에서 카오스 엔지니어링을 사용하여 복원력 향상 및 고객 경험 향상 AWS

AWS 권장 가이드

AWS 권장 가이드: 에서 카오스 엔지니어링을 사용하여 복원력 향상 및 고객 경험 향상 AWS

Table of Contents

에서 카오스 엔지니어링을 사용하여 복원력 향상 및 고객 경험 향상 AWS

Laurent Domb, Amazon Web Services Federal Financials의 수석 기술자

2025년 4월(문서 기록)

카오스 엔지니어링은 프로덕션 환경에서 격렬한 조건을 견딜 수 있는 조직 및 애플리케이션의 기능에 대한 신뢰를 구축하기 위해 애플리케이션을 실험하는 분야입니다. 이는 애플리케이션과 조직이 사람, 프로세스 및 기술 전반에 제어된 장애를 도입하여 서비스 장애를 흡수하고 이에 적응하며 궁극적으로 복구할 수 있는지 확인하는 것을 목표로 하는 복원력에 대한 선제적 접근 방식입니다. 또한 약점이 프로덕션 중단 또는 기타 중단을 일으킬 수 있기 전에 약점을 식별하고 제거하는 것이 목적입니다.

Amazon은 장애 발생에도 불구하고 작동하는 것이 정상적인 작동 방식이라는 점에서 분산 시스템에서 장애가 불가피하다는 점을 이해합니다. 서비스 간 상호 작용이 실패할 수 있으므로 다양한 장애 모드에서 서비스가 어떻게 반응하는지 이해하고 종속성 장애, 재시도 폭풍, 손상된 가용 영역, 호스트 리소스 소진과 같은 주요 취약성에 복원력이 뛰어난 서비스를 구축해야 합니다.

재시도 폭풍의 예를 들어 보겠습니다. 클라이언트의 현지화된 장애는 여러 서비스에 상당한 영향을 미칠 수 있습니다. 이를 일반적으로 나비 효과라고 합니다. 재시도 폭풍은 실패한 종속성이 클라이언트와 해당 클라이언트의 클라이언트를 트리거하여 실패한 작업을 재시도하여 트래픽이 기하급수적으로 증가하는 나비 효과의 표현입니다. 성능 저하를 처리하는 동안 트래픽을 재시도하는 것 외에도 일반 트래픽에 응답해야 하기 때문에 서비스가 오버로드됩니다.

분산 시스템의 복잡성 증가에 대한 대응으로 카오스 엔지니어링이 등장했습니다. 카오스 이론, 시스템 사고 및 엔지니어링의 원칙을 결합하여 예상치 못한 이벤트 및 동작에 복원력이 뛰어난 복잡한 시스템을 설계하고 관리하는 다학제적 접근 방식입니다. 카오스 엔지니어링의 핵심은 불확실성과 예측 불가능한 조건에서 복잡한 시스템의 동작을 이해하고 관리하는 것입니다. 결과 예측 및 제어에 의존하는 엔지니어링에 대한 기존 접근 방식은 분산 시스템의 복잡하고 동적인 특성을 처리하기에 충분하지 않은 경우가 많습니다. 이러한 시스템이 성장함에 따라 단일 개인의 이해 범위를 초과하는 경우가 많습니다.

카오스 엔지니어링은 시스템에 장애를 의도적으로 주입하여 프로덕션에서 발견되기 전에 약점을 발견하는 개념, 기법 및 도구를 제공합니다. 이러한 선제적 접근 방식을 통해 조직은 시스템이 스트레스가 많은 조건에서 작동할 것이라는 확신을 구축할 수 있습니다. 카오스 엔지니어링은 여전히 진화하는 관행이지만 복잡성과 상호 연결성 증가에 직면하여 최신 컴퓨팅 시스템을 설계, 관리 및 운영하기 위한 근본적인 전환을 나타냅니다.

이 가이드의 다음 섹션에서는 카오스 엔지니어링의 이점을 설명하고, 카오스 엔지니어링 실험을 수행하는 방법을 설명하고, 조직에서 대규모 카오스 엔지니어링을 구현하기 위해 취할 수 있는 접근 방식을 설명합니다. 카오스 엔지니어링 실험의 템플릿으로 사용할 수 있는 샘플 실험 계획 및 실험 결과 문서도 포함되어 있습니다.

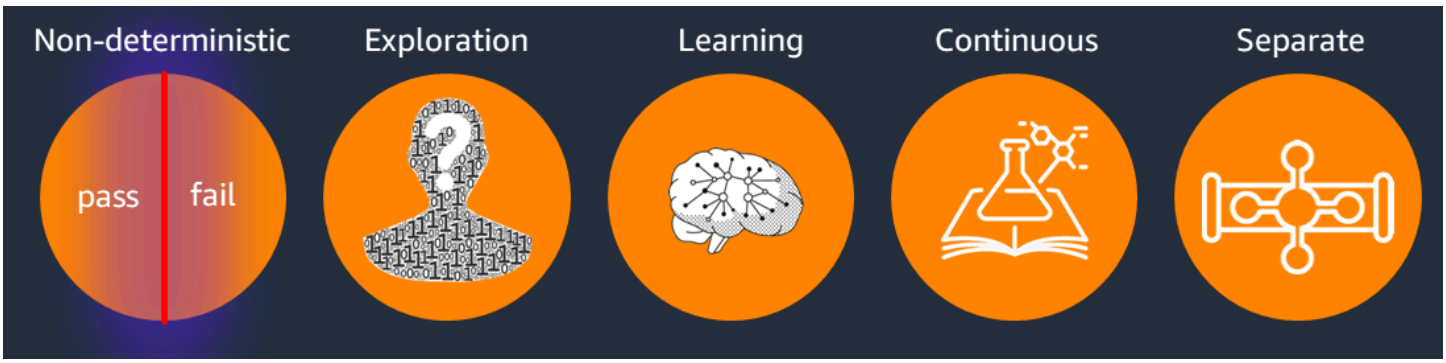
다음 섹션에서는 카오스 엔지니어링의 특성이 단위, 연기 또는 통합 테스트와 같은 기존 복원력 테스트와 어떻게 다른지 살펴봅니다.

개요

카오스 엔지니어링과 복원력 테스트 비교

복원력 테스트는 결정적입니다. 즉, 애플리케이션에서 구현된 회로 차단기, 재시도, 장애 조치 또는 폴백과 같은 복원력 메커니즘에 대해 알려진 특성을 검증합니다. 이러한 애플리케이션 구성 요소가 사용자 영향을 최소화하거나 전혀 주지 않으면서 제어된 중단을 어떻게 흡수하는지 확인합니다. 따라서 복원력 테스트는 통과/실패 결과를 생성하기 위해 애플리케이션 구성 요소에 주입되는 알려진 장애 모드의 검증에 중점을 둡니다. 복원력 태세에 회귀가 발생하지 않도록 파이프라인의 한 단계로 복원력 테스트를 지속적으로 실행해야 합니다. 복원력 테스트에서는 실제 구성 요소에 대해 테스트를 실행하지 않고 특정 구성 요소를 시뮬레이션하는 모APIs를 실행하는 경우가 많습니다. 이 접근 방식을 사용하면 제어된 환경에서 장애 시나리오를 일관되고 재현 가능하게 테스트할 수 있으므로 자동화된 파이프라인 통합 및 회귀 테스트에 적합합니다.

반면 카오스 엔지니어링은 비결정적입니다. 즉, 가설 기반이며 애플리케이션과 해당 종속성(사람, 프로세스 및 기술)이 예상치 못한 장애 모드를 어떻게 흡수, 적응 및 복구하는지에 대한 멘탈 모델을 검증합니다. 따라서 카오스 엔지니어링은 결함을 조기에 발견하고 대규모 이벤트로 전환되기 전에 이를 해결하는 것을 목표로 알 수 없는 장애 모드의 end-to-end 확인에 중점을 둡니다. 카오스 엔지니어링은 지속적인 학습을 촉진하며 별도의 카오스 파이프라인 또는 임시 실험을 통해 연습해야 합니다. 이를 통해 코드 배포 시 개발자의 생산성을 저해하지 않고 언제든지 여러 실험을 실행할 수 있습니다.



카오스 엔지니어링 프로세스는 종종 카오스 게임 데이로 시작합니다. 카오스 게임 데이는 팀이 의도적으로 제어된 장애 또는 장애를 애플리케이션에 주입하는 전용 이벤트입니다. 게임 데이는 점진적입니다. 하위 수준 환경(예: 개발 또는 테스트)에서 시작되며 신뢰도가 높아지면 점차 상위 수준 환경(예: 스테이징 및 사전 프로덕션)으로 이동합니다. 팀은 이러한 환경을 체계적으로 이동함으로써 주입된 장애가 프로덕션에 도달하기 전에 시스템이 제대로 견딜 수 있는지 확인할 수 있습니다. 이러한 체계적인 진행은 카오스 실험이 프로덕션 환경에서 수행될 때까지 팀이 시스템의 복원력 기능에 대한 상당한 신뢰를 구축하도록 보장합니다. 게임 데이 프로세스는 예상치 못한 프로덕션 중단 시 학습 스트레스를 없애면서 애플리케이션 아키텍처 및 운영 관행의 약점과 취약성을 식별하는 선제적 접근 방식입니다.

카오스 엔지니어링의 가치

오늘날의 세계에서는 복잡한 시스템이 보편적입니다. 금융 시장부터 의료에 이르기까지 우리 라이프의 여러 측면에서 중요한 역할을 합니다. 이러한 시스템은 항상 작동해야 합니다. 그러나 복잡한 시스템은 종종 심각한 결과를 초래할 수 있는 예상치 못한 이벤트 및 동작에 취약합니다. 조직은 중단이 발생할지 여부를 걱정하는 대신 중단을 계획해야 합니다. 핵심 또는 미션 크리티컬 비즈니스 서비스에 시나리오 테스트를 적용하여 이를 수행할 수 있습니다. 카오스 엔지니어링이 작동하는 곳입니다.

카오스 엔지니어링은 위험을 완화하고 복원력을 개선하는 데 도움이 되는 복잡한 시스템을 관리하는 접근 방식을 제공합니다. 카오스 실험을 준비하려면 팀이 시스템 동작에 대한 가설을 개발해야 합니다. 이 가설은 시스템 구축 방식과 운영 방식에 대한 이해를 심화합니다. 이 준비를 통해 발견되지 않은 상태로 남을 수 있는 멘탈 격차, 아키텍처 인사이트 및 운영 지식이 드러나는 경우가 많습니다. 복잡한 시스템이 장애에 어떻게 반응하는지 이해함으로써 카오스 엔지니어링은 시스템 설계 및 관리에서 투명성과 책임을 강화합니다. 조직이 카오스 엔지니어링을 더 자주 수행할수록 운영에 더 잘 대비할 수 있습니다. 카오스 엔지니어링을 사용하면 사용자 영향을 최소화하거나 전혀 주지 않으면서 구성 요소 장애를 견딜 수 있는 복원력 있는 애플리케이션을 설계하는 모범 사례를 수립할 수 있습니다. 이렇게 하면 중요한 애플리케이션이 예상 서비스 수준 및 내충격성 내에서 작동하는 동시에 자체 시스템에 대한 팀의 지식을 지속적으로 높일 수 있습니다.

