AWS Garis Dasar Ketahanan Startup

AWS Panduan Preskriptif

AWS Panduan Preskriptif: AWS Garis Dasar Ketahanan Startup

Table of Contents

AWS Garis Dasar Ketahanan Startup

Amazon Web Services (kontributor)

Maret 2026 (sejarah dokumen)

Setiap pendiri startup tahu ketegangannya: tim berlomba untuk mengirimkan fitur baru sementara pelanggan mengharapkan setiap rilis hanya berfungsi. Anda bangun untuk pemberitahuan tentang pemadaman tadi malam, keluhan pelanggan mengalir masuk, dan peta jalan Anda sudah dikemas dengan fitur yang dijanjikan. Ini adalah kenyataan sehari-hari bagi startup — menyeimbangkan tekanan untuk berinovasi dengan kebutuhan untuk mempertahankan layanan yang stabil.

Naluri untuk memprioritaskan fitur baru adalah wajar. Investor menginginkan pertumbuhan, pelanggan meminta lebih banyak kemampuan, dan pesaing tidak tinggal diam. Namun pengalaman menunjukkan bahwa masalah ketahanan dapat menggagalkan bahkan startup yang paling inovatif sekalipun. Gangguan layanan tidak hanya berarti kehilangan pendapatan; itu mengikis kepercayaan yang telah Anda kerjakan dengan sangat keras untuk membangun dengan pelanggan Anda. Faktanya, keandalan yang kokoh dapat menjadi pembeda diam Anda di pasar yang ramai.

Membangun ketahanan ke dalam startup Anda tidak berarti melambat atau investasi infrastruktur besar-besaran. Ini tentang membuat pilihan cerdas sejak dini yang mencegah masalah di kemudian hari. Anggap saja seperti membangun rumah. Jauh lebih mudah untuk meletakkan fondasi yang kokoh di awal daripada memperbaiki masalah struktural setelah semuanya dibangun. Dengan menenun praktik tangguh ke dalam arsitektur dan operasi awal Anda, Anda menciptakan keunggulan kompetitif. Anda tidak hanya memberikan produk; Anda memberikan pengalaman yang dapat diandalkan yang membangun loyalitas pelanggan.

AWS Startup Resiliency Baseline (AWS SRB) dibuat khusus untuk tim yang menghadapi tantangan ini. Ini menyediakan jalur praktis untuk membangun keandalan ke dalam sistem Anda sambil mempertahankan kecepatan yang membuat startup istimewa. Tidak ada proses berat atau kompleksitas skala perusahaan, hanya pola praktis yang tumbuh dengan bisnis Anda.

Panduan ini menjelaskan cara mengimplementasikan AWS SRB di lingkungan startup. Ini membantu Anda mengidentifikasi apa yang benar-benar penting untuk ketahanan, di mana memfokuskan sumber daya Anda yang terbatas, dan bagaimana membangun praktik yang berskala seiring pertumbuhan perusahaan Anda.

Lingkup bimbingan

Saat menjelajahi langkah-langkah praktis membangun ketahanan startup, penting untuk memahami di mana panduan ini sesuai dengan lintasan pertumbuhan organisasi Anda yang lebih luas. Garis Dasar Ketahanan AWS Startup diselaraskan dengan kerangka siklus hidup ketahanan. Ini berfungsi sebagai peta jalan awal, yang dirancang untuk membantu Anda menerapkan langkah-langkah ketahanan mendasar dengan overhead pengembangan minimal saat Anda menerapkan aplikasi pertama Anda. AWS Anggap saja sebagai starter kit Anda untuk membangun sistem andal yang dapat pulih secara efektif dari gangguan.

The AWS Well-Architected Framework dan panduan ketahanan ini saling melengkapi untuk membantu Anda membangun infrastruktur yang aman dan andal. Panduan ini menyelami lebih dalam praktik ketahanan sementara AWS Well-Architected Framework menyediakan serangkaian praktik terbaik arsitektur yang lebih luas. Anda dapat menilai arsitektur Anda terhadap praktik-praktik ini menggunakan AWS Well-Architected Toolin your Akun AWS.

