan

AnyCompany Retail mendorong eksperimen. Anggota tim dapat menggunakan 20% dari minggu kerja mereka untuk bereksperimen atau mempelajari teknologi baru. Mereka memiliki lingkungan sandbox di mana mereka dapat berinovasi. Pengujian A/B digunakan untuk fitur baru guna memvalidasinya dengan umpan balik nyata pengguna.

Langkah implementasi

1. Bekerjasamalah dengan pimpinan di seluruh organisasi Anda untuk mendukung eksperimen. Anggota tim harus didorong untuk melakukan eksperimen dengan cara yang aman.
2. Berikan kepada anggota tim Anda lingkungan di mana mereka dapat bereksperimen dengan aman. Mereka harus memiliki akses ke lingkungan yang seperti produksi.
   a. Anda dapat menggunakan Akun AWS terpisah untuk membuat lingkungan sandbox untuk eksperimen. AWS Control Tower dapat digunakan untuk menyediakan akun-akun ini.

3. Gunakan tanda fitur dan pengujian A/B untuk bereksperimen dengan aman dan kumpulkan umpan balik pengguna.
   a. Tanda Fitur AWS AppConfig memberikan kemampuan untuk membuat tanda fitur.
   b. Evidently Amazon CloudWatch dapat digunakan untuk menjalankan pengujian A/B selama deployment terbatas.
   c. Anda dapat menggunakan versi AWS Lambda untuk melakukan deployment versi baru fungsi untuk pengujian beta.

Tingkat upaya untuk rencana implementasi: Tinggi. Memberikan kepada anggota tim lingkungan untuk bereksperimen dan cara yang aman untuk melakukan eksperimen dapat memerlukan investasi besar. Anda mungkin juga harus memodifikasi kode aplikasi untuk menggunakan tanda fitur atau mendukung pengujian A/B.

Sumber daya

Praktik terbaik terkait:

- OPS11-BP02 Menjalankan analisis setelah insiden - Belajar dari insiden merupakan pendorong penting untuk inovasi bersama dengan eksperimen.
- OPS11-BP03 Mengimplementasikan loop umpan balik - Siklus umpan balik merupakan bagian penting dari eksperimen.

Dokumen terkait:

- Gambaran Mendalam tentang Budaya Amazon: Eksperimen, Kegagalan, dan Obsesi
- Praktik terbaik untuk membuat dan mengelola akun sandbox di AWS
- Buat Budaya Eksperimen yang Dimampukan oleh Cloud
- Memampukan eksperimen dan inovasi di cloud di SulAmérica Seguros
- Bereksperimen Lebih Sering, Gagal Lebih Jarang
- Mengatur Lingkungan AWS Anda Menggunakan Beberapa Akun - OU Sandbox
- Menggunakan Tanda Fitur AWS AppConfig

Video terkait:

- AWS On Air dengan Evidently Amazon CloudWatch | Acara AWS

- On Air San Fran Summit 2022 AWS dengan Integrasi Tanda Fitur AWS AppConfig dengan Jira
- AWS re:Invent 2022 - Deployment bukan rilis: Kontrol peluncuran Anda dengan tanda fitur (BOA305-R)
- Secara Terprogram Buat Akun AWS dengan AWS Control Tower
- Menyiapkan Lingkungan AWS Multiakun yang Menggunakan Praktik Terbaik untuk AWS Organizations

Contoh terkait:

- Sandbox Inovasi AWS
- Personalisasi Menyeluruh 101 untuk E-Commerce

Layanan terkait:

- Amazon CloudWatch Evidently
- AWS AppConfig
- AWS Control Tower

OPS03-BP06 Mendorong dan mendukung anggota tim untuk mempertahankan dan mengembangkan tingkat keterampilan mereka

Tim harus mengembangkan tingkat keterampilan mereka untuk mengadopsi perkembangan teknologi, serta untuk mengimbangi perubahan permintaan dan tanggung jawab dalam mendukung beban kerja Anda. Perkembangan keterampilan menggunakan teknologi dapat menjadi sumber kepuasan tim dan mendorong inovasi. Dukung anggota tim Anda untuk mendapatkan dan mempertahankan sertifikasi industri yang memvalidasi dan mengakui perkembangan keterampilan mereka. Terapkan pelatihan silang untuk mendorong transfer pengetahuan dan meminimalkan dampak signifikan yang terjadi karena kehilangan anggota tim berpengalaman yang memiliki keterampilan dan pengetahuan terkait lembaga. Berikan waktu khusus yang terstruktur untuk pembelajaran.

AWS menyediakan sumber daya, termasuk Pusat Sumber Daya untuk Memulai AWS, Blog AWS, AWS Online Tech Talks, Acara dan Webinar AWS, serta Lab AWS Well-Architected, yang menyediakan panduan, contoh, dan ringkasan mendetail untuk mengedukasi tim Anda.

AWS juga membagikan pola dan praktik terbaik yang telah kami pelajari melalui operasi AWS di Amazon Builders' Library serta berbagai macam materi edukasi bermanfaat lainnya dari Blog AWS dan Official AWS Podcast.

Anda harus memanfaatkan sumber daya edukasi yang disediakan oleh AWS seperti lab Well-Architected, AWS Dukungan (Pusat Pengetahuan AWS, Forum Diskusi AWS, dan Pusat AWS Dukungan) dan Dokumentasi AWS untuk mengedukasi tim Anda. Hubungi AWS Dukungan melalui Pusat AWS Dukungan jika Anda memiliki pertanyaan seputar AWS.

AWS Training and Certification menyediakan beberapa pelatihan gratis melalui kursus digital mandiri tentang dasar-dasar AWS. Anda juga dapat mengikuti pelatihan yang dipandu instruktur untuk mendukung perkembangan keterampilan AWS tim Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Dorong dan dukung anggota tim untuk mempertahankan dan mengembangkan tingkat keterampilan mereka: Diperlukan edukasi yang berkelanjutan untuk mengadopsi teknologi baru, mendorong inovasi, dan mengimbangi perubahan permintaan serta tanggung jawab dalam mendukung beban kerja Anda.
  - Sediakan sumber daya untuk kepentingan edukasi: Sediakan waktu khusus yang terstruktur, akses ke materi pelatihan, sumber daya lab, dan dukung partisipasi untuk mengikuti konferensi dan organisasi profesional yang memberikan kesempatan untuk belajar dari pendidik dan rekan. Berikan akses bagi anggota tim junior untuk belajar dari anggota tim senior atau biarkan tim junior meniru pekerjaan tim senior serta melihat metode dan keterampilan mereka. Dorong pembelajaran tentang konten yang tidak terkait langsung dengan pekerjaan agar mereka memiliki pandangan yang lebih luas.
  - Edukasi tim dan interaksi antartim: Buat rencana untuk kebutuhan anggota tim terkait pembelajaran berkelanjutan. Berikan kesempatan kepada anggota tim untuk bergabung dengan tim lain (sementara atau seterusnya) guna berbagi keterampilan dan praktik terbaik yang bermanfaat bagi organisasi Anda.
  - Dukung untuk mendapatkan dan mempertahankan sertifikasi industri: Dukung anggota tim Anda untuk mendapatkan dan mempertahankan sertifikasi industri yang memvalidasi kemampuan yang telah mereka pelajari, serta akui pencapaian mereka.

Sumber daya

Dokumen terkait:

- Pusat Sumber Daya untuk Memulai AWS
- Blog AWS
- Kepatuhan AWS Cloud
- Forum Diskusi AWS
- Dokumentasi AWS
- AWS Online Tech Talks
- Acara dan Webinar AWS
- Pusat Pengetahuan AWS
- AWS Dukungan
- AWS Training and Certification
- Lab AWS Well-Architected,
- Amazon Builders' Library
- Official AWS Podcast.

OPS03-BP07 Bekali tim dengan sumber daya dengan sesuai

Pertahankan kapasitas anggota tim, serta sediakan alat dan sumber daya untuk mendukung kebutuhan beban kerja Anda. Pemberian tugas yang terlalu banyak kepada anggota tim meningkatkan risiko insiden yang diakibatkan oleh kesalahan manusia. Investasi alat dan sumber daya (misalnya, menyediakan otomatisasi untuk aktivitas yang sering dilakukan) dapat meningkatkan efektivitas tim, serta memungkinkan mereka untuk mendukung aktivitas tambahan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Bekali tim dengan sumber daya yang sesuai: Pastikan Anda memiliki pemahaman tentang keberhasilan tim Anda serta faktor yang berkontribusi dalam keberhasilan atau ketidakberhasilan mereka. Dukung tim dengan sumber daya yang sesuai.
  - Pahami kinerja tim: Ukur pencapaian hasil operasional dan pengembangan aset oleh tim Anda. Lacak perubahan pada output dan tingkat kesalahan dari waktu ke waktu. Berinteraksilah

dengan tim untuk memahami tantangan terkait pekerjaan yang memengaruhi mereka (misalnya, meningkatnya tanggung jawab, perubahan teknologi, kehilangan personel, atau peningkatan pelanggan yang didukung).

- Pahami dampak pada kinerja mereka: Tetap berinteraksi dengan tim Anda sehingga Anda memahami bagaimana keadaan mereka dan apakah ada faktor eksternal yang memengaruhi mereka. Ketika tim Anda terdampak oleh faktor eksternal, evaluasi kembali tujuan dan sesuaikan target sebagaimana mestinya. Identifikasi rintangan yang menghambat kemajuan tim Anda. Bertindaklah sebagai perwakilan tim Anda untuk membantu mengatasi rintangan dan menghapus beban yang tidak perlu.
- Sediakan sumber daya yang diperlukan tim untuk meraih keberhasilan: Tinjau secara teratur apakah sumber daya masih layak, apakah diperlukan sumber daya tambahan, dan buat penyesuaian yang tepat untuk mendukung tim.

OPS03-BP08 Pendapat yang beragam didukung dan dicari di dalam dan lintas tim

Manfaatkan keragaman lintas organisasi untuk mencari berbagai perspektif unik. Gunakan perspektif ini untuk meningkatkan inovasi, menantang asumsi Anda, dan mengurangi risiko bias konfirmasi. Kembangkan inklusi, keragaman, dan kemudahan akses dalam tim Anda untuk mendapatkan perspektif yang menguntungkan.

Budaya organisasi berdampak langsung pada retensi dan kepuasan kerja anggota tim. Dukung keterlibatan dan kemampuan anggota tim Anda untuk mendukung keberhasilan bisnis Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

- Cari pendapat dan perspektif yang beragam: Dorong kontribusi dari semua orang. Beri suara untuk kelompok yang kurang terwakili. Rotasikan peran dan tanggung jawab dalam rapat.
  - Perluas peran dan tanggung jawab: Sediakan kesempatan bagi anggota tim untuk mengambil peran yang mungkin jarang bisa mereka ambil. Mereka akan mendapatkan pengalaman dan perspektif dari peran tersebut, serta dari interaksi dengan anggota tim baru yang mungkin tidak akan berinteraksi dengan mereka di luar peran tersebut. Mereka akan membawa pengalaman dan perspektif mereka ke peran baru serta anggota tim yang berinteraksi dengan mereka. Begitu perspektif meningkat, kesempatan bisnis tambahan bisa muncul, atau kesempatan baru untuk peningkatan bisa teridentifikasi. Buat anggota tim bergantian dalam melakukan tugas umum yang biasanya dilakukan anggota lain untuk memahami tuntutan dan dampak melakukan tugas tersebut.

- Sediakan lingkungan yang aman dan ramah: Miliki kebijakan dan kontrol yang melindungi mental dan keselamatan fisik anggota tim dalam organisasi Anda. Anggota tim harus bisa berinteraksi tanpa rasa takut akan pembalasan. Ketika anggota tim merasa aman dan diterima, mereka mungkin menjadi lebih terlibat dan produktif. Makin beragam organisasi Anda, makin baik pemahaman Anda tentang orang-orang yang Anda dukung termasuk pelanggan Anda. Ketika anggota tim Anda merasa nyaman, merasa bebas untuk berbicara, dan yakin bahwa suara mereka akan didengar, mereka lebih berpeluang untuk membagikan wawasan berharga (misalnya, peluang pemasaran, kebutuhan aksesibilitas, segmen pasar yang belum terlayani, risiko yang tidak diketahui di lingkungan Anda).
- Dukung anggota tim untuk berpartisipasi penuh: Sediakan sumber daya yang diperlukan bagi karyawan Anda untuk berpartisipasi penuh pada semua aktivitas yang berkaitan dengan pekerjaan. Anggota tim yang menghadapi tantangan harian telah mengembangkan keterampilan untuk pekerjaan di sekitar mereka. Keterampilan yang dikembangkan secara khusus ini bisa memberi keuntungan yang signifikan bagi organisasi Anda. Mendukung anggota tim dengan akomodasi yang diperlukan akan meningkatkan keuntungan yang bisa Anda terima dari kontribusi mereka.

Persiapan

Pertanyaan

- OPS 4. Bagaimana Anda mengimplementasikan observabilitas dalam beban kerja Anda?
- OPS 5. Bagaimana cara mengurangi kecacatan, mempermudah perbaikan, dan meningkatkan aliran ke dalam produksi?
- OPS 6. Bagaimana cara memitigasi risiko deployment?
- OPS 7. Bagaimana cara mengetahui bahwa Anda siap untuk mendukung beban kerja?

OPS 4. Bagaimana Anda mengimplementasikan observabilitas dalam beban kerja Anda?

Implementasikan observabilitas dalam beban kerja Anda sehingga Anda dapat memahami statusnya dan membuat keputusan berbasis data berdasarkan persyaratan bisnis.

Praktik terbaik

- OPS04-BP01 Identifikasikan indikator performa utama
- OPS04-BP02 Mengimplementasikan telemetri aplikasi

- OPS04-BP03 Mengimplementasikan telemetri pengalaman pengguna
- OPS04-BP04 Mengimplementasikan telemetri dependensi
- OPS04-BP05 Mengimplementasikan penelusuran terdistribusi

OPS04-BP01 Identifikasikan indikator performa utama

Untuk mengimplementasikan observabilitas dalam beban kerja, Anda memulainya dengan memahami statusnya dan mengambil keputusan berbasis data berdasarkan persyaratan bisnis. Salah satu cara paling efektif untuk memastikan keselarasan antara kegiatan pemantauan dan tujuan bisnis adalah dengan menentukan serta memantau indikator kinerja utama (KPI).

Hasil yang diinginkan: Praktik observabilitas yang efisien yang sangat selaras dengan tujuan bisnis, sehingga memastikan upaya pemantauan selalu memenuhi hasil bisnis yang nyata.

Antipola umum:

- KPI yang tidak ditentukan: Bekerja tanpa KPI yang jelas dapat menyebabkan terlalu banyak atau terlalu sedikit pemantauan, sehingga sinyal-sinyal vital menjadi terlewatkan.
- KPI statis: Tidak meninjau atau menyempurnakan KPI seiring perkembangan beban kerja atau tujuan bisnis.
- Ketidakselarasan: Berfokus pada metrik teknis yang tidak berkorelasi langsung dengan hasil bisnis atau yang lebih sulit untuk berkorelasi dengan masalah dunia nyata.

Manfaat menjalankan praktik terbaik ini:

- Kemudahan identifikasi masalah: KPI bisnis sering memunculkan masalah secara lebih jelas daripada metrik teknis. Pengamatan pada KPI bisnis dapat mengenali masalah dengan lebih efektif daripada memilah-milah banyak metrik teknis.
- Keselarasan bisnis: Memastikan bahwa kegiatan pemantauan secara langsung mendukung tujuan bisnis.
- Efisiensi: Prioritaskan pemantauan sumber daya dan perhatian pada metrik yang penting.
- Proaktif: Kenali dan atasi masalah sebelum memunculkan dampak bisnis yang lebih luas.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Untuk menentukan KPI beban kerja secara efektif:

1. Mulailah dengan hasil bisnis: Sebelum menyelami metrik, pahami dahulu hasil bisnis yang diinginkan. Apakah peningkatan penjualan, keterlibatan pengguna yang lebih tinggi, atau waktu respons yang lebih cepat?
2. Korelasikan metrik teknis dengan tujuan bisnis: Tidak semua metrik teknis memiliki dampak langsung terhadap hasil bisnis. Identifikasikan metrik yang berdampak langsung terhadap hasil bisnis, tetapi sering kali lebih mudah mengidentifikasi masalah menggunakan KPI bisnis.
3. Gunakan Amazon CloudWatch: Gunakan CloudWatch untuk menentukan dan memantau metrik yang mewakili KPI Anda.
4. Tinjau dan perbarui KPI secara rutin: Saat beban kerja dan bisnis Anda berkembang, jaga agar KPI Anda tetap relevan.
5. Libatkan pemangku kepentingan: Libatkan tim teknis dan bisnis dalam menentukan dan meninjau KPI.

Tingkat upaya untuk rencana implementasi: Sedang

Sumber daya

Praktik terbaik terkait:

- the section called “OPS04-BP02 Mengimplementasikan telemetri aplikasi”
- the section called “OPS04-BP03 Mengimplementasikan telemetri pengalaman pengguna”
- the section called “OPS04-BP04 Mengimplementasikan telemetri dependensi”
- the section called “OPS04-BP05 Mengimplementasikan penelusuran terdistribusi”

Dokumen terkait:

- Praktik Terbaik Observabilitas AWS
- Panduan Pengguna CloudWatch
- Kursus Skill Builder Observabilitas AWS

Video terkait:

- Mengembangkan strategi observabilitas

Contoh terkait:

- Lokakarya One Observability

OPS04-BP02 Mengimplementasikan telemetri aplikasi

Telemetri aplikasi berfungsi sebagai fondasi observabilitas beban kerja Anda. Sangat penting menghadirkan telemetri yang menawarkan wawasan yang dapat ditindaklanjuti tentang keadaan aplikasi Anda serta pencapaian hasil teknis dan bisnis. Dari pemecahan masalah hingga pengukuran dampak fitur baru atau memastikan keselarasan dengan indikator kinerja utama (KPI) bisnis, telemetri aplikasi menjadi patokan bagi cara Anda membangun, mengoperasikan, dan mengembangkan beban kerja Anda.

Metrik, log, dan jejak merupakan tiga pilar utama observabilitas. Ketiganya berfungsi sebagai alat diagnostik yang menggambarkan keadaan aplikasi Anda. Seiring waktu, tiga hal ini membantu menciptakan garis acuan dan mengidentifikasi anomali. Namun, untuk memastikan keselarasan antara aktivitas pemantauan dan tujuan bisnis, KPI harus ditentukan dan dipantau. KPI bisnis sering kali mempermudah identifikasi masalah dibandingkan dengan metrik teknis saja.

Jenis telemetri lainnya, seperti pemantauan pengguna nyata (RUM) dan transaksi sintetis, melengkapi sumber-sumber data primer ini. RUM menawarkan wawasan tentang interaksi pengguna waktu nyata, sedangkan transaksi sintetis menyimulasikan perilaku pengguna potensial, sehingga membantu mendeteksi kemacetan sebelum pengguna nyata mengalaminya.

Hasil yang diinginkan: Dapatkan wawasan yang dapat ditindaklanjuti tentang performa beban kerja Anda. Wawasan ini memungkinkan Anda mengambil keputusan proaktif tentang pengoptimalan performa, mencapai peningkatan stabilitas beban kerja, merampingkan proses CI/CD, dan memanfaatkan sumber daya secara efektif.

Antipola umum:

- Observabilitas yang tidak lengkap: Mengabaikan penggunaan observabilitas di setiap lapisan beban kerja, sehingga mengakibatkan titik buta yang dapat mengaburkan performa sistem vital dan wawasan perilaku.
- Tampilan data terfragmentasi: Ketika data tersebar di beberapa alat dan sistem, mempertahankan pandangan yang menyeluruh tentang kondisi dan performa beban kerja Anda menjadi sulit dilakukan.

- Masalah yang dilaporkan pengguna: Tanda kurangnya deteksi masalah yang proaktif melalui telemetri dan pemantauan KPI bisnis.

Manfaat menjalankan praktik terbaik ini:

- Pengambilan keputusan berbasis informasi: Dengan wawasan dari telemetri dan KPI bisnis, Anda dapat mengambil keputusan berbasis data.
- Peningkatan efisiensi operasional: Pemanfaatan sumber daya berbasis data menghasilkan efektivitas biaya.
- Penyempurnaan stabilitas beban kerja: Deteksi dan penyelesaian masalah yang lebih cepat yang menghasilkan peningkatan waktu aktif.
- Perampingan proses CI/CD: Wawasan dari data telemetri memfasilitasi penyempurnaan proses dan pengiriman kode yang andal.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Untuk mengimplementasikan telemetri aplikasi untuk beban kerja Anda, gunakan layanan AWS seperti Amazon CloudWatch dan AWS X-Ray. Amazon CloudWatch menyediakan rangkaian alat pemantauan yang komprehensif, sehingga Anda dapat mengamati sumber daya dan aplikasi Anda di lingkungan AWS dan on-premise. Layanan ini mengumpulkan, melacak, dan menganalisis metrik, menggabungkan dan memantau data log, dan merespons perubahan dalam sumber daya Anda, menyempurnakan pemahaman Anda tentang bagaimana beban kerja Anda beroperasi. Secara bersamaan, AWS X-Ray memungkinkan Anda melacak, menganalisis, dan men-debug aplikasi Anda, sehingga memberi Anda pemahaman yang mendalam tentang perilaku beban kerja Anda. Dengan fitur seperti peta layanan, distribusi latensi, dan lini waktu penelusuran, X-Ray memberikan wawasan tentang performa beban kerja Anda dan hambatan yang memengaruhinya.

Langkah implementasi

1. Identifikasikan data apa yang akan dikumpulkan: Pastikan metrik, log, dan jejak penting yang akan menawarkan wawasan substansial tentang kondisi, performa, dan perilaku beban kerja Anda.
2. Deploy agen CloudWatch : Agen CloudWatch berperan penting dalam penyediaan metrik dan log sistem serta aplikasi dari beban kerja Anda dan infrastruktur yang mendasarinya. Agen CloudWatch juga dapat digunakan untuk mengumpulkan OpenTelemetry atau jejak X-Ray dan mengirimkannya ke X-Ray.

3. Tentukan dan pantau KPI bisnis: Tetapkan metrik kustom yang selaras dengan hasil bisnis.
4. Lengkapi aplikasi Anda dengan AWS X-Ray: Selain men-deploy agen CloudWatch, sangat penting untuk melengkapi aplikasi Anda untuk menghasilkan data jejak. Proses ini dapat memberikan wawasan lebih lanjut tentang perilaku dan performa beban kerja Anda.
5. Lakukan standardisasi pengumpulan data di seluruh aplikasi Anda: Lakukan standardisasi praktik pengumpulan data di seluruh aplikasi Anda. Keseragaman bermanfaat dalam mengorelasikan dan menganalisis data, sehingga memberikan pandangan yang komprehensif tentang perilaku aplikasi Anda.
6. Analisis dan bertindak berdasarkan data: Setelah pengumpulan dan normalisasi data dilakukan, gunakan Amazon CloudWatch untuk analisis metrik dan log, dan AWS X-Ray untuk analisis jejak. Analisis tersebut dapat menghasilkan wawasan penting tentang kondisi, performa, dan perilaku beban kerja Anda, sehingga memandu proses pengambilan keputusan Anda.

Tingkat upaya untuk rencana implementasi: Tinggi

Sumber daya

Praktik terbaik terkait:

- OPS04-BP01 Identifikasikan indikator performa utama
- OPS04-BP03 Mengimplementasikan telemetri pengalaman pengguna
- OPS04-BP04 Mengimplementasikan telemetri dependensi
- OPS04-BP05 Mengimplementasikan penelusuran terdistribusi

Dokumen terkait:

- Praktik Terbaik Observabilitas AWS
- Panduan Pengguna CloudWatch
- Panduan AWS X-Ray untuk Pengembang
- Menginstrumentasikan sistem terdistribusi untuk visibilitas operasional
- Kursus Skill Builder Observabilitas AWS
- Apa yang Baru dengan Amazon CloudWatch
- Apa yang Baru dengan AWS X-Ray

Video terkait:

- AWS re:Invent 2022 - Praktik terbaik observabilitas di Amazon
- AWS re:Invent 2022 - Mengembangkan strategi observabilitas

Contoh terkait:

- Lokakarya One Observability
- Pustaka Solusi AWS: Pemantauan Aplikasi dengan Amazon CloudWatch

OPS04-BP03 Mengimplementasikan telemetri pengalaman pengguna

Memperoleh wawasan yang mendalam tentang pengalaman dan interaksi pelanggan dengan aplikasi Anda adalah hal krusial. Pemantauan pengguna nyata (RUM) dan transaksi sintetis menjadi alat yang ampuh untuk tujuan ini. RUM menyediakan data tentang interaksi pengguna nyata yang memberikan perspektif kepuasan pengguna tanpa filter, sementara transaksi sintetis mensimulasikan interaksi pengguna, sehingga membantu mendeteksi potensi masalah bahkan sebelum berdampak pada pengguna nyata.

Hasil yang diinginkan: Pandangan yang menyeluruh tentang pengalaman pelanggan, deteksi masalah yang proaktif, dan optimalisasi interaksi pengguna untuk memberikan pengalaman digital yang mulus.

Antipola umum:

- Aplikasi tanpa pemantauan pengguna nyata (RUM):
  - Deteksi masalah yang tertunda: Tanpa RUM, Anda mungkin tidak menyadari kemacetan atau masalah performa sampai pengguna mengeluh. Pendekatan reaktif ini dapat menyebabkan ketidakpuasan pelanggan.
  - Tidak adanya wawasan pengalaman pengguna: Tanpa menggunakan RUM, Anda kehilangan data penting yang menunjukkan bagaimana pengguna nyata berinteraksi dengan aplikasi Anda, sehingga membatasi kemampuan Anda untuk mengoptimalkan pengalaman pengguna.
- Aplikasi tanpa transaksi sintetis:
  - Kasus edge yang terlewatkan: Transaksi sintetis membantu Anda menguji jalur dan fungsi yang mungkin tidak sering digunakan oleh pengguna biasa tetapi sangat penting untuk fungsi bisnis tertentu. Tanpanya, jalur-jalur tersebut bisa mengalami kesalahan fungsi dan luput dari perhatian.

- Memeriksa masalah saat aplikasi tidak digunakan: Pengujian sintetis rutin dapat mensimulasikan saat-saat ketika pengguna nyata tidak berinteraksi secara aktif dengan aplikasi Anda, sehingga memastikan sistem selalu berfungsi dengan benar.

Manfaat menjalankan praktik terbaik ini:

- Deteksi masalah proaktif: Identifikasikan dan atasi potensi masalah sebelum berdampak pada pengguna nyata.
- Pengalaman pengguna yang dioptimalkan: Umpan balik yang berkelanjutan dari RUM membantu menyempurnakan dan meningkatkan pengalaman pengguna secara keseluruhan.
- Wawasan tentang performa perangkat dan browser: Memahami performa aplikasi Anda di berbagai perangkat dan browser, sehingga memungkinkan pengoptimalan lebih lanjut.
- Alur kerja bisnis yang divalidasi: Transaksi sintetis yang rutin memastikan fungsionalitas inti dan jalur-jalur kritis tetap berjalan dan efisien.
- Performa aplikasi yang ditingkatkan: Manfaatkan wawasan yang dikumpulkan dari data pengguna nyata untuk meningkatkan responsivitas dan keandalan aplikasi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Untuk memanfaatkan RUM dan transaksi sintetis untuk telemetri aktivitas pengguna, AWS menawarkan layanan seperti Amazon CloudWatch RUM dan Amazon CloudWatch Synthetics. Metrik, log, dan jejak, ditambah dengan data aktivitas pengguna, memberikan pandangan yang komprehensif tentang status operasional aplikasi dan pengalaman pengguna.

Langkah implementasi

1. Lakukan deployment Amazon CloudWatch RUM: Integrasikan aplikasi Anda dengan CloudWatch RUM untuk mengumpulkan, menganalisis, dan menyajikan data pengguna nyata.
   a. Gunakan perpustakaan JavaScript CloudWatch RUM untuk mengintegrasikan RUM dengan aplikasi Anda.
   b. Siapkan dasbor untuk memvisualisasikan dan memantau data pengguna nyata.
2. Konfigurasikan CloudWatch Synthetics: Buat canary, atau rutinitas terprogram, yang mensimulasikan interaksi pengguna dengan aplikasi Anda.
   a. Tentukan alur kerja dan jalur aplikasi kritis.

   b. Rancang canary menggunakan skrip CloudWatch Synthetics untuk mensimulasikan interaksi pengguna untuk jalur-jalur tersebut.
   c. Jadwalkan dan pantau canary agar berjalan pada interval tertentu, sehingga memastikan pemeriksaan performa yang konsisten.
3. Analisis dan tindak lanjuti data: Manfaatkan data dari RUM dan transaksi sintetis untuk mendapatkan wawasan dan mengambil tindakan korektif ketika anomali terdeteksi. Gunakan dasbor dan alarm CloudWatch untuk tetap memutakhirkan informasi.

Tingkat upaya untuk rencana implementasi: Sedang

Sumber daya

Praktik terbaik terkait:

- OPS04-BP01 Identifikasikan indikator performa utama
- OPS04-BP02 Mengimplementasikan telemetri aplikasi
- OPS04-BP04 Mengimplementasikan telemetri dependensi
- OPS04-BP05 Mengimplementasikan penelusuran terdistribusi

Dokumen terkait:

- Panduan Amazon CloudWatch RUM
- Panduan Amazon CloudWatch Synthetics

Video terkait:

- Mengoptimalkan aplikasi melalui wawasan pengguna akhir dengan RUM
- AWS on Air ft. Pemantauan Pengguna Nyata untuk Amazon CloudWatch

Contoh terkait:

- Lokakarya One Observability
- Repositori Git untuk Klien Web Amazon CloudWatch RUM
- Menggunakan Amazon CloudWatch Synthetics untuk mengukur waktu pemuatan halaman

OPS04-BP04 Mengimplementasikan telemetri dependensi

Telemetri dependensi sangat penting untuk memantau kondisi dan performa layanan dan komponen eksternal yang diandalkan oleh beban kerja Anda. Hal ini memberikan wawasan berharga tentang keterjangkauan, batas waktu, dan peristiwa penting lainnya yang terkait dengan dependensi seperti DNS, basis data, atau API pihak ketiga. Dengan menginstrumentasi aplikasi Anda agar menghasilkan metrik, log, dan jejak tentang dependensi ini, Anda mendapatkan pemahaman yang lebih jelas tentang potensi kemacetan, masalah performa, atau kegagalan yang dapat memengaruhi beban kerja Anda.

Hasil yang diinginkan: Dependensi yang diandalkan beban kerja Anda menunjukkan performa sesuai harapan, sehingga Anda dapat secara proaktif mengatasi masalah dan memastikan performa beban kerja yang optimal.

Antipola umum:

- Mengabaikan dependensi eksternal: Hanya berfokus pada metrik aplikasi internal sambil mengabaikan metrik yang berkaitan dengan dependensi eksternal.
- Kurangnya pemantauan proaktif: Menunggu masalah muncul alih-alih terus memantau kondisi dan performa dependensi.
- Pemantauan model silo: Menggunakan beberapa alat pemantauan yang berbeda-beda sehingga wawasan tentang kondisi dependensi menjadi terfragmentasi dan tidak konsisten.

Manfaat menjalankan praktik terbaik ini:

- Peningkatan keandalan beban kerja: Dengan memastikan bahwa dependensi eksternal terus-menerus tersedia dan berkinerja optimal.
- Deteksi dan penyelesaian masalah yang lebih cepat: Secara proaktif mengidentifikasi dan menangani masalah pada dependensi sebelum berdampak pada beban kerja.
- Pandangan menyeluruh: Mendapatkan pandangan yang menyeluruh tentang komponen internal dan eksternal yang memengaruhi kondisi beban kerja.
- Peningkatan skalabilitas beban kerja: Dengan memahami batas skalabilitas dan karakteristik performa dependensi eksternal.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Implementasikan telemetri dependensi dengan memulai dari identifikasi layanan, infrastruktur, dan proses yang digunakan oleh beban kerja Anda. Ukur seperti apa kondisi yang baik ketika dependensi berfungsi sesuai harapan, kemudian tentukan data apa yang diperlukan untuk mengukurnya. Dengan informasi tersebut, Anda dapat membuat dasbor dan peringatan yang memberikan wawasan kepada tim operasi Anda tentang status dependensi tersebut. Gunakan alat AWS untuk menemukan dan mengukur dampak ketika dependensi tidak dapat menunjukkan hasil sesuai kebutuhan. Selalu tinjau ulang strategi Anda agar memperhitungkan perubahan prioritas, sasaran, dan wawasan yang diperoleh.

Langkah implementasi

Untuk mengimplementasikan telemetri dependensi secara efektif:

1. Identifikasikan dependensi eksternal: Lakukan kolaborasi dengan pemangku kepentingan untuk menentukan dependensi eksternal yang diandalkan oleh beban kerja Anda. Dependensi eksternal dapat mencakup layanan seperti basis data eksternal, API pihak ketiga, rute konektivitas jaringan ke lingkungan lain, dan layanan DNS. Langkah pertama menuju telemetri dependensi yang efektif adalah memiliki pemahaman yang menyeluruh tentang apa saja dependensi tersebut.
2. Kembangkan strategi pemantauan: Setelah Anda memiliki gambaran yang jelas tentang dependensi eksternal Anda, rancanglah strategi pemantauan yang disesuaikan dengan dependensi tersebut. Ini melibatkan pemahaman tingkat kekritisan setiap dependensi, perilaku yang diharapkan, dan perjanjian atau target tingkat layanan (SLA atau SLT) terkait. Siapkan peringatan proaktif untuk memberi tahu Anda tentang perubahan status atau penyimpangan performa.
3. Manfaatkan Amazon CloudWatch Internet Monitor: Layanan ini menawarkan wawasan tentang internet global, sehingga membantu memahami pemadaman atau gangguan yang mungkin memengaruhi dependensi eksternal Anda.
4. Mutakhirkan informasi dengan Dasbor AWS Health: Layanan ini memberikan peringatan dan panduan remediasi ketika AWS mengalami peristiwa yang dapat memengaruhi layanan Anda.
5. Lengkapi aplikasi Anda dengan AWS X-Ray: AWS X-Ray memberikan wawasan tentang bagaimana performa aplikasi dan dependensi yang mendasarinya. Dengan melacak permintaan dari awal hingga akhir, Anda dapat mengidentifikasi kemacetan atau kegagalan dalam layanan eksternal atau komponen yang diandalkan oleh aplikasi Anda.
6. Gunakan Amazon DevOps Guru: Layanan berbasis pembelajaran mesin ini mengidentifikasi masalah operasional, memprediksi kapan masalah kritis mungkin terjadi, dan merekomendasikan

tindakan spesifik yang harus diambil. Layanan ini sangat bermanfaat untuk mendapatkan wawasan tentang dependensi dan menentukan bahwa dependensi bukan sumber masalah operasional.

7. Pantau secara teratur: Terus pantau metrik dan log yang berkaitan dengan dependensi eksternal. Siapkan peringatan untuk perilaku tak terduga atau performa yang menurun.
8. Lakukan validasi setelah perubahan: Setiap kali ada pembaruan atau perubahan pada salah satu dependensi eksternal, lakukan validasi performa dan periksa keselarasannya dengan persyaratan aplikasi Anda.

Tingkat upaya untuk rencana implementasi: Sedang

Sumber daya

Praktik terbaik terkait:

- OPS04-BP01 Identifikasikan indikator performa utama
- OPS04-BP02 Mengimplementasikan telemetri aplikasi
- OPS04-BP03 Mengimplementasikan telemetri pengalaman pengguna
- OPS04-BP05 Mengimplementasikan penelusuran terdistribusi

Dokumen terkait:

- Apa itu AWS Health?
- Menggunakan Amazon CloudWatch Internet Monitor
- Panduan AWS X-Ray untuk Pengembang
- Panduan Pengguna Amazon DevOps Guru

Video terkait:

- Visibilitas tentang bagaimana masalah internet memengaruhi performa aplikasi
- Pengantar Amazon DevOps Guru

Contoh terkait:

- Mendapatkan wawasan operasional dengan AIOps menggunakan Amazon DevOps Guru

- AWS Health Aware

OPS04-BP05 Mengimplementasikan penelusuran terdistribusi

Penelusuran terdistribusi menawarkan cara untuk memantau dan memvisualisasikan permintaan yang melintasi berbagai komponen sistem terdistribusi. Dengan menangkap data jejak dari berbagai sumber dan menganalisisnya dalam tampilan terpadu, tim dapat lebih memahami bagaimana permintaan mengalir, di mana kemacetan terjadi, dan di mana upaya pengoptimalan harus difokuskan.

Hasil yang diinginkan: Dapatkan tampilan menyeluruh permintaan yang mengalir melewati sistem terdistribusi Anda, sehingga memungkinkan debugging yang presisi, performa yang dioptimalkan, dan pengalaman pengguna yang lebih baik.

Antipola umum:

- Instrumentasi yang tidak konsisten: Tidak semua layanan dalam sistem terdistribusi diinstrumentasi untuk penelusuran.
- Mengabaikan latensi: Hanya berfokus pada kesalahan dan tidak mempertimbangkan latensi atau penurunan performa bertahap.

Manfaat menjalankan praktik terbaik ini:

- Gambaran umum sistem yang komprehensif: Memvisualisasikan seluruh jalur permintaan, dari masuk hingga keluar.
- Debugging yang disempurnakan: Mengidentifikasi dengan cepat di mana kegagalan atau masalah performa terjadi.
- Pengalaman pengguna yang ditingkatkan: Memantau dan mengoptimalkan berdasarkan data pengguna aktual, memastikan sistem memenuhi tuntutan dunia nyata.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Mulailah dengan mengidentifikasi semua elemen beban kerja Anda yang memerlukan instrumentasi. Setelah semua komponen diperhitungkan, manfaatkan alat seperti AWS X-Ray dan OpenTelemetry untuk mengumpulkan data jejak untuk dianalisis dengan alat seperti X-Ray dan Amazon CloudWatch ServiceLens Map. Lakukan peninjauan rutin dengan developer, dan lengkapi diskusi tersebut

dengan alat seperti Amazon DevOps Guru, Analitik X-Ray, dan Wawasan X-Ray untuk membantu mengungkap temuan yang lebih mendalam. Buat peringatan dari data jejak untuk memberi tahu kapan hasil, sebagaimana didefinisikan dalam rencana pemantauan beban kerja, mengandung risiko.

Langkah implementasi

Untuk mengimplementasikan penelusuran terdistribusi secara efektif:

1. Adopsi AWS X-Ray: Integrasikan X-Ray ke dalam aplikasi Anda untuk mendapatkan wawasan tentang perilakunya, memahami performanya, dan mengenali kemacetan. Manfaatkan Wawasan X-Ray untuk analisis jejak otomatis.
2. Lengkapi layanan Anda: Verifikasi bahwa setiap layanan, dari fungsi AWS Lambda hingga instans EC2, mengirimkan data jejak. Makin banyak layanan yang Anda lengkapi, maka makin jelas tampilan yang menyeluruh.
3. Sertakan Pemantauan Pengguna Nyata CloudWatch dan pemantauan sintetis: Integrasikan Pemantauan Pengguna Nyata (RUM) dan pemantauan sintetis dengan X-Ray. Hal ini memungkinkan perekaman pengalaman pengguna dunia nyata dan simulasi interaksi pengguna untuk mengidentifikasi potensi masalah.
4. Gunakan agen CloudWatch: Agen ini dapat mengirimkan jejak dari X-Ray atau OpenTelemetry, sehingga meningkatkan kedalaman wawasan yang diperoleh.
5. Gunakan Amazon DevOps Guru: DevOps Guru menggunakan data dari X-Ray, CloudWatch, AWS Config, dan AWS CloudTrail untuk memberikan rekomendasi yang dapat ditindaklanjuti.
6. Lakukan analisis jejak: Tinjau data jejak secara rutin untuk membedakan pola, anomali, atau kemacetan yang dapat memengaruhi performa aplikasi Anda.
7. Siapkan peringatan: Konfigurasikan alarm di CloudWatch untuk pola yang tidak biasa atau latensi yang meluas, sehingga memungkinkan penanganan masalah secara proaktif.
8. Peningkatan berkelanjutan: Tinjau ulang strategi penelusuran Anda saat layanan ditambahkan atau dimodifikasi untuk menangkap semua titik data yang relevan.

Tingkat upaya untuk rencana implementasi: Sedang

Sumber daya

Praktik terbaik terkait:

- OPS04-BP01 Identifikasikan indikator performa utama
- OPS04-BP02 Mengimplementasikan telemetri aplikasi

- OPS04-BP03 Mengimplementasikan telemetri pengalaman pengguna
- OPS04-BP04 Mengimplementasikan telemetri dependensi

Dokumen terkait:

- Panduan AWS X-Ray untuk Pengembang
- Panduan Pengguna agen Amazon CloudWatch
- Panduan Pengguna Amazon DevOps Guru

Video terkait:

- Gunakan Wawasan AWS X-Ray
- AWS on Air ft. Observabilitas: Amazon CloudWatch dan AWS X-Ray

Contoh terkait:

- Menginstrumentasi Aplikasi Anda dengan AWS X-Ray

OPS 5. Bagaimana cara mengurangi kecacatan, mempermudah perbaikan, dan meningkatkan aliran ke dalam produksi?

Adopsi pendekatan yang meningkatkan aliran perubahan ke dalam produksi, yang memampukan pemfaktoran ulang, umpan balik cepat atas kualitas, dan perbaikan bug. Ini mempercepat perubahan yang bermanfaat memasuki produksi, membatasi masalah yang di-deploy, dan mencapai identifikasi cepat serta perbaikan masalah akibat aktivitas deployment.

Praktik terbaik

- OPS05-BP01 Menggunakan kontrol versi
- OPS05-BP02 Menguji dan memvalidasi perubahan
- OPS05-BP03 Menggunakan sistem manajemen konfigurasi
- OPS05-BP04 Menggunakan sistem manajemen build dan deployment
- OPS05-BP05 Melakukan manajemen patch
- OPS05-BP06 Membagikan standar desain
- OPS05-BP07 Mengimplementasikan praktik untuk meningkatkan kualitas kode

- OPS05-BP08 Menggunakan beberapa lingkungan
- OPS05-BP09 Membuat perubahan yang sering, kecil, dan dapat dikembalikan
- OPS05-BP10 Mengotomatiskan integrasi dan deployment sepenuhnya

OPS05-BP01 Menggunakan kontrol versi

Gunakan kontrol versi untuk memungkinkan pelacakan perubahan dan rilis.

Banyak layanan AWS menawarkan kemampuan kontrol versi. Gunakan sistem kontrol revisi atau sumber seperti AWS CodeCommit untuk mengelola kode dan artefak lain, seperti templat AWS CloudFormation yang dikontrol versi dari infrastruktur Anda.

Hasil yang diinginkan: Tim Anda berkolaborasi mengerjakan kode. Saat digabungkan, kode tersebut konsisten dan tidak ada perubahan yang hilang. Kesalahan mudah dibatalkan melalui versioning yang benar.

Antipola umum:

- Anda telah mengembangkan dan menyimpan kode di stasiun kerja Anda. Anda mengalami kegagalan penyimpanan yang tidak dapat dipulihkan di stasiun kerja lalu kode Anda hilang.
- Setelah menimpa kode yang ada dengan perubahan Anda, Anda memulai ulang aplikasi namun sudah tidak dapat beroperasi lagi. Anda tidak bisa membatalkan perubahan.
- Anda memiliki write lock pada file laporan yang perlu diedit orang lain. Mereka meminta Anda untuk berhenti mengerjakannya agar mereka bisa menyelesaikan tugas mereka.
- Tim penelitian Anda telah mengerjakan analisis mendetail yang membentuk pekerjaan mendatang Anda. Seseorang secara tidak sengaja menyimpan daftar belanjanya dan menimpa laporan akhir. Anda tidak bisa membatalkan perubahan dan harus membuat ulang laporan tersebut.

Manfaat menjalankan praktik terbaik ini: Dengan menggunakan kemampuan kontrol versi, Anda dapat secara mudah kembali ke versi sebelumnya dengan status baik, dan membatasi risiko kehilangan aset.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Pelihara aset di repositori dengan kontrol versi. Tindakan ini mendukung pelacakan perubahan, deployment versi baru, deteksi perubahan pada versi yang ada, dan pengembalian ke versi

sebelumnya (misalnya, kembali ke versi dengan status baik apabila terjadi kegagalan). Integrasikan kemampuan kontrol versi sistem manajemen konfigurasi Anda ke dalam prosedur Anda.

Sumber daya

Praktik terbaik terkait:

- OPS05-BP04 Menggunakan sistem manajemen build dan deployment

Dokumen terkait:

- Apa Itu AWS CodeCommit?

Video terkait:

- Pengantar AWS CodeCommit

OPS05-BP02 Menguji dan memvalidasi perubahan

Setiap perubahan yang di-deploy harus diuji untuk menghindari kesalahan dalam produksi. Praktik terbaik ini difokuskan untuk menguji perubahan dari kontrol versi hingga build artefak. Di samping perubahan kode aplikasi, pengujian harus menyertakan infrastruktur, konfigurasi, kontrol keamanan, dan prosedur operasi. Ada banyak bentuk pengujian, dari uji unit hingga analisis komponen perangkat lunak (SCA). Makin ke kiri pengujian dalam proses integrasi dan pengiriman perangkat lunak menghasilkan tingkat kepastian kualitas artefak yang lebih tinggi.

Organisasi Anda harus mengembangkan standar pengujian untuk semua artefak perangkat lunak. Pengujian otomatis dapat mengurangi kerja yang melelahkan dan mencegah kesalahan pengujian manual. Uji manual mungkin diperlukan dalam beberapa kasus. Developer harus memiliki akses ke hasil uji otomatis untuk menciptakan loop umpan balik yang meningkatkan kualitas perangkat lunak.

Hasil yang diinginkan: Perubahan perangkat lunak Anda diuji sebelum dikirim. Pengembang memiliki akses ke hasil pengujian dan validasi. Organisasi memiliki standar pengujian yang berlaku untuk semua perubahan perangkat lunak.

Antipola umum:

- Anda men-deploy perubahan perangkat lunak baru tanpa pengujian apa pun. Perangkat lunak gagal berjalan dalam produksi, dan mengakibatkan matinya sistem.

- Grup keamanan baru di-deploy dengan CloudFormation tanpa diuji dalam lingkungan pra-produksi. Grup keamanan tersebut menjadikan aplikasi Anda tidak terjangkau oleh pelanggan Anda.
- Sebuah metode diubah tanpa pengujian unit. Perangkat lunak gagal saat di-deploy ke produksi.

Manfaat menjalankan praktik terbaik ini: Tingkat kegagalan perubahan deployment perangkat lunak menjadi berkurang. Kualitas perangkat lunak meningkat. Developer memiliki kesadaran yang lebih tinggi tentang kelayakan kode mereka. Kebijakan keamanan dapat diluncurkan dengan penuh keyakinan untuk mendukung kepatuhan organisasi. Perubahan infrastruktur seperti pembaruan kebijakan penskalaan otomatis diuji di awal untuk memenuhi kebutuhan lalu lintas.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Pengujian dilakukan pada semua perubahan, dari kode aplikasi hingga infrastruktur, sebagai bagian dari praktik integrasi berkelanjutan Anda. Hasil pengujian dipublikasikan sehingga developer memiliki umpan balik yang cepat. Organisasi memiliki standar pengujian bahwa semua perubahan harus lulus.

Contoh pelanggan

Sebagai bagian dari pipeline integrasi berkelanjutan mereka, AnyCompany Retail melakukan beberapa jenis pengujian pada semua artefak perangkat lunak. Mereka mempraktikkan pengembangan yang didorong pengujian sehingga semua perangkat lunak memiliki pengujian-pengujian unit. Begitu artefak dibangun, mereka menjalankan pengujian menyeluruh. Setelah pengujian putaran pertama selesai, mereka menjalankan pemindaian keamanan aplikasi statis, yang mencari kerentanan yang dikenali. Developer menerima pesan setelah setiap gerbang pengujian dilalui. Setelah semua pengujian selesai, artefak perangkat lunak disimpan di dalam repositori artefak.

Langkah implementasi

1. Bekerjalah dengan pemangku kepentingan di organisasi Anda untuk mengembangkan standar pengujian untuk artefak perangkat lunak. Pengujian standar apa yang harus dilalui oleh semua artefak? Apakah ada persyaratan kepatuhan atau tata kelola yang harus disertakan di dalam cakupan pengujian? Apakah Anda perlu melakukan pengujian kualitas kode? Setelah pengujian selesai, siapa yang perlu mengetahuinya?
   a. Perintah Rujukan Pipeline Deployment AWS berisi daftar tepercaya untuk jenis-jenis pengujian yang dapat dilakukan pada artefak perangkat lunak sebagai bagian dari pipeline integrasi.

2. Instrumentasikan aplikasi Anda dengan pengujian yang diperlukan berdasarkan standar pengujian perangkat lunak Anda. Setiap set pengujian harus selesai dalam waktu kurang dari sepuluh menit. Pengujian harus berjalan sebagai bagian dari pipeline integrasi.
   a. Amazon CodeGuru Reviewer dapat menguji kode aplikasi Anda untuk mendeteksi kecacatan.
   b. Anda dapat menggunakan AWS CodeBuild untuk melakukan pengujian pada artefak perangkat lunak.
   c. AWS CodePipeline dapat mengorkestrasi pengujian perangkat lunak Anda ke dalam pipeline.

Sumber daya

Praktik terbaik terkait:

- OPS05-BP01 Menggunakan kontrol versi
- OPS05-BP06 Membagikan standar desain
- OPS05-BP10 Mengotomatiskan integrasi dan deployment sepenuhnya

Dokumen terkait:

- Adopsi pendekatan pengembangan yang didorong pengujian
- Pipeline Pengujian CloudFormation Otomatis dengan TaskCat dan CodePipeline
- Membangun pipeline CI/CD DevSecOps AWS yang menyeluruh dengan alat-alat SCA, SAST, dan DAST sumber terbuka
- Memulai pengujian aplikasi nirserver
- Pipeline CI/CD adalah pemandu utama rilisan saya
- Laporan resmi Mempraktikkan Integrasi Berkelanjutan dan Pengiriman Berkelanjutan di AWS

Video terkait:

- AWS re:Invent 2020: Infrastruktur yang dapat diuji: Pengujian integrasi di AWS
- AWS Summit ANZ 2021 - Mendorong strategi yang mengutamakan pengujian dengan CDK dan pengembangan yang didorong pengujian
- Menguji Infrastruktur sebagai Kode dengan AWS CDK

Sumber daya terkait:

- Arsitektur Rujukan Pipeline Deployment AWS - Aplikasi
- Pipeline DevSecOps Kubernetes AWS
- Lokakarya Kebijakan sebagai Kode – Pengembangan yang Didorong Pengujian
- Menjalankan pengujian unit untuk aplikasi Node.js dari GitHub dengan menggunakan AWS CodeBuild
- Menggunakan Serverspec untuk pengembangan kode infrastruktur yang didorong pengujian

Layanan terkait:

- Amazon CodeGuru Reviewer
- AWS CodeBuild
- AWS CodePipeline

OPS05-BP03 Menggunakan sistem manajemen konfigurasi

Gunakan sistem manajemen konfigurasi untuk membuat dan melacak perubahan konfigurasi. Sistem ini mengurangi kesalahan yang disebabkan oleh proses manual dan meminimalkan tingkat upaya untuk melakukan deployment perubahan.

Manajemen konfigurasi statis menetapkan nilai saat menginisialisasi sumber daya yang diharapkan tetap konsisten selama masa pakai sumber daya. Beberapa contoh menyertakan pengaturan konfigurasi untuk web atau server aplikasi pada instans, atau menentukan konfigurasi layanan AWS dalam Konsol Manajemen AWS atau melalui AWS CLI.

Manajemen konfigurasi dinamis menetapkan nilai saat inisialisasi. Nilai ini dapat atau diharapkan berubah selama masa pakai sumber daya. Misalnya, Anda dapat menetapkan toggle fitur untuk mengaktifkan fungsionalitas dalam kode melalui perubahan konfigurasi, atau mengubah tingkat detail log selama insiden untuk memperoleh lebih banyak data, lalu mengubahnya kembali setelah insiden menghilangkan log yang saat ini tidak dibutuhkan dan pengeluaran yang terkait dengannya.

Di AWS, Anda dapat menggunakan AWS Config untuk terus mengawasi konfigurasi sumber daya AWS Anda di seluruh akun dan Wilayah. Dengan demikian, Anda dapat melacak riwayat konfigurasi mereka, memahami bagaimana perubahan konfigurasi akan memengaruhi sumber daya lainnya, dan mengauditnya terhadap konfigurasi yang diharapkan atau diinginkan dengan menggunakan Aturan AWS Config dan Paket Konformasi AWS Config.

Jika Anda memiliki konfigurasi dinamis di aplikasi Anda yang berjalan di instans Amazon EC2, AWS Lambda, kontainer, perangkat seluler, atau perangkat IoT, Anda dapat menggunakan AWS AppConfig untuk mengonfigurasi, memvalidasi, men-deploy, dan memantaunya di seluruh lingkungan Anda.

Di AWS, Anda dapat membuat pipeline integrasi berkelanjutan/deployment berkelanjutan (CI/CD) menggunakan layanan seperti Alat Developer AWS (misalnya, AWS CodeCommit, AWS CodeBuild, AWS CodePipeline, AWS CodeDeploy, dan AWS CodeStar).

Hasil yang diinginkan: Anda mengonfigurasi, memvalidasi, dan melakukan deployment sebagai bagian dari pipeline integrasi berkelanjutan, pengiriman berkelanjutan (CI/CD) Anda. Anda memantau untuk memvalidasi bahwa konfigurasi sudah benar. Hal ini meminimalkan dampak apa pun terhadap pelanggan dan pengguna akhir.

Antipola umum:

- Anda memperbarui konfigurasi server web secara manual di seluruh armada dan beberapa server menjadi tidak responsif karena kesalahan pembaruan.
- Anda memperbarui armada server aplikasi selama berjam-jam. Inkonsistensi dalam konfigurasi selama perubahan menyebabkan perilaku tak terduga.
- Seseorang telah memperbarui grup keamanan Anda dan server web Anda tidak lagi dapat diakses. Tanpa mengetahui apa yang telah diubah, Anda menghabiskan banyak waktu untuk menyelidiki masalah tersebut sehingga memperpanjang waktu pemulihan.
- Anda mendorong konfigurasi pra-produksi ke dalam produksi melalui CI/CD tanpa validasi. Anda mengekspos pengguna dan pelanggan ke data dan layanan yang salah.

Manfaat menjalankan praktik terbaik ini: Mengadopsi sistem manajemen konfigurasi meminimalkan tingkat upaya untuk membuat dan melacak perubahan, serta mengurangi frekuensi kesalahan yang disebabkan prosedur manual. Sistem manajemen konfigurasi memberikan jaminan sehubungan dengan persyaratan tata kelola, kepatuhan, dan peraturan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Sistem manajemen konfigurasi digunakan untuk melacak dan mengimplementasikan perubahan pada konfigurasi aplikasi dan lingkungan. Sistem manajemen konfigurasi juga digunakan untuk mengurangi kesalahan yang disebabkan oleh proses manual, membuat perubahan konfigurasi berulang dan dapat diaudit, dan mengurangi tingkat upaya.

Langkah implementasi

1. Identifikasikan pemilik konfigurasi.
   a. Buat agar pemilik konfigurasi menyadari kebutuhan kepatuhan, tata kelola, atau peraturan apa pun.
2. Identifikasikan item konfigurasi dan hasil kerja.
   a. Item konfigurasi adalah semua konfigurasi aplikasi dan lingkungan yang dipengaruhi oleh deployment dalam pipeline CI/CD Anda.
   b. Hasil kerja mencakup kriteria keberhasilan, validasi, dan hal-hal yang harus dipantau.
3. Pilih alat untuk manajemen konfigurasi berdasarkan kebutuhan bisnis dan pipeline pengiriman Anda.
4. Pertimbangkan deployment tertimbang seperti deployment canary untuk perubahan konfigurasi yang signifikan guna meminimalkan dampak konfigurasi yang salah.
5. Integrasikan manajemen konfigurasi Anda ke dalam pipeline CI/CD Anda.
6. Validasikan semua perubahan yang didorong.

Sumber daya

Praktik terbaik terkait:

- OPS06-BP01 Antisipasikan perubahan yang tidak berhasil
- OPS06-BP02 Menguji deployment
- OPS06-BP03 Menggunakan strategi deployment yang aman
- OPS06-BP04 Mengotomatiskan pengujian dan pengembalian (rollback)

Dokumen terkait:

- AWS Control Tower
- Akselerator Zona Pendaratan AWS
- AWS Config
- Apa itu AWS Config?
- AWS AppConfig
- Apa itu AWS CloudFormation?
- Alat Developer AWS

Video terkait:

- AWS re:Invent 2022 - Tata kelola dan kepatuhan proaktif untuk beban kerja AWS
- AWS re:Invent 2020: Capai kepatuhan sebagai kode menggunakan AWS Config
- Kelola dan Deploy Konfigurasi Aplikasi dengan AWS AppConfig

OPS05-BP04 Menggunakan sistem manajemen build dan deployment

Gunakan sistem manajemen build dan deployment. Sistem ini mengurangi kesalahan yang disebabkan oleh proses manual dan meminimalkan tingkat upaya untuk melakukan deployment perubahan.

Di AWS, Anda dapat membangun pipeline integrasi berkelanjutan/deployment berkelanjutan (CI/CD) menggunakan layanan seperti Alat Pengembang AWS (misalnya, AWS CodeCommit, AWS CodeBuild, AWS CodePipeline, AWS CodeDeploy, dan AWS CodeStar).

Hasil yang diinginkan: Sistem manajemen build dan deployment Anda mendukung sistem integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD) organisasi Anda yang menyediakan kemampuan untuk mengotomatisasi peluncuran aman dengan konfigurasi yang benar.

Antipola umum:

- Setelah menyusun kode pada sistem pengembangan, Anda menyalin file yang dapat dieksekusi ke sistem produksi namun file tersebut gagal untuk memulai. File log lokal mengindikasikan bahwa kegagalan tersebut dikarenakan hilangnya dependensi.
- Anda berhasil membangun aplikasi Anda dengan fitur baru pada lingkungan pengembangan dan memberikan kodenya ke tim jaminan kualitas (QA). Kode tersebut gagal dalam QA karena ada aset statis yang hilang.
- Pada hari Jumat, setelah berupaya keras, Anda berhasil membangun aplikasi Anda secara manual di lingkungan pengembangan Anda termasuk fitur yang baru dikodekan. Pada hari Senin, Anda tidak dapat mengulangi langkah-langkah yang membuat Anda berhasil membangun aplikasi.
- Anda melakukan pengujian yang telah Anda buat untuk rilisan baru Anda. Kemudian Anda menghabiskan minggu selanjutnya untuk mempersiapkan lingkungan pengujian dan melakukan seluruh pengujian integrasi yang ada disusul dengan pengujian kinerja. Kode baru tersebut memiliki dampak kinerja yang tidak dapat diterima dan harus dikembangkan ulang dan kemudian diuji ulang.

Manfaat menerapkan praktik terbaik ini: Dengan menyediakan mekanisme untuk mengatasi aktivitas build dan deployment, Anda mengurangi upaya yang diperlukan untuk melakukan tugas berulang, membebaskan anggota tim Anda untuk fokus pada tugas kreatif mereka yang berharga, serta mengurangi terjadinya kesalahan akibat prosedur manual.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

Sistem manajemen build dan deployment digunakan untuk melacak dan mengimplementasikan perubahan, mengurangi kesalahan yang disebabkan oleh proses manual, dan mengurangi upaya yang diperlukan untuk deployment yang aman. Otomatiskan sepenuhnya pipeline integrasi dan deployment dari check-in kode hingga build, pengujian, deployment, dan validasi. Hal ini mempersingkat waktu tunggu, mengurangi biaya, mendorong peningkatan frekuensi perubahan, mengurangi tingkat upaya, dan meningkatkan kolaborasi.

Langkah implementasi

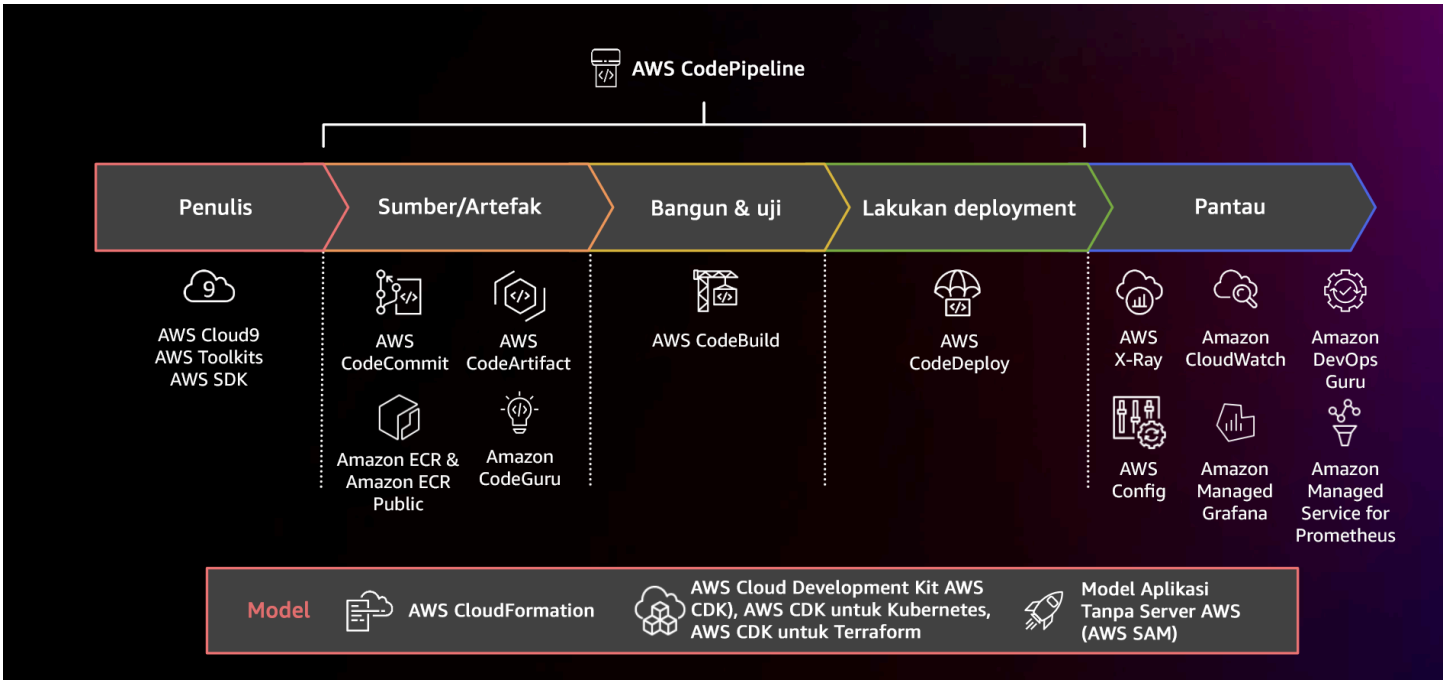


Diagram yang menunjukkan pipeline CI/CD menggunakan AWS CodePipeline dan layanan terkait

1. Gunakan AWS CodeCommit untuk mengontrol versi, menyimpan, dan mengelola aset (seperti dokumen, kode sumber, dan file biner).
2. Gunakan CodeBuild untuk mengompilasi kode sumber Anda, menjalankan pengujian unit, dan memproduksi artefak yang siap untuk deployment.

3. Gunakan CodeDeploy sebagai layanan deployment yang mengotomatiskan deployment aplikasi ke instans Amazon EC2 , instans on-premise, fungsi AWS Lambda nirserver, atau Amazon ECS.
4. Pantau deployment Anda.

Sumber daya

Praktik terbaik terkait:

- OPS06-BP04 Mengotomatiskan pengujian dan pengembalian (rollback)

Dokumen terkait:

- Alat Pengembang AWS
- Apa Itu AWS CodeCommit?
- Apa itu AWS CodeBuild?
- AWS CodeBuild
- Apa itu AWS CodeDeploy?

Video terkait:

- AWS re:Invent 2022 - Praktik terbaik AWS Well-Architected untuk DevOps di AWS

OPS05-BP05 Melakukan manajemen patch

Lakukan manajemen patch untuk mendapatkan fitur, menangani permasalahan, dan menjaga kepatuhan terhadap tata kelola. Otomatiskan manajemen patch untuk mengurangi kesalahan yang disebabkan oleh proses manual, menskalakan, dan mengurangi upaya untuk melakukan patch.

Manajemen patch dan kerentanan adalah bagian dari aktivitas manajemen manfaat dan risiko Anda. Lebih baik miliki infrastruktur tetap dan deploy beban kerja pada status yang diketahui baik dan terverifikasi. Jika tidak memungkinkan, opsi yang tersisa ialah menerapkan patching.

Amazon EC2 Image Builder menyediakan pipeline untuk memperbarui image mesin. Sebagai bagian dari manajemen patch, pertimbangkan Amazon Machine Images (AMI) menggunakan Pipeline image AMI atau image kontainer dengan Pipeline image Docker, sementara AWS Lambda memberikan pola bagi runtime kustom dan pustaka tambahan untuk menghapus kerentanan.

Anda harus mengelola pembaruan pada Amazon Machine Images untuk image Linux atau Windows Server menggunakan Amazon EC2 Image Builder. Anda dapat menggunakan Amazon Elastic Container Registry (Amazon ECR) dengan pipeline yang sudah ada untuk mengelola image Amazon ECS dan mengelola image Amazon EKS. Lambda mencakup fitur manajemen versi.

Patching tidak boleh dilakukan pada sistem produksi tanpa mengujinya terlebih dahulu di lingkungan yang aman. Patch hanya bisa diterapkan jika mendukung hasil operasi atau bisnis. Di AWS, Anda dapat menggunakan AWS Systems Manager Patch Manager untuk mengotomatiskan proses patching sistem terkelola dan menjadwalkan aktivitas menggunakan Periode Pemeliharaan Systems Manager.

Hasil yang diinginkan: Image AMI dan kontainer Anda diberikan patch, diperbarui, dan siap diluncurkan. Anda dapat melacak status semua image yang di-deploy dan mengetahui kepatuhan patch. Anda dapat melaporkan status saat ini dan memiliki proses untuk memenuhi kebutuhan kepatuhan Anda.

Antipola umum:

- Anda diberi tugas untuk menerapkan semua patch keamanan baru dalam waktu dua jam yang menyebabkan beberapa pemadaman akibat ketidaksesuaian aplikasi dengan patch.
- Pustaka yang tidak di-patch menimbulkan konsekuensi yang tidak dinginkan karena pihak yang tidak diketahui memanfaatkan kerentanan di dalamnya untuk mengakses beban kerja Anda.
- Anda melakukan patch pada lingkungan pengembangan secara otomatis tanpa memberi tahu pengembang. Anda menerima beberapa keluhan dari pengembang bahwa lingkungan mereka berhenti beroperasi sesuai dengan yang diharapkan.
- Anda belum menerapkan patch pada perangkat lunak komersial siap pakai di instans tetap. Ketika Anda mengalami masalah pada perangkat lunak dan menghubungi vendornya, Anda diberi tahu bahwa versi tersebut tidak didukung dan Anda harus melakukan patch pada tingkat tertentu untuk menerima bantuan.
- Patch yang baru-baru ini dirilis untuk perangkat lunak enkripsi yang Anda gunakan memiliki peningkatan kinerja yang signifikan. Sistem Anda yang tidak di-patch tetap memiliki masalah kinerja akibat tidak dilakukannya patching.
- Anda diberi tahu tentang kerentanan zero-day yang memerlukan perbaikan darurat dan Anda harus menerapkan patch pada semua lingkungan Anda secara manual.

Manfaat menjalankan praktik terbaik ini: Dengan menjalankan proses manajemen patch, termasuk kriteria Anda untuk patching dan metodologi untuk distribusi ke seluruh lingkungan Anda, Anda dapat

menskalakan dan melaporkan tingkat patch. Ini memberikan jaminan seputar patching keamanan dan memastikan visibilitas yang jelas tentang status perbaikan yang diketahui sedang dilakukan. Hal ini mendorong adopsi fitur dan kemampuan yang diinginkan, penyingkiran masalah secara cepat, dan kepatuhan yang berkelanjutan terhadap tata kelola. Implementasikan sistem manajemen dan otomatisasi untuk mengurangi tingkat upaya untuk men-deploy patch dan mengurangi kesalahan yang disebabkan oleh proses manual.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

Lakukan patch pada sistem untuk menyelesaikan masalah, untuk mendapatkan fitur atau kemampuan yang diinginkan, dan untuk tetap patuh terhadap kebijakan tata kelola serta persyaratan dukungan vendor. Pada sistem tetap, deploy dengan rangkaian patch yang sesuai untuk mencapai hasil yang diinginkan. Otomatiskan mekanisme manajemen patch untuk mengurangi waktu yang telah berlalu untuk melakukan patch, untuk mencegah kesalahan yang disebabkan oleh proses manual, dan mengurangi upaya dalam melakukan patch.

Langkah implementasi

Untuk Amazon EC2 Image Builder:

1. Menggunakan Amazon EC2 Image Builder, tentukan detail pipeline:
    a. Buat pipeline image dan beri nama
    b. Tentukan jadwal pipeline dan zona waktu
    c. Konfigurasikan dependensi apa pun
2. Pilih resep:
    a. Pilih resep yang sudah ada atau buat resep baru
    b. Pilih jenis image
    c. Beri nama dan versi resep Anda
    d. Pilih image dasar Anda
    e. Tambahkan komponen build dan tambahkan ke registri target
3. Opsional - tentukan konfigurasi infrastruktur Anda.
4. Opsional - tentukan pengaturan konfigurasi.
5. Tinjau pengaturan.
6. Pertahankan kebersihan resep secara teratur.

Untuk Systems Manager Patch Manager:

1. Buat dasar patch.
2. Pilih metode operasi patching.
3. Aktifkan pelaporan dan pemindaian kepatuhan.

Sumber daya

Praktik terbaik terkait:

- OPS06-BP04 Mengotomatiskan pengujian dan pengembalian (rollback)

Dokumen terkait:

- Apa itu Amazon EC2 Image Builder
- Buat pipeline image menggunakan Amazon EC2 Image Builder
- Buat pipeline image kontainer
- AWS Systems Manager Patch Manager
- Bekerja dengan Patch Manager
- Bekerja dengan laporan kepatuhan patch
- Alat Developer AWS

Video terkait:

- CI/CD untuk Aplikasi Nirserver di AWS
- Mendesain dengan Mempertimbangkan Operasional

Contoh terkait:

- Well-Architected Labs - Manajemen Inventaris dan Patch
- Tutorial AWS Systems Manager Patch Manager

OPS05-BP06 Membagikan standar desain

Bagikan praktik terbaik kepada seluruh tim untuk meningkatkan kesadaran dan memaksimalkan manfaat dari upaya pengembangan. Dokumentasikan dan jaga agar hal ini selalu mutakhir seiring

evolusi arsitektur Anda. Jika standar bersama telah diterapkan di dalam organisasi Anda, tersedianya mekanisme sangat penting untuk meminta penambahan, perubahan, dan pengecualian terhadap standar. Tanpa opsi ini, standar akan menjadi penghambat inovasi.

Hasil yang diinginkan: Standar desain dibagikan ke semua tim dalam organisasi Anda. Standar ini didokumentasi dan dijaga agar selalu mutakhir seiring perkembangan praktik terbaik.

Antipola umum:

- Dua tim pengembangan masing-masing telah membuat layanan autentikasi pengguna. Pengguna Anda harus mempertahankan rangkaian kredensial terpisah untuk setiap bagian sistem yang ingin diakses.
- Setiap tim mengelola infrastruktur mereka sendiri. Persyaratan kepatuhan baru memaksakan perubahan pada infrastruktur Anda dan setiap tim mengimplementasikannya dengan cara yang berbeda.

Manfaat menjalankan praktik terbaik ini: Penggunaan standar bersama mendukung adopsi praktik terbaik dan memaksimalkan manfaat upaya pengembangan. Pendokumentasian dan pembaruan standar desain membuat organisasi Anda selalu mengikuti praktik terbaik dan persyaratan kepatuhan serta keamanan yang mutakhir.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Berbagi praktik terbaik yang ada, standar desain, daftar periksa, prosedur operasi, panduan, dan persyaratan tata kelola dengan semua tim. Miliki prosedur untuk meminta perubahan, penambahan, dan pengecualian standar desain untuk mendukung peningkatan dan inovasi. Buat tim mengetahui tentang konten yang dipublikasikan. Miliki mekanisme untuk menjaga agar standar desain selalu mutakhir seiring kemunculan praktik terbaik baru.

Contoh pelanggan

AnyCompany Retail memiliki tim arsitektur lintas fungsi yang membuat pola arsitektur perangkat lunak. Tim ini membangun arsitektur dengan kepatuhan dan tata kelola bawaan. Tim yang mengadopsi standar bersama ini mendapatkan manfaat dari memiliki kepatuhan dan tata kelola bawaan. Mereka dapat membangun di atas standar desain dengan cepat. Tim arsitektur mengadakan rapat setiap kuartal untuk mengevaluasi pola arsitektur dan memperbaruinya jika perlu.

Langkah implementasi

1. Identifikasikan tim lintas fungsi yang memegang kepemilikan atas pengembangan dan pembaruan standar desain. Tim ini harus bekerja sama dengan pemangku kepentingan di seluruh organisasi Anda untuk mengembangkan standar desain, standar operasi, daftar periksa, panduan, dan persyaratan tata kelola. Dokumentasikan standar desain dan bagikan dalam organisasi Anda.
   a. AWS Service Catalog dapat digunakan untuk membuat portofolio yang mewakili standar desain menggunakan infrastruktur sebagai kode. Anda dapat berbagi portofolio dengan semua akun.
2. Miliki mekanisme untuk menjaga agar standar desain selalu mutakhir seiring teridentifikasinya praktik terbaik baru.
3. Jika standar desain diterapkan secara terpusat, miliki proses untuk meminta perubahan, pembaruan, dan pengecualian.

Tingkat upaya untuk rencana implementasi: Sedang. Untuk mengembangkan proses untuk membuat dan berbagi standar desain mungkin diperlukan kerja sama dan koordinasi dengan para pemangku kepentingan di seluruh organisasi Anda.

Sumber daya

Praktik terbaik terkait:

- OPS01-BP03 Evaluasi persyaratan tata kelola - Persyaratan tata kelola memengaruhi standar desain.
- OPS01-BP04 Evaluasi persyaratan kepatuhan - Kepatuhan adalah input penting dalam membuat standar desain.
- OPS07-BP02 Memastikan peninjauan yang konsisten terkait kesiapan operasional - Daftar periksa kesiapan operasional merupakan mekanisme untuk mengimplementasikan standar desain ketika mendesain beban kerja Anda.
- OPS11-BP01 Miliki proses untuk peningkatan berkelanjutan - Memperbarui standar desain merupakan bagian dari peningkatan berkelanjutan.
- OPS11-BP04 Menjalankan manajemen pengetahuan - Sebagai bagian dari praktik manajemen pengetahuan Anda, dokumentasikan dan bagikan standar desain.

Dokumen terkait:

- Otomatiskan AWS Backup dengan AWS Service Catalog

- Akun AWS Service Catalog yang Ditingkatkan Pabrik
- Bagaimana Expedia Group membangun penawaran Basis Data sebagai Layanan (DBaaS) menggunakan AWS Service Catalog
- Mempertahankan visibilitas tentang penggunaan pola arsitektur cloud
- Menyederhanakan pembagian portofolio AWS Service Catalog Anda dalam pengaturan AWS Organizations

Video terkait:

- AWS Service Catalog – Memulai
- AWS re:Invent 2020: Mengelola portofolio AWS Service Catalog Anda layaknya ahli

Contoh terkait:

- Arsitektur Referensi AWS Service Catalog
- Lokakarya AWS Service Catalog

Layanan terkait:

- AWS Service Catalog

OPS05-BP07 Mengimplementasikan praktik untuk meningkatkan kualitas kode

Implementasikan praktik untuk meningkatkan kualitas kode dan meminimalkan kecacatan. Beberapa contohnya termasuk, pengembangan yang didorong pengujian, peninjauan kode, pengadopsian standar, dan pemrograman berpasangan. Sertakan praktik-praktik ini ke dalam integrasi berkelanjutan dan proses penyampaian hasil Anda.

Hasil yang diinginkan: Organisasi Anda menggunakan praktik terbaik seperti peninjauan kode atau pemrograman berpasangan untuk meningkatkan kualitas kode. Developer dan operator mengadopsi praktik terbaik dalam kualitas kode sebagai bagian dari siklus hidup pengembangan perangkat lunak.

Antipola umum:

- Anda mempercayakan kode ke cabang utama aplikasi tanpa peninjauan kode. Perubahan otomatis melakukan deployment ke produksi dan menyebabkan penghentian produksi.

- Aplikasi baru dikembangkan tanpa pengujian integrasi, unit, atau menyeluruh. Tidak ada cara untuk menguji aplikasi sebelum deployment.
- Tim Anda membuat perubahan manual pada produksi untuk mengatasi kecacatan. Perubahan tidak melalui proses pengujian atau peninjauan kode dan tidak ditangkap atau dicatat melalui proses penyampaian hasil dan integrasi berkelanjutan.

Manfaat menjalankan praktik terbaik ini: Dengan mengadopsi praktik untuk meningkatkan kualitas kode, Anda dapat membantu meminimalkan masalah yang terjadi di produksi. Kualitas kode akan meningkat dengan penggunaan praktik terbaik seperti pemrograman berpasangan dan peninjauan kode.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Implementasikan praktik untuk meningkatkan kualitas kode guna meminimalkan kecacatan sebelum dilakukan deployment terhadapnya. Gunakan praktik seperti pengembangan yang didorong pengujian, peninjauan kode, dan pemrograman berpasangan untuk meningkatkan kualitas pengembangan Anda.

Contoh pelanggan

AnyCompany Retail mengadopsi beberapa praktik untuk meningkatkan kualitas kode. Mereka telah mengadopsi pengembangan yang didorong pengujian sebagai standar untuk menulis aplikasi. Untuk beberapa fitur baru, developer mereka memasangkan program menjadi satu selama sprint. Setiap permintaan penarikan akan melewati peninjauan kode oleh developer senior sebelum diintegrasikan dan dilakukan deployment.

Langkah implementasi

1. Adopsi praktik kualitas kode seperti pengembangan yang didorong pengujian, peninjauan kode, dan pemrograman berpasangan ke dalam proses penyampaian hasil dan integrasi berkelanjutan Anda. Gunakan teknik-teknik ini untuk meningkatkan kualitas perangkat lunak.
   a. Amazon CodeGuru Reviewer dapat memberikan rekomendasi pemrograman untuk kode Python dan Java menggunakan machine learning.
   b. Anda dapat membuat lingkungan pengembangan bersama dengan AWS Cloud9 di mana Anda dapat berkolaborasi dalam mengembangkan kode.

Tingkat upaya untuk rencana implementasi: Sedang. Ada banyak cara untuk mengimplementasikan praktik terbaik ini, tetapi membuat organisasi mau mengadopsinya mungkin merupakan hal yang sulit.

Sumber daya

Praktik terbaik terkait:

- OPS05-BP06 Membagikan standar desain - Anda dapat berbagi standar desain sebagai bagian dari praktik kualitas kode Anda.

Dokumen terkait:

- Panduan Perangkat Lunak Tangkas
- Pipeline CI/CD adalah pemandu utama rilisan saya
- Otomatiskan peninjauan kode dengan Amazon CodeGuru Reviewer
- Adopsi pendekatan pengembangan yang didorong pengujian
- Bagaimana DevFactory membangun aplikasi yang lebih baik dengan Amazon CodeGuru
- Tentang Pemrograman Berpasangan
- RENGA Inc. mengotomatiskan peninjauan kode dengan Amazon CodeGuru
- Seni Pengembangan Tangkas: Pengembangan yang Didorong Pengujian
- Mengapa peninjauan kode itu penting (dan sesungguhnya menghemat waktu!)

Video terkait:

- AWS re:Invent 2020: Peningkatan berkelanjutan kualitas kode dengan Amazon CodeGuru
- AWS Summit ANZ 2021 - Mendorong strategi yang mengutamakan pengujian dengan CDK dan pengembangan yang didorong pengujian

Layanan terkait:

- Amazon CodeGuru Reviewer
- Amazon CodeGuru Profiler
- AWS Cloud9

OPS05-BP08 Menggunakan beberapa lingkungan

Gunakan beberapa lingkungan untuk bereksperimen, mengembangkan, dan menguji beban kerja Anda. Gunakan tingkat kontrol berjenjang seiring lingkungan mendekati tahap produksi untuk mendapatkan keyakinan bahwa beban kerja Anda beroperasi sesuai keinginan ketika di-deploy.

Hasil yang diinginkan: Anda memiliki beberapa lingkungan yang mencerminkan kebutuhan kepatuhan dan tata kelola Anda. Anda menguji dan mempromosikan kode melalui lingkungan di jalur Anda menuju produksi.

Antipola umum:

- Anda sedang melakukan pengembangan di sebuah lingkungan pengembangan bersama dan developer lain menimpa perubahan kode Anda.
- Kontrol keamanan terbatas di lingkungan pengembangan bersama Anda melarang Anda melakukan eksperimen dengan layanan dan fitur baru.
- Anda melakukan pengujian beban pada sistem produksi Anda dan menyebabkan pemadaman untuk pengguna Anda.
- Kesalahan fatal yang menyebabkan hilangnya data terjadi di produksi. Di lingkungan produksi, Anda mencoba membuat ulang kondisi yang menyebabkan data hilang tersebut sehingga Anda dapat mengidentifikasi bagaimana hal tersebut terjadi dan mencegahnya agar tidak terjadi lagi. Untuk mencegah kejadian hilang data lainnya selama pengujian, Anda terpaksa menjadikan aplikasi tidak tersedia untuk pengguna.
- Anda mengoperasikan layanan multi-tenant dan tidak dapat mendukung permintaan lingkungan khusus yang diajukan pelanggan.
- Anda mungkin tidak selalu melakukan pengujian, tetapi ketika Anda menguji, Anda melakukannya di lingkungan produksi.
- Anda percaya bahwa dengan satu lingkungan tunggal, cakupan dampak perubahan hanya terjadi di dalam lingkungan tersebut.

Manfaat menjalankan praktik terbaik ini: Anda dapat mendukung beberapa lingkungan pengembangan, pengujian, dan produksi secara serentak tanpa menciptakan konflik antar developer atau komunitas pengguna.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Gunakan beberapa lingkungan dan sediakan lingkungan sandbox developer dengan kontrol minim untuk membantu eksperimen. Sediakan lingkungan pengembangan individu untuk membantu kerja secara paralel, sehingga ketangkasan pengembangan meningkat. Implementasikan kontrol yang lebih kuat di lingkungan ketika mendekati produksi agar developer dapat berinovasi. Gunakan infrastruktur sebagai kode dan sistem manajemen konfigurasi untuk men-deploy lingkungan yang dikonfigurasi sesuai dengan kontrol yang ada di dalam produksi guna memastikan sistem beroperasi sesuai keinginan saat di-deploy. Saat lingkungan tidak digunakan, nonaktifkan untuk menghindari biaya terkait sumber daya tidak terpakai (misalnya sistem pengembangan di malam hari dan di akhir pekan). Deploy lingkungan setara produksi saat melakukan pengujian beban untuk meningkatkan hasil yang valid.

Sumber daya

Dokumen terkait:

- Penjadwal Instans di AWS
- Apa itu AWS CloudFormation?

OPS05-BP09 Membuat perubahan yang sering, kecil, dan dapat dikembalikan

Gunakan perubahan yang sering, kecil, dan dapat dikembalikan untuk mengurangi cakupan perubahan. Ketika digunakan bersamaan dengan sistem manajemen perubahan, sistem manajemen konfigurasi, dan sistem build serta pengiriman, perubahan yang sering, kecil, dan dapat dikembalikan dapat mengurangi cakupan dan dampak perubahan. Hal ini menghasilkan pemecahan masalah yang lebih efektif dan remediasi yang lebih cepat dengan opsi untuk membatalkan perubahan.

Antipola umum:

- Anda men-deploy versi baru aplikasi Anda setiap tiga bulan sekali dengan periode perubahan yang mengharuskan layanan inti dinonaktifkan.
- Anda sering membuat perubahan pada skema basis data Anda tanpa melacak perubahan dalam sistem manajemen Anda.
- Anda melakukan pembaruan manual di tempat, menimpa instalasi dan konfigurasi yang ada, dan tidak memiliki rencana roll-back yang jelas.

Manfaat menjalankan praktik terbaik ini: Upaya pengembangan menjadi lebih cepat dengan sering men-deploy perubahan kecil. Ketika berukuran kecil, perubahan jauh lebih mudah diidentifikasi jika terdapat konsekuensi yang tidak diinginkan, serta lebih mudah untuk dikembalikan. Ketika perubahan dapat dikembalikan, risiko implementasi perubahan menjadi lebih kecil karena pemulihannya lebih mudah. Proses perubahan memiliki risiko yang lebih kecil dan dampak kegagalan perubahan menjadi berkurang.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Gunakan perubahan yang sering, kecil, dan dapat dikembalikan untuk mengurangi cakupan dan dampak perubahan. Hal ini memudahkan pemecahan masalah, membantu remediasi yang lebih cepat, dan menyediakan opsi untuk membatalkan perubahan. Hal ini juga meningkatkan rasio nilai yang dapat Anda berikan ke bisnis.

Sumber daya

Praktik terbaik terkait:

- OPS05-BP03 Menggunakan sistem manajemen konfigurasi
- OPS05-BP04 Menggunakan sistem manajemen build dan deployment
- OPS06-BP04 Mengotomatiskan pengujian dan pengembalian (rollback)

Dokumen terkait:

- Mengimplementasikan Layanan Mikro di AWS
- Layanan Mikro - Observabilitas

OPS05-BP10 Mengotomatiskan integrasi dan deployment sepenuhnya

Otomatiskan build, deployment, dan pengujian beban kerja. Hal ini mengurangi kesalahan yang disebabkan oleh proses manual, dan mengurangi upaya untuk melakukan deployment perubahan.

Terapkan metadata menggunakan Tag Sumber Daya dan AWS Resource Groups mengikuti strategi pemberian tag yang konsisten untuk membantu identifikasi sumber daya Anda. Berikan tag pada sumber daya Anda untuk pengaturan, akuntansi biaya, kontrol akses, dan penargetan pelaksanaan aktivitas operasi yang diotomatiskan.

Hasil yang diinginkan: Developer menggunakan alat untuk mengirimkan kode dan mencapai produksi. Developer tidak harus masuk ke dalam Konsol Manajemen AWS untuk memberikan pembaruan. Terdapat jejak audit penuh untuk perubahan dan konfigurasi, sehingga memenuhi kebutuhan tata kelola dan kepatuhan. Proses dapat diulang dan distandardisasi di seluruh tim. Developer bebas memusatkan perhatian pada pengembangan dan pendorongan kode, sehingga meningkatkan produktivitas.

Antipola umum:

- Pada hari Jumat, Anda selesai menulis kode baru untuk cabang fitur Anda. Pada hari Senin, setelah menjalankan skrip pengujian kualitas kode dan setiap skrip pengujian unit, Anda mendaftarkan kode untuk rilisan terjadwal berikutnya.
- Anda ditugaskan untuk membuat kode perbaikan untuk sebuah masalah besar yang memengaruhi banyak pelanggan di tahap produksi. Setelah menguji perbaikan tersebut, Anda melakukan commit kode Anda dan mengirimkan manajemen perubahan melalui email untuk meminta persetujuan deployment ke produksi.
- Sebagai developer, Anda masuk ke Konsol Manajemen AWS untuk membuat lingkungan pengembangan baru menggunakan metode dan sistem non-standar.

Manfaat menjalankan praktik terbaik ini: Dengan mengimplementasikan sistem manajemen build dan deployment otomatis, Anda mengurangi kesalahan yang disebabkan proses manual dan mengurangi upaya untuk melakukan deployment perubahan yang membantu anggota tim Anda berkonsentrasi menghadirkan nilai bisnis. Anda meningkatkan kecepatan pengiriman selama proses menuju produksi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Anda menggunakan sistem manajemen build dan deployment untuk melacak dan mengimplementasikan perubahan, mengurangi kesalahan yang disebabkan oleh proses manual, dan mengurangi upaya. Otomatiskan sepenuhnya pipeline integrasi dan deployment dari check-in kode hingga build, pengujian, deployment, dan validasi. Hal ini mengurangi waktu tunggu, mendorong peningkatan frekuensi perubahan, mengurangi tingkat upaya, meningkatkan kecepatan masuk pasar, menghasilkan peningkatan produktivitas, dan meningkatkan keamanan kode Anda selama proses Anda menuju produksi.

Sumber daya

Praktik terbaik terkait:

- OPS05-BP03 Menggunakan sistem manajemen konfigurasi
- OPS05-BP04 Menggunakan sistem manajemen build dan deployment

Dokumen terkait:

- Apa itu AWS CodeBuild?
- Apa itu AWS CodeDeploy?

Video terkait:

- AWS re\:Invent 2022 - AWS Praktik terbaik Well-Architected untuk DevOps di AWS

OPS 6. Bagaimana cara memitigasi risiko deployment?

Adopsi pendekatan yang memberikan umpan balik cepat atas kualitas dan mencapai pemulihan cepat dari perubahan yang tidak memiliki hasil yang tidak diinginkan. Menggunakan praktik-praktik ini memitigasi dampak masalah akibat deployment perubahan.

Praktik terbaik

- OPS06-BP01 Antisipasikan perubahan yang tidak berhasil
- OPS06-BP02 Menguji deployment
- OPS06-BP03 Menggunakan strategi deployment yang aman
- OPS06-BP04 Mengotomatiskan pengujian dan pengembalian (rollback)

OPS06-BP01 Antisipasikan perubahan yang tidak berhasil

Rencanakan untuk kembali ke keadaan yang diketahui pasti baik, atau perbaiki di lingkungan produksi jika deployment menyebabkan hasil yang tidak diinginkan. Adanya kebijakan untuk menetapkan rencana semacam ini bermanfaat bagi semua tim dalam mengembangkan strategi untuk pulih dari perubahan yang gagal. Beberapa contoh strategi adalah langkah deployment dan rollback, kebijakan perubahan, penanda fitur, pemisahan lalu lintas, dan pergeseran lalu lintas. Rilis tunggal dapat mencakup beberapa perubahan komponen yang terkait. Strategi harus memberikan kemampuan untuk bertahan atau pulih dari kegagalan perubahan komponen apa pun.

Hasil yang diinginkan: Anda telah menyiapkan rencana pemulihan yang mendetail untuk perubahan Anda apabila perubahan tersebut tidak berhasil. Selain itu, Anda telah mengurangi ukuran rilis untuk meminimalkan dampak potensial pada komponen beban kerja lainnya. Hasilnya, Anda telah mengurangi dampak bisnis Anda dengan mempersingkat potensi waktu henti yang diakibatkan oleh perubahan yang gagal dan meningkatkan fleksibilitas serta efisiensi waktu pemulihan.

Antipola umum:

- Anda melakukan deployment dan aplikasi Anda menjadi tidak stabil tetapi tampaknya ada pengguna aktif di sistem. Anda harus memutuskan apakah akan mengembalikan perubahan yang akan berdampak pada pengguna aktif atau menunggu untuk mengembalikan perubahan karena tahu bagaimana pun juga pengguna dapat terkena dampaknya.
- Setelah membuat perubahan rutin, lingkungan baru Anda dapat diakses tetapi salah satu subnet Anda menjadi tidak dapat dijangkau. Anda harus memutuskan apakah akan mengembalikan semuanya atau mencoba memperbaiki subnet yang tidak dapat diakses tersebut. Sementara Anda sedang memutuskan hal ini, subnet tersebut tetap tidak dapat dijangkau.
- Sistem Anda tidak dirancang dapat diperbarui dengan rilis-rilis yang lebih kecil. Akibatnya, Anda mengalami kesulitan dalam membatalkan perubahan massal tersebut selama deployment yang gagal.
- Anda tidak menggunakan infrastruktur sebagai kode (IaC) dan Anda melakukan pembaruan manual pada infrastruktur Anda sehingga mengakibatkan konfigurasi yang tidak diinginkan. Anda tidak dapat melacak dan membatalkan perubahan manual secara efektif.
- Karena Anda belum mengukur peningkatan frekuensi deployment Anda, tim Anda kesulitan mengurangi ukuran perubahan mereka dan meningkatkan rencana rollback mereka untuk setiap perubahan, yang berimbas pada risiko yang lebih besar dan tingkat kegagalan yang meningkat.
- Anda tidak mengukur total durasi pemadaman yang disebabkan oleh perubahan yang tidak berhasil. Tim Anda tidak dapat memprioritaskan dan meningkatkan proses deployment serta efektivitas rencana pemulihannya.

Manfaat menjalankan praktik terbaik ini: Memiliki rencana untuk pulih dari perubahan yang gagal meminimalkan rata-rata waktu untuk pulih (MTTR) dan mengurangi dampak bisnis Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Kebijakan dan praktik yang konsisten serta terdokumentasi yang diadopsi oleh tim rilis memungkinkan organisasi untuk merencanakan apa yang harus terjadi apabila terjadi kegagalan perubahan. Kebijakan harus memungkinkan perbaikan maju (fix forward) dalam keadaan tertentu. Dalam situasi apa pun, rencana perbaikan maju atau rollback harus didokumentasikan dan diuji dengan baik sebelum deployment ke produksi langsung sehingga waktu yang diperlukan untuk mengembalikan perubahan dapat diminimalkan.

Langkah implementasi

1. Dokumentasikan kebijakan yang mengharuskan tim memiliki rencana efektif untuk mengembalikan perubahan dalam periode tertentu.
   a. Kebijakan harus menentukan kapan situasi perbaikan maju diperbolehkan.
   b. Rencana rollback yang terdokumentasi harus dapat diakses oleh semua pihak yang terlibat.
   c. Tentukan persyaratan untuk rollback (misalnya, ketika ternyata ada perubahan tidak sah yang telah di-deploy).
2. Analisis tingkat dampak semua perubahan yang berkaitan dengan setiap komponen beban kerja.
   a. Biarkan perubahan berulang untuk distandardisasi, dijadikan templat, dan diotorisasi di awal jika perubahan tersebut mengikuti alur kerja konsisten yang memberlakukan kebijakan perubahan.
   b. Kurangi potensi dampak perubahan apa pun dengan menjadikan ukuran perubahan lebih kecil sehingga pemulihan membutuhkan waktu yang lebih singkat dan menyebabkan lebih sedikit dampak bisnis.
   c. Pastikan prosedur rollback mengembalikan kode ke keadaan yang pasti baik untuk menghindari insiden jika memungkinkan.
3. Integrasikan alat dan alur kerja untuk menegakkan kebijakan Anda secara terprogram.
4. Buat agar data tentang perubahan dapat dilihat oleh pemilik beban kerja lain untuk meningkatkan kecepatan diagnosis perubahan yang gagal yang tidak dapat dibatalkan.
   a. Ukur keberhasilan praktik ini menggunakan data perubahan yang terlihat dan identifikasi peningkatan iteratif.
5. Gunakan alat pemantauan untuk memverifikasi keberhasilan atau kegagalan deployment untuk mempercepat pengambilan keputusan saat melakukan rollback.
6. Ukur durasi pemadaman Anda selama perubahan yang gagal untuk terus meningkatkan kualitas rencana pemulihan Anda.

Tingkat upaya untuk rencana implementasi: Sedang

Sumber daya

Praktik terbaik terkait:

- OPS06-BP04 Mengotomatiskan pengujian dan pengembalian (rollback)

Dokumen terkait:

- AWS Builders Library | Memastikan Keamanan Rollback Selama Deployment
- AWS Laporan Resmi | Manajemen Perubahan di Cloud

Video terkait:

- re:Invent 2019 | Pendekatan Amazon untuk deployment ketersediaan tinggi

OPS06-BP02 Menguji deployment

Uji prosedur rilis dalam tahap praproduksi dengan menggunakan konfigurasi deployment, kontrol keamanan, langkah, dan prosedur yang sama seperti dalam tahap produksi. Lakukan validasi bahwa semua langkah yang di-deploy selesai sesuai harapan, seperti dengan memeriksa file, konfigurasi, dan layanan. Uji lebih lanjut semua perubahan dengan pengujian fungsional, integrasi, dan beban, beserta pemantauan apa pun seperti pemeriksaan kondisi. Dengan melakukan pengujian ini, Anda dapat mengidentifikasi masalah deployment lebih awal dengan peluang untuk merencanakan dan menanggulanginya sebelum produksi.

Anda dapat membuat lingkungan paralel sementara untuk menguji setiap perubahan. Otomatiskan deployment lingkungan pengujian menggunakan infrastruktur sebagai kode (IaC) untuk membantu mengurangi jumlah pekerjaan yang terlibat dan memastikan stabilitas, konsistensi, dan pengiriman fitur yang lebih cepat.

Hasil yang diinginkan: Organisasi Anda mengadopsi budaya pengembangan berbasis pengujian yang mencakup pengujian deployment. Ini memastikan tim fokus menghadirkan nilai bisnis, bukan mengelola rilis. Tim terlibat sejak dini setelah identifikasi risiko deployment untuk menentukan arah mitigasi yang tepat.

Antipola umum:

- Selama rilis produksi, deployment yang belum teruji sering menyebabkan masalah yang memerlukan penyelesaian dan eskalasi.
- Rilis Anda berisi infrastruktur sebagai kode (IaC) yang memperbarui sumber daya yang ada. Anda tidak yakin apakah IaC berjalan dengan sukses atau menyebabkan dampak pada sumber daya.
- Anda men-deploy sebuah fitur baru ke aplikasi Anda. Fitur tersebut tidak berfungsi sesuai keinginan dan masalah ini baru dapat diketahui setelah dilaporkan oleh pengguna yang terdampak.
- Anda memperbarui sertifikat Anda. Anda tidak sengaja menginstal sertifikat ke komponen yang salah, yang akhirnya tidak terdeteksi dan berdampak pada pengunjung situs web karena koneksi yang aman ke situs web tidak dapat dibuat.

Manfaat menjalankan praktik terbaik ini: Pengujian ekstensif selama tahap praproduksi dalam prosedur deployment serta perubahan yang dimunculkannya dapat meminimalkan potensi dampak terhadap produksi yang disebabkan oleh langkah-langkah deployment. Hal ini meningkatkan kepercayaan diri selama rilis produksi dan meminimalkan dukungan operasional tanpa memperlambat penyampaian perubahan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Menguji proses deployment sama pentingnya dengan menguji perubahan yang dihasilkan dari deployment Anda. Hal ini dapat dicapai dengan menguji langkah-langkah deployment Anda di lingkungan praproduksi yang semaksimal mungkin mencerminkan produksi. Masalah umum, seperti langkah deployment yang tidak lengkap atau salah, atau kesalahan konfigurasi, dapat terdeteksi sebelum masuk ke tahap produksi. Selain itu, Anda dapat menguji langkah-langkah pemulihan Anda.

Contoh pelanggan

Sebagai bagian dari pipeline integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD), AnyCompany Retail melakukan langkah-langkah yang ditentukan yang diperlukan untuk merilis pembaruan infrastruktur dan perangkat lunak bagi pelanggannya di dalam lingkungan mirip produksi. Pipeline tersebut terdiri dari prapemeriksaan untuk mendeteksi penyimpangan (mendeteksi perubahan pada sumber daya yang dilakukan di luar IaC Anda) di dalam sumber daya sebelum deployment, serta memvalidasi tindakan yang dilakukan IaC setelah inisiasi. Tahap ini memvalidasi langkah-langkah deployment, seperti memverifikasi bahwa file dan konfigurasi tertentu sudah siap dan layanan sudah dalam status berjalan serta merespons dengan benar pemeriksaan kondisi pada host lokal sebelum didaftarkan ulang dengan penyeimbang beban. Selain itu, semua perubahan

menandai sejumlah pengujian otomatis, seperti pengujian fungsional, keamanan, regresi, integrasi, dan beban.

Langkah implementasi

1. Lakukan pemeriksaan prainstal pada produksi untuk mencerminkan lingkungan praproduksi.
   a. Gunakan deteksi penyimpangan untuk mendeteksi apabila sumber daya telah diubah di luar CloudFormation.
   b. Gunakan set perubahan untuk memvalidasi bahwa maksud pembaruan tumpukan sesuai dengan tindakan yang dilakukan oleh CloudFormation saat rangkaian perubahan dimulai.
2. Ini memicu langkah persetujuan manual di AWS CodePipeline untuk mengotorisasi deployment ke lingkungan praproduksi.
3. Gunakan konfigurasi deployment seperti file AWS CodeDeploy AppSpec untuk menentukan langkah deployment dan validasi.
4. Jika perlu, integrasikan AWS CodeDeploy dengan layanan AWS lain atau integrasikan AWS CodeDeploy dengan produk dan layanan partner.
5. Pantau deployment menggunakan Amazon CloudWatch, AWS CloudTrail, dan notifikasi peristiwa Amazon SNS.
6. Lakukan pengujian otomatis pasca-deployment, termasuk pengujian fungsional, keamanan, regresi, integrasi, dan beban.
7. Pecahkan masalah deployment.
8. Validasi yang berhasil dari langkah-langkah sebelumnya seharusnya menginisiasi alur kerja persetujuan manual untuk memberikan otorisasi deployment ke produksi.

Tingkat upaya untuk rencana implementasi: Tinggi

Sumber daya

Praktik terbaik terkait:

- OPS05-BP02 Menguji dan memvalidasi perubahan

Dokumen terkait:

- AWS Builders' Library | Mengotomatiskan deployment hands-off yang aman | Menguji Deployment

- AWS Laporan Resmi | Mempraktikkan Integrasi Berkelanjutan dan Pengiriman Berkelanjutan di AWS
- Kisah Apollo - Mesin Deployment Amazon
- Cara menguji dan melakukan debug AWS CodeDeploy secara lokal sebelum mengirimkan kode Anda
- Mengintegrasikan Pengujian Konektivitas Jaringan dengan Deployment Infrastruktur

Video terkait:

- re:Invent 2020 | Menguji perangkat lunak dan sistem di Amazon

Contoh terkait:

- Tutorial | Melakukan Deployment dan layanan Amazon ECS dengan uji validasi

OPS06-BP03 Menggunakan strategi deployment yang aman

Peluncuran produksi yang aman mengontrol aliran perubahan yang bermanfaat dengan tujuan untuk meminimalkan dampak yang dirasakan oleh pelanggan dari perubahan tersebut. Kontrol keselamatan menyediakan mekanisme inspeksi untuk memvalidasi hasil yang diinginkan dan membatasi ruang lingkup dampak dari cacat apa pun yang disebabkan oleh perubahan atau dari kegagalan deployment. Peluncuran yang aman dapat mencakup strategi seperti feature-flag, one-box, bergulir (rilis canary), immutable, pemisahan lalu lintas, dan deployment blue/green.

Hasil yang diinginkan: Organisasi Anda menggunakan sistem integrasi berkelanjutan pengiriman berkelanjutan (CI/CD) yang menyediakan kemampuan untuk mengotomatiskan peluncuran yang aman. Tim diharuskan menggunakan strategi peluncuran aman yang tepat.

Antipola umum:

- Anda melakukan deployment perubahan yang tidak berhasil ke seluruh produksi sekaligus. Akibatnya, semua pelanggan merasakan dampaknya secara bersamaan.
- Cacat akibat deployment serentak ke semua sistem memerlukan rilis darurat. Diperlukan waktu beberapa hari untuk memperbaikinya untuk semua pelanggan.
- Untuk mengelola rilis produksi diperlukan perencanaan dan partisipasi beberapa tim. Hal ini menghambat kemampuan Anda untuk sering memperbarui fitur bagi pelanggan Anda.

- Anda melakukan deployment yang dapat diubah dengan memodifikasi sistem yang sudah ada. Setelah mengetahui bahwa perubahan tidak berhasil, Anda terpaksa memodifikasi sistem lagi untuk memulihkan versi yang lama sehingga memperpanjang waktu pemulihan Anda.

Manfaat menjalankan praktik terbaik ini: Deployment otomatis menyeimbangkan kecepatan peluncuran dengan menghadirkan perubahan yang bermanfaat secara konsisten kepada pelanggan. Pembatasan dampak dapat mencegah kegagalan deployment yang mahal dan memaksimalkan kemampuan tim untuk merespons kegagalan secara efisien.

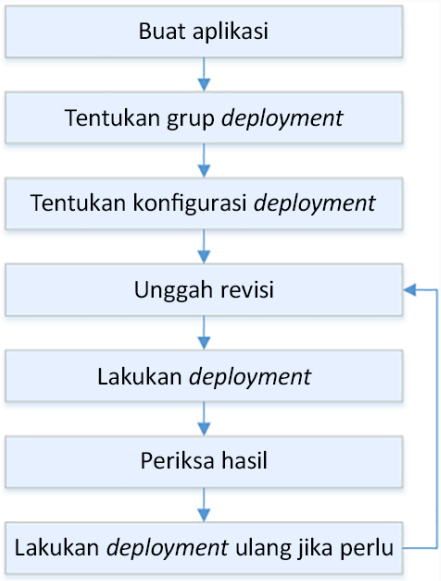
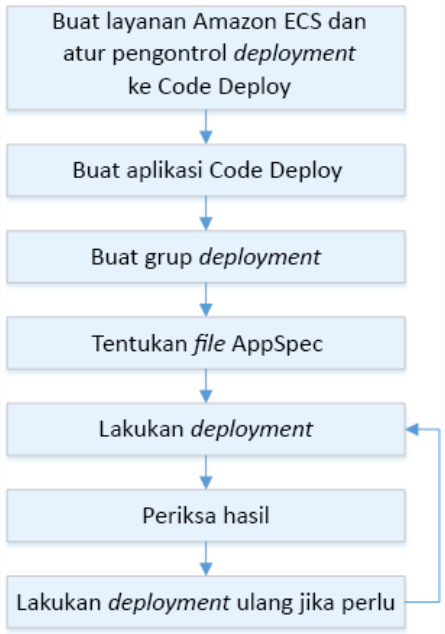
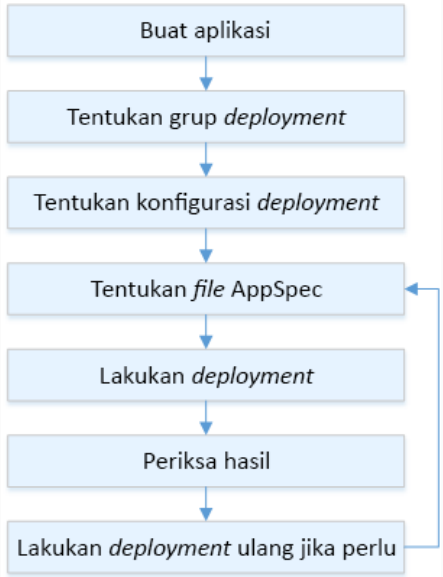
Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Kegagalan pengiriman berkelanjutan dapat menyebabkan berkurangnya ketersediaan layanan dan buruknya pengalaman pelanggan. Untuk memaksimalkan tingkat keberhasilan deployment, terapkan kontrol keamanan dalam proses rilis menyeluruh (end-to-end) untuk meminimalkan kesalahan deployment, dengan tujuan mencapai nol kegagalan deployment.

Contoh pelanggan

AnyCompany Retail memiliki misi untuk mencapai deployment dengan waktu henti yang minim hingga nol, yang berarti tidak ada dampak yang dirasakan oleh penggunanya selama deployment. Untuk mencapainya, perusahaan telah membuat pola deployment (lihat diagram alur kerja berikut), seperti deployment blue/green dan bergulir (rolling). Semua tim mengadopsi satu atau beberapa pola tersebut di dalam pipeline CI/CD mereka.

Alur kerja CodeDeploy untuk Amazon EC2	Alur kerja CodeDeploy untuk Amazon ECS	Alur kerja CodeDeploy untuk Lambda
		

Langkah implementasi

1. Gunakan alur kerja persetujuan untuk memulai urutan langkah peluncuran produksi setelah promosi ke produksi.
2. Gunakan sistem deployment otomatis seperti AWS CodeDeploy. Opsi deployment AWS CodeDeploy meliputi deployment pengganti untuk EC2/On-Premise dan deployment blue/green untuk EC2/On-Premise, AWS Lambda, dan Amazon ECS (lihat diagram alur kerja sebelumnya).
   a. Jika perlu, integrasikan AWS CodeDeploy dengan layanan AWS lain atau integrasikan AWS CodeDeploy dengan produk dan layanan partner.
3. Gunakan deployment blue/green untuk basis data seperti Amazon Aurora dan Amazon RDS.
4. Pantau deployment menggunakan Amazon CloudWatch, AWS CloudTrail, dan notifikasi peristiwa Amazon SNS.
5. Lakukan pengujian otomatis pasca-deployment termasuk pengujian fungsional, keamanan, regresi, integrasi, dan beban apa pun.
6. Pecahkan masalah deployment.

Tingkat upaya untuk rencana implementasi: Sedang

Sumber daya

Praktik terbaik terkait:

- OPS05-BP02 Menguji dan memvalidasi perubahan
- OPS05-BP09 Membuat perubahan yang sering, kecil, dan dapat dikembalikan
- OPS05-BP10 Mengotomatiskan integrasi dan deployment sepenuhnya

Dokumen terkait:

- AWS Builders Library | Mengotomatiskan deployment hands-off yang aman | Deployment produksi
- AWS Builders Library | Pipeline CI/CD saya adalah kapten rilis saya | Rilis produksi otomatis yang aman
- Laporan Resmi AWS | Mempraktikkan Integrasi Berkelanjutan dan Pengiriman Berkelanjutan di AWS | Metode deployment
- AWS CodeDeploy Panduan Pengguna
- Mulai konfigurasi deployment di AWS CodeDeploy
- Konfigurasikan canary API Gateway untuk meluncurkan deployment
- Jenis Deployment Amazon ECS
- Deployment Blue/Green Terkelola Penuh di Amazon Aurora dan Amazon RDS
- Deployment Blue/Green dengan AWS Elastic Beanstalk

Video terkait:

- re:Invent 2020 | Hands-off: Mengotomatiskan pipeline pengiriman berkelanjutan di Amazon
- re:Invent 2019 | Pendekatan deployment ketersediaan tinggi Amazon

Contoh terkait:

- Coba Sampel Deployment Blue/Green di AWS CodeDeploy
- Lokakarya | Membangun pipeline CI/CD untuk deployment canary Lambda menggunakan AWS CDK
- Lokakarya | Deployment Blue/Green dan Canary untuk EKS dan ECS

- Lokakarya | Membangun Pipeline CI/CD Lintas Akun

OPS06-BP04 Mengotomatiskan pengujian dan pengembalian (rollback)

Untuk meningkatkan kecepatan, keandalan, dan keyakinan pada proses deployment Anda, miliki strategi untuk kemampuan pengujian dan rollback otomatis di lingkungan praproduksi dan produksi. Otomatiskan pengujian saat melakukan deployment ke produksi untuk menyimulasikan interaksi manusia dan sistem yang memverifikasi perubahan yang sedang di-deploy. Otomatiskan rollback untuk kembali ke keadaan pasti baik sebelumnya dengan cepat. Rollback harus dimulai secara otomatis pada kondisi yang telah ditentukan di awal seperti ketika hasil perubahan yang Anda inginkan tidak tercapai atau ketika pengujian otomatis mengalami kegagalan. Mengotomatiskan kedua aktivitas ini dapat memperbaiki tingkat keberhasilan untuk deployment Anda, meminimalkan waktu pemulihan, dan mengurangi potensi dampak terhadap bisnis.

Hasil yang diinginkan: Strategi pengujian dan rollback otomatis Anda diintegrasikan ke dalam pipeline integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD) Anda. Pemantauan Anda dapat memvalidasi berdasarkan kriteria keberhasilan Anda dan memulai rollback otomatis setelah kegagalan. Hal ini meminimalkan dampak apa pun terhadap pelanggan dan pengguna akhir. Misalnya, ketika semua hasil pengujian telah terpenuhi, Anda meneruskan kode Anda ke lingkungan produksi tempat pengujian regresi otomatis dimulai, dengan memanfaatkan kasus pengujian yang sama. Jika hasil pengujian regresi tidak sesuai dengan harapan, maka rollback otomatis akan dimulai dalam alur kerja pipeline.

Antipola umum:

- Sistem Anda tidak dirancang dapat diperbarui dengan rilis-rilis yang lebih kecil. Akibatnya, Anda mengalami kesulitan dalam membatalkan perubahan massal tersebut selama deployment yang gagal.
- Proses deployment Anda terdiri dari serangkaian langkah manual. Setelah melakukan deployment perubahan ke beban kerja, Anda memulai pengujian pasca-deployment. Setelah pengujian, Anda menyadari bahwa beban kerja Anda tidak dapat dioperasikan dan koneksi pelanggan terputus. Kemudian Anda mulai melakukan rollback ke versi sebelumnya. Semua langkah manual ini menghambat pemulihan sistem secara keseluruhan dan menyebabkan dampak yang berkepanjangan terhadap pelanggan Anda.
- Anda menghabiskan waktu mengembangkan kasus pengujian otomatis untuk fungsionalitas yang tidak sering digunakan dalam aplikasi Anda, sehingga memperkecil laba atas investasi dalam kemampuan pengujian otomatis Anda.

- Rilis Anda terdiri dari aplikasi, infrastruktur, patch, dan pembaruan konfigurasi yang tidak bergantung satu sama lain. Namun, Anda memiliki satu pipeline CI/CD yang mengirimkan semua perubahan sekaligus. Kegagalan pada satu komponen memaksa Anda untuk mengembalikan semua perubahan, menjadikan rollback Anda kompleks dan tidak efisien.
- Tim Anda menyelesaikan tugas pengodean dalam sprint satu dan memulai tugas sprint dua, tetapi rencana Anda tidak menyertakan pengujian sampai sprint tiga. Akibatnya, pengujian otomatis mengungkap cacat dari sprint satu yang harus diselesaikan sebelum pengujian kiriman sprint dua dapat dimulai dan seluruh rilis menjadi tertunda, sehingga menurunkan nilai pengujian otomatis Anda.
- Kasus pengujian regresi otomatis Anda untuk rilis produksi sudah selesai, tetapi Anda tidak memantau kondisi beban kerja. Karena Anda tidak memiliki visibilitas apakah layanan telah dimulai ulang atau belum, Anda tidak yakin apakah rollback diperlukan atau rollback sudah terjadi.

Manfaat menjalankan praktik terbaik ini: Pengujian otomatis meningkatkan transparansi proses pengujian Anda dan kemampuan Anda untuk mencakup lebih banyak fitur dalam periode waktu yang lebih singkat. Dengan menguji dan memvalidasi perubahan dalam produksi, Anda dapat segera mengidentifikasi masalah. Peningkatan konsistensi dengan alat pengujian otomatis memungkinkan deteksi cacat yang lebih baik. Dengan melakukan rollback otomatis ke versi sebelumnya, dampak pada pelanggan diminimalkan. Rollback otomatis pada akhirnya memunculkan keyakinan yang lebih tinggi pada kemampuan deployment Anda dengan mengurangi dampak bisnis. Secara keseluruhan, kemampuan ini mengurangi waktu pengiriman sekaligus memastikan kualitas.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Otomatiskan pengujian lingkungan yang di-deploy untuk mengonfirmasi hasil yang diinginkan dengan lebih cepat. Otomatiskan pengembalian ke keadaan yang diketahui baik sebelumnya ketika hasil yang ditetapkan di awal tidak tercapai, untuk mempersingkat waktu pemulihan dan mengurangi kesalahan yang disebabkan oleh proses manual. Integrasikan alat pengujian dengan alur kerja pipeline Anda untuk menguji dan meminimalkan input manual secara konsisten. Prioritaskan otomatisasi kasus pengujian, seperti kasus pengujian yang mengurangi risiko terbesar dan perlu sering diuji dengan setiap perubahan. Selain itu, otomatiskan rollback berdasarkan kondisi tertentu yang telah ditentukan di awal dalam rencana pengujian Anda.

Langkah implementasi

1. Bangun siklus pengujian untuk siklus hidup pengembangan Anda yang menentukan setiap tahap proses pengujian mulai dari perencanaan persyaratan hingga pengembangan kasus pengujian, konfigurasi alat, pengujian otomatis, dan penutupan kasus pengujian.
   a. Buat pendekatan pengujian khusus beban kerja dari strategi pengujian Anda secara keseluruhan.
   b. Pertimbangkan strategi pengujian berkelanjutan jika diperlukan di seluruh siklus pengembangan.
2. Pilih alat otomatis untuk pengujian dan rollback berdasarkan kebutuhan bisnis dan investasi pipeline Anda.
3. Tentukan kasus pengujian mana yang ingin Anda otomatisasi dan mana yang harus dilakukan secara manual. Anda dapat menentukannya berdasarkan prioritas nilai bisnis dari fitur yang sedang diuji. Selaraskan semua anggota tim dengan rencana ini dan pastikan akuntabilitas untuk melakukan pengujian manual.
   a. Terapkan kemampuan pengujian otomatis ke kasus pengujian tertentu yang cocok untuk otomatisasi, seperti kasus berulang atau yang sering dijalankan, kasus yang memerlukan tugas berulang, atau kasus yang diperlukan di beberapa konfigurasi.
   b. Tentukan skrip otomatisasi pengujian serta kriteria keberhasilan di dalam alat otomatisasi sehingga otomatisasi alur kerja yang berkelanjutan dapat dimulai ketika ada kasus tertentu yang gagal.
   c. Tentukan kriteria kegagalan khusus untuk rollback otomatis.
4. Prioritaskan otomatisasi pengujian untuk mendorong hasil yang konsisten dengan pengembangan kasus pengujian menyeluruh di mana kompleksitas dan interaksi manusia memiliki risiko kegagalan yang lebih tinggi.
5. Integrasikan pengujian otomatis dan alat rollback Anda ke dalam pipeline CI/CD Anda.
   a. Kembangkan kriteria keberhasilan yang jelas untuk perubahan Anda.
   b. Pantau dan amati untuk mendeteksi kriteria ini dan secara otomatis membalikkan perubahan ketika kriteria rollback tertentu terpenuhi.
6. Lakukan berbagai jenis pengujian produksi otomatis, seperti:
   a. Pengujian A/B untuk menunjukkan hasil yang dibandingkan dengan versi saat ini antara dua kelompok pengujian pengguna.
   b. Pengujian canary yang memungkinkan Anda untuk meluncurkan perubahan Anda pada subset pengguna sebelum merilisnya ke semua pengguna.

   c. Pengujian penandaan fitur (feature flag) yang memungkinkan satu per satu fitur dari versi baru untuk ditandai atau dihapus tandanya dari luar aplikasi sehingga setiap fitur baru dapat divalidasi satu per satu.
   d. Pengujian regresi untuk memverifikasi fungsionalitas baru dengan komponen yang saling terkait.
7. Pantau aspek operasional aplikasi, transaksi, dan interaksi dengan aplikasi dan komponen lain. Kembangkan laporan untuk menunjukkan keberhasilan perubahan berdasarkan beban kerja sehingga Anda dapat mengidentifikasi bagian otomatisasi dan alur kerja apa yang dapat dioptimalkan lebih lanjut.
   a. Kembangkan laporan hasil pengujian yang membantu Anda mengambil keputusan cepat terkait apakah prosedur rollback perlu dimulai.
   b. Terapkan strategi yang memungkinkan rollback otomatis berdasarkan kondisi kegagalan yang telah ditentukan di awal yang dihasilkan dari satu atau beberapa metode pengujian Anda.
8. Kembangkan kasus pengujian otomatis untuk memungkinkan penggunaan ulang di seluruh perubahan berulang di masa mendatang.

Tingkat upaya untuk rencana implementasi: Sedang

Sumber daya

Praktik terbaik terkait:

- OPS06-BP01 Antisipasikan perubahan yang tidak berhasil
- OPS06-BP02 Menguji deployment

Dokumen terkait:

- AWS Builders Library | Memastikan keamanan rollback selama deployment
- Deployment ulang dan membatalkan deployment dengan AWS CodeDeploy
- 8 praktik terbaik saat mengotomatiskan deployment Anda dengan AWS CloudFormation

Contoh terkait:

- Pengujian UI nirserver menggunakan Selenium, AWS Lambda, AWS Fargate, dan Alat Developer AWS

Video terkait:

- re:Invent 2020 | Hands-off: Mengotomatiskan pipeline pengiriman berkelanjutan di Amazon
- re:Invent 2019 | Pendekatan deployment ketersediaan tinggi Amazon

OPS 7. Bagaimana cara mengetahui bahwa Anda siap untuk mendukung beban kerja?

Evaluasi kesiapan operasional beban kerja, proses, dan prosedur, serta personel Anda untuk memahami risiko operasional terkait beban kerja Anda.

Praktik terbaik

- OPS07-BP01 Memastikan kemampuan personel
- OPS07-BP02 Memastikan peninjauan yang konsisten terkait kesiapan operasional
- OPS07-BP03 Menggunakan runbook untuk menjalankan prosedur
- OPS07-BP04 Menggunakan buku panduan untuk menyelidiki masalah
- OPS07-BP05 Membuat keputusan yang tepat untuk melakukan deployment sistem dan perubahan
- OPS07-BP06 Mengaktifkan rencana dukungan untuk beban kerja produksi

OPS07-BP01 Memastikan kemampuan personel

Miliki mekanisme untuk memvalidasi bahwa Anda memiliki jumlah personel terlatih yang sesuai untuk mendukung beban kerja. Mereka harus diberi pelatihan tentang platform dan layanan yang membentuk beban kerja Anda. Berikan kepada mereka pengetahuan yang diperlukan untuk mengoperasikan beban kerja. Anda harus memiliki cukup banyak personel terlatih untuk mendukung pengoperasian normal beban kerja dan menyelesaikan masalah terkait insiden yang muncul. Miliki cukup banyak personel sehingga Anda dapat melakukan rotasi untuk personel yang siap tugas mendadak dan personel yang liburan guna menghindari personel menjadi terlalu lelah.

Hasil yang diinginkan:

- Ada cukup banyak personel terlatih untuk mendukung beban kerja pada saat beban kerja tersedia.
- Anda memberikan pelatihan untuk personel tentang perangkat lunak dan layanan yang membentuk beban kerja Anda.

Antipola umum:

- Melakukan deployment beban kerja tanpa anggota tim yang terlatih untuk mengoperasikan platform dan layanan yang digunakan.
- Tidak memiliki cukup banyak personel untuk mendukung rotasi personel yang siap tugas mendadak atau personel yang sedang libur.

Manfaat menjalankan praktik terbaik ini:

- Memiliki anggota tim yang terampil memungkinkan dukungan yang efektif untuk beban kerja.
- Dengan cukup banyak anggota tim, Anda dapat mendukung beban kerja dan rotasi personel yang siap tugas mendadak sekaligus mengurangi risiko personel terlalu lelah.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Validasikan bahwa terdapat personel yang terlatih dengan memadai untuk mendukung beban kerja. Pastikan Anda memiliki jumlah anggota tim yang cukup untuk menangani aktivitas operasional normal, termasuk rotasi personel siap tugas mendadak.

Contoh pelanggan

AnyCompany Retail memastikan tim yang mendukung beban kerja memiliki staf yang terlatih dalam jumlah yang sesuai. Mereka memiliki cukup banyak rekayasawan untuk mendukung rotasi personel yang siap tugas mendadak. Personel mendapatkan pelatihan tentang perangkat lunak dan platform yang merupakan dasar pembangunan beban kerja dan mereka didorong untuk mendapatkan sertifikasi. Ada cukup banyak personel sehingga orang dapat mengambil cuti sambil tetap ada dukungan untuk beban kerja dan rotasi personel yang siap tugas mendadak.

Langkah implementasi

1. Tetapkan jumlah personel yang memadai untuk mengoperasikan dan mendukung beban kerja Anda, termasuk tugas mendadak.
2. Latih personel Anda tentang perangkat lunak dan platform yang membentuk beban kerja Anda.
   a. AWS Training and Certification memiliki pustaka kursus tentang AWS. Kursus-kursus ini disediakan gratis dan berbayar, baik secara online maupun tatap muka.
   b. AWS melakukan hosting acara dan webinar di mana Anda belajar dari para ahli AWS.
3. Evaluasi ukuran dan keterampilan tim secara teratur seiring perubahan kondisi pengoperasian dan beban kerja. Sesuaikan ukuran dan keterampilan tim agar memenuhi persyaratan operasional.

Tingkat upaya untuk rencana implementasi: Tinggi. Mempekerjakan dan melatih tim untuk mendukung beban kerja dapat memerlukan upaya yang cukup besar tetapi memiliki manfaat jangka panjang yang substansial.

Sumber daya

Praktik Terbaik Terkait:

- OPS11-BP04 Menjalankan manajemen pengetahuan - Anggota tim harus memiliki informasi yang diperlukan untuk mengoperasikan dan mendukung beban kerja. Manajemen pengetahuan merupakan kunci untuk penyediaan tersebut.

Dokumen terkait:

- Acara dan Webinar AWS
- AWS Training and Certification

OPS07-BP02 Memastikan peninjauan yang konsisten terkait kesiapan operasional

Gunakan Peninjauan Kesiapan Operasional (ORR) untuk memvalidasi bahwa Anda dapat mengoperasikan beban kerja Anda. ORR adalah mekanisme yang dikembangkan di Amazon untuk memvalidasi bahwa tim dapat mengoperasikan beban kerja mereka dengan aman. ORR adalah proses peninjauan dan inspeksi menggunakan daftar periksa persyaratan. ORR adalah pengalaman layanan mandiri yang digunakan tim untuk memastikan beban kerja mereka. ORR mencakup praktik terbaik dari pelajaran yang kami dapatkan selama bertahun-tahun membangun perangkat lunak.

Daftar periksa ORR terdiri dari rekomendasi arsitektur, proses operasional, manajemen peristiwa, dan kualitas rilis. Proses Koreksi Kesalahan (CoE) kami merupakan pendorong utama item-item ini. Analisis pascainsiden Anda sendiri harus mendorong pengembangan ORR Anda. ORR tidak hanya tentang mengikuti praktik terbaik tapi juga mencegah kemungkinan peristiwa yang telah Anda lihat sebelumnya. Terakhir, keamanan, pengelolaan, dan kepatuhan persyaratan juga dapat disertakan dalam ORR.

Jalankan ORR sebelum beban kerja meluncur ke ketersediaan umum dan kemudian ke seluruh siklus pengembangan perangkat lunak. Menjalankan ORR sebelum peluncuran meningkatkan kemampuan Anda untuk mengoperasikan beban kerja dengan aman. Jalankan kembali ORR Anda secara berkala pada beban kerja untuk mengetahui penyimpangan dari praktik terbaik. Anda dapat memiliki daftar periksa ORR untuk peluncuran layanan baru dan ORR untuk peninjauan berkala. Ini membantu Anda untuk tetap up to date dengan praktik terbaik yang muncul dan menggabungkan

pelajaran yang didapatkan dari analisis pascainsiden. Saat penggunaan cloud Anda matang, Anda dapat membangun persyaratan ORR ke dalam arsitektur Anda secara default.

Hasil yang diinginkan: Anda memiliki daftar periksa ORR dengan praktik terbaik untuk organisasi Anda. ORR dilakukan sebelum peluncuran beban kerja. ORR dijalankan secara berkala selama kursus siklus beban kerja.

Antipola umum:

- Anda meluncurkan beban kerja tanpa mengetahui apakah Anda dapat mengoperasikannya.
- Persyaratan pengelolaan dan keamanan tidak diikutsertakan ketika menyertifikasi beban kerja untuk peluncuran.
- Beban kerja tidak dievaluasi kembali secara berkala.
- Beban kerja diluncurkan tanpa diterapkannya prosedur yang diperlukan.
- Anda melihat pengulangan kegagalan akar masalah yang sama di beberapa beban kerja.

Manfaat menjalankan praktik terbaik ini:

- Beban kerja Anda mencakup praktik terbaik arsitektur, proses, dan manajemen.
- Pelajaran yang didapatkan digabungkan dalam proses ORR.
- Prosedur yang diperlukan tersedia ketika beban kerja diluncurkan.
- ORR dijalankan di seluruh siklus perangkat lunak beban kerja Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

ORR adalah dua hal: proses dan daftar periksa. Proses ORR Anda harus diadopsi oleh organisasi Anda dan didukung oleh sponsor eksekutif. Minimal, ORR harus dilakukan sebelum beban kerja meluncur ke ketersediaan umum. Jalankan ORR di seluruh siklus pengembangan perangkat lunak untuk tetap up to date dengan praktik terbaik atau persyaratan baru. Daftar periksa ORR harus mencakup item konfigurasi, persyaratan keamanan dan pengelolaan, serta praktik terbaik dari organisasi Anda. Seiring waktu, Anda dapat menggunakan layanan, seperti AWS Config, AWS Security Hub CSPM, dan Pagar Pembatas AWS Control Tower, untuk membangun praktik terbaik dari ORR ke pagar pembatas untuk deteksi praktik terbaik secara otomatis.

Contoh pelanggan

Setelah beberapa insiden produksi, AnyCompany Retail memutuskan untuk menerapkan proses ORR. Mereka membangun daftar periksa yang terdiri dari praktik terbaik, persyaratan pengelolaan dan kepatuhan, serta pelajaran yang didapatkan dari pemadaman. Beban kerja baru melakukan ORR sebelum diluncurkan. Setiap beban kerja melakukan ORR setiap tahun dengan sebagian praktik terbaik untuk menggabungkan praktik terbaik dan persyaratan baru yang ditambahkan ke daftar periksa ORR. Seiring waktu, AnyCompany Retail menggunakan AWS Config untuk mendeteksi beberapa praktik terbaik, yang mempercepat proses ORR.

Langkah implementasi

Untuk mempelajari selengkapnya tentang ORR, baca: laporan resmi Peninjauan Kesiapan Operasional (ORR). Laporan resmi ini menyediakan detail informasi tentang riwayat proses ORR, cara membangun praktik ORR Anda sendiri, dan cara mengembangkan daftar periksa ORR Anda. Langkah-langkah berikut ini merupakan versi singkat dari dokumen tersebut. Untuk pemahaman yang mendalam tentang apa itu ORR dan bagaimana membangunnya, sebaiknya baca laporan resmi tersebut.

1. Kumpulkan pemangku kepentingan utama, termasuk perwakilan dari keamanan, operasi, dan pengembangan.
2. Minta setiap pemangku kepentingan untuk menyediakan setidaknya satu persyaratan. Untuk iterasi pertama, coba batasi jumlah item menjadi 30 atau kurang.
   - Lampiran B: Contoh pertanyaan ORR dari laporan resmi Peninjauan Kesiapan Operasional (ORR) yang berisi sampel pertanyaan yang dapat Anda gunakan untuk memulai.
3. Kumpulkan persyaratan Anda ke dalam lembar kerja.
   - Anda dapat menggunakan lensa kustom di AWS Well-Architected Tool untuk mengembangkan ORR Anda dan membagikannya ke seluruh akun dan Organisasi AWS Anda.
4. Identifikasi satu beban kerja untuk diberikan ORR. Idealnya adalah beban kerja sebelum peluncuran atau beban kerja internal.
5. Pelajari daftar periksa ORR dan catat semua penemuan yang dibuat. Penemuannya mungkin akan buruk jika terdapat mitigasi. Untuk penemuan yang minim mitigasi, tambahkan beban kerja ke backlog item Anda dan implementasikan sebelum peluncuran.
6. Lanjutkan penambahan praktik terbaik dan persyaratan ke daftar periksa ORR Anda seiring waktu.

Pelanggan Dukungan dengan Enterprise Support dapat mengajukan permintaan Lokakarya Peninjauan Kesiapan Operasional dari Manajer Akun Teknis mereka. Lokakarya ini adalah sesi penelusuran mundur (working backward) interaktif untuk mengembangkan daftar periksa ORR Anda.

Tingkat upaya untuk rencana implementasi: Tinggi. Untuk mengadopsi praktik ORR pada organisasi Anda diperlukan sponsor eksekutif dan dukungan pemangku kepentingan. Buat dan perbarui daftar periksa dengan masukan dari seluruh organisasi Anda.

Sumber daya

Praktik Terbaik Terkait:

- OPS01-BP03 Evaluasi persyaratan tata kelola – Persyaratan tata kelola sangat sesuai untuk daftar periksa ORR.
- OPS01-BP04 Evaluasi persyaratan kepatuhan – Terkadang persyaratan kepatuhan tercantum di daftar periksa ORR. Terkadang persyaratan kepatuhan adalah proses yang terpisah.
- OPS03-BP07 Bekali tim dengan sumber daya dengan sesuai – Kemampuan tim merupakan kandidat yang bagus untuk persyaratan ORR.
- OPS06-BP01 Antisipasikan perubahan yang tidak berhasil – Rencana rollback atau rollfoward harus dibuat sebelum Anda meluncurkan beban kerja Anda.
- OPS07-BP01 Memastikan kemampuan personel – Untuk mendukung beban kerja, Anda harus memiliki personel yang diperlukan.
- SEC01-BP03 Mengidentifikasi dan memvalidasi tujuan kontrol – Tujuan kontrol keamanan menyempurnakan persyaratan ORR.
- REL13-BP01 Menetapkan sasaran pemulihan untuk waktu henti dan kehilangan data – Rencana pemulihan bencana merupakan persyaratan ORR yang bagus.
- COST02-BP01 Mengembangkan kebijakan berdasarkan keperluan organisasi Anda – Kebijakan manajemen biaya bagus untuk dicantumkan dalam daftar ORR Anda.

Dokumen terkait:

- AWS Control Tower - Pagar Pembatas di AWS Control Tower
- AWS Well-Architected Tool - Lensa Kustom
- Templat Peninjauan Kesiapan Operasional oleh Adrian Hornsby
- Laporan Resmi Peninjauan Kesiapan Operasional (ORR)

Video terkait:

- AWS Dukungan Anda | Membangun Peninjauan Kesiapan Operasional (ORR) yang Efektif

Contoh terkait:

- Sampel Lensa Peninjauan Kesiapan Operasional (ORR)

Layanan terkait:

- AWS Config
- AWS Control Tower
- AWS Security Hub CSPM
- AWS Well-Architected Tool

OPS07-BP03 Menggunakan runbook untuk menjalankan prosedur

Sebuah runbook adalah proses terdokumentasi untuk mencapai hasil tertentu. Runbook terdiri dari serangkaian langkah yang diikuti seseorang untuk menyelesaikan sesuatu. Runbook telah digunakan dalam operasi sejak masa-masa awal industri penerbangan. Dalam operasi cloud, kita menggunakan runbook untuk mengurangi risiko dan mencapai hasil yang diinginkan. Dalam bentuk paling sederhananya, runbook adalah daftar periksa untuk menyelesaikan tugas.

Runbook adalah bagian penting dari operasi beban kerja Anda. Mulai dari orientasi anggota tim baru hingga melakukan deployment rilis utama, runbook adalah proses terkodifikasi yang memberikan hasil konsisten, siapa pun yang menggunakannya. Runbook harus dipublikasikan di lokasi sentral dan diperbarui seiring prosesnya berkembang karena memperbarui runbook adalah komponen utama dari proses manajemen perubahan. Runbook juga harus menyertakan panduan tentang penanganan kesalahan, alat, izin, pengecualian, dan eskalasi jika terjadi masalah.

Saat organisasi Anda matang, mulailah mengotomatiskan runbook. Mulailah dengan runbook yang singkat dan sering digunakan. Gunakan bahasa skrip untuk mengotomatiskan langkah-langkah atau mempermudah pelaksanaan langkah-langkah. Seiring Anda mengotomatiskan beberapa runbook pertama, Anda akan mendedikasikan waktu untuk mengotomatiskan runbook yang lebih kompleks. Seiring waktu, sebagian besar runbook Anda harus diotomatiskan dalam cara tertentu.

Hasil yang diinginkan: Tim Anda memiliki kumpulan panduan langkah demi langkah untuk melakukan tugas beban kerja. Runbook berisi hasil yang diinginkan, alat dan izin yang diperlukan, serta petunjuk untuk penanganan kesalahan. Runbook disimpan di lokasi sentral dan sering diperbarui.

Antipola umum:

- Mengandalkan memori untuk menyelesaikan setiap langkah dari suatu proses.
- Menerapkan perubahan secara manual tanpa daftar periksa.
- Anggota tim yang berbeda-beda melakukan proses yang sama, tetapi dengan langkah atau hasil yang berbeda.
- Membiarkan runbook tidak sinkron dengan perubahan sistem dan otomatisasi.

Manfaat menjalankan praktik terbaik ini:

- Mengurangi tingkat kesalahan untuk tugas manual.
- Operasi dilakukan secara konsisten.
- Anggota tim baru dapat mulai melakukan tugas dengan lebih cepat.
- Runbook dapat diotomatiskan untuk mengurangi upaya yang diperlukan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Runbook dapat memiliki beberapa bentuk, bergantung pada tingkat kematangan organisasi Anda. Minimal, runbook harus terdiri dari dokumen teks langkah demi langkah. Hasil yang diinginkan harus ditunjukkan dengan jelas. Dokumentasikan dengan jelas izin atau alat khusus yang diperlukan. Berikan panduan mendetail tentang penanganan kesalahan dan eskalasi jika terjadi kesalahan. Cantumkan pemilik runbook dan publikasikan di lokasi sentral. Setelah runbook Anda didokumentasikan, validasikan dengan meminta orang lain di tim Anda untuk menjalankannya. Seiring prosedur berkembang, perbarui runbook Anda sesuai dengan proses manajemen perubahan Anda.

Runbook teks Anda harus diotomatiskan seiring organisasi Anda makin matang. Dengan layanan seperti otomatisasi AWS Systems Manager, Anda dapat mentransformasikan teks biasa menjadi otomatisasi yang dapat dijalankan dengan beban kerja Anda. Otomatisasi ini dapat dijalankan sebagai respons terhadap peristiwa, sehingga mengurangi beban operasional untuk memelihara beban kerja Anda.

Contoh pelanggan

AnyCompany Retail harus melakukan pembaruan skema basis data selama deployment perangkat lunak. Tim Operasi Cloud bekerja sama dengan Tim Administrasi Basis Data untuk membuat runbook guna menerapkan perubahan ini secara manual. Runbook ini mencantumkan setiap langkah

prosesnya dalam bentuk daftar periksa. Runbook ini berisi bagian tentang penanganan kesalahan jika terjadi kesalahan. Mereka memublikasikan runbook di wiki internal mereka bersama dengan runbook mereka yang lain. Tim Operasi Cloud berencana untuk mengotomatiskan runbook dalam sprint mendatang.

Langkah implementasi

Jika Anda belum memiliki repositori dokumen, repositori kontrol versi adalah tempat yang tepat untuk mulai membangun pustaka runbook Anda. Anda dapat membangun runbook Anda menggunakan Markdown. Kami telah menyediakan contoh templat runbook yang dapat Anda gunakan untuk mulai membangun runbook.

```
# Judul Runbook ## Info Runbook | ID Runbook | Deskripsi | Alat yang Digunakan
 | Izin Khusus | Penulis Runbook | Terakhir Diperbarui | POC Eskalasi |
 |-------|-------|-------|-------|-------|-------|-------| | RUN001 | Apa tujuan
 penggunaan runbook ini? Apa hasil yang diinginkan? | Alat | Izin | Nama Anda |
 21-9-2022 | Nama Eskalasi | ## Langkah 1. Langkah pertama 2. Langkah kedua
```

1. Jika Anda belum memiliki repositori atau wiki dokumentasi, buat repositori kontrol versi baru di sistem kontrol versi Anda.
2. Identifikasi proses yang tidak memiliki runbook. Proses yang ideal adalah proses yang dilakukan secara semireguler, sedikit jumlah langkahnya, dan memiliki kegagalan berdampak rendah.
3. Di repositori dokumen Anda, buat draf dokumen Markdown baru menggunakan templat tersebut. Isi `Judul Runbook` dan bidang yang diperlukan di bagian `Info Runbook`.
4. Dimulai dengan langkah pertama, isi bagian `Langkah` dalam runbook ini.
5. Berikan runbook ini kepada para anggota tim. Minta mereka menggunakan runbook ini untuk memvalidasi langkah-langkahnya. Jika ada sesuatu yang belum dimasukkan atau memerlukan kejelasan, perbarui runbook ini.
6. Publikasikan runbook ini ke penyimpanan dokumentasi internal Anda. Setelah dipublikasikan, beri tahu tim Anda dan pemangku kepentingan lainnya.
7. Seiring waktu, Anda akan membangun pustaka runbook. Saat pustaka tersebut tumbuh, mulailah bekerja untuk mengotomatiskan runbook.

Tingkat upaya untuk rencana implementasi: Rendah. Standar minimum untuk runbook adalah panduan teks langkah demi langkah. Mengotomatiskan runbook dapat meningkatkan upaya implementasi.

Sumber daya

Praktik terbaik terkait:

- OPS02-BP02 Proses dan Prosedur memiliki pemilik teridentifikasi: Runbook harus memiliki pemilik yang bertanggung jawab untuk memeliharanya.
- OPS07-BP04 Menggunakan buku panduan untuk menyelidiki masalah: Runbook dan playbook mirip satu sama lain dengan satu perbedaan utama: runbook memiliki hasil yang diinginkan. Dalam banyak kasus, runbook dipicu setelah playbook mengidentifikasi akar penyebab.
- OPS10-BP01 Menggunakan proses untuk manajemen peristiwa, insiden, dan masalah: Runbook adalah bagian dari praktik manajemen yang baik terkait peristiwa, insiden, dan masalah.
- OPS10-BP02 Menjalankan proses untuk setiap peringatan: Runbook dan playbook harus digunakan untuk menanggapi peringatan. Seiring waktu, reaksi ini harus diotomatiskan.
- OPS11-BP04 Menjalankan manajemen pengetahuan: Memelihara runbook adalah bagian penting dari manajemen pengetahuan.

Dokumen terkait:

- Mencapai Keunggulan Operasional menggunakan playbook dan runbook otomatis
- AWS Systems Manager: Bekerja dengan runbook
- Playbook migrasi untuk migrasi besar AWS - Tugas 4: Meningkatkan runbook migrasi Anda
- Gunakan runbook Otomatisasi AWS Systems Manager untuk menyelesaikan tugas operasional

Video terkait:

- AWS re:Invent 2019: Panduan mandiri untuk runbook, laporan insiden, dan respons insiden (SEC318-R1)
- Cara mengotomatiskan Operasi IT di AWS | Amazon Web Services
- Integrasikan Skrip ke dalam AWS Systems Manager

Contoh terkait:

- AWS Systems Manager: Panduan otomatisasi
- AWS Systems Manager: Pulihkan volume root dari snapshot runbook terbaru
- Membangun runbook respons insiden AWS menggunakan notebook Jupyter dan CloudTrail Lake

- Gitlab - Runbook
- Rubix - Pustaka Python untuk membuat runbook di Notebook Jupyter
- Menggunakan Document Builder untuk membuat runbook kustom
- Lab Well-Architected: Mengotomatiskan operasi dengan Playbook dan Runbook

Layanan terkait:

- Otomatisasi AWS Systems Manager

OPS07-BP04 Menggunakan buku panduan untuk menyelidiki masalah

Playbook adalah panduan mendetail yang digunakan untuk menyelidiki insiden. Ketika terjadi sebuah insiden, playbook digunakan untuk menyelidiki, membuat cakupan dampak, dan mengidentifikasi akar masalah. Playbook digunakan untuk berbagai skenario, dari deployment yang gagal hingga insiden keamanan. Dalam banyak kasus, playbook mengidentifikasi akar masalah yang dimitigasi menggunakan runbook. Playbook adalah komponen pokok dalam rencana respons insiden organisasi Anda.

Playbook yang baik memiliki sejumlah fitur utama. Playbook memberikan panduan secara mendetail bagi pengguna, dalam proses penemuan. Dengan berpikir secara holistik, langkah apa saja yang sebaiknya diikuti seseorang untuk mendiagnosis insiden? Tetapkan secara jelas di dalam playbook jika alat-alat khusus atau izin yang dinaikkan diperlukan di dalam playbook. Memiliki rencana komunikasi untuk memberi informasi kepada para pemangku kepentingan mengenai status penyelidikan adalah komponen utama. Dalam situasi ketika akar penyebab tidak dapat diidentifikasi, playbook harus memiliki rencana eskalasi. Jika akar masalah diidentifikasi, playbook harus mengarah ke runbook yang menjelaskan cara menyelesaikannya. Playbook harus disimpan secara terpusat dan dipelihara secara rutin. Jika playbook digunakan untuk pemberitahuan khusus, bekali tim Anda dengan penunjuk ke playbook di dalam pemberitahuan tersebut.

Otomatisasi playbook Anda seiring kematangan organisasi. Mulai dengan playbook yang mencakup insiden berisiko rendah. Gunakan penulisan skrip untuk mengotomatiskan langkah-langkah penemuan. Pastikan Anda memiliki runbook pendamping untuk memitigasi akar masalah umum.

Hasil yang diinginkan: Organisasi Anda memiliki playbook untuk insiden umum. Playbook disimpan di lokasi terpusat dan tersedia untuk anggota tim Anda. Playbook sering diperbarui. Runbook pendamping dibuat untuk akar masalah apa pun yang diketahui.

Antipola umum:

- Tidak ada cara standar untuk menyelidiki insiden.
- Anggota tim mengandalkan memori otot atau pengetahuan institusional untuk memecahkan masalah kegagalan deployment.
- Anggota tim baru mempelajari cara menyelidiki permasalahan melalui coba-coba.
- Praktik terbaik untuk menyelidiki permasalahan tidak dibagikan ke seluruh tim.

Manfaat menjalankan praktik terbaik ini:

- Playbook meningkatkan upaya Anda untuk memitigasi insiden.
- Anggota tim yang berbeda-beda dapat menggunakan playbook yang sama untuk mengidentifikasi akar masalah secara konsisten.
- Setelah akar masalah diketahui kemudian bisa dikembangkan runbook, sehingga dapat mempercepat waktu pemulihan.
- Playbook memungkinkan anggota tim untuk mulai berkontribusi lebih cepat.
- Tim dapat menskalakan proses mereka dengan playbook yang dapat diulang.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Bagaimana Anda membangun dan menggunakan playbook bergantung pada kematangan organisasi Anda. Jika Anda baru mengenal cloud, bangun playbook dalam bentuk teks di dalam repositori dokumen pusat. Seiring kematangan organisasi, playbook dapat menjadi semi-otomatis dengan bahasa skrip seperti Python. Skrip-skrip ini dapat dijalankan di dalam notebook Jupyter untuk mempercepat penemuan. Organisasi tingkat lanjut memiliki playbook yang sepenuhnya otomatis untuk permasalahan umum yang diperbaiki secara otomatis dengan runbook.

Mulai bangun playbook Anda dengan mengidentifikasi insiden-insiden umum yang terjadi pada beban kerja Anda. Pilih playbook untuk insiden berisiko rendah dan dengan akar masalah yang telah dipersempit menjadi beberapa permasalahan untuk mengawalinya. Setelah Anda memiliki playbook untuk skenario yang lebih sederhana, beralihlah ke skenario berisiko lebih tinggi atau skenario dengan akar masalah yang tidak dikenal dengan baik.

Playbook teks Anda harus diotomatiskan seiring pematangan organisasi Anda. Dengan layanan seperti AWS Systems Manager Automations, teks biasa dapat ditransformasikan menjadi otomatis. Otomatisasi ini dapat dijalankan terhadap beban kerja untuk mempercepat penyelidikan. Otomatisasi

ini dapat diaktifkan untuk merespons peristiwa, sehingga mengurangi rata-rata waktu untuk menemukan dan menyelesaikan insiden.

Pelanggan dapat menggunakan Manajer Insiden AWS Systems Manager untuk merespons insiden. Layanan ini menyediakan satu antarmuka untuk memeriksa insiden, memberi informasi kepada pemangku kepentingan selama penemuan dan mitigasi, dan berkolaborasi melalui insiden. Layanan ini menggunakan AWS Systems Manager Automations untuk mempercepat deteksi dan pemulihan.

Contoh pelanggan

Insiden produksi memberikan dampak pada AnyCompany Retail. Rekayasawan yang siap dipanggil kapan saja (on-call) menggunakan playbook untuk menyelidiki permasalahan. Seiring mereka mengikuti langkah-langkahnya, mereka terus memutakhirkan pemangku kepentingan utama yang diidentifikasi di dalam playbook. Rekayasawan mengidentifikasi akar masalah sebagai kondisi pacu di dalam layanan backend. Menggunakan runbook, rekayasawan meluncurkan ulang layanan, sehingga AnyCompany Retail dapat kembali online.

Langkah implementasi

Jika Anda belum memiliki repositori dokumen, kami menyarankan pembuatan repositori kontrol versi untuk pustaka playbook Anda. Anda dapat membangun playbook Anda menggunakan Markdown, yang kompatibel dengan sebagian besar sistem otomatisasi playbook. Jika Anda memulai dari nol, gunakan contoh templat playbook berikut ini.

```
# Judul Playbook ## Info Playbook | ID Playbook | Deskripsi | Alat
 yang Digunakan | Izin Khusus | Penyusun Playbook | Terakhir Diperbarui
 | POC Eskalasi | Pemangku Kepentingan | Rencana Komunikasi |
 |-------|-------|-------|-------|-------|-------|-------|-------|-------| | RUN001
 | Untuk apa playbook ini? Untuk insiden apa playbook ini? | Alat | Izin | Nama Anda
 | 21-09-2022 | Nama Eskalasi | Nama Pemangku Kepentingan | Bagaimana pembaruan akan
 disampaikan selama penyelidikan? | ## Langkah 1. Langkah pertama 2. Langkah kedua
```

1. Jika Anda belum memiliki repositori dokumen atau wiki, buat repositori kontrol versi baru untuk playbook Anda di sistem kontrol versi Anda.
2. Identifikasi permasalahan umum yang memerlukan penyelidikan. Ini sebaiknya adalah skenario dengan akar masalah yang dibatasi ke beberapa permasalahan dan penyelesaiannya berisiko rendah.
3. Menggunakan templat Markdown, lengkapi bagian `Nama Playbook` dan bidang di bawah `Info Playbook`.

4. Lengkapi langkah-langkah pemecahan masalah. Sampaikan sejelas mungkin tindakan yang akan dilakukan atau area apa saja yang harus Anda selidiki.
5. Berikan playbook tersebut kepada anggota tim dan minta mereka mempelajari dan memvalidasinya. Jika terdapat hal yang terlewat atau tidak jelas, perbarui playbook.
6. Terbitkan playbook di dalam repositori dokumen Anda dan informasikan kepada tim dan pemangku kepentingan.
7. Pustaka playbook ini akan tumbuh seiring Anda menambahkan lebih banyak playbook. Setelah Anda memiliki beberapa playbook, mulailah mengotomatiskannya menggunakan alat seperti AWS Systems Manager Automations untuk terus menyinkronkan otomatisasi dan playbook.

Tingkat upaya untuk rencana implementasi: Rendah. Playbook Anda harus berupa dokumen teks yang disimpan di lokasi terpusat. Organisasi yang lebih matang akan beralih ke otomatisasi playbook.

Sumber daya

Praktik terbaik terkait:

- OPS02-BP02 Proses dan Prosedur memiliki pemilik teridentifikasi: Runbook harus memiliki pemilik yang bertanggung jawab untuk memeliharanya.
- OPS07-BP03 Menggunakan runbook untuk menjalankan prosedur: Runbook dan playbook mirip, tetapi dengan satu perbedaan utama: runbook memiliki hasil yang diinginkan. Dalam banyak kasus, runbook digunakan setelah playbook mengidentifikasi akar penyebab.
- OPS10-BP01 Menggunakan proses untuk manajemen peristiwa, insiden, dan masalah: Playbook adalah bagian dari praktik manajemen yang baik terkait peristiwa, insiden, dan masalah.
- OPS10-BP02 Menjalankan proses untuk setiap peringatan: Runbook dan playbook harus digunakan untuk menanggapi peringatan. Seiring waktu, reaksi ini harus diotomatiskan.
- OPS11-BP04 Menjalankan manajemen pengetahuan: Memelihara playbook adalah bagian penting dari manajemen pengetahuan.

Dokumen terkait:

- Mencapai Keunggulan Operasional menggunakan playbook dan runbook otomatis
- AWS Systems Manager: Bekerja dengan runbook
- Gunakan runbook AWS Systems Manager Automations untuk menyelesaikan tugas operasional

Video terkait:

- AWS re:Invent 2019: Panduan mandiri untuk runbook, laporan insiden, dan respons insiden (SEC318-R1)
- Manajer Insiden AWS Systems Manager - Lokakarya Virtual AWS
- Integrasikan Skrip ke dalam AWS Systems Manager

Contoh terkait:

- Kerangka Kerja Playbook Pelanggan AWS
- AWS Systems Manager: Panduan otomatisasi
- Membangun runbook respons insiden AWS menggunakan notebook Jupyter dan CloudTrail Lake
- Rubix – Pustaka Python untuk membuat runbook di Notebook Jupyter
- Menggunakan Document Builder untuk membuat runbook kustom
- Lab Well-Architected: Mengotomatiskan operasi dengan Playbook dan Runbook
- Lab Well-Architect: Playbook respons insiden dengan Jupyter

Layanan terkait:

- AWS Systems Manager Automation
- Manajer Insiden AWS Systems Manager

OPS07-BP05 Membuat keputusan yang tepat untuk melakukan deployment sistem dan perubahan

Miliki proses untuk perubahan yang sukses dan tidak sukses pada beban kerja Anda. Pre-mortem adalah latihan simulasi tim terhadap kegagalan untuk mengembangkan strategi mitigasi. Gunakan pre-mortem untuk mengantisipasi kegagalan dan menciptakan prosedur ketika diperlukan. Evaluasi manfaat dan risiko dari deployment perubahan ke beban kerja Anda. Verifikasi apakah semua perubahan mematuhi tata kelola.

Hasil yang diinginkan:

- Anda mengambil keputusan yang tepat ketika melakukan deployment perubahan ke beban kerja Anda.
- Perubahan mematuhi tata kelola.

Antipola umum:

- Melakukan deployment perubahan ke beban kerja tanpa proses untuk menangani deployment yang gagal.
- Membuat perubahan pada lingkungan produksi Anda yang tidak mematuhi persyaratan tata kelola.
- Melakukan deployment versi baru beban kerja Anda tanpa menetapkan garis dasar untuk pemanfaatan sumber daya.

Manfaat menjalankan praktik terbaik ini:

- Anda siap untuk menangani perubahan yang tidak sukses pada beban kerja Anda.
- Perubahan pada beban kerja Anda mematuhi kebijakan tata kelola.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Gunakan pre-mortem guna mengembangkan proses untuk perubahan yang tidak sukses. Dokumentasikan proses Anda untuk perubahan yang tidak sukses. Pastikan semua perubahan mematuhi tata kelola. Evaluasi manfaat dan risiko deployment perubahan ke beban kerja Anda.

Contoh pelanggan

AnyCompany Retail melakukan pre-mortem secara teratur guna memvalidasi proses mereka untuk perubahan yang tidak sukses. Mereka mendokumentasikan proses mereka di Wiki bersama dan sering kali memperbaruinya. Semua perubahan mematuhi persyaratan tata kelola.

Langkah implementasi

1. Ambil keputusan yang tepat ketika melakukan deployment perubahan ke beban kerja Anda. Tetapkan dan tinjau kriteria untuk deployment yang sukses. Kembangkan skenario atau kriteria yang akan memicu pengembalian perubahan ke versi sebelumnya. Pikirkan manfaat deployment perubahan dibandingkan risiko perubahan yang tidak sukses.
2. Verifikasi apakah semua perubahan mematuhi kebijakan tata kelola.
3. Gunakan pre-mortem guna membuat rencana untuk perubahan yang tidak sukses dan mendokumentasikan strategi mitigasi. Jalankan sesi latihan table-top untuk memperagakan perubahan yang tidak sukses dan memvalidasi prosedur pengembalian ke versi sebelumnya.

Tingkat upaya untuk rencana implementasi: Sedang. Mengimplementasikan praktik pre-mortem memerlukan koordinasi dan upaya dari para pemangku kepentingan dalam seluruh organisasi Anda

Sumber daya

Praktik terbaik terkait:

- OPS01-BP03 Evaluasi persyaratan tata kelola - Persyaratan tata kelola merupakan faktor kunci dalam menentukan apakah akan melakukan deployment perubahan.
- OPS06-BP01 Antisipasikan perubahan yang tidak berhasil - Buat rencana untuk memitigasi deployment yang gagal dan gunakan pre-mortem untuk memvalidasinya.
- OPS06-BP02 Menguji deployment - Setiap perubahan perangkat lunak harus diuji dengan tepat sebelum deployment untuk mengurangi kecacatan dalam produksi.
- OPS07-BP01 Memastikan kemampuan personel - Memiliki cukup banyak personel yang terlatih untuk mendukung beban kerja sangat penting dalam mengambil keputusan yang tepat dalam hal deployment perubahan sistem.

Dokumen terkait:

- Amazon Web Services: Risiko dan Kepatuhan
- Model Tanggung Jawab Bersama AWS
- Tata Kelola dalam AWS Cloud: Keseimbangan yang Tepat Antara Ketangkasan dan Keamanan

OPS07-BP06 Mengaktifkan rencana dukungan untuk beban kerja produksi

Aktifkan dukungan untuk perangkat lunak dan layanan yang diandalkan beban kerja produksi Anda. Pilih tingkat dukungan yang sesuai untuk memenuhi kebutuhan tingkat layanan produksi Anda. Rencana dukungan untuk dependensi ini diperlukan untuk berjaga-jaga jika ada gangguan layanan atau masalah perangkat lunak. Dokumentasikan rencana dukungan dan cara meminta dukungan untuk semua vendor perangkat lunak dan layanan. Implementasikan mekanisme yang memverifikasi bahwa titik kontak dukungan selalu yang terbaru.

Hasil yang diinginkan:

- Implementasikan rencana dukungan untuk perangkat lunak dan layanan yang diandalkan beban kerja produksi.
- Pilih rencana dukungan yang sesuai berdasarkan kebutuhan tingkat layanan.

- Dokumentasikan rencana dukungan, tingkat dukungan, dan cara meminta dukungan.

Antipola umum:

- Anda tidak memiliki rencana dukungan untuk vendor perangkat lunak yang penting. Beban kerja Anda terkena dampaknya dan Anda tidak dapat melakukan apa-apa untuk mempercepat perbaikan atau mendapatkan informasi terbaru yang tepat waktu dari vendor.
- Developer yang merupakan titik utama kontak untuk vendor perangkat lunak tidak lagi bekerja di perusahaan. Anda tidak dapat menghubungi dukungan vendor secara langsung. Anda harus meluangkan waktu menelusuri dan mencari-cari dalam sistem kontak generik, sehingga menambah waktu yang diperlukan untuk merespons ketika diperlukan.
- Penghentian produksi terjadi pada vendor perangkat lunak. Tidak ada dokumentasi tentang cara mengajukan kasus dukungan.

Manfaat menjalankan praktik terbaik ini:

- Dengan tingkat dukungan yang sesuai, Anda dapat memperoleh respons dalam kerangka waktu yang diperlukan untuk memenuhi kebutuhan tingkat layanan.
- Sebagai pelanggan yang didukung, Anda dapat melapor jika ada masalah produksi.
- Vendor layanan dan perangkat lunak dapat membantu menyelesaikan masalah selama insiden.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Aktifkan rencana dukungan untuk vendor perangkat lunak dan layanan yang diandalkan beban kerja produksi Anda. Atur rencana dukungan yang sesuai untuk memenuhi kebutuhan tingkat layanan Anda. Untuk pelanggan AWS, ini artinya mengaktifkan Business Support AWS atau yang lebih tinggi pada akun apa pun tempat Anda memiliki beban kerja produksi. Temui vendor dukungan secara teratur untuk mendapatkan informasi terbaru mengenai kontak, proses, dan penawaran dukungan. Dokumentasikan cara meminta dukungan dari vendor perangkat lunak dan layanan, termasuk cara melapor jika ada penghentian. Implementasikan mekanisme untuk menjaga agar kontak selalu yang terbaru.

Contoh pelanggan

Di AnyCompany Retail, semua dependensi layanan dan perangkat lunak komersial memiliki rencana dukungan. Contohnya, mereka mengaktifkan AWS Enterprise Support di semua akun dengan beban kerja produksi. Semua developer dapat membuka kasus dukungan bila ada masalah. Ada halaman wiki dengan informasi tentang cara meminta dukungan, siapa yang harus diberi tahu, dan praktik terbaik untuk mempercepat kasus.

Langkah implementasi

1. Bekerjasamalah dengan pemangku kepentingan di organisasi Anda untuk mengidentifikasi vendor perangkat lunak dan layanan yang diandalkan beban kerja Anda. Dokumentasikan dependensi ini.
2. Tentukan kebutuhan tingkat layanan untuk beban kerja Anda. Pilih rencana dukungan yang selaras dengannya.
3. Untuk layanan dan perangkat lunak komersial, tetapkan rencana dukungan dengan vendor.
    a. Dengan berlangganan AWS Business Support atau lebih tinggi untuk semua akun produksi, waktu respons AWS Dukungan akan lebih cepat dan hal ini sangat disarankan. Jika Anda tidak memiliki dukungan premium, Anda harus memiliki rencana tindakan untuk menangani masalah, yang memerlukan bantuan dari AWS Dukungan. AWS Dukungan memberikan kombinasi alat dan teknologi, orang, dan program yang dirancang untuk secara proaktif membantu Anda mengoptimalkan performa, menurunkan biaya, dan berinovasi dengan lebih cepat. AWS Business Support memberikan manfaat tambahan, termasuk akses ke AWS Trusted Advisor dan AWS Personal Health Dashboard dan waktu respons yang lebih cepat.
4. Dokumentasikan rencana dukungan di alat manajemen pengetahuan Anda. Sertakan cara untuk meminta dukungan, siapa yang harus diberi tahu jika kasus dukungan diajukan, dan cara untuk melapor selama insiden. Wiki merupakan mekanisme yang bagus untuk memungkinkan semua orang membuat pembaruan yang diperlukan pada dokumentasi ketika mereka mengetahui tentang perubahan untuk mendukung proses atau kontak.

Tingkat upaya untuk rencana implementasi: Rendah. Sebagian besar vendor perangkat lunak dan layanan menawarkan pilihan rencana dukungan. Mendokumentasikan dan berbagi praktik terbaik terkait dukungan di sistem manajemen pengetahuan Anda akan memastikan tim Anda mengetahui tindakan yang harus dilakukan jika ada masalah produksi.

Sumber daya

Praktik terbaik terkait:

- OPS02-BP02 Proses dan Prosedur memiliki pemilik teridentifikasi

Dokumen terkait:

- AWS Dukungan Plans

Layanan terkait:

- AWS Business Support
- AWS Enterprise Support

Jalankan

Pertanyaan

- OPS 8. Bagaimana cara memanfaatkan observabilitas beban kerja di organisasi Anda?
- OPS 9. Bagaimana cara memahami kondisi operasi Anda?
- OPS 10. Bagaimana cara mengelola peristiwa operasi dan beban kerja?

OPS 8. Bagaimana cara memanfaatkan observabilitas beban kerja di organisasi Anda?

Pastikan kondisi beban kerja yang optimal dengan memanfaatkan observabilitas. Manfaatkan metrik, log, dan jejak yang relevan untuk mendapatkan pandangan komprehensif tentang performa beban kerja Anda dan mengatasi masalah secara efisien.

Praktik terbaik

- OPS08-BP01 Menganalisis metrik beban kerja
- OPS08-BP02 Menganalisis log beban kerja
- OPS08-BP03 Menganalisis jejak beban kerja
- OPS08-BP04 Membuat peringatan yang dapat ditindaklanjuti
- OPS08-BP05 Membuat dasbor

OPS08-BP01 Menganalisis metrik beban kerja

Setelah mengimplementasikan telemetri aplikasi, analisis metrik yang dikumpulkan secara rutin. Latensi, permintaan, kesalahan, dan kapasitas (atau kuota) memang memberikan wawasan tentang performa sistem, tetapi memprioritaskan peninjauan metrik hasil bisnis adalah hal yang sangat

penting. Ini memastikan Anda mengambil keputusan berbasis data yang selaras dengan tujuan bisnis Anda.

Hasil yang diinginkan: Wawasan akurat tentang performa beban kerja yang mendorong keputusan berdasarkan informasi data, sehingga memastikan keselarasan dengan tujuan bisnis.

Antipola umum:

- Menganalisis metrik secara terpisah tanpa mempertimbangkan dampaknya terhadap hasil bisnis.
- Ketergantungan berlebihan pada metrik teknis sambil mengesampingkan metrik bisnis.
- Peninjauan metrik jarang dilakukan, sehingga peluang pengambilan keputusan waktu nyata terlewatkan.

Manfaat menjalankan praktik terbaik ini:

- Peningkatan pemahaman tentang korelasi antara performa teknis dan hasil bisnis.
- Perbaikan proses pengambilan keputusan yang berlandaskan data waktu nyata.
- Identifikasikan dan mitigasi masalah secara proaktif sebelum hasil bisnis terkena dampaknya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Manfaatkan alat seperti Amazon CloudWatch untuk melakukan analisis metrik. Layanan AWS seperti AWS Cost Anomaly Detection dan Amazon DevOps Guru dapat digunakan untuk mendeteksi anomali, terutama ketika ambang batas statis tidak diketahui atau ketika pola perilaku lebih cocok untuk deteksi anomali.

Langkah implementasi

1. Analisis dan tinjau: Tinjau dan tafsirkan metrik beban kerja Anda secara rutin.
    a. Prioritaskan metrik hasil bisnis daripada metrik teknis murni.
    b. Pahami signifikansi lonjakan, penurunan, atau pola dalam data Anda.
2. Manfaatkan Amazon CloudWatch: Gunakan Amazon CloudWatch untuk mendapatkan tampilan terpusat dan analisis mendalam.
    a. Konfigurasikan dasbor CloudWatch untuk memvisualisasikan metrik Anda dan membandingkannya dari waktu ke waktu.

   b. Gunakan persentil di CloudWatch untuk mendapatkan pandangan yang jelas tentang distribusi metrik, yang dapat membantu dalam mendefinisikan SLA dan memahami penyimpangan.
   c. Siapkan AWS Cost Anomaly Detection untuk mengidentifikasi pola yang tidak biasa tanpa bergantung pada ambang batas statis.
   d. Implementasikan observabilitas lintas akun CloudWatch untuk memantau dan memecahkan masalah aplikasi yang terjadi di beberapa akun di dalam suatu Wilayah.
   e. Gunakan Wawasan Metrik CloudWatch untuk mengkueri dan menganalisis data metrik di seluruh akun dan Wilayah, sehingga tren dan anomali dapat terdeteksi.
   f. Terapkan CloudWatch Metric Math untuk mengubah, menggabungkan, atau melakukan perhitungan pada metrik Anda untuk mendapatkan wawasan yang lebih mendalam.
3. Gunakan Amazon DevOps Guru: Sertakan Amazon DevOps Guru untuk memanfaatkan deteksi anomali yang disempurnakan dengan machine learning-nya untuk mengidentifikasi tanda-tanda awal masalah operasional untuk aplikasi nirserver Anda dan memperbaikinya sebelum berdampak pada pelanggan Anda.
4. Lakukan optimalisasi berdasarkan wawasan: Ambil keputusan cerdas berdasarkan analisis metrik Anda untuk menyesuaikan dan meningkatkan beban kerja Anda.

Tingkat upaya untuk Rencana Implementasi: Sedang

Sumber daya

Praktik terbaik terkait:

- OPS04-BP01 Identifikasikan indikator performa utama
- OPS04-BP02 Mengimplementasikan telemetri aplikasi

Dokumen terkait:

- The Wheel Blog - Menekankan pentingnya peninjauan metrik secara terus-menerus
- Persentil itu penting
- Menggunakan AWS Cost Anomaly Detection
- observabilitas lintas akun CloudWatch
- Mengkueri metrik dengan Wawasan Metrik CloudWatch

Video terkait:

- Mengaktifkan Observabilitas Lintas Akun di Amazon CloudWatch
- Pengantar Amazon DevOps Guru
- Menganalisis Metrik secara Berkelanjutan Menggunakan AWS Cost Anomaly Detection

Contoh terkait:

- Lokakarya One Observability
- Mendapatkan wawasan operasional dengan AIOps menggunakan Amazon DevOps Guru

OPS08-BP02 Menganalisis log beban kerja

Menganalisis log beban kerja secara rutin sangatlah penting untuk mendapatkan pemahaman yang lebih mendalam tentang aspek-aspek operasional aplikasi Anda. Dengan memilah-milah, memvisualisasikan, dan menafsirkan data log secara efisien, Anda dapat terus mengoptimalkan performa dan keamanan aplikasi.

Hasil yang diinginkan: Wawasan yang kaya tentang perilaku dan operasi aplikasi yang berasal dari analisis log yang menyeluruh, sehingga memastikan deteksi dan mitigasi masalah yang proaktif.

Antipola umum:

- Mengabaikan analisis log sampai masalah kritis muncul.
- Tidak menggunakan rangkaian alat lengkap yang tersedia untuk analisis log, sehingga wawasan kritis terlewatkan.
- Hanya mengandalkan tinjauan log manual tanpa memanfaatkan kemampuan otomatisasi dan kueri.

Manfaat menjalankan praktik terbaik ini:

- Identifikasikan kemacetan operasional, ancaman keamanan, dan masalah potensial lain secara proaktif.
- Pemanfaatan data log yang efisien untuk optimalisasi aplikasi berkelanjutan.
- Peningkatan pemahaman tentang perilaku aplikasi, sehingga membantu upaya debugging dan pemecahan masalah.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Amazon CloudWatch Logs adalah alat yang ampuh untuk analisis log. Fitur terintegrasi seperti Wawasan dan Wawasan Kontributor CloudWatch Logs membuat proses perolehan informasi yang bermakna dari log menjadi intuitif dan efisien.

Langkah implementasi

1. Siapkan CloudWatch Logs: Konfigurasikan aplikasi dan layanan untuk mengirim log ke CloudWatch Logs.
2. Siapkan Wawasan CloudWatch Logs: Gunakan CloudWatch Logs Insights untuk mencari dan menganalisis data log Anda secara interaktif.
   a. Buat kueri untuk mengekstrak pola, memvisualisasikan data log, dan memperoleh wawasan yang dapat ditindaklanjuti.
3. Manfaatkan Wawasan Kontributor Gunakan CloudWatch Contributor Insights untuk mengidentifikasi sumber data teratas dalam dimensi kardinalitas tinggi seperti alamat IP atau agen pengguna.
4. Implementasikan filter metrik CloudWatch Logs: Konfigurasikan filter metrik log CloudWatch untuk mengubah data log menjadi metrik yang dapat ditindaklanjuti. Ini memungkinkan Anda untuk mengatur alarm atau menganalisis pola lebih lanjut.
5. Tinjauan dan penyempurnaan rutin: Tinjau strategi analisis log Anda secara berkala untuk menangkap semua informasi yang relevan dan terus mengoptimalkan performa aplikasi.

Tingkat upaya untuk rencana implementasi: Sedang.

Sumber daya

Praktik terbaik terkait:

- OPS04-BP01 Identifikasikan indikator performa utama
- OPS04-BP02 Mengimplementasikan telemetri aplikasi
- OPS08-BP01 Menganalisis metrik beban kerja

Dokumen terkait:

- Menganalisis Data Log dengan Wawasan CloudWatch Logs

- Menggunakan Wawasan Kontributor CloudWatch
- Membuat dan Mengelola Filter Metrik Log CloudWatch Logs

Video terkait:

- Menganalisis Data Log dengan Wawasan CloudWatch Logs
- Menggunakan Wawasan Kontributor CloudWatch untuk Menganalisis Data Kardinalitas Tinggi

Contoh terkait:

- Contoh Kueri CloudWatch Logs
- Lokakarya One Observability

OPS08-BP03 Menganalisis jejak beban kerja

Menganalisis data jejak sangatlah penting untuk mencapai pandangan yang komprehensif tentang perjalanan operasional aplikasi. Dengan memvisualisasikan dan memahami interaksi antara berbagai komponen, performa dapat disesuaikan, kemacetan dapat diidentifikasi, dan pengalaman pengguna dapat ditingkatkan.

Hasil yang diinginkan: Dapatkan visibilitas yang jelas tentang operasi terdistribusi aplikasi Anda, sehingga memungkinkan penyelesaian masalah yang lebih cepat dan pengalaman pengguna yang disempurnakan.

Antipola umum:

- Mengabaikan data jejak, dan hanya mengandalkan log serta metrik.
- Tidak mengorelasikan data jejak dengan log terkait.
- Mengabaikan metrik yang berasal dari jejak, seperti latensi dan tingkat kesalahan.

Manfaat menjalankan praktik terbaik ini:

- Perbaiki kualitas pemecahan masalah dan kurangi rata-rata waktu penyelesaian (MTTR).
- Dapatkan wawasan tentang dependensi dan dampaknya.
- Identifikasikan dan perbaikan masalah performa secara cepat.
- Memanfaatkan metrik yang berasal dari jejak untuk pengambilan keputusan yang bijak.

- Pengalaman pengguna yang ditingkatkan melalui interaksi komponen yang dioptimalkan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

AWS X-Ray menawarkan rangkaian fitur komprehensif untuk analisis data jejak, sehingga menyediakan pandangan yang menyeluruh tentang interaksi layanan, memantau aktivitas pengguna, dan mendeteksi masalah performa. Fitur seperti ServiceLens, Wawasan X-Ray, Analitik X-Ray, dan Amazon DevOps Guru meningkatkan kedalaman wawasan yang dapat ditindaklanjuti yang berasal dari data jejak.

Langkah implementasi

Langkah-langkah berikut ini menawarkan pendekatan terstruktur untuk menerapkan analisis data jejak secara efektif menggunakan layanan AWS:

1. Integrasikan AWS X-Ray: Pastikan X-Ray terintegrasi dengan aplikasi Anda untuk menangkap data jejak.
2. Analisis metrik X-Ray: Selidiki metrik yang berasal dari jejak X-Ray seperti latensi, tingkat permintaan, tingkat kesalahan, dan distribusi waktu respons menggunakan peta layanan untuk memantau kondisi aplikasi.
3. Gunakan ServiceLens: Manfaatkan peta ServiceLens untuk meningkatkan observabilitas layanan dan aplikasi Anda. Fitur ini memungkinkan tampilan jejak, metrik, log, alarm, dan informasi kondisi lainnya secara terpadu.
4. Aktifkan Wawasan X-Ray:
   a. Aktifkan Wawasan X-Ray untuk deteksi anomali dalam jejak secara otomatis.
   b. Periksa wawasan untuk menentukan pola dan memastikan akar masalah, seperti peningkatan tingkat kesalahan atau latensi.
   c. Pelajari lini waktu wawasan untuk mendapatkan analisis kronologis pada masalah yang terdeteksi.
5. Gunakan Analitik X-Ray: Analitik X-Ray memungkinkan Anda menjelajahi data jejak, menentukan pola, dan mengekstrak wawasan secara menyeluruh.
6. Gunakan grup di X-Ray: Buat grup di X-Ray untuk memfilter jejak berdasarkan kriteria seperti latensi tinggi, sehingga memungkinkan analisis yang lebih tertarget.
7. Sertakan Amazon DevOps Guru: Libatkan Amazon DevOps Guru untuk mendapatkan manfaat dari model machine learning yang mengenali anomali operasional dalam jejak.

8. Gunakan CloudWatch Synthetics: Gunakan CloudWatch Synthetics untuk membuat canary untuk terus memantau titik akhir dan alur kerja Anda. Canary ini dapat terintegrasi dengan X-Ray untuk menyediakan data jejak untuk analisis aplikasi yang sedang diuji secara mendalam.
9. Gunakan Pemantauan Pengguna Nyata (RUM): Dengan AWS X-Ray dan CloudWatch RUM, Anda dapat menganalisis dan men-debug jalur permintaan mulai dari pengguna akhir aplikasi Anda hingga layanan AWS terkelola di hilir. Ini membantu Anda mengidentifikasi tren latensi dan kesalahan yang berdampak pada pengguna Anda.
10. Korelasikan dengan log: Korelasikan data jejak dengan log terkait di dalam tampilan jejak X-Ray untuk perspektif yang mendetail tentang perilaku aplikasi. Ini memungkinkan Anda melihat peristiwa log yang terkait langsung dengan transaksi yang dilacak.

Tingkat upaya untuk rencana implementasi: Sedang.

Sumber daya

Praktik terbaik terkait:

- OPS08-BP01 Menganalisis metrik beban kerja
- OPS08-BP02 Menganalisis log beban kerja

Dokumen terkait:

- Menggunakan ServiceLens untuk Memantau Kondisi Aplikasi
- Menjelajahi Data Jejak dengan X-Ray Analytics
- Mendeteksi Anomali di dalam Jejak dengan Wawasan X-Ray
- Pemantauan Berkelanjutan dengan CloudWatch Synthetics

Video terkait:

- Menganalisis dan Men-debug Aplikasi Menggunakan Amazon CloudWatch Synthetics dan AWS X-Ray
- Gunakan Wawasan AWS X-Ray

Contoh terkait:

- Lokakarya One Observability

- Mengimplementasikan X-Ray dengan AWS Lambda
- Templat Canary CloudWatch Synthetics

OPS08-BP04 Membuat peringatan yang dapat ditindaklanjuti

Sangat penting mendeteksi dan merespons penyimpangan dalam perilaku aplikasi Anda segera. Lebih penting lagi adalah mengenali ketika hasil yang didasarkan pada indikator kinerja utama (KPI) terpapar risiko atau ketika anomali tak terduga muncul. Mendasarkan peringatan pada KPI memastikan bahwa sinyal yang Anda terima berkaitan langsung dengan dampak bisnis atau operasional. Pendekatan terhadap peringatan yang dapat ditindaklanjuti ini mempromosikan respons proaktif dan membantu mempertahankan performa dan keandalan sistem.

Hasil yang diinginkan: Terima peringatan yang tepat waktu, relevan, dan dapat ditindaklanjuti untuk identifikasi dan mitigasi potensi masalah dengan cepat, terutama ketika hasil KPI berisiko.

Antipola umum:

- Mengonfigurasi terlalu banyak peringatan non-kritis, yang mengakibatkan kewalahan.
- Tidak memprioritaskan peringatan berdasarkan KPI, sehingga dampak masalah terhadap bisnis menjadi sulit dipahami.
- Mengabaikan penanganan akar masalah, yang berimbas pada peringatan yang repetitif untuk masalah yang sama.

Manfaat menjalankan praktik terbaik ini:

- Berkurangnya kewalahan akibat peringatan dengan memusatkan perhatian pada peringatan yang dapat ditindaklanjuti dan relevan.
- Waktu aktif dan keandalan sistem yang lebih baik melalui deteksi dan mitigasi masalah secara proaktif.
- Kolaborasi tim yang disempurnakan dan penyelesaian masalah yang lebih cepat melalui integrasi alat-alat peringatan dan komunikasi populer.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Untuk membuat mekanisme peringatan yang efektif, sangat penting untuk menggunakan metrik, log, dan data jejak yang menandai kapan hasil yang didasarkan pada KPI mengandung risiko atau terdapat anomali yang terdeteksi.

Langkah implementasi

1. Tentukan indikator kinerja utama (KPI): Identifikasikan KPI aplikasi Anda. Peringatan harus dikaitkan dengan KPI tersebut agar mencerminkan dampak bisnis secara akurat.
2. Implementasikan deteksi anomali:
   - Gunakan AWS Cost Anomaly Detection: Siapkan AWS Cost Anomaly Detection untuk secara otomatis mendeteksi pola yang tidak biasa, sehingga memastikan peringatan hanya dihasilkan untuk anomali asli.
   - Gunakan Wawasan X-Ray:
     a. Siapkan Wawasan X-Ray untuk mendeteksi anomali dalam data jejak.
     b. Konfigurasikan notifikasi untuk Wawasan X-Ray untuk menerima peringatan tentang masalah yang terdeteksi.
   - Integrasikan dengan DevOps Guru:
     a. Manfaatkan Amazon DevOps Guru untuk kemampuan machine learning-nya dalam mendeteksi anomali operasional pada data yang ada.
     b. Buka pengaturan notifikasi di dalam DevOps Guru untuk menyiapkan peringatan anomali.
3. Implementasikan peringatan yang dapat ditindaklanjuti: Rancang peringatan yang menyediakan informasi yang memadai untuk tindakan cepat.
4. Kurangi kewalahan akibat alarm: Minimalkan peringatan non-kritis. Tim yang kewalahan dengan banyaknya peringatan yang tidak penting dapat menyebabkan terlewatkannya masalah kritis dan mengurangi efektivitas mekanisme peringatan secara keseluruhan.
5. Siapkan alarm komposit: Gunakan alarm komposit Amazon CloudWatch untuk menggabungkan beberapa alarm.
6. Integrasikan dengan alat peringatan: Sertakan alat-alat seperti Ops Genie dan PagerDuty.
7. Libatkan Amazon Q Developer in chat applications Integrasikan Amazon Q Developer in chat applicationsuntuk mengirimkan peringatan ke Chime, Microsoft Teams, dan Slack.
8. Buat peringatan berdasarkan log: Gunakan filter metrik log di CloudWatch untuk membuat alarm berdasarkan peristiwa log tertentu.
9. Tinjau dan lakukan iterasi: Tinjau dan sempurnakan konfigurasi peringatan secara rutin.

Tingkat upaya untuk rencana implementasi: Sedang.

Sumber daya

Praktik terbaik terkait:

- OPS04-BP01 Identifikasikan indikator performa utama
- OPS04-BP02 Mengimplementasikan telemetri aplikasi
- OPS04-BP03 Mengimplementasikan telemetri pengalaman pengguna
- OPS04-BP04 Mengimplementasikan telemetri dependensi
- OPS04-BP05 Mengimplementasikan penelusuran terdistribusi
- OPS08-BP01 Menganalisis metrik beban kerja
- OPS08-BP02 Menganalisis log beban kerja
- OPS08-BP03 Menganalisis jejak beban kerja

Dokumen terkait:

- Menggunakan Alarm Amazon CloudWatch
- Membuat alarm komposit
- Membuat alarm CloudWatch berdasarkan deteksi anomali
- Notifikasi DevOps Guru
- Notifikasi Wawasan X-Ray
- Pantau, operasikan, dan pecahkan masalah sumber daya AWS Anda dengan ChatOps interaktif
- Panduan Integrasi Amazon CloudWatch | PagerDuty
- Integrasikan OPSGenie dengan Amazon CloudWatch

Video terkait:

- Membuat Alarm Komposit di Amazon CloudWatch
- Ikhtisar Amazon Q Developer in chat applications
- AWS on Air ft. Perintah Mutatif di Amazon Q Developer in chat applications

Contoh terkait:

- Alarm, manajemen insiden, dan remediasi di cloud dengan Amazon CloudWatch
- Tutorial: Membuat aturan Amazon EventBridge yang mengirimkan notifikasi ke Amazon Q Developer in chat applications
- Lokakarya One Observability

OPS08-BP05 Membuat dasbor

Dasbor adalah tampilan yang berpusat pada manusia tentang data telemetri beban kerja Anda. Meskipun menyediakan antarmuka visual yang vital, dasbor tidak boleh menggantikan mekanisme peringatan, melainkan hanya melengkapinya. Ketika dibuat dengan cermat, dasbor tidak hanya dapat menawarkan wawasan cepat tentang kondisi dan kinerja sistem, tetapi juga dapat menyajikan informasi waktu nyata kepada pemangku kepentingan tentang hasil bisnis dan dampak masalah.

Hasil yang diinginkan: Wawasan yang jelas dan dapat ditindaklanjuti tentang kondisi sistem dan bisnis menggunakan representasi visual.

Antipola umum:

- Dasbor yang terlalu rumit dengan terlalu banyak metrik.
- Mengandalkan dasbor tanpa peringatan untuk deteksi anomali.
- Tidak memperbarui dasbor seiring perkembangan beban kerja.

Manfaat menjalankan praktik terbaik ini:

- Visibilitas langsung tentang metrik sistem kritis dan KPI.
- Komunikasi dan pemahaman pemangku kepentingan yang ditingkatkan.
- Wawasan cepat tentang dampak masalah operasional.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Dasbor yang berpusat pada bisnis

Dasbor yang disesuaikan dengan KPI bisnis melibatkan lebih banyak pemangku kepentingan. Meskipun individu ini mungkin tidak tertarik pada metrik sistem, mereka tertarik untuk memahami implikasi bisnis dari angka-angka ini. Dasbor yang berpusat pada bisnis memastikan semua metrik

teknis dan operasional yang dipantau dan dianalisis selaras dengan tujuan bisnis menyeluruh. Penyelarasan ini memberikan kejelasan, memastikan semua orang memiliki pemahaman yang sama mengenai hal-hal yang penting dan hal-hal yang tidak penting. Selain itu, dasbor yang menyoroti KPI bisnis cenderung lebih mudah ditindaklanjuti. Pemangku kepentingan dapat dengan cepat memahami kondisi operasi, area yang perlu diperhatikan, dan potensi dampak terhadap hasil bisnis.

Dengan mempertimbangkan hal ini, saat membuat dasbor Anda, pastikan ada keseimbangan antara metrik teknis dan KPI bisnis. Keduanya penting tetapi melayani audiens yang berbeda. Idealnya, Anda harus memiliki dasbor yang memberikan pandangan menyeluruh tentang kondisi dan performa sistem sekaligus menekankan hasil bisnis utama serta implikasinya.

Dasbor Amazon CloudWatch adalah halaman beranda yang dapat dikustomisasi di konsol CloudWatch yang dapat Anda gunakan untuk memantau sumber daya dalam satu tampilan, bahkan sumber daya yang tersebar di Wilayah AWS dan akun yang berbeda-beda.

Langkah implementasi

1. Buat dasbor dasar: Buat dasbor baru di CloudWatch, dan berikan nama yang jelas.
2. Gunakan widget Markdown: Sebelum menjelajahi metrik, gunakan widget Markdown untuk menambahkan konteks tekstual di bagian atas dasbor Anda. Widget ini akan menjelaskan cakupan dasbor, tingkat pentingnya metrik yang ditampilkan, dan juga dapat diisi dengan tautan ke dasbor serta alat pemecahan masalah lainnya.
3. Buat variabel dasbor: Gabungkan variabel dasbor seperlunya agar tampilan dasbor menjadi dinamis dan fleksibel.
4. Buat widget metrik: Tambahkan widget metrik untuk memvisualisasikan berbagai metrik yang dihasilkan oleh aplikasi Anda, lalu sesuaikan semua widget agar efektif menampilkan kondisi sistem dan hasil bisnis.
5. Kueri Wawasan Log: Manfaatkan CloudWatch Logs Insights untuk mendapatkan metrik yang dapat ditindaklanjuti dari log Anda dan menampilkan wawasan ini di dasbor Anda.
6. Siapkan alarm: Integrasikan alarm CloudWatch ke dasbor agar Anda dapat mengetahui dengan cepat metrik yang melanggar ambang batasnya.
7. Gunakan Contributor Insights: Sertakan CloudWatch Contributor Insights untuk menganalisis bidang dengan kardinalitas tinggi dan mendapatkan pemahaman yang lebih jelas tentang kontributor utama sumber daya Anda.
8. Rancang widget kustom: Untuk kebutuhan spesifik yang tidak terpenuhi oleh widget standar, pertimbangkan untuk membuat widget kustom. Widget kustom dapat menarik dari berbagai sumber data atau menyajikan data dengan cara yang unik.

9. Lakukan iterasi dan sempurnakan: Saat aplikasi Anda berkembang, tinjau kembali dasbor Anda secara teratur untuk memastikan relevansinya.

Sumber daya

Praktik terbaik terkait:

- OPS04-BP01 Identifikasikan indikator performa utama
- OPS08-BP01 Menganalisis metrik beban kerja
- OPS08-BP02 Menganalisis log beban kerja
- OPS08-BP03 Menganalisis jejak beban kerja
- OPS08-BP04 Membuat peringatan yang dapat ditindaklanjuti

Dokumen terkait:

- Membangun Dasbor untuk Visibilitas Operasional
- Menggunakan Dasbor Amazon CloudWatch

Video terkait:

- Membuat Dasbor CloudWatch Lintas Akun & Lintas Wilayah
- AWS re:Invent 2021 - Mendapatkan visibilitas korporasi dengan dasbor operasional AWS Cloud

Contoh terkait:

- Lokakarya One Observability
- Pemantauan Aplikasi dengan Amazon CloudWatch

OPS 9. Bagaimana cara memahami kondisi operasi Anda?

Tetapkan, catat, dan analisis metrik operasi untuk mendapatkan visibilitas peristiwa operasi sehingga Anda dapat mengambil tindakan yang tepat.

Praktik terbaik

- OPS09-BP01 Mengukur sasaran operasi dan KPI dengan metrik

- OPS09-BP02 Mengomunikasikan status dan tren untuk memastikan visibilitas beroperasi
- OPS09-BP03 Meninjau metrik operasi dan memprioritaskan perbaikan

OPS09-BP01 Mengukur sasaran operasi dan KPI dengan metrik

Dapatkan sasaran dan KPI yang menentukan keberhasilan operasi dari organisasi Anda dan pastikan metrik mencerminkan hal ini. Tetapkan garis acuan sebagai titik referensi dan evaluasi kembali secara rutin. Kembangkan mekanisme untuk mengumpulkan metrik-metrik tersebut dari tim untuk dievaluasi.

Hasil yang diinginkan:

- Sasaran dan KPI untuk tim operasi organisasi telah dipublikasikan dan dibagikan.
- Metrik yang mencerminkan KPI ini ditetapkan. Di antara contohnya adalah:
  - Kedalaman antrean tiket atau rata-rata umur tiket
  - Jumlah tiket yang dikelompokkan berdasarkan jenis masalah
  - Waktu yang dihabiskan untuk mengurusi masalah dengan atau tanpa prosedur operasi standar (SOP)
  - Jumlah waktu yang dihabiskan untuk pulih dari push kode yang gagal
  - Volume panggilan

Antipola umum:

- Tenggat waktu deployment tidak terpenuhi karena developer disibukkan dengan tugas pemecahan masalah. Tim pengembangan menuntut lebih banyak personel, tetapi tidak dapat mengukur berapa orang yang mereka butuhkan karena waktu yang tersita tidak dapat diukur.
- Meja Tingkat 1 disiapkan untuk menangani panggilan pengguna. Seiring waktu, makin banyak beban kerja ditambahkan, tetapi tidak ada personel yang dialokasikan ke meja Tingkat 1 tersebut. Kepuasan pelanggan sangat rendah karena waktu panggilan meningkat dan masalah berlarut-larut tanpa penyelesaian, tetapi manajemen tidak melihat indikator permasalahan ini, sehingga tidak ada tindakan yang dilakukan.
- Beban kerja yang bermasalah diserahkan kepada tim operasi terpisah untuk pemeliharaan. Tidak seperti beban kerja lainnya, beban kerja tersebut tidak dilengkapi dengan dokumentasi dan runbook yang tepat. Akibatnya, tim menghabiskan waktu lebih lama untuk memecahkan masalah dan mengurusi kegagalan. Namun, tidak ada metrik yang mendokumentasikan hal ini, sehingga akuntabilitas menjadi sulit.

Manfaat menjalankan praktik terbaik ini: Ketika pemantauan beban kerja menunjukkan status aplikasi dan layanan kita, tim operasi pemantauan memberi pemilik wawasan tentang perubahan pada pemakai beban kerja tersebut, seperti perubahan kebutuhan bisnis. Ukur efektivitas tim-tim tersebut dan evaluasi mereka berdasarkan sasaran bisnis dengan membuat metrik yang dapat mencerminkan status operasi. Metrik dapat menyoroti masalah dukungan atau mengidentifikasi ketika terjadi pergeseran dari target tingkat layanan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Jadwalkan waktu dengan para pemimpin bisnis dan pemangku kepentingan untuk menentukan sasaran layanan secara keseluruhan. Tentukan tugas apa saja yang seharusnya dijalankan oleh berbagai tim operasi dan tantangan apa yang dapat mereka hadapi. Dengan menggunakan hal ini, lakukan curah pendapat indikator kinerja utama (KPI) yang mungkin mencerminkan semua sasaran operasi ini. Indikator tersebut mungkin berupa kepuasan pelanggan, waktu dari konsepsi fitur hingga deployment, waktu penyelesaian masalah rata-rata, dan lain-lain.

Berpatokan pada KPI, identifikasi metrik dan sumber data yang mungkin paling mencerminkan semua sasaran ini. Kepuasan pelanggan dapat berupa kombinasi dari berbagai metrik seperti waktu tunggu atau respons panggilan, skor kepuasan, dan jenis-jenis masalah yang disampaikan. Waktu deployment mungkin merupakan jumlah waktu yang diperlukan untuk pengujian dan deployment, serta perbaikan pasca-deployment yang perlu ditambahkan. Statistik yang menunjukkan waktu yang dihabiskan untuk berbagai jenis masalah (atau jumlah masalah tersebut) dapat memberikan wawasan tentang bagian yang memerlukan upaya tertarget.

Sumber daya

Dokumen terkait:

- Quick - Menggunakan KPI
- Amazon CloudWatch - Menggunakan Metrik
- Membangun Dasbor
- Cara melacak KPI pengoptimalan biaya Anda dengan Dasbor KPI

OPS09-BP02 Mengomunikasikan status dan tren untuk memastikan visibilitas beroperasi

Anda perlu mengetahui keadaan operasi Anda dan arah trennya untuk mengidentifikasi kapan hasil mungkin berisiko, apakah pekerjaan tambahan dapat didukung, atau efek perubahan terhadap tim

Anda. Selama peristiwa operasi, halaman status yang dapat dijadikan acuan oleh pengguna dan tim operasi untuk mendapatkan informasi dapat mengurangi tekanan pada saluran komunikasi dan menyebarkan informasi secara proaktif.

Hasil yang diinginkan:

- Pemimpin operasi memiliki wawasan sekilas untuk melihat volume panggilan seperti apa yang sedang dioperasikan oleh tim mereka dan upaya apa yang mungkin sedang dilakukan, seperti deployment.
- Peringatan disebarkan kepada pemangku kepentingan dan komunitas pengguna ketika terjadi dampak terhadap operasi normal.
- Kepemimpinan dan pemangku kepentingan organisasi dapat memeriksa halaman status sebagai respons terhadap peringatan atau dampak, dan memperoleh informasi seputar peristiwa operasional, seperti titik kontak, informasi tiket, dan perkiraan waktu pemulihan.
- Laporan tersedia bagi para pemimpin dan pemangku kepentingan lainnya untuk menunjukkan statistik operasi seperti volume panggilan selama periode waktu tertentu, skor kepuasan pengguna, jumlah tiket tertunda, dan usia mereka.

Antipola umum:

- Terdapat beban kerja yang tidak aktif, sehingga sebuah layanan menjadi tidak tersedia. Volume panggilan melonjak karena para pengguna ingin mengetahui apa yang terjadi. Manajer menambah volume tersebut dengan permintaan informasinya tentang siapa yang mengurusi masalah. Berbagai tim operasi melipatgandakan upaya untuk melakukan penyelidikan.
- Keinginan untuk kemampuan baru menyebabkan beberapa personel dialihkan ke upaya rekayasa. Tidak ada pengisian ulang yang disediakan, dan waktu penyelesaian masalah melonjak. Informasi ini tidak ditangkap, dan pimpinan baru menyadari hal ini setelah beberapa minggu dan pengguna menyampaikan ketidakpuasan.

Manfaat menjalankan praktik terbaik ini: Selama peristiwa operasional yang berdampak pada bisnis, banyak waktu dan tenaga yang bisa terbuang untuk meminta informasi dari berbagai tim yang sedang berusaha memahami situasinya. Dengan membuat halaman status dan dasbor yang disebarluaskan, pemangku kepentingan dapat dengan cepat memperoleh informasi seperti apakah ada masalah yang terdeteksi, siapa yang memimpin penanganan masalah tersebut, atau kapan operasi diperkirakan akan kembali normal. Dengan begitu, anggota tim terhindar dari membuang-

buang waktu untuk mengomunikasikan status kepada orang lain dan lebih berkonsentrasi menangani masalah.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Buat dasbor yang menunjukkan metrik utama saat ini untuk tim operasi Anda, dan buat dasbor tersebut mudah diakses oleh pemimpin operasi serta manajemen.

Buat halaman status yang dapat diperbarui dengan cepat untuk menunjukkan apabila insiden atau peristiwa sedang berlangsung, siapa yang bertanggung jawab, dan siapa yang mengoordinasikan respons. Bagikan langkah atau solusi apa pun yang harus dipertimbangkan pengguna di halaman ini, dan sebarkan luaskan lokasinya. Imbau pengguna untuk memeriksa lokasi ini terlebih dahulu ketika mereka dihadapkan dengan masalah yang tidak diketahui.

Kumpulkan dan sediakan laporan yang menunjukkan kondisi operasi dari waktu ke waktu, dan distribusikan hal ini kepada para pemimpin dan pengambil keputusan untuk menggambarkan pekerjaan operasi beserta tantangan dan kebutuhan.

Bagikan kepada tim metrik dan laporan yang paling mencerminkan sasaran dan KPI dan bagian yang paling menerima pengaruhnya dalam mendorong perubahan. Luangkan waktu khusus untuk aktivitas ini untuk meningkatkan pentingnya operasi di dalam tim dan antartim.

Sumber daya

Dokumen terkait:

- Mengukur Kemajuan
- Membangun dasbor untuk visibilitas operasi

Solusi terkait:

- Operasi Data

OPS09-BP03 Meninjau metrik operasi dan memprioritaskan perbaikan

Menyisihkan waktu dan sumber daya khusus untuk meninjau keadaan operasi memastikan bahwa pelayanan lini bisnis sehari-hari tetap menjadi prioritas. Kumpulkan para pemimpin operasi dan

pemangku kepentingan untuk secara rutin meninjau metrik, menegaskan kembali atau memodifikasi sasaran dan tujuan, dan memprioritaskan perbaikan.

Hasil yang diinginkan:

- Para pemimpin operasi dan staf secara rutin bertemu untuk meninjau metrik selama periode pelaporan tertentu. Tantangan dikomunikasikan, keberhasilan dirayakan, dan pelajaran yang dipetik dibagikan.
- Pemangku kepentingan dan pemimpin bisnis secara rutin diberi pengarahan tentang keadaan operasi dan diminta memberikan masukan mengenai sasaran, KPI, dan inisiatif masa depan. Kompromi antara pelayanan, operasi, dan pemeliharaan dibahas dan dimasukkan ke dalam konteks.

Antipola umum:

- Sebuah produk baru diluncurkan, tetapi tim operasi Tingkat 1 dan Tingkat 2 tidak cukup terlatih untuk mendukung atau diberikan staf tambahan. Metrik yang menunjukkan penurunan waktu resolusi tiket dan peningkatan volume insiden tidak terlihat oleh para pemimpin. Tindakan diambil beberapa minggu kemudian ketika jumlah langganan mulai turun karena ketidakpuasan pengguna yang beralih ke platform lain.
- Proses manual untuk melakukan pemeliharaan pada beban kerja telah berlangsung sejak lama. Meskipun sudah ada keinginan untuk melakukan otomatisasi, prioritas yang diberikan rendah mengingat rendahnya signifikansi sistem. Namun seiring waktu, sistem menjadi makin penting dan sekarang proses manual ini menyita sebagian besar waktu operasional. Tidak ada sumber daya yang dijadwalkan untuk menyediakan peningkatan peralatan untuk operasi, sehingga menyebabkan kelelahan pada staf saat beban kerja meningkat. Pimpinan menyadari hal ini setelah ada laporan bahwa para staf beralih ke kompetitor.

Manfaat menjalankan praktik terbaik ini: Beberapa organisasi kesulitan untuk mengalokasikan waktu dan perhatian yang sama untuk pemberian layanan dan produk atau penawaran baru. Ketika ini terjadi, lini bisnis dapat menderita karena tingkat layanan yang diharapkan perlahan-lahan memburuk. Alasannya adalah operasi tidak berubah dan berkembang sesuai dengan perkembangan bisnis, dan bisa segera tertinggal. Tanpa peninjauan rutin pada wawasan yang dikumpulkan oleh operasi, risiko terhadap bisnis mungkin baru terlihat ketika semua sudah terlambat. Dengan pengalokasian waktu untuk meninjau metrik dan prosedur baik di antara staf operasi maupun dengan pimpinan, peran penting yang dimiliki oleh operasi terus dapat dilihat, dan risiko dapat diidentifikasi jauh sebelum mencapai tingkat kritis. Tim operasi mendapatkan wawasan yang

lebih baik tentang perubahan dan inisiatif bisnis yang akan datang, sehingga upaya proaktif dapat dilakukan. Visibilitas kepemimpinan ke dalam metrik operasi menunjukkan peran yang dimiliki oleh tim operasional dalam kepuasan pelanggan, baik internal maupun eksternal, dan memungkinkan mereka mempertimbangkan pilihan prioritas dengan lebih baik, atau memastikan bahwa operasional memiliki waktu dan sumber daya untuk berubah dan berkembang seiring munculnya inisiatif bisnis dan beban kerja baru.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Luangkan waktu khusus untuk meninjau metrik operasi antara pemangku kepentingan dan tim operasional dan meninjau data laporan. Masukkan laporan-laporan tersebut ke dalam konteks tujuan dan sasaran organisasi untuk menentukan apakah semuanya terpenuhi. Identifikasikan sumber ambiguitas di mana sasaran tidak jelas, atau di mana mungkin ada ketidaksesuaian antara apa yang diminta dan apa yang diberikan.

Identifikasikan di mana waktu, personel, dan alat dapat membantu mencapai hasil operasi. Tentukan KPI mana yang akan menerima dampaknya dan target kesuksesan apa yang harus dimiliki. Tinjau ulang secara rutin untuk memastikan operasi memiliki sumber daya yang memadai untuk mendukung lini bisnis.

Sumber daya

Dokumen terkait:

- Amazon Athena
- Metrik Amazon CloudWatch dan referensi dimensi
- Amazon Quick
- AWS Glue
- AWS Glue Data Catalog
- Kumpulkan metrik dan log dari Instans Amazon EC2 serta server on-premise dengan Agen Amazon CloudWatch
- Menggunakan metrik Amazon CloudWatch

OPS 10. Bagaimana cara mengelola peristiwa operasi dan beban kerja?

Siapkan dan validasikan prosedur untuk merespons peristiwa guna meminimalkan gangguannya pada beban kerja Anda.

Praktik terbaik

- OPS10-BP01 Menggunakan proses untuk manajemen peristiwa, insiden, dan masalah
- OPS10-BP02 Menjalankan proses untuk setiap peringatan
- OPS10-BP03 Memprioritaskan kejadian operasional berdasarkan dampaknya terhadap bisnis
- OPS10-BP04 Tetapkan jalur eskalasi
- OPS10-BP05 Membuat rencana komunikasi pelanggan untuk gangguan
- OPS10-BP06 Mengomunikasikan status melalui dasbor
- OPS10-BP07 Otomatiskan respons terhadap peristiwa

OPS10-BP01 Menggunakan proses untuk manajemen peristiwa, insiden, dan masalah

Organisasi Anda memiliki proses untuk menangani peristiwa, insiden, dan masalah. Peristiwa adalah hal-hal yang terjadi dalam beban kerja Anda, tetapi mungkin tidak memerlukan intervensi. Insiden adalah peristiwa yang memerlukan intervensi. Masalah adalah peristiwa berulang yang memerlukan intervensi atau tidak dapat diselesaikan. Anda memerlukan proses untuk mengurangi dampak peristiwa ini pada bisnis Anda dan memastikan bahwa Anda merespons dengan tepat.

Ketika insiden dan masalah terjadi pada beban kerja Anda, Anda memerlukan proses untuk menanganinya. Bagaimana Anda akan mengomunikasikan status peristiwa dengan pemangku kepentingan? Siapa yang mengawasi pelaksanaan respons? Apa alat yang Anda gunakan untuk memitigasi peristiwa? Ini adalah contoh dari beberapa pertanyaan yang perlu Anda jawab untuk memiliki proses respons yang solid.

Proses harus didokumentasikan di lokasi sentral dan tersedia bagi siapa saja yang terlibat dalam beban kerja Anda. Jika Anda tidak memiliki wiki atau penyimpanan dokumen sentral, repositori kontrol versi dapat digunakan. Anda akan terus memperbarui rencana ini seiring berkembangnya proses Anda.

Masalah merupakan kandidat untuk otomatisasi. Peristiwa ini mengambil waktu Anda yang seharusnya dihabiskan untuk berinovasi. Mulailah dengan membangun proses berulang untuk memitigasi masalah. Seiring waktu, fokuslah untuk mengotomatiskan mitigasi atau memperbaiki

masalah mendasar. Tindakan ini akan membebaskan waktu yang kemudian dapat dihabiskan untuk melakukan peningkatan dalam beban kerja Anda.

Hasil yang diinginkan: Organisasi Anda memiliki proses untuk menangani peristiwa, insiden, dan masalah. Proses ini didokumentasikan dan disimpan di lokasi sentral. Dokumentasinya akan diperbarui seiring proses ini berubah.

Antipola umum:

- Sebuah insiden terjadi pada akhir pekan dan teknisi yang berjaga tidak tahu harus melakukan tindakan apa.
- Seorang pelanggan mengirimi Anda email bahwa aplikasi Anda tidak beroperasi. Anda melakukan booting ulang server untuk memperbaikinya. Hal ini sering terjadi.
- Ada insiden yang mengharuskan banyak tim bekerja secara independen untuk mencoba menyelesaikannya.
- Deployment terjadi dalam beban kerja Anda tanpa didokumentasikan.

Manfaat menjalankan praktik terbaik ini:

- Anda memiliki jejak audit peristiwa dalam beban kerja Anda.
- Waktu Anda untuk pulih dari insiden berkurang.
- Anggota tim dapat menyelesaikan insiden dan masalah secara konsisten.
- Ada upaya yang lebih terkonsolidasi ketika menyelidiki sebuah insiden.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Menerapkan praktik terbaik ini berarti Anda melacak peristiwa beban kerja. Anda memiliki proses untuk menangani insiden dan masalah. Proses ini didokumentasikan, dibagikan, dan sering diperbarui. Masalah diidentifikasi, diprioritaskan, dan diperbaiki.

Contoh pelanggan

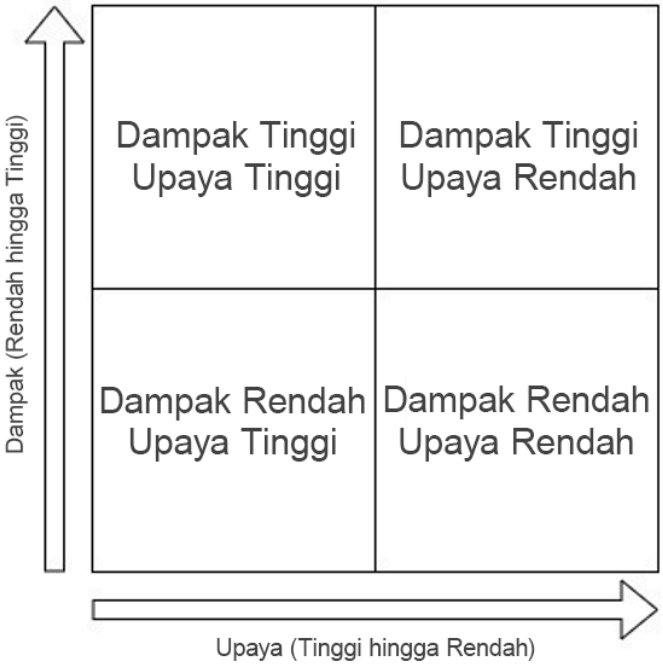
AnyCompany Retail mengkhususkan sebuah bagian dari wiki internal mereka untuk proses penanganan manajemen peristiwa, insiden, dan masalah. Semua peristiwa dikirim ke Amazon EventBridge. Masalah diidentifikasi sebagai OpsItems di AWS Systems Manager OpsCenter dan diprioritaskan untuk diperbaiki, sehingga mengurangi tenaga kerja yang tidak terdiferensiasi. Seiring

proses ini berubah, dokumentasinya diperbarui di wiki internal mereka. Mereka menggunakan Manajer Insiden AWS Systems Manager untuk mengelola insiden dan mengoordinasikan upaya mitigasi.

Langkah implementasi

1. Peristiwa
   - Lacak peristiwa yang terjadi dalam beban kerja Anda, meskipun tidak diperlukan intervensi manusia.
   - Bekerja sama dengan pemangku kepentingan beban kerja untuk mengembangkan daftar peristiwa yang harus dilacak. Beberapa contohnya adalah deployment yang diselesaikan atau patching yang berhasil.
   - Anda dapat menggunakan layanan seperti Amazon EventBridge atau Amazon Simple Notification Service untuk menghasilkan peristiwa kustom untuk pelacakan.
2. Insiden
   - Mulailah dengan mendefinisikan rencana komunikasi untuk insiden. Pemangku kepentingan mana yang harus diinformasikan? Bagaimana Anda akan terus menginformasikan mereka? Siapa yang mengawasi upaya koordinasi? Kami merekomendasikan untuk membuat saluran obrolan internal untuk komunikasi dan koordinasi.
   - Tentukan jalur eskalasi untuk tim yang mendukung beban kerja Anda, terutama jika tim ini tidak memiliki rotasi jaga. Berdasarkan tingkat dukungan Anda, Anda juga dapat mengajukan kasus ke Dukungan.
   - Buat buku playbook untuk menyelidiki insiden. Playbook Ini harus berisi rencana komunikasi dan langkah penyelidikan yang mendetail. Sertakan tindakan memeriksa Dasbor AWS Health dalam penyelidikan Anda.
   - Dokumentasikan rencana respons insiden Anda. Komunikasikan rencana manajemen insiden agar pelanggan internal dan eksternal memahami aturan pelibatan dan apa yang diharapkan dari mereka. Latih anggota tim Anda tentang cara menggunakannya.
   - Pelanggan dapat menggunakan Incident Manager untuk mengatur dan mengelola rencana respons insiden mereka.
   - Pelanggan Enterprise Support dapat meminta Lokakarya Manajemen Insiden dari Manajer Akun Teknis mereka. Lokakarya berpemandu ini akan menguji rencana respons insiden yang ada dan membantu Anda mengidentifikasi area yang perlu ditingkatkan.
3. Masalah
   - Masalah harus diidentifikasi dan dilacak dalam sistem ITSM Anda.

- Identifikasi semua masalah yang diketahui dan prioritaskan berdasarkan tingkat upaya perbaikan dan dampak pada beban kerja.



- Selesaikan masalah yang berdampak tinggi dan memerlukan tingkat upaya yang rendah terlebih dahulu. Setelah masalah tersebut diselesaikan, lanjutkan ke masalah yang termasuk dalam kuadran upaya rendah berdampak rendah.
- Anda dapat menggunakan Systems Manager OpsCenter untuk mengidentifikasi masalah ini, menyediakan runbook yang sesuai, dan melacaknya.

Tingkat upaya untuk rencana implementasi: Sedang. Anda memerlukan proses dan alat untuk menerapkan praktik terbaik ini. Dokumentasikan proses Anda dan sediakan dokumentasi ini untuk siapa saja yang terkait dengan beban kerja. Perbarui dokumentasi ini secara rutin. Anda memiliki proses untuk mengelola dan memitigasi atau memperbaiki masalah.

Sumber daya

Praktik terbaik terkait:

- OPS07-BP03 Menggunakan runbook untuk menjalankan prosedur: Masalah yang diketahui memerlukan runbook terkait agar upaya mitigasinya konsisten.
- OPS07-BP04 Menggunakan buku panduan untuk menyelidiki masalah: Insiden harus diselidiki menggunakan playbook.

- OPS11-BP02 Menjalankan analisis setelah insiden: Selalu lakukan pemeriksaan pascainsiden setelah Anda pulih dari suatu insiden.

Dokumen terkait:

- Atlassian - Manajemen insiden di era DevOps
- Panduan Respons Insiden Keamanan AWS
- Manajemen Insiden di Era DevOps dan SRE
- PagerDuty - Apa itu Manajemen Insiden?

Video terkait:

- AWS re:Invent 2020: Manajemen insiden di organisasi terdistribusi
- AWS re:Invent 2021 - Membangun aplikasi generasi baru dengan arsitektur berbasis peristiwa
- AWS Mendukung Anda | Latihan Diskusi Menjelajahi Manajemen Insiden
- Manajer Insiden AWS Systems Manager - Lokakarya Virtual AWS
- AWS What's Next bersama Incident Manager | Acara AWS

Contoh terkait:

- Lokakarya Alat Manajemen dan Tata Kelola AWS - OpsCenter
- Layanan Proaktif AWS – Lokakarya Manajemen Insiden
- Membangun aplikasi berbasis peristiwa dengan Amazon EventBridge
- Membangun arsitektur berbasis peristiwa di AWS

Layanan terkait:

- Amazon EventBridge
- Amazon SNS
- Dasbor AWS Health
- Manajer Insiden AWS Systems Manager
- AWS Systems Manager OpsCenter

OPS10-BP02 Menjalankan proses untuk setiap peringatan

Tetapkan respons (runbook atau buku pedoman) dengan baik, dengan pemilik yang teridentifikasi secara khusus, untuk peristiwa apa pun yang diatur peringatannya. Ini memastikan respons yang efektif dan cepat terhadap peristiwa operasi dan mencegah peristiwa yang dapat ditindaklanjuti dihalangi oleh notifikasi yang kurang bernilai.

Antipola umum:

- Sistem pemantauan memberikan aliran koneksi yang disetujui bersama dengan pesan lainnya. Volume pesan sangat besar sehingga Anda melewatkan pesan kesalahan berkala yang perlu diintervensi.
- Anda menerima peringatan bahwa situs web terhenti. Tidak ada proses yang ditentukan jika hal seperti ini terjadi. Anda dipaksa untuk melakukan tindakan ad hoc untuk mendiagnosis dan menyelesaikan masalah. Mengembangkan proses ini seiring berjalannya waktu akan memperpanjang waktu pemulihan.

Manfaat menerapkan praktik terbaik ini: Dengan memperingatkan hanya ketika tindakan diperlukan, Anda mencegah peringatan bernilai rendah menutupi peringatan bernilai tinggi. Dengan memiliki proses untuk setiap peringatan yang dapat ditindaklanjuti, Anda mengaktifkan respons yang konsisten dan cepat terhadap peristiwa di lingkungan Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Proses per peringatan: Peristiwa apa pun yang Anda aktifkan peringatannya harus memiliki respons (runbook atau buku pedoman) yang jelas dengan pemilik yang teridentifikasi secara khusus (misalnya, individu, tim, atau peran) yang bertanggung jawab atas penyelesaian yang berhasil. Kinerja respons dapat diotomatiskan atau dilakukan oleh tim lain tetapi pemiliknya bertanggung jawab untuk memastikan proses memberikan hasil yang diharapkan. Dengan memiliki proses ini, Anda memastikan respons yang efektif dan cepat terhadap peristiwa operasi dan mencegah peristiwa yang dapat ditindaklanjuti dihalangi oleh notifikasi yang kurang bernilai. Misalnya, penskalaan otomatis dapat diterapkan untuk menskalakan front end web, tetapi tim operasi mungkin bertanggung jawab untuk memastikan bahwa aturan dan batas penskalaan otomatis sesuai untuk kebutuhan beban kerja.

Sumber daya

Dokumen terkait:

- Fitur Amazon CloudWatch
- Apa itu Amazon CloudWatch Events?

Video terkait:

- Build a Monitoring Plan

OPS10-BP03 Memprioritaskan kejadian operasional berdasarkan dampaknya terhadap bisnis

Ketika ada beberapa kejadian yang memerlukan intervensi, pastikan untuk mengatasi kejadian yang paling signifikan terhadap bisnis terlebih dahulu. Dampak dapat termasuk kematian atau cedera fisik, kerugian finansial, atau rusaknya reputasi dan kepercayaan.

Antipola umum:

- Anda menerima permintaan dukungan untuk menambahkan konfigurasi printer bagi pengguna. Saat sedang menangani masalah tersebut, Anda menerima permintaan dukungan yang menyatakan bahwa situs retail terhenti. Setelah menyelesaikan konfigurasi pencetak untuk pengguna, Anda mulai menangani masalah yang dialami situs web.
- Anda menerima pemberitahuan bahwa sistem pembayaran dan situs web retail Anda terhenti. Anda tidak tahu mana masalah yang harus diprioritaskan.

Manfaat menerapkan praktik terbaik ini: Dengan memprioritaskan insiden yang dampaknya paling besar terhadap bisnis, Anda dapat menetapkan manajemen untuk dampak tersebut.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Prioritaskan peristiwa operasional berdasarkan dampaknya terhadap bisnis: Ketika ada beberapa kejadian yang memerlukan intervensi, atasi kejadian yang paling signifikan terhadap bisnis terlebih dahulu. Dampak dapat termasuk kematian atau cedera fisik, kerugian finansial, atau rusaknya reputasi atau kepercayaan.

OPS10-BP04 Tetapkan jalur eskalasi

Tetapkan jalur eskalasi di runbook dan playbook Anda, termasuk apa yang memicu eskalasi, dan prosedur untuk eskalasi. Secara spesifik identifikasi pemilik untuk setiap tindakan guna memastikan respons yang efektif dan tepat waktu terhadap peristiwa operasi.

Identifikasi ketika keputusan manusia diperlukan sebelum tindakan diambil. Bekerja samalah dengan pengambil keputusan untuk mengambil keputusan tersebut lebih awal, dan untuk mendapatkan terlebih dulu persetujuan atas tindakan, sehingga MTTR tidak menjadi lebih lama karena menunggu respons.

Antipola umum:

- Situs retail Anda tidak berfungsi. Anda tidak memahami runbook untuk memulihkan situs itu. Anda mulai menelepon kolega dengan harapan seseorang akan dapat membantu Anda.
- Anda menerima kasus permintaan dukungan untuk aplikasi yang tidak dapat dijangkau. Anda tidak memiliki izin untuk administrasi sistem. Anda tidak tahu siapa yang memilikinya. Anda berusaha menghubungi pemilik sistem yang membuka kasus tersebut dan tidak mendapatkan respons. Anda tidak memiliki kontak untuk sistem dan kolega Anda tidak tahu.

Manfaat menerapkan praktik terbaik ini: Dengan menetapkan eskalasi, pemicu untuk eskalasi, dan prosedur untuk eskalasi, Anda memungkinkan penambahan sumber daya secara sistematis ke insiden dengan tingkat yang sesuai untuk dampaknya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Tetapkan jalur eskalasi: Tetapkan jalur eskalasi di runbook dan playbook Anda, termasuk apa yang memicu eskalasi, dan prosedur untuk eskalasi. Contohnya, eskalasi masalah dari rekayasawan dukungan ke rekayasawan dukungan senior ketika runbook tidak dapat menyelesaikan masalah, atau ketika jangka waktu yang ditetapkan sebelumnya telah lewat. Contoh lain dari jalur eskalasi yang benar adalah dari rekayasawan dukungan senior ke tim pengembangan untuk beban kerja ketika playbook tidak dapat mengidentifikasi jalur ke perbaikan, atau ketika jangka waktu yang ditetapkan sebelumnya telah lewat. Secara spesifik identifikasi pemilik untuk setiap tindakan guna memastikan respons yang efektif dan tepat waktu terhadap peristiwa operasi. Eskalasi dapat mencakup pihak ketiga. Contohnya, penyedia konektivitas jaringan atau vendor perangkat lunak. Eskalasi dapat mencakup pengambil keputusan resmi yang diidentifikasi untuk sistem yang terkena dampak.

OPS10-BP05 Membuat rencana komunikasi pelanggan untuk gangguan

Buat dan uji rencana komunikasi tentang gangguan sistem yang dapat Anda andalkan agar pelanggan dan pemangku kepentingan Anda selalu mendapatkan informasi selama terjadi gangguan. Komunikasikan langsung ke pengguna Anda baik ketika layanan yang mereka gunakan terkena dampaknya, dan ketika layanan kembali normal.

Hasil yang diinginkan:

- Anda memiliki rencana komunikasi untuk situasi yang berbeda-beda, dari pemeliharaan terjadwal hingga kegagalan besar tak terduga, termasuk pemberlakuan rencana pemulihan bencana.
- Dalam komunikasi Anda, Anda memberikan informasi yang jelas dan transparan tentang masalah sistem untuk membantu pelanggan menghindari meragukan performa sistem mereka.
- Anda menggunakan halaman status dan pesan kesalahan kustom untuk mengurangi lonjakan permintaan di pusat bantuan dan menjaga agar pengguna tetap mendapatkan informasi.
- Rencana komunikasi diuji secara teratur untuk memverifikasi bahwa rencana tersebut memiliki performa sesuai yang dimaksud ketika gangguan yang sesungguhnya terjadi.

Antipola umum:

- Gangguan beban kerja terjadi tetapi Anda tidak memiliki rencana komunikasi. Pengguna membuat sistem pengajuan masalah Anda kewalahan karena terlalu banyak permintaan akibat tidak adanya informasi tentang gangguan.
- Anda mengirimkan pemberitahuan lewat email kepada pengguna selama gangguan berlangsung. Pemberitahuan tersebut tidak berisi jangka waktu untuk pemulihan layanan sehingga pengguna tidak dapat membuat rencana untuk mengatasi gangguan.
- Terdapat rencana komunikasi untuk gangguan tetapi tidak pernah diuji. Gangguan terjadi dan rencana komunikasi gagal karena langkah yang sangat penting terlewatkan, dan hal tersebut seharusnya dapat diketahui dalam pengujian.
- Selama gangguan, Anda mengirimkan kepada pengguna pemberitahuan yang disertai terlalu banyak informasi teknis mendetail dan informasi dalam NDA AWS Anda.

Manfaat menjalankan praktik terbaik ini:

- Mempertahankan komunikasi selama gangguan memastikan pelanggan diberi visibilitas tentang progres masalah dan perkiraan waktu resolusinya.

- Mengembangkan rencana komunikasi yang jelas akan memverifikasi bahwa pelanggan dan pengguna akhir Anda mendapatkan informasi sehingga mereka dapat mengambil langkah tambahan yang diperlukan untuk memitigasi dampak gangguan.
- Dengan komunikasi yang tepat dan peningkatan kesadaran akan gangguan terencana dan tidak terencana, Anda dapat meningkatkan tingkat kepuasan pelanggan, membatasi reaksi yang tidak diinginkan, dan mendorong retensi pelanggan.
- Komunikasi gangguan secara tepat waktu dan transparan meningkatkan keyakinan dan menjalin kepercayaan yang diperlukan untuk memelihara hubungan antara Anda dan pelanggan.
- Strategi komunikasi yang terbukti selama gangguan atau krisis akan mengurangi spekulasi dan gosip yang dapat menghalangi kemampuan Anda untuk pulih.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Rencana komunikasi yang selalu memberikan informasi kepada pelanggan Anda selama gangguan bersifat holistik dan meliputi beberapa antarmuka, termasuk halaman kesalahan yang dilihat pelanggan, pesan kesalahan API kustom, spanduk status sistem, dan halaman status kondisi. Jika sistem Anda mencakup pengguna terdaftar, Anda dapat berkomunikasi melalui saluran pesan seperti email, SMS, atau notifikasi push untuk mengirimkan konten pesan yang dipersonalisasi ke pelanggan Anda.

