Meningkatkan ketahanan dan meningkatkan pengalaman pelanggan dengan menggunakan chaos engineering AWS

AWS Panduan Preskriptif

AWS Panduan Preskriptif: Meningkatkan ketahanan dan meningkatkan pengalaman pelanggan dengan menggunakan chaos engineering AWS

Table of Contents

Meningkatkan ketahanan dan meningkatkan pengalaman pelanggan dengan menggunakan chaos engineering pada AWS

Laurent Domb, Kepala Teknolog, Keuangan Federal, Amazon Web Services

April 2025 (riwayat dokumen)

Chaos engineering adalah disiplin bereksperimen pada aplikasi untuk membangun kepercayaan pada kemampuan organisasi dan aplikasi Anda untuk menahan kondisi yang bergejolak dalam produksi. Ini adalah pendekatan proaktif terhadap ketahanan, dengan tujuan untuk memverifikasi apakah aplikasi dan organisasi Anda mampu menyerap, beradaptasi, dan akhirnya pulih dari gangguan layanan dengan memperkenalkan kegagalan terkontrol di seluruh orang, proses, dan teknologi. Tujuannya juga untuk mengidentifikasi dan menghilangkan kelemahan sebelum dapat menyebabkan pemadaman atau gangguan lain dalam produksi.

Di Amazon, kami memahami bahwa kegagalan tidak dapat dihindari dalam sistem terdistribusi, sampai-sampai berfungsi meskipun ada kegagalan adalah mode operasi normal. Karena interaksi antar layanan pasti akan gagal, Anda perlu memahami bagaimana layanan Anda bereaksi selama berbagai mode kegagalan dan membangun layanan yang tahan terhadap kerentanan utama seperti kegagalan ketergantungan, badai coba lagi, Zona Ketersediaan yang terganggu, dan kehabisan sumber daya host.

Mari kita ambil contoh badai coba lagi. Kegagalan lokal pada klien dapat berdampak pada beberapa layanan secara signifikan. Ini biasa disebut sebagai efek kupu-kupu. Badai coba lagi adalah manifestasi dari efek kupu-kupu di mana ketergantungan yang gagal memicu klien, dan klien klien tersebut, untuk mencoba kembali operasi yang gagal, yang mengarah pada pertumbuhan lalu lintas yang eksponensial. Layanan menjadi kelebihan beban karena mereka harus menanggapi lalu lintas reguler selain mencoba lagi lalu lintas sambil menangani penurunan kinerja.

Rekayasa kekacauan telah muncul sebagai respons terhadap meningkatnya kompleksitas sistem terdistribusi. Ini adalah pendekatan multidisiplin yang menggabungkan prinsip-prinsip dari teori kekacauan, pemikiran sistem, dan rekayasa untuk merancang dan mengelola sistem kompleks yang tahan terhadap peristiwa dan perilaku tak terduga. Pada intinya, rekayasa kekacauan berkaitan dengan pemahaman dan pengelolaan perilaku sistem yang kompleks dalam kondisi ketidakpastian dan ketidakpastian. Ia mengakui bahwa pendekatan tradisional untuk rekayasa, yang bergantung

pada memprediksi dan mengendalikan hasil, seringkali tidak cukup untuk berurusan dengan sifat kompleks dan dinamis dari sistem terdistribusi. Ketika sistem ini tumbuh, mereka sering melampaui ruang lingkup pemahaman setiap individu.

Chaos engineering menyediakan konsep, teknik, dan alat untuk sengaja menyuntikkan kegagalan ke dalam sistem untuk mengungkap kelemahan sebelum mereka terwujud dalam produksi. Pendekatan proaktif ini memungkinkan organisasi untuk membangun kepercayaan bahwa sistem mereka akan bekerja dalam kondisi stres. Meskipun rekayasa kekacauan masih merupakan praktik yang berkembang, ini merupakan perubahan mendasar menuju merancang, mengelola, dan mengoperasikan sistem komputasi modern agar tangguh dalam menghadapi peningkatan kompleksitas dan keterkaitan.

Bagian berikut dari panduan ini membahas manfaat rekayasa kekacauan, menjelaskan cara melakukan eksperimen rekayasa kekacauan, dan menjelaskan pendekatan yang dapat Anda ambil untuk menerapkan rekayasa kekacauan dalam skala besar di organisasi Anda. Juga termasuk contoh perencanaan eksperimen dan dokumen hasil eksperimen yang dapat Anda gunakan sebagai templat untuk eksperimen rekayasa kekacauan Anda.

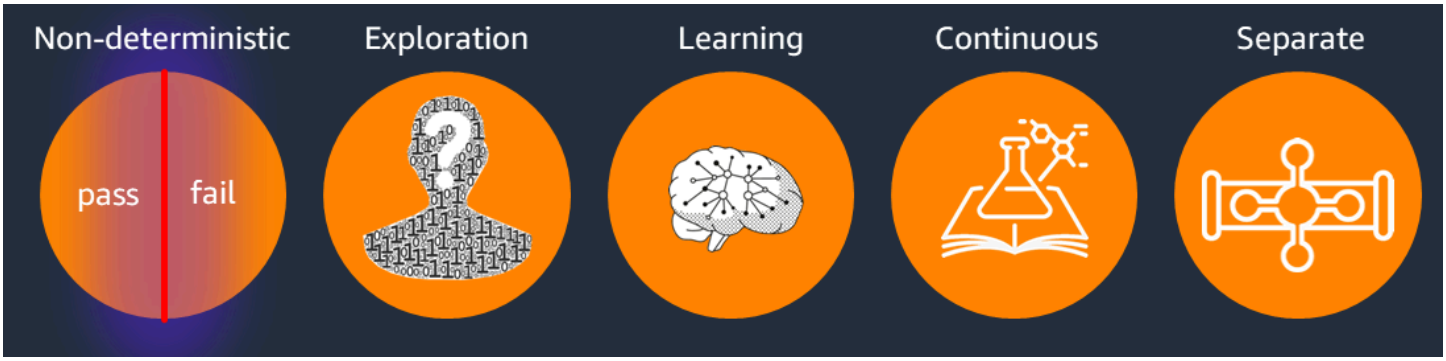
Bagian selanjutnya mengeksplorasi bagaimana karakteristik rekayasa kekacauan berbeda dari pengujian ketahanan tradisional seperti unit, asap, atau tes integrasi.

Gambaran umum

Membandingkan pengujian ketahanan dengan rekayasa kekacauan

Pengujian ketahanan bersifat deterministik. Artinya, ini memvalidasi karakteristik yang diketahui tentang mekanisme ketahanan, seperti pemutus sirkuit, percobaan ulang, kegagalan atau fallback, yang telah diterapkan dalam aplikasi Anda. Ini menegaskan bagaimana komponen aplikasi ini menyerap gangguan terkontrol dengan dampak minimal atau tanpa pengguna. Oleh karena itu, pengujian ketahanan berfokus pada validasi mode kegagalan yang diketahui yang disuntikkan ke dalam komponen aplikasi dengan tujuan menghasilkan hasil. pass/fail Anda harus menjalankan pengujian ketahanan terus menerus sebagai langkah dalam jalur Anda untuk memastikan bahwa Anda tidak memperkenalkan regresi pada postur ketahanan Anda. Dalam pengujian ketahanan, Anda sering tidak menjalankan pengujian terhadap komponen nyata, tetapi API tiruan yang mensimulasikan komponen tertentu. Pendekatan ini memungkinkan pengujian skenario kegagalan yang konsisten dan dapat direproduksi dalam lingkungan yang terkendali, sehingga cocok untuk integrasi pipa otomatis dan pengujian regresi.

Sebaliknya, rekayasa kekacauan bersifat non-deterministik. Artinya, ini berbasis hipotesis dan memverifikasi model mental Anda tentang bagaimana aplikasi dan ketergantungannya (orang, proses, dan teknologi) menyerap, beradaptasi, dan akhirnya pulih dari mode kegagalan yang tidak terduga. Oleh karena itu, chaos engineering berfokus pada verifikasi end-to-end dari mode kegagalan yang tidak diketahui, dengan tujuan menangkap cacat lebih awal, dan memperbaikinya sebelum berubah menjadi peristiwa skala besar. Rekayasa kekacauan mendorong pembelajaran berkelanjutan dan harus dipraktikkan melalui pipeline chaos terpisah atau eksperimen ad-hoc yang memungkinkan Anda menjalankan beberapa eksperimen kapan saja tanpa menghalangi produktivitas pengembang Anda dalam menerapkan kode.



Proses rekayasa kekacauan sering dimulai dengan hari permainan chaos, yang merupakan acara khusus di mana tim sengaja menyuntikkan kesalahan atau kegagalan yang dikendalikan ke dalam aplikasi mereka. Hari permainan progresif: Dimulai di lingkungan tingkat yang lebih rendah (seperti pengembangan atau pengujian) dan secara bertahap maju ke lingkungan tingkat yang lebih tinggi (seperti pementasan dan pra-produksi) saat kepercayaan meningkat. Dengan bergerak secara sistematis melalui lingkungan ini, tim dapat memverifikasi bahwa sistem mereka mentolerir kesalahan yang disuntikkan dengan benar sebelum mereka mencapai produksi. Perkembangan metodis ini memastikan bahwa pada saat eksperimen chaos dilakukan di lingkungan produksi, tim telah membangun kepercayaan besar pada kemampuan ketahanan sistem mereka. Proses hari permainan adalah pendekatan proaktif untuk mengidentifikasi kelemahan dan kerentanan dalam arsitektur aplikasi dan praktik operasional, sambil menghilangkan stres belajar selama pemadaman produksi yang tidak terduga.

Nilai rekayasa kekacauan

Sistem yang kompleks ada di mana-mana di dunia saat ini. Mereka memainkan peran penting dalam banyak aspek kehidupan kita, dari pasar keuangan hingga perawatan kesehatan. Kami berharap sistem ini selalu beroperasi. Namun, sistem yang kompleks seringkali rentan terhadap peristiwa dan perilaku tak terduga yang dapat memiliki konsekuensi signifikan. Organizations perlu merencanakan gangguan daripada bertanya-tanya apakah itu akan terjadi. Mereka dapat melakukannya dengan menerapkan pengujian skenario di seluruh layanan bisnis kritis atau mission-critical mereka. Di sinilah chaos engineering ikut bermain.

Chaos engineering menawarkan pendekatan untuk mengelola sistem kompleks yang dapat membantu mengurangi risiko dan meningkatkan ketahanan. Proses mempersiapkan eksperimen chaos membutuhkan tim untuk mengembangkan hipotesis tentang perilaku sistem mereka, yang memperdalam pemahaman mereka tentang bagaimana sistem dibangun dan bagaimana mereka beroperasi. Persiapan ini sering mengungkapkan kesenjangan mental, wawasan arsitektur, dan

pengetahuan operasional yang mungkin tetap belum ditemukan. Dengan memajukan pemahaman tentang bagaimana sistem yang kompleks bereaksi terhadap kegagalan, chaos engineering mempromosikan transparansi dan akuntabilitas yang lebih besar dalam desain dan manajemen sistem. Semakin sering organisasi Anda mempraktikkan rekayasa kekacauan, semakin siap mereka secara operasional. Rekayasa kekacauan membantu Anda menetapkan praktik terbaik untuk merancang aplikasi tangguh yang dapat bertahan dari kegagalan komponen dengan dampak minimal atau tanpa pengguna. Ini memastikan bahwa aplikasi penting beroperasi dalam tingkat layanan yang diharapkan dan toleransi dampak, sambil terus meningkatkan pengetahuan tim tentang sistem mereka sendiri.