Model tanggung jawab bersama

Sangat penting untuk mengklarifikasi aspek penting dalam membangun sistem tangguh di cloud: kemitraan antara AWS dan startup Anda. Ketahanan di cloud beroperasi pada model tanggung jawab bersama, dengan AWS dan tim Anda masing-masing memainkan peran berbeda dalam memastikan ketahanan sistem.

AWS bertanggung jawab atas ketahanan infrastruktur cloud yang mendasari yang mendukung layanan mereka. Anggap ini sebagai fondasi di mana Anda membangun aplikasi Anda. Ini termasuk menjaga ketahanan pusat data, komponen jaringan, dan inti yang Layanan AWS Anda gunakan dalam arsitektur Anda.

Tanggung jawab startup Anda berfokus pada ketahanan sistem yang Anda bangun dan terapkan di atas fondasi ini. Rekomendasi dalam panduan ini termasuk dalam domain tanggung jawab Anda. Dengan menerapkan praktik ini secara serius, Anda dapat membuat aplikasi yang menggunakan AWS infrastruktur yang andal dan menggabungkan kemampuan pemulihan yang kuat.

Model ini berarti Anda tidak pernah membangun ketahanan secara terpisah. Anda membangun fondasi yang terbukti dan andal sambil memfokuskan upaya Anda pada aspek-aspek yang secara langsung berdampak pada pelanggan dan operasi bisnis Anda. Panduan ini mengeksplorasi cara memaksimalkan model tanggung jawab bersama ini, menggunakan AWS infrastruktur yang andal

sambil menerapkan langkah-langkah praktis untuk memastikan bahwa aplikasi Anda pulih dengan anggun dari gangguan.

Tahap 1: Tetapkan tujuan

Bayangkan tim Anda berada di sprint terakhir sebelum peluncuran produk utama. Fitur-fitur baru ini merupakan terobosan, dan kegembiraan investor sedang membangun. Kemudian, selama penerapan rutin, layanan inti Anda turun. Ketika keluhan pelanggan membanjiri email Anda, dua pertanyaan menjadi sangat jelas: Berapa lama Anda bisa offline? Data apa yang bisa Anda hilangkan?

Berharap semuanya akan bekerja dengan baik bukanlah strategi yang baik. Anda memerlukan cara sistematis untuk memutuskan di mana ketahanan paling penting dan di mana tidak. Di sinilah analisis dampak bisnis (BIA) menjadi kritis. Ini membantu Anda membuat keputusan berdasarkan informasi tentang di mana berinvestasi dalam ketahanan. BIA membantu Anda memahami bagian mana dari sistem Anda yang benar-benar membutuhkan keandalan yang kokoh dan mana yang dapat mentolerir beberapa fleksibilitas.

Mulailah dengan memetakan perjalanan pengguna inti Anda. Untuk masing-masing, tanyakan pada diri Anda hal-hal berikut:

- Apa dampaknya jika ini terganggu?
- Seberapa cepat kita harus memulihkan layanan?
- Data apa yang penting untuk dilindungi?

Ini bukan hanya latihan teknis; ini membantu Anda memahami dampak bisnis dari masalah keandalan. Kehilangan pendapatan hanyalah permulaan. Pertimbangkan bagaimana pemadaman dapat mengikis kepercayaan pelanggan, melanggar persyaratan peraturan, atau memberi keunggulan kepada pesaing.

Dari analisis ini, Anda akan memperoleh dua angka penting untuk setiap perjalanan pengguna: tujuan waktu pemulihan (RTO) dan tujuan titik pemulihan (RPO). RTO mendefinisikan seberapa cepat Anda harus memulihkan perjalanan itu. RPO mendefinisikan berapa banyak kehilangan data yang dapat ditoleransi pelanggan Anda. Target yang digerakkan oleh bisnis ini kemudian memandu komponen mana yang Anda pilih dan bagaimana Anda merancangnya, tanpa rekayasa berlebihan setiap bagian dari sistem.

Keindahan dari pendekatan ini adalah membantu Anda memfokuskan sumber daya terbatas Anda di tempat yang paling penting. Mungkin pemrosesan transaksi inti Anda membutuhkan pemulihan yang hampir instan dan nol kehilangan data, tetapi mesin rekomendasi Anda dapat mentolerir waktu

henti yang lebih lama. Dengan menetapkan tujuan yang jelas, Anda membuat kerangka kerja yang memungkinkan Anda melanjutkan pengembangan fitur yang cepat sambil membangun ketahanan secara strategis.

