Unable to locate subtitle

AWS Well-Architected Framework

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

Table of Contents

AWS Well-Architected Framework

게시 날짜: 2023년 4월 10일(문서 개정)

AWS Well-Architected Framework를 활용하면 AWS에서 시스템을 구축하면서 내리게 되는 결정의 장단점을 이해할 수 있습니다. 이 프레임워크를 사용하면 클라우드에서 안정적이고 안전하며 효율적이고 경제적이며 지속 가능한 시스템을 설계하고 운영하기 위한 설계 모범 사례를 알아볼 수 있습니다.

소개

AWS Well-Architected Framework를 활용하면 AWS에서 시스템을 구축하면서 내리게 되는 결정의 장단점을 이해할 수 있습니다. 이 프레임워크를 사용하면 AWS 클라우드에서 안전하고 안정적이며 효율적이고 경제적이면서 지속 가능한 워크로드를 설계하고 운영하기 위한 설계 모범 사례를 알아볼 수 있습니다. 또한 모범 사례를 기준으로 아키텍처를 일관적으로 측정하고 개선할 영역을 식별할 수 있습니다. 아키텍처 검토 프로세스는 아키텍처 결정에 대한 구조적인 대화이며 감사 메커니즘이 아닙니다. 시스템을 제대로 설계하면 비즈니스 성공 가능성을 높일 수 있습니다.

AWS 솔루션스 아키텍트는 광범위한 업종 및 사용 사례에 걸쳐 오랫동안 솔루션을 설계해 왔습니다. AWS(은)는 수천 고객의 AWS 아키텍처를 설계하고 검토해 왔습니다. 그리고 이러한 경험을 바탕으로 클라우드 시스템 설계의 모범 사례와 핵심 전략을 밝혀냈습니다.

AWS Well-Architected Framework 설명서에는 특정 아키텍처가 클라우드 모범 사례에 적합한지를 파악하는 데 도움이 되는 기본적인 질문이 다수 수록되어 있습니다. 이 프레임워크를 이용하면 클라우드 기반의 현대적 시스템에 대해 고객이 기대하는 품질 기준에 따라 일관적인 방식으로 시스템을 평가하고 그러한 품질을 달성하기 위해 필요한 수정 조치를 확인할 수 있습니다. AWS(이)가 진화를 거듭하고 Amazon이 고객과의 협력을 통해 점점 더 많은 지식을 쌓아가면서 앞으로도 Well-Architected 개념을 더욱 정교하게 가다듬고자 합니다.

이 프레임워크는 최고 기술 책임자(CTO), 아키텍트, 개발자와 같은 기술 업무 담당자와 운영 팀원을 위해 작성되었습니다. 클라우드 워크로드를 설계하고 운영할 때 사용할 AWS 모범 사례와 전략에 대해 설명하고 구현 세부 정보 및 아키텍처 패턴에 대한 링크를 제공합니다. 자세한 내용은 AWS Well-Architected 홈페이지를 참조하세요.

AWS에서는 워크로드 무료 검토 서비스도 제공됩니다. AWS Well-Architected Tool(AWS WA Tool)은 AWS Well-Architected Framework를 사용하여 아키텍처를 검토 및 측정하는 일관된 프로세스를 제공하는 클라우드 서비스입니다. AWS WA Tool은 워크로드를 더 안정적이고 안전하며 효율적이고 경제적으로 만드는 권장 사항을 제공합니다.

모범 사례를 적용할 수 있도록 모범 사례를 구현한 실무 경험이 포함된 설명서 및 코드 리포지토리를 제공하는 AWS Well-Architected Labs를 만들었습니다. 또한 AWS Well-Architected 파트너 프로그램의 멤버인 AWS 파트너 네트워크(APN) 파트너를 선택하여 팀을 구성했습니다. 이러한 AWS 파트너는 풍부한 AWS 지식을 보유하고 있으며, 이를 통해 워크로드를 검토하고 개선하는 데 도움을 줄 수 있습니다.

정의

AWS의 전문가들은 매일 고객과 함께 클라우드의 모범 사례를 활용하여 시스템을 설계합니다. 설계의 진화에 발맞춰 고객의 아키텍처에 더할 것과 뺄 것을 결정할 수 있도록 지원합니다. 그리고 고객이 이러한 시스템을 실제 환경에 배포하는 과정에서 해당 시스템의 성능 수준과 그러한 결정의 결과를 배우게 됩니다.

AWS는 이렇게 얻은 교훈을 토대로 고객 및 파트너가 아키텍처를 평가할 수 있는 일관적인 모범 사례 및 아키텍처가 AWS 모범 사례에 얼마나 잘 맞는지 평가할 수 있는 여러 가지 질문을 제공하는 AWS Well-Architected Framework를 개발했습니다.

AWS Well-Architected Framework는 운영 우수성, 보안, 신뢰성, 성능 효율성, 비용 최적화 및 지속 가능성이라는 6가지 원칙을 기반으로 합니다.

테이블 1. AWS Well-Architected Framework의 원칙

이름	설명
운영 우수성	The ability to support development and run workloads effectively, gain insight into their operations, and to continuously improve supporting processes and procedures to deliver business value.
보안	The security pillar describes how to take advantage of cloud technologies to protect data, systems, and assets in a way that can improve your security posture.
신뢰성	The reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it’s expected to.

이름	설명
	This includes the ability to operate and test the workload through its total lifecycle. This paper provides in-depth, best practice guidance for implementing reliable workloads on AWS.
성능 효율성	The ability to use computing resources efficient ly to meet system requirements, and to maintain that efficiency as demand changes and technologies evolve.
비용 최적화	The ability to run systems to deliver business value at the lowest price point.
지속 가능성	The ability to continually improve sustainability impacts by reducing energy consumption and increasing efficiency across all components of a workload by maximizing the benefits from the provisioned resources and minimizing the total resources required.

AWS Well-Architected Framework에서는 다음 용어를 사용합니다.

- 구성 요소는 요구 사항에 맞춰 함께 제공되는 코드, 구성 및 AWS 리소스입니다. 구성 요소는 대개 기술 소유권의 단위이며 각기 분리되어 있습니다.
- 워크로드는 비즈니스 가치를 제공하는 일련의 구성 요소를 식별하는 데 사용되는 용어입니다. 워크로드는 일반적으로 비즈니스 및 기술 책임자가 전달하는 세부 정보의 수준입니다.
- 아키텍처는 워크로드에서 구성 요소가 연동되는 방식입니다. 아키텍처 다이어그램은 주로 구성 요소들의 통신 및 상호 작용 방식을 중점적으로 다룹니다.
- 마일스톤은 설계, 구현, 테스트, 가동 및 프로덕션으로 이어지는 제품 수명 주기 전반에 걸쳐 아키텍처가 개선되는 과정에서 발생하는 주요 변화 시점을 표시합니다.
- 조직 내에서 기술 포트폴리오는 기업을 운영하는 데 필요한 워크로드 모음입니다.
- 작업 수준은 구현에 필요한 작업 시간, 활동 및 복잡성을 분류하는 것입니다. 각 조직이 조직의 유효 작업 수준을 적절하게 분류하려면 팀의 규모와 전문성, 추가 컨텍스트의 워크로드 복잡성을 고려해야 합니다.

- 높음: 이 작업을 완료하는 데는 몇 주 또는 몇 달이 걸릴 수 있으며, 여러 스토리, 릴리스 및 작업으로 나눌 수 있습니다.
- 중간: 이 작업을 완료하는 데는 며칠 또는 몇 주가 걸릴 수 있으며, 여러 릴리스 및 작업으로 나눌 수 있습니다.
- 낮음: 이 작업을 완료하는 데는 몇 시간 또는 며칠이 걸릴 수 있으며, 여러 작업으로 나눌 수 있습니다.

워크로드를 설계할 때는 업무 상황에 따라 이러한 원칙들을 절충해야 합니다. 이러한 비즈니스 의사 결정에 따라 엔지니어링 우선순위가 달라질 수 있습니다. 개발 환경의 신뢰성이 다소 낮아지더라도 지속 가능성에 미치는 영향을 개선하고 비용을 절감하도록 최적화하거나, 미션 크리티컬 솔루션의 경우 비용 및 지속 가능성에 미치는 영향이 증가하는 상황을 감수하고 신뢰성을 기준으로 최적화할 수 있습니다. 전자 상거래 솔루션의 경우 성능이 매출과 고객 구매 성향에 영향을 미칠 수 있습니다. 보안 및 운영 우수성은 일반적으로 다른 원칙과 절충 관계에 있지 않습니다.

아키텍처

온프레미스 환경에서 고객은 종종 다른 제품 또는 기능 팀에 대한 중첩되는 역할을 통해 모범 사례를 따르는지를 확인하는 기술 아키텍처에 대한 중앙 집중형 팀을 보유하고 있습니다. 일반적으로 기술 아키텍처 팀에는 테크니컬 아키텍트(인프라), 솔루션스 아키텍트(소프트웨어), 데이터 아키텍트, 네트워킹 아키텍트 및 보안 아키텍트와 같은 일련의 역할이 포함됩니다. 이러한 팀에서는 대개 TOGAF 또는 Zachman Framework를 엔터프라이즈 아키텍처 기능의 일부로 사용합니다.

AWS(은)는 해당 기능을 중앙 팀을 두어 집중화하지 않고, 각각의 개별 팀에 나눠 주는 것을 선호합니다. 의사 결정 권한을 분산하게 될 경우, 개별 팀이 내부 기준을 충족하도록 관리해야 하는 등 여러 위험 요소가 생기게 됩니다. AWS는 2가지 방법으로 이러한 위험을 완화합니다. 첫째로, 각 팀이 해당 기능을 갖추는 방법을 중점적으로 설명하는 사례(작업 수행 방식, 프로세스, 표준 및 용인된 규범)를 도입하고, 팀이 충족해야 하는 표준에 대한 기준을 높일 수 있는 전문가를 배치합니다. 둘째, 자동화된 검사를 수행하여 표준이 충족되는지 확인하는 메커니즘을 구현합니다.

> “좋은 의도만으로는 부족합니다. 무언가를 해내려면 좋은 메커니즘이 필요한 법이죠.“ - Jeff Bezos.

이 말은 사람의 노력을 규칙이나 프로세스 준수 여부를 확인하는 메커니즘(종종 자동화됨)으로 대체하는 것을 의미합니다. 이러한 분산된 접근 방식은 Amazon 리더십 원칙에 근간을 두고 있으며, 고객으로

부터 출발하는 거꾸로 일하는 문화를 모든 역할에 걸쳐 조성합니다. 거꾸로 일하기는 혁신 프로세스의 기본 요소입니다. AWS는 고객 및 고객이 원하는 것에서부터 출발하며, 이를 기반으로 작업을 정의하고 방향을 잡습니다. 고객 중심의 팀은 고객의 요구 사항에 대응하여 제품을 개발합니다.

아키텍처 측면에서, 이는 모든 팀이 아키텍처를 생성하고 모범 사례를 따르는 기능을 갖추고 있음을 의미합니다. AWS는 새로운 팀이 이러한 기능을 확보하거나 기존 팀이 기준을 높이도록 돕기 위해, 설계를 검토하고 AWS 모범 사례가 무엇인지 이해하는 데 도움을 주는 수석 엔지니어 가상 커뮤니티에 액세스할 수 있게 해줍니다. 수석 엔지니어 커뮤니티는 모범 사례를 가시화하고 쉽게 액세스할 수 있도록 최선을 다합니다. 예를 들어 모범 사례를 실제 사례에 적용하는 데 중점을 둔 간략한 회의를 통해 이를 수행합니다. 이러한 회의 내용은 기록되어 새 팀원을 위한 온보딩 자료의 일부로 사용할 수 있습니다.

AWS의 모범 사례는 수천 개의 시스템을 인터넷 규모로 운영한 경험에서 비롯된 것입니다. AWS는 데이터를 사용하여 모범 사례를 정의하는 것을 선호하지만, 수석 엔지니어와 같은 주제 전문가를 활용하여 설정하기도 합니다. 수석 엔지니어가 새로운 모범 사례를 확인하게 되면 커뮤니티로 활동하여 팀이 이를 따를 수 있도록 안내합니다. 시간이 지나면서 이러한 모범 사례는 내부 검토 절차 및 규정 준수를 시행하는 메커니즘으로 공식화됩니다. Well-Architected 프레임워크는 내부 검토 프로세스를 고객 중심으로 구현한 결과이며, 주요 현장에서 활동하는 솔루션스 아키텍처 및 내부 엔지니어링 팀에서 엔지니어링 사고를 체계화한 것입니다. Well-Architected 프레임워크는 이러한 결과를 활용할 수 있는 확장 가능한 메커니즘입니다.

AWS는 아키텍처를 분산 담당하는 주요 엔지니어링 커뮤니티의 접근 방식을 따르면, 고객 요구 사항 중심 Well-Architected 엔터프라이즈 아키텍처가 실현될 수 있다고 믿습니다. 모든 워크로드 전반에 걸쳐 Well-Architected 검토를 수행하는 기술 리더(CTO 또는 개발 관리자)를 두면, 기술 포트폴리오의 위험을 더 효과적으로 이해할 수 있습니다. 이 접근 방식을 사용하면 수석 엔지니어가 특정 영역에 대한 사고방식을 여러 팀과 공유할 수 있는 메커니즘, 교육 또는 간략한 회의를 통해 조직에서 해결 가능한 전반적인 주제를 식별할 수 있습니다.

일반 설계 원칙

Well-Architected Framework는 다음과 같은 여러 가지 일반 설계 원칙을 확립하여 우수한 클라우드 설계를 촉진합니다.

- 필요 용량에 대한 추측 불필요: 워크로드를 배포할 때 용량을 잘못 결정하면 결과적으로 고가의 유휴 리소스를 방치하게 되거나 제한된 용량으로 인한 성능 문제를 처리해야 하는 이중고를 겪습니다. 하지만 클라우드 컴퓨팅에서는 이러한 문제가 사라집니다. 필요한 만큼 용량을 많이 또는 적게 사용하다가 자동으로 확장하거나 축소할 수 있기 때문입니다.
- 프로덕션 규모의 테스트 시스템: 클라우드에서는 온디맨드 방식으로 프로덕션 규모의 테스트 환경을 만들고, 테스트를 완료한 다음 해당 리소스를 폐기할 수 있습니다. 테스트 환경을 실행하는 동안

에만 비용을 지불하면 되기 때문에 온프레미스 테스트 비용의 몇분의 일에 불과한 가격으로 실제 환경을 시뮬레이션할 수 있습니다.

- 자동화를 통해 아키텍처 실험 수행 과정 간소화: 자동화를 사용하면 적은 비용으로 워크로드를 생성하고 복제할 수 있으며, 수작업으로 인한 비용을 줄일 수 있습니다. 자동화 과정의 변경 사항을 추적하고, 그 효과를 감사하고, 필요하면 이전의 파라미터로 되돌릴 수 있습니다.
- 아키텍처의 지속적인 혁신: 기존 환경에서는 정해진 방식의 일회성 이벤트로 아키텍처를 결정하는 경우가 많고 시스템의 수명 주기 중 메이저 버전 업그레이드는 몇 차례 이루어지지 않습니다. 기업과 경영 상황은 계속 변화하는데, 시스템이 변화무쌍한 비즈니스 요구 사항을 만족하도록 혁신할 때 초기의 오판이 두고두고 발목을 잡을 수 있습니다. 그러나 클라우드에는 온디맨드 방식의 자동화 및 테스트 기능이 있어 설계 변경에 따른 위험이 줄어듭니다. 따라서 시간이 지날수록 시스템은 진화하고, 기업은 혁신을 표준 사례로 활용할 수 있게 됩니다.
- 데이터를 사용하여 아키텍처 구동: 클라우드에서는 아키텍처 선택에 따른 워크로드 동작에 미치는 영향에 대한 데이터를 수집할 수 있습니다. 그러므로 사실에 근거하여 어떻게 워크로드를 개선할지 결정할 수 있습니다. 클라우드 인프라는 코드이므로 이 데이터를 장기적으로 아키텍처 선택 및 개선을 위한 정보로 활용할 수 있습니다.
- 게임 데이를 통한 개선: 프로덕션 환경에서 이벤트를 시뮬레이션하기 위한 게임 데이를 정기적으로 실시하여 아키텍처 및 프로세스가 어떻게 작동하는지 테스트할 수 있습니다. 그러면 어느 분야에서 개선이 필요한지 파악하고, 조직이 이벤트에 대처하는 경험을 쌓도록 도울 수 있습니다.

프레임워크의 원칙

소프트웨어 시스템을 제작하는 것은 건물을 짓는 것과 매우 비슷합니다. 토대가 단단하지 않으면 구조적 문제가 발생하여 건물의 기능이 약해지는 것은 물론 건물 자체가 무너질 수 있습니다. 기술 솔루션을 설계할 때 운영 우수성, 보안, 안정성, 성능 효율성, 비용 최적화 및 지속 가능성이라는 여섯 가지 기반 원칙을 간과하면 기대 및 요구에 충실한 시스템을 구축하기가 어려울 수 있습니다. 이러한 기반 원칙을 아키텍처에 통합하면 안정적이고 효율적인 시스템을 구축하는 데 도움이 됩니다. 또한 이를 바탕으로 기능적 요구 사항 등 설계의 다른 측면에 집중할 수 있게 됩니다.

원칙

- 운영 우수성
- 보안
- 안정성
- 성능 효율성
- 비용 최적화
- 지속 가능성

운영 우수성

운영 우수성 원칙은 효과적인 개발 및 워크로드 실행을 지원하고, 작업에 대한 인사이트를 얻고, 지원 프로세스 및 절차를 지속적으로 개선하여 비즈니스 가치를 제공할 수 있는 능력을 포함합니다.

운영 우수성 원칙에서는 설계 원리 개요, 모범 사례 및 질문 사항을 제공합니다. 구현 방법에 대한 선제적인 가이드는 운영 우수성 원칙 백서.

주제

- 설계 원칙
- 정의
- 모범 사례
- 리소스

설계 원칙

클라우드의 운영 우수성에는 5가지 설계 원칙이 있습니다.

- 코드를 통한 운영: 애플리케이션 코드를 위해 사용하였던 엔지니어링 원칙을 클라우드에서 인프라를 포함한 환경에 적용할 수 있습니다. 클라우드에서는 전체 워크로드(애플리케이션, 인프라)를 코드로 정의하고 코드와 함께 업데이트할 수 있고 운영 절차를 코드로 구현하고 이벤트에 대응하여 이를 트리거하면 실행을 자동화할 수 있습니다. 코드로 운영을 수행하면 인적 오류가 제한되고 이벤트에 대한 일관된 대응이 가능합니다.
- 자주 발생하고 되돌릴 수 있는 사소한 변경 내용 적용: 요소를 정기적으로 업데이트할 수 있도록 워크로드를 설계하고, 실패할 경우 되돌릴 수 있는 작은 증분으로 변경 내용을 적용합니다(가능하면 고객에게 영향을 주지 않도록).
- 수시로 운영 절차 수정: 운영 절차를 사용할 때 개선할 여지가 있는지 확인합니다. 워크로드가 개선되면 절차도 적절하게 개선합니다. 정기적인 게임 데이를 설정하여 모든 절차가 효과가 있으며 팀이 이러한 절차에 친숙한지 여부를 검토하고 검증합니다.
- 실패 예측: "사전 분석(pre-mortem)" 연습을 수행하여 잠재적인 실패 소스를 식별하고 이를 해소하거나 완화할 수 있도록 합니다. 실패 시나리오를 테스트하고 그에 따른 영향을 이해했는지 여부를 검증합니다. 응답 절차를 테스트하여 효과가 있는지, 팀이 이 실행 단계에 친숙한지 확인합니다. 정기적인 게임 데이를 준비하여 시뮬레이션된 이벤트에 대한 팀의 대응 및 워크로드를 테스트합니다.
- 모든 운영상 실패로부터 학습: 모든 운영상 이벤트 및 실패로부터 파악한 내용을 통해 개선합니다. 팀 전반 및 조직 전체에서 파악한 내용을 공유합니다.

정의

클라우드의 운영 우수성에는 4가지 모범 사례 영역이 있습니다.

- 조직
- 준비
- 운영
- 개선

조직의 경영진이 비즈니스 목표를 정합니다. 조직은 요구 사항과 우선순위를 파악하고, 이를 통해 비즈니스 성과를 실현할 수 있도록 업무를 구성하고 수행해야 합니다. 또한 워크로드에서 이를 지원하는 데 필요한 정보를 생성해야 합니다. 워크로드를 통합, 배포 및 제공하는 서비스를 구현하면 반복적인 프로세스를 자동화하여 프로덕션 환경에 유익한 변경 사항을 지속적으로 더 많이 적용할 수 있습니다.

워크로드 운영에 내재된 위험이 있을 수 있습니다. 이러한 위험을 파악하고 정보에 근거하여 프로덕션 환경에 적용할지 여부를 결정해야 합니다. 그리고 팀에서 워크로드를 지원할 수 있어야 합니다. 원하는 비즈니스 성과에서 도출된 비즈니스 및 운영 지표를 통해 워크로드 상태, 운영 활동, 인시던트에 대한

대응 능력을 파악할 수 있습니다. 우선순위는 비즈니스 요구 사항과 비즈니스 환경 변화에 따라 달라집니다. 이를 피드백 루프로 활용하여 조직과 워크로드 운영을 지속적으로 개선합니다.

모범 사례

주제

조직

적절한 업무 수행의 기준이 되는 우선순위를 설정하려면 팀이 전체 워크로드, 워크로드 내 각 팀원의 역할 그리고 공동의 업무 목표를 파악해야 합니다. 우선순위를 잘 정하면 운영 개선 작업의 이점을 극대화할 수 있습니다. 실무 팀, 개발 팀, 운영 팀 등의 주요 이해관계자와 함께 내부 및 외부 고객 요구 사항을 평가하여 주력할 영역을 결정합니다. 고객 요구 사항을 평가하면 비즈니스 성과를 달성하기 위해 어떤 지원이 필요한지 철저하게 파악할 수 있습니다. 조직의 거버넌스와 규정 준수 요구 사항 및 산업 표준과 같은 외부 요인에 따라 특정 작업을 반드시 또는 집중적으로 수행해야 하는 의무 사항이나 지침을 파악해야 합니다. 내부 거버넌스 및 외부 규정 준수 요구 사항의 변경 내용을 식별할 수 있는 메커니즘이 있는지 확인합니다. 요구 사항이 식별되지 않는 것으로 결론을 내릴 때에는 신중하게 판단하여 내린 결론인지 재차 확인해야 합니다. 주기적으로 우선순위를 검토하여 요구 사항의 변화에 따라 우선순위를 업데이트합니다.

비즈니스에 대한 위협 요소(예: 비즈니스상의 위험 및 법적 책임, 정보 보안 위협)를 평가하고 위험 목록에서 이 정보를 관리합니다. 위험의 영향과 상충하는 이해 관계나 대안 사이의 장단점을 평가합니다. 예를 들어, 비용 최적화보다 새로운 기능의 시장 출시를 앞당기는 데 더 역점을 둘 수 있습니다. 아니면 리팩터링 없이 시스템 마이그레이션 작업을 간소화하기 위해 비관계형 데이터용 솔루션으로 관계형 데이터베이스를 선택할 수도 있습니다. 주력할 영역을 결정할 때 정보를 토대로 적절한 결정을 내릴 수 있도록 이점과 위험을 관리합니다. 일부 위험이나 선택은 한동안 감수할 수 있거나, 관련 위험을 완화할 수도 있겠지만, 위험을 감수할 수 없는 경우에는 위험을 해결하기 위한 조치를 취해야 합니다.

팀은 비즈니스 성과를 달성하기 위해 맡은 역할을 파악해야 합니다. 그리고 다른 팀의 성공을 위해 자신의 팀이 해야 할 역할과 해당 팀이 해야 할 역할을 파악하고, 목표를 공유해야 합니다. 맡은 책임, 소유권, 의사 결정 방식 및 의사 결정권자를 파악하면 역량을 집중하고 팀의 이점을 극대화할 수 있습니다. 팀의 요구 사항은 팀에서 지원하는 고객, 소속된 조직, 팀 구성 및 워크로드의 특성에 따라 결정됩니다. 당연히 단일 운영 모델로는 모든 팀과 조직 내에서 그들이 맡은 워크로드를 지원할 수 없습니다.

애플리케이션, 워크로드, 플랫폼 및 인프라 구성 요소마다 소유자가 명시되어 있고, 각 프로세스와 절차의 정의 및 실행을 담당하는 소유자가 각각 명시되어 있는지 확인합니다.

각 구성 요소, 프로세스 및 절차의 비즈니스 가치, 이러한 리소스가 배치되거나 활동이 수행되는 이유, 그러한 소유권이 존재하는 이유를 파악하면 팀원의 작업을 알 수 있습니다. 팀원이 적절하게 행동하고 책임과 소유권을 식별하는 메커니즘이 마련되도록 팀원의 책임을 명확하게 정의합니다. 혁신에 제약이 없도록 추가, 변경 및 예외를 요청하는 메커니즘을 마련합니다. 팀 간의 협력을 통해 서로를 지원하는 방법과 비즈니스 성과를 설명하는 계약을 정의합니다.

팀원이 효과적으로 조치를 취하고 비즈니스 성과를 지원할 수 있도록 팀원에 대한 지원을 제공합니다. 관련 최고 경영진이 기대치를 설정하고 성공 여부를 측정해야 합니다. 최고 경영진은 조직이 발전하고 모범 사례를 도입하도록 하는 동인이자 후원자이자 지지자입니다. 팀원에게 성과가 위험한 상태일 때 영향을 최소화하기 위한 조치를 취할 수 있는 권한을 주고, 위험이 있다고 판단될 때 문제 해결과 사고 방지를 위해 의사 결정권자 및 이해관계자에게 에스컬레이션하도록 합니다. 팀원이 시기 적절하고 적절한 조치를 취할 수 있도록 알려진 위험과 계획된 이벤트에 대한 시기 적절하고 명확하며 실행 가능한 커뮤니케이션을 제공합니다.

실험을 권장하여 학습을 가속화하고 팀원의 관심과 참여를 유지합니다. 팀은 새로운 기술을 도입하고 수요와 책임의 변화를 지원하기 위해 기술을 발전시켜야 합니다. 학습을 위한 전용 구조 시간을 제공하여 이를 지원하고 장려합니다. 팀원이 성공과 비즈니스 성과 지원을 위한 확장에 필요한 리소스 즉, 도구와 팀원을 모두 확보하고 있는지 확인합니다. 조직 간의 다양성을 활용하여 여러 가지 고유한 관점을 모색합니다. 이러한 관점을 통해 혁신을 증진하고, 기존의 추정 사항에 의문을 제기하며, 확증 편향의 위험을 줄일 수 있습니다. 팀 내에서 포용성, 다양성 및 접근성을 높여 유익한 관점을 확보합니다.

조직에 적용되는 외부 규제 또는 규정 준수 요구 사항이 있다면 팀원이 우선순위에 대한 영향을 확인할 수 있도록 AWS Cloud Compliance 에서 제공하는 리소스를 사용하여 관련 정보를 제공해야 합니다. Well-Architected 프레임워크에서는 학습, 평가 및 개선을 강조합니다. 아키텍처를 평가하고 시간에 따라 확장 가능한 설계를 구현하는 일관된 접근 방식을 제공합니다. AWS에서 선보이는 AWS Well-Architected Tool은 개발 전의 접근 방식, 프로덕션 환경에 적용하기 전의 워크로드 상태, 프로덕션 환경에서의 워크로드 상태를 검토합니다. 워크로드를 최신 AWS 아키텍처 모범 사례와 비교하고, 워크로드의 전반적인 상태를 모니터링하며, 잠재적 위험에 대한 인사이트를 얻을 수 있습니다. AWS Trusted Advisor는 우선순위 결정에 도움이 될 수 있는 최적화 방안을 알려 주는 핵심 검사 세트에 액세스할 수 있는 도구입니다. Business 및 Enterprise Support 고객에게는 우선순위를 더욱 자세히 결정하는 데 사용할 수 있는 추가 검사 기능이 제공됩니다. 이러한 기능을 사용하면 보안, 안정성, 성능 및 비용 최적화 영역을 중점적으로 확인할 수 있습니다.

AWS를 활용하면 선택한 방식이 워크로드에 미치는 영향을 효과적으로 파악하도록 팀에 AWS 및 해당 서비스 관련 정보를 제공할 수 있습니다. AWS Support(AWS 지식 센터, AWS 토론 포럼, AWS Support 센터) 및 AWS 설명서에 나와 있는 리소스를 사용해서 팀을 교육해야 합니다. AWS Support

센터를 통해 AWS Support 팀에 문의하여 AWS 관련 질문의 답을 찾을 수도 있습니다. AWS는 AWS 운영을 통해 학습한 모범 사례와 패턴을 Amazon Builders' Library에서 공유합니다. AWS 블로그 및 공식 AWS 팟캐스트에서도 기타 여러 가지 유용한 정보를 확인할 수 있습니다. AWS Training and Certification에서는 AWS 기초에 관한 자습형 디지털 과정을 통해 무료 교육을 제공합니다. 강사 주도형 교육에 등록하여 팀이 AWS 기술을 연마하도록 추가로 지원할 수도 있습니다.

운영 모델 관리를 위해 AWS Organizations와 같이 여러 계정에 걸쳐 환경을 중앙 집중식으로 관리할 수 있는 도구나 서비스를 사용해야 합니다. AWS Control Tower와 같은 서비스는 계정 설정을 위한 블루프린트(운영 모델 지원)를 정의하고, AWS Organizations를 통해 지속적으로 거버넌스를 적용하며, 새로운 계정의 프로비저닝을 자동화할 수 있도록 함으로써 이 관리 기능을 확장합니다. 관리형 서비스 공급자(예: AWS Managed Services, AWS Managed Services 파트너 또는 AWS 파트너 네트워크의 관리형 서비스 공급자)는 클라우드 환경을 구현하는 전문성을 제공하며, 보안 및 규정 준수 요구 사항과 비즈니스 목표를 지원합니다. Managed Services를 운영 모델에 추가하면 시간과 리소스를 절약할 수 있으며, 새로운 기술과 기능을 개발하는 대신 내부 팀을 간소화하며, 팀이 비즈니스를 차별화하는 전략적 결과에 집중할 수 있습니다.

다음은 운영 우수성 고려 사항에 중점을 둔 질문입니다. 운영 우수성 관련 질문 및 모범 사례 목록은 부록을 참조하십시오.

OPS 1: 귀사의 운영 우선순위를 결정하는 요인은 무엇입니까?

모든 직원이 효율적인 업무 수행을 위한 역할과 리소스 우선 순위 설정을 위한 공동의 목표를 파악하고 있어야 합니다. 그러면 운영 개선 작업의 이점을 극대화할 수 있습니다.

OPS 2: 비즈니스 성과를 지원하기 위해 조직을 어떻게 구성합니까?

팀은 비즈니스 성과를 달성하기 위해 맡은 역할을 파악해야 합니다. 그리고 다른 팀의 성공을 위해 자신의 팀이 해야 할 역할과 해당 팀이 해야 할 역할을 파악하고, 목표를 공유해야 합니다. 맡은 책임, 소유권, 의사 결정 방식 및 의사 결정권자를 파악하면 역량을 집중하고 팀의 이점을 극대화할 수 있습니다.

OPS 3: 조직 문화는 비즈니스 성과를 어떻게 지원합니까?

팀원이 효과적으로 조치를 취하고 비즈니스 성과를 지원할 수 있도록 팀원에 대한 지원을 제공합니다.

특정 시점에 우선순위 중 일부를 중점적으로 처리해야 할 수도 있습니다. 필요한 기능을 개발하고 위험을 관리하려면 워크로드 우선순위를 장기적으로 적절하게 절충해야 합니다. 우선순위를 정기적으로 검토하고 요구 사항이 바뀌면 우선순위를 업데이트합니다. 책임과 소유권을 정의하지 않았거나 알지 못하는 경우 필요한 활동을 적시에 처리하지 못하고 해당 요구 사항을 해결하기 위한 작업이 중복되고 잠재적으로 상충될 위험이 있습니다. 조직 문화는 팀원의 업무 만족도와 팀원 이직률에 직접적인 영향을 미칩니다. 팀원의 참여와 역량을 통해 비즈니스의 성공을 뒷받침할 수 있습니다. 혁신과 아이디어를 실현하려면 실험이 필요합니다. 원치 않는 결과가 나와도 성공하지 못하는 경로를 알게 되었으므로 실험에는 성공한 것으로 인정합니다.

준비

운영 우수성 달성을 준비하려면 워크로드 및 예상되는 워크로드 동작을 파악해야 합니다. 그러면 워크로드가 상태 관련 인사이트를 제공하도록 설계할 수 있으며, 워크로드를 지원하는 절차를 작성할 수 있습니다.

문제를 관찰하고 조사할 수 있도록 모든 구성 요소에서 지표, 로그, 이벤트, 추적 등 내부 상태를 파악하는 데 필요한 정보를 제공하도록 워크로드를 설계합니다. 반복을 통해, 워크로드 상태를 모니터링하고, 성과 실현에 실패할 위험이 있는 경우 이를 식별하며, 효과적으로 대응하는 데 필요한 원격 측정을 개발합니다. 워크로드를 계측할 때 상태를 파악할 수 있는 광범위한 정보 세트를 캡처합니다(예: 상태 변경 사항, 사용자 활동, 권한 있는 액세스, 사용률 카운터). 이때 필터를 사용하여 시간 경과에 따라 가장 유용한 정보를 선택할 수 있습니다.

프로덕션 환경으로 변경 사항을 전달하는 흐름을 개선할 수 있는 방식을 도입합니다. 이 방식은 리팩터링, 품질과 관련된 빠른 피드백 및 버그 수정을 지원해야 합니다. 이러한 방식을 도입하면 유용한 변경 사항을 프로덕션 환경으로 빠르게 전달할 수 있고, 문제 배포 가능성을 제한할 수 있으며, 배포 활동을 통해 발생하거나 환경에서 발생된 문제를 빠르게 파악하고 해결할 수 있습니다.

품질과 관련한 피드백을 빠르게 제공하며, 적절한 성과를 달성하는 데 도움이 되지 않는 변경을 수행한 경우 신속하게 복구할 수 있는 방식을 도입합니다. 이러한 사례를 사용하면 변경 사항 배포로 인해 발생하는 문제의 영향을 완화할 수 있습니다. 필요한 경우 더 빠르게 대응하고 변경 사항을 테스트 및 확인할 수 있도록 부적절한 변경을 수행한 경위의 계획을 수립합니다. 계획된 활동에 영향을 미치는 변경 위험을 제어할 수 있도록 환경의 계획된 활동을 알고 있어야 합니다. 되돌릴 수 있는 소규모 변경을 자주 수행하도록 하여 변경 범위를 제한합니다. 그러면 문제를 더 쉽게 해결할 수 있으며 변경 사항 롤백 옵션을 사용해 문제 해결 시간을 단축할 수 있습니다. 또한 중요한 변경 사항의 이점을 더 자주 누릴 수 있다는 의미이기도 합니다.

워크로드, 프로세스, 절차 및 직원의 운영 준비 상태를 평가하여 워크로드와 관련된 운영 위험을 파악합니다. 수동 또는 자동화된 체크리스트를 비롯한 일관된 프로세스를 사용해 워크로드 또는 변경에 응

답하는 준비 여부를 확인해야 합니다. 이렇게 하면 문제 해결 계획을 세워야 하는 영역도 파악할 수 있습니다. 일상 활동을 문서화한 런북과 문제 해결 프로세스를 안내하는 플레이북을 준비합니다. 이점과 위험을 파악하여 프로덕션에 변경 사항 적용에 대해 정보에 입각한 결정을 내립니다.

AWS에서 전체 워크로드(애플리케이션, 인프라, 정책, 거버넌스, 운영)를 코드로 확인할 수 있습니다. 즉, 애플리케이션 코드에 사용하는 것과 동일한 엔지니어링 분야를 스택의 모든 요소에 적용하고 이를 팀 또는 조직 간에 공유하여 개발 작업의 이점을 확대할 수 있습니다. 클라우드에서 운영을 코드로 사용하고 워크로드, 운영 절차 및 사례 실패 개발을 위해 안전하게 실험하는 기능을 사용합니다. AWS CloudFormation을 사용하면 운영 제어 수준이 점점 증가하는 일관된 템플릿 형식의 샌드박스 개발, 테스트 및 생산 환경을 갖출 수 있습니다.

다음은 운영 우수성 고려 사항에 중점을 둔 질문입니다.

OPS 4: 운영 상태를 파악할 수 있도록 어떻게 워크로드를 설계하십니까?

모든 구성 요소에서 지표, 로그, 추적 등의 내부 상태를 파악하는 데 필요한 정보를 제공하도록 워크로드를 설계합니다. 이렇게 하면 효율적으로 적절한 대응을 할 수 있습니다.

OPS 5: 귀사는 어떻게 결함을 줄이고 수정 작업을 쉽게 수행하고 프로덕션으로 이어지는 흐름을 개선하십니까?

프로덕션 환경으로 변경 사항을 전달하는 흐름을 개선할 수 있는 방식을 도입합니다. 이 방식은 리팩터링, 품질과 관련된 빠른 피드백 및 버그 수정을 지원해야 합니다. 이러한 방식을 도입하면 유용한 변경 사항을 프로덕션 환경으로 빠르게 전달할 수 있고, 문제 배포 가능성을 제한할 수 있으며, 배포 활동을 통해 발생하는 문제를 빠르게 파악하고 해결할 수 있습니다.

OPS 6: 배포 위험을 어떻게 최소화하고 있습니까?

품질과 관련한 피드백을 빠르게 제공하며, 적절한 성과를 달성하는 데 도움이 되지 않는 변경을 수행한 경우 신속하게 복구할 수 있는 방식을 도입합니다. 이러한 사례를 사용하면 변경 사항 배포로 인해 발생하는 문제의 영향을 완화할 수 있습니다.

OPS 7: 귀사가 워크로드를 지원할 준비가 되어있는지 어떻게 알 수 있습니까?

워크로드, 프로세스, 절차 및 직원의 운영 준비 상태를 평가하여 워크로드와 관련된 운영 위험을 파악합니다.

운영 활동을 코드로 구현하여 운영 인력의 생산성을 최대화하고, 오류율을 최소화하고, 자동화된 응답을 사용할 수 있습니다. 해당하는 경우에는 “사전 분석(pre-mortem)” 기능을 사용하여 장애를 예측하고 절차를 생성합니다. 리소스 태그 및 AWS Resource Groups을 사용하여 메타데이터를 적용하고 일관된 태그 지정 전략을 시행하면 리소스를 식별할 수 있습니다. 리소스에 조직, 비용 회계, 액세스 제어에 대한 리소스에 태그를 지정하여 자동화된 운영 활동을 실행할 대상을 설정합니다. 클라우드의 탄력성을 활용하는 배포 실습을 도입하여 개발 활동을 용이하게 하고 시스템을 사전 배포할 수 있도록 함으로써 보다 빠른 구현을 달성합니다. 워크로드를 평가하는 데 사용하는 체크리스트를 변경할 때는 해당 변경으로 인해 더 이상 규정을 준수하지 않는 라이브 시스템에 대해 수행할 작업을 계획합니다.

운영

워크로드 운영의 성공은 비즈니스 및 고객 성과 달성에 따라 측정됩니다. 예상 결과를 정의하고, 성공을 측정하는 방법을 결정하고 이러한 계산에 사용될 지표를 식별하여 워크로드와 운영이 성공적인지 여부를 결정합니다. 운영 상태에는 워크로드 상태, 워크로드 지원 시 수행되는 운영 활동의 상태와 성공이 모두 포함됩니다(예: 배포 및 인시던트 응답). 개선, 조사 및 개입에 대한 지표 기준선을 설정하고 지표를 수집 및 분석한 후 운영 성공에 대한 이해 및 시간에 따라 어떻게 변하는지를 확인합니다. 수집된 지표를 사용하여 고객과 비즈니스 요구 사항을 충족하는지 여부를 확인하고 개선 영역을 식별합니다.

운영 우수성을 달성하려면 효과적이고 효율적인 운영 이벤트 관리가 필요합니다. 이는 계획된 운영 이벤트 및 계획되지 않은 운영 이벤트 모두에 적용됩니다. 사전에 파악된 이벤트에 대해 런북을 작성하여 사용하고, 문제 조사 및 해결에 도움이 되는 해결책을 지원하는 데는 플레이북을 사용합니다. 비즈니스 및 고객 영향을 기반으로 이벤트 응답의 우선순위를 지정합니다. 이벤트 응답에 대해 알람이 발생하는지, 연결된 실행 프로세스가 있는지 여부를 식별된 담당자와 함께 확인합니다. 이벤트를 해결하는 데 필요한 인력을 미리 정하고 에스컬레이션 트리거를 포함하여 필요할 경우 긴급성과 영향을 기반으로 추가 인력의 참여를 유도합니다. 권한이 있는 개인을 식별하고 참여시켜 이전에 해결되지 않은 이벤트 대응에 대해 대응 과정이 비즈니스에 영향을 미쳤는지 확인합니다.

타겟(예: 고객, 비즈니스, 개발자, 운영)에 맞는 알림 및 대시보드를 통해 워크로드 운영 상태를 전달하여 적절한 조치를 취하고 기대 사항을 관리하고 정상 운영이 다시 시작될 때 알림을 받을 수 있도록 합니다.

AWS에서는 AWS의 기본 지표 및 워크로드에서 수집된 지표가 나와 있는 대시보드 보기를 생성할 수 있습니다. CloudWatch 또는 서드파티 애플리케이션을 활용하여 운영 활동의 비즈니스, 워크로드 및 운영 수준 보기를 표시하고 집계할 수 있습니다. AWS에서는 AWS X-Ray, CloudWatch, CloudTrail, VPC 흐름 로그 등 로깅 기능을 통해 워크로드 인사이트를 제공하여 워크로드 문제를 파악하고 근본 원인 분석 및 해결을 지원합니다.

다음은 운영 우수성 고려 사항에 중점을 둔 질문입니다.

OPS 8: 워크로드의 상태를 어떻게 파악하십니까?

워크로드 지표를 정의, 캡처 및 분석하면 워크로드 이벤트에 대한 가시성을 확보하여 적절한 조치를 취할 수 있습니다.

OPS 9: 운영 상태를 어떻게 파악하십니까?

운영 지표를 정의, 캡처 및 분석하면 운영 이벤트에 대한 가시성을 확보하여 적절한 조치를 취할 수 있습니다.

OPS 10: 워크로드 및 운영 이벤트를 어떻게 관리하십니까?

이벤트로 인해 워크로드가 중단될 가능성을 최소화할 수 있도록 이벤트 대응을 위한 절차를 준비/확인합니다.

수집하는 모든 지표는 비즈니스 요구 사항과 지원되는 성과에 부합해야 합니다. 잘 알려진 이벤트에 대한 스크립팅된 응답을 개발하고 이벤트 인식에 대한 응답으로 성능을 자동화합니다.

개선

운영 우수성을 유지하려면 학습하고 공유하고 지속적으로 개선해야 합니다. 연속적이고 증분적 개선을 이뤄내는 데에 주력하여 작업 주기를 조절합니다. 고객에게 영향을 미치는 모든 이벤트에 대한 사후 분석을 수행합니다. 재발 제한 또는 방지를 위한 기여 요인과 예방 조치를 파악합니다. 영향을 받는 커뮤니티와 함께 기여 요소를 적절히 알립니다. 워크로드 및 운영 절차 모두를 포함하여 개선의 여지(예: 기능 요청, 문제 해결, 규정 준수 요구 사항)를 정기적으로 평가하고 우선순위를 조정합니다.

절차 내에 피드백 루프를 포함시켜 개선할 영역을 빠르게 식별하고 운영 실행을 통해 학습한 내용을 파악합니다.

팀 전반에 걸쳐 파악한 내용을 공유하여 이러한 내용의 이점도 함께 공유합니다. 파악한 내용 내의 추세를 분석하고 운영 지표에 대해 팀 교차 후행 분석을 수행하여 개선할 여지 및 방법을 식별합니다. 개선하려는 변경 사항을 적용하고 결과를 평가하여 성공 여부를 확정합니다.

AWS에서 Amazon S3로 로그 데이터를 내보내거나 Amazon S3로 로그를 직접 전송하여 장기 보관할 수 있습니다. AWS Glue를 사용하면 Amazon S3에서 분석 목적으로 로그 데이터를 검색하고 준비하며, AWS Glue Data Catalog에 관련 메타데이터를 저장할 수 있습니다. Amazon Athena에서 AWS Glue와의 기본 통합을 통해 로그 데이터를 분석하고 표준 SQL을 사용하여 쿼리할 수 있습니다. Amazon QuickSight와 같은 비즈니스 인텔리전스 도구를 사용하면 데이터를 시각화하고 탐색하며 분석할 수 있습니다. 개선을 이끌 수 있는 추세와 관심 이벤트를 찾습니다.

다음은 운영 우수성 고려 사항에 중점을 둔 질문입니다.

OPS 11: 귀사는 어떻게 운영을 지속적으로 개선하고 있습니까?

시간과 리소스를 할애하여 점진적 개선을 지속적으로 수행하면 운영 효율성을 높일 수 있습니다.

성공적인 운영 개선은 잦은 소규모 개선, 안전한 환경 및 실험, 개발, 테스트 개선에 대한 시간 제공, 그리고 실패로부터 학습 독려하는 환경을 통해 이루어집니다. 샌드박스, 개발, 테스트 및 생산 환경에 대한 운영 지원을 통해 운영 제어 수준을 점점 높아지도록 하고, 개발을 촉진하며, 생산 단계에 배포된 변경에서 성공적인 결과가 예측 가능하도록 합니다.

리소스

운영 우수성 모범 사례에 대한 자세한 내용은 다음 리소스를 참조하십시오.

설명서

- DevOps 및 AWS

백서

- 운영 우수성 원칙

동영상

- Amazon의 DevOps

보안

보안 원칙에는 클라우드 기술을 활용하여 보안을 강화하고 데이터, 시스템 및 자산을 보호하는 능력이 포함됩니다.

보안 원칙은 설계 원칙 개요, 모범 사례 및 질문 사항을 제공합니다. 구현 방법에 대한 선제적인 가이드는 보안 원칙 백서.

주제

- 설계 원칙
- 정의
- 모범 사례
- 리소스

설계 원칙

클라우드에는 7가지 보안 설계 원칙이 있습니다.

- 강력한 자격 증명 기반 구현: 최소 권한 원칙을 구현하고 AWS 리소스와의 각 상호작용에 필요한 권한을 적절히 부여하여 업무를 분리합니다. 자격 증명 관리를 중앙 집중화하고 장기적인 정적 자격 증명에 대한 의존도를 해소하는 것을 목표로 합니다.
- 추적 기능 활성화: 실시간으로 환경에 대한 작업 및 변경 사항을 모니터링하고 알림을 전송하며 감사합니다. 로그 및 지표 수집을 시스템과 통합하여 자동으로 조사하고 조치를 취합니다.
- 모든 계층에 보안 적용: 여러 보안 제어 기능을 통해 심층 방어 방식을 적용합니다. 모든 계층(예: 네트워크 엣지, VPC, 로드 밸런싱, 모든 인스턴스 및 컴퓨팅 서비스, 운영 체제, 애플리케이션, 코드)에 적용됩니다.
- 보안 모범 사례의 자동 적용: 자동화된 소프트웨어 기반의 보안 메커니즘은 더욱 빠르게 안전한 확장 능력을 향상 시켜주며 비용 효율적입니다. 버전 제어가 가능한 템플릿에서 코드로 정의되고 관리되는 제어 기능의 구현을 비롯한 보안 아키텍처를 생성합니다.
- 전송 및 보관 중인 데이터 보호: 데이터를 민감도 수준으로 분류하고 적절한 경우 암호화, 토큰화 및 액세스 제어와 같은 메커니즘을 사용합니다.
- 사람들이 데이터에 쉽게 액세스할 수 없도록 유지: 데이터의 직접 액세스 또는 수동 처리의 필요성을 줄이거나 없애기 위한 메커니즘 및 도구를 사용합니다. 이를 통해 민감한 데이터를 처리할 때 잘못된 취급이나 수정 및 수작업으로 인한 오류의 위험을 줄일 수 있습니다.

- 보안 이벤트에 대비: 조직의 요구 사항에 부합하는 인시던트 관리 및 조사 정책과 프로세스를 마련하여 인시던트에 대비합니다. 인시던트 대응 시뮬레이션을 실행하고 자동화된 도구를 사용하여 감지, 조사 및 복구 속도를 높입니다.

정의

클라우드의 보안에는 6가지 모범 사례 영역이 있습니다.

- 보안
- 자격 증명 및 액세스 관리
- 탐지
- 인프라 보호
- 데이터 보호
- 사고 대응

워크로드를 설계하기 전에 보안에 영향을 미치는 업무의 수행 방식을 마련해야 합니다. 작업을 수행할 수 있는 대상 및 작업 내용을 제어할 수 있어야 합니다. 또한 보안 사고를 식별하고 시스템과 서비스를 보호하며 데이터 보호를 통해 데이터의 기밀성과 무결성을 유지할 수 있기를 원합니다. 보안 사고에 대응하기 위한 잘 정의된 프로세스를 마련하고 숙련해야 합니다. 이는 금전적 손해 방지 또는 규제 의무 준수 등 목표 달성을 뒷받침하는 중요한 도구이자 기법입니다.

AWS 공동 책임 모델은 클라우드를 채택한 조직의 보안 및 규정 준수 목표를 달성하는 데 도움을 줍니다. 클라우드 서비스를 뒷받침하는 인프라를 AWS가 물리적으로 보호하기 때문에 AWS 고객은 서비스를 이용하여 목표를 달성하는 데 집중할 수 있습니다. 나아가 AWS 클라우드에서는 보안 데이터에 더 폭넓게 액세스할 수 있으며, 보안 이벤트에 대응하는 자동화된 접근 방식을 제공합니다.

모범 사례

주제

보안

워크로드를 안전하게 운영하려면 모든 보안 영역에 포괄적 모범 사례를 적용해야 합니다. 조직 및 워크로드 수준에서 운영 우수성에 정의된 요구 사항과 프로세스를 가져와 모든 영역에 적용합니다.

AWS 및 업계 권장 사항 및 위협 인텔리전스를 최신 상태로 유지하면 위협 모델 및 제어 목표를 발전시키는 데 도움이 됩니다. 보안 프로세스, 테스트 및 검증을 자동화함으로써 보안 작업을 확장할 수 있습니다.

다음은 보안 고려 사항에 중점을 둔 질문입니다. 보안 관련 질문 및 모범 사례 목록은 부록을 참조하십시오.

SEC 1: 워크로드를 안전하게 운영하려면 어떻게 해야 합니까?

워크로드를 안전하게 운영하려면 모든 보안 영역에 포괄적 모범 사례를 적용해야 합니다. 조직 및 워크로드 수준에서 운영 우수성에 정의된 요구 사항과 프로세스를 가져와 모든 영역에 적용합니다. AWS의 권장 사항, 업계 리소스 및 위협 인텔리전스를 최신 상태로 유지하면 위협 모델 및 제어 목표를 발전시키는 데 도움이 됩니다. 보안 프로세스, 테스트 및 검증을 자동화함으로써 보안 작업을 확장할 수 있습니다.

AWS에서는 다양한 워크로드를 기능 및 규정 준수 또는 데이터 민감도 요구 사항에 따라 계정별로 분리하는 것이 좋습니다.

자격 증명 및 액세스 관리

자격 증명 및 액세스 관리는 정보 보안 프로그램의 핵심 요소로, 허가되고 인증된 사용자 및 구성 요소에 한해 허용되는 방식으로만 리소스에 액세스할 수 있도록 하는 것을 말합니다. 예를 들어 보안 주체(계정에서 작업을 수행할 수 있는 계정, 사용자, 역할 및 서비스)를 정의하고, 이러한 보안 주체에 맞게 정의된 정책을 구축하고, 강력한 자격 증명 관리를 구현합니다. 이러한 권한 관리 요소가 인증 및 권한 부여의 핵심 개념을 이룹니다.

AWS에서는 기본적으로 AWS 서비스 및 리소스에 대한 사용자 및 프로그램 액세스를 고객이 직접 제어할 수 있도록 하는 AWS Identity and Access Management(IAM) 서비스로 권한 관리를 지원합니다. 사용자, 그룹, 역할 또는 리소스에 대한 권한을 세부 정책으로 지정할 수 있습니다. 또한 복잡성, 재사용, 멀티 팩터 인증(MFA) 등 강력한 암호를 요구할 수 있는 기능도 있습니다. 기존의 디렉터리 서비스와 연동되도록 할 수도 있습니다. 시스템이 AWS에 액세스해야 하는 워크로드의 경우 IAM이 역할, 인스턴스 프로파일, 아이덴티티 페더레이션, 임시 보안 인증 정보를 통해 보안 액세스를 보장합니다.

다음은 보안 고려 사항에 중점을 둔 질문입니다.

SEC 2: 사람과 시스템에 대한 자격 증명은 어떻게 관리합니까?

안전한 AWS 워크로드 운영에 접근할 때 관리해야 하는 두 가지 유형의 자격 증명이 있습니다. 액세스 권한을 관리하고 부여하는 데 필요한 자격 증명의 유형을 이해하면 적절한 자격 증명이 적절한 조건에서 적절한 리소스에 액세스할 수 있도록 보장할 수 있습니다.

인적 자격 증명: 관리자, 개발자, 운영자 및 최종 사용자가 AWS 환경과 애플리케이션에 액세스하려면 자격 증명이 필요합니다. 이들은 조직의 구성원이거나 협업하는 외부 사용자로, 웹 브라우저, 클라이언트 애플리케이션 또는 대화형 명령줄 도구를 통해 AWS 리소스와 상호작용합니다.

시스템 자격 증명: 서비스 애플리케이션, 운영 도구 및 워크로드에서 AWS 서비스에 요청하려면 (예: 데이터 읽기) 자격 증명이 필요합니다. 이러한 자격 증명에는 Amazon EC2 인스턴스 또는 AWS Lambda 함수와 같이 AWS 환경에서 실행되는 시스템이 포함됩니다. 액세스가 필요한 외부 당사자의 시스템 자격 증명을 관리할 수도 있습니다. 또한 AWS 환경에 액세스해야 하는 시스템이 AWS 외부에 있을 수도 있습니다.

SEC 3: 사람과 시스템에 대한 권한은 어떻게 관리합니까?

AWS 및 워크로드에 액세스해야 하는 사람 및 시스템 자격 증명에 대한 액세스를 제어하는 권한을 관리합니다. 권한은 누가 어떤 조건에서 무엇에 액세스할 수 있는지를 제어합니다.

자격 증명은 어떠한 사용자 또는 시스템과도 공유할 수 없습니다. 사용자 액세스 권한은 암호 요구 사항 및 MFA 적용을 포함하는 모범 사례와 함께 최소한의 권한 접근 방식을 사용하여 부여해야 합니다. AWS 서비스에 대한 API 호출을 포함한 프로그래밍 방식의 액세스는 AWS Security Token Service에서 발행한 것과 같은 일시적이거나 권한이 제한된 보안 인증 정보를 사용하여 수행해야 합니다.

AWS는 Identity and Access Management를 사용하여 도울 수 있는 리소스를 제공합니다. 모범 사례를 알아보려면 다음에 대한 실습을 살펴보십시오. 자격 증명 및 인증 관리, 인적 액세스 제어및 프로그래밍 방식 액세스 제어참조.

탐지

탐지 제어를 사용하여 잠재적 보안 위협 또는 인시던트를 식별할 수 있습니다. 이러한 제어는 일반적인 거버넌스 프레임워크의 핵심 부분으로, 품질 프로세스, 법률 또는 규정 준수 의무, 위협 식별 및 대응 과

정을 지원하는 데 사용됩니다. 탐지 제어의 종류는 여러 가지입니다. 예를 들어, 자산 및 해당 세부 속성의 인벤토리를 만들어 두면 보다 효과적인 의사 결정(및 수명 주기 전반의 제어)이 이루어지고, 이를 운영의 기준으로 삼을 수 있습니다. 또한 내부 감사를 통해 정보 시스템과 관련된 제어 기능을 검사하여 실제 사례가 정책 및 요건에 맞는지, 정의된 조건에 따라 올바른 자동 알림이 설정되어 있는지 확인할 수 있습니다. 이러한 제어 기능은 조직 내에서 변칙적 활동 범위를 식별하고 파악하는 데 도움이 되는 중요한 대응 요소입니다.

AWS에서는 로그, 이벤트 및 모니터링(감사, 자동 분석, 경보 지원)을 처리하여 탐지 제어를 구현할 수 있습니다. CloudTrail 로그, AWS API 호출 및 CloudWatch는 경보와 함께 지표 모니터링을 제공하며, AWS Config은 구성 내역을 제공합니다. Amazon GuardDuty는 악성 또는 인증되지 않은 동작을 지속적으로 모니터링하여 AWS 계정 및 워크로드를 보호하도록 지원하는 관리형 위협 탐지 서비스입니다. 또한 서비스 수준 로그도 사용 가능한데, 예를 들어 Amazon Simple Storage Service(Amazon S3)를 사용하여 액세스 요청을 기록할 수 있습니다.

다음은 보안 고려 사항에 중점을 둔 질문입니다.

SEC 4: 보안 관련 이벤트를 어떻게 감지하나요?

이벤트는 로그와 지표에서 캡처하고 분석하는 방식으로 파악할 수 있습니다. 보안 이벤트 및 잠재적 위협에 대한 조치를 취하면 워크로드를 보호할 수 있습니다.

Well-Architected 워크로드에서 로그 관리가 중요한 이유는 보안 또는 포렌식부터 규제 또는 법적 요구 사항에 이르기까지 다양합니다. 잠재적 보안 인시던트를 식별하려면 로그를 분석하고 이에 대응해야 합니다. AWS에서는 데이터 보존 기간을 지정하거나 데이터를 보존, 아카이빙 또는 최종적으로 삭제할 위치를 정의할 수 있어 로그 관리를 보다 쉽게 구현하는 기능을 제공합니다. 이렇게 하면 더 단순하고 경제적인 방식으로, 예측 가능하고 안정적으로 데이터를 처리할 수 있습니다.

인프라 보호

모범 사례와 업계 규정 또는 규제 의무를 준수하기 위해서는 인프라 보호가 필요하며, 여기에는 심층 방어 및 MFA(멀티 팩터 인증) 등의 제어 방법이 포함됩니다. 지속적으로 클라우드 또는 온프레미스에서 작업을 성공적으로 수행하려면 반드시 이러한 방법을 사용해야 합니다.

AWS에서는 AWS 기본 지원 기술을 사용하거나 AWS Marketplace에서 제공되는 파트너 제품 및 서비스를 사용하여 상태 저장 및 상태 비저장 방식의 패킷 검사를 구현할 수 있습니다. Amazon Virtual Private Cloud(Amazon VPC)를 사용하여 안전하고 확장 가능한 프라이빗 환경을 만들고, 여기에서 게이트웨이, 라우팅 테이블, 퍼블릭 및 프라이빗 서브넷 같은 토폴로지를 정의해야 합니다.

다음은 보안 고려 사항에 중점을 둔 질문입니다.

SEC 5: 네트워크 리소스는 어떻게 보호합니까?

인터넷이든 프라이빗 네트워크이든 상관없이 어떤 형태든 네트워크 연결이 있는 워크로드에는 외부 및 내부 네트워크 기반 위협으로부터 보호하기 위한 다중 방어 계층이 필요합니다.

SEC 6: 컴퓨팅 리소스를 어떻게 보호합니까?

워크로드의 컴퓨팅 리소스는 다계층 방어를 통해 내/외부 위협으로부터 보호해야 합니다. 컴퓨팅 리소스에는 EC2 인스턴스, 컨테이너, AWS Lambda 함수, 데이터베이스 서비스, IoT 디바이스 등이 포함됩니다.

어떤 환경이든 여러 단계의 방어 계층을 두는 것이 좋습니다. 인프라 보호의 경우 클라우드 및 온프레미스 모델을 망라하여 효과를 발휘하는 다양한 인프라 보호 개념과 방법이 있습니다. 경계 보호를 적용하고, 수신 및 송신 지점을 모니터링하고, 종합적인 로깅과 모니터링, 알림을 이용하는 것은 모두 효과적인 정보 보안 계획의 핵심 요소입니다.

AWS 고객은 Amazon Elastic Compute Cloud(Amazon EC2), Amazon Elastic Container Service(Amazon ECS) 컨테이너 또는 AWS Elastic Beanstalk 인스턴스의 구성을 맞춤 조정하거나 강화할 수 있고, 변경 불가능한 Amazon Machine Image(AMI)를 통해 이러한 구성을 유지할 수 있습니다. 이렇게 하면 Auto Scaling에 의한 트리거 또는 수동 방식을 통해 이 AMI로 실행되는 모든 새 가상 서버(인스턴스)가 이 강화된 구성을 얻게 됩니다.

데이터 보호

시스템을 설계하려면 먼저 보안과 관련된 기본적인 관례부터 마련해야 합니다. 예를 들어 데이터 분류는 민감도에 따라 조직의 데이터를 구분하는 하나의 방법이고 암호화는 무단 액세스 사용자가 데이터를 해석하지 못하게 만들어 데이터를 보호하는 방법입니다. 이는 금전적 손해 방지 또는 규제 의무 준수 등 목표 달성을 뒷받침하는 중요한 도구이자 기법입니다.

AWS에서는 다음과 같은 방식으로 데이터 보호를 실현합니다.

- AWS 고객은 데이터에 대한 완전한 통제력을 유지합니다.
- AWS는 정기적인 키 교체 등 키 관리 및 데이터 암호화를 더 간편하게 처리하도록 지원합니다. AWS를 통해 작업을 편리하게 자동하거나 사용자가 직접 유지 관리할 수도 있습니다.

- 파일 액세스, 변경 사항 등 중요한 콘텐츠가 수록된 상세 로그를 확인할 수 있습니다.
- AWS는 탁월한 복원력을 목표로 스토리지 시스템을 설계했습니다. 예를 들어 Amazon S3 Standard, S3 Standard–IA, S3 One Zone-IA 및 Amazon Glacier는 지정된 기간 동안 객체에 대해 99.999999999%의 내구성을 제공할 수 있도록 설계되었습니다. 이 내구성 수준은 연평균 0.000000001%의 객체 예측 손실에 해당합니다.
- 광범위한 데이터 수명 주기 관리 프로세스에 포함될 수 있는 버전 관리는 우발적인 덮어쓰기나 삭제 및 그와 유사한 손해를 방지할 수 있습니다.
- AWS는 절대로 리전 간에 데이터를 이동하지 않습니다. 특정 리전에 저장된 콘텐츠는 사용자가 명시적으로 기능을 활성화하거나 그 기능을 제공하는 서비스를 이용하지 않는 한 해당 리전을 벗어나지 않습니다.

다음은 보안 고려 사항에 중점을 둔 질문입니다.

SEC 7: 데이터는 어떻게 분류합니까?

분류는 적절한 보호 및 보존 제어 수준을 결정하는 데 도움이 되도록 중요도와 민감도를 기준으로 데이터를 분류하는 방법을 제공합니다.

SEC 8: 저장된 데이터는 어떻게 보호합니까?

무단 액세스 또는 취급 부주의의 위험을 줄이기 위해 여러 제어 기능을 구현하여 저장된 데이터를 보호합니다.

SEC 9: 전송 중인 데이터는 어떻게 보호합니까?

무단 액세스 또는 손실의 위험을 줄이기 위해 여러 제어 기능을 구현하여 전송 중인 데이터를 보호합니다.

AWS는 저장 데이터 및 전송 데이터를 암호화할 수 있는 여러 수단을 제공합니다. 데이터를 암호화하기 쉽도록 AWS 서비스에 각종 기능을 내장했습니다. 예를 들어, Amazon S3에 대해 SSE(서버 측 암호화)를 구현하여 데이터를 암호화된 형태로 저장하기 쉽게 만들었습니다. 또한 흔히 SSL termination 이라고 부르는 전체 HTTPS 암호화 및 복호화 프로세스를 Elastic Load Balancing(ELB)을 통해 처리하도록 설정할 수도 있습니다.

사고 대응

고도의 예방 및 탐지 제어를 사용하더라도 조직은 잠재적 보안 인시던트에 대응하고 그 영향을 완화하기 위한 프로세스를 마련해야 합니다. 워크로드의 아키텍처가 인시던트 발생 시 보안 팀이 효과적으로 시스템을 격리 또는 억제하고 운영을 알려진 정상 상태로 복구하는 능력에 지대한 영향을 미칩니다. 보안 인시던트보다 앞서 도구 및 액세스를 마련하고 실전 연습을 통해 인시던트 대응을 정기적으로 연습한다면 아키텍처가 적기에 조사 및 복구를 수용할 수 있게 할 수 있습니다.

AWS에서는 다음과 같은 방식으로 효과적인 인시던트 대응을 지원합니다.

- 파일 액세스, 변경 사항 등 중요한 콘텐츠가 수록된 상세 로그를 확인할 수 있습니다.
- 이벤트는 자동으로 처리될 수 있으며, AWS API를 사용하여 대응을 자동화하는 도구를 트리거합니다.
- AWS CloudFormation을 사용하여 도구 및 "안전한 공간"을 사전 프로비저닝할 수 있습니다. 이를 통해 안전하고 격리된 환경에서 과학 수사를 진행할 수 있습니다.

다음은 보안 고려 사항에 중점을 둔 질문입니다.

SEC 10: 인시던트를 어떻게 예상하고 대응하며 어떻게 사후 복구합니까?
조직의 업무 중단을 최소화할 수 있도록 보안 인시던트를 제때 효고적으로 조사 및 대응하고 사후 복구하려면 철저한 준비가 필요합니다.

보안 팀에게 신속하게 액세스를 부여할 수 있는 절차를 마련하고 인스턴스 격리와 포렌식을 위해 데이터 및 상태 캡처를 자동화합니다.

리소스

보안 관련 AWS 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하십시오.

설명서

- AWS 클라우드 보안
- AWS 규정 준수
- AWS 보안 블로그

백서

- 보안 원칙
- AWS 보안 개요
- AWS 위험 및 규정 준수

동영상

- AWS Security State of the Union
- 공동 책임 개요

안정성

안정성 원칙에서는 워크로드의 기능이 필요한 때에 기능을 정확하고 일관되게 수행하는 역량에 대해 다룹니다. 여기에는 전체 수명 주기에 걸쳐 워크로드를 운영 및 테스트할 수 있는 기능이 포함됩니다. 이 백서는 AWS에서 안정적인 워크로드를 구현하기 위한 세부적인 모범 사례 지침을 제공합니다.

안정성 원칙에서는 설계 원칙 개요, 모범 사례 및 질문 사항을 제공합니다. 구현 방법에 대한 선제적인 가이드는 안정성 원칙 백서.

주제

- 설계 원칙
- 정의
- 모범 사례
- 리소스

설계 원칙

클라우드에는 5가지 신뢰성 설계 원칙이 있습니다.

- 장애 자동 복구: 워크로드의 KPI(핵심 성능 지표)를 모니터링하면 임계값이 위반될 때 자동화를 트리거할 수 있습니다. 이러한 KPI는 서비스 운영의 기술적 측면이 아닌 비즈니스 가치를 측정한 값이어야 합니다. KPI를 모니터링하면 장애 추적 및 자동 알림을 지원하고, 자동화된 복구 프로세스에 따라 장애 지점을 우회하거나 복구할 수 있습니다. 보다 정교한 자동화를 구현할 경우 장애가 발생하기 전에 예측하여 해결하는 것도 가능합니다.

- 복구 절차 테스트: 온프레미스 환경에서 테스트는 워크로드가 특정 시나리오에서 작동하는 것을 증명하기 위해 시행됩니다. 일반적으로 복구 전략을 검증하기 위해 테스트하지는 않습니다. 클라우드에서는 워크로드의 장애 과정을 테스트하고 복구 절차를 검증할 수 있습니다. 자동화를 사용하여 다양한 장애를 시뮬레이션하거나 이전에 장애로 이어졌던 시나리오를 재현할 수 있습니다. 이 접근 방식은 장애 경로를 노출한 후 실제 장애 시나리오가 발생하기 전에 테스트하고 수정하여 위험을 줄일 수 있습니다.
- 수평적 확장으로 워크로드 전체 가용성 증대: 단일의 큰 리소스를 다수의 작은 리소스로 대체하여 단일 장애가 전체 워크로드에 미치는 영향을 축소합니다. 요청을 더 작은 리소스 여러 개로 분산시키면 단일 장애 지점을 회피할 수 있습니다.
- 용량 추정 불필요: 워크로드에 대한 수요가 해당 워크로드의 용량을 넘어서는 리소스 포화 상태는 온프레미스 워크로드에서 흔히 발생하는 장애의 원인입니다(서비스 거부 공격의 대상). 클라우드에서는 수요 및 워크로드 사용량을 모니터링하고 리소스 추가 또는 제거를 자동화함으로써 프로비저닝 과다 또는 부족 현상 없이 최적의 수준으로 수요를 충족할 수 있습니다. 클라우드에도 제한은 있지만 할당량을 어느 정도 제어하고 관리하는 것이 가능합니다(Service Quotas 및 제약 조건 관리 참조).
- 자동화 변경 사항 관리: 인프라 변경은 자동화를 통해 수행되어야 합니다. 자동화 변경을 통하여 인프라를 관리해야 하며 이후에 이러한 변경을 추적하고 검토할 수 있습니다.

정의

클라우드의 안정성에는 4가지 모범 사례 영역이 있습니다.