Mempersiapkan kondisi buruk

Saat membangun AWS, Anda menggunakan berbagai jenis layanan, termasuk layanan zona seperti Amazon Elastic Compute Cloud (Amazon EC2), layanan Regional seperti Amazon Simple Storage Service (Amazon S3), layanan global seperti (IAM), layanan perangkat lunak pihak ketiga sebagai layanan AWS Identity and Access Management (SaaS), dan layanan lokal. Setiap jenis layanan memperlihatkan domain kegagalan berbeda yang perlu Anda perhitungkan. Bagaimana Anda mempersiapkan diri untuk peristiwa yang ditimbulkan sendiri, atau peristiwa yang disebabkan oleh pihak ketiga yang tidak dapat dikendalikan oleh organisasi Anda?

Untuk membantu memahami bagaimana aplikasi Anda dapat merespons kondisi buruk, Anda dapat menggunakan AWS Fault Injection Service (AWS FIS). AWS FIS adalah layanan yang dikelola sepenuhnya untuk menjalankan eksperimen injeksi kesalahan dengan cara yang terkontrol. Anda dapat menggunakan layanan ini untuk AWS menyuntikkan skenario yang disediakan seperti gangguan daya Availability Zone dan masalah konektivitas lintas wilayah, atau membuat eksperimen Anda sendiri dengan menyatukan berbagai macam tindakan kesalahan yang disediakan oleh layanan. AWS FIS memungkinkan tim Anda untuk terus berlatih dan mempelajari bagaimana aplikasi mereka akan bereaksi terhadap kesalahan umum dan memperbaiki cacat saat mereka mendeteksi mereka.

Mempraktikkan rekayasa kekacauan terkontrol

Prinsip-prinsip kunci dari eksperimen chaos terkontrol adalah:

- Mulailah dengan lingkungan yang mirip dengan lingkungan produksi Anda.
- Tetapkan hipotesis dan hentikan kondisi untuk eksperimen Anda.
- Mulai dari yang kecil.

- Lakukan kontrol atas eksperimen kekacauan Anda.
- Atur ruang lingkup dampak.
- Ketahui baseline layanan Anda.
- Jadwalkan eksperimen.
- Remediasi terlebih dahulu, lalu bereksperimen.
- Pantau eksperimen dengan cermat.
- Belajarlah dari hasil Anda.
- Prioritaskan temuan, remediasi, dan verifikasi.
- Sebarkan pembelajaran di seluruh organisasi Anda.

Agar berhasil menskalakan rekayasa kekacauan, Anda harus menerapkan eksperimen chaos dengan cara yang terkendali. Saat Anda menggunakan AWS FIS, Anda dapat membuat kondisi berhenti dengan menggunakan CloudWatchalarm Amazon. Anda dapat memasukkan kondisi ini ke dalam templat eksperimen untuk memastikan bahwa eksperimen dihentikan jika di luar batas dan diputar kembali ke keadaan terakhir yang diketahui. AWS FIS juga menyediakan tuas pengaman. Saat Anda menggunakan tuas ini, AWS FIS hentikan dan putar kembali semua eksperimen yang sedang berjalan di akun di Wilayah AWS, termasuk eksperimen multi-akun, dan mencegah eksperimen baru dimulai. Ini mencegah injeksi kesalahan selama periode waktu tertentu, seperti jam perdagangan, acara penjualan, atau peluncuran produk, atau sebagai respons terhadap alarm kesehatan aplikasi. Tuas pengaman tetap aktif sampai dilepaskan secara manual.

Saat Anda melakukan eksperimen chaos, Anda harus menentukan pengamanan untuk mencegah efek samping yang tidak diinginkan di lingkungan, terutama jika ada kemungkinan eksperimen tersebut akan memengaruhi aplikasi yang sedang diproduksi. Saat Anda merencanakan eksperimen, antisipasi efek samping yang mungkin ditimbulkannya pada aplikasi lain di lingkungan. Misalnya, aplikasi lain dapat menerima pesan yang salah dari aplikasi yang merupakan bagian dari eksperimen, mengalami volume permintaan yang tinggi, atau menghadapi pertentangan sumber daya jika mereka berbagi infrastruktur. Dokumentasikan risiko ini dan atasi masalah yang diketahui atau tidak dapat diterima sebelum Anda menjalankan eksperimen.

Memulai dengan rekayasa kekacauan

Sebelum Anda melakukan percobaan, kami sarankan Anda meletakkan beberapa hal penting untuk memaksimalkan praktik rekayasa kekacauan Anda. Hal-hal penting ini meliputi:

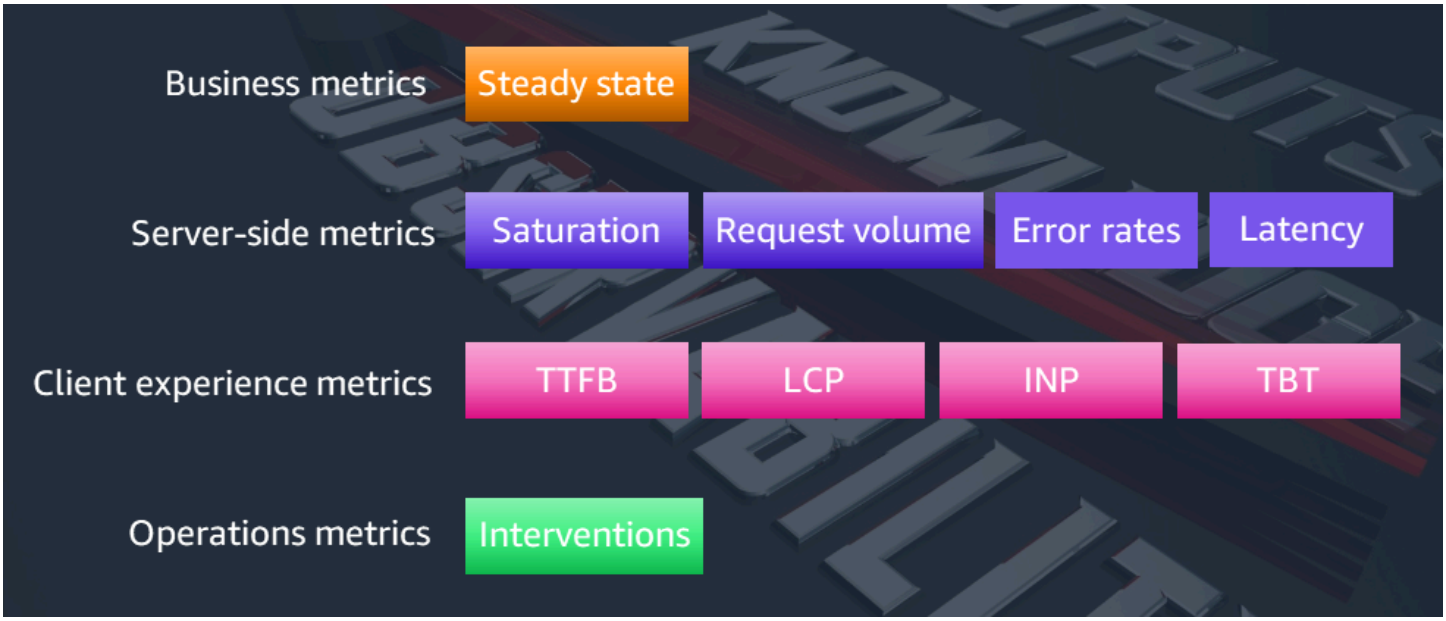
- Observabilitas (metrik, logging, penelusuran permintaan)
- Daftar peristiwa atau kesalahan dunia nyata yang ingin Anda jelajahi
- Sponsor ketahanan organisasi melalui dukungan kepemimpinan
- Prioritisasi temuan kritis, berdasarkan potensi dampak bisnis, di atas fitur baru yang ditemukan saat menjalankan eksperimen chaos

Observabilitas untuk eksperimen kekacauan

Observabilitas, yang terdiri dari metrik, logging, dan penelusuran permintaan, memainkan peran kunci dalam rekayasa kekacauan. Anda akan ingin memahami metrik bisnis, metrik sisi server, metrik pengalaman klien, dan metrik operasi saat menjalankan eksperimen. Tanpa observabilitas, Anda tidak akan dapat mendefinisikan perilaku kondisi tunak atau membuat eksperimen yang bermakna untuk memverifikasi apakah hipotesis Anda tentang aplikasi Anda benar.

Metrik-metrik

Diagram berikut menunjukkan jenis metrik yang dapat Anda lacak untuk eksperimen chaos untuk berbagai jenis aplikasi.



- Metrik bisnis — Steady state menunjukkan operasi normal sistem Anda dan ditentukan oleh metrik bisnis Anda. Ini dapat diwakili oleh transaksi per detik (TPS), aliran klik per detik, pesanan per detik, atau pengukuran serupa. Aplikasi Anda menunjukkan kondisi mapan saat beroperasi seperti yang diharapkan. Oleh karena itu, verifikasi bahwa aplikasi Anda sehat sebelum Anda menjalankan eksperimen. Steady state tidak selalu berarti bahwa tidak akan ada dampak pada aplikasi ketika kesalahan terjadi, karena persentase kesalahan dapat berada dalam batas yang dapat diterima. Kondisi mapan adalah baseline Anda. Misalnya, kondisi tunak sistem pembayaran dapat didefinisikan sebagai pemrosesan 300 TPS dengan tingkat keberhasilan 99 persen dan waktu pulang-pergi 500 ms. Secara visual, pikirkan kondisi tunak sebagai elektrokardiogram (EKG). Jika kondisi stabil sistem Anda tiba-tiba berfluktuasi, Anda tahu bahwa ada masalah dengan layanan Anda.
- Server-side metrik — Untuk memahami kinerja sumber daya selama percobaan, Anda memerlukan wawasan tentang kinerjanya sebelum, selama, dan setelah percobaan. Untuk mengukur dampak sumber daya Anda AWS, Anda dapat menggunakan Amazon CloudWatch. CloudWatch adalah layanan yang memantau aplikasi, merespons perubahan kinerja, mengoptimalkan penggunaan sumber daya, dan memberikan wawasan tentang kesehatan operasional. Selama eksperimen, Anda akan ingin menangkap metrik sisi server seperti saturasi, volume permintaan, tingkat kesalahan, dan latensi.
- Metrik pengalaman pelanggan — Aktif AWS, Anda dapat menangkap metrik pengguna nyata dengan menggunakan CloudWatchRUM untuk mensimulasikan permintaan pengguna melalui alat seperti Locust, Grafana k6, Selenium, atau Dalang. Metrik pengguna nyata sangat penting bagi organisasi yang melakukan eksperimen rekayasa kekacauan. Dengan memantau bagaimana

pengguna nyata terkena dampak selama percobaan, tim dapat memperoleh gambaran yang akurat tentang bagaimana kesalahan dan gangguan akan mempengaruhi pelanggan dalam produksi. Metrik pengalaman klien utama adalah Time to First Byte (TTFB), Largest Contentful Paint (LCP), Interaction to Next Paint (INP), dan Total Blocking Time (TBT).