불리한 조건 준비

를 빌드할 때 Amazon Elastic Compute Cloud(Amazon EC2)와 같은 영역 서비스, Amazon Simple Storage Service(Amazon S3)와 같은 리전 서비스, AWS Identity and Access Management (IAM)과 같은 글로벌 서비스, 타사 서비스형 소프트웨어(SaaS) 서비스, 온프레미스 서비스 등 다양한 유형의 서비스를 AWS사용합니다. 각 서비스 유형은 고려해야 할 다양한 장애 도메인을 노출합니다. 자체적으로 발생한 이벤트 또는 조직이 제어할 수 없는 타사에서 발생한 이벤트에 어떻게 대비하나요?

애플리케이션이 부정적인 조건에 어떻게 대응할 수 있는지 이해하는 데 도움이 되도록 AWS Fault Injection Service (AWS FIS)를 사용할 수 있습니다. AWS FIS 는 제어된 방식으로 결함 주입 실험을 실

행하기 위한 완전 관리형 서비스입니다. 이 서비스를 사용하여 가용 영역 전원 중단 및 리전 간 연결 문제와 같은 AWS제공 시나리오를 주입하거나 서비스에서 제공하는 다양한 결함 작업을 함께 연결하여 자체 실험을 구축할 수 있습니다.를 AWS FIS 사용하면 팀이 애플리케이션이 일반적인 결함에 어떻게 반응하는지 지속적으로 연습하고 학습하고 결함을 감지할 때 결함을 해결할 수 있습니다.

제어된 카오스 엔지니어링 연습

카오스 제어 실험의 주요 원칙은 다음과 같습니다.

- 프로덕션 환경과 유사한 환경으로 시작합니다.
- 실험에 대한 가설 및 중지 조건을 설정합니다.
- 가볍게 시작하세요.
- 카오스 실험을 연습합니다.
- 영향 범위를 설정합니다.
- 서비스 기준을 파악합니다.
- 실험을 예약합니다.
- 먼저 문제를 해결한 다음 실험합니다.
- 실험을 면밀히 모니터링합니다.
- 결과에서 배웁니다.
- 조사 결과의 우선순위를 지정하고 문제를 해결하며 확인합니다.
- 조직 전체에 학습 내용을 전파합니다.

카오스 엔지니어링을 성공적으로 확장하려면 카오스 실험을 제어된 방식으로 구현해야 합니다. 를 사용할 때 Amazon CloudWatch 경보를 사용하여 중지 조건을 생성할 AWS FIS수 있습니다. 이러한 조건을 실험 템플릿에 통합하여 범위를 벗어나 마지막으로 알려진 상태로 롤백될 경우 실험이 중지되도록 할 수 있습니다. AWS FIS 또한는 안전 레버를 제공합니다. 이러한 레버를 사용하면는 다중 계정 실험을 AWS 리전포함하여의 계정에서 실행 중인 모든 실험을 AWS FIS 중지 및 롤백하고 새 실험이 시작되지 않도록 합니다. 이렇게 하면 거래 시간, 판매 이벤트 또는 제품 출시와 같은 특정 기간 동안 또는 애플리케이션 상태 경보에 대한 응답으로 오류가 주입되는 것을 방지할 수 있습니다. 안전 레버는 수동으로 해제될 때까지 계속 작동합니다.

카오스 실험을 수행할 때는 특히 실험이 프로덕션 환경에 있는 애플리케이션에 영향을 미칠 가능성이 있는 경우 환경에서 원치 않는 부작용을 방지하기 위한 보호 조치를 정의해야 합니다. 실험을 계획할 때 환경의 다른 애플리케이션에 미칠 수 있는 부정적인 영향을 예상합니다. 예를 들어 다른 애플리케이션은 실험의 일부인 애플리케이션에서 잘못된 메시지를 수신하거나, 요청 볼륨이 많거나, 인프라를 공

유하는 경우 리소스 경합이 발생할 수 있습니다. 실험을 실행하기 전에 이러한 위험을 문서화하고 알려진 문제나 허용되지 않는 문제를 해결합니다.

카오스 엔지니어링 시작하기

실험을 수행하기 전에 카오스 엔지니어링 사례를 최대한 활용하기 위한 몇 가지 필수 사항을 마련하는 것이 좋습니다. 이러한 필수 사항은 다음과 같습니다.

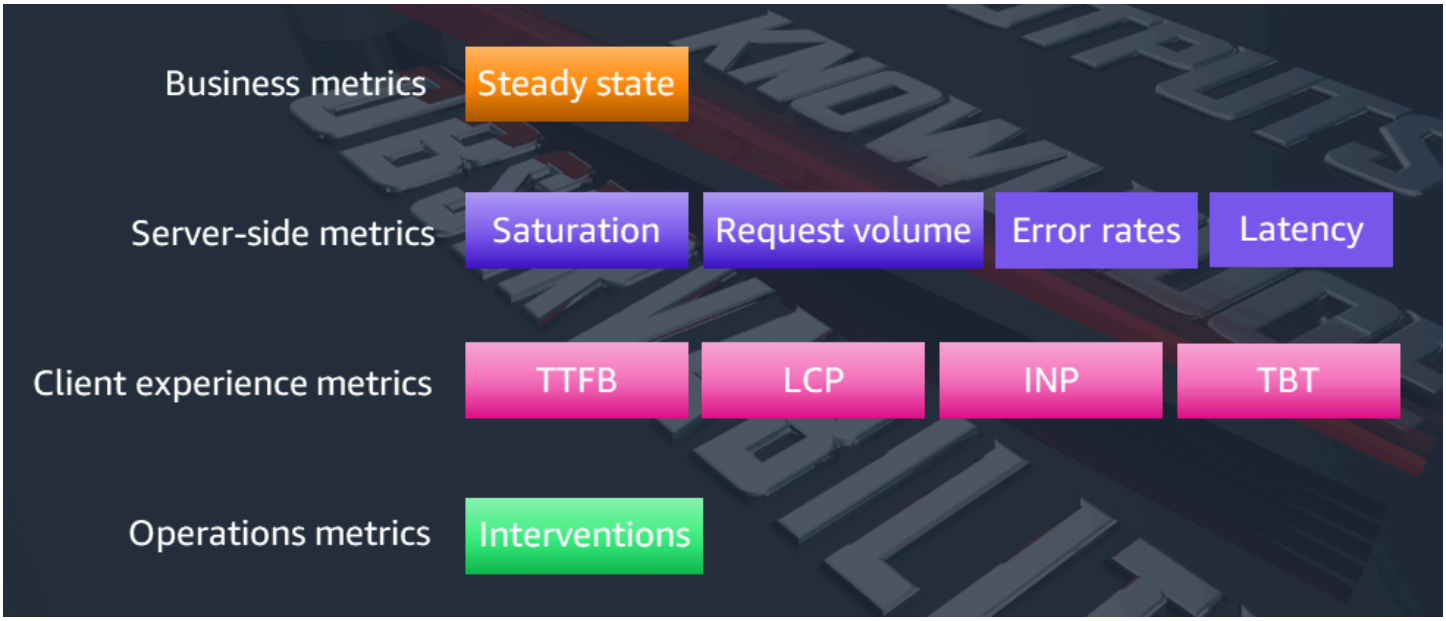
- 관찰성(지표, 로깅, 요청 추적)
- 탐색하려는 실제 이벤트 또는 결함 목록
- 리더십 동의를 통한 조직 복원력 후원
- 카오스 실험을 실행할 때 발견된 새로운 기능보다 비즈니스에 미치는 잠재적 영향을 기반으로 중요한 결과의 우선 순위 지정

카오스 실험에 대한 관찰성

지표, 로깅 및 요청 추적으로 구성된 관찰성은 카오스 엔지니어링에서 중요한 역할을 합니다. 실험을 실행할 때 비즈니스 지표, 서버 측 지표, 클라이언트 경험 지표 및 운영 지표를 이해하는 것이 좋습니다. 관찰성이 없으면 안정 상태 동작을 정의하거나 애플리케이션에 대한 가설이 true로 유지되는지 확인하는 의미 있는 실험을 만들 수 없습니다.

Metrics

다음 다이어그램은 다양한 유형의 애플리케이션에 대한 카오스 실험을 위해 추적할 수 있는 지표 유형을 보여줍니다.



- 비즈니스 지표 - 정상 상태는 시스템의 정상 작동을 나타내며 비즈니스 지표에 의해 정의됩니다. 초당 트랜잭션(TPS), 초당 클릭 스트림, 초당 주문 또는 유사한 측정으로 표현할 수 있습니다. 애플리케이션이 예상대로 작동하면 안정 상태를 나타냅니다. 따라서 실험을 실행하기 전에 애플리케이션이 정상인지 확인합니다. 장애의 백분율이 허용 한도 내에 있을 수 있으므로 장애가 발생할 때 애플리케이션에 영향을 미치지 않는다는 의미는 아닙니다. 안정 상태는 사용자의 기준입니다. 예를 들어 결제 시스템의 안정 상태는 성공률이 99%이고 왕복 시간이 500ms인 300TPS의 처리로 정의될 수 있습니다. 시각적으로 정상 상태를 EKG(심전도)로 생각하세요. 시스템의 안정 상태가 갑자기 변동하면 서비스에 문제가 있는 것입니다.
- 서버 측 지표 - 실험 중에 리소스의 성능을 이해하려면 실험 전, 도중, 후에 리소스의 성능에 대한 인사이트가 필요합니다. 리소스가에 미치는 영향을 측정하려면 Amazon CloudWatch를 사용할 AWS 수 있습니다. CloudWatch는 애플리케이션을 모니터링하고, 성능 변화에 대응하고, 리소스 사용을 최적화하고, 운영 상태에 대한 인사이트를 제공하는 서비스입니다. 실험 중에는 포화도, 요청 볼륨, 오류율 및 지연 시간과 같은 서버 측 지표를 캡처해야 합니다.
- 고객 경험 지표 - 에서는 CloudWatch RUM을 사용하여 Locust AWS, Grafana k6, Selenium 또는 Puppeteer와 같은 도구를 통해 사용자 요청을 시뮬레이션하여 실제 사용자 지표를 캡처할 수 있습니다. 실제 사용자 지표는 카오스 엔지니어링 실험을 수행하는 조직에 매우 중요합니다. 실제 사용자가 실험 중에 어떤 영향을 받는지 모니터링하면 팀은 결함과 중단이 프로덕션 고객에게 어떤 영향을 미치는지 정확하게 파악할 수 있습니다. 주요 클라이언트 경험 지표는 첫 번째 바이트까지의 시간(TTFB), 가장 큰 콘텐츠 페인트(LCP), 다음 페인트와의 상호 작용(INP), 총 차단 시간(TBT)입니다.
- 작업 지표 - 중재는 복제 컨트롤러 또는 자동 조정과 같은 메커니즘을 사용하여 포드, 작업 또는 EC2 인스턴스를 재부팅하는 동안 성공적인 클라이언트 요청 지연 시간과 같은 자동화된 방식으로 장애를 얼마나 성공적으로 완화하는지 측정합니다. 장애 발생 시 자동으로 개입할 수 있다는 것은 좋은 사용자 경험과 직접적인 상관관계가 있습니다. 시간이 지남에 따라 이러한 완화 메커니즘에 드리프트가 있는지 이해하는 것이 중요합니다. 성공 및 실패한 자동 완화 모두에 대한 지표를 정의하면 시스템 전체에서 잠재적 회귀를 식별하는 데 도움이 되는 가이드포스트를 생성할 수 있습니다.