- 기반
- 워크로드 아키텍처
- 변경 관리
- 장애 관리

안정성을 달성하려면 기반에서 시작해야 합니다. 이 기반은 서비스 할당량과 네트워크 토폴로지로 워크로드를 수용하는 환경을 의미합니다. 분산 시스템의 워크로드 아키텍처는 장애를 예방하고 완화하도록 설계되어야 합니다. 워크로드는 수요 또는 요구 사항의 변경을 처리해야 하며, 장애를 감지하고 자동으로 복구되도록 설계되어야 합니다.

모범 사례

주제

- 기반

- 워크로드 아키텍처
- 변경 사항 관리
- 장애 관리

기반

기반에 관한 요구 사항은 그 범위가 단일 워크로드 또는 프로젝트 이상으로 확장됩니다. 시스템을 설계할 때는 먼저 안정성을 좌우하는 기반에 관한 요구 사항부터 갖춰야 합니다. 예를 들어, 데이터 센터의 네트워크 대역폭을 충분히 확보해야 합니다.

AWS에서는 이러한 기반에 관련된 요구 사항이 대부분 이미 통합되어 있거나 필요에 따라 적용할 수 있습니다. 클라우드는 거의 한계가 없도록 설계되었기 때문에 충분한 네트워킹 및 컴퓨팅 용량에 대한 요구 사항을 충족할 책임은 AWS에 있습니다. 따라서 고객은 리소스 크기와 할당을 필요에 따라 자유롭게 변경할 수 있습니다.

다음은 안정성 고려 사항에 중점을 둔 질문입니다. 안정성 관련 질문 및 모범 사례 목록은 부록을 참조하십시오.

REL 1: 서비스 할당량과 제약 조건은 어떻게 관리합니까?

클라우드 기반 워크로드 아키텍처에는 서비스 할당량(서비스 한도라고도 함)이라는 것이 있습니다. 이러한 할당량은 실수로 필요한 것보다 많은 리소스를 프로비저닝하는 것을 방지하고 API 작업에 대한 요청 속도를 제한하여 서비스를 침해로부터 보호하기 위해 존재합니다. 광섬유 케이블을 통해 비트를 전송할 수 있는 속도나 물리적 디스크의 스토리지 양과 같은 리소스 제약 조건도 있습니다.

REL 2: 네트워크 토폴로지는 어떻게 계획합니까?

워크로드는 여러 환경에 존재할 수 있습니다. 여기에는 다중 클라우드 환경(퍼블릭 액세스 및 프라이빗)과 기존 데이터 센터 인프라가 포함됩니다. 따라서 시스템 내부 및 시스템 간 연결, 퍼블릭 IP 주소 관리, 프라이빗 IP 주소 관리 및 도메인 이름 확인과 같은 네트워크 고려 사항을 계획에 포함해야 합니다.

클라우드 기반 워크로드 아키텍처에는 서비스 할당량(서비스 한도라고도 함)이라는 것이 있습니다. 이러한 할당량은 실수로 필요한 것보다 많은 리소스를 프로비저닝하는 것을 방지하고 API 작업에 대한 요청 속도를 제한하여 서비스를 침해로부터 보호하기 위해 존재합니다. 워크로드는 여러 환경에 존재

할 수 있습니다. 모든 워크로드 환경에 대해 이러한 할당량을 모니터링하고 관리해야 합니다. 여기에는 다중 클라우드 환경(퍼블릭 액세스 및 프라이빗)이 포함되며 기존 데이터 센터 인프라도 포함될 수 있습니다. 따라서 시스템 내부 및 시스템 간 연결, 퍼블릭 IP 주소 관리, 프라이빗 IP 주소 관리 및 도메인 이름 확인과 같은 네트워크 고려 사항을 계획에 포함해야 합니다.

워크로드 아키텍처

안정적인 워크로드는 소프트웨어와 인프라에 대한 사전 설계 결정에서 시작됩니다. 선택한 아키텍처는 모든 Well-Architected 원칙에서 워크로드 동작에 영향을 미칩니다. 안정성을 달성하려면 특정 패턴을 따라야 합니다.

AWS에서는 워크로드 개발자가 사용할 언어와 기술을 선택할 수 있습니다. AWS SDK는 AWS 서비스를 위한 언어별 API를 제공하여 코드 작성의 복잡성을 제거합니다. 이러한 SDK와 언어 선택을 통해 개발자는 여기에 나열된 안정성 모범 사례를 구현할 수 있습니다. 또한 개발자는 다음에서 Amazon의 소프트웨어 구축 및 운영 방법에 대해 자세히 알아볼 수 있습니다. Amazon Builders' Library.

다음은 안정성 고려 사항에 중점을 둔 질문입니다.

REL 3: 워크로드 서비스 아키텍처는 어떻게 설계합니까?

SOA(서비스 지향 아키텍처) 또는 마이크로서비스 아키텍처를 사용하여 확장성과 안정성이 뛰어난 워크로드를 구축합니다. SOA(서비스 지향 아키텍처)는 서비스 인터페이스를 통해 소프트웨어 구성 요소를 재사용 가능하게 만드는 방식입니다. 마이크로서비스 아키텍처는 구성 요소를 더 작고 간단하게 만듭니다.

REL 4: 분산 시스템에서 장애 방지를 위한 상호 작용은 어떻게 설계합니까?

분산 시스템에서 구성 요소(예: 서버 또는 서비스)는 통신 네트워크를 사용하여 상호 연결됩니다. 워크로드는 이러한 네트워크에서 데이터 손실 또는 지연 시간이 발생하더라도 안정적으로 작동해야 합니다. 분산 시스템의 구성 요소는 다른 구성 요소나 워크로드에 부정적인 영향을 미치지 않는 방식으로 작동해야 합니다. 여기에 나온 모범 사례는 장애를 방지하고 MTBF(평균 장애 간격)를 개선합니다.

REL 5: 분산 시스템에서 장애 완화 또는 극복을 위한 상호 작용은 어떻게 설계합니까?

분산 시스템에서 구성 요소(예: 서버 또는 서비스)는 통신 네트워크를 사용하여 상호 연결됩니다. 워크로드는 이러한 네트워크에서 데이터 손실 또는 지연 시간이 발생하더라도 안정적으로 작동해야 합니다. 분산 시스템의 구성 요소는 다른 구성 요소나 워크로드에 부정적인 영향을 미치지 않는 방식으로 작동해야 합니다. 이러한 모범 사례를 준수하면 워크로드가 스트레스 또는 장애를 견디고, 더 빠르게 이를 복구하며, 이러한 장애의 영향을 완화할 수 있습니다. 그러면 결과적으로 MTTR(평균 복구 시간)이 개선됩니다.

변경 사항 관리

워크로드의 안정적인 운영을 위해서는 워크로드 또는 환경에 대한 변경을 예상하고 수용해야 합니다. 변경에는 수요 급증과 같이 워크로드에 적용되는 변경은 물론 기능 배포 및 보안 패치와 같은 워크로드 내부의 변경이 포함됩니다.

AWS를 사용하면 워크로드 동작을 모니터링하고 KPI에 대한 대응을 자동화할 수 있습니다. 예를 들어 워크로드의 사용자가 증가하면 워크로드 서버를 추가할 수 있습니다. 워크로드 변경 권한을 가진 사용자를 관리하고 이러한 변경 기록을 감사할 수 있습니다.

다음은 안정성 고려 사항에 중점을 둔 질문입니다.

REL 6: 워크로드 리소스는 어떻게 모니터링합니까?

로그와 지표는 워크로드의 상태를 파악할 수 있는 유용한 도구입니다. 로그 및 지표를 모니터링하여 임계값을 초과하거나 중요한 이벤트가 발생하면 알림을 보내도록 워크로드를 구성할 수 있습니다. 모니터링을 수행하면 워크로드가 저성능 임계값을 초과하거나 장애가 발생할 때를 인식하고 이에 대응하여 자동으로 복구할 수 있습니다.

REL 7: 수요 변경에 따라 조정되도록 워크로드를 설계하려면 어떻게 해야 합니까?

확장 가능한 워크로드는 리소스를 자동으로 추가하거나 제거하여 특정 시기의 수요에 리소스 공급을 맞출 수 있는 탄력성을 제공합니다.

REL 8: 변경 사항은 어떻게 적용합니까?

새로운 기능을 배포하고 워크로드와 운영 환경에서 알려진 소프트웨어를 실행하고 예측 가능한 방식으로 패치 또는 교체할 수 있도록 하려면 변경 사항을 제어해야 합니다. 이러한 변경이 제어되지 않으면 변경의 영향을 예측하거나 변경으로 인해 발생하는 문제를 해결하기가 어려워집니다.

수요 변화에 따라 리소스를 자동으로 추가하거나 제거하도록 워크로드를 설계하면 안정성이 향상될 뿐 아니라 비즈니스 성공의 가능성도 높아집니다. 모니터링을 통해 KPI가 통상적인 수준을 벗어나면 담당 팀에 자동으로 알려 줍니다. 환경에 대한 변경 사항이 자동으로 로깅되므로 안정성에 영향을 미칠 가능성이 있는 작업을 감사하여 신속하게 파악할 수 있습니다. 변경 관리 제어를 통해 규칙을 적용함으로써 필요한 수준의 안정성을 확보할 수 있습니다.

장애 관리

통상적인 수준의 복잡한 시스템에는 장애가 발생하기 마련입니다. 안정성을 유지하려면 워크로드에서 장애가 발생할 때 이를 인식하고 가용성에 미치는 영향을 방지하는 조치를 취해야 합니다. 워크로드는 장애를 견디는 동시에 문제를 자동으로 복구할 수 있어야 합니다.

AWS에서는 자동화를 활용하여 모니터링 데이터에 대응합니다. 예를 들어, 특정 지표가 임계값을 넘어서면 자동화된 작업을 트리거하여 문제를 해결할 수 있습니다. 또한 운영 환경에서 장애가 발생한 리소스를 진단하여 수정하는 대신, 일단 새 리소스로 대체한 다음 운영 환경이 아닌 외부에서 장애 리소스를 분석해 볼 수도 있습니다. 클라우드에서는 저렴한 비용으로 전체 시스템의 임시 버전을 설정할 수 있기 때문에 전체 복구 프로세스를 자동으로 테스트하는 것이 가능합니다.

다음은 안정성 고려 사항에 중점을 둔 질문입니다.

REL 9: 데이터는 어떻게 백업합니까?

RTO(복구 시간 목표) 및 RPO(복구 시점 목표)에 대한 요구 사항을 충족하도록 데이터, 애플리케이션 및 구성을 백업합니다.

REL 10: 장애 격리를 사용하여 워크로드를 보호하려면 어떻게 해야 합니까?

장애 격리 경계는 워크로드 내부 장애의 영향을 제한된 수의 구성 요소로 제한합니다. 경계 외부의 구성 요소는 장애가 발생하더라도 영향을 받지 않습니다. 다수의 장애 격리 경계를 사용하여 워크로드에 미치는 영향을 제한할 수 있습니다.

REL 11: 구성 요소 장애를 견디도록 워크로드를 설계하려면 어떻게 해야 합니까?

고가용성 및 낮은 MTTR(평균 복구 시간)이 요구되는 워크로드는 복원력을 고려하여 설계해야 합니다.

REL 12: 안정성은 어떻게 테스트합니까?

프로덕션 환경의 스트레스에 대한 복원력을 가지도록 워크로드를 설계한 후 설계대로 작동하고 예상한 복원력을 제공하는지 확인할 수 있는 유일한 방법은 테스트입니다.

REL 13: DR(재해 복구)를 어떻게 계획합니까?

DR 전략의 시작은 백업 및 이중화 워크로드 구성 요소를 갖추는 것입니다. RTO 및 RPO는 워크로드 복원을 위한 목표입니다. 비즈니스 요구 사항에 따라 이러한 목표를 설정합니다. 워크로드 리소스 및 데이터의 위치와 기능을 고려하여 이러한 목표를 충족하는 전략을 구현합니다. 중단 가능성과 복구 비용도 워크로드에 대한 재해 복구 옵션을 갖추는 것의 비즈니스 가치를 파악하는 데 도움이 되는 주요 요소입니다.

정기적으로 데이터를 백업하고 백업 파일을 테스트하여 논리적 오류와 물리적 오류를 모두 복구할 수 있는지 확인하십시오. 빈번한 워크로드 자동 테스트를 통해 장애 원인을 파악하고 복구 방식을 살펴보는 것이 장애 관리의 열쇠입니다 정기 일정에 따라 이 테스트를 수행하고, 중요한 워크로드 변경 이후에도 이 테스트가 트리거되는지 확인해야 합니다. Recovery Time Objective(RTO), Recovery Point Objective(RPO) 등의 KPI를 적극적으로 추적하여 장애 테스트 시나리오 등에서 워크로드의 복원력을 평가합니다. KPI를 추적하면 단일 장애 지점을 파악 및 완화하는 데 도움이 됩니다. 목표는 워크로드 복구 프로세스를 철저히 테스트함으로써 모든 데이터를 복구할 수 있으며 문제가 지속되더라도 고객에게 계속 서비스를 제공할 수 있다는 확신을 얻는 것입니다. 통상적인 프로덕션 프로세스와 마찬가지로 복구 프로세스도 제대로 실행해야 합니다.

리소스

안정성 관련 AWS 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하십시오.

설명서

- AWS 설명서

- AWS 글로벌 인프라
- AWS Auto Scaling: 조정 계획 작동 방식
- AWS Backup이란 무엇입니까?

백서

- 신뢰성 원칙: AWS Well-Architected
- AWS에서 마이크로서비스 구현

성능 효율성

성능 효율성 원칙은 컴퓨팅 리소스를 시스템 요구 사항에 맞게 효율적으로 사용하고, 수요 변화 및 기술 진화에 발맞춰 그러한 효율성을 유지할 수 있는 역량을 포함합니다.

성능 효율성 원칙에서는 설계 원칙 개요, 모범 사례 및 질문 사항을 제공합니다. 구현 방법에 대한 선제적인 가이드는 성능 효율성 원칙 백서.

주제

- 설계 원칙
- 정의
- 모범 사례
- 리소스

설계 원칙

클라우드에는 5가지 성능 효율성 설계 원칙이 있습니다.

- 고급 기술의 대중화: 팀에서 고급 기술을 손쉽게 구현할 수 있도록 복잡한 태스크를 클라우드 공급업체에 위임합니다. IT 팀에 새로운 기술의 호스팅 및 실행에 대해 알아볼 것을 요청하는 대신 기술을 서비스 형태로 사용하는 것을 고려해보십시오. 예를 들어 NoSQL 데이터베이스, 미디어 트랜스코딩 및 기계 학습은 모두 전문 지식이 요구되는 기술입니다. 클라우드에서는 이러한 기술이 팀에서 사용할 수 있는 서비스 형식으로 제공되므로 팀은 리소스 프로비저닝 및 관리가 아닌 제품 개발에 집중할 수 있습니다.
- 몇 분 만에 전세계에 배포: 전 세계의 여러 AWS 리전에 워크로드를 배포하면 최소한의 비용으로 지연 시간을 줄이고 고객 경험을 개선할 수 있습니다.

- 서버리스 아키텍처 사용: 서버리스 아키텍처에서는 물리적 서버를 실행하고 유지 관리하지 않고도 기존의 컴퓨팅 활동을 수행할 수 있습니다. 예를 들어 서버리스 스토리지 서비스를 정적 웹 사이트로 사용하고(웹 서버 불필요) 이벤트 서비스를 통해 코드를 호스팅할 수 있습니다. 이렇게 하면 물리적 서버 관리로 인한 운영 부담이 없어집니다. 또한 이러한 관리형 서비스는 클라우드 규모에서 운영되므로 트랜잭션 비용을 절감할 수 있습니다.
- 실험 횟수 증가: 자동화할 수 있는 가상 리소스를 활용하면 여러 가지 인스턴스, 스토리지 또는 구성에 대한 비교 테스트를 신속하게 수행할 수 있습니다.
- 기계적 동조 고려: 클라우드 서비스가 어떻게 사용되는지 파악하고 항상 워크로드 목표에 가장 부합하는 기술 접근 방식을 사용합니다. 예를 들어 데이터베이스 또는 스토리지 접근 방식을 선택할 때는 데이터 접근 패턴을 고려합니다.

정의

클라우드의 성능 효율성에는 4가지 모범 사례 영역이 있습니다.

- 선택의 폭
- 검토
- 모니터링
- 절충

고성능 아키텍처 구축 시 데이터 기반 접근 방식을 취합니다. 개괄적 설계부터 리소스 유형 선택 및 구성에 이르는 아키텍처의 모든 측면에 대한 데이터를 수집합니다.

정기적으로 선택 사항을 검토하면서 진화를 거듭하는 AWS 클라우드를 최대한 활용할 수 있습니다. 모니터링을 수행하면 예상 성능과의 차이를 확인할 수 있습니다. 압축 또는 캐싱을 사용하거나 일관성 요구 사항을 완화하는 등의 절충을 통해 아키텍처를 설계하면 성능을 개선할 수 있습니다.

모범 사례

주제

- 선택
- 검토
- 모니터링
- 절충

선택

특정 워크로드에 대한 최적의 솔루션은 다양하며, 종종 여러 접근 방식이 결합된 솔루션을 사용합니다. Well-Architected 워크로드의 경우 다수의 솔루션이 사용되며 다양한 특성을 사용하여 성능을 높일 수 있습니다.

AWS 리소스는 다양한 유형과 구성으로 제공되므로, 워크로드 요구 사항에 가장 근접한 접근 방식을 쉽게 찾을 수 있습니다. 또한 온프레미스 인프라에서는 쉽게 사용할 수 없는 옵션도 제공됩니다. 예를 들어 Amazon DynamoDB와 같은 관리형 서비스는 완전관리형 NoSQL 데이터베이스로, 어떤 규모에서도 지연 시간이 한 자릿수 밀리초 단위로 매우 짧습니다.

다음은 성능 효율성 고려 사항에 중점을 둔 질문입니다. 성능 효율성 관련 질문 및 모범 사례 목록은 부록을 참조하십시오.

PERF 1: 최고의 성능을 제공하는 아키텍처를 선택하려면 어떻게 해야 합니까?

워크로드에서 최적의 성능을 얻으려면 여러 접근 방식을 취해야 합니다. Well-Architected 시스템은 다수의 솔루션 및 기능을 사용하여 성능을 개선합니다.

데이터 기반 접근 방식으로 아키텍처에 적합한 패턴 및 구현을 선택하면 경제적인 솔루션을 구축할 수 있습니다. AWS 솔루션스 아키텍트, AWS 참조 아키텍처 및 AWS 파트너 네트워크(APN) 파트너는 업계 지식을 바탕으로 사용자가 아키텍처를 선택하는 과정을 도울 수 있습니다. 하지만 아키텍처를 최적화하려면 벤치마킹 또는 로드 테스트를 통해 데이터를 획득해야 합니다.

아키텍처는 여러 아키텍처 접근 방식(예: 이벤트 기반, ETL 또는 파이프라인)을 결합하게 될 가능성이 높습니다. 아키텍처 구현에는 아키텍처 성능 최적화에 적합한 AWS 서비스를 사용하게 됩니다. 다음 섹션에서는 고려할 네 가지 주요 리소스 유형(컴퓨팅, 스토리지, 데이터베이스, 네트워크)에 대해 살펴봅니다.

컴퓨팅

성능 요구 사항을 충족하고 비용 및 작업 효율을 대폭 개선하는 컴퓨팅 리소스를 선택하면 동일한 수의 리소스로 더 많은 것을 달성할 수 있습니다. 컴퓨팅 옵션을 평가할 때는 워크로드 성능 및 비용에 대한 요구 사항을 인지하고 이를 사용하여 정보에 기반을 둔 의사 결정을 내려야 합니다.

AWS에서는 3가지 형식의 컴퓨팅 기능(인스턴스, 컨테이너, 함수)이 제공됩니다.

- 인스턴스 는 가상화된 서버이기 때문에 버튼을 누르거나 API를 호출하는 방법으로 기능을 변경할 수 있습니다. 클라우드에서는 리소스를 한번 결정하면 그대로 고정되는 것이 아니므로 다양한 서버 유

형을 시험해 볼 수 있습니다. AWS에서는 이러한 가상 서버 인스턴스를 다양한 제품군과 규모로 제공하며, 솔리드 스테이트 드라이브(SSD)와 그래픽 처리 장치(GPU)를 비롯한 폭넓은 기능을 제공합니다.

- 컨테이너 는 애플리케이션 및 종속성을 리소스가 격리된 프로세스에서 실행할 수 있는 운영 체제 가상화 방식입니다. AWS Fargate는 컨테이너용 서버리스 컴퓨팅입니다. 컴퓨팅 환경의 설치, 구성 및 관리를 제어해야 한다면 Amazon EC2를 사용할 수 있습니다. Amazon Elastic Container Service(ECS) 또는 Amazon Elastic Kubernetes Service(EKS)와 같은 다수의 컨테이너 오케스트레이션 플랫폼 중에서 선택할 수도 있습니다.
- 함수 기능은 실행하려는 코드에서 실행 환경을 추상화합니다. 예를 들어, AWS Lambda를 이용하면 인스턴스를 실행하지 않고도 코드를 실행할 수 있습니다.

다음은 성능 효율성 고려 사항에 중점을 둔 질문입니다.

PERF 2: 컴퓨팅 솔루션을 어떻게 선택합니까?

워크로드에 가장 적합한 컴퓨팅 솔루션은 애플리케이션 설계, 사용량 패턴 및 구성 설정에 따라 다릅니다. 아키텍처는 다양한 구성 요소에 대해 서로 다른 컴퓨팅 솔루션을 사용하고 다양한 기능을 활성화하여 성능을 개선할 수 있습니다. 아키텍처에 대해 잘못된 컴퓨팅 솔루션을 선택하면 성능 효율성 저하로 이어질 수 있습니다.

컴퓨팅 사용을 설계할 때는 이용 가능한 탄력성 메커니즘을 활용하여 수요 변화에 맞춰 성능을 유지할 수 있는 충분한 용량을 확보해야 합니다.

스토리지

클라우드 스토리지는 워크로드에 사용되는 정보를 보관하는 클라우드 컴퓨팅의 중요한 구성 요소입니다. 클라우드 스토리지는 일반적으로 기존 온프레미스 스토리지 시스템보다 안정성, 확장성 및 보안이 뛰어납니다. 객체, 블록 및 파일 스토리지 서비스와 워크로드에 사용할 클라우드 데이터 마이그레이션 옵션 중에서 선택합니다.

AWS에서 스토리지는 객체, 블록, 파일이라는 3가지 형태로 제공됩니다.

- 객체 스토리지 는 모든 인터넷 위치에서 사용자가 생성한 콘텐츠, 활성 아카이브, 서버리스 컴퓨팅, 빅 데이터 스토리지 또는 백업 및 복구를 위한 데이터에 액세스할 수 있게 하는 확장 가능하고 내구성이 뛰어난 플랫폼을 제공합니다. Amazon Simple Storage Service(Amazon S3)는 업계 최고의 확장성, 데이터 가용성, 보안 및 성능을 제공하는 객체 스토리지 서비스입니다. Amazon S3는

99.999999999%의 내구성을 제공하도록 설계되었으며 전 세계 회사를 위한 수백만 개의 애플리케이션에 대한 데이터를 저장합니다.

- 블록 스토리지 는 DAS(Direct-Attached Storage) 또는 SAN(Storage Area Network)과 유사한 고가용성의 일관적이며 지연 시간이 짧은 블록 스토리지를 각 가상 호스트에 제공합니다. Amazon Elastic Block Store(Amazon EBS)는 EC2 인스턴스에서 영구 스토리지에 액세스할 수 있어야 하는 워크로드를 위해 설계되었으며, 적절한 스토리지 용량, 성능 및 비용으로 애플리케이션을 튜닝하는 데 도움이 됩니다.
- 파일 스토리지 를 사용하면 여러 시스템에서 공유 파일 시스템에 액세스할 수 있습니다. Amazon Elastic File System(EFS)과 같은 파일 스토리지 솔루션은 대용량 콘텐츠 리포지토리, 개발 환경, 미디어 스토어 또는 사용자 홈 디렉터리와 같은 사용 사례에 적합합니다. Amazon FSx를 사용하면 주요 파일 시스템을 비용 효율적으로 손쉽게 시작하고 실행할 수 있으므로, 널리 사용되는 오픈 소스 및 상용 라이선스 파일 시스템의 풍부한 기능 세트와 빠른 성능을 활용할 수 있습니다.

다음은 성능 효율성 고려 사항에 중점을 둔 질문입니다.

PERF 3: 스토리지 솔루션을 어떻게 선택합니까?

시스템에 대한 최적의 스토리지 솔루션은 액세스 방식 종류(블록, 파일, 객체), 액세스 패턴(랜덤 또는 순차), 필요한 처리량, 액세스 빈도(온라인, 오프라인, 보관), 업데이트 빈도(WORM, 동적) 및 가용성과 내구성 제약 사항에 따라 다릅니다. 설계가 잘된 시스템은 여러 스토리지 솔루션을 사용하며, 다양한 기능을 통해 성능을 개선하고 리소스를 효율적으로 사용할 수 있도록 지원합니다.

원하는 성능을 달성하려면 스토리지 솔루션을 선택할 때 액세스 패턴에 일치하는지 확인하는 것이 중요합니다.

데이터베이스

클라우드는 워크로드에서 발생하는 다양한 문제를 해결할 수 있도록 특별히 구축된 데이터베이스 서비스를 제공합니다. 관계형, 키-값, 문서, 인메모리, 그래프, 시계열 및 원장 데이터베이스를 비롯하여 특별히 구축된 다양한 데이터베이스 엔진 중에서 선택할 수 있습니다. 특정 문제 또는 문제 그룹을 해결할 수 있는 최고의 데이터베이스를 선택할 수 있으므로 제한적이고 획일적인 범용 데이터베이스에서 벗어나 고객의 성능 요구 사항을 충족하는 애플리케이션을 구축하는 데 집중할 수 있습니다.

AWS에서는 관계형, 키-값, 문서, 인메모리, 그래프, 시계열 및 원장 데이터베이스를 비롯해서 다양한 목적별 데이터베이스 엔진 중에서 선택할 수 있습니다. AWS 데이터베이스를 사용하면 서버 프로비저닝, 패치 적용, 설정, 구성, 백업 또는 복구와 같은 데이터베이스 관리 작업을 신경 쓸 필요가 없습니다.

AWS에서 클러스터를 지속적으로 모니터링하여 자가 복구 스토리지 및 자동화된 확장 기능을 통해 워크로드를 끊김 없이 운영하므로, 보다 가치 높은 애플리케이션 개발에 집중할 수 있습니다.

다음은 성능 효율성 고려 사항에 중점을 둔 질문입니다.

PERF 4: 데이터베이스 솔루션을 어떻게 선택합니까?

시스템에 대한 최적의 데이터베이스 솔루션은 가용성, 일관성, 파티션 허용 오차, 지연 시간, 내구성, 확장성, 쿼리 기능에 대한 요구 사항에 따라 다릅니다. 여러 시스템은 다양한 하위 시스템에 서로 다른 데이터베이스 솔루션을 사용하고 다양한 기능을 활성화하여 성능을 개선할 수 있습니다. 시스템에 대해 잘못된 데이터베이스 솔루션 및 기능을 선택하면 성능 효율성이 저하될 수 있습니다.

워크로드의 데이터베이스 접근 방식은 성능 효율성에 상당한 영향을 미칩니다. 데이터베이스 영역은 데이터 기반 접근 방식이 아니라 조직의 기본값에 따라 선택되는 경우가 많습니다. 스토리지와 마찬가지로 워크로드의 액세스 패턴을 고려하는 것이 중요하며, 다른 비데이터베이스 솔루션(예: 그래프, 시계열 또는 인메모리 스토리지 데이터베이스 사용)이 보다 효율적으로 문제를 해결할 수 있는지 여부도 고려해야 합니다.

네트워크

네트워크는 모든 워크로드 구성 요소 사이에 있기 때문에 워크로드 성능 및 동작에 긍정적, 부정적인 영향을 미칠 수 있습니다. 또한 클러스터 성능을 높이는 데 있어서 심층적인 네트워크 지식이 요구되는 HPC(고성능 컴퓨팅)와 같이 네트워크 성능에 크게 의존하는 워크로드도 있습니다. 따라서 대역폭, 지연 시간, 지터 및 처리량에 대한 워크로드 요구 사항을 결정해야 합니다.

AWS에서 네트워킹은 가상화되고 다양한 유형 및 구성으로 제공됩니다. 따라서 요구에 부응하는 네트워킹 방법을 보다 쉽게 찾을 수 있습니다. AWS에서는 제품 기능(예: 강화된 네트워킹, Amazon EBS 최적화 인스턴스, Amazon S3 Transfer Acceleration, 동적 Amazon CloudFront)을 지원하여 네트워크 트래픽을 최적화합니다. 나아가 AWS는 네트워킹 기능(예: Amazon Route 53 지연 속도 기반 라우팅, Amazon VPC 엔드포인트, AWS Direct Connect, AWS Global Accelerator)까지 제공하여 네트워크 거리 또는 지터를 줄여 줍니다.

다음은 성능 효율성 고려 사항에 중점을 둔 질문입니다.

PERF 5: 네트워킹 솔루션을 구성하려면 어떻게 해야 합니까?

워크로드에 대한 최적의 네트워크 솔루션은 지연 시간, 처리량 요구 사항, 지터 및 대역폭에 따라 다릅니다. 위치 옵션은 사용자 또는 온프레미스 리소스와 같은 물리적 제약에 따라 결정됩니다. 엣지 로케이션 또는 리소스 배치를 통해 이러한 제약을 상쇄할 수 있습니다.

네트워크를 배포할 때는 위치를 고려해야 합니다. 리소스를 사용할 위치 가까이 해당 리소스가 배치되도록 선택하여 거리를 줄일 수 있습니다. 워크로드가 변경되면 네트워킹 지표를 사용하여 네트워킹 구성을 변경합니다. 리전, 배치 그룹 및 엣지 서비스를 활용하여 성능을 크게 개선할 수 있습니다. 클라우드 기반 네트워크는 빠르게 재구축되거나 수정될 수 있으므로 성능 효율성을 유지하려면 네트워크 아키텍처를 지속적으로 변경해야 합니다.

검토

클라우드 기술은 빠르게 발전하고 있습니다. 따라서 지속적으로 성능을 개선하려면 워크로드 구성 요소에 최신 기술 및 접근 방식이 사용되고 있는지 확인해야 합니다. 또한 워코로드 구성 요소에 대한 변경을 지속적으로 평가하고 고려하여 성능 및 비용 목표가 충족되고 있는지 확인해야 합니다. 기계 학습 및 AI(인공 지능)과 같은 새로운 기술을 사용하면 새로운 고객 경험을 구축하고 모든 비즈니스 워크로드에 걸쳐 혁신을 달성할 수 있습니다.

고객의 요구 사항의 의해 주도되는 AWS의 지속적인 혁신을 활용하십시오. AWS는 정기적으로 새로운 리전, 엣지 로케이션, 서비스 및 기능을 출시합니다. 이러한 릴리스를 적용하면 아키텍처의 성능 효율성을 개선할 수 있습니다.

다음은 성능 효율성 고려 사항에 중점을 둔 질문입니다.

PERF 6: 새 릴리스를 활용하기 위해 워크로드를 어떻게 발전시킵니까?

워크로드 설계 시 선택할 수 있는 옵션은 한정되어 있습니다. 그러나 시간이 지나면 워크로드의 성능을 개선할 수 있는 새로운 기술과 접근 방식을 사용할 수 있게 됩니다.

아키텍처의 성능 저하는 성능 검토 프로세스가 없거나 효과적이지 않은 경우 주로 발생합니다. 아키텍처의 성능이 불량한 경우 이 성능 검토 프로세스를 시행하면 Deming의 PDCA(계획-작업-검사-조치) 주기를 적용해 반복 과정을 개선할 수 있습니다.

모니터링

워크로드를 구현한 후에는 고객이 영향을 받기 전에 모든 문제를 해결할 수 있도록 성능을 모니터링해야 합니다. 모니터링 지표를 사용하여 임계값을 초과할 경우 경보가 생성되도록 해야 합니다.

Amazon CloudWatch는 워크로드 모니터링, 시스템 전체 성능 변경에 대한 대응, 리소스 사용률 최적화 및 운영 상태의 통합 보기 확인에 필요한 데이터와 유용한 인사이트를 제공하는 모니터링 및 관찰 서비스입니다. CloudWatch는 AWS 및 온프레미스 서버에서 실행되는 워크로드의 로그, 지표 및 이벤트 형태로 모니터링 및 운영 데이터를 수집합니다. AWS X-Ray는 개발자가 프로덕션의 분산 애플리케이션을 분석하고 디버깅하는 데 도움이 됩니다. AWS X-Ray를 사용하면 애플리케이션의 성능을 파악하고 근본 원인을 발견하며 성능 병목 현상을 식별할 수 있습니다. 이러한 분석 정보를 활용해 문제에 빠르게 대응하고 워크로드가 원활하게 실행되는 상태를 유지할 수 있습니다.

다음은 성능 효율성 고려 사항에 중점을 둔 질문입니다.

PERF 7: 리소스 성능을 모니터링하려면 어떻게 해야 합니까?

시스템 성능은 시간이 지남에 따라 저하될 수 있습니다. 시스템 성능을 모니터링하여 성능 저하 상태를 식별하고 운영 체제 또는 애플리케이션 로드와 같은 내부 또는 외부 요인을 해결합니다.

효율적인 모니터링 솔루션에서는 오탐이 발생하지 않아야 합니다. 자동 트리거는 수동 작업으로 인한 오류를 방지하고 문제를 해결하는 데 걸리는 시간을 줄일 수 있습니다. 프로덕션 환경에서 시뮬레이션이 진행되는 게임 데이를 계획하여 경보 솔루션을 테스트하고 해당 솔루션이 문제를 정확하게 인지하는지를 확인합니다.

절충

솔루션을 설계할 때는 최적의 방식이 적용되도록 각 방식의 장단점을 고려해야 합니다. 상황에 따라 더 우수한 성능을 제공하기 위해 일관성/내구성/공간 및 시간/지연 시간 중 우선적으로 고려할 요소를 선택할 수 있습니다.

AWS를 사용하면 몇 분 내에 전 세계의 여러 위치에 리소스를 배포하여 최종 사용자에게 더욱 가깝게 다가갈 수 있습니다. 또한 데이터베이스 시스템과 같은 정보 스토어에 읽기 전용 복제본을 동적으로 추가하여 기본 데이터베이스의 로드를 줄일 수 있습니다.

다음은 성능 효율성 고려 사항에 중점을 둔 질문입니다.

PERF 8: 절충을 통해 성능을 개선하려면 어떻게 해야 합니까?

솔루션을 설계할 때 절충이 필요한 영역을 결정하면 최적의 접근 방식을 선택할 수 있습니다. 일관성, 내구성 및 공간을 포기하는 대신, 시간 및 지연 시간을 선택하여 성능을 개선할 수 있는 경우가 많습니다.

워크로드를 변경할 때는 지표를 수집 및 평가하여 변경의 영향을 확인합니다. 시스템 및 최종 사용자에 대한 영향을 모두 측정하여 절충 작업이 워크로드에 미치는 영향을 파악합니다. 로드 테스트 등의 체계적인 방식을 사용해 개별 요소 트레이드-오프 시, 성능을 개선할 수 있는지 여부를 파악합니다.

리소스

성능 효율성 관련 AWS 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하세요.

설명서

- Amazon S3 성능 최적화
- Amazon EBS 볼륨 성능

백서

- 성능 효율성 원칙

동영상

- AWS re:Invent 2019: Amazon EC2 foundations(CMP211-R2)
- AWS re:Invent 2019: Leadership session: Storage state of the union(STG201-L)
- AWS re:Invent 2019: Leadership session: AWS purpose-built databases(DAT209-L)
- AWS re:Invent 2019: Connectivity to AWS and hybrid AWS network architectures(NET317-R1)
- AWS re:Invent 2019: Powering next-gen Amazon EC2: Deep dive into the Nitro system(CMP303-R2)
- AWS re:Invent 2019: Scaling up to your first 10 million users(ARC211-R)

비용 최적화

비용 최적화 원칙은 시스템을 실행하여 최저 가격으로 비즈니스 가치를 제공할 수 있는 역량을 포함합니다.

비용 최적화 부문에서는 설계 원리 개요, 모범 사례 및 질문 사항을 제공합니다. 구현 방법에 대한 선제적인 가이드는 비용 최적화 원칙 백서를 참조하십시오.

주제

설계 원칙

클라우드에는 5가지 비용 최적화 설계 원칙이 있습니다.

- 클라우드 재무 관리 구현: 클라우드에서 재정적 성공을 거두고 비즈니스 가치 실현을 가속화하려면 클라우드 재무 관리/비용 최적화에 투자해야 합니다. 조직이 이 새로운 기술 및 사용 관리 영역에서 역량을 쌓기 위해서는 시간과 리소스를 할애해야 합니다. 보안 또는 운영 우수성 역량과 마찬가지로 지식 강화, 프로그램, 리소스 및 프로세스를 통해 역량을 쌓아 비용 효율적인 조직이 되어야 합니다.
- 소비 모델 도입: 정교한 예측 기능을 사용하지 않아도 필요한 컴퓨팅 리소스에만 비용을 지불하고 비즈니스 요구 사항에 따라 사용량을 늘리거나 줄입니다. 예를 들어 개발 및 테스트 환경은 주로 주중 근무일에 하루 8시간 동안만 사용됩니다. 사용되지 않는 동안 이러한 리소스를 중단하여 잠재적으로 75%의 비용을 절감할 수 있습니다(40시간과 168시간의 차이).
- 전반적인 효율성 측정: 워크로드의 비즈니스 결과 및 워크로드 제공과 관련된 비용을 측정합니다. 이러한 측정을 통해 늘어난 성과와 절감한 비용으로 얻을 수 있는 이익을 확인할 수 있습니다.
- 획일적인 업무 부담에 대한 비용 지출 중단: 서버를 랙에 설치하고 스태킹하며, 서버에 전원을 공급하는 등의 까다로운 데이터 센터 운영 작업을 AWS가 처리합니다. 또한 관리형 서비스를 통해 운영 체제 및 애플리케이션을 관리하는 운영 부담을 덜어줍니다. 따라서 IT 인프라 대신 고객과 비즈니스 프로젝트에 집중할 수 있습니다.
- 비용 분석 및 기여도 파악: 클라우드를 사용하면 손쉽게 시스템의 사용량과 비용을 정확하게 식별할 수 있어 개별 워크로드 소유자가 IT 비용에 대한 투명한 기여도를 확인할 수 있습니다. 이를 통해 ROI(투자 대비 수익률)를 측정할 수 있으며 워크로드 소유자는 리소스를 최적화하고 비용을 절감하는 기회를 얻을 수 있습니다.

정의

클라우드의 비용 최적화에는 5가지 모범 사례 영역이 있습니다.

- 클라우드 재무 관리 시행
- 지출 및 사용량 인식
- 비용 효율적인 리소스
- 수요 관리 및 리소스 공급
- 시간 경과에 따른 최적화

Well-Architected 프레임워크의 다른 원칙과 마찬가지로 비교 분석을 통해 하나를 선택해야 합니다. 예를 들어 출시 시간에 최적화할지 아니면 비용에 최적화할지 선택합니다. 선결제 비용 최적화에 투자하는 것보다 출시 시간을 단축하거나 새로운 기능을 배포하거나 단순히 납기를 준수하는 등 속도를 가장 높일 수 있도록 최적화하는 것이 가장 좋은 경우도 있습니다. 데이터를 고려하지 않고 급하게 설계 결정이 내려지는 경우도 있으며, 가장 비용 최적화된 구축 벤치마킹에 시간을 쓰기보다 “만약을 대비해” 과잉 지출을 하려는 유혹은 항상 존재합니다. 이러한 경우에는 배포가 과다하게 프로비저닝되고 제대로 최적화되지 않을 수 있습니다. 하지만 온프레미스 환경에서 클라우드로 리소스를 그대로 이전한 후에 최적화를 수행해야 하는 경우에는 이러한 방식을 선택해야 합니다. 사전에 비용 최적화 전략에 적당한 노력을 들여 모범 사례를 일관적으로 준수하고 불필요한 오버프로비저닝을 방지하면 클라우드의 경제적 이점을 더 빨리 실현할 수 있습니다. 다음 섹션에서는 클라우드 재무 관리의 초기 및 지속적인 구현과 워크로드의 비용 최적화를 위한 기술과 모범 사례를 제공합니다.

모범 사례

주제

클라우드 재무 관리 시행

클라우드가 도입됨에 따라 기술 팀은 승인, 조달 및 인프라 배포 주기 단축으로 더 빠르게 혁신합니다. 비즈니스 가치를 실현하고 재정적 성공을 거두려면 클라우드에서의 재무 관리에 대한 새로운 접근 방

식이 필요합니다. 이 접근 방식은 클라우드 재무 관리이며 조직 전반에 걸친 지식 구축, 프로그램, 리소스 및 프로세스를 구현하여 조직 전체의 역량을 강화합니다.

많은 조직은 서로 다른 우선순위가 지정된 여러 단위로 구성됩니다. 합의를 통해 정한 일련의 재무 목표에 따라 조직을 조정하고 목표 달성을 위한 메커니즘을 조직에 제공할 수 있으면 조직의 효율성이 향상됩니다. 역량 있는 조직은 더 빠르게 혁신하고 구축하며, 더 민첩하고, 내부 또는 외부 요인에 적응합니다.

AWS에서 Cost and Usage Report(CUR)와 함께 Cost Explorer와 필요에 따라 Amazon Athena 및 Amazon QuickSight를 활용하여 비용과 사용량에 대한 조직 전체의 인식을 재고할 수 있습니다. AWS Budgets는 비용 및 사용량 알림을 사전에 제공합니다. AWS 블로그는 새로운 서비스 릴리스를 숙지할 수 있도록 신규 서비스와 기능 정보를 제공합니다.

다음은 비용 최적화 고려 사항에 중점을 둔 질문입니다. 비용 최적화 관련 질문 및 모범 사례 목록은 부록을 참조하십시오.

COST 1: 클라우드 재무 관리를 어떻게 구현합니까?

조직은 클라우드 재무 관리를 시행하여 비용과 사용량을 최적화하고 AWS에서 규모를 확대하면서 비즈니스 가치를 실현하고 재정적 성공을 거둘 수 있습니다.

비용 최적화 역할을 구성할 때는 팀원을 활용하고 CFM 및 비용 최적화 전문가로 팀을 보완합니다. 기존 팀원들은 조직이 현재 어떻게 작용하며 어떻게 개선 사항을 신속하게 실행하는지 알게 됩니다. 또한 분석 및 프로젝트 관리와 같은 보조 또는 전문 기술을 보유한 인력 투입도 고려하십시오.

조직에서 비용에 대한 인식을 이행할 때는 기존 프로그램과 프로세스를 바탕으로 개선하거나 구축합니다. 새로 구축하는 것보다 기존 프로세스와 프로그램에 추가하는 것이 훨씬 더 빠릅니다. 이를 통해 훨씬 더 빠르게 성과를 올릴 수 있습니다.

지출 및 사용량 인식

클라우드에서는 향상된 유연성과 민첩성을 바탕으로 혁신을 촉진하고 개발 및 배포를 가속화할 수 있습니다. 이는 하드웨어 사양을 식별하고, 가격 견적을 협상하고, 주문 번호를 관리하고, 배송을 예약한 후 리소스 배포하기와 같은 온프레미스 인프라 프로비저닝과 연관된 시간 및 수동 프로세스를 제거합니다. 하지만 사용이 편리하고 온디맨드 용량이 사실상 무제한으로 제공되면 지출을 새로운 방식으로 고려해야 합니다.

많은 비즈니스는 다양한 팀에서 운영하는 여러 시스템으로 구성되어 있습니다. 개별 조직 또는 제품 소유자에게 리소스 비용을 부여하는 기능은 효율적인 사용 행동 양식으로 이어지고 낭비되는 요소를 줄

여 줍니다. 또한 정확한 비용 기여도를 통해 수익성 높은 제품을 파악하고 예산을 어디에 할당할지에 대해 더 근거 있는 결정을 내릴 수 있습니다.

AWS에서는 AWS Organizations 또는 AWS Control Tower를 사용하여 계정 구조를 형성합니다. 이를 통해 비용과 사용량을 할당하고 분리할 수 있습니다. 리소스 태깅을 사용하여 사용량과 비용에 비즈니스 및 조직 정보를 적용할 수도 있습니다. AWS Cost Explorer를 사용하여 비용과 사용량을 파악하거나 Amazon Athena와 Amazon QuickSight로 사용자 지정 대시보드 및 분석을 만들 수 있습니다. 비용 및 사용량 제어는 AWS Budgets를 통한 알림과 AWS Identity and Access Management(IAM) 및 Service Quotas를 사용한 제어 기능을 통해 이루어집니다.

다음은 비용 최적화 고려 사항에 중점을 둔 질문입니다.

COST 2: 사용량을 어떻게 관리합니까?

목표 달성 과정에서 발생하는 비용을 적정 수준으로 유지하는 정책과 메커니즘을 설정합니다. '견제와 균형' 방식을 도입하면 비용을 과도하게 지출하지 않고 혁신을 이룰 수 있습니다.

COST 3: 사용량과 비용을 어떻게 모니터링합니까?

비용을 모니터링하고 적절하게 할당하기 위한 정책 및 절차를 구성합니다. 이렇게 하면 이 워크로드의 비용 효율성을 측정하고 개선할 수 있습니다.

COST 4: 리소스를 어떻게 폐기합니까?

프로젝트 시작부터 마지막까지의 전체 과정에서 변경 제어 및 리소스 관리를 구현합니다. 그러면 사용되지 않은 리소스를 종료하여 낭비되는 리소스를 줄일 수 있습니다.

비용 할당 태그를 사용하여 AWS 사용량과 비용을 분류하고 추적할 수 있습니다. AWS 리소스(예: EC2 인스턴스 또는 S3 버킷)에 태그를 적용할 경우 AWS는 사용량과 태그가 포함된 비용 및 사용량 보고서를 생성합니다. 조직 카테고리(예: 비용 센터, 워크로드 이름 또는 소유자)를 나타내는 태그를 적용하여 여러 서비스 전반에서 비용을 조직화할 수 있습니다.

비용과 사용량 보고 및 모니터링에서 적절한 수준의 세부 정보와 세분화를 사용해야 합니다. 개략적인 인사이트와 추세를 위해서는 AWS Cost Explorer에서 일 단위의 세부 수준을 사용합니다. 심층적

인 분석과 검사를 위해서는 시간 단위의 CUR(비용 및 사용 보고서)과 함께 AWS Cost Explorer 또는 Amazon Athena와 Amazon QuickSight에서 시간 단위의 세부 수준을 사용합니다.

태그가 지정된 리소스와 엔터티 수명 주기 추적(직원, 프로젝트)을 결합하면 조직에 더 이상 가치를 제공하지 않아 폐기해야 할 고립된 리소스나 프로젝트를 식별할 수 있습니다. 예상되는 초과 지출에 대한 통지를 받도록 결제 알림을 설정할 수 있습니다.

비용 효율적인 리소스

워크로드에 적합한 인스턴스와 리소스 사용은 비용 절감의 핵심입니다. 예를 들어 보고 프로세스에서 보다 작은 서버를 운영하는 데는 5시간이 걸리지만 2배로 비싼 더 큰 서버를 운영하는 데는 1시간이 걸릴 수 있습니다. 두 서버가 모두 동일한 결과를 내지만 보다 작은 서버는 시간에 따라 더 높은 비용이 발생합니다.

잘 설계된 워크로드는 상당히 긍정적인 비용적 영향을 미칠 수 있는 가장 비용 효율적인 리소스를 사용합니다. 또한 관리형 서비스를 사용하여 비용을 절감할 기회도 얻게 됩니다. 예를 들어 이메일을 전송하는 서버를 유지 관리하는 것 외에 메시지당을 기준으로 부과되는 서비스를 사용할 수 있습니다.

AWS는 유연하고 비용 효율적인 요금 옵션을 매우 다양하게 제공하므로, 요구 사항에 가장 적합한 방식으로 Amazon EC2 및 다른 서비스의 인스턴스를 사용할 수 있습니다. 온디맨드 인스턴스 최소 약정이 필요 없으며 시간 단위로 컴퓨팅 파워 비용을 지불할 수 있습니다. Savings Plans 및 예약 인스턴스 는 온디맨드 요금보다 최대 75% 할인된 요금을 제공합니다. 스팟 인스턴스를 사용하면 미사용 Amazon EC2 용량을 활용할 수 있으며 온디맨드 요금보다 최대 90% 할인된 요금을 제공합니다. 스팟 인스턴스 는 시스템이 상태 비저장 웹 서버, 일괄 처리 또는 HPC 및 빅 데이터를 사용하는 경우 등 개별 서버가 동적으로 오고갈 수 있는 서버 플릿 사용을 용인할 수 있는 데에 적합합니다.

적합한 서비스 선택으로 사용량 및 비용도 절감할 수 있습니다. 예를 들어, CloudFront를 사용하면 데이터 전송을 최소화하거나 소모 비용을 완전히 해소할 수 있으며, Amazon Aurora on RDS를 활용하면 값비싼 데이터베이스 라이선싱 비용을 해소할 수 있습니다.

다음은 비용 최적화 고려 사항에 중점을 둔 질문입니다.

COST 5: 서비스를 선택할 때 비용을 어떻게 평가합니까?

Amazon EC2, Amazon EBS 및 Amazon S3는 기본 구성 AWS 서비스입니다. Amazon RDS 및 Amazon DynamoDB와 같은 관리형 서비스는 더 높은 수준이거나 애플리케이션 수준의 AWS 서비스입니다. 기본 구성 서비스와 관리형 서비스를 적절히 선택하여 이 워크로드의 비용을 최적화할 수 있습니다. 예를 들어 관리형 서비스를 사용하면 관리 및 운영 고정 비용을 상당 부분 줄이고, 응용 프로그램 및 비즈니스 관련 활동에 집중할 수 있습니다.

COST 6: 리소스 유형, 크기 및 수 선택을 통해 비용 목표를 어떻게 달성합니까?

진행 중인 태스크에 대해 적절한 리소스 크기와 리소스 수를 선택해야 합니다. 가장 비용 효율적인 유형, 크기 및 수를 선택하여 리소스 낭비를 최소화할 수 있습니다.

COST 7: 비용 절감을 위해 가격 책정 모델을 어떻게 사용합니까?

해당 리소스에 대해 비용을 최소화하는 데 가장 적합한 요금 모델을 사용합니다.

COST 8: 데이터 전송 요금을 위한 계획은 어떻게 합니까?

비용 최소화를 위한 아키텍처 관련 사항을 결정할 수 있도록 데이터 전송 요금을 계획하고 모니터링해야 합니다. 아키텍처를 약간이라도 효율적으로 변경하면 장기적으로 운영 비용을 크게 줄일 수 있습니다.

서비스 선택 시 비용을 고려하고 Cost Explorer 및 AWS Trusted Advisor와 같은 도구를 통해 AWS 사용량을 정기적으로 검토하면서 사용률을 적극적으로 모니터링하고, 이에 따라 배포를 조절할 수 있습니다.

수요 관리 및 리소스 공급

클라우드로 이전하면 필요한 용량에 대한 비용만 지불하면 됩니다. 필요할 때 워크로드 수요에 맞춰 리소스를 공급할 수 있으므로 비용이 많이 들고 비경제적인 오버프로비저닝이 필요하지 않습니다. 또한 조절, 버퍼 또는 대기열을 통해 수요를 수정하여 수요를 원활하게 하고 더 적은 리소스로 수요를 처리함으로써 비용을 낮추거나 나중에 배치 서비스로 수요를 처리할 수 있습니다.

AWS에서는 워크로드 수요에 맞춰 리소스를 자동으로 프로비저닝할 수 있습니다. 수요 또는 시간 기반 접근 방식을 사용하는 Auto Scaling을 활용하면 필요한 만큼 리소스를 추가하고 제거할 수 있습니다. 수요의 변화를 예측할 수 있으면 비용을 더 많이 절약하고 워크로드 수요에 리소스를 맞출 수 있습니다. Amazon API Gateway를 사용하여 조절을 실행하거나 Amazon SQS를 사용하여 워크로드에서 대기열을 구현할 수 있습니다. 둘 다 워크로드 구성 요소에 대한 수요를 수정할 수 있습니다.

다음은 비용 최적화 고려 사항에 중점을 둔 질문입니다.

COST 9: 수요와 리소스 공급은 어떻게 관리합니까?

비용과 성능을 적절하게 절충한 워크로드에서는 비용을 결제한 모든 리소스가 사용되는지 확인하고, 사용률이 매우 낮은 인스턴스가 없도록 해야 합니다. 사용률 지표가 매우 높거나 낮으면 조직의 운영 비용(사용률이 너무 높아 성능이 저하됨)이 늘어나거나 과도한 프로비저닝으로 AWS 지출 금액이 낭비되는 등 조직에 악영향을 미칩니다.

수요 및 공급 리소스를 수정하도록 설계할 때는 사용 패턴, 새 리소스를 프로비저닝하는 데 걸리는 시간 및 수요 패턴의 예측 가능성을 적극적으로 고려하십시오. 수요를 관리할 때 대기열 또는 버퍼의 크기가 적절한지 그리고 필요한 시간 내에 워크로드 수요에 응답하고 있는지 확인해야 합니다.

시간 경과에 따른 최적화

AWS에서 신규 서비스와 기능이 출시되면 기존에 결정한 아키텍처 관련 사항을 검토하여 비용 측면에서 여전히 가장 효율적인 결정인지 확인하는 것이 좋습니다. 요구 사항이 바뀌면 더 이상 필요하지 않은 리소스, 전체 서비스 및 시스템을 과감하게 폐기합니다.

새로운 기능 또는 리소스 유형을 구현하면 변경 사항을 구현하는 데 필요한 노력을 최소화하면서 워크로드를 점진적으로 최적화할 수 있습니다. 이를 통해 장기적 효율성이 지속적으로 향상되고 최첨단 기술을 계속 활용하여 운영 비용을 절감할 수 있습니다. 구성 요소를 교체하거나 새 구성 요소를 워크로드에 신규 서비스와 함께 추가할 수도 있습니다. 이렇게 하면 효율성이 크게 향상될 수 있으므로 정기적으로 워크로드를 검토하고 신규 서비스와 기능을 구현하는 것이 반드시 필요합니다.

다음은 비용 최적화 고려 사항에 중점을 둔 질문입니다.

COST 10: 새로운 서비스를 어떻게 평가합니까?

AWS에서 신규 서비스와 기능이 출시되면 기존에 결정한 아키텍처 관련 사항을 검토하여 비용 측면에서 여전히 가장 효율적인 결정인지 확인하는 것이 좋습니다.

정기적으로 배포를 검토할 때 최신 서비스가 비용을 절약하는 데 어떻게 도움이 될 수 있는지 평가합니다. 예를 들어 Amazon Aurora on RDS를 사용하면 관계형 데이터베이스의 비용을 줄일 수 있습니다. Lambda와 같은 서버리스를 사용하면 코드를 실행하기 위해 인스턴스를 운영하고 관리할 필요가 없습니다.

리소스

비용 최적화 관련 AWS 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하십시오.

설명서

- AWS 설명서

백서

- 비용 최적화 원칙

지속 가능성

지속 가능성 원칙은 환경 영향, 특히 에너지 소비 및 효율성에 중점을 두고 있는데, 이는 건축가가 자원 사용을 줄이기 위한 직접적인 조치를 알아낼 수 있는 중요한 수단이기 때문입니다. 구현 방법에 대한 권장 가이드는 지속 가능성 원칙 백서에 나와 있습니다.

주제

- 설계 원칙
- 정의
- 모범 사례

설계 원칙

클라우드에는 6가지 지속 가능성 설계 원칙이 있습니다.

- 영향 이해: 클라우드 워크로드의 영향을 측정하고 워크로드의 향후 영향을 모델링합니다. 고객의 제품 사용으로 인한 영향과 최종 폐기 및 사용 중지로 인한 영향을 포함하여 영향의 모든 원인을 포함합니다. 작업 단위당 필요한 리소스 및 배출량을 검토하여 생산량을 클라우드 워크로드의 총 영향과 비교합니다. 이 데이터를 사용하여 핵심 성과 지표(KPI)를 설정하고, 영향을 줄이면서 생산성을 개선하는 방법을 평가하며, 시간 경과에 따른 제안된 변경의 영향을 예측할 수 있습니다.
- 지속 가능성 목표 수립: 각 클라우드 워크로드에 대해 트랜잭션당 필요한 컴퓨팅 및 스토리지 리소스의 절감과 같은 장기적인 지속 가능성 목표를 설정합니다. 기존 워크로드에 대한 지속 가능성 개선의 투자 수익률을 모델링하고 소유자에게 지속 가능성 목표에 투자하는 데 필요한 리소스를 제공합니다. 성장을 계획하고 워크로드를 설계하여 성장으로 인해 사용자당 또는 트랜잭션당 등 적절한 단위

에 대해 측정된 영향 강도를 줄입니다. 목표를 통해 비즈니스 또는 조직의 보다 광범위한 지속 가능성 목표를 지원하고, 회귀를 식별하며, 잠재적 개선 영역의 우선 순위를 지정할 수 있습니다.

- 활용률 극대화: 워크로드 크기를 적절하게 조정하고 효율적인 설계를 구현하여 높은 활용률을 보장하고 기본 하드웨어의 에너지 효율성을 극대화합니다. 호스트당 기준 전력 소비로 인해 30% 활용률로 실행되는 호스트 두 개는 60%로 실행되는 호스트 하나보다 효율성이 떨어집니다. 동시에 유휴 리소스, 프로세싱 및 스토리지를 없애거나 최소화하여 워크로드에 전력을 공급하는 데 필요한 총 에너지를 줄입니다.
- 보다 효율적인 최신 하드웨어와 소프트웨어 제품 및 서비스 예측 및 도입: 파트너와 공급업체가 업스트림 개선을 통해 클라우드 워크로드의 영향을 줄일 수 있도록 지원합니다. 새롭고 더 효율적인 하드웨어와 소프트웨어 제품 및 서비스를 지속적으로 모니터링하고 평가합니다. 새롭고 효율적인 기술을 신속하게 도입할 수 있도록 유연성을 고려하여 설계합니다.
- 관리형 서비스 사용: 광범위한 고객 기반에서 서비스를 공유하면 리소스 활용률을 극대화하여 클라우드 워크로드를 지원하는 데 필요한 인프라의 양을 줄일 수 있습니다. 예를 들어, 고객은 워크로드를 AWS 클라우드로 마이그레이션하고 AWS가 대규모로 운영하며 효율적인 운영을 책임지는 서버리스 컨테이너용 AWS Fargate 등 관리형 서비스를 채택하여 전력 및 네트워킹과 같은 일반적인 데이터 센터 구성 요소의 영향을 공유할 수 있습니다. Amazon S3 수명 주기 구성 또는 Amazon EC2 Auto Scaling을 사용하여 자주 액세스하지 않는 데이터를 콜드 스토리지로 자동 이동함으로써 수요에 맞게 용량을 조정하는 등 영향을 최소화할 수 있는 관리형 서비스를 사용합니다.
- 클라우드 워크로드의 다운스트림 영향 감소: 서비스를 사용하는 데 필요한 에너지 또는 리소스의 양을 줄입니다. 고객이 서비스를 사용하기 위해 디바이스를 업그레이드할 필요가 없도록 하거나 필요성을 줄입니다. Device Farm을 사용하여 테스트함으로써 예상되는 영향을 파악하고 고객과의 테스트를 통해 서비스 사용으로 인한 실제 영향을 이해합니다.

정의

클라우드의 지속 가능성에는 6가지 모범 사례 영역이 있습니다.

- 리전 선택
- 사용자 행동 패턴
- 소프트웨어 및 아키텍처 패턴
- 데이터 패턴
- 하드웨어 패턴
- 개발 및 배포 프로세스

클라우드에서의 지속 가능성이란 주로 프로비저닝된 리소스의 이점을 극대화하고 필요한 총 리소스를 최소화하면서 워크로드의 모든 구성 요소에서 에너지 절감과 효율성에 초점을 맞춘 지속적인 노력을 말합니다. 이러한 노력은 초기에 효율적인 프로그래밍 언어를 선택하고, 현대적인 알고리즘을 채택하고, 능률적인 데이터 스토리지 기술을 사용하여 적절한 규모의 효과적인 컴퓨팅 인프라를 배포하고 고성능 최종 사용자 하드웨어 요구 사항을 최소화하는 등 다양하게 나타날 수 있습니다.

모범 사례

주제

리전 선택

비즈니스 요구 사항과 지속 가능성 목표를 기반으로 워크로드를 구현할 리전을 선택합니다.

다음은 지속 가능성 고려 사항에 중점을 둔 질문입니다. 지속 가능성 관련 질문 및 모범 사례 목록은 부록을 참조하십시오.)

SUS 1: 지속 가능성 목표를 지원하기 위한 리전은 어떻게 선택합니까?
Amazon 재생 에너지 프로젝트 근처의 리전 및 그리드의 탄소 집약도가 다른 위치(또는 리전)보다 낮은 리전을 선택합니다.

사용자 행동 패턴

사용자가 워크로드 및 기타 리소스를 사용하는 방식을 통해 지속 가능성 목표를 달성하기 위한 개선 사항을 식별할 수 있습니다. 사용자 로드에 지속적으로 일치하도록 인프라 크기를 조정하고 사용자를 지원하는 데 필요한 최소 리소스만 배포되도록 합니다. 고객 요구 사항에 맞게 서비스 수준을 조정합니

다. 사용자가 리소스를 소비하는 데 필요한 네트워크를 제한하도록 리소스를 배치합니다. 사용하지 않는 기존 자산을 제거합니다. 생성된 자산 중 사용되지 않는 자산을 식별하고 생성을 중지합니다. 팀원에게 지속 가능성에 미치는 영향을 최소화하면서 요구 사항을 지원하는 디바이스를 제공합니다.

다음은 지속 가능성 고려 사항에 중점을 둔 질문입니다.

SUS 2: 사용자 행동 패턴을 활용하여 지속 가능성 목표를 지원하려면 어떻게 해야 합니까?

사용자가 워크로드 및 기타 리소스를 사용하는 방식을 통해 지속 가능성 목표를 달성하기 위한 개선 사항을 식별할 수 있습니다. 사용자 로드에 지속적으로 일치하도록 인프라 크기를 조정하고 사용자를 지원하는 데 필요한 최소 리소스만 배포되도록 합니다. 고객 요구 사항에 맞게 서비스 수준을 조정합니다. 사용자가 리소스를 소비하는 데 필요한 네트워크를 제한하도록 리소스를 배치합니다. 사용하지 않는 기존 자산을 제거합니다. 생성된 자산 중 사용되지 않는 자산을 식별하고 생성을 중지합니다. 팀원에게 지속 가능성에 미치는 영향을 최소화하면서 요구 사항을 지원하는 디바이스를 제공합니다.

사용자 로드에 맞게 인프라 크기 조정: 활용률이 낮거나 없는 기간을 식별하고 리소스의 크기를 조정하여 초과 용량을 제거하고 효율성을 개선합니다.

SLA를 지속 가능성 목표에 맞추기: 가용성 또는 데이터 보존 기간과 같은 서비스 수준 계약(SLA)을 정의 및 업데이트하여 워크로드를 지원하는 동시에 비즈니스 요구 사항을 지속적으로 충족하는 데 필요한 리소스 수를 최소화합니다.

사용하지 않는 자산의 생성 및 유지 관리 배제: 애플리케이션 자산(사전 컴파일된 보고서, 데이터 집합, 정적 이미지 등)과 자산 액세스 패턴을 분석하여 중복성, 활용률 저하 및 잠재적 폐기 대상을 식별합니다. 생성된 자산을 중복 콘텐츠(예: 중복되거나 공통된 데이터 집합 및 출력이 포함된 월간 보고서)로 통합하여 출력을 복제할 때 소비되는 리소스를 제거합니다. 사용되지 않는 자산(예: 더 이상 판매되지 않는 제품 이미지)을 폐기하여 리소스를 확보하고 워크로드를 지원하는 데 사용되는 리소스 수를 줄입니다.

사용자 위치에 대한 워크로드의 지리적 배치 최적화: 네트워크 액세스 패턴을 분석하여 고객이 접속하는 지리적 위치를 식별합니다. 워크로드를 지원하는 데 필요한 총 네트워크 리소스를 줄이기 위해 네트워크 트래픽이 이동해야 하는 거리가 적은 리전 및 서비스를 선택합니다.

수행된 활동에 대한 팀원 리소스 최적화: 팀원에게 제공되는 리소스를 최적화하여 팀원에게 필요한 지원을 충분히 제공하면서도 지속 가능성에 미치는 영향을 최소화합니다. 예를 들어, 활용률이 낮은 고성능 단일 사용자 시스템 대신 활용률이 높은 공유 클라우드 데스크톱에서 렌더링 및 컴파일과 같은 복잡한 작업을 수행합니다.

소프트웨어 및 아키텍처 패턴

로드 평준화를 수행하고 배포된 리소스의 높은 활용률을 일관되게 유지하여 소비되는 리소스를 최소화하기 위한 패턴을 구현합니다. 구성 요소는 시간 경과에 따른 사용자 행동의 변화로 인해 사용 부족으로 인해 유휴 상태가 될 수 있습니다. 패턴과 아키텍처를 수정하여 활용률이 낮은 구성 요소를 통합함으로써 전체 활용률을 높입니다. 더 이상 필요하지 않은 구성 요소를 폐기합니다. 워크로드 구성 요소의 성능을 이해하고 리소스를 가장 많이 사용하는 구성 요소를 최적화합니다. 고객이 서비스에 액세스하고 패턴을 구현하는 데 사용하는 디바이스를 숙지하여 디바이스 업그레이드 필요성을 최소화합니다.

다음은 지속 가능성 고려 사항에 중점을 둔 질문입니다.

SUS 3: 소프트웨어 및 아키텍처 패턴을 활용하여 지속 가능성 목표를 지원하려면 어떻게 해야 합니까?

로드 평준화를 수행하고 배포된 리소스의 높은 활용률을 일관되게 유지하여 소비되는 리소스를 최소화하기 위한 패턴을 구현합니다. 구성 요소는 시간 경과에 따른 사용자 행동의 변화로 인해 사용 부족으로 인해 유휴 상태가 될 수 있습니다. 패턴과 아키텍처를 수정하여 활용률이 낮은 구성 요소를 통합함으로써 전체 활용률을 높입니다. 더 이상 필요하지 않은 구성 요소를 폐기합니다. 워크로드 구성 요소의 성능을 이해하고 리소스를 가장 많이 사용하는 구성 요소를 최적화합니다. 고객이 서비스에 액세스하고 패턴을 구현하는 데 사용하는 디바이스를 숙지하여 디바이스 업그레이드 필요성을 최소화합니다.

비동기식 및 예약된 작업을 위한 소프트웨어와 아키텍처 최적화: 효율적인 소프트웨어 설계 및 아키텍처를 사용하여 작업 단위당 필요한 평균 리소스를 최소화합니다. 구성 요소를 균일하게 활용하여 작업 간에 유휴 상태인 리소스를 줄이고 로드 급증의 영향을 최소화하는 메커니즘을 구현합니다.

사용 빈도가 낮거나 전혀 없는 워크로드 구성 요소 제거 또는 리팩터링: 워크로드 활동을 모니터링하여 시간 경과에 따른 개별 구성 요소의 활용률 변화를 파악합니다. 사용되지 않아 더 이상 필요하지 않은 구성 요소와 활용률이 낮은 구성 요소를 리팩터링하여 낭비되는 리소스를 제한합니다.

가장 많은 시간 또는 리소스를 소모하는 코드 영역 최적화: 워크로드 활동을 모니터링하여 가장 많은 리소스를 사용하는 애플리케이션 구성 요소를 식별합니다. 이러한 구성 요소 내에서 실행되는 코드를 최적화하여 성능을 극대화하면서 리소스 사용을 최소화합니다.

고객 디바이스 및 장비에 대한 영향 최소화: 고객이 서비스를 사용하기 위해 이용하는 디바이스와 장비, 예상 수명 주기, 이러한 구성 요소 교체가 재정 및 지속 가능성에 미치는 영향을 이해합니다. 소프트

웨어 패턴 및 아키텍처를 구현하여 고객이 디바이스를 교체하고 장비를 업그레이드해야 하는 필요성을 최소화합니다. 예를 들어, 이전 하드웨어 및 운영 체제 버전과 역호환되는 코드를 사용하여 새로운 기능을 구현하거나 대상 디바이스의 저장 용량을 초과하지 않도록 페이로드 크기를 관리합니다.

데이터 액세스 및 저장 패턴을 가장 잘 지원하는 소프트웨어 패턴 및 아키텍처 사용: 데이터가 워크로드 내에서 사용되고, 사용자가 소비하고, 전송 및 저장되는 방식을 이해합니다. 데이터 처리 및 스토리지 요구 사항을 최소화하는 기술을 선택합니다.

데이터 패턴

로드 평준화를 수행하고 배포된 리소스의 높은 활용률을 일관되게 유지하여 소비되는 리소스를 최소화하기 위한 패턴을 구현합니다. 구성 요소는 시간 경과에 따른 사용자 행동의 변화로 인해 사용 부족으로 인해 유휴 상태가 될 수 있습니다. 패턴과 아키텍처를 수정하여 활용률이 낮은 구성 요소를 통합함으로써 전체 활용률을 높입니다. 더 이상 필요하지 않은 구성 요소를 폐기합니다. 워크로드 구성 요소의 성능을 이해하고 리소스를 가장 많이 사용하는 구성 요소를 최적화합니다. 고객이 서비스에 액세스하고 패턴을 구현하는 데 사용하는 디바이스를 숙지하여 디바이스 업그레이드 필요성을 최소화합니다.