- Metrik operasi — Intervensi mengukur seberapa berhasil Anda mengurangi kesalahan secara otomatis–misalnya, latensi permintaan klien yang berhasil selama reboot pod, tugas, atau instans EC2 dengan mekanisme seperti pengontrol replikasi atau penskalaan otomatis. Mampu melakukan intervensi secara otomatis selama kesalahan secara langsung berkorelasi dengan pengalaman pengguna yang baik. Memahami jika ada penyimpangan dalam mekanisme mitigasi ini dari waktu ke waktu sangat penting. Dengan mendefinisikan metrik untuk mitigasi otomatis yang berhasil dan gagal, Anda membuat pos panduan yang membantu mengidentifikasi potensi regresi di seluruh sistem Anda.

Pencatatan log

Pencatatan terpusat adalah kunci untuk memahami status komponen aplikasi Anda sebelum, selama, dan setelah eksperimen chaos. Kami menyarankan Anda mengumpulkan log dari semua komponen aplikasi Anda untuk membangun tampilan konsolidasi tentang apa yang dilakukan setiap komponen pada saat eksperimen disuntikkan. Ini memberikan gambaran yang jelas tentang aliran eksperimen ujung ke ujung.

Pelacakan permintaan

Penelusuran permintaan memungkinkan Anda mengamati aliran permintaan tunggal apa pun di seluruh komponen dalam aplikasi Anda untuk mendapatkan pemahaman yang komprehensif tentang dampak kegagalan yang disuntikkan terhadap sistem dan dependensinya. Dengan menelusuri permintaan, Anda dapat melihat bagaimana kegagalan menyebar melalui berbagai layanan dan komponen, sehingga Anda dapat menilai cakupan gangguan dengan lebih baik. Untuk melacak permintaan Anda AWS, Anda dapat menggunakan AWS X-Ray.

Skenario kegagalan untuk menyuntikkan dalam eksperimen kekacauan

Tujuan dari menyuntikkan kesalahan umum ke dalam aplikasi Anda adalah untuk memahami bagaimana aplikasi bereaksi terhadap kejadian tak terduga ini, sehingga Anda dapat membuat mekanisme mitigasi dan membuat sistem Anda tahan terhadap kesalahan tersebut. Selain itu, Anda

harus menggunakan rekayasa kekacauan untuk memutar ulang skenario kegagalan historis untuk memverifikasi bahwa mekanisme mitigasi Anda masih berfungsi seperti yang diharapkan dan tidak melayang seiring waktu.

Pertimbangkan peristiwa-peristiwa berikut ketika Anda merencanakan eksperimen rekayasa kekacauan Anda.

Mode kegagalan	Deskripsi
Kerusakan server	Reboot instans EC2, hapus pod Kubernetes atau tugas Amazon Elastic Container Service (Amazon ECS) untuk memahami bagaimana aplikasi Anda bereaksi terhadap crash tersebut.
Kesalahan API	Menyuntikkan kesalahan ke dalam AWS dan API layanan Anda sendiri untuk memahami perilaku aplikasi.
Masalah jaringan	Memperkenalkan latensi atau kemacetan, atau memblokir koneksi untuk meniru masalah jaringan dunia nyata.
AWS Gangguan Availability Zone	Putar ulang kerusakan dari seluruh Availability Zone untuk memverifikasi pemulihan di seluruh zona.
Wilayah AWS gangguan konektivitas	Putar ulang gangguan jaringan Wilayah AWS untuk memverifikasi bagaimana sumber daya di Wilayah sekunder bereaksi terhadap peristiwa semacam itu.
Kegagalan basis data	Gagal atas replika database atau data yang rusak, atau membuat instance database tidak dapat dijangkau, untuk memahami dampak pada aplikasi dan strategi pemulihan Anda.
Jeda dalam database dan replikasi Amazon S3	Jeda replikasi database atau Amazon Simple Storage Service (Amazon S3) di seluruh

Mode kegagalan	Deskripsi
	Availability Zone atau Wilayah AWS untuk memahami dampak aplikasi hilir.
Degradasi penyimpanan	Jeda I/O, hapus volume Amazon Elastic Block Store (Amazon EBS), atau hapus file untuk memverifikasi daya tahan dan pemulihan data.
Gangguan ketergantungan	Hapus atau turunkan kinerja layanan hilir atau hulu yang Anda andalkan, termasuk layanan pihak ketiga, untuk memahami aliran dan dampak end-to-end terhadap pelanggan Anda.
Lonjakan lalu lintas	Hasilkan lonjakan lalu lintas pengguna untuk menguji kemampuan penskalaan otomatis, dan lihat bagaimana waktu boot dingin dapat memengaruhi keseluruhan status aplikasi Anda.
Kelelahan sumber daya	Maksimalkan CPU, memori, dan ruang disk untuk memverifikasi degradasi aplikasi Anda yang anggun.
Kegagalan Cascading	Memulai kegagalan utama yang mengalir ke aplikasi dan komponen hilir.
Penerapan yang buruk	Luncurkan perubahan atau konfigurasi yang bermasalah untuk memverifikasi mekanisme rollback.

Sponsor ketahanan organisasi

Rekayasa kekacauan memberikan nilai paling besar ketika diterapkan di seluruh organisasi Anda. Kami menyarankan Anda bekerja dengan sponsor eksekutif yang dapat membantu menetapkan tujuan ketahanan di seluruh organisasi Anda, menghilangkan rasa takut, ketidakpastian, dan

keraguan tentang domain, dan memulai proses transformasi untuk membuat ketahanan menjadi tanggung jawab semua orang.

Untuk mendukung kasus bisnis membangun praktik rekayasa kekacauan, lampirkan upaya rekayasa kekacauan ke proyek bisnis penting Anda. Menjadikan ketahanan sebagai aset dan pendorong akselerasi akan membantu Anda melacak kesuksesan dari waktu ke waktu. Mulailah dengan hitungan insiden kritis per bulan atau per kuartal, waktu rata-rata untuk pulih, dan dampak yang ditimbulkan insiden ini terhadap pelanggan dan organisasi Anda. Tetapkan tujuan bersama tim Anda untuk mengurangi jumlah insiden selama periode 6 hingga 12 bulan karena perbaikan dilakukan di seluruh tumpukan aplikasi Anda sebagai respons terhadap eksperimen rekayasa kekacauan.

Ukur apakah insiden sangat berulang. Misalnya, katakanlah sertifikat TLS yang kedaluwarsa menyebabkan downtime karena klien tidak dapat membuat koneksi tepercaya. Jika beberapa insiden terjadi dalam setahun karena kedaluwarsa beberapa sertifikat TLS, Anda dapat menjalankan eksperimen kedaluwarsa sertifikat TLS dan memverifikasi bahwa tim Anda mendapatkan peringatan atau dapat secara otomatis mengurangi masalah tersebut. Ini akan membantu memastikan bahwa Anda menjadi tangguh terhadap kesalahan tersebut.

Untuk melacak kemajuan dalam rekayasa kekacauan dari waktu ke waktu, tangkap metrik berikut untuk membantu menyoroti nilai rekayasa kekacauan di seluruh siklus hidup aplikasi:

- Tingkat insiden berkurang — Lacak jumlah insiden produksi dari waktu ke waktu dan korelasikan angka ini dengan adopsi rekayasa kekacauan. Harapannya adalah bahwa tingkat insiden parah akan menurun.
- Peningkatan waktu rata-rata untuk resolusi (MTTR) - Hitung waktu rata-rata yang diperlukan untuk menyelesaikan insiden dan melacak data ini untuk melihat apakah itu membaik dengan rekayasa kekacauan dari waktu ke waktu.
- Peningkatan ketersediaan aplikasi — Gunakan metrik tingkat layanan untuk menunjukkan peningkatan ketersediaan saat ketahanan aplikasi meningkat melalui eksperimen chaos.
- Waktu yang lebih cepat ke pasar - Rekayasa kekacauan dapat memberikan kepercayaan diri untuk meluncurkan penawaran inovatif lebih cepat, karena Anda tahu bahwa aplikasi Anda tangguh. Lacak peningkatan kecepatan pelepasan produk.
- Pengurangan biaya operasional — Mengukur apakah indikator seperti kebisingan peringatan, beban operasional, dan upaya manual untuk mengelola aplikasi berkurang dengan praktik kekacauan yang berlaku.

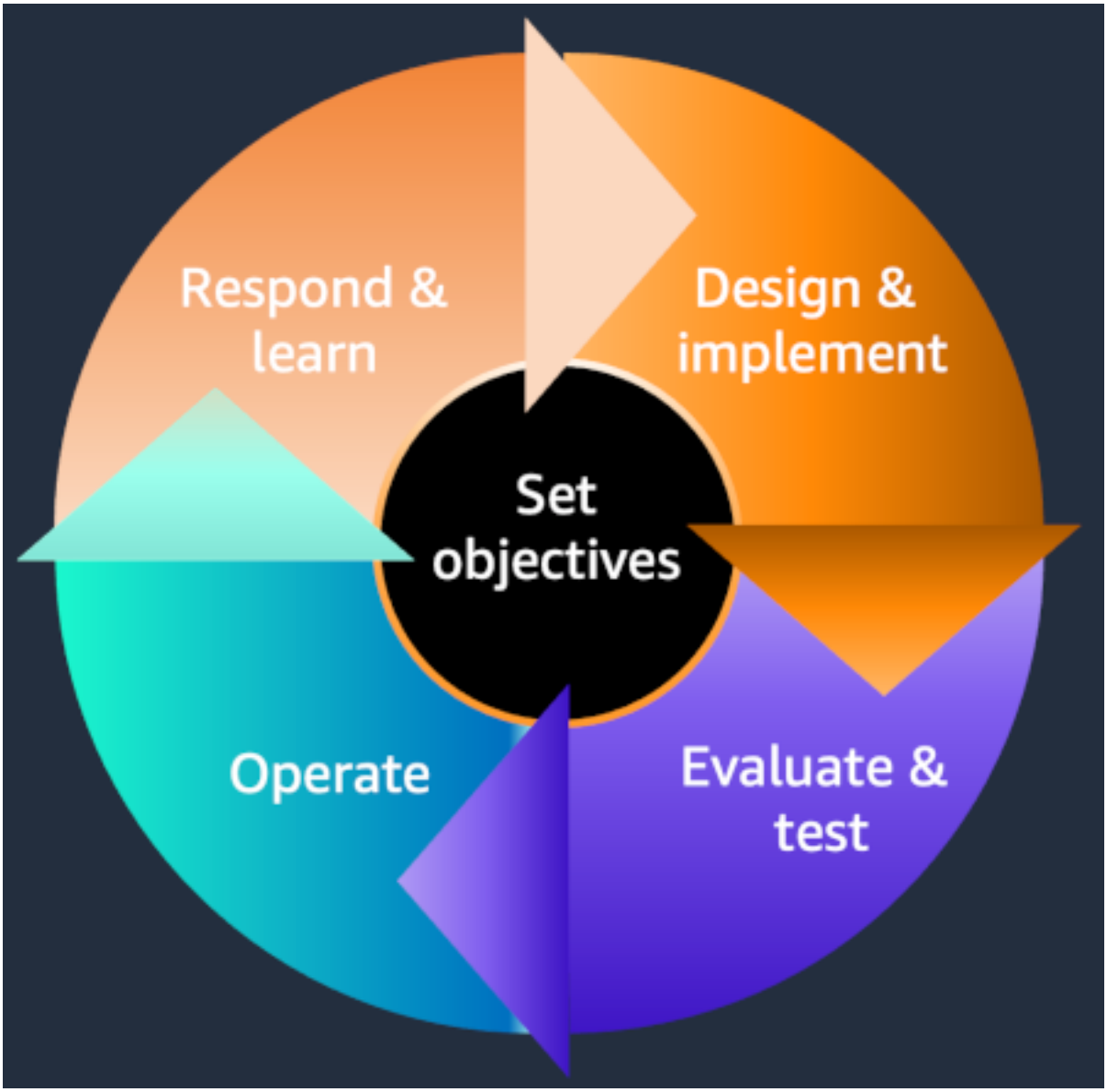
- Meningkatkan kepercayaan diri — Pengembang survei, insinyur keandalan situs (SRE), dan staf teknis lainnya untuk mengukur apakah rekayasa kekacauan meningkatkan kepercayaan mereka dalam ketahanan aplikasi. Persepsi penting.
- Pengalaman pelanggan yang lebih baik - Hubungkan rekayasa kekacauan ke hasil positif bagi pelanggan, seperti lebih sedikit gangguan layanan, kemunduran, dan pemadaman.