로깅

중앙 집중식 로깅은 카오스 실험 전, 도중, 후에 애플리케이션의 구성 요소 상태를 이해하는 데 중요합니다. 모든 애플리케이션 구성 요소에서 로그를 수집하여 실험이 주입될 때 각 구성 요소가 수행한 작업에 대한 통합 보기를 구축하는 것이 좋습니다. 이를 통해 end-to-end 실험 흐름을 명확하게 파악할 수 있습니다.

요청 추적

요청 추적을 사용하면 애플리케이션의 구성 요소에서 단일 요청의 흐름을 관찰하여 주입된 장애가 시스템 및 해당 종속성에 미치는 영향을 포괄적으로 이해할 수 있습니다. 요청을 추적하면 장애가 다양한 서비스 및 구성 요소를 통해 전파되는 방식을 확인할 수 있으므로 중단 범위를 더 잘 평가할 수 있습니다. 요청을 추적하려면를 사용할 AWS수 있습니다AWS X-Ray.

카오스 실험에 주입하는 실패 시나리오

애플리케이션에 일반적인 결함을 주입하는 목적은 애플리케이션이 이러한 예상치 못한 이벤트에 어떻게 반응하는지 이해하여 완화 메커니즘을 생성하고 이러한 결함에 대해 시스템을 복원할 수 있도록 하는 것입니다. 또한 카오스 엔지니어링을 사용하여 과거 장애 시나리오를 재생하여 완화 메커니즘이 예상대로 계속 작동하고 시간이 지남에 따라 드리프트되지 않았는지 확인해야 합니다.

카오스 엔지니어링 실험을 계획할 때 다음 이벤트를 고려하세요.

실패 모드	설명
서버 장애	EC2 인스턴스를 재부팅하고 Kubernetes 포드 또는 Amazon Elastic Container Service(Amazon ECS) 작업을 삭제하여 애플리케이션이 이러한 충돌에 어떻게 반응하는지 파악합니다.
API 오류	AWS 및 자체 서비스 APIs에 결함을 주입하여 애플리케이션 동작을 이해합니다.
네트워크 문제	지연 시간 또는 혼잡을 도입하거나 연결을 차단하여 실제 네트워크 문제를 모방합니다.
AWS 가용 영역 장애	전체 가용 영역의 장애를 재생하여 영역 간 복구를 확인합니다.
AWS 리전 연결 장애	에서 네트워크 장애를 재생 AWS 리전 하여 보조 리전의 리소스가 이러한 이벤트에 어떻게 반응하는지 확인합니다.
데이터베이스 실패	데이터베이스 복제본 또는 손상된 데이터를 장애 조치하거나 데이터베이스 인스턴스에 연결할

실패 모드	설명
	수 없도록 하여 애플리케이션 및 복구 전략에 미치는 영향을 파악합니다.
데이터베이스 및 Amazon S3 복제에서 일시 중지	가용 영역에서 데이터베이스 또는 Amazon Simple Storage Service(Amazon S3) 복제 AWS 리전 를 일시 중지하거나 다운스트림 애플리케이션 영향을 파악합니다.
스토리지 성능 저하	I/O를 일시 중지하거나, Amazon Elastic Block Store(Amazon EBS) 볼륨을 제거하거나, 파일을 삭제하여 데이터 내구성 및 복구를 확인합니다.
종속성 장애	타사 서비스를 포함하여 의존하는 다운스트림 또는 업스트림 서비스의 성능을 저하시키거나 저하시켜 end-to-end 흐름과 영향을 파악합니다.
트래픽 급증	사용자 트래픽의 스파이크를 생성하여 자동 조정 기능을 테스트하고 콜드 부팅 시간이 전체 애플리케이션 상태에 어떤 영향을 미칠 수 있는지 확인합니다.
리소스 소진	CPU, 메모리 및 디스크 공간을 최대화하여 애플리케이션의 정상적인 성능 저하를 확인합니다.
계단식 실패	다운스트림 애플리케이션 및 구성 요소로 캐스케이드되는 기본 장애를 시작합니다.
잘못된 배포	문제가 있는 변경 사항 또는 구성을 롤아웃하여 롤백 메커니즘을 확인합니다.

조직 복원력 후원

카오스 엔지니어링은 조직 전체에 적용될 때 가장 큰 가치를 제공합니다. 조직 전체에서 복원력 목표를 설정하고, 도메인에 대한 두려움, 불확실성 및 의심을 제거하고, 모든 사람의 책임을 복원하는 변환 프로세스를 시작할 수 있는 경영진 후원자와 협력하는 것이 좋습니다.

카오스 엔지니어링 관행을 구축하는 비즈니스 사례를 지원하려면 카오스 엔지니어링 노력을 중요한 비즈니스 프로젝트에 연결합니다. 복원력을 가속화를 위한 자산과 동인으로 만들면 시간 경과에 따른 성공을 추적하는 데 도움이 됩니다. 월별 또는 분기당 중요한 인시던트 수, 평균 복구 시간, 이러한 인시던트가 고객과 조직에 미치는 영향부터 시작합니다. 카오스 엔지니어링 실험에 대응하여 애플리케이션 스택에서 개선이 이루어지므로 6~12개월 동안 인시던트 수를 줄이도록 팀과 목표를 설정합니다.

인시던트가 매우 반복적인지 측정합니다. 예를 들어 클라이언트가 신뢰할 수 있는 연결을 설정할 수 없기 때문에 만료된 TLS 인증서가 가동 중지를 초래한다고 가정해 보겠습니다. 여러 TLS 인증서 만료로 인해 1년에 여러 인시던트가 발생하는 경우 TLS 인증서 만료 실험을 실행하여 팀이 알림을 받거나 이러한 문제를 자동으로 완화할 수 있는지 확인할 수 있습니다. 이렇게 하면 이러한 결함에 대한 복원력을 높일 수 있습니다.

시간 경과에 따른 카오스 엔지니어링의 진행 상황을 추적하려면 다음 지표를 캡처하여 애플리케이션의 수명 주기 전반에 걸쳐 카오스 엔지니어링의 가치를 강조할 수 있습니다.

- 인시던트 발생률 감소 - 시간 경과에 따른 프로덕션 인시던트 수를 추적하고이 수를 카오스 엔지니어링 채택과 연관시킵니다. 심각한 인시던트의 발생률은 감소할 것으로 예상됩니다.
- 평균 해결 시간(MTTR) 개선 - 인시던트를 해결하는 데 걸리는 평균 시간을 계산하고이 데이터를 추적하여 시간이 지남에 따라 카오스 엔지니어링으로 개선되는지 확인합니다.
- 애플리케이션 가용성 향상 - 카오스 실험을 통해 애플리케이션 복원력이 증가함에 따라 서비스 수준 지표를 사용하여 가용성 개선을 보여줍니다.
- 시장 출시 시간 단축 - 카오스 엔지니어링은 애플리케이션이 복원력이 있음을 알기 때문에 혁신적인 제품을 더 빠르게 출시할 수 있다는 확신을 줄 수 있습니다. 제품 릴리스 속도 증가를 추적합니다.
- 운영 비용 절감 - 카오스 관행이 적용됨에 따라 애플리케이션 관리를 위한 알림 노이즈, 운영 부하 및 수동 작업과 같은 지표가 감소하는지 정량화합니다.
- 신뢰도 향상 - 개발자, 사이트 신뢰성 엔지니어(SREs) 및 기타 기술 인력을 조사하여 카오스 엔지니어링이 애플리케이션 복원력에 대한 신뢰도를 높였는지 확인합니다. 인식이 중요합니다.
- 고객 경험 개선 – 카오스 엔지니어링을 서비스 중단, 롤백 및 중단 감소와 같은 고객에게 긍정적인 결과에 연결합니다.

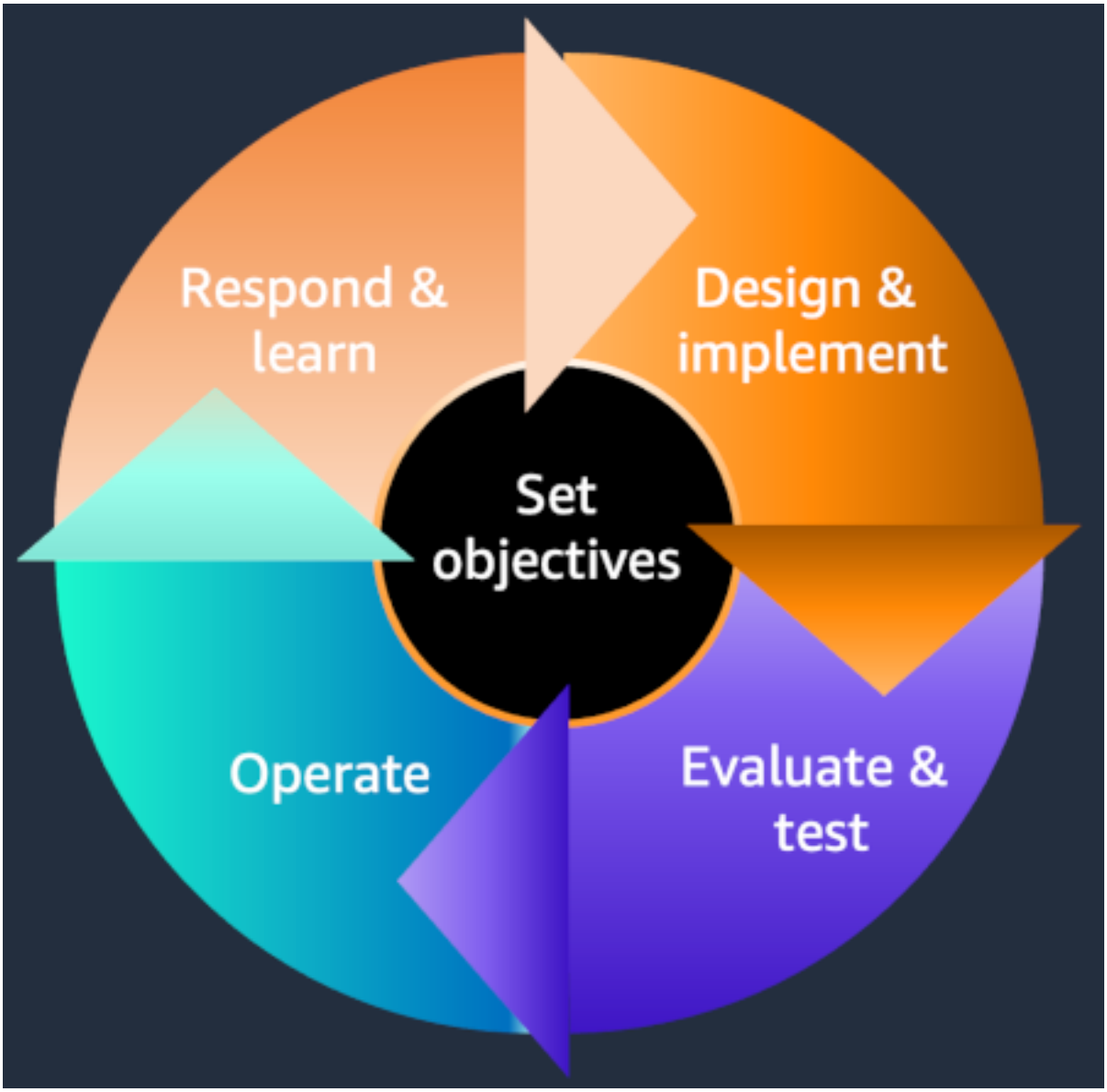
문제 해결 우선 순위 지정

카오스 실험을 수행할 때 애플리케이션이 의도한 대로 작동하지 않는 개선 영역을 식별할 수 있습니다. 이러한 항목의 수정은 백로그에서 작동하며 기능 개발과 같은 다른 작업과 함께 우선 순위를 지정해야 합니다. 향후 실패를 방지하려면 이러한 개선 사항에 시간을 할애하는 것이 좋습니다. 이러한 학습 및 문제 해결 작업은 발생할 수 있는 영향 수준에 따라 우선순위를 정하는 것이 좋습니다. 애플리케이션의