Dokumentasikan tujuan-tujuan ini dengan jelas. Mereka tidak hanya untuk tim teknik Anda. Ketika Anda melempar ke pelanggan bisnis atau melalui uji tuntas teknis dengan investor, dokumentasi ini menunjukkan bahwa Anda telah berpikir kritis tentang kelangsungan bisnis.

Target ini berkembang seiring pertumbuhan startup Anda. Kebutuhan ketahanan ribuan pengguna pertama Anda berbeda dari kebutuhan klien bisnis pertama Anda. Mulailah dengan tujuan yang dapat Anda penuhi secara realistis hari ini, tetapi rencanakan bagaimana mereka akan memperketat saat Anda menskalakan.

Panduan ini mengeksplorasi bagaimana menerapkan langkah-langkah ketahanan yang memenuhi tujuan ini. Menetapkan target ini adalah langkah pertama Anda yang penting. Mereka adalah kompas Anda untuk menavigasi ketegangan konstan antara inovasi dan stabilitas, membantu Anda membangun sistem yang secara andal memberikan nilai kepada pelanggan Anda.

Tahap 2: Desain dan implementasi

Bagian ini membahas mengubah tujuan ketahanan Anda menjadi kenyataan. Anda telah memetakan apa yang paling penting bagi bisnis Anda, dan sekarang saatnya untuk membangunnya. Bagaimana Anda membangun ketahanan tanpa memperlambat inovasi?

Pikirkan layanan yang AWS dikelola sebagai jalan pintas ketahanan. Alih-alih membakar jam teknik yang berharga untuk memelihara infrastruktur, gunakan layanan yang menangani redundansi untuk Anda. Misalnya, pertimbangkan Amazon Simple Storage Service (Amazon S3). Ini secara otomatis menyimpan beberapa salinan data Anda dalam a Wilayah AWS untuk daya tahan. Itu tidak memerlukan kode tambahan atau tugas pager larut malam.

Bagaimana dengan komponen aplikasi inti Anda? Pilihan cerdas dapat melipatgandakan dampak tim Anda. Pertimbangkan database yang merupakan tulang punggung layanan Anda. Alih-alih membangun sistem replikasi Anda sendiri, pertimbangkan untuk menggunakan Amazon Aurora, yang secara otomatis menangani failover. Fitur-fitur ini mungkin lebih mahal, tetapi mereka mengalihkan fokus tim Anda dari memelihara infrastruktur untuk memecahkan masalah bisnis. Biaya ini dapat diimbangi melalui pengiriman fitur yang lebih cepat dan menghindari kehilangan pendapatan selama pemadaman.

Terkadang startup perlu membangun solusi khusus. Itulah sifat dari startup yang inovatif. Ketika Anda melakukannya, tetap sederhana namun cerdas. Sebarkan aplikasi Anda di beberapa Availability Zone dengan menggunakan Elastic Load Balancing dan grup Amazon EC2 Auto Scaling. Tetapkan kapasitas minimum grup Auto Scaling untuk menangani lalu lintas dasar Anda meskipun salah satu Availability Zone gagal. Ini memberikan ketahanan terhadap kegagalan lokal tanpa pola arsitektur yang kompleks. Ketika startup Anda tumbuh dan pelanggan menuntut ketahanan yang lebih tinggi, Anda dapat berevolusi ke pendekatan yang lebih canggih.

Kami menyarankan Anda menjaga lingkungan produksi dan pengembangan Anda secara terpisah Akun AWS. Sangat menggoda untuk mencampurnya saat Anda bergerak cepat, tetapi batas ini adalah jaring pengaman Anda. Ini mencegah eksperimen yang bermaksud baik dari menjatuhkan layanan produksi Anda. Anggap saja sebagai asuransi untuk budaya pengembangan “bergerak cepat dan hancurkan sesuatu” Anda - hancurkan hal-hal dalam pengembangan, jaga produksi tetap stabil.