Memprioritaskan remediasi

Saat Anda melakukan eksperimen chaos, Anda cenderung mengidentifikasi area untuk perbaikan di mana aplikasi tidak berfungsi sebagaimana dimaksud. Remediasi barang-barang tersebut akan menjadi pekerjaan di backlog Anda yang harus diprioritaskan bersama dengan pekerjaan lain seperti pengembangan fitur. Kami menyarankan Anda meluangkan waktu untuk penyempurnaan ini untuk menghindari kegagalan di masa depan. Pertimbangkan untuk memprioritaskan pembelajaran dan tugas-tugas remediasi ini berdasarkan tingkat dampak yang mungkin ditimbulkannya. Temuan yang secara langsung berdampak pada ketahanan atau keamanan aplikasi Anda harus memiliki prioritas di atas fitur baru, untuk menghindari dampak pelanggan. Jika tim berjuang untuk memprioritaskan pekerjaan remediasi daripada pengembangan fitur, pertimbangkan untuk menghubungi sponsor eksekutif Anda untuk memastikan bahwa prioritas ditetapkan berdasarkan toleransi risiko bisnis.

Menerapkan rekayasa kekacauan pada AWS

Rekayasa kekacauan adalah bagian dari tahap evaluasi dan pengujian siklus hidup AWS ketahanan, seperti yang diilustrasikan dalam diagram berikut. Aplikasi terdistribusi tidak beroperasi secara terpisah dari aplikasi atau klien lain, jadi kami sarankan Anda meninjau seluruh siklus hidup ketahanan. Perubahan konstan untuk aplikasi terdistribusi saat jaringan berkembang, aplikasi hulu dan hilir mengalami pergeseran, dan penggunaan klien berubah seiring waktu.



Untuk memahami bagaimana perubahan ini pada aplikasi Anda dapat memengaruhi ketahanannya, jadikan chaos engineering sebagai bagian dari operasi Anda sehari-hari. Anda dapat menerapkan eksperimen kekacauan dengan berbagai cara:

- Ad hoc — Anda dapat melakukan eksperimen chaos sebagai eksperimen satu kali untuk mengatasi masalah atau pertanyaan tertentu.
- Chaos game days — Ini adalah peristiwa terstruktur dan berulang yang dirancang untuk memverifikasi keandalan dan ketahanan aplikasi. Tujuan dari chaos game day adalah untuk mengidentifikasi potensi masalah ketahanan atau kekurangan di seluruh orang, proses, dan teknologi, dan untuk mempraktikkan proses dan prosedur untuk mengidentifikasi, mengurangi, dan menanggapi insiden.
- Chaos pipeline — Integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD) adalah tentang membangun fitur baru dan menerapkannya dengan aman di seluruh lingkungan. Untuk mengimplementasikan eksperimen rekayasa kekacauan, buat pipeline chaos yang terpisah dari CI/CD pipeline Anda. Untuk memahami alasannya, mari kita asumsikan bahwa Anda ingin menambahkan eksperimen rekayasa kekacauan tunggal ke CI/CD pipa Anda yang menyuntikkan peningkatan kehilangan paket untuk komponen hilir. Eksperimen itu berjalan 3 kali dan membutuhkan waktu 5 menit untuk menyelesaikannya setiap kali. Kehilangan paket meningkat dari 10 persen menjadi 20 persen menjadi 30 persen dengan setiap proses, dan percobaan membutuhkan waktu 15 menit secara keseluruhan untuk menyelesaikannya. Jika Anda memiliki 100 penerapan paralel, Anda harus menunggu 1500 menit hingga satu percobaan selesai. Jika Anda memiliki 10 eksperimen untuk dijalankan, dampaknya terhadap pengembang Anda tidak akan tertahankan. Dalam skala besar, chaos engineering membutuhkan pipeline sendiri yang memungkinkan Anda menjalankan eksperimen secara paralel dengan proses siklus hidup pengembangan perangkat lunak (SDLC) Anda.
- Penyebaran kenari - Canary menyediakan lingkungan pengujian untuk eksperimen kekacauan. Dengan mengarahkan sebagian kecil lalu lintas ke layanan kenari atau menggunakan metode seperti pencerminan lalu lintas atau pemutaran ulang, Anda dapat memverifikasi infrastruktur baru atau perubahan kode tanpa dampak pada sistem produksi stabil Anda. Anda dapat menjalankan eksperimen terhadap kenari dan menyuntikkan kesalahan seperlunya, karena Anda dapat membatasi ruang lingkup dampak pada pengguna akhir.
- Eksperimen terjadwal — Anda dapat menjadwalkan eksperimen untuk memverifikasi mekanisme pemulihan yang dapat diprediksi untuk aplikasi Anda. Gunakan eksperimen terjadwal untuk memutar ulang peristiwa umum untuk menangkap bagaimana sistem Anda dapat pulih dari peristiwa seperti menghentikan instans EC2 di belakang grup penskalaan otomatis, menghapus pod Kubernetes, atau menghapus tugas Amazon ECS.

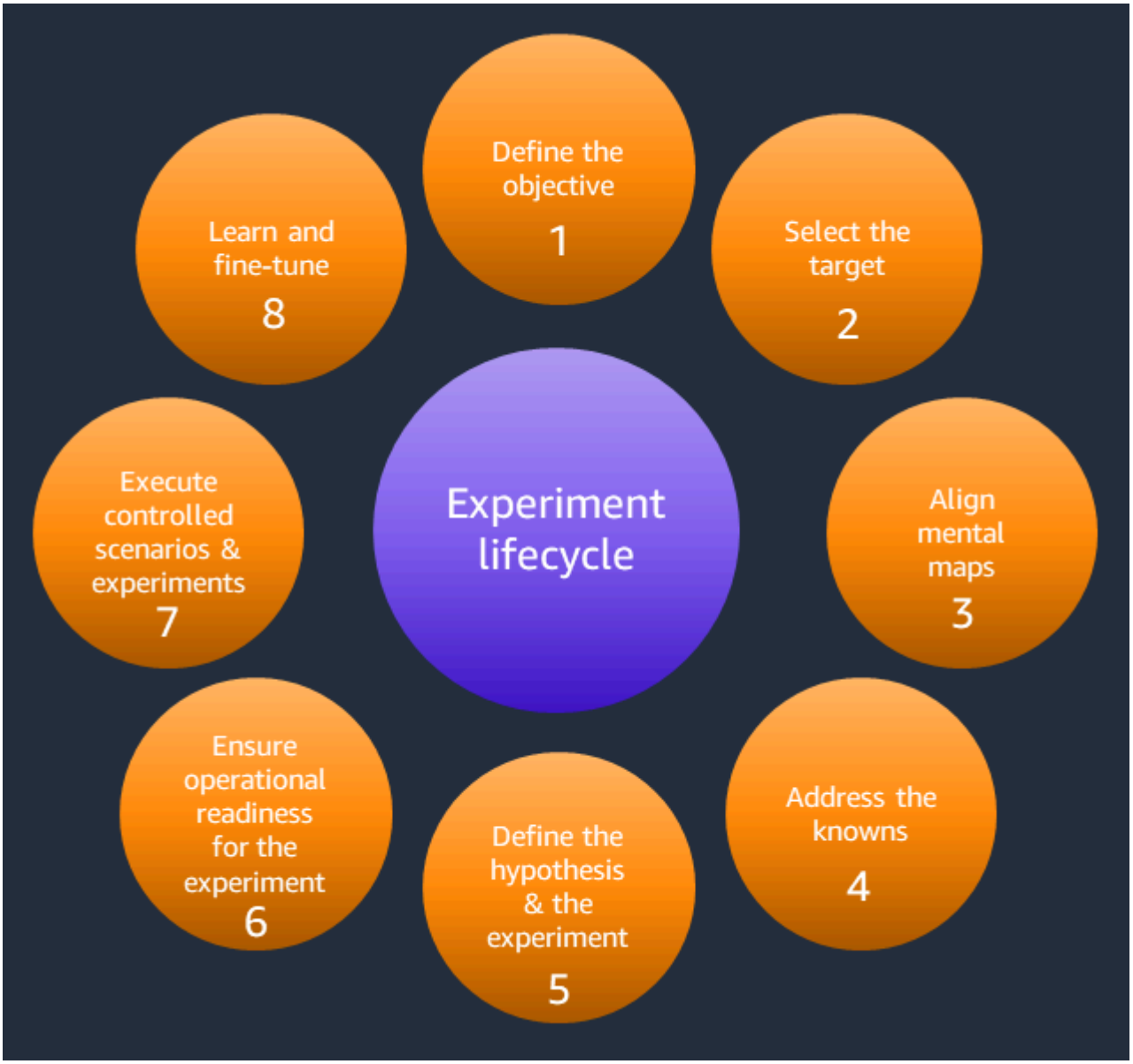
Siklus hidup eksperimen rekayasa kekacauan berkelanjutan

Seperti yang dibahas di bagian sebelumnya, Anda dapat menerapkan eksperimen rekayasa kekacauan dengan berbagai cara. Dalam semua kasus, kunci untuk membangun eksperimen chaos yang berarti adalah memahami aplikasi, insiden historis, dan remediasi yang diterapkan, dan untuk memahami dengan jelas area yang akan diselidiki, seperti ketahanan atau keamanan. Pengetahuan Anda tentang aplikasi membantu Anda merumuskan hipotesis tentang potensi kelemahan aplikasi dan memahami bagaimana aplikasi akan mendeteksi, memperbaiki, dan memulihkan ketika kesalahan disuntikkan.

Siklus hidup eksperimen chaos mencakup langkah-langkah berikut:

1. Tentukan tujuan percobaan.
2. Pilih aplikasi target.
3. Sejajarkan peta mental.
4. Atasi masalah yang diketahui dengan aplikasi Anda.
5. Tentukan hipotesis dan eksperimen.
6. Pastikan kesiapan operasional untuk percobaan.
7. Jalankan skenario dan eksperimen terkontrol.
8. Belajar dari dan menyempurnakan eksperimen.

Langkah-langkah ini diilustrasikan dalam diagram dan dibahas di bagian berikut.