복원력 또는 보안에 직접적인 영향을 미치는 결과는 고객에게 영향을 미치지 않도록 새로운 기능보다 우선해야 합니다. 팀이 기능 개발보다 문제 해결 작업의 우선순위를 정하는 데 어려움을 겪는 경우 경영진 후원자에게 문의하여 비즈니스 위험 허용 범위를 기반으로 우선순위를 설정하는 것이 좋습니다.

에서 카오스 엔지니어링 구현 AWS

카오스 엔지니어링은 다음 다이어그램과 같이 AWS 복원력 수명 주기의 평가 및 테스트 단계의 일부입니다. 분산 애플리케이션은 다른 애플리케이션 또는 클라이언트와 별도로 작동하지 않으므로 전체 복원력 수명 주기를 검토하는 것이 좋습니다. 네트워크가 발전하고, 업스트림 및 다운스트림 애플리케이션이 변화하고, 시간이 지남에 따라 클라이언트 사용량이 변화함에 따라 분산 애플리케이션에서는 변화가 일정합니다.



애플리케이션에 대한 이러한 변경 사항이 복원력에 어떤 영향을 미칠 수 있는지 이해하려면 카오스 엔지니어링을 day-to-day 작업의 일부로 삼으세요. 카오스 실험을 다양한 방식으로 구현할 수 있습니다.

- 임시 - 카오스 실험을 일회성 실험으로 수행하여 특정 문제나 질문을 해결할 수 있습니다.
- 카오스 게임 데이 - 애플리케이션의 신뢰성과 복원력을 확인하도록 설계된 구조화되고 반복적인 이벤트입니다. 카오스 게임 데이의 목적은 사람, 프로세스 및 기술 전반의 잠재적 복원력 문제 또는 결함을 식별하고 인시던트 식별, 완화 및 대응을 위한 프로세스와 절차를 연습하는 것입니다.
- 카오스 파이프라인 - 지속적 통합 및 지속적 전달(CI/CD)은 새로운 기능을 구축하고 환경 전체에 안전하게 배포하는 것입니다. 카오스 엔지니어링 실험을 구현하려면 CI/CD 파이프라인과 별도의 카오스 파이프라인을 생성합니다. 이유를 이해하기 위해 다운스트림 구성 요소에 증가하는 패킷 손실을 주입하는 단일 카오스 엔지니어링 실험을 CI/CD 파이프라인에 추가한다고 가정해 보겠습니다. 이 실험은 3회 실행되며 매번 완료하는 데 5분이 걸립니다. 패킷 손실은 실행마다 10%에서 20%로, 30%로 증가하며, 실험을 완료하는 데 전체적으로 15분이 걸립니다. 병렬 배포가 100개 있는 경우 단일 실험이 완료될 때까지 1500분을 기다려야 합니다. 실험을 10회 실행하면 개발자에게 미치는 영향을 감당할 수 없습니다. 대규모로 카오스 엔지니어링에는 소프트웨어 개발 수명 주기(SDLC) 프로세스와 병렬로 실험을 실행할 수 있는 자체 파이프라인이 필요합니다.
- 카나리아 배포 - 카나리아는 카나리아 실험을 위한 테스트 환경을 제공합니다. 적은 비율의 트래픽을 카나리아 서비스로 전달하거나 트래픽 미러링 또는 재생과 같은 방법을 사용하여 안정적인 프로덕션 시스템에 영향을 주지 않고 새 인프라 또는 코드 변경을 확인할 수 있습니다. 최종 사용자에게 미치는 영향 범위를 제한할 수 있으므로 카나리아에 대해 실험을 실행하고 필요에 따라 결함을 주입할 수 있습니다.
- 예약된 실험 - 실험을 예약하여 애플리케이션의 예측 가능한 복구 메커니즘을 확인할 수 있습니다. 예약된 실험을 사용하여 일반적으로 알려진 이벤트를 재생하여 자동 조정 그룹 뒤의 EC2 인스턴스 종료, Kubernetes 포드 제거 또는 Amazon ECS 작업 삭제와 같은 이벤트에서 시스템이 복구되는 방법을 캡처합니다.

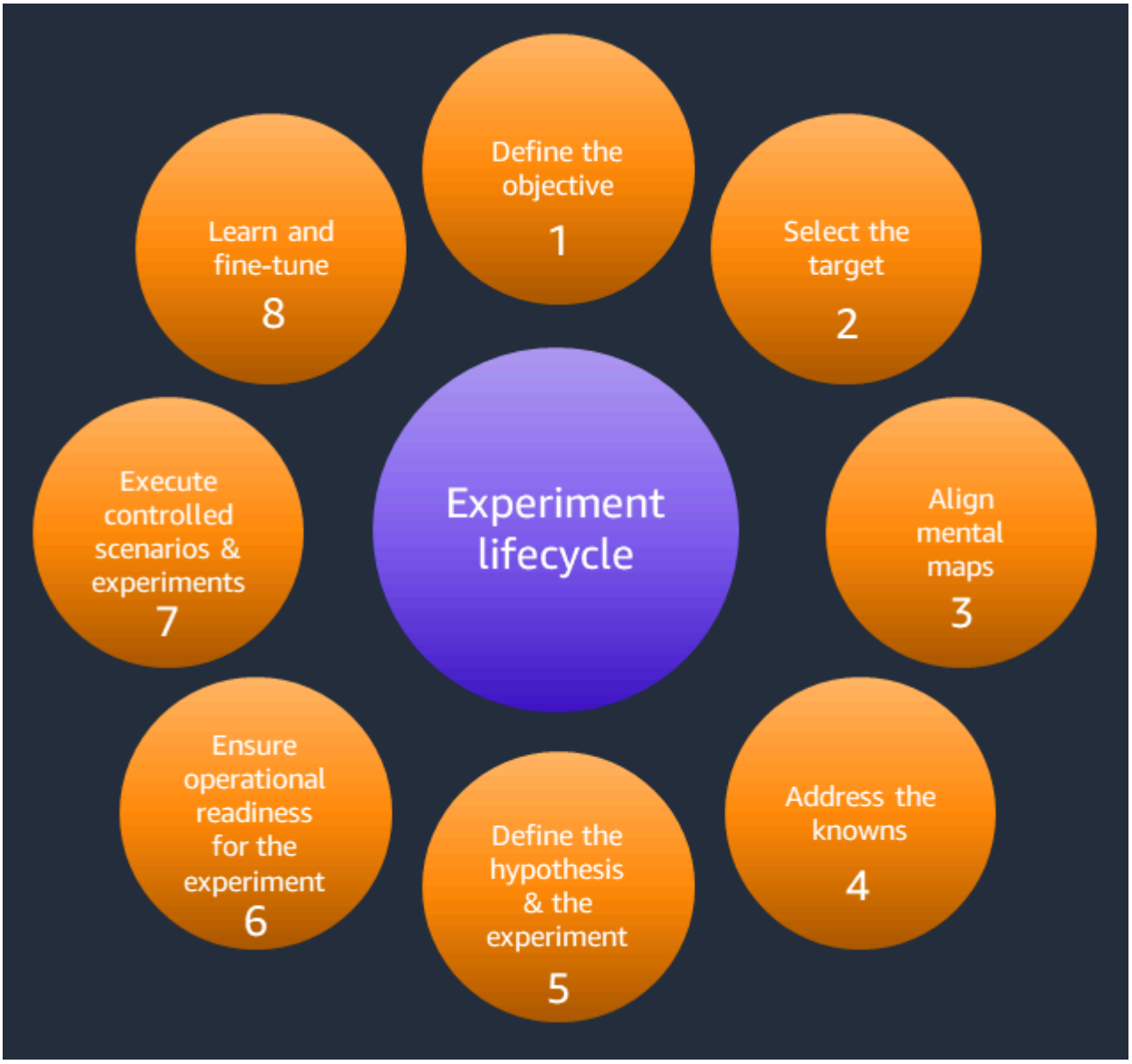
지속적인 카오스 엔지니어링 실험 수명 주기

이전 단원에서 설명한 대로 다양한 방식으로 카오스 엔지니어링 실험을 구현할 수 있습니다. 모든 경우에 의미 있는 카오스 실험을 구축하는 핵심은 애플리케이션, 과거 인시던트 및 구현된 문제 해결을 이해하고 복원력 또는 보안과 같은 조사 영역을 명확하게 이해하는 것입니다. 애플리케이션에 대한 지식은 애플리케이션의 잠재적 약점에 대한 가설을 수립하고 장애가 주입될 때 애플리케이션의 탐지, 해결 및 복구 방법을 이해하는 데 도움이 됩니다.

카오스 실험 수명 주기에는 다음 단계가 포함됩니다.

1. 실험의 목표를 정의합니다.
2. 대상 애플리케이션을 선택합니다.
3. 멘탈 맵을 정렬합니다.
4. 애플리케이션의 알려진 문제를 해결합니다.
5. 가설과 실험을 정의합니다.
6. 실험의 운영 준비 상태를 확인합니다.
7. 제어된 시나리오 및 실험을 실행합니다.
8. 실험에서 배우고 미세 조정합니다.

이러한 단계는 다이어그램에 설명되어 있으며 다음 단원에서 설명합니다.



목표 정의 및 기대치 설정

각 실험 전에 목표와 기대치가 구체적이고 측정 가능하며 달성 가능하고 관련성이 있고 시간 제한이 있는지 확인합니다. 다음을 명확하게 정의합니다.