Jika aplikasi Anda bergantung pada layanan pihak ketiga, rencanakan kegagalannya. Ketika prosesor pembayaran Anda mengalami masalah, dapatkah sistem Anda menanganinya dengan anggun? Bangun opsi pemutus sirkuit dan fallback sederhana. Mungkin mengantri transaksi tersebut

alih-alih menampilkan pesan kesalahan. Pelanggan Anda akan menghargai bahwa Anda membuat semuanya berfungsi, meskipun tidak sempurna.

Dokumentasikan saat Anda membangun, tetapi tetap praktis. Fokus pada merekam alasan di balik keputusan penting, dan buat buku pedoman pemulihan sederhana. Sangat penting untuk menyiapkan ini ketika insiden terjadi.

Anda tidak membangun untuk ketahanan yang sempurna; Anda sedang membangun untuk ketahanan yang tepat. Setiap jam yang dihabiskan untuk ketahanan over-engineering adalah satu jam yang tidak dihabiskan untuk fitur yang diminta pelanggan. Gunakan layanan AWS terkelola sebagai fondasi Anda, tambahkan ketahanan yang ditargetkan di tempat yang paling penting, dan buat jalur yang jelas untuk meningkatkan ketahanan seiring pertumbuhan bisnis Anda.

Bab berikutnya membahas bagaimana memvalidasi pilihan desain ini tanpa membakar sumber daya teknik. Untuk startup, pengujian harus menjadi dorongan yang masuk akal dan investasi cerdas dalam ketahanan aplikasi Anda.

Tahap 3: Evaluasi dan uji

Anda telah membangun fondasi yang tangguh, tetapi bagaimana Anda tahu itu benar-benar bekerja? Menguji ketahanan mungkin terdengar seperti kemewahan saat Anda berlomba untuk membuktikan kecocokan pasar produk. Namun, ada cara cerdas untuk melakukan ini tanpa menggagalkan pengembangan fitur Anda. Bab ini menjelaskan pengujian ramping dan praktis yang sesuai dengan kecepatan startup.

Mulailah dengan AWS Resilience Hub, dan anggap itu sebagai alat penilaian arsitektur awal. Ini memberikan tinjauan dasar yang bermanfaat tentang fondasi ketahanan arsitektur. Ini membantu Anda mengevaluasi apakah pengaturan infrastruktur dasar selaras dengan tujuan pemulihan Anda dengan memeriksa pola konfigurasi umum dan potensi titik kegagalan tunggal. Ini dapat menandai celah yang jelas, seperti kehilangan beberapa konfigurasi Availability Zone atau kebijakan pencadangan yang tidak lengkap. Resilience Hub melengkapi, tetapi tidak menggantikan, tinjauan arsitektur yang bijaksana dan pengujian jalur kritis yang ditargetkan.

Untuk memvalidasi tujuan pemulihan terdokumentasi Anda, jadwalkan tes pemulihan bulanan AWS Backupdi lingkungan pengembangan Anda. Meskipun membutuhkan waktu rekayasa, mungkin lebih murah daripada menemukan cadangan Anda tidak berfungsi selama insiden nyata. Jadikan itu bagian dari siklus pengembangan reguler Anda, seperti menjalankan pengujian unit atau ulasan kode. Tujuannya bukan kesempurnaan; itu keyakinan bahwa Anda dapat pulih ketika Anda membutuhkannya.

Ketika startup Anda tumbuh dan pelanggan mulai bergantung pada Anda lebih berat, secara bertahap tingkatkan permainan pengujian Anda. Saat Anda menerapkan fitur baru, sertakan pemeriksaan ketahanan dasar dalam pipeline Anda. Coba eksperimen chaos sederhana dengan menggunakan AWS Fault Injection Service. Mulailah di lingkungan praproduksi Anda dan mulailah dari yang kecil. Uji bagaimana aplikasi Anda menangani respons API yang tertunda dalam pengembangan sebelum mempertimbangkan eksperimen produksi apa pun. Saat kepercayaan diri Anda tumbuh, perluas tes ini secara bertahap, tetapi selalu validasi dalam praproduksi terlebih dahulu. Untuk startup, melanggar sesuatu dalam produksi cukup berisiko tanpa melakukannya dengan sengaja.