Tentukan tujuan dan tetapkan harapan

Sebelum setiap percobaan, pastikan bahwa tujuan dan harapan Anda spesifik, terukur, dapat dicapai, relevan, dan terikat waktu. Tentukan dengan jelas hal-hal berikut:

- Identifikasi potensi kegagalan atau kelemahan dalam sistem dan layanan, untuk memahami bagaimana hal itu dapat berdampak pada pengguna. Ini termasuk mengidentifikasi kemungkinan mode kegagalan, seperti kerusakan server, kegagalan jaringan, atau bug perangkat lunak, dan menilai potensi dampaknya pada kinerja dan keandalan sistem secara keseluruhan.

- Mengukur dampak kegagalan dengan mendefinisikan indikator risiko utama (KRI) pada sistem dan layanan Anda. Ini termasuk mengukur efek kegagalan ketika metrik seperti latensi, throughput, dan tingkat kesalahan menyimpang dari kondisi tunaknya. Dengan memahami dampak penyimpangan tersebut, Anda dapat memprioritaskan upaya untuk mengurangi kegagalan berdasarkan risiko bisnis.
- Mengembangkan dan memverifikasi strategi untuk mengurangi atau mencegah kegagalan. Ini termasuk mengidentifikasi solusi potensial, seperti redundansi, koreksi kesalahan, atau strategi mundur, dan menguji efektivitasnya dalam lingkungan yang terkendali. Dengan memverifikasi strategi ini, Anda dapat memastikan bahwa Anda efektif dalam mencegah atau mengurangi kegagalan, dan dapat menerapkannya dalam sistem produksi Anda dengan percaya diri.
- Meningkatkan respon insiden dan proses pemulihan bencana. Dengan memutar ulang kegagalan dalam lingkungan yang terkendali, Anda dapat menguji proses respons insiden, mengidentifikasi potensi kemacetan atau celah, dan memperbaiki prosedur pemulihan bencana. Ini membantu memastikan bahwa Anda siap untuk merespons dengan cepat dan efektif jika terjadi kegagalan yang tidak terduga.

Pilih aplikasi target

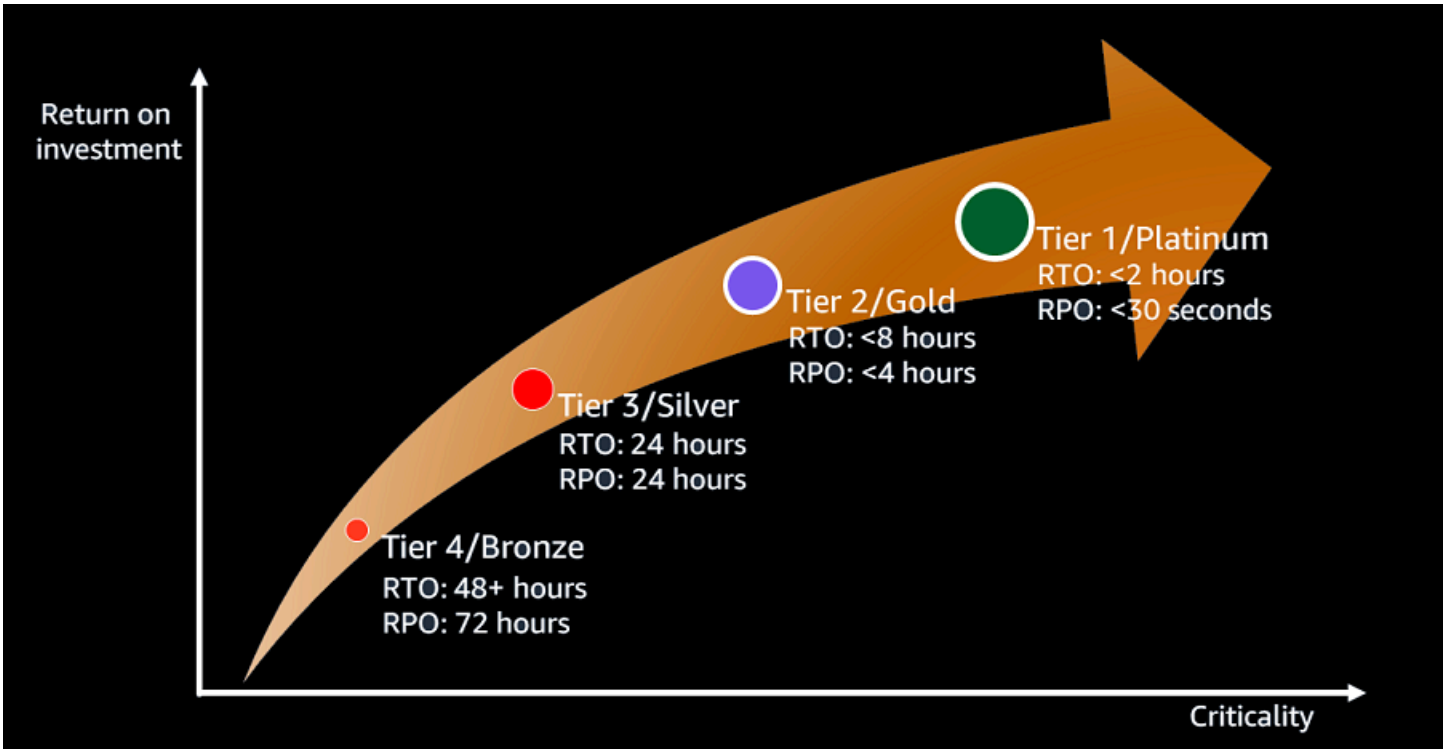
Rekayasa kekacauan adalah teknik yang ampuh tetapi membutuhkan prioritas yang bijaksana untuk memaksimalkan nilai. Saat memutuskan di mana harus memfokuskan upaya rekayasa kekacauan, mulailah dengan mempertimbangkan layanan penting bisnis Anda. Minta tim Anda untuk mengulangi tahapan siklus hidup pengembangan perangkat lunak, dan mulailah menyuntikkan kesalahan di lingkungan pengujian terlebih dahulu. Business-critical aplikasi secara langsung terkait dengan pendapatan, pengalaman pelanggan, dan operasi inti. Eksperimen kekacauan pada layanan ini dapat mengungkap kerentanan yang dapat sangat berdampak pada organisasi - dan berpotensi seluruh pasar - jika tidak ditangani. Misalnya, fokus pada layanan yang dihadapi pelanggan seperti sistem perdagangan atau sistem pesanan terlebih dahulu. Memprioritaskan layanan pusat ini memberikan perlindungan paling besar per investasi waktu.

Setelah layanan kritis, lihat komponen dasar seperti database, antrian pesan, jaringan, dan API layanan bersama. Ini dapat digunakan sebagai komponen atau layanan bersama di seluruh organisasi Anda, sehingga kegagalan mereka akan menyebabkan masalah yang meluas. Mengonfirmasi ketahanan layanan infrastruktur memberikan keyakinan bahwa mereka tidak akan melumpuhkan aplikasi dependen di atasnya. Misalnya, eksperimen rekayasa kekacauan yang menjatuhkan cluster Kafka mengungkapkan banyak hal tentang toleransi kesalahan aplikasi hilir.

Meskipun infrastruktur sistem tidak secara langsung menghadapi pelanggan, ini adalah target rekayasa kekacauan utama.

Jangan lupa untuk memetakan kesenjangan mental orang, proses, informasi fasilitas, dan ketergantungan pihak ketiga, karena ini dapat menyebabkan gangguan besar jika tidak selaras dengan tujuan toleransi dampak organisasi Anda. Untuk informasi lebih lanjut tentang mengukur ROI rekayasa kekacauan, baca Mengukur ROI rekayasa kekacauan dalam dokumen strategi Berinvestasi dalam rekayasa kekacauan sebagai kebutuhan strategis.

Diagram berikut menunjukkan laba atas investasi untuk menjalankan eksperimen chaos pada berbagai tingkatan layanan.



Sejajarkan peta mental (penemuan aplikasi)

Saat Anda menjalankan eksperimen ad-hoc atau hari permainan, Anda akan memulai proses penemuan aplikasi dengan mengadakan sesi papan tulis yang berfokus pada pemetaan detail aplikasi Anda. (Jika Anda menjalankan eksperimen di pipeline chaos, Anda akan sudah menyelaraskan peta mental itu, dengan mendefinisikan aplikasi target.) Pendekatan yang baik untuk memahami kesenjangan mental adalah meminta anggota tim paling junior menggambar diagram aplikasi terlebih dahulu, dan kemudian meminta lebih banyak anggota staf senior untuk

menambahkan diagram secara progresif. Ini akan mengungkap kesenjangan dalam pemahaman di seluruh tingkat pengalaman.

Diagram harus menggambarkan dependensi hulu dan hilir langsung dari aplikasi, serta integrasi pihak ketiga yang penting. Pastikan bahwa ada keselarasan pada aliran permintaan yang diharapkan melalui aplikasi. Memetakan alur kerja utama dan perjalanan pengguna untuk mendapatkan kejelasan tentang bagaimana pelanggan menggunakan aplikasi. Pertimbangkan untuk menggunakan diagram urutan untuk menangkap informasi ini.

Setelah sesi kolaboratif ini, tim harus memiliki model mental bersama dari aplikasi, dependensi kritisnya, dan kemampuan pemantauannya, dan pemahaman tentang risiko untuk membuat keputusan berdasarkan informasi untuk melanjutkan, atau membatalkan, eksperimen kekacauan potensial.

Mengatasi masalah yang diketahui dengan aplikasi Anda

Eksperimen rekayasa kekacauan dirancang untuk secara proaktif memunculkan cacat dalam suatu aplikasi. Dengan menyuntikkan kegagalan seperti peningkatan latensi, reboot server, atau gangguan daya Availability Zone, Anda dapat memverifikasi kemampuan aplikasi Anda untuk mentolerir gangguan yang realistis. Namun, proses ini mengasumsikan tingkat stabilitas dan kesehatan yang mendasari dalam aplikasi target. Menjalankan eksperimen chaos pada aplikasi yang sudah bermasalah berisiko menutupi masalah yang lebih dalam.

Sebelum melakukan rekayasa kekacauan, tim harus menyelesaikan cacat, bug, dan masalah kinerja yang diketahui dalam aplikasi mereka.

Tentukan hipotesis dan eksperimen

Insiden masa lalu yang telah menyebabkan gangguan pada aplikasi Anda atau aplikasi lain dalam organisasi Anda dapat berfungsi sebagai sumber yang sangat baik untuk ide eksperimen kekacauan. Misalnya, apakah pemadaman sebelumnya dipicu oleh kesalahan konfigurasi atau pola ketahanan yang hilang? Meninjau sejarah insiden dan memutar ulang akar penyebab kegagalan dunia nyata melalui eksperimen kekacauan adalah cara yang efektif untuk mengembangkan ketahanan terhadap masalah serupa di masa depan.

Sumber konsep eksperimen lain yang berharga dapat datang langsung dari para insinyur, arsitek, dan operator yang paling akrab dengan aplikasi. Memungkinkan anggota tim untuk mengirimkan

skenario kegagalan hipotetis yang mereka yakini dapat mengganggu aplikasi secara signifikan memungkinkan Anda mengumpulkan ide berdasarkan pengetahuan orang dalam. Tim aplikasi kemudian dapat mengevaluasi skenario mana yang diusulkan ini yang mungkin memiliki dampak potensial terbesar atau mengekspos risiko terbesar yang tidak diketahui. Menargetkan eksperimen chaos untuk skenario berisiko tinggi dan kurang dipahami dapat menghasilkan pembelajaran penting dan mencegah masalah di masa depan.