다음은 지속 가능성 고려 사항에 중점을 둔 질문입니다.

SUS 4: 데이터 액세스 및 사용 패턴을 활용하여 지속 가능성 목표를 지원하려면 어떻게 해야 합니까?

데이터 관리 원칙을 구현하여 워크로드를 지원하는 데 필요한 프로비저닝된 스토리지와 이를 사용하는 데 필요한 리소스를 줄입니다. 데이터를 이해하고 데이터의 비즈니스 가치와 데이터 사용 방식을 가장 잘 지원하는 스토리지 기술과 구성을 사용합니다. 요구 사항이 감소하면 데이터를 더 효율적이고 성능이 낮은 스토리지로 수명 주기를 변경하고 더 이상 필요하지 않은 데이터는 삭제합니다.

데이터 분류 정책 구현: 데이터를 분류하여 비즈니스 성과에 미치는 분류의 영향을 파악합니다. 이 정보를 사용하여 데이터를 보다 에너지 효율적인 스토리지로 이동하거나 안전하게 삭제할 수 있는 시기를 결정합니다.

데이터 액세스 및 스토리지 패턴을 지원하는 기술 사용: 데이터 액세스 및 저장 방법을 가장 잘 지원하는 스토리지를 사용하여 워크로드를 지원하면서 프로비저닝된 리소스를 최소화합니다. 예를 들어, SSD(Solid State Device)는 자기 드라이브보다 에너지 집약적이므로 활성 데이터 사용 사례에만 사용해야 합니다. 자주 액세스하지 않는 데이터에는 에너지 효율적인 아카이빙 클래스 스토리지를 사용합니다.

수명 주기 정책을 사용하여 불필요한 데이터 삭제: 모든 데이터의 수명 주기를 관리하고 삭제 일정을 자동으로 적용하여 워크로드의 총 스토리지 요구 사항을 최소화합니다.

블록 스토리지에서 과다 프로비저닝 최소화: 프로비저닝된 스토리지의 총량을 최소화하려면 워크로드에 적합한 크기가 할당된 블록 스토리지를 생성하면 됩니다. 탄력적 볼륨을 사용하면 컴퓨팅 리소스에 연결된 스토리지의 크기를 조정할 필요 없이 데이터가 증가함에 따라 스토리지를 확장할 수 있습니다. 탄력적 볼륨을 정기적으로 검토하고 과다 프로비저닝된 볼륨을 현재 데이터 크기에 맞게 축소합니다.

불필요하거나 중복된 데이터 제거: 필요한 경우에만 데이터를 복제하여 총 스토리지 사용을 최소화합니다. 파일 및 블록 수준에서 데이터를 중복 제거하는 백업 기술을 사용합니다. SLA를 충족하는 데 필요한 경우를 제외하고 RAID(Redundant Array of Independent Drives) 구성의 사용을 제한합니다.

공유 파일 시스템 또는 객체 스토리지를 사용하여 공용 데이터에 액세스: 공유 스토리지와 단일 정보 소스를 도입하여 데이터 중복을 방지하고 워크로드의 총 스토리지 요구 사항을 줄입니다. 필요한 경우에만 공유 스토리지에서 데이터를 가져옵니다. 사용하지 않는 볼륨을 분리하여 리소스를 확보합니다. 네트워크 간 데이터 이동을 최소화합니다. 공유 스토리지를 사용하고 리전 데이터 스토어의 데이터에 액세스하여 워크로드의 데이터 이동을 지원하는 데 필요한 총 네트워킹 리소스를 최소화할 수 있습니다.

다시 생성하기 어려운 경우에만 데이터 백업: 스토리지 소비를 최소화하려면 비즈니스 가치가 있거나 규정 준수 요구 사항을 충족하는 데 필요한 데이터만 백업하면 됩니다. 백업 정책을 검토하고 복구 시나리오에서 가치를 제공하지 않는 임시 스토리지는 제외합니다.

하드웨어 패턴

하드웨어 관리 방식을 변경하여 지속 가능성에 미치는 워크로드의 영향을 줄일 수 있는 기회를 모색합니다. 프로비저닝 및 배포에 필요한 하드웨어의 양을 최소화하고 개별 워크로드에 가장 효율적인 하드웨어를 선택합니다.

다음은 지속 가능성 고려 사항에 중점을 둔 질문입니다.

SUS 5: 하드웨어 관리 및 사용 방식이 비즈니스의 지속 가능성 목표를 어떻게 지원하고 있습니까?

하드웨어 관리 방식을 변경하여 지속 가능성에 미치는 워크로드의 영향을 줄일 수 있는 기회를 모색합니다. 프로비저닝 및 배포에 필요한 하드웨어의 양을 최소화하고 개별 워크로드에 가장 효율적인 하드웨어를 선택합니다.

요구 사항을 충족하는 데 필요한 최소한의 하드웨어 사용: 클라우드의 기능을 사용하여 워크로드 구현을 자주 변경할 수 있습니다. 변화하는 요구 사항에 따라 배포된 구성 요소를 업데이트합니다.

영향이 가장 적은 인스턴스 유형 사용: 새로운 인스턴스 유형의 릴리스를 지속적으로 모니터링하고 기계 학습 훈련 및 추론, 비디오 트랜스코딩과 같은 특정 워크로드를 지원하도록 설계된 인스턴스 유형을 포함하여 에너지 효율성 개선의 이점을 활용합니다.

관리형 서비스 사용: 관리형 서비스는 배포된 하드웨어의 높은 평균 활용률과 지속 가능성 최적화를 유지하는 책임을 AWS로 이전합니다. 관리형 서비스를 사용하여 지속 가능성에 미치는 서비스의 영향을 서비스의 모든 테넌트에 분산하여 개인의 기여도를 낮춥니다.

GPU 사용 최적화: 그래픽 처리 장치(GPU)는 높은 전력 소비의 원인이 될 수 있으며 렌더링, 트랜스코딩, 기계 학습 훈련 및 모델링과 같은 많은 GPU 워크로드는 매우 가변적입니다. 필요한 시간 동안만 GPU 인스턴스를 실행하고 필요하지 않은 경우 자동화를 통해 GPU 인스턴스를 폐기하여 리소스 사용을 최소화합니다.

개발 및 배포 패턴

개발, 테스트 및 배포 방식을 변경하여 지속 가능성에 미치는 영향을 줄일 수 있는 기회를 모색합니다.

다음은 지속 가능성 고려 사항에 중점을 둔 질문입니다.

SUS 6: 개발 및 배포 프로세스가 비즈니스의 지속 가능성 목표를 어떻게 지원하고 있습니까?

개발, 테스트 및 배포 방식을 변경하여 지속 가능성에 미치는 영향을 줄일 수 있는 기회를 모색합니다.

지속 가능성 개선을 신속하게 도입할 수 있는 방법 채택: 잠재적 개선 사항을 프로덕션 환경에 배포하기 전에 테스트하고 검증합니다. 개선 사항으로 실현될 미래의 잠재적 이익을 계산할 때 테스트 비용을 고려합니다. 소규모 개선 사항을 제공할 수 있도록 저비용 테스트 방법을 개발합니다.

워크로드를 최신 상태로 유지: 최신 운영 체제, 라이브러리 및 애플리케이션을 통해 워크로드 효율성을 개선하고 보다 효율적인 기술을 더 쉽게 도입할 수 있습니다. 공급 업체가 자체적인 지속 가능성 목표를 충족할 수 있는 기능을 제공함에 따라, 최신 소프트웨어에는 워크로드의 지속 가능성에 미치는 영향을 보다 정확하게 측정하는 기능이 포함될 수도 있습니다.

구축 환경의 사용률 제고: 자동화와 코드형 인프라를 사용하여 필요 시 사전 프로덕션 환경을 가동하고 사용하지 않을 때는 해당 환경을 종료합니다. 일반적인 패턴은 개발 담당 팀원의 근무 시간과 일치하는 가용 기간을 예약하는 것입니다. 최대 절전 모드는 상태를 유지하고 필요할 때만 인스턴스를 빠르게 온라인으로 전환할 수 있는 유용한 도구입니다. 버스트 용량이 포함된 인스턴스 유형, 스팟 인스턴스, 탄력적 데이터베이스 서비스, 컨테이너 및 기타 기술을 사용하여 사용량에 따라 개발 및 테스트 용량을 조정합니다.

테스트에 관리형 Device Farm 사용: 관리형 Device Farm은 하드웨어 제조 및 리소스 사용이 지속 가능성에 미치는 영향을 여러 테넌트에 분산시킵니다. 관리형 Device Farm은 다양한 디바이스 유형을 제공하며, 사용 빈도가 낮은 오래된 하드웨어를 지원하고 불필요한 디바이스 업그레이드로 인한 고객의 지속 가능성에 미치는 영향을 방지할 수 있습니다.

리소스

지속 가능성 관련 AWS 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하십시오.

백서

- 지속 가능성 원칙

동영상

- The Climate Pledge(기후 협약)

검토 프로세스

아키텍처 검토는 깊이 있는 분석을 촉진할 수 있도록 결과에 대한 부담을 지지 않는 접근과 함께 일관적인 방식으로 수행되어야 합니다. 감사가 아니라 대화로 진행되는 가벼운 프로세스(며칠이 아닌 몇 시간)여야 합니다. 아키텍처를 검토하는 목적은 해결해야 할 영역이나 개선해야 할 부분을 파악하는 것입니다. 검토 결과는 워크로드를 사용하는 고객의 경험을 개선해야 하는 일련의 작업입니다.

"아키텍처" 섹션에서 설명한 것처럼 각 팀원에게 아키텍처의 품질에 대한 책임 부여를 원할 수 있습니다. 아키텍처를 구축하는 팀원은 공식 검토 회의를 개최하는 대신 Well-Architected 프레임워크를 사용하여 아키텍처를 계속 검토하는 것이 좋습니다. 지속적인 접근 방식을 사용하면 아키텍처가 발전함에 따라 팀원이 답변을 업데이트하고 기능을 전달할 때 아키텍처를 개선할 수 있습니다.

AWS Well-Architected Framework는 AWS가 시스템 및 서비스를 내부적으로 검토하는 방식에 맞춰 조정됩니다. 이는 아키텍처 접근 방식에 영향을 미치는 일련의 설계 원칙 및 사람들이 근본 원인 분석(RCA)에 있는 영역을 무시하지 않도록 보장하는 질문을 전제로 합니다. 내부 시스템, AWS 서비스 또는 고객과 관련하여 중대한 문제가 발생할 때마다 AWS는 RCA를 검토하여 사용 중인 검토 프로세스를 개선할 수 있는지 확인합니다.

검토는 제품 수명 주기의 주요 이정표에 적용되어야 하며, 단방향 방식을 방지하기 위해 설계 단계 초기에 적용되어야 합니다. 수명 주기에서 여러 번 검토를 적용해야 합니다. (대부분의 의사 결정은 번복할 수 있는 양방향 방식으로 이루어집니다. 이러한 결정 과정에서는 간단한 프로세스를 사용할 수 있습니다. 일방향 방식은 번복하기 어렵거나 불가능하기 때문에 확정하기 전에 검토를 더 많이 진행해야 합니다.) 프로덕션 환경에 적용한 후 워크로드를 가동하면 새로운 기능을 추가하고 기술 구현 방식을 변경함에 따라 계속 개선할 수 있습니다. 워크로드 아키텍처는 시간에 따라 변화합니다. 아키텍처 특성의 성능 저하를 방지하려면 개선 과정에서 재발을 막는 사례를 따라야 합니다. 중요한 아키텍처 변경을 수행할 때에는 Well-Architected 검토를 비롯한 일련의 재발을 방지할 수 있는 프로세스를 따라야 합니다.

일회성 스냅샷 또는 독립적인 측정 방식으로 검토를 진행하려면 적합한 모든 사람과 충분히 대화를 나눴는지 확인해야 합니다. 이러한 검토 과정에서 처음으로 팀이 구현한 내용을 분명하게 이해하게 되는 경우를 종종 경험하곤 합니다. 다른 팀의 워크로드를 검토할 때 효과적인 접근 방식은 대부분의 질문에 대한 답변을 수집할 수 있는 아키텍처에 대한 일련의 비공식 대화를 갖는 것입니다. 그런 다음 모호한 위험 또는 예측되는 위험 영역을 명확하게 파악하거나 자세히 알아볼 수 있는 한두 번의 회의를 진행할 수 있습니다.

회의를 쉽게 진행하기 위해 제안할 수 있는 몇 가지 항목은 다음과 같습니다.

- 화이트보드가 있는 회의실
- 다이어그램 또는 설계도 인쇄물

- 답변하기 위해 추가 조사가 필요한 질문 목록(예: “암호화를 활성화했는가?“)

검토를 마친 후에는 비즈니스 상황에 따라 우선 순위를 지정할 수 있는 문제 목록을 보유해야 합니다. 또한 이러한 문제가 팀의 일상적인 업무에 미치는 영향을 고려하고자 할 수도 있습니다. 이러한 문제를 초기에 해결하면 반복되는 문제를 해결하는 대신 비즈니스 가치를 창출하는 데 더 많은 시간을 할애할 수 있습니다. 문제를 해결할 때 검토 결과를 업데이트하면 아키텍처가 어떻게 개선되고 있는지 확인할 수 있습니다.

이러한 작업 이후 검토의 가치가 분명하게 나타나는 한편, 새롭게 접하는 팀은 처음에 반감을 가질 수 있다는 점을 확인하게 될 수 있습니다. 다음은 몇 가지 이의 제기 사례이며, 해당 팀에 검토의 이점을 설명함으로써 해결할 수 있습니다.

- “너무 바쁩니다!“(팀이 대규모 출시를 준비 중일 때 자주 나오는 말)
  - 대규모 출시를 준비하는 중이라면, 준비가 원활하게 진행되면 좋을 것입니다. 이러한 검토 과정을 진행하면 놓쳤을 수도 있는 문제를 이해할 수 있습니다.
  - 위험을 파악하고 기능 제공 로드맵에 따라 완화 계획을 수립하려면 제품 수명 주기 초반에 검토를 수행하는 것이 좋습니다.
- “이러한 결과에 대처할 시간이 없습니다!”(슈퍼볼 경기처럼 목표로 하고 있는 고정된 이벤트가 있을 때 자주 나오는 말)
  - 이 이벤트는 이동할 수 없습니다. 아키텍처에 대한 위험을 파악하지 않은 채 그대로 진행하고 싶으십니까? 이러한 모든 문제를 해결하지 못하더라도, 문제가 현실화되는 경우 해당 문제를 처리하는 데 도움이 되는 지침서가 있습니다.
- “다른 업체가 AWS의 솔루션 구현 방식에 대한 비밀을 알게 되길 바라진 않습니다!”
  - Well-Architected Framework의 질문을 팀에 제시하면 어떠한 질문에서도 상업적 또는 기술적 독점 정보가 드러나지 않음을 알 수 있습니다.

조직 내 팀과 여러 검토를 수행하면서 주제별 문제를 식별할 수 있습니다. 예를 들어, 일부 팀은 특정 기반 또는 주제에서 다양한 문제를 갖고 있다는 것을 확인할 수 있습니다. 전체적인 방식으로 모든 검토를 수행하고, 해당 주제별 문제를 해결하는 데 도움이 될 수 있는 메커니즘, 교육 또는 기본 엔지니어링 관련 정보를 식별하고자 할 수 있습니다.

결론

AWS Well-Architected Framework는 클라우드상의 안정적이고, 안전하며, 효율적이고, 경제적이면서 지속 가능한 시스템을 설계하고 운영하기 위한 6가지 원칙을 기반으로 한 설계 모범 사례를 제공합니다. 이 프레임워크에서는 기존 아키텍처 또는 제안된 아키텍처를 검토할 수 있는 몇 가지 질문과 각 원칙에 대한 AWS 모범 사례를 제시합니다. 아키텍처에 이 프레임워크를 사용하면 기능적 요구 사항에 초점을 둔 안정적이고 효율적인 시스템을 구축할 수 있습니다.

기여자

다음은 본 문서를 작성하는 데 도움을 준 개인 및 조직입니다.

- Brian Carlson, Well-Architected 운영 부문 담당자, Amazon Web Services
- Ben Potter, Well-Architected 보안 부문 담당자(Amazon Web Services)
- Seth Eliot, Well-Architected 신뢰성 부문 담당자, Amazon Web Services
- Eric Pullen, Amazon Web Services 선임 솔루션스 아키텍트
- Rodney Lester, 수석 솔루션스 아키텍트, Amazon Web Services
- Jon Steele, 선임 기술 계정 관리자, Amazon Web Services
- Max Ramsay, 수석 보안 솔루션스 아키텍트, Amazon Web Services
- Callum Hughes, 솔루션스 아키텍트, Amazon Web Services
- Aden Leirer, Well-Architected 콘텐츠 프로그램 관리자, Amazon Web Services

추가 자료

AWS 아키텍처 센터

AWS Cloud Compliance

AWS Well-Architected 파트너 프로그램

AWS Well-Architected Tool

AWS Well-Architected 홈페이지

운영 우수성 원칙 백서

보안 원칙 백서

안정성 원칙 백서

성능 효율성 원칙 백서

비용 최적화 원칙 백서

지속 가능성 원칙 백서

Amazon Builders' Library

문서 개정

이 백서의 업데이트에 대한 알림을 받으려면 RSS 피드를 구독하세요.

변경 사항	설명	날짜
백서 업데이트	모범 사례를 권장 가이드와 함께 업데이트하고 새로운 모범 사례를 추가했습니다. COST 11 질문을 추가했습니다.	April 10, 2023
마이너 업데이트	작업 수준에 대한 정의를 추가하고 부록에 모범 사례를 업데이트했습니다.	October 20, 2022
백서 업데이트	지속 가능성 원칙을 추가하고 링크를 업데이트했습니다.	December 2, 2021
주요 업데이트	프레임워크에 지속 가능성 원칙이 추가되었습니다.	November 20, 2021
마이너 업데이트	포용적이지 않은 표현을 삭제했습니다.	April 22, 2021
마이너 업데이트	여러 링크를 수정했습니다.	March 10, 2021
마이너 업데이트	문서 전반의 사소한 편집상 변경 사항입니다.	July 15, 2020
새 프레임워크 관련 업데이트	대부분의 질문과 답변을 검토하고 다시 작성합니다.	July 8, 2020
백서 업데이트	AWS Well-Architected Tool, AWS Well-Architected Labs 및 AWS Well-Architected 파트너 링크를 추가하고, 여러 언어 버전의 프레임워크를 지원하도록	July 1, 2019

	마이너한 사항 몇 가지를 수정했습니다.	
백서 업데이트	질문이 한 번에 하나의 주제에 대한 내용만 다루도록 대다수 질문과 대답을 검토하여 재작성했습니다. 이 과정에서 이전 질문 중 몇 개가 여러 질문으로 분할되었습니다. 워크로드, 구성 요소 등의 일반적인 용어 정의가 추가되었습니다. 기본 본문의 질문 표시가 변경되어 설명 텍스트가 포함되었습니다.	November 1, 2018
백서 업데이트	업데이트를 통해 질문 텍스트를 더 단순하게 작성하고 답변을 표준화했으며 가독성을 개선했습니다.	June 1, 2018
백서 업데이트	운영 우수성을 원칙 앞부분으로 옮겨 다시 작성했으며, 이에 따라 다른 원칙을 구성했습니다. AWS의 발전을 반영하기 위해 다른 원칙을 수정했습니다.	November 1, 2017
백서 업데이트	운영 우수성 원칙을 포함하도록 프레임워크를 업데이트하고, 중복을 줄이기 위해 다른 원칙을 개정 및 업데이트했으며, 수천 명의 고객과의 검토를 통해 얻게 된 내용을 통합했습니다.	November 1, 2016
마이너 업데이트	부록을 최신 Amazon CloudWatch Logs 정보로 업데이트했습니다.	November 1, 2015

최초 게시	AWS Well-Architected Framework를 게시했습니다.	October 1, 2015

부록: 질문 및 모범 사례

이 부록에는 AWS Well-Architected Framework의 모든 질문과 모범 사례가 요약되어 있습니다.

원칙

- 운영 우수성
- 보안
- 신뢰성
- 성능 효율성
- 비용 최적화
- 지속 가능성

운영 우수성

운영 우수성 원칙은 효과적인 개발 및 워크로드 실행을 지원하고, 운영에 대한 인사이트를 얻고, 지원 프로세스 및 절차를 지속적으로 개선하여 비즈니스 가치를 제공할 수 있는 능력을 포함합니다. 구현 방법에 대한 권장 가이드는 운영 우수성 원칙 백서에서 확인할 수 있습니다.

모범 사례 영역

- 조직
- 준비
- 운영
- 개선

조직

질문

- OPS 1 귀사의 운영 우선순위를 결정하는 요인은 무엇입니까?
- OPS 2 비즈니스 성과를 지원하기 위해 조직을 어떻게 구성합니까?
- OPS 3 조직 문화는 비즈니스 성과를 어떻게 지원합니까?

OPS 1 귀사의 운영 우선순위를 결정하는 요인은 무엇입니까?

모든 직원이 효율적인 업무 수행을 위한 역할과 리소스 우선 순위 설정을 위한 공동의 목표를 파악하고 있어야 합니다. 그러면 운영 개선 작업의 이점을 극대화할 수 있습니다.

모범 사례

- OPS01-BP01 외부 고객 요구 평가
- OPS01-BP02 내부 고객 요구 평가
- OPS01-BP03 거버넌스 요구 사항 평가
- OPS01-BP04 규정 준수 요구 사항 평가
- OPS01-BP05 위협 환경 평가
- OPS01-BP06 장단점 평가
- OPS01-BP07 이점 및 위험 관리

OPS01-BP01 외부 고객 요구 평가

실무 팀, 개발 팀, 운영 팀 등의 주요 이해관계자와 함께 외부 고객 요구 충족을 위해 주력할 영역을 결정합니다. 이렇게 하면 원하는 비즈니스 성과 달성에 필요한 운영 지원을 철저하게 파악할 수 있습니다.

일반적인 안티 패턴:

- 핵심 업무 시간 이외에 고객 지원을 하기로 결정했지만 과거 지원 요청 데이터를 검토하지 않았습니다. 이것이 고객에게 영향을 미치는지 여부는 알 수 없습니다.
- 새로운 기능을 개발 중이지만 고객이 원하는 기능이고 원하는 형태인지 확인하지 않았으며 요구 사항과 제공 방법을 확인하기 위한 실험도 진행하지 않습니다.

이 모범 사례 정립의 이점: 요구가 충족되는 경우 고객으로 남을 가능성이 훨씬 더 높습니다. 외부 고객 요구를 평가하고 이해하면 비즈니스 가치를 제공하기 위해 작업의 우선순위를 정하는 방법을 알 수 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 비즈니스 요구 사항 파악: 비즈니스의 성공은 실무 팀, 개발 팀, 운영 팀을 비롯한 모든 이해관계자가 서로 목표와 이해를 공유하면서 실현됩니다.

- 외부 고객의 비즈니스 목표, 요구 사항 및 우선순위 검토: 실무 팀, 개발 팀, 운영 팀 등의 주요 이해관계자와 함께 외부 고객의 목표, 요구 사항 및 우선순위를 논의합니다. 이렇게 하면 비즈니스 및 고객 성과 달성에 필요한 운영 지원을 철저하게 파악할 수 있습니다.
- 공유된 이해 관계 수립: 워크로드의 비즈니스 기능과 워크로드 작동 과정에서 각 팀의 역할을 비롯하여 이러한 요소가 내외부 고객의 공동 비즈니스 목표를 지원하는 방식에 대한 공유된 이해 관계를 정립합니다.

리소스

관련 문서:

- AWS Well-Architected Framework 개념 – 피드백 루프

OPS01-BP02 내부 고객 요구 평가

실무 팀, 개발 팀, 운영 팀 등의 주요 이해관계자와 함께 내부 고객 요구 충족을 위한 주력할 영역을 결정합니다. 이렇게 하면 비즈니스 성과 달성에 필요한 운영 지원을 철저하게 파악할 수 있습니다.

설정한 우선순위를 활용해 가장 영향력이 큰 개선 작업 부분부터 중점적으로 수행합니다. 팀 기술 개발, 워크로드 성능 개선, 비용 절감, 런북 자동화, 모니터링 기능 향상 등을 예로 들 수 있습니다. 요구 사항이 변경되면 우선 순위를 업데이트합니다.

일반적인 안티 패턴:

- 네트워크를 보다 쉽게 관리할 수 있도록 제품 팀의 IP 주소 할당을 상의하지 않고 변경하기로 결정했습니다. 이것이 제품 팀에 어떤 영향을 미칠지 알 수 없습니다.
- 새로운 개발 도구를 구현하고 있지만 내부 고객에게 이 도구가 필요한지 여부나 기존 사례와 호환되는지 여부를 확인하지 않았습니다.
- 새 모니터링 시스템을 구현하고 있지만 고려해야 할 모니터링 또는 보고 요구 사항이 있는지 내부 고객에게 문의하지 않았습니다.

이 모범 사례 수립의 이점: 내부 고객 요구를 평가하고 이해하면 비즈니스 가치를 제공하기 위해 작업의 우선순위를 정하는 방법을 알 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 비즈니스 요구 사항 파악: 비즈니스의 성공은 실무 팀, 개발 팀, 운영 팀을 비롯한 모든 이해관계자가 서로 목표와 이해를 공유하면서 실현됩니다.
  - 내부 고객의 비즈니스 목표, 요구 사항 및 우선순위 검토: 실무 팀, 개발 팀, 운영 팀 등의 주요 이해 관계자와 함께 내부 고객의 목표, 요구 사항 및 우선순위를 논의합니다. 이렇게 하면 비즈니스 및 고객 성과 달성에 필요한 운영 지원을 철저하게 파악할 수 있습니다.
  - 공유된 이해 관계 수립: 워크로드의 비즈니스 기능과 워크로드 작동 과정에서 각 팀의 역할을 비롯하여 이러한 요소가 내외부 고객의 공동 비즈니스 목표를 지원하는 방식에 대한 공유된 이해 관계를 정립합니다.

리소스

관련 문서:

- AWS Well-Architected Framework 개념 – 피드백 루프

OPS01-BP03 거버넌스 요구 사항 평가

거버넌스는 기업이 비즈니스 목표를 달성하기 위해 사용하는 정책, 규칙 또는 프레임워크의 집합입니다. 거버넌스 요구 사항은 조직 내에서 생성됩니다. 이러한 요구 사항은 선택한 기술 유형이나 워크로드 운영 방식에 영향을 미칠 수 있습니다. 워크로드에 조직 거버넌스 요구 사항을 통합합니다. 적합성은 거버넌스 요구 사항을 구현했음을 입증할 수 있는 역량입니다.

원하는 결과:

- 거버넌스 요구 사항은 워크로드의 아키텍처 설계 및 운영에 통합됩니다.
- 거버넌스 요구 사항을 준수했다는 증거를 제공할 수 있습니다.
- 거버넌스 요구 사항은 정기적으로 검토 및 업데이트됩니다.

일반적인 안티 패턴:

- 조직에서는 루트 계정에서 다중 인증을 사용해야 합니다. 이 요구 사항을 구현하지 못했으며 루트 계정이 손상되었습니다.
- 워크로드를 설계하는 동안 IT 부서에서 승인하지 않은 인스턴스 유형을 선택합니다. 워크로드를 시작할 수 없으므로 재설계를 수행해야 합니다.

- 재해 복구 계획을 마련해야 합니다. 재해 복구 계획을 마련하지 않아 워크로드에 오랜 중단이 발생합니다.
- 팀에서 새 인스턴스를 사용하려고 하지만, 이를 허용하도록 거버넌스 요구 사항이 업데이트되지 않았습니다.

이 모범 사례 확립의 이점:

- 다음과 같은 거버넌스 요구 사항은 대규모 조직 정책에 따라 워크로드를 조정합니다.
- 거버넌스 요구 사항은 조직의 업계 표준 및 모범 사례를 반영합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

이해관계자 및 거버넌스 조직과 협력하여 거버넌스 요구 사항을 파악합니다. 거버넌스 요구 사항을 워크로드에 포함합니다. 거버넌스 요구 사항을 준수했다는 증거를 제시할 수 있어야 합니다.

고객 사례

AnyCompany Retail의 클라우드 운영 팀은 조직 전체의 이해관계자와 협력하여 거버넌스 요구 사항을 개발합니다. 예를 들어, Amazon EC2 인스턴스에 대한 SSH 액세스를 금지합니다. 팀이 시스템에 액세스해야 하는 경우 AWS Systems Manager Session Manager를 사용해야 합니다. 클라우드 운영 팀은 새로운 서비스가 제공되면 거버넌스 요구 사항을 정기적으로 업데이트합니다.

구현 단계

1. 중앙 집중식 팀을 포함하여 워크로드의 이해관계자를 식별합니다.
2. 이해관계자와 협력하여 거버넌스 요구 사항을 파악합니다.
3. 목록을 생성했으면 개선 항목의 우선순위를 지정하고 워크로드에 구현하기 시작합니다.
    a. AWS Config와 같은 서비스를 사용하여 코드형 거버넌스를 생성하고 거버넌스 요구 사항이 준수되는지 확인합니다.
    b. AWS Organizations를 사용하는 경우 서비스 제어 정책을 활용하여 거버넌스 요구 사항을 구현할 수 있습니다.
4. 구현을 검증하는 문서를 제공합니다.

구현 계획의 작업 수준: 중간. 누락된 거버넌스 요구 사항을 구현하면 워크로드 재작업이 발생할 수 있습니다.

리소스

관련 모범 사례:

- OPS01-BP04 규정 준수 요구 사항 평가 - 규정 준수는 거버넌스와 유사하지만, 조직 외부에서 이루어집니다.

관련 문서:

- AWS Management and Governance Cloud Environment Guide(AWS 관리 및 거버넌스 클라우드 환경 가이드)
- 다중 계정 환경에서의 AWS Organizations 서비스 제어 정책 모범 사례
- Governance in the AWS 클라우드: The Right Balance Between Agility and Safety(AWS 클라우드의 거버넌스: 민첩성과 안전성 사이의 적절한 균형)
- 거버넌스, 위험 및 규정 준수(GRC)란 무엇입니까?

관련 동영상:

- AWS Management and Governance: Configuration, Compliance, and Audit - AWS Online Tech Talks(AWS 관리 및 거버넌스: 구성, 규정 준수 및 감사 - AWS Online Tech Talks)
- AWS re:Inforce 2019: Governance for the Cloud Age (DEM12-R1)(AWS re:Inforce 2019: 클라우드 시대를 위한 거버넌스(DEM12-R1))
- AWS re:Invent 2020: Achieve compliance as code using AWS Config(AWS re:Invent 2020: AWS Config를 사용하여 코드로 규정 준수 달성)
- AWS re:Invent 2020: Agile governance on AWS GovCloud (US)(AWS re:Invent 2020: AWS GovCloud(US)의 민첩한 거버넌스)

관련 예시:

- AWS Config 규정 준수 팩 샘플

관련 서비스:

- AWS Config
- AWS Organizations - 서비스 제어 정책

OPS01-BP04 규정 준수 요구 사항 평가

규제, 산업 및 내부 규정 준수 요구 사항은 조직의 우선순위를 정의하는 데 있어 중요한 동인입니다. 규정 준수 프레임워크로 인해 특정 기술이나 지리적 위치를 사용하지 못하게 될 수 있습니다. 외부 규정 준수 프레임워크가 확인되지 않은 경우 실사를 적용합니다. 규정 준수를 검증하는 감사 또는 보고서를 생성합니다.

제품이 특정 규정 준수 표준을 충족한다고 광고하는 경우 지속적인 규정 준수를 보장하는 내부 프로세스가 있어야 합니다. 규정 준수 표준의 예로는 PCI DSS, FedRAMP 및 HIPAA가 있습니다. 적용 가능한 규정 준수 표준은 솔루션이 저장하거나 전송하는 데이터 유형, 솔루션이 지원하는 지리적 리전 등 다양한 요인에 의해 결정됩니다.

원하는 결과:

- 규제, 산업 및 내부 규정 준수 요구 사항이 아키텍처 선택에 통합됩니다.
- 규정 준수를 검증하고 감사 보고서를 생성할 수 있습니다.

일반적인 안티 패턴:

- 워크로드의 일부는 결제 카드 산업 데이터 보안 표준(PCI-DSS) 프레임워크에 속하지만, 워크로드는 신용 카드 데이터를 암호화되지 않은 상태로 저장합니다.
- 소프트웨어 개발자와 아키텍트는 조직이 준수해야 하는 규정 준수 프레임워크를 알지 못합니다.
- 연간 시스템 및 조직 제어(SOC2) 유형 II 감사가 곧 시작되는데 제어 수단이 마련되어 있는지 확인할 수 없습니다.

이 모범 사례 확립의 이점:

- 워크로드에 적용되는 규정 준수 요구 사항을 평가하고 이해하면 비즈니스 가치를 제공하기 위한 작업의 우선순위를 정하는 방법을 알 수 있습니다.
- 규정 준수 프레임워크와 일치하는 올바른 위치와 기술을 선택할 수 있습니다.
- 감사 기능에 적합하도록 워크로드를 설계하면 규정 준수 프레임워크를 준수하고 있음을 입증할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

이 모범 사례를 구현하면 아키텍처 설계 프로세스에 규정 준수 요구 사항을 통합할 수 있습니다. 팀원은 필요한 규정 준수 프레임워크를 알고 있습니다. 프레임워크에 따라 규정 준수를 검증합니다.

고객 사례

AnyCompany Retail에서는 고객의 신용 카드 정보를 저장합니다. 카드 스토리지 팀의 개발자들은 PCI-DSS 프레임워크를 준수해야 한다는 점을 알고 있습니다. 이들은 신용 카드 정보가 PCI-DSS 프레임워크에 따라 안전하게 저장되고 액세스되는지 확인하기 위한 조치를 취했습니다. 그리고 매년 보안 팀과 협력하여 규정 준수를 검증합니다.

구현 단계

1. 보안 및 거버넌스 팀과 협력하여 워크로드가 준수해야 하는 산업, 규제 또는 내부 규정 준수 프레임워크를 결정합니다. 규정 준수 프레임워크를 워크로드에 통합합니다.
   a. AWS Compute Optimizer, AWS Security Hub와 같은 서비스를 통해 AWS 리소스의 지속적인 규정 준수를 검증합니다.
2. 규정 준수 요구 사항에 대해 교육하여 팀원이 워크로드를 적절하게 운영하고 발전시킬 수 있도록 지원합니다. 규정 준수 요구 사항은 아키텍처 및 기술 선택에 포함되어야 합니다.
3. 규정 준수 프레임워크에 따라 감사 또는 규정 준수 보고서를 생성해야 할 수도 있습니다. 조직과 협력하여 이 프로세스를 최대한 자동화하세요.
   a. AWS Audit Manager와 같은 서비스를 사용하여 규정 준수를 검증하고 감사 보고서를 생성합니다.
   b. AWS Artifact를 사용하여 AWS 보안 및 규정 준수 문서를 다운로드할 수 있습니다.

구현 계획의 작업 수준: 중간. 규정 준수 프레임워크를 구현하기란 어려울 수 있습니다. 감사 보고서 또는 규정 준수 문서를 생성하면 복잡성이 가중됩니다.

리소스

관련 모범 사례:

- SEC01-BP03 제어 목표 파악 및 검증 - 보안 제어 목표는 전반적인 규정 준수의 중요한 부분입니다.
- SEC01-BP06 파이프라인에서 보안 제어의 테스트 및 검증 자동화 - 파이프라인의 일부로 보안 제어를 검증합니다. 또한, 새 변경 사항에 대한 규정 준수 문서를 생성할 수 있습니다.
- SEC07-BP02 데이터 보호 제어 정의 - 많은 규정 준수 프레임워크는 데이터 처리 및 스토리지 정책에 기반을 두고 있습니다.

- SEC10-BP03 포렌식 역량 확보 - 포렌식 기능을 종종 규정 준수 감사에 사용할 수 있습니다.

관련 문서:

- AWS 규정 준수 센터
- AWS 규정 준수 리소스
- AWS 위험 및 규정 준수 백서
- AWS 공동 책임 모델
- 규정 준수 프로그램 제공 AWS 범위 내 서비스

관련 동영상:

- AWS re:Invent 2020: Achieve compliance as code using AWS Compute Optimizer(AWS re:Invent 2020: AWS Config를 사용하여 코드로 규정 준수 달성)
- AWS re:Invent 2021 - Cloud compliance, assurance, and auditing(AWS re:Invent 2021 - 클라우드 규정 준수, 보증 및 감사)
- AWS Summit ATL 2022 - Implementing compliance, assurance, and auditing on AWS (COP202)(AWS Summit ATL 2022 - AWS에서 규정 준수, 보증 및 감사 구현(COP202))

관련 예시:

- PCI DSS 및 AWS의 AWS 기초 보안 모범 사례

관련 서비스:

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

OPS01-BP05 위협 환경 평가

비즈니스에 대한 위협 요소(예: 경쟁, 비즈니스상의 위험 및 법적 책임, 운영상의 위험, 정보 보안 위협)를 평가하고 위험 목록에서 최신 정보를 관리합니다. 주력할 영역을 결정할 때 위험의 영향을 포함시킵니다.

유효 Well-Architected 프레임워크 에서는 학습, 평가 및 개선을 강조합니다. 아키텍처를 평가하고 시간에 따라 확장 가능한 설계를 구현하는 일관된 접근 방식을 제공합니다. AWS에서 선보이는 AWS Well-Architected Tool 은 개발 전의 접근 방식, 프로덕션 환경에 적용하기 전의 워크로드 상태, 프로덕션 환경에서의 워크로드 상태를 검토합니다. 이를 최신 AWS 아키텍처 모범 사례와 비교하고, 워크로드의 전반적인 상태를 모니터링하며, 잠재적 위험에 대한 인사이트를 얻을 수 있습니다.

AWS 고객은 AWS 모범 사례를 기준으로 하여 아키텍처를 평가하는 미션 크리티컬 워크로드에 대한 안내형 AWS Well-Architected Review 서비스를 이용할 수 있습니다. Enterprise Support 고객이라면 Operations Review를 이용할 수도 있습니다.

여러 팀의 구성원이 이러한 검토에 참여하면 워크로드 자체, 그리고 팀에서 각 역할을 맡은 구성원이 효율적인 워크로드 처리에 기여할 수 있는 방법을 공통된 방식으로 파악할 수 있습니다. 검토를 통해 확인된 요구 사항을 참조하여 우선 순위를 결정할 수 있습니다.

AWS Trusted Advisor 는 우선순위 결정에 도움이 될 수 있는 최적화 방안을 알려주는 핵심 검사 세트에 액세스할 수 있는 도구입니다. Business 및 Enterprise Support 고객에게는 우선순위를 더욱 자세히 결정하는 데 사용할 수 있는 추가 검사 기능이 제공됩니다. 이러한 기능을 사용하면 보안, 안정성, 성능 및 비용 최적화 영역을 중점적으로 확인할 수 있습니다.

일반적인 안티 패턴:

- 제품에서 이전 버전의 소프트웨어 라이브러리를 사용하고 있습니다. 워크로드에 의도하지 않은 영향을 줄 수 있는 문제에 대한 라이브러리의 보안 업데이트에 대해 모릅니다.
- 경쟁업체가 제품에 대한 많은 고객 불만 사항을 해결하는 제품 버전을 출시했습니다. 이러한 알려진 문제 해결에 우선순위를 지정하지 않았습니다.
- 규제 기관이 법률 규정 준수 요구 사항을 준수하지 않는 귀사와 같은 회사를 추적하고 있습니다. 미해결 규정 준수 요구 사항 해결에 우선순위를 지정하지 않았습니다.

이 모범 사례 수립의 이점: 조직 및 워크로드에 대한 위협을 식별하고 이해하면 해결할 위협, 우선순위 및 이를 수행하는 데 필요한 리소스를 결정하는 데 도움이 됩니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 위협 환경 평가: 주력할 영역을 결정할 때 영향력을 고려할 수 있도록 업무상의 위협 요소(예: 경쟁, 업무상의 위험/책임, 운영상의 위험, 정보 보안 위협)를 평가합니다.
  - 최신 AWS 보안 공지

- AWS Trusted Advisor
- 위협 모델 유지 관리: 잠재적 위협, 계획되어 실행 중인 완화 조치 및 각 우선순위를 식별하는 위협 모델을 수립하고 유지 관리합니다. 인시던트로 나타나는 위협의 가능성, 해당 인시던트로부터 복구하는 비용 및 예상되는 피해, 이러한 인시던트를 방지하는 비용을 검토합니다. 위협 모델의 내용이 변경됨에 따라 우선순위를 수정합니다.

리소스

관련 문서:

- AWS 클라우드 규정 준수
- 최신 AWS 보안 공지
- AWS Trusted Advisor

OPS01-BP06 장단점 평가

주력할 영역을 결정하거나 수행할 조치를 선택할 때 정보를 토대로 결정을 내릴 수 있도록 상충하는 이해 관계나 대안 사이의 장단점을 평가합니다. 예를 들어, 비용 최적화보다 새로운 기능의 시장 출시를 앞당기는 데 더 역점을 둘 수 있습니다. 아니면 데이터 유형에 최적화된 데이터베이스로 마이그레이션하고 애플리케이션을 업데이트하는 대신, 시스템 마이그레이션 작업을 간소화하기 위해 비관계형 데이터용 솔루션으로 관계형 데이터베이스를 선택할 수도 있습니다.

AWS를 활용하면 선택한 방식이 워크로드에 미치는 영향을 효과적으로 파악하도록 팀에 AWS 및 해당 서비스 관련 정보를 제공할 수 있습니다. 팀에 관련 정보를 제공하는 데 AWS Support (AWS 지식 센터, AWS 토론 포럼및 AWS Support 센터) 및 AWS 설명서 를 이용해야 합니다. AWS 관련 문의 사항에 대해 지원을 받으려면 AWS Support 센터를 통해 AWS Support에 지원을 요청하십시오.

AWS는 다음에서 AWS의 운영을 통해 학습한 모범 사례와 패턴을 공유합니다. Amazon Builders' Library참조. 다음에서도 기타 여러 가지 유용한 정보를 확인할 수 있습니다. AWS 블로그 및 공식 AWS 팟캐스트.

일반적인 안티 패턴:

- 관계형 데이터베이스를 사용하여 시계열 및 비관계형 데이터를 관리하고 있습니다. 사용 중인 데이터 형식을 지원하도록 최적화된 데이터베이스 옵션이 있지만 솔루션 간의 장단점을 평가하지 않아 이점을 모릅니다.
- 투자자는 PCI DSS(Payment Card Industry Data Security Standards)를 준수함을 입증할 것을 요청합니다. 요청을 충족하는 것과 현재 개발 작업을 계속하는 것 사이의 장단점을 고려하지 않습니다.

대신 규정 준수를 입증하지 않고 개발 작업을 계속 진행합니다. 투자자는 플랫폼 보안과 투자에 대한 우려 사항으로 인해 회사의 지원을 중단합니다.

이 모범 사례 수립의 이점: 선택에 따른 영향과 결과를 이해하면 옵션의 우선순위를 정할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 장단점 평가: 주력할 영역을 결정할 때 정보를 토대로 적절한 결정을 내릴 수 있도록 상충하는 이해 관계의 장단점을 평가합니다. 예를 들어 새로운 기능의 출시 기간을 단축하는 것이 비용 최적화보다 우선시될 수 있습니다.
- AWS를 활용하면 선택한 방식이 워크로드에 미치는 영향을 효과적으로 파악하도록 팀에 AWS 및 해당 서비스 관련 정보를 제공할 수 있습니다. AWS Support(AWS 지식 센터, AWS 토론 포럼, AWS Support 센터) 및 AWS 설명서에 나와 있는 리소스를 사용해서 팀을 교육해야 합니다. AWS 관련 문의 사항에 대해 지원을 받으려면 AWS Support 센터를 통해 AWS Support에 지원을 요청하십시오.
- 또한 AWS는 AWS의 운영을 통해 학습한 모범 사례와 패턴을 Amazon Builders' Library에서 공유합니다. AWS 블로그 및 공식 AWS 팟캐스트에서도 기타 여러 가지 유용한 정보를 확인할 수 있습니다.

리소스

관련 문서:

- AWS 블로그
- AWS 클라우드 규정 준수
- AWS 토론 포럼
- AWS 설명서
- AWS 지식 센터
- AWS Support
- AWS Support 센터
- Amazon Builders' Library
- 공식 AWS 팟캐스트

OPS01-BP07 이점 및 위험 관리

주력할 영역을 결정할 때 정보를 토대로 적절한 결정을 내릴 수 있도록 이점과 위험을 관리합니다. 예를 들어 새 기능을 고객에게 제공하기 위해 해결되지 않은 문제가 남아 있는 워크로드를 배포하는 것이 이득일 수 있습니다. 관련 위험을 완화할 수도 있겠지만, 위험을 감수할 수 없는 경우에는 위험을 해결하기 위한 조치를 취해야 합니다.

특정 시점에 우선순위 중 일부를 중점적으로 처리해야 할 수도 있습니다. 필요한 기능을 개발하고 위험을 관리하려면 워크로드 우선순위를 장기적으로 적절하게 절충해야 합니다. 요구 사항이 변경되면 우선순위를 업데이트합니다.

일반적인 안티 패턴:

- 한 개발자가 인터넷에서 찾은 필요한 모든 것을 수행하는 라이브러리를 포함하기로 했습니다. 출처를 알 수 없는 이 라이브러리의 도입에 따른 위험을 평가하지 않았으며 취약성 또는 악성 코드가 포함되어 있는지 알 수 없습니다.
- 기존 문제를 해결하는 대신 새 기능을 개발하고 배포하기로 결정했습니다. 배포 전까지 이 기능에 남아 있는 문제의 위험을 평가하지 않았으며 고객에게 어떤 영향이 있을지 모릅니다.
- 규정 준수 팀의 불특정 우려 때문에 고객이 자주 요청하는 기능을 배포하지 않기로 결정했습니다.

이 모범 사례 수립의 이점: 선택 시 얻을 수 있는 이점을 파악하고 조직의 위험을 인식하면 정보에 입각한 결정을 내릴 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 이점 및 위험 관리: 결정으로 인해 누릴 수 있는 이점과 수반되는 위험의 균형을 잡습니다.
  - 이점 파악: 비즈니스 목표, 필요 사항 및 우선순위에 따른 이점을 파악합니다. 출시 리드타임, 보안, 신뢰성, 성능 및 비용이 이에 해당합니다.
  - 위험 요소 파악: 비즈니스 목표, 필요 사항 및 우선순위에 따른 위험 요소를 파악합니다. 출시 리드타임, 보안, 신뢰성, 성능 및 비용이 이에 해당합니다.
  - 위험 대비 이점 평가 및 정보에 입각한 의사 결정: 실무/개발/운영 팀을 비롯한 주요 이해관계자의 목표, 필요 사항 및 우선순위를 기준으로 하여 이점과 위험의 영향을 파악합니다. 위험 발생 가능성 및 해당 위험이 주는 영향의 비용 대비 이점의 가치를 평가합니다. 예를 들어, 안정성보다 시장 진입 속도를 강조하면 경쟁 우위를 제공할 수 있습니다. 하지만 안정성 문제가 있는 경우 가동 시간이 단축될 수 있습니다.

OPS 2 비즈니스 성과를 지원하기 위해 조직을 어떻게 구성합니까?

팀은 비즈니스 성과를 달성하기 위해 맡은 역할을 파악해야 합니다. 그리고 다른 팀의 성공을 위해 자신의 팀이 해야 할 역할과 해당 팀이 해야 할 역할을 파악하고, 목표를 공유해야 합니다. 맡은 책임, 소유권, 의사 결정 방식 및 의사 결정권자를 파악하면 역량을 집중하고 팀의 이점을 극대화할 수 있습니다.

모범 사례

- OPS02-BP01 리소스 소유자 식별
- OPS02-BP02 프로세스 및 절차의 소유자 식별
- OPS02-BP03 운영 활동에서 성능을 담당하는 소유자 식별
- OPS02-BP04 팀원이 담당해야 하는 업무 파악
- OPS02-BP05 책임과 소유권을 식별하는 메커니즘
- OPS02-BP06 추가, 변경 및 예외를 요청하는 메커니즘
- OPS02-BP07 미리 정의되었거나 협상된 팀 간 책임

OPS02-BP01 리소스 소유자 식별

워크로드의 리소스에는 변경 제어, 문제 해결 및 기타 기능에 대한 소유자가 식별되어 있어야 합니다. 소유자는 워크로드, 계정, 인프라, 플랫폼 및 애플리케이션에 대해 할당됩니다. 소유권은 중앙 레지스터 또는 리소스에 첨부된 메타데이터와 같은 도구를 사용하여 기록됩니다. 구성 요소의 비즈니스 가치는 구성 요소에 적용되는 프로세스와 절차를 알려 줍니다.

원하는 결과:

- 리소스는 메타데이터 또는 중앙 레지스터를 사용하여 소유자를 식별했습니다.
- 팀원은 누가 리소스를 소유하는지 식별할 수 있습니다.
- 계정에는 가능한 경우 단일 소유자가 있습니다.

일반적인 안티 패턴:

- AWS 계정의 대체 연락처가 입력되지 않았습니다.
- 리소스에는 소유한 팀을 식별하는 태그가 없습니다.
- 이메일 매핑이 없는 ITSM 대기열이 있습니다.
- 두 팀이 중요한 인프라의 소유권을 중복으로 보유하고 있습니다.

이 모범 사례 확립의 이점:

- 소유권이 할당되어 리소스에 대한 변경 제어가 간단해집니다.
- 문제를 해결할 때 올바른 소유자를 참여시킬 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

환경의 리소스 사용 사례에 대한 소유권 의미를 정의합니다. 소유권이란 리소스의 변경 사항을 감독하거나, 문제 해결 중에 리소스를 지원하거나, 재정적으로 책임을 지는 사람을 의미할 수 있습니다. 이름, 연락처 정보, 조직 및 팀을 포함하여 리소스 소유자를 지정하고 기록합니다.

고객 사례

AnyCompany Retail은 소유권을 리소스에 대한 변경 및 지원 권한이 있는 팀 또는 개인으로 정의합니다. 이들은 AWS Organizations를 사용하여 AWS 계정을 관리합니다. 대체 계정 연락처는 그룹 받은 편지함을 사용하여 구성되고 있습니다. 각 ITSM 대기열은 이메일 별칭에 매핑됩니다. 태그는 AWS 리소스를 소유하는 사용자를 식별합니다. 다른 플랫폼 및 인프라의 경우 소유권 및 연락처 정보를 식별하는 Wiki 페이지가 있습니다.

구현 단계

1. 먼저 조직의 소유권을 정의합니다. 소유권은 리소스에 대한 위험을 부담하거나, 리소스를 변경하거나, 문제 해결 시 리소스를 지원하는 사람을 의미할 수 있습니다. 소유권은 리소스의 재정적 또는 관리적 소유권을 의미할 수도 있습니다.
2. AWS Organizations를 사용하여 계정을 관리하세요 계정의 대체 연락처를 중앙에서 관리할 수 있습니다.
   a. 연락처 정보에 회사 소유의 이메일 주소와 전화번호를 사용하면 상급자가 조직을 퇴사한 경우에도 액세스할 수 있습니다. 예를 들어 청구, 운영 및 보안에 대한 별도의 이메일 배포 목록을 생성하고, 각 활성 AWS 계정에서 이를 청구, 보안 및 운영 연락처로 구성합니다. 누군가 휴가 중이거나 역할이 변경되거나 퇴사하더라도 여러 사람이 AWS 알림을 수신하고 응답할 수 있게 됩니다.
   b. 계정을 AWS Organizations에서 관리하지 않는 경우 대체 계정 연락처가 AWS에서 필요한 경우 적절한 담당자와 연락할 수 있도록 도와줍니다. 계정의 대체 연락처가 개인이 아닌 그룹을 지정하도록 구성합니다.
3. 태그를 사용하여 AWS 리소스 소유자를 식별합니다. 소유자와 해당 연락처 정보를 별도의 태그로 지정할 수 있습니다.
   a. AWS Config 규칙을 사용하여 리소스에 필요한 소유권 태그가 있는지 확인할 수 있습니다.

   b. 조직의 태그 지정 전략을 개발하는 방법에 대한 자세한 지침은 AWS 태그 지정 모범 사례 백서를 참조하세요.
4. 다른 리소스, 플랫폼 및 인프라의 경우 소유권을 식별하는 문서를 생성합니다. 모든 팀원이 여기에 액세스할 수 있어야 합니다.

구현 계획의 작업 수준: 낮음. 계정 연락처 정보 및 태그를 활용하여 AWS 리소스 소유권을 할당합니다. 다른 리소스의 경우 Wiki의 표와 같이 간단한 방식을 사용하여 소유권 및 연락처 정보를 기록하거나 ITSM 도구를 사용하여 소유권을 매핑할 수 있습니다.

리소스

관련 모범 사례:

- OPS02-BP02 프로세스 및 절차의 소유자 식별 - 리소스를 지원하는 프로세스 및 절차는 리소스 소유권에 따라 달라집니다.
- OPS02-BP04 팀원이 담당해야 하는 업무 파악 - 팀원은 본인이 어떤 리소스를 소유하고 있는지 이해해야 합니다.
- OPS02-BP05 책임과 소유권을 식별하는 메커니즘 - 소유권은 태그 또는 계정 연락처와 같은 메커니즘을 사용하여 검색할 수 있어야 합니다.

관련 문서:

- AWS Account Management - Updating contact information(AWS 계정 관리 -연락처 정보 업데이트)
- AWS Config Rules - required-tags(AWS Config 규칙 - required-tags)
- AWS Organizations - 조직의 대체 연락처 업데이트
- AWS 태그 지정 모범 사례 백서

관련 예시:

- AWS Config Rules - Amazon EC2 with required tags and valid values(AWS Config 규칙 - 필수 태그 및 유효 값과 EC2)

관련 서비스:

- AWS Config
- AWS Organizations

OPS02-BP02 프로세스 및 절차의 소유자 식별

개별 프로세스와 절차의 정의에 대한 소유권이 있는 사람, 그러한 특정 프로세스와 절차가 사용되는 이유, 그리고 소유권이 존재하는 이유를 파악합니다. 특정 프로세스 및 절차가 사용되는 이유를 이해하면 개선 기회를 파악할 수 있습니다.

이 모범 사례 수립의 이점: 소유권을 파악하면 누가 개선 사항 승인 또는 구현을 수행하거나 둘 다 수행할 수 있는지 알 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 프로세스 및 절차에서 정의를 담당하는 소유자 식별: 환경에서 사용되는 프로세스 및 절차와 정의를 담당하는 개인 또는 팀을 캡처합니다.
  - 프로세스 및 절차 식별: 워크로드 지원을 위해 수행되는 운영 활동을 식별합니다. 검색 가능한 위치에 이러한 활동을 문서화합니다.
  - 프로세스 또는 절차 정의 소유자 지정: 활동 지정을 담당하는 개인 또는 팀을 고유하게 식별합니다. 이들은 올바른 권한, 액세스 및 도구와 적절한 기술을 갖춘 팀원이 활동을 성공적으로 수행할 수 있도록 할 책임이 있습니다. 해당 활동을 수행하는 데 문제가 있는 경우 활동을 수행하는 팀원은 활동 개선에 필요한 상세한 피드백을 제공할 책임이 있습니다.
  - 활동 아티팩트의 메타데이터에서 소유권 캡처: AWS Systems Manager, 문서 및 AWS Lambda와 같은 서비스에서 함수로 자동화된 절차를 사용하면 메타데이터 정보를 태그로 캡처할 수 있습니다. 태그 또는 리소스 그룹을 사용하여 리소스 소유권을 캡처하고 소유권 및 연락처 정보를 지정합니다. AWS Organizations를 사용하여 태그 지정 정책을 생성하고 소유권 및 연락처 정보가 캡처되도록 합니다.

OPS02-BP03 운영 활동에서 성능을 담당하는 소유자 식별

정의된 워크로드를 대상으로 하는 특정 활동 수행에 대한 책임 소재와 그러한 책임이 존재하는 이유를 파악합니다. 활동 수행에 대한 책임 소재를 파악하면 누가 작업을 수행하고, 누가 결과를 확인하며, 누가 활동 소유자에게 피드백을 제공할지 알 수 있습니다.

이 모범 사례 정립의 이점: 활동 수행에 대한 책임 소재를 파악하면 작업이 필요할 때 누구에게 알릴지, 누가 작업을 수행하고 결과를 확인하며 활동 소유자에게 피드백을 제공할지 알 수 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 운영 활동에서 성과를 담당하는 소유자 식별: 환경에서 사용되는 프로세스 및 절차 수행에 대한 책임 소재를 파악합니다.
  - 프로세스 및 절차 식별: 워크로드 지원을 위해 수행되는 운영 활동을 식별합니다. 검색 가능한 위치에 이러한 활동을 문서화합니다.
  - 각 활동 수행에 대한 책임 소재 정의: 활동을 담당하는 팀을 식별합니다. 활동 세부 정보, 활동 수행에 필요한 기술, 올바른 권한, 액세스 및 도구를 갖추고 있는지 확인합니다. 활동 수행 조건(예: 이벤트 또는 일정)을 파악해야 합니다. 조직의 구성원이 특정 요구 사항에 대해 연락할 팀이나 개인을 식별할 수 있도록 이 정보를 검색할 수 있게 설정합니다.

OPS02-BP04 팀원이 담당해야 하는 업무 파악

역할의 책임과 비즈니스 성과에 기여하는 방법을 파악하면 작업의 우선순위와 역할이 중요한 이유를 알 수 있습니다. 이를 통해 팀원은 요구 사항을 인식하고 적절하게 대응할 수 있습니다.

이 모범 사례 확립의 이점: 책임을 파악하면 이를 바탕으로 결정을 내리고, 조치를 취하고, 적절한 소유자에게 활동을 전달할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

- 팀원의 역할 및 책임 파악: 팀원의 역할과 책임을 식별하고 맡은 역할에 대한 기대치를 이해하도록 합니다. 조직의 구성원이 특정 요구 사항에 대해 연락할 팀이나 개인을 식별할 수 있도록 이 정보를 검색할 수 있게 설정합니다.

OPS02-BP05 책임과 소유권을 식별하는 메커니즘

개인이나 팀을 식별하지 못할 경우 소유권을 할당하거나 요구 사항 충족을 위한 계획을 수립할 권한이 있는 사람에게 정해진 경로를 통해 사안을 에스컬레이션할 수 있습니다.

이 모범 사례 정립의 이점: 책임 또는 소유권이 있는 사람을 파악하면 적절한 팀이나 팀원에게 연락하여 요청을 하거나 태스크를 전환할 수 있습니다. 책임 또는 소유권을 할당하거나 요구 사항 충족을 위한 계획을 수립할 권한이 있는 사람을 식별하면 휴지 및 요구 사항 미충족의 위험이 줄어듭니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 책임과 소유권을 식별하는 메커니즘: 조직 구성원에게 소유권과 책임을 발견하고 식별할 수 있는 액세스 가능한 메커니즘을 제공합니다. 이러한 메커니즘을 통해 조직 구성원은 특정 요구 사항에 대해 연락할 팀 또는 개인을 식별할 수 있습니다.

OPS02-BP06 추가, 변경 및 예외를 요청하는 메커니즘

프로세스, 절차 및 리소스의 소유자에게 요청을 보낼 수 있습니다. 요청에는 추가, 변경 및 예외가 포함됩니다. 이러한 요청은 변경 관리 프로세스를 거칩니다. 이점과 위험을 평가한 후 요청이 적절한지 판단하고 정보에 입각한 의사 결정을 통해 실현 가능한 경우에 요청을 승인해야 합니다.

원하는 결과:

- 할당된 소유권에 따라 프로세스, 절차 및 리소스 변경을 요청할 수 있습니다.
- 변경은 이익과 위험을 저울질하면서 의도적으로 이루어집니다.

일반적인 안티 패턴:

- 애플리케이션 배포 방법을 업데이트해야 하지만, 운영 팀의 배포 프로세스 변경을 요청할 수 있는 방법이 없습니다.
- 재해 복구 계획을 업데이트해야 하지만, 변경을 요청할 식별된 소유자가 없습니다.

이 모범 사례 확립의 이점:

- 프로세스, 절차 및 리소스는 요구 사항의 변화에 따라 달라질 수 있습니다.
- 소유자는 정보에 입각하여 변경 시점을 결정할 수 있습니다.
- 변경은 의도적인 방식으로 이루어집니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

이 모범 사례를 구현하려면 프로세스, 절차 및 리소스에 대한 변경을 요청할 수 있어야 합니다. 변경 관리 프로세스는 간단할 수 있습니다. 변경 관리 프로세스를 문서화합니다.

고객 사례

AnyCompany Retail은 책임 할당(RACI) 매트릭스를 사용하여 프로세스, 절차 및 리소스에 대한 변경 사항을 책임지는 소유자를 식별합니다. 문서화된 변경 관리 프로세스가 있어 간편하고 쉽게 변경 작업을 수행할 수 있습니다. RACI 매트릭스와 프로세스를 바탕으로 누구나 변경 요청을 제출할 수 있습니다.

구현 단계

1. 워크로드 및 각 워크로드의 소유자에 대한 프로세스, 절차 및 리소스를 식별합니다. 지식 관리 시스템에 문서화합니다.
   a. OPS02-BP01 리소스 소유자 식별, OPS02-BP02 프로세스 및 절차의 소유자 식별 또는 OPS02-BP03 운영 활동에서 성능을 담당하는 소유자 식별을 구현하지 않았다면 먼저 구현하세요.
2. 조직의 이해관계자와 협력하여 변경 관리 프로세스를 개발합니다. 프로세스에는 리소스, 프로세스 및 절차에 대한 추가, 변경 및 예외가 포함되어야 합니다.
   a. AWS Systems Manager Change Manager를 워크로드 리소스의 변경 관리 플랫폼으로 사용할 수 있습니다.
3. 지식 관리 시스템에 변경 관리 프로세스를 문서화합니다.

구현 계획의 작업 수준: 중간. 변경 관리 프로세스를 개발하려면 조직 전체의 여러 이해관계자와 의견을 조율해야 합니다.

리소스

관련 모범 사례:

- OPS02-BP01 리소스 소유자 식별 - 변경 관리 프로세스를 마련하기 전에 리소스 소유자를 식별해야 합니다.
- OPS02-BP02 프로세스 및 절차의 소유자 식별 - 변경 관리 프로세스를 개발하기 전에 프로세스 소유자를 식별해야 합니다.
- OPS02-BP03 운영 활동에서 성능을 담당하는 소유자 식별 - 변경 관리 프로세스를 개발하기 전에 운영 활동 소유자를 식별해야 합니다.

관련 문서:

- AWS 권장 가이드 - Foundation palybook for AWS large migrations: Creating RACI matrices(AWS 대규모 마이그레이션을 위한 기초 플레이북: RACI 매트릭스 생성)
- 클라우드에서 변경 관리 백서

관련 서비스:

- AWS Systems Manager Change Manager

OPS02-BP07 미리 정의되었거나 협상된 팀 간 책임

팀 간에 서로 협력하고 지원하는 방식에 관한 내용을 정의하거나 협상합니다(예: 응답 시간, 서비스 수준 목표 또는 서비스 수준에 관한 계약). 팀 간 통신 채널이 문서화되어 있습니다. 팀의 작업이 비즈니스 성과에 미치는 영향, 그리고 다른 팀과 조직의 성과에 미치는 영향을 이해하면 작업의 우선순위를 파악하고 적절하게 대응할 수 있습니다.

책임과 소유권을 정의하지 않았거나 알지 못하는 경우 필요한 활동을 적시에 처리하지 못하게 되며 해당 요구 사항을 해결하기 위한 작업이 중복되고 잠재적으로는 상충될 위험이 있습니다.

원하는 결과:

- 팀 간 작업 또는 지원 계약에 동의하고 문서화합니다.
- 서로 지원하거나 협력하는 팀은 통신 채널과 대응 기대치를 정의합니다.

일반적인 안티 패턴:

- 프로덕션에서 문제가 발생하고 2개의 개별 팀이 서로 독립적으로 문제 해결을 시작합니다. 이렇게 서로 분리되어 작업하면 운영 중단이 길어집니다.
- 운영 팀은 개발 팀의 도움이 필요하지만, 응답 시간에 대해서는 합의된 바가 없습니다. 요청이 백로그에 쌓여 있습니다.

이 모범 사례 확립의 이점:

- 팀이 서로 상호 작용하고 지원하는 방법을 알게 됩니다.
- 응답성에 대한 기대치를 알게 됩니다.
- 통신 채널이 명확하게 정의됩니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

이 모범 사례를 구현한다면 팀이 서로 협력하는 방식에 모호함이 사라집니다. 공식적인 합의에서는 팀이 협력하거나 서로를 지원하는 방식을 규정합니다. 팀 간 통신 채널이 문서화되어 있습니다.

고객 사례

AnyCompany Retail의 SRE 팀은 개발 팀과 서비스 수준에 관한 계약을 체결합니다. 개발 팀이 티켓팅 시스템에서 요청할 때마다 15분 안에 응답받기를 기대할 수 있습니다. 사이트 운영 중단이 발생하면 SRE 팀이 개발 팀의 지원을 받아 조사를 주도합니다.

구현 단계

1. 조직 전체의 이해관계자와 협력하여 프로세스 및 절차를 기반으로 팀 간의 계약을 개발합니다.
   a. 프로세스 또는 절차를 두 팀 간에 공유하는 경우 각 팀이 협력할 방식에 대한 런북을 작성합니다.
   b. 팀 간에 종속성이 있는 경우 요청에 대한 응답 SLA에 동의합니다.
2. 지식 관리 시스템에 책임을 문서화합니다.

구현 계획의 작업 수준: 중간. 기존에 팀 간 합의가 이루어지지 않은 경우 조직 전체의 이해관계자와 합의하는 데 노력이 필요할 수 있습니다.

리소스

관련 모범 사례:

- OPS02-BP02 프로세스 및 절차의 소유자 식별 - 팀 간 합의를 수립하기 전에 프로세스 소유권을 확인해야 합니다.
- OPS02-BP03 운영 활동에서 성능을 담당하는 소유자 식별 - 팀 간 합의를 수립하기 전에 운영 활동 소유권을 확인해야 합니다.

관련 문서:

- AWS Executive Insights - Empowering Innovation with the Two-Pizza Team(AWS 경영진 인사이트 - 피자 두 판 규모의 팀으로 혁신 강화)
- AWS 기반 DevOps 소개 - 피자 두 판 규모의 팀

OPS 3 조직 문화는 비즈니스 성과를 어떻게 지원합니까?

팀원이 효과적으로 조치를 취하고 비즈니스 성과를 지원할 수 있도록 팀원에 대한 지원을 제공합니다.

모범 사례

- OPS03-BP01 경영진의 후원 확보

- OPS03-BP02 팀원에게 성과 달성이 위태로울 때 조치를 취할 수 있는 권한 부여
- OPS03-BP03 에스컬레이션 장려
- OPS03-BP04 시기 적절하고 명확하며 실행 가능한 커뮤니케이션
- OPS03-BP05 실험 장려
- OPS03-BP06 팀원의 기술 유지와 증진 지원 및 장려
- OPS03-BP07 리소스 팀 적절히 관리
- OPS03-BP08 팀 내부 및 여러 팀 간에 의견의 다양성 추구 및 장려

OPS03-BP01 경영진의 후원 확보

최고 경영진은 조직에 대한 기대치를 명확하게 설정하고 성공 여부를 평가합니다. 최고 경영진은 조직이 발전하고 모범 사례를 도입하도록 하는 동인이자 후원자이자 지지자입니다.

이 모범 사례 수립의 이점: 경영진의 참여, 명확하게 기대치 전달 및 목표 공유를 통해 팀원은 기대 사항을 파악할 수 있습니다. 성공 평가를 통해 후원자 또는 대리인의 개입을 통해 해결할 수 있도록 성공의 장벽을 파악할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 경영진의 후원 확보: 최고 경영진은 조직에 대한 기대치를 명확하게 설정하고 성공 여부를 평가합니다. 최고 경영진은 조직이 발전하고 모범 사례를 도입하도록 하는 동인이자 후원자이자 지지자입니다.
  - 기대치 설정: 측정 방법을 포함하여 조직의 목표를 정의하고 게시합니다.
  - 목표 달성 추적: 성과가 위태로운 상태일 경우 적절한 조치를 취할 수 있도록 목표의 점진적 달성을 정기적으로 측정하고 결과를 공유합니다.
  - 목표를 달성하는 데 필요한 리소스 제공: 리소스가 여전히 적절한지 또는 새로운 정보, 목표 변경, 책임 또는 비즈니스 환경에 따라 추가 리소스가 필요한지 정기적으로 검토합니다.
  - 팀 지지: 팀과 지속적으로 협력하여 팀의 업무 방식과 팀에 영향을 미치는 외부 요인을 파악합니다. 팀이 외부 요인의 영향을 받는 경우 목표를 재평가하고 적절하게 목표를 조정합니다. 팀 진행을 방해하는 장애물을 파악합니다. 팀을 대신해 장애물을 해결하고 불필요한 부담을 제거하는 데 도움을 줍니다.
  - 모범 사례 도입을 위한 동인: 수량화할 수 있는 이점을 제공하고 생산자와 채택자를 인지하는 모범 사례를 확인합니다. 추가적인 채택을 장려하여 달성된 이점을 확대합니다.