Alat komunikasi pelanggan

Sebagai garis pertahanan pertama, aplikasi seluler dan web harus memberikan pesan kesalahan yang ramah dan informatif selama gangguan serta memiliki kemampuan untuk mengarahkan ulang lalu lintas ke halaman status. Amazon CloudFront adalah jaringan pengiriman konten (CDN) terkelola penuh yang disertai kemampuan untuk membuat dan menghadirkan konten kesalahan kustom. Halaman kesalahan kustom di CloudFront merupakan lapisan pertama yang bagus untuk pesan pelanggan terkait gangguan di tingkat komponen. CloudFront dapat juga menyederhanakan pengelolaan dan pengaktifan halaman status untuk menangkap semua permintaan selama gangguan terencana atau tidak terencana.

Pesan kesalahan API kustom dapat membantu mendeteksi dan mengurangi dampak ketika gangguan terisolasi ke layanan terpisah. Amazon API Gateway memungkinkan Anda mengonfigurasi respons kustom untuk API REST Anda. Hal ini memungkinkan Anda memberikan pesan yang jelas dan bermakna kepada konsumen API ketika API Gateway tidak dapat menjangkau layanan backend. Pesan kustom juga dapat digunakan untuk mendukung konten spanduk gangguan dan

pemberitahuan ketika fitur sistem tertentu mengalami penurunan kualitas akibat gangguan di tingkat layanan.

Pesan langsung adalah pesan pelanggan jenis paling personal. Amazon Pinpoint adalah layanan terkelola untuk komunikasi multi-saluran yang dapat diskalakan. Amazon Pinpoint memungkinkan Anda membangun kampanye yang dapat menyiarkan pesan secara luas ke seluruh pelanggan yang terkena dampak melalui SMS, email, pesan suara, notifikasi push, atau saluran kustom yang Anda tentukan. Ketika Anda mengelola pesan dengan Amazon Pinpoint, kampanye pesan dibuat dengan baik, dapat diuji, dan dapat diterapkan secara cerdas pada segmen pelanggan yang ditarget. Setelah dibuat, kampanye dapat dijadwalkan atau dipicu oleh peristiwa dan kampanya dapat diuji dengan mudah.

Contoh pelanggan

Ketika beban kerja terganggu, AnyCompany Retail mengirimkan pemberitahuan lewat email ke pengguna mereka. Email tersebut menerangkan fungsionalitas bisnis apa yang terganggu dan memberikan perkiraan yang realistis tentang kapan layanan akan pulih. Selain itu, mereka memiliki halaman status yang menunjukkan informasi dalam waktu nyata tentang kondisi beban kerja mereka. Rencana komunikasi diuji dalam lingkungan pengembangan dua kali per tahun untuk memvalidasi keefektifannya.

Langkah implementasi

1. Tentukan saluran komunikasi untuk strategi pesan Anda. Pertimbangkan aspek arsitektur dari aplikasi Anda dan tentukan strategi terbaik untuk memberikan umpan balik kepada pelanggan. Hal ini dapat mencakup satu atau lebih strategi panduan yang dijelaskan, termasuk halaman status dan kesalahan, respons kesalahan API kustom, atau pesan langsung.
2. Desain halaman status untuk aplikasi Anda. Jika Anda telah menentukan bahwa halaman kesalahan kustom atau status sesuai untuk pelanggan Anda, Anda harus mendesain konten dan pesan Anda untuk halaman tersebut. Halaman kesalahan menjelaskan kepada pengguna mengapa aplikasi tidak tersedia, kapan aplikasi mungkin tersedia lagi, dan apa yang dapat mereka lakukan sementara ini. Jika aplikasi Anda menggunakan Amazon CloudFront Anda dapat menampilkan respons kesalahan kustom atau menggunakan Lambda di Edge untuk menerjemahkan kesalahan dan menulis ulang konten halaman. CloudFront juga memungkinkan penukaran destinasi dari konten aplikasi Anda ke asal konten Amazon S3 statis yang berisi halaman status gangguan atau pemeliharaan Anda.
3. Desain status kesalahan API yang benar untuk layanan Anda. Pesan kesalahan yang dihasilkan oleh API Gateway ketika layanan backend tidak dapat dicapainya, serta pengecualian tingkat

layanan, mungkin tidak berisi pesan yang ramah dan sesuai untuk ditampilkan ke pengguna akhir. Tanpa harus membuat perubahan kode pada layanan backend Anda, Anda dapat mengonfigurasi API Gateway respons kesalahan kustom untuk memetakan kode respons HTTP ke pesan kesalahan API yang dikurasi.

4. Desain pesan dari perspektif bisnis sehingga pesan relevan untuk pengguna akhir sistem Anda dan tidak berisi informasi teknis mendetail. Pertimbangkan audiensi Anda dan sesuaikan pesan Anda. Contohnya, Anda mungkin mengarahkan pengguna internal ke proses manual atau solusi alternatif yang memanfaatkan sistem pengganti. Pengguna eksternal dapat diminta untuk menunggu sampai sistem pulih, atau berlangganan pengiriman pembaruan informasi untuk menerima pemberitahuan segera setelah sistem pulih. Tentukan pesan yang disetujui untuk beberapa skenario, termasuk gangguan tak terduga, pemeliharaan terencana, dan kegagalan sistem parsial di mana fitur tertentu mungkin mengalami penurunan kualitas atau tidak tersedia.
5. Buat sebagai templat dan otomatiskan pesan Anda untuk pelanggan. Setelah Anda membuat konten pesan, Anda dapat menggunakan Amazon Pinpoint atau alat lain untuk mengotomatiskan kampanye pesan Anda. Dengan Amazon Pinpoint Anda dapat membuat segmen pelanggan target untuk pengguna spesifik yang terpengaruh dan mengubah pesan menjadi templat. Tinjau tutorial Amazon Pinpoint untuk memahami cara membuat kampanye pesan.
6. Hindari kemampuan pesan dengan penggabungan erat di sistem yang dilihat pelanggan Anda. Strategi pesan Anda tidak boleh memiliki dependensi keras pada layanan atau penyimpanan data sistem untuk memverifikasi bahwa Anda bisa sukses mengirimkan pesan ketika Anda mengalami gangguan. Pertimbangkan untuk membuat kemampuan mengirimkan pesan dari lebih dari satu Zona Ketersediaan atau Wilayah untuk ketersediaan pesan. Jika Anda menggunakan layanan AWS untuk mengirimkan pesan, manfaatkan operasi bidang data daripada operasi bidang kendali untuk memunculkan pesan Anda.

Tingkat upaya untuk rencana implementasi: Tinggi. Mengembangkan rencana komunikasi, dan mekanisme untuk mengirimkannya, dapat memerlukan upaya yang cukup besar.

Sumber daya

Praktik terbaik terkait:

- OPS07-BP03 Menggunakan runbook untuk menjalankan prosedur - Rencana komunikasi Anda harus memiliki runbook yang terkait dengannya sehingga personel Anda tahu cara merespons.
- OPS11-BP02 Menjalankan analisis setelah insiden - Setelah gangguan, lakukan analisis pasca-insiden guna mengidentifikasi mekanisme untuk mencegah gangguan lain.

Dokumen terkait:

- Pola Penanganan Kesalahan di Amazon API Gateway dan AWS Lambda
- Respons Amazon API Gateway

Contoh terkait:

- Dasbor AWS Health
- Ringkasan Peristiwa Layanan AWS di Wilayah Virginia Utara (US-EAST-1)

Layanan terkait:

- AWS Dukungan
- Perjanjian Pelanggan AWS
- Amazon CloudFront
- Amazon API Gateway
- Amazon Pinpoint
- Amazon S3

OPS10-BP06 Mengomunikasikan status melalui dasbor

Menyediakan dasbor yang disesuaikan untuk audiens target mereka (misalnya, tim teknis internal, pimpinan, dan pelanggan) guna mengomunikasikan status operasi bisnis saat ini dan memberikan metrik kepentingan.

Anda dapat membuat dasbor menggunakan Dasbor Amazon CloudWatch dengan halaman beranda yang dapat disesuaikan di konsol CloudWatch. Dengan layanan kecerdasan bisnis seperti Quick Anda dapat membuat dan memublikasikan dasbor interaktif yang menampilkan kondisi operasional dan beban kerja Anda (misalnya, tingkat pesanan, pengguna terhubung, dan waktu transaksi). Buat Dasbor yang memberikan tampilan tingkat bisnis dan sistem mengenai metrik Anda.

Antipola umum:

- Atas permintaan, Anda menjalankan laporan tentang pemanfaatan aplikasi Anda saat ini untuk manajemen.
- Selama insiden, Anda dihubungi setiap dua puluh menit oleh pemilik sistem yang ingin mengetahui apakah insiden sudah teratasi.

Manfaat menerapkan praktik terbaik ini: Dengan membuat dasbor, Anda mengaktifkan akses layanan mandiri untuk pelanggan Anda agar mereka mengetahui jika mereka harus melakukan suatu tindakan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Komunikasikan status melalui dasbor: Sediakan dasbor yang disesuaikan untuk audiens target mereka (misalnya, tim teknis internal, pimpinan, dan pelanggan) guna mengomunikasikan status operasi bisnis saat ini dan menyediakan metrik kepentingan. Menyediakan opsi layanan mandiri untuk informasi status dapat mengurangi disrupsi permintaan penanganan status dari tim operator lapangan. Contohnya termasuk dasbor Amazon CloudWatch dan Dasbor AWS Health.
  - Dasbor CloudWatch membuat dan menggunakan tampilan metrik yang disesuaikan

Sumber daya

Dokumen terkait:

- Quick
- Dasbor CloudWatch membuat dan menggunakan tampilan metrik yang disesuaikan

OPS10-BP07 Otomatiskan respons terhadap peristiwa

Otomatiskan respons terhadap peristiwa untuk mengurangi kesalahan yang disebabkan oleh proses manual, dan untuk memastikan respons yang konsisten dan tepat waktu.

Ada sejumlah cara untuk mengotomatiskan tindakan runbook dan playbook di AWS. Untuk merespons peristiwa dari perubahan keadaan di sumber daya AWS Anda, atau dari peristiwa kustom Anda sendiri, Anda harus membuat aturan CloudWatch Events untuk memicu respons melalui target CloudWatch (contohnya, fungsi Lambda, topik Amazon Simple Notification Service (Amazon SNS), tugas Amazon ECS, dan Otomatisasi AWS Systems Manager).

Untuk merespons metrik yang melampaui ambang batas untuk sumber daya (contohnya, waktu tunggu), Anda harus membuat alarm CloudWatch untuk melakukan satu atau lebih tindakan menggunakan tindakan CloudWatch Events, tindakan Auto Scaling, atau untuk mengirimkan notifikasi ke topik Amazon SNS. Jika Anda harus melakukan tindakan kustom untuk merespons alarm, panggil Lambda melalui notifikasi Amazon SNS. Gunakan Amazon SNS untuk mempublikasikan notifikasi peristiwa dan pesan eskalasi agar orang selalu tahu.

AWS juga mendukung sistem pihak ketiga melalui API dan SDK layanan AWS. Ada sejumlah alat pemantauan yang disediakan oleh Partner AWS dan pihak ketiga yang memungkinkan pemantauan, notifikasi, dan respons. Beberapa alat ini antara lain New Relic, Splunk, Loggly, SumoLogic, dan Datadog.

Anda harus selalu menyediakan prosedur manual yang sangat penting untuk digunakan ketika prosedur otomatis gagal

Antipola umum:

- Developer memeriksa kodenya. Peristiwa ini bisa saja digunakan untuk mulai membangun kemudian melakukan pengujian tetapi tidak ada yang terjadi.
- Aplikasi Anda mencatat kesalahan spesifik sebelum berhenti berfungsi. Prosedur untuk memulai ulang aplikasi dipahami dengan baik dan dapat diberi skrip. Anda dapat menggunakan log event untuk memanggil skrip dan memulai ulang aplikasi. Tetapi, ketika kesalahan terjadi pada hari Minggu jam 3 pagi, Anda dibangunkan karena Anda adalah sumber daya yang siap dipanggil untuk memperbaiki sistem tersebut.

Manfaat menerapkan praktik terbaik ini: Dengan menggunakan respons otomatis terhadap peristiwa, Anda mengurangi waktu untuk merespons dan membatasi timbulnya kesalahan akibat aktivitas manual.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Rendah

Panduan implementasi

- Otomatiskan respons terhadap peristiwa: Otomatiskan respons terhadap peristiwa untuk mengurangi kesalahan yang disebabkan oleh proses manual, dan untuk memastikan respons yang konsisten dan tepat waktu.
  - Apa itu Amazon CloudWatch Events?
  - Membuat aturan CloudWatch Events yang memicu peristiwa
  - Membuat aturan CloudWatch Events yang memicu AWS panggilan API menggunakan AWS CloudTrail
  - Contoh peristiwa CloudWatch Events dari layanan yang didukung

Sumber daya

Dokumen terkait:

- Amazon CloudWatch Fitur
- Contoh peristiwa CloudWatch Events dari layanan yang didukung
- Membuat aturan CloudWatch Events yang memicu AWS panggilan API menggunakan AWS CloudTrail
- Membuat aturan CloudWatch Events yang memicu peristiwa
- Apa itu Amazon CloudWatch Events?

Video terkait:

- Buat Rencana Pemantauan

Contoh terkait:

Kembangkan

Pertanyaan

- OPS 11. Bagaimana cara mengembangkan operasi?

OPS 11. Bagaimana cara mengembangkan operasi?

Luangkan waktu dan sumber daya khusus untuk peningkatan bertahap yang hampir berkelanjutan untuk meningkatkan dan efisiensi operasi Anda.

Praktik terbaik

- OPS11-BP01 Miliki proses untuk peningkatan berkelanjutan
- OPS11-BP02 Menjalankan analisis setelah insiden
- OPS11-BP03 Mengimplementasikan loop umpan balik
- OPS11-BP04 Menjalankan manajemen pengetahuan
- OPS11-BP05 Menetapkan pendorong untuk perbaikan
- OPS11-BP06 Memvalidasi wawasan
- OPS11-BP07 Melakukan peninjauan metrik operasi
- OPS11-BP08 Mendokumentasikan dan membagikan pelajaran yang didapatkan
- OPS11-BP09 Mengalokasikan waktu untuk membuat peningkatan

OPS11-BP01 Miliki proses untuk peningkatan berkelanjutan

Evaluasi beban kerja Anda berdasarkan praktik terbaik arsitektur internal dan eksternal Lakukan peninjauan beban kerja setidaknya satu kali setahun. Prioritaskan peluang perbaikan ke dalam jadwal pengembangan perangkat lunak Anda.

Hasil yang diinginkan:

- Anda menganalisis beban kerja berdasarkan praktik terbaik arsitektur setidaknya satu kali setahun.
- Peluang perbaikan memperoleh prioritas yang setara dalam proses pengembangan perangkat lunak Anda.

Antipola umum:

- Anda belum menjalankan peninjauan arsitektur pada beban kerja Anda sejak deployment beberapa tahun lalu.
- Peluang perbaikan menerima prioritas yang lebih rendah dan tetap berada di backlog.
- Tidak ada standar untuk mengimplementasikan modifikasi terhadap praktik terbaik untuk organisasi.

Manfaat menjalankan praktik terbaik ini:

- Beban kerja Anda selalu dimutakhirkan dengan praktik terbaik arsitektur.
- Mengembangkan beban kerja dilakukan secara cermat.
- Anda dapat memanfaatkan praktik terbaik organisasi untuk meningkatkan semua beban kerja.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Minimal satu kali dalam setahun, Anda melakukan peninjauan arsitektur beban kerja. Menggunakan praktik terbaik internal dan eksternal, evaluasi beban kerja Anda dan identifikasi peluang perbaikan. Prioritaskan peluang perbaikan ke dalam jadwal pengembangan perangkat lunak Anda.

Contoh pelanggan

Semua beban kerja di AnyCompany Retail menjalani proses peninjauan arsitektur tahunan. Mereka mengembangkan daftar periksa praktik terbaik mereka sendiri yang berlaku untuk semua beban

kerja. Menggunakan fitur Lensa Kustom AWS Well-Architected Tool, mereka melakukan peninjauan menggunakan alat dan lensa praktik terbaik kustom mereka. Peluang perbaikan yang dihasilkan dari peninjauan diberikan prioritas dalam sprint perangkat lunak mereka.

Langkah implementasi

1. Lakukan peninjauan arsitektur berkala pada beban kerja produksi Anda setidaknya satu kali dalam setahun. Gunakan standar arsitektur terdokumentasi yang menyertakan praktik terbaik khusus AWS.
   a. Kami menyarankan Anda menggunakan standar yang ditetapkan secara internal untuk peninjauan ini. Jika Anda tidak memiliki standar internal, kami menyarankan Anda menggunakan Kerangka Kerja AWS Well-Architected.
   b. Anda dapat menggunakan AWS Well-Architected Tool untuk membuat Lensa Kustom praktik terbaik internal Anda dan melakukan peninjauan arsitektur Anda.
   c. Pelanggan dapat menghubungi Arsitek Solusi AWS mereka untuk melakukan Peninjauan Kerangka Kerja Well-Architected terpandu pada beban kerja mereka.
2. Prioritaskan peluang perbaikan yang diidentifikasi selama peninjauan ke dalam proses pengembangan perangkat lunak Anda.

Tingkat upaya untuk rencana implementasi: Rendah Anda dapat menggunakan Kerangka Kerja AWS Well-Architected untuk melakukan peninjauan arsitektur tahunan Anda.

Sumber daya

Praktik Terbaik Terkait:

- OPS11-BP02 Menjalankan analisis setelah insiden - Analisis pascainsiden adalah penghasil item perbaikan lainnya. Masukkan pelajaran yang diperoleh ke dalam daftar praktik terbaik arsitektur internal Anda.
- OPS11-BP08 Mendokumentasikan dan membagikan pelajaran yang didapatkan - Bagikan praktik terbaik arsitektur internal yang sedang Anda kembangkan kepada seluruh organisasi Anda.

Dokumen terkait:

- AWS Well-Architected Tool - Lensa kustom
- Laporan Resmi AWS Well-Architected - Proses peninjauan

- Kustomisasi Peninjauan Well-Architected menggunakan Lensa Kustom dan AWS Well-Architected Tool
- Mengimplementasikan siklus hidup Lensa Kustom AWS Well-Architected di dalam organisasi Anda

Video terkait:

- Lab Well-Architected - Level 100: Lensa Kustom di AWS Well-Architected Tool

Contoh terkait:

- AWS Well-Architected Tool

OPS11-BP02 Menjalankan analisis setelah insiden

Tinjau peristiwa yang memengaruhi pelanggan, dan identifikasi faktor yang berkontribusi serta tindakan pencegahannya. Gunakan informasi ini untuk mengembangkan mitigasi guna meminimalkan atau mencegah kemungkinan terjadi lagi. Kembangkan prosedur untuk respons efektif dan cepat. Komunikasikan faktor yang berkontribusi dan tindakan korektif yang diperlukan, yang disesuaikan dengan audiens target.

Antipola umum:

- Anda mengelola server aplikasi. Kira-kira setiap 23 jam 55 menit, semua sesi aktif Anda dihapus. Anda berupaya mengidentifikasi masalah yang terjadi di server aplikasi Anda. Anda menduga bahwa ini mungkin masalah jaringan, tetapi tidak dapat memperoleh bantuan dari tim jaringan karena mereka terlalu sibuk. Anda tidak menetapkan proses di awal yang dapat Anda jadikan panduan untuk mendapatkan dukungan dan mengumpulkan informasi yang dibutuhkan guna mengetahui masalah yang sedang terjadi.
- Anda mengalami kehilangan data di dalam beban kerja Anda. Hal ini baru pertama kali terjadi dan penyebabnya belum jelas. Anda menganggap bahwa kejadian ini tidak penting karena Anda dapat membuat ulang data. Kehilangan data makin sering terjadi dan memengaruhi pelanggan Anda. Hal ini juga menambah beban operasional Anda karena harus memulihkan data yang hilang.

Manfaat menerapkan praktik terbaik ini: Dengan proses yang telah ditetapkan di awal untuk menentukan komponen, kondisi, tindakan, dan kejadian yang berkontribusi terhadap insiden, Anda dapat mengidentifikasi peluang untuk pengembangan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Gunakan sebuah proses untuk mengetahui faktor yang berkontribusi: Tinjau semua insiden yang memengaruhi pelanggan. Buat sebuah proses untuk mengidentifikasi dan mendokumentasi faktor yang berkontribusi terhadap insiden agar Anda dapat mengembangkan mitigasi untuk membatasi atau mencegah kejadian serupa serta mengembangkan prosedur untuk merespons dengan cepat dan efektif. Komunikasikan penyebab utama sebagaimana diperlukan, yang disesuaikan dengan audiens target.

OPS11-BP03 Mengimplementasikan loop umpan balik

Loop umpan balik menyediakan wawasan yang dapat ditindaklanjuti yang mendorong pengambilan keputusan. Masukkan loop umpan balik ke dalam prosedur dan beban kerja Anda. Ini membantu Anda mengidentifikasi permasalahan dan area yang memerlukan perbaikan. Loop umpan balik juga memvalidasi investasi yang dilakukan dalam upaya perbaikan. Loop umpan balik ini adalah landasan untuk meningkatkan beban kerja Anda secara berkelanjutan.

Loop umpan balik dibagi ke dalam dua kategori: umpan balik langsung dan analisis retrospektif. Umpan balik langsung (immediate feedback) dikumpulkan melalui peninjauan kinerja dan hasil dari aktivitas operasi. Umpan balik ini berasal dari anggota tim, pelanggan, atau output otomatis dari aktivitas. Umpan balik langsung diterima dari hal-hal seperti pengujian A/B dan pengiriman fitur baru, dan ini penting untuk gagal cepat (fail fast).

Analisis retrospektif dilakukan secara rutin untuk menangkap umpan balik dari peninjauan metrik dan hasil operasional dari waktu ke waktu. Retrospektif ini terjadi pada akhir sprint, secara terjadwal, atau setelah perilisan atau peristiwa besar. Tipe loop umpan balik ini memvalidasi investasi dalam operasi atau beban kerja Anda. Loop umpan balik ini membantu Anda mengukur keberhasilan dan memvalidasi strategi Anda.

Hasil yang diinginkan: Anda menggunakan umpan balik langsung dan analisis retrospektif untuk mendorong perbaikan. Terdapat mekanisme untuk mendapatkan umpan balik pengguna dan anggota tim. Analisis retrospektif digunakan untuk mengidentifikasi tren-tren yang mendorong perbaikan.

Antipola umum:

- Anda meluncurkan fitur baru tetapi tidak ada cara untuk menerima umpan balik pelanggan tentangnya.

- Setelah berinvestasi dalam perbaikan operasi, Anda tidak melakukan analisis retrospektif untuk memvalidasinya.
- Anda mengumpulkan umpan balik pelanggan tetapi tidak meninjaunya secara rutin.
- Loop umpan balik mendatangkan item-item tindakan yang diajukan tetapi item-item tersebut tidak disertakan dalam proses pengembangan perangkat lunak.
- Pelanggan tidak menerima umpan balik tentang perbaikan yang mereka ajukan.

Manfaat menjalankan praktik terbaik ini:

- Anda dapat bekerja mundur dari pelanggan untuk mendorong fitur-fitur baru.
- Budaya organisasi Anda dapat merespons perubahan lebih cepat.
- Tren digunakan untuk mengidentifikasi peluang perbaikan.
- Retrospektif memvalidasi investasi yang dilakukan pada beban kerja dan operasi Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Dengan mengimplementasikan praktik terbaik ini, Anda dapat menggunakan umpan balik langsung serta analisis retrospektif. Loop umpan balik ini mendorong perbaikan. Terdapat banyak mekanisme untuk umpan balik langsung, termasuk survei, jajak pendapat pelanggan, atau formulir umpan balik. Organisasi Anda juga menggunakan retrospektif untuk mengidentifikasi peluang perbaikan dan memvalidasi inisiatif.

Contoh pelanggan

AnyCompany Retail membuat sebuah formulir web yang digunakan pelanggan untuk memberikan umpan balik atau melaporkan permasalahan. Selama scrum mingguan, umpan balik pengguna dievaluasi oleh tim pengembangan perangkat lunak. Umpan balik digunakan secara rutin sebagai landasan pengembangan platform mereka. Mereka melakukan analisis retrospektif di akhir setiap sprint untuk mengidentifikasi item yang ingin mereka tingkatkan.

Langkah implementasi

1. Umpan balik langsung
    - Anda memerlukan mekanisme untuk menjangkau umpan balik dari pelanggan dan anggota tim. Aktivitas operasi Anda juga dapat dikonfigurasi untuk menghadirkan umpan balik otomatis.

- Organisasi Anda perlu meninjau umpan balik ini, menentukan hal-hal yang harus ditingkatkan, dan menjadwalkan perbaikan.
- Umpan balik harus ditambahkan ke dalam proses pengembangan perangkat lunak Anda.
- Seiring Anda melakukan perbaikan, sampaikan tindak lanjut kepada pemberi umpan balik.
  - Anda dapat menggunakan AWS Systems Manager OpsCenter untuk membuat dan melacak perbaikan ini dalam bentuk OpsItems.

2. Analisis retrospektif
   - Lakukan retrospektif di akhir siklus pengembangan, pada jadwal yang ditetapkan, atau setelah perilisan besar.
   - Kumpulkan pemangku kepentingan yang terlibat dalam beban kerja untuk rapat retrospektif.
   - Buat tiga kolom di papan tulis atau lembar kerja: Hentikan, Mulai, dan Pertahankan
     - Hentikan adalah untuk apa pun yang Anda ingin tidak dilakukan lagi oleh tim Anda.
     - Mulai adalah gagasan yang ingin mulai Anda lakukan.
     - Pertahankan adalah untuk item-item yang ingin tetap Anda lakukan.
   - Berkelilinglah dan kumpulkan umpan balik dari para pemangku kepentingan.
   - Buat prioritas umpan balik. Tugaskan tindakan dan pemangku kepentingan ke item-item Mulai atau Pertahankan.
   - Tambahkan tindakan ke proses pengembangan perangkat lunak dan komunikasikan pembaruan status ke pemangku kepentingan seiring Anda melakukan perbaikan.

Tingkat upaya untuk rencana implementasi: Sedang. Untuk mengimplementasikan praktik terbaik, Anda memerlukan cara untuk menyerap umpan balik langsung dan menganalisisnya. Selain itu, Anda perlu membangun proses analisis retrospektif.

Sumber daya

Praktik terbaik terkait:

- OPS01-BP01 Mengevaluasi kebutuhan pelanggan eksternal: Loop umpan balik adalah mekanisme untuk mengumpulkan kebutuhan pelanggan eksternal.
- OPS01-BP02 Mengevaluasi kebutuhan pelanggan internal: Pemangku kepentingan internal dapat menggunakan loop umpan balik untuk mengomunikasikan kebutuhan dan persyaratan.
- OPS11-BP02 Menjalankan analisis setelah insiden: Analisis pascainsiden adalah bentuk analisis retrospektif yang penting yang dilakukan setelah insiden.

- OPS11-BP07 Melakukan peninjauan metrik operasi: Peninjauan metrik operasi mengidentifikasi tren dan area perbaikan.

Dokumen terkait:

- 7 Perangkap yang Perlu Dihindari Saat Membangun CCOE
- Playbook Tim Atlassian - Retrospektif
- Definisi Email: Loop Umpan Balik
- Membangun Loop Umpan Balik Berdasarkan Tinjauan AWS Well-Architected Framework
- Metodologi Garasi IBM - Melakukan retrospektif
- Investopedia - Siklus PDCS
- Memaksimalkan Efektivitas Developer oleh Tim Cochran
- Laporan Resmi Peninjauan Kesiapan Operasional (ORR) - Iterasi
- TIL CSI - Continual Service Improvement (Perbaikan Layanan Berkelanjutan)
- Saat Toyota bertemu e-commerce: Bersandar pada Amazon

Video terkait:

- Membangun Loop Umpan Balik Pelanggan yang Efektif

Contoh terkait:

- Astuto - alat umpan balik pelanggan sumber terbuka
- Solusi AWS - QnABot di AWS
- Fider - Platform untuk mengatur umpan balik pelanggan

Layanan terkait:

- AWS Systems Manager OpsCenter

OPS11-BP04 Menjalankan manajemen pengetahuan

Manajemen pengetahuan membantu anggota tim menemukan informasi untuk melakukan pekerjaan mereka. Di dalam organisasi yang mau belajar, informasi dibagikan secara bebas sehingga individu

diberdayakan. Informasi dapat ditemukan atau dicari. Informasi bersifat akurat dan mutakhir. Ada mekanisme untuk membuat informasi baru, memperbarui informasi yang sudah ada, dan mengarsipkan informasi yang kedaluwarsa. Contoh paling umum dari platform manajemen pengetahuan adalah sistem manajemen konten seperti wiki.

Hasil yang diinginkan:

- Anggota tim memiliki akses ke informasi yang akurat secara tepat waktu.
- Informasi dapat dicari.
- Ada mekanisme untuk menambahkan, memperbarui, dan mengarsipkan informasi.

Antipola umum:

- Tidak ada penyimpanan pengetahuan terpusat. Anggota tim mengelola catatan mereka sendiri di mesin mereka secara lokal.
- Anda memiliki wiki yang di-hosting secara mandiri tetapi tidak ada mekanisme untuk mengelola informasi, yang mengakibatkan informasi menjadi kedaluwarsa.
- Seseorang melihat ada informasi yang kurang tetapi tidak ada proses untuk meminta penambahannya ke wiki. Mereka menambahkannya sendiri tetapi mereka melewatkan langkah yang penting, sehingga mengakibatkan terjadinya gangguan.

Manfaat menjalankan praktik terbaik ini:

- Anggota tim diberdayakan karena informasi dibagikan secara bebas.
- Anggota tim baru menjalani masa orientasi dengan lebih cepat karena dokumentasinya mutakhir dan dapat dicari.
- Informasi bersifat tepat waktu, akurat, dan dapat ditindaklanjuti.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Manajemen pengetahuan adalah segi penting dari organisasi yang mau belajar. Untuk memulai, Anda memerlukan tempat penyimpanan terpusat guna menyimpan pengetahuan Anda (contoh yang umum yakni wiki yang di-hosting secara mandiri). Anda harus membuat proses untuk menambahkan, memperbarui, dan mengarsipkan pengetahuan. Buat standar untuk apa yang harus didokumentasikan dan izinkan semua orang memberi kontribusi.

Contoh pelanggan

AnyCompany Retail melakukan hosting Wiki internal tempat semua pengetahuan disimpan. Anggota tim didorong untuk menambahkan pengetahuan seiring pengerjaan tugas mereka sehari-hari. Setiap kuartal sekali, tim lintas fungsi mengevaluasi halaman mana yang paling jarang diperbarui dan menentukan apakah halaman tersebut harus diarsipkan atau diperbarui.

Langkah implementasi

1. Mulai dengan identifikasi sistem manajemen konten tempat pengetahuan akan disimpan. Dapatkan kesepakatan para pemangku kepentingan dari seluruh organisasi Anda.
   a. Jika Anda belum memiliki sistem manajemen konten, pertimbangkan untuk menjalankan wiki yang di-hosting secara mandiri atau menggunakan tempat penyimpanan pengontrolan versi sebagai titik awal.
2. Kembangkan runbook untuk menambahkan, memperbarui, dan mengarsipkan informasi. Didik tim Anda tentang proses-proses ini.
3. Identifikasi pengetahuan apa yang harus disimpan di sistem manajemen konten. Mulai dengan aktivitas harian (runbook dan playbook) yang dilakukan anggota tim. Bekerja samalah dengan para pemangku kepentingan untuk memprioritaskan pengetahuan yang akan ditambahkan.
4. Bekerja samalah dengan para pemangku kepentingan secara berkala untuk mengidentifikasi informasi yang kedaluwarsa dan arsipkan atau mutakhirkan.

Tingkat upaya untuk rencana implementasi: Sedang. Jika Anda belum memiliki sistem manajemen konten, Anda dapat membuat wiki yang di-hosting secara mandiri atau menggunakan tempat penyimpanan dokumen dengan pengontrolan versi.

Sumber daya

Praktik terbaik terkait:

- OPS11-BP08 Mendokumentasikan dan membagikan pelajaran yang didapatkan - Manajemen pengetahuan memfasilitasi pembagian informasi tentang pelajaran yang didapatkan.

Dokumen terkait:

- Atlassian - Manajemen Pengetahuan

Contoh terkait:

- DokuWiki
- Gollum
- MediaWiki
- Wiki.js

OPS11-BP05 Menetapkan pendorong untuk perbaikan

Identifikasi pendorong untuk perbaikan untuk membantu Anda mengevaluasi dan memprioritaskan peluang.

Di AWS, Anda dapat mengagregasi log semua aktivitas operasi, beban kerja, dan infrastruktur Anda untuk membuat riwayat aktivitas yang mendetail. Kemudian, Anda dapat menggunakan alat-alat AWS untuk menganalisis operasi dan kesehatan beban kerja Anda seiring waktu (misalnya untuk mengidentifikasi tren, mengaitkan peristiwa dan aktivitas dengan hasil, dan membandingkan serta mengkontraskan antarlingkungan dan lintas sistem) untuk mengungkap peluang perbaikan berdasarkan pendorong Anda.

Anda harus menggunakan CloudTrail untuk melacak aktivitas API (melalui Konsol Manajemen AWS, CLI, SDK, dan API) untuk mengetahui apa yang terjadi di seluruh akun Anda. Lacak aktivitas deployment Alat pengembang AWS Anda dengan CloudTrail dan CloudWatch. Ini akan menambahkan riwayat aktivitas mendetail untuk deployment Anda serta hasilnya ke data log CloudWatch Logs Anda.

Ekspor data log Anda ke Amazon S3 untuk penyimpanan jangka panjang. Menggunakan AWS Glue, Anda menemukan dan mempersiapkan data log Anda di Amazon S3 untuk analitik. Gunakan Amazon Athena, melalui integrasi native-nya dengan AWS Glue, untuk menganalisis data log Anda. Gunakan alat kecerdasan bisnis seperti Quick untuk memvisualisasi, menjelajahi, dan menganalisis data Anda

Antipola umum:

- Anda memiliki skrip yang berfungsi tetapi tidak elegan. Anda menginvestasikan waktu untuk menulis ulang skrip tersebut. Kini skrip tersebut terlihat sangat bagus.
- Perusahaan rintisan Anda sedang mencoba mendapatkan pendanaan lain dari sebuah pemodal ventura. Mereka meminta Anda mendemonstrasikan kepatuhan terhadap PCI DSS. Anda ingin membuat mereka terkesan sehingga Anda mendokumentasikan kepatuhan, tetapi Anda melewatkan tanggal pengiriman untuk seorang pelanggan dan kehilangan pelanggan tersebut. Ini bukan tindakan yang salah tetapi sekarang Anda bertanya-tanya apakah itu tindakan yang tepat.

Manfaat menjalankan praktik terbaik ini: Dengan menentukan kriteria yang ingin Anda gunakan untuk perbaikan, Anda dapat meminimalkan dampak motivasi berbasis peristiwa atau investasi emosional.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

- Pahami pendorong perbaikan: Anda sebaiknya hanya melakukan perubahan pada suatu sistem saat hasil yang diinginkan didukung.
  - Kemampuan yang diinginkan: Evaluasi fitur dan kemampuan yang diinginkan saat mengevaluasi peluang untuk perbaikan.
    - Yang Baru dengan AWS
  - Masalah yang tidak dapat diterima: Evaluasi masalah, bug, dan kerentanan yang tidak dapat diterima saat mengevaluasi peluang untuk perbaikan.
    - Buletin Keamanan Terkini AWS
    - AWS Trusted Advisor
  - Persyaratan kepatuhan: Evaluasi pembaruan dan perubahan yang diperlukan untuk mempertahankan kepatuhan terhadap peraturan, kebijakan, atau agar tetap memperoleh dukungan pihak ketiga, saat meninjau peluang untuk perbaikan.
    - Kepatuhan AWS
    - Program Kepatuhan AWS
    - Berita Terbaru Kepatuhan AWS

Sumber daya

Dokumen terkait:

- Amazon Athena
- Quick
- Kepatuhan AWS
- Berita Terbaru Kepatuhan AWS
- Program Kepatuhan AWS
- AWS Glue
- Buletin Keamanan Terkini AWS
- AWS Trusted Advisor

- Ekspor data log Anda ke Amazon S3
- Yang Baru dengan AWS

OPS11-BP06 Memvalidasi wawasan

Tinjau respons dan hasil analisis Anda dengan tim lintas fungsi serta pemilik bisnis. Gunakan tinjauan tersebut untuk menetapkan pemahaman umum, mengidentifikasi dampak tambahan, dan menentukan alur tindakan. Sesuaikan respons sebagaimana mestinya.

Antipola umum:

- Anda menemukan pemanfaatan CPU pada sistem sebesar 95% dan menjadikan hal itu sebagai prioritas untuk menemukan cara mengurangi beban pada sistem. Anda menentukan tindakan terbaik yang perlu dinaikkan skalanya. Sistemnya adalah transkoder dan sistem tersebut diskalakan untuk menjalankan 95% pemanfaatan CPU sepanjang waktu. Pemilik sistem dapat menjelaskan situasinya kepada Anda jika Anda menghubunginya. Waktu Anda terbuang.
- Pemilik sistem menyatakan bahwa sistem mereka bersifat kritis terhadap misi. Sistemnya tidak ditempatkan di lingkungan dengan keamanan tinggi. Untuk meningkatkan keamanan, Anda mengimplementasikan kontrol detektif dan preventif yang diperlukan untuk sistem yang kritis terhadap misi. Anda memberi tahu pemilik sistem bahwa pekerjaannya sudah selesai dan dia akan dikenakan biaya untuk sumber daya tambahan. Dalam diskusi setelah pemberitahuan ini, pemilik sistem mengetahui bahwa ada ketentuan formal untuk sistem yang kritis terhadap misi yang tidak dipenuhi oleh sistem ini.

Manfaat menerapkan praktik terbaik ini: Dengan memvalidasi wawasan bersama pemilik bisnis dan orang yang ahli di bidangnya, Anda dapat menetapkan pemahaman umum dan memandu peningkatan dengan lebih efektif.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Validasikan wawasan: Berinteraksi dengan pemilik bisnis dan orang yang ahli di bidangnya untuk memastikan ada pemahaman dan kesepakatan bersama tentang makna data yang dikumpulkan. Identifikasikan masalah tambahan, dampak potensial, dan tentukan alur tindakan.

OPS11-BP07 Melakukan peninjauan metrik operasi

Lakukan analisis retrospektif rutin terhadap metrik operasi dengan peserta lintas tim dari berbagai area bisnis. Gunakan tinjauan ini untuk mengidentifikasi peluang perbaikan, potensi pilihan tindakan, dan untuk membagikan pelajaran yang diperoleh.

Cari peluang perbaikan di semua lingkungan Anda (misalnya pengembangan, pengujian, dan produksi).

Antipola umum:

- Terdapat promosi ritel penting yang terganggu oleh jadwal pemeliharaan Anda. Bisnis tidak tahu bahwa ada jadwal pemeliharaan standar yang dapat ditunda jika terdapat peristiwa lain yang memengaruhi bisnis.
- Anda mengalami pemadaman berkepanjangan karena menggunakan pustaka bermasalah yang biasa digunakan di organisasi Anda. Sejak saat itu Anda beralih ke pustaka yang andal. Tim-tim lain di organisasi Anda tidak tahu bahwa mereka terpapar risiko. Jika Anda rutin melakukan pertemuan dan meninjau insiden ini, mereka akan mengetahui risiko tersebut.
- Kinerja transkoder Anda terus mengalami penurunan secara bertahap dan memengaruhi tim media. Saat ini kondisinya belum parah. Anda tidak akan memiliki kesempatan untuk tahu sampai kondisinya cukup buruk hingga menyebabkan insiden. Seandainya Anda meninjau metrik operasi dengan tim media, akan ada peluang untuk melakukan perubahan pada metrik, mengenali pengalaman mereka, dan mengatasi masalah.
- Anda tidak meninjau kepuasan Anda terhadap SLA pelanggan. Anda memiliki kecenderungan untuk tidak memenuhi SLA pelanggan. Terdapat denda finansial jika Anda tidak memenuhi SLA pelanggan. Jika rutin melakukan pertemuan untuk meninjau metrik untuk SLA ini, Anda akan memiliki kesempatan untuk mengenali dan menangani masalah.

Manfaat menjalankan praktik terbaik ini: Dengan melakukan pertemuan rutin untuk meninjau metrik operasi, peristiwa, dan insiden, Anda dapat menjaga pemahaman bersama lintas tim, membagikan pelajaran yang didapatkan, dan dapat memprioritaskan serta menargetkan perbaikan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak ditetapkan: Sedang

Panduan implementasi

- Tinjauan metrik operasi: Lakukan analisis retrospektif rutin terhadap metrik operasi dengan peserta lintas tim dari berbagai area bisnis. Libatkan pemangku kepentingan, termasuk tim bisnis,

pengembangan, dan operasi, untuk memvalidasi temuan dari umpan balik langsung dan analisis retrospektif, serta untuk membagikan pelajaran yang didapatkan. Gunakan wawasan mereka untuk mengidentifikasi peluang perbaikan dan potensi pilihan tindakan.

- Amazon CloudWatch
- Menggunakan metrik Amazon CloudWatch
- Memublikasikan metrik kustom
- Metrik Amazon CloudWatch dan referensi dimensi

Sumber daya

Dokumen terkait:

- Amazon CloudWatch
- Metrik Amazon CloudWatch dan referensi dimensi
- Memublikasikan metrik kustom
- Menggunakan metrik Amazon CloudWatch

OPS11-BP08 Mendokumentasikan dan membagikan pelajaran yang didapatkan

Dokumentasikan dan bagikan pelajaran yang didapatkan dari aktivitas operasional sehingga Anda dapat menggunakannya secara internal dan di seluruh tim.

Anda harus membagikan pelajaran yang didapatkan oleh tim Anda guna meningkatkan manfaat di seluruh organisasi Anda. Anda perlu membagikan informasi dan sumber daya untuk mencegah kesalahan yang dapat dihindari dan memudahkan upaya pengembangan. Dengan demikian, Anda dapat fokus menghadirkan fitur-fitur yang diinginkan.

Gunakan AWS Identity and Access Management (IAM) untuk menetapkan izin yang memungkinkan akses terkontrol ke sumber daya yang ingin Anda bagikan di dalam dan antarakun. Anda harus menggunakan repositori AWS CodeCommit terkontrol versi untuk membagikan pustaka aplikasi, prosedur dalam skrip, dokumentasi prosedur, dan dokumentasi sistem lainnya. Bagikan standar komputasi Anda dengan membagikan akses ke AMI Anda dan dengan memberikan otorisasi penggunaan fungsi Lambda Anda di seluruh akun. Anda juga harus membagikan standar infrastruktur Anda dalam bentuk templat AWS CloudFormation.

Melalui API dan SDK AWS, Anda dapat mengintegrasikan alat dan repositori eksternal dan pihak ketiga (seperti GitHub, BitBucket, dan SourceForge). Ketika membagikan hal-hal yang Anda pelajari

dan kembangkan, berhati-hatilah untuk menyusun izin guna memastikan integritas repositori yang dibagikan.

Antipola umum:

- Anda mengalami pemadaman berkepanjangan karena Anda menggunakan pustaka bermasalah yang biasa digunakan di organisasi Anda. Sejak saat itu Anda beralih ke pustaka yang andal. Tim-tim lain di organisasi Anda tidak tahu bahwa mereka terpapar risiko. Andai saja Anda mendokumentasikan dan membagikan pengalaman Anda dengan pustaka ini, mereka pasti tahu tentang risiko tersebut.
- Anda mengidentifikasi sebuah masalah di dalam layanan mikro yang digunakan bersama secara internal yang menyebabkan terganggunya sesi. Anda pun memperbarui panggilan Anda ke layanan guna menghindari masalah tersebut. Tim-tim lain di organisasi Anda tidak tahu bahwa mereka terpapar risiko. Andai saja Anda mendokumentasikan dan membagikan pengalaman Anda dengan pustaka ini, mereka pasti tahu tentang risiko tersebut.
- Anda menemukan cara untuk mengurangi secara signifikan persyaratan pemanfaatan CPU untuk salah satu layanan mikro Anda. Anda tidak tahu bahwa tim lain bisa memanfaatkan teknik ini. Andai saja Anda mendokumentasikan dan membagikan pengalaman Anda dengan pustaka ini, mereka pasti memiliki peluang untuk melakukannya.

Manfaat menjalankan praktik terbaik ini: Bagikan pelajaran yang didapatkan untuk mendukung perbaikan dan memaksimalkan manfaat pengalaman.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

- Dokumentasikan dan bagikan pelajaran yang didapatkan: Miliki prosedur untuk mendokumentasikan pelajaran didapatkan dari aktivitas operasional dan analisis retrospektif agar dapat digunakan oleh tim lain.
  - Bagikan pembelajaran: Miliki prosedur untuk membagikan pelajaran yang didapatkan serta artefak terkait ke seluruh tim. Sebagai contoh, bagikan prosedur, panduan, tata kelola, dan praktik terbaik yang telah diperbarui melalui wiki yang dapat diakses. Bagikan skrip, kode, dan pustaka melalui repositori umum.
    - Mendelegasikan akses ke lingkungan AWS Anda
    - Bagikan repositori AWS CodeCommit
    - Otorisasi fungsi AWS Lambda secara mudah

- Membagikan AMI kepada Akun AWS tertentu
- Percepat pembagian tempat dengan URL desainer AWS CloudFormation
- Menggunakan AWS Lambda dengan Amazon SNS

Sumber daya

Dokumen terkait:

- Otorisasi fungsi AWS Lambda secara mudah
- Bagikan repositori AWS CodeCommit
- Membagikan AMI kepada Akun AWS tertentu
- Percepat pembagian tempat dengan URL desainer AWS CloudFormation
- Menggunakan AWS Lambda dengan Amazon SNS

Video terkait:

- Mendelegasikan akses ke lingkungan AWS Anda

OPS11-BP09 Mengalokasikan waktu untuk membuat peningkatan

Dedikasikan waktu dan sumber daya dalam proses Anda untuk memungkinkan peningkatan bertahap yang berkelanjutan.

Di AWS, Anda dapat membuat duplikat lingkungan sementara, menurunkan risiko, usaha, serta biaya eksperimen dan pengujian. Lingkungan duplikat ini dapat digunakan untuk menguji kesimpulan dari analisis dan eksperimen Anda, serta mengembangkan dan menguji peningkatan terencana.

Antipola umum:

- Ada masalah kinerja yang diketahui dalam aplikasi Anda. Ini ditambahkan ke backlog di balik setiap implementasi fitur terencana. Jika peringkat fitur terencana yang ditambahkan tetap konstan, masalah kinerja tidak akan pernah tertangani.
- Untuk mendukung peningkatan berkelanjutan yang disetujui, administrator dan developer menggunakan seluruh waktu tambahan mereka untuk memilih dan mengimplementasikan peningkatan. Tidak ada peningkatan yang diselesaikan.

Manfaat menerapkan praktik terbaik ini: Dengan mendedikasikan waktu dan sumber daya dalam proses, Anda memungkinkan peningkatan bertahap yang berkelanjutan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Rendah

Panduan implementasi

- Alokasikan waktu untuk membuat peningkatan: Dedikasikan waktu dan sumber daya dalam proses Anda untuk memungkinkan peningkatan bertahap yang berkelanjutan. Implementasikan perubahan guna meningkatkan dan mengevaluasi hasil untuk menentukan keberhasilan. Jika hasilnya tidak memenuhi tujuan, dan peningkatan masih menjadi prioritas, lakukan tindakan alternatif.

Keamanan

Pilar Keamanan berkenaan dengan kemampuan untuk melindungi data, sistem, dan aset untuk memanfaatkan teknologi cloud guna meningkatkan keamanan Anda. Anda dapat menemukan panduan preskriptif tentang implementasi di Laporan resmi Pilar Keamanan.

Area praktik terbaik

- Fondasi keamanan
- Manajemen identitas dan akses
- Deteksi
- Perlindungan infrastruktur
- Perlindungan data
- Respons insiden
- Keamanan aplikasi

Fondasi keamanan

Pertanyaan

- SEC 1. Bagaimana cara Anda mengoperasikan beban kerja Anda dengan aman?

SEC 1. Bagaimana cara Anda mengoperasikan beban kerja Anda dengan aman?

Untuk mengoperasikan beban kerja Anda dengan aman, Anda harus menerapkan praktik terbaik yang menyeluruh ke setiap area keamanan. Pilih persyaratan dan proses yang telah Anda tetapkan

dalam keunggulan operasional di tingkat organisasi dan beban kerja, lalu terapkan ke semua area. Terus mengikuti rekomendasi terbaru dari AWS dan industri serta informasi ancaman akan membantu Anda mengevolusikan model ancaman dan tujuan kontrol. Otomatisasi proses, pengujian, dan validasi keamanan memungkinkan Anda menskalakan operasi keamanan Anda.

Praktik terbaik

- SEC01-BP01 Memisahkan beban kerja menggunakan akun
- SEC01-BP02 Mengamankan properti dan pengguna root akun
- SEC01-BP03 Identifikasikan dan validasikan tujuan kontrol
- SEC01-BP04 Ikuti info terbaru tentang ancaman keamanan
- SEC01-BP05 Mengikuti informasi terbaru tentang rekomendasi keamanan
- SEC01-BP06 Mengotomatiskan pengujian dan validasi kontrol keamanan di pipeline
- SEC01-BP07 Mengidentifikasi ancaman dan memprioritaskan mitigasi menggunakan model ancaman
- SEC01-BP08 Mengevaluasi dan mengimplementasikan fitur serta layanan keamanan baru secara rutin

SEC01-BP01 Memisahkan beban kerja menggunakan akun

Terapkan pagar pembatas umum dan isolasi antarlingkungan (seperti produksi, pengembangan, dan pengujian) dan beban kerja melalui strategi multiakun. Pemisahan di tingkat akun sangat disarankan karena hal ini dapat memberikan batasan isolasi yang kuat untuk keamanan, tagihan, dan akses.

Hasil yang diinginkan: Struktur akun yang mengisolasi operasi cloud, beban kerja yang tidak saling berkaitan, dan lingkungan dalam akun terpisah, sehingga meningkatkan keamanan di seluruh infrastruktur cloud.

Antipola umum:

- Menempatkan beberapa beban kerja yang tidak saling berkaitan dengan berbagai tingkat sensitivitas data ke dalam akun yang sama.
- Struktur unit organisasi (OU) yang tidak ditentukan dengan baik.

Manfaat menjalankan praktik terbaik ini:

- Mengurangi cakupan dampak jika beban kerja tidak sengaja diakses.

- Tata kelola akses secara terpusat ke layanan, sumber daya, dan Wilayah AWS.
- Keamanan infrastruktur cloud terjaga dengan kebijakan dan administrasi terpusat pada layanan keamanan.
- Pembuatan akun dan proses pemeliharaan otomatis.
- Audit infrastruktur terpusat untuk persyaratan kepatuhan dan peraturan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Akun AWS memberikan batas isolasi keamanan di antara beban kerja atau sumber daya yang beroperasi pada tingkat sensitivitas yang berbeda. AWS menyediakan alat untuk mengelola beban kerja cloud Anda dalam skala besar melalui strategi multiakun untuk memanfaatkan batasan isolasi ini. Untuk panduan tentang konsep, pola, dan implementasi strategi multiakun di AWS, lihat Mengelola Lingkungan AWS Anda Menggunakan Beberapa Akun.

Ketika Anda memiliki beberapa Akun AWS di bawah manajemen pusat, akun Anda harus dikelola dalam hierarki yang ditentukan oleh lapisan unit organisasi (OU). Kontrol keamanan kemudian dapat diatur dan diterapkan ke OU dan akun anggota, yang menciptakan kontrol pencegahan yang konsisten pada akun anggota di organisasi. Kontrol keamanan diwariskan, memungkinkan Anda memfilter izin yang tersedia untuk akun anggota yang berada di tingkat yang lebih rendah dalam hierarki OU. Untuk membuat desain yang baik, memanfaatkan pewarisan ini untuk mengurangi jumlah dan kerumitan kebijakan keamanan yang diperlukan untuk mencapai kontrol keamanan yang diinginkan untuk setiap akun anggota.

AWS Organizations dan AWS Control Tower adalah dua layanan yang dapat Anda gunakan untuk mengimplementasikan dan mengelola struktur multiakun ini di lingkungan AWS Anda. Dengan AWS Organizations, Anda bisa mengatur akun ke dalam hierarki yang ditentukan oleh satu atau beberapa lapisan OU, dan setiap OU berisi sejumlah akun anggota. Kebijakan kontrol layanan (SCP) memungkinkan administrator organisasi untuk menetapkan kontrol pencegahan terperinci pada akun anggota, dan AWS Config dapat digunakan untuk membuat kontrol proaktif dan detektif pada akun anggota. Banyak layanan AWS berintegrasi dengan AWS Organizations untuk memberikan kontrol administratif terdelegasi dan melakukan tugas khusus layanan di semua akun anggota dalam organisasi.

Selain AWS Organizations, AWS Control Tower menyediakan penyiapan praktik terbaik sekali klik untuk lingkungan AWS multiakun dengan zona landasan. Zona landasan adalah titik masuk ke

lingkungan multiakun yang dibuat oleh Control Tower. Control Tower memiliki beberapa manfaat dibanding AWS Organizations. Tiga manfaat yang memberikan tata kelola akun yang lebih baik adalah:

- Pagar pembatas wajib terintegrasi yang diterapkan secara otomatis ke akun yang diterima di organisasi.
- Pagar pembatas opsional yang dapat diaktifkan atau dinonaktifkan untuk serangkaian OU tertentu.
- AWS Control Tower Account Factory menyediakan deployment otomatis untuk akun yang berisi dasar dan opsi konfigurasi yang telah disetujui sebelumnya di dalam organisasi Anda.

Langkah implementasi

1. Rancang struktur unit organisasi: Rancangan struktur organisasi yang tepat dapat mengurangi beban manajemen yang diperlukan untuk membuat dan mengelola kebijakan kontrol layanan dan kontrol keamanan lainnya. Struktur unit organisasi Anda harus selaras dengan struktur beban kerja, sensitivitas data, dan kebutuhan bisnis Anda.
2. Buat zona landasan untuk lingkungan multiakun: Zona landasan membentuk konsistensi keamanan dan landasan infrastruktur, sehingga organisasi Anda dapat dengan cepat mengembangkan, meluncurkan, dan melakukan deployment beban kerja. Anda dapat menggunakan zona landasan kustom atau AWS Control Tower untuk mengatur lingkungan Anda.
3. Terapkan pagar pembatas: Implementasikan pagar pembatas yang stabil untuk lingkungan Anda melalui zona landasan. AWS Control Tower menyediakan sederet kontrol wajib dan opsional yang dapat di-deploy. Deployment kontrol wajib dilakukan secara otomatis saat mengimplementasikan Control Tower. Lihat daftar kontrol yang sangat direkomendasikan dan opsional, kemudian implementasikan kontrol yang sesuai dengan kebutuhan Anda.
4. Batasi akses ke Wilayah yang baru ditambahkan: Untuk Wilayah AWS baru, sumber daya IAM seperti pengguna dan peran hanya disebarkan ke Wilayah yang Anda tentukan. Tindakan ini dapat dilakukan melalui konsol saat menggunakan Control Tower, atau dengan menyesuaikan kebijakan izin IAM di AWS Organizations.
5. Pertimbangkan AWS CloudFormation StackSets: StackSets membantu Anda saat melakukan deployment kebijakan, peran, dan grup IAM ke Wilayah dan Akun AWS yang berbeda dari templat yang disetujui.

Sumber daya

Praktik Terbaik Terkait:

- SEC02-BP04 Mengandalkan penyedia identitas terpusat

Dokumen terkait:

- AWS Control Tower
- Panduan Audit Keamanan AWS
- Praktik Terbaik IAM
- Menggunakan CloudFormation StackSets untuk menyediakan sumber daya di beberapa wilayah dan Akun AWS
- Pertanyaan Umum Organizations
- Terminologi dan konsep AWS Organizations
- Praktik Terbaik untuk Kebijakan Kontrol Layanan dalam Lingkungan Multiakun AWS Organizations
- Panduan Referensi Manajemen Akun AWS
- Mengatur Lingkungan AWS Anda Menggunakan Beberapa Akun

Video terkait:

- Aktifkan adopsi AWS dalam skala besar dengan otomatisasi dan tata kelola
- Praktik Terbaik Keamanan dengan Cara Well-Architected
- Membangun dan Mengatur Banyak Akun menggunakan AWS Control Tower
- Mengaktifkan Control Tower untuk Organisasi yang Ada

Lokakarya terkait:

- Control Tower Immersion Day (Hari Imersi Control Tower)

SEC01-BP02 Mengamankan properti dan pengguna root akun

Pengguna root adalah pengguna yang memiliki hak istimewa paling banyak dalam Akun AWS, dengan akses administratif penuh ke semua sumber daya di dalam akun, dan dalam beberapa kasus tidak dapat dibatasi oleh kebijakan keamanan. Menonaktifkan akses terprogram ke pengguna root, menerapkan kontrol yang sesuai untuk pengguna root, serta tidak menggunakan pengguna root secara rutin membantu mengurangi risiko tersebarnya kredensial root secara tidak sengaja dan penyusupan di lingkungan cloud.

Hasil yang diharapkan: Mengamankan pengguna root membantu mengurangi kemungkinan terjadinya kerusakan yang disengaja maupun tidak disengaja akibat penyalahgunaan kredensial pengguna root. Menerapkan kontrol detektif juga dapat memberikan peringatan kepada personel yang tepat saat ada tindakan dilakukan menggunakan pengguna root.

Antipola umum:

- Menggunakan pengguna root untuk tugas selain yang memerlukan kredensial pengguna root.
- Tidak menguji rencana darurat secara rutin untuk memverifikasi fungsi infrastruktur, proses, dan personel penting dalam keadaan darurat.
- Hanya mempertimbangkan alur masuk akun biasa dan tidak mempertimbangkan atau menguji metode pemulihan akun lainnya.
- Tidak menangani hal-hal yang digunakan dalam alur pemulihan akun seperti DNS, server email, dan penyedia telepon sebagai bagian dari perimeter keamanan penting.

Manfaat menjalankan praktik terbaik ini: Pengamanan akses ke pengguna root membangun kepercayaan bahwa tindakan di akun Anda terkontrol dan diaudit.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

AWS menawarkan alat untuk membantu mengamankan akun Anda. Namun, karena beberapa tindakan ini tidak diaktifkan secara default, Anda harus mengambil tindakan langsung untuk mengimplementasikannya. Pertimbangkan rekomendasi berikut sebagai langkah-langkah dasar untuk mengamankan Akun AWS Anda. Saat mengimplementasikan langkah-langkah ini, penting halnya untuk membangun sebuah proses penilaian dan pemantauan kontrol keamanan secara berkelanjutan.

Saat pertama kali membuat Akun AWS, Anda memulai dengan satu identitas yang memiliki akses lengkap ke semua layanan dan sumber daya AWS di akun tersebut. Identitas ini disebut pengguna root Akun AWS. Anda dapat masuk sebagai pengguna root menggunakan alamat email dan kata sandi yang digunakan untuk membuat akun. Karena tingginya tingkat akses yang diberikan untuk pengguna root AWS, Anda harus membatasi penggunaan pengguna root AWS hanya untuk tugas yang secara khusus membutuhkannya. Kredensial masuk pengguna root harus diamankan secara ketat. Selalu aktifkan autentikasi multi-faktor (MFA) untuk pengguna root Akun AWS.

Selain alur autentikasi normal untuk masuk ke pengguna root Anda menggunakan nama pengguna, kata sandi, perangkat autentikasi multi-faktor (MFA), ada alur pemulihan akun untuk masuk ke

pengguna root Akun AWS Anda dengan mengakses ke alamat email dan nomor telepon yang terkait dengan akun Anda. Oleh karena itu, pastikan Anda mengamankan akun email pengguna root yang digunakan untuk mengirimkan email pemulihan dan nomor telepon yang terkait dengan akun tersebut. Selain itu, pertimbangkan potensi rantai dependensi apabila alamat email yang terkait dengan pengguna root di-host di server email atau sumber daya layanan nama domain (DNS) dari Akun AWS yang sama.

Saat menggunakan AWS Organizations, ada beberapa Akun AWS yang masing-masing memiliki pengguna root. Satu akun ditetapkan sebagai akun manajemen dan beberapa lapisan akun anggota kemudian dapat ditambahkan di bawah akun manajemen. Prioritaskan pengamanan pengguna root di akun manajemen Anda, lalu hubungi pengguna root akun anggota Anda. Strategi pengamanan pengguna root akun manajemen Anda dapat berbeda dari pengguna root akun anggota, dan Anda dapat menerapkan kontrol keamanan preventif pada pengguna root akun anggota Anda.

Langkah implementasi

Langkah-langkah implementasi berikut direkomendasikan untuk membuat kontrol bagi pengguna root. Jika berlaku, referensi silang untuk rekomendasi dilakukan ke AWS Foundations Benchmark untuk CIS versi 1.4.0. Selain langkah-langkah ini, baca Panduan praktik terbaik AWS untuk mengamankan sumber daya dan Akun AWS Anda.

Kontrol preventif

1. Siapkan informasi kontak yang akurat untuk akun.
    a. Informasi ini digunakan untuk alur pemulihan kehilangan kata sandi, alur pemulihan kehilangan akun perangkat MFA, dan untuk komunikasi penting terkait keamanan dengan tim Anda.
    b. Gunakan alamat email yang di-host oleh domain perusahaan Anda, sebaiknya dari daftar distribusi, sebagai alamat email pengguna root Anda. Menggunakan daftar distribusi memberikan redundansi tambahan dan keberlanjutan akses ke akun root dalam waktu lama dibanding menggunakan akun email individu.
    c. Nomor telepon yang tercantum pada informasi kontak harus berupa telepon khusus dan aman untuk tujuan ini. Nomor telepon ini tidak boleh dicantumkan atau dibagikan kepada siapa pun.
2. Jangan membuat kunci akses untuk pengguna root. Jika ada kunci akses, langsung hapus (CIS 1.4).
    a. Hilangkan kredensial terprogram yang sudah lama (kunci rahasia dan akses) untuk pengguna root.

   b. Jika kunci akses pengguna root sudah ada, Anda harus mengubah proses yang menggunakan kunci tersebut untuk menggunakan kunci akses sementara dari peran AWS Identity and Access Management (IAM), kemudian hapus kunci akses pengguna root.

3. Tentukan apakah Anda perlu menyimpan kredensial untuk pengguna root.

   a. Jika Anda menggunakan AWS Organizations untuk membuat akun anggota baru, kata sandi awal untuk pengguna root pada akun anggota baru akan ditetapkan ke nilai acak yang tidak akan ditampilkan kepada Anda. Pertimbangkan untuk menggunakan alur pengaturan ulang kata sandi dari akun manajemen AWS Organization Anda untuk mendapatkan akses ke akun anggota jika diperlukan.

   b. Untuk Akun AWS atau akun AWS Organization manajemen terpisah, coba buat dan simpan kredensial dengan aman untuk pengguna root. Mengaktifkan MFA untuk pengguna root.

4. Aktifkan kontrol pencegahan untuk pengguna root akun anggota di lingkungan multiakun AWS.

   a. Pertimbangkan untuk mengaktifkan pagar pembatas preventif Jangan Izinkan Pembuatan Kunci Akses Root untuk Pengguna Root untuk akun anggota.

   b. Pertimbangkan untuk mengaktifkan pagar pembatas preventif Jangan Izinkan Tindakan sebagai Pengguna Root untuk akun anggota.

5. Jika Anda memerlukan kredensial untuk pengguna root:

   a. Gunakan kata sandi yang kompleks.

   b. Aktifkan autentikasi multi-faktor (MFA) untuk pengguna root, khususnya untuk akun manajemen (pembayar) AWS Organizations (CIS 1.5).

   c. Pertimbangkan perangkat MFA pada perangkat keras untuk ketahanan dan keamanan, karena perangkat sekali pakai dapat mengurangi kemungkinan perangkat yang berisi kode MFA Anda dapat digunakan kembali untuk tujuan lain. Pastikan baterai pada perangkat MFA perangkat keras diganti secara rutin. (CIS 1.6)

      - Untuk mengonfigurasi MFA bagi pengguna root, ikuti petunjuk untuk mengaktifkan MFA virtual atau perangkat MFA perangkat keras.

   d. Pertimbangkan untuk mendaftarkan beberapa perangkat MFA sebagai cadangan. Satu akun bisa memiliki maksimal 8 perangkat MFA.

      - Perlu diperhatikan bahwa mendaftarkan lebih dari satu perangkat MFA untuk pengguna root akan otomatis menonaktifkan alur untuk memulihkan akun Anda jika perangkat MFA hilang.

   e. Simpan kata sandi dengan aman, dan pertimbangkan dependensi melingkar jika menyimpan kata sandi secara elektronik. Jangan gunakan cara penyimpanan kata sandi yang memerlukan akses ke Akun AWS yang sama untuk mendapatkannya.

6. Opsional: Coba terapkan jadwal rotasi kata sandi untuk pengguna root secara berkala.
    - Praktik terbaik manajemen kredensial bergantung pada persyaratan peraturan dan kebijakan Anda. Pengguna root yang dilindungi oleh MFA tidak mengandalkan kata sandi sebagai satu faktor autentikasi.
    - Mengubah kata sandi pengguna root secara berkala mengurangi risiko pengungkapan kata sandi secara tidak sengaja yang dapat memicu penyalahgunaan.

Kontrol deteksi

- Buat alarm untuk mendeteksi penggunaan kredensial root (CIS 1.7). Mengaktifkan Amazon GuardDuty akan memantau dan memberikan peringatan terkait penggunaan kredensial API pengguna root melalui temuan RootCredentialUsage.
- Evaluasi dan implementasikan kontrol deteksi yang termasuk dalam Paket konformasi Pilar Keamanan AWS Well-Architected untuk AWS Config, atau jika menggunakan AWS Control Tower, kontrol yang sangat direkomendasikan dalam Control Tower.

Panduan operasional

- Tentukan siapa di organisasi Anda yang harus memiliki akses ke kredensial pengguna root.
    - Gunakan aturan dua orang sehingga tidak ada satu orang pun yang memiliki akses ke semua kredensial dan MFA yang diperlukan untuk mendapatkan akses pengguna root.
    - Pastikan bahwa organisasi, dan bukan perorangan, yang memegang kendali atas nomor telepon dan alias email yang terkait dengan akun (yang digunakan untuk alur pengaturan ulang kata sandi dan MFA).
- Gunakan pengguna root hanya untuk keperluan khusus (CIS 1.7).
    - Pengguna root AWS tidak boleh digunakan untuk tugas sehari-hari, bahkan tugas administratif. Hanya masuk sebagai pengguna root saat melakukan tugas AWS yang memerlukan pengguna root. Semua tindakan lainnya harus dilakukan oleh pengguna lain dengan peran yang sesuai.
- Periksa secara berkala apakah akses ke pengguna root berfungsi dengan baik sehingga prosedurnya telah teruji sebelum terjadi situasi darurat yang memerlukan penggunaan kredensial pengguna root.
- Periksa secara berkala apakah alamat email yang terkait dengan akun dan yang tercantum dalam Kontak Alternatif berfungsi dengan baik. Pantau kotak masuk email untuk melihat apakah ada notifikasi keamanan yang Anda terima <abuse@amazon.com>. Selain itu, pastikan semua nomor telepon yang terkait dengan akun saat ini berfungsi dengan baik.

- Siapkan prosedur respons insiden untuk merespons penyalahgunaan akun root. Lihat Panduan Respons Insiden Keamanan AWS dan praktik terbaik di bagian Respons Insiden dalam laporan resmi Pilar Keamanan untuk informasi lebih lanjut tentang membangun strategi respons insiden untuk Akun AWS Anda.

Sumber daya

Praktik Terbaik Terkait:

- SEC01-BP01 Memisahkan beban kerja menggunakan akun
- SEC02-BP01 Menggunakan mekanisme masuk yang kuat
- SEC03-BP02 Memberikan hak akses paling rendah
- SEC03-BP03 Menerapkan proses akses darurat
- SEC10-BP05 Menyediakan akses di awal

Dokumen terkait:

- AWS Control Tower
- Panduan Audit Keamanan AWS
- Praktik Terbaik IAM
- Amazon GuardDuty – peringatan penggunaan kredensial root
- Panduan langkah demi langkah tentang pemantauan untuk penggunaan kredensial root melalui CloudTrail
- Token MFA yang disetujui untuk digunakan dengan AWS
- Menerapkan akses pemicu peringatan di AWS
- 10 elemen keamanan teratas yang perlu ditingkatkan di Akun AWS Anda
- Apa yang harus saya lakukan ketika mendapati aktivitas tidak sah di akun Akun AWS saya?

Video terkait:

- Aktifkan adopsi AWS dalam skala besar dengan otomatisasi dan tata kelola
- Praktik Terbaik Keamanan dengan Cara Well-Architected
- Membatasi penggunaan kredensial root AWS dari AWS re:inforce 2022 – Praktik terbaik keamanan dengan AWS IAM

Lab dan contoh terkait:

- Lab: Akun AWS dan pengguna root

SEC01-BP03 Identifikasikan dan validasikan tujuan kontrol

Berdasarkan persyaratan kepatuhan dan risiko yang diidentifikasi dari model ancaman Anda, dapatkan dan validasikan tujuan kontrol dan kontrol yang perlu Anda terapkan pada beban kerja Anda. Validasi berkelanjutan terhadap tujuan kontrol dan kontrol dapat membantu Anda mengukur efektivitas mitigasi risiko.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Identifikasikan persyaratan kepatuhan: Temukan persyaratan organisasi, legal, dan kepatuhan yang harus dipatuhi oleh beban kerja Anda.
- Identifikasikan sumber daya kepatuhan AWS: Identifikasikan sumber daya yang disediakan oleh AWS untuk membantu Anda dengan kepatuhan.
  - https://aws.amazon.com/compliance/
  - https://aws.amazon.com/artifact/

Sumber daya

Dokumen terkait:

- AWSPanduan Audit Keamanan
- Buletin Keamanan

Video terkait:

- AWS Security Hub CSPM: Kelola Peringatan Keamanan dan Otomatiskan Kepatuhan
- Praktik Terbaik Keamanan dengan Cara Well-Architected

SEC01-BP04 Ikuti info terbaru tentang ancaman keamanan

Kenali vektor serangan dengan terus mengikuti perkembangan ancaman keamanan terbaru untuk membantu Anda menentukan dan menerapkan kontrol yang sesuai. Gunakan AWS Managed

Services untuk memudahkan Anda menerima pemberitahuan tentang perilaku yang tidak diharapkan atau tidak biasa di akun AWS Anda. Lakukan investigasi menggunakan alat Partner AWS atau feed informasi ancaman pihak ketiga sebagai bagian dari alur informasi keamanan Anda. Dengan Daftar Kerentanan dan Paparan Umum (CVE) mencantumkan kerentanan keamanan siber yang diuraikan secara publik dan dapat Anda gunakan untuk terus mengikuti perkembangan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Berlangganan ke sumber inteligensi ancaman: Tinjau informasi inteligensi ancaman secara rutin dari berbagai sumber yang relevan dengan teknologi yang digunakan di beban kerja Anda.
  - Daftar Kerentanan dan Paparan Umum
- Pertimbangkan layanan AWS Shield Advanced : Layanan ini menyediakan visibilitas waktu nyata ke sumber inteligensi, jika beban kerja Anda dapat diakses internet.

Sumber daya

Dokumen terkait:

- Panduan Audit Keamanan AWS
- AWS Shield
- Buletin Keamanan

Video terkait:

- Security Best Practices the Well-Architected Way

SEC01-BP05 Mengikuti informasi terbaru tentang rekomendasi keamanan

Ikuti terus rekomendasi keamanan AWS dan industri untuk mengembangkan postur keamanan beban kerja Anda. Buletin Keamanan AWS berisi informasi penting tentang notifikasi keamanan dan privasi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

- Ikuti informasi terbaru AWS: Lakukan langganan atau kunjungi secara rutin untuk mendapatkan saran, tips, dan trik baru.
  - Lab AWS Well-Architected
  - Blog keamanan AWS
  - Dokumentasi layanan AWS
- Lakukan langganan ke berita-berita industri: Rutin tinjau umpan berita dari berbagai sumber terkait teknologi yang digunakan di beban kerja Anda.
  - Contoh: Daftar Kerentanan dan Paparan Umum

Sumber daya

Dokumen terkait:

- Buletin Keamanan

Video terkait:

- Praktik Terbaik Keamanan dengan Cara Well-Architected

SEC01-BP06 Mengotomatiskan pengujian dan validasi kontrol keamanan di pipeline

Tetapkan acuan dasar (baseline) dan templat yang aman untuk mekanisme keamanan yang telah diuji dan divalidasi sebagai bagian dari build, pipeline, dan proses Anda. Gunakan alat dan otomatisasi untuk menguji dan memvalidasi semua kontrol keamanan secara terus-menerus. Misalnya, pindai item seperti image mesin dan templat infrastruktur sebagai kode (IaC) untuk menemukan kerentanan keamanan, kejanggalan, dan penyimpangan dari acuan dasar yang telah ditetapkan pada setiap tahap. AWS CloudFormation Guard dapat membantu Anda memverifikasi bahwa templat CloudFormation aman, menghemat waktu, dan mengurangi risiko kesalahan konfigurasi.

Mengurangi jumlah kesalahan konfigurasi keamanan yang dimasukkan ke dalam lingkungan produksi adalah hal penting—makin banyak kontrol kualitas dan pengurangan kecacatan yang dapat Anda lakukan dalam proses build, makin baik hasilnya. Rancang pipeline integrasi berkelanjutan dan deployment berkelanjutan (CI/CD) untuk menguji masalah keamanan setiap kali memungkinkan. Pipeline CI/CD menawarkan kesempatan untuk menyempurnakan keamanan pada tiap-tiap tahap

build dan pengiriman. Peralatan keamanan CI/CD juga harus terus diperbarui untuk memitigasi ancaman yang berkembang.

Lacak perubahan pada konfigurasi beban kerja Anda untuk membantu audit kepatuhan, manajemen perubahan, dan penyelidikan yang mungkin berlaku untuk Anda. Anda dapat menggunakan AWS Config untuk merekam dan mengevaluasi sumber daya AWS dan pihak ketiga Anda. Ini memungkinkan Anda untuk mengaudit dan menilai secara berkelanjutan keseluruhan kepatuhan terhadap aturan dan paket kesesuaian, yakni kumpulan aturan dengan tindakan perbaikan.

Pelacakan perubahan harus menyertakan perubahan terencana, yang merupakan bagian dari proses kontrol perubahan organisasi Anda (terkadang disebut MACD—Memindah, Menambah, Mengubah, Menghapus), perubahan tidak terencana, dan perubahan yang tidak diinginkan, seperti insiden. Perubahan dapat terjadi pada infrastruktur, tetapi mungkin juga berkaitan dengan kategori lain, seperti perubahan pada repositori kode, perubahan image mesin dan inventaris aplikasi, perubahan proses dan kebijakan, atau perubahan dokumentasi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

- Otomatiskan manajemen konfigurasi: Tegakkan dan validasi konfigurasi keamanan secara otomatis menggunakan layanan atau alat manajemen konfigurasi.
  - AWS Systems Manager
  - AWS CloudFormation
  - Menyiapkan Pipeline CI/CD di AWS

Sumber daya

Dokumen terkait:

- Cara menggunakan kebijakan kontrol layanan untuk menetapkan pagar pembatas izin di seluruh akun dalam Organisasi AWS Anda

Video terkait:

- Mengelola Lingkungan AWS Multiakun Menggunakan AWS Organizations
- Praktik Terbaik Keamanan dengan Cara Well-Architected

SEC01-BP07 Mengidentifikasi ancaman dan memprioritaskan mitigasi menggunakan model ancaman

Lakukan pemodelan ancaman untuk mengidentifikasi dan menyediakan daftar potensi ancaman terbaru serta mitigasi terkait untuk beban kerja Anda. Tentukan prioritas ancaman dan sesuaikan mitigasi kontrol keamanan Anda untuk mencegah, mendeteksi, dan merespons ancaman. Periksa kembali dan kelola sesuai dengan konteks beban kerja Anda, serta lanskap keamanan yang terus berubah.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Apa itu pemodelan ancaman?

“Pemodelan ancaman digunakan untuk mengidentifikasi, mengomunikasikan, serta memahami ancaman dan mitigasi untuk melindungi suatu nilai.” – The Open Web Application Security Project (OWASP) Application Threat Modeling

Mengapa pemodelan ancaman sebaiknya dilakukan?

Sistem begitu kompleks. Kompleksitas dan kemampuannya akan makin meningkat seiring waktu, sehingga memberikan nilai bisnis yang lebih banyak dan meningkatkan keterlibatan serta kepuasan pelanggan. Artinya, keputusan desain IT perlu mempertimbangkan peningkatan jumlah kasus penggunaan. Perubahan kompleksitas dan jumlah kasus penggunaan ini biasanya menjadikan pendekatan tidak terstruktur tidak efektif untuk menemukan dan memitigasi ancaman. Maka, Anda memerlukan pendekatan sistematis untuk menghitung potensi ancaman terhadap sistem, serta untuk melakukan dan memprioritaskan mitigasi untuk memastikan organisasi Anda dapat meningkatkan postur keamanan sistem secara keseluruhan dengan maksimal meski dengan sumber daya terbatas.

Pemodelan ancaman dirancang untuk memberikan pendekatan sistematis ini, bertujuan untuk menemukan dan mengatasi masalah dalam proses desain lebih awal, saat biaya dan upaya mitigasi relatif rendah dibandingkan setelahnya dalam siklus hidup. Pendekatan ini selaras dengan prinsip industri keamanan shift-left. Pada intinya, pemodelan ancaman akan terintegrasi dengan proses manajemen risiko organisasi serta membantu menentukan kontrol mana yang akan diimplementasikan menggunakan pendekatan menurut ancaman.

Kapan seharusnya pemodelan ancaman dilakukan?

Mulai pemodelan ancaman dalam siklus hidup beban kerja Anda sedini mungkin. Hal ini membuat Anda lebih fleksibel dalam menentukan tindakan untuk mengatasi ancaman yang teridentifikasi.

Seperti halnya bug perangkat lunak, makin dini Anda mengidentifikasi ancaman, makin hemat biaya penanganannya. Model ancaman adalah dokumen hidup (living document) dan akan terus berkembang seiring perubahan beban kerja Anda. Periksa kembali model ancaman Anda dari waktu ke waktu, termasuk saat terjadi perubahan besar, perubahan dalam lanskap ancaman, atau saat mengadopsi fitur atau layanan baru.

Langkah implementasi

Bagaimana cara menjalankan pemodelan ancaman?

Pemodelan ancaman bisa dijalankan dengan banyak cara. Seperti halnya bahasa pemrograman, setiap model memiliki kelebihan dan kekurangannya masing-masing. Pilih cara yang paling tepat untuk Anda. Salah satu pendekatannya adalah memulai dengan Shostack's 4 Question Frame for Threat Modeling yang berisi pertanyaan terbuka agar pengujian pemodelan ancaman Anda terstruktur:

1. Apa yang sedang Anda kerjakan?

   Pertanyaan ini bertujuan untuk membantu memahami dan menentukan sistem yang sedang Anda bangun serta detail sistem tersebut yang relevan dengan keamanan. Membuat model atau diagram adalah salah satu cara paling populer untuk menjawab pertanyaan ini karena dapat membantu Anda memvisualisasikan apa yang sedang Anda bangun, misalnya, menggunakan diagram alur data. Menuliskan asumsi dan detail penting tentang sistem Anda juga dapat membantu Anda menentukan cakupan. Hal ini membantu menyatukan fokus semua orang yang berkontribusi dalam model ancaman, serta menghindari topik di luar cakupan (termasuk versi lama sistem Anda) yang memakan banyak waktu. Misalnya, jika Anda sedang membangun aplikasi web, sepertinya tidak perlu membuang waktu melakukan pemodelan ancaman untuk urutan boot tepercaya sistem operasi untuk klien browser karena desain Anda tidak akan memengaruhi hal ini.
2. Apa saja potensi permasalahannya?

   Di sini Anda dapat mengidentifikasi ancaman terhadap sistem Anda. Ancaman adalah tindakan atau peristiwa yang disengaja atau tidak disengaja, yang dampaknya tidak diharapkan dan dapat memengaruhi keamanan sistem Anda. Tanpa mengetahui dengan jelas apa saja potensi permasalahannya, Anda tidak akan tahu cara penanganannya.

   Tidak ada daftar khusus tentang apa saja masalah yang dapat terjadi. Pembuatan daftar ini memerlukan diskusi dan kolaborasi antara setiap individu di dalam tim Anda serta pihak terkait yang terlibat dalam pengujian pemodelan ancaman. Anda dapat melengkapi diskusi dengan model untuk mengidentifikasi ancaman, seperti STRIDE, yang menunjukkan berbagai kategori evaluasi:

Penipuan (Spoofing), Gangguan (Tampering), Penolakan (Repudiation), Kebocoran Informasi (Information Disclosure), Penolakan Layanan (Denial of Service), dan Peningkatan Hak Istimewa (Elevation of Privilege). Selain itu, Anda mungkin ingin melengkapi diskusi dengan meninjau daftar yang sudah ada dan penelitian untuk inspirasi, termasuk OWASP Top 10, HiTrust Threat Catalog, dan katalog ancaman milik organisasi Anda.

3. Apa tindakan yang akan Anda lakukan?

   Sama seperti pertanyaan sebelumnya, tidak ada daftar khusus untuk semua kemungkinan mitigasi. Input dalam langkah ini adalah ancaman yang teridentifikasi, pelaku, dan area peningkatan dari langkah sebelumnya.

   Keamanan dan kepatuhan adalah tanggung jawab bersama antara Anda dan AWS. Perlu dipahami bahwa pertanyaan “Tindakan apa yang akan kita lakukan?” harus disertai dengan pertanyaan “Siapa yang bertanggung untuk melakukan hal ini?”. Memahami keseimbangan tanggung jawab antara Anda dan AWS membantu Anda menentukan cakupan pengujian pemodelan ancaman ke mitigasi dalam kontrol Anda, yang biasanya merupakan gabungan dari opsi konfigurasi layanan AWS dan mitigasi dari sistem Anda sendiri.

   Untuk porsi tanggung jawab bersama AWS, Anda akan melihat bahwa layanan AWS masuk dalam cakupan banyak program kepatuhan. Program-program tersebut akan membantu Anda memahami penerapan kontrol yang andal di AWS untuk menjaga keamanan dan kepatuhan cloud. Laporan audit dari program-program ini dapat diunduh oleh pelanggan AWS melalui AWS Artifact.

   Apa pun layanan AWS yang Anda gunakan, selalu ada elemen yang menjadi tanggung jawab pelanggan, dan mitigasi yang diselaraskan dengan tanggung jawab ini harus disertakan dalam model ancaman Anda. Untuk mitigasi kontrol keamanan bagi layanan AWS sendiri, Anda perlu mempertimbangkan implementasi kontrol keamanan di seluruh domain, termasuk domain seperti manajemen akses dan identitas (autentikasi dan otorisasi), perlindungan data (diam dan bergerak), keamanan infrastruktur, pencatatan log, dan pemantauan. Dokumentasi untuk setiap layanan AWS memiliki bab keamanan khusus yang menyediakan panduan tentang kontrol keamanan yang perlu dipertimbangkan sebagai mitigasi. Pertimbangkan kode yang Anda tulis dan dependensi kodenya karena hal ini sangat penting, serta tentukan kontrol yang dapat Anda terapkan untuk mengatasi ancaman. Kontrol ini dapat berupa validasi input, penanganan sesi, dan penanganan ikatan. Fokus pada kode kustom karena sebagian besar kerentanan seringnya terjadi di area ini.

4. Apakah kita melakukannya dengan baik?

Tujuannya adalah agar tim dan organisasi Anda dapat meningkatkan kualitas model ancaman dan kecepatan dalam melakukan pemodelan ancaman dari waktu ke waktu. Peningkatan ini adalah hasil dari gabungan praktik, pembelajaran, pengajaran, dan peninjauan. Agar Anda dan tim Anda dapat mempelajari lebih lanjut dan melakukan praktik langsung, sebaiknya ikuti lokakarya atau kursus pelatihan Pemodelan ancaman yang benar bagi pembangun. Selain itu, jika Anda mencari panduan tentang cara mengintegrasikan pemodelan ancaman ke dalam siklus hidup pengembangan organisasi Anda, lihat posting Cara memanfaatkan pemodelan ancaman di halaman AWS Security Blog.

Komposer Ancaman

Untuk membantu dan memandu Anda dalam membuat model ancaman, pertimbangkan untuk menggunakan alat Komposer Ancaman, yang bertujuan untuk mempercepat waktu untuk mencapai nilai saat membuat model ancaman. Alat ini membantu Anda melakukan hal berikut:

- Menulis pernyataan ancaman berguna yang selaras dengan tata bahasa ancaman yang bekerja dalam alur kerja non-linear alami
- Menghasilkan model ancaman yang dapat dibaca manusia
- Membuat model ancaman yang dapat dibaca mesin untuk memungkinkan Anda memperlakukan model ancaman sebagai kode
- Membantu Anda mengidentifikasi area peningkatan kualitas dan cakupan dengan cepat menggunakan Dasbor Wawasan

Untuk referensi lebih lanjut, kunjungi Komposer Ancaman dan beralih ke Ruang Kerja Contoh yang ditentukan sistem.

Sumber daya

Praktik Terbaik Terkait:

- SEC01-BP03 Identifikasikan dan validasikan tujuan kontrol
- SEC01-BP04 Ikuti info terbaru tentang ancaman keamanan
- SEC01-BP05 Mengikuti informasi terbaru tentang rekomendasi keamanan
- SEC01-BP08 Mengevaluasi dan mengimplementasikan fitur serta layanan keamanan baru secara rutin

Dokumen terkait:

- Cara memanfaatkan pemodelan ancaman (AWS Security Blog)
- NIST: Panduan untuk Pemodelan Ancaman Sistem yang Terpusat pada Data

Video terkait:

- AWS Summit ANZ 2021 - Cara memanfaatkan pemodelan ancaman
- AWS Summit ANZ 2022 - Menskalakan keamanan – Pengoptimalan untuk pengiriman yang cepat dan aman

Pelatihan terkait:

- Pemodelan ancaman yang benar bagi pembangun – Pelatihan mandiri virtual AWS Skill Builder
- Pemodelan ancaman yang benar bagi pembangun – AWS Workshop

Alat terkait:

- Komposer Ancaman

SEC01-BP08 Mengevaluasi dan mengimplementasikan fitur serta layanan keamanan baru secara rutin

Evaluasikan dan implementasikan fitur serta layanan keamanan dari Partner AWS dan AWS yang memungkinkan peningkatan postur keamanan beban kerja. Blog Keamanan AWS menyoroti fitur dan layanan baru AWS, panduan implementasi, dan panduan keamanan umum. Yang Baru dengan AWS? adalah cara terbaik untuk tetap mengetahui fitur, layanan, dan pengumuman baru dari AWS.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Rendah

Panduan implementasi

- Rencanakan peninjauan rutin: Buat kalender aktivitas peninjauan yang mencakup persyaratan kepatuhan, evaluasi fitur dan layanan keamanan baru AWS, serta tetap mengikuti berita terbaru industri.
- Temukan fitur dan layanan AWS: Temukan fitur keamanan yang tersedia untuk layanan yang Anda gunakan, dan tinjau fitur baru setelah dirilis.

  - Blog keamanan AWS
  - Buletin keamanan AWS
  - Dokumentasi layanan AWS
- Tentukan proses onboarding layanan AWS: Tentukan proses untuk onboarding layanan baru AWS. Sertakan cara Anda mengevaluasi layanan baru AWS untuk fungsionalitas, dan persyaratan kepatuhan untuk beban kerja Anda.
- Uji coba fitur dan layanan baru: Uji coba fitur dan layanan baru setelah dirilis di lingkungan nonproduksi yang direplikasi menyerupai lingkungan produksi Anda.
- Implementasikan mekanisme pertahanan lainnya: Implementasikan mekanisme otomatis untuk melindungi beban kerja, dan menelusuri opsi yang tersedia.
  - Mengatasi sumber daya AWS yang tidak patuh dengan Aturan AWS Config

Sumber daya

Video terkait:

- Praktik Terbaik Keamanan dengan Cara Well-Architected

Manajemen identitas dan akses

Pertanyaan

- SEC 2. Bagaimana cara mengelola autentikasi untuk manusia dan mesin?
- SEC 3. Bagaimana cara mengelola izin untuk manusia dan mesin?

SEC 2. Bagaimana cara mengelola autentikasi untuk manusia dan mesin?

Ada dua jenis identitas yang harus Anda kelola ketika menentukan pendekatan terhadap pengoperasian beban kerja AWS yang aman. Pemahaman tentang jenis identitas yang harus Anda kelola dan berikan akses akan membantu Anda memverifikasi identitas yang tepat memiliki akses ke sumber daya yang tepat dalam kondisi yang tepat.

Identitas Manusia: Administrator, developer, operator, dan pengguna akhir Anda memerlukan identitas untuk mengakses lingkungan dan aplikasi AWS Anda. Ini adalah anggota organisasi Anda, atau pengguna eksternal yang berkolaborasi dengan Anda, dan yang berinteraksi dengan sumber daya AWS Anda melalui browser web, aplikasi klien, atau alat baris perintah interaktif.

Identitas Mesin: Aplikasi layanan, alat operasional, dan beban kerja Anda memerlukan identitas untuk membuat permintaan ke layanan AWS, misalnya, untuk membaca data. Identitas ini mencakup mesin yang dijalankan di lingkungan AWS Anda, seperti instans Amazon EC2 atau fungsi AWS Lambda. Anda juga dapat mengelola identitas mesin untuk pihak eksternal yang membutuhkan akses. Selain itu, Anda mungkin juga memiliki mesin di luar AWS yang memerlukan akses ke lingkungan AWS Anda.

Praktik terbaik

- SEC02-BP01 Menggunakan mekanisme masuk yang kuat
- SEC02-BP02 Menggunakan kredensial sementara
- SEC02-BP03 Menyimpan dan menggunakan rahasia secara aman
- SEC02-BP04 Mengandalkan penyedia identitas terpusat
- SEC02-BP05 Mengaudit dan merotasi kredensial secara berkala
- SEC02-BP06 Manfaatkan grup dan atribut pengguna

SEC02-BP01 Menggunakan mekanisme masuk yang kuat

Proses masuk (autentikasi menggunakan kredensial masuk) dapat menimbulkan risiko jika tidak menggunakan mekanisme seperti autentikasi multi-faktor (MFA), khususnya ketika kredensial tanpa sengaja bocor atau mudah ditebak. Untuk mengurangi risiko ini, gunakan mekanisme masuk yang kuat dengan menerapkan MFA dan kebijakan kata sandi yang kuat.

Hasil yang diinginkan: Mengurangi risiko adanya akses yang tidak diinginkan ke kredensial di AWS menggunakan mekanisme masuk yang kuat bagi pengguna AWS Identity and Access Management (IAM), pengguna root Akun AWS, AWS IAM Identity Center (pengganti AWS Single Sign-On), dan penyedia identitas pihak ketiga. Tujuan ini dapat dicapai dengan MFA, menerapkan kebijakan kata sandi yang kuat, dan mendeteksi perilaku masuk tidak wajar.

Antipola umum:

- Tidak menerapkan kebijakan kata sandi yang kuat untuk identitas Anda, termasuk kata sandi yang kompleks dan MFA.
- Kredensial yang sama digunakan oleh pengguna yang berbeda.
- Tidak menggunakan kontrol deteksi untuk aktivitas masuk yang mencurigakan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Ada banyak cara bagi identitas manusia masuk untuk ke AWS. Ini adalah praktik terbaik AWS untuk mengandalkan penyedia identitas terpusat menggunakan federasi (federasi langsung atau dengan AWS IAM Identity Center) saat melakukan autentikasi ke AWS. Dalam kasus tersebut, Anda harus membuat proses masuk yang aman dengan penyedia identitas atau Microsoft Active Directory.

Saat pertama kali membuka Akun AWS, Anda akan memulainya dengan pengguna root Akun AWS. Hanya gunakan akun pengguna root untuk mengonfigurasikan akses bagi pengguna Anda (dan untuk tugas yang memerlukan pengguna root). Penting halnya untuk segera mengaktifkan MFA bagi akun pengguna root setelah membuka Akun AWS Anda dan mengamankan pengguna root menggunakan panduan praktik terbaik AWS.

Jika Anda membuat pengguna di AWS IAM Identity Center, amankan proses masuk dalam layanan tersebut. Untuk identitas konsumen, Anda dapat menggunakan Amazon Cognito user pools dan mengamankan proses masuk dalam layanan tersebut, atau dengan salah satu penyedia identitas yang didukung Amazon Cognito user pools.

Jika Anda menggunakan pengguna AWS Identity and Access Management (IAM), amankan proses masuk menggunakan IAM.

Apa pun metode masuknya, menerapkan kebijakan masuk yang kuat adalah hal yang sangat penting.

Langkah implementasi

Berikut ini adalah rekomendasi mekanisme masuk yang kuat secara umum. Pengaturan aktual yang Anda konfigurasikan harus diatur sesuai kebijakan perusahaan Anda atau gunakan standar seperti NIST 800-63.

- Terapkan MFA. Ini adalah praktik terbaik IAM untuk menerapkan MFA untuk beban kerja dan identitas manusia. Mengaktifkan MFA akan memberikan lapisan keamanan tambahan yang mengharuskan pengguna untuk memberikan kredensial masuk dan kata sandi sekali pakai (OTP) atau string yang dibuat dan diverifikasi secara kriptografik dari perangkat untuk perangkat keras.
- Terapkan batas minimum karakter kata sandi, yang merupakan faktor utama dari kekuatan kata sandi.
- Terapkan kompleksitas kata sandi agar tidak mudah ditebak.
- Izinkan pengguna mengubah kata sandi mereka sendiri.

- Buat identitas individu, bukan kredensial bersama. Dengan membuat identitas individu, Anda dapat memberikan kredensial keamanan yang unik kepada setiap pengguna. Pengguna individu juga memungkinkan pengauditan aktivitas setiap pengguna.

Rekomendasi IAM Identity Center:

- IAM Identity Center memberikan kebijakan kata sandi yang telah ditentukan sebelumnya apabila menggunakan direktori default yang menerapkan persyaratan jumlah karakter, kompleksitas, dan penggunaan ulang kata sandi.
- Aktifkan MFA dan konfigurasikan pengaturan sesuai konteks (context-aware) atau selalu aktif (always-on) untuk MFA saat sumber identitas merupakan AWS Managed Microsoft AD, AD Connector, atau direktori default.
- Izinkan pengguna untuk mendaftarkan perangkat MFA miliknya.

Rekomendasi direktori Amazon Cognito user pools:

- Konfigurasikan pengaturan Kekuatan kata sandi.
- Terapkan MFA bagi pengguna.
- Gunakan pengaturan keamanan lanjutan Amazon Cognito user pools untuk fitur seperti autentikasi adaptif yang dapat memblokir aktivitas masuk yang mencurigakan.

Rekomendasi pengguna IAM:

- Idealnya, Anda menggunakan IAM Identity Center atau federasi langsung. Namun, Anda mungkin membutuhkan pengguna IAM. Dalam kasus seperti itu, atur kebijakan kata sandi untuk pengguna IAM. Anda dapat menggunakan kebijakan kata sandi untuk menentukan persyaratan, seperti minimum karakter atau apakah kata sandi harus terdiri dari karakter nonalfabet.
- Buat kebijakan IAM untuk menerapkan mekanisme masuk MFA sehingga pengguna dapat mengelola perangkat MFA dan kata sandi miliknya.

Sumber daya

Praktik Terbaik Terkait:

- SEC02-BP03 Menyimpan dan menggunakan rahasia secara aman
- SEC02-BP04 Mengandalkan penyedia identitas terpusat

- SEC03-BP08 Membagikan sumber daya secara aman dalam organisasi Anda

Dokumen terkait:

- Kebijakan Kata Sandi AWS IAM Identity Center (pengganti AWS Single Sign-On)
- Kebijakan kata sandi pengguna IAM
- Mengatur kata sandi pengguna root Akun AWS
- Kebijakan kata sandi Amazon Cognito
- Kredensial AWS
- Praktik terbaik keamanan IAM

Video terkait:

- Mengelola izin pengguna dalam skala besar dengan AWS IAM Identity Center
- Menguasai identitas di setiap lapisan susunan

SEC02-BP02 Menggunakan kredensial sementara

Saat melakukan autentikasi jenis apa pun, sebaiknya gunakan kredensial sementara daripada kredensial jangka panjang untuk mengurangi atau menghindari risiko seperti pengungkapan, pembagian, dan pencurian kredensial.

Hasil yang diinginkan: Untuk mengurangi risiko kredensial jangka panjang, sebisa mungkin gunakan kredensial sementara untuk identitas mesin dan manusia. Kredensial jangka panjang menimbulkan banyak risiko, misalnya, dapat diunggah ke repositori GitHub publik dalam bentuk kode. Dengan kredensial sementara, Anda dapat secara signifikan mengurangi risiko penyusupan kredensial.

Antipola umum:

- Developer memilih menggunakan kunci akses jangka panjang dari IAM users dibanding memperoleh kredensial sementara dari CLI menggunakan federasi.
- Developer menyematkan kunci akses jangka panjang dalam kodenya dan mengunggah kode tersebut ke repositori Git publik.
- Developer menyematkan kunci akses jangka panjang di aplikasi seluler yang kemudian dibuat tersedia di toko aplikasi.

- Pengguna membagikan kunci akses jangka panjang kepada pengguna lainnya, atau karyawan yang sudah keluar dari perusahaan tetapi masih memiliki kunci akses jangka panjang.
- Menggunakan kunci akses jangka panjang untuk identitas mesin meski kredensial sementara dapat digunakan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Gunakan kredensial keamanan sementara, bukan kredensial jangka panjang untuk semua permintaan CLI dan API AWS. Permintaan CLI dan API ke layanan AWS harus, hampir di setiap kasus, ditandatangani menggunakan kunci akses AWS. Permintaan ini dapat ditandatangani dengan kredensial jangka panjang maupun sementara. Satu-satunya situasi yang perlu menggunakan kredensial jangka panjang, disebut juga kunci akses jangka panjang, adalah ketika Anda menggunakan pengguna IAM atau pengguna root Akun AWS. Saat Anda bergabung ke AWS atau mengambil peran IAM melalui metode lainnya, kredensial sementara akan dibuat. Bahkan ketika Anda mengakses Konsol Manajemen AWS menggunakan kredensial masuk, kredensial sementara akan dibuat untuk Anda untuk melakukan panggilan ke layanan AWS. Anda hanya memerlukan kredensial jangka panjang untuk beberapa situasi saja; hampir semua tugas dapat dilakukan menggunakan kredensial sementara.

Menghindari penggunaan kredensial jangka panjang dan mengutamakan kredensial sementara harus diikuti dengan penerapan strategi pengurangan penggunaan pengguna IAM untuk mengutamakan federasi dan peran IAM. Meski sebelumnya pengguna IAM sudah digunakan untuk identitas mesin dan manusia, kini sebaiknya jangan gunakan pengguna tersebut untuk menghindari risiko dalam penggunaan kunci akses jangka panjang.

Langkah implementasi

Untuk identitas manusia seperti karyawan, administrator, developer, operator, dan pelanggan:

- Anda harus mengandalkan penyedia identitas terpusat dan dan mengharuskan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS menggunakan kredensial sementara. Federasi untuk pengguna Anda dapat dilakukan dengan federasi langsung ke setiap Akun AWS atau menggunakan AWS IAM Identity Center (pengganti AWS IAM Identity Center) dan penyedia identitas yang Anda pilih. Selain mengurangi penggunaan kredensial jangka panjang, federasi memberikan berbagai manfaat atas penggunaan pengguna IAM. Pengguna Anda juga dapat meminta kredensial sementara dari baris perintah untuk federasi langsung atau

menggunakan IAM Identity Center. Artinya, ada beberapa kasus penggunaan yang memerlukan kredensial jangka panjang atau pengguna IAM untuk pengguna Anda.

- Saat memberi pihak ketiga, seperti penyedia perangkat lunak sebagai layanan (SaaS), akses ke sumber daya di Akun AWS Anda, Anda dapat menggunakan peran lintas akun dan kebijakan berbasis sumber daya.
- Jika Anda perlu memberi aplikasi untuk konsumen atau pelanggan akses ke sumber daya AWS Anda, Anda dapat menggunakan kolam identitas Amazon Cognito atau Amazon Cognito user pools untuk menyediakan kredensial sementara. Izin untuk kredensial ini dikonfigurasikan lewat peran IAM. Anda juga dapat menentukan peran IAM terpisah dengan izin terbatas untuk pengguna tamu yang tidak diautentikasi.

Untuk identitas mesin, Anda mungkin perlu menggunakan kredensial jangka panjang. Dalam kasus tersebut, Anda harus mewajibkan beban kerja untuk menggunakan kredensial sementara dengan peran IAM untuk mengakses AWS.

- Untuk Amazon Elastic Compute Cloud (Amazon EC2), Anda dapat menggunakan peran untuk Amazon EC2.

- AWS Lambda memungkinkan Anda untuk mengonfigurasikan peran eksekusi Lambda guna memberikan izin layanan untuk melakukan tindakan AWS menggunakan kredensial sementara. Ada banyak model serupa untuk layanan AWS yang digunakan untuk memberikan kredensial sementara menggunakan peran IAM.
- Untuk perangkat IoT, Anda dapat menggunakan penyedia kredensial AWS IoT Core untuk meminta kredensial sementara.
- Untuk sistem on-premise atau sistem yang berjalan di luar AWS yang memerlukan akses ke sumber daya AWS, Anda dapat menggunakan IAM Roles Anywhere.

Dalam beberapa skenario, kredensial sementara tidak dapat digunakan dan Anda mungkin perlu menggunakan kredensial jangka panjang. Dalam situasi tersebut, audit dan rotasikan kredensial secara berkala serta rotasikan kunci akses secara rutin untuk kasus penggunaan yang memerlukan kredensial jangka panjang. Beberapa contoh yang dapat mengharuskan kredensial jangka panjang termasuk plugin WordPress dan klien pihak ketiga AWS. Dalam situasi yang mengharuskan Anda menggunakan kredensial jangka panjang, atau untuk kredensial selain kunci akses AWS, seperti masuk ke basis data, Anda dapat menggunakan layanan yang dirancang untuk menangani manajemen rahasia, seperti AWS Secrets Manager. Secrets Manager memudahkan manajemen,

rotasi, dan penyimpanan rahasia terenkripsi dengan aman menggunakan layanan yang didukung. Untuk informasi lebih lanjut tentang merotasi kredensial jangka panjang, lihat merotasi kunci akses.

Sumber daya

Praktik Terbaik Terkait:

- SEC02-BP03 Menyimpan dan menggunakan rahasia secara aman
- SEC02-BP04 Mengandalkan penyedia identitas terpusat
- SEC03-BP08 Membagikan sumber daya secara aman dalam organisasi Anda

Dokumen terkait:

- Kredensial Keamanan Sementara
- Kredensial AWS
- Praktik Terbaik Keamanan IAM
- Peran IAM
- IAM Identity Center
- Penyedia Identitas dan Federasi
- Merotasi Kunci Akses
- Solusi Partner Keamanan: Akses dan Kontrol Akses
- Pengguna Root Akun AWS

Video terkait:

- Mengelola izin pengguna dalam skala besar dengan AWS IAM Identity Center (pengganti AWS IAM Identity Center)
- Menguasai identitas di setiap lapisan susunan

SEC02-BP03 Menyimpan dan menggunakan rahasia secara aman

Beban kerja memerlukan kemampuan otomatis untuk membuktikan identitasnya ke basis data, sumber daya, dan layanan pihak ketiga. Hal ini dapat dilakukan menggunakan kredensial akses rahasia, seperti kunci akses API, kata sandi, dan token OAuth. Menggunakan layanan yang

dibuat khusus untuk menyimpan, mengelola, dan merotasi kredensial ini membantu mengurangi kemungkinan penyusupan kredensial.

Hasil yang diinginkan: Mengimplementasikan mekanisme untuk mengelola kredensial aplikasi secara aman, yang mencapai tujuan berikut:

- Mengidentifikasi rahasia apa yang diperlukan untuk beban kerja.
- Mengurangi kebutuhan kredensial jangka panjang yang diperlukan dan menggunakan kredensial jangka pendek jika memungkinkan.
- Membangun penyimpanan yang aman dan rotasi otomatis untuk kredensial jangka panjang yang tersisa.
- Mengaudit akses ke rahasia yang ada di beban kerja.
- Pemantauan berkelanjutan untuk memverifikasi bahwa tidak ada rahasia yang disematkan di kode sumber selama proses pengembangan.
- Mengurangi kemungkinan pengungkapan kredensial secara tidak sengaja.

Antipola umum:

- Tidak merotasi kredensial.
- Menyimpan kredensial jangka panjang dalam kode sumber atau file konfigurasi.
- Menyimpan kredensial diam tanpa dienkripsi.

Manfaat menjalankan praktik terbaik ini:

- Rahasia yang disimpan diamankan dengan enkripsi saat diam maupun bergerak.
- Akses ke kredensial harus melewati API (bayangkan ini seperti mesin penjual otomatis kredensial).
- Akses ke kredensial (baca dan tulis) diaudit dan dicatat.
- Pemisahan masalah (Separation of concerns): rotasi kredensial dilakukan oleh komponen terpisah, yang dapat dipisahkan dari bagian arsitektur lainnya.
- Rahasia didistribusikan secara otomatis ke komponen perangkat lunak sesuai permintaan dan rotasi dilakukan di lokasi pusat.
- Akses ke kredensial dapat dikontrol dengan sangat ketat.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Dahulu, kredensial digunakan untuk mengautentikasi basis data, API pihak ketiga, token, dan rahasia lainnya yang mungkin disematkan dalam kode sumber atau dalam file lingkungan. AWS menyediakan beberapa mekanisme untuk menyimpan kredensial ini secara aman, merotasinya secara otomatis, dan mengaudit penggunaannya.

Cara terbaik untuk mengelola rahasia adalah dengan mengikuti panduan penghapusan, penggantian, dan rotasi. Kredensial yang paling aman adalah kredensial yang tidak perlu Anda simpan, kelola, atau tangani. Jika ada kredensial yang sudah tidak digunakan untuk menjalankan beban kerja, Anda dapat menghapusnya.

Apabila kredensial masih diperlukan untuk menjalankan beban kerja dengan benar, kredensial jangka panjangnya mungkin bisa diganti dengan kredensial sementara atau jangka pendek. Misalnya, daripada melakukan hard-coding kunci akses rahasia AWS, coba ganti kredensial jangka panjang tersebut dengan kredensial sementara menggunakan peran IAM.

Beberapa rahasia yang sudah lama ada mungkin tidak dapat dihapus atau diganti. Rahasia tersebut dapat disimpan dalam layanan seperti AWS Secrets Manager, tempat rahasia bisa disimpan, dikelola, dan dirotasi secara rutin dan terpusat.

Pengauditan file konfigurasi dan kode sumber beban kerja dapat menunjukkan berbagai jenis kredensial. Tabel berikut merangkum strategi yang digunakan untuk menangani jenis kredensial umum:

Credential type	Description	Suggested strategy
IAM access keys	AWS IAM access and secret keys used to assume IAM roles inside of a workload	Replace: Use Peran IAM assigned to the compute instances (such as Amazon EC2 or AWS Lambda) instead. For interoperability with third parties that require access to resources in your Akun AWS, ask if they support Akses lintas akun AWS. For mobile apps, consider using temporary credentials through Kolam identitas (identitas

Credential type	Description	Suggested strategy
		gabungan) Amazon Cognito. For workloads running outside of AWS, consider IAM Roles Anywhere or AWS Systems Manager Hybrid Activations.
SSH keys	Secure Shell private keys used to log into Linux EC2 instances, manually or as part of an automated process	Replace: Use AWS Systems Manager or EC2 Instance Connect to provide programmatic and human access to EC2 instances using IAM roles.
Application and database credentials	Passwords – plain text string	Rotate: Store credentials in AWS Secrets Manager and establish automated rotation if possible.
Amazon RDS and Aurora Admin Database credentials	Passwords – plain text string	Replace: Use the Integrasi Secrets Manager dengan Amazon RDS or Amazon Aurora. In addition, some RDS database types can use IAM roles instead of passwords for some use cases (for more detail, see Autentikasi basis data IAM).
OAuth tokens	Secret tokens – plain text string	Rotate: Store tokens in AWS Secrets Manager and configure automated rotation.
API tokens and keys	Secret tokens – plain text string	Rotate: Store in AWS Secrets Manager and establish automated rotation if possible.

Antipola umumnya adalah menyematkan kunci akses IAM ke dalam kode sumber, file konfigurasi, atau aplikasi seluler. Saat kunci akses IAM diperlukan untuk berkomunikasi dengan layanan AWS, gunakan kredensial keamanan sementara (jangka-pendek). Kredensial jangka pendek tersebut dapat diberikan melalui peran IAM untuk instans EC2, peran eksekusi untuk fungsi Lambda, peran IAM Cognito untuk akses pengguna seluler, dan kebijakan IoT Core untuk perangkat IoT. Saat beroperasi dengan pihak ketiga, utamakan mendelegasikan akses ke peran IAM dengan akses yang diperlukan ke sumber daya akun Anda daripada mengonfigurasikan pengguna IAM lalu mengirim kunci akses rahasia kepada pihak ketiga untuk pengguna tersebut.

Dalam banyak kasus, beban kerja memerlukan penyimpanan rahasia agar dapat saling beroperasi dengan sumber daya dan layanan lainnya. AWS Secrets Manager dibuat khusus untuk mengelola kredensial ini secara aman, sekaligus penyimpanan, penggunaan, dan rotasi token API, kata sandi, serta kredensial lainnya.

AWS Secrets Manager menyediakan lima kemampuan utama untuk memastikan keamanan penyimpanan dan penanganan kredensial sensitif: enkripsi diam, enkripsi bergerak, pengauditan menyeluruh, kontrol akses terperinci, dan rotasi kredensial yang dapat diperluas. Layanan manajemen rahasia lainnya dari Partner AWS atau solusi yang dikembangkan secara lokal yang memberikan kemampuan dan jaminan serupa juga dapat digunakan.

Langkah implementasi

1. Identifikasi jalur kode yang berisi kredensial hard-coding menggunakan alat otomatis seperti Amazon CodeGuru.
   - Gunakan Amazon CodeGuru untuk memindai repositori kode Anda. Setelah peninjauan selesai, filter `Type=Secrets` di CodeGuru untuk menemukan baris kode yang bermasalah.
2. Identifikasi kredensial yang dapat dihapus atau diganti.
   a. Identifikasi kredensial yang sudah tidak diperlukan, lalu tandai untuk dihapus.
   b. Untuk Kunci Rahasia AWS yang tersemat dalam kode sumber, ganti dengan peran IAM yang terkait dengan sumber daya yang diperlukan. Jika bagian beban kerja Anda berada di luar AWS tetapi memerlukan kredensial IAM untuk mengakses sumber daya AWS, pertimbangkan untuk menggunakan IAM Roles Anywhere atau AWS Systems Manager Hybrid Activations.
3. Untuk rahasia lama lainnya dari pihak ketiga yang memerlukan penggunaan strategi rotasi, integrasikan Secrets Manager ke dalam kode Anda untuk mengambil rahasia pihak ketiga pada waktu proses.
   a. Konsol CodeGuru dapat secara otomatis membuat rahasia di Secrets Manager menggunakan kredensial yang ditemukan.

b. Integrasikan pengambilan rahasia dari Secrets Manager ke dalam kode aplikasi Anda.
   - Fungsi Lambda nirserver dapat menggunakan ekstensi Lambda bahasa agnostik.
   - Untuk instans atau kontainer EC2, AWS menyediakan contoh kode sisi klien untuk pengambilan rahasia dari Secrets Manager dalam beberapa bahasa pemrograman yang populer.

4. Tinjau basis kode Anda secara berkala dan pindai kembali untuk memverifikasi bahwa tidak ada rahasia baru yang ditambahkan ke kode.
   - Pertimbangkan untuk menggunakan alat bantu seperti git-secrets untuk mencegah masuknya rahasia baru ke repositori kode sumber Anda.
5. Pantau aktivitas Secrets Manager untuk mengetahui apakah ada penggunaan yang tidak diharapkan, akses rahasia yang tidak sesuai, atau upaya penghapusan rahasia.
6. Kurangi akses manusia ke kredensial. Batasi akses membaca, menulis, dan memodifikasi kredensial untuk peran IAM khusus untuk tujuan ini, serta hanya sediakan akses untuk mengambil peran ke sebagian kecil pengguna operasional.

Sumber daya

Praktik Terbaik Terkait:

- SEC02-BP02 Menggunakan kredensial sementara
- SEC02-BP05 Mengaudit dan merotasi kredensial secara berkala

Dokumen terkait:

- Mulai menggunakan AWS Secrets Manager
- Penyedia Identitas dan Federasi
- Amazon CodeGuru Memperkenalkan Pendeteksi Rahasia
- Bagaimana AWS Secrets Manager menggunakan AWS Key Management Service
- Enkripsi dan dekripsi rahasia di Secrets Manager
- Entri blog Secrets Manager
- Amazon RDS mengumumkan integrasi dengan AWS Secrets Manager

Video terkait:

- Praktik Terbaik untuk Mengelola, Mengambil, dan Merotasi Rahasia dalam Skala Besar
- Temukan Rahasia yang Sudah Diberi Hard-Code Menggunakan Pendeteksi Rahasia Amazon CodeGuru
- Mengamankan Rahasia untuk Beban Kerja Hibrida Menggunakan AWS Secrets Manager

Lokakarya terkait:

- Menyimpan, mengambil, dan mengelola kredensial sensitif di AWS Secrets Manager
- AWS Systems Manager Hybrid Activations

SEC02-BP04 Mengandalkan penyedia identitas terpusat

Untuk identitas tenaga kerja (karyawan dan kontraktor), andalkan penyedia identitas yang memungkinkan Anda mengelola identitas di tempat terpusat. Ini akan mempermudah pengelolaan akses di beberapa aplikasi dan sistem, karena Anda membuat, menetapkan, mengelola, mencabut, dan mengaudit akses dari satu lokasi.

Hasil yang diinginkan: Anda memiliki penyedia identitas terpusat tempat Anda mengelola pengguna tenaga kerja, kebijakan autentikasi (misalnya mengharuskan autentikasi multifaktor (MFA)), dan otorisasi ke sistem dan aplikasi (misalnya menetapkan akses berdasarkan keanggotaan atau atribut grup pengguna) secara terpusat. Pengguna tenaga kerja Anda masuk ke penyedia identitas pusat dan melakukan penggabungan (masuk tunggal) ke aplikasi internal dan eksternal, sehingga pengguna tidak perlu mengingat lebih dari satu kredensial. Penyedia identitas Anda terintegrasi dengan sistem sumber daya manusia (SDM) Anda sehingga perubahan personel secara otomatis disinkronkan ke penyedia identitas Anda. Misalnya, jika seseorang keluar dari organisasi Anda, Anda dapat secara otomatis mencabut akses ke aplikasi dan sistem gabungan (termasuk AWS). Anda telah mengaktifkan pencatatan log audit mendetail di penyedia identitas Anda dan memantau log tersebut untuk perilaku pengguna yang tidak biasa.

Antipola umum:

- Anda tidak menggunakan federasi dan masuk tunggal. Pengguna tenaga kerja Anda membuat akun dan kredensial pengguna terpisah di beberapa aplikasi dan sistem.
- Anda belum mengotomatiskan siklus hidup identitas untuk pengguna tenaga kerja, seperti dengan mengintegrasikan penyedia identitas Anda dengan sistem SDM Anda. Saat pengguna keluar dari organisasi atau beralih jabatan, Anda mengikuti proses manual untuk menghapus atau memperbarui catatan mereka di beberapa aplikasi dan sistem.

Manfaat menjalankan praktik terbaik ini: Dengan menggunakan penyedia identitas yang terpusat, Anda memiliki satu tempat untuk mengelola identitas dan kebijakan pengguna tenaga kerja, kemampuan untuk menetapkan akses aplikasi kepada pengguna dan grup, dan kemampuan untuk memantau aktivitas masuk pengguna. Melalui integrasi dengan sistem sumber daya manusia (SDM), ketika pengguna beralih jabatan, perubahan ini disinkronkan dengan penyedia identitas yang secara otomatis memperbarui aplikasi dan izin yang ditetapkan. Ketika pengguna keluar dari organisasi Anda, identitas mereka secara otomatis dinonaktifkan di penyedia identitas, sehingga akses mereka ke aplikasi dan sistem gabungan dicabut.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Panduan untuk pengguna tenaga kerja yang mengakses AWS

Pengguna tenaga kerja seperti karyawan dan kontraktor di organisasi Anda mungkin memerlukan akses ke AWS menggunakan Konsol Manajemen AWS atau AWS Command Line Interface (AWS CLI) untuk menjalankan fungsi pekerjaan mereka. Anda dapat memberikan akses AWS kepada pengguna tenaga kerja Anda dengan melakukan federasi dari penyedia identitas terpusat Anda ke AWS pada dua tingkat: federasi langsung ke setiap Akun AWS atau federasi ke beberapa akun di organisasi AWS.

- Untuk menggabungkan pengguna tenaga kerja Anda secara langsung dengan setiap Akun AWS, Anda dapat menggunakan penyedia identitas terpusat untuk digabungkan ke AWS Identity and Access Management dalam akun tersebut. Fleksibilitas IAM memungkinkan Anda mengaktifkan Penyedia Identitas SAML 2.0 atau Open ID Connect (OIDC) secara terpisah untuk setiap Akun AWS dan gunakan atribut pengguna gabungan untuk kontrol akses. Pengguna tenaga kerja Anda akan menggunakan browser web mereka untuk masuk ke penyedia identitas dengan memberikan kredensialnya (seperti kata sandi dan kode token MFA). Penyedia identitas mengeluarkan pernyataan SAFL ke browser mereka yang dikirimkan ke URL masuk Konsol Manajemen AWS agar pengguna dapat melakukan masuk tunggal ke Konsol Manajemen AWS dengan mengambil Peran IAM. Pengguna Anda juga dapat memperoleh kredensial API AWS sementara untuk digunakan di AWS CLI atau SDK AWS dari AWS STS dengan mengambil peran IAM menggunakan pernyataan SAFL dari penyedia identitas.
- Untuk menggabungkan pengguna tenaga kerja Anda dengan beberapa akun di organisasi AWS Anda, Anda dapat menggunakan AWS IAM Identity Center untuk mengelola akses secara terpusat bagi pengguna tenaga kerja Anda ke Akun AWS dan aplikasi. Anda mengaktifkan Pusat Identitas untuk organisasi Anda dan mengonfigurasi sumber identitas Anda. IAM Identity Center

menyediakan direktori sumber identitas default yang dapat Anda gunakan untuk mengelola pengguna dan grup Anda. Atau, Anda dapat memilih sumber identitas eksternal dengan terhubung ke penyedia identitas eksternal Anda menggunakan SAFL 2.0 dan secara otomatis menyediakan pengguna dan grup menggunakan SCIM, atau terhubung ke Direktori Microsoft AD Anda menggunakan Directory Service. Setelah sumber identitas dikonfigurasi, Anda dapat menetapkan akses kepada pengguna dan grup ke Akun AWS dengan menentukan kebijakan hak akses paling rendah di seperangkat izin. Pengguna tenaga kerja Anda dapat melakukan autentikasi melalui penyedia identitas pusat Anda untuk masuk ke portal akses AWS dan melakukan masuk tunggal ke Akun AWS dan aplikasi cloud yang ditetapkan untuk mereka. Pengguna Anda dapat mengonfigurasi AWS CLI v2 untuk mengautentikasi dengan Pusat Identitas dan mendapatkan kredensial untuk menjalankan perintah AWS CLI. Pusat Identitas juga memungkinkan akses masuk tunggal ke aplikasi AWS seperti Amazon SageMaker AI Studio dan portal AWS IoT Sitewise Monitor.

Setelah Anda mengikuti panduan di atas, pengguna tenaga kerja Anda tidak perlu lagi menggunakan IAM users dan grup IAM untuk operasi normal saat mengelola beban kerja di AWS. Sebaliknya, pengguna dan grup Anda dikelola di luar AWS dan pengguna dapat mengakses sumber daya AWS sebagai identitas gabungan. Identitas gabungan menggunakan grup yang ditentukan oleh penyedia identitas terpusat Anda. Anda harus mengidentifikasi dan menghapus grup IAM, IAM users, dan kredensial pengguna jangka panjang (kata sandi dan kunci akses) yang sudah tidak diperlukan di situs Akun AWS Anda. Anda dapat menemukan kredensial yang tidak digunakan menggunakan laporan kredensial IAM, menghapus IAM users terkait, dan menghapus grup IAM. Anda dapat menerapkan Kebijakan Kontrol Layanan (SCP) ke organisasi Anda yang membantu mencegah pembuatan IAM users dan grup IAM baru, sehingga memaksa akses ke AWS hanya terjadi melalui identitas gabungan.

Panduan untuk pengguna aplikasi Anda

Anda dapat mengelola identitas pengguna aplikasi Anda, seperti aplikasi seluler, menggunakan Amazon Cognito sebagai penyedia identitas terpusat Anda. Amazon Cognito memungkinkan autentikasi, otorisasi, dan manajemen pengguna untuk web dan aplikasi seluler Anda. Amazon Cognito menyediakan tempat penyimpanan identitas yang menyesuaikan skala dengan jutaan pengguna, mendukung federasi identitas sosial dan korporasi, serta menawarkan fitur keamanan canggih untuk membantu melindungi pengguna dan bisnis Anda. Anda dapat mengintegrasikan aplikasi web atau seluler kustom Anda dengan Amazon Cognito untuk menambahkan autentikasi pengguna dan kontrol akses ke aplikasi Anda dalam hitungan menit. Dibangun di atas standar

identitas terbuka seperti SAFL dan Open ID Connect (OIDC), Amazon Cognito mendukung berbagai peraturan kepatuhan dan terintegrasi dengan sumber daya pengembangan frontend dan backend.

Langkah implementasi

Langkah-langkah untuk pengguna tenaga kerja yang mengakses AWS

- Gabungkan pengguna tenaga kerja Anda ke AWS menggunakan penyedia identitas terpusat melalui salah satu pendekatan berikut:
  - Gunakan IAM Identity Center untuk mengaktifkan masuk tunggal ke beberapa Akun AWS di organisasi AWS Anda dengan cara menggabungkan dengan penyedia identitas Anda.
  - Gunakan IAM untuk menghubungkan penyedia identitas Anda secara langsung ke setiap Akun AWS, sehingga memungkinkan akses mendetail gabungan.
- Identifikasikan dan hapus IAM users dan grup IAM yang digantikan dengan identitas gabungan.

Langkah-langkah untuk pengguna aplikasi Anda

- Gunakan Amazon Cognito sebagai penyedia identitas terpusat menuju aplikasi Anda.
- Integrasikan aplikasi kustom Anda dengan Amazon Cognito menggunakan OpenID Connect dan OAuth. Anda dapat mengembangkan aplikasi kustom menggunakan pustaka Amplify yang menyediakan antarmuka sederhana untuk diintegrasikan dengan berbagai layanan AWS, seperti Amazon Cognito untuk autentikasi.

Sumber daya

Praktik terbaik Well-Architected terkait:

- SEC02-BP06 Manfaatkan grup dan atribut pengguna
- SEC03-BP02 Memberikan hak akses paling rendah
- SEC03-BP06 Mengelola akses berdasarkan siklus hidup

Dokumen terkait:

- Federasi identitas di AWS
- Praktik terbaik keamanan di IAM
- Praktik terbaik AWS Identity and Access Management

- Memulai dengan administrasi terdelegasi IAM Identity Center
- Cara menggunakan kebijakan yang dikelola pelanggan di IAM Identity Center untuk kasus penggunaan lanjutan
- AWS CLI v2: penyedia kredensial IAM Identity Center

Video terkait:

- AWS re:Inforce 2022 - Pembahasan mendalam AWS Identity and Access Management (IAM)
- AWS re:invent 2022 - Menyederhanakan akses tenaga kerja Anda dengan IAM Identity Center
- AWS re:Invent 2018: Menguasai Identitas di Setiap Lapisan Susunan

Contoh terkait:

- Lokakarya: Menggunakan AWS IAM Identity Center untuk mencapai manajemen identitas yang kuat
- Lokakarya: Identitas nirserver

Alat terkait:

- Partner Kompetensi Keamanan AWS: Manajemen Identitas dan Akses
- saml2aws

SEC02-BP05 Mengaudit dan merotasi kredensial secara berkala

Audit dan rotasikan kredensial secara berkala guna membatasi seberapa lama kredensial dapat digunakan untuk mengakses sumber daya Anda. Kredensial jangka panjang menimbulkan banyak risiko, tetapi risiko ini dapat dikurangi dengan merotasikan kredensial jangka panjang secara berkala.

Hasil yang diinginkan: Mengimplementasikan rotasi kredensial untuk mengurangi risiko terkait penggunaan kredensial jangka panjang. Melakukan audit dan perbaikan secara rutin untuk penggunaan yang tidak mematuhi kebijakan rotasi kredensial.

Antipola umum:

- Tidak mengaudit penggunaan kredensial.
- Menggunakan kredensial jangka panjang saat tidak diperlukan.

- Menggunakan kredensial jangka panjang dan tidak merotasinya secara rutin.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Jika Anda tidak dapat mengandalkan kredensial sementara dan memerlukan kredensial jangka panjang, lakukan audit kredensial untuk memastikan bahwa kontrol yang ditentukan seperti autentikasi multi-faktor (MFA) telah diterapkan, dirotasi secara rutin, dan memiliki tingkat akses yang sesuai.

Validasi berkala, sebaiknya melalui alat otomatis, diperlukan untuk memverifikasi penerapan kontrol yang tepat. Untuk identitas manusia, Anda harus mewajibkan pengguna untuk mengubah kata sandi mereka secara rutin dan menonaktifkan kunci akses yang ditukar dengan kredensial sementara. Setelah Anda beralih dari pengguna AWS Identity and Access Management (IAM) ke pengguna identitas terpusat, Anda dapat membuat laporan kredensial untuk mengaudit pengguna Anda.

Anda juga sebaiknya menerapkan dan memantau MFA dalam penyedia identitas Anda. Anda dapat mengonfigurasikan Aturan AWS Config, atau menggunakan Standar Keamanan AWS Security Hub CSPM, untuk memantau apakah pengguna mengaktifkan MFA. Pertimbangkan untuk menggunakan IAM Roles Anywhere guna memberikan kredensial sementara untuk identitas mesin. Dalam situasi yang tidak memungkinkan penggunaan peran IAM dan kredensial sementara, pengauditan dan rotasi kunci akses perlu sering dilakukan.

Langkah implementasi

- Audit kredensial secara rutin: Mengaudit identitas yang dikonfigurasikan dalam penyedia identitas dan IAM Anda membantu memastikan bahwa hanya identitas yang diotorisasi yang memiliki akses ke beban kerja Anda. Identitas tersebut mencakup, tetapi tidak terbatas pada, pengguna IAM, pengguna AWS IAM Identity Center, pengguna Active Directory, atau pengguna dalam penyedia identitas hulu yang berbeda. Misalnya, hapus orang yang keluar dari organisasi, serta hapus peran lintas akun yang sudah tidak diperlukan. Terapkan proses untuk secara berkala mengaudit izin ke layanan yang diakses oleh entitas IAM. Tindakan ini akan membantu Anda mengidentifikasi kebijakan yang perlu diubah untuk menghapus izin yang tidak digunakan. Gunakan laporan kredensial dan AWS Identity and Access Management Access Analyzer untuk mengaudit izin dan kredensial IAM. Anda dapat menggunakan Amazon CloudWatch untuk mengonfigurasi alarm untuk panggilan API tertentu dalam lingkungan AWS Anda. Amazon GuardDuty juga dapat memberikan peringatan terkait aktivitas yang tidak diharapkan, yang mungkin menandakan akses yang terlalu permisif atau akses yang tidak diinginkan ke kredensial IAM.

- Lakukan rotasi kredensial secara rutin: Ketika Anda tidak dapat menggunakan kredensial sementara, rotasikan kunci akses IAM jangka panjang secara rutin (maksimum 90 hari sekali). Tindakan ini akan membatasi waktu penggunaan kredensial untuk mengakses sumber daya Anda jika kunci akses bocor tanpa sepengetahuan Anda. Untuk informasi tentang merotasi kunci akses bagi pengguna IAM, lihat Merotasi kunci akses.
- Tinjau izin IAM: Untuk meningkatkan keamanan Akun AWS Anda, tinjau dan pantau setiap kebijakan IAM Anda secara rutin. Pastikan kebijakan tersebut memenuhi prinsip hak akses paling rendah.
- Pertimbangkan untuk mengotomatiskan pembaruan dan pembuatan sumber daya IAM: IAM Identity Center mengotomatiskan banyak tugas IAM, seperti manajemen kebijakan dan peran. Atau, AWS CloudFormation dapat digunakan untuk mengotomatiskan deployment sumber daya IAM, termasuk kebijakan dan peran, untuk mengurangi kemungkinan kesalahan akibat kelalaian manusia karena templat dapat diverifikasi serta dikelola dengan kendali versi.
- Gunakan IAM Roles Anywhere untuk mengganti pengguna IAM untuk identitas mesin: IAM Roles Anywhere memungkinkan Anda untuk menggunakan peran dalam area yang secara tradisional tidak bisa digunakan, seperti server on-premise. IAM Roles Anywhere menggunakan sertifikat X.509 tepercaya untuk mengautentikasi ke AWS serta menerima kredensial sementara. Dengan IAM Roles Anywhere, Anda tidak perlu merotasi kredensial ini karena kredensial jangka panjang tidak lagi disimpan dalam lingkungan on-premise Anda. Perlu diketahui bahwa Anda harus memantau dan merotasi sertifikat X.509 sebelum kedaluwarsa.

Sumber daya

Praktik Terbaik Terkait:

- SEC02-BP02 Menggunakan kredensial sementara
- SEC02-BP03 Menyimpan dan menggunakan rahasia secara aman

Dokumen terkait:

- Mulai menggunakan AWS Secrets Manager
- Praktik Terbaik IAM
- Penyedia Identitas dan Federasi
- Solusi Partner Keamanan: Akses dan Kontrol Akses
- Kredensial Keamanan Sementara

- Mendapatkan laporan kredensial untuk Akun AWS Anda

Video terkait:

- Praktik Terbaik untuk Mengelola, Mengambil, dan Merotasi Rahasia dalam Skala Besar
- Mengelola izin pengguna dalam skala besar dengan AWS IAM Identity Center
- Menguasai identitas di setiap lapisan susunan

Contoh terkait:

- Lab Well-Architected - Pembersihan Pengguna IAM Otomatis
- Lab Well-Architected Deployment Otomatis Peran dan Grup IAM

SEC02-BP06 Manfaatkan grup dan atribut pengguna

Seiring meningkatnya jumlah pengguna yang dikelola, Anda perlu menentukan cara agar dapat mengelolanya dalam skala besar. Tempatkan pengguna yang memiliki persyaratan keamanan yang sama dalam grup yang ditentukan oleh penyedia identitas Anda, dan terapkan mekanisme untuk memastikan atribut pengguna yang dapat digunakan untuk kontrol akses (misalnya departemen atau lokasi) sudah benar dan diperbarui. Gunakan grup dan atribut tersebut untuk mengontrol akses, bukan pengguna individual. Dengan demikian, Anda dapat mengelola akses secara terpusat cukup dengan satu kali mengubah keanggotaan atau atribut grup pengguna dengan seperangkat izin, daripada memperbarui banyak kebijakan satu per satu saat akses pengguna perlu diubah. Anda dapat menggunakan AWS IAM Identity Center (IAM Identity Center) untuk mengelola grup dan atribut pengguna. IAM Identity Center mendukung atribut yang paling sering digunakan, baik dimasukkan secara manual selama pembuatan pengguna atau disediakan secara otomatis menggunakan mesin sinkronisasi, seperti yang ditetapkan dalam spesifikasi Sistem untuk Manajemen Identitas Lintas Domain (SCIM).

Tempatkan pengguna yang memiliki persyaratan keamanan yang sama dalam grup yang ditentukan oleh penyedia identitas Anda, dan terapkan mekanisme untuk memastikan atribut pengguna yang dapat digunakan untuk kontrol akses (misalnya departemen atau lokasi) sudah benar dan diperbarui. Gunakan grup dan atribut tersebut, bukan pengguna individual, untuk mengontrol akses. Dengan demikian, Anda dapat mengelola akses secara terpusat cukup dengan satu kali mengubah keanggotaan atau atribut grup pengguna, daripada memperbarui banyak kebijakan satu per satu saat akses pengguna perlu diubah.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Rendah

Panduan implementasi

- Jika Anda menggunakan AWS IAM Identity Center (IAM Identity Center), konfigurasikan grup: IAM Identity Center memberikan kemampuan untuk mengonfigurasi grup pengguna dan menetapkan grup untuk tingkat izin yang diinginkan.
  - AWS Masuk Tunggal - Kelola Identitas
- Pelajari lebih lanjut tentang kontrol akses berbasis atribut (ABAC): ABAC adalah strategi otorisasi yang menetapkan izin berdasarkan atribut.
  - Apa Itu ABAC untuk AWS?
  - Lab: Kontrol Akses Berbasis Tanda IAM untuk EC2

Sumber daya

Dokumen terkait:

- Mulai Menggunakan AWS Secrets Manager
- Praktik Terbaik IAM
- Penyedia Identitas dan Federasi
- Pengguna Root Akun AWS

Video terkait:

- Praktik Terbaik untuk Mengelola, Mengambil, dan Merotasi Secret dalam Skala Besar
- Mengelola izin pengguna dalam skala besar dengan AWS IAM Identity Center
- Menguasai identitas di setiap lapisan beban kerja

Contoh terkait:

- Lab: Kontrol Akses Berbasis Tanda IAM untuk EC2

SEC 3. Bagaimana cara mengelola izin untuk manusia dan mesin?

Kelola izin untuk mengontrol akses ke identitas manusia dan identitas mesin yang memerlukan akses ke AWS dan beban kerja Anda. Izin akan mengontrol siapa yang dapat mengakses hal tertentu, beserta kondisinya.

Praktik terbaik

- SEC03-BP01 Menetapkan persyaratan akses
- SEC03-BP02 Memberikan hak akses paling rendah
- SEC03-BP03 Menerapkan proses akses darurat
- SEC03-BP04 Mengurangi izin secara terus-menerus
- SEC03-BP05 Menentukan pagar pembatas izin untuk organisasi Anda
- SEC03-BP06 Mengelola akses berdasarkan siklus hidup
- SEC03-BP07 Menganalisis akses lintas akun dan publik
- SEC03-BP08 Membagikan sumber daya secara aman dalam organisasi Anda
- SEC03-BP09 Membagikan sumber daya secara aman kepada pihak ketiga

SEC03-BP01 Menetapkan persyaratan akses

Tiap-tiap komponen atau sumber daya beban kerja Anda perlu diakses oleh administrator, pengguna akhir, atau komponen lainnya. Miliki penetapan yang jelas tentang siapa atau apa yang harus memiliki akses ke tiap-tiap komponen, pilih tipe identitas dan metode autentikasi serta otorisasi yang tepat.

Antipola umum:

- Hard-coding atau menyimpan rahasia di dalam aplikasi Anda.
- Memberikan izin kustom untuk tiap pengguna.
- Menggunakan kredensial berumur panjang.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Tiap-tiap komponen atau sumber daya beban kerja Anda perlu diakses oleh administrator, pengguna akhir, atau komponen lainnya. Miliki penetapan yang jelas tentang siapa atau apa yang harus

memiliki akses ke tiap-tiap komponen, pilih tipe identitas dan metode autentikasi serta otorisasi yang tepat.

Akses rutin ke Akun AWS di dalam organisasi harus disediakan menggunakan akses gabungan atau penyedia identitas terpusat. Anda juga sebaiknya memusatkan manajemen identitas Anda dan memastikan terdapat praktik yang matang untuk mengintegrasikan akses AWS ke siklus hidup akses karyawan Anda. Misalnya, saat seorang karyawan berganti peran pekerjaan dengan level akses berbeda, keanggotaan grupnya juga harus berubah agar sesuai dengan persyaratan akses barunya.

Saat menetapkan persyaratan akses untuk identitas non-manusia, tentukan aplikasi dan komponen mana yang memerlukan akses dan bagaimana izin diberikan. Menggunakan IAM role yang dibangun dengan model akses hak akses paling rendah adalah pendekatan yang disarankan. Kebijakan yang Dikelola AWS menyediakan kebijakan IAM yang telah ditetapkan sebelumnya yang mencakup kasus-kasus penggunaan paling umum.

Layanan AWS, seperti AWS Secrets Manager dan AWS Systems Manager Parameter Store, dan membantu memisahkan rahasia dari aplikasi atau beban kerja secara aman pada kasus-kasus yang tidak memungkinkan penggunaan IAM role. Di Secrets Manager, Anda dapat membuat rotasi otomatis untuk kredensial Anda. Anda dapat menggunakan Systems Manager untuk merujuk parameter di skrip, perintah, dokumen SSM, konfigurasi, dan alur kerja otomatisasi Anda menggunakan nama unik yang telah Anda tentukan saat membuat parameter tersebut.

Anda dapat menggunakan AWS Identity and Access Management Roles Anywhere untuk mendapatkan kredensial keamanan sementara di IAM untuk beban kerja yang berjalan di luar AWS. Beban kerja Anda dapat menggunakan kebijakan IAM dan IAM role yang sama dengan yang Anda gunakan dengan aplikasi AWS untuk mengakses sumber daya AWS.

Jika memungkinkan, gunakan kredensial sementara jangka pendek, bukan kredensial statis jangka panjang. Untuk skenario di mana Anda memerlukan pengguna IAM dengan akses terprogram dan kredensial jangka panjang, gunakan informasi yang terakhir digunakan kunci akses untuk merotasi dan menghapus kunci akses.

Sumber daya

Dokumen terkait:

- Kontrol akses berbasis atribut (ABAC)
- AWS IAM Identity Center
- IAM Roles Anywhere

- Kebijakan yang dikelola AWS untuk IAM Identity Center
- Ketentuan kebijakan AWS IAM
- Kasus penggunaan IAM
- Hapus kredensial yang tidak diperlukan
- Bekerja dengan Kebijakan
- Cara mengontrol akses ke sumber daya AWS berdasarkan Akun AWS, OU, atau organisasi
- Identifikasi, atur, dan kelola rahasia secara mudah menggunakan pencarian yang ditingkatkan di AWS Secrets Manager

Video terkait:

- Menjadi Master Kebijakan IAM dalam 60 Menit atau Kurang
- Pemisahan Tugas, Hak Akses Paling Rendah, Delegasi, dan CI/CD
- Merampingkan manajemen identitas dan akses untuk inovasi

SEC03-BP02 Memberikan hak akses paling rendah

Salah satu praktik terbaik adalah memberikan hanya akses yang diperlukan identitas untuk melakukan tindakan tertentu pada sumber daya tertentu dalam kondisi tertentu. Gunakan atribut grup dan identitas untuk menetapkan izin secara dinamis dalam skala besar daripada menentukan izin satu per satu untuk setiap pengguna. Misalnya, Anda dapat memberikan akses kepada sebuah grup developer untuk mengelola sumber daya untuk proyek mereka saja. Dengan cara ini, jika seorang developer keluar dari proyek, akses developer tersebut secara otomatis dicabut tanpa mengubah kebijakan akses dasar.

Hasil yang diinginkan: Pengguna hanya memiliki izin yang diperlukan untuk melakukan pekerjaannya. Pengguna hanya diberi akses ke lingkungan produksi untuk melakukan tugas tertentu dalam jangka waktu terbatas dan akses harus dicabut setelah tugas tersebut selesai. Izin harus dicabut jika sudah tidak digunakan, termasuk saat pengguna beralih ke proyek atau jabatan kerja lain. Hak akses administrator hanya boleh diberikan kepada sekelompok kecil administrator yang tepercaya. Izin harus ditinjau secara rutin untuk menghindari creep izin. Akun sistem atau mesin hanya boleh diberi rangkaian izin paling sedikit yang diperlukan untuk menyelesaikan tugas mereka.

Antipola umum:

- Memberikan izin administrator kepada para pengguna secara default.

- Menggunakan pengguna root untuk aktivitas harian.
- Membuat kebijakan yang terlalu permisif, tetapi tanpa hak istimewa administrator penuh.
- Tidak meninjau izin untuk mengetahui apakah izin tersebut memberikan hak akses paling rendah.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Prinsip hak akses paling rendah menyatakan bahwa identitas hanya boleh diizinkan untuk melakukan rangkaian tindakan sekecil mungkin yang diperlukan untuk menyelesaikan tugas tertentu. Hal ini menyeimbangkan kegunaan, efisiensi, dan keamanan. Pengoperasian berdasarkan prinsip ini membantu membatasi akses yang tidak diinginkan dan membantu memantau siapa saja yang memiliki akses ke sumber daya yang mana. Pengguna dan peran IAM secara default tidak memiliki izin apa pun. Pengguna root memiliki akses penuh secara default dan harus secara ketat dikontrol, dimonitor, dan digunakan hanya untuk tugas yang memerlukan akses root.

Kebijakan IAM digunakan untuk memberikan izin secara eksplisit ke peran IAM atau sumber daya tertentu. Contohnya, kebijakan berbasis identitas dapat dilampirkan ke grup IAM, sedangkan bucket S3 dapat dikontrol oleh kebijakan berbasis sumber daya.

Saat membuat kebijakan IAM, Anda dapat menentukan tindakan layanan, sumber daya, dan kondisi yang harus terpenuhi agar AWS dapat memberikan atau menolak akses. AWS mendukung beragam kondisi untuk membantu Anda menyaring akses. Contohnya, dengan menggunakan kunci kondisi `PrincipalOrgID`, Anda dapat menolak tindakan jika pemohon bukan bagian dari Organisasi AWS Anda.

Anda juga dapat mengontrol permintaan yang dibuat oleh layanan AWS atas nama Anda, seperti AWS CloudFormation yang membuat fungsi AWS Lambda, dengan menggunakan kunci kondisi `CalledVia`. Anda sebaiknya menggunakan berbagai macam kebijakan secara berlapis untuk membuat sistem pertahanan yang mendalam dan membatasi izin keseluruhan untuk pengguna Anda. Anda juga bisa membatasi izin yang dapat diberikan beserta kondisinya. Misalnya, Anda dapat mengizinkan tim aplikasi membuat kebijakan IAM sendiri untuk sistem yang mereka buat, tetapi Anda juga harus menerapkan Batasan Izin untuk membatasi izin maksimum yang bisa didapatkan sistem.

Langkah implementasi

- Implementasikan kebijakan hak akses paling rendah: Tetapkan kebijakan akses dengan hak paling rendah ke grup dan peran IAM untuk mencerminkan peran atau fungsi pengguna yang telah Anda tetapkan.

  - Kebijakan dasar untuk penggunaan API: Salah satu cara untuk menentukan izin yang diperlukan adalah dengan meninjau log AWS CloudTrail. Dengan peninjauan ini, Anda dapat membuat izin yang disesuaikan dengan tindakan yang benar-benar dilakukan oleh pengguna di dalam AWS. IAM Access Analyzer dapat menghasilkan kebijakan IAM secara otomatis berdasarkan aktivitas. Anda dapat menggunakan Penasihat Akses IAM di tingkat organisasi atau akun guna melacak informasi yang terakhir diakses untuk kebijakan tertentu.
- Pertimbangkan penggunaan kebijakan terkelola AWS untuk fungsi tugas. Saat akan membuat kebijakan izin yang disesuaikan secara mendetail, mungkin sulit untuk mengetahui cara memulainya. AWS memiliki kebijakan terkelola untuk peran tugas umum, misalnya penagihan, administrator basis data, dan ilmuwan data. Kebijakan ini dapat membantu mempersempit akses yang dimiliki pengguna selagi menentukan cara menerapkan kebijakan hak akses paling rendah.
- Hapus izin yang tidak diperlukan: Hapus izin yang tidak diperlukan dan ketatkan kebijakan yang terlalu longgar. Pembuatan kebijakan IAM Access Analyzer dapat membantu menyaring kebijakan izin.
- Pastikan pengguna memiliki akses yang terbatas ke lingkungan produksi: Pengguna seharusnya hanya memiliki akses ke lingkungan produksi dengan kasus penggunaan yang valid. Setelah pengguna menyelesaikan tugas tertentu yang memerlukan akses produksi, akses harus dicabut. Pembatasan akses ke lingkungan produksi membantu mencegah kejadian tak terduga yang memengaruhi produksi dan memperkecil cakupan dampak akses yang tidak diharapkan.
- Pertimbangkan batasan izin: Batasan izin adalah fitur untuk menggunakan kebijakan terkelola yang menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas ke entitas IAM. Batasan izin memungkinkan entitas melakukan tindakan hanya yang diizinkan oleh kebijakan berbasis identitas serta batasan izinnya.
- Pertimbangkan tanda sumber daya untuk izin: Model kontrol akses berbasis atribut yang menggunakan tanda sumber daya memungkinkan Anda memberikan akses berdasarkan tujuan sumber daya, pemilik, lingkungan, atau kriteria lain. Misalnya, Anda dapat menggunakan tanda sumber daya untuk membedakan lingkungan pengembangan dan produksi. Dengan tanda ini, Anda dapat membatasi developer agar hanya dapat mengakses lingkungan pengembangan. Dengan memadukan kebijakan pemberian tanda dan izin, Anda dapat memiliki akses sumber daya yang terperinci tanpa harus menentukan kebijakan kustom dan rumit untuk setiap fungsi tugas.
- Gunakan kebijakan kontrol layanan untuk AWS Organizations. Kebijakan kontrol layanan secara terpusat mengontrol izin maksimum yang tersedia bagi akun anggota di organisasi Anda. Fungsi penting dari kebijakan kontrol layanan adalah memungkinkan Anda membatasi izin pengguna root di dalam akun anggota. Pertimbangkan juga untuk menggunakan AWS Control Tower, yang

menyediakan kontrol terkelola preskriptif yang makin melengkapi AWS Organizations. Anda juga dapat menentukan kontrol sendiri di dalam Control Tower.

- Buat kebijakan siklus hidup pengguna untuk organisasi Anda: Kebijakan siklus hidup pengguna menentukan tugas yang harus dilakukan saat pengguna ditambahkan ke AWS, mengubah peran atau cakupan pekerjaan, atau sudah tidak memerlukan akses ke AWS. Peninjauan izin harus dilakukan selama setiap langkah dalam siklus hidup pengguna untuk memverifikasi bahwa izin dibatasi dengan sesuai dan untuk menghindari creep izin.
- Buat jadwal rutin untuk meninjau izin dan menghapus izin yang tidak diperlukan: Anda harus rutin meninjau akses pengguna untuk memverifikasi bahwa pengguna tidak memiliki akses yang terlalu leluasa. AWS Config dan IAM Access Analyzer dapat membantu Anda saat mengaudit izin pengguna.
- Buat matriks peran kerja: Matrik peran kerja memberikan visualiasi untuk berbagai peran dan tingkat akses yang diperlukan dalam jejak AWS Anda. Dengan matriks peran kerja, Anda dapat menentukan dan memisahkan izin berdasarkan tanggung jawab pengguna di dalam organisasi. Gunakan grup daripada menerapkan izin secara langsung ke pengguna atau peran satu per satu.

Sumber Daya

Dokumen terkait:

- Berikan hak akses paling rendah
- Batasan izin untuk entitas IAM
- Teknik untuk menulis kebijakan IAM hak akses paling rendah
- IAM Access Analyzer mempermudah implementasi izin hak akses paling rendah dengan menghasilkan kebijakan IAM berdasarkan aktivitas akses
- Delegasikan manajemen izin ke developer menggunakan batasan izin IAM
- Mempersempit Izin menggunakan informasi yang terakhir kali diakses
- Jenis kebijakan IAM dan kapan harus digunakan
- Menguji kebijakan IAM dengan simulator kebijakan IAM
- Pagar Pembatas di AWS Control Tower
- Arsitektur Zero Trust: Sebuah perspektif AWS
- Cara mengimplementasikan prinsip hak akses paling rendah dengan CloudFormation StackSets
- Kontrol akses berbasis atribut (ABAC)
- Mengurangi cakupan kebijakan dengan melihat aktivitas pengguna

- Lihat akses peran
- Gunakan Penandaan untuk Mengelola Lingkungan Anda dan Meningkatkan Akuntabilitas
- Strategi Penandaan AWS
- Penandaan sumber daya AWS

Video terkait:

- Manajemen izin generasi baru
- Zero Trust: Sebuah perspektif AWS
- Bagaimana cara menggunakan batasan izin untuk membatasi pengguna dan peran guna mencegah eskalasi hak akses?

Contoh terkait:

- Lab: Batasan izin IAM yang mendelegasikan pembuatan peran
- Lab: Kontrol akses berbasis tanda IAM untuk EC2

SEC03-BP03 Menerapkan proses akses darurat

Buat proses yang memungkinkan akses darurat ke beban kerja Anda jika terjadi masalah pada penyedia identitas terpusat Anda.

Anda harus merancang proses untuk berbagai mode kegagalan yang dapat mengakibatkan peristiwa darurat. Misalnya, dalam keadaan normal, pengguna tenaga kerja Anda melakukan federasi ke cloud menggunakan penyedia identitas terpusat (SEC02-BP04) untuk mengelola beban kerja mereka. Namun, jika penyedia identitas terpusat Anda gagal, atau konfigurasi untuk federasi di cloud diubah, maka pengguna tenaga kerja Anda mungkin tidak dapat melakukan federasi ke cloud. Proses akses darurat memungkinkan administrator yang berwenang untuk mengakses sumber daya cloud Anda melalui cara alternatif (seperti bentuk federasi alternatif atau akses pengguna langsung) untuk memperbaiki masalah dengan konfigurasi federasi atau beban kerja Anda. Proses akses darurat digunakan sampai mekanisme federasi normal dipulihkan.

Hasil yang diinginkan:

- Anda telah menentukan dan mendokumentasikan mode kegagalan yang terhitung sebagai keadaan darurat: pertimbangkan keadaan normal Anda dan sistem yang diandalkan oleh

pengguna Anda untuk mengelola beban kerja mereka. Pertimbangkan bagaimana setiap dependensi ini dapat gagal dan menyebabkan keadaan darurat. Anda dapat menemukan pertanyaan dan praktik terbaik di Pilar Keandalan yang berguna untuk mengidentifikasi mode kegagalan dan merancang sistem yang lebih tangguh untuk meminimalkan kemungkinan kegagalan.

- Anda telah mendokumentasikan langkah-langkah yang harus diikuti untuk mengonfirmasi kegagalan sebagai keadaan darurat. Misalnya, Anda dapat meminta administrator identitas Anda untuk memeriksa status penyedia identitas utama dan siaga Anda dan, jika keduanya tidak tersedia, umumkan peristiwa darurat untuk kegagalan penyedia identitas.
- Anda telah menentukan proses akses darurat khusus untuk setiap jenis mode darurat atau kegagalan. Pengkhususan ini dapat mengurangi godaan di pihak pengguna Anda untuk terlalu sering menggunakan proses umum untuk semua jenis keadaan darurat. Proses akses darurat Anda menggambarkan keadaan di mana setiap proses harus digunakan dan, sebaliknya, situasi di mana proses tidak boleh digunakan dan menunjuk ke proses alternatif yang mungkin berlaku.
- Proses Anda didokumentasikan dengan baik dengan instruksi yang mendetail dan playbook yang dapat diikuti dengan cepat dan efisien. Ingatlah bahwa peristiwa darurat dapat menjadi saat yang memusingkan bagi pengguna Anda dan mereka sedang di bawah tekanan waktu yang ekstrem, jadi rancanglah proses Anda sesederhana mungkin.

Antipola umum:

- Anda tidak memiliki proses akses darurat yang terdokumentasi dengan baik dan teruji dengan baik. Pengguna Anda tidak siap menghadapi keadaan darurat dan mengikuti proses improvisasi ketika peristiwa darurat muncul.
- Proses akses darurat Anda bergantung pada sistem yang sama (seperti penyedia identitas terpusat) dengan mekanisme akses normal Anda. Ini artinya, kegagalan sistem tersebut dapat memengaruhi mekanisme akses normal dan darurat Anda dan mengganggu kemampuan Anda untuk pulih dari kegagalan.
- Proses akses darurat Anda digunakan dalam situasi non-darurat. Misalnya, pengguna Anda sering menyalahgunakan proses akses darurat karena mereka merasa lebih mudah melakukan perubahan secara langsung daripada mengirimkan perubahan melalui pipeline.
- Proses akses darurat Anda tidak menghasilkan log yang memadai untuk mengaudit proses, atau log tersebut tidak dipantau untuk mendapatkan peringatan potensi penyalahgunaan proses.

Manfaat menjalankan praktik terbaik ini:

- Dengan memiliki proses akses darurat yang terdokumentasi dengan baik dan teruji dengan baik, Anda dapat mengurangi waktu yang dibutuhkan pengguna untuk merespons dan menyelesaikan peristiwa darurat. Hal ini dapat menghasilkan lebih sedikit waktu henti dan ketersediaan yang lebih tinggi untuk layanan yang Anda berikan kepada pelanggan Anda.
- Anda dapat melacak setiap permintaan akses darurat dan mendeteksi serta memberikan peringatan adanya upaya penyalahgunaan proses untuk peristiwa non-darurat.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Bagian ini memberikan panduan dalam membuat proses akses darurat untuk beberapa mode kegagalan yang berkaitan dengan beban kerja yang di-deploy di AWS, dimulai dengan panduan umum yang berlaku untuk semua mode kegagalan dan dilanjutkan dengan panduan khusus berdasarkan jenis mode kegagalan.

Panduan umum untuk semua mode kegagalan

Pertimbangkan hal berikut saat Anda merancang proses akses darurat untuk mode kegagalan:

- Dokumentasikan kondisi awal dan asumsi tentang proses tersebut: kapan proses tersebut harus digunakan dan kapan proses tersebut tidak boleh digunakan. Penting untuk memiliki detail mode kegagalan dan mendokumentasikan asumsi, seperti keadaan sistem terkait lainnya. Misalnya, proses untuk Mode Kegagalan 2 mengasumsikan bahwa penyedia identitas tersedia, tetapi konfigurasi di AWS dimodifikasi atau telah kedaluwarsa.
- Sejak awal, buat sumber daya yang dibutuhkan oleh proses akses darurat (SEC10-BP05). Misalnya, buat Akun AWS akses darurat di awal dengan IAM users dan peran IAM, dan peran IAM lintas akun di semua akun beban kerja. Hal ini memastikan bahwa semua sumber daya ini siap dan tersedia ketika peristiwa darurat terjadi. Dengan membuat sumber daya di awal, Anda tidak bergantung pada API AWS bidang kendali (yang digunakan untuk membuat dan memodifikasi sumber daya AWS) yang mungkin tidak tersedia dalam keadaan darurat. Selanjutnya, dengan membuat sumber daya IAM di awal, Anda tidak perlu memperhitungkan potensi penundaan disebabkan konsistensi akhir.
- Sertakan proses akses darurat sebagai bagian dari rencana manajemen insiden Anda (SEC10-BP02). Dokumentasikan bagaimana peristiwa darurat dilacak dan dikomunikasikan kepada orang lain di organisasi Anda seperti tim sejawat, pimpinan Anda, dan, jika ada, secara eksternal kepada pelanggan dan partner bisnis Anda.

- Tentukan proses permintaan akses darurat di sistem alur kerja permintaan layanan yang ada jika Anda memilikinya. Biasanya, sistem alur kerja semacam ini memungkinkan Anda membuat formulir penerimaan informasi untuk mengumpulkan informasi tentang permintaan, melacak permintaan melalui setiap tahap alur kerja, dan menambahkan langkah persetujuan otomatis dan manual. Hubungkan setiap permintaan dengan peristiwa darurat terkait yang dilacak dalam sistem manajemen insiden Anda. Dengan memiliki sistem yang seragam untuk akses darurat, Anda dapat melacak permintaan tersebut dalam sistem tunggal, menganalisis tren penggunaan, dan meningkatkan kualitas proses Anda.
- Pastikan proses akses darurat Anda hanya dapat dimulai oleh pengguna yang berwenang dan memerlukan persetujuan dari rekan sejawat atau manajemen pengguna yang sesuai. Proses persetujuan harus beroperasi secara efektif baik di dalam maupun di luar jam kerja. Tentukan bagaimana permintaan persetujuan mengizinkan pemberi persetujuan sekunder jika pemberi persetujuan utama tidak tersedia dan ditingkatkan ke rantai manajemen Anda hingga disetujui.
- Pastikan bahwa proses tersebut menghasilkan log dan peristiwa audit yang mendetail, baik untuk upaya yang berhasil maupun yang gagal untuk mendapatkan akses darurat. Pantau proses permintaan serta mekanisme akses darurat untuk mendeteksi penyalahgunaan atau akses yang tidak sah. Korelasikan aktivitas dengan peristiwa darurat yang sedang berlangsung dari sistem manajemen insiden Anda dan munculkan peringatan ketika tindakan terjadi di luar periode waktu yang diharapkan. Misalnya, Anda harus memantau dan memperingatkan aktivitas dalam Akun AWS akses darurat, karena akun tersebut tidak boleh digunakan dalam operasi normal.
- Uji proses akses darurat secara berkala untuk memverifikasi bahwa langkah-langkahnya jelas dan memberikan tingkat akses yang benar dengan cepat dan efisien. Proses akses darurat Anda harus diuji sebagai bagian dari simulasi respons insiden (SEC10-BP07) dan tes pemulihan bencana (REL13-BP03).

Mode Kegagalan 1: Penyedia identitas yang digunakan untuk federasi ke AWS tidak tersedia

Seperti yang dijelaskan dalam SEC02-BP04 Mengandalkan penyedia identitas terpusat, kami sarankan Anda mengandalkan penyedia identitas terpusat untuk memfederasi pengguna tenaga kerja Anda untuk memberikan akses ke Akun AWS. Anda dapat melakukan federasi ke beberapa Akun AWS di organisasi AWS Anda menggunakan IAM Identity Center, atau Anda dapat melakukan federasi ke Akun AWS secara terpisah menggunakan IAM. Dalam kedua kasus tersebut, pengguna tenaga kerja melakukan autentikasi dengan penyedia identitas terpusat Anda sebelum diarahkan ke titik akhir masuk AWS ke masuk tunggal.

Apabila penyedia identitas terpusat Anda tidak tersedia, pengguna tenaga kerja Anda tidak dapat melakukan federasi ke Akun AWS atau mengelola beban kerja mereka. Dalam peristiwa darurat ini, Anda dapat menyediakan proses akses darurat untuk sekelompok kecil administrator untuk mengakses Akun AWS untuk melakukan tugas-tugas penting yang tidak dapat ditunda sampai penyedia identitas terpusat Anda kembali aktif. Misalnya, penyedia identitas Anda tidak tersedia selama 4 jam dan selama periode tersebut Anda perlu mengubah batas atas grup Amazon EC2 Auto Scaling di sebuah akun Produksi untuk menangani lonjakan lalu lintas pelanggan yang tidak terduga. Administrator darurat Anda harus mengikuti proses akses darurat untuk mendapatkan akses ke Akun AWS khusus produksi dan membuat perubahan yang diperlukan.

Proses akses darurat tersebut bergantung pada Akun AWS akses darurat yang telah dibuat sebelumnya yang digunakan semata-mata untuk akses darurat dan memiliki sumber daya AWS (seperti peran IAM danIAM users) untuk mendukung proses akses darurat. Selama operasi normal, tidak ada yang boleh mengakses akun akses darurat tersebut dan Anda harus memantau dan memperingatkan penyalahgunaan akun ini (untuk lebih jelasnya, lihat bagian panduan umum sebelumnya).

Akun akses darurat memiliki peran IAM akses darurat dengan izin untuk mengambil peran lintas akun di dalam Akun AWS yang memerlukan akses darurat. Peran IAM ini telah dibuat sebelumnya dan dikonfigurasi dengan kebijakan kepercayaan yang mempercayai peran IAM akun darurat.

Proses akses darurat dapat menggunakan salah satu pendekatan berikut:

- Anda dapat membuat satu set IAM users di awal untuk administrator darurat Anda di dalam akun akses darurat dengan kata sandi yang kuat dan token MFA terkait. Set IAM users ini memiliki izin untuk mengambil peran IAM yang kemudian memungkinkan akses lintas akun ke Akun AWS tempat akses darurat diperlukan. Kami sarankan Anda membuat pengguna sesedikit mungkin dan menetapkan setiap pengguna ke satu administrator darurat. Selama keadaan darurat, pengguna administrator darurat masuk ke akun akses darurat menggunakan kata sandi dan kode token MFA mereka, beralih ke peran IAM akses darurat di dalam akun darurat, dan akhirnya beralih ke peran IAM akses darurat di akun beban kerja untuk melakukan tindakan perubahan darurat. Kelebihan pendekatan ini adalah setiap IAM user ditugaskan ke satu administrator darurat dan Anda dapat mengetahui pengguna mana yang masuk dengan meninjau peristiwa CloudTrail. Kelemahannya adalah Anda harus mempertahankan beberapa IAM users dengan kata sandi berumur panjang dan token MFA yang terkait.
- Anda dapat menggunakan pengguna root Akun AWS akses darurat untuk masuk ke akun akses darurat, mengambil peran IAM untuk akses darurat, dan mengambil peran lintas akun di akun beban kerja. Kami merekomendasikan pengaturan kata sandi yang kuat dan beberapa token

MFA untuk pengguna root. Kami juga menyarankan Anda menyimpan kata sandi dan token MFA di brankas kredensial korporasi aman yang memberlakukan autentikasi dan otorisasi yang kuat. Anda harus mengamankan kata sandi dan faktor pengaturan ulang token MFA: atur alamat email akun ke daftar distribusi email yang dipantau oleh administrator keamanan cloud Anda, dan nomor telepon akun ke nomor telepon bersama yang juga dipantau oleh administrator keamanan. Keunggulan pendekatan ini adalah ada satu set kredensial pengguna root untuk dikelola. Kelemahannya adalah karena ini merupakan pengguna bersama, beberapa administrator memiliki kemampuan untuk masuk sebagai pengguna root. Anda harus mengaudit peristiwa log brankas korporasi Anda untuk mengidentifikasi administrator mana yang menggunakan kata sandi pengguna root.

Mode Kegagalan 2: Konfigurasi penyedia identitas di AWS dimodifikasi atau telah kedaluwarsa

Agar pengguna tenaga kerja Anda dapat melakukan federasi ke Akun AWS, Anda dapat mengonfigurasi IAM Identity Center dengan penyedia identitas eksternal atau membuat Penyedia Identitas IAM (SEC02-BP04). Biasanya, Anda mengonfigurasinya dengan mengimpor dokumen XML metadata SAML yang disediakan oleh penyedia identitas Anda. Dokumen metadata XML tersebut mencakup sertifikat X.509 yang sesuai dengan kunci privat yang digunakan oleh penyedia identitas untuk menandatangani pernyataan SAML-nya.

Konfigurasi di sisi AWS ini dapat diubah atau dihapus secara tidak sengaja oleh administrator. Dalam skenario lain, sertifikat X.509 yang diimpor ke dalam AWS dapat kedaluwarsa dan XML metadata baru dengan sertifikat baru belum diimpor ke AWS. Kedua skenario ini dapat mengganggu federasi ke AWS untuk pengguna tenaga kerja Anda, yang mengakibatkan keadaan darurat.

Dalam keadaan darurat seperti ini, Anda dapat memberikan akses ke AWS kepada administrator identitas Anda untuk memperbaiki masalah federasi tersebut. Misalnya, administrator identitas Anda menggunakan proses akses darurat untuk masuk ke Akun AWS akses darurat, beralih ke peran di akun administrator Pusat Identitas, dan memperbarui konfigurasi penyedia identitas eksternal dengan mengimpor dokumen XML metadata SAML terbaru dari penyedia identitas Anda untuk mengaktifkan kembali federasi. Setelah federasi diperbaiki, pengguna tenaga kerja Anda melanjutkan penggunaan proses operasi normal untuk melakukan federasi ke akun beban kerja mereka.

Anda dapat mengikuti pendekatan yang dijelaskan dalam Mode Kegagalan 1 sebelumnya untuk membuat proses akses darurat. Anda dapat memberikan hak akses paling sedikit kepada administrator identitas Anda untuk mengakses hanya akun administrator Pusat Identitas dan melakukan tindakan pada Pusat Identitas di akun tersebut.

Mode Kegagalan 3: Gangguan Pusat Identitas

Apabila terjadi gangguan IAM Identity Center atau Wilayah AWS, kami sarankan Anda menyiapkan konfigurasi yang dapat Anda gunakan untuk menyediakan akses sementara ke Konsol Manajemen AWS.

Proses akses darurat tersebut menggunakan federasi langsung dari penyedia identitas Anda ke IAM dalam akun darurat. Untuk detail tentang proses dan pertimbangan desain, lihat Menyiapkan akses darurat ke Konsol Manajemen AWS.

Langkah implementasi

Langkah-langkah umum untuk semua mode kegagalan

- Buat Akun AWS yang ditujukan khusus untuk proses akses darurat. Di awal, buat sumber daya IAM yang dibutuhkan di dalam akun seperti peran IAM atau IAM users, dan Penyedia Identitas IAM opsional. Selain itu, buat di awal peran IAM lintas akun di dalam Akun AWS beban kerja dengan hubungan kepercayaan dengan IAM peran yang sesuai di akun akses darurat. Anda dapat menggunakan CloudFormation StackSets dengan AWS Organizations untuk membuat sumber daya tersebut di akun anggota di dalam organisasi Anda.

- Buat AWS Organizations kebijakan kontrol layanan (SCP) untuk menyangkal penghapusan dan modifikasi peran IAM lintas akun di Akun AWS anggota.
- Aktifkan CloudTrail untuk Akun AWS akses darurat dan kirimkan peristiwa jejak ke bucket S3 pusat di Akun AWS pengumpulan log Anda. Jika Anda menggunakan AWS Control Tower untuk menyiapkan dan mengatur lingkungan multiakun AWS Anda, maka setiap akun yang Anda buat menggunakan AWS Control Tower atau daftarkan di AWS Control Tower memiliki CloudTrail yang diaktifkan secara default dan dikirim ke bucket S3 dalam Akun AWS arsip log khusus.
- Pantau aktivitas di akun akses darurat dengan membuat aturan EventBridge yang cocok saat login konsol dan aktivitas API berdasarkan peran IAM darurat. Kirimkan notifikasi ke pusat operasi keamanan Anda ketika aktivitas terjadi di luar peristiwa darurat yang sedang berlangsung yang dilacak dalam sistem manajemen insiden Anda.

Langkah-langkah tambahan untuk Mode Kegagalan 1: Penyedia identitas yang digunakan untuk melakukan federasi ke AWS tidak tersedia dan Mode Kegagalan 2: Konfigurasi penyedia identitas di AWS dimodifikasi atau telah kedaluwarsa

- Buat sumber daya di awal tergantung mekanisme yang Anda pilih untuk akses darurat:

- Menggunakan IAM users buat IAM users di awal dengan kata sandi yang kuat serta perangkat MFA terkait.
- Menggunakan pengguna root akun darurat: konfigurasikan pengguna root dengan kata sandi yang kuat dan simpan kata sandi di dalam brankas kredensial korporasi Anda. Kaitkan beberapa perangkat MFA fisik dengan pengguna root dan simpan perangkat di lokasi yang dapat diakses dengan cepat oleh anggota tim administrator darurat Anda.

Langkah-langkah tambahan untuk Mode Kegagalan 3: Gangguan pusat identitas

- Seperti yang dijelaskan dalam Menyiapkan akses darurat ke Konsol Manajemen AWS, di Akun AWS akses darurat, buat sebuah Penyedia Identitas IAM untuk mengaktifkan federasi SAFL langsung dari penyedia identitas Anda.
- Buat grup operasi darurat di IdP Anda tanpa anggota.
- Buat peran IAM yang sesuai dengan grup operasi darurat di akun akses darurat.

Sumber daya

Praktik terbaik Well-Architected terkait:

- SEC02-BP04 Mengandalkan penyedia identitas terpusat
- SEC03-BP02 Memberikan hak akses paling rendah
- SEC10-BP02 Membuat rencana manajemen insiden
- SEC10-BP07 Menjalankan game day

Dokumen terkait:

- Menyiapkan akses darurat ke Konsol Manajemen AWS
- Mengaktifkan pengguna federasi SAFL 2.0 untuk mengakses Konsol Manajemen AWS
- Akses “pecah kaca”

Video terkait:

- AWS re:invent 2022 - Menyederhanakan akses tenaga kerja Anda dengan IAM Identity Center
- AWS re:Inforce 2022 - Pembahasan mendalam AWS Identity and Access Management (IAM)

Contoh terkait:

- Peran “Pecah Kaca” AWS
- Kerangka kerja playbook pelanggan AWS
- Contoh playbook respons insiden AWS

SEC03-BP04 Mengurangi izin secara terus-menerus

Jika tim Anda telah menentukan akses yang diperlukan, hapus izin yang tidak diperlukan dan tetapkan proses peninjauan untuk mendapatkan izin hak akses paling rendah. Pantau secara terus-menerus dan hapus identitas serta izin yang tidak diperlukan, baik untuk akses manusia maupun mesin.

Hasil yang diinginkan: Kebijakan izin harus mematuhi hak akses paling rendah. Setelah penetapan tugas dan peran pekerjaan sudah lebih baik, kebijakan izin Anda perlu ditinjau untuk menghapus izin yang tidak perlu. Pendekatan ini mempersempit cakupan dampak akibat kebocoran kredensial secara tidak sengaja, atau diakses tanpa otorisasi.

Antipola umum:

- Memberikan izin administrator kepada para pengguna secara default.
- Membuat kebijakan yang terlalu permisif, tetapi tanpa hak istimewa administrator penuh.
- Menyimpan kebijakan izin meski sudah tidak diperlukan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Setelah tim dan proyek mulai, kebijakan izin permisif mungkin digunakan untuk menumbuhkan inovasi dan ketangkasan. Misalnya, di dalam lingkungan pengembangan atau pengujian, developer dapat diberi akses ke seperangkat layanan AWS. Sebaiknya evaluasi akses secara terus-menerus dan batasi akses hanya untuk layanan dan tindakan layanan yang diperlukan untuk menyelesaikan tugas saat ini. Sebaiknya evaluasi ini dilakukan untuk identitas manusia maupun mesin. Identitas mesin, sering disebut sebagai akun layanan atau sistem, adalah identitas yang memberikan AWS akses ke aplikasi atau server. Akses ini penting terutama dalam lingkungan produksi, yang apabila izinnya terlalu permisif, dampaknya bisa luas dan berpotensi mengekspos data konsumen.

AWS menyediakan berbagai metode untuk membantu mengidentifikasi pengguna, peran, izin, dan kredensial yang tidak diperlukan. AWS juga dapat membantu menganalisis aktivitas akses oleh pengguna dan peran IAM, termasuk kunci akses terkait, dan akses ke sumber daya AWS, misalnya objek di bucket Amazon S3. Pembuatan kebijakan AWS Identity and Access Management Access Analyzer dapat membantu Anda menciptakan kebijakan pembatasan izin berdasarkan layanan dan tindakan aktual yang berinteraksi dengan pengguna utama. Kontrol akses berbasis atribut (ABAC) dapat membantu menyederhanakan manajemen izin. Dengan kontrol ini, Anda dapat memberikan izin kepada pengguna menggunakan atribut mereka tanpa perlu melampirkan kebijakan izin secara langsung ke setiap pengguna.

Langkah implementasi

- Gunakan AWS Identity and Access Management Access Analyzer: IAM Access Analyzer membantu mengidentifikasi sumber daya di organisasi dan akun Anda, seperti bucket Amazon Simple Storage Service (Amazon S3) atau peran IAM yang dibagikan kepada entitas eksternal.
- Gunakan pembuatan kebijakan IAM Access Analyzer: Pembuatan kebijakan IAM Access Analyzer membantu Anda membuat kebijakan izin terperinci berdasarkan aktivitas pengguna atau peran IAM.
- Tentukan rentang waktu yang diterima serta kebijakan penggunaan untuk pengguna dan peran IAM: Gunakan stempel waktu yang terakhir diakses untuk mengidentifikasi pengguna dan peran yang tidak perlu lalu hapus. Tinjau informasi layanan dan tindakan yang terakhir diakses untuk mengidentifikasi dan menentukan cakupan izin bagi pengguna dan peran tertentu. Misalnya, Anda dapat menggunakan informasi yang terakhir diakses untuk mengidentifikasi tindakan Amazon S3 tertentu yang diperlukan oleh peran aplikasi dan membatasi akses hanya untuk tindakan tersebut. Fitur informasi yang terakhir diakses tersedia di Konsol Manajemen AWS dan secara terprogram memungkinkan Anda menggabungkannya ke dalam alur kerja infrastruktur dan alat otomatis Anda.
- Pertimbangkan pencatatan log peristiwa data di AWS CloudTrail: Secara default, CloudTrail tidak mencatat log peristiwa data seperti aktivitas tingkat objek Amazon S3 (misalnya, `GetObject` dan `DeleteObject`) atau aktivitas tabel Amazon DynamoDB (misalnya, `PutItem` dan `DeleteItem`). Pertimbangkan untuk mengaktifkan pencatatan log pada peristiwa ini untuk menentukan pengguna dan peran apa yang perlu mengakses objek Amazon S3 dan item tabel DynamoDB tertentu.

Sumber daya

Dokumen terkait:

- Memberikan hak akses paling rendah

- Menghapus kredensial yang tidak diperlukan
- Apa itu AWS CloudTrail?
- Mengelola Kebijakan
- Pencatatan log dan pemantauan DynamoDB
- Mengaktifkan pencatatan log peristiwa CloudTrail untuk bucket dan objek Amazon S3
- Mendapatkan laporan kredensial untuk Akun AWS Anda

Video terkait:

- Menjadi Master Kebijakan IAM dalam 60 Menit atau Kurang
- Pemisahan Tugas, Hak Akses Paling Rendah, Delegasi, dan CI/CD
- AWS re:Inforce 2022 - Lebih dalam tentang AWS Identity and Access Management (IAM)

SEC03-BP05 Menentukan pagar pembatas izin untuk organisasi Anda

Tetapkan kontrol umum yang membatasi akses ke semua identitas di organisasi Anda. Misalnya, Anda dapat membatasi akses untuk Wilayah AWS tertentu, atau mencegah operator Anda menghapus dari sumber daya umum, seperti IAM role yang digunakan untuk tim keamanan pusat Anda.

Antipola umum:

- Menjalankan beban kerja di akun administrator Organisasi Anda.
- Menjalankan beban kerja produksi dan non-produksi di akun yang sama.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Seiring Anda menumbuhkan dan mengelola beban kerja tambahan di AWS, Anda harus memisahkan semua beban kerja ini menggunakan akun dan mengelola akun-akun tersebut menggunakan AWS Organizations. Kami menyarankan Anda membuat pagar pembatas izin umum yang membatasi akses ke semua identitas di organisasi Anda. Misalnya, Anda dapat membatasi akses ke Wilayah AWS tertentu, atau mencegah tim Anda menghapus sumber daya umum, seperti IAM role yang digunakan oleh tim keamanan pusat Anda.

Anda dapat memulainya dengan mengimplementasikan contoh kebijakan kontrol layanan, seperti mencegah pengguna menonaktifkan layanan utama. SCP menggunakan bahasa kebijakan IAM dan memungkinkan Anda untuk menerapkan kontrol yang dipatuhi semua principal (pengguna dan peran). Anda dapat membatasi akses ke tindakan atau sumber daya layanan tertentu, dan berdasarkan kondisi tertentu untuk memenuhi kebutuhan kontrol akses organisasi Anda. Jika perlu, Anda dapat menetapkan pengecualian pada pagar pembatas Anda. Misalnya, Anda dapat membatasi tindakan layanan untuk semua entitas IAM di dalam akun kecuali untuk peran administrator tertentu.

Kami menyarankan Anda untuk tidak menjalankan beban kerja di akun manajemen Anda. Akun manajemen sebaiknya digunakan untuk menata kelola dan men-deploy pagar pembatas keamanan yang akan memengaruhi akun-akun anggota. Beberapa layanan AWS mendukung penggunaan akun administrator yang didelegasikan. Saat tersedia, Anda harus menggunakan akun delegasi ini sebagai pengganti akun manajemen. Anda harus membatasi secara ketat akses ke akun administrator Organisasi.

Menggunakan strategi multi-akun memungkinkan Anda untuk memiliki fleksibilitas yang lebih besar dalam menerapkan pagar pembatas ke beban kerja Anda. Arsitektur Rujukan Keamanan AWS memberikan panduan preskriptif tentang cara merancang struktur akun Anda. Layanan AWS seperti AWS Control Tower menyediakan kemampuan untuk mengelola kontrol preventif serta detektif secara terpusat di seluruh organisasi Anda. Tetapkan tujuan yang jelas untuk tiap akun atau OU di dalam organisasi dan batasi kontrol yang sejalan dengan tujuan tersebut.

Sumber daya

Dokumen terkait:

- AWS Organizations
- Kebijakan kontrol layanan (SCP)
- Dapatkan hasil maksimal dari kebijakan kontrol layanan di lingkungan multi-akun
- Arsitektur Referensi Keamanan AWS (AWS SRA)

Video terkait:

- Tegakkan Pagar Pembatas Preventif menggunakan Kebijakan Kontrol Layanan
- Membangun tata kelola pada skala besar dengan AWS Control Tower
- Mendalami AWS Identity and Access Management

SEC03-BP06 Mengelola akses berdasarkan siklus hidup

Integrasikan kontrol akses dengan siklus hidup operator dan aplikasi serta penyedia federasi terpusat. Misalnya, hapus akses pengguna saat mereka keluar dari organisasi atau berganti peran.

Saat Anda mengelola beban kerja menggunakan akun terpisah, akan ada kasus saat Anda perlu membagikan sumber daya kepada akun-akun tersebut. Sebaiknya bagikan sumber menggunakan AWS Resource Access Manager (AWS RAM). Layanan ini memungkinkan Anda untuk membagikan sumber daya AWS di dalam Unit Organisasi dan AWS Organizations Anda. Menggunakan AWS RAM, akses ke sumber daya bersama secara otomatis diberikan atau dicabut ketika akun dimasukkan atau dikeluarkan dari Organisasi atau Unit Organisasi mereka. Hal ini memastikan bahwa sumber daya hanya dibagikan dengan akun yang Anda maksudkan saja.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Rendah

Panduan implementasi

Siklus hidup akses pengguna mengimplementasikan kebijakan siklus hidup akses pengguna untuk pengguna yang baru bergabung, perubahan fungsi tugas, serta pengguna yang keluar, sehingga hanya pengguna saat ini yang memiliki akses.

Sumber daya

Dokumen terkait:

- Kontrol akses berbasis atribut (ABAC)
- Berikan hak akses paling rendah
- IAM Access Analyzer
- Hapus kredensial yang tidak diperlukan
- Bekerja dengan Kebijakan

Video terkait:

- Become an IAM Policy Master in 60 Minutes or Less
- Separation of Duties, Least Privilege, Delegation, and CI/CD

SEC03-BP07 Menganalisis akses lintas akun dan publik

Pantau secara terus-menerus temuan yang menyoroti akses lintas akun dan publik. Kurangi akses publik dan akses lintas akun hanya ke sumber daya yang memerlukan akses ini.

Hasil yang diinginkan: Mengetahui mana sumber daya AWS yang dapat dibagikan dan kepada siapa. Pantau secara terus-menerus dan audit sumber daya bersama untuk memastikan sumber daya tersebut hanya dibagikan kepada pengguna utama yang sah.

Antipola umum:

- Tidak menyimpan inventaris sumber daya bersama.
- Tidak mengikuti proses persetujuan akses lintas akun atau publik ke sumber daya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Jika akun Anda berada di AWS Organizations, Anda dapat memberikan akses sumber daya ke seluruh organisasi, unit organisasi tertentu, atau akun individu. Jika akun Anda bukan anggota suatu organisasi, Anda dapat berbagi sumber daya dengan akun individu. Anda dapat memberikan akses lintas akun langsung menggunakan kebijakan berbasis sumber daya — contohnya, kebijakan bucket Amazon Simple Storage Service (Amazon S3) — atau dengan mengizinkan pengguna utama di akun lain menggunakan suatu peran IAM di akun Anda. Saat menggunakan kebijakan sumber daya, pastikan bahwa akses tersebut hanya diberikan kepada pengguna utama yang sah. Tentukan proses untuk menyetujui semua sumber daya yang diperlukan untuk tersedia secara publik.

AWS Identity and Access Management Access Analyzer menggunakan keamanan yang dapat dibuktikan untuk mengidentifikasi semua jalur akses ke sumber daya dari luar akunnya. Keamanan tersebut meninjau kebijakan sumber daya secara terus-menerus, dan melaporkan temuan akses lintas akun dan publik untuk memudahkan Anda menganalisis potensi akses yang meluas. Pertimbangkan untuk mengonfigurasi IAM Access Analyzer dengan AWS Organizations untuk memastikan Anda memiliki visibilitas ke semua akun Anda. IAM Access Analyzer juga mendukung Anda untuk melakukan pratinjau temuan sebelum melakukan deployment izin sumber daya. Hal ini memungkinkan Anda untuk memvalidasi bahwa perubahan kebijakan hanya memberikan akses lintas akun dan publik tertentu ke sumber daya Anda. Saat merancang akses multiakun, Anda dapat menggunakan kebijakan kepercayaan untuk mengontrol dalam kasus seperti apa suatu peran bisa didapatkan. Misalnya, Anda dapat menggunakan kunci kondisi `PrincipalOrgId` untuk menolak upaya untuk mendapatkan peran dari luar AWS Organizations Anda.

AWS Config dapat melaporkan sumber daya yang konfigurasinya salah, dan melalui pemeriksaan kebijakan AWS Config, dapat mendeteksi sumber daya dengan konfigurasi akses publik. Layanan seperti AWS Control Tower dan AWS Security Hub CSPM menyederhanakan deployment kontrol deteksi dan pagar pembatas di seluruh AWS Organizations untuk mengidentifikasi dan memulihkan sumber daya yang terekspos ke publik. Misalnya, AWS Control Tower memiliki pagar pembatas terkelola yang dapat mendeteksi adanya snapshot Amazon EBS yang dapat dipulihkan di Akun AWS.

Langkah implementasi

- Pertimbangkan untuk mengaktifkan AWS Config untuk AWS Organizations: AWS Config memungkinkan Anda mengumpulkan temuan dari banyak akun di dalam satu AWS Organizations ke akun administrator yang ditunjuk. Layanan ini memberikan tampilan komprehensif dan membantu Anda melakukan deployment Aturan AWS Config di seluruh akun untuk mendeteksi sumber daya yang dapat diakses publik.
- Konfigurasikan AWS Identity and Access Management Access Analyzer: IAM Access Analyzer membantu Anda mengidentifikasi sumber daya di organisasi dan akun Anda, seperti bucket Amazon S3 atau peran IAM yang dibagikan kepada entitas eksternal.
- Gunakan perbaikan otomatis di AWS Config untuk merespons perubahan dalam konfigurasi akses publik di bucket Amazon S3: Anda dapat secara otomatis mengaktifkan kembali pengaturan blokir akses publik untuk bucket Amazon S3.
- Implementasikan pemantauan dan peringatan untuk mengidentifikasi apakah Amazon S3 dapat diakses publik: Anda harus menerapkan pemantauan dan peringatan untuk mengidentifikasi saat Amazon S3 Blokir Akses Publik dinonaktifkan, dan saat bucket Amazon S3 dapat diakses publik. Selain itu, jika Anda menggunakan AWS Organizations, Anda dapat membuat kebijakan kontrol layanan yang mencegah perubahan pada kebijakan akses publik Amazon S3. AWS Trusted Advisor memeriksa apakah ada bucket Amazon S3 yang memiliki izin akses terbuka. Izin bucket yang memberikan, mengunggah, atau menghapus akses ke semua orang akan menciptakan potensi masalah keamanan dengan mengizinkan siapa pun untuk menambahkan, mengubah, atau menghapus item dalam bucket. Pemeriksaan Trusted Advisor memeriksa izin bucket eksplisit dan kebijakan bucket terkait yang mungkin mengganti izin bucket. Anda juga dapat menggunakan AWS Config untuk memantau bucket Amazon S3 Anda untuk akses publik. Untuk informasi selengkapnya, kunjungi Cara Menggunakan AWS Config untuk Memantau dan Merespons Bucket Amazon S3 yang Mengizinkan Akses Publik. Saat meninjau akses, penting untuk mengetahui jenis data yang ada di bucket Amazon S3. Amazon Macie membantu menemukan dan melindungi data sensitif seperti PII, PHI, dan kredensial seperti kunci AWS atau privat.

Sumber daya

Dokumen terkait:

- Menggunakan AWS Identity and Access Management Access Analyzer
- Pustaka kontrol AWS Control Tower
- Standar Praktik Terbaik Keamanan Dasar AWS
- Aturan Terkelola AWS Config
- Referensi pemeriksaan AWS Trusted Advisor
- Memantau hasil pemeriksaan AWS Trusted Advisor dengan Amazon EventBridge
- Mengelola Aturan AWS Config Seluruh Akun di Organisasi Anda
- AWS Config dan AWS Organizations

Video terkait:

- Praktik Terbaik untuk mengamankan lingkungan multiakun Anda
- Memahami IAM Access Analyzer Lebih Dalam

SEC03-BP08 Membagikan sumber daya secara aman dalam organisasi Anda

Seiring meningkatnya jumlah beban kerja, Anda mungkin perlu membagikan akses ke sumber daya dalam beban kerja tersebut atau berulang kali menyediakan sumber daya tersebut di seluruh akun. Anda mungkin memiliki konsep untuk membagi lingkungan Anda dalam beberapa kelompok, seperti lingkungan pengembangan, pengujian, dan produksi. Konsep pemisahan ini tidak akan membatasi Anda untuk berbagi secara aman. Dengan membagikan komponen yang tumpang tindih, Anda dapat mengurangi overhead operasional dan memungkinkan pengalaman yang konsisten tanpa ada yang terlewatkan sambil membuat sumber daya yang sama berulang kali.

Hasil yang diinginkan: Meminimalkan akses yang tidak diinginkan menggunakan metode yang aman untuk berbagi sumber daya dalam organisasi, dan membantu inisiatif pencegahan kehilangan data. Mengurangi overhead operasional daripada mengelola komponen satu per satu, yang akan mengurangi kesalahan dari pembuatan komponen yang sama secara manual berulang kali, serta meningkatkan skalabilitas beban kerja. Anda dapat memperoleh manfaat dari pengurangan waktu hingga resolusi di skenario kegagalan multi-titik, dan meningkatkan keyakinan Anda dalam menentukan kapan komponen tidak diperlukan lagi. Untuk panduan preskriptif dalam menganalisis

sumber daya yang dibagikan secara eksternal, lihat SEC03-BP07 Menganalisis akses lintas akun dan publik.

Antipola umum:

- Tidak ada proses untuk terus memantau dan memberikan peringatan otomatis terkait pembagian secara eksternal yang tidak terduga.
- Tidak ada acuan terkait apa yang boleh dan tidak boleh dibagikan.
- Kebijakan terbuka luas secara default, bukannya berbagi secara eksplisit ketika diperlukan.
- Membuat sumber daya dasar secara manual, yang tumpang tindih saat diperlukan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Rancang pola dan kontrol akses Anda untuk mengelola penggunaan sumber daya yang dibagikan secara aman dan hanya dengan entitas tepercaya. Pantau sumber daya yang dibagikan dan tinjau akses sumber daya yang dibagikan secara terus-menerus serta tetap waspada terhadap pembagian yang tidak terduga atau tidak tepat. Lihat Menganalisis akses lintas akun dan akses publik untuk membantu Anda menetapkan tata kelola guna mengurangi akses eksternal hanya ke sumber daya yang memerlukannya, dan menetapkan proses untuk terus memantau dan memberikan peringatan secara otomatis.

Berbagi lintas akun dalam AWS Organizations didukung oleh sejumlah layanan AWS, seperti AWS Security Hub CSPM, Amazon GuardDuty, dan AWS Backup. Layanan tersebut memungkinkan data untuk dibagikan ke akun pusat, dapat diakses dari akun pusat, atau mengelola sumber daya dan data dari akun pusat. Misalnya, AWS Security Hub CSPM dapat mentransfer temuan dari akun individu ke akun pusat sehingga Anda dapat melihat semua temuan. AWS Backup dapat melakukan pencadangan untuk sumber daya dan membagikannya ke seluruh akun. Anda dapat menggunakan AWS Resource Access Manager (AWS RAM) untuk membagikan sumber daya umum, seperti subnet VPC dan lampiran Transit Gateway, AWS Network Firewall, atau jalur Amazon SageMaker AI.

Untuk membatasi akun Anda agar hanya berbagi sumber daya dalam organisasi, gunakan kebijakan kontrol layanan (SCP) untuk mencegah akses ke pengguna utama eksternal. Saat membagikan sumber daya, kombinasikan kontrol berbasis identitas dan kontrol jaringan untuk membuat perimeter data bagi organisasi Anda guna membantu melindungi dari akses yang tidak diinginkan. Perimeter data adalah kumpulan pagar pembatas preventif untuk membantu memverifikasi bahwa hanya identitas yang Anda percaya yang mengakses sumber daya tepercaya dari jaringan yang dikenal.

Kontrol ini menetapkan batas yang sesuai terkait sumber daya apa yang dapat dibagikan, serta mencegah dibagikan atau bocornya sumber daya yang tidak semestinya dibagikan. Misalnya, sebagai bagian dari perimeter data, Anda dapat menggunakan kebijakan titik akhir VPC dan persyaratan `:PrincipalOrgId AWS` untuk memastikan bahwa identitas yang mengakses bucket Amazon S3 adalah milik organisasi Anda. Penting diketahui bahwa SCP tidak berlaku untuk peran terkait layanan (LSR) atau pengguna utama layanan AWS.

Saat menggunakan Amazon S3, nonaktifkan ACL untuk bucket Amazon S3 Anda dan gunakan kebijakan IAM untuk menentukan kontrol akses. Untuk membatasi akses ke Amazon S3 asal dari Amazon CloudFront, migrasikan identitas akses asal (OAI) ke kontrol akses awal (OAC) yang mendukung fitur tambahan, termasuk enkripsi di sisi server dengan AWS Key Management Service.

Dalam beberapa kasus, Anda mungkin ingin mengizinkan pembagian sumber daya ke luar organisasi Anda atau memberikan pihak ketiga akses ke sumber daya Anda. Untuk panduan preskriptif tentang manajemen izin untuk membagikan sumber daya secara eksternal, lihat Manajemen izin.

Langkah implementasi

1. Gunakan AWS Organizations.

   AWS Organizations adalah layanan manajemen akun yang memungkinkan Anda untuk menggabungkan beberapa Akun AWS ke dalam organisasi yang Anda buat dan kelola secara terpusat. Anda dapat mengelompokkan akun ke dalam unit organisasi (OU) dan melampirkan kebijakan yang berbeda ke setiap OU untuk membantu memenuhi kebutuhan anggaran, keamanan, dan kepatuhan. Anda juga dapat mengontrol cara layanan kecerdasan buatan (AI) dan machine learning (ML) AWS mengumpulkan dan menyimpan data, serta menggunakan manajemen multiakun layanan AWS yang terintegrasi dengan Organizations.
2. Integrasikan AWS Organizations dengan layanan AWS.

   Saat Anda mengaktifkan layanan AWS untuk melakukan tugas atas nama Anda di akun anggota organisasi, AWS Organizations membuat peran terkait layanan IAM untuk layanan tersebut di setiap akun anggota. Anda harus mengelola akses tepercaya menggunakan Konsol Manajemen AWS, API AWS, atau AWS CLI. Untuk panduan preskriptif tentang mengaktifkan akses tepercaya, lihat Menggunakan AWS Organizations dengan layanan AWS lainnya dan Layanan AWS yang dapat digunakan dengan Organizations.
3. Buat perimeter data.

   Perimeter AWS biasanya direpresentasikan sebagai organisasi yang dikelola oleh AWS Organizations. Selain sistem dan jaringan on-premise, mengakses sumber daya AWS adalah hal

yang banyak orang kategorikan sebagai salah satu perimeter AWS Saya. Perimeter bertujuan untuk memverifikasi bahwa akses diizinkan jika identitas dipercaya, sumber daya dipercaya, dan jaringan dikenal.

a. Menentukan dan mengimplementasikan perimeter.

   Ikuti langkah yang dijelaskan dalam Implementasi perimeter dalam Membangun Perimeter di laporan resmi AWS untuk setiap persyaratan otorisasi. Untuk panduan preskriptif tentang melindungi lapisan jaringan, lihat Melindungi jaringan.

b. Tetap pantau dan waspada.

   AWS Identity and Access Management Access Analyzer membantu mengidentifikasi sumber daya di organisasi Anda dan akun yang dibagikan kepada entitas eksternal. Anda dapat mengintegrasikan IAM Access Analyzer dengan AWS Security Hub CSPM guna mengirimkan dan menggabungkan temuan untuk sumber daya dari IAM Access Analyzer ke Security Hub CSPM untuk membantu menganalisis postur keamanan lingkungan Anda. Untuk mengaktifkan integrasi, aktifkan IAM Access Analyzer dan Security Hub CSPM di setiap Wilayah di setiap akun. Anda juga dapat menggunakan Aturan AWS Config untuk mengaudit konfigurasi dan memberikan peringatan kepada pihak yang sesuai menggunakan Amazon Q Developer in chat applications dengan AWS Security Hub CSPM. Anda kemudian dapat menggunakan dokumen AWS Systems Manager Automation untuk memperbaiki sumber daya yang melanggar kepatuhan.

c. Untuk panduan preskriptif tentang pemantauan dan peringatan secara berkelanjutan terkait sumber daya yang dibagikan secara eksternal, lihat Menganalisis akses lintas akun dan publik.

4. Gunakan fitur berbagi sumber daya di layanan AWS dan batasi sesuai kebutuhan.

   Banyak layanan AWS dapat Anda gunakan untuk membagikan sumber daya kepada akun lainnya, atau menargetkan sumber daya di akun lainnya, seperti Amazon Machine Images (AMI) dan AWS Resource Access Manager (AWS RAM). Batasi API `ModifyImageAttribute` guna menentukan akun tepercaya untuk berbagi AMI. Tentukan persyaratan `ram:RequestedAllowsExternalPrincipals` saat menggunakan AWS RAM untuk membatasi berbagi hanya ke organisasi Anda, untuk membantu mencegah akses dari identitas yang tidak tepercaya. Untuk panduan preskriptif dan pertimbangan, lihat Berbagi sumber daya dan target eksternal.

5. Gunakan AWS RAM untuk berbagi dengan aman dalam akun atau dengan Akun AWS lainnya.

   AWS RAM membantu Anda secara aman membagikan sumber daya yang Anda buat kepada peran dan pengguna di akun Anda serta kepada Akun AWS lainnya. Dalam lingkungan multiakun,

AWS RAM memungkinkan Anda untuk membuat sumber daya satu kali dan membagikannya kepada akun lain. Pendekatan ini membantu mengurangi overhead operasional sekaligus memberikan konsistensi, visibilitas, dan auditabilitas melalui integrasi dengan Amazon CloudWatch dan AWS CloudTrail, yang tidak Anda dapatkan saat menggunakan akses lintas akun.

Jika ada sumber daya yang sebelumnya dibagikan menggunakan kebijakan berbasis sumber daya, Anda dapat menggunakan API `PromoteResourceShareCreatedFromPolicy` atau yang setara untuk mendukung berbagi sumber daya ke berbagi sumber daya AWS RAM penuh.

Dalam beberapa kasus, Anda mungkin memerlukan beberapa langkah tambahan untuk berbagi sumber daya. Misalnya, untuk membagikan snapshot terenkripsi, Anda perlu membagikan kunci AWS KMS.

Sumber daya

Praktik Terbaik Terkait:

- SEC03-BP07 Menganalisis akses lintas akun dan publik
- SEC03-BP09 Membagikan sumber daya secara aman kepada pihak ketiga
- SEC05-BP01 Membuat lapisan jaringan

Dokumen terkait:

- Pemilik bucket yang memberikan izin lintas akun ke objek yang tidak dimilikinya
- Cara menggunakan Kebijakan Kepercayaan dengan IAM
- Membangun Perimeter Data di AWS
- Cara menggunakan ID eksternal saat memberikan akses ke sumber daya AWSAnda kepada pihak ketiga
- Layanan AWS yang dapat digunakan dengan AWS Organizations
- Membuat perimeter data di AWS: Izinkan hanya identitas tepercaya untuk mengakses data perusahaan

Video terkait:

- Akses Terperinci dengan AWS Resource Access Manager

- Mengamankan perimeter data Anda dengan titik akhir VPC
- Membuat perimeter data di AWS

Alat terkait:

- Contoh Kebijakan Perimeter Data

SEC03-BP09 Membagikan sumber daya secara aman kepada pihak ketiga

Keamanan lingkungan cloud tidak berhenti di organisasi Anda. Organisasi Anda mungkin menggunakan pihak ketiga untuk mengelola sebagian data Anda. Manajemen izin untuk sistem yang dikelola pihak ketiga harus mengikuti praktik akses sesuai kebutuhan menggunakan prinsip hak akses paling rendah dengan kredensial sementara. Melalui kerja sama dengan pihak ketiga, Anda dapat mengurangi cakupan dampak sekaligus risiko dari akses yang tidak diinginkan.

Hasil yang diinginkan: Kredensial AWS Identity and Access Management (IAM) jangka panjang, kunci akses IAM, dan kunci rahasia yang terkait dengan pengguna dapat digunakan oleh siapa saja selama kredensialnya valid dan aktif. Menggunakan peran IAM dan kredensial sementara membantu Anda meningkatkan kekukuhan keamanan dengan mengurangi upaya manajemen kredensial jangka panjang, termasuk manajemen dan overhead operasional terkait detail sensitif tersebut. Dengan pengidentifikasi unik universal (UUID) untuk ID eksternal dalam kebijakan kepercayaan IAM, dan menjaga kebijakan IAM untuk peran IAM di bawah kendali Anda, Anda dapat mengaudit dan memverifikasi bahwa akses yang diberikan kepada pihak ketiga tidak terlalu permisif. Untuk panduan preskriptif dalam menganalisis sumber daya yang dibagikan secara eksternal, lihat SEC03-BP07 Menganalisis akses lintas akun dan publik.

Antipola umum:

- Menggunakan kebijakan kepercayaan IAM default tanpa persyaratan apa pun.
- Menggunakan kunci akses dan kredensial IAM jangka panjang.
- Menggunakan kembali ID eksternal.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Anda mungkin ingin mengizinkan pembagian sumber daya ke luar AWS Organizations atau memberi pihak ketiga akses ke akun Anda. Misalnya, pihak ketiga mungkin menyediakan solusi pemantauan

yang perlu mengakses sumber daya di akun Anda. Dalam kasus tersebut, buat peran lintas akun IAM yang hanya memiliki hak akses sesuai yang dibutuhkan oleh pihak ketiga tersebut. Selain itu, tentukan kebijakan kepercayaan menggunakan persyaratan ID eksternal. Saat menggunakan ID eksternal, Anda atau pihak ketiga dapat membuat ID unik untuk setiap pelanggan, pihak ketiga, atau penghunian. Setelah dibuat, ID unik tidak boleh dikontrol siapa pun selain Anda. Pihak ketiga harus mengimplementasikan proses untuk memberikan ID eksternal melalui cara yang aman, dapat diaudit, dan diproduksi kembali.

Anda juga dapat menggunakan IAM Roles Anywhere guna mengelola peran IAM untuk aplikasi di luar AWS yang menggunakan API AWS.

Hapus peran tersebut jika pihak ketiga sudah tidak perlu mengakses lingkungan Anda. Hindari menyediakan kredensial jangka panjang kepada pihak ketiga. Selalu waspadai layanan AWS lainnya yang mendukung fitur berbagi. Misalnya, AWS Well-Architected Tool memungkinkan berbagi beban kerja dengan Akun AWS lainnya, dan AWS Resource Access Manager membantu membagikan sumber daya AWS yang Anda miliki secara aman kepada akun lain.

Langkah implementasi

1. Gunakan peran lintas akun untuk memberikan akses kepada akun eksternal.

   Peran lintas akun mengurangi jumlah informasi sensitif yang disimpan oleh akun eksternal dan pihak ketiga untuk memberikan layanan kepada pelanggannya. Peran lintas akun memungkinkan Anda untuk memberikan akses ke sumber daya AWS di akun Anda kepada pihak ketiga secara aman, seperti AWS Partner atau akun lainnya di organisasi Anda, dan Anda pun tetap dapat mengelola dan mengaudit akses tersebut.

   Pihak ketiga mungkin memberikan layanan kepada Anda dari infrastruktur hibrida atau menarik data ke lokasi di luar situs. IAM Roles Anywhere membantu Anda memungkinkan beban kerja pihak ketiga berinteraksi dengan beban kerja AWS Anda secara aman dan makin mengurangi kebutuhan kredensial jangka panjang.

   Anda tidak boleh menggunakan kredensial jangka panjang, atau kunci akses yang terkait dengan pengguna, untuk menyediakan akses kepada akun eksternal. Sebaiknya gunakan peran lintas akun untuk memberikan akses lintas akun.
2. Gunakan ID eksternal dengan pihak ketiga.

   Menggunakan ID eksternal memungkinkan Anda untuk menunjuk siapa yang dapat mengambil peran di kebijakan kepercayaan IAM. Kebijakan kepercayaan mungkin mengharuskan pengguna

yang mengambil peran menegaskan persyaratan dan target operasi. Dengan cara ini, pemilik akun dapat mengizinkan peran tersebut untuk diambil hanya dalam keadaan tertentu. Fungsi utama ID eksternal adalah untuk mencegah dan menangani masalah confused deputy.

Gunakan ID eksternal jika Anda adalah pemilik Akun AWS dan sudah mengonfigurasi peran untuk pihak ketiga yang mengakses Akun AWS lainnya selain akun Anda, atau jika Anda mengambil peran atas nama pelanggan yang lain. Jalin kerja sama dengan pihak ketiga atau AWS Partner untuk menentukan persyaratan ID eksternal yang akan disertakan dalam kebijakan kepercayaan IAM.

3. Gunakan ID eksternal yang unik secara universal.

   Implementasikan proses yang membuat nilai unik acak yang unik untuk ID eksternal, seperti pengidentifikasi unik universal (UUID). Pihak ketiga yang menggunakan kembali ID eksternal untuk pengguna yang berbeda tidak menangani masalah confused deputy karena pelanggan A mungkin dapat melihat data pelanggan B menggunakan peran ARN pelanggan B serta duplikat ID eksternal. Dalam lingkungan multipenyewa yang di dalamnya ada pihak ketiga yang mendukung beberapa pelanggan dengan Akun AWS yang berbeda, pihak ketiga tersebut harus menggunakan ID unik yang berbeda sebagai ID eksternal untuk setiap Akun AWS. Pihak ketiga bertanggung jawab untuk mendeteksi duplikat ID eksternal dan memetakan setiap pelanggan secara aman ke ID eksternal masing-masing. Pihak ketiga harus menguji untuk memverifikasi bahwa pihaknya hanya dapat mengambil peran saat menentukan ID eksternal. Pihak ketiga dilarang menyimpan ARN peran pelanggan dan ID eksternal hingga ID eksternal diperlukan.

   ID eksternal bukan sesuatu yang rahasia, tetapi tidak boleh berupa nilai yang mudah ditebak, seperti nomor telepon, nama, atau ID akun. Buat ID eksternal menjadi bidang hanya baca sehingga ID eksternal tidak dapat diubah untuk tujuan meniru penyiapan.

   Anda atau pihak ketiga dapat membuat ID eksternal. Bentuk proses untuk menentukan siapa yang bertanggung jawab dalam pembuatan ID. Siapa pun entitas pembuat ID eksternalnya, pihak ketiga menjaga keunikan dan formatnya tetap konsisten untuk semua pelanggan.

4. Hentikan kredensial jangka panjang yang disediakan pelanggan.

   Hentikan penggunaan kredensial jangka panjang dan gunakan peran lintas akun atau IAM Roles Anywhere. Jika Anda harus menggunakan kredensial jangka panjang, buat rencana atau migrasikan ke akses berbasis peran. Untuk detail tentang manajemen kunci, lihat Manajemen Identitas. Selain itu, jalin kerja sama dengan tim Akun AWS Anda dan pihak ketiga untuk menyusun runbook mitigasi risiko. Untuk panduan preskriptif tentang merespons dan memitigasi potensi dampak insiden keamanan, lihat Respons insiden.

5. Verifikasi bahwa penyiapan memiliki panduan preskriptif atau diotomatisasi.

   Kebijakan yang dibuat untuk akses lintas akun di akun Anda harus mematuhi prinsip hak akses paling rendah. Pihak ketiga harus menyediakan dokumen kebijakan peran atau mekanisme penyiapan otomatis yang menggunakan templat AWS CloudFormation atau yang setara. Hal ini mengurangi potensi kesalahan yang bisa terjadi pada pembuatan kebijakan manual dan menyediakan jejak yang dapat diaudit. Untuk informasi lebih lanjut tentang menggunakan templat CloudFormation untuk membuat peran lintas akun, lihat Peran Lintas Akun.

   Pihak ketiga harus menyediakan mekanisme penyiapan otomatis yang dapat diaudit. Namun, dengan dokumen kebijakan peran yang menguraikan akses yang diperlukan, Anda harus mengotomatiskan penyiapan peran. Anda harus memantau perubahan dengan deteksi penyimpangan menggunakan templat CloudFormation atau yang setara sebagai bagian dari praktik audit.

6. Antisipasi perubahan.

   Struktur akun Anda, kebutuhan Anda akan pihak ketiga, atau penawaran layanan yang disediakan dapat berubah. Anda harus mengantisipasi perubahan dan kegagalan, dan membuat rencana yang sesuai dengan orang, proses, dan teknologi yang tepat. Audit tingkat akses yang Anda berikan secara berkala, dan terapkan metode deteksi untuk memberi tahu Anda tentang perubahan yang tidak terduga. Pantau dan audit penggunaan peran dan penyimpanan data ID eksternal. Anda harus bersiap untuk mencabut akses pihak ketiga, baik untuk sementara atau secara permanen, jika ada perubahan atau pola akses yang tidak terduga. Selain itu, ukur dampak atas operasi pencabutan Anda, termasuk waktu yang diperlukan untuk melakukannya, orang yang terlibat, biaya, dan dampak terhadap sumber daya lainnya.

   Untuk panduan preskriptif tentang metode deteksi, lihat Praktik terbaik deteksi.

Sumber daya

Praktik Terbaik Terkait:

- SEC02-BP02 Menggunakan kredensial sementara
- SEC03-BP05 Menentukan pagar pembatas izin untuk organisasi Anda
- SEC03-BP06 Mengelola akses berdasarkan siklus hidup
- SEC03-BP07 Menganalisis akses lintas akun dan publik
- SEC04 Deteksi

Dokumen terkait:

- Pemilik bucket yang memberikan izin lintas akun ke objek yang tidak dimilikinya
- Cara menggunakan kebijakan kepercayaan dengan peran IAM
- Mendelegasikan akses di seluruh Akun AWS menggunakan peran IAM
- Bagaimana cara mengakses sumber daya di Akun AWS lainnya menggunakan IAM?
- Praktik terbaik keamanan dalam IAM
- Logika evaluasi kebijakan lintas akun
- Cara menggunakan ID eksternal saat memberikan akses ke sumber daya AWS Anda kepada pihak ketiga
- Mengumpulkan Informasi dari Sumber Daya AWS CloudFormation yang Dibuat di Akun Eksternal dengan Sumber Daya Kustom
- Menggunakan ID Eksternal Secara Aman untuk Mengakses Akun AWS Milik Pihak Lain
- Memperluas peran IAM ke beban kerja di luar IAM dengan IAM Roles Anywhere

Video terkait:

- Bagaimana caranya mengizinkan pengguna atau peran di Akun AWS yang terpisah untuk mengakses Akun AWS saya?
- AWS re:Invent 2018: Menjadi Master Kebijakan IAM dalam 60 Menit atau Kurang
- Pusat Pengetahuan AWS Live: Praktik Terbaik IAM dan Keputusan Rancangan

Contoh terkait:

- Well-Architected Lab - Pengambilan peran IAM lintas akun Lambda (Level 300)
- Mengonfigurasi akses lintas akun ke Amazon DynamoDB
- AWS STS Network Query Tool

Deteksi

Pertanyaan

- SEC 4. Bagaimana cara mendeteksi dan menyelidiki peristiwa keamanan?

SEC 4. Bagaimana cara mendeteksi dan menyelidiki peristiwa keamanan?

Catat dan analisis peristiwa dari log dan metrik untuk mendapatkan visibilitas. Ambil tindakan atas peristiwa keamanan dan potensi ancaman untuk membantu mengamankan beban kerja Anda.

Praktik terbaik

- SEC04-BP01 Mengonfigurasi pencatatan log layanan dan aplikasi
- SEC04-BP02 Menganalisis log, temuan, dan metrik secara terpusat
- SEC04-BP03 Mengotomatiskan respons untuk peristiwa
- SEC04-BP04 Implementasikan peristiwa keamanan yang dapat ditindaklanjuti

SEC04-BP01 Mengonfigurasi pencatatan log layanan dan aplikasi

Menyimpan log peristiwa keamanan dari layanan dan aplikasi. Hal ini merupakan prinsip fundamental dalam keamanan untuk audit, penyelidikan, dan kasus penggunaan operasional, serta merupakan persyaratan keamanan umum yang didorong oleh prosedur, kebijakan, dan standar tata kelola, risiko, serta kepatuhan (GRC).

Hasil yang diinginkan: Sebuah organisasi harus dapat dengan andal dan konsisten mengambil log peristiwa keamanan dari aplikasi dan layanan AWS dengan cepat saat diperlukan untuk memenuhi proses atau kewajiban internal, seperti respons insiden keamanan. Sebaiknya pusatkan log untuk mendapatkan hasil operasional yang lebih baik.

Antipola umum:

- Log disimpan tanpa batas waktu yang jelas atau dihapus terlalu cepat.
- Semua orang dapat mengakses log.
- Sepenuhnya menggunakan proses manual untuk tata kelola dan penggunaan log.
- Menyimpan setiap jenis log untuk berjaga-jaga jika diperlukan.
- Memeriksa integritas log hanya jika diperlukan.

Manfaat menjalankan praktik terbaik ini: Mengimplementasikan mekanisme analisis akar masalah (RCA) untuk insiden keamanan dan sumber bukti untuk kewajiban tata kelola, risiko, dan kepatuhan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Selama penyelidikan keamanan atau kasus penggunaan lain berdasarkan kebutuhan Anda, Anda harus dapat meninjau log yang relevan untuk mencatat dan memahami seluruh cakupan serta garis waktu insiden. Log juga diperlukan untuk pembuatan peringatan yang mengindikasikan bahwa tindakan tertentu telah terjadi. Sangat penting untuk memilih, mengaktifkan, menyimpan, dan menyiapkan mekanisme kueri dan pengambilan serta pembuatan peringatan.

Langkah implementasi

- Pilih dan aktifkan sumber log. Sebelum melakukan penyelidikan keamanan, Anda perlu mengambil log yang relevan untuk merekonstruksi aktivitas secara surut di Akun AWS. Pilih dan aktifkan sumber log yang relevan dengan beban kerja Anda.

  Kriteria pemilihan sumber log harus didasarkan pada kasus penggunaan yang diperlukan oleh bisnis Anda. Tetapkan jejak untuk setiap Akun AWS menggunakan AWS CloudTrail atau jejak AWS Organizations, dan konfigurasikan bucket Amazon S3 untuk jejak tersebut.

  AWS CloudTrail adalah layanan pencatatan log yang melacak panggilan API yang dibuat terhadap Akun AWS yang merekam aktivitas layanan AWS. Layanan tersebut diaktifkan secara default dengan peristiwa manajemen retensi 90 hari yang dapat diambil melalui riwayat Peristiwa CloudTrail menggunakan Konsol Manajemen AWS, AWS CLI, atau AWS SDK. Untuk visibilitas dan retensi peristiwa data yang lebih lama, buat jejak CloudTrail dan kaitkan dengan bucket Amazon S3, serta dengan grup log Amazon CloudWatch (opsional). Anda juga dapat membuat CloudTrail Lake, yang menyimpan log CloudTrail hingga tujuh tahun dan menyediakan fasilitas kueri berbasis SQL.

  AWS menyarankan agar pelanggan yang menggunakan VPC mengaktifkan lalu lintas jaringan dan log DNS menggunakan Log Arus VPC dan log kueri Amazon Route 53 Resolver, serta mengalirkannya ke bucket Amazon S3 atau grup log CloudWatch. Anda dapat membuat log arus VPC untuk VPC, subnet, dan antarmuka jaringan. Untuk Log Arus VPC, Anda dapat memilih cara dan tempat penggunaan Log Arus untuk mengurangi biaya.

  Log AWS CloudTrail, Log Arus VPC, dan log kueri Route 53 Resolver merupakan sumber pencatatan log dasar untuk mendukung penyelidikan keamanan di AWS. Anda juga dapat menggunakan Danau Keamanan Amazon untuk mengumpulkan, menormalisasi, dan menyimpan data log ini dalam format Apache Parquet dan Open Cybersecurity Schema Framework (OCSF), yang siap untuk kueri. Danau Keamanan juga mendukung log AWS lainnya dan log dari sumber pihak ketiga.

Layanan AWS dapat membuat log yang tidak direkam oleh sumber log dasar, seperti log Elastic Load Balancing, log AWS WAF, log perekam AWS Config, temuan Amazon GuardDuty, log audit Amazon Elastic Kubernetes Service (Amazon EKS), dan log aplikasi serta sistem operasi instans Amazon EC2. Untuk daftar lengkap opsi pencatatan log dan pemantauan, lihat Lampiran A: Penentuan kemampuan cloud – Pencatatan Log dan Peristiwa dalam Panduan Respons Insiden Keamanan AWS.

- Pelajari kemampuan pencatatan log untuk setiap aplikasi dan layanan AWS: Setiap layanan dan aplikasi AWS memberikan opsi untuk penyimpanan log yang masing-masing dilengkapi dengan kemampuan retensi dan siklus hidup. Dua layanan penyimpanan yang paling umum adalah Amazon Simple Storage Service (Amazon S3) dan Amazon CloudWatch. Untuk periode retensi yang panjang, sebaiknya gunakan Amazon S3 untuk efektivitas biaya dan kemampuan siklus hidup yang fleksibel. Jika opsi pencatatan log utama adalah Log Amazon CloudWatch, sebagai opsi, Anda dapat mempertimbangkan untuk mengarsipkan log yang jarang diakses ke Amazon S3.
- Pilih penyimpanan log: Pilihan penyimpanan log umumnya dikaitkan dengan alat kueri yang digunakan, kemampuan retensi, seberapa familier, dan biaya. Opsi utama untuk penyimpanan log adalah bucket Amazon S3 atau grup Log CloudWatch.

  Bucket Amazon S3 menyediakan penyimpanan yang tahan lama dan hemat biaya, dengan kebijakan siklus hidup opsional. Log yang disimpan di bucket Amazon S3 dapat dikueri menggunakan layanan seperti Amazon Athena.

  Grup log CloudWatch menyediakan penyimpanan yang tahan lama dan fasilitas kueri bawaan melalui Wawasan Log CloudWatch.
- Identifikasi retensi log yang sesuai: Saat Anda menggunakan bucket Amazon S3 atau grup log CloudWatch untuk menyimpan log, Anda harus menetapkan siklus hidup yang memadai untuk setiap sumber log guna mengoptimalkan biaya penyimpanan dan pengambilan. Pada umumnya, pelanggan memiliki waktu antara tiga bulan hingga satu tahun untuk melakukan kueri log, dengan periode retensi hingga tujuh tahun. Pilihan ketersediaan dan retensi harus selaras dengan persyaratan keamanan dan gabungan dari undang-undang, peraturan, serta kewajiban bisnis.
- Aktifkan pencatatan log untuk setiap layanan dan aplikasi AWS dengan kebijakan retensi dan siklus hidup yang tepat: Untuk setiap aplikasi dan layanan AWS di organisasi Anda, cari panduan konfigurasi pencatatan log khusus:
  - Mengonfigurasi Jejak AWS CloudTrail
  - Mengonfigurasi Log Arus VPC

  - Mengonfigurasi Ekspor Temuan Amazon GuardDuty
  - Mengonfigurasi perekaman AWS Config
  - Mengonfigurasi lalu lintas ACL web AWS WAF
  - Mengonfigurasi log lalu lintas jaringan AWS Network Firewall
  - Mengonfigurasi log akses Elastic Load Balancing
  - Mengonfigurasi log kueri Amazon Route 53 Resolver
  - Mengonfigurasi log Amazon RDS
  - Mengonfigurasi log Bidang Kendali Amazon EKS
  - Mengonfigurasi agen Amazon CloudWatch untuk instans Amazon EC2 dan server on-premise
- Pilih dan implementasikan mekanisme kueri untuk log: Untuk kueri log, Anda dapat menggunakan CloudWatch Wawasan Log untuk data yang disimpan di grup log CloudWatch, dan Amazon Athena serta Amazon OpenSearch Service untuk data yang disimpan di Amazon S3. Anda dapat menggunakan alat kueri pihak ketiga seperti layanan informasi keamanan dan manajemen peristiwa (SIEM).

  Proses pemilihan alat kueri harus mempertimbangkan aspek manusia, proses, dan teknologi operasi keamanan Anda. Pilih alat yang memenuhi persyaratan operasional, bisnis, dan keamanan, serta dapat diakses dan dipelihara dalam jangka panjang. Perlu diingat bahwa alat kueri berfungsi secara optimal saat jumlah log yang dipindai masih berada dalam batasan alat tersebut. Tidak jarang terdapat beberapa alat kueri karena adanya kendala biaya atau teknis.

  Misalnya, Anda menggunakan informasi keamanan dan manajemen peristiwa pihak ketiga untuk menjalankan kueri data selama 90 hari terakhir, tetapi menggunakan Athena untuk menjalankan kueri di atas 90 hari karena biaya penyerapan log SIEM. Terlepas dari implementasi, pastikan pendekatan Anda meminimalkan jumlah alat yang diperlukan untuk memaksimalkan efisiensi operasional, khususnya selama penyelidikan peristiwa keamanan.
- Gunakan log untuk peringatan: AWS memberikan peringatan melalui beberapa layanan keamanan:
  - AWS Config memantau dan merekam konfigurasi sumber daya AWS Anda serta membantu mengotomatiskan evaluasi dan perbaikan berdasarkan konfigurasi yang diinginkan.
  - Amazon GuardDuty adalah layanan deteksi ancaman yang terus memantau aktivitas mencurigakan dan perilaku tidak sah untuk melindungi Akun AWS dan beban kerja Anda. GuardDuty menyerap, menggabungkan, dan menganalisis informasi dari berbagai sumber, seperti manajemen dan peristiwa data AWS CloudTrail, log DNS, Log Arus VPC, dan log Audit Amazon EKS. GuardDuty mengambil aliran data independen secara langsung dari CloudTrail, Log Arus VPC, log kueri DNS, dan Amazon EKS. Anda tidak perlu mengelola kebijakan bucket

Amazon S3 atau mengubah cara Anda mengumpulkan dan menyimpan log. Sebaiknya tetap simpan log tersebut untuk tujuan penyelidikan dan kepatuhan.

- AWS Security Hub CSPM menyediakan satu tempat yang mengumpulkan, mengatur, dan memprioritaskan peringatan keamanan atau temuan Anda dari beberapa layanan AWS serta produk pihak ketiga opsional untuk menampilkan peringatan keamanan dan status kepatuhan secara komprehensif.

Anda juga dapat menggunakan mesin pembuat peringatan kustom untuk peringatan keamanan yang tidak dicakup oleh layanan ini atau untuk peringatan tertentu yang relevan dengan lingkungan Anda. Untuk informasi tentang pembuatan peringatan dan deteksi tersebut, lihat Deteksi dalam Panduan Respons Insiden Keamanan AWS.

Sumber daya

Praktik Terbaik Terkait:

- SEC04-BP02 Menganalisis log, temuan, dan metrik secara terpusat
- SEC07-BP04 Menentukan manajemen siklus hidup data
- SEC10-BP06 Melakukan deployment alat di awal

Dokumen terkait:

- Panduan Respons Insiden Keamanan AWS
- Memulai Danau Keamanan Amazon
- Memulai: Amazon CloudWatch Logs
- Solusi Partner Keamanan: Pencatatan Log dan Pemantauan

Video terkait:

- AWS re:Invent 2022 - Memperkenalkan Danau Keamanan Amazon

Contoh terkait:

- Assisted Log Enabler untuk AWS
- Ekspor Temuan AWS Security Hub CSPM Historis

Alat terkait:

- Snowflake for Cybersecurity

SEC04-BP02 Menganalisis log, temuan, dan metrik secara terpusat

Tim operasi keamanan mengandalkan pengumpulan data dan penggunaan alat pencarian untuk menemukan potensi peristiwa yang menjadi perhatian, yang mungkin menandakan aktivitas yang tidak diotorisasi atau perubahan yang tidak diinginkan. Namun, menganalisis data yang terkumpul dan memproses informasi secara manual saja tidak cukup untuk mengimbangi volume informasi yang mengalir dari arsitektur kompleks. Analisis dan pelaporan saja belum cukup untuk memfasilitasi penetapan sumber daya yang tepat untuk mengerjakan peristiwa pada waktu yang diinginkan.

Praktik terbaik untuk membangun tim operasi keamanan yang matang adalah dengan mengintegrasikan aliran peristiwa keamanan dan temuan ke dalam notifikasi dan sistem alur kerja seperti sistem ticketing, sistem masalah atau bug, atau sistem informasi keamanan dan manajemen peristiwa (SIEM) lainnya. Hal ini mengalihkan alur kerja dari email dan laporan statis, sehingga Anda dapat merutekan, mengeskalasi, dan mengelola peristiwa atau temuan. Banyak organisasi yang juga mengintegrasikan peringatan keamanan ke dalam obrolan atau kolaborasi mereka, dan platform produktivitas developer. Untuk organisasi yang memulai otomatisasi, sistem ticketing yang didorong API dan berlatensi rendah menawarkan berbagai fleksibilitas saat merencanakan apa yang harus diotomatiskan terlebih dahulu.