Sumber ide ketiga berasal dari melakukan pemodelan ketahanan untuk mengantisipasi kondisi yang akan menyebabkan kerugian bisnis yang teridentifikasi. Beberapa latihan pemodelan ketahanan memiliki pendekatan berbasis komponen untuk membangun model ketahanan sedangkan yang lain memiliki pendekatan berbasis sistem. Pendekatan berbasis komponen mengajukan pertanyaan, “Apa yang terjadi ketika komponen x berada di bawah beban ekstrim atau gagal?” Tim yang mengembangkan model ketahanan kemudian berspekulasi efek skenario seperti itu pada aplikasi yang lebih luas, dan mengidentifikasi pemantauan dan kontrol pencegahan yang saat ini ada untuk mendeteksi dan mengurangi efek skenario. Atau, pendekatan berbasis sistem mengikuti proses top-down untuk menyoroti keadaan aplikasi yang tidak diinginkan — seperti, “Etalase web menunjukkan tingkat inventaris basi” — dan mengundang tim aplikasi untuk mengantisipasi kondisi atau kondisi mana yang akan menyebabkan aplikasi berperilaku seperti ini.

Memastikan kesiapan operasional untuk percobaan

Anda memerlukan indikator yang dapat diukur untuk mengukur dampak kondisi buruk pada aplikasi dan perilakunya, seperti yang dijelaskan sebelumnya di bagian observabilitas. Mampu mengukur perilaku aplikasi memungkinkan Anda untuk menentukan apakah kondisi buruk memengaruhi aplikasi dan berapa besarnya.

Cara terbaik untuk memahami apakah ada dampak pada aplikasi Anda adalah dengan mengukur kondisi tunaknya. Steady state mengukur seperti apa operasi normal dan biasanya selaras dengan indikator pengalaman bisnis dan klien untuk aplikasi tertentu. Sebelum Anda melanjutkan ke langkah berikutnya, pastikan Anda memiliki observabilitas untuk memahami dampak, dan mekanisme rollback siap jika eksperimen tidak berjalan seperti yang diharapkan.

Jalankan eksperimen dan skenario terkontrol

Di AWS, kami tidak menyarankan melakukan eksperimen kekacauan awal pada aplikasi yang sedang diproduksi. Tujuan dari eksperimen chaos adalah untuk mempelajari sesuatu yang baru tentang bagaimana aplikasi berperilaku dalam kondisi stres. Perilaku aplikasi mungkin tidak dapat

diprediksi selama percobaan, jadi melakukan eksperimen untuk pertama kalinya dalam produksi dapat memiliki konsekuensi yang berdampak pada pelanggan. Oleh karena itu, Anda harus selalu menjalankan eksperimen chaos awal di lingkungan tingkat rendah yang memiliki potensi minimal untuk memengaruhi pengguna dunia nyata, dan kemudian mengulangi lingkungan Anda setelah Anda memverifikasi dan yakin bahwa aplikasi Anda dapat menyerap, beradaptasi, dan memulihkan dari tindakan yang disuntikkan..

Rencanakan setiap percobaan secara menyeluruh dengan menggunakan dokumen yang menangkap detail kunci, mirip dengan dokumen perencanaan eksperimen yang disediakan dalam lampiran. Beberapa bidang penting untuk dimasukkan adalah definisi keadaan tunak, hipotesis, dan metode injeksi kegagalan. Perencanaan, pelaksanaan, dan analisis eksperimen chaos dapat dicakup dalam satu artefak.

Setelah Anda menyelesaikan rencana tertulis untuk percobaan, siapkan kode yang diperlukan untuk menyuntikkan gangguan yang direncanakan yang diuraikan dalam dokumen.

Untuk menangkap dampak potensial selama percobaan, pastikan bahwa mekanisme observabilitas sudah ada. Jika Anda belum memiliki cara otomatis untuk menangkap hasil eksperimen, seperti laporan AWS FIS eksperimen, identifikasi anggota tim yang akan membuat catatan selama eksekusi, menangkap tangkapan layar dasbor, dan memimpin tim melalui eksperimen.

Belajar dan menyempurnakan

Setelah setiap percobaan, berkumpul sebagai tim untuk meninjau dan merenungkan eksperimen kekacauan. Lakukan upaya sadar untuk mempertahankan pola pikir yang tidak bersalah. Tujuan Anda haruslah memiliki dialog terbuka dan konstruktif yang berfokus sepenuhnya pada memperoleh pembelajaran maksimal, bukan menyalahkan.

Mulailah dengan meninjau definisi kondisi tunak dan hipotesis untuk percobaan. Apakah aplikasi berperilaku seperti yang diharapkan? Apakah ada kejutan yang membatalkan asumsi? Diskusikan pengamatan tentang bagaimana aplikasi bereaksi selama percobaan, baik dan buruk. Data yang dikumpulkan–metrik, log, tangkapan layar, dan sebagainya—harus menceritakan kisah persis apa yang terjadi.

Dekati tinjauan data ini dengan rasa ingin tahu alih-alih penilaian, dan identifikasi area di mana perbaikan dapat dilakukan pada desain aplikasi, dokumentasi, pemantauan, atau kemampuan lain berdasarkan pembelajaran. Item tindakan ini ditangkap sebagai proyek tindak lanjut untuk membuat aplikasi lebih tangguh.

Melalui pendekatan tanpa cela ini, Anda dapat melakukan percakapan jujur tentang apa yang salah dan bagaimana Anda dapat memperbaikinya. Asumsikan niat positif dari semua orang yang terlibat, dan percayalah bahwa mereka bekerja menuju hasil yang baik. Tujuan bersama Anda adalah pertumbuhan dan kemajuan organisasi melalui pembelajaran dan adaptasi berkelanjutan. Ulasan eksperimen Chaos yang dilakukan dengan cara yang konstruktif dan tidak bercela memberikan ruang yang aman bagi tim Anda untuk mendapatkan wawasan berharga yang membuat aplikasi dan organisasi Anda lebih andal dan tangguh dalam jangka panjang. Fokusnya tetap pada pembelajaran, bukan pada orang-orang. Untuk menyebarkan pembelajaran di seluruh tim Anda, publikasikan laporan hasil eksperimen di tempat sentral dan iklankan temuan sehingga orang lain dapat belajar darinya.

Menskalakan rekayasa kekacauan di seluruh organisasi Anda

Ketika organisasi Anda mengadopsi rekayasa kekacauan, standardisasi dan menerapkannya akan menghadirkan tantangan. Pada tahap awal kematangan, tim yang berbeda cenderung menggunakan perkakas dan variasi yang berbeda dari proses rekayasa kekacauan yang dijelaskan di bagian sebelumnya. Pada saat yang sama, beberapa tim mungkin tidak memprioritaskan atau mengadopsi rekayasa kekacauan sama sekali, terlepas dari potensi manfaatnya. Bagian berikut memberikan panduan tentang cara mengatasi tantangan ini.

Secara keseluruhan, pendekatan Anda terhadap rekayasa kekacauan harus dirancang untuk mencapai keseimbangan antara kepemimpinan terpusat dan partisipasi terdesentralisasi. Keseimbangan ini membantu memastikan bahwa rekayasa kekacauan diintegrasikan ke dalam proses pengembangan dan pembelajaran dibagikan di seluruh organisasi Anda.

Membangun praktik rekayasa kekacauan

Standarisasi praktik rekayasa kekacauan dapat mempercepat adopsi. Berbagi pembelajaran dari eksperimen lintas tim dapat memperbesar laba atas investasi rekayasa kekacauan.

Bangun pusat keunggulan terpusat, atau kumpulkan sekelompok ahli materi pelajaran, sebagai bagian dari praktik rekayasa kekacauan Anda. Sebagai fungsi kecil dan terpusat, tim ini dapat berfungsi di seluruh pengembangan perangkat lunak, infrastruktur, keamanan, dan tim bisnis dan mempertahankan standar yang digunakan oleh tim tersebut. Untuk kesederhanaan, pusat keunggulan disebut tim praktik terpusat, dan kelompok yang menerapkan rekayasa kekacauan disebut tim praktik di sisa panduan ini.

Peran tim praktik terpusat

Tim praktik terpusat bertanggung jawab untuk mengembangkan dan menerapkan praktik rekayasa kekacauan di seluruh organisasi. Mereka bekerja sama dengan tim yang berlatih untuk membimbing mereka dalam merancang dan melakukan eksperimen, dan memastikan bahwa eksperimen tersebut berharga bagi bisnis. Tim praktik terpusat juga memberikan panduan dan dukungan kepada tim pengembangan, infrastruktur, dan keamanan untuk membantu mereka mengintegrasikan rekayasa kekacauan ke dalam proses pengembangan mereka.

Tanggung jawab utama dari tim praktik rekayasa kekacauan terpusat meliputi:

- Pemberdayaan — Fungsi rekayasa kekacauan terpusat bertindak sebagai fasilitator untuk memperkenalkan praktik rekayasa kekacauan melalui hari-hari permainan dan lokakarya. Mereka memandu tim dalam proses rekayasa kekacauan, termasuk memilih skenario kegagalan, mendefinisikan hipotesis, dan menghasilkan laporan untuk dibagikan dengan organisasi yang lebih luas. Tim praktik terpusat harus memiliki materi pelatihan dan bekerja untuk meningkatkan keterampilan tim yang berlatih dalam penggunaan rekayasa kekacauan.
- Penasihat — Tim praktik terpusat juga dapat bertindak dalam peran penasihat untuk mengawasi eksperimen yang dilakukan oleh tim yang berlatih. Pengalaman dan pengetahuan mereka dapat memastikan bahwa eksperimen memberikan nilai bagi bisnis dan dilakukan dengan cara yang aman. Demikian pula, tim dapat mengawasi eksekusi dan tanya jawab eksperimen untuk membimbing orang-orang yang baru mengenal rekayasa kekacauan.
- Pemasaran dan pelacakan nilai — Mengkomunikasikan nilai bisnis dari rekayasa kekacauan adalah kunci keberhasilan program semacam itu. Setiap tim yang berpartisipasi dalam eksperimen rekayasa kekacauan harus mengumpulkan data dari eksperimen di seluruh bisnis dan menunjukkan nilai investasi organisasi ke dalam rekayasa kekacauan. Ini termasuk mengukur dan merayakan jumlah insiden yang dihindari selama setiap percobaan, waktu henti yang akan terjadi jika percobaan gagal, dan dampak keseluruhan terhadap bisnis jika skenario kegagalan terjadi dalam produksi. Dengan mengumpulkan dan memusatkan data tersebut dari seluruh tim, dan membuat data tersedia di seluruh organisasi, tim praktik terpusat dapat melacak dan memengaruhi nilai yang berasal dari adopsi rekayasa kekacauan di seluruh organisasi.
- Standar — Tim praktik terpusat harus memiliki dan memelihara proses untuk melakukan eksperimen chaos, templat untuk perencanaan dan pelaporan eksperimen, dan perkakas yang digunakan untuk melakukan eksperimen.

  Tim pusat harus memiliki dan mengelola templat perencanaan eksperimen, templat laporan eksperimen, dokumentasi proses, dan materi pemberdayaan. Dokumentasi praktik terbaik dan materi pemberdayaan memberikan panduan kepada tim yang berlatih tentang topik-topik seperti pagar pembatas yang dapat mereka gunakan untuk membatasi dampak eksperimen, kapan harus melakukan eksperimen dalam produksi, dan bagaimana mengembangkan penggunaan rekayasa kekacauan dari waktu ke waktu. Untuk contoh template dan output, lihat lampiran.