- 팀의 진화를 이끄는 동력: 지속적으로 개선하는 문화를 조성합니다. 개인 및 조직의 성장과 개발을 장려합니다. 시간 경과에 따른 점진적 달성을 필요로 하는, 추구할 장기적 목표를 제공합니다. 이러한 비전을 조정하여 변화하는 요구 사항, 비즈니스 목표 및 비즈니스 환경을 보완합니다.

OPS03-BP02 팀원에게 성과 달성이 위태로울 때 조치를 취할 수 있는 권한 부여

워크로드 소유자는 성과가 위험한 상태일 때 팀원들이 응답할 수 있도록 지침과 범위를 정의했습니다. 에스컬레이션 메커니즘은 이벤트가 정의된 범위를 벗어날 경우 수행할 조치를 정할 때 사용됩니다.

이 모범 사례 수립의 이점: 변경 사항을 조기에 테스트하고 검증함으로써 최소화된 비용으로 문제를 해결하고 고객에게 미치는 영향을 제한할 수 있습니다. 배포 전에 테스트하면 오류 도입을 최소화할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 팀원에게 성과 달성이 위태로울 때 조치를 취할 수 있는 권한 부여: 팀원에게 효과적으로 대응하는 데 필요한 기술을 연습할 수 있는 권한, 도구 및 기회를 제공합니다.
  - 팀원에게 대응하는 데 필요한 기술을 연습할 기회 제공: 프로세스와 절차를 안전하게 테스트하고 교육할 수 있는 안전한 대체 환경을 제공합니다. 팀원들이 안전한 시뮬레이션 환경에서 실제 인시던트에 대응하는 경험을 쌓을 수 있도록 실전 연습을 수행합니다.
  - 팀원의 조치 수행 권한 정의 및 승인: 특히 팀원이 지원하는 워크로드 및 구성 요소에 대한 권한과 액세스를 할당하여 조치를 취할 수 있는 팀원 권한을 정의합니다. 성과가 위험한 상태일 때 조치를 취할 수 있는 권한이 팀원에게 있음을 확인합니다.

OPS03-BP03 에스컬레이션 장려

성과 실현에 실패할 위험이 있는 경우 의사 결정권자와 이해관계자에게 문제를 에스컬레이션하는 메커니즘이 마련되어 있으며 팀원은 이를 적극적으로 활용해야 합니다. 위험을 식별하고 인시던트를 방지할 수 있도록 에스컬레이션을 조기에 자주 수행해야 합니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 조기의 잦은 에스컬레이션 장려: 조직 차원에서 에스컬레이션을 조기에 자주 진행하는 게 모범 사례임을 받아들입니다. 에스컬레이션이 근거가 없는 것으로 판명될 수 있으며 에스컬레이션하지 않아

해당 기회를 놓치는 것보다 인시던트를 방지할 기회를 갖는 것이 낫다는 것을 조직 차원에서 인정하고 수락합니다.

- 에스컬레이션을 위한 메커니즘 보유: 에스컬레이션이 발생하는 시점과 방법을 정의하는 문서화된 절차가 있어야 합니다. 조치를 취하거나 조치와 연락처 정보를 승인할 수 있는 높은 권한이 있는 사람들을 문서화합니다. 팀원이 위험 요소를 처리할 수 있는 사람에게 문제를 이관했거나 워크로드 운영에 대한 위험과 책임을 부담하는 사람에게 연락했다고 만족할 때까지 에스컬레이션이 계속되어야 합니다. 궁극적으로 워크로드와 관련된 모든 의사 결정을 소유하는 사람이 여기에 해당합니다. 에스컬레이션에는 위험의 특성, 워크로드의 중요성, 영향을 받는 사람, 영향의 종류, 긴급성, 즉 영향이 예상되는 시기가 포함되어야 합니다.
- 에스컬레이션하는 직원 보호: 팀원이 부적절한 의사 결정권자나 이해관계자에게로 문제를 에스컬레이션하는 경우 받을 수 있는 불이익으로부터 팀원을 보호하는 정책을 마련합니다. 이러한 일이 발생하는지 여부를 식별하고 적절하게 대응할 수 있는 메커니즘이 마련되어 있습니다.

OPS03-BP04 시기 적절하고 명확하며 실행 가능한 커뮤니케이션

알려진 위험과 예정된 이벤트를 팀원에게 적시에 알리는 데 사용되는 메커니즘이 마련되어 있습니다. 조치가 필요한지 여부와 어떤 조치가 필요한지 판단하고 적시에 조치를 취할 수 있도록 필요한 컨텍스트, 세부 정보 및 시간(가능한 경우)이 제공됩니다. 예를 들어 소프트웨어 취약성에 대한 알림을 제공하여 패치를 신속하게 적용하도록 하거나, 예정된 판매 프로모션에 대한 알림을 제공하여 서비스 중단의 위험을 방지하기 위한 변경 동결 기능을 구현하게 할 수 있습니다. 대기 중인 활동을 팀원이 식별할 수 있도록 예정된 이벤트를 변경 일정이나 유지 관리 일정에 기록할 수 있습니다.

원하는 결과:

- 의사소통으로 상황, 세부 사항 및 예상 시간을 알 수 있습니다.
- 팀원은 의사소통에 대응하여 언제 어떻게 행동해야 하는지 명확하게 이해하게 됩니다.
- 변경 일정을 활용하여 예상되는 변경 사항에 대한 공지를 제공합니다.

일반적인 안티 패턴:

- 긍정 오류 알림이 일주일에 여러 번 발생합니다. 알림이 발생할 때마다 음소거해야 합니다.
- 보안 그룹을 변경하라는 메시지가 표시되지만, 변경이 필요한 예상 시점이 제공되지 않습니다.
- 시스템이 확장되지만 작업이 필요하지 않을 때 채팅에서 지속적으로 알림을 받습니다. 채팅 채널을 피하고 중요한 알림을 놓치게 됩니다.
- 운영 팀이 모르는 새에 프로덕션이 변경됩니다. 변경 내용은 알림을 트리거하고 팀이 철야 근무를 해야 합니다.

이 모범 사례 확립의 이점:

- 알림이 누적되면서 조직이 받는 피로를 방지합니다.
- 팀원은 필요한 상황 정보와 기대 수준을 알고 행동할 수 있습니다.
- 변경 기간 동안 변경할 수 있어 위험이 줄어듭니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

이 모범 사례를 구현하려면 조직 전체의 이해관계자와 협력하여 통신 표준에 동의해야 합니다. 이러한 표준을 조직에 공개적으로 알리세요. 긍정 오류 또는 항상 켜지는 알림을 식별하고 제거합니다. 변경 일정을 통해 팀원이 작업을 수행할 수 있는 시점과 보류 중인 활동을 알 수 있습니다. 통신을 통해 필요한 상황 정보를 바탕으로 명확한 작업이 수행되는지 확인합니다.

고객 사례

AnyCompany Retail은 채팅을 주 통신 매체로 사용합니다. 알림 및 기타 정보가 특정 채널에 쌓입니다. 누군가가 조치를 취해야 할 때 원하는 결과가 명확하게 명시되어 있으며, 대부분의 경우 참조할 수 있는 런북이나 플레이북이 제공됩니다. 이들은 변경 일정을 통해 프로덕션 시스템에 대한 주요 변경을 예약합니다.

구현 단계

1. 긍정 오류 알림 또는 지속적으로 트리거되는 알림을 분석합니다. 사용자의 개입이 필요할 때 트리거되도록 제거하거나 변경합니다. 알림이 트리거되면 런북 또는 플레이북을 제공합니다.
   a. AWS Systems Manager 문서를 사용하여 알림 플레이북과 런북을 작성할 수 있습니다.
2. 적절한 대응을 가능하게 하는 충분한 정보와 함께 명확하고 실행 가능한 방식으로 위험 또는 계획된 이벤트를 알리는 메커니즘이 마련되어 있습니다. 이메일 목록 또는 채팅 채널을 사용하여 계획된 이벤트 전에 알림을 보냅니다.
   a. AWS Chatbot을 조직 메시징 플랫폼에서 알림을 보내고 이벤트에 응답하는 데 사용할 수 있습니다.
3. 계획된 이벤트를 검색할 수 있는 액세스 가능한 정보 출처를 제공합니다. 동일한 시스템의 계획된 이벤트에 대한 알림을 제공합니다.
   a. AWS Systems Manager Change Calendar는 변경이 발생 가능한 경우 변경 기간을 설정하는 데 사용할 수 있습니다. 이를 통해 팀원이 안전하게 변경할 수 있는 시점을 알 수 있습니다.
4. 취약성 알림과 패치 정보를 모니터링하여 워크로드 구성 요소와 관련된 잠재적 위험 및 취약성을 파악합니다. 팀원에게 조치를 취할 수 있도록 알림을 제공합니다.

a. AWS 보안 공지를 구독하여 AWS의 취약성 알림을 받아볼 수 있습니다.

리소스

관련 모범 사례:

- OPS07-BP03 런북을 사용한 절차 수행 - 결과가 알려졌을 때 런북을 제공하여 소통을 실행할 수 있도록 지원합니다.
- OPS07-BP04 플레이북을 사용하여 문제 조사 - 결과를 알 수 없는 경우 플레이북을 통해 소통을 실현할 수 있습니다.

관련 문서:

- AWS 보안 공지
- OpenCVE

관련 예시:

- Well-Architected 실습: 인벤토리 및 패치 관리(레벨 100)

관련 서비스:

- AWS Chatbot
- AWS Systems Manager Change Calendar
- AWS Systems Manager 문서

OPS03-BP05 실험 장려

실험은 새로운 아이디어를 제품과 기능으로 탈바꿈하는 촉매제입니다. 실험은 학습을 가속화하고 팀원의 관심과 참여를 유지합니다. 팀원은 혁신을 추진하기 위해 자주 실험하도록 장려됩니다. 원하지 않는 결과가 나오더라도 하지 말아야 할 것을 알았다는 사실만으로 실험은 가치가 있습니다. 원치 않는 결과가 나온 성공한 실험에 대해 팀원에게 불이익을 가하지 않습니다.

원하는 결과:

- 조직이 혁신을 촉진하기 위해 실험을 장려합니다.
- 실험을 통해 배울 수 있는 기회가 주어집니다.

일반적인 안티 패턴:

- A/B 테스트를 진행하려고 하는데 실험을 실행할 수 있는 메커니즘이 없습니다. UI 변경 사항을 테스트할 수 없는 상태에서 배포합니다. 이는 부정적인 고객 경험으로 이어집니다.
- 회사에는 스테이지와 프로덕션 환경만 있습니다. 새 기능이나 제품을 실험할 샌드박스 환경이 없어 프로덕션 환경에서 실험해야 합니다.

이 모범 사례 확립의 이점:

- 실험은 혁신을 불러옵니다.
- 실험을 통해 사용자의 피드백에 신속하게 반응할 수 있습니다.
- 조직은 학습하는 문화를 조성할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

실험은 안전한 방법으로 실행되어야 합니다. 여러 환경을 활용하여 프로덕션 리소스를 손상시키지 않고 실험할 수 있습니다. A/B 테스트 및 기능 플래그를 사용하여 실험을 테스트합니다. 팀원에게 샌드박스 환경에서 실험할 수 있는 기능을 제공합니다.

고객 사례

AnyCompany Retail은 실험을 장려합니다. 팀원은 주당 근무 시간의 20%를 새로운 기술을 실험하거나 학습하는 데 사용할 수 있습니다. 이들은 혁신을 가능케 하는 샌드박스 환경을 사용하고 있습니다. A/B 테스트는 새로운 기능을 실제 사용자 피드백으로 검증하기 위해 사용됩니다.

구현 단계

1. 조직 전체에서 경영진과 협력하여 실험을 지원합니다. 팀원이 안전한 방법으로 실험하도록 장려해야 합니다.
2. 팀원이 안전하게 실험할 수 있는 환경을 제공합니다. 프로덕션 환경과 같은 환경에 액세스할 수 있어야 합니다.
   a. 별도의 AWS 계정을 사용하여 실험용 샌드박스 환경을 생성할 수 있습니다. AWS Control Tower를 사용하여 이러한 계정을 프로비저닝할 수 있습니다.
3. 기능 플래그 및 A/B 테스트를 사용하여 안전하게 실험하고 사용자 피드백을 수집합니다.
   a. AWS AppConfig 기능 플래그는 기능 플래그를 만드는 기능을 제공합니다.

b. Amazon CloudWatch Evidently는 제한된 배포 환경에서 A/B 테스트를 실행하는 데 사용할 수 있습니다.

c. AWS Lambda 버전을 사용하여 베타 테스트를 위해 새로운 기능 버전을 배포할 수 있습니다.

구현 계획의 작업 수준: 높음. 팀원에게 실험할 환경과 실험을 안전하게 수행할 방법을 제공하려면 상당한 투자가 필요할 수 있습니다. 기능 플래그를 사용하거나 A/B 테스트를 지원하기 위해 애플리케이션 코드를 수정해야 할 수도 있습니다.

리소스

관련 모범 사례:

- OPS11-BP02 인시던트 사후 분석 수행 - 실험과 마찬가지로 인시던트로부터 배우는 일은 혁신을 이끄는 중요한 동인입니다.
- OPS11-BP03 피드백 루프 구현 - 피드백 루프는 실험의 중요한 부분입니다.

관련 문서:

- An Inside Look at the Amazon Culture: Experimentation, Failure, and Customer Obsession(Amazon 문화 들여다보기: 실험, 실패 및 고객 중심)
- Best practices for creating and managing sandbox accounts in AWS(AWS에서 샌드박스 계정을 생성하고 관리하기 위한 모범 사례)
- Create a Culture of Experimentation Enabled by the Cloud(클라우드를 통한 실험 문화 조성)
- Enabling experimentation and innovation in the cloud at SulAmérica Seguros(SulAmérica Seguros의 클라우드 내 실험 및 혁신 지원)
- Experiment More, Fail Less(실험할수록 줄어드는 실패)
- 여러 계정을 사용하여 AWS 환경 구성 - 샌드박스 OU
- Using AWS AppConfig Feature Flags(AWS AppConfig 기능 플래그 사용)

관련 동영상:

- AWS On Air ft. Amazon CloudWatch Evidently | AWS Events (AWS On Air - Amazon CloudWatch Evidently 소개 | AWS Events)
- AWS On Air San Fran Summit 2022 ft. AWS AppConfig Feature Flags integration with Jira (AWS On Air San Fran Summit 2022 - Jira와 AWS AppConfig 기능 플래그 통합)

- AWS re:Invent 2022 - A deployment is not a release: Control your launches w/feature flags (BOA305-R)(AWS re:Invent 2022 - 배포는 릴리스가 아님: 기능 플래그를 사용한 출시 제어 (BOA305-R))
- Programmatically Create an AWS 계정 with AWS Control Tower(AWS Control Tower를 통해 프로그래밍 방식으로 AWS 계정 생성)
- AWS Organizations의 모범 사례를 사용하는 다중 계정 AWS 환경 설정

관련 예시:

- AWS Innovation Sandbox
- End-to-end Personalization 101 for E-Commerce(전자 상거래를 위한 엔드 투 엔드 개인화 101)

관련 서비스:

- Amazon CloudWatch Evidently
- AWS AppConfig
- AWS Control Tower

OPS03-BP06 팀원의 기술 유지와 증진 지원 및 장려

팀은 새로운 기술을 도입하고 워크로드 지원 책임과 수요 변화를 지원하기 위해 기술 역량을 키워야 합니다. 새로운 기술 영역의 기술 증진은 팀원의 만족도를 높이고 혁신을 뒷받침합니다. 발전하는 기술을 검증하고 인증하는 업계 자격증을 획득하고 관리하도록 팀원을 독려합니다. 지식이 효과적으로 전달되도록 하고, 제도적 지식을 갖춘 경험 많고 숙련된 직원을 잃은 경우에 중대한 영향이 발생할 위험을 줄일 수 있도록 교차 교육을 실시합니다. 학습을 위해 체계적으로 정해진 교육 시간을 제공합니다.

AWS에서는 AWS 시작하기 리소스 센터, AWS 블로그, AWS Online Tech Talks, AWS 이벤트 및 웨비나 및 AWS Well-Architected 실습같은 다양한 리소스를 제공합니다. 이러한 리소스에서는 팀을 대상으로 교육을 진행하는 데 활용할 수 있는 지침, 예제 및 자세한 연습 과정을 제공합니다.

AWS는 다음에서 AWS의 운영을 통해 학습한 모범 사례와 패턴을 공유합니다. Amazon Builders' Library 또한 다음을 통해 도움이 되는 방대한 기타 교육 자료를 제공합니다. AWS 블로그 및 공식 AWS 팟캐스트.

팀에 관련 정보를 제공하는 데 AWS에서 지원하는 교육 리소스, 즉 Well-Architected 실습과 AWS Support (AWS 지식 센터, AWS 토론 양식및 AWS Support 센터) 및 AWS 설명서 를 이용해야 합니다.

AWS 관련 문의 사항에 대해 지원을 받으려면 AWS Support 센터를 통해 AWS Support에 지원을 요청하십시오.

AWS 교육 and Certification 에서는 AWS 기초에 관한 자습형 디지털 과정을 통해 무료 교육을 제공합니다. 강사 주도형 교육에 등록하여 팀이 AWS 기술을 연마하도록 추가로 지원할 수도 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 팀원의 기술 유지와 증진 지원 및 장려: 새로운 기술을 도입하고, 혁신을 지원하고, 워크로드 지원 책임과 수요 변화를 뒷받침하려면 지속적인 교육이 필요합니다.
  - 교육 리소스 제공: 체계적인 시간, 교육 자료 이용, 실습 리소스 및 교사와 동료에게서 배울 수 있는 기회를 제공하는 컨퍼런스 및 전문 조직에 참석 지원을 제공합니다. 수습 팀원에게 선임 팀원의 지도와 조언을 받을 수 있는 기회를 제공하고 선임 팀원의 일과 방법 및 기술을 배울 수 있게 합니다. 보다 넓은 시야를 확보하기 위해 작업과 직접적으로 관련되지 않은 콘텐츠에 대해 학습하도록 장려합니다.
  - 팀 교육 및 팀 간 참여: 팀원의 지속적인 교육 요구 사항을 계획합니다. 팀원이 다른 팀(임시로 또는 영구적으로)에 합류하여 전체 조직에 도움이 되는 기술과 모범 사례를 공유할 수 있는 기회를 제공합니다.
  - 업계 자격증 획득 및 유지 지원: 팀원이 배운 내용을 확인하고 그 성과를 인정하는 업계 자격증을 획득하고 유지하도록 지원합니다.

리소스

관련 문서:

- AWS 시작하기 리소스 센터
- AWS 블로그
- AWS 클라우드 규정 준수
- AWS 토론 양식
- AWS 설명서
- AWS Online Tech Talks
- AWS 이벤트 및 웨비나
- AWS 지식 센터
- AWS Support

- AWS 교육 and Certification
- AWS Well-Architected 실습,
- Amazon Builders' Library
- 공식 AWS 팟캐스트.

OPS03-BP07 리소스 팀 적절히 관리

워크로드 요구 사항을 지원할 수 있도록 팀원 역량을 유지하고 도구와 리소스를 제공합니다. 과중한 업무를 수행하는 팀원은 인적 오류로 인한 인시던트의 위험이 큽니다. 자주 수행하는 활동을 자동화하는 등 도구 및 리소스에 투자하면 팀의 효율성이 높아져 팀원이 더 많은 활동을 지원할 수 있게 됩니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 리소스 팀 적절히 관리: 팀의 성공을 깊이 있게 이해하고, 성공 또는 실패를 부른 요인을 파악해야 합니다. 적절한 리소스로 팀 지원 활동을 합니다.
  - 팀 성과 이해: 팀의 운영 성과 달성과 자산 개발을 측정합니다. 시간 경과에 따른 출력 및 오류 발생률의 변화를 추적합니다. 팀과 협력하여 업무에 영향을 미치는 문제(예: 책임 증가, 기술 변화, 인력 손실 또는 지원 고객 증가)를 파악합니다.
  - 팀 성과에 미치는 영향 파악: 팀과 지속적으로 협력하여 팀의 업무 방식과 팀에 영향을 미치는 외부 요인을 파악합니다. 팀이 외부 요인의 영향을 받는 경우 목표를 재평가하고 적절하게 목표를 조정합니다. 팀 진행을 방해하는 장애물을 파악합니다. 팀을 대신해 장애물을 해결하고 불필요한 부담을 제거하는 데 도움을 줍니다.
  - 팀의 성공에 필요한 리소스 제공: 리소스가 여전히 적절한지, 추가 리소스가 필요한지 정기적으로 검토하고 지원 팀을 적절히 조정합니다.

OPS03-BP08 팀 내부 및 여러 팀 간에 의견의 다양성 추구 및 장려

조직 간의 다양성을 활용하여 여러 가지 고유한 관점을 모색합니다. 이러한 관점을 통해 혁신을 증진하고, 기존의 추정 사항에 의문을 제기하며, 확증 편향의 위험을 줄일 수 있습니다. 팀 내에서 포용성, 다양성 및 접근성을 높여 유익한 관점을 확보합니다.

조직 문화는 팀원의 업무 만족도와 팀원 이직률에 직접적인 영향을 미칩니다. 팀원의 참여와 역량을 통해 비즈니스의 성공을 뒷받침할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 다양한 의견과 관점 모색: 모든 사람이 기여하도록 장려합니다. 대표가 불분명한 소수 그룹의 의견을 경청합니다. 미팅에서 역할과 책임을 교대로 맡습니다.
  - 역할 및 책임 확대: 다른 상황에서는 맡을 수 없는 역할을 담당할 수 있는 기회를 팀원에게 제공합니다. 마찬가지로 팀원은 다른 상황에서는 불가능할 수 있는 새로운 팀원과의 상호 작용 및 역할에서 경험과 관점을 얻습니다. 팀원은 상호 작용하는 새로운 역할과 팀원에 자신의 경험과 관점을 적용합니다. 관점이 증가함에 따라 추가적인 비즈니스 기회가 나타나거나 새로운 개선 기회를 찾을 수 있습니다. 팀원들로 하여금 다른 구성원들이 수행하는 일반적인 작업을 번갈아 수행하도록 하여 해당 작업의 요구 사항과 영향에 대한 이해를 돕습니다.
  - 안전하고 환영받는 환경 제공: 조직 내 팀원의 정신적, 신체적 안전을 보호하는 정책과 규제 수단을 마련합니다. 팀원은 보복에 대한 두려움 없이 상호 작용할 수 있어야 합니다. 팀원들이 안전하고 환영 받는다고 느낄 때 참여와 생산성이 향상됩니다. 조직이 다양할수록 고객을 비롯하여 지원하는 인력을 더 잘 이해할 수 있습니다. 팀원들이 편하고 자유롭게 이야기할 수 있고 자신의 의견이 존중된다고 확신할 때 마케팅 기회, 접근성 요구 사항, 소외된 시장 부문, 환경에서 알려지지 않은 위험과 같은 귀중한 인사이트를 공유할 가능성이 더 높아집니다.
  - 팀원의 완전한 참여 지원: 직원이 모든 업무 관련 활동에 완전히 참여하는 데 필요한 리소스를 제공합니다. 일상적인 문제에 직면하는 팀원들은 이러한 문제를 해결할 수 있는 기술을 개발했습니다. 이러한 고유하게 개발된 기술은 조직에 상당한 이점을 가져올 수 있습니다. 팀원들에게 필요한 설비를 지원하면 그들의 조력을 통해 얻을 수 있는 혜택이 늘어납니다.

준비

질문

- OPS 4 운영 상태를 파악할 수 있도록 어떻게 워크로드를 설계하십니까?
- OPS 5 귀사는 어떻게 결함을 줄이고 수정 작업을 쉽게 수행하고 프로덕션으로 이어지는 흐름을 개선하십니까?
- OPS 6 배포 위험을 어떻게 최소화하고 있습니까?
- OPS 7 귀사가 워크로드를 지원할 준비가 되어있는지 어떻게 알 수 있습니까?

OPS 4 운영 상태를 파악할 수 있도록 어떻게 워크로드를 설계하십니까?

모든 구성 요소에서 지표, 로그, 추적 등의 내부 상태를 파악하는 데 필요한 정보를 제공하도록 워크로드를 설계합니다. 이렇게 하면 효율적으로 적절한 대응을 할 수 있습니다.

모범 사례

- OPS04-BP01 애플리케이션 텔레메트리 구현
- OPS04-BP02 워크로드 원격 측정 구현 및 구성
- OPS04-BP03 사용자 활동 원격 측정 구현
- OPS04-BP04 종속성 원격 측정 구현
- OPS04-BP05 트랜잭션 추적 기능 구현

OPS04-BP01 애플리케이션 텔레메트리 구현

애플리케이션 텔레메트리 기능은 워크로드를 관찰하기 위한 기반입니다. 애플리케이션은 애플리케이션의 상태와 비즈니스 성과 달성에 대한 인사이트를 제공하는 텔레메트리 기능을 지원해야 합니다. 문제 해결부터 새로운 기능의 영향력 측정까지, 애플리케이션 텔레메트리 기능은 워크로드의 구축, 운영 및 발전 방법을 제시합니다.

애플리케이션 텔레메트리 기능은 지표와 로그로 구성됩니다. 지표는 맥박이나 온도와 같은 진단 정보라고 할 수 있습니다. 지표는 종합적으로 사용되어 애플리케이션의 상태를 설명합니다. 시간의 흐름에 따라 지표를 수집하면 기준을 설정하고 이상 징후를 탐지할 수 있습니다. 로그는 애플리케이션이 내부 상태 또는 발생한 이벤트와 관련해서 보내는 메시지입니다. 로깅되는 이벤트의 예로는 오류 코드, 거래 식별자, 사용자 활동을 들 수 있습니다.

원하는 결과:

- 애플리케이션은 비즈니스 성과 달성 여부와 상태에 대한 인사이트를 알려 주는 지표와 로그를 제공합니다.
- 지표와 로그는 워크로드의 모든 애플리케이션에 대해 중앙 집중식으로 저장됩니다.

일반적인 안티 패턴:

- 애플리케이션이 텔레메트리를 내보내지 않습니다. 무언가 잘못되었을 때 고객이 제공하는 정보에 의존할 수 밖에 없습니다.
- 고객이 애플리케이션이 응답하지 않다고 보고했습니다. 텔레메트리가 없으며 현재 사용자 경험을 파악하기 위해 애플리케이션을 직접 사용하지 않고 문제가 존재하는지 확인하거나 문제를 특징 지을 수 없습니다.

이 모범 사례 확립의 이점:

- 애플리케이션의 상태, 사용자 경험 및 비즈니스 성과 달성을 파악할 수 있습니다.
- 애플리케이션 상태 변화에 신속하게 대처할 수 있습니다.
- 애플리케이션 상태 추세를 파악할 수 있습니다.
- 정보를 바탕으로 애플리케이션 개선을 위한 결정을 내릴 수 있습니다.
- 애플리케이션 문제를 신속하게 감지하고 해결할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

애플리케이션 텔레메트리는 텔레메트리를 저장할 위치를 파악하고, 애플리케이션 상태를 설명하는 텔레메트리를 식별하고, 애플리케이션이 텔레메트리를 내보내도록 계측하는 3단계로 구현됩니다.

고객 사례

AnyCompany Retail에는 마이크로서비스 기반 아키텍처가 있습니다. 이 회사에서는 아키텍처 설계 과정에서 각 마이크로서비스의 상태를 이해하는 데 도움이 되는 애플리케이션 텔레메트리를 식별했습니다. 예를 들어, 사용자 장바구니 서비스에서 장바구니에 추가, 구매 포기, 장바구니에 항목을 추가하는 데 걸린 시간 등의 이벤트에 대한 텔레메트리를 내보냅니다. 모든 마이크로서비스에서 오류, 경고 및 트랜잭션 정보를 기록합니다. 텔레메트리 데이터가 저장 및 분석을 위해 Amazon CloudWatch로 보내집니다.

구현 단계

1. 워크로드에서 애플리케이션의 텔레메트리를 저장할 중앙 위치를 파악합니다. 이 위치는 텔레메트리 수집 및 분석 기능을 둘 다 지원해야 합니다. 이상 탐지 및 자동화된 인사이트는 권장 기능입니다.
   a. Amazon CloudWatch에서 텔레메트리 수집, 대시보드, 분석 및 이벤트 생성 기능을 제공합니다.
2. 어떤 텔레메트리가 필요한지 파악하려면 애플리케이션의 상태가 어떻습니까?라는 질문에 답변하는 것부터 시작하세요. 애플리케이션에서 이러한 질문의 답을 줄 수 있는 로그와 지표를 내보내야 합니다. 기존 애플리케이션 텔레메트리로 해당 질문에 답할 수 없는 경우 비즈니스 및 엔지니어링 이해관계자와 협력하여 텔레메트리 요구 사항 목록을 작성하세요.
   a. 새로운 애플리케이션 텔레메트리를 식별하고 개발할 때는 AWS 계정 팀에 전문 기술 조언을 요청할 수 있습니다.
3. 추가 애플리케이션 텔레메트리가 확인되면 엔지니어링 이해관계자와 협력하여 애플리케이션을 계측합니다.

a. AWS Distro for Open Telemetry에서는 애플리케이션 텔레메트리를 수집하는 에이전트, 라이브러리, API를 제공합니다. 이 예에서는 사용자 지정 지표를 통해 JavaScript 애플리케이션을 계측하는 방법을 보여 줍니다.
b. AWS에서 제공하는 관찰성 서비스를 이해하려는 경우 One Observability Workshop을 진행하거나 AWS 계정 팀의 지원을 요청하세요.
c. 애플리케이션 텔레메트리에 대한 자세한 내용은 Amazon Builder 라이브러리에서 운영 가시성을 위한 분산 시스템 계측 문서를 참조하세요. 이 문서에서는 Amazon이 애플리케이션을 계측하는 방법과 어떻게 고유한 계측 지침을 개발하기 위한 가이드의 역할을 할 수 있는지 설명합니다.

구현 계획의 작업 수준: 높음. 애플리케이션을 계측하고 텔레메트리 스토리지를 중앙화하려면 상당한 투자가 필요할 수 있습니다.

리소스

관련 모범 사례:

the section called “OPS04-BP02 워크로드 원격 측정 구현 및 구성” – 애플리케이션 텔레메트리는 워크로드 텔레메트리를 구성하는 요소입니다. 전체 워크로드의 상태를 이해하려면 워크로드를 구성하는 개별 애플리케이션의 상태를 파악해야 합니다.

the section called “OPS04-BP03 사용자 활동 원격 측정 구현” – 사용자 활동 텔레메트리는 애플리케이션 텔레메트리의 하위 집합인 경우가 많습니다. 장바구니에 추가 이벤트, 클릭 스트림 또는 완료된 트랜잭션과 같은 사용자 활동은 사용자 경험에 대한 인사이트를 제공합니다.

the section called “OPS04-BP04 종속성 원격 측정 구현” – 종속성 점검은 애플리케이션 텔레메트리와 관련이 있으며, 애플리케이션에 계측될 수 있습니다. 애플리케이션이 DNS 또는 데이터베이스와 같은 외부 종속성에 의존하는 경우 애플리케이션이 도달 가능성, 시간 초과, 기타 이벤트에 대한 로그와 지표를 내보낼 수 있습니다.

the section called “OPS04-BP05 트랜잭션 추적 기능 구현” – 워크로드 전체에서 트랜잭션을 추적하려면 각 애플리케이션이 공유 이벤트를 처리하는 방법에 대한 정보를 생성해야 합니다. 애플리케이션 텔레메트리를 통해 개별 애플리케이션에서 해당 이벤트를 처리하는 방식이 내보내집니다.

the section called “OPS08-BP02 워크로드 지표 정의” – 워크로드 지표는 워크로드의 주요 상태를 나타냅니다. 주요 애플리케이션 지표는 워크로드 지표의 일부입니다.

관련 문서:

- AWS Builders Library – 운영 가시성을 위한 분산 시스템 계측

- AWS Distro for OpenTelemetry
- AWS Well-Architected 운영 우수성 백서 - 텔레메트리 설계
- 필터를 사용하여 로그 이벤트에서 지표 생성
- Amazon CloudWatch를 통한 로깅 및 모니터링 구현
- AWS Distro for OpenTelemetry를 사용한 애플리케이션 상태 및 성능 모니터링
- 신규 – Amazon CloudWatch 에이전트를 통해 사용자 지정 애플리케이션 지표를 효과적으로 모니터링하는 방법
- AWS에서의 관찰성
- 시나리오 – CloudWatch로 지표 게시
- 구축 시작 - 애플리케이션을 효과적으로 모니터링하는 방법
- AWS SDK에서 CloudWatch 사용

관련 동영상:

- AWS re:Invent 2021 - Observability the open-source way(AWS re:Invent 2021 - 오픈 소스 방식의 관찰성)
- Collect Metrics and Logs from Amazon EC2 instances with the CloudWatch Agent(CloudWatch 에이전트를 사용하여 Amazon EC2 인스턴스에서 지표 및 로그 수집)
- How to Easily Setup Application Monitoring for Your AWS Workloads - AWS Online Tech Talks(AWS 워크로드에 적합한 애플리케이션 모니터링을 쉽게 설정하는 방법 - AWS Online Tech Talks)
- Mastering Observability of Your Serverless Applications - AWS Online Tech Talks(서버리스 애플리케이션의 관찰성 마스터링 - AWS Online Tech Talks)
- Open Source Observability with AWS - AWS Virtual Workshop(AWS를 통한 오픈 소스 관찰성 - AWS 가상 워크숍)

관련 예시:

- AWS 로깅 및 모니터링 예시 리소스
- AWS 솔루션: Amazon CloudWatch 모니터링 프레임워크
- AWS 솔루션: 중앙 집중식 로깅
- One Observability Workshop

관련 서비스:

- Amazon CloudWatch

OPS04-BP02 워크로드 원격 측정 구현 및 구성

내부 상태와 현재 상태 관련 정보(예: API 호출 볼륨, HTTP 상태 코드, 크기 조정 이벤트)를 내보내도록 워크로드를 설계 및 구성합니다. 이 정보를 사용하면 대응이 필요한 경우를 확인할 수 있습니다.

서비스, 즉 Amazon CloudWatch 와 같은 서비스를 사용하여 워크로드 구성 요소(예: 다음의 API 로그 - AWS CloudTrail, AWS Lambda 지표, Amazon VPC 흐름 로그및 기타 서비스)는 SAP NetWeaver Guide Finder 및 SAP NetWeaver Security Guide를 참조하세요.

일반적인 안티 패턴:

- 고객이 성능 저하에 대해 불만을 제기하고 있습니다. 애플리케이션에 대한 최근 변경 사항이 없으므로 워크로드 구성 요소에 문제가 있다고 의심됩니다. 성능 저하를 유발하는 구성 요소를 확인하기 위해 분석할 원격 측정이 없습니다.
- 애플리케이션에 연결할 수 없습니다. 원격 측정이 부족하여 네트워크 문제인지 확인할 수 없습니다.

이 모범 사례 수립의 이점: 워크로드 내부에서 어떤 일이 일어나는지 파악하면 필요한 경우 대응이 가능합니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 로그 및 지표 원격 측정 구현: 워크로드를 계측하여 내부 상태 및 비즈니스 성과 달성에 대한 정보를 내보냅니다. 이 정보를 사용하여 대응이 필요한 경우를 확인합니다.
  - Amazon CloudWatch를 사용하여 VM에 대한 관찰성 향상 - AWS Online Tech Talks
  - Amazon CloudWatch 작동 방식
  - Amazon CloudWatch란 무엇입니까?
  - Amazon CloudWatch 지표 사용
  - Amazon CloudWatch Logs란 무엇입니까?
    - 워크로드 원격 측정 구현 및 구성: 내부 상태와 현재 상태 관련 정보(예: API 호출 볼륨, HTTP 상태 코드, 크기 조정 이벤트)를 내보내도록 워크로드를 설계 및 구성합니다.
      - Amazon CloudWatch 지표 및 차원 참조

- AWS CloudTrail
- AWS CloudTrail란 무엇입니까?
- VPC 흐름 로그

리소스

관련 문서:

- AWS CloudTrail
- Amazon CloudWatch 설명서
- Amazon CloudWatch 지표 및 차원 참조
- Amazon CloudWatch 작동 방식
- Amazon CloudWatch 지표 사용
- VPC 흐름 로그
- AWS CloudTrail란 무엇입니까?
- Amazon CloudWatch Logs란 무엇입니까?
- Amazon CloudWatch란 무엇입니까?

관련 동영상:

- AWS에서의 애플리케이션 성능 관리
- Amazon CloudWatch를 사용하여 VM에 대한 관찰 가능성 향상
- Amazon CloudWatch를 사용하여 VM에 대한 관찰성 향상 - AWS Online Tech Talks

OPS04-BP03 사용자 활동 원격 측정 구현

애플리케이션 코드를 계측하여 사용자 활동 관련 정보를 내보냅니다. 사용자 활동의 예로는 클릭 스트림 또는 시작, 중단 및 완료된 트랜잭션이 있습니다. 이 정보를 사용하면 애플리케이션 사용 방법과 사용 패턴을 파악하고 대응이 필요한 경우를 확인할 수 있습니다. 실제 사용자 활동을 캡처하면 프로덕션 워크로드를 모니터링하고 테스트하는 데 사용할 수 있는 가상 활동을 구축할 수 있습니다.

원하는 결과:

- 워크로드가 모든 애플리케이션에서 사용자 활동에 대한 원격 측정을 수행합니다.

- 가상 사용자 활동을 통해 사용량이 적은 시간대에 애플리케이션을 모니터링할 수 있습니다.

일반적인 안티 패턴:

- 개발자는 사용자 원격 측정 없이 새로운 기능을 배포했습니다. 고객이 이 기능을 사용하고 있는지는 고객에게 묻지 않고는 알 수 없습니다.
- 프런트 엔드 애플리케이션에 배포하면 활용률이 향상됩니다. 사용자 활동 원격 측정이 부족하기 때문에 정확한 문제를 식별하기가 어렵습니다.
- 사용량이 적은 시간대에 애플리케이션에 문제가 발생합니다. 가상 사용자 활동을 구성하지 않아 사용자가 온라인에 접속하는 아침까지 문제를 인식하지 못합니다.

이 모범 사례 확립의 이점:

- 일반적인 사용자 패턴 또는 예상치 못한 동작을 이해하여 비즈니스 목표에 맞게 애플리케이션의 기능을 최적화합니다.
- 사용자의 관점에서 애플리케이션을 모니터링하여 연결 끊김 또는 느린 클릭 응답과 같은 사용자 경험의 문제를 탐지합니다
- 영향을 받는 사용자가 수행한 단계를 추적하여 문제의 근본 원인을 식별합니다.
- 가상 사용자 활동은 사용량이 적은 시간 동안 성능 저하의 조기 경고 신호를 제공하여 실제 사용자가 영향을 받기 전에 수정 조치를 취할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

애플리케이션 코드를 설계하여 사용자 활동 관련 정보를 내보냅니다. 이 정보를 사용하면 애플리케이션 사용 방법과 사용 패턴을 파악하고 대응이 필요한 경우를 확인할 수 있습니다. 가상 사용자 활동을 통해 사용량이 적은 시간대의 애플리케이션 성능에 대한 인사이트를 제공합니다.

고객 사례

AnyCompany Retail은 애플리케이션의 여러 계층에서 사용자 활동 원격 측정을 구현합니다. 프런트 엔드 원격 측정은 포인터와 이동 이벤트를 추적하는 반면, 백엔드 마이크로서비스는 사용자의 장바구니에 항목을 추가하고 결제하는 것과 같은 원격 측정 추적 이벤트를 전송합니다. 이들 모두 사용자 경험에 대한 관찰성을 제공합니다. AnyCompany Retail에서는 또한 워크로드에 사용자가 적을 때 문제를 파악하기 위해 가상 사용자 원격 측정을 활용합니다.

구현 단계

1. 사용자 활동에 대한 원격 측정(지표, 이벤트, 로그, 추적)을 전송하도록 애플리케이션을 계측합니다. 계측되면 사용자가 사용자 인터페이스와 상호 작용할 때 프런트 엔드 구성 요소가 원격 측정을 자동으로 전송합니다. 백엔드 애플리케이션은 사용자 이벤트 및 트랜잭션에 대한 원격 측정을 전송합니다.
   a. Amazon CloudWatch RUM은 프런트 엔드 애플리케이션에 대한 최종 사용자 경험 관련 인사이트를 제공할 수 있습니다.
   b. AWS Distro for Open Telemetry를 사용하여 애플리케이션에서 원격 측정을 계측하고 캡처할 수 있습니다.
   c. Amazon Pinpoint에서는 캠페인을 통해 사용자 행동을 분석하여 사용자 참여에 대한 인사이트를 제공할 수 있습니다.
   d. Enterprise Support 고객은 기술 지원 관리자에게 모니터링 전략 구축 워크숍을 요청할 수 있습니다. 이 워크숍은 워크로드를 관찰하는 전략을 수립하는 데 도움이 됩니다.
2. 애플리케이션을 모니터링하기 위한 가상 사용자 활동을 설정하세요. 가상 사용자 활동은 사용자 작업을 시뮬레이션하여 애플리케이션이 제대로 작동하는지 확인합니다.
   a. Amazon CloudWatch Synthetics는 Canary를 통해 사용자 활동을 시뮬레이션할 수 있습니다.

구현 계획의 작업 수준: 높음. 사용자 활동 원격 측정을 수집하기 위해 애플리케이션을 완전히 계측하려면 상당한 개발 노력이 필요할 수 있습니다.

리소스

관련 모범 사례:

- OPS04-BP01 애플리케이션 텔레메트리 구현 - 사용자 활동 원격 측정을 구축하려면 애플리케이션 원격 측정이 필요합니다.
- OPS04-BP02 워크로드 원격 측정 구현 및 구성 - 일부 사용자 활동 원격 측정은 워크로드 원격 측정으로도 간주될 수 있습니다.

관련 문서:

- 애플리케이션을 효과적으로 모니터링하는 방법

관련 동영상:

- AWS re:Invent 2020: Monitoring production services at Amazon(AWS re:Invent 2020: Amazon에서 프로덕션 서비스 모니터링)
- AWS re:Invent 2021 - Optimize applications through end user insights with Amazon CloudWatch RUM(AWS re:Invent 2021 - Amazon CloudWatch RUM을 통해 최종 사용자 인사이트로 애플리케이션 최적화)
- Testing and Monitoring APIs on AWS - AWS Online Tech Talks(AWS에서 API 테스트 및 모니터링 - AWS Online Tech Talks)

관련 예시:

- Amazon CloudWatch RUM 웹 클라이언트
- AWS Distro for Open Telemetry
- Implementing Real User Monitoring of Amplify Application using Amazon CloudWatch RUM(Amazon CloudWatch RUM을 사용하여 Amplify 애플리케이션의 실제 사용자 모니터링 구현)
- One Observability Workshop

관련 서비스:

- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- Amazon Pinpoint

OPS04-BP04 종속성 원격 측정 구현

워크로드가 사용하는 리소스의 상태 관련 정보를 내보내도록 워크로드를 설계 및 구성합니다. 이러한 리소스는 워크로드의 외부 리소스입니다. 외부 종속성의 예로는 외부 데이터베이스, DNS, 네트워크 연결 등이 있습니다. 이 정보를 사용하여 응답이 필요한 시기를 판단하고 워크로드 상태에 대한 추가 상황 정보를 제공합니다.

원하는 결과:

- 워크로드에서 외부 종속성의 상태에 대한 원격 측정을 전송합니다.
- 종속성이 비정상적인 경우 알림이 표시됩니다.

일반적인 안티 패턴:

- 사용자가 사이트에 연결할 수 없습니다. DNS 공급자가 작동하는지 확인하기 위해 수동으로 점검하지 않고는 DNS 문제가 사유인지 확인할 수 없습니다.
- 장바구니 애플리케이션에서 트랜잭션을 완료할 수 없습니다. 확인을 위해 연락하지 않고는 신용 카드 처리 공급자에게 문제가 있는지 확인할 수 없습니다.

이 모범 사례 확립의 이점:

- 외부 종속성을 모니터링하면 문제를 미리 알 수 있습니다.
- 종속성 상태를 알고 있으면 문제 해결에 도움이 됩니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

이해관계자와 협력하여 워크로드가 의존하는 외부 종속성을 식별합니다. 외부 종속성에는 외부 데이터베이스, API 또는 워크로드와 다른 환경의 리소스 간 네트워크 연결이 포함될 수 있습니다. 종속성 상태를 알리고 상태가 변경될 경우 사전에 알림을 제공하는 모니터링 전략을 개발합니다.

고객 사례

AnyCompany Retail의 전자 상거래 워크로드는 다른 환경에 위치한 데이터베이스에 의존합니다. 매일 밤, 전자 상거래 플랫폼에서 사용할 수 있도록 데이터가 데이터베이스에 채워집니다. 네트워크 연결 및 데이터베이스 지원은 다른 팀의 책임입니다. 전자 상거래 팀은 네트워크 연결이 끊기고, 데이터베이스에 연결할 수 없으며, 작업이 완료되지 않을 때 이를 알리기 위해 여러 Canary 알림을 구성했습니다.

구현 단계

1. 워크로드가 의존하는 외부 종속성을 식별합니다. 원격 측정을 구현하여 종속성의 상태 또는 연결 가능성을 추적합니다.
   a. AWS 고객은 AWS Health Dashboard를 사용하여 AWS 서비스 상태를 모니터링하고 상태 이벤트에 대한 알림을 받을 수 있습니다.
   b. Amazon CloudWatch Synthetics는 API, URL 및 웹 사이트 콘텐츠를 모니터링하는 데 사용할 수 있습니다.
2. 종속성이 비정상적이거나 연결할 수 없는 경우 조직에 알리도록 알림을 설정합니다.
   a. Enterprise Support 고객은 기술 지원 관리자에게 모니터링 전략 구축 워크숍을 요청할 수 있습니다. 이 워크숍은 워크로드를 관찰하는 전략을 수립하는 데 도움이 됩니다.

3. 종속성이 비정상적인 경우 종속성 담당자를 파악합니다. 종속성 소유자, 서비스 계약 및 에스컬레이션 프로세스를 문의하는 방법을 문서화합니다.

구현 계획의 작업 수준: 중간. 종속성 원격 측정을 구현하려면 사용자 지정 모니터링 솔루션을 개발해야 할 수 있습니다.

리소스

관련 모범 사례:

- OPS04-BP01 애플리케이션 텔레메트리 구현 - 애플리케이션 원격 측정에 종속성 모니터링을 구축할 수 있습니다.

관련 문서:

- Monitor your private internal endpoints 24x7 using CloudWatch Synthetics(CloudWatch Synthetics를 사용한 연중무휴 프라이빗 내부 엔드포인트 모니터링)

관련 동영상:

- AWS re:Invent 2018: Monitor All Your Things: Amazon CloudWatch in Action with BBC(AWS re:Invent 2018: 100% 모니터링: BBC와 함께 알아보는 Amazon CloudWatch 활용 사례)
- AWS re:Invent 2022 - Developing an observability strategy(AWS re:Invent 2022 - 관찰성 전략 개발)
- AWS re:Invent 2022 - Observability best practices at Amazon(AWS re:Invent 2022 - Amazon에서의 관찰성 모범 사례)

관련 예시:

- One Observability Workshop
- Well-Architected 실습 - 종속성 모니터링

관련 서비스:

- Amazon CloudWatch Synthetics
- AWS Health

OPS04-BP05 트랜잭션 추적 기능 구현

단일 논리적 작업의 결과로 트리거되고 워크로드의 다양한 경계에 걸쳐 통합되는 이벤트를 내보내도록 애플리케이션 코드를 구현하고 워크로드 구성 요소를 구성합니다. 맵을 생성하여 워크로드 및 서비스에서 트레이스가 어떻게 흐르는지 확인합니다. 구성 요소 간의 관계에 대한 인사이트를 얻고 문제를 식별하고 분석합니다. 수집된 정보를 사용하면 대응이 필요한 경우를 확인하고 문제의 원인을 파악할 수 있습니다.

원하는 결과:

- 워크로드 전체에서 트랜잭션 트레이스를 수집하여 구성 요소 간의 관계에 대한 인사이트를 얻습니다.
- 트랜잭션 및 이벤트가 워크로드에서 어떻게 흐르는지 더 잘 이해할 수 있도록 맵을 생성합니다.

일반적인 안티 패턴:

- 여러 계정에 걸쳐 서버리스 마이크로서비스 아키텍처를 구현했습니다. 고객에게 간헐적인 성능 문제가 있습니다. 트랜잭션 추적 기능이 없어 원인이 되는 기능이나 구성 요소를 파악할 수 없습니다.
- 워크로드에 성능 병목 현상이 발생합니다. 트랜잭션 추적 기능이 없어 애플리케이션 구성 요소 간의 관계를 확인하고 성능 병목 현상을 식별할 수 없습니다.
- 추적에 사용되는 식별자는 전역적으로 고유하지 않으므로 워크로드 동작을 분석할 때 추적 충돌이 발생합니다.

이 모범 사례 확립의 이점:

- 워크로드 전반의 트랜잭션 흐름을 파악하면 워크로드 트랜잭션의 예상 동작에 대한 인사이트를 얻을 수 있습니다.
- 예상 동작과 워크로드 전반의 예상 동작 변형을 파악할 수 있으므로 필요한 경우 대응할 수 있습니다.
- 생성된 위치와 관계없이 고유하게 생성된 식별자로 트랜잭션을 정확히 파악할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

시스템 구성 요소 간의 트랜잭션 흐름 관련 정보를 내보내도록 애플리케이션 및 워크로드를 설계합니다. 트랜잭션에 포함할 데이터는 전역적으로 고유한 트랜잭션 식별자, 트랜잭션 단계, 활성 구성 요소

및 활동 완료 시간입니다. 이 정보를 사용하여 진행 중인 활동과 완료된 활동, 그리고 완료된 활동의 결과를 확인합니다.

고객 사례

AnyCompany Retail에서 모든 트랜잭션에는 전역적으로 고유한 UUID가 생성됩니다. 이 UUID는 트랜잭션 중에 마이크로서비스 간에 전달됩니다. UUID는 사용자가 워크로드와 상호 작용할 때 트랜잭션 트레이스를 생성하는 데 사용됩니다. 워크로드 토폴로지의 맵은 트레이스와 함께 생성되며 워크로드 문제를 해결하고 성능을 개선하는 데 사용됩니다.

구현 단계

1. 워크로드에서 애플리케이션을 계측하여 트랜잭션 트레이스를 내보냅니다. 각 트랜잭션에 대한 고유 식별자를 생성하고 애플리케이션 간에 식별자를 전달하면 됩니다.
   a. AWS Distro for OpenTelemetry에서 자동 계측을 사용하여 애플리케이션 코드를 수정하지 않고 기존 애플리케이션에서 트레이스를 구현할 수 있습니다.
2. 애플리케이션 토폴로지의 맵을 생성합니다. 이러한 맵을 사용하면 성능을 개선하고 인사이트를 얻고 문제 해결에 도움을 받을 수 있습니다.
   a. AWS X-Ray는 워크로드에서 애플리케이션의 맵을 생성할 수 있습니다.

구현 계획의 작업 수준: 중간. 트랜잭션 트레이스를 구현하려면 어느 정도의 개발 작업이 필요할 수 있습니다.

리소스

관련 모범 사례:

- OPS04-BP01 애플리케이션 텔레메트리 구현 - 애플리케이션 텔레메트리를 먼저 구현해야 하며 여기에는 트랜잭션 추적 기능과 처리가 포함됩니다.

관련 문서:

- Discover application issues and get notifications with AWS X-Ray Insights(AWS X-Ray Insights로 애플리케이션 문제를 발견하고 알림 받기)
- How Wealthfront utilizes AWS X-Ray to analyze and debug distributed applications(Wealthfront가 AWS X-Ray를 활용하여 분산 애플리케이션을 분석하고 디버그하는 방법)
- New for AWS Distro for OpenTelemetry – Tracing Support is Now Generally Available(AWS Distro for OpenTelemetry의 새로운 기능 - 추적 지원 이제 정식 출시)

관련 동영상:

- AWS re:Invent 2018: Deep Dive into AWS X-Ray: Monitor Modern Applications (DEV324)(AWS re:Invent 2018: AWS X-Ray 심층 분석: 최신 애플리케이션 모니터링(DEV324))
- AWS re:Invent 2022 - Building observable applications with OpenTelemetry (BOA310)(AWS re:Invent 2022 - OpenTelemetry로 관측 가능한 애플리케이션 구축(BOA310))
- AWS re:Invent 2022 - Observability the open-source way(COP301-R)(AWS re:Invent 2022 - 오픈 소스 방식의 관측성(COP301-R))
- Capturing Trace Data with the AWS Distro for OpenTelemetry(AWS Distro for OpenTelemetry로 트레이스 데이터 캡처)
- Optimize Application Performance with AWS X-Ray(AWS X-Ray로 애플리케이션 성능 최적화)

관련 예시:

- AWS X-Ray 다중 API Gateway 추적 예

관련 서비스:

- AWS Distro for OpenTelemetry
- AWS X-Ray

OPS 5 귀사는 어떻게 결함을 줄이고 수정 작업을 쉽게 수행하고 프로덕션으로 이어지는 흐름을 개선하십니까?

프로덕션 환경으로 변경 사항을 전달하는 흐름을 개선할 수 있는 방식을 도입합니다. 이 방식은 리팩터링, 품질과 관련된 빠른 피드백 및 버그 수정을 지원해야 합니다. 이러한 방식을 도입하면 유용한 변경 사항을 프로덕션 환경으로 빠르게 전달할 수 있고, 문제 배포 가능성을 제한할 수 있으며, 배포 활동을 통해 발생하는 문제를 빠르게 파악하고 해결할 수 있습니다.

모범 사례

- OPS05-BP01 버전 관리 사용
- OPS05-BP02 변경 사항 테스트 및 확인
- OPS05-BP03 구성 관리 시스템 사용
- OPS05-BP04 빌드 및 배포 관리 시스템 사용
- OPS05-BP05 패치 관리 수행

- OPS05-BP06 설계 표준 공유
- OPS05-BP07 코드 품질 개선을 위한 사례 구현
- OPS05-BP08 여러 환경 사용
- OPS05-BP09 되돌릴 수 있는 작은 단위의 변경 내용 자주 적용
- OPS05-BP10 통합 및 배포 완전 자동화

OPS05-BP01 버전 관리 사용

버전 관리를 사용하면 변경 사항과 릴리스를 추적할 수 있습니다.

많은 AWS 서비스가 버전 관리 기능을 제공합니다. 수정본 또는 소스 제어 시스템(예: AWS CodeCommit)을 사용하여 코드 및 버전을 관리하는 인프라의 AWS CloudFormation 템플릿과 같은 기타 아티팩트를 관리합니다.

일반적인 안티 패턴:

- 워크스테이션에서 코드를 개발하고 저장해 왔습니다. 코드가 손실된 워크스테이션에서 복구할 수 없는 스토리지 장애가 발생했습니다.
- 기존 코드를 변경 사항으로 덮어쓴 후 애플리케이션을 다시 시작하면 애플리케이션이 더 이상 작동하지 않습니다. 변경 사항으로 되돌릴 수 없습니다.
- 다른 사람이 편집해야 하는 보고서 파일에 대한 쓰기 잠금이 있습니다. 태스크를 완료할 수 있도록 태스크 작업 중지를 요청하는 연락을 받습니다.
- 연구 팀은 향후 작업을 결정할 세부 분석을 수행해 왔습니다. 누군가 실수로 최종 보고서에 쇼핑 목록을 저장했습니다. 변경 사항을 되돌릴 수 없으며 보고서를 다시 생성해야 합니다.

이 모범 사례 정립의 이점: 버전 관리 기능을 사용하면 쉽게 알려진 정상 상태와 이전 버전으로 되돌리고 자산 손실 위험을 제한할 수 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 버전 관리 사용: 버전 제어 리포지토리에서 자산을 유지 관리합니다. 이렇게 하면 변경 사항을 추적하고, 새 버전을 배포하고, 기존 버전의 변경 사항을 감지하고, 장애 시 알려진 정상 상태로 롤백하는 등 이전 버전으로 되돌릴 수 있습니다. 구성 관리 시스템의 버전 제어 기능을 프로시저에 통합합니다.
  - AWS CodeCommit 소개

- AWS CodeCommit란 무엇입니까?

리소스

관련 문서:

- AWS CodeCommit란 무엇입니까?

관련 동영상:

- AWS CodeCommit 소개

OPS05-BP02 변경 사항 테스트 및 확인

프로덕션 환경에서 오류가 발생하지 않도록 배포된 모든 변경은 테스트해야 합니다. 이 모범 사례는 버전 관리에서부터 아티팩트 빌드까지 변경을 테스트하는 데 중점을 둡니다. 애플리케이션 코드 변경 외에도 테스트에는 인프라, 구성, 보안 제어 및 운영 절차를 포함해야 합니다. 테스트는 단위 테스트에서부터 소프트웨어 구성 요소 분석(SCA)에 이르기까지 형태가 다양합니다. 소프트웨어 통합 및 전달 프로세스에서 테스트를 좀 더 초기 단계에 수행하면 더 확실하게 아티팩트 품질이 향상됩니다.

조직에서는 모든 소프트웨어 아티팩트에 대한 테스트 표준을 개발해야 합니다. 자동화된 테스트는 수고를 덜고 테스트의 수작업 오류를 방지합니다. 경우에 따라 수동 테스트가 필요할 수 있습니다. 개발자는 소프트웨어 품질을 개선하는 피드백 루프를 생성할 수 있도록 자동화된 시험 결과에 액세스할 수 있어야 합니다.

원하는 결과:

- 모든 소프트웨어 변경 사항은 제공되기 전에 테스트됩니다.
- 개발자가 테스트 결과에 액세스할 수 있습니다.
- 조직에 모든 소프트웨어 변경에 적용되는 테스트 표준이 있습니다.

일반적인 안티 패턴:

- 아무런 테스트 없이 새로운 소프트웨어 변경 사항을 배포했습니다. 프로덕션 환경에서 실행에 실패하면 가동 중단으로 이어집니다.
- 새로운 보안 그룹이 프로덕션 전 환경에서 테스트 없이 AWS CloudFormation을 사용하여 배포됩니다. 보안 그룹이 고객이 앱에 연결할 수 없도록 합니다.

- 메서드가 수정되었으나 단위 테스트가 수행되지 않습니다. 소프트웨어가 프로덕션 환경에 배포되면 장애가 발생합니다.

이 모범 사례 확립의 이점:

- 소프트웨어 배포의 변경 실패율이 줄어듭니다.
- 소프트웨어 품질이 개선됩니다.
- 개발자가 코드의 가시성에 대한 인식을 높였습니다.
- 조직의 규정 준수를 지원한다는 확인을 가지고 보안 정책을 롤아웃할 수 있습니다.
- 트래픽 수요를 충족하기 위해 자동 크기 조정 정책 업데이트 등과 같은 인프라 변경을 사전에 테스트합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

지속적인 통합 방침의 일부로 애플리케이션 코드에서부터 인프라까지 모든 변경에 대해 테스트가 수행됩니다. 개발자가 빠르게 피드백을 얻을 수 있도록 테스트 결과가 게시됩니다. 조직에 모든 변경이 통과해야 하는 테스트 표준이 있습니다.

고객 사례

지속적 통합 파이프라인의 일부로, AnyCompany Retail에서는 모든 소프트웨어 아티팩트에 대해 여러 가지 유형의 테스트를 수행합니다. 테스트 기반 개발을 수행하기 때문에 모든 소프트웨어에 단위 테스트가 있습니다. 아티팩트가 구축되면 엔드 투 엔드 테스트를 실행합니다. 테스트의 1차 라운드가 완료된 후 알려진 취약점을 찾는 정적 애플리케이션 보안 검사를 실행합니다. 각 테스트 관문을 통과할 때마다 개발자에게 메시지가 전송됩니다. 모든 테스트가 완료되면 소프트웨어 아티팩트는 아티팩트 리포지토리에 저장됩니다.

구현 단계

1. 조직 내 이해관계자와 함께 소프트웨어 아티팩트를 위한 테스트 표준을 개발합니다. 모든 아티팩트가 어떤 표준 테스트를 통과해야 하나요? 테스트 범위에 포함해야 하는 규정 준수 또는 거버넌스 요구 사항이 있나요? 코드 품질 테스트를 수행해야 하나요? 테스트가 완료되면 누구에게 알려야 하나요?
   a. AWS 배포 파이프라인 참조 아키텍처에는 통합 파이프라인의 일부로 소프트웨어 아티팩트에 대해 수행할 수 있는 테스트 유형이 나열된 신뢰할 수 있는 목록이 포함되어 있습니다.

2. 소프트웨어 테스트 표준을 기준으로 필수 테스트를 통해 애플리케이션을 계측합니다. 각 테스트 세트는 10분 이내에 완료해야 합니다. 테스트는 통합 파이프라인의 일부로 실행되어야 합니다.
   a. Amazon CodeGuru Reviewer에서는 결함을 찾기 위해 애플리케이션 코드를 테스트할 수 있습니다.
   b. AWS CodeBuild를 사용하여 소프트웨어 아티팩트에 대한 테스트를 수행할 수 있습니다.
   c. AWS CodePipeline에서는 소프트웨어 테스트를 파이프라인으로 오케스트레이션할 수 있습니다.

리소스

관련 모범 사례:

- OPS05-BP01 버전 관리 사용 - 모든 소프트웨어 아티팩트는 버전이 관리되는 리포지토리를 기반으로 해야 합니다.
- OPS05-BP06 설계 표준 공유 - 조직의 소프트웨어 테스트 표준을 보고 디자인 표준을 알 수 있습니다.
- OPS05-BP10 통합 및 배포 완전 자동화 - 소프트웨어 테스트는 더 큰 통합 및 배포 파이프라인의 일부로 자동으로 실행되어야 합니다.

관련 문서:

- 테스트 기반 개발 접근 방식 채택
- TaskCat 및 CodePipeline으로 자동화된 AWS CloudFormation 테스트 파이프라인
- 오픈 소스 SCA, SAST 및 DAST 도구를 사용하여 엔드 투 엔드 AWS DevSecOps CI/CD 파이프라인 구축
- 서버리스 애플리케이션 테스트로 시작하기
- CI/CD 파이프라인이 릴리스에 매우 중요함
- AWS에서 지속적 통합 및 지속적 전달 사례 백서

관련 동영상:

- AWS re:Invent 2020: Testable infrastructure: Integration testing on AWS(AWS re:Invent 2020: 테스트 가능한 인프라: AWS에서의 통합 테스트)
- AWS Summit ANZ 2021 - Driving a test-first strategy with CDK and test driven development(Summit ANZ 2021 - CDK 및 테스트 기반 개발로 테스트 우선 전략 추진)

- Testing Your Infrastructure as Code with AWS CDK(AWS CDK를 사용하여 코드형 인프라 테스트)

관련 리소스:

- AWS 배포 파이프라인 참조 아키텍처 - 애플리케이션
- AWS Kubernetes DevSecOps 파이프라인
- 코드형 정책 워크숍 – 테스트 기반 개발
- AWS CodeBuild를 사용하여 GitHub에서 Node.js 애플리케이션에 대한 단위 테스트 실행
- 인프라 코드 테스트 기반 개발에 Serverspec 사용

관련 서비스:

- Amazon CodeGuru Reviewer
- AWS CodeBuild
- AWS CodePipeline

OPS05-BP03 구성 관리 시스템 사용

구성 관리 시스템을 사용하면 구성을 변경하고 변경 사항을 추적할 수 있습니다. 이러한 시스템에서는 수동 프로세스에서 발생하는 오류와 변경 사항 배포를 위한 작업량을 줄일 수 있습니다.

정적 구성 관리는 리소스를 초기화할 때 리소스 수명 주기 전체에 걸쳐 일관성을 유지할 것으로 예상되는 값을 설정합니다. 예를 들어, 인스턴스의 웹 또는 애플리케이션 서버의 구성을 설정하거나 AWS 서비스 구성을 정의하는 작업이 포함됩니다(AWS Management Console 내에서 또는 AWS CLI를 통해).

동적 구성 관리는 초기화 시 리소스 수명 주기 동안 변경될 수 있거나 변경될 것으로 예상되는 값을 설정합니다. 예를 들어, 구성 변경을 통해 코드의 기능을 활성화하도록 기능 전환을 설정하거나, 인시던트 중에 로그 세부 정보 수준을 변경하여 더 많은 데이터를 캡처하고 나서 인시던트 이후에 다시 변경함으로써 불필요한 로그와 관련 비용이 발생하지 않도록 할 수 있습니다.

인스턴스, 컨테이너, 서버리스 기능 또는 디바이스에서 실행 중인 애플리케이션에 동적 구성을 적용한 경우 AWS AppConfig 를 사용하여 환경 전반에 걸쳐 이를 관리하고 배포할 수 있습니다.

AWS에서는 AWS Config 를 사용하여 계정과 리전 전체에서 AWS 리소스 구성을 계속해서 모니터링할 수 있습니다. 이를 통해 구성 이력을 추적하고, 구성 변경이 다른 리소스에 어떤 영향을 미치는지 이해하며, AWS Config 규칙 및 AWS Config 규정 준수 팩을 사용하여 예상되거나 원하는 구성을 기준으로 감사할 수 있습니다.

AWS에서는 AWS 개발자 도구 (예: AWS CodeCommit, AWS CodeBuild, AWS CodePipeline, AWS CodeDeploy및 AWS CodeStar)와 같은 서비스를 사용하여 CI/CD(지속적 통합/지속적 전달) 파이프라인을 구축할 수 있습니다.

변경 구현에 영향을 줄 수 있는 중요한 비즈니스나 운영 활동 또는 이벤트가 계획된 경우 변경 일정을 정하고 이를 추적합니다. 활동을 조정하여 이러한 계획에 관한 위험을 관리합니다. AWS Systems Manager Change Calendar 에서는 변경의 개시 또는 종료에 해당하는 시간 블록과 변경 이유를 문서화하고, 다른 AWS 계정과 이 정보를 공유 할 수 있습니다(기타 AWS 계정 대상). AWS Systems Manager Automation 스크립트는 변경 일정 상태를 준수하도록 구성할 수 있습니다.

AWS Systems Manager Maintenance Windows 를 사용하여 지정된 시간에 AWS SSM Run Command 또는 Automation 스크립트, AWS Lambda 호출 또는 AWS Step Functions 활동 수행을 예약할 수 있습니다. 이러한 활동이 평가에 포함될 수 있도록 변경 일정에서 활동을 표시합니다.

일반적인 안티 패턴:

- 플릿 전체에서 웹 서버 구성을 수동으로 업데이트하면 업데이트 오류로 인해 여러 서버가 응답하지 않게 됩니다.
- 여러 시간 동안 애플리케이션 서버 플릿을 수동으로 업데이트합니다. 변경 중 구성 불일치로 인해 예기치 않은 동작이 발생합니다.
- 누군가가 보안 그룹을 업데이트했으며 웹 서버에 더 이상 액세스할 수 없습니다. 변경된 사항을 알지 못하면 문제를 조사하는 데 상당한 시간이 들어서 복구 시간이 늘어납니다.

이 모범 사례 수립의 이점: 구성 관리 시스템을 도입하면 변경 수행 및 추적을 위한 작업량과 수동 절차로 인한 오류 발생 빈도가 줄어듭니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 구성 관리 시스템 사용: 구성 관리 시스템을 사용하면 변경 사항을 추적/구현하고, 수동 프로세스로 인해 발생하는 오류를 줄이고, 작업량을 줄일 수 있습니다.
  - 인프라 구성 관리
  - AWS Config
  - AWS Config란 무엇입니까?
  - AWS CloudFormation 소개
  - AWS CloudFormation이란 무엇인가요?
  - AWS OpsWorks

- AWS OpsWorks란 무엇입니까?
- AWS Elastic Beanstalk 소개
- AWS Elastic Beanstalk란 무엇입니까?

리소스

관련 문서:

- AWS AppConfig
- AWS 개발자 도구
- AWS OpsWorks
- AWS Systems Manager Change Calendar
- AWS Systems Manager Maintenance Windows
- 인프라 구성 관리
- AWS CloudFormation이란 무엇인가요?
- AWS Config란 무엇입니까?
- AWS Elastic Beanstalk란 무엇입니까?
- AWS OpsWorks란 무엇입니까?

관련 동영상:

- AWS CloudFormation 소개
- AWS Elastic Beanstalk 소개

OPS05-BP04 빌드 및 배포 관리 시스템 사용

빌드 및 배포 관리 시스템을 사용합니다. 이러한 시스템에서는 수동 프로세스에서 발생하는 오류와 변경 사항 배포를 위한 작업량을 줄일 수 있습니다.

AWS에서는 다음과 같은 서비스를 사용하여 지속적 통합 및 지속적 배포(CI/CD) 파이프라인을 구축할 수 있습니다. AWS 개발자 도구 (예: AWS CodeCommit, AWS CodeBuild, AWS CodePipeline, AWS CodeDeploy 및 AWS CodeStar)는 SAP NetWeaver Guide Finder 및 SAP NetWeaver Security Guide를 참조하세요.