Praktik terbaik ini tidak hanya berlaku untuk peristiwa keamanan yang dibuat dari pesan log yang menggambarkan aktivitas pengguna atau peristiwa jaringan, tetapi juga dari perubahan yang terdeteksi dalam infrastruktur. Kemampuan untuk mendeteksi perubahan, menentukan apakah perubahan tersebut sesuai, dan kemudian mengarahkan informasi tersebut ke alur kerja remediasi begitu penting dalam memelihara dan memvalidasi arsitektur yang aman, saat terjadi perubahan yang tidak diinginkan tetapi sulit dideteksi sehingga eksekusinya saat ini tidak dapat dicegah dengan kombinasi konfigurasi AWS Identity and Access Management(IAM) dan AWS Organizations.

Amazon GuardDuty dan AWS Security Hub CSPM memberikan gabungan, deduplikasi, dan mekanisme analisis untuk catatan log, yang juga dibuat tersedia untuk Anda via layanan AWS lainnya. GuardDuty menyerap, menggabungkan, dan menganalisis, informasi dari sumber seperti manajemen AWS CloudTrail dan peristiwa data, log DNS VPC, dan Log Alur VPC. Security Hub CSPM dapat menyerap, menggabungkan, dan menganalisis hasil dari GuardDuty, AWS Config, Amazon Inspector, Amazon Macie, AWS Firewall Manager, dan sebagian besar produk keamanan pihak ketiga yang tersedia di AWS Marketplace, dan jika langsung dibangun, kode milik Anda

sendiri. GuardDuty dan Security Hub CSPM memiliki model Administrator-Anggota yang dapat mengagregatkan temuan dan wawasan dari beberapa akun, dan Security Hub CSPM sering digunakan oleh pelanggan SIEM on-premise sebagai log sisi AWS dan peringatan praprosesor dan agregator yang dapat diserap Amazon EventBridge melalui prosesor dan penerus berbasis AWS Lambda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Evaluasikan kemampuan pemrosesan log: Evaluasikan opsi yang tersedia untuk pemrosesan log.
  - Gunakan Amazon OpenSearch Service untuk mencatat dan memantau (hampir) semuanya
  - Temukan partner yang ahli dalam pencatatan log dan pemantauan.
- Sebagai awal untuk menganalisis log CloudTrail, uji Amazon Athena.
  - Konfigurasikan Athena untuk menganalisis log CloudTrail
- Implementasikan sentralisasi pencatatan dalam AWS: Lihat solusi contoh AWS berikut untuk memusatkan pencatatan dari berbagai sumber.
  - Solusi sentralisasi pencatatan
- Implementasikan sentralisasi pencatatan dengan partner: APN Partner memiliki solusi untuk membantu Anda menganalisis log secara terpusat.
  - Pencatatan dan Pemantauan

Sumber daya

Dokumen terkait:

- AWS Answers: Pencatatan Log Terpusat
- AWS Security Hub CSPM
- Amazon CloudWatch
- Amazon EventBridge
- Mulai menggunakan: Amazon CloudWatch Logs
- Solusi Partner Keamanan: Pencatatan Log dan Pemantauan

Video terkait:

- Konfigurasi dan Kepatuhan Sumber Daya Pemantauan Terpusat
- Memperbaiki Temuan Amazon GuardDuty dan AWS Security Hub CSPM
- Manajemen ancaman di cloud: Amazon GuardDuty dan AWS Security Hub CSPM

SEC04-BP03 Mengotomatiskan respons untuk peristiwa

Menggunakan otomatisasi untuk menyelidiki dan menangani peristiwa dapat mengurangi upaya manusia dan potensi kesalahan, serta memungkinkan Anda untuk menskalakan kemampuan penyelidikan. Tinjauan rutin dapat membantu Anda menyesuaikan alat otomatisasi, dan mengulanginya secara iteratif.

Di AWS, menyelidiki peristiwa menarik dan informasi tentang potensi perubahan yang tidak diharapkan ke alur kerja otomatis dapat dicapai menggunakan Amazon EventBridge. Layanan ini menyediakan mesin aturan yang dapat diskalakan dan dirancang untuk mengelola format peristiwa AWS native (seperti peristiwa AWS CloudTrail), serta peristiwa kustom yang dapat dihasilkan dari aplikasi Anda. Amazon GuardDuty juga memungkinkan Anda untuk merutekan peristiwa ke sistem alur kerja untuk mereka yang membangun sistem respons insiden (AWS Step Functions), atau ke Akun Keamanan pusat, atau ke bucket untuk analisis lebih jauh.

Mendeteksi perubahan dan merutekan informasi ini ke alur kerja yang sesuai dapat dilakukan dengan menggunakan Aturan AWS Config dan Paket Konformasi. AWS Config mendeteksi perubahan ke layanan dalam cakupan (walaupun dengan latensi yang lebih tinggi dari EventBridge) dan membuat peristiwa yang dapat di-parse menggunakan Aturan AWS Config untuk rollback, penerapan kebijakan kepatuhan, dan melanjutkan informasi ke sistem, seperti mengubah platform manajemen dan sistem ticketing operasional. Selain menulis fungsi Lambda Anda sendiri untuk merespons peristiwa AWS Config, Anda juga dapat memanfaatkan Kit Pengembangan Aturan AWS Config, dan pustaka sumber terbuka Aturan AWS Config. Paket konformasi adalah kumpulan Aturan AWS Config dan tindakan perbaikan yang Anda deploy sebagai entitas tunggal yang ditulis sebagai templat YAML. Sebuah sampel templat paket konformasi tersedia untuk Pilar Keamanan Well-Architected.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Implementasikan peringatan yang diotomatiskan dengan GuardDuty: GuardDuty adalah layanan deteksi ancaman yang terus-menerus memantau aktivitas berbahaya dan perilaku yang tidak diotorisasi, untuk melindungi Akun AWS dan beban kerja Anda. Aktifkan GuardDuty dan konfigurasikan peringatan yang diotomatiskan.

- Otomatiskan proses penyelidikan: Kembangkan proses otomatis yang menyelidiki peristiwa serta melaporkan informasi ke administrator untuk menghemat waktu.
  - Lab: Penggunaan Amazon GuardDuty

Sumber daya

Dokumen terkait:

- AWS Jawaban: Pencatatan Log Terpusat
- AWS Security Hub CSPM
- Amazon CloudWatch
- Amazon EventBridge
- Mulai menggunakan: Amazon CloudWatch Logs
- Solusi Partner Keamanan: Pencatatan Log dan Pemantauan
- Menyiapkan Amazon GuardDuty

Video terkait:

- Centrally Monitoring Resource Configuration and Compliance
- Memperbaiki Temuan Amazon GuardDuty dan AWS Security Hub CSPM
- Manajemen ancaman di cloud: Amazon GuardDuty dan AWS Security Hub CSPM

Contoh terkait:

- Lab: Deployment Otomatis Kontrol Deteksi

SEC04-BP04 Implementasikan peristiwa keamanan yang dapat ditindaklanjuti

Buat peringatan yang dikirimkan ke tim Anda dan dapat ditindaklanjuti oleh mereka. Pastikan peringatan mencakup informasi yang relevan bagi tim untuk mengambil tindakan. Untuk setiap mekanisme deteksi yang Anda miliki, Anda juga harus memiliki proses, dalam bentuk runbook atau playbook, untuk menyelidiki. Contohnya, ketika Anda mengaktifkan Amazon GuardDuty, ini menghasilkan temuan yang berbeda.. Anda harus memiliki entri runbook untuk setiap jenis temuan, contohnya, jika ditemukan trojan, runbook Anda memiliki instruksi mudah yang menginstruksikan seseorang untuk menyelidiki dan memperbaiki.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Rendah

Panduan implementasi

- Temukan metrik yang tersedia untuk layanan AWS: Temukan metrik yang tersedia melalui Amazon CloudWatch untuk layanan yang Anda gunakan.
  - Dokumentasi layanan AWS
  - Menggunakan Metrik Amazon CloudWatch
- Konfigurasikan alarm Amazon CloudWatch.
  - Menggunakan Alarm Amazon CloudWatch

Sumber daya

Dokumen terkait:

- Amazon CloudWatch
- Amazon EventBridge
- Solusi Partner Keamanan: Logging dan Pemantauan

Video terkait:

- Konfigurasi dan Kepatuhan Sumber Daya Pemantauan Terpusat
- Memperbaiki Temuan Amazon GuardDuty dan AWS Security Hub CSPM
- Manajemen ancaman di cloud: Amazon GuardDuty dan AWS Security Hub CSPM

Perlindungan infrastruktur

Pertanyaan

- SEC 5. Bagaimana cara melindungi sumber daya jaringan Anda?
- SEC 6. Bagaimana cara melindungi sumber daya komputasi Anda?

SEC 5. Bagaimana cara melindungi sumber daya jaringan Anda?

Setiap beban kerja yang memiliki suatu bentuk konektivitas jaringan, baik internet atau jaringan privat, memerlukan beberapa lapisan pertahanan untuk membantu melindungi dari ancaman eksternal dan internal berbasis jaringan.

Praktik terbaik

- SEC05-BP01 Membuat lapisan jaringan
- SEC05-BP02 Mengontrol lalu lintas di semua lapisan
- SEC05-BP03 Mengotomatiskan perlindungan jaringan
- SEC05-BP04 Mengimplementasikan inspeksi dan perlindungan

SEC05-BP01 Membuat lapisan jaringan

Kelompokkan komponen dengan persyaratan sensitivitas yang sama ke dalam lapisan-lapisan untuk meminimalkan cakupan potensi dampak dari akses yang tidak sah. Misalnya, sebuah klaster basis data di cloud privat virtual (VPC) yang tidak memerlukan akses internet harus ditempatkan dalam subnet yang tidak memiliki rute ke atau dari internet. Lalu lintas hanya boleh mengalir dari sumber daya terdekat berikutnya yang paling tidak sensitif. Pertimbangkan aplikasi web di balik penyeimbang beban. Basis data Anda tidak boleh dapat diakses secara langsung dari penyeimbang beban. Hanya logika bisnis dan server web yang boleh memiliki akses langsung ke basis data Anda.

Hasil yang diinginkan: Membuat jaringan berlapis. Jaringan berlapis membantu mengelompokkan komponen jaringan yang serupa secara logis. Jaringan ini memperkecil potensi cakupan dampak dari akses jaringan yang tidak sah. Jaringan berlapis yang tepat mempersulit pengguna yang tidak sah untuk beralih ke sumber daya tambahan di lingkungan AWS Anda. Selain mengamankan jalur jaringan internal, Anda juga perlu melindungi edge jaringan, seperti aplikasi web dan titik akhir API.

Antipola umum:

- Membuat semua sumber daya dalam satu VPC atau subnet.
- Menggunakan grup keamanan yang terlalu permisif.
- Gagal menggunakan subnet.
- Mengizinkan akses langsung ke penyimpanan data seperti basis data.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Komponen seperti instans Amazon Elastic Compute Cloud (Amazon EC2), klaster basis data Amazon Relational Database Service (Amazon RDS), dan fungsi AWS Lambda yang memiliki persyaratan keterjangkauan yang sama dapat disegmentasikan menjadi lapisan yang dibentuk oleh subnet. Pertimbangkan untuk melakukan deployment beban kerja nirserver, seperti fungsi Lambda, dalam VPC atau di balik Amazon API Gateway. Tugas AWS Fargate yang tidak memerlukan akses internet harus ditempatkan di subnet yang tidak memiliki rute ke atau dari internet. Pendekatan berlapis ini memitigasi dampak kesalahan konfigurasi satu lapisan yang dapat mengizinkan akses yang tidak diinginkan. Untuk AWS Lambda, Anda dapat menjalankan fungsi di VPC untuk memanfaatkan kontrol berbasis VPC.

Untuk konektivitas jaringan yang dapat mencakup ribuan VPC, Akun AWS, dan jaringan on-premise, Anda harus menggunakan AWS Transit Gateway. Transit Gateway bertindak sebagai hub yang mengontrol cara perutean lalu lintas di antara semua jaringan yang terhubung, yang bertindak seperti jari-jari roda. Lalu lintas antara Amazon Virtual Private Cloud (Amazon VPC) dan Transit Gateway tetap berada dalam jaringan privat AWS, sehingga mengurangi paparan eksternal terkait pengguna yang tidak sah dan potensi masalah keamanan. Peering Antarwilayah Transit Gateway juga mengenkripsi lalu lintas Antarwilayah tanpa titik kegagalan tunggal atau hambatan bandwidth.

Langkah implementasi

- Gunakan Reachability Analyzer untuk menganalisis jalur antara sumber dan tujuan berdasarkan konfigurasi: Reachability Analyzer memungkinkan Anda untuk mengotomatiskan verifikasi konektivitas ke dan dari sumber daya yang terhubung ke VPC. Perhatikan bahwa analisis ini dilakukan dengan meninjau konfigurasi (tidak ada paket jaringan yang dikirimkan dalam menjalankan analisis ini).
- Gunakan Penganalisis Akses Jaringan Amazon VPC untuk mengidentifikasi akses jaringan yang tidak diinginkan ke sumber daya: Penganalisis Akses Jaringan Amazon VPC memungkinkan Anda untuk menentukan persyaratan akses jaringan dan mengidentifikasi jalur jaringan potensial.
- Pertimbangkan apakah sumber daya harus ada di subnet publik: Jangan menempatkan sumber daya di subnet publik VPC Anda kecuali sumber daya benar-benar harus menerima lalu lintas jaringan masuk dari sumber publik.
- Buat subnet di VPC Anda: Buat subnet untuk setiap lapisan jaringan (dalam grup yang menyertakan beberapa Zona Ketersediaan) untuk meningkatkan segmentasi mikro. Selain itu, pastikan Anda telah mengaitkan tabel rute yang benar ke subnet Anda untuk mengontrol perutean dan konektivitas internet.

- Gunakan AWS Firewall Manager untuk mengelola grup keamanan VPC Anda: AWS Firewall Manager membantu mengurangi beban manajemen dalam penggunaan beberapa grup keamanan.
- Gunakan AWS WAF untuk melindungi dari kerentanan web umum: AWS WAF dapat membantu meningkatkan keamanan edge dengan memeriksa lalu lintas untuk kerentanan web umum, misalnya injeksi SQL. Layanan ini juga dapat Anda gunakan untuk membatasi alamat IP yang berasal dari negara atau lokasi geografis tertentu.
- Gunakan Amazon CloudFront sebagai jaringan distribusi konten (CDN): Amazon CloudFront dapat membantu mempercepat aplikasi web dengan menyimpan data lebih dekat ke pengguna. Hal ini juga meningkatkan keamanan edge dengan menerapkan HTTPS, membatasi akses ke area geografis, dan memastikan bahwa lalu lintas jaringan hanya dapat mengakses sumber daya saat dirutekan melalui CloudFront.
- Gunakan Amazon API Gateway saat membuat antarmuka pemrograman aplikasi (API): Amazon API Gateway membantu menerbitkan, memantau, dan mengamankan API WebSocket, HTTPS, dan REST.

Sumber daya

Dokumen terkait:

- AWS Firewall Manager
- Amazon Inspector
- Keamanan Amazon VPC
- Reachability Analyzer
- Penganalisis Akses Jaringan Amazon VPC

Video terkait:

- Arsitektur referensi AWS Transit Gateway untuk banyak VPC
- Perlindungan dan Akselerasi Aplikasi dengan Amazon CloudFront, AWS WAF, dan AWS Shield
- AWS re:Inforce 2022 - Validasikan kontrol akses jaringan efektif di AWS
- AWS re:Inforce 2022 - Perlindungan tingkat lanjut terhadap bot menggunakan AWS WAF

Contoh terkait:

- Well-Architected Lab - Deployment Otomatis VPC

- Lokakarya: Penganalisis Akses Jaringan Amazon VPC

SEC05-BP02 Mengontrol lalu lintas di semua lapisan

Ketika merancang topologi jaringan, Anda harus memeriksa persyaratan konektivitas setiap komponen. Misalnya, periksa apakah komponen memerlukan aksesibilitas internet (masuk dan keluar), konektivitas ke VPC, layanan edge, dan pusat data eksternal.

Dengan VPC, Anda dapat menentukan topologi jaringan yang menjangkau Wilayah AWS dengan rentang alamat IPv4 privat yang Anda atur, atau rentang alamat IPv6 yang dipilih oleh AWS. Anda harus menerapkan beberapa kontrol dengan pendekatan pertahanan mendalam untuk lalu lintas masuk dan keluar, termasuk penggunaan grup keamanan (firewall inspeksi stateful), ACL Jaringan, subnet, dan tabel rute. Di dalam VPC, Anda dapat membuat subnet di Zona Ketersediaan. Masing-masing subnet dapat memiliki tabel rute terkait yang menentukan aturan perutean untuk mengelola jalur yang digunakan lalu lintas dalam subnet. Anda dapat menentukan subnet yang dapat dirutekan internet dengan rute yang mengarah ke gateway NAT atau internet yang terikat ke VPC, atau melalui VPC lainnya.

Saat diluncurkan di dalam VPC, instans, basis data Amazon Relational Database Service(Amazon RDS), atau layanan lainnya akan memiliki grup keamanannya sendiri per antarmuka jaringan. Firewall ini berada di luar lapisan sistem operasi dan dapat digunakan untuk menentukan aturan bagi lalu lintas masuk dan keluar yang diizinkan. Anda juga dapat menentukan hubungan antargrup keamanan. Misalnya, instans dalam grup keamanan tingkat basis data hanya menerima lalu lintas dari instans dalam tingkat aplikasi, dengan merujuk ke grup keamanan yang diterapkan ke instans yang terlibat. Namun jika Anda menggunakan protokol non-TCP, Anda tidak perlu memiliki instans Amazon Elastic Compute Cloud(Amazon EC2) yang dapat diakses langsung dengan internet (bahkan dengan port yang dibatasi oleh grup keamanan) tanpa penyeimbang beban, atau CloudFront. Hal ini akan membantu melindunginya dari akses yang tidak diharapkan melalui sistem operasi atau masalah aplikasi. Subnet juga dapat memiliki ACL jaringan yang terikat dengannya, yang berperan sebagai firewall stateless. Anda harus mengonfigurasi jaringan ACL untuk mempersempit cakupan lalu lintas yang diizinkan antarlapisan, Anda juga harus menentukan aturan masuk dan keluar.

Beberapa layanan AWS memerlukan komponen untuk mengakses internet guna membuat panggilan API, tempat titik akhir API AWS berada. Layanan AWS lain menggunakan titik akhir VPC di dalam Amazon VPC Anda. Banyak layanan AWS, termasuk Amazon S3 dan Amazon DynamoDB, yang mendukung titik akhir VPC. Selain itu, teknologi ini telah digeneralisasi dalam AWS PrivateLink. Sebaiknya gunakan pendekatan ini untuk mengakses layanan AWS, layanan pihak ketiga, dan

layanan milik Anda yang di-host di VPC lain secara aman. Semua lalu lintas jaringan di AWS PrivateLink tetap dalam jalur utama AWS global dan tidak akan melintasi internet. Konektivitas hanya dapat diinisiasi oleh konsumen layanan, bukan oleh penyedia layanan. Dengan AWS PrivateLink untuk akses layanan eksternal, Anda dapat menciptakan VPC terisolasi tanpa akses internet dan membantu melindungi VPC Anda dari vektor ancaman eksternal. Layanan pihak ketiga dapat menggunakan AWS PrivateLink agar konsumen dapat terhubung ke layanan dari VPC mereka melalui alamat IP privat. Untuk aset VPC yang perlu membuat koneksi keluar ke internet, ini dapat ditetapkan khusus keluar (satu jalur) melalui gateway NAT yang dikelola AWS, gateway internet khusus keluar, atau proksi web yang Anda buat dan kelola.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Kontrol lalu lintas jaringan di VPC: Implementasikan praktik terbaik VPC untuk mengontrol lalu lintas.
  - Keamanan Amazon VPC
  - titik akhir VPC
  - Grup keamanan Amazon VPC
  - ACL jaringan
- Kontrol lalu lintas di edge: Implementasikan layanan edge, seperti Amazon CloudFront, untuk memberikan lapisan perlindungan tambahan dan fitur lainnya.
  - Kasus penggunaan Amazon CloudFront
  - AWS Global Accelerator
  - Firewall Aplikasi Web AWS (AWS WAF)
  - Amazon Route 53
  - Perutean Masuk Amazon VPC
- Kontrol lalu lintas jaringan privat: Implementasikan layanan yang melindungi lalu lintas privat untuk beban kerja Anda.
  - Peering Amazon VPC
  - Layanan Titik Akhir Amazon VPC (AWS PrivateLink)
  - Amazon VPC Transit Gateway
  - AWS Direct Connect
  - AWS Site-to-Site VPN

- AWS Client VPN
- Amazon S3 Access Points

Sumber daya

Dokumen terkait:

- AWS Firewall Manager
- Amazon Inspector
- Mulai menggunakan AWS WAF

Video terkait:

- Arsitektur referensi AWS Transit Gateway untuk banyak VPC
- Perlindungan dan Akselerasi Aplikasi dengan Amazon CloudFront, AWS WAF, dan AWS Shield

Contoh terkait:

- Lab: Deployment Otomatis VPC

SEC05-BP03 Mengotomatiskan perlindungan jaringan

Otomatiskan mekanisme perlindungan untuk memberikan jaringan perlindungan mandiri berdasarkan deteksi anomali dan kecerdasan ancaman. Misalnya, deteksi gangguan dan alat pencegahan yang dapat beradaptasi dengan ancaman masa kini serta mengurangi dampaknya. Firewall aplikasi web adalah contoh tempat Anda mengotomatiskan perlindungan jaringan, misalnya, dengan menggunakan solusi Otomatisasi Keamanan AWS WAF (https://github.com/awslabs/aws-waf-security-automations) untuk secara otomatis memblokir permintaan yang berasal dari alamat IP yang terkait dengan penyebab ancaman yang diketahui.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Otomatiskan perlindungan untuk lalu lintas berbasis web: AWS menawarkan solusi yang menggunakan AWS CloudFormation untuk secara otomatis melakukan deployment rangkaian aturan AWS WAF yang didesain untuk memfilter serangan berbasis web yang umum. Pengguna

dapat memilih dari fitur perlindungan sebelum dikonfigurasi yang menentukan aturan yang disertakan dalam daftar kontrol akses web (web ACL) AWS WAF.

  - Otomatisasi keamanan AWS WAF
- Pertimbangkan solusi AWS Partner: Partner AWS menawarkan ratusan produk industri terkemuka yang setara, serupa, atau dapat diintegrasikan dengan kontrol yang sudah ada di lingkungan on-premise Anda. Produk-produk ini melengkapi layanan AWS untuk memungkinkan Anda melakukan deployment arsitektur keamanan yang menyeluruh dan pengalaman yang lancar di seluruh lingkungan cloud dan on-premise Anda.
  - Keamanan infrastruktur

Sumber daya

Dokumen terkait:

- AWS Firewall Manager
- Amazon Inspector
- Keamanan Amazon VPC
- Mulai menggunakan AWS WAF

Video terkait:

- AWS Transit Gateway reference architectures for many VPCs
- Application Acceleration and Protection with Amazon CloudFront, AWS WAF, and AWS Shield

Contoh terkait:

- Lab: Deployment Otomatis VPC

SEC05-BP04 Mengimplementasikan inspeksi dan perlindungan

Inspeksi dan filter lalu lintas Anda di setiap lapisan. Anda dapat melakukan inspeksi terhadap konfigurasi VPC untuk potensi akses yang tidak diinginkan menggunakan VPC Network Access Analyzer. Anda dapat menentukan persyaratan akses jaringan Anda serta mengidentifikasi jalur jaringan yang berpotensi tidak memenuhi syarat tersebut. Untuk komponen transaksi melalui protokol berbasis HTTP, firewall aplikasi web dapat membantu melindungi dari serangan yang umum. AWS WAF adalah firewall aplikasi web yang memungkinkan Anda untuk memantau dan

memblokir permintaan HTTP sesuai dengan aturan yang dapat Anda konfigurasikan yang diteruskan ke API Amazon API Gateway, Amazon CloudFront, atau Application Load Balancer. Untuk mulai menggunakan AWS WAF, Anda dapat memulai dengan Peraturan yang Dikelola AWS yang digabungkan dengan milik Anda sendiri, atau gunakan integrasi partner yang ada..

Untuk mengelola AWS WAF, perlindungan AWS Shield Advanced, dan grup keamanan Amazon VPC di seluruh AWS Organizations, Anda dapat menggunakan AWS Firewall Manager. Hal ini memungkinkan Anda untuk mengonfigurasikan dan mengelola aturan firewall di seluruh akun dan aplikasi Anda secara terpusat, sehingga lebih mudah untuk menskalakan penerapan aturan umum. Penerapan ini juga memungkinkan Anda merespons serangan dengan cepat, menggunakan AWS Shield Advanced, atau solusi yang dapat secara otomatis memblokir permintaan yang tidak diinginkan ke aplikasi web Anda. Firewall Manager juga menerapkan AWS Network Firewall. AWS Network Firewall adalah layanan terkelola yang menggunakan mesin aturan untuk memberi Anda kontrol fine-grained terhadap lalu lintas jaringan stateful dan stateless. Layanan ini mendukung spesifikasi sistem perlindungan gangguan (IPS) sumber terbuka yang sesuai dengan aturan Suricata untuk aturan yang membantu melindungi beban kerja Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Rendah

Panduan implementasi

- Konfigurasikan Amazon GuardDuty: GuardDuty adalah layanan pendeteksi ancaman yang secara terus-menerus memantau aktivitas berbahaya dan perilaku yang tidak diotorisasi untuk melindungi Akun AWS dan beban kerja Anda. Aktifkan GuardDuty dan konfigurasikan peringatan yang diotomatiskan.
  - Amazon GuardDuty
  - Lab: Deployment Otomatis Kontrol Deteksi
- Konfigurasikan Log Alur cloud privat virtual (VPC): Log Alur VPC adalah fitur yang memungkinkan Anda untuk mendokumentasikan informasi tentang lalu lintas IP ke dan dari antarmuka jaringan di VPC Anda. Data log alur dapat dipublikasikan ke Amazon CloudWatch Logs dan Amazon Simple Storage Service (Amazon S3). Setelah Anda membuat log alur, Anda dapat mengambil dan melihat datanya di lokasi tujuan yang telah dipilih.
- Pertimbangkan traffic mirroring VPC: Traffic mirroring adalah fitur Amazon VPC yang dapat Anda gunakan untuk menyalin lalu lintas jaringan dari antarmuka jaringan elastis instans Amazon Elastic Compute Cloud (Amazon EC2) dan mengirimnya ke alat pemantauan dan keamanan luar jaringan untuk inspeksi konten, pemantauan ancaman, dan pemecahan masalah.
  - Traffic mirroring VPC

Sumber daya

Dokumen terkait:

- AWS Firewall Manager
- Amazon Inspector
- Keamanan Amazon VPC
- Mulai menggunakan AWS WAF

Video terkait:

- Arsitektur referensi AWS Transit Gateway untuk banyak VPC
- Perlindungan dan Akselerasi Aplikasi dengan Amazon CloudFront, AWS WAF, dan AWS Shield

Contoh terkait:

- Lab: Deployment Otomatis VPC

SEC 6. Bagaimana cara melindungi sumber daya komputasi Anda?

Sumber daya komputasi di beban kerja Anda memerlukan beberapa lapisan pertahanan untuk membantu melindungi dari ancaman eksternal dan internal. Sumber daya komputasi meliputi instans EC2, kontainer, fungsi AWS Lambda, layanan basis data, perangkat IoT, dan banyak lagi.

Praktik terbaik

- SEC06-BP01 Melakukan manajemen kerentanan
- SEC06-BP02 Mengurangi permukaan serangan
- SEC06-BP03 Mengimplementasikan layanan terkelola
- SEC06-BP04 Mengotomatiskan perlindungan komputasi
- SEC06-BP05 Memberikan kemampuan melakukan tindakan dari jarak jauh
- SEC06-BP06 Memvalidasi integritas perangkat lunak

SEC06-BP01 Melakukan manajemen kerentanan

Seringlah memindai dan melakukan patching kerentanan pada kode, dependensi, dan infrastruktur Anda untuk membantu mencegah ancaman baru.

Hasil yang diinginkan: Membuat dan memelihara program manajemen kerentanan: Memindai dan melakukan patch sumber daya secara rutin, seperti instans Amazon EC2, kontainer Amazon Elastic Container Service (Amazon ECS), dan beban kerja Amazon Elastic Kubernetes Service (Amazon EKS). Mengonfigurasi jendela pemeliharaan untuk sumber daya yang dikelola AWS, seperti basis data Amazon Relational Database Service (Amazon RDS). Menggunakan pemindaian kode statis untuk memeriksa kode sumber aplikasi untuk masalah umum. Pertimbangkan uji penetrasi aplikasi web jika organisasi Anda memiliki keterampilan yang diperlukan atau dapat menggunakan bantuan dari luar.

Antipola umum:

- Tidak memiliki program manajemen kerentanan.
- Menjalankan patching sistem tanpa mempertimbangkan tingkat keparahan atau penghindaran risiko.
- Menggunakan perangkat lunak yang sudah lewat tanggal akhir masa pakai (EOL) dari vendor.
- Melakukan deployment kode ke dalam produksi sebelum menganalisis masalah keamanan.

Manfaat menjalankan praktik terbaik ini:

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Program manajemen kerentanan mencakup penilaian keamanan, mengidentifikasi masalah, memprioritaskan, dan menjalankan operasi patch sebagai bagian dari penyelesaian masalah. Otomatisasi adalah kunci agar dapat terus memindai beban kerja untuk menemukan masalah dan paparan jaringan yang tidak diinginkan serta melakukan perbaikan. Mengotomatiskan pembuatan dan pembaruan sumber daya menghemat waktu dan mengurangi risiko kesalahan konfigurasi yang menciptakan masalah lebih lanjut. Program manajemen kerentanan yang dirancang dengan baik juga harus mempertimbangkan pengujian kerentanan selama tahap pengembangan dan deployment siklus hidup perangkat lunak. Mengimplementasikan manajemen kerentanan selama pengembangan dan deployment membantu mengurangi kemungkinan kerentanan dapat masuk ke lingkungan produksi Anda.

Mengimplementasikan program manajemen kerentanan memerlukan pemahaman yang baik terkait Model Tanggung Jawab Bersama AWS dan bagaimana kaitannya dengan beban kerja tertentu. Dalam Model Tanggung Jawab Bersama, AWS bertanggung jawab untuk melindungi infrastruktur AWS Cloud. Infrastruktur ini terdiri dari perangkat keras, perangkat lunak, jaringan, dan fasilitas yang

menjalankan layanan AWS Cloud. Anda bertanggung jawab atas keamanan di cloud, misalnya, data aktual, konfigurasi keamanan, dan tugas manajemen instans Amazon EC2, serta memastikan bahwa klasifikasi dan konfigurasi objek Amazon S3 Anda sudah tepat. Pendekatan terhadap manajemen kerentanan juga dapat bervariasi, bergantung pada layanan yang digunakan. Misalnya, AWS mengelola patching untuk layanan basis data relasional terkelola kami, Amazon RDS, tetapi Anda akan bertanggung jawab atas patching basis data yang di-hosting secara mandiri.

AWS memiliki berbagai layanan untuk membantu program manajemen kerentanan. Amazon Inspector terus memindai beban kerja AWS untuk menemukan masalah perangkat lunak dan akses jaringan yang tidak diinginkan. AWS Systems Manager Patch Manager membantu mengelola patching di seluruh instans Amazon EC2. Amazon Inspector dan Systems Manager dapat dilihat di AWS Security Hub CSPM, layanan manajemen postur keamanan cloud yang membantu mengotomatiskan pemeriksaan keamanan AWS dan memusatkan peringatan keamanan.

Amazon CodeGuru dapat membantu mengidentifikasi potensi masalah di aplikasi Java dan Python menggunakan analisis kode statis.

Langkah implementasi

- Konfigurasikan Amazon Inspector: Amazon Inspector secara otomatis mendeteksi instans Amazon EC2 yang baru saja diluncurkan, fungsi Lambda, dan gambar kontainer yang dimasukkan ke Amazon ECR serta segera memindainya untuk menemukan masalah perangkat lunak, potensi kecacatan, dan paparan jaringan yang tidak diinginkan.
- Pindai kode sumber: Pindai pustaka dan dependensi untuk menemukan masalah dan kecacatan. Amazon CodeGuru dapat memindai dan memberikan rekomendasi untuk memperbaiki masalah keamanan umum untuk aplikasi Java dan Python. OWASP Foundation menerbitkan daftar Alat Analisis Kode Sumber (juga disebut sebagai alat SAST).
- Implementasikan mekanisme untuk memindai dan melakukan patching lingkungan yang ada, serta pemindaian sebagai bagian dari proses pembuatan jalur CI/CD: Implementasikan mekanisme untuk memindai dan melakukan patching masalah di dependensi dan sistem operasi untuk membantu melindungi dari ancaman baru. Jalankan mekanisme tersebut secara rutin. Penting untuk memahami manajemen kerentanan perangkat lunak saat Anda ingin menerapkan patch atau mengatasi masalah perangkat lunak. Prioritaskan perbaikan potensi masalah keamanan dengan menanamkan penilaian kerentanan lebih awal ke dalam jalur integrasi berkelanjutan/pengiriman berkelanjutan (CI/CD). Pendekatan dapat bervariasi berdasarkan layanan AWS yang digunakan. Untuk memeriksa potensi masalah terkait perangkat lunak yang dijalankan di instans Amazon EC2, tambahkan Amazon Inspector ke jalur untuk memberikan peringatan dan menghentikan proses pembuatan jika masalah atau potensi kecacatan terdeteksi. Amazon Inspector memantau

sumber daya secara berkelanjutan. Anda juga dapat menggunakan produk sumber terbuka seperti Pemeriksaan Dependensi OWASP, Snyk, OpenVAS, manajer paket, dan alat AWS Partner untuk manajemen kerentanan.

- Gunakan AWS Systems Manager: Anda bertanggung jawab atas manajemen patch sumber daya AWS Anda, termasuk instans Amazon Elastic Compute Cloud (Amazon EC2), Amazon Machine Image (AMI), dan sumber daya komputasi lainnya. AWS Systems Manager Patch Manager mengotomatiskan proses patching instans terkelola dengan pembaruan terkait keamanan dan jenis pembaruan lainnya. Patch Manager dapat digunakan untuk menerapkan patch pada instans Amazon EC2 untuk sistem operasi dan aplikasi, termasuk aplikasi Microsoft, paket layanan Windows, dan pembaruan versi minor untuk instans berbasis Linux. Selain Amazon EC2, Patch Manager juga dapat digunakan untuk melakukan patching server on-premise.

  Untuk daftar sistem operasi yang didukung, lihat Sistem operasi yang didukung di Panduan Pengguna Systems Manager. Anda dapat memindai instans untuk hanya melihat laporan patch yang hilang, atau Anda dapat memindai dan secara otomatis memasang semua patch yang hilang.
- Gunakan AWS Security Hub CSPM: Security Hub CSPM menyediakan tampilan komprehensif untuk status keamanan Anda di AWS. Program ini mengumpulkan data keamanan di berbagai layanan AWS dan mengumpulkan temuan tersebut dalam format standar, memungkinkan Anda untuk memprioritaskan temuan keamanan di seluruh layanan AWS.
- Gunakan AWS CloudFormation: AWS CloudFormation adalah layanan infrastruktur sebagai kode (IaC) yang dapat membantu manajemen kerentanan dengan mengotomatiskan deployment sumber daya dan menstandarkan arsitektur sumber daya di berbagai akun dan lingkungan.

Sumber daya

Dokumen terkait:

- AWS Systems Manager
- Gambaran Umum Keamanan AWS Lambda
- Amazon CodeGuru
- Manajemen Kerentanan Otomatis yang Ditingkatkan untuk Beban Kerja Cloud dengan Amazon Inspector Baru
- Mengotomatiskan manajemen kerentanan dan perbaikan di AWS menggunakan Amazon Inspector dan AWS Systems Manager – Bagian 1

Video terkait:

- Mengamankan Layanan Kontainer dan Nirserver
- Praktik terbaik keamanan untuk layanan metadata instans Amazon EC2

SEC06-BP02 Mengurangi permukaan serangan

Kurangi paparan Anda ke akses yang tidak diinginkan dengan penguatan sistem operasi serta meminimalkan penggunaan komponen, pustaka, dan layanan sekali pakai eksternal. Mulai dengan mengurangi komponen yang tidak digunakan untuk seluruh beban kerja, baik itu paket sistem operasi atau aplikasi, untuk beban kerja berbasis Amazon Elastic Compute Cloud (Amazon EC2), maupun modul perangkat lunak eksternal pada kode Anda. Anda dapat menemukan beberapa panduan konfigurasi penguatan dan keamanan untuk sistem operasi umum dan perangkat lunak server. Misalnya, Anda dapat memulai dengan Pusat Keamanan Internet dan lakukan iterasi.

Di Amazon EC2, Anda dapat membuat Amazon Machine Image (AMI) Anda sendiri, yang telah di-patch dan diperkuat, untuk membantu memenuhi persyaratan keamanan spesifik bagi organisasi Anda. Patch dan kontrol keamanan lain yang Anda terapkan di AMI efektif pada saat dibuat—sifatnya tidak dinamis kecuali Anda memodifikasinya setelah peluncuran, misalnya, dengan AWS Systems Manager.

Anda dapat menyederhanakan proses membangun AMI yang aman dengan EC2 Image Builder. EC2 Image Builder secara signifikan mengurangi upaya yang diperlukan untuk membuat dan memelihara image emas tanpa otomatisasi penulisan dan pemeliharaan. Ketika pembaruan perangkat lunak sudah tersedia, Image Builder secara otomatis memproduksi image baru tanpa mengharuskan pengguna memulai pembangunan image secara manual. EC2 Image Builder memungkinkan Anda untuk memvalidasi fungsionalitas dan keamanan image Anda dengan mudah sebelum menggunakannya dalam produksi dengan pengujian yang disediakan AWS serta pengujian Anda sendiri. Anda juga dapat menerapkan pengaturan keamanan yang disediakan AWS untuk mengamankan image Anda secara lebih lanjut untuk memenuhi kriteria keamanan internal. Misalnya, Anda dapat memproduksi image yang sesuai dengan standar Security Technical Implementation Guide (STIG) menggunakan templat yang disediakan oleh AWS.

Menggunakan alat analisis kode statis pihak ketiga, Anda dapat mengidentifikasi masalah keamanan umum seperti batas input fungsi yang tidak diperiksa, dan juga kelemahan dan paparan umum (common vulnerabilities and exposures, CVE) yang dapat diterapkan. Anda dapat menggunakan Amazon CodeGuru untuk bahasa yang didukung. Alat pemeriksaan dependensi juga dapat digunakan untuk menentukan apakah pustaka yang ditautkan kode Anda merupakan versi terbaru, bebas dari CVE, serta memiliki syarat perizinan yang memenuhi persyaratan kebijakan perangkat lunak Anda.

Menggunakan Amazon Inspector, Anda dapat melakukan penilaian konfigurasi terhadap instans Anda untuk CVE yang diketahui, menilai tolok ukur keamanan, serta mengotomatiskan pemberitahuan kecacatan. Amazon Inspector berjalan di instans produksi atau di jalur build, serta memberi tahu pengembang dan teknisi ketika ada temuan. Anda dapat mengakses temuan secara terprogram dan mengarahkan tim Anda ke backlog dan sistem pelacakan bug. EC2 Image Builder dapat digunakan untuk memelihara image server (AMI) dengan patching otomatis, penegakan kebijakan keamanan yang disediakan oleh AWS, serta kustomisasi lainnya. Ketika menggunakan kontainer, terapkan Pemindaian Image ECR pada jalur build Anda dan secara rutin terhadap repositori image Anda untuk mencari CVE dalam kontainer Anda.

Ketika Amazon Inspector dan alat lainnya mengidentifikasi dengan efektif konfigurasi dan CVE apa pun yang ada, metode lain diperlukan untuk menguji beban kerja Anda pada tingkat aplikasi. Fuzzing adalah metode yang terkenal untuk menemukan bug menggunakan otomatisasi untuk menginjeksi data dengan kesalahan bentuk ke bidang input dan area lain pada aplikasi Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

- Perkuat sistem operasi: Konfigurasikan sistem operasi agar memenuhi praktik terbaik.
  - Mengamankan Amazon Linux
  - Mengamankan Microsoft Windows Server
- Perkuat sumber daya terkontainerisasi: Konfigurasikan sumber daya terkontainerisasi untuk memenuhi praktik terbaik keamanan.
- Terapkan praktik terbaik AWS Lambda.
  - Praktik terbaik AWS Lambda

Sumber daya

Dokumen terkait:

- AWS Systems Manager
- Mengganti Host Bastion dengan Amazon EC2 Systems Manager
- Gambaran Umum Keamanan AWS Lambda

Video terkait:

- Menjalankan beban kerja dengan keamanan tinggi di Amazon EKS
- Mengamankan Layanan Kontainer dan Nirserver
- Praktik terbaik keamanan untuk layanan metadata instans Amazon EC2

Contoh terkait:

- Lab: Deployment Firewall Aplikasi Web secara Otomatis

SEC06-BP03 Mengimplementasikan layanan terkelola

Implementasikan layanan yang mengelola sumber daya seperti Amazon Relational Database Service (Amazon RDS), AWS Lambda, dan Amazon Elastic Container Service (Amazon ECS), untuk mengurangi tugas pemeliharaan keamanan sebagai bagian dari model tanggung jawab bersama. Contohnya, Amazon RDS membantu Anda mengatur, mengoperasikan, dan menskalakan basis data relasional, mengotomatiskan tugas administrasi seperti penyediaan perangkat keras, pengaturan basis data, patching, dan pencadangan. Ini berarti Anda memiliki lebih banyak waktu luang untuk berkonsentrasi mengamankan aplikasi Anda dengan cara lain yang disebutkan dalam AWS Well-Architected Framework. Lambda memungkinkan Anda menjalankan kode tanpa menyediakan atau mengelola server, sehingga Anda hanya perlu fokus pada konektivitas, permintaan, dan keamanan di tingkat kode–bukan infrastruktur atau sistem operasi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

- Jelajahi layanan yang tersedia: Jelajahi, uji, dan terapkan layanan yang mengelola sumber daya, seperti Amazon RDS, AWS Lambda, dan Amazon ECS.

Sumber daya

Dokumen terkait:

- Situs web AWS
- AWS Systems Manager
- Mengganti Host Bastion dengan Amazon EC2 Systems Manager
- Gambaran Umum Keamanan AWS Lambda

Video terkait:

- Menjalankan beban kerja dengan keamanan tinggi di Amazon EKS
- Mengamankan Layanan Kontainer dan Nirserver
- Praktik terbaik keamanan untuk layanan metadata instans Amazon EC2

Contoh terkait:

- Lab: Permohonan Sertifikat Publik AWS Certificate Manager

SEC06-BP04 Mengotomatiskan perlindungan komputasi

Otomatiskan mekanisme komputasi protektif Anda termasuk manajemen kelemahan, pengurangan permukaan serangan, serta manajemen sumber daya. Otomatisasi akan membantu Anda menginvestasikan waktu untuk mengamankan aspek-aspek lain dalam beban kerja Anda, dan mengurangi risiko kesalahan manusia.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Otomatiskan manajemen konfigurasi: Tegakkan dan validasi konfigurasi keamanan secara otomatis menggunakan layanan atau alat manajemen konfigurasi.
  - AWS Systems Manager
  - AWS CloudFormation
  - Lab: Deployment otomatis VPC
  - Lab: Deployment otomatis aplikasi web EC2

- Otomatiskan patching instans Amazon Elastic Compute Cloud (Amazon EC2): AWS Systems Manager Patch Manager mengotomatiskan proses patching instans terkelola dengan pembaruan terkait keamanan dan jenis pembaruan lainnya. Anda dapat menggunakan Patch Manager guna menerapkan patch untuk sistem operasi maupun aplikasi.
  - AWS Systems Manager Patch Manager
  - Patching multiakun dan multiwilayah terpusat dengan AWS Systems Manager Automation

- Implementasikan deteksi dan pencegahan intrusi: Implementasikan alat deteksi dan pencegahan intrusi untuk memantau dan menghentikan aktivitas berbahaya pada instans.
- Pertimbangkan solusi AWS Partner: Partner AWS menawarkan ratusan produk industri terkemuka yang setara, serupa, atau dapat diintegrasikan dengan kontrol yang sudah ada di lingkungan on-premise Anda. Produk-produk ini melengkapi layanan AWS untuk memungkinkan Anda melakukan deployment arsitektur keamanan yang menyeluruh dan pengalaman yang lancar di seluruh lingkungan cloud dan on-premise Anda.
  - Keamanan infrastruktur

Sumber daya

Dokumen terkait:

- AWS CloudFormation
- AWS Systems Manager
- AWS Systems Manager Patch Manager
- Patching multiakun dan multiwilayah terpusat dengan AWS Systems Manager Automation
- Keamanan infrastruktur
- Mengganti Host Bastion dengan Amazon EC2 Systems Manager
- Gambaran Umum Keamanan AWS Lambda

Video terkait:

- Running high-security workloads on Amazon EKS
- Securing Serverless and Container Services
- Security best practices for the Amazon EC2 instance metadata service

Contoh terkait:

- Lab: Deployment Firewall Aplikasi Web secara Otomatis
- Lab: Deployment otomatis aplikasi web EC2

SEC06-BP05 Memberikan kemampuan melakukan tindakan dari jarak jauh

Menghapus kemampuan akses interaktif dapat mengurangi risiko kesalahan akibat kelalaian manusia, dan kemungkinan dibutuhkannya manajemen atau konfigurasi manual. Misalnya, gunakan alur kerja manajemen perubahan untuk melakukan deployment instans Amazon Elastic Compute Cloud (Amazon EC2) menggunakan infrastruktur sebagai kode, selanjutnya kelola instans Amazon EC2 menggunakan alat seperti AWS Systems Manager, bukannya menerapkan akses langsung melalui host bastion. AWS Systems Managerdapat mengotomatiskan berbagai tugas pemeliharaan dan deployment, menggunakan fitur yang mencakup alur kerja otomatisasi, dokumen (buku pedoman), dan run command (jalankan perintah). Tumpukan AWS CloudFormation dibangun dari pipeline dan dapat mengotomatiskan tugas manajemen serta deployment infrastruktur Anda tanpa menggunakan Konsol Manajemen AWS atau API secara langsung.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Rendah

Panduan implementasi

- Ganti akses konsol: Ganti akses konsol (SSH atau RDP) ke instans dengan AWS Systems Manager Run Command untuk mengotomatiskan tugas manajemen.

- AWS Systems Manager Run Command

Sumber daya

Dokumen terkait:

- AWS Systems Manager
- AWS Systems Manager Run Command
- Mengganti Host Bastion dengan Amazon EC2 Systems Manager
- Gambaran Umum Keamanan AWS Lambda

Video terkait:

- Jalankan beban kerja dengan keamanan tinggi di Amazon EKS
- Mengamankan Layanan Kontainer dan Nirserver
- Praktik terbaik keamanan untuk layanan metadata instans Amazon EC2

Contoh terkait:

- Lab: Deployment Otomatis Firewall Aplikasi Web

SEC06-BP06 Memvalidasi integritas perangkat lunak

Implementasikan mekanisme (misalnya, penandatanganan kode) untuk memvalidasi bahwa perangkat lunak, kode, dan pustaka yang digunakan di beban kerja berasal dari sumber tepercaya dan belum pernah dimodifikasi. Misalnya, Anda harus memverifikasi sertifikat penandatanganan kode biner dan skrip untuk mengonfirmasi penulis, serta memastikan sertifikat tersebut belum pernah dimodifikasi sejak dibuat oleh penulisnya. AWS Signer dapat membantu memastikan kepercayaan dan integritas kode Anda dengan mengelola secara terpusat siklus hidup penandatanganan kode, termasuk sertifikat penandatanganan serta kunci privat dan publik. Anda dapat mempelajari cara menggunakan pola tingkat lanjut dan praktik terbaik penandatanganan kode dengan AWS Lambda. Selain itu, checksum perangkat lunak yang Anda unduh, dibandingkan dengan checksum dari penyedia, dapat membantu memastikan bahwa perangkat belum pernah dimodifikasi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Rendah

Panduan implementasi

- Selidiki mekanisme: Penandatanganan kode adalah sebuah mekanisme yang dapat digunakan untuk memvalidasi integritas perangkat lunak.
  - NIST: Pertimbangan Keamanan untuk Penandatanganan Kode

Sumber daya

Dokumen terkait:

- AWS Signer
- Baru – Penandatanganan Kode, Kontrol Integritas dan Kepercayaan untuk AWS Lambda

Perlindungan data

Pertanyaan

- SEC 7. Bagaimana cara mengklasifikasikan data Anda?
- SEC 8. Bagaimana cara melindungi data diam Anda?

- SEC 9. Bagaimana cara melindungi data bergerak Anda?

SEC 7. Bagaimana cara mengklasifikasikan data Anda?

Klasifikasi memberikan cara untuk mengategorikan data, berdasarkan tingkat kekritisan dan sensitivitas untuk membantu Anda menentukan kontrol retensi dan perlindungan yang sesuai.

Praktik terbaik

- SEC07-BP01 Mengidentifikasi data dalam beban kerja Anda
- SEC07-BP02 Menentukan kontrol perlindungan data
- SEC07-BP03 Mengotomatisasi identifikasi dan klasifikasi
- SEC07-BP04 Menentukan manajemen siklus hidup data

SEC07-BP01 Mengidentifikasi data dalam beban kerja Anda

Penting untuk memahami jenis dan klasifikasi data yang sedang diproses oleh beban kerja Anda, proses bisnis terkait, tempat penyimpanan data, dan pemilik data. Anda juga harus memahami persyaratan hukum dan kepatuhan yang berlaku dari beban kerja Anda, dan kontrol data apa yang perlu diterapkan. Mengidentifikasi data adalah langkah pertama dalam perjalanan klasifikasi data.

Manfaat menjalankan praktik terbaik ini:

Dengan klasifikasi data, pemilik beban kerja dapat mengidentifikasi lokasi penyimpanan data sensitif dan menentukan bagaimana data tersebut dapat diakses dan dibagikan.

Klasifikasi data bertujuan untuk menjawab pertanyaan berikut:

- Jenis data apa yang Anda miliki?

  Data dapat berupa:

  - Kekayaan intelektual (IP), seperti rahasia dagang, paten, atau perjanjian kontrak.
  - Informasi kesehatan yang dilindungi (PHI), seperti rekam medis yang berisi informasi riwayat kesehatan seseorang.
  - Informasi pengenal pribadi (PII), seperti nama, alamat, tanggal lahir, dan nomor registrasi atau ID nasional.
  - Data kartu kredit, seperti Nomor Rekening Utama (PAN), nama pemilik kartu, tanggal kedaluwarsa, dan nomor kode layanan.

    - Di mana data sensitif disimpan?
    - Siapa yang dapat mengakses, mengubah, dan menghapus data?
    - Memahami izin pengguna adalah hal yang penting dalam menghindari potensi kesalahan penanganan data.
- Siapa yang dapat melakukan operasi membuat, membaca, memperbarui, dan menghapus (CRUD)?
    - Antisipasi potensi peningkatan hak akses dengan memahami siapa yang dapat mengelola izin ke data.
- Dampak bisnis seperti apa yang mungkin terjadi jika data secara tidak sengaja diungkapkan, diubah, atau dihapus?
    - Pahami konsekuensi risiko jika data diubah, dihapus, atau diungkapkan secara tidak sengaja.

Dengan mengetahui jawaban atas pertanyaan ini, Anda dapat mengambil tindakan berikut:

- Mengurangi cakupan data sensitif (seperti jumlah lokasi data sensitif) dan membatasi akses ke data sensitif hanya untuk pengguna yang disetujui.
- Memahami berbagai jenis data sehingga Anda dapat menerapkan mekanisme dan teknik perlindungan data yang sesuai, seperti enkripsi, pencegahan kehilangan data, serta manajemen identitas dan akses.
- Mengoptimalkan biaya dengan memberikan tujuan kontrol yang tepat untuk data.
- Tanpa ragu menjawab pertanyaan dari regulator dan auditor mengenai jenis dan jumlah data, dan bagaimana data dengan sensitivitas yang berbeda dipisahkan dari satu sama lain.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Klasifikasi data adalah tindakan mengidentifikasi sensitivitas data. Aktivitas ini mungkin melibatkan pemberian tag untuk memudahkan pencarian dan pelacakan data. Klasifikasi data juga mengurangi duplikasi data, yang dapat membantu mengurangi biaya penyimpanan dan pencadangan sekaligus mempercepat proses pencarian.

Gunakan layanan seperti Amazon Macie untuk mengotomatiskan penemuan dan klasifikasi data sensitif dalam skala besar. Layanan lain, seperti Amazon EventBridge dan AWS Config, dapat digunakan untuk mengotomatiskan perbaikan masalah keamanan data, seperti bucket Amazon

Simple Storage Service (Amazon S3) tidak terenkripsi dan volume EBS Amazon EC2 atau sumber daya data yang tidak diberi tag. Untuk daftar lengkap integrasi layanan AWS, lihat dokumentasi EventBridge.

Mendeteksi PII dalam data yang tidak terstruktur seperti email pelanggan, tiket dukungan, ulasan produk, dan media sosial dapat dilakukan menggunakan Amazon Comprehend, yang merupakan layanan pemrosesan bahasa alami (NLP) yang menggunakan machine learning (ML) untuk menemukan wawasan dan hubungan seperti orang, tempat, sentimen, serta topik dalam teks yang tidak terstruktur. Untuk daftar layanan AWS yang dapat membantu identifikasi data, lihat Teknik umum untuk mendeteksi data PHI dan PII menggunakan layanan AWS.

Metode lain yang mendukung klasifikasi dan perlindungan data adalah Pemberian tag sumber daya AWS. Pemberian tag memungkinkan Anda menetapkan metadata ke sumber daya AWS yang dapat digunakan untuk mengelola, mengidentifikasi, mengatur, mencari, dan memfilter sumber daya.

Dalam beberapa kasus, Anda mungkin memilih untuk memberi tag seluruh sumber daya (seperti bucket S3), khususnya saat beban kerja atau layanan tertentu diharapkan menyimpan proses atau transmisi dari klasifikasi data yang sudah umum.

Jika perlu, Anda dapat memberi tag bucket S3 dan bukan pada objek individu untuk kemudahan administrasi dan pemeliharaan keamanan.

Langkah implementasi

Deteksi data sensitif dalam Amazon S3:

1. Sebelum memulai, pastikan Anda memiliki izin yang sesuai untuk mengakses konsol Amazon Macie dan operasi API. Untuk detail tambahan, lihat Mulai menggunakan Amazon Macie.
2. Gunakan Amazon Macie untuk menjalankan penemuan data otomatis saat data sensitif berada di Amazon S3.
   - Gunakan panduan Mulai Menggunakan Amazon Macie untuk mengonfigurasi repositori untuk hasil penemuan data sensitif dan membuat tugas penemuan untuk data sensitif.
   - Cara menggunakan Amazon Macie untuk pratinjau data sensitif di bucket S3.

     Secara default, Macie menganalisis objek menggunakan set pengidentifikasi data terkelola yang kami rekomendasikan untuk penemuan data sensitif otomatis. Anda dapat menyesuaikan analisis dengan mengonfigurasi Macie untuk menggunakan pengidentifikasi data terkelola tertentu, pengidentifikasi data kustom, dan daftar yang diizinkan saat melakukan penemuan data sensitif otomatis untuk akun atau organisasi Anda. Anda dapat menyesuaikan cakupan analisis

dengan mengecualikan bucket tertentu (misalnya, bucket S3 yang biasanya menyimpan data pencatatan log AWS).

3. Untuk mengonfigurasi dan menggunakan penemuan data sensitif otomatis, lihat Menjalankan penemuan data sensitif otomatis dengan Amazon Macie.
4. Anda juga dapat mempertimbangkan Penemuan Data Otomatis untuk Amazon Macie.

Deteksi data sensitif dalam Amazon RDS:

Untuk informasi lebih lanjut tentang penemuan data di basis data Amazon Relational Database Service (Amazon RDS), lihat Mengaktifkan klasifikasi data untuk basis data Amazon RDS dengan Macie.

Deteksi data sensitif dalam DynamoDB:

- Mendeteksi data sensitif di DynamoDB dengan Macie menjelaskan cara menggunakan Amazon Macie untuk mendeteksi data sensitif di tabel Amazon DynamoDB dengan mengekspor data ke Amazon S3 untuk pemindaian.

Solusi Partner AWS:

- Pertimbangkan untuk menggunakan ekstensi AWS Partner Network. Partner AWS memiliki alat ekstensi dan kerangka kerja kepatuhan yang terintegrasi langsung dengan layanan AWS. Partner dapat memberikan solusi tata kelola dan kepatuhan yang disesuaikan untuk membantu memenuhi kebutuhan organisasi Anda.
- Untuk solusi kustom dalam klasifikasi data, lihat Tata kelola data dalam peraturan dan persyaratan kepatuhan.

Anda dapat secara otomatis menerapkan standar pemberian tag yang diadopsi oleh organisasi Anda dengan membuat dan melakukan deployment kebijakan menggunakan AWS Organizations. Kebijakan tag memungkinkan Anda menentukan aturan yang menentukan nama kunci yang valid dan nilai apa yang valid untuk setiap kunci. Anda dapat memilih untuk hanya memantau, yang dapat Anda gunakan untuk mengevaluasi dan menghapus tag yang ada. Setelah tag mematuhi standar yang dipilih, Anda dapat mengaktifkan penerapan di kebijakan tag untuk mencegah pembuatan tag yang tidak patuh. Untuk detail lebih lanjut, lihat Mengamankan tag sumber daya yang digunakan untuk otorisasi menggunakan kebijakan kontrol layanan di AWS Organizations dan contoh kebijakan di mencegah perubahan tag selain oleh pengguna utama yang sah.

- Untuk mulai menggunakan kebijakan tag di AWS Organizations, sangat disarankan untuk mengikuti alur kerja di Mulai menggunakan kebijakan tag sebelum melanjutkan ke kebijakan tag yang lebih tinggi. Memahami efek pelampiran kebijakan tag sederhana ke satu akun sebelum menerapkannya ke seluruh unit organisasi (OU) atau organisasi dapat memberikan gambaran akan efek kebijakan tag sebelum Anda menerapkan kepatuhan ke kebijakan tag. Mulai menggunakan kebijakan tag menyediakan tautan ke instruksi untuk tugas terkait kebijakan yang lebih tinggi.
- Pertimbangkan untuk mengevaluasi layanan dan fitur AWS lainnya yang mendukung klasifikasi data, yang tercantum dalam laporan resmi Klasifikasi Data.

Sumber daya

Dokumen terkait:

- Mulai Menggunakan Amazon Macie
- Penemuan data otomatis dengan Amazon Macie
- Mulai menggunakan kebijakan tag
- Mendeteksi entitas PII

Blog terkait:

- Cara menggunakan Amazon Macie untuk pratinjau data sensitif di bucket S3.
- Menjalankan penemuan data sensitif dengan Amazon Macie.
- Teknik umum untuk mendeteksi data PHI dan PII menggunakan Layanan AWS
- Mendeteksi dan mengedit PII menggunakan Amazon Comprehend
- Menggunakan tag sumber daya yang digunakan untuk otorisasi menggunakan kebijakan kontrol layanan di AWS Organizations
- Melakukan klasifikasi data untuk basis data Amazon RDS dengan Macie
- Mendeteksi data sensitif di DynamoDB dengan Macie
-

Video terkait:

- Keamanan yang diarahkan peristiwa menggunakan Amazon Macie

- Amazon Macie untuk perlindungan dan tata kelola data
- Menyaring temuan data sensitif dengan daftar yang diizinkan

SEC07-BP02 Menentukan kontrol perlindungan data

Lindungi data sesuai dengan tingkat klasifikasinya. Contohnya, amankan data dengan klasifikasi publik menggunakan rekomendasi yang relevan sekaligus melindungi data sensitif dengan kontrol tambahan.

Dengan menggunakan tag sumber daya, pisahkan AWSakun berdasarkan sensitivitas (dan berpotensi juga untuk setiap peringatan, enklave, atau komunitas minat), kebijakan IAM, SCP AWS Organizations, AWS Key Management Service (AWS KMS), dan AWS CloudHSM, Anda dapat menentukan dan menerapkan kebijakan Anda untuk klasifikasi dan perlindungan data dengan enkripsi. Contohnya, jika Anda memiliki proyek dengan bucket S3 yang berisi data yang sangat penting atau instans Amazon Elastic Compute Cloud (Amazon EC2) yang memproses data rahasia, Anda dapat menandainya dengan tag `Project=ABC` . Hanya tim langsung Anda yang mengetahui arti kode proyek ini, dan ini menyediakan cara untuk menggunakan kontrol akses berbasis atribut. Anda dapat menentukan tingkatan akses ke kunci enkripsi AWS KMS melalui kebijakan dan pemberian kunci untuk memastikan hanya layanan yang sesuai yang memiliki akses ke konten sensitif melalui mekanisme yang aman. Jika Anda membuat keputusan otorisasi berdasarkan tag, Anda harus memastikan bahwa izin pada tag telah ditentukan dengan tepat menggunakan kebijakan tag di AWS Organizations.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

- Tentukan identifikasi dan skema klasifikasi data Anda: Identifikasi dan klasifikasi data Anda dilakukan untuk menilai potensi dampak dan tipe data yang Anda simpan dan siapa saja yang dapat mengaksesnya.
  - Dokumentasi AWS
- Temukan kontrol AWS yang tersedia: Untuk layanan AWS yang sedang atau akan Anda gunakan, temukan kontrol keamanannya. Banyak layanan memiliki bagian keamanan dalam dokumentasinya.
  - Dokumentasi AWS
- Kenali sumber daya kepatuhan AWS: Kenali sumber daya yang disediakan oleh AWS untuk membantu.

- https://aws.amazon.com/compliance/

Sumber daya

Dokumen terkait:

- Dokumentasi AWS
- Laporan Resmi Klasifikasi Data
- Mulai menggunakan Amazon Macie
- Teks Hilang

Video terkait:

- Memperkenalkan Amazon Macie Baru

SEC07-BP03 Mengotomatisasi identifikasi dan klasifikasi

Otomatisasi identifikasi dan klasifikasi data dapat membantu Anda mengimplementasikan kontrol yang tepat. Menggunakan otomatisasi untuk hal ini, sebagai ganti akses langsung dari orang, dapat mengurangi risiko kesalahan dan eksposur manusia. Anda harus mengevaluasi menggunakan alat, seperti Amazon Macie, yang menggunakan machine learning untuk menemukan, mengelompokkan, dan melindungi data sensitif secara otomatis di AWS. Amazon Macie mengenali data sensitif, seperti informasi pengenal pribadi (PII) atau kekayaan intelektual, dan membekali Anda dengan dasbor dan peringatan yang memberikan visibilitas tentang bagaimana data ini diakses atau dipindahkan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

- Gunakan Inventaris Amazon Simple Storage Service (Amazon S3): Inventaris Amazon S3 adalah salah satu alat yang dapat Anda gunakan untuk mengaudit serta melaporkan replikasi dan status enkripsi objek.
  - Inventaris Amazon S3
- Pertimbangkan Amazon Macie: Amazon Macie menggunakan machine learning untuk secara otomatis menemukan dan mengelompokkan data yang disimpan di Amazon S3.
  - Amazon Macie

Sumber daya

Dokumen terkait:

- Amazon Macie
- Inventaris Amazon S3
- Laporan Resmi Klasifikasi Data
- Mulai menggunakan Amazon Macie

Video terkait:

- Memperkenalkan Amazon Macie Baru

SEC07-BP04 Menentukan manajemen siklus hidup data

Strategi siklus hidup yang ditentukan harus berdasarkan tingkat sensitivitas serta persyaratan hukum dan organisasi. Aspek-aspek yang perlu diperhatikan mencakup durasi mempertahankan data, proses penghancuran data, manajemen akses data, transformasi data, dan berbagi data. Saat memilih metodologi klasifikasi data, seimbangkan ketergunaan dan akses. Anda juga harus mengakomodasi beberapa tingkat akses dan perbedaan untuk mengimplementasikan pendekatan yang aman tetapi masih dapat digunakan untuk masing-masing tingkat. Selalu gunakan pendekatan pertahanan mendalam dan kurangi akses manusia ke data dan mekanisme untuk mentransformasi, menghapus, atau menyalin data. Misalnya, pengguna perlu diautentikasi oleh aplikasi, dan berikan izin akses yang diperlukan untuk melakukan tindakan dari jarak jauh kepada aplikasi, bukan pengguna. Selain itu, pastikan bahwa pengguna berasal dari jalur jaringan tepercaya dan memerlukan akses ke kunci dekripsi. Gunakan alat seperti dasbor dan pelaporan otomatis untuk memberikan informasi data kepada pengguna daripada memberi mereka akses langsung ke data.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Rendah

Panduan implementasi

- Identifikasikan jenis data: Identifikasikan jenis data yang disimpan atau diproses di beban kerja. Data tersebut dapat berupa teks, gambar, basis data biner, dan lainnya.

Sumber daya

Dokumen terkait:

- Laporan Resmi Klasifikasi Data
- Mulai menggunakan Amazon Macie

Video terkait:

- Memperkenalkan Amazon Macie Baru

SEC 8. Bagaimana cara melindungi data diam Anda?

Lindungi data diam dengan mengimplementasikan beberapa kontrol, untuk mengurangi risiko akses tanpa otorisasi atau penanganan yang salah.

Praktik terbaik

- SEC08-BP01 Mengimplementasikan manajemen kunci yang aman
- SEC08-BP02 Menerapkan enkripsi data diam
- SEC08-BP03 Mengotomatiskan perlindungan data diam
- SEC08-BP04 Menerapkan kontrol akses
- SEC08-BP05 Menggunakan mekanisme untuk mencegah orang mengakses data

SEC08-BP01 Mengimplementasikan manajemen kunci yang aman

Manajemen kunci yang aman mencakup penyimpanan, rotasi, kontrol akses, dan pemantauan materi kunci yang diperlukan untuk mengamankan data diam untuk beban kerja Anda.

Hasil yang diinginkan: Mekanisme manajemen kunci yang dapat diskalakan, dapat diulang, dan dapat diotomatiskan. Mekanisme ini harus memberikan kemampuan untuk menerapkan hak akses paling rendah ke materi kunci, memberikan keseimbangan yang tepat antara ketersediaan, kerahasiaan, dan integritas kunci. Akses ke kunci harus dipantau, dan materi kunci dirotasi melalui proses otomatis. Materi kunci tidak boleh diakses oleh identitas manusia.

Antipola umum:

- Akses manusia ke materi kunci yang tidak terenkripsi.
- Membuat algoritma kriptografi kustom.
- Izin yang terlalu luas untuk mengakses materi kunci.

Manfaat menjalankan praktik terbaik ini: Dengan membuat mekanisme manajemen kunci yang aman untuk beban kerja Anda, Anda dapat membantu memberikan perlindungan untuk konten Anda dari akses yang tidak sah. Selain itu, Anda mungkin harus mematuhi persyaratan peraturan untuk mengenkripsi data Anda. Solusi manajemen kunci yang efektif dapat memberikan mekanisme teknis yang selaras dengan peraturan tersebut untuk melindungi materi kunci.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Banyak persyaratan peraturan dan praktik terbaik yang menyertakan enkripsi data diam sebagai kontrol keamanan mendasar. Untuk mematuhi kontrol ini, beban kerja Anda memerlukan mekanisme untuk menyimpan dan mengelola materi kunci yang digunakan untuk mengenkripsi data diam Anda.

AWS menawarkan AWS Key Management Service (AWS KMS) untuk menyediakan penyimpanan yang tahan lama, aman, dan redundan untuk kunci AWS KMS. Banyak layanan AWS terintegrasi dengan AWS KMS untuk mendukung enkripsi data Anda. AWS KMS menggunakan modul keamanan perangkat keras yang divalidasi FIPS 140-2 Level 3 untuk melindungi kunci Anda. Tidak ada mekanisme untuk mengekspor kunci AWS KMS ke dalam bentuk teks biasa.

Saat men-deploy beban kerja menggunakan strategi multiakun, merupakan salah satu praktik terbaik untuk menyimpan kunci AWS KMS di akun yang sama dengan beban kerja yang menggunakannya. Dalam model terdistribusi ini, tanggung jawab untuk mengelola kunci AWS KMS diemban oleh tim aplikasi. Dalam kasus penggunaan lainnya, organisasi dapat memilih untuk menyimpan kunci AWS KMS ke dalam sebuah akun terpusat. Struktur terpusat ini memerlukan kebijakan tambahan untuk mengaktifkan akses lintas akun yang diperlukan agar akun beban kerja dapat mengakses kunci yang disimpan di akun terpusat tersebut, tetapi mungkin lebih ideal untuk kasus penggunaan di mana satu kunci digunakan bersama-sama di beberapa Akun AWS.

Terlepas dari lokasi penyimpanan materi kunci, akses ke kunci harus dikontrol dengan ketat melalui penggunaan kebijakan kunci dan kebijakan IAM Kebijakan kunci adalah cara utama untuk mengontrol akses ke kunci AWS KMS. Selain itu, pemberian kunci AWS KMS dapat memberikan akses ke layanan AWS untuk mengenkripsi dan mendekripsi data atas nama Anda. Luangkan waktu untuk mempelajari praktik terbaik untuk kontrol akses ke kunci AWS KMS Anda.

Salah satu praktik terbaik adalah memantau penggunaan kunci enkripsi untuk mendeteksi pola akses yang tidak biasa. Operasi yang dijalankan menggunakan kunci yang dikelola AWS dan kunci yang dikelola pelanggan yang disimpan AWS KMS dapat dicatatkan dalam log di AWS CloudTrail dan harus ditinjau secara berkala. Perhatian khusus harus diberikan pada pemantauan peristiwa

penghancuran kunci. Untuk mengurangi penghancuran materi kunci yang tidak disengaja atau berbahaya, peristiwa penghancuran kunci tidak langsung menghapus materi kunci. Upaya untuk menghapus kunci di AWS KMS tunduk pada masa tunggu, yakni secara default 30 hari, sehingga memberikan waktu kepada administrator untuk meninjau tindakan ini dan membatalkan permintaan jika perlu.

Sebagian besar layanan AWS menggunakan AWS KMS secara transparan bagi Anda - satu-satunya persyaratan Anda adalah memutuskan apakah akan menggunakan kunci yang dikelola AWS atau dikelola pelanggan. Jika beban kerja Anda memerlukan penggunaan langsung AWS KMS untuk mengenkripsi atau mendekripsi data, praktik terbaiknya adalah menggunakan enkripsi amplop untuk melindungi data Anda. Perintah SDK Enkripsi AWS dapat menyediakan primitive enkripsi sisi klien aplikasi Anda untuk mengimplementasikan enkripsi amplop dan terintegrasi dengan AWS KMS.

Langkah implementasi

1. Tentukan opsi manajemen kunci (yang dikelola AWS atau dikelola pelanggan) yang tepat untuk kunci Anda.
   - Untuk memudahkan penggunaan, AWS menawarkan kunci yang dimiliki AWS dan yang dikelola AWS untuk sebagian besar layanan, yang menyediakan kemampuan enkripsi data diam tanpa perlu mengelola materi kunci atau kebijakan kunci.
   - Saat menggunakan kunci yang dikelola pelanggan, pertimbangkan penyimpanan kunci default untuk memberikan keseimbangan terbaik antara ketangkasan, keamanan, kedaulatan data, dan ketersediaan. Kasus-kasus penggunaan lain mungkin memerlukan penggunaan penyimpanan kunci kustom dengan AWS CloudHSM atau penyimpanan kunci eksternal.
2. Tinjau daftar layanan yang sedang Anda gunakan untuk beban kerja Anda untuk memahami bagaimana AWS KMS terintegrasi dengan layanan. Misalnya, instans EC2 dapat menggunakan volume EBS terenkripsi, yang memverifikasi bahwa snapshot Amazon EBS yang dibuat dari volume tersebut juga dienkripsi menggunakan kunci yang dikelola pelanggan dan mengurangi pengungkapan data snapshot yang tidak terenkripsi secara tidak disengaja.
   - Bagaimana layanan AWS menggunakan AWS KMS
   - Untuk informasi mendetail tentang opsi enkripsi yang ditawarkan oleh layanan AWS, lihat topik Enkripsi Data Diam di panduan pengguna atau panduan developer untuk layanan tersebut.
3. Implementasikan AWS KMS: AWS KMS memudahkan Anda membuat dan mengelola kunci serta mengontrol penggunaan enkripsi di berbagai layanan AWS dan dalam aplikasi Anda.
   - Memulai: AWS Key Management Service (AWS KMS)
   - Tinjau praktik terbaik untuk kontrol akses ke kunci AWS KMS Anda.

4. Pertimbangkan AWS Encryption SDK: Gunakan AWS Encryption SDK dengan integrasi AWS KMS ketika aplikasi Anda harus mengenkripsi data di sisi klien.
   - AWS Encryption SDK
5. Aktifkan IAM Access Analyzer agar secara otomatis meninjau dan memberi tahu jika ada kebijakan kunci AWS KMS yang terlalu luas.
6. Aktifkan Security Hub CSPM agar menerima notifikasi jika ada kebijakan kunci yang salah konfigurasi, kunci yang dijadwalkan untuk dihapus, atau kunci tanpa pengaktifan rotasi otomatis.
7. Tentukan tingkat pencatatan log yang sesuai untuk kunci AWS KMS Anda. Karena panggilan ke AWS KMS, termasuk peristiwa hanya-baca, dicatat ke log, jumlah log CloudTrail yang terkait dengan AWS KMS bisa jadi sangat banyak.
   - Beberapa organisasi lebih suka memisahkan aktivitas pencatatan log AWS KMS ke dalam jalur terpisah. Untuk detail selengkapnya, lihat bagian Mencatatkan log panggilan API AWS KMS dengan CloudTrail dalam panduan AWS KMS untuk developer.

Sumber daya

Dokumen terkait:

- AWS Key Management Service
- Layanan dan alat kriptografi AWS
- Melindungi Data Amazon S3 Menggunakan Enkripsi
- Enkripsi amplop
- Janji kedaulatan digital
- Menjelaskan operasi kunci AWS KMS, membawa kunci Anda sendiri, penyimpanan kunci kustom, dan portabilitas teks sandi
- Detail kriptografi AWS Key Management Service

Video terkait:

- Cara Kerja Enkripsi di AWS
- Mengamankan Penyimpanan Blok di AWS
- Perlindungan data AWS: Menggunakan gembok, kunci, tanda tangan, dan sertifikat

Contoh terkait:

- Mengimplementasikan mekanisme kontrol akses lanjutan menggunakan AWS KMS

SEC08-BP02 Menerapkan enkripsi data diam

Anda harus menerapkan penggunaan enkripsi untuk data diam. Enkripsi menjaga kerahasiaan data sensitif jika terjadi akses tidak sah atau pengungkapan yang tidak disengaja.

Hasil yang diinginkan: Data pribadi harus dienkripsi secara default saat diam. Enkripsi membantu menjaga kerahasiaan data dan memberikan lapisan perlindungan tambahan terhadap pengungkapan atau eksfiltrasi data yang disengaja atau tidak disengaja. Data yang dienkripsi tidak dapat dibaca atau diakses tanpa membuka enkripsi data terlebih dahulu. Semua data tersimpan yang tidak dienkripsi harus diinventarisasi dan dikontrol.

Antipola umum:

- Tidak menggunakan konfigurasi enkripsikan secara default.
- Memberikan akses yang terlalu permisif ke kunci dekripsi.
- Tidak memantau penggunaan kunci enkripsi dan dekripsi.
- Menyimpan data tidak terenkripsi.
- Menggunakan kunci enkripsi yang sama untuk semua data tanpa memperhatikan penggunaan, jenis, dan klasifikasi data.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Petakan kunci enkripsi ke klasifikasi data dalam beban kerja Anda. Pendekatan ini membantu melindungi dari akses yang terlalu permisif saat menggunakan kunci enkripsi tunggal atau yang sangat kecil untuk data Anda (lihat SEC07-BP01 Mengidentifikasi data dalam beban kerja Anda).

AWS Key Management Service (AWS KMS) terintegrasi dengan berbagai layanan AWS untuk mempermudah enkripsi data diam. Misalnya, di Amazon Simple Storage Service (Amazon S3), Anda dapat mengatur enkripsi default pada bucket agar semua objek baru dienkripsi secara otomatis. Saat menggunakan AWS KMS, pertimbangkan seberapa ketat pembatasan data yang perlu dilakukan. Kunci AWS KMS default dan yang dikontrol layanan dikelola dan digunakan atas nama Anda oleh AWS. Untuk data sensitif yang memerlukan akses terperinci ke kunci enkripsi yang mendasarinya, pertimbangkan kunci yang dikelola pelanggan (CMK). Anda memiliki kontrol penuh atas CMK, termasuk rotasi dan manajemen akses melalui penggunaan kebijakan kunci.

Selain itu, Amazon Elastic Compute Cloud (Amazon EC2) dan Amazon S3 mendukung penerapan enkripsi dengan mengatur enkripsi default. Anda dapat menggunakan Aturan AWS Config untuk memeriksa secara otomatis apakah Anda sedang menggunakan enkripsi. Misalnya, untuk volume Amazon Elastic Block Store (Amazon EBS), instans Amazon Relational Database Service (Amazon RDS), dan bucket Amazon S3.

AWS juga menyediakan opsi untuk enkripsi sisi klien, sehingga Anda dapat mengenkripsi data sebelum mengunggahnya ke cloud. AWS Encryption SDK menyediakan cara untuk mengenkripsi data Anda menggunakan enkripsi amplop. Anda memberikan kunci pembungkus, dan AWS Encryption SDK menghasilkan kunci data unik untuk setiap objek data yang dienkripsi. Pertimbangkan AWS CloudHSM jika Anda memerlukan modul keamanan perangkat keras (HSM) penyewa tunggal terkelola. AWS CloudHSM memungkinkan Anda membuat, mengimpor, dan mengelola kunci kriptografi pada HSM tervalidasi FIPS 140-2 level 3. Beberapa kasus penggunaan AWS CloudHSM termasuk melindungi kunci pribadi untuk menerbitkan otoritas sertifikat (CA), dan mengaktifkan enkripsi data transparan (TDE) untuk basis data Oracle. SDK Klien AWS CloudHSM menyediakan perangkat lunak yang dapat Anda gunakan untuk mengenkripsi data sisi klien menggunakan kunci yang disimpan di dalam AWS CloudHSM sebelum mengunggah data Anda ke AWS. Amazon DynamoDB Encryption Client juga mendukung enkripsi dan penandatanganan item sebelum diunggah ke tabel DynamoDB.

Langkah implementasi

- Terapkan enkripsi data diam untuk Amazon S3: Implementasikan enkripsi default bucket Amazon S3.

  Konfigurasikan enkripsi default untuk volume Amazon EBS baru: Tentukan bahwa Anda ingin membuat semua volume Amazon EBS baru dalam bentuk terenkripsi, dengan opsi penggunaan kunci default yang disediakan oleh AWS, atau kunci yang Anda buat.

  Konfigurasikan Amazon Machine Image (AMI) terenkripsi: Menyalin AMI yang ada dengan enkripsi aktif akan mengenkripsi volume root dan snapshot secara otomatis.

  Konfigurasikan enkripsi Amazon RDS: Konfigurasikan enkripsi untuk klaster dan snapshot basis data Amazon RDS Anda saat diam menggunakan opsi enkripsi.

  Buat dan konfigurasikan kunci AWS KMS dengan kebijakan yang membatasi akses ke pengguna utama yang sesuai untuk setiap klasifikasi data: Misalnya, buat satu kunci AWS KMS untuk mengenkripsi data produksi dan satu kunci untuk mengenkripsi data pengembangan atau pengujian. Anda juga dapat menyediakan kunci akses ke Akun AWS lainnya. Pertimbangkan

untuk memiliki akun yang berbeda untuk lingkungan pengembangan dan produksi Anda. Jika lingkungan produksi Anda perlu mendekripsi artefak di akun pengembangan, Anda dapat mengedit kebijakan CMK yang digunakan untuk mengenkripsi artefak pengembangan agar akun produksi dapat mendekripsi artefak tersebut. Kemudian lingkungan produksi dapat menyerap data yang didekripsi untuk digunakan dalam produksi.

Konfigurasikan enkripsi di layanan AWS tambahan: Untuk layanan AWS lain yang Anda gunakan, lihat dokumentasi keamanan untuk layanan terkait guna menentukan opsi enkripsi untuk layanan tersebut.

Sumber daya

Dokumen terkait:

- AWS Crypto Tools
- Dokumentasi AWS
- AWS Encryption SDK
- Laporan Resmi Detail Kriptografi AWS KMS
- AWS Key Management Service
- Layanan dan alat kriptografi AWS
- Enkripsi Amazon EBS
- Enkripsi default untuk volume Amazon EBS
- Mengenkripsi Sumber Daya Amazon RDS
- Bagaimana cara mengaktifkan enkripsi default untuk bucket Amazon S3?
- Melindungi Data Amazon S3 Menggunakan Enkripsi

Video terkait:

- Cara Kerja Enkripsi di AWS
- Mengamankan Penyimpanan Blok di AWS

SEC08-BP03 Mengotomatiskan perlindungan data diam

Gunakan alat otomatis untuk memvalidasi dan menegakkan kontrol data diam secara terus menerus, misalnya, memastikan bahwa hanya ada sumber daya penyimpanan terenkripsi. Anda bisa

mengotomatiskan validasi bahwa semua volume EBS telah dienkripsi menggunakan Aturan AWS Config. AWS Security Hub CSPM juga dapat memverifikasi beberapa kontrol yang berbeda melalui pemeriksaan otomatis berdasarkan standar keamanan. Selain itu, Aturan AWS Config Anda secara otomatis bisa memperbaiki sumber daya yang tidak patuh.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Data diam mewakili data yang Anda pertahankan di penyimpanan non-volatile selama durasi apa pun di beban kerja Anda. Data ini mencakup penyimpanan blok, penyimpanan objek, basis data, arsip, perangkat IoT, dan medium penyimpanan lain di mana datanya dipertahankan. Melindungi data diam Anda dapat mengurangi risiko akses yang tidak sah, ketika enkripsi dan kontrol akses yang tepat diimplementasikan.

Terapkan enkripsi data diam: Anda harus memastikan bahwa satu-satunya cara untuk menyimpan data adalah dengan menggunakan enkripsi. AWS KMS terintegrasi secara mulus dengan beberapa layanan AWS untuk mempermudah Anda mengenkripsi semua data diam Anda. Misalnya, di Amazon Simple Storage Service (Amazon S3) Anda bisa mengatur enkripsi default pada bucket sehingga semua objek baru terenkripsi secara otomatis. Selain itu, Amazon EC2 dan Amazon S3 mendukung penegakan enkripsi dengan mengatur enkripsi default. Anda dapat menggunakan AWS Managed Config Rules untuk memeriksa secara otomatis bahwa Anda menggunakan enkripsi, misalnya, untuk volume EBS, instans Amazon Relational Database Service (Amazon RDS), dan bucket Amazon S3.

Sumber daya

Dokumen terkait:

- AWS Crypto Tools
- AWS Encryption SDK

Video terkait:

- Cara Kerja Enkripsi di AWS
- Mengamankan Penyimpanan Blok di AWS

SEC08-BP04 Menerapkan kontrol akses

Untuk membantu melindungi data diam, terapkan kontrol akses menggunakan mekanisme, seperti isolasi dan versioning, serta terapkan prinsip hak akses paling rendah. Cegah pemberian akses publik ke data Anda.

Hasil yang diinginkan: Memastikan hanya pengguna yang sah yang dapat mengakses data sesuai kebutuhan. Melindungi data Anda dengan pencadangan dan versioning rutin untuk mencegah pengubahan atau penghapusan data yang disengaja atau tidak disengaja. Mengisolasi data penting dari data lain untuk melindungi kerahasiaan dan integritas data tersebut.

Antipola umum:

- Menyimpan data dengan kebutuhan atau klasifikasi sensitivitas yang berbeda secara bersamaan.
- Menggunakan izin yang terlalu permisif pada kunci dekripsi.
- Salah mengklasifikasi data.
- Tidak menyimpan pencadangan terperinci untuk data penting.
- Memberikan akses terus-menerus ke data produksi.
- Tidak mengaudit akses data atau meninjau izin secara rutin

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Beberapa kontrol dapat membantu melindungi data diam, termasuk akses (menggunakan hak akses paling rendah), isolasi, dan versioning. Akses ke data Anda harus diaudit menggunakan mekanisme deteksi, seperti AWS CloudTrail, dan log tingkat layanan, seperti log akses Amazon Simple Storage Service (Amazon S3). Anda harus membuat daftar data mana yang dapat diakses publik, dan menyusun rencana untuk mengurangi jumlah data yang tersedia untuk publik dari waktu ke waktu.

Kunci Vault Amazon Glacier dan Kunci Objek Amazon S3 memberikan kontrol akses wajib untuk objek di Amazon S3—begitu kebijakan vault dikunci dengan opsi kepatuhan, kebijakan tersebut tidak akan dapat diubah hingga kedaluwarsa, bahkan oleh pengguna root sekalipun.

Langkah implementasi

- Terapkan akses kontrol: Terapkan akses kontrol dengan hak akses paling rendah, termasuk akses ke kunci enkripsi.

- Pisahkan data berdasarkan tingkat klasifikasi yang berbeda: Gunakan berbagai Akun AWS untuk tingkat klasifikasi, dan kelola akun tersebut menggunakan AWS Organizations.
- Tinjau kebijakan AWS Key Management Service (AWS KMS): Tinjau tingkat akses yang diberikan di kebijakan AWS KMS.
- Tinjau izin objek dan bucket Amazon S3: Tinjau tingkat akses yang diberikan dalam kebijakan bucket S3 secara rutin. Praktik terbaiknya adalah menghindari penggunaan bucket yang dapat dibaca atau ditulis oleh publik. Coba gunakan AWS Config untuk mendeteksi bucket yang tersedia untuk publik, dan Amazon CloudFront untuk menyajikan konten dari Amazon S3. Pastikan bucket yang seharusnya tidak mengizinkan akses publik telah dikonfigurasi dengan benar untuk mencegah akses publik. Secara default, semua bucket S3 bersifat privat, dan hanya dapat diakses oleh pengguna yang telah diberi akses secara eksplisit.
- Aktifkan AWS IAM Access Analyzer: IAM Access Analyzer menganalisis bucket Amazon S3 dan menghasilkan temuan saat kebijakan S3 memberikan izin ke entitas eksternal.
- Aktifkan versioning Amazon S3 dan kunci objek jika perlu.
- Gunakan Inventaris Amazon S3: Inventaris Amazon S3 dapat digunakan untuk mengaudit serta melaporkan replikasi dan status enkripsi objek S3.
- Tinjau izin Amazon EBS dan berbagi AMI: Dengan izin berbagi, Anda dapat membagikan gambar dan volume kepada Akun AWS eksternal ke beban kerja Anda.
- Tinjau AWS Resource Access Manager: Berbagi secara berkala untuk menentukan apakah sumber daya harus terus dibagikan. Dengan Resource Access Manager, Anda dapat membagikan sumber daya seperti kebijakan AWS Network Firewall, aturan Amazon Route 53 Resolver, dan subnet dalam Amazon VPC Anda. Audit sumber daya yang dibagikan secara rutin dan hentikan pembagian sumber daya yang sudah tidak perlu dibagikan.

Sumber daya

Praktik Terbaik Terkait:

- SEC03-BP01 Menetapkan persyaratan akses
- SEC03-BP02 Memberikan hak akses paling rendah

Dokumen terkait:

- Laporan Resmi Detail Kriptografi AWS KMS
- Pengantar Manajemen Izin Akses ke Sumber Daya Amazon S3

- Gambaran umum manajemen akses ke sumber daya AWS KMS Anda
- Aturan AWS Config
- Amazon S3 + Amazon CloudFront: Kombinasi Fantastis di Cloud
- Menggunakan versioning
- Mengunci Objek Menggunakan Kunci Objek Amazon S3
- Membagikan Snapshot Amazon EBS
- AMI Bersama
- Meng-hosting aplikasi satu halaman aktif Amazon S3

Video terkait:

- Mengamankan Penyimpanan Blok di AWS

SEC08-BP05 Menggunakan mekanisme untuk mencegah orang mengakses data

Cegah semua pengguna dari mengakses sistem dan data sensitif secara langsung saat kondisi operasional normal. Misalnya, gunakan alur kerja manajemen perubahan untuk mengelola instans Amazon Elastic Compute Cloud (Amazon EC2) menggunakan alat, bukan dengan mengizinkan akses langsung atau host bastion. Hal ini dapat dicapai dengan AWS Systems Manager Automation, yaitu menggunakan dokumen otomatis yang berisi langkah yang Anda gunakan untuk menjalankan tugas. Dokumen tersebut dapat disimpan di kontrol sumber, ditinjau oleh rekan sebelum dijalankan, dan diuji secara menyeluruh untuk meminimalkan risiko dibandingkan akses shell. Pengguna bisnis dapat menggunakan dasbor, bukan akses langsung ke penyimpanan data, untuk menjalankan kueri. Ketika pipeline CI/CD tidak digunakan, tentukan proses dan kontrol mana yang diperlukan agar dapat menyediakan mekanisme akses break-glass nonaktif secara normal.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Rendah

Panduan implementasi

- Implementasikan mekanisme untuk mencegah orang dari mengakses data: Mekanisme termasuk menggunakan dasbor, seperti Quick, untuk menampilkan data ke pengguna ketimbang membuat kueri secara langsung.
  - Quick

- Otomatiskan manajemen konfigurasi: Lakukan tindakan dari jarak jauh, terapkan dan validasikan konfigurasi keamanan secara otomatis menggunakan layanan atau alat manajemen konfigurasi. Hindari penggunaan host bastion atau mengakses instans EC2 secara langsung.
  - AWS Systems Manager
  - AWS CloudFormation
  - Pipeline CI/CD untuk templat AWS CloudFormation di AWS

Sumber daya

Dokumen terkait:

- Laporan Resmi Detail Kriptografi AWS KMS

Video terkait:

- How Encryption Works in AWS
- Securing Your Block Storage on AWS

SEC 9. Bagaimana cara melindungi data bergerak Anda?

Lindungi data bergerak dengan mengimplementasikan beberapa kontrol untuk mengurangi risiko akses tanpa otorisasi atau hilangnya data.

Praktik terbaik

- SEC09-BP01 Mengimplementasikan manajemen sertifikat dan kunci keamanan
- SEC09-BP02 Menerapkan enkripsi data bergerak
- SEC09-BP03 Mengotomatiskan deteksi akses data yang tidak dimaksudkan
- SEC09-BP04 Sahkan komunikasi jaringan

SEC09-BP01 Mengimplementasikan manajemen sertifikat dan kunci keamanan

Sertifikat Keamanan Lapisan Pengangkutan (TLS) digunakan untuk mengamankan komunikasi jaringan dan menetapkan identitas situs web, sumber daya, dan beban kerja di internet, serta jaringan privat.

Hasil yang diinginkan: Sistem manajemen sertifikat aman yang dapat menyediakan, men-deploy, menyimpan, dan memperpanjang sertifikat di dalam infrastruktur kunci publik (PKI). Mekanisme manajemen kunci dan sertifikat yang aman mencegah pengungkapan materi kunci privat sertifikat dan secara otomatis memperpanjang sertifikat secara berkala. Mekanisme ini juga terintegrasi dengan layanan lain untuk menyediakan komunikasi jaringan yang aman dan identitas untuk sumber daya mesin di dalam beban kerja Anda. Materi kunci tidak boleh diakses oleh identitas manusia.

Antipola umum:

- Melakukan langkah-langkah manual selama proses deployment atau perpanjangan sertifikat.
- Kurang memperhatikan hierarki otoritas sertifikat (CA) saat merancang CA privat.
- Menggunakan sertifikat yang ditandatangani sendiri untuk sumber daya publik.

Manfaat menjalankan praktik terbaik ini:

- Sederhanakan manajemen sertifikat melalui deployment dan perpanjangan otomatis
- Dorong enkripsi data bergerak menggunakan sertifikat TLS
- Peningkatan keamanan dan auditabilitas tindakan sertifikat yang dilakukan oleh otoritas sertifikat
- Manajemen tugas-tugas manajemen di berbagai lapisan hierarki CA

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Beban kerja modern banyak memanfaatkan komunikasi jaringan terenkripsi menggunakan protokol PKI seperti TLS. Manajemen sertifikat PKI mungkin kompleks, tetapi penyediaan, deployment, dan perpanjangan sertifikat secara otomatis dapat mengurangi gesekan yang berkaitan dengan manajemen sertifikat.

AWS menyediakan dua layanan untuk mengelola sertifikat PKI tujuan umum: AWS Certificate Manager dan AWS Private Certificate Authority (AWS Private CA). ACM adalah layanan primer yang digunakan oleh pelanggan untuk menyediakan, mengelola, dan melakukan deployment sertifikat untuk digunakan di beban kerja publik maupun AWS privat. ACM mengeluarkan sertifikat menggunakan AWS Private CA dan terintegrasi dengan banyak layanan terkelola AWS lainnya untuk menyediakan sertifikat TLS yang aman untuk beban kerja.

Dengan AWS Private CA, Anda dapat membuat otoritas sertifikat root atau subordinat Anda sendiri dan menerbitkan sertifikat TLS melalui API. Anda dapat menggunakan jenis-jenis sertifikat ini dalam

skenario di mana Anda mengontrol dan mengelola rantai kepercayaan pada sisi klien koneksi TLS. Selain kasus penggunaan TLS, AWS Private CA dapat digunakan untuk menerbitkan sertifikat ke pod Kubernetes, atestasi produk perangkat Matter, penandatanganan kode, dan kasus penggunaan lain dengan templat kustom. Anda juga dapat menggunakan IAM Roles Anywhere untuk memberikan kredensial IAM sementara ke beban kerja on-premise yang telah diberikan sertifikat X.509 yang ditandatangani oleh CA Privat Anda.

Selain ACM dan AWS Private CA, AWS IoT Core memberikan dukungan khusus untuk penyediaan, manajemen, dan deployment sertifikat PKI ke perangkat IoT. AWS IoT Core menyediakan mekanisme khusus untuk memasukkan perangkat IoT ke dalam infrastruktur kunci publik Anda dalam skala besar.

Pertimbangan untuk membangun hierarki CA privat

Ketika Anda perlu membuat CA privat, penting untuk berhati-hati dalam merancang hierarki CA dengan benar di awal. Salah satu praktik terbaiknya adalah men-deploy setiap tingkat hierarki CA Anda ke dalam Akun AWS yang terpisah saat membuat hierarki CA privat. Langkah sengaja ini mengurangi luas permukaan untuk setiap tingkat di dalam hierarki CA, sehingga mempermudah penemuan anomali dalam data log CloudTrail dan mengurangi ruang lingkup akses atau dampak jika terdapat akses tidak sah ke salah satu akun. CA root harus berada di akun terpisahnya sendiri dan hanya boleh digunakan untuk menerbitkan satu atau beberapa sertifikat CA perantara.

Kemudian, buat satu atau beberapa CA perantara di akun yang terpisah dari akun CA root untuk menerbitkan sertifikat bagi pengguna akhir, perangkat, atau beban kerja lainnya. Terakhir, terbitkan sertifikat dari CA root Anda ke CA perantara, yang pada gilirannya akan menerbitkan sertifikat kepada pengguna akhir atau perangkat Anda. Untuk informasi selengkapnya tentang perencanaan deployment CA dan perancangan hierarki CA, termasuk perencanaan ketahanan, replikasi lintas wilayah, berbagi CA di seluruh organisasi, dan lainnya, lihat Merencanakan deployment AWS Private CA Anda.

Langkah implementasi

1. Tentukan layanan AWS relevan yang diperlukan untuk kasus penggunaan Anda:
   - Banyak kasus penggunaan dapat memanfaatkan infrastruktur kunci publik AWS yang sudah ada dengan menggunakan AWS Certificate Manager. ACM dapat digunakan untuk melakukan deployment sertifikat TLS untuk server web, penyeimbang beban, atau penggunaan lain untuk sertifikat yang dipercaya secara publik.
   - Pertimbangkan AWS Private CA ketika Anda perlu membuat hierarki otoritas sertifikat privat Anda sendiri atau memerlukan akses ke sertifikat yang dapat diekspor. ACM kemudian dapat

digunakan untuk menerbitkan banyak jenis sertifikat entitas akhir menggunakan AWS Private CA.

- Untuk kasus penggunaan di mana sertifikat harus disediakan dalam skala besar untuk perangkat Internet untuk Segala (IoT) yang disematkan, pertimbangkan AWS IoT Core.

2. Implementasikan perpanjangan sertifikat otomatis jika memungkinkan:
   - Gunakan perpanjangan yang dikelola ACM untuk sertifikat yang diterbitkan oleh ACM bersama dengan layanan terkelola AWS yang terintegrasi.
3. Bangun jalur pencatatan dan audit:
   - Aktifkan log CloudTrail untuk melacak akses ke akun yang memiliki otoritas sertifikat. Pertimbangkan mengonfigurasi validasi integritas file log di CloudTrail untuk memverifikasi keaslian data log.
   - Buat dan tinjau secara berkala laporan audit yang mencantumkan sertifikat yang telah diterbitkan atau dicabut oleh CA privat Anda. Laporan ini dapat diekspor ke bucket S3.
   - Saat men-deploy CA pribadi, Anda juga perlu membuat bucket S3 untuk menyimpan Daftar Pencabutan Sertifikat (CRL). Untuk panduan mengonfigurasi bucket S3 ini berdasarkan persyaratan beban kerja Anda, lihat Merencanakan daftar pencabutan sertifikat (CRL).

Sumber daya

Praktik terbaik terkait:

- SEC02-BP02 Menggunakan kredensial sementara
- SEC08-BP01 Mengimplementasikan manajemen kunci yang aman
- SEC09-BP04 Sahkan komunikasi jaringan

Dokumen terkait:

- Cara meng-host dan mengelola seluruh infrastruktur sertifikat privat di AWS.
- Cara mengamankan hierarki CA Privat ACM skala korporasi untuk otomotif dan manufaktur
- Praktik terbaik CA privat
- Cara menggunakan AWS RAM untuk membagikan CA Privat ACM Anda lintas akun

Video terkait:

- Mengaktifkan CA Privat AWS Certificate Manager (lokakarya)

Contoh terkait:

- Lokakarya CA privat
- Lokakarya Manajemen Perangkat IOT (termasuk penyediaan perangkat)

Alat terkait:

- Plugin ke Kubernetes cert-manager untuk menggunakan AWS Private CA

SEC09-BP02 Menerapkan enkripsi data bergerak

Terapkan persyaratan enkripsi yang Anda tentukan berdasarkan kebijakan, kewajiban regulasi, dan standar organisasi Anda untuk membantu memenuhi persyaratan organisasi, hukum, dan kepatuhan. Hanya gunakan protokol yang dienkripsi ketika mengirimkan data sensitif di luar cloud privat virtual (VPC) Anda. Enkripsi membantu menjaga kerahasiaan data, bahkan ketika data berada di jaringan tidak tepercaya.

Hasil yang diinginkan: Semua data harus dienkripsi saat bergerak menggunakan protokol TLS dan rangkaian sandi yang aman. Lalu lintas jaringan antara sumber daya Anda dan internet harus dienkripsi untuk mengurangi akses tidak sah ke data. Lalu lintas jaringan yang hanya berada dalam lingkungan AWS internal Anda harus sebisa mungkin dienkripsi menggunakan TLS. Jaringan internal AWS dienkripsi secara default dan lalu lintas jaringan di dalam VPC tidak dapat dipalsukan atau dilacak kecuali pihak yang tidak sah telah memperoleh akses ke sumber daya yang menghasilkan lalu lintas (seperti instans Amazon EC2 dan kontainer Amazon ECS). Pertimbangkan untuk melindungi lalu lintas antarjaringan dengan jaringan privat virtual (VPN) IPsec.

Antipola umum:

- Menggunakan versi komponen SSL, TLS, rangkaian sandi yang tidak digunakan lagi (misalnya, SSL v3.0, kunci RSA 1024-bit, dan sandi RC4).
- Mengizinkan lalu lintas (HTTP) tidak terenkripsi ke atau dari sumber daya yang dapat diakses publik.
- Tidak memantau dan tidak mengganti sertifikat X.509 sebelum kedaluwarsa.
- Menggunakan sertifikat X.509 yang Anda buat sendiri untuk TLS.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Layanan AWS menyediakan titik akhir HTTPS menggunakan TLS untuk komunikasi, memberikan enkripsi data bergerak saat berkomunikasi dengan API AWS. Protokol yang tidak aman, seperti HTTP, dapat diaudit dan diblokir di VPC melalui penggunaan grup keamanan. Permintaan HTTP juga dapat secara otomatis diarahkan ke HTTPS di Amazon CloudFront atau pada Application Load Balancer. Anda memiliki kendali penuh atas sumber daya komputasi Anda untuk mengimplementasikan enkripsi data bergerak di seluruh layanan Anda. Selain itu, Anda dapat menggunakan sambungan VPN ke dalam VPC Anda dari jaringan eksternal atau AWS Direct Connect untuk mendukung enkripsi lalu lintas. Pastikan klien Anda melakukan panggilan ke API AWS menggunakan minimal TLS 1.2, karena AWS menghentikan penggunaan TLS 1.0 dan 1.1 pada Juni 2023. Jika Anda memiliki persyaratan khusus, tersedia solusi pihak ketiga di AWS Marketplace.

Langkah implementasi

- Terapkan enkripsi data bergerak: Persyaratan enkripsi yang Anda tetapkan harus didasarkan pada standar dan praktik terbaik terbaru dan hanya mengizinkan protokol yang aman. Misalnya, konfigurasikan grup keamanan untuk hanya mengizinkan protokol HTTPS ke penyeimbang beban aplikasi atau instans Amazon EC2.
- Konfigurasikan protokol yang aman di layanan edge: Konfigurasikan HTTPS dengan Amazon CloudFront dan gunakan profil keamanan yang sesuai dengan postur keamanan dan kasus penggunaan Anda.
- Gunakan VPN untuk koneksi eksternal: Pertimbangkan penggunaan VPN IPsec untuk mengamankan koneksi antartitik atau antarjaringan untuk membantu memberikan integritas dan privasi data.
- Konfigurasikan protokol yang aman di penyeimbang beban: Pilih kebijakan keamanan yang menyediakan rangkaian sandi terkuat yang didukung oleh klien yang akan terhubung ke pendengar. Membuat pendengar HTTPS untuk Application Load Balancer Anda.
- Konfigurasikan protokol yang aman di Amazon Redshift: Konfigurasikan klaster Anda untuk meminta koneksi lapisan soket aman (SSL) atau keamanan lapisan pengangkutan (TLS).
- Konfigurasikan protokol yang aman: Baca dokumentasi layanan AWS untuk menentukan kemampuan enkripsi data bergerak.
- Konfigurasikan akses yang aman saat melakukan pengunggahan ke bucket Amazon S3: Gunakan kontrol kebijakan bucket Amazon S3 untuk menerapkan akses aman ke data.

- Pertimbangkan penggunaan AWS Certificate Manager: ACM memungkinkan Anda untuk menyediakan, mengelola, dan melakukan deployment sertifikat TLS publik untuk digunakan dengan layanan AWS.
- Pertimbangkan penggunaan AWS Private Certificate Authority untuk kebutuhan PKI privat: AWS Private CA memungkinkan Anda membuat hierarki otoritas sertifikat (CA) pribadi untuk menerbitkan sertifikat X.509 entitas akhir yang dapat digunakan untuk membuat saluran TLS terenkripsi.

Sumber daya

Dokumen terkait:

- Dokumentasi AWS
- Menggunakan HTTPS dengan CloudFront
- Menghubungkan VPC ke jaringan jarak jauh menggunakan AWS Virtual Private Network
- Membuat pendengar HTTPS untuk Application Load Balancer Anda
- Tutorial: Mengonfigurasi SSL/TLS di Amazon Linux 2
- Menggunakan SSL/TLS untuk mengenkripsi koneksi ke instans DB
- Mengonfigurasi opsi-opsi keamanan untuk koneksi

SEC09-BP03 Mengotomatiskan deteksi akses data yang tidak dimaksudkan

Gunakan alat seperti Amazon GuardDuty untuk secara otomatis mendeteksi aktivitas mencurigakan atau mencoba memindahkan data di luar batas yang telah ditetapkan. Misalnya, GuardDuty dapat mendeteksi aktivitas baca Amazon Simple Storage Service (Amazon S3) yang tidak seperti biasanya dengan Temuan Exfiltration:S3/AnomalousBehavior. Selain GuardDuty, Log Alur Amazon VPCyang mendokumentasikan informasi lalu lintas jaringan, dapat digunakan dengan Amazon EventBridge untuk memicu deteksi koneksi tidak normal, baik yang berhasil maupun yang ditolak. Amazon S3 Access Analyzer dapat membantu mengukur data apa yang dapat diakses di dalam bucket Amazon S3 Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Otomatiskan deteksi akses data yang tidak dimaksudkan: Gunakan alat atau mekanisme deteksi untuk secara otomatis mendeteksi upaya untuk memindahkan data di luar batas yang ditetapkan; misalnya, untuk mendeteksi sistem basis data yang menyalin data ke host tidak dikenal.
  - Log Alur VPC
- Pertimbangkan Amazon Macie: Amazon Macie adalah layanan privasi data dan keamanan data terkelola secara penuh yang menggunakan machine learning dan pencocokan pola untuk menemukan dan melindungi data sensitif Anda di AWS.
  - Amazon Macie

Sumber daya

Dokumen terkait:

- Log Alur VPC
- Amazon Macie

SEC09-BP04 Sahkan komunikasi jaringan

Verifikasi identitas komunikasi dengan menggunakan protokol yang mendukung autentikasi, seperti Keamanan Lapisan Pengangkutan (TLS) atau IPsec.

Rancang beban kerja Anda untuk menggunakan protokol jaringan yang aman dan terautentikasi setiap kali berkomunikasi antara layanan, aplikasi, atau ke pengguna. Menggunakan protokol jaringan yang mendukung autentikasi dan otorisasi memberikan kontrol yang lebih kuat atas aliran jaringan dan mengurangi dampak akses yang tidak sah.

Hasil yang diinginkan: Beban kerja dengan arus lalu lintas bidang data dan bidang kontrol yang jelas antara layanan. Arus lalu lintas menggunakan protokol jaringan yang diautentikasi dan dienkripsi jika memungkinkan secara teknis.

Antipola umum:

- Alur lalu lintas yang tidak dienkripsi atau tidak diautentikasi dalam beban kerja Anda.
- Penggunaan kembali kredensial autentikasi oleh beberapa pengguna atau entitas.
- Hanya mengandalkan kontrol jaringan sebagai mekanisme kontrol akses.

- Membuat mekanisme autentikasi kustom, bukan mengandalkan mekanisme autentikasi standar industri.
- Alur lalu lintas yang terlalu permisif antara komponen layanan atau sumber daya lain di VPC.

Manfaat menjalankan praktik terbaik ini:

- Membatasi ruang lingkup dampak untuk akses tidak sah ke satu bagian dari beban kerja.
- Memberikan tingkat jaminan yang lebih tinggi bahwa tindakan hanya dilakukan oleh entitas yang diautentikasi.
- Meningkatkan pemisahan layanan dengan menggambarkan dengan jelas dan menerapkan antarmuka transfer data yang diinginkan.
- Meningkatkan pemantauan, pembuatan log, dan respons insiden melalui atribusi permintaan dan antarmuka komunikasi yang digambarkan dengan jelas.
- Memberikan pertahanan mendalam untuk beban kerja Anda dengan menggabungkan kontrol jaringan dengan kontrol autentikasi dan otorisasi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Pola lalu lintas jaringan beban kerja Anda dapat dikelompokkan ke dalam dua kategori:

- Lalu lintas timur-barat mewakili arus lalu lintas antara layanan yang membentuk beban kerja.
- Lalu lintas utara-selatan mewakili arus lalu lintas antara beban kerja Anda dan konsumen.

Meskipun enkripsi lalu lintas utara-selatan merupakan praktik umum, pengamanan lalu lintas timur-barat menggunakan protokol yang diautentikasi merupakan hal yang kurang umum. Praktik keamanan modern menyebutkan bahwa desain jaringan saja tidak cukup untuk memberikan hubungan yang dapat dipercaya antara dua entitas. Ketika dua layanan dapat berada dalam batas jaringan yang sama, sebaiknya enkripsi, autentikasi, dan otorisasi komunikasi di antara layanan-layanan tersebut tetap dilakukan.

Sebagai contoh, API layanan AWS menggunakan protokol tanda tangan Signature Version 4 (SIGv4) AWS untuk mengautentikasi pemanggil, dari jaringan mana pun permintaan tersebut berasal. Autentikasi ini memastikan bahwa API AWS dapat memverifikasi identitas yang meminta tindakan,

dan identitas tersebut kemudian dapat digabungkan dengan kebijakan untuk membuat keputusan otorisasi guna menentukan apakah tindakan tersebut harus diizinkan atau tidak.

Layanan seperti Amazon VPC Lattice dan Amazon API Gateway memungkinkan Anda menggunakan protokol tanda tangan SigV4 yang sama untuk menambahkan autentikasi dan otorisasi ke lalu lintas timur-barat dalam beban kerja Anda sendiri. Jika sumber daya di luar lingkungan AWS Anda perlu berkomunikasi dengan layanan yang memerlukan autentikasi dan otorisasi berbasis SigV4, Anda dapat menggunakan AWS Identity and Access Management (IAM) Roles Anywhere pada sumber daya non-AWS untuk memperoleh kredensial AWS sementara. Kredensial ini dapat digunakan untuk menandatangani permintaan ke layanan menggunakan SigV4 untuk memberi otorisasi akses.

Mekanisme umum lainnya untuk mengautentikasi lalu lintas timur-barat adalah autentikasi timbal balik TLS (mTLS). Banyak Internet untuk Segala (IoT), aplikasi bisnis-ke-bisnis, dan layanan mikro menggunakan mTLS untuk memvalidasi identitas kedua sisi komunikasi TLS melalui penggunaan sertifikat X.509 sisi klien dan server. Sertifikat ini dapat dikeluarkan oleh AWS Private Certificate Authority (AWS Private CA). Anda dapat menggunakan layanan seperti Amazon API Gateway dan AWS App Mesh untuk menyediakan autentikasi mTLS untuk komunikasi antar- atau intra-beban kerja. Meskipun mTLS menyediakan informasi autentikasi untuk kedua sisi komunikasi TLS, mekanisme untuk otorisasi tidak disediakan.

Akhirnya, OAuth 2.0 dan OpenID Connect (OIDC) adalah dua protokol yang biasanya digunakan untuk mengontrol akses ke layanan oleh pengguna, tetapi kini juga mulai populer untuk lalu lintas antar-layanan. API Gateway menyediakan pemberi otorisasi JSON Web Token (JWT), yang memungkinkan beban kerja membatasi akses ke rute API menggunakan JWT yang dikeluarkan dari penyedia identitas OIDC atau OAuth 2.0. Cakupan OAuth2 dapat digunakan sebagai sumber untuk keputusan otorisasi dasar, tetapi pemeriksaan otorisasi masih perlu diimplementasikan di lapisan aplikasi, dan cakupan OAuth2 saja tidak dapat mendukung kebutuhan otorisasi yang lebih kompleks.

Langkah implementasi

- Tentukan dan dokumentasikan alur jaringan beban kerja Anda: Langkah pertama dalam menerapkan strategi pertahanan mendalam adalah menentukan arus lalu lintas beban kerja Anda.
  - Buat diagram alur data yang secara jelas menggambarkan bagaimana data ditransmisikan antara berbagai layanan yang membentuk beban kerja Anda. Diagram ini merupakan langkah pertama untuk menerapkan alur tersebut melalui saluran jaringan yang diautentikasi.
  - Instrumentasikan beban kerja Anda dalam fase pengembangan dan pengujian untuk memvalidasi bahwa diagram alur data mencerminkan perilaku beban kerja secara akurat pada saat runtime.

    - Diagram alur data juga dapat berguna saat melakukan latihan pemodelan ancaman, seperti yang dijelaskan dalam SEC01-BP07 Mengidentifikasi ancaman dan memprioritaskan mitigasi menggunakan model ancaman.
- Tetapkan kontrol jaringan: Pertimbangkan kemampuan AWS untuk menetapkan kontrol jaringan yang selaras dengan alur data Anda. Meskipun batas jaringan seharusnya tidak menjadi satu-satunya kontrol keamanan, batas tersebut menyediakan lapisan pada strategi pertahanan mendalam untuk melindungi beban kerja Anda.
    - Gunakan grup keamanan untuk menetapkan dan membatasi alur data antarsumber daya.
    - Pertimbangkan penggunaan AWS PrivateLink untuk berkomunikasi dengan layanan AWS dan layanan pihak ketiga yang mendukung AWS PrivateLink. Data yang dikirim melalui titik akhir antarmuka AWS PrivateLink tetap berada di dalam tulang punggung jaringan AWS dan tidak melintasi Internet publik.
- Implementasikan autentikasi dan otorisasi di seluruh layanan dalam beban kerja Anda: Pilih kumpulan layanan AWS yang paling tepat untuk menyediakan alur lalu lintas yang dienkripsi dan diautentikasi dalam beban kerja Anda.
    - Pertimbangkan Amazon VPC Lattice untuk mengamankan komunikasi antar layanan. VPC Lattice dapat menggunakan autentikasi SigV4 yang dikombinasikan dengan kebijakan autentikasi untuk mengontrol akses antar layanan.
    - Untuk komunikasi antar layanan menggunakan mTLS, pertimbangkan API Gateway atau App Mesh. AWS Private CA dapat digunakan untuk membuat hierarki CA pribadi yang mampu mengeluarkan sertifikat untuk digunakan dengan mTLS.
    - Saat mengintegrasikan dengan layanan menggunakan OAuth 2.0 atau OIDC, pertimbangkan API Gateway menggunakan pemberi otorisasi JWT.
    - Untuk komunikasi antara beban kerja Anda dan perangkat IoT, pertimbangkan AWS IoT Core, yang menyediakan beberapa opsi untuk enkripsi lalu lintas jaringan dan autentikasi.
- Pantau akses yang tidak sah: Terus pantau saluran komunikasi yang tidak diinginkan, pengguna utama tidak berwenang yang mencoba mengakses sumber daya yang dilindungi, dan pola akses yang tidak tepat lainnya.
    - Jika VPC Lattice digunakan untuk mengelola akses ke layanan Anda, pertimbangkan untuk mengaktifkan dan memantau log akses VPC Lattice. Log akses ini mencakup informasi tentang entitas yang meminta, informasi jaringan termasuk VPC sumber dan tujuan, dan metadata permintaan.
    - Pertimbangkan untuk mengaktifkan log alur VPC untuk menangkap metadata pada alur jaringan dan meninjau anomali secara berkala.

- Lihat Panduan Respons Insiden Keamanan AWS dan bagian Respons Insiden dari pilar keamanan Kerangka Kerja AWS Well-Architected untuk panduan lebih lanjut tentang merencanakan, menyimulasikan, dan menanggapi insiden keamanan.

Sumber daya

Praktik Terbaik Terkait:

- SEC03-BP07 Menganalisis akses lintas akun dan publik
- SEC02-BP02 Menggunakan kredensial sementara
- SEC01-BP07 Mengidentifikasi ancaman dan memprioritaskan mitigasi menggunakan model ancaman

Dokumen terkait:

- Evaluating access control methods to secure Amazon API Gateway APIs
- Configuring mutual TLS authentication for a REST API
- How to secure API Gateway HTTP endpoints with JWT authorizer
- Authorizing direct calls to AWS services using AWS IoT Core credential provider
- Panduan Respons Insiden Keamanan AWS

Video terkait:

- AWS re:invent 2022: Introducing VPC Lattice
- AWS re:invent 2020: Serverless API authentication for HTTP APIs on AWS

Contoh terkait:

- Amazon VPC Lattice Workshop
- Zero-Trust Episode 1 – The Phantom Service Perimeter workshop

Respons insiden

Pertanyaan

- SEC 10. Bagaimana cara mengantisipasi, merespons, dan pulih dari insiden?

SEC 10. Bagaimana cara mengantisipasi, merespons, dan pulih dari insiden?

Dengan kontrol deteksi dan preventif yang matang sekalipun, organisasi Anda harus mengimplementasikan mekanisme untuk merespons dan memitigasi potensi dampak insiden keamanan. Persiapan Anda sangat berpengaruh pada kemampuan tim Anda untuk beroperasi secara efektif selama insiden, untuk mengisolasi, membatasi, dan melakukan forensik terhadap masalah, serta untuk memulihkan operasi ke kondisi yang baik dan dikenal. Menetapkan alat dan akses sebelum terjadi insiden keamanan, lalu secara rutin melatih respons insiden melalui game day, membantu memastikan bahwa Anda dapat melakukan pemulihan sembari tetap meminimalkan gangguan bisnis.

Praktik terbaik

- SEC10-BP01 Identifikasikan sumber daya eksternal dan personel kunci
- SEC10-BP02 Membuat rencana manajemen insiden
- SEC10-BP03 Menyiapkan kemampuan forensik
- SEC10-BP04 Mengembangkan dan menguji playbook respons insiden keamanan
- SEC10-BP05 Menyediakan akses di awal
- SEC10-BP06 Melakukan deployment alat di awal
- SEC10-BP07 Menjalankan simulasi
- SEC10-BP08 Menetapkan kerangka kerja untuk belajar dari insiden

SEC10-BP01 Identifikasikan sumber daya eksternal dan personel kunci

Identifikasikan kewajiban legal, sumber daya, dan personel internal serta eksternal yang dapat membantu organisasi Anda merespons insiden.

Saat Anda menentukan pendekatan Anda terhadap respons insiden di cloud, bersama dengan tim lainnya (seperti penasihat hukum, pimpinan, pemangku kepentingan bisnis, Layanan AWS Support, dan lainnya), Anda harus mengidentifikasi personel kunci, pemangku kepentingan, dan kontak yang relevan. Untuk mengurangi dependensi dan mempercepat waktu respons, pastikan tim Anda, tim keamanan spesialis, dan pemberi respons paham tentang layanan yang Anda gunakan dan memiliki kesempatan untuk praktik langsung.

Sebaiknya identifikasikan partner keamanan AWS eksternal yang dapat memberi Anda ahli dari luar perusahaan dan memberikan perspektif yang berbeda untuk melengkapi kemampuan respons

Anda. Partner keamanan tepercaya Anda dapat membantu Anda mengidentifikasi potensi risiko atau ancaman yang belum Anda kenali dengan baik.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

- Identifikasi personel utama dalam organisasi Anda: Miliki daftar kontak personel di dalam organisasi Anda yang perlu Anda libatkan untuk merespons dan melakukan pemulihan dari insiden.
- Identifikasi partner eksternal: Bekerja samalah dengan partner eksternal yang dapat membantu Anda merespons dan melakukan pemulihan dari insiden, jika diperlukan.

Sumber daya

Dokumen terkait:

- Panduan Respons Insiden AWS

Video terkait:

- Prepare for and respond to security incidents in your AWS environment

Contoh terkait:

SEC10-BP02 Membuat rencana manajemen insiden

Dokumen pertama yang harus dikembangkan untuk merespons insiden adalah rencana respons insiden. Rencana respons insiden dirancang untuk menjadi landasan bagi program dan strategi respons insiden Anda.

Manfaat menjalankan praktik terbaik ini: Mengembangkan proses respons insiden yang menyeluruh dan jelas adalah kunci untuk program respons insiden yang sukses dan dapat diskalakan. Ketika peristiwa keamanan terjadi, langkah dan alur kerja yang jelas dapat membantu Anda merespons secara tepat waktu. Anda mungkin sudah memiliki proses respons insiden yang ada. Terlepas dari status Anda saat ini, penting untuk memperbarui, mengiterasi, dan menguji proses respons insiden Anda secara rutin.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Rencana manajemen insiden sangat penting untuk merespons, memitigasi, dan pulih dari potensi dampak insiden keamanan. Rencana manajemen insiden adalah proses terstruktur untuk mengidentifikasi, memperbaiki, dan merespons insiden keamanan secara tepat waktu.

Cloud memiliki banyak peran dan persyaratan operasional yang sama yang juga ditemukan di lingkungan on-premise. Saat membuat rencana manajemen insiden, penting untuk mempertimbangkan strategi respons dan pemulihan yang paling selaras dengan hasil bisnis dan persyaratan kepatuhan Anda. Sebagai contoh, jika Anda mengoperasikan beban kerja di AWS yang patuh terhadap FedRAMP di Amerika Serikat, ada gunanya Anda mematuhi NIST SP 800-61 Panduan Penanganan Keamanan Komputer. Begitu juga, saat mengoperasikan beban kerja dengan data informasi pengenal pribadi (PII) Eropa, pertimbangkan skenario seperti cara Anda melindungi dan merespons permasalahan terkait residensi data yang diatur oleh Peraturan Perlindungan Data Umum (GDPR) Uni Eropa.

Saat membangun rencana manajemen insiden untuk beban kerja di AWS, mulailah dengan Model Tanggung Jawab Bersama AWS untuk membangun pendekatan pertahanan mendalam untuk respons insiden. Dalam model ini, AWS mengelola keamanan cloud, dan Anda bertanggung jawab atas keamanan di cloud. Ini artinya Anda mempertahankan kontrol dan bertanggung jawab atas kontrol keamanan yang ingin Anda implementasikan. Panduan Respons Insiden Keamanan AWS menguraikan konsep utama dan panduan mendasar untuk membangun rencana manajemen insiden berorientasi cloud.

Rencana manajemen insiden yang efektif harus diiterasi secara berkelanjutan, dan harus tetap mutakhir sesuai tujuan operasi cloud Anda. Pertimbangkan menggunakan rencana implementasi yang diuraikan di bawah seiring Anda membuat dan mengembangkan rencana manajemen insiden Anda.

Langkah implementasi

Tentukan peran dan tanggung jawab

Penanganan peristiwa keamanan membutuhkan disiplin lintas organisasi dan keinginan untuk bertindak. Dalam struktur organisasi Anda, harus ada banyak orang yang bertanggung jawab, akuntabel, memberi konsultasi, atau terus mendapatkan informasi selama insiden, seperti perwakilan dari sumber daya manusia (SDM), tim eksekutif, dan legal. Pertimbangkan peran dan tanggung jawab ini, dan apakah pihak ketiga harus dilibatkan. Perhatikan bahwa banyak kawasan memiliki undang-undang setempat yang mengatur apa yang seharusnya dan tidak boleh dilakukan.

Meskipun pembuatan bagan peran yang bertanggung jawab, akuntabel, memberi konsulatasi, mendapat informasi (RACI) untuk rencana respons keamanan terdengar birokratis, tindakan ini dapat memudahkan komunikasi yang cepat dan langsung serta menjelaskan secara gamblang kepemimpinan di berbagai tahap peristiwa.

Selama insiden, pelibatan pemilik dan developer aplikasi dan sumber daya yang terkena dampak adalah hal yang sangat penting karena mereka adalah pakar bidang (SME) yang dapat memberikan informasi dan konteks untuk membantu mengukur dampak. Pastikan untuk mempraktikkan dan membangun hubungan dengan developer dan pemilik aplikasi sebelum Anda mengandalkan keahlian mereka untuk merespons insiden. Pemilik aplikasi atau SME, seperti administrator atau rekayasawan cloud Anda, mungkin perlu bertindak dalam situasi ketika lingkungan kurang dikenal atau kompleks, atau ketika perespons tidak memiliki akses.

Terakhir, partner tepercaya dapat dilibatkan dalam penyelidikan atau respons karena mereka dapat memberikan keahlian tambahan dan pengawasan yang berharga. Ketika Anda tidak memiliki semua keterampilan ini di tim Anda sendiri, Anda mungkin ingin menggunakan bantuan dari pihak eksternal.

Pahami dukungan dan tim respons AWS

- AWS Dukungan
  - Dukungan menawarkan berbagai paket yang menyediakan akses ke alat dan keahlian yang mendukung kesuksesan dan kesehatan operasional solusi AWS Anda. Jika Anda memerlukan dukungan teknis dan lebih banyak sumber daya untuk membantu merencanakan, men-deploy, dan mengoptimalkan lingkungan AWS Anda, Anda dapat memilih paket dukungan yang paling sesuai dengan kasus penggunaan AWS Anda.
  - Pertimbangkan Pusat Dukungan di Konsol Manajemen AWS (diperlukan login) sebagai titik kontak utama untuk mendapatkan dukungan atas masalah yang memengaruhi sumber daya AWS Anda. Akses ke Dukungan dikontrol oleh AWS Identity and Access Management. Untuk informasi selengkapnya tentang mendapatkan akses ke fitur Dukungan, lihat Memulai dengan Dukungan.
- Tim Respons Insiden Pelanggan (CIRT) AWS
  - Tim Respons Insiden Pelanggan (CIRT) AWS adalah tim AWS global 24/7 khusus yang memberikan dukungan kepada pelanggan selama peristiwa keamanan aktif di sisi pelanggan dalam Model Tanggung Jawab Bersama AWS.
  - Ketika CIRT AWS mendukung Anda, mereka memberikan bantuan dengan evaluasi awal dan pemulihan untuk peristiwa keamanan aktif di AWS. Mereka dapat membantu menganalisis akar masalah melalui penggunaan log layanan AWS dan memberi Anda saran-saran pemulihan.

Mereka juga dapat memberikan rekomendasi dan praktik terbaik keamanan untuk membantu Anda menghindari peristiwa keamanan di masa depan.

  - Pelanggan AWS dapat melibatkan CIRT AWS melalui kasus Dukungan.
- Dukungan respons DDoS
  - AWS menawarkan AWS Shield, yang menyediakan layanan perlindungan penolakan layanan terdistribusi (DDoS) yang terkelola yang melindungi aplikasi web yang berjalan di AWS. Shield menyediakan deteksi yang selalu aktif dan mitigasi inline otomatis yang dapat meminimalkan waktu henti dan latensi aplikasi, sehingga pelanggan tidak perlu melibatkan Dukungan untuk mendapatkan manfaat dari perlindungan DDoS. Terdapat dua tingkatan Shield: AWS Shield Standard dan AWS Shield Advanced. Untuk mempelajari perbedaan antara kedua tingkatan ini, lihat Dokumentasi fitur Shield.
- AWS Managed Services (AMS)
  - AWS Managed Services (AMS) menyediakan manajemen yang berkelanjutan untuk infrastruktur AWS Anda, sehingga Anda dapat berfokus pada aplikasi Anda. Dengan mengimplementasikan praktik terbaik untuk memelihara infrastruktur Anda, AMS membantu mengurangi overhead dan risiko operasional. AMS mengotomatiskan kegiatan umum seperti permintaan perubahan, pemantauan, manajemen patch, keamanan, dan layanan pencadangan serta menyediakan layanan siklus hidup penuh untuk menyediakan, menjalankan, dan mendukung infrastruktur Anda.
  - AMS bertanggung jawab untuk deployment serangkaian kontrol detektif keamanan dan menyediakan respons lini depan selama 24/7 terhadap peringatan. Saat peringatan dimulai, AMS mengikuti rangkaian standar playbook otomatis dan manual untuk memastikan konsistensi respons. Playbook ini dibagikan kepada pelanggan AMS selama orientasi sehingga mereka dapat mengembangkan dan mengoordinasikan respons dengan AMS.

Kembangkan rencana respons insiden

Rencana respons insiden dirancang untuk menjadi landasan bagi program dan strategi respons insiden Anda. Rencana respons insiden harus dalam dokumen resmi. Rencana respons insiden biasanya menyertakan bagian-bagian ini:

- Ikhtisar tim respons insiden: Menguraikan tujuan dan fungsi tim respons insiden.
- Peran dan tanggung jawab: Mencantumkan daftar pemangku kepentingan respons insiden dan memperinci peran mereka ketika insiden terjadi.
- Rencana komunikasi: Berisi detail informasi kontak dan cara Anda berkomunikasi selama insiden.

- Metode komunikasi cadangan: Salah satu praktik terbaik adalah memiliki komunikasi di luar saluran normal (out-of-band) sebagai cadangan untuk komunikasi insiden. Contoh aplikasi yang menyediakan saluran komunikasi out-of-band yang aman adalah AWS Wickr.
- Tahapan respons insiden dan tindakan yang harus dilakukan: Menyebutkan satu per satu tahapan respons insiden (misalnya mendeteksi, menganalisis, memberantas, mengendalikan, dan memulihkan), termasuk tindakan umum yang harus dilakukan dalam tahapan-tahapan tersebut.
- Penetapan tingkat keparahan dan prioritas insiden: Menjelaskan cara mengklasifikasikan tingkat keparahan insiden, cara memprioritaskan insiden, dan bagaimana penetapan keparahan memengaruhi prosedur eskalasi.

Meskipun bagian-bagian ini sudah umum ada di perusahaan dari berbagai ukuran dan industri, rencana respons insiden di setiap organisasi berbeda-beda. Anda perlu membangun rencana respons insiden yang paling cocok untuk organisasi Anda.

Sumber daya

Praktik terbaik terkait:

- SEC04 (Bagaimana cara mendeteksi dan menyelidiki peristiwa keamanan?)

Dokumen terkait:

- Respons Insiden Keamanan AWS
- NIST: Panduan Penanganan Insiden Keamanan Komputer

SEC10-BP03 Menyiapkan kemampuan forensik

Sebelum terjadinya insiden keamanan, pertimbangkan untuk mengembangkan kemampuan forensik untuk mendukung investigasi event keamanan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Konsep dari forensik on-premise tradisional berlaku untuk AWS. Untuk informasi penting untuk mulai membangun kemampuan forensik di AWS Cloud, lihat Strategi lingkungan investigasi forensik di AWS Cloud.

Setelah Anda menyiapkan lingkungan dan struktur Akun AWS Anda untuk forensik, tentukan teknologi yang diperlukan untuk secara efektif melakukan metodologi yang baik secara forensik di empat fase:

- Pengumpulan: Kumpulkan log AWS yang relevan, seperti AWS Config, Log Aliran VPC, dan log tingkat host. Kumpulkan snapshot, cadangan, dan dump memori dari sumber daya AWS yang terkena dampak jika tersedia.
- Pemeriksaan: Periksa data yang dikumpulkan dengan mengekstraksi dan menilai informasi yang relevan.
- Analisis: Lakukan analisis data yang dikumpulkan untuk memahami insiden dan menarik kesimpulan darinya.
- Pelaporan: Sajikan informasi yang dihasilkan dari fase analisis.

Langkah implementasi

Persiapkan lingkungan forensik Anda

AWS Organizations membantu Anda mengelola dan mengatur lingkungan AWS secara terpusat saat Anda mengembangkan dan menskalakan sumber daya AWS. Sebuah organisasi AWS menggabungkan Akun AWS Anda sehingga Anda dapat mengelolanya sebagai satu unit. Anda dapat menggunakan unit organisasi (OU) untuk mengumpulkan akun-akun untuk dikelola sebagai satu unit.

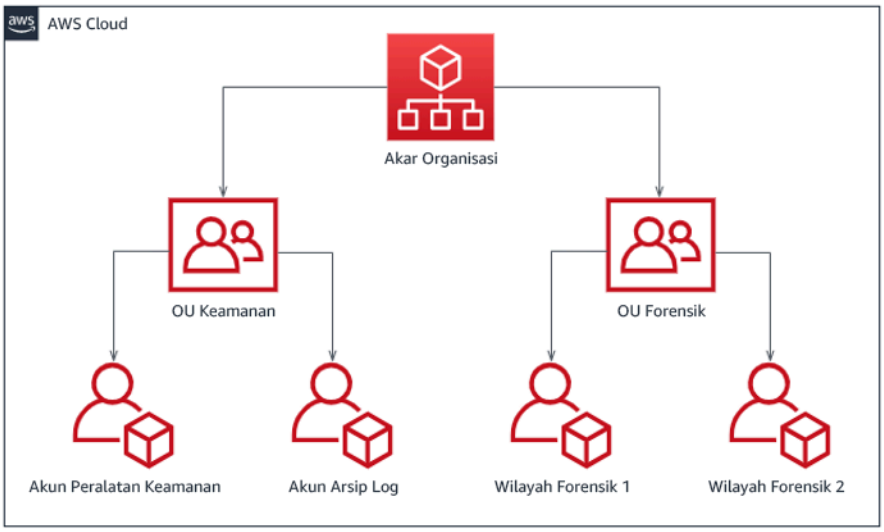
Untuk respons insiden, ada manfaatnya memiliki struktur Akun AWS yang mendukung fungsi respons insiden, yang mencakup OU Keamanan dan OU Forensik. Di dalam OU keamanan, Anda harus memiliki akun untuk:

- Pengarsipan log: Agregasikan log dalam Akun AWS pengarsipan log dengan izin terbatas.
- Alat keamanan: Pusatkan layanan keamanan di Akun AWS alat keamanan. Akun ini beroperasi sebagai administrator yang didelegasikan untuk layanan keamanan.

Dalam OU forensik, Anda memiliki opsi untuk mengimplemetasikan satu atau beberapa akun forensik untuk setiap Region tempat Anda beroperasi, tergantung mana yang paling sesuai untuk model bisnis dan operasional Anda. Jika Anda membuat akun forensik per Region, Anda dapat memblokir pembuatan sumber daya AWS di luar Region tersebut dan mengurangi risiko sumber daya disalin ke region yang tidak diinginkan. Misalnya, jika Anda hanya beroperasi di US East (N. Virginia) Region (`us-east-1`) dan US West (Oregon) (`us-west-2`), maka Anda akan memiliki dua akun di OU forensik: satu untuk `us-east-1` dan satu untuk `us-west-2`.

Anda dapat membuat Akun AWS forensik untuk beberapa Region. Anda harus berhati-hati dalam menyalin sumber daya AWS ke akun tersebut untuk memastikan keselarasan dengan persyaratan kedaulatan data Anda. Karena diperlukan waktu untuk menyediakan akun baru, akun forensik harus dibuat dan dilengkapi jauh sebelum insiden sehingga tim perespons dapat dipersiapkan untuk menggunakannya secara efektif untuk merespons.

Diagram berikut ini menampilkan contoh struktur akun termasuk OU forensik dengan akun forensik per Region:



Struktur akun per Region untuk respons insiden

Rekam cadangan dan snapshot

Menyiapkan cadangan sistem dan basis data utama sangat penting untuk pemulihan dari insiden keamanan dan untuk tujuan forensik. Dengan cadangan di tempat, Anda dapat memulihkan sistem Anda ke keadaan aman sebelumnya. Pada AWS, Anda dapat membuat snapshot berbagai sumber daya. Snapshot memberi Anda cadangan titik waktu dari sumber daya tersebut. Ada banyak layanan AWS yang dapat mendukung Anda dalam pencadangan dan pemulihan. Untuk detail tentang layanan dan pendekatan ini untuk pencadangan dan pemulihan, lihat Panduan Preskriptif Pencadangan dan Pemulihan dan Gunakan cadangan untuk memulihkan dari insiden keamanan.

Terutama berkaitan dengan situasi seperti ransomware, cadangan Anda harus terlindungi dengan baik. Untuk panduan tentang mengamankan cadangan Anda, lihat 10 praktik terbaik keamanan untuk mengamankan cadangan di AWS. Selain mengamankan cadangan Anda, Anda harus secara rutin menguji proses pencadangan dan pemulihan Anda untuk memastikan teknologi dan proses yang Anda miliki berfungsi seperti yang diharapkan.

Mengotomatiskan forensik

Selama event keamanan, tim respons insiden Anda harus dapat mengumpulkan dan menganalisis bukti dengan cepat sambil mempertahankan akurasi selama jangka waktu sebelum dan sesudah event (seperti menangkap log yang berkaitan dengan event atau sumber daya tertentu atau mengumpulkan dump memori suatu instans Amazon EC2). Ini adalah hal yang menantang dan memakan waktu bagi tim respons insiden untuk mengumpulkan bukti yang relevan secara manual, terutama di sejumlah besar instans dan akun. Selain itu, pengumpulan manual bisa rentan terhadap kesalahan manusia. Untuk alasan-alasan ini, Anda harus mengembangkan dan mengimplementasikan otomatisasi untuk forensik sebanyak mungkin.

AWS menawarkan sejumlah sumber daya otomatisasi untuk forensik, yang tercantum di bagian Sumber Daya berikut ini. Sumber daya ini adalah contoh-contoh pola forensik yang telah kami kembangkan dan telah diimplementasikan oleh pelanggan. Meskipun merupakan arsitektur referensi yang berguna untuk memulai, pertimbangkan untuk memodifikasinya atau membuat pola otomatisasi forensik baru berdasarkan lingkungan, persyaratan, alat, dan proses forensik Anda.

Sumber daya

Dokumen terkait:

- Panduan Respons Insiden Keamanan AWS - Mengembangkan Kemampuan Forensik
- Panduan Respons Insiden Keamanan AWS - Sumber Daya Forensik
- Strategi lingkungan investigasi forensik di AWS Cloud
- Cara mengotomatiskan pengumpulan disk forensik di AWS
- Panduan Preskriptif AWS - Mengotomatiskan respons insiden dan forensik

Video terkait:

- Mengotomatiskan Respons Insiden dan Forensik

Contoh terkait:

- Respons Insiden Otomatis dan Kerangka Forensik
- Orkestrator Forensik Otomatis untuk Amazon EC2

SEC10-BP04 Mengembangkan dan menguji playbook respons insiden keamanan

Bagian penting dari penyiapan proses respons insiden Anda adalah mengembangkan playbook. Playbook respons insiden memberikan serangkaian panduan preskriptif dan langkah-langkah yang harus diikuti ketika event keamanan terjadi. Memiliki struktur dan langkah yang jelas menyederhanakan respons dan mengurangi kemungkinan kesalahan manusia.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Playbook harus dibuat untuk skenario insiden seperti:

- Insiden yang diperkirakan: Playbook harus dibuat untuk insiden yang Anda antisipasi. Ini mencakup ancaman seperti denial of service (DoS), ransomware, dan kompromi kredensial.
- Temuan atau pemberitahuan keamanan yang diketahui: Playbook harus dibuat untuk temuan dan pemberitahuan keamanan Anda yang diketahui, seperti temuan . Anda mungkin menerima temuan GuardDuty dan berpikir, “Lalu apa?” Untuk mencegah kesalahan penanganan atau pengabaian temuan GuardDuty, buat playbook untuk setiap temuan GuardDuty potensial. Beberapa detail dan panduan perbaikan dapat ditemukan di dokumentasi GuardDuty. Perlu dicatat bahwa GuardDuty tidak diaktifkan secara default dan dikenakan biaya. Untuk detail selengkapnya tentang GuardDuty, lihat Lampiran A: Definisi kemampuan cloud - Visibilitas dan pembuatan pemberitahuan.

Playbook harus berisi langkah-langkah teknis untuk diselesaikan oleh analis keamanan untuk menyelidiki dan merespons insiden keamanan potensial secara memadai.

Langkah implementasi

Item yang perlu disertakan dalam playbook meliputi:

- Gambaran umum playbook: Risiko atau skenario insiden apa yang ditangani oleh playbook ini? Apa tujuan dari playbook?
- Prasyarat: Log, mekanisme deteksi, dan alat otomatis apa yang diperlukan untuk skenario insiden ini? Apa notifikasi yang diharapkan?
- Komunikasi dan informasi eskalasi: Siapa yang terlibat dan apa informasi kontak mereka? Apa tanggung jawab setiap pemangku kepentingan?
- Langkah-langkah respons: Di seluruh fase respons insiden, langkah taktis apa yang harus dilakukan? Kueri apa yang harus dijalankan analis? Kode apa yang harus dijalankan untuk mencapai hasil yang diinginkan?

  - Deteksi: Bagaimana insiden akan dideteksi?
  - Analisis: Bagaimana cakupan dampak ditentukan?
  - Membendung: Bagaimana insiden akan diisolasi untuk membatasi cakupan?
  - Memberantas: Bagaimana ancaman akan disingkirkan dari lingkungan?
  - Memulihkan: Bagaimana sistem atau sumber daya yang terdampak akan dikembalikan ke produksi?
- Hasil yang diharapkan: Setelah kueri dan kode dijalankan, apa hasil yang diharapkan dari playbook?

Sumber daya

Praktik terbaik Well-Architected terkait:

- SEC10-BP02 - Membuat rencana manajemen insiden

Dokumen terkait:

- Kerangka Kerja untuk Playbook Respons Insiden
- Mengembangkan Playbook Respons Insiden Anda sendiri
- Contoh Playbook Respons Insiden
- Membangun runbook respons insiden AWS menggunakan playbook Jupyter dan CloudTrail Lake

SEC10-BP05 Menyediakan akses di awal

Verifikasi staf respons insiden memiliki akses yang benar yang telah disediakan sebelumnya di AWS untuk mengurangi waktu yang diperlukan untuk penyelidikan hingga pemulihan.

Antipola umum:

- Menggunakan akun root untuk merespons insiden.
- Mengubah akun-akun pengguna yang ada.
- Memanipulasi izin IAM secara langsung saat menyediakan peningkatan hak akses yang sedang dibutuhkan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

AWS menyarankan Anda untuk sebisa mungkin mengurangi atau menghilangkan kebergantungan pada kredensial berumur panjang, dan memilih kredensial sementara dan mekanisme peningkatan hak akses hanya saat diperlukan. Kredensial berumur panjang rentang terkena risiko keamanan dan meningkatkan biaya overhead operasional. Untuk sebagian besar tugas manajemen, serta tugas respons insiden, kami menyarankan Anda untuk mengimplementasikan federasi identitas bersamaan dengan peningkatan sementara untuk akses administratif. Di model ini, seorang pengguna meminta peningkatan ke tingkat hak akses yang lebih tinggi (seperti peran respons insiden) dan, apabila pengguna tersebut memenuhi syarat peningkatan hak, permintaan tersebut dikirimkan ke seorang pemberi persetujuan. Jika permintaan tersebut disetujui, pengguna menerima serangkaian kredensial AWS sementara yang dapat digunakan untuk menyelesaikan tugas-tugas mereka. Setelah kredensial ini kedaluwarsa, pengguna harus mengirimkan permintaan peningkatan baru.

Kami menyarankan penggunaan peningkatan hak akses sementara di sebagian besar skenario respons insiden. Cara tepat untuk melakukannya adalah dengan menggunakan AWS Security Token Service dan kebijakan sesi untuk membuat cakupan akses.

Terdapat skenario di mana identitas terfederasi tidak tersedia, seperti:

- Pemadaman yang berkaitan dengan penyedia identitas (IdP) yang terganggu.
- Kesalahan konfigurasi atau kesalahan manusiawi yang menyebabkan rusaknya sistem manajemen akses terfederasi.
- Aktivitas berbahaya seperti peristiwa distributed denial of service (DDoS) atau yang menyebabkan sistem tidak tersedia.

Pada kasus-kasus di atas, harus terdapat akses mendesak (break glass) yang dikonfigurasi untuk mengizinkan penyelidikan dan perbaikan peristiwa secara cepat. Sebaiknya gunakan juga pengguna IAM dengan izin yang tepat untuk menjalankan tugas dan mengakses sumber daya AWS. Gunakan kredensial root hanya untuk tugas yang memerlukan akses pengguna root. Untuk memverifikasi bahwa tim respons insiden memiliki tingkat akses yang tepat ke AWS dan sistem yang relevan lainnya, sebaiknya sediakan akun-akun pengguna khusus sejak awal. Akun pengguna tersebut memerlukan akses istimewa, dan harus dikontrol dan dipantau secara ketat. Akun-akun tersebut harus dibuat dengan hak akses paling sedikit yang diperlukan untuk menjalankan tugas yang diperlukan, dan tingkat akses harus didasarkan pada playbook yang dibuat sebagai bagian dari rencana manajemen insiden.

Gunakan pengguna dan peran yang dibuat khusus sebagai praktik terbaik. Peningkatan akses pengguna atau peran sementara melalui penambahan kebijakan IAM menjadikannya tidak jelas terkait akses apa yang dimiliki pengguna selama insiden, dan terdapat risiko tidak dicabutnya peningkatan hak akses tersebut.

Penting untuk menghapus dependensi sebanyak mungkin untuk memastikan akses dapat diperoleh dalam sebanyak mungkin skenario kegagalan. Untuk mendukung hal ini, buatlah playbook untuk memastikan pengguna respons insiden dibuat sebagai pengguna AWS Identity and Access Management di dalam akun keamanan khusus, dan bukan dikelola melalui solusi Federasi atau masuk tunggal (SSO) yang ada. Tiap-tiap perespons harus memiliki akun dengan nama mereka sendiri. Konfigurasi akun harus menegakkan kebijakan kata sandi yang kuat dan autentikasi multi-faktor (MFA). Jika playbook respons insiden hanya memerlukan akses ke Konsol Manajemen AWS, pengguna tidak boleh memiliki kunci akses yang dikonfigurasi dan harus dilarang secara tegas untuk membuat kunci akses. Hal ini dapat dikonfigurasi dengan kebijakan IAM atau kebijakan kontrol layanan (SCP) sebagaimana disebutkan dalam Praktik Terbaik Keamanan AWS untu AWS Organizations SCP. Pengguna tidak boleh memiliki hak ases selain kemampuan untuk mengambil peran respons insiden di akun-akun lainnya.

Selama insiden, mungkin diperlukan pemberian akses ke individu internal atau eksternal untuk mendukung aktivitas penyelidikan, perbaikan, atau pemulihan. Pada kasus ini, gunakan mekanisme playbook yang disebutkan sebelumnya, dan harus ada proses untuk memverifikasi bahwa akses tambahan apa pun segera dicabut setelah insiden selesai.

Untuk memastikan bahwa penggunaan peran respons insiden dapat dipantau dan diaudit dengan layak, penting untuk tidak membagikan akun pengguna IAM yang dibuat untuk tujuan ini kepada individu lain, serta tidak menggunakan pengguna root Akun AWS kecuali diperlukan untuk tugas tertentu. Jika pengguna root diperlukan (sebagai contoh, akses IAM ke akun tertentu tidak tersedia), gunakan proses terpisah dengan playbook yang tersedia untuk memverifikasi ketersediaan kata sandi pengguna root dan token MFA.

Untuk mengonfigurasi kebijakan IAM untuk peran respons insiden, pertimbangkan menggunakan IAM Access Analyzer untuk menghasilkan kebijakan berdasarkan log AWS CloudTrail. Untuk melakukannya, berikan akses administrator ke peran respons insiden di akun non-produksi dan jalankan playbook Anda. Setelah selesai, kebijakan dapat dibuat yang hanya mengizinkan tindakan yang diambil. Kebijakan ini kemudian dapat diterapkan ke semua peran respons insiden di semua akun. Anda mungkin ingin membuat kebijakan IAM terpisah untuk setiap playbook untuk mempermudah manajemen dan audit. Contoh playbook dapat mencakup rencana respons untuk ransomware, pembobolan data, hilangnya akses produksi, dan skenario lain.

Gunakan akun pengguna respons insiden untuk mengambil peran IAM respons insiden khusus di Akun AWS lain. Peran-peran ini harus dikonfigurasi hanya agar dapat diambil oleh pengguna di akun keamanan, dan hubungan kepercayaan harus mewajibkan bahwa principal pemanggil telah mengautentikasi menggunakan MFA. Peran-peran tersebut harus menggunakan kebijakan IAM dengan cakupan yang ketat untuk mengontrol akses. Pastikan bahwa semua permintaan `AssumeRole` untuk peran-peran ini dicatat dalam log di CloudTrail dan dibuatkan peringatan, dan bahwa tindakan apa pun yang diambil menggunakan peran-peran ini dicatat dalam log.

Sangat disarankan akun pengguna IAM serta IAM role disebutkan secara jelas agar dapat ditemukan dengan mudah di log CloudTrail. Contohnya adalah dengan menamai akun IAM dengan *<USER_ID>*`-BREAK-GLASS` dan IAM role dengan `BREAK-GLASS-ROLE`.

CloudTrail digunakan untuk membuat log aktivitas API di akun AWS Anda dan harus digunakan untuk mengonfigurasi peringatan penggunaan peran respons insiden. Lihat postingan blog tentang konfigurasi perintanan saat kunci root digunakan. Instruksi dapat dimodifikasi untuk mengonfigurasi filter-ke-filter metrik Amazon CloudWatch pada peristiwa `AssumeRole` terkait dengan IAM role respons insiden:

```
{ $.eventName = "AssumeRole" && $.requestParameters.roleArn =
 "<INCIDENT_RESPONSE_ROLE_ARN>" && $.userIdentity.invokedBy NOT EXISTS && $.eventType !
= "AwsServiceEvent" }
```

Karena peran respons insiden kemungkinan memiliki tingkat akses yang tinggi, peringatan-peringatan ini harus menjangkau grup yang luas dan ditindaklanjuti segera.

Selama insiden, terdapat kemungkinan bahwa perespons mungkin memerlukan akses ke sistem yang tidak diamankan secara langsung oleh IAM. Sistem-sistem tersebut dapat mencakup instans Amazon Elastic Compute Cloud, basis data Amazon Relational Database Service, atau platform perangkat lunak sebagai layanan (SaaS). Sangat disarankan untuk tidak menggunakan protokol native seperti SSH atau RDP, melainkan AWS Systems Manager Session Manager untuk semua akses administratif ke instans Amazon EC2. Akses ini dapat dikontrol menggunakan IAM, yang aman dan diaudit. Memungkinkan juga untuk mengotomatisasi bagian-bagian playbook Anda menggunakan dokumen AWS Systems Manager Run Command, yang dapat mengurangi kesalahan pengguna dan mempercepat waktu pemulihan. Untuk akses ke basis data dan alat-alat pihak ketiga, kami sarankan menyimpan kredensial akses di AWS Secrets Manager dan memberikan akses ke peran perespons insiden.

Terakhir, manajemen akun pengguna IAM perespons insiden harus ditambahkan ke proses Joiners, Movers, dan Leavers Anda dan ditinjau serta diuji secara berkala untuk memastikan bahwa yang diizinkan hanyalah akses yang diinginkan.

Sumber daya

Dokumen terkait:

- Mengelola peningkatan akses sementara ke lingkungan AWS Anda
- Panduan Respons Insiden Keamanan AWS
- AWS Elastic Disaster Recovery
- Manajer Insiden AWS Systems Manager
- Mengatur kebijakan kata sandi akun untuk pengguna IAM
- Menggunakan autentikasi multi-faktor (MFA) di AWS
- Mengonfigurasi Akses Lintas Akun dengan MFA
- Menggunakan IAM Access Analyzer untuk menghasilkan kebijakan IAM
- Praktik Terbaik untuk Kebijakan Kontrol Layanan AWS Organizations di Lingkungan Multi-akun
- Cara Menerima Notifikasi Ketika Kunci Akses Root Akun AWS Anda Digunakan
- Membuat izin sesi mendetail menggunakan kebijakan terkelola IAM

Video terkait:

- Mengotomatiskan Respons Insiden dan Forensik di AWS
- Panduan mandiri untuk runbook, laporan insiden, dan respons insiden
- Bersiap dan merespons insiden keamanan di lingkungan AWS Anda

Contoh terkait:

- Lab: Penyiapan dan Pengguna Root Akun AWS
- Lab: Respons insiden dengan Konsol AWS dan CLI

SEC10-BP06 Melakukan deployment alat di awal

Pastikan personel keamanan sejak awal telah melakukan deployment alat yang tepat untuk mengurangi waktu investigasi melalui pemulihan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Untuk mengotomatiskan respons keamanan dan fungsi operasi, Anda dapat menggunakan set API dan alat yang komprehensif dari AWS. Anda dapat sepenuhnya mengotomatiskan manajemen identitas, keamanan jaringan, perlindungan data, dan kemampuan pemantauan, serta menyediakannya menggunakan metode pengembangan perangkat lunak populer yang sudah Anda gunakan. Saat Anda membangun otomatisasi keamanan, sistem Anda dapat memantau, meninjau, dan menginisiasi respons, tanpa memerlukan orang untuk memantau posisi keamanan Anda dan memberikan reaksi terhadap peristiwa secara manual.

Jika tim respons insiden Anda terus merespons peringatan dengan cara yang sama, mereka berisiko mengalami kelelahan alarm (alarm fatigue). Seiring berjalannya waktu, tim dapat menjadi tidak peka terhadap peringatan sehingga dapat membuat kesalahan saat menangani situasi biasa atau melewatkan peringatan yang tidak biasa. Otomatisasi membantu mencegah kelelahan alarm dengan menggunakan fungsi yang memproses peringatan biasa dan repetitif, sehingga manusia cukup menangani insiden yang sensitif dan unik. Integrasi sistem deteksi anomali, seperti Amazon GuardDuty, Wawasan AWS CloudTrail, dan Deteksi Anomali Amazon CloudWatch, dapat mengurangi beban dari peringatan umum berbasis ambang batas.

Anda dapat memperbaiki proses manual dengan mengotomatiskan langkah-langkah dalam proses secara terprogram. Setelah Anda menentukan perbaikan pola pada peristiwa, Anda dapat menguraikan pola tersebut menjadi logika yang dapat ditindaklanjuti, dan menulis kode untuk menjalankan logika tersebut. Pemberi respons selanjutnya dapat menjalankan kode tersebut untuk memperbaiki masalah. Seiring berjalannya waktu, Anda dapat mengotomatiskan lebih banyak langkah, dan pada akhirnya secara otomatis menangani semua jenis insiden yang biasa muncul.

Selama penyelidikan keamanan, Anda harus dapat meninjau log yang relevan untuk mencatat dan memahami seluruh cakupan serta garis waktu insiden. Log juga diperlukan untuk pembuatan peringatan yang mengindikasikan bahwa tindakan tertentu telah terjadi. Sangat penting untuk memilih, mengaktifkan, menyimpan, dan menyiapkan mekanisme kueri dan pengambilan, serta pembuatan peringatan. Selain itu, cara yang efektif untuk menyediakan alat untuk mencari data log adalah Amazon Detective.

AWS menawarkan lebih dari 200 layanan cloud dan ribuan fitur. Kami menyarankan Anda meninjau layanan yang dapat mendukung dan menyederhanakan strategi respons insiden Anda.

Selain pencatatan log, Anda harus mengembangkan dan menerapkan strategi pemberian tag. Pemberian tag dapat membantu memberikan konteks seputar tujuan sebuah sumber daya AWS. Pemberian tag juga dapat digunakan untuk otomatisasi.

Langkah implementasi

Pilih dan atur log untuk analisis dan pembuatan peringatan

Lihat dokumentasi berikut tentang cara mengonfigurasi pencatatan log untuk respons insiden:

- Strategi pencatatan log untuk respons insiden keamanan
- SEC04-BP01 Mengonfigurasi pencatatan log layanan dan aplikasi

Aktifkan layanan keamanan untuk mendukung deteksi dan respons

AWS menyediakan kemampuan deteksi, pencegahan, dan responsif asli, dan layanan lainnya dapat digunakan untuk merancang solusi keamanan khusus. Untuk daftar layanan yang paling relevan untuk respons insiden keamanan, lihat Definisi kemampuan cloud.

Mengembangkan dan menerapkan strategi pemberian tag

Memperoleh informasi kontekstual tentang kasus penggunaan bisnis dan pemangku kepentingan internal yang relevan seputar sumber daya AWS bisa jadi sulit dilakukan. Salah satu cara untuk melakukannya adalah dalam bentuk tag, yang menetapkan metadata ke sumber daya AWS Anda dan terdiri dari kunci dan nilai yang ditentukan pengguna. Anda dapat membuat tag untuk mengategorikan sumber daya berdasarkan tujuan, pemilik, lingkungan, jenis data yang diproses, dan kriteria lain pilihan Anda.

Memiliki strategi pemberian tag yang konsisten dapat mempercepat waktu respons dan meminimalkan waktu yang dihabiskan untuk konteks organisasi dengan memungkinkan Anda mengidentifikasi dan membedakan informasi kontekstual tentang sumber daya AWS dengan cepat. Tag juga dapat berfungsi sebagai mekanisme untuk memulai otomatisasi respons. Untuk detail selengkapnya tentang apa yang harus diberi tag, lihat Memberi tag pada sumber daya AWS Anda. Anda harus terlebih dahulu menentukan tag yang ingin Anda terapkan di seluruh organisasi Anda. Setelah itu, Anda akan menerapkan dan menegakkan strategi pemberian tag ini. Untuk detail selengkapnya tentang implementasi dan penegakan, lihat Menerapkan strategi pemberian tag sumber daya AWS menggunakan Kebijakan Tag dan Kebijakan Kontrol Layanan (SCP) AWS.

Sumber daya

Praktik terbaik Well-Architected terkait:

- SEC04-BP01 Mengonfigurasi pencatatan log layanan dan aplikasi
- SEC04-BP02 Menganalisis log, temuan, dan metrik secara terpusat

Dokumen terkait:

- Strategi pencatatan log untuk respons insiden keamanan
- Definisi kemampuan cloud respons insiden

Contoh terkait:

- Deteksi dan Respons Ancaman dengan Amazon GuardDuty dan Amazon Detective
- Lokakarya Security Hub
- Manajemen Kerentanan dengan Amazon Inspector

SEC10-BP07 Menjalankan simulasi

Organisasi tumbuh dan berkembang dari waktu ke waktu, begitu juga dengan lanskap ancaman. Oleh karena itu, penting untuk terus-menerus mengkaji kemampuan Anda dalam merespons insiden. Menjalankan simulasi (juga dikenal dengan nama game day) adalah salah satu metode yang dapat digunakan untuk melakukan penilaian ini. Simulasi menggunakan skenario peristiwa keamanan dunia nyata yang dirancang untuk meniru taktik, teknik, dan prosedur (TTP) pelaku ancaman dan memungkinkan organisasi untuk melatih dan mengevaluasi kemampuan respons insiden mereka dengan merespons peristiwa dunia maya tiruan ini, yang bisa saja benar-benar terjadi di dunia nyata.

Manfaat menerapkan praktik terbaik ini: Simulasi memiliki berbagai manfaat:

- Memvalidasi kesiapan dunia maya dan membangun kepercayaan diri perespons insiden Anda.
- Menguji akurasi dan efisiensi alat dan alur kerja.
- Menyempurnakan metode komunikasi dan eskalasi yang selaras dengan rencana respons insiden Anda.
- Memberikan kesempatan untuk merespons vektor-vektor yang kurang umum.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Terdapat tiga jenis simulasi utama:

- Latihan simulasi meja: Pendekatan simulasi meja adalah sesi berbasis diskusi yang melibatkan berbagai pemangku kepentingan respons insiden untuk melatih peran dan tanggung jawab menggunakan alat komunikasi dan playbook yang lazim. Fasilitasi latihan umumnya dapat dilakukan selama sehari penuh di lokasi virtual, lokasi fisik, atau gabungan keduanya. Karena berbasis diskusi, latihan meja berfokus pada proses, orang, dan kolaborasi. Teknologi merupakan bagian tak terpisahkan dari diskusi, tetapi penggunaan nyata alat atau skrip respons insiden umumnya bukan bagian dari latihan meja.
- Latihan tim ungu: Latihan tim ungu meningkatkan level kolaborasi antara perespons insiden (tim biru) dan yang bermain sebagai pelaku ancaman (tim merah). Tim biru terdiri dari anggota pusat operasi keamanan (SOC), tetapi juga bisa melibatkan pemangku kepentingan lain yang akan terlibat selama peristiwa dunia maya nyata. Tim merah terdiri dari tim uji penetrasi atau pemangku kepentingan utama yang terlatih dalam hal keamanan ofensif. Tim merah bekerja secara kolaboratif dengan fasilitator latihan saat merancang skenario sehingga skenario benar-benar akurat dan layak. Selama latihan tim ungu, fokus utamanya adalah mekanisme deteksi, alat, dan prosedur operasi standar (SOP) yang mendukung upaya respons insiden.
- Latihan tim merah: Selama latihan tim merah, pelanggaran (tim merah) melakukan simulasi untuk mencapai satu tujuan atau serangkaian tujuan tertentu dari ruang lingkup yang telah ditentukan. Para pelindung (tim biru) tidak harus mengetahui ruang lingkup dan durasi latihan, sehingga penilaian menjadi lebih realistis tentang cara mereka merespons insiden yang sebenarnya. Karena latihan tim merah bisa bersifat invasif, berhati-hatilah dan terapkan kontrol untuk memverifikasi bahwa latihan tidak menyebabkan kerusakan nyata pada lingkungan Anda.

Pertimbangkan untuk memfasilitasi simulasi dunia maya secara berkala. Setiap jenis latihan dapat memberikan manfaat unik bagi peserta dan organisasi secara keseluruhan, sehingga Anda dapat memilih untuk memulai dengan jenis simulasi yang lebih sederhana (seperti latihan meja) dan secara bertahap beralih ke jenis simulasi yang lebih kompleks (latihan tim merah). Anda harus memilih jenis simulasi berdasarkan kematangan keamanan dan sumber daya Anda, serta hasil yang Anda inginkan. Beberapa pelanggan mungkin tidak mau melakukan latihan tim merah disebabkan kompleksitas dan biayanya.

Langkah implementasi

Terlepas dari jenis simulasi yang Anda pilih, simulasi umumnya mengikuti langkah-langkah implementasi berikut:

1. Tentukan elemen latihan inti: Tentukan skenario simulasi dan tujuan simulasi. Keduanya harus atas izin pimpinan.
2. Identifikasi pemangku kepentingan utama: Minimal, latihan membutuhkan fasilitator dan peserta latihan. Tergantung skenario, pemangku kepentingan tambahan seperti pimpinan tim legal, komunikasi, atau eksekutif dapat dilibatkan.
3. Bangun dan uji skenario: Skenario mungkin perlu dirombak selama penyusunan apabila elemen tertentu tidak layak. Skenario akhir adalah output yang diharapkan pada tahap ini.
4. Fasilitasi simulasi: Jenis simulasi menentukan fasilitas yang digunakan (skenario berbasis kertas versus skenario tersimulasi yang sangat teknis). Fasilitator harus menyelaraskan taktik fasilitas mereka dengan objek latihan dan mereka harus sebisa mungkin melibatkan semua peserta latihan untuk memberikan manfaat paling optimal.
5. Kembangkan laporan pascatindakan (AAR): Identifikasi area yang berjalan dengan baik, area yang memerlukan perbaikan, dan potensi celah. AAR harus mengukur efektivitas simulasi serta respons tim terhadap simulasi peristiwa sehingga kemajuan dapat dilacak dari waktu ke waktu dengan simulasi di masa mendatang.

Sumber daya

Dokumen terkait:

- Panduan Respons Insiden AWS

Video terkait:

- AWS GameDay - Edisi Keamanan

SEC10-BP08 Menetapkan kerangka kerja untuk belajar dari insiden

Menerapkan kerangka kerja belajar dari pengalaman dan kemampuan analisis akar masalah tidak hanya dapat membantu meningkatkan kemampuan respons insiden, tetapi juga membantu mencegah insiden berulang. Dengan belajar dari setiap kejadian, Anda dapat membantu menghindari mengulangi kesalahan, paparan, atau kesalahan konfigurasi yang sama, sehingga tidak hanya

meningkatkan postur keamanan Anda, tetapi juga meminimalkan waktu yang hilang untuk situasi yang dapat dicegah.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Penting untuk menerapkan kerangka kerja belajar dari pengalaman yang secara umum menetapkan dan mencapai poin-poin berikut ini:

- Kapan belajar dari pengalaman diadakan?
- Apa yang terlibat dalam proses belajar dari pengalaman?
- Bagaimana belajar dari pengalaman dilaksanakan?
- Siapa yang terlibat dalam prosesnya, dan bagaimana?
- Bagaimana area perbaikan akan diidentifikasi?
- Bagaimana Anda akan memastikan bawa perbaikan dilacak dan diimplementasikan secara efektif?

Kerangka kerja ini tidak boleh fokus pada individu atau menyalahkan individu, tetapi harus fokus pada perbaikan alat dan proses.

Langkah implementasi

Selain hasil tingkat tinggi yang dicantumkan sebelumnya, penting untuk memastikan bahwa Anda mengajukan pertanyaan yang tepat untuk mendapatkan nilai paling besar (informasi yang mengarah pada perbaikan yang dapat ditindaklanjuti) dari proses tersebut. Pertimbangkan pertanyaan-pertanyaan ini untuk membantu Anda memulai dalam membangun diskusi belajar dari pengalaman Anda:

- Apa insidennya?
- Kapan insiden tersebut pertama kali diidentifikasi?
- Bagaimana insiden tersebut diidentifikasi?
- Sistem apa yang diperingatkan tentang aktivitas tersebut?
- Sistem, layanan, dan data apa yang terlibat?
- Apa yang terjadi secara khusus?
- Apa yang berjalan dengan baik?

- Apa yang tidak berjalan dengan baik?
- Proses atau prosedur mana yang gagal atau gagal diskalakan untuk merespons insiden tersebut?
- Apa yang dapat diperbaiki dalam area-area berikut:
  - Personel
    - Apakah orang-orang yang perlu dihubungi benar-benar sedang lowong dan apakah daftar kontak sudah diperbarui?
    - Apakah orang-orang melewatkan pelatihan atau kemampuan yang diperlukan untuk merespons dan menyelidiki insiden tersebut secara efektif?
    - Apakah sumber daya yang tepat siap dan tersedia?
  - Proses
    - Apakah proses dan prosedur diikuti?
    - Apakah proses dan prosedur didokumentasikan dan tersedia untuk (jenis) insiden ini?
    - Apakah proses dan prosedur yang diperlukan tidak tersedia?
    - Apakah perespons dapat memperoleh akses tepat waktu ke informasi yang diperlukan untuk merespons masalah ini?
  - Teknologi
    - Apakah sistem peringatan yang ada secara efektif mengidentifikasi dan memperingatkan tentang aktivitas?
    - Bagaimana kita bisa mengurangi waktu deteksi hingga 50%?
    - Apakah peringatan yang ada memerlukan perbaikan atau peringatan baru perlu dibuat untuk (jenis) insiden ini?
    - Apakah alat yang ada memungkinkan penyelidikan (pencarian/analisis) insiden yang efektif?
    - Apa yang dapat dilakukan untuk membantu mengidentifikasi (jenis) insiden ini dengan lebih cepat?
    - Apa yang dapat dilakukan untuk membantu mencegah (jenis) insiden ini terjadi lagi?
    - Siapa yang memiliki rencana perbaikan dan bagaimana Anda akan menguji bahwa rencana tersebut telah diterapkan?
    - Bagaimana garis waktu untuk mengimplementasikan dan menguji pemantauan tambahan atau kontrol dan proses pencegahan?

Daftar ini bukanlah daftar lengkap, melainkan dimaksudkan sebagai titik awal untuk mengidentifikasi kebutuhan organisasi dan bisnis dan bagaimana Anda dapat menganalisisnya agar dapat belajar

secara efektif dari insiden dan terus meningkatkan postur keamanan Anda. Yang paling penting adalah memulai dengan menyertakan kerangka kerja belajar dari pengalaman sebagai bagian standar dari proses respons insiden, dokumentasi, dan harapan seluruh pemangku kepentingan.

Sumber daya

Dokumen terkait:

- Panduan Respons Insiden Keamanan AWS - Menetapkan kerangka kerja untuk belajar dari insiden
- Panduan NCSC CAF - Belajar dari pengalaman

Keamanan aplikasi

Pertanyaan

- SEC 11. Bagaimana cara menyertakan dan memvalidasi karakteristik keamanan aplikasi sepanjang siklus hidup desain, pengembangan, dan deployment?

SEC 11. Bagaimana cara menyertakan dan memvalidasi karakteristik keamanan aplikasi sepanjang siklus hidup desain, pengembangan, dan deployment?

Melatih karyawan, menguji menggunakan otomatisasi, memahami dependensi, dan memvalidasi karakteristik keamanan alat dan aplikasi akan membantu mengurangi kemungkinan masalah keamanan dalam beban kerja produksi.

Praktik terbaik

- SEC11-BP01 Pelatihan untuk keamanan aplikasi
- SEC11-BP02 Otomatiskan pengujian sepanjang siklus hidup pengembangan dan rilis
- SEC11-BP03 Lakukan uji penetrasi secara teratur
- SEC11-BP04 Peninjauan kode manual
- SEC11-BP05 Pusatkan layanan untuk paket dan dependensi
- SEC11-BP06 Lakukan deployment perangkat lunak secara terprogram
- SEC11-BP07 Nilai karakteristik keamanan pipeline secara teratur
- SEC11-BP08 Buat program yang menanamkan kepemilikan keamanan dalam tim beban kerja

SEC11-BP01 Pelatihan untuk keamanan aplikasi

Berikan pelatihan kepada builder dalam organisasi Anda mengenai praktik umum untuk pengembangan dan pengoperasian aplikasi yang aman. Adopsi praktik pengembangan yang berfokus pada keamanan akan membantu mengurangi kemungkinan munculnya masalah yang hanya terdeteksi pada tahap peninjauan keamanan.

Hasil yang diinginkan: Perangkat lunak harus didesain dan dikembangkan dengan mempertimbangkan keamanan. Saat builder di sebuah organisasi berlatih praktik pengembangan aman yang dimulai dengan model ancaman, langkah ini meningkatkan keseluruhan kualitas dan keamanan perangkat lunak yang dibuat. Pendekatan ini dapat mempersingkat waktu untuk mengirimkan perangkat lunak atau fitur karena tidak perlu banyak pengerjaan ulang setelah tahap peninjauan keamanan.

Untuk tujuan praktik terbaik ini, pengembangan aman merujuk pada perangkat lunak yang sedang ditulis dan alat atau sistem yang mendukung siklus hidup pengembangan perangkat lunak (SDLC).

Antipola umum:

- Menunggu sampai peninjauan keamanan, lalu mempertimbangkan karakteristik keamanan sistem.
- Menyerahkan semua keputusan keamanan kepada tim keamanan.
- Gagal menyampaikan cara keputusan diambil di SDLC terkait ekspektasi keseluruhan keamanan atau kebijakan organisasi.
- Terlambat melibatkan diri dalam proses peninjauan keamanan.

Manfaat menjalankan praktik terbaik ini:

- Memiliki pengetahuan yang lebih baik seputar persyaratan organisasi untuk keamanan pada fase awal siklus pengembangan.
- Dapat mengidentifikasi dan mengatasi potensi masalah keamanan lebih cepat, sehingga dapat mengirim fitur lebih cepat.
- Peningkatan kualitas perangkat lunak dan sistem.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Sediakan pelatihan kepada builder di organisasi Anda. Memulai dengan kursus pemodelan ancaman adalah dasar yang baik guna membantu pelatihan untuk keamanan. Idealnya, builder harus dapat mengakses secara mandiri informasi yang relevan dengan beban kerja mereka. Akses ini membantu mereka mengambil keputusan yang tepat tentang karakteristik keamanan sistem yang mereka bangun tanpa perlu bertanya kepada tim lain. Proses melibatkan tim keamanan untuk peninjauan harus diatur dengan jelas dan mudah diikuti. Langkah-langkah di proses peninjauan harus disertakan dalam pelatihan keamanan. Jika tersedia templat atau pola implementasi yang diketahui, keduanya harus mudah dicari dan berhubungan dengan persyaratan keamanan keseluruhan. Pertimbangkan untuk menggunakan AWS CloudFormation, AWS Cloud Development Kit (AWS CDK) Constructs, Service Catalog, atau alat pembuat templat lainnya untuk mengurangi kebutuhan akan konfigurasi kustom.

Langkah implementasi

- Mulai latih builder dengan memberikan kursus terkait pemodelan ancaman untuk membangun dasar yang baik, dan bimbing mereka untuk mengetahui cara memikirkan tentang keamanan.
- Berikan akses ke AWS Training dan Sertifikasi, industri, atau pelatihan Partner AWS.
- Berikan pelatihan terkait proses peninjauan keamanan organisasi Anda, yang menguraikan pembagian tanggung jawab antara tim keamanan, tim beban kerja, dan pemegang kepentingan lainnya.
- Publikasikan panduan layanan mandiri terkait cara memenuhi persyaratan keamanan Anda, termasuk templat dan contoh kode, jika tersedia.
- Dapatkan umpan balik secara rutin dari tim builder terkait pengalaman mereka seputar pelatihan dan proses peninjauan keamanan, dan gunakan umpan balik tersebut untuk meningkatkan kualitasnya.
- Gunakan kampanye game day atau bug bash untuk membantu menurunkan jumlah masalah, dan mengasah kemampuan builder Anda.

Sumber daya

Praktik Terbaik Terkait:

- SEC11-BP08 Buat program yang menanamkan kepemilikan keamanan dalam tim beban kerja

Dokumen terkait:

- AWS Training dan Sertifikasi
- Cara berpikir tentang tata kelola keamanan cloud
- Cara memanfaatkan pemodelan ancaman
- Mempercepat pelatihan – AWS Skills Guild

Video terkait:

- Keamanan proaktif: Pertimbangan dan pendekatan

Contoh terkait:

- Lokakarya tentang pemodelan ancaman
- Kesadaran industri untuk developer

Layanan terkait:

- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK) (AWS CDK) Constructs
- Service Catalog
- AWS BugBust

SEC11-BP02 Otomatiskan pengujian sepanjang siklus hidup pengembangan dan rilis

Otomatiskan pengujian untuk karakteristik keamanan sepanjang siklus hidup pengembangan dan rilis. Otomatisasi mempermudah identifikasi yang konsisten dan berulang atas potensi masalah dalam perangkat lunak sebelum rilis, yang mengurangi risiko masalah keamanan dalam perangkat lunak yang disediakan.

Hasil yang diinginkan: Tujuan pengujian otomatis adalah menyediakan cara terprogram dalam mendeteksi potensi masalah lebih dini dan lebih sering sepanjang siklus hidup pengembangan. Saat Anda mengotomatiskan pengujian regresi, Anda dapat menjalankan kembali pengujian fungsional dan nonfungsional untuk memastikan bahwa perangkat lunak yang diuji sebelumnya masih berfungsi seperti yang diharapkan setelah perubahan. Saat Anda mendefinisikan pengujian unit keamanan untuk memeriksa apakah ada kesalahan konfigurasi umum, seperti autentikasi yang rusak atau hilang, Anda dapat mengidentifikasi dan memperbaiki masalah ini lebih dini dalam proses pengembangan.

Otomatisasi pengujian menggunakan kasus pengujian yang dibuat berdasarkan tujuan untuk validasi aplikasi, berdasarkan persyaratan aplikasi dan fungsionalitas yang diinginkan. Hasil pengujian otomatis berdasarkan perbandingan output pengujian yang dibuat dengan output yang diharapkan, sehingga mempercepat keseluruhan siklus hidup pengujian. Metodologi pengujian seperti pengujian regresi dan rangkaian pengujian unit adalah pilihan yang terbaik untuk otomatisasi. Otomatisasi pengujian karakteristik keamanan memungkinkan builder menerima umpan balik otomatis tanpa harus menunggu peninjauan keamanan. Pengujian otomatis dalam bentuk analisis kode statis atau dinamis dapat meningkatkan kualitas kode dan membantu mendeteksi potensi masalah perangkat lunak lebih dini dalam siklus hidup pengembangan.

Antipola umum:

- Tidak menyampaikan kasus pengujian dan hasil pengujian dari pengujian otomatis.
- Hanya menjalankan pengujian otomatis segera sebelum rilis.
- Mengotomatiskan kasus pengujian dengan berulang kali mengubah persyaratan.
- Gagal memberikan panduan mengenai cara menangani hasil pengujian keamanan.

Manfaat menjalankan praktik terbaik ini:

- Menurunkan dependensi pada orang yang mengevaluasi karakteristik keamanan sistem.
- Memiliki temuan yang konsisten di beberapa aliran kerja meningkatkan konsisten.
- Menurunkan kemungkinan munculnya masalah keamanan dalam produksi perangkat lunak.
- Periode waktu lebih pendek antara deteksi dan penyelesaian karena mengidentifikasi masalah perangkat lunak lebih dini.
- Meningkatkan visibilitas perilaku sistemik atau berulang di beberapa aliran kerja, yang dapat digunakan untuk mendorong peningkatan berskala organisasi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Saat Anda membuat perangkat lunak, adopsi beragam mekanisme untuk pengujian perangkat lunak guna memastikan bahwa Anda menguji aplikasi untuk kedua persyaratan aplikasi, berdasarkan logika bisnis aplikasi, dan persyaratan nonfungsional, yang fokus pada keandalan, performa, dan keamanan aplikasi.

Pengujian keamanan aplikasi statis (SAST) menganalisis kode sumber Anda untuk mendeteksi pola keamanan anomali dan memberikan indikasi untuk kode yang rawan cacat. SAST mengandalkan input statis, seperti dokumentasi (spesifikasi persyaratan, dokumentasi desain, dan spesifikasi desain) dan sumber kode aplikasi untuk menguji beragam masalah keamanan yang diketahui. Penganalisis kode statis dapat membantu mempercepat analisis kode dalam volume besar. NIST Quality Group menyediakan perbandingan Penganalisis Keamanan Kode Sumber, yang menyertakan alat sumber terbuka untuk Pemindai Kode Bita dan Pemindai Kode Biner.

Lengkapi pengujian statis Anda dengan metodologi pengujian keamanan analisis dinamis (DAST), yang menjalankan pengujian terhadap aplikasi yang berjalan untuk mengidentifikasi potensi perilaku yang tidak diharapkan. Pengujian dinamis dapat digunakan untuk mendeteksi potensi masalah yang tidak terdeteksi melalui analisis statis. Pengujian di tahap repositori kode, build, dan pipeline memungkinkan Anda memeriksa berbagai jenis potensi masalah agar tidak masuk ke dalam kode Anda. Amazon CodeWhisperer menyediakan rekomendasi kode, termasuk pemindaian keamanan, di IDE builder. Amazon CodeGuru Reviewer dapat mengidentifikasi masalah penting, masalah keamanan, dan bug yang sulit ditemukan selama pengembangan aplikasi, dan menyediakan rekomendasi untuk meningkatkan kualitas kode.

Lokakarya Keamanan untuk Developer menggunakan alat developer AWS, seperti AWS CodeBuild, AWS CodeCommit, dan AWS CodePipeline, untuk otomatisasi pipeline rilis yang menyertakan metodologi pengujian SAST dan DAST.

Saat Anda menjalani SDLC, buat proses iteratif yang menyertakan peninjauan aplikasi berkala bersama tim keamanan Anda. Umpan balik yang didapatkan dari peninjauan keamanan ini harus diatasi dan divalidasi sebagai bagian dari peninjauan kesiapan rilis Anda. Tinjauan ini membuat postur keamanan aplikasi yang kokoh, dan memberikan umpan balik yang dapat ditindaklanjuti kepada builder untuk menangani potensi masalah.

Langkah implementasi

- Implementasikan IDE yang konsisten, peninjauan kode, dan alat CI/CD yang menyertakan pengujian keamanan.
- Pertimbangkan posisi yang tepat dalam SDLC untuk memblokir pipeline daripada hanya memberi tahu builder bahwa masalah perlu diselesaikan.
- Lokakarya Keamanan untuk Developer memberikan contoh mengintegrasikan pengujian statis dan dinamis ke dalam pipeline rilis.
- Menjalankan pengujian atau analisis kode menggunakan alat otomatis, seperti Amazon CodeWhisperer yang diintegrasikan dengan IDE developer, dan Amazon CodeGuru Reviewer

untuk memindai kode dalam penerapan, membantu builder mendapatkan umpan balik pada waktu yang tepat.

- Saat membangun menggunakan AWS Lambda, Anda dapat menggunakan Amazon Inspector untuk memindai kode aplikasi dalam fungsi Anda.
- Lokakarya CI/CD AWS menyediakan titik awal dalam membangun pipeline CI/CD pada AWS.
- Saat pengujian otomatis disertakan dalam pipeline CI/CD, Anda harus menggunakan sistem tiket untuk melacak notifikasi dan penyelesaian masalah perangkat lunak.
- Untuk pengujian keamanan yang mungkin menghasilkan temuan, menautkan ke panduan untuk penyelesaian membantu builder meningkatkan kualitas kode.
- Analisis temuan secara berkala dari alat otomatis untuk memprioritaskan otomatisasi berikutnya, pelatihan builder, atau kampanye kesadaran.

Sumber daya

Dokumen terkait:

- Pengiriman Berkelanjutan dan Deployment Berkelanjutan

- Partner Kompetensi DevOps AWS
- Partner Kompetensi Keamanan AWS untuk Keamanan Aplikasi
- Memilih pendekatan CI/CD Well-Architected
- Memantau peristiwa CodeCommit di Amazon EventBridge dan Amazon CloudWatch Events
- Deteksi rahasia dalam Peninjauan Amazon CodeGuru
- Percepat deployment di AWS dengan tata kelola yang efektif
- Cara AWS memanfaatkan otomatisasi deployment yang aman tanpa campur tangan

Video terkait:

- Tanpa campur tangan: Mengotomatiskan pipeline pengiriman berkelanjutan di Amazon
- Mengotomatiskan pipeline CI/CD lintas akun

Contoh terkait:

- Kesadaran industri untuk developer

- Tata Kelola AWS CodePipeline (GitHub)
- Lokakarya Keamanan untuk Developer
- Lokakarya CI/CD AWS

SEC11-BP03 Lakukan uji penetrasi secara teratur

Lakukan uji penetrasi perangkat lunak secara teratur. Mekanisme ini membantu mengidentifikasi potensi masalah perangkat lunak yang tidak dapat dideteksi oleh pengujian otomatis atau peninjauan kode manual. Mekanisme ini juga membantu Anda memahami efikasi kontrol deteksi Anda. Uji penetrasi harus mencoba untuk menentukan apakah perangkat lunak dapat dibuat untuk berkinerja dengan cara-cara yang tak terduga, seperti mengungkapkan data yang seharusnya dilindungi, atau memberikan izin yang lebih luas daripada yang diharapkan.

Hasil yang diinginkan: Uji penetrasi digunakan untuk mendeteksi, menyelesaikan, dan memvalidasi karakteristik keamanan aplikasi Anda. Uji penetrasi yang teratur dan terjadwal harus dilakukan sebagai bagian dari siklus hidup pengembangan perangkat lunak (SDLC). Temuan dari uji penetrasi harus diatasi sebelum perangkat lunak dirilis. Anda harus menganalisis temuan dari uji penetrasi untuk mengidentifikasi apakah ada masalah yang dapat ditemukan menggunakan otomatisasi. Memiliki uji penetrasi yang teratur dan dapat diulangi serta menyertakan mekanisme umpan balik yang aktif membantu menginformasikan panduan kepada builder dan meningkatkan kualitas perangkat lunak.

Antipola umum:

- Hanya melakukan uji penetrasi untuk masalah keamanan yang diketahui atau umum.
- Melakukan uji penetrasi aplikasi tanpa pustaka dan alat pihak ketiga yang dependen.
- Hanya melakukan uji penetrasi untuk masalah keamanan paket, dan tidak mengevaluasi logika bisnis yang diimplementasikan.

Manfaat menjalankan praktik terbaik ini:

- Peningkatan kredibilitas karakteristik keamanan dari perangkat lunak sebelum rilis.
- Peluang untuk mengidentifikasi pola aplikasi yang dipilih, yang menghasilkan kualitas perangkat lunak yang lebih baik.
- Loop umpan balik yang mengidentifikasi lebih dini di siklus pengembangan di mana otomatisasi atau pelatihan tambahan dapat meningkatkan karakteristik keamanan perangkat lunak.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Uji penetrasi adalah langkah pengujian keamanan terstruktur untuk menjalankan skenario pelanggaran keamanan terencana guna mendeteksi, menyelesaikan, dan memvalidasi kontrol keamanan. Uji penetrasi dimulai dengan pengintaian, yang mana data dikumpulkan berdasarkan desain aplikasi dan dependensinya saat ini. Daftar kurasi skenario pengujian khusus keamanan dibuat dan dijalankan. Tujuan utama pengujian ini adalah mengungkap masalah keamanan di aplikasi Anda, yang dapat dieksploitasi untuk mendapatkan akses yang tidak direncanakan ke lingkungan Anda, atau akses yang tidak diotorisasi ke data Anda. Anda harus melakukan uji penetrasi saat meluncurkan fitur baru atau setiap kali aplikasi Anda menjalani perubahan besar pada implementasi teknis atau fungsi.

Anda harus mengidentifikasi tahap yang paling sesuai dalam siklus hidup pengembangan untuk melakukan uji penetrasi. Pengujian ini harus dilakukan pada waktu yang hampir mendekati status rilis fungsionalitas sistem yang direncanakan, tetapi ada waktu yang cukup untuk menyelesaikan masalah yang ada.

Langkah implementasi

- Miliki proses terstruktur mengenai cara uji penetrasi dicakup. Merancang proses berdasarkan model ancaman adalah cara yang baik dalam mempertahankan konteks.
- Identifikasi tempat yang sesuai dalam siklus pengembangan untuk melakukan uji penetrasi. Hal ini harus dilakukan saat ada perubahan minim yang diharapkan pada aplikasi, tetapi ada waktu yang cukup untuk melakukan penyelesaian masalah.
- Latih builder Anda untuk mengetahui apa saja yang diharapkan dari temuan uji penetrasi dan cara mendapatkan informasi dalam penyelesaian.
- Gunakan alat untuk mempercepat alat uji penetrasi dengan mengotomatiskan pengujian yang umum atau dapat diulang.
- Analisis temuan uji penetrasi untuk mengidentifikasi masalah keamanan sistemik, dan gunakan data ini untuk menginformasikan pengujian tambahan yang diotomatisasi dan pendidikan builder yang sedang berlangsung.

Sumber daya

Praktik Terbaik Terkait:

- SEC11-BP01 Pelatihan untuk keamanan aplikasi
- SEC11-BP02 Otomatiskan pengujian sepanjang siklus hidup pengembangan dan rilis

Dokumen terkait:

- Uji Penetrasi AWS menyediakan panduan mendetail untuk uji penetrasi pada AWS
- Percepat deployment di AWS dengan tata kelola yang efektif
- Partner Kompetensi Keamanan AWS
- Modernisasi arsitektur uji penetrasi Anda di AWS Fargate
- Simulator injeksi Kesalahan AWS

Contoh terkait:

- Otomatiskan pengujian API dengan AWS CodePipeline (GitHub)
- Pembantu keamanan yang diotomatiskan (GitHub)

SEC11-BP04 Peninjauan kode manual

Lakukan peninjauan kode manual atas perangkat lunak yang Anda hasilkan. Proses ini membantu memverifikasi bahwa orang yang menulis kode bukan satu-satunya orang yang memeriksa kualitas kode.

Hasil yang diinginkan: Menyertakan langkah peninjauan kode manual selama pengembangan meningkatkan kualitas perangkat lunak yang ditulis, membantu mengasah kemampuan anggota tim yang kurang berpengalaman, dan memberikan peluang untuk mengidentifikasi titik yang cocok untuk otomatisasi. Peninjauan kode manual dapat didukung oleh pengujian dan alat otomatis.

Antipola umum:

- Tidak melakukan peninjauan kode sebelum deployment.
- Penulis dan peninjau kode adalah orang yang sama.
- Tidak menggunakan otomatisasi untuk membantu atau mengatur peninjauan kode.
- Tidak melatih builder agar memahami keamanan aplikasi sebelum mereka meninjau kode.

Manfaat menjalankan praktik terbaik ini:

- Peningkatan kualitas kode.
- Peningkatan konsistensi pengembangan kode sepanjang penggunaan ulang pendekatan umum.
- Penurunan jumlah masalah yang ditemukan selama uji penetrasi dan tahap-tahap terakhir.
- Peningkatan transfer ilmu di dalam tim.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Langkah peninjauan harus diimplementasikan sebagai bagian dari keseluruhan alur manajemen kode. Spesifikasinya bergantung pada pendekatan yang digunakan untuk pencabangan, permintaan penarikan, dan penggabungan. Anda mungkin menggunakan AWS CodeCommit atau solusi pihak ketiga seperti GitHub, GitLab, atau Bitbucket. Apa pun metode yang Anda gunakan, penting untuk memastikan bahwa proses Anda memerlukan peninjauan kode sebelum di-deploy di lingkungan produksi. Menggunakan alat seperti Amazon CodeGuru Reviewer dapat mempermudah pengaturan proses peninjauan kode.

Langkah implementasi

- Implementasikan langkah peninjauan manual sebagai bagian dari alur manajemen kode Anda dan lakukan peninjauan ini sebelum melanjutkan.
- Pertimbangkan Amazon CodeGuru Reviewer untuk mengelola dan membantu dalam peninjauan kode.
- Implementasikan alur persetujuan yang mengharuskan peninjauan kode selesai sebelum kode dapat lanjut ke tahap berikutnya.
- Pastikan ada proses untuk mengidentifikasi masalah yang ditemukan selama peninjauan kode manual yang dapat dideteksi secara otomatis.
- Integrasikan langkah peninjauan kode manual menggunakan cara yang selaras dengan praktik pengembangan kode Anda.

Sumber daya

Praktik Terbaik Terkait:

- SEC11-BP02 Otomatiskan pengujian sepanjang siklus hidup pengembangan dan rilis

Dokumen terkait:

- Mengerjakan permintaan penarikan di repositori AWS CodeCommit
- Mengerjakan templat aturan persetujuan di AWS CodeCommit
- Tentang permintaan penarikan di GitHub
- Otomatiskan peninjauan kode dengan Amazon CodeGuru Reviewer
- Mengotomatisasi deteksi kerentanan keamanan dan bug di pipeline CI/CD menggunakan CLI Amazon CodeGuru Reviewer

Video terkait:

- Peningkatan berkelanjutan kualitas kode dengan Amazon CodeGuru

Contoh terkait:

- Lokakarya Keamanan untuk Developer

SEC11-BP05 Pusatkan layanan untuk paket dan dependensi

Berikan layanan terpusat agar tim builder dapat memperoleh paket perangkat lunak dan dependensi lainnya. Hal ini memungkinkan validasi paket sebelum paket disertakan dalam perangkat lunak yang Anda tulis, dan memberikan sumber data untuk analisis perangkat lunak yang digunakan dalam organisasi Anda.

Hasil yang diinginkan: Perangkat lunak yang terdiri dari sekumpulan paket perangkat lunak lainnya selain kode yang ditulis. Ini mempermudah untuk menggunakan implementasi fungsionalitas yang berulang kali digunakan, seperti pengurai JSON atau pustaka enkripsi. Memusatkan secara logis sumber untuk paket dan dependensi ini memberikan mekanisme bagi tim keamanan untuk memvalidasi karakteristik paket sebelum paket digunakan. Pendekatan ini juga menurunkan risiko masalah tidak terduga yang disebabkan oleh perubahan dalam paket yang ada, atau oleh tim builder, termasuk paket arbitrer langsung dari internet. Gunakan pendekatan ini sehubungan dengan alur pengujian manual dan otomatis untuk meningkatkan kredibilitas kualitas perangkat lunak yang dikembangkan.

Antipola umum:

- Menarik paket dari repositori arbitrer di internet.
- Tidak menguji paket baru sebelum menyediakannya kepada builder.

Manfaat menjalankan praktik terbaik ini:

- Pemahaman lebih baik mengenai paket apa yang digunakan di perangkat lunak yang dibangun.
- Dapat memberi tahu tim beban kerja saat paket perlu diperbarui berdasarkan pemahaman siapa yang menggunakan apa.
- Menurunkan risiko penyertaan paket bermasalah di perangkat lunak Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Sediakan layanan terpusat untuk paket dan dependensi dengan cara yang mudah digunakan bagi builder. Layanan terpusat dapat dipusatkan secara logis daripada diimplementasikan sebagai sistem monolitik. Pendekatan ini memungkinkan Anda menyediakan layanan dengan cara yang memenuhi kebutuhan builder Anda. Anda harus mengimplementasikan cara yang efisien untuk menambahkan paket ke repositori saat pembaruan terjadi atau persyaratan baru muncul. Layanan AWS seperti AWS CodeArtifact atau solusi partner AWS serupa menyediakan cara untuk mengirim kapabilitas ini.

Langkah Implementasi:

- Implementasikan layanan repositori terpusat secara logis yang tersedia di semua lingkungan tempat perangkat lunak dikembangkan.
- Sertakan akses ke repositori sebagai bagian dari proses vending Akun AWS.
- Buat otomatisasi untuk menguji paket sebelum paket dipublikasikan ke repositori.
- Pertahankan metrik paket yang paling sering digunakan, bahasa, dan tim dengan jumlah perubahan tertinggi.
- Sediakan mekanisme otomatis untuk tim builder guna meminta paket baru dan memberikan umpan balik.
- Pindai paket secara rutin di repositori Anda untuk mengidentifikasi potensi dampak masalah yang baru ditemukan.

Sumber daya

Praktik Terbaik Terkait:

- SEC11-BP02 Otomatiskan pengujian sepanjang siklus hidup pengembangan dan rilis

Dokumen terkait:

- Percepat deployment di AWS dengan tata kelola yang efektif
- Perketat keamanan paket Anda dengan toolkit CodeArtifact Package Origin Control
- Mendeteksi masalah keamanan dalam pencatatan log dengan Amazon CodeGuru Reviewer
- Level rantai Pasokan untuk Artefak Perangkat Lunak (SLSA)

Video terkait:

- Keamanan proaktif: Pertimbangan dan pendekatan
- Filosofi Keamanan AWS (re:Invent 2017)
- Saat keamanan, keselamatan, dan urgensi menjadi penting: Menangani Log4Shell

Contoh terkait:

- Pipeline Publikasi Paket Beberapa Wilayah (GitHub)
- Memublikasikan Modul Node.js di AWS CodeArtifact menggunakan AWS CodePipeline (GitHub)
- Sampel Pipeline AWS CDK Java CodeArtifact (GitHub)
- Distribusikan paket NuGet .NET pribadi dengan AWS CodeArtifact (GitHub)

SEC11-BP06 Lakukan deployment perangkat lunak secara terprogram

Lakukan deployment perangkat lunak secara terprogram jika memungkinkan. Pendekatan ini mengurangi kemungkinan terjadinya kegagalan deployment atau masalah tak terduga karena kesalahan manusia.

Hasil yang diinginkan: Meminimalkan campur tangan manusia dari data adalah prinsip utama pengembangan yang aman di AWS Cloud. Prinsip ini termasuk cara Anda melakukan deployment pada perangkat lunak.

Dengan tidak bergantung pada orang untuk men-deploy perangkat lunak, Anda akan mendapatkan manfaat peningkatan kredibilitas bahwa apa yang Anda uji adalah apa yang di-deploy, dan deployment dilakukan secara konsisten setiap kali dijalankan. Perangkat lunak tidak perlu diubah agar berfungsi di lingkungan yang berbeda. Menggunakan prinsip pengembangan aplikasi dua belas faktor, terutama eksternalisasi konfigurasi, memungkinkan Anda men-deploy kode yang sama ke beberapa lingkungan tanpa memerlukan perubahan. Menandatangani paket perangkat lunak

secara kriptografis adalah cara yang baik untuk memastikan bahwa tidak ada yang berubah di antara lingkungan. Keseluruhan hasil dari pendekatan ini adalah penurunan risiko proses perubahan dan peningkatan konsistensi rilis perangkat lunak.

Antipola umum:

- Men-deploy perangkat lunak secara manual ke tahap produksi.
- Melakukan perubahan secara manual ke perangkat lunak agar dapat menyesuaikan dengan lingkungan yang berbeda.

Manfaat menjalankan praktik terbaik ini:

- Peningkatan kredibilitas dalam proses rilis perangkat lunak.
- Penurunan risiko kegagalan perubahan yang berdampak pada fungsionalitas bisnis.
- Peningkatan jadwal rilis karena risiko terhadap perubahan lebih rendah.
- Kapabilitas pengembalian (rollback) otomatis untuk peristiwa tidak terduga selama deployment.
- Kemampuan untuk membuktikan secara kriptografis bahwa perangkat lunak yang diuji adalah perangkat lunak yang di-deploy.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Buat struktur Akun AWS Anda untuk menghapus akses manusia yang persisten dari lingkungan dan gunakan alat CI/CD untuk melakukan deployment. Rancang aplikasi Anda sehingga data konfigurasi khusus lingkungan diperoleh dari sumber eksternal, seperti AWS Systems Manager Parameter Store. Tanda tangani paket setelah paket diuji, dan validasikan tanda tangan ini selama deployment. Konfigurasikan pipeline CI/CD Anda untuk mendorong kode aplikasi dan menggunakan canary untuk mengonfirmasi deployment yang berhasil. Gunakan alat seperti AWS CloudFormation atau AWS CDK untuk menentukan infrastruktur Anda, lalu gunakan AWS CodeBuild dan AWS CodePipeline untuk melakukan operasi CI/CD.

Langkah implementasi

- Bangun pipeline CI/CD yang ditetapkan dengan baik untuk menyederhanakan proses deployment.
- Menggunakan AWS CodeBuild dan AWS Code Pipeline untuk menyediakan kemampuan CI/CD mempermudah untuk mengintegrasikan pengujian keamanan ke jalur Anda.

- Ikuti panduan pemisahan lingkungan di laporan resmi Mengatur Lingkungan AWS Anda Menggunakan Beberapa Akun.
- Pastikan tidak ada akses manusia yang persisten ke lingkungan tempat beban kerja produksi berjalan.
- Rancang aplikasi Anda untuk mendukung eksternalisasi data konfigurasi.
- Pertimbangkan untuk men-deploy menggunakan model deployment blue/green.
- Implementasikan canary untuk memvalidasi deployment perangkat lunak yang berhasil.
- Gunakan alat kriptografis seperti AWS Signer atau AWS Key Management Service (AWS KMS) untuk menandatangani dan memverifikasi paket perangkat lunak yang Anda deploy.

Sumber daya

Praktik Terbaik Terkait:

- SEC11-BP02 Otomatiskan pengujian sepanjang siklus hidup pengembangan dan rilis

Dokumen terkait:

- Lokakarya CI/CD AWS
- Percepat deployment di AWS dengan tata kelola yang efektif
- Mengotomatiskan deployment aman tanpa campur tangan
- Penandatanganan kode menggunakan AWS Certificate Manager Private CA dan kunci asimetris AWS Key Management Service
- Penandatanganan Kode, Kontrol Integritas dan Kepercayaan untuk AWS Lambda

Video terkait:

- Tanpa campur tangan: Mengotomatiskan pipeline pengiriman berkelanjutan di Amazon

Contoh terkait:

- Deployment Blue/Green dengan AWS Fargate

SEC11-BP07 Nilai karakteristik keamanan pipeline secara teratur

Terapkan prinsip Pilar Keamanan Well-Architected pada pipeline Anda, dengan perhatian khusus pada pemisahan izin. Nilai karakteristik keamanan infrastruktur pipeline Anda secara teratur. Mengelola keamanan pipeline secara efektif akan memungkinkan Anda memberikan keamanan perangkat lunak yang lolos melalui pipeline.

Hasil yang diinginkan: Pipeline yang digunakan untuk membangun dan men-deploy perangkat lunak Anda harus mengikuti rekomendasi praktik yang sama seperti beban kerja lainnya di lingkungan Anda. Pengujian yang diimplementasikan di pipeline seharusnya tidak dapat diedit oleh builder yang menggunakannya. Pipeline seharusnya hanya memiliki izin yang diperlukan untuk deployment yang dilakukannya dan harus mengimplementasikan perlindungan untuk menghindari deployment ke lingkungan yang salah. Pipeline tidak boleh bergantung pada kredensial jangka panjang, dan harus dikonfigurasikan untuk memberikan status sehingga integritas lingkungan build dapat divalidasi.

Antipola umum:

- Pengujian keamanan yang dapat dilewati oleh builder.
- Izin yang terlalu luas untuk pipeline deployment.
- Pipeline tidak dikonfigurasikan untuk memvalidasi input.
- Tidak rutin meninjau izin yang terkait dengan infrastruktur CI/CD Anda.
- Penggunaan kredensial jangka panjang atau yang diberi hardcode.

Manfaat menjalankan praktik terbaik ini:

- Kredibilitas lebih tinggi pada integritas perangkat lunak yang dibangun dan di-deploy melalui pipeline.
- Kemampuan untuk menghentikan deployment saat ada aktivitas yang mencurigakan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Memulai dengan layanan CI/CD terkelola yang mendukung peran IAM menurunkan risiko kebocoran kredensial. Menerapkan prinsip Pilar Keamanan ke infrastruktur pipeline CI/CD Anda dapat membantu Anda menentukan titik mana yang dapat ditingkatkan keamanannya. Mengikuti Arsitektur Rujukan Pipeline Deployment AWS adalah titik awal yang baik untuk membangun lingkungan CI/CD Anda. Meninjau secara rutin implementasi pipeline dan menganalisis log untuk menemukan perilaku

tidak terduga dapat membantu Anda memahami pola penggunaan pipeline yang digunakan untuk men-deploy perangkat lunak.

Langkah implementasi

- Mulai dengan Arsitektur Rujukan Pipeline Deployment AWS.
- Pertimbangkan untuk menggunakan AWS IAM Access Analyzer agar dapat secara terprogram membuat kebijakan IAM hak akses paling rendah untuk pipeline.
- Integrasikan pipeline Anda dengan pemantauan dan peringatan sehingga Anda mendapatkan notifikasi aktivitas tidak terduga atau tidak normal, untuk layanan terkelola AWS, Amazon EventBridge memungkinkan Anda merutekan data ke target seperti AWS Lambda atau Amazon Simple Notification Service (Amazon SNS).

Sumber daya

Dokumen terkait:

- Arsitektur Rujukan Pipeline Deployment AWS
- Pemantauan AWS CodePipeline
- Praktik terbaik keamanan untuk AWS CodePipeline

Contoh terkait:

- Dasbor pemantauan DevOps (GitHub)

SEC11-BP08 Buat program yang menanamkan kepemilikan keamanan dalam tim beban kerja

Buat program atau mekanisme yang memberdayakan tim builder untuk membuat keputusan keamanan tanpa perangkat lunak yang mereka buat. Tim keamanan Anda masih harus memvalidasi keputusan ini selama peninjauan, tetapi menanamkan kepemilikan keamanan dalam tim builder memungkinkan beban kerja dibangun dengan lebih cepat dan lebih aman. Mekanisme ini juga mendukung budaya kepemilikan yang secara positif memengaruhi operasi sistem yang Anda buat.

Hasil yang diinginkan: Untuk menanamkan kepemilikan keamanan dan pengambilan keputusan dalam tim builder, Anda dapat melatih builder mengenai cara berpikir tentang keamanan atau meningkatkan pelatihan mereka bersama orang keamanan yang diikutsertakan atau dikaitkan dengan tim builder. Pendekatan mana pun bisa dilakukan dan memungkinkan tim mengambil

keputusan keamanan yang lebih berkualitas pada awal siklus pengembangan. Model kepemilikan ini didasarkan pada pelatihan untuk keamanan aplikasi. Memulai dengan model ancaman untuk beban kerja tertentu akan membantu fokus pada pemikiran desain dalam konteks yang sesuai. Manfaat lain dalam memiliki komunitas builder yang fokus pada keamanan, atau kelompok rekayasawan keamanan yang bekerja sama dengan tim builder, adalah pemahaman yang lebih mendalam mengenai cara perangkat lunak ditulis. Pemahaman ini membantu Anda menentukan area peningkatan selanjutnya dalam kemampuan otomatisasi Anda.

Antipola umum:

- Menyerahkan semua keputusan desain keamanan kepada tim keamanan.
- Tidak menangani persyaratan keamanan cukup dini dalam proses pengembangan.
- Tidak memperoleh umpan balik dari builder dan orang keamanan dalam pengoperasian program.

Manfaat menjalankan praktik terbaik ini:

- Waktu penyelesaian peninjauan keamanan lebih cepat.
- Penurunan masalah keamanan yang hanya terdeteksi pada tahap peninjauan keamanan.
- Peningkatan keseluruhan kualitas perangkat lunak yang ditulis.
- Peluang untuk mengidentifikasi dan memahami masalah sistemik atau area dari peningkatan bernilai tinggi.
- Penurunan jumlah pengerjaan ulang yang diperlukan akibat temuan dari peninjauan keamanan.
- Peningkatan persepsi fungsi keamanan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Mulai dengan panduan di SEC11-BP01 Pelatihan untuk keamanan aplikasi. Lalu, identifikasi model operasional untuk program yang menurut Anda paling sesuai dengan organisasi Anda. Dua pola utamanya adalah melatih builder atau menyertakan orang keamanan ke dalam tim builder. Setelah Anda memutuskan pendekatan awal, Anda harus melakukan uji coba dengan satu grup atau grup kecil tim beban kerja untuk membuktikan model mana yang sesuai dengan organisasi Anda. Dukungan kepemimpinan dari bagian builder dan keamanan organisasi membantu pengiriman dan kesuksesan program. Saat Anda membangun program, penting untuk memilih metrik yang dapat digunakan untuk menunjukkan nilai program. Belajar dari cara AWS menangani masalah ini

adalah pengalaman pembelajaran yang baik. Praktik terbaik ini sangat berfokus pada budaya dan perubahan organisasi. Alat yang Anda gunakan harus mendukung kolaborasi antara komunitas keamanan dan builder.

Langkah implementasi

- Mulai dengan melatih builder Anda untuk memahami keamanan aplikasi.
- Buat komunitas dan program orientasi untuk mengedukasi builder.
- Pilih nama untuk program. Pelindung, Jawara, atau Pendukung adalah nama yang sering digunakan.
- Identifikasi model yang akan digunakan: latih builder, sertakan rekayasawan keamanan, atau miliki peran keamanan afinitas.
- Identifikasi sponsor proyek dari grup keamanan, builder, dan grup lain yang berpotensi.
- Lacak metrik untuk jumlah orang yang terlibat dalam program, waktu yang dihabiskan untuk peninjauan, dan umpan balik dari orang keamanan dan builder. Gunakan metrik ini untuk membuat peningkatan.

Sumber daya

Praktik Terbaik Terkait:

- SEC11-BP01 Pelatihan untuk keamanan aplikasi
- SEC11-BP02 Otomatiskan pengujian sepanjang siklus hidup pengembangan dan rilis

Dokumen terkait:

- Cara memanfaatkan pemodelan ancaman
- Cara berpikir tentang tata kelola keamanan cloud

Video terkait:

- Keamanan proaktif: Pertimbangan dan pendekatan

Keandalan

Pilar keandalan berkenaan dengan kemampuan beban kerja untuk menjalankan fungsinya dengan benar dan konsisten sesuai ekspektasi. Anda dapat menemukan panduan preskriptif tentang implementasi di Laporan Resmi Pilar Keandalan.

Area praktik terbaik

- Fondasi
- Arsitektur beban kerja
- Manajemen perubahan
- Manajemen kegagalan

Fondasi

Pertanyaan

- REL 1. Bagaimana cara mengelola Kuota Layanan dan batasan?
- REL 2. Bagaimana cara merencanakan topologi jaringan Anda?

REL 1. Bagaimana cara mengelola Kuota Layanan dan batasan?

Untuk arsitektur beban kerja berbasis cloud, ada Kuota Layanan (yang juga disebut sebagai batas layanan). Kuota ini ada untuk mencegah tanpa sengaja memberikan sumber daya lebih daripada yang Anda butuhkan dan untuk membatasi tingkat permintaan di operasi API sehingga melindungi layanan dari penyalahgunaan. Ada juga batas sumber daya, misalnya, laju Anda dapat mendorong bit di kabel serat optik, atau jumlah penyimpanan di disk secara fisik.

Praktik terbaik

- REL01-BP01 Kesadaran tentang kuota dan kendala layanan
- REL01-BP02 Mengelola kuota layanan di seluruh akun dan wilayah
- REL01-BP03 Mengakomodasi kuota layanan tetap dan kendala melalui arsitektur
- REL01-BP04 Memantau dan mengelola kuota
- REL01-BP05 Mengotomatiskan manajemen kuota
- REL01-BP06 Memastikan adanya selisih yang memadai antara kuota saat ini dan penggunaan maksimum untuk mengakomodasi failover

REL01-BP01 Kesadaran tentang kuota dan kendala layanan

Perhatikan kuota default Anda dan kelola permintaan penambahan kuota untuk arsitektur beban kerja Anda. Ketahui kendala sumber daya cloud mana, seperti disk atau jaringan, yang berpotensi memberi dampak.

Hasil yang diinginkan: Pelanggan dapat mencegah penurunan kualitas atau gangguan layanan dalam Akun AWS mereka dengan mengimplementasikan pedoman yang tepat untuk memantau metrik utama, peninjauan infrastruktur, dan langkah-langkah perbaikan otomatisasi untuk memverifikasi tidak tercapainya kuota dan kendala layanan yang dapat menyebabkan penurunan kualitas dan gangguan layanan.

Antipola umum:

- Melakukan deployment beban kerja tanpa memahami kuota keras dan lunak serta batasannya untuk layanan yang digunakan.
- Melakukan deployment beban kerja pengganti tanpa menganalisis dan mengonfigurasi ulang kuota yang diperlukan atau menghubungi tim Dukungan sebelumnya.
- Berasumsi bahwa layanan cloud tidak memiliki batasan dan layanan dapat digunakan tanpa mempertimbangkan angka permintaan, batas, hitungan, kuantitas.
- Berasumsi bahwa kuota akan ditingkatkan secara otomatis.
- Tidak mengetahui proses dan lini waktu permintaan kuota.
- Berasumsi bahwa kuota layanan cloud default selalu sama untuk setiap layanan di semua wilayah.
- Berasumsi bahwa kendala layanan dapat ditembus dan sistem akan diskalakan secara otomatis dan meningkatkan batas di luar kendala sumber daya.
- Tidak menguji aplikasi saat lalu lintas memuncak untuk membebani pemanfaatan sumber dayanya.
- Melakukan pengadaan sumber daya tanpa analisis ukuran sumber daya yang diperlukan.
- Berlebihan dalam pengadaan kapasitas dengan memilih jenis sumber daya yang jauh melampaui kebutuhan riil atau perkiraan lalu lintas puncak.
- Tidak menilai persyaratan kapasitas untuk tingkat lalu lintas baru sebelum peristiwa pelanggan baru atau deployment teknologi baru.

Manfaat menjalankan praktik terbaik ini: Pemantauan dan manajemen otomatis kuota layanan serta kendala sumber daya dapat mengurangi kegagalan secara proaktif. Perubahan pada pola lalu lintas untuk layanan pelanggan dapat menyebabkan gangguan atau penurunan kualitas jika praktik terbaik tidak diikuti. Dengan memantau dan mengelola nilai-nilai ini di semua wilayah dan semua akun,

aplikasi dapat memiliki ketahanan yang lebih baik selama peristiwa yang tidak direncanakan atau tidak diinginkan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Service Quotas adalah sebuah layanan AWS yang membantu Anda mengelola kuota Anda untuk lebih dari 250 layanan AWS dari satu lokasi. Di samping mencari nilai kuota, Anda juga dapat meminta dan melacak peningkatan kuota dari konsol Service Quotas atau menggunakan SDK AWS. AWS Trusted Advisor menawarkan pemeriksaan kuota layanan yang menampilkan penggunaan dan kuota Anda untuk beberapa aspek dari beberapa layanan. Kuota layanan default per layanan juga ada di dalam dokumentasi AWS berdasarkan layanan masing-masing (misalnya, lihat Kuota Amazon VPC).

Beberapa batas layanan, seperti batas tingkat pada API yang diberikan throttling diatur di dalam Amazon API Gateway itu sendiri dengan mengonfigurasi rencana penggunaan. Batas yang diatur sebagai konfigurasi di layanannya masing-masing diantaranya adalah IOPS yang Disediakan, penyimpanan Amazon RDS yang dialokasikan, dan alokasi volume Amazon EBS. Amazon Elastic Compute Cloud memiliki dasbor batas layanannya sendiri yang dapat membantu Anda mengelola instans Anda, Amazon Elastic Block Store, dan batas alamat IP Elastis. Jika Anda memiliki kasus penggunaan di mana kuota layanan memengaruhi kinerja aplikasi Anda dan tidak dapat disesuaikan dengan kebutuhan Anda, hubungi Dukungan untuk mengetahui apakah terdapat langkah mitigasi.

Kuota layanan bisa menurut Wilayah atau bersifat global. Layanan AWS yang mencapai kuotanya tidak akan berfungsi sebagaimana mestinya dalam penggunaan normal dan dapat menyebabkan gangguan atau penurunan kualitas layanan. Misalnya, suatu kuota layanan membatasi jumlah DL Amazon EC2 yang dapat digunakan di sebuah Wilayah dan batasan tersebut dapat dicapai selama peristiwa penskalaan lalu lintas menggunakan grup Auto Scaling (ASG).

Kuota layanan untuk setiap akun harus dinilai secara rutin dalam hal penggunaan untuk menentukan batas layanan yang tepat untuk akun tersebut. Kuota layanan ini dibuat sebagai pagar pembatas operasional, agar Anda tidak melakukan pengadaan sumber daya lebih dari yang dibutuhkan tanpa disadari. Kuota layanan juga berfungsi untuk membatasi angka permintaan pada operasi API guna melindungi layanan dari penyalahgunaan.

Kendala layanan berbeda dari kuota layanan. Kendala layanan mewakili batasan sumber daya tertentu yang ditentukan oleh jenis sumber daya tersebut. Kendala tersebut dapat berupa kapasitas penyimpanan (misalnya, gp2 memiliki batas ukuran 1 GB - 16 TB) atau disk throughput (10.0000 iops). Sangat penting untuk merancang dan terus menilai kendala jenis sumber daya untuk

penggunaan yang mungkin mencapai batasnya. Jika suatu kendala tercapai di luar perkiraan, aplikasi dan layanan akun dapat terganggu atau mengalami penurunan kualitas.

Jika terdapat kasus penggunaan di mana kuota layanan memengaruhi kinerja aplikasi dan tidak dapat disesuaikan dengan kebutuhan, hubungi Dukungan untuk mengetahui apakah terdapat langkah mitigasi. Untuk detail selengkapnya tentang penyesuaian kuota tetap, lihat REL01-BP03 Mengakomodasi kuota layanan tetap dan kendala melalui arsitektur.

Terdapat sejumlah layanan dan alat AWS untuk membantu memantau dan mengelola Service Quotas. Layanan dan alat harus dimanfaatkan untuk menyediakan pemeriksaan level kuota secara otomatis atau manual.

- AWS Trusted Advisor menawarkan pemeriksaan kuota layanan yang menampilkan penggunaan dan kuota Anda untuk beberapa aspek dari beberapa layanan. Alat ini dapat membantu mengidentifikasi layanan yang mendekati kuota.
- Konsol Manajemen AWS menyediakan metode untuk menampilkan nilai kuota layanan, mengelola, meminta kuota baru, memantau status permintaan kuota, dan menampilkan riwayat kuota.
- AWS CLI dan CDK menawarkan metode terprogram untuk mengelola dan memantau level serta penggunaan kuota layanan secara otomatis.

Langkah implementasi

Untuk Service Quotas:

- Pelajari AWS Service Quotas.
- Untuk mengetahui kuota layanan Anda saat ini, tentukan layanan (seperti IAM Access Analyzer) yang digunakan. Terdapat sekitar 250 layanan AWS yang dikontrol oleh kuota layanan. Lalu, tentukan nama kuota layanan tertentu yang dapat digunakan di dalam setiap akun dan wilayah. Terdapat sekitar 3000 nama kuota layanan per wilayah.
- Perkuat analisis kuota ini dengan AWS Config untuk menemukan semua sumber daya AWS yang digunakan di dalam Akun AWS Anda.
- Gunakan data AWS CloudFormation untuk menentukan sumber daya AWS Anda yang digunakan. Lihat sumber daya yang dibuat baik di Konsol Manajemen AWS maupun dengan perintah `list-stack-resources` AWS CLI. Anda juga dapat melihat sumber daya yang dikonfigurasi untuk diterapkan di templat itu sendiri.
- Tentukan semua layanan yang diperlukan oleh beban kerja Anda dengan melihat kode deployment.

- Tentukan kuota layanan yang berlaku. Gunakan informasi yang dapat diakses secara terprogram dari Trusted Advisor dan Service Quotas.
- Bangun metode pemantauan otomatis (lihat REL01-BP02 Mengelola kuota layanan di seluruh akun dan wilayah dan REL01-BP04 Memantau dan mengelola kuota) untuk memberi peringatan dan pemberitahuan jika kuota layanan mendekati atau sudah mencapai batas.
- Bangun metode otomatis dan terprogram untuk memeriksa apakah kuota layanan telah diubah di satu wilayah tetapi tidak diubah di wilayah lain di dalam akun yang sama (lihat REL01-BP02 Mengelola kuota layanan di seluruh akun dan wilayah dan REL01-BP04 Memantau dan mengelola kuota).
- Otomatiskan pemindaian log dan metrik aplikasi untuk menentukan apakah terdapat kesalahan kuota atau kendala layanan. Jika terdapat kesalahan, kirimkan peringatan ke sistem pemantauan.
- Bangun prosedur rekayasa untuk menghitung perubahan yang diperlukan dalam kuota (lihat REL01-BP05 Mengotomatiskan manajemen kuota) setelah diidentifikasi bahwa diperlukan kuota yang lebih besar untuk layanan tertentu.
- Buat alur kerja pengadaan dan persetujuan untuk meminta perubahan dalam kuota layanan. Sertakan di dalamnya alur kerja pengecualian untuk mengantisipasi jika permintaan ditolak atau disetujui sebagian.
- Buat metode rekayasa untuk meninjau kuota layanan sebelum pengadaan dan menggunakan layanan AWS baru sebelum digulirkan ke produksi atau lingkungan yang berisi data (misalnya akun pengujian beban).

Untuk kendala layanan:

- Bangun metode pemantauan dan metrik untuk memberi peringatan jika pembacaan sumber daya mendekati kendala sumber dayanya. Manfaatkan CloudWatch apabila diperlukan untuk pemantauan metrik atau log.
- Bangun ambang batas peringatan untuk setiap sumber daya yang memiliki kendala yang dapat berpengaruh pada aplikasi atau sistem.
- Ciptakan prosedur manajemen alur kerja dan infrastruktur untuk mengubah jenis sumber daya jika pemanfaatan mendekati kendala. Alur kerja ini harus mencakup pengujian beban sebagai praktik terbaik untuk memverifikasi bahwa jenis sumber daya baru tersebut sudah tepat dengan kendala baru.
- Migrasikan sumber daya yang diidentifikasi ke jenis sumber daya baru yang disarankan, menggunakan prosedur dan proses yang ada.

Sumber daya

Praktik Terbaik Terkait:

- REL01-BP02 Mengelola kuota layanan di seluruh akun dan wilayah
- REL01-BP03 Mengakomodasi kuota layanan tetap dan kendala melalui arsitektur
- REL01-BP04 Memantau dan mengelola kuota
- REL01-BP05 Mengotomatiskan manajemen kuota
- REL01-BP06 Memastikan adanya selisih yang memadai antara kuota saat ini dan penggunaan maksimum untuk mengakomodasi failover
- REL03-BP01 Memilih cara untuk menyegmentasi beban kerja
- REL10-BP01 Melakukan deployment beban kerja ke beberapa lokasi
- REL11-BP01 Memantau semua komponen beban kerja untuk mendeteksi kegagalan
- REL11-BP03 Mengotomatisasi pemulihan di semua lapisan
- REL12-BP05 Menguji ketahanan menggunakan chaos engineering

Dokumen terkait:

- Pilar Keandalan Kerangka Kerja AWS Well-Architected: Ketersediaan
- AWS Service Quotas (sebelumnya disebut batas layanan)
- AWS Trusted Advisor Pemeriksaan Praktik Terbaik (lihat bagian Batas Layanan)
- Pemantau batas AWS di AWS Answers
- Batas Layanan Amazon EC2
- Apa itu Service Quotas?
- Cara Meminta Peningkatan Kuota
- Endpoint dan kuota layanan
- Panduan Pengguna Service Quotas
- Pemantau Kuota untuk AWS
- Batas Isolasi Kesalahan AWS
- Ketersediaan dengan redundansi
- AWS untuk Data
- Apa itu Integrasi Berkelanjutan?

- Apa itu Pengiriman Berkelanjutan?
- Partner APN: partner yang dapat membantu manajemen konfigurasi
- Mengelola siklus hidup akun di dalam lingkungan SaaS akun per tenant di AWS
- Mengelola dan memanfatau throttling API di dalam beban kerja Anda
- Lihat rekomendasi AWS Trusted Advisor pada skala besar dengan AWS Organizations
- Mengotomatiskan Peningkatan Batas Layanan dan Dukungan Korporat dengan AWS Control Tower

Video terkait:

- AWS Live re:Inforce 2019 - Service Quotas
- Melihat dan Mengelola Kuota untuk Layanan AWS Menggunakan Service Quotas
- Demo Kuota AWS IAM

Alat terkait:

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP02 Mengelola kuota layanan di seluruh akun dan wilayah

Jika Anda menggunakan beberapa akun atau Wilayah, minta kuota yang sesuai di semua lingkungan tempat beban kerja produksi Anda dijalankan.

Hasil yang diinginkan: Layanan dan aplikasi tidak boleh dipengaruhi oleh habisnya kuota layanan untuk konfigurasi yang meliputi akun atau Wilayah atau yang memiliki desain ketahanan menggunakan failover zona, Wilayah, atau akun.

Antipola umum:

- Membiarkan penggunaan sumber daya di satu Wilayah terisolasi bertambah, tanpa mekanisme untuk mempertahankan kapasitas di wilayah lainnya.
- Mengatur semua kuota di Wilayah isolasi secara manual dan independen.
- Tidak mempertimbangkan efek arsitektur ketahanan (seperti aktif atau pasif) dalam kebutuhan kuota masa depan selama penurunan kualitas di dalam Wilayah non-primer.
- Tidak mengevaluasi kuota secara rutin dan membuat perubahan yang diperlukan di setiap Wilayah dan akun tempat beban kerja dijalankan.
- Tidak memanfaatkan templat pemintaan kuota untuk meminta penambahan di beberapa Wilayah dan akun.
- Tidak memperbarui kuota layanan disebabkan pemikiran yang tidak tepat bahwa peningkatan kuota memiliki dampak biaya seperti permintaan pemesanan komputasi.

Manfaat menjalankan praktik terbaik ini: Memverifikasi bahwa Anda dapat menangani beban saat ini di wilayah atau akun sekunder jika layanan wilayah tidak tersedia. Hal ini dapat membantu mengurangi jumlah kesalahan atau tingkat penurunan kualitas yang terjadi selama region loss.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Kuota layanan dilacak per akun. Setiap kuota disesuaikan dengan Wilayah AWS, kecuali ditetapkan sebaliknya. Selain lingkungan produksi, kelola juga kuota di semua lingkungan non-produksi yang berlaku agar pengujian dan pengembangan tidak terhambat. Untuk mempertahankan tingkat ketahanan yang tinggi diperlukan penilaian kuota layanan secara kontinu (baik otomatis maupun manual).