- 시스템 및 서비스의 잠재적 장애 또는 약점을 식별하여 사용자에게 어떤 영향을 미칠 수 있는지 파악합니다. 여기에는 서버 충돌, 네트워크 장애 또는 소프트웨어 버그와 같은 가능한 장애 모드를 식별하고 시스템의 전체 성능 및 신뢰성에 미치는 잠재적 영향을 평가하는 것이 포함됩니다.
- 시스템 및 서비스에 대한 주요 위험 지표(KRIs)를 정의하여 장애의 영향을 정량화합니다. 여기에는 지연 시간, 처리량 및 오류율과 같은 지표가 안정 상태에서 벗어나는 경우 실패의 영향을 측정하는

것이 포함됩니다. 이러한 편차의 영향을 이해하면 비즈니스 위험을 기반으로 장애를 완화하기 위한 노력의 우선순위를 정할 수 있습니다.

- 장애를 완화하거나 방지하기 위한 전략을 개발하고 확인합니다. 여기에는 중복성, 오류 수정 또는 대체 전략과 같은 잠재적 솔루션 식별과 제어된 환경에서의 효과 테스트가 포함됩니다. 이러한 전략을 확인하면 장애를 예방하거나 완화하는 데 효과적이며 확신을 가지고 프로덕션 시스템에 배포할 수 있습니다.
- 인시던트 대응 및 재해 복구 프로세스를 개선합니다. 제어된 환경에서 장애를 재생하면 인시던트 대응 프로세스를 테스트하고, 잠재적 병목 현상 또는 격차를 식별하고, 재해 복구 절차를 개선할 수 있습니다. 이렇게 하면 예기치 않은 장애가 발생할 경우 빠르고 효과적으로 대응할 수 있습니다.

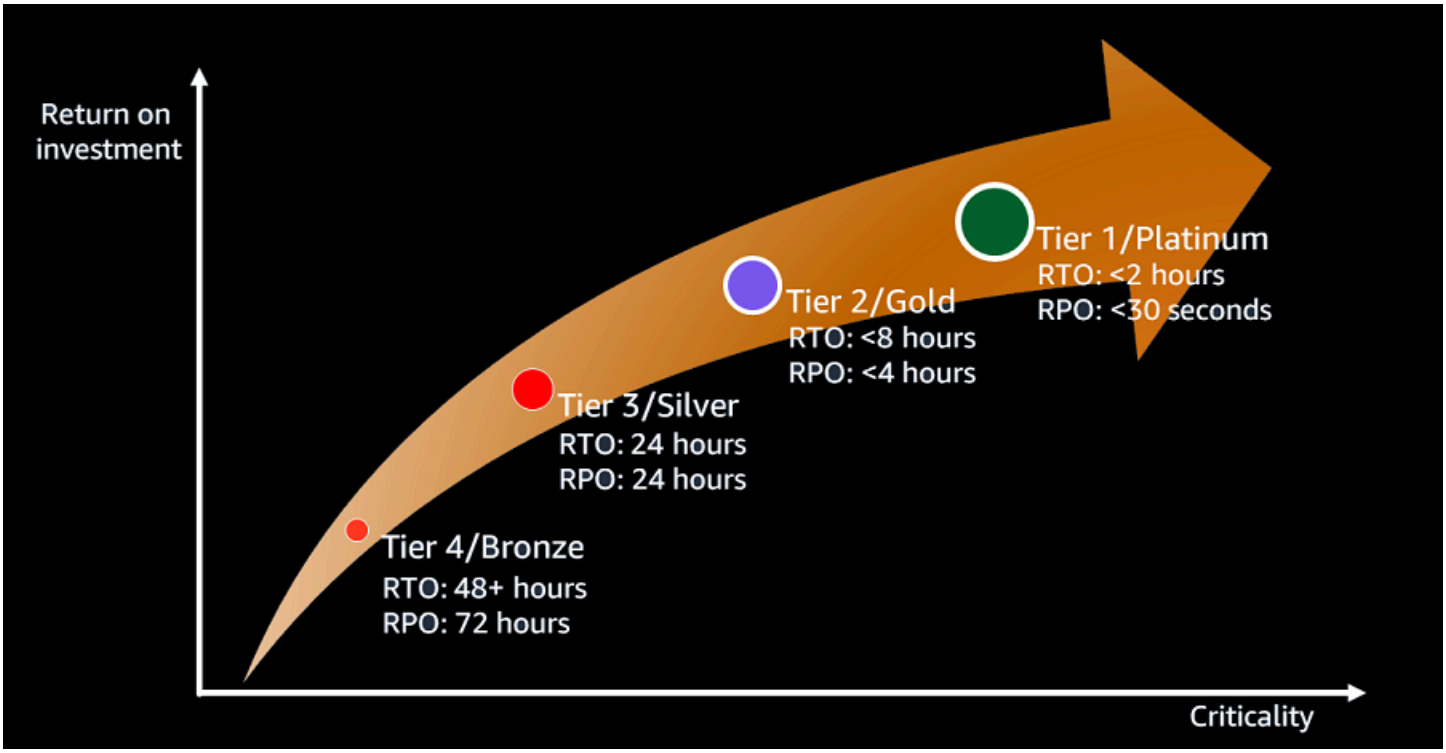
대상 애플리케이션 선택

카오스 엔지니어링은 강력한 기술이지만 가치를 극대화하려면 신중한 우선순위 지정이 필요합니다. 카오스 엔지니어링 작업에 집중할 위치를 결정할 때는 먼저 비즈니스의 중요한 서비스를 고려하세요. 팀에게 소프트웨어 개발 수명 주기 단계를 반복하고 먼저 테스트 환경에 결함을 주입하도록 요청합니다. 비즈니스 크리티컬 애플리케이션은 수익, 고객 경험 및 핵심 운영과 직접적인 관련이 있습니다. 이러한 서비스에 대한 혼란스러운 실험을 통해 조직과 잠재적으로 전체 시장에 심각한 영향을 미칠 수 있는 취약성이 해결되지 않는 경우 발견할 수 있습니다. 예를 들어 먼저 거래 시스템 또는 주문 시스템과 같은 고객 대면 서비스에 집중합니다. 이러한 중앙 서비스의 우선순위를 지정하면 시간 투자당 가장 많은 보호를 제공합니다.

중요한 서비스 후에는 데이터베이스, 메시지 대기열, 네트워크 및 공유 서비스 APIs와 같은 기본 구성 요소를 살펴봅니다. 이러한 구성 요소는 조직 전체에서 공유 구성 요소 또는 서비스로 사용될 수 있으므로 장애가 발생하면 광범위한 문제가 발생합니다. 인프라 서비스의 복원력을 확인하면 인프라 서비스 위에 종속 애플리케이션이 충돌하지 않을 것이라는 확신을 얻을 수 있습니다. 예를 들어 Kafka 클러스터를 중단하는 카오스 엔지니어링 실험은 다운스트림 애플리케이션의 내결함성에 대해 많은 것을 드러냅니다. 시스템 인프라는 고객을 직접 대면하지는 않지만 주요 카오스 엔지니어링 대상입니다.

사람, 프로세스, 시설 정보 및 서드 파티 종속성의 정신적 격차를 매핑하는 것을 잊지 마세요. 이러한 격차가 조직의 내충격성 목표에 맞지 않으면 심각한 중단을 일으킬 수 있기 때문입니다. 카오스 엔지니어링의 ROI를 측정하는 방법에 대한 자세한 내용은 전략적 필요성으로 카오스 엔지니어링에 투자 전략 문서에서 카오스 엔지니어링의 ROI 정량화를 참조하세요.

다음 다이어그램은 다양한 서비스 계층에서 카오스 실험을 실행하기 위한 투자 수익을 보여줍니다.



멘탈 맵 정렬(애플리케이션 검색)

임시 실험 또는 게임 데이를 실행할 때 애플리케이션의 세부 정보를 매핑하는 데 초점을 맞춘 화이트보딩 세션을 개최하여 애플리케이션 검색 프로세스를 시작합니다. (카오스 파이프라인에서 실험을 실행하는 경우 대상 애플리케이션을 정의하여 멘탈 맵을 이미 정렬했을 것입니다.) 멘탈 격차를 이해하는 좋은 방법은 하급 팀원이 먼저 애플리케이션의 다이어그램을 그린 다음 더 많은 고위 직원에게 다이어그램에 점진적으로 추가하도록 요청하는 것입니다. 이렇게 하면 경험 수준 전반의 이해 격차를 발견할 수 있습니다.

다이어그램은 애플리케이션의 직접 업스트림 및 다운스트림 종속성과 중요한 타사 통합을 모두 보여야 합니다. 애플리케이션을 통한 요청의 예상 흐름에 정렬이 있는지 확인합니다. 주요 워크플로와 사용자 여정을 매핑하여 고객이 애플리케이션을 사용하는 방법을 명확하게 파악할 수 있습니다. 시퀀스 다이어그램을 사용하여이 정보를 캡처하는 것이 좋습니다.

이 협업 세션 후 팀은 애플리케이션의 공유된 멘탈 모델, 중요한 종속성 및 모니터링 기능, 정보에 입각한 의사 결정을 내려 잠재적 카오스 실험을 진행하거나 취소할 수 있는 위험에 대한 이해가 있어야 합니다.

애플리케이션의 알려진 문제 해결

카오스 엔지니어링 실험은 애플리케이션의 결함을 사전에 표시하도록 설계되었습니다. 지연 시간 증가, 서버 재부팅 또는 가용 영역 전원 장애와 같은 장애를 주입하여 실제 중단을 허용할 수 있는 애플리케이션의 기능을 확인할 수 있습니다. 그러나이 프로세스는 대상 애플리케이션의 기본 수준의 안정성과 상태를 가정합니다. 이미 문제가 있는 애플리케이션에서 카오스 실험을 실행하면 더 깊은 문제가 마스킹될 위험이 있습니다.

카오스 엔지니어링을 수행하기 전에 팀은 애플리케이션에서 알려진 결함, 버그 및 성능 문제를 해결해야 합니다.

가설 및 실험 정의

애플리케이션 또는 조직 내 다른 애플리케이션에 중단을 일으킨 과거 인시던트는 카오스 실험 아이디어를 위한 훌륭한 소스 역할을 할 수 있습니다. 예를 들어 구성 오류 또는 누락된 복원력 패턴으로 인해 이전 중단이 트리거되었습니까? 카오스 실험을 통해 인시던트 기록을 검토하고 실제 실패의 근본 원인을 재생하는 것은 향후 유사한 문제에 대한 복원력을 개발하는 효과적인 방법입니다.

실험 개념의 또 다른 중요한 소스는 애플리케이션에 가장 익숙한 엔지니어, 아키텍트 및 운영자로부터 직접 얻을 수 있습니다. 팀원이 애플리케이션을 크게 방해할 수 있다고 생각되는 가상의 실패 시나리오를 제출하도록 허용하면 내부자 지식을 기반으로 아이디어를 수집할 수 있습니다. 그런 다음 애플리케이션 팀은 이러한 제안된 시나리오 중 잠재적 영향이 가장 큰 시나리오를 평가하거나 알려지지 않은 가장 큰 위험을 노출할 수 있습니다. 위험도 높고 이해도가 낮은 시나리오에 대한 카오스 실험을 대상으로 하면 중요한 학습을 생성하고 향후 문제를 방지할 수 있습니다.