일반적인 안티 패턴:

- 개발 시스템에서 코드를 컴파일한 후 실행 파일을 프로덕션 시스템에 복사하면 실행 파일이 시작되지 않습니다. 로컬 로그 파일은 누락된 종속성으로 인해 실패했음을 나타냅니다.
- 개발 환경에서 새로운 기능을 사용하여 애플리케이션을 성공적으로 빌드하고 코드를 품질 보증(QA) 팀에 제공합니다. 정적 자산이 누락되어 QA에 실패합니다.
- 금요일에는 많은 노력을 기울이고 새로 코딩된 기능을 포함하여 개발 환경에서 수동으로 애플리케이션을 성공적으로 빌드했습니다. 월요일에는 애플리케이션을 성공적으로 빌드할 수 있는 단계를 반복할 수 없습니다.
- 새 릴리스에 대해 생성한 테스트를 수행합니다. 그리고 다음 주에 테스트 환경을 설정하고 모든 기존 통합 테스트를 수행한 후 성능 테스트를 수행합니다. 새 코드는 용인할 수 없는 성능 영향을 미치므로 재개발한 후 다시 테스트해야 합니다.

이 모범 사례 수립의 이점: 빌드 및 배포 활동을 관리하는 메커니즘을 제공하여 반복적인 작업 수행을 위한 작업량을 줄이고, 팀원들이 고가치 창조 작업에 집중할 수 있게 하고, 수동 절차에서 발생하는 오류의 도입을 제한할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 빌드 및 배포 관리 시스템 사용: 빌드 및 배포 관리 시스템을 사용하면 변경 사항을 추적/구현하고, 수동 프로세스로 인해 발생하는 오류와 작업량을 줄일 수 있습니다. 코드 체크 인에서 빌드, 테스트, 배포 및 확인까지의 전체 통합 및 배포 파이프라인을 완전히 자동화합니다. 이렇게 하면 리드 시간을 단축하고 변경을 더 자주 수행할 수 있으며 작업량을 줄일 수 있습니다.
  - AWS CodeBuild란 무엇입니까?
  - 소프트웨어 개발을 위한 지속적 통합 모범 사례
  - Slalom: AWS의 서버리스 애플리케이션용 CI/CD
  - AWS CodeDeploy 소개 - Amazon Web Services를 통해 자동화된 소프트웨어 배포
  - AWS CodeDeploy란 무엇입니까?

리소스

관련 문서:

- AWS 개발자 도구
- AWS CodeBuild란 무엇입니까?
- AWS CodeDeploy란 무엇입니까?

관련 동영상:

- 소프트웨어 개발을 위한 지속적 통합 모범 사례
- AWS CodeDeploy 소개 - Amazon Web Services를 통해 자동화된 소프트웨어 배포
- Slalom: AWS의 서버리스 애플리케이션용 CI/CD

OPS05-BP05 패치 관리 수행

패치 관리를 수행하면 기능을 확인하고, 문제를 해결하고, 거버넌스 규정 준수 상태를 유지할 수 있습니다. 그리고 패치 관리를 자동화하면 수동 프로세스에서 발생하는 오류와 패치를 위한 작업량을 줄일 수 있습니다.

패치 및 취약성 관리는 이점 관리 및 위험 관리 활동의 일부입니다. 변경이 불가능한 인프라를 보유하고 검증된 정상 상태의 워크로드를 배포하는 것이 좋습니다. 이 방식을 실현할 수 없으면 남은 방법은 패치를 적용하는 것입니다.

취약성을 없애기 위해 머신 이미지, 컨테이너 이미지 또는 Lambda 사용자 지정 런타임 및 추가 라이브러리 를 업데이트하는 것도 패치 관리에 속합니다. Linux 또는 Windows Server용 Amazon Machine Image 다음을 사용하여 (AMI)에 대한 업데이트를 관리해야 합니다. EC2 Image Builder. 기존 파이프라인과 함께 Amazon Elastic Container Registry 를 사용하여 Amazon ECS 이미지 및 Amazon EKS 이미지를 관리할 수 있습니다. AWS Lambda에는 버전 관리 기능이 있습니다.

패치는 먼저 안전한 환경에서 테스트를 거치지 않고는 프로덕션 환경에서 수행해서는 안 됩니다. 패치는 운영 또는 비즈니스 성과를 지원하는 경우에만 적용해야 합니다. AWS에서는 AWS Systems Manager Patch Manager 를 사용하여 관리형 시스템에 패치를 적용하는 프로세스를 자동화하고 다음을 사용하여 이 활동을 예약할 수 있습니다. AWS Systems Manager Maintenance Windows.

일반적인 안티 패턴:

- 2시간 내에 최신 보안 패치를 모두 적용해야 하는데 애플리케이션과 패치가 호환되지 않아 여러 번 중단될 수 있습니다.
- 패치가 적용되지 않은 라이브러리는 알 수 없는 당사자가 워크로드에 액세스하기 위해 해당 라이브러리의 취약성을 이용하므로 의도하지 않은 결과를 초래합니다.
- 개발자에게 알리지 않고 개발자 환경에 자동으로 패치를 적용합니다. 개발자가 환경이 예상대로 작동하지 않는다는 불만을 여러 번 제기합니다.
- 영구 인스턴스에서 자체 상용 소프트웨어에 패치를 적용하지 않았습니다. 소프트웨어에 문제가 있어서 공급자에게 문의하면 해당 버전이 지원되지 않으며 지원을 받으려면 특정 수준으로 패치해야 한다는 답을 듣습니다.

- 사용한 암호화 소프트웨어에 대해 최근에 릴리스된 패치의 성능이 크게 향상되었습니다. 패치가 적용되지 않은 시스템에 성능 문제가 있습니다.

이 모범 사례 정립의 이점: 패치 적용 기준 및 환경 전체에 배포를 위한 방법론을 포함하여 패치 관리 프로세스를 설정하면 이러한 프로세스의 이점을 실현하고 그 영향을 제어할 수 있습니다. 이를 통해 원하는 기능을 도입하고, 문제를 제거하고, 거버넌스를 지속적으로 준수할 수 있습니다. 패치 관리 시스템 및 자동화를 구현하여 패치 배포를 위한 작업량을 줄이고 수동 프로세스로 인한 오류를 제한합니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 패치 관리: 원하는 기능을 생성하고 거버넌스 정책과 공급업체 지원 요구 사항을 준수하는 상태를 유지할 수 있도록 시스템에 패치를 적용하여 문제를 해결합니다. 변경 불가능한 시스템에서는 원하는 결과를 달성할 수 있도록 설정된 적절한 패치를 배포합니다. 패치 관리 메커니즘을 자동화하면 패치에 걸리는 시간, 수동 프로세스에서 발생하는 오류 및 패치를 위한 작업량을 줄일 수 있습니다.
  - AWS Systems Manager Patch Manager

리소스

관련 문서:

- AWS 개발자 도구
- AWS Systems Manager Patch Manager

관련 동영상:

- AWS의 서버리스 애플리케이션용 CI/CD
- 운영 설계를 염두에 두세요

관련 예시:

- Well-Architected 랩 - 인벤토리 및 패치 관리

OPS05-BP06 설계 표준 공유

여러 팀이 모범 사례를 공유하면 표준에 대한 인지도를 높이고 개발 작업의 이점을 극대화할 수 있습니다. 아키텍처가 변경됨에 따라 표준을 문서화하고 최신 상태를 유지합니다. 조직에 공유 표준이 적용되

면 표준에 대한 추가, 변경 및 예외 처리를 요청하는 메커니즘을 확보해야 합니다. 이 옵션이 없으면 표준이 혁신의 제약 요인이 됩니다.

원하는 결과:

- 설계 표준이 조직 내 팀 전반에 공유됩니다.
- 모범 사례의 개선에 따라 표준이 문서화되고 최신 상태로 유지됩니다.

일반적인 안티 패턴:

- 두 개발 팀이 각각 사용자 인증 서비스를 만들었습니다. 사용자는 액세스하려는 시스템의 각 부분에 대해 별도의 자격 증명 세트를 유지해야 합니다.
- 각 팀은 자체 보유 인프라를 관리합니다. 새로운 규정 준수 요구 사항으로 인해 인프라를 변경해야 하며 각 팀은 이를 다른 방식으로 구현합니다.

이 모범 사례 확립의 이점:

- 공유 표준을 사용하여 모범 사례 도입을 지원하고 개발 작업의 이점을 극대화합니다.
- 설계 표준을 문서화하고 업데이트하면 조직에서 모범 사례와 보안 및 규정 준수 요구 사항을 최신 상태로 유지할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

팀 간에 기존 모범 사례, 설계 표준, 체크리스트, 운영 절차, 지침 및 거버넌스 요구 사항을 공유합니다. 개선 및 혁신을 지원하기 위해 설계 표준에 대한 변경 사항, 추가 및 예외를 요청할 절차를 마련합니다. 팀이 게시된 콘텐츠를 알게 합니다. 새로운 모범 사례가 나타남에 따라 설계 표준을 최신 상태로 유지하는 메커니즘을 확보합니다.

고객 사례

AnyCompany Retail에는 소프트웨어 아키텍처 패턴을 생성하는 다기능 아키텍처 팀이 있습니다. 이 팀은 규정 준수 및 거버넌스가 기본으로 포함된 아키텍처를 구축합니다. 이러한 공유 표준을 도입하는 팀은 기본으로 포함된 규정 준수 및 거버넌스의 이점을 활용할 수 있습니다. 설계 표준을 기반으로 신속하게 구축할 수 있습니다. 아키텍처 팀은 분기별로 만나 아키텍처 패턴을 평가하고 필요한 경우 업데이트합니다.

구현 단계

1. 설계 표준 개발 및 업데이트를 담당할 다기능 팀을 식별합니다. 이 팀은 조직 전체의 이해 관계자와 협력하여 설계 표준, 운영 절차, 체크리스트, 지침 및 거버넌스 요구 사항을 개발합니다. 설계 표준을 문서화하고 조직 내에서 공유합니다.
   a. AWS Service Catalog는 코드형 인프라를 사용하여 설계 표준을 나타내는 포트폴리오를 생성하는 데 사용할 수 있습니다. 계정 간에 포트폴리오를 공유할 수 있습니다.
2. 새로운 모범 사례가 식별되면 설계 표준을 최신 상태로 유지할 수 있는 메커니즘을 확보합니다.
3. 설계 표준을 중앙 집중식으로 적용되는 경우 변경, 업데이트 및 면제를 요청하는 프로세스를 마련합니다.

구현 계획의 작업 수준: 중간. 설계 표준을 만들고 공유하는 프로세스를 개발하려면 조직 전반의 이해 관계자와 조율하고 협력해야 합니다.

리소스

관련 모범 사례:

- OPS01-BP03 거버넌스 요구 사항 평가 - 거버넌스 요구 사항은 설계 표준에 영향을 미칩니다.
- OPS01-BP04 규정 준수 요구 사항 평가 - 규정 준수는 설계 표준을 만드는 데 중요한 요소입니다.
- OPS07-BP02 일관된 방식으로 운영 준비 상태 검토 - 운영 준비 상태 체크리스트는 워크로드를 설계할 때 설계 표준을 구현하는 메커니즘입니다.
- OPS11-BP01 지속적인 개선을 위한 프로세스 마련 - 설계 표준 업데이트는 지속적인 개선의 일부입니다.
- OPS11-BP04 지식 관리 수행 - 지식 관리 방침의 일부로 설계 표준을 문서화하고 공유합니다.

관련 문서:

- Automate AWS Backups with AWS Service Catalog(AWS Service Catalog로 AWS 백업 자동화)
- AWS Service Catalog Account Factory-Enhanced(AWS Service Catalog Account Factory 개선)
- How Expedia Group built Database as a Service (DBaaS) offering using AWS Service Catalog(Expedia Group에서 AWS Service Catalog를 사용하여 DBaaS(Database as a Service) 제품을 구축한 방법)
- Maintain visibility over the use of cloud architecture patterns(클라우드 아키텍처 패턴 사용에 대한 가시성 유지)
- Simplify sharing your AWS Service Catalog portfolios in an AWS Organizations setup(AWS Organizations 설정에서 AWS Service Catalog 포트폴리오 공유 간소화)

관련 동영상:

- AWS Service Catalog – Getting Started(AWS Service Catalog – 시작하기)
- AWS re:Invent 2020: Manage your AWS Service Catalog portfolios like an expert(AWS re:Invent 2020: 전문가처럼 AWS Service Catalog 포트폴리오 관리)

관련 예시:

- AWS Service Catalog 참조 아키텍처
- AWS Service Catalog 워크숍

관련 서비스:

- AWS Service Catalog

OPS05-BP07 코드 품질 개선을 위한 사례 구현

코드 품질을 개선하고 결함을 최소화하는 사례를 구현합니다. 테스트 기반 개발, 코드 검토, 표준 도입 및 페어 프로그래밍 등을 몇 가지 예로 들 수 있습니다. 이러한 사례를 지속적 통합 및 전달 프로세스에 통합합니다.

원하는 결과:

- 조직에서는 코드 검토 또는 페어 프로그래밍과 같은 모범 사례를 사용하여 코드 품질을 개선합니다.
- 개발자와 운영자는 소프트웨어 개발 수명 주기의 일부로 코드 품질 모범 사례를 채택합니다.

일반적인 안티 패턴:

- 코드 검토 없이 애플리케이션의 기본 분기에 코드를 커밋합니다. 변경 사항은 프로덕션에 자동으로 배포되고 중단이 발생합니다.
- 단위, 엔드 투 엔드 또는 통합 테스트 없이 새 애플리케이션을 개발합니다. 배포 전에 애플리케이션을 테스트할 방법이 없습니다.
- 팀은 결함을 해결하기 위해 프로덕션에서 수동으로 변경합니다. 변경 사항은 테스트 또는 코드 검토 단계를 거치지 않으며 지속적 통합 및 전달 프로세스를 통해 캡처되거나 기록되지 않습니다.

이 모범 사례 확립의 이점:

- 코드 품질 개선을 위한 사례를 도입하면 프로덕션에서 발생하는 문제를 최소화할 수 있습니다.
- 페어 프로그래밍 및 코드 검토와 같은 모범 사례를 사용하면 코드 품질이 향상됩니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

배포되기 전에 결함을 최소화하기 위해 코드 품질을 개선하는 사례를 구현합니다. 테스트 기반 개발, 코드 검토, 페어 프로그래밍과 같은 방법을 사용하여 개발 품질을 높입니다.

고객 사례

AnyCompany Retail은 코드 품질을 개선하기 위해 몇 가지 사례를 채택합니다. 전에는 애플리케이션 작성을 위한 표준으로 테스트 기반 개발 방식을 채택했습니다. 일부 새로운 기능의 경우 개발자가 스프린트 중에 페어 프로그래밍을 하도록 합니다. 모든 풀 요청은 통합 및 배포되기 전에 책임 개발자가 코드를 검토합니다.

구현 단계

1. 테스트 기반 개발, 코드 검토, 페어 프로그래밍과 같은 코드 품질 관련 사례를 지속적 통합 및 전달 프로세스에 도입합니다. 이러한 기술을 사용하여 소프트웨어 품질을 개선합니다.
   a. Amazon CodeGuru Reviewer는 기계 학습을 사용하여 Java 및 Python 코드에 대한 프로그래밍 권장 사항을 제공하면 됩니다.
   b. 코드 개발에 대해 협업할 수 있는 공유 개발 환경은 AWS Cloud9으로 생성할 수 있습니다.

구현 계획의 작업 수준: 중간. 이 모범 사례를 구현하는 방법에는 여러 가지가 있지만 조직에서 채택하는 것은 어려울 수 있습니다.

리소스

관련 모범 사례:

- OPS05-BP06 설계 표준 공유 - 코드 품질 관련 사례의 일부로 설계 표준을 공유할 수 있습니다.

관련 문서:

- Agile Software Guide(애자일 소프트웨어 가이드)
- CI/CD 파이프라인이 릴리스에 매우 중요함

- Automate code reviews with Amazon CodeGuru Reviewer(Amazon CodeGuru Reviewer를 사용한 코드 검토 자동화)
- 테스트 기반 개발 접근 방식 채택
- How DevFactory builds better applications with Amazon CodeGuru(DevFactory가 Amazon CodeGuru를 사용하여 더 나은 애플리케이션을 구축하는 방법)
- On Pair Programming (페어 프로그래밍 사용)
- RENGA Inc. automates code reviews with Amazon CodeGuru(RENGA Inc., Amazon CodeGuru로 코드 검토 자동화)
- The Art of Agile Development: Test-Driven Development(애자일 개발 기술: 테스트 기반 개발)
- Why code reviews matter (and actually save time!)(코드 검토가 중요한 이유(그리고 시간이 절약되는 이유))

관련 동영상:

- AWS re:Invent 2020: Continuous improvement of code quality with Amazon CodeGuru(AWS re:Invent 2020: Amazon CodeGuru를 통한 코드 품질의 지속적인 개선)
- AWS Summit ANZ 2021 - Driving a test-first strategy with CDK and test driven development(Summit ANZ 2021 - CDK 및 테스트 기반 개발로 테스트 우선 전략 추진)

관련 서비스:

- Amazon CodeGuru Reviewer
- Amazon CodeGuru Profiler
- AWS Cloud9

OPS05-BP08 여러 환경 사용

여러 환경을 사용하여 워크로드를 실험, 개발 및 테스트합니다. 프로덕션 환경에 배포하는 단계에 가까워질수록 제어 수준을 높이면 배포되었을 때 워크로드가 의도한 대로 작동할 것이라는 신뢰성을 높일 수 있습니다.

일반적인 안티 패턴:

- 공유 개발 환경에서 개발을 수행하고 있으며 다른 개발자가 코드 변경 사항을 덮어씁니다.
- 공유 개발 환경에 대한 제한적인 보안 제어로 인해 새로운 서비스와 기능을 실험할 수 없습니다.

- 프로덕션 시스템에서 로드 테스트를 수행하고 사용자를 중단시킵니다.
- 프로덕션 환경에서 데이터 손실을 일으키는 심각한 오류가 발생했습니다. 데이터 손실이 어떻게 발생했는지 파악하고 다시 발생하지 않도록 프로덕션 환경에서 데이터 손실을 일으키는 조건을 재현하려고 합니다. 테스트 중 추가 데이터 손실을 방지하기 위해 사용자가 애플리케이션을 이용할 수 없도록 해야 합니다.
- 멀티 테넌트 서비스를 운영 중이며 전용 환경에 대한 고객 요청을 지원할 수 없습니다.
- 항상 테스트하지 못할 수 있지만, 테스트할 때는 프로덕션 환경에서 해야 합니다.
- 단일 환경의 단순성이 환경 내 변경 사항의 영향 범위를 재정의합니다.

이 모범 사례 정립의 이점: 여러 환경을 배포하여 개발자 또는 사용자 커뮤니티 간에 충돌을 일으키지 않고 여러 동시 개발, 테스트 및 프로덕션 환경을 지원할 수 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 여러 환경 사용: 실험이 가능한 최소한의 제어 기능이 있는 샌드박스 환경을 개발자에게 제공합니다. 개별 개발 환경을 제공하면 병렬 작업이 가능하므로 개발을 더 빠르게 진행할 수 있습니다. 프로덕션 환경과 인접한 환경에는 더욱 엄격한 제어 기능을 구현하여 개발자가 혁신을 이룰 수 있도록 합니다. 코드형 인프라 및 구성 관리 시스템을 사용하여 프로덕션 환경의 제어 기능과 일치하는 방식으로 구성된 환경을 배포합니다. 그러면 배포된 시스템이 정상적으로 작동합니다. 사용되고 있지 않은 환경은 유휴 리소스 관련 비용이 발생하지 않도록 해제합니다. 예를 들어 개발 시스템은 야간 시간과 주말에 해제합니다. 로드 테스트 시에는 올바른 결과를 얻을 수 있도록 프로덕션 환경과 동일한 환경을 배포합니다.
  - AWS CloudFormation란 무엇입니까?
  - AWS Lambda를 사용하여 Amazon EC2 인스턴스를 정기적으로 중지 및 시작하려면 어떻게 해야 합니까?

리소스

관련 문서:

- AWS Lambda를 사용하여 Amazon EC2 인스턴스를 정기적으로 중지 및 시작하려면 어떻게 해야 합니까?
- AWS CloudFormation란 무엇입니까?

OPS05-BP09 되돌릴 수 있는 작은 단위의 변경 내용 자주 적용

되돌릴 수 있는 소규모 변경 작업을 자주 수행하면 변경의 영향과 범위가 감소합니다. 이렇게 하면 문제를 더 빠르고 쉽게 해결할 수 있으며 변경 사항 롤백 옵션을 사용할 수 있습니다.

일반적인 안티 패턴:

- 분기별로 애플리케이션의 새 버전을 배포합니다.
- 데이터베이스 스키마를 자주 변경합니다.
- 수동 인 플레이스 업데이트를 수행하여 기존 설치 및 구성을 덮어씁니다.

이 모범 사례 수립의 이점: 작은 변경 사항을 자주 배포하여 개발 작업의 이점을 더 빠르게 파악합니다. 변경 사항이 작으면 의도하지 않은 결과가 있는지 파악하기가 훨씬 더 쉽습니다. 변경 사항을 되돌릴 수 있는 경우 복구가 간소화됨에 따라 변경 사항을 구현할 위험이 적습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 되돌릴 수 있는 작은 단위의 변경 내용 자주 적용: 되돌릴 수 있는 작은 단위의 변경 내용을 자주 적용하면 변경의 범위와 그로 인한 영향을 줄일 수 있습니다. 이렇게 하면 문제를 더 빠르고 쉽게 해결할 수 있으며 변경 사항 롤백 옵션을 사용할 수 있습니다. 또한 업무에 유용한 기능을 더 빠르게 제공할 수 있습니다.

OPS05-BP10 통합 및 배포 완전 자동화

워크로드 빌드, 배포 및 테스트를 자동화합니다. 이렇게 하면 수동 프로세스에서 발생하는 오류와 변경 사항 배포를 위한 작업을 줄일 수 있습니다.

메타데이터를 리소스 태그 및 AWS Resource Groups 을 통해 적용하고, 일관된 태깅 전략 을 시행하면 리소스를 식별할 수 있습니다. 리소스에 조직, 비용 회계, 액세스 제어에 대한 리소스에 태그를 지정하여 자동화된 운영 활동을 실행할 대상을 설정합니다.

일반적인 안티 패턴:

- 금요일에는 기능 브랜치에 대한 새 코드 작성을 마칩니다. 월요일에는 코드 품질 테스트 스크립트와 각 단위 테스트 스크립트를 실행한 후 예정된 다음 릴리스를 위해 코드를 체크인합니다.

- 프로덕션 환경에서 많은 고객에게 영향을 미치는 중요한 문제에 대한 수정을 코딩해야 합니다. 수정 사항을 테스트한 후 코드 및 이메일 변경 관리를 커밋하여 프로덕션에 배포하기 위한 승인을 요청합니다.

이 모범 사례 수립의 이점: 자동화된 빌드 및 배포 관리 시스템을 구현하면 수동 프로세스로 인한 오류와 변경 사항 배포를 위한 작업이 줄어 팀원이 비즈니스 가치를 제공하는 데 집중할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 빌드 및 배포 관리 시스템 사용: 빌드 및 배포 관리 시스템을 사용하면 변경 사항을 추적/구현하고, 수동 프로세스로 인해 발생하는 오류와 작업량을 줄일 수 있습니다. 코드 체크 인에서 빌드, 테스트, 배포 및 확인까지의 전체 통합 및 배포 파이프라인을 완전히 자동화합니다. 이렇게 하면 리드 시간을 단축하고 변경을 더 자주 수행할 수 있으며 작업량을 줄일 수 있습니다.
  - AWS CodeBuild란 무엇입니까?
  - 소프트웨어 개발을 위한 지속적 통합 모범 사례
  - Slalom: AWS의 서버리스 애플리케이션용 CI/CD
  - AWS CodeDeploy 소개 - Amazon Web Services를 통해 자동화된 소프트웨어 배포
  - AWS CodeDeploy란 무엇입니까?

리소스

관련 문서:

- AWS CodeBuild란 무엇입니까?
- AWS CodeDeploy란 무엇입니까?

관련 동영상:

- 소프트웨어 개발을 위한 지속적 통합 모범 사례
- AWS CodeDeploy 소개 - Amazon Web Services를 통해 자동화된 소프트웨어 배포
- Slalom: AWS의 서버리스 애플리케이션용 CI/CD

OPS 6 배포 위험을 어떻게 최소화하고 있습니까?

품질과 관련한 피드백을 빠르게 제공하며, 적절한 성과를 달성하는 데 도움이 되지 않는 변경을 수행한 경우 신속하게 복구할 수 있는 방식을 도입합니다. 이러한 사례를 사용하면 변경 사항 배포로 인해 발생하는 문제의 영향을 완화할 수 있습니다.

모범 사례

- OPS06-BP01 변경이 적절하지 못한 경우에 대한 계획 수립
- OPS06-BP02 변경 사항 테스트 및 확인
- OPS06-BP03 배포 관리 시스템 사용
- OPS06-BP04 제한된 배포를 사용하여 테스트
- OPS06-BP05 병렬 환경을 사용하여 배포
- OPS06-BP06 작게 자주 발생하고 되돌릴 수 있는 변경 내용 배포
- OPS06-BP07 통합 및 배포 완전 자동화
- OPS06-BP08 테스트 및 롤백 자동화

OPS06-BP01 변경이 적절하지 못한 경우에 대한 계획 수립

변경을 수행했는데 적절한 성과를 달성하는 데 도움이 되지 않는다면 알려진 정상 상태로 되돌릴 수 있는 계획을 세우거나 프로덕션 환경에서 관련 문제를 해결합니다. 이와 같이 준비를 하면 문제에 더욱 신속하게 대응함으로써 복구 시간을 단축할 수 있습니다.

일반적인 안티 패턴:

- 배포를 수행했으며 애플리케이션이 불안정해졌지만 시스템에 활성 사용자가 있는 것 같습니다. 변경 사항을 롤백하고 활성 사용자에게 영향을 줄 것인지 아니면 사용자에게 영향을 줄 수 있음을 알고 변경 사항을 롤백할 때까지 기다려야 합니다.
- 라우팅을 변경한 후에는 새 환경에 액세스할 수 있지만, 서브넷 중 하나에 연결할 수 없게 됩니다. 모든 것을 롤백할지 아니면 액세스할 수 없는 서브넷을 수정할지 결정해야 합니다. 이러한 결정을 내리는 동안 서브넷에는 계속 연결할 수 없습니다.

이 모범 사례 수립의 이점: 계획을 수립하면 변경에 실패하여 발생하는 평균 복구 시간(MTTR)이 줄어들어 최종 사용자에게 미치는 영향을 완화할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 변경이 적절하지 못한 경우에 대한 계획 수립: 변경을 수행했는데 적절한 성과를 달성하는 데 도움이 되지 않는다면 알려진 정상 상태로 되돌릴 수 있는(변경 사항 롤백) 계획을 세우거나 프로덕션 환경에서 관련 문제를 해결합니다(변경 사항 롤포워드). 배포 실패 시 롤백할 수 없는 변경 사항이 확인되면 해당 변경 사항을 커밋하기 전에 적절한 판단에 따라 조치를 결정합니다.

OPS06-BP02 변경 사항 테스트 및 확인

수명 주기의 모든 단계에서 변경 사항을 테스트하고 결과를 검증하여 새 기능을 확인하고 배포 실패의 영향과 위험을 최소화합니다.

AWS에서는 실험과 테스트의 위험, 작업량 및 비용을 줄일 수 있는 임시 병렬 환경을 생성할 수 있습니다. 이러한 환경의 배포를 AWS CloudFormation 을 사용하여 자동화하면 임시 환경을 일관되게 구현할 수 있습니다.

일반적인 안티 패턴:

- 애플리케이션에 새로운 기능을 배포합니다. 기능이 작동하지 않습니다. 모릅니다.
- 인증서를 업데이트합니다. 잘못된 구성 요소에 실수로 인증서를 설치합니다. 모릅니다.

이 모범 사례 정립의 이점: 배포 후 변경 사항을 테스트하고 확인하면 문제를 조기에 식별하여 고객에게 미치는 영향을 완화할 수 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 변경 사항 테스트 및 확인: 개발, 테스트, 프로덕션 등 수명 주기의 모든 단계에서 변경 사항을 테스트하고 결과를 검증하여 새 기능을 확인하고 배포 실패의 영향과 위험을 최소화합니다.
  - AWS Cloud9
  - AWS Cloud9란 무엇입니까?
  - 코드 전달 전에 AWS CodeDeploy를 로컬로 테스트 및 디버깅하는 방법

리소스

관련 문서:

- AWS Cloud9

- AWS 개발자 도구
- 코드 전달 전에 AWS CodeDeploy를 로컬로 테스트 및 디버깅하는 방법
- AWS Cloud9란 무엇입니까?

OPS06-BP03 배포 관리 시스템 사용

배포 관리 시스템을 사용하여 변경을 추적하고 구현합니다. 이렇게 하면 수동 프로세스에서 발생하는 오류와 변경 사항 배포를 위한 작업을 줄일 수 있습니다.

AWS에서는 AWS 개발자 도구 (예: AWS CodeCommit, AWS CodeBuild, AWS CodePipeline, AWS CodeDeploy및 AWS CodeStar)와 같은 서비스를 사용하여 지속적 통합/지속적 배포(CI/CD) 파이프라인을 구축할 수 있습니다.

일반적인 안티 패턴:

- 플릿 전체에서 애플리케이션 서버에 대한 업데이트를 수동으로 배포하면 업데이트 오류로 인해 여러 서버가 응답하지 않게 됩니다.
- 여러 시간 동안 애플리케이션 서버 플릿에 수동으로 배포합니다. 변경 중 버전 불일치로 인해 예기치 않은 동작이 발생합니다.

이 모범 사례 수립의 이점: 배포 관리 시스템을 도입하면 변경 사항 배포를 위한 작업량과 수동 절차로 인한 오류 빈도가 줄어듭니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 배포 관리 시스템 사용: 배포 관리 시스템을 사용하여 변경을 추적하고 구현합니다. 이렇게 하면 수동 프로세스에서 발생하는 오류와 변경 사항 배포를 위한 작업량을 줄일 수 있습니다. 코드 체크 인에서 테스트, 배포 및 확인까지의 전체 통합 및 배포 파이프라인을 자동화합니다. 이렇게 하면 리드 시간을 단축하고 변경을 더 자주 수행할 수 있으며 작업량을 더욱 줄일 수 있습니다.
  - AWS CodeDeploy 소개 - Amazon Web Services를 통해 자동화된 소프트웨어 배포
  - AWS CodeDeploy란 무엇인가요?
  - AWS Elastic Beanstalk란 무엇입니까?
  - Amazon API Gateway란 무엇입니까?

리소스

관련 문서:

- AWS CodeDeploy 사용 설명서
- AWS 개발자 도구
- AWS CodeDeploy에서의 샘플 블루/그린 배포 시도
- AWS CodeDeploy란 무엇인가요?
- AWS Elastic Beanstalk란 무엇입니까?
- Amazon API Gateway란 무엇입니까?

관련 동영상:

- AWS를 사용한 고급 연속 딜리버리 기술의 심층 분석
- AWS CodeDeploy 소개 - Amazon Web Services를 통해 자동화된 소프트웨어 배포

OPS06-BP04 제한된 배포를 사용하여 테스트

전체 배포를 진행하기 전에 기존 시스템과 함께 제한된 배포를 사용해 테스트를 진행하여 원하는 결과를 달성할 수 있는지를 확인합니다. 예를 들어 Canary 배포 테스트나 원박스 배포를 사용합니다.

일반적인 안티 패턴:

- 실패한 변경 사항을 모든 프로덕션에 한 번에 배포합니다. 모릅니다.

이 모범 사례 정립의 이점: 제한된 배포 후 변경 사항을 테스트하고 확인하면 고객에게 미치는 영향을 최소화하면서 문제를 조기에 식별하여 고객에게 미치는 영향을 더욱 완화할 수 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 제한된 배포를 사용한 테스트: 전체 배포를 진행하기 전에 기존 시스템과 함께 제한된 배포를 사용하여 테스트를 진행하고 원하는 결과를 달성할 수 있는지를 확인합니다. 예를 들어 Canary 배포 테스트나 원박스 배포를 사용합니다.
  - AWS CodeDeploy 사용 설명서
  - AWS Elastic Beanstalk를 통한 블루/그린 배포

- API Gateway Canary 릴리스 배포 설정
- AWS CodeDeploy에서의 샘플 블루/그린 배포 시도
- AWS CodeDeploy에서 배포 구성 사용

리소스

관련 문서:

- AWS CodeDeploy 사용 설명서
- AWS Elastic Beanstalk를 통한 블루/그린 배포
- API Gateway Canary 릴리스 배포 설정
- AWS CodeDeploy에서의 샘플 블루/그린 배포 시도
- AWS CodeDeploy에서 배포 구성 사용

OPS06-BP05 병렬 환경을 사용하여 배포

병렬 환경에 변경 사항을 구현한 다음 새 환경으로 이전합니다. 배포가 정상 완료되었음이 확인될 때까지는 이전 환경을 유지합니다. 이렇게 하면 만일의 경우 이전 환경을 롤백할 수 있으므로 복구 시간을 최소화할 수 있습니다.

일반적인 안티 패턴:

- 기존 시스템을 수정하여 변경 가능한 배포를 수행합니다. 변경이 실패했음을 발견한 후에는 이전 버전 복원을 위해 시스템을 다시 수정하여 복구 시간을 연장해야 합니다.
- 유지 관리 기간 동안 이전 환경을 폐기한 다음 새 환경 구축을 시작합니다. 절차를 몇 시간 진행한 상태에서 배포에서 복구할 수 없는 문제가 발견되었습니다. 매우 지친 상태에서 이전 배포 절차를 찾아 이전 환경을 다시 구축하기 시작해야 합니다.

이 모범 사례 수립의 이점: 병렬 환경을 사용하면 새 환경을 미리 배포하고 원하는 경우 해당 환경으로 전환할 수 있습니다. 새 환경이 성공적이지 않으면 원래 환경으로 다시 전환하여 신속하게 복구할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 병렬 환경을 사용한 배포: 병렬 환경에 변경 사항을 구현한 다음 새로운 환경으로 이전하거나 전환할 수 있습니다. 배포가 정상 완료되었음이 확인될 때까지는 이전 환경을 유지합니다. 이렇게 하면 만일의 경우 이전 환경을 롤백할 수 있으므로 복구 시간을 최소화할 수 있습니다. 예를 들어 블루/그린 배포를 수행하여 변경 불가능한 인프라를 사용합니다.
  - AWS CodeDeploy에서 배포 구성 사용
  - AWS Elastic Beanstalk를 통한 블루/그린 배포
  - API Gateway Canary 릴리스 배포 설정
  - AWS CodeDeploy에서의 샘플 블루/그린 배포 시도

리소스

관련 문서:

- AWS CodeDeploy 사용 설명서
- AWS Elastic Beanstalk를 통한 블루/그린 배포
- API Gateway Canary 릴리스 배포 설정
- AWS CodeDeploy에서의 샘플 블루/그린 배포 시도
- AWS CodeDeploy에서 배포 구성 사용

관련 동영상:

- AWS를 사용한 고급 연속 딜리버리 기술의 심층 분석

OPS06-BP06 작게 자주 발생하고 되돌릴 수 있는 변경 내용 배포

되돌릴 수 있는 소규모 변경 작업을 자주 수행하면 변경의 범위가 감소합니다. 그러면 문제를 더 쉽게 해결할 수 있으며 변경 사항 롤백 옵션을 사용해 문제 해결 시간을 단축할 수 있습니다.

일반적인 안티 패턴:

- 분기별로 애플리케이션의 새 버전을 배포합니다.
- 데이터베이스 스키마를 자주 변경합니다.
- 수동 인 플레이스 업데이트를 수행하여 기존 설치 및 구성을 덮어씁니다.

이 모범 사례 정립의 이점: 작은 변경 사항을 자주 배포하여 개발 작업의 이점을 더 빠르게 파악합니다. 변경 사항이 작으면 의도하지 않은 결과가 있는지 파악하기가 훨씬 더 쉽습니다. 변경 사항을 되돌릴 수 있는 경우 복구가 간소화됨에 따라 변경 사항을 구현할 위험이 적습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 되돌릴 수 있는 작은 단위의 변경 내용 자주 배포: 되돌릴 수 있는 작은 단위의 변경 내용을 자주 적용하면 변경 범위를 줄일 수 있습니다. 그러면 문제를 더 쉽게 해결할 수 있으며 변경 사항 롤백 옵션을 사용해 문제 해결 시간을 단축할 수 있습니다.

OPS06-BP07 통합 및 배포 완전 자동화

워크로드 빌드, 배포 및 테스트를 자동화합니다. 이렇게 하면 수동 프로세스에서 발생하는 오류와 변경 사항 배포를 위한 작업을 줄일 수 있습니다.

메타데이터를 리소스 태그 및 AWS Resource Groups 을 통해 적용하고, 일관된 태깅 전략 을 시행하면 리소스를 식별할 수 있습니다. 리소스에 조직, 비용 회계, 액세스 제어에 대한 리소스에 태그를 지정하여 자동화된 운영 활동을 실행할 대상을 설정합니다.

일반적인 안티 패턴:

- 금요일에는 기능 브랜치에 대한 새 코드 작성을 마칩니다. 월요일에는 코드 품질 테스트 스크립트와 각 단위 테스트 스크립트를 실행한 후 예정된 다음 릴리스를 위해 코드를 체크인합니다.
- 프로덕션 환경에서 많은 고객에게 영향을 미치는 중요한 문제에 대한 수정을 코딩해야 합니다. 수정 사항을 테스트한 후 코드 및 이메일 변경 관리를 커밋하여 프로덕션에 배포하기 위한 승인을 요청합니다.

이 모범 사례 수립의 이점: 자동화된 빌드 및 배포 관리 시스템을 구현하면 수동 프로세스로 인한 오류와 변경 사항 배포를 위한 작업이 줄어 팀원이 비즈니스 가치를 제공하는 데 집중할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 빌드 및 배포 관리 시스템 사용: 빌드 및 배포 관리 시스템을 사용하면 변경 사항을 추적/구현하고, 수동 프로세스로 인해 발생하는 오류와 작업량을 줄일 수 있습니다. 코드 체크 인에서 빌드, 테스트, 배포 및 확인까지의 전체 통합 및 배포 파이프라인을 완전히 자동화합니다. 이렇게 하면 리드 시간을 단축하고 변경을 더 자주 수행할 수 있으며 작업량을 줄일 수 있습니다.

- AWS CodeBuild란 무엇입니까?
- 소프트웨어 개발을 위한 지속적 통합 모범 사례
- Slalom: AWS의 서버리스 애플리케이션용 CI/CD
- AWS CodeDeploy 소개 - Amazon Web Services를 통해 자동화된 소프트웨어 배포
- AWS CodeDeploy란 무엇입니까?
- AWS를 사용한 고급 지속적 전달 기술 심층 분석

리소스

관련 문서:

- AWS CodeDeploy에서의 샘플 블루/그린 배포 시도
- AWS CodeBuild란 무엇입니까?
- AWS CodeDeploy란 무엇입니까?

관련 동영상:

- 소프트웨어 개발을 위한 지속적 통합 모범 사례
- AWS를 사용한 고급 지속적 전달 기술 심층 분석
- AWS CodeDeploy 소개 - Amazon Web Services를 통해 자동화된 소프트웨어 배포
- Slalom: AWS의 서버리스 애플리케이션용 CI/CD

OPS06-BP08 테스트 및 롤백 자동화

배포된 환경의 테스트를 자동화하여 원하는 성과를 달성할 수 있는지를 확인합니다. 원하는 결과를 달성할 수 없는 경우 알려진 정상 상태로 롤백하는 과정을 자동화하면 수동 프로세스에서 발생하는 오류를 줄이고 복구 시간을 최소화할 수 있습니다.

일반적인 안티 패턴:

- 변경 사항을 워크로드에 배포합니다. 변경 사항이 완료되면 배포 후 테스트를 시작합니다. 작업이 완료되면 워크로드가 작동하지 않으며 고객 연결이 끊어짐을 알게 됩니다. 그런 다음 이전 버전으로 롤백을 시작합니다. 문제 감지에 시간이 오래 걸리면 수동 재배포에 의해 복구 시간이 연장됩니다.

이 모범 사례 수립의 이점: 배포 후 변경 사항을 테스트하고 검증하면 문제를 즉시 식별할 수 있습니다. 이전 버전으로 자동 롤백하면 고객에게 미치는 영향이 최소화됩니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 테스트 및 롤백 자동화: 배포된 환경에서 테스트를 자동화하여 원하는 성과를 달성할 수 있는지를 확인합니다. 원하는 결과를 달성할 수 없는 경우 알려진 정상 상태로 롤백하는 과정을 자동화하면 수동 프로세스에서 발생하는 오류를 줄이고 복구 시간을 최소화할 수 있습니다. 예를 들어 배포 후에 세부 통합 사용자 트랜잭션을 수행하여 결과를 확인한 후 트랜잭션이 실패했다면 변경을 롤백합니다.
  - AWS CodeDeploy로 재배포 및 배포 롤백

리소스

관련 문서:

- AWS CodeDeploy로 재배포 및 배포 롤백

OPS 7 귀사가 워크로드를 지원할 준비가 되어있는지 어떻게 알 수 있습니까?

워크로드, 프로세스, 절차 및 직원의 운영 준비 상태를 평가하여 워크로드와 관련된 운영 위험을 파악합니다.

모범 사례

- OPS07-BP01 직원의 역량 확보
- OPS07-BP02 일관된 방식으로 운영 준비 상태 검토
- OPS07-BP03 런북을 사용한 절차 수행
- OPS07-BP04 플레이북을 사용하여 문제 조사
- OPS07-BP05 정보에 입각하여 시스템 및 변경 사항 배포 결정
- OPS07-BP06 프로덕션 워크로드에 대한 지원 플랜 활성화

OPS07-BP01 직원의 역량 확보

워크로드를 지원하기 위해 적절한 수의 숙련된 인력이 있는지 확인하는 메커니즘을 확보합니다. 워크로드를 구성하는 플랫폼과 서비스에 대해 교육을 받아야 합니다. 워크로드를 운영하는 데 필요한 지식을 제공합니다. 워크로드의 정상 작동을 지원하고 발생하는 인시던트 문제를 해결할 수 있도록 충분한

교육을 받은 직원이 있어야 합니다. 번아웃을 방지하기 위해 업무 대기 기간 및 휴가 기간 동안 다른 인력이 순환할 수 있도록 충분한 인원을 확보합니다.

원하는 결과:

- 워크로드를 사용 가능할 때 워크로드를 지원할 수 있도록 충분한 교육을 받은 직원이 있습니다.
- 워크로드를 구성하는 소프트웨어 및 서비스에 대한 직원 교육을 제공합니다.

일반적인 안티 패턴:

- 사용 중인 플랫폼과 서비스를 운영하도록 훈련된 팀원 없이 워크로드를 배포합니다.
- 대기 근무를 지원하거나 휴가를 내는 직원을 대체할 인력이 충분하지 않습니다.

이 모범 사례 확립의 이점:

- 숙련된 팀원이 있으면 워크로드를 효과적으로 지원할 수 있습니다.
- 충분한 팀원이 있으면 번아웃의 위험을 줄이면서 워크로드 및 대기 근무를 지원할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

워크로드를 지원할 숙련된 인력이 충분히 있는지 검증합니다. 대기 근무를 포함하여 정상적인 운영 활동을 처리할 수 있는 팀원이 충분히 있는지 확인합니다.

고객 사례

AnyCompany Retail은 워크로드를 지원하는 팀이 적절한 인력으로 구성되어 있는지 및 교육을 받았는지 확인합니다. 대기 근무를 지원할 엔지니어가 충분히 있습니다. 직원은 워크로드가 구축된 소프트웨어 및 플랫폼에 대한 교육을 받으며, 인증을 획득하도록 권장됩니다. 워크로드와 대기 근무를 계속 지원하면서 휴가를 낼 수 있을 정도로 인력이 충분합니다.

구현 단계

1. 대기 근무를 비롯하여 워크로드를 운영하고 지원하기에 적절한 수의 직원을 할당합니다.
2. 워크로드를 구성하는 소프트웨어 및 플랫폼에 대해 직원을 교육합니다.
    a. AWS Training and Certification 페이지에는 AWS에 대한 교육 과정 라이브러리가 있습니다. 무료 및 유료의 온라인, 오프라인 과정을 제공합니다.

   b. AWS 전문가로부터 배울 수 있는 이벤트 및 웨비나를 AWS가 주최합니다.
3. 운영 조건 및 워크로드 변화에 따라 팀 규모와 기술을 정기적으로 평가합니다. 운영 요구 사항에 맞게 팀 규모와 기술을 조정합니다.

구현 계획의 작업 수준: 높음. 워크로드를 지원하기 위해 팀을 고용하고 교육하는 데 상당한 노력이 필요할 수 있지만 장기적으로 상당한 이점이 있습니다.

리소스

관련 모범 사례:

- OPS11-BP04 지식 관리 수행 - 팀 구성원은 워크로드를 운영하고 지원하는 데 필요한 정보를 가지고 있어야 합니다. 지식 관리는 이를 제공하는 열쇠입니다.

관련 문서:

- AWS 이벤트 및 웨비나
- AWS Training and Certification

OPS07-BP02 일관된 방식으로 운영 준비 상태 검토

ORR(운영 준비 상태 검토)을 사용하여 워크로드를 운영할 수 있는지 검증할 수 있습니다. ORR은 팀에서 워크로드를 안전하게 운영할 수 있는지 검증할 수 있도록 Amazon에서 개발한 메커니즘입니다. ORR은 요구 사항의 체크리스트를 사용한 검토 및 검사 프로세스입니다. ORR은 팀이 자체 워크로드를 인증하는 데 사용하는 셀프 서비스 경험입니다. ORR에는 다년간의 소프트웨어 구축을 통해 얻은 교훈을 바탕으로 한 모범 사례가 포함되어 있습니다.

ORR 체크리스트는 아키텍처 권장 사항, 운영 프로세스, 이벤트 관리 및 릴리스 품질로 구성되어 있습니다. 오류 수정(CoE) 프로세스는 이러한 항목을 위한 주요 동인입니다. 자체적인 인시던트 사후 분석을 통해 자체 ORR의 발전이 이루어져야 합니다. ORR은 모범 사례를 따르는 것 뿐만 아니라 이전에 경험한 이벤트의 재발을 방지하는 것도 포함됩니다. 마지막으로, 보안, 거버넌스 및 규정 준수 요구 사항 또한 ORR에 포함될 수 있습니다.

워크로드를 일반적인 사용 용도로 시작하기 전에 ORR을 실행한 다음 소프트웨어 개발 수명 주기 전반에 걸쳐 실행합니다. 시작 전에 ORR을 실행하면 워크로드를 안전하게 실행할 수 있는 역량이 향상됩니다. 모범 사례에서 벗어난 부분이 있는지 파악할 수 있도록 워크로드에서 ORR을 주기적으로 다시 실행합니다. 새로운 서비스 출시를 위한 ORR 체크리스트 및 주기적 검토를 위한 ORR을 준비해 둘 수 있습니다. 이렇게 하면 인시던트 사후 분석으로부터 얻은 교훈을 반영하고 포함할 수 있는 새로운 모범

사례를 항상 최신 상태로 유지할 수 있습니다. 클라우드 사용이 성숙해지면 아키텍처에 ORR 요구 사항을 기본으로 구축할 수 있습니다.

원하는 결과: 조직을 위한 모범 사례가 포함된 ORR 체크리스트를 보유합니다. 워크로드 시작 전에 ORR을 수행합니다. 워크로드 수명 주기 동안 ORR을 주기적으로 실행합니다.

일반적인 안티 패턴:

- 운영 가능 여부를 알 수 없는 상태에서 워크로드를 시작합니다.
- 워크로드의 시작을 인증하는 과정에 거버넌스 및 보안 요구 사항이 포함되어 있지 않습니다.
- 워크로드를 주기적으로 재평가하지 않습니다.
- 워크로드 시작 시 필요한 절차를 갖추고 있지 않습니다.
- 여러 워크로드에서 동일한 근본 원인 실패가 반복됩니다.

이 모범 사례 확립의 이점:

- 워크로드에 아키텍처, 프로세스 및 관리 모범 사례가 포함됩니다.
- 얻은 교훈이 ORR 프로세스에 포함됩니다.
- 워크로드 시작 시 필요한 절차가 갖춰져 있습니다.
- 워크로드의 소프트웨어 수명 주기 전반에 걸쳐 ORR이 실행됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

ORR은 프로세스와 체크리스트로 이루어져 있습니다. ORR 프로세스는 조직에서 채택해야 하며 경영진 후원자가 지원해야 합니다. 최소한, 워크로드가 일반적인 사용을 시작하기 전에 ORR을 수행해야 합니다. 소프트웨어 개발 수명 주기 전반에 걸쳐 ORR을 실행하여 모범 사례나 새 요구 사항이 최신 상태로 포함되도록 해야 합니다. ORR 체크리스트에는 구성 항목, 보안 및 거버넌스 요구 사항, 조직의 모범 사례가 포함되어야 합니다. 시간이 지남에 따라 AWS Config, AWS Security Hub 및 AWS Control Tower 가드레일과 같은 서비스를 사용하여 모범 사례의 자동 탐지를 위해 ORR의 모범 사례를 가드레일에 구축할 수 있습니다.

고객 사례

몇 번의 프로덕션 인시던트 후 AnyCompany Retail은 ORR 프로세스를 구현하기로 했습니다. 이를 위해 모범 사례, 거버넌스 및 규정 준수 요구 사항, 그리고 중단으로부터 얻은 교훈을 통해 구성된 체크리

스트를 구축했습니다. 새 워크로드를 시작하기 전에 ORR을 수행합니다. 모든 워크로드는 ORR 체크리스트에 추가되는 새로운 모범 사례 및 요구 사항을 통합하기 위해 모범 사례의 하위 집합이 포함된 연간 ORR을 수행합니다. 시간이 지나면서 AnyCompany Retail은 AWS Config 를 사용하여 일부 모범 사례를 탐지하고 ORR 프로세스의 속도를 높였습니다.

구현 단계

ORR에 대해 자세히 알아보려면 ORR(운영 준비 상태 검토) 백서를 확인하세요. ORR 프로세스의 이력, 자체적인 ORR 사례를 구축하는 방법, ORR 체크리스트를 개발하는 방법에 대한 자세한 정보를 제공합니다. 다음 단계는 해당 문서의 축약 버전입니다. ORR이 무엇인지와 구축 방법을 심층적으로 이해하려면 이 백서를 읽어보시는 것이 좋습니다.

1. 보안, 운영 및 개발 담당자를 포함한 핵심 이해 관계자를 한 자리에 모읍니다.
2. 각 이해 관계자가 한 가지 이상의 요구 사항을 제공하도록 합니다. 첫 반복의 경우 항목의 수를 30개 이하로 제한합니다.
   - 부록 B: ORR 질문 예시 는 ORR(운영 준비 상태 검토) 백서에 수록되어 있으며 시작 시 사용 가능한 샘플 질문이 포함되어 있습니다.
3. 요구 사항을 스프레드시트에 수집합니다.
   - 이때 사용자 지정 렌즈 (AWS Well-Architected Tool 에 포함)를 사용하여 ORR을 개발하고 계정 및 AWS Organization 전체에서 이를 공유할 수 있습니다.
4. ORR을 수행할 하나의 워크로드를 식별합니다. 출시 전 워크로드나 내부 워크로드가 가장 좋습니다.
5. ORR 체크리스트를 실행하고 탐색 내용을 기록합니다. 완화 조치가 적용된 경우 탐색 결과가 좋지 않을 수 있습니다. 완화 조치가 부족한 탐색 결과에 대해서는 항목의 백로그에 이를 추가하고 시작 전에 구현합니다.
6. 시간이 지나는 동안 ORR 체크리스트에 모범 사례 및 요구 사항을 계속 추가합니다.

Enterprise Support를 이용하는 AWS Support 고객은 기술 지원 관리자에게 운영 준비 상태 검토 워크숍 을 요청할 수 있습니다. 워크숍은 ORR 체크리스트 개발을 위한 대화형 프로세스 뒤집기 세션입니다.

구현 계획의 작업 수준: 높음. 조직에서 ORR 사례를 도입하려면 경영진의 후원과 이해 관계자의 승인이 필요합니다. 조직 전체의 의견을 받아 체크리스트를 구축 및 업데이트해야 합니다.

리소스

관련 모범 사례:

- OPS01-BP03 거버넌스 요구 사항 평가 – 거버넌스 요구 사항은 ORR 체크리스트에 매우 적합합니다.
- OPS01-BP04 규정 준수 요구 사항 평가 – 규정 준수 요구 사항이 ORR 체크리스트에 포함되는 경우도 있습니다. 그 외에는 별도의 프로세스입니다.
- OPS03-BP07 리소스 팀 적절히 관리 – 팀 역량은 ORR 요구 사항을 위한 좋은 후보입니다.
- OPS06-BP01 변경이 적절하지 못한 경우에 대한 계획 수립 – 롤백이나 롤포워드 계획은 워크로드를 시작하기 전에 수립해야 합니다.
- OPS07-BP01 직원의 역량 확보 – 워크로드를 지원하려면 필수 인력이 있어야 합니다.
- SEC01-BP03 제어 목표 파악 및 검증 – 보안 제어 목표는 우수한 ORR 요구 사항을 수립하는 데 도움이 됩니다.
- REL13-BP01 가동 중단 시간 및 데이터 손실 시의 복구 목표 정의 – 재해 복구 계획은 ORR 요구 사항으로 적합합니다.
- COST02-BP01 조직 요구 사항에 따라 정책 개발 – 비용 관리 정책은 ORR 체크리스트에 포함하는 것이 좋습니다.

관련 문서:

- AWS Control Tower - AWS Control Tower의 가드레일
- AWS Well-Architected Tool - 사용자 지정 렌즈
- Adrian Hornsby의 운영 준비 상태 검토 템플릿
- ORR(운영 준비 상태 검토) 백서

관련 동영상:

- AWS Supports You | 효과적인 ORR(운영 준비 상태 검토) 구축

관련 예시:

- 샘플 ORR(운영 준비 상태 검토) 렌즈

관련 서비스:

- AWS Config
- AWS Control Tower

- AWS Security Hub
- AWS Well-Architected Tool

OPS07-BP03 런북을 사용한 절차 수행

런북 은 특정 결과를 달성하기 위해 문서화된 프로세스입니다. 런북은 누군가가 어떤 것을 수행하기 위해 따르는 일련의 단계로 구성됩니다. 런북은 항공 산업 초창기부터 운영에 사용되어 왔습니다. Amazon은 클라우드 운영 시 런북을 사용하여 위험을 줄이고 원하는 결과를 얻습니다. 가장 간단하게 표현하자면, 런북은 작업 완료를 위한 체크리스트입니다.

런북은 워크로드 운영을 위해 필수적인 부분입니다. 새로운 팀원의 온보딩부터 주요 릴리스의 배포에 이르기까지 런북은 사용자가 누구든 일관된 결과를 얻을 수 있는 코드화된 프로세스입니다. 런북 업데이트는 변경 관리 프로세스의 중요한 구성 요소이기 때문에 런북은 중앙 위치에서 게시되고 프로세스가 발전함에 따라 업데이트됩니다. 또한 오류 처리, 도구, 권한, 예외 및 문제 발생 시 에스컬레이션에 대한 지침도 포함해야 합니다.

조직이 성숙해지면 런북 자동화를 시작합니다. 간단하고 자주 사용하는 런북으로 시작합니다. 스크립팅 언어를 사용하여 단계를 자동화하거나 단계를 수행하기 쉽게 만듭니다. 처음 런북을 몇 개 자동화해 보면 더 복잡한 런북을 자동화하는 데 시간을 할애하게 될 것입니다. 시간이 흐르면 대부분의 런북이 자동화되어야 합니다.

원하는 결과: 팀에 워크로드 작업을 수행하기 위한 단계별 가이드 모음이 있습니다. 런북에는 원하는 결과, 필요한 도구, 권한 및 오류 처리 지침이 들어 있습니다. 런북은 중앙 위치에 저장해 두고 자주 업데이트합니다.

일반적인 안티 패턴:

- 프로세스의 각 단계를 완료하기 위해 메모리를 사용합니다.
- 체크리스트 없이 변경 사항을 수동으로 배포합니다.
- 동일한 프로세스를 팀원 여러 명이 수행하지만 사용하는 단계와 결과가 다릅니다.
- 런북이 시스템 변경 사항과 동기화되지 않고 자동화와 상관 없이 변경됩니다.

이 모범 사례 확립의 이점:

- 수동 작업의 오류 발생률이 감소합니다.
- 작업이 일관된 방식으로 수행됩니다.
- 새로운 팀원이 작업 수행을 더 빨리 시작할 수 있습니다.

- 런북을 자동화해 노력을 줄일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

런북은 조직의 성숙도에 따라 여러 가지 형태일 수 있습니다. 최소한 단계별 텍스트 문서로 구성되어야 합니다. 원하는 결과가 명확하게 명시되어 있어야 합니다. 필요한 특수 권한 및 도구도 확실하게 기록해야 합니다. 오류 처리 및 문제 발생 시 에스컬레이션에 대한 자세한 지침을 제공합니다. 런북 소유자를 나열하고 런북을 중앙 위치에 게시합니다. 런북을 문서화하면 다른 팀원이 실행해보도록 하여 확인합니다. 절차가 발전하면 변경 관리 프로세스에 따라 런북을 업데이트합니다.

텍스트 런북은 조직이 성숙함에 따라 자동화해야 합니다. 또한 AWS Systems Manager 자동화등과 같은 서비스를 사용하여 일반 텍스트를 워크로드에 대해 실행할 수 있는 자동화로 변환할 수 있습니다. 이러한 자동화는 이벤트에 대응하여 실행해 워크로드 유지를 위한 운영 부담을 줄일 수 있습니다.

고객 사례

AnyCompany Retail은 소프트웨어를 배포하는 중 데이터베이스 스키마 업데이트를 수행해야 합니다. 클라우드 운영 팀은 데이터베이스 관리 팀과 협력하여 이러한 변경 사항을 수동으로 배포하기 위한 런북을 빌드했습니다. 이 런북에는 프로세스의 각 단계를 체크리스트 형식으로 나열되어 있습니다. 또한 문제 발생 시 오류 처리에 대한 섹션이 포함되어 있습니다. 팀은 내부 Wiki에 다른 런북과 함께 이 런북을 게시했습니다. 클라우드 운영 팀은 향후 스프린트에서 런북을 자동화할 계획입니다.

구현 단계

기존 문서 리포지토리가 없는 경우에는 버전 제어 리포지토리에서 런북 라이브러리 빌드를 시작하는 것이 좋습니다. 런북은 Markdown을 사용하여 빌드할 수 있습니다. 런북 빌드를 시작하는 데 사용할 수 있는 런북 템플릿 예시를 제공합니다.

```
# ## ## ## ## ## | ## ID | ## | ### ## | ## ## | ## ### | ### #### | ###### POC |
 |-------|-------|-------|-------|-------|-------|-------| | RUN001 | ## ## ### ## | ##
 | ## | ## | 2022-09-21 | ###### ## | ## ## 1 1## 2. 2##
```

1. 기존 문서 리포지토리 또는 Wiki가 없는 경우 버전 관리 시스템에서 새로운 버전 관리 리포지토리를 생성합니다.
2. 런북이 없는 프로세스를 파악합니다. 이상적인 프로세스는 반규칙적으로 수행되며 단계 수가 적고 장애 영향이 적은 프로세스입니다.

3. 문서 리포지토리에서 템플릿을 사용하여 새로운 Markdown 문서 초안을 작성합니다. 그런 다음 ## ## 을 작성하고 ## ## ## ## ### #####.
4. 첫 번째 단계를 시작하고 런북의 ## 부분을 작성합니다.
5. 팀원에게 런북을 제공하고 런북을 사용하여 단계를 확인하도록 합니다. 누락된 부분이 있거나 명확히 설명해야 할 부분이 있다면 런북을 업데이트합니다.
6. 내부 문서 저장소에 런북을 게시합니다. 게시한 다음 팀 및 다른 이해관계자에게 알립니다.
7. 시간이 흐르면 런북 라이브러리를 구축합니다. 라이브러리가 커지면 런북 자동화 작업을 시작합니다.

구현 계획의 작업 수준: 낮음. 런북의 최소 표준은 단계별 텍스트 가이드입니다. 런북 자동화는 구현 작업을 늘릴 수 있습니다.

리소스

관련 모범 사례:

- OPS02-BP02 프로세스 및 절차의 소유자 식별: 런북에는 런북 유지 관리를 담당하는 소유자가 있어야 합니다.
- OPS07-BP04 플레이북을 사용하여 문제 조사: 런북과 플레이북은 서로 비슷하지만 런북에는 원하는 결과가 있다는 중요한 차이점이 있습니다. 대부분의 경우 플레이북이 근본 원인을 식별하면 런북이 트리거됩니다.
- OPS10-BP01 이벤트, 인시던트 및 문제 관리 프로세스 사용: 런북은 적절한 이벤트, 인시던트, 문제 관리 방침의 일부입니다.
- OPS10-BP02 알림별 프로세스 마련: 런북과 플레이북은 알림에 대응하기 위해 사용해야 합니다. 시간이 흐르면 이러한 대응은 자동화해야 합니다.
- OPS11-BP04 지식 관리 수행: 런북 유지 관리는 지식 관리의 중요한 부분입니다.

관련 문서:

- 자동화된 플레이북 및 런북을 사용하여 운영 우수성 달성
- AWS Systems Manager: 런북을 사용한 작업
- AWS 대규모 마이그레이션을 위한 마이그레이션 플레이북 - 작업 4: 마이그레이션 런북 개선
- AWS Systems Manager 자동화 런북을 사용하여 운영 작업 해결

관련 동영상:

- AWS re:Invent 2019: 런북, 인시던트 보고서, 인시던트 대응에 대한 DIY 가이드(SEC318-R1)
- AWS에서 IT 운영을 자동화하는 방법 | Amazon Web Services
- AWS Systems Manager(으)로 스크립트 통합

관련 예시:

- AWS Systems Manager: 자동화 시연
- AWS Systems Manager: 최신 스냅샷 런북에서 루트 볼륨 복원
- Jupyter Notebook 및 CloudTrail Lake를 사용하여 AWS 인시던트 응답 런북 빌드
- Gitlab - 런북
- Rubix - Jupyter Notebook의 런북 빌드를 위한 Python 라이브러리
- Document Builder를 사용하여 사용자 지정 런북 만들기
- Well-Architected 실습: 플레이북 및 런북으로 운영 자동화

관련 서비스:

- AWS Systems Manager 자동화

OPS07-BP04 플레이북을 사용하여 문제 조사

플레이북은 인시던트를 조사하는 데 사용하는 단계별 지침입니다. 인시던트가 발생하면 플레이북을 사용하여 조사하고, 영향의 범위를 살펴보고, 근본 원인을 파악합니다. 플레이북은 배포 실패부터 보안 인시던트까지 다양한 시나리오에 사용됩니다. 대부분의 경우, 플레이북으로 근본 원인을 파악하고 런북을 사용하여 이를 완화합니다. 플레이북은 조직의 인시던트 대응 계획을 위한 필수 구성 요소입니다.

우수한 플레이북에는 몇 가지 주요 기능이 있으며 이를 통해 사용자에게 탐색 프로세스를 단계별로 안내합니다. 외부 관점에서 생각할 때, 인시던트를 진단하기 위해 어떤 단계를 따라야 할까요? 플레이북에 특수 도구나 승격된 권한이 필요한 경우 플레이북에서 이를 명확하게 정의합니다. 이해 관계자에게 조사 상황을 알리기 위한 커뮤니케이션 계획을 수립하는 것이 중요합니다. 근본 원인을 파악할 수 없는 경우에 대비한 에스컬레이션 계획도 있어야 합니다. 근본 원인이 파악되었다면 플레이북은 해결 방법을 설명하는 런북을 명시해야 합니다. 플레이북은 중앙 집중식으로 저장하고 정기적으로 유지 관리해야 합니다. 플레이북이 특정 알림에 사용되는 경우, 알림에 플레이북에 대한 포인터를 추가하여 팀에 제공해야 합니다.

조직이 성숙해지면 플레이북을 자동화합니다. 위험성이 낮은 인시던트를 다루는 플레이북으로 시작합니다. 스크립팅을 사용하여 검색 단계를 자동화합니다. 일반적인 근본 원인을 완화하는 데 사용할 수 있는 지원 런북을 반드시 갖추도록 합니다.

원하는 결과: 조직에 일반적인 인시던트를 위한 플레이북이 있습니다. 플레이북을 중앙 위치에 저장해 두고 팀원들이 사용할 수 있습니다. 플레이북이 자주 업데이트됩니다. 알려진 모든 근본 원인에 대한 지원 런북이 구축되어 있습니다.

일반적인 안티 패턴:

- 인시던트를 조사하기 위한 표준 방식이 없습니다.
- 팀원들이 기억이나 제도적 지식에 의존하여 배포 실패 문제를 해결합니다.
- 새로운 팀원이 시행 착오를 거쳐 문제 조사 방법을 배웁니다.
- 문제 조사의 모범 사례가 팀 내에서 공유되고 있지 않습니다.

이 모범 사례 확립의 이점:

- 플레이북은 인시던트를 완화하는 데 큰 도움이 됩니다.
- 다양한 팀원이 동일한 플레이북을 사용함으로써 일관적인 방법으로 근본 원인을 파악할 수 있습니다.
- 알려진 근본 원인의 경우 이에 대비하여 개발된 런북을 통해 복구 시간을 앞당길 수 있습니다.
- 플레이북을 통해 팀원들이 더 빨리 문제 해결에 참여할 수 있습니다.
- 팀이 반복 가능한 플레이북을 통해 프로세스를 확장할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

플레이북의 구축 및 사용 방법은 조직의 성숙도에 따라 다릅니다. 클라우드가 처음인 경우 플레이북을 중앙 문서 리포지토리에 텍스트 형식으로 구축합니다. 조직이 성숙해지면서 Python과 같은 스크립팅 언어를 통해 플레이북을 반자동화할 수 있습니다. 이러한 스크립트를 Jupyter Notebook 내부에서 실행하여 탐색 속도를 높일 수 있습니다. 완전히 성숙된 조직은 런북으로 자동 복구할 수 있는 일반적인 문제에 대한 완전히 자동화된 플레이북을 보유합니다.

워크로드에 발생하는 일반적인 인시던트를 리스팅하여 플레이북의 구축을 시작할 수 있습니다. 시작하려면 위험성이 낮고 근본 원인이 몇 가지 문제로 좁혀진 인시던트에 대한 플레이북을 선택합니다. 간

단한 시나리오에 대한 플레이북을 갖춘 후에는 근본 원인이 잘 알려지지 않았고 위험성이 더 높은 시나리오로 넘어가도록 합니다.

텍스트 플레이북은 조직이 성숙해지면 자동화해야 합니다. 또한 AWS Systems Manager Automations등과 같은 서비스를 사용하여 일반 텍스트를 자동화로 변환할 수 있습니다. 이러한 자동화는 워크로드에 대해 실행함으로써 조사 속도를 높일 수 있습니다. 이벤트에 대한 대응으로 이러한 자동화를 활성화하여 인시던트를 발견하고 해결하는 데 걸리는 평균 시간을 단축할 수 있습니다.

고객은 AWS Systems Manager Incident Manager 를 사용하여 인시던트에 대응합니다. 이 서비스는 인시던트를 분류하고, 복구 및 완화 과정에서 이해 관계자에게 이를 알리고, 인시던트 전반에서 협업할 수 있는 단일 인터페이스를 제공합니다. 또한 AWS Systems Manager Automations를 사용하여 탐지 및 복구 속도를 높입니다.

고객 사례

AnyCompany Retail에 생산 인시던트가 발생했고 당직 근무 중인 엔지니어가 플레이북을 사용하여 문제를 조사했습니다. 단계에 따라 진행하면서 플레이북에서 파악한 주요 이해 관계자에게 계속 최신 정보를 보고했습니다. 엔지니어는 백엔드 서비스의 경합 상태가 근본 원인임을 확인했습니다. 엔지니어는 런북에 따라 서비스를 다시 시작하고 AnyCompany Retail을 온라인으로 전환했습니다.

구현 단계

기존 문서 리포지토리가 없는 경우 플레이북 라이브러리에 대한 버전 제어 리포지토리를 생성하는 것이 좋습니다. 플레이북은 대부분의 플레이북 자동화 시스템과 호환되는 Markdown을 사용하여 구축할 수 있습니다. 처음부터 시작하는 경우 다음 예시 플레이북 템플릿을 사용합니다.

```
# #### ## ## #### ## | #### ID | ## | ### ## | ## ## | #
### ### | ## #### ## | ###### POC | ## ### | ###### ## |
 |-------|-------|-------|-------|-------|-------|-------|-------|-------| | RUN001 | #
 ##### ### #####? ## ##### #####? | ## | ## | ### ## | 2022-09-21 | ###### ## | ## ###
 ## | ## # ##### ### #####? | ## ## 1. 1## 2. 2##
```

1. 기존 문서 리포지토리 또는 Wiki가 없는 경우 버전 관리 시스템에서 플레이북에 대한 새로운 버전 관리 리포지토리를 생성합니다.
2. 조사가 필요한 일반적인 문제를 파악합니다. 근본 원인이 몇 가지 문제로 한정되어 있고 해결 방법의 위험성이 낮은 시나리오여야 합니다.
3. Markdown 템플릿을 사용하여 #### ## 섹션과 #### ##필드를 작성합니다.
4. 문제 해결 단계를 작성합니다. 수행해야 하는 작업 또는 조사해야 하는 영역을 최대한 명확하게 작성합니다.