Dengan makin banyaknya beban kerja yang meliputi beberapa Wilayah dikarenakan implementasi desain yang menggunakan pendekatan Aktif/Aktif, Aktif/Pasif – Panas, Aktif/Pasif-Dingin, dan Aktif/Pasif-Pilot Light, sangat penting memahami semua level kuota Wilayah dan akun. Pola lalu lintas terdahulu tidak selalu menjadi indikator yang tepat bahwa kuota layanan diatur dengan benar.

Sama pentingnya, batas nama kuota layanan tidak selalu sama untuk setiap Wilayah. Di satu Wilayah, nilainya bisa jadi lima, sedangkan di wilayah lain nilainya bisa jadi sepuluh. Manajemen kuota-kuota ini harus meliputi semua layanan, akun, dan Wilayah yang sama untuk menyediakan ketahanan yang konsisten saat beban meningkat.

Sesuaikan semua perbedaan kuota layanan di semua Wilayah yang berbeda (Wilayah Aktif atau Wilayah Pasif) dan buat proses untuk menyesuaikan semua perbedaan tersebut secara kontinu. Rencana pengujian failover Wilayah pasif sangat jarang diskalakan ke kapasitas aktif puncak, sehingga kegiatan game day atau table top bisa gagal menemukan perbedaan kuota layanan antar-Wilayah dan juga mempertahankan batas-batas yang tepat.

Penyimpangan kuota layanan, kondisi di mana batas kuota layanan untuk kuota bernama tertentu diubah di salah satu Wilayah tetapi tidak di semua Wilayah, sangat penting untuk dilacak dan dinilai. Mengubah kuota di Wilayah yang memiliki lalu lintas atau yang berpotensi mendatangkan lalu lintas harus dipertimbangkan.

- Pilih Wilayah dan akun yang relevan berdasarkan persyaratan layanan, latensi, peraturan, dan pemulihan bencana (DR) Anda.
- Identifikasikan kuota layanan di semua akun, Wilayah, dan Zona Ketersediaan yang relevan. Batasannya mencakup akun dan Wilayah. Nilai-nilai ini harus dibandingkan untuk mengetahui perbedaannya.

Langkah implementasi

- Tinjau nilai Service Quotas yang mungkin telah melampaui level risiko penggunaan a. AWS Trusted Advisor menyediakan peringatan untuk pelanggaran 80% dan 90% ambang batas.
- Tinjau nilai untuk kuota layanan di Wilayah Pasif mana pun (dalam desain Aktif/Pasif). Verifikasi bahwa muatan akan berhasil berjalan di dalam Wilayah sekunder apabila terjadi kegagalan di dalam Wilayah primer.
- Otomatiskan penilaian jika penyelewengan kuota layanan apa pun telah terjadi antar-Wilayah di dalam akun yang sama dan lakukan tindakan yang semestinya untuk mengubah batas.
- Jika Unit Organisasi (OU) pelanggan disusun dengan cara yang didukung, templat kuota layanan harus diperbarui agar mencerminkan perubahan pada kuota apa pun yang harus diterapkan ke beberapa Wilayah dan akun.
  - Buat templat dan kaitkan Wilayah dengan perubahan kuota.
  - Tinjau semua templat kuota layanan yang ada untuk mengetahui jika ada perubahan yang diperlukan (Wilayah, batas, dan akun).

Sumber daya

Praktik Terbaik Terkait:

- REL01-BP01 Kesadaran tentang kuota dan kendala layanan
- REL01-BP03 Mengakomodasi kuota layanan tetap dan kendala melalui arsitektur
- REL01-BP04 Memantau dan mengelola kuota
- REL01-BP05 Mengotomatiskan manajemen kuota
- REL01-BP06 Memastikan adanya selisih yang memadai antara kuota saat ini dan penggunaan maksimum untuk mengakomodasi failover
- REL03-BP01 Memilih cara untuk menyegmentasi beban kerja
- REL10-BP01 Melakukan deployment beban kerja ke beberapa lokasi
- REL11-BP01 Memantau semua komponen beban kerja untuk mendeteksi kegagalan
- REL11-BP03 Mengotomatisasi pemulihan di semua lapisan
- REL12-BP05 Menguji ketahanan menggunakan chaos engineering

Dokumen terkait:

- Pilar Keandalan Kerangka Kerja AWS Well-Architected: Ketersediaan
- AWS Service Quotas (sebelumnya disebut batas layanan)
- AWS Trusted Advisor Pemeriksaan Praktik Terbaik (lihat bagian Batas Layanan)
- Pemantau batas AWS di AWS Answers
- Batas Layanan Amazon EC2
- Apa itu Service Quotas?
- Cara Meminta Peningkatan Kuota
- Endpoint dan kuota layanan
- Panduan Pengguna Service Quotas
- Pemantau Kuota untuk AWS
- Batas Isolasi Kesalahan AWS
- Ketersediaan dengan redundansi
- AWS untuk Data
- Apa itu Integrasi Berkelanjutan?

- Apa itu Pengiriman Berkelanjutan?
- Partner APN: partner yang dapat membantu manajemen konfigurasi
- Mengelola siklus hidup akun di dalam lingkungan SaaS akun per tenant di AWS
- Mengelola dan memanfatau throttling API di dalam beban kerja Anda
- Lihat rekomendasi AWS Trusted Advisor pada skala besar dengan AWS Organizations
- Mengotomatiskan Peningkatan Batas Layanan dan Dukungan Korporat dengan AWS Control Tower

Video terkait:

- AWS Live re:Inforce 2019 - Service Quotas
- Melihat dan Mengelola Kuota untuk Layanan AWS Menggunakan Service Quotas
- Demo Kuota AWS IAM

Layanan terkait:

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP03 Mengakomodasi kuota layanan tetap dan kendala melalui arsitektur

Waspadai kuota layanan, kendala layanan, dan batas sumber daya fisik yang tidak dapat diubah. Rancang arsitektur untuk aplikasi dan layanan untuk mencegah batasan ini memengaruhi keandalan.

Contohnya antara lain bandwidth jaringan, ukuran payload invokasi fungsi nirserver, tingkat burst throttle untuk gateway API, dan sambungan pengguna serentak ke basis data.

Hasil yang diinginkan: Performa aplikasi atau layanan sesuai yang diharapkan dalam kondisi normal dan lalu lintas tinggi. Aplikasi dan layanan telah dirancang untuk berfungsi dengan batasan untuk kuota layanan atau kendala tetap sumber daya tersebut.

Antipola umum:

- Memilih desain yang menggunakan sumber daya layanan, tidak menyadari bahwa terdapat kendala desain yang akan menyebabkan desain ini gagal begitu Anda menyesuaikan skala.
- Melakukan tolok ukur yang tidak realistis dan akan mencapai kuota tetap layanan selama pengujian. Contohnya, menjalankan pengujian pada batas burst tetapi dalam jangka waktu yang lama.
- Memilih desain yang tidak dapat diskalakan atau dimodifikasi jika kuota layanan tetap akan terlampaui. Contohnya, ukuran payload SQS sebesar 256 KB.
- Observabilitas belum dirancang dan diimplementasikan untuk memantau dan memberikan peringatan tentang ambang batas kuota layanan yang mungkin berisiko selama lalu lintas sedang tinggi.

Manfaat menjalankan praktik terbaik ini: Verifikasi bahwa aplikasi akan beroperasi di bawah semua tingkat beban layanan yang diproyeksikan tanpa gangguan atau penurunan kualitas.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Tidak seperti sumber daya atau kuota layanan lunak yang diganti dengan unit kapasitas lebih tinggi, kuota tetap layanan AWS tidak dapat diubah. Ini berarti semua jenis layanan AWS ini harus dievaluasi untuk mengetahui potensi batas kapasitas keras ketika digunakan dalam desain aplikasi.

Batas keras ditunjukkan di konsol Service Quotas. Jika kolom menunjukkan `DAPAT DISESUAIKAN = Tidak`, layanan memiliki batas keras. Batas keras juga ditunjukkan di beberapa halaman konfigurasi sumber daya. Contohnya, Lambda memiliki batas keras spesifik yang tidak dapat disesuaikan.

Sebagai contoh, ketika mendesain aplikasi python untuk beroperasi dalam fungsi Lambda, aplikasi harus dievaluasi untuk menentukan apakah ada peluang Lambda akan beroperasi lebih lama dari 15

menit. Jika kode mungkin akan dijalankan lebih dari batas kuota layanan ini, desain atau teknologi lain harus dipertimbangkan. Jika batas ini tercapai setelah deployment produksi, aplikasi akan mengalami penurunan kualitas dan gangguan sampai dapat diperbaiki. Tidak seperti kuota lunak, tidak ada metode untuk mengubah batas-batas ini meskipun dalam kondisi darurat Keparahan 1.

Setelah aplikasi telah di-deploy ke lingkungan pengujian, strategi harus digunakan untuk menemukan apakah batas keras dapat tercapai. Pengujian stres, pengujian beban, dan pengujian kekacauan harus menjadi bagian dari rencana pengujian pendahuluan.

Langkah implementasi

- Tinjau daftar lengkap layanan AWS yang dapat digunakan dalam fase desain aplikasi.
- Tinjau batas kuota lunak dan batas kuota keras untuk semua layanan ini. Tidak semua batas ditunjukkan di konsol Service Quotas. Beberapa layanan menjelaskan batas-batas ini di lokasi lain.
- Saat Anda mendesain aplikasi Anda, tinjau pendorong teknologi dan bisnis beban kerja Anda, seperti hasil bisnis, kasus penggunaan, sistem yang dependen, target ketersediaan, dan objek pemulihan bencana. Biarkan pendorong teknologi dan bisnis Anda memandu proses untuk mengidentifikasi sistem terdistribusi yang tepat untuk beban kerja Anda.
- Analisis beban layanan di berbagai Wilayah dan akun. Banyak batas keras yang berbasis wilayah untuk layanan. Tetapi, beberapa batas berbasis akun.
- Analisis arsitektur ketangguhan untuk penggunaan sumber daya selama kegagalan zona dan kegagalan Wilayah. Jika progres desain multi-Wilayah menggunakan pendekatan aktif/aktif, aktif/pasif - panas, aktif/pasif - dingin, dan aktif/pasif - pilot light, kasus-kasus kegagalan ini akan menyebabkan penggunaan yang lebih tinggi. Hal ini akan menimbulkan potensi kasus penggunaan untuk mencapai batas keras.

Sumber daya

Praktik Terbaik Terkait:

- REL01-BP01 Kesadaran tentang kuota dan kendala layanan
- REL01-BP02 Mengelola kuota layanan di seluruh akun dan wilayah
- REL01-BP04 Memantau dan mengelola kuota
- REL01-BP05 Mengotomatiskan manajemen kuota
- REL01-BP06 Memastikan adanya selisih yang memadai antara kuota saat ini dan penggunaan maksimum untuk mengakomodasi failover

- REL03-BP01 Memilih cara untuk menyegmentasi beban kerja
- REL10-BP01 Melakukan deployment beban kerja ke beberapa lokasi
- REL11-BP01 Memantau semua komponen beban kerja untuk mendeteksi kegagalan
- REL11-BP03 Mengotomatisasi pemulihan di semua lapisan
- REL12-BP05 Menguji ketahanan menggunakan chaos engineering

Dokumen terkait:

- Pilar Keandalan Kerangka Kerja AWS Well-Architected: Ketersediaan
- AWS Service Quotas (sebelumnya disebut batas layanan)
- AWS Trusted Advisor Pemeriksaan Praktik Terbaik (lihat bagian Batas Layanan)
- Pemantau batas AWS di AWS Answers
- Batas Layanan Amazon EC2
- Apa itu Service Quotas?
- Cara Meminta Peningkatan Kuota
- Endpoint dan kuota layanan
- Panduan Pengguna Service Quotas
- Pemantau Kuota untuk AWS
- Batas Isolasi Kesalahan AWS
- Ketersediaan dengan redundansi
- AWS untuk Data
- Apa itu Integrasi Berkelanjutan?
- Apa itu Pengiriman Berkelanjutan?
- Partner APN: partner yang dapat membantu manajemen konfigurasi
- Mengelola siklus hidup akun di dalam lingkungan SaaS akun per tenant di AWS
- Mengelola dan memanfatau throttling API di dalam beban kerja Anda
- Lihat rekomendasi AWS Trusted Advisor pada skala besar dengan AWS Organizations
- Mengotomatiskan Peningkatan Batas Layanan dan Dukungan Korporat dengan AWS Control Tower
- Tindakan, sumber daya, dan kunci kondisi untuk Service Quotas

Video terkait:

- AWS Live re:Inforce 2019 - Service Quotas
- Melihat dan Mengelola Kuota untuk Layanan AWS Menggunakan Service Quotas
- Demo Kuota AWS IAM
- AWS re:Invent 2018: Menutup Lingkaran dan Membuka Pikiran: Cara Mengendalikan Sistem, Besar dan Kecil

Alat terkait:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP04 Memantau dan mengelola kuota

Evaluasi potensi penggunaan Anda dan tingkatkan kuota dengan semestinya, sehingga terjadi pertumbuhan penggunaan sesuai rencana.

Hasil yang diinginkan: Sistem aktif dan otomatis yang mengelola dan memantau telah di-deploy. Solusi operasi ini memastikan ambang batas penggunaan kuota hampir dicapai. Hal ini akan secara proaktif diperbaiki oleh perubahan kuota yang diminta.

Antipola umum:

- Tidak mengonfigurasi pemantauan untuk memeriksa ambang batas kuota layanan
- Tidak mengonfigurasi pemantauan untuk batas keras, meskipun nilai-nilai tersebut tidak dapat diubah.

- Berasumsi bahwa jumlah waktu yang diperlukan untuk meminta dan mendapatkan perubahan kuota lunak adalah segera atau singkat.
- Mengonfigurasi alarm untuk ketika kuota layanan sudah dekat, namun tidak memiliki proses untuk merespons pemberitahuan.
- Hanya mengonfigurasi alarm untuk layanan yang didukung oleh AWS Service Quotas dan tidak memantau layanan AWS lain.
- Tidak mempertimbangkan manajemen kuota untuk beberapa desain ketangguhan Wilayah, seperti pendekatan aktif/aktif, aktif/pasif - panas, aktif/pasif - dingin, dan aktif/pasif - pilot light.
- Tidak menilai perbedaan kuota antara Wilayah.
- Tidak menilai kebutuhan di setiap Wilayah untuk permintaan peningkatan kuota spesifik.
- Tidak memanfaatkan templat untuk manajemen kuota multi-Wilayah.

Manfaat menjalankan praktik terbaik ini: Dengan melacak AWS Service Quotas secara otomatis dan memantau penggunaan Anda berdasarkan kuota tersebut, Anda dapat mengetahui ketika batas kuota hampir terpenuhi. Anda juga dapat menggunakan data pemantauan ini untuk membantu membatasi penurunan kualitas karena kehabisan kuota.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Untuk layanan yang didukung, Anda dapat memantau kuota Anda dengan mengonfigurasikan berbagai layanan yang berbeda yang dapat menilai kemudian mengirimkan pemberitahuan atau alarm. Hal ini dapat membantu memantau penggunaan dan dapat memberitahukan kuota yang akan habis kepada Anda. Alarm ini dapat dipicu dari AWS Config, fungsi Lambda, Amazon CloudWatch, atau dari AWS Trusted Advisor. Anda juga dapat menggunakan filter metrik di Log CloudWatch untuk mencari dan mengekstrak pola dalam log untuk menentukan apakah penggunaan sudah mendekati ambang batas kuota.

Langkah implementasi

Untuk pemantauan:

- Catat pemakaian sumber daya saat ini (misalnya, bucket atau instans). Gunakan operasi API layanan, seperti Amazon EC2 `DescribeInstances` API, untuk mengumpulkan pemakaian sumber daya saat ini.
- Catat kuota saat ini yang penting dan berlaku untuk layanan menggunakan:

  - AWS Service Quotas
  - AWS Trusted Advisor
  - Dokumentasi AWS
  - Halaman spesifik layanan AWS
  - AWS Command Line Interface (AWS CLI)
  - AWS Cloud Development Kit (AWS CDK)
- Gunakan AWS Service Quotas, yakni layanan AWS yang membantu Anda mengelola kuota untuk lebih dari 250 layanan AWS dari satu lokasi.
- Gunakan batas layanan Trusted Advisor untuk memantau batas layanan Anda saat ini di berbagai ambang batas.
- Gunakan riwayat kuota layanan (konsol atau AWS CLI) untuk memeriksa peningkatan regional.
- Bandingkan perubahan kuota layanan di setiap Wilayah dan setiap akun untuk membuat kesetaraan, jika diperlukan.

Untuk manajemen:

- Otomatis: Siapkan aturan kustom AWS Config untuk memindai kuota layanan di berbagai Wilayah dan bandingkan untuk mengetahui perbedaannya.
- Otomatis: Siapkan fungsi Lambda terjadwal untuk memindai kuota layanan di berbagai Wilayah dan bandingkan untuk mengetahui perbedaannya.
- Manual: Pindai kuota layanan melalui AWS CLI, API, atau Konsol AWS untuk memindai kuota layanan di berbagai Wilayah dan bandingkan untuk mengetahui perbedaannya. Laporkan perbedaannya.
- Jika perbedaan kuota diidentifikasi antara Wilayah, minta perubahan kuota, jika perlu.
- Tinjau hasil dari semua permintaan.

Sumber daya

Praktik Terbaik Terkait:

- REL01-BP01 Kesadaran tentang kuota dan kendala layanan
- REL01-BP02 Mengelola kuota layanan di seluruh akun dan wilayah
- REL01-BP03 Mengakomodasi kuota layanan tetap dan kendala melalui arsitektur

- REL01-BP05 Mengotomatiskan manajemen kuota
- REL01-BP06 Memastikan adanya selisih yang memadai antara kuota saat ini dan penggunaan maksimum untuk mengakomodasi failover
- REL03-BP01 Memilih cara untuk menyegmentasi beban kerja
- REL10-BP01 Melakukan deployment beban kerja ke beberapa lokasi
- REL11-BP01 Memantau semua komponen beban kerja untuk mendeteksi kegagalan
- REL11-BP03 Mengotomatisasi pemulihan di semua lapisan
- REL12-BP05 Menguji ketahanan menggunakan chaos engineering

Dokumen terkait:

- Pilar Keandalan Kerangka Kerja AWS Well-Architected: Ketersediaan
- AWS Service Quotas (sebelumnya disebut batas layanan)
- AWS Trusted Advisor Pemeriksaan Praktik Terbaik (lihat bagian Batas Layanan)
- Pemantau batas AWS di AWS Answers
- Batas Layanan Amazon EC2
- Apa itu Service Quotas?
- Cara Meminta Peningkatan Kuota
- Endpoint dan kuota layanan
- Panduan Pengguna Service Quotas
- Pemantau Kuota untuk AWS
- Batas Isolasi Kesalahan AWS
- Ketersediaan dengan redundansi
- AWS untuk Data
- Apa itu Integrasi Berkelanjutan?
- Apa itu Pengiriman Berkelanjutan?
- Partner APN: partner yang dapat membantu manajemen konfigurasi
- Mengelola siklus hidup akun di dalam lingkungan SaaS akun per tenant di AWS
- Mengelola dan memanfatau throttling API di dalam beban kerja Anda
- Lihat rekomendasi AWS Trusted Advisor pada skala besar dengan AWS Organizations

- Mengotomatiskan Peningkatan Batas Layanan dan Dukungan Korporat dengan AWS Control Tower
- Tindakan, sumber daya, dan kunci kondisi untuk Service Quotas

Video terkait:

- AWS Live re:Inforce 2019 - Service Quotas
- Melihat dan Mengelola Kuota untuk Layanan AWS Menggunakan Service Quotas
- Demo Kuota AWS IAM
- AWS re:Invent 2018: Menutup Lingkaran dan Membuka Pikiran: Cara Mengendalikan Sistem, Besar dan Kecil

Alat terkait:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP05 Mengotomatiskan manajemen kuota

Implementasikan alat untuk memperingatkan Anda saat ambang batas terlampaui. Anda dapat mengotomatiskan permintaan peningkatan kuota dengan menggunakan API AWS Service Quotas, Anda dapat mengotomatiskan permintaan peningkatan kuota.

Jika Anda mengintegrasikan Basis Data Manajemen Konfigurasi (CMDB) atau sistem ticketing dengan Service Quotas, Anda dapat mengotomatiskan permintaan peningkatan kuota dan kuota saat

ini. Selain SDK AWS, Service Quotas menawarkan otomatisasi menggunakan AWS Command Line Interface (AWS CLI).

Antipola umum:

- Melacak kuota dan penggunaan dalam spreadsheet.
- Menjalankan laporan pada penggunaan harian, mingguan, bulanan, lalu membandingkan penggunaan dengan kuota.

Manfaat menerapkan praktik terbaik ini: Dengan pelacakan kuota layanan AWS dan pemantauan terhadap penggunaan kuota, Anda dapat mengetahui ketika kuota hampir penuh. Anda dapat mengatur otomatisasi untuk membantu meminta peningkatan kuota saat diperlukan. Anda dapat mempertimbangkan pengurangan kuota saat penggunaan Anda cenderung tidak selaras untuk benar-benar mengoptimalkan manfaat dari pengurangan risiko (dalam kasus kredensial yang disusupi) dan penghematan biaya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Atur pemantauan otomatis: Implementasikan alat dengan menggunakan SDK untuk memperingatkan saat ambang batas terlampaui.
  - Gunakan Service Quotas dan tingkatkan layanan dengan solusi pemantauan kuota otomatis, seperti AWS Limit Monitor atau penawaran dari AWS Marketplace.
    - Apa itu Service Quotas?
    - Monitor Kuota di AWS - Solusi AWS
  - Atur respons terpicu berdasarkan ambang batas kuota menggunakan Amazon SNS dan API AWS Service Quotas.
  - Uji otomatisasi.
    - Konfigurasikan ambang batas.
    - Integrasikan dengan peristiwa perubahan dari AWS Config, pipeline deployment, Amazon EventBridge, atau pihak ketiga.
    - Atur ambang batas kuota rendah secara artifisial untuk menguji respons.
    - Atur pemicu untuk mengambil tindakan yang sesuai berdasarkan notifikasi dan hubungi AWS Dukungan jika diperlukan.
    - Picu peristiwa perubahan secara manual.

- Jalankan game day untuk menguji proses perubahan peningkatan kuota.

Sumber daya

Dokumen terkait:

- Partner APN: partner yang dapat membantu manajemen konfigurasi
- AWS Marketplace: Produk CMDB yang membantu melacak batasan
- AWS Service Quotas (yang sebelumnya dikenal sebagai batas layanan)
- Pemeriksaan Praktik Terbaik AWS Trusted Advisor (lihat bagian Batas Layanan)
- Monitor Kuota di AWS - Solusi AWS
- Batas Layanan Amazon EC2
- Apa itu Service Quotas?

Video terkait:

- AWS Live re:Inforce 2019 - Service Quotas

REL01-BP06 Memastikan adanya selisih yang memadai antara kuota saat ini dan penggunaan maksimum untuk mengakomodasi failover

Ketika sumber daya gagal atau tidak dapat diakses, sumber daya tersebut mungkin masih dihitung untuk kuota sampai berhasil dihentikan. Verifikasi apakah kuota meliputi tumpang tindih sumber daya yang gagal atau tidak dapat diakses dan penggantiannya. Anda harus mempertimbangkan kasus penggunaan seperti kegagalan jaringan, kegagalan Zona Ketersediaan, atau kegagalan Wilayah ketika menghitung selisih ini.

Hasil yang diinginkan: Kegagalan kecil atau besar dalam sumber daya atau kemampuan akses sumber daya dapat diatasi dalam ambang batas layanan saat ini. Kegagalan zona, kegagalan jaringan, atau bahkan kegagalan Wilayah telah dipertimbangkan dalam perencanaan sumber daya.

Antipola umum:

- Mengatur kuota layanan berdasarkan kebutuhan saat ini tanpa memperhitungkan skenario failover.
- Tidak mempertimbangkan prinsipal stabilitas statis ketika menghitung kuota puncak untuk layanan.
- Tidak mempertimbangkan potensi sumber daya yang tidak dapat diakses dalam menghitung kuota total yang diperlukan untuk setiap Wilayah.

- Tidak mempertimbangkan batas isolasi kesalahan layanan AWS untuk beberapa layanan dan potensi pola penggunaan abnormalnya.

Manfaat menjalankan praktik terbaik ini: Ketika peristiwa gangguan layanan memengaruhi ketersediaan aplikasi, cloud memungkinkan Anda untuk mengimplementasikan strategi guna memitigasi atau memulihkan dari peristiwa ini. Strategi tersebut sering kali mencakup pembuatan sumber daya tambahan untuk menggantikan sumber daya yang gagal atau tidak dapat diakses. Strategi kuota Anda akan mengakomodasi kondisi failover ini dan tidak menambahkan lapisan degradasi tambahan akibat tercapainya batas layanan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Ketika mengevaluasi batas kuota, pertimbangkan kasus failover yang dapat terjadi karena degradasi. Jenis kasus failover berikut ini harus dipertimbangkan:

- VPC yang terganggu atau tidak dapat diakses.
- Subnet yang tidak dapat diakses.
- Zona Ketersediaan telah cukup mengalami degradasi sehingga memengaruhi kemampuan akses banyak sumber daya.
- Berbagai titik keluar dan masuk atau rute jaringan terblokir atau berubah.
- Wilayah telah cukup mengalami degradasi sehingga memengaruhi kemampuan akses banyak sumber daya.
- Ada beberapa sumber daya tetapi tidak semua terpengaruh oleh kegagalan di Wilayah atau Zona Ketersediaan.

Kegagalan seperti yang tercantum di atas dapat menjadi pemicu awal terjadinya peristiwa failover. Keputusan untuk failover bersifat unik untuk setiap situasi dan pelanggan, karena dampak bisnisnya bisa sangat berbeda. Tetapi, ketika memutuskan untuk failover aplikasi atau layanan secara operasional, perencanaan kapasitas sumber daya di lokasi failover dan kuota terkait harus ditangani sebelum peristiwa terjadi.

Tinjau kuota layanan untuk setiap layanan dengan mempertimbangkan puncak yang lebih tinggi dari normal yang mungkin terjadi. Puncak ini mungkin terkait dengan sumber daya yang dapat dicapai karena jaringan atau izin tetapi masih aktif. Sumber daya aktif yang tidak dihentikan akan masih dihitung untuk memenuhi batas kuota layanan.

Langkah implementasi

- Verifikasi bahwa ada selisih yang memadai antara kuota layanan saat ini dan penggunaan maksimum untuk mengakomodasi failover atau hilangnya kemampuan akses.
- Tentukan kuota layanan, perhitungkan pola deployment, persyaratan ketersediaan, dan peningkatan pemakaian.
- Minta peningkatan kuota jika perlu. Rencanakan waktu yang diperlukan agar permintaan peningkatan kuota dapat terpenuhi.
- Tentukan persyaratan keandalan (juga disebut sebagai jumlah angka sembilan Anda).
- Tetapkan skenario kesalahan (misalnya kehilangan komponen, Zona Ketersediaan, atau Wilayah).
- Tetapkan metodologi deployment (misalnya canary, blue/green, red/black, atau rolling).
- Sertakan buffer yang sesuai (misalnya, 15%) ke batas saat ini.
- Sertakan perhitungan untuk stabilitas statis (Zona dan Wilayah) apabila sesuai.
- Rencanakan peningkatan pemakaian (misalnya, memantau tren pemakaian).
- Pertimbangkan dampak stabilitas statis untuk beban kerja Anda yang paling penting. Nilai sumber daya yang sesuai dengan sistem stabil secara statis di semua Wilayah dan Zona Ketersediaan.
- Pertimbangkan penggunaan Reservasi Kapasitas Sesuai Permintaan untuk menjadwalkan kapasitas sebelum failover. Hal ini dapat menjadi strategi yang bermanfaat untuk penjadwalan bisnis yang paling penting guna mengurangi potensi risiko mendapatkan kuantitas dan jenis sumber daya yang benar selama failover.

Sumber daya

Praktik Terbaik Terkait:

- REL01-BP01 Kesadaran tentang kuota dan kendala layanan
- REL01-BP02 Mengelola kuota layanan di seluruh akun dan wilayah
- REL01-BP03 Mengakomodasi kuota layanan tetap dan kendala melalui arsitektur
- REL01-BP04 Memantau dan mengelola kuota
- REL01-BP05 Mengotomatiskan manajemen kuota
- REL03-BP01 Memilih cara untuk menyegmentasi beban kerja
- REL10-BP01 Melakukan deployment beban kerja ke beberapa lokasi
- REL11-BP01 Memantau semua komponen beban kerja untuk mendeteksi kegagalan

- REL11-BP03 Mengotomatisasi pemulihan di semua lapisan
- REL12-BP05 Menguji ketahanan menggunakan chaos engineering

Dokumen terkait:

- Pilar Keandalan Kerangka Kerja AWS Well-Architected: Ketersediaan
- AWS Service Quotas (sebelumnya disebut batas layanan)
- AWS Trusted Advisor Pemeriksaan Praktik Terbaik (lihat bagian Batas Layanan)
- Pemantau batas AWS di AWS Answers
- Batas Layanan Amazon EC2
- Apa itu Service Quotas?
- Cara Meminta Peningkatan Kuota
- Endpoint dan kuota layanan
- Panduan Pengguna Service Quotas
- Pemantau Kuota untuk AWS
- Batas Isolasi Kesalahan AWS
- Ketersediaan dengan redundansi
- AWS untuk Data
- Apa itu Integrasi Berkelanjutan?
- Apa itu Pengiriman Berkelanjutan?
- Partner APN: partner yang dapat membantu manajemen konfigurasi
- Mengelola siklus hidup akun di dalam lingkungan SaaS akun per tenant di AWS
- Mengelola dan memanfatau throttling API di dalam beban kerja Anda
- Lihat rekomendasi AWS Trusted Advisor pada skala besar dengan AWS Organizations
- Mengotomatiskan Peningkatan Batas Layanan dan Dukungan Korporat dengan AWS Control Tower
- Tindakan, sumber daya, dan kunci kondisi untuk Service Quotas

Video terkait:

- AWS Live re:Inforce 2019 - Service Quotas
- Melihat dan Mengelola Kuota untuk Layanan AWS Menggunakan Service Quotas

- Demo Kuota AWS IAM
- AWS re:Invent 2018: Menutup Lingkaran dan Membuka Pikiran: Cara Mengendalikan Sistem, Besar dan Kecil

Alat terkait:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL 2. Bagaimana cara merencanakan topologi jaringan Anda?

Sering kali beban kerja ada di beberapa lingkungan. Ini termasuk beberapa lingkungan cloud (baik yang dapat diakses publik maupun privat) dan kemungkinan infrastruktur pusat data Anda yang ada. Rencana harus mencakup pertimbangan jaringan seperti konektivitas di dalam dan antarsistem, pengelolaan alamat IP publik, pengelolaan alamat IP privat, dan resolusi nama domain.

Praktik terbaik

- REL02-BP01 Menggunakan konektivitas jaringan dengan ketersediaan tinggi untuk titik akhir publik beban kerja Anda
- REL02-BP02 Menyediakan konektivitas redundan antara jaringan privat di cloud dan lingkungan on-premise
- REL02-BP03 Pastikan alokasi subnet IP menjelaskan ekspansi dan ketersediaan
- REL02-BP04 Mengutamakan topologi hub-and-spoke daripada mesh many-to-many
- REL02-BP05 Terapkan rentang alamat IP privat yang tidak tumpang tindih di semua ruang alamat privat tempat semuanya terhubung

REL02-BP01 Menggunakan konektivitas jaringan dengan ketersediaan tinggi untuk titik akhir publik beban kerja Anda

Membuat konektivitas jaringan dengan ketersediaan tinggi untuk titik akhir publik beban kerja Anda dapat membantu Anda mengurangi waktu henti karena hilangnya konektivitas dan meningkatkan ketersediaan serta SLA beban kerja Anda. Untuk mencapai ini, gunakan DNS dengan ketersediaan tinggi, jaringan pengiriman konten (CDN), gateway API, penyeimbangan beban, atau proksi mundur.

Hasil yang diinginkan: Sangat penting untuk merencanakan, membuat, dan mengoperasikan konektivitas jaringan dengan ketersediaan tinggi untuk titik akhir publik Anda. Jika beban kerja Anda menjadi tidak terjangkau karena hilangnya konektivitas, meskipun beban kerja Anda beroperasi dan tersedia, pelanggan Anda akan menganggap sistem Anda tidak berfungsi. Dengan menggabungkan konektivitas jaringan yang tangguh dan memiliki ketersediaan tinggi untuk titik akhir publik beban kerja Anda, bersama dengan arsitektur tangguh untuk beban kerja itu sendiri, Anda dapat memberikan tingkat layanan dan ketersediaan yang sebaik mungkin untuk pelanggan Anda.

AWS Global Accelerator, Amazon CloudFront, Amazon API Gateway, URL Fungsi AWS Lambda, AWS AppSync API, dan Elastic Load Balancing (ELB) memberikan titik akhir publik dengan ketersediaan tinggi. Amazon Route 53 memberikan layanan DNS dengan ketersediaan tinggi untuk resolusi nama domain guna memverifikasi bahwa alamat titik akhir publik Anda dapat diatur.

Anda juga dapat mengevaluasi peralatan perangkat lunak AWS Marketplace untuk penyeimbangan beban dan proksi.

Antipola umum:

- Mendesain beban kerja dengan ketersediaan tinggi tanpa merencanakan konektivitas jaringan dan DNS untuk ketersediaan tinggi.
- Menggunakan alamat internet publik di instans atau kontainer secara individu dan mengelola konektivitasnya dengan DNS.
- Menggunakan alamat IP dan bukannya nama domain untuk mencari layanan.
- Tidak menguji skenario di mana konektivitas ke titik akhir publik Anda hilang.
- Tidak menganalisis pola distribusi dan kebutuhan throughput jaringan.
- Tidak menguji dan merencanakan skenario di mana konektivitas jaringan internet ke titik akhir publik beban kerja Anda mungkin terganggu.
- Memberikan konten (seperti halaman web, aset statis, atau file media) ke area geografis besar dan tidak menggunakan CDN.

- Tidak membuat rencana untuk serangan penolakan layanan terdistribusi (DDoS). Serangan DDoS berisiko menghalangi lalu lintas sah dan menurunkan ketersediaan untuk pengguna Anda.

Manfaat menjalankan praktik terbaik ini: Mendesain konektivitas jaringan yang tangguh dan memiliki ketersediaan tinggi memastikan beban kerja Anda dapat diakses dan tersedia bagi pengguna.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Inti dari pembuatan konektivitas jaringan dengan ketersediaan tinggi untuk titik akhir publik adalah pengarahan rute lalu lintas. Untuk memverifikasi lalu lintas Anda dapat menjangkau titik akhir, DNS harus dapat mengatur nama domain ke alamat IP-nya yang bersangkutan. Gunakan Sistem Nama Domain (DNS) yang dapat diskalakan dan memiliki ketersediaan tinggi seperti Amazon Route 53 untuk mengelola data DNS domain Anda. Anda juga dapat menggunakan pemeriksaan kondisi yang disediakan oleh Amazon Route 53. Pemeriksaan kondisi memverifikasi bahwa aplikasi Anda dapat dijangkau, tersedia, dan berfungsi, dan pemeriksaan ini dapat diatur sedemikian sehingga menyerupai perilaku pengguna Anda, seperti meminta halaman web atau URL tertentu. Jika terjadi kegagalan, Amazon Route 53 merespons permintaan resolusi DNS dan mengarahkan lalu lintas ke titik akhir dengan kondisi bagus saja. Anda juga dapat mempertimbangkan penggunaan kemampuan Perutean Berbasis Latensi dan DNS Geo yang ditawarkan oleh Amazon Route 53.

Untuk memverifikasi bahwa beban kerja itu sendiri memiliki ketersediaan tinggi, gunakan Elastic Load Balancing (ELB). Amazon Route 53 dapat digunakan untuk menargetkan lalu lintas ke ELB, yang mendistribusikan lalu lintas ke instans komputasi target. Anda juga dapat menggunakan Amazon API Gateway bersama dengan AWS Lambda untuk solusi nirserver. Pelanggan juga dapat menjalankan beban kerja di beberapa Wilayah AWS. Dengan pola aktif/aktif multi-lokasi, beban kerja dapat menghadirkan lalu lintas dari beberapa Wilayah. Dengan pola aktif/pasif multi-lokasi, beban kerja menghadirkan lalu lintas dari wilayah aktif sementara data direplikasikan ke wilayah sekunder dan menjadi aktif untuk berjaga-jaga jika terjadi kegagalan di wilayah utama. Kemudian pemeriksaan kondisi Route 53 dapat digunakan untuk mengontrol failover DNS dari titik akhir mana pun di Wilayah utama ke titik akhir di Wilayah sekunder, dengan memverifikasi bahwa beban kerja Anda dapat dijangkau dan tersedia bagi pengguna Anda.

Amazon CloudFront memberikan API sederhana untuk mendistribusikan konten dengan laju transfer data tinggi dan latensi rendah dengan melayani permintaan menggunakan jaringan lokasi edge di seluruh dunia. Jaringan pengiriman konten (CDN) melayani pelanggan dengan menghadirkan konten yang berada di atau di-cache di lokasi yang dekat dengan pengguna. Hal ini juga meningkatkan

ketersediaan aplikasi Anda karena beban untuk konten dialihkan dari server Anda ke CloudFront, yakni ke lokasi edge-nya. Lokasi edge dan cache edge regional menyimpan cache salinan konten Anda dekat dengan penonton Anda sehingga konten dapat diambil dengan cepat, konten lebih mudah dijangkau, dan beban kerja memiliki ketersediaan lebih tinggi.

Untuk beban kerja dengan pengguna yang tersebar secara geografis,AWS Global Accelerator membantu Anda meningkatkan ketersediaan dan performa aplikasi. AWS Global Accelerator memberikan alamat IP statis Anycast yang berfungsi sebagai titik masuk tetap ke aplikasi Anda yang di-host di satu atau lebih Wilayah AWS. Hal ini memungkinkan lalu lintas masuk ke jaringan global AWS sedekat mungkin ke pengguna Anda, yang meningkatkan keterjangkauan dan ketersediaan beban kerja Anda. AWS Global Accelerator juga memantau kondisi titik akhir aplikasi Anda menggunakan TCP, HTTP, dan pemeriksaan kondisi HTTPS. Setiap perubahan kondisi atau konfigurasi titik akhir Anda memicu pengarahan ulang lalu lintas pengguna ke titik akhir dengan kondisi bagus yang memberikan ketersediaan dan performa terbaik bagi pengguna Anda. Selain itu, AWS Global Accelerator memiliki desain yang mengisolasi kesalahan yang menggunakan dua alamat IPv4 status yang dilayani oleh zona jaringan mandiri sehingga meningkatkan ketersediaan aplikasi Anda.

Untuk membantu melindungi pelanggan dari serangan DDoS, AWS memberikan AWS Shield Standard. Shield Standard tersedia sudah secara otomatis diaktifkan dan melindungi dari serangan infrastruktur umum (lapisan 3 dan 4) seperti SYN/UDP flood dan serangan refleksi untuk mendukung ketersediaan tinggi aplikasi Anda di AWS. Untuk perlindungan tambahan dari serangan yang lebih besar dan lebih canggih (seperti UDP flood), serangan penghentian layanan (seperti TCP SYN flood), dan untuk membantu melindungi aplikasi Anda dijalankan di Amazon Elastic Compute Cloud (Amazon EC2), Elastic Load Balancing (ELB), Amazon CloudFront, AWS Global Accelerator, dan Route 53, Anda dapat mempertimbangkan penggunaan AWS Shield Advanced. Untuk perlindungan dari serangan lapisan Aplikasi seperti HTTP POST atau GET flood, gunakan AWS WAF. AWS WAF dapat menggunakan alamat IP, header HTTP, bodi HTTP, string URI, injeksi SQL, dan kondisi skrip lintas situs untuk menentukan apakah permintaan harus diblokir atau diizinkan.

Langkah implementasi

1. Siapkan DNS dengan ketersediaan tinggi: Amazon Route 53 adalah layanan web sistem nama domain (DNS) yang dapat diskalakan dan memiliki ketersediaan tinggi. Route 53 menghubungkan permintaan pengguna dengan aplikasi internet yang dijalankan di AWS atau on-premise. Untuk informasi selengkapnya, lihat mengonfigurasi Amazon Route 53 sebagai layanan DNS Anda.
2. Siapkan pemeriksaan kondisi: Ketika menggunakan Route 53, verifikasi bahwa hanya target dengan kondisi bagus yang dapat diselesaikan. Mulai dengan membuat pemeriksaan kondisi

Route 53 dan mengonfigurasi failover DNS. Aspek-aspek berikut penting untuk dipertimbangkan ketika mempersiapkan pemeriksaan kondisi:

a. Bagaimana Amazon Route 53 menentukan apakah pemeriksaan kondisi bagus

b. Membuat, memperbarui, dan menghapus pemeriksaan kondisi

c. Memantau status pemeriksaan kondisi dan mendapatkan pemberitahuan

d. Praktik terbaik untuk Amazon Route 53 DNS

3. Hubungkan layanan DNS Anda ke titik akhir Anda.

   a. Ketika menggunakan Elastic Load Balancing sebagai target untuk lalu lintas Anda, buat catatan alias menggunakan Amazon Route 53 yang menunjuk ke titik akhir wilayah penyeimbang beban Anda. Selama pembuatan catatan alias, atur opsi Evaluasi kondisi target ke Ya.

   b. Untuk beban kerja nirserver atau API privat ketika API Gateway digunakan, gunakan Route 53 untuk mengarahkan lalu lintas ke API Gateway.

4. Tentukan jaringan pengiriman konten.

   a. Untuk pengiriman konten menggunakan lokasi edge yang lebih dekat dengan pengguna, mulai dengan memahami cara CloudFront memberikan konten.

   b. Mulai dengan distribusi CloudFront sederhana. Kemudian CloudFront mengetahui dari mana Anda ingin konten dikirimkan, dan detail tentang cara melacak dan mengelola pengiriman konten. Aspek-aspek berikut penting untuk dipahami dan dipertimbangkan ketika mempersiapkan distribusi CloudFront:

      i. Cara kerja caching dengan lokasi edge CloudFront

      ii. Meningkatkan proporsi permintaan yang dihadirkan secara langsung dari cache CloudFront (rasio sukses cache)

      iii. Menggunakan Amazon CloudFront Origin Shield

      iv. Mengoptimalkan ketersediaan tinggi dengan failover asalah CloudFront

5. Siapkan perlindungan lapisan aplikasi: AWS WAF membantu Anda melindungi dari bot dan eksploitasi web umum yang dapat memengaruhi ketersediaan, mengancam keamanan, atau memakai sumber daya secara berlebihan. Untuk mendapatkan pemahaman lebih mendalam, tinjau cara kerja AWS WAF dan kapan Anda siap untuk mengimplementasikan perlindungan dari HTTP POST DAN GET flood lapisan aplikasi, tinjau Memulai AWS WAF. Anda juga dapat menggunakan AWS WAF dengan CloudFront, lihat dokumentasi tentang cara AWS WAF berfungsi dengan fitur Amazon CloudFront.

6. Siapkan perlindungan DDoS tambahan: Menurut default, semua pelanggan AWS menerima perlindungan dari serangan DDoS lapisan transpor dan jaringan paling sering terjadi yang

menargetkan situs web atau aplikasi Anda, dengan menggunakan AWS Shield Standard tanpa biaya tambahan. Untuk perlindungan tambahan aplikasi yang dilihat internet, dijalankan di Amazon EC2, Elastic Load Balancing, Amazon CloudFront, AWS Global Accelerator, dan Amazon Route 53, Anda dapat mempertimbangkan AWS Shield Advanced dan meninjau contoh tentang arsitektur yang tangguh terhadap DDoS. Untuk melindungi beban kerja dan titik akhir publik Anda dari serangan DDoS, tinjau Memulai dengan AWS Shield Advanced.

Sumber daya

Praktik Terbaik Terkait:

- REL10-BP01 Melakukan deployment beban kerja ke beberapa lokasi
- REL10-BP02 Memilih lokasi yang sesuai untuk deployment multilokasi
- REL11-BP04 Mengandalkan bidang data dan bukan bidang kendali selama pemulihan
- REL11-BP06 Mengirimkan notifikasi ketika peristiwa memengaruhi ketersediaan

Dokumen terkait:

- Partner APN: partner yang dapat membantu merencanakan jaringan Anda
- AWS Marketplace untuk Infrastruktur Jaringan
- Apa Itu AWS Global Accelerator?
- Apa itu Amazon CloudFront?
- Apa itu Amazon Route 53?
- Apa itu Elastic Load Balancing?
- Kemampuan Konektivitas Jaringan - Menetapkan Fondasi Cloud Anda
- Apa itu Amazon API Gateway?
- Apa itu AWS WAF, AWS Shield, dan AWS Firewall Manager?
- Apa itu Pengontrol Pemulihan Aplikasi Amazon Route 53?
- Konfigurasi pemeriksaan kondisi kustom untuk failover DNS

Video terkait:

- AWS re:Invent 2022 - Meningkatkan performa dan ketersediaan dengan AWS Global Accelerator

- AWS re:Invent 2020: Manajemen lalu lintas global dengan Amazon Route 53
- AWS re:Invent 2022 - Mengoperasikan aplikasi Multi-AZ dengan ketersediaan tinggi
- AWS re:Invent 2022 - Memahami infrastruktur jaringan AWS
- AWS re:Invent 2022 - Membangun jaringan tangguh

Contoh terkait:

- Pemulihan Bencana dengan Pengontrol Pemulihan Aplikasi (ARC) Amazon Route 53
- Lokakarya Keandalan
- Lokakarya AWS Global Accelerator

REL02-BP02 Menyediakan konektivitas redundan antara jaringan privat di cloud dan lingkungan on-premise

Gunakan beberapa koneksi AWS Direct Connect atau terowongan VPN antara jaringan privat yang di-deploy secara terpisah. Gunakan beberapa lokasi Direct Connect untuk ketersediaan tinggi. Ketika menggunakan beberapa Wilayah AWS, pastikan ada redundansi setidaknya di dalam dua di antaranya. Anda dapat mengevaluasi peralatan AWS Marketplace yang menghentikan VPN. Ketika menggunakan peralatan AWS Marketplace, lakukan deployment instans redundan untuk ketersediaan tinggi di Zona Ketersediaan yang berbeda.

AWS Direct Connect adalah layanan cloud yang memudahkan Anda menetapkan koneksi jaringan khusus dari lingkungan on-premise ke AWS. Dengan menggunakan Gateway Direct Connect, pusat data on-premise dapat dihubungkan ke beberapa VPC AWS yang tersebar di seluruh Wilayah AWS.

Redundansi ini menangani kemungkinan kesalahan yang berdampak pada ketangguhan konektivitas:

- Bagaimana cara bertahan dari kesalahan dalam topologi?
- Apa yang terjadi jika Anda salah mengonfigurasi sesuatu dan menghapus konektivitas?
- Apakah Anda akan mampu untuk menangani peningkatan lalu lintas atau penggunaan layanan yang tidak terduga?
- Apakah Anda akan mampu untuk menahan percobaan serangan Distributed Denial of Service (DDoS)?

Saat menghubungkan VPC ke pusat data on-premise melalui VPN, sebaiknya pertimbangkan persyaratan ketahanan dan bandwidth yang diperlukan ketika memilih vendor dan ukuran instans

yang dibutuhkan untuk menjalankan peralatan. Jika Anda menggunakan perangkat VPN yang tidak tangguh dalam implementasinya, maka Anda harus memiliki koneksi redundan melalui perangkat kedua. Untuk semua skenario ini, Anda perlu menentukan waktu yang dapat diterima untuk pemulihan dan pengujian guna memastikan bahwa Anda memenuhi persyaratan tersebut.

Jika Anda memilih untuk menghubungkan VPC ke pusat data menggunakan koneksi Direct Connect dan koneksi ini harus selalu tersedia, gunakan koneksi Direct Connect yang redundan dari setiap pusat data. Koneksi redundan harus menggunakan koneksi Direct Connect kedua yang berbeda lokasi dengan yang pertama. Jika Anda memiliki beberapa pusat data, pastikan bahwa koneksinya berakhir di lokasi yang berbeda. Gunakan Kit Alat Ketahanan Direct Connect untuk membantu menyiapkan ini.

Jika Anda memilih untuk melakukan failover VPN melalui internet menggunakan Site-to-Site VPN, penting untuk dipahami bahwa hal ini mendukung hingga 1,25 Gbps throughput per terowongan VPN, tetapi tidak mendukung Equal Cost Multi Path (ECMP) untuk lalu lintas keluar dalam kasus beberapa terowongan VPN Terkelola AWS yang berakhir pada VGW yang sama. Sebaiknya jangan gunakan VPN Terkelola AWS sebagai cadangan koneksi Direct Connect kecuali jika Anda dapat menoleransi kecepatan kurang dari 1 Gbps selama failover.

Anda juga dapat menggunakan titik akhir VPC agar VPC terhubung secara privat ke layanan yang didukung AWS dan layanan titik akhir VPC yang didukung oleh AWS PrivateLink tanpa melewati internet publik. Titik akhir adalah perangkat virtual. Titik akhir dapat diskalakan secara horizontal, redundan, dan merupakan komponen VPC yang selalu tersedia. Titik akhir memungkinkan komunikasi antarinstans dalam layanan dan VPC tanpa memaksakan risiko ketersediaan atau batasan bandwidth pada lalu lintas jaringan.

Antipola umum:

- Hanya memiliki satu penyedia konektivitas antara jaringan on-site dan AWS.
- Menggunakan kemampuan konektivitas dari koneksi AWS Direct Connect, tetapi hanya memiliki satu koneksi.
- Hanya memiliki satu jalur untuk konektivitas VPN.

Manfaat menerapkan praktik terbaik ini: Dengan mengimplementasikan konektivitas redundan antara lingkungan cloud dan lingkungan perusahaan atau on-premise, Anda dapat memastikan bahwa layanan dependen antara dua lingkungan tersebut dapat berkomunikasi dengan andal.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Pastikan bahwa Anda memiliki konektivitas yang tersedia antara AWS dan lingkungan on-premise. Gunakan beberapa koneksi AWS Direct Connect atau terowongan VPN antara jaringan privat yang di-deploy secara terpisah. Gunakan beberapa lokasi Direct Connect untuk ketersediaan tinggi. Ketika menggunakan beberapa Wilayah AWS, pastikan ada redundansi setidaknya di dalam dua di antaranya. Anda dapat mengevaluasi peralatan AWS Marketplace yang menghentikan VPN. Ketika menggunakan peralatan AWS Marketplace, lakukan deployment instans redundan untuk ketersediaan tinggi di Zona Ketersediaan yang berbeda.
  - Pastikan bahwa Anda memiliki koneksi redundan ke lingkungan on-premise. Anda mungkin memerlukan koneksi redundan ke beberapa Wilayah AWS untuk mencapai ketersediaan yang dibutuhkan.
    - Rekomendasi Ketangguhan AWS Direct Connect
    - Menggunakan Koneksi VPN Site-to-Site Redundan untuk Menyediakan Failover
      - Gunakan layanan operasi API untuk mengidentifikasi penggunaan yang tepat dari sirkuit Direct Connect.
        - DescribeConnections
        - DescribeConnectionsOnInterconnect
        - DescribeDirectConnectGatewayAssociations
        - DescribeDirectConnectGatewayAttachments
        - DescribeDirectConnectGateways
        - DescribeHostedConnections
        - DescribeInterconnects
      - Jika hanya ada satu koneksi Direct Connect atau Anda tidak memilikinya sama sekali, atur terowongan VPN redundan ke gateway privat virtual Anda.
        - Apa itu VPN Site-to-Site AWS?
  - Tangkap konektivitas saat ini (misalnya, gateway pribadi virtual, peralatan AWS Marketplace).
    - Gunakan layanan operasi API untuk memasukkan konfigurasi koneksi Direct Connect.
      - DescribeConnections
      - DescribeConnectionsOnInterconnect
      - DescribeDirectConnectGatewayAssociations
      - DescribeDirectConnectGatewayAttachments
      - DescribeDirectConnectGateways

    - DescribeHostedConnections
    - DescribeInterconnects
- Gunakan layanan operasi API untuk mengumpulkan gateway privat virtual ketika tabel rute menggunakannya.
    - DescribeVpnGateways
    - DescribeRouteTables
- Gunakan layanan operasi API untuk mengumpulkan aplikasi AWS Marketplace ketika tabel rute menggunakannya.
    - DescribeRouteTables

Sumber daya

Dokumen terkait:

- Partner APN: partner yang dapat membantu merencanakan jaringan Anda
- Rekomendasi Ketangguhan AWS Direct Connect
- AWS Marketplace untuk Infrastruktur Jaringan
- Laporan Resmi Opsi Konektivitas Amazon Virtual Private Cloud
- Konektivitas jaringan ketersediaan tinggi (HA) beberapa pusat data
- Menggunakan Koneksi VPN Site-to-Site Redundan untuk Menyediakan Failover
- Menggunakan Kit Alat Ketahanan Direct Connect untuk memulai
- Titik Akhir VPC dan Layanan Titik Akhir VPC (AWS PrivateLink)
- Apa Itu Amazon VPC?
- Apa Itu Transit Gateway?
- Apa itu VPN Site-to-Site AWS?
- Bekerja dengan Gateway Direct Connect

Video terkait:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs (NET406-R1)

REL02-BP03 Pastikan alokasi subnet IP menjelaskan ekspansi dan ketersediaan

Rentang alamat IP Amazon VPC harus cukup besar untuk mengakomodasi persyaratan beban kerja, termasuk pertimbangan ekspansi mendatang dan alokasi alamat IP ke subnet di seluruh Zona Ketersediaan. Ini mencakup penyeimbang beban, instans EC2, dan aplikasi berbasis kontainer.

Ketika Anda merencanakan topologi jaringan Anda, langkah pertama adalah menetapkan ruang alamat IP itu sendiri. Rentang alamat IP privat (mengikuti pedoman RFC 1918) harus dialokasikan untuk setiap VPC. Akomodasikan persyaratan berikut sebagai bagian dari proses ini:

- Berikan ruang alamat IP untuk lebih dari satu VPC per Wilayah.
- Di dalam VPC, berikan ruang untuk beberapa subnet yang meliputi beberapa Zona Ketersediaan.
- Selalu biarkan ruang blok CIDR yang tidak digunakan di dalam VPC untuk ekspansi mendatang.
- Pastikan ada ruang alamat IP untuk memenuhi kebutuhan armada sementara instans EC2 yang mungkin Anda gunakan, seperti Armada Spot untuk machine learning, klaster Amazon EMR, atau klaster Amazon Redshift.
- Perhatikan, empat alamat IP pertama dan alamat IP terakhir di setiap blok CIDR subnet disimpan dan tidak tersedia untuk Anda gunakan.
- Anda harus merencanakan untuk melakukan deploy blok CIDR VPC besar. Perhatikan, blok CIDR VPC awal yang dialokasikan ke VPC Anda tidak dapat diubah atau dihapus, tetapi Anda dapat menambahkan tambahan blok CIDR yang tidak tumpang tindih ke VPC. CIDR IPv4 subnet tidak dapat diubah, tetapi CIDR IPv6 dapat diubah. Ingat, deployment VPC yang sebesar mungkin (/16) menghasilkan lebih dari 65.000 alamat IP. Di ruang alamat IP 10.x.x.x saja, Anda dapat menyediakan 255 VPC seperti ini. Oleh karena itu, Anda harus cenderung terlalu besar daripada terlalu kecil untuk mempermudah pengelolaan VPC Anda.

Antipola umum:

- Membuat VPC yang kecil.
- Membuat subnet kecil lalu harus menambahkan subnet ke konfigurasi seiring pertumbuhan.
- Salah memperkirakan jumlah alamat IP yang dapat digunakan penyeimbang beban elastis.
- Melakukan deploy banyak penyeimbang beban lalu lintas tinggi ke subnet yang sama.

Manfaat menerapkan praktik terbaik ini: Ini memastikan bahwa Anda dapat mengakomodasi pertumbuhan beban kerja Anda dan terus memberikan ketersediaan saat Anda menaikkan skala.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Rencanakan jaringan Anda untuk mengakomodasi pertumbuhan, kepatuhan terhadap peraturan, dan integrasi dengan yang lain. Pertumbuhan dapat lebih besar dari yang diperkirakan, kepatuhan terhadap peraturan dapat berubah, dan koneksi jaringan privat atau akuisisi dapat sulit diimplementasikan tanpa perencanaan yang baik.
  - Pilih Wilayah dan Akun AWS yang relevan berdasarkan persyaratan layanan, latensi, peraturan, dan pemulihan bencana (DR) Anda.
  - Identifikasi kebutuhan Anda untuk deployment VPC regional.
  - Identifikasi ukuran VPC.
    - Tentukan apakah Anda akan melakukan deploy konektivitas multi-VPC.
      - Apa Itu Gateway Transit?
      - Konektivitas Multi-VPC Satu Wilayah
    - Tentukan apakah Anda membutuhkan jaringan terpisah untuk persyaratan peraturan.
    - Buat VPC yang sebesar mungkin. Blok CIDR VPC awal yang dialokasikan ke VPC Anda tidak dapat diubah atau dihapus, tetapi Anda dapat menambahkan tambahan blok CIDR yang tidak tumpang tindih ke VPC. Tetapi, ini dapat memotong rentang alamat Anda.
    - Buat VPC yang sebesar mungkin. Blok CIDR VPC awal yang dialokasikan ke VPC Anda tidak dapat diubah atau dihapus, tetapi Anda dapat menambahkan tambahan blok CIDR yang tidak tumpang tindih ke VPC. Tetapi, ini dapat memotong rentang alamat Anda.

Sumber daya

Dokumen terkait:

- Partner APN: partner yang dapat membantu merencanakan jaringan Anda
- AWS Marketplace untuk Infrastruktur Jaringan
- Laporan Resmi Opsi Konektivitas Amazon Virtual Private Cloud
- Konektivitas jaringan ketersediaan tinggi (HA) beberapa pusat data
- Konektivitas Multi-VPC Satu Wilayah
- Apa Itu Amazon VPC?

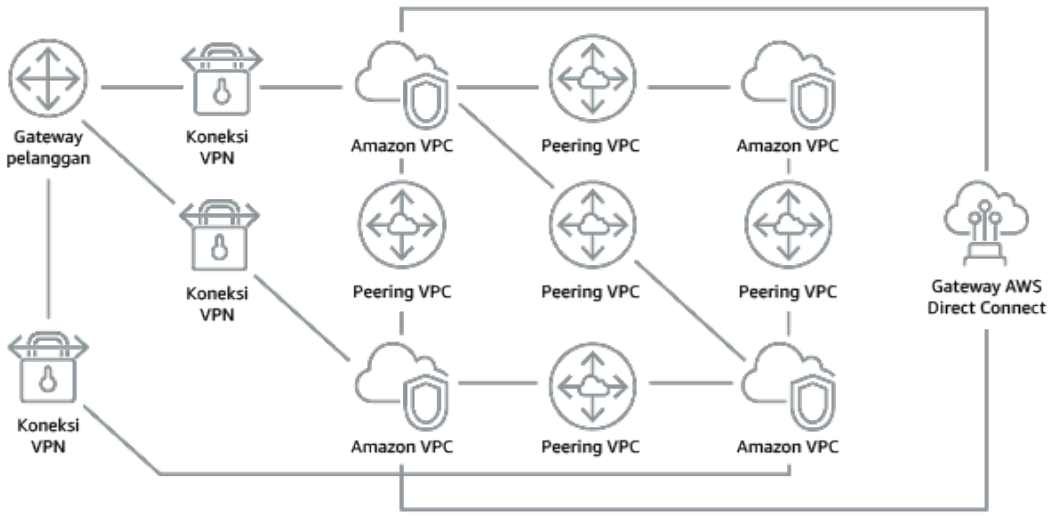
Video terkait:

- AWS re:Invent 2018: Desain VPC Tingkat Lanjut dan Kemampuan Baru untuk Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Arsitektur referensi Gateway Transit untuk berbagai VPC (NET406-R1)

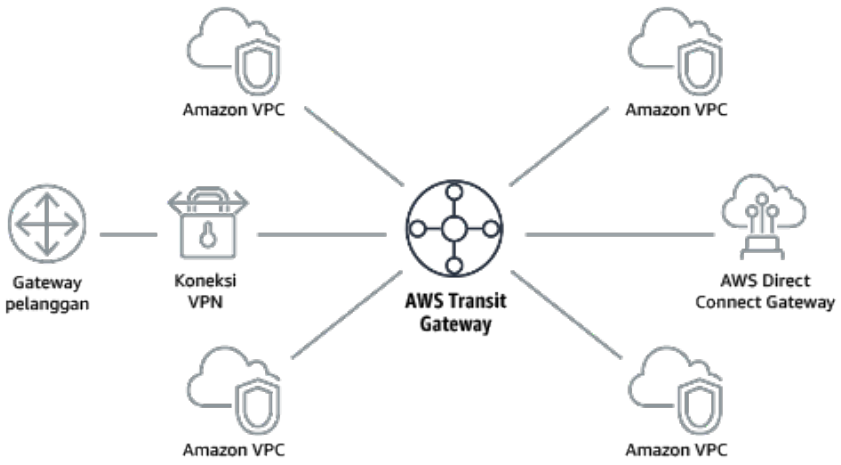
REL02-BP04 Mengutamakan topologi hub-and-spoke daripada mesh many-to-many

Jika ada lebih dari dua ruang alamat jaringan (misalnya, jaringan VPC dan on-premise) yang terhubung melalui peering VPC, AWS Direct Connect, atau VPN, gunakan model hub-and-spoke, seperti yang disediakan oleh AWS Transit Gateway.

Jika hanya ada dua jaringan, Anda dapat langsung menghubungkannya satu sama lain, tetapi seiring dengan bertambahnya jaringan, kompleksitas koneksi mesh ini tidak dapat dipertahankan. AWS Transit Gateway menyediakan model hub-and-spoke yang mudah dipertahankan, yang memungkinkan perutean lalu lintas ke beberapa jaringan.



Gambar 1: Tanpa AWS Transit Gateway: Anda perlu membuat peer setiap Amazon VPC satu sama lain dan ke setiap lokasi lokal menggunakan koneksi VPN, yang dapat menjadi kompleks seiring dengan meningkatnya skala.



Gambar 2: Dengan AWS Transit Gateway: Anda dapat langsung menghubungkan Amazon VPC atau VPN ke AWS Transit Gateway, dan ini merutekan lalu lintas ke dan dari setiap VPC atau VPN.

Antipola umum:

- Menggunakan peering VPC untuk menghubungkan lebih dari dua VPC.
- Membuat beberapa sesi BGP untuk setiap VPC guna membuat konektivitas yang memperluas penyebaran Cloud Privat Virtual (VPC) ke beberapa Wilayah AWS.

Manfaat menerapkan praktik terbaik ini: Seiring bertambahnya jumlah jaringan, kompleksitas koneksi mesh ini tidak dapat dipertahankan. AWS Transit Gateway menyediakan model hub-and-spoke yang mudah dipertahankan, yang memungkinkan perutean lalu lintas ke beberapa jaringan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Mengutamakan topologi hub-and-spoke daripada mesh many-to-many. Jika ada lebih dari dua ruang alamat jaringan (jaringan VPC, on-premise) yang terhubung melalui peering VPC, AWS Direct Connect, atau VPN, gunakan model hub-and-spoke, seperti yang disediakan oleh AWS Transit Gateway.
  - Jika hanya dua jaringan, Anda dapat langsung menghubungkannya satu sama lain, tetapi seiring dengan bertambahnya jaringan, kompleksitas koneksi mesh ini tidak dapat dipertahankan. AWS Transit Gateway menyediakan model hub-and-spoke yang mudah dipertahankan, yang memungkinkan perutean lalu lintas ke beberapa jaringan.

- Apa Itu Transit Gateway?

Sumber daya

Dokumen terkait:

- Partner APN: partner yang dapat membantu merencanakan jaringan Anda
- AWS Marketplace untuk Infrastruktur Jaringan
- Konektivitas jaringan ketersediaan tinggi (HA) beberapa pusat data
- Titik Akhir VPC dan Layanan Titik Akhir VPC (AWS PrivateLink)
- Apa Itu Amazon VPC?
- Apa Itu Transit Gateway?

Video terkait:

- AWS re:Invent 2018: Desain VPC Tingkat Lanjut dan Kemampuan Baru untukAmazon VPC (NET303)
- AWS re:Invent 2019: Arsitektur referensi AWS Transit Gateway untuk banyak VPC (NET406-R1)

REL02-BP05 Terapkan rentang alamat IP privat yang tidak tumpang tindih di semua ruang alamat privat tempat semuanya terhubung

Rentang alamat IP untuk setiap VPC Anda tidak boleh tumpang tindih ketika peering atau dihubungkan lewat VPN. Anda juga harus menghindari konflik alamat IP antara lingkungan on-premise dan VPC atau dengan penyedia cloud lain yang Anda gunakan. Selain itu, Anda harus memiliki cara untuk mengalokasikan rentang alamat IP privat ketika dibutuhkan.

Sistem manajemen alamat IP (IPAM) dapat membantu hal ini. Beberapa IPAM tersedia dari AWS Marketplace.

Antipola umum:

- Menggunakan rentang IP yang sama di VPC Anda seperti yang Anda miliki on-premise atau di jaringan korporasi Anda.
- Tidak melacak rentang IP VPC yang digunakan untuk deployment beban kerja Anda.

Manfaat menerapkan praktik terbaik ini: Perencanaan aktif jaringan Anda akan memastikan bahwa Anda tidak memiliki beberapa kejadian alamat IP yang sama di jaringan yang saling terhubung. Ini mencegah timbulnya masalah perutean di bagian beban kerja yang menggunakan aplikasi yang berbeda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Pantau dan kelola penggunaan CIDR Anda. Evaluasi potensi penggunaan Anda di AWS, tambahkan rentang CIDR ke VPC yang ada, dan buat VPC untuk memungkinkan pertumbuhan yang direncanakan dalam penggunaan.
  - Catat konsumsi CIDR saat ini (misalnya, VPC, subnet)
    - Gunakan operasi API layanan untuk mengumpulkan konsumsi CIDR saat ini.
  - Catat penggunaan subnet Anda saat ini.
    - Gunakan operasi API layanan untuk mengumpulkan subnet per VPC di setiap Wilayah.
      - DescribeSubnets
    - Catat penggunaan saat ini.
    - Tentukan apakah Anda telah membuat rentang IP yang tumpang tindih.
    - Hitung kapasitas cadangan.
    - Identifikasi rentang IP yang tumpang tindih. Anda dapat memigrasikan ke rentang alamat baru atau menggunakan peralatan Network and Port Translation (NAT) dari AWS Marketplace jika Anda harus menghubungkan rentang yang tumpang tindih.

Sumber daya

Dokumen terkait:

- Partner APN: partner yang dapat membantu merencanakan jaringan Anda
- AWS Marketplace untuk Infrastruktur Jaringan
- Laporan Resmi Opsi Konektivitas Amazon Virtual Private Cloud
- Konektivitas jaringan ketersediaan tinggi (HA) beberapa pusat data
- Apa Itu Amazon VPC?
- Apa itu IPAM?

Video terkait:

- AWS re:Invent 2018: Desain VPC Tingkat Lanjut dan Kemampuan Baru untuk Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Arsitektur referensi Gateway Transit untuk berbagai VPC (NET406-R1)

Arsitektur beban kerja

Pertanyaan

- REL 3. Bagaimana cara mendesain arsitektur layanan beban kerja Anda?
- REL 4. Bagaimana cara mendesain interaksi di sistem terdistribusi untuk mencegah kegagalan?
- REL 5. Bagaimana cara mendesain interaksi di sistem terdistribusi untuk memitigasi atau bertahan dari kegagalan?

REL 3. Bagaimana cara mendesain arsitektur layanan beban kerja Anda?

Bangun beban kerja yang andal dan dapat diskalakan dengan mudah menggunakan arsitektur berorientasi layanan (SOA) atau arsitektur layanan mikro. Arsitektur berorientasi layanan (SOA) adalah praktik untuk membuat komponen perangkat lunak dapat digunakan ulang lewat antarmuka layanan. Arsitektur layanan mikro melangkah lebih jauh untuk membuat komponen menjadi lebih kecil dan lebih sederhana.

Praktik terbaik

- REL03-BP01 Memilih cara untuk menyegmentasi beban kerja
- REL03-BP02 Bangun layanan yang berfokus pada domain dan fungsionalitas bisnis khusus
- REL03-BP03 Memberikan kontrak layanan per API

REL03-BP01 Memilih cara untuk menyegmentasi beban kerja

Segmentasi beban kerja penting saat menentukan persyaratan ketahanan aplikasi Anda. Arsitektur monolitik harus dihindari jika memungkinkan. Sebagai gantinya, pertimbangkan dengan cermat komponen aplikasi mana yang dapat dipecah menjadi layanan mikro. Bergantung pada persyaratan aplikasi Anda, solusinya mungkin merupakan kombinasi arsitektur berorientasi layanan (SOA) dengan layanan mikro jika memungkinkan. Beban kerja yang mampu berada dalam kondisi stateless akan lebih mampu di-deploy sebagai layanan mikro.

Hasil yang diinginkan: Beban kerja harus dapat didukung, dapat diskalakan, dan di-coupling selonggar mungkin.

Saat membuat pilihan tentang cara menyegmentasikan beban kerja Anda, seimbangkan manfaat dengan kerumitannya. Hal yang tepat untuk produk baru yang mengejar jadwal peluncuran pertama akan berbeda dengan hal yang dibutuhkan oleh beban kerja yang dibangun untuk diskalakan dari awal. Saat memfaktor ulang monolit yang ada, Anda perlu mempertimbangkan seberapa baik aplikasi akan mendukung dekomposisi menuju kondisi stateless. Dengan memecah layanan menjadi bagian-bagian yang lebih kecil, tim kecil yang diberi tanggung jawab khusus akan dapat mengembangkan dan mengelolanya. Namun, layanan yang lebih kecil dapat menimbulkan kompleksitas yang mencakup kemungkinan peningkatan latensi, debugging yang lebih kompleks, dan peningkatan beban operasional.

Antipola umum:

- Dalam layanan mikro, Death Star adalah situasi saat komponen atomik menjadi sangat saling bergantung sehingga kegagalan salah satu komponen akan menghasilkan kegagalan yang jauh lebih besar, sehingga komponen ini pun menjadi kaku dan rapuh seperti monolit.

Manfaat menjalankan praktik ini:

- Segmen yang lebih spesifik akan menghasilkan ketangkasan, fleksibilitas organisasi, dan skalabilitas yang lebih besar.
- Dampak gangguan layanan yang berkurang.
- Komponen-komponen aplikasi mungkin memiliki persyaratan ketersediaan yang berbeda-beda, yang dapat didukung oleh segmentasi yang lebih atomik.
- Tanggung jawab yang ditentukan khusus untuk tim yang mendukung beban kerja.

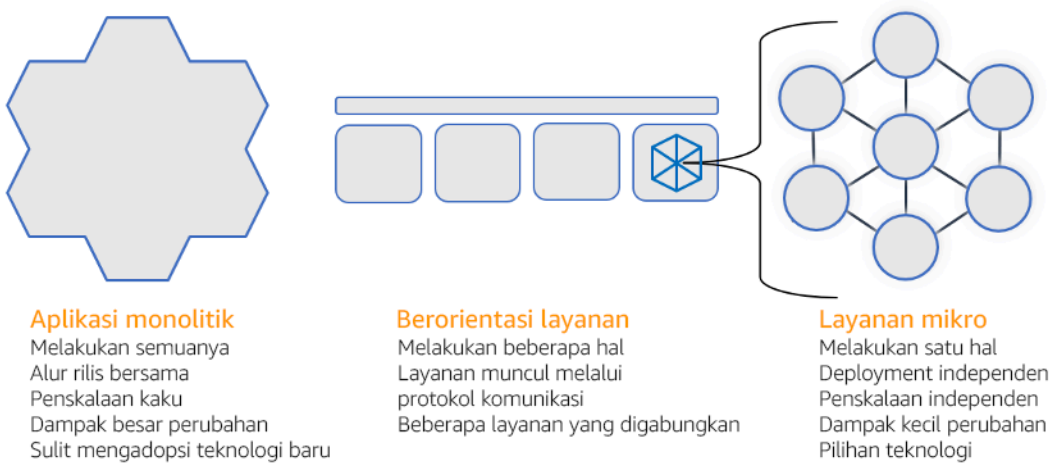
Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Pilih jenis arsitektur berdasarkan cara beban kerja disegmentasikan. Pilih arsitektur SOA atau layanan mikro (atau dalam beberapa kasus yang jarang terjadi, arsitektur monolitik). Bahkan jika Anda memilih untuk memulai arsitektur monolit, Anda harus memastikan bahwa arsitektur tersebut modular dan pada akhirnya dapat berkembang menjadi SOA atau layanan mikro seiring dengan skala produk Anda dengan adopsi pengguna. SOA dan layanan mikro masing-masing menawarkan segmentasi yang lebih kecil, yang lebih disarankan sebagai arsitektur modern yang dapat diskalakan

dan andal, tetapi ada tarik ulur yang perlu dipertimbangkan, terutama saat melakukan deployment arsitektur layanan mikro.

Salah satu tarik ulur utama adalah Anda sekarang memiliki arsitektur komputasi terdistribusi yang dapat mempersulit dalam memenuhi persyaratan latensi pengguna dan ada kerumitan tambahan dalam proses debugging dan penelusuran interaksi pengguna. Anda dapat menggunakan AWS X-Ray untuk membantu Anda memecahkan masalah ini. Efek lain yang perlu dipertimbangkan adalah peningkatan kompleksitas operasional seiring Anda meningkatkan jumlah aplikasi yang Anda kelola, yang memerlukan deployment banyak komponen independen.



Arsitektur monolitik, berorientasi layanan, dan layanan mikro

Langkah implementasi

- Tentukan arsitektur yang sesuai untuk memfaktor ulang atau membangun aplikasi Anda. SOA dan layanan mikro masing-masing menawarkan segmentasi yang lebih kecil, serta diutamakan sebagai arsitektur modern yang dapat diskalakan dan diandalkan. SOA dapat menjadi kompensasi yang baik untuk mencapai segmentasi yang lebih kecil sembari menghindari beberapa kompleksitas dari layanan mikro. Untuk detail selengkapnya, lihat Tarik Ulur Layanan Mikro.
- Jika dapat diterima beban kerja dan didukung organisasi, Anda harus menggunakan arsitektur layanan mikro untuk mencapai ketangkasan dan keandalan terbaik. Untuk detail selengkapnya, lihat Menerapkan Layanan Mikro di AWS.
- Pertimbangkan untuk mengikuti karakteristik pohon Strangler Fig , yaitu pola untuk memfaktor ulang monolit menjadi komponen yang lebih kecil. Hal ini memerlukan penggantian komponen aplikasi tertentu secara bertahap dengan aplikasi dan layanan baru. AWS Migration Hub Refactor Spaces bertindak sebagai titik awal untuk pemfaktoran ulang secara bertahap. Untuk detail

selengkapnya, lihat Memigrasikan beban kerja lama di on-premise dengan lancar menggunakan pola strangler.

- Implementasi layanan mikro mungkin memerlukan mekanisme penemuan layanan untuk memungkinkan layanan terdistribusi ini berkomunikasi satu sama lain. AWS App Mesh dapat digunakan dengan arsitektur berorientasi layanan untuk menyediakan penemuan dan akses layanan yang andal. AWS Cloud Map juga dapat digunakan untuk penemuan layanan berbasis DNS yang dinamis.
- Jika Anda bermigrasi dari monolit ke SOA, Amazon MQ dapat membantu menjembatani kesenjangan sebagai bus layanan saat mendesain ulang aplikasi lama di cloud.
- Untuk monolit yang ada dengan satu basis data bersama, pilih cara mengatur ulang data menjadi segmen yang lebih kecil. Segmentasi ini dapat didasarkan pada unit bisnis, pola akses, atau struktur data. Pada titik ini dalam proses pemfaktoran ulang, Anda harus memilih untuk melanjutkan dengan jenis basis data relasional atau nonrelasional (NoSQL). Untuk detail selengkapnya, lihat Dari SQL ke NoSQL.

Tingkat upaya untuk rencana implementasi: Tinggi

Sumber daya

Praktik terbaik terkait:

- REL03-BP02 Bangun layanan yang berfokus pada domain dan fungsionalitas bisnis khusus

Dokumen terkait:

- Amazon API Gateway: Mengonfigurasi API REST Menggunakan OpenAPI
- Apa itu Arsitektur Berorientasi Layanan?
- Konteks Terikat (pola sentral di Desain yang Didorong Domain)
- Mengimplementasikan Layanan Mikro di AWS
- Kompensasi Layanan Mikro
- Layanan mikro - definisi istilah arsitektur baru ini
- Layanan mikro di AWS
- Apa itu AWS App Mesh?

Contoh terkait:

- Lokakarya Modernisasi Aplikasi Iteratif

Video terkait:

- Memberikan Keunggulan dengan Layanan Mikro di AWS

REL03-BP02 Bangun layanan yang berfokus pada domain dan fungsionalitas bisnis khusus

Arsitektur berorientasi layanan (SOA) menetapkan layanan dengan fungsi yang digambarkan dengan baik berdasarkan kebutuhan bisnis. Layanan mikro menggunakan model domain dan konteks yang dibatasi untuk menarik batas-batas layanan di sepanjang batas konteks bisnis. Berfokus pada domain dan fungsionalitas bisnis dapat membantu tim untuk menentukan persyaratan keandalan sendiri untuk layanan mereka. Konteks yang dibatasi mengisolasi dan memisahkan logika bisnis, sehingga memungkinkan tim memiliki penalaran yang lebih baik tentang bagaimana menangani kegagalan.

Hasil yang diinginkan: Rekayasawan dan pemangku kepentingan bisnis bersama-sama menetapkan konteks yang dibatasi dan menggunakannya untuk merancang sistem sebagai layanan yang memenuhi fungsi bisnis tertentu. Tim-tim ini menggunakan praktik yang telah lazim seperti event storming untuk menentukan persyaratan. Aplikasi baru dirancang sebagai batas layanan yang ditetapkan dengan baik dan penggabungan longgar. Monolit yang ada diurai menjadi konteks-konteks yang dibatasi dan desain sistem beralih ke arsitektur SOA atau layanan mikro. Ketika monolit difaktorkan ulang, pendekatan lazim seperti konteks gelembung dan pola penguraian monolit diterapkan.

Layanan berorientasi domain dijalankan sebagai satu atau beberapa proses yang statusnya tidak sama. Layanan-layanan tersebut secara independen merespons fluktuasi permintaan dan menangani skenario kesalahan dengan berpatokan pada persyaratan khusus domain.

Antipola umum:

- Tim dibentuk berdasarkan domain-domain teknis tertentu seperti UI dan UX, perangkat lunak perantara (middleware), atau basis data, bukan berdasarkan domain bisnis tertentu.
- Aplikasi melibatkan tanggung jawab domain. Layanan yang mencakup konteks yang dibatasi bisa lebih sulit untuk dipelihara, memerlukan upaya pengujian yang lebih besar, dan memerlukan banyak tim domain untuk berpartisipasi dalam pembaruan perangkat lunak.
- Dependensi domain, seperti pustaka entitas domain, dibagikan di seluruh layanan sehingga perubahan untuk satu domain layanan memerlukan perubahan pada domain layanan lainnya

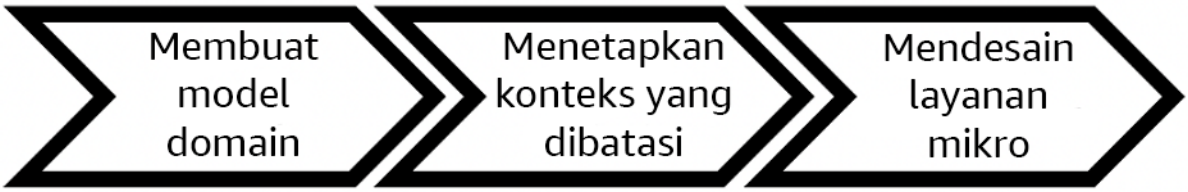
- Kontrak layanan dan logika bisnis tidak mengekspresikan entitas dalam bahasa domain yang umum dan konsisten, sehingga menghasilkan lapisan terjemahan yang merumitkan sistem dan meningkatkan upaya debugging.

Manfaat menjalankan praktik terbaik ini: Aplikasi dirancang sebagai layanan independen yang dibatasi oleh domain bisnis dan menggunakan bahasa bisnis umum. Layanan dapat diuji dan dapat di-deploy secara independen. Layanan memenuhi persyaratan ketahanan khusus domain untuk domain yang diterapkan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Keputusan berbasis domain (DDD) adalah pendekatan dasar perancangan dan pembangunan perangkat lunak berdasarkan domain bisnis. Bekerja dengan kerangka kerja yang ada memudahkan pembuatan layanan yang berfokus pada domain bisnis. Saat bekerja dengan aplikasi monolitik yang ada, Anda dapat memanfaatkan pola penguraian yang menyediakan teknik-teknik yang sudah lazim untuk memodernisasi aplikasi menjadi layanan.



Keputusan berbasis domain

Langkah implementasi

- Tim dapat menyelenggarakan lokakarya event storming untuk mengidentifikasi peristiwa, perintah, agregat, dan domain secara cepat dalam format catatan tempel ringan.
- Setelah entitas dan fungsi domain dibentuk dalam konteks domain, Anda dapat membagi domain Anda ke dalam layanan-layanan menggunakan konteks yang dibatasi, dengan mengelompokkan entitas dengan fitur dan atribut yang serupa. Dengan model yang dibagi ke dalam konteks, muncul templat untuk membatasi layanan mikro.
  - Misalnya, entitas situs web Amazon.com dapat meliputi paket, pengantaran, jadwal, harga, diskon, dan mata uang.

    - Paket, pengantaran, dan jadwal dikelompokkan ke dalam konteks pengiriman, sedangkan harga, diskon, dan mata uang dikelompokkan ke dalam konteks harga.
- Mengurai monolit menjadi layanan mikro menjelaskan pola-pola untuk pemfaktoran ulang layanan mikro. Menggunakan pola-pola penguraian berdasarkan kemampuan bisnis, subdomain, atau transaksi selaras dengan pendekatan berbasis domain.
- Teknik-teknik taktis seperti konteks gelembung memungkinkan Anda memasukkan DDD di dalam aplikasi yang ada atau aplikasi warisan tanpa penulisan ulang di awal dan komitmen penuh terhadap DDD. Dalam pendekatan konteks gelembung, konteks terbatas yang kecil dibuat menggunakan pemetaan layanan dan koordinasi, atau lapisan antikorupsi, yang melindungi model domain yang baru ditentukan dari pengaruh eksternal.

Setelah tim melakukan analisis domain dan menentukan entitas serta kontrak layanan, mereka dapat memanfaatkan layanan AWS untuk menerapkan desain berbasis domain mereka sebagai layanan berbasis cloud.

- Mulai pengembangan Anda dengan menentukan pengujian yang menggunakan aturan bisnis domain Anda. Pengembangan berbasis pengujian (TDD) dan pengembangan berbasis perilaku (BDD) membantu tim menjaga layanan tetap fokus pada pemecahan masalah bisnis.
- Pilih layanan AWS yang paling memenuhi persyaratan domain bisnis dan arsitektur layanan mikro Anda:
    - AWS Nirserver memungkinkan tim Anda untuk fokus pada logika domain tertentu, bukan pada pengelolaan server dan infrastruktur.
    - Kontainer di AWS menyederhanakan pengelolaan infrastruktur Anda, sehingga Anda dapat fokus pada persyaratan domain Anda.
    - Basis data yang dirancang khusus membantu Anda mencocokkan persyaratan domain Anda dengan jenis basis data yang paling sesuai.
- Membangun arsitektur heksagonal di AWS menguraikan kerangka kerja untuk membangun logika bisnis menjadi layanan yang bekerja mundur dari domain bisnis untuk memenuhi persyaratan fungsional dan kemudian melampirkan adaptor integrasi. Pola-pola yang memisahkan detail antarmuka dari logika bisnis dengan layanan AWS membantu tim untuk berfokus pada fungsionalitas domain dan meningkatkan kualitas perangkat lunak.

Sumber daya

Praktik terbaik terkait:

- REL03-BP01 Memilih cara untuk menyegmentasi beban kerja
- REL03-BP03 Memberikan kontrak layanan per API

Dokumen terkait:

- Layanan Mikro AWS
- Mengimplementasikan Layanan Mikro di AWS
- Cara memecah Monolit menjadi Layanan-Layanan Mikro
- Mulai Menggunakan DDD di Tengah-Tengah Sistem Warisan
- Desain Berbasis Domain: Mengatasi Kompleksitas di Dalam Inti Perangkat Lunak
- Membangun arsitektur heksagonal di AWS
- Mengurai monolit menjadi layanan mikro
- Event Storming
- Pesan Antara Konteks-Konteks yang Dibatasi
- Layanan mikro
- Pengembangan berbasis pengujian
- Pengembangan berbasis perilaku

Contoh terkait:

- Lokakarya Cloud-Native Korporat
- Merancang Layanan Mikro Cloud-Native di AWS (dari DDD/EventStormingWorkshop)

Alat terkait:

- Basis Data AWS Cloud
- Nirserver di AWS
- Kontainer di AWS

REL03-BP03 Memberikan kontrak layanan per API

Kontrak layanan adalah perjanjian terdokumentasi antara produsen dan konsumen API yang ditetapkan dalam definisi API yang dapat dibaca mesin. Strategi versioning kontrak memungkinkan

konsumen untuk terus menggunakan API yang ada dan memigrasikan aplikasi mereka ke API yang lebih baru ketika mereka siap. Deployment produsen dapat terjadi kapan saja, selama kontrak dipatuhi. Tim layanan dapat menggunakan tumpukan teknologi pilihan mereka untuk memenuhi kontrak API.

Hasil yang diinginkan:

Antipola umum: Aplikasi yang dibangun dengan arsitektur berorientasi layanan atau layanan mikro dapat beroperasi secara independen sementara tetap memiliki dependensi runtime yang terintegrasi. Perubahan yang di-deploy ke konsumen atau produsen API tidak mengganggu stabilitas sistem secara keseluruhan ketika kedua belah pihak mematuhi kontrak API yang sama. Komponen yang berkomunikasi melalui API layanan dapat melakukan rilis fungsional independen, peningkatan ke dependensi runtime, atau melakukan failover ke situs pemulihan bencana (DR) dengan sedikit atau tanpa dampak terhadap satu sama lain. Selain itu, layanan diskret dapat menyesuaikan skala secara independen dengan menyerap permintaan sumber daya tanpa mengharuskan layanan lain untuk menyesuaikan skala secara serempak.

- Membuat API layanan tanpa skema strongly-typed. Hal ini menghasilkan API yang tidak dapat digunakan untuk menghasilkan pengikatan dan muatan API yang tidak dapat divalidasi secara terprogram.
- Tidak mengadopsi strategi versioning, yang memaksa konsumen API untuk memperbarui dan melepaskan atau gagal saat kontrak layanan berkembang.
- Pesan kesalahan yang membocorkan detail implementasi layanan yang mendasari, bukan menggambarkan kegagalan integrasi dalam bahasa dan konteks domain.
- Tidak menggunakan kontrak API untuk mengembangkan kasus pengujian dan implementasi API simulasi untuk memungkinkan pengujian komponen layanan secara independen.

Manfaat menjalankan praktik terbaik ini: Sistem terdistribusi yang terdiri dari komponen-komponen yang berkomunikasi melalui kontrak layanan API dapat meningkatkan keandalan. Developer dapat mengidentifikasi potensi masalah di awal proses pengembangan dengan pemeriksaan tipe selama kompilasi untuk memverifikasi bahwa bidang-bidang yang diperlukan ada dan permintaan serta respons mematuhi kontrak API. Kontrak API menyediakan antarmuka dokumentasi mandiri yang jelas untuk API dan menyediakan interoperabilitas yang lebih baik antara sistem dan bahasa pemrograman yang berbeda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Setelah mengidentifikasi domain bisnis dan menentukan segmentasi beban kerja, Anda dapat mengembangkan API layanan. Pertama-tama, tentukan kontrak layanan yang dapat dibaca mesin untuk API, lalu implementasikan strategi versioning API. Setelah siap mengintegrasikan layanan melalui protokol umum seperti REST, GraphQL, atau peristiwa asinkron, Anda dapat memasukkan layanan AWS ke dalam arsitektur Anda untuk mengintegrasikan komponen Anda dengan kontrak API strongly-typed.

Layanan AWS untuk kontrak API layanan

Sertakan layanan AWS seperti Amazon API Gateway, AWS AppSync, dan Amazon EventBridge ke dalam arsitektur Anda untuk menggunakan kontrak layanan API dalam aplikasi Anda. Amazon API Gateway membantu Anda terintegrasi dengan layanan AWS native langsung serta layanan web lainnya. API Gateway mendukung spesifikasi dan versioning OpenAPI. AWS AppSync adalah titik akhir GraphQL terkelola yang Anda konfigurasikan dengan menetapkan skema GraphQL untuk menetapkan antarmuka layanan untuk kueri, mutasi, dan langganan. Amazon EventBridge menggunakan skema peristiwa untuk menetapkan peristiwa dan menghasilkan kode binding untuk peristiwa Anda.

Langkah implementasi

- Pertama-tama, tetapkan kontrak untuk API Anda. Kontrak akan mengekspresikan kemampuan suatu API serta menetapkan objek dan bidang data strongly-typed untuk input dan output API.
- Saat mengonfigurasi API di API Gateway, Anda dapat mengimpor dan mengekspor OpenAPI Specification untuk titik akhir Anda.
  - Mengimpor definisi OpenAPI menyederhanakan pembuatan API Anda dan dapat diintegrasikan dengan infrastruktur AWS sebagai alat kode seperti AWS Serverless Application Model dan AWS Cloud Development Kit (AWS CDK).
  - Mengekspor definisi API menyederhanakan integrasi dengan alat pengujian API dan menyediakan spesifikasi integrasi untuk konsumen layanan.
- Anda dapat menetapkan dan mengelola API GraphQL dengan AWS AppSync dengan cara menetapkan file skema GraphQL untuk menghasilkan antarmuka kontrak Anda dan menyederhanakan interaksi dengan model REST kompleks, beberapa tabel basis data, atau layanan warisan.
- Proyek AWS Amplify yang terintegrasi dengan AWS AppSync menghasilkan file kueri JavaScript strongly-typed untuk digunakan dalam aplikasi Anda serta pustaka klien GraphQL AWS AppSync untuk tabel Amazon DynamoDB .

- Saat Anda menggunakan peristiwa layanan dari Amazon EventBridge, peristiwa mematuhi skema yang sudah ada di dalam registri skema atau yang Anda definisikan dengan OpenAPI Spec. Dengan skema yang didefinisikan dalam registri, Anda juga dapat menghasilkan binding klien dari kontrak skema untuk mengintegrasikan kode Anda dengan peristiwa.
- Memperluas atau melakukan versioning API Anda. Memperluas API adalah opsi yang lebih sederhana saat menambahkan bidang yang dapat dikonfigurasi dengan bidang opsional atau nilai default untuk bidang wajib.
  - Kontrak berbasis JSON untuk protokol seperti REST dan GraphQL bisa ideal untuk perluasan kontrak.
  - Kontrak berbasis XML untuk protokol seperti SOAP harus diuji dengan konsumen layanan untuk menentukan kelayakan perluasan kontrak.
- Saat melakukan versioning API, pertimbangkan implementasi versioning proksi yang menggunakan facade untuk mendukung versi sehingga logika dapat dipertahankan dalam satu basis kode.
  - Dengan API Gateway Anda dapat menggunakan pemetaan permintaan dan respons untuk menyederhanakan penyerapan perubahan kontrak dengan membuat facade untuk memberikan nilai default untuk bidang baru atau untuk membuang bidang yang dihapus dari permintaan atau respons. Dengan pendekatan ini, layanan yang mendasari dapat mempertahankan basis kode tunggal.

Sumber daya

Praktik terbaik terkait:

- REL03-BP01 Memilih cara untuk menyegmentasi beban kerja
- REL03-BP02 Bangun layanan yang berfokus pada domain dan fungsionalitas bisnis khusus
- REL04-BP02 Mengimplementasikan dependensi yang digabungkan secara longgar
- REL05-BP03 Mengontrol dan membatasi panggilan percobaan ulang
- REL05-BP05 Mengatur batas waktu klien

Dokumen terkait:

- Apa itu API (Antarmuka Pemrograman Aplikasi)?
- Mengimplementasikan Layanan Mikro di AWS
- Kompensasi Layanan Mikro

- Layanan mikro - definisi istilah arsitektur baru ini
- Layanan mikro di AWS
- Bekerja dengan ekstensi API Gateway ke OpenAPI
- OpenAPI-Specification
- GraphQL: Skema dan Jenis
- Binding kode Amazon EventBridge

Contoh terkait:

- Amazon API Gateway: Mengonfigurasi API REST Menggunakan OpenAPI
- Aplikasi CRUD Amazon API Gateway ke Amazon DynamoDB menggunakan OpenAPI
- Pola integrasi aplikasi modern di era nirserver: Integrasi Layanan API Gateway
- Mengimplementasikan versioning API Gateway berbasis header dengan Amazon CloudFront
- AWS AppSync: Membangun aplikasi klien

Video terkait:

- Menggunakan OpenAPI di AWS SAM untuk mengelola API Gateway

Alat terkait:

- Amazon API Gateway
- AWS AppSync
- Amazon EventBridge

REL 4. Bagaimana cara mendesain interaksi di sistem terdistribusi untuk mencegah kegagalan?

Sistem terdistribusi mengandalkan jaringan komunikasi untuk membuat interkoneksi komponen, seperti server atau layanan. Beban kerja Anda harus beroperasi secara andal terlepas latensi atau hilangnya data di jaringan-jaringan ini. Komponen dari sistem terdistribusi harus beroperasi dengan cara yang tidak secara negatif memengaruhi beban kerja atau komponen-komponen lain. Berbagai praktik terbaik ini mencegah kegagalan dan meningkatkan waktu rata-rata antara kegagalan (MTBF).

Praktik terbaik

- REL04-BP01 Mengidentifikasi jenis sistem terdistribusi yang diperlukan
- REL04-BP02 Mengimplementasikan dependensi yang digabungkan secara longgar
- REL04-BP03 Melakukan tugas konstan
- REL04-BP04 Menjadikan semua respons idempoten

REL04-BP01 Mengidentifikasi jenis sistem terdistribusi yang diperlukan

Sistem terdistribusi hard real-time memerlukan respons yang diberikan secara sinkron dan cepat, sedangkan sistem soft real-time memiliki jendela waktu yang lebih fleksibel untuk respons, dalam hitungan menit atau lebih. Sistem offline menangani respons melalui batch atau pemrosesan asinkron. Sistem terdistribusi hard real-time memiliki persyaratan keandalan yang paling ketat.

Tantangan yang paling sulit dengan sistem terdistribusi adalah sistem terdistribusi hard real-time, yang dikenal juga sebagai layanan permintaan/balasan. Hal yang membuatnya sulit adalah permintaan yang masuk tidak dapat diprediksikan dan respons yang diberikan harus cepat (misalnya, pelanggan menunggu respons dengan aktif). Contohnya mencakup server web front-end, urutan pipeline, transaksi kartu kredit, setiap API AWS, dan telefoni.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Identifikasikan jenis sistem terdistribusi yang diperlukan. Tantangan dengan sistem terdistribusi meliputi latensi, penskalaan, pemahaman atas API jaringan, mengonversi dan membatalkan konversi data, serta kompleksitas algoritme seperti Paxos. Ketika sistem tumbuh lebih besar dan lebih terdistribusi, apa yang tadinya merupakan kasus edge teoretis berubah menjadi kejadian biasa.
  - Amazon Builders' Library: Tantangan dengan sistem terdistribusi
    - Sistem terdistribusi hard real-time memerlukan respons yang diberikan secara sinkron dan cepat.
    - Sistem soft real-time memiliki jendela waktu yang lebih fleksibel untuk respons, dalam hitungan menit atau lebih.
    - Sistem offline menangani respons melalui batch atau pemrosesan asinkron.
    - Sistem terdistribusi hard real-time memiliki persyaratan keandalan yang paling ketat.

Sumber daya

Dokumen terkait:

- Amazon EC2: Memastikan Idempotensi
- Amazon Builders' Library: Tantangan dengan sistem terdistribusi
- Amazon Builders' Library: Keandalan, kerja konstan, dan pilihan yang tepat
- Apa Itu Amazon EventBridge?
- Apa Itu Amazon Simple Queue Service?

Video terkait:

- AWS New York Summit 2019: Pengantar Arsitektur Berbasis Peristiwa dan Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: Cara Mengontrol Sistem, ARC337 Besar dan Kecil (mencakup penggabungan longgar, kerja konstan, stabilitas statis)
- AWS re:Invent 2019: Beralih ke arsitektur berbasis peristiwa (SVS308)

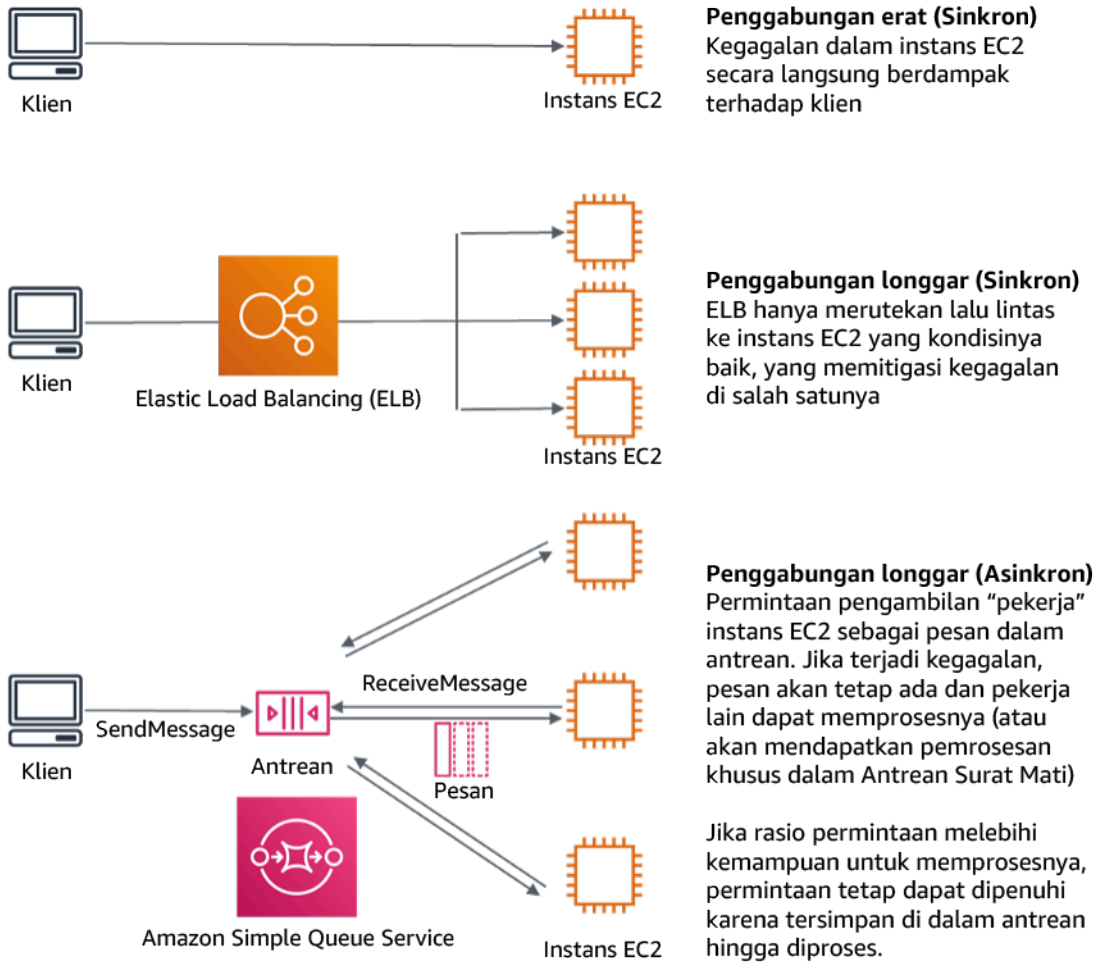
REL04-BP02 Mengimplementasikan dependensi yang digabungkan secara longgar

Dependensi seperti sistem pengantrean, sistem streaming, alur kerja, dan penyeimbang beban digabungkan secara longgar. Penggabungan longgar membantu memisahkan perilaku suatu komponen dari komponen lainnya yang bergantung pada komponen tersebut, sehingga meningkatkan ketahanan dan ketangkasan.

Dalam sistem penggabungan erat, perubahan pada satu komponen dapat menyebabkan perubahan pada komponen lain yang bergantung padanya, yang mengakibatkan penurunan performa di semua komponen. Penggabungan longgar menghilangkan dependensi ini sehingga komponen-komponen yang bergantung hanya perlu mengetahui antarmuka versi terbaru dan yang dipublikasikan. Implementasi penggabungan longgar antar dependensi memisahkan kegagalan pada salah satu dependensi agar tidak memengaruhi dependensi lain.

Penggabungan longgar memungkinkan Anda untuk mengubah kode atau menambahkan fitur ke sebuah komponen sambil meminimalkan risiko pada komponen lain yang bergantung pada komponen tersebut. Hal ini juga memungkinkan ketahanan granular pada tingkat komponen sehingga Anda dapat menskalakan ke luar atau bahkan mengubah implementasi yang mendasari dependensi.

Agar makin meningkatkan ketahanan melalui penggabungan longgar, jadikan interaksi komponen asinkron apabila memungkinkan. Model ini cocok untuk interaksi apa pun yang tidak memerlukan respons cepat dan ketika terdaftarnya suatu permintaan cukup perlu diketahui. Ini melibatkan satu komponen yang menghasilkan peristiwa dan komponen lain yang menggunakannya. Kedua komponen tersebut tidak terintegrasi melalui interaksi titik ke titik langsung, tetapi biasanya melalui lapisan penyimpanan tahan lama perantara, seperti antrean Amazon SQS atau platform data streaming seperti Amazon Kinesis, atau AWS Step Functions.



Gambar 4: Dependensi seperti sistem pengantrean dan penyeimbang beban digabungkan secara longgar.

Antrean Amazon SQS dan Penyeimbang Beban Elastis hanyalah dua cara untuk menambahkan lapisan perantara untuk penggabungan longgar. Arsitektur yang didorong peristiwa juga dapat dibangun di AWS Cloud menggunakan Amazon EventBridge, yang dapat mengabstraksi klien (penghasil peristiwa) dari layanan yang mereka andalkan (pemakai peristiwa). Amazon Simple Notification Service (Amazon SNS) adalah solusi efektif ketika Anda memerlukan olah pesan dari

banyak ke banyak dengan throughput tinggi dan berbasis push. Menggunakan topik Amazon SNS, sistem penerbit Anda dapat menyebarkan pesan ke titik akhir pelanggan dalam jumlah besar untuk pemrosesan paralel.

Meskipun antrean menawarkan sejumlah manfaat, di sebagian besar sistem waktu nyata yang keras, permintaan yang lebih lama dari waktu ambang batas (sering kali dalam hitungan detik) harus dianggap basi (klien telah menyerah dan sudah tidak menunggu respons), dan tidak diproses. Dengan begitu, permintaan yang lebih baru (dan kemungkinan masih valid) dapat diproses sebagai gantinya.

Hasil yang diinginkan: Menerapkan dependensi penggabungan longgar memungkinkan Anda untuk meminimalkan area kegagalan ke tingkat komponen, yang membantu mendiagnosis dan menyelesaikan masalah. Cara ini juga dapat menyederhanakan siklus pengembangan, sehingga memungkinkan tim untuk menerapkan perubahan pada tingkat modular tanpa memengaruhi performa komponen lain yang bergantung padanya. Pendekatan ini memberikan kemampuan untuk menskalakan ke luar pada tingkat komponen berdasarkan kebutuhan sumber daya, serta pemanfaatan komponen yang berkontribusi terhadap efektivitas biaya.

Antipola umum:

- Melakukan deployment beban kerja monolitik.
- Memanggil API antar tingkatan beban kerja secara langsung tanpa kemampuan failover atau pemrosesan permintaan secara asinkron.
- Penggabungan erat menggunakan data bersama. Sistem yang digabungkan secara longgar sebaiknya tidak berbagi data melalui basis data bersama atau bentuk penyimpanan data yang digabungkan secara erat, yang dapat menimbulkan kembali penggabungan erat dan menghambat skalabilitas.
- Mengabaikan tekanan balik. Beban kerja Anda harus memiliki kemampuan untuk memperlambat atau menghentikan data yang masuk ketika komponen tidak dapat memprosesnya pada kecepatan yang sama.

Manfaat menetapkan praktik terbaik ini: Penggabungan longgar membantu mengisolasi perilaku komponen dari komponen lain yang bergantung padanya, sehingga meningkatkan ketahanan dan ketangkasan. Kegagalan di salah satu komponen dipisahkan dari komponen lain.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Implementasikan dependensi yang digabungkan secara longgar. Ada berbagai solusi yang memungkinkan Anda membangun aplikasi yang digabungkan secara longgar. Ini meliputi, beberapa di antaranya, layanan untuk mengimplementasikan antrean yang dikelola sepenuhnya, alur kerja otomatis, reaksi terhadap peristiwa, dan API yang dapat membantu mengisolasi perilaku komponen dari komponen lain, dan dengan demikian meningkatkan ketahanan dan ketangkasan.

- Membangun arsitektur yang didorong peristiwa: Amazon EventBridge membantu Anda membangun arsitektur berbasis peristiwa yang digabungkan secara longgar dan terdistribusi.
- Menerapkan antrean dalam sistem terdistribusi: Anda dapat menggunakan Amazon Simple Queue Service (Amazon SQS) untuk mengintegrasikan dan memisahkan sistem terdistribusi.
- Kontainerisasi komponen sebagai layanan mikro: Layanan mikro memungkinkan tim untuk membangun aplikasi yang terdiri dari komponen independen kecil yang berkomunikasi melalui API yang ditentukan dengan jelas. Amazon Elastic Container Service (Amazon ECS), dan Amazon Elastic Kubernetes Service (Amazon EKS) dapat membantu Anda memulai lebih cepat dengan kontainer.
- Kelola alur kerja dengan Step Functions: Step Functions membantu Anda mengoordinasikan beberapa layanan AWS menjadi alur kerja yang fleksibel.
- Manfaatkan arsitektur olah pesan publikasi-berlangganan (pub/sub): Amazon Simple Notification Service (Amazon SNS) menyediakan pengiriman pesan dari penerbit ke pelanggan (juga dikenal sebagai produsen dan konsumen).

Langkah implementasi

- Komponen dalam arsitektur yang didorong peristiwa dimulai oleh peristiwa. Peristiwa adalah tindakan yang terjadi dalam sistem, seperti pengguna menambahkan item ke keranjang. Ketika suatu tindakan berhasil, sebuah peristiwa dihasilkan, yang menggerakkan komponen berikutnya dari sistem.
  - Building Event-driven Applications with Amazon EventBridge
  - AWS re:Invent 2022 - Designing Event-Driven Integrations using Amazon EventBridge
- Sistem olah pesan terdistribusi memiliki tiga bagian utama yang perlu diimplementasikan untuk arsitektur berbasis antrean. Bagian-bagian tersebut meliputi komponen sistem terdistribusi, antrean yang digunakan untuk pemisahan (didistribusikan di server Amazon SQS), dan pesan dalam antrean. Sistem tipikal memiliki produsen yang memulai pesan ke dalam antrean, dan konsumen

yang menerima pesan dari antrean tersebut. Antrean menyimpan pesan di beberapa server Amazon SQS untuk redundansi.

  - Basic Amazon SQS architecture
  - Send Messages Between Distributed Applications with Amazon Simple Queue Service
- Layanan mikro, jika dimanfaatkan dengan baik, akan meningkatkan pemeliharaan dan mendongkrak skalabilitas, karena komponen yang digabungkan secara longgar dikelola oleh tim independen. Hal ini juga memungkinkan isolasi perilaku ke satu komponen jika terjadi perubahan.
  - Mengimplementasikan Layanan Mikro di AWS
  - Let's Architect! Architecting microservices with containers
- Dengan AWS Step Functions Anda dapat membangun aplikasi terdistribusi, mengotomatiskan proses, mengorkestrasi layanan mikro, serta berbagai hal lainnya. Orkestrasi beberapa komponen ke dalam alur kerja otomatis memungkinkan Anda untuk memisahkan dependensi dalam aplikasi Anda.
  - Create a Serverless Workflow with AWS Step Functions and AWS Lambda
  - Mulai menggunakan AWS Step Functions

Sumber daya

Dokumen terkait:

- Amazon EC2: Ensuring Idempotency
- Amazon Builders' Library: Tantangan dengan sistem terdistribusi
- Amazon Builders' Library: Keandalan, tugas konstan, dan pilihan yang tepat
- Apa Itu Amazon EventBridge?
- Apa Itu Amazon Simple Queue Service?
- Break up with your monolith
- Orchestrate Queue-based Microservices with AWS Step Functions and Amazon SQS
- Basic Amazon SQS architecture
- Queue-Based Architecture

Video terkait:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)

- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)
- AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda (API304)
- AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda
- AWS re:Invent 2022 - Designing event-driven integrations using Amazon EventBridge
- AWS re:Invent 2017: Elastic Load Balancing Deep Dive and Best Practices

REL04-BP03 Melakukan tugas konstan

Sistem dapat gagal mengalami kegagalan saat ada perubahan besar dan cepat pada beban. Misalnya, jika beban kerja Anda sedang melakukan pemeriksaan kondisi yang memantau kondisi dari ribuan server, beban kerja Anda harus mengirimkan payload berukuran sama (snapshot penuh berisi status saat ini) setiap saat. Saat tidak ada server yang gagal, atau semuanya gagal, sistem pemeriksaan kondisi melakukan tugas konstan tanpa perubahan besar dan cepat.

Misalnya, jika sistem pemeriksaan kondisi sedang memantau 100.000 server, beban di dalamnya kecil, dengan tingkat kegagalan server normal yang ringan. Namun, jika sebuah peristiwa besar menjadikan separuh server tidak sehat, sistem pemeriksaan kondisi akan kewalahan untuk memperbarui sistem notifikasi dan menyampaikan status ke kliennya. Jadi sebagai gantinya, sistem pemeriksaan kondisi harus mengirimkan snapshot penuh berisi status saat ini setiap saat. 100.000 status sehat server, masing-masing diwakili satu bit, hanyalah satu payload berukuran 12,5 KB. Saat tidak ada server yang gagal, atau semuanya gagal, sistem pemeriksaan kondisi melakukan tugas konstan, dan perubahan yang besar dan cepat bukanlah ancaman untuk stabilitas sistem. Seperti inilah Amazon Route 53 menangani pemeriksaan kondisi untuk titik akhir (seperti alamat IP) untuk menentukan bagaimana pengguna akhir dirutekan ke sana.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Rendah

Panduan implementasi

- Lakukan tugas konstan sehingga sistem tidak gagal saat terdapat perubahan beban yang besar dan cepat.
- Implementasikan dependensi yang digabungkan secara longgar. Dependensi seperti sistem pengantrean, sistem streaming, alur kerja, dan penyeimbang beban digabungkan secara longgar.

Penggabungan longgar membantu memisahkan perilaku suatu komponen dari komponen lainnya yang bergantung pada komponen tersebut, sehingga meningkatkan ketahanan dan ketangkasan.

- Amazon Builders' Library: Keandalan, kerja konstan, dan secangkir kopi yang enak
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (mencakup tugas konstan)
  - Untuk contoh sistem pemeriksaan kondisi yang memantau 100.000 server, rekayasa beban kerja sehingga ukuran payload tetap sama berapa pun jumlah keberhasilan atau kegagalan.

Sumber daya

Dokumen terkait:

- Amazon EC2: Memastikan Idempotensi
- Amazon Builders' Library: Tantangan dengan sistem terdistribusi
- Amazon Builders' Library: Keandalan, kerja konstan, dan secangkir kopi yang enak

Video terkait:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (mencakup tugas konstan)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (mencakup penggabungan longgar, kerja konstan, stabilitas statis)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)

REL04-BP04 Menjadikan semua respons idempoten

Layanan idempoten menjanjikan setiap permintaan diselesaikan tepat satu kali, sehingga pembuatan beberapa permintaan yang sama memiliki efek yang sama seperti membuat satu permintaan. Layanan idempoten memudahkan klien untuk mengimplementasikan percobaan ulang tanpa takut permintaan akan salah diproses beberapa kali. Untuk melakukan ini, klien dapat mengeluarkan permintaan API dengan token idempotensi—token yang sama digunakan setiap permintaan diulang. API layanan idempoten menggunakan token untuk mengembalikan respons yang identik dengan respons yang dikembalikan saat pertama kali permintaan diselesaikan.

Dalam sistem terdistribusi, mudah untuk melakukan tindakan paling banyak satu kali (klien hanya membuat satu permintaan), atau setidaknya satu kali (tetap meminta sampai klien mendapat konfirmasi berhasil). Namun, sulit untuk menjamin suatu tindakan bersifat idempoten, yang berarti tindakan dilakukan tepat satu kali, sehingga pembuatan beberapa permintaan identik memiliki efek yang sama seperti membuat satu permintaan. Menggunakan token idempotensi di API, layanan dapat menerima permintaan yang bermutasi satu kali atau lebih tanpa membuat rekaman ganda atau efek samping.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Menjadikan semua respons idempoten. Layanan idempoten menjanjikan setiap permintaan diselesaikan tepat satu kali, sehingga pembuatan beberapa permintaan yang sama memiliki efek yang sama seperti membuat satu permintaan.
  - Klien dapat mengeluarkan permintaan API dengan token idempotensi—token yang sama digunakan setiap permintaan diulang. API layanan idempoten menggunakan token untuk mengembalikan respons yang identik dengan respons yang dikembalikan saat pertama kali permintaan diselesaikan.
    - Amazon EC2: Memastikan Idempotensi

Sumber daya

Dokumen terkait:

- Amazon EC2: Memastikan Idempotensi
- Amazon Builders' Library: Tantangan dengan sistem terdistribusi
- Amazon Builders' Library: Keandalan, kerja konstan, dan secangkir kopi yang enak

Video terkait:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (mencakup penggabungan longgar, kerja konstan, stabilitas statis)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)

REL 5. Bagaimana cara mendesain interaksi di sistem terdistribusi untuk memitigasi atau bertahan dari kegagalan?

Sistem terdistribusi mengandalkan jaringan komunikasi untuk membuat interkoneksi komponen (seperti server atau layanan). Beban kerja Anda harus beroperasi secara andal terlepas latensi atau hilangnya data di jaringan-jaringan ini. Komponen dari sistem terdistribusi harus beroperasi dengan cara yang tidak secara negatif memengaruhi beban kerja atau komponen-komponen lain. Berbagai praktik terbaik ini memungkinkan beban kerja bertahan dari stres atau kegagalan, lebih cepat pulih darinya, dan memitigasi dampak gangguan tersebut. Hasilnya yakni peningkatan dalam waktu rata-rata untuk pemulihan (MTTR).

Praktik terbaik

- REL05-BP01 Mengimplementasikan degradasi yang tepat (graceful degradation) untuk mengubah dependensi keras yang berlaku menjadi dependensi lunak
- REL05-BP02 Membatasi (throttling) permintaan
- REL05-BP03 Mengontrol dan membatasi panggilan percobaan ulang
- REL05-BP04 Melakukan gagal cepat (fail fast) dan membatasi antrean
- REL05-BP05 Mengatur batas waktu klien
- REL05-BP06 Menjadikan layanan stateless jika memungkinkan
- REL05-BP07 Mengimplementasikan tuas darurat

REL05-BP01 Mengimplementasikan degradasi yang tepat (graceful degradation) untuk mengubah dependensi keras yang berlaku menjadi dependensi lunak

Komponen aplikasi harus terus menjalankan fungsi intinya bahkan jika dependensi menjadi tidak tersedia. Komponen mungkin menyajikan data yang sedikit basi, data alternatif, atau bahkan tidak menyajikan data sama sekali. Hal ini memastikan fungsi sistem secara keseluruhan hanya terhambat secara minimum oleh kegagalan lokal sekaligus memberikan nilai bisnis utama.

Hasil yang diinginkan: Saat dependensi komponen tidak optimum, komponen tersebut masih dapat berfungsi, meskipun terbatas atau terdegradasi. Mode-mode kegagalan komponen harus dipandang sebagai operasi normal. Alur kerja harus dirancang sedemikian rupa sehingga kegagalan tersebut tidak menyebabkan kegagalan total atau setidaknya hanya menyebabkan keadaan yang dapat diprediksi dan dapat dipulihkan.

Antipola umum:

- Tidak mengidentifikasi fungsi bisnis inti yang dibutuhkan. Tidak menguji bahwa komponen berfungsi bahkan selama kegagalan dependensi.
- Tidak menyajikan data jika terjadi kesalahan atau ketika hanya ada satu dari beberapa dependensi yang tidak tersedia dan hasil parsial masih dapat dikembalikan.
- Menciptakan keadaan yang tidak konsisten ketika transaksi gagal sebagian.
- Tidak memiliki cara alternatif untuk mengakses tempat penyimpanan parameter pusat.
- Membatalkan atau mengosongkan status lokal sebagai akibat dari penyegaran yang gagal tanpa mempertimbangkan konsekuensi tindakan tersebut.

Manfaat menjalankan praktik terbaik ini: Degradasi bertahap (graceful degradation) meningkatkan ketersediaan sistem secara keseluruhan dan mempertahankan fungsionalitas fungsi-fungsi yang paling penting, bahkan selama kegagalan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Menerapkan degradasi yang tepat membantu meminimalkan dampak kegagalan dependensi pada fungsi komponen. Idealnya, sebuah komponen mendeteksi kegagalan dependensi dan menanganinya dengan cara yang berdampak minim pada pelanggan atau komponen lain.

Merancang untuk degradasi yang tepat berarti mempertimbangkan potensi mode kegagalan selama desain dependensi. Untuk setiap mode kegagalan, miliki cara untuk menghadirkan sebagian besar atau setidaknya fungsionalitas paling penting dari komponen kepada pemanggil atau pelanggan. Pertimbangan ini dapat menjadi persyaratan tambahan yang dapat diuji dan diverifikasi. Idealnya, sebuah komponen mampu menjalankan fungsi intinya dengan cara yang dapat diterima bahkan ketika satu atau beberapa dependensi gagal.

Ini bukan hanya pembahasan teknis, melainkan juga pembahasan bisnis. Semua persyaratan bisnis penting dan harus dipenuhi jika memungkinkan. Namun, menanyakan apa yang seharusnya terjadi ketika tidak semua persyaratan tersebut dapat dipenuhi adalah hal yang wajar. Suatu sistem dapat dirancang agar tersedia dan konsisten, tetapi dalam keadaan yang mengharuskan satu persyaratan untuk dikorbankan, mana yang lebih penting? Untuk pemrosesan pembayaran, jawabannya mungkin adalah konsistensi. Untuk aplikasi waktu nyata, jawabannya mungkin adalah ketersediaan. Untuk situs web yang digunakan langsung oleh pelanggan, jawabannya mungkin tergantung pada ekspektasi pelanggan.

Seberapa pentingnya, ini tergantung persyaratan komponen dan apa yang seharusnya dianggap sebagai fungsi intinya. Misalnya:

- Situs web ecommerce mungkin menampilkan data dari berbagai sistem seperti rekomendasi yang dipersonalisasi, produk dengan peringkat tertinggi, dan status pesanan pelanggan di halaman arahan. Ketika salah satu sistem hulu gagal, masih masuk akal untuk menampilkan semua daripada halaman kesalahan kepada pelanggan.
- Sebuah komponen yang menjalankan penulisan batch masih dapat melanjutkan pemrosesan batch jika salah satu operasi gagal. Implementasi mekanisme percobaan ulang harus sederhana. Hal ini dapat dilakukan dengan mengembalikan informasi tentang operasi yang berhasil, yang telah gagal, dan mengapa operasi tersebut gagal ke pemanggil, atau dengan menempatkan permintaan yang gagal ke dalam antrean surat mati untuk mengimplementasikan percobaan ulang asinkron. Informasi tentang operasi yang gagal juga harus dibuat log.
- Sistem yang memproses transaksi harus memverifikasi bahwa semua pembaruan individual dijalankan atau tidak sama sekali. Untuk transaksi terdistribusi, pola saga dapat digunakan untuk kembali ke operasi sebelumnya jika operasi selanjutnya dari transaksi yang sama gagal. Di sini, fungsi intinya adalah menjaga konsistensi.
- Sistem-sistem time-critical harus mampu menangani dependensi yang tidak merespons secara tepat waktu. Dalam kasus-kasus ini, pola pemutus sirkuit dapat digunakan. Ketika respons dari dependensi mulai mencapai batas waktu, sistem dapat beralih ke keadaan ditutup di mana tidak ada panggilan tambahan yang dibuat.
- Sebuah aplikasi dapat membaca parameter dari tempat penyimpanan parameter. Membuat image kontainer dengan serangkaian parameter default akan membantu agar apabila tempat penyimpanan parameter tidak tersedia image tersebut dapat digunakan.

Perhatikan bahwa jalur yang diambil jika terjadi kegagalan komponen perlu diuji dan harus jauh lebih sederhana daripada jalur utama. Umumnya, strategi fallback harus dihindari.

Langkah implementasi

Identifikasi dependensi eksternal dan internal. Pertimbangkan jenis-jenis kegagalan yang bisa terjadi di dalamnya. Pikirkan tentang cara-cara yang meminimalkan dampak negatif pada pelanggan serta sistem hulu dan hilir selama kegagalan-kegagalan tersebut.

Berikut ini adalah daftar dependensi dan cara melakukan degradasi yang tepat ketika dependensi gagal:

1. Kegagalan dependensi sebagian: Sebuah komponen dapat melakukan beberapa permintaan ke sistem-sistem hilir, baik beberapa permintaan ke satu sistem atau satu permintaan ke beberapa sistem. Tergantung konteks bisnis, mungkin ada berbagai cara penanganan yang sesuai (untuk detail lebih lanjut, lihat contoh-contoh sebelumnya dalam Panduan implementasi).
2. Sistem hilir tidak dapat memproses permintaan karena beban tinggi: Jika permintaan ke sistem hilir terus-menerus gagal, percobaan ulang tidak perlu dilanjutkan. Tindakan ini dapat menciptakan beban tambahan pada sistem yang sudah kelebihan beban dan mempersulit pemulihan. Pola pemutus sirkuit dapat digunakan di sini, yang memantau kegagalan panggilan ke sistem hilir. Jika ada banyak panggilan yang gagal, permintaan akan berhenti dikirimkan ke sistem hilir dan hanya sesekali panggilan dibiarkan masuk untuk menguji apakah sistem hilir sudah tersedia kembali.
3. Tempat penyimpanan parameter tidak tersedia: Untuk mengubah tempat penyimpanan parameter, caching dependensi lunak atau sane default yang disertakan di dalam image kontainer atau mesin dapat digunakan. Perhatikan bahwa default ini harus selalu diperbarui dan disertakan dalam rangkaian pengujian.
4. Layanan pemantauan atau dependensi non-fungsional lainnya tidak tersedia: Jika sebuah komponen sebentar-sebentar tidak dapat mengirim log, metrik, atau jejak ke layanan pemantauan pusat, langkah terbaiknya sering kali adalah tetap menjalankan fungsi-fungsi bisnis seperti biasa. Diam-diam tidak membuat log atau mendorong metrik dalam waktu yang lama sering kali tidak dapat diterima. Selain itu, beberapa kasus penggunaan mungkin memerlukan entri audit lengkap untuk memenuhi persyaratan kepatuhan.
5. Sebuah instans utama dari basis data relasional mungkin tidak tersedia: Amazon Relational Database Service, seperti hampir semua basis data relasional, hanya dapat memiliki satu instans penulis utama. Hal ini menciptakan satu titik kegagalan untuk beban kerja tulis dan menjadikan penskalaan lebih sulit. Hal ini dapat diatasi sebagiannya dengan menggunakan konfigurasi Multi-AZ untuk ketersediaan tinggi atau Nirserver Amazon Aurora untuk penskalaan yang lebih baik. Untuk persyaratan ketersediaan yang sangat tinggi, ada baiknya untuk tidak bergantung pada penulis utama sama sekali. Untuk kueri yang hanya membaca, replika baca dapat digunakan, yang memberikan redundansi dan kemampuan untuk melakukan scale-out, bukan hanya scale-up. Tulis dapat di-buffer, misalnya dalam antrean Amazon Simple Queue Service, sehingga permintaan tulis dari pelanggan masih dapat diterima bahkan jika penulis utama tidak tersedia untuk sementara.

Sumber daya

Dokumen terkait:

- Amazon API Gateway: Membatasi Permintaan API untuk Peningkatan Throughput
- CircuitBreaker (rangkuman Pemutus Sirkuit dari buku “Release It!”)
- Kesalahan Percobaan Ulang dan Mundur Eksponensial di AWS
- Michael Nygard “Release It! Design and Deploy Production-Ready Software” (Rancang dan Lakukan Deployment Perangkat Lunak yang Siap Diproduksi)
- Pustaka Pengembang Amazon: Menghindari fallback dalam sistem terdistribusi
- Pustaka Pengembang Amazon: Menghindari backlog antrean yang tidak dapat diatasi
- Pustaka Pengembang Amazon: Tantangan dan strategi caching
- Pustaka Pengembang Amazon: Batas waktu, percobaan ulang, dan mundur (backoff) dengan jitter

Video terkait:

- Percobaan ulang, mundur, dan jitter: AWS re:Invent 2019: Memperkenalkan Pustaka Pengembang Amazon (DOP328)

Contoh terkait:

- Lab Well-Architected: Level 300: Mengimplementasikan Pemeriksaan Kondisi dan Mengelola Dependensi untuk Meningkatkan Keandalan

REL05-BP02 Membatasi (throttling) permintaan

Batasi permintaan untuk mengurangi keletihan sumber daya karena peningkatan permintaan yang tidak terduga. Permintaan di bawah tingkat throttling akan diproses, sedangkan permintaan di atas batas yang ditentukan akan ditolak dengan memunculkan pesan bahwa permintaan telah dibatasi.

Hasil yang diinginkan: Lonjakan volume besar baik dari peningkatan lalu lintas pelanggan yang tiba-tiba, serangan membanjir, atau banjir percobaan ulang akan diminimalkan dengan throttling permintaan, sehingga beban kerja dapat melanjutkan pemrosesan volume permintaan normal yang didukung.

Antipola umum:

- Throttle titik akhir API tidak diimplementasikan atau dibiarkan pada nilai default tanpa mempertimbangkan volume yang diharapkan.
- Titik akhir API tidak diberi uji beban atau batas throttling tidak diuji.

- Membatasi angka permintaan tanpa mempertimbangkan ukuran atau kompleksitas permintaan.
- Menguji laju permintaan maksimum atau ukuran permintaan maksimum, tetapi tidak menguji keduanya bersama-sama.
- Sumber daya tidak disediakan untuk batas yang sama yang ditetapkan dalam pengujian.
- Rencana penggunaan belum dikonfigurasi atau dipertimbangkan untuk konsumen API aplikasi ke aplikasi (A2A).
- Tidak ada konfigurasi pengaturan konkurensi maksimum pada konsumen antrean yang diskalakan secara horizontal.
- Pembatasan tingkat per alamat IP belum diimplementasikan.

Manfaat menjalankan praktik terbaik ini: Beban kerja yang menetapkan batas throttle dapat beroperasi secara normal dan berhasil memproses beban permintaan yang diterima selama lonjakan volume yang tidak terduga. Lonjakan permintaan yang tiba-tiba atau terus menerus pada API dan antrean dibatasi dan tidak menghabiskan sumber daya pemrosesan permintaan. Batas angka permintaan membatasi setiap peminta sehingga volume lalu lintas yang tinggi dari satu alamat IP atau konsumen API tidak akan menghabiskan sumber daya atau berimbas pada konsumen lain.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Layanan harus dirancang untuk memproses kapasitas permintaan yang diketahui; kapasitas ini dapat ditetapkan melalui pengujian beban. Jika laju kedatangan permintaan melampaui batas, respons yang tepat menandakan bahwa permintaan telah dibatasi. Hal ini memungkinkan konsumen untuk menangani kesalahan dan mencoba ulang di lain waktu.

Saat layanan Anda memerlukan implementasi throttling, pertimbangkan mengimplementasikan algoritme bucket token, yang menghitung satu token sebagai satu permintaan. Token diisi ulang dengan laju throttle per detik dan dikosongkan secara asinkron oleh satu token per permintaan.

Algoritme bucket token.

Amazon API Gateway mengimplementasikan algoritme bucket token sesuai dengan batas yang dimiliki akun dan wilayah dan dapat dikonfigurasi per klien dengan rencana penggunaan. Selain itu, Amazon Simple Queue Service (Amazon SQS) dan Amazon Kinesis dapat melakukan buffer permintaan untuk memuluskan laju permintaan, dan memungkinkan tingkat throttling yang lebih tinggi untuk permintaan yang dapat diatasi. Terakhir, Anda dapat menerapkan pembatasan laju dengan AWS WAF untuk membatasi konsumen API tertentu yang menghasilkan beban yang terlalu tinggi.

Langkah implementasi

Anda dapat mengonfigurasi API Gateway dengan batas throttling untuk API dan mengembalikan pesan kesalahan `429 Terlalu Banyak Permintaan` ketika batas terlampaui. Anda dapat menggunakan AWS WAF dengan titik akhir AWS AppSync dan API Gateway Anda untuk mengaktifkan pembatasan laju per alamat IP. Selain itu, apabila sistem Anda dapat mentoleransi pemrosesan asinkron, Anda dapat memasukkan pesan ke dalam antrean atau aliran guna mempercepat respons terhadap klien layanan, yang memungkinkan Anda melakukan lonjakan ke tingkat throttle yang lebih tinggi.

Dengan pemrosesan asinkron, ketika Anda telah mengonfigurasi Amazon SQS sebagai sumber peristiwa untuk AWS Lambda, Anda dapat mengonfigurasi konkurensi maksimum untuk mencegah angka peristiwa yang tinggi memakai kuota eksekusi serentak akun yang tersedia yang diperlukan untuk layanan lain dalam beban kerja atau akun Anda.

Meskipun API Gateway menyediakan implementasi bucket token yang dikelola, apabila Anda tidak dapat menggunakan API Gateway, Anda dapat memanfaatkan implementasi sumber terbuka bahasa khusus (lihat contoh terkait di Sumber Daya) bucket token untuk layanan Anda.

- Pahami dan konfigurasi batas throttling API Gateway di tingkat akun per wilayah, API per tahap, dan kunci API per tingkat paket penggunaan.
- Terapkan aturan pembatas laju AWS WAF ke titik akhir API Gateway dan AWS AppSync untuk melindungi dari permintaan yang membanjir dan memblokir IP berbahaya. Aturan pembatas laju juga dapat dikonfigurasi pada kunci API AWS AppSync untuk konsumen A2A.
- Pertimbangkan apakah Anda memerlukan kontrol throttling yang lebih besar daripada pembatasan laju untuk API AWS AppSync, dan jika demikian, konfigurasikan API Gateway di depan titik akhir AWS AppSync Anda.
- Ketika antrean Amazon SQS diatur sebagai pemicu untuk konsumen antrean Lambda, tetapkan konkurensi maksimum ke nilai yang memproses cukup banyak untuk memenuhi tujuan tingkat layanan Anda tetapi tidak menggunakan batas konkurensi yang memengaruhi fungsi Lambda lain. Pertimbangkan untuk menetapkan konkurensi cadangan pada fungsi Lambda lain di akun dan wilayah yang sama saat Anda menggunakan antrean dengan Lambda.
- Gunakan API Gateway dengan integrasi layanan native ke Amazon SQS atau Kinesis untuk melakukan buffer permintaan.
- Jika Anda tidak dapat menggunakan API Gateway, lihat pustaka bahasa khusus untuk mengimplementasikan algoritme bucket token untuk beban kerja Anda. Periksa bagian contoh dan lakukan riset sendiri untuk menemukan pustaka yang cocok.
- Uji batas yang ingin Anda tetapkan, atau yang ingin Anda izinkan untuk ditingkatkan, dan dokumentasikan batas yang diuji.
- Jangan tingkatkan batas melebihi apa yang Anda tetapkan dalam pengujian. Saat meningkatkan batas, verifikasi bahwa sumber daya yang disediakan sudah setara atau lebih besar daripada yang ada dalam skenario pengujian sebelum menerapkan peningkatan.

Sumber daya

Praktik terbaik terkait:

- REL04-BP03 Melakukan tugas konstan
- REL05-BP03 Mengontrol dan membatasi panggilan percobaan ulang

Dokumen terkait:

- Amazon API Gateway: Membatasi Permintaan API untuk Peningkatan Throughput
- AWS WAF: Pernyataan aturan berbasis laju
- Memperkenalkan konkurensi maksimum AWS Lambda saat menggunakan Amazon SQS sebagai sumber peristiwa
- AWS Lambda: Konkurensi Maksimum

Contoh terkait:

- Tiga aturan berbasis laju AWS WAF yang paling penting
- Java Bucket4j
- Bucket token Python
- Bucket token Node
- Pembatasan Tingkat Threading Sistem .NET

Video terkait:

- Mengimplementasikan praktik terbaik keamanan API GraphQL dengan AWS AppSync

Alat terkait:

- Amazon API Gateway
- AWS AppSync
- Amazon SQS
- Amazon Kinesis
- AWS WAF

REL05-BP03 Mengontrol dan membatasi panggilan percobaan ulang

Gunakan mundur eksponensial untuk mencoba ulang permintaan dengan interval yang makin lama antara setiap percobaan ulang. Terapkan jitter antara percobaan ulang untuk mengacak interval percobaan ulang. Batasi jumlah percobaan ulang maksimum.

Hasil yang diinginkan: Komponen di sistem perangkat lunak terdistribusi biasanya mencakup server, penyeimbang beban, basis data, dan server DNS. Selama operasi normal, komponen-komponen ini dapat merespons permintaan dengan kesalahan yang bersifat sementara atau terbatas, dan juga kesalahan yang persisten terlepas dari percobaan ulang. Ketika klien membuat permintaan ke layanan, permintaan tersebut mengonsumsi sumber daya termasuk memori, thread, koneksi, port, atau sumber daya terbatas lainnya. Mengontrol dan membatasi percobaan ulang adalah strategi untuk melepaskan dan meminimalkan konsumsi sumber daya sehingga komponen sistem yang ada di bawah tekanan tidak kewalahan.

Ketika permintaan klien mengalami batas waktu atau menerima respons kesalahan, mereka harus menentukan apakah akan mencoba lagi atau tidak. Jika mereka mencoba lagi, mereka melakukannya dengan mundur eksponensial dengan jitter dan nilai coba ulang maksimum. Karena itu, layanan dan proses backend mendapat kelonggaran beban dan waktu untuk pulih secara mandiri, sehingga menghasilkan pemulihan yang lebih cepat dan pelayanan permintaan yang berhasil.

Antipola umum:

- Mengimplementasikan percobaan ulang tanpa menambahkan mundur eksponensial, jitter, dan nilai coba ulang maksimum. Mundur dan jitter membantu menghindari lonjakan lalu lintas semu yang disebabkan percobaan ulang yang dikoordinasikan secara tidak sengaja pada interval umum.
- Mengimplementasikan percobaan ulang tanpa menguji efeknya atau berasumsi bahwa percobaan ulang sudah terintegrasi ke dalam SDK tanpa menguji skenario percobaan ulang.
- Tidak memahami kode kesalahan yang dipublikasikan dari dependensi, yang menyebabkan percobaan ulang semua kesalahan, termasuk kesalahan dengan penyebab jelas yang menunjukkan tidak adanya izin, kesalahan konfigurasi, atau kondisi lain yang jelas tidak akan terselesaikan tanpa intervensi manual.
- Tidak menangani praktik observabilitas, termasuk pemantauan dan peringatan tentang kegagalan layanan berulang sehingga masalah yang mendasari dapat diketahui dan diatasi.
- Mengembangkan mekanisme percobaan ulang kustom saat kemampuan coba ulang bawaan atau pihak ketiga sudah mencukupi.
- Mencoba ulang pada beberapa lapisan tumpukan aplikasi dengan cara yang makin memperparah upaya-upaya percobaan ulang sehingga makin menyita sumber daya dalam badai percobaan ulang. Pastikan Anda memahami bagaimana kesalahan-kesalahan ini memengaruhi aplikasi Anda dan dependensi yang Anda andalkan, lalu terapkan percobaan ulang hanya pada satu tingkat.
- Mencoba ulang panggilan layanan yang tidak idempoten, sehingga menyebabkan efek samping yang tidak terduga seperti hasil-hasil ganda.

Manfaat menjalankan praktik terbaik ini: Percobaan ulang membantu klien memperoleh hasil yang diinginkan ketika permintaan gagal tetapi juga menyita lebih banyak waktu server untuk mendapatkan respons berhasil yang mereka inginkan. Ketika kegagalan jarang terjadi atau sementara, percobaan ulang dapat berfungsi dengan baik. Ketika kegagalan disebabkan oleh kelebihan beban sumber daya, percobaan ulang dapat memperburuk keadaan. Menambahkan mundur eksponensial dengan jitter ke percobaan ulang klien memungkinkan server pulih ketika kegagalan disebabkan oleh kelebihan beban sumber daya. Jitter menghindarkan penyelarasan permintaan menjadi lonjakan, dan mundur dapat mengurangi eskalasi beban yang disebabkan oleh penambahan percobaan ulang ke beban permintaan normal. Terakhir, penting untuk mengonfigurasi jumlah coba ulang maksimum atau waktu yang telah berlalu untuk menghindari terciptanya backlog yang menghasilkan kegagalan yang metastabil.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Mengontrol dan membatasi panggilan percobaan ulang. Gunakan mundur eksponensial untuk percobaan ulang setelah interval yang makin lama. Masukkan jitter untuk mengacak interval percobaan ulang dan batasi jumlah percobaan ulang maksimum.

Beberapa SDK AWS mengimplementasikan percobaan ulang dan mundur eksponensial secara default. Gunakan implementasi AWS bawaan ini jika diperlukan untuk beban kerja Anda. Implementasikan logika serupa dalam beban kerja Anda saat memanggil layanan yang idempoten dan apabila percobaan ulang meningkatkan ketersediaan klien Anda. Tentukan batas waktu dan kapan harus berhenti mencoba ulang berdasarkan kasus penggunaan Anda. Buat dan latih skenario pengujian untuk kasus penggunaan percobaan ulang tersebut.

Langkah implementasi

- Tentukan lapisan optimal dalam tumpukan aplikasi Anda untuk mengimplementasikan percobaan ulang untuk layanan yang diandalkan aplikasi Anda.
- Waspadai SDK yang ada yang menerapkan strategi percobaan ulang yang telah terbukti dengan mundur eksponensial dan jitter untuk bahasa pilihan Anda, dan pilih opsi ini daripada menulis implementasi percobaan ulang Anda sendiri.
- Verifikasikan bahwa layanan bersifat idempoten sebelum menerapkan percobaan ulang. Setelah percobaan ulang diterapkan, pastikan keduanya diuji dan latihlah secara rutin dalam produksi.
- Saat memanggil API layanan AWS, gunakan SDK AWS dan AWS CLI dan pahami opsi-opsi konfigurasi percobaan ulang. Tentukan apakah konfigurasi default cocok untuk kasus penggunaan Anda, uji, dan sesuaikan sesuai kebutuhan.

Sumber daya

Praktik terbaik terkait:

- REL04-BP04 Menjadikan semua respons idempoten
- REL05-BP02 Membatasi (throttling) permintaan
- REL05-BP04 Melakukan gagal cepat (fail fast) dan membatasi antrean
- REL05-BP05 Mengatur batas waktu klien
- REL11-BP01 Memantau semua komponen beban kerja untuk mendeteksi kegagalan

Dokumen terkait:

- Kesalahan Percobaan Ulang dan Mundur Eksponensial di AWS
- Amazon Builders' Library: Batas waktu, percobaan ulang, dan mundur (backoff) dengan jitter
- Mundur Eksponensial dan Jitter
- Menjadikan percobaan ulang aman dengan API idempoten

Contoh terkait:

- Spring Retry
- Resilience4j Retry

Video terkait:

- Percobaan ulang, mundur, dan jitter: AWS re:Invent 2019: Memperkenalkan Pustaka Pengembang Amazon (DOP328)

Alat terkait:

- SDK dan Alat-Alat AWS: Perilaku percobaan ulang
- AWS Command Line Interface: Percobaan ulang AWS CLI

REL05-BP04 Melakukan gagal cepat (fail fast) dan membatasi antrean

Ketika layanan tidak berhasil merespons permintaan, lakukanlah gagal cepat (fail fast). Hal ini memungkinkan pelepasan sumber daya yang terkait dengan permintaan, dan mengizinkan layanan

untuk melakukan pemulihan jika kehabisan sumber daya. Gagal cepat adalah pola desain perangkat lunak mapan yang dapat dimanfaatkan untuk membangun beban kerja yang sangat andal di cloud. Antrean juga merupakan pola integrasi korporat yang mapan yang dapat memperlancar beban dan memungkinkan klien untuk melepaskan sumber daya ketika pemrosesan asinkron dapat ditoleransi. Ketika layanan berhasil merespons dalam kondisi normal tetapi gagal ketika laju permintaan terlalu tinggi, gunakan antrean untuk melakukan buffer permintaan. Namun, jangan sampai ada penumpukan backlog antrean panjang yang dapat mengakibatkan diprosesnya permintaan yang telah kedaluwarsa dan telah ditinggalkan klien.

Hasil yang diinginkan: Ketika sistem berebut sumber daya, mengalami waktu habis, pengecualian, atau grey failure (kegagalan samar-samar) yang menyebabkan target tingkat layanan tidak dapat dicapai, strategi gagal cepat memungkinkan pemulihan sistem yang lebih cepat. Sistem yang harus menyerap lonjakan lalu lintas dan dapat mengakomodasi pemrosesan asinkron dapat meningkatkan keandalan dengan memungkinkan klien untuk secara cepat melepaskan permintaan dengan menggunakan antrean untuk melakukan buffer permintaan ke layanan backend. Ketika melakukan buffer permintaan ke antrean, strategi manajemen antrean diimplementasikan untuk menghindari backlog yang terlalu membebani.

Antipola umum:

- Mengimplementasikan antrean pesan tetapi tidak mengonfigurasi antrean surat mati (DLQ) atau alarm pada volume DLQ untuk mendeteksi kegagalan sistem.
- Tidak mengukur usia pesan dalam antrean, yaitu ukuran latensi untuk mengetahui kapan konsumen antrean tertinggal atau mengalami kesalahan yang menyebabkan percobaan ulang.
- Tidak menghapus pesan-pesan yang menumpuk dari antrean, padahal tidak ada gunanya memproses pesan-pesan tersebut jika kebutuhan bisnis sudah tidak ada.
- Mengonfigurasi antrean first in first out (FIFO), padahal antrean last in first out (LIFO) lebih memenuhi kebutuhan klien, misalnya ketika pengurutan yang ketat tidak diperlukan dan pemrosesan backlog menunda semua permintaan baru dan sensitif waktu sehingga semua klien merasa tingkat layanan gagal dipenuhi.
- Mengekspos antrean internal ke klien, bukan mengekspos API yang mengelola masuknya pekerjaan dan menempatkan permintaan ke dalam antrean internal.
- Menggabungkan terlalu banyak jenis permintaan kerja ke dalam satu antrean yang dapat memperburuk kondisi backlog dengan menyebarkan permintaan sumber daya di seluruh jenis permintaan.
- Memproses permintaan yang kompleks dan sederhana dalam antrean yang sama, sehingga mengabaikan perbedaan kebutuhan pemantauan, batas waktu, dan alokasi sumber daya.

- Tidak memvalidasi input atau menggunakan pernyataan untuk mengimplementasikan mekanisme gagal cepat dalam perangkat lunak yang menaikkan pengecualian ke komponen dengan level lebih tinggi yang dapat menangani kesalahan secara mulus.
- Tidak menghapus sumber daya yang rusak dari perutean permintaan, terutama ketika kegagalan samar-samar yang menunjukkan keberhasilan sekaligus kegagalan akibat crash dan mulai ulang, kegagalan dependensi intermiten, kapasitas yang menurun, atau hilangnya paket jaringan.

Manfaat menjalankan praktik terbaik ini: Sistem yang gagal cepat lebih mudah untuk di-debug dan diperbaiki, dan sering mengekspos masalah dalam pengodean dan konfigurasi sebelum rilis dipublikasikan ke tahap produksi. Sistem yang menggabungkan strategi antrean yang efektif memberikan ketahanan dan keandalan yang lebih baik terhadap lonjakan lalu lintas dan kondisi gangguan sistem intermiten.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Strategi gagal cepat dapat dikodekan ke dalam solusi perangkat lunak serta dikonfigurasi ke dalam infrastruktur. Selain gagal cepat, antrean adalah teknik arsitektur yang sederhana namun ampuh untuk memisahkan komponen-komponen sistem dan memperlancar beban. Amazon CloudWatch menyediakan kemampuan untuk memantau dan memberikan alarm kegagalan. Setelah sistem diketahui mengalami kegagalan, strategi mitigasi dapat dipanggil, termasuk gagal dan menjauh (fail away) dari sumber daya yang terdampak. Ketika sistem mengimplementasikan antrean dengan Amazon SQS dan teknologi antrean lainnya untuk melancarkan beban, sistem harus mempertimbangkan bagaimana mengelola backlog antrean, serta kegagalan konsumsi pesan.

Langkah implementasi

- Implementasikan pernyataan terprogram atau metrik tertentu dalam perangkat lunak Anda dan gunakan untuk memperingatkan secara eksplisit tentang masalah sistem. Amazon CloudWatch membantu Anda membuat metrik dan alarm berdasarkan pola log aplikasi dan instrumentasi SDK.
- Gunakan metrik dan alarm CloudWatch untuk gagal dan menjauh dari sumber daya terdampak yang menambahkan latensi ke pemrosesan atau berulang kali gagal memproses permintaan.
- Gunakan pemrosesan asinkron dengan merancang API untuk menerima permintaan dan menambahkan permintaan ke antrean internal menggunakan Amazon SQS kemudian menanggapi klien penghasil pesan dengan pesan keberhasilan sehingga klien dapat melepaskan sumber daya dan beralih dengan pekerjaan lain sementara konsumen antrean backend memproses permintaan.

- Ukur dan pantau latensi pemrosesan antrean dengan menghasilkan metrik CloudWatch setiap kali Anda melepaskan sebuah pesan dari antrean dengan membandingkan sekarang dengan stempel waktu pesan.
- Ketika kegagalan menghambat keberhasilan pemrosesan pesan atau volume lalu lintas melonjak sehingga tidak dapat diproses dalam batas perjanjian tingkat layanan, sisihkan lalu lintas yang lebih lama atau berlebih ke antrean spillover. Hal ini memungkinkan pemrosesan prioritas pada pekerjaan baru, dan pekerjaan yang lebih lama ketika kapasitas tersedia. Teknik ini mirip dengan pemrosesan LIFO dan memungkinkan pemrosesan sistem yang normal untuk semua pekerjaan baru.
- Gunakan antrean surat mati atau redrive untuk memindahkan pesan yang tidak dapat diproses dari backlog ke lokasi yang dapat dicari ulang dan diselesaikan lain waktu
- Coba lagi atau, apabila dapat ditoleransi, singkirkan pesan lama dengan membandingkan sekarang dengan stempel waktu pesan dan membuang pesan yang sudah tidak relevan dengan klien yang melakukan permintaan.

Sumber daya

Praktik terbaik terkait:

- REL04-BP02 Mengimplementasikan dependensi yang digabungkan secara longgar
- REL05-BP02 Membatasi (throttling) permintaan
- REL05-BP03 Mengontrol dan membatasi panggilan percobaan ulang
- REL06-BP02 Menetapkan dan menghitung metrik (Agregasi)
- REL06-BP07 Memantau pelacakan permintaan menyeluruh melalui sistem Anda

Dokumen terkait:

- Menghindari backlog antrian yang terlalu membebani
- Gagal Cepat (Fail Fast)
- Bagaimana cara mencegah peningkatan backlog pesan dalam antrean Amazon SQS saya?
- Elastic Load Balancing: Peralihan Zona
- Pengontrol Pemulihan Aplikasi Amazon Route 53: Kontrol perutean untuk failover lalu lintas

Contoh terkait:

- Pola Integrasi Korporat: Saluran Surat Mati

Video terkait:

- AWS re:Invent 2022 - Mengoperasikan aplikasi Multi-AZ dengan ketersediaan tinggi

Alat terkait:

- Amazon SQS
- Amazon MQ
- AWS IoT Core
- Amazon CloudWatch

REL05-BP05 Mengatur batas waktu klien

Atur batas waktu secara tepat pada koneksi dan permintaan, verifikasikan waktu tersebut secara sistematis, dan jangan selalu bergantung pada nilai default karena nilai tersebut mengabaikan hal-hal spesifik tentang beban kerja.

Hasil yang diinginkan: Batas waktu klien harus mempertimbangkan biaya untuk klien, server, dan beban kerja yang berkaitan dengan proses tunggu permintaan yang memerlukan waktu sangat lama untuk diselesaikan. Karena penyebab batas waktu tidak mungkin diketahui secara pasti, klien harus menggunakan pengetahuan tentang layanan untuk membangun ekspektasi tentang kemungkinan penyebab dan batas waktu yang tepat

Koneksi klien mengalami waktu habis berdasarkan nilai yang dikonfigurasi. Setelah mengalami batas waktu, klien mengambil keputusan untuk mundur dan mencobanya lagi atau membuka pemutus sirkuit. Pola-pola ini mencegah mengeluarkan permintaan yang dapat memperburuk kondisi kesalahan yang menyebabkannya.

Antipola umum:

- Tidak menyadari batas waktu sistem atau batas waktu default.
- Tidak menyadari waktu penyelesaian permintaan normal.
- Tidak menyadari kemungkinan penyebab permintaan membutuhkan waktu yang terlalu lama untuk diselesaikan, atau biaya untuk klien, layanan, atau kinerja beban kerja yang berkaitan dengan proses tunggu penyelesaian ini.

- Tidak menyadari kemungkinan jaringan rusak yang menyebabkan permintaan gagal hanya setelah batas waktu tercapai, dan biaya untuk klien dan kinerja beban kerja karena tidak mengadopsi batas waktu yang lebih singkat.
- Tidak menguji skenario batas waktu baik untuk koneksi maupun permintaan.
- Mengatur batas waktu terlalu tinggi, yang berimbas pada waktu tunggu yang lama dan meningkatkan pemanfaatan sumber daya.
- Mengatur batas waktu terlalu rendah, sehingga mengakibatkan kegagalan buatan.
- Mengabaikan pola-pola untuk menangani kesalahan batas waktu untuk panggilan jarak jauh seperti pemutus sirkuit dan percobaan ulang.
- Tidak mempertimbangkan pemantauan untuk angka kesalahan panggilan layanan, target latensi di tingkat layanan, dan outlier latensi. Metrik-metrik ini dapat memberikan wawasan tentang batas waktu yang agresif atau permisif

Manfaat menjalankan praktik terbaik ini: Waktu tunggu panggilan jarak jauh dikonfigurasi dan sistem dirancang untuk menangani batas waktu secara perlahan sehingga sumber daya dihemat ketika panggilan jarak jauh merespons terlalu lambat dan kesalahan batas waktu ditangani secara perlahan oleh klien layanan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Atur batas waktu koneksi dan batas waktu permintaan untuk panggilan dependensi layanan apa pun, serta secara umum untuk panggilan apa pun di seluruh proses. Banyak kerangka kerja yang menawarkan kemampuan batas waktu bawaan, tetapi Anda harus tetap memperhatikan bahwa nilai default bawaan bisa saja tidak terbatas atau lebih tinggi dari yang dapat diterima untuk sasaran layanan Anda. Nilai yang terlalu tinggi mengurangi kegunaan batas waktu karena sumber daya terus terpakai saat klien menunggu terjadinya batas waktu. Nilai yang terlalu rendah akan menyebabkan lalu lintas yang tinggi di backend serta meningkatkan latensi karena terlalu banyak permintaan yang dicoba ulang. Dalam beberapa kasus, hal ini dapat menyebabkan penghentian total karena semua permintaan dicoba ulang.

Pertimbangkan hal berikut saat menentukan strategi batas waktu:

- Permintaan mungkin membutuhkan waktu pemrosesan yang lebih lama dari biasanya dikarenakan kontennya, gangguan pada layanan target, atau kegagalan partisi jaringan.

- Permintaan dengan konten yang terlalu mahal dapat mengonsumsi sumber daya server dan klien yang tidak perlu. Dalam hal ini, membatasi waktu dan tidak mencoba ulang permintaan tersebut dapat menghemat sumber daya. Layanan juga harus melindungi diri dari konten yang terlalu mahal dengan throttle dan batas waktu sisi server.
- Permintaan yang memakan waktu terlalu lama karena gangguan layanan dapat diberikan batas waktu dan dicoba ulang. Pertimbangan harus diberikan pada biaya layanan untuk permintaan dan percobaan ulang, tetapi jika penyebabnya adalah gangguan yang terbatas di suatu tempat, percobaan ulang kemungkinan tidak mahal dan akan mengurangi konsumsi sumber daya klien. Batas waktu juga dapat melepaskan sumber daya server, tergantung sifat gangguan tersebut.
- Permintaan yang membutuhkan waktu penyelesaian yang lama karena permintaan atau respons gagal dikirimkan oleh jaringan dapat diberikan batas waktu dan dicoba ulang. Karena permintaan atau respons tidak dikirimkan, kegagalan akan terjadi, terlepas dari lamanya batas waktu. Memberikan batas waktu pada kasus ini tidak akan melepaskan sumber daya server, tetapi akan melepaskan sumber daya klien dan meningkatkan kinerja beban kerja.

Manfaatkan pola desain yang mapan seperti percobaan ulang dan pemutus sirkuit untuk menangani batas waktu dengan lancar dan mendukung pendekatan gagal cepat. SDK AWS dan AWS CLI memungkinkan konfigurasi batas waktu koneksi dan permintaan serta percobaan ulang dengan mundur eksponensial dan jitter. Fungsi AWS Lambda mendukung konfigurasi batas waktu, dan dengan AWS Step Functions, Anda dapat membangun pemutus sirkuit rendah kode yang memanfaatkan integrasi siap pakai dengan layanan dan SDK AWS. AWS App Mesh Envoy memberikan kemampuan batas waktu dan pemutus sirkuit.

Langkah implementasi

- Konfigurasikan batas waktu pada panggilan layanan jarak jauh dan manfaatkan fitur batas waktu bahasa bawaan atau pustaka batas waktu sumber terbuka.
- Saat beban kerja Anda melakukan panggilan dengan SDK AWS, tinjau dokumentasi untuk konfigurasi batas waktu untuk bahasa khusus.
  - Python
  - PHP
  - .NET
  - Ruby
  - Java
  - Go

  - Node.js
  - C++
- Saat menggunakan SDK AWS atau perintah AWS CLI dalam beban kerja Anda, konfigurasikan nilai batas waktu default dengan mengatur konfigurasi AWS default untuk `connectTimeoutInMillis` dan `tlsNegotiationTimeoutInMillis`.
- Terapkan opsi baris perintah `cli-connect-timeout` dan `cli-read-timeout` untuk mengontrol perintah AWS CLI satu kali ke layanan AWS.
- Pantau panggilan layanan jarak jauh untuk batas waktu, dan atur alarm pada kesalahan persisten sehingga Anda dapat menangani skenario kesalahan secara proaktif.
- Implementasikan Metrik CloudWatch dan deteksi anomali CloudWatch pada angka kesalahan panggilan, target latensi di tingkat layanan, dan outlier latensi untuk memberikan wawasan tentang pengelolaan batas waktu yang terlalu agresif atau permisif.
- Konfigurasikan batas waktu pada fungsi Lambda.
- Klien API Gateway harus mengimplementasikan percobaan ulang mereka sendiri saat menangani batas waktu. API Gateway mendukung batas waktu integrasi 50 milidetik hingga 29 detik untuk integrasi hilir dan tidak mencoba ulang saat integrasi meminta batas waktu.
- Implementasikan pola pemutus sirkuit untuk menghindari pembuatan panggilan jarak jauh ketika waktu habis. Buka sirkuit untuk menghindari kegagalan panggilan dan tutup sirkuit saat panggilan merespons secara normal.
- Untuk beban kerja berbasis kontainer, pelajari fitur App Mesh Envoy untuk memanfaatkan batas waktu dan pemutus sirkuit bawaan.
- Gunakan AWS Step Functions untuk membuat pemutus sirkuit rendah kode untuk panggilan layanan jarak jauh, terutama saat memanggil SDK native AWS dan integrasi Step Functions yang didukung untuk menyederhanakan beban kerja Anda.

Sumber daya

Praktik terbaik terkait:

- REL05-BP03 Mengontrol dan membatasi panggilan percobaan ulang
- REL05-BP04 Melakukan gagal cepat (fail fast) dan membatasi antrean
- REL06-BP07 Memantau pelacakan permintaan menyeluruh melalui sistem Anda

Dokumen terkait:

- SDK AWS: Percobaan Ulang dan Batas Waktu
- Amazon Builders' Library: Batas waktu, percobaan ulang, dan mundur (backoff) dengan jitter
- Kuota Amazon API Gateway dan catatan penting
- AWS Command Line Interface: Opsi baris perintah
- AWS SDK for Java 2.x: Mengonfigurasi Batas Waktu API
- AWS Botocore menggunakan objek konfigurasi dan Config Reference
- AWS SDK untuk .NET: Percobaan Ulang dan Batas Waktu
- AWS Lambda: Mengonfigurasi opsi fungsi Lambda

Contoh terkait:

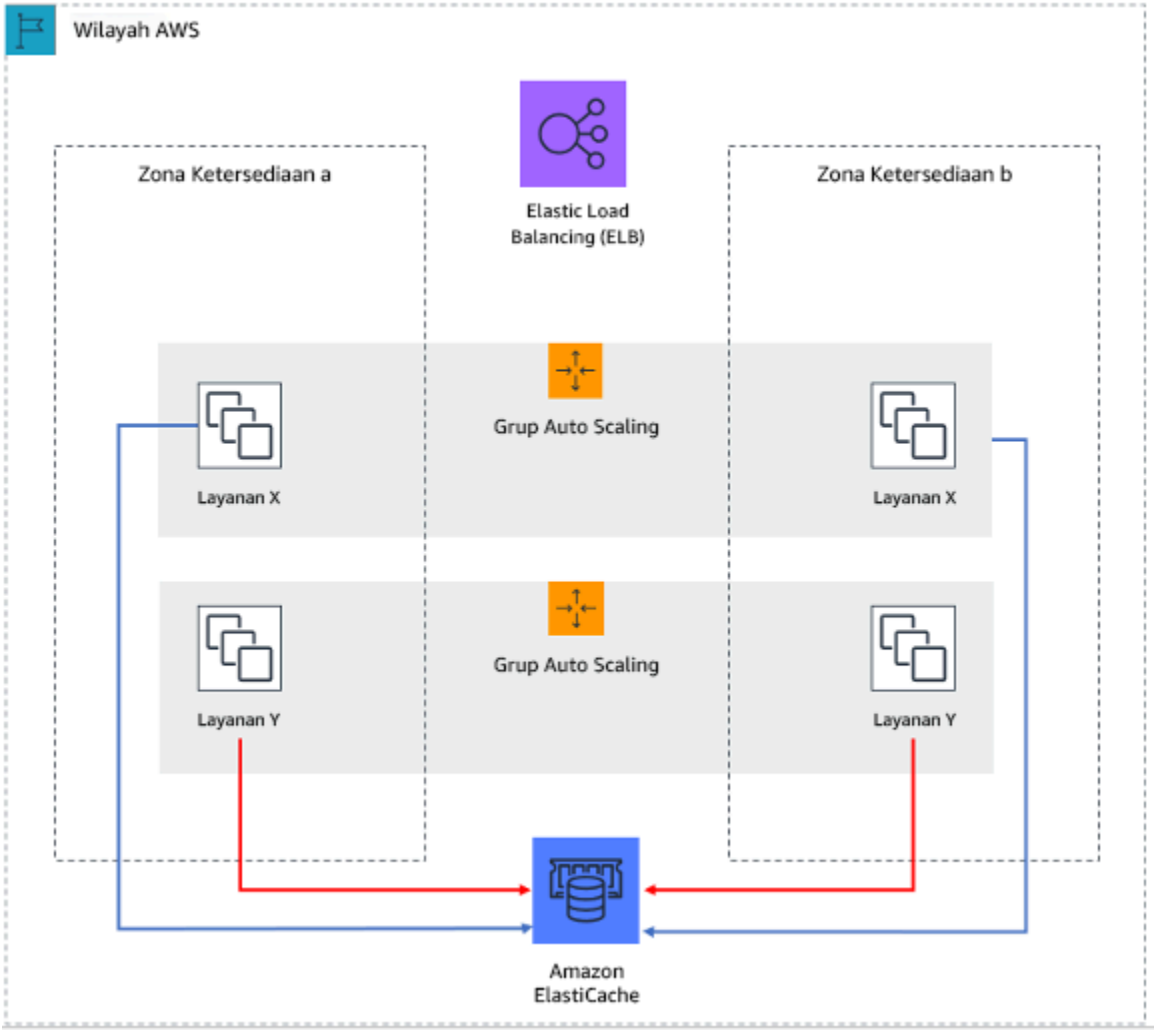
- Menggunakan pola pemutus sirkuit dengan AWS Step Functions dan Amazon DynamoDB
- Martin Fowler: CircuitBreaker

Alat terkait:

- SDK AWS
- Fungsi AWS Lambda
- Amazon SQS
- AWS Step Functions
- AWS Command Line Interface

REL05-BP06 Menjadikan layanan stateless jika memungkinkan

Layanan seharusnya tidak memerlukan state, atau seharusnya mengalihkan state sedemikian rupa sehingga di antara permintaan klien yang berbeda, tidak ada dependensi di penyimpanan data lokal di disk dan memori. Ini memungkinkan server diganti sesuka hati tanpa menyebabkan dampak ketersediaan. Amazon ElastiCache atau Amazon DynamoDB merupakan tujuan yang baik untuk state yang dialihkan.



Gambar 7: Pada aplikasi web stateless ini, state sesi dialihkan ke Amazon ElastiCache.

Ketika pengguna atau layanan berinteraksi dengan aplikasi, mereka sering melakukan serangkaian interaksi yang membentuk sesi. Sesi adalah data unik bagi pengguna yang lama berada di antara permintaan ketika mereka menggunakan aplikasi. Aplikasi stateless adalah aplikasi yang tidak memerlukan pengetahuan tentang interaksi sebelumnya dan tidak menyimpan informasi sesi.

Setelah dirancang menjadi stateless, Anda dapat menggunakan layanan komputasi nirserver, seperti AWS Lambda atau AWS Fargate.

Selain penggantian server, manfaat lain aplikasi stateless adalah kemampuannya untuk menyesuaikan skala secara horizontal karena sumber daya komputasi yang tersedia (seperti instans EC2 dan fungsi AWS Lambda) dapat melayani permintaan apa pun.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

- Menjadikan aplikasi Anda stateless. Aplikasi stateless memungkinkan penskalaan horizontal dan toleran terhadap kegagalan simpul individual.
  - Hapus state yang sebenarnya dapat disimpan di parameter permintaan.
  - Setelah memeriksa apakah state diperlukan, pindahkan pelacakan state apa pun ke cache multi-zona yang tangguh atau penyimpanan data seperti Amazon ElastiCache, Amazon RDS, Amazon DynamoDB atau solusi data terdistribusi pihak ketiga. Simpan state yang tidak dapat dipindah ke penyimpanan data tangguh.
    - Beberapa data (seperti cookie) dapat diteruskan di header atau parameter kueri.
    - Lakukan pemfaktoran ulang untuk menghapus state yang dapat dengan cepat diteruskan di permintaan.
    - Beberapa data mungkin tidak terlalu diperlukan per permintaan dan dapat diambil sesuai permintaan.
    - Hapus data yang dapat diambil secara asinkron.
    - Tentukan penyimpanan data yang memenuhi kebutuhan state yang diperlukan.
    - Pertimbangkan basis data NoSQL untuk data non-rasional.

Sumber daya

Dokumen terkait:

- Amazon Builders' Library: Menghindari fallback dalam sistem terdistribusi
- Amazon Builders' Library: Menghindari backlog antrean yang tidak dapat diatasi
- Amazon Builders' Library: Tantangan dan strategi caching

REL05-BP07 Mengimplementasikan tuas darurat

Tuas darurat adalah proses cepat yang dapat memitigasi dampak ketersediaan pada beban kerja.

Tuas darurat bekerja dengan cara menonaktifkan, melakukan throttling, atau mengubah perilaku komponen atau dependensi menggunakan mekanisme yang diketahui dan diuji. Hal ini dapat mengurangi gangguan beban kerja yang disebabkan oleh kelelahan sumber daya karena permintaan yang meningkat secara tidak terduga dan mengurangi dampak kegagalan pada komponen non-kritis dalam beban kerja Anda.

Hasil yang diinginkan: Dengan mengimplementasikan tuas darurat, Anda dapat membuat proses yang telah diketahui dengan baik untuk menjaga ketersediaan komponen kritis dalam beban kerja Anda. Beban kerja akan mengalami degradasi perlahan (graceful degradation) dan terus menjalankan fungsi-fungsi kritis bisnisnya selama aktivasi tuas darurat. Untuk detail lebih lanjut tentang degradasi perlahan, lihat REL05-BP01 Mengimplementasikan degradasi perlahan untuk mengubah dependensi keras yang berlaku menjadi dependensi lunak.

Antipola umum:

- Kegagalan dependensi non-kritis berdampak pada ketersediaan beban kerja inti Anda.
- Tidak menguji atau memverifikasi perilaku komponen kritis selama gangguan komponen non-kritis.
- Tidak ada kriteria yang jelas dan deterministik yang ditentukan untuk pengaktifan atau penonaktifan tuas darurat.

Manfaat menetapkan praktik terbaik ini: Mengimplementasikan tuas darurat dapat meningkatkan ketersediaan komponen kritis dalam beban kerja Anda dengan menyediakan proses yang telah ditetapkan kepada penyedia resolusi untuk merespons lonjakan permintaan yang tidak terduga atau kegagalan dependensi non-kritis.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

- Identifikasi komponen kritis dalam beban kerja Anda.
- Buat agar rancangan dan arsitek komponen kritis dalam beban kerja Anda dapat menahan kegagalan komponen non-kritis.
- Lakukan pengujian untuk memvalidasi perilaku komponen kritis Anda selama kegagalan komponen non-kritis.
- Tentukan dan pantau metrik atau pemicu yang relevan untuk memulai prosedur tuas darurat.
- Tentukan prosedur (manual atau otomatis) yang mencakup tuas darurat.

Langkah implementasi

- Identifikasi komponen kritis bagi bisnis dalam beban kerja Anda.
  - Setiap komponen teknis dalam beban kerja Anda harus dipetakan ke fungsi bisnisnya yang relevan dan diberi peringkat sebagai kritis atau non-kritis. Contoh-contoh fungsionalitas kritis

dan non-kritis di Amazon dapat dilihat di Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second.

- Ini adalah keputusan teknis sekaligus bisnis, dan bervariasi berdasarkan organisasi dan beban kerja.

- Buat agar rancangan dan arsitek komponen kritis dalam beban kerja Anda dapat menahan kegagalan komponen non-kritis.
  - Selama analisis dependensi, pertimbangkan semua mode kegagalan yang dapat terjadi, dan verifikasikan bahwa mekanisme tuas darurat Anda memberikan fungsionalitas kritis pada komponen hilir.
- Lakukan pengujian untuk memvalidasi perilaku komponen kritis Anda saat tuas darurat Anda diaktifkan.
  - Hindari perilaku bimodal. Untuk detail lebih lanjut, lihat REL11-BP05 Menggunakan stabilitas statis untuk mencegah perilaku bimodal.
- Tentukan, pantau, dan munculkan peringatan pada metrik yang relevan untuk memulai prosedur tuas darurat.
  - Beban kerja Anda menentukan metrik yang tepat untuk dipantau. Beberapa contoh metrik adalah latensi atau jumlah permintaan yang gagal ke sebuah dependensi.
- Tentukan prosedur, manual atau otomatis, yang mencakup tuas darurat.
  - Prosedur bisa meliputi mekanisme seperti pelepasan beban, permintaan throttling, atau implementasi degradasi perlahan.

Sumber daya

Praktik Terbaik Terkait:

- REL05-BP01 Mengimplementasikan degradasi yang tepat (graceful degradation) untuk mengubah dependensi keras yang berlaku menjadi dependensi lunak
- REL05-BP02 Membatasi (throttling) permintaan
- REL11-BP05 Menggunakan stabilitas statis untuk mencegah perilaku bimodal

Dokumen terkait:

- Mengotomatiskan deployment aman tanpa campur tangan
- Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second

Video terkait:

- AWS re:Invent 2020: Reliability, consistency, and confidence through immutability

Manajemen perubahan

Pertanyaan

- REL 6. Bagaimana cara memantau sumber daya beban kerja Anda?
- REL 7. Bagaimana cara mendesain beban kerja Anda untuk beradaptasi dengan perubahan dalam permintaan?
- REL 8. Bagaimana cara mengimplementasikan perubahan?

REL 6. Bagaimana cara memantau sumber daya beban kerja Anda?

Log dan metrik merupakan alat yang luar biasa untuk mendapatkan wawasan tentang kondisi beban kerja Anda. Anda dapat mengonfigurasikan beban kerja Anda untuk memantau log dan metrik serta mengirimkan notifikasi ketika ambang batas terlampaui atau peristiwa signifikan terjadi. Pemantauan memungkinkan beban kerja Anda mengenali ketika ambang batas kinerja rendah terlampaui atau kegagalan terjadi, sehingga pemulihan dapat terjadi secara otomatis untuk menanggapinya.

Praktik terbaik

- REL06-BP01 Memantau semua komponen untuk beban kerja (Pembuatan)
- REL06-BP02 Menetapkan dan menghitung metrik (Agregasi)
- REL06-BP03 Mengirimkan notifikasi (Pemrosesan dan pembuatan alarm waktu nyata)
- REL06-BP04 Mengotomatiskan respons (Peringatan dan pemrosesan waktu nyata)
- REL06-BP05 Analitik
- REL06-BP06 Lakukan peninjauan secara teratur
- REL06-BP07 Memantau pelacakan permintaan menyeluruh melalui sistem Anda

REL06-BP01 Memantau semua komponen untuk beban kerja (Pembuatan)

Pantau komponen beban kerja dengan Amazon CloudWatch atau alat pihak ketiga. Pantau layanan AWS dengan Dasbor AWS Health.

Semua komponen beban kerja Anda harus dipantau, mencakup front-end, logika bisnis, dan tingkat penyimpanan. Tetapkan metrik utama, jelaskan cara mengekstraknya dari log (jika diperlukan), dan tetapkan ambang batas untuk memicu peristiwa alarm yang sesuai. Pastikan metrik relevan dengan indikator kinerja utama (KPI) beban kerja Anda, dan gunakan metrik dan log untuk mengidentifikasi tanda-tanda peringatan dini penurunan layanan. Contohnya, metrik yang terkait dengan hasil bisnis seperti jumlah pesanan yang berhasil diproses per menit, dapat menunjukkan masalah beban kerja lebih cepat dari metrik teknis, seperti Pemanfaatan CPU. Gunakan Dasbor AWS Health untuk tampilan yang dipersonalisasi tentang kinerja dan ketersediaan layanan AWS yang mendasari sumber daya AWS Anda.

Pemantauan di cloud menawarkan peluang baru. Sebagian besar penyedia cloud telah mengembangkan hook yang dapat disesuaikan dan dapat menghadirkan wawasan untuk membantu Anda memantau beberapa lapisan beban kerja Anda. Layanan AWS seperti Amazon CloudWatch menerapkan algoritme statis dan machine learning untuk terus menganalisis metrik sistem dan aplikasi, menentukan garis dasar normal dan anomali permukaan dengan sedikit campur tangan pengguna. Algoritme deteksi anomali memperhitungkan perubahan musiman dan tren metrik.

AWS menyediakan banyak informasi pemantauan dan log untuk digunakan untuk menentukan metrik khusus beban kerja, proses perubahan permintaan, dan mengadopsi teknik machine learning, terlepas dari keahlian ML.

Selain itu, pantau semua titik akhir eksternal Anda untuk memastikan independensinya dari implementasi dasar Anda. Pemantauan aktif ini dapat dilakukan dengan transaksi sintetis (kadang disebut sebagai canary pengguna, tetapi bedakan dengan deployment canary) yang secara berkala menjalankan sejumlah tugas umum yang sesuai dengan tindakan yang dilakukan oleh klien beban kerja. Buat tugas-tugas ini berdurasi singkat dan pastikan untuk tidak membebani beban kerja Anda saat pengujian. Amazon CloudWatch Synthetics memungkinkan Anda untuk membuat canary sintesis untuk memantau titik akhir dan API Anda. Anda juga dapat menggabungkan simpul klien canary sintetis dengan konsol AWS X-Ray untuk mengidentifikasi canary sintetis mana yang mengalami masalah berupa error, fault, atau tingkat throttling untuk jangka waktu yang dipilih.

Hasil yang Diharapkan:

Kumpulkan dan gunakan metrik kritis dari semua komponen beban kerja untuk memastikan keandalan beban kerja dan pengalaman pengguna yang optimal. Dengan mendeteksi bahwa beban kerja tidak mencapai hasil bisnis, Anda dapat dengan cepat mengumumkan bencana dan pulih dari insiden.

Antipola umum:

- Hanya memantau antarmuka eksternal beban kerja Anda.
- Tidak menghasilkan metrik khusus beban kerja dan hanya bergantung pada metrik yang diberikan kepada Anda oleh layanan AWS yang digunakan oleh beban kerja Anda.
- Hanya menggunakan metrik teknis di beban kerja Anda dan tidak memantau metrik apa pun yang terkait dengan KPI non-teknis yang menerima kontribusi dari beban kerja Anda.
- Mengandalkan lalu lintas produksi dan pemeriksaan kondisi sederhana untuk memantau dan mengevaluasi state beban kerja.

Manfaat menjalankan praktik terbaik ini: Dengan memantau semua tingkatan di beban kerja Anda, Anda dapat lebih cepat mengantisipasi dan menyelesaikan masalah di komponen dalam beban kerja.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

1. Mengaktifkan pencatatan log jika tersedia. Data pemantauan harus diperoleh dari semua komponen beban kerja. Aktifkan pencatatan log tambahan, seperti S3 Access Logs, dan aktifkan beban kerja Anda untuk mencatat log data spesifik beban kerja. Kumpulkan metrik rata-rata CPU, I/O jaringan, dan I/O disk dari layanan seperti Amazon ECS, Amazon EKS, Amazon EC2, Elastic Load Balancing, AWS Auto Scaling, dan Amazon EMR. Lihat Layanan AWS yang Memublikasikan Metrik CloudWatch untuk daftar layanan AWS yang memublikasikan metrik ke CloudWatch.
2. Tinjau semua metrik default dan telusuri celah pengumpulan data apa pun. Setiap layanan menghasilkan metrik default. Dengan mengumpulkan metrik default, Anda dapat lebih memahami dependensi antar komponen beban kerja dan bagaimana keandalan dan kinerja komponen memengaruhi beban kerja. Anda juga dapat membuat dan memublikasikan metrik Anda sendiri ke CloudWatch menggunakan AWS CLI atau API. Ini
3. Evaluasi semua metrik untuk menentukan mana yang harus dibuatkan peringatan untuk setiap layanan AWS di beban kerja Anda. Anda dapat memilih subset metrik yang memiliki dampak besar dalam keandalan beban kerja. Berfokus pada metrik dan ambang batas memungkinkan Anda untuk menyempurnakan jumlah pemberitahuan dan dapat membantu meminimalkan positif palsu.
4. Tetapkan peringatan dan proses pemulihan beban kerja Anda setelah peringatan dipicu. Dengan menetapkan peringatan, Anda dapat dengan cepat memberi tahu, mengeskalasi, dan mengikuti langkah-langkah yang diperlukan untuk pemulihan dari insiden dan memenuhi Sasaran Waktu Pemulihan (RTO) yang Anda tentukan. Anda dapat menggunakan Alarm Amazon CloudWatch untuk memanggil alur kerja otomatis dan memulai prosedur pemulihan berdasarkan ambang batas yang ditentukan.

5. Jelajahi penggunaan transaksi sintetis untuk mengumpulkan data yang relevan tentang state beban kerja. Pemantauan sintetis mengikuti rute yang sama dan menjalankan tindakan yang sama seperti pelanggan, sehingga memungkinkan Anda untuk terus memverifikasi pengalaman pelanggan Anda bahkan saat Anda tidak memiliki lalu lintas pelanggan apa pun pada beban kerja Anda. Dengan menggunakan transaksi sintetis, Anda dapat menemukan masalah sebelum pelanggan Anda menemukannya.

Sumber daya

Praktik Terbaik Terkait:

- REL11-BP03 Mengotomatisasi pemulihan di semua lapisan

Dokumen terkait:

- Memulai Dasbor AWS Health Anda – Kondisi akun Anda
- Layanan AWS yang Memublikasikan Metrik CloudWatch
- Mengakses Log untuk Penyeimbang Beban Jaringan Anda
- Akes log untuk penyeimbang beban aplikasi Anda
- Mengakses Amazon CloudWatch Logs untuk AWS Lambda
- Pencatatan Log Akses Server S3
- Mengaktifkan Log Akses untuk Penyeimbang Beban Klasik Anda
- Mengekspor data log ke Amazon S3
- Menginstal agen CloudWatch di instans Amazon EC2
- Memublikasikan Metrik Kustom
- Menggunakan Dasbor Amazon CloudWatch
- Menggunakan Metrik Amazon CloudWatch
- Menggunakan Canary (Amazon CloudWatch Synthetics)
- Apa itu Amazon CloudWatch Logs?

Panduan pengguna:

- Membuat trail
- Memantau metrik memori dan disk untuk instans Linux Amazon EC2
- Menggunakan CloudWatch Logs dengan instans kontainer

- VPC Flow Logs
- Apa itu Amazon DevOps Guru?
- Apa itu AWS X-Ray?

Blog terkait:

- Melakukan debug dengan Amazon CloudWatch Synthetics dan AWS X-Ray

Contoh dan lokakarya terkait:

- AWS Well-Architected Labs: Keunggulan Operasional - Pemantauan Dependensi
- Amazon Builders' Library: Instrumentasi sistem terdistribusi untuk visibilitas operasional
- Lokakarya observabilitas

REL06-BP02 Menetapkan dan menghitung metrik (Agregasi)

Simpan data log dan terapkan filter saat diperlukan untuk menghitung metrik, seperti jumlah log peristiwa tertentu, atau latensi yang dihitung dari stempel waktu log peristiwa.

Amazon CloudWatch dan Amazon S3 berfungsi sebagai lapisan agregasi dan penyimpanan utama. Untuk beberapa layanan, seperti AWS Auto Scaling dan Elastic Load Balancing, metrik default disediakan secara default untuk beban CPU atau latensi permintaan rata-rata di seluruh klaster atau instans. Untuk layanan streaming, seperti VPC Flow Logs dan AWS CloudTrail, data peristiwa diteruskan ke CloudWatch Logs dan Anda perlu menetapkan dan menerapkan filter metrik untuk mengekstrak metrik dari data peristiwa. Ini memberi Anda data rangkaian waktu, yang dapat berfungsi sebagai input ke alarm CloudWatch yang Anda tetapkan untuk memicu peringatan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Tetapkan dan hitung metrik (Agregasi). Simpan data log dan terapkan filter saat diperlukan untuk menghitung metrik, seperti jumlah log peristiwa tertentu, atau latensi yang dihitung dari stempel waktu log peristiwa.
  - Filter metrik menetapkan ketentuan dan pola yang harus dicari di data log saat dikirim ke CloudWatch Logs. CloudWatch Logs menggunakan filter metrik untuk mengubah data log menjadi metrik CloudWatch numerik yang dapat Anda buatkan grafik atau aktifkan alarm.

    - Mencari dan Menyaring Data Log
- Gunakan pihak ketiga tepercaya untuk mengagregasi log.
    - Ikuti instruksi pihak ketiga. Sebagian besar produk pihak ketiga terintegrasi dengan CloudWatch dan Amazon S3.
- Beberapa layanan AWS dapat memublikasikan log langsung ke Amazon S3. Jika kebutuhan utama Anda untuk log adalah penyimpanan di Amazon S3, Anda dapat dengan mudah meminta layanan yang memproduksi log untuk mengirimnya langsung ke Amazon S3 tanpa mengatur infrastruktur tambahan.
    - Mengirim Log Langsung ke Amazon S3

Sumber daya

Dokumen terkait:

- Contoh Kueri Amazon CloudWatch Logs Insight
- Melakukan debug dengan Amazon CloudWatch Synthetics dan AWS X-Ray
- One Observability Workshop
- Mencari dan Menyaring Data Log
- Mengirim Log Langsung ke Amazon S3
- Amazon Builders' Library: Menginstrumentasi sistem terdistribusi untuk visibilitas operasional

REL06-BP03 Mengirimkan notifikasi (Pemrosesan dan pembuatan alarm waktu nyata)

Ketika organisasi mendeteksi potensi masalah, mereka mengirimkan notifikasi dan peringatan waktu nyata kepada personel dan sistem yang sesuai untuk merespons masalah ini dengan cepat dan efektif.

Hasil yang diinginkan: Respons cepat terhadap event operasional dapat terjadi melalui konfigurasi alarm yang relevan berdasarkan metrik layanan dan aplikasi. Ketika ambang batas alarm dilanggar, personel dan sistem yang sesuai diberitahu sehingga mereka dapat mengatasi masalah mendasar.

Antipola umum:

- Mengonfigurasi alarm dengan ambang batas yang terlalu tinggi, sehingga mengakibatkan kegagalan untuk mengirim notifikasi penting.

- Mengonfigurasi alarm dengan ambang batas yang terlalu rendah, yang menyebabkan tidak adanya tindakan atas pemberitahuan penting karena kebisingan notifikasi yang berlebihan.
- Tidak memperbarui alarm dan ambang batasnya saat penggunaan berubah.
- Untuk alarm yang paling sesuai untuk ditangani melalui tindakan otomatis, mengirim notifikasi ke personel alih-alih menghasilkan tindakan otomatis, menyebabkan terkirimnya notifikasi yang berlebihan.

Manfaat menjalankan praktik terbaik ini: Mengirimkan notifikasi dan pemberitahuan waktu nyata kepada personel dan sistem yang sesuai memungkinkan deteksi dini masalah dan respons cepat terhadap insiden operasional.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Beban kerja harus dilengkapi dengan pemrosesan dan pembuatan alarm waktu nyata untuk meningkatkan pendeteksian masalah yang dapat memengaruhi ketersediaan aplikasi dan berfungsi sebagai pemicu respons otomatis. Organisasi dapat melakukan pemrosesan dan pembuatan alarm waktu nyata dengan menciptakan peringatan dengan metrik yang ditentukan untuk menerima notifikasi setiap kali peristiwa signifikan terjadi atau sebuah metrik melebihi ambang batas.

Amazon CloudWatch memungkinkan Anda untuk membuat metrik dan alarm komposit menggunakan alarm CloudWatch berdasarkan ambang batas statis, deteksi anomali, dan kriteria lainnya. Untuk detail selengkapnya tentang jenis alarm yang dapat Anda konfigurasikan menggunakan CloudWatch, lihat bagian alarm dalam dokumentasi CloudWatch.

Anda dapat membuat tampilan metrik dan pemberitahuan yang disesuaikan dari sumber daya AWS Anda untuk tim Anda menggunakan dasbor CloudWatch. Halaman beranda yang dapat disesuaikan di konsol CloudWatch memungkinkan Anda memantau sumber daya dalam satu tampilan di beberapa Region.

Alarm dapat melakukan satu atau beberapa tindakan, seperti mengirimkan notifikasi ke topik Amazon SNS, melakukan tindakan Amazon EC2 atau tindakan Amazon EC2 Auto Scaling , atau membuat OpsItem atau insiden di AWS Systems Manager.

Amazon CloudWatch menggunakan Amazon SNS untuk mengirimkan notifikasi ketika alarm berubah status, sehingga memberikan pengiriman pesan dari penerbit (produsen) ke pelanggan (konsumen). Untuk detail selengkapnya tentang pengaturan notifikasi Amazon SNS, lihat Mengonfigurasi Amazon SNS.

CloudWatch mengirim event EventBridge setiap kali alarm CloudWatch dibuat, diperbarui, dihapus, atau statusnya berubah. Anda dapat menggunakan EventBridge dengan event ini untuk membuat aturan yang melakukan tindakan, seperti memberi tahu Anda setiap kali status alarm berubah atau secara otomatis memicu event di akun Anda menggunakan Otomatisasi Systems Manager.

Kapan Anda harus menggunakan EventBridge atau Amazon SNS?

Baik EventBridge maupun Amazon SNS dapat digunakan untuk mengembangkan aplikasi berbasis event, dan pilihan Anda akan tergantung pada kebutuhan spesifik Anda.

Amazon EventBridge direkomendasikan ketika Anda ingin membangun aplikasi yang bereaksi terhadap event dari aplikasi Anda sendiri, aplikasi SaaS, dan layanan AWS. EventBridge adalah satu-satunya layanan berbasis event yang terintegrasi langsung dengan partner SaaS pihak ketiga. EventBridge juga secara otomatis menyerap peristiwa dari lebih dari 200 layanan AWS tanpa mengharuskan developer untuk membuat sumber daya apa pun di akun mereka.

EventBridge menggunakan struktur berbasis JSON yang ditentukan untuk event, dan membantu Anda membuat aturan yang diterapkan di seluruh badan event untuk memilih event untuk diteruskan ke target. EventBridge saat ini mendukung lebih dari 20 layanan AWS sebagai target, termasuk AWS Lambda, Amazon SQS, Amazon SNS, Amazon Kinesis Data Streams, dan Amazon Data Firehose.

Amazon SNS direkomendasikan untuk aplikasi yang membutuhkan fan out tinggi (ribuan atau jutaan titik akhir). Pola umum yang kita lihat adalah bahwa pelanggan menggunakan Amazon SNS sebagai target aturan mereka untuk memfilter event yang mereka butuhkan dan fan out ke beberapa titik akhir.

Pesan memiliki sifat yang tidak terstruktur dan bisa dalam format apa pun. Amazon SNS mendukung penerusan pesan ke enam jenis target yang berbeda, termasuk Lambda, Amazon SQS, titik akhir HTTP/S, SMS, push seluler, dan email. Amazon SNS latensi tipikal di bawah 30 milidetik. Berbagai layanan AWS mengirimkan pesan Amazon SNS dengan mengonfigurasi layanan untuk melakukannya (lebih dari 30, termasuk Amazon EC2, Amazon S3, dan Amazon RDS).

Langkah implementasi

1. Buat alarm menggunakan alarm Amazon CloudWatch.
    a. Alarm metrik memonitor metrik CloudWatch tunggal atau ekspresi yang bergantung pada metrik CloudWatch. Alarm memulai satu atau beberapa tindakan berdasarkan nilai metrik atau ekspresi dibandingkan dengan ambang batas selama interval waktu tertentu. Tindakan tersebut dapat terdiri dari pengiriman notifikasi ke topik Amazon SNS, melakukan tindakan Amazon EC2

atau tindakan Amazon EC2 Auto Scaling , atau membuat OpsItem atau acara di AWS Systems Manager.

b. Alarm komposit terdiri dari ekspresi aturan yang mempertimbangkan kondisi alarm dari alarm-alarm lain yang telah Anda buat. Alarm komposit hanya memasuki status alarm jika semua kondisi aturan terpenuhi. Alarm yang ditentukan dalam ekspresi aturan suatu alarm komposit dapat mencakup alarm metrik dan alarm komposit tambahan. Alarm komposit dapat mengirim notifikasi Amazon SNS ketika statusnya berubah dan dapat membuat Systems Manager OpsItems atau insiden ketika memasuki keadaan alarm, tetapi tidak dapat melakukan tindakan Amazon EC2 atau Auto Scaling.

2. Siapkan notifikasi Amazon SNS. Saat membuat alarm CloudWatch, Anda dapat menyertakan sebuah topik Amazon SNS untuk mengirimkan notifikasi saat status alarm berubah.
3. Buat aturan di EventBridge yang cocok dengan alarm CloudWatch yang ditentukan. Setiap aturan mendukung beberapa target, termasuk fungsi Lambda. Misalnya, Anda dapat menentukan alarm yang dimulai saat ruang disk yang tersedia hampir habis, yang memicu sebuah fungsi Lambda melalui aturan EventBridge, untuk mengosongkan ruang. Untuk detail selengkapnya tentang target EventBridge, lihat Target EventBridge.

Sumber daya

Praktik terbaik Well-Architected terkait:

- REL06-BP01 Memantau semua komponen untuk beban kerja (Pembuatan)
- REL06-BP02 Menetapkan dan menghitung metrik (Agregasi)
- REL12-BP01 Menggunakan buku pedoman untuk menyelidiki kegagalan

Dokumen terkait:

- Amazon CloudWatch
- Wawasan CloudWatch Logs
- Menggunakan alarm Amazon CloudWatch
- Menggunakan dasbor Amazon CloudWatch
- Menggunakan metrik Amazon CloudWatch
- Menyiapkan notifikasi Amazon SNS
- deteksi anomali CloudWatch

- Perlindungan data CloudWatch Logs
- Amazon EventBridge
- Amazon Simple Notification Service

Video terkait:

- video observabilitas reinvent 2022
- AWS re:Invent 2022 - Praktik terbaik observabilitas di Amazon

Contoh terkait:

- Lokakarya One Observability
- Amazon EventBridge ke AWS Lambda dengan kontrol umpan balik oleh Alarm Amazon CloudWatch

REL06-BP04 Mengotomatiskan respons (Peringatan dan pemrosesan waktu nyata)

Gunakan otomatisasi untuk melakukan tindakan ketika peristiwa terdeteksi, misalnya, mengganti komponen yang rusak.

Pemrosesan alarm waktu nyata secara otomatis diimplementasikan agar sistem dapat mengambil tindakan korektif yang cepat dan berupaya mencegah kegagalan atau penurunan layanan ketika alarm terpicu. Respons otomatis terhadap alarm dapat mencakup penggantian komponen yang gagal, penyesuaian kapasitas komputasi, pengalihan lalu lintas ke host yang sehat, zona ketersediaan, atau wilayah lain, dan pemberitahuan operator.

Hasil yang diinginkan: Alarm waktu nyata diidentifikasi, dan pemrosesan alarm secara otomatis diatur untuk menginvokasi tindakan yang tepat yang diambil untuk mempertahankan tujuan tingkat layanan dan perjanjian tingkat layanan (SLA). Otomatisasi dapat berupa berbagai hal, dari aktivitas pemulihan diri sebuah komponen hingga failover seluruh situs.

Antipola umum:

- Tidak memiliki inventaris atau katalog alarm waktu nyata utama yang jelas.
- Tidak ada respons otomatis terhadap alarm kritis (misalnya, penskalaan otomatis berjalan ketika komputasi hampir habis).
- Tindakan respons alarm yang kontradiktif.

- Tidak ada prosedur operasi standar (SOP) untuk diikuti operator ketika mereka menerima pemberitahuan peringatan.
- Tidak memantau perubahan konfigurasi, padahal perubahan konfigurasi yang tidak terdeteksi dapat menyebabkan waktu henti untuk beban kerja.
- Tidak memiliki strategi untuk membatalkan perubahan konfigurasi yang tidak diinginkan.

Manfaat menetapkan praktik terbaik ini: Mengotomatiskan pemrosesan alarm dapat meningkatkan ketahanan sistem. Sistem mengambil tindakan korektif secara otomatis, sehingga mengurangi aktivitas manual yang memberi peluang adanya intervensi manusia yang rawan kesalahan. Operasi beban kerja memenuhi tujuan ketersediaan, dan mengurangi gangguan layanan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Untuk mengelola peringatan secara efektif dan mengotomatiskan responsnya, kategorikan peringatan berdasarkan tingkat kekritisan dan dampaknya, dokumentasikan prosedur respons, dan rencanakan respons sebelum menentukan peringkat tugas.

Identifikasi tugas yang membutuhkan tindakan tertentu (sering kali diperinci dalam runbook), dan periksa semua runbook dan playbook untuk menentukan tugas mana yang dapat diotomatisasi. Jika tindakan dapat digambarkan dengan jelas, tindakan tersebut sering kali dapat diotomatisasi. Jika tindakan tidak dapat diotomatisasi, dokumentasikan langkah-langkah manual dalam SOP dan latih operator untuk melakukannya. Terus cari peluang otomatisasi pada proses manual agar Anda dapat membuat dan menerapkan rencana untuk mengotomatiskan respons peringatan.

Langkah implementasi

1. Buat inventaris alarm: Untuk mendapatkan daftar semua alarm, Anda dapat memanfaatkan AWS CLI menggunakan perintah Amazon CloudWatch`describe-alarms`. Bergantung pada berapa banyak alarm yang telah Anda siapkan, Anda mungkin harus menggunakan paginasi untuk mengambil subset alarm untuk setiap panggilan, atau menggunakan SDK AWS untuk mendapatkan alarm menggunakan panggilan API.
2. Dokumentasikan semua tindakan alarm: Perbarui runbook dengan semua alarm dan tindakannya, baik itu manual maupun otomatis. AWS Systems Manager menyediakan runbook yang sudah ditetapkan sebelumnya. Untuk informasi selengkapnya tentang runbook, lihat Working with runbooks. Untuk detail tentang cara melihat konten runbook, lihat View runbook content.

3. Menyiapkan dan mengelola tindakan alarm: Untuk alarm apa pun yang memerlukan tindakan, tentukan tindakan otomatis menggunakan SDK CloudWatch. Misalnya, Anda dapat mengubah status instans Amazon EC2 secara otomatis berdasarkan alarm CloudWatch dengan membuat dan mengaktifkan tindakan pada alarm atau menonaktifkan tindakan pada alarm.

   Anda juga dapat menggunakan Amazon EventBridge untuk merespons peristiwa sistem secara otomatis, seperti masalah ketersediaan aplikasi atau perubahan sumber daya. Anda dapat membuat aturan untuk menunjukkan peristiwa mana yang perlu diperhatikan, dan tindakan yang harus diambil apabila peristiwa cocok dengan sebuah aturan. Tindakan yang dapat dimulai secara otomatis termasuk menginvokasi fungsi AWS Lambda, menginvokasi Amazon EC2 `Run Command`, merelai peristiwa tersebut ke Amazon Kinesis Data Streams, dan melihat Otomatiskan Amazon EC2 menggunakan EventBridge.

4. Prosedur Operasi Standar (SOP): Berdasarkan komponen aplikasi Anda, AWS Resilience Hub merekomendasikan beberapa templat SOP. Anda dapat menggunakan SOP ini untuk mendokumentasikan semua proses yang harus diikuti operator jika peringatan muncul. Anda juga dapat menyusun SOP berdasarkan rekomendasi Resilience Hub, dan untuk ini Anda memerlukan aplikasi Resilience Hub dengan kebijakan ketahanan terkait, serta penilaian ketahanan historis terhadap aplikasi tersebut. Rekomendasi untuk SOP Anda berasal dari penilaian ketahanan.

   Resilience Hub bekerja dengan Systems Manager untuk mengotomatiskan langkah-langkah dalam SOP Anda dengan menyediakan sejumlah dokumen SSM yang dapat Anda gunakan sebagai dasar untuk SOP tersebut. Misalnya, Resilience Hub mungkin merekomendasikan SOP untuk menambahkan ruang disk berdasarkan dokumen otomatisasi SSM yang sudah ada.

5. Lakukan tindakan otomatis menggunakan Amazon DevOps Guru: Anda dapat menggunakan Amazon DevOps Guru untuk secara otomatis memantau perilaku anomali pada sumber daya aplikasi, serta memberikan rekomendasi terarah untuk mempercepat waktu perbaikan serta identifikasi masalah. DenganDevOps Guru, Anda dapat memantau aliran data operasional hampir secara waktu nyata dari berbagai sumber termasuk metrik Amazon CloudWatch, AWS Config, AWS CloudFormation, dan AWS X-Ray. Anda juga dapat menggunakan DevOps Guru untuk secara otomatis membuat OpsItems di OpsCenter dan mengirim peristiwa ke EventBridge untuk otomatisasi tambahan.

Sumber daya

Praktik Terbaik Terkait:

- REL06-BP01 Memantau semua komponen untuk beban kerja (Pembuatan)

- REL06-BP02 Menetapkan dan menghitung metrik (Agregasi)
- REL06-BP03 Mengirimkan notifikasi (Pemrosesan dan pembuatan alarm waktu nyata)
- REL08-BP01 Menggunakan runbook untuk aktivitas standar seperti deployment

Dokumen terkait:

- Otomatisasi AWS Systems Manager
- Membuat Aturan EventBridge yang Memicu Peristiwa dari Sumber Daya AWS
- Lokakarya One Observability
- Amazon Builders' Library: Instrumentasi sistem terdistribusi untuk visibilitas operasional
- Apa itu Amazon DevOps Guru?
- Bekerja dengan Dokumen Otomatisasi (Buku Pedoman)

Video terkait:

- AWS re:Invent 2022 - Praktik terbaik observabilitas di Amazon
- AWS re:Invent 2020: Automate anything with AWS Systems Manager
- Introduction to AWS Resilience Hub
- Create Custom Ticket Systems for Amazon DevOps Guru Notifications
- Enable Multi-Account Insight Aggregation with Amazon DevOps Guru

Contoh terkait:

- Lokakarya Keandalan
- Amazon CloudWatch and Systems Manager Workshop

REL06-BP05 Analitik

Kumpulkan riwayat metrik dan file log dan analisis ini untuk mendapatkan wawasan beban kerja dan tren lebih luas.

Wawasan Amazon CloudWatch Logs mendukung bahasa kueri sederhana tapi luar biasa yang dapat Anda gunakan untuk menganalisis data log. Amazon CloudWatch Logs juga mendukung langganan yang memungkinkan data mengalir dengan lancar ke Amazon S3 di mana Anda

dapat menggunakannya atau Amazon Athena untuk kueri data. Amazon CloudWatch Logs juga mendukung kueri dengan berbagai macam format. Lihat Format Data dan SerDes yang didukung di Panduan Pengguna Amazon Athena untuk informasi selengkapnya. Untuk analisis set file log yang sangat besar, Anda dapat menjalankan klaster Amazon EMR untuk melakukan analisis skala petabita.

Ada sejumlah alat yang disediakan oleh Partner AWS dan pihak ketiga yang memungkinkan agregasi, pemrosesan, penyimpanan, dan analitik. Alat ini antara lain yakni New Relic, Splunk, Loggly, Logstash, CloudHealth, dan Nagios. Tetapi, generasi luar sistem dan log aplikasi bersifat unik untuk setiap penyedia cloud, dan sering kali unik untuk setiap layanan.

Bagian proses pemantauan yang sering kali tidak diperhatikan adalah manajemen data. Anda harus menentukan persyaratan retensi untuk memantau data, kemudian terapkan kebijakan siklus hidup yang sesuai. Amazon S3 mendukung manajemen siklus hidup di tingkat bucket S3. Manajemen siklus hidup ini dapat diterapkan secara berbeda ke jalur yang berbeda di bucket. Menjelang akhir siklus hidup, Anda dapat melakukan transisi data ke Amazon Glacier untuk penyimpanan jangka panjang, kemudian kedaluwarsa setelah akhir jangka waktu retensi tercapai. Kelas penyimpanan Bertingkat Cerdas S3 didesain untuk mengoptimalkan biaya dengan secara otomatis memindahkan data ke tingkat akses yang paling hemat biaya, tanpa memengaruhi performa atau tambahan biaya operasional.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Wawasan CloudWatch Logs memampukan Anda untuk secara interaktif mencari dan menganalisis data log Anda di Amazon CloudWatch Logs.
  - Menganalisis Data Log dengan Wawasan CloudWatch Logs
  - Kueri Sampel Wawasan Amazon CloudWatch Logs
- Gunakan Amazon CloudWatch Logs untuk mengirimkan log ke Amazon S3 di mana Anda dapat menggunakannya atau Amazon Athena untuk kueri data.
  - Bagaimana cara menganalisis log akses server Amazon S3 menggunakan Athena?
    - Buat kebijakan siklus hidup S3 untuk bucket log akses server Anda. Konfigurasikan kebijakan siklus hidup untuk secara berkala menghapus file log. Dengan melakukan tindakan ini, maka jumlah data yang dianalisis Athena untuk setiap kueri akan berkurang.
      - Bagaimana Cara Membuat Kebijakan Siklus Hidup untuk Bucket S3?

Sumber daya

Dokumen terkait:

- Kueri Sampel Wawasan Amazon CloudWatch Logs
- Menganalisis Data Log dengan Wawasan CloudWatch Logs
- Debugging dengan Amazon CloudWatch Synthetics and AWS X-Ray
- Bagaimana Cara Membuat Kebijakan Siklus Hidup untuk Bucket S3?
- Bagaimana cara menganalisis log akses server Amazon S3 menggunakan Athena?
- Satu Lokakarya Pengamatan
- Amazon Builders' Library: Menginstrumentasi sistem terdistribusi untuk visibilitas operasional

REL06-BP06 Lakukan peninjauan secara teratur

Sering kali tinjau bagaimana pemantauan beban kerja diimplementasikan dan perbarui berdasarkan perubahan dan peristiwa yang signifikan.

Pemantauan yang efektif didorong oleh metrik bisnis utama. Pastikan metrik-metrik ini diakomodasi di beban kerja Anda seiring dengan perubahan prioritas bisnis.

Mengaudit pemantauan Anda akan membantu memastikan Anda tahu kapan aplikasi memenuhi sasaran ketersediaannya. Analisis akar masalah memerlukan kemampuan untuk menemukan apa yang telah terjadi ketika ada kegagalan. AWS memberikan layanan yang memungkinkan Anda untuk melacak keadaan layanan Anda selama insiden:

- Amazon CloudWatch Logs: Anda dapat menyimpan log Anda di dalam layanan ini dan memeriksa kontennya.
- Wawasan Amazon CloudWatch Logs: Adalah layanan terkelola penuh yang memampukan Anda untuk menganalisis log yang sangat besar dalam hitungan detik. Layanan ini memberikan kepada Anda visualisasi dan kueri cepat dan interaktif.
- AWS Config: Anda dapat melihat infrastruktur AWS apa yang digunakan di berbagai titik waktu.
- AWS CloudTrail: Anda dapat melihat API AWS mana yang dipanggil pada waktu apa dan oleh prinsipal apa.

Di AWS, kami mengadakan rapat mingguan untuk meninjau performa operasional dan untuk berbagi pembelajaran antara tim. Karena ada begitu banyak tim di AWS, kami menciptakan Roda (The

Wheel) untuk secara acak memilih beban kerja yang akan ditinjau. Menetapkan irama yang teratur untuk peninjauan performa operasional dan berbagi pengetahuan meningkatkan kemampuan Anda untuk mencapai performa lebih tinggi dari tim operasional Anda.

Antipola umum:

- Hanya mengumpulkan metrik default.
- Menetapkan strategi pemantauan dan tidak pernah meninjaunya.
- Tidak membahas pemantauan ketika ada deployment perubahan besar.

Manfaat menerapkan praktik terbaik ini: Secara teratur meninjau pemantauan Anda memampukan antisipasi potensi masalah, dan bukannya bereaksi terhadap notifikasi ketika masalah yang diantisipasi sesungguhnya terjadi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Buat beberapa dasbor untuk beban kerja. Anda harus memiliki dasbor tingkat teratas yang berisi metrik bisnis utama, serta metrik teknis yang telah Anda identifikasi sebagai paling relevan untuk kondisi beban kerja yang diproyeksikan sesuai penggunaan yang bervariasi. Anda juga harus memiliki dasbor yang dapat diinspeksi untuk berbagai tingkat aplikasi dan ketergantungan.
  - Menggunakan Dasbor Amazon CloudWatch
- Jadwalkan dan lakukan peninjauan dasbor beban kerja secara teratur. Lakukan inspeksi dasbor secara teratur. Anda mungkin memiliki irama yang berbeda untuk kedalaman inspeksi Anda.
  - Inspeksi apakah ada tren dalam metrik. Bandingkan nilai metrik dengan nilai historis untuk melihat apakah ada tren yang mungkin menandakan bahwa sesuatu perlu diselidiki. Contohnya antara lain: meningkatkan latensi, menurunkan fungsi bisnis utama, dan meningkatkan respons kegagalan.
  - Inspeksi apakah ada penyimpangan/anomali dalam metrik Anda. Rerata atau median dapat menutupi penyimpangan dan anomali. Lihat nilai tertinggi dan nilai terendah dalam kerangka waktu dan selidiki penyebab skor yang ekstrem. Saat Anda terus mengeliminasi penyebab-penyebab ini, menurunkan definisi ekstrem akan memungkinkan Anda untuk terus meningkatkan konsistensi performa beban kerja Anda.
  - Cari perubahan mendadak dalam perilaku. Perubahan cepat dalam jumlah atau arah metrik dapat menandakan telah ada perubahan dalam aplikasi, atau ada faktor eksternal yang mungkin perlu Anda tambahkan metrik tambahan untuk dilacak.

Sumber daya

Dokumen terkait:

- Kueri Sampel Wawasan Amazon CloudWatch Logs
- Debugging dengan Amazon CloudWatch Synthetics and AWS X-Ray
- Satu Lokakarya Pengamatan
- Amazon Builders' Library: Menginstrumentasi sistem terdistribusi untuk visibilitas operasional
- Menggunakan Dasbor Amazon CloudWatch

REL06-BP07 Memantau pelacakan permintaan menyeluruh melalui sistem Anda

Lacak permintaan yang sedang diproses melalui komponen layanan agar tim produk dapat lebih mudah menganalisis dan menemukan serta memperbaiki masalah dan meningkatkan kinerja.

Hasil yang diinginkan: Beban kerja dengan penelusuran yang komprehensif di semua komponen memudahkan pencarian dan perbaikan masalah, sehingga meningkatkan rata-rata waktu penyelesaian (MTTR) kesalahan dan latensi dengan menyederhanakan penemuan akar masalah. Penelusuran yang menyeluruh akan mempersingkat waktu yang diperlukan untuk menemukan komponen yang terdampak dan mencari tahu akar masalah kesalahan atau latensi secara mendetail.

Antipola umum:

- Penelusuran digunakan untuk beberapa komponen, tidak semuanya. Misalnya, tanpa penelusuran untuk AWS Lambda, tim mungkin tidak memahami dengan jelas latensi yang disebabkan oleh cold start dalam beban kerja fluktuatif.
- Canary sintetis atau pemantauan pengguna nyata (RUM) tidak dikonfigurasi dengan penelusuran. Tanpa canary atau RUM, telemetri interaksi klien dihilangkan dari analisis jejak yang berimbas pada profil kinerja yang tidak lengkap.
- Beban kerja hybrid mencakup alat penelusuran cloud native dan pihak ketiga, tetapi langkah-langkah belum dilakukan untuk memilih dan sepenuhnya mengintegrasikan solusi penelusuran tunggal. Berdasarkan solusi penelusuran yang dipilih, SDK penelusuran cloud-native harus digunakan untuk melengkapi instrumen yang bukan cloud-native, atau alat pihak ketiga harus dikonfigurasi untuk menyerap telemetri pelacakan cloud-native.

Manfaat menjalankan praktik terbaik ini: Saat tim pengembangan menerima peringatan masalah, mereka dapat melihat gambaran utuh tentang interaksi komponen sistem, termasuk korelasi tiap

komponen dengan pembuatan log, kinerja, dan kegagalan. Karena penelusuran memudahkan identifikasi akar masalah secara visual, waktu penyelidikan akar masalah menjadi lebih singkat. Tim yang memahami interaksi komponen secara detail mengambil keputusan yang lebih baik dan lebih cepat saat menyelesaikan masalah. Keputusan seperti kapan harus memanggil failover pemulihan bencana (DR) atau lokasi terbaik untuk menerapkan strategi penyembuhan mandiri dapat ditingkatkan dengan menganalisis jejak sistem, dan pada akhirnya meningkatkan kepuasan pelanggan terhadap layanan Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Tim yang mengoperasikan aplikasi yang terdistribusi dapat menggunakan alat penelusuran untuk membuat pengidentifikasi korelasi, mengumpulkan jejak permintaan, dan membuat peta layanan komponen-komponen yang terhubung. Semua komponen aplikasi harus disertakan dalam jejak permintaan termasuk klien layanan, gateway perangkat lunak perantara (middleware) dan bus peristiwa, komponen komputasi, dan penyimpanan, termasuk penyimpanan nilai kunci dan basis data. Sertakan canary sintetis dan pemantauan pengguna nyata dalam konfigurasi penelusuran menyeluruh Anda untuk mengukur interaksi dan latensi klien jarak jauh sehingga Anda dapat secara akurat mengevaluasi kinerja sistem Anda berdasarkan perjanjian dan tujuan tingkat layanan Anda.

Anda dapat menggunakan layanan instrumentasi AWS X-Ray dan Pemantauan Aplikasi Amazon CloudWatch untuk memberikan tampilan utuh permintaan yang diproses melalui aplikasi Anda. X-Ray mengumpulkan telemetri aplikasi dan memungkinkan Anda untuk memvisualisasikan dan menyaringnya di seluruh muatan, fungsi, jejak, layanan, API, dan dapat diaktifkan untuk komponen sistem, dengan rendah kode atau tanpa kode. Pemantauan aplikasi CloudWatch mencakup ServiceLens untuk mengintegrasikan jejak Anda dengan metrik, log, dan alarm. Pemantauan aplikasi CloudWatch juga mencakup Syntethics untuk memantau titik akhir dan API Anda, serta pemantauan pengguna nyata untuk melengkapi klien aplikasi web Anda.

Langkah implementasi

- Gunakan AWS X-Ray pada semua layanan native yang didukung seperti Amazon S3, AWS Lambda, dan Amazon API Gateway. Semua layanan AWS ini mengaktifkan X-Ray dengan tombol konfigurasi menggunakan infrastruktur sebagai kode, SDK AWS, atau Konsol Manajemen AWS.
- Aplikasi instrumen AWS Distro for Open Telemetry dan X-Ray atau agen pengumpulan pihak ketiga.

- Tinjau Panduan AWS X-Ray untuk Pengembang untuk implementasi bahasa pemrograman khusus. Bagian dokumentasi ini menjelaskan cara menginstrumentasi permintaan HTTP, kueri SQL, dan proses lain yang spesifik untuk bahasa pemrograman aplikasi Anda.
- Gunakan penelusuran X-Ray untuk Amazon CloudWatch Synthetic Canaries dan Amazon CloudWatch RUM untuk menganalisis jalur permintaan dari klien pengguna akhir Anda melalui infrastruktur AWS hilir Anda.
- Konfigurasikan metrik dan alarm CloudWatch berdasarkan telemetri canary dan kesehatan sumber daya sehingga tim menerima peringatan masalah dengan cepat, kemudian dapat mempelajari jejak dan peta layanan dengan ServiceLens.
- Aktifkan integrasi X-Ray untuk alat penelusuran pihak ketiga seperti Datadog, New Relic, atau Dynatrace jika Anda menggunakan alat pihak ketiga untuk solusi penelusuran utama Anda.

Sumber daya

Praktik terbaik terkait:

- REL06-BP01 Memantau semua komponen untuk beban kerja (Pembuatan)
- REL11-BP01 Memantau semua komponen beban kerja untuk mendeteksi kegagalan

Dokumen terkait:

- Apa itu AWS X-Ray?
- Amazon CloudWatch: Pemantauan Aplikasi
- Melakukan debug dengan Amazon CloudWatch Synthetics dan AWS X-Ray
- Amazon Builders' Library: Instrumentasi sistem terdistribusi untuk visibilitas operasional
- Mengintegrasikan AWS X-Ray dengan layanan AWS lain
- AWS Distro for OpenTelemetry dan AWS X-Ray
- Amazon CloudWatch: Menggunakan pemantauan sintetis
- Amazon CloudWatch: Gunakan CloudWatch RUM
- Mengatur canary sintetis Amazon CloudWatch dan alarm Amazon CloudWatch
- Ketersediaan dan Lainnya: Memahami dan Meningkatkan Ketangguhan Sistem Terdistribusi di AWS

Contoh terkait:

- Lokakarya One Observability

Video terkait:

- AWS re:Invent 2022 - Cara memantau aplikasi di beberapa akun
- Cara Memantau Aplikasi AWS Anda

Alat terkait:

- AWS X-Ray
- Amazon CloudWatch
- Amazon Route 53

REL 7. Bagaimana cara mendesain beban kerja Anda untuk beradaptasi dengan perubahan dalam permintaan?

Beban kerja yang dapat diskalakan memberikan elastisitas untuk menambahkan atau mengeluarkan sumber daya secara otomatis sehingga sangat sesuai dengan permintaan saat ini pada titik waktu tertentu.

Praktik terbaik

- REL07-BP01 Menggunakan otomatisasi ketika mendapatkan atau menskalakan sumber daya
- REL07-BP02 Mendapatkan sumber daya setelah deteksi gangguan pada beban kerja
- REL07-BP03 Menambah sumber daya berdasarkan deteksi bahwa beban kerja memerlukan lebih banyak sumber daya
- REL07-BP04 Menguji beban untuk beban kerja Anda

REL07-BP01 Menggunakan otomatisasi ketika mendapatkan atau menskalakan sumber daya

Ketika mengganti sumber daya yang terganggu atau menskalakan beban kerja Anda, otomatiskan proses menggunakan AWS Managed Services (AMS), seperti Amazon S3 dan AWS Auto Scaling. Anda juga dapat menggunakan alat pihak ketiga dan SDK AWS untuk mengotomatiskan penskalaan.

AWS Managed Services mencakup Amazon S3, Amazon CloudFront, AWS Auto Scaling, AWS Lambda, Amazon DynamoDB, AWS Fargate, dan Amazon Route 53.

Dengan AWS Auto Scaling, Anda dapat mendeteksi dan mengganti instans yang terganggu. Dengan ini, Anda juga dapat membuat rencana penskalaan untuk sumber daya, termasuk instans Amazon EC2 dan Armada Spot, tugas Amazon ECS , tabel dan indeks Amazon DynamoDB , serta Replika Amazon Aurora .

Saat menskalakan instans EC2, pastikan bahwa Anda menggunakan Zona Ketersediaan (disarankan minimal tiga) serta menambahkan atau menghapus kapasitas untuk menjaga keseimbangan di seluruh Zona Ketersediaan ini. Tugas ECS atau pod Kubernetes (saat menggunakan Amazon Elastic Kubernetes Service) juga harus didistribusikan ke beberapa Zona Ketersediaan.

Ketika menggunakan AWS Lambda, instans menskalakan secara otomatis. Setiap kali notifikasi diterima untuk fungsi Anda, AWS Lambda langsung mencari kapasitas bebas di dalam armada komputasinya, serta menjalankan kode Anda sesuai konkurensi yang dialokasikan. Anda harus memastikan bahwa konkurensi yang diperlukan telah dikonfigurasikan di Lambda tertentu, dan di dalam Service Quotas.

Amazon S3 diskalakan secara otomatis untuk menangani tingkat permintaan tinggi. Misalnya, aplikasi Anda dapat memenuhi minimum 3.500 permintaan PUT/COPY/POST/DELETE atau 5.500 permintaan GET/HEAD per detik per prefiks dalam bucket. Tidak ada batasan jumlah prefiks dalam bucket. Anda dapat meningkatkan kinerja baca atau tulis Anda dengan memparalelkan pembacaan. Misalnya, jika Anda membuat 10 prefiks dalam sebuah bucket Amazon S3 untuk memparalelkan pembacaan, Anda dapat menskalakan kinerja baca Anda hingga 55.000 permintaan baca per detik.

Konfigurasikan dan gunakan Amazon CloudFront atau jaringan pengiriman konten (CDN) tepercaya. CDN dapat memberikan waktu respons pengguna akhir yang lebih cepat untuk konten dari cache, sehingga mengurangi kebutuhan untuk menskalakan beban kerja Anda.

Antipola umum:

- Mengimplementasikan grup Auto Scaling untuk pemulihan otomatis, tetapi tidak mengimplementasikan elastisitas.
- Menggunakan penskalaan otomatis untuk merespons peningkatan yang signifikan di lalu lintas.
- Melakukan deployment aplikasi yang sangat stateful, menghilangkan opsi elastisitas.

Manfaat menerapkan praktik terbaik ini: Otomatisasi menghilangkan potensi kesalahan manual dalam melakukan deployment dan penonaktifan sumber daya. Otomatisasi menghilangkan risiko pembengkakan biaya dan penolakan layanan akibat lambatnya respons saat dibutuhkan untuk melakukan deployment atau penonaktifan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Konfigurasikan dan gunakan AWS Auto Scaling. Ini memantau aplikasi Anda dan secara otomatis menyesuaikan kapasitas untuk mempertahankan kinerja yang stabil dan dapat diprediksi dengan biaya serendah mungkin. Menggunakan AWS Auto Scaling, Anda dapat mengonfigurasi penskalaan aplikasi untuk beberapa sumber daya di beberapa layanan.
  - Apa itu AWS Auto Scaling?
    - Konfigurasikan Penskalaan Otomatis dalam instans Amazon EC2 dan Armada Spot, tugas Amazon ECS, tabel dan indeks Amazon DynamoDB, Replika Amazon Aurora, serta perangkat AWS Marketplace Anda sebagai dapat diterapkan.
      - Mengelola kapasitas throughput secara otomatis dengan Penskalaan Otomatis DynamoDB.
        - Gunakan operasi API layanan untuk menentukan peringatan, kebijakan penskalaan, waktu warm up, dan waktu cool down.
- Gunakan Elastic Load Balancing. Penyeimbang beban dapat mendistribusikan beban berdasarkan jalur atau berdasarkan konektivitas jaringan.
  - Apa itu Elastic Load Balancing?
    - Application Load Balancers dapat mendistribusikan beban berdasarkan jalur.
      - Apa itu Application Load Balancer?
        - Konfigurasikan Application Load Balancer untuk mendistribusikan lalu lintas ke berbagai beban kerja berdasarkan nama domain.
        - Application Load Balancers dapat digunakan untuk mendistribusikan beban dengan cara yang terintegrasi dengan AWS Auto Scaling untuk mengelola permintaan.
          - Menggunakan penyeimbang beban dengan grup Auto Scaling.
    - Penyeimbang Beban Jaringan dapat mendistribusikan beban berdasarkan koneksi.
      - Apa itu Penyeimbang Beban Jaringan?
        - Konfigurasikan Penyeimbang Beban Jaringan untuk mendistribusikan lalu lintas ke berbagai beban kerja menggunakan TCP, atau untuk mendapatkan set konstan alamat IP untuk beban kerja Anda.
        - Penyeimbang Beban Jaringan dapat digunakan untuk mendistribusikan beban yang terintegrasi dengan AWS Auto Scaling untuk mengelola permintaan.
- Gunakan penyedia DNS dengan ketersediaan tinggi. Nama DNS memungkinkan pengguna Anda untuk memasukkan nama tanpa perlu memasukkan alamat IP guna mengakses beban kerja Anda

dan mendistribusikan informasi ini ke cakupan yang ditentukan, biasanya untuk pengguna beban kerja secara global.

- Gunakan Amazon Route 53 atau penyedia DNS tepercaya.
  - Apa itu Amazon Route 53?
- Gunakan Route 53 untuk mengelola penyeimbang beban dan distribusi CloudFront Anda.
  - Tentukan domain dan subdomain yang akan Anda kelola.
  - Buat set catatan yang sesuai menggunakan catatan ALIAS atau CNAME.
    - Bekerja dengan catatan

- Gunakan jaringan global AWS untuk optimasi jalur pengguna Anda ke aplikasi. AWS Global Accelerator memantau kondisi titik akhir aplikasi Anda secara terus-menerus dan mengalihkan lalu lintas ke titik akhir yang sehat dalam kurang dari 30 detik.
  - AWS Global Accelerator adalah layanan yang meningkatkan ketersediaan dan kinerja aplikasi Anda untuk pengguna global atau lokal. Ini menyediakan alamat IP statis yang berperan sebagai titik entri tetap ke titik akhir aplikasi Anda dalam satu atau beberapa Wilayah AWS, seperti Application Load Balancers, Penyeimbang Beban Jaringan, atau instans Amazon EC2 Anda.
    - Apa Itu AWS Global Accelerator?
- Konfigurasikan dan gunakan Amazon CloudFront atau jaringan pengiriman konten (CDN) tepercaya. Jaringan pengiriman konten dapat memberikan waktu respons pengguna akhir yang lebih cepat serta memenuhi permintaan konten yang dapat mengakibatkan penskalaan yang tidak perlu dari beban kerja Anda.
  - Apa itu Amazon CloudFront?
    - Konfigurasikan distribusi Amazon CloudFront untuk beban kerja Anda, atau gunakan CDN pihak ketiga.
      - Anda dapat membatasi akses ke beban kerja Anda agar hanya dapat diakses dari CloudFront menggunakan rentang IP untuk CloudFront dalam grup keamanan titik akhir atau kebijakan akses Anda.

Sumber daya

Dokumen terkait:

- Partner APN: partner yang dapat membantu Anda membuat solusi komputasi yang diotomatiskan.
- AWS Auto Scaling: Cara Kerja Rencana Penskalaan
- AWS Marketplace: produk yang dapat digunakan dengan penskalaan otomatis

- Mengelola Kapasitas Throughput Secara Otomatis dengan Penskalaan Otomatis DynamoDB.
- Menggunakan penyeimbang beban dengan grup Auto Scaling.
- Apa Itu AWS Global Accelerator?
- Apa Itu Amazon EC2 Auto Scaling?
- Apa itu AWS Auto Scaling?
- Apa itu Amazon CloudFront?
- Apa itu Amazon Route 53?
- Apa itu Elastic Load Balancing?
- Apa itu Penyeimbang Beban Jaringan?
- Apa itu Application Load Balancer?
- Bekerja dengan catatan

REL07-BP02 Mendapatkan sumber daya setelah deteksi gangguan pada beban kerja

Skalakan sumber daya secara reaktif saat diperlukan jika ketersediaan terganggu, guna memulihkan ketersediaan beban kerja.

Anda terlebih dahulu harus mengonfigurasi pemeriksaan kondisi dan kriteria pada pemeriksaan ini agar memberikan penanda saat ketersediaan terganggu oleh kurangnya sumber daya. Lalu, beri tahu personel yang bersangkutan untuk menskalakan sumber daya secara manual, atau mulai otomatisasi untuk menskalakannya secara otomatis.