세 번째 아이디어 출처는 복원력 모델링을 수행하여 식별된 비즈니스 손실로 이어질 수 있는 조건을 예측하는 것입니다. 일부 복원력 모델링 연습에는 복원력 모델을 구축하는 구성 요소 기반 접근 방식이 있는 반면, 다른 연습에는 시스템 기반 접근 방식이 있습니다. 구성 요소 기반 접근 방식은 "구성 요소 x가 과도한 부하를 받거나 실패한 경우 어떻게 됩니까?"라는 질문을 합니다. 그런 다음 복원력 모델을 개발하는 팀은 이러한 시나리오가 더 광범위한 애플리케이션에 미치는 영향을 추측하고 시나리오의 영향을 감지하고 완화하기 위해 현재 시행 중인 모니터링 및 예방 제어를 식별합니다. 또는 시스템 기반 접근 방식은 하향식 프로세스를 따라 "웹 스토어 프런트에서 오래된 인벤토리 수준을 표시하고 있습니다"와 같은 원치 않는 애플리케이션 상태를 강조하고 애플리케이션 팀이 이러한 방식으로 애플리케이션이 동작할 수 있는 조건을 예측하도록 초대합니다.

실험의 운영 준비 상태 확인

앞의 관찰성 섹션에 설명된 대로 애플리케이션 및 동작에 대한 부정적인 조건의 영향을 측정하려면 정량화 가능한 지표가 필요합니다. 애플리케이션의 동작을 측정할 수 있으면 부정적인 조건이 애플리케이션에 영향을 미쳤는지 여부와 규모를 확인할 수 있습니다.

애플리케이션에 영향을 미치는지 여부를 이해하는 가장 좋은 방법은 안정 상태를 측정하는 것입니다. 정상 상태는 정상적인 운영의 모습을 측정하며 일반적으로 특정 애플리케이션의 비즈니스 및 클라이언트 경험 지표와 일치합니다. 다음 단계로 넘어가기 전에 영향을 이해할 수 있는 관찰성이 있는지 확인하고 실험이 예상대로 진행되지 않을 경우 롤백 메커니즘을 준비합니다.

제어된 실험 및 시나리오 실행

에서는 프로덕션 중인 애플리케이션에 대해 초기 카오스 실험을 수행하지 않는 AWS것이 좋습니다. 카오스 실험의 목적은 스트레스가 많은 조건에서 애플리케이션이 작동하는 방식에 대해 새로운 것을 배우는 것입니다. 실험 중에 애플리케이션의 동작을 예측할 수 없으므로 프로덕션 환경에서 처음으로 실험을 수행하면 고객에게 영향을 미칠 수 있습니다. 따라서 항상 실제 사용자에게 영향을 미칠 가능성이 최소인 하위 레벨 환경에서 초기 카오스 실험을 실행한 다음 애플리케이션이 주입된 작업을 흡수, 적응 및 복구할 수 있는지 확인한 후 환경을 반복해야 합니다.

부록에 제공된 실험 계획 문서와 마찬가지로 주요 세부 정보를 캡처하는 문서를 사용하여 각 실험을 철저히 계획합니다. 포함할 중요 필드 중 일부는 정상 상태 정의, 가설 및 실패 주입 방법입니다. 카오스 실험의 계획, 실행 및 분석은 단일 아티팩트에서 다룰 수 있습니다.

실험에 대한 서면 계획을 완료한 후 문서에 설명된 계획된 중단을 주입하는 데 필요한 코드를 준비합니다.

실험 중 잠재적 영향을 캡처하려면 관찰성 메커니즘이 마련되어 있는지 확인합니다. AWS FIS 실험 보고서와 같은 실험 결과를 캡처할 수 있는 자동화된 방법이 아직 없는 경우 실행 중에 메모를 작성하고 대시보드 스크린샷을 캡처하며 실험을 통해 팀을 리드할 팀원을 식별합니다.

학습 및 미세 조정

각 실험이 끝나면 팀으로 모여 카오스 실험을 검토하고 숙고합니다. 비난 없는 사고방식을 유지하기 위해 의식적으로 노력하세요. 목표는 비난하지 않고 최대한의 학습을 도출하는 데 전적으로 초점을 맞춘 개방적이고 건설적인 대화를 하는 것입니다.

먼저 실험에 대한 안정 상태 정의와 가설을 검토합니다. 애플리케이션이 예상대로 작동했나요? 가정을 무효화한 놀람이 있었나요? 실험 중에 애플리케이션이 어떻게 반응했는지, 좋은지, 나쁜지에 대한 관

찰을 논의합니다. 지표, 로그, 스크린샷 등 수집된 데이터는 정확히 어떤 일이 발생했는지에 대한 스토리를 전달해야 합니다.

판단력 대신에이 데이터 검토에 접근하고 학습 내용을 기반으로 애플리케이션 설계, 문서화, 모니터링 또는 기타 기능을 개선할 수 있는 영역을 식별합니다. 이러한 작업 항목은 애플리케이션을 더 복원력 있게 만들기 위한 후속 프로젝트로 캡처됩니다.

이 비난 없는 접근 방식을 통해 무엇이 잘못되었는지, 어떻게 해결할 수 있는지에 대해 솔직한 대화를 나눌 수 있습니다. 관련된 모든 사람의 긍정적인 의도를 가정하고 좋은 결과를 얻기 위해 노력하고 있다고 신뢰합니다. 공유된 목표는 지속적인 학습과 적응을 통한 조직의 성장과 발전입니다. 건설적이고 비난하지 않는 방식으로 수행되는 카오스 실험 검토는 팀이 애플리케이션과 조직을 장기적으로 더 안정적이고 복원력 있게 만드는 귀중한 인사이트를 얻을 수 있는 안전한 공간을 제공합니다. 초점은 사람이 아닌 학습에 있습니다. 팀 전체에 학습 내용을 분산하려면 실험 결과 보고서를 중앙 위치에 게시하고 다른 사람이 학습할 수 있도록 결과를 알립니다.

조직 전체의 카오스 엔지니어링 규모 조정

조직이 카오스 엔지니어링을 채택함에 따라 이를 표준화하고 구현하는 데는 과제가 있습니다. 성숙 초기 단계에서는 팀마다 이전 섹션에 설명된 카오스 엔지니어링 프로세스의 다양한 도구와 변형을 사용할 가능성이 높습니다. 동시에 일부 팀은 잠재적 이점에도 불구하고 카오스 엔지니어링을 우선시하거나 채택하지 않을 수 있습니다. 다음 섹션에서는 이러한 문제를 해결하는 방법에 대한 지침을 제공합니다.

전반적으로 카오스 엔지니어링에 대한 접근 방식은 중앙 집중식 리더십과 분산형 참여 간의 균형을 맞추도록 설계되어야 합니다. 이 균형은 카오스 엔지니어링이 개발 프로세스에 통합되고 학습이 조직 전체에서 공유되도록 하는 데 도움이 됩니다.

카오스 엔지니어링 관행 수립

카오스 엔지니어링 관행을 표준화하면 채택을 가속화할 수 있습니다. 팀 간에 실험에서 배운 내용을 공유하면 카오스 엔지니어링 투자에 대한 수익을 높일 수 있습니다.

카오스 엔지니어링 관행의 일환으로 중앙 집중식 우수성 센터를 구축하거나 주제 전문가 그룹을 구성합니다. 소규모 중앙 집중식 기능인이 팀은 소프트웨어 개발, 인프라, 보안 및 비즈니스 팀 전반에서 작동하고 해당 팀이 사용하는 표준을 유지할 수 있습니다. 간소화를 위해 우수성의 중심을 중앙 집중식 연습 팀이라고 하고, 카오스 엔지니어링을 적용하는 그룹을이 가이드의 나머지 부분에서 연습 팀이라고 합니다.

중앙 집중식 연습 팀의 역할

중앙 집중식 연습 팀은 조직 전체에서 카오스 엔지니어링 사례를 개발하고 구현할 책임이 있습니다. 이들은 실습 팀과 긴밀하게 협력하여 실험을 설계 및 수행하고 실험이 비즈니스에 유용한지 확인합니다. 또한 중앙 집중식 연습 팀은 카오스 엔지니어링을 개발 프로세스에 통합하는 데 도움이 되는 개발, 인프라 및 보안 팀에 지침과 지원을 제공합니다.

중앙 집중식 카오스 엔지니어링 실무 팀의 주요 책임은 다음과 같습니다.

- 활성화 - 중앙 집중식 카오스 엔지니어링 함수는 게임 데이와 워크숍을 통해 카오스 엔지니어링 연습을 소개하는 진행자 역할을 합니다. 실패 시나리오 선택, 가설 정의, 더 광범위한 조직과 공유할 보고서 생성 등 카오스 엔지니어링 프로세스에서 팀을 안내합니다. 중앙 집중식 연습 팀은 훈련 자료를 소유하고 연습 팀이 카오스 엔지니어링을 사용할 수 있도록 역량을 강화하기 위해 노력해야 합니다.
- 공지 사항 - 중앙 집중식 연습 팀은 자문 역할을 맡아 연습 팀이 수행하는 실험을 감독할 수도 있습니다. 이들의 경험과 지식은 실험이 비즈니스에 가치를 제공하고 안전한 방식으로 수행되도록 할 수 있

습니다. 마찬가지로 팀은 실험 실행을 감독하고 실험 결과를 보고하여 카오스 엔지니어링을 처음 접하는 사람들을 안내할 수 있습니다.

- 마케팅 및 가치 추적 - 카오스 엔지니어링의 비즈니스 가치를 전달하는 것은 이러한 프로그램의 성공에 매우 중요합니다. 카오스 엔지니어링 실험에 참여하는 각 팀은 비즈니스 전반의 실험에서 데이터를 수집하고 카오스 엔지니어링에 대한 조직의 투자 가치를 입증해야 합니다. 여기에는 각 실험 중에 회피된 인시던트 수, 실험이 실패한 경우 발생했을 가동 중지 시간, 프로덕션에서 장애 시나리오가 발생한 경우 비즈니스에 미치는 전반적인 영향을 정량화하고 축하하는 것이 포함됩니다. 팀 전체에서 이러한 데이터를 수집 및 중앙 집중화하고 조직 전체에서 데이터를 사용할 수 있도록 함으로써 중앙 집중식 연습 팀은 조직 전체의 카오스 엔지니어링 채택에서 파생된 가치를 추적하고 영향을 미칠 수 있습니다.
- 표준 - 중앙 집중식 연습 팀은 카오스 실험을 수행하는 프로세스, 실험 계획 및 보고를 위한 템플릿, 실험을 수행하는 데 사용되는 도구를 소유하고 유지해야 합니다.

  중앙 팀은 실험 계획 템플릿, 실험 보고서 템플릿, 프로세스 설명서 및 지원 자료를 소유하고 관리해야 합니다. 모범 사례 설명서 및 지원 자료는 실험의 영향을 제한하는 데 사용할 수 있는 가드레일, 프로덕션 환경에서 실험을 수행하는 시기, 시간이 지남에 따라 카오스 엔지니어링 사용을 발전시키는 방법과 같은 주제에 대해 팀을 연습하는 데 지침을 제공합니다. 템플릿 및 출력의 예는 부록을 참조하십시오.