5. 팀원에게 플레이북을 전달하여 살펴보고 확인할 수 있도록 합니다. 누락되거나 명확하지 않은 사항이 있는 경우 플레이북을 업데이트합니다.
6. 문서 리포지토리에 플레이북을 게시하고 팀과 모든 이해 관계자에게 이를 알립니다.
7. 더 많은 플레이북을 추가할수록 이 플레이북 라이브러리는 더 발전하게 됩니다. 여러 플레이북이 있다면 플레이북의 자동화와 동기화를 유지할 수 있도록 AWS Systems Manager Automations와 같은 도구를 사용하여 자동화를 시작합니다.

구현 계획의 작업 수준: 낮음. 플레이북은 중앙 위치에 저장되는 텍스트 문서여야 합니다. 더 성숙한 조직은 플레이북 자동화를 진행합니다.

리소스

관련 모범 사례:

- OPS02-BP02 프로세스 및 절차의 소유자 식별: 플레이북에는 플레이북 유지 관리를 담당하는 소유자가 있어야 합니다.
- OPS07-BP03 런북을 사용한 절차 수행: 런북과 플레이북은 비슷하지만 런북에는 원하는 결과가 있다는 중요한 차이점이 있습니다. 대부분의 경우 플레이북으로 근본 원인을 파악하고 난 뒤 런북이 사용됩니다.
- OPS10-BP01 이벤트, 인시던트 및 문제 관리 프로세스 사용: 플레이북은 적절한 이벤트, 인시던트, 문제 관리 방침의 일부입니다.
- OPS10-BP02 알림별 프로세스 마련: 런북과 플레이북은 알림에 대응하는 데 사용해야 합니다. 시간이 흐르면 이러한 대응은 자동화되어야 합니다.
- OPS11-BP04 지식 관리 수행: 플레이북 유지 관리는 지식 관리의 중요한 부분입니다.

관련 문서:

- 자동화된 플레이북 및 런북을 사용하여 운영 우수성 달성
- AWS Systems Manager: 런북을 사용한 작업
- AWS Systems Manager Automation 런북을 사용하여 운영 작업 해결

관련 동영상:

- AWS re:Invent 2019: 런북, 인시던트 보고서, 인시던트 대응에 대한 DIY 가이드(SEC318-R1)
- AWS Systems Manager Incident Manager - AWS 가상 워크숍

- AWS Systems Manager로 스크립트 통합

관련 예시:

- AWS 고객 플레이북 프레임워크
- AWS Systems Manager: 자동화 시연
- Jupyter Notebook 및 CloudTrail Lake를 사용하여 AWS 인시던트 대응 런북 구축
- Rubix - Jupyter Notebook에서 런북을 구축하기 위한 Python 라이브러리
- Document Builder를 사용하여 사용자 지정 런북 만들기
- Well-Architected 실습: 플레이북 및 런북으로 운영 자동화
- Well-Architected 실습: Jupyter를 사용한 인시던트 대응 플레이북

관련 서비스:

- AWS Systems Manager Automation
- AWS Systems Manager Incident Manager

OPS07-BP05 정보에 입각하여 시스템 및 변경 사항 배포 결정

워크로드에 대한 변경 성공과 변경 실패를 처리하는 프로세스를 갖춥니다. 사전 분석(pre-mortem)이란 팀이 완화 전략을 개발하기 위해 실패를 시뮬레이션하는 연습입니다. 사전 분석 기능을 사용하여 실패를 예측하고 적절한 절차를 생성합니다. 변경 사항을 워크로드에 배포할 때의 이점과 위험을 평가합니다. 모든 변경 사항이 거버넌스를 준수하는지 확인합니다.

원하는 결과:

- 워크로드에 변경 사항을 배포할 때 정보에 입각한 결정을 내립니다.
- 변경 사항은 거버넌스를 준수합니다.

일반적인 안티 패턴:

- 실패한 배포를 처리하는 프로세스 없이 워크로드에 변경 사항을 배포합니다.
- 거버넌스 요구 사항을 준수하지 않는 변경 사항을 프로덕션 환경에 적용합니다.
- 리소스 사용률에 대한 기준을 설정하지 않고 새 워크로드 버전을 배포합니다.

이 모범 사례 확립의 이점:

- 워크로드 변경에 실패한 경우에 대비합니다.
- 워크로드에 대한 변경 사항은 거버넌스 정책을 준수합니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

사전 분석을 사용하여 변경 실패에 대비한 프로세스를 개발합니다. 변경 실패에 대비한 프로세스를 문서화합니다. 모든 변경 사항이 거버넌스를 준수하는지 확인합니다. 변경 사항을 워크로드에 배포할 때의 이점과 위험을 평가합니다.

고객 사례

AnyCompany Retail은 변경 실패에 대비한 프로세스를 검증하기 위해 정기적으로 사전 분석을 수행합니다. 공유 Wiki에 프로세스를 문서화하고 자주 업데이트합니다. 모든 변경 사항은 거버넌스 요구 사항을 준수합니다.

구현 단계

1. 워크로드에 변경 사항을 배포할 때 정보에 입각한 결정을 내립니다. 성공적인 배포를 위한 기준을 설정하고 검토합니다. 변경 롤백을 트리거하는 시나리오 또는 기준을 개발합니다. 실패한 변경의 위험과 변경 사항 배포의 이점을 비교합니다.
2. 모든 변경 사항이 거버넌스 정책을 준수하는지 확인합니다.
3. 사전 분석을 사용하여 변경이 실패한 경우에 대한 계획을 수립하고 완화 전략을 문서화합니다. 실패한 변경을 모델링하고 롤백 절차를 검증하기 위해 탁상 연습을 실행합니다.

구현 계획의 작업 수준: 보통. 사전 분석 사례를 구현하려면 조직 전반의 이해 관계자의 조율과 노력이 필요합니다.

리소스

관련 모범 사례:

- OPS01-BP03 거버넌스 요구 사항 평가 - 거버넌스 요구 사항은 변경 사항 배포 여부를 결정하는 핵심 요소입니다.
- OPS06-BP01 변경이 적절하지 못한 경우에 대한 계획 수립 - 배포 실패를 완화하기 위한 계획을 수립하고 사전 분석을 사용하여 이를 검증합니다.

- OPS06-BP02 변경 사항 테스트 및 확인 - 프로덕션 결함을 줄이기 위해 배포 전에 모든 소프트웨어 변경 사항을 적절하게 테스트해야 합니다.
- OPS07-BP01 직원의 역량 확보 - 워크로드를 지원할 수 있는 충분한 교육을 받은 인력을 확보하는 것은 정보에 입각한 시스템 변경 사항 배포 결정을 내리는 데 필수적입니다.

관련 문서:

- Amazon Web Services: 위험 및 규정 준수
- AWS 공동 책임 모델
- Governance in the AWS 클라우드: The Right Balance Between Agility and Safety(AWS 클라우드의 거버넌스: 민첩성과 안전 사이의 올바른 균형)

OPS07-BP06 프로덕션 워크로드에 대한 지원 플랜 활성화

프로덕션 워크로드가 의존하는 모든 소프트웨어 및 서비스에 대한 지원을 활성화합니다. 프로덕션 서비스 수준 요구 사항을 충족하는 적절한 지원 수준을 선택합니다. 이러한 종속성 지원 플랜은 서비스 중단 또는 소프트웨어 문제가 생긴 경우에 필요합니다. 모든 서비스 및 소프트웨어 공급업체에 대한 지원 플랜과 지원 요청 방법을 문서화합니다. 지원 담당 연락처가 최신 상태로 유지되는지 확인하는 메커니즘을 구현합니다.

원하는 결과:

- 프로덕션 워크로드가 의존하는 소프트웨어 및 서비스의 지원 플랜을 구현합니다.
- 서비스 수준 요구 사항에 따라 적절한 지원 플랜을 선택합니다.
- 지원 플랜, 지원 수준 및 지원 요청 방법을 문서화합니다.

일반적인 안티 패턴:

- 중요한 소프트웨어 공급업체에 대한 지원 플랜이 없습니다. 워크로드가 이러한 문제로 인해 영향을 받는데도, 문제를 신속하게 해결하거나 공급업체로부터 적시에 업데이트를 제공받기 위한 어떠한 조치도 취할 수 없습니다.
- 소프트웨어 공급업체의 주 담당자였던 개발자가 퇴사했습니다. 공급업체 지원 팀과 직접 소통할 수 없습니다. 일반 연락처 시스템을 재검색하고 탐색하는 데 시간을 할애해야 하므로, 필요할 때 응답하는 데 걸리는 시간이 늘어납니다.
- 소프트웨어 공급업체에서 프로덕션 중단이 발생합니다. 지원 사례를 제출하는 방법을 문서화한 설명서가 없습니다.

이 모범 사례 확립의 이점:

- 적절한 지원 수준을 사용하면 서비스 수준 요구 사항을 충족하는 데 필요한 시간 안에 응답을 받을 수 있습니다.
- 지원받는 고객은 프로덕션 문제가 생긴 경우 에스컬레이션할 수 있습니다.
- 소프트웨어 및 서비스 공급업체는 인시던트 발생 시 문제 해결을 지원할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

프로덕션 워크로드가 의존하는 모든 소프트웨어 및 서비스 공급업체에 대한 지원 플랜을 활성화합니다. 서비스 수준 요구 사항을 충족하는 적절한 지원 플랜을 수립합니다. AWS 고객의 경우 프로덕션 워크로드가 있는 모든 계정에서 AWS Business Support 이상을 사용할 수 있습니다. 지원 공급업체와 정기적으로 만나 지원 오퍼링, 프로세스 및 연락처에 대한 최신 정보를 확인하세요. 운영 중단 시 에스컬레이션하는 방법을 포함하여 소프트웨어 및 서비스 공급업체에 지원을 요청하는 방법을 문서화합니다. 지원 연락처를 최신 상태로 유지하기 위한 메커니즘을 구현합니다.

고객 사례

AnyCompany Retail에서는 모든 상용 소프트웨어 및 서비스 종속성에 지원 플랜을 마련했습니다. 예를 들어, 프로덕션 워크로드를 보유한 모든 계정에서 AWS Enterprise Support를 사용 중입니다. 문제가 발생하면 개발자 누구나 지원 사례를 제출할 수 있습니다. 지원을 요청하는 방법, 통지할 대상, 사례를 신속하게 처리하기 위한 모범 사례 정보가 나와 있는 Wiki 페이지가 있습니다.

구현 단계

1. 조직의 이해관계자와 협력하여 워크로드가 의존하는 소프트웨어 및 서비스 공급업체를 식별합니다. 이러한 종속성을 문서화합니다.
2. 워크로드에 대한 서비스 수준 요구 사항을 결정합니다. 이에 맞는 지원 플랜을 선택합니다.
3. 상용 소프트웨어 및 서비스의 경우 공급업체와 함께 지원 플랜을 수립합니다.
    a. 모든 프로덕션 계정에 대해 AWS Business Support 이상을 구독하면 AWS Support로부터 더 빠르게 응답을 받을 수 있으므로, AWS Business Support 이상을 구독할 것을 강력히 권장합니다. 프리미엄 지원을 받지 못하는 경우 AWS Support의 도움이 필요한 문제를 처리할 수 있는 실행 플랜을 마련해야 합니다. AWS Support는 사용자가 성능을 최적화하고, 비용을 절감하고, 혁신 속도를 높이는 데 적극적으로 도움이 될 수 있도록 설계된 다양한 도구 및 기술, 인력, 프로그램을 제공합니다. AWS Business Support는 AWS Trusted Advisor 및 AWS Personal Health Dashboard에 액세스하고 응답 시간을 단축하는 등의 추가적인 이점을 제공합니다.

4. 지식 관리 도구에 지원 플랜을 문서화합니다. 지원 요청 방법, 지원 사례가 접수될 경우 통지 대상, 인시던트 발생 시의 에스컬레이션 방법을 포함합니다. Wiki는 지원 프로세스나 연락처의 변경 사항을 알게 된 누구나 문서를 필요에 따라 업데이트할 수 있도록 지원하는 좋은 메커니즘입니다.

구현 계획의 작업 수준: 낮음. 대부분의 소프트웨어 및 서비스 공급업체는 옵트인 지원 플랜을 제공합니다. 지식 관리 시스템에서 지원 모범 사례를 문서화하고 공유하면 프로덕션 문제가 발생할 때 팀이 무엇을 해야 하는지 알 수 있습니다.

리소스

관련 모범 사례:

- OPS02-BP02 프로세스 및 절차의 소유자 식별

관련 문서:

- AWS Support 플랜

관련 서비스:

- AWS Business Support
- AWS Enterprise Support

운영

질문

- OPS 8 워크로드의 상태를 어떻게 파악하십니까?
- OPS 9 운영 상태를 어떻게 파악하십니까?
- OPS 10 워크로드 및 운영 이벤트를 어떻게 관리하십니까?

OPS 8 워크로드의 상태를 어떻게 파악하십니까?

워크로드 지표를 정의, 캡처 및 분석하면 워크로드 이벤트에 대한 가시성을 확보하여 적절한 조치를 취할 수 있습니다.

모범 사례

OPS08-BP01 핵심 성과 지표 파악

원하는 비즈니스 성과(예: 주문율, 고객 유지율, 이익 및 운영 지출 비교)과 고객 성과(예: 고객 만족도)를 기반으로 KPI(핵심 성과 지표)를 파악합니다. 그리고 KPI를 평가하여 워크로드의 성공 여부를 결정합니다.

일반적인 안티 패턴:

- 경영진으로부터 워크로드가 얼마나 성공적으로 비즈니스 요구를 충족하고 있는지에 대한 질문을 받지만 성공 여부를 판단하기 위한 준거 기준이 없습니다.
- 조직에서 운영하는 자체 상용 애플리케이션이 비용 효율적인지 판단할 수 없습니다.

이 모범 사례 정립의 이점: 핵심 성과 지표를 파악하면 워크로드 상태 및 성공 여부를 테스트하여 비즈니스 성과를 달성할 수 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 핵심 성과 지표 파악: 원하는 비즈니스 성과와 고객 성과를 기준으로 핵심 성과 지표(KPI)를 확인합니다. 그리고 KPI를 평가하여 워크로드의 성공 여부를 결정합니다.

OPS08-BP02 워크로드 지표 정의

워크로드 상태를 측정하는 데 사용할 지표를 정의합니다. 워크로드 상태는 비즈니스 성과(KPI) 달성과 워크로드 구성 요소 및 애플리케이션의 상태로 측정됩니다. KPI의 예로는 방치된 장바구니, 제출된 주문, 비용, 가격 및 할당된 워크로드 비용이 있습니다. 여러 구성 요소에서 텔레메트리를 수집할 수 있지

만 전반적인 워크로드 상태에 대한 인사이트를 제공하는 하위 집합을 선택합니다. 비즈니스 요구 사항의 변화에 따라 시간이 지나면서 워크로드 지표를 조정합니다.

원하는 결과:

- 비즈니스 성과를 반영하는 KPI 달성 여부를 확인하는 지표를 식별했습니다.
- 워크로드 상태에 대한 일관된 뷰를 보여주는 지표가 있습니다.
- 워크로드 지표는 비즈니스 요구 사항의 변화에 따라 주기적으로 평가됩니다.

일반적인 안티 패턴:

- 워크로드의 모든 애플리케이션을 모니터링하고 있지만 워크로드가 비즈니스 성과를 달성하고 있는지 확인할 수 없습니다.
- 워크로드 지표를 정의했지만 비즈니스 KPI와 관련이 없습니다.

이 모범 사례 확립의 이점:

- 비즈니스 성과 달성 여부와 비교하여 워크로드를 측정할 수 있습니다.
- 워크로드가 정상 상태인지 또는 개입이 필요한지 파악할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

이 모범 사례의 목표는 “내 워크로드는 상태가 양호합니까?”라는 질문에 답할 수 있도록 하는 것입니다. 워크로드 상태는 비즈니스 결과 달성과 워크로드의 애플리케이션 및 구성 요소 상태에 따라 결정됩니다. 비즈니스 KPI에서부터 거꾸로 작업하여 지표를 식별합니다. 구성 요소 및 애플리케이션의 주요 지표를 식별합니다. 비즈니스 요구 사항의 변화에 따라 주기적으로 워크로드 지표를 검토합니다.

고객 사례

워크로드 상태는 AnyCompany Retail에서 애플리케이션 및 구성 요소 지표 모음에 의해 결정됩니다. 비즈니스 KPI부터 시작하여 비즈니스 성과를 달성하고 있음을 보여줄 수 있는 주문율과 같은 지표를 식별합니다. 또한 페이지 응답과 같은 주요 애플리케이션 지표와 오픈 데이터베이스 연결과 같은 구성 요소 지표도 포함됩니다. 분기별로 워크로드 지표를 다시 평가하여 워크로드 상태를 결정하는 데 여전히 유효한지 확인합니다.

구현 단계

1. 비즈니스 KPI를 시작으로 비즈니스 성과를 달성하고 있음을 보여주는 지표를 파악합니다. 지표가 없는 KPI가 있는 경우 누락된 비즈니스 KPI에 대한 추가 지표로 워크로드를 구성합니다.
   a. 애플리케이션에서 Amazon CloudWatch로 사용자 지정 지표를 게시할 수 있습니다.
   b. AWS Distro for OpenTelemetry를 사용하면 기존 애플리케이션에서 지표를 수집하고 새 지표를 추가할 수 있습니다.
   c. Enterprise Support 고객은 기술 지원 관리자에게 모니터링 전략 구축 워크숍을 요청할 수 있습니다. 이 워크숍은 워크로드에 대한 관측성 전략을 구축하는 데 도움이 됩니다.
2. 워크로드의 애플리케이션 및 구성 요소에 대한 지표를 식별합니다. 개별 구성 요소 및 애플리케이션의 상태를 보여주는 주요 지표는 무엇입니까? 애플리케이션과 구성 요소는 서로 다른 지표를 내보낼 수 있지만 전반적인 상태를 보여주는 1~3개의 주요 지표를 선택합니다.
3. 워크로드 지표를 주기적으로 평가하는 메커니즘을 구현합니다. 비즈니스 KPI가 변경되면 이해 관계자와 협력하여 워크로드 지표를 업데이트합니다. 워크로드 구성 요소 및 애플리케이션이 진화함에 따라 워크로드 지표를 조정합니다.

구현 계획의 작업 수준: 중간. 애플리케이션에 비즈니스 KPI에 대한 지표를 추가하려면 적당한 노력이 필요할 수 있습니다.

리소스

관련 모범 사례:

- OPS04-BP01 애플리케이션 텔레메트리 구현 - 애플리케이션은 비즈니스 성과를 지원하는 텔레메트리를 내보내야 합니다.
- OPS04-BP02 워크로드 원격 측정 구현 및 구성 - 비즈니스 성과를 지원하는 워크로드 지표를 정의하려면 먼저 텔레메트리를 내보내도록 워크로드를 구성해야 합니다.
- OPS08-BP01 핵심 성과 지표 파악 - 워크로드 지표를 선택하기 전에 먼저 핵심 성과 지표를 파악해야 합니다.

관련 문서:

- Adding metrics and traces to your application on Amazon EKS with AWS Distro for OpenTelemetry, AWS X-Ray, and Amazon CloudWatch(AWS Distro for OpenTelemetry, AWS X-Ray 및 Amazon CloudWatch를 사용하여 Amazon EKS의 애플리케이션에 지표 및 트레이스 추가)
- 운영 가시성을 위한 분산 시스템 계측
- 상태 확인 구현

- 애플리케이션을 효과적으로 모니터링하는 방법
- How to better monitor your custom application metrics using Amazon CloudWatch Agent(Amazon CloudWatch 에이전트를 사용하여 사용자 지정 애플리케이션 지표를 더 잘 모니터링하는 방법)

관련 동영상:

- AWS re:Invent 2020: Monitoring production services at Amazon(AWS re:Invent 2020: Amazon에서 프로덕션 서비스 모니터링)
- AWS re:Invent 2022 - Building observable applications with OpenTelemetry (BOA310)(AWS re:Invent 2022 - OpenTelemetry로 관측 가능한 애플리케이션 구축(BOA310))
- How to Easily Setup Application Monitoring for Your AWS Workloads - AWS Online Tech Talks(AWS 워크로드에 적합한 애플리케이션 모니터링을 쉽게 설정하는 방법 - AWS Online Tech Talks)
- Mastering Observability of Your Serverless Applications - AWS Online Tech Talks(서버리스 애플리케이션의 관측성 마스터링 - AWS Online Tech Talks)

관련 예시:

- One Observability Workshop

관련 서비스:

- Amazon CloudWatch
- AWS Distro for OpenTelemetry

OPS08-BP03 워크로드 지표 수집 및 분석

워크로드 지표를 정기적, 사전적으로 검토하여 추세를 파악하고 대응이 필요한지 판단하며 비즈니스 성과를 달성했는지 검증합니다. 워크로드 애플리케이션 및 구성 요소의 지표를 중앙 위치로 집계합니다. 대시보드 및 분석 도구를 사용하여 텔레메트리를 분석하고 워크로드 상태를 확인합니다. 조직의 이해 관계자와 정기적으로 워크로드 상태를 검토하는 메커니즘을 구현합니다.

원하는 결과:

- 워크로드 지표는 중앙 위치에서 수집됩니다.
- 대시보드 및 분석 도구는 워크로드 상태 추세를 분석하는 데 사용됩니다.

- 조직과 정기적으로 워크로드 지표를 검토합니다.

일반적인 안티 패턴:

- 조직은 서로 다른 두 가지 관측성 플랫폼의 워크로드에서 지표를 수집합니다. 플랫폼이 서로 호환되지 않기 때문에 워크로드 상태를 확인할 수 없습니다.
- 워크로드 구성 요소의 오류 발생률이 서서히 증가하고 있습니다. 조직에서 정기적으로 워크로드 지표를 검토하지 않기 때문에 이러한 추세를 알지 못합니다. 일주일 후에 이 구성 요소에 장애가 발생하여 워크로드가 손상됩니다.

이 모범 사례 확립의 이점:

- 워크로드 상태 및 비즈니스 성과 달성에 대한 인식이 높아졌습니다.
- 워크로드 상태 추세는 시간이 지나면서 생성될 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

중앙 위치에서 워크로드 지표를 수집합니다. 대시보드 및 분석 도구를 사용하여 워크로드 지표를 분석하여 워크로드 상태에 대한 인사이트를 얻고, 워크로드 상태 추세를 생성하고, 비즈니스 성과 달성 여부를 확인합니다. 워크로드 지표를 주기적으로 검토하는 메커니즘을 구현합니다.

고객 사례

AnyCompany Retail은 매주 수요일에 워크로드 지표를 검토합니다. 전사적으로 이해 관계자를 모아 이전 주의 지표를 검토합니다. 회의가 진행되는 동안 분석 도구로 수집한 추세와 인사이트를 강조합니다. 내부 대시보드에는 모든 직원이 보고 검색할 수 있는 주요 워크로드 지표가 함께 게시됩니다.

구현 단계

1. 워크로드 상태와 관련된 워크로드 지표를 식별합니다. 비즈니스 KPI부터 시작하여 워크로드 상태를 전반적으로 보여주는 애플리케이션, 구성 요소 및 플랫폼에 대한 지표를 식별합니다.
   a. Amazon CloudWatch에 사용자 지정 지표를 게시할 수 있습니다. Amazon CloudWatch 에이전트를 활용하여 Amazon EC2 인스턴스 및 온프레미스 서버에서 지표 및 로그를 수집할 수 있습니다.
   b. AWS Distro for OpenTelemetry를 사용하면 기존 애플리케이션에서 지표를 수집하고 새 지표를 추가할 수 있습니다.

   c. Enterprise Support 고객은 기술 지원 관리자에게 모니터링 전략 구축 워크숍을 요청할 수 있습니다. 이 워크숍은 워크로드에 대한 관측성 전략을 구축하는 데 도움이 됩니다.
2. 중앙 플랫폼에서 워크로드 지표를 수집합니다. 워크로드 지표가 서로 다른 플랫폼 간에 분할되면 추세를 분석하고 개발하기 어려울 수 있습니다. 플랫폼에는 대시보드와 분석 기능이 있어야 합니다.
   a. Amazon CloudWatch는 워크로드 지표를 수집하고 저장할 수 있습니다. 다중 계정 토폴로지에서는 로그 아카이브 계정이라고 하는 중앙 로깅 및 모니터링 계정을 사용하는 것이 좋습니다.
3. 워크로드 지표의 통합 대시보드를 구축합니다. 지표를 검토하고 추세를 분석할 때 이 뷰를 사용합니다.
   a. 사용자 지정 CloudWatch 대시보드를 생성하여 통합 뷰에서 워크로드 지표를 수집할 수 있습니다.
4. 워크로드 지표 검토 프로세스를 구현합니다. 주별, 격주별 또는 월별로 기술 및 비기술 인력을 포함한 이해 관계자와 워크로드 지표를 검토합니다. 이러한 검토 세션을 사용하여 추세를 식별하고 워크로드 상태에 대한 인사이트를 확보합니다.

구현 계획의 작업 수준: 높음. 워크로드 지표가 중앙에서 수집되지 않는 경우 지표를 하나의 플랫폼에 통합하기 위해 상당한 투자를 해야 할 수 있습니다.

리소스

관련 모범 사례:

- OPS08-BP01 핵심 성과 지표 파악 - 워크로드 지표를 선택하기 전에 먼저 핵심 성과 지표를 파악해야 합니다.
- OPS08-BP02 워크로드 지표 정의 - 워크로드 지표를 수집 및 분석하기 전에 이를 먼저 정의해야 합니다.

관련 문서:

- Power operational insights with Amazon QuickSight(Amazon QuickSight로 운영 인사이트 강화)
- Using Amazon CloudWatch dashboards custom widgets(Amazon CloudWatch 대시보드 사용자 지정 위젯 사용)

관련 동영상:

- Create Cross Account & Cross Region CloudWatch Dashboards(교차 계정 및 교차 리전 CloudWatch 대시보드 생성)

- Monitor AWS Resources Using Amazon CloudWatch Dashboards(Amazon CloudWatch 대시보드를 사용하여 AWS 리소스 모니터링)

관련 예시:

- AWS 관리 및 거버넌스 도구 워크숍 - CloudWatch 대시보드
- Well-Architected Labs - Level 100: Monitoring with CloudWatch Dashboards(Well-Architected 실습 - 레벨 100: CloudWatch 대시보드로 모니터링)

관련 서비스:

- Amazon CloudWatch
- AWS Distro for OpenTelemetry

OPS08-BP04 워크로드 지표 기준선 설정

워크로드 지표의 기준선을 설정하면 워크로드 상태 및 성능을 이해하는 데 도움이 됩니다. 기준선을 사용하면 성능이 저하되거나 초과되는 애플리케이션 및 구성 요소를 식별할 수 있습니다. 워크로드 기준선은 문제가 발생하기 전에 문제를 완화할 가능성을 높여 줍니다. 기준선은 활동 패턴을 개발하고 지표가 예상 값에서 벗어날 때 이상 탐지를 구현하는 데 기본이 됩니다.

원하는 결과:

- 정상 조건에서 워크로드에 대한 지표의 기준선 수준이 있습니다.
- 워크로드가 정상적으로 작동하는지 확인할 수 있습니다.

일반적인 안티 패턴:

- 새 기능을 배포한 후 요청 지연 시간이 감소합니다. 들어오는 처리된 요청과 전체 지연 시간의 복합 지표에 대한 기준선이 설정되지 않았습니다. 변경으로 인해 개선되었는지 또는 결함이 발생했는지 확인할 수 없습니다.
- 사용자 활동이 갑자기 급증했지만 지표 기준선을 설정하지 않았습니다. 활동 급증은 애플리케이션에서 천천히 메모리 누수로 이어집니다. 결국 워크로드가 오프라인 상태가 됩니다.

이 모범 사례 확립의 이점:

- 주요 구성 요소 및 애플리케이션에 대한 지표를 사용하여 워크로드의 정상적인 활동 패턴을 파악합니다.
- 워크로드, 해당 애플리케이션 및 구성 요소가 정상적으로 작동하는지 또는 개입이 필요한지 확인할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

기록 데이터를 사용하여 워크로드의 애플리케이션 및 구성 요소에 대한 워크로드 지표의 기준선을 설정합니다. 지표 검토 회의 및 문제 해결 과정에서 지표 기준선을 활용합니다. 주기적으로 워크로드 성능을 검토하고 아키텍처가 변경됨에 따라 기준선을 조정합니다.

고객 사례

AnyCompany Retail에서는 모든 구성 요소 및 애플리케이션에 대한 기준선이 설정되어 있습니다. 기록 데이터를 바탕으로 AnyCompany Retail은 2개월의 지표 측정 기간에 걸쳐 워크로드 지표 기준선을 세웠습니다. 2개월마다 기준선을 재평가하고 실제 데이터를 기반으로 조정합니다.

구현 단계

1. 워크로드 지표에서 시작하여 거꾸로 작업하면서 기록 데이터를 기반으로 주요 구성 요소 및 애플리케이션에 대한 지표 기준선을 설정합니다. 구성 요소 또는 애플리케이션당 지표 수를 제한하고 모니터 피로를 방지합니다.
   a. Amazon CloudWatch Metrics Insights를 사용하여 대규모로 지표를 쿼리하고 추세와 패턴을 식별할 수 있습니다.
   b. Amazon CloudWatch 이상 탐지 기능은 기계 학습 알고리즘으로 지표에 대한 동작 패턴을 식별하고, 기준선을 결정하며, 이상 항목을 표면화합니다.
   c. Amazon DevOps Guru는 기계 학습을 사용하여 워크로드의 운영 문제를 탐지하는 기능을 제공합니다.
   d. Enterprise Support 고객은 기술 지원 관리자에게 모니터링 전략 구축 워크숍을 요청할 수 있습니다. 이 워크숍은 워크로드에 대한 관측성 전략을 구축하는 데 도움이 됩니다.
2. 특히 중요한 비즈니스 이벤트 이전에 워크로드 지표 기준선을 주기적으로 검토하는 메커니즘을 마련합니다. 적어도 분기에 한 번은 기록 데이터를 사용하여 워크로드 지표 기준선을 평가합니다. 지표 검토 회의에서 기준선을 사용합니다.

구현 계획의 작업 수준: 낮음. 워크로드 지표를 설정한 후 기준선을 설정하려면 정상적인 동작 패턴을 식별하기에 충분한 데이터를 수집해야 할 수 있습니다.

리소스

관련 모범 사례:

- OPS08-BP02 워크로드 지표 정의 - 기준선을 결정하기 전에 먼저 워크로드 지표를 설정해야 합니다.
- OPS08-BP03 워크로드 지표 수집 및 분석 - 지표 기준선을 설정하기 전에 워크로드 지표를 수집하고 분석해야 합니다.
- OPS08-BP05 워크로드의 예상 활동 패턴 파악 - 이 모범 사례는 기준선 위에 구축되어 사용 추세가 생성됩니다.
- OPS08-BP06 워크로드 성과가 위험한 상태이면 알림 생성 - 임계값을 식별하고 알림을 개발하려면 지표 기준선이 필요합니다.
- OPS08-BP07 워크로드 이상이 감지되면 알림 생성 - 이상 탐지를 위해서는 지표 기준선을 설정해야 합니다.

관련 문서:

- AWS Observability Best Practices - Alarms(AWS 관측성 모범 사례 - 경보)
- 애플리케이션을 효과적으로 모니터링하는 방법
- How to set up CloudWatch Anomaly Detection to set dynamic alarms, automate actions, and drive online sales(CloudWatch 이상 탐지 기능을 설정하여 동적 경보를 설정하고 조치를 자동화하고 온라인 판매를 촉진하는 방법)
- Operationalizing CloudWatch Anomaly Detection(CloudWatch 이상 탐지 운영)

관련 동영상:

- AWS re:Invent 2020: Monitoring production services at Amazon(AWS re:Invent 2020: Amazon에서 프로덕션 서비스 모니터링)
- AWS re:Invent 2021- Get insights from operational metrics at scale with CloudWatch Metrics Insights(AWS re:Invent 2021 - CloudWatch Metrics Insights를 사용하여 대규모 운영 지표에서 인사이트 확보)
- AWS re:Invent 2022 - Developing an observability strategy (COP302)(AWS re:Invent 2022 - 관측성 전략 개발(COP302))

- AWS Summit DC 2022 - Monitoring and observability for modern applications(AWS Summit DC 2022 - 최신 애플리케이션을 위한 모니터링 및 관측성)
- AWS Summit SF 2022 - Full-stack observability and application monitoring with (COP310)AWS(AWS Summit SF 2022 - AWS를 사용한 전체 스택 관측성 및 애플리케이션 모니터링(COP310))

관련 예시:

- AWS CloudTrail 및 Amazon CloudWatch 통합 워크숍

관련 서비스:

- Amazon CloudWatch
- Amazon DevOps Guru

OPS08-BP05 워크로드의 예상 활동 패턴 파악

필요한 경우 적절히 대응할 수 있도록 비정상적인 동작을 식별할 워크로드 활동 패턴을 설정합니다.

CloudWatch에서는 CloudWatch 이상 탐지 기능을 통해 통계 및 기계 학습 알고리즘을 적용해 정상 지표 동작을 나타내는 예상되는 값의 범위를 생성합니다.

Amazon DevOps Guru 를 사용하여 이벤트 상관 관계, 로그 분석, 기계 학습 적용을 통해 워크로드 원격 측정을 분석하여 비정상적인 동작을 식별할 수 있습니다. 예기치 않은 동작이 감지되면 관련 지표 및 이벤트와 함께 동작을 해결하기 위한 권장 사항이 제공됩니다.

일반적인 안티 패턴:

- 네트워크 사용률 로그를 검토하여 네트워크 사용률이 오전 11시 30분에서 오후 1시 30분 사이에 증가한 다음 오후 4시 30분부터 오후 6시까지 다시 증가했음을 확인합니다. 정상으로 간주되어야 하는지 여부를 알 수 없습니다.
- 웹 서버는 매일 밤 3시에 재부팅됩니다. 예상된 동작인지 알 수 없습니다.

이 모범 사례 수립의 이점: 행동 패턴을 파악하면 예기치 않은 행동을 인식하고 필요한 경우 조치를 취할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 워크로드의 예상 활동 패턴 파악: 워크로드 활동 패턴을 설정하여 워크로드의 동작이 필요한 값의 범위를 벗어나는 경우를 확인합니다. 그러면 필요 시 적절하게 대응할 수 있습니다.

리소스

관련 문서:

- Amazon DevOps Guru
- CloudWatch 이상 탐지

OPS08-BP06 워크로드 성과가 위험한 상태이면 알림 생성

워크로드 성과가 위험한 상태이면 필요 시 적절히 대응할 수 있도록 알림을 생성합니다.

이전에는 자동화된 응답을 트리거하는 데 사용할 수 있는 이벤트 또는 경보를 알릴 수 있는 지표 임계값을 식별했습니다.

AWS에서는 Amazon CloudWatch Synthetics를 통해 고객과 동일한 작업을 수행하여 엔드포인트 및 API를 모니터링하는 canary 스크립트를 작성할 수 있습니다. 생성된 원격 측정과 획득한 인사이트를 바탕으로 고객이 영향을 받기 전에 문제를 식별할 수 있습니다.

또한 CloudWatch Logs Insights 에서 특별히 구축된 쿼리 언어를 사용해 로그 데이터를 대화식으로 검색하고 분석할 수 있습니다. CloudWatch Logs Insights는 자동으로 AWS 서비스에서 로그의 필드와 JSON 형식의 사용자 지정 로그 이벤트를 검색합니다. 그러면 로그 볼륨 및 쿼리 복잡성에 대한 지원을 확장하고 몇 초 안에 답변을 제공하므로 인시던트의 원인을 파악하는 데 도움이 됩니다.

일반적인 안티 패턴:

- 네트워크에 연결되어 있지 않습니다. 아무도 이 상황을 모릅니다. 아무도 이유를 파악하려고 하거나 연결 복원 조치를 취하고 있지 않습니다.
- 패치 후 영구 인스턴스를 사용할 수 없게 되어 사용자 작업이 중단됩니다. 사용자가 지원 사례를 개설했습니다. 아무도 알림을 받지 않았습니다. 아무도 조치를 취하지 않습니다.

이 모범 사례 정립의 이점: 비즈니스 성과가 위험에 처하고 조치를 취해야 한다는 사실을 파악함으로써 인시던트의 영향을 예방하거나 완화할 수 있는 기회를 얻게 됩니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 워크로드 성과가 위험한 상태이면 알림 생성: 워크로드 성과가 위험한 상태이면 알림을 생성합니다. 그러면 필요할 때 적절하게 대응할 수 있습니다.
  - Amazon CloudWatch Events란 무엇입니까?
  - Amazon CloudWatch 경보 생성
  - Amazon SNS 알림을 사용하여 Lambda 함수 호출

리소스

관련 문서:

- Amazon CloudWatch Synthetics를 통해
- CloudWatch Logs Insights
- Amazon CloudWatch 경보 생성
- Amazon SNS 알림을 사용하여 Lambda 함수 호출
- Amazon CloudWatch Events란 무엇입니까?

OPS08-BP07 워크로드 이상이 감지되면 알림 생성

워크로드에서 이상이 감지되면 필요 시 적절히 대응할 수 있도록 알림을 생성합니다.

시간에 따른 워크로드 지표를 분석하면 이벤트를 정의하거나 이벤트 응답으로 경보를 울리기 위해 정량화할 수 있는 동작의 패턴을 설정할 수 있습니다.

훈련된 후에는 CloudWatch 이상 탐지 기능을 사용하여 탐지된 이상 현상에 대한 경보 를 생성하거나 비교를 위해 지표 데이터의 그래프 에서 중첩된 예상되는 값을 제공할 수 있습니다.

일반적인 안티 패턴:

- 소매 웹 사이트 매출이 갑자기 급증했습니다. 아무도 이 상황을 모릅니다. 아무도 이러한 급증을 초래하는 원인을 파악하려고 하지 않습니다. 아무도 추가 로드 발생 시에 훌륭한 고객 경험을 보장하기 위한 조치를 취하고 있지 않습니다.
- 패치를 적용한 후 영구 서버가 재부팅되어 사용자 작업이 중단되는 경우가 많습니다. 서버는 일반적으로 최대 3회까지 재부팅되지만 그 이상 부팅되지는 않습니다. 아무도 이 상황을 모릅니다. 아무도 이런 일이 발생하는 이유를 파악하려고 하지 않습니다.

이 모범 사례 정립의 이점: 워크로드 동작의 패턴을 파악하면 예기치 않은 동작을 식별하고 필요 시 조치를 취할 수 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 워크로드에 이상이 감지되면 알림 생성: 워크로드에서 이상 상태가 감지되면 알림을 생성합니다. 그러면 필요할 때 적절하게 대응할 수 있습니다.
  - Amazon CloudWatch Events란 무엇입니까?
  - Amazon CloudWatch 경보 생성
  - Amazon SNS 알림을 사용하여 Lambda 함수 호출

리소스

관련 문서:

- Amazon CloudWatch 경보 생성
- CloudWatch 이상 탐지
- Amazon SNS 알림을 사용하여 Lambda 함수 호출
- Amazon CloudWatch Events란 무엇입니까?

OPS08-BP08 성과 달성 여부와 KPI 및 지표의 효율성 확인

워크로드 운영을 실무 수준에서 확인할 수 있는 보기를 생성합니다. 그러면 요구를 충족하고 있는지를 확인할 수 있으며 업무 목표 달성을 위해 개선해야 하는 영역을 파악할 수 있습니다. 또한 KPI와 지표의 효율성을 확인하고 필요한 경우 KPI/지표를 수정합니다.

AWS는 AWS 서비스 API 및 SDK(예: Grafana, Kibana, Logstash)를 통해 타사 로그 분석 시스템 및 비즈니스 인텔리전스 도구도 지원합니다.

일반적인 안티 패턴:

- 페이지 응답 시간은 고객 만족도에 기여하는 것으로 간주된 적은 없습니다. 페이지 응답 시간에 대한 지표 또는 임계값을 설정한 적이 없습니다. 고객이 느린 속도에 대해 불만을 제기하고 있습니다.
- 최소 응답 시간 목표를 달성하지 않았습니다. 응답 시간 개선을 위해 애플리케이션 서버를 스케일 업했습니다. 이제 상당한 마진으로 응답 시간 목표를 초과 달성하고 비용을 지불하고 있는 미사용 용량도 상당히 확보하게 됩니다.

이 모범 사례 수립의 이점: KPI와 지표를 검토하고 수정하면 워크로드가 어떻게 비즈니스 성과 달성을 지원하는지 이해하고 비즈니스 목표 달성을 위해 개선이 필요한 영역을 식별할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 성과 달성 여부와 KPI 및 지표의 효율성 확인: 워크로드 운영을 실무 수준에서 확인할 수 있는 보기를 생성합니다. 그러면 요구를 충족하고 있는지 확인할 수 있으며 비즈니스 목표 달성을 위해 개선해야 하는 영역을 파악할 수 있습니다. 또한 KPI와 지표의 효율성을 확인하고 필요한 경우 KPI/지표를 수정합니다.
  - Amazon CloudWatch 대시보드 사용
  - 로그 분석이란 무엇일까요?

리소스

관련 문서:

- Amazon CloudWatch 대시보드 사용
- 로그 분석이란 무엇일까요?

OPS 9 운영 상태를 어떻게 파악하십니까?

운영 지표를 정의, 캡처 및 분석하면 운영 이벤트에 대한 가시성을 확보하여 적절한 조치를 취할 수 있습니다.

모범 사례

- OPS09-BP01 핵심 성과 지표 파악
- OPS09-BP02 운영 지표 정의
- OPS09-BP03 운영 지표 수집 및 분석
- OPS09-BP04 운영 지표 기준 설정
- OPS09-BP05 운영의 예상 활동 패턴 파악
- OPS09-BP06 운영 성과가 위험한 상태이면 알림 생성
- OPS09-BP07 운영 이상이 감지되면 알림 생성
- OPS09-BP08 성과 달성 여부와 KPI 및 지표의 효율성 확인

OPS09-BP01 핵심 성과 지표 파악

원하는 비즈니스 성과(예: 새로운 기능 제공)와 고객 성과(예: 고객 지원 사례)를 기반으로 핵심 성과 지표(KPI)를 파악합니다. 그리고 KPI를 평가하여 운영의 성공 여부를 결정합니다.

일반적인 안티 패턴:

- 경영진으로부터 운영이 얼마나 성공적으로 비즈니스 목표를 달성하고 있는지에 대한 질문을 받지만 성공 여부를 판단하기 위한 준거 기준이 없습니다.
- 유지 관리 기간이 비즈니스 성과에 영향을 미치는지 판단할 수 없습니다.

이 모범 사례 정립의 이점: 핵심 성과 지표를 파악하면 운영 상태 및 성공 여부를 테스트하여 비즈니스 성과를 달성할 수 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 핵심 성과 지표 파악: 원하는 비즈니스 성과와 고객 성과를 기준으로 핵심 성과 지표(KPI)를 확인합니다. 그리고 KPI를 평가하여 운영의 성공 여부를 결정합니다.

OPS09-BP02 운영 지표 정의

KPI 성과(예: 성공한 배포와 실패한 배포)를 측정하는 데 사용할 운영 지표를 정의합니다. 운영 활동 상태(예: 인시던트의 MTTD(평균 탐지 시간) 및 인시던트의 MTTR(평균 복구 시간))를 측정하는 데 사용할 운영 지표를 정의합니다. 그런 다음, 해당 지표를 평가해 운영 과정에서 적절한 성과를 달성할 수 있는지를 확인하고 운영 활동 상태를 파악합니다.

일반적인 안티 패턴:

- 운영 지표는 팀이 합리적이라고 생각하는 것을 기반으로 합니다.
- 지표 계산에 잘못된 결과를 산출하는 오류가 있습니다.
- 작업 활동에 대해 정의된 지표가 없습니다.

이 모범 사례 정립의 이점: 운영 지표를 정의하고 평가하여 운영 활동의 상태를 파악하고 비즈니스 성과 달성 여부를 측정할 수 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 운영 지표 정의: 운영 지표를 정의하여 KPI의 성과를 측정합니다. 운영 및 해당 활동의 상태를 측정하는 데 사용할 운영 지표를 정의합니다. 그런 다음 해당 지표를 평가해 운영 과정에서 적절한 성과를 달성할 수 있는지를 확인하고 운영 상태를 파악합니다.
  - 사용자 지정 지표 게시
  - 로그 데이터 검색 및 필터링
  - Amazon CloudWatch 지표 및 차원 참조

리소스

관련 문서:

- AWS Answers: 중앙 집중식 로깅
- Amazon CloudWatch 지표 및 차원 참조
- Amazon CloudWatch Events를 사용하여 파이프라인 상태에서 변경 감지 및 대처
- 사용자 지정 지표 게시
- 로그 데이터 검색 및 필터링

관련 동영상:

- 모니터링 플랜 세우기

OPS09-BP03 운영 지표 수집 및 분석

지표를 정기적으로 사전 예방 차원에서 점검하여 추세를 확인하고 어느 부분에 적절한 대응이 필요한지 파악합니다.

운영 활동 및 운영 API 호출의 실행에서 CloudWatch Logs와 같은 서비스로 로그 데이터를 집계해야 합니다. 운영 활동의 성과에 대한 인사이트를 얻을 수 있도록 필요한 로그 콘텐츠를 관찰하여 지표를 생성합니다.

AWS에서는 Amazon S3로 로그 데이터를 내보내거나 또는 장기 보관을 위해 을 Amazon S3 로 로그를 직접 전송할 수 있습니다. 여러분은 AWS Glue를 사용하여 다음에 관련 메타데이터를 저장하면서 분석을 위해 Amazon S3의 로그 데이터를 검색 및 준비할 수 있습니다. AWSAWS Glue Data Catalog. Amazon Athena에서 AWS Glue와의 기본 통합을 통해 로그 데이터를 분석하고 표준 SQL을 사용해 쿼

리할 수 있습니다. 여러분은 Amazon QuickSight 와 같은 비즈니스 인텔리전스 도구를 사용하여 데이터를 시각화하고 탐색하며 분석할 수 있습니다.

일반적인 안티 패턴:

- 새로운 기능의 일관된 제공이 핵심 성능 지표로 간주됩니다. 배포가 발생하는 빈도를 측정할 방법이 없습니다.
- 배포, 롤백된 배포, 패치 및 롤백된 패치를 로깅하여 작업 활동을 추적하지만 아무도 지표를 검토하지 않습니다.
- 손실된 데이터베이스를 15분 내에 복원해야 하는 복구 시간 목표가 있습니다. 이 목표는 시스템이 배포되고 사용자가 없을 때 정의되었습니다. 현재 1만 명의 사용자를 보유하고 있으며, 운영한 지 2년이 지났습니다. 최근 복원에 2시간이 넘게 걸렸습니다. 이는 기록되지 않았으며 아무도 모릅니다.

이 모범 사례 수립의 이점: 운영 지표를 수집하고 분석하면 운영 상태를 파악하고 운영 또는 비즈니스 성과 달성에 영향을 미칠 수 있는 추세에 대한 인사이트를 얻을 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 운영 지표 수집 및 분석: 사전 예방 차원에서 지표를 정기적으로 점검하여 추세를 확인하고 어느 부분에 적절한 대응이 필요한지를 파악합니다.
  - Amazon CloudWatch 지표 사용
  - Amazon CloudWatch 지표 및 차원 참조
  - CloudWatch 에이전트를 사용하여 Amazon EC2 인스턴스 및 온프레미스 서버에서 지표 및 로그 수집

리소스

관련 문서:

- Amazon Athena
- Amazon CloudWatch 지표 및 차원 참조
- Amazon QuickSight
- AWS Glue
- AWSAWS Glue Data Catalog

- CloudWatch 에이전트를 사용하여 Amazon EC2 인스턴스 및 온프레미스 서버에서 지표 및 로그 수집
- Amazon CloudWatch 지표 사용

OPS09-BP04 운영 지표 기준 설정

지표의 기준을 설정해 성능이 기준보다 높은/낮은 운영 활동을 확인하고 각 프로세스의 성능을 비교할 수 있는 기준으로 필요한 값을 제공합니다.

일반적인 안티 패턴:

- 예상되는 배포 시간을 묻는 메시지가 표시됩니다. 배포에 걸리는 시간을 측정하지 않았으며 예상 시간을 확인할 수 없습니다.
- 애플리케이션 서버 문제에서 복구하는 데 얼마나 걸릴지 묻는 메시지가 표시됩니다. 첫 번째 고객 연락처에서 복구하는 데 걸리는 시간에 대한 정보가 없습니다. 모니터링을 통해 첫 번째 문제 식별에서 복구하는 데 걸리는 시간에 대한 정보가 없습니다.
- 주말 동안 몇 명의 지원 인력이 필요한지에 대한 질문을 받았습니다. 주말 동안 몇 가지 지원 사례가 일반적인지 모르며 추정을 제공할 수 없습니다.
- 손실된 데이터베이스를 15분 내에 복원해야 하는 복구 시간 목표가 있습니다. 이 목표는 시스템이 배포되고 사용자가 없을 때 정의되었습니다. 현재 1만 명의 사용자를 보유하고 있으며, 운영한 지 2년이 지났습니다. 데이터베이스에 대한 복원 시간이 어떻게 변경되었는지에 대한 정보가 없습니다.

이 모범 사례 정립의 이점: 기준 지표 값을 정의하면 현재 지표 값과 지표 추세를 평가하여 조치가 필요한지 여부를 결정할 수 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 운영의 예상 활동 패턴 파악: 운영 활동 패턴을 설정하여 동작이 필요한 값의 범위를 벗어나는 경우를 확인합니다. 그러면 필요 시 적절하게 대응할 수 있습니다.

OPS09-BP05 운영의 예상 활동 패턴 파악

필요한 경우 적절하게 대응할 수 있도록 비정상적인 활동을 식별할 운영 활동 패턴을 설정합니다.

일반적인 안티 패턴:

- 최근에 배포 실패율이 크게 증가했습니다. 각 실패를 독립적으로 해결합니다. 실패 원인이 배포 관리 시스템에 익숙하지 않은 신입 직원의 배포임을 인식하지 못합니다.

이 모범 사례 수립의 이점: 동작 패턴을 파악하면 예기치 않은 동작을 확인하고 필요 시 조치를 취할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 운영의 예상 활동 패턴 파악: 운영 활동 패턴을 설정하여 동작이 필요한 값의 범위를 벗어나는 경우를 확인합니다. 그러면 필요 시 적절하게 대응할 수 있습니다.

OPS09-BP06 운영 성과가 위험한 상태이면 알림 생성

운영 성과에 위험이 있을 때마다 알림이 발생하고 적절한 조치가 이루어져야 합니다. 운영 성과는 프로덕션의 워크로드를 지원하는 모든 활동입니다. 여기에는 새로운 버전의 애플리케이션 배포부터 중단 복구까지의 모든 것이 포함됩니다. 운영 성과는 비즈니스 성과와 동일한 중요성이 있는 것으로 다루어야 합니다.

소프트웨어 팀은 주요 운영 지표 및 활동을 파악하고 이를 위한 알림을 구축해야 합니다. 알림은 적시에 이루어지고 실행 가능해야 합니다. 알림이 발생하면 해당 런북 또는 플레이북에 대한 참조가 포함되어 있어야 합니다. 해당 조치가 없는 알림은 알림 피로감으로 이어집니다.

원하는 결과: 운영 활동에 위험이 있는 경우 조치를 취할 수 있도록 알림이 전송됩니다. 알림에는 알림이 발생한 이유에 대한 컨텍스트와 조사를 위한 플레이북 또는 완화를 위한 런북이 명시됩니다. 가능한 경우, 런북이 자동화되고 알림이 전송됩니다.

일반적인 안티 패턴:

- 인시던트를 조사 중이며 지원 사례가 접수되고 있습니다. 지원 사례가 서비스 수준 계약(SLA)을 침해하지만 어떤 알림도 발생하지 않습니다.
- 자정으로 예약된 프로덕션 배포가 막바지 코드 변경으로 인해 지연됩니다. 알림이 발생하지 않고 배포가 중단됩니다.
- 프로덕션 중단이 발생하지만 알림이 전송되지 않습니다.
- 배포 시간이 지속적으로 예상보다 늦어지고 있습니다. 조사를 위한 조치가 이루어지지 않습니다.

이 모범 사례 확립의 이점:

- 운영 성과에 위험이 있을 때 알림을 생성함으로써, 문제 발생 전에 워크로드를 지원할 수 있는 기능이 확대됩니다.
- 우수한 운영 성과를 통해 비즈니스 성과가 개선됩니다.
- 운영 문제의 탐지 및 개선 조치가 향상됩니다.
- 전반적인 운영 상태가 향상됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

운영 성과에 대한 알림을 생성하기 전에 운영 성과를 정의해야 합니다. 조직에 가장 중요한 운영 활동이 어떤 것인지 정의하는 것으로 시작합니다. 2시간 이내에 프로덕션에 배포하거나 정해진 시간 내에 지원 사례에 응답합니까? 조직은 핵심 운영 활동 및 그 측정 방법을 정의하여 모니터링, 개선 및 알림이 이루어지도록 해야 합니다. 워크로드 및 운영 텔레메트리를 저장 및 분석할 중앙 위치가 필요합니다. 운영 성과가 위험할 경우 동일한 메커니즘에서 알림을 생성할 수 있어야 합니다.

고객 사례

AnyCompany Retail에서 일상적인 배포 작업 중 CloudWatch 알람이 트리거되었습니다. 배포 리드 타임에 위반이 발생했습니다. Amazon EventBridge가 AWS Systems Manager OpsCenter에서 OpsItem을 생성했습니다. 클라우드 운영 팀이 플레이북을 사용하여 문제를 조사했고, 스키마 변경이 예상보다 오래 걸렸음을 파악했습니다. 당직 근무 중인 개발자에게 알림이 생성되고 배포를 계속 모니터링했습니다. 배포가 완료된 후 클라우드 운영 팀이 OpsItem을 해결했습니다. 사후 기간 동안 팀에서 인시던트를 분석합니다.

구현 단계

1. 운영 KPI, 지표 및 활동을 파악하지 않았다면 이 질문에 대한 앞선 모범 사례를 구현하는 것이 좋습니다(OPS09-BP01 - OPS09-BP05).
    - AWS Support 고객(Enterprise Support 고객)은 기술 지원 관리자에게 운영 KPI 워크숍 을 요청할 수 있습니다. 추가 비용 없이 제공되는 이러한 협업 워크숍을 통해 비즈니스 목표에 따른 운영 KPI 및 지표를 정의할 수 있습니다. 자세한 내용은 기술 지원 관리자에게 문의하시기 바랍니다.
2. 운영 활동, KPI 및 지표를 설정했다면 관찰성 플랫폼에서 알림을 구성해야 합니다. 알림은 플레이북 또는 런북과 같이 이와 연관된 활동이 있어야 합니다. 활동이 없는 알림은 피하는 것이 좋습니다.
3. 시간이 지나면서 운영 지표, KPI 및 활동을 평가하여 개선 영역을 파악합니다. 운영자의 런북 및 플레이북에서 피드백을 수집하여 알림 대응 개선 영역을 파악합니다.

4. 알림은 오탐으로 플래그를 지정하는 메커니즘을 포함해야 하며 이것은 지표 임계값의 검토로 이어져야 합니다.

구현 계획의 작업 수준: 보통. 이 모범 사례를 구현하기 전에 갖춰야 하는 몇 가지 모범 사례가 있습니다. 운영 활동이 파악되고 운영 KPI가 설정되었다면 알림을 설정해야 합니다.

리소스

관련 모범 사례:

- OPS02-BP03 운영 활동에서 성능을 담당하는 소유자 식별: 모든 운영 활동 및 성과는 책임이 있는 식별된 소유자가 있어야 합니다. 이 소유자는 성과가 위험할 때 알림을 받는 대상입니다.
- OPS03-BP02 팀원에게 성과 달성이 위태로울 때 조치를 취할 수 있는 권한 부여: 알림이 발생하면 팀은 문제를 해결하기 위한 조치를 취하는 에이전시가 있어야 합니다.
- OPS09-BP01 핵심 성과 지표 파악: 운영 성과에 대한 알림은 운영 KPI를 파악하는 것에서 시작합니다.
- OPS09-BP02 운영 지표 정의: 알림 생성을 시작하기 전에 이 모범 사례를 확립합니다.
- OPS09-BP03 운영 지표 수집 및 분석: 알림을 구축하려면 중앙에서 수집하는 운영 지표가 필요합니다.
- OPS09-BP04 운영 지표 기준 설정: 운영 지표 기준은 알림을 조정하고 알림 피로감을 예방하기 위한 기능을 제공합니다.
- OPS09-BP05 운영의 예상 활동 패턴 파악: 운영 이벤트의 활동 패턴을 이해함으로써 알림의 정확도를 개선할 수 있습니다.
- OPS09-BP08 성과 달성 여부와 KPI 및 지표의 효율성 확인: 운영 성과 달성을 평가하여 KPI 및 지표가 유효한지 확인합니다.
- OPS10-BP02 알림별 프로세스 마련: 모든 알림에는 연관된 런북이나 플레이북이 있어야 하며 알림을 받는 사람에게 컨텍스트를 제공해야 합니다.
- OPS11-BP02 인시던트 사후 분석 수행: 알림 후에는 인시던트 사후 분석을 수행하여 개선이 필요한 영역을 파악합니다.

관련 문서:

- AWS 배포 파이프라인 참조 아키텍처: 애플리케이션 파이프라인 아키텍처
- GitLab: Agile/DevOps Metrics 시작하기

관련 동영상:

- AWS Systems Manager OpsCenter를 사용하여 운영 문제 집계 및 해결
- AWS Systems Manager OpsCenter와 Amazon CloudWatch 알람의 통합
- Amazon EventBridge를 사용하여 AWS Systems Manager OpsCenter에 데이터 소스 통합

관련 예시:

- Amazon EC2 Systems Manager Automation 및 AWS Health를 사용하여 Amazon EC2 알림 등에 대한 개선 조치 자동화
- 2022년 AWS 관리 및 거버넌스 도구 워크숍 - 운영
- AWS의 DevOps 모니터링 대시보드를 사용한 지표 수집, 분석 및 시각화

관련 서비스:

- Amazon EventBridge
- AWS Support 사전 예방 서비스 - 운영 KPI 워크숍
- AWS Systems Manager OpsCenter
- CloudWatch 이벤트

OPS09-BP07 운영 이상이 감지되면 알림 생성

운영에서 이상이 감지되면 필요 시 적절히 대응할 수 있도록 알림을 생성합니다.

시간에 따른 운영 지표를 분석하면 이벤트를 정의하거나 이벤트 응답으로 경보를 울리기 위해 정량화할 수 있는 동작의 패턴을 설정할 수 있습니다.

훈련된 후에는 CloudWatch 이상 탐지 기능을 사용하여 탐지된 이상 현상에 대한 경보 를 생성하거나 비교를 위해 지표 데이터의 그래프 에서 중첩된 예상되는 값을 제공할 수 있습니다.

Amazon DevOps Guru 를 사용하여 이벤트 상관 관계, 로그 분석, 기계 학습 적용을 통해 워크로드 원격 측정을 분석하여 비정상적인 동작을 식별할 수 있습니다. 유효한 인사이트가 관련 데이터, 권장 사항과 함께 표시됩니다.

일반적인 안티 패턴:

- 인스턴스 플릿에 패치를 적용하고 있습니다. 테스트 환경에서 패치를 성공적으로 테스트했습니다. 플릿에서 많은 비율의 인스턴스에 대해 패치가 실패하고 있습니다. 아무 작업도 하지 않습니다.

- 금요일이 끝나면 배포가 시작된다는 점에 유의하십시오. 조직에 화요일과 목요일에 사전 정의된 유지 관리 기간이 있습니다. 아무 작업도 하지 않습니다.

이 모범 사례 정립의 이점: 운영 동작의 패턴을 파악하면 예기치 않은 동작을 식별하고 필요 시 조치를 취할 수 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 운영에 이상이 감지되면 알림 생성: 운영에서 이상 상태가 감지되면 알림을 생성합니다. 그러면 필요할 때 적절하게 대응할 수 있습니다.
  - Amazon CloudWatch Events란 무엇입니까?
  - Amazon CloudWatch 경보 생성
  - Amazon SNS 알림을 사용하여 Lambda 함수 호출

리소스

관련 문서:

- Amazon DevOps Guru
- CloudWatch 이상 탐지
- Amazon CloudWatch 경보 생성
- Amazon CloudWatch Events를 사용하여 파이프라인 상태에서 변경 감지 및 대처
- Amazon SNS 알림을 사용하여 Lambda 함수 호출
- Amazon CloudWatch Events란 무엇입니까?

OPS09-BP08 성과 달성 여부와 KPI 및 지표의 효율성 확인

운영 활동을 실무 수준에서 확인할 수 있는 보기를 생성합니다. 그러면 요구를 충족하고 있는지를 확인할 수 있으며 업무 목표 달성을 위해 개선해야 하는 영역을 파악할 수 있습니다. 또한 KPI와 지표의 효율성을 확인하고 필요한 경우 KPI/지표를 수정합니다.

AWS는 AWS 서비스 API 및 SDK(예: Grafana, Kibana, Logstash)를 통해 타사 로그 분석 시스템 및 비즈니스 인텔리전스 도구도 지원합니다.

일반적인 안티 패턴:

- 개발 팀 수가 증가함에 따라 배포 빈도가 증가했습니다. 정의된 예상 배포 수는 매주 한 번입니다. 매일 정기적으로 배포하고 있습니다. 배포 시스템의 문제이고 배포가 불가능한 경우 며칠 동안 감지되지 않습니다.
- 이전에 비즈니스에서 월요일부터 금요일까지 핵심 업무 시간 동안에만 지원을 제공했습니다. 인시던트에 대해 '익일(영업일 기준)' 응답 시간 목표를 설정했습니다. 최근에 2시간의 응답 시간을 목표로 연중무휴 24시간 지원 서비스를 제공하기 시작했습니다. 야간에 근무하는 직원은 과중한 업무에 압도되고 고객은 만족하지 않습니다. '익일(영업일 기준)' 목표에 대해 보고하기 때문에 인시던트 대응 시간에 문제가 있다는 징후는 없습니다.

이 모범 사례 정립의 이점: KPI와 지표를 검토하고 수정하면 워크로드가 어떻게 비즈니스 성과 달성을 지원하는지 이해하고 비즈니스 목표 달성을 위해 개선이 필요한 영역을 식별할 수 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 성과 달성 여부와 KPI 및 지표의 효율성 확인: 운영 활동을 실무 수준에서 확인할 수 있는 보기를 생성합니다. 그러면 요구를 충족하고 있는지 확인할 수 있으며 비즈니스 목표 달성을 위해 개선해야 하는 영역을 파악할 수 있습니다. 또한 KPI와 지표의 효율성을 확인하고 필요한 경우 KPI/지표를 수정합니다.
  - Amazon CloudWatch 대시보드 사용
  - 로그 분석이란 무엇일까요?

리소스

관련 문서:

- Amazon CloudWatch 대시보드 사용
- 로그 분석이란 무엇일까요?

OPS 10 워크로드 및 운영 이벤트를 어떻게 관리하십니까?

이벤트로 인해 워크로드가 중단될 가능성을 최소화할 수 있도록 이벤트 대응을 위한 절차를 준비/확인합니다.

모범 사례

- OPS10-BP01 이벤트, 인시던트 및 문제 관리 프로세스 사용

- OPS10-BP02 알림별 프로세스 마련
- OPS10-BP03 비즈니스 영향을 기반으로 운영 이벤트의 우선순위 지정
- OPS10-BP04 에스컬레이션 경로 정의
- OPS10-BP05 중단에 대한 고객 커뮤니케이션 계획 정의
- OPS10-BP06 대시보드를 통해 상태 전달
- OPS10-BP07 이벤트 대응 자동화

OPS10-BP01 이벤트, 인시던트 및 문제 관리 프로세스 사용

조직에는 이벤트, 인시던트 및 문제를 처리하기 위한 프로세스가 있습니다. 이벤트 는 워크로드에서 발생하는 일이지만 개입이 필요하지 않을 수 있습니다. 인시던트 는 개입이 필요한 이벤트입니다. 문제 는 개입이 필요하거나 해결할 수 없는 반복 이벤트입니다. 이러한 이벤트가 비즈니스에 미치는 영향을 줄일 수 있는 프로세스가 필요하며 적절하게 대응하는지 확인해야 합니다.

인시던트 및 문제가 워크로드에 발생하면 처리하기 위한 프로세스가 필요합니다. 이해관계자에게 이벤트 상태를 어떻게 전달할 수 있을까요? 대응 주도를 감독하는 사람은 누구인가요? 이벤트로 인한 피해를 줄이기 위해 사용하는 도구는 무엇인가요? 이는 확실한 대응 프로세스를 갖추기 위해 답변해야 하는 질문의 몇 가지 예입니다.

프로세스는 중앙 위치에 문서화해 두어야 하며 워크로드와 관련된 사람은 누구나 사용할 수 있어야 합니다. 중앙 Wiki 또는 문서 저장소가 없다면 버전 관리 리포지토리를 사용할 수 있습니다. 프로세스 발전에 맞춰 이러한 계획을 최신 상태로 유지하게 됩니다.

문제는 자동화 후보입니다. 이러한 이벤트는 혁신 역량에서 시간을 빼앗아 갑니다. 문제를 완화하기 위한 반복 프로세스를 구축하는 것부터 시작하세요. 시간이 흐른 후에는 완화 프로세스 자동화 또는 기본 문제 수정에 집중하세요. 그러면 워크로드 개선에 투자할 시간을 확보할 수 있습니다.

원하는 결과: 조직에는 이벤트, 인시던트 및 문제를 처리하기 위한 프로세스가 있습니다. 이러한 프로세스는 문서화되어 중앙 위치에 저장되고 프로세스가 변함에 따라 업데이트됩니다.

일반적인 안티 패턴:

- 인시던트가 주말에 발생했는데 당직 근무 중인 엔지니어가 무엇을 해야 할지 모릅니다.
- 고객은 여러분에게 애플리케이션이 다운되었다는 이메일을 보내고 여러분은 서버를 재부팅하여 문제를 해결합니다. 이러한 상황이 빈번하게 발생합니다.
- 한 가지 인시던트를 여러 팀에서 해결하기 위해 따로 노력합니다.
- 워크로드에서 배포가 있었는데, 기록되지 않습니다.

이 모범 사례 확립의 이점:

- 워크로드에서 이벤트를 감사 추적합니다.
- 인시던트에서 복구 시간이 단축됩니다.
- 팀원이 일관된 방식으로 인시던트와 문제를 해결할 수 있습니다.
- 인시던트를 조사할 때 더욱 통합된 노력을 기울일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

이 모범 사례를 구현하면 워크로드 이벤트를 추적하게 됩니다. 인시던트 및 문제를 처리하기 위한 프로세스를 보유하게 되며, 프로세스는 문서화되고 공유되며 자주 업데이트됩니다. 문제가 파악되면 우선순위가 지정되고 해결됩니다.

고객 사례

AnyCompany Retail은 이벤트, 인시던트, 문제 관리를 위한 프로세스 전용 내부 Wiki를 갖추고 있습니다. 모든 이벤트는 다음 프로그램으로 전송됩니다. Amazon EventBridge. 문제는 AWS Systems Manager OpsCenter 에서 OpsItems로 식별되고 문제를 해결하도록 우선순위가 지정되어 획일적인 작업이 줄어듭니다. 프로세스가 변경되면 내부 Wiki에서 업데이트됩니다. 프로세스는 AWS Systems Manager Incident Manager 을(를) 사용하여 인시던트를 관리하고 피해를 줄이기 위한 작업을 조정합니다.

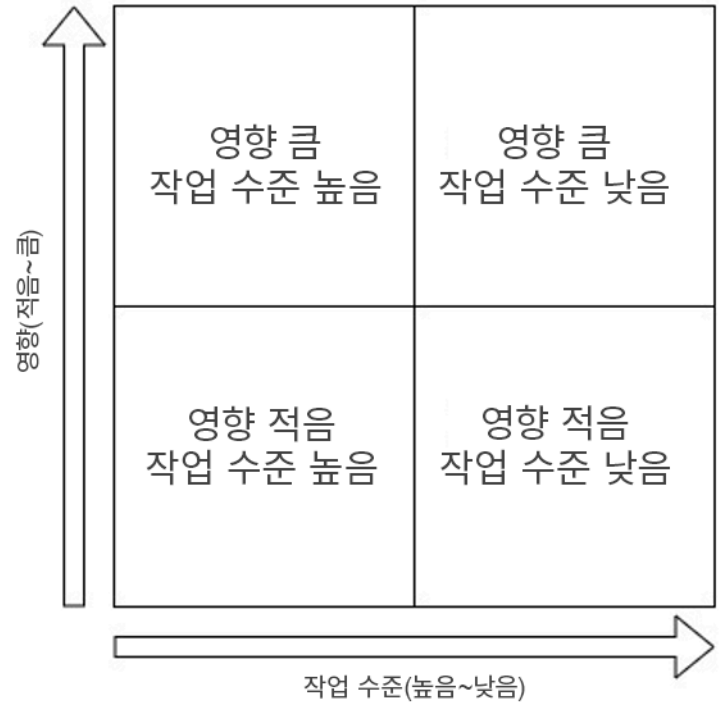
구현 단계

1. 이벤트
   - 인간의 개입이 필요 없는 경우에도 워크로드에서 발생한 이벤트를 추적합니다.
   - 워크로드 이해관계자와 협력하여 추적해야 할 이벤트 목록을 작성합니다. 이러한 이벤트의 몇 가지 예시로는 완료된 배포 또는 성공적인 패치 등이 있습니다.
   - 또한 Amazon EventBridge 또는 Amazon Simple Notification Service 등과 같은 서비스를 사용하여 추적할 사용자 지정 이벤트를 생성할 수 있습니다.
2. 인시던트
   - 인시던트에 대한 의사소통 계획을 정의하는 것으로 시작합니다. 인시던트에 대해 반드시 알아야 하는 이해 관계자는 누구인가요? 이해관계자를 루프 내에서 어떻게 유지하나요? 작업 조정은 누가 감독하나요? 의사소통 및 조정을 위한 내부 채팅 채널을 마련하는 것이 좋습니다.