Skala dapat disesuaikan secara manual untuk beban kerja Anda (misalnya, mengubah jumlah instans EC2 di grup Auto Scaling atau modifikasi throughput tabel DynamoDB melalui Konsol Manajemen AWS atau AWS CLI). Namun, otomatisasi harus digunakan apabila memungkinkan (lihat Menggunakan otomatisasi ketika mendapatkan atau menskalakan sumber daya).

Hasil yang diinginkan: Aktivitas penskalaan (baik secara otomatis maupun manual) diinisiasi untuk memulihkan ketersediaan setelah terdeteksinya kegagalan atau menurunnya pengalaman pelanggan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Terapkan observabilitas dan pemantauan di semua komponen dalam beban kerja Anda untuk memantau pengalaman pelanggan dan mendeteksi kegagalan. Tentukan prosedur, manual atau

otomatis, yang menskalakan sumber daya yang diperlukan. Untuk informasi lebih lanjut, lihat REL11-BP01 Memantau semua komponen beban kerja untuk mendeteksi kegagalan.

Langkah implementasi

- Tentukan prosedur, manual atau otomatis, yang menskalakan sumber daya yang dibutuhkan.
  - Prosedur penskalaan tergantung pada bagaimana rancangan berbagai komponen dalam beban kerja Anda.
  - Prosedur penskalaan juga bervariasi, tergantung teknologi dasar yang digunakan.
    - Komponen yang menggunakan AWS Auto Scaling dapat menggunakan rencana penskalaan untuk mengonfigurasi serangkaian instruksi guna menskalakan sumber daya Anda. Jika Anda bekerja dengan AWS CloudFormation atau menambahkan tag ke sumber daya AWS, Anda dapat menyiapkan rencana penskalaan untuk berbagai set sumber daya per aplikasi. Auto Scaling menyediakan saran strategi penskalaan yang disesuaikan untuk tiap-tiap sumber daya. Setelah Anda membuat rencana penskalaan, Auto Scaling menggabungkan metode penskalaan dinamis dan penskalaan prediktif untuk mendukung strategi penskalaan Anda. Untuk detail selengkapnya, lihat Cara kerja rencana penskalaan.
    - Amazon EC2 Auto Scaling memverifikasi bahwa jumlah instans Amazon EC2 Anda yang tersedia sudah tepat untuk menangani beban aplikasi Anda. Anda membuat koleksi instans EC2, yang disebut grup Auto Scaling. Anda dapat menentukan jumlah instans minimum dan maksimum di setiap grup Auto Scaling, dan Amazon EC2 Auto Scaling memastikan bahwa grup Anda tidak pernah berada di bawah atau di atas batas ini. Untuk lebih jelasnya, lihat Apa itu Amazon EC2 Auto Scaling?
    - Penskalaan otomatis Amazon DynamoDB menggunakan layanan Application Auto Scaling untuk menyesuaikan secara dinamis kapasitas throughput yang disediakan atas nama Anda, sebagai respons terhadap pola lalu lintas aktual. Ini memungkinkan tabel atau indeks sekunder global untuk meningkatkan kapasitas baca dan tulis yang disediakan untuk menangani peningkatan lalu lintas yang mendadak, tanpa throttling. Untuk detail selengkapnya, lihat Mengelola kapasitas throughput secara otomatis dengan penskalaan otomatis DynamoDB.

Sumber daya

Praktik Terbaik Terkait:

- REL07-BP01 Menggunakan otomatisasi ketika mendapatkan atau menskalakan sumber daya
- REL11-BP01 Memantau semua komponen beban kerja untuk mendeteksi kegagalan

Dokumen terkait:

- AWS Auto Scaling: Cara Kerja Rencana Penskalaan
- Mengelola Kapasitas Throughput Secara Otomatis dengan Penskalaan Otomatis DynamoDB
- Apa Itu Amazon EC2 Auto Scaling?

REL07-BP03 Menambah sumber daya berdasarkan deteksi bahwa beban kerja memerlukan lebih banyak sumber daya

Skalakan sumber daya secara proaktif untuk memenuhi permintaan dan menghindari dampak ketersediaan.

Banyak layanan AWS yang melakukan penskalaan secara otomatis untuk memenuhi permintaan. Dengan menggunakan instans Amazon EC2 atau klaster Amazon ECS, Anda dapat mengonfigurasikan penskalaan otomatis ini agar muncul berdasarkan penggunaan metrik yang sesuai dengan permintaan untuk beban kerja Anda. Untuk Amazon EC2, rata-rata pemanfaatan CPU, jumlah permintaan penyeimbang beban, atau bandwidth jaringan dapat digunakan untuk menskalakan ke luar (atau menskalakan ke dalam) instans EC2. Untuk Amazon ECS, rata-rata pemanfaatan CPU, jumlah permintaan penyeimbang beban, dan pemanfaatan memori dapat digunakan untuk menskalakan ke luar (atau menskalakan ke dalam) tugas ECS. Menggunakan Penskalaan Otomatis Target di AWS, penskala otomatis berperan seperti termostat, yang menambahkan atau menghapus sumber daya untuk mempertahankan nilai target (misalnya, 70% pemanfaatan CPU) yang Anda tentukan.

AWS Auto Scaling juga dapat melakukan Penskalaan Otomatis Prediktif, yang menggunakan machine learning untuk menganalisis setiap beban kerja historis sumber daya dan memperkirakan beban untuk dua hari mendatang secara rutin.

Little’s Law membantu menghitung banyaknya instans komputasi (instans EC2, fungsi Lambda bersamaan, dll.) yang Anda butuhkan.

$L = \lambda W$

L = jumlah instans (atau konkurensi nilai tengah dalam sistem)

λ = rasio rata-rata permintaan yang diterima (permintaan/detik)

W = waktu rata-rata yang diperlukan setiap permintaan di dalam sistem (detik)

Misalnya, dengan laju 100 rps (permintaan per detik), jika setiap permintaan memerlukan 0,5 detik untuk diproses, Anda akan memerlukan 50 instans untuk memenuhi permintaan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Tambahkan sumber daya berdasarkan deteksi bahwa beban kerja memerlukan lebih banyak sumber daya. Skalakan sumber daya secara proaktif untuk memenuhi permintaan dan menghindari dampak ketersediaan.
  - Hitung banyaknya sumber daya komputasi yang akan Anda butuhkan (konkurensi komputasi) untuk menangani rasio permintaan tertentu.
    - Seputar Little's Law
  - Jika Anda memiliki pola historis untuk penggunaan, atur penskalaan terjadwal untuk penskalaan otomatis Amazon EC2.
    - Penskalaan Terjadwal untuk Amazon EC2 Auto Scaling
  - Gunakan penskalaan prediktif AWS.
    - Penskalaan Prediktif untuk EC2, Didukung oleh Machine Learning

Sumber daya

Dokumen terkait:

- AWS Auto Scaling: Cara Kerja Rencana Penskalaan
- AWS Marketplace: produk yang dapat digunakan dengan penskalaan otomatis
- Mengelola Kapasitas Throughput Secara Otomatis dengan Penskalaan Otomatis DynamoDB.
- Penskalaan Prediktif untuk EC2, Didukung oleh Machine Learning
- Penskalaan Terjadwal untuk Amazon EC2 Auto Scaling
- Seputar Little's Law
- Apa Itu Amazon EC2 Auto Scaling?

REL07-BP04 Menguji beban untuk beban kerja Anda

Adopsi metodologi pengujian beban untuk mengukur apakah aktivitas penskalaan memenuhi persyaratan beban kerja.

Pengujian beban yang berkelanjutan penting untuk dilakukan. Pengujian beban harus menemukan titik nadir dan menguji kinerja beban kerja Anda. AWS memudahkan penyiapan lingkungan pengujian sementara yang memodelkan skala beban kerja produksi Anda. Di cloud, Anda dapat membuat lingkungan pengujian berskala produksi sesuai permintaan, menyelesaikan pengujian, kemudian menonaktifkan sumber dayanya. Karena Anda hanya membayar lingkungan pengujian saat sedang berjalan, Anda dapat menyimulasikan lingkungan langsung Anda dengan biaya yang lebih murah daripada pengujian on-premise.

Pengujian beban di produksi juga harus dipertimbangkan sebagai bagian dari aktivitas game day di mana sistem produksi diberikan tekanan, selama jam-jam penggunaan pelanggan yang lebih rendah, dengan semua personel siap menerjemahkan hasilnya dan menangani masalah yang muncul.

Antipola umum:

- Melakukan pengujian beban di lingkungan deployment yang tidak memiliki konfigurasi yang sama dengan produksi Anda.
- Melakukan pengujian beban hanya pada beban kerja Anda secara terpisah-pisah, bukan pada keseluruhan beban kerja.
- Melakukan pengujian beban dengan subset permintaan, bukan set permintaan riil yang representatif.
- Melakukan pengujian beban ke faktor keselamatan kecil di atas beban yang diharapkan.

Manfaat menjalankan praktik terbaik ini: Anda mengetahui komponen apa saja di dalam arsitektur Anda yang gagal saat menerima beban, dan mampu mengidentifikasi metrik apa saja yang perlu diamati sebagai indikator bahwa Anda mendekati beban tersebut tepat waktu untuk mengatasi masalah dan mencegah dampak kegagalan tersebut.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

- Lakukan pengujian beban untuk mengidentifikasi aspek dalam beban kerja Anda yang menunjukkan bahwa Anda harus menambah atau menghapus kapasitas. Pengujian beban harus memiliki lalu lintas representatif yang serupa dengan yang Anda terima di lingkungan produksi. Tingkatkan beban sambil mengamati metrik yang telah Anda instrumentasi untuk menentukan metrik mana yang menunjukkan kapan Anda harus menambah atau menghapus sumber daya.
  - Pengujian Beban Terdistribusi di AWS: simulasikan ribuan pengguna terhubung

- Identifikasi gabungan permintaan. Anda mungkin memiliki gabungan permintaan yang beragam, sehingga Anda harus melihat berbagai kerangka waktu saat mengidentifikasi gabungan lalu lintas.
- Implementasikan pendorong beban. Anda dapat menggunakan aplikasi kode kustom, sumber terbuka, atau komersial untuk mengimplementasikan pendorong beban.
- Lakukan uji beban di awal menggunakan kapasitas kecil. Anda melihat beberapa dampak langsung dengan mendorong beban ke kapasitas yang lebih kecil, kemungkinan seukuran satu instans atau kontainer.
- Uji beban dengan kapasitas yang lebih besar. Efek akan berbeda di beban yang terdistribusi, sehingga Anda harus menguji di lingkungan yang semirip mungkin dengan lingkungan produksi.

Sumber daya

Dokumen terkait:

- Pengujian Beban Terdistribusi di AWS: simulasikan ribuan pengguna terhubung

REL 8. Bagaimana cara mengimplementasikan perubahan?

Perubahan terkontrol diperlukan untuk melakukan deployment fungsionalitas baru, dan untuk memverifikasi bahwa beban kerja dan lingkungan operasi menjalankan perangkat lunak yang dikenal dan dapat di-patch atau diganti dengan cara yang dapat diprediksi. Jika perubahan-perubahan ini tidak terkontrol, maka akan sulit untuk memprediksi efek dari perubahan-perubahan tersebut, atau untuk mengatasi masalah yang timbul sebagai akibatnya.

Praktik terbaik

- REL08-BP01 Menggunakan runbook untuk aktivitas standar seperti deployment
- REL08-BP02 Integrasikan pengujian fungsional sebagai bagian dari deployment Anda
- REL08-BP03 Mengintegrasikan pengujian ketahanan sebagai bagian dari deployment Anda
- REL08-BP04 Melakukan deployment menggunakan infrastruktur tetap
- REL08-BP05 Melakukan deployment perubahan dengan otomatisasi

REL08-BP01 Menggunakan runbook untuk aktivitas standar seperti deployment

Runbook adalah prosedur terdokumentasi untuk mencapai hasil tertentu. Gunakan runbook untuk melakukan aktivitas standar, baik yang dilakukan secara manual maupun otomatis. Contohnya adalah men-deploy beban kerja, mem-patch beban kerja, atau membuat modifikasi DNS.

Misalnya, terapkan proses untuk memastikan keamanan pembatalan selama deployment. Memastikan bahwa Anda dapat membatalkan deployment tanpa gangguan terhadap pelanggan adalah sesuatu yang penting dalam menciptakan keandalan layanan.

Untuk prosedur runbook, mulailah dengan proses manual efektif yang valid, implementasikan dalam kode, dan picu agar berjalan secara otomatis saat diperlukan.

Bahkan untuk beban kerja canggih yang diotomatiskan dalam tingkat tinggi, runbook tetap bermanfaat untuk menjalankan aktivitas game day atau memenuhi persyaratan pelaporan dan audit yang ketat.

Ingat bahwa buku pedoman digunakan untuk merespons insiden tertentu, sedangkan runbook digunakan untuk mencapai hasil tertentu. Sering kali, runbook ditujukan untuk aktivitas rutin, sedangkan buku pedoman digunakan untuk merespons peristiwa nonrutin.

Antipola umum:

- Melakukan perubahan tidak terencana pada konfigurasi di lingkungan produksi.
- Melewatkan langkah-langkah dalam rencana Anda untuk men-deploy lebih cepat, sehingga mengakibatkan kegagalan deployment.
- Membuat perubahan tanpa menguji pembatalan perubahan.

Manfaat menjalankan praktik terbaik ini: Perencanaan perubahan yang efektif meningkatkan kemampuan Anda untuk berhasil mengeksekusi perubahan karena Anda mengetahui semua sistem yang terpengaruh. Validasi perubahan di lingkungan pengujian meningkatkan kepercayaan diri Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

- Aktifkan respons yang cepat dan konsisten terhadap peristiwa yang dipahami dengan baik dengan cara mendokumentasikan prosedur di dalam runbook.
  - AWS Well-Architected Framework: Konsep: Runbook

- Gunakan prinsip infrastruktur sebagai kode untuk menetapkan infrastruktur Anda. Dengan menggunakan AWS CloudFormation (atau pihak ketiga tepercaya) untuk menetapkan infrastruktur Anda, Anda dapat menggunakan perangkat lunak kontrol versi untuk membuat versi baru dan melacak perubahan.
  - Gunakan AWS CloudFormation (atau penyedia pihak ketiga tepercaya) untuk menetapkan infrastruktur Anda.
    - Apa itu AWS CloudFormation?
  - Buat templat singular dan terpisah-pisah, menggunakan prinsip desain perangkat lunak yang baik.
    - Tentukan izin, templat, dan pihak-pihak yang bertanggung jawab untuk implementasi.
      - Mengontrol akses dengan AWS Identity and Access Management.
    - Gunakan kontrol sumber, seperti AWS CodeCommit atau alat pihak ketiga tepercaya, untuk kontrol versi.
      - Apa Itu AWS CodeCommit?

Sumber daya

Dokumen terkait:

- Partner APN: partner yang dapat membantu Anda membuat solusi deployment yang diotomatisasi
- AWS Marketplace: produk yang dapat digunakan untuk mengotomatisasi deployment Anda
- AWS Well-Architected Framework: Konsep: Runbook
- Apa itu AWS CloudFormation?
- Apa Itu AWS CodeCommit?

Contoh terkait:

- Mengotomatiskan operasi dengan Buku Pedoman dan Runbook

REL08-BP02 Integrasikan pengujian fungsional sebagai bagian dari deployment Anda

Uji fungsional dijalankan sebagai bagian dari deployment otomatis. Jika kriteria untuk sukses tidak terpenuhi, maka alur akan dihentikan atau dikembalikan.

Pengujian ini dijalankan dalam lingkungan praproduksi, yang dilaksanakan sebelum perkembangan produksi. Idealnya, ini dilakukan sebagai bagian dari alur deployment.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Integrasikan pengujian fungsional sebagai bagian dari deployment Anda. Uji fungsional dijalankan sebagai bagian dari deployment otomatis. Jika kriteria untuk sukses tidak terpenuhi, maka alur akan dihentikan atau dikembalikan.
  - Panggil AWS CodeBuild selama 'Tindakan Pengujian' dari alur rilis perangkat lunak Anda yang dimodelkan di AWS CodePipeline. Kemampuan ini memungkinkan Anda untuk dengan mudah menjalankan berbagai macam pengujian terhadap kode Anda, seperti uji unit, analisis kode statis, dan uji integrasi.
    - AWS CodePipeline Menambahkan Dukungan untuk Unit dan Pengujian Integrasi Kustom dengan AWS CodeBuild
  - Gunakan solusi AWS Marketplace untuk melaksanakan pengujian otomatis sebagai bagian dari alur hasil pengiriman perangkat lunak Anda.
    - Otomatisasi uji perangkat lunak

Sumber daya

Dokumen terkait:

- AWS CodePipeline Menambahkan Dukungan untuk Unit dan Pengujian Integrasi Kustom dengan AWS CodeBuild
- Otomatisasi uji perangkat lunak
- Apa Itu AWS CodePipeline?

REL08-BP03 Mengintegrasikan pengujian ketahanan sebagai bagian dari deployment Anda

Pengujian ketahanan (menggunakan prinsip-prinsip chaos engineering) dijalankan sebagai bagian dari pipeline deployment otomatis dalam lingkungan praproduksi.

Pengujian tersebut dilaksanakan dan dijalankan di lingkungan praproduksi. Pengujian harus dijalankan dalam produksi sebagai bagian dari game day.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Integrasikan pengujian ketahanan sebagai bagian dari deployment Anda. Gunakan Chaos Engineering, bidang ilmu yang bereksperimen pada sistem guna membangun kepercayaan pada kemampuan beban kerja untuk menahan kondisi turbulen dalam produksi.
  - Pengujian ketahanan memasukkan kesalahan atau degradasi sumber daya untuk menilai apakah beban kerja merespons dengan desain ketahanannya.
    - Lab Well-Architected: Level 300: Pengujian Ketahanan EC2, RDS, dan S3
  - Pengujian ini dapat dijalankan secara rutin di lingkungan praproduksi dalam pipeline deployment otomatis.
  - Pengujian harus dijalankan dalam produksi sebagai bagian dari game day terjadwal.
  - Dengan menggunakan prinsip-prinsip Chaos Engineering, ajukan hipotesis tentang cara beban kerja bekerja di berbagai gangguan, kemudian uji hipotesis dengan menggunakan pengujian ketahanan.
    - Prinsip-prinsip Chaos Engineering

Sumber daya

Dokumen terkait:

- Prinsip-prinsip Chaos Engineering
- Apa itu Simulator Injeksi Kesalahan AWS?

Contoh terkait:

- Lab Well-Architected: Level 300: Pengujian Ketahanan EC2, RDS, dan S3

REL08-BP04 Melakukan deployment menggunakan infrastruktur tetap

Infrastruktur tetap adalah model yang menuntut bahwa tidak ada pembaruan, patch keamanan, atau perubahan konfigurasi yang terjadi di tempat pada beban kerja produksi. Saat perubahan diperlukan, arsitektur dibangun ke infrastruktur baru dan di-deploy ke dalam produksi.

Ikuti strategi penerapan infrastruktur tetap untuk meningkatkan keandalan, konsistensi, dan keterulangan dalam deployment beban kerja Anda.

Hasil yang diinginkan: Dengan infrastruktur tetap, tidak ada modifikasi di tempat yang diizinkan untuk menjalankan sumber daya infrastruktur dalam beban kerja. Sebaliknya, ketika perubahan diperlukan, kumpulan sumber daya infrastruktur baru yang diperbarui, yang berisi semua perubahan yang diperlukan, di-deploy secara paralel dengan sumber daya Anda yang ada. Deployment ini divalidasi secara otomatis, dan jika berhasil, lalu lintas dialihkan secara bertahap ke kumpulan sumber daya baru.

Strategi deployment ini berlaku di antaranya untuk pembaruan perangkat lunak, patch keamanan, perubahan infrastruktur, pembaruan konfigurasi, dan pembaruan aplikasi.

Antipola umum:

- Menerapkan perubahan di tempat untuk menjalankan sumber daya infrastruktur.

Manfaat menjalankan praktik terbaik ini:

- Meningkatnya konsistensi di seluruh lingkungan: Karena tidak ada perbedaan sumber daya infrastruktur di seluruh lingkungan, konsistensi meningkat dan pengujian menjadi lebih sederhana.
- Berkurangnya penyimpangan konfigurasi: Dengan mengganti sumber daya infrastruktur dengan konfigurasi yang diketahui dan dikontrol versinya, infrastruktur diatur ke status yang diketahui, diuji, dan tepercaya, sehingga menghindari penyimpangan konfigurasi.
- Deployment atomik yang dapat diandalkan: Deployment hanya berujung pada dua hal: berhasil diselesaikan atau tidak ada perubahan, sehingga konsistensi dan keandalan dalam proses deployment meningkat.
- Deployment yang disederhanakan: Deployment disederhanakan karena tidak memerlukan pembaruan dukungan. Pembaruan hanyalah deployment baru.
- Deployment yang lebih aman dengan proses rollback dan pemulihan yang cepat: Deployment lebih aman karena versi kerja sebelumnya tidak berubah. Anda dapat melakukan rollback jika kesalahan terdeteksi.
- Postur keamanan yang lebih baik: Karena perubahan pada infrastruktur tidak diizinkan, mekanisme akses jarak jauh (seperti SSH) dapat dinonaktifkan. Hal ini mengurangi vektor serangan, sehingga meningkatkan postur keamanan organisasi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Automasi

Saat menentukan strategi penyebaran infrastruktur tetap, sebaiknya gunakan otomatisasi sebanyak mungkin untuk meningkatkan keterulangan dan meminimalkan potensi kesalahan manusia. Untuk detail selengkapnya, lihat REL08-BP05 Melakukan deployment perubahan dengan otomatisasi dan Mengotomatiskan deployment aman tanpa campur tangan.

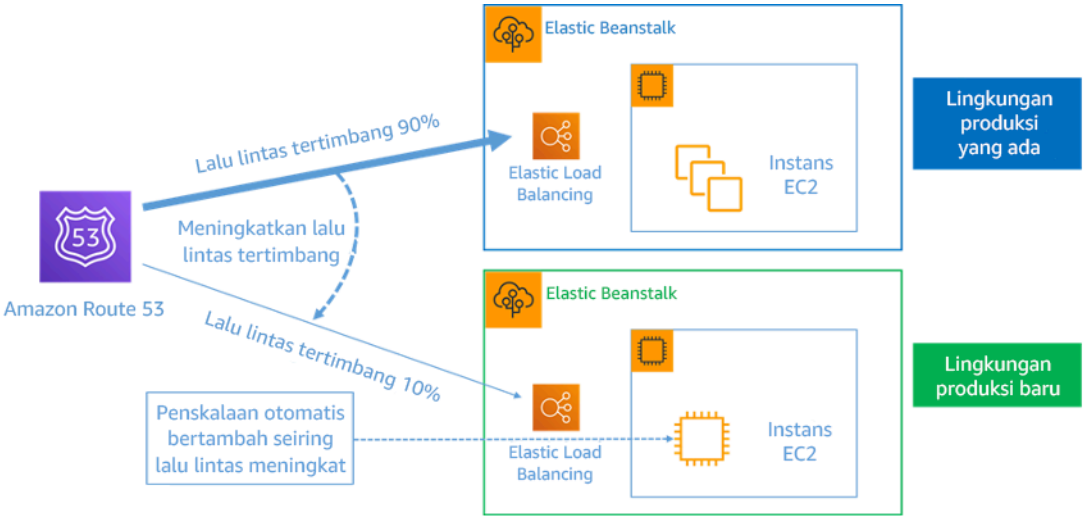
Dengan Infrastructure sebagai Kode (IaC), langkah-langkah penyediaan infrastruktur, orkestrasi, dan deployment ditentukan dalam cara yang terprogram, deskriptif, dan deklaratif dan disimpan dalam sistem kontrol sumber. Memanfaatkan infrastruktur sebagai kode makin memudahkan otomatisasi deployment infrastruktur dan membantu mewujudkan ketetapan infrastruktur.

Pola deployment

Ketika perubahan dalam beban kerja diperlukan, strategi deployment tetap mengharuskan deployment sumber daya infrastruktur yang baru, termasuk semua perubahan yang diperlukan. Penting agar kumpulan sumber daya baru ini mengikuti pola rollout yang meminimalkan dampak pengguna. Ada dua strategi utama untuk deployment ini:

Deployment canary: Praktik mengarahkan sejumlah kecil pelanggan ke versi baru, yang biasanya dijalankan di instans layanan tunggal (canary). Lalu, Anda meneliti secara mendalam setiap perubahan perilaku atau kesalahan yang dihasilkan. Anda dapat menghapus lalu lintas dari canary jika menemui masalah kritis dan mengembalikan pengguna ke versi sebelumnya. Jika deployment berhasil, Anda dapat melanjutkan melakukan deployment pada kecepatan yang diinginkan, sambil memantau perubahan kesalahan, hingga deployment sudah dilakukan sepenuhnya. AWS CodeDeploy dapat dikonfigurasi dengan konfigurasi deployment yang memungkinkan deployment canary.

Deployment blue/green: Serupa dengan deployment canary, tetapi di sini deployment armada penuh aplikasi dilakukan secara paralel. Anda mengubah deployment di dua tumpukan (blue dan green). Sekali lagi, Anda mengirimkan lalu lintas ke versi baru, dan kembali ke versi lama jika Anda melihat masalah dengan deployment. Biasanya semua lalu lintas dialihkan sekaligus, tetapi Anda juga dapat menggunakan sebagian lalu lintas ke setiap versi untuk meningkatkan adopsi versi baru menggunakan kemampuan perutean DNS tertimbang dari Amazon Route 53. AWS CodeDeploy dan AWS Elastic Beanstalk dapat dikonfigurasikan dengan konfigurasi deployment yang memungkinkan deployment blue/green.



Gambar 8: Deployment blue/green dengan AWS Elastic Beanstalk dan Amazon Route 53

Deteksi penyimpangan

Penyimpangan didefinisikan sebagai perubahan apa pun yang menyebabkan sumber daya infrastruktur memiliki status atau konfigurasi yang berbeda dengan apa yang diharapkan. Setiap jenis perubahan konfigurasi yang tidak dikelola bertentangan dengan gagasan infrastruktur tetap, dan harus dideteksi dan diperbaiki agar infrastruktur tetap berhasil diimplementasikan.

Langkah implementasi

- Larang modifikasi di tempat pada sumber daya infrastruktur yang sedang berjalan.
  - Anda dapat menggunakan AWS Identity and Access Management (IAM) untuk menentukan siapa atau apa yang dapat mengakses layanan dan sumber daya di AWS, mengelola izin dengan ketat secara terpusat, dan menganalisis akses untuk menyempurnakan izin di AWS.
- Otomatiskan deployment sumber daya infrastruktur untuk meningkatkan keterulangan dan meminimalkan potensi kesalahan manusia.
  - Seperti yang dijelaskan dalam laporan resmi Pengantar DevOps di AWS, otomatisasi merupakan landasan dalam layanan AWS dan didukung secara internal di semua layanan, fitur, dan penawaran.
  - Melakukan prapembuatan Amazon Machine Image (AMI) Anda dapat mempercepat waktu peluncurannya. EC2 Image Builder adalah layanan AWS yang dikelola sepenuhnya yang membantu Anda mengotomatiskan pembuatan, pemeliharaan, validasi, berbagi, dan deployment AMI kustom Linux atau Windows yang disesuaikan, aman, dan terbaru.
  - Beberapa layanan yang mendukung otomatisasi adalah:

    - AWS Elastic Beanstalk adalah layanan yang digunakan untuk dengan cepat melakukan deployment dan menskalakan aplikasi web yang dikembangkan dengan Java, .NET, PHP, Node.js, Python, Ruby, Go, dan Docker pada server yang sudah dikenal seperti Apache, NGINX, Passenger, dan IIS.
    - AWS Proton membantu tim platform menghubungkan dan mengoordinasikan semua alat berbeda yang dibutuhkan tim pengembangan Anda untuk penyediaan infrastruktur, deployment kode, pemantauan, dan pembaruan. AWS Proton memungkinkan penyediaan infrastruktur sebagai kode dan deployment aplikasi nirserver dan berbasis kontainer secara otomatis.
- Memanfaatkan infrastruktur sebagai kode memudahkan otomatisasi deployment infrastruktur, dan membantu mewujudkan ketetapan infrastruktur. AWS menyediakan layanan yang memungkinkan pembuatan, deployment, dan pemeliharaan infrastruktur dengan cara yang terprogram, deskriptif, dan deklaratif.
    - AWS CloudFormation membantu developer membuat sumber daya AWS dengan cara yang teratur dan dapat diprediksi. Sumber daya ditulis dalam file teks menggunakan format JSON atau YAML. Templat memerlukan sintaks dan struktur tertentu yang bergantung pada jenis sumber daya yang dibuat dan dikelola. Anda menulis sumber daya Anda di JSON atau YAML dengan editor kode apa pun seperti AWS Cloud9, memeriksanya ke dalam sistem kontrol versi, dan kemudian CloudFormation membangun layanan yang ditentukan dengan cara yang aman dan dapat diulang.
    - AWS Serverless Application Model(AWS SAM) adalah kerangka kerja sumber terbuka yang dapat Anda gunakan untuk membangun aplikasi nirserver di AWS. AWS SAM terintegrasi dengan layanan AWS lainnya, dan merupakan pengembangan dari CloudFormation.
    - AWS Cloud Development Kit (AWS CDK) adalah kerangka pengembangan perangkat lunak sumber terbuka untuk membuat model dan menyediakan sumber daya aplikasi cloud Anda menggunakan bahasa pemrograman yang sudah dipahami. Anda dapat menggunakan AWS CDK untuk membuat model infrastruktur aplikasi menggunakan TypeScript, Python, Java, dan .NET. AWS CDK menggunakan CloudFormation di latar belakang untuk menyediakan sumber daya dengan cara yang aman dan dapat diulang.
    - AWS Cloud Control API memperkenalkan seperangkat API yang umum yaitu Membuat, Membaca, Memperbarui, Menghapus, dan Mencantumkan (CRUDL) untuk membantu developer mengelola infrastruktur cloud dengan mudah dan konsisten. API umum Cloud Control API memungkinkan developer untuk mengelola siklus hidup layanan AWS dan pihak ketiga secara seragam.
- Implementasikan pola deployment yang meminimalkan dampak pengguna.

  - Deployment canary:
    - Set up an API Gateway canary release deployment
    - Create a pipeline with canary deployments for Amazon ECS using AWS App Mesh
  - Deployment blue/green: laporan resmi Deployment Blue/Green di AWS menjelaskan contoh teknik dalam mengimplementasikan strategi deployment blue/green.
- Deteksi konfigurasi atau penyimpangan status. Untuk detail selengkapnya, lihat Detecting unmanaged configuration changes to stacks and resources.

Sumber daya

Praktik Terbaik Terkait:

- REL08-BP05 Melakukan deployment perubahan dengan otomatisasi

Dokumen terkait:

- Mengotomatiskan deployment aman tanpa campur tangan
- Leveraging AWS CloudFormation to create an immutable infrastructure at Nubank
- Infrastruktur sebagai kode
- Implementing an alarm to automatically detect drift in AWS CloudFormation stacks

Video terkait:

- AWS re:Invent 2020: Reliability, consistency, and confidence through immutability

REL08-BP05 Melakukan deployment perubahan dengan otomatisasi

Deployment dan patching diotomatisasi untuk menyingkirkan dampak negatif.

Membuat perubahan pada sistem produksi adalah salah satu area risiko terbesar untuk banyak organisasi. Kami menganggap deployment sebagai masalah kelas pertama untuk diatasi bersamaan dengan masalah-masalah bisnis yang ditangani oleh perangkat lunak. Saat ini, ini berarti penggunaan otomatisasi kapan saja memungkinkan dalam operasi, termasuk untuk menguji dan melakukan deployment perubahan, menambah atau menghapus kapasitas, dan memigrasikan data. AWS CodePipeline memungkinkan Anda mengelola langkah-langkah yang diperlukan untuk merilis beban kerja Anda. Ini mencakup status deployment menggunakan AWS CodeDeploy untuk

mengotomatisasi deployment kode aplikasi ke instans Amazon EC2, instans on-premise, fungsi Lambda nirserver, atau layanan Amazon ECS.

> **Rekomendasi**
>
> Meskipun kebijaksanaan konvensional menyarankan Anda untuk melibatkan manusia untuk prosedur operasional paling sulit, kami justru menyarankan Anda mengotomatisasi prosedur paling sulit untuk alasan tersebut.

Antipola umum:

- Melakukan perubahan secara manual.
- Melewatkan langkah-langkah dalam otomatisasi Anda melalui alur kerja darurat.
- Tidak mengikuti rencana Anda.

Manfaat menjalankan praktik terbaik ini: Penggunaan otomatisasi untuk melakukan deployment semua perubahan dapat menyingkirkan potensi munculnya kesalahan manusia dan menghadirkan kemampuan untuk menguji sebelum mengubah produksi guna memastikan rencana Anda sudah lengkap.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

- Otomatiskan pipeline deployment Anda. Pipeline deployment memungkinkan Anda untuk memanggil pengujian dan deteksi anomali secara otomatis, serta memberi Anda pilihan untuk menghentikan pipeline pada langkah tertentu sebelum deployment produksi atau membatalkan perubahan secara otomatis.
  - Amazon Builders' Library: Memastikan keamanan pembatalan selama deployment
  - Amazon Builders' Library: Melaju lebih cepat dengan pengiriman berkelanjutan
    - Gunakan AWS CodePipeline (atau produk pihak ketiga tepercaya) untuk menetapkan dan menjalankan pipeline Anda.
      - Konfigurasikan pipeline untuk mulai saat ada perubahan yang dimasukkan ke repositori kode Anda.
        - Apa Itu AWS CodePipeline?

- Gunakan Amazon Simple Notification Service (Amazon SNS) dan Amazon Simple Email Service (Amazon SES) untuk mengirimkan notifikasi tentang masalah di dalam pipeline atau integrasikan dengan alat obrolan tim, seperti Amazon Chime.
  - Apa Itu Amazon Simple Notification Service?
  - Apa Itu Amazon SES?
  - Apa itu Amazon Chime?
  - Otomatiskan pesan obrolan dengan webhooks.

Sumber daya

Dokumen terkait:

- Partner APN: partner yang dapat membantu Anda membuat solusi deployment yang diotomatisasi
- AWS Marketplace: produk yang dapat digunakan untuk mengotomatisasi deployment Anda
- Otomatiskan pesan obrolan dengan webhooks.
- Amazon Builders' Library: Memastikan keamanan pembatalan selama deployment
- Amazon Builders' Library: Melaju lebih cepat dengan pengiriman berkelanjutan
- Apa Itu AWS CodePipeline?
- Apa Itu CodeDeploy?
- AWS Systems Manager Patch Manager
- Apa Itu Amazon SES?
- Apa Itu Amazon Simple Notification Service?

Video terkait:

- AWS Summit 2019: CI/CD di AWS

Manajemen kegagalan

Pertanyaan

- REL 9. Bagaimana cara mencadangkan data?
- REL 10. Bagaimana cara menggunakan isolasi kesalahan untuk melindungi beban kerja?

- REL 11. Bagaimana Anda mendesain beban kerja agar dapat bertahan jika terjadi kegagalan komponen?
- REL 12. Bagaimana cara menguji keandalan?
- REL 13. Bagaimana cara Anda mempersiapkan pemulihan bencana (DR)?

REL 9. Bagaimana cara mencadangkan data?

Cadangkan data, aplikasi, dan konfigurasi untuk memenuhi persyaratan Anda untuk sasaran waktu pemulihan (RTO) dan sasaran titik pemulihan (RPO).

Praktik terbaik

- REL09-BP01 Mengidentifikasi dan mencadangkan data yang perlu dicadangkan, atau memproduksi ulang data dari sumber
- REL09-BP02 Mengamankan dan mengenkripsikan cadangan
- REL09-BP03 Melakukan pencadangan data secara otomatis.
- REL09-BP04 Melakukan pemulihan data secara berkala untuk memverifikasi integritas dan proses pencadangan

REL09-BP01 Mengidentifikasi dan mencadangkan data yang perlu dicadangkan, atau memproduksi ulang data dari sumber

Pahami dan gunakan kemampuan pencadangan sumber daya dan layanan data yang digunakan oleh beban kerja. Sebagian besar layanan menyediakan kemampuan untuk mencadangkan data beban kerja.

Hasil yang diinginkan: Sumber data telah diidentifikasi dan diklasifikasikan berdasarkan tingkat kekritisan. Lalu, bangun strategi untuk pemulihan data berdasarkan RPO. Strategi ini melibatkan pencadangan sumber-sumber data, atau memiliki kemampuan untuk memproduksi ulang data dari sumber lain. Untuk kasus kehilangan data, strategi yang diimplementasikan memungkinkan pemulihan atau produksi ulang data dalam RPO dan RTO yang ditetapkan.

Fase kemapanan cloud: Fondasi

Antipola umum:

- Tidak mengetahui semua sumber data untuk beban kerja serta tingkat kekritisannya.
- Tidak melakukan pencadangan sumber data kritis.

- Melakukan pencadangan hanya beberapa sumber data tanpa menggunakan tingkat kekritisan sebagai kriteria.
- Tidak ada RPO yang ditetapkan, atau frekuensi pencadangan tidak memenuhi RPO.
- Tidak mengevaluasi apakah cadangan diperlukan atau apakah data dapat diproduksi ulang dari sumber lain.

Manfaat menjalankan praktik terbaik ini: Mengidentifikasi tempat-tempat yang memerlukan pencadangan dan mengimplementasikan mekanisme untuk membuat cadangan, atau mampu memproduksi ulang data dari sumber eksternal, semuanya dapat meningkatkan kemampuan untuk memulihkan dan mengembalikan data selama penghentian.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Semua penyimpanan data AWS menawarkan kemampuan pencadangan. Layanan seperti Amazon RDS dan Amazon DynamoDB memberikan dukungan tambahan pada pencadangan otomatis yang memungkinkan pemulihan titik waktu (PITR), yang memungkinkan Anda untuk memulihkan cadangan ke waktu kapan pun hingga lima menit atau kurang sebelum waktu saat ini. Banyak layanan AWS yang menawarkan kemampuan untuk menyalin cadangan ke Wilayah AWS lain. AWS Backup adalah alat yang memberikan kepada Anda kemampuan untuk memusatkan dan mengotomatiskan perlindungan data di layanan AWS. AWS Elastic Disaster Recovery memungkinkan Anda menyalin beban kerja server penuh dan mempertahankan perlindungan data berkelanjutan dari on-premise, lintas AZ, atau lintas Wilayah, dengan Sasaran Titik Pemulihan (RPO) yang diukur dalam detik.

Amazon S3 dapat digunakan sebagai tujuan pencadangan untuk sumber daya yang dikelola mandiri dan yang dikelola oleh AWS. Layanan AWS seperti Amazon EBS, Amazon RDS, dan Amazon DynamoDB memiliki kemampuan bawaan untuk membuat cadangan. Perangkat lunak pencadangan pihak ketiga juga dapat digunakan.

Data on-premise dapat dicadangkan ke AWS Cloud menggunakan AWS Storage Gateway atau AWS DataSync. Bucket Amazon S3 dapat digunakan untuk menyimpan data ini di AWS. Amazon S3 menawarkan beberapa tingkat penyimpanan seperti Amazon Glacier atau Amazon Glacier Deep Archive untuk mengurangi biaya penyimpanan data.

Anda mungkin dapat memenuhi kebutuhan pemulihan data dengan memproduksi ulang data dari sumber lain. Contohnya, simpul replika Amazon ElastiCache atau replika baca Amazon RDS dapat digunakan untuk memproduksi ulang data jika yang data utama hilang. Jika sumber seperti ini dapat digunakan untuk memenuhi Sasaran Titik Pemulihan (RPO) dan Sasaran Waktu Pemulihan

(RTO), Anda mungkin tidak memerlukan cadangan. Contoh lainnya, jika bekerja dengan Amazon EMR, pencadangan penyimpanan data HDFS Anda mungkin tidak diperlukan, selama Anda dapat memproduksi ulang data ke Amazon EMR dari Amazon S3.

Ketika menyeleksi strategi pencadangan, pertimbangkan waktu yang diperlukan untuk memulihkan data. Waktu yang diperlukan untuk memulihkan data tergantung pada tipe cadangan (untuk kasus strategi pencadangan), atau kompleksitas mekanisme produksi ulang data. Waktu ini termasuk dalam RTO untuk beban kerja.

Langkah implementasi

1. Mengidentifikasi semua sumber data untuk beban kerja. Data dapat disimpan di sejumlah sumber daya seperti basis data, volume, sistem file, sistem pencatatan log, dan penyimpanan objek. Lihat bagian Sumber Daya untuk menemukan Dokumen terkait tentang berbagai layanan AWS tempat data disimpan, dan kemampuan pencadangan yang disediakan oleh layanan-layanan ini.
2. Klasifikasikan sumber data berdasarkan tingkat kekritisan. Set data yang berbeda akan memiliki tingkat kekritisan yang berbeda untuk suatu beban kerja, sehingga memiliki persyaratan untuk ketahanan yang berbeda-beda. Misalnya, beberapa data mungkin kritis dan memerlukan RPO hampir nol, sedangkan data lain mungkin tidak terlalu kritis dan dapat mentoleransi RPO yang lebih tinggi dan beberapa hilang data. Demikian juga, set data yang berbeda mungkin memiliki persyaratan RTO yang berbeda.
3. Gunakan AWS atau layanan pihak ketiga untuk membuat cadangan data. AWS Backup adalah layanan terkelola yang memungkinkan pembuatan cadangan berbagai sumber data di AWS. AWS Elastic Disaster Recovery menangani replikasi data otomatis sub-detik ke Wilayah AWS. Sebagian besar layanan AWS juga memiliki kemampuan native untuk membuat cadangan. AWS Marketplace juga memiliki banyak solusi untuk menyediakan kemampuan-kemampuan ini. Lihat Sumber Daya yang disebutkan di bawah ini untuk mendapatkan informasi tentang cara membuat cadangan data dari berbagai layanan AWS.
4. Untuk data yang tidak dicadangkan, buat mekanisme produksi ulang data. Anda mungkin memilih untuk tidak mencadangkan data yang dapat diproduksi ulang dari sumber lain karena berbagai alasan. Mungkin terdapat situasi di mana produksi ulang data dari sumber lain saat diperlukan lebih murah daripada membuat cadangan, karena mungkin ada biaya terkait penyimpanan cadangan. Contoh lainnya adalah ketika pemulihan dari cadangan memerlukan waktu lebih lama daripada produksi ulang data dari sumber lain, sehingga mengakibatkan pelanggaran RTO. Pada situasi-situasi demikian, pertimbangkan semua kompromi dan bangun proses yang ditetapkan dengan baik terkait bagaimana data dapat diproduksi ulang dari sumber-sumber ini saat pemulihan data diperlukan. Misalnya, jika Anda telah memuat data dari Amazon S3 ke gudang data (seperti

Amazon Redshift), atau klaster MapReduce (seperti Amazon EMR) untuk melakukan analisis pada data tersebut, ini mungkin adalah contoh data yang dapat diproduksi ulang dari sumber lain. Selama hasil dari semua analisis ini disimpan di suatu tempat atau dapat diproduksi ulang, Anda tidak akan mengalami kehilangan data akibat kegagalan pada gudang data atau klaster MapReduce. Contoh lain data yang dapat diproduksi ulang dari sumber lain adalah cache (seperti Amazon ElastiCache) atau replika baca RDS.

5. Buat jadwal rutin pencadangan data. Membuat cadangan sumber data adalah proses berkala dan frekuensinya tergantung pada RPO.

Tingkat upaya untuk rencana implementasi: Sedang

Sumber daya

Praktik Terbaik Terkait:

REL13-BP01 Tetapkan sasaran pemulihan untuk waktu henti dan kehilangan data

REL13-BP02 Menggunakan strategi pemulihan yang ditentukan untuk memenuhi sasaran pemulihan

Dokumen terkait:

- Apa Itu AWS Backup?
- Apa itu AWS DataSync?
- Apa itu Gateway Volume?
- Partner APN: partner yang dapat membantu terkait pencadangan
- AWS Marketplace: produk yang dapat digunakan untuk pencadangan
- Amazon EBS Snapshots
- Mencadangkan Amazon EFS
- Mencadangkan Amazon FSx untuk Windows File Server
- Pencadangan dan Pemulihan untuk ElastiCache for Redis
- Membuat Snapshot Klaster DB di Neptune
- Membuat Snapshot DB
- Membuat Aturan EventBridge yang Memicu Berdasarkan Jadwal
- Replika Lintas-Wilayah dengan Amazon S3
- EFS-ke-EFS AWS Backup
- Mengekspor Data Log ke Amazon S3

- Manajemen siklus hidup objek
- Pemulihan dan Pencadangan Sesuai Permintaan untuk DynamoDB
- Pemulihan titik waktu untuk DynamoDB
- Bekerja dengan Snapshot Indeks Amazon OpenSearch Service
- Apa itu AWS Elastic Disaster Recovery?

Video terkait:

- AWS re:Invent 2021 - Pencadangan, pemulihan bencana, dan perlindungan ransomware dengan AWS
- Demo AWS Backup: Pencadangan Lintas Akun dan Lintas Wilayah
- AWS re:Invent 2019: Memahami AWS Backup, dengan Rackspace (STG341)

Contoh terkait:

- Well-Architected Lab - Mengimplementasikan Replikasi Lintas Wilayah (CRR) Dua Arah untuk Amazon S3
- Well-Architected Lab - Pengujian Pencadangan dan Pemulihan Data
- Well-Architected Lab: Pencadangan dan Pemulihan dengan Failback untuk Beban Kerja Analitik
- Well-Architected Lab: Pemulihan Bencana - Pencadangan dan Pemulihan

REL09-BP02 Mengamankan dan mengenkripsikan cadangan

Kontrol dan deteksi akses ke cadangan menggunakan autentikasi dan otorisasi. Gunakan enkripsi untuk mencegah dan mendeteksi jika integritas data cadangan terancam.

Antipola umum:

- Memiliki akses yang sama ke cadangan dan otomatisasi pemulihan seperti yang dilakukan pada data.
- Tidak mengenkripsi cadangan.

Manfaat menjalankan praktik terbaik ini: Mengamankan cadangan Anda akan mencegah gangguan terhadap data, dan enkripsi data mencegah akses ke data tersebut jika tidak sengaja terekspos.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Kontrol dan deteksi akses ke cadangan menggunakan autentikasi dan otorisasi seperti AWS Identity and Access Management (IAM). Gunakan enkripsi untuk mencegah dan mendeteksi jika integritas data cadangan terancam.

Amazon S3 mendukung beberapa metode enkripsi data diam. Dengan menggunakan enkripsi di sisi server, Amazon S3 menerima objek sebagai data yang tidak terenkripsi dan mengenkripsinya saat disimpan. Dengan menggunakan enkripsi di sisi klien, aplikasi beban kerja bertanggung jawab untuk mengenkripsi data sebelum mengirimkannya ke Amazon S3. Kedua metode tersebut memungkinkan Anda untuk menggunakan AWS Key Management Service (AWS KMS) guna menciptakan dan menyimpan kunci data. Anda dapat menyediakan kunci Anda sendiri dan bertanggung jawab atas kunci tersebut. Dengan menggunakan AWS KMS, Anda dapat menetapkan kebijakan menggunakan IAM terkait siapa yang dapat dan tidak dapat mengakses kunci data dan data terdekripsi.

Untuk Amazon RDS, cadangan juga akan dienkripsi jika Anda memilih untuk mengenkripsikan basis data. Cadangan DynamoDB selalu terenkripsi. Ketika menggunakan AWS Elastic Disaster Recovery, semua data bergerak dan data diam dienkripsi. Dengan Elastic Disaster Recovery, data diam dapat dienkripsi menggunakan Kunci Enkripsi Volume enkripsi Amazon EBS atau kunci kustom yang dikelola pelanggan.

Langkah implementasi

1. Gunakan enkripsi untuk setiap penyimpanan data. Jika sumber data terenkripsi, maka cadangannya juga akan terenkripsi.
   - Gunakan enkripsi di Amazon RDS.. Anda dapat mengonfigurasi enkripsi diam menggunakan AWS Key Management Service saat membuat instans RDS.
   - Gunakan enkripsi di volume Amazon EBS.. Anda dapat mengonfigurasi enkripsi default atau menentukan kunci unik saat pembuatan volume.
   - Gunakan enkripsi Amazon DynamoDB yang diperlukan. DynamoDB mengenkripsi semua data diam. Anda dapat menggunakan kunci AWS KMS yang dimiliki AWS atau kunci KMS yang dikelola AWS, menentukan kunci yang disimpan di akun Anda.
   - Enkripsikan data yang disimpan di Amazon EFS. Konfigurasikan enkripsi saat Anda membuat sistem file.
   - Konfigurasikan enkripsi di Wilayah sumber dan tujuan. Anda dapat mengonfigurasi enkripsi diam di Amazon S3 menggunakan kunci yang disimpan di KMS, tetapi kuncinya bersifat spesifik Wilayah. Anda dapat menentukan kunci tujuan saat mengonfigurasi replikasi.

- Pilih apakah akan menggunakan enkripsi Amazon EBS default atau kustom untuk Elastic Disaster Recovery. Opsi ini akan mengenkripsi data diam yang direplikasi di disk Subnet Area Staging dan disk yang direplikasi.

2. Implementasikan izin hak akses paling rendah untuk mengakses cadangan. Ikuti praktik terbaik untuk membatasi akses ke cadangan, snapshot, dan replika sesuai dengan praktik terbaik keamanan.

Sumber daya

Dokumen terkait:

- AWS Marketplace: produk yang dapat digunakan untuk pencadangan
- Enkripsi Amazon EBS
- Amazon S3: Melindungi Data Menggunakan Enkripsi
- Konfigurasi Tambahan CRR: Mereplikasi Objek yang Dibuat dengan Enkripsi di Sisi Server (SSE) Menggunakan Kunci Enkripsi yang disimpan di AWS KMS
- Enkripsi Diam DynamoDB
- Mengenkripsi Sumber Daya Amazon RDS
- Mengenkripsi Data dan Metadata di Amazon EFS
- Enkripsi untuk Cadangan di AWS
- Mengelola Tabel Terenkripsi
- Pilar Keamanan - AWS Well-Architected Framework
- Apa itu AWS Elastic Disaster Recovery?

Contoh terkait:

- Well-Architected Lab - Mengimplementasikan Replikasi Lintas Wilayah (CRR) Dua Arah untuk Amazon S3

REL09-BP03 Melakukan pencadangan data secara otomatis.

Konfigurasikan pencadangan untuk dilakukan secara otomatis berdasarkan jadwal berkala mengacu pada Sasaran Titik Pemulihan (RPO), atau berdasarkan perubahan dalam set data. Set data kritis dengan persyaratan data hilang yang rendah perlu dicadangkan otomatis secara rutin, sedangkan

data yang tidak terlalu kritis di mana beberapa data hilang masih dapat diterima dapat dicadangkan tidak terlalu sering.

Hasil yang diinginkan: Proses otomatis yang membuat cadangan sumber data dengan jadwal yang ditetapkan.

Antipola umum:

- Melakukan pencadangan secara manual.
- Menggunakan sumber daya yang memiliki kemampuan pencadangan, tetapi tidak termasuk pencadangan dalam otomatisasi Anda.

Manfaat menjalankan praktik terbaik ini: Otomatisasi pencadangan memverifikasi pencadangan dilakukan secara teratur berdasarkan RPO Anda dan memberi tahu Anda jika pencadangan tidak dilakukan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

AWS Backup dapat digunakan untuk membuat cadangan data otomatis untuk berbagai sumber data AWS. Instans Amazon RDS dapat dicadangkan hampir secara berkelanjutan setiap lima menit dan objek Amazon S3 dapat dicadangkan hampir secara berkelanjutan setiap lima belas menit, dan memungkinkan pemulihan titik waktu (PITR) ke titik waktu tertentu di dalam riwayat pencadangan. Untuk sumber data AWS lainnya, seperti volume Amazon EBS, tabel Amazon DynamoDB, atau sistem file Amazon FSx, AWS Backup dapat menjalankan pencadangan otomatis setiap satu jam. Layanan ini juga menawarkan kemampuan pencadangan native. Layanan AWS yang menawarkan pencadangan otomatis dengan pemulihan titik waktu antara lain Amazon DynamoDB, Amazon RDS, dan Amazon Keyspaces (untuk Apache Cassandra) - ini dapat dipulihkan ke titik waktu tertentu dalam riwayat pencadangan. Sebagian besar layanan penyimpanan data AWS lainnya menawarkan kemampuan untuk menjadwalkan pencadangan berkala, dengan frekuensi setiap satu jam.

Amazon RDS dan Amazon DynamoDB menawarkan pencadangan berkelanjutan dengan pemulihan titik waktu. Versioning Amazon S3, setelah diaktifkan, bersifat otomatis. Amazon Data Lifecycle Manager dapat digunakan untuk mengotomatiskan pembuatan, penyalinan, dan penghapusan snapshot Amazon EBS. Layanan ini juga dapat mengotomatiskan pembuatan, penyalinan, penghentian, dan pembatalan registrasi Amazon Machine Images (AMI) yang dicadangkan Amazon EBS dan snapshot Amazon EBS yang melandasinya.

AWS Elastic Disaster Recovery memberikan replikasi tingkat blok yang berkelanjutan dari lingkungan sumber (on-premise atau AWS) ke wilayah pemulihan target. Snapshot Amazon EBS titik waktu dibuat dan dikelola secara otomatis oleh layanan.

Untuk tampilan otomatisasi dan riwayat pencadangan terpusat, AWS Backup menyediakan solusi pencadangan berbasis kebijakan yang terkelola penuh. Layanan ini memusatkan dan mengotomatiskan pencadangan data di beberapa layanan AWS di cloud serta on-premise menggunakan AWS Storage Gateway.

Selain versioning, Amazon S3 dilengkapi dengan replikasi. Seluruh bucket S3 dapat direplikasi secara otomatis ke bucket lain di Wilayah AWS yang sama atau berbeda.

Langkah implementasi

1. Identifikasi sumber data yang saat ini dicadangkan secara manual. Untuk detail selengkapnya, lihat REL09-BP01 Mengidentifikasi dan mencadangkan data yang perlu dicadangkan, atau memproduksi ulang data dari sumber.
2. Tentukan RPO untuk beban kerja. Untuk detail selengkapnya, lihat REL13-BP01 Tetapkan sasaran pemulihan untuk waktu henti dan kehilangan data.
3. Gunakan solusi cadangan otomatis atau layanan terkelola. AWS Backup adalah layanan terkelola penuh yang mempermudah pemusatan dan pengotomatisan perlindungan data di seluruh layanan AWS, di cloud, dan on-premise. Dengan menggunakan rencana cadangan di AWS Backup, buat aturan yang menetapkan sumber daya yang akan dicadangkan, dan frekuensi pembuatan cadangan ini. Frekuensi ini harus mengacu pada RPO yang ditetapkan pada Langkah 2. Untuk panduan praktik langsung tentang cara membuat cadangan otomatis menggunakan AWS Backup, lihat Pengujian Pencadangan dan Pemulihan Data. Kemampuan pencadangan native ditawarkan oleh sebagian besar layanan AWS yang menyimpan data. Misalnya, RDS dapat dimanfaatkan untuk pencadangan otomatis dengan pemulihan titik waktu (PITR).
4. Untuk sumber daya yang tidak didukung oleh solusi pencadangan otomatis atau layanan terkelola seperti sumber data on-premise atau antrean pesan, pertimbangkan penggunaan solusi pihak ketiga tepercaya untuk membuat cadangan otomatis. Pilihan lainnya, Anda dapat membuat otomatisasi untuk melakukannya menggunakan AWS CLI atau SDK. Anda dapat menggunakan Fungsi AWS Lambda atau AWS Step Functions untuk menetapkan logika yang terlibat dalam pembuatan cadangan data, dan gunakan Amazon EventBridge untuk melaksanakannya dengan frekuensi yang didasarkan pada RPO Anda.

Tingkat upaya untuk Rencana Implementasi: Rendah

Sumber daya

Dokumen terkait:

- Partner APN: partner yang dapat membantu terkait pencadangan
- AWS Marketplace: produk yang dapat digunakan untuk pencadangan
- Membuat Aturan EventBridge yang Memicu Berdasarkan Jadwal
- Apa Itu AWS Backup?
- Apa Itu AWS Step Functions?
- Apa itu AWS Elastic Disaster Recovery?

Video terkait:

- AWS re:Invent 2019: Memahami AWS Backup, dengan Rackspace (STG341)

Contoh terkait:

- Well-Architected Lab - Pengujian Pencadangan dan Pemulihan Data

REL09-BP04 Melakukan pemulihan data secara berkala untuk memverifikasi integritas dan proses pencadangan

Validasikan bahwa implementasi proses pencadangan Anda memenuhi Sasaran Waktu Pemulihan (RTO) dan Sasaran Titik Pemulihan (RPO) dengan melakukan uji pemulihan.

Hasil yang diinginkan: Data dari cadangan dipulihkan secara berkala menggunakan mekanisme yang ditentukan dengan baik untuk memverifikasi bahwa pemulihan tersebut dapat dilakukan dalam sasaran waktu pemulihan (RTO) yang ditetapkan untuk beban kerja. Verifikasikan bahwa pemulihan dari pencadangan menghasilkan sumber daya yang berisi data asli tanpa ada data yang rusak atau tidak dapat diakses, serta dengan kehilangan data dalam sasaran titik pemulihan (RPO).

Antipola umum:

- Memulihkan cadangan, tetapi tidak mengambil data atau membuat kueri data apa pun untuk memastikan pemulihan dapat digunakan.
- Dengan anggapan bahwa cadangan sudah ada.

- Dengan anggapan bahwa cadangan sistem dapat dioperasikan sepenuhnya dan data dapat dipulihkan dari sistem.
- Dengan anggapan bahwa waktu untuk memulihkan data dari cadangan termasuk dalam RTO untuk beban kerja.
- Dengan anggapan bahwa data dalam cadangan termasuk dalam RPO untuk beban kerja.
- Memulihkan apabila diperlukan, tanpa menggunakan runbook, atau di luar prosedur otomatis yang ditetapkan.

Manfaat menjalankan praktik terbaik ini: Pengujian pemulihan cadangan memastikan data dapat dipulihkan saat dibutuhkan tanpa perlu khawatir data akan hilang atau rusak, bahwa restorasi dan pemulihan dapat dilakukan dalam batas RTO untuk beban kerja, dan kehilangan data apa pun termasuk dalam RPO untuk beban kerja.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Pengujian kemampuan pencadangan dan pemulihan meningkatkan keyakinan pada kemampuan untuk menjalankan tindakan ini selama pemadaman. Pulihkan cadangan ke lokasi baru secara berkala dan lakukan pengujian untuk memverifikasi integritas data. Beberapa pengujian umum yang harus dilakukan yakni, memeriksa apakah semua data tersedia, tidak rusak, dapat diakses, dan setiap kehilangan data termasuk dalam RPO untuk beban kerja. Pengujian tersebut dapat juga membantu memastikan apakah mekanisme pemulihan cukup cepat untuk mengakomodasi RTO beban kerja.

Dengan menggunakan AWS, Anda dapat mempertahankan lingkungan pengujian dan memulihkan cadangan untuk menilai kemampuan RTO dan RPO, serta menjalankan pengujian pada konten dan integritas data.

Selain itu, Amazon RDS dan Amazon DynamoDB memungkinkan pemulihan titik waktu (PITR). Dengan menggunakan pencadangan berkelanjutan, Anda dapat memulihkan set data ke statusnya pada waktu dan tanggal yang ditentukan.

apakah semua data tersedia, tidak rusak, dapat diakses, dan kehilangan data apa pun termasuk dalam RPO untuk beban kerja. Pengujian tersebut dapat juga membantu memastikan apakah mekanisme pemulihan cukup cepat untuk mengakomodasi RTO beban kerja.

AWS Elastic Disaster Recovery menawarkan snapshot pemulihan titik waktu volume Amazon EBS secara berkelanjutan. Saat server sumber direplikasi, status titik waktu dicatat seiring waktu

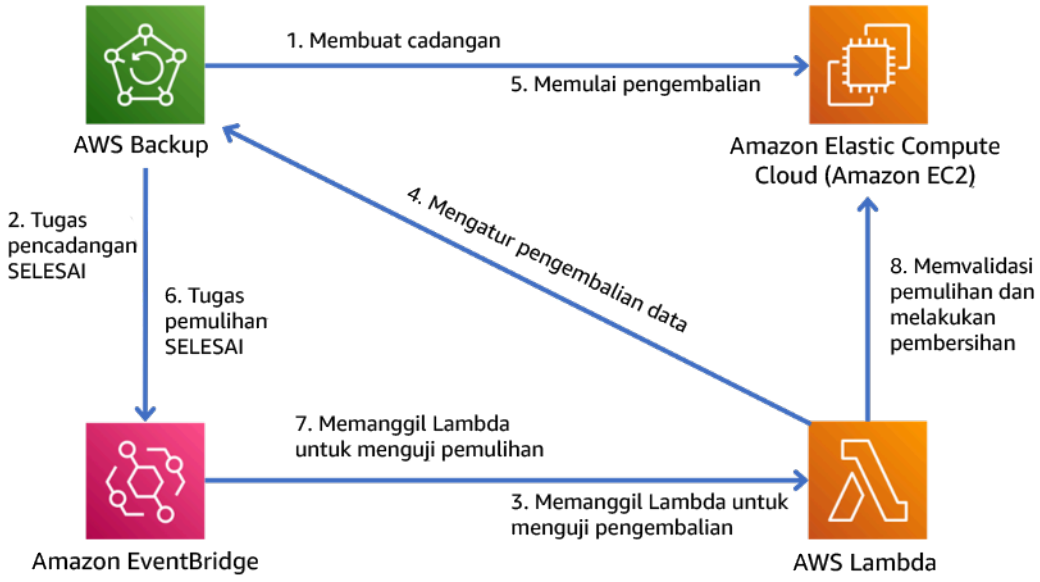
berdasarkan kebijakan yang dikonfigurasi. Elastic Disaster Recovery membantu Anda memverifikasi integritas snapshot ini dengan meluncurkan instans untuk tujuan pengujian dan latihan tanpa mengarahkan ulang lalu lintas.

Langkah implementasi

1. Identifikasi sumber data yang dicadangkan saat ini dan lokasi penyimpanan cadangan tersebut. Untuk panduan implementasi, lihat REL09-BP01 Mengidentifikasi dan mencadangkan data yang perlu dicadangkan, atau memproduksi ulang data dari sumber.
2. Tetapkan kriteria validasi data untuk setiap sumber data. Jenis data yang berbeda akan memiliki properti data yang berbeda, yang dapat memerlukan mekanisme validasi yang berbeda. Pertimbangkan bagaimana data ini dapat divalidasi sebelum Anda yakin untuk menggunakannya dalam produksi. Beberapa cara umum untuk memvalidasi adalah dengan menggunakan data dan properti pencadangan seperti jenis data, format, checksum, ukuran, atau kombinasi darinya dengan logika validasi kustom. Misalnya, hal ini dapat dilakukan dengan perbandingan nilai checksum antara sumber daya yang dipulihkan dan sumber data pada waktu cadangan dibuat.
3. Tetapkan RTO dan RPO untuk memulihkan data berdasarkan kekritisan data. Untuk panduan implementasi, lihat REL13-BP01 Tetapkan sasaran pemulihan untuk waktu henti dan kehilangan data.
4. Nilai kemampuan pemulihan Anda. Tinjau strategi pencadangan dan pemulihan untuk memahami apakah hal tersebut memenuhi RTO dan RPO, serta sesuaikan strategi yang dibutuhkan. Dengan menggunakan AWS Resilience Hub, Anda dapat menjalankan penilaian beban kerja. Penilaian tersebut mengevaluasi konfigurasi aplikasi terhadap kebijakan dan pelaporan ketahanan jika target RTO dan RPO dapat dipenuhi.
5. Lakukan pemulihan pengujian dengan menggunakan proses yang ditetapkan saat ini yang digunakan dalam produksi untuk pemulihan data. Proses ini bergantung pada cara sumber data asli dicadangkan, format dan lokasi penyimpanan cadangan tersebut, atau apakah data direproduksi dari sumber lainnya. Contohnya, jika Anda menggunakan layanan terkelola seperti AWS Backup, hal ini bisa sederhana seperti memulihkan cadangan ke sumber daya baru. Jika Anda menggunakan AWS Elastic Disaster Recovery, Anda dapat meluncurkan latihan pemulihan.
6. Validasikan pemulihan data dari sumber daya yang dipulihkan berdasarkan kriteria yang ditetapkan sebelumnya untuk validasi data. Apakah data yang direstorasi dan dipulihkan memiliki sebagian besar catatan atau item terbaru pada waktu pencadangan? Apakah data ini termasuk dalam RPO untuk beban kerja?
7. Ukur waktu yang diperlukan untuk restorasi dan pemulihan dan bandingkan dengan RTO yang telah Anda tetapkan. Apakah data ini termasuk dalam RTO untuk beban kerja? Misalnya,

bandingkan stempel waktu dari kapan proses pemulihan dimulai dan kapan validasi pemulihan selesai untuk menghitung waktu yang diperlukan proses ini. Semua panggilan API AWS diberi cap waktu dan informasi ini tersedia di AWS CloudTrail. Ketika informasi ini dapat menyediakan detail waktu kapan proses pemulihan dimulai, stempel waktu akhir untuk kapan validasi diselesaikan harus dicatat melalui logika validasi. Jika menggunakan proses otomatis, maka layanan seperti Amazon DynamoDB dapat digunakan untuk menyimpan informasi ini. Selain itu, banyak layanan AWS yang menyediakan riwayat peristiwa berisi informasi dengan stempel waktu tentang kapan tindakan diambil. Di dalam AWS Backup, tindakan pencadangan dan pemulihan disebut sebagai tugas, dan tugas tersebut berisi informasi cap waktu sebagai bagian dari metadata yang dapat digunakan untuk mengukur waktu yang diperlukan untuk restorasi dan pemulihan.

8. Beri notifikasi kepada para pemangku kepentingan jika validasi data gagal, atau jika waktu yang diperlukan untuk restorasi dan pemulihan melebihi RTO yang ditetapkan untuk beban kerja. Ketika mengimplementasikan otomatisasi untuk melakukan tindakan ini, seperti dalam lab ini, layanan seperti Amazon Simple Notification Service (Amazon SNS) dapat digunakan untuk mengirimkan notifikasi push seperti email atau SMS kepada para pemangku kepentingan. Pesan ini juga dapat dipublikasikan di aplikasi olahpesan seperti Amazon Chime, Slack, atau Microsoft Teams atau digunakan untuk membuat tugas sebagai OpsItems menggunakan AWS Systems Manager OpsCenter.
9. Otomatiskan proses ini untuk menjalankannya secara berkala. Misalnya, layanan seperti AWS Lambda atau State Machine di AWS Step Functions dapat digunakan untuk mengotomatiskan proses pemulihan, dan Amazon EventBridge dapat digunakan untuk memicu alur kerja otomatisasi ini secara berkala seperti yang ditampilkan dalam diagram arsitektur di bawah ini. Pelajar cara untuk Mengotomatiskan validasi pemulihan data dengan AWS Backup. Selain itu, Well-Architected lab ini memberikan pengalaman praktik langsung mengenai salah satu cara untuk melakukan otomatisasi untuk beberapa langkah di sini.



Gambar 9. Proses pencadangan dan pemulihan otomatis

Tingkat upaya untuk Rencana Implementasi: Sedang hingga tinggi, bergantung pada kompleksitas kriteria validasi.

Sumber daya

Dokumen terkait:

- Mengotomatiskan validasi pemulihan data dengan AWS Backup
- Partner APN: partner yang dapat membantu terkait pencadangan
- AWS Marketplace: produk yang dapat digunakan untuk pencadangan
- Membuat Aturan EventBridge yang Memicu Berdasarkan Jadwal
- Pemulihan dan pencadangan sesuai permintaan untuk DynamoDB
- Apa Itu AWS Backup?
- Apa Itu AWS Step Functions?
- Apa itu AWS Elastic Disaster Recovery
- AWS Elastic Disaster Recovery

Contoh terkait:

- Well-Architected Lab: Pengujian Pencadangan dan Pemulihan Data

REL 10. Bagaimana cara menggunakan isolasi kesalahan untuk melindungi beban kerja?

Batas isolasi kesalahan membatasi efek kegagalan di dalam beban kerja hingga jumlah komponen yang terbatas. Komponen di luar batas ini tidak terpengaruh oleh kegagalan tersebut. Menggunakan beberapa batas isolasi kesalahan, Anda dapat membatasi dampak pada beban kerja Anda.

Praktik terbaik

- REL10-BP01 Melakukan deployment beban kerja ke beberapa lokasi
- REL10-BP02 Memilih lokasi yang sesuai untuk deployment multilokasi
- REL10-BP03 Mengotomatiskan pemulihan untuk komponen yang dibatasi dalam satu lokasi
- REL10-BP04 Menggunakan arsitektur bulkhead untuk membatasi cakupan dampak

REL10-BP01 Melakukan deployment beban kerja ke beberapa lokasi

Distribusikan sumber daya dan data beban kerja ke beberapa Zona Ketersediaan atau, jika diperlukan, ke beberapa Wilayah AWS. Lokasi tersebut dapat beragam sesuai kebutuhan.

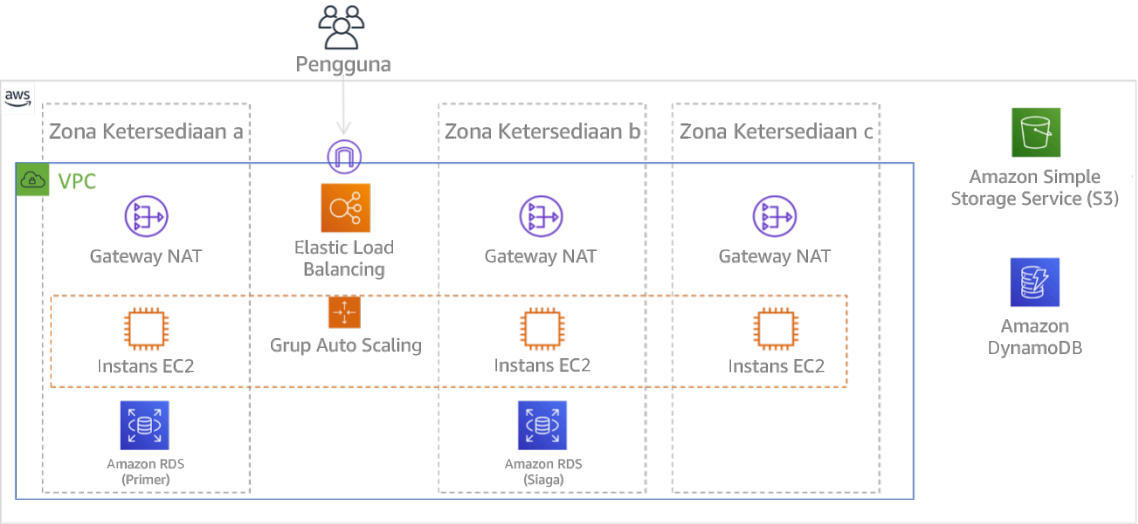
Salah satu prinsip dasar untuk desain layanan di AWS adalah menghindari titik kegagalan tunggal dalam infrastruktur fisik yang mendasarinya. Hal ini memotivasi kami untuk membangun sistem dan perangkat lunak yang menggunakan beberapa Zona Ketersediaan dan tahan terhadap kegagalan dari satu zona. Dengan cara yang serupa, sistem dibangun agar tahan terhadap kegagalan dari satu simpul komputasi, satu volume penyimpanan, atau satu instans basis data. Ketika membangun sistem yang mengandalkan komponen redundan, penting untuk memastikan bahwa komponen dapat beroperasi secara independen, dan dalam kasus Wilayah AWS, secara otomatis. Manfaat yang diperoleh dari kalkulasi ketersediaan teoretis dengan komponen redundan hanya valid jika dapat dibuktikan kebenarannya.

Zona Ketersediaan (AZ)

Wilayah AWS terdiri atas beberapa Zona Ketersediaan yang dirancang agar menjadi independen satu sama lain. Setiap Zona Ketersediaan dipisahkan oleh jarak fisik yang cukup dari zona lain untuk menghindari skenario kegagalan terkait karena bahaya lingkungan seperti kebakaran, banjir, dan tornado. Setiap Zona Ketersediaan juga memiliki infrastruktur fisik independen: koneksi khusus ke daya utilitas, sumber daya cadangan mandiri, layanan mekanis independen, dan konektivitas jaringan independen di dalam dan di luar Zona Ketersediaan. Desain ini membatasi kesalahan dalam satu sistem hingga hanya satu AZ yang terdampak. Meskipun terpisah secara geografis, Zona Ketersediaan berada di wilayah yang sama yang memungkinkan jaringan dengan latensi rendah dan

throughput tinggi. Seluruh Wilayah AWS (di semua Zona Ketersediaan, terdiri atas beberapa pusat data yang independen secara fisik) dapat dibuat menjadi target deployment logika tunggal untuk beban kerja, termasuk kemampuan untuk mereplikasi data secara sinkron (misalnya antarbasis data). Hal ini memungkinkan Anda untuk menggunakan Zona Ketersediaan dalam konfigurasi aktif/aktif atau aktif/siaga.

Zona Ketersediaan bersifat independen, dan oleh karena itu ketersediaan beban kerja meningkat saat beban kerja dirancang untuk menggunakan beberapa zona. Beberapa layanan AWS (termasuk bidang data instans Amazon EC2) di-deploy sebagai layanan zonal yang ketat dan memiliki sifat yang sama dengan Zona Ketersediaan tempatnya berada. Instans Amazon EC2 di AZ lainnya tidak akan terdampak dan tetap berfungsi. Dengan cara yang serupa, jika kesalahan di Zona Ketersediaan menyebabkan basis data Amazon Aurora gagal, instans Aurora replika baca di AZ yang tidak terdampak dapat dipindahkan ke AZ utama secara otomatis. Sebaliknya, layanan AWS regional seperti Amazon DynamoDB secara internal menggunakan beberapa Zona Ketersediaan dalam konfigurasi aktif/aktif guna mencapai tujuan desain ketersediaan untuk layanan tersebut, tanpa perlu mengonfigurasi penempatan AZ.



Gambar 9: Arsitektur multitingkat di-deploy di tiga Zona Ketersediaan. Perhatikan bahwa Amazon S3 dan Amazon DynamoDB selalu Multi-AZ secara otomatis. ELB juga di-deploy ke tiga zona.

Ketika umumnya bidang kendali AWS memberikan kemampuan untuk mengelola sumber daya di seluruh Wilayah (beberapa Zona Ketersediaan), bidang kendali tertentu (termasuk Amazon EC2 dan Amazon EBS) memiliki kemampuan untuk memfilter hasil hingga satu Zona Ketersediaan. Saat ini sudah dilakukan, permintaan hanya diproses di Zona Ketersediaan tertentu, mengurangi eksposur gangguan di Zona Ketersediaan lainnya. Contoh AWS CLI ini menggambarkan cara mendapatkan informasi instans Amazon EC2 hanya dari Zona Ketersediaan us-east-2c:

```
AWS ec2 describe-instances --filters Name=availability-zone,Values=us-east-2c
```

AWS Local Zones

AWS Local Zones bertindak serupa dengan Zona Ketersediaan dalam Wilayah AWS masing-masing sehingga dapat dipilih sebagai lokasi penempatan untuk sumber daya AWS zonal seperti subnet dan instans EC2. Hal yang membuatnya istimewa adalah mereka tidak berada di Wilayah AWS terkait, tetapi dekat dengan populasi yang besar, industri, dan pusat IT ketika tidak ada lagi Wilayah AWS. Namun zona-zona ini tetap mampu mempertahankan bandwidth tinggi, koneksi yang aman di antara beban kerja di zona lokal dan yang dijalankan di Wilayah AWS. Anda harus menggunakan AWS Local Zones untuk melakukan deployment beban kerja secara lebih dekat dengan pengguna untuk persyaratan latensi rendah.

Amazon Global Edge Network

Amazon Global Edge Network terdiri atas lokasi edge di kota seluruh dunia. Amazon CloudFront menggunakan jaringan ini untuk mengirimkan konten kepada pengguna akhir dengan latensi lebih rendah. AWS Global Accelerator memungkinkan Anda membuat titik akhir beban kerja di lokasi edge tersebut untuk memberikan onboarding ke jaringan global AWS yang dekat dengan pengguna. Amazon API Gateway memungkinkan titik akhir API yang dioptimasi edge menggunakan distribusi CloudFront agar klien mendapatkan akses melalui lokasi edge terdekat.

Wilayah AWS

Wilayah AWS dirancang agar menjadi otonom, akan tetapi, untuk menggunakan pendekatan multi-Wilayah, Anda perlu melakukan deployment salinan layanan yang dikhususkan untuk masing-masing Wilayah.

Pendekatan multi-Wilayah biasa digunakan untuk strategi pemulihan bencana yang memenuhi tujuan pemulihan saat satu peristiwa berskala besar terjadi. Lihat Rencanakan Pemulihan Bencana (DR) untuk informasi lebih lanjut tentang strategi ini. Namun, di sini kami lebih fokus pada ketersediaan, yang berupaya memberikan tujuan waktu aktif rata-rata dari waktu ke waktu. Untuk tujuan ketersediaan tinggi, arsitektur multi-wilayah umumnya akan dirancang menjadi aktif/aktif, dengan setiap salinan layanan (di wilayah masing-masing) yang aktif (permintaan layanan).

> **Rekomendasi**
>
> Tujuan ketersediaan untuk sebagian besar beban kerja dapat dipenuhi menggunakan strategi Multi-AZ dalam satu Wilayah AWS. Pertimbangkan arsitektur multi-Wilayah hanya saat beban

kerja memiliki persyaratan ketersediaan yang sangat tinggi, atau tujuan bisnis lain yang memerlukan arsitektur multi-Wilayah.

AWS memberikan kemampuan untuk mengoperasikan layanan lintas wilayah. Misalnya, AWS menyediakan replikasi data asinkron yang berkelanjutan menggunakan Replikasi Amazon Simple Storage Service (Amazon S3), Replika Baca Amazon RDS (termasuk Replika Baca Aurora), dan Tabel Global Amazon DynamoDB. Dengan replikasi berkelanjutan, versi data tersedia untuk penggunaan segera di setiap Wilayah aktif.

Dengan menggunakan AWS CloudFormation, Anda dapat menentukan infrastruktur dan melakukan deployment secara konsisten di seluruh Akun AWS dan seluruh Wilayah AWS. AWS CloudFormation StackSets meningkatkan fungsionalitas ini dengan memungkinkan Anda untuk membuat, memperbarui, atau menghapus tumpukan AWS CloudFormation di seluruh akun atau wilayah dalam satu kali operasi. Untuk deployment instans Amazon EC2, AMI (Amazon Machine Image) digunakan untuk memasok informasi seperti konfigurasi perangkat keras dan perangkat lunak yang diinstal. Anda dapat mengimplementasikan pipeline Amazon EC2 Image Builder yang membuat AMI yang diperlukan dan menyalinnya ke wilayah aktif. Hal ini memastikan AMI Emas memiliki segala yang dibutuhkan untuk melakukan deployment dan menskalakan beban kerja di setiap wilayah baru.

Untuk merutekan lalu lintas, Amazon Route 53 dan AWS Global Accelerator mengaktifkan definisi kebijakan yang menentukan titik akhir wilayah aktif yang dituju pengguna. Dengan Global Accelerator, Anda dapat mengatur panggilan lalu lintas untuk mengontrol persentase lalu lintas yang diarahkan ke setiap titik akhir aplikasi. Route 53 mendukung pendekatan persentase ini, dan juga beberapa kebijakan lain yang tersedia, termasuk kebijakan berdasarkan latensi dan geoproksimitas. Global Accelerator secara otomatis memanfaatkan jaringan server edge AWS yang luas, untuk mengarahkan lalu lintas ke pusat jaringan AWS secepatnya, sehingga menghasilkan latensi permintaan yang lebih rendah.

Semua kemampuan ini dioperasikan untuk menjaga setiap otonomi Wilayah. Ada beberapa pengecualian untuk pendekatan ini, termasuk layanan yang menyediakan pengiriman edge global, (seperti Amazon CloudFront dan Amazon Route 53), serta dengan bidang kendali untuk layanan AWS Identity and Access Management (IAM). Sebagian besar layanan dioperasikan sepenuhnya dalam satu Wilayah.

Pusat data on-premise

Rancang pengalaman hybrid jika memungkinkan, untuk beban kerja yang dijalankan di pusat data on-premise. AWS Direct Connect menyediakan koneksi jaringan khusus dari premise Anda ke AWS sehingga Anda dapat menjalankan beban kerja di kedua sistem.

Opsi lainnya adalah menjalankan layanan dan infrastruktur AWS on-premise dengan menggunakan AWS Outposts. AWS Outposts adalah layanan terkelola penuh yang memperluas infrastruktur AWS, layanan AWS, API, dan alat untuk pusat data. Infrastruktur perangkat keras yang sama yang digunakan di AWS Cloud juga diinstal di pusat data. Selanjutnya, AWS Outposts dihubungkan ke Wilayah AWS yang terdekat. Anda dapat menggunakan AWS Outposts untuk mendukung beban kerja yang memiliki latensi rendah atau persyaratan pemrosesan data lokal.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Gunakan beberapa Zona Ketersediaan dan Wilayah AWS. Distribusikan sumber daya dan data beban kerja ke beberapa Zona Ketersediaan atau, jika diperlukan, ke beberapa Wilayah AWS. Lokasi tersebut dapat beragam sesuai kebutuhan.
  - Layanan regional di-deploy secara permanen di seluruh Zona Ketersediaan.
    - Ini termasuk Amazon S3, Amazon DynamoDB, dan AWS Lambda (saat tidak terhubung ke VPC)
  - Lakukan deployment kontainer, instans, dan beban kerja berdasarkan fungsi ke dalam beberapa Zona Ketersediaan. Gunakan penyimpanan data multi-zona, termasuk cache. Gunakan fitur EC2 Auto Scaling, penempatan tugas ECS, dan konfigurasi fungsi AWS Lambda saat menjalankan VPC dan klaster ElastiCache.
    - Gunakan subnet yang berada di Zona Ketersediaan terpisah saat melakukan deployment grup Auto Scaling.
      - Misalnya: Mendistribusikan instans di seluruh Zona Ketersediaan
      - Strategi penempatan tugas Amazon ECS
      - Mengonfigurasi fungsi AWS Lambda untuk mengakses sumber daya di Amazon VPC
      - Memilih Zona Ketersediaan dan Wilayah
    - Gunakan subnet di Zona Ketersediaan terpisah saat melakukan deployment grup Auto Scaling.
      - Misalnya: Mendistribusikan instans di seluruh Zona Ketersediaan
    - Gunakan parameter penempatan tugas ECS, yang menentukan grup subnet DB.
      - Strategi penempatan tugas Amazon ECS

    - Gunakan subnet di beberapa Zona Ketersediaan saat mengonfigurasi fungsi untuk dijalankan di VPC.
        - Mengonfigurasi fungsi AWS Lambda untuk mengakses sumber daya di Amazon VPC
    - Gunakan beberapa Zona Ketersediaan dengan klaster ElastiCache.
        - Memilih Zona Ketersediaan dan Wilayah
- Jika beban kerja harus di-deploy di beberapa Wilayah, pilih strategi multi-Wilayah. Sebagian besar kebutuhan keandalan dapat dipenuhi dalam satu Wilayah AWS menggunakan strategi multi-Zona Ketersediaan. Gunakan strategi multi-Wilayah jika diperlukan untuk memenuhi kebutuhan bisnis.
    - AWS re:Invent 2018: Pola Arsitektur untuk Aplikasi Aktif-Aktif Multi-Wilayah (ARC209-R2)
        - Mencadangkan ke Wilayah AWS lain dapat menambah lapisan jaminan lain bahwa data akan tersedia saat dibutuhkan.
        - Beberapa beban kerja memiliki persyaratan regulasi yang memerlukan penggunaan strategi multi-Wilayah.
- Evaluasi AWS Outposts untuk beban kerja. Jika beban kerja memerlukan latensi rendah ke pusat data on-premise atau memiliki persyaratan pemrosesan data lokal. Jalankan layanan dan infrastruktur AWS on premise menggunakan AWS Outposts
    - Apa itu AWS Outposts?
- Tentukan apakah AWS Local Zones membantu menyediakan layanan untuk pengguna. Jika Anda memiliki persyaratan latensi rendah, periksa apakah AWS Local Zones berada dekat dengan pengguna. Jika iya, manfaatkan hal tersebut untuk melakukan deployment beban kerja dengan lebih dekat ke pengguna tersebut.
    - Pertanyaan Umum AWS Local Zones

Sumber daya

Dokumen terkait:

- Infrastruktur Global AWS
- Pertanyaan Umum AWS Local Zones
- Strategi penempatan tugas Amazon ECS
- Memilih Zona Ketersediaan dan Wilayah
- Misalnya: Mendistribusikan instans di seluruh Zona Ketersediaan
- Tabel Global: Replikasi Multi-Wilayah dengan DynamoDB

- Menggunakan basis data Amazon Aurora
- Membuat Aplikasi Multi-Wilayah dengan seri blog Layanan AWS
- Apa itu AWS Outposts?

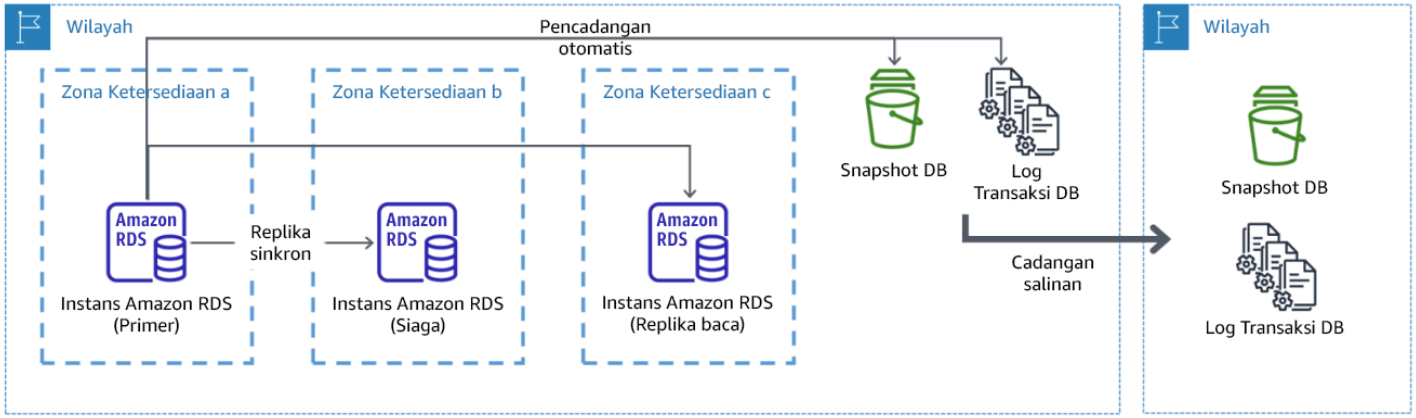
Video terkait:

- AWS re:Invent 2018: Pola Arsitektur untuk Aplikasi Aktif-Aktif Multi-Wilayah (ARC209-R2)
- AWS re:Invent 2019: Inovasi dan operasi infrastruktur jaringan global AWS (NET339)

REL10-BP02 Memilih lokasi yang sesuai untuk deployment multilokasi

Hasil yang diinginkan:

Untuk ketersediaan tinggi, selalu (jika memungkinkan) lakukan deployment komponen beban kerja ke beberapa Zona Ketersediaan (AZ), seperti yang ditampilkan dalam Gambar 10. Untuk beban kerja dengan persyaratan ketahanan yang sangat tinggi, evaluasikan dengan cermat opsi untuk arsitektur multi-Wilayah.



Gambar 10: Deployment basis data multi-AZ yang tangguh dengan pencadangan ke Wilayah AWS lainnya

Antipola umum:

- Memilih untuk merancang arsitektur multi-Wilayah saat arsitektur multi-AZ dapat memenuhi persyaratan.
- Tidak memperhitungkan dependensi antarkomponen aplikasi jika ketahanan dan persyaratan multilokasi antarkomponen tersebut berbeda.

Manfaat menerapkan praktik terbaik ini:

Untuk ketahanan, Anda harus menggunakan pendekatan yang membangun lapisan pertahanan. Satu lapisan melindungi terhadap gangguan yang lebih kecil dan lebih umum dengan membangun arsitektur yang memiliki ketersediaan tinggi menggunakan beberapa AZ. Lapisan pertahanan lainnya ditujukan untuk memberikan perlindungan terhadap peristiwa langka seperti bencana alam yang meluas dan gangguan tingkat Wilayah. Lapisan kedua ini melibatkan perancangan aplikasi agar menjangkau beberapa Wilayah AWS.

- Perbedaan antara ketersediaan 99,5% dan ketersediaan 99,99% adalah lebih dari 3,5 jam per bulan. Ketersediaan beban kerja yang diharapkan hanya dapat mencapai “empat angka sembilan” jika berada dalam beberapa AZ.
- Dengan menjalankan beban kerja di beberapa AZ, Anda dapat mengisolasi kesalahan dalam daya, pendinginan, dan jaringan, serta sebagian besar bencana alam seperti kebakaran dan banjir.
- Mengimplementasikan strategi multi-Wilayah untuk beban kerja membantu melindunginya dari bencana alam yang menjangkau dan memengaruhi wilayah geografis yang luas di suatu negara, atau kesalahan teknis yang mencakup seluruh Wilayah. Perhatikan bahwa mengimplementasikan arsitektur multi-Wilayah dapat menjadi sangat kompleks, dan biasanya tidak diperlukan untuk sebagian besar beban kerja.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

Untuk peristiwa bencana yang didasarkan pada gangguan atau hilangnya sebagian dari satu Zona Ketersediaan, mengimplementasikan beban kerja yang memiliki ketersediaan tinggi di beberapa Zona Ketersediaan dalam satu Wilayah AWS dapat membantu mitigasi bencana alam dan teknis. Setiap Wilayah AWS terdiri atas beberapa Zona Ketersediaan, masing-masing diisolasi dari kesalahan di zona lain dan dipisahkan oleh jarak yang cukup. Namun, untuk peristiwa bencana yang menyertakan risiko hilangnya beberapa komponen Zona Ketersediaan, yang jaraknya cukup jauh satu sama lain, Anda harus mengimplementasikan opsi pemulihan bencana untuk memitigasi kesalahan dalam cakupan Wilayah. Untuk beban kerja yang memerlukan ketahanan sangat tinggi (infrastruktur yang sangat penting, aplikasi terkait kesehatan, infrastruktur sistem keuangan, dll.), strategi multi-Wilayah mungkin diperlukan.

Langkah Implementasi

1. Evaluasikan beban kerja dan tentukan apakah ketahanan yang diperlukan dapat dipenuhi oleh pendekatan multi-AZ (satu Wilayah AWS), atau apakah pendekatan multi-Wilayah diperlukan. Mengimplementasikan arsitektur multi-Wilayah untuk memenuhi persyaratan tersebut akan menimbulkan kompleksitas tambahan, dengan demikian pertimbangkan secara cermat kasus penggunaan Anda dan persyaratannya. Persyaratan ketahanan dapat hampir selalu dipenuhi menggunakan satu Wilayah AWS. Pertimbangkan persyaratan yang memungkinkan berikut saat menentukan apakah Anda perlu menggunakan beberapa Wilayah:
   a. Pemulihan Bencana (DR): Untuk peristiwa bencana yang didasarkan pada gangguan atau kehilangan sebagian dari satu Zona Ketersediaan, mengimplementasikan beban kerja yang memiliki ketersediaan tinggi di beberapa Zona Ketersediaan dalam satu Wilayah AWS dapat membantu mitigasi bencana alam dan teknis. Untuk peristiwa bencana yang menyertakan risiko kehilangan beberapa komponen Zona Ketersediaan, yang jaraknya cukup jauh satu sama lain, Anda harus mengimplementasikan pemulihan bencana di seluruh Wilayah untuk memitigasi bencana alam atau kesalahan teknis dalam cakupan Wilayah.
   b. Ketersediaan tinggi (HA): Arsitektur multi-Wilayah (menggunakan beberapa AZ di setiap Wilayah) dapat digunakan untuk mencapai ketersediaan yang lebih tinggi dari empat angka 9 (> 99,99%).
   c. Pelokalan tumpukan: Saat melakukan deployment beban kerja ke audiens global, Anda dapat melakukan deployment tumpukan yang dilokalkan di Wilayah AWS yang berbeda untuk melayani audiens di Wilayah tersebut. Pelokalan dapat mencakup bahasa, mata uang, dan jenis data yang disimpan.
   d. Proksimitas kepada pengguna: Saat melakukan deployment beban kerja ke audiens global, Anda dapat mengurangi latensi dengan melakukan deployment tumpukan di Wilayah AWS yang dekat dengan tempat pengguna akhir.
   e. Residensi data: Beberapa beban kerja bergantung pada persyaratan residensi data, ketika data dari pengguna tertentu harus tetap berada dalam batasan negara tertentu. Berdasarkan regulasi dalam pertanyaan, Anda dapat memilih untuk melakukan deployment seluruh tumpukan, atau datanya saja, ke Wilayah AWS dalam batas tersebut.
2. Berikut beberapa contoh fungsionalitas multi-AZ yang disediakan oleh layanan AWS:
   a. Untuk melindungi beban kerja menggunakan EC2 atau ECS, lakukan deployment Elastic Load Balancer di depan sumber daya komputasi. Selanjutnya, Elastic Load Balancing menyediakan solusi untuk mendeteksi instans di zona yang kondisinya tidak baik dan merutekan lalu lintas ke zona yang kondisinya baik.

   i. Mulai menggunakan Application Load Balancers

   ii. Mulai menggunakan Penyeimbang Beban Jaringan

b. Dalam kasus instans EC2 yang menjalankan perangkat lunak komersial siap pakai yang tidak mendukung penyeimbangan beban, Anda dapat mencapai bentuk toleransi kesalahan dengan mengimplementasikan metodologi pemulihan bencana multi-AZ.

   i. the section called “REL13-BP02 Menggunakan strategi pemulihan yang ditentukan untuk memenuhi sasaran pemulihan”

c. Untuk tugas Amazon ECS, lakukan deployment secara merata di tiga AZ untuk mencapai keseimbangan ketersediaan dan biaya.

   i. Praktik terbaik ketersediaan Amazon ECS | Kontainer

d. Untuk non-Aurora Amazon RDS, Anda dapat memilih Multi-AZ sebagai opsi konfigurasi. Saat instans basis data utama mengalami kegagalan, Amazon RDS secara otomatis mendorong basis data standby untuk menerima lalu lintas di zona ketersediaan lainnya. Replika baca multi-Wilayah juga dapat dibuat untuk meningkatkan ketahanan.

   i. Deployment Multi AZ Amazon RDS

   ii. Membuat replika baca di Wilayah AWS yang berbeda

3. Berikut beberapa contoh fungsionalitas multi-Wilayah yang disediakan oleh layanan AWS:

a. Untuk beban kerja Amazon S3, ketika ketersediaan multi-AZ disediakan secara otomatis oleh layanan, pertimbangkan Poin Akses Multi-Wilayah jika deployment multi-Wilayah diperlukan.

   i. Poin Akses Multi-Wilayah di Amazon S3

b. Untuk tabel DynamoDB, ketika ketersediaan multi-AZ disediakan secara otomatis oleh layanan, Anda dapat mengonversi tabel yang ada ke tabel global dengan mudah untuk memperoleh manfaat dari beberapa wilayah.

   i. Konversi Tabel Amazon DynamoDB Wilayah Tunggal menjadi Tabel Global

c. Jika beban kerja didahului oleh Application Load Balancers atau Penyeimbang Beban Jaringan, gunakan AWS Global Accelerator untuk meningkatkan ketersediaan aplikasi dengan mengarahkan lalu lintas ke beberapa wilayah yang memiliki titik akhir dengan kondisi baik.

   i. Titik akhir untuk akselerator standar di AWS Global Accelerator - AWS Global Accelerator (amazon.com)

d. Untuk aplikasi yang memanfaatkan AWS EventBridge, pertimbangkan bus lintas Wilayah untuk meneruskan peristiwa ke Wilayah lain yang dipilih.

   i. Mengirim dan menerima peristiwa Amazon EventBridge di antara beberapa Wilayah AWS

e. Untuk basis data Amazon Aurora, pertimbangkan basis data global Aurora, yang menjangkau beberapa wilayah AWS. Klaster yang sudah ada juga dapat diubah untuk menambahkan Wilayah baru.
   i. Mulai menggunakan basis data global Amazon Aurora
f. Jika beban kerja mencakup kunci enkripsi AWS Key Management Service (AWS KMS), pertimbangkan apakah kunci multi-Wilayah sesuai untuk aplikasi.
   i. Kunci Multi-Wilayah di AWS KMS
g. Untuk fitur layanan AWS lainnya, lihat seri blog ini di Seri Membuat Aplikasi Multi-Wilayah dengan Layanan AWS

Tingkat upaya untuk Rencana Implementasi: Sedang hingga Tinggi

Sumber daya

Dokumen terkait:

- Seri Membuat Aplikasi Multi-Wilayah dengan Layanan AWS
- Arsitektur Pemulihan Bencana (DR) di AWS, Bagian IV: Multi-situs Aktif/Aktif)
- Infrastruktur Global AWS
- Pertanyaan Umum AWS Local Zones
- Arsitektur Pemulihan Bencana (DR) di AWS, Bagian I: Strategi untuk Pemulihan di Cloud
- Pemulihan bencana di cloud tidak sama dengan biasanya
- Tabel Global: Replikasi Multi-Wilayah dengan DynamoDB

Video terkait:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- Auth0: Arsitektur Ketersediaan Tinggi Multi-Wilayah yang Menskalakan hingga 1,5B+ Login Sebulan dengan failover otomatis

Contoh terkait:

- Arsitektur Pemulihan Bencana (DR) di AWS, Bagian I: Strategi untuk Pemulihan di Cloud
- DTCC mencapai ketangguhan yang lebih tinggi dari yang dapat dilakukan di on-premise

- Expedia Group menggunakan arsitektur multi-Wilayah dan multi-Zona Ketersediaan dengan layanan DNS eksklusif untuk menambah ketangguhan pada aplikasi
- Uber: Pemulihan Bencana untuk Kafka Multi-Wilayah
- Netflix: Aktif-Aktif untuk Ketahanan Multi-Wilayah
- Cara kami membangun Residensi Data untuk Atlassian Cloud
- Intuit TurboTax dijalankan di dua Wilayah

REL10-BP03 Mengotomatiskan pemulihan untuk komponen yang dibatasi dalam satu lokasi

Jika komponen beban kerja hanya dapat dijalankan di satu Zona Ketersediaan atau di pusat data on-premise, implementasikan kemampuan untuk membangun kembali beban kerja sepenuhnya dalam lingkup tujuan pemulihan yang telah ditetapkan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Jika praktik terbaik untuk melakukan deployment beban kerja ke beberapa lokasi tidak memungkinkan karena pembatasan teknologi, Anda harus mengimplementasikan jalur alternatif menuju ketahanan. Anda harus mengotomatiskan kemampuan untuk membuat ulang infrastruktur yang dibutuhkan, melakukan deployment ulang aplikasi, dan membuat ulang data yang diperlukan untuk kasus ini.

Misalnya, Amazon EMR meluncurkan semua simpul untuk klaster tertentu yang tersedia dalam Zona Ketersediaan yang sama karena menjalankan klaster di zona yang sama dapat meningkatkan kinerja aliran tugas berkat tingkat akses data yang lebih tinggi. Jika komponen ini tidak dibutuhkan untuk ketahanan beban kerja, Anda harus mencari cara lain untuk melakukan deployment klaster dan datanya. Selain itu, untuk Amazon EMR, Anda harus menyediakan redundansi selain dengan menggunakan Multi-AZ. Anda dapat menyediakan beberapa simpul. Dengan menggunakan Sistem File EMR (EMRFS), data di EMR dapat dipulihkan di Amazon S3, yang kemudian dapat direplikasi di beberapa Zona Ketersediaan atau Wilayah AWS.

Dengan cara yang serupa, Amazon Redshift secara default menyediakan klaster dalam Zona Ketersediaan yang dipilih secara acak dalam Wilayah AWS yang Anda pilih. Semua simpul klaster disediakan dalam zona yang sama.

Untuk beban kerja stateful berbasis server yang di-deploy ke pusat data on-premise, Anda dapat menggunakan AWS Elastic Disaster Recovery untuk melindungi beban kerja Anda di AWS.

Jika Anda sudah di-host di AWS, Anda dapat menggunakan Elastic Disaster Recovery untuk melindungi beban kerja Anda di Wilayah atau Zona Ketersediaan alternatif. Elastic Disaster Recovery menggunakan replikasi tingkat blok secara berkelanjutan ke area staging ringan untuk memberikan pemulihan aplikasi on-premise dan aplikasi berbasis cloud secara cepat dan andal.

Langkah implementasi

1. Implementasikan pemulihan mandiri. Lakukan deployment instans atau kontainer dengan menggunakan penskalaan otomatis jika memungkinkan. Jika penskalaan otomatis tidak dapat digunakan, gunakan pemulihan otomatis untuk instans EC2 atau implementasikan otomatisasi pemulihan mandiri berdasarkan Amazon EC2 atau peristiwa siklus hidup kontainer ECS.
   - Gunakan grup Amazon EC2 Auto Scaling untuk instans atau beban kerja kontainer yang tidak memiliki persyaratan untuk alamat IP instans tunggal, alamat IP pribadi, alamat IP Elastis, dan metadata instans.
     - Data pengguna templat peluncuran dapat digunakan untuk mengimplementasikan otomatisasi yang dapat memulihkan sebagian besar beban kerja secara mandiri.
   - Gunakan pemulihan otomatis instans Amazon EC2untuk beban kerja yang memerlukan instans tunggal alamat ID, alamat IP pribadi, alamat IP elastis, dan instans metadata.
     - Pemulihan Otomatis akan mengirimkan peringatan status pemulihan kepada topik SNS saat kegagalan instans terdeteksi.
   - Gunakan peristiwa siklus hidup instans Amazon EC2 atau peristiwa Amazon ECS untuk mengotomatiskan pemulihan mandiri jika penskalaan otomatis atau pemulihan EC2 tidak dapat digunakan.
     - Gunakan peristiwa untuk memicu otomatisasi yang akan memulihkan komponen Anda berdasarkan proses logika yang diperlukan.
   - Lindungi beban kerja stateful yang dibatasi di satu lokasi menggunakan AWS Elastic Disaster Recovery.

Sumber daya

Dokumen terkait:

- Peristiwa Amazon ECS
- Pengait siklus hidup Amazon EC2 Auto Scaling
- Pulihkan instans Anda.
- Penskalaan otomatis layanan

- Apa Itu Amazon EC2 Auto Scaling?
- AWS Elastic Disaster Recovery

REL10-BP04 Menggunakan arsitektur bulkhead untuk membatasi cakupan dampak

Implementasikan arsitektur bulkhead (juga disebut sebagai arsitektur berbasis sel) untuk membatasi efek kegagalan dalam beban kerja hingga jumlah komponen yang terbatas.

Hasil yang diinginkan: Arsitektur berbasis sel menggunakan beberapa instans terisolasi beban kerja, di mana setiap instans disebut sebagai sel. Setiap sel bersifat mandiri, tidak berbagi status dengan sel lain, dan menangani subset permintaan beban kerja secara keseluruhan. Hal ini mengurangi potensi dampak kegagalan, seperti pembaruan perangkat lunak yang buruk, ke satu sel individu dan permintaan yang diprosesnya. Jika beban kerja menggunakan 10 sel untuk melayani 100 permintaan, ketika kegagalan terjadi, 90% dari seluruh permintaan akan tidak dipengaruhi oleh kegagalan.

Antipola umum:

- Membiarkan sel bertumbuh tanpa batas.
- Menerapkan pembaruan kode atau deployment ke semua sel pada waktu yang sama.
- Berbagi status atau komponen antara sel (dengan pengecualian lapisan router).
- Menambahkan bisnis yang kompleks atau mengarahkan rute logika ke lapisan router.
- Tidak meminimalkan interaksi lintas sel.

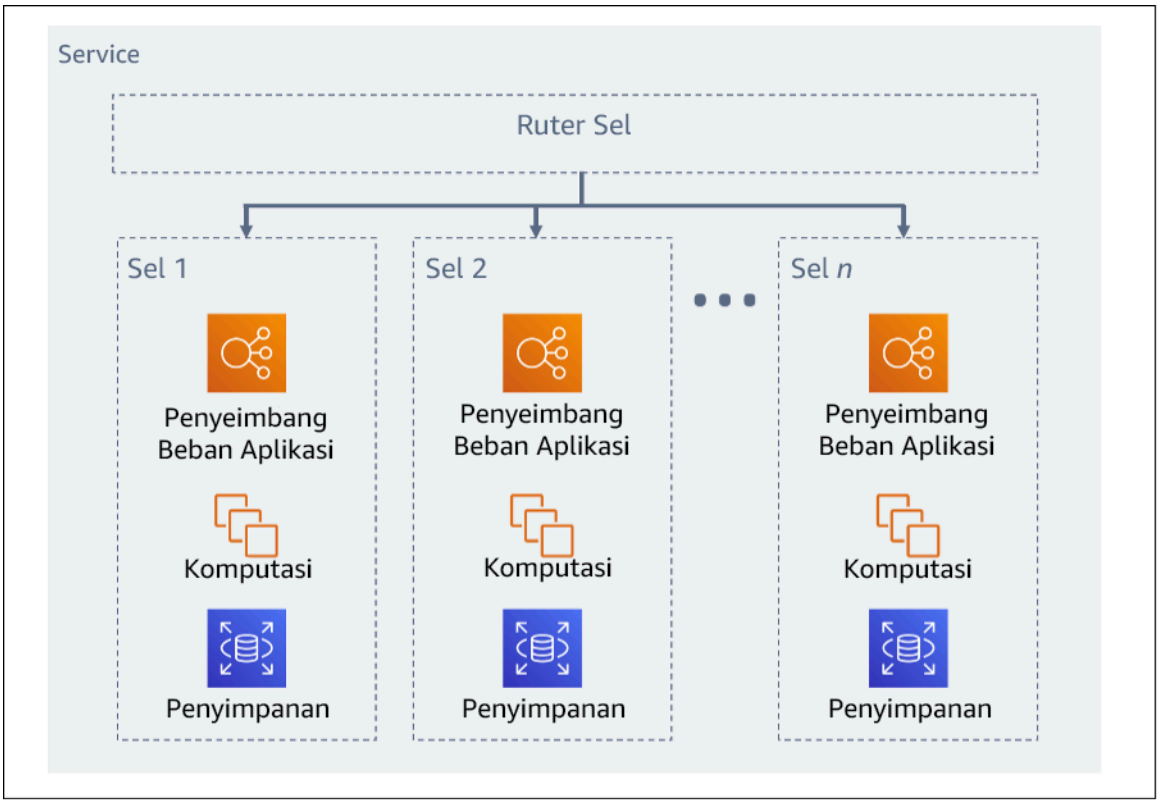
Manfaat menjalankan praktik terbaik ini: Dengan arsitektur berbasis sel, banyak jenis kegagalan umum dibatasi dalam lingkup sel itu sendiri, yang memberikan isolasi kesalahan tambahan. Batas kesalahan ini dapat memberikan ketangguhan terhadap jenis kegagalan yang sulit dibatasi, seperti deployment kode yang gagal atau permintaan yang rusak atau memicu mode kegagalan tertentu (juga disebut sebagai permintaan poison pill).

Panduan implementasi

Di kapal, bulkhead memastikan kebocoran lambung kapal hanya dibatasi dalam satu bagian lambung kapal saja. Di sistem yang kompleks, pola ini sering kali direplikasi untuk memungkinkan isolasi kesalahan. Batas isolasi kesalahan membatasi efek kegagalan di dalam beban kerja hingga jumlah komponen yang terbatas. Komponen di luar batas ini tidak terpengaruh oleh kegagalan tersebut. Menggunakan beberapa batas isolasi kesalahan, Anda dapat membatasi dampak pada beban kerja

Anda. Di AWS, pelanggan dapat menggunakan beberapa Zona Ketersediaan dan Wilayah untuk memberikan isolasi kesalahan, tetapi konsep isolasi kesalahan dapat diperluas ke arsitektur beban kerja juga.

Beban kerja secara keseluruhan adalah sel-sel yang dipartisi oleh kunci partisi. Kunci ini harus sesuai dengan grain layanan, atau cara alami beban kerja layanan dapat dibagi lebih lanjut dengan interaksi lintas sel yang minim. Contoh kunci partisi yakni ID pelanggan, ID sumber daya, atau parameter lainnya yang dapat diakses dengan mudah dalam sebagian besar panggilan API. Lapisan perutean sel mendistribusikan permintaan ke masing-masing sel berdasarkan kunci partisi dan menyampaikan satu titik akhir ke klien.



Gambar 11: Arsitektur berbasis sel

Langkah implementasi

Ketika mendesain arsitektur berbasis sel, ada beberapa pertimbangan desain untuk dipikirkan:

1. Kunci partisi: Pemilihan kunci partisi harus dipertimbangkan baik-baik.
    - Kunci ini harus sesuai dengan grain layanan, atau cara alami beban kerja layanan dapat dibagi lebih lanjut dengan interaksi lintas sel yang minim. Contohnya, `ID pelanggan` atau `ID sumber daya`.

- Kunci partisi harus tersedia dalam semua permintaan, baik secara langsung atau dengan cara yang dapat disimpulkan dengan pasti dan mudah oleh parameter lain.

2. Pemetaan sel persisten: Layanan sebelumnya dalam proses hanya boleh berinteraksi dengan satu sel selama siklus hidup sumber dayanya.
   - Bergantung pada beban kerjanya, strategi migrasi sel mungkin diperlukan untuk memigrasikan data dari satu sel ke yang lain. Kemungkinan skenario ketika migrasi sel mungkin diperlukan yakni jika sumber daya atau pengguna tertentu di beban kerja Anda menjadi terlalu besar dan memerlukan sel khusus.
   - Sel tidak boleh berbagi status atau komponen antara sel.
   - Oleh karena itu, interaksi lintas sel harus dihindari atau dijaga agar tetap minim, karena interaksi tersebut menimbulkan dependensi antara sel, sehingga mengurangi peningkatan isolasi kesalahan.
3. Lapisan router: Lapisan router adalah komponen bersama antara sel, oleh karena itu tidak dapat mengikuti strategi kompartementalisasi yang sama seperti sel.
   - Sebaiknya lapisan router mendistribusikan permintaan ke sel secara individu menggunakan algoritme pemetaan partisi dengan cara yang efisien secara komputasi, seperti menggabungkan fungsi hash kriptografi dan aritmetika modul untuk memetakan kunci partisi ke sel.
   - Untuk menghindari dampak multi-sel, lapisan perutean harus tetap sesederhana mungkin dan dapat diskalakan sehorizontal mungkin, yang memerlukan penghindaran logika bisnis kompleks di dalam lapisan ini. Hal ini memiliki manfaat tambahan mempermudah pemahaman ekspektasi perilakunya di setiap waktu, yang memberikan kemampuan untuk diuji secara menyeluruh. Sebagaimana dijelaskan oleh Colm MacCárthaigh dalam Reliability, constant work, and a good cup of coffee, desain sederhana dan pola kerja konstan menghasilkan sistem yang andal dan mengurangi anti-kerentanan.
4. Ukuran sel: Sel harus memiliki ukuran maksimum dan tidak boleh diizinkan untuk bertumbuh melampauinya.
   - Ukuran maksimum harus diidentifikasi dengan melakukan pengujian yang menyeluruh, sampai titik rusak tercapai dan margin pengoperasian yang aman ditetapkan. Untuk detail selengkapnya tentang cara mengimplementasikan praktik pengujian, lihat REL07-BP04 Menguji beban untuk beban kerja Anda
   - Beban kerja secara keseluruhan harus bertumbuh dengan menambahkan sel tambahan, sehingga beban kerja dapat diskalakan seiring peningkatan permintaan.
5. Strategi Multi-AZ atau Multi-Wilayah: Beberapa lapisan ketangguhan harus dimanfaatkan untuk melindungi dari berbagai macam domain kegagalan.

- Untuk ketahanan, Anda harus menggunakan pendekatan yang membangun lapisan pertahanan. Satu lapisan melindungi dari gangguan yang lebih kecil dan lebih umum dengan membangun arsitektur yang memiliki ketersediaan tinggi menggunakan beberapa AZ. Lapisan pertahanan lainnya ditujukan untuk memberikan perlindungan terhadap peristiwa langka seperti bencana alam yang meluas dan gangguan tingkat Wilayah. Lapisan kedua ini melibatkan perancangan aplikasi agar menjangkau beberapa Wilayah AWS. Mengimplementasikan strategi multi-Wilayah untuk beban kerja membantu melindunginya dari bencana alam yang menjangkau dan memengaruhi wilayah geografis yang luas di suatu negara, atau kesalahan teknis yang mencakup seluruh Wilayah. Perhatikan bahwa mengimplementasikan arsitektur multi-Wilayah dapat menjadi sangat kompleks, dan biasanya tidak diperlukan untuk sebagian besar beban kerja. untuk detail selengkapnya, lihat REL10-BP02 Memilih lokasi yang sesuai untuk deployment multilokasi.

6. Deployment kode: Strategi deployment kode bergiliran harus didahulukan dibandingkan deployment perubahan kode ke semua sel pada waktu yang sama.
   - Hal ini akan membantu meminimalkan potensi kegagalan pada beberapa sel karena deployment yang buruk atau kesalahan manusia. Untuk detail selengkapnya, lihat Mengotomatiskan deployment aman tanpa campur tangan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Sumber daya

Praktik Terbaik Terkait:

- REL07-BP04 Menguji beban untuk beban kerja Anda
- REL10-BP02 Memilih lokasi yang sesuai untuk deployment multilokasi

Dokumen terkait:

- Reliability, constant work, and a good cup of coffee
- AWS dan Kompartementalisasi
- Isolasi beban kerja menggunakan shuffle-sharding
- Mengotomatiskan deployment aman tanpa campur tangan

Video terkait:

- AWS re:Invent 2018: Menutup Lingkaran dan Membuka Pikiran: Cara Mengendalikan Sistem, Besar dan Kecil
- AWS re:Invent 2018: Cara AWS Meminimalkan Radius Dampak Kegagalan (ARC338)
- Shuffle-sharding: AWS re:Invent 2019: Memperkenalkan Pustaka Pengembang Amazon (DOP328)
- AWS Summit ANZ 2021 - Segala sesuatu gagal, setiap waktu: Mendesain agar memiliki ketangguhan

Contoh terkait:

- Well-Architected Lab - Isolasi kesalahan dengan shuffle sharding

REL 11. Bagaimana Anda mendesain beban kerja agar dapat bertahan jika terjadi kegagalan komponen?

Beban kerja dengan persyaratan untuk ketersediaan tinggi dan waktu rata-rata untuk pemulihan (MTTR) rendah harus didesain dan dikonfigurasi agar tangguh.

Praktik terbaik

- REL11-BP01 Memantau semua komponen beban kerja untuk mendeteksi kegagalan
- REL11-BP02 Melakukan failover ke sumber daya yang sehat
- REL11-BP03 Mengotomatisasi pemulihan di semua lapisan
- REL11-BP04 Mengandalkan bidang data dan bukan bidang kendali selama pemulihan
- REL11-BP05 Menggunakan stabilitas statis untuk mencegah perilaku bimodal
- REL11-BP06 Mengirimkan notifikasi ketika peristiwa memengaruhi ketersediaan
- REL11-BP07 Merancang produk Anda agar memenuhi target ketersediaan dan perjanjian tingkat layanan (SLA) waktu aktif

REL11-BP01 Memantau semua komponen beban kerja untuk mendeteksi kegagalan

Terus pantau kondisi beban kerja agar Anda dan sistem otomatis Anda langsung mengetahui penurunan kualitas atau kegagalan ketika muncul. Pantau indikator kinerja utama (KPI) berdasarkan nilai bisnis.

Semua mekanisme pemulihan dan penyembuhan harus dimulai dengan kemampuan untuk mendeteksi masalah secara cepat. Kegagalan teknis harus dideteksi terlebih dahulu sehingga dapat

diatasi. Namun, ketersediaan didasarkan pada kemampuan beban kerja Anda untuk menghadirkan nilai bisnis, sehingga indikator kinerja utama (KPI) yang mengukurnya perlu menjadi bagian dari strategi deteksi dan perbaikan Anda.

Hasil yang diinginkan: Komponen penting dari suatu beban kerja dipantau secara independen untuk mendeteksi dan memperingatkan adanya kegagalan pada saat dan di bagian mana kegagalan tersebut terjadi.

Antipola umum:

- Tidak ada alarm yang dikonfigurasi, sehingga pemadaman terjadi tanpa notifikasi.
- Alarm tersedia, tetapi pada ambang batas yang tidak menyediakan waktu yang cukup untuk bereaksi.
- Metrik tidak dikumpulkan cukup sering untuk memenuhi sasaran waktu pemulihan (RTO).
- Hanya antarmuka beban kerja yang terlihat oleh pelanggan yang aktif dipantau.
- Hanya mengumpulkan metrik teknis, dan mengabaikan metrik fungsi bisnis.
- Tidak ada metrik yang mengukur pengalaman pengguna beban kerja.
- Terlalu banyak pemantau yang dibuat.

Manfaat menjalankan praktik terbaik ini: Pemantauan yang sesuai di semua lapisan memungkinkan Anda menghemat waktu pemulihan karena berkurangnya waktu deteksi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Identifikasikan semua beban kerja yang akan ditinjau untuk pemantauan. Setelah Anda mengidentifikasi semua komponen beban kerja yang perlu dipantau, selanjutnya Anda perlu menentukan interval pemantauan. Interval pemantauan akan berdampak langsung pada seberapa cepat pemulihan dapat dimulai berdasarkan waktu yang diperlukan untuk mendeteksi kegagalan. Rata-rata waktu deteksi (MTTD) adalah lamanya waktu antara terjadinya kegagalan dan ketika operasi perbaikan dimulai. Daftar layanan harus luas dan lengkap.

Pemantauan harus mencakup semua lapisan tumpukan aplikasi termasuk aplikasi, platform, infrastruktur, dan jaringan.

Strategi pemantauan Anda harus mempertimbangkan dampak kegagalan abu-abu. Untuk detail lebih lanjut tentang kegagalan abu-abu, lihat Kegagalan abu-abu di laporan resmi Pola Ketangguhan Multi-AZ Lanjutan.

Langkah implementasi

- Interval pemantauan Anda bergantung pada seberapa cepat Anda harus pulih. Waktu pemulihan Anda didorong oleh waktu yang diperlukan untuk pulih, sehingga Anda harus menentukan frekuensi pengumpulan dengan cara menghitung waktu ini serta sasaran waktu pemulihan (RTO) Anda.
- Konfigurasikan pemantauan mendetail untuk komponen dan layanan terkelola.
  - Tentukan apakah pemantauan mendetail untuk instans EC2 dan Auto Scaling diperlukan. Pemantauan mendetail menyediakan metrik interval satu menit, sedangkan pemantauan default menyediakan metrik interval lima menit.
  - Tentukan apakah pemantauan yang ditingkatkan untuk RDS diperlukan. Pemantauan yang ditingkatkan menggunakan agen di instans RDS untuk memperoleh informasi bermanfaat tentang berbagai alur atau proses.
  - Tentukan persyaratan pemantauan komponen nirserver penting untuk Lambda, API Gateway, Amazon EKS, Amazon ECS, dan semua jenis penyeimbang beban.
  - Tentukan persyaratan pemantauan komponen penyimpanan untuk Amazon S3, Amazon FSx, Amazon EFS, dan Amazon EBS.
- Buat metrik kustom untuk mengukur indikator kinerja kunci (KPI) bisnis. Beban kerja mengimplementasikan fungsi-fungsi bisnis utama, yang harus digunakan sebagai KPI yang membantu mengidentifikasi kapan terjadinya masalah tidak langsung.
- Pantau pengalaman pengguna untuk mendeteksi kegagalan menggunakan canary pengguna. Pengujian transaksi sintetis (juga disebut pengujian canary, tetapi tidak sama dengan deployment canary) yang dapat menjalankan dan menyimulasikan perilaku pelanggan adalah salah satu proses pengujian yang paling penting. Jalankan pengujian ini secara konstan terhadap titik akhir beban kerja Anda dari beragam lokasi jarak jauh.
- Buat metrik kustom yang melacak pengalaman pengguna. Jika Anda dapat menginstrumentasi pengalaman pelanggan, Anda dapat menentukan saat pengalaman pelanggan mengalami degradasi.
- Atur alarm untuk mendeteksi saat ada bagian dari beban kerja Anda yang tidak berfungsi dengan baik, dan untuk menunjukkan kapan harus menskalakan sumber daya secara otomatis. Alarm dapat ditampilkan secara visual di dasbor, mengirimkan peringatan melalui Amazon SNS atau email, dan menggunakan Auto Scaling untuk menaikkan atau menurunkan skala sumber daya beban kerja.

- Buat dasbor untuk memvisualisasikan metrik Anda. Dasbor dapat digunakan untuk melihat tren, penyimpangan, dan indikator potensi masalah lainnya, atau menyediakan penanda untuk masalah yang ingin Anda selidiki.
- Buat pemantauan penelusuran terdistribusi untuk layanan Anda. Dengan pemantauan terdistribusi, Anda dapat memahami cara kerja aplikasi Anda dan layanan dasar dalam mengidentifikasi dan memecahkan akar masalah dan galat kinerja.
- Buat dasbor sistem pemantauan (menggunakan CloudWatch atau X-Ray) dan pengumpulan data di Wilayah dan akun terpisah.
- Buat integrasi untuk pemantauan Amazon Health Aware untuk memungkinkan visibilitas pemantauan ke sumber daya AWS yang mungkin mengalami degradasi. Untuk beban kerja yang penting untuk bisnis, solusi ini menyediakan akses ke peringatan proaktif dan waktu nyata untuk layanan AWS.

Sumber daya

Praktik terbaik terkait:

- Definisi Ketersediaan
- REL11-BP06 Mengirimkan Notifikasi ketika peristiwa memengaruhi ketersediaan

Dokumen terkait:

- Amazon CloudWatch Synthetics memungkinkan Anda untuk membuat canary pengguna
- Aktifkan atau Nonaktifkan Pemantauan Mendetail untuk Instans Anda
- Pemantauan yang Ditingkatkan
- Memantau Grup dan Instans Auto Scaling Anda Menggunakan Amazon CloudWatch
- Memublikasikan Metrik Kustom
- Menggunakan Alarm Amazon CloudWatch
- Menggunakan Dasbor CloudWatch
- Menggunakan Dasbor CloudWatch Lintas Akun dan Lintas Wilayah
- Menggunakan Penelusuran X-Ray Lintas Akun dan Lintas Wilayah
- Memahami ketersediaan
- Mengimplementasikan Amazon Health Aware (AHA)

Video terkait:

- Memitigasi kegagalan abu-abu

Contoh terkait:

- Lab Well-Architected: Level 300: Mengimplementasikan Pemeriksaan Kondisi dan Mengelola Dependensi untuk Meningkatkan Keandalan
- Lokakarya One Observability: Menjelajahi X-Ray

Alat terkait:

- CloudWatch
- CloudWatch X-Ray

REL11-BP02 Melakukan failover ke sumber daya yang sehat

Jika terjadi kegagalan sumber daya, sumber daya yang sehat harus terus melayani permintaan. Untuk kerusakan lokasi (seperti Zona Ketersediaan atau Wilayah AWS) pastikan Anda memiliki sistem untuk melakukan failover ke sumber daya yang sehat di lokasi yang tidak terkena gangguan.

Saat merancang layanan, distribusikan beban di seluruh sumber daya, Zona Ketersediaan, atau Wilayah. Oleh karena itu, kegagalan sumber daya individu atau gangguan dapat dimitigasi dengan mengalihkan lalu lintas ke sumber daya sehat yang masih ada. Pertimbangkan bagaimana layanan ditemukan dan dirutekan jika terjadi kegagalan.

Rancang layanan Anda dengan mempertimbangkan pemulihan kesalahan. Di AWS, kami merancang layanan untuk meminimalkan waktu untuk pulih dari kegagalan dan dampak terhadap data. Layanan kami utamanya menggunakan penyimpanan data yang mengenali permintaan hanya setelah disimpan dalam waktu lama di beberapa replika di dalam suatu Wilayah. Layanan dan sumber daya ini dibangun untuk menggunakan isolasi berbasis sel dan menggunakan isolasi kesalahan yang disediakan oleh Zona Ketersediaan. Kami banyak menggunakan otomatisasi di dalam prosedur operasional kami. Kami juga mengoptimalkan fungsionalitas “ganti dan mulai ulang” kami untuk pulih secara cepat dari gangguan.

Pola dan desain yang memungkinkan failover bervariasi untuk setiap layanan platform AWS. Banyak layanan terkelola native AWS adalah layanan yang secara native multi-Zona Ketersediaan (seperti

Lambda atau API Gateway). Layanan AWS lain (seperti EC2 dan EKS) memerlukan desain praktik terbaik khusus untuk mendukung failover sumber daya atau penyimpanan data di seluruh AZ.

Pemantauan harus disiapkan untuk memeriksa apakah sumber daya failover sehat, melacak kemajuan sumber daya yang melakukan failover, dan memantau pemulihan proses bisnis.

Hasil yang diinginkan: Sistem mampu secara otomatis atau manual menggunakan sumber daya baru untuk pulih dari degradasi.

Antipola umum:

- Perencanaan kegagalan bukan bagian dari fase perencanaan dan desain.
- RTO dan RPO tidak ditetapkan.
- Pemantauan yang tidak memadai untuk mendeteksi sumber daya yang gagal.
- Pemisahan domain kegagalan yang layak.
- Kegagalan Multi-Wilayah tidak dipertimbangkan.
- Deteksi kegagalan terlalu sensitif atau agresif saat memutuskan untuk melakukan failover.
- Tidak menguji atau memvalidasi desain failover.
- Melakukan otomatisasi pemulihan otomatis, tetapi tidak memberikan notifikasi bahwa pemulihan diperlukan.
- Kurangnya periode peredaman untuk menghindari gagal kembali yang terlalu cepat.

Manfaat menjalankan praktik terbaik ini: Anda dapat membangun sistem yang lebih tangguh yang mempertahankan keandalan saat mengalami kegagalan dengan melakukan degradasi secara mulus dan pulih dengan cepat.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Layanan AWS, seperti Elastic Load Balancing dan Amazon EC2 Auto Scaling, membantu mendistribusikan beban di seluruh sumber daya dan Zona Ketersediaan. Oleh karena itu, kegagalan sumber daya individu (seperti instans EC2) atau gangguan pada Zona Ketersediaan dapat dimitigasi dengan mengalihkan lalu lintas ke sumber daya sehat yang masih ada.

Untuk beban kerja multi-Wilayah, desainnya lebih rumit. Misalnya, replika baca lintas Wilayah memungkinkan Anda untuk melakukan deployment data ke beberapa Wilayah AWS. Namun, failover

masih diperlukan untuk mempromosikan replika baca ke primer kemudian mengarahkan lalu lintas Anda ke titik akhir baru. Amazon Route 53, Route 53 ARC, CloudFront, dan AWS Global Accelerator dapat membantu merutekan lalu lintas di seluruh Wilayah AWS.

Layanan AWS, seperti Amazon S3, Lambda, API Gateway, Amazon SQS, Amazon SNS, Amazon SES, Amazon Pinpoint, Amazon ECR, AWS Certificate Manager, EventBridge, atau Amazon DynamoDB, secara otomatis di-deploy ke beberapa Zona Ketersediaan oleh AWS. Jika terjadi kegagalan, layanan-layanan AWS ini secara otomatis merutekan lalu lintas ke lokasi yang sehat. Data disimpan secara redundan di beberapa Zona Ketersediaan dan tetap tersedia.

Untuk Amazon RDS, Amazon Aurora, Amazon Redshift, Amazon EKS, atau Amazon ECS, Multi-AZ adalah opsi konfigurasi. AWS dapat mengarahkan lalu lintas ke instans sehat jika failover dimulai. Tindakan failover ini dapat diambil oleh AWS atau sebagaimana diperlukan oleh pelanggan

Untuk instans Amazon EC2, Amazon Redshift, tugas Amazon ECS, atau pod Amazon EKS, Anda memilih Zona Ketersediaan mana untuk deployment. Untuk beberapa desain, Elastic Load Balancing memberikan solusi untuk mendeteksi instans di zona yang tidak sehat dan merutekan lalu lintas ke zona yang sehat. Elastic Load Balancing juga dapat merutekan lalu lintas ke komponen di pusat data on-premise Anda.

Untuk failover lalu lintas Multi-Wilayah, pengalihan rute dapat memanfaatkan Amazon Route 53, ARC, AWS Global Accelerator, Route 53 Private DNS for VPCs, atau CloudFront untuk menyediakan cara untuk menentukan domain internet dan menetapkan kebijakan perutean, termasuk pemeriksaan kondisi, untuk merutekan lalu lintas ke Wilayah yang sehat. AWS Global Accelerator menyediakan alamat IP statis yang bertindak sebagai titik masuk tetap ke aplikasi Anda, lalu merutekan ke titik akhir di Wilayah AWS yang Anda pilih, menggunakan jaringan global AWS, bukan internet, demi performa dan keandalan yang lebih baik.

Langkah implementasi

- Buat desain failover untuk semua aplikasi dan layanan yang sesuai. Isolasi setiap komponen arsitektur dan buat desain failover yang memenuhi RTO dan RPO untuk setiap komponen.
- Konfigurasikan lingkungan yang lebih rendah (seperti pengembangan atau pengujian) dengan semua layanan yang diharuskan memiliki rencana failover. Deploy solusi menggunakan infrastruktur sebagai kode (IaC) untuk memastikan kemampuan pengulangan.
- Konfigurasikan lokasi pemulihan seperti Wilayah kedua untuk mengimplementasikan dan menguji desain failover. Jika perlu, sumber daya untuk pengujian dapat dikonfigurasi secara sementara untuk membatasi biaya tambahan.

- Tentukan rencana failover mana yang diotomatisasi olehAWS, yang dapat diotomatisasi oleh proses DevOps, dan mana yang mungkin dilakukan secara manual. Dokumentasikan dan ukur RTO dan RPO setiap layanan.
- Buat playbook failover dan sertakan semua langkah untuk melakukan failover setiap sumber daya, aplikasi, dan layanan.
- Buat playbook failback dan sertakan semua langkah untuk melakukan failback (dengan pengaturan waktu) setiap sumber daya, aplikasi, dan layanan
- Buat rencana untuk memulai dan melatih playbook. Gunakan simulasi dan pengujian kekacauan untuk menguji langkah-langkah dan otomatisasi playbook.
- Untuk gangguan lokasi (seperti Zona Ketersediaan atau Wilayah AWS), pastikan Anda memiliki sistem untuk melakukan failover ke sumber daya yang sehat di lokasi yang tidak terkena gangguan. Periksa kuota, tingkat penskalaan otomatis, dan sumber daya yang berjalan sebelum pengujian failover.

Sumber daya

Praktik terbaik Well-Architected terkait:

- REL13 - Merencanakan DR
- REL10 - Menggunakan isolasi kesalahan untuk melindungi beban kerja Anda

Dokumen terkait:

- Menetapkan Target RTO dan RPO
- Menyiapkan ARC dengan penyeimbang beban aplikasi
- Failover menggunakan perutean Tertimbang Route 53
- DR dengan ARC
- EC2 dengan penskalaan otomatis
- Deployment EC2 - Multi-AZ
- Deployment ECS - Multi-AZ
- Mengalihkan lalu lintas menggunakan ARC
- Lambda dengan Application Load Balancer dan Failover
- Replikasi ACM dan Failover

- Replikasi Penyimpanan Parameter dan Failover
- Replikasi lintas wilayah ECR dan Failover
- Konfigurasi replikasi lintas wilayah manajer rahasia
- Mengaktifkan replikasi lintas wilayah untuk EFS dan Failover
- Replikasi Lintas Wilayah EFS dan Failover
- Failover Jaringan
- Failover titik akhir S3 menggunakan MRAP
- Membuat replikasi lintas wilayah untuk S3
- Failover API Gateway Wilayah dengan ARC
- Failover menggunakan akselerator global multiwilayah
- Failover dengan DRS
- Membuat Mekanisme Pemulihan Bencana Menggunakan Amazon Route 53

Contoh terkait:

- Pemulihan Bencana di AWS
- Pemulihan Bencana Elastis di AWS

REL11-BP03 Mengotomatisasi pemulihan di semua lapisan

Setelah kegagalan dideteksi, gunakan kemampuan otomatis untuk melakukan tindakan perbaikan. Degradasi dapat dipulihkan secara otomatis melalui mekanisme servis internal atau memerlukan sumber daya untuk dimulai ulang atau dihapus melalui tindakan remediasi.

Untuk aplikasi yang dikelola secara mandiri dan perbaikan lintas-Wilayah, desain pemulihan dan proses perbaikan otomatis dapat ditarik dari praktik terbaik yang ada.

Kemampuan untuk memulai ulang atau menghapus sumber daya adalah alat yang penting untuk meremediasi kegagalan. Salah satu praktik terbaik adalah membuat layanan stateless jika memungkinkan. Praktik ini mencegah hilangnya data atau ketersediaan pada saat mulai ulang sumber daya. Di cloud, Anda dapat (dan umumnya harus) mengganti seluruh sumber daya (misalnya, instans komputasi atau fungsi nirserver) sebagai bagian dari mulai ulang. Mulai ulang itu sendiri adalah cara yang mudah dan andal untuk pulih dari kegagalan. Ada berbagai jenis kegagalan yang terjadi di dalam beban kerja. Kegagalan dapat terjadi di perangkat keras, perangkat lunak, komunikasi, dan operasi.

Memulai ulang atau mencoba ulang juga berlaku untuk permintaan jaringan. Terapkan pendekatan pemulihan yang sama ke waktu habis jaringan serta kegagalan dependensi yakni ketika dependensi menunjukkan kesalahan. Kedua peristiwa tersebut memiliki efek yang serupa terhadap sistem, sehingga alih-alih berupaya untuk menjadikan masing-masing sebagai kasus spesial, terapkan strategi serupa berupa coba ulang terbatas dengan mundur eksponensial dan jitter. Kemampuan untuk memulai ulang adalah mekanisme pemulihan yang disertakan dalam komputasi berorientasi pemulihan dan arsitektur klaster ketersediaan tinggi.

Hasil yang diinginkan: Tindakan otomatis dilakukan untuk meremediasi deteksi kegagalan.

Antipola umum:

- Menyediakan sumber daya tanpa penskalaan otomatis.
- Melakukan deployment aplikasi di instans atau kontainer secara terpisah.
- Melakukan deployment aplikasi yang tidak dapat dilakukan ke beberapa lokasi tanpa menggunakan pemulihan otomatis.
- Memulihkan secara manual aplikasi yang gagal dipulihkan oleh penskalaan otomatis dan pemulihan otomatis.
- Tidak ada otomatisasi untuk failover basis data.
- Tidak ada metode otomatis untuk mengalihkan rute lalu lintas ke titik akhir baru.
- Tidak ada replikasi penyimpanan.

Manfaat menjalankan praktik terbaik ini: Pemulihan otomatis dapat mengurangi waktu rata-rata pemulihan dan meningkatkan ketersediaan Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Desain untuk Amazon EKS atau layanan Kubernetes lainnya harus mencakup replika minimum dan maksimum atau set stateful dan penyesuaian ukuran klaster dan grup simpul minimum. Mekanisme ini menyediakan jumlah minimum sumber daya pemrosesan yang tersedia secara terus-menerus sambil secara otomatis memulihkan kegagalan apa pun menggunakan bidang kendali Kubernetes.

Pola desain yang diakses melalui penyeimbang beban menggunakan klaster komputasi harus memanfaatkan grup Auto Scaling. Elastic Load Balancing (ELB) secara otomatis mendistribusikan lalu lintas aplikasi yang masuk di beberapa target dan perangkat virtual di satu atau beberapa Zona Ketersediaan (AZ).

Desain berbasis komputasi klaster yang tidak menggunakan penyeimbangan beban harus dirancang ukurannya untuk kehilangan setidaknya satu simpul. Dengan begitu, layanan dapat terus berjalan dalam kapasitas yang kemungkinan lebih rendah saat memulihkan simpul baru. Contoh layanannya adalah Mongo, DynamoDB Accelerator, Amazon Redshift, Amazon EMR, Cassandra, Kafka, MSK-EC2, Couchbase, ELK, dan Amazon OpenSearch Service. Banyak dari layanan ini dapat dirancang dengan fitur pemulihan otomatis tambahan. Beberapa teknologi klaster harus menghasilkan peringatan atas hilangnya simpul yang memicu alur kerja otomatis atau manual untuk membuat ulang simpul baru. Alur kerja ini dapat diotomatisasi menggunakan AWS Systems Manager untuk meremediasi masalah dengan cepat.

Amazon EventBridge dapat digunakan untuk memantau dan memfilter peristiwa seperti alarm CloudWatch atau perubahan status pada layanan AWS lain. Berdasarkan informasi peristiwa, layanan ini kemudian dapat memanggil AWS Lambda, Otomatisasi Systems Manager, atau target lain untuk menjalankan logika remediasi kustom pada beban kerja Anda. Amazon EC2 Auto Scaling dapat dikonfigurasi untuk memeriksa kondisi instans EC2. Jika instans sedang dalam status apa pun selain running (berjalan), atau jika status sistem terganggu, Amazon EC2 Auto Scaling menganggap instans tersebut tidak sehat dan meluncurkan instans pengganti. Untuk penggantian skala besar (seperti hilangnya seluruh Zona Ketersediaan), stabilitas statis lebih disarankan untuk ketersediaan tinggi.

Langkah implementasi

- Gunakan grup Auto Scaling untuk men-deploy tingkatan dalam beban kerja. Auto Scaling dapat melakukan pemulihan mandiri untuk aplikasi stateless serta menambahkan dan menghapus kapasitas.
- Untuk instans komputasi yang disebutkan sebelumnya, gunakan penyeimbangan beban dan pilih jenis penyeimbang beban yang sesuai.
- Pertimbangkan pemulihan untuk Amazon RDS. Dengan instans siaga, konfigurasikan untuk failover otomatis ke instans siaga. Untuk Amazon RDS Read Replica, alur kerja otomatis diperlukan untuk membuat replika baca primer.
- Implementasikan pemulihan otomatis pada instans EC2 yang telah melakukan deployment aplikasi yang tidak dapat di-deploy di beberapa lokasi, dan dapat menoleransi boot ulang setelah kegagalan. Pemulihan otomatis dapat digunakan untuk mengganti perangkat keras yang mengalami kegagalan dan memulai ulang instans ketika aplikasi tidak dapat diterapkan di beberapa lokasi. Metadata instans dan alamat IP terkait disimpan, serta volume EBS dan pasang poin ke Amazon Elastic File System atau File Systems for Lustre dan Windows. Jika menggunakan

AWS OpsWorks, Anda dapat mengonfigurasi pemulihan otomatis instans EC2 pada tingkat lapisan.

- Implementasikan pemulihan otomatis menggunakan AWS Step Functions dan AWS Lambda ketika Anda tidak dapat menggunakan penskalaan otomatis atau pemulihan otomatis, atau ketika pemulihan otomatis gagal. Ketika Anda tidak dapat menggunakan penskalaan otomatis, dan tidak dapat menggunakan pemulihan otomatis atau pemulihan otomatis gagal, Anda dapat mengotomatiskan pemulihan menggunakan AWS Step Functions dan AWS Lambda.
- Amazon EventBridge dapat digunakan untuk memantau dan memfilter peristiwa seperti alarm CloudWatch atau perubahan status di layanan AWS lain. Berdasarkan informasi peristiwa, layanan ini kemudian dapat menginvokasi AWS Lambda (atau target lainnya) untuk menjalankan logika remediasi kustom pada beban kerja Anda.

Sumber daya

Praktik terbaik terkait:

- Definisi Ketersediaan
- REL11-BP01 Memantau semua komponen beban kerja untuk mendeteksi kegagalan

Dokumen terkait:

- Cara Kerja AWS Auto Scaling
- Pemulihan Otomatis Amazon EC2
- Amazon Elastic Block Store (Amazon EBS)
- Amazon Elastic File System (Amazon EFS)
- Apa itu Amazon FSx for Lustre?
- Apa itu Amazon FSx for Windows File Server?
- AWS OpsWorks: Menggunakan Auto Healing untuk Mengganti Instans yang Gagal
- Apa itu AWS Step Functions?
- Apa itu AWS Lambda?
- Apa Itu Amazon EventBridge?
- Menggunakan Alarm Amazon CloudWatch
- Failover Amazon RDS
- SSM - Otomatisasi Systems Manager

- Praktik Terbaik Arsitektur Tangguh

Video terkait:

- Penyediaan dan Penskalaan OpenSearch Service Secara Otomatis
- Failover Amazon RDS Secara Otomatis

Contoh terkait:

- Lokakarya di Auto Scaling
- Lokakarya Failover Amazon RDS

Alat terkait:

- CloudWatch
- CloudWatch X-Ray

REL11-BP04 Mengandalkan bidang data dan bukan bidang kendali selama pemulihan

Bidang kendali menyediakan API administratif yang digunakan untuk membuat, membaca, dan mendeskripsikan, memperbarui, menghapus, dan mencantumkan (CRUDL) sumber daya, sedangkan bidang data menangani lalu lintas layanan sehari-hari. Saat mengimplementasikan respons pemulihan atau mitigasi terhadap peristiwa yang berpotensi berdampak pada ketahanan, fokuslah pada penggunaan operasi bidang kontrol dalam jumlah minim untuk memulihkan, mengubah skala, mengembalikan, memperbaiki, atau melakukan failover layanan. Tindakan bidang data harus menggantikan aktivitas apa pun selama peristiwa degradasi ini.

Misalnya, berikut ini adalah semua tindakan bidang kendali: meluncurkan instans komputasi baru, membuat penyimpanan blok, dan mendeskripsikan layanan antrean. Saat Anda meluncurkan instans komputasi, bidang kendali harus melakukan beberapa tugas seperti menemukan host fisik dengan kapasitas, mengalokasikan antarmuka jaringan, menyiapkan volume penyimpanan blok lokal, menghasilkan kredensial, dan menambahkan aturan keamanan. Orkestrasi bidang kendali cenderung rumit.

Hasil yang diinginkan: Ketika sumber daya memasuki keadaan terganggu, sistem mampu pulih secara otomatis atau manual dengan mengalihkan lalu lintas dari sumber daya yang terganggu ke sumber daya yang sehat.

Antipola umum:

- Ketergantungan pada pengubahan catatan DNS untuk mengalihkan lalu lintas.
- Ketergantungan pada operasi penskalaan bidang kendali untuk menggantikan komponen yang terganggu karena sumber daya yang disediakan tidak memadai.
- Mengandalkan tindakan bidang kendali ekstensif multi-API dan multi layanan untuk meremediasi kategori gangguan apa pun.

Manfaat menjalankan praktik terbaik ini: Peningkatan tingkat keberhasilan untuk remediasi otomatis dapat mengurangi waktu rata-rata Anda untuk pemulihan dan meningkatkan ketersediaan beban kerja.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang: Untuk jenis degradasi layanan tertentu, bidang kendali terkena pengaruh. Ketergantungan pada penggunaan bidang kendali secara ekstensif untuk remediasi dapat meningkatkan waktu pemulihan (RTO) dan rata-rata waktu untuk pulih (MTTR).

Panduan implementasi

Untuk membatasi tindakan bidang data, nilai setiap layanan untuk menemukan tindakan yang diperlukan untuk memulihkan layanan.

Manfaatkan Amazon Application Recovery Controller (ARC) untuk mengalihkan lalu lintas DNS. Fitur-fitur ini terus-menerus memantau kemampuan aplikasi Anda untuk pulih dari kegagalan, dan memampukan Anda untuk mengontrol pemulihan aplikasi di beberapa Wilayah AWS, Zona Ketersediaan, dan on-premise.

Kebijakan perutean Route 53 menggunakan bidang kendali, jadi jangan mengandalkannya untuk pemulihan. Bidang data Route 53 menjawab kueri DNS dan melakukan serta mengevaluasi pemeriksaan kondisi. Bidang data ini terdistribusi secara global dan didesain untuk perjanjian tingkat layanan (SLA) dengan ketersediaan 100%..

Konsol dan API manajemen Route 53 di mana Anda membuat, memperbarui, dan menghapus sumber daya Route 53 dijalankan di bidang kendali yang didesain untuk memprioritaskan durabilitas dan konsistensi tinggi yang Anda perlukan ketika mengelola DNS. Untuk mencapai hal ini, bidang kendali ditempatkan di satu Wilayah: US East (N. Virginia). Walaupun kedua sistem dibangun agar sangat andal, bidang kendali tidak disertakan dalam SLA. Mungkin ada peristiwa langka di mana desain tangguh bidang data memungkinkannya untuk mempertahankan ketersediaan sedangkan

bidang kendali tidak. Untuk mekanisme failover dan pemulihan bencana, gunakan fungsi bidang data untuk memberikan keandalan yang sebaik mungkin.

Untuk Amazon EC2, gunakan desain stabilitas statis untuk membatasi tindakan bidang kendali. Tindakan bidang kendali mencakup peningkatan skala sumber daya secara individu atau menggunakan grup Auto Scaling (ASG). Untuk tingkat ketahanan tertinggi, berikan kapasitas yang cukup di klaster yang digunakan untuk failover. Jika ambang kapasitas ini harus dibatasi, tetapkan throttle pada keseluruhan sistem menyeluruh untuk membatasi lalu lintas total yang mencapai set sumber daya terbatas.

Untuk layanan seperti Amazon DynamoDB, Amazon API Gateway, penyeimbang beban, dan nirserver AWS Lambda, penggunaan layanan-layanan tersebut memanfaatkan bidang data. Namun, pembuatan fungsi baru, penyeimbang beban, gateway API, atau tabel DynamoDB adalah tindakan bidang kendali dan harus diselesaikan sebelum degradasi sebagai persiapan peristiwa dan latihan tindakan failover. Untuk Amazon RDS, tindakan bidang data memungkinkan akses ke data.

Untuk informasi selengkapnya tentang bidang data, bidang kendali, dan bagaimana AWS membangun layanan untuk memenuhi target ketersediaan tinggi, lihat Stabilitas statis menggunakan Zona Ketersediaan.

Pahami operasi mana yang ada di bidang data dan mana yang ada di bidang kendali.

Langkah implementasi

Untuk setiap beban kerja yang perlu dipulihkan setelah peristiwa degradasi, evaluasi runbook failover, desain ketersediaan tinggi, desain perbaikan otomatis, atau rencana pemulihan sumber daya HA. Identifikasikan setiap tindakan yang mungkin dianggap sebagai tindakan bidang kendali.

Pertimbangkan mengubah tindakan kendali ke tindakan bidang data:

- Auto Scaling (bidang kendali) dibandingkan dengan sumber daya Amazon EC2 yang telah diskalakan sebelumnya (bidang data)
- Migrasikan ke Lambda dan metode penskalaannya (bidang data) atau Amazon EC2 dan ASG (bidang kendali)
- Nilai desain apa pun menggunakan Kubernetes dan sifat tindakan bidang kendali. Menambahkan pod adalah tindakan bidang data di Kubernetes. Tindakan harus dibatasi ke penambahan pod dan bukan ke penambahan simpul. Jika menggunakan simpul yang disediakan secara berlebihan adalah metode yang lebih disukai untuk membatasi tindakan bidang kendali

Pertimbangkan pendekatan alternatif yang memungkinkan tindakan bidang data untuk memengaruhi remediasi yang sama.

- Route 53 Rekam perubahan (bidang kendali) atau ARC (bidang data)
- Route 53 Pemeriksaan kondisi untuk pembaruan yang lebih otomatis

Pertimbangkan beberapa layanan di Wilayah sekunder, jika layanan sangat penting, untuk memungkinkan lebih banyak tindakan bidang kendali dan bidang data di Wilayah yang tidak terdampak.

- Amazon EC2 Auto Scaling atau Amazon EKS di Wilayah primer dibandingkan dengan Amazon EC2 Auto Scaling atau Amazon EKS di Wilayah sekunder dan merutekan lalu lintas ke Wilayah sekunder (tindakan bidang kendali)
- Membuat replika baca di primer sekunder atau mencoba tindakan yang sama di Wilayah primer (tindakan bidang kendali)

Sumber daya

Praktik terbaik terkait:

- Definisi Ketersediaan
- REL11-BP01 Memantau semua komponen beban kerja untuk mendeteksi kegagalan

Dokumen terkait:

- Partner APN: partner yang dapat membantu dengan otomatisasi toleransi kesalahan Anda
- AWS Marketplace: produk yang dapat digunakan untuk toleransi kesalahan
- Amazon Builders' Library: Menghindari kelebihan beban dalam sistem terdistribusi dengan mengontrol layanan lebih kecil
- API Amazon DynamoDB (bidang kendali dan bidang data)
- Pelaksanaan AWS Lambda (dibagi menjadi bidang kendali dan bidang data)
- Bidang Data AWS Elemental MediaStore
- Membangun aplikasi yang sangat tangguh menggunakan Pengontrol Pemulihan Aplikasi Amazon Route 53, Bagian 1: Tumpukan Wilayah Tunggal

- Membangun aplikasi yang sangat tangguh menggunakan Pengontrol Pemulihan Aplikasi Amazon Route 53, Bagian 2: Tumpukan Multi-Wilayah
- Membuat Mekanisme Pemulihan Bencana Menggunakan Amazon Route 53
- Apa itu Pengontrol Pemulihan Aplikasi Route 53?
- Bidang Kendali dan bidang data Kubernetes

Video terkait:

- Kembali ke Dasar - Menggunakan Stabilitas Statis
- Membangun beban kerja multi-lokasi yang tangguh menggunakan layanan global AWS

Contoh terkait:

- Memperkenalkan Pengontrol Pemulihan Aplikasi Amazon Route 53
- Amazon Builders' Library: Menghindari kelebihan beban dalam sistem terdistribusi dengan mengontrol layanan lebih kecil
- Membangun aplikasi yang sangat tangguh menggunakan Pengontrol Pemulihan Aplikasi Amazon Route 53, Bagian 1: Tumpukan Wilayah Tunggal
- Membangun aplikasi yang sangat tangguh menggunakan Pengontrol Pemulihan Aplikasi Amazon Route 53, Bagian 2: Tumpukan Multi-Wilayah
- Stabilitas statis menggunakan Zona Ketersediaan

Alat terkait:

- Amazon CloudWatch
- AWS X-Ray

REL11-BP05 Menggunakan stabilitas statis untuk mencegah perilaku bimodal

Beban kerja harus stabil secara statis dan hanya beroperasi dalam mode normal tunggal. Perilaku bimodal adalah ketika beban kerja Anda menunjukkan perilaku yang berbeda dalam mode normal dan mode kegagalan.

Misalnya, Anda mungkin mencoba pulih dari kegagalan Zona Ketersediaan dengan meluncurkan instans baru di Zona Ketersediaan yang berbeda. Hal ini dapat menyebabkan respons bimodal

selama mode kegagalan. Sebagai gantinya, Anda harus membangun beban kerja yang stabil secara statis dan beroperasi dalam satu mode saja. Dalam contoh ini, instans-instans tersebut seharusnya telah disediakan di Zona Ketersediaan kedua sebelum terjadinya kegagalan. Desain stabilitas statis ini memastikan beban kerja hanya beroperasi dalam satu mode.

Hasil yang diinginkan: Beban kerja tidak menunjukkan perilaku bimodal selama mode normal dan mode kegagalan.

Antipola umum:

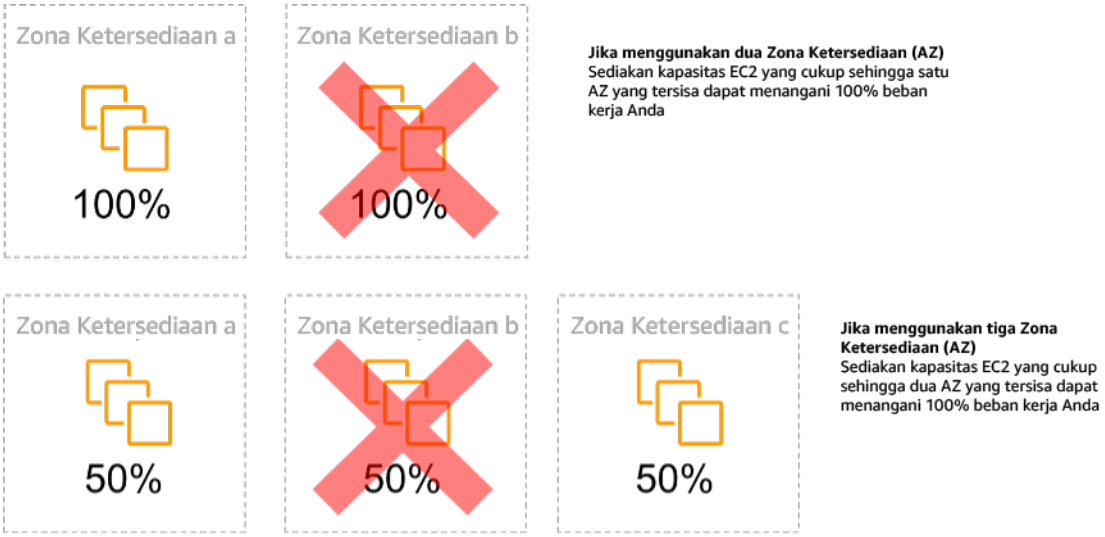
- Berasumsi bahwa sumber daya selalu dapat disediakan terlepas dari ruang lingkup kegagalannya.
- Mencoba memperoleh sumber daya secara dinamis selama kegagalan.
- Tidak menyediakan sumber daya yang memadai di seluruh zona atau Wilayah sampai terjadi kegagalan.
- Mempertimbangkan desain stabil statis hanya untuk sumber daya komputasi.

Manfaat menjalankan praktik terbaik ini: Beban kerja yang berjalan dengan desain yang stabil secara statis mampu memiliki hasil yang dapat diprediksi selama peristiwa normal dan kegagalan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Perilaku bimodal terjadi ketika beban kerja Anda menunjukkan perilaku yang berbeda dalam mode normal dan mode gagal, (misalnya, mengandalkan peluncuran instans baru jika Zona Ketersediaan gagal). Contoh perilaku bimodal adalah ketika Amazon EC2 yang stabil menyediakan instans yang cukup di setiap Zona Ketersediaan untuk menangani beban dari beban kerja jika satu AZ disingkirkan. Elastic Load Balancing atau Amazon Route 53 akan melakukan pemeriksaan kondisi untuk memindahkan beban dari instans yang terganggu. Setelah lalu lintas dipindahkan, gunakan AWS Auto Scaling untuk mengganti instans secara asinkron dari zona yang gagal dan luncurkan di zona sehat. Stabilitas statis untuk deployment komputasi (seperti kontainer atau instans EC2) akan menghasilkan keandalan tertinggi.



Stabilitas statis instans EC2 di seluruh Zona Ketersediaan

Ini harus ditimbang berdasarkan biaya untuk model ini serta nilai bisnis untuk memelihara beban kerja di bawah semua kasus ketahanan. Menyediakan kapasitas komputasi yang lebih sedikit dan mengandalkan peluncuran instans baru apabila terjadi kegagalan memang lebih murah, tetapi untuk kegagalan berskala besar (seperti gangguan pada Zona Ketersediaan atau Regional), pendekatan ini kurang efektif karena bergantung pada bidang operasional serta sumber daya yang tersedia di zona atau Wilayah yang tidak terdampak.

Solusi Anda harus mengukur keandalan berdasarkan kebutuhan biaya untuk beban kerja Anda. Arsitektur stabilitas statis berlaku untuk berbagai arsitektur termasuk instans komputasi yang tersebar di Zona Ketersediaan, desain replika baca basis data, desain klaster Kubernetes (Amazon EKS), dan arsitektur failover multi-Wilayah.

Anda juga dapat menerapkan desain yang lebih stabil secara statis dengan menggunakan lebih banyak sumber daya di setiap zona. Dengan menggunakan lebih banyak zona, Anda mengurangi jumlah komputasi tambahan yang Anda perlukan untuk stabilitas statis.

Contoh perilaku bimodal adalah batas waktu jaringan yang dapat menyebabkan sistem mencoba melakukan refresh status konfigurasi seluruh sistem. Ini akan menambahkan beban yang tak terduga ke komponen lain, dan dapat menyebabkan komponen tersebut gagal, sehingga berimbas pada konsekuensi lain yang tak terduga. Putaran umpan balik negatif ini memengaruhi ketersediaan beban kerja Anda. Sebagai gantinya, Anda dapat membangun sistem yang stabil secara statis dan beroperasi dalam satu mode saja. Desain yang stabil secara statis akan melakukan tugas yang konstan, dan selalu menyegarkan status konfigurasi dengan irama yang teratur. Ketika panggilan gagal, beban kerja akan menggunakan nilai yang sebelumnya di-cache, dan memulai alarm.

Contoh perilaku bimodal lainnya adalah memperbolehkan klien untuk melewati cache beban kerja Anda ketika kegagalan terjadi. Ini mungkin terlihat seperti solusi yang mengakomodasi kebutuhan klien, tetapi hal ini secara signifikan dapat mengubah permintaan di beban kerja Anda dan kemungkinan akan mengakibatkan kegagalan.

Lakukan penilaian beban kerja kritis untuk menentukan beban kerja apa yang memerlukan jenis desain ketahanan ini. Untuk beban kerja yang dianggap kritis, setiap komponen aplikasi harus ditinjau. Contoh jenis layanan yang memerlukan evaluasi stabilitas statis adalah:

- Komputasi: Amazon EC2, EKS-EC2, ECS-EC2, EMR-EC2
- Basis Data: Amazon Redshift, Amazon RDS, Amazon Aurora
- Penyimpanan: Amazon S3 (Zona Tunggal), Amazon EFS (mount), Amazon FSx (mount)
- Penyeimbang beban: Di bawah desain tertentu

Langkah implementasi

- Bangun sistem yang stabil secara statis dan hanya beroperasi dalam satu mode. Dalam hal ini, sediakan cukup instans di setiap Zona Ketersediaan atau Wilayah untuk menangani kapasitas beban kerja jika satu Zona Ketersediaan atau Wilayah dihapus. Berbagai layanan dapat digunakan untuk perutean ke sumber daya yang sehat, seperti:
  - Perutean DNS Lintas Wilayah
  - Perutean Multiwilayah Amazon S3 MRAP
  - AWS Global Accelerator
  - Amazon Application Recovery Controller (ARC)
- Konfigurasikan replika baca basis data untuk memperhitungkan hilangnya satu instans utama atau replika baca. Jika lalu lintas dilayani oleh replika baca, kuantitas di setiap Zona Ketersediaan dan setiap Wilayah harus sama dengan kebutuhan keseluruhan jika terjadi kegagalan zona atau Wilayah.
- Konfigurasikan data kritis di dalam penyimpanan Amazon S3 yang dirancang agar stabil secara statis untuk data yang disimpan jika terjadi kegagalan Zona Ketersediaan. Jika kelas penyimpanan Amazon S3 One Zone-IA digunakan, ini tidak boleh dianggap stabil secara statis, karena hilangnya zona tersebut akan meminimalkan akses ke data yang disimpan.
- Penyeimbang beban terkadang dikonfigurasi secara salah atau secara bawaan untuk melayani satu Zona Ketersediaan tertentu. Dalam hal ini, desain yang stabil secara statis mungkin adalah

menyebarkan beban kerja di beberapa AZ dalam desain yang lebih kompleks. Desain asli dapat digunakan untuk mengurangi lalu lintas antarzona untuk alasan keamanan, latensi, atau biaya.

Sumber daya

Praktik terbaik Well-Architected terkait:

- Definisi Ketersediaan
- REL11-BP01 Memantau semua komponen beban kerja untuk mendeteksi kegagalan
- REL11-BP04 Mengandalkan bidang data dan bukan bidang kendali selama pemulihan

Dokumen terkait:

- Meminimalkan Ketergantungan dalam Rencana Pemulihan Bencana
- Amazon Builders' Library: Stabilitas statis menggunakan Zona Ketersediaan
- Batas Isolasi Kesalahan
- Stabilitas statis menggunakan Zona Ketersediaan
- RDS Multi-Zona
- Meminimalkan Ketergantungan dalam Rencana Pemulihan Bencana
- Perutean DNS Lintas Wilayah
- Perutean Multiwilayah Amazon S3 MRAP
- AWS Global Accelerator
- ARC
- Amazon S3 Zona Tunggal
- Penyeimbangan Beban Lintas Zona

Video terkait:

- Stabilitas statis di AWS: AWS re:Invent 2019: Memperkenalkan Amazon Builders’ Library (DOP328)

Contoh terkait:

- Amazon Builders' Library: Stabilitas statis menggunakan Zona Ketersediaan

REL11-BP06 Mengirimkan notifikasi ketika peristiwa memengaruhi ketersediaan

Notifikasi dikirimkan setelah pelanggaran ambang batas terdeteksi, bahkan apabila peristiwa yang menyebabkan masalah tersebut sudah diatasi secara otomatis.

Pemulihan otomatis menjadikan beban kerja Anda andal. Namun, kemampuan ini juga menyembunyikan masalah dasar yang perlu diatasi. Implementasikan pemantauan peristiwa yang baik agar Anda dapat mendeteksi pola masalah, termasuk masalah yang ditangani oleh pemulihan otomatis, sehingga Anda dapat mengatasi akar masalahnya.

Sistem yang tangguh dirancang sedemikian rupa sehingga peristiwa degradasi langsung dikomunikasikan kepada tim yang tepat. Notifikasi ini harus dikirim melalui satu atau banyak saluran komunikasi.

Hasil yang diinginkan: Pemberitahuan langsung dikirim ke tim operasi ketika ambang batas dilanggar, seperti tingkat kesalahan, latensi, atau metrik indikator kinerja utama (KPI) penting lainnya, sehingga masalah ini diselesaikan sesegera mungkin dan dampak terhadap pengguna dapat dicegah atau diminimalkan.

Antipola umum:

- Mengirimkan terlalu banyak alarm.
- Mengirimkan alarm yang tidak dapat ditindaklanjuti.
- Mengatur ambang alarm terlalu tinggi (terlalu sensitif) atau terlalu rendah (kurang sensitif).
- Tidak mengirimkan alarm untuk dependensi eksternal.
- Tidak mempertimbangkan kegagalan abu-abu saat merancang pemantauan dan alarm.
- Melakukan otomatisasi pemulihan, tetapi tidak memberikan notifikasi kepada tim yang tepat bahwa pemulihan diperlukan.

Manfaat menjalankan praktik terbaik ini: Notifikasi pemulihan membuat tim operasional dan bisnis menyadari adanya degradasi layanan sehingga mereka dapat segera bereaksi untuk meminimalkan waktu deteksi rata-rata (MTTD) dan waktu perbaikan rata-rata (MTTR). Notifikasi peristiwa pemulihan juga menjamin bahwa Anda tidak mengabaikan masalah yang jarang terjadi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang. Kegagalan mengimplementasikan mekanisme pemantauan dan notifikasi peristiwa secara tepat dapat mengakibatkan kegagalan dalam mendeteksi pola masalah, termasuk masalah yang ditangani oleh

pemulihan otomatis. Sebuah tim hanya akan menyadari adanya degradasi sistem ketika pengguna menghubungi layanan pelanggan atau secara kebetulan.

Panduan implementasi

Saat menetapkan strategi pemantauan, alarm yang dipicu adalah peristiwa umum. Peristiwa ini kemungkinan berisi pengidentifikasi untuk alarm, status alarm (seperti `IN ALARM` atau `OK`), dan detail mengenai pemicunya. Dalam banyak kasus, peristiwa alarm seharusnya dideteksi dan email notifikasi dikirimkan. Ini adalah contoh tindakan pada alarm. Notifikasi alarm sangat penting dalam hal observabilitas karena memberi tahu orang yang tepat bahwa terdapat sebuah masalah. Namun, ketika tindakan terhadap peristiwa sudah matang di dalam solusi observabilitas Anda, tindakan tersebut dapat secara otomatis memperbaiki masalah tanpa memerlukan campur tangan manusia.

Setelah alarm pemantauan KPI ditetapkan, peringatan seharusnya dikirimkan ke tim yang tepat ketika ambang batas terlampaui. Peringatan tersebut juga dapat digunakan untuk memicu proses otomatis yang akan mencoba memperbaiki degradasi.

Untuk pemantauan ambang batas yang lebih kompleks, alarm komposit harus dipertimbangkan. Alarm komposit menggunakan sejumlah alarm pemantauan KPI untuk membuat peringatan berdasarkan logika bisnis operasional. CloudWatch Alarm dapat dikonfigurasi untuk mengirimkan email, atau untuk mencatatkan insiden di dalam sistem pelacakan insiden pihak ketiga menggunakan integrasi Amazon SNS atau Amazon EventBridge.

Langkah implementasi

Buat berbagai jenis alarm berdasarkan bagaimana beban kerja dipantau, seperti:

- Alarm aplikasi digunakan untuk mendeteksi apabila ada bagian dari beban kerja Anda yang tidak berfungsi dengan baik.
- Alarm infrastruktur menunjukkan kapan sumber daya perlu diskalakan. Alarm dapat ditampilkan secara visual di dasbor, mengirimkan peringatan melalui Amazon SNS atau email, dan bekerja sama dengan Auto Scaling untuk mengecilkan atau memperluas skala sumber daya beban kerja.
- Alarm statis sederhana dapat dibuat untuk memantau apabila metrik melanggar ambang batas statis selama periode evaluasi tertentu.
- Alarm komposit dapat memperhitungkan alarm-alarm kompleks dari berbagai sumber.
- Setelah alarm dibuat, buat peristiwa notifikasi yang sesuai. Anda dapat langsung menginvokasi API Amazon SNS untuk mengirimkan notifikasi dan menautkan otomatisasi apa pun untuk perbaikan atau komunikasi.

- Integrasikan Amazon Health Aware untuk memungkinkan visibilitas pemantauan ke sumber daya AWS yang mungkin mengalami degradasi. Untuk beban kerja yang penting untuk bisnis, solusi ini menyediakan akses ke peringatan proaktif dan waktu nyata untuk layanan AWS.

Sumber daya

Praktik terbaik Well-Architected terkait:

- Definisi Ketersediaan

Dokumen terkait:

- Membuat Alarm CloudWatch Berdasarkan Ambang Batas Statis
- Apa Itu Amazon EventBridge?
- Apa Itu Amazon Simple Notification Service?
- Memublikasikan Metrik Kustom
- Menggunakan Alarm Amazon CloudWatch
- Amazon Health Aware (AHA)
- Menyiapkan Alarm komposit CloudWatch
- Apa yang baru di Observabilitas AWS pada re:Invent 2022

Alat terkait:

- CloudWatch
- CloudWatch X-Ray

REL11-BP07 Merancang produk Anda agar memenuhi target ketersediaan dan perjanjian tingkat layanan (SLA) waktu aktif

Rancang produk Anda agar memenuhi target ketersediaan dan perjanjian tingkat layanan (SLA) waktu aktif. Jika Anda memublikasi atau secara pribadi menyetujui target ketersediaan atau SLA yang berlaku, verifikasikan bahwa proses operasional dan arsitektur Anda didesain untuk mendukungnya.

Hasil yang diinginkan: Setiap aplikasi memiliki target yang ditetapkan untuk ketersediaan dan SLA untuk metrik performa, yang dapat dipantau dan dipertahankan untuk memenuhi hasil bisnis.

Antipola umum:

- Mendesain dan melakukan deployment beban kerja tanpa menetapkan SLA apa pun.
- Metrik SLA ditetapkan ke tingkat tinggi tanpa rasional atau persyaratan bisnis.
- Menetapkan SLA tanpa memperhitungkan dependensi dan SLA yang mendasarinya.
- Desain aplikasi dibuat tanpa mempertimbangkan Model Tanggung Jawab Bersama untuk Ketangguhan.

Manfaat menjalankan praktik terbaik ini: Mendesain aplikasi berdasarkan target ketangguhan utama membantu Anda memenuhi tujuan bisnis dan ekspektasi pelanggan. Tujuan ini membantu mendorong proses desain aplikasi yang mengevaluasi berbagai macam teknologi dan mempertimbangkan beragam kompromi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Desain aplikasi harus memperhitungkan berbagai rangkaian persyaratan yang didapatkan dari tujuan bisnis, operasional, dan finansial. Dalam persyaratan operasional, beban kerja harus memiliki target metrik ketangguhan tertentu sehingga dapat dipantau dan dukung dengan sesuai. Metrik ketangguhan tidak boleh ditetapkan atau didapatkan setelah melakukan deployment beban kerja. Metrik ketangguhan harus ditetapkan selama fase desain dan membantu memandu berbagai keputusan dan kompromi.

- Setiap beban kerja harus memiliki rangkaian metrik ketangguhannya sendiri. Metrik-metrik tersebut mungkin berbeda dari aplikasi bisnis yang lain.
- Mengurangi dependensi dapat memiliki dampak positif pada ketersediaan. Setiap beban kerja harus mempertimbangkan dependensinya serta SLA-nya. Secara umum, pilih dependensi dengan target ketersediaan yang setara dengan atau lebih besar dari target beban kerja Anda.
- Pertimbangkan desain yang dipasangkan secara longgar sehingga beban kerja Anda dapat beroperasi dengan benar meskipun ada gangguan dependensi, apabila mungkin.
- Kurangi dependensi bidang kendali, terutama selama pemulihan atau degradasi. Evaluasi desain yang secara statis stabil untuk beban kerja yang penting bagi misi. Gunakan penghematan sumber daya untuk meningkatkan ketersediaan dependensi tersebut di beban kerja.
- Observabilitas dan instrumentasi sangat penting untuk mencapai SLA dengan mengurangi Waktu Rata-Rata ke Deteksi (MTTD) dan Waktu Rata-Rata ke Perbaikan (MTTR).

- Kegagalan lebih jarang (MTBF lebih lama), waktu deteksi kegagalan lebih pendek (MTTD lebih singkat), dan waktu perbaikan lebih singkat (MTTR lebih singkat) adalah tiga faktor yang digunakan untuk meningkatkan ketersediaan di sistem terdistribusi.
- Menetapkan dan memenuhi metrik ketangguhan untuk beban kerja merupakan fondasi dari desain yang efektif. Desain tersebut harus memperhitungkan kompromi terkait kompleksitas desain, dependensi layanan, performa, penskalaan, dan biaya.

Langkah implementasi

- Tinjau dan dokumentasikan desain beban kerja sambil mempertimbangkan pertanyaan berikut:
  - Di mana bidang kendali digunakan di beban kerja?
  - Bagaimana beban kerja mengimplementasikan toleransi kesalahan?
  - Apa saja pola desain untuk penskalaan, penskalaan otomatis, redundansi, dan komponen dengan ketersediaan tinggi?
  - Apa saja persyaratan untuk ketersediaan dan konsistensi data?
  - Apakah ada pertimbangan untuk penghematan sumber daya atau stabilitas statis sumber daya?
  - Apa saja dependensi layanan?
- Tetapkan metrik SLA berdasarkan arsitektur beban kerja sambil bekerja sama dengan para pemangku kepentingan. Pertimbangkan SLA semua dependensi yang digunakan oleh beban kerja.
- Setelah target SLA ditetapkan, optimalkan arsitektur untuk memenuhi SLA.
- Setelah desain yang akan memenuhi SLA dibuat, implementasikan perubahan operasional, otomatisasi proses, dan runbook yang juga akan memiliki fokus pada pengurangan MTTD dan MTTR.
- Setelah di-deploy, pantau dan laporkan SLA.

Sumber daya

Praktik Terbaik Terkait:

- REL03-BP01 Memilih cara untuk menyegmentasi beban kerja
- REL10-BP01 Melakukan deployment beban kerja ke beberapa lokasi
- REL11-BP01 Memantau semua komponen beban kerja untuk mendeteksi kegagalan
- REL11-BP03 Mengotomatisasi pemulihan di semua lapisan
- REL12-BP05 Menguji ketahanan menggunakan chaos engineering

- REL13-BP01 Tetapkan sasaran pemulihan untuk waktu henti dan kehilangan data
- Memahami kesehatan beban kerja

Dokumen terkait:

- Ketersediaan dengan redundansi
- Pilar keandalan - Ketersediaan
- Mengukur ketersediaan
- Batas Isolasi Kesalahan AWS
- Model Tanggung Jawab Bersama untuk Ketangguhan
- Stabilitas statis menggunakan Zona Ketersediaan
- Perjanjian Tingkat Layanan (SLA) AWS
- Panduan untuk Arsitektur Berbasis Sel di AWS
- Infrastruktur AWS
- Laporan resmi Pola Ketangguhan Multi-AZ Lanjutan

Layanan terkait:

- Amazon CloudWatch
- AWS Config
- AWS Trusted Advisor

REL 12. Bagaimana cara menguji keandalan?

Setelah Anda mendesain beban kerja Anda agar tangguh terhadap tekanan produksi, pengujian adalah satu-satunya cara untuk memverifikasi bahwa beban kerja akan beroperasi sesuai desain, dan memberikan ketangguhan yang Anda harapkan.

Praktik terbaik

- REL12-BP01 Menggunakan buku pedoman untuk menyelidiki kegagalan
- REL12-BP02 Menjalankan analisis setelah insiden
- REL12-BP03 Menguji persyaratan fungsional

- REL12-BP04 Menguji persyaratan penskalaan dan kinerja
- REL12-BP05 Menguji ketahanan menggunakan chaos engineering
- REL12-BP06 Mengadakan game day secara rutin

REL12-BP01 Menggunakan buku pedoman untuk menyelidiki kegagalan

Dokumentasikan proses penyelidikan di buku pedoman agar dapat memberikan respons yang cepat dan konsisten terhadap skenario kegagalan yang tidak benar-benar dipahami. Buku pedoman adalah langkah-langkah yang telah ditetapkan di awal untuk mengidentifikasi faktor yang menyebabkan skenario kegagalan. Hasil dari langkah proses apa pun digunakan untuk menentukan langkah berikutnya yang akan dilakukan sampai masalah diidentifikasi atau dieskalasi.

Buku pedoman adalah perencanaan proaktif yang harus Anda lakukan, agar Anda dapat mengambil tindakan reaktif secara efektif. Ketika skenario kegagalan yang tidak tercakup dalam buku pedoman dialami di lingkungan produksi, tangani masalah terlebih dahulu (padamkan api). Lalu lihat kembali langkah-langkah yang telah Anda ambil untuk mengatasi masalah tersebut dan gunakan untuk menambahkan entri baru dalam buku pedoman.

Ingat bahwa buku pedoman digunakan untuk merespons insiden tertentu, sedangkan runbook digunakan untuk mencapai hasil tertentu. Sering kali, runbook digunakan untuk untuk aktivitas rutin, dan buku pedoman digunakan untuk merespons peristiwa nonrutin.

Antipola umum:

- Berencana untuk melakukan deployment beban kerja tanpa mengetahui proses untuk mendiagnosis masalah atau merespons insiden.
- Keputusan yang tidak direncanakan tentang sistem mana saja yang dikumpulkan log dan metriknya saat menyelidiki peristiwa.
- Tidak mempertahankan metrik dan peristiwa cukup lama agar dapat mengambil data.

Manfaat menjalankan praktik terbaik ini: Pencatatan runbook memastikan prosedur dapat diikuti secara konsisten. Kodifikasi runbook membatasi munculnya kesalahan dari aktivitas manual. Buku pedoman otomatis dapat menghemat waktu respons peristiwa dengan menghilangkan keharusan campur tangan anggota tim atau memberikan informasi tambahan ketika campur tangan mereka dimulai.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

- Gunakan buku pedoman untuk mengidentifikasi masalah. Buku pedoman adalah proses yang didokumentasikan untuk menyelidiki masalah. Dokumentasikan proses penyelidikan di buku pedoman agar dapat memberikan respons yang cepat dan konsisten terhadap skenario kegagalan. Buku pedoman harus memuat informasi dan panduan yang dapat digunakan oleh orang yang cukup terampil untuk mengumpulkan informasi, mengidentifikasi potensi sumber kegagalan, mengisolasi kesalahan, dan menentukan faktor penyebabnya (lakukan analisis pascainsiden).
  - Implementasikan buku pedoman sebagai kode. Jalankan operasi sebagai kode dengan membuat skrip buku pedoman Anda untuk memastikan konsistensi dan mengurangi kesalahan yang disebabkan proses manual. Buku pedoman dapat terdiri dari beberapa skrip sesuai dengan banyaknya langkah yang diperlukan untuk mengidentifikasi faktor penyebab masalah. Aktivitas runbook dapat dipicu atau dijalankan sebagai bagian dari aktivitas buku pedoman, atau mempercepat eksekusi buku pedoman untuk merespons peristiwa yang teridentifikasi.
    - Otomatiskan buku pedoman operasional Anda dengan AWS Systems Manager
    - AWS Systems Manager Run Command
    - AWS Systems Manager Automation
    - Apa itu AWS Lambda?
    - Apa Itu Amazon EventBridge?
    - Menggunakan Alarm Amazon CloudWatch

Sumber daya

Dokumen terkait:

- AWS Systems Manager Automation
- AWS Systems Manager Run Command
- Otomatiskan buku pedoman operasional Anda dengan AWS Systems Manager
- Menggunakan Alarm Amazon CloudWatch
- Menggunakan Canary (Amazon CloudWatch Synthetics)
- Apa Itu Amazon EventBridge?
- Apa itu AWS Lambda?

Contoh terkait:

- Mengotomatiskan operasi dengan Buku Pedoman dan Runbook

REL12-BP02 Menjalankan analisis setelah insiden

Tinjau peristiwa yang memengaruhi pelanggan, dan identifikasi faktor yang berkontribusi serta tindakan pencegahannya. Gunakan informasi ini untuk mengembangkan mitigasi guna meminimalkan atau mencegah kemungkinan terjadi lagi. Kembangkan prosedur untuk respons efektif dan cepat. Komunikasikan faktor yang berkontribusi dan tindakan koreksi yang diperlukan, yang disesuaikan dengan audiens target. Miliki metode untuk mengomunikasikan penyebab ini ke lainnya seperti yang diperlukan.

Menilai alasan mengapa pengujian yang ada tidak dapat menemukan masalahnya. Menambahkan pengujian untuk kasus ini jika pengujian belum ada.

Hasil yang diinginkan: Tim Anda memiliki pendekatan yang konsisten dan disepakati untuk menangani analisis pascainsiden. Salah satu mekanismenya adalah proses koreksi kesalahan (COE). Proses COE membantu tim Anda mengidentifikasi, memahami, dan mengatasi akar penyebab insiden, sekaligus membangun mekanisme dan pagar pembatas untuk membatasi kemungkinan insiden yang sama terjadi lagi.

Antipola umum:

- Menemukan faktor-faktor yang berkontribusi, tetapi tidak terus-menerus mencari lebih dalam untuk masalah potensial dan pendekatan lainnya untuk memitigasi.
- Hanya mengidentifikasi penyebab kesalahan manusia, dan tidak memberikan pelatihan atau otomatisasi apa pun yang dapat mencegah kesalahan manusia.
- Fokus menyalahkan, bukan memahami akar penyebabnya, sehingga tercipta budaya ketakutan dan menghambat komunikasi terbuka
- Tidak berbagi wawasan, yang membuat temuan analisis insiden hanya diketahui kelompok kecil saja, sehingga orang lain tidak dapat belajar dari pengalaman tersebut
- Tidak ada mekanisme untuk mencatat pengetahuan institusional, sehingga wawasan yang berharga hilang karena pelajaran yang didapat tidak diabadikan dalam bentuk praktik terbaik yang diperbarui dan mengakibatkan insiden berulang dengan akar penyebab yang sama atau serupa

Manfaat menyusun praktik terbaik ini: Dengan melakukan analisis setelah insiden dan membagikan hasilnya, beban kerja lain akan dapat memitigasi risiko jika beban kerja sudah mengimplementasikan

faktor penyumbang yang sama, sehingga mitigasi atau pemulihan otomatis dapat diimplementasikan sebelum insiden terjadi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Analisis setelah insiden yang baik memberikan peluang untuk mengusulkan solusi umum terhadap masalah dengan pola arsitektur yang digunakan di tempat lainnya dalam sistem.

Dokumentasi dan penanganan masalah merupakan landasan proses COE. Sebaiknya tentukan cara standar untuk mendokumentasikan akar penyebab kritis, dan memastikan penyebab tersebut ditinjau dan ditangani. Tetapkan kepemilikan yang jelas untuk proses analisis setelah insiden. Tunjuk individu atau tim penanggung jawab yang akan mengawasi penyelidikan dan tindak lanjut insiden.

Dorong budaya yang berfokus pada pembelajaran dan peningkatan, bukan menyalahkan. Tekankan bahwa tujuannya adalah untuk mencegah insiden di kemudian hari, bukan untuk menghukum individu.

Kembangkan prosedur yang jelas untuk melakukan analisis setelah insiden. Prosedur ini harus menguraikan langkah-langkah yang harus diambil, informasi yang akan dikumpulkan, dan pertanyaan-pertanyaan penting yang harus dicari jawabannya selama analisis. Selidiki insiden secara menyeluruh, tidak hanya pada penyebab langsung guna mengidentifikasi akar penyebab dan faktor penyumbangnya. Gunakan teknik seperti Analisis Lima Mengapa untuk menggali lebih dalam masalah yang mendasarinya.

Simpan repositori pelajaran yang didapat dari analisis insiden. Pengetahuan institusional ini dapat digunakan sebagai referensi untuk insiden dan upaya pencegahan ke depannya. Bagikan temuan dan wawasan dari analisis setelah insiden, dan pertimbangkan untuk mengadakan pertemuan tinjauan setelah insiden terbuka untuk membahas pelajaran yang didapatkan.

Langkah implementasi

- Saat melakukan analisis setelah insiden, pastikan tidak menyalahkan siapa pun dalam proses tersebut. Dengan begitu, orang-orang yang terlibat dalam insiden tersebut bersikap rasional terhadap tindakan korektif yang diusulkan dan mendorong penilaian mandiri yang jujur serta kolaborasi di seluruh tim.
- Tentukan cara standar untuk mendokumentasikan masalah kritis. Contoh struktur untuk dokumen tersebut:
  - Apa yang terjadi?

    - Apa dampaknya terhadap pelanggan dan bisnis Anda?
    - Apa akar penyebabnya?
    - Data apa yang Anda miliki untuk mendukung hal ini?
        - Misalnya, metrik dan grafik
    - Apa implikasi pilar kritis, terutama keamanan?
        - Saat merancang beban kerja, Anda memilah pilar-pilar sesuai dengan konteks bisnis Anda. Keputusan bisnis ini dapat menentukan prioritas rekayasa Anda. Anda dapat mengoptimalkan untuk mengurangi biaya dengan mengorbankan keandalan dalam lingkungan pengembangan, atau, untuk solusi yang sangat penting, Anda dapat mengoptimalkan keandalan dengan biaya yang lebih tinggi. Keamanan selalu menjadi hal yang didahulukan dan diutamakan, karena Anda harus melindungi pelanggan Anda.
    - Pelajaran apa hal yang Anda dapatkan?
    - Tindakan korektif apa yang Anda ambil?
        - Item tindakan
        - Item terkait
- Buat prosedur operasi standar yang jelas untuk melakukan analisis setelah insiden.
- Siapkan proses pelaporan insiden standar. Dokumentasikan semua insiden secara komprehensif, termasuk laporan insiden awal, log, komunikasi, dan tindakan yang diambil selama insiden.
- Ingatlah bahwa insiden tidak harus berupa terhentinya sistem. Insiden juga bisa berupa near-miss, atau performa sistem yang tidak sesuai harapan meski tetap memenuhi fungsi bisnisnya.
- Terus tingkatkan proses analisis setelah insiden Anda berdasarkan umpan balik dan pelajaran yang dipetik.
- Tangkap temuan utama dalam sistem manajemen pengetahuan, dan pertimbangkan pola apa pun yang perlu ditambahkan ke dalam panduan developer atau daftar periksa sebelum deployment.

Sumber daya

Dokumen terkait:

- Mengapa Anda harus mengembangkan koreksi kesalahan (COE)

Video terkait:

- Amazon’s approach to failing successfully

- AWS re:Invent 2021 - Amazon Builders' Library: Operational Excellence at Amazon

REL12-BP03 Menguji persyaratan fungsional

Gunakan teknik seperti pengujian unit dan pengujian integrasi yang memvalidasi fungsionalitas.

Anda akan meraih hasil terbaik saat pengujian ini dijalankan secara otomatis sebagai bagian dari tindakan deployment dan build. Misalnya, dengan menggunakan AWS CodePipeline, developer melakukan perubahan ke repositori sumber tempat CodePipeline mendeteksi perubahan secara otomatis. Perubahan tersebut dibangun, dan pengujian dijalankan. Setelah pengujian selesai, kode yang dibangun di-deploy ke server penahapan untuk pengujian. Dari server penahapan, CodePipeline menjalankan lebih banyak pengujian, seperti integrasi atau pengujian beban. Setelah berhasil menyelesaikan pengujian tersebut, CodePipeline melakukan deployment kode yang telah diuji dan disetujui ke instans produksi.

Selain itu, pengalaman menunjukkan bahwa pengujian transaksi sintetis (juga disebut sebagai pengujian canary, tetapi bedakan dengan deployment canary) yang dapat menjalankan dan menyimulasikan perilaku pelanggan adalah salah satu proses pengujian yang paling penting. Jalankan pengujian ini secara konstan terhadap titik akhir beban kerja dari berbagai lokasi jarak jauh. Amazon CloudWatch Synthetics memungkinkan Anda untuk membuat canary untuk memantau titik akhir dan API.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Uji persyaratan fungsional. Hal ini termasuk pengujian unit dan pengujian integrasi yang memvalidasi fungsionalitas yang disyaratkan.
  - Gunakan CodePipeline dengan AWS CodeBuild untuk menguji kode dan menjalankan build
  - AWS CodePipeline Menambahkan Dukungan untuk Unit dan Pengujian Integrasi Kustom dengan AWS CodeBuild
  - Pengiriman Berkelanjutan dan Integrasi Berkelanjutan
  - Menggunakan Canary (Amazon CloudWatch Synthetics)
  - Otomatisasi uji perangkat lunak

Sumber daya

Dokumen terkait:

- Partner APN: partner yang dapat membantu implementasi pipeline integrasi berkelanjutan
- AWS CodePipeline Menambahkan Dukungan untuk Unit dan Pengujian Integrasi Kustom dengan AWS CodeBuild
- AWS Marketplace: produk yang dapat digunakan untuk integrasi berkelanjutan
- Pengiriman Berkelanjutan dan Integrasi Berkelanjutan
- Otomatisasi uji perangkat lunak
- Gunakan CodePipeline dengan AWS CodeBuild untuk menguji kode dan menjalankan build
- Menggunakan Canary (Amazon CloudWatch Synthetics)

REL12-BP04 Menguji persyaratan penskalaan dan kinerja

Gunakan teknik-teknik seperti pengujian beban untuk memvalidasi bahwa beban kerja memenuhi persyaratan kinerja dan penskalaan.

Di dalam cloud, Anda dapat membuat lingkungan pengujian dalam skala produksi sesuai permintaan untuk beban kerja Anda. Jika Anda menjalankan pengujian ini di infrastruktur yang skalanya diturunkan, Anda harus menskalakan hasil observasi Anda menurut apa yang Anda perkirakan terjadi di dalam produksi. Pengujian kinerja dan beban juga dapat dilakukan dalam produksi jika Anda ingin berhati-hati agar tidak berdampak pada pengguna aktual. Tandai data pengujian Anda agar tidak tercampur dengan data pengguna nyata dan mengubah laporan statistik atau produksi.

Dengan pengujian, Anda dapat memastikan bahwa sumber daya dasar, pengaturan penskalaan, kuota layanan, dan desain ketahanan Anda beroperasi sebagaimana mestinya saat menerima beban.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

- Uji persyaratan penskalaan dan kinerja. Jalankan pengujian beban untuk memvalidasi bahwa beban kerja memenuhi persyaratan kinerja dan penskalaan.
  - Pengujian Beban Terdistribusi di AWS: simulasikan ribuan pengguna terhubung
  - Apache JMeter
    - Lakukan deployment aplikasi ke lingkungan yang menyerupai lingkungan produksi Anda, lalu eksekusi pengujian beban.
      - Gunakan infrastruktur sebagai konsep kode untuk membuat lingkungan semirip mungkin dengan lingkungan produksi Anda.

Sumber daya

Dokumen terkait:

- Pengujian Beban Terdistribusi di AWS: simulasikan ribuan pengguna terhubung
- Apache JMeter

REL12-BP05 Menguji ketahanan menggunakan chaos engineering

Jalankan eksperimen chaos secara rutin di lingkungan yang berada dalam atau sedekat mungkin dengan produksi untuk memahami bagaimana sistem Anda merespons kondisi yang merugikan.

Hasil yang diinginkan:

Ketahanan beban kerja diverifikasi secara rutin dengan menerapkan chaos engineering dalam bentuk eksperimen injeksi kesalahan atau injeksi beban tak terduga. Selain itu, terdapat pengujian ketahanan yang memvalidasi perilaku sesuai ekspektasi yang diketahui dari beban kerja Anda selama berlangsungnya sebuah peristiwa. Gabungkan chaos engineering dan pengujian ketahanan agar Anda percaya bahwa beban kerja dapat bertahan dari kegagalan komponen dan dapat pulih dari gangguan tak terduga dengan dampak minimal atau tanpa dampak.

Antipola umum:

- Menentukan desain untuk mendapatkan ketahanan, tetapi tidak memverifikasi bagaimana beban kerja berfungsi secara keseluruhan saat terjadi kesalahan.
- Tidak pernah bereksperimen dalam kondisi dunia nyata dan dengan beban yang diharapkan.
- Tidak memperlakukan eksperimen Anda sebagai kode atau memeliharanya melalui siklus pengembangan.
- Tidak menjalankan eksperimen chaos baik sebagai bagian dari alur CI/CD Anda maupun di luar deployment.
- Tidak menggunakan analisis pascainsiden terdahulu saat menentukan kesalahan mana yang akan digunakan dalam eksperimen.

Manfaat menjalankan praktik terbaik ini: Injeksi kesalahan untuk memverifikasi ketahanan beban kerja Anda akan membuat Anda percaya bahwa prosedur pemulihan dari desain Anda yang tangguh akan efektif jika terjadi kesalahan nyata.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Chaos engineering memberi tim Anda kemampuan untuk terus menginjeksi gangguan (simulasi) dunia nyata dengan cara yang terkontrol di tingkat penyedia layanan, infrastruktur, beban kerja, dan komponen, dengan dampak minimal atau tanpa dampak bagi pelanggan Anda. Hal ini memungkinkan tim Anda belajar dari kesalahan serta mengamati, mengukur, dan meningkatkan ketahanan beban kerja Anda, serta memvalidasi bahwa peringatan akan diluncurkan dan tim mendapatkan notifikasi jika terjadi suatu peristiwa.

Jika dilakukan terus-menerus, chaos engineering dapat menunjukkan kekurangan dalam beban kerja Anda yang, jika dibiarkan tidak ditangani, dapat berdampak negatif pada ketersediaan dan pengoperasian.

> **Note**
>
> Chaos engineering adalah bidang ilmu yang bereksperimen pada sistem guna membangun kepercayaan pada kemampuan sistem untuk bertahan dari kondisi gangguan dalam produksi. – Prinsip-prinsip Chaos Engineering

Jika sistem mampu bertahan dari gangguan ini, eksperimen chaos harus dipertahankan sebagai pengujian regresi otomatis. Dengan demikian, eksperimen chaos harus dilakukan sebagai bagian dari siklus hidup pengembangan sistem (SDLC) Anda dan sebagai bagian dari alur CI/CD Anda.

Untuk memastikan bahwa beban kerja Anda dapat bertahan dari kegagalan komponen, lakukan injeksi peristiwa dunia nyata sebagai bagian dari eksperimen Anda. Misalnya, lakukan eksperimen dengan kehilangan instans Amazon EC2 atau failover instans basis data Amazon RDS utama, lalu verifikasi bahwa beban kerja Anda tidak terpengaruh (atau hanya sedikit terpengaruh). Gunakan kombinasi kesalahan komponen untuk menyimulasikan peristiwa yang mungkin disebabkan oleh gangguan di Zona Ketersediaan.

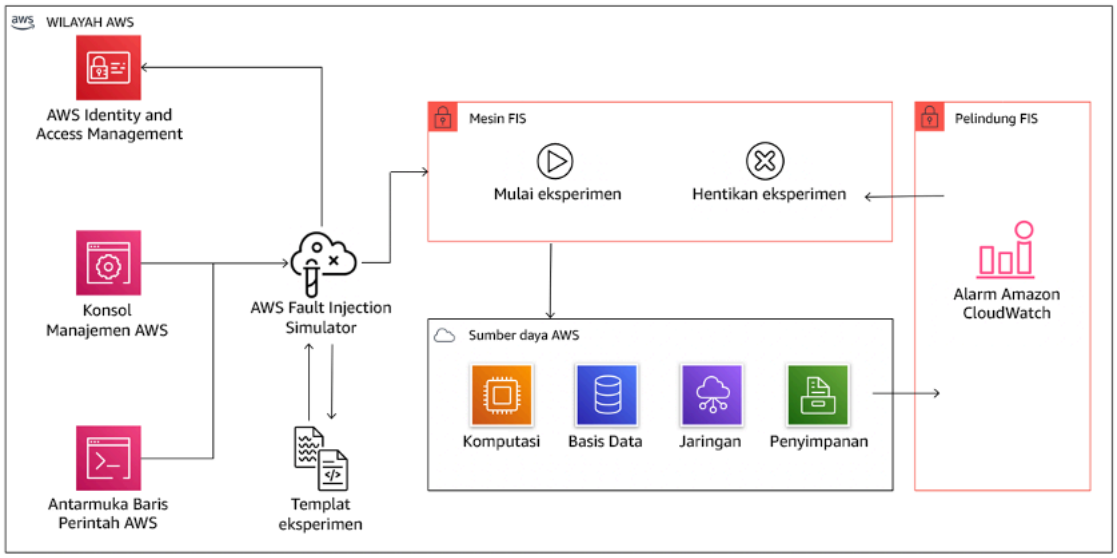
Untuk kesalahan tingkat aplikasi (seperti crash), Anda dapat memulai dengan stressor seperti kehabisan memori dan daya CPU.

Untuk memvalidasi mekanisme fallback atau failover untuk dependensi eksternal karena gangguan jaringan yang terputus-putus, komponen Anda harus menyimulasikan peristiwa tersebut dengan memblokir akses ke penyedia pihak ketiga selama durasi tertentu yang dapat berlangsung dari hitungan detik hingga jam.

Mode degradasi lainnya dapat menyebabkan berkurangnya fungsionalitas dan respons yang lambat, sehingga sering kali mengakibatkan gangguan pada layanan Anda. Degradasi ini umumnya disebabkan oleh peningkatan latensi pada layanan yang sangat penting dan komunikasi jaringan yang tidak dapat diandalkan (paket yang tidak dikirim). Eksperimen dengan kesalahan ini, termasuk efek jaringan seperti latensi, pesan yang tidak terkirim, dan kegagalan DNS, dapat mencakup ketidakmampuan untuk meresolusi nama, menjangkau layanan DNS, atau membuat koneksi ke layanan yang dependen.

Alat chaos engineering:

AWS Fault Injection Service (AWS FIS) adalah layanan terkelola penuh untuk menjalankan eksperimen injeksi kesalahan yang dapat digunakan sebagai bagian dari alur CD Anda, atau di luar alur. AWS FIS adalah pilihan yang baik untuk digunakan selama game day chaos engineering. Layanan ini mendukung penerapan kesalahan secara bersamaan di berbagai jenis sumber daya, termasuk Amazon EC2, Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), dan Amazon RDS. Kesalahan ini termasuk menghentikan sumber daya, memaksa failover, membebani CPU atau memori, throttling, latensi, dan kehilangan paket. Karena layanan ini terintegrasi dengan Amazon CloudWatch Alarms, Anda dapat mengatur kondisi berhenti sebagai pagar pembatas untuk melakukan rollback jika eksperimen menyebabkan dampak tak terduga.



AWS Fault Injection Service terintegrasi dengan sumber daya AWS untuk memungkinkan Anda menjalankan eksperimen injeksi kesalahan untuk beban kerja Anda.

Ada juga beberapa opsi pihak ketiga untuk eksperimen injeksi kesalahan. Opsi ini mencakup alat sumber terbuka seperti Chaos Toolkit, Chaos Mesh, dan Litmus Chaos, serta opsi komersial

seperti Gremlin. Untuk memperluas cakupan kesalahan yang dapat diinjeksikan di AWS, AWS FIS terintegrasi dengan Chaos Mesh dan Litmus Chaos, sehingga Anda dapat mengoordinasikan alur kerja injeksi kesalahan di antara beberapa alat. Misalnya, Anda dapat menjalankan pengujian pada CPU sebuah pod menggunakan kesalahan Chaos Mesh atau Litmus sambil menghentikan sebagian simpul klaster yang dipilih secara acak menggunakan tindakan kesalahan AWS FIS.

Langkah implementasi

- Tentukan kesalahan mana yang akan digunakan untuk eksperimen.

  Lakukan penilaian desain beban kerja Anda untuk mengetahui ketahanannya. Desain tersebut (yang dibuat menggunakan praktik terbaik dari Well-Architected Framework) memperhitungkan risiko berdasarkan dependensi krusial, peristiwa terdahulu, masalah yang diketahui, dan persyaratan kepatuhan. Buat daftar yang berisi setiap elemen desain yang dimaksudkan untuk menjaga ketahanan dan kesalahan yang akan dimitigasi oleh elemen desain tersebut. Untuk informasi lebih lanjut tentang cara membuat daftar tersebut, lihat laporan resmi Operational Readiness Review yang memandu Anda tentang cara membuat proses untuk mencegah pengulangan insiden sebelumnya. Proses Analisis Mode dan Efek Kegagalan (FMEA) memberi Anda kerangka kerja untuk melakukan analisis tingkat komponen terhadap kegagalan dan bagaimana dampaknya terhadap beban kerja Anda. FMEA diuraikan secara lebih mendetail oleh Adrian Cockcroft dalam Failure Modes and Continuous Resilience.
- Tetapkan prioritas untuk setiap kesalahan.

  Mulailah dengan kategorisasi yang umum seperti tinggi, sedang, atau rendah. Untuk menilai prioritas, pertimbangkan frekuensi kesalahan dan dampak kegagalan terhadap beban kerja secara keseluruhan.

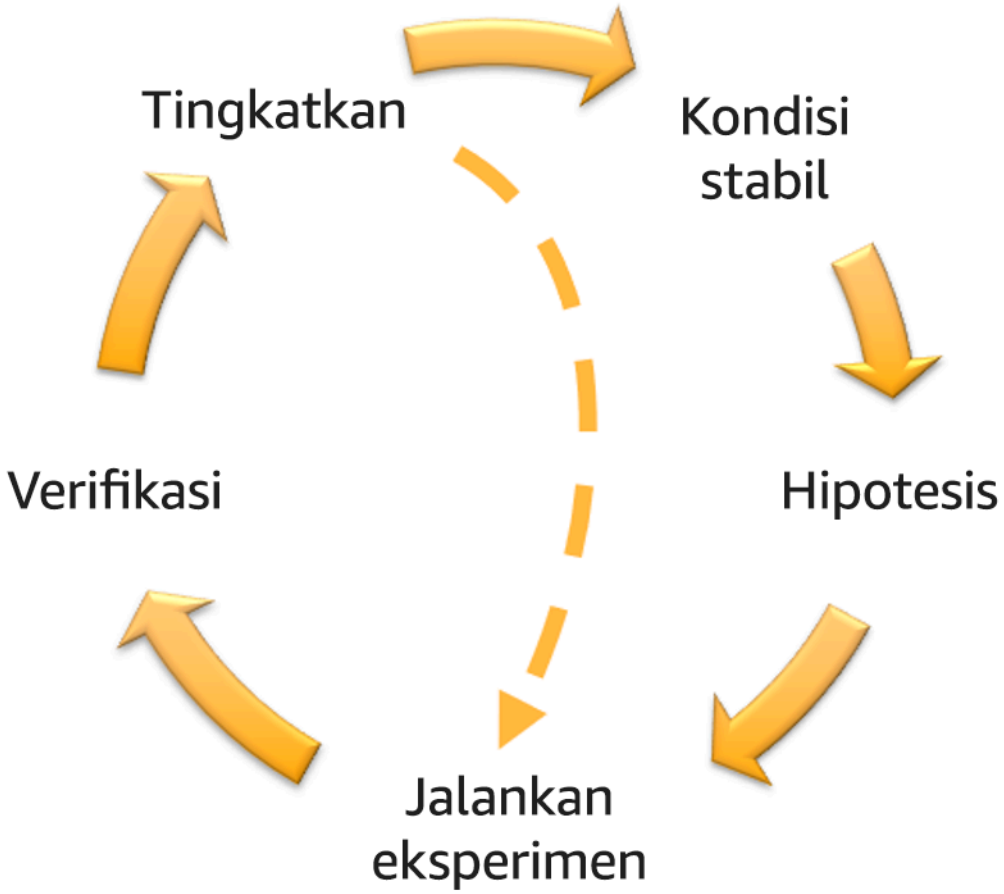
  Saat mempertimbangkan frekuensi kesalahan tertentu, lakukan analisis pada data terdahulu untuk beban kerja ini jika tersedia. Jika tidak tersedia, gunakan data dari beban kerja lain yang berjalan di lingkungan yang serupa.

  Ketika mempertimbangkan dampak dari kesalahan tertentu, makin besar cakupan kesalahan, biasanya makin besar dampaknya. Pertimbangkan juga desain dan tujuan beban kerja. Misalnya, kemampuan untuk mengakses penyimpanan data sumber sangat krusial untuk beban kerja yang melakukan transformasi dan analisis data. Dalam hal ini, Anda akan memprioritaskan eksperimen untuk kesalahan akses, serta akses yang di-throttling dan penyisipan latensi.

  Analisis pascainsiden adalah sumber data yang baik untuk memahami frekuensi dan dampak mode kegagalan.

Gunakan prioritas yang ditetapkan untuk menentukan kesalahan mana yang akan digunakan terlebih dahulu dalam eksperimen beserta urutannya agar dapat mengembangkan eksperimen injeksi kesalahan baru.

- Untuk setiap eksperimen yang Anda lakukan, gunakan roda chaos engineering dan ketahanan berkelanjutan.



Roda chaos engineering dan ketahanan berkelanjutan yang menggunakan metode ilmiah dari Adrian Hornsby.

- Definisikan kondisi stabil sebagai output terukur dari beban kerja yang menunjukkan perilaku normal.

  Beban kerja Anda menunjukkan kondisi stabil jika beroperasi dengan andal dan seperti yang diharapkan. Oleh karena itu, validasikan bahwa beban kerja Anda berkondisi baik sebelum menentukan kondisi stabil. Dalam kondisi stabil, bukan berarti tidak akan ada dampak pada beban kerja saat terjadi kesalahan, karena sejumlah kesalahan tertentu mungkin berada dalam

batas yang dapat diterima. Kondisi stabil adalah acuan dasar yang akan Anda amati selama eksperimen, yang akan menunjukkan anomali jika hipotesis yang Anda tentukan pada langkah berikutnya tidak berjalan seperti yang diharapkan.

Misalnya, kondisi stabil sistem pembayaran dapat didefinisikan sebagai pemrosesan 300 TPS dengan tingkat keberhasilan 99% dan waktu round-trip 500 md.

- Bentuk hipotesis tentang bagaimana beban kerja akan bereaksi terhadap kesalahan.

  Hipotesis yang baik didasarkan pada bagaimana beban kerja diharapkan akan memitigasi kesalahan untuk mempertahankan kondisi stabil. Hipotesis menyatakan bahwa dengan kesalahan jenis tertentu, sistem atau beban kerja akan terus berkondisi stabil karena beban kerja ini dirancang dengan mitigasi tertentu. Jenis spesifik kesalahan dan mitigasi harus ditentukan dalam hipotesis.

  Templat berikut dapat digunakan untuk hipotesis (tetapi pernyataan lain juga dapat diterima):

  > **Note**
  >
  > Jika *[kesalahan tertentu]* terjadi, beban kerja *[nama beban kerja]* akan *[deskripsikan kontrol mitigasi]* untuk mempertahankan *[dampak metrik bisnis atau teknis]*.

  Misalnya:

  - Jika 20% dari total simpul dalam grup simpul Amazon EKS dihapus, Transaction Create API akan terus melayani persentil ke-99 dari permintaan dalam waktu kurang dari 100 md (kondisi stabil). Simpul Amazon EKS akan pulih dalam waktu lima menit, dan pod akan dijadwalkan dan memproses lalu lintas dalam waktu delapan menit setelah dimulainya eksperimen. Peringatan akan diaktifkan dalam waktu tiga menit.
  - Jika terjadi kegagalan instans Amazon EC2 tunggal, pemeriksaan kondisi Elastic Load Balancing untuk sistem pemesanan akan membuat Elastic Load Balancing hanya mengirim permintaan ke instans berkondisi baik yang tersisa, sedangkan Amazon EC2 Auto Scaling mengganti instans yang gagal, sehingga mempertahankan peningkatan kesalahan sisi server (5xx) sebanyak kurang dari 0,01% (kondisi stabil).
  - Jika instans basis data Amazon RDS utama gagal, beban kerja pengumpulan data Rantai Pasokan akan melakukan failover dan terhubung ke instans basis data Amazon RDS yang

siaga untuk mempertahankan kesalahan baca atau tulis basis data selama kurang dari 1 menit (kondisi stabil).

- Jalankan eksperimen dengan menginjeksikan kesalahan.

  Eksperimen secara default harus memiliki kemampuan fail-safe dan ditoleransi oleh beban kerja. Jika Anda tahu bahwa beban kerja akan gagal, jangan jalankan eksperimen. Chaos engineering harus digunakan untuk menemukan “known-unknown” atau “unknown-unknown”. “Known-unknown” adalah hal-hal yang Anda ketahui, tetapi tidak sepenuhnya dipahami, dan “unknown-unknown” adalah hal-hal yang tidak Anda ketahui atau pahami sepenuhnya. Bereksperimen dengan beban kerja yang Anda tahu dalam kondisi rusak tidak akan memberi Anda wawasan baru. Eksperimen Anda harus direncanakan dengan cermat, memiliki cakupan dampak yang jelas, dan menyediakan mekanisme rollback yang dapat diterapkan jika terjadi gangguan tak terduga. Jika uji tuntas Anda menunjukkan bahwa beban kerja Anda dapat bertahan dalam eksperimen, lanjutkan eksperimen. Ada beberapa opsi untuk menginjeksikan kesalahan. Untuk beban kerja di AWS, AWS FIS menyediakan banyak simulasi kesalahan standar yang disebut tindakan. Anda juga dapat menentukan tindakan kustom yang berjalan di AWS FIS menggunakan dokumen AWS Systems Manager.

  Kami tidak menyarankan penggunaan skrip kustom untuk eksperimen chaos, kecuali jika skrip tersebut memiliki kemampuan untuk memahami status terkini beban kerja, mampu menghasilkan log, dan menyediakan mekanisme untuk rollback dan kondisi berhenti jika memungkinkan.

  Kerangka kerja atau kumpulan alat efektif yang mendukung chaos engineering harus melacak kondisi terkini eksperimen, menghasilkan log, dan menyediakan mekanisme rollback untuk mendukung pelaksanaan eksperimen yang terkontrol. Mulailah dengan layanan andal seperti AWS FIS yang memungkinkan Anda melakukan eksperimen dengan cakupan yang jelas dan mekanisme keamanan yang melakukan rollback jika eksperimen menimbulkan gangguan tak terduga. Untuk mempelajari tentang beragam variasi eksperimen menggunakan AWS FIS, lihat juga lab Aplikasi Tangguh dan Well-Architected dengan Chaos Engineering. Selain itu, AWS Resilience Hub akan menganalisis beban kerja Anda dan membuat eksperimen yang dapat Anda pilih untuk diterapkan dan dijalankan di AWS FIS.

  > **Note**
  >
  > Untuk setiap eksperimen, pahami dengan jelas cakupan dan dampaknya. Kami merekomendasikan bahwa kesalahan harus disimulasikan terlebih dahulu di lingkungan nonproduksi sebelum dijalankan dalam produksi.

Eksperimen harus dijalankan dalam produksi dengan beban dunia nyata menggunakan deployment canary yang melakukan deployment sistem kontrol dan eksperimental, jika memungkinkan. Menjalankan eksperimen selama waktu sepi adalah praktik yang baik untuk mengurangi potensi dampak saat pertama kali bereksperimen dalam produksi. Selain itu, jika menggunakan lalu lintas pelanggan yang sebenarnya akan menimbulkan terlalu banyak risiko, Anda dapat menjalankan eksperimen menggunakan lalu lintas sintetis di infrastruktur produksi terhadap deployment kontrol dan eksperimental. Jika tidak dapat menggunakan produksi, jalankan eksperimen di lingkungan praproduksi yang semirip mungkin dengan produksi.

Anda harus membuat dan memantau pagar pembatas untuk memastikan eksperimen tidak memengaruhi lalu lintas produksi atau sistem lain di luar batas yang dapat diterima. Tetapkan kondisi berhenti untuk menghentikan eksperimen jika mencapai ambang batas pada metrik pagar pembatas yang Anda tentukan. Hal ini harus mencakup metrik untuk kondisi stabil beban kerja, serta metrik berdasarkan komponen yang diinjeksi dengan kesalahan. Sebuah pemantauan sintetis (juga dikenal sebagai user canary) adalah salah satu metrik yang biasanya harus Anda sertakan sebagai proksi pengguna. Kondisi berhenti untuk AWS FIS didukung sebagai bagian dari templat eksperimen, sehingga memungkinkan maksimal lima kondisi berhenti per templat.

Salah satu prinsip chaos adalah meminimalkan cakupan eksperimen dan dampaknya:

Meskipun harus ada kelonggaran untuk beberapa dampak negatif dalam jangka pendek, Chaos Engineer bertanggung jawab dan berkewajiban untuk memastikan gangguan dari eksperimen diminimalkan dan dikendalikan.

Metode untuk memverifikasi cakupan dan dampak potensial adalah dengan melakukan eksperimen di lingkungan nonproduksi terlebih dahulu, memverifikasi bahwa ambang batas untuk kondisi berhenti diaktifkan seperti yang diharapkan selama eksperimen dan kemampuan pengamatan diterapkan untuk menemukan pengecualian, bukan langsung bereksperimen dalam produksi.

Saat menjalankan eksperimen injeksi kesalahan, verifikasikan bahwa semua pihak yang bertanggung jawab sudah mengetahui informasi yang jelas. Berkomunikasilah dengan tim yang sesuai seperti tim operasi, tim keandalan layanan, dan dukungan pelanggan untuk memberi tahu mereka kapan eksperimen akan dijalankan dan apa yang diharapkan. Berikan alat komunikasi kepada berbagai tim ini untuk memberi tahu tim tertentu yang menjalankan eksperimen jika muncul efek yang merugikan.

Anda harus memulihkan beban kerja dan sistem yang mendasarinya kembali ke kondisi awal yang diketahui berfungsi baik. Sering kali, desain beban kerja yang tangguh akan pulih sendiri. Namun, beberapa desain yang salah atau eksperimen yang gagal dapat membuat beban kerja Anda berada dalam kondisi kegagalan yang tidak terduga. Pada akhir eksperimen, Anda harus menyadari hal ini dan memulihkan beban kerja dan sistem. Dengan AWS FIS, Anda dapat mengatur konfigurasi rollback (juga disebut post action) dalam parameter tindakan. Post action mengembalikan target ke keadaan sebelum tindakan dijalankan. Baik diotomatiskan (seperti menggunakan AWS FIS) maupun manual, post action ini harus menjadi bagian dari playbook yang menjelaskan cara mendeteksi dan menangani kegagalan.

- Verifikasikan hipotesisnya.

  Prinsip-prinsip Chaos Engineering memberikan panduan tentang cara memverifikasi kondisi stabil beban kerja Anda:

  Fokus pada output terukur dari suatu sistem, bukan atribut internal sistem. Pengukuran output tersebut selama periode waktu yang singkat merupakan proksi untuk kondisi stabil sistem. Throughput sistem secara keseluruhan, tingkat kesalahan, dan persentil latensi semuanya dapat menjadi metrik penting yang merepresentasikan perilaku kondisi stabil. Dengan berfokus pada pola perilaku sistemik selama eksperimen, chaos engineering memverifikasi bahwa sistem berfungsi, bukan mencoba memvalidasi cara kerjanya.

  Dalam dua contoh sebelumnya, kami menyertakan metrik kondisi stabil dengan peningkatan kesalahan sisi server (5xx) sebanyak kurang dari 0,01% serta kesalahan baca dan tulis basis data selama kurang dari satu menit.

  Kesalahan 5xx adalah metrik yang baik karena merupakan konsekuensi dari mode kegagalan yang akan dialami langsung oleh klien yang menggunakan beban kerja. Pengukuran kesalahan basis data cocok digunakan sebagai konsekuensi langsung dari kesalahan, tetapi juga harus dilengkapi dengan pengukuran dampak klien seperti permintaan pelanggan yang gagal atau kesalahan yang muncul bagi klien. Selain itu, sertakan pemantauan sintetis (juga dikenal sebagai user canary) pada API atau URI apa pun yang diakses langsung oleh klien yang menggunakan beban kerja Anda.

- Tingkatkan desain beban kerja agar memiliki ketahanan.

  Jika kondisi stabil tidak dipertahankan, selidiki cara desain beban kerja dapat ditingkatkan untuk mengurangi kesalahan, dengan menerapkan praktik terbaik dari pilar Keandalan AWS Well-Architected. Panduan dan sumber daya tambahan dapat ditemukan di AWS Builder's Library,

yang berisi artikel tentang cara meningkatkan pemeriksaan kondisi Anda atau menerapkan percobaan ulang dengan backoff dalam kode aplikasi Anda, dll.

Setelah perubahan ini diterapkan, jalankan eksperimen lagi (ditunjukkan dengan garis putus-putus pada roda chaos engineering) untuk mengetahui keefektifannya. Jika langkah verifikasi menunjukkan bahwa hipotesisnya benar, beban kerja akan berada dalam kondisi stabil, dan siklusnya berlanjut.

- Jalankan eksperimen secara rutin.

  Eksperimen chaos adalah sebuah siklus, dan eksperimen harus dijalankan secara rutin sebagai bagian dari chaos engineering. Setelah beban kerja memenuhi hipotesis eksperimen, eksperimen harus diotomatiskan untuk terus berjalan sebagai bagian regresi dalam alur CI/CD Anda. Untuk mempelajari cara melakukannya, lihat blog tentang cara menjalankan eksperimen AWS FIS menggunakan AWS CodePipeline. Lab tentang eksperimen AWS FIS berulang dalam alur CI/CD memungkinkan Anda melakukan praktik langsung.

  Eksperimen injeksi kesalahan juga merupakan bagian dari game day (lihat REL12-BP06 Mengadakan game day secara rutin). Game day mensimulasikan kegagalan atau peristiwa untuk memverifikasi sistem, proses, dan respons tim. Tujuannya adalah untuk benar-benar menerapkan tindakan yang perlu dilakukan oleh tim seolah memang terjadi peristiwa yang tidak diharapkan.

- Catat dan simpan hasil eksperimen.

  Hasil eksperimen injeksi kesalahan harus dicatat dan dijadikan persisten. Sertakan semua data yang diperlukan (seperti waktu, beban kerja, dan kondisi) agar dapat menganalisis hasil dan tren eksperimen nantinya. Contoh hasilnya dapat mencakup tangkapan layar dasbor, dump CSV dari basis data metrik Anda, atau catatan ketik manual yang berisi peristiwa dan pengamatan dari eksperimen. Pencatatan log eksperimen dengan AWS FIS dapat menjadi bagian dari pencatatan data ini.

Sumber daya

Praktik terbaik terkait:

- REL08-BP03 Mengintegrasikan pengujian ketahanan sebagai bagian dari deployment Anda
- REL13-BP03 Menguji implementasi pemulihan bencana untuk memvalidasi implementasi

Dokumen terkait:

- Apa itu AWS Fault Injection Service?
- Apa itu AWS Resilience Hub?
- Prinsip-prinsip Chaos Engineering
- Chaos Engineering: Merencanakan eksperimen pertama Anda
- Rekayasa Ketahanan: Belajar untuk Mengatasi Kegagalan
- Kisah Chaos Engineering
- Menghindari fallback dalam sistem terdistribusi
- Deployment Canary untuk Eksperimen Chaos

Video terkait:

- AWS re:Invent 2020: Menguji ketahanan menggunakan chaos engineering (ARC316)
- AWS re:Invent 2019: Meningkatkan ketahanan dengan chaos engineering (DOP309-R1)
- AWS re:Invent 2019: Melakukan chaos engineering di dunia nirserver (CMY301)

Contoh terkait:

- Lab Well-Architected: Level 300: Pengujian Ketahanan Amazon EC2, Amazon RDS, dan Amazon S3
- Lab Chaos Engineering di AWS
- lab Aplikasi Tangguh dan Well-Architected dengan Chaos Engineering
- Lab Chaos Nirserver
- Lab Ukur dan Tingkatkan Ketahanan Aplikasi Anda dengan AWS Resilience Hub

Alat terkait:

- AWS Fault Injection Service
- AWS Marketplace: Platform Chaos Engineering Gremlin
- Chaos Toolkit
- Chaos Mesh
- Litmus

REL12-BP06 Mengadakan game day secara rutin

Manfaatkan game day untuk secara rutin melatih prosedur Anda dalam merespons peristiwa dan kegagalan. Buat game day semirip mungkin dengan produksi (termasuk lingkungan produksi) bersama orang-orang yang akan terlibat dalam skenario kegagalan aktual. Game day menerapkan tindakan yang diperlukan guna memastikan peristiwa produksi tidak berdampak pada pengguna.

Game day menyimulasikan kegagalan atau peristiwa untuk menguji respons tim, sistem, dan proses. Tujuannya adalah untuk benar-benar menerapkan tindakan yang perlu dilakukan oleh tim seolah memang terjadi peristiwa yang tidak diharapkan. Hal ini akan membantu Anda memahami sisi mana yang perlu ditingkatkan dan membantu mengembangkan pengalaman organisasi dalam menangani peristiwa. Aktivitas ini harus dilakukan secara rutin untuk memperkuat memori otot dalam merespons kejadian tersebut.

Setelah desain ketangguhan Anda diterapkan dan diuji dalam lingkungan nonproduksi, game day dapat menjadi cara untuk memastikan bahwa segala sesuatu akan berjalan sesuai rencana ketika produksi. Game day, terutama yang dilakukan untuk pertama kali, merupakan aktivitas “wajib untuk semua tim”. Rekayasawan dan operasi akan diberitahu kapan ini dilakukan, dan apa yang akan terjadi. Runbook telah diterapkan. Simulasi peristiwa, termasuk peristiwa kegagalan yang mungkin terjadi, dieksekusi di sistem produksi dengan cara yang sudah ditentukan, dan dampaknya dievaluasi. Jika sistem beroperasi sesuai rancangan, deteksi dan pemulihan mandiri akan berlangsung dengan sedikit atau tanpa dampak. Namun, jika timbul dampak negatif, pengujian akan diulang dan masalah beban kerja diperbaiki, secara manual jika perlu (menggunakan runbook). Karena game day biasanya berlangsung di dalam produksi, semua pencegahan harus dilakukan guna memastikan bahwa ketersediaan untuk pelanggan tidak terganggu.

Antipola umum:

- Mendokumentasikan prosedur Anda, tetapi tidak pernah melatihnya.
- Tidak melibatkan pembuat keputusan bisnis dalam pengujian pelatihan.

Manfaat menerapkan praktik terbaik ini: Mengadakan game day secara rutin memastikan bahwa staf mengikuti kebijakan dan prosedur ketika insiden aktual terjadi, dan memvalidasi bahwa kebijakan dan prosedur tersebut sudah sesuai.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Jadwalkan game day untuk menggunakan runbook dan buku pedoman Anda secara rutin. Game day harus mengikutsertakan semua orang yang akan terlibat dalam kejadian produksi: pemilik bisnis, staf pengembangan, staf operasional, dan tim respons insiden.
  - Jalankan pengujian beban atau kinerja Anda, kemudian jalankan injeksi kegagalan.
  - Cari anomali dalam runbook Anda dan peluang untuk menggunakan buku pedoman Anda.
    - Jika Anda tidak mengikuti runbook, perbaiki runbook atau koreksi perilakunya. Jika Anda menggunakan buku pedoman, identifikasi buku pedoman yang seharusnya digunakan atau buat yang baru.

Sumber daya

Dokumen terkait:

- Apa itu AWS GameDay?

Video terkait:

- AWS re:Invent 2019: Meningkatkan ketahanan dengan chaos engineering (DOP309-R1)

Contoh terkait:

- Lab AWS Well-Architected - Pengujian Ketangguhan

REL 13. Bagaimana cara Anda mempersiapkan pemulihan bencana (DR)?

Memiliki cadangan dan komponen beban kerja berlebih adalah permulaan dari strategi DR Anda. RTO dan RPO merupakan tujuan Anda untuk pemulihan beban kerja Anda. Tetapkan ini berdasarkan kebutuhan bisnis. Implementasikan strategi untuk memenuhi tujuan-tujuan ini, sambil mempertimbangkan lokasi dan fungsi data dan sumber daya beban kerja. Probabilitas gangguan dan biaya pemulihan juga merupakan faktor penting yang membantu menginformasikan nilai bisnis dari penyediaan pemulihan bencana untuk beban kerja.

Praktik terbaik

- REL13-BP01 Tetapkan sasaran pemulihan untuk waktu henti dan kehilangan data

- REL13-BP02 Menggunakan strategi pemulihan yang ditentukan untuk memenuhi sasaran pemulihan
- REL13-BP03 Menguji implementasi pemulihan bencana untuk memvalidasi implementasi
- REL13-BP04 Mengelola penyimpangan konfigurasi di lokasi atau Wilayah Pemulihan Bencana (DR)
- REL13-BP05 Mengotomatiskan pemulihan

REL13-BP01 Tetapkan sasaran pemulihan untuk waktu henti dan kehilangan data

Beban kerja memiliki sasaran waktu pemulihan (RTO) dan sasaran titik pemulihan (RPO).

Sasaran Waktu Pemulihan (RTO) adalah penundaan maksimum yang dapat diterima antara gangguan layanan dan pemulihan layanan. Ini menentukan apa yang dianggap sebagai jendela waktu yang dapat diterima ketika layanan tidak tersedia.

Sasaran Titik Pemulihan (RPO) adalah jumlah waktu maksimum yang dapat diterima sejak titik pemulihan data terakhir. Ini menentukan apa yang dianggap sebagai kehilangan data yang dapat diterima antara titik pemulihan terakhir dan gangguan layanan.