Kuncinya adalah keseimbangan. Setiap jam yang dihabiskan untuk pengujian adalah satu jam yang tidak dihabiskan untuk membangun fitur baru. Tetapi beberapa tes strategis dapat mencegah jenis pemadaman yang kehilangan kepercayaan pelanggan. Gunakan alat otomatis yang disediakan oleh AWS untuk melakukan angkat berat, dan fokus pada pengujian yang paling penting bagi

pelanggan Anda. Ini membantu Anda membangun kepercayaan pada ketahanan aplikasi Anda tanpa memperlambat inovasi.

Bab berikutnya mengeksplorasi bagaimana mengembangkan fondasi ini sebagai skala startup Anda.

Tahap 4: Beroperasi

Anda telah membangun aplikasi yang tangguh dan mengujinya. Sekarang realitas sehari-hari menjaganya tetap berjalan. Tetapi dalam startup, Anda tidak dapat menonton semua operasi, dan Anda tidak boleh mencobanya. Kuncinya adalah tetap waspada terhadap apa yang penting tanpa memberikan terlalu banyak metrik atau membebani tim Anda.

Mulailah dengan perspektif pelanggan. Burung kenari Amazon CloudWatch Synthetics bertindak sebagai pelanggan otomatis. Mereka terus menguji perjalanan pengguna yang kritis. Mintalah mereka masuk, simulasikan pembelian menggunakan akun uji, atau akses fitur-fitur utama, terutama selama jam-jam tersibuk Anda. Ini membantu Anda memahami pengalaman pelanggan dan membantu Anda menangkap masalah sebelum pengguna nyata melakukannya. Ketika kenari gagal, Anda langsung tahu bahwa ada sesuatu yang salah dari sudut pandang pelanggan.

Bangun di atas fondasi ini dengan pemantauan terfokus pada infrastruktur pendukung. Sinyal apa yang memberi tahu Anda bahwa ada masalah? Amazon CloudWatch membantu Anda membangun dasbor yang melacak tanda-tanda ini. Jangan hanya memantau metrik teknis; ikat mereka dengan dampak bisnis. Misalnya, penggunaan CPU yang tinggi penting, tetapi itu karena itu mungkin menurunkan pengalaman pelanggan yang Anda lacak dengan kenari.

Sebagai pendekatan praktis, petakan pemantauan Anda ke perjalanan pelanggan Anda. Jika Anda menjalankan platform perangkat lunak sebagai layanan (SaaS), Anda mungkin peduli dengan waktu respons API, tingkat keberhasilan otentikasi, dan ketersediaan fitur inti. Siapkan peringatan yang memberi tahu Anda saat metrik ini melayang. Namun, bersikaplah selektif. Setiap peringatan harus menuntut tindakan. Jika tim Anda mulai mengabaikan peringatan karena “mungkin bukan apa-apa,” Anda telah menyetel terlalu banyak atau melacak metrik yang salah.

Rutekan peringatan ini melalui alat yang sudah digunakan tim Anda. Jika teknisi Anda tinggal di aplikasi perpesanan tertentu, kirim peringatan di sana. Tujuannya adalah kesadaran cepat tanpa menciptakan proses baru. Ketika peringatan menyala, tim Anda harus tahu persis apa artinya dan apa yang harus dilakukan.

Jaga agar dokumentasi operasional Anda ramping dan praktis. Simpan runbook dengan kode Anda di kontrol versi, tetapi ingat bahwa itu bukan novel. Ketika sesuatu rusak, tim Anda membutuhkan langkah yang jelas dan dapat ditindaklanjuti. Setiap peringatan harus ditautkan ke runbook yang sesuai, dan setiap runbook harus menjawab tiga pertanyaan:

- Apa yang pecah?

- Mengapa ini penting?
- Bagaimana cara memperbaikinya?

Menerapkan proses manajemen insiden sederhana. Anda tidak memerlukan kerangka kerja yang rumit, cukup definisi yang jelas tentang apa yang merupakan insiden dan siapa yang harus dihubungi ketika segala sesuatunya meningkat. Simpan log insiden karena membantu Anda meningkatkan ketahanan aplikasi Anda.