- 특히, 팀에 당직 순환 근무자가 없는 경우 워크로드를 지원하는 팀에 대한 에스컬레이션 경로를 정의하세요. 지원 수준에 따라 AWS Support를 사용하여 사례를 제출할 수도 있습니다.
- 인시던트를 조사하기 위한 플레이북을 생성합니다. 플레이북에는 의사소통 계획 및 자세한 조사 단계를 포함해야 합니다. 조사에 AWS Health Dashboard 확인을 포함하세요.
- 인시던트 대응 계획을 문서화합니다. 내부 및 외부 고객이 참여 규칙과 자신에게 기대되는 행동을 이해할 수 있도록 인시던트 관리 규칙을 전달합니다. 이러한 규칙을 사용하는 방법을 팀원에게 교육합니다.
- 고객은 Incident Manager 를 사용하여 인시던트 대응 규칙을 설정 및 관리할 수 있습니다.
- Enterprise Support 고객은 기술 지원 관리자의 인시던트 관리 워크숍 을 요청할 수 있습니다. 이 안내 워크숍에서는 기존 인시던트 대응 계획을 테스트하고 개선할 수 있는 영역을 식별하도록 돕습니다.

3. 문제

- 문제는 ITSM 시스템에서 식별하고 추적해야 합니다.
- 알려진 문제를 모두 식별하고 해결에 필요한 작업 수준 및 워크로드에 미치는 영향별로 우선순위를 지정합니다.



- 미치는 영향이 크지만 노력이 적게 드는 문제부터 먼저 해결합니다. 해결되면 영향력이 낮고 노력이 적게 드는 문제로 진행합니다.
- Systems Manager OpsCenter 를 사용하여 문제를 식별하고 해당 문제에 런북을 첨부한 다음 문제를 추적할 수 있습니다.

구현 계획의 작업 수준: 보통. 모범 사례를 구현하기 위한 프로세스 및 도구가 둘 다 필요합니다. 프로세스를 문서화하고 워크로드와 관련된 모든 사람들이 사용할 수 있도록 설정합니다. 프로세스를 자주 업데이트합니다. 문제를 관리하고 문제를 완화 또는 해결하기 위한 프로세스가 있습니다.

리소스

관련 모범 사례:

- OPS07-BP03 런북을 사용한 절차 수행: 일관된 완화 작업을 위해 알려진 문제에 연결된 런북이 필요합니다.
- OPS07-BP04 플레이북을 사용하여 문제 조사: 런북을 사용하여 인시던트를 조사해야 합니다.
- OPS11-BP02 인시던트 사후 분석 수행: 인시던트에서 복구한 후에는 항상 사후 분석을 수행합니다.

관련 문서:

- Atlassian - Incident management in the age of DevOps(Atlassian - DevOps 시대에 인시던트 관리)
- AWS Security Incident Response Guide(AWS 보안 인시던트 대응 안내서)
- Incident Management in the Age of DevOps and SRE(DevOps 및 SRE 시대에 인시던트 관리)
- PagerDuty - What is Incident Management?(PagerDuty - 인시던트 관리란 무엇인가요?)

관련 동영상:

- AWS re:Invent 2020: Incident management in a distributed organization(AWS re:Invent 2020: 분산 조직에서 인시던트 관리)
- AWS re:Invent 2021 - Building next-gen applications with event-driven architectures(AWS re:Invent 2021 - 이벤트 기반 아키텍처로 차세대 애플리케이션 구축)
- AWS Supports You | Exploring the Incident Management Tabletop Exercise(인시던트 관리 살펴보기 탁상 연습)
- AWS Systems Manager Incident Manager - AWS 가상 워크숍
- AWS What's Next ft. Incident Manager | AWS 이벤트

관련 예시:

- AWS Management and Governance Tools Workshop - OpsCenter(AWS 관리 및 거버넌스 도구 워크숍 - OpsCenter)

- AWS Proactive Services – Incident Management Workshop(AWS 사전 예방 서비스 – 인시던트 관리 워크숍)
- Building an event-driven application with Amazon EventBridge(Amazon EventBridge를 사용하여 이벤트 기반 애플리케이션 구축)
- Building event-driven architectures on AWS(AWS에 이벤트 기반 아키텍처 구축)

관련 서비스:

- Amazon EventBridge
- Amazon SNS
- AWS Health Dashboard
- AWS Systems Manager Incident Manager
- AWS Systems Manager OpsCenter

OPS10-BP02 알림별 프로세스 마련

경계심을 갖는 이벤트가 있는 경우, 특정하게 식별된 소유자를 지정함과 동시에 명확하게 정의된 대응 방법(런북 또는 지침서)을 마련합니다. 이렇게 하면 운영 이벤트에 빠르고 효과적으로 대응할 수 있으며 중요하지 않은 알림 때문에 실행 가능한 이벤트를 제대로 확인하지 못하는 상황을 방지할 수 있습니다.

일반적인 안티 패턴:

- 모니터링 시스템은 승인된 연결 스트림을 다른 메시지와 함께 제공합니다. 메시지 볼륨이 너무 커서 개입이 필요한 주기적인 오류 메시지를 놓칠 수 있습니다.
- 웹 사이트가 중단되었다는 알림이 표시됩니다. 이 경우에 대해 정의된 프로세스가 없습니다. 문제를 진단하고 해결하기 위해 임시 접근 방식을 취해야 합니다. 작업을 진행하면서 이 프로세스를 개발하면 복구 시간이 길어집니다.

이 모범 사례 수립의 이점: 조치가 필요한 경우에만 알림을 보내서 중요하지 않은 알림으로 인해 중요한 알림을 놓치게 되는 상황을 막을 수 있습니다. 실행 가능한 알림을 위한 프로세스를 마련하면 환경의 이벤트에 대해 일관되고 신속한 응답이 가능합니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 알림별 프로세스: 알림 생성 대상 이벤트에 대해서는 런북이나 플레이북을 통해 대응 방법을 적절하게 정의해야 하며, 정상적인 프로세스 완료를 담당하는 소유자(예: 개인, 팀, 역할)를 구체적으로 명시해야 합니다. 대응 작업은 자동화되거나 다른 팀이 수행할 수 있지만 프로세스가 예상된 결과를 제공하는지 여부에 대한 책임은 소유자에게 있습니다. 이러한 프로세스를 마련해 두면 운영 이벤트에 빠르고 효과적으로 대응할 수 있으며 중요하지 않은 알림 때문에 실행 가능한 이벤트를 제대로 확인하지 못하는 상황을 방지할 수 있습니다. 예를 들어 웹 프런트 엔드의 크기를 조정하려면 자동 조정 기능을 적용할 수 있지만, 운영 팀은 자동 조정 규칙 및 한도가 워크로드 요구 사항에 적합한지 확인해야 합니다.

리소스

관련 문서:

- Amazon CloudWatch 기능
- Amazon CloudWatch Events란 무엇입니까?

관련 동영상:

- 모니터링 플랜 세우기

OPS10-BP03 비즈니스 영향을 기반으로 운영 이벤트의 우선순위 지정

여러 이벤트에 대해 조치를 취해야 할 때는 실무에 가장 큰 영향을 주는 이벤트를 먼저 해결해야 합니다. 이러한 영향에는 사망 또는 부상, 재정적 손실, 평판 또는 신뢰의 손상이 포함될 수 있습니다.

일반적인 안티 패턴:

- 사용자의 프린터 구성 추가를 위한 지원 요청을 받습니다. 이 문제를 해결하는 동안 소매 사이트가 다운되었다는 지원 요청을 받습니다. 사용자의 프린터 구성을 완료한 후 웹 사이트 문제 해결에 착수합니다.
- 소매 웹 사이트와 급여 시스템이 모두 다운되었다는 알림을 받습니다. 어느 것에 우선순위를 두어야 하는지 알 수 없습니다.

이 모범 사례 수립의 이점: 비즈니스에 가장 큰 영향을 미치는 인시던트에 대한 대응의 우선순위를 정하면 이러한 영향을 관리할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 비즈니스 영향에 기반하여 운영 이벤트 우선순위 지정: 여러 이벤트에 대해 조치를 취해야 할 때는 실무에 가장 큰 영향을 주는 이벤트를 먼저 해결해야 합니다. 이러한 영향에는 사망 또는 부상, 재정적 손실, 규정 위반 또는 평판이나 신뢰의 손상이 포함될 수 있습니다.

OPS10-BP04 에스컬레이션 경로 정의

에스컬레이션을 트리거하는 요소와 에스컬레이션 절차를 포함한 에스컬레이션 경로를 런북과 플레이북에 정의합니다. 운영 이벤트에 즉시 효율적으로 대응할 수 있도록 각 작업의 소유자를 구체적으로 명시합니다.

작업을 수행하기 전에 사람의 결정이 필요한 경우를 확인합니다. 의사 결정권자와 협력하여 미리 의사 결정을 내리고 작업을 사전에 승인합니다. 그러면 답변을 기다리기 위한 MTTR이 연장되지 않습니다.

일반적인 안티 패턴:

- 소매 사이트가 다운되었습니다. 사이트 복구를 위한 런북을 봐도 모르겠습니다. 도움을 구하기 위해 동료에게 전화를 걸기 시작합니다.
- 연결할 수 없는 애플리케이션에 대한 지원 사례를 받습니다. 시스템을 관리할 권한이 없습니다. 누구에게 권한이 있는지 알 수 없습니다. 사례를 개시한 시스템 소유자에게 연락을 시도하는데 응답이 없습니다. 시스템에 대한 연락처가 없으며 동료가 시스템에 익숙하지 않습니다.

이 모범 사례 수립의 이점: 에스컬레이션, 에스컬레이션 트리거 및 에스컬레이션 절차를 정의하면 영향에 대해 적절한 속도로 인시던트에 리소스를 체계적으로 추가할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 에스컬레이션 경로 정의: 에스컬레이션을 트리거하는 요소와 에스컬레이션 절차를 포함한 에스컬레이션 경로를 런북과 플레이북에 정의합니다. 예를 들어 런북을 통해 문제를 해결할 수 없거나 미리 정의된 시간이 지난 경우에는 지원 엔지니어가 수석 지원 엔지니어에게 문제를 에스컬레이션하도록 정의합니다. 플레이북을 통해 문제 해결 경로를 확인할 수 없거나 미리 정의된 시간이 지난 경우에는 수석 지원 엔지니어가 개발 팀에게 문제를 에스컬레이션할 수도 있습니다. 운영 이벤트에 즉시 효율적으로 대응할 수 있도록 각 작업의 소유자를 구체적으로 명시합니다. 에스컬레이션 과정에는 제3자

가 포함될 수 있습니다. 제3자의 예로는 네트워크 연결 공급자, 소프트웨어 공급업체 등이 있습니다. 영향을 받는 시스템에 대해 권한이 부여된 의사 결정자가 에스컬레이션 과정에 참여할 수 있습니다.

OPS10-BP05 중단에 대한 고객 커뮤니케이션 계획 정의

중단 중에 고객과 이해 관계자에게 지속적으로 정보를 제공하는 데 사용할 수 있는 시스템 중단에 대한 커뮤니케이션 계획을 정의하고 테스트합니다. 사용자가 이용하는 서비스가 영향을 받거나 서비스가 정상으로 돌아올 때 모두 사용자와 직접 커뮤니케이션합니다.

원하는 결과:

- 예약된 유지 보수부터 재해 복구 계획 호출을 비롯한 예기치 않은 대규모 장애에 이르는 다양한 상황에 대한 커뮤니케이션 계획이 있습니다.
- 커뮤니케이션에서 시스템 문제에 대한 정보를 명확하고 투명하게 제공하여 고객이 시스템 성능을 추측하지 않도록 돕습니다.
- 사용자 지정 오류 메시지 및 상태 페이지를 사용하여 급격하게 늘어나는 헬프데스크 요청을 줄이고 사용자에게 정보를 제공합니다.
- 커뮤니케이션 계획은 실제 중단이 발생할 때 의도한 대로 수행되는지 확인하기 위해 정기적으로 테스트됩니다.

일반적인 안티 패턴:

- 워크로드 중단이 발생했지만 커뮤니케이션 계획이 없습니다. 사용자는 중단에 대한 정보가 없기 때문에 문제 티켓 시스템에 요청이 폭주합니다.
- 중단 중에 사용자에게 이메일 알림을 보냅니다. 여기에는 서비스 복원 일정이 포함되어 있지 않으므로 사용자는 중단과 관련된 계획을 세울 수 없습니다.
- 중단에 대한 커뮤니케이션 계획이 있지만 테스트를 진행한 적이 없습니다. 중단이 발생하고 테스트 진행 중에 발견할 수 있는 중요한 단계를 놓쳤기 때문에 커뮤니케이션 계획이 실패합니다.
- 중단 중에 AWS NDA에 해당하는 기술 세부 사항 및 정보가 너무 많이 담긴 알림을 사용자에게 보냅니다.

이 모범 사례 확립의 이점:

- 중단 중에 커뮤니케이션을 계속 주고받으면 고객이 문제 진행 상황과 예상 해결 시간을 확인할 수 있습니다.

- 잘 정의된 커뮤니케이션 계획을 개발하면 고객과 최종 사용자가 중단의 영향을 완화하기 위해 필요한 추가 단계를 수행할 수 있도록 충분한 정보를 얻었는지 확인할 수 있습니다.
- 적절한 커뮤니케이션과 계획된 중단 및 계획되지 않은 가동 중단에 대한 인식 개선을 통해 고객 만족도를 높이고 의도하지 않은 반응을 제한하며 고객 유지를 촉진할 수 있습니다.
- 투명하고 시의적절한 시스템 중단 커뮤니케이션은 확신을 주고 고객과의 관계를 유지하는 데 필요한 신뢰를 형성합니다.
- 중단 또는 위기 동안 입증된 커뮤니케이션 전략은 복구 능력을 방해할 수 있는 추측과 소문을 줄입니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

중단 중에 고객에게 계속 정보를 제공하는 커뮤니케이션 계획은 종합적이며, 고객에게 표시되는 오류 페이지, 사용자 지정 API 오류 메시지, 시스템 상태 배너 및 상태 페이지를 비롯한 여러 인터페이스를 포함합니다. 시스템에 등록된 사용자가 포함된 경우 이메일, SMS 또는 푸시 알림과 같은 메시징 채널을 통해 통신하여 개인화된 메시지 콘텐츠를 고객에게 보낼 수 있습니다.

고객 커뮤니케이션 도구

1차 방어선으로 웹 및 모바일 애플리케이션은 중단 시 친근하고 유용한 정보가 담긴 오류 메시지를 제공하고, 트래픽을 상태 페이지로 리디렉션할 수 있어야 합니다. Amazon CloudFront는 사용자 지정 오류 콘텐츠를 정의하고 제공하는 기능이 포함된 완전관리형 콘텐츠 전송 네트워크(CDN)입니다. CloudFront의 사용자 지정 오류 페이지는 구성 요소 수준의 중단에 활용하기에 좋은 첫 번째 고객 메시징 계층입니다. 또한 CloudFront는 계획되거나 계획되지 않은 중단 중에 모든 요청을 가로채는 상태 페이지의 활성화와 관리를 간소화할 수 있습니다.

사용자 지정 API 오류 메시지는 중단이 개별 서비스로 분리될 때 영향을 감지하고 줄이는 데 도움이 될 수 있습니다.Amazon API Gateway를 사용하면 REST API에 대한 사용자 지정 응답을 구성할 수 있습니다. 이를 통해 API Gateway가 백엔드 서비스에 연결할 수 없을 때 API 소비자에게 명확하고 의미 있는 메시지를 제공할 수 있습니다. 사용자 지정 메시지는 서비스 계층 중단으로 인해 특정 시스템 기능이 저하될 때 중단 배너 콘텐츠 및 알림을 지원하는 데 사용할 수도 있습니다.

다이렉트 메시징은 가장 개인화된 고객 메시징 유형입니다. Amazon Pinpoint는 확장 가능한 다중 채널 커뮤니케이션을 위한 관리형 서비스입니다. Amazon Pinpoint를 사용하면 SMS, 이메일, 음성, 푸시 알림 또는 정의한 사용자 지정 채널을 통해 영향을 받는 고객층 전체에 광범위하게 메시지를 브로드캐스트할 수 있는 캠페인을 구축할 수 있습니다. Amazon Pinpoint로 메시징을 관리하면 메시지 캠페인이

잘 정의되고 테스트 가능하며 대상 고객 세그먼트에 지능적으로 적용될 수 있습니다. 캠페인이 설정되면 이벤트로 예약하거나 트리거할 수 있으며 쉽게 테스트할 수 있습니다.

고객 사례

워크로드가 손상되면 AnyCompany Retail은 사용자에게 이메일 알림을 보냅니다. 이메일은 어떤 비즈니스 기능이 손상되었는지 설명하고 서비스가 복원될 시기에 대한 현실적인 추정치를 제공합니다. 또한 워크로드 상태에 대한 실시간 정보를 보여주는 상태 페이지가 있습니다. 커뮤니케이션 계획은 효과적인지 확인하기 위해 1년에 두 번 개발 환경에서 테스트됩니다.

구현 단계

1. 메시징 전략을 시행할 커뮤니케이션 채널을 결정합니다. 애플리케이션의 아키텍처 측면을 고려하고 고객에게 피드백을 제공하기 위한 최상의 전략을 결정합니다. 여기에는 오류 및 상태 페이지, 사용자 지정 API 오류 응답 또는 다이렉트 메시징을 포함하여 설명된 하나 이상의 지침 전략이 포함될 수 있습니다.
2. 애플리케이션의 상태 페이지를 설계합니다. 상태 또는 사용자 정의 오류 페이지가 고객에게 적합하다고 판단되면 해당 페이지에 대한 콘텐츠 및 메시지를 설계해야 합니다. 오류 페이지에서는 사용자에게 애플리케이션을 사용할 수 없는 이유, 다시 사용할 수 있는 시기, 그 동안 할 수 있는 작업을 설명합니다. 애플리케이션에서 Amazon CloudFront를 사용하는 경우 사용자 지정 오류 응답을 제공하거나 엣지에서 Lambda를 사용하여 오류를 변환하고 페이지 콘텐츠를 다시 작성할 수 있습니다. 또한 CloudFront를 사용하면 애플리케이션 콘텐츠에서 유지 보수 또는 중단 상태 페이지가 포함된 정적 Amazon S3 콘텐츠 오리진으로 대상을 바꿀 수 있습니다.
3. 서비스에 대한 올바른 API 오류 상태 집합을 설계합니다. 백엔드 서비스에 도달할 수 없을 때 API Gateway에서 생성되는 오류 메시지와 서비스 계층 예외에는 최종 사용자에게 표시하기에 적합한 친숙한 메시지가 포함되어 있지 않을 수 있습니다. 백엔드 서비스의 코드를 변경하지 않고도 HTTP 응답 코드를 선별된 API 오류 메시지에 매핑하도록 API Gateway 사용자 지정 오류 응답을 구성할 수 있습니다.
4. 시스템의 최종 사용자와 관련이 있고 기술적 세부 사항을 포함하지 않도록 비즈니스 관점에서 메시징을 디자인합니다. 대상을 고려하고 메시지를 조정합니다. 예를 들어 내부 사용자에게 대체 시스템을 활용하는 해결 방법이나 수동 프로세스로 안내할 수 있습니다. 외부 사용자는 시스템이 복원될 때까지 기다리거나 업데이트를 구독하여 시스템이 복원된 후 알림을 받도록 요청받을 수 있습니다. 예기치 않은 중단, 계획된 유지 보수, 특정 기능이 저하되거나 사용할 수 없는 부분적인 시스템 오류를 비롯한 여러 시나리오에 대해 승인된 메시징을 정의합니다.
5. 고객 메시징을 템플릿화하고 자동화합니다. 메시지 콘텐츠를 설정한 후에는 Amazon Pinpoint 또는 기타 도구를 사용하여 메시징 캠페인을 자동화할 수 있습니다. Amazon Pinpoint를 사용하면 영향을

받는 특정 사용자에 대한 고객 대상 세그먼트를 생성하고 메시지를 템플릿으로 변환할 수 있습니다. 메시징 캠페인 설정 방법을 이해하려면 Amazon Pinpoint 자습서를 검토합니다.

6. 메시징 기능을 고객용 시스템에 밀결합하지 않도록 합니다. 중단이 발생할 때 성공적으로 메시지를 보낼 수 있는지 확인하기 위해 메시징 전략이 시스템 데이터 스토어 또는 서비스에 크게 의존해서는 안 됩니다. 메시징 가용성을 위해 둘 이상의 가용 영역 또는 리전에서 메시지를 보내는 기능을 구축하는 것을 고려합니다. AWS 서비스를 사용하여 메시지를 보내는 경우 컨트롤 플레인 작업보다 데이터 영역 작업을 활용하여 메시징을 호출합니다.

구현 계획의 작업 수준: 높음. 커뮤니케이션 계획과 이를 전송하는 메커니즘을 개발하려면 상당한 노력이 필요할 수 있습니다.

리소스

관련 모범 사례:

- OPS07-BP03 런북을 사용한 절차 수행 - 커뮤니케이션 계획에는 담당자가 대응 방법을 알 수 있도록 이와 관련된 런북이 있어야 합니다.
- OPS11-BP02 인시던트 사후 분석 수행 - 중단 후에는 사고 후 분석을 수행하여 또 다른 중단을 방지하기 위한 메커니즘을 식별합니다.

관련 문서:

- Error Handling Patterns in Amazon API Gateway and AWS Lambda(Amazon API Gateway 및 AWS Lambda의 오류 처리 패턴)
- Amazon API Gateway responses(Amazon API Gateway 응답)

관련 예시:

- AWS Health 대시보드
- Summary of the AWS Service Event in the Northern Virginia (US-EAST-1) Region(버지니아 북부(US-EAST-1) 리전의 AWS 서비스 이벤트 요약)

관련 서비스:

- AWS Support
- AWS 이용계약

- Amazon CloudFront
- Amazon API Gateway
- Amazon Pinpoint
- Amazon S3

OPS10-BP06 대시보드를 통해 상태 전달

목표 대상(예: 내부 기술 팀, 리더십 및 고객)에게 맞춘 대시보드를 제공하여 비즈니스의 현재 운영 상태를 알리고 관심 있는 지표를 제공합니다.

대시보드는 Amazon CloudWatch 대시보드 를 사용하여 CloudWatch 콘솔을 통해 사용자 지정 가능한 홈 페이지에서 생성할 수 있습니다. 그리고 Amazon QuickSight 와 같은 비즈니스 인텔리전스 서비스를 사용하면 워크로드 및 운영 상태(예: 주문율, 연결된 사용자 수 및 트랜잭션 시간)에 대한 대화형 대시보드를 생성하고 게시할 수 있습니다. 그리고 이러한 지표의 시스템 및 비즈니스 수준의 보기를 제공하는 대시보드를 생성합니다.

일반적인 안티 패턴:

- 요청 시 관리를 위해 애플리케이션의 현재 사용률에 대한 보고서를 실행합니다.
- 인시던트 발생 시 해결 여부를 확인하고자 관련 시스템 소유자가 20분마다 연락합니다.

이 모범 사례 수립의 이점: 대시보드를 생성하면 정보에 직접 액세스할 권한을 활성화하여 고객이 스스로 정보를 파악하고 조치를 취해야 하는지 판단할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 대시보드를 통해 상태 전달: 목표 대상(예: 내부 기술 팀, 리더십 및 고객)에 맞춤화된 대시보드를 제공하여 비즈니스의 현재 운영 상태를 전달하고 관심 있는 지표를 안내합니다. 상태 정보 확인을 위한 셀프 서비스 옵션을 제공하면 운영 팀의 필딩 요청이 중단되는 상황을 줄일 수 있습니다. 예로는 Amazon CloudWatch 대시보드와 AWS Health Dashboard가 있습니다.
  - 사용자 지정 지표 보기를 생성 및 사용하는 CloudWatch 대시보드

리소스

관련 문서:

- Amazon QuickSight
- 사용자 지정 지표 보기를 생성 및 사용하는 CloudWatch 대시보드

OPS10-BP07 이벤트 대응 자동화

이벤트 대응을 자동화하면 수동 프로세스에서 발생하는 오류를 줄일 수 있으며 일관된 방식으로 즉시 대응할 수 있습니다.

AWS에서 여러 방법으로 런북 및 플레이북 작업을 자동화할 수 있습니다. AWS 리소스의 상태 변경으로 인해 발생하는 이벤트나 사용자 지정 이벤트에 응답하려는 경우 CloudWatch Events 규칙 을 생성하여 CloudWatch 대상(예: Lambda 함수, Amazon Simple Notification Service(Amazon SNS) 주제, Amazon ECS 작업, AWS Systems Manager Automation)을 통해 응답을 트리거해야 합니다.

리소스의 임계값(예: 대기 시간)을 초과하는 지표에 응답하려는 경우에는 CloudWatch 경보 를 생성하여 Amazon EC2 작업, Auto Scaling 작업을 통해 하나 이상의 작업을 수행하거나 Amazon SNS 주제에 알림을 보내면 됩니다. 경보에 대응하여 사용자 지정 작업을 수행해야 하는 경우 Amazon SNS 알림을 통해 Lambda를 호출합니다. 직원들이 정보를 계속 확인할 수 있도록 Amazon SNS를 사용하여 이벤트 알림 및 에스컬레이션 메시지를 게시합니다.

AWS는 AWS 서비스 API 및 SDK를 통해 서드 파티 시스템도 지원합니다. AWS 파트너 및 서드 파티에서 제공하는 다양한 모니터링 도구를 모니터링, 알림 및 응답에 사용할 수 있습니다. 이러한 도구의 예로는 New Relic, Splunk, Loggly, SumoLogic, Datadog 등이 있습니다.

자동 절차에서 오류가 발생하는 경우를 대비해서 중요 수동 절차를 사용 가능한 상태로 유지해야 합니다.

일반적인 안티 패턴:

- 개발자가 코드를 확인합니다. 빌드를 시작한 다음 테스트를 수행하는 데 이 이벤트가 사용되었을 수 있지만 아무 일도 일어나지 않습니다.
- 애플리케이션이 작업을 중지하기 전에 특정 오류를 기록합니다. 애플리케이션을 다시 시작하는 절차는 잘 이해하고 스크립팅할 수 있습니다. 로그 이벤트를 사용하여 스크립트를 호출하고 애플리케이션을 다시 시작할 수 있습니다. 대신 일요일 오전 3시에 오류가 발생하여 시스템 수정을 담당하는 당직 직원으로서 호출을 받습니다.

이 모범 사례 수립의 이점: 이벤트에 대한 자동 응답을 사용하여 응답 시간을 줄이고 수동 활동으로 인한 오류 발생을 제한할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 이벤트에 대한 응답 자동화: 이벤트 응답을 자동화하면 수동 프로세스에서 발생하는 오류를 줄일 수 있으며 일관된 방식으로 즉시 대응할 수 있습니다.
  - Amazon CloudWatch Events란 무엇입니까?
  - 이벤트에서 트리거되는 CloudWatch Events 규칙 생성
  - AWS CloudTrail를 사용하여 AWS API 호출에서 트리거되는 CloudWatch Events 규칙 생성
  - 지원되는 서비스의 CloudWatch Events 이벤트 예제

리소스

관련 문서:

- Amazon CloudWatch 기능
- 지원되는 서비스의 CloudWatch Events 이벤트 예제
- AWS CloudTrail를 사용하여 AWS API 호출에서 트리거되는 CloudWatch Events 규칙 생성
- 이벤트에서 트리거되는 CloudWatch Events 규칙 생성
- Amazon CloudWatch Events란 무엇입니까?

관련 동영상:

- 모니터링 플랜 세우기

관련 예시:

개선

질문

- OPS 11 귀사는 어떻게 운영을 지속적으로 개선하고 있습니까?

OPS 11 귀사는 어떻게 운영을 지속적으로 개선하고 있습니까?

시간과 리소스를 할애하여 점진적 개선을 지속적으로 수행하면 운영 효율성을 높일 수 있습니다.

모범 사례

- OPS11-BP01 지속적인 개선을 위한 프로세스 마련
- OPS11-BP02 인시던트 사후 분석 수행
- OPS11-BP03 피드백 루프 구현
- OPS11-BP04 지식 관리 수행
- OPS11-BP05 개선 추진 요인 정의
- OPS11-BP06 인사이트 검증
- OPS11-BP07 운영 지표 검토 수행
- OPS11-BP08 파악한 내용 문서화 및 공유
- OPS11-BP09 개선을 위한 시간 할애

OPS11-BP01 지속적인 개선을 위한 프로세스 마련

내부 및 외부 아키텍처 모범 사례를 기준으로 워크로드를 평가합니다. 매년 1회 이상 워크로드 검토를 실시합니다. 소프트웨어 개발 단계에서 개선 기회의 우선순위를 지정합니다.

원하는 결과:

- 매년 1회 이상 아키텍처 모범 사례를 기준으로 워크로드를 분석합니다.
- 소프트웨어 개발 프로세스에서 개선 기회에 동일한 우선순위가 부여됩니다.

일반적인 안티 패턴:

- 몇 년 전에 배포된 이후 워크로드에 대한 아키텍처 검토를 수행한 적이 없습니다.
- 개선 기회의 우선순위가 낮게 지정되어 계속 밀려 있는 상태입니다.
- 조직에 맞춰 모범 사례를 수정하기 위한 표준이 없습니다.

이 모범 사례 확립의 이점:

- 워크로드가 최신 아키텍처 모범 사례에 맞춰 유지됩니다.
- 워크로드 발전은 신중한 방식으로 이루어집니다.
- 조직의 모범 사례를 활용하여 모든 워크로드를 개선할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

매년 1회 이상 워크로드의 아키텍처 검토를 수행합니다. 내부 및 외부 모범 사례를 사용하여 워크로드를 평가하고 개선 기회를 식별합니다. 소프트웨어 개발 단계에서 개선 기회의 우선순위를 지정합니다.

고객 사례

AnyCompany Retail의 모든 워크로드는 연 1회 아키텍처 검토 프로세스를 거칩니다. 모든 워크로드에 적용되는 자체 모범 사례 체크리스트를 개발했습니다. AWS Well-Architected Tool의 Custom Lens 기능을 사용합니다. 도구 및 모범 사례 Custom Lens를 사용하여 검토를 수행합니다. 검토를 통해 생성된 개선 기회에는 소프트웨어 스프린트에서 우선순위가 지정됩니다.

구현 단계

1. 매년 1회 이상 프로덕션 워크로드의 주기적 아키텍처 검토를 수행합니다. AWS 관련 모범 사례를 포함한 문서화된 아키텍처 표준을 사용합니다.
   a. 이러한 검토에는 내부에서 정의된 표준을 사용하는 것이 좋습니다. 내부 표준이 없는 경우에는 AWS Well-Architected Framework를 사용하는 것이 좋습니다.
   b. AWS Well-Architected Tool을 사용하여 내부 모범 사례의 Custom Lens를 생성하고 아키텍처 검토를 수행할 수 있습니다.
   c. 고객은 담당 AWS Solutions Architect에 연락하여 안내에 따라 워크로드의 Well-Architected Framework 검토를 수행할 수 있습니다.
2. 소프트웨어 개발 프로세스 중 검토 과정에서 식별된 개선 기회의 우선순위를 지정합니다.

구현 계획의 작업 수준: 낮음. AWS Well-Architected Framework를 사용하여 연간 아키텍처 검토를 수행할 수 있습니다.

리소스

관련 모범 사례:

- OPS11-BP02 인시던트 사후 분석 수행 - 인시던트 후 분석은 개선할 부분을 찾을 수 있는 또 다른 기회입니다. 얻은 교훈을 내부 아키텍처 모범 사례 목록에 추가하세요.
- OPS11-BP08 파악한 내용 문서화 및 공유 - 자체 아키텍처 모범 사례를 개발하면 조직 간에 공유하세요.

관련 문서:

- AWS Well-Architected Tool - Custom Lens
- AWS Well-Architected 백서 - 검토 프로세스
- Custom Lens 및 AWS Well-Architected Tool을 사용하여 Well-Architected 검토 사용자 지정
- 조직에서 AWS Well-Architected Custom Lens 수명 주기 구현

관련 동영상:

- Well-Architected Labs - Level 100: Custom Lenses on AWS Well-Architected Tool(Well-Architected 실습 - 레벨 100: AWS Well-Architected Tool의 Custom Lens)

관련 예시:

- AWS Well-Architected Tool

OPS11-BP02 인시던트 사후 분석 수행

고객에게 영향을 주는 이벤트를 검토하고 기여 요인과 예방 조치를 식별합니다. 이 정보를 사용하여 재발을 제한하거나 방지하는 완화 기능을 개발합니다. 신속하고 효과적인 대응을 위한 절차를 개발합니다. 목표 대상에 맞게 적절히 발생 요인과 수정 조치를 전달합니다.

일반적인 안티 패턴:

- 애플리케이션 서버를 관리합니다. 약 23시간 55분마다 모든 활성 세션이 종료됩니다. 애플리케이션 서버에서 무엇이 잘못되었는지 파악하려고 했습니다. 네트워크 문제일 수도 있다고 생각하지만 네트워크 팀이 너무 바쁜 관계로 지원을 받을 수 없습니다. 지원을 받고 진행 상황을 파악하는 데 필요한 정보를 수집하기 위해 따라야 할 사전 정의된 프로세스가 없습니다.
- 워크로드 내에서 데이터가 손실되었습니다. 이런 일은 처음이며 그 원인이 명확하지 않습니다. 데이터를 다시 생성할 수 있으므로 대수롭지 않은 일로 생각합니다. 데이터 손실이 발생하면서 고객에게 영향을 미치는 빈도가 증가합니다. 또한 이로 인해 누락된 데이터를 복원할 때 운영 부담이 가중됩니다.

이 모범 사례 정립의 이점: 인시던트에 기여한 구성 요소, 조건, 작업 및 이벤트를 결정하기 위해 사전 정의된 프로세스를 사용하면 개선 기회를 파악할 수 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 프로세스를 사용하여 기여 요인 확인: 고객에게 영향을 미치는 모든 인시던트를 검토합니다. 재발을 제한하거나 방지하기 위한 완화책을 개발하고 빠르고 효과적인 대응을 위한 절차를 개발할 수 있도록 인시던트의 기여 요인을 식별하고 문서화하는 프로세스를 마련합니다. 적절한 경우 근본 원인을 알리고 목표 대상에게 맞춤화된 프로세스를 마련합니다.

OPS11-BP03 피드백 루프 구현

피드백 루프는 의사 결정을 추진하는 실행 가능한 인사이트를 제공합니다. 절차와 워크로드에 피드백 루프를 구축하세요. 이를 통해 문제와 개선이 필요한 영역을 파악할 수 있습니다. 또한 개선에 대한 투자를 검증합니다. 이러한 피드백 루프는 워크로드를 지속적으로 향상하기 위한 기반입니다.

피드백 루프의 두 가지 카테고리: 즉각적 피드백 및 후행 분석. 즉각적 피드백은 운영 활동의 성과 및 결과를 검토하여 수집합니다. 이 피드백은 팀원, 고객 또는 자동화된 활동 출력으로부터 제공됩니다. A/B 테스트 및 새로운 기능 전달과 같은 사항을 통해 즉각적 피드백을 수신하며, 빠른 실패에 필수입니다.

시간 경과에 따른 운영 결과 및 지표 검토 결과의 피드백을 얻을 수 있도록 후행 분석이 정기적으로 수행됩니다. 이러한 후행 분석은 스프린트 후반, 정기적인 주기, 또는 주요 릴리스나 이벤트 이후 수행합니다. 이러한 유형의 피드백 루프는 운영 또는 워크로드의 투자를 검증합니다. 이를 통해 성공 여부를 측정하고 결과를 검증할 수 있습니다.

원하는 결과: 즉각적 피드백 및 후행 분석을 사용하여 개선을 추진할 수 있습니다. 사용자 및 팀원의 피드백을 얻을 수 있는 메커니즘이 있습니다. 후행 분석은 개선을 추진하는 트렌드를 파악하는 데 사용됩니다.

일반적인 안티 패턴:

- 새로운 기능을 출시했지만 이에 대한 고객 피드백을 받을 수 있는 방법이 없습니다.
- 운영 개선에 투자한 후 이를 검증할만한 후행 분석을 수행하지 않습니다.
- 고객 피드백을 수집하지만 이를 정기적으로 검토하지 않습니다.
- 피드백 루프를 통해 제안된 조치 항목을 얻지만 소프트웨어 개발 프로세스에 포함되지 않습니다.
- 고객이 제안한 개선 사항에 대한 피드백을 받지 못합니다.

이 모범 사례 확립의 이점:

- 고객의 입장에서 시작한 역방향 작업을 통해 새로운 기능을 이끌어낼 수 있습니다.
- 조직 문화가 변화에 빠르게 반응할 수 있습니다.

- 트렌드를 사용하여 개선 기회를 파악할 수 있습니다.
- 후행 분석을 통해 워크로드 및 운영에 대한 투자를 검증할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

이 모범 사례를 구현하면 즉각적인 피드백과 후행 분석을 모두 사용하게 됩니다. 이러한 피드백 루프를 통해 개선을 추진할 수 있습니다. 설문 조사, 고객 투표, 피드백 양식 등 즉각적 피드백을 위한 다양한 메커니즘이 있습니다. 조직에서는 후행 분석도 사용하여 개선 기회를 파악하고 이니셔티브를 검증합니다.

고객 사례

AnyCompany Retail은 고객이 피드백을 제공하고 문제를 보고할 수 있는 웹 양식을 만들었습니다. 주간 스크럼 기간 동안 소프트웨어 개발 팀이 사용자 피드백을 평가합니다. 피드백은 플랫폼의 평가를 추진하는 데 정기적으로 사용됩니다. 각 스프린트의 후반에는 후행 분석을 수행하여 개선하고자 하는 항목을 파악합니다.

구현 단계

1. 즉각적 피드백
   - 고객 및 팀원으로부터 피드백을 수신할 수 있는 메커니즘이 필요합니다. 또한 자동 피드백을 제공하도록 운영 활동을 구성할 수 있습니다.
   - 조직은 이 피드백을 검토하고, 개선할 점을 결정하며 개선 일정을 지정하는 프로세스가 필요합니다.
   - 피드백이 소프트웨어 개발 프로세스에 반드시 추가되어야 합니다.
   - 개선 사항이 있을 때 피드백 제출자에게 후속 조치를 취합니다.
     - 이때 AWS Systems Manager OpsCenter 를 사용하여 이러한 개선 사항을 OpsItems로 생성하고 추적할 수 있습니다.
2. 후행 분석
   - 개발 주기의 마지막, 정기적인 주기, 또는 주요 릴리스 이후에 후행 분석을 수행합니다.
   - 후행 분석 회의를 위해 워크로드에 관련된 이해 관계자를 모읍니다.
   - 화이트보드나 스프레드시트에 중지, 시작 및 유지라는 세 개의 열을 만듭니다.
     - 중지 는 팀에서 수행을 중지하고자 하는 항목입니다.
     - 시작 은 시작하고자 하는 아이디어입니다.

    - 유지 는 계속 하고자 하는 항목입니다.
- 회의실을 한 바퀴 돌며 이해 관계자들의 피드백을 수렴합니다.
- 피드백의 우선순위를 정합니다. 모든 시작 또는 유지 항목에 대한 활동 및 이해 관계자를 할당합니다.
- 소프트웨어 개발 프로세스에 해당 활동을 추가하고 개선 작업을 수행할 때 이해 관계자에게 상태 업데이트를 전달합니다.

구현 계획의 작업 수준: 보통. 이 모범 사례를 구현하려면 즉각적인 피드백을 수렴하고 이를 분석할 수 있는 방법이 필요합니다. 또한 후행 분석 프로세스를 확립해야 합니다.

리소스

관련 모범 사례:

- OPS01-BP01 외부 고객 요구 평가: 피드백 루프는 외부 고객의 요구를 수집할 수 있는 메커니즘입니다.
- OPS01-BP02 내부 고객 요구 평가: 내부 이해 관계자는 피드백 루프를 사용하여 필요 및 요구 사항을 논의합니다.
- OPS11-BP02 인시던트 사후 분석 수행: 인시던트 사후 분석은 인시던트 후 수행하는 후행 분석의 중요한 양식입니다.
- OPS11-BP07 운영 지표 검토 수행: 운영 지표 검토는 개선을 위한 트렌드와 영역을 파악합니다.

관련 문서:

- CCOE 구축 시 피해야 할 7가지 위험
- Atlassian Team 플레이북 - 후행 분석
- 이메일 정의: 피드백 루프
- AWS Well-Architected 프레임워크 검토를 기반으로 피드백 루프 확립
- IBM Garage 방법론 - 후행 분석 진행
- Investopedia – PDCS 주기
- Tim Cochran의 개발자 효율성 극대화
- ORR(운영 준비 상태 검토) 백서 - 반복
- TIL CSI - 지속적인 서비스 개선
- Toyota가 전자 상거래를 만났을 때: Amazon으로부터 배우기

관련 동영상:

- 효과적인 고객 피드백 루프 구축

관련 예시:

- Astuto - 오픈 소스 고객 피드백 도구
- AWS 솔루션 - AWS의 QnABot
- Fider - 고객 피드백 구성을 위한 플랫폼

관련 서비스:

- AWS Systems Manager OpsCenter

OPS11-BP04 지식 관리 수행

지식 관리는 팀원들이 업무 수행에 필요한 정보를 찾는 데 도움이 됩니다. 학습하는 조직에서는 개인에게 유용한 정보가 자유롭게 공유됩니다. 정보를 발견하거나 검색할 수 있습니다. 정확한 최신 정보입니다. 새로운 정보를 생성하고, 기존 정보를 업데이트하고, 오래된 정보를 보관하는 메커니즘이 있습니다. 지식 관리 플랫폼의 가장 일반적인 예로는 Wiki와 같은 콘텐츠 관리 시스템을 들 수 있습니다.

원하는 결과:

- 팀원이 적시에 정확한 정보에 액세스할 수 있습니다.
- 정보 검색이 가능합니다.
- 정보를 추가, 업데이트 및 보관하는 메커니즘이 있습니다.

일반적인 안티 패턴:

- 중앙 집중식 지식 스토리지가 없습니다. 팀 구성원은 로컬 컴퓨터에서 자신의 메모를 관리합니다.
- 셀프 호스팅된 Wiki가 있지만 정보를 관리하는 메커니즘이 없어 정보가 최신 상태가 아닙니다.
- 누군가 누락된 정보를 식별하지만 팀 Wiki에 추가하도록 요청할 프로세스가 없습니다. 이를 직접 추가하지만 중요한 단계를 놓쳐 중단으로 이어집니다.

이 모범 사례 확립의 이점:

- 정보가 자유롭게 공유되기 때문에 팀원의 역량이 강화됩니다.

- 문서가 최신 상태이고 검색 가능하기 때문에 새로운 팀 구성원이 더 빨리 온보딩됩니다.
- 정보는 시의적절하고 정확하며 실행 가능합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

지식 관리는 학습하는 조직의 중요한 측면입니다. 시작하려면 지식을 저장할 중앙 리포지토리가 필요합니다(일반적인 예: 셀프 호스팅된 Wiki). 지식을 추가, 업데이트 및 보관하는 프로세스를 마련해야 합니다. 문서화해야 하는 항목에 대한 표준을 개발하고 모든 사람이 기여하도록 합니다.

고객 사례

AnyCompany Retail은 모든 지식이 저장되는 내부 Wiki를 호스팅합니다. 팀원은 일상 업무를 수행하면서 지식 베이스에 추가하도록 권장됩니다 다기능 팀은 분기별로 가장 적게 업데이트된 페이지를 평가하고 아카이브할지 또는 업데이트할지 결정합니다.

구현 단계

1. 먼저 지식이 저장될 콘텐츠 관리 시스템을 식별합니다. 조직 전체의 이해 관계자로부터 동의를 얻습니다.
   a. 기존 콘텐츠 관리 시스템이 없는 경우 셀프 호스팅된 Wiki를 실행하거나 버전 관리 리포지토리에서 시작하는 것이 좋습니다.
2. 정보를 추가, 업데이트 및 보관하기 위한 런북을 개발합니다. 팀에 이러한 프로세스를 알려줍니다.
3. 콘텐츠 관리 시스템에 어떤 지식을 저장해야 하는지 식별합니다. 팀원이 수행하는 일상 업무(런북 및 플레이북)부터 시작합니다. 이해 관계자와 협력하여 추가되는 지식의 우선순위를 정합니다.
4. 주기적으로 이해 관계자와 협력하여 오래된 정보를 식별하여 아카이브하거나 최신 정보를 가져옵니다.

구현 계획의 작업 수준: 중간. 기존 콘텐츠 관리 시스템이 없는 경우 셀프 호스팅된 Wiki 또는 버전 관리 문서 리포지토리를 설정할 수 있습니다.

리소스

관련 모범 사례:

- OPS11-BP08 파악한 내용 문서화 및 공유 - 지식 관리는 학습한 내용에 대한 정보 공유를 용이하게 합니다.

관련 문서:

- Atlassian - 지식 관리

관련 예시:

- DokuWiki
- Gollum
- MediaWiki
- Wiki.js

OPS11-BP05 개선 추진 요인 정의

개선 기회를 평가하고 우선순위를 지정할 수 있도록 개선 추진 요인을 파악합니다.

AWS에서는 모든 운영 활동, 워크로드 및 인프라의 로그를 집계하여 상세 활동 이력을 생성할 수 있습니다. 그런 후에는 AWS 도구를 통해 시간별 운영 및 워크로드 상태를 분석하여 추진 요인을 기반으로 개선 기회를 파악할 수 있습니다. 예를 들어, 추세를 파악하고, 이벤트/활동과 성과의 상관 관계를 지정하고, 여러 환경과 시스템을 비교/대조할 수 있습니다.

CloudTrail을 사용해 AWS Management Console, CLI, SDK 및 API를 통해 API 활동을 추적하여 모든 계정에서 수행되는 작업을 확인해야 합니다. CloudTrail 및 CloudWatch를 사용하여 AWS 개발자 도구 배포 활동을 추적합니다. 이렇게 하면 배포의 자세한 활동 이력과 해당 결과가 CloudWatch Logs 로그 데이터에 추가됩니다.

장기간 저장을 위해 Amazon S3로 로그 데이터 내보내기 작업을 수행합니다. 여러분은 AWS Glue를 사용하여 분석을 위해 Amazon S3의 로그 데이터를 검색 및 준비할 수 있습니다. 또한 Amazon Athena를 사용하면 AWS Glue와의 기본 통합을 통해 로그 데이터를 분석할 수 있습니다. 여러분은 Amazon QuickSight 와 같은 비즈니스 인텔리전스 도구를 사용하여 데이터를 시각화하고 탐색하며 분석할 수 있습니다.

일반적인 안티 패턴:

- 작동하지만 부적절한 스크립트가 있습니다. 시간을 들여 재작성합니다. 이제 완벽합니다.
- 스타트업이 벤처 투자자로부터 또 다른 자금 지원을 받으려고 합니다. 고객은 PCI DSS 규정 준수를 입증하기를 원합니다. 고객의 요구를 들어주고 싶지만 규정 준수를 문서화하고 고객의 배송 날짜를 놓치면 고객을 잃게 됩니다. 잘못된 일은 아니었지만 옳은 일이었는지 궁금합니다.

이 모범 사례 수립의 이점: 개선에 사용할 기준을 결정하면 이벤트 기반 동기 또는 감정적 투자의 영향을 최소화할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 개선 추진 요인 파악: 원하는 결과가 지원되는 경우에만 시스템을 변경해야 합니다.
  - 필요한 기능: 개선 기회를 평가할 때 필요한 기능을 평가합니다.
    - AWS의 새로운 소식
  - 반드시 수정해야 할 문제: 개선 기회를 평가할 때 반드시 수정해야 할 문제, 버그 및 취약점을 평가합니다.
    - 최신 AWS 보안 공지
    - AWS Trusted Advisor
  - 규정 준수 요건: 개선 기회를 검토할 때 규정/정책 준수 상태를 유지하거나 타사의 지원을 계속 받으려는 데 필요한 업데이트와 변경 사항을 평가합니다.
    - AWS 규정 준수
    - AWS 규정 준수 프로그램
    - AWS 규정 준수 최신 소식

리소스

관련 문서:

- Amazon Athena
- Amazon QuickSight
- AWS 규정 준수
- AWS 규정 준수 최신 소식
- AWS 규정 준수 프로그램
- AWS Glue
- 최신 AWS 보안 공지
- AWS Trusted Advisor
- 장기간 저장을 위해 Amazon S3로 로그 데이터 내보내기
- AWS의 새로운 소식

OPS11-BP06 인사이트 검증

여러 부문의 팀 및 비즈니스 소유자와 함께 분석 결과와 응답을 검토합니다. 이러한 검토에서는 개선 가능성을 공통적으로 파악하고, 추가적인 영향을 확인하고, 조치 과정을 결정할 수 있습니다. 필요에 따라 대응 내용을 조정합니다.

일반적인 안티 패턴:

- 시스템에서 CPU 사용률이 95%이며 시스템의 부하를 줄이는 방법을 찾기 위해 이를 최우선 과제로 삼습니다. 가장 좋은 조치는 확장하는 것입니다. 시스템은 트랜스코더이며 항상 95%의 CPU 사용률로 실행되도록 확장됩니다. 시스템 소유자에게 문의했다면 상황을 설명할 수 있었을 것입니다. 시간이 낭비되었습니다.
- 시스템 소유자는 시스템이 미션 크리티컬하다고 주장합니다. 시스템이 보안 수준이 높은 환경에 배치되지 않았습니다. 보안을 강화하기 위해 미션 크리티컬 시스템에 필요한 탐지 및 예방 제어 기능을 추가로 구현합니다. 작업이 완료되고 추가 리소스에 대한 요금이 청구될 것임을 시스템 소유자에게 알립니다. 알림을 받고 진행한 논의에서 시스템 소유자는 시스템이 충족하지 못하는 미션 크리티컬 시스템에 대한 공식적인 정의가 있음을 알게 됩니다.

이 모범 사례 정립의 이점: 비즈니스 소유자 및 주제 전문가와 함께 인사이트를 확인하여 공통된 이해를 확립하고 개선 사항을 좀 더 효과적으로 안내할 수 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 인사이트 검증: 비즈니스 소유자 및 주제별 전문가와 협력하여 수집한 데이터의 의미에 대한 공통된 이해와 동의가 있는지 확인합니다. 추가 우려 사항, 잠재적 영향을 식별하고 조치 과정을 결정합니다.

OPS11-BP07 운영 지표 검토 수행

다양한 실무 영역의 여러 팀 구성원들과 함께 운영 지표 후행 분석을 정기적으로 수행합니다. 이러한 검토에서는 개선 기회와 진행 가능한 조치 과정을 파악하고 배운 내용을 공유할 수 있습니다.

개발, 테스트, 프로덕션 등 모든 환경에서 향상 기회를 모색해야 합니다.

일반적인 안티 패턴:

- 유지 관리 기간으로 인해 상당한 소매 프로모션이 중단되었습니다. 기업에서는 비즈니스에 영향을 미치는 다른 이벤트가 있는 경우 지연될 수 있는 표준 유지 관리 기간이 있음을 모릅니다.

- 조직에서 일반적으로 사용되는 버그가 많은 라이브러리 사용으로 인해 장기간 가동이 중단되었습니다. 이후 신뢰할 수 있는 라이브러리로 마이그레이션했습니다. 조직의 다른 팀들은 그들이 위험에 처해 있다는 것을 알지 못합니다. 정기적으로 만나고 이 인시던트를 검토했다면 이들도 위험을 알았을 것입니다.
- 트랜스코더의 성능이 지속적으로 저하되어 미디어 팀에 영향을 미치고 있습니다. 아직 심각하지는 않습니다. 인시던트를 발생시키기에 충분히 나쁜 상태가 될 때까지 알 수 없습니다. 미디어 팀과 운영 지표를 검토했다면 지표의 변화와 그 경험을 인식하고 문제를 해결할 기회가 있었을 것입니다.
- 고객 SLA 만족도를 검토하고 있지 않습니다. 고객 SLA를 충족하지 못하는 추세입니다. 고객 SLA를 충족하지 못할 경우 재정적 징벌이 부과될 수 있습니다. 이러한 SLA의 지표를 정기적으로 검토했다면 문제를 파악하고 해결할 수 있는 기회가 있었을 것입니다.

이 모범 사례 정립의 이점: 운영 지표, 이벤트 및 인시던트를 검토하기 위해 정기적으로 회의를 진행하여 팀 전체에서 공통된 이해를 유지하고, 배운 내용을 공유하고, 개선 사항의 우선순위와 대상을 정합니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 운영 지표 검토: 다양한 실무 영역의 여러 팀 구성원들과 함께 운영 지표 후행 분석을 정기적으로 수행합니다. 실무 팀, 개발 팀, 운영 팀 등의 이해 관계자와 함께 즉각적인 피드백 및 후행 분석에서 발견된 사항을 확인하고 파악한 내용을 공유합니다. 그리고 이러한 인사이트를 활용하여 개선 기회와 진행 가능한 조치 과정을 확인합니다.
  - Amazon CloudWatch
  - Amazon CloudWatch 지표 사용
  - 사용자 지정 지표 게시
  - Amazon CloudWatch 지표 및 차원 참조

리소스

관련 문서:

- Amazon CloudWatch
- Amazon CloudWatch 지표 및 차원 참조
- 사용자 지정 지표 게시
- Amazon CloudWatch 지표 사용

OPS11-BP08 파악한 내용 문서화 및 공유

운영 활동 과정에서 파악한 내용을 문서화하고 공유하여 내부적으로, 그리고 여러 팀 간에 사용할 수 있도록 하십시오.

조직 전체에서 관련 이점을 더욱 효율적으로 활용하려면 팀에서 파악한 내용을 공유해야 합니다. 피할 수 있는 오류를 방지하고 개발 작업을 쉽게 수행하기 위해 정보와 리소스를 공유할 수 있습니다. 이렇게 하면 원하는 기능을 제공하는 데 집중할 수 있습니다.

AWS Identity and Access Management(IAM)를 사용하여 계정 내/계정 간에 공유할 리소스 액세스를 제어할 수 있는 권한을 정의합니다. 그런 다음 버전 제어 AWS CodeCommit 리포지토리를 사용하여 애플리케이션 라이브러리, 스크립팅된 절차, 절차 설명서 및 기타 시스템 설명서를 공유해야 합니다. 여러 계정에 Lambda 함수 사용 권한을 부여하고 AMI 액세스를 공유하는 방식을 통해 컴퓨팅 표준을 공유합니다. 인프라 표준도 AWS CloudFormation 템플릿으로 공유해야 합니다.

AWS API 및 SDK를 사용하면 GitHub, BitBucket, SourceForge 등의 외부 및 타사 도구와 리포지토리를 통합할 수 있습니다. 파악한 내용과 개발한 기능을 공유할 때는 공유 리포지토리 무결성을 유지할 수 있도록 권한의 구조를 지정해야 합니다.

일반적인 안티 패턴:

- 조직에서 일반적으로 사용되는 버그가 많은 라이브러리 사용으로 인해 장기간 가동이 중단되었습니다. 이후 신뢰할 수 있는 라이브러리로 마이그레이션했습니다. 조직의 다른 팀들은 그들이 위험에 처해 있다는 것을 알지 못합니다. 이 라이브러리에 대한 경험을 문서화하고 공유했다면 그들도 위험에 대해 알았을 것입니다.
- 내부적으로 공유된 마이크로서비스에서 세션 중단을 일으키는 엣지 사례를 발견했습니다. 이 엣지 사례를 방지하기 위해 서비스에 대한 호출을 업데이트했습니다. 조직의 다른 팀들은 그들이 위험에 처해 있다는 것을 알지 못합니다. 이 라이브러리에 대한 경험을 문서화하고 공유했다면 그들도 위험에 대해 알았을 것입니다.
- 마이크로서비스 중 하나에 대한 CPU 사용률 요구 사항을 크게 줄일 수 있는 방법을 찾았습니다. 다른 팀에서 이 기술을 활용할 수 있는지 여부는 알 수 없습니다. 이 라이브러리에 대한 경험을 문서화하고 공유했다면 그들에게도 그렇게 할 기회가 있었을 것입니다.

이 모범 사례 수립의 이점: 개선을 지원하고 경험의 이점을 극대화하기 위해 파악한 내용을 공유합니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 파악한 내용 문서화 및 공유: 운영 활동을 실행하면서 파악한 내용과 후행 분석 결과를 문서화하는 절차를 마련하여 다른 팀에서도 사용할 수 있도록 합니다.
  - 파악한 내용 공유: 파악한 내용 및 관련 작업 결과를 팀 간에 공유할 수 있는 절차를 마련합니다. 예를 들어, 접속 가능한 위키를 통해 새로워진 절차, 지침, 거버넌스 및 모범 사례를 공유하십시오. 스크립트, 코드 및 라이브러리는 공동 리포지토리를 통해 공유할 수 있습니다.
    - AWS 환경에 대한 액세스 위임
    - AWS CodeCommit 리포지토리 공유
    - 간편한 AWS Lambda 함수 인증
    - 특정 AWS 계정과 AMI 공유
    - AWS CloudFormation Designer URL로 템플릿 공유 속도 향상
    - Amazon SNS에서 AWS Lambda 사용

리소스

관련 문서:

- 간편한 AWS Lambda 함수 인증
- AWS CodeCommit 리포지토리 공유
- 특정 AWS 계정과 AMI 공유
- AWS CloudFormation Designer URL로 템플릿 공유 속도 향상
- Amazon SNS에서 AWS Lambda 사용

관련 동영상:

- AWS 환경에 대한 액세스 위임

OPS11-BP09 개선을 위한 시간 할애

프로세스 내에서 전담 리소스와 시간을 할애하여 가능한 범위 내에서 점진적 개선을 지속적으로 수행합니다.

AWS에서는 실험과 테스트의 위험, 작업량 및 비용을 줄일 수 있는 환경의 임시 복제본을 생성할 수 있습니다. 이렇게 복제된 환경을 사용하여 분석의 결론을 테스트하고, 실험을 진행하고, 계획된 향상 내용을 개발/테스트할 수 있습니다.

일반적인 안티 패턴:

- 애플리케이션 서버에 알려진 성능 문제가 있습니다. 이는 계획된 모든 기능 구현 뒤의 백로그에 추가됩니다. 추가 예정인 계획된 기능의 비율이 일정하게 유지되는 경우 성능 문제는 해결되지 않습니다.
- 지속적인 개선 지원을 위해 관리자 및 개발자가 개선 사항을 선택하고 구현하는 데 여분의 시간을 모두 할애하는 것을 승인합니다. 개선이 완료되지 않습니다.

이 모범 사례 정립의 이점: 프로세스 내에서 전담 리소스와 시간을 할애하여 가능한 범위 내에서 점진적 개선을 지속적으로 수행합니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 개선을 위한 시간 할애: 프로세스에 리소스와 시간을 투자하여 가능한 범위 내에서 점진적이고 지속적인 개선을 수행합니다. 변경 사항을 적용하여 결과를 개선하고, 평가를 통하여 성공 여부를 확정합니다. 결과가 목표에 미치지 못하지만 여전히 개선을 우선해야 한다면 다른 대안을 찾아서 진행합니다.

보안

보안 원칙에는 클라우드 기술을 활용하여 보안을 강화하고 데이터, 시스템 및 자산을 보호하는 능력이 포함됩니다. 구현 방법에 대한 권장 가이드는 보안 원칙 백서에서 확인할 수 있습니다.

모범 사례 영역

- 보안 기초
- 자격 증명 및 액세스 관리
- 탐지
- 인프라 보호
- 데이터 보호
- 사고 대응
- 애플리케이션 보안

보안 기초

질문

SEC 1 워크로드를 안전하게 운영하려면 어떻게 해야 합니까?

워크로드를 안전하게 운영하려면 모든 보안 영역에 포괄적 모범 사례를 적용해야 합니다. 조직 및 워크로드 수준에서 운영 우수성에 정의된 요구 사항과 프로세스를 가져와 모든 영역에 적용합니다. AWS 및 업계 권장 사항 및 위협 인텔리전스를 최신 상태로 유지하면 위협 모델 및 제어 목표를 발전시키는 데 도움이 됩니다. 보안 프로세스, 테스트 및 검증을 자동화함으로써 보안 작업을 확장할 수 있습니다.

모범 사례

- SEC01-BP01 계정을 사용하여 워크로드 분리
- SEC01-BP02 계정 루트 사용자 및 속성 보호
- SEC01-BP03 제어 목표 파악 및 검증
- SEC01-BP04 최신 보안 위협 정보 파악
- SEC01-BP05 최신 보안 권장 사항 파악
- SEC01-BP06 파이프라인에서 보안 제어의 테스트 및 검증 자동화
- SEC01-BP07 위협 모델을 사용하여 위협 식별 및 완화 조치의 우선순위 지정
- SEC01-BP08 새로운 보안 서비스 및 기능을 정기적으로 평가 및 구현

SEC01-BP01 계정을 사용하여 워크로드 분리

다중 계정 전략을 통해 환경(예: 프로덕션, 개발 및 테스트)과 워크로드 간에 일반적인 가드레일 및 격리를 설정합니다. 계정 수준의 분리는 보안, 청구 및 액세스에 대한 강력한 격리 경계를 제공하므로 강력하게 권장됩니다.

원하는 결과: 클라우드 운영, 관련 없는 워크로드 및 환경을 별도의 계정으로 격리하여 클라우드 인프라 전반의 보안을 강화하는 계정 구조입니다.

일반적인 안티 패턴:

- 데이터 민감도 수준이 서로 다른 관련 없는 여러 워크로드를 동일한 계정에 배치합니다.
- 잘못 정의된 조직 단위(OU) 구조입니다.

이 모범 사례 확립의 이점:

- 워크로드가 의도치 않게 액세스되는 경우 영향 범위 감소

- AWS 서비스, 리소스 및 리전에 대한 액세스 권한의 중앙 거버넌스
- 보안 서비스의 정책 및 중앙 집중식 관리를 통해 클라우드 인프라의 보안 유지 관리
- 자동화된 계정 생성 및 유지 관리 프로세스
- 규정 준수 및 규제 요구 사항에 대한 인프라의 중앙 집중식 감사

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

AWS 계정는 서로 다른 민감도 수준에서 작동하는 워크로드 또는 리소스 간에 보안 격리 경계를 제공합니다. AWS는 다중 계정 전략을 통해 대규모로 클라우드 워크로드를 관리할 수 있는 도구를 제공하여 이 격리 경계를 활용할 수 있습니다. AWS에 대한 다중 계정 전략의 개념, 패턴 및 구현에 대한 지침은 여러 계정을 사용하여 AWS 환경 구성을 참조하세요.

중앙 관리 하에 여러 AWS 계정가 있는 경우 조직 단위(OU) 계층으로 정의된 계층 구조로 계정을 구성해야 합니다. 그런 다음 보안 제어를 구성하고 OU 및 멤버 계정에 적용하여 조직의 멤버 계정에 일관된 예방적인 제어를 설정할 수 있습니다. 보안 제어는 상속되므로 OU 계층 구조의 하위 수준에 있는 멤버 계정이 사용 가능한 권한을 필터링할 수 있습니다. 우수한 설계는 이 상속을 활용하여 각 멤버 계정에 대해 원하는 보안 제어를 달성하는 데 필요한 보안 정책의 수와 복잡성을 줄입니다.

AWS Organizations 및 AWS Control Tower는 AWS 환경에서 이 다중 계정 구조를 구현하고 관리하는 데 사용할 수 있는 두 가지 서비스입니다. AWS Organizations를 사용하면 하나 이상의 OU 계층으로 정의된 계층 구조로 계정을 구성할 수 있습니다. 각 OU에는 여러 멤버 계정이 포함되어 있습니다. 서비스 제어 정책(SCP)을 사용하면 조직 관리자가 멤버 계정에 대한 세분화된 예방 제어를 설정할 수 있으며, AWS Config를 사용하면 멤버 계정에 대한 사전 예방 및 탐지 제어를 설정할 수 있습니다. 많은 AWS 서비스가 AWS Organizations와 통합되어 위임된 관리 제어를 제공하고 조직의 모든 멤버 계정 전체의 서비스별 작업을 수행합니다.

AWS Organizations를 기반의 AWS Control Tower는 랜딩 존이 있는 다중 계정 AWS 환경에 대한 원클릭 모범 사례 설정을 제공합니다. 랜딩 존은 Control Tower가 구축한 다중 계정 환경의 진입점입니다. Control Tower는 AWS Organizations에 비해 몇 가지 이점을 제공합니다. 개선된 계정 거버넌스를 제공하는 세 가지 이점은 다음과 같습니다.

- 조직에 참여하도록 승인된 계정에 자동으로 적용되는 통합 필수 보안 가드레일
- 주어진 OU 세트에 대해 활성화 또는 비활성화할 수 있는 선택적 가드레일
- AWS Control Tower Account Factory는 조직 내에서 사전 승인된 기준 및 구성 옵션이 포함된 계정의 자동 배포를 제공합니다.

구현 단계

1. 조직 단위 구조 설계: 적절하게 설계된 조직 단위 구조는 서비스 제어 정책 및 기타 보안 제어를 생성하고 유지 관리하는 데 필요한 관리 부담을 줄여줍니다. 조직 단위 구조는 비즈니스 요구 사항, 데이터 민감도 및 워크로드 구조에 맞춰 조정해야 합니다.
2. 다중 계정 환경을 위한 랜딩 존 생성: 랜딩 존은 조직이 워크로드를 신속하게 개발, 실행 및 배포할 수 있는 일관된 보안 및 인프라 기반을 제공합니다. 맞춤형으로 구축된 랜딩 존 또는 AWS Control Tower를 사용하여 환경을 오케스트레이션할 수 있습니다.
3. 가드레일 설정: 랜딩 존을 통해 환경에 일관된 보안 가드레일을 구현합니다. AWS Control Tower는 배포할 수 있는 필수 및 선택적 제어 목록을 제공합니다. 필수 제어는 Control Tower를 구현할 때 자동으로 배포됩니다. 적극 권장되는 선택적 제어 목록을 검토하고 요구 사항에 적합한 제어를 구현합니다.
4. 새로 추가된 리전에 대한 액세스 권한 제한: 새 AWS 리전의 경우 사용자 및 역할과 같은 IAM 리소스는 사용자가 지정하는 리전으로만 전파됩니다. 이 작업은 Control Tower를 사용할 때 콘솔을 통해 수행하거나 AWS Organizations에서 IAM 권한 정책을 조정하여 수행할 수 있습니다.
5. AWS CloudFormation StackSets 고려: StackSets를 사용하면 IAM 정책, 역할, 그룹을 포함한 리소스를 승인된 템플릿에서 다양한 AWS 계정과 리전에 배포할 수 있습니다.

리소스

관련 모범 사례:

- SEC02-BP04 중앙 집중식 자격 증명 공급자 사용

관련 문서:

- AWS Control Tower
- AWS 보안 감사 지침
- IAM 모범 사례
- CloudFormation StackSets를 사용하여 여러 AWS 계정 및 리전에 리소스 프로비저닝
- Organizations FAQ
- AWS Organizations 용어 및 개념
- AWS Organizations 다중 계정 환경의 서비스 제어 정책 모범 사례
- AWS 계정 관리 참조 가이드

- 여러 계정을 사용하여 AWS 환경 구성

관련 동영상:

- Enable AWS adoption at scale with automation and governance(자동화 및 거버넌스를 통해 대규모 AWS 도입 지원)
- Security Best Practices the Well-Architected Way(Well-Architected 방식의 보안 모범 사례)
- Building and Governing Multiple Accounts using AWS Control Tower(AWS Control Tower를 사용하여 여러 계정 구축 및 관리)
- Enable Control Tower for Existing Organizations(기존 조직에 Control Tower 활성화)

관련 워크숍:

- Control Tower Immersion Day

SEC01-BP02 계정 루트 사용자 및 속성 보호

루트 사용자는 계정 내의 모든 리소스에 대한 전체 관리 액세스 권한이 있는 AWS 계정에서 가장 권한이 높은 사용자이며 경우에 따라 보안 정책의 제약을 받을 수 없습니다. 루트 사용자에 대한 프로그래밍 방식 액세스 권한을 비활성화하고, 루트 사용자에 대한 적절한 제어를 설정하며, 루트 사용자의 일상적인 사용을 방지하면 루트 보안 인증 정보가 의도치 않게 노출되어 클라우드 환경이 손상될 위험을 줄일 수 있습니다.

원하는 결과: 루트 사용자를 보호하면 루트 사용자 보안 인증 정보의 남용으로 인해 우발적이거나 의도적인 손상이 발생할 가능성을 줄일 수 있습니다. 탐지 제어를 설정하면 루트 사용자를 사용하여 작업을 수행할 때 적절한 담당자에게 알릴 수도 있습니다.