Nilai RTO dan RPO merupakan pertimbangan penting ketika memilih strategi Pemulihan Bencana (DR) yang sesuai untuk beban kerja Anda. Sasaran-sasaran ini ditentukan oleh bisnis, kemudian digunakan oleh tim teknis untuk memilih dan mengimplementasikan strategi DR.

Hasil yang Diinginkan:

Setiap beban kerja memiliki penetapan RTO dan RPO, yang ditetapkan berdasarkan dampak bisnis. Beban kerja ditetapkan ke tingkat yang telah ditetapkan sebelumnya, yang menetapkan ketersediaan layanan dan kehilangan data yang dapat diterima, dengan RTO dan RPO terkait. Jika penetapan tingkat tersebut tidak dapat dilakukan, maka ini dapat diberi tingkat khusus yang disesuaikan per beban kerja, dengan maksud untuk membuat tingkat di lain waktu. RTO dan RPO digunakan sebagai salah satu pertimbangan utama untuk pemilihan implementasi strategi pemulihan bencana untuk beban kerja. Pertimbangan tambahan dalam memilih strategi DR yakni kendala biaya, ketergantungan beban kerja, dan persyaratan operasional.

Untuk RTO, pahami dampak berdasarkan durasi pemadaman. Apakah implikasinya linier, atau adakah implikasi non-linier? (contohnya, setelah empat jam, Anda mematikan jalur produksi sampai dimulainya giliran kerja berikutnya).

Matriks pemulihan bencana, seperti berikut ini, dapat membantu Anda memahami bagaimana kritikalitas beban kerja berkaitan dengan sasaran pemulihan. (Perhatikan, nilai aktual untuk sumbu X dan Y harus disesuaikan dengan kebutuhan organisasi Anda).

Matriks Pemulihan Bencana						
		Sasaran Titik Pemulihan				
		< 1 Menit	< 1 Jam	< 6 Jam	< 1 Hari	+ 1 Hari
Sasaran Waktu Pemulihan	< 10 Menit	Kritis	Kritis	Tinggi	Sedang	Sedang
	< 2 Jam	Kritis	Tinggi	Sedang	Sedang	Rendah
	< 8 Jam	Tinggi	Sedang	Sedang	Rendah	Rendah
	< 24 Jam	Sedang	Sedang	Rendah	Rendah	Rendah
	Lebih dari 24 Jam	Sedang	Rendah	Rendah	Rendah	Rendah

Gambar 16: Matriks pemulihan bencana

Antipola umum:

- Tidak ditetapkan sasaran pemulihan.
- Memilih sasaran pemulihan semaunya.
- Memilih sasaran pemulihan yang terlalu longgar dan tidak memenuhi tujuan bisnis.
- Tidak memahami dampak waktu henti dan kehilangan data.
- Memilih sasaran pemulihan yang tidak realistis, seperti tanpa adanya waktu untuk pemulihan dan tanpa adanya kehilangan data, yang mungkin tidak dapat dicapai untuk konfigurasi beban kerja Anda.
- Memilih sasaran pemulihan yang lebih ketat daripada tujuan bisnis yang sesungguhnya. Ini memaksakan implementasi DR yang lebih mahal dan lebih rumit dibandingkan yang dibutuhkan beban kerja.
- Memilih sasaran pemulihan yang tidak kompatibel dengan sasaran beban kerja yang bergantung.
- Sasaran pemulihan Anda tidak mempertimbangkan persyaratan kepatuhan terhadap peraturan.
- RTO dan RPO ditetapkan untuk beban kerja, tetapi tidak pernah diuji.

Manfaat menerapkan praktik terbaik ini: Sasaran pemulihan Anda untuk waktu dan kehilangan data diperlukan untuk memandu implementasi DR Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

Untuk beban kerja tertentu, Anda harus memahami dampak waktu henti dan kehilangan data pada bisnis Anda. Umumnya, dampak akan semakin meningkat jika waktu henti atau kehilangan data semakin besar, tetapi bentuk peningkatan ini bisa berbeda, tergantung pada jenis beban kerjanya. Contohnya, Anda mungkin dapat menoleransi waktu henti hingga satu jam dengan dampak kecil, tetapi setelah itu dampaknya meningkat dengan cepat. Ada banyak bentuk dampak pada bisnis, termasuk kerugian moneter (seperti hilangnya pendapatan), hilangnya kepercayaan pelanggan (dan dampak pada reputasi), masalah operasional (seperti penurunan produktivitas atau gaji tidak terbayarkan), dan risiko yang terkait dengan peraturan. Gunakan langkah-langkah berikut untuk memahami dampak-dampak ini, dan tetapkan RTO dan RPO untuk beban kerja Anda.

Langkah Implementasi

1. Tentukan pemangku kepentingan bisnis Anda untuk beban kerja ini, dan libatkan mereka untuk mengimplementasikan langkah-langkah ini. Sasaran pemulihan untuk beban kerja merupakan keputusan bisnis. Kemudian tim teknis bekerja dengan pemangku kepentingan bisnis untuk menggunakan sasaran-sasaran ini untuk memilih strategi DR.

   > **Note**
   >
   > Untuk langkah 2 dan 3, Anda dapat menggunakan the section called “Lembar kerja implementasi”.

2. Kumpulkan informasi yang diperlukan untuk mengambil keputusan dengan menjawab pertanyaan-pertanyaan di bawah ini.
3. Apakah Anda memiliki kategori atau tingkat kritikalitas untuk dampak beban kerja di organisasi Anda?
   a. Jika ya, tetapkan beban kerja ini ke salah satu kategori
   b. Jika tidak, maka tetapkan kategori-kategori ini. Buat lima kategori atau lebih sedikit dan sempurnakan rentang sasaran waktu pemulihan Anda untuk setiap kategori. Contoh kategori antara lain: kritis, tinggi, sedang, rendah. Untuk memahami cara pemetaan beban kerja ke kategori, pertimbangkan apakah beban kerja itu kritis untuk misi perusahaan, penting bagi bisnis, atau tidak mendorong bisnis.
   c. Tetapkan RTO dan RPO beban kerja berdasarkan kategori. Selalu pilih kategori yang lebih ketat (RTO dan RPO lebih rendah) daripada nilai mentah yang dihitung saat memasuki langkah

ini. Jika ini menghasilkan perubahan nilai yang besar dan tidak sesuai, maka pertimbangkan untuk membuat kategori baru.

4. Berdasarkan jawaban-jawaban ini, tetapkan nilai RTO dan RPO ke beban kerja. Ini dapat dilakukan secara langsung, atau dengan menetapkan beban kerja ke tingkat layanan yang ditetapkan sebelumnya.
5. Dokumentasikan rencana pemulihan bencana (DRP) untuk beban kerja ini, yang merupakan bagian dari rencana keberlangsungan bisnis (BCP) organisasi Anda, di lokasi yang dapat diakses oleh pemangku kepentingan dan tim beban kerja
   a. Catat RTO dan RPO, dan informasi yang digunakan untuk menentukan nilai-nilai ini. Sertakan strategi yang digunakan untuk mengevaluasi dampak beban kerja pada bisnis
   b. Catat metrik lain selain RTO dan RPO yang Anda lacak, atau rencanakan untuk melacak sasaran pemulihan bencana
   c. Anda akan menambahkan detail strategi DR Anda dan runbook pada rencana ini ketika Anda membuat ini.
6. Dengan mencari kritikalitas beban kerja di dalam matriks seperti yang ada dalam Gambar 15, Anda dapat mulai menetapkan tingkat layanan yang ditetapkan di muka untuk organisasi Anda.
7. Setelah Anda mengimplementasikan strategi DR (atau bukti konsep untuk strategi DR) sesuai the section called “REL13-BP02 Menggunakan strategi pemulihan yang ditentukan untuk memenuhi sasaran pemulihan”, uji strategi ini untuk menentukan RPC (Kemampuan Titik Pemulihan) dan RTC (Kemampuan Waktu Pemulihan) aktual beban kerja. Jika ini tidak memenuhi sasaran pemulihan target, maka bekerjalah dengan pemangku kepentingan bisnis Anda untuk menyesuaikan sasaran-sasaran tersebut, atau buat perubahan pada strategi DR yang memungkinkan untuk memenuhi sasaran target.

Pertanyaan utama

1. Berapakah waktu henti maksimum untuk beban kerja sebelum timbul dampak serius pada bisnis?
   a. Tentukan kerugian moneter (dampak finansial langsung) pada bisnis per menit jika beban kerja terganggu.
   b. Pertimbangkan bahwa dampak tidak selalu linier. Pada awalnya, dampak bisa terbatas, tetapi kemudian meningkat dengan cepat melampaui titik kritis dalam waktu.
2. Berapakah jumlah data maksimum yang bisa hilang sebelum timbul dampak serius pada bisnis?
   a. Pertimbangkan nilai ini untuk penyimpanan data Anda yang paling kritis. Identifikasi kritikalitas masing-masing untuk penyimpanan data lainnya.

   b. Dapatkah data beban kerja dibuat jika hilang? Jika hal ini secara operasional lebih mudah daripada mencadangkan dan memulihkan, maka pilih RPO berdasarkan kritikalitas data sumber yang digunakan untuk membuat ulang data beban kerja.

3. Apa saja sasaran pemulihan dan harapan ketersediaan beban kerja yang hal ini andalkan (hilir), atau beban kerja yang mengandalkan hal ini (hulu)?
   a. Pilih sasaran pemulihan yang memampukan beban kerja ini untuk memenuhi persyaratan ketergantungan hulu
   b. Pilih sasaran pemulihan yang dapat dicapai mengingat kemampuan pemulihan ketergantungan hilir. Ketergantungan hilir non-kritis (yang dapat Anda “tangani”) dapat dikecualikan. Atau, bekerjalah dengan ketergantungan hilir kritis atau tingkatkan kemampuan pemulihannya apabila perlu.

Pertanyaan tambahan

Pertimbangkan pertanyaan-pertanyaan ini, dan bagaimana pertanyaan tersebut mungkin berlaku pada beban kerja ini:

4. Apakah Anda memiliki RTO dan RPO yang berbeda, tergantung pada jenis pemadaman (Wilayah vs. AZ, dll.)?
5. Apakah ada waktu spesifik (musim, acara penjualan, peluncuran produk) ketika RTO/RPO Anda mungkin berubah? Jika ya, apakah batas waktu dan pengukurannya yang berbeda?
6. Berapa jumlah pelanggan yang akan terkena dampak jika beban kerja terganggu?
7. Apakah dampak pada reputasi jika beban kerja terganggu?
8. Dampak operasional lain apakah yang dapat timbul jika beban kerja terganggu? Contohnya, dampak pada produktivitas karyawan jika sistem email tidak tersedia, atau jika sistem Gaji tidak dapat mengirimkan transaksi.
9. Bagaimanakah RTO dan RPO beban kerja sesuai dengan Strategi DR Organisasi dan Bidang Bisnis?
10. Apakah ada kewajiban kontrak internal untuk memberikan layanan? Apakah ada penalti jika tidak memenuhinya?
11. Apa saja kendala kepatuhan atau peraturan terkait data?

Lembar kerja implementasi

Anda dapat menggunakan lembar kerja ini untuk langkah implementasi 2 dan 3. Anda dapat menyesuaikan lembar kerja ini agar cocok dengan kebutuhan spesifik Anda, seperti menambahkan pertanyaan tambahan.

Langkah 2: Pertanyaan utama	Berlaku untuk beban kerja?	RTO beban kerja	RPO beban kerja	Penyesuaian RTO.	Penyesuaian RPO.	Instruksi
[1] waktu maksimum beban kerja dapat mengalami waktu henti						diukur pada waktunya dari mulai pemadaman hingga pemulihan
[2] jumlah maksimum data yang bisa hilang						diukur pada waktunya sejak set data dapat dipulihkan berkualitas baik terakhir diketahui
[3a] dependensi hulu						masukkan tujuan pemulihan hilir yang paling ketat
[3b] dependensi hilir						masukkan tujuan pemulihan hilir yang paling longgar
[3a] dependensi hulu yang direkonsiliasi						Jika nilai hulu kurang dari nilai saat ini dan nilai hilir lebih besar, bekerjalah dengan
[3b] dependensi hilir yang direkonsiliasi						dependensi untuk merekonsiliasi dan masukkan nilai yang direkonsiliasi di sini
[3] dependensi						turunkan nilai untuk memenuhi dependensi hulu atau naikkan berdasarkan kemampuan dependensi hilir
Langkah 2: Pertanyaan tambahan						Tunjukkan apakah pertanyaan berlaku. Jika tidak, lewati
RTO/RPO dasar						Turunkan nilai RTO dan RPO dari atas ke sini
[4] tipe pemadaman	[ ]Y / [ ]N					Masukkan tujuan pemulihan untuk tipe peristiwa dengan persyaratan paling ketat
[5] tujuan berbasis waktu khusus	[ ]Y / [ ]N					Masukkan tujuan pemulihan untuk waktu-waktu dengan persyaratan paling ketat
[6] pelanggan terganggu	[ ]Y / [ ]N					Buat grafik pelanggan yang terkena dampak saat fungsi mengalami waktu henti atau data hilang. Gunakan grafik tersebut untuk memasukkan RTO dan RPO maksimum yang diizinkan berdasarkan dampak pelanggan
[7] dampak reputasi	[ ]Y / [ ]N					Bekerjalah dengan bisnis untuk menentukan RTO dan RPO berdasarkan dampak terhadap reputasi
[8] dampak operasi	[ ]Y / [ ]N					Masukkan RTO dan RPO maksimum berdasarkan dampak operasional
[9] penyelarasan organisasi	[ ]Y / [ ]N					Masukkan RTO dan RPO maksimum untuk beban kerja tipe ini sesuai LOB dan persyaratan organisasi
[10] kewajiban kontrak	[ ]Y / [ ]N					Masukkan RTO dan RPO maksimum berdasarkan kewajiban kontrak
[11] kepatuhan peraturan	[ ]Y / [ ]N					Masukkan RTO dan RPO maksimum berdasarkan kepatuhan peraturan yang berlaku
target berdasarkan pertanyaan tambahan						Ambil nilai minimum (nilai yang lebih ketat) dari Q 4-11 dan masukkan di sini
target yang disesuaikan						Jika tujuan di baris di atas tidak dapat diakomodasi, bekerjalah dengan pemangku kepentingan untuk mengurai hambatan, dan masukkan jumlah minimum baru di sini
RTO/RPO yang disesuaikan						Masukkan nilai RPO/RTO acuan, atau target yang disesuaikan, mana saja yang lebih rendah
Langkah 3						
Peta ke kategori atau tingkat yang ditetapkan sebelumnya						Sesuaikan kedua nilai ke bawah (lebih ketat) agar selaras dengan tingkat yang ditetapkan terdekat

Lembar kerja

Tingkat upaya untuk Rencana Implementasi: Rendah

Sumber daya

Praktik Terbaik Terkait:

- the section called “REL09-BP04 Melakukan pemulihan data secara berkala untuk memverifikasi integritas dan proses pencadangan”
- the section called “REL13-BP02 Menggunakan strategi pemulihan yang ditentukan untuk memenuhi sasaran pemulihan”
- the section called “REL13-BP03 Menguji implementasi pemulihan bencana untuk memvalidasi implementasi”

Dokumen terkait:

- Blog Arsitektur AWS: Seri Pemulihan Bencana
- Pemulihan Bencana Beban Kerja di AWS: Pemulihan di Cloud (Laporan Resmi AWS)
- Mengelola kebijakan ketangguhan dengan Pusat Ketangguhan AWS
- Partner APN: partner yang dapat membantu pemulihan bencana
- AWS Marketplace: produk yang dapat digunakan untuk pemulihan bencana

Video terkait:

- AWS re:Invent 2018: Pola Arsitektur untuk Aplikasi Aktif-Aktif Multi-Wilayah (ARC209-R2)
- Pemulihan Bencana Beban Kerja di AWS

REL13-BP02 Menggunakan strategi pemulihan yang ditentukan untuk memenuhi sasaran pemulihan

Tentukan strategi pemulihan bencana (DR) yang memenuhi sasaran pemulihan beban kerja. Pilih strategi seperti pencadangan dan pemulihan, standby (aktif/pasif), atau aktif/aktif.

Hasil yang diinginkan: Strategi DR ditentukan dan diimplementasikan untuk setiap beban kerja agar beban kerja dapat mencapai sasaran DR. Strategi DR antara beban kerja menggunakan pola yang dapat digunakan kembali (seperti strategi yang telah dijelaskan sebelumnya),

Antipola umum:

- Mengimplementasikan prosedur pemulihan yang tidak konsisten untuk beban kerja dengan sasaran DR yang serupa.
- Membiarkan strategi DR diimplementasikan secara ad-hoc saat bencana terjadi.
- Tidak memiliki rencana untuk pemulihan bencana.
- Dependensi pada operasi bidang kendali selama pemulihan.

Manfaat menjalankan praktik terbaik ini:

- Dengan strategi pemulihan yang ditentukan, Anda dapat menggunakan prosedur tes dan peralatan umum.
- Menggunakan strategi pemulihan yang ditentukan akan meningkatkan penyebaran pengetahuan antara tim dan implementasi DR pada beban kerja milik mereka.

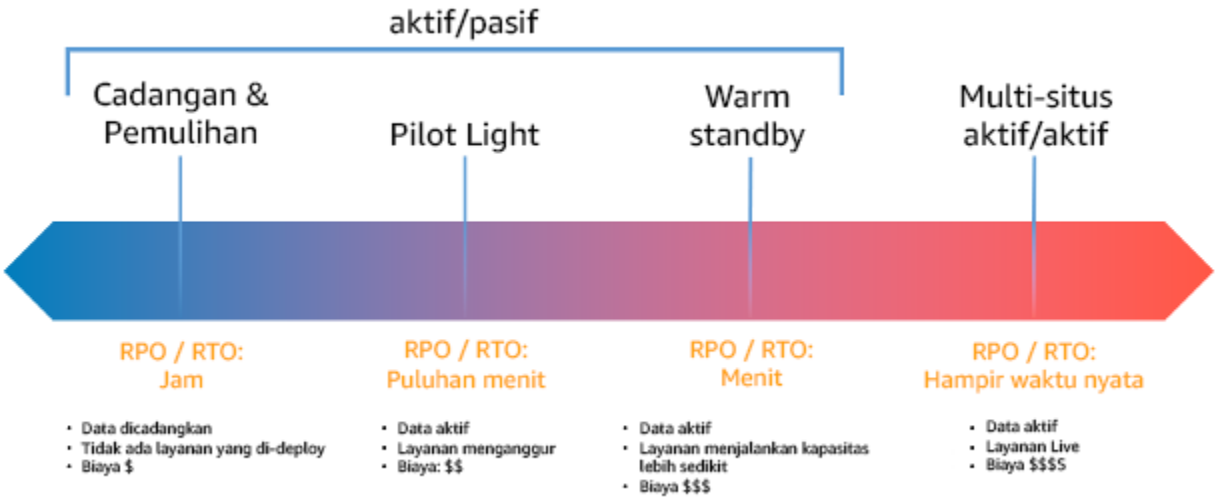
Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi. Tanpa strategi DR yang direncanakan, diimplementasikan, dan diuji, Anda akan kesulitan mencapai sasaran pemulihan ketika bencana terjadi.

Panduan implementasi

Strategi DR mengandalkan kemampuan untuk mempertahankan beban kerja di situs pemulihan jika lokasi utama tidak dapat menjalankan beban kerja. Sasaran pemulihan yang paling umum adalah RTO dan RPO, seperti yang didiskusikan dalam REL13-BP01 Tetapkan sasaran pemulihan untuk waktu henti dan kehilangan data.

Strategi DR di beberapa Zona Ketersediaan (AZ) dalam Wilayah AWS tunggal, dapat menyediakan mitigasi bencana seperti kebakaran, banjir, dan pemadaman listrik besar-besaran. Anda dapat menggunakan strategi DR yang menggunakan beberapa Wilayah jika memang perlu mengimplementasikan perlindungan terhadap peristiwa yang membuat beban kerja tidak dapat dijalankan di Wilayah AWS.

Anda harus memilih salah satu dari strategi berikut saat merancang strategi DR di beberapa Wilayah. Strategi didaftar dan diurutkan berdasarkan biaya dan kompleksitas dari kecil ke besar, serta diurutkan berdasarkan RTO dan RPO dari besar ke kecil. Wilayah Pemulihan berarti Wilayah AWS selain dari yang utama yang digunakan untuk beban kerja Anda.



Gambar 17: Strategi pemulihan bencana (DR)

- Pencadangan dan pemulihan (RPO dalam jam, RTO dalam 24 jam atau kurang): Cadangkan data dan aplikasi ke dalam Wilayah pemulihan. Menggunakan pencadangan otomatis atau berkelanjutan dapat mengaktifkan pemulihan titik waktu, yang dalam beberapa kasus dapat

menurunkan RPO hingga 5 menit. Saat terjadi bencana, Anda akan melakukan deployment infrastruktur (menggunakan infrastruktur sebagai kode untuk mengurangi RTO), melakukan deploymennt kode, dan memulihkan data yang dicadangkan untuk memulihkan dari bencana di Wilayah pemulihan.

- Pilot light (RPO dalam menit, RTO dalam kelipatan sepuluh menit): Sediakan salinan infrastruktur beban kerja inti di Wilayah pemulihan. Replikasikan data ke Wilayah pemulihan dan buat cadangan di sana. Sumber daya yang diperlukan untuk mendukung replikasi dan pencadangan data, misalnya basis data dan penyimpanan objek, selalu aktif. Elemen lainnya seperti server aplikasi atau komputasi nirserver tidak di-deploy, tetapi dapat dibuat saat dibutuhkan dengan kode aplikasi dan konfigurasi yang diperlukan.
- Warm standby (RPO dalam detik, RTO dalam menit): Pertahankan beban kerja dalam versi yang diturunkan skalanya tetapi berfungsi sepenuhnya yang selalu dijalankan di Wilayah pemulihan. Sistem bisnis kritis sepenuhnya digandakan dan selalu diaktifkan, tetapi dengan armada yang diturunkan skalanya. Data direplikasi dan berada dalam Wilayah pemulihan. Saat memasuki waktu pemulihan, sistem dinaikkan skalanya dengan cepat untuk menangani beban produksi. Semakin warm standby dinaikkan skalanya, akan semakin rendah pengandalan RTO dan bidang kendali. Ketika diskalakan sepenuhnya, ini disebut sebagai hot standby.
- Multi-Wilayah (multi-situs) aktif-aktif (RPO mendekati nol, RTO berpotensi nol): Beban kerja di-deploy ke, dan aktif menangani lalu lintas dari, beberapa Wilayah AWS. Strategi ini perlu menyinkronkan data di seluruh Wilayah. Konflik potensial yang disebabkan oleh menulis catatan yang sama di dua replika wilayah yang berbeda harus dihindari atau ditangani, karena bisa menjadi kompleks. Replikasi data bermanfaat untuk sinkronisasi data dan akan melindungi Anda terhadap beberapa jenis bencana, tetapi tidak melindungi terhadap kerusakan atau kehilangan data kecuali solusi juga disertai opsi untuk pemulihan titik waktu.

> **Note**
>
> Perbedaan antara pilot light dan warm standby terkadang sulit dimengerti. Keduanya menyertakan lingkungan di Wilayah pemulihan dengan salinan aset wilayah utama. Perbedaannya adalah pilot light tidak dapat memproses permintaan tanpa lebih dulu melakukan tindakan tambahan, sedangkan warm standby dapat menangani lalu lintas (pada kapasitas yang dikurangi) dengan cepat. Pilot light mengharuskan Anda mengaktifkan server, menaikkan skala, dan mungkin mengharuskan Anda melakukan deployment infrastruktur tambahan (bukan inti). Sementara itu, warm standby hanya meminta Anda untuk menaikkan skala (semuanya sudah di-deploy dan dijalankan). Pilih berdasarkan kebutuhan RTO dan RPO Anda.

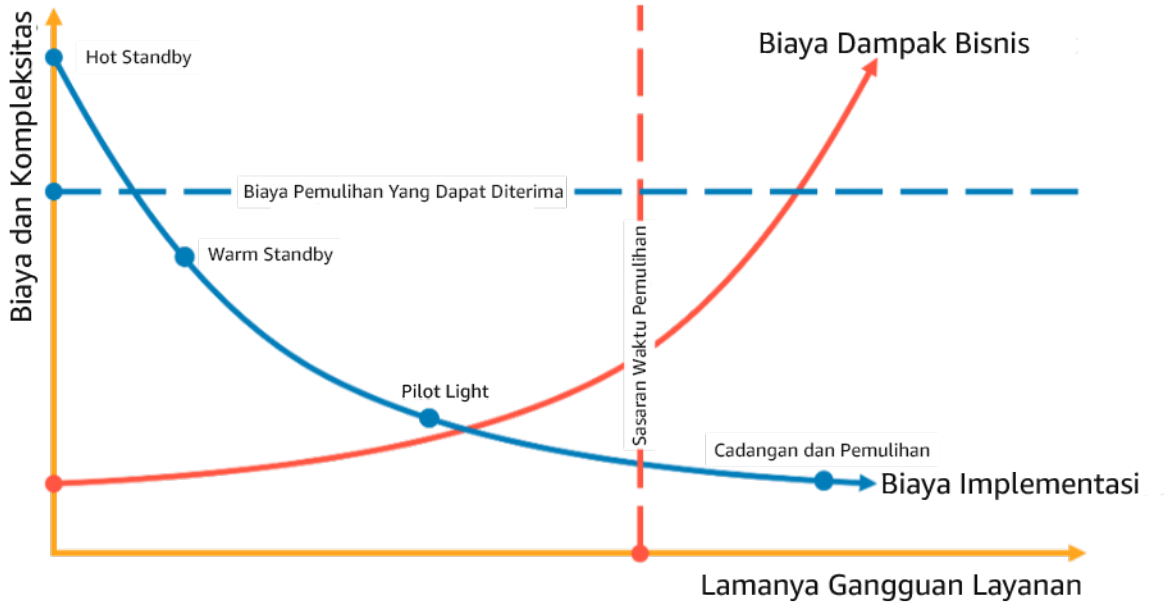
Apabila ada kekhawatiran tentang biaya, dan Anda ingin mencapai sasaran RPO dan RTO yang serupa dengan yang ditetapkan dalam strategi warm standby, Anda dapat mempertimbangkan solusi cloud native, seperti AWS Elastic Disaster Recovery, yang mengambil pendekatan pilot light dan menawarkan target RPO dan RTO lebih baik.

Langkah implementasi

1. Tentukan strategi DR yang akan memenuhi persyaratan pemulihan untuk beban kerja ini.

Saat memilih strategi DR, Anda harus memilih antara mengurangi waktu henti dan kehilangan data (RTO dan RPO) dan meningkatkan biaya dan kompleksitas untuk mengimplementasikan strategi, atau sebaliknya. Sebaiknya hindari strategi yang lebih sulit dari yang dibutuhkan, karena hal ini akan menambah biaya yang tidak perlu.

Misalnya, dalam diagram berikut, bisnis telah menentukan RTO maksimum yang diizinkan serta batas yang dapat digunakan pada strategi pemulihan layanan. Berdasarkan sasaran bisnis, strategi DR pilot light atau warm standby akan memenuhi kriteria biaya dan RTO.



Gambar 18: Pemilihan strategi DR berdasarkan RTO dan biaya

Untuk mempelajari selengkapnya, lihat Rencana Keberlangsungan Bisnis (BCP).

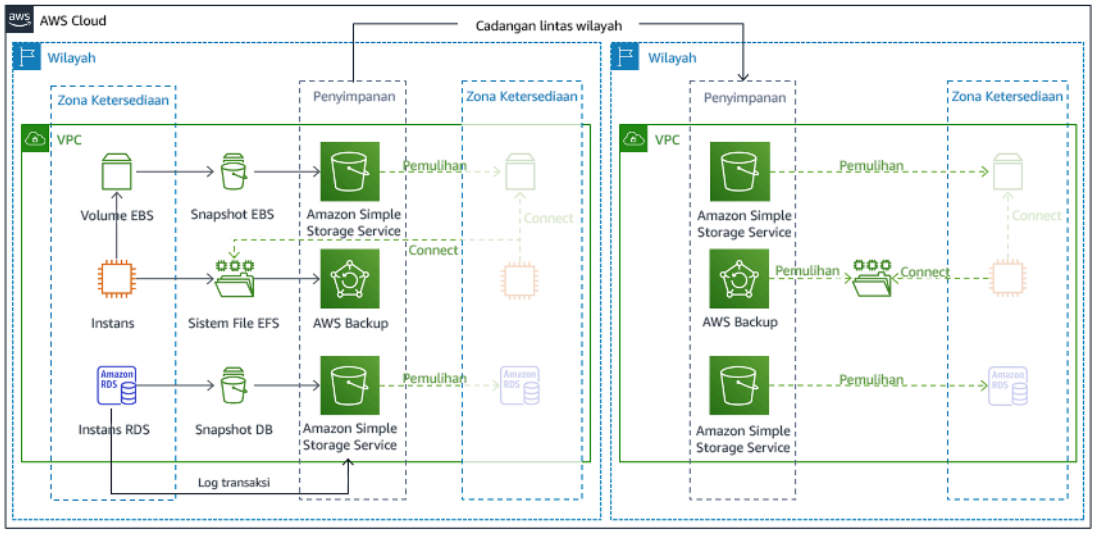
2. Tinjau pola tentang bagaimana strategi DR yang dipilih dapat diimplementasikan.

Langkah ini digunakan untuk memahami cara Anda mengimplementasikan strategi yang dipilih. Strategi dijelaskan menggunakan Wilayah AWS sebagai situs utama dan pemulihan. Namun, Anda juga dapat memilih untuk menggunakan Zona Ketersediaan dalam Wilayah tunggal sebagai strategi DR, yang menggunakan beberapa elemen dari berbagai strategi tersebut.

Dalam langkah berikut ini, Anda dapat menerapkan strategi pada beban kerja spesifik Anda.

Pencadangan dan pemulihan

Pencadangan dan pemulihan adalah strategi yang tidak terlalu kompleks untuk diimplementasikan, tetapi akan memerlukan waktu dan usaha lebih untuk mengembalikan beban kerja, sehingga RTO dan RPO menjadi lebih tinggi. Sebaiknya selalu buat cadangan data, dan salin cadangan tersebut ke situs lain (misalnya Wilayah AWS lain).

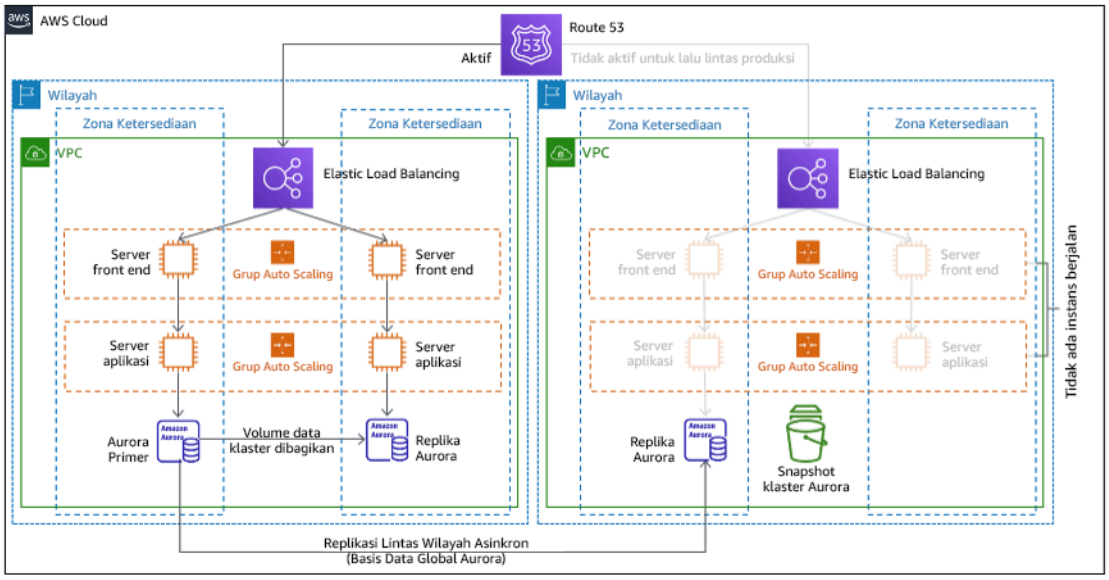


Gambar 19: Arsitektur pencadangan dan pemulihan

Untuk detail selengkapnya tentang strategi ini, lihat Arsitektur Pemulihan Bencana (DR) di AWS, Bagian II: Pencadangan dan Pemulihan dengan Pemulihan Cepat.

Pilot light

Dengan pendekatan pilot light, Anda mereplikasi data dari Wilayah utama ke Wilayah pemulihan. Sumber daya inti yang digunakan untuk infrastruktur beban kerja di-deploy di Wilayah pemulihan. Namun, sumber daya tambahan dan dependensi lainnya masih diperlukan untuk membuat tumpukan fungsional ini. Misalnya, dalam gambar 20, tidak ada instans komputasi yang di-deploy.

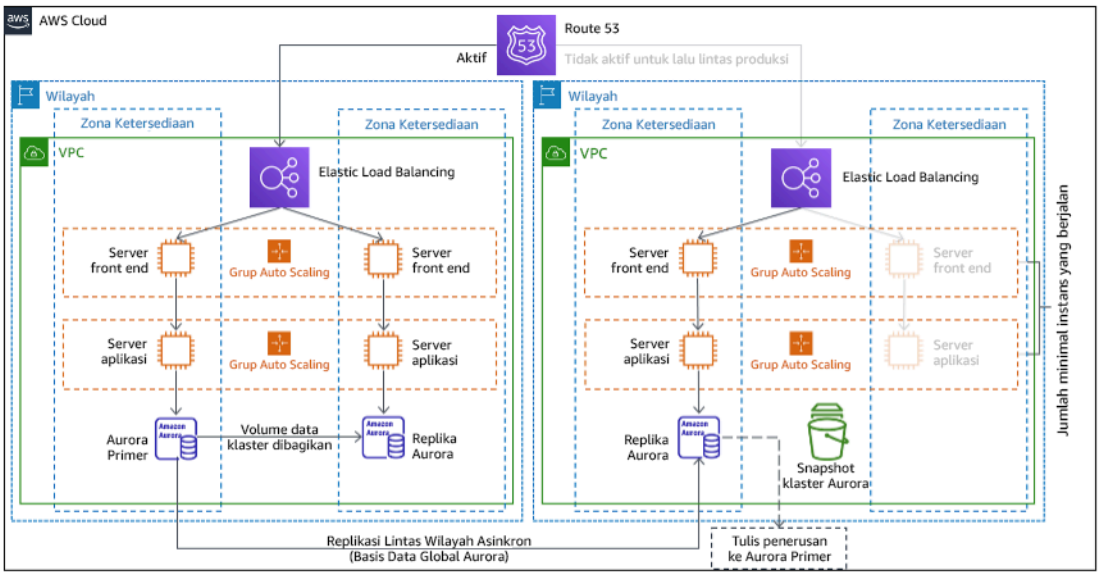


Gambar 20: Arsitektur pilot light

Untuk detail selengkapnya tentang strategi ini, lihat Arsitektur Pemulihan Bencana (DR) di AWS, Bagian III: Pilot Light dan Warm Standby.

Warm standby

Pendekatan warm standby memastikan ada salinan lingkungan produksi yang skalanya diturunkan tetapi berfungsi sepenuhnya di Wilayah lainnya. Pendekatan ini memperpanjang konsep pilot light dan mempercepat waktu pemulihan karena beban kerja selalu aktif di Wilayah lainnya. Jika Wilayah pemulihan di-deploy pada kapasitas penuh, hal ini disebut dengan hot standby.



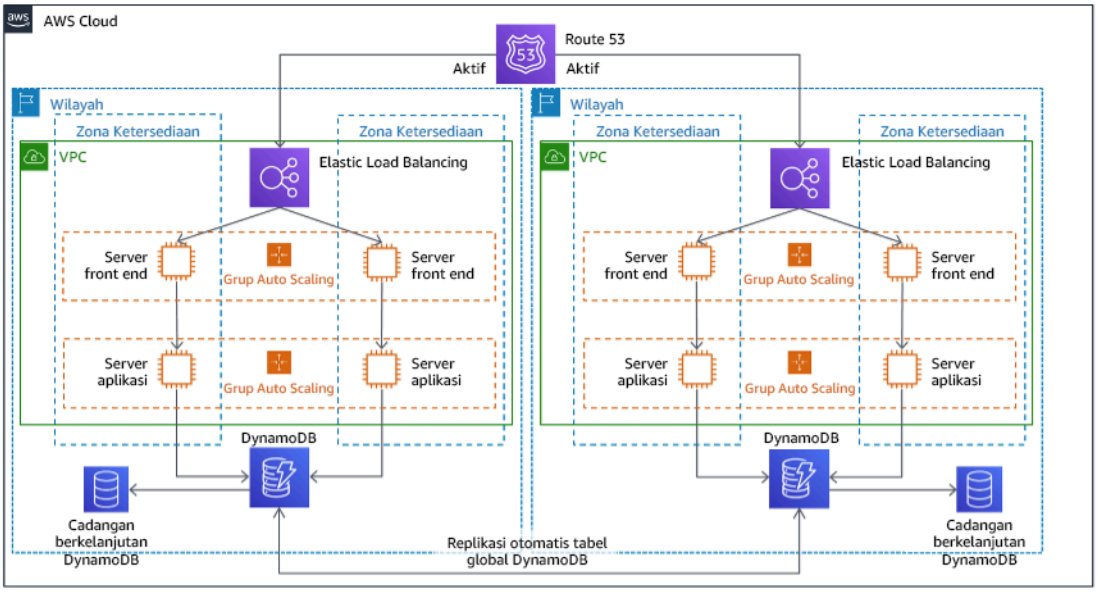
Gambar 21: Arsitektur warm standby

Saat menggunakan warm standby atau pilot light, Anda perlu menaikkan skala sumber daya di Wilayah pemulihan. Untuk memverifikasi kapasitas tersedia ketika diperlukan, pertimbangkan penggunaan reservasi kapasitas untuk instans EC2. Jika menggunakan AWS Lambda, maka konkurensi yang disediakan dapat menyediakan lingkungan pelaksanaan sehingga siap untuk merespons dengan segera ke panggilan fungsi Anda.

Untuk detail selengkapnya tentang strategi ini, lihat Arsitektur Pemulihan Bencana (DR) di AWS, Bagian III: Pilot Light dan Warm Standby.

Multi-situs aktif/aktif

Anda dapat menjalankan beban kerja secara bersamaan di beberapa Wilayah sebagai bagian dari strategi multi-situs aktif/aktif. Multi-situs aktif/aktif menjalankan lalu lintas dari semua wilayah ke wilayah tempatnya di-deploy. Pelanggan dapat memilih strategi ini untuk alasan selain DR. Strategi ini dapat digunakan untuk meningkatkan ketersediaan, atau saat melakukan deployment beban kerja ke audiens global (untuk menempatkan titik akhir lebih dekat dengan pengguna dan/atau melakukan deployment tumpukan yang dilokalkan untuk audiens di wilayah tersebut). Sebagai strategi DR, jika beban kerja tidak dapat didukung di salah satu dari Wilayah AWS tempatnya di-deploy, Wilayah tersebut dievakuasi, dan Wilayah sisanya digunakan untuk mempertahankan ketersediaan. Multi-situs aktif/aktif adalah strategi DR yang paling sulit dioperasikan, dan sebaiknya hanya dipilih saat persyaratan bisnis mengharuskannya.



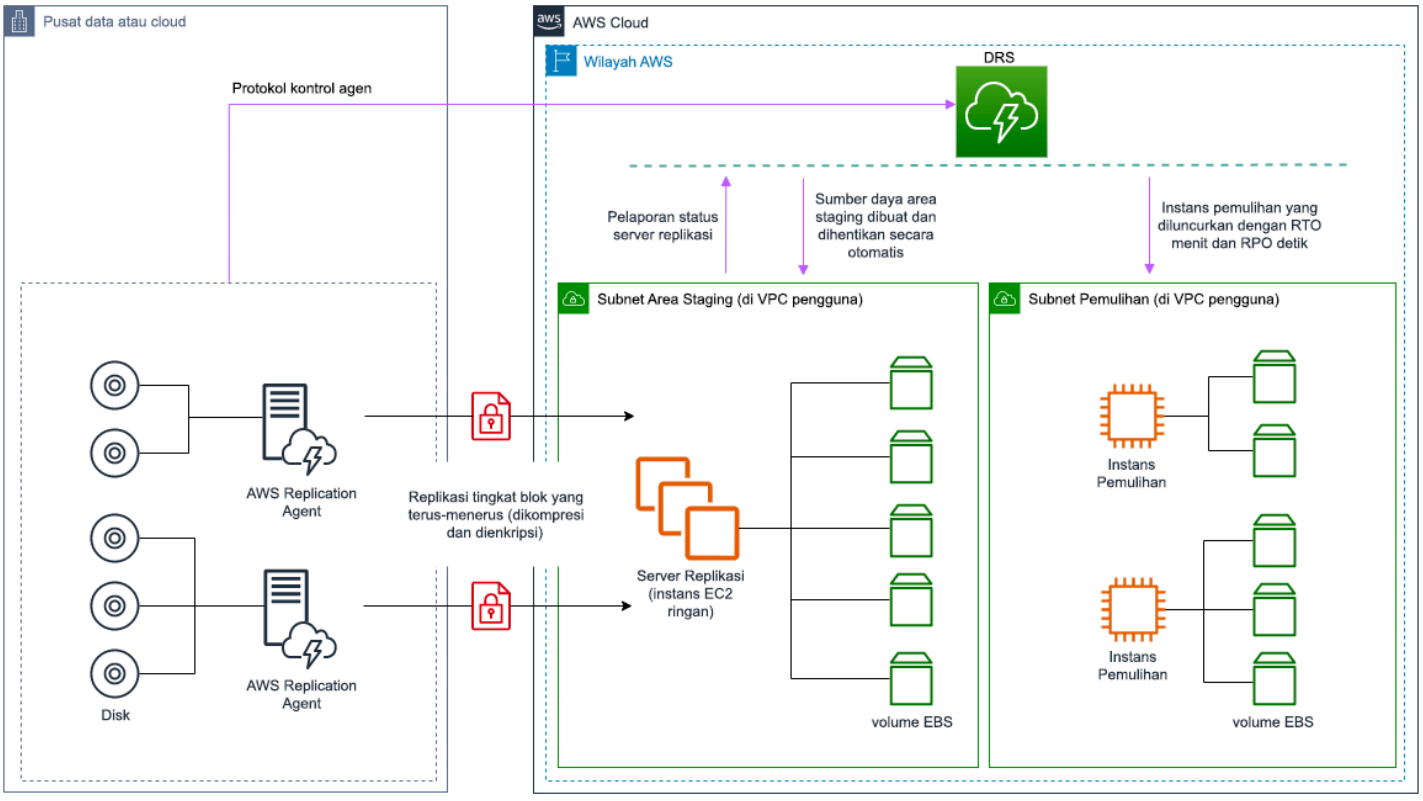
Gambar 22: Arsitektur multi-situs aktif/aktif

Untuk detail selengkapnya tentang strategi ini, lihat Arsitektur Pemulihan Bencana (DR) di AWS, Bagian IV: Multi-situs Aktif/Aktif.

AWS Elastic Disaster Recovery

Jika Anda mempertimbangkan strategi pilot light atau warm standby untuk pemulihan bencana, AWS Elastic Disaster Recovery dapat memberikan pendekatan alternatif dengan peningkatan manfaat. Elastic Disaster Recovery dapat menawarkan target RPO dan RTO yang serupa dengan warm standby, tetapi mempertahankan pendekatan pilot light dengan biaya rendah. Elastic Disaster Recovery mereplikasi data Anda dari wilayah utama ke Wilayah pemulihan, menggunakan perlindungan data berkelanjutan untuk mencapai RPO yang diukur dalam detik dan RTO yang dapat diukur dalam menit. Hanya sumber daya yang diperlukan untuk mereplikasi data yang di-deploy di wilayah pemulihan, yang menekan biaya tetap rendah, serupa dengan strategi pilot light. Ketika menggunakan Elastic Disaster Recovery, layanan mengoordinasi dan mengatur pemulihan sumber daya komputasi ketika dimulai sebagai bagian dari failover atau latihan.

Arsitektur umum AWS Elastic Disaster Recovery (AWS DRS)



Gambar 23: arsitektur AWS Elastic Disaster Recovery

Praktik tambahan untuk melindungi data

Dengan semua strategi, Anda juga harus melakukan mitigasi terhadap bencana data. Replikasi data berkelanjutan melindungi Anda terhadap beberapa jenis bencana, tetapi tidak melindungi terhadap kerusakan atau kehilangan data kecuali strategi juga disertai versioning data yang disimpan atau opsi pemulihan titik waktu. Selain replika, Anda juga harus mencadangkan data yang direplikasi di situs pemulihan untuk membuat pencadangan titik waktu.

Menggunakan beberapa Zona Ketersediaan (AZ) dalam Wilayah AWS tunggal

Saat menggunakan beberapa AZ dalam Wilayah tunggal, implementasi DR Anda menggunakan beberapa elemen dari strategi di atas. Anda harus terlebih dahulu membuat arsitektur ketersediaan tinggi (HA) menggunakan beberapa AZ yang ditampilkan dalam Gambar 23. Arsitektur ini memanfaatkan pendekatan multi-situs aktif/aktif, karena instans Amazon EC2 dan Penyeimbang Beban Elastis memiliki sumber daya yang di-deploy di beberapa AZ, yang secara aktif menangani permintaan. Arsitektur ini juga mendemonstrasikan hot standby, di mana jika instans Amazon RDS utama gagal (atau AZ itu sendiri gagal), maka instans standby dipromosikan ke utama.

Gambar 24: Arsitektur Multi-AZ

Selain arsitektur HA ini, Anda perlu menambahkan cadangan data yang dibutuhkan untuk menjalankan beban kerja. Hal ini sangat penting untuk data yang dibatasi ke zona tunggal seperti volume Amazon EBS atau klaster Amazon Redshift. Jika sebuah AZ gagal, Anda perlu memulihkan data ini ke AZ lainnya. Jika memungkinkan, Anda perlu menyalin cadangan data ke Wilayah AWS sebagai lapisan perlindungan tambahan.

Pendekatan alternatif yang kurang umum untuk DR multi-AZ Wilayah tunggal diilustrasikan di posting blog ini, Membangun aplikasi yang sangat tangguh menggunakan Pengontrol Pemulihan Aplikasi Amazon Route 53, Bagian 1: Tumpukan Wilayah Tunggal. Strategi yang digunakan di sini adalah mempertahankan isolasi sebanyak mungkin di antara AZ, seperti bagaimana Wilayah dioperasikan. Dengan menggunakan strategi alternatif ini, Anda dapat memilih pendekatan aktif/aktif atau aktif/pasif.

> **Note**
>
> Beberapa beban kerja memiliki persyaratan residensi data peraturan. Jika ini diterapkan untuk beban kerja di lokalitas yang saat ini hanya memiliki satu Wilayah AWS, maka multi-Wilayah tidak akan sesuai untuk kebutuhan bisnis. Strategi multi-AZ memberikan perlindungan yang baik terhadap sebagian besar bencana.

3. Evaluasikan sumber daya beban kerja, dan seperti apa konfigurasinya di Wilayah pemulihan sebelum failover (selama operasi normal).

Untuk infrastruktur dan sumber daya AWS, gunakan infrastruktur sebagai kode seperti AWS CloudFormation atau alat pihak ketiga seperti Hashicorp Terraform. Untuk melakukan deployment di beberapa akun dan Wilayah dengan operasi tunggal, Anda dapat menggunakan AWS CloudFormation StackSets. Untuk strategi Multi-situs aktif/aktif dan Hot Standby, infrastruktur yang di-deploy di Wilayah pemulihan memiliki sumber daya yang sama seperti Wilayah utama. Untuk strategi Pilot Light dan Warm Standby, infrastruktur yang di-deploy memerlukan tindakan tambahan agar berubah menjadi siap produksi. Dengan menggunakan parameter dan logika bersyarat CloudFormation, Anda dapat mengontrol tumpukan yang di-deploy agar aktif atau standby dengan templat tunggal. Ketika menggunakan Elastic Disaster Recovery, layanan akan mereplikasi dan mengatur pemulihan konfigurasi aplikasi dan sumber daya komputasi.

Semua strategi DR memerlukan sumber data yang dicadangkan dalam Wilayah AWS, dan cadangan tersebut disalin ke Wilayah pemulihan. AWS Backup memberikan tampilan terpusat tempat Anda dapat mengonfigurasi, menjadwalkan, dan memantau cadangan untuk sumber daya ini. Untuk Pilot Light, Warm Standby, dan Multi-situs aktif/aktif, Anda juga harus mereplikasi data dari Wilayah utama ke sumber daya data di Wilayah pemulihan, seperti instans DB Amazon Relational Database Service (Amazon RDS) atau tabel Amazon DynamoDB. Dengan demikian, sumber data ini aktif dan siap menangani permintaan di Wilayah pemulihan.

Untuk mempelajari lebih lanjut tentang cara layanan AWS beroperasi di seluruh Wilayah, lihat seri blog ini di Membuat Aplikasi Multi-Wilayah dengan Layanan AWS.

4. Tentukan dan implementasikan cara Anda mempersiapkan Wilayah untuk failover saat dibutuhkan (selama peristiwa bencana).

Untuk multi-situs aktif/aktif, failover berarti mengevakuasi Wilayah dan mengandalkan Wilayah aktif yang tersisa. Secara umum, Wilayah tersebut siap menerima lalu lintas. Untuk strategi Pilot Light dan Warm Standby, tindakan pemulihan perlu mencakup deployment sumber daya yang hilang, seperti instans EC2 dalam Gambar 20, juga sumber daya yang hilang lainnya.

Untuk semua strategi di atas, Anda mungkin perlu mengubah instans hanya-baca basis data menjadi instans baca/tulis.

Untuk pencadangan dan pemulihan, pemulihan data dari cadangan menghasilkan sumber daya untuk data tersebut seperti volume EBS, instans RDS DB, dan tabel DynamoDB. Anda juga perlu memulihkan infrastruktur dan melakukan deployment kode. Anda dapat menggunakan AWS Backup untuk memulihkan data di Wilayah pemulihan. Lihat REL09-BP01 Mengidentifikasi dan mencadangkan data yang perlu dicadangkan, atau memproduksi ulang data dari sumber untuk detail lebih lanjut. Saat membangun kembali infrastruktur, Anda juga membuat sumber daya seperti instans EC2 sebagai tambahan untuk Amazon Virtual Private Cloud (Amazon VPC), subnet, dan grup keamanan yang diperlukan. Anda dapat mengotomatiskan banyak proses pemulihan. Untuk mempelajari caranya, lihat posting blog ini.

5. Tentukan dan implementasikan cara Anda merutekan kembali lalu lintas ke failover saat dibutuhkan (selama peristiwa bencana).

Operasi failover ini dapat dimulai secara otomatis dan manual. Failover yang dimulai secara otomatis berdasarkan pemeriksaan kondisi atau alarm harus digunakan dengan hati-hati karena failover yang tidak perlu (alarm palsu) dapat dikenakan biaya seperti ketidaktersediaan dan kehilangan data. Oleh karena itu, Failover yang dimulai secara manual sering digunakan. Dalam kasus ini, Anda masih harus mengotomatiskan langkah failover, sehingga inisiasi manual akan seperti menekan tombol.

Ada beberapa opsi manajemen lalu lintas yang perlu dipertimbangkan saat menggunakan layanan AWS. Salah satu opsinya adalah menggunakan Amazon Route 53. Dengan menggunakan Amazon Route 53, Anda dapat mengaitkan beberapa titik akhir IP di satu Wilayah AWS atau lebih dengan nama domain Route 53. Untuk mengimplementasikan failover yang dimulai secara manual, Anda dapat menggunakan Pengontrol Pemulihan Aplikasi Amazon Route 53, yang memberikan API

bidang data dengan ketersediaan tinggi untuk merutekan kembali lalu lintas ke Wilayah pemulihan. Saat mengimplementasikan failover, gunakan operasi bidang data dan hindari bidang kendali yang dideskripsikan di REL11-BP04 Mengandalkan bidang data dan bukan bidang kendali selama pemulihan.

Untuk mempelajari selengkapnya tentang hal ini dan opsi lainnya, lihat bagian ini di Laporan Resmi Pemulihan Bencana.

6. Rancang rencana terkait bagaimana beban kerja akan failback.

Failback adalah saat Anda mengembalikan operasi beban kerja ke Wilayah utama, setelah bencana berakhir. Penyediaan infrastruktur dan kode untuk Wilayah utama umumnya mengikuti langkah yang sama yang digunakan saat memulai, dengan mengandalkan infrastruktur sebagai kode dan pipeline deployment kode. Tantangan failback adalah mengembalikan penyimpanan data, dan memastikan konsistensi dengan Wilayah pemulihan dalam operasi.

Dalam status failed over, basis data dalam Wilayah pemulihan bersifat waktu nyata dan memiliki data terbaru. Tujuannya adalah untuk menyinkronkan kembali dari Wilayah pemulihan ke Wilayah utama, memastikannya tetap terbaru.

Hal ini dilakukan secara otomatis untuk beberapa layanan AWS. Jika menggunakan tabel global Amazon DynamoDB, meskipun tabel di Wilayah utama menjadi tidak tersedia, saat kembali online, DynamoDB akan melanjutkan penulisan yang tertunda. Jika menggunakan Basis Data Global Amazon Aurora dan menggunakan failover terencana dan terkelola, maka topologi replikasi yang ada untuk basis data global Aurora dipertahankan. Dengan demikian, instans baca/tulis sebelumnya di Wilayah utama akan menjadi replika dan menerima pembaruan dari Wilayah pemulihan.

Dalam kasus saat ini tidak dibuat otomatis, Anda perlu menetapkan ulang basis data di Wilayah utama sebagai replika dari basis data di Wilayah pemulihan. Dalam banyak kasus, ini akan melibatkan penghapusan basis data utama yang lama dan membuat replika yang baru. Misalnya, untuk instruksi tentang cara melakukan ini dengan Basis Data Global Amazon Aurora yang mengasumsikan failover tak terencana, lihat lab ini: Fail Back Basis Data Global.

Setelah failover, jika Anda dapat tetap menjalankannya di Wilayah pemulihan, pertimbangkan untuk membuat ini menjadi Wilayah utama yang baru. Anda masih harus melakukan semua langkah di atas untuk membuat Wilayah utama sebelumnya menjadi Wilayah pemulihan. Beberapa organisasi melakukan rotasi terjadwal, menukar Wilayah utama dan pemulihan secara berkala (misalnya setiap tiga bulan).

Semua langkah yang diperlukan untuk failover dan failback harus diperiksa di buku pedoman yang tersedia untuk semua anggota tim dan ditinjau secara berkala.

Ketika menggunakan Elastic Disaster Recovery, layanan akan membantu mengatur dan mengotomatiskan proses failback. Untuk detail selengkapnya, lihat Melakukan failback.

Tingkat upaya untuk rencana implementasi: Tinggi

Sumber daya

Praktik Terbaik Terkait:

- the section called “REL09-BP01 Mengidentifikasi dan mencadangkan data yang perlu dicadangkan, atau memproduksi ulang data dari sumber”
- the section called “REL11-BP04 Mengandalkan bidang data dan bukan bidang kendali selama pemulihan”
- the section called “REL13-BP01 Tetapkan sasaran pemulihan untuk waktu henti dan kehilangan data”

Dokumen terkait:

- Blog Arsitektur AWS: Seri Pemulihan Bencana
- Pemulihan Bencana Beban Kerja di AWS: Pemulihan di Cloud (Laporan Resmi AWS)
- Opsi pemulihan bencana di cloud
- Bangun solusi backend aktif-aktif nirserver multi-wilayah dalam satu jam
- Backend nirserver multi-wilayah — dimuat ulang
- RDS: Mereplikasi Replika Baca di Seluruh Wilayah
- Route 53: Mengonfigurasi Failover DNS
- S3: Replika Lintas-Wilayah
- Apa Itu AWS Backup?
- Apa itu Pengontrol Pemulihan Aplikasi Route 53?
- AWS Elastic Disaster Recovery
- HashiCorp Terraform: Memulai - AWS
- Partner APN: partner yang dapat membantu pemulihan bencana
- AWS Marketplace: produk yang dapat digunakan untuk pemulihan bencana

Video terkait:

- Pemulihan Bencana Beban Kerja di AWS
- AWS re:Invent 2018: Pola Arsitektur untuk Aplikasi Aktif-Aktif Multi-Wilayah (ARC209-R2)
- Memulai AWS Elastic Disaster Recovery | Amazon Web Services

Contoh terkait:

- Well-Architected Lab - Pemulihan Bencana - Seri lokakarya yang mengilustrasikan strategi DR

REL13-BP03 Menguji implementasi pemulihan bencana untuk memvalidasi implementasi

Secara rutin uji failover ke situs pemulihan Anda untuk memastikan operasi yang baik dan RTO serta RPO terpenuhi.

Antipola umum:

- Tidak pernah melakukan failover di lingkungan produksi.

Manfaat menjalankan praktik terbaik ini: Pengujian rencana pemulihan bencana secara rutin memverifikasi bahwa rencana tersebut akan berfungsi saat diperlukan, dan tim Anda tahu cara menjalankan strategi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Pola untuk dihindari adalah mengembangkan jalur pemulihan yang sangat jarang dilakukan. Misalnya, Anda mungkin memiliki penyimpanan data sekunder yang digunakan untuk kueri hanya-baca. Saat Anda menulis ke penyimpanan data dan penyimpanan primer gagal, Anda mungkin ingin melakukan failover ke penyimpanan data sekunder. Jika Anda tidak sering menguji failover ini, Anda mungkin akan mendapati bahwa asumsi Anda tentang kemampuan penyimpanan data sekunder ternyata salah. Kapasitas sekunder, yang selama ini mungkin mencukupi saat terakhir Anda uji, mungkin sudah tidak mampu mentoleransi beban di bawah skenario ini. Pengalaman kami menunjukkan bahwa satu-satunya pemulihan kesalahan yang berfungsi adalah jalur yang Anda uji secara sering. Inilah alasan memiliki sedikit jalur pemulihan adalah yang terbaik. Anda dapat membuat pola pemulihan dan mengujinya secara rutin. Jika Anda memiliki jalur pemulihan yang kompleks atau kritis, Anda tetap perlu secara rutin melatih kegagalan tersebut dalam lingkungan

produksi agar Anda yakin bahwa jalur pemulihan tersebut berfungsi. Pada contoh yang baru saja kita bahas, Anda harus melakukan failover ke penyimpanan siaga secara rutin, terlepas ada tidaknya kebutuhan.

Langkah implementasi

1. Rekayasa beban kerja Anda untuk pemulihan. Uji jalur pemulihan Anda secara rutin. Komputasi yang berorientasi pada pemulihan mengidentifikasi karakteristik dalam sistem yang meningkatkan pemulihan: isolasi dan redundansi, kemampuan di seluruh sistem untuk membatalkan perubahan, kemampuan untuk memantau dan menentukan kondisi, kemampuan untuk menyediakan diagnostik, pemulihan otomatis, desain modular, dan kemampuan untuk memulai ulang. Latih jalur pemulihan untuk memverifikasi bahwa Anda dapat menyelesaikan pemulihan dalam waktu yang ditentukan ke status yang ditentukan. Gunakan runbook selama pemulihan ini untuk mendokumentasikan masalah dan menemukan solusinya sebelum pengujian berikutnya.
2. Untuk beban kerja berbasis Amazon EC2, gunakan AWS Elastic Disaster Recovery untuk mengimplementasikan dan meluncurkan instans latihan untuk strategi DR Anda. AWS Elastic Disaster Recovery menyediakan kemampuan untuk menjalankan latihan secara efisien, yang membantu Anda bersiap untuk peristiwa failover. Anda juga dapat sering-sering meluncurkan instans menggunakan Elastic Disaster Recovery untuk tujuan pengujian dan latihan tanpa mengarahkan ulang lalu lintas.

Sumber daya

Dokumen terkait:

- Partner APN: partner yang dapat membantu pemulihan bencana
- Blog Arsitektur AWS: Seri Pemulihan Bencana
- AWS Marketplace: produk yang dapat digunakan untuk pemulihan bencana
- AWS Elastic Disaster Recovery
- Pemulihan Bencana Beban Kerja di AWS: Pemulihan di Cloud (Laporan Resmi AWS)
- Bersiap untuk Failover AWS Elastic Disaster Recovery
- Proyek Berkeley/Stanford komputasi berorientasi pemulihan
- Apa itu Simulator Injeksi Kesalahan AWS?

Video terkait:

- AWS re:Invent 2018: Pola Arsitektur untuk Aplikasi Aktif-Aktif Multi-Wilayah
- AWS re:Invent 2019: Pencadangan dan pemulihan serta solusi pemulihan bencana dengan AWS

Contoh terkait:

- Well-Architected Lab - Pengujian Ketangguhan

REL13-BP04 Mengelola penyimpangan konfigurasi di lokasi atau Wilayah Pemulihan Bencana (DR)

Pastikan infrastruktur, data, dan konfigurasi diperlukan di lokasi atau Wilayah DR. Misalnya, periksa apakah AMI dan kuota layanan sudah mutakhir.

AWS Config terus memantau dan merekam konfigurasi sumber daya AWS Anda. Layanan ini dapat mendeteksi penyimpangan dan memicu AWS Systems Manager Automation untuk memperbaikinya dan memunculkan alarm. AWS CloudFormation juga dapat mendeteksi penyimpangan dalam tumpukan yang telah Anda deploy.

Antipola umum:

- Gagal melakukan pembaruan pada lokasi pemulihan Anda, saat Anda membuat perubahan konfigurasi atau infrastruktur pada lokasi primer.
- Tidak mempertimbangkan potensi pembatasan (seperti perbedaan layanan) di lokasi primer dan pemulihan Anda.

Manfaat menjalankan praktik terbaik ini: Lingkungan DR yang sesuai dengan lingkungan Anda saat ini menjamin pemulihan yang lengkap.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

- Pastikan pipeline pengiriman Anda menjangkau lokasi primer dan cadangan Anda. Pipeline pengiriman untuk men-deploy aplikasi ke lingkungan produksi harus menyebarkan ke semua lokasi strategi pemulihan bencana yang ditentukan, termasuk lingkungan pengembangan dan pengujian.
- Aktifkan AWS Config untuk melacak lokasi dengan potensi penyimpangan. Gunakan aturan AWS Config untuk membuat sistem yang menerapkan strategi pemulihan bencana Anda dan menghasilkan pemberitahuan saat mendeteksi penyimpangan.
  - Mengatasi Sumber Daya AWS yang Tidak Patuh dengan Aturan AWS Config

  - AWS Systems Manager Automation
- Gunakan AWS CloudFormation untuk men-deploy infrastruktur Anda. AWS CloudFormation dapat mendeteksi penyimpangan antara yang ditentukan oleh templat CloudFormation Anda dan apa yang sebenarnya di-deploy.
  - AWS CloudFormation: Mendeteksi Penyimpangan di Seluruh Tumpukan CloudFormation

Sumber daya

Dokumen terkait:

- Partner APN: partner yang dapat membantu pemulihan bencana
- Blog Arsitektur AWS: Seri Pemulihan Bencana
- AWS CloudFormation: Mendeteksi Penyimpangan di Seluruh Tumpukan CloudFormation
- AWS Marketplace: produk yang dapat digunakan untuk pemulihan bencana
- AWS Systems Manager Automation
- Pemulihan Bencana Beban Kerja di AWS: Pemulihan di Cloud (Laporan Resmi AWS)
- Bagaimana cara mengimplementasikan solusi Manajemen Konfigurasi Infrastruktur di AWS?
- Mengatasi Sumber Daya AWS yang Tidak Patuh dengan Aturan AWS Config

Video terkait:

- AWS re:Invent 2018: Pola Arsitektur untuk Aplikasi Multi-Wilayah Aktif-Aktif (ARC209-R2)

REL13-BP05 Mengotomatiskan pemulihan

Gunakan AWS atau alat pihak ketiga untuk mengotomatiskan pemulihan sistem dan merutekan lalu lintas ke situs DR atau Wilayah.

Berdasarkan pemeriksaan kondisi yang dikonfigurasi, layanan AWS, seperti Elastic Load Balancing dan AWS Auto Scaling, dapat mendistribusikan beban ke Zona Ketersediaan yang kondisinya baik, sedangkan layanan seperti Amazon Route 53 dan AWS Global Accelerator, dapat merutekan beban ke Wilayah AWS yang kondisinya baik. Pengontrol Pemulihan Aplikasi Amazon Route 53 membantu Anda mengelola dan mengoordinasikan failover menggunakan fitur pemeriksaan kesiapan dan kontrol perutean. Fitur tersebut terus memantau kemampuan aplikasi untuk pulih dari kegagalan, sehingga Anda dapat mengontrol pemulihan aplikasi di beberapa Wilayah AWS, Zona Ketersediaan, dan on-premise.

Untuk beban kerja yang ada di pusat data fisik atau virtual atau cloud pribadi, AWS Elastic Disaster Recovery, tersedia melalui AWS Marketplace, memungkinkan organisasi untuk mengatur strategi pemulihan bencana otomatis ke AWS. CloudEndure juga mendukung pemulihan bencana lintas Wilayah/lintas AZ di AWS.

Antipola umum:

- Mengimplementasikan failover dan failback otomatis yang serupa dapat menyebabkan flapping saat kesalahan terjadi.

Manfaat menerapkan praktik terbaik ini: Pemulihan otomatis mengurangi waktu pemulihan dengan menghilangkan peluang untuk kesalahan manual.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

- Otomatiskan jalur pemulihan. Untuk pemulihan pendek, tindakan dan penilaian manusia tidak dapat digunakan untuk skenario ketersediaan tinggi. Sistem harus pulih secara otomatis dalam setiap situasi.
  - Gunakan CloudEndure Disaster Recovery untuk Failback dan Failover otomatis. CloudEndure Disaster Recovery terus mereplikasi mesin (termasuk sistem operasi, konfigurasi status sistem, basis data, aplikasi, dan file) ke dalam area penahapan rendah biaya di Akun AWS target dan Wilayah utama. Dalam kasus bencana, Anda dapat menginstruksikan CloudEndure Disaster Recovery untuk meluncurkan mesin dalam status yang tersedia sepenuhnya dalam hitungan menit secara otomatis.
    - Menjalankan Failover dan Failback Pemulihan Bencana
    - CloudEndure Disaster Recovery

Sumber daya

Dokumen terkait:

- Partner APN: partner yang dapat membantu pemulihan bencana
- Blog Arsitektur AWS: Seri Pemulihan Bencana
- AWS Marketplace: produk yang dapat digunakan untuk pemulihan bencana
- AWS Systems Manager Automation

- CloudEndure Disaster Recovery ke AWS
- Pemulihan Bencana Beban Kerja di AWS: Pemulihan di Cloud (Laporan Resmi AWS)

Video terkait:

- AWS re:Invent 2018: Pola Arsitektur untuk Aplikasi Aktif-Aktif Multi-Wilayah (ARC209-R2)

Efisiensi kinerja

Pilar Efisiensi Kinerja menyertakan kemampuan untuk menggunakan sumber daya komputasi dengan efisien agar memenuhi persyaratan sistem, dan untuk memelihara efisiensi tersebut seiring dengan perubahan permintaan dan perkembangan teknologi. Anda dapat menemukan panduan preskriptif tentang implementasi di laporan resmi Pilar Efisiensi Kinerja.

Area praktik terbaik

- Pemilihan arsitektur
- Komputasi dan perangkat keras
- Manajemen data
- Jaringan dan pengiriman konten
- Proses dan budaya

Pemilihan arsitektur

Pertanyaan

- PERF 1. Bagaimana cara memilih sumber daya dan arsitektur cloud yang sesuai untuk beban kerja Anda?

PERF 1. Bagaimana cara memilih sumber daya dan arsitektur cloud yang sesuai untuk beban kerja Anda?

Solusi yang optimal bervariasi untuk beban kerja tertentu, dan solusi sering kali menggabungkan beberapa pendekatan. Beban kerja yang dirancang dengan baik menggunakan beberapa solusi dan memungkinkan berbagai fitur guna meningkatkan kinerja.

Praktik terbaik

- PERF01-BP01 Mempelajari dan memahami layanan serta fitur cloud yang tersedia
- PERF01-BP02 Menggunakan panduan dari penyedia cloud Anda atau mitra yang tepat untuk mempelajari pola arsitektur dan praktik terbaik
- PERF01-BP03 Mempertimbangkan biaya dalam keputusan arsitektur
- PERF01-BP04 Mengevaluasi bagaimana kompromi berdampak pada pelanggan dan efisiensi arsitektur
- PERF01-BP05 Menggunakan kebijakan dan arsitektur referensi
- PERF01-BP06 Menggunakan tolok ukur untuk mendorong keputusan arsitektur
- PERF01-BP07 Menggunakan pendekatan berbasis data untuk pilihan arsitektur

PERF01-BP01 Mempelajari dan memahami layanan serta fitur cloud yang tersedia

Terus pelajari dan temukan layanan serta konfigurasi yang tersedia yang membantu Anda mengambil keputusan arsitektur yang lebih baik dan meningkatkan efisiensi kinerja dalam arsitektur beban kerja Anda.

Antipola umum:

- Anda menggunakan cloud sebagai pusat data kolokasi.
- Anda tidak memodernisasi aplikasi Anda setelah migrasi ke cloud.
- Anda hanya menggunakan satu tipe penyimpanan untuk semua hal yang perlu dipertahankan.
- Anda menggunakan tipe instans yang paling sesuai dengan standar Anda saat ini, tetapi lebih besar dari yang diperlukan.
- Anda melakukan deployment dan mengelola teknologi yang tersedia sebagai layanan terkelola.

Manfaat menjalankan praktik terbaik ini: Dengan mempertimbangkan layanan dan konfigurasi baru, Anda mungkin dapat meningkatkan kinerja, mengurangi biaya, dan mengoptimalkan upaya yang diperlukan untuk memelihara beban kerja Anda. Ini juga dapat membantu Anda mempercepat waktu perolehan nilai untuk produk yang didukung cloud.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

AWS terus-menerus merilis layanan dan fitur baru yang dapat meningkatkan kinerja dan mengurangi biaya beban kerja cloud. Mengikuti perkembangan layanan dan fitur baru ini sangat penting

untuk menjaga efisiensi kinerja di cloud. Modernisasi arsitektur beban kerja juga membantu Anda mempercepat produktivitas, mendorong inovasi, dan membuka lebih banyak peluang pertumbuhan.

Langkah implementasi

- Buat inventaris arsitektur dan perangkat lunak beban kerja untuk layanan terkait. Tentukan kategori produk mana yang akan dipelajari lebih lanjut.
- Jelajahi penawaran AWS untuk mengidentifikasi dan mempelajari layanan serta opsi konfigurasi yang relevan yang dapat membantu Anda meningkatkan kinerja dan mengurangi biaya serta kompleksitas operasional.
  - Apa yang Baru dengan AWS?
  - Blog AWS
  - AWS Skill Builder
  - Acara dan Webinar AWS
  - AWS Training and Certifications
  - Saluran YouTube AWS
  - Lokakarya AWS
  - Komunitas AWS
- Gunakan lingkungan sandbox (non-produksi) untuk mempelajari dan bereksperimen dengan layanan baru tanpa dikenakan biaya tambahan.

Sumber daya

Dokumen terkait:

- Pusat Arsitektur AWS
- AWS Partner Network
- Pustaka Solusi AWS
- Pusat Pengetahuan AWS
- Membangun aplikasi modern di AWS

Video terkait:

- Ini Arsitektur saya

Contoh terkait:

- Sampel AWS
- Contoh SDK AWS

PERF01-BP02 Menggunakan panduan dari penyedia cloud Anda atau mitra yang tepat untuk mempelajari pola arsitektur dan praktik terbaik

Gunakan sumber daya perusahaan cloud, seperti dokumentasi, arsitek solusi, layanan profesional, atau partner yang tepat untuk memandu keputusan arsitektur Anda. Semua sumber daya ini membantu meninjau dan meningkatkan arsitektur Anda untuk kinerja yang optimal.

Antipola umum:

- Anda menggunakan AWS sebagai penyedia cloud umum.
- Anda menggunakan layanan AWS dengan cara yang tidak sesuai dengan tujuan desainnya.
- Anda mengikuti semua panduan tanpa mempertimbangkan konteks bisnis Anda.

Manfaat menjalankan praktik terbaik ini: Menggunakan panduan dari penyedia cloud atau partner yang tepat dapat membantu Anda membuat pilihan arsitektur yang tepat untuk beban kerja Anda dan memberi Anda kepercayaan diri dalam keputusan Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

AWS menawarkan berbagai panduan, dokumentasi, dan sumber daya yang dapat membantu Anda membangun dan mengelola beban kerja cloud yang efisien. Dokumentasi AWS menyediakan contoh kode, tutorial, dan penjelasan layanan yang mendetail. Selain dokumentasi, AWS menyediakan program pelatihan dan sertifikasi, arsitek solusi, dan layanan profesional yang dapat membantu pelanggan menjelajahi berbagai aspek layanan cloud dan menerapkan arsitektur cloud yang efisien di AWS.

Manfaatkan semua sumber daya ini untuk mendapatkan wawasan tentang pengetahuan dan praktik terbaik yang berharga, menghemat waktu, dan mencapai hasil yang lebih baik di AWS Cloud.

Langkah implementasi

- Tinjau dokumentasi serta panduan AWS dan ikuti praktik terbaik. Semua sumber daya ini dapat membantu Anda memilih dan mengonfigurasi layanan secara efektif dan mencapai kinerja yang lebih baik.
  - Dokumentasi AWS (seperti panduan pengguna dan laporan resmi)
  - Blog AWS
  - AWS Training and Certifications
  - Saluran YouTube AWS
- Bergabunglah dengan acara partner AWS (seperti AWS Global Summits, AWS Re:Invent, grup pengguna, dan lokakarya) untuk belajar dari para ahli AWS tentang praktik terbaik untuk menggunakan layanan AWS.
  - Acara dan Webinar AWS
  - Lokakarya AWS
  - Komunitas AWS
- Hubungi AWS untuk mendapatkan bantuan saat Anda memerlukan panduan tambahan atau informasi produk. Arsitek Solusi AWS dan Layanan Profesional AWS menyediakan panduan untuk implementasi solusi. Partner AWS menyediakan keahlian AWS untuk membantu Anda menghadirkan ketangkasan dan inovasi untuk bisnis Anda.
- Gunakan Dukungan jika Anda membutuhkan dukungan teknis untuk menggunakan layanan secara efektif. Rencana Dukungan kami dirancang untuk memberi Anda perpaduan alat yang tepat dan akses ke keahlian sehingga Anda dapat berhasil dengan AWS sambil mengoptimalkan kinerja, mengelola risiko, dan menjaga biaya tetap terkendali.

Sumber daya

Dokumen terkait:

- Pusat Arsitektur AWS
- Pustaka Solusi AWS
- Pusat Pengetahuan AWS
- AWS Enterprise Support

Video terkait:

- Ini Arsitektur saya

Contoh terkait:

- Sampel AWS
- Contoh SDK AWS

PERF01-BP03 Mempertimbangkan biaya dalam keputusan arsitektur

Pertimbangkan biaya dalam keputusan arsitektur Anda untuk meningkatkan pemanfaatan sumber daya dan efisiensi kinerja beban kerja cloud Anda. Ketika Anda menyadari implikasi biaya dari beban kerja cloud Anda, Anda kemungkinan akan memanfaatkan sumber daya yang efisien dan mengurangi praktik pemborosan.

Antipola umum:

- Anda hanya menggunakan satu kelompok instans.
- Anda tidak mengevaluasi solusi berlisensi dibandingkan dengan solusi sumber terbuka.
- Anda tidak menentukan kebijakan siklus hidup penyimpanan.
- Anda tidak meninjau layanan dan fitur baru dari AWS Cloud.
- Anda hanya menggunakan penyimpanan blok.

Manfaat menjalankan praktik terbaik ini: Dengan mempertimbangkan biaya dalam pengambilan keputusan, Anda dapat menggunakan sumber daya yang lebih efisien dan mengeksplorasi investasi lainnya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Mengoptimalkan beban kerja untuk biaya dapat meningkatkan pemanfaatan sumber daya dan menghindari pemborosan dalam beban kerja cloud. Mempertimbangkan biaya dalam keputusan arsitektur biasanya mencakup penyesuaian ukuran komponen beban kerja dan menghadirkan elastisitas, yang menghasilkan peningkatan efisiensi kinerja beban kerja cloud.

Langkah implementasi

- Tetapkan sasaran biaya seperti batas anggaran untuk beban kerja cloud Anda.

- Identifikasi komponen utama (seperti instans dan penyimpanan) yang menambah biaya beban kerja Anda. Anda dapat menggunakan AWS Kalkulator Harga dan AWS Cost Explorer untuk mengidentifikasi pendorong biaya utama dalam beban kerja Anda.

- Gunakan Praktik terbaik pengoptimalan biaya Well-Architected untuk mengoptimalkan komponen kunci ini untuk biaya.
- Teruslah memantau dan menganalisis biaya untuk mengidentifikasi peluang pengoptimalan biaya dalam beban kerja Anda.
  - Gunakan AWS Budgets untuk mendapatkan pemberitahuan adanya biaya yang tidak dapat diterima.
  - Gunakan AWS Compute Optimizer atau AWS Trusted Advisor untuk mendapatkan rekomendasi pengoptimalan biaya.
  - Gunakan AWS Cost Anomaly Detection untuk mendapatkan deteksi anomali biaya dan analisis akar masalah secara otomatis.

Sumber daya

Dokumen terkait:

- Tinjauan Mendetail tentang Dasbor Inteligensi Biaya
- Pusat Arsitektur AWS
- AWS Partner Network
- Pustaka Solusi AWS
- Pusat Pengetahuan AWS

Video terkait:

- Ini Arsitektur saya
- Mengoptimalkan performa dan biaya untuk komputasi AWS Anda

Contoh terkait:

- Sampel AWS
- Contoh SDK AWS

- Betulkan ukuran dengan pengaktifan penggunaan Memori dan Compute Optimizer
- Kode AWS Compute Optimizer Demo

PERF01-BP04 Mengevaluasi bagaimana kompromi berdampak pada pelanggan dan efisiensi arsitektur

Saat mengevaluasi peningkatan terkait kinerja, tentukan pilihan mana yang berdampak pada efisiensi beban kerja dan pelanggan Anda. Misalnya, jika menggunakan penyimpanan data nilai-kunci dapat meningkatkan kinerja sistem, penting untuk mengevaluasi bagaimana dampak sifat eventual consistency-nya nanti terhadap pelanggan.

Antipola umum:

- Anda berasumsi bahwa semua kinerja yang dimiliki harus diimplementasikan, meskipun ada kompromi untuk implementasi.
- Anda hanya mengevaluasi perubahan beban kerja ketika masalah kinerja telah mencapai titik kritis.

Manfaat menjalankan praktik terbaik ini: Ketika Anda mengevaluasi potensi peningkatan terkait performa, Anda harus menentukan apakah kompromi untuk perubahan dapat diterima dengan persyaratan beban kerja. Dalam beberapa kasus, Anda mungkin harus mengimplementasikan beberapa kontrol tambahan untuk mengimbangi kompensasi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

Identifikasi area kritis dalam arsitektur Anda dalam hal dampak terhadap kinerja dan pelanggan. Tentukan cara Anda mewujudkan peningkatan, kompromi seperti apa yang ditimbulkan peningkatan, serta bagaimana pengaruhnya terhadap sistem dan pengalaman pengguna. Misalnya, mengimplementasikan pembuatan cache data dapat membantu meningkatkan kinerja secara signifikan tetapi memerlukan strategi yang jelas terkait cara dan waktu untuk memperbarui atau menonaktifkan data yang di-cache guna mencegah perilaku sistem yang tidak sesuai.

Langkah implementasi

- Pahami persyaratan beban kerja dan SLA Anda.
- Tentukan faktor evaluasi secara jelas. Faktor-faktor mungkin berhubungan dengan biaya, keandalan, keamanan, dan kinerja beban kerja Anda.

- Pilih arsitektur dan layanan yang dapat memenuhi kebutuhan Anda.
- Lakukan eksperimen dan bukti konsep (POC) untuk mengevaluasi faktor kompromi dan dampak terhadap pelanggan dan efisiensi arsitektur. Biasanya, beban kerja dengan ketersediaan tinggi, berkinerja tinggi, dan aman mengonsumsi lebih banyak sumber daya cloud sekaligus memberikan pengalaman pelanggan yang lebih baik.

Sumber daya

Dokumen terkait:

- Amazon Builders' Library
- KPI Quick
- Amazon CloudWatch RUM
- Dokumentasi X-Ray
- Memahami pola ketahanan dan kompromi untuk merancang secara efisien di cloud

Video terkait:

- Buat Rencana Pemantauan
- Optimalkan aplikasi dengan Amazon CloudWatch RUM
- Demo Amazon CloudWatch Synthetics

Contoh terkait:

- Ukur waktu pemuatan halaman dengan Amazon CloudWatch Synthetics
- Klien Web Amazon CloudWatch RUM

PERF01-BP05 Menggunakan kebijakan dan arsitektur referensi

Gunakan kebijakan internal dan arsitektur referensi yang ada saat memilih layanan dan konfigurasi agar lebih efisien saat merancang dan mengimplementasikan beban kerja Anda.

Antipola umum:

- Anda mengizinkan berbagai macam teknologi yang berdampak pada biaya manajemen biaya perusahaan.

Manfaat menjalankan praktik terbaik ini: Dengan menetapkan kebijakan untuk pilihan arsitektur, teknologi, dan vendor, keputusan dapat diambil dengan lebih cepat.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Adanya kebijakan internal dalam memilih sumber daya dan arsitektur memberikan standar dan pedoman untuk diikuti ketika membuat pilihan arsitektur. Pedoman tersebut merampingkan proses pengambilan keputusan saat memilih layanan cloud yang tepat dan dapat membantu meningkatkan efisiensi kinerja. Lakukan deployment beban kerja Anda menggunakan arsitektur referensi atau kebijakan. Integrasikan layanan ke dalam deployment cloud, lalu gunakan pengujian kinerja untuk memastikan bahwa Anda dapat terus memenuhi persyaratan kinerja.

Langkah implementasi

- Pahami dengan jelas persyaratan beban kerja cloud Anda.
- Tinjau kebijakan internal dan eksternal untuk mengidentifikasi kebijakan yang paling relevan.
- Gunakan arsitektur referensi yang sesuai yang disediakan oleh AWS atau praktik terbaik industri Anda.
- Buat rangkaian yang terdiri dari kebijakan, standar, arsitektur referensi, dan pedoman preskriptif untuk situasi umum. Tindakan tersebut memungkinkan tim Anda bergerak lebih cepat. Sesuaikan aset untuk bidang Anda jika perlu.
- Validasi kebijakan dan arsitektur referensi ini untuk beban kerja Anda di lingkungan sandbox.
- Terus ikuti perkembangan standar industri dan pembaruan AWS untuk memastikan kebijakan dan arsitektur referensi Anda membantu mengoptimalkan beban kerja cloud Anda.

Sumber daya

Dokumen terkait:

- Pusat Arsitektur AWS
- AWS Partner Network
- Pustaka Solusi AWS
- Pusat Pengetahuan AWS

Video terkait:

- Ini Arsitektur saya

Contoh terkait:

- Sampel AWS
- Contoh SDK AWS

PERF01-BP06 Menggunakan tolok ukur untuk mendorong keputusan arsitektur

Lakukan tolok ukur pada kinerja beban kerja yang ada untuk memahami kinerjanya di cloud dan mendorong keputusan arsitektur berdasarkan data tersebut.

Antipola umum:

- Anda mengandalkan tolok ukur umum yang tidak mewakili karakteristik beban kerja Anda.
- Anda bergantung pada persepsi dan tanggapan pelanggan sebagai satu-satunya tolok ukur.

Manfaat menjalankan praktik terbaik ini: Melalui tolok ukur implementasi Anda saat ini, Anda dapat mengukur peningkatan performa.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Gunakan benchmarking dengan pengujian sintetis untuk menilai kinerja komponen beban kerja Anda. Benchmarking umumnya dapat disiapkan dengan lebih cepat daripada pengujian beban dan digunakan untuk mengevaluasi teknologi untuk komponen tertentu. Benchmarking sering digunakan pada awal proyek baru, saat Anda tidak memiliki solusi lengkap untuk memuat pengujian.

Anda dapat merancang pengujian tolok ukur kustom atau menggunakan pengujian standar industri, misalnya TPC-DS, untuk melakukan tolok ukur beban kerja Anda. Tolok ukur industri sangat membantu saat memperbandingkan lingkungan. Tolok ukur kustom bermanfaat untuk menargetkan jenis operasi tertentu yang ingin dibuat dalam arsitektur.

Saat melakukan tolok ukur, penting untuk menyiapkan lingkungan terlebih dahulu untuk memastikan hasil yang valid. Jalankan tolok ukur yang sama beberapa kali untuk memastikan Anda memperoleh variasi apa pun dari waktu ke waktu.

Karena tolok ukur umumnya lebih cepat untuk menjalankan pengujian daripada memuatnya, maka tolok ukur dapat digunakan terlebih dahulu dalam deployment pipeline dan memberikan umpan balik pada deviasi kinerja. Saat Anda mengevaluasi perubahan yang signifikan dalam komponen atau layanan, tolok ukur dapat menjadi cara cepat guna menentukan apakah perubahan memang perlu dibuat. Menggunakan benchmarking bersama dengan pengujian beban begitu penting karena pengujian beban memberi tahu Anda tentang bagaimana kinerja beban kerja Anda dalam produksi.

Langkah implementasi

- Tentukan metrik (seperti pemanfaatan CPU, latensi, atau throughput) untuk mengevaluasi kinerja beban kerja Anda.
- Identifikasi dan siapkan alat tolok ukur yang sesuai dengan beban kerja Anda. Anda dapat menggunakan layanan AWS (seperti Amazon CloudWatch) atau alat pihak ketiga yang kompatibel dengan beban kerja Anda.
- Lakukan pengujian tolok ukur Anda dan pantau metrik selama pengujian.
- Analisis dan dokumentasikan hasil tolok ukur untuk mengidentifikasi setiap kemacetan dan masalah.
- Gunakan hasil pengujian untuk mengambil keputusan arsitektur dan menyesuaikan beban kerja Anda. Termasuk di dalamnya mungkin adalah mengubah layanan atau mengadopsi fitur baru.
- Uji ulang beban kerja Anda setelah penyesuaian.

Sumber daya

Dokumen terkait:

- Pusat Arsitektur AWS
- AWS Partner Network
- Pustaka Solusi AWS
- Pusat Pengetahuan AWS
- RUM Amazon CloudWatch
- Amazon CloudWatch Synthetics

Video terkait:

- Ini Arsitektur saya
- Optimalkan aplikasi dengan Amazon CloudWatch RUM

- Demo Amazon CloudWatch Synthetics

Contoh terkait:

- Sampel AWS
- Contoh SDK AWS
- Pengujian Beban Terdistribusi
- Ukur waktu pemuatan halaman dengan Amazon CloudWatch Synthetics
- Klien Web Amazon CloudWatch RUM

PERF01-BP07 Menggunakan pendekatan berbasis data untuk pilihan arsitektur

Tentukan pendekatan yang jelas dan berbasis data untuk pilihan arsitektur guna memastikan layanan dan konfigurasi cloud yang tepat digunakan untuk memenuhi kebutuhan bisnis spesifik Anda.

Antipola umum:

- Anda berasumsi bahwa arsitektur Anda saat ini statis dan tidak perlu diperbarui dari waktu ke waktu.
- Pilihan arsitektur Anda didasarkan pada tebakan dan asumsi.
- Anda memperkenalkan perubahan arsitektur seiring waktu tanpa justifikasi.

Manfaat menjalankan praktik terbaik ini: Dengan memiliki pendekatan yang terdefinisi dengan baik dalam membuat pilihan arsitektur, Anda menggunakan data untuk memengaruhi desain beban kerja Anda dan mengambil keputusan berdasarkan informasi dari waktu ke waktu.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Gunakan pengalaman internal dan pengetahuan tentang cloud, atau sumber daya eksternal seperti kasus penggunaan yang dipublikasi atau laporan resmi untuk memilih sumber daya dan layanan di arsitektur Anda. Anda harus memiliki proses yang terdefinisi dengan baik yang mendorong eksperimen dan tolok ukur dengan layanan yang bisa digunakan pada beban kerja Anda.

Backlog untuk beban kerja kritis tidak boleh hanya terdiri dari cerita pengguna yang memberikan fungsionalitas yang relevan dengan bisnis dan pengguna, melainkan juga harus berisi cerita teknis

yang membentuk landasan arsitektur untuk beban kerja. Landasan ini didasarkan pada kemajuan teknologi baru serta layanan baru dan mengadopsinya berdasarkan data dan pembenaran yang tepat. Hal ini memastikan bahwa arsitektur tetap relevan di masa depan dan tidak jalan di tempat.

Langkah implementasi

- Lakukan interaksi dengan pemangku kepentingan utama untuk menentukan persyaratan beban kerja, termasuk kinerja, ketersediaan, dan pertimbangan biaya. Pertimbangkan faktor-faktor seperti jumlah pengguna dan pola penggunaan untuk beban kerja Anda.
- Ciptakan landasan arsitektur atau backlog teknologi yang diprioritaskan bersamaan dengan backlog fungsional.
- Evaluasi dan nilai berbagai layanan cloud (untuk detail selengkapnya, lihat PERF01-BP01 Mempelajari dan memahami layanan serta fitur cloud yang tersedia).
- Jelajahi pola-pola arsitektur yang berbeda, seperti layanan mikro atau nirserver, yang memenuhi persyaratan kinerja Anda (untuk detail selengkapnya, lihat PERF01-BP02 Menggunakan panduan dari penyedia cloud Anda atau mitra yang tepat untuk mempelajari pola arsitektur dan praktik terbaik).

- Belajar dari tim lain, diagram arsitektur, dan sumber daya seperti Arsitek Solusi AWS, Pusat Arsitektur AWS, dan AWS Partner Network, untuk membantu Anda memilih arsitektur yang tepat untuk beban kerja Anda.

- Tentukan metrik kinerja seperti throughput dan waktu respons yang dapat membantu Anda mengevaluasi kinerja beban kerja Anda.
- Lakukan eksperimen dan gunakan metrik yang ditentukan untuk memvalidasi kinerja arsitektur yang dipilih.
- Teruslah memantau dan melakukan penyesuaian sesuai kebutuhan untuk mempertahankan kinerja optimal arsitektur Anda.
- Dokumentasikan arsitektur dan keputusan pilihan Anda sebagai referensi untuk pembaruan dan pembelajaran di masa mendatang.
- Teruslah meninjau dan memperbarui pendekatan pemilihan arsitektur berdasarkan pembelajaran, teknologi baru, dan metrik yang menunjukkan kebutuhan perubahan atau masalah dalam pendekatan saat ini.

Sumber daya

Dokumen terkait:

- Pustaka Solusi AWS
- Pusat Pengetahuan AWS

Video terkait:

- Ini Arsitektur saya

Contoh terkait:

- Sampel AWS
- Contoh SDK AWS

Komputasi dan perangkat keras

PERF 2. Bagaimana cara memilih dan menggunakan sumber daya komputasi dalam beban kerja Anda?

Pilihan komputasi yang optimal untuk beban kerja tertentu bervariasi berdasarkan desain aplikasi, pola penggunaan, dan pengaturan konfigurasi. Arsitektur dapat menggunakan pilihan komputasi yang berbeda untuk berbagai komponen, dan memungkinkan fitur yang berbeda untuk meningkatkan kinerja. Memilih pilihan komputasi yang salah untuk arsitektur dapat menyebabkan efisiensi kinerja menjadi lebih rendah.

Praktik terbaik

- PERF02-BP01 Memilih opsi komputasi terbaik untuk beban kerja Anda
- PERF02-BP02 Memahami konfigurasi dan fitur komputasi yang tersedia
- PERF02-BP03 Mengumpulkan komputasi metrik terkait
- PERF02-BP04 Mengonfigurasi dan menyesuaikan ukuran sumber daya komputasi
- PERF02-BP05 Menskalakan sumber daya komputasi Anda secara dinamis
- PERF02-BP06 Menggunakan akselerator komputasi berbasis perangkat keras yang dioptimalkan

PERF02-BP01 Memilih opsi komputasi terbaik untuk beban kerja Anda

Dengan memilih opsi komputasi yang paling tepat untuk beban kerja, Anda dapat meningkatkan kinerja, mengurangi biaya infrastruktur yang tidak perlu, dan menurunkan upaya operasional yang diperlukan untuk memelihara beban kerja Anda.

Antipola umum:

- Anda menggunakan opsi komputasi yang sama yang digunakan secara on-premise.
- Anda tidak mengetahui opsi, fitur, dan solusi komputasi cloud, dan bagaimana solusi tersebut dapat meningkatkan kinerja komputasi Anda.
- Anda melakukan pengadaan opsi komputasi yang berlebihan untuk memenuhi persyaratan penskalaan atau kinerja ketika ada opsi komputasi lain yang lebih sesuai dengan karakteristik beban kerja Anda.

Manfaat menjalankan praktik terbaik ini: Dengan mengidentifikasi persyaratan komputasi dan mengevaluasi opsi-opsi yang tersedia, Anda dapat membuat beban kerja Anda lebih hemat sumber daya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Untuk mengoptimalkan beban kerja cloud Anda demi efisiensi kinerja, penting untuk memilih opsi komputasi yang paling sesuai dengan kasus penggunaan dan persyaratan kinerja Anda. AWS menyediakan berbagai opsi komputasi yang sesuai untuk berbagai beban kerja di cloud. Misalnya, Anda dapat menggunakan Amazon EC2 untuk meluncurkan dan mengelola server virtual, AWS Lambda untuk menjalankan kode tanpa harus menyediakan atau mengelola server, Amazon ECS atau Amazon EKS untuk menjalankan dan mengelola kontainer, atau AWS Batch untuk memproses data dalam volume besar secara paralel. Berdasarkan skala dan kebutuhan komputasi Anda, Anda harus memilih dan mengonfigurasi solusi komputasi yang optimal untuk situasi Anda. Anda juga dapat mempertimbangkan untuk menggunakan beberapa jenis solusi komputasi dalam satu beban kerja, karena masing-masing memiliki kelebihan dan kekurangannya sendiri.

Langkah-langkah berikut ini memandu Anda dalam memilih opsi komputasi yang tepat agar sesuai dengan karakteristik beban kerja dan persyaratan kinerja Anda.

Langkah implementasi

1. Pahami persyaratan komputasi beban kerja Anda. Persyaratan utama yang harus dipertimbangkan antara lain kebutuhan pemrosesan, pola lalu lintas, pola akses data, kebutuhan penskalaan, dan persyaratan latensi.
2. Pelajari berbagai opsi komputasi yang tersedia untuk beban kerja Anda di AWS (seperti yang diuraikan dalam PERF01-BP01 Mempelajari dan memahami layanan serta fitur cloud yang tersedia. Berikut adalah beberapa opsi komputasi utama, karakteristiknya, dan kasus penggunaan umumnya:

Layanan AWS	Karakteristik utama	Kasus penggunaan umum
Amazon Elastic Compute Cloud (Amazon EC2)	Memiliki opsi khusus untuk perangkat keras, persyaratan lisensi, banyak pilihan family instans yang berbeda, jenis prosesor, dan akselerator komputasi	Migrasi angkat dan geser, aplikasi monolitik, lingkungan hybrid, aplikasi perusahaan
Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS)	Deployment mudah, lingkungan konsisten, layanan mikro	dapat diskalakan, lingkungan hybrid
AWS Lambda	Layanan komputasi nirserver yang menjalankan kode sebagai respons terhadap peristiwa dan secara otomatis mengelola sumber daya komputasi yang mendasari nya.	Layanan mikro, aplikasi yang didorong peristiwa
AWS Batch	Menyediakan dan menskalakan secara efisien dan dinamis Amazon Elastic Container Service (Amazon ECS), Amazon Elastic	HPC, melatih model ML

Layanan AWS	Karakteristik utama	Kasus penggunaan umum
	Kubernetes Service (Amazon EKS), dan AWS Fargate sumber daya komputasi, dengan opsi untuk menggunakan Instans Sesuai Permintaan atau Spot berdasarkan kebutuhan pekerjaan Anda	
Amazon Lightsail	Aplikasi Linux dan Windows yang telah dikonfigurasi sebelumnya untuk menjalank an beban kerja kecil	Aplikasi web sederhana, situs web kustom

3. Lakukan evaluasi biaya (seperti biaya per jam atau transfer data) dan overhead manajemen (seperti patching dan penskalaan) yang terkait dengan setiap opsi komputasi.
4. Lakukan uji coba dan uji tolok ukur di lingkungan nonproduksi untuk mengidentifikasi opsi komputasi mana yang paling sesuai dengan kebutuhan beban kerja Anda.
5. Setelah menguji coba dan mengidentifikasi solusi komputasi baru Anda, rencanakan migrasi dan validasikan metrik kinerja Anda.
6. Gunakan alat pemantauan AWS seperti Amazon CloudWatch dan layanan optimasi seperti AWS Compute Optimizer untuk terus mengoptimalkan sumber daya komputasi Anda berdasarkan pola penggunaan dunia nyata.

Sumber daya

Dokumen terkait:

- Komputasi Cloud dengan AWS
- Tipe Instans Amazon EC2
- Kontainer Amazon EKS: Simpul Pekerja Amazon EKS
- Kontainer Amazon ECS: Instans Kontainer Amazon ECS
- Fungsi: Konfigurasi Fungsi Lambda
- Panduan Preskriptif untuk Kontainer

- Panduan Preskriptif untuk Nirserver

Video terkait:

- How to choose compute option for startups
- Mengoptimalkan performa dan biaya untuk komputasi AWS Anda
- Fondasi Amazon EC2
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- Melakukan deployment model ML untuk inferensi dengan performa tinggi dan biaya rendah
- Komputasi yang lebih baik, lebih cepat, dan lebih murah: Optimisasi biaya Amazon EC2

Contoh terkait:

- Memigrasikan Aplikasi web ke kontainer
- Jalankan Hello World Nirserver

PERF02-BP02 Memahami konfigurasi dan fitur komputasi yang tersedia

Pahami opsi dan fitur konfigurasi yang tersedia bagi layanan komputasi Anda untuk membantu Anda menyediakan jumlah sumber daya yang tepat dan meningkatkan efisiensi kinerja.

Antipola umum:

- Anda tidak mengevaluasi opsi komputasi atau family instans yang tersedia berdasarkan karakteristik beban kerja.
- Anda menyediakan sumber daya komputasi secara berlebihan untuk memenuhi persyaratan permintaan puncak.

Manfaat menjalankan praktik terbaik ini: Pahami fitur dan konfigurasi komputasi AWS sehingga Anda dapat menggunakan solusi komputasi yang dioptimalkan untuk memenuhi karakteristik dan kebutuhan beban kerja Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Setiap solusi komputasi memiliki konfigurasi dan fitur unik yang tersedia untuk mendukung berbagai karakteristik dan persyaratan beban kerja. Pelajari bagaimana opsi-opsi tersebut melengkapi beban kerja Anda, dan tentukan opsi konfigurasi yang terbaik untuk aplikasi Anda. Contoh dari opsi tersebut meliputi family instans, ukuran, fitur (GPU, I/O), lonjakan, waktu habis, ukuran fungsi, instans kontainer, dan konkurensi. Jika beban kerja Anda telah menggunakan opsi komputasi yang sama selama lebih dari empat pekan dan Anda mengantisipasi bahwa karakteristiknya akan tetap sama di masa depan, Anda dapat menggunakan AWS Compute Optimizer untuk mengetahui apakah opsi komputasi Anda saat ini cocok untuk beban kerja dari perspektif CPU dan memori.

Langkah implementasi

1. Pahami persyaratan beban kerja (seperti kebutuhan CPU, memori, dan latensi).
2. Tinjau dokumentasi dan praktik terbaik AWS untuk mempelajari rekomendasi opsi konfigurasi yang dapat membantu meningkatkan kinerja komputasi. Berikut adalah beberapa opsi konfigurasi utama yang perlu dipertimbangkan:

Opsi konfigurasi	Contoh
Jenis instans	• Instans komputasi yang dioptimalkan ideal untuk beban kerja yang membutuhkan rasio vCPU terhadap memori yang lebih tinggi. • Instans memori yang dioptimalkan mengirimkan sejumlah besar memori untuk mendukung beban kerja intensif memori. • Instans penyimpanan yang dioptimalkan didesain untuk beban kerja yang memerlukan akses baca dan tulis sekuensial (IOPS) yang tinggi ke penyimpanan lokal.
Model harga	• Instans Sesuai Permintaan memungkinkan Anda menggunakan kapasitas komputasi per jam atau per detik tanpa komitmen

Opsi konfigurasi	Contoh
	jangka panjang. Instans ini bagus untuk lonjakan di atas kebutuhan dasar kinerja. • Savings Plans menawarkan penghemat an yang signifikan atas Instans Sesuai Permintaan dengan komitmen untuk menggunakan daya komputasi dalam jumlah tertentu selama jangka waktu satu atau tiga tahun. • Instans Spot memungkinkan Anda memanfaatkan kapasitas instans yang tidak terpakai untuk beban kerja stateless dan toleran terhadap kesalahan.
Auto Scaling	Gunakan konfigurasi Auto Scaling untuk mencocokkan sumber daya komputasi dengan pola lalu lintas.
Penyesuaian ukuran	• Gunakan Compute Optimizer untuk mendapatkan rekomendasi berbasis machine learning mengenai konfigura si komputasi yang paling cocok dengan karakteristik komputasi Anda. • Gunakan AWS Lambda Power Tuning untuk memilih konfigurasi terbaik untuk fungsi Lambda Anda.
Akselerator komputasi berbasis perangkat keras	• Instans komputasi terakselerasi menjalank an fungsi seperti pemrosesan grafis atau pencocokan pola data secara lebih efisien daripada alternatif berbasis CPU. • Untuk beban kerja machine learning, manfaatkan perangkat keras yang dibuat khusus untuk beban kerja Anda, seperti AWS Trainium, AWS Inferentia, dan Amazon EC2 DL1

Sumber daya

Dokumen terkait:

- Komputasi Cloud dengan AWS
- Tipe Instans Amazon EC2
- Kontrol Status Prosesor untuk Instans Amazon EC2 Anda
- Kontainer Amazon EKS: Simpul Pekerja Amazon EKS
- Kontainer Amazon ECS: Instans Kontainer Amazon ECS
- Fungsi: Konfigurasi Fungsi Lambda

Video terkait:

- Fondasi Amazon EC2
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- Mengoptimalkan performa dan biaya untuk komputasi AWS Anda

Contoh terkait:

- Betulkan ukuran dengan pengaktifan penggunaan Memori dan Compute Optimizer
- Kode AWS Compute Optimizer Demo

PERF02-BP03 Mengumpulkan komputasi metrik terkait

Rekam dan lacak metrik terkait komputasi untuk lebih memahami kinerja sumber daya komputasi Anda dan meningkatkan kinerja serta pemanfaatannya.

Antipola umum:

- Anda hanya menggunakan pencarian file log manual untuk metrik.
- Anda hanya menggunakan metrik default yang dicatat oleh perangkat lunak pemantauan Anda.
- Anda hanya meninjau metrik ketika terdapat masalah.

Manfaat menjalankan praktik terbaik ini: Mengumpulkan metrik terkait kinerja akan membantu Anda menyelaraskan kinerja aplikasi dengan persyaratan bisnis untuk memastikan Anda memenuhi

kebutuhan beban kerja Anda. Ini juga dapat membantu Anda terus meningkatkan kinerja dan pemanfaatan sumber daya dalam beban kerja Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Beban kerja dapat menghasilkan data dalam jumlah besar seperti metrik, log, dan event. Di AWS Cloud, mengumpulkan metrik adalah langkah penting untuk meningkatkan keamanan, efisiensi biaya, kinerja, dan keberlanjutan. AWS menyediakan berbagai metrik terkait kinerja menggunakan layanan pemantauan seperti Amazon CloudWatch untuk memberi Anda wawasan yang berharga. Metrik seperti penggunaan CPU, penggunaan memori, I/O disk, serta lalu lintas masuk dan keluar jaringan dapat memberikan wawasan tentang hambatan kinerja atau tingkat penggunaan. Gunakan metrik ini sebagai bagian dari pendekatan berdasarkan data yang digunakan untuk mengatur dan mengoptimalkan sumber daya beban kerja Anda. Dalam kasus yang ideal, Anda harus mengumpulkan semua metrik yang terkait dengan sumber daya komputasi Anda dalam satu platform dengan kebijakan retensi yang diterapkan untuk mendukung sasaran biaya dan operasional.

Langkah implementasi

1. Identifikasi metrik terkait kinerja apa saja yang relevan dengan beban kerja Anda. Anda harus mengumpulkan metrik seputar pemanfaatan sumber daya dan cara cloud Anda beroperasi (seperti waktu respons dan throughput).
   a. Metrik default Amazon EC2
   b. Metrik default Amazon ECS
   c. Metrik default Amazon EKS
   d. Metrik default Lambda
   e. Metrik disk dan memori Amazon EC2
2. Pilih dan siapkan solusi pembuatan log dan pemantauan yang tepat untuk beban kerja Anda.
   a. Observabilitas native AWS
   b. AWS Distro for OpenTelemetry
   c. Amazon Managed Service for Prometheus
3. Tentukan filter dan agregasi yang diperlukan untuk metrik berdasarkan persyaratan beban kerja Anda.
   a. Mengukur metrik aplikasi kustom dengan Amazon CloudWatch Logs dan filter metrik
   b. Mengumpulkan metrik kustom dengan pembuatan tag strategis Amazon CloudWatch

4. Konfigurasikan kebijakan retensi data untuk metrik Anda agar sesuai dengan sasaran keamanan dan operasional Anda.
    a. Retensi data default untuk metrik CloudWatch
    b. Retensi data default untuk CloudWatch Logs
5. Jika diperlukan, buat alarm dan notifikasi untuk metrik Anda agar membantu Anda merespons masalah terkait kinerja secara proaktif.
    a. Membuat alarm untuk metrik kustom menggunakan deteksi anomali Amazon CloudWatch
    b. Membuat metrik dan alarm untuk halaman web tertentu dengan Amazon CloudWatch RUM
6. Gunakan otomatisasi untuk melakukan deployment agen agregasi log dan metrik Anda.
    a. Otomatisasi AWS Systems Manager
    b. OpenTelemetry Collector

Sumber daya

Dokumen terkait:

- Dokumentasi Amazon CloudWatch
- Kumpulkan metrik dan log dari instans Amazon EC2 serta server on-premise dengan Agen CloudWatch
- Mengakses Amazon CloudWatch Logs untuk AWS Lambda
- Menggunakan CloudWatch Logs dengan instans kontainer
- Publikasikan metrik kustom
- Jawaban AWS: Pencatatan Log Terpusat
- Layanan AWS yang Memublikasikan Metrik CloudWatch
- Memantau Amazon EKS pada AWS Fargate

Video terkait:

- Manajemen Kinerja Aplikasi di AWS

Contoh terkait:

- Tingkat 100: Pemantauan dengan Dasbor CloudWatch

- Tingkat 100: Pemantauan instans Windows EC2 dengan Dasbor CloudWatch
- Tingkat 100: Pemantauan instans Amazon Linux EC2 dengan Dasbor CloudWatch

PERF02-BP04 Mengonfigurasi dan menyesuaikan ukuran sumber daya komputasi

Konfigurasikan dan tentukan ukuran yang tepat untuk sumber daya agar sesuai dengan persyaratan kinerja beban kerja Anda dan hindari sumber daya dengan pemanfaatan yang terlalu rendah atau terlalu tinggi.

Antipola umum:

- Anda mengabaikan persyaratan kinerja beban kerja yang menghasilkan sumber daya komputasi dengan pemanfaatan yang terlalu rendah atau terlalu tinggi.
- Anda hanya memilih instans terbesar atau terkecil untuk semua beban kerja.
- Anda hanya menggunakan satu family instans untuk kemudahan manajemen.
- Anda mengabaikan rekomendasi dari AWS Cost Explorer atau Compute Optimizer untuk penentuan ukuran yang tepat.
- Anda tidak mengevaluasi ulang beban kerja untuk kesesuaian tipe instans baru.
- Anda hanya mengesahkan sejumlah kecil konfigurasi instans untuk organisasi Anda.

Manfaat menjalankan praktik terbaik ini: Penyesuaian ukuran yang tepat untuk sumber daya komputasi memastikan pengoperasian yang optimal di cloud dengan menghindari penyediaan sumber daya yang terlalu banyak dan terlalu sedikit. Penyesuaian ukuran sumber daya komputasi secara tepat biasanya menghasilkan kinerja yang lebih baik dan pengalaman pelanggan yang ditingkatkan, sekaligus menurunkan biaya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Penentuan ukuran yang tepat memungkinkan organisasi untuk mengoperasikan infrastruktur cloud mereka dengan cara yang efisien dan hemat biaya sambil menangani kebutuhan bisnis mereka. Penyediaan sumber daya cloud yang berlebihan dapat menyebabkan biaya tambahan, sementara penyediaan yang kurang dapat mengakibatkan kinerja yang buruk dan pengalaman pelanggan yang negatif. AWS menyediakan alat seperti AWS Compute Optimizer dan AWS Trusted Advisor yang menggunakan data historis untuk memberikan rekomendasi untuk menyesuaikan ukuran sumber daya komputasi yang tepat.

Langkah implementasi

- Pilih tipe instans yang paling sesuai dengan kebutuhan Anda:
  - Bagaimana cara memilih jenis instans Amazon EC2 yang tepat untuk beban kerja saya?
  - Pemilihan jenis instans berdasarkan atribut untuk Armada Amazon EC2
  - Membuat grup Auto Scaling menggunakan pemilihan jenis instans berdasarkan atribut
  - Mengoptimalkan biaya komputasi Kubernetes Anda dengan konsolidasi Karpenter
- Analisa berbagai karakteristik kinerja beban kerja Anda serta kaitannya dengan penggunaan memori, jaringan, dan CPU. Gunakan data ini untuk memilih sumber daya yang paling sesuai dengan profil beban kerja dan sasaran kinerja Anda.
- Pantau penggunaan sumber daya Anda menggunakan alat pemantauan AWS sepertiAmazon CloudWatch.
- Pilih konfigurasi yang tepat untuk sumber daya komputasi.
  - Untuk beban kerja sementara, evaluasi metrik Amazon CloudWatch instans seperti `CPUUtilization` untuk mengidentifikasi apakah pemanfaatan instans terlalu rendah atau terlalu tinggi.
  - Untuk beban kerja stabil, periksa alat penyesuaian ukuran AWS seperti AWS Compute Optimizer dan AWS Trusted Advisor secara berkala untuk mengidentifikasi peluang untuk mengoptimalkan dan menyesuaikan ukuran sumber daya komputasi dengan tepat.
    - Lab Well-Architected: Rekomendasi Penyesuaian Ukuran
    - Lab Well-Architected - Penyesuaian Ukuran dengan Compute Optimizer
- Uji perubahan konfigurasi di lingkungan nonproduksi sebelum diterapkan di lingkungan langsung.
- Terus evaluasi ulang penawaran komputasi baru dan bandingkan berdasarkan kebutuhan beban kerja Anda.

Sumber daya

Dokumen terkait:

- Komputasi Cloud dengan AWS
- Tipe Instans Amazon EC2
- Kontainer Amazon ECS: Instans Kontainer Amazon ECS
- Kontainer Amazon EKS: Simpul Pekerja Amazon EKS
- Fungsi: Konfigurasi Fungsi Lambda

- Kontrol Status Prosesor untuk Instans Amazon EC2 Anda

Video terkait:

- Fondasi Amazon EC2
- Komputasi yang lebih baik, lebih cepat, dan lebih murah: Optimisasi biaya Amazon EC2
- Melakukan deployment model ML untuk inferensi dengan performa tinggi dan biaya rendah
- Mengoptimalkan performa dan biaya untuk komputasi AWS Anda
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- Menyederhanakan Pemrosesan Data untuk Meningkatkan Inovasi dengan Alat Nirserver

Contoh terkait:

- Betulkan ukuran dengan pengaktifan penggunaan Memori dan Compute Optimizer
- Kode AWS Compute Optimizer Demo

PERF02-BP05 Menskalakan sumber daya komputasi Anda secara dinamis

Gunakan elastisitas cloud untuk menaikkan atau menurunkan skala sumber daya komputasi Anda secara dinamis agar sesuai dengan kebutuhan Anda dan hindari kapasitas penyediaan yang berlebihan atau terlalu sedikit untuk beban kerja Anda.

Antipola umum:

- Anda bereaksi pada alarm dengan meningkatkan kapasitas secara manual.
- Anda menggunakan pedoman penyesuaian ukuran yang sama (umumnya infrastruktur statis) seperti di on-premise.
- Anda membiarkan peningkatan kapasitas setelah peristiwa penskalaan, bukannya menurunkan kembali skala.

Manfaat menjalankan praktik terbaik ini: Mengonfigurasi dan menguji elastisitas sumber daya komputasi dapat membantu Anda menghemat dana, mempertahankan tolok ukur kinerja, dan meningkatkan keandalan saat lalu lintas berubah.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

AWS memberikan fleksibilitas untuk menaikkan atau menurunkan skala sumber daya Anda secara dinamis melalui berbagai mekanisme penskalaan untuk memenuhi perubahan permintaan. Digabungkan dengan metrik yang terkait dengan komputasi, penskalaan dinamis memungkinkan beban kerja untuk merespons perubahan secara otomatis dan menggunakan rangkaian optimal sumber daya komputasi untuk mencapai tujuannya.

Anda dapat menggunakan sejumlah pendekatan yang berbeda untuk menyesuaikan pasokan sumber daya dengan permintaan.

- Pendekatan pelacakan target: Pantau metrik penskalaan Anda dan tingkatkan atau turunkan kapasitas secara otomatis sesuai kebutuhan.
- Penskalaan prediktif: Lakukan penskalaan dalam mengantisipasi tren harian dan mingguan.
- Pendekatan berbasis jadwal: Tetapkan jadwal penskalaan Anda sendiri sesuai dengan perubahan beban yang dapat diprediksi.
- Penskalaan layanan: Pilih layanan (seperti nirserver) yang secara otomatis menskalakan sesuai rancangan.

Anda harus memastikan bahwa deployment beban kerja dapat menangani peristiwa kenaikan dan penurunan skala.

Langkah implementasi

- Kontainer, fungsi, dan instans komputasi menyediakan mekanisme bagi elastisitas melalui kombinasi dengan penskalaan otomatis atau sebagai fitur layanan. Berikut beberapa contoh mekanisme penskalaan otomatis:

Mekanisme Penskalaan Otomatis	Di mana harus menggunakan
Amazon EC2 Auto Scaling	Untuk memastikan Anda memiliki jumlah yang tepat untuk instans Amazon EC2 yang tersedia guna menangani beban pengguna untuk aplikasi Anda.
Application Auto Scaling	Untuk secara otomatis menskalakan sumber daya bagi layanan AWS individu di luar Amazon EC2, seperti fungsi AWS Lambda

Mekanisme Penskalaan Otomatis	Di mana harus menggunakan
	atau layanan Amazon Elastic Container Service (Amazon ECS) .
Kubernetes Cluster Autoscaler/Karpenter	Untuk secara otomatis menskalakan klaster Kubernetes.

- Penskalaan sering dibahas terkait dengan layanan komputasi seperti Instans Amazon EC2 atau fungsi AWS Lambda. Pastikan juga untuk mempertimbangkan konfigurasi layanan nonkomputasi seperti AWS Glue untuk mengimbangi permintaan.
- Pastikan metrik untuk penskalaan cocok dengan karakteristik beban kerja yang sedang digunakan. Jika Anda men-deploy aplikasi transkode video, 100% pemanfaatan CPU adalah hal normal dan tidak boleh menjadi metrik primer Anda. Gunakan kedalaman antrean tugas transkode sebagai gantinya. Anda dapat menggunakan metrik yang disesuaikan untuk kebijakan penskalaan Anda jika diperlukan. Untuk memilih metrik yang tepat, pertimbangkan panduan berikut untuk Amazon EC2:
  - Metrik harus merupakan metrik pemanfaatan yang valid dan mendeskripsikan tingkat kesibukan suatu instans.
  - Nilai metrik harus meningkat atau menurun secara proporsional dengan jumlah instans dalam grup Auto Scaling.
- Pastikan Anda menggunakan penskalaan dinamis, bukan penskalaan manual untuk grup Auto Scaling Anda. Sebaiknya gunakan juga kebijakan penskalaan pelacakan target dalam penskalaan dinamis.
- Pastikan deployment beban kerja dapat menangani event penskalaan (naik dan turun). Sebagai contoh, Anda dapat menggunakan Riwayat aktivitas guna memastikan aktivitas penskalaan untuk grup Auto Scaling.
- Evaluasi beban kerja Anda untuk pola terprediksi dan secara proaktif skalakan saat Anda mengantisipasi perubahan terencana dan terprediksi dalam permintaan. Dengan penskalaan prediktif, Anda dapat meniadakan kebutuhan untuk menyediakan kapasitas secara berlebih. untuk detail selengkapnya, lihat Penskalaan Prediktif dengan Amazon EC2 Auto Scaling.

Sumber daya

Dokumen terkait:

- Komputasi Cloud dengan AWS

- Tipe Instans Amazon EC2
- Kontainer Amazon ECS: Instans Kontainer Amazon ECS
- Kontainer Amazon EKS: Simpul Pekerja Amazon EKS
- Fungsi: Konfigurasi Fungsi Lambda
- Kontrol Status Prosesor untuk Instans Amazon EC2 Anda
- Pendalaman tentang Amazon ECS Cluster Auto Scaling
- Memperkenalkan Karpenter - Kubernetes Cluster Autoscaler Sumber Terbuka Berkinerja Tinggi

Video terkait:

- Fondasi Amazon EC2
- Komputasi yang lebih baik, lebih cepat, dan lebih murah: Optimisasi biaya Amazon EC2
- Mengoptimalkan performa dan biaya untuk komputasi AWS Anda
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- Bangun lingkungan komputasi yang hemat biaya, energi, dan sumber daya

Contoh terkait:

- Contoh Grup Amazon EC2 Auto Scaling
- Menerapkan Penskalaan Otomatis dengan Karpenter

PERF02-BP06 Menggunakan akselerator komputasi berbasis perangkat keras yang dioptimalkan

Gunakan akselerator perangkat keras untuk melakukan fungsi tertentu secara lebih efisien daripada alternatif berbasis CPU.

Antipola umum:

- Dalam beban kerja Anda, Anda belum melakukan uji tolok ukur instans tujuan umum dengan instans yang dibuat khusus yang dapat memberikan kinerja lebih tinggi dan biaya lebih rendah.
- Anda menggunakan akselerator komputasi berbasis perangkat keras untuk tugas yang bisa lebih efisien jika menggunakan alternatif berbasis CPU.
- Anda tidak memantau penggunaan GPU.

Manfaat menjalankan praktik terbaik ini: Dengan menggunakan akselerator berbasis perangkat keras, seperti unit pemrosesan grafis (GPU) dan field programmable gate array (FPGA), Anda dapat melakukan fungsi pemrosesan tertentu dengan lebih efisien.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Instans komputasi terakselerasi menyediakan akses ke akselerator komputasi berbasis perangkat keras seperti GPU dan FPGA. Akselerator perangkat keras ini menjalankan fungsi-fungsi tertentu seperti pemrosesan grafis atau pencocokan pola data secara lebih efisien daripada alternatif berbasis CPU. Banyak beban kerja yang terakselerasi, seperti perenderan, transkode, dan machine learning, memiliki variabel tinggi sehubungan dengan penggunaan sumber daya. Jalankan perangkat keras ini hanya ketika diperlukan, dan nonaktifkan instans GPU secara otomatis saat tidak diperlukan untuk meningkatkan keseluruhan efisiensi kinerja.

Langkah implementasi

- Identifikasi instans komputasi terakselerasi mana yang dapat memenuhi kebutuhan Anda.
- Untuk beban kerja machine learning, manfaatkan perangkat keras yang dibuat khusus untuk beban kerja Anda, seperti AWS Trainium, AWS Inferentia, dan Amazon EC2 DL1. Instans AWS Inferentia seperti instans Inf2 menawarkan hingga 50% peningkatan kinerja/watt dibandingkan instans Amazon EC2 yang setara.
- Kumpulkan metrik penggunaan untuk instans komputasi terakselerasi Anda. Misalnya, Anda dapat menggunakan agen CloudWatch untuk mengumpulkan metrik seperti `utilization_gpu` dan `utilization_memory` untuk GPU Anda seperti yang ditunjukkan di Kumpulkan metrik GPU NVIDIA dengan Amazon CloudWatch.
- Optimalkan kode, operasi jaringan, dan pengaturan akselerator perangkat keras untuk memastikan perangkat keras yang mendasarinya dimanfaatkan sepenuhnya.
  - Optimalkan pengaturan GPU
  - Pemantauan dan Pengoptimalan GPU dalam AMI Deep Learning
  - Mengoptimalkan I/O untuk penyetelan kinerja GPU pelatihan deep learning di Amazon SageMaker AI
- Gunakan driver GPU dan pustaka berkinerja tinggi terbaru.
- Gunakan otomatisasi untuk melepaskan instans GPU ketika tidak digunakan.

Sumber daya

Dokumen terkait:

- Instans GPU
- Instans dengan AWS Trainium
- Instans dengan AWS Inferentia
- Let's Architect! Merancang dengan chip dan akselerator kustom

- Komputasi Dipercepat
- Instans VT1 Amazon EC2
- Bagaimana cara memilih jenis instans Amazon EC2 yang tepat untuk beban kerja saya?
- Pilih akselerator AI dan kompilasi model terbaik untuk inferensi penglihatan komputer dengan Amazon SageMaker AI

Video terkait:

- Cara memilih instans GPU Amazon EC2 untuk deep learning
- Melakukan Deployment Inferensi Deep Learning yang Hemat Biaya

Manajemen data

PERF 3. Bagaimana cara menyimpan, mengelola, dan mengakses data dalam beban kerja Anda?

Solusi manajemen data yang optimal untuk sistem tertentu bervariasi berdasarkan jenis data (blok, file, atau objek), pola akses (acak atau berurutan), throughput yang diperlukan, frekuensi akses (online, offline, arsip), frekuensi pembaruan (WORM, dinamis), dan ketersediaan serta batasan daya tahan. Beban kerja Well-Architected menggunakan penyimpanan data yang dibuat khusus yang memungkinkan berbagai fitur untuk meningkatkan kinerja.

Praktik terbaik

- PERF03-BP01 Menggunakan penyimpanan data yang dibuat khusus yang paling mendukung persyaratan akses dan penyimpanan data Anda
- PERF03-BP02 Evaluasi opsi konfigurasi yang tersedia untuk penyimpanan data

- PERF03-BP03 Mengumpulkan dan merekam metrik kinerja penyimpanan data
- PERF03-BP04 Menerapkan strategi untuk meningkatkan kinerja kueri di penyimpanan data
- PERF03-BP05 Mengimplementasikan pola akses data yang memanfaatkan caching

PERF03-BP01 Menggunakan penyimpanan data yang dibuat khusus yang paling mendukung persyaratan akses dan penyimpanan data Anda

Pahami karakteristik data (seperti dapat dibagikan, ukuran, ukuran cache, pola akses, latensi, throughput, dan persistensi data) untuk memilih penyimpanan data khusus (penyimpanan atau basis data) yang tepat untuk beban kerja Anda.

Antipola umum:

- Anda bertahan dengan satu solusi basis data disebabkan pengetahuan dan pengalaman internal tentang satu jenis solusi basis data tertentu.
- Anda berasumsi bahwa semua beban kerja memiliki persyaratan penyimpanan dan akses data yang serupa.
- Anda belum mengimplementasikan katalog data untuk menginventarisasi aset data Anda.

Manfaat menjalankan praktik terbaik ini: Dengan memahami karakteristik dan persyaratan data, Anda dapat menentukan teknologi penyimpanan yang paling efisien dan berkinerja paling tinggi sesuai dengan kebutuhan beban kerja Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Saat memilih dan menerapkan penyimpanan data, pastikan karakteristik penyimpanan, kueri, dan penskalaan mendukung persyaratan data beban kerja. AWS menyediakan banyak teknologi penyimpanan data dan basis data termasuk penyimpanan blok, penyimpanan objek, penyimpanan streaming, sistem file, relasional, nilai kunci, dokumen, penyimpanan dalam memori, grafik, deret waktu, dan basis data buku besar. Setiap solusi manajemen data memiliki opsi dan konfigurasi yang tersedia bagi Anda untuk mendukung kasus penggunaan dan model data Anda. Dengan memahami karakteristik dan persyaratan data, Anda dapat melepaskan diri dari teknologi penyimpanan monolitik dan pendekatan satu-untuk-semua yang terbatas guna memfokuskan diri pada manajemen data yang tepat.

Langkah implementasi

- Lakukan inventaris berbagai jenis data yang ada dalam beban kerja Anda.
- Pahami dan dokumentasikan karakteristik serta persyaratan data, termasuk:
  - Tipe data (tidak terstruktur, semi-terstruktur, relasional)
  - Volume dan pertumbuhan data
  - Ketahanan data: persisten, sementara, transien
  - Persyaratan ACID (atomisitas, konsistensi, isolasi, durabilitas)
  - Pola akses data (intensif baca atau intensif tulis)
  - Latensi
  - Throughput
  - IOPS (operasi input/output per detik)
  - Periode retensi data
- Pelajari berbagai penyimpanan data yang tersedia untuk beban kerja Anda di AWS yang dapat memenuhi karakteristik data Anda (sebagaimana diuraikan dalam PERF01-BP01 Mempelajari dan memahami layanan serta fitur cloud yang tersedia). Berikut adalah beberapa contoh teknologi penyimpanan AWS serta karakteristik utamanya:

Tipe	Layanan AWS	Karakteristik utama
Penyimpanan objek	Amazon S3	Skalabilitas tak terbatas, ketersediaan tinggi, dan berbagai opsi aksesibilitas. Mentransfer dan mengakses objek masuk dan keluar dari Amazon S3 dapat menggunakan layanan, seperti Transfer Acceleration atau Access Points, untuk mendukung lokasi, kebutuhan keamanan, dan pola akses Anda.

Tipe	Layanan AWS	Karakteristik utama
Penyimpanan pengarsipan	Amazon Glacier	Dirancang untuk pengarsipan data.
Penyimpanan streaming	Amazon Kinesis Amazon Managed Streaming for Apache Kafka (Amazon MSK)	Penyerapan dan penyimpan an data streaming yang efisien.
Sistem file bersama	Amazon Elastic File System (Amazon EFS)	Sistem file yang dapat dipasang yang dapat diakses oleh berbagai jenis solusi komputasi.
Sistem file bersama	Amazon FSx	Dibangun berdasarkan solusi komputasi AWS terbaru untuk mendukung empat sistem file yang umum digunakan: NetApp ONTAP, OpenZFS, Windows File Server, dan Lustre. Amazon FSx memiliki latensi, throughput, dan IOPS yang bervariasi per sistem file dan hal ini harus dipertimbangkan saat memilih sistem file yang tepat untuk kebutuhan beban kerja Anda.

Tipe	Layanan AWS	Karakteristik utama
Penyimpanan blok	Amazon Elastic Block Store (Amazon EBS)	Layanan penyimpanan blok kinerja tinggi yang dapat diskalakan yang dirancang untuk Amazon Elastic Compute Cloud (Amazon EC2). Amazon EBS mencakup penyimpanan yang didukung SSD untuk beban kerja transaksional intensif IOPS dan penyimpanan yang didukung HDD untuk beban kerja intensif throughput.
Basis data relasional	Amazon Aurora, Amazon RDS, Amazon Redshift.	Didesain untuk mendukung transaksi ACID (atomisit as, konsistensi, isolasi, durabilitas), dan mempertah ankan integritas referensi al serta konsistensi data yang tinggi. Banyak aplikasi tradisional, perencanaan sumber daya perusahaan (ERP), manajemen hubungan pelanggan (CRM), dan perdagangan elektronik menggunakan basis data relasional untuk menyimpan data mereka.

Tipe	Layanan AWS	Karakteristik utama
Basis data nilai-kunci	Amazon DynamoDB	Dioptimalkan untuk pola akses umum, biasanya untuk menyimpan dan mengambil data dalam volume besar. Aplikasi web dengan lalu lintas tinggi, sistem perdagangan elektronik, dan aplikasi gaming merupakan kasus penggunaan umum untuk basis data nilai kunci.
Basis data dokumen	Amazon DocumentDB	Dirancang untuk menyimpan data semi-terstruktur sebagai dokumen mirip JSON. Basis data ini membantu developer dengan cepat membangun dan memperbarui aplikasi seperti manajemen konten, katalog, dan profil pengguna.
Basis data dalam memori	Amazon ElastiCache , Amazon MemoryDB for Redis	Digunakan untuk aplikasi yang memerlukan akses waktu nyata ke data, latensi rendah, dan throughput paling tinggi. Anda dapat menggunakan basis data dalam memori untuk caching aplikasi, manajemen sesi, papan peringkat game, penyimpanan fitur ML latensi rendah, sistem olah pesan layanan mikro, dan mekanisme streaming throughput tinggi

Tipe	Layanan AWS	Karakteristik utama
Basis data grafik	Amazon Neptune	Digunakan untuk aplikasi yang harus menavigasi dan melakukan kueri jutaan hubungan antara set data grafik yang sangat terhubung dan latensi milidetik pada skala besar. Banyak perusahaan menggunakan basis data grafik untuk mesin rekomendasi, jaringan sosial, dan deteksi penipuan.
Basis Data Deret Waktu	Amazon Timestream	Digunakan untuk mengumpulkan, mempersatukan, dan mengambil wawasan secara efisien dari data yang berubah seiring waktu. Aplikasi internet untuk segala (IoT), DevOps, dan telemetri industri dapat menggunakan basis data deret waktu.
Kolom lebar	Amazon Keyspaces (untuk Apache Cassandra)	Menggunakan tabel, baris, dan kolom, tetapi tidak seperti basis data relasional, nama dan format kolomnya bervariasi dari baris ke baris di tabel yang sama. Biasanya Anda akan melihat penyimpanan kolom lebar di aplikasi industri skala tinggi untuk pemeliharaan perlengkapan, pengelolaan armada, dan pengoptimalan rute.

Tipe	Layanan AWS	Karakteristik utama
Buku besar	Amazon Quantum Ledger Database (Amazon QLDB)	Memberikan otoritas terpusat yang tepercaya untuk mempertahankan data transaksi yang dapat diskalakan, tetap, dan dapat diverifikasi secara kriptogra fis untuk setiap aplikasi. Basis data buku besar digunakan untuk sistem catatan, rantai pasokan, registrasi, dan bahkan transaksi perbankan .

- Jika Anda membangun platform data, manfaatkan arsitektur data modern di AWS untuk mengintegrasikan danau data, gudang data, dan tempat penyimpanan data Anda yang dibuat khusus.
- Pertanyaan kunci yang perlu Anda pertimbangkan saat memilih penyimpanan data untuk beban kerja Anda adalah sebagai berikut:

Pertanyaan	Hal-hal yang perlu dipertimbangkan
Bagaimana data terstruktur?	• Jika data tidak terstruktur, pertimbangkan penyimpanan objek seperti Amazon S3 atau basis data NoSQL seperti Amazon DocumentDB • Untuk data nilai kunci, pertimbangkan DynamoDB, Amazon ElastiCache (Redis OSS) atau Amazon MemoryDB
Apa tingkat integritas referensial yang dibutuhkan?	• Untuk kendala utama asing, basis data relasional seperti Amazon RDS dan Aurora dapat memberikan tingkat integritas ini. • Biasanya, dalam model data NoSQL, Anda akan melakukan denormalisasi data

Pertanyaan	Hal-hal yang perlu dipertimbangkan
	menjadi satu dokumen atau kumpulan dokumen untuk diambil dalam satu permintaan dan bukannya digabungkan dalam berbagai dokumen atau tabel.
Apakah diperlukan kepatuhan terhadap ACID (atomisitas, konsistensi, isolasi, durabilitas)?	• Jika diperlukan sifat ACID yang terkait dengan basis data relasional, pertimbangkan basis data relasional seperti Amazon RDS dan Aurora. • Jika konsistensi yang kuat diperlukan untuk basis data NoSQL, Anda dapat menggunakan bacaan sangat konsisten dengan DynamoDB.
Bagaimana persyaratan penyimpanan akan berubah seiring waktu? Bagaimana dampaknya pada skalabilitas?	• Basis data nirserver seperti DynamoDB dan Amazon Quantum Ledger Database (Amazon QLDB) akan melakukan penskalaan secara dinamis. • Basis data relasional memiliki batas atas terkait penyimpanan yang tersedia, dan sering kali harus dipartisi secara horizontal menggunakan mekanisme seperti serpihan setelah penyimpanan tersebut mencapai batas ini.
Berapakah proporsi kueri baca dibandingkan dengan kueri tulis? Apakah caching akan meningkatkan performa?	• Beban kerja yang sarat baca dapat diuntungkan dari lapisan caching, seperti ElastiCache atau DAX jika basis datanya DynamoDB. • Bacaan juga dapat dilimpahkan ke replika baca dengan basis data relasional seperti Amazon RDS.

Pertanyaan	Hal-hal yang perlu dipertimbangkan
Apakah penyimpanan dan modifikasi (OLTP - Pemrosesan Transaksi Online) atau pengambilan dan pelaporan (OLAP - Pemrosesan Analitik Online) memiliki prioritas lebih tinggi?	• Untuk pemrosesan transaksional baca apa adanya throughput tinggi, pertimbangkan basis data NoSQL seperti DynamoDB. • Untuk throughput tinggi dan pola baca yang kompleks (seperti join) dengan konsistensi, gunakan Amazon RDS. • Untuk kueri analitik, pertimbangkan basis data kolom seperti Amazon Redshift atau ekspor data ke Amazon S3 dan melakukan analisis menggunakan Athena atau Amazon Quick.
Tingkat durabilitas apa yang diperlukan data?	• Aurora secara otomatis mereplikasi data Anda di tiga Zona Ketersediaan dalam satu Wilayah, yang artinya data Anda sangat tahan lama dengan lebih sedikit kemungkin an hilangnya data. • DynamoDB secara otomatis direplikasi di beberapa Zona Ketersediaan, memberikan durabilitas data dan ketersediaan tinggi. • Amazon S3 memberikan 11 sembilan durabilitas. Banyak layanan basis data seperti Amazon RDS dan DynamoDB mendukung ekspor data ke Amazon S3 untuk pengarsipan dan retensi jangka panjang.
Apakah ada keinginan untuk beralih dari mesin basis data komersial atau biaya lisensi?	• Pertimbangkan mesin sumber terbuka seperti PostgreSQL dan MySQL di Amazon RDS atau Aurora. • Manfaatkan AWS Database Migration Service dan AWS Schema Conversion Tool untuk melakukan migrasi dari mesin basis data komersial ke sumber terbuka

Pertanyaan	Hal-hal yang perlu dipertimbangkan
Apakah harapan operasional untuk basis data? Apakah beralih ke layanan terkelola merupakan masalah utama?	• Pemanfaatan Amazon RDS sebagai ganti Amazon EC2, dan DynamoDB atau Amazon DocumentDB sebagai ganti hosting mandiri basis data NoSQL dapat mengurangi biaya tambahan operasional.
Bagaimana basis data diakses saat ini? Apakah hanya akses aplikasi, atau adakah pengguna kecerdasan bisnis (BI) dan aplikasi umum lain yang terhubung?	• Jika Anda memiliki ketergantungan pada alat eksternal maka Anda mungkin harus mempertahankan kompatibilitas dengan basis data yang didukungnya. Amazon RDS sepenuhnya kompatibel dengan berbagai versi mesin yang didukungnya, termasuk Microsoft SQL Server, Oracle, MySQL, dan PostgreSQL.

- Lakukan uji coba dan uji tolok ukur di lingkungan nonproduksi untuk mengidentifikasi penyimpanan data mana yang paling sesuai dengan kebutuhan beban kerja Anda.

Sumber daya

Dokumen terkait:

- Jenis Volume Amazon EBS
- Penyimpanan Amazon EC2
- Amazon EFS: Performa Amazon EFS
- Performa Amazon FSx for Lustre
- Performa Amazon FSx for Windows File Server
- Amazon Glacier: Dokumentasi Amazon Glacier
- Amazon S3: Pertimbangan Tingkat Permintaan dan Performa
- Penyimpanan Cloud dengan AWS
- Karakteristik I/O Amazon EBS
- Basis Data Cloud dengan AWS
- Caching Basis Data AWS

- DynamoDB Accelerator
- Praktik terbaik Amazon Aurora
- Performa Amazon Redshift
- 10 kiat performa terbaik Amazon Athena
- Praktik terbaik Amazon Redshift Spectrum
- Praktik terbaik Amazon DynamoDB
- Pilih antara Amazon EC2 dan Amazon RDS
- Praktik Terbaik untuk Mengimplementasikan Amazon ElastiCache

Video terkait:

- Pendalaman tentang Amazon EBS
- Mengoptimalkan kinerja penyimpanan Anda dengan Amazon S3
- Lakukan modernisasi aplikasi dengan basis data yang dibuat khusus
- Penjelasan penyimpanan Amazon Aurora: Cara kerjanya
- Pendalaman Amazon DynamoDB: Pola desain lanjutan

Contoh terkait:

- Driver CSI Amazon EFS
- Driver CSI Amazon EBS
- Utilitas Amazon EFS
- Penskalaan Otomatis Amazon EBS
- Contoh Amazon S3
- Optimalkan Pola Data menggunakan Pembagian Data Amazon Redshift
- Migrasi Basis Data
- MS SQL Server - Demo Replikasi AWS Database Migration Service (AWS DMS)
- Lokakarya Praktik Langsung Modernisasi Basis Data
- Sampel Amazon Neptune

PERF03-BP02 Evaluasi opsi konfigurasi yang tersedia untuk penyimpanan data

Pahami dan evaluasi berbagai fitur dan opsi konfigurasi yang tersedia untuk penyimpanan data Anda guna mengoptimalkan ruang penyimpanan dan kinerja untuk beban kerja Anda.

Antipola umum:

- Anda hanya menggunakan satu jenis penyimpanan, seperti Amazon EBS, untuk semua beban kerja.
- Anda menggunakan IOPS yang tersedia untuk semua beban kerja tanpa pengujian dunia nyata terhadap semua tingkat penyimpanan.
- Anda tidak memahami opsi konfigurasi pada solusi manajemen data yang Anda pilih.
- Anda hanya mengandalkan peningkatan ukuran instans tanpa mempertimbangkan opsi konfigurasi lain yang tersedia.
- Anda tidak menguji karakteristik penskalaan penyimpanan data Anda.

Manfaat menjalankan praktik terbaik ini: Dengan menjelajahi dan melakukan eksperimen dengan konfigurasi penyimpanan data, Anda mungkin dapat mengurangi biaya infrastruktur, meningkatkan performa, serta mengurangi upaya pengelolaan beban kerja.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Beban kerja dapat memiliki satu atau beberapa penyimpanan data yang digunakan berdasarkan persyaratan penyimpanan dan akses data. Untuk mengoptimalkan biaya dan efisiensi kinerja, Anda harus mengevaluasi pola akses data guna menentukan konfigurasi penyimpanan data yang sesuai. Saat mencoba opsi-opsi penyimpanan data, pertimbangkan beberapa aspek seperti opsi penyimpanan, memori, komputasi, replika baca, persyaratan konsistensi, pooling koneksi, dan opsi cache. Coba beberapa opsi konfigurasi ini untuk meningkatkan metrik efisiensi kinerja.

Langkah implementasi

- Pahami konfigurasi saat ini (seperti tipe instans, ukuran penyimpanan, atau versi mesin basis data) penyimpanan data Anda.
- Tinjau dokumentasi dan praktik terbaik AWS untuk mempelajari rekomendasi opsi konfigurasi yang dapat membantu meningkatkan kinerja penyimpanan data Anda. Berikut ini adalah opsi-opsi penyimpanan data utama yang perlu dipertimbangkan:

Opsi konfigurasi	Contoh
Melimpahkan beban baca (seperti replika baca dan caching)	• Untuk tabel DynamoDB, Anda dapat meringankan beban baca menggunakan DAX untuk caching. • Anda dapat membuat klaster Amazon ElastiCache (Redis OSS) dan mengonfig urasikan aplikasi Anda untuk membaca dari cache terlebih dahulu, kembali ke basis data jika item yang diminta tidak tersedia. • Basis data relasional seperti Amazon RDS dan Aurora, serta basis data NoSQL yang tersedia seperti Neptune dan Amazon DocumentDB semua mendukung penambahan replika baca untuk mengurang i porsi baca beban kerja. • Basis data nirserver seperti DynamoDB akan menskalakan secara otomatis. Pastikan unit kapasitas baca (RSU) yang tersedia cukup untuk mengatasi beban kerja.

Opsi konfigurasi	Contoh
Menskalakan penulisan (seperti serpihan kunci partisi atau memperkenalkan antrean)	• Untuk basis data relasional, Anda dapat memperbesar ukuran instans untuk mengakomodasi tambahan beban kerja atau menambah IOPS yang tersedia untuk memfasilitasi kenaikan throughput pada penyimpanan yang mendasari. • Anda juga dapat membuat antrean di depan basis data, bukan menulis secara langsung ke basis data. Dengan pola ini, Anda dapat memisahkan penyerapan dari basis data dan mengontrol tingkat aliran, sehingga basis data tidak kewalahan. • Mengganti pembuatan transaksi berdurasi pendek dengan pembuatan batch permintaan penulisan dapat membantu meningkatkan throughput dalam basis data relasional dengan volume penulisan tinggi. • Basis data nirserver seperti DynamoDB dapat menskalakan throughput tulis secara otomatis atau dengan menyesuaikan unit kapasitas tulis (WCU) yang tersedia, bergantung pada mode kapasitasnya. • Anda tetap dapat menjumpai masalah dengan partisi panas ketika Anda mencapai batas throughput pada kunci partisi tertentu. Hal ini dapat dikurangi dengan memilih distribusi kunci partisi yang lebih merata atau dengan memisah penulisan kunci partisi.

Opsi konfigurasi	Contoh
Kebijakan untuk mengelola siklus hidup set data Anda	• Anda dapat menggunakan Siklus Hidup Amazon S3 untuk mengelola objek Anda sepanjang siklus hidupnya. Jika pola akses Anda tidak diketahui, berubah, atau tidak dapat diprediksi, Anda dapat menggunak an Amazon S3 Intelligent-Tiering, yang memantau pola akses dan secara otomatis memindahkan objek yang belum diakses ke tingkat akses dengan biaya lebih rendah. Anda dapat memanfaatkan metrik Lensa Penyimpanan Amazon S3 untuk mengident ifikasi peluang dan celah optimisasi dalam manajemen siklus hidup. • Manajemen siklus hidup Amazon EFS secara otomatis mengelola penyimpan an file untuk sistem file Anda.
Manajemen koneksi dan pooling	• Proksi Amazon RDS dapat digunakan dengan Amazon RDS dan Aurora untuk mengelola koneksi ke basis data. • Basis data nirserver seperti DynamoDB tidak terkait dengan koneksi apa pun, tetapi pertimbangkan kapasitas yang tersedia atau kebijakan penskalaan otomatis untuk mengatasi lonjakan beban.

- Lakukan uji coba dan uji tolok ukur di lingkungan nonproduksi untuk mengidentifikasi opsi konfigurasi mana yang dapat memenuhi persyaratan beban kerja Anda.
- Setelah bereksperimen, rencanakan migrasi dan validasikan metrik kinerja Anda.
- Gunakan alat pemantauan AWS (seperti Amazon CloudWatch) dan optimisasi (seperti Lensa Penyimpanan Amazon S3) untuk terus mengoptimalkan penyimpanan data Anda menggunakan pola penggunaan dunia nyata.

Sumber daya

Dokumen terkait:

- Penyimpanan Cloud dengan AWS
- Jenis Volume Amazon EBS
- Penyimpanan Amazon EC2
- Amazon EFS: Performa Amazon EFS
- Performa Amazon FSx for Lustre
- Performa Amazon FSx for Windows File Server
- Amazon Glacier: Dokumentasi Amazon Glacier
- Amazon S3: Pertimbangan Tingkat Permintaan dan Performa
- Penyimpanan Cloud dengan AWS
- Penyimpanan Cloud dengan AWS
- Karakteristik I/O Amazon EBS
- Basis Data Cloud dengan AWS
- Caching Basis Data AWS
- DynamoDB Accelerator
- Praktik terbaik Amazon Aurora
- Performa Amazon Redshift
- 10 kiat performa terbaik Amazon Athena
- Praktik terbaik Amazon Redshift Spectrum
- Praktik terbaik Amazon DynamoDB

Video terkait:

- Pendalaman tentang Amazon EBS
- Mengoptimalkan kinerja penyimpanan Anda dengan Amazon S3
- Lakukan modernisasi aplikasi dengan basis data yang dibuat khusus
- Penjelasan penyimpanan Amazon Aurora: Cara kerjanya
- Pendalaman Amazon DynamoDB: Pola desain lanjutan

Contoh terkait:

- Driver CSI Amazon EFS
- Driver CSI Amazon EBS
- Utilitas Amazon EFS
- Penskalaan Otomatis Amazon EBS
- Contoh Amazon S3
- Contoh Amazon DynamoDB
- Sampel migrasi Basis Data AWS
- Lokakarya Modernisasi Basis Data
- Menggunakan parameter di Amazon RDS for Postgress DB

PERF03-BP03 Mengumpulkan dan merekam metrik kinerja penyimpanan data

Lacak dan rekam metrik kinerja yang relevan untuk penyimpanan data Anda guna memahami kinerja solusi manajemen data Anda. Metrik-metrik ini dapat membantu Anda mengoptimalkan penyimpanan data Anda, memastikan terpenuhinya persyaratan beban kerja Anda, dan memberikan gambaran umum yang jelas tentang kinerja beban kerja.

Antipola umum:

- Anda hanya menggunakan pencarian file log manual untuk metrik.
- Anda hanya memublikasikan metrik ke alat-alat internal yang digunakan tim Anda dan tidak memiliki gambaran yang komprehensif tentang beban kerja Anda.
- Anda hanya menggunakan metrik default yang dicatat oleh perangkat lunak pemantauan Anda yang dipilih.
- Anda hanya meninjau metrik ketika terdapat masalah.
- Anda hanya memantau metrik tingkat sistem dan tidak merekam metrik akses atau penggunaan data.

Manfaat menjalankan praktik terbaik ini: Memiliki dasar acuan kinerja membantu Anda memahami perilaku normal dan persyaratan beban kerja. Pola abnormal dapat diidentifikasi dan diperbaiki lebih cepat sehingga meningkatkan kinerja dan keandalan penyimpanan data.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Untuk memantau kinerja penyimpanan data, Anda harus merekam beberapa metrik kinerja secara berkala. Dengan begitu Anda dapat mendeteksi anomali serta mengukur kinerja berdasarkan metrik bisnis untuk memastikan kebutuhan beban kerja Anda terpenuhi.

Metrik harus menyertakan metrik sistem dasar yang mendukung penyimpanan data serta metrik basis data. Metrik sistem dasar dapat meliputi metrik pemanfaatan CPU, memori, penyimpanan disk yang tersedia, I/O disk, rasio cache hit, dan jaringan masuk serta keluar, sedangkan metrik basis data dapat meliputi transaksi per detik, kueri teratas, rata-rata laju kueri, waktu respons, penggunaan indeks, penguncian tabel, batas waktu kueri, dan jumlah koneksi yang terbuka. Data ini sangat penting untuk mengetahui kinerja beban kerja dan bagaimana solusi manajemen data digunakan. Gunakan metrik ini sebagai bagian dari pendekatan berbasis data yang digunakan untuk mengatur dan mengoptimalkan sumber daya beban kerja Anda.

Gunakan alat, pustaka, dan sistem yang merekam pengukuran kinerja terkait kinerja basis data.

Langkah implementasi

1. Identifikasi metrik kinerja utama yang perlu dilacak oleh penyimpanan data Anda.
    a. Metrik dan dimensi Amazon S3
    b. Memantau metrik untuk instans Amazon RDS
    c. Memantau beban DB dengan Wawasan Performa di Amazon RDS
    d. Ikhtisar Pemantauan yang Ditingkatkan
    e. Metrik dan dimensi DynamoDB
    f. Memantau DynamoDB Accelerator
    g. Memantau Amazon MemoryDB dengan Amazon CloudWatch
    h. Metrik Mana yang Harus Saya Pantau?
    i. Memantau kinerja klaster Amazon Redshift
    j. Metrik dan dimensi Timestream
    k. Metrik Amazon CloudWatch untuk Amazon Aurora
    l. Pencatatan log dan pemantauan di Amazon Keyspaces (for Apache Cassandra)
    m. Memantau Sumber Daya Amazon Neptune
2. Gunakan solusi pencatatan log dan pemantauan yang disetujui untuk mengumpulkan metrik ini. Amazon CloudWatch dapat mengumpulkan metrik di seluruh sumber daya dalam arsitektur Anda. Anda juga dapat mengumpulkan dan memublikasikan metrik kustom untuk memunculkan

metrik turunan (derived metric) atau bisnis. Gunakan CloudWatch atau solusi pihak ketiga untuk menetapkan alarm yang memberikan indikasi saat ambang batas terlampaui.

3. Periksa apakah pemantauan penyimpanan data dapat terbantu dengan solusi machine learning yang mendeteksi anomali kinerja.
   a. Amazon DevOps Guru untuk Amazon RDS menyediakan visibilitas masalah kinerja dan memberikan saran tindakan perbaikan.
4. Konfigurasikan retensi data dalam solusi pemantauan dan pencatatan log Anda agar sesuai dengan tujuan keamanan dan operasional Anda.
   a. Retensi data default untuk metrik CloudWatch
   b. Retensi data default untuk CloudWatch Logs

Sumber daya

Dokumen terkait:

- Caching Basis Data AWS
- 10 kiat performa terbaik Amazon Athena
- Praktik terbaik Amazon Aurora
- DynamoDB Accelerator
- Praktik terbaik Amazon DynamoDB
- Praktik terbaik Amazon Redshift Spectrum
- Performa Amazon Redshift
- Basis Data Cloud dengan AWS
- Wawasan Kinerja Amazon RDS

Video terkait:

- Basis data yang dibuat khusus AWS
- Penjelasan penyimpanan Amazon Aurora: Cara kerjanya
- Pendalaman Amazon DynamoDB: Pola desain lanjutan
- Praktik Terbaik untuk Memantau Beban Kerja Redis di Amazon ElastiCache

Contoh terkait:

- Tingkat 100: Pemantauan dengan Dasbor CloudWatch
- Kerangka Kerja Pengumpulan Metrik Penyerapan Set Data AWS
- Lokakarya Pemantauan Amazon RDS

PERF03-BP04 Menerapkan strategi untuk meningkatkan kinerja kueri di penyimpanan data

Terapkan strategi untuk mengoptimalkan data dan meningkatkan kueri data untuk memungkinkan skalabilitas yang lebih besar dan kinerja yang efisien untuk beban kerja Anda.

Antipola umum:

- Anda tidak mempartisi data di penyimpanan data Anda.
- Anda menyimpan data hanya dalam satu format file di penyimpanan data Anda.
- Anda tidak menggunakan indeks di penyimpanan data Anda.

Manfaat menjalankan praktik terbaik ini: Optimisasi data dan performa kueri menghasilkan efisiensi yang lebih tinggi, biaya lebih rendah, dan pengalaman pengguna yang lebih baik.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Pengoptimalan data dan penyetelan kueri adalah aspek penting efisiensi kinerja di penyimpanan data, karena berdampak pada kinerja dan responsivitas seluruh beban kerja cloud. Kueri yang tidak dioptimalkan dapat menghasilkan penggunaan dan kemacetan sumber daya yang lebih besar, yang mengurangi keseluruhan efisiensi sebuah penyimpanan data.

Pengoptimalan data mencakup beberapa teknik untuk memastikan penyimpanan dan akses data yang efisien. Hal ini juga membantu meningkatkan performa kueri di penyimpanan data. Strategi utama mencakup partisi data, kompresi data, dan denormalisasi data, yang membantu data dioptimalkan untuk penyimpanan dan akses.

Langkah implementasi

- Pahami dan analisis kueri data penting yang dilakukan di penyimpanan data Anda.
- Identifikasi kueri lambat di penyimpanan data Anda dan gunakan rencana kueri untuk memahami statusnya saat ini.
  - Menganalisis rencana kueri di Amazon Redshift

    - Menggunakan EXPLAIN dan EXPLAIN ANALYZE di Athena
- Terapkan strategi untuk meningkatkan kinerja kueri. Beberapa strategi utamanya meliputi:
    - Menggunakan format file kolom (seperti Parquet atau ORC).
    - Mengompresi data di penyimpanan data untuk mengurangi ruang penyimpanan dan operasi I/O.
    - Partisi data untuk membagi data menjadi bagian-bagian yang lebih kecil dan mengurangi waktu pemindaian data.
        - Mempartisi data di Athena
        - Partisi dan distribusi data
    - Pengindeksan data pada kolom umum dalam kueri.
    - Pilih operasi gabungan yang tepat untuk kueri. Saat Anda menggabungkan dua tabel, tentukan tabel yang lebih besar di sisi kiri gabungan dan tabel yang lebih kecil di sisi kanan gabungan.
    - Solusi caching terdistribusi untuk meningkatkan latensi dan mengurangi jumlah operasi I/O basis data.
    - Pemeliharaan rutin seperti mengeksekusi statistik.
- Lakukan eksperimen dan uji strategi di lingkungan nonproduksi.

Sumber daya

Dokumen terkait:

- Praktik terbaik Amazon Aurora
- Performa Amazon Redshift
- 10 kiat performa terbaik Amazon Athena
- Caching Basis Data AWS
- Praktik Terbaik untuk Mengimplementasikan Amazon ElastiCache
- Mempartisi data di Athena

Video terkait:

- Optimalkan Pola Data menggunakan Pembagian Data Amazon Redshift
- Mengoptimalkan Kueri Amazon Athena dengan Alat Analisis Kueri Baru

Contoh terkait:

- Driver CSI Amazon EFS

PERF03-BP05 Mengimplementasikan pola akses data yang memanfaatkan caching

Implementasikan pola akses yang dapat memanfaatkan caching data untuk pengambilan cepat data yang sering diakses.

Antipola umum:

- Anda menyimpan cache data yang sering berubah.
- Anda mengandalkan data dalam cache seolah-olah data tersebut disimpan dengan durabilitas tinggi dan selalu tersedia.
- Anda tidak mempertimbangkan konsistensi data cache Anda.
- Anda tidak memantau efisiensi implementasi caching Anda.

Manfaat menjalankan praktik terbaik ini: Menyimpan data dalam cache dapat meningkatkan latensi baca, throughput baca, pengalaman pengguna, dan efisiensi secara keseluruhan, serta mengurangi biaya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Cache adalah komponen perangkat lunak atau perangkat keras yang dimaksudkan untuk menyimpan data sehingga permintaan di masa mendatang untuk data yang sama dapat dilayani lebih cepat atau lebih efisien. Data yang disimpan dalam cache dapat direkonstruksi jika hilang dengan mengulangi perhitungan sebelumnya atau mengambilnya dari tempat penyimpanan data lain.

Caching data dapat menjadi salah satu strategi paling efektif untuk meningkatkan performa aplikasi Anda secara keseluruhan dan mengurangi beban pada sumber data primer yang mendasarinya. Data dapat di-cache di berbagai tingkatan dalam aplikasi, seperti di dalam aplikasi yang membuat panggilan jarak jauh, yang dikenal sebagai caching sisi klien, atau dengan menggunakan layanan sekunder cepat untuk menyimpan data, yang dikenal sebagai caching jarak jauh.

Caching sisi klien

Dengan caching sisi klien, setiap klien (aplikasi atau layanan yang mengkueri penyimpanan data backend) dapat menyimpan hasil kueri unik mereka secara lokal selama jangka waktu tertentu.

Hal ini dapat mengurangi jumlah permintaan di seluruh jaringan ke sebuah penyimpanan data dengan memeriksa cache klien lokal terlebih dahulu. Jika hasilnya tidak ada, aplikasi kemudian dapat mengkueri penyimpanan data tersebut dan menyimpan hasilnya secara lokal. Dengan pola ini, setiap klien dapat menyimpan data di lokasi terdekat (klien itu sendiri), sehingga menghasilkan latensi serendah mungkin. Klien juga dapat terus melayani beberapa kueri ketika penyimpanan data backend tidak tersedia, sehingga meningkatkan ketersediaan sistem secara keseluruhan.

Salah satu kelemahan pendekatan ini adalah ketika ada beberapa klien yang dilibatkan, semuanya dapat menyimpan data cache yang sama secara lokal. Hal ini mengakibatkan penggunaan penyimpanan duplikat dan inkonsistensi data antara klien-klien tersebut. Salah satu klien mungkin melakukan caching hasil kueri, dan satu menit kemudian klien lainnya dapat menjalankan kueri yang sama dan mendapatkan hasil yang berbeda.

Caching jarak jauh

Untuk mengatasi masalah duplikat data antarklien, layanan eksternal yang cepat, atau cache jarak jauh, dapat digunakan untuk menyimpan data yang dikueri. Alih-alih memeriksa penyimpanan data lokal, setiap klien akan memeriksa cache jarak jauh sebelum mengkueri penyimpanan data backend. Strategi ini memungkinkan respons yang lebih konsisten antarklien, efisiensi yang lebih baik pada data yang disimpan, dan volume data cache yang lebih tinggi karena ruang penyimpanannya diskalakan tanpa terikat klien.

Kelemahan cache jarak jauh adalah keseluruhan sistem mungkin mengalami latensi yang lebih tinggi karena diperlukan lompatan jaringan tambahan untuk memeriksa cache jarak jauh. Caching sisi klien dapat digunakan bersama caching jarak jauh untuk caching multitingkat sehingga dapat memperbaiki latensi.

Langkah implementasi

1. Identifikasikan basis data, API, dan layanan jaringan yang dapat memanfaatkan caching. Layanan yang memiliki beban kerja baca yang berat, memiliki rasio baca-tulis yang tinggi, atau mahal untuk diskalakan dapat memanfaatkan caching.
   - Caching Basis Data
   - Mengaktifkan caching API untuk meningkatkan responsivitas
2. Identifikasikan jenis strategi caching yang tepat yang paling sesuai dengan pola akses Anda.
   - Strategi caching
   - Solusi Caching AWS
3. Ikuti Praktik Terbaik Caching untuk penyimpanan data Anda.

4. Konfigurasikan strategi pembatalan cache, seperti time-to-live (TTL), untuk semua data yang menyeimbangkan kesegaran data dan mengurangi tekanan pada penyimpanan data backend.
5. Aktifkan fitur seperti percobaan ulang koneksi otomatis, mundur eksponensial, batas waktu sisi klien, dan pooling koneksi di dalam klien, jika tersedia, karena fitur-fitur tersebut dapat meningkatkan performa dan keandalan.
   - Praktik terbaik: Klien Redis dan Amazon ElastiCache (Redis OSS)
6. Pantau laju hit cache dengan target 80% atau lebih tinggi. Nilai yang lebih rendah mungkin menunjukkan ukuran cache yang tidak mencukupi atau pola akses yang tidak diuntungkan dengan caching.
   - Metrik mana yang harus saya pantau?
   - Praktik terbaik untuk memantau beban kerja Redis di Amazon ElastiCache
   - Praktik terbaik pemantauan dengan Amazon ElastiCache (Redis OSS) menggunakan Amazon CloudWatch
7. Implementasikan replikasi data untuk melimpahkan baca ke beberapa instans dan meningkatkan performa dan ketersediaan pembacaan data.

Sumber daya

Dokumen terkait:

- Menggunakan Amazon ElastiCache Well-Architected Lens
- Praktik terbaik pemantauan dengan Amazon ElastiCache (Redis OSS) menggunakan Amazon CloudWatch
- Metrik Mana yang Harus Saya Pantau?
- Laporan resmi Performa pada Skala Besar dengan Amazon ElastiCache
- Tantangan dan strategi caching

Video terkait:

- Jalur Pembelajaran Amazon ElastiCache
- Desain untuk keberhasilan dengan praktik terbaik Amazon ElastiCache

Contoh terkait:

- Meningkatkan performa basis data MySQL dengan strategi pemberian tag Amazon ElastiCache (Redis OSS)

Jaringan dan pengiriman konten

PERF 4. Bagaimana cara memilih dan mengonfigurasi sumber daya jaringan dalam beban kerja Anda?

Solusi basis data paling efektif untuk sistem bervariasi berdasarkan persyaratan untuk ketersediaan, konsistensi, dan toleransi partisi, latensi, daya tahan, skalabilitas, dan kemampuan kueri. Banyak sistem menggunakan solusi basis data yang berlainan untuk berbagai subsistem, dan mengaktifkan fitur yang berlainan untuk meningkatkan performa. Memilih fitur untuk sistem dan solusi basis data yang salah dapat mengakibatkan efisiensi performa yang lebih rendah.

Praktik terbaik

- PERF04-BP01 Memahami bagaimana jaringan memengaruhi performa
- PERF04-BP02 Mengevaluasi fitur jaringan yang tersedia
- PERF04-BP03 Memilih konektivitas khusus atau VPN yang tepat untuk beban kerja Anda
- PERF04-BP04 Menggunakan penyeimbangan beban untuk mendistribusikan lalu lintas di berbagai sumber daya
- PERF04-BP05 Memilih protokol jaringan untuk meningkatkan performa
- PERF04-BP06 Memilih lokasi beban kerja Anda berdasarkan kebutuhan jaringan
- PERF04-BP07 Mengoptimalkan konfigurasi jaringan berdasarkan metrik

PERF04-BP01 Memahami bagaimana jaringan memengaruhi performa

Analisis dan pahami bagaimana keputusan terkait jaringan memengaruhi beban kerja Anda untuk memberikan performa yang efisien dan pengalaman pengguna yang lebih baik.

Antipola umum:

- Semua lalu lintas mengalir melalui pusat data Anda.
- Anda merutekan semua lalu lintas melalui firewall pusat, bukan menggunakan alat keamanan jaringan cloud-native.
- Anda menyediakan koneksi AWS Direct Connect tanpa memahami persyaratan penggunaan aktual.

- Anda tidak mempertimbangkan karakteristik beban kerja dan biaya overhead enkripsi ketika menentukan solusi jaringan Anda.
- Anda menggunakan konsep dan strategi on-premise untuk solusi jaringan di cloud.

Manfaat menjalankan praktik terbaik ini: Memahami bagaimana jaringan memengaruhi kinerja beban kerja membantu Anda mengidentifikasi potensi hambatan, meningkatkan pengalaman pengguna, meningkatkan keandalan, dan menurunkan pemeliharaan operasional saat beban kerja berubah.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Jaringan bertanggung jawab atas konektivitas antara komponen aplikasi, layanan cloud, jaringan edge, dan data on-premise, oleh karena itu, jaringan dapat sangat memengaruhi performa beban kerja. Selain performa beban kerja, pengalaman pengguna juga dapat terpengaruh oleh latensi jaringan, bandwidth, protokol, lokasi, kemacetan jaringan, jitter, throughput, dan aturan perutean.

Miliki daftar terdokumentasi kebutuhan jaringan dari beban kerja termasuk latensi, ukuran paket, aturan perutean, protokol, dan pola lalu lintas pendukung. Tinjau solusi jaringan yang tersedia dan identifikasi layanan mana yang memenuhi karakteristik jaringan beban kerja Anda. Jaringan berbasis cloud dapat dengan cepat dibangun kembali, sehingga diperlukan peningkatan arsitektur jaringan Anda seiring berjalannya waktu untuk meningkatkan efisiensi kinerja.

Langkah Implementasi:

1. Tentukan dan dokumentasikan persyaratan performa jaringan, termasuk metrik seperti latensi jaringan, bandwidth, protokol, lokasi, pola lalu lintas (lonjakan dan frekuensi), throughput, enkripsi, inspeksi, dan aturan perutean.
2. Pelajari tentang layanan jaringan utama AWS seperti VPC, AWS Direct Connect, Elastic Load Balancing (ELB), dan Amazon Route 53.
3. Rekam karakteristik jaringan utama berikut:

Karakteristik	Alat dan metrik
Karakteristik jaringan dasar	• VPC Flow Logs • AWS Transit Gateway Flow Logs • Metrik AWS Transit Gateway

Karakteristik	Alat dan metrik
	• Metrik AWS PrivateLink
Karakteristik jaringan aplikasi	• Elastic Fabric Adapter • Metrik AWS App Mesh • Metrik Amazon API Gateway
Karakteristik jaringan edge	• Metrik Amazon CloudFront • Metrik Amazon Route 53 • Metrik AWS Global Accelerator
Karakteristik jaringan hybrid	• Metrik Direct Connect • Metrik AWS Site-to-Site VPN • Metrik AWS Client VPN • Metrik AWS Cloud WAN
Karakteristik jaringan keamanan	• Metrik AWS Shield, AWS WAF, dan AWS Network Firewall
Karakteristik penelusuran	• AWS X-Ray • Reachability Analyzer VPC • Network Access Analyzer • Amazon Inspector • Amazon CloudWatch RUM

4. Buat tolok ukur dan uji kinerja jaringan:

    a. Buat tolok ukur throughput jaringan, karena beberapa faktor yang dapat memengaruhi kinerja jaringan Amazon EC2 saat instans berada di VPC yang sama. Ukur bandwidth jaringan antar instans Linux Amazon EC2 di VPC yang sama.

    b. Jalankan pengujian beban untuk bereksperimen dengan solusi dan opsi jaringan

Sumber daya

Dokumen terkait:

- Application Load Balancer
- Peningkatan Jaringan EC2 di Linux
- Jaringan yang Ditingkatkan EC2 di Windows
- Grup Penempatan EC2
- Memungkinkan Jaringan yang Ditingkatkan dengan Elastic Network Adapter (ENA) di Instans Linux
- Network Load Balancer
- Produk Jaringan dengan AWS
- Transit Gateway
- Beralih ke perutean berbasis latensi di Amazon Route 53
- Titik akhir VPC
- VPC Flow Logs

Video terkait:

- Konektivitas ke arsitektur jaringan AWS dan AWS hybrid
- Mengoptimalkan Performa Jaringan untuk Instans Amazon EC2
- Meningkatkan Kinerja Jaringan Global untuk Aplikasi
- Praktik Terbaik Instans EC2 dan Pengoptimalan Kinerja
- Mengoptimalkan Kinerja Jaringan untuk instans Amazon EC2
- Praktik terbaik dan tip jaringan dengan Well-Architected Framework
- Praktik terbaik jaringan AWS dalam migrasi skala besar

Contoh terkait:

- AWS Transit Gateway dan Solusi Keamanan yang Dapat Diskalakan
- Lokakarya Jaringan AWS

PERF04-BP02 Mengevaluasi fitur jaringan yang tersedia

Evaluasi fitur jaringan di cloud yang dapat meningkatkan kinerja. Ukur dampak fitur-fitur ini melalui pengujian, metrik, dan analisis. Misalnya, manfaatkan fitur tingkat jaringan yang tersedia untuk mengurangi latensi, jarak jaringan, atau masalah kecepatan (jitter).

Antipola umum:

- Anda hanya menggunakan satu Wilayah karena di sanalah lokasi fisik kantor pusat Anda.
- Anda menggunakan firewall, bukan grup keamanan, untuk memfilter lalu lintas.
- Anda lebih memilih melanggar TLS untuk pemeriksaan lalu lintas daripada mengandalkan grup keamanan, kebijakan titik akhir, dan fungsionalitas cloud-native lainnya.
- Anda hanya menggunakan segmentasi berbasis subnet, bukan grup keamanan.

Manfaat menjalankan praktik terbaik ini: Mengevaluasi semua fitur dan opsi layanan dapat meningkatkan performa beban kerja Anda, menurunkan biaya infrastruktur, mengurangi upaya yang diperlukan untuk memelihara beban kerja Anda, dan meningkatkan postur keamanan Anda secara keseluruhan. Anda dapat menggunakan backbone AWS global memberikan pengalaman jaringan yang optimal bagi pelanggan Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

AWS menawarkan layanan seperti AWS Global Accelerator dan Amazon CloudFront yang dapat membantu meningkatkan performa jaringan, sementara sebagian besar layanan AWS memiliki fitur produk (seperti fitur Amazon S3 Transfer Acceleration) untuk mengoptimalkan lalu lintas jaringan.

Tinjau opsi konfigurasi terkait jaringan mana yang tersedia untuk Anda serta bagaimana dampaknya terhadap beban kerja Anda. Optimalisasi performa bergantung pada pemahaman tentang bagaimana opsi-opsi ini berinteraksi dengan arsitektur Anda serta dampaknya terhadap performa terukur dan pengalaman pengguna.

Langkah implementasi

- Buat daftar komponen beban kerja.
  - Pertimbangkan untuk menggunakan AWS Cloud WAN untuk membangun, mengelola, dan memantau jaringan organisasi Anda saat membangun jaringan global terpadu.
  - Pantau jaringan global dan inti Anda dengan metrik Amazon CloudWatch Logs. Manfaatkan Amazon CloudWatch RUM, yang memberikan wawasan untuk membantu mengidentifikasi, memahami, dan menyempurnakan pengalaman digital pengguna.
  - Lihat latensi jaringan agregat antara Wilayah AWS dan Zona Ketersediaan, serta di dalam setiap Zona Ketersediaan, menggunakan AWS Network Manager untuk mendapatkan wawasan tentang bagaimana performa aplikasi Anda berkaitan dengan performa jaringan AWS yang mendasarinya.

  - Gunakan alat basis data manajemen konfigurasi (CMDB) yang ada atau layanan seperti AWS Config untuk membuat inventaris beban kerja Anda dan cara mengonfigurasinya.
- Jika ini adalah beban kerja yang ada, identifikasi dan dokumentasikan tolok ukur untuk metrik performa Anda, yang fokus pada hambatan dan area yang perlu ditingkatkan. Metrik jaringan terkait performa akan berbeda per beban kerja berdasarkan persyaratan bisnis dan karakteristik beban kerja. Sebagai permulaan, metrik ini mungkin penting untuk ditinjau untuk beban kerja Anda: bandwidth, latensi, kehilangan paket, jitter, dan transmisi ulang.
- Jika ini adalah beban kerja baru, lakukan pengujian beban untuk mengidentifikasi hambatan performa.
- Untuk hambatan performa yang Anda identifikasi, tinjau opsi konfigurasi untuk solusi Anda guna mengidentifikasi peluang peningkatan performa. Lihat opsi dan fitur jaringan utama berikut:

Peluang peningkatan	Solusi
Jalur atau rute jaringan	Gunakan Network Access Analyzer untuk mengidentifikasi jalur atau rute.
Protokol jaringan	Lihat PERF04-BP05 Memilih protokol jaringan untuk meningkatkan performa
Topologi jaringan	Evaluasi tarik ulur operasional dan performa Anda antara Peering VPC dan AWS Transit Gateway saat menghubungkan beberapa akun. AWS Transit Gateway menyederh anakan cara Anda menyambungkan silang semua VPC Anda, yang bisa saja berada di ribuan Akun AWS dan ke dalam jaringan on-premise. Bagikan AWS Transit Gateway Anda di antara beberapa akun menggunakan AWS Resource Access Manager. Lihat PERF04-BP03 Memilih konektivitas khusus atau VPN yang tepat untuk beban kerja Anda
Layanan jaringan	AWS Global Accelerator adalah layanan jaringan yang meningkatkan performa

Peluang peningkatan	Solusi
	lalu lintas pengguna Anda hingga 60% menggunakan infrastruktur jaringan global AWS. Amazon CloudFront dapat meningkatkan performa pengiriman konten beban kerja dan memperbaiki latensi Anda secara global. Gunakan Lambda@Edge untuk menjalank an fungsi yang menyesuaikan konten yang dikirimkan CloudFront lebih dekat ke pengguna, mengurangi latensi, dan meningkat kan performa. Amazon Route 53 menawarkan perutean berbasis latensi, perutean geolokasi, perutean geoproksimitas, dan perutean berbasis IP opsi untuk membantu Anda meningkatkan performa beban kerja Anda bagi audiens global. Identifikasikan opsi perutean mana yang akan mengoptimalkan performa beban kerja Anda dengan meninjau lalu lintas beban kerja dan lokasi pengguna Anda saat beban kerja Anda terdistribusi secara global.

Peluang peningkatan	Solusi
Fitur sumber daya penyimpanan	Amazon S3 Transfer Acceleration adalah fitur yang memungkinkan pengguna eksternal mendapatkan manfaat pengoptimalan jaringan dari CloudFront untuk mengungga h data ke Amazon S3. Hal ini meningkatkan kemampuan transfer data dalam jumlah besar dari lokasi jarak jauh yang tidak memiliki koneksi khusus ke AWS Cloud. Titik Akses Multi-Wilayah Amazon S3 mereplikasi konten ke beberapa Wilayah dan menyederhanakan beban kerja dengan menyediakan satu titik akses. Saat Titik Akses Multi-Wilayah digunakan, Anda dapat meminta atau menulis data ke Amazon S3 dengan layanan yang mengidentifikasi bucket latensi terendah.

Peluang peningkatan	Solusi
Fitur sumber daya komputasi	Antarmuka Jaringan Elastis (ENA) yang digunakan oleh instans Amazon EC2, kontainer, dan fungsi Lambda dibatasi berdasarkan per alur. Tinjau grup penempatan Anda untuk mengoptimalkan throughput jaringan EC2. Untuk menghindari kemacetan pada basis per alur, rancang aplikasi Anda sedemikian rupa hingga menggunakan beberapa alur. Untuk memantau dan mendapatkan visibilitas tentang metrik jaringan terkait komputasi Anda, gunakan CloudWatch Metrics dan ethtool. Perintah `ethtool` disertakan dalam driver ENA dan mengekspos metrik terkait jaringan tambahan yang dapat dipublikasikan sebagai metrik kustom ke CloudWatch. Amazon Elastic Network Adapters (ENA) memberikan pengoptimalan lebih lanjut dengan memberikan throughput yang lebih baik untuk instans Anda dalam grup penempatan klaster. Elastic Fabric Adapter (EFA) adalah antarmuka jaringan untuk instans Amazon EC2 yang memungkinkan Anda menjalankan beban kerja yang memerlukan komunikasi antarsimpul tingkat tinggi dalam skala besar di AWS. Instans yang dioptimalkan Amazon EBS menggunakan tumpukan konfigurasi yang dioptimalkan dan menyediakan kapasitas khusus tambahan untuk meningkatkan I/O Amazon EBS.

Sumber daya

Dokumen terkait:

- Instans yang Dioptimalkan Amazon EBS
- Application Load Balancer
- Peningkatan Jaringan EC2 di Linux
- Jaringan yang Ditingkatkan EC2 di Windows
- Grup Penempatan EC2
- Memungkinkan Jaringan yang Ditingkatkan dengan Elastic Network Adapter (ENA) di Instans Linux
- Network Load Balancer
- Produk Jaringan dengan AWS
- AWS Transit Gateway
- Beralih ke Perutean Berbasis Latensi di Amazon Route 53
- Titik Akhir VPC
- VPC Flow Logs

Video terkait:

- Konektivitas ke arsitektur jaringan AWS dan AWS hybrid
- Mengoptimalkan Performa Jaringan untuk Instans Amazon EC2
- AWS Global Accelerator

Contoh terkait:

- AWS Transit Gateway dan Solusi Keamanan yang Dapat Diskalakan
- Lokakarya Jaringan AWS

PERF04-BP03 Memilih konektivitas khusus atau VPN yang tepat untuk beban kerja Anda

Ketika diperlukan konektivitas hybrid untuk menghubungkan sumber daya on-premise dan cloud, sediakan bandwidth yang memadai untuk memenuhi persyaratan performa Anda. Perkirakan persyaratan bandwidth dan latensi untuk beban kerja hybrid Anda. Angka-angka ini akan mendorong persyaratan penyesuaian ukuran Anda.

Antipola umum:

- Anda hanya mengevaluasi solusi VPN untuk persyaratan enkripsi jaringan Anda.
- Anda tidak mengevaluasi opsi cadangan atau konektivitas redundan.
- Anda tidak mengidentifikasi semua persyaratan beban kerja (kebutuhan enkripsi, protokol, bandwidth, dan lalu lintas).

Manfaat menjalankan praktik terbaik ini: Memilih dan mengonfigurasi solusi konektivitas yang tepat akan meningkatkan keandalan beban kerja dan memaksimalkan performa. Dengan mengidentifikasi persyaratan beban kerja, membuat perencanaan ke depan, dan mengevaluasi solusi hybrid, Anda dapat meminimalkan perubahan jaringan fisik yang mahal dan biaya operasional sekaligus meningkatkan kecepatan perolehan nilai.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Kembangkan arsitektur jaringan hybrid berdasarkan persyaratan bandwidth Anda. Direct Connect memungkinkan Anda untuk menghubungkan jaringan on-premise Anda secara privat dengan AWS. Layanan ini ideal ketika Anda memerlukan bandwidth tinggi dan latensi rendah sambil mencapai performa yang konsisten. Sambungan VPN membangun sambungan yang aman lewat internet. VPN digunakan ketika yang diperlukan hanyalah sambungan sementara, ketika biaya menjadi pertimbangan, atau sebagai kontingensi sambil menunggu terbentuknya konektivitas jaringan fisik yang kuat saat menggunakan Direct Connect.

Jika persyaratan bandwidth Anda tinggi, Anda dapat mempertimbangkan beberapa layanan Direct Connect atau VPN. Lalu lintas dapat diberikan penyeimbangan beban di seluruh layanan, meskipun kami tidak merekomendasikan penyeimbangan beban antara Direct Connect dan VPN dikarenakan perbedaan latensi dan bandwidth.

Langkah implementasi

1. Perkirakan persyaratan bandwidth dan latensi aplikasi yang sudah Anda miliki.
    a. Untuk beban kerja lama yang akan beralih ke AWS, manfaatkan data dari sistem pemantauan jaringan internal Anda.
    b. Untuk beban kerja baru atau lama yang data pemantauannya tidak Anda miliki, hubungi pemilik produk untuk menentukan metrik performa yang memadai dan memberikan pengalaman pengguna yang baik.

2. Pilih sambungan khusus atau VPN sebagai opsi konektivitas Anda. Berdasarkan semua persyaratan beban kerja Anda (kebutuhan enkripsi, bandwidth, dan lalu lintas), Anda dapat memilih AWS Direct Connect atau Site-to-Site VPN (atau keduanya). Diagram berikut dapat membantu Anda memilih jenis sambungan yang tepat.
    a. AWS Direct Connect menyediakan konektivitas khusus ke lingkungan AWS, mulai dari 50 Mbps hingga 100 Gbps, menggunakan sambungan khusus atau sambungan yang di-host. Layanan ini memberi Anda latensi yang terkelola dan terkontrol serta bandwidth yang tersedia agar beban kerja Anda dapat terhubung ke lingkungan lain secara efisien. Dengan menggunakan partner AWS Direct Connect, Anda dapat memiliki konektivitas menyeluruh dari beberapa lingkungan, yang memberikan jaringan lebih luas dengan performa konsisten. AWS menawarkan bandwidth sambungan langsung dengan penskalaan menggunakan 100 Gbps native, link aggregation group (LAG), atau BGP equal-cost multipath (ECMP).
    b. AWS Site-to-Site VPN memberikan layanan VPN terkelola yang mendukung keamanan protokol internet (IPsec). Ketika sambungan VPN dibuat, setiap sambungan VPN mencakup dua terowongan untuk ketersediaan tinggi.
3. Ikuti dokumentasi AWS untuk memilih opsi konektivitas yang tepat:
    a. Jika Anda memutuskan untuk menggunakanDirect Connect, pilih bandwidth yang sesuai untuk konektivitas Anda.
    b. Jika Anda menggunakan AWS Site-to-Site VPN di beberapa lokasi untuk terhubung ke Wilayah AWS, gunakan sambungan Site-to-Site VPN yang dipercepat untuk mendapatkan peluang peningkatan performa jaringan.
    c. Jika desain jaringan Anda terdiri dari koneksi VPN IPsec lewat AWS Direct Connect, pertimbangkan menggunakan VPN IP Privat untuk meningkatkan keamanan dan mencapai segmentasi. VPN IP Privat Site-to-Site AWS di-deploy di atas antarmuka virtual transit (VIF).
    d. AWS Direct Connect SiteLink memungkinkan pembuatan koneksi latensi rendah dan redundan antara semua pusat data Anda di seluruh dunia dengan mengirimkan data melalui jalur tercepat antara lokasi-lokasi AWS Direct Connect, dengan melewati Wilayah AWS.
4. Lakukan validasi penyiapan konektivitas Anda sebelum deployment ke produksi. Lakukan pengujian keamanan dan performa untuk memastikan persyaratan bandwidth, keandalan, latensi, dan kepatuhan Anda terpenuhi.
5. Pantau performa dan penggunaan konektivitas Anda secara teratur dan optimalkan jika diperlukan.

Bagan alur performa penentu.

Sumber daya

Dokumen terkait:

- Network Load Balancer
- Produk Jaringan dengan AWS
- AWS Transit Gateway
- Beralih ke Perutean berbasis latensi di Amazon Route 53
- Titik Akhir VPC
- Site-to-Site VPN
- Membangun Infrastruktur Jaringan AWS Multi-VPC yang Aman dan Dapat Diskalakan
- Direct Connect
- Client VPN

Video terkait:

- Konektivitas ke arsitektur jaringan AWS dan AWS hybrid
- Mengoptimalkan Performa Jaringan untuk Instans Amazon EC2
- AWS Global Accelerator
- Direct Connect
- AWS Transit Gateway Connect
- Solusi VPN
- Keamanan dengan Solusi VPN

Contoh terkait:

- AWS Transit Gateway dan Solusi Keamanan yang Dapat Diskalakan
- Lokakarya Jaringan AWS

PERF04-BP04 Menggunakan penyeimbangan beban untuk mendistribusikan lalu lintas di berbagai sumber daya

Distribusikan lalu lintas di berbagai sumber daya atau layanan untuk memanfaatkan elastisitas yang ada di cloud untuk beban kerja Anda. Anda juga dapat menggunakan penyeimbang beban untuk

memindahkan beban penghentian enkripsi guna meningkatkan performa dan keandalan serta untuk mengelola dan merutekan lalu lintas secara efektif.

Antipola umum:

- Anda tidak mempertimbangkan persyaratan beban kerja Anda ketika memilih jenis penyeimbang beban.
- Anda tidak memanfaatkan fitur penyeimbang beban untuk mengoptimalkan performa.
- Beban kerja terpapar langsung ke internet tanpa penyeimbang beban.
- Anda merutekan semua lalu lintas internet melalui penyeimbang beban yang ada.
- Anda menggunakan penyeimbangan beban TCP umum dan membuat setiap simpul komputasi menangani enkripsi SSL.

Manfaat menjalankan praktik terbaik ini: Penyeimbang beban menangani berbagai beban lalu lintas aplikasi Anda dalam satu atau beberapa Zona Ketersediaan dan menghadirkan ketersediaan yang tinggi, penskalaan otomatis, dan pemanfaatan yang lebih baik untuk beban kerja Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Penyeimbang beban berfungsi sebagai titik masuk untuk beban kerja Anda, yakni titik asal penyeimbang beban mendistribusikan lalu lintas ke target backend Anda, seperti kontainer atau instans komputasi, untuk meningkatkan pemanfaatan.

Memilih jenis penyeimbang beban yang tepat adalah langkah pertama untuk mengoptimalkan arsitektur Anda. Mulai dengan mencantumkan karakteristik beban kerja Anda, seperti protokol (misalnya, TCP, HTTP, TLS, atau WebSockets), jenis target (seperti instans, kontainer, atau nirserver), persyaratan aplikasi (seperti sambungan berdurasi lama, autentikasi pengguna, atau keeratan), dan penempatan (seperti Wilayah, Zona Lokal, Outpost, atau isolasi zona).

AWS menyediakan beberapa model untuk aplikasi Anda untuk menggunakan penyeimbangan beban. Application Load Balancer sangat ideal untuk menyeimbangkan beban lalu lintas HTTP dan HTTPS dan menyediakan perutean permintaan lanjutan yang ditargetkan pada pengiriman arsitektur aplikasi modern, termasuk layanan mikro dan kontainer.

Network Load Balancer sangat ideal untuk menyeimbangkan beban lalu lintas TCP yang memerlukan kinerja ekstrem. Penyeimbang beban ini mampu menangani jutaan permintaan per detik sekaligus

membuat latensi tetap rendah, serta dioptimalkan untuk menangani pola lalu lintas yang tidak stabil dan mendadak.

Elastic Load Balancing menyediakan manajemen sertifikat terintegrasi dan dekripsi SSL/TLS, memberikan fleksibilitas kepada Anda untuk mengelola pengaturan SSL penyeimbang beban secara terpusat serta memindahkan beban yang banyak menggunakan CPU dari beban kerja Anda.

Setelah memilih penyeimbang beban yang tepat, Anda dapat mulai memanfaatkan fitur-fiturnya untuk mengurangi jumlah upaya yang harus dilakukan backend guna melayani lalu lintas.

Contohnya, dengan menggunakan Application Load Balancer (ALB) dan Network Load Balancer (NLB), Anda dapat melakukan pemindahan beban enkripsi SSL/TLS, yang merupakan peluang untuk menghindari handshake TLS yang sarat CPU diselesaikan oleh target Anda dan juga untuk meningkatkan manajemen sertifikat.

Ketika Anda mengonfigurasi pemindahan beban SSL/TLS di penyeimbang beban, penyeimbang beban menjadi bertanggung jawab atas enkripsi lalu lintas dari dan ke klien sekaligus memberikan lalu lintas tidak terenkripsi ke backend Anda, sehingga membebaskan sumber daya backend Anda dan meningkatkan waktu respons untuk klien.

Application Load Balancer juga dapat melayani lalu lintas HTTP/2 tanpa harus mendukungnya di target Anda. Keputusan sederhana ini dapat meningkatkan waktu respons aplikasi Anda, karena HTTP/2 menggunakan sambungan TCP dengan lebih efisien.

Persyaratan latensi beban kerja Anda harus dipertimbangkan ketika menentukan arsitektur. Sebagai contoh, jika Anda memiliki aplikasi yang sensitif latensi, Anda dapat memutuskan untuk menggunakan Network Load Balancer, yang menawarkan latensi yang sangat rendah. Alternatifnya, Anda dapat memutuskan untuk membawa beban kerja lebih dekat ke pelanggan dengan memanfaatkan Application Load Balancer di Zona Lokal AWS atau bahkan AWS Outposts.

Pertimbangan lain untuk beban kerja yang sensitif latensi adalah penyeimbangan beban lintas zona. Dengan penyeimbangan beban lintas zona, setiap simpul penyeimbang beban mendistribusikan lalu lintas ke target terdaftar di semua Zona Ketersediaan yang diaktifkan.

Gunakan Auto Scaling yang terintegrasi dengan penyeimbang beban Anda. Salah satu aspek penting dari sistem dengan performa yang efisien berkaitan dengan penyesuaian ukuran sumber daya backend Anda. Untuk melakukannya, Anda dapat memanfaatkan integrasi penyeimbang beban untuk sumber daya target backend. Dengan menggunakan integrasi penyeimbang beban dengan grup Auto Scaling, target akan ditambahkan atau disingkirkan dari penyeimbang beban sebagaimana

diperlukan untuk merespons lalu lintas masuk. Penyeimbang beban juga dapat diintegrasikan dengan Amazon ECS dan Amazon EKS untuk beban kerja dalam kontainer.

- Amazon ECS - Penyeimbangan beban layanan
- Penyeimbangan beban aplikasi di Amazon EKS
- Penyeimbangan beban jaringan di Amazon EKS

Langkah implementasi

- Tentukan persyaratan penyeimbangan beban Anda termasuk volume yang sangat besar, ketersediaan, dan skalabilitas aplikasi.
- Pilih jenis penyeimbang beban yang tepat untuk aplikasi Anda.
  - Gunakan Application Load Balancer untuk beban kerja HTTP/HTTPS.
  - Gunakan Network Load Balancer untuk beban kerja non-HTTP yang dijalankan di TCP atau UDP.
  - Gunakan kombinasi keduanya (ALB sebagai target NLB) jika Anda ingin memanfaatkan fitur kedua produk. Contohnya, Anda dapat melakukan hal ini jika Anda ingin menggunakan IP statis NLB bersama dengan perutean berbasis header HTTP dari ALB, atau jika Anda ingin memaparkan beban kerja HTTP Anda ke AWS PrivateLink.
  - Untuk melihat perbandingan lengkap penyeimbang beban, lihat perbandingan produk ELB.
- Gunakan pemindahan beban SSL/TLS jika memungkinkan.
  - Konfigurasikan pendengar HTTPS/TLS dengan Application Load Balancer dan Network Load Balancer yang diintegrasikan dengan AWS Certificate Manager.
  - Perhatikan, beberapa beban kerja mungkin memerlukan enkripsi menyeluruh karena alasan kepatuhan. Jika demikian, enkripsi wajib diaktifkan di target.
  - Untuk praktik terbaik keamanan, lihat SEC09-BP02 Menerapkan enkripsi data bergerak.
- Pilih algoritma perutean yang tepat (khusus ALB).
  - Algoritma perutean dapat membuat perbedaan tentang seberapa baik target backend Anda digunakan, oleh karena itu juga membuat perbedaan dalam dampaknya pada performa. Misalnya, ALB menyediakan dua opsi untuk algoritma perutean:
  - Permintaan tertunda paling sedikit: Gunakan untuk mendapatkan distribusi beban yang lebih baik ke target backend Anda untuk kasus ketika permintaan untuk aplikasi Anda bervariasi dalam tingkat kompleksitas atau target Anda bervariasi dalam kemampuan pemrosesannya.

    - Round robin: Gunakan ketika permintaan dan target serupa, atau jika Anda harus mendistribusikan permintaan sama rata di antara target.
- Pertimbangkan isolasi zona atau lintas zona.
    - Gunakan penonaktifan lintas zona (isolasi zona) untuk peningkatan latensi dan domain kegagalan zona. Opsi ini dinonaktifkan secara default di NLB dan di ALB, Anda dapat menonaktifkannya per grup target.
    - Gunakan pengaktifan lintas zona untuk peningkatan ketersediaan dan fleksibilitas. Secara default, lintas zona diaktifkan untuk ALB dan di NLB, Anda dapat mengaktifkannya per grup target.
- Aktifkan keep-alive HTTP untuk beban kerja HTTP Anda (khusus ALB). Dengan fitur ini, penyeimbang beban dapat menggunakan ulang sambungan backend sampai waktu tetap aktif habis, sehingga meningkatkan waktu respons dan permintaan HTTP Anda serta mengurangi pemanfaatan sumber daya di target backend Anda. Untuk detail cara melakukan hal ini untuk Apache dan Nginx, lihat Apa saja pengaturan yang optimal untuk menggunakan Apache atau NGINX sebagai server backend untuk ELB?
- Aktifkan pemantauan untuk penyeimbang beban Anda.
    - Aktifkan log akses untuk Application Load Balancer dan Network Load Balancer.
    - Bidang utama yang harus dipertimbangkan untuk ALB adalah `request_processing_time`, `request_processing_time`, dan `response_processing_time`.
    - Bidang utama yang harus dipertimbangkan untuk NLB adalah `connection_time` dan `tls_handshake_time`.
    - Bersiaplah untuk melakukan kueri log ketika Anda memerlukannya. Anda dapat menggunakan Amazon Athena untuk mengkueri log ALB dan log NLB.
    - Buat alarm untuk metrik yang terkait dengan performa seperti `TargetResponseTime` untuk ALB..

Sumber daya

Dokumen terkait:

- perbandingan produk ELB
- Infrastruktur Global AWS
- Meningkatkan Performa dan Mengurangi Biaya Menggunakan Afinitas Zona Ketersediaan

- Langkah demi langkah untuk Analisis Log dengan Amazon Athena
- Mengkueri log Application Load Balancer
- Memantau Application Load Balancers Anda
- Memantau Network Load Balancer Anda
- Menggunakan Elastic Load Balancing untuk mendistribusikan lalu lintas ke seluruh instans di grup Auto Scaling Anda

Video terkait:

- AWS re:Invent 2018: Elastic Load Balancing: Pembahasan Mendalam dan Praktik Terbaik
- AWS re:Invent 2021 - Cara memilih penyeimbang beban yang tepat untuk beban kerja AWS Anda
- AWS re:Inforce 2022 - Cara menggunakan Elastic Load Balancing untuk meningkatkan postur keamanan Anda dalam skala besar
- AWS re:Invent 2019: Mendapatkan hasil maksimal dari Elastic Load Balancing untuk berbagai beban kerja

Contoh terkait:

- Sampel CDK dan CloudFormation untuk Analisis Log dengan Amazon Athena

PERF04-BP05 Memilih protokol jaringan untuk meningkatkan performa

Buat keputusan terkait protokol untuk komunikasi antara sistem dan jaringan berdasarkan dampaknya terhadap kinerja beban kerja.

Ada hubungan antara latensi dan bandwidth untuk mencapai throughput. Jika transfer file Anda menggunakan Transmission Control Protocol (TCP), latensi yang lebih tinggi kemungkinan besar akan mengurangi throughput secara keseluruhan. Ada pendekatan untuk memperbaiki hal ini dengan penyesuaian TCP dan pengoptimalan protokol transfer, tetapi salah satu solusinya adalah menggunakan User Datagram Protocol (UDP)).

Antipola umum:

- Anda menggunakan TCP untuk semua beban kerja tanpa memperhatikan persyaratan kinerjanya.

Manfaat menjalankan praktik terbaik ini: Memverifikasi bahwa protokol yang tepat telah digunakan untuk komunikasi antara pengguna dan komponen beban kerja akan membantu meningkatkan pengalaman pengguna secara keseluruhan untuk aplikasi Anda. Misalnya, UDP tanpa koneksi memungkinkan kecepatan tinggi, tetapi tidak menawarkan transmisi ulang atau keandalan tinggi. TCP adalah protokol berfitur lengkap, tetapi memerlukan biaya tambahan yang lebih besar untuk memproses paket.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

Jika Anda memiliki kemampuan untuk memilih protokol yang berbeda-beda untuk aplikasi Anda dan Anda memiliki keahlian di bidang ini, optimalkan aplikasi dan pengalaman pengguna akhir Anda dengan menggunakan protokol yang berbeda. Perhatikan bahwa pendekatan ini memiliki tingkat kesulitan yang tinggi dan hanya boleh dicoba jika Anda telah mengoptimalkan aplikasi Anda dengan cara lain terlebih dahulu.

Pertimbangan utama dalam meningkatkan performa beban kerja Anda yakni pemahaman persyaratan latensi dan throughput, lalu pemilihan protokol jaringan yang mengoptimalkan performa.

Kapan penggunaan TCP harus dipertimbangkan

TCP memberikan pengiriman data yang andal, dan dapat digunakan untuk komunikasi antara komponen beban kerja di mana keandalan dan jaminan pengiriman data merupakan hal yang penting. Banyak aplikasi berbasis web mengandalkan protokol berbasis TCP, seperti HTTP dan HTTPS, untuk membuka soket TCP untuk komunikasi antara komponen-komponen aplikasi. Email dan transfer data file adalah penerapan umum yang juga menggunakan TCP, karena ini adalah mekanisme transfer yang sederhana dan andal antara komponen-komponen aplikasi. Menggunakan TLS dengan TCP dapat menambahkan beberapa overhead ke komunikasi, yang dapat mengakibatkan peningkatan latensi dan pengurangan throughput, tetapi memiliki keunggulan dari segi keamanan. Overhead ini terutama berasal dari penambahan overhead proses handshake, yang dapat memerlukan beberapa perjalanan pulang pergi agar selesai. Setelah handshake selesai, overhead enkripsi dan dekripsi data relatif kecil.

Kapan penggunaan UDP harus dipertimbangkan

UDP adalah protokol dengan orientasi nirkoneksi, oleh karena itu, cocok untuk aplikasi yang membutuhkan transmisi cepat dan efisien, seperti data VoIP, pemantauan, dan log. Selain itu, pertimbangkan untuk menggunakan UDP jika Anda memiliki komponen beban kerja yang merespons

kueri kecil dari banyak klien untuk memastikan performa beban kerja yang optimal. Keamanan Lapisan Pengangkutan Datagram (DTLS) merupakan ekuivalen UDP untuk Keamanan Lapisan Pengangkutan (TLS). Ketika menggunakan DTLS dengan UDP, overhead berasal dari enkripsi dan dekripsi data, karena proses handshake disederhanakan. DTLS juga menambahkan sejumlah kecil overhead ke paket UDP, karena mencakup bidang tambahan untuk menunjukkan parameter keamanan dan untuk mendeteksi gangguan.

Kapan penggunaan SRD harus dipertimbangkan

Scalable reliable datagram (SRD) adalah protokol transpor jaringan yang dioptimalkan untuk beban kerja throughput tinggi karena kemampuannya untuk menjalankan lalu lintas penyeimbang beban melintasi beberapa jalur dan pulih dengan cepat dari penurunan paket atau kegagalan tautan. Oleh karena itu, SRD paling sesuai digunakan untuk beban kerja komputasi performa tinggi (HPC) yang memerlukan komunikasi latensi rendah dan throughput tinggi antara simpul komputasi. Hal ini dapat mencakup tugas pemrosesan paralel seperti simulasi, pemodelan, dan analisis data yang melibatkan banyak transfer data antara simpul.

Langkah implementasi

1. Gunakan AWS Global Accelerator dan AWS Transfer Family untuk memperbaiki throughput aplikasi transfer file online Anda. Layanan AWS Global Accelerator membantu Anda mendapatkan latensi lebih rendah antara perangkat klien dan beban kerja Anda di AWS. Dengan AWS Transfer Family, Anda dapat menggunakan protokol berbasis TCP seperti Secure Shell File Transfer Protocol (SFTP) dan File Transfer Protocol over SSL (FTPS) untuk menskalakan dengan aman dan mengelola transfer file ke layanan penyimpanan AWS.
2. Gunakan latensi jaringan untuk menentukan apakah TCP sesuai untuk komunikasi antara komponen beban kerja. Jika latensi jaringan antara server dan aplikasi klien Anda tinggi, maka handshake tiga arah TCP dapat memerlukan beberapa waktu, sehingga memengaruhi responsivitas aplikasi Anda. Metrik seperti time to first byte (TTFB) dan round-trip time (RTT) dapat digunakan untuk mengukur latensi jaringan. Jika beban kerja Anda menyajikan konten dinamis kepada pengguna, pertimbangkan untuk menggunakan Amazon CloudFrontyang membuat sambungan persisten ke masing-masing asal konten dinamis untuk menyingkirkan waktu penyiapan sambungan yang akan memperlambat setiap permintaan klien.
3. Menggunakan TLS dengan TCP atau UDP dapat mengakibatkan peningkatan latensi dan pengurangan throughput untuk beban kerja Anda karena dampak enkripsi dan dekripsi. Untuk beban kerja tersebut, pertimbangkan pemindahan beban SSL/TLS di Elastic Load Balancing untuk meningkatkan performa beban kerja dengan mengizinkan penyeimbang beban menangani proses enkripsi dan dekripsi SSL/TLS, bukan menggunakan instans backend. Hal ini dapat membantu

mengurangi pemanfaatan CPU di instans backend, yang dapat meningkatkan performa dan kapasitas.

4. Gunakan Network Load Balancer (NLB) untuk melakukan deployment layanan yang mengandalkan protokol UDP, seperti autentikasi dan otorisasi, logging, DNS, IoT, dan media streaming, untuk meningkatkan performa dan keandalan beban kerja Anda. NLB mendistribusikan lalu lintas UDP masuk di beberapa target, sehingga Anda dapat menskalakan beban kerja secara horizontal, meningkatkan kapasitas, dan mengurangi overhead satu target.
5. Untuk beban kerja Komputasi Performa Tinggi (HPC) Anda, pertimbangkan untuk menggunakan fungsi Adaptor Jaringan Elastis (ENA) Ekspres yang menggunakan protokol SRD untuk meningkatkan performa jaringan dengan memberikan bandwidth satu aliran yang lebih tinggi (25 Gbps) dan latensi ekor lebih rendah (99,9 persentil) untuk lalu lintas jaringan antara instans EC2.
6. Gunakan Application Load Balancer (ALB) untuk mengarahkan dan menyeimbangkan beban lalu lintas gRPC (Remote Procedure Calls) Anda antara komponen beban kerja atau antara layanan dan klien gRPC. gRPC menggunakan protokol HTTP/2 berbasis TCP untuk transpor dan gRPC memberikan manfaat terkait performa, seperti jejak jaringan lebih ringan, kompresi, serialisasi biner yang efisien, dukungan untuk berbagai bahasa, dan streaming dua arah.

Sumber daya

Dokumen terkait:

- Instans yang Dioptimalkan Amazon EBS
- Application Load Balancer
- Peningkatan Jaringan EC2 di Linux
- Jaringan yang Ditingkatkan EC2 di Windows
- Grup Penempatan EC2
- Memungkinkan Jaringan yang Ditingkatkan dengan Elastic Network Adapter (ENA) di Instans Linux
- Network Load Balancer
- Produk Jaringan dengan AWS
- AWS Transit Gateway
- Beralih ke Perutean Berbasis Latensi di Amazon Route 53
- Titik Akhir VPC
- VPC Flow Logs

Video terkait:

- Konektivitas ke arsitektur jaringan AWS dan AWS hybrid
- Mengoptimalkan Performa Jaringan untuk Instans Amazon EC2

Contoh terkait:

- AWS Transit Gateway dan Solusi Keamanan yang Dapat Diskalakan
- Lokakarya Jaringan AWS

PERF04-BP06 Memilih lokasi beban kerja Anda berdasarkan kebutuhan jaringan

Evaluasi opsi untuk penempatan sumber daya guna mengurangi latensi jaringan dan meningkatkan throughput, yang memberikan pengalaman pengguna optimal dengan mengurangi beban halaman dan waktu transfer data.

Antipola umum:

- Anda menggabungkan semua sumber daya beban kerja ke dalam satu lokasi geografis.
- Anda memilih Wilayah terdekat dengan lokasi Anda tetapi tidak dekat dengan pengguna akhir beban kerja.

Manfaat menjalankan praktik terbaik ini: Pengalaman pengguna sangat dipengaruhi oleh latensi antara pengguna dan aplikasi Anda. Dengan menggunakan Wilayah AWS yang sesuai dan jaringan global privat AWS, Anda dapat mengurangi latensi dan memberikan pengalaman yang lebih baik kepada pengguna jarak jauh.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Sumber daya, seperti instans Amazon EC2, ditempatkan di Zona Ketersediaan di dalam Wilayah AWS, Zona Lokal AWS, AWS Outposts, atau zona AWS Wavelength . Pemilihan lokasi ini memengaruhi latensi jaringan dan throughput dari lokasi pengguna tertentu. Layanan edge seperti Amazon CloudFront dan AWS Global Accelerator juga dapat digunakan untuk meningkatkan performa jaringan dengan caching konten di lokasi edge atau memberikan kepada pengguna jalur optimal ke beban kerja melalui jaringan global AWS.

Amazon EC2 menyediakan grup penempatan untuk jaringan. Grup penempatan adalah pengelompokan logis instans untuk mengurangi latensi. Menggunakan grup penempatan dengan jenis instans yang didukung dan Elastic Network Adapter (ENA) memungkinkan beban kerja dapat berpartisipasi di jaringan 25 Gbps berlatensi rendah dan lebih sedikit jitter. Grup penempatan direkomendasikan untuk beban kerja yang memanfaatkan latensi jaringan yang rendah, throughput jaringan yang tinggi, atau keduanya.

Layanan yang sensitif terhadap latensi dikirimkan di lokasi edge menggunakan jaringan global AWS, seperti Amazon CloudFront. Lokasi edge ini biasanya menyediakan layanan seperti jaringan pengiriman konten (CDN) dan sistem nama domain (DNS). Dengan memiliki layanan ini di edge, beban kerja dapat merespons dengan latensi yang rendah untuk meminta konten atau resolusi DNS. Layanan-layanan ini juga menyediakan layanan geografis seperti penargetan geografis konten (menyediakan konten yang berbeda berdasarkan lokasi pengguna akhir) atau perutean berbasis latensi untuk mengarahkan pengguna akhir ke Wilayah terdekat (latensi minimum).

Gunakan layanan edge untuk mengurangi latensi dan memungkinkan caching konten. Konfigurasikan kontrol cache dengan benar untuk DNS dan HTTP/HTTPS untuk mendapat manfaat maksimal dari pendekatan ini.

Langkah implementasi

- Tangkap informasi tentang lalu lintas IP ke dan dari antarmuka jaringan.
  - Pencatatan log lalu lintas IP menggunakan VPC Flow Logs
  - Cara alamat IP klien dipertahankan di AWS Global Accelerator
- Analisis pola akses jaringan di beban kerja Anda untuk mengidentifikasi cara pengguna menggunakan aplikasi Anda.
  - Gunakan alat pemantauan, seperti Amazon CloudWatch dan AWS CloudTrail, untuk mengumpulkan data tentang aktivitas jaringan.
  - Analisis data untuk mengidentifikasi pola akses jaringan.
- Pilih Wilayah untuk deployment beban kerja Anda berdasarkan elemen utama berikut:
  - Lokasi data Anda: Untuk aplikasi dengan banyak data (seperti big data dan machine learning), kode aplikasi harus dijalankan sedekat mungkin dengan data.
  - Lokasi pengguna Anda: Untuk aplikasi yang ditampilkan kepada pengguna, pilih Wilayah yang dekat dengan pengguna beban kerja Anda.
  - Kendala lainnya: Pertimbangkan kendala seperti biaya dan kepatuhan sebagaimana dijelaskan dalam Hal-Hal yang Perlu Dipertimbangkan saat Memilih Wilayah untuk Beban Kerja

- Gunakan Zona Lokal AWS untuk menjalankan beban kerja seperti rendering video. Zona Lokal memungkinkan Anda mendapatkan semua manfaat dari komputasi dan sumber daya penyimpanan yang lebih dekat dengan pengguna akhir.
- Gunakan AWS Outposts untuk beban kerja yang harus tetap berada on-premise dan di tempat Anda ingin beban kerja tersebut berfungsi dengan lancar bersama beban kerja Anda yang lain di AWS.
- Aplikasi seperti streaming video live dengan resolusi tinggi, audio dengan fidelity tinggi, dan realitas tertambah atau realitas virtual (AR/VR) memerlukan latensi yang sangat rendah untuk perangkat 5G. Untuk aplikasi semacam itu, pertimbangkan AWS Wavelength. AWS Wavelength menyematkan layanan komputasi dan penyimpanan AWS dalam jaringan 5G, sehingga dapat menyediakan infrastruktur komputasi edge seluler untuk mengembangkan, men-deploy, dan menskalakan aplikasi berlatensi sangat rendah.
- Gunakan caching lokal atau Solusi Caching AWS untuk aset yang sering digunakan guna meningkatkan performa, mengurangi pergerakan data, dan mengurangi dampak lingkungan.

Layanan	Kapan harus digunakan
Amazon CloudFront	Gunakan untuk meng-cache konten statis seperti gambar, skrip, dan video, serta konten dinamis seperti respons API atau aplikasi web.
Amazon ElastiCache	Gunakan untuk meng-cache konten bagi aplikasi web.
DynamoDB Accelerator	Gunakan untuk menambahkan percepatan dalam memori ke tabel DynamoDB Anda.

- Gunakan layanan yang dapat membantu Anda menjalankan kode lebih dekat dengan pengguna beban kerja Anda seperti berikut:

Layanan	Kapan harus digunakan
Lambda@Edge	Gunakan untuk operasi dengan banyak komputasi yang dimulai saat objek tidak ada dalam cache.

Layanan	Kapan harus digunakan
Fungsi Amazon CloudFront	Gunakan untuk kasus penggunaan sederhana seperti permintaan HTTP atau manipulasi respons yang dapat dimulai oleh fungsi dengan masa pakai singkat.
AWS IoT Greengrass	Gunakan untuk menjalankan komputasi lokal, olahpesan, dan caching data untuk perangkat yang terhubung.

- Beberapa aplikasi memerlukan titik masuk tetap atau performa lebih tinggi dengan mengurangi jitter dan latensi bita pertama, dan meningkatkan throughput. Aplikasi-aplikasi ini dapat memperoleh manfaat dari layanan jaringan yang menyediakan alamat IP anycast statis dan pengakhiran TCP di lokasi edge. AWS Global Accelerator dapat meningkatkan performa untuk aplikasi Anda hingga sebesar 60% dan memberikan failover cepat untuk arsitektur multi-wilayah. AWS Global Accelerator memberikan kepada Anda alamat IP anycast statis yang berfungsi sebagai titik masuk tetap untuk aplikasi Anda yang di-hosting di satu atau beberapa Wilayah AWS. Alamat IP ini mengizinkan lalu lintas masuk ke jaringan global AWS sedekat mungkin ke pengguna Anda. AWS Global Accelerator mengurangi waktu penyiapan sambungan awal dengan membuat sambungan TCP antara klien dan lokasi edge AWS yang terdekat ke klien. Tinjau penggunaan AWS Global Accelerator untuk meningkatkan performa beban kerja TCP/UDP Anda dan memberikan failover cepat untuk arsitektur multi-Wilayah.

Sumber daya

Praktik terbaik terkait:

- COST07-BP02 Mengimplementasikan Wilayah berdasarkan biaya
- COST08-BP03 Mengimplementasikan layanan untuk mengurangi biaya transfer data
- REL10-BP01 Melakukan deployment beban kerja ke beberapa lokasi
- REL10-BP02 Memilih lokasi yang sesuai untuk deployment multilokasi
- SUS01-BP01 Memilih Wilayah berdasarkan persyaratan bisnis dan tujuan keberlanjutan
- SUS02-BP04 Mengoptimalkan penempatan geografis beban kerja berdasarkan persyaratan jaringannya
- SUS04-BP07 Meminimalkan perpindahan data di jaringan

Dokumen terkait:

- Infrastruktur Global AWS
- Zona Lokal AWS dan AWS Outposts, memilih teknologi yang tepat untuk beban kerja edge Anda
- Grup penempatan
- Zona Lokal AWS
- AWS Outposts
- AWS Wavelength
- Amazon CloudFront
- AWS Global Accelerator
- AWS Direct Connect
- AWS Site-to-Site VPN
- Amazon Route 53

Video terkait:

- Video Penjelas Zona Lokal AWS
- AWS Outposts: Ikhtisar dan Cara Kerjanya
- AWS re:Invent 2021 - AWS Outposts: Membawa pengalaman AWS ke on-premise
- AWS re:Invent 2020: AWS Wavelength: Menjalankan aplikasi dengan latensi sangat rendah di edge 5G
- AWS re:Invent 2022 - Zona Lokal AWS: Membangun aplikasi untuk edge terdistribusi
- AWS re:Invent 2021 - Membangun situs web latensi rendah dengan Amazon CloudFront
- AWS re:Invent 2022 - Meningkatkan performa dan ketersediaan dengan AWS Global Accelerator
- AWS re:Invent 2022 - Membangun jaringan area luas global menggunakan AWS
- AWS re:Invent 2020: Manajemen lalu lintas global dengan Amazon Route 53

Contoh terkait:

- Lokakarya AWS Global Accelerator
- Menangani Penulisan Ulang dan Pengarahan Ulang menggunakan Fungsi Edge

PERF04-BP07 Mengoptimalkan konfigurasi jaringan berdasarkan metrik

Gunakan data yang telah terkumpul dan dianalisis untuk mengambil keputusan yang tepat terkait pengoptimalan konfigurasi jaringan Anda.

Antipola umum:

- Anda beranggapan bahwa semua masalah kinerja disebabkan oleh aplikasi.
- Anda hanya menguji performa jaringan dari lokasi yang dekat dari tempat deployment beban kerja.
- Anda menggunakan konfigurasi default untuk semua layanan jaringan.
- Anda menyediakan terlalu banyak sumber daya jaringan untuk memberikan kapasitas yang memadai.

Manfaat menjalankan praktik terbaik ini: Dengan mengumpulkan metrik jaringan AWS yang diperlukan dan mengimplementasikan alat pemantauan jaringan, Anda dapat memahami performa jaringan dan mengoptimalkan konfigurasi jaringan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Memantau lalu lintas ke dan dari VPC, subnet, atau antarmuka jaringan sangat penting untuk memahami cara memanfaatkan sumber daya jaringan AWS dan mengoptimalkan konfigurasi jaringan. Dengan menggunakan alat jaringan AWS berikut ini, Anda dapat lebih jauh memeriksa informasi tentang penggunaan lalu lintas, akses jaringan, dan log.

Langkah implementasi

- Identifikasikan metrik kinerja utama seperti latensi atau kehilangan paket untuk dikumpulkan. AWS menyediakan beberapa alat yang dapat membantu Anda mengumpulkan metrik-metrik ini. Dengan menggunakan alat-alat berikut, Anda dapat lebih lanjut memeriksa informasi tentang penggunaan lalu lintas, akses jaringan, dan log:

Alat AWS	Di mana harus menggunakan
Amazon VPC IP Address Manager.	Gunakan IPAM untuk merencanakan, melacak, dan memantau alamat IP untuk AWS dan beban kerja on-premise Anda. Ini

Alat AWS	Di mana harus menggunakan
	adalah praktik terbaik untuk mengoptimalkan alokasi dan penggunaan alamat IP.
VPC Flow Logs	Gunakan VPC Flow Log untuk menangkap informasi mendetail tentang lalu lintas ke dan dari antarmuka jaringan di VPC Anda. Dengan VPC Flow Log, Anda dapat mendiagnosis aturan grup keamanan yang terlalu ketat atau longgar dan menentukan arah lalu lintas ke dan dari antarmuka jaringan.
AWS Transit Gateway Flow Logs	Gunakan AWS Transit Gateway Flow Logs untuk menangkap informasi tentang lalu lintas IP yang masuk dan keluar gateway transit Anda.
Logging kueri DNS	Informasi log tentang kueri DNS publik atau privat yang diterima Route 53. Dengan log DNS, Anda dapat mengoptimalkan konfigura si DNS dengan memahami domain atau sub-domain yang diminta atau lokasi EDGE Route 53 yang merespons kueri DNS.
Reachability Analyzer	Reachability Analyzer untuk menganali sis dan melakukan debug keterjangkauan jaringan. Reachability Analyzer adalah alat analisis konfigurasi yang memungkinkan Anda melakukan pengujian konektivitas antara sumber daya sumber dan sumber daya destinasi di VPC Anda. Alat ini membantu Anda memverifikasi bahwa konfigurasi jaringan Anda sesuai dengan konektivitas yang ditarget.

Alat AWS	Di mana harus menggunakan
Network Access Analyzer	Network Access Analyzer membantu Anda memahami akses jaringan ke sumber daya Anda. Anda dapat menggunakan Network Access Analyzer untuk menentukan persyaratan akses jaringan Anda serta mengidentifikasi jalur jaringan yang berpotensi tidak memenuhi persyaratan yang Anda tentukan. Dengan mengoptimalkan konfigurasi jaringan Anda yang bersangkutan, Anda dapat memahami dan memverifikasi status jaringan Anda dan menunjukkan apakah jaringan Anda di AWS memenuhi persyaratan kepatuhan Anda.
Amazon CloudWatch	Gunakan Amazon CloudWatch dan aktifkan metrik yang sesuai untuk opsi jaringan. Pastikan Anda memilih metrik jaringan yang tepat untuk beban kerja Anda. Contohnya , Anda dapat mengaktifkan metrik untuk Penggunaan Alamat Jaringan VPC, Gateway NAT VPC, AWS Transit Gateway, terowongan VPN, AWS Network Firewall, Elastic Load Balancing, dan AWS Direct Connect. Terus-menerus memantau metrik merupakan praktik yang bagus untuk mengamati dan memahami penggunaan dan status jaringan Anda, yang membantu Anda mengoptimalkan konfigurasi jaringan berdasarkan pengamatan Anda.

Alat AWS	Di mana harus menggunakan
AWS Network Manager	Dengan menggunakan AWS Network Manager, Anda dapat memantau kinerja waktu nyata dan historis dari Jaringan Global AWS untuk tujuan operasional dan perencana an. Network Managermenyediakan latensi jaringan agregat antara Wilayah AWS dan Zona Ketersediaan dan dalam setiap Zona Ketersediaan, sehingga Anda dapat lebih memahami bagaimana performa aplikasi Anda terkait dengan performa jaringan AWS yang mendasarinya.
Amazon CloudWatch RUM	Gunakan Amazon CloudWatch RUM untuk mengumpulkan metrik yang memberi Anda wawasan yang membantu Anda mengident ifikasi, memahami, dan meningkatkan pengalaman pengguna.

- Identifikasikan sumber data terbesar dan pola lalu lintas aplikasi menggunakan VPC dan AWS Transit Gateway Flow Logs.
- Nilai dan optimalkan arsitektur jaringan Anda saat ini termasuk VPC, subnet, dan perutean. Sebagai contoh, Anda dapat mengevaluasi bagaimana AWS Transit Gateway atau peering VPC yang berbeda dapat membantu Anda meningkatkan jaringan dalam arsitektur Anda.
- Nilai jalur perutean di jaringan Anda untuk memastikan digunakannya jalur terpendek antartujuan. Network Access Analyzer dapat membantu Anda melakukannya.

Sumber daya

Dokumen terkait:

- VPC Flow Logs
- Logging kueri DNS publik
- Apa itu IPAM?
- Apa itu Reachability Analyzer?

- Apa itu Network Access Analyzer?
- Metrik CloudWatch untuk VPC Anda
- Mengoptimalkan performa dan mengurangi biaya untuk analitik jaringan dengan VPC Flow Logs dalam format Apache Parquet
- Memantau jaringan global dan inti Anda dengan metrik Amazon CloudWatch
- Memantau sumber daya dan lalu lintas jaringan terus-menerus

Video terkait:

- Praktik terbaik dan tip jaringan dengan AWS Well-Architected Framework
- Memantau dan memecahkan masalah lalu lintas jaringan

Contoh terkait:

- Lokakarya Jaringan AWS
- Pemantauan Jaringan AWS

Proses dan budaya

PERF 5. Bagaimana praktik dan budaya organisasi Anda berkontribusi pada efisiensi performa dalam beban kerja Anda?

Saat merancang beban kerja, ada prinsip dan praktik yang dapat Anda adopsi untuk membantu Anda menjalankan beban kerja cloud berkinerja tinggi yang efisien dengan lebih baik. Untuk mengadopsi budaya yang mendorong efisiensi performa beban kerja cloud, pertimbangkan prinsip dan praktik utama ini:

Praktik terbaik

- PERF05-BP01 Membuat indikator kinerja utama (KPI) untuk mengukur kesehatan dan kinerja beban kerja
- PERF05-BP02 Menggunakan solusi pemantauan untuk memahami area dengan kinerja paling penting
- PERF05-BP03 Menetapkan proses untuk meningkatkan kinerja beban kerja
- PERF05-BP04 Menguji beban untuk beban kerja Anda

- PERF05-BP05 Menggunakan otomatisasi untuk secara proaktif memulihkan masalah terkait kinerja
- PERF05-BP06 Menjaga kemutakhiran beban kerja dan layanan Anda
- PERF05-BP07 Meninjau metrik dalam interval yang selaras

PERF05-BP01 Membuat indikator kinerja utama (KPI) untuk mengukur kesehatan dan kinerja beban kerja

Identifikasi KPI yang secara kuantitatif dan kualitatif mengukur kinerja beban kerja. KPI membantu Anda mengukur kesehatan dan kinerja beban kerja yang terkait dengan tujuan bisnis.

Antipola umum:

- Anda hanya memantau metrik tingkat sistem untuk memperoleh wawasan tentang beban kerja Anda dan tidak memahami dampak bisnis pada metrik-metrik tersebut.
- Anda berasumsi bahwa KPI Anda sudah dipublikasikan dan dibagikan sebagai data metrik standar.
- Anda tidak menetapkan KPI kuantitatif yang dapat diukur.
- Anda tidak menyelaraskan KPI dengan tujuan atau strategi bisnis.

Manfaat menjalankan praktik terbaik ini: Mengidentifikasi KPI tertentu yang mewakili kesehatan dan performa beban kerja dapat membantu menyelaraskan tim pada prioritas mereka dan menentukan hasil bisnis yang sukses. Ketika metrik-metrik tersebut kepada semua departemen, akan ada visibilitas dan kesepakatan tentang ambang batas, harapan, dan dampak bisnis.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

KPI memungkinkan tim bisnis dan rekayasa untuk menyepakati pengukuran tujuan dan strategi serta bagaimana faktor-faktor tersebut bekerja bersama untuk menciptakan hasil bisnis. Misalnya, beban kerja situs web mungkin menggunakan waktu muat halaman sebagai indikasi kinerja secara keseluruhan. Metrik ini adalah salah satu dari beberapa poin data yang mengukur pengalaman pengguna. Selain mengidentifikasi ambang batas waktu muat halaman, Anda harus mendokumentasikan hasil yang diharapkan atau risiko bisnis yang diperkirakan jika kinerja ideal tidak dipenuhi. Waktu muat halaman yang lama memengaruhi pengguna akhir Anda secara langsung, mengurangi tingkat pengalaman pengguna mereka, dan dapat menyebabkan hilangnya pelanggan. Saat Anda menetapkan ambang batas KPI Anda, gabungkan ambang batas industri serta harapan pengguna akhir Anda. Misalnya, jika ambang batas industri saat ini adalah halaman web dimuat

dalam waktu dua detik, tetapi pengguna akhir Anda mengharapkan halaman web dimuat dalam waktu satu detik, maka Anda harus mempertimbangkan kedua poin data ini ketika menetapkan KPI.

Tim Anda harus mengevaluasi KPI beban kerja Anda menggunakan data granular waktu nyata dan data historis sebagai rujukan dan membuat dasbor yang menjalankan penghitungan metrik pada data KPI Anda untuk menghasilkan wawasan operasi dan pemanfaatan. KPI harus didokumentasikan dan mencakup ambang batas yang disepakati yang mendukung tujuan, dan harus dipetakan ke metrik-metrik yang dipantau. KPI harus ditinjau ulang ketika tujuan bisnis, strategi, dan kebutuhan pengguna akhir berubah.