Kuncinya adalah menemukan sweet spot antara kewaspadaan dan overhead. Gunakan AWS alat untuk mengotomatiskan apa yang Anda bisa, fokus pada metrik pemantauan yang memengaruhi pelanggan, dan menjaga proses Anda cukup ringan untuk berkembang seiring pertumbuhan Anda.

Bab berikutnya mengeksplorasi bagaimana menumbuhkan pola pikir ketahanan tanpa mengorbankan kecepatan dan inovasi yang membuat startup istimewa. Pada akhirnya, ketahanan adalah tentang orang-orang seperti halnya tentang teknologi.

Tahap 5: Menanggapi dan belajar

Saat Anda menjalankan startup, proses post-mortem yang kompleks dapat memperlambat tim Anda. Bab ini mengeksplorasi bagaimana belajar dari insiden tanpa mengubahnya menjadi latihan birokrasi.

Integrasikan pembelajaran insiden ke dalam ritme Anda yang ada. Jika tim Anda sudah mengadakan pertemuan rutin, gunakan sepuluh menit untuk membahas insiden baru-baru ini. Fokus pada pertanyaan praktis, seperti:

- Apakah runbook membantu?
- Apakah peringatan terjadi pada waktu yang tepat?
- Bisakah layanan AWS terkelola mencegah hal ini?

Tetap fokus pada tindakan, bukan menyalahkan. Dalam sebuah startup, Anda tidak membangun sistem yang sempurna; Anda sedang membangun satu yang menjadi lebih baik setiap kali terjadi kesalahan.

Anda dapat menggunakan sistem tiket Anda untuk melacak insiden; tidak perlu alat khusus. Buat template sederhana yang mencakup timeline insiden, dampak pelanggan, langkah pemulihan yang diambil, dan pelajaran yang dipetik. Kamera ini menjadi memori institusional jika Anda menggunakannya secara aktif. Tinjau insiden masa lalu selama orientasi untuk mempercepat insinyur baru. Referensi mereka dalam ulasan arsitektur saat merancang sistem serupa. Tarik mereka ke hari permainan untuk membuat skenario kegagalan realistis berdasarkan peristiwa aktual. Template menangkap apa yang terjadi, dan penggunaan reguler mengubahnya menjadi pembelajaran organisasi.

Saat startup tumbuh, pola muncul. Mungkin komponen tertentu gagal lebih sering, atau mungkin jenis perubahan tertentu menyebabkan masalah. Gunakan pola-pola ini untuk memandu investasi ketahanan. Jika kegagalan database menyebabkan masalah, pertimbangkan untuk meningkatkan beberapa pengaturan Availability Zone Anda. Jika gangguan layanan pihak ketiga adalah tema umum, pertimbangkan untuk meningkatkan pemutus sirkuit.

Tujuannya bukan untuk mencegah setiap kegagalan yang mungkin terjadi. Itu tidak mungkin dan akan memperlambat Anda terlalu banyak. Tujuannya adalah untuk belajar dengan cepat, beradaptasi dengan cepat, dan menjaga aplikasi cukup andal saat Anda berkembang pesat. Gunakan setiap insiden sebagai kesempatan untuk membuat sistem Anda sedikit lebih tangguh, tim Anda sedikit lebih berpengetahuan, dan pelanggan Anda sedikit lebih percaya diri dalam layanan Anda. Untuk startup,

kecepatan dan pembelajaran mengalahkan kesempurnaan. Buat proses ringan yang membantu Anda belajar dari insiden tanpa memperlambat inovasi. Praktik ketahanan terbaik adalah yang benar-benar digunakan tim Anda.

Langkah selanjutnya untuk startup

Ingat penyebaran larut malam yang membuat jantung Anda berdebar kencang? Atau saat itu ketika pelanggan utama bertanya tentang langkah-langkah ketahanan Anda? Perjalanan ketahanan startup bukan tentang menghilangkan momen-momen ini — ini tentang menghadapinya dengan percaya diri.

Panduan ini menjelaskan jalur praktis menuju ketahanan, termasuk: menetapkan target pemulihan yang realistis, menggunakan layanan AWS terkelola, menerapkan pengujian lean, memantau apa yang penting, dan belajar dari insiden. Pendekatan ini sangat kuat untuk startup karena tumbuh bersama Anda. Kerangka kerja yang sama yang membantu startup Anda pulih dari masalah saat ini membangun fondasi untuk melayani pelanggan perusahaan besok.