  Tim praktik terpusat juga harus memiliki proses untuk melakukan eksperimen, termasuk komunikasi dan eskalasi, dan kapan dan bagaimana berkomunikasi dengan tim lain dalam organisasi sebelum atau selama percobaan. Prosesnya juga harus menguraikan kapan pagar pembatas diperlukan.

Tim praktik terpusat juga harus memilih dan memiliki alat inti untuk melakukan eksperimen kekacauan (misalnya, alat seperti AWS FIS). Pemilihan dan implementasi alat pelengkap, seperti alat pembangkit beban, harus diserahkan kepada tim yang berlatih untuk memutuskan. Tim yang berlatih harus dapat menyesuaikan keseluruhan proses dan perkakas agar sesuai dengan kebutuhan mereka.

Peran tim yang berlatih

Tim terpusat bertanggung jawab untuk mendorong strategi rekayasa kekacauan secara keseluruhan, sedangkan tim yang berlatih berpartisipasi dalam proses dan memiliki pengembangan dan pelaksanaan eksperimen. Ini membantu memastikan bahwa eksperimen relevan dengan setiap produk atau layanan tertentu, dan bahwa pembelajaran dapat ditindaklanjuti dan dapat diterapkan untuk meningkatkan keandalan dan ketahanan produk. Tim praktik terpusat bertindak sebagai mentor dan pemilik standar dan proses rekayasa kekacauan organisasi. Namun, untuk mencegah tim terpusat menjadi hambatan, tim latihan individu perlu belajar dari praktik pusat untuk melakukan eksperimen kekacauan untuk diri mereka sendiri.

Membangun komunitas praktik

Selain membuat tim terpusat, kami menyarankan Anda untuk membentuk komunitas praktisi informal yang tertarik pada rekayasa kekacauan. Komunitas ini menyediakan platform untuk berbagi pengetahuan, praktik terbaik, dan pengalaman di seluruh tim yang berlatih dan organisasi yang lebih luas.

Komunitas praktik dapat dioperasikan oleh tim praktik teknik chaos terpusat, tetapi siapa pun di dalam organisasi dapat menjadi anggota komunitas. Tim terpusat dapat memanfaatkan komunitas praktik untuk menyiarkan pembaruan dan pembelajaran sumber, dan untuk mengumpulkan umpan balik dari tim yang berlatih yang menggunakan standar dan proses yang dikelola oleh tim terpusat. Komunitas akan bertindak sebagai loop umpan balik untuk menginformasikan tim terpusat tentang efektivitas praktik rekayasa kekacauan di seluruh tim yang berlatih. Tim praktik terpusat kemudian dapat menyesuaikan dokumentasi mereka dan artefak pendukung untuk mendukung tim produk dengan sebaik-baiknya.

Memasukkan rekayasa kekacauan ke dalam ketahanan operasional Anda

Eksperimen kekacauan adalah investasi oleh bisnis Anda untuk mencegah insiden dalam produksi. Penting untuk menentukan di mana bisnis dapat mewujudkan pengembalian terbesar atas investasi ini. Organisasi dapat bekerja dengan tim praktik rekayasa kekacauan terpusat untuk memperbarui standarnya dan menentukan produk mana yang cukup penting untuk memerlukan eksperimen kekacauan.

Proses pengembangan sistem

Rekayasa kekacauan dan eksperimen chaos harus dilakukan berulang kali sebagai bagian dari siklus hidup aplikasi. Mirip dengan bagaimana tim secara teratur melakukan tes pemulihan bencana, mereka harus melakukan eksperimen kekacauan dan hari permainan secara terus menerus dan berkala sepanjang tahun. Pendekatan ini meningkatkan bagaimana organisasi mengantisipasi, mengamati, dan menanggapi insiden.

Kesimpulan

Disiplin rekayasa kekacauan telah berkembang jauh selama dekade terakhir. Ini telah diadopsi di berbagai industri, dan telah membantu organisasi menciptakan layanan yang tangguh dan meningkatkan kepuasan pelanggan. (Untuk contoh bagaimana organisasi menerapkan praktik ini, lihat Kisah Rekayasa Kekacauan.) Chaos engineering memungkinkan organisasi untuk mengurangi risiko untuk aplikasi mission-critical mereka dengan menyuntikkan kesalahan terkontrol di semua tingkat tumpukan aplikasi mereka, termasuk layanan penyedia cloud. Memiliki kemampuan untuk mempengaruhi seluruh tumpukan aplikasi dengan cara yang terkontrol memungkinkan ketahanan berkelanjutan, peningkatan keunggulan operasional, observabilitas, dan arsitektur berorientasi pemulihan tanpa tekanan pemadaman produksi. Pendekatan ini juga mengarah pada praktik pengujian ketahanan yang lebih baik di seluruh organisasi. Untuk mulai merangkul rekayasa kekacauan, jalankan hari permainan kekacauan atau lokakarya untuk menunjukkan nilai yang dapat diberikan oleh rekayasa kekacauan kepada organisasi Anda.

Sumber daya

AWS FIS implementasi model kegagalan:

- Gunakan AWS Fault Injection Service untuk menunjukkan ketahanan aplikasi multi-wilayah dan Multi-AZ (posting blog)AWS
- AWS Fault Injection Simulator mendukung eksperimen rekayasa chaos di Amazon EKS Pods (posting AWS blog)
- Mengumumkan AWS Fault Injection Simulator fitur baru untuk beban kerja Amazon ECS (posting blog)AWS
- Tingkatkan ketahanan aplikasi dengan metrik volume Amazon EBS dan AWS Fault Injection Simulator (posting blog)AWS
- Rekayasa kekacauan memanfaatkan AWS Fault Injection Simulator di AWS lingkungan multi-akun (AWS posting blog)
- Eksperimen kekacauan di Amazon RDS menggunakan AWS Fault Injection Simulator (posting AWS blog)
- Membangun aplikasi tanpa server yang tangguh menggunakan rekayasa kekacauan (posting blog)AWS
- Gunakan FIS untuk mengganggu instance spot (lokakarya)AWS
- Mengotomatiskan Rekayasa Kekacauan di Jalur Pipa Pengiriman Anda (PosAWS komunitas)

Sumber daya lainnya:

- Berinvestasi dalam rekayasa kekacauan sebagai kebutuhan strategis
- Kisah Rekayasa Kekacauan
- CloudWatchDokumentasi Amazon
- Dokumentasi Amazon CloudWatch RUM
- Dokumentasi AWS X-Ray
- Dokumentasi AWS FIS

Lampiran: Contoh dokumen

Dokumen sampel yang disediakan di bagian ini menggunakan situs adopsi hewan peliharaan fiktif (PetSite) sebagai aplikasi target untuk eksperimen kekacauan.

- Dokumen perencanaan eksperimen
- Dokumen hasil percobaan

Dokumen perencanaan eksperimen

Keadaan mantap

Nama proses	Situs adopsi hewan peliharaan
Arsitektur fisik	(Tautan ke diagram arsitektur.)
Arsitektur logis	(Tautan ke diagram logis.)
Tentukan kondisi tunak	Waktu buka halaman rata-rata, diukur dengan menggunakan Largest Contentful Paint (LCP), untuk situs adopsi hewan peliharaan adalah 2,5 detik atau kurang dengan latensi 99 persentil (P99) 4,0 detik atau kurang dengan baseline 5000 pengguna bersamaan.
Metrik kondisi stabil	Metrik LCP ditangkap di seluruh basis pengguna, dan metrik emas (latensi, throughput, tingkat kesalahan, saturasi).
Observabilitas kondisi mapan	LCP akan ditangkap oleh browser pengguna, dikirim ke Amazon CloudWatch, dan diperiksa dengan CloudWatch RUM. Selama periode 60 detik, rata-rata dan waktu P99 LCP akan dikumpulkan untuk semua permintaan dalam periode tersebut. Top-level metrik emas ditangkap dengan menggunakan CloudWatch.

Nama proses	Situs adopsi hewan peliharaan
Proses untuk mencapai kondisi tunak	Grafana K6 akan digunakan untuk membuat beban yang mensimulasikan tingkat lalu lintas produksi normal sekitar 5K pengguna bersamaan.

Persyaratan observabilitas

Tim harus dapat melihat yang berikut:

- Steady state: Apa yang akan diamati untuk memverifikasi bahwa aplikasi dalam kondisi normal?
- Kondisi kegagalan: Bagaimana kondisi kegagalan akan muncul di dasbor? Contoh:
  - Alarm yang harus dipicu
  - Log yang harus dihasilkan
- Dampak kegagalan: Apa yang harus diamati untuk melihat komponen yang diharapkan terkena dampak (ruang lingkup dampak)?
- Pemulihan: Bagaimana pemulihan akan dilihat dan diukur untuk menangkap MTTR?
- Debug: Detail pemecahan masalah tentang kegagalan eksperimen.

Tabel berikut memberikan saran dan contoh untuk bagan persyaratan observabilitas. Anda harus menentukan apa yang harus diamati berdasarkan eksperimen spesifik Anda.

Apa yang perlu diperhatikan	Tautan ke alat observabilitas	Apa yang sedang diamati
Sumber masukan	Dasbor Grafana K6	• Menjalankan jumlah kontainer • Permintaan per detik
Kesehatan aplikasi secara keseluruhan	• CloudWatch Dasbor adopsi hewan peliharaan	• Jumlah simpul sehat Amazon EKS

Apa yang perlu diperhatikan	Tautan ke alat observabilitas	Apa yang sedang diamati
	• Dasbor pengalaman pengguna adopsi hewan peliharaan (RUM)	• Pemanfaatan CPU node Amazon EKS
Alur kerja kesehatan	CloudWatch Dasbor adopsi hewan peliharaan	Waktu LCP, metrik emas
Jejak	X-Ray Dasbor adopsi hewan peliharaan	• Minta latensi • Jumlah permintaan • Hitungan kegagalan
Beberapa catatan	CloudWatch Log adopsi hewan peliharaan	Setiap kesalahan yang ditemui oleh pod akan dikeluarkan ke CloudWatch Log.