Langkah implementasi

1. Identifikasi dan dokumentasikan pemangku kepentingan bisnis utama.
2. Bekerjalah dengan para pemangku kepentingan ini untuk menentukan dan mendokumentasikan tujuan beban kerja Anda.
3. Tinjau praktik terbaik industri untuk mengidentifikasi KPI relevan yang diselaraskan dengan sasaran beban kerja Anda.
4. Gunakan praktik terbaik industri dan sasaran beban kerja Anda untuk menetapkan target KPI beban kerja Anda. Gunakan informasi ini untuk mengatur ambang batas KPI untuk tingkat keparahan atau alarm.
5. Identifikasi dan dokumentasikan risiko dan dampak jika KPI tidak terpenuhi.
6. Identifikasi dan dokumentasikan metrik yang dapat membantu Anda menetapkan KPI.
7. Gunakan alat pemantauan seperti Amazon CloudWatch atau AWS Config untuk mengumpulkan metrik dan mengukur KPI.
8. Gunakan dasbor untuk memvisualisasikan dan mengomunikasikan KPI kepada pemangku kepentingan.
9. Tinjau dan analisis metrik secara rutin untuk mengidentifikasi area beban kerja yang perlu ditingkatkan.
10. Tinjau ulang KPI ketika sasaran bisnis atau kinerja beban kerja berubah.

Sumber daya

Dokumen terkait:

- Dokumentasi CloudWatch

- AWS Partner Pemantauan, Pencatatan Log, dan Performa
- Dokumentasi X-Ray
- Menggunakan dasbor Amazon CloudWatch
- KPI Quick

Video terkait:

- AWS re:Invent 2019: Menaikkan skala ke 10 juta pengguna pertama Anda
- Atasi kekacauan: Dapatkan wawasan dan visibilitas operasional
- Buat Rencana Pemantauan

Contoh terkait:

- Membuat dasbor dengan Quick

PERF05-BP02 Menggunakan solusi pemantauan untuk memahami area dengan kinerja paling penting

Pahami dan identifikasi area di mana peningkatan kinerja beban kerja akan memiliki dampak positif pada efisiensi atau pengalaman pelanggan. Contohnya, situs web yang memiliki banyak interaksi pelanggan dapat memperoleh manfaat dari penggunaan layanan edge untuk memindahkan penyampaian konten lebih dekat ke pelanggan.

Antipola umum:

- Anda berasumsi bahwa metrik komputasi standar seperti penggunaan CPU atau tekanan memori sudah cukup untuk menangkap masalah performa.
- Anda hanya menggunakan metrik default yang dicatat oleh perangkat lunak pemantauan Anda yang dipilih.
- Anda hanya meninjau metrik ketika terdapat masalah.

Manfaat menjalankan praktik terbaik ini: Pemahaman tentang area yang memerlukan kinerja tinggi membantu para pemilik beban kerja dalam memantau KPI dan memprioritaskan peningkatan berdampak tinggi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Siapkan pelacakan menyeluruh untuk mengidentifikasi pola lalu lintas, latensi, dan area kinerja kritis. Pantau pola akses data Anda untuk kueri yang lambat atau data dengan fragmentasi dan partisi yang buruk. Identifikasi area beban kerja terbatas menggunakan pengujian atau pemantauan beban.

Tingkatkan efisiensi kinerja dengan memahami arsitektur, pola lalu lintas, dan pola akses data Anda, serta identifikasi latensi dan waktu pemrosesan Anda. Identifikasi potensi hambatan yang bisa memengaruhi pengalaman pelanggan selama beban kerja berkembang. Setelah menginvestigasi area-area tersebut, lihat solusi mana yang dapat Anda deploy untuk menghilangkan masalah kinerja tersebut.

Langkah implementasi

1. Siapkan pemantauan menyeluruh untuk mengetahui semua komponen dan metrik beban kerja. Berikut adalah contoh solusi pemantauan di AWS.

Layanan	Di mana harus menggunakan
Pemantauan Pengguna Nyata (RUM) Amazon CloudWatch	Untuk menyerap metrik performa aplikasi dari sesi sisi klien dan frontend pengguna nyata.
AWS X-Ray	Untuk melacak lalu lintas melalui lapisan aplikasi dan mengidentifikasi latensi antara komponen dan dependensi. Gunakan peta layanan X-Ray untuk melihat hubungan dan latensi antar komponen beban kerja.
Wawasan Kinerja Amazon Relational Database Service	Untuk melihat metrik kinerja basis data dan mengidentifikasi peningkatan kinerja.
Amazon RDS Enhanced Monitoring	Untuk melihat metrik kinerja OS basis data.
Amazon DevOps Guru	Untuk mendeteksi pola operasi yang tidak normal sehingga Anda dapat mengidentifikasi masalah operasional sebelum berdampak pada pelanggan Anda.

2. Lakukan pengujian untuk membuat metrik, mengidentifikasi pola lalu lintas, hambatan, dan area kinerja kritis. Berikut adalah beberapa contoh cara melakukan pengujian:

- Siapkan CloudWatch Synthetic Canaries untuk meniru aktivitas pengguna berbasis browser secara terprogram menggunakan ekspresi tingkat dan tugas cron Linux untuk menghasilkan metrik yang konsisten dari waktu ke waktu.
- Gunakan Pengujian Beban Terdistribusi AWS untuk menghasilkan lalu lintas puncak atau menguji beban kerja pada tingkat pertumbuhan yang diharapkan.

3. Evaluasi metrik dan telemetri untuk mengidentifikasi area kinerja kritis Anda. Tinjau area-area ini dengan tim Anda untuk mendiskusikan pemantauan dan solusi untuk menghindari hambatan.
4. Lakukan eksperimen dengan peningkatan kinerja serta ukur perubahannya dengan data. Sebagai contoh, Anda dapat menggunakan CloudWatch Evidently untuk menguji peningkatan baru dan dampak kinerja pada beban kerja Anda.

Sumber daya

Dokumen terkait:

- Amazon Builders' Library
- Dokumentasi X-Ray
- Amazon CloudWatch RUM
- Amazon DevOps Guru

Video terkait:

- Pustaka Amazon Builders: 25 tahun keunggulan operasional Amazon
- Pemantauan Visual Aplikasi dengan Amazon CloudWatch Synthetics

Contoh terkait:

- Ukur waktu pemuatan halaman dengan Amazon CloudWatch Synthetics
- Klien Web Amazon CloudWatch RUM
- SDK X-Ray untuk Node.js
- SDK X-Ray untuk Python
- SDK X-Ray untuk Java
- SDK X-Ray untuk .Net
- SDK X-Ray untuk Ruby

- X-Ray Daemon
- Pengujian Beban Terdistribusi di AWS

PERF05-BP03 Menetapkan proses untuk meningkatkan kinerja beban kerja

Menetapkan proses untuk mengevaluasi layanan, pola desain, tipe sumber daya, dan konfigurasi baru saat sudah tersedia. Misalnya, jalankan pengujian kinerja yang sudah ada pada penawaran instans baru untuk menentukan potensinya untuk beban kerja Anda.

Antipola umum:

- Anda berasumsi bahwa arsitektur Anda saat ini statis dan tidak akan diperbarui dari waktu ke waktu.
- Anda memperkenalkan metrik arsitektur seiring waktu tanpa justifikasi metrik.

Manfaat menjalankan praktik terbaik ini: Setelah proses untuk membuat perubahan arsitektur ditetapkan, Anda dapat menggunakan data yang dikumpulkan untuk memengaruhi desain beban kerja Anda seiring waktu.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Kinerja beban kerja Anda memiliki beberapa kendala utama. Dokumentasikan kendala-kendala tersebut untuk mengetahui jenis inovasi apa saja yang mungkin meningkatkan kinerja beban kerja Anda. Gunakan informasi ini ketika mempelajari layanan atau teknologi baru ketika sudah tersedia untuk mengidentifikasi cara-cara untuk menghilangkan kendala atau bottleneck.

Identifikasi kendala kinerja utama untuk beban kerja Anda. Dokumentasikan kendala performa beban kerja Anda sehingga Anda tahu jenis-jenis inovasi apa yang dapat meningkatkan performa beban kerja Anda.

Langkah implementasi

- Identifikasi KPI kinerja beban kerja Anda seperti yang diuraikan dalam PERF05-BP01 Membuat indikator kinerja utama (KPI) untuk mengukur kesehatan dan kinerja beban kerja untuk menjadi garis acuan beban kerja Anda.
- Gunakan alat pengamatan AWS untuk mengumpulkan metrik kinerja dan mengukur KPI.

- Lakukan analisis mendalam untuk mengidentifikasi area (seperti konfigurasi dan kode aplikasi) di dalam beban kerja Anda yang berkinerja buruk seperti yang diuraikan dalam PERF05-BP02 Menggunakan solusi pemantauan untuk memahami area dengan kinerja paling penting.
- Gunakan alat analisis dan kinerja Anda untuk mengidentifikasi strategi pengoptimalan kinerja.
- Gunakan sandbox atau lingkungan praproduksi untuk memvalidasi efektivitas strategi.
- Implementasikan perubahan dalam produksi dan terus pantau kinerja beban kerja.
- Dokumentasikan perbaikan dan komunikasikan hal tersebut kepada para pemangku kepentingan.

Sumber daya

Dokumen terkait:

- Blog AWS
- Yang Baru dengan AWS

Video terkait:

- Saluran YouTube AWS Events
- Saluran YouTube AWS Online Tech Talks
- Saluran YouTube Amazon Web Services

Contoh terkait:

- AWS Github
- AWS Skill Builder

PERF05-BP04 Menguji beban untuk beban kerja Anda

Uji beban untuk beban kerja Anda untuk memverifikasi bahwa beban kerja Anda dapat menangani beban produksi dan mengidentifikasi kemacetan kinerja apa pun.

Antipola umum:

- Anda melakukan uji beban bagian beban kerja secara terpisah-pisah, bukan seluruh beban kerja.
- Anda melakukan uji beban pada infrastruktur yang tidak sama dengan lingkungan produksi Anda.

- Anda hanya melakukan pengujian beban pada beban yang diharapkan, tidak lebih, untuk membantu memperkirakan area yang mungkin akan bermasalah di masa depan.
- Anda melakukan pengujian beban tanpa mempelajari Kebijakan Pengujian Amazon EC2 dan mengirimkan Formulir Pengajuan Peristiwa Simulasi. Ini mengakibatkan pengujian Anda gagal dijalankan, karena terlihat seperti peristiwa penolakan layanan.

Manfaat menjalankan praktik terbaik ini: Mengukur kinerja Anda dalam uji beban akan menunjukkan di mana Anda akan terdampak saat beban meningkat. Hal ini bisa memberi Anda kemampuan untuk mengantisipasi perubahan yang diperlukan sebelum berdampak pada beban kerja Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Pengujian beban di cloud adalah proses untuk mengukur kinerja beban kerja cloud dalam kondisi realistis dengan beban pengguna yang diharapkan. Proses ini melibatkan penyediaan lingkungan cloud mirip produksi, penggunaan alat pengujian beban untuk menghasilkan beban, dan analisis metrik untuk menilai kemampuan penanganan beban kerja Anda yang realistis. Uji beban harus dijalankan menggunakan versi data produksi yang sintetis atau sudah dibersihkan (menghapus informasi sensitif atau pengidentifikasi). Lakukan uji beban secara otomatis sebagai bagian dari pipeline pengiriman Anda, dan bandingkan hasilnya terhadap KPI dan ambang batas yang telah ditentukan sebelumnya. Proses ini membantu Anda terus mencapai kinerja yang dibutuhkan.

Langkah implementasi

- Siapkan lingkungan pengujian berdasarkan lingkungan produksi Anda. Anda dapat menggunakan layanan AWS untuk menjalankan lingkungan skala produksi untuk menguji arsitektur Anda.
- Pilih dan konfigurasikan alat pengujian beban yang sesuai dengan beban kerja Anda.
- Tentukan skenario dan parameter pengujian beban (seperti durasi pengujian dan jumlah pengguna).
- Lakukan skenario pengujian pada skala besar. Manfaatkan AWS Cloud untuk menguji beban kerja Anda untuk mengetahui di mana letak kesalahan penskalaannya, atau apakah penskalaannya berada di jalur nonlinier. Misalnya, gunakan Instans Spot untuk menghasilkan beban dengan biaya rendah dan temukan hambatan sebelum dialami di lingkungan produksi.
- Pantau dan rekam metrik kinerja (seperti throughput dan waktu respons). Amazon CloudWatch dapat mengumpulkan metrik di seluruh sumber daya dalam arsitektur Anda. Anda juga dapat

mengumpulkan dan memublikasikan metrik kustom untuk memunculkan metrik turunan (derived metric) atau bisnis.

- Analisis hasil untuk mengidentifikasi hambatan kinerja dan area untuk perbaikan.
- Dokumentasikan dan laporkan proses dan hasil pengujian beban.

Sumber daya

Dokumen terkait:

- AWS CloudFormation
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- Pengujian Beban Terdistribusi di AWS

Video terkait:

- Memecahkan Masalah dengan Solusi AWS: Pengujian Beban Terdistribusi

- Optimalkan aplikasi dengan Amazon CloudWatch RUM
- Demo Amazon CloudWatch Synthetics

Contoh terkait:

- Pengujian Beban Terdistribusi di AWS

PERF05-BP05 Menggunakan otomatisasi untuk secara proaktif memulihkan masalah terkait kinerja

Gunakan indikator kinerja utama (KPI), yang digabungkan dengan sistem pamantauan dan peringatan, untuk menangani masalah terkait kinerja secara proaktif.

Antipola umum:

- Anda hanya membekali staf operasional dengan kemampuan untuk membuat perubahan operasional pada beban kerja.
- Anda membiarkan semua alarm disaring ke tim operasi tanpa perbaikan proaktif.

Manfaat menerapkan praktik terbaik ini: Perbaikan tindakan alarm yang proaktif memungkinkan staf dukungan untuk berkonsentrasi pada item-item yang tidak dapat ditindaklanjuti secara otomatis. Ini membantu staf operasi menangani semua alarm tanpa kewalahan dan mereka hanya berkonsentrasi pada alarm yang kritis.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Gunakan alarm untuk memicu tindakan otomatis untuk memperbaiki masalah ketika memungkinkan. Teruskan alarm ke personel yang mampu merespons jika respons otomatis tidak memungkinkan. Misalnya, Anda mungkin memiliki sistem yang dapat memprediksi nilai dan alarm indikator kinerja utama (KPI) yang diharapkan ketika melanggar ambang batas tertentu, atau alat yang dapat menghentikan atau membatalkan deployment secara otomatis jika KPI berada di luar nilai yang diharapkan.

Implementasikan proses yang menyediakan visibilitas tentang kinerja saat beban kerja Anda berjalan. Bangun dasbor pemantauan dan buat norma acuan untuk harapan kinerja guna menentukan apakah beban kerja berkinerja secara optimal.

Langkah implementasi

- Identifikasi dan pahami masalah kinerja yang dapat diperbaiki secara otomatis. Gunakan solusi pemantauan AWS seperti Amazon CloudWatch atau AWS X-Ray untuk membantu Anda lebih memahami akar penyebab masalah.
- Buat rencana dan proses perbaikan langkah demi langkah yang dapat digunakan untuk memperbaiki masalah secara otomatis.
- Konfigurasikan pemicu untuk memulai proses perbaikan secara otomatis. Misalnya, Anda dapat menentukan pemicu untuk memulai ulang instans secara otomatis ketika mencapai ambang batas pemanfaatan CPU tertentu.
- Gunakan layanan dan teknologi AWS untuk mengotomatiskan proses perbaikan. Sebagai contoh, AWS Systems Manager Automation menyediakan cara yang aman dan dapat diskalakan untuk mengotomatiskan proses perbaikan.
- Uji proses perbaikan otomatis di lingkungan praproduksi.
- Setelah pengujian, implementasikan proses perbaikan di lingkungan produksi dan terus pantau untuk mengidentifikasi area untuk perbaikan.

Sumber daya

Dokumen terkait:

- Dokumentasi CloudWatch
- Partner AWS Partner Network Pemantauan, Pencatatan Log, dan Performa
- Dokumentasi X-Ray
- Menggunakan Alarm dan Tindakan Alarm di CloudWatch

Video terkait:

- Mengotomatiskan operasi cloud secara cerdas
- Menyiapkan kontrol dalam skala besar di lingkungan AWS Anda
- Mengotomatiskan manajemen dan kepatuhan patch menggunakan AWS
- Bagaimana Amazon menggunakan metrik yang lebih baik untuk meningkatkan kinerja situs web

Contoh terkait:

- Kustomisasi Alarm CloudWatch Logs

PERF05-BP06 Menjaga kemutakhiran beban kerja dan layanan Anda

Terus ikuti informasi tentang layanan dan fitur cloud baru untuk mengadopsi fitur yang efisien, menghilangkan masalah, dan meningkatkan efisiensi kinerja beban kerja Anda secara keseluruhan.

Antipola umum:

- Anda berasumsi bahwa arsitektur Anda saat ini statis dan tidak akan diperbarui seiring waktu.
- Anda tidak memiliki sistem atau koordinasi rutin untuk mengevaluasi apakah perangkat lunak dan paket yang diperbarui kompatibel dengan beban kerja Anda.

Manfaat menjalankan praktik terbaik ini: Dengan menetapkan proses untuk tetap mutakhir pada layanan dan penawaran baru, Anda dapat menerapkan fitur dan kemampuan baru, menyelesaikan masalah, dan meningkatkan kinerja beban kerja.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Evaluasi cara meningkatkan performa saat layanan, pola desain, dan fitur produk baru tersedia. Tentukan mana hal-hal yang dapat meningkatkan kinerja atau menambah efisiensi beban kerja melalui evaluasi, diskusi internal, atau analisis eksternal. Tentukan proses untuk mengevaluasi pembaruan, fitur baru, dan layanan yang relevan dengan beban kerja Anda. Misalnya, bangun bukti konsep yang memanfaatkan teknologi baru atau berkonsultasi dengan grup internal. Saat mencoba layanan atau ide baru, jalankan pengujian kinerja untuk mengukur pengaruhnya terhadap kinerja beban kerja.

Langkah implementasi

- Buat inventaris perangkat lunak dan arsitektur beban kerja Anda dan identifikasi komponen yang perlu diperbarui.
- Identifikasi sumber berita dan pembaruan yang terkait dengan komponen beban kerja Anda. Sebagai contoh, Anda dapat berlangganan blog “Apa yang Baru” di AWS untuk produk yang sesuai dengan komponen beban kerja Anda. Anda dapat berlangganan umpan RSS atau mengelola langganan email Anda.
- Tentukan jadwal untuk mengevaluasi layanan dan fitur baru untuk beban kerja Anda.
  - Anda dapat menggunakan Inventaris AWS Systems Manager untuk mengumpulkan metadata sistem operasi (OS), aplikasi, dan instans dari instans Amazon EC2 dan secara cepat memahami instans mana yang menjalankan perangkat lunak dan konfigurasi yang diperlukan oleh kebijakan perangkat lunak Anda dan instans mana yang perlu diperbarui.
- Pahami cara memperbarui komponen beban kerja Anda. Manfaatkan ketangkasan di cloud untuk menguji dengan cepat bagaimana fitur baru dapat meningkatkan beban kerja Anda untuk mendapatkan efisiensi performa.
- Gunakan otomatisasi untuk proses pembaruan guna mengurangi tingkat upaya dalam melakukan deployment fitur baru dan membatasi kesalahan yang disebabkan oleh proses manual.
  - Anda dapat menggunakan CI/CD untuk secara otomatis memperbarui AMI, image kontainer, dan artefak lain yang terkait dengan aplikasi cloud Anda.
  - Anda dapat menggunakan alat-alat seperti AWS Systems Manager Patch Manager untuk mengotomatiskan proses pembaruan sistem dan menjadwalkan aktivitas menggunakan Periode Pemeliharaan AWS Systems Manager.
- Dokumentasikan proses Anda untuk mengevaluasi pembaruan dan layanan baru. Bekali pemilik Anda dengan waktu dan ruang yang dibutuhkan untuk meneliti, menguji, bereksperimen, serta memvalidasi pembaruan dan layanan baru. Lihat kembali persyaratan dan KPI bisnis

terdokumentasi untuk membantu memprioritaskan pembaruan mana yang akan menciptakan dampak bisnis yang positif.

Sumber daya

Dokumen terkait:

- Blog AWS
- Yang Baru dengan AWS

Video terkait:

- Saluran YouTube AWS Events
- Saluran YouTube AWS Online Tech Talks
- Saluran YouTube Amazon Web Services

Contoh terkait:

- Well-Architected Labs - Manajemen Inventaris dan Patch
- Lab: AWS Systems Manager

PERF05-BP07 Meninjau metrik dalam interval yang selaras

Sebagai bagian pemeliharaan rutin, atau sebagai respons terhadap peristiwa atau insiden, tinjau metrik mana yang dikumpulkan. Gunakan tinjauan ini untuk mengidentifikasi metrik mana yang penting untuk menangani masalah dan metrik mana yang merupakan tambahan. Jika dilacak, metrik tersebut dapat memudahkan Anda mengidentifikasi, mengatasi, dan mencegah masalah.

Antipola umum:

- Anda mengizinkan metrik untuk tetap dalam status alarm selama periode waktu yang lebih lama.
- Anda membuat alarm yang tidak dapat ditindaklanjuti oleh sistem otomatisasi.

Manfaat menjalankan praktik terbaik ini: Tinjau secara terus-menerus metrik yang dikumpulkan untuk memverifikasi metrik tersebut dapat mengidentifikasi, mengatasi, atau mencegah masalah. Metrik juga dapat kedaluwarsa jika Anda membiarkannya berada dalam status alarm untuk waktu yang lama.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

Terus-menerus tingkatkan pemantauan dan pengumpulan metrik. Sebagai bagian dari tindakan merespons insiden atau peristiwa, evaluasikan mana metrik yang berguna untuk mengatasi masalah dan mana metrik yang dapat membantu tetapi saat ini tidak terdeteksi. Gunakan metode ini untuk meningkatkan kualitas metrik yang Anda kumpulkan agar Anda dapat mencegah, atau agar dapat lebih cepat menangani, insiden pada masa mendatang.

Sebagai bagian dari tindakan merespons insiden atau peristiwa, evaluasikan mana metrik yang berguna untuk mengatasi masalah dan mana metrik yang dapat membantu tetapi saat ini tidak terdeteksi. Gunakan ini untuk meningkatkan kualitas metrik yang Anda kumpulkan agar dapat mencegah atau dapat lebih cepat mengatasi insiden di masa mendatang.

Langkah implementasi

1. Tentukan metrik kinerja penting yang perlu dipantau, yang selaras dengan tujuan beban kerja Anda.
2. Tetapkan garis acuan dan nilai yang diinginkan untuk setiap metrik.
3. Tetapkan frekuensi (seperti mingguan atau bulanan) untuk meninjau metrik penting.
4. Dalam setiap tinjauan, nilai tren dan penyimpangan dari nilai garis acuan. Cari setiap anomali atau hambatan performa.
5. Untuk masalah yang teridentifikasi, lakukan analisis akar penyebab secara mendalam untuk memahami alasan utama di balik masalah tersebut.
6. Dokumentasikan temuan Anda dan gunakan strategi untuk menangani masalah dan hambatan yang teridentifikasi.
7. Terus nilai dan tingkatkan proses peninjauan metrik.

Sumber Daya

Dokumen terkait:

- Dokumentasi CloudWatch
- Kumpulkan metrik dan log dari Instans Amazon EC2 serta server on-premise dengan Agen CloudWatch
- Partner AWS Partner Network Pemantauan, Pencatatan Log, dan Performa

- Dokumentasi X-Ray

Video terkait:

- Menyiapkan kontrol dalam skala besar di lingkungan AWS Anda
- Bagaimana Amazon menggunakan metrik yang lebih baik untuk meningkatkan kinerja situs web

Contoh terkait:

- Membuat dasbor dengan Quick
- Tingkat 100: Pemantauan dengan Dasbor CloudWatch

Optimisasi biaya

Pilar Optimasi Biaya mencakup kemampuan untuk menjalankan sistem guna menghadirkan nilai bisnis dengan harga yang paling rendah. Anda dapat menemukan panduan preskriptif tentang implementasi di laporan resmi Pilar Optimasi Biaya.

Area praktik terbaik

- Mempraktikkan Manajemen Keuangan Cloud
- Kesadaran akan penggunaan dan pengeluaran
- Sumber daya yang hemat
- Kelola sumber daya pasokan dan permintaan
- Pengoptimalan dari waktu ke waktu

Mempraktikkan Manajemen Keuangan Cloud

Pertanyaan

- COST 1. Bagaimana cara mengimplementasikan manajemen keuangan cloud?

COST 1. Bagaimana cara mengimplementasikan manajemen keuangan cloud?

Mengimplementasikan Manajemen Finansial Cloud membantu organisasi untuk mewujudkan nilai bisnis dan kesuksesan finansial karena mereka mengoptimalkan biaya, penggunaan, dan skala di AWS.

Praktik terbaik

- COST01-BP01 Membangun kepemilikan pengoptimalan biaya
- COST01-BP02 Buat kemitraan antara bagian keuangan dan teknologi
- COST01-BP03 Tetapkan prakiraan dan anggaran cloud
- COST01-BP04 Mengimplementasikan kesadaran biaya dalam proses organisasi Anda
- COST01-BP05 Melaporkan dan memberi tahu tentang pengoptimalan biaya
- COST01-BP06 Memantau biaya secara proaktif
- COST01-BP07 Mengikuti perkembangan rilisan layanan baru
- COST01-BP08 Menciptakan budaya sadar biaya
- COST01-BP09 Menghitung nilai bisnis dari optimasi biaya

COST01-BP01 Membangun kepemilikan pengoptimalan biaya

Buat tim (tim Cloud Business Office, Cloud Center of Excellence, atau FinOps) yang bertanggung jawab membangun dan memelihara budaya sadar biaya (cost awareness) di seluruh organisasi Anda. Pemilik optimasi biaya dapat berupa individu atau tim (membutuhkan orang-orang dari tim keuangan, teknologi, dan bisnis) yang memahami seluruh organisasi dan keuangan cloud.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Ini adalah pengantar fungsi atau tim Cloud Business Office (CBO) atau Cloud Center of Excellence (CCOE) yang bertanggung jawab membangun dan memelihara budaya sadar biaya (cost awareness) di komputasi cloud. Fungsi ini dapat terdiri dari individu yang sudah ada, tim di dalam organisasi Anda, atau tim baru yang terdiri dari pemangku kepentingan utama organisasi, keuangan, dan teknologi dari seluruh organisasi.

Fungsi ini (individu atau tim) memprioritaskan dan menggunakan sebagian besar waktunya untuk aktivitas manajemen dan pengoptimalan biaya. Untuk organisasi kecil, fungsi ini mungkin

memerlukan persentase waktu yang lebih sedikit dibandingkan dengan fungsi purnawaktu di korporasi yang lebih besar.

Fungsi ini (individu atau tim) memprioritaskan dan menggunakan sebagian besar waktunya untuk aktivitas manajemen dan pengoptimalan biaya. Untuk organisasi kecil, fungsi ini mungkin memerlukan persentase waktu yang lebih sedikit untuk aktivitas manajemen dan optimasi biaya dibandingkan dengan fungsi purnawaktu di korporasi yang lebih besar.

Fungsi ini memerlukan pendekatan multidisiplin, dengan kemampuan di bidang manajemen proyek, ilmu data, analisis keuangan, dan pengembangan perangkat lunak atau infrastruktur. Fungsi ini dapat meningkatkan efisiensi beban kerja dengan menjalankan pengoptimalan biaya dalam tiga kepemilikan yang berbeda:

- Tersentralisasi: Melalui tim yang ditunjuk seperti tim FinOps, tim Cloud Financial Management (CFM), Cloud Business Office (CBO), atau Cloud Center of Excellence (CCoE), pelanggan dapat merancang dan menerapkan mekanisme tata kelola dan mendorong praktik terbaik di seluruh perusahaan.
- Terdesentralisasi: Memengaruhi tim-tim teknologi untuk menjalankan optimasi biaya.
- Hybrid: Gabungan dari tim tersentralisasi dan terdesentralisasi dapat bekerja sama untuk menjalankan optimasi biaya.

Fungsi ini dapat diukur dari kemampuannya menjalankan dan menyampaikan tujuan pengoptimalan biaya (misalnya, metrik efisiensi beban kerja).

Anda harus mendapatkan sponsor eksekutif untuk fungsi ini, yang merupakan faktor keberhasilan utama. Sponsor ini dianggap sebagai penyokong penggunaan cloud hemat biaya, serta memberikan dukungan peningkatan kepada tim guna memastikan bahwa aktivitas pengoptimalan biaya ditangani menurut tingkat prioritas yang ditentukan oleh organisasi. Jika tidak, panduan dapat diabaikan dan peluang penghematan biaya tidak akan diprioritaskan. Bersama-sama, sponsor dan tim membantu organisasi Anda menggunakan cloud secara efisien dan menghadirkan nilai bisnis.

Jika Anda memiliki paket dukungan Business, Enterprise-On-Ramp, atau Enterprise dan memerlukan bantuan dalam membangun tim atau fungsi ini, hubungi ahli Cloud Financial Management (CFM) Anda melalui tim akun Anda.

Langkah implementasi

- Tentukan anggota utama: Semua bagian yang relevan dari organisasi Anda harus berkontribusi dan berminat pada manajemen biaya. Tim umum di dalam organisasi biasanya berisi: tim

keuangan, pemilik produk atau aplikasi, manajemen, dan teknis (DevOps). Beberapa dari mereka terlibat secara purnawaktu (keuangan atau teknis), sementara yang lain dilibatkan secara berkala sesuai kebutuhan. Individu atau tim yang melakukan CFM memerlukan rangkaian keahlian berikut:

- Pengembangan perangkat lunak: apabila ada skrip dan otomatisasi yang sedang dibuat.
- Rekayasa infrastruktur: untuk men-deploy skrip, mengotomatisasi proses, dan memahami bagaimana layanan atau sumber daya disediakan.
- Kecakapan operasi: CFM adalah tentang beroperasi di cloud secara efisien dengan mengukur, memantau, memodifikasi, merencanakan, dan menskalakan penggunaan cloud secara efisien.

- Tentukan tujuan dan metrik: Fungsi harus memberikan nilai kepada organisasi dengan cara yang berbeda. Tujuan-tujuan tersebut ditetapkan dan terus berkembang seiring dengan perkembangan organisasi. Aktivitas umum mencakup: membuat dan menjalankan program edukasi tentang pengoptimalan biaya di seluruh organisasi, mengembangkan standar di seluruh organisasi, seperti pemantauan dan pelaporan untuk pengoptimalan biaya, dan menetapkan sasaran beban kerja dalam pengoptimalan. Fungsi juga harus melaporkan kemampuan pengoptimalan biaya mereka secara rutin kepada organisasi.

  Anda dapat menentukan indikator kinerja utama (KPI) berbasis nilai atau biaya. Saat Anda menentukan KPI, Anda dapat menghitung biaya yang diharapkan sehubungan dengan efisiensi dan hasil bisnis yang diharapkan. KPI berbasis nilai mengaitkan metrik biaya dan penggunaan dengan pendorong nilai bisnis dan membantu merasionalisasi perubahan dalam pengeluaran AWS. Langkah pertama untuk menyimpulkan KPI berbasis nilai adalah bekerja sama, lintas organisasi, untuk memilih dan menyepakati serangkaian KPI standar.
- Adakan koordinasi rutin: Grup (keuangan, teknologi, dan tim bisnis) harus melakukan rapat secara teratur untuk meninjau tujuan dan metrik mereka. Koordinasi ini biasanya membahas peninjauan status organisasi, program apa pun yang sedang berlangsung, serta metrik pengoptimalan dan keuangan secara keseluruhan. Selanjutnya, beban kerja utama dilaporkan dengan lebih mendetail.

  Selama tinjauan rutin ini, Anda dapat meninjau efisiensi beban kerja (biaya) dan hasil bisnis. Misalnya, kenaikan biaya 20% untuk beban kerja dapat diselaraskan dengan peningkatan penggunaan pelanggan. Dalam kasus ini, kenaikan biaya 20% dapat diartikan sebagai investasi. Rapat koordinasi rutin ini dapat membantu tim mengidentifikasi KPI nilai yang memberikan makna bagi seluruh organisasi.

Sumber daya

Dokumen terkait:

- Blog CCOE AWS
- Membuat Kantor Bisnis Cloud
- CCOE - Cloud Center of Excellence

Video terkait:

- Kisah Sukses CCOE Vanguard

Contoh terkait:

- Menggunakan Cloud Center of Excellence (CCOE) untuk Mentransformasi Keseluruhan Perusahaan
- Membangun CCOE untuk mentransformasi keseluruhan perusahaan
- 7 Perangkap yang Perlu Dihindari Saat Membangun CCOE

COST01-BP02 Buat kemitraan antara bagian keuangan dan teknologi

Libatkan tim keuangan dan teknologi dalam diskusi biaya dan penggunaan di semua tahap perjalanan cloud Anda. Tim secara teratur bertemu dan membahas topik-topik seperti target dan tujuan organisasi, kondisi biaya dan penggunaan saat ini, serta praktik keuangan dan akuntansi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

Tim teknologi berinovasi lebih cepat di cloud karena siklus deployment infrastruktur, pengadaan, dan persetujuan yang lebih pendek. Ini dapat membutuhkan penyesuaian untuk organisasi keuangan yang sebelumnya terbiasa menjalankan proses yang memerlukan waktu lama dan banyak sumber daya untuk mendapatkan dan melakukan deployment modal di lingkungan on-premise dan pusat data, dan alokasi biaya hanya berdasarkan persetujuan proyek.

Dari perspektif organisasi keuangan dan pengadaan, proses penganggaran modal, permintaan modal, persetujuan, pengadaan, dan pemasangan infrastruktur fisik adalah proses yang telah dipelajari dan distandardisasi selama beberapa dekade:

- Tim rekayasa atau IT biasanya adalah pemohon
- Berbagai tim keuangan bertindak sebagai pemberi persetujuan dan pengadaan
- Tim operasi mengumpulkan, menumpuk, dan menyerahkan infrastruktur siap pakai

Dengan penerapan cloud, pengadaan dan penggunaan infrastruktur tidak lagi terikat pada rantai dependensi. Dalam model cloud, tim teknologi dan produk tidak lagi hanya builder, tetapi operator dan pemilik produk mereka, bertanggung jawab atas sebagian besar aktivitas yang secara historis terkait dengan tim keuangan dan operasi, termasuk pengadaan dan deployment.

Hal yang diperlukan untuk menyediakan sumber daya cloud adalah akun pengguna, dan serangkaian izin yang tepat. Hal ini juga yang mengurangi risiko IT dan keuangan; yang berarti tim hanya perlu melakukan beberapa klik atau panggilan API untuk menghentikan sumber daya cloud yang tidak digunakan atau tidak diperlukan. Hal ini juga yang memungkinkan tim teknologi berinovasi lebih cepat – ketangkasan dan kemampuan untuk menjalankan dan kemudian menghentikan eksperimen. Sementara sifat variabel penggunaan cloud dapat memengaruhi prediktabilitas dari perspektif

penganggaran dan prakiraan modal, cloud memberi organisasi kemampuan untuk mengurangi biaya penyediaan yang berlebihan, serta mengurangi biaya peluang yang terkait dengan kekurangan penyediaan yang konservatif.

Buat kemitraan antara pemangku kepentingan penting dalam bidang keuangan dan teknologi untuk menghasilkan pemahaman bersama akan tujuan organisasi dan kembangkan mekanisme agar berhasil secara finansial dalam model pengeluaran variabel komputasi cloud. Tim yang relevan dalam organisasi Anda harus dilibatkan dalam diskusi biaya dan penggunaan di semua tahap perjalanan cloud Anda, termasuk:

- Kepala bagian keuangan: CFO, pengontrol keuangan, perencana keuangan, analis bisnis, bagian pengadaan, pengelolaan pemasok, dan utang usaha harus memahami model cloud dari proses penagihan bulanan, opsi pembelian, dan konsumsi. Tim keuangan perlu bermitra dengan tim teknologi untuk membuat dan menyosialisasikan kisah nilai IT, sehingga membantu tim bisnis memahami bagaimana pengeluaran teknologi terkait dengan hasil bisnis. Dengan cara ini,

pengeluaran teknologi tidak dipandang sebagai biaya, melainkan sebagai investasi. Karena perbedaan mendasar antara cloud (seperti laju perubahan dalam penggunaan, harga bayar sesuai pemakaian, harga berjenjang, model harga, dan informasi penggunaan serta penagihan mendetail) dibandingkan dengan operasi on-premise, maka penting organisasi keuangan memahami bagaimana penggunaan cloud dapat memengaruhi aspek bisnis, termasuk proses pengadaan, pelacakan insentif, alokasi biaya, dan laporan keuangan.

- Kepala bagian teknologi: Kepala bagian teknologi (termasuk pemilik aplikasi dan produk) harus sadar akan persyaratan keuangan (misalnya, batas anggaran) serta persyaratan bisnis (misalnya, perjanjian tingkat layanan). Ini memungkinkan beban kerja diimplementasikan untuk mencapai tujuan yang diinginkan organisasi.

Kemitraan antara bagian keuangan dan teknologi memberikan manfaat berikut:

- Tim keuangan dan teknologi memiliki visibilitas hampir dalam waktu nyata ke biaya dan penggunaan.
- Tim keuangan dan teknologi menetapkan prosedur pengoperasian standar untuk menangani varian pengeluaran cloud.
- Pemangku kepentingan dalam bidang keuangan berfungsi sebagai penasihat strategis dalam kaitannya dengan bagaimana modal digunakan untuk membeli diskon komitmen (misalnya, Instans Terpesan atau AWS Savings Plans), dan bagaimana cloud digunakan untuk mengembangkan organisasi.
- Proses pengadaan dan utang usaha yang ada digunakan dengan cloud.
- Tim keuangan dan teknologi berkolaborasi dalam memprediksi AWS biaya dan penggunaan di waktu mendatang untuk menyelaraskan dan menyusun anggaran organisasi.
- Komunikasi lintas organisasi yang lebih baik melalui bahasa bersama, dan pemahaman bersama akan konsep keuangan.

Pemangku kepentingan tambahan di dalam organisasi Anda yang harus dilibatkan dalam diskusi biaya dan penggunaan termasuk:

- Pemilik unit bisnis: Pemilik unit bisnis harus memahami model bisnis cloud sehingga mereka dapat memberikan pengarahan kepada unit bisnis dan seluruh perusahaan. Pengetahuan cloud ini sangat penting ketika ada kebutuhan untuk memprediksi pertumbuhan dan penggunaan beban kerja, dan ketika mengevaluasi opsi pembelian jangka lebih panjang, seperti Instans Terpesan atau Savings Plans.

- Tim rekayasa: Membangun kemitraan antara tim keuangan dan teknologi sangat penting untuk membangun budaya sadar biaya yang mendorong para rekayasawan untuk mengambil tindakan di Manajemen Finansial Cloud (CFM). Salah satu masalah umum dari CFM atau praktisi operasi keuangan dan tim keuangan adalah membuat para rekayasawan memahami seluruh bisnis di cloud, mengikuti praktik terbaik, dan mengambil tindakan yang direkomendasikan.
- Pihak ketiga: Jika organisasi Anda menggunakan pihak ketiga (misalnya, konsultan atau alat), pastikan mereka selaras dengan tujuan keuangan Anda, dan mereka dapat menunjukkan keselarasan tersebut melalui model keterlibatan dan laba atas investasi (ROI) mereka. Umumnya, pihak ketiga akan berkontribusi dalam pelaporan dan analisis beban kerja apa pun yang mereka kelola, dan mereka akan memberikan analisis biaya beban kerja apa pun yang mereka desain.

Menerapkan CFM dan mencapai kesuksesan memerlukan kolaborasi di seluruh tim keuangan, teknologi, dan bisnis, serta perubahan dalam cara pengeluaran cloud dikomunikasikan dan dievaluasi di seluruh organisasi. Ikut sertakan tim rekayasa sehingga mereka dapat menjadi bagian dari diskusi biaya dan penggunaan ini di semua tahap, dan dorong mereka untuk mengikuti praktik terbaik dan mengambil tindakan yang disepakati.

Langkah implementasi

- Tentukan anggota utama: Verifikasi bahwa semua anggota yang relevan dari tim keuangan dan teknologi Anda ambil bagian dalam kemitraan. Anggota yang relevan dari tim keuangan adalah mereka yang memiliki interaksi dengan tagihan cloud. Umumnya ini adalah CFO, pengontrol keuangan, perencana keuangan, analis bisnis, bagian pengadaan, dan pengelolaan pemasok. Umumnya anggota bagian teknologi adalah pemilik aplikasi dan produk, manajer teknis, dan perwakilan dari semua tim yang membangun di cloud. Anggota-anggota lainnya dapat mencakup pemilik unit bisnis, misalnya bagian pemasaran, yang akan memengaruhi penggunaan produk, dan pihak ketiga seperti konsultan, untuk mencapai keselarasan dengan tujuan dan mekanisme Anda, dan untuk membantu dalam pelaporan.
- Tentukan topik untuk diskusi: Tentukan topik yang bersifat umum lintas tim, atau yang akan membutuhkan pemahaman bersama. Ikuti biaya dari waktu biaya dibuat, sampai tagihan dibayar. Catat setiap anggota yang terlibat, dan proses organisasi yang harus diterapkan. Pahami setiap langkah atau proses yang dilewatinya dan informasi terkait, seperti model harga yang tersedia, harga berjenjang, model diskon, penetapan anggaran, dan persyaratan keuangan.
- Adakan koordinasi rutin: Untuk menciptakan kemitraan keuangan dan teknologi, buatlah koordinasi komunikasi yang teratur untuk menciptakan dan mempertahankan keselarasan. Grup tersebut harus mengadakan pertemuan koordinasi secara rutin untuk membahas tujuan dan metrik.

Koordinasi ini biasanya membahas peninjauan status organisasi, program apa pun yang sedang berlangsung, serta metrik pengoptimalan dan keuangan secara keseluruhan. Kemudian beban kerja utama dilaporkan dengan lebih mendetail.

Sumber daya

Dokumen terkait:

- Blog Berita AWS

COST01-BP03 Tetapkan prakiraan dan anggaran cloud

Sesuaikan proses pemrakiraan dan penganggaran yang ada di organisasi agar kompatibel dengan biaya dan penggunaan cloud yang sangat bervariasi. Prosesnya harus dinamis menggunakan algoritma berbasis pendorong bisnis atau berbasis tren, atau kombinasi dari keduanya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Pelanggan menggunakan cloud untuk efisiensi, kecepatan, dan ketangkasan, yang menimbulkan biaya dan penggunaan yang sangat bervariasi. Biaya dapat berkurang (atau terkadang bertambah) seiring dengan peningkatan efisiensi beban kerja, atau seiring dengan deployment fitur dan beban kerja baru. Beban kerja dapat diskalakan untuk melayani lebih banyak pelanggan, yang akan meningkatkan biaya dan penggunaan cloud Anda. Sumber daya sekarang lebih mudah diakses daripada sebelumnya. Elastisitas cloud juga menghadirkan elastisitas biaya dan prakiraan. Proses penganggaran organisasi yang sudah ada harus dimodifikasi untuk menyertakan variabilitas ini.

Anggaran biasanya disiapkan untuk satu tahun dan tidak berubah-ubah, sehingga membutuhkan kepatuhan yang ketat dari semua orang yang terlibat. Sebaliknya, pemrakiraan lebih fleksibel, sehingga memungkinkan penyesuaian ulang sepanjang tahun dan memberikan proyeksi dinamis selama periode satu, dua, atau tiga tahun. Baik penganggaran maupun pemrakiraan memainkan peran penting dalam membangun ekspektasi keuangan di kalangan berbagai pemangku kepentingan teknologi dan bisnis. Pemrakiraan dan implementasi yang akurat juga menghadirkan akuntabilitas bagi para pemangku kepentingan yang bertanggung jawab langsung atas penyediaan biaya sejak awal, serta dapat meningkatkan kesadaran biaya mereka secara keseluruhan.

Sesuaikan proses pemrakiraan dan penganggaran yang ada agar menjadi lebih dinamis menggunakan algoritma berbasis tren (menggunakan biaya historis sebagai masukan), atau

menggunakan algoritma berbasis pendorong bisnis (misalnya, peluncuran produk baru atau ekspansi Regional), yang ideal untuk lingkungan penganggaran dinamis dan bervariasi, atau kombinasi antara pendorong bisnis dan tren.

Anda dapat menggunakan AWS Cost Explorer untuk pemrakiraan berbasis tren dalam rentang waktu mendatang yang ditentukan berdasarkan pengeluaran Anda sebelumnya. Mesin prakiraan AWS Cost Explorer menyegmentasikan data historis Anda berdasarkan jenis tagihan (misalnya, Instans Terpesan) dan menggunakan kombinasi machine learning dan model berbasis aturan untuk memprediksi pengeluaran di semua jenis tagihan satu per satu.

Identifikasi pendorong bisnis yang dapat berdampak pada biaya penggunaan Anda dan lakukan prakiraan untuk masing-masing secara terpisah untuk memastikan bahwa perkiraan penggunaan sudah diperhitungkan di awal. Beberapa pendorong terkait dengan tim IT dan produk di dalam organisasi. Pendorong bisnis lainnya, seperti acara pemasaran, promosi, merger, dan akuisisi, dikenal oleh pemimpin penjualan, pemasaran, dan bisnis Anda, dan penting juga untuk berkolaborasi dan memperhitungkan semua pendorong permintaan tersebut. Anda perlu bekerja sama dengan mereka untuk memahami dampaknya terhadap pendorong internal baru.

Setelah Anda menentukan prakiraan berbasis tren menggunakan Cost Explorer atau alat lainnya, gunakan AWS Kalkulator Harga untuk memperkirakan kasus penggunaan AWS dan biaya mendatang berdasarkan pada penggunaan yang diharapkan (lalu lintas, permintaan per detik, atau instans Amazon EC2 yang diperlukan). Anda juga dapat menggunakannya untuk merencanakan pengeluaran, menemukan peluang penghematan biaya, dan membuat keputusan yang tepat saat menggunakan AWS. Penting untuk melacak keakuratan prakiraan tersebut karena anggaran harus ditetapkan berdasarkan perhitungan dan estimasi prakiraan ini.

Gunakan AWS Budgets untuk menetapkan anggaran khusus pada tingkat mendetail dengan menentukan periode waktu, pengulangan, atau jumlah (tetap atau variabel), dan menambahkan filter seperti layanan, Wilayah AWS, dan tag. Untuk tetap mendapatkan informasi tentang performa anggaran yang ada, Anda dapat membuat dan menjadwalkan Laporan AWS Budgets agar dikirim melalui email kepada Anda dan pemangku kepentingan secara teratur. Anda juga dapat membuat Peringatan AWS Budgets berdasarkan biaya aktual, yang bersifat reaktif, atau biaya yang diprakirakan, yang menyediakan waktu untuk menerapkan mitigasi terhadap potensi kelebihan biaya. Anda dapat menerima peringatan saat biaya atau penggunaan melampaui, atau jika diprakirakan akan melampaui, jumlah yang dianggarkan.

Gunakan AWS Cost Anomaly Detection untuk mencegah atau mengurangi biaya tak terduga dan meningkatkan kontrol tanpa memperlambat inovasi. AWS Cost Anomaly Detection memanfaatkan

machine learning untuk mengidentifikasi pengeluaran yang tidak wajar dan akar penyebab, agar Anda dapat mengambil tindakan secara cepat. Dengan tiga langkah sederhana, Anda dapat membuat pemantauan kontekstual Anda sendiri dan menerima peringatan saat pengeluaran yang tidak wajar terdeteksi.

Sebagaimana disebutkan dalam pilar optimasi biaya Well-Architected bagian Kemitraan Keuangan dan Teknologi, penting untuk memiliki kemitraan dan koordinasi antara IT, keuangan, dan pemangku kepentingan lainnya untuk memverifikasi bahwa mereka semua menggunakan alat atau proses yang sama untuk konsistensi. Dalam kasus yang mungkin memerlukan perubahan anggaran, menambah frekuensi rapat singkat koordinasi dapat membantu bereaksi terhadap perubahan tersebut secara lebih cepat.

Langkah implementasi

- Analisis prakiraan berbasis tren: Gunakan alat prakiraan berbasis tren pilihan seperti AWS Cost Explorer dan Amazon Forecast. Analisis biaya penggunaan Anda pada dimensi-dimensi yang berbeda seperti kategori layanan, akun, tag, dan biaya. Jika prakiraan lanjutan diperlukan, impor data AWS Cost and Usage Report Anda ke Amazon Forecast (yang menerapkan regresi linier sebagai bentuk machine learning ke prakiraan).
- Analisis prakiraan berbasis pendorong: Identifikasi dampak pendorong bisnis terhadap penggunaan cloud Anda dan lakukan prakiraan pada masing-masing secara terpisah untuk menghitung perkiraan biaya penggunaan di awal. Bekerjasamalah dengan pemilik unit bisnis dan pemangku kepentingan untuk memahami dampak pada pendorong baru dan menghitung perkiraan perubahan biaya untuk menentukan anggaran yang akurat.
- Perbarui proses pemrakiraan dan penganggaran yang ada: Tetapkan proses penganggaran prakiraan Anda berdasarkan metode prakiraan yang diadopsi seperti metode berbasis tren, berbasis pendorong bisnis, atau kombinasi keduanya. Anggaran harus terhitung dan realistis, berdasarkan proses pemrakiraan ini.
- Konfigurasikan peringatan dan notifikasi: Gunakan Pemberitahuan AWS Budgets dan AWS Cost Anomaly Detection untuk mendapatkan pemberitahuan dan notifikasi.
- Lakukan peninjauan rutin dengan pemangku kepentingan utama: Misalnya, pemangku kepentingan di tim IT, keuangan, platform, dan bidang bisnis lainnya, harus menyelaraskan dengan perubahan dalam arah dan penggunaan bisnis.

Sumber daya

Dokumen terkait:

- AWS Cost Explorer
- AWS Cost and Usage Report
- Prakiraan Quick
- Amazon Forecast
- AWS Budgets
- Blog Berita AWS

Video terkait:

- Cara Menggunakan AWS Budgets untuk melacak pengeluaran dan penggunaan
- Serial Optimasi Biaya AWS: AWS Budgets

Contoh terkait:

- Memahami dan membangun prakiraan berbasis pendorong
- Cara membangun dan mendorong budaya prakiraan
- Cara meningkatkan prakiraan biaya cloud Anda
- Menggunakan alat yang tepat untuk prakiraan biaya cloud Anda

COST01-BP04 Mengimplementasikan kesadaran biaya dalam proses organisasi Anda

Implementasikan kesadaran biaya, transparansi, dan akuntabilitas biaya ke dalam proses baru atau yang sudah ada yang memengaruhi penggunaan, serta manfaatkan proses yang sudah ada untuk kesadaran biaya. Implementasikan kesadaran biaya ke dalam pelatihan karyawan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

Kesadaran biaya harus diimplementasikan di dalam proses-proses baru dan yang sudah ada. Ini adalah salah satu kemampuan dasar dan prasyarat untuk praktik terbaik lainnya. Disarankan untuk menggunakan ulang dan memodifikasi proses yang sudah ada jika memungkinkan—hal ini meminimalkan dampak terhadap ketangkasan dan kecepatan. Laporkan biaya cloud kepada tim teknologi dan pengambil keputusan di tim bisnis dan keuangan untuk meningkatkan kesadaran biaya, dan tetapkan indikator kinerja utama (KPI) efisiensi untuk pemangku kepentingan keuangan dan

bisnis. Saran berikut ini akan membantu mengimplementasikan kesadaran biaya di dalam beban kerja Anda:

- Pastikan manajemen perubahan mencakup pengukuran biaya untuk menghitung dampak keuangan dari perubahan Anda. Hal ini membantu Anda secara proaktif menangani masalah terkait biaya dan menyoroti penghematan biaya.
- Pastikan pengoptimalan biaya adalah komponen inti kemampuan operasional Anda. Misalnya, Anda dapat memanfaatkan proses manajemen insiden yang sudah ada untuk menyelidiki dan mengidentifikasi akar masalah untuk anomali biaya dan penggunaan atau kelebihan biaya.
- Percepat penghematan biaya dan realisasi nilai bisnis melalui otomatisasi atau penggunaan alat. Saat membahas biaya implementasi, atur percakapan agar turut membahas komponen laba atas investasi (ROI) guna mencari pembenaran atas investasi waktu atau uang.
- Alokasikan biaya cloud dengan menerapkan showback atau chargeback untuk pengeluaran cloud, termasuk pengeluaran untuk opsi pembelian berbasis komitmen, layanan bersama, dan pembelian marketplace untuk mendukung penggunaan cloud yang paling sadar biaya.
- Sertakan pelatihan tentang kesadaran biaya dalam program pelatihan dan pengembangan yang sudah ada di seluruh organisasi Anda. Disarankan untuk menyertakan pelatihan dan sertifikasi berkelanjutan di dalamnya. Hal ini akan menghasilkan sebuah organisasi yang mampu mengelola biaya dan penggunaan secara mandiri.
- Manfaatkan alat native AWS gratis seperti AWS Cost Anomaly Detection, AWS Budgets, dan Laporan AWS Budgets.

Saat organisasi secara konsisten menerapkan praktik Manajemen Finansial Cloud (CFM), perilaku tersebut tertanam dalam cara kerja dan pengambilan keputusan. Hasilnya adalah budaya yang lebih sadar biaya, mulai dari pengembang yang merancang aplikasi yang baru di cloud, hingga manajer keuangan yang menganalisis ROI pada investasi cloud baru ini.

Langkah implementasi

- Identifikasi proses organisasi yang relevan: Setiap unit organisasi meninjau proses mereka dan mengidentifikasi proses yang berdampak pada biaya dan penggunaan. Proses apa pun yang memunculkan pembuatan atau penghentian sumber daya perlu turut ditinjau. Cari proses yang dapat mendukung kesadaran biaya di dalam bisnis Anda, seperti manajemen dan pelatihan insiden.
- Membangun budaya swadaya dan sadar biaya: Pastikan semua pemangku kepentingan yang relevan menyelaraskan dengan penyebab perubahan dan dampak sebagai biaya sehingga mereka

memahami biaya cloud. Ini akan memungkinkan organisasi Anda membangun budaya inovasi swadaya dan sadar biaya.

- Perbarui proses dengan kesadaran biaya: Setiap proses dimodifikasi agar menjadi proses yang sadar biaya. Proses mungkin memerlukan pemeriksaan awal tambahan, seperti penilaian dampak biaya, atau pemeriksaan akhir yang memvalidasi terjadinya perubahan yang diharapkan dalam hal biaya dan penggunaan. Proses pendukung seperti pelatihan dan manajemen insiden dapat dibuat agar menyertakan item-item untuk biaya dan penggunaan.

Untuk mendapatkan bantuan, hubungi pakar CFM melalui tim Akun Anda, atau jelajahi sumber daya dan dokumen terkait di bawah.

Sumber daya

Dokumen terkait:

- Manajemen Finansial Cloud AWS

Contoh terkait:

- Strategi untuk Manajemen Biaya Cloud Efisien
- Seri Blog Pengendalian Biaya No. 3: Cara Menangani Biaya Tak Terduga
- Panduan Pemula untuk AWS Cost Management

COST01-BP05 Melaporkan dan memberi tahu tentang pengoptimalan biaya

Siapkan anggaran cloud dan konfigurasikan mekanisme untuk mendeteksi anomali dalam penggunaan. Konfigurasikan alat-alat terkait untuk peringatan biaya dan penggunaan sesuai target yang telah ditentukan sebelumnya dan terima notifikasi ketika terdapat penggunaan yang melebihi target tersebut. Lakukan pertemuan rutin untuk menganalisis efektivitas biaya beban kerja Anda dan meningkatkan kesadaran biaya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Anda harus secara rutin melaporkan pengoptimalan biaya dan penggunaan di dalam organisasi Anda. Anda dapat mengimplementasikan sesi-sesi khusus untuk membahas kinerja biaya, atau

sertakan pengoptimalan biaya di dalam siklus pelaporan operasional rutin untuk beban kerja Anda. Gunakan layanan dan alat untuk memantau kinerja biaya Anda secara teratur dan menerapkan peluang penghematan biaya.

Lihat biaya dan penggunaan Anda dengan beberapa filter dan perincian dengan menggunakan AWS Cost Explorer, yang menyediakan dasbor dan laporan seperti biaya berdasarkan layanan atau akun, biaya harian, atau biaya pasar. Lacak progres biaya dan penggunaan Anda berdasarkan anggaran yang dikonfigurasi dengan Laporan AWS Budgets.

Gunakan AWS Budgets untuk mengatur anggaran khusus dalam melacak biaya dan penggunaan Anda serta menanggapi dengan cepat pemberitahuan yang diterima dari email atau notifikasi Amazon Simple Notification Service (Amazon SNS) jika Anda melebihi ambang batas Anda. Tetapkan periode anggaran sesuai keinginan Anda, yaitu harian, bulanan, triwulanan, atau tahunan, dan buat batas anggaran khusus agar terus mendapatkan informasi tentang progres biaya dan penggunaan aktual atau yang diprakirakan berdasarkan ambang batas anggaran Anda. Anda juga dapat mengatur peringatan dan tindakan terhadap pemberitahuan tersebut agar berjalan secara otomatis atau melalui proses persetujuan saat target anggaran terlampaui.

Implementasikan notifikasi biaya dan penggunaan untuk memastikan perubahan biaya dan penggunaan dapat ditindaklanjuti secara cepat jika tidak terduga. AWS Cost Anomaly Detection memungkinkan Anda mengurangi biaya tak terduga dan meningkatkan kontrol tanpa memperlambat inovasi. AWS Cost Anomaly Detection mengidentifikasi pengeluaran yang tidak wajar dan akar masalah, yang membantu mengurangi risiko penagihan tak terduga. Dengan tiga langkah mudah, Anda dapat membuat pemantauan kontekstual Anda sendiri dan menerima peringatan saat pengeluaran yang tidak wajar terdeteksi.

Anda juga dapat menggunakan Quick dengan data AWS Cost and Usage Report (CUR), untuk memberikan pelaporan yang sangat disesuaikan dengan data yang lebih mendetail. Quick memungkinkan Anda menjadwalkan laporan dan menerima email Laporan Biaya berkala untuk biaya dan penggunaan historis, atau peluang penghematan biaya. Coba solusi Dasbor Inteligensi Biaya (CID) kami yang dibangun berdasarkan Quick, yang memberi Anda visibilitas lanjutan.

Gunakan AWS Trusted Advisor, yang memberikan panduan untuk memverifikasi apakah sumber daya yang disediakan selaras dengan praktik terbaik AWS untuk pengoptimalan biaya.

Lihat rekomendasi Savings Plans Anda melalui grafik visual berdasarkan biaya dan penggunaan mendetail Anda. Grafik per jam menunjukkan pengeluaran Sesuai Permintaan beserta komitmen Savings Plans yang disarankan, sehingga memberikan wawasan tentang perkiraan penghematan, cakupan Savings Plans, dan pemanfaatan Savings Plans. Ini membantu organisasi untuk

memahami bagaimana Savings Plans mereka berlaku untuk setiap jam pengeluaran tanpa harus menginvestasikan waktu dan sumber daya untuk membangun model-model analisis pengeluaran.

Buat laporan secara berkala yang berisi sorotan Savings Plans, Instans Terpesan, dan rekomendasi penyesuaian ukuran Amazon EC2 dari AWS Cost Explorer untuk mulai mengurangi biaya yang terkait dengan beban kerja berkondisi stabil serta sumber daya yang tidak aktif dan kurang dimanfaatkan. Identifikasi dan dapatkan kembali pengeluaran yang terkait dengan pemborosan cloud untuk sumber daya yang di-deploy. Pemborosan cloud terjadi saat sumber daya berukuran tidak tepat dibuat atau ditemukan pola penggunaan yang berbeda dari yang diharapkan. Ikuti praktik terbaik AWS untuk mengurangi pemborosan Anda atau minta tim akun dan partner Anda untuk membantu Anda mengoptimalkan serta menghemat biaya cloud Anda.

Buat laporan secara teratur untuk opsi pembelian yang lebih baik bagi sumber daya Anda guna menurunkan biaya unit untuk beban kerja Anda. Opsi pembelian seperti Savings Plans, Instans Terpesan, atau Instans Spot Amazon EC2 menawarkan penghematan biaya terbesar untuk beban kerja yang toleran terhadap kesalahan dan memungkinkan pemangku kepentingan (pemilik bisnis, tim keuangan, dan tim teknologi) untuk menjadi bagian dari diskusi komitmen ini.

Bagikan laporan yang berisi peluang atau pengumuman rilis baru yang dapat membantu Anda mengurangi total biaya kepemilikan (TCO) cloud. Terapkan layanan baru, Wilayah, fitur, solusi, atau cara baru untuk mencapai pengurangan biaya lebih lanjut.

Langkah implementasi

- Mengonfigurasi AWS Budgets: Konfigurasikan AWS Budgets di semua akun untuk beban kerja Anda. Tetapkan anggaran untuk keseluruhan pengeluaran akun, serta anggaran untuk beban kerja menggunakan tag.
  - Lab Well-Architected: Biaya dan Penggunaan Tata Kelola
- Melaporkan pengoptimalan biaya: Siapkan siklus rutin untuk membahas dan menganalisis efisiensi beban kerja. Menggunakan metrik yang telah dibangun, buat laporan tentang metrik-metrik yang dicapai serta biaya untuk mencapainya. Identifikasi dan perbaiki tren negatif apa pun, serta tren positif yang dapat Anda pupuk di seluruh organisasi Anda. Pelaporan harus melibatkan perwakilan dari tim dan pemilik aplikasi, keuangan, dan pengambil keputusan utama sehubungan dengan pengeluaran cloud.

Sumber daya

Dokumen terkait:

- AWS Cost Explorer
- AWS Trusted Advisor
- AWS Budgets
- AWS Cost and Usage Report
- Praktik Terbaik AWS Budgets
- Analitik Amazon S3

Contoh terkait:

- Lab Well-Architected: Biaya dan Penggunaan Tata Kelola
- Cara utama untuk mulai mengoptimalkan biaya cloud AWS

COST01-BP06 Memantau biaya secara proaktif

Gunakan alat dan dasbor untuk memantau biaya beban kerja secara proaktif. Tinjau biaya secara teratur dengan alat yang dikonfigurasi atau alat siap pakai, jangan hanya memeriksa biaya dan kategori saat Anda menerima notifikasinya. Memantau dan menganalisis biaya secara proaktif akan membantu mengidentifikasi tren positif dan memungkinkan Anda mempromosikan tren tersebut ke seluruh organisasi Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

Sebaiknya pantau biaya dan penggunaan di organisasi Anda secara proaktif, tidak hanya saat terdapat pengecualian atau anomali. Dasbor yang dapat dilihat dari kantor atau lingkungan kerja Anda memastikan orang yang memiliki peran penting dapat mengakses informasi yang diperlukan, serta menunjukkan bahwa organisasi Anda fokus pada pengoptimalan biaya. Dengan dasbor yang terlihat ini, Anda dapat mendorong hasil secara aktif dan menerapkannya di seluruh organisasi Anda.

Buat rutinitas harian atau berulang untuk menggunakan AWS Cost Explorer atau dasbor lain seperti Amazon Quick untuk melihat biaya dan menganalisis secara proaktif. Lakukan analisis penggunaan dan biaya layanan AWS pada tingkat akun AWS, tingkat beban kerja, atau tingkat layanan AWS tertentu dengan pengelompokan dan pemfilteran, dan lakukan validasi apakah penggunaan dan biaya tersebut diharapkan atau tidak. Gunakan perincian dan tag tingkat sumber daya dan per jam untuk memfilter dan mengidentifikasi biaya yang timbul untuk sumber daya teratas. Anda juga dapat

membuat laporan Anda sendiri dengan Dasbor Inteligensi Biaya, solusi Amazon Quick yang dibuat Arsitek Solusi AWS, dan bandingkan anggaran Anda dengan biaya dan penggunaan aktual.

Langkah implementasi

- Melaporkan pengoptimalan biaya: Siapkan siklus rutin untuk membahas dan menganalisis efisiensi beban kerja. Menggunakan metrik yang dibuat, laporkan metrik-metrik yang dicapai serta biaya untuk mencapainya. Identifikasikan dan perbaiki tren negatif apa pun, serta identifikasi tren positif yang dapat Anda pupuk di seluruh organisasi Anda. Pelaporan harus melibatkan perwakilan dari tim dan pemilik aplikasi, keuangan, dan manajemen.
- Buat dan aktifkan AWS Budgets perincian harian agar biaya dan penggunaan mengambil tindakan tepat waktu untuk mencegah kemungkinan kelebihan biaya: AWS Budgets memungkinkan Anda mengonfigurasi notifikasi peringatan, sehingga Anda terus mendapatkan informasi terbaru jika salah satu jenis anggaran Anda berada di luar ambang batas yang sudah dikonfigurasi sebelumnya. Cara terbaik untuk memanfaatkan AWS Budgets adalah menetapkan biaya dan penggunaan yang diharapkan sebagai batas Anda, agar nilai apa pun di atas anggaran Anda dapat dianggap sebagai pengeluaran yang berlebihan.
- Buat AWS Cost Anomaly Detection untuk pemantauan biaya: AWS Cost Anomaly Detection menggunakan teknologi Machine Learning canggih untuk mengidentifikasi pengeluaran yang tidak wajar dan akar masalah, sehingga Anda dapat mengambil tindakan secara cepat. Hal ini memungkinkan Anda mengonfigurasi pemantauan biaya yang menentukan segmen pengeluaran yang ingin Anda evaluasi (misalnya, tiap-tiap layanan AWS, akun anggota, tag alokasi biaya, dan kategori biaya), dan memungkinkan Anda mengatur kapan, di mana, dan bagaimana Anda menerima notifikasi peringatan. Untuk setiap pemantauan, kaitkan beberapa langganan peringatan untuk pemilik bisnis dan tim teknologi, termasuk nama, ambang batas dampak biaya, dan frekuensi peringatan (tiap-tiap peringatan, ringkasan harian, ringkasan mingguan) untuk setiap langganan.
- Gunakan AWS Cost Explorer atau integrasikan data AWS Cost and Usage Report (CUR) dengan dasbor Amazon Quick untuk memvisualisasikan biaya organisasi: AWS Cost Explorer memiliki antarmuka yang mudah digunakan yang memungkinkan Anda memvisualisasikan, memahami, dan mengelola biaya dan penggunaan AWS dari waktu ke waktu. Fitur Dasbor Inteligensi Biaya adalah dasbor yang dapat disesuaikan dan dapat diakses untuk membantu membuat landasan alat manajemen dan pengoptimalan biaya Anda sendiri.

Sumber daya

Dokumen terkait:

- AWS Budgets
- AWS Cost Explorer
- Anggaran Biaya dan Penggunaan Harian
- AWS Cost Anomaly Detection

Contoh terkait:

- Lab Well-Architected: Visualisasi
- Lab Well-Architected: Visualisasi Tingkat Lanjut
- Lab Well-Architected: Dasbor Inteligensi Cloud
- Lab Well-Architected: Visualisasi Biaya
- Peringatan AWS Cost Anomaly Detection dengan Slack

COST01-BP07 Mengikuti perkembangan rilisan layanan baru

Berkonsultasilah secara rutin dengan ahli atau Partner AWS untuk mempertimbangkan layanan dan fitur mana yang dapat menekan biaya. Pelajari blog AWS dan sumber informasi lain.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

Panduan implementasi

AWS terus menambahkan kemampuan baru sehingga Anda dapat memanfaatkan teknologi terkini untuk bereksperimen dan berinovasi secara lebih cepat. Anda mungkin dapat mengimplementasikan layanan dan fitur AWS baru untuk meningkatkan efisiensi biaya di beban kerja Anda. Tinjau secara rutin Manajemen Biaya AWS, Blog Berita AWS, Blog Manajemen Biaya AWS, dan Apa yang Baru dengan AWS untuk mendapatkan informasi tentang rilisan layanan dan fitur baru. Postingan Apa yang Baru memberikan gambaran singkat tentang semua pengumuman perluasan layanan, fitur, dan Wilayah AWS yang dirilis.

Langkah implementasi

- Berlangganan ke blog: Kunjungi halaman blog AWS dan berlangganan ke Blog “Apa yang Baru” dan blog relevan lain. Anda dapat mendaftar di halaman preferensi komunikasi dengan alamat email Anda.

- Berlangganan ke Berita AWS: Tinjau secara rutin Blog Berita AWS dan Apa yang Baru dengan AWS untuk mendapatkan informasi tentang rilisan layanan dan fitur baru. Berlangganan umpan RSS atau gunakan email Anda untuk mengikuti pengumuman dan rilis.
- Ikuti Penurunan Harga AWS: Potongan harga reguler pada semua layanan kami sudah menjadi cara standar bagi AWS untuk melimpahkan efisiensi ekonomi yang diperoleh dari skala kami kepada pelanggan. Sejak April 2022, AWS telah menurunkan harga 115 kali sejak diluncurkan pada 2006. Jika Anda memiliki keputusan bisnis yang tertunda karena masalah harga, Anda dapat meninjaunya kembali setelah pengurangan harga dan integrasi layanan baru. Anda dapat mempelajari tentang upaya pengurangan harga sebelumnya, termasuk instans Amazon Elastic Compute Cloud (Amazon EC2), dalam kategori pengurangan harga dalam Blog Berita AWS.
- Acara dan pertemuan AWS: Hadiri AWS Summit, dan pertemuan lokal dengan organisasi lainnya dari area setempat. Jika Anda tidak dapat hadir secara langsung, coba hadiri acara virtual untuk mendapatkan berbagai informasi dari pakar AWS dan mengetahui kasus bisnis pelanggan lainnya.
- Bertemu dengan tim akun Anda: Jadwalkan koordinasi rutin dengan tim akun Anda, adakan pertemuan, serta diskusikan tren industri dan layanan AWS. Bicarakan dengan manajer akun, Arsitek Solusi, dan tim dukungan Anda.

Sumber daya

Dokumen terkait:

- Manajemen Biaya AWS
- Apa yang Baru dengan AWS
- Blog Berita AWS

Contoh terkait:

- Amazon EC2 – Mengoptimalkan dan Menghemat Biaya IT Anda Selama 15 Tahun
- Blog Berita AWS - Pengurangan Harga

COST01-BP08 Menciptakan budaya sadar biaya

Implementasikan perubahan atau program untuk menciptakan budaya sadar biaya di seluruh organisasi. Sebaiknya mulai dari cakupan kecil. Kemudian, seiring meningkatnya kemampuan dan penggunaan cloud oleh organisasi Anda, implementasikan program dengan cakupan yang lebih luas dan besar.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Rendah

Panduan implementasi

Budaya sadar biaya memungkinkan Anda menskalakan pengoptimalan biaya dan Manajemen Finansial Cloud (tim operasi keuangan, pusat keunggulan cloud, operasi cloud, dan sebagainya) melalui praktik terbaik yang dilakukan secara organik dan terdesentralisasi di seluruh organisasi Anda. Kesadaran biaya ini memungkinkan Anda membuat kemampuan tingkat tinggi di seluruh organisasi dengan upaya yang minimal, dibandingkan dengan pendekatan tersentralisasi dari atas ke bawah yang kaku.

Menghadirkan kesadaran biaya dalam komputasi cloud, terutama untuk pendorong biaya utama dalam komputasi cloud, memungkinkan tim memahami hasil yang diharapkan untuk setiap perubahan dalam perspektif biaya. Tim yang mengakses lingkungan cloud harus mengetahui model harga dan perbedaan antara pusat data on-premise tradisional dan komputasi cloud.

Manfaat utama budaya sadar biaya adalah bahwa tim teknologi mengoptimalkan biaya secara proaktif dan secara kontinu (misalnya, biaya tersebut dianggap sebagai persyaratan nonfungsional saat merancang beban kerja baru atau membuat perubahan pada beban kerja yang ada), bukan melakukan pengoptimalan biaya reaktif seperlunya.

Perubahan kecil dalam budaya bisa berdampak besar terhadap efisiensi beban kerja saat ini dan masa mendatang. Contohnya adalah:

- Memberikan visibilitas dan memunculkan kesadaran dalam tim rekayasa untuk memahami fungsi dan dampak budaya tersebut dari segi biaya.
- Menerapkan mekanisme yang kompetitif pada biaya dan penggunaan di seluruh organisasi. Hal ini dapat dilakukan melalui dasbor yang terlihat secara publik, atau laporan yang membandingkan biaya dan penggunaan normal di antara berbagai tim (misalnya, biaya per beban kerja, biaya per transaksi).
- Penghargaan atas efisiensi biaya. Berikan penghargaan atas pencapaian pengoptimalan biaya yang dilakukan dengan sukarela atau tanpa diminta baik secara publik maupun privat, dan belajar dari kesalahan agar tidak mengulanginya di masa depan.
- Membuat persyaratan organisasi dari atas ke bawah untuk beban kerja yang dijalankan dengan anggaran yang telah ditentukan sebelumnya.
- Mempertanyakan kebutuhan bisnis yang akan dipenuhi oleh perubahan, dan dampak biaya dari perubahan yang diminta pada infrastruktur arsitektur atau konfigurasi beban kerja untuk memastikan Anda hanya membayar sesuai kebutuhan.

- Memastikan perencana perubahan mengetahui perubahan yang diharapkan yang memiliki dampak biaya, dan bahwa perubahan tersebut dikonfirmasi oleh pemangku kepentingan untuk memberikan hasil bisnis dengan biaya yang efektif.

Langkah implementasi

- Laporkan biaya cloud ke tim teknologi: Untuk meningkatkan kesadaran biaya, dan menetapkan KPI efisiensi bagi pemangku kepentingan keuangan dan bisnis.
- Beri tahu pemangku kepentingan atau anggota tim tentang perubahan yang direncanakan: Buat item agenda untuk membahas perubahan yang direncanakan dan dampak manfaat biaya pada beban kerja selama rapat perubahan mingguan.
- Bertemu dengan tim akun Anda: Koordinasikan rapat rutin dengan tim akun Anda, dan diskusikan tren industri dan layanan AWS. Bicarakan dengan manajer akun, arsitek, dan tim dukungan Anda.
- Bagikan kisah sukses: Bagikan kisah sukses tentang pengurangan biaya untuk beban kerja, Akun AWS, atau organisasi apa pun untuk mendukung sikap dan dorongan positif seputar pengoptimalan biaya.
- Pelatihan: Pastikan tim teknis atau anggota tim mendapatkan pelatihan terkait kesadaran biaya sumber daya di AWS Cloud.
- Acara dan pertemuan AWS: Hadiri AWS Summit lokal, dan pertemuan lokal dengan organisasi lainnya dari area setempat.
- Berlangganan ke blog: Buka halaman blog AWS dan berlangganan Blog “Apa yang Baru” dan blog lain yang relevan untuk mengikuti rilis baru, implementasi, contoh, dan perubahan yang diinformasikan oleh AWS.

Sumber daya

Dokumen terkait:

- Blog AWS
- Manajemen Biaya AWS
- Blog Berita AWS

Contoh terkait:

- Manajemen Finansial Cloud AWS

- Lab AWS Well-Architected: Manajemen Finansial Cloud

COST01-BP09 Menghitung nilai bisnis dari optimasi biaya

Penghitungan nilai bisnis dari optimasi biaya memungkinkan Anda memahami seluruh set keuntungan untuk organisasi Anda. Karena optimasi biaya adalah investasi yang diperlukan, penghitungan nilai bisnis memungkinkan Anda untuk menjelaskan laba atas investasi kepada pemangku kepentingan. Penghitungan nilai bisnis dapat membantu Anda memperoleh lebih banyak dukungan dari pemangku kepentingan untuk investasi optimasi biaya mendatang, dan menyediakan kerangka kerja untuk mengukur hasil bagi aktivitas optimasi biaya organisasi Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Menghitung nilai bisnis berarti mengukur manfaat yang diperoleh bisnis dari tindakan dan keputusan yang mereka ambil. Nilai bisnis dapat berwujud (seperti berkurangnya biaya atau meningkatnya keuntungan) atau tidak berwujud (seperti meningkatnya reputasi merek atau meningkatnya kepuasan pelanggan).

Menghitung nilai bisnis dari pengoptimalan biaya berarti menentukan seberapa besar nilai atau manfaat yang Anda dapatkan dari upaya Anda mengefisiensikan pengeluaran. Misalnya, jika perusahaan mengeluarkan 100.000 USD untuk melakukan deployment beban kerja di AWS lalu mengoptimalkannya, biaya baru menjadi hanya 80.000 USD tanpa mengorbankan kualitas atau output. Dalam skenario ini, nilai bisnis yang terhitung dari pengoptimalan biaya adalah penghematan sebesar 20.000 USD. Namun selain penghematan, bisnis mungkin juga menghitung nilai dalam hal waktu pengiriman yang lebih cepat, kepuasan pelanggan yang meningkat, atau metrik lain yang dihasilkan dari upaya pengoptimalan biaya. Pemangku kepentingan perlu membuat keputusan tentang potensi nilai pengoptimalan biaya, biaya untuk mengoptimalkan beban kerja, dan nilai labanya.

Selain melaporkan penghematan dari optimasi biaya, Anda direkomendasikan untuk menghitung nilai tambah yang dihadirkan. Keuntungan optimasi biaya biasanya dihitung dari biaya terendah per hasil bisnis. Sebagai contoh, Anda bisa menghitung penghematan biaya Amazon Elastic Compute Cloud(Amazon EC2) ketika Anda membeli Savings Plans, yang mengurangi biaya dan mempertahankan tingkat output beban kerja. Anda dapat menghitung pengurangan biaya pada pengeluaran untuk AWS ketika instans Amazon EC2 yang tidak aktif dihapus, atau volume Amazon Elastic Block Store (Amazon EBS) yang tidak dilampirkan dihapus.

Namun, keuntungan dari optimasi biaya bukan sekadar pengurangan atau peniadaan biaya. Pertimbangkan pengambilan data tambahan untuk mengukur peningkatan efisiensi dan nilai bisnis.

Langkah implementasi

- Evaluasi manfaat bisnis: Ini adalah proses menganalisis dan menyesuaikan biaya AWS Cloud dengan cara yang memaksimalkan manfaat yang diterima dari setiap nominal yang dikeluarkan. Alih-alih berfokus pada pengurangan biaya tanpa nilai bisnis, pertimbangkan manfaat bisnis dan laba atas investasi untuk pengoptimalan biaya, yang dapat membawa lebih banyak nilai dari uang yang Anda keluarkan. Intinya adalah mengeluarkan uang dengan bijak dan melakukan investasi dan pengeluaran di area yang paling banyak menghasilkan laba.
- Analisis prakiraan biaya AWS: Dengan melakukan prakiraan, pemangku kepentingan keuangan dapat menetapkan ekspektasi dengan pemangku kepentingan organisasi internal dan eksternal lain, dan membantu meningkatkan prediktabilitas finansial organisasi Anda. AWS Cost Explorer dapat digunakan untuk melakukan prakiraan biaya dan penggunaan Anda.

Sumber daya

Dokumen terkait:

- Ekonomi AWS Cloud
- Blog AWS
- Manajemen Biaya AWS
- Blog Berita AWS
- laporan resmi Pilar Keandalan Well-Architected
- AWS Cost Explorer

Video terkait:

- Unlock Business Value with Windows on AWS

Contoh terkait:

- Measuring and Maximizing the Business Value of Customer 360
- The Business Value of Adopting Amazon Web Services Managed Databases
- The Business Value of Amazon Web Services for Independent Software Vendors

- Business Value of Cloud Modernization
- The Business Value of Migration to Amazon Web Services

Kesadaran akan penggunaan dan pengeluaran

Pertanyaan

- COST 2. Bagaimana cara mengatur penggunaan?
- COST 3. Bagaimana cara memantau biaya dan penggunaan Anda?
- COST 4. Bagaimana cara melakukan penonaktifan sumber daya?

COST 2. Bagaimana cara mengatur penggunaan?

Tetapkan kebijakan dan mekanisme untuk memverifikasi bahwa biaya yang ditimbulkan memang diperlukan untuk mencapai tujuan. Dengan menerapkan pendekatan yang mengontrol dan menjaga keseimbangan, Anda dapat berinovasi tanpa mengeluarkan dana yang berlebihan.

Praktik terbaik

- COST02-BP01 Mengembangkan kebijakan berdasarkan keperluan organisasi Anda
- COST02-BP02 Mengimplementasikan sasaran dan target
- COST02-BP03 Mengimplementasikan struktur akun
- COST02-BP04 Mengimplementasikan grup dan peran
- COST02-BP05 Mengimplementasikan kontrol biaya
- COST02-BP06 Melacak siklus hidup proyek

COST02-BP01 Mengembangkan kebijakan berdasarkan keperluan organisasi Anda

Kembangkan kebijakan yang menentukan cara organisasi Anda mengelola sumber daya dan periksa sumber daya secara berkala. Kebijakan harus mencakup aspek biaya sumber daya dan beban kerja, termasuk pembuatan, pengubahan, dan penonaktifan selama masa pakai sumber daya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Memahami pendorong dan biaya organisasi Anda sangat penting untuk mengelola biaya dan penggunaan Anda secara efektif, serta untuk mengenali peluang penghematan biaya. Umumnya

organisasi mengoperasikan beberapa beban kerja yang dijalankan oleh beberapa tim. Tim-tim ini dapat berada dalam unit organisasi yang berlainan, masing-masing dengan aliran pendapatannya sendiri. Kemampuan untuk mengaitkan biaya sumber daya dengan beban kerja, tiap-tiap organisasi, atau pemilik produk mendorong perilaku penggunaan yang efisien dan membantu mengurangi pemborosan. Pemantauan biaya dan penggunaan yang akurat membantu Anda memahami seberapa optimal suatu beban kerja, serta seberapa menguntungkan produk dan unit organisasi. Pengetahuan ini memungkinkan pengambilan keputusan yang lebih tepat tentang target pengalokasian sumber daya di dalam organisasi Anda. Kesadaran tentang penggunaan pada semua tingkatan di organisasi sangat penting untuk mendorong perubahan, karena perubahan pada penggunaan akan mendorong perubahan pada biaya. Pertimbangkan untuk mengambil pendekatan dengan beberapa aspek untuk menyadari penggunaan dan pengeluaran Anda.

Langkah pertama dalam menjalankan tata kelola adalah dengan menggunakan keperluan organisasi Anda untuk mengembangkan kebijakan penggunaan cloud Anda. Kebijakan tersebut menentukan cara organisasi Anda menggunakan cloud dan cara mengelola sumber daya. Kebijakan harus mencakup semua aspek dalam sumber daya dan beban kerja yang berkaitan dengan biaya dan penggunaan, termasuk pembuatan, pengubahan, serta penonaktifan selama masa pakai sumber daya. Verifikasikan bahwa kebijakan dan prosedur diikuti dan diterapkan untuk setiap perubahan di lingkungan cloud. Selama rapat manajemen perubahan IT Anda, ajukan pertanyaan untuk mengetahui dampak biaya dari perubahan yang direncanakan apakah meningkat atau menurun, justifikasi bisnis, dan hasil yang diharapkan.

Kebijakan harus dibuat sederhana agar dapat dipahami dengan mudah dan diimplementasikan secara efektif di seluruh organisasi. Kebijakan juga harus mudah diikuti dan ditafsirkan (sehingga digunakan) serta spesifik (tidak ada salah tafsir antar tim). Selain itu, kebijakan perlu diperiksa secara berkala (seperti mekanisme kami) dan diperbarui seiring perubahan kondisi atau prioritas bisnis pelanggan yang dapat menjadikan kebijakan usang.

Mulailah dengan kebijakan tingkat tinggi yang sangat umum, seperti Wilayah geografis yang digunakan atau waktu dalam sehari saat sumber daya harus dijalankan. Sempurnakan kebijakan-kebijakan tersebut secara bertahap untuk unit dan beban kerja organisasi yang beragam. Kebijakan yang umum mencakup layanan dan fitur yang dapat digunakan (misalnya, penyimpanan yang beperforma lebih rendah di lingkungan pengujian dan pengembangan), jenis sumber daya yang dapat digunakan oleh grup yang berbeda-beda (misalnya, ukuran sumber daya terbesar dalam akun pengembangan adalah ukuran sedang), dan jangka waktu sumber daya tersebut akan digunakan (sementara, jangka pendek, atau untuk jangka waktu tertentu).

Contoh kebijakan

Berikut ini adalah contoh kebijakan yang dapat Anda kaji untuk membuat kebijakan tata kelola cloud Anda sendiri, yang berfokus pada pengoptimalan biaya. Pastikan Anda menyesuaikan kebijakan berdasarkan kebutuhan organisasi dan permintaan pemangku kepentingan Anda.

- Nama kebijakan: Tentukan nama kebijakan yang jelas, seperti Kebijakan Optimalisasi Sumber Daya dan Penurunan Biaya.
- Tujuan: Jelaskan alasan kebijakan ini harus digunakan serta hasil yang diharapkan. Tujuan dari kebijakan ini adalah untuk memverifikasi bahwa diperlukan biaya yang minim untuk men-deploy dan menjalankan beban kerja yang diinginkan untuk memenuhi persyaratan bisnis.
- Cakupan: Tentukan dengan jelas siapa yang harus menggunakan kebijakan ini dan kapan kebijakan ini harus digunakan, seperti Tim DevOps X menggunakan kebijakan ini pada pelanggan us-east untuk lingkungan X (produksi atau non-produksi).

Pernyataan kebijakan

1. Pilih us-east-1 atau beberapa wilayah us-east berdasarkan lingkungan beban kerja dan kebutuhan bisnis Anda (pengembangan, pengujian penerimaan pengguna, praproduksi, atau produksi).
2. Jadwal instans Amazon EC2 dan Amazon RDS untuk berjalan antara pukul enam pagi dan delapan malam (Waktu Standar Timur (EST)).
3. Hentikan semua instans Amazon EC2 yang tidak digunakan setelah delapan jam dan instans Amazon RDS yang tidak digunakan setelah 24 jam tidak aktif.
4. Hentikan semua instans Amazon EC2 yang tidak digunakan setelah 24 jam tidak aktif di lingkungan non-produksi. Ingatkan pemilik instans Amazon EC2 (berdasarkan tag) untuk meninjau instans Amazon EC2 mereka yang dihentikan dalam produksi dan beri tahu mereka bahwa instans Amazon EC2 mereka akan diakhiri dalam waktu 72 jam jika tidak digunakan.
5. Gunakan keluarga dan ukuran instans umum seperti m5.large, kemudian ubah ukuran instans berdasarkan penggunaan CPU dan memori menggunakan AWS Compute Optimizer.
6. Prioritaskan menggunakan penskalaan otomatis untuk menyesuaikan jumlah instans yang berjalan secara dinamis berdasarkan lalu lintas.
7. Gunakan instans spot untuk beban kerja non-kritis.
8. Kaji persyaratan kapasitas untuk memberikan komitmen untuk paket penyimpanan atau instans cadangan untuk beban kerja yang dapat diprediksi dan beri tahu Tim Manajemen Keuangan Cloud.
9. Gunakan kebijakan siklus hidup Amazon S3 untuk memindahkan data yang jarang diakses ke tingkat penyimpanan yang lebih murah. Jika tidak ada kebijakan penyimpanan yang ditentukan,

gunakan Amazon S3 Intelligent Tiering untuk memindahkan objek ke tingkat yang diarsipkan secara otomatis.

10. Pantau pemanfaatan sumber daya dan atur alarm untuk memicu peristiwa penskalaan menggunakan Amazon CloudWatch.

11. Untuk tiap Akun AWS, gunakan AWS Budgets untuk menetapkan anggaran biaya dan penggunaan untuk akun Anda berdasarkan unit bisnis dan pusat biaya.

12. Menggunakan AWS Budgets untuk menetapkan anggaran biaya dan penggunaan untuk akun Anda dapat membantu Anda terus memantau pengeluaran dan menghindari tagihan tidak terduga, sehingga Anda dapat lebih mengontrol biaya Anda.

Prosedur: Berikan prosedur mendetail untuk menerapkan kebijakan ini atau beri referensi ke dokumen lain yang menjelaskan cara menerapkan setiap pernyataan kebijakan. Bagian ini harus memberikan petunjuk langkah demi langkah untuk menjalankan persyaratan kebijakan.

Untuk menerapkan kebijakan ini, Anda dapat menggunakan berbagai alat pihak ketiga atau aturan AWS Config untuk memeriksa kepatuhan terhadap pernyataan kebijakan dan memicu tindakan perbaikan otomatis menggunakan fungsi AWS Lambda. Anda juga dapat menggunakan AWS Organizations untuk menegakkan kebijakan. Selain itu, Anda harus secara rutin meninjau penggunaan sumber daya Anda dan menyesuaikan kebijakan apabila diperlukan untuk memastikan kebijakan tersebut terus memenuhi kebutuhan bisnis Anda.

Langkah implementasi

- Bertemu dengan para pemangku kepentingan: Untuk mengembangkan kebijakan, mintalah pemangku kepentingan (kantor bisnis cloud, rekayasawan, atau pengambil keputusan fungsional untuk penegakan kebijakan) di dalam organisasi Anda untuk menentukan kebutuhan mereka dan mendokumentasikannya. Gunakan pendekatan berulang dengan memulai dari hal paling umum dan menyempurnakannya secara berkelanjutan hingga ke unit terkecil di setiap langkahnya. Anggota tim yang terlibat adalah mereka yang berkepentingan langsung dengan beban kerja, seperti unit organisasi atau pemilik aplikasi, serta grup pendukung, seperti tim keamanan dan keuangan.
- Minta konfirmasi: Pastikan semua tim menyetujui kebijakan terkait siapa yang dapat mengakses dan melakukan deployment ke AWS Cloud. Pastikan mereka mengikuti kebijakan organisasi Anda serta konfirmasikan bahwa pembuatan sumber daya mereka sejalan dengan kebijakan dan prosedur yang disetujui.

- Buat sesi pelatihan orientasi: Minta semua anggota organisasi baru untuk menyelesaikan kursus pelatihan orientasi untuk menciptakan kesadaran biaya dan membentuk persyaratan organisasi. Mereka boleh mempertimbangkan berbagai kebijakan dari pengalaman mereka sebelumnya atau tidak mempertimbangkannya sama sekali.
- Tentukan lokasi untuk beban kerja Anda: Tentukan lokasi pengoperasian beban kerja Anda, termasuk negara dan wilayah di dalam negara tersebut. Informasi ini digunakan untuk pemetaan ke Wilayah AWS dan Zona Ketersediaan.
- Tentukan dan kelompokkan layanan serta sumber daya: Tentukan layanan yang diperlukan beban kerja. Untuk setiap layanan, tentukan jenis, ukuran, dan jumlah sumber daya yang diperlukan. Tentukan grup sumber daya berdasarkan fungsi, seperti server aplikasi atau penyimpanan basis data. Sumber daya dapat dimiliki oleh beberapa grup.
- Tentukan dan kelompokkan pengguna berdasarkan fungsi: Tentukan pengguna yang berinteraksi dengan beban kerja, dengan berfokus pada apa yang mereka kerjakan dan cara mereka menggunakan beban kerja, bukan posisi atau jabatan mereka di organisasi. Kelompokkan pengguna atau fungsi yang serupa menjadi satu. Anda dapat menggunakan kebijakan yang dikelola AWS sebagai panduan.
- Tentukan tindakan: Dengan lokasi, sumber daya, dan pengguna yang telah diidentifikasi di awal, tentukan tindakan yang diperlukan oleh masing-masing untuk meraih hasil beban kerja dari masa pakainya (pengembangan, operasi, dan penonaktifan). Identifikasikan tindakannya berdasarkan grup, bukan masing-masing elemen dalam grup, di setiap lokasi. Mulailah dari hal yang umum dengan membaca atau menulis, kemudian sesuaikan tindakan-tindakan tertentu untuk setiap layanan.
- Tentukan periode peninjauan: Beban kerja dan keperluan organisasi dapat berubah seiring waktu. Tentukan jadwal peninjauan beban kerja untuk memastikan ini tetap selaras dengan prioritas organisasi.
- Dokumentasikan kebijakan: Lakukan verifikasi bahwa kebijakan yang telah ditentukan dapat diakses saat dibutuhkan oleh organisasi Anda. Kebijakan-kebijakan ini digunakan untuk mengimplementasikan, memelihara, dan mengaudit akses lingkungan Anda.

Sumber daya

Dokumen terkait:

- Manajemen Perubahan di Cloud
- Kebijakan Terkelola AWS untuk berbagai Fungsi Pekerjaan

- Strategi penagihan beberapa akun AWS
- Tindakan, Sumber Daya, dan Kunci Kondisi untuk Layanan AWS
- Manajemen dan Tata Kelola AWS
- Kontrol akses ke Wilayah AWS menggunakan kebijakan IAM
- Zona Ketersediaan (AZ) dan Wilayah Infrastruktur Global

Video terkait:

- Manajemen dan Tata Kelola AWS dalam Skala Besar

Contoh terkait:

- VMware - Apa yang Dimaksud dengan Kebijakan Cloud?

COST02-BP02 Mengimplementasikan sasaran dan target

Implementasikan sasaran serta target biaya dan penggunaan untuk beban kerja Anda. Sasaran memberikan arah untuk organisasi Anda tentang hasil yang diharapkan, dan target memberikan hasil terukur spesifik yang harus dicapai untuk beban kerja Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Kembangkan sasaran serta target biaya dan penggunaan untuk organisasi Anda. Sebagai organisasi yang sedang berkembang di AWS, penting untuk menetapkan dan melacak sasaran untuk optimalisasi biaya. Sasaran atau indikator kinerja utama (KPI) ini dapat mencakup hal-hal seperti persentase pengeluaran sesuai permintaan, atau adopsi layanan tertentu yang dioptimalkan seperti instans AWS Graviton atau tipe volume EBS gp3. Menetapkan sasaran yang jelas dan dapat dicapai dapat membantu Anda untuk terus mengukur peningkatan efisiensi yang penting bagi operasional bisnis yang sedang berlangsung. Sasaran memberikan panduan dan arah kepada organisasi Anda terkait hasil yang diharapkan. Target memberikan hasil yang jelas dan spesifik yang harus dicapai. Singkatnya, sasaran adalah arah yang ingin Anda tuju sedangkan target adalah seberapa jauh lagi jarak ke arah tersebut dan kapan sasaran tersebut harus dicapai (menggunakan panduan “spesifik, terukur, dapat ditetapkan, realistis, dan tepat waktu”, atau SMART (Specific, Measurable Assignable, Realistic, Timely). Contoh sasaran adalah penggunaan platform harus meningkat secara signifikan, hanya dengan peningkatan biaya yang minim (nonlinier). Contoh target adalah

peningkatan penggunaan platform 20%, dengan peningkatan biaya kurang dari lima persen. Contoh sasaran umum lainnya adalah beban kerja harus lebih efisien setiap enam bulan. Target yang menyertainya adalah metrik biaya per bisnis harus turun lima persen setiap enam bulan.

Sasaran untuk optimalisasi biaya adalah meningkatkan efisiensi beban kerja, yakni mengurangi biaya per hasil bisnis untuk beban kerja tersebut dari waktu ke waktu. Sebaiknya implementasikan sasaran ini ke semua beban kerja, serta tetapkan target seperti peningkatan efisiensi sebesar lima persen setiap enam bulan hingga satu tahun. Hal ini dapat dicapai di cloud melalui pengembangan kemampuan dalam optimalisasi biaya, serta melalui perilisan layanan dan fitur baru.

Penting untuk memiliki visibilitas hampir waktu nyata terhadap KPI dan peluang penghematan terkait serta melacak kemajuan Anda dari waktu ke waktu. Untuk mulai menetapkan dan melacak sasaran KPI, kami merekomendasikan dasbor KPI dari Kerangka kerja Cloud Intelligence Dashboards (CID). Berdasarkan data dari AWS Cost and Usage Report, dasbor KPI menyediakan serangkaian KPI optimalisasi biaya yang direkomendasikan dengan kemampuan untuk menetapkan sasaran khusus dan melacak kemajuan dari waktu ke waktu.

Jika Anda memiliki solusi lain yang memungkinkan Anda menetapkan dan melacak sasaran KPI, pastikan solusi tersebut diadopsi oleh semua pemangku kepentingan manajemen keuangan cloud di organisasi Anda.

Langkah implementasi

- Tetapkan tingkat penggunaan yang diharapkan: Sebagai permulaan, fokuslah pada tingkat penggunaan. Berinteraksilah dengan pemilik aplikasi, pemasaran, dan tim bisnis yang lebih besar untuk memahami tingkat penggunaan yang diharapkan untuk beban kerja. Bagaimana permintaan pelanggan akan berubah seiring waktu, dan apakah akan ada perubahan akibat peningkatan musiman atau kampanye pemasaran?
- Tetapkan pengadaan sumber daya dan biaya beban kerja: Setelah tingkat penggunaan ditetapkan, hitung perubahan pada sumber daya beban kerja yang diperlukan untuk memenuhi tingkat penggunaan ini. Anda mungkin perlu meningkatkan ukuran atau jumlah sumber daya untuk suatu komponen beban kerja, meningkatkan transfer data, atau mengubah komponen beban kerja ke layanan lain pada tingkat tertentu. Tentukan berapa biaya yang akan dikeluarkan pada setiap poin utama ini, dan berapa perubahan biaya ketika terdapat perubahan pada penggunaan.
- Tetapkan sasaran bisnis: Dengan mengambil output dari perkiraan perubahan penggunaan dan biaya, gabungkan dengan perkiraan perubahan dalam teknologi, atau program apa pun yang sedang Anda jalankan, dan kembangkan sasaran untuk beban kerja. Sasaran harus mencakup penggunaan, biaya, serta hubungan di antara keduanya. Sasaran harus sederhana, berada pada

tingkat tinggi, dan membantu orang memahami apa yang diharapkan bisnis dalam hal hasil (seperti memastikan sumber daya yang tidak terpakai dipertahankan di bawah tingkat biaya tertentu). Anda tidak perlu menentukan sasaran untuk setiap jenis sumber daya yang tidak terpakai atau menentukan biaya yang menyebabkan kerugian untuk sasaran dan target. Pastikan terdapat program organisasi (misalnya, pengembangan kemampuan seperti pelatihan dan pendidikan) jika terdapat perubahan yang diperkirakan pada biaya tanpa perubahan pada penggunaan.

- Tetapkan target: Untuk setiap sasaran yang telah ditetapkan, tentukan target yang terukur. Jika sasarannya adalah untuk meningkatkan efisiensi beban kerja, targetnya adalah menghitung jumlah peningkatan (biasanya pada output bisnis untuk setiap dolar yang dikeluarkan) dan kapan peningkatan tersebut akan tercapai. Misalnya, jika Anda menetapkan sasaran untuk meminimalkan pemborosan karena kelebihan penyediaan, target Anda dapat berupa pemborosan akibat kelebihan penyediaan komputasi di tingkat pertama beban kerja produksi tidak boleh melebihi 10% dari biaya komputasi tingkatan dan pemborosan akibat kelebihan penyediaan komputasi di tingkat kedua beban kerja produksi tidak boleh melebihi 5% dari biaya komputasi tingkatan.

Sumber daya

Dokumen terkait:

- Kebijakan terkelola AWS untuk berbagai fungsi pekerjaan
- Strategi multiakun AWS untuk zona landasan AWS Control Tower Anda
- Kontrol akses ke Wilayah AWS menggunakan kebijakan IAM
- Sasaran SMART

Video terkait:

- Lab Well-Architected: Sasaran dan Target (Tingkat 100)

Contoh terkait:

- Lab Well-Architected: Menonaktifkan sumber daya (Sasaran dan Target)
- Lab Well-Architected: Jenis, Ukuran, dan Jumlah Sumber Daya (Sasaran dan Target)

COST02-BP03 Mengimplementasikan struktur akun

Implementasikan struktur akun yang memetakan ke organisasi Anda. Hal ini dapat membantu alokasi dan pengelolaan biaya di organisasi Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

AWS Organizations memungkinkan Anda membuat beberapa Akun AWS yang dapat membantu Anda melakukan tata kelola lingkungan secara terpusat seiring Anda menskalakan beban kerja Anda di AWS. Anda dapat memodelkan hierarki organisasi Anda dengan mengelompokkan Akun AWS dalam struktur unit organisasi (OU) dan membuat beberapa Akun AWS di bawah setiap OU. Untuk membuat struktur akun, Anda perlu memutuskan mana dari Akun AWS Anda yang akan menjadi akun manajemen terlebih dahulu. Setelah itu, Anda dapat membuat Akun AWS baru atau memilih akun-akun yang sudah ada sebagai akun anggota berdasarkan struktur akun rancangan Anda dengan mengikuti praktik terbaik akun manajemen dan praktik terbaik akun anggota.

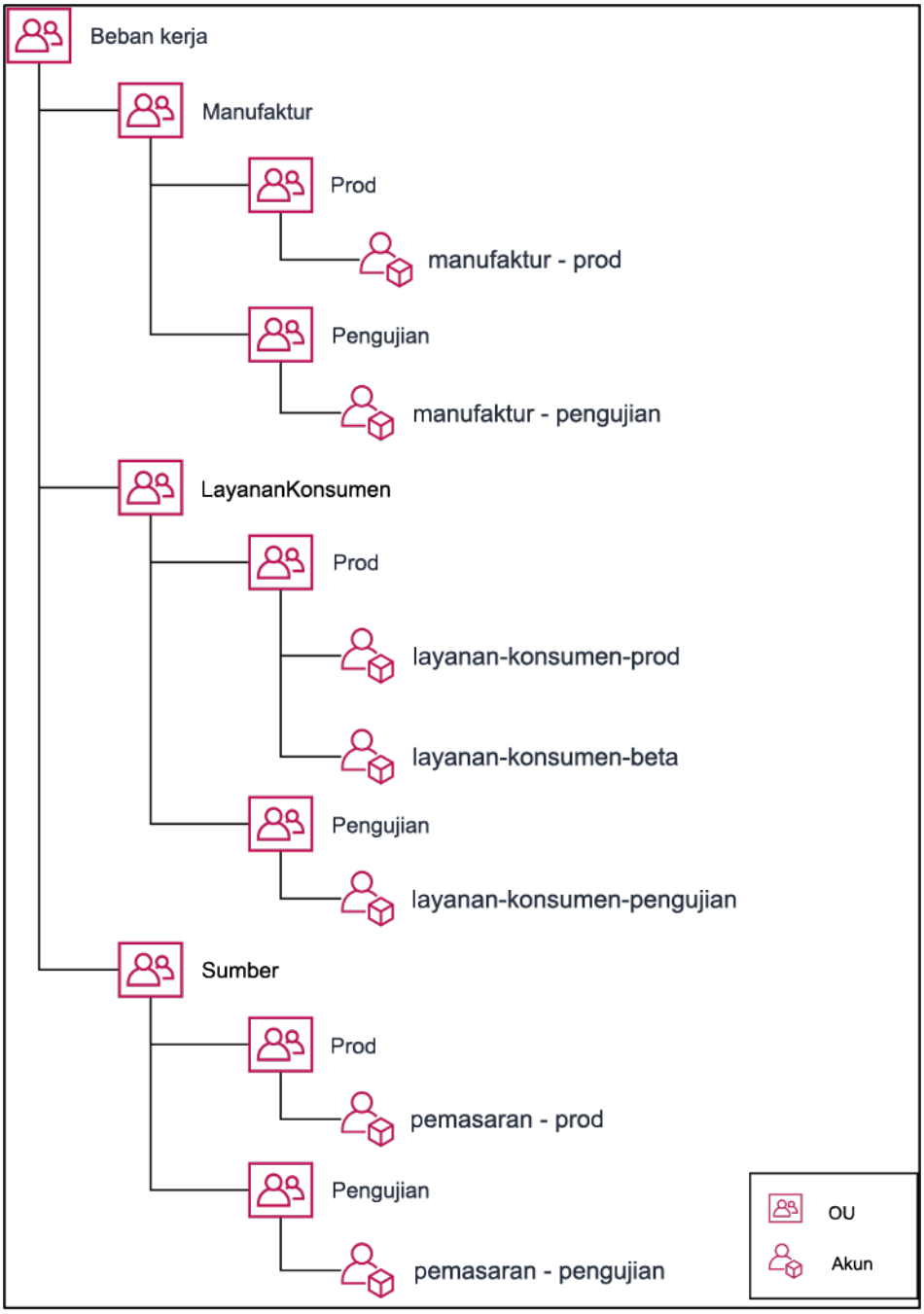
Disarankan untuk memiliki setidaknya satu akun manajemen dengan satu akun anggota yang terhubung ke sana, terlepas dari ukuran organisasi Anda dan penggunaannya. Semua sumber daya beban kerja hanya boleh berada di akun anggota dan tidak boleh ada sumber daya yang dibuat di dalam akun manajemen. Jumlah Akun AWS yang harus Anda miliki akan bergantung pada kebutuhan Anda. Evaluasi model biaya dan operasional Anda saat ini dan pada masa mendatang untuk memastikan bahwa struktur Akun AWS Anda sudah sesuai dengan tujuan organisasi Anda. Beberapa perusahaan membuat banyak Akun AWS untuk alasan bisnis, misalnya:

- Isolasi penagihan dan fiskal atau administratif diperlukan di antara unit organisasi, pusat biaya, atau beban kerja tertentu.
- Layanan AWS memiliki batasan tersendiri untuk beban kerja tertentu.
- Ada persyaratan untuk isolasi dan pemisahan antara beban kerja dan sumber daya.

Di dalam AWS Organizations, tagihan yang terkonsolidasi membuat konstruksi antara satu atau beberapa akun anggota dan akun manajemen. Dengan akun anggota, Anda dapat mengisolasi serta membedakan biaya dan penggunaan berdasarkan grup. Praktik terbaik yang umum dilakukan adalah membuat akun anggota terpisah untuk setiap unit organisasi (seperti keuangan, pemasaran, dan penjualan), atau untuk setiap siklus hidup lingkungan (seperti pengembangan, pengujian, dan produksi), atau untuk setiap beban kerja (beban kerja a, b, dan c), kemudian menggabungkan akun yang terhubung ini menggunakan penagihan terkonsolidasi.

Penagihan terkonsolidasi membantu menggabungkan pembayaran untuk beberapa Akun AWS anggota dalam satu akun manajemen, dan tetap memberikan visibilitas untuk setiap aktivitas akun yang dihubungkan. Karena biaya dan penggunaan digabungkan dalam akun manajemen, Anda dapat memaksimalkan diskon volume layanan Anda, dan memaksimalkan penggunaan diskon komitmen Anda (Savings Plans dan Instans Terpesan) untuk mencapai diskon tertinggi.

Diagram berikut ini menunjukkan bagaimana Anda dapat menggunakan AWS Organizations dengan unit organisasi (OU) untuk mengelompokkan beberapa akun, dan menempatkan beberapa Akun AWS di setiap OU. Disarankan untuk menggunakan OU untuk berbagai kasus penggunaan dan beban kerja yang menyediakan pola untuk mengatur akun.



Contoh pengelompokan beberapa Akun AWS di bawah unit-unit organisasi.

AWS Control Tower dapat mengatur dan mengonfigurasi beberapa akun AWS dengan cepat, memastikan tata kelolanya sesuai dengan persyaratan organisasi Anda.

Langkah implementasi

- Tentukan persyaratan pemisahan: Persyaratan pemisahan adalah kombinasi dari beberapa faktor, termasuk keamanan, keandalan, dan konstruksi keuangan. Siapkan setiap faktor secara berurutan

dan tentukan apakah beban kerja atau lingkungan beban kerja harus dipisahkan dari beban kerja lainnya. Keamanan meningkatkan perekatan pada persyaratan akses dan data. Keandalan mengelola batasan sehingga lingkungan dan beban kerja tidak memengaruhi hal lain. Pelajari pilar keamanan dan keandalan Kerangka Kerja Well-Architected secara berkala dan ikuti praktik-praktik terbaik yang disediakan. Konstruksi keuangan menciptakan pemisahan keuangan yang ketat (pusat biaya, kepemilikan beban kerja, dan akuntabilitas yang berbeda). Contoh umum pemisahan adalah menjalankan beban kerja pengujian dan produksi di akun terpisah, atau menggunakan akun terpisah agar data penagihan dan faktur dapat diberikan ke unit bisnis atau departemen individu di dalam organisasi atau pemangku kepentingan yang memiliki akun.

- Tentukan persyaratan pengelompokan: Persyaratan pengelompokan tidak menggantikan persyaratan pemisahan, tetapi digunakan untuk membantu manajemen. Kelompokkan lingkungan atau beban kerja serupa yang tidak perlu dipisah. Misalnya, kelompokkan beberapa lingkungan pengujian atau pengembangan dari satu atau beberapa beban kerja menjadi satu.
- Tentukan struktur akun: Dengan menggunakan pemisahan dan pengelompokan ini, tentukan akun untuk setiap kelompok dan penuhi persyaratan pemisahan. Akun-akun ini adalah akun anggota atau akun yang Anda hubungkan. Dengan mengelompokkan akun anggota ini ke dalam satu akun manajemen atau pembayar, Anda menggabungkan penggunaan untuk memperbesar volume diskon di semua akun, yang menyediakan satu tagihan untuk semua akun. Anda dapat memisahkan data penagihan dan menampilkannya secara terpisah untuk setiap akun anggota. Jika akun anggota tidak ingin memperlihatkan data penagihan atau penggunaannya kepada akun lain, atau harus ada tagihan terpisah dari AWS, tentukan beberapa akun manajemen atau pembayar. Dalam hal ini, setiap akun anggota memiliki akun manajemen atau pembayarnya masing-masing. Sumber daya harus selalu disimpan di akun anggota atau terkait. Akun manajemen atau pembayar hanya digunakan untuk manajemen.

Sumber daya

Dokumen terkait:

- Menggunakan Tag Alokasi Biaya
- Kebijakan terkelola AWS untuk fungsi tugas
- Strategi penagihan beberapa akun AWS
- Kontrol akses ke Wilayah AWS menggunakan kebijakan IAM
- AWS Control Tower
- AWS Organizations

- Praktik terbaik untuk akun manajemen dan akun anggota
- Mengatur lingkungan AWS Anda Menggunakan Beberapa Akun
- Mengaktifkan instans terpesan bersama dan diskon Savings Plans
- Penagihan terkonsolidasi
- Penagihan terkonsolidasi

Contoh terkait:

- Memisahkan CUR dan Berbagi Akses

Video terkait:

- Memperkenalkan AWS Organizations
- Menyiapkan Lingkungan AWS Multiakun yang Menggunakan Praktik Terbaik untuk AWS Organizations

Contoh terkait:

- Lab Well-Architected: Membuat Organisasi AWS (Level 100)
- Memisahkan AWS Cost and Usage Report dan Berbagi Akses
- Menentukan Strategi Multiakun AWS untuk perusahaan telekomunikasi
- Praktik Terbaik untuk Mengoptimalkan Akun AWS
- Praktik Terbaik untuk Unit Organisasi dengan AWS Organizations

COST02-BP04 Mengimplementasikan grup dan peran

Implementasikan grup dan peran yang selaras dengan kebijakan Anda serta kontrol siapa saja yang dapat membuat, mengubah, atau menonaktifkan instans dan sumber daya di setiap grup. Misalnya, implementasikan grup pengembangan, pengujian, dan produksi. Ini berlaku untuk solusi pihak ketiga dan layanan AWS.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Peran dan grup pengguna merupakan blok pembangun dasar dalam desain dan implementasi sistem yang aman dan efisien. Peran dan grup membantu organisasi menyeimbangkan perlunya kontrol dengan kebutuhan akan fleksibilitas dan produktivitas, yang pada akhirnya mendukung tujuan organisasi dan kebutuhan pengguna. Seperti yang direkomendasikan di bagian Manajemen identitas dan akses dalam Pilar Keamanan Kerangka Kerja AWS Well-Architected, Anda memerlukan manajemen identitas dan izin yang kuat untuk menyediakan akses ke sumber daya yang tepat untuk orang yang tepat dalam kondisi yang tepat. Pengguna hanya menerima akses yang diperlukan untuk menyelesaikan tugasnya. Ini meminimalkan risiko yang terkait dengan akses yang tidak sah atau penyalahgunaan.

Setelah mengembangkan kebijakan, Anda dapat membuat peran dan grup logis pengguna di organisasi Anda. Ini memungkinkan Anda untuk menetapkan izin, mengontrol penggunaan, dan membantu menerapkan mekanisme kontrol akses yang kuat, yang mencegah akses tidak sah ke informasi sensitif. Awali dengan pengelompokan orang tingkat tinggi. Hal ini biasanya selaras dengan unit organisasi dan peran pekerjaan (misalnya, administrator sistem di Departemen IT, pengontrol keuangan, atau analis bisnis). Grup tersebut mengategorikan orang yang memiliki tugas serupa dan memerlukan akses serupa. Peran menentukan aktivitas grup. Mengelola izin untuk grup dan peran lebih mudah dibandingkan untuk pengguna individu. Peran dan grup menetapkan izin secara konsisten dan sistematis bagi semua pengguna, sehingga mencegah kesalahan dan inkonsistensi.

Ketika peran pengguna berubah, administrator dapat menyesuaikan akses di tingkat peran atau grup, bukan mengonfigurasi ulang akun setiap pengguna. Misalnya, administrator sistem di TI memerlukan akses untuk membuat semua sumber daya, sedangkan tim analitik hanya perlu membuat sumber daya analitik.

Langkah implementasi

- Implementasikan grup: Dengan grup pengguna yang ditentukan dalam kebijakan organisasi Anda, implementasikan grup yang sesuai, jika perlu. Untuk praktik terbaik tentang pengguna, grup, dan autentikasi, lihat Pilar Keamanan Kerangka Kerja AWS Well-Architected.
- Implementasikan peran dan kebijakan: Dengan menggunakan tindakan yang ditentukan dalam kebijakan organisasi Anda, buat kebijakan akses dan peran yang diperlukan. Untuk praktik terbaik tentang peran dan kebijakan, lihat Pilar Keamanan Kerangka Kerja AWS Well-Architected.

Sumber daya

Dokumen terkait:

- Kebijakan terkelola AWS untuk fungsi tugas
- Strategi penagihan beberapa akun AWS
- Pilar Keamanan Kerangka Kerja AWS Well-Architected
- AWS Identity and Access Management (IAM)
- Kebijakan AWS Identity and Access Management

Video terkait:

- Why use Identity and Access Management

Contoh terkait:

- Akses dan Identitas Dasar Lab Well-Architected
- Mengontrol akses ke Wilayah AWS menggunakan kebijakan IAM
- Starting your Cloud Financial Management journey: Cloud cost operations

COST02-BP05 Mengimplementasikan kontrol biaya

Implementasikan kontrol berdasarkan kebijakan organisasi serta grup dan peran yang ditetapkan. Ini semua memastikan bahwa biaya hanya dikenakan sesuai yang ditetapkan oleh persyaratan organisasi seperti kontrol akses ke wilayah atau tipe sumber daya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Langkah pertama yang umum dalam mengimplementasikan kontrol biaya adalah mengatur notifikasi ketika peristiwa biaya atau penggunaan terjadi di luar kebijakan. Anda dapat bertindak cepat dan memverifikasi apakah diperlukan tindakan perbaikan, tanpa membatasi atau mengganggu beban kerja atau aktivitas baru. Setelah mengetahui batas beban kerja dan lingkungan, Anda dapat menegakkan tata kelola. AWS Budgets memungkinkan Anda untuk mengatur notifikasi dan menetapkan anggaran bulanan untuk biaya, penggunaan, diskon komitmen (Savings Plans dan Instans Terpesan) AWS Anda. Anda dapat membuat anggaran pada tingkat biaya agregat (misalnya semua biaya), atau pada tingkat yang lebih mendetail yakni hanya menyertakan dimensi tertentu seperti akun yang dihubungkan, layanan, tag, atau Zona Ketersediaan.

Setelah Anda mengatur batas anggaran dengan AWS Budgets, gunakan AWS Cost Anomaly Detection untuk mengurangi biaya tidak terduga. AWS Cost Anomaly Detection adalah layanan manajemen biaya yang menggunakan machine learning untuk memantau biaya dan penggunaan Anda secara kontinu guna mendeteksi pengeluaran yang tidak wajar. Hal ini membantu Anda mengidentifikasi pengeluaran yang tidak wajar dan akar masalah, sehingga Anda dapat mengambil tindakan secara cepat. Pertama-tama, buat pemantau biaya di AWS Cost Anomaly Detection, lalu pilih preferensi pemberitahuan Anda dengan mengatur ambang batas biaya (seperti pemberitahuan anomali dengan dampak lebih dari $1.000). Setelah Anda menerima pemberitahuan, Anda dapat menganalisis akar masalah penyebab anomali dan dampaknya pada biaya Anda. Anda juga dapat memantau dan melakukan analisis anomali Anda sendiri di AWS Cost Explorer.

Tegakkan kebijakan tata kelola di AWS melalui AWS Identity and Access Management dan AWS Organizations Service Control Policies (SCP). IAM memungkinkan Anda untuk mengelola akses secara aman ke layanan dan sumber daya AWS. Menggunakan IAM, Anda dapat mengontrol siapa yang dapat membuat atau mengelola sumber daya AWS, tipe sumber daya yang dapat dibuat, dan di mana sumber daya tersebut dapat dibuat. Hal ini meminimalkan kemungkinan sumber daya dibuat di luar kebijakan yang ditetapkan. Gunakan peran dan kelompok yang dibuat sebelumnya dan tetapkan kebijakan IAM untuk menegakkan penggunaan yang tepat. SCP menawarkan kontrol terpusat pada izin maksimum yang tersedia untuk semua akun di organisasi Anda, yang menjaga akun-akun Anda tetap berada di dalam pedoman kontrol akses Anda. SCP hanya tersedia di organisasi yang mengaktifkan semua fitur, dan Anda dapat mengonfigurasi SCP agar menolak atau mengizinkan tindakan untuk akun anggota secara default. Untuk detail selengkapnya tentang implementasi manajemen akses, lihat laporan resmi Pilar Keamanan Well-Architected.

Tata kelola juga dapat diimplementasikan melalui manajemen Kuota layanan AWS. Dengan memastikan kuota layanan diatur dengan biaya overhead minimum dan dipelihara secara akurat, Anda dapat meminimalkan pembuatan sumber daya di luar kebutuhan organisasi. Untuk meraih hal ini, Anda harus memahami seberapa cepat kebutuhan Anda dapat berubah, memahami proyek yang sedang berlangsung (baik pembuatan maupun penonaktifan sumber daya), dan mempertimbangkan seberapa cepat perubahan kuota dapat diimplementasikan. Kuota layanan dapat digunakan untuk meningkatkan kuota Anda saat diperlukan.

Langkah implementasi

- Implementasikan notifikasi pengeluaran: Menggunakan kebijakan organisasi yang Anda tetapkan, buat AWS Budgets untuk memberi tahu Anda saat pengeluaran berada di luar kebijakan Anda. Konfigurasikan beberapa anggaran biaya, satu untuk masing-masing akun, yang memberi tahu Anda tentang keseluruhan pengeluaran akun. Konfigurasikan anggaran biaya tambahan di

dalam masing-masing akun untuk unit yang lebih kecil di dalam akun. Unit-unit tersebut berbeda-beda tergantung struktur akun Anda. Beberapa contohnya adalah Wilayah AWS, beban kerja (menggunakan tag), atau layanan AWS. Konfigurasikan daftar distribusi email sebagai penerima notifikasi, bukan akun email individu. Anda dapat mengonfigurasi anggaran riil ketika jumlah terlampaui, atau gunakan prakiraan anggaran untuk memberitahukan prakiraan penggunaan. Anda juga dapat melakukan konfigurasi Tindakan Anggaran AWS yang dapat menegakkan kebijakan IAM atau SCP, atau menghentikan instans Amazon EC2 atau Amazon RDS target. Tindakan Anggaran dapat dijalankan secara otomatis atau memerlukan persetujuan alur kerja.

- Implementasikan notifikasi pengeluaran anomali: Gunakan AWS Cost Anomaly Detection untuk mengurangi biaya tidak terduga di dalam organisasi Anda dan analisis akar masalah potensi pengeluaran anomali. Setelah Anda membuat pemantau biaya untuk mengidentifikasi pengeluaran tidak wajar dengan tingkat detail yang Anda tentukan dan mengonfigurasi notifikasi di AWS Cost Anomaly Detection, Anda akan menerima pemberitahuan ketika pengeluaran tidak wajar terdeteksi. Hal ini memungkinkan Anda untuk menganalisis akar masalah penyebab anomali tersebut dan memahami dampaknya terhadap biaya Anda. Gunakan Kategori Biaya AWS sambil melakukan konfigurasi AWS Cost Anomaly Detection untuk mengidentifikasi tim proyek atau tim unit bisnis mana yang dapat menganalisis akar masalah biaya tidak terduga dan mengambil tindakan yang diperlukan secara tepat waktu.
- Implementasikan kontrol pada penggunaan: Menggunakan kebijakan organisasi yang Anda tetapkan, implementasikan kebijakan dan peran IAM untuk menentukan tindakan apa yang dapat dilakukan oleh pengguna dan tindakan yang tidak dapat mereka lakukan. Beberapa kebijakan organisasi dapat disertakan di satu kebijakan AWS. Seperti saat Anda menetapkan kebijakan, mulailah secara umum lalu terapkan kontrol secara lebih mengerucut di masing-masing langkah. Batas layanan juga merupakan kontrol penggunaan yang efektif. Implementasikan batas layanan yang tepat pada semua akun Anda.

Sumber daya

Dokumen terkait:

- Kebijakan terkelola AWS untuk fungsi tugas
- Strategi penagihan beberapa akun AWS
- Kontrol akses ke Wilayah AWS menggunakan kebijakan IAM
- AWS Budgets
- AWS Cost Anomaly Detection
- Kontrol Biaya AWS Anda

Video terkait:

- Cara Menggunakan AWS Budgets untuk melacak pengeluaran dan penggunaan

Contoh terkait:

- Contoh kebijakan manajemen akses IAM
- Contoh kebijakan kontrol layanan
- Tindakan Anggaran AWS
- Buat Kebijakan IAM untuk mengontrol akses ke sumber daya Amazon EC2 menggunakan Tag
- Batasi akses Identitas IAM ke sumber daya Amazon EC2 tertentu
- Buat Kebijakan IAM untuk membatasi penggunaan Amazon EC2 berdasarkan kelompok
- Lab Well-Architected: Tata Kelola Biaya dan Penggunaan (Level 100)
- Lab Well-Architected: Tata Kelola Biaya dan Penggunaan (Level 200)
- Integrasi Slack untuk Cost Anomaly Detection menggunakan Amazon Q Developer in chat applications

COST02-BP06 Melacak siklus hidup proyek

Lacak, ukur, dan audit siklus hidup proyek, tim, dan lingkungan untuk menghindari penggunaan dan pembayaran sumber daya yang tidak perlu.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Pelacakan siklus hidup proyek secara efektif memungkinkan organisasi untuk memiliki kontrol yang lebih baik atas biaya melalui perencanaan, manajemen, dan pengoptimalan sumber daya, waktu, dan kualitas yang lebih baik. Wawasan yang diperoleh melalui pelacakan sangat berharga untuk membuat keputusan yang lebih tepat yang berperan dalam efektivitas biaya dan keberhasilan proyek secara keseluruhan.

Melacak seluruh siklus kerja beban kerja membantu Anda memahami kapan beban kerja atau komponen beban kerja tidak lagi diperlukan. Beban kerja dan komponen yang ada mungkin tampak sedang digunakan, tetapi ketika AWS merilis layanan atau fitur baru, beban kerja dan komponen tersebut dapat dinonaktifkan atau diadopsi. Periksa tahapan beban kerja sebelumnya. Setelah beban

kerja diproduksi, lingkungan sebelumnya bisa dinonaktifkan atau jauh dikurangi kapasitasnya sampai diperlukan lagi.

AWS menyediakan beberapa layanan manajemen dan tata kelola yang bisa Anda gunakan untuk melacak siklus hidup entitas. Anda dapat menggunakan AWS Config atau AWS Systems Manager untuk menyediakan inventaris sumber daya dan konfigurasi AWS Anda yang mendetail. Anda disarankan untuk mengintegrasikan proyek atau sistem manajemen aset yang sudah ada agar proyek dan produk aktif di dalam organisasi Anda tetap terlacak. Dengan menggabungkan sistem Anda saat ini dengan set peristiwa dan metrik yang kaya yang disediakan oleh AWS, Anda dapat membangun tampilan peristiwa siklus hidup yang signifikan serta secara proaktif mengelola sumber daya untuk mengurangi biaya yang tidak perlu.

Serupa dengan Manajemen Siklus Hidup Aplikasi (ALM), pelacakan siklus hidup proyek harus melibatkan beberapa proses, alat, dan tim yang bekerja bersama, seperti desain dan pengembangan, pengujian, produksi, dukungan, dan redundansi beban kerja.

Dengan memantau setiap fase siklus hidup proyek secara cermat, organisasi memperoleh wawasan penting dan kontrol yang lebih baik, sehingga dapat memfasilitasi perencanaan, implementasi, dan penyelesaian proyek dengan baik. Pengawasan yang cermat ini memverifikasi bahwa proyek tidak hanya memenuhi standar kualitas, tetapi juga disampaikan tepat waktu dan tidak melampaui anggaran, sehingga meningkatkan efisiensi biaya secara keseluruhan.

Untuk detail selengkapnya tentang penerapan pelacakan siklus hidup entitas, lihat laporan resmi Pilar Keunggulan Operasional AWS Well-Architected.

Langkah implementasi

- Tetapkan proses pemantauan siklus hidup proyek: Tim Cloud Center of Excellence harus menetapkan proses pemantauan siklus hidup proyek. Tetapkan pendekatan terstruktur dan sistematis untuk memantau beban kerja agar dapat meningkatkan kontrol, visibilitas, dan performa proyek. Jadikan proses pemantauan transparan, kolaboratif, dan berfokus pada peningkatan berkelanjutan untuk memaksimalkan efektivitas dan nilainya.
- Lakukan tinjauan beban kerja: Sebagaimana ditentukan oleh kebijakan organisasi Anda, tetapkan jadwal reguler untuk mengaudit proyek-proyek yang ada dan lakukan tinjauan beban kerja. Besarnya upaya yang dilakukan untuk audit harus sebanding dengan perkiraan risiko, nilai, atau biaya bagi organisasi. Area utama yang disertakan dalam audit adalah risiko insiden atau pemadaman terhadap organisasi, nilai atau kontribusi terhadap organisasi (diukur dalam bentuk pendapatan atau reputasi merek), biaya beban kerja (diukur dalam bentuk total biaya sumber daya dan biaya operasional), dan penggunaan beban kerja (diukur dalam bentuk jumlah hasil organisasi

per unit waktu). Jika area-area ini berubah selama siklus hidup, diperlukan penyesuaian beban kerja, seperti penonaktifan penuh atau sebagian.

Sumber daya

Dokumen terkait:

- Guidance for Tagging on AWS
- What Is ALM (Application Lifecycle Management)?
- Kebijakan terkelola AWS untuk fungsi tugas

Contoh terkait:

- Kontrol akses ke Wilayah AWS menggunakan kebijakan IAM

Alat terkait:

- AWS Config
- AWS Systems Manager
- AWS Budgets
- AWS Organizations
- AWS CloudFormation

COST 3. Bagaimana cara memantau biaya dan penggunaan Anda?

Tetapkan kebijakan dan prosedur untuk memantau dan mengalokasikan biaya Anda dengan tepat. Hal ini memungkinkan Anda mengukur dan meningkatkan efisiensi biaya beban kerja ini.

Praktik terbaik

- COST03-BP01 Mengonfigurasi sumber informasi yang mendetail
- COST03-BP02 Menambahkan informasi organisasi ke biaya dan penggunaan
- COST03-BP03 Mengidentifikasi kategori atribusi biaya
- COST03-BP04 Membangun metrik organisasi
- COST03-BP05 Mengonfigurasikan alat manajemen penagihan dan biaya
- COST03-BP06 Mengalokasikan biaya berdasarkan metrik beban kerja

COST03-BP01 Mengonfigurasi sumber informasi yang mendetail

Konfigurasikan alat manajemen biaya dan pelaporan untuk tingkat detail per jam guna memberikan informasi biaya dan penggunaan yang mendetail, sehingga memungkinkan analitik dan transparansi yang lebih mendalam. Konfigurasikan beban kerja Anda guna menghasilkan atau memperoleh entri log untuk setiap hasil bisnis yang dicapai.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Dengan informasi penagihan yang mendetail seperti tingkat detail per jam di dalam alat manajemen biaya, organisasi dapat melacak pemakaian mereka dengan lebih mendetail dan mengidentifikasi beberapa penyebab kenaikan biaya. Sumber-sumber data ini menyediakan tampilan paling akurat untuk biaya dan penggunaan di seluruh organisasi Anda.

AWS Cost and Usage Report menyediakan tingkat detail harian atau per jam, tarif, biaya, dan atribut penggunaan untuk semua layanan AWS yang dikenai biaya. Semua dimensi yang memungkinkan ada di dalam CUR, termasuk: pemberian tag, lokasi, atribut sumber daya, dan ID akun.

Konfigurasi CUR Anda dengan penyesuaian berikut ini:

- Sertakan ID sumber daya
- Segarkan CUR secara otomatis
- Tingkat detail per jam
- Versioning: Timpa laporan yang ada
- Integrasi data: Athena (Format dan kompresi Parquet)

Gunakan AWS Glue untuk menyiapkan data untuk analisis, dan gunakan Amazon Athena untuk melakukan analisis data, menggunakan SQL untuk mengkueri data. Anda juga dapat menggunakan Quick untuk membangun visualisasi kustom dan kompleks serta mendistribusikannya ke seluruh organisasi Anda.

Langkah implementasi

- Konfigurasi laporan biaya dan penggunaan: Menggunakan konsol penagihan, konfigurasi setidaknya satu laporan biaya dan penggunaan. Konfigurasi laporan dengan tingkat detail per jam yang menyertakan semua pengidentifikasi dan ID sumber daya. Anda juga dapat membuat laporan

lain dengan tingkat detail berbeda untuk menyediakan informasi rangkuman dengan tingkat lebih tinggi.

- Konfigurasi tingkat detail per jam di Cost Explorer: Aktifkan Per jam dan Data Tingkat Sumber Daya untuk mengakses data biaya dan penggunaan dengan perincian per jam selama 14 hari terakhir dan perincian tingkat sumber daya.
- Konfigurasi pembuatan log aplikasi: Verifikasi bahwa aplikasi Anda membuat log untuk hasil bisnis yang dicapai sehingga hasil tersebut dapat dilacak dan diukur. Pastikan tingkat detail data ini setidaknya per jam sehingga sesuai dengan data biaya dan penggunaan. Untuk detail selengkapnya tentang pencatatan log dan pemantauan, lihat Pilar Keunggulan Operasional Well-Architected.

Sumber daya

Dokumen terkait:

- AWS Cost and Usage Report
- AWS Glue
- Quick
- Harga Manajemen Biaya AWS
- Memberi tag pada sumber daya AWS
- Menganalisis biaya dengan AWS Budgets
- Menganalisis biaya dengan Cost Explorer
- Mengelola AWS Cost and Usage Report
- Pilar Keunggulan Operasional Well-Architected

Contoh terkait:

- Penyiapan Akun AWS
- Tampilan Baru dan Kasus Penggunaan Umum AWS Cost Explorer

COST03-BP02 Menambahkan informasi organisasi ke biaya dan penggunaan

Tentukan skema pemberian tag berdasarkan organisasi, atribut beban kerja, dan kategori alokasi biaya agar Anda dapat memfilter dan mencari sumber daya atau memantau biaya dan penggunaan

di alat manajemen biaya. Implementasikan pemberian tag yang konsisten untuk semua sumber daya jika memungkinkan berdasarkan tujuan, tim, lingkungan, atau kriteria lain yang relevan dengan bisnis Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Implementasikan pemberian tag di AWS untuk menambahkan informasi organisasi ke sumber daya Anda, yang kemudian akan ditambahkan ke informasi biaya dan penggunaan. Tag adalah pasangan kunci-nilai—kunci perlu ditentukan dan harus unik di seluruh organisasi, dan nilainya unik untuk grup sumber daya. Contoh pasangan kunci-nilai adalah `Environment` sebagai kunci dan `Production` sebagai nilai. Semua sumber daya dalam lingkungan produksi pasti memiliki pasangan kunci-nilai ini. Dengan penandaan, Anda dapat melacak dan mengelola biaya dengan informasi organisasi yang relevan dan bermanfaat. Anda dapat menerapkan tag yang merepresentasikan kategori organisasi (seperti pusat biaya, nama aplikasi, proyek, atau pemilik), dan mengidentifikasi beban kerja serta karakteristik beban kerja (misalnya pengujian atau produksi) untuk mengaitkan biaya dan penggunaan di seluruh organisasi.

Saat menerapkan tag ke sumber daya AWS (misalnya instans Amazon Elastic Compute Cloud atau bucket Amazon Simple Storage Service) dan mengaktifkan tag, AWS menambahkan informasi ini ke Laporan Biaya dan Penggunaan. Anda dapat menjalankan laporan dan melakukan analisis pada sumber daya yang diberi tag dan tidak diberi tag untuk meningkatkan kepatuhan terhadap kebijakan manajemen biaya dan memastikan atribusi yang akurat.

Membuat dan mengimplementasikan standar penandaan AWS di seluruh akun organisasi memungkinkan Anda untuk mengelola lingkungan AWS yang seragam dan konsisten. Gunakan Kebijakan Tag di AWS Organizations untuk menentukan aturan terkait cara tag dapat digunakan pada sumber daya AWS di akun AWS Organizations Anda. Kebijakan Tag memungkinkan Anda mengadopsi pendekatan terstandardisasi untuk pemberian tag sumber daya AWS

AWS Tag Editor memungkinkan Anda menambahkan, menghapus, dan mengelola tag di berbagai sumber daya. Degan Tag Editor, Anda dapat mencari sumber daya yang ingin Anda beri tag, lalu mengelola tag untuk sumber daya tersebut dalam hasil pencarian Anda.

AWS Cost Categories memungkinkan Anda menetapkan makna organisasi ke biaya tanpa memerlukan tag pada sumber daya. Anda dapat memetakan informasi biaya dan penggunaan ke struktur organisasi internal yang unik. Anda menentukan aturan kategori untuk memetakan dan mengategorikan biaya menggunakan dimensi penagihan, seperti akun dan tag. Selain

penandaan, hal ini memberikan kemampuan manajemen pada tingkat yang berbeda. Anda juga dapat memetakan akun dan tag spesifik untuk beberapa proyek.

Langkah implementasi

- Tentukan skema pemberian tag: Kumpulkan semua pemangku kepentingan dari seluruh bisnis Anda untuk menentukan skema. Hal ini umumnya melibatkan orang-orang yang memiliki peran di bidang teknis, keuangan, dan manajemen. Kumpulkan daftar tag yang wajib dimiliki oleh semua sumber daya serta tag yang sebaiknya dimiliki oleh sumber daya. Verifikasikan bahwa nama dan nilai tag konsisten di seluruh organisasi.
- Berikan tag pada sumber daya: Dengan kategori atribusi biaya yang telah ditentukan, tempatkan tag pada semua sumber daya di beban kerja berdasarkan kategori. Gunakan alat seperti CLI, Tag Editor, atau AWS Systems Manager untuk meningkatkan efisiensi.
- Implementasikan AWS Cost Categories: Anda dapat membuat Kategori Biaya tanpa mengimplementasikan pemberian tag. Kategori biaya menggunakan dimensi biaya dan penggunaan yang sudah ada. Buat aturan kategori dari skema dan implementasikan ke kategori biaya.
- Pemberian tag otomatis: Untuk memastikan Anda mempertahankan pemberian tag tingkat tinggi pada semua sumber daya, otomatiskan pemberian tag agar sumber daya diberi tag secara otomatis saat dibuat. Gunakan layanan seperti AWS CloudFormation untuk memastikan sumber daya diberi tag saat dibuat. Anda juga dapat membuat solusi kustom untuk memberikan tag secara otomatis menggunakan fungsi Lambda atau menggunakan layanan mikro yang memindai beban kerja secara berkala dan menghapus semua sumber daya yang tidak diberi tag, yang ideal untuk lingkungan pengujian dan pengembangan.
- Pantau dan jalankan laporan pemberian tag: Untuk memastikan Anda mempertahankan pemberian tag tingkat tinggi di seluruh organisasi, jalankan laporan dan pantau tag pada seluruh beban kerja Anda. Anda dapat menggunakan AWS Cost Explorer untuk melihat biaya sumber daya yang diberi tag dan tidak diberi tag atau menggunakan layanan seperti Tag Editor. Tinjau secara berkala jumlah sumber daya yang tidak diberi tag dan ambil tindakan untuk menambahkan tag hingga tingkat pemberian tag yang diinginkan tercapai.

Sumber daya

Dokumen terkait:

- Praktik Terbaik Pemberian Tag
- Tag Sumber Daya AWS CloudFormation

- AWS Cost Categories
- Memberikan tag pada sumber daya AWS
- Menganalisis biaya dengan AWS Budgets
- Menganalisis biaya dengan Cost Explorer
- Mengelola Laporan Biaya dan Penggunaan AWS

Video terkait:

- Bagaimana cara memberikan tag pada sumber daya AWS saya untuk membagi tagihan berdasarkan pusat biaya atau proyek
- Memberikan Tag pada Sumber Daya AWS

Contoh terkait:

- Berikan tag pada sumber daya AWS baru secara otomatis berdasarkan identitas atau peran

COST03-BP03 Mengidentifikasi kategori atribusi biaya

Identifikasi kategori organisasi seperti unit bisnis, departemen, atau proyek yang dapat digunakan untuk mengalokasikan biaya di dalam organisasi Anda ke entitas pengonsumsi internal. Gunakan kategori tersebut untuk menegakkan akuntabilitas pengeluaran, menciptakan kesadaran biaya, dan mendorong perilaku pemakaian yang efektif.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Proses kategorisasi biaya sangat penting dalam penganggaran, akuntansi, pelaporan keuangan, pengambilan keputusan, benchmarking, dan manajemen proyek. Dengan mengklasifikasikan dan membuat kategori pengeluaran, tim dapat lebih memahami jenis biaya yang mereka keluarkan selama perjalanan cloud mereka, sehingga membantu tim dalam mengambil keputusan yang tepat dan mengelola anggaran secara efektif.

Akuntabilitas pengeluaran cloud sangat bermanfaat untuk menghadirkan manajemen permintaan dan biaya yang disiplin. Hasilnya adalah penghematan biaya cloud yang jauh lebih besar untuk organisasi yang mengalokasikan sebagian besar pengeluaran cloud mereka untuk unit bisnis atau tim yang

memakainya. Selain itu, pengalokasian pengeluaran cloud membantu organisasi dalam mengadopsi lebih banyak praktik terbaik tata kelola cloud yang terpusat.

Bekerjasamalah dengan tim keuangan Anda atau pemangku kepentingan lain yang relevan untuk memahami persyaratan tentang bagaimana biaya harus dialokasikan di dalam organisasi Anda selama rapat koordinasi rutin Anda. Biaya beban kerja harus dialokasikan pada seluruh siklus hidup, termasuk pengembangan, pengujian, produksi, dan penonaktifan. Pahami bagaimana biaya dikenakan untuk pembelajaran, pengembangan staf, dan pencetusan ide yang berkaitan dengan organisasi. Dengan begitu, akun yang akan digunakan untuk tujuan ini dapat dialokasikan dengan tepat ke anggaran pelatihan dan pengembangan, bukan anggaran biaya IT umum.

Setelah menentukan kategori atribusi biaya dengan pemangku kepentingan di organisasi, gunakan Kategori Biaya AWS untuk mengelompokkan informasi biaya dan penggunaan Anda ke dalam kategori yang bermakna di AWS Cloud seperti biaya untuk proyek tertentu atau Akun AWS untuk departemen atau unit bisnis. Anda dapat membuat kategori kustom dan memetakan informasi biaya dan penggunaan ke dalam kategori tersebut berdasarkan aturan yang Anda tentukan menggunakan berbagai dimensi seperti akun, tag, layanan, atau jenis biaya. Setelah kategori biaya disiapkan, Anda dapat melihat informasi biaya dan penggunaan berdasarkan kategori tersebut sehingga memungkinkan organisasi Anda membuat keputusan strategis dan pembelian yang lebih baik. Kategori tersebut juga akan terlihat di AWS Cost Explorer, AWS Budgets, dan AWS Cost and Usage Report.

Misalnya, buat kategori biaya untuk unit bisnis Anda (Tim DevOps), dan dalam setiap kategori, buat beberapa aturan (aturan untuk setiap subkategori) dengan beberapa dimensi (Akun AWS, tag alokasi biaya, layanan, atau jenis biaya) berdasarkan pengelompokan yang Anda tentukan. Dengan kategori biaya, Anda dapat mengatur biaya menggunakan mesin berbasis aturan. Aturan yang Anda konfigurasikan akan mengatur biaya ke dalam kategori. Dalam aturan ini, Anda dapat memfilter menggunakan beberapa dimensi untuk setiap kategori seperti Akun AWS tertentu, layanan AWS, atau jenis biaya. Anda kemudian dapat menggunakan kategori tersebut untuk berbagai produk di konsol AWS Manajemen Penagihan dan Biaya dan Manajemen Biaya .. Produk ini mencakup AWS Cost Explorer, AWS Budgets, AWS Cost and Usage Report, dan AWS Cost Anomaly Detection.

Misalnya, diagram berikut menunjukkan cara mengelompokkan biaya dan informasi penggunaan di organisasi Anda dengan memiliki beberapa tim (kategori biaya), beberapa lingkungan (aturan), dan setiap lingkungan yang memiliki beberapa sumber daya atau aset (dimensi).

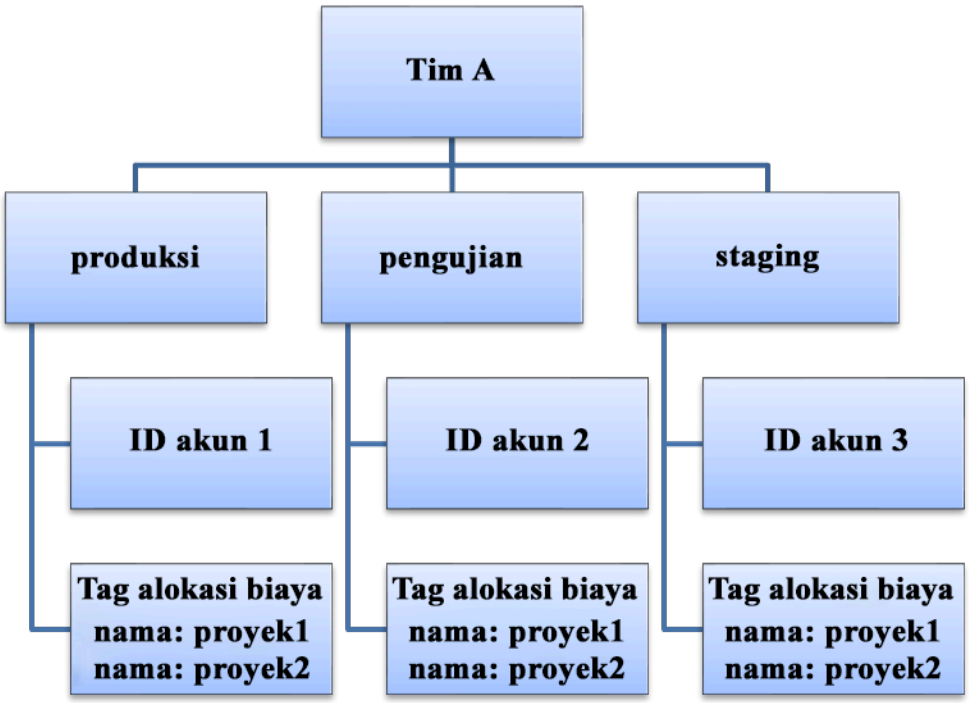


Diagram organisasi biaya dan penggunaan

Anda juga dapat membuat pengelompokan biaya menggunakan kategori biaya. Setelah Anda membuat kategori biaya (perlu waktu 24 jam setelah membuat kategori biaya agar catatan penggunaan Anda dapat diperbarui dengan nilai), kategori biaya tersebut akan muncul di AWS Cost Explorer, AWS Budgets, AWS Cost and Usage Report, dan AWS Cost Anomaly Detection. Di AWS Cost Explorer dan AWS Budgets, kategori biaya muncul dalam bentuk dimensi penagihan tambahan. Anda dapat menggunakannya untuk memfilter nilai kategori biaya tertentu atau mengelompokkan berdasarkan kategori biaya.

Langkah implementasi

- Tentukan kategori organisasi Anda: Lakukan rapat dengan para pemangku kepentingan dan unit bisnis internal untuk menentukan kategori yang sesuai dengan struktur dan persyaratan organisasi Anda. Kategori ini akan secara langsung dipetakan ke struktur kategori keuangan yang ada, seperti unit bisnis, anggaran, pusat biaya, dan departemen. Lihat hasil yang diberikan cloud untuk bisnis Anda, seperti pelatihan dan edukasi, karena ini juga merupakan kategori organisasi.
- Tentukan kategori fungsional Anda: Lakukan rapat dengan para pemangku kepentingan dan unit bisnis internal untuk menentukan kategori yang sesuai dengan fungsi yang Anda miliki dalam bisnis Anda. Hal ini dapat berupa nama aplikasi atau beban kerja, serta jenis lingkungan, seperti produksi, pengujian, atau pengembangan.

- Menentukan Kategori Biaya AWS: Buat kategori biaya untuk mengatur informasi biaya dan penggunaan dengan menggunakan Kategori Biaya AWS dan memetakan biaya dan penggunaan AWS Anda ke kategori yang bermakna. Beberapa kategori dapat ditetapkan pada satu sumber daya, dan satu sumber daya bisa masuk dalam kategori-kategori yang berlainan, jadi tentukan kategorinya sebanyak yang Anda butuhkan sehingga Anda dapat mengelola biaya Anda dalam struktur yang dikategorikan menggunakan Kategori Biaya AWS.

Sumber daya

Dokumen terkait:

- Memberi tag pada sumber daya AWS
- Menggunakan Tag Alokasi Biaya
- Menganalisis biaya dengan AWS Budgets
- Menganalisis biaya dengan Cost Explorer
- Mengelola AWS Cost and Usage Report
- Kategori Biaya AWS
- Mengelola biaya Anda dengan Kategori Biaya AWS
- Membuat kategori biaya
- Memberikan tag pada kategori biaya
- Membagi biaya dalam kategori biaya
- Fitur Kategori Biaya AWS

Contoh terkait:

- Mengatur biaya dan penggunaan Anda dengan Kategori Biaya AWS
- Mengelola biaya Anda dengan Kategori Biaya AWS
- Lab Well-Architected: Visualisasi Biaya dan Penggunaan
- Lab Well-Architected: Kategori Biaya

COST03-BP04 Membangun metrik organisasi

Bangun metrik-metrik organisasi yang diperlukan untuk beban kerja ini. Contoh metrik beban kerja adalah laporan pelanggan yang dibuat, atau halaman web yang disajikan untuk pelanggan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Pahami bagaimana output beban kerja Anda diukur berdasarkan keberhasilan bisnis. Masing-masing beban kerja umumnya memiliki satu set kecil berisi output-output utama yang mengindikasikan kinerja. Jika Anda memiliki beban kerja yang kompleks dengan banyak komponen, Anda dapat memprioritaskan daftar, atau menetapkan dan melacak metrik untuk setiap komponen. Bekerjalah dengan tim Anda untuk memahami metrik mana yang akan digunakan. Unit ini akan digunakan untuk memahami efisiensi beban kerja, atau biaya untuk masing-masing output bisnis.

Langkah implementasi

- Tentukan hasil beban kerja: Lakukan pertemuan dengan pemangku kepentingan dalam bisnis dan tetapkan hasil untuk beban kerja. Ini adalah pengukur utama penggunaan pelanggan dan harus berupa metrik bisnis dan bukan metrik teknis. Harus ada sedikit metrik tingkat tinggi (kurang dari lima) per beban kerja. Jika beban kerja memunculkan beberapa hasil untuk kasus-kasus penggunaan yang berbeda, kelompokkan ke dalam satu metrik.
- Tentukan hasil komponen beban kerja: Opsi lainnya adalah jika Anda memiliki beban kerja besar dan kompleks, atau dapat dengan mudah mengurai beban kerja Anda ke dalam komponen (seperti layanan mikro) dengan input dan output yang ditetapkan dengan baik, tentukan metrik untuk masing-masing komponen. Upaya harus mencerminkan nilai dan biaya komponen. Mulailah dengan komponen yang paling besar menuju komponen yang lebih kecil.

Sumber daya

Dokumen terkait:

- Memberikan tag pada sumber daya AWS
- Menganalisis biaya dengan AWS Budgets
- Menganalisis biaya dengan Cost Explorer
- Mengelola Laporan Biaya dan Penggunaan AWS

COST03-BP05 Mengonfigurasikan alat manajemen penagihan dan biaya

Konfigurasikan alat manajemen biaya sesuai dengan kebijakan organisasi Anda untuk mengelola dan mengoptimalkan pengeluaran cloud. Hal ini mencakup layanan, alat, dan sumber daya untuk

mengatur dan melacak data biaya dan penggunaan, mengoptimalkan kontrol melalui penagihan terkonsolidasi dan izin akses, meningkatkan perencanaan melalui penganggaran dan prakiraan, menerima notifikasi atau peringatan, serta menurunkan biaya secara lebih lanjut dengan sumber daya dan optimisasi harga.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Untuk membangun akuntabilitas yang kuat, strategi akun Anda harus dipertimbangkan terlebih dahulu sebagai bagian dari strategi alokasi biaya Anda. Jika benar di tahap ini, Anda mungkin tidak perlu berupaya lebih jauh. Jika tidak, dapat muncul ketidaksadaran dan titik-titik masalah lebih lanjut.

Untuk mendorong akuntabilitas pengeluaran cloud, pengguna harus memiliki akses ke alat yang memberikan visibilitas tentang biaya dan penggunaan mereka. Semua beban kerja dan tim sebaiknya memiliki beberapa alat yang telah dikonfigurasi untuk detail dan tujuan berikut:

- Atur: Menetapkan acuan dasar alokasi dan tata kelola biaya dengan strategi pemberian tag dan kategorisasi Anda sendiri.
- Atur: Tetapkan acuan dasar alokasi dan tata kelola biaya dengan strategi pemberian tag dan taksonomi Anda sendiri. Berikan tag pada sumber daya AWS dan kategorikan sumber daya tersebut secara bermakna berdasarkan struktur organisasi Anda (unit bisnis, departemen, atau proyek). Beri tag pada nama akun untuk pusat-pusat biaya tertentu dan petakan dengan Kategori Biaya AWS untuk mengelompokkan akun unit bisnis tertentu untuk pusat biaya mereka sehingga pemilik unit bisnis dapat melihat konsumsi beberapa akun di satu tempat.
- Akses: Melacak informasi penagihan di seluruh organisasi dalam penagihan terkonsolidasi dan verifikasi bahwa pemangku kepentingan yang tepat dan pemilik bisnis memiliki akses.
- Kontrol: Bangun mekanisme tata kelola yang efektif dengan pagar pembatas yang tepat untuk mencegah skenario tak terduga saat menggunakan Kebijakan Kontrol Layanan (SCP), kebijakan tag, dan pemberitahuan anggaran. Misalnya, Anda dapat mengizinkan tim untuk membuat sumber daya di Wilayah yang dipilih hanya dengan menggunakan mekanisme kontrol yang efektif.
- Status Saat Ini: Mengonfigurasi dasbor yang menampilkan tingkat biaya dan penggunaan saat ini. Dasbor harus tersedia di tempat yang mudah terlihat di dalam lingkungan kerja, mirip dengan dasbor operasi. Anda dapat menggunakan Cloud Intelligence Dashboard (CID) atau produk lain yang didukung untuk menciptakan visibilitas ini.
- Notifikasi: Berikan notifikasi saat biaya atau penggunaan melebihi batas yang ditentukan dan ketika anomali terjadi dengan AWS Budgets atau AWS Cost Anomaly Detection.

- Laporan: Merangkum semua informasi biaya dan penggunaan serta tingkatkan kesadaran dan akuntabilitas pengeluaran cloud Anda dengan data biaya yang mendetail dan dapat diatribusikan. Laporan harus relevan dengan tim yang memakainya dan idealnya harus berisi rekomendasi.
- Pelacakan: Menampilkan biaya dan penggunaan saat ini dibandingkan dengan tujuan atau target yang telah ditentukan.
- Analisis: Membantu anggota tim melakukan analisis kustom dan mendalam dengan detail per jam, menggunakan semua dimensi yang memungkinkan.
- Inspeksi: Terus memantau deployment sumber daya dan peluang optimisasi biaya Anda. Dapatkan notifikasi (menggunakan Amazon CloudWatch, Amazon SNS, atau Amazon SES) untuk deployment sumber daya di tingkat organisasi dan tinjau rekomendasi optimasi biaya (misalnya, AWS Compute Optimizer atau AWS Trusted Advisor).
- Tren: Menampilkan variabilitas pada biaya dan penggunaan selama periode waktu yang diperlukan, dengan detail yang diperlukan.
- Prakiraan: Menampilkan perkiraan biaya mendatang, memperkirakan penggunaan sumber daya Anda, dan melakukan pengeluaran dengan dasbor prakiraan yang Anda buat.

Anda dapat menggunakan alat AWS seperti AWS Cost Explorer, AWS Manajemen Penagihan dan Biaya, atau AWS Budgets untuk hal-hal inti, atau Anda dapat mengintegrasikan data CUR dengan Amazon Athena dan Quick untuk memberikan kemampuan ini untuk tampilan yang lebih mendetail. Jika Anda tidak memiliki keterampilan inti atau kapasitas di dalam organisasi Anda, Anda dapat bekerja sama dengan AWS ProServ, AWS Managed Services (AMS), atau AWS Partner dan menggunakan alat-alat mereka. Anda juga dapat menggunakan alat pihak ketiga, tetapi pastikan terlebih dahulu bahwa biayanya akan memberikan nilai yang sepadan bagi organisasi Anda.

Langkah implementasi

- Izinkan akses berbasis tim ke alat: Konfigurasikan akun Anda dan buat grup yang memiliki akses ke laporan biaya dan penggunaan yang diperlukan untuk konsumsi mereka dan gunakan AWS Identity and Access Management untuk mengontrol akses ke alat-alat seperti AWS Cost Explorer. Grup tersebut harus menyertakan perwakilan dari semua tim yang memiliki atau mengelola sebuah aplikasi. Hal ini bertujuan untuk memastikan setiap tim dapat mengakses informasi biaya dan penggunaan untuk melacak konsumsi mereka.
- Mengonfigurasi AWS Budgets: Konfigurasikan AWS Budgets di semua akun untuk beban kerja Anda. Tetapkan anggaran untuk pengeluaran akun secara keseluruhan serta anggaran untuk beban kerja menggunakan tag. Konfigurasikan notifikasi di AWS Budgets untuk mendapatkan

peringatan saat Anda melebihi jumlah yang dianggarkan, atau saat perkiraan biaya Anda melebihi anggaran.

- Mengonfigurasi AWS Cost Explorer: Konfigurasikan AWS Cost Explorer untuk beban kerja dan akun Anda guna memvisualisasikan data biaya untuk analisis lebih lanjut. Buat dasbor untuk beban kerja yang melacak pengeluaran secara keseluruhan, metrik penggunaan utama untuk beban kerja, dan perkiraan biaya mendatang berdasarkan data biaya historis Anda.
- Mengonfigurasi AWS Cost Anomaly Detection: Gunakan AWS Cost Anomaly Detection untuk akun, layanan inti, atau kategori biaya yang Anda buat guna memantau biaya dan penggunaan serta mendeteksi pengeluaran yang tidak biasa. Anda bisa mendapatkan peringatan satu per satu dalam laporan gabungan dan mendapatkan peringatan dalam email atau topik Amazon SNS yang memungkinkan Anda menganalisis dan menentukan akar penyebab anomali, serta mengidentifikasi faktor yang mendorong kenaikan biaya.
- Mengonfigurasikan alat lanjutan: Anda memiliki opsi untuk membuat alat kustom untuk organisasi Anda guna memberikan detail dan granularitas tambahan. Anda dapat mengimplementasikan kemampuan analisis lanjutan menggunakan Amazon Athena, dan dasbor menggunakan Quick. Pertimbangkan untuk menggunakan solusi CID yang memiliki dasbor lanjutan yang telah dikonfigurasi. Tersedia juga AWS Partner yang dapat Anda ajak bekerja sama dan adopsi solusi manajemen cloud-nya untuk mengaktifkan pemantauan dan optimisasi tagihan cloud di satu lokasi yang praktis.

Sumber daya

Dokumen terkait:

- Manajemen Biaya AWS
- Memberi tag Sumber daya AWS
- Menganalisis biaya dengan AWS Budgets
- Menganalisis biaya dengan Cost Explorer
- Mengelola AWS Cost and Usage Report
- Kategori Biaya AWS
- Manajemen Keuangan Cloud dengan AWS
- Contoh kebijakan kontrol layanan
- Partner APN AWS – Manajemen Biaya

Video terkait:

- Melakukan Deployment Dasbor Inteligensi Cloud
- Dapatkan Peringatan tentang Metrik atau KPI FinOps atau Optimisasi Biaya

Contoh terkait:

- Lab Well-Architected - Pengaturan Akun AWS
- Lab Well-Architected: Visualisasi Penagihan
- Lab Well-Architected: Biaya dan Penggunaan Tata Kelola
- Lab Well-Architected: Analisis Biaya dan Penggunaan
- Lab Well-Architected: Visualisasi Biaya dan Penggunaan
- Lab Well-Architected: Dasbor Inteligensi Cloud
- Cara menggunakan SCP untuk mengatur pagar pembatas izin di seluruh akun

COST03-BP06 Mengalokasikan biaya berdasarkan metrik beban kerja

Alokasikan biaya beban kerja berdasarkan metrik penggunaan atau hasil bisnis untuk mengukur efisiensi biaya beban kerja. Implementasikan proses untuk menganalisis data biaya dan penggunaan, yang dapat menyediakan wawasan dan kemampuan charge back.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Optimalisasi biaya menghadirkan hasil bisnis pada titik harga terendah, yang hanya dapat dicapai dengan mengalokasikan biaya beban kerja berdasarkan metrik beban kerja (diukur berdasarkan efisiensi beban kerja). Pantau metrik beban kerja yang ditetapkan melalui file log atau pemantauan aplikasi lain. Kombinasikan data ini dengan biaya beban kerja, yang dapat diperoleh dengan melihat biaya dengan nilai tag khusus atau ID akun. Disarankan melakukan analisis ini pada level per jam. Efisiensi umumnya akan berubah jika Anda memiliki beberapa komponen biaya statis (misalnya basis data backend yang berjalan secara permanen) dengan beragam laju permintaan (misalnya puncak penggunaan pada pukul sembilan pagi hingga lima sore, dengan sedikit permintaan pada malam hari). Memahami hubungan antara biaya statis dan variabel akan membantu Anda berfokus pada aktivitas optimalisasi Anda.

Membuat metrik beban kerja untuk sumber daya bersama mungkin lebih sulit dibandingkan dengan sumber daya seperti aplikasi dalam kontainer pada Amazon Elastic Container Service (Amazon ECS) dan Amazon API Gateway. Namun, ada beberapa cara tertentu bagi Anda untuk mengkategorikan penggunaan dan melacak biaya. Jika Anda perlu melacak sumber daya bersama Amazon ECS dan AWS Batch, Anda dapat mengaktifkan data alokasi biaya terpisah di AWS Cost Explorer. Dengan data alokasi biaya terpisah, Anda dapat memahami dan mengoptimalkan biaya serta penggunaan aplikasi dalam kontainer dan mengalokasikan biaya aplikasi kembali ke entitas bisnis masing-masing berdasarkan cara pemakaian sumber daya komputasi dan memori bersama mereka. Jika Anda memiliki penggunaan fungsi API Gateway dan AWS Lambda bersama, Anda dapat menggunakan AWS Application Cost Profiler untuk mengkategorikan konsumsi mereka berdasarkan `ID Penyewa` atau `ID Pelanggan`.

Langkah implementasi

- Alokasikan biaya ke metrik beban kerja: Menggunakan metrik yang ditetapkan dan tag yang dikonfigurasi, buat metrik yang mengombinasikan output beban kerja dan biaya beban kerja. Gunakan layanan analitik seperti Amazon Athena dan Amazon Quick untuk membuat dasbor efisiensi untuk beban kerja keseluruhan, dan komponen apa pun.

Sumber daya

Dokumen terkait:

- Memberi tag pada sumber daya AWS
- Menganalisis biaya dengan AWS Budgets
- Menganalisis biaya dengan Cost Explorer
- Mengelola Laporan Biaya dan Penggunaan AWS

Contoh terkait:

- Meningkatkan visibilitas biaya Amazon ECS dan AWS Batch dengan Data Alokasi Biaya Terpisah AWS

COST 4. Bagaimana cara melakukan penonaktifan sumber daya?

Implementasikan kontrol perubahan dan manajemen sumber daya dari awal proyek hingga akhir masa pakai. Hal ini memastikan Anda akan mematikan atau mengakhiri sumber daya yang tidak digunakan agar tidak boros.

Praktik terbaik

- COST04-BP01 Lacak sumber daya sepanjang masa pakainya
- COST04-BP02 Mengimplementasikan proses penonaktifan
- COST04-BP03 Menonaktifkan sumber daya
- COST04-BP04 Menonaktfikan sumber daya secara otomatis
- COST04-BP05 Menegakkan kebijakan retensi data

COST04-BP01 Lacak sumber daya sepanjang masa pakainya

Tentukan dan implementasikan metode untuk melacak sumber daya dan kaitannya dengan sistem sepanjang masa pakainya. Anda dapat menggunakan pemberian tag untuk mengidentifikasi beban kerja atau fungsi sumber daya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Nonaktifkan sumber daya beban kerja yang sudah tidak diperlukan. Contoh yang umum adalah sumber daya yang digunakan untuk pengujian: setelah pengujian selesai, sumber daya dapat dikeluarkan. Melacak sumber daya dengan tag (dan menjalankan laporan atas tag-tag tersebut) dapat membantu Anda mengidentifikasi aset untuk dinonaktifkan karena tidak digunakan atau yang lisensinya akan kedaluwarsa. Menggunakan tag merupakan cara efektif untuk melacak sumber daya, dengan memberi label sumber daya dengan fungsinya, atau tanggal kapan sumber daya dapat dinonaktifkan. Maka pelaporan dapat dijalankan atas tag ini. Contoh nilai untuk pemberian tag pada fitur adalah `pengujian fitur X` untuk mengidentifikasi tujuan sumber daya dalam siklus pakai beban kerja. Contoh lainnya adalah menggunakan `LifeSpan` atau TTL untuk sumber daya, seperti nama dan nilai kunci tag yang akan dihapus untuk menetapkan periode atau waktu tertentu untuk penonaktifan.

Langkah implementasi

- Implementasikan skema pemberian tag: Implementasikan skema pemberian tag yang mengidentifikasi beban kerja untuk sumber daya, sambil verifikasi bahwa semua sumber daya dalam beban kerja sudah diberi tag dengan benar. Pemberian tag membantu Anda membuat kategori sumber daya berdasarkan tujuan, tim, lingkungan, atau kriteria lain yang relevan dengan bisnis Anda. Untuk detail selengkapnya tentang kasus penggunaan pemberian tag, strategi, dan teknik, lihat Praktik Terbaik Pemberian Tag AWS.
- Implementasikan pemantauan throughput dan output beban kerja: Implementasikan pemantauan atau peringatan throughput beban kerja, yang dipicu oleh permintaan input atau penyelesaian output. Konfigurasikan untuk memberikan notifikasi ketika permintaan beban kerja atau output menurun hingga nol, yang menandakan sumber daya beban kerja sudah tidak digunakan. Sertakan faktor waktu jika beban kerja secara berkala menurun hingga nol dalam kondisi normal. Untuk detail selengkapnya tentang sumber daya yang tidak digunakan atau kurang dimanfaatkan, lihat Pemeriksaan Optimasi Biaya AWS Trusted Advisor.
- Kelompokkan sumber daya AWS: Buat grup untuk sumber daya AWS. Anda dapat menggunakan AWS Resource Groups untuk mengatur dan mengelola sumber daya AWS Anda yang berada di Wilayah AWS yang sama. Anda dapat menambahkan tag ke sebagian besar sumber daya Anda untuk membantu mengidentifikasi dan menyortir sumber daya Anda di dalam organisasi Anda. Gunakan Tag Editor untuk menambahkan tag ke sumber daya yang didukung secara massal. Pertimbangkan menggunakan AWS Service Catalog untuk membuat, mengelola, dan mendistribusikan portofolio produk-produk yang disetujui kepada pengguna akhir dan mengelola siklus hidup produk.

Sumber daya

Dokumen terkait:

- AWS Auto Scaling
- AWS Trusted Advisor
- Pemeriksaan Optimasi Biaya AWS Trusted Advisor
- Memberikan tag pada sumber daya AWS
- Memublikasikan Metrik Kustom

Video terkait:

- Cara mengoptimalkan biaya menggunakan AWS Trusted Advisor

Contoh terkait:

- Mengatur sumber daya AWS
- Mengoptimalkan biaya menggunakan AWS Trusted Advisor

COST04-BP02 Mengimplementasikan proses penonaktifan

Implementasikan proses untuk mengidentifikasi dan menonaktifkan sumber daya tidak terpakai

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Implementasikan proses standar di organisasi Anda untuk mengidentifikasi dan menyingkirkan sumber daya yang tidak digunakan. Proses tersebut harus menetapkan frekuensi pelaksanaan pencarian, dan proses untuk menyingkirkan sumber daya guna memverifikasi terpenuhinya semua persyaratan organisasi.

Langkah implementasi

- Buat dan implementasikan proses penonaktifan: Bekerjalah dengan developer dan pemilik beban kerja untuk membangun proses penonaktifan untuk beban kerja dan sumber dayanya. Proses tersebut harus mencakup metode untuk memverifikasi apakah beban kerja sedang digunakan, begitu juga dengan setiap sumber daya beban kerja. Buat detail langkah-langkah yang diperlukan untuk menonaktifkan sumber daya, yakni menghapusnya dari layanan sambil memastikan kepatuhan terhadap semua persyaratan peraturan. Semua sumber daya terkait harus disertakan, seperti lisensi atau penyimpanan terlampir. Beri tahu pemilik beban kerja bahwa proses penonaktifan telah dijalankan.

  Gunakan langkah-langkah penonaktifan berikut ini untuk memandu Anda terkait hal-hal yang harus diperiksa sebagai bagian dari proses Anda:

  - Identifikasi sumber daya yang akan dinonaktifkan: Identifikasi sumber daya yang layak dinonaktifkan di dalam AWS Cloud Anda. Rekam semua informasi yang diperlukan dan jadwalkan penonaktifan. Dalam lini waktu Anda, pastikan untuk mempertimbangkan apakah (dan kapan) masalah yang tidak terduga muncul selama proses.
  - Lakukan koordinasi dan komunikasi: Bekerjalah dengan pemilik beban kerja untuk mengonfirmasikan sumber daya yang akan dinonaktifkan.
  - Rekam metadata dan buat cadangan: Rekam metadata (seperti IP publik, Wilayah, AZ, VPC, Subnet, dan Grup Keamanan) dan buat cadangan (seperti snapshot Amazon Elastic Block Store

atau mengambil AMI, laporan kunci, dan ekspor Sertifikat) jika diperlukan untuk sumber daya di dalam lingkungan produksi atau jika sumber daya tersebut sangat penting.

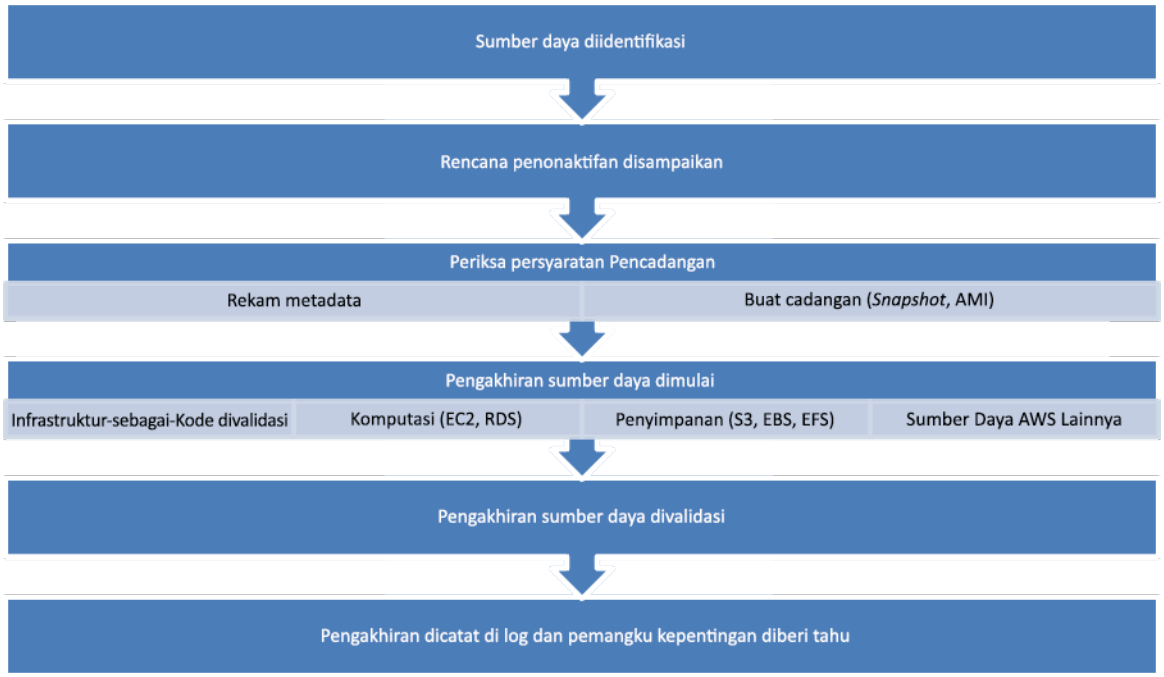
- Validasikan infrastruktur sebagai kode: Tentukan apakah sumber daya di-deploy dengan CloudFormation, Terraform, AWS Cloud Development Kit (AWS CDK), atau alat deployment infrastruktur sebagai kode lainnya sehingga dapat di-deploy ulang jika perlu.
- Cegah akses: Terapkan kontrol restriktif selama jangka waktu tertentu, untuk mencegah penggunaan sumber daya ketika Anda menentukan apakah sumber daya tersebut diperlukan. Verifikasi bahwa lingkungan sumber daya dapat dikembalikan ke status aslinya jika diperlukan.
- Ikuti proses penonaktifan internal Anda: Ikuti tugas-tugas administratif dan proses penonaktifan organisasi Anda, seperti menghilangkan sumber daya dari domain organisasi Anda, menghilangkan catatan DNS, dan menghilangkan sumber daya dari alat manajemen konfigurasi Anda, alat pemantauan, alat otomatisasi, dan alat keamanan.

Jika sumber daya adalah instans Amazon EC2, pelajari daftar berikut ini. Untuk detail selengkapnya, lihat Bagaimana cara menghapus atau mengakhiri sumber daya Amazon EC2 saya?

- Hentikan atau akhiri semua instans dan penyeimbang beban Amazon EC2 Anda. Instans Amazon EC2 dapat dilihat di konsol dalam waktu singkat setelah diakhiri. Anda tidak menerima tagihan untuk instans apa pun yang tidak memiliki status berjalan
- Hapus infrastruktur Auto Scaling Anda.
- Lepaskan semua Host Khusus.
- Hapus semua volume Amazon EBS dan snapshot Amazon EBS.
- Lepaskan semua Alamat IP elastis.
- Batalkan pendaftaran semua Amazon Machine Image (AMI).
- Akhiri lingkungan AWS Elastic Beanstalk Anda.

Jika sumber daya adalah objek di dalam penyimpanan Amazon Glacier dan jika Anda menghapus sebuah arsip sebelum memenuhi durasi penyimpanan minimum, Anda akan dikenakan biaya penghapusan dini prorata. Durasi penyimpanan minimum Amazon Glacier bergantung pada kelas penyimpanan yang digunakan. Untuk ringkasan durasi penyimpanan minimum untuk setiap kelas penyimpanan, lihat Kinerja di kelas-kelas penyimpanan Amazon S3. Untuk detail tentang penghitungan biaya penghapusan dini, lihat harga Amazon S3.

Bagan alur proses penonaktifan sederhana berikut ini menguraikan langkah-langkah penonaktifan. Sebelum menonaktifkan sumber daya, verifikasikan bahwa sumber daya yang telah Anda identifikasi untuk dinonaktifkan tidak sedang digunakan oleh organisasi.



Alur penonaktifan sumber daya.

Sumber daya

Dokumen terkait:

- AWS Auto Scaling
- AWS Trusted Advisor
- AWS CloudTrail

Video terkait:

- Hapus tumpukan CloudFormation tetapi pertahankan beberapa sumber daya
- Temukan pengguna mana yang meluncurkan instans Amazon EC2

Contoh terkait:

- Hapus atau akhiri sumber daya Amazon EC2
- Temukan pengguna mana yang meluncurkan instans Amazon EC2

COST04-BP03 Menonaktifkan sumber daya

Nonaktifkan sumber daya yang dipicu oleh peristiwa seperti audit berkala, atau perubahan penggunaan. Penonaktifan umumnya dilakukan secara berkala dan dapat dilakukan secara manual atau otomatis.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Frekuensi dan upaya untuk mencari sumber daya yang tidak digunakan harus mencerminkan potensi penghematan, sehingga akun dengan biaya kecil harus dianalisis lebih jarang daripada akun dengan biaya yang lebih besar. Pencarian dan peristiwa penonaktifan bisa dipicu oleh perubahan status pada beban kerja, seperti produk yang mendekati akhir masa pakai atau mengalami penggantian. Pencarian dan peristiwa penonaktifan mungkin juga dipicu oleh peristiwa eksternal, seperti perubahan kondisi pasar atau penghentian produk.

Langkah implementasi

- Nonaktifkan sumber daya: Ini adalah tahap depresiasi sumber daya AWS yang sudah tidak diperlukan atau mengakhiri perjanjian lisensi. Selesaikan semua pemeriksaan akhir sebelum beralih ke tahap penghapusan dan menonaktifkan sumber daya untuk mencegah gangguan yang tidak diinginkan seperti mengambil snapshot atau cadangan. Nonaktifkan setiap sumber daya yang telah teridentifikasi tidak terpakai menggunakan proses penonaktifan.

Sumber daya

Dokumen terkait:

- AWS Auto Scaling
- AWS Trusted Advisor

Contoh terkait:

- Lab Well-Architected: Menonaktifkan sumber daya (Level 100)

COST04-BP04 Menonaktfikan sumber daya secara otomatis

Rancang beban kerja Anda agar menangani pengakhiran sumber daya secara anggun ketika Anda mengidentifikasi dan menonaktifkan sumber daya non-kritis, sumber daya yang tidak diperlukan, atau sumber daya dengan pemanfaatan yang rendah.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Gunakan otomatisasi untuk mengurangi atau menyingkirkan biaya terkait untuk proses penonaktifan. Dengan merancang beban kerja agar menjalankan penonaktifan otomatis, Anda akan mengurangi biaya beban kerja secara keseluruhan selama masa pakainya. Anda dapat menggunakan AWS Auto Scaling untuk melakukan proses penonaktifan. Anda juga dapat mengimplementasikan kode kustom menggunakan API atau SDK untuk menonaktfikan sumber daya beban kerja secara otomatis.

Aplikasi modern dibangun dengan prioritas nirserver, yakni strategi yang mengutamakan adopsi layanan nirserver. AWS mengembangkan layanan nirserver untuk ketiga lapisan tumpukan Anda: komputasi, integrasi, dan tempat penyimpanan data. Menggunakan arsitektur nirserver, Anda dapat menghemat biaya selama periode lalu lintas rendah dengan menaikkan dan menurunkan skala secara otomatis.

Langkah implementasi

- Implementasikan AWS Auto Scaling: Untuk sumber daya yang didukung, konfigurasikan dengan AWS Auto Scaling. AWS Auto Scaling dapat membantu Anda mengoptimalkan pemanfaatan dan efisiensi biaya Anda saat memakai layanan AWS. Ketika permintaan menurun, AWS Auto Scaling akan menghapus kelebihan kapasitas sumber daya secara otomatis sehingga Anda dapat terhindar dari pengeluaran yang berlebihan.
- Konfigurasikan CloudWatch untuk mengakhiri instans: Instans dapat dikonfigurasi agar berakhir menggunakan alarm CloudWatch. Menggunakan metrik dari proses penonaktifan, implementasikan alarm dengan tindakan Amazon Elastic Compute Cloud. Verifikasi operasi di lingkungan non-produksi sebelum peluncuran.
- Implementasikan kode di dalam beban kerja: Anda dapat menggunakan AWS SDK atau AWS CLI untuk menonaktifkan sumber daya beban kerja. Implementasikan kode di dalam aplikasi yang terintegrasi dengan AWS dan akhiri atau hapus sumber daya yang sudah tidak digunakan.
- Gunakan layanan nirserver: Prioritaskan pembangunan arsitektur nirserver dan arsitektur berbasis peristiwa di AWS untuk membangun dan menjalankan aplikasi Anda. AWS menawarkan beberapa

layanan teknologi nirserver yang secara bawaan menyediakan pemanfaatan sumber daya yang dioptimalkan secara otomatis dan penonaktifan otomatis (penskalaan ke dalam dan penskalaan ke luar). Dengan aplikasi nirserver, pemanfaatan sumber daya dioptimalkan secara otomatis dan Anda tidak pernah membayar pengadaan yang berlebihan.

Sumber daya

Dokumen terkait:

- AWS Auto Scaling
- AWS Trusted Advisor
- Nirserver di AWS
- Membuat Alarm untuk Menghentikan, Mengakhiri, Mem-boot Ulang, atau Memulihkan Instans
- Mulai Menggunakan Amazon EC2 Auto Scaling
- Menambahkan tindakan mengakhiri ke alarm Amazon CloudWatch

Contoh terkait:

- Menjadwalkan penghapusan otomatis tumpukan AWS CloudFormation
- Lab Well-Architected – Nonaktifkan sumber daya secara otomatis (Level 100)
- Servian AWS Auto Cleanup

COST04-BP05 Menegakkan kebijakan retensi data

Tetapkan kebijakan retensi data pada sumber daya yang didukung untuk menangani penghapusan objek sesuai persyaratan organisasi Anda. Identifikasi dan hapus sumber daya yang tidak diperlukan atau tidak digunakan dan objek yang sudah tidak diperlukan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Gunakan kebijakan retensi data dan kebijakan siklus hidup guna mengurangi biaya terkait untuk proses penonaktifan dan biaya penyimpanan untuk sumber daya yang diidentifikasi. Penetapan kebijakan retensi data dan kebijakan siklus hidup untuk menjalankan migrasi dan penghapusan kelas penyimpanan otomatis akan mengurangi keseluruhan biaya penyimpanan di sepanjang masa pakainya. Anda dapat menggunakan Amazon Data Lifecycle Manager untuk mengotomatiskan pembuatan dan penghapusan snapshot Amazon Elastic Block Store dan Amazon Machine Image

(AMI) yang didukung Amazon EBS, dan menggunakan Amazon S3 Intelligent-Tiering atau konfigurasi siklus hidup Amazon S3 untuk mengelola siklus hidup objek Amazon S3 Anda. Anda juga dapat mengimplementasikan kode kustom menggunakan API atau SDK untuk menciptakan kebijakan siklus hidup dan aturan kebijakan untuk objek yang akan dihapus secara otomatis.

Langkah implementasi

- Gunakan Amazon Data Lifecycle Manager: Gunakan kebijakan siklus hidup pada Amazon Data Lifecycle Manager untuk mengotomatiskan penghapusan snapshot Amazon EBS dan AMI yang didukung Amazon EBS.
- Atur konfigurasi siklus hidup pada bucket: Gunakan konfigurasi siklus hidup Amazon S3 pada bucket untuk menetapkan tindakan bagi Amazon S3 yang akan dijalankan selama siklus hidup suatu objek, serta penghapusan pada akhir siklus hidup objek tersebut, berdasarkan persyaratan bisnis Anda.

Sumber daya

Dokumen terkait:

- AWS Trusted Advisor
- Amazon Data Lifecycle Manager
- Cara mengatur konfigurasi siklus hidup pada bucket Amazon S3

Video terkait:

- Mengotomatiskan Snapshot Amazon EBS dengan Amazon Data Lifecycle Manager
- Mengosongkan bucket Amazon S3 menggunakan aturan konfigurasi siklus hidup

Contoh terkait:

- Mengosongkan bucket Amazon S3 menggunakan aturan konfigurasi siklus hidup
- Lab Well-Architected: Menonaktifkan sumber daya secara otomatis (Level 100)

Sumber daya yang hemat

Pertanyaan

- COST 5. Bagaimana cara mengevaluasi biaya ketika Anda memilih layanan?
- COST 6. Bagaimana cara memenuhi target biaya ketika Anda memilih jenis, ukuran, dan jumlah sumber daya?
- COST 7. Bagaimana cara menggunakan model harga untuk mengurangi biaya?
- COST 8. Bagaimana cara Anda merencanakan biaya transfer data?

COST 5. Bagaimana cara mengevaluasi biaya ketika Anda memilih layanan?

Amazon EC2, Amazon EBS, dan Amazon S3 adalah layanan blok penyusun AWS. Layanan terkelola, seperti Amazon RDS dan Amazon DynamoDB, adalah layanan AWS dengan tingkat lebih tinggi, atau tingkat aplikasi. Dengan memilih blok penyusun dan layanan terkelola yang sesuai, Anda dapat mengoptimalkan biaya beban kerja ini. Contohnya, dengan menggunakan layanan terkelola, Anda dapat mengurangi atau menghilangkan sebagian besar dari biaya tambahan untuk administrasi dan operasi, sehingga Anda bebas untuk mengerjakan aplikasi dan aktivitas yang terkait dengan bisnis.

Praktik terbaik

- COST05-BP01 Identifikasi persyaratan organisasi untuk biaya
- COST05-BP02 Menganalisis semua komponen beban kerja
- COST05-BP03 Menjalankan analisis menyeluruh setiap komponen
- COST05-BP04 Memilih perangkat lunak dengan lisensi hemat biaya
- COST05-BP05 Memilih komponen beban kerja ini untuk mengoptimalkan biaya selaras dengan prioritas organisasi
- COST05-BP06 Melakukan analisis biaya untuk penggunaan berbeda seiring waktu

COST05-BP01 Identifikasi persyaratan organisasi untuk biaya

Bekerja dengan anggota tim untuk menentukan keseimbangan antara pengoptimalan biaya dan pilar lainnya, seperti keandalan dan performa, untuk beban kerja ini.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Di sebagian besar organisasi, departemen teknologi informasi (IT) terdiri dari beberapa tim kecil, masing-masing dengan agenda dan area fokusnya sendiri, yang mencerminkan spesialisasi dan

keterampilan anggota timnya. Anda perlu memahami tujuan keseluruhan, prioritas, dan sasaran organisasi Anda, serta bagaimana setiap departemen atau proyek berkontribusi terhadap tujuan ini. Mengategorikan semua sumber daya penting, termasuk personel, peralatan, teknologi, bahan, dan layanan eksternal, sangat penting untuk mencapai tujuan organisasi dan perencanaan anggaran yang komprehensif. Mengadopsi pendekatan sistematis terhadap identifikasi dan pemahaman biaya ini merupakan hal yang sangat penting untuk menyusun rencana biaya yang realistis dan matang untuk organisasi.

Ketika memilih layanan untuk beban kerja Anda, penting bagi Anda untuk memahami prioritas organisasi Anda. Ciptakan keseimbangan antara pengoptimalan biaya dan pilar Kerangka Kerja AWS Well-Architected lainnya, seperti performa dan keandalan. Proses ini harus dilakukan secara sistematis dan teratur untuk mencerminkan perubahan dalam tujuan organisasi, kondisi pasar, dan dinamika operasional. Beban kerja yang biayanya dioptimalkan penuh adalah solusi yang paling selaras dengan persyaratan organisasi Anda, tidak selalu berarti biaya yang paling rendah. Bertemulah dengan semua tim dalam organisasi Anda, seperti tim produk, bisnis, teknis, dan keuangan, untuk mengumpulkan informasi. Evaluasi dampak kompromi antar kepentingan yang bertentangan atau pendekatan alternatif, untuk membantu mengambil keputusan yang lebih tepat saat menentukan ke mana upaya perlu difokuskan atau saat memilih rencana tindakan.

Misalnya, meningkatkan kecepatan masuk pasar untuk fitur baru dapat diprioritaskan daripada optimalisasi biaya, atau Anda bisa memilih basis data relasional untuk data non-relasional guna menyederhanakan upaya migrasi sistem, dibandingkan bermigrasi ke basis data yang dioptimalkan untuk tipe data Anda dan memperbarui aplikasi Anda.

Langkah implementasi

- Identifikasi persyaratan organisasi untuk biaya: Bertemulah dengan anggota-anggota tim dari organisasi Anda, termasuk yang ada di pengelolaan produk, pemilik aplikasi, tim pengembangan dan operasional, serta peran manajemen dan keuangan. Prioritaskan pilar Well-Architected untuk beban kerja ini dan komponennya. Output-nya harus berupa daftar pilar secara berurutan. Anda juga dapat menambahkan bobot pada masing-masing pilar untuk menunjukkan berapa fokus tambahan yang dimiliki sebuah pilar, atau seberapa serupa fokus antara dua pilar.
- Atasi dan dokumentasikan utang teknis: Selama tinjauan beban kerja, atasi utang teknis. Dokumentasikan item backlog untuk meninjau kembali beban kerja pada masa mendatang, dengan tujuan memfaktor ulang atau merancang ulang untuk mengoptimalkannya lebih lanjut. Sangat penting untuk secara jelas mengkomunikasikan kompromi yang dilakukan kepada pemangku kepentingan lainnya.

Sumber daya

Praktik Terbaik Terkait:

- REL11-BP07 Merancang produk Anda agar memenuhi target ketersediaan dan perjanjian tingkat layanan (SLA) waktu aktif
- OPS01-BP06 Mengevaluasi kompromi

Dokumen terkait:

- Kalkulator Total Biaya Kepemilikan (TCO) AWS
- Kelas penyimpanan Amazon S3
- Produk cloud

COST05-BP02 Menganalisis semua komponen beban kerja

Verifikasi bahwa setiap beban kerja telah dianalisis, terlepas dari ukuran atau biaya saat ini. Upaya peninjauan harus menggambarkan manfaat potensial, seperti biaya saat ini dan yang diperkirakan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Komponen beban kerja, yang dirancang untuk memberikan nilai bisnis kepada organisasi, dapat mencakup berbagai layanan. Untuk setiap komponen, Anda mungkin memilih layanan AWS Cloud tertentu guna memenuhi kebutuhan bisnis. Pilihan ini dapat dipengaruhi oleh faktor-faktor seperti pemahaman atau pengalaman sebelumnya dalam menggunakan layanan tersebut.

Setelah mengidentifikasi persyaratan organisasi Anda (seperti yang disebutkan dalam COST05-BP01 Identifikasi persyaratan organisasi untuk biaya), lakukan analisis menyeluruh pada semua komponen dalam beban kerja Anda. Analisis setiap komponen dengan mempertimbangkan biaya dan ukuran saat ini serta proyeksinya. Pertimbangkan biaya analisis terhadap potensi penghematan beban kerja selama siklus hidupnya. Upaya yang dikeluarkan untuk menganalisis semua komponen beban kerja ini harus sesuai dengan potensi penghematan atau peningkatan yang diantisipasi dari pengoptimalan komponen spesifik tersebut. Misalnya, apabila biaya sumber daya yang diajukan adalah 10 USD per bulan, dan dalam prakiraan beban tidak akan melebihi 15 USD per bulan, mengerahkan usaha seharian penuh untuk mengurangi biaya hingga 50% (lima dolar per bulan) dapat melampaui manfaat potensial selama masa pakai sistem. Menggunakan perkiraan berdasarkan data yang lebih cepat dan efisien akan memberikan hasil terbaik secara keseluruhan untuk komponen ini.

Beban kerja dapat berubah seiring waktu, dan rangkaian layanan yang tepat dapat menjadi tidak optimal jika penggunaan atau arsitektur beban kerja berubah. Analisis pilihan layanan harus menggabungkan tingkat penggunaan dan status beban kerja saat ini serta di masa mendatang. Mengimplementasikan layanan untuk penggunaan atau status beban kerja di masa mendatang dapat menghemat biaya keseluruhan dengan meminimalkan atau tanpa memerlukan usaha untuk membuat perubahan di masa mendatang. Misalnya, menggunakan Amazon EMR Serverless mungkin pada awalnya merupakan pilihan yang tepat. Namun, karena konsumsi untuk layanan tersebut meningkat, transisi ke Amazon EMR di Amazon EC2 dapat mengurangi biaya untuk komponen beban kerja tersebut.

Tinjauan strategis dari semua komponen beban kerja, terlepas dari atributnya saat ini, berpotensi menghasilkan peningkatan penting dan penghematan keuangan seiring waktu. Upaya yang dicurahkan dalam proses tinjauan ini harus terencana, dengan pertimbangan yang cermat terhadap potensi keuntungan yang mungkin direalisasikan.

AWS Cost Explorer dan AWS Cost and Usage Report (CUR) dapat menganalisis biaya Bukti Konsep (PoC) atau lingkungan yang sedang berjalan. Anda juga dapat menggunakan AWS Kalkulator Harga untuk memperkirakan biaya beban kerja.

Langkah implementasi

- Buat daftar komponen beban kerja: Buat daftar komponen beban kerja Anda. Ini digunakan sebagai verifikasi untuk memastikan bahwa setiap komponen telah dianalisis. Upaya yang dilakukan harus sesuai dengan kekritisan beban kerja sesuai prioritas organisasi Anda. Mengelompokkan sumber daya menurut fungsinya dapat meningkatkan efisiensi (misalnya penyimpanan basis data produksi, jika terdapat beberapa basis data).
- Prioritaskan daftar komponen: Ambil daftar komponen dan prioritaskan sesuai urutan upaya. Ini biasanya diurutkan berdasarkan pada biaya komponen, dari yang paling mahal ke yang paling murah, atau diurutkan sesuai kekritisan sebagaimana ditentukan oleh prioritas organisasi Anda.
- Lakukan analisis: Untuk setiap komponen di dalam daftar, tinjau opsi dan layanan yang tersedia, kemudian pilih opsi yang paling sesuai dengan prioritas organisasi Anda.

Sumber daya

Dokumen terkait:

- AWS Kalkulator Harga
- AWS Cost Explorer

- Kelas penyimpanan Amazon S3
- Produk cloud

COST05-BP03 Menjalankan analisis menyeluruh setiap komponen

Lihat keseluruhan biaya organisasi dari setiap komponen. Hitung total biaya kepemilikan dengan mempertimbangkan faktor biaya operasi dan manajemen, terutama jika menggunakan layanan terkelola oleh penyedia cloud. Hasil upaya peninjauan harus menggambarkan manfaat potensial (misalnya, waktu yang digunakan untuk menganalisis sebanding dengan biaya komponen).

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Pertimbangkan waktu yang dapat dihemat yang memungkinkan tim Anda untuk berfokus pada penghentian utang teknis, inovasi, fitur yang menambah nilai, dan membangun sesuatu yang membedakan bisnis Anda dari yang lain. Misalnya, Anda mungkin perlu mengangkat dan menggeser (juga disebut host ulang) basis data Anda dari lingkungan on-premise Anda ke cloud secepat mungkin dan kemudian mengoptimalkannya. Sebaiknya cari tahu kemungkinan penghematan yang diperoleh menggunakan layanan terkelola di AWS yang dapat menghilangkan atau mengurangi biaya lisensi. Layanan terkelola di AWS menghilangkan beban administratif dan operasional pemeliharaan layanan, seperti patching atau upgrading OS, dan memungkinkan Anda untuk berfokus pada inovasi.

Karena layanan terkelola beroperasi dengan skala cloud, biaya yang ditawarkan per transaksi atau layanan dapat lebih rendah. Anda dapat melakukan optimalisasi potensial untuk mencapai beberapa manfaat nyata, tanpa mengubah arsitektur inti aplikasi. Sebagai contoh, Anda mungkin berusaha mengurangi waktu yang digunakan untuk mengelola instans basis data dengan cara bermigrasi ke platform basis data sebagai layanan seperti Amazon Relational Database Service (Amazon RDS) atau memigrasikan aplikasi Anda ke platform yang terkelola penuh seperti AWS Elastic Beanstalk.

Biasanya, layanan terkelola memiliki atribut yang dapat Anda atur untuk memastikan kapasitas yang memadai. Anda harus mengatur dan memantau atribut ini agar kapasitas Anda yang berlebih diminimalkan dan kinerja dimaksimalkan. Anda dapat mengubah atribut AWS Managed Services menggunakan Konsol Manajemen AWS atau API dan SDK AWS untuk menyelaraskan kebutuhan sumber daya dengan perubahan permintaan. Misalnya, Anda dapat menambah atau mengurangi jumlah simpul di klaster Amazon EMR (atau klaster Amazon Redshift) untuk menskalakan ke luar atau ke dalam.

Anda juga dapat mengemas beberapa instans dalam sumber daya AWS untuk memungkinkan penggunaan densitas yang lebih tinggi. Misalnya, Anda dapat menyiapkan beberapa basis data kecil dalam satu instans basis data Amazon Relational Database Service (Amazon RDS). Seiring dengan meningkatnya penggunaan, Anda dapat memigrasikan satu dari beberapa basis data ke instans basis data khusus Amazon RDS menggunakan proses pemulihan dan snapshot.

Ketika menyiapkan beban kerja dalam layanan terkelola, Anda harus memahami persyaratan untuk menyesuaikan kapasitas layanan. Persyaratan ini biasanya berupa waktu, upaya, dan dampak apa pun terhadap operasi beban kerja normal. Sumber daya yang disiapkan harus memberikan waktu untuk perubahan apa pun, siapkan biaya tambahan yang diperlukan untuk melakukan hal ini. Upaya kontinu yang diperlukan untuk mengubah layanan dapat dikurangi hingga menjadi hampir nol menggunakan API dan SDK yang diintegrasikan dengan alat pemantauan dan sistem, seperti Amazon CloudWatch.

Amazon RDS, Amazon Redshift, dan Amazon ElastiCache menyediakan layanan basis data terkelola. Amazon Athena, Amazon EMR, dan Amazon OpenSearch Service menyediakan layanan analitik terkelola.

AMS adalah layanan yang mengoperasikan infrastruktur AWS atas nama partner dan pelanggan perusahaan. Layanan ini menyediakan lingkungan yang aman dan patuh sebagai tempat deployment beban kerja Anda. AMS menggunakan model operasi cloud perusahaan dengan otomatisasi untuk memungkinkan Anda memenuhi persyaratan perusahaan, memindahkan ke cloud dengan lebih cepat, serta mengurangi biaya untuk manajemen berkelanjutan.

Langkah implementasi

- Jalankan analisis yang menyeluruh: Menggunakan daftar komponen, kerjakan setiap komponen mulai dari prioritas tertinggi ke yang terendah. Untuk komponen yang diprioritaskan dan membutuhkan biaya mahal, jalankan analisis tambahan dan evaluasi semua opsi yang ada serta dampaknya dalam jangka panjang. Untuk komponen dengan prioritas rendah, ukur apakah perubahan penggunaan akan mengubah prioritas komponen, kemudian jalankan analisis upaya yang sesuai.
- Bandingkan sumber daya terkelola dan tidak terkelola: Pertimbangkan biaya operasional untuk sumber daya yang Anda kelola dan bandingkan dengan sumber daya terkelola AWS. Misalnya, tinjau basis data Anda yang berjalan di instans Amazon EC2 dan bandingkan dengan opsi-opsi Amazon RDS (layanan terkelola AWS) atau Amazon EMR dibandingkan dengan menjalankan Apache Spark di Amazon EC2. Saat beralih dari beban kerja yang dikelola mandiri ke beban kerja terkelola penuh AWS, pelajari opsi-opsi Anda secara cermat. Tiga faktor paling penting untuk

dipertimbangkan adalah tipe layanan terkelola yang ingin Anda gunakan, proses yang akan Anda gunakan untuk memigrasikan data Anda dan memahami model tanggung jawab bersama AWS.

Sumber daya

Dokumen terkait:

- Kalkulator Total Biaya Kepemilikan (TCO) AWS
- Kelas penyimpanan Amazon S3
- Produk AWS Cloud
- Model Tanggung Jawab Bersama AWS

Video terkait:

- Mengapa beralih ke basis data terkelola?
- Apa itu Amazon EMR dan bagaimana cara menggunakannya untuk memproses data?

Contoh terkait:

- Alasan beralih ke basis data terkelola
- Konsolidasikan data dari beberapa basis data SQL Server yang identik ke dalam satu basis data Amazon RDS for SQL Server tunggal menggunakan AWS DMS
- Kirimkan data dalam skala besar ke Amazon Managed Streaming for Apache Kafka (Amazon MSK)
- Memigrasikan aplikasi web ASP.NET ke AWS Elastic Beanstalk

COST05-BP04 Memilih perangkat lunak dengan lisensi hemat biaya

Perangkat lunak sumber terbuka meniadakan biaya lisensi perangkat lunak yang dapat menambah biaya yang besar pada beban kerja. Ketika perangkat lunak berlisensi diperlukan, hindari lisensi yang terikat ke atribut arbitrer seperti CPU, carilah lisensi yang terikat dengan output atau hasil. Besar kecilnya biaya lisensi ini lebih sesuai dengan manfaat yang disediakan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Sumber terbuka berasal dari konteks pengembangan perangkat lunak untuk menunjukkan bahwa perangkat lunak tersebut memenuhi kriteria distribusi gratis tertentu. Perangkat lunak sumber terbuka terdiri dari kode sumber yang dapat diperiksa, dimodifikasi, dan disempurnakan oleh siapa pun. Berdasarkan persyaratan bisnis, keterampilan rekayasawan, perkiraan penggunaan, atau ketergantungan teknologi lainnya, organisasi dapat mempertimbangkan untuk menggunakan perangkat lunak sumber terbuka di AWS untuk meminimalkan biaya lisensi mereka. Dengan kata lain, biaya lisensi perangkat lunak dapat dikurangi melalui penggunaan perangkat lunak sumber terbuka. Penggunaan jenis perangkat lunak ini dapat memberikan pengaruh besar pada biaya beban kerja seiring berubahnya ukuran beban kerja.

Ukur manfaat perangkat lunak berlisensi terhadap total biaya untuk mengoptimalkan beban kerja Anda. Modelkan perubahan dalam lisensi dan bagaimana pengaruhnya terhadap biaya beban kerja Anda. Jika vendor mengubah biaya lisensi basis data Anda, selidiki bagaimana pengaruhnya terhadap keseluruhan efisiensi beban kerja Anda. Pertimbangkan riwayat pengumuman harga dari vendor Anda untuk mengetahui tren perubahan lisensi di seluruh produk mereka. Biaya lisensi juga dapat berubah terlepas dari throughput atau penggunaan, seperti lisensi yang berubah berdasarkan perangkat keras (lisensi terikat CPU). Lisensi jenis ini harus dihindari karena biaya dapat meningkat pesat tanpa hasil yang seimbang.

Misalnya, mengoperasikan instans Amazon EC2 di us-east-1 dengan sistem operasi Linux memungkinkan Anda memangkas biaya sekitar 45%, dibandingkan dengan menjalankan instans Amazon EC2 lain yang berjalan di Windows.

AWS Kalkulator Harga menawarkan cara komprehensif untuk membandingkan biaya berbagai sumber daya dengan opsi lisensi yang berbeda, seperti instans Amazon RDS dan mesin basis data yang berbeda. Selain itu, AWS Cost Explorer memberikan perspektif yang tak ternilai untuk biaya beban kerja yang ada, terutama beban kerja dengan lisensi yang berbeda. Untuk manajemen lisensi, AWS License Manager menawarkan metode yang efisien untuk mengawasi dan menangani lisensi perangkat lunak. Pelanggan dapat menerapkan dan mengoperasionalkan perangkat lunak sumber terbuka pilihan mereka di AWS Cloud.

Langkah implementasi

- Analisis opsi lisensi: Tinjau ketentuan lisensi perangkat lunak yang tersedia. Cari versi sumber terbuka yang memiliki fungsionalitas yang diperlukan, dan cari tahu apakah manfaat dari perangkat lunak berlisensi lebih besar daripada biayanya. Ketentuan yang menguntungkan adalah yang menyelaraskan biaya dengan manfaat yang disediakan.

- Analisis penyedia perangkat lunak: Tinjau harga historis atau perubahan lisensi dari vendor. Cari perubahan yang tidak selaras dengan hasil, seperti ketentuan merugikan yang mengharuskan perangkat lunak dijalankan di perangkat keras atau platform vendor tertentu. Selain itu, cari tahu bagaimana mereka melakukan audit, dan sanksi yang dapat dikenakan.

Sumber daya

Dokumen terkait:

- Open Source at AWS
- Kalkulator Total Biaya Kepemilikan (TCO) AWS
- Kelas penyimpanan Amazon S3
- Produk cloud

Contoh terkait:

- Blog Sumber Terbuka
- Blog Sumber Terbuka AWS
- Penilaian Perizinan dan Optimisasi

COST05-BP05 Memilih komponen beban kerja ini untuk mengoptimalkan biaya selaras dengan prioritas organisasi

Pertimbangkan biaya saat memilih semua komponen untuk beban kerja Anda. Termasuk di antaranya adalah menggunakan layanan terkelola dan tingkat aplikasi atau nirserver, kontainer, atau arsitektur yang didorong peristiwa agar dapat menekan keseluruhan biaya. Minimalkan biaya lisensi menggunakan perangkat lunak sumber terbuka, perangkat lunak yang tidak memiliki biaya lisensi, atau alternatif untuk menekan biaya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Pertimbangkan biaya layanan dan opsi saat memilih semua komponen. Termasuk di dalamnya adalah menggunakan layanan tingkat aplikasi dan terkelola, seperti Amazon Relational Database Service (Amazon RDS), Amazon DynamoDB, Amazon Simple Notification Service (Amazon SNS), dan Amazon Simple Email Service (Amazon SES) untuk mengurangi biaya organisasi keseluruhan.

Gunakan nirserver dan kontainer untuk komputasi, seperti AWS Lambda dan Amazon Simple Storage Service (Amazon S3) untuk situs web statis. Lakukan kontainerisasi aplikasi Anda jika memungkinkan dan gunakan Layanan Kontainer Terkelola AWS seperti Amazon Elastic Container Service (Amazon ECS) atau Amazon Elastic Kubernetes Service (Amazon EKS).

Minimalkan biaya lisensi dengan menggunakan perangkat lunak sumber terbuka, atau perangkat lunak yang tidak memiliki ongkos lisensi (misalnya Amazon Linux untuk beban kerja komputasi atau migrasikan basis data ke Amazon Aurora).

Anda dapat menggunakan layanan nirserver atau tingkat aplikasi seperti Lambda, Amazon Simple Queue Service (Amazon SQS), Amazon SNS, dan Amazon SES. Semua layanan ini menyingkirkan kebutuhan Anda untuk mengelola sumber daya, dan menyediakan fungsi eksekusi kode, layanan pengantrean, dan pengiriman pesan. Manfaat lain layanan-layanan ini adalah menskalakan kinerja dan biaya sesuai dengan penggunaan, sehingga memungkinkan alokasi dan atribusi biaya yang efisien.

Jika menggunakan arsitektur yang didorong peristiwa juga memungkinkan dengan layanan nirserver. Arsitektur yang didorong peristiwa didasarkan pada push, sehingga semuanya terjadi sesuai permintaan saat peristiwa muncul di dalam router. Dengan demikian, Anda tidak akan membayar polling yang terjadi terus-menerus untuk memeriksa peristiwa. Hasilnya adalah konsumsi bandwith jaringan berkurang, penggunaan CPU berkurang, kapasitas armada tidak aktif berkurang, dan handshake SSL/TLS berkurang.

Untuk informasi selengkapnya tentang nirserver, lihat Laporan resmi lensa Aplikasi Nirserver Well-Architected

Langkah implementasi

- Pilih tiap layanan untuk mengoptimalkan biaya: Menggunakan daftar dan analisis yang Anda prioritaskan, pilih setiap opsi yang menyediakan pasangan terbaik untuk prioritas organisasi Anda. Alih-alih meningkatkan kapasitas untuk memenuhi permintaan, pertimbangkan opsi-opsi lain yang dapat memberi Anda kinerja yang lebih baik dengan biaya yang lebih rendah. Sebagai contoh, jika Anda perlu meninjau lalu lintas basis data yang Anda perkirakan di AWS, pertimbangkan untuk meningkatkan ukuran instans atau menggunakan layanan Amazon ElastiCache (Redis atau Memcached) untuk menyediakan mekanisme dalam cache untuk basis data Anda.
- Evaluasi arsitektur yang didorong peristiwa: Dengan menggunakan arsitektur nirserver, Anda juga dapat membangun arsitektur yang didorong peristiwa untuk aplikasi berbasis layanan mikro terdistribusi, yang membantu Anda membangun solusi yang dapat diskalakan, tangguh, tangkas, dan hemat biaya.

Sumber daya

Dokumen terkait:

- AWS Kalkulator Total Biaya Kepemilikan (TCO)
- AWS Nirserver
- Apa Itu Arsitektur yang Didorong Peristiwa?
- Amazon S3 kelas penyimpanan
- Produk cloud
- Amazon ElastiCache (Redis OSS)

Contoh terkait:

- Mulai menggunakan arsitektur yang didorong peristiwa
- Arsitektur yang didorong peristiwa
- Bagaimana Statsig beroperasi 100x lebih hemat biaya menggunakan Amazon ElastiCache (Redis OSS)
- Praktik terbaik menggunakan fungsi AWS Lambda

COST05-BP06 Melakukan analisis biaya untuk penggunaan berbeda seiring waktu

Beban kerja bisa berubah dari waktu ke waktu Beberapa layanan atau fitur lebih hemat biaya pada tingkat penggunaan yang berbeda. Dengan melakukan analisis pada setiap komponen dari waktu ke waktu serta pada penggunaan yang diperkirakan, beban kerja tetap hemat biaya di sepanjang masa pakainya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Ketika AWS merilis layanan dan fitur baru, layanan optimal untuk beban kerja Anda mungkin berubah. Upaya yang diperlukan harus mencerminkan keuntungan potensial. Frekuensi peninjauan beban kerja tergantung pada persyaratan organisasi Anda. Jika beban kerja memiliki biaya yang signifikan, penerapan layanan baru lebih dini akan memaksimalkan penghematan biaya, sehingga manfaatnya bisa lebih besar jika peninjauan lebih sering dilakukan. Pemicu peninjauan lainnya adalah perubahan pola penggunaan. Perubahan yang signifikan pada penggunaan bisa menandakan bahwa layanan alternatif akan lebih optimal.

Jika Anda perlu memindahkan data ke AWS Cloud, Anda dapat memilih berbagai macam layanan yang ditawarkan oleh AWS dan alat-alat partner untuk membantu Anda memigrasikan set data Anda, baik berupa file, basis data, citra mesin, volume blok, bahkan cadangan berupa pita. Misalnya, untuk memindahkan data dalam jumlah besar ke dan dari AWS atau memproses data di edge, Anda dapat menggunakan salah satu perangkat yang dibuat khusus AWS untuk memindahkan data offline berukuran petabita secara hemat biaya. Contoh lainnya adalah untuk kecepatan transfer data yang lebih tinggi, layanan koneksi langsung mungkin lebih murah daripada VPN, yang menyediakan konektivitas konsisten yang diperlukan untuk bisnis Anda.

Berdasarkan analisis biaya untuk berbagai penggunaan seiring waktu, tinjau aktivitas penskalaan Anda. Analisis hasilnya untuk melihat apakah kebijakan penskalaan dapat disesuaikan untuk menambahkan instans dengan beberapa tipe instans dan opsi pembelian. Tinjau pengaturan Anda untuk melihat apakah pengaturan minimum dapat dikurangi untuk melayani permintaan pengguna tetapi dengan ukuran armada yang lebih kecil, dan tambahkan lebih banyak sumber daya untuk memenuhi permintaan tinggi yang diperkirakan.

Jalankan analisis biaya untuk berbagai penggunaan seiring waktu dengan berdiskusi dengan pemangku kepentingan di dalam organisasi Anda dan gunakan fitur prakiraan dari AWS Cost Explorer untuk memprediksi potensi dampak perubahan layanan. Pantau pemicu tingkat penggunaan menggunakan AWS Budgets, alarm penagihan CloudWatch, dan AWS Cost Anomaly Detection untuk mengidentifikasi dan mengimplementasikan layanan yang paling hemat biaya lebih dini.

Langkah implementasi

- Tentukan pola penggunaan yang diprediksi: Bersama organisasi Anda, seperti pemasaran dan pemilik produk, dokumentasikan seperti apa pola penggunaan yang diperkirakan dan yang diprediksi untuk beban kerja. Bersama pemangku kepentingan bisnis, diskusikan riwayat dan prakiraan peningkatan biaya dan penggunaan dan pastikan peningkatan tersebut sesuai dengan persyaratan bisnis. Identifikasi hari, minggu, atau bulan kalender yang Anda perkirakan ada lebih banyak pengguna yang menggunakan sumber daya AWS Anda, yang menandakan bahwa Anda harus meningkatkan kapasitas sumber daya yang ada atau mengadopsi layanan tambahan untuk menekan biaya dan meningkatkan kinerja.
- Jalankan analisis biaya pada penggunaan yang diprediksi: Menggunakan pola penggunaan yang telah ditetapkan, jalankan analisis pada setiap poin ini. Upaya analisis harus mencerminkan hasil potensial. Sebagai contoh, jika ada perubahan besar pada penggunaan, analisis yang mendalam harus dilakukan untuk memastikan biaya dan perubahan yang terjadi. Dengan kata lain, saat biaya meningkat, penggunaan untuk bisnis juga harus meningkat.

Sumber daya

Dokumen terkait:

- Kalkulator Total Biaya Kepemilikan (TCO) AWS
- Kelas penyimpanan Amazon S3
- Produk cloud
- Amazon EC2 Auto Scaling
- Migrasi Data Cloud
- AWS Snow Family

Video terkait:

- AWS OpsHub for Snow Family

COST 6. Bagaimana cara memenuhi target biaya ketika Anda memilih jenis, ukuran, dan jumlah sumber daya?

Pastikan Anda memilih jumlah sumber daya dan ukuran sumber daya yang sesuai untuk tugas yang ada. Anda meminimalkan pemborosan dengan memilih jenis, ukuran, dan jumlah yang paling hemat.

Praktik terbaik

- COST06-BP01 Melakukan pemodelan biaya
- COST06-BP02 Memilih jenis, ukuran, dan jumlah sumber daya berdasarkan data
- COST06-BP03 Pilih tipe, ukuran, dan jumlah sumber daya secara otomatis berdasarkan metrik

COST06-BP01 Melakukan pemodelan biaya

Identifikasi persyaratan organisasi (seperti kebutuhan bisnis dan komitmen yang ada) dan lakukan pemodelan biaya (keseluruhan biaya) dari beban kerja serta setiap komponennya. Lakukan aktivitas tolok ukur untuk beban kerja di bawah berbagai beban yang diprediksi lalu bandingkan biayanya. Upaya pemodelan harus mencerminkan manfaat potensial. Misalnya, waktu yang digunakan sebanding dengan biaya komponen.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Lakukan pemodelan biaya untuk beban kerja Anda dan setiap komponennya guna memahami keseimbangan antar sumber daya, dan temukan ukuran yang tepat untuk setiap sumber daya di dalam beban kerja, dengan tingkat kinerja tertentu. Pemahaman tentang pertimbangan biaya dapat menjadi landasan untuk kasus bisnis dan proses pengambilan keputusan organisasi Anda saat mengevaluasi hasil realisasi nilai untuk deployment beban kerja yang direncanakan.

Lakukan aktivitas tolok ukur untuk beban kerja di bawah berbagai beban yang diprediksi lalu bandingkan biayanya. Hasil upaya pemodelan harus menggambarkan manfaat potensial, misalnya, waktu yang digunakan sebanding dengan biaya komponen atau penghematan yang diprediksi. Untuk praktik terbaik, lihat bagian Tinjauan dari Pilar Efisiensi Kinerja Kerangka Kerja Well-Architected AWS.

Sebagai contoh, untuk membuat pemodelan biaya untuk suatu beban kerja yang terdiri dari sumber daya komputasi, AWS Compute Optimizer dapat membantunya dengan pemodelan biaya untuk beban kerja yang sedang berjalan. Layanan ini menyediakan rekomendasi penyesuaian ukuran untuk sumber daya komputasi berdasarkan riwayat penggunaan. Pastikan CloudWatch Agents di-deploy ke instans Amazon EC2 agar dapat mengumpulkan metrik memori yang membantu Anda mendapatkan rekomendasi yang lebih akurat di dalam AWS Compute Optimizer. Ini adalah sumber data ideal untuk sumber daya komputasi karena ini adalah layanan gratis, yang menggunakan machine learning untuk memberikan beberapa rekomendasi tergantung tingkat risiko.

Terdapat beberapa layanan yang dapat Anda gunakan dengan log kustom sebagai sumber data untuk menyesuaikan ukuran operasi untuk layanan dan komponen beban kerja lain, seperti AWS Trusted Advisor, Amazon CloudWatch dan Amazon CloudWatch Logs. AWS Trusted Advisor memeriksa sumber daya dan menandai sumber daya dengan penggunaan rendah yang dapat membantu Anda menyesuaikan ukuran sumber daya Anda dan membuat pemodelan biaya.

Berikut ini adalah beberapa rekomendasi untuk data dan metrik pemodelan biaya:

- Pemantauan harus mencerminkan pengalaman pengguna secara akurat. Pilih tingkat detail yang tepat untuk periode waktu dan dengan cermat pilih persentil maksimum atau ke-99, bukan rata-rata.
- Pilih tingkat detail yang tepat untuk periode waktu analisis yang diperlukan untuk mencakup siklus beban kerja apa pun. Sebagai contoh, jika dilakukan analisis dua minggu, Anda mungkin mengabaikan siklus pemanfaatan tinggi bulanan, yang dapat menyebabkan pengadaan yang terlalu rendah.

- Pilih layanan AWS yang tepat untuk beban kerja yang Anda rencanakan dengan mempertimbangkan komitmen Anda saat ini, model harga yang dipilih untuk beban kerja lain, dan kemampuan untuk berinovasi lebih cepat dan berfokus pada nilai bisnis inti Anda.

Langkah implementasi

- Lakukan pemodelan biaya untuk sumber daya: Deploy beban kerja atau bukti konsep ke dalam akun terpisah dengan tipe dan ukuran sumber daya tertentu untuk diuji. Jalankan beban kerja dengan data pengujian dan rekam hasil output, beserta data biaya untuk periode pengujian. Kemudian, deploy ulang beban kerja atau ubah tipe dan ukuran sumber daya lalu jalankan ulang pengujian. Sertakan biaya lisensi produk apa pun yang mungkin Anda gunakan dengan sumber daya ini serta perkiraan biaya operasi (tenaga kerja atau teknisi) untuk men-deploy dan mengelola sumber daya ini selama membuat pemodelan biaya. Pertimbangkan pemodelan biaya untuk periode tertentu (jam, harian, bulanan, tahunan, atau tiga tahun).

Sumber daya

Dokumen terkait:

- AWS Auto Scaling
- Mengidentifikasi Peluang untuk Menyesuaikan Ukuran
- Fitur Amazon CloudWatch
- Optimalisasi Biaya: Penyesuaian Ukuran Amazon EC2
- AWS Compute Optimizer
- Kalkulator Harga AWS

Contoh terkait:

- Lakukan Pemodelan Biaya yang Didorong Data
- Hitung biaya konfigurasi sumber daya AWS yang direncanakan
- Pilih alat AWS yang tepat

COST06-BP02 Memilih jenis, ukuran, dan jumlah sumber daya berdasarkan data

Pilih jenis atau ukuran sumber daya berdasarkan data tentang karakteristik sumber daya dan beban kerja. Misalnya, intensif komputasi, memori, throughput, atau tulis. Pilihan ini biasanya dibuat

menggunakan beban kerja versi sebelumnya (on-premise), menggunakan dokumentasi, atau menggunakan sumber informasi lain untuk beban kerja.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Amazon EC2 menyediakan berbagai pilihan jenis instans dengan tingkat CPU, memori, penyimpanan, dan kapasitas jaringan yang berbeda untuk kasus penggunaan yang berbeda. Jenis-jenis instans ini memadukan kemampuan CPU, memori, penyimpanan, dan jaringan yang berbeda-beda, sehingga Anda bisa lebih fleksibel saat memilih kombinasi sumber daya yang tepat untuk proyek Anda. Setiap jenis instans tersedia dalam berbagai ukuran, sehingga Anda dapat menyesuaikan sumber daya berdasarkan tuntutan beban kerja Anda. Untuk menentukan jenis instans yang Anda butuhkan, kumpulkan detail tentang persyaratan sistem aplikasi atau perangkat lunak yang akan Anda jalankan pada instans Anda. Detail ini sebaiknya mencakup hal-hal berikut:

- Sistem operasi
- Jumlah core CPU
- Core GPU
- Jumlah memori sistem (RAM)
- Jenis dan ruang penyimpanan
- Persyaratan bandwidth jaringan

Identifikasi tujuan persyaratan komputasi dan instans mana yang diperlukan, lalu jelajahi berbagai rangkaian instans Amazon EC2. Amazon menawarkan rangkaian jenis instans berikut:

- Instans Tujuan Umum
- Komputasi Dioptimalkan
- Memori Dioptimalkan
- Penyimpanan Dioptimalkan
- Komputasi Dipercepat
- HPC Dioptimalkan

Untuk pemahaman yang lebih mendalam tentang tujuan spesifik dan kasus penggunaan yang dapat dipenuhi oleh rangkaian instans Amazon EC2 tertentu, lihat Jenis Instans AWS.

Pengumpulan persyaratan sistem sangat penting bagi Anda untuk memilih rangkaian instans yang spesifik dan jenis instans yang paling sesuai dengan kebutuhan Anda. Nama jenis instans terdiri dari nama rangkaian dan ukuran instans. Misalnya, instans t2.micro berasal dari rangkaian T2 dan berukuran mikro.

Pilih jenis atau ukuran sumber daya berdasarkan karakteristik sumber daya dan beban kerja (misalnya, intensif komputasi, memori, throughput, atau tulis). Pilihan ini biasanya dibuat menggunakan pemodelan biaya, beban kerja versi sebelumnya (seperti versi on-premise), menggunakan dokumentasi, atau menggunakan sumber informasi lain tentang beban kerja (laporan resmi atau solusi yang dipublikasikan). Menggunakan kalkulator harga atau alat manajemen biaya AWS dapat membantu dalam membuat keputusan yang lebih tepat tentang jenis, ukuran, dan konfigurasi instans.

Langkah implementasi

- Pilih sumber daya berdasarkan data: Gunakan data pemodelan biaya Anda untuk memilih tingkat penggunaan beban kerja yang diantisipasi, dan pilih jenis serta ukuran sumber daya yang ditentukan. Dengan mengandalkan data pemodelan biaya, tentukan jumlah CPU virtual, total memori (GiB), volume penyimpanan instans lokal (GB), volume Amazon EBS, dan tingkat performa jaringan, dengan mempertimbangkan kecepatan transfer data yang diperlukan untuk instans. Selalu buat pilihan berdasarkan analisis mendetail dan data yang akurat untuk mengoptimalkan performa sekaligus mengelola biaya secara efektif.

Sumber daya

Dokumen terkait:

- Jenis Instans AWS
- AWS Auto Scaling
- Fitur Amazon CloudWatch
- Optimisasi Biaya: Rightsizing EC2

Video terkait:

- Selecting the right Amazon EC2 instance for your workloads
- Right Size Your Service

Contoh terkait:

- It just got easier to discover and compare Amazon EC2 instance types

COST06-BP03 Pilih tipe, ukuran, dan jumlah sumber daya secara otomatis berdasarkan metrik

Gunakan metrik dari beban kerja yang sedang berjalan untuk memilih ukuran dan tipe yang tepat untuk mengoptimalkan biaya. Sediakan throughput, ukuran, dan penyimpanan secara tepat untuk layanan komputasi, penyimpanan, data, dan jaringan. Ini dapat dilakukan dengan loop umpan balik seperti penskalaan otomatis atau dengan kode kustom di dalam beban kerja.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Buat loop umpan balik di dalam beban kerja yang menggunakan metrik aktif dari beban kerja yang berjalan untuk melakukan perubahan pada beban kerja tersebut. Anda dapat menggunakan layanan terkelola, seperti AWS Auto Scaling, yang Anda konfigurasikan untuk melakukan operasi penyesuaian ukuran untuk Anda. AWS juga menyediakan API, SDK, dan fitur-fitur yang memungkinkan agar sumber daya dapat dimodifikasi dengan upaya minimal. Anda dapat memprogram beban kerja agar menghentikan dan menjalankan instans Amazon EC2 untuk memungkinkan perubahan ukuran atau tipe instans. Hal ini menyediakan manfaat penyesuaian ukuran sambil menghilangkan hampir semua biaya operasional yang diperlukan untuk melakukan perubahan.

Beberapa layanan AWS memiliki pemilihan tipe atau ukuran otomatis secara bawaan, seperti Amazon Simple Storage ServiceIntelligent-Tiering. Intelligent-Tiering secara otomatis memindahkan data Anda antara dua tingkat akses, akses sering dan akses jarang, berdasarkan pola penggunaan Anda.

Langkah implementasi

- Tingkatkan observabilitas dengan mengonfigurasi metrik beban kerja: Rekam metrik-metrik utama untuk beban kerja. Metrik-metrik tersebut menyediakan indikasi pengalaman pelanggan, seperti output beban kerja, dan penyelarasan dengan perbedaan antartipe dan ukuran sumber daya, seperti penggunaan CPU dan memori. Untuk sumber daya komputasi, analisis data kinerja untuk menyesuaikan ukuran instans Amazon EC2 Anda. Identifikasi instans tidak aktif dan instans dengan pemanfaatan terlalu rendah. Metrik utama yang dicari adalah penggunaan CPU dan pemanfaatan memori (misalnya, 40% pemanfaatan CPU pada 90% waktu seperti yang

dijelaskan dalam Menyesuaikan Ukuran dengan AWS Compute Optimizer dan Pemanfaatan Memori Diaktifkan). Identifikasi instans dengan penggunaan CPU maksimum dan pemanfaatan memori kurang dari 40% selama jangka waktu empat minggu. Ini adalah instans untuk disesuaikan ukurannya guna memangkas biaya. Untuk sumber daya penyimpanan seperti Amazon S3, Anda dapat menggunakan Lensa Penyimpanan Amazon S3, yang memungkinkan Anda melihat 28 metrik di berbagai kategori pada tingkat bucket, dan data riwayat 14 hari di dasbor secara default. Anda dapat memfilter dasbor Lensa Penyimpanan Amazon S3 Anda berdasarkan ringkasan dan pengoptimalan biaya atau peristiwa untuk menganalisis metrik-metrik tertentu.