Pikirkan ketahanan seperti peta jalan produk; Anda tidak membangun semuanya sekaligus. Mulailah dengan melindungi layanan inti Anda. Tambahkan pemantauan untuk perjalanan pelanggan yang kritis. Menerapkan pengujian dasar yang menangkap masalah yang jelas. Dokumentasikan apa yang Anda pelajari. Setiap langkah dapat dikelola dan praktis, dan yang paling penting, mereka tidak menghentikan Anda dari fitur pengiriman.

Ketika startup tumbuh, kebutuhan ketahanan berkembang. Pelanggan bisnis mungkin mengajukan pertanyaan yang lebih sulit. Lalu lintas puncak dapat mencapai titik tertinggi baru. Ekspansi internasional membawa tantangan baru. Karena Anda telah membangun ketahanan ke dalam fondasi Anda, Anda siap untuk beradaptasi.

Tujuan untuk startup bukanlah uptime yang sempurna atau nol insiden. Tujuannya adalah membangun sistem yang cukup andal untuk mendapatkan kepercayaan pelanggan sambil mempertahankan kecepatan yang membuat startup istimewa. Dengan mengikuti pendekatan ini, Anda dapat menciptakan sesuatu yang lebih berharga daripada menciptakan sistem yang sempurna; Anda dapat membuat satu yang menjadi lebih baik setiap hari dengan belajar dari setiap tantangan sambil mengikuti ambisi startup Anda.

Perjalanan ketahanan tidak berakhir. Ini berkembang dengan setiap fitur yang Anda kirimkan, setiap pelanggan yang Anda menangkan, dan setiap insiden yang Anda tangani. Panduan ini membantu Anda membangun kerangka kerja untuk membantu Anda membuat langkah maju yang percaya diri dan praktis. Pada akhirnya, kesuksesan startup bukan tentang memilih antara ketahanan dan kecepatan. Ini tentang menemukan cara cerdas untuk memberikan keduanya.

Sumber daya

AWS Kerangka Well-Architected

- Pilar Keandalan
  - REL05- BP01 Menerapkan degradasi anggun untuk mengubah dependensi keras yang berlaku menjadi dependensi lunak
  - REL06- BP01 Pantau semua komponen untuk beban kerja
  - REL06- BP03 Kirim pemberitahuan (Pemrosesan waktu nyata dan mengkhawatirkan)
  - REL08- BP03 Integrasikan pengujian ketahanan sebagai bagian dari penerapan Anda
- Model Tanggung Jawab Bersama untuk Ketahanan

AWS Bimbingan Preskriptif

- Pendekatan Backup dan Recovery pada AWS
- Kerangka siklus hidup ketahanan: Pendekatan berkelanjutan untuk peningkatan ketahanan

AWS Sumber daya lainnya

- AWS Batas Isolasi Kesalahan (AWS Whitepapers)
- Menetapkan Target RPO dan RTO untuk Aplikasi Cloud (AWS posting blog)
- Ikhtisar Opsi Penerapan di AWS (AWS Whitepaper)

Kontributor

Individu berikut berkontribusi pada panduan ini.

Mengotorisasi

- Dylan McCarroll, Arsitek Solusi Asosiasi, AWS
- Igor Fil, Arsitek Solusi Startup, AWS
- Parth Shah, Arsitek Solusi Senior, AWS
- Vrajesh Prajapati, Arsitek Solusi, AWS

Meninjau

- Clark Richey, Teknolog Utama, AWS
- Bruno Emer, Ahli Teknologi Utama, AWS

Penulisan teknis

- Lilly AbouHarb, Penulis Teknis Senior, AWS

Riwayat dokumen

Tabel berikut menjelaskan perubahan signifikan pada panduan ini. Jika Anda ingin diberi tahu tentang pembaruan masa depan, Anda dapat berlangganan umpan RSS.

Perubahan	Deskripsi	Tanggal
Pembaruan yang signifikan	Rekomendasi dan penggunaan yang diperbarui Layanan AWS di seluruh.	Maret 6, 2026
Publikasi awal	—	Januari 24, 2024

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.