Definisi eksperimen

Nama percobaan	Stres CPU Amazon EKS PetSite pod
Kode sumber percobaan	(Tautan ke repositori sumber eksperimen.)
Deskripsi percobaan	Eksperimen ini mengeksplorasi bagaimana peningkatan penggunaan CPU dari pod PetSite aplikasi akan berdampak pada pengalama n pelanggan secara keseluruhan. Dengan menyuntikkan stress CPU ke setiap PetSite pod yang sedang berjalan, kita akan dapat memahami apakah ada dampak terhadap pelanggan dan sejauh mana dampaknya.
Persyaratan atau parameter eksperimen	Beban aplikasi: Rata-rata produksi

Nama percobaan	Stres CPU Amazon EKS PetSite pod
	Pemilih label pod: `app=petsite`
Durasi eksperimen	10 menit
Lingkungan	Lingkungan uji alfa
Sumber daya target percobaan	PetSite pod aplikasi
Garis dasar percobaan yang diperkenalkan melalui alat pembangkit beban	• 54% permintaan memiliki LCP <2,5 detik. • 46% permintaan memiliki LCP <4 detik. • Tidak ada kesalahan yang diamati.
Kondisi backoff	Tidak ada

Hipotesis

Bagaimana jika	Dampak	Pemulihan
Apa yang akan terjadi pada kondisi stabil jika pod PetSite aplikasi mengalami atau menyebabkan lebih dari 60% pemanfaatan CPU selama 10 menit di bawah lalu lintas tingkat produksi normal?	Waktu LCP akan tetap di bawah 2,5 detik untuk P50 pengguna dengan P99 4,0 detik atau kurang. Konsumen harus dapat memuat halaman PetSite arahan.	Deteksi: • Stres CPU akan dideteksi oleh alarm yang dikonfigu rasi. CloudWatch • Metrik LCP juga akan menghasilkan alarm untuk degradasi pengalaman pengguna. Self-healing: • Sifat terdistribusi dari arsitektur microservice

Bagaimana jika	Dampak	Pemulihan
		berarti bahwa banyak instance Pod berjalan di beberapa Availability Zone. • Pesawat kontrol kluster EKS akan mengalihkan lalu lintas dari pod yang terpengaruh, dan akan meluncurkan pod baru pada node pekerja. Pemulihan: Ketika pemanfaatan CPU kembali normal, LCP harus pulih secara otomatis.

Proses percobaan

Sesuaikan contoh proses langkah demi langkah berikut dengan eksperimen spesifik Anda:

1. Validasi akses ke, dan fungsionalitas, semua Amazon CloudWatch, CloudWatch RUM, dan AWS X-Ray dasbor.
2. Validasi kesehatan lingkungan aplikasi:
   a. Konfirmasikan bahwa cluster EKS sehat dengan menggunakan CloudWatch dasbor.
   b. Kunjungi penerapan aplikasi situs adopsi hewan peliharaan uji di URL contoh.
3. Memulai beban untuk mencapai kondisi tunak:
   a. Konfirmasikan bahwa generator beban sedang berjalan dan mengirim 5000 permintaan per detik.
   b. Biarkan 5 menit agar aplikasi mencapai kondisi tunaknya.
   c. Konfirmasikan kondisi tunak aplikasi dengan menggunakan dasbor CloudWatch RUM.
4. Memulai kesalahan (percobaan):
   a. Buka AWS FIS konsol.

b. Jalankan eksperimen pet-adoption-pod-stress.

c. Konfirmasikan bahwa percobaan sedang berjalan di konsol.

5. Amati dampak kesalahan pada aplikasi Anda:

   a. Tangkap tangkapan layar dari CloudWatch RUM dan CloudWatch dasbor, dan catat titik data anomali apa pun.

   b. Setelah percobaan selesai AWS FIS, ambil tangkapan layar tambahan untuk merekam jika aplikasi kembali ke kondisi tunak tanpa adanya stres, dan catat anomali apa pun pada titik data.

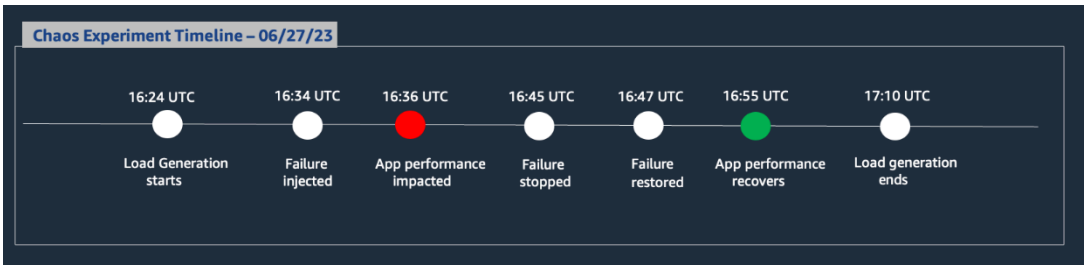
   c. Jika kondisi tunak tidak dilanjutkan, ambil langkah-langkah untuk memulihkan aplikasi dan mencatat langkah-langkah yang diambil.

6. Validasi bahwa lingkungan telah kembali normal:

   - Tinjau semua metrik bisnis, pengalaman pengguna, aplikasi, dan infrastruktur untuk memverifikasi bahwa sistem telah kembali ke keadaan yang diketahui. Tangkap tangkapan layar dasbor jika membantu.

Garis waktu percobaan

Pastikan Anda menangkap garis waktu eksperimen ujung-ke-ujung, dimulai dengan pembuatan beban, injeksi kesalahan, pengamatan dampak, dan pemulihan aplikasi, dan berakhir saat Anda menghentikan pembuatan beban. Ini diilustrasikan dalam contoh berikut.



Hasil percobaan

Eksperimen menjalankan ID	Hasil percobaan
PET-ADOPT-EXP-23	(Tautan ke hasil eksperimen.)

Cacat yang teridentifikasi

- Cluster Kubernetes tidak mendeteksi kerusakan CPU pada PetSite pod, sehingga tidak menjadwalkan penerapan tambahan.
- Tidak ada peningkatan tingkat kesalahan 4XX atau 5XX sebagai hasil dari percobaan ini.
- Kita perlu menyesuaikan pemeriksaan kesehatan pod untuk memperhitungkan dampak ke LCP ketika ada kendala sumber daya.

Dokumen hasil percobaan

Konfigurasi

Dokumentasikan konfigurasi spesifik untuk percobaan. Contoh:

- Pembuatan beban diatur untuk mensimulasikan pengguna 5K yang mengeluarkan total 85 permintaan per detik.

Prasyarat

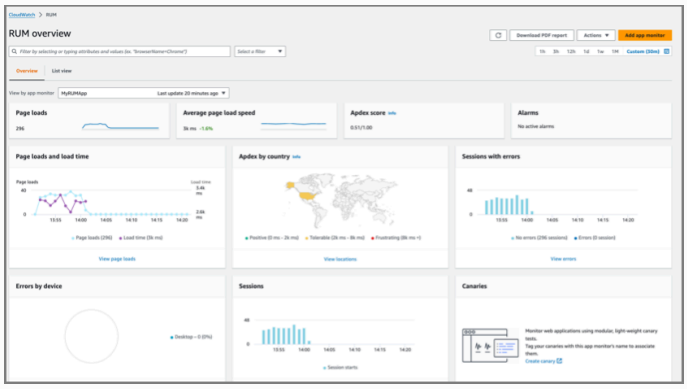
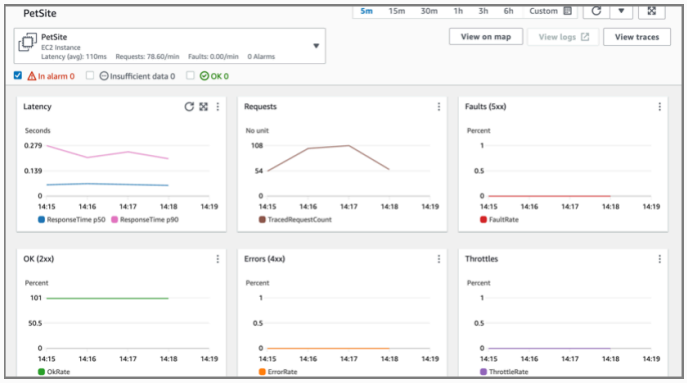
- Memverifikasi bahwa situs adopsi hewan peliharaan berjalan di lingkungan pengujian alfa.
- Memverifikasi bahwa template eksperimen telah dikonfigurasi untuk menerapkan stress CPU ke pod PetSite aplikasi yang berjalan di cluster EKS. Pod aplikasi diidentifikasi oleh label Kubernetes. `app=petsite`
- Beban dikonfirmasi berjalan dan menghasilkan 85 permintaan per detik.

Keadaan mantap

Dokumentasikan langkah-langkah yang diambil untuk mencapai kondisi tunak dan bagaimana Anda memverifikasinya. Contoh:

Untuk penerapan pengujian situs adopsi hewan peliharaan, beban 85 RPS sedang dihasilkan untuk mensimulasikan kondisi tunak. CloudWatch RUM dan CloudWatch dasbor ditinjau untuk memverifikasi bahwa semua metrik bisnis dan aplikasi berada dalam rentang normal sebelum pelaksanaan percobaan.

Data observabilitas:

Expected	Diamati
• LCP kurang dari 4 detik untuk P99 permintaan. • Latensi respons kurang dari 500 ms. • Tidak ada kesalahan 4XX atau 5XX.	 

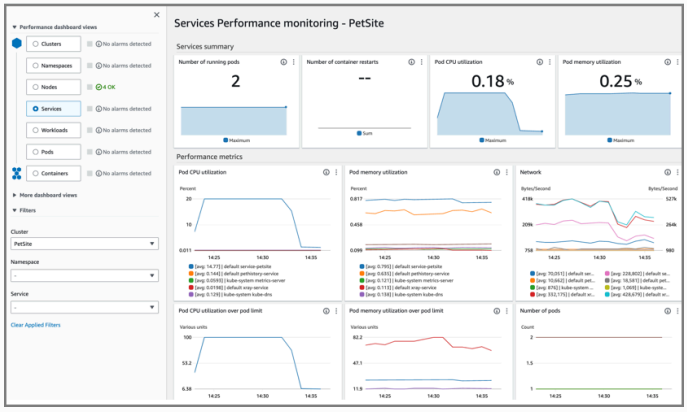
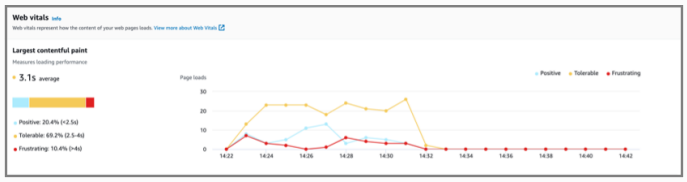
Injeksi kesalahan

AWS FIS digunakan untuk menyuntikkan kesalahan dengan menggunakan templat percobaan (berikan tautan). Eksperimen diatur untuk berjalan selama 10 menit, dan rollback dikonfigurasi jika node pekerja mengalami stres CPU lebih dari 60 persen.

Pengamatan kesalahan

CloudWatch RUM dan CloudWatch dasbor ditinjau untuk melacak kondisi tunak aplikasi (ditentukan dengan menggunakan metrik LCP). Tangkapan layar ditangkap dalam tabel berikut.

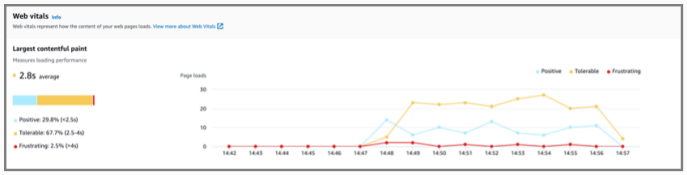
Data observabilitas:

Expected	Diamati
• LCP harus tetap di bawah 4 detik untuk P99. • Waktu respons harus tetap di bawah 500 ms. • Tidak ada kesalahan 4XX atau 5XX yang harus ditemui.	 

Pemulihan

Setelah stress telah dihapus (AWS FIS percobaan telah menyelesaikan dan menghapus stres CPU dari pod), aplikasi harus melanjutkan kondisi tunak normalnya. Tidak ada intervensi manual yang diperlukan.

Data observabilitas:

Expected	Diamati (tangkapan layar)
LCP P99 harus di bawah 4 detik dengan rata-rata di bawah 2,5 detik.	

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
Publikasi awal	—	April 4, 2025

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