  또한 중앙 집중식 연습 팀은 커뮤니케이션 및 에스컬레이션, 실험 전 또는 실험 중에 조직의 다른 팀과 소통하는 시기와 방법을 포함하여 실험을 수행하는 프로세스를 소유해야 합니다. 가드레일이 필요한 경우에도 프로세스를 간략하게 설명해야 합니다.

  또한 중앙 집중식 연습 팀은 카오스 실험을 수행하기 위한 핵심 도구(예:와 같은 도구)를 선택하고 소유해야 합니다 AWS FIS. 로드 생성 도구와 같은 보조 도구의 선택 및 구현은 실무 팀이 결정해야 합니다. 연습 팀은 전체 프로세스와 도구를 필요에 가장 잘 맞게 조정할 수 있어야 합니다.

실무 팀의 역할

중앙 집중식 팀은 전반적인 카오스 엔지니어링 전략을 주도하는 반면, 실무 팀은 프로세스에 참여하고 실험 개발 및 실행을 담당합니다. 이를 통해 실험이 각 특정 제품 또는 서비스와 관련이 있고 학습이 실행 가능하며 제품의 신뢰성과 복원력을 개선하기 위해 적용될 수 있는지 확인할 수 있습니다. 중앙 집중식 연습 팀은 조직의 카오스 엔지니어링 표준 및 프로세스의 조언자이자 소유자 역할을 합니다. 그러나 중앙 집중식 팀이 병목 현상이 되지 않도록 하려면 개별 연습 팀이 중앙 연습에서 학습하여 자체적으로 카오스 실험을 수행해야 합니다.

연습 커뮤니티 구축

중앙 집중식 팀을 만드는 것 외에도 카오스 엔지니어링에 관심이 있는 비공식 실무자 커뮤니티를 구축하는 것이 좋습니다. 이 커뮤니티는 실무 팀과 광범위한 조직에서 지식, 모범 사례 및 경험을 공유할 수 있는 플랫폼을 제공합니다.

중앙 집중식 카오스 엔지니어링 실무 팀이 실무 커뮤니티를 운영할 수 있지만 조직 내 모든 사람이 커뮤니티의 구성원이 될 수 있습니다. 중앙 집중식 팀은 연습 커뮤니티를 활용하여 업데이트 및 소스 학습을 브로드캐스트하고 중앙 집중식 팀이 관리하는 표준 및 프로세스를 사용하는 연습 팀으로부터 피드백을 수집할 수 있습니다. 커뮤니티는 피드백 루프 역할을 하여 중앙 집중식 팀에 연습 팀 전반의 카오스 엔지니어링 관행의 효과를 알립니다. 그런 다음 중앙 집중식 연습 팀은 제품 팀을 가장 잘 지원하도록 설명서 및 지원 아티팩트를 조정할 수 있습니다.

운영 복원력에 카오스 엔지니어링 통합

카오스 실험은 프로덕션 환경에서 인시던트를 방지하기 위한 비즈니스의 투자입니다. 비즈니스가이 투자에서 가장 큰 수익을 실현할 수 있는 위치를 결정해야 합니다. 조직은 중앙 집중식 카오스 엔지니어링 실무 팀과 협력하여 표준을 업데이트하고 카오스 실험이 필요할 만큼 중요한 제품을 결정할 수 있습니다.

시스템 개발 프로세스

카오스 엔지니어링 및 카오스 실험은 애플리케이션 수명 주기의 일부로 반복적으로 수행해야 합니다. 팀이 정기적으로 재해 복구 테스트를 수행하는 방식과 마찬가지로 카오스 실험과 게임 데이를 연중 지속적으로 주기적으로 수행해야 합니다. 이 접근 방식은 조직이 인시던트를 예상, 관찰 및 대응하는 방법을 개선합니다.

결론

카오스 엔지니어링 분야는 지난 10년 동안 크게 발전했습니다. 이는 다양한 산업에서 채택되었으며 조직이 복원력 있는 서비스를 만들고 고객 만족도를 높이는 데 도움이 되었습니다. (조직이 이러한 관행을 구현한 방법에 대한 예는 카오스 엔지니어링 스토리를 참조하세요.) 카오스 엔지니어링을 통해 조직은 클라우드 공급자 서비스를 포함한 애플리케이션 스택의 모든 수준에서 제어된 장애를 주입하여 미션 크리티컬 애플리케이션에 대한 위험을 줄일 수 있습니다. 제어된 방식으로 전체 애플리케이션 스택에 영향을 미치는 기능을 사용하면 프로덕션 중단의 스트레스 없이 지속적인 복원력, 운영 우수성, 관찰성 및 복구 지향 아키텍처를 개선할 수 있습니다. 또한이 접근 방식은 조직 전체의 복원력 테스트 관행을 개선합니다. 카오스 엔지니어링을 수용하려면 카오스 게임 데이 또는 워크숍을 실행하여 카오스 엔지니어링이 조직에 제공할 수 있는 가치를 보여줍니다.

리소스

AWS FIS 장애 모델 구현:

- AWS Fault Injection Service 를 사용하여 다중 리전 및 다중 AZ 애플리케이션 복원력 시연(AWS 블로그 게시물)
- AWS Fault Injection Simulator, Amazon EKS 포드에서 카오스 엔지니어링 실험 지원(AWS 블로그 게시물)
- Amazon ECS 워크로드에 대한 AWS Fault Injection Simulator의 새로운 기능 발표(AWS 블로그 게시물)
- Amazon EBS 볼륨 지표 및 AWS Fault Injection Simulator를 사용하여 애플리케이션 복원력 개선(AWS 블로그 게시물)
- 다중 계정 AWS 환경에서 AWS Fault Injection Simulator를 활용한 카오스 엔지니어링(AWS 블로그 게시물)
- AWS Fault Injection Simulator를 사용한 Amazon RDS의 카오스 실험(AWS 블로그 게시물)
- 카오스 엔지니어링을 사용하여 복원력이 뛰어난 서버리스 애플리케이션 구축(AWS 블로그 게시물)
- FIS를 사용하여 스팟 인스턴스 중단(AWS 워크숍)
- 전송 파이프라인에서 카오스 엔지니어링 자동화(AWS 커뮤니티 게시물)

기타 리소스:

- 전략적 필요성으로 카오스 엔지니어링에 투자
- 카오스 엔지니어링 스토리
- Amazon CloudWatch 설명서
- Amazon CloudWatch RUM 설명서
- AWS X-Ray 설명서
- AWS FIS 설명서

부록: 샘플 문서

이 섹션에 제공된 샘플 문서는 가상의 반려동물 채택 사이트(PetSite)를 카오스 실험의 대상 애플리케이션으로 사용합니다.

- 실험 계획 문서
- 실험 결과 문서

실험 계획 문서

정상 상태

프로세스 이름	반려동물 입양 사이트
물리적 아키텍처	(아키텍처 다이어그램 링크)
논리적 아키텍처	(논리적 다이어그램 링크)
안정 상태 정의	반려 동물 채택 사이트에 대해 LCP(가장 큰 콘텐츠 페인트)를 사용하여 측정한 평균 페이지 로드 시간은 2.5초 이하이고, 99 백분위수 지연 시간(P99)은 4.0초 이하이며, 기준은 동시 사용자 5,000명입니다.
정상 상태 지표	사용자 기반에서 캡처된 LCP 지표 및 골든 지표(지연 시간, 처리량, 오류율, 포화도).
정상 상태 관찰성	LCP는 사용자의 브라우저에서 캡처되어 Amazon CloudWatch로 전송되고 CloudWatch RUM으로 검사됩니다. 60초 동안 평균 및 P99 LCP 시간은 해당 기간의 모든 요청에 대해 집계됩니다. 최상위 골든 지표는 CloudWatch를 사용하여 캡처됩니다.
안정 상태를 달성하기 위한 프로세스	Grafana K6는 약 5K00명의 동시 사용자의 정상 프로덕션 트래픽 수준을 시뮬레이션하는 로드를 생성하는 데 사용됩니다.

관찰성 요구 사항

팀은 다음을 볼 수 있어야 합니다.

- 정상 상태: 애플리케이션이 정상 상태인지 확인하기 위해 무엇을 관찰하나요?
- 실패 조건: 대시보드에 실패 조건이 어떻게 표시되나요? 예제:
  - 트리거해야 하는 경보
  - 생성해야 하는 로그
- 장애 영향: 영향을 받을 것으로 예상되는 구성 요소를 보려면 무엇을 관찰해야 합니까(영향 범위)?
- 복구: MTTR을 캡처하기 위해 복구를 어떻게 보고 측정하나요?
- 디버그: 실험 실패에 대한 세부 정보 문제 해결.

다음 표에는 관찰성 요구 사항 차트에 대한 제안과 예제가 나와 있습니다. 특정 실험을 기반으로 관찰해야 할 사항을 정의해야 합니다.

관찰해야 할 사항	관찰성 도구 링크	관찰 대상
입력 소스	Grafana K6 대시보드	• 실행 중인 컨테이너 수 • 초당 요청
전체 애플리케이션 상태	• 반려동물 입양 CloudWatch 대시보드 • 반려동물 입양 사용자 경험 대시보드(RUM)	• Amazon EKS 정상 노드 수 • Amazon EKS 노드 CPU 사용률
워크플로 상태	반려동물 입양 CloudWatch 대시보드	LCP 시간, 골든 지표
트레이스	반려동물 입양 X-Ray 대시보드	• 요청 지연 시간 • 요청 수 • 실패 수

관찰해야 할 사항	관찰성 도구 링크	관찰 대상
로그	반려동물 입양 CloudWatch Logs	포드에서 발생하는 모든 오류는 CloudWatch Logs에 발행됩니다.

실험 정의

실험 이름	Amazon EKS PetSite 포드 CPU 스트레스
실험 소스 코드	(실험 소스 리포지토리 링크)
실험 설명	이 실험에서는 PetSite 애플리케이션 포드의 CPU 사용량 증가가 전반적인 고객 경험에 미치는 영향을 살펴봅니다. 실행 중인 각 PetSite 포드에 CPU 스트레스를 주입 하면 고객에게 영향이 있는지 여부와 해당 영향의 범위를 이해할 수 있습니다.
실험 요구 사항 또는 파라미터	애플리케이션 로드: 프로덕션 평균 포드 레이블 선택기: `app=petsite`
실험 기간	10분
환경	알파 테스트 환경
실험 대상 리소스	PetSite 애플리케이션 포드
로드 생성 도구를 통해 도입되는 실험 기준	• 요청의 54%는 LCP가 <2.5초입니다. • 요청의 46%는 LCP가 <4초입니다. • 오류는 관찰되지 않습니다.
백오프 조건	없음

가설

다음과 같은 경우	영향	복구
PetSite 애플리케이션 포드가 정상적인 프로덕션 수준 트래픽에서 10분 동안 60% 이상의 CPU 사용률을 경험하거나 유발한 경우 안정적인 상태는 어떻게 되나요?	P99가 4.0초 이하인 사용자의 P50에서는 LCP 시간이 2.5초 미만으로 유지됩니다. 소비자가 PetSite 랜딩 페이지를 로드할 수 있어야 합니다.	감지: • CPU 스트레스는 CloudWatch에 구성된 경보에 의해 감지됩니다. • LCP 지표는 사용자 경험 저하에 대한 경보도 생성합니다. 자가 복구: • 마이크로서비스 아키텍처의 분산 특성으로 인해 많은 포드 인스턴스가 여러 가용 영역에서 실행되고 있습니다. • EKS 클러스터 컨트롤 플레인은 영향을 받는 포드에서 트래픽을 이동하고 작업자 노드에서 새 포드를 시작합니다. 복구: CPU 사용률이 정상으로 돌아오면 LCP가 자동으로 복구됩니다.