- Lihat rekomendasi penyesuaian ukuran: Gunakan rekomendasi penyesuaian ukuran di AWS Compute Optimizer dan alat penyesuaian ukuran Amazon EC2 di konsol Manajemen Biaya, atau tinjau AWS Trusted Advisor yang menyesuaikan ukuran sumber daya Anda untuk melakukan penyesuaian pada beban kerja Anda. Penting untuk menggunakan alat yang tepat ketika menyesuaikan ukuran sumber daya yang berbeda dan ikuti pedoman penyesuaian ukuran baik untuk instans Amazon EC2, kelas penyimpanan AWS, atau tipe instans Amazon RDS. Untuk sumber daya penyimpanan, Anda dapat menggunakan Lensa Penyimpanan Amazon S3, yang memberi Anda visibilitas tentang penggunaan penyimpanan objek dan tren aktivitas, serta memberikan rekomendasi yang dapat ditindaklanjuti untuk mengoptimalkan biaya dan menerapkan praktik terbaik perlindungan data. Menggunakan rekomendasi kontekstual yang diambil oleh Lensa Penyimpanan Amazon S3 dari analisis metrik di seluruh organisasi Anda, Anda dapat mengambil langkah-langkah langsung untuk mengoptimalkan penyimpanan.
- Pilih tipe dan ukuran sumber daya secara otomatis berdasarkan metrik: Menggunakan metrik beban kerja, pilih sumber daya beban kerja Anda secara manual atau otomatis. Untuk sumber daya komputasi, konfigurasi AWS Auto Scaling atau implementasi kode di dalam aplikasi Anda dapat menghemat energi yang diperlukan jika diperlukan perubahan yang sering, dan ini dapat berpotensi mengimplementasikan perubahan lebih awal daripada proses manual. Anda dapat meluncurkan dan secara otomatis menskalakan armada Instans Sesuai Permintaan dan Instans Spot di dalam satu grup Auto Scaling. Selain menerima diskon karena menggunakan Instans Spot, Anda dapat menggunakan Instans Terpesan atau Savings Plan untuk menerima tarif diskon untuk harga Instans Sesuai Permintaan biasa. Semua kombinasi faktor tersebut membantu Anda mengoptimalkan penghematan biaya Anda untuk instans Amazon EC2 dan menentukan skala dan kinerja yang diinginkan untuk aplikasi Anda. Anda juga dapat menggunakan strategi pemilihan tipe berbasis atribut (ABS) di Auto Scaling Groups (ASG), yang memungkinkan Anda untuk menyatakan persyaratan instans Anda dalam bentuk rangkaian atribut, seperti vCPU, memori, dan penyimpanan. Anda dapat menggunakan tipe instans generasi lebih baru secara otomatis ketika tipe tersebut dirilis dan mengakses cakupan kapasitas yang lebih luas dengan Instans Spot Amazon EC2. Armada Amazon EC2 dan Amazon EC2 Auto Scaling memilih dan meluncurkan

instans yang cocok dengan atribut yang ditentukan, sehingga tipe instans tidak perlu dipilih secara manual. Untuk sumber daya penyimpanan, Anda dapat menggunakan fitur Amazon S3 Intelligent Tiering dan Amazon EFS Infrequent Access, yang memungkinkan Anda untuk memilih kelas penyimpanan secara otomatis yang menghadirkan penghematan biaya penyimpanan otomatis saat pola akses data berubah, tanpa dampak kinerja atau biaya overhead operasional.

Sumber daya

Dokumen terkait:

- AWS Auto Scaling
- Penyesuaian Ukuran AWS
- AWS Compute Optimizer
- Fitur Amazon CloudWatch
- CloudWatch Penyiapan
- CloudWatch Memublikasikan Metrik Kustom
- Mulai Menggunakan Amazon EC2 Auto Scaling
- Lensa Penyimpanan Amazon S3
- Amazon S3 Intelligent-Tiering
- Amazon EFS Infrequent Access
- Meluncurkan Instans Amazon EC2 Menggunakan SDK

Video terkait:

- Menyesuaikan Ukuran Layanan Anda

Contoh terkait:

- Pemilihan Tipe Instans berbasis atribut untuk Auto Scaling untuk Armada Amazon EC2
- Mengoptimalkan Amazon Elastic Container Service untuk biaya menggunakan penskalaan terjadwal
- Penskalaan Prediktif dengan Amazon EC2 Auto Scaling
- Mengoptimalkan Biaya dan Memperoleh Visibilitas tentang Penggunaan dengan Lensa Penyimpanan Amazon S3

- Lab Well-Architected: Rekomendasi Penyesuaian Ukuran (Level 100)
- Lab Well-Architected: Menyesuaikan Ukuran dengan Mengaktifkan AWS Compute Optimizer dan Pemanfaatan Memori (Level 200)

COST 7. Bagaimana cara menggunakan model harga untuk mengurangi biaya?

Gunakan model harga yang paling sesuai untuk sumber daya Anda untuk meminimalkan pengeluaran.

Praktik terbaik

- COST07-BP01 Melakukan analisis model harga
- COST07-BP02 Memilih Wilayah berdasarkan biaya
- COST07-BP03 Memilih perjanjian pihak ketiga dengan ketentuan hemat biaya
- COST07-BP04 Mengimplementasikan model harga untuk semua komponen beban kerja ini
- COST07-BP05 Melakukan analisis model harga di tingkat akun manajemen

COST07-BP01 Melakukan analisis model harga

Analisis setiap komponen beban kerja. Tentukan apakah komponen dan sumber daya akan dijalankan dalam waktu yang lama (untuk diskon komitmen), atau bersifat dinamis dan dijalankan dalam waktu singkat (untuk spot atau sesuai permintaan). Lakukan analisis pada beban kerja menggunakan rekomendasi dalam alat manajemen biaya dan terapkan aturan bisnis pada rekomendasi tersebut untuk mendapatkan hasil tinggi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

AWS memiliki beberapa model harga yang memungkinkan Anda membayar sumber daya Anda dengan cara yang paling hemat biaya, yang sesuai dengan kebutuhan organisasi Anda dan bergantung pada produk. Bekerjalah dengan tim Anda untuk menentukan model harga yang paling sesuai. Sering kali model harga terdiri dari kombinasi beberapa opsi, yang ditentukan oleh ketersediaan Anda

Instans Sesuai Permintaan memungkinkan Anda membayar kapasitas basis data atau komputasi per jam atau per detik (minimum 60 detik), tergantung pada instans mana yang Anda jalankan, tanpa komitmen jangka panjang atau pembayaran di muka.

Savings Plans merupakan model harga fleksibel yang menawarkan harga murah untuk penggunaan Amazon EC2, Lambda, dan AWS Fargate, sebagai imbalan atas komitmen terhadap sejumlah penggunaan yang konsisten (diukur dalam dolar per jam) selama jangka waktu satu tahun atau tiga tahun.

Instans Spot adalah mekanisme harga Amazon EC2 yang memungkinkan Anda meminta kapasitas komputasi cadangan dengan tarif diskon per jam (diskon hingga 90% dari harga sesuai permintaan) tanpa komitmen di muka.

Instans Terpesan memberi Anda diskon hingga 75 persen dengan pembayaran kapasitas di muka. Untuk detail selengkapnya, lihat Mengoptimalkan biaya dengan reservasi.

Anda mungkin memilih untuk menyertakan Savings Plans untuk sumber daya yang terkait dengan lingkungan produksi, kualitas, dan pengembangan. Alternatifnya, karena sumber daya sandbox hanya diaktifkan saat diperlukan, Anda mungkin memilih model sesuai permintaan untuk sumber daya dalam lingkungan tersebut. Gunakan Instans Spot Amazon untuk mengurangi biaya Amazon EC2 atau gunakan Savings Plans Komputasi untuk mengurangi biaya Amazon EC2, Fargate, dan Lambda. Alat rekomendasi AWS Cost Explorer memberikan peluang untuk diskon komitmen dengan Savings Plans.

Jika Anda pernah membeli Instans Terpesan untuk Amazon EC2 di masa lalu atau Anda telah menjalankan praktik alokasi biaya di dalam organisasi Anda, Anda dapat terus menggunakan Instans Terpesan Amazon EC2 sementara ini. Tetapi, kami menyarankan agar Anda membuat strategi untuk menggunakan Savings Plans di waktu mendatang sebagai mekanisme penghematan biaya yang lebih fleksibel. Anda dapat menyegarkan Rekomendasi Savings Plans (SP) di AWS Cost Management untuk membuat Rekomendasi Savings Plans baru kapan saja. Gunakan Instans Terpesan (RI) untuk mengurangi biaya Amazon RDS, Amazon Redshift, Amazon ElastiCache, dan Amazon OpenSearch Service. Savings Plans dan Instans Terpesan tersedia dalam tiga opsi: semua pembayaran di muka, sebagian pembayaran di muka, dan tanpa pembayaran di muka. Gunakan rekomendasi yang diberikan di AWS Cost Explorer rekomendasi pembelian RI dan SP.

Untuk menemukan peluang untuk beban kerja Spot, gunakan tampilan penggunaan secara keseluruhan per jam, dan cari periode penggunaan atau elastisitas yang berubah secara teratur. Anda dapat menggunakan Instans Spot untuk berbagai aplikasi yang fleksibel dan toleran terhadap kesalahan. Contohnya antara lain server web tanpa status, titik akhir API, aplikasi analitik dan big data, beban kerja terkontainerisasi, CI/CD, dan beban kerja fleksibel lainnya.

Analisis instans Amazon EC2 dan Amazon RDS Anda untuk mengetahui apakah instans tersebut dapat dimatikan ketika Anda tidak menggunakannya (seusai jam kerja dan akhir pekan). Pendekatan

ini akan memungkinkan Anda mengurangi biaya hingga 70% atau lebih dibandingkan penggunaan 24/7. Jika Anda memiliki klaster Amazon Redshift yang hanya perlu disediakan pada waktu tertentu, Anda dapat menghentikan sejenak klaster dan melanjutkannya nanti. Ketika klaster Amazon Redshift atau Instans Amazon EC2 dan Amazon RDS dihentikan, penagihan komputasi terhenti dan hanya biaya penyimpanan yang berlaku.

Perhatikan bahwa reservasi Kapasitas Sesuai Permintaan (ODCR) bukanlah diskon harga. Reservasi Kapasitas dikenakan biaya sesuai tarif Sesuai Permintaan yang setara, baik Anda menjalankan instans dalam kapasitas terpesan atau tidak. Reservasi ini harus dipertimbangkan saat Anda harus memberikan cukup banyak kapasitas untuk sumber daya yang menurut rencana akan Anda jalankan. ODCR tidak harus terikat dengan komitmen jangka panjang, karena ODCR dapat dibatalkan saat Anda tidak lagi membutuhkannya, tetapi ODCR juga dapat mendapat manfaat dari diskon yang diberikan Savings Plans atau Instans Terpesan.

Langkah implementasi

- Analisis elastisitas beban kerja: Menggunakan granularitas per jam dalam Cost Explorer atau dasbor kustom, analisis elastisitas beban kerja Anda. Cari perubahan teratur dalam jumlah instans yang dijalankan. Instans berdurasi pendek merupakan kandidat untuk Instans Spot atau Armada Spot.
  - Well-Architected Lab: Cost Explorer
  - Well-Architected Lab: Visualisasi Biaya
- Tinjau kontrak harga yang ada: Tinjau komitmen atau kontrak saat ini untuk kebutuhan jangka panjang. Analisis apa yang Anda miliki saat ini dan berapa banyak komitmen tersebut digunakan. Manfaatkan diskon kontrak atau perjanjian perusahaan yang sudah ada sebelumnya. Perjanjian Perusahaan memberikan kepada pelanggan opsi untuk menyesuaikan perjanjian yang paling sesuai dengan kebutuhan mereka. Untuk komitmen jangka panjang, pertimbangkan diskon harga terpesan, Instans Terpesan atau Savings Plans untuk jenis instans, kelompok instans, Wilayah AWS, dan Zona Ketersediaan tertentu.
- Lakukan analisis diskon komitmen: Menggunakan Cost Explorer di akun Anda, tinjau rekomendasi Savings Plans dan Instans Terpesan. Untuk memverifikasi bahwa Anda mengimplementasikan rekomendasi yang benar dengan risiko dan diskon yang diperlukan, ikuti Well-Architected Lab.

Sumber daya

Dokumen terkait:

- Mengakses rekomendasi Instans Terpesan
- Opsi pembelian instans
- AWS Perusahaan

Video terkait:

- Hemat hingga 90% dan jalankan beban kerja produksi di Spot

Contoh terkait:

- Well-Architected Lab: Cost Explorer
- Well-Architected Lab: Visualisasi Biaya
- Well-Architected Lab: Model Harga

COST07-BP02 Memilih Wilayah berdasarkan biaya

> Praktik terbaik ini diperbarui dengan panduan baru pada 13 Juli 2023.

Penetapan harga sumber daya dapat berbeda di setiap Wilayah. Identifikasi perbedaan biaya sesuai Wilayah dan lakukan deployment di Wilayah dengan biaya yang lebih tinggi hanya untuk memenuhi persyaratan latensi, residensi data, dan kedaulatan data. Dengan mempertimbangkan biaya Wilayah Anda dapat memperoleh harga keseluruhan yang paling rendah untuk beban kerja ini.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Infrastruktur AWS Cloud bersifat global, di-host di berbagai lokasi di seluruh dunia, dan dibangun berdasarkan Wilayah AWS, Zona Ketersediaan, Zona Lokal, AWS Outposts, dan Zona Wavelength. Wilayah adalah lokasi fisik di dunia dan setiap Wilayah merupakan area geografis yang terpisah dengan beberapa Zona Ketersediaan yang dimiliki AWS. Zona Ketersediaan yang merupakan beberapa lokasi terpisah di dalam setiap Wilayah terdiri dari satu atau beberapa pusat data khusus, masing-masing dengan daya, jaringan, dan konektivitas redundan.

Setiap Wilayah AWS beroperasi sesuai kondisi pasar lokal, dan harga sumber daya berbeda-beda di setiap Wilayah karena perbedaan biaya seperti lahan, fiber, listrik, dan pajak. Pilih Wilayah tertentu

untuk mengoperasikan komponen atau seluruh solusi sehingga Anda dapat menjalankannya dengan harga serendah mungkin secara global. Gunakan Kalkulator AWS untuk menghitung perkiraan biaya beban kerja Anda di berbagai Wilayah dengan mencari layanan berdasarkan tipe lokasi (Wilayah, zona wavelength, dan zona lokal) dan Wilayah.

Saat Anda merancang solusi, praktik terbaik yang dapat diterapkan adalah berusaha menempatkan sumber daya komputasi lebih dekat dengan pengguna untuk memberikan latensi yang lebih rendah dan kedaulatan data yang kuat. Pilih lokasi geografis berdasarkan persyaratan bisnis, privasi data, kinerja, dan keamanan Anda. Untuk aplikasi dengan pengguna akhir global, gunakan beberapa lokasi.

Gunakan Wilayah yang menyediakan harga lebih rendah untuk layanan AWS untuk men-deploy beban kerja Anda jika Anda tidak memiliki keharusan memenuhi persyaratan bisnis, privasi data, dan keamanan. Sebagai contoh, jika Wilayah default Anda adalah ap-southeasth-2 (Sydney), dan jika tidak ada pembatasan (misalnya privasi data dan keamanan) untuk menggunakan Wilayah lain, men-deploy instans Amazon EC2 nonkritis (pengembangan dan pengujian) di Wilayah north-east-1 (N. Virginia) akan lebih murah.

	Kepatuhan	*Latensi*	*Biaya*	*Layanan/Fitur*
Wilayah 1	✓	15 milidetik	$$	✓
Wilayah 2	✓	20 milidetik	$$$	X
Wilayah 3	✓	80 milidetik	$	✓
Wilayah 4	✓	15 milidetik	$$	✓
Wilayah 5	✓	20 milidetik	$$$	X
Wilayah 6	✓	15 milidetik	$	✓
Wilayah 7	✓	80 milidetik	$	✓
Wilayah 8	✓	15 milidetik	$	X

Tabel matriks fitur Wilayah

Tabel matriks di atas menunjukkan kepada kita bahwa Wilayah 4 adalah opsi terbaik untuk skenario yang diberikan ini karena latensinya rendah dibandingkan dengan wilayah lain, layanannya tersedia, dan ini adalah Wilayah yang paling murah.

Langkah implementasi

- Tinjau harga Wilayah AWS: Analisis biaya beban kerja di Wilayah saat ini. Mulai dengan harga tertinggi dari jenis layanan dan penggunaan, lalu hitung biayanya di Wilayah yang tersedia. Jika perkiraan penghematan lebih banyak dari biaya pemindahan komponen atau beban kerja, lakukan migrasi ke Wilayah baru.
- Tinjau persyaratan ulasan untuk deployment multi-Wilayah: Analisis persyaratan dan kewajiban bisnis Anda (privasi data, keamanan, atau kinerja) untuk mencari tahu apakah ada pembatasan bagi Anda untuk menggunakan beberapa Wilayah. Gunakan beberapa Wilayah jika tidak ada kewajiban yang membatasi Anda untuk menggunakan Wilayah tunggal.
- Analisis transfer data yang diperlukan: Pertimbangkan biaya transfer data saat memilih Wilayah. Simpan data Anda dekat dengan sumber dayanya dan dengan pelanggan Anda. Pilih Wilayah AWS berbiaya lebih rendah di mana data mengalir dan transfer data hanya sedikit. Tergantung kebutuhan bisnis Anda untuk transfer data, Anda dapat menggunakan Amazon CloudFront, AWS PrivateLink, AWS Direct Connect, dan AWS Virtual Private Network untuk mengurangi biaya jaringan Anda, meningkatkan kinerja, dan menambah keamanan.

Sumber daya

Dokumen terkait:

- Mengakses rekomendasi Instans Terpesan
- Penetapan harga Amazon EC2
- Opsi pembelian instans
- Tabel Wilayah

Video terkait:

- Hemat hingga 90% dan jalankan beban kerja produksi di Spot

Contoh terkait:

- Ikhtisar Biaya Transfer Data untuk Arsitektur Umum

- Pertimbangan Biaya untuk Deployment Global
- Hal-Hal yang Perlu Dipertimbangkan saat Memilih Wilayah untuk Beban Kerja
- Lab Well-Architected: Membatasi penggunaan layanan berdasarkan Wilayah (Level 200)

COST07-BP03 Memilih perjanjian pihak ketiga dengan ketentuan hemat biaya

Perjanjian dan ketentuan yang hemat biaya memastikan biaya layanan ini dapat disesuaikan dengan manfaat yang disediakan. Pilih perjanjian dan harga yang dapat diskalakan ketika ada keuntungan tambahan yang disediakan untuk organisasi Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Ada beberapa produk di pasar yang dapat membantu Anda mengelola biaya di lingkungan cloud Anda. Produk-produk tersebut mungkin memiliki fitur yang berbeda-beda sesuai kebutuhan pelanggan; misalnya, beberapa di antaranya berfokus pada tata kelola biaya atau visibilitas biaya, sedangkan produk lainnya berfokus pada pengoptimalan biaya. Salah satu faktor penting dalam pengoptimalan dan tata kelola biaya yang efektif adalah penggunaan alat yang tepat dengan fitur yang sesuai kebutuhan dan model harga yang tepat. Model harga untuk produk-produk ini berbeda. Beberapa produk mengenakan persentase tertentu dari tagihan bulanan Anda, sementara produk lainnya mengenakan persentase dari realisasi penghematan Anda. Idealnya, Anda seharusnya hanya membayar sesuai apa yang Anda butuhkan.

Saat Anda menggunakan solusi atau layanan pihak ketiga di cloud, penting untuk memastikan bahwa struktur harganya selaras dengan hasil yang Anda inginkan. Harga harus sesuai dengan hasil dan nilai yang disediakan. Contohnya, dalam perangkat lunak yang mengambil persentase dari penghematan yang dihasilkannya, makin banyak Anda menghemat (hasil), makin tinggi juga biayanya. Perjanjian lisensi yang mengharuskan Anda membayar lebih banyak ketika pengeluaran Anda meningkat mungkin bukan cara terbaik bagi Anda untuk mengoptimalkan biaya. Namun, jika vendor tersebut menawarkan manfaat yang jelas untuk semua elemen dalam tagihan Anda, biaya yang meningkat ini mungkin dapat dijustifikasi.

Misalnya, solusi yang menyediakan rekomendasi untuk Amazon EC2 dan mengenakan sekian persen dari seluruh tagihan Anda akan menjadi makin mahal jika Anda menggunakan layanan lain yang tidak memberikan manfaat. Contoh lainnya adalah layanan terkelola yang dikenakan biaya sekian persen dari biaya sumber daya yang dikelola. Ukuran instans yang lebih besar tidak selalu memerlukan lebih banyak upaya manajemen, tetapi dapat dikenakan biaya lebih banyak. Pastikan

bahwa perjanjian harga layanan ini mencakup fitur atau program pengoptimalan biaya di layanannya untuk mendorong efisiensi.

Pelanggan mungkin merasa produk-produk yang ada di pasar ini lebih canggih atau mudah digunakan. Anda perlu mempertimbangkan biaya produk ini dan memikirkan potensi hasil pengoptimalan biaya dalam jangka panjang.

Langkah implementasi

- Analisis perjanjian dan ketentuan pihak ketiga: Tinjau harga dalam perjanjian pihak ketiga. Lakukan pemodelan untuk berbagai tingkat penggunaan Anda, dan pertimbangkan biaya baru seperti penggunaan layanan baru, atau peningkatan pada layanan saat ini akibat pertumbuhan beban kerja. Tentukan apakah biaya tambahan memberikan keuntungan yang diperlukan pada bisnis Anda.

Sumber daya

Dokumen terkait:

- Mengakses rekomendasi Instans Terpesan
- Opsi pembelian instans

Video terkait:

- Hemat hingga 90% dan jalankan beban kerja produksi di Spot

COST07-BP04 Mengimplementasikan model harga untuk semua komponen beban kerja ini

Sumber daya yang berjalan secara permanen harus menggunakan kapasitas terpesan seperti Savings Plans atau Instans Terpesan. Kapasitas jangka pendek dikonfigurasi untuk menggunakan Instans Spot, atau Armada Spot. Instans Sesuai Permintaan hanya digunakan untuk beban kerja jangka pendek yang tidak bisa dihentikan dan tidak berjalan cukup lama untuk kapasitas terpesan, yakni antara 25% sampai 75% dari periode, tergantung pada tipe sumber daya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

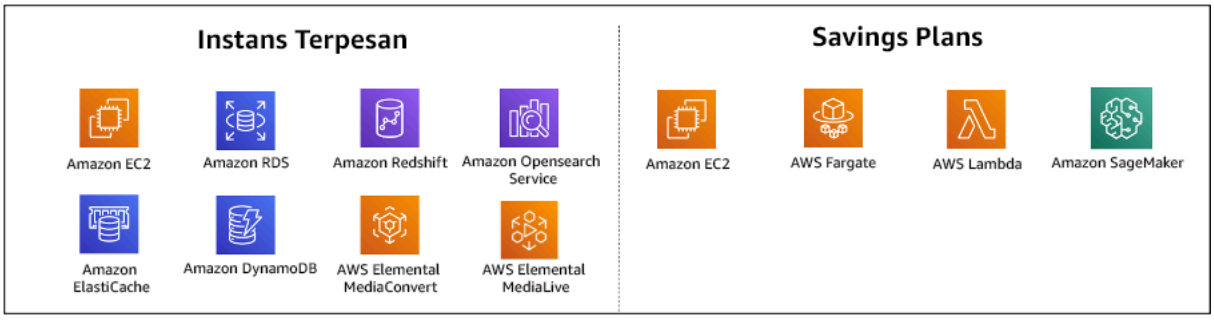
Panduan implementasi

Untuk meningkatkan efisiensi biaya, AWS memberikan beberapa rekomendasi komitmen berdasarkan riwayat penggunaan Anda. Anda dapat menggunakan rekomendasi ini untuk memahami

apa yang dapat Anda hemat, dan bagaimana penggunaan komitmen tersebut. Anda dapat menggunakan layanan ini dengan cara Sesuai Permintaan, Spot, atau membuat komitmen untuk jangka waktu tertentu dan mengurangi biaya sesuai permintaan dengan Instans Terpesan (RI) dan Savings Plans (SP). Selain memahami setiap komponen beban kerja dan beberapa layanan AWS, Anda juga perlu memahami diskon komitmen, opsi pembelian, dan Instans Spot untuk layanan ini agar dapat mengoptimalkan beban kerja Anda.

Pertimbangkan persyaratan komponen beban kerja Anda, dan pahami model harga yang berbeda untuk layanan ini. Tentukan persyaratan ketersediaan komponen ini. Tentukan apakah ada beberapa sumber daya independen yang melakukan fungsinya pada beban kerja, serta apa persyaratan beban kerja dari waktu ke waktu. Bandingkan biaya sumber daya menggunakan model harga Sesuai Permintaan default dan model lain yang dapat diterapkan. Pertimbangkan potensi perubahan apa pun pada sumber daya atau komponen beban kerja.

Sebagai contoh, mari kita lihat Arsitektur Aplikasi Web ini di AWS. Beban kerja sampel ini terdiri dari beberapa layanan AWS, seperti Amazon Route 53, AWS WAF, Amazon CloudFront, instans Amazon EC2, instans Amazon RDS, Penyeimbang Beban, penyimpanan Amazon S3, dan Amazon Elastic File System (Amazon EFS). Anda perlu meninjau setiap layanan ini, dan mengidentifikasi peluang penghematan biaya potensial dengan model harga yang berbeda. Beberapa di antaranya mungkin memenuhi syarat untuk RI atau SP, sedangkan beberapa lainnya mungkin hanya tersedia berdasarkan permintaan. Seperti yang ditunjukkan gambar berikut, beberapa layanan AWS dapat dibeli dengan komitmen menggunakan RI atau SP.



Layanan AWS dibeli dengan komitmen menggunakan Instans Terpesan dan Savings Plans

Langkah implementasi

- Terapkan model harga: Menggunakan hasil analisis, beli Savings Plans, Instans Terpesan, atau terapkan Instans Spot. Jika ini adalah pembelian komitmen pertama Anda, pilih lima atau 10 rekomendasi teratas dalam daftar, lalu pantau dan analisis hasilnya selama satu atau dua bulan ke depan. AWS Cost Management Console membantu Anda dalam proses ini. Tinjau rekomendasi RI atau SP dari konsol, sesuaikan rekomendasi (jenis, pembayaran, serta jangka waktu), dan

tinjau komitmen per jam (misalnya $20 per jam), lalu tambahkan ke keranjang. Diskon berlaku secara otomatis untuk penggunaan yang memenuhi syarat. Beli sejumlah kecil diskon komitmen dalam siklus reguler (misalnya setiap 2 pekan atau setiap bulan). Implementasikan Instans Spot untuk beban kerja yang bisa dihentikan atau stateless. Terakhir, pilih instans Amazon EC2 sesuai permintaan dan alokasikan sumber daya untuk persyaratan yang tersisa.

- Siklus peninjauan beban kerja: Implementasikan siklus peninjauan untuk beban kerja yang secara spesifik menganalisis cakupan model harga. Setelah beban kerja memiliki cakupan yang diperlukan, beli diskon komitmen tambahan secara parsial (setiap beberapa bulan), atau sesuai dengan perubahan penggunaan organisasi Anda.

Sumber daya

Dokumen terkait:

- Memahami rekomendasi Savings Plans Anda
- Mengakses rekomendasi Instans Terpesan
- Cara Membeli Instans Terpesan
- Opsi pembelian instans
- Instans Spot
- Model reservasi untuk layanan AWS lainnya
- Layanan Savings Plans yang Didukung

Video terkait:

- Hemat hingga 90% dan jalankan beban kerja produksi di Spot

Contoh terkait:

- Apa yang harus Anda pertimbangkan sebelum membeli Savings Plans?
- Bagaimana cara menggunakan Cost Explorer untuk menganalisis pengeluaran dan penggunaan saya?

COST07-BP05 Melakukan analisis model harga di tingkat akun manajemen

Periksa alat manajemen biaya dan tagihan dan lihat diskon yang direkomendasikan dengan komitmen dan reservasi untuk melakukan analisis secara teratur di tingkat akun manajemen.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Melakukan pemodelan biaya secara teratur membantu Anda mengimplementasikan peluang untuk mengoptimalkan di beberapa beban kerja. Misalnya, jika beberapa beban kerja menggunakan Instans Sesuai Permintaan, di tingkat agregat, risiko perubahan akan lebih rendah, dan implementasi diskon berbasis komitmen dapat menghasilkan biaya keseluruhan yang lebih rendah. Disarankan agar analisis dilakukan dalam siklus waktu dua minggu hingga satu bulan secara teratur. Ini memungkinkan Anda untuk melakukan pembelian penyesuaian kecil, sehingga cakupan model harga Anda terus berevolusi dengan beban kerja yang terus berubah serta komponennya.

Gunakan alat rekomendasi AWS Cost Explorer untuk menemukan peluang untuk diskon komitmen dalam akun manajemen Anda. Rekomendasi di tingkat akun manajemen dihitung dengan mempertimbangkan penggunaan di semua akun di organisasi AWS Anda yang memiliki diskon Instans Terpesan (RI) atau Savings Plans (SP). Rekomendasi tersebut juga dihitung ketika berbagi diskon diaktifkan untuk merekomendasikan komitmen yang memaksimalkan penghematan di seluruh akun.

Meskipun pembelian di tingkat akun manajemen mengoptimalkan penghematan maksimum pada banyak kasus, mungkin terdapat situasi di mana Anda dapat mempertimbangkan pembelian SP di tingkat akun tertaut, seperti ketika Anda ingin agar diskon diterapkan terlebih dahulu untuk penggunaan pada akun tertaut tersebut. Rekomendasi akun anggota dihitung di tingkat akun individu, untuk memaksimalkan penghematan untuk setiap akun terisolasi. Jika akun Anda memiliki komitmen RI dan SP, urutan penerapannya adalah:

1. RI Zona
2. RI Standar
3. RI Konvertibel
4. Instance Savings Plan
5. Compute Savings Plan

Jika Anda membeli SP di tingkat akun manajemen, penghematan akan diterapkan berdasarkan persentase diskon tertinggi hingga terendah. SP di tingkat akun manajemen melihat semua akun yang ditautkan dan menerapkan penghematan di tempat-tempat dengan nilai diskon tertinggi. Jika Anda ingin membatasi tempat-tempat penghematan diterapkan, Anda dapat membeli Savings Plan di tingkat akun tertaut dan setiap kali akun tersebut menjalankan layanan komputasi yang memenuhi syarat, diskon akan diterapkan di sana terlebih dahulu. Ketika akun tersebut tidak menjalankan layanan komputasi yang memenuhi syarat, diskon akan dibagikan ke akun tertaut lainnya di bawah akun manajemen yang sama. Berbagi diskon diaktifkan secara default, tetapi dapat dinonaktifkan jika diperlukan.

Dalam Keluarga Penagihan Gabungan, Savings Plans diterapkan terlebih dahulu ke penggunaan akun pemilik, kemudian ke penggunaan akun lain. Ini hanya terjadi jika Anda mengaktifkan fitur berbagi. Savings Plans Anda diterapkan ke persentase penghematan tertinggi Anda terlebih dahulu. Jika ada beberapa penggunaan dengan persentase penghematan yang sama, Savings Plans diterapkan pada penggunaan pertama dengan tarif Savings Plans terendah. Savings Plans terus berlaku sampai tidak ada lagi penggunaan yang tersisa atau komitmen Anda habis. Sisa penggunaan lainnya akan ditagih dengan tarif Sesuai Permintaan. Anda dapat menyegarkan Rekomendasi Savings Plans di Manajemen Biaya AWS untuk membuat Rekomendasi Savings Plans baru kapan saja.

Setelah menganalisis fleksibilitas instans, Anda dapat memberikan komitmen dengan mengikuti rekomendasi. Buat pemodelan biaya dengan menganalisis biaya jangka pendek beban kerja dengan potensi opsi sumber daya yang berlainan, menganalisis model harga AWS dan menyelaraskannya dengan persyaratan bisnis Anda untuk menemukan peluang optimisasi biaya dan total biaya kepemilikan.

Langkah implementasi

Lakukan analisis diskon komitmen: Gunakan Cost Explorer di akun Anda, tinjau rekomendasi Savings Plans dan Instans Terpesan. Pastikan Anda memahami rekomendasi Savings Plans, dan perkirakan pengeluaran bulanan serta penghematan bulanan Anda. Tinjau rekomendasi di tingkat akun manajemen yang dihitung dengan mempertimbangkan penggunaan di semua akun anggota di organisasi AWS Anda yang disertai pengaktifan pembagian diskon RI atau Savings Plans, untuk memaksimalkan penghematan di seluruh akun. Anda dapat memverifikasi bahwa Anda telah mengimplementasikan rekomendasi yang benar dengan risiko dan diskon yang diperlukan dengan mengikuti lab Well-Architected.

Sumber daya

Dokumen terkait:

- Bagaimana cara kerja harga AWS?
- Opsi pembelian instans
- Ikhtisar Savings Plan
- Rekomendasi Savings Plan
- Mengakses rekomendasi Instans Terpesan
- Memahami rekomendasi Savings Plans Anda
- Bagaimana Savings Plans diterapkan pada penggunaan AWS Anda?
- Savings Plans dengan Penagihan Gabungan
- Mengaktifkan instans terpesan bersama dan diskon Savings Plans

Video terkait:

- Hemat hingga 90% dan jalankan beban kerja produksi di Spot

Contoh terkait:

- Lab Well-Architected AWS: Model Harga (Tingkat 200)
- Lab Well-Architected AWS: Analisis Model Harga (Tingkat 200)
- Apa yang harus saya pertimbangkan sebelum membeli Savings Plans?
- Bagaimana cara menggunakan Savings Plans bergulir untuk mengurangi risiko komitmen?
- Kapan Menggunakan Instans Spot

COST 8. Bagaimana cara Anda merencanakan biaya transfer data?

Pastikan Anda merencanakan dan memantau biaya transfer data sehingga Anda dapat mengambil keputusan arsitektur untuk meminimalkan biaya. Perubahan arsitektur yang kecil, tetapi efektif dapat secara drastis mengurangi biaya operasional Anda seiring waktu.

Praktik terbaik

- COST08-BP01 Melakukan pemodelan transfer data

- COST08-BP02 Memilih komponen untuk mengoptimalkan biaya transfer data
- COST08-BP03 Mengimplementasikan layanan untuk mengurangi biaya transfer data

COST08-BP01 Melakukan pemodelan transfer data

Kumpulkan persyaratan organisasi dan lakukan pemodelan transfer data beban kerja dan setiap komponennya. Ini mengidentifikasi titik biaya terendah untuk persyaratan transfer data saat ini.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Saat merancang solusi di cloud, biaya transfer data biasanya diabaikan karena sudah terbiasa merancang arsitektur menggunakan pusat data on-premise atau ketidaktahuan. Biaya transfer data di AWS ditentukan oleh sumber, tujuan, dan volume lalu lintas. Memperhitungkan biaya-biaya ini dalam fase desain dapat menghasilkan penghematan biaya. Sangat penting untuk memahami di mana transfer data terjadi dalam beban kerja Anda, berapa biaya transfer, dan manfaat terkaitnya agar total biaya kepemilikan (TCO) dapat diperkirakan secara akurat. Dengan demikian, Anda dapat membuat keputusan yang lebih tepat untuk mengubah atau menerima keputusan arsitektur. Misalnya, Anda memiliki konfigurasi Multi-Zona Ketersediaan yang datanya Anda replikasi antara Zona Ketersediaan tersebut.

Anda membuat model komponen layanan yang mentransfer data dalam beban kerja Anda, dan memutuskan bahwa ini adalah biaya yang dapat diterima (mirip dengan membayar komputasi dan penyimpanan di kedua Zona Ketersediaan) untuk mencapai keandalan dan ketahanan yang diperlukan. Buat model biaya untuk berbagai tingkat penggunaan. Penggunaan beban kerja dapat berubah dari waktu ke waktu, dan beberapa layanan bisa menjadi lebih hemat biaya pada level tertentu.

Saat membuat model transfer data Anda, pikirkan berapa banyak data yang diserap dan dari mana data tersebut berasal. Selain itu, pertimbangkan berapa banyak data yang diproses dan berapa banyak penyimpanan atau kapasitas komputasi yang dibutuhkan. Selama pemodelan, ikuti praktik terbaik jaringan untuk arsitektur beban kerja Anda guna mengoptimalkan potensi biaya transfer data Anda.

AWS Kalkulator Harga dapat membantu Anda melihat perkiraan biaya untuk layanan AWS tertentu dan transfer data yang diharapkan. Jika Anda memiliki beban kerja yang sudah berjalan (untuk tujuan pengujian atau di lingkungan pra-produksi), gunakan AWS Cost Explorer atau AWS Cost and Usage

Report (CUR) untuk memahami dan membuat model biaya transfer data Anda. Konfigurasikan bukti konsep (PoC) atau uji beban kerja Anda, dan lakukan pengujian dengan simulasi beban realistis. Anda dapat membuat model biaya untuk berbagai permintaan beban kerja.

Langkah implementasi

- Identifikasi persyaratan: Apa tujuan utama dan persyaratan bisnis untuk transfer data yang direncanakan antara sumber dan tujuan? Apa hasil bisnis yang diharapkan akhirnya nanti? Kumpulkan persyaratan bisnis dan tentukan hasil yang diharapkan.
- Identifikasi sumber dan tujuan: Apa sumber data dan tujuan transfer data, misalnya di dalam Wilayah AWS, ke layanan AWS, atau ke internet?
  - Transfer data dalam sebuah Wilayah AWS
  - Transfer data antara Wilayah AWS
  - Transfer data ke internet
- Identifikasi klasifikasi data: Apa klasifikasi data untuk transfer data ini? Apa jenis data tersebut? Seberapa besar datanya? Seberapa sering data harus ditransfer? Apakah data sensitif?
- Identifikasi layanan atau alat AWS yang akan digunakan: Layanan AWS mana yang digunakan untuk transfer data ini? Apakah layanan yang sudah disediakan untuk beban kerja lain dapat digunakan?
- Hitung biaya transfer data: Gunakan Harga AWS pemodelan transfer data yang Anda buat sebelumnya guna menghitung biaya transfer data untuk beban kerja. Hitung biaya transfer data di berbagai tingkat penggunaan, baik saat penggunaan beban kerja berkurang maupun bertambah. Jika ada banyak opsi arsitektur beban kerja, maka hitung biaya untuk setiap opsi sebagai perbandingan.
- Hubungkan biaya dengan hasil: Untuk setiap biaya transfer data yang dikenakan, tentukan hasil yang akan dicapai untuk beban kerja. Jika transfer antarkomponen, hasilnya mungkin untuk pemisahan. Jika dilakukan di antara Zona Ketersediaan, tujuannya mungkin untuk redundansi.
- Buat pemodelan transfer data: Setelah mengumpulkan semua informasi, buat pemodelan transfer data dasar konseptual untuk sejumlah kasus penggunaan dan beban kerja yang berbeda.

Sumber daya

Dokumen terkait:

- Solusi caching AWS
- Harga AWS

- Harga Amazon EC2
- Harga Amazon VPC
- Memahami biaya transfer data

Video terkait:

- Memantau dan Mengoptimalkan Biaya Transfer Data Anda
- S3 Transfer Acceleration

Contoh terkait:

- Ikhtisar Biaya Transfer Data untuk Arsitektur Umum
- Panduan Preskriptif AWS untuk Jaringan

COST08-BP02 Memilih komponen untuk mengoptimalkan biaya transfer data

Semua komponen diseleksi, dan arsitektur didesain untuk mengurangi biaya transfer data. Termasuk di dalamnya adalah menggunakan komponen seperti optimalisasi jaringan area luas (WAN) dan konfigurasi Multi-Zona Ketersediaan (AZ)

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Perancangan untuk transfer data meminimalkan biaya transfer data. Ini mungkin melibatkan penggunaan jaringan pengiriman untuk menemukan data yang lebih dekat dengan pengguna, atau penggunaan tautan jaringan khusus dari lokasi Anda ke AWS. Anda juga dapat menggunakan optimalisasi WAN dan optimalisasi aplikasi untuk mengurangi jumlah data yang ditransfer antar komponen.

Saat mentransfer data ke atau di dalam AWS Cloud, penting untuk mengetahui tujuan berdasarkan berbagai kasus penggunaan, sifat data, dan sumber daya jaringan yang tersedia untuk memilih layanan AWS yang tepat guna mengoptimalkan transfer data. AWS menawarkan berbagai layanan transfer data yang disesuaikan untuk beragam persyaratan migrasi data. Pilih opsi penyimpanan data dan transfer data yang tepat berdasarkan kebutuhan bisnis dalam organisasi Anda.

Saat merencanakan atau meninjau arsitektur beban kerja Anda, pertimbangkan hal berikut:

- Gunakan titik akhir VPC dalam AWS: Titik akhir VPC memungkinkan koneksi pribadi antara VPC Anda dan layanan AWS yang didukung. Hal ini memungkinkan Anda untuk menghindari penggunaan internet publik, yang dapat menimbulkan biaya transfer data.
- Gunakan gateway NAT: Gunakan gateway NAT agar instans di subnet privat dapat terhubung ke internet atau ke layanan di luar VPC Anda. Periksa apakah sumber daya di belakang gateway NAT yang mengirimkan lalu lintas terbanyak berada di Zona Ketersediaan yang sama dengan gateway NAT. Jika tidak, buat gateway NAT baru di Zona Ketersediaan yang sama dengan sumber daya untuk mengurangi biaya transfer data lintas AZ.
- Gunakan AWS Direct Connect: Direct Connect melewati internet publik dan membuat koneksi privat langsung antara jaringan on-premise Anda dan AWS. Hal ini bisa lebih hemat biaya dan konsisten daripada mentransfer data dalam jumlah besar melalui internet.
- Hindari transfer data melintasi batas-batas Regional: Transfer data antar-Wilayah AWS (dari satu Wilayah ke Wilayah lainnya) biasanya dikenakan biaya. Jika ingin memilih jalur multi-Wilayah, sebaiknya putuskan dengan cermat. Untuk detail selengkapnya, lihat Skenario Multi-Wilayah.
- Pantau transfer data: Gunakan Amazon CloudWatch dan log aliran VPC untuk menangkap detail tentang transfer data dan penggunaan jaringan Anda. Analisis informasi lalu lintas jaringan yang ditangkap di VPC Anda, seperti alamat IP atau jangkauan ke dan dari antarmuka jaringan.
- Analisis penggunaan jaringan Anda: Gunakan alat pengukuran dan pelaporan sepertiAWS Cost Explorer, CUDOS Dasboards, atau CloudWatch untuk memahami biaya transfer data beban kerja Anda.

Langkah implementasi

- Pilih komponen untuk transfer data: Gunakan pemodelan transfer data yang dijelaskan di COST08-BP01 Melakukan pemodelan transfer data, fokuslah pada di mana biaya terbesar saat ini untuk transfer data, atau di mana kira-kira biaya terbesarnya jika penggunaan beban kerja berubah. Cari arsitektur alternatif, atau komponen tambahan yang menghapus atau mengurangi kebutuhan untuk transfer data, atau menghemat biayanya.

Sumber daya

Praktik Terbaik Terkait:

- COST08-BP01 Melakukan pemodelan transfer data
- COST08-BP03 Mengimplementasikan layanan untuk mengurangi biaya transfer data

Dokumen terkait:

- Migrasi Data Cloud
- Solusi caching AWS
- Kirim konten lebih cepat dengan Amazon CloudFront

Contoh terkait:

- Ikhtisar Biaya Transfer Data untuk Arsitektur Umum
- AWS Network Optimization Tips
- Mengoptimalkan performa dan mengurangi biaya untuk analitik jaringan dengan VPC Flow Log dalam format Apache Parquet

COST08-BP03 Mengimplementasikan layanan untuk mengurangi biaya transfer data

Implementasikan layanan untuk mengurangi transfer data. Misalnya, gunakan lokasi edge atau jaringan pengiriman konten (CDN) untuk mengirimkan konten ke pengguna akhir, membangun lapisan caching di depan server aplikasi atau basis data Anda, dan gunakan koneksi jaringan khusus alih-alih VPN untuk terhubung ke cloud.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Ada berbagai layanan AWS yang dapat membantu Anda mengoptimalkan penggunaan transfer data jaringan Anda. Tergantung komponen beban kerja, jenis, dan arsitektur cloud Anda, semua layanan ini dapat membantu Anda mengompresi, melakukan caching, serta berbagi dan mendistribusikan lalu lintas Anda di cloud.

- Amazon CloudFront adalah jaringan pengiriman konten global yang mengirimkan data dengan latensi rendah dan kecepatan transfer yang tinggi. Layanan ini meng-cache data di lokasi edge di seluruh dunia, yang mengurangi beban di sumber daya Anda. Dengan menggunakan CloudFront, Anda dapat mengurangi upaya administratif dalam pengiriman konten ke sejumlah besar pengguna di seluruh dunia, dengan latensi minimum. Paket penghematan keamanan dapat membantu Anda menghemat hingga 30% pada penggunaan CloudFront Anda jika Anda berencana meningkatkan penggunaan Anda dari waktu ke waktu.

- AWS Direct Connect memungkinkan Anda untuk menetapkan koneksi jaringan khusus ke AWS. Ini dapat mengurangi biaya jaringan, meningkatkan bandwidth, dan menyediakan pengalaman jaringan yang lebih konsisten daripada koneksi berbasis internet.
- Site-to-Site VPN memungkinkan Anda untuk membangun koneksi aman dan privat antara jaringan privat Anda dan jaringan global AWS. Layanan ini ideal untuk kantor kecil atau partner bisnis karena menyediakan konektivitas yang disederhanakan, serta dengan layanan yang dikelola penuh dan elastis.
- Titik akhir VPC memungkinkan konektivitas antara layanan AWS lewat jaringan privat dan dapat digunakan untuk mengurangi transfer data publik serta Gateway NAT . Titik akhir VPC gateway tidak memiliki tarif per jam, serta mendukung Amazon S3 dan Amazon DynamoDB. Titik akhir VPC antarmuka disediakan oleh AWS PrivateLink dan memiliki tarif per jam serta biaya penggunaan per GB.
- biaya gateway NAT menyediakan penskalaan dan manajemen bawaan untuk mengurangi biaya dibandingkan dengan instans NAT mandiri. Tempatkan gateway NAT di dalam Zona Ketersediaan yang sama dengan instans lalu lintas tinggi dan pertimbangkan menggunakan titik akhir VPC untuk instans yang perlu mengakses Amazon DynamoDB atau Amazon S3 guna mengurangi biaya transfer dan pemrosesan data.
- Gunakan perangkat AWS Snow Family yang memiliki sumber daya komputasi untuk mengumpulkan dan memproses data di edge. Perangkat AWS Snow Family (Snowball Edge, Snowball Edge dan Snowmobile) memungkinkan Anda memindahkan data seukuran petabyte ke AWS Cloud secara hemat biaya dan offline.

Langkah implementasi

- Implementasikan layanan: Pilih layanan jaringan AWS yang berlaku berdasarkan layanan Anda, jenis beban kerja yang menggunakan pemodelan transfer data, dan meninjau Log Aliran VPC. Amati area dengan biaya terbesar dan alur volume tertinggi. Tinjau layanan AWS dan nilai apakah ada layanan yang mengurangi atau menghapus transfer, khususnya jaringan dan pengiriman konten. Cari juga layanan caching yang menyediakan akses berulang ke data, atau data dalam jumlah besar.

Sumber daya

Dokumen terkait:

- AWS Direct Connect

- AWS Jelajahi Produk Kami
- solusi caching AWS
- Amazon CloudFront
- AWS Snow Family
- Paket Penghematan Keamanan Amazon CloudFront

Video terkait:

- Memantau dan Mengoptimalkan Biaya Transfer Data Anda
- Serial Pengoptimalan Biaya AWS: CloudFront
- Bagaimana cara mengurangi biaya transfer data untuk gateway NAT saya?

Contoh terkait:

- Cara melakukan tolak bayar layanan bersama: Contoh AWS Transit Gateway
- Memahami detail transfer data AWS secara mendalam dari laporan biaya dan penggunaan menggunakan kueri Athena dan QuickSight
- Ikhtisar Biaya Transfer Data untuk Arsitektur Umum
- Menggunakan AWS Cost Explorer untuk menganalisis biaya transfer data
- Mengoptimalkan Biaya arsitektur AWS Anda dengan memanfaatkan fitur Amazon CloudFront
- Bagaimana cara mengurangi biaya transfer data untuk gateway NAT saya?

Kelola sumber daya pasokan dan permintaan

Pertanyaan

- COST 9. Bagaimana cara mengelola permintaan dan memasok sumber daya?

COST 9. Bagaimana cara mengelola permintaan dan memasok sumber daya?

Untuk beban kerja yang memiliki pengeluaran dan performa seimbang, pastikan semua beban kerja yang Anda bayar akan digunakan dan hindari tingkat penggunaan instans yang terlalu rendah. Metrik penggunaan yang melenceng ke salah satu arah memiliki dampak buruk pada organisasi Anda, baik dalam biaya operasional (performa yang menurun akibat penggunaan yang berlebihan), atau pemborosan pengeluaran AWS (akibat pengadaan yang berlebihan).

Praktik terbaik

- COST09-BP01 Melakukan analisis pada permintaan beban kerja
- COST09-BP02 Implementasikan buffer atau throttle untuk mengelola permintaan
- COST09-BP03 Menyediakan sumber daya secara dinamis

COST09-BP01 Melakukan analisis pada permintaan beban kerja

Analisis permintaan beban kerja dari waktu ke waktu. Verifikasikan bahwa analisis ini mencakup tren musiman dan merepresentasikan secara akurat kondisi operasi di sepanjang masa pakai beban kerja penuh. Upaya analisis harus mencerminkan potensi manfaat, misalnya, waktu yang digunakan sebanding dengan biaya beban kerja.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

Panduan implementasi

Analisis permintaan beban kerja untuk komputasi cloud melibatkan pemahaman pola dan karakteristik tugas komputasi yang dimulai di lingkungan cloud. Analisis ini membantu pengguna mengoptimalkan alokasi sumber daya, mengelola biaya, dan memastikan tingkat kinerja yang diperlukan.

Ketahui persyaratan beban kerja. Persyaratan organisasi Anda harus menunjukkan waktu respons beban kerja untuk permintaan. Waktu respons dapat digunakan untuk menentukan apakah permintaan dikelola, atau apakah pasokan sumber daya harus berubah untuk memenuhi permintaan.

Analisis harus mencakup prediktabilitas dan pengulangan permintaan, tingkat perubahan dalam permintaan, serta jumlah perubahan dalam permintaan. Lakukan analisis dalam periode yang cukup lama sehingga menyertakan variasi musiman apa pun, seperti pemrosesan akhir bulan atau puncak liburan.

Upaya analisis harus mencerminkan potensi manfaat implementasi penskalaan. Amati perkiraan total biaya komponen serta peningkatan atau penurunan penggunaan dan biaya di sepanjang masa pakai beban kerja.

Berikut ini adalah beberapa aspek kunci yang perlu dipertimbangkan saat melakukan analisis permintaan beban kerja untuk komputasi cloud:

1. Metrik pemanfaatan sumber daya dan performa: Analisis bagaimana sumber daya AWS digunakan dari waktu ke waktu. Tentukan pola penggunaan puncak dan di luar puncak untuk mengoptimalkan alokasi sumber daya dan strategi penskalaan. Pantau metrik kinerja seperti waktu

respons, latensi, throughput, dan tingkat kesalahan. Metrik-metrik ini membantu menilai kondisi dan efisiensi infrastruktur cloud secara keseluruhan.

2. Perilaku penskalaan pengguna dan aplikasi: Pahami perilaku pengguna dan bagaimana hal tersebut memengaruhi permintaan beban kerja. Pemeriksaan pola lalu lintas pengguna bermanfaat dalam meningkatkan pengiriman konten dan responsivitas aplikasi. Analisis bagaimana beban kerja diskalakan seiring meningkatnya permintaan. Tentukan apakah parameter penskalaan otomatis dikonfigurasi dengan benar dan efektif untuk menangani fluktuasi beban.
3. Jenis beban kerja: Identifikasi berbagai jenis beban kerja yang berjalan di cloud, seperti pemrosesan batch, pemrosesan data waktu nyata, aplikasi web, basis data, atau machine learning. Setiap jenis beban kerja mungkin memiliki persyaratan sumber daya dan profil kinerja yang berbeda-beda.
4. Perjanjian tingkat layanan (SLA): Bandingkan kinerja aktual dengan SLA untuk memastikan kepatuhan dan mengidentifikasi area yang perlu ditingkatkan.

Anda dapat menggunakan Amazon CloudWatch untuk mengumpulkan dan melacak metrik, memantau file log, mengatur alarm, dan secara otomatis bereaksi terhadap perubahan sumber daya Anda AWS. Anda dapat menggunakan Amazon CloudWatch untuk memperoleh visibilitas di seluruh sistem tentang pemanfaatan sumber daya, kinerja aplikasi, dan kondisi operasional.

Dengan AWS Trusted Advisor, Anda dapat menyediakan sumber daya Anda sesuai praktik terbaik untuk meningkatkan kinerja dan keandalan sistem, meningkatkan keamanan, dan mencari peluang penghematan biaya. Anda juga dapat menonaktifkan instans non-produksi dan menggunakan Amazon CloudWatch dan Auto Scaling agar sesuai dengan peningkatan atau penurunan permintaan.

Akhirnya, Anda dapat menggunakan AWS Cost Explorer atau Quick dengan file AWS Cost and Usage Report (CUR) atau log aplikasi Anda untuk melakukan analisis permintaan beban kerja lanjutan.

Secara keseluruhan, analisis permintaan beban kerja yang komprehensif memungkinkan organisasi untuk mengambil keputusan yang berdasar tentang penyediaan sumber daya, penskalaan, dan pengoptimalan, yang menghasilkan perbaikan kinerja, efisiensi biaya, dan kepuasan pengguna.

Langkah implementasi

- Analisis data beban kerja yang ada: Analisis data dari beban kerja yang ada, versi beban kerja sebelumnya, atau pola penggunaan yang diprediksi. Gunakan Amazon CloudWatch, file log, dan data pemantauan untuk mendapatkan wawasan tentang bagaimana beban kerja digunakan. Analisis siklus beban kerja secara penuh, dan kumpulkan data untuk perubahan musiman apa

pun seperti peristiwa akhir bulan atau akhir tahun. Upaya yang tercermin dalam analisis harus mencerminkan karakteristik beban kerja. Upaya terbesar harus ditempatkan pada beban kerja bernilai tinggi dengan perubahan permintaan terbesar. Upaya terkecil harus ditempatkan pada beban kerja bernilai rendah dengan perubahan permintaan yang minim.

- Prakirakan pengaruh luar: Temui anggota tim dari seluruh organisasi yang dapat memengaruhi atau mengubah permintaan pada beban kerja. Tim umum terdiri dari penjualan, pemasaran, atau pengembangan bisnis. Bekerjalah dengan mereka untuk mengetahui siklus operasi mereka, dan apakah ada peristiwa yang akan mengubah permintaan beban kerja. Prakirakan permintaan beban kerja dengan data ini.

Sumber daya

Dokumen terkait:

- Amazon CloudWatch
- AWS Trusted Advisor
- AWS X-Ray
- AWS Auto Scaling
- Penjadwal Instans AWS
- Memulai dengan Amazon SQS
- AWS Cost Explorer
- Quick

Video terkait:

Contoh terkait:

- Pantau, Lacak, dan Analisis untuk pengoptimalan biaya
- Mencari dan menganalisis log di CloudWatch

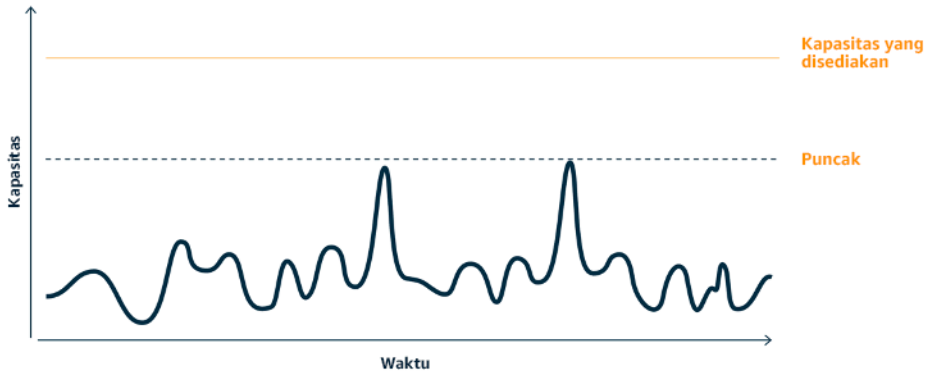
COST09-BP02 Implementasikan buffer atau throttle untuk mengelola permintaan

Buffering dan throttling memodifikasi permintaan di beban kerja Anda, meratakan fluktuasi. Implementasikan throttling ketika klien Anda mencoba ulang. Implementasikan buffering untuk menyimpan permintaan dan menunda pemrosesan ke lain waktu. Pastikan throttle dan buffer Anda didesain sehingga klien menerima respons dalam waktu yang diperlukan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Mengimplementasikan buffer atau throttle sangat penting dalam komputasi cloud untuk mengelola permintaan dan mengurangi penyediaan kapasitas yang diperlukan untuk beban kerja Anda. Untuk performa yang optimal, penting agar Anda mengukur total permintaan, termasuk puncak, laju perubahan permintaan, dan waktu respons yang diperlukan. Ketika klien memiliki kemampuan untuk mengirim ulang permintaan mereka, maka penerapan throttling menjadi praktis. Sebaliknya, untuk klien yang tidak memiliki fungsi coba ulang, mengimplementasikan solusi buffer merupakan pendekatan yang ideal. Buffer tersebut itu merampingkan masuknya permintaan dan mengoptimalkan interaksi aplikasi dengan kecepatan operasional yang bervariasi.



Kurva permintaan dengan dua puncak terpisah yang memerlukan penyediaan kapasitas tinggi

Asumsikan beban kerja dengan kurva permintaan yang ditunjukkan pada gambar berikut. Beban kerja ini memiliki dua puncak. Untuk menangani puncak-puncak ini, kapasitas sumber daya sebagaimana ditunjukkan oleh garis oranye disediakan. Sumber daya dan energi yang digunakan untuk beban kerja ini tidak ditunjukkan dengan area kurva permintaan, tetapi dengan area garis kapasitas yang disediakan, karena kapasitas yang disediakan diperlukan untuk menangani kedua puncak ini. Meratakan kurva permintaan beban kerja dapat membantu Anda mengurangi kapasitas tersedia untuk beban kerja dan mengurangi dampaknya pada lingkungan. Untuk memperlancar puncak, pertimbangkan untuk mengimplementasikan solusi throttling atau buffering.

Untuk memahaminya lebih lanjut, mari kita bahas apa itu throttling dan buffering.

Throttling: Jika sumber permintaan memiliki kemampuan coba ulang, Anda dapat mengimplementasikan throttling. Throttling memberi tahu sumber bahwa jika sistem saat ini tidak dapat melayani permintaan, sumber harus mencoba lagi nanti. Sumber menunggu untuk jangka waktu tertentu, lalu mencoba kembali permintaan. Implementasi throttling memiliki

manfaat membatasi jumlah maksimum sumber daya dan biaya beban kerja. Di AWS, Anda dapat menggunakan Amazon API Gateway untuk mengimplementasikan throttling.

Berbasis buffer: Pendekatan berbasis buffer menggunakan produsen (komponen yang mengirim pesan ke antrean), konsumen (komponen yang menerima pesan dari antrean), dan antrean (yang menyimpan pesan) untuk menyimpan pesan. Pesan dibaca oleh konsumen dan diproses, sehingga pesan dapat dijalankan dengan tingkat yang memenuhi persyaratan bisnis konsumen. Dengan menggunakan metodologi buffer-sentris, pesan dari produsen disimpan dalam antrean atau aliran, siap diakses oleh konsumen dalam kecepatan yang selaras dengan tuntutan operasional mereka.

Di AWS, Anda dapat memilih dari berbagai layanan untuk mengimplementasikan pendekatan buffering. Amazon Simple Queue Service(Amazon SQS) adalah layanan terkelola yang menyediakan antrean yang memungkinkan satu konsumen membaca pesan individual. Amazon Kinesis menyediakan aliran yang memungkinkan banyak konsumen membaca pesan yang sama.

Buffering dan throttling dapat memperlancar setiap puncak dengan memodifikasi permintaan pada beban kerja Anda. Gunakan throttling saat klien mencoba kembali tindakan dan gunakan buffering untuk menahan permintaan dan memprosesnya nanti. Ketika menggunakan pendekatan berbasis buffer, rancang dan konfigurasi beban kerja untuk melayani permintaan dalam waktu yang diperlukan, verifikasi bahwa Anda dapat menangani permintaan kerja duplikat. Analisis permintaan secara keseluruhan, tingkat perubahan, dan waktu respons yang diperlukan untuk ukuran throttle atau buffer yang tepat.

Langkah implementasi

- Analisis persyaratan klien: Analisis permintaan klien untuk menentukan apakah mereka dapat melakukan percobaan ulang. Untuk klien yang tidak dapat melakukan percobaan ulang, buffer perlu diimplementasikan. Analisis permintaan secara keseluruhan, tingkat perubahan, dan waktu respons yang diperlukan untuk menentukan ukuran throttle atau buffer yang diperlukan.
- Implementasikan buffer atau throttle: Implementasikan buffer atau throttle dalam beban kerja. Antrean seperti Amazon Simple Queue Service (Amazon SQS) dapat memberikan buffer untuk komponen beban kerja Anda. Amazon API Gateway dapat memberikan throttling untuk komponen beban kerja Anda.

Sumber daya

Praktik Terbaik Terkait:

- SUS02-BP06 Mengimplementasikan buffering atau throttling untuk meratakan kurva permintaan

- REL05-BP02 Membatasi (throttling) permintaan

Dokumen terkait:

- AWS Auto Scaling
- Penjadwal Instans AWS
- Amazon API Gateway
- Amazon Simple Queue Service
- Mulai menggunakan Amazon SQS
- Amazon Kinesis

Video terkait:

- Memilih Layanan Olah Pesan yang Tepat untuk Aplikasi Terdistribusi Anda

Contoh terkait:

- Mengelola dan memanfatau throttling API di dalam beban kerja Anda
- Throttling a tiered, multi-tenant REST API at scale using API Gateway
- Enabling Tiering and Throttling in a Multi-Tenant Amazon EKS SaaS Solution Using Amazon API Gateway
- Integrasi Aplikasi Menggunakan Antrean dan Pesan

COST09-BP03 Menyediakan sumber daya secara dinamis

Sumber daya disediakan sesuai dengan perencanaan. Penyediaan dapat berdasarkan permintaan, yaitu melalui penskalaan otomatis, atau berdasarkan waktu, yaitu permintaan dapat diprediksi dan sumber daya disediakan berdasarkan waktu. Metode ini dapat meminimalkan penyediaan yang terlalu banyak atau terlalu sedikit.

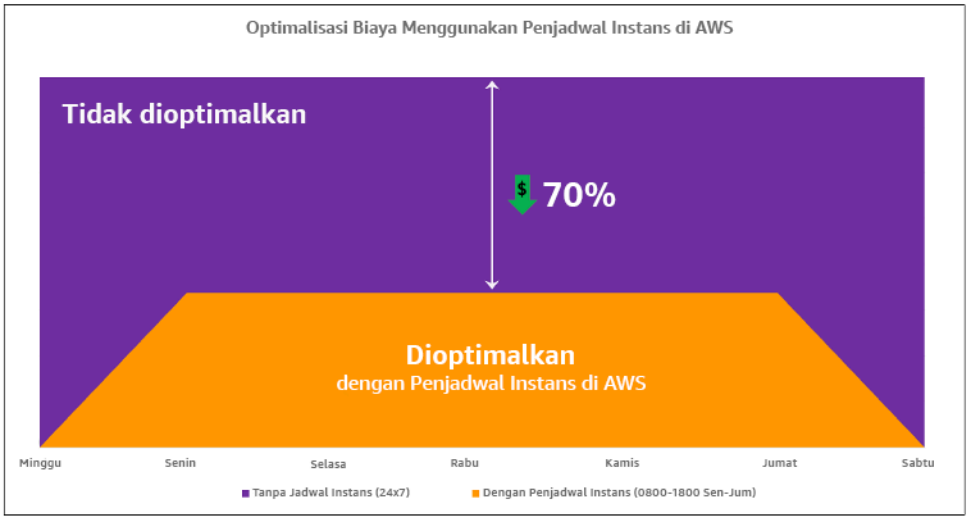
Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Ada sejumlah cara bagi pelanggan AWS untuk meningkatkan sumber daya yang tersedia untuk aplikasi mereka dan menyediakan sumber daya untuk memenuhi permintaan. Salah satu opsinya

adalah dengan menggunakan Penjadwal Instans AWS, yang mengotomatiskan pengaktifan dan penghentian instans Amazon Elastic Compute Cloud (Amazon EC2) dan Amazon Relational Database Service (Amazon RDS). Opsi lainnya adalah menggunakan AWS Auto Scaling, yang memungkinkan Anda untuk secara otomatis menskalakan sumber daya komputasi berdasarkan permintaan aplikasi atau layanan Anda. Memasok sumber daya berdasarkan permintaan akan memungkinkan Anda membayar sumber daya yang Anda gunakan saja, menekan biaya dengan meluncurkan sumber daya hanya saat dibutuhkan, dan menghentikannya saat tidak dibutuhkan.

Penjadwal Instans AWS memungkinkan Anda untuk mengonfigurasi penghentian dan pengaktifan instans Amazon EC2 dan Amazon RDS Anda pada waktu yang telah ditentukan sehingga Anda dapat memenuhi permintaan untuk sumber daya yang sama dalam pola waktu yang konsisten, misalnya setiap hari pengguna mengakses instans Amazon EC2 pada pukul delapan pagi dan mereka tidak perlu mengaksesnya setelah pukul enam sore. Solusi ini membantu mengurangi biaya operasional dengan menghentikan sumber daya yang tidak digunakan dan memulainya saat diperlukan.



Optimalisasi biaya dengan Penjadwal Instans AWS.

Anda juga dapat dengan mudah mengonfigurasi jadwal untuk instans Amazon EC2 Anda di seluruh akun dan Wilayah Anda dengan antarmuka pengguna (UI) yang sederhana menggunakan Pengaturan Cepat AWS Systems Manager. Anda dapat menjadwalkan instans Amazon EC2 atau Amazon RDS dengan Penjadwal Instans AWS dan Anda dapat menghentikan dan memulai instans yang ada. Namun, Anda tidak dapat menghentikan dan memulai instans yang merupakan bagian dari grup Auto Scaling (ASG) Anda atau yang mengelola layanan seperti Amazon Redshift atau Amazon OpenSearch Service. Grup Auto Scaling memiliki penjadwalan sendiri untuk instans di dalam grup dan instans-instans tersebut adalah hasil buatan.

AWS Auto Scaling membantu menyesuaikan kapasitas Anda untuk menjaga kinerja yang stabil dan terprediksi dengan biaya serendah mungkin untuk memenuhi permintaan yang berubah. Ini adalah layanan gratis dan terkelola penuh untuk menskalakan kapasitas aplikasi Anda yang terintegrasi dengan instans Amazon EC2 dan Armada Spot, Amazon ECS, Amazon DynamoDB, dan Amazon Aurora. Auto Scaling menyediakan pencarian sumber daya secara otomatis yang dapat dikonfigurasi untuk membantu Anda menemukan sumber daya dalam beban kerja Anda, dilengkapi dengan strategi penskalaan bawaan untuk mengoptimalkan kinerja, biaya, atau keseimbangan antara keduanya, serta memberikan penskalaan prediktif untuk membantu menangani lonjakan yang rutin terjadi.

Ada beberapa opsi penskalaan yang tersedia untuk menskalakan grup Auto Scaling Anda:

- Pertahankan level instans saat ini setiap saat
- Skalakan secara manual
- Skalakan berdasarkan jadwal
- Skalakan berdasarkan permintaan
- Gunakan penskalaan prediktif

Kebijakan Auto Scaling berbeda-beda dan dapat dikategorikan ke dalam kebijakan penskalaan dinamis dan terjadwal. Kebijakan dinamis ditujukan untuk penskalaan manual atau dinamis, sedangkan kebijakan terjadwal ditujukan untuk penskalaan terjadwal atau prediktif. Anda dapat menggunakan kebijakan penskalaan untuk penskalaan dinamis, terjadwal, dan prediktif. Anda juga dapat menggunakan metrik atau alarm dari Amazon CloudWatch untuk memicu peristiwa penskalaan untuk beban kerja Anda. Sebaiknya Anda menggunakan templat peluncuran, yang memungkinkan Anda untuk mengakses fitur dan perbaikan terbaru. Tidak semua fitur Auto Scaling tersedia saat Anda menggunakan konfigurasi peluncuran. Misalnya, Anda tidak dapat membuat grup Auto Scaling yang meluncurkan Instans Spot dan Sesuai Permintaan atau yang menentukan beberapa jenis instans. Anda harus menggunakan templat peluncuran untuk mengonfigurasi fitur-fitur ini. Saat menggunakan templat peluncuran, kami sarankan Anda melakukan versioning masing-masing. Dengan versioning templat peluncuran, Anda dapat membuat subset dari set lengkap parameter. Kemudian, Anda dapat menggunakannya kembali untuk membuat versi lain dari templat peluncuran yang sama.

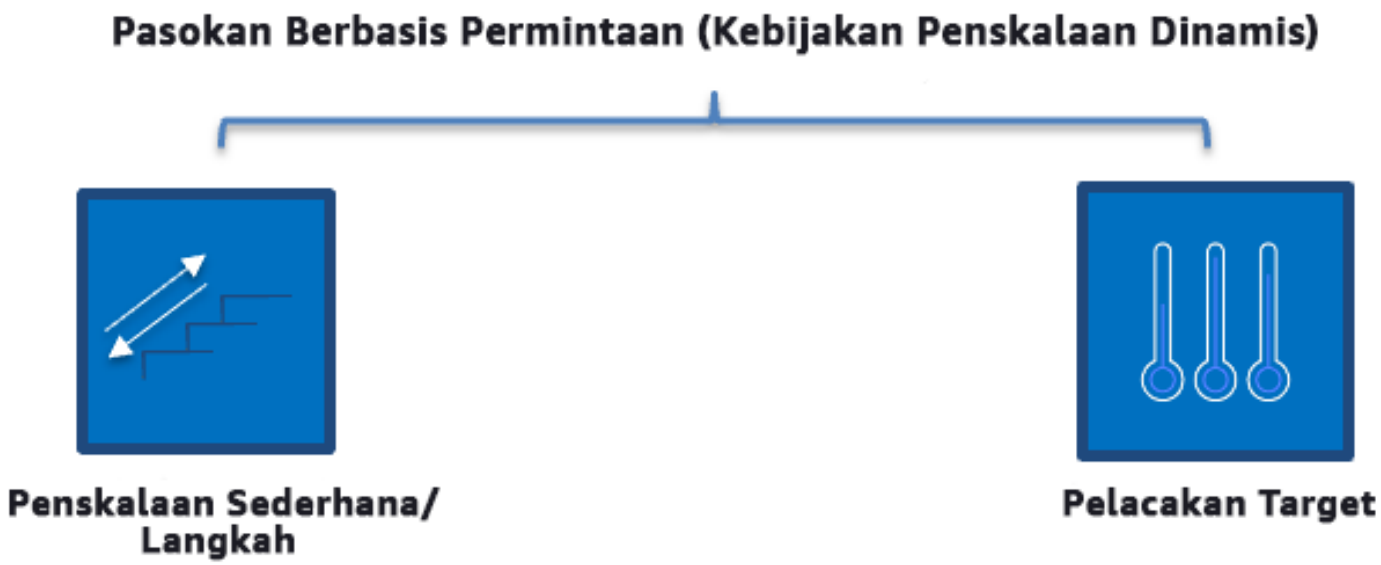
Anda dapat menggunakan AWS Auto Scaling atau menggabungkan penskalaan di dalam kode Anda dengan API atau SDK AWS. Hal ini akan menghemat biaya beban kerja secara keseluruhan dengan menghilangkan biaya operasional yang diperlukan untuk membuat perubahan secara manual di lingkungan Anda, dan perubahan dapat dilakukan dengan lebih cepat. Hal ini juga menyesuaikan

pengadaan sumber daya beban kerja Anda dengan permintaan Anda kapan saja. Agar dapat mengikuti praktik terbaik ini dan menyediakan sumber daya secara dinamis untuk organisasi Anda, Anda harus memahami penskalaan horizontal dan vertikal di AWS Cloud, serta sifat aplikasi yang berjalan di instans Amazon EC2. Sebaiknya tim Manajemen Keuangan Cloud Anda bekerja dengan tim teknis untuk mengikuti praktik terbaik ini.

Elastic Load Balancing (Elastic Load Balancing) membantu penskalaan dengan mendistribusikan permintaan ke beberapa sumber daya. Dengan menggunakan ASG dan Elastic Load Balancing, Anda dapat mengelola permintaan masuk dengan merutekan lalu lintas secara optimal sehingga tidak ada instans yang kewalahan di dalam grup Auto Scaling. Permintaan akan didistribusikan di antara semua target dari grup target secara round-robin tanpa mempertimbangkan kapasitas atau pemanfaatan.

Metrik yang umum dapat berupa metrik Amazon EC2 standar, seperti pemanfaatan CPU, throughput jaringan, dan latensi respons atau permintaan terobservasi Elastic Load Balancing. Jika memungkinkan, Anda harus menggunakan metrik yang menggambarkan pengalaman pelanggan, biasanya berupa metrik kustom yang berasal dari kode aplikasi di dalam beban kerja Anda. Untuk menguraikan cara memenuhi permintaan secara dinamis dalam dokumen ini, kami akan mengelompokkan Auto Scaling ke dalam dua kategori yakni model penyediaan berdasarkan permintaan dan berdasarkan waktu, dan kami akan menjelaskan masing-masing.

Penyediaan berdasarkan permintaan: Manfaatkan elastisitas cloud untuk memasok sumber daya guna memenuhi permintaan yang berubah dengan mengandalkan kondisi permintaan yang mendekati waktu nyata. Untuk penyediaan berdasarkan permintaan, gunakan fitur layanan atau API untuk mengelompokkan jumlah sumber daya cloud secara terprogram di arsitektur Anda. Hal ini memungkinkan Anda untuk menskalakan komponen di arsitektur Anda, serta meningkatkan jumlah sumber daya saat permintaan melonjak guna mempertahankan kinerja, dan mengurangi kapasitas saat permintaan menurun untuk mengurangi biaya.

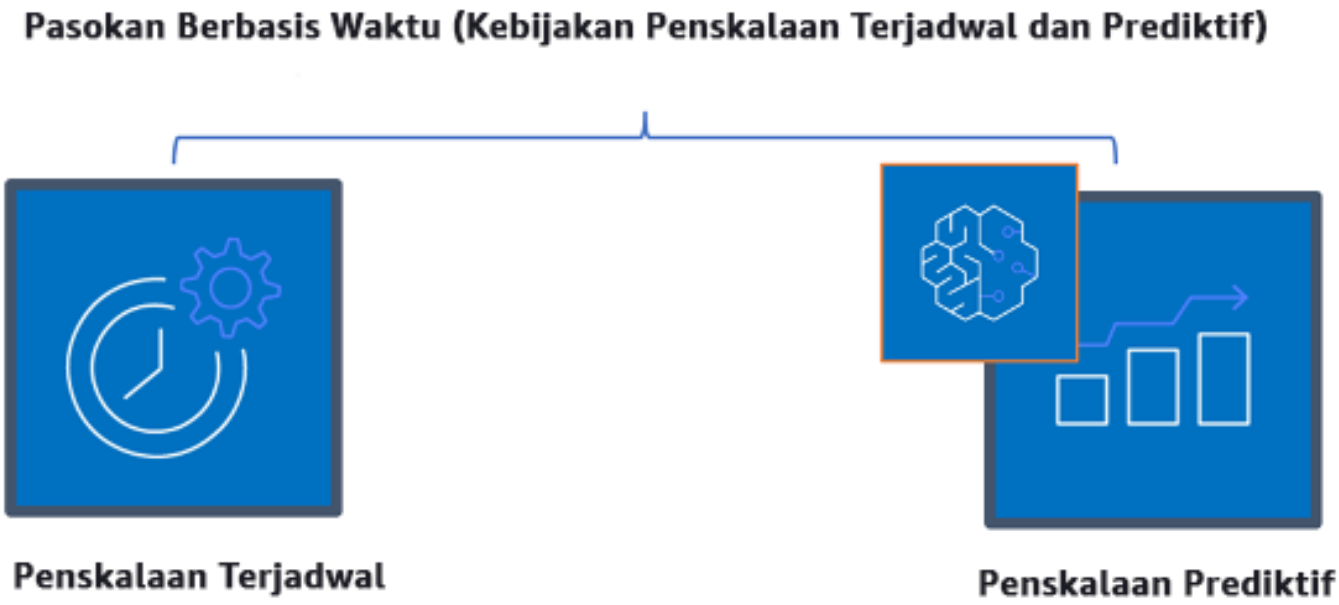


Kebijakan penskalaan dinamis berdasarkan permintaan

- Penskalaan Sederhana/Langkah: Memantau metrik dan menambahkan/menghapus instans sesuai langkah yang ditentukan oleh pelanggan secara manual.
- Pelacakan Target: Mekanisme kontrol mirip termostat yang secara otomatis menambahkan atau menghapus instans untuk mempertahankan metrik pada target yang ditentukan pelanggan.

Jika Anda menggunakan pendekatan berdasarkan permintaan saat merancang, selalu pertimbangkan dua hal yang utama. Pertama, ketahui seberapa cepat Anda harus menyediakan sumber daya baru. Kedua, ketahui bahwa ukuran margin antara penyediaan dan permintaan akan berubah. Anda harus siap menangani rasio perubahan dalam permintaan dan juga siap dengan kegagalan sumber daya.

Penyediaan berdasarkan waktu: Pendekatan berdasarkan waktu selaras dengan kapasitas sumber daya untuk permintaan yang dapat diprediksi atau telah ditentukan berdasarkan waktu. Pendekatan ini biasanya tidak bergantung pada tingkat pemanfaatan sumber daya. Pendekatan berdasarkan waktu memastikan ketersediaan sumber daya pada waktu tertentu saat diperlukan serta dapat disediakan tanpa penundaan yang disebabkan sistem dan prosedur menghidupkan atau pemeriksaan konsistensi. Menggunakan pendekatan berdasarkan waktu, Anda dapat menyediakan sumber daya tambahan atau meningkatkan kapasitas selama periode sibuk.



Kebijakan penskalaan berdasarkan waktu

Anda dapat menggunakan penskalaan otomatis yang terjadwal atau prediktif untuk menerapkan pendekatan berdasarkan waktu. Beban kerja dapat dijadwalkan untuk penskalaan ke luar atau ke dalam pada waktu yang ditentukan (misalnya, awal jam kerja), sehingga sumber daya tersedia saat pengguna datang atau permintaan meningkat. Penskalaan prediktif menggunakan pola untuk menskalakan ke luar sedangkan penskalaan terjadwal menggunakan waktu yang ditetapkan di awal untuk menskalakan ke luar. Anda juga dapat menggunakan strategi pemilihan jenis instans berbasis atribut (ABS) di grup Auto Scaling, yang memungkinkan Anda untuk menyatakan persyaratan instans Anda dalam bentuk rangkaian atribut, seperti vCPU, memori, dan penyimpanan. Ini juga memungkinkan Anda untuk menggunakan tipe instans generasi lebih baru secara otomatis ketika tipe tersebut dirilis dan mengakses cakupan kapasitas yang lebih luas dengan Instans Spot Amazon EC2. Armada Amazon EC2 dan Amazon EC2 Auto Scaling memilih dan meluncurkan instans yang cocok dengan atribut yang ditentukan, sehingga tipe instans tidak perlu dipilih secara manual.

Anda juga dapat memanfaatkan API serta SDK AWS dan AWS CloudFormation untuk menyiapkan atau menarik secara otomatis seluruh lingkungan saat Anda memerlukannya. Pendekatan ini ideal untuk lingkungan pengujian atau pengembangan yang hanya berjalan pada jam kerja atau periode waktu yang ditentukan. Anda dapat menggunakan API untuk menskalakan ukuran sumber daya di dalam suatu lingkungan (penskalaan vertikal). Misalnya, Anda dapat menaikkan skala beban kerja

produksi dengan mengubah ukuran atau kelas instans. Hal ini dapat dicapai dengan menghentikan lalu memulai instans, kemudian memilih kelas dan ukuran instans yang berbeda. Teknik ini juga dapat diterapkan ke sumber daya lain, seperti Volume Elastis Amazon EBS, yang dapat diubah untuk meningkatkan ukuran, menyesuaikan kinerja (IOPS), atau mengubah jenis volume saat sedang digunakan.

Jika Anda menggunakan pendekatan berdasarkan waktu saat merancang, selalu pertimbangkan dua hal utama. Pertama, seberapa konsisten pola penggunaannya? Kedua, apa dampak dari perubahan pola tersebut? Anda dapat meningkatkan akurasi prediksi dengan memantau beban kerja Anda dan menggunakan kecerdasan bisnis. Jika Anda mendapati perubahan yang signifikan dalam pola penggunaan, Anda dapat menyesuaikan waktu untuk memastikan bahwa cakupan tersedia.

Langkah implementasi

- Konfigurasikan penskalaan terjadwal: Untuk perubahan permintaan yang dapat diprediksi, penskalaan berdasarkan waktu dapat memberikan jumlah sumber daya yang benar pada waktu yang tepat. Hal ini juga bermanfaat jika pembuatan dan konfigurasi sumber daya tidak cukup cepat untuk merespons perubahan permintaan. Menggunakan analisis beban kerja konfigurasikan penskalaan terjadwal menggunakan AWS Auto Scaling. Untuk mengonfigurasi penjadwalan berdasarkan waktu, Anda dapat menggunakan penskalaan prediktif dari penskalaan terjadwal untuk menambah jumlah instans Amazon EC2 di dalam grup Auto Scaling Anda sebelumnya berdasarkan perkiraan dan prediksi perubahan beban.
- Konfigurasikan penskalaan prediktif: Penskalaan prediktif memungkinkan Anda untuk menambah jumlah instans Amazon EC2 di dalam grup Auto Scaling Anda sebelum pola harian atau mingguan dalam arus lalu lintas. Jika Anda sering mengalami lonjakan lalu lintas dan aplikasi Anda perlu waktu lama untuk memulai, Anda sebaiknya mempertimbangkan untuk menggunakan penskalaan prediktif. Penskalaan prediktif dapat membantu Anda melakukan penskalaan lebih cepat dengan memulai kapasitas sebelum beban yang diperkirakan dibandingkan dengan penskalaan dinamis saja, yang memiliki sifat reaktif. Sebagai contoh, jika pengguna mulai menggunakan beban kerja Anda pada awal jam kerja dan tidak menggunakannya setelah jam kerja, maka penskalaan prediktif dapat menambah kapasitas sebelum jam kerja, sehingga tidak terjadi keterlambatan penskalaan dinamis dalam menanggapi perubahan lalu lintas.
- Konfigurasikan penskalaan otomatis dinamis: Untuk mengonfigurasi penskalaan berdasarkan metrik beban kerja yang aktif, gunakan Auto Scaling. Gunakan analisis dan konfigurasikan Auto Scaling untuk melakukan peluncuran pada tingkat sumber daya yang benar, dan pastikan beban kerja menyesuaikan skala pada waktu yang diinginkan. Anda dapat meluncurkan dan secara otomatis menskalakan armada Instans Sesuai Permintaan dan Instans Spot di dalam

satu grup Auto Scaling. Selain menerima diskon karena menggunakan Instans Spot, Anda dapat menggunakan Instans Terpesan atau Savings Plan untuk menerima tarif diskon untuk harga Instans Sesuai Permintaan biasa. Semua kombinasi faktor tersebut membantu Anda mengoptimalkan penghematan biaya Anda untuk instans Amazon EC2 dan membantu Anda mendapatkan skala dan kinerja yang diinginkan untuk aplikasi Anda.

Sumber daya

Dokumen terkait:

- AWS Auto Scaling
- Penjadwal Instans AWS
- Skalakan ukuran grup Auto Scaling Anda
- Memulai dengan Amazon EC2 Auto Scaling
- Memulai dengan Amazon SQS
- Penskalaan Terjadwal untuk Amazon EC2 Auto Scaling
- Penskalaan prediktif untuk Amazon EC2 Auto Scaling

Video terkait:

- Kebijakan Penskalaan Pelacakan Target untuk Auto Scaling
- Penjadwal Instans AWS

Contoh terkait:

- Pemilihan Tipe Instans Berbasis Atribut untuk Auto Scaling untuk Armada Amazon EC2
- Mengoptimalkan Amazon Elastic Container Service untuk biaya menggunakan penskalaan terjadwal
- Penskalaan Prediktif dengan Amazon EC2 Auto Scaling
- Bagaimana cara menggunakan Penjadwal Instans dengan CloudFormation untuk menjadwalkan instans Amazon EC2?

Pengoptimalan dari waktu ke waktu

Pertanyaan

- COST 10. Bagaimana cara mengevaluasi layanan baru?
- COST 11. Bagaimana cara mengevaluasi biaya upaya?

COST 10. Bagaimana cara mengevaluasi layanan baru?

Saat AWS merilis fitur dan layanan baru, praktik yang terbaik adalah meninjau keputusan arsitektur yang ada untuk memastikan masih merupakan yang paling hemat.

Praktik terbaik

- COST10-BP01 Mengembangkan proses peninjauan beban kerja
- COST10-BP02 Meninjau dan menganalisis beban kerja ini secara teratur

COST10-BP01 Mengembangkan proses peninjauan beban kerja

Kembangkan proses yang menentukan kriteria dan proses untuk peninjauan beban kerja. Upaya peninjauan harus mencerminkan potensi manfaat. Misalnya, beban kerja inti atau beban kerja dengan nilai di atas sepuluh persen dari tagihan ditinjau setiap kuartal atau setiap enam bulan, sementara beban kerja di bawah sepuluh persen ditinjau setiap tahun.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Tinggi

Panduan implementasi

Untuk memiliki beban kerja yang paling hemat biaya, Anda harus meninjau beban kerja secara rutin untuk mengetahui apakah ada peluang untuk menerapkan layanan, fitur, dan komponen baru. Untuk mendapatkan biaya yang secara keseluruhan lebih rendah, proses harus seimbang dengan potensi jumlah penghematan. Misalnya, beban kerja yang biayanya 50% dari seluruh pengeluaran Anda harus ditinjau secara lebih rutin, dan lebih menyeluruh, daripada beban kerja yang hanya lima persen dari seluruh pengeluaran Anda. Pertimbangkan beberapa faktor eksternal atau volatilitas. Jika beban kerja melayani lokasi geografis atau segmen pasar spesifik, dan perubahan pada area tersebut diprediksi, peninjauan yang lebih sering dapat menghasilkan penghematan biaya. Faktor lain dalam peninjauan adalah upaya untuk mengimplementasikan perubahan. Jika ada biaya besar dalam pengujian dan validasi perubahan, peninjauan harus dilakukan lebih jarang.

Pertimbangkan biaya jangka panjang dalam memelihara komponen dan sumber daya yang usang dan warisan serta ketidakmampuan untuk mengimplementasikan fitur-fitur baru di dalamnya. Biaya pengujian dan validasi saat ini mungkin lebih besar dari manfaat yang diajukan. Namun, seiring

waktu, biaya untuk membuat perubahan mungkin meningkat secara signifikan karena meningkatnya kesenjangan antara beban kerja dan teknologi saat ini, sehingga biayanya menjadi makin besar. Misalnya, biaya pindah ke bahasa pemrograman baru saat ini mungkin tidak hemat biaya. Namun, dalam jangka waktu lima tahun, biaya orang memiliki keterampilan bahasa tersebut mungkin meningkat, dan akibat pertumbuhan beban kerja, Anda akan memindahkan sistem yang lebih besar ke bahasa baru tersebut sehingga memerlukan upaya yang lebih banyak daripada sebelumnya.

Urai beban kerja Anda ke dalam komponen, tentukan biaya komponen (perkiraan saja sudah cukup), kemudian buat daftar faktor (misalnya, upaya dan pasar eksternal) di sebelah setiap komponen. Gunakan indikator ini untuk menentukan frekuensi peninjauan untuk setiap beban kerja. Misalnya, Anda mungkin memiliki server web dengan biaya yang tinggi, upaya perubahan yang rendah, dan faktor eksternal yang tinggi, sehingga frekuensi peninjauannya tinggi. Basis data pusat mungkin memerlukan biaya sedang, upaya perubahannya tinggi, dan faktor eksternalnya rendah, sehingga frekuensi peninjauannya sedang.

Tetapkan proses untuk mengevaluasi layanan, pola desain, tipe sumber daya, dan konfigurasi baru untuk mengoptimalkan biaya beban kerja Anda saat sudah tersedia. Serupa dengan proses peninjauan pilar kinerja dan peninjauan pilar keandalan, identifikasi, validasi, dan prioritaskan aktivitas optimalisasi dan perbaikan serta penyelesaian masalah dan sertakan hal ini ke dalam backlog Anda.

Langkah implementasi

- Tentukan frekuensi peninjauan: Tentukan seberapa sering beban kerja dan komponennya harus ditinjau. Alokasikan waktu dan sumber daya untuk perbaikan kontinu dan tinjau frekuensi untuk meningkatkan efisiensi dan optimalisasi beban kerja Anda. Ini adalah kombinasi faktor yang mungkin berbeda-beda dari satu beban kerja ke beban kerja lain dalam organisasi Anda dan antara komponen-komponen dalam beban kerja tersebut. Faktor yang umum antara lain tingkat kepentingan bagi organisasi yang diukur dalam hal pendapatan dan merek, biaya total untuk menjalankan beban kerja (termasuk biaya operasi dan sumber daya), kompleksitas beban kerja, tingkat kesulitan implementasi perubahan, perjanjian lisensi perangkat lunak apa pun, dan apakah perubahan akan mendatangkan peningkatan yang signifikan pada biaya lisensi akibat lisensi yang merugikan. Komponen dapat ditentukan secara fungsi atau secara teknis, seperti server web dan basis data, atau sumber daya komputasi dan penyimpanan. Seimbangkan faktor sesuai kebutuhan dan kembangkan periode bagi beban kerja serta komponennya. Anda mungkin memutuskan untuk meninjau beban kerja penuh setiap 18 bulan, meninjau server web setiap enam bulan, basis data setiap 12 bulan, komputasi dan penyimpanan jangka pendek setiap enam bulan, serta penyimpanan jangka panjang setiap 12 bulan.

- Tentukan ketelitian peninjauan: Tentukan seberapa besar upaya yang dikeluarkan untuk meninjau beban kerja atau komponen beban kerja. Seperti halnya frekuensi peninjauan, ini adalah keseimbangan antara beberapa faktor. Evaluasi dan prioritaskan peluang untuk peningkatan guna memfokuskan upaya ke hal-hal yang memberikan manfaat paling besar sekaligus memperkirakan seberapa besar upaya yang diperlukan untuk aktivitas-aktivitas tersebut. Jika hasil yang diperkirakan tidak memenuhi tujuan, dan upaya yang diperlukan memerlukan biaya lebih besar, lakukan iterasi menggunakan tindakan alternatif. Proses peninjauan Anda harus mencakup waktu dan sumber daya yang didedikasikan untuk melakukan peningkatan yang bertambah terus menerus. Sebagai contoh, Anda mungkin memutuskan untuk menghabiskan satu pekan untuk menganalisis komponen basis data, satu minggu analisis untuk sumber daya komputasi, dan empat jam untuk peninjauan penyimpanan.

Sumber daya

Dokumen terkait:

- Blog Berita AWS
- Jenis Komputasi Cloud
- Yang Baru dengan AWS

Contoh terkait:

- Layanan Proaktif Dukungan AWS
- Peninjauan beban kerja rutin untuk beban kerja SAP

COST10-BP02 Meninjau dan menganalisis beban kerja ini secara teratur

Beban kerja yang sudah ada ditinjau secara teratur berdasarkan setiap proses yang ditetapkan untuk mengetahui apakah layanan baru dapat diadopsi, layanan yang sudah ada dapat diganti, atau beban kerja dapat dirancang ulang.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

AWS secara terus-menerus menambahkan fitur baru sehingga Anda dapat bereksperimen dan berinovasi dengan lebih cepat dengan teknologi terbaru. AWS Yang Baru memerinci bagaimana

AWS melakukan tindakan ini dan memberikan ikhtisar ringkas tentang pengumuman ekspansi Regional, fitur, dan layanan AWS pada saat dirilis. Anda dapat mendalami peluncuran yang telah diumumkan dan menggunakannya untuk meninjau dan menganalisis beban kerja Anda yang sudah ada. Untuk mewujudkan manfaat dari fitur dan layanan AWS baru, Anda meninjau di beban kerja Anda dan mengimplementasikan fitur serta layanan baru sesuai yang diperlukan. Ini artinya Anda mungkin harus mengganti layanan yang sudah ada yang Anda gunakan untuk beban kerja, atau memodernisasikan beban kerja Anda untuk mengadopsi layanan AWS baru ini. Contohnya, Anda mungkin meninjau beban kerja Anda dan mengganti komponen pesan dengan Amazon Simple Email Service. Ini menghilangkan biaya operasi dan pemeliharaan armada instans, sekaligus memberikan semua fungsionalitas dengan harga yang lebih murah.

Untuk menganalisis beban kerja Anda dan menyorot potensi peluang, Anda juga harus mempertimbangkan cara baru untuk membangun solusi, bukan hanya tentang layanan baru. Tinjau video Ini adalah Arsitektur Saya di AWS untuk mempelajari tentang desain arsitektur pelanggan lainnya, tantangan mereka, dan solusi mereka. Lihat seri Sepenuhnya untuk mengetahui aplikasi layanan AWS dalam dunia nyata dan kisah pelanggan. Anda juga dapat menonton seri video Kembali ke Hal Mendasar yang menjelaskan, memeriksa, dan memerinci praktik terbaik pola arsitektur cloud dasar. Sumber daya lain yakni video Cara Membangun Ini, yang didesain untuk membantu orang dengan ide-ide besar terkait cara mewujudkan produk layak minimum (MVP) menggunakan layanan AWS. Ini adalah cara bagi builder dari seluruh dunia yang memiliki ide kuat untuk mendapatkan panduan arsitektural dari Solutions Architects AWS. Terakhir, Anda dapat meninjau materi sumber daya Memulai, yang memiliki tutorial langkah demi langkah.

Sebelum menjalankan proses peninjauan Anda, ikuti persyaratan bisnis Anda untuk persyaratan beban kerja, keamanan, dan privasi data untuk menggunakan persyaratan performa dan Wilayah atau layanan tertentu sekaligus mengikuti proses peninjauan yang telah disepakati.

Langkah implementasi

- Tinjau beban kerja secara teratur: Menggunakan proses yang telah ditetapkan, lakukan peninjauan dengan frekuensi yang telah ditentukan. Pastikan Anda memberikan upaya yang cukup di setiap komponen. Proses ini serupa dengan proses desain awal ketika Anda memilih layanan untuk pengoptimalan biaya. Analisis layanan dan manfaatnya, kali ini perhitungkan biaya perubahan, tidak hanya manfaat jangka panjang saja.
- Implementasikan layanan baru: Jika hasil analisis mengatakan perubahan harus diimplementasikan, pertama, buat garis dasar beban kerja guna mengetahui biaya saat ini untuk setiap output. Implementasikan perubahan, kemudian lakukan analisis untuk mengonfirmasi biaya baru untuk setiap output.

Sumber daya

Dokumen terkait:

- Blog Berita AWS
- Yang Baru dengan AWS
- Dokumentasi AWS
- AWS Memulai
- Sumber Daya Umum AWS

Video terkait:

- AWS - Ini adalah Arsitektur Saya
- AWS - Kembali ke Hal Mendasar
- AWS - Seri Sepenuhnya
- Cara Membangun Proses Ini

COST 11. Bagaimana cara mengevaluasi biaya upaya?

Praktik terbaik

- COST11-BP01 Melakukan otomatisasi untuk operasi

COST11-BP01 Melakukan otomatisasi untuk operasi

Evaluasi biaya dari upaya untuk operasi di cloud. Kuantifikasikan pengurangan waktu dan upaya untuk tugas administrasi, deployment, dan operasi lainnya menggunakan otomatisasi. Evaluasi waktu dan biaya yang diperlukan untuk upaya operasi dan otomatiskan tugas administrasi untuk mengurangi upaya manusia apabila mungkin.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Mengotomatiskan operasi akan meningkatkan konsistensi dan skalabilitas, memberikan visibilitas, keandalan, dan fleksibilitas lebih tinggi, mengurangi biaya, dan mempercepat inovasi dengan membebaskan sumber daya manusia dan meningkatkan metrik. Otomatisasi mengurangi frekuensi tugas manual, meningkatkan efisiensi, dan menguntungkan perusahaan dengan memberikan pengalaman yang konsisten dan andal saat melakukan deployment, memberikan,

atau mengoperasikan beban kerja. Anda dapat membebaskan sumber daya infrastruktur dari tugas operasional manual dan menggunakannya untuk inovasi dan tugas nilai lebih tinggi, sehingga meningkatkan hasil bisnis. Perusahaan memerlukan cara yang terbukti dan teruji untuk mengelola beban kerja mereka di cloud. Solusi tersebut harus aman, cepat, dan hemat biaya, dengan risiko minimum dan keandalan maksimum.

Mulai dengan memprioritaskan operasi Anda berdasarkan upaya yang diperlukan dengan melihat biaya operasi secara keseluruhan di cloud. Contohnya, berapa lama waktu yang diperlukan untuk melakukan deployment sumber daya baru di cloud, membuat perubahan optimisasi pada yang sudah ada, atau mengimplementasikan konfigurasi yang diperlukan? Lihat biaya total dari tindakan manusia dengan memperhitungkan biaya operasi dan manajemen. Prioritaskan otomatisasi untuk tugas administrasi guna mengurangi upaya manusia. Upaya peninjauan harus mencerminkan potensi manfaat. Contohnya, waktu yang diluangkan untuk melakukan tugas secara manual dibandingkan secara otomatis. Prioritaskan otomatisasi aktivitas bernilai tinggi yang repetitif. Aktivitas yang menimbulkan risiko lebih tinggi dalam hal kesalahan manusia biasanya merupakan tempat yang baik untuk memulai otomatisasi karena sering kali risikonya menimbulkan biaya operasional tambahan yang tidak diinginkan (seperti jam kerja tambahan untuk tim operasi).

Dengan menggunakan layanan AWS, alat, atau produk pihak ketiga, Anda dapat memilih otomatisasi AWS mana yang akan diimplementasikan dan disesuaikan untuk persyaratan spesifik Anda. Tabel berikut menunjukkan beberapa kemampuan dan fungsi operasi inti yang dapat Anda capai dengan layanan AWS untuk mengotomatiskan administrasi dan operasi:

- AWS Audit Manager: Terus audit penggunaan AWS Anda untuk menyederhanakan penilaian kepatuhan dan risiko
- AWS Backup: Kelola dan otomatiskan perlindungan data secara terpusat.
- AWS Config: Konfigurasikan sumber daya komputasi, nilai, audit, dan evaluasi konfigurasi dan inventaris sumber daya.
- AWS CloudFormation: Luncurkan sumber daya yang mudah tersedia dengan infrastruktur sebagai kode.
- AWS CloudTrail: Manajemen perubahan IT, kepatuhan, dan kontrol.
- Amazon EventBridge: Jadwalkan acara dan picu AWS Lambda untuk mengambil tindakan.
- AWS Lambda: Otomatiskan proses repetitif dengan memicunya menggunakan acara atau dengan menjalankannya sesuai jadwal tetap dengan Amazon EventBridge.
- AWS Systems Manager: Mulai dan hentikan beban kerja, patch sistem operasi, otomatiskan konfigurasi, dan lakukan manajemen yang berkelanjutan.

- AWS Step Functions: Jadwalkan pekerjaan dan otomatiskan alur kerja.
- AWS Service Catalog: Pemakaian templat dan infrastruktur sebagai kode dengan kepatuhan dan kontrol.

Pertimbangkan penghematan waktu yang memungkinkan tim Anda untuk berfokus pada penghentian utang teknis, inovasi, dan fitur yang menambah nilai. Misalnya, Anda mungkin perlu mengangkat dan menggeser lingkungan on-premise Anda ke cloud secepat mungkin dan kemudian mengoptimalkannya. Sebaiknya cari tahu penghematan apa saja yang dapat Anda realisasikan dengan layanan terkelola penuh menggunakan AWS yang menghilangkan atau mengurangi biaya lisensi seperti Amazon Relational Database Service, Amazon EMR, Amazon WorkSpaces, dan Amazon SageMaker AI. Layanan terkelola menghilangkan beban administratif dan operasional pemeliharaan layanan, sehingga Anda dapat berfokus pada inovasi. Selain itu, karena layanan terkelola beroperasi di skala cloud, biaya yang ditawarkan per transaksi atau layanan dapat lebih rendah.

Jika Anda ingin mengadopsi otomatisasi dengan segera menggunakan layanan dan produk AWS dan jika tidak ada keterampilannya dalam organisasi Anda, hubungi AWS Managed Services (AMS), Layanan Profesional AWS, atau Partner AWS untuk meningkatkan adopsi otomatisasi dan meningkatkan keunggulan operasional Anda di cloud.

AWS Managed Services (AMS) adalah layanan yang mengoperasikan infrastruktur AWS atas nama partner dan pelanggan perusahaan. Layanan ini menyediakan lingkungan yang aman dan patuh sebagai tempat deployment beban kerja Anda. AMS menggunakan model operasi cloud perusahaan dengan otomatisasi untuk memungkinkan Anda memenuhi persyaratan perusahaan, memindahkan ke cloud dengan lebih cepat, serta mengurangi biaya untuk manajemen berkelanjutan.

AWS Layanan Profesional juga dapat membantu Anda mendapatkan hasil bisnis yang diinginkan dan mengotomatiskan operasi dengan AWS. AWS Layanan Profesional memberikan praktik spesialisasi global untuk mendukung upaya Anda dalam fokus area komputasi cloud perusahaan. Praktik spesialisasi memberikan panduan bertarget melalui praktik terbaik, kerangka kerja, alat, dan layanan untuk semua area subjek industri, teknologi, dan solusi. Praktik ini membantu pelanggan melakukan deployment operasi IT otomatis yang andal dan tangkas, dan kemampuan tata kelola yang dioptimalkan untuk pusat cloud.

Langkah implementasi

- Bangun satu kali dan lakukan banyak deployment: Gunakan infrastruktur sebagai kode seperti AWS CloudFormation, AWS SDK, atau AWS Command Line Interface (AWS CLI) untuk melakukan

deployment satu kali dan gunakan berulang kali untuk lingkungan yang sama atau untuk skenario pemulihan bencana. Beri tanda saat melakukan deployment untuk melacak pemakaian Anda sebagaimana ditetapkan dalam praktik terbaik lainnya. Gunakan AWS Launch Wizard untuk mengurangi waktu deployment sejumlah besar beban kerja populer perusahaan. AWS Launch Wizard memandu Anda dalam penentuan ukuran, konfigurasi, dan deployment beban kerja perusahaan mengikuti praktik terbaik AWS. Anda juga dapat menggunakan AWS Service Catalog, yang membantu Anda membuat dan mengelola templat yang disetujui infrastruktur sebagai kode untuk digunakan di AWS sehingga siapa pun dapat menemukan sumber daya cloud yang disetujui dengan layanan mandiri.

- Otomatiskan operasi: Jalankan operasi rutin secara otomatis tanpa intervensi manusia. Dengan menggunakan alat dan layanan AWS, Anda dapat memilih otomatisasi AWS mana yang akan diimplementasikan dan disesuaikan untuk persyaratan spesifik Anda. Contohnya, gunakan EC2 Image Builder untuk membangun, menguji, dan melakukan deployment mesin virtual dan citra kontainer untuk digunakan di AWS atau on-premise. Jika tindakan yang Anda inginkan tidak dapat dilakukan dengan layanan AWS atau Anda memerlukan tindakan lebih kompleks dengan sumber daya penyaring, maka otomatiskan operasi Anda menggunakan AWS CLI atau alat SDK AWS. AWS CLI memberikan kemampuan untuk mengotomatiskan seluruh proses pengontrolan dan pengelolaan layanan AWS lewat skrip tanpa menggunakan Konsol AWS. Pilih SDK AWS yang Anda sukai untuk berinteraksi dengan layanan AWS. Untuk contoh kode lainnya, lihat repositori contoh Kode SDK AWS.

Sumber daya

Dokumen terkait:

- Memodernisasikan operasi di AWS Cloud
- Layanan AWS untuk Otomatisasi
- Otomatisasi AWS Systems Manager
- Otomatisasi AWS untuk operasi dan administrasi SAP
- AWS Managed Services
- Layanan Profesional AWS
- Infrastruktur dan otomatisasi

Contoh terkait:

- Menginovasikan kembali operasi otomatis (Bagian I)

- Menginovasikan kembali operasi otomatis (Bagian II)
- Otomatisasi AWS untuk operasi dan administrasi SAP
- Otomatisasi IT dengan AWS Lambda
- Repositori Contoh Kode AWS
- Sampel AWS

Pelestarian Lingkungan

Pilar pelestarian lingkungan mencakup pemahaman tentang dampak layanan yang digunakan, menghitung dampak melalui seluruh siklus hidup beban kerja, dan menerapkan prinsip-prinsip desain dan praktik terbaik untuk mengurangi dampak-dampak ini saat membangun beban kerja cloud. Anda dapat menemukan panduan preskriptif tentang implementasi di Laporan resmi Pilar Pelestarian Lingkungan.

Area praktik terbaik

- Pemilihan wilayah
- Penyelarasan dengan permintaan
- Perangkat lunak dan arsitektur
- Data
- Perangkat keras dan layanan
- Proses dan budaya

Pemilihan wilayah

Pertanyaan

- SUS 1 Bagaimana cara memilih Wilayah untuk beban kerja Anda?

SUS 1 Bagaimana cara memilih Wilayah untuk beban kerja Anda?

Pilihan Wilayah untuk beban kerja Anda sangat memengaruhi KPI-nya, termasuk kinerja, biaya, dan jejak karbon. Untuk meningkatkan semua KPI ini secara efektif, sebaiknya pilih Wilayah untuk beban kerja Anda berdasarkan persyaratan bisnis dan tujuan pelestarian lingkungan Anda.

Praktik terbaik

- SUS01-BP01 Memilih Wilayah berdasarkan persyaratan bisnis dan tujuan keberlanjutan

SUS01-BP01 Memilih Wilayah berdasarkan persyaratan bisnis dan tujuan keberlanjutan

Pilih Wilayah untuk beban kerja Anda berdasarkan persyaratan bisnis dan tujuan keberlanjutan Anda untuk mengoptimalkan KPI, termasuk kinerja, biaya, dan jejak karbon.

Antipola umum:

- Anda memilih Wilayah beban kerja berdasarkan lokasi Anda sendiri.
- Anda menggabungkan semua sumber daya beban kerja ke dalam satu lokasi geografis.

Manfaat menjalankan praktik terbaik ini: Menempatkan beban kerja dekat dengan proyek-proyek energi terbarukan Amazon atau Wilayah dengan intensitas karbon terpublikasi yang rendah dapat membantu menurunkan jejak karbon beban kerja cloud.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

AWS Cloud adalah jaringan Wilayah dan titik kehadiran (PoP) yang terus-menerus berekspansi, dengan infrastruktur jaringan global yang menghubungkan semuanya. Pilihan Wilayah untuk beban kerja Anda sangat memengaruhi KPI-nya, termasuk kinerja, biaya, dan jejak karbon. Untuk meningkatkan semua KPI ini secara efektif, Anda harus memilih Wilayah untuk beban kerja Anda berdasarkan persyaratan bisnis dan tujuan keberlanjutan Anda.

Langkah implementasi

- Ikuti langkah-langkah ini untuk mengevaluasi dan merangkum Wilayah potensial untuk beban kerja Anda berdasarkan persyaratan bisnis Anda termasuk kepatuhan, fitur yang tersedia, biaya, dan latensi:
  - Konfirmasikan bahwa Wilayah-Wilayah tersebut mematuhi persyaratan peraturan setempat.
  - Gunakan Daftar Layanan Wilayah AWS untuk memeriksa apakah Wilayah memiliki layanan dan fitur yang Anda perlukan untuk menjalankan beban kerja Anda.
  - Hitung biaya beban kerja pada setiap Wilayah menggunakan AWS Kalkulator Harga.
  - Uji latensi jaringan antara lokasi pengguna akhir Anda dan setiap Wilayah AWS.

- Pilih Wilayah di dekat proyek-proyek energi terbarukan Amazon dan Wilayah dengan jaringan energi yang memiliki intensitas karbon terpublikasi yang lebih rendah daripada lokasi (atau Wilayah) lain.
  - Identifikasi pedoman keberlanjutan yang relevan untuk melacak dan membandingkan emisi karbon dari tahun ke tahun berdasarkan Protokol Gas Rumah Kaca (metode berbasis pasar dan berbasis lokasi).
  - Pilih wilayah berdasarkan metode yang Anda gunakan untuk melacak emisi karbon. Untuk detail selengkapnya tentang memilih Wilayah berdasarkan pedoman keberlanjutan, lihat Cara memilih Wilayah untuk beban kerja Anda berdasarkan tujuan keberlanjutan.

Sumber daya

Dokumen terkait:

- Memahami estimasi emisi karbon Anda
- Amazon di Seluruh Dunia
- Metodologi Energi Terbarukan
- Hal-Hal yang Perlu Dipertimbangkan saat Memilih Wilayah untuk Beban Kerja

Video terkait:

- Merancang secara berkelanjutan dan mengurangi jejak karbon AWS Anda

Penyelarasan dengan permintaan

Pertanyaan

- SUS 2 Bagaimana cara menyelaraskan sumber daya cloud dengan permintaan Anda?

SUS 2 Bagaimana cara menyelaraskan sumber daya cloud dengan permintaan Anda?

Cara pengguna dan aplikasi menggunakan beban kerja Anda dan sumber daya lainnya dapat membantu Anda mengidentifikasi peningkatan untuk memenuhi tujuan pelestarian lingkungan. Skalakan infrastruktur agar dapat terus sesuai dengan permintaan dan verifikasi bahwa Anda hanya menggunakan sumber daya minimum yang diperlukan untuk mendukung pengguna Anda. Selaraskan tingkat layanan dengan kebutuhan pelanggan. Posisikan sumber daya guna membatasi

jaringan yang diperlukan pengguna dan aplikasi untuk memakainya. Singkirkan aset yang tidak digunakan. Bekali anggota tim Anda dengan perangkat yang mendukung kebutuhan mereka dan meminimalkan dampak terhadap pelestarian lingkungan.

Praktik terbaik

- SUS02-BP01 Menskalakan infrastruktur beban kerja secara dinamis
- SUS02-BP02 Menyelaraskan SLA dengan tujuan keberlanjutan
- SUS02-BP03 Menghentikan pembuatan dan pemeliharaan aset yang tak terpakai
- SUS02-BP04 Mengoptimalkan penempatan geografis beban kerja berdasarkan persyaratan jaringannya
- SUS02-BP05 Mengoptimalkan sumber daya anggota tim untuk aktivitas yang dijalankan
- SUS02-BP06 Mengimplementasikan buffering atau throttling untuk meratakan kurva permintaan

SUS02-BP01 Menskalakan infrastruktur beban kerja secara dinamis

Gunakan elastisitas cloud dan skalakan infrastruktur Anda secara dinamis untuk menyesuaikan pasokan sumber daya cloud dengan permintaan dan menghindari kelebihan penyediaan kapasitas di beban kerja Anda.

Antipola umum:

- Anda tidak menskalakan infrastruktur Anda dengan beban pengguna.
- Anda secara manual menskalakan infrastruktur Anda sepanjang waktu.
- Anda membiarkan peningkatan kapasitas setelah peristiwa penskalaan, bukannya menurunkan kembali skala.

Manfaat menerapkan praktik terbaik ini: Mengonfigurasikan dan menguji elastisitas beban kerja akan membantu menyesuaikan pasokan sumber daya cloud dengan permintaan secara efisien dan menghindari kelebihan penyediaan kapasitas. Anda dapat memanfaatkan elastisitas di cloud untuk menskalakan kapasitas secara otomatis selama dan setelah lonjakan permintaan. Hal ini bertujuan untuk memastikan Anda hanya menggunakan jumlah sumber daya yang benar-benar diperlukan untuk memenuhi persyaratan bisnis Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Cloud menyediakan fleksibilitas untuk memperluas atau mengurangi sumber daya Anda secara dinamis melalui beragam mekanisme untuk memenuhi perubahan sesuai permintaan. Menyesuaikan pasokan dengan permintaan secara optimal akan memberikan dampak lingkungan terendah untuk beban kerja.

Permintaan dapat bersifat tetap atau bervariasi, sehingga akan memerlukan metrik dan otomatisasi untuk memastikan manajemen permintaan tersebut tidak menyulitkan. Aplikasi dapat diskalakan secara vertikal (naik atau turun) dengan mengubah ukuran instans, secara horizontal (ke dalam atau ke luar) dengan mengubah jumlah instans, atau kombinasi keduanya.

Anda dapat menggunakan sejumlah pendekatan yang berbeda untuk menyesuaikan pasokan sumber daya dengan permintaan.

- Pendekatan pelacakan target: Pantau metrik penskalaan Anda dan tingkatkan atau turunkan kapasitas secara otomatis sesuai kebutuhan.
- Penskalaan prediktif: Lakukan penskalaan dalam mengantisipasi tren harian dan mingguan.
- Pendekatan berbasis jadwal: Tetapkan jadwal penskalaan Anda sendiri sesuai dengan perubahan beban yang dapat diprediksi.
- Penskalaan layanan: Pilih layanan (seperti nirserver) yang diskalakan secara native berdasarkan desain atau sediakan penskalaan otomatis sebagai fitur.

Identifikasi periode penggunaan rendah atau nol dan skalakan sumber daya untuk menghapus kapasitas berlebih dan meningkatkan efisiensi.

Langkah implementasi

- Elastisitas menyesuaikan pasokan sumber daya yang Anda miliki dengan permintaan untuk sumber daya tersebut. Instans, kontainer, dan fungsi menyediakan mekanisme untuk elastisitas, baik dalam kombinasi dengan penskalaan otomatis maupun sebagai fitur layanan. AWS menyediakan serangkaian mekanisme penskalaan otomatis untuk memastikan beban kerja dapat diturunkan skalanya dengan cepat dan mudah selama periode beban pengguna yang rendah. Berikut beberapa contoh mekanisme penskalaan otomatis:

Auto scaling mechanism	Where to use
Amazon EC2 Auto Scaling	Gunakan untuk memastikan Anda memiliki jumlah instans Amazon EC2 yang tepat untuk menangani beban pengguna aplikasi Anda.
Application Auto Scaling	Gunakan untuk secara otomatis menskalakan sumber daya bagi layanan AWS masing-masing di luar Amazon EC2, seperti fungsi Lambda atau layanan Amazon Elastic Container Service (Amazon ECS).
Kubernetes Cluster Autoscaler	Gunakan untuk secara otomatis menskalakan klaster Kubernetes di AWS.

- Penskalaan sering dibahas terkait dengan layanan komputasi seperti instans Amazon EC2 atau fungsi AWS Lambda. Pertimbangkan konfigurasi layanan nonkomputasi seperti unit kapasitas baca dan tulis Amazon DynamoDB atau serpihan (shard) Amazon Kinesis Data Streams untuk disesuaikan dengan permintaan.
- Pastikan bahwa metrik untuk peningkatan atau penurunan skala telah divalidasi terhadap jenis beban kerja yang di-deploy. Jika Anda men-deploy aplikasi transkode video, 100% pemanfaatan CPU adalah hal normal dan tidak boleh menjadi metrik primer Anda. Anda dapat menggunakan metrik yang disesuaikan (seperti pemanfaatan memori) untuk kebijakan penskalaan jika diperlukan. Untuk memilih metrik yang tepat, pertimbangkan panduan berikut untuk Amazon EC2:
  - Metrik harus merupakan metrik pemanfaatan yang valid dan mendeskripsikan tingkat kesibukan suatu instans.
  - Nilai metrik harus meningkat atau menurun secara proporsional dengan jumlah instans dalam grup Auto Scaling.
- Gunakan penskalaan dinamis, bukan penskalaan manual untuk grup Auto Scaling Anda. Selain itu, sebaiknya gunakan kebijakan penskalaan pelacakan target dalam penskalaan dinamis Anda.
- Pastikan deployment beban kerja dapat menangani peristiwa penskalaan ke dalam dan ke luar. Buat skenario pengujian untuk peristiwa penskalaan ke dalam guna memastikan beban kerja berperilaku seperti yang diharapkan dan tidak memengaruhi pengalaman pengguna (seperti kehilangan sesi lekat (sticky session)). Anda dapat menggunakan Riwayat aktivitas untuk memverifikasi aktivitas penskalaan untuk grup Auto Scaling.

- Evaluasi beban kerja Anda untuk pola terprediksi dan secara proaktif skalakan saat Anda mengantisipasi perubahan terencana dan terprediksi dalam permintaan. Dengan penskalaan prediktif, Anda dapat meniadakan kebutuhan untuk menyediakan kapasitas secara berlebih. Untuk detail selengkapnya, lihat Penskalaan Prediktif dengan Amazon EC2 Auto Scaling.

Sumber daya

Dokumen terkait:

- Mulai Menggunakan Amazon EC2 Auto Scaling
- Penskalaan Prediktif untuk EC2, Didukung oleh Machine Learning
- Analisis perilaku pengguna menggunakan Amazon OpenSearch Service, Amazon Data Firehose, dan Kibana
- Apa yang dimaksud dengan Amazon CloudWatch?
- Memantau beban DB dengan Performance Insights di Amazon RDS
- Memperkenalkan Dukungan Native untuk Penskalaan Prediktif dengan Amazon EC2 Auto Scaling
- Memperkenalkan Karpenter - Kubernetes Cluster Autoscaler Sumber Terbuka yang Beperforma Tinggi
- Pendalaman tentang Amazon ECS Cluster Auto Scaling

Video terkait:

- Bangun lingkungan komputasi yang hemat biaya, energi, dan sumber daya
- Komputasi yang lebih baik, lebih cepat, dan lebih murah: Optimisasi biaya Amazon EC2 (CMP202-R1)

Contoh terkait:

- Lab: Contoh Grup Amazon EC2 Auto Scaling
- Lab: Memperkenalkan Penskalaan Otomatis dengan Karpenter

SUS02-BP02 Menyelaraskan SLA dengan tujuan keberlanjutan

Tinjau dan optimalkan perjanjian tingkat layanan (SLA) beban kerja berdasarkan tujuan keberlanjutan Anda untuk meminimalkan sumber daya yang diperlukan untuk mendukung beban kerja Anda sambil terus memenuhi kebutuhan bisnis.

Antipola umum:

- SLA beban kerja tidak diketahui atau ambigu.
- Anda menetapkan SLA hanya demi ketersediaan dan kinerja.
- Anda menggunakan pola desain yang sama (seperti arsitektur Multi-AZ) untuk semua beban kerja Anda.

Manfaat menjalankan praktik terbaik ini: Menyelaraskan SLA dengan tujuan keberlanjutan menghasilkan penggunaan sumber daya yang optimal sambil memenuhi kebutuhan bisnis.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

SLA menetapkan tingkat layanan yang diharapkan dari beban kerja cloud, seperti waktu respons, ketersediaan, dan retensi data. SLA memengaruhi arsitektur, penggunaan sumber daya, dan dampak lingkungan beban kerja cloud. Secara rutin, tinjau SLA dan buat pilihan kompromi yang secara signifikan mengurangi penggunaan sumber daya dengan penurunan yang dapat diterima dalam hal tingkat layanan.

Langkah implementasi

- Tetapkan atau desain ulang SLA yang mendukung tujuan keberlanjutan Anda sambil memenuhi persyaratan bisnis Anda, bukan melampauinya.
- Ambil pilihan kompromi yang secara signifikan mengurangi dampak keberlanjutan dengan penurunan yang dapat diterima dalam hal tingkat layanan.
  - Keberlanjutan dan keandalan: Beban kerja dengan ketersediaan tinggi cenderung mengonsumsi lebih banyak sumber daya.
  - Keberlanjutan dan kinerja: Penggunaan lebih banyak sumber daya untuk meningkatkan kinerja dapat mendatangkan dampak lingkungan yang lebih tinggi.
  - Keberlanjutan dan keamanan: Beban kerja yang aman secara berlebihan dapat memiliki dampak lingkungan yang lebih tinggi.

- Gunakan pola desain seperti layanan mikro di AWS yang mengutamakan fungsi-fungsi yang krusial untuk bisnis, dan izinkan tingkat layanan (seperti waktu respons atau tujuan waktu pemulihan) yang lebih rendah untuk fungsi-fungsi nonkritis.

Sumber daya

Dokumen terkait:

- Perjanjian Tingkat Layanan (SLA) AWS
- Pentingnya Perjanjian Tingkat Layanan untuk Penyedia SaaS

Video terkait:

- Menghadirkan arsitektur pelestarian lingkungan dengan performa tinggi
- Bangun lingkungan komputasi yang hemat biaya, energi, dan sumber daya

SUS02-BP03 Menghentikan pembuatan dan pemeliharaan aset yang tak terpakai

Nonaktifkan aset yang tak terpakai di beban kerja Anda untuk mengurangi jumlah sumber daya cloud yang diperlukan untuk mendukung permintaan Anda dan meminimalkan limbah.

Antipola umum:

- Anda tidak menganalisis aplikasi Anda untuk mengetahui aset yang redundan atau tidak diperlukan lagi.
- Anda tidak menyingkirkan aset yang redundan atau tidak diperlukan lagi.

Manfaat menjalankan praktik terbaik ini: Menyingkirkan aset yang tak terpakai membebaskan sumber daya dan meningkatkan efisiensi beban kerja secara keseluruhan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Aset yang tak terpakai mengonsumsi sumber daya cloud seperti ruang penyimpanan dan daya komputasi. Dengan mengidentifikasi dan mengeliminasi aset-aset ini, Anda dapat membebaskan berbagai sumber daya ini, sehingga arsitektur cloud akan lebih efisien. Lakukan analisis aset aplikasi secara teratur, yakni aset seperti laporan pra-kompilasi, set data, gambar statis, dan pola

akses aset untuk mengidentifikasi redundansi, pemanfaatan yang terlalu rendah, dan potensi target penonaktifan. Singkirkan aset redundan tersebut untuk mengurangi limbah sumber daya di beban kerja Anda.

Langkah implementasi

- Gunakan alat pemantauan untuk mengidentifikasi aset statis yang tidak diperlukan lagi.
- Sebelum menyingkirkan aset apa pun, evaluasi dampak penyingkirannya di arsitektur.
- Kembangkan rencana dan singkirkan aset yang tidak diperlukan lagi.
- Gabungkan aset tumpang tindih yang dihasilkan untuk menghindari redundansi pemrosesan.
- Perbarui aplikasi Anda hingga tidak lagi membuat dan menyimpan aset yang tidak diperlukan.
- Arahkan pihak ketiga untuk berhenti memproduksi dan menyimpan aset yang dikelola atas nama Anda yang tidak diperlukan lagi.
- Arahkan pihak ketiga untuk menggabungkan aset redundan yang dibuat atas nama Anda.
- Tinjau secara teratur beban kerja Anda untuk mengidentifikasi dan menyingkirkan aset yang tak terpakai.

Sumber daya

Dokumen terkait:

- Mengoptimalkan Infrastruktur AWS Anda untuk Pelestarian Lingkungan, Bagian II: Penyimpanan
- Bagaimana cara menghentikan sumber daya aktif yang tidak saya butuhkan lagi di Akun AWS saya?

Video terkait:

- Bagaimana cara memeriksa kemudian menyingkirkan sumber daya aktif yang tidak saya butuhkan lagi di Akun AWS saya?

SUS02-BP04 Mengoptimalkan penempatan geografis beban kerja berdasarkan persyaratan jaringannya

Pilih layanan dan lokasi cloud untuk beban kerja Anda yang mengurangi jarak yang harus ditempuh lalu lintas jaringan dan menurunkan total sumber daya jaringan yang diperlukan untuk mendukung beban kerja Anda.

Antipola umum:

- Anda memilih Wilayah beban kerja berdasarkan lokasi Anda sendiri.
- Anda menggabungkan semua sumber daya beban kerja ke dalam satu lokasi geografis.
- Semua lalu lintas mengalir melalui pusat data Anda.

Manfaat menjalankan praktik terbaik ini: Menempatkan beban kerja dekat dengan penggunanya akan menghasilkan latensi terendah sambil mengurangi pergerakan data di seluruh jaringan dan mengurangi dampak lingkungan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Infrastruktur AWS Cloud dibangun di seputar opsi lokasi seperti Wilayah, Zona Ketersediaan, grup penempatan, dan lokasi edge seperti AWS Outposts dan Zona Lokal AWS. Opsi lokasi ini bertanggung jawab untuk memelihara konektivitas antara komponen aplikasi, layanan cloud, jaringan edge, dan pusat data on-premise.

Analisis pola akses jaringan di beban kerja Anda untuk mengidentifikasi cara menggunakan opsi lokasi cloud ini dan mengurangi jarak yang harus ditempuh lalu lintas jaringan.

Langkah implementasi

- Analisis pola akses jaringan di beban kerja Anda untuk mengidentifikasi cara pengguna menggunakan aplikasi Anda.
  - Gunakan alat pemantauan, seperti Amazon CloudWatch dan AWS CloudTrail, untuk mengumpulkan data tentang aktivitas jaringan.
  - Analisis data untuk mengidentifikasi pola akses jaringan.
- Pilih Wilayah untuk deployment beban kerja Anda berdasarkan elemen utama berikut:
  - Tujuan Pelestarian Lingkungan Anda: seperti dijelaskan dalam Pemilihan wilayah.
  - Lokasi data Anda: Untuk aplikasi dengan banyak data (seperti big data dan machine learning), kode aplikasi harus dijalankan sedekat mungkin dengan data.
  - Lokasi pengguna Anda: Untuk aplikasi yang ditampilkan kepada pengguna, pilih Wilayah yang dekat dengan pengguna beban kerja Anda.
  - Kendala lainnya: Pertimbangkan kendala seperti biaya dan kepatuhan sebagaimana dijelaskan dalam Hal-Hal yang Perlu Dipertimbangkan saat Memilih Wilayah untuk Beban Kerja.

- Gunakan caching lokal atau Solusi Caching AWS untuk aset yang sering digunakan guna meningkatkan performa, mengurangi pergerakan data, dan mengurangi dampak lingkungan.

Layanan	Kapan harus digunakan
Amazon CloudFront	Gunakan untuk meng-cache konten statis seperti gambar, skrip, dan video, serta konten dinamis seperti respons API atau aplikasi web.
Amazon ElastiCache	Gunakan untuk meng-cache konten bagi aplikasi web.
DynamoDB Accelerator	Gunakan untuk menambahkan percepatan dalam memori ke tabel DynamoDB Anda.

- Gunakan layanan yang dapat membantu Anda menjalankan kode lebih dekat dengan pengguna beban kerja Anda:

Layanan	Kapan harus digunakan
Lambda@Edge	Gunakan untuk operasi dengan banyak komputasi yang dimulai saat objek tidak ada dalam cache.
Fungsi Amazon CloudFront	Gunakan untuk kasus penggunaan sederhana seperti permintaan HTTP atau manipulas i respons yang dapat dimulai oleh fungsi dengan masa pakai singkat.
AWS IoT Greengrass	Gunakan untuk menjalankan komputasi lokal, olahpesan, dan caching data untuk perangkat yang terhubung.

- Gunakan pooling koneksi untuk mengizinkan penggunaan ulang koneksi dan mengurangi sumber daya yang diperlukan.
- Gunakan penyimpanan data terdistribusi yang tidak mengandalkan koneksi persisten dan pembaruan sinkron untuk mendapatkan konsistensi guna melayani populasi wilayah.

- Ganti kapasitas jaringan statis yang disediakan di awal dengan kapasitas dinamis bersama, dan bagikan dampak pelestarian lingkungan kapasitas jaringan kepada pelanggan lain.

Sumber daya

Dokumen terkait:

- Mengoptimalkan Infrastruktur AWS untuk Pelestarian Lingkungan, Bagian III: Jaringan
- Dokumentasi Amazon ElastiCache
- Apa itu Amazon CloudFront?
- Fitur Utama Amazon CloudFront

Video terkait:

- Menjelaskan transfer data di AWS
- Menskalakan kinerja jaringan pada instans Amazon EC2 generasi berikutnya

Contoh terkait:

- Lokakarya Jaringan AWS
- Arsitektur untuk keberlanjutan - Meminimalkan pergerakan data lintas jaringan

SUS02-BP05 Mengoptimalkan sumber daya anggota tim untuk aktivitas yang dijalankan

Optimalkan sumber daya yang disediakan bagi anggota tim untuk meminimalkan dampak pelestarian lingkungan sambil mendukung kebutuhan mereka.

Antipola umum:

- Anda mengabaikan dampak dari perangkat yang digunakan oleh anggota tim Anda pada efisiensi aplikasi cloud secara keseluruhan.
- Anda secara manual mengelola dan memperbarui sumber daya yang digunakan oleh anggota tim.

Manfaat menjalankan praktik terbaik ini: Mengoptimalkan sumber daya anggota tim meningkatkan efisiensi keseluruhan aplikasi yang diaktifkan oleh cloud.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Pahami sumber daya yang digunakan anggota tim Anda untuk mengonsumsi layanan Anda, ekspektasi siklus hidup mereka, dan dampak finansial serta dampak pada pelestarian lingkungan. Implementasikan strategi untuk mengoptimalkan berbagai sumber daya ini. Sebagai contoh, lakukan operasi yang kompleks, seperti rendering dan kompilasi, pada infrastruktur yang dapat diskalakan dan sangat banyak digunakan, bukan di sistem pengguna tunggal berdaya tinggi namun jarang digunakan.

Langkah implementasi

- Sediakan workstation dan perangkat lainnya untuk menyesuaikan dengan cara penggunaannya.
- Gunakan streaming aplikasi dan desktop virtual untuk membatasi persyaratan perangkat dan pemutakhiran.
- Alihkan tugas yang sarat dengan memori atau prosesor ke cloud untuk menggunakan elastisitasnya.
- Evaluasi dampak proses dan sistem atas siklus hidup perangkat, dan pilih solusi yang meminimalkan persyaratan untuk penggantian perangkat sekaligus memenuhi persyaratan bisnis.
- Implementasikan manajemen jarak jauh untuk perangkat guna mengurangi perjalanan bisnis yang diperlukan.
  - AWS Systems ManagerFleet Manager adalah pengalaman antarmuka (UI) pengguna terpadu yang membantu Anda mengelola simpul yang beroperasi di AWS atau on-premise dari jarak jauh.

Sumber daya

Dokumen terkait:

- Apa itu Amazon WorkSpaces?
- Pengoptimal Biaya untuk Amazon WorkSpaces
- Dokumentasi Amazon AppStream 2.0
- NICE DCV

Video terkait:

- Mengelola biaya untuk Amazon WorkSpaces di AWS

SUS02-BP06 Mengimplementasikan buffering atau throttling untuk meratakan kurva permintaan

Buffering dan throttling meratakan kurva permintaan dan mengurangi kapasitas tersedia yang diperlukan untuk beban kerja Anda.

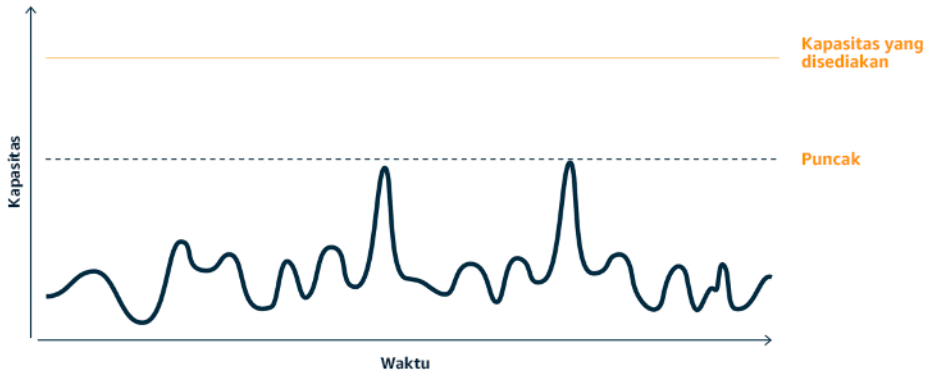
Antipola umum:

- Anda memproses permintaan klien dengan segera walaupun tidak diperlukan.
- Anda tidak menganalisis persyaratan untuk permintaan klien.

Manfaat menjalankan praktik terbaik ini: Meratakan kurva permintaan mengurangi kapasitas tersedia yang diperlukan untuk beban kerja. Mengurangi kapasitas tersedia artinya konsumsi energi berkurang dan dampak pada lingkungan juga berkurang.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

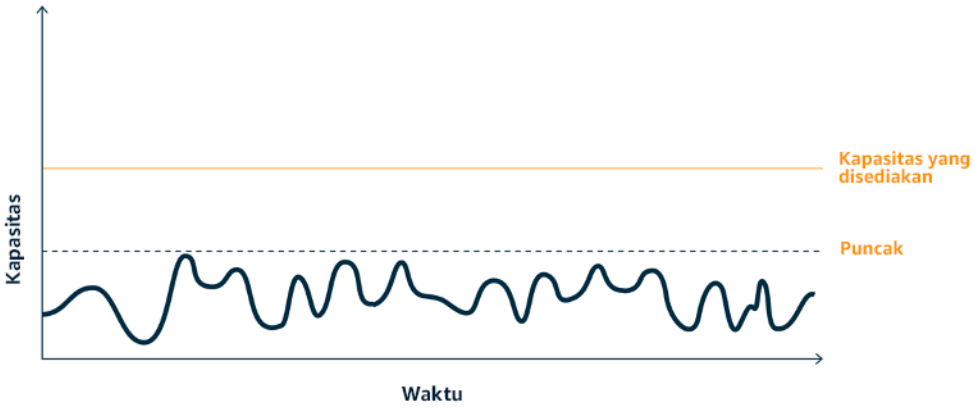
Meratakan kurva permintaan beban kerja dapat membantu Anda mengurangi kapasitas tersedia untuk beban kerja dan mengurangi dampaknya pada lingkungan. Asumsikan beban kerja dengan kurva permintaan yang ditunjukkan pada gambar di bawah ini. Beban kerja ini memiliki dua puncak. Untuk menangani puncak-puncak ini, kapasitas sumber daya sebagaimana ditunjukkan oleh garis oranye disediakan. Sumber daya dan energi yang digunakan untuk beban kerja ini tidak diindikasikan oleh area di bawah kurva permintaan, tetapi oleh area di bawah garis kapasitas tersedia, karena kapasitas tersedia diperlukan untuk menangani kedua puncak ini.



Kurva permintaan dengan dua puncak terpisah yang memerlukan penyediaan kapasitas tinggi.

Anda dapat menggunakan buffering atau throttling untuk memodifikasi kurva permintaan dan meratakan puncak, yang berarti konsumsi lebih sedikit energi dan penyediaan kapasitas lebih

rendah. Implementasikan throttling ketika klien Anda dapat mencoba ulang. Implementasikan buffering untuk menyimpan permintaan dan menunda pemrosesan ke lain waktu.



Efek throttling pada kurva permintaan dan kapasitas tersedia.

Langkah implementasi

- Analisis permintaan klien untuk menentukan cara merespons permintaan. Pertanyaan yang harus dipertimbangkan antara lain:
  - Dapatkah permintaan ini diproses secara asinkron?
  - Apakah klien memiliki kemampuan untuk mencoba ulang?
- Jika klien memiliki kemampuan untuk coba ulang, maka Anda dapat mengimplementasikan throttling, yang memberi tahu sumber bahwa jika sumber tidak dapat melayani permintaan pada saat ini maka sumber harus mencoba lagi nanti.
  - Anda dapat menggunakan Amazon API Gateway untuk mengimplementasikan throttling.
- Untuk klien yang tidak dapat mencoba ulang, buffer harus diimplementasikan untuk meratakan kurva permintaan. Buffer menunda pemrosesan permintaan, sehingga aplikasi yang dijalankan pada tingkat yang berlainan dapat berkomunikasi secara efektif. Pendekatan berbasis buffer menggunakan antrean atau aliran untuk menerima pesan dari produsen. Pesan dibaca oleh konsumen dan diproses, sehingga pesan dapat dijalankan dengan tingkat yang memenuhi persyaratan bisnis konsumen.
  - Amazon Simple Queue Service (Amazon SQS) merupakan layanan terkelola yang memberikan antrean yang memungkinkan satu konsumen membaca pesan secara individu.
  - Amazon Kinesis memberikan aliran yang memungkinkan banyak konsumen membaca pesan yang sama.

- Analisis permintaan secara keseluruhan, tingkat perubahan, dan waktu respons yang diperlukan untuk ukuran throttle atau buffer yang tepat.

Sumber daya

Dokumen terkait:

- Mulai menggunakan Amazon SQS
- Integrasi Aplikasi Menggunakan Antrean dan Pesan

Video terkait:

- Memilih Layanan Olah Pesan yang Tepat untuk Aplikasi Terdistribusi Anda

Perangkat lunak dan arsitektur

Pertanyaan

- SUS 3 Bagaimana cara memanfaatkan pola arsitektur dan perangkat lunak untuk mendukung tujuan berkelanjutan Anda?

SUS 3 Bagaimana cara memanfaatkan pola arsitektur dan perangkat lunak untuk mendukung tujuan berkelanjutan Anda?

Implementasikan pola untuk melancarkan beban dan mempertahankan penggunaan yang tinggi dan konsisten atas sumber daya yang di-deploy guna meminimalkan sumber daya yang dipakai. Komponen dapat menjadi tidak aktif akibat kurangnya pemakaian, karena adanya perubahan perilaku pengguna dari waktu ke waktu. Revisi pola dan arsitektur untuk menggabungkan komponen dengan pemanfaatan rendah guna meningkatkan pemanfaatan secara keseluruhan. Pensiunkan komponen yang tidak lagi diperlukan. Pahami performa komponen beban kerja Anda, dan optimalkan komponen yang memakai sumber daya terbanyak. Ketahui perangkat yang digunakan pelanggan untuk mengakses layanan Anda, dan implementasikan pola untuk meminimalkan kebutuhan pemutakhiran perangkat.

Praktik terbaik

- SUS03-BP01 Optimalkan perangkat lunak dan arsitektur untuk tugas yang asinkron dan terjadwal

- SUS03-BP02 Menyingkirkan atau memfaktor ulang komponen beban kerja yang jarang atau tidak pernah digunakan
- SUS03-BP03 Mengoptimalkan area kode yang memakai waktu atau sumber daya paling banyak
- SUS03-BP04 Mengoptimalkan dampak pada perangkat dan perlengkapan
- SUS03-BP05 Menggunakan pola perangkat lunak dan arsitektur yang paling mendukung pola akses dan penyimpanan data

SUS03-BP01 Optimalkan perangkat lunak dan arsitektur untuk tugas yang asinkron dan terjadwal

Gunakan pola arsitektur dan perangkat lunak yang efisien seperti berbasis antrean untuk mempertahankan pemanfaatan sumber daya ter-deploy yang terus-menerus tinggi.

Antipola umum:

- Anda melakukan pengadaan sumber daya secara berlebihan di beban kerja cloud Anda untuk memenuhi lonjakan permintaan tidak terduga.
- Arsitektur Anda tidak memisahkan pengirim dan penerima pesan asinkron dengan komponen pesan.

Manfaat menjalankan praktik terbaik ini:

- Pola arsitektur dan perangkat lunak yang efisien meminimalkan sumber daya tidak terpakai di dalam beban kerja Anda dan meningkatkan keseluruhan efisiensi.
- Anda dapat menskalakan pemrosesan tanpa terikat penerimaan pesan asinkron.
- Melalui komponen pesan, Anda memiliki persyaratan ketersediaan yang longgar yang dapat Anda penuhi dengan sumber daya yang lebih sedikit.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Gunakan pola arsitektur yang efisien seperti event-driven architecture (arsitektur berbasis peristiwa) yang menghasilkan bahkan pemanfaatan komponen dan meminimalkan pengadaan yang berlebihan dalam beban kerja Anda. Menggunakan pola arsitektur yang efisien meminimalkan sumber daya tidak aktif akibat kurangnya pemakaian yang disebabkan oleh perubahan permintaan seiring berjalannya waktu.

Pahami persyaratan komponen beban kerja Anda dan adopsi pola arsitektur yang meningkatkan keseluruhan pemanfaatan sumber daya. Pensiunkan komponen yang sudah tidak diperlukan.

Langkah implementasi

- Analisis permintaan untuk beban kerja Anda guna menentukan cara meresponsnya.
- Untuk permintaan atau tugas yang tidak memerlukan respons sinkron, gunakan arsitektur berbasis antrean dan pekerja penskalaan otomatis untuk memaksimalkan pemanfaatan. Berikut ini adalah beberapa contoh kapan Anda mungkin perlu mempertimbangkan arsitektur berbasis antrean:

Queuing mechanism	Description
Antrean tugas AWS Batch	Tugas AWS Batch dikirimkan ke antrean tugas dan menunggu untuk dijadwalkan berjalan di lingkungan komputasi.
Amazon Simple Queue Service dan Instans Spot Amazon EC2	Memasangkan Amazon SQS dan Instans Spot untuk membangun arsitektur yang efisien dan tahan terhadap kesalahan.

- Untuk permintaan atau tugas yang dapat diproses kapan saja, gunakan mekanisme penjadwalan untuk memproses tugas dalam batch untuk mendapatkan efisiensi yang lebih tinggi. Berikut beberapa contoh mekanisme penjadwalan di AWS:

Scheduling mechanism	Description
Penjadwal Amazon EventBridge	Sebuah kemampuan dari Amazon EventBridge yang memungkinkan Anda untuk membuat, menjalankan, dan mengelola tugas terjadwal pada skala besar.
Jadwal berbasis waktu AWS Glue	Tentukan jadwal berbasis waktu untuk perayap dan tugas Anda di AWS Glue.
Tugas terjadwal Amazon Elastic Container Service (Amazon ECS)	Amazon ECS mendukung pembuatan tugas terjadwal. Tugas terjadwal menggunakan aturan Amazon EventBridge untuk menjalank

Scheduling mechanism	Description
	an tugas baik secara terjadwal atau dalam rangka merespons peristiwa EventBridge.
Penjadwal Instans	Konfigurasikan jadwal mulai dan berhenti untuk Instans Amazon EC2 dan Amazon Relational Database Service Anda.

- Jika Anda menggunakan mekanisme polling dan webhook dalam arsitektur Anda, gantilah dengan peristiwa. Gunakan arsitektur berbasis peristiwa untuk membangun beban kerja yang sangat efisien.
- Manfaatkan nirserver di AWS untuk menyingkirkan infrastruktur yang disediakan secara berlebihan.
- Sesuaikan ukuran setiap komponen arsitektur Anda untuk menghindari sumber daya yang tidak aktif karena menunggu input.

Sumber daya

Dokumen terkait:

- Apa itu Amazon Simple Queue Service?
- Apa itu Amazon MQ?
- Menskalakan berdasarkan Amazon SQS
- Apa itu AWS Step Functions?
- Apa itu AWS Lambda?
- Menggunakan AWS Lambda dengan Amazon SQS
- Apa itu Amazon EventBridge?

Video terkait:

- Beralih ke arsitektur berbasis peristiwa

SUS03-BP02 Menyingkirkan atau memfaktor ulang komponen beban kerja yang jarang atau tidak pernah digunakan

Singkirkan komponen yang tidak digunakan dan sudah tidak diperlukan, dan faktorkan ulang komponen dengan sedikit pemanfaatan, untuk meminimalkan limbah di beban kerja Anda.

Antipola umum:

- Anda tidak secara teratur memeriksa tingkat penggunaan masing-masing komponen beban kerja Anda.
- Anda tidak memeriksa dan menganalisis rekomendasi dari alat penyesuaian ukuran AWS seperti AWS Compute Optimizer.

Manfaat menjalankan praktik terbaik ini: Menyingkirkan komponen yang tidak digunakan akan meminimalkan limbah dan meningkatkan efisiensi beban kerja cloud secara keseluruhan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Tinjau beban kerja Anda untuk mengidentifikasi komponen yang pasif atau tidak digunakan. Tindakan ini adalah proses peningkatan yang berulang, yang dapat dipicu oleh perubahan dalam permintaan atau rilis layanan cloud baru. Contohnya, penurunan signifikan dalam waktu pelaksanaan fungsi AWS Lambda dapat menjadi indikator dibutuhkannya pengurangan ukuran memori. Selain itu, ketika AWS merilis layanan dan fitur baru, arsitektur dan layanan optimal untuk beban kerja Anda mungkin berubah.

Terus pantau aktivitas beban kerja Anda dan cari peluang untuk meningkatkan tingkat pemanfaatan masing-masing komponen. Dengan menyingkirkan komponen yang pasif dan melakukan aktivitas penyesuaian ukuran, Anda memenuhi persyaratan bisnis dengan sesedikit mungkin sumber daya cloud.

Langkah implementasi

- Pantau dan tangkap metrik pemanfaatan untuk komponen penting beban kerja Anda (seperti pemanfaatan CPU, pemanfaatan memori, atau throughput jaringan di metrik Amazon CloudWatch).
- Untuk beban kerja yang stabil, periksa AWS alat penyesuaian ukuran seperti AWS Compute Optimizer secara teratur untuk mengidentifikasi komponen yang pasif, tidak digunakan, atau kurang dimanfaatkan.
- Untuk beban kerja sementara, evaluasi metrik pemanfaatan untuk mengidentifikasi komponen yang pasif, tidak digunakan, atau kurang dimanfaatkan.
- Pensiunkan komponen dan aset terkait (seperti gambar Amazon ECR) yang tidak diperlukan lagi.
- Faktor ulang atau gabungkan komponen yang kurang dimanfaatkan dengan sumber daya lain untuk meningkatkan efisiensi pemanfaatan. Contohnya, Anda dapat menyediakan beberapa basis

data kecil sebagai satu instans basis data Amazon RDS dan bukannya menjalankan basis data di instans individu yang kurang dimanfaatkan.

- Pahami sumber daya yang disediakan oleh beban kerja Anda untuk menyelesaikan unit kerja.

Sumber daya

Dokumen terkait:

- AWS Trusted Advisor
- Apa itu Amazon CloudWatch?
- Pembersihan Otomatis Gambar yang Tidak Digunakan di Amazon ECR

Contoh terkait:

- Well-Architected Lab - Penyesuaian Ukuran dengan AWS Compute Optimizer
- Well-Architected Lab - Optimisasi Pola Perangkat Keras dan Pengamatan KPI Pelestarian Lingkungan

SUS03-BP03 Mengoptimalkan area kode yang memakai waktu atau sumber daya paling banyak

Optimalkan kode Anda yang dijalankan di dalam berbagai macam komponen arsitektur Anda untuk meminimalkan penggunaan sumber daya sambil memaksimalkan performa.

Antipola umum:

- Anda mengabaikan optimisasi kode Anda untuk penggunaan sumber daya.
- Anda biasanya merespons masalah performa dengan meningkatkan sumber daya.
- Proses pengembangan dan peninjauan kode Anda tidak melacak perubahan performa.

Manfaat menjalankan praktik terbaik ini: Menggunakan kode yang efisien akan meminimalkan penggunaan sumber daya dan meningkatkan performa.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Setiap area fungsional harus diperiksa, termasuk kode untuk aplikasi dengan arsitektur cloud, untuk mengoptimalkan penggunaan sumber dayanya dan performanya. Terus pantau performa beban kerja

Anda di lingkungan pembangunan dan produksi dan identifikasi peluang untuk meningkatkan snippet kode yang memiliki penggunaan sumber daya sangat tinggi. Adopsi proses peninjauan secara teratur untuk mengidentifikasi bug atau antipola di dalam kode Anda yang menggunakan sumber daya secara tidak efisien. Manfaatkan algoritme sederhana dan efisien yang memberikan hasil yang sama untuk kasus penggunaan Anda.

Langkah implementasi

- Saat mengembangkan beban kerja Anda, adopsi proses peninjauan kode otomatis untuk meningkatkan kualitas dan mengidentifikasi bug dan antipola.
  - Otomatiskan peninjauan kode dengan Amazon CodeGuru Reviewer
  - Mendeteksi bug konkurensi dengan Amazon CodeGuru
  - Meningkatkan kualitas kode untuk aplikasi Python menggunakan Amazon CodeGuru
- Saat Anda menjalankan beban kerja Anda, pantau sumber daya untuk mengidentifikasi komponen dengan persyaratan sumber daya tinggi per unit kerja sebagai target untuk peninjauan kode.
- Untuk peninjauan kode, gunakan profiler kode untuk mengidentifikasi area kode yang menggunakan waktu atau sumber daya paling banyak sebagai target untuk dioptimalkan.
  - Mengurangi jejak karbon organisasi Anda dengan Amazon CodeGuru Profiler
  - Memahami penggunaan memori di aplikasi Java Anda dengan Amazon CodeGuru Profiler
  - Meningkatkan pengalaman pelanggan dan mengurangi biaya dengan Amazon CodeGuru Profiler
- Gunakan sistem operasi dan bahasa pemrograman paling efisien untuk beban kerja. Untuk informasi mendetail tentang bahasa pemrograman hemat energi (termasuk Rust), lihat Pelestarian lingkungan dengan Rust.
- Ganti algoritme yang banyak memerlukan komputasi dengan versi yang lebih sederhana dan lebih efisien, yang akan memberikan hasil yang sama.
- Singkirkan kode yang tidak perlu seperti penyortiran dan pemformatan.

Sumber daya

Dokumen terkait:

- Apa itu Amazon CodeGuru Profiler?
- Instans FPGA
- SDK AWS di Alat-Alat untuk Membangun di AWS

Video terkait:

- Tingkatkan Efisiensi Kode Menggunakan Amazon CodeGuru Profiler
- Otomatiskan Peninjauan Kode dan Rekomendasi Performa Aplikasi dengan Amazon CodeGuru

SUS03-BP04 Mengoptimalkan dampak pada perangkat dan perlengkapan

Pahami perangkat dan perlengkapan yang digunakan dalam arsitektur Anda dan gunakan strategi untuk mengurangi penggunaannya. Tindakan ini dapat meminimalkan dampak beban kerja cloud Anda pada lingkungan secara keseluruhan.

Antipola umum:

- Anda mengabaikan dampak dari perangkat yang digunakan oleh pelanggan Anda pada lingkungan.
- Anda secara manual mengelola dan memperbarui sumber daya yang digunakan oleh pelanggan.

Manfaat menjalankan praktik terbaik ini: Mengimplementasikan fitur dan pola perangkat lunak yang dioptimalkan untuk perangkat pelanggan dapat mengurangi dampak beban kerja cloud pada lingkungan secara keseluruhan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Mengimplementasikan fitur dan pola perangkat lunak yang dioptimalkan untuk perangkat pelanggan dapat mengurangi dampak pada lingkungan dengan beberapa cara:

- Mengimplementasikan fitur baru yang kompatibel dengan versi lama dapat mengurangi jumlah penggantian perangkat keras.
- Mengoptimalkan aplikasi untuk beroperasi secara efisien di perangkat dapat membantu mengurangi pemakaian energinya dan memperpanjang masa pakai baterainya (jika bertenaga baterai).
- Mengoptimalkan aplikasi untuk perangkat dapat juga mengurangi transfer data lewat jaringan.

Pahami perangkat dan perlengkapan yang digunakan dalam arsitektur Anda, ekspektasi siklus hidupnya, dan dampak dari penggantian komponen-komponen tersebut. Implementasikan fitur dan

pola perangkat lunak yang dapat membantu meminimalkan pemakaian energi perangkat, keharusan pelanggan untuk mengganti perangkat dan juga melakukan pemutakhiran perangkat secara manual.

Langkah implementasi

- Inventarisasikan perangkat yang digunakan dalam arsitektur Anda. Perangkat dapat berupa perangkat seluler, tablet, perangkat IOT, lampu pintar, atau bahkan perangkat pintar dalam pabrik.
- Optimalkan aplikasi yang beroperasi pada perangkat:
  - Gunakan strategi seperti menjalankan tugas di latar belakang untuk mengurangi pemakaian energinya.
  - Perhitungkan bandwidth jaringan dan latensi saat membangun payload, dan implementasikan kemampuan yang membantu aplikasi bekerja dengan baik pada tautan yang memiliki bandwidth rendah dan latensi tinggi.
  - Ubah format payload dan file ke format optimal yang diperlukan oleh perangkat. Contohnya, Anda dapat menggunakan Amazon Elastic Transcoder atau AWS Elemental MediaConvert untuk mengubah format file media digital besar dan berkualitas tinggi ke format yang dapat diputar pengguna di perangkat seluler, tablet, browser web, dan televisi yang terhubung.
  - Lakukan aktivitas yang membutuhkan banyak komputasi di sisi server (seperti render gambar), atau gunakan streaming aplikasi untuk meningkatkan pengalaman pengguna pada perangkat yang lebih lama.
  - Segmentasikan dan beri nomor halaman pada output, terutama untuk sesi interaktif, guna mengelola payload dan membatasi persyaratan penyimpanan lokal.
- Gunakan mekanisme otomatis lewat udara (OTA) untuk melakukan deployment pembaruan ke satu atau lebih perangkat.
  - Anda dapat menggunakan pipeline CI/CD untuk memperbarui aplikasi seluler.
  - Anda dapat menggunakan AWS IoT Device Management untuk mengelola perangkat terhubung dalam skala besar dari jarak jauh.
- Untuk menguji fitur baru dan pembaruan, gunakan device farm terkelola dengan set perangkat keras representatif dan ulang pengembangan untuk memaksimalkan perangkat yang didukung. Untuk detail selengkapnya, lihat SUS06-BP04 Menggunakan device farm terkelola untuk pengujian.

Sumber daya

Dokumen terkait:

- Apa itu AWS Device Farm?
- Dokumentasi Amazon AppStream 2.0
- NICE DCV
- Tutorial OTA untuk memperbarui firmware di perangkat yang menjalankan FreeRTOS

Video terkait:

- Pengantar AWS Device Farm

SUS03-BP05 Menggunakan pola perangkat lunak dan arsitektur yang paling mendukung pola akses dan penyimpanan data

Pahami bagaimana data digunakan di dalam beban kerja Anda, dipakai oleh pengguna Anda, ditransfer, dan disimpan. Gunakan pola perangkat lunak dan arsitektur yang paling mendukung akses dan penyimpanan data untuk meminimalkan sumber daya komputasi, jaringan, dan penyimpanan yang diperlukan untuk mendukung beban kerja.

Antipola umum:

- Anda berasumsi bahwa semua beban kerja memiliki pola penyimpanan dan akses data yang serupa.
- Anda hanya menggunakan satu tingkat penyimpanan, dengan anggapan semua beban kerja masuk dalam tingkat tersebut.
- Anda berasumsi bahwa pola akses data tidak akan berubah.
- Arsitektur Anda mendukung potensi lonjakan akses data yang tinggi, yang mengakibatkan sumber daya tetap tidak aktif dalam sebagian besar waktu.

Manfaat menjalankan praktik terbaik ini: Memilih dan mengoptimalkan arsitektur Anda berdasarkan akses data dan pola penyimpanan akan membantu mengurangi kompleksitas pengembangan dan meningkatkan pemanfaatan secara keseluruhan. Memahami kapan harus menggunakan tabel global, partisi data, dan caching akan membantu Anda mengurangi biaya operasional dan menskalakan sesuai kebutuhan beban kerja Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Gunakan pola arsitektur dan perangkat lunak yang paling sesuai dengan karakteristik data dan pola akses Anda. Contohnya, gunakan arsitektur data modern di AWS yang memungkinkan Anda menggunakan layanan yang dibuat khusus dan dioptimalkan untuk kasus penggunaan analitik unik Anda. Pola arsitektur ini memungkinkan pemrosesan data yang efisien dan mengurangi penggunaan sumber daya.

Langkah implementasi

- Analisis karakteristik data dan pola akses Anda untuk mengidentifikasi konfigurasi yang benar untuk sumber daya cloud Anda. Karakteristik utama yang perlu dipertimbangkan antara lain:
  - Jenis data: terstruktur, semi-terstruktur, tidak terstruktur
  - Pertumbuhan data: dibatasi, tidak dibatasi
  - Ketahanan data: persisten, sementara, transien
  - Pola akses: baca atau tulis, frekuensi pembaruan, berfluktuasi, atau konsisten
- Gunakan pola arsitektur yang paling mendukung pola akses dan penyimpanan data.
  - Mari Merancang! Arsitektur data modern
  - Basis data di AWS: Alat yang Tepat untuk Tugas yang Tepat
- Gunakan teknologi yang berfungsi secara native dengan data terkompresi.
- Gunakan layanan analitik yang dibuat khusus untuk pemrosesan data di arsitektur Anda.
- Gunakan mesin basis data yang paling mendukung pola kueri dominan Anda. Kelola indeks basis data Anda untuk memastikan pelaksanaan kueri yang efisien. Untuk detail selengkapnya, lihat Basis Data AWS.
- Pilih protokol jaringan yang mengurangi jumlah kapasitas jaringan yang dipakai di arsitektur Anda.

Sumber daya

Dokumen terkait:

- Format file Dukungan Kompresi Athena
- MENYALIN dari format data kolom dengan Amazon Redshift
- Mengubah Format Catatan Input Anda di Firehose
- Opsi Format untuk Input dan Output ETL di AWS Glue
- Meningkatkan performa kueri di Amazon Athena dengan Mengubah ke Format Kolom

- Memuat file data terkompresi dari Amazon S3 dengan Amazon Redshift
- Memantau beban DB dengan Wawasan Performa di Amazon Aurora
- Memantau beban DB dengan Wawasan Performa di Amazon RDS
- Kelas penyimpanan Intelligent-Tiering Amazon S3

Video terkait:

- Membangun arsitektur data modern di AWS

Data

Pertanyaan

- SUS 4 Bagaimana cara memanfaatkan kebijakan dan pola manajemen data untuk mendukung tujuan pelestarian lingkungan Anda?

SUS 4 Bagaimana cara memanfaatkan kebijakan dan pola manajemen data untuk mendukung tujuan pelestarian lingkungan Anda?

Implementasikan praktik manajemen data untuk mengurangi penyimpanan yang diberikan dan diperlukan untuk mendukung beban kerja Anda, serta sumber daya yang diperlukan untuk menggunakannya. Pahami data Anda, dan gunakan konfigurasi dan teknologi penyimpanan penggunaan yang paling efektif untuk mendukung nilai bisnis data dan cara data digunakan. Buat siklus hidup data di penyimpanan yang lebih efisien dan memiliki kinerja lebih rendah ketika persyaratan berkurang, dan hapus data yang tidak lagi diperlukan.

Praktik terbaik

- SUS04-BP01 Mengimplementasikan kebijakan klasifikasi data
- SUS04-BP02 Menggunakan teknologi yang mendukung pola akses dan penyimpanan data
- SUS04-BP03 Menggunakan kebijakan untuk mengelola siklus hidup set data Anda
- SUS04-BP04 Menggunakan elastisitas dan otomatisasi untuk memperluas sistem file atau penyimpanan blok
- SUS04-BP05 Menyingkirkan data yang tidak diperlukan atau redundan
- SUS04-BP06 Menggunakan sistem file atau penyimpanan bersama untuk mengakses data umum
- SUS04-BP07 Meminimalkan perpindahan data di jaringan

- SUS04-BP08 Hanya mencadangkan data saat sulit untuk dibuat ulang

SUS04-BP01 Mengimplementasikan kebijakan klasifikasi data

Kelompokkan data untuk memahami tingkat kekritisannya terhadap hasil bisnis dan pilih tingkat penyimpanan hemat energi yang tepat untuk menyimpan data.

Antipola umum:

- Anda tidak mengidentifikasi aset data dengan karakteristik serupa (seperti sensitivitas, kekritisan bisnis, atau persyaratan peraturan) yang diproses atau disimpan.
- Anda belum mengimplementasikan katalog data untuk menginventarisasi aset data Anda.

Manfaat menjalankan praktik terbaik ini: Dengan mengimplementasikan kebijakan klasifikasi data, Anda dapat menentukan tingkat penyimpanan paling hemat energi untuk data.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Klasifikasi data melibatkan identifikasi jenis-jenis data yang sedang diproses dan disimpan di sistem informasi yang dimiliki atau dioperasikan oleh organisasi. Klasifikasi data juga melibatkan penentuan tingkat kekritisan data dan kemungkinan dampak penyusupan, kehilangan, atau penyalahgunaan data.

Implementasikan kebijakan klasifikasi data dengan bekerja mundur dari penggunaan data kontekstual dan membuat skema kategorisasi yang mempertimbangkan tingkat kekritisan set data tertentu terhadap operasi organisasi.

Langkah implementasi

- Lakukan inventaris berbagai jenis data yang ada untuk beban kerja Anda.
  - Untuk detail selengkapnya tentang kategori data, lihat Laporan Resmi Klasifikasi Data.
- Tentukan kekritisan, kerahasiaan, integritas, dan ketersediaan data berdasarkan risiko terhadap organisasi. Gunakan persyaratan ini untuk mengelompokkan data ke dalam satu tingkat klasifikasi data yang Anda adopsi.
  - Sebagai contoh, lihat Empat langkah sederhana untuk mengklasifikasikan data Anda dan mengamankan startup Anda.

- Audit lingkungan Anda secara berkala untuk data yang tidak ditandai dan tidak diklasifikasikan, serta klasifikasikan dan tandai data dengan tepat.
  - Sebagai contoh, lihat Katalog Data dan perayap di AWS Glue.
- Bangun katalog data yang menyediakan kemampuan audit dan tata kelola.
- Tentukan dan dokumentasikan prosedur penanganan untuk setiap kelas data.
- Gunakan otomatisasi untuk mengaudit lingkungan Anda secara kontinu guna mengidentifikasi data yang tidak ditandai dan tidak diklasifikasikan, serta klasifikasikan dan tandai data dengan tepat.

Sumber daya

Dokumen terkait:

- Memanfaatkan AWS Cloud untuk Mendukung Klasifikasi Data
- Kebijakan tag dari AWS Organizations

Video terkait:

- Mewujudkan ketangkasan dengan tata kelola data di AWS

SUS04-BP02 Menggunakan teknologi yang mendukung pola akses dan penyimpanan data

Gunakan teknologi penyimpanan yang paling mendukung cara data Anda diakses dan disimpan untuk meminimalkan sumber daya yang disediakan sambil mendukung beban kerja Anda.

Antipola umum:

- Anda berasumsi bahwa semua beban kerja memiliki pola penyimpanan dan akses data yang serupa.
- Anda hanya menggunakan satu tingkat penyimpanan, dengan anggapan semua beban kerja masuk dalam tingkat tersebut.
- Anda berasumsi bahwa pola akses data tidak akan berubah.

Manfaat menjalankan praktik terbaik ini: Memilih dan mengoptimalkan teknologi penyimpanan Anda berdasarkan pola akses dan penyimpanan data akan membantu Anda mengurangi sumber daya cloud yang diperlukan untuk memenuhi kebutuhan bisnis dan meningkatkan keseluruhan efisiensi beban kerja cloud.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Pilih solusi penyimpanan yang paling sesuai dengan pola akses Anda, atau pertimbangkan untuk mengubah pola akses untuk menyesuaikan dengan solusi penyimpanan guna memaksimalkan efisiensi kinerja.

- Evaluasi karakteristik dan pola akses data Anda untuk mengumpulkan karakteristik utama kebutuhan penyimpanan Anda. Karakteristik utama yang perlu dipertimbangkan antara lain:
  - Tipe data: terstruktur, semi-terstruktur, tidak terstruktur
  - Pertumbuhan data: dibatasi, tidak dibatasi
  - Daya tahan data: persisten, sementara, transien
  - Pola akses: baca atau tulis, frekuensi, berfluktuasi, atau konsisten
- Migrasikan data ke teknologi penyimpanan yang tepat yang mendukung karakteristik dan pola akses data Anda. Berikut adalah beberapa contoh teknologi penyimpanan AWS serta karakteristik utamanya:

Tipe	Teknologi	Karakteristik utama
Penyimpanan objek	Amazon S3	Layanan penyimpanan objek dengan skalabilitas tak terbatas, ketersediaan tinggi, dan berbagai opsi aksesibilitas. Mentransfer dan mengakses objek masuk dan keluar dari Amazon S3 dapat menggunakan layanan, seperti Transfer Acceleration atau Access Points, untuk mendukung lokasi, kebutuhan keamanan, dan pola akses Anda.
Penyimpanan pengarsipan	Amazon Glacier	Kelas penyimpanan Amazon S3 yang dibuat untuk pengarsipan data.

Tipe	Teknologi	Karakteristik utama
Sistem file bersama	Amazon Elastic File System (Amazon EFS)	Sistem file mountable yang dapat diakses oleh berbagai jenis solusi komputasi . Amazon EFS secara otomatis memperbesar dan memperkecil penyimpanan serta dioptimalkan performan ya agar memberikan latensi rendah yang konsisten.
Sistem file bersama	Amazon FSx	Dibangun berdasarkan solusi komputasi AWS terbaru untuk mendukung empat sistem file yang umum digunakan: NetApp ONTAP, OpenZFS, Windows File Server, dan Lustre. Amazon FSx memiliki latensi, throughput, dan IOPS yang bervariasi per sistem file dan hal ini harus dipertimbangkan saat memilih sistem file yang tepat untuk kebutuhan beban kerja Anda.

Tipe	Teknologi	Karakteristik utama
Penyimpanan blok	Amazon Elastic Block Store (Amazon EBS)	Layanan penyimpanan blok kinerja tinggi yang dapat diskalakan yang dirancang untuk Amazon Elastic Compute Cloud (Amazon EC2). Amazon EBS mencakup penyimpanan yang didukung SSD untuk beban kerja transaksional intensif IOPS dan penyimpanan yang didukung HDD untuk beban kerja intensif throughput.
Basis data relasional	Amazon Aurora, Amazon RDS, Amazon Redshift	Didesain untuk mendukung transaksi ACID (atomisit as, konsistensi, isolasi, durabilitas), dan mempertah ankan integritas referensi al serta konsistensi data yang tinggi. Banyak aplikasi tradisional, perencanaan sumber daya perusahaan (ERP), manajemen hubungan pelanggan (CRM), dan sistem perdagangan elektronik menggunakan basis data relasional untuk menyimpan data mereka.

Tipe	Teknologi	Karakteristik utama
Basis data nilai-kunci	Amazon DynamoDB	Dioptimalkan untuk pola akses umum, biasanya untuk menyimpan dan mengambil data dalam volume besar. Aplikasi web dengan lalu lintas tinggi, sistem perdagangan elektronik, dan aplikasi gaming merupakan kasus penggunaan umum untuk basis data nilai kunci.

- Untuk sistem penyimpanan dengan ukuran tetap, seperti Amazon EBS atau Amazon FSx, pantau ruang penyimpanan yang tersedia dan otomatiskan alokasi penyimpanan saat ambang batas tercapai. Anda dapat memanfaatkan Amazon CloudWatch untuk mengumpulkan dan menganalisis metrik yang berbeda-beda untuk Amazon EBS dan Amazon FSx.
- Kelas Penyimpanan Amazon S3 dapat dikonfigurasi pada level objek dan satu bucket dapat berisi objek yang disimpan di semua kelas penyimpanan.
- Anda juga dapat menggunakan kebijakan Siklus Hidup Amazon S3 untuk mengalihkan objek secara otomatis antar kelas-kelas penyimpanan atau menghapus data tanpa perubahan aplikasi apa pun. Secara umum, Anda harus memilih mana yang penting antara efisiensi sumber daya, latensi akses, dan keandalan saat mempertimbangkan semua mekanisme penyimpanan ini.

Sumber daya

Dokumen terkait:

- Jenis volume Amazon EBS
- Penyimpanan instans Amazon EC2
- Amazon S3 Intelligent-Tiering
- Karakteristik I/O Amazon EBS
- Menggunakan kelas penyimpanan Amazon S3
- Apa itu Amazon Glacier?

Video terkait:

- Pola Arsitektur untuk Danau Data di AWS
- Pendalaman tentang Amazon EBS (STG303-R1)
- Optimalkan kinerja penyimpanan Anda dengan Amazon S3 (STG343)
- Membangun arsitektur data modern di AWS

Contoh terkait:

- Driver CSI Amazon EFS
- Driver CSI Amazon EBS
- Utilitas Amazon EFS
- Penskalaan Otomatis Amazon EBS
- Contoh Amazon S3

SUS04-BP03 Menggunakan kebijakan untuk mengelola siklus hidup set data Anda

Kelola siklus hidup semua data Anda dan terapkan penghapusan secara otomatis untuk meminimalkan total penyimpanan yang diperlukan untuk beban kerja Anda.

Antipola umum:

- Anda menghapus data secara manual.
- Anda tidak menghapus data beban kerja Anda sama sekali.
- Anda tidak mengalihkan data ke tingkat penyimpanan yang lebih hemat energi berdasarkan persyaratan retensi dan aksesnya.

Manfaat menjalankan praktik terbaik ini: Penggunaan kebijakan siklus hidup data memastikan akses dan retensi data yang efisien dalam suatu beban kerja.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Set data biasanya memiliki persyaratan retensi dan akses data yang berbeda-beda selama masa hidupnya. Misalnya, aplikasi Anda mungkin memerlukan akses yang sering ke sejumlah set data dalam jangka waktu terbatas. Setelah itu, set-set data tersebut jarang diakses.

Untuk mengelola set data Anda secara efisien di sepanjang siklus hidupnya, konfigurasikan kebijakan siklus hidup, yakni aturan yang menetapkan cara menangani set data.

Dengan Aturan konfigurasi siklus hidup, Anda dapat meminta layanan penyimpanan tertentu untuk mengalihkan suatu set data ke tingkat penyimpanan yang lebih hemat energi, mengarsipkannya, atau menghapusnya.

Langkah implementasi

- Klasifikasikan set data dalam beban kerja Anda.
- Tetapkan prosedur penanganan untuk setiap kelas data.
- Atur kebijakan siklus hidup otomatis untuk menegakkan aturan siklus hidup. Berikut ini adalah beberapa cara menyiapkan kebijakan siklus hidup otomatis untuk berbagai layanan penyimpanan AWS:

Storage service	How to set automated lifecycle policies
Amazon S3	Anda dapat menggunakan Siklus Hidup Amazon S3 untuk mengelola objek-objek Anda di sepanjang siklus hidupnya. Jika pola akses Anda tidak diketahui, berubah-u bah, atau tidak dapat diprediksi, Anda dapat menggunakan Amazon S3 Intelligent-Tierin g, yang memantau pola akses dan secara otomatis memindahkan objek yang belum diakses ke tingkat akses dengan biaya lebih rendah. Anda dapat memanfaatkan metrik Lensa Penyimpanan Amazon S3 untuk mengidentifikasi peluang dan celah pengoptim alan dalam manajemen siklus hidup.
Amazon Elastic Block Store	Anda dapat menggunakan Amazon Data Lifecycle Manager untuk mengotomatiskan pembuatan, retensi, dan penghapusan snapshot Amazon EBS dan AMI yang dicadangkan Amazon EBS.

Storage service	How to set automated lifecycle policies
Amazon Elastic File System	Manajemen siklus hidup Amazon EFS secara otomatis mengelola penyimpanan file untuk sistem file Anda.
Amazon Elastic Container Registry	Kebijakan siklus hidup Amazon ECR mengotomatiskan pembersihan image kontainer Anda dengan mengakhiri masa berlaku image berdasarkan usia atau jumlah.
AWS Elemental MediaStore	Anda dapat menggunakan kebijakan siklus hidup objek yang mengatur seberapa lama objek harus disimpan di kontainer MediaStore.

- Hapus volume, snapshot, dan data yang tidak digunakan yang sudah melebihi masa retensinya. Manfaatkan fitur layanan native seperti Amazon DynamoDB Time To Live atau retensi log Amazon CloudWatch untuk penghapusan.
- Agregasikan dan kompresi data jika memungkinkan berdasarkan aturan siklus hidup.

Sumber daya

Dokumen terkait:

- Optimalkan aturan Siklus Hidup Amazon S3 Anda dengan Analisis Kelas Penyimpanan Amazon S3
- Mengevaluasi Sumber Daya dengan Aturan AWS Config

Video terkait:

- Sederhanakan Siklus Hidup Data Anda dan Optimalkan Biaya Penyimpanan dengan Siklus Hidup Amazon S3
- Kurangi Biaya Penyimpanan Anda Menggunakan Lensa Penyimpanan Amazon S3

SUS04-BP04 Menggunakan elastisitas dan otomatisasi untuk memperluas sistem file atau penyimpanan blok

Gunakan elastisitas dan otomatisasi untuk memperluas sistem file atau penyimpanan blok seiring pertumbuhan data untuk meminimalkan total penyimpanan yang disediakan.

Antipola umum:

- Anda membeli sistem file atau penyimpanan blok besar untuk keperluan di waktu mendatang.
- Anda memberikan persediaan berlebih untuk operasi input dan output per detik (IOPS) sistem file Anda.
- Anda tidak memantau pemanfaatan volume data Anda.

Manfaat menjalankan praktik terbaik ini: Meminimalkan penyediaan yang berlebihan untuk sistem penyimpanan mengurangi sumber daya yang tidak aktif dan meningkatkan efisiensi beban kerja Anda secara keseluruhan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Buat sistem file dan penyimpanan blok dengan alokasi ukuran, throughput, dan latensi yang sesuai untuk beban kerja Anda. Gunakan elastisitas dan otomatisasi untuk memperluas sistem file atau penyimpanan blok seiring pertumbuhan data untuk meminimalkan total penyimpanan yang disediakan.

Langkah implementasi

- Untuk penyimpanan berukuran tetap seperti Amazon EBS, pastikan Anda memantau kapasitas penyimpanan yang digunakan dibandingkan dengan keseluruhan ukuran penyimpanan, dan buat otomatisasi jika memungkinkan untuk meningkatkan ukuran penyimpanan saat mencapai ambang batas.
- Gunakan volume elastis dan layanan data blok terkelola untuk mengotomatisasi alokasi penyimpanan tambahan seiring tumbuhnya data persisten Anda. Contohnya, Anda dapat menggunakan Elastic Volumes Amazon EBS untuk mengubah ukuran volume, jenis volume, atau menyesuaikan performa volume Amazon EBS Anda.
- Pilih kelas penyimpanan, mode performa, dan mode throughput yang tepat untuk sistem file Anda guna memenuhi kebutuhan bisnis, tidak melebihinya.

  - Performa Amazon EFS
  - Performa volume Amazon EBS di instans Linux
- Atur target tingkat pemanfaatan untuk volume data Anda, dan ubah ukuran volume di luar rentang yang diperkirakan.
- Sesuaikan ukuran volume hanya-baca agar sesuai dengan data.
- Migrasikan data ke penyimpanan objek untuk menghindari penyediaan kapasitas yang berlebihan dari ukuran volume tetap di penyimpanan blok.
- Secara rutin tinjau volume elastis dan sistem file untuk menghentikan volume yang tidak aktif dan memperkecil sumber daya dengan penyediaan berlebihan agar sesuai dengan ukuran data saat ini.

Sumber daya

Dokumen terkait:

- Dokumentasi Amazon FSx
- Apa itu Amazon Elastic File System?

Video terkait:

- Memahami Elastic Volumes Amazon EBS
- Strategi Optimisasi Cuplikan dan Amazon EBS untuk Peningkatan Performa dan Penghematan Biaya
- Mengoptimalkan Amazon EFS untuk biaya dan performa, menggunakan praktik terbaik

SUS04-BP05 Menyingkirkan data yang tidak diperlukan atau redundan

Hapus data yang tidak diperlukan atau redundan untuk meminimalkan sumber daya penyimpanan yang diperlukan untuk menyimpan set data Anda.

Antipola umum:

- Anda menduplikasi data yang dapat diperoleh atau dibuat ulang dengan mudah
- Anda mencadangkan semua data tanpa mempertimbangkan tingkat kekritisannya.
- Anda menghapus data tidak rutin, hanya pada peristiwa operasional, atau tidak menghapusnya sama sekali.

- Anda menyimpan data secara redundan dengan mengabaikan durabilitas layanan penyimpanan.
- Anda mengaktifkan versioning Amazon S3 tanpa alasan bisnis apa pun.

Manfaat menjalankan praktik terbaik ini: Penghapusan data yang tidak diperlukan dapat mengurangi ukuran penyimpanan yang diperlukan untuk beban kerja Anda serta dampak beban kerja terhadap lingkungan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Jangan menyimpan data yang tidak Anda perlukan. Otomatiskan penghapusan data yang tidak diperlukan. Gunakan teknologi yang menghilangkan data ganda pada tingkat file dan blok. Manfaatkan fitur replikasi dan redundansi data native dari layanan.

Langkah implementasi

- Evaluasi apakah Anda dapat menghindari menyimpan data menggunakan set data yang saat ini tersedia untuk publik di AWS Data Exchange dan Data Terbuka di AWS.
- Gunakan mekanisme yang dapat membatalkan duplikasi data pada tingkat blok dan objek. Berikut ini adalah beberapa contoh cara membatalkan duplikasi data di AWS:

Storage service	Deduplication mechanism
Amazon S3	Gunakan AWS Lake Formation FindMatches untuk menemukan catatan pencocokan di sebuah set data (termasuk tanpa pengidentifikasi) menggunakan FindMatches ML Transform baru.
Amazon FSx	Aktifkan pembatalan duplikasi data di Amazon FSx untuk Windows.
Snapshot Amazon Elastic Block Store	Snapshot adalah cadangan bertahap, yang berarti penyimpanan hanya dilakukan untuk blok di perangkat yang telah berubah setelah snapshot terbaru Anda.

- Analisis akses data untuk mengidentifikasi data yang tidak diperlukan. Otomatiskan kebijakan siklus hidup. Manfaatkan fitur layanan native seperti Amazon DynamoDB Time To Live, Siklus Hidup Amazon S3, atau retensi log Amazon CloudWatch untuk penghapusan.
- Gunakan kemampuan virtualisasi data di AWS untuk mempertahankan data di sumbernya dan menghindari duplikasi data.
  - Virtualisasi Data Cloud Native di AWS
  - Lab: Mengoptimalkan Pola Data Menggunakan Fitur Berbagi Data Amazon Redshift
- Gunakan teknologi pencadangan yang dapat membuat cadangan bertahap.
- Manfaatkan durabilitas Amazon S3 dan replikasi Amazon EBS untuk memenuhi tujuan durabilitas Anda, bukan teknologi yang dikelola mandiri (seperti rangkaian disk independen yang redundan (RAID)).
- Pusatkan log dan lacak data, batalkan duplikasi entri log yang identik, dan buat mekanisme untuk menyesuaikan verbositas saat diperlukan.
- Pra-isi cache hanya saat ada alasan yang dibenarkan.
- Lakukan pemantauan dan otomatisasi cache untuk menyesuaikan ukuran cache dengan tepat.
- Singkirkan deployment dan aset usang dari penyimpanan objek dan cache edge saat mendorong versi baru untuk beban kerja Anda.

Sumber daya

Dokumen terkait:

- Ubah retensi data log di CloudWatch Logs
- Pembatalan duplikasi data di Amazon FSx untuk Windows File Server
- Fitur Amazon FSx untuk ONTAP termasuk pembatalan duplikasi data
- Membatalkan File di Amazon CloudFront
- Menggunakan AWS Backup untuk mencadangkan dan memulihkan sistem file Amazon EFS
- Apa yang dimaksud dengan Amazon CloudWatch Logs?
- Bekerja dengan cadangan di Amazon RDS

Video terkait:

- Pencocokan Fuzzy dan Pembatalan Duplikasi Data dengan ML Transforms untuk AWS Lake Formation

Contoh terkait:

- Bagaimana cara menganalisis log akses server Amazon S3 menggunakan Amazon Athena?

SUS04-BP06 Menggunakan sistem file atau penyimpanan bersama untuk mengakses data umum

Adopsi sistem file atau penyimpanan bersama untuk menghindari duplikasi data dan memungkinkan infrastruktur yang lebih efisien untuk beban kerja Anda.

Antipola umum:

- Anda menyediakan penyimpanan untuk setiap klien secara individu.
- Anda tidak melepaskan volume data dari klien yang tidak aktif.
- Anda tidak memberikan akses ke penyimpanan di semua platform dan sistem.

Manfaat menjalankan praktik terbaik ini: Menggunakan sistem file atau penyimpanan bersama memungkinkan pemberian data ke satu atau lebih pemakai tanpa harus menyalin data tersebut. Hal ini membantu mengurangi sumber daya penyimpanan yang diperlukan untuk beban kerja.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Jika Anda memiliki beberapa pengguna atau aplikasi yang mengakses set data yang sama, penggunaan teknologi penyimpanan bersama sangatlah penting agar infrastruktur untuk beban kerja Anda efisien. Teknologi penyimpanan bersama memberikan lokasi sentral untuk menyimpan dan mengelola set data dan menghindari duplikasi data. Teknologi ini juga memastikan konsistensi data di berbagai sistem yang berlainan. Lebih lanjut, teknologi penyimpanan bersama memungkinkan penggunaan daya komputasi yang lebih efisien, karena beberapa sumber daya komputasi dapat mengakses dan memproses data pada saat yang sama secara paralel.

Hanya ambil data dari layanan penyimpanan bersama ini sesuai kebutuhan dan lepaskan volume yang tidak digunakan untuk membebaskan sumber daya.

Langkah implementasi

- Migrasikan data ke penyimpanan bersama ketika data memiliki beberapa pemakai. Berikut beberapa contoh teknologi penyimpanan bersama di AWS:

Storage option	When to use
Multi-Attach Amazon EBS	Amazon EBS Multi-Attach memungkinkan Anda melampirkan satu volume SSD IOPS yang Tersedia (io1 atau io2) ke beberapa instans yang berada di Zona Ketersediaan yang sama.
Amazon EFS	Lihat Kapan Harus Memilih Amazon EFS.
Amazon FSx	Lihat Memilih Sistem File Amazon FSx.
Amazon S3	Aplikasi yang tidak memerlukan struktur sistem file dan didesain untuk berfungsi dengan penyimpanan objek dapat menggunakan Amazon S3 sebagai solusi penyimpanan objek yang dapat diskalakan besar-besaran, tahan lama, dan murah.

- Salin data ke sistem file atau ambil data dari sistem file bersama hanya jika dibutuhkan. Contohnya, Anda dapat membuat sistem file Amazon FSx for Lustreyang didukung oleh Amazon S3 dan hanya memuat subset data yang diperlukan untuk tugas pemrosesan ke Amazon FSx.
- Hapus data sesuai pola penggunaan Anda sebagaimana dijelaskan di SUS04-BP03 Menggunakan kebijakan untuk mengelola siklus hidup set data Anda.
- Lepaskan volume dari klien yang tidak menggunakannya secara aktif.

Sumber daya

Dokumen terkait:

- Menghubungkan sistem file Anda ke bucket Amazon S3
- Menggunakan Amazon EFS untuk AWS Lambda di aplikasi nirserver Anda
- Intelligent-Tiering Amazon EFS Mengoptimalkan Biaya untuk Beban Kerja dengan Mengubah Pola Akses
- Menggunakan Amazon FSx dengan repositori data on-premise Anda

Video terkait:

- Optimisasi biaya penyimpanan dengan Amazon EFS

SUS04-BP07 Meminimalkan perpindahan data di jaringan

Gunakan sistem file atau penyimpanan objek bersama untuk mengakses data umum dan meminimalkan total sumber daya jaringan yang diperlukan untuk mendukung perpindahan data beban kerja Anda.

Antipola umum:

- Anda menyimpan semua data di Wilayah AWS yang sama terlepas di mana pengguna data berada.
- Anda tidak mengoptimalkan format dan ukuran data sebelum memindahkannya melalui jaringan.

Manfaat menjalankan praktik terbaik ini: Mengoptimalkan perpindahan data di seluruh jaringan mengurangi total sumber daya jaringan yang diperlukan untuk beban kerja dan memperkecil dampaknya pada lingkungan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Memindahkan data ke berbagai bagian dalam organisasi memerlukan sumber daya komputasi, jaringan, dan penyimpanan. Gunakan teknik untuk meminimalkan perpindahan data dan meningkatkan efisiensi beban kerja Anda secara keseluruhan.

Langkah implementasi

- Pertimbangkan kedekatan jarak ke data atau pengguna sebagai faktor penentu ketika memilih Wilayah untuk beban kerja Anda.
- Partisi layanan yang digunakan secara Regional sehingga data khusus Wilayahnya disimpan di Wilayah tempat penggunaan data.
- Gunakan format file yang efisien (seperti Parquet atau ORC) dan kompresi data sebelum memindahkannya melalui jaringan.
- Jangan pindahkan data yang tidak digunakan. Beberapa contoh tindakan yang dapat membantu Anda menghindari pemindahan data yang tidak digunakan:

- Kurangi respons API untuk data yang relevan saja.
- Kumpulkan data apabila terperinci (informasi tingkat catatan tidak diperlukan).
- Lihat Lab Well-Architected - Mengoptimalkan Pola Data Menggunakan Fitur Berbagi Data Amazon Redshift.
- Pertimbangkan Berbagi data lintas akun di AWS Lake Formation.

- Gunakan layanan yang dapat membantu Anda menjalankan kode lebih dekat dengan pengguna beban kerja Anda.

Layanan	Kapan harus digunakan
Lambda@Edge	Gunakan untuk operasi dengan banyak komputasi yang dijalankan saat objek tidak ada dalam cache.
Fungsi CloudFront	Gunakan untuk kasus penggunaan sederhana seperti permintaan HTTP atau manipulasi respons yang dapat dimulai oleh fungsi dengan masa pakai singkat.
AWS IoT Greengrass	Jalankan komputasi lokal, olahpesan, dan caching data untuk perangkat yang terhubung.

Sumber daya

Dokumen terkait:

- Mengoptimalkan Infrastruktur AWS untuk Pelestarian Lingkungan, Bagian III: Jaringan
- Infrastruktur Global AWS
- Fitur Utama Amazon CloudFront meliputi Jaringan Edge Global CloudFront
- Mengompresi permintaan HTTP di Amazon OpenSearch Service
- Kompresi data menengah dengan Amazon EMR
- Memuat file kompresi data dari Amazon S3 ke Amazon Redshift
- Menyajikan file kompresi dengan Amazon CloudFront

Video terkait:

- Menjelaskan transfer data di AWS

Contoh terkait:

- Arsitektur untuk keberlanjutan - Meminimalkan pergerakan data lintas jaringan

SUS04-BP08 Hanya mencadangkan data saat sulit untuk dibuat ulang

Hindari mencadangkan data yang tidak memiliki nilai bisnis untuk meminimalkan persyaratan sumber daya penyimpanan untuk beban kerja Anda.

Antipola umum:

- Anda tidak memiliki strategi cadangan untuk data Anda.
- Anda mencadangkan data yang dapat dibuat ulang dengan mudah.

Manfaat menjalankan praktik terbaik ini: Menghindari pencadangan data yang tidak penting mengurangi sumber daya penyimpanan yang diperlukan untuk beban kerja dan memperkecil dampaknya pada lingkungan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Menghindari pencadangan data yang tidak perlu dapat membantu menurunkan biaya dan mengurangi sumber daya penyimpanan yang digunakan oleh beban kerja. Hanya cadangkan data yang memiliki nilai bisnis atau yang diperlukan untuk memenuhi persyaratan kepatuhan. Periksa kebijakan pencadangan dan jangan sertakan penyimpanan sementara yang tidak memberikan nilai dalam skenario pemulihan.

Langkah implementasi

- Implementasikan kebijakan klasifikasi data sebagaimana dijelaskan di SUS04-BP01 Mengimplementasikan kebijakan klasifikasi data.
- Gunakan kritikalitas klasifikasi data Anda dan desain strategi pencadangan berdasarkan sasaran waktu pemulihan (RTO) dan sasaran titik pemulihan (RPO). Hindari mencadangkan data yang tidak penting.
  - Jangan sertakan data yang dapat dibuat ulang dengan mudah.

    - Jangan sertakan data sementara dari cadangan Anda.
    - Jangan sertakan salinan lokal data, kecuali apabila waktu yang diperlukan untuk memulihkan data tersebut dari lokasi umum melebihi perjanjian tingkat layanan (SLA) Anda.
- Gunakan solusi otomatis atau layanan terkelola untuk mencadangkan data yang penting bagi bisnis.
    - AWS Backup adalah layanan terkelola penuh yang mempermudah pemusatan dan pengotomatisan perlindungan data di seluruh layanan AWS, di cloud, dan on-premise. Untuk panduan praktik langsung tentang cara membuat cadangan otomatis menggunakan AWS Backup, lihat Well-Architected Labs - Pengujian Pencadangan dan Pemulihan Data.
    - Otomatiskan cadangan dan optimalkan biaya cadangan untuk Amazon EFS menggunakan AWS Backup.

Sumber daya

Praktik terbaik terkait:

- REL09-BP01 Mengidentifikasi dan mencadangkan semua data yang perlu dicadangkan, atau memproduksi ulang data dari sumber
- REL09-BP03 Melakukan pencadangan data secara otomatis
- REL13-BP02 Menggunakan strategi pemulihan untuk memenuhi sasaran pemulihan

Dokumen terkait:

- Menggunakan AWS Backup untuk mencadangkan dan memulihkan sistem file Amazon EFS
- Snapshot Amazon EBS
- Bekerja dengan cadangan di Amazon Relational Database Service
- Partner APN: partner yang dapat membantu pencadangan
- AWS Marketplace: produk yang dapat digunakan untuk pencadangan
- Mencadangkan Amazon EFS
- Mencadangkan Amazon FSx untuk Windows File Server
- Pencadangan dan Pemulihan untuk Amazon ElastiCache (Redis OSS)

Video terkait:

- AWS re:Invent 2021 - Pencadangan, pemulihan bencana, dan perlindungan ransomware dengan AWS
- Demo AWS Backup: Pencadangan Lintas Akun dan Lintas Wilayah
- AWS re:Invent 2019: Memahami AWS Backup, dengan Rackspace (STG341)

Contoh terkait:

- Well-Architected Lab - Pengujian Pencadangan dan Pemulihan Data
- Lab Well-Architected - Pencadangan dan Pemulihan dengan Failback untuk Beban Kerja Analitik
- Lab Well-Architected - Pemulihan Bencana - Pencadangan dan Pemulihan

Perangkat keras dan layanan

Pertanyaan

- SUS 5 Bagaimana cara Anda memilih dan menggunakan perangkat keras serta layanan cloud di arsitektur Anda untuk mendukung tujuan pelestarian lingkungan Anda?

SUS 5 Bagaimana cara Anda memilih dan menggunakan perangkat keras serta layanan cloud di arsitektur Anda untuk mendukung tujuan pelestarian lingkungan Anda?

Cari peluang untuk mengurangi dampak beban kerja terhadap pelestarian lingkungan dengan membuat perubahan pada praktik manajemen perangkat keras Anda. Minimalkan jumlah perangkat keras yang perlu disediakan dan di-deploy, serta pilih perangkat keras dan layanan yang paling efisien untuk setiap beban kerja Anda.

Praktik terbaik

- SUS05-BP01 Menggunakan perangkat keras dalam jumlah minim untuk memenuhi kebutuhan Anda
- SUS05-BP02 Menggunakan jenis instans dengan dampak paling sedikit
- SUS05-BP03 Menggunakan layanan terkelola
- SUS05-BP04 Mengoptimalkan penggunaan akselerator komputasi berbasis perangkat keras

SUS05-BP01 Menggunakan perangkat keras dalam jumlah minim untuk memenuhi kebutuhan Anda

Gunakan perangkat keras dalam jumlah minim untuk beban kerja Anda guna memenuhi kebutuhan bisnis secara efisien.

Antipola umum:

- Anda tidak memantau pemanfaatan sumber daya.
- Anda memiliki sumber daya dengan tingkat pemanfaatan rendah di arsitektur Anda.
- Anda tidak meninjau pemanfaatan perangkat keras statis untuk menentukan apakah harus diubah ukurannya.
- Anda tidak menetapkan target pemanfaatan perangkat keras untuk infrastruktur komputasi Anda berdasarkan KPI bisnis.

Manfaat menjalankan praktik terbaik ini: Menyesuaikan ukuran sumber daya cloud Anda membantu mengurangi dampak beban kerja pada lingkungan, menghemat uang, dan mempertahankan tolok ukur performa.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Pilih secara optimal jumlah total perangkat keras yang diperlukan untuk beban kerja Anda guna meningkatkan efisiensinya secara keseluruhan. AWS Cloud memberikan fleksibilitas untuk memperluas atau mengurangi jumlah sumber daya secara dinamis melalui beragam mekanisme, seperti AWS Auto Scaling, dan memenuhi perubahan sesuai permintaan. Layanan ini juga menyediakan API dan SDK yang memungkinkan sumber daya dapat dimodifikasi dengan upaya minimal. Gunakan kemampuan ini untuk membuat perubahan dengan sering pada implementasi beban kerja Anda. Selain itu, gunakan panduan penyesuaian ukuran dari alat AWS untuk secara efisien mengoperasikan sumber daya cloud Anda dan memenuhi kebutuhan bisnis.

Langkah implementasi

- Pilih jenis instans yang paling sesuai dengan kebutuhan Anda.
  - Bagaimana cara memilih jenis instans Amazon EC2 yang tepat untuk beban kerja saya?
  - Pemilihan jenis Instans berdasarkan atribut untuk Armada Amazon EC2.
  - Buat grup Auto Scaling menggunakan pemilihan jenis instans berdasarkan atribut.

- Skalakan menggunakan peningkatan kecil untuk beban kerja variabel.
- Gunakan beberapa opsi pembelian komputasi untuk menyeimbangkan fleksibilitas instans, skalabilitas, dan penghematan biaya.
  - Instans Sesuai Permintaan paling sesuai untuk beban kerja baru, berfluktuasi, dan stateful yang tidak dapat berupa jenis instans, lokasi, atau fleksibel waktunya.
  - Instans Spot merupakan cara yang bagus untuk menambahkan opsi lain untuk aplikasi yang fleksibel dan toleransi terhadap kesalahan.
  - Manfaatkan Compute Savings Plans untuk beban kerja steady state yang memungkinkan fleksibilitas jika kebutuhan Anda (seperti AZ, Wilayah, kelompok instans, atau jenis instans) berubah.
- Gunakan keragaman zona ketersediaan dan instans untuk memaksimalkan ketersediaan aplikasi dan memanfaatkan kapasitas yang berlebih apabila mungkin.
- Gunakan rekomendasi penyesuaian ukuran dari alat AWS untuk melakukan penyesuaian pada beban kerja Anda.
  - AWS Compute Optimizer
  - AWS Trusted Advisor
- Negosiasikan perjanjian tingkat layanan (SLA) agar kapasitas dapat dikurangi sementara di saat otomatisasi melakukan deployment sumber daya pengganti.

Sumber daya

Dokumen terkait:

- Mengoptimalkan Infrastruktur AWS untuk Pelestarian Lingkungan, Bagian I: Komputasi
- Pemilihan Jenis Instans Berdasarkan Atribut untuk Auto Scaling untuk Armada Amazon EC2
- Dokumentasi AWS Compute Optimizer
- Mengoperasikan Lambda: Optimisasi performa
- Dokumentasi Penskalaan Otomatis

Video terkait:

- Bangun lingkungan komputasi yang hemat biaya, energi, dan sumber daya

Contoh terkait:

- Well-Architected Lab: Menyesuaikan Ukuran dengan Mengaktifkan AWS Compute Optimizer dan Pemanfaatan Memori (Level 200)

SUS05-BP02 Menggunakan jenis instans dengan dampak paling sedikit

Terus pantau dan gunakan jenis instans baru untuk memanfaatkan peningkatan penghematan energi.

Antipola umum:

- Anda hanya menggunakan satu kelompok instans.
- Anda hanya menggunakan instans x86.
- Anda menentukan satu jenis instans dalam konfigurasi Amazon EC2 Auto Scaling Anda.
- Anda menggunakan instans AWS dengan cara yang tidak dirancang untuk instans tersebut (misalnya, Anda menggunakan instans komputasi yang dioptimalkan untuk beban kerja intensif memori).
- Anda tidak mengevaluasi jenis instans baru secara teratur.
- Anda tidak melihat rekomendasi dari alat rightsizing AWS seperti AWS Compute Optimizer.

Manfaat menjalankan praktik terbaik ini: Dengan memanfaatkan instans hemat energi dan berukuran tepat, Anda dapat jauh mengurangi dampak lingkungan dan biaya beban kerja Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Menggunakan instans yang efisien di beban kerja cloud sangat penting untuk menurunkan penggunaan sumber daya dan menghemat biaya. Terus pantau rilis instans jenis baru dan manfaatkan peningkatan penghematan energi, termasuk jenis instans yang dirancang untuk mendukung beban kerja spesifik seperti pelatihan dan inferensi machine learning, serta transkode video.

Langkah implementasi

- Pelajari dan jelajahi jenis instans yang dapat menurunkan dampak lingkungan beban kerja Anda.
  - Berlangganan ke Apa yang Baru dengan AWS untuk mendapatkan informasi terbaru terkait teknologi dan instans AWS terbaru.
  - Pelajari tentang jenis instans AWS yang berbeda-beda.

  - Pelajari tentang instans berbasis AWS Graviton yang menawarkan performa terbaik per watt untuk penggunaan energi di Amazon EC2 dengan menonton re:Invent 2020 - Pendalaman tentang instans Amazon EC2 yang didukung prosesor AWS Graviton2 dan Pendalaman tentang AWS Graviton3 dan instans C7g Amazon EC2.
- Rencanakan dan transisikan beban kerja Anda ke jenis instans dengan dampak paling kecil.
  - Tentukan proses untuk mengevaluasi fitur atau instans baru untuk beban kerja Anda. Manfaatkan ketangkasan di cloud untuk menguji dengan cepat bagaimana jenis instans baru dapat meningkatkan pelestarian lingkungan beban kerja Anda. Gunakan metrik proksi untuk mengukur berapa banyak sumber daya yang Anda perlukan untuk menyelesaikan satu unit pekerjaan.
  - Jika memungkinkan, ubah beban kerja menjadi menggunakan jumlah vCPU yang berbeda dan jumlah memori yang berbeda guna memaksimalkan pilihan jenis instans.
  - Pertimbangkan untuk mengalihkan beban kerja Anda ke instans berbasis Graviton guna meningkatkan efisiensi performa beban kerja Anda.
    - AWS Graviton Fast Start
    - Pertimbangan saat mentransisikan beban kerja ke instans Amazon Elastic Compute Cloud berbasis AWS Graviton
    - AWS Graviton2 untuk ISV
  - Pertimbangkan untuk memilih opsi AWS Graviton dalam penggunaan layanan terkelola AWS Anda.
  - Migrasikan beban kerja ke Wilayah yang menawarkan instans dengan dampak paling sedikit terhadap pelestarian lingkungan dan masih memenuhi kebutuhan bisnis Anda.
  - Untuk beban kerja machine learning, manfaatkan perangkat keras yang dibuat khusus untuk beban kerja Anda seperti AWS Trainium, AWS Inferentia, dan Amazon EC2 DL1. Instans AWS Inferentia seperti Inf2 menawarkan kinerja per watt hingga 50% lebih baik daripada instans berbasis Amazon EC2 yang setara.
  - Gunakan Amazon SageMaker AI Inference Recommender untuk menyesuaikan titik akhir inferensi ML secara tepat.
  - Untuk beban kerja yang berfluktuasi (beban kerja yang jarang memerlukan kapasitas tambahan), gunakan instans performa burstable.
  - Untuk beban kerja stateless dan toleran terhadap kesalahan, gunakan Instans Spot Amazon EC2 guna meningkatkan pemanfaatan cloud secara keseluruhan, serta mengurangi dampak terhadap pelestarian lingkungan dari sumber daya yang tidak digunakan.

- Operasikan dan optimalkan instans beban kerja Anda.
  - Untuk beban kerja sementara, evaluasi metrik Amazon CloudWatch instans seperti `CPUUtilization` untuk mengidentifikasi apakah instans tidak aktif atau kurang dimanfaatkan.
  - Untuk beban kerja stabil, lihat alat rightsizing AWS seperti AWS Compute Optimizer secara berkala untuk mengidentifikasi peluang guna mengoptimalkan dan menyesuaikan ukuran instans dengan tepat.
    - Lab Well-Architected: Rekomendasi Penyesuaian Ukuran
    - Lab Well-Architected - Penyesuaian Ukuran dengan Compute Optimizer
    - Lab Well-Architected - Optimisasi Pola Perangkat Keras dan Pengamatan KPI Pelestarian Lingkungan

Sumber daya

Dokumen terkait:

- Mengoptimalkan Infrastruktur AWS untuk Pelestarian Lingkungan, Bagian I: Komputasi
- AWS Graviton
- Amazon EC2 DL1
- Armada Reservasi Kapasitas Amazon EC2
- Armada Spot Amazon EC2
- Fungsi: Konfigurasi Fungsi Lambda
- Pemilihan jenis instans berdasarkan atribut untuk Armada Amazon EC2
- Membangun Aplikasi yang Berkelanjutan, Efisien, dan Dioptimalkan untuk Biaya di AWS
- Bagaimana Dasbor Pelestarian Lingkungan Contino Membantu Pelanggan Mengoptimalkan Jejak Karbon Mereka

Video terkait:

- Pendalaman tentang instans Amazon EC2 yang didukung prosesor AWS Graviton2
- Pendalaman tentang AWS Graviton3 dan instans C7g Amazon EC2
- Bangun lingkungan komputasi yang hemat biaya, energi, dan sumber daya

Contoh terkait:

- Solusi: Panduan untuk Mengoptimalkan Beban Kerja Deep Learning untuk Pelestarian Lingkungan di AWS
- Lab Well-Architected: Rekomendasi Penyesuaian Ukuran
- Lab Well-Architected - Penyesuaian Ukuran dengan Compute Optimizer
- Lab Well-Architected - Optimisasi Pola Perangkat Keras dan Pengamatan KPI Pelestarian Lingkungan
- Lab Well-Architected - Migrasi Layanan ke Graviton

SUS05-BP03 Menggunakan layanan terkelola

Gunakan layanan terkelola untuk beroperasi dengan lebih efisien di cloud.

Antipola umum:

- Anda menggunakan instans Amazon EC2 dengan pemanfaatan rendah untuk menjalankan aplikasi Anda.
- Tim internal Anda hanya mengelola beban kerja, tanpa ada waktu untuk berfokus pada inovasi atau simplifikasi.
- Anda melakukan deployment dan memelihara teknologi untuk tugas-tugas yang dapat dijalankan dengan lebih efisien di layanan terkelola.

Manfaat menjalankan praktik terbaik ini:

- Menggunakan layanan terkelola mengalihkan tanggung jawab ke AWS, yang memiliki wawasan atas jutaan pelanggan yang dapat membantu mendorong efisiensi dan inovasi baru.
- Layanan terkelola mendistribusikan dampak lingkungan dari layanan ke banyak pengguna karena bidang kendali multi-prinsip.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Layanan terkelola mengalihkan tanggung jawab ke AWS untuk mempertahankan pemanfaatan tinggi dan optimisasi pelestarian lingkungan dari deployment perangkat keras. Layanan terkelola juga menghilangkan beban administratif dan operasional pemeliharaan layanan, sehingga tim Anda dapat memiliki lebih banyak waktu dan berfokus pada inovasi.

Tinjau beban kerja Anda untuk mengidentifikasi komponen yang dapat digantikan oleh layanan terkelola AWS. Contoh, Amazon RDS, Amazon Redshift, dan Amazon ElastiCache memberikan layanan basis data terkelola. Amazon Athena, Amazon EMR, dan Amazon OpenSearch Service memberikan layanan analitik terkelola.

Langkah implementasi

1. Inventarisasikan beban kerja Anda untuk layanan dan komponen.
2. Nilai dan identifikasi komponen yang dapat digantikan oleh layanan terkelola. Berikut ini adalah beberapa contoh kapan Anda mungkin perlu mempertimbangkan penggunaan layanan terkelola:

Task	What to use on AWS
Hosting basis data	Gunakan instans Amazon Relational Database Service (Amazon RDS) terkelola dan bukannya memelihara instans Amazon RDS Anda sendiri di Amazon Elastic Compute Cloud (Amazon EC2).
Hosting beban kerja kontainer	Gunakan AWS Fargate, dan bukannya implementasi infrastruktur kontainer Anda sendiri.
Hosting aplikasi web	Gunakan AWS Amplify Hosting sebagai layanan hosting dan CI/CD terkelola penuh untuk situs web statis dan render aplikasi web sisi server.

3. Identifikasi dependensi dan buat rencana migrasi. Perbarui runbook dan playbook sesuai dengannya.
   - AWS Application Discovery Service secara otomatis mengumpulkan dan menyajikan informasi terperinci tentang dependensi dan pemanfaatan aplikasi untuk membantu Anda membuat keputusan yang lebih tepat saat merencanakan migrasi Anda.
4. Uji layanan sebelum migrasi ke layanan terkelola.
5. Gunakan rencana migrasi untuk mengganti layanan yang di-host mandiri dengan layanan terkelola.

6. Terus pantau layanan setelah migrasi selesai untuk membuat penyesuaian sebagaimana diperlukan dan optimalkan layanan.

Sumber daya

Dokumen terkait:

- Produk AWS Cloud
- Kalkulator Total Biaya Kepemilikan (TCO) AWS
- Amazon DocumentDB
- Amazon Elastic Kubernetes Service (EKS)
- Amazon Managed Streaming for Apache Kafka (Amazon MSK)

Video terkait:

- Operasi cloud dalam skala besar dengan AWS Managed Services

SUS05-BP04 Mengoptimalkan penggunaan akselerator komputasi berbasis perangkat keras

Optimalkan penggunaan instans komputasi terakselerasi Anda untuk mengurangi permintaan infrastruktur fisik beban kerja Anda.

Antipola umum:

- Anda tidak memantau penggunaan GPU.
- Anda menggunakan instans tujuan umum untuk beban kerja, padahal instans yang dibuat khusus dapat menghadirkan kinerja lebih tinggi, biaya lebih rendah, dan kinerja per watt yang lebih baik.
- Anda menggunakan akselerator komputasi berbasis perangkat keras untuk tugas yang akan lebih efisien jika menggunakan alternatif berbasis CPU.

Manfaat menjalankan praktik terbaik ini: Dengan mengoptimalkan penggunaan akselerator berbasis perangkat keras, Anda dapat mengurangi permintaan infrastruktur fisik untuk beban kerja Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Jika Anda memerlukan kemampuan pemrosesan tinggi, Anda dapat memanfaatkan instans komputasi terakselerasi, yang menyediakan akses ke akselerator komputasi berbasis perangkat keras seperti unit pemrosesan grafis (GPU) dan field programmable gate array (FPGA). Akselerator perangkat keras ini menjalankan fungsi-fungsi tertentu seperti pemrosesan grafis atau pencocokan pola data secara lebih efisien daripada alternatif berbasis CPU. Banyak beban kerja yang terakselerasi, seperti perenderan, transkode, dan machine learning, memiliki variabel tinggi sehubungan dengan penggunaan sumber daya. Jalankan perangkat keras ini hanya ketika diperlukan, dan nonaktifkan instans GPU secara otomatis saat tidak diperlukan, guna meminimalkan sumber daya yang digunakan.

Langkah implementasi

- Identifikasi instans komputasi terakselerasi mana yang dapat memenuhi kebutuhan Anda.
- Untuk beban kerja machine learning, manfaatkan perangkat keras yang dibuat khusus untuk beban kerja Anda, seperti AWS Trainium, AWS Inferentia, dan Amazon EC2 DL1. Instans AWS Inferentia seperti instans Inf2 menawarkan hingga 50% peningkatan kinerja per watt dibandingkan instans Amazon EC2 yang setara.
- Kumpulkan metrik penggunaan untuk instans komputasi terakselerasi Anda. Misalnya, Anda dapat menggunakan agen CloudWatch untuk mengumpulkan metrik seperti `utilization_gpu` dan `utilization_memory` untuk GPU Anda seperti yang ditunjukkan di Kumpulkan metrik GPU NVIDIA dengan Amazon CloudWatch.
- Optimalkan kode, operasi jaringan, dan pengaturan akselerator perangkat keras untuk memastikan perangkat keras yang mendasarinya dimanfaatkan sepenuhnya.
  - Optimalkan pengaturan GPU
  - Pemantauan dan Pengoptimalan GPU dalam AMI Deep Learning
  - Mengoptimalkan I/O untuk penyetelan kinerja GPU pelatihan deep learning di Amazon SageMaker AI
- Gunakan driver GPU dan pustaka berkinerja tinggi terbaru.
- Gunakan otomatisasi untuk melepaskan instans GPU ketika tidak digunakan.

Sumber daya

Dokumen terkait:

- Komputasi Dipercepat

- Mari Merancang! Merancang dengan chip dan akselerator kustom
- Bagaimana cara memilih jenis instans Amazon EC2 yang tepat untuk beban kerja saya?
- Instans VT1 Amazon EC2
- Pilih akselerator AI dan kompilasi model terbaik untuk inferensi penglihatan komputer dengan Amazon SageMaker AI

Video terkait:

- Cara memilih instans GPU Amazon EC2 untuk deep learning
- Melakukan Deployment Inferensi Deep Learning yang Hemat Biaya

Proses dan budaya

Pertanyaan

- SUS 6 Bagaimana proses organisasi Anda mendukung tujuan pelestarian lingkungan Anda?

SUS 6 Bagaimana proses organisasi Anda mendukung tujuan pelestarian lingkungan Anda?

Cari peluang untuk mengurangi dampak operasi Anda terhadap pelestarian lingkungan dengan membuat perubahan pada praktik deployment, pengujian, dan pengembangan.

Praktik terbaik

- SUS06-BP01 Mengadopsi metode yang dapat menghadirkan peningkatan pelestarian lingkungan dengan cepat
- SUS06-BP02 Selalu pastikan beban kerja Anda mutakhir
- SUS06-BP03 Meningkatkan pemanfaatan lingkungan build
- SUS06-BP04 Menggunakan device farm terkelola untuk pengujian

SUS06-BP01 Mengadopsi metode yang dapat menghadirkan peningkatan pelestarian lingkungan dengan cepat

Adopsi metode dan proses untuk memvalidasi potensi peningkatan, meminimalkan biaya pengujian, dan memberikan peningkatan kecil.

Antipola umum:

- Meninjau aplikasi Anda untuk pelestarian lingkungan adalah tugas yang dilakukan hanya satu kali pada awal proyek.
- Beban kerja Anda telah menjadi kedaluwarsa, karena proses rilis terlalu merepotkan guna melakukan perubahan kecil untuk efisiensi sumber daya.
- Anda tidak memiliki mekanisme untuk meningkatkan beban kerja untuk pelestarian lingkungan.

Manfaat menjalankan praktik terbaik ini: Dengan menetapkan proses untuk menghadirkan dan melacak peningkatan pelestarian lingkungan, Anda akan dapat terus mengadopsi fitur dan kemampuan baru, menghilangkan masalah, dan meningkatkan efisiensi beban kerja.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Sedang

Panduan implementasi

Uji dan validasi potensi peningkatan pelestarian lingkungan sebelum melakukan deployment peningkatan ini ke produksi. Pertimbangkan biaya pengujian saat menghitung potensi manfaat sebuah peningkatan untuk masa depan. Kembangkan metode pengujian berbiaya rendah untuk memberikan peningkatan kecil.

Langkah implementasi

- Tambahkan persyaratan pelestarian lingkungan ke backlog pengembangan Anda.
- Gunakan proses peningkatan berulang untuk mengidentifikasi, mengevaluasi, memprioritaskan, menguji, dan melakukan deployment peningkatan ini.
- Terus tingkatkan dan sederhanakan proses pengembangan Anda. Sebagai contoh, Otomatiskan proses pengiriman perangkat lunak Anda menggunakan pipeline integrasi dan pengiriman berkelanjutan (CI/CD) untuk menguji dan melakukan deployment potensi peningkatan untuk mengurangi tingkat upaya dan membatasi kesalahan yang disebabkan oleh proses manual.
- Kembangkan dan uji potensi peningkatan menggunakan komponen representatif yang dapat digunakan pada tingkat minimum untuk mengurangi biaya pengujian.
- Terus nilai dampak peningkatan dan buat penyesuaian jika diperlukan.

Sumber daya

Dokumen terkait:

- AWS memungkinkan solusi pelestarian lingkungan
- Praktik pengembangan tangkas yang dapat diskalakan berdasarkan AWS CodeCommit

Video terkait:

- Menghadirkan arsitektur pelestarian lingkungan dengan performa tinggi

Contoh terkait:

- Well-Architected Lab - Mengubah laporan biaya & penggunaan menjadi laporan efisiensi

SUS06-BP02 Selalu pastikan beban kerja Anda mutakhir

Selalu pastikan beban kerja Anda mutakhir untuk mengadopsi fitur yang efisien, menghilangkan masalah, dan meningkatkan efisiensi beban kerja Anda secara keseluruhan.

Antipola umum:

- Anda berasumsi bahwa arsitektur Anda saat ini statis dan tidak akan diperbarui seiring waktu.
- Anda tidak memiliki sistem atau koordinasi rutin untuk mengevaluasi apakah perangkat lunak dan paket yang diperbarui kompatibel dengan beban kerja Anda.

Manfaat menjalankan praktik terbaik ini: Dengan menetapkan proses untuk memastikan beban kerja Anda mutakhir, Anda dapat menerapkan fitur dan kemampuan baru, menyelesaikan masalah, dan meningkatkan efisiensi beban kerja.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Sistem operasi, runtime, perangkat lunak perantara (middleware), pustaka, dan aplikasi yang mutakhir dapat meningkatkan efisiensi beban kerja serta memudahkan pengadopsian teknologi yang lebih efisien. Perangkat lunak yang mutakhir juga dapat menyertakan fitur-fitur untuk mengukur dampak beban kerja terhadap pelestarian lingkungan secara lebih akurat, mengingat vendor juga menghadirkan fitur-fitur untuk memenuhi tujuan pelestarian lingkungan mereka sendiri. Secara teratur jaga agar beban kerja Anda mutakhir dengan fitur dan rilis terbaru.

Langkah implementasi

- Tentukan proses dan jadwal untuk mengevaluasi fitur atau instans baru untuk beban kerja Anda. Manfaatkan ketangkasan di cloud untuk menguji dengan cepat bagaimana fitur baru dapat meningkatkan beban kerja Anda untuk:
  - Mengurangi dampak pelestarian lingkungan.
  - Memperoleh efisiensi performa.
  - Menghilangkan penghalang untuk peningkatan terencana.
  - Meningkatkan kemampuan Anda dalam mengukur dan mengelola dampak terhadap pelestarian lingkungan.
- Buat inventaris perangkat lunak dan arsitektur beban kerja Anda dan identifikasi komponen yang perlu diperbarui.
  - Anda dapat menggunakan AWS Systems Manager Inventory untuk mengumpulkan metadata sistem operasi (OS), aplikasi, dan instans dari instans Amazon EC2 dan secara cepat memahami instans mana yang menjalankan perangkat lunak dan konfigurasi yang diperlukan oleh kebijakan perangkat lunak Anda dan instans mana yang perlu diperbarui.
- Pahami cara memperbarui komponen beban kerja Anda.

Workload component	How to update
Gambar mesin	Gunakan EC2 Image Builder untuk mengelola pembaruan Amazon Machine Image (AMI) untuk gambar server Linux atau Windows.
Gambar kontainer	Gunakan Amazon Elastic Container Registry (Amazon ECR) dengan pipeline Anda yang ada untuk mengelola Amazon Elastic Container Service (Amazon ECS) gambar.
AWS Lambda	AWS Lambda mencakup fitur manajemen versi.

- Gunakan otomatisasi untuk proses pembaruan guna mengurangi tingkat upaya dalam melakukan deployment fitur baru dan membatasi kesalahan yang disebabkan oleh proses manual.
  - Anda dapat menggunakan CI/CD untuk secara otomatis memperbarui AMI, gambar kontainer, dan artefak lain yang terkait dengan aplikasi cloud Anda.

- Anda dapat menggunakan alat seperti AWS Systems Manager Patch Manager untuk mengotomatiskan proses pembaruan sistem, dan menjadwalkan aktivitas menggunakan AWS Systems Manager Maintenance Windows.

Sumber daya

Dokumen terkait:

- Pusat Arsitektur AWS
- Yang Baru dengan AWS
- Alat Developer AWS

Contoh terkait:

- Well-Architected Labs - Manajemen Inventaris dan Patch
- Lab: AWS Systems Manager

SUS06-BP03 Meningkatkan pemanfaatan lingkungan build

Tingkatkan pemanfaatan sumber daya untuk mengembangkan, menguji, dan membangun beban kerja Anda.

Antipola umum:

- Anda secara manual menyediakan atau menghentikan lingkungan build Anda.
- Anda mempertahankan lingkungan build terus berjalan terlepas dari aktivitas pengujian, build, atau rilis (misalnya, menjalankan lingkungan di luar jam kerja anggota tim pengembangan Anda).
- Anda menyediakan terlalu banyak sumber daya untuk lingkungan build Anda.

Manfaat menjalankan praktik terbaik ini: Dengan meningkatkan pemanfaatan lingkungan build, Anda dapat meningkatkan efisiensi beban kerja cloud Anda secara keseluruhan sekaligus mengalokasikan sumber daya kepada para builder untuk mengembangkan, menguji, dan membangun secara efisien.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Gunakan otomatisasi dan infrastruktur sebagai kode untuk mengaktifkan lingkungan build saat diperlukan dan menonaktifkannya saat tidak digunakan. Hal yang umum dilakukan adalah menjadwalkan periode ketersediaan yang bertepatan dengan jam kerja anggota tim pengembangan. Lingkungan uji Anda harus sangat mirip dengan konfigurasi produksi. Tetapi, cari peluang untuk menggunakan jenis instans dengan kapasitas lonjakan, Instans Spot Amazon EC2, layanan basis data penskalaan otomatis, kontainer, dan teknologi nirserver untuk menyesuaikan pengembangan dan menguji kapasitas dengan penggunaan. Batasi volume data untuk tepat memenuhi persyaratan pengujian. Jika menggunakan data produksi dalam pengujian, jelajahi kemungkinan berbagi data dari produksi dan tidak memindahkan data ke mana-mana.

Langkah implementasi

- Gunakan infrastruktur sebagai kode untuk menyediakan lingkungan build Anda.
- Gunakan otomatisasi untuk mengelola siklus hidup pengembangan dan menguji lingkungan serta memaksimalkan efisiensi sumber daya build Anda.
- Gunakan strategi untuk memaksimalkan pemanfaatan lingkungan pengembangan dan pengujian.
  - Gunakan lingkungan representatif yang dapat digunakan pada tingkat minimum untuk mengembangkan dan menguji potensi peningkatan.
  - Gunakan teknologi nirserver jika mungkin.
  - Gunakan Instans Sesuai Permintaan untuk membantu perangkat developer Anda.
  - Gunakan jenis instans dengan kapasitas lonjakan, Instans Spot, dan teknologi lainnya untuk menyesuaikan kapasitas build dengan penggunaan.
  - Adopsi layanan cloud native untuk akses shell instans yang aman daripada melakukan deployment armada host bastion.
  - Skalakan secara otomatis sumber daya build Anda menurut tugas build.

Sumber daya

Dokumen terkait:

- Manajer Sesi Systems Manager AWS
- Instans performa burstable Amazon EC2
- Apa itu AWS CloudFormation?
- Apa itu AWS CodeBuild?

- Penjadwal Instans di AWS

Video terkait:

- Praktik Terbaik Integrasi Berkelanjutan

SUS06-BP04 Menggunakan device farm terkelola untuk pengujian

Gunakan device farm terkelola untuk secara efisien menguji fitur baru pada serangkaian perangkat keras representatif.

Antipola umum:

- Anda menguji dan melakukan deployment aplikasi di masing-masing perangkat fisik secara manual.
- Anda tidak menggunakan layanan pengujian aplikasi untuk menguji dan berinteraksi dengan aplikasi Anda (contohnya, Android, iOS, dan aplikasi web) pada perangkat fisik nyata.

Manfaat menjalankan praktik terbaik ini: Menggunakan device farm terkelola untuk menguji aplikasi yang didukung cloud memberikan sejumlah manfaat:

- Device farm disertai fitur yang lebih efisien untuk menguji aplikasi di berbagai macam perangkat.
- Device farm terkelola menghilangkan kebutuhan akan infrastruktur internal untuk pengujian.
- Device farm menawarkan berbagai macam jenis perangkat, termasuk perangkat keras yang lebih lama dan kurang populer, yang menghilangkan kebutuhan akan pemutakhiran perangkat yang tidak perlu.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak dijalankan: Rendah

Panduan implementasi

Menggunakan device farm terkelola dapat membantu Anda menyederhanakan proses pengujian untuk fitur baru di serangkaian perangkat keras representatif. Device farm terkelola menawarkan berbagai jenis perangkat, termasuk perangkat keras yang lebih lama dan kurang populer, serta menghindari dampak pelestarian lingkungan pelanggan akibat pemutakhiran perangkat yang tidak perlu.

Langkah implementasi

- Tetapkan persyaratan pengujian Anda dan rencanakan (seperti jenis pengujian, sistem operasi, dan jadwal pengujian).
  - Anda dapat menggunakan Amazon CloudWatch RUM untuk mengumpulkan dan menganalisis data di sisi klien dan membentuk rencana pengujian Anda.
- Pilih device farm terkelola yang dapat mendukung persyaratan pengujian Anda. Contohnya, Anda dapat menggunakan AWS Device Farm untuk menguji dan memahami dampak perubahan Anda pada serangkaian perangkat keras representatif.
- Gunakan integrasi berkelanjutan/deployment berkelanjutan (CI/CD) untuk menjadwalkan dan menjalankan pengujian Anda.
  - Mengintegrasikan AWS Device Farm dengan pipeline CI/CD Anda untuk menjalankan uji Selenium di semua browser
  - Membangun dan menguji aplikasi iOS dan iPadOS dengan DevOps AWS dan layanan mobile
- Terus tinjau hasil pengujian Anda dan buat peningkatan yang perlu.

Sumber daya

Dokumen terkait:

- Daftar perangkat AWS Device Farm
- Melihat dasbor RUM CloudWatch

Contoh terkait:

- Contoh Aplikasi AWS Device Farm untuk Android
- Contoh Aplikasi AWS Device Farm untuk iOS
- Uji Web Appium untuk AWS Device Farm

Video terkait:

- Mengoptimalkan aplikasi melalui wawasan pengguna akhir dengan Amazon CloudWatch RUM

Pemberitahuan

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

Hak Cipta © 2021, Amazon Web Services, Inc. atau afiliasinya.