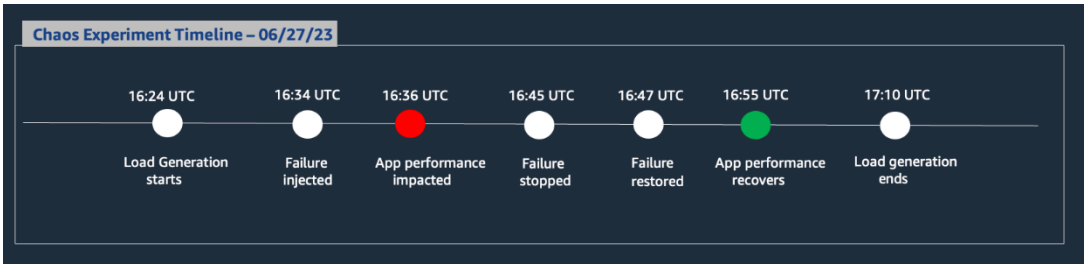
실험 프로세스

다음 예제 step-by-step 프로세스를 특정 실험에 맞게 조정합니다.

1. 모든 Amazon CloudWatch, CloudWatch RUM 및 AWS X-Ray 대시보드에 대한 액세스 및 기능을 검증합니다.
2. 애플리케이션 환경의 상태를 검증합니다.
   a. CloudWatch 대시보드를 사용하여 EKS 클러스터가 정상인지 확인합니다.
   b. 예제 URL에서 테스트 반려동물 채택 사이트 애플리케이션 배포를 방문하세요.
3. 부하를 시작하여 안정 상태를 달성합니다.
   a. 로드 생성기가 실행 중이고 초당 5,000개의 요청을 전송하는지 확인합니다.
   b. 애플리케이션이 안정 상태에 도달할 때까지 5분 정도 기다립니다.
   c. CloudWatch RUM 대시보드를 사용하여 애플리케이션의 안정 상태를 확인합니다.
4. 결함 시작(실험):
   a. AWS FIS 콘솔을 엽니다.
   b. pet-adoption-pod-stress 실험을 실행합니다.
   c. 콘솔에서 실험이 실행 중인지 확인합니다.
5. 장애가 애플리케이션에 미치는 영향을 관찰합니다.
   a. CloudWatch RUM 및 CloudWatch 대시보드에서 스크린샷을 캡처하고 변칙적인 데이터 포인트를 기록해 둡니다.
   b. 실험이 완료된 후 추가 스크린샷을 AWS FIS캡처하여 애플리케이션이 스트레스 없이 안정 상태로 돌아가는지 여부를 기록하고 데이터 포인트에 이상을 기록합니다.
   c. 안정 상태가 재개되지 않으면 애플리케이션을 복구하는 단계를 수행하고 수행한 단계를 기록합니다.
6. 환경이 정상으로 돌아왔는지 확인합니다.
   - 모든 비즈니스, 사용자 경험, 애플리케이션 및 인프라 지표를 검토하여 시스템이 알려진 상태로 돌아왔는지 확인합니다. 도움이 되는 경우 대시보드 스크린샷을 캡처합니다.

실험 타임라인

로드 생성, 결함 주입, 영향 관찰 및 애플리케이션 복구부터 시작하여 로드 생성을 중지할 때 종료되는 end-to-end 실험의 타임라인을 캡처해야 합니다. 다음 예제에 설명되어 있습니다.



실험 결과

실험 실행 ID	실험 결과
PET-ADOPT-EXP-23	(실험 결과에 대한 링크)

확인된 결함

- Kubernetes 클러스터가 PetSite 포드의 CPU 장애를 감지하지 못했으므로 추가 배포를 예약하지 않았습니다.
- 이 실험의 결과로 4XX 또는 5XX 오류율이 증가하지 않았습니다.
- 리소스 제약이 있을 때 LCP에 미치는 영향을 설명하기 위해 포드의 상태 확인을 조정해야 합니다.

실험 결과 문서

구성

실험의 특정 구성을 문서화합니다. 예제:

- 초당 총 85K000명의 사용자를 시뮬레이션하도록 설정된 로드 생성.

사전 조건

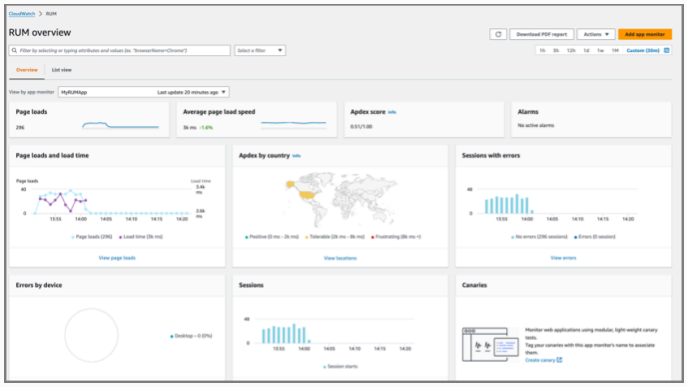
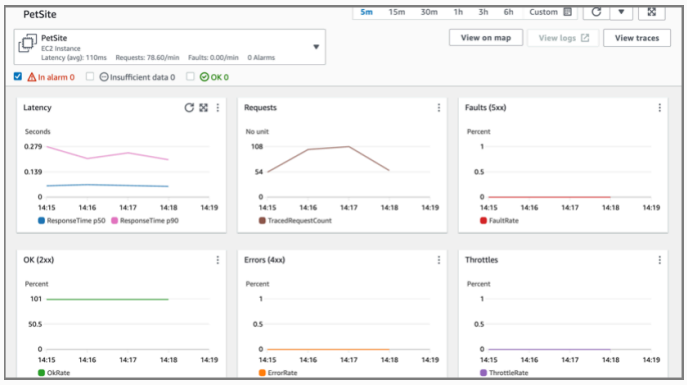
- 알파 테스트 환경에서 반려동물 입양 사이트가 실행되고 있는지 확인했습니다.
- 실험 템플릿이 EKS 클러스터에서 실행 중인 PetSite 애플리케이션 포드에 CPU 스트레스를 적용하도록 구성되었는지 확인했습니다. 애플리케이션 포드는 Kubernetes 레이블 로 식별되었습니다`app=petsite`.
- 로드가 실행 중이고 초당 85개의 요청을 생성하는 것으로 확인되었습니다.

정상 상태

안정 상태를 달성하기 위해 수행한 단계와 이를 확인한 방법을 문서화합니다. 예제:

반려동물 채택 사이트의 테스트 배포의 경우 안정 상태를 시뮬레이션하기 위해 85RPS의 부하가 생성됩니다. CloudWatch RUM 및 CloudWatch 대시보드를 검토하여 실험 실행 전에 모든 비즈니스 및 애플리케이션 지표가 정상 범위 내에 있는지 확인했습니다.

관찰성 데이터:

예상	관찰됨
• 요청의 P99에 대한 LCP는 4초 미만입니다. • 응답 지연 시간은 500ms 미만입니다. • 4XX 또는 5XX 오류는 없습니다.	 

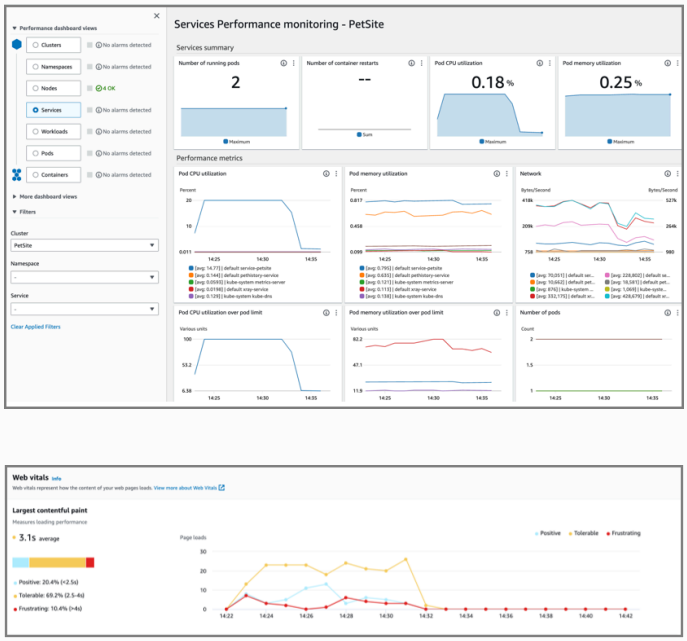
결함 주입

AWS FIS 는 실험 템플릿(링크 제공)을 사용하여 결함을 주입하는 데 사용되었습니다. 실험은 10분 동안 실행되도록 설정되었으며 작업자 노드에 60% 이상의 CPU 스트레스가 발생한 경우 롤백이 구성되었습니다.

결함 관찰

CloudWatch RUM 및 CloudWatch 대시보드를 검토하여 애플리케이션의 안정 상태(LCP 지표를 사용하여 정의)를 추적했습니다. 스크린샷은 다음 표에 캡처되었습니다.

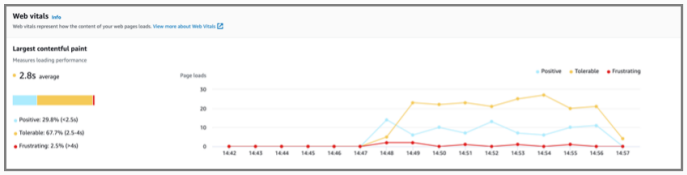
관찰성 데이터:

예상	관찰됨
• P99의 경우 LCP는 4초 미만으로 유지되어야 합니다. • 응답 시간은 500ms 미만이어야 합니다. • 4XX 또는 5XX 오류가 발생하지 않아야 합니다.	

복구

스트레스가 제거된 후(AWS FIS 실험이 완료되고 포드에서 CPU 스트레스가 제거됨) 애플리케이션은 정상 안정 상태를 재개해야 합니다. 수동 개입은 필요하지 않습니다.

관찰성 데이터:

예상	관찰됨(스크린샷)
LCP P99는 4초 미만이어야 하며 평균은 2.5초 미만이어야 합니다.	

문서 기록

아래 표에 이 가이드의 주요 변경 사항이 설명되어 있습니다. 향후 업데이트에 대한 알림을 받으려면 RSS 피드를 구독하십시오.

변경 사항	설명	날짜
최초 게시	—	2025년 4월 4일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.