일반적인 안티 패턴:

- 루트 사용자 보안 인증 정보가 필요한 몇 가지 작업 이외의 작업에 루트 사용자를 사용합니다.
- 긴급 상황에서의 중요한 인프라, 프로세스 및 인력 기능을 확인하기 위한 비상 계획을 정기적으로 테스트하는 데 소홀합니다.
- 일반적인 계정 로그인 흐름만 고려하고 대체 계정 복구 방법을 고려하거나 테스트하는 데 소홀합니다.
- DNS, 이메일 서버 및 전화 공급자가 계정 복구 흐름에서 사용되므로 중요한 보안 경계의 일부로 처리하지 않습니다.

이 모범 사례 확립의 이점: 루트 사용자에 대한 액세스를 보호하면 계정의 작업이 제어되고 감사된다는 확신을 얻을 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

AWS는 계정을 보호하는 데 도움이 되는 다양한 도구를 제공합니다. 그러나 이러한 조치는 대부분 기본적으로 활성화되어 있지 않으므로 이를 구현하기 위해서는 직접적인 조치를 취해야 합니다. AWS 계정 보호를 위한 기본 단계로 다음 권장 사항을 고려해 보세요. 이러한 단계를 구현할 때 보안 제어를 지속적으로 평가하고 모니터링하는 프로세스를 구축하는 것이 중요합니다.

AWS 계정를 처음 생성할 때 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 단일 자격 증명을 생성하는 것으로 시작합니다. 이때 생성하는 자격 증명을 AWS 계정 루트 사용자라고 합니다. 계정을 생성할 때 사용한 이메일 주소와 암호를 사용하여 루트 사용자로 로그인할 수 있습니다. AWS 루트 사용자에게 부여된 높은 액세스 권한으로 인해 특별히 이를 필요로 하는 작업을 수행하려면 AWS 루트 사용자의 사용을 제한해야 합니다. 루트 사용자 로그인 보안 인증 정보는 철저하게 보호되어야 하며 AWS 계정 루트 사용자에 대해 항상 다중 인증(MFA)을 활성화해야 합니다.

사용자 이름, 암호 및 다중 인증(MFA) 디바이스를 사용하여 루트 사용자에 로그인하는 일반적인 인증 흐름 외에도 이메일 주소 및 계정과 연결된 전화번호에 대한 액세스 권한이 부여된 AWS 계정 루트 사용자에 로그인하는 계정 복구 흐름이 있습니다. 따라서 복구 이메일이 전송되는 루트 사용자 이메일 계정을 보호하는 것과 계정과 연결된 전화번호를 보호하는 것이 동일하게 중요합니다. 또한 루트 사용자와 연결된 이메일 주소가 동일한 AWS 계정의 이메일 서버 또는 도메인 이름 서비스(DNS) 리소스에서 호스팅되는 잠재적인 순환 종속성도 고려하세요.

AWS Organizations를 사용하는 경우 각각 루트 사용자가 있는 여러 AWS 계정가 있습니다. 하나의 계정이 관리 계정으로 지정되면 여러 계층의 멤버 계정을 관리 계정 아래 추가할 수 있습니다. 관리 계정의 루트 사용자 보호에 우선순위를 지정한 다음 멤버 계정 루트 사용자의 주소를 기재합니다. 관리 계정의 루트 사용자를 보호하기 위한 전략은 멤버 계정 루트 사용자와 다를 수 있으며, 멤버 계정 루트 사용자에게 예방 보안 제어를 적용할 수 있습니다.

구현 단계

루트 사용자에 대한 제어를 설정하려면 다음 구현 단계를 수행하는 것이 좋습니다. 해당하는 경우 권장 사항은 CIS AWS Foundations 벤치마크 버전 1.4.0과 상호 참조됩니다. 이러한 단계 외에도 AWS 계정 및 리소스 보호에 대한 AWS 모범 사례 지침을 참조하세요.

예방 제어

1. 계정에 대한 정확한 연락처 정보를 설정합니다.
    a. 이 정보는 분실한 암호 복구 흐름, 분실한 MFA 디바이스 계정 복구 흐름 및 팀과의 중요한 보안 관련 커뮤니케이션에 사용됩니다.
    b. 회사 도메인에서 호스팅하는 이메일 주소(배포 목록 선호)를 루트 사용자의 이메일 주소로 사용합니다. 개인의 이메일 계정이 아닌 배포 목록을 사용하면 장기간에 걸쳐 루트 계정에 액세스할 수 있는 추가 중복성과 연속성이 제공됩니다.
    c. 연락처 정보에 표시된 전화번호는 이 목적을 위한 보안 전용 전화여야 합니다. 전화번호를 표시하거나 다름 사람과 공유해서는 안 됩니다.
2. 루트 사용자의 액세스 키를 생성하지 않습니다. 액세스 키가 있으면 제거합니다(CIS 1.4).
    a. 루트 사용자에 대한 수명이 긴 프로그래밍 방식 보안 인증 정보(액세스 및 보안 암호 키)를 제거합니다.
    b. 루트 사용자 액세스 키가 이미 있는 경우 해당 키를 사용하여 AWS Identity and Access Management(IAM) 역할에서 임시 액세스 키를 사용하도록 프로세스를 전환한 다음 루트 사용자 액세스 키를 삭제해야 합니다.
3. 루트 사용자의 보안 인증 정보를 저장해야 하는지 여부를 결정합니다.
    a. AWS Organizations를 사용하여 새 멤버 계정을 생성하는 경우 새 멤버 계정에 대한 루트 사용자의 초기 암호가 사용자에게 노출되지 않는 임의의 값으로 설정됩니다. 필요한 경우 AWS 조직 관리 계정의 암호 재설정 흐름을 사용하여 멤버 계정에 대한 액세스 권한을 얻는 것이 좋습니다.
    b. 독립 실행형 AWS 계정 또는 관리 AWS 조직 계정의 경우 루트 사용자에 대한 보안 인증 정보를 생성하고 안전하게 저장하는 것이 좋습니다. 루트 사용자에 대해 MFA를 활성화합니다.
4. AWS 다중 계정 환경에서 멤버 계정 루트 사용자에 대한 예방 제어를 활성화합니다.
    a. 멤버 계정에 대해 루트 사용자의 루트 액세스 키 생성 차단 예방 가드레일을 활성화하는 것이 좋습니다.
    b. 멤버 계정에 대해 루트 사용자로서의 작업 차단 예방 가드레일을 활성화하는 것이 좋습니다.
5. 루트 사용자에 대한 보안 인증 정보가 필요한 경우:
    a. 복잡한 암호를 사용합니다.
    b. 루트 사용자, 특히 AWS Organizations 관리(지급인) 계정에 대해 다중 인증(MFA)을 활성화합니다(CIS 1.5).
    c. 단일 사용 디바이스는 MFA 코드가 포함된 디바이스가 다른 용도로 재사용될 가능성을 줄일 수 있으므로 복원력 및 보안을 위해 하드웨어 MFA 디바이스를 고려합니다. 배터리로 구동되는 하드웨어 MFA 디바이스가 정기적으로 대체되는지 확인합니다. (CIS 1.6)
        - 루트 사용자에 대해 MFA를 구성하려면 가상 MFA 또는 하드웨어 MFA 디바이스를 활성화하는 지침을 따르세요.

d. 백업을 위해 여러 MFA 디바이스를 등록하는 것이 좋습니다. 계정당 최대 8개의 MFA 디바이스가 허용됩니다.
   - 루트 사용자에 대해 둘 이상의 MFA 디바이스를 등록하면 MFA 디바이스가 손실된 경우 계정을 복구하는 흐름이 자동으로 비활성화됩니다.

e. 암호를 안전하게 저장하고, 암호를 전자적으로 저장하는 경우 순환 종속성을 고려합니다. 암호를 획득하는 데 동일한 AWS 계정에 대한 액세스 권한이 필요한 방식으로 암호를 저장하지 않습니다.

6. 선택 사항: 루트 사용자에 대한 주기적인 암호 교체 일정을 설정하는 것이 좋습니다.
   - 보안 인증 정보 관리 모범 사례는 규정 및 정책 요구 사항에 따라 다릅니다. MFA로 보호되는 루트 사용자는 단일 인증 요소로 암호에 의존하지 않습니다.
   - 주기적으로 루트 사용자 암호를 변경하면 의도치 않게 노출된 암호가 남용될 위험이 줄어듭니다.

탐지 제어

- 루트 보안 인증 정보의 사용을 감지하는 경보를 생성합니다(CIS 1.7). Amazon GuardDuty를 활성화하면 RootCredentialUsage 결과를 통해 루트 사용자 API 보안 인증 정보 사용을 모니터링하고 알립니다.
- AWS Config용 AWS Well-Architected 보안 원칙 규정 준수 팩에 포함된 탐지 제어를 평가 및 구현하거나 AWS Control Tower를 사용하는 경우 Control Tower 내에서 사용 가능한 강력하게 권장되는 제어를 평가 및 구현합니다.

운영 지침

- 조직에서 루트 사용자 보안 인증 정보에 대한 액세스 권한을 갖는 담당자를 결정합니다.
  - 루트 사용자 액세스 권한을 획득하는 데 필요한 모든 보안 인증 정보 및 MFA에 대한 액세스 권한을 한 명의 개인이 갖지 않도록 2인 규칙을 사용합니다.
  - 한 명의 개인이 아닌 조직에서 계정과 연결된 전화번호 및 이메일 별칭(암호 재설정 및 MFA 재설정 흐름에 사용됨)에 대한 제어 권한을 유지 관리하는지 확인합니다.
- 루트 사용자는 예외적으로만 사용합니다(CIS 1.7).
  - AWS 루트 사용자는 관리 작업이라 하더라도 일상 작업에는 사용하면 안 됩니다. 루트 사용자가 필요한 AWS 작업을 수행하려면 루트 사용자로만 로그인합니다.. 다른 모든 작업은 적절한 역할이 있는 다른 사용자가 수행해야 합니다.
- 루트 사용자 보안 인증 정보를 사용해야 하는 긴급 상황이 발생하기 전에 절차를 테스트할 수 있도록 루트 사용자에 대한 액세스 권한이 작동하는지 주기적으로 확인합니다.

- 계정과 연결된 이메일 주소와 대체 연락처에 나열된 이메일 주소가 작동하는지 주기적으로 확인합니다. 다음에서 수신된 보안 알림이 있는지 이러한 이메일 받은 편지함을 <abuse@amazon.com>모니터링합니다. 또한 계정과 연결된 모든 전화번호가 작동하는지 확인합니다.
- 루트 계정 남용에 대응하기 위한 인시던트 대응 절차를 준비합니다. AWS 계정에 대한 인시던트 대응 전략을 구축하는 방법에 대한 자세한 내용은 AWS 보안 인시던트 대응 가이드 및 보안 원칙 백서의 인시던트 대응 섹션에 있는 모범 사례를 참조하세요.

리소스

관련 모범 사례:

- SEC01-BP01 계정을 사용하여 워크로드 분리
- SEC02-BP01 강력한 로그인 메커니즘 사용
- SEC03-BP02 최소 권한 액세스 부여
- SEC03-BP03 긴급 액세스 프로세스 설정
- SEC10-BP05 액세스 권한 사전 프로비저닝

관련 문서:

- AWS Control Tower
- AWS 보안 감사 지침
- IAM 모범 사례
- Amazon GuardDuty – 루트 보안 인증 정보 사용 알림
- CloudTrail을 통한 루트 보안 인증 정보 사용 모니터링에 대한 단계별 지침
- AWS에서 사용하도록 승인된 MFA 토큰
- AWS에서 브레이크 글라스 액세스 구현
- AWS 계정에서 개선해야 할 10가지 보안 항목
- AWS 계정에서 무단 활동이 발견되면 어떻게 해야 하나요?

관련 동영상:

- Enable AWS adoption at scale with automation and governance(자동화 및 거버넌스를 통해 대규모 AWS 도입 지원)
- Security Best Practices the Well-Architected Way(Well-Architected 방식의 보안 모범 사례)

- Limiting use of AWS root credentials from AWS re:inforce 2022 – Security best practices with AWS IAM(AWS re:inforce 2022 – AWS IAM의 보안 모범 사례의 AWS 루트 보안 인증 정보 사용 제한)

관련 예시 및 실습:

- 실습: AWS 계정 and root user(AWS 계정 및 루트 사용자)

SEC01-BP03 제어 목표 파악 및 검증

위협 모델에서 식별된 규정 준수 요구 사항 및 위험을 기준으로, 워크로드에 적용해야 하는 제어 목표와 제어 항목을 도출하고 검증합니다. 제어 목표 및 제어에 대한 지속적인 검증은 위험 완화의 효과를 측정하는 데 도움이 됩니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 규정 준수 요구 사항 파악: 워크로드가 준수해야 하는 조직/법적/규정 준수 요구 사항을 파악합니다.
- AWS 규정 준수 리소스 파악: 규정 준수를 지원하기 위해 AWS에서 제공하는 리소스를 파악합니다.
  - https://aws.amazon.com/compliance/
  - https://aws.amazon.com/artifact/

리소스

관련 문서:

- AWS 보안 감사 지침
- 보안 공지

관련 동영상:

- AWS Security Hub: 보안 알림 관리 및 규정 준수 자동화
- Well-Architected 방식의 보안 모범 사례

SEC01-BP04 최신 보안 위협 정보 파악

적절한 제어 수단을 정의하고 구현하는 데 도움이 되도록 최신 보안 위협 정보를 바탕으로 공격 벡터를 파악합니다. AWS Managed Services를 사용하면 AWS 계정에서 예기치 않거나 일반적이지 않은 동작에 대한 알림을 더 쉽게 받을 수 있습니다. 보안 정보 흐름의 일부로 AWS 파트너 도구 또는 서드 파티 위협 정보 피드를 사용하여 조사합니다. 이 일반적인 취약점 및 노출(CVE) 목록 에는 최신 정보를 얻기 위해 사용할 수 있는 공개된 사이버 보안 취약점이 수록되어 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 위협 정보 출처 구독: 워크로드에 사용된 기술과 관련이 있는 여러 출처의 위협 정보를 정기적으로 검토합니다.
  - 일반적인 취약점 및 노출 목록
- 실시간으로 워크로드를 모니터링하고, 보안 문제를 식별하고, 근본 원인 분석 및 수정을 신속하게 처리할 수 있도록 AWS Shield Advanced 서비스: 인터넷을 통해 워크로드에 액세스할 수 있는 경우 인텔리전스 소스에 대한 실시간에 가까운 가시성을 제공합니다.

리소스

관련 문서:

- AWS 보안 감사 지침
- AWS Shield
- 보안 공지

관련 동영상:

- Well-Architected 방식의 보안 모범 사례

SEC01-BP05 최신 보안 권장 사항 파악

워크로드의 보안 태세를 강화하기 위해 최신 AWS 및 업계 보안 권장 사항을 모두 파악합니다. AWS 보안 공지 에는 보안 및 개인정보 알림에 대한 중요한 정보가 포함되어 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- AWS 업데이트 팔로우: 새로운 권장 사항, 팁 및 요령을 구독하거나 정기적으로 확인합니다.
  - AWS Well-Architected 실습
  - AWS 보안 블로그
  - AWS 서비스 설명서
- 업계 뉴스 구독: 워크로드에 사용된 기술과 관련이 있는 여러 출처의 뉴스 피드를 정기적으로 검토합니다.
  - 예시: 일반적인 취약점 및 노출 목록

리소스

관련 문서:

- 보안 공지

관련 동영상:

- Well-Architected 방식의 보안 모범 사례

SEC01-BP06 파이프라인에서 보안 제어의 테스트 및 검증 자동화

빌드, 파이프라인 및 프로세스의 일부로서 테스트 및 검증되는 보안 메커니즘에 대한 보안 기준과 템플릿을 설정합니다. 도구와 자동화를 사용하여 모든 보안 제어를 지속적으로 테스트하고 검증합니다. 예를 들어 시스템 이미지와 인프라 같은 항목을 코드 템플릿으로 삼아 단계마다 설정된 기준에 따라 보안 취약성, 불규칙성, 드리프트를 검사합니다. AWS CloudFormation Guard는 CloudFormation 템플릿이 안전하고 시간을 절약해 주며 구성 오류의 위험을 줄여주는지 확인하는 데 도움이 됩니다.

프로덕션 환경에 적용되는 잘못된 보안 구성의 수를 줄여야 하므로, 빌드 프로세스에서 품질 관리와 결함 감소 과정을 많이 수행할수록 좋습니다. 가능한 경우 항상 보안 문제를 테스트하도록 지속적 통합 및 지속적 배포(CI/CD) 파이프라인을 설계합니다. CI/CD 파이프라인은 빌드 및 전달의 각 단계에서 보안을 강화할 기회를 제공합니다. 새로운 위협을 완화하기 위해 CI/CD 보안 도구도 업데이트해야 합니다.

워크로드 구성의 변경 사항을 추적하여 규정 준수 감사, 관리 변경, 적용 가능한 조사에 도움을 받습니다. AWS Config를 사용하여 AWS 및 서드 파티 리소스를 기록하고 평가할 수 있습니다. 이를 통해 전

체적인 규정 준수를 규칙 및 규정 준수 팩(해결 조치가 포함된 규칙의 모음)으로 지속적으로 감사 및 평가할 수 있습니다.

변경 추적에는 조직의 변화 제어 프로세스에 포함된 정기적 변경 사항(MACD - 이동/추가/변경/삭제라고도 함), 계획하지 않은 변경 사항 또는 사고 등 예기치 않은 변경 사항도 포함됩니다. 인프라에서 변경이 이루어지기도 하지만 코드 리포지토리 변경, 머신 이미지 및 애플리케이션 인벤토리 변경, 프로세스 및 정책 변경 또는 문서 변경 등 다른 카테고리와 관련된 경우도 있을 수 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 구성 관리 자동화: 구성 관리 서비스 또는 도구를 사용하여 보안 구성을 자동으로 적용하고 확보합니다.
  - AWS Systems Manager
  - AWS CloudFormation
  - AWS에서 CI/CD 파이프라인 설정

리소스

관련 문서:

- AWS Organization의 여러 계정에서 서비스 제어 정책을 사용해 권한 가드레일을 설정하는 방법

관련 동영상:

- AWS Organizations를 사용하여 다중 계정 AWS 환경 관리
- Well-Architected 방식의 보안 모범 사례

SEC01-BP07 위협 모델을 사용하여 위협 식별 및 완화 조치의 우선순위 지정

위협 모델링을 수행하여 워크로드에 대한 잠재적 위협 및 관련 완화 조치의 최신 등록을 식별하고 유지 관리합니다. 보안 위협 우선순위를 지정하고 보안 제어 완화 조치를 조정하여 방지, 감지 및 대응합니다. 워크로드와 진화하는 보안 환경에 맞춰 이를 보완하고 유지 관리합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

위협 모델링이란 무엇인가요?

정의에 따르면 "위협 모델링은 가치 있는 것을 보호한다는 맥락에서 위협과 완화 조치를 식별, 전달 및 이해하기 위해 작동합니다." – OWASP(Open Web Application Security Project) 애플리케이션 위협 모델링

위협 모델을 사용해야 하는 이유는 무엇인가요?

시스템은 복잡하고 시간이 지남에 따라 점점 더 복잡해지고 기능이 향상되어 더 많은 비즈니스 가치를 제공하고 고객 만족도와 참여도를 향상시킵니다. 즉, IT 설계를 결정할 때는 계속해서 증가하는 사용 사례를 고려해야 합니다. 이러한 복잡성과 사용 사례 순열의 수는 일반적으로 위협을 찾고 완화하는 데 구조화되지 않은 접근 방식을 비효율적으로 만듭니다. 대신 시스템에 대한 잠재적인 위협을 열거할 뿐만 아니라 조직의 제한된 리소스가 시스템의 전체 보안 태세를 개선하는 데 최대한 영향을 미칠 수 있도록 완화 조치를 고안하고 우선순위를 지정하는 체계적인 접근 방식이 필요합니다.

위협 모델링은 수명 주기 후반에 비해 상대적으로 완화 조치 관련 비용과 노력이 적게 필요한 설계 프로세스 초기에 문제를 찾아 해결하는 것을 목표로 이러한 체계적인 접근 방식을 제공하도록 설계되었습니다. 이 접근 방식은 'shift-left' 보안이라는 업계 원칙에 부합합니다. 궁극적으로 위협 모델링은 조직의 위험 관리 프로세스와 통합되며 위협 기반 접근 방식을 사용하여 구현할 제어에 대한 결정을 내리는 데 도움이 됩니다.

위협 모델링은 언제 수행해야 하나요?

워크로드의 수명 주기에서 가능한 한 빠르게 위협 모델링을 시작합니다. 그러면 식별한 위협에 대해 유연성을 높일 수 있습니다. 소프트웨어 버그와 마찬가지로 위협을 조기에 식별할수록 위협을 해결하는 것이 더 비용 효율적입니다. 위협 모델은 최신 상태를 유지해야 하는 문서로 워크로드가 변경됨에 따라 계속 진화해야 합니다. 주요 변경 사항이 있거나 위협 환경이 변경되거나 새로운 기능 또는 서비스를 채택하는 경우를 포함하여 시간이 지남에 따라 위협 모델을 보완합니다.

구현 단계

위협 모델링을 수행하려면 어떻게 해야 하나요?

위협 모델링을 수행하는 방법에는 여러 가지가 있습니다. 프로그래밍 언어와 마찬가지로 각각 장단점이 있으므로 자신에게 가장 적합한 방법을 선택해야 합니다. 한 가지 접근 방식은 위협 모델링 연습에 구조를 제공하기 위해 개방형 질문을 제시하는 위협 모델링에 대한 Shostack의 4가지 질문 프레임으로 시작하는 것입니다.

1. 어떤 작업을 하고 있나요?

   이 질문의 목적은 구축 중인 시스템과 보안과 관련된 해당 시스템에 대한 세부 정보를 이해하고 동의하는 데 도움을 주기 위한 것입니다. 모델이나 다이어그램을 생성하는 것은 데이터 흐름 다이어그램을 사용하여 구축 항목을 시각화하는 데 도움이 되므로 이 질문에 답하는 가장 일반적인 방법입니다. 시스템에 대한 권한 수임과 중요한 세부 정보를 작성하는 것도 범위를 정의하는 데 도움이 됩니다. 이를 통해 위협 모델에 기여하는 모든 담당자가 동일한 작업에 집중하고 범위 이외의 주제(시스템의 오래된 버전 포함)로 벗어나 시간을 허비하는 것을 방지할 수 있습니다. 예를 들어 웹 애플리케이션을 구축하는 경우, 사용자 설계를 통해 영향을 미칠 수 없기 때문에 브라우저 클라이언트에 대한 운영 체제의 신뢰할 수 있는 부팅 시퀀스에 대해 위협 모델링을 수행할 필요가 없습니다.

2. 잘못될 일이 뭐가 있을까요?

   이 질문을 통해 시스템에 대한 위협을 식별합니다. 위협은 원치 않는 영향을 미치고 시스템의 보안에 영향을 미칠 수 있는 우발적이거나 의도적인 작업 또는 이벤트입니다. 무엇이 잘못될 수 있는지에 대한 명확한 이해 없이는 위협에 대해 대응할 수 있는 방법이 아무 것도 없습니다.

   무엇이 잘못될 수 있는지에 대한 표준 목록은 없습니다. 이 목록을 작성하려면 팀 내의 모든 구성원과 위협 모델링 수행에 관여하는 관련 대상자 간의 브레인스토밍과 협업이 필요합니다. STRIDE와 같은 위협 식별 모델을 사용하여 브레인스토밍을 지원할 수 있습니다. 이 모델은 스푸핑, 변조, 거부, 정보 공개, 서비스 거부 및 권한 상승과 같이 평가할 다양한 범주를 제안합니다. 또한 OWASP Top 10, HiTrust Threat Catalog, 조직의 자체 위협 카탈로그를 포함하여 아이디어를 얻을 수 있는 기존 목록과 연구를 검토하여 브레인스토밍을 지원할 수 있습니다.

3. 이에 대해 무엇을 할 수 있나요?

   이전 질문의 경우와 마찬가지로 가능한 모든 완화 조치에 대한 표준 목록은 없습니다. 이 단계에 대한 입력은 이전 단계에서 식별된 위협, 행위자 및 개선 영역입니다.

   보안 및 규정 준수는 사용자와 AWS의 공동 책임입니다. '이에 대해 무엇을 할 수 있나요?'라고 질문할 때 '이에 대한 책임은 누구에게 있나요?'라고 질문하는 것임을 이해하는 것이 중요합니다. 사용자와 AWS 간의 책임 균형을 이해하면 위협 모델링 수행의 범위를 관리되는 완화 조치로 지정하는 데 도움이 됩니다. 이러한 완화 조치는 일반적으로 AWS 서비스 구성 옵션과 자체 시스템별 완화 조치의 조합으로 이루어집니다.

   AWS 부분의 공동 책임에 대한 경우, AWS 서비스가 많은 규정 준수 프로그램의 범위에 속해 있다는 것을 알 수 있습니다. 이러한 프로그램을 통해 클라우드의 보안 및 규정 준수를 유지하기 위해 AWS에 마련된 강력한 제어 기능을 이해할 수 있습니다. 이러한 프로그램의 감사 보고서는 AWS 고객이 AWS Artifact에서 다운로드할 수 있습니다.

사용 중인 AWS 서비스에 관계없이 항상 고객 책임의 요소가 있으며 이러한 책임에 맞는 완화 조치가 위협 모델에 포함되어야 합니다. AWS 서비스 자체에 대한 보안 제어 완화 조치를 위해 자격 증명 및 액세스 관리(인증 및 권한 부여), 데이터 보호(저장 데이터 및 전송 중 데이터), 인프라 보안, 로깅, 모니터링과 같은 도메인을 포함한 도메인 전반에서 보안 제어 구현을 고려해야 합니다. 각 AWS 서비스에 대한 설명서에는 완화 조치로 고려할 보안 제어에 대한 지침을 제공하는 보안 전용 섹션이 있습니다. 작성 중인 코드와 해당 코드 종속성을 고려하고 이러한 위협을 해결하기 위해 마련할 수 있는 제어에 대해 생각하는 것이 중요합니다. 이러한 제어는 입력 검증, 세션 처리 및 경계 처리와 같은 것일 수 있습니다. 대부분의 취약성은 사용자 지정 코드에서 발생하는 경우가 많으므로 이 영역에 집중하세요.

4. 잘 수행했나요?

   목표는 팀과 조직이 위협 모델의 품질과 시간이 지남에 따라 위협 모델링을 수행하는 속도를 모두 개선하는 것입니다. 이러한 개선은 연습, 학습, 지도, 검토의 조합에서 비롯됩니다. 더 자세히 알아보고 실습하려면 사용자와 팀이 Threat modeling the right way for builders(빌더에게 적합한 위협 모델링) 교육 과정 또는 워크숍을 완료하는 것이 좋습니다. 또한 위협 모델링을 조직의 애플리케이션 개발 수명 주기에 통합하는 방법에 대한 지침은 AWS 보안 블로그에서 How to approach threat modeling(위협 모델링 접근 방식) 게시물을 참조하세요.

리소스

관련 모범 사례:

- SEC01-BP03 제어 목표 파악 및 검증
- SEC01-BP04 최신 보안 위협 정보 파악
- SEC01-BP05 최신 보안 권장 사항 파악
- SEC01-BP08 새로운 보안 서비스 및 기능을 정기적으로 평가 및 구현

관련 문서:

- 위협 모델링 접근 방식(AWS 보안 블로그)
- NIST: 데이터 중심 시스템 위협 모델링을 위한 가이드

관련 동영상:

- AWS Summit ANZ 2021 - How to approach threat modelling(AWS Summit ANZ 2021 - 위협 모델링 접근 방식)

- AWS Summit ANZ 2022 - Scaling security – Optimise for fast and secure delivery(AWS Summit ANZ 2022 - 보안 확장 – 빠르고 안전한 제공을 위한 최적화)

관련 교육:

- Threat modeling the right way for builders(빌더에게 적합한 위협 모델링) – AWS Skill Builder 자습형 온라인 교육
- Threat modeling the right way for builders(빌더에게 적합한 위협 모델링) – AWS 워크숍

SEC01-BP08 새로운 보안 서비스 및 기능을 정기적으로 평가 및 구현

워크로드의 보안 상태를 개선할 수 있는 AWS 및 AWS 파트너의 보안 서비스와 기능을 평가하고 구현합니다. AWS 보안 블로그에서는 새로운 AWS 서비스 및 기능, 구현 가이드, 일반적인 보안 가이드를 강조합니다. AWS의 새로운 소식은 모든 신규 AWS 기능, 서비스, 공지 사항을 파악하는 유용한 방법입니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 일반 검토 계획: 규정 준수 요구 사항, 새 AWS 보안 기능/서비스 평가, 업계 최신 소식 확인 등의 검토 활동 일정을 생성합니다.
- AWS 서비스 및 기능 발견: 사용 중인 서비스에 적용 가능한 보안 기능을 검색하고 새로 릴리스되는 기능을 검토합니다.
  - AWS 보안 블로그
  - AWS 보안 공지
  - AWS 서비스 설명서
- AWS 서비스 온보딩 프로세스 정의: 새로운 AWS 서비스를 온보딩하는 프로세스를 정의합니다. 이 과정에서 새 AWS 서비스의 기능과 워크로드의 규정 준수 요구 사항을 평가할 방법도 정의합니다.
- 신규 서비스 및 기능 테스트: 프로덕션 환경을 거의 동일하게 복제한 비프로덕션 환경에서, 새로 릴리스하는 서비스 및 기능을 테스트합니다.
- 다른 방어 메커니즘 구현: 워크로드를 방어하기 위한 자동화된 메커니즘을 구현하고 사용 가능한 옵션을 살펴봅니다.
  - AWS Config 규칙에 따른 규정 미준수 AWS 리소스 문제 해결

리소스

관련 동영상:

- Well-Architected 방식의 보안 모범 사례

자격 증명 및 액세스 관리

질문

- SEC 2 사람과 시스템에 대한 인증은 어떻게 관리합니까?
- SEC 3 사람과 시스템에 대한 권한은 어떻게 관리합니까?

SEC 2 사람과 시스템에 대한 인증은 어떻게 관리합니까?

안전한 AWS 워크로드 운영에 접근할 때 관리해야 하는 2가지 유형의 자격 증명이 있습니다. 액세스 권한을 관리하고 부여하는 데 필요한 자격 증명의 유형을 이해하면 적절한 자격 증명이 적절한 조건에서 적절한 리소스에 액세스할 수 있도록 보장할 수 있습니다.

인적 자격 증명: 관리자, 개발자, 운영자 및 최종 사용자가 AWS 환경과 애플리케이션에 액세스하려면 자격 증명이 필요합니다. 이들은 조직의 구성원이거나 협업하는 외부 사용자로, 웹 브라우저, 클라이언트 애플리케이션 또는 대화형 명령줄 도구를 통해 AWS 리소스와 상호 작용합니다.

시스템 자격 증명: 서비스 애플리케이션, 운영 도구 및 워크로드에서 AWS 서비스에 요청하려면(예: 데이터 읽기) 자격 증명이 필요합니다. 이러한 자격 증명에는 Amazon EC2 인스턴스 또는 AWS Lambda 함수와 같이 AWS 환경에서 실행되는 시스템이 포함됩니다. 액세스가 필요한 외부 당사자의 시스템 자격 증명을 관리할 수도 있습니다. 또한 AWS 환경에 액세스해야 하는 시스템이 AWS 외부에 있을 수도 있습니다.

모범 사례

- SEC02-BP01 강력한 로그인 메커니즘 사용
- SEC02-BP02 임시 보안 인증 정보 사용
- SEC02-BP03 안전하게 보안 암호 저장 및 사용
- SEC02-BP04 중앙 집중식 자격 증명 공급자 사용
- SEC02-BP05 정기적으로 보안 인증 정보 감사 및 교체
- SEC02-BP06 사용자 그룹 및 속성 활용

SEC02-BP01 강력한 로그인 메커니즘 사용

로그인(로그인 보안 인증 정보를 사용한 인증)은 다중 인증(MFA)과 같은 메커니즘을 사용하지 않을 때, 특히 로그인 보안 인증 정보가 의도치 않게 공개되었거나 쉽게 추측되는 상황에서 위험을 초래할 수 있습니다. 강력한 로그인 메커니즘을 사용하면 MFA 및 강력한 암호 정책을 요구하여 이러한 위험을 줄일 수 있습니다.

원하는 결과: AWS Identity and Access Management(IAM) 사용자, AWS 계정 루트 사용자, AWS IAM Identity Center(AWS Single Sign-On의 후속 서비스), 타사 자격 증명 공급자에 대한 강력한 로그인 메커니즘을 사용하여 AWS의 보안 인증 정보에 대한 의도치 않은 액세스 위험을 줄입니다. 즉, MFA를 요구하고 강력한 암호 정책을 적용하며 비정상적인 로그인 동작을 감지합니다.

일반적인 안티 패턴:

- 복잡한 암호 및 MFA를 포함하여 자격 증명에 대한 강력한 암호 정책을 적용하지 않습니다.
- 다른 사용자 간에 동일한 보안 인증 정보를 공유합니다.
- 의심스러운 로그인에 대한 탐지 제어를 사용하지 않습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

인적 자격 증명으로 AWS에 로그인하는 방법에는 여러 가지가 있습니다. AWS로 인증할 때 페더레이션(직접 페더레이션 또는 AWS IAM Identity Center 사용)을 사용하는 중앙 집중식 자격 증명 공급자를 사용하는 것이 AWS 모범 사례입니다. 이 경우 자격 증명 공급자 또는 Microsoft Active Directory를 사용하여 보안 로그인 프로세스를 설정해야 합니다.

AWS 계정를 처음 열면 AWS 계정 루트 사용자로 시작합니다. 루트 사용자 계정은 사용자(및 루트 사용자가 필요한 작업)에 대한 액세스 권한을 설정할 때만 사용해야 합니다. AWS 계정를 연 직후에는 계정 루트 사용자에 대해 MFA를 활성화하고 AWS 모범 사례 가이드를 사용하여 루트 사용자를 보호하는 것이 중요합니다.

AWS IAM Identity Center에서 사용자를 생성하는 경우 해당 서비스의 로그인 프로세스를 보호하세요. 소비자 자격 증명의 경우 Amazon Cognito user pools를 사용하고 해당 서비스에서 로그인 프로세스를 보호하거나 Amazon Cognito user pools가 지원하는 자격 증명 공급자 중 하나를 사용할 수 있습니다.

AWS Identity and Access Management(IAM) 사용자를 사용하는 경우 IAM을 사용하여 로그인 프로세스를 보호합니다.

로그인 방법에 관계없이 강력한 로그인 정책을 적용하는 것이 중요합니다.

구현 단계

다음은 일반적인 강력한 로그인 권장 사항입니다. 구성하는 실제 설정은 회사 정책에 따라 설정하거나 NIST 800-63과 같은 표준을 사용해야 합니다.

- MFA가 필요합니다. 인적 자격 증명 및 워크로드에 대해 MFA를 요구하는 것이 IAM 모범 사례입니다. MFA를 활성화하면 사용자가 로그인 보안 인증 정보와 일회용 암호(OTP) 또는 하드웨어 디바이스에서 암호로 확인 및 생성된 문자열을 제공해야 하는 추가 보안 계층이 제공됩니다.
- 암호 강도의 기본 요소인 최소 암호 길이를 적용합니다.
- 암호 복잡성을 적용하여 암호를 추측하기 어렵게 만듭니다.
- 사용자가 자신의 암호를 변경할 수 있도록 허용합니다.
- 공유 보안 인증 정보 대신 개별 자격 증명을 생성합니다. 개별 자격 증명을 생성하여 각 사용자에게 고유한 보안 보안 인증 정보를 제공할 수 있습니다. 개별 사용자는 각 사용자의 활동을 감사할 수 있는 기능을 제공합니다.

IAM Identity Center 권장 사항:

- IAM Identity Center는 암호 길이, 복잡성 및 재사용 요구 사항을 설정하는 기본 디렉터리를 사용할 때 사전 정의된 암호 정책을 제공합니다.
- MFA를 활성화하고 자격 증명 소스가 기본 디렉터리, AWS Managed Microsoft AD 또는 AD Connector인 경우 MFA에 대한 컨텍스트 인식 또는 상시 설정을 구성합니다.
- 사용자가 자신의 MFA 디바이스를 등록하도록 허용합니다.

Amazon Cognito user pools 디렉터리 권장 사항:

- 암호 강도 설정을 구성합니다.
- 사용자에게 MFA 설정을 요구합니다.
- 의심스러운 로그인을 차단할 수 있는 적응형 인증과 같은 기능에 대해 Amazon Cognito user pools 고급 보안 설정을 사용합니다.

IAM 사용자 권장 사항:

- IAM Identity Center 또는 직접 페더레이션을 사용하는 것이 좋습니다. 그러나 IAM 사용자가 필요할 수 있습니다. 이 경우 IAM 사용자에 대해 암호 정책을 설정합니다. 암호 정책을 사용하여 최소 길이 또는 알파벳 이외 문자 포함 여부 등과 같은 요구 사항을 정의할 수 있습니다.

- 사용자가 자신의 암호화 MFA 디바이스를 관리할 수 있도록 MFA 로그인을 적용하는 IAM 정책을 생성합니다.

리소스

관련 모범 사례:

- SEC02-BP03 안전하게 보안 암호 저장 및 사용
- SEC02-BP04 중앙 집중식 자격 증명 공급자 사용
- SEC03-BP08 안전하게 조직과 리소스 공유

관련 문서:

- AWS IAM Identity Center(AWS Single Sign-On의 후속 서비스) 암호 정책
- IAM 사용자 암호 정책
- AWS 계정 루트 사용자 암호 설정
- Amazon Cognito 암호 정책
- AWS 보안 인증 정보
- IAM 보안 모범 사례

관련 동영상:

- Managing user permissions at scale with AWS IAM Identity Center(AWS IAM Identity Center를 사용하여 대규모로 사용자 권한 관리)
- Mastering identity at every layer of the cake(케이크의 모든 계층에서 자격 증명 마스터링)

SEC02-BP02 임시 보안 인증 정보 사용

모든 유형의 인증을 수행할 때 보안 인증 정보가 의도치 않게 공개, 공유 또는 도난 당하는 위험을 줄이거나 제거하기 위해 장기 보안 인증 정보 대신 임시 보안 인증 정보를 사용하는 것이 가장 좋습니다.

원하는 결과: 장기 보안 인증 정보의 위험을 줄이려면 가능한 한 인적 자격 증명과 시스템 자격 증명 모두에 대해 임시 보안 인증 정보를 사용합니다. 장기 보안 인증 정보는 많은 위험을 초래합니다. 예를 들어 퍼블릭 GitHub 리포지토리에 코드를 업로드할 수 있습니다. 임시 보안 인증 정보를 사용하면 보안 인증 정보가 손상될 가능성이 크게 줄어듭니다.

일반적인 안티 패턴:

- 개발자가 페더레이션을 사용하여 CLI에서 임시 보안 인증 정보를 획득하는 대신 IAM users의 장기 액세스 키를 사용합니다.
- 개발자가 장기 액세스 키를 코드에 포함하고 해당 코드를 퍼블릭 Git 리포지토리에 업로드합니다.
- 개발자가 앱 스토어에서 사용할 수 있도록 장기 액세스 키를 모바일 앱에 포함합니다.
- 사용자가 다른 사용자와 장기 액세스 키를 공유하거나 직원이 장기 액세스 키를 소유한 상태로 퇴사합니다.
- 임시 보안 인증 정보를 사용할 수 있는 경우에도 시스템 자격 증명에 장기 액세스 키를 사용합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

모든 AWS API 및 CLI 요청에 대해 장기 보안 인증 정보 대신 임시 보안 인증 정보를 사용합니다. AWS 서비스에 대한 API 및 CLI 요청은 거의 모든 경우에 AWS 액세스 키를 사용하여 서명해야 합니다. 이러한 요청은 임시 또는 장기 보안 인증 정보로 서명할 수 있습니다. 장기 액세스 키라고도 하는 장기 보안 인증 정보를 사용해야 하는 유일한 경우는 IAM 사용자 또는 AWS 계정 루트 사용자를 사용하는 경우입니다. 다른 방법을 통해 AWS에 페더레이션하거나 IAM 역할을 수임할 때 임시 보안 인증 정보가 생성됩니다. 로그인 보안 인증 정보를 사용하여 AWS Management Console에 액세스하는 경우에도 AWS 서비스를 호출할 수 있도록 임시 보안 인증 정보가 생성됩니다. 장기 보안 인증 정보가 필요한 경우는 거의 없으며 임시 보안 인증 정보를 사용하여 대부분의 작업을 수행할 수 있습니다.

임시 보안 인증 정보를 선호하여 장기 보안 인증 정보 사용을 피하는 것은 페더레이션 및 IAM 역할을 선호하여 IAM 사용자의 사용을 줄이는 전략과 함께 수행해야 합니다. 과거에는 IAM 사용자가 인적 자격 증명과 시스템 자격 증명 모두에 사용되었지만 이제는 장기 액세스 키 사용의 위험을 피하기 위해 사용하지 않는 것이 좋습니다.

구현 단계

직원, 관리자, 개발자, 운영자 및 고객과 같은 인적 자격 증명의 경우:

- 중앙 집중식 자격 증명 공급자를 사용해야 하며 인적 사용자가 임시 보안 인증 정보를 사용하여 AWS에 액세스하려면 자격 증명 공급자와의 페더레이션을 사용해야 합니다. 사용자에 대한 페더레이션은 각 AWS 계정에 대한 직접 페더레이션을 사용하거나 AWS IAM Identity Center(AWS IAM Identity Center의 후속 서비스) 및 선택한 자격 증명 공급자를 사용하여 수행할 수 있습니다. 페더레이션은 장기 보안 인증 정보를 제거하는 것 외에도 IAM 사용자를 사용하는 것보다 많은 이점을 제공

합니다. 사용자는 직접 페더레이션에 대한 명령줄에서 또는 IAM Identity Center를 사용하여 임시 보안 인증 정보를 요청할 수도 있습니다. 즉, IAM 사용자 또는 사용자에 대한 장기 보안 인증 정보가 필요한 사용 사례가 거의 없습니다.

- 서비스형 소프트웨어(SaaS) 공급자와 같은 타사에 AWS 계정의 리소스에 대한 액세스 권한을 부여할 때 크로스 계정 역할 및 리소스 기반 정책을 사용할 수 있습니다.
- 소비자 또는 고객에게 AWS 리소스에 대한 액세스 권한을 부여해야 하는 경우 Amazon Cognito 자격 증명 풀 또는 Amazon Cognito user pools을 사용하여 임시 보안 인증 정보를 제공할 수 있습니다. 보안 인증 정보에 대한 권한은 IAM 역할을 통해 구성됩니다. 또한 인증되지 않은 게스트 사용자의 경우, 권한이 제한된 별도의 IAM 역할을 정의할 수 있습니다.

시스템 자격 증명의 경우 장기 보안 인증 정보를 사용해야 할 수 있습니다. 이러한 경우 워크로드에서 AWS에 액세스하려면 IAM 역할과 함께 임시 보안 인증 정보를 사용해야 합니다.

- Amazon Elastic Compute Cloud(Amazon EC2)의 경우 Amazon EC2에 대한 역할을 사용할 수 있습니다.

- AWS Lambda를 사용하면 임시 보안 인증 정보를 사용하여 AWS 작업을 수행할 수 있는 서비스 권한을 부여하도록 Lambda 실행 역할을 구성할 수 있습니다. IAM 역할을 사용하여 임시 보안 인증 정보를 부여하는 AWS 서비스에 대한 다른 유사한 모델이 많이 있습니다.
- IoT 디바이스의 경우 AWS IoT Core 보안 인증 정보 공급자를 사용하여 임시 보안 인증 정보를 요청할 수 있습니다.
- 온프레미스 시스템 또는 AWS 리소스에 액세스해야 하는 AWS 외부에서 실행되는 시스템의 경우 IAM Roles Anywhere를 사용할 수 있습니다.

임시 보안 인증 정보를 사용할 수 없고 장기 보안 인증 정보를 사용해야 하는 시나리오가 있습니다. 이러한 상황에서는 보안 인증 정보를 주기적으로 감사 및 교체하고 장기 보안 인증 정보가 필요한 사용 사례를 위해 정기적으로 액세스 키를 교체합니다. 장기 보안 인증 정보가 필요할 수 있는 몇 가지 예로는 WordPress 플러그인 및 타사 AWS 클라이언트가 있습니다. 장기 보안 인증 정보를 사용해야 하는 상황이나 데이터베이스 로그인과 같이 AWS 액세스 키 이외의 보안 인증 정보에 대해 AWS Secrets Manager와 같은 보안 암호 관리를 처리하도록 설계된 서비스를 사용할 수 있습니다. Secrets Manager를 사용하면 지원되는 서비스를 사용하여 암호화된 보안 암호를 간단하게 관리 및 교체하고 안전하게 저장할 수 있습니다. 장기 보안 인증 정보 교체에 대한 자세한 내용은 액세스 키 교체를 참조하세요.

리소스

관련 모범 사례:

- SEC02-BP03 안전하게 보안 암호 저장 및 사용
- SEC02-BP04 중앙 집중식 자격 증명 공급자 사용
- SEC03-BP08 안전하게 조직과 리소스 공유

관련 문서:

- 임시 보안 인증 정보
- AWS 보안 인증 정보
- IAM 보안 모범 사례
- IAM 역할
- IAM Identity Center
- 자격 증명 공급자 및 페더레이션
- 액세스 키 교체
- 보안 파트너 솔루션: 액세스 및 액세스 제어
- AWS 계정 루트 사용자

관련 동영상:

- Managing user permissions at scale with AWS IAM Identity Center(successor to AWS IAM Identity Center)(AWS IAM Identity Center(AWS IAM Identity Center의 후속 버전)를 사용하여 대규모로 사용자 권한 관리)
- Mastering identity at every layer of the cake(케이크의 모든 계층에서 자격 증명 마스터링)

SEC02-BP03 안전하게 보안 암호 저장 및 사용

워크로드에는 데이터베이스, 리소스 및 타사 서비스에 대한 아이덴티티를 증명하는 자동화된 기능이 필요합니다. 이는 API 액세스 키, 암호, OAuth 토큰과 같은 보안 암호 액세스 보안 인증 정보를 사용하여 수행됩니다. 특별 제작된 서비스를 사용하여 이러한 보안 인증 정보를 저장, 관리 및 교체하면 해당 보안 인증 정보가 손상될 가능성을 줄이는 데 도움이 됩니다.

원하는 결과: 다음 목표를 달성하는 애플리케이션 보안 인증 정보를 안전하게 관리하기 위한 메커니즘을 구현합니다.

- 워크로드에 필요한 보안 암호를 식별합니다.

- 가능한 경우 장기 보안 인증 정보를 단기 보안 인증 정보로 대체하여 필요한 장기 보안 인증 정보의 수를 줄입니다.
- 나머지 장기 보안 인증 정보의 안전한 저장 및 자동 교체를 설정합니다.
- 워크로드에 존재하는 보안 암호에 대한 액세스 권한을 감사합니다.
- 개발 프로세스 중에 소스 코드에 보안 암호가 포함되어 있지 않은지 확인하기 위해 지속적으로 모니터링합니다.
- 보안 인증 정보가 의도치 않게 공개될 가능성을 줄입니다.

일반적인 안티 패턴:

- 자격 증명을 교체하지 않습니다.
- 소스 코드 또는 구성 파일에 장기 보안 인증 정보를 저장합니다.
- 보안 인증 정보를 저장 시 암호화되지 않은 상태로 저장합니다.

이 모범 사례 확립의 이점:

- 보안 암호는 저장 시 및 전송 중 암호화된 상태로 저장됩니다.
- 보안 인증 정보에 대한 액세스 권한은 API를 통해 제한됩니다(보안 인증 정보 벤딩 머신으로 간주).
- 보안 인증 정보에 대한 액세스 권한(읽기 및 쓰기 모두)은 감사되고 로깅됩니다.
- 우려 사항 분리: 보안 인증 정보 교체는 아키텍처의 나머지 부분과 분리될 수 있는 별도의 구성 요소에 의해 수행됩니다.
- 보안 암호는 필요에 따라 소프트웨어 구성 요소에 자동으로 배포되며 중앙 위치에서 교체가 수행됩니다.
- 보안 인증 정보에 대한 액세스 권한은 세분화된 방식으로 제어할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

과거에는 데이터베이스, 타사 API, 토큰 및 기타 보안 암호에 인증하는 데 사용되는 보안 인증 정보가 소스 코드나 환경 파일에 포함되었을 수 있습니다. AWS는 이러한 보안 인증 정보를 안전하게 저장하고 자동으로 교체하며 사용을 감사하는 여러 메커니즘을 제공합니다.

보안 암호 관리에 접근하는 가장 좋은 방법은 제거, 대체 및 교체 지침을 따르는 것입니다. 가장 안전한 보안 인증 정보는 저장, 관리 또는 처리할 필요가 없는 보안 인증 정보입니다. 안전하게 제거할 수 있는 워크로드 기능에 더 이상 필요하지 않은 보안 인증 정보가 있을 수 있습니다.

워크로드의 적절한 기능을 위해 여전히 필요한 보안 인증 정보의 경우 장기 보안 인증 정보를 임시 또는 단기 보안 인증 정보로 대체할 기회가 있을 수 있습니다. 예를 들어 AWS 비밀 액세스 키를 하드 코딩하는 대신 IAM 역할을 사용하여 해당 장기 보안 인증 정보를 임시 보안 인증 정보로 대체하는 것이 좋습니다.

일부 수명이 긴 보안 암호는 제거하거나 대체하지 못할 수 있습니다. 이러한 보안 암호는 AWS Secrets Manager와 같은 서비스에 저장될 수 있으며 정기적으로 중앙에서 저장, 관리 및 교체할 수 있습니다.

워크로드의 소스 코드 및 구성 파일을 감사하면 여러 유형의 보안 인증 정보를 확인할 수 있습니다. 다음 표에는 일반적인 유형의 보안 인증 정보를 처리하기 위한 전략이 요약되어 있습니다.

Credential type	Description	Suggested strategy
IAM access keys	AWS IAM access and secret keys used to assume IAM roles inside of a workload	Replace: Use IAM 역할 assigned to the compute instances (such as Amazon EC2 or AWS Lambda) instead. For interoperability with third parties that require access to resources in your AWS 계정, ask if they support AWS 크로스 계정 액세스. For mobile apps, consider using temporary credentials through Amazon Cognito 자격 증명 풀 (페더레이션 자격 증명). For workloads running outside of AWS, consider IAM Roles Anywhere or AWS Systems Manager 하이브리드 활성화.
SSH keys	Secure Shell private keys used to log into Linux EC2	Replace: Use AWS Systems Manager or EC2 Instance Connect to provide

Credential type	Description	Suggested strategy
	instances, manually or as part of an automated process	programmatic and human access to EC2 instances using IAM roles.
Application and database credentials	Passwords – plain text string	Rotate: Store credentials in AWS Secrets Manager and establish automated rotation if possible.
Amazon RDS and Aurora Admin Database credentials	Passwords – plain text string	Replace: Use the Amazon RDS와 Secrets Manager 통합 or Amazon Aurora. In addition, some RDS database types can use IAM roles instead of passwords for some use cases (for more detail, see IAM 데이터베이스 인증).
OAuth tokens	Secret tokens – plain text string	Rotate: Store tokens in AWS Secrets Manager and configure automated rotation.
API tokens and keys	Secret tokens – plain text string	Rotate: Store in AWS Secrets Manager and establish automated rotation if possible.

일반적인 안티 패턴은 소스 코드, 구성 파일 또는 모바일 앱 내부에 IAM 액세스 키를 포함하는 것입니다. AWS 서비스와 통신하기 위해 IAM 액세스 키가 필요한 경우 임시(단기) 보안 인증 정보를 사용합니다. 이러한 단기 보안 인증 정보는 EC2 인스턴스의 경우 IAM 역할, Lambda 함수의 경우 실행 역할, 모바일 사용자 액세스의 경우 Cognito IAM 역할 및 IoT 기기의 경우 IoT Core 정책을 통해 제공할 수 있습니다. 타사와 인터페이스할 때 IAM 사용자를 구성하고 타사에 해당 사용자의 비밀 액세스 키를 보내는 것보다 계정 리소스에 필수 액세스 권한이 있는 IAM 역할에 대한 액세스 권한을 위임하는 것이 좋습니다.

워크로드에 다른 서비스 및 리소스와 상호 운용하는 데 필요한 보안 암호를 저장해야 하는 경우가 많습니다. AWS Secrets Manager는 이러한 보안 인증 정보는 물론 API 토큰, 암호 및 기타 보안 인증 정보의 저장, 사용 및 교체를 안전하게 관리하기 위해 특별히 제작되었습니다.

AWS Secrets Manager는 민감한 보안 인증 정보의 안전한 저장 및 처리를 보장하는 5가지 주요 기능, 즉 저장 시 암호화, 전송 중 암호화, 종합적 감사, 세분화된 액세스 제어, 확장 가능한 보안 인증 정보 교체를 제공합니다. AWS 파트너의 기타 보안 암호 관리 서비스 또는 유사한 기능과 보증을 제공하는 현지 개발 솔루션도 허용됩니다.

구현 단계

1. Amazon CodeGuru 같은 자동화 도구를 사용하여 하드 코딩된 보안 인증 정보가 포함된 코드 경로를 식별합니다.
   - Amazon CodeGuru를 사용하여 코드 리포지토리를 스캔합니다. 검토가 완료되면 CodeGuru에서 `Type=Secrets`를 필터링하여 문제가 있는 코드 줄을 찾습니다.
2. 제거하거나 대체할 수 있는 보안 인증 정보를 식별합니다.
   a. 더 이상 필요하지 않은 보안 인증 정보를 식별하고 제거하도록 표시합니다.
   b. 소스 코드에 포함된 AWS 보안 암호 키의 경우 필요한 리소스와 연결된 IAM 역할로 대체합니다. 워크로드의 일부가 AWS 외부에 있지만 AWS 리소스에 액세스하기 위해 IAM 보안 인증 정보가 필요한 경우 IAM Roles Anywhere 또는 AWS Systems Manager 하이브리드 활성화를 고려합니다.
3. 교체 전략을 사용해야 하는 타사의 수명이 긴 보안 암호의 경우 Secrets Manager를 코드에 통합하여 런타임 시 타사 보안 암호를 검색합니다.
   a. CodeGuru 콘솔은 검색된 보안 인증 정보를 사용하여 자동으로 Secrets Manager에 보안 암호를 생성할 수 있습니다.
   b. Secrets Manager의 보안 암호 검색을 애플리케이션 코드에 통합합니다.
      - 서버리스 Lambda 함수는 언어에 구애받지 않는 Lambda 확장을 사용할 수 있습니다.
      - EC2 인스턴스 또는 컨테이너의 경우 AWS는 널리 사용되는 여러 프로그래밍 언어로 Secrets Manager에서 보안 암호를 검색하기 위한 클라이언트 측 코드 예를 제공합니다.
4. 주기적으로 코드 베이스를 검토하고 다시 스캔하여 코드에 추가된 새 보안 암호가 없는지 확인합니다.
   - git-secrets와 같은 도구를 사용하여 소스 코드 리포지토리에 새 보안 암호를 커밋되지 않도록 하는 것이 좋습니다.
5. Secrets Manager 활동을 모니터링하여 예상치 못한 사용, 부적절한 보안 암호 액세스 또는 보안 암호 삭제 시도가 있는지 확인합니다.

6. 보안 인증 정보에 대한 인적 노출을 줄입니다. 보안 인증 정보를 읽고 쓰고 수정할 수 있는 액세스 권한을 이 목적을 위한 전용 IAM 역할로 제한하고, 해당 역할을 수임할 수 있는 액세스 권한을 일부 운영 사용자에게만 제공합니다.

리소스

관련 모범 사례:

- SEC02-BP02 임시 보안 인증 정보 사용
- SEC02-BP05 정기적으로 보안 인증 정보 감사 및 교체

관련 문서:

- AWS Secrets Manager 시작하기
- 자격 증명 공급자 및 페더레이션
- Amazon CodeGuru에서 Secrets Detector 도입
- AWS Secrets Manager의 AWS Key Management Service 사용 방법
- Secrets Manager에서 보안 암호 암호화 및 복호화
- Secrets Manager 블로그 항목
- Amazon RDS에서 AWS Secrets Manager와의 통합 발표

관련 동영상:

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale(대규모 보안 암호 관리, 검색, 교체 모범 사례)
- Find Hard-Coded Secrets Using Amazon CodeGuru Secrets Detector(Amazon CodeGuru Secrets Detector를 사용하여 하드 코딩된 보안 암호 찾기)
- Securing Secrets for Hybrid Workloads Using AWS Secrets Manager(AWS Secrets Manager를 사용하여 하이브리드 워크로드에 대한 보안 암호 보호)

관련 워크숍:

- AWS Secrets Manager에서 민감한 보안 인증 정보 저장, 검색, 관리
- AWS Systems Manager 하이브리드 활성화

SEC02-BP04 중앙 집중식 자격 증명 공급자 사용

인력 자격 증명의 경우 중앙 집중식 위치에서 자격 증명을 관리할 수 있는 자격 증명 공급자를 사용합니다. 이렇게 하면 단일 위치에서 액세스를 생성, 관리 및 취소하므로 여러 애플리케이션과 서비스에 대한 액세스를 더 쉽게 관리할 수 있습니다. 예를 들어 누군가가 퇴사한다면 한 곳에서 모든 애플리케이션 및 서비스(AWS 포함)에 대한 액세스 권한을 취소할 수 있습니다. 이렇게 하면 자격 증명을 여러 개 만들 필요가 없고, 기존 인사(HR) 프로세스와 통합할 수도 있습니다.

개인 AWS 계정과 페더레이션하려면 AWS Identity and Access Management의 SAML 2.0 기반 공급자를 통해 AWS용 중앙 집중식 자격 증명을 사용할 수 있습니다. AWS에서 직접 호스팅하든, AWS 외부에서 호스팅하든, AWS Partner에서 제공하든 SAML 2.0 프로토콜과 호환되기만 하면 어느 공급자를 사용해도 됩니다. AWS 계정과 선택한 공급자 사이의 페더레이션을 활용하면 SAML 어설션을 사용해 임시 보안 인증 정보를 얻어 사용자나 애플리케이션에 AWS API 작업을 호출할 액세스 권한을 부여할 수 있습니다. 웹 기반 SSO(Single Sign-On)도 지원되므로 사용자가 로그인 웹 사이트에서 AWS Management Console에 로그인할 수 있습니다.

AWS Organizations 내 여러 계정에 페더레이션하려면 AWS IAM Identity Center(IAM Identity Center)에서 자격 증명 소스를 구성하여 사용자와 그룹이 저장될 위치를 지정하면 됩니다. 구성이 완료되면 자격 증명 공급자가 실제 소스가 되며, System for Cross-domain Identity Management(SCIM) v2.0 프로토콜을 사용해 정보를 동기화 할 수 있습니다. 그러면 사용자나 그룹을 검색하여 이들에게 AWS 계정, 클라우드 애플리케이션 또는 둘 모두에 IAM Identity Center 액세스 권한을 부여할 수 있습니다.

IAM Identity Center는 AWS Organizations와 통합되므로 자격 증명 공급자를 한 번 구성하면 조직에서 관리하는 기존 및 새 계정에 대한 액세스 권한을 부여 할 수 있습니다. IAM Identity Center는 사용자 및 그룹을 관리하는 데 사용할 수 있는 기본 스토어를 제공합니다. IAM Identity Center 스토어를 사용하는 경우, 최소 권한의 모범 사례를 염두에 두고 사용자와 그룹을 생성하고 해당 액세스 수준을 AWS 계정 및 애플리케이션에 할당합니다. 또는 SAML 2.0을 사용하여 외부 자격 증명 공급자에 연결 하거나 Microsoft AD Directory에 연결 (AWS Directory Service 사용)할 수 있습니다. 구성이 완료되면 중앙 자격 증명 공급자를 통해 인증한 후 AWS Management Console 또는 AWS 모바일 앱에 로그인할 수 있습니다.

최종 사용자 또는 워크로드 소비자(예: 모바일 앱)를 관리할 때는 다음을 사용할 수 있습니다. Amazon Cognito. 이는 웹 및 모바일 앱의 인증, 권한 부여 및 사용자 관리 등의 기능을 제공합니다. 사용자는 사용자 이름과 암호를 통해 직접 로그인하거나, Amazon, Apple, Facebook 또는 Google 등 타사를 통해 로그인할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 중앙화된 관리 액세스: AWS 계정과 ID 제공업체(idP) 간의 신뢰 관계를 구축하기 위해 Identity and Access Management(IAM) 자격 증명 공급자 엔터티를 생성합니다. IAM는 OpenID Connect (OIDC) 또는 SAML 2.0(Security Assertion Markup Language 2.0)과 호환되는 idP를 지원합니다.
  - 자격 증명 공급자 및 연동
- 중앙화된 애플리케이션 액세스: 애플리케이션 액세스 중앙화에는 Amazon Cognito를 고려하십시오. Amazon Cognito를 사용하면 웹 및 모바일 앱에 사용자 가입/로그인 기능 및 액세스 제어를 빠르고 손쉽게 추가할 수 있습니다. Amazon Cognito 는 수백만 명의 사용자로 확장되며 Facebook, Google 및 Amazon과 같은 소셜 자격 증명 공급자와 SAML 2.0을 통한 엔터프라이즈 자격 증명 공급자를 통한 로그인을 지원합니다.

- 오래된 IAM 사용자 및 그룹 제거: ID 제공업체(idP)를 사용하기 시작한 후 더 이상 필요하지 않은 IAM 사용자 및 그룹을 제거합니다.
  - 사용되지 않는 보안 인증 정보 찾기
  - IAM 그룹 삭제

리소스

관련 문서:

- IAM 모범 사례
- 보안 파트너 솔루션: 액세스 및 액세스 제어
- 임시 보안 자격 증명
- AWS 계정 루트 사용자

관련 동영상:

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale(대규모 보안 암호 관리, 검색, 교체 모범 사례)
- Managing user permissions at scale with AWS IAM Identity Center(AWS SSO를 사용하여 대규모로 사용자 권한 관리)
- Mastering identity at every layer of the cake

SEC02-BP05 정기적으로 보안 인증 정보 감사 및 교체

보안 인증 정보를 주기적으로 감사하고 교체하여 리소스에 액세스하는 데 보안 인증 정보를 사용할 수 있는 기간을 제한합니다. 장기 보안 인증 정보는 많은 위험을 초래하며 이러한 위험은 장기 보안 인증 정보를 정기적으로 교체하여 줄일 수 있습니다.

원하는 결과: 장기 보안 인증 정보 사용과 관련된 위험을 줄이는 데 도움이 되는 보안 인증 정보 교체를 구현합니다. 보안 인증 정보 교체 정책 미준수를 정기적으로 감사하고 개선합니다.

일반적인 안티 패턴:

- 보안 인증 정보 사용을 감사하지 않습니다.
- 장기 보안 인증 정보를 불필요하게 사용합니다.
- 장기 보안 인증 정보를 사용하고 정기적으로 교체하지 않습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

임시 보안 인증 정보를 사용할 수 없으며 장기 보안 인증 정보가 필요한 경우 보안 인증 정보를 감사하여 정의된 제어(예: 다중 인증(MFA))가 적용되고 정기적으로 교체되며 적절한 액세스 수준을 보유하는지 확인합니다.

올바른 제어 기능이 적용되는지 확인하려면 주기적인 검증(가능한 자동화된 도구 사용)을 실시해야 합니다. 인적 자격 증명의 경우, 사용자가 주기적으로 암호를 변경하고 액세스 키 사용을 중지하며 그 대신 임시 보안 인증 정보를 사용하도록 규정해야 합니다. AWS Identity and Access Management(IAM) 사용자에서 중앙 집중식 자격 증명으로 이동할 때 보안 인증 정보 보고서를 생성하여 사용자를 감사할 수 있습니다.

또한 자격 증명 공급자에서 MFA를 적용하고 모니터링하는 것이 좋습니다. AWS Config 규칙를 설정하거나 AWS Security Hub 보안 표준을 사용하여 사용자가 MFA를 활성화했는지 모니터링할 수 있습니다. 시스템 자격 증명에 대한 임시 보안 인증 정보를 제공하려면 IAM Roles Anywhere를 사용하는 것이 좋습니다. IAM 역할 및 임시 보안 인증 정보를 사용할 수 없는 상황에서는 빈번한 감사 및 교체 액세스 키가 필요합니다.

구현 단계

- 정기적으로 보안 인증 정보 감사: 자격 증명 공급자 및 IAM에 구성된 보안 인증 정보를 감사하면 권한이 부여된 자격 증명만 워크로드에 액세스할 수 있는지 확인할 수 있습니다. 이러한 자격 증명에

는 IAM 사용자, AWS IAM Identity Center 사용자, Active Directory 사용자 또는 다른 업스트림 자격 증명 공급자의 사용자가 포함될 수 있지만 이에 국한되지 않습니다. 예를 들어 퇴사자의 계정을 제거하고 더 이상 필요하지 않은 크로스 계정 역할을 제거합니다. IAM 엔터티가 액세스하는 서비스에 대한 권한을 정기적으로 감사하는 프로세스가 있어야 합니다. 이렇게 하면 사용되지 않는 권한을 제거하기 위해 수정해야 하는 정책을 식별하는 데 도움이 됩니다. 보안 인증 정보 보고서 및 AWS Identity and Access Management Access Analyzer를 사용하여 IAM 자격 증명 및 권한을 감사합니다. Amazon CloudWatch를 사용하여 AWS 환경 내에서 호출된 특정 API 호출에 대한 경보를 설정할 수 있습니다. Amazon GuardDuty는 예상치 못한 활동에 대해 알릴 수도 있습니다. 이는 IAM 보안 인증 정보에 대한 지나치게 관대한 액세스 또는 의도치 않은 액세스를 나타낼 수 있습니다.

- 정기적으로 보안 인증 정보 교체: 임시 보안 인증 정보를 사용할 수 없는 경우 장기 IAM 액세스 키를 정기적으로 교체합니다(최대 90일마다). 자신도 모르게 액세스 키가 의도치 않게 공개된 경우 보안 인증 정보를 사용하여 리소스에 액세스할 수 있는 기간이 제한됩니다. IAM 사용자의 액세스 키 교체에 대한 자세한 내용은 액세스 키 교체를 참조하세요.
- IAM 권한 검토: AWS 계정의 보안을 강화하기 위해 각 IAM 정책을 정기적으로 검토하고 모니터링합니다. 정책이 최소 권한 원칙을 준수하는지 확인합니다.
- IAM 리소스 생성 및 업데이트 자동화 고려: IAM Identity Center는 역할 및 정책 관리와 같은 많은 IAM 작업을 자동화합니다. 또는 AWS CloudFormation을 사용하면 템플릿을 확인하고 버전을 제어할 수 있으므로, 역할 및 정책을 포함한 IAM 리소스 배포를 자동화하여 인적 오류가 발생할 가능성을 줄일 수 있습니다.
- IAM Roles Anywhere를 사용하여 시스템 자격 증명의 IAM 사용자 대체: IAM Roles Anywhere를 사용하면 온프레미스 서버와 같이 기존에는 사용할 수 없었던 영역에서 역할을 사용할 수 있습니다. IAM Roles Anywhere는 신뢰할 수 있는 X.509 인증서를 사용하여 AWS에 인증하고 임시 보안 인증 정보를 받습니다. IAM Roles Anywhere를 사용하면 장기 보안 인증 정보가 온프레미스 환경에 더 이상 저장되지 않으므로 이러한 보안 인증 정보를 교체할 필요가 없습니다. 만료가 가까워지면 X.509 인증서를 모니터링하고 교체해야 합니다.

리소스

관련 모범 사례:

- SEC02-BP02 임시 보안 인증 정보 사용
- SEC02-BP03 안전하게 보안 암호 저장 및 사용

관련 문서:

- AWS Secrets Manager 시작하기

- IAM 모범 사례
- 자격 증명 공급자 및 페더레이션
- 보안 파트너 솔루션: 액세스 및 액세스 제어
- 임시 보안 인증 정보
- AWS 계정의 보안 인증 정보 보고서 가져오기

관련 동영상:

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale(대규모 보안 암호 관리, 검색, 교체 모범 사례)
- Managing user permissions at scale with AWS IAM Identity Center(AWS IAM Identity Center를 사용하여 대규모로 사용자 권한 관리)
- Mastering identity at every layer of the cake(케이크의 모든 계층에서 자격 증명 마스터링)

관련 예시:

- Well-Architected Lab - Automated IAM User Cleanup(자동화된 IAM 사용자 정리)
- Well-Architected Lab - Automated Deployment of IAM Groups and Roles(IAM 그룹 및 역할의 자동 배포)

SEC02-BP06 사용자 그룹 및 속성 활용

관리하는 사용자 수가 늘어나면서 사용자를 체계적으로 정리하여 대규모로 관리할 방법을 결정해야 합니다. 공통 보안 요구 사항이 있는 사용자들을 자격 증명 공급자가 정의한 그룹에 배치하고, 액세스 제어에 사용할 수 있는 사용자 속성(예: 부서 또는 위치)이 정확하게 업데이트되었는지 확인하는 메커니즘을 적절히 설정합니다. 액세스를 제어할 때는 개별적인 사용자가 아니라 이러한 그룹과 속성을 사용합니다. 이를 통해 사용자의 액세스 권한을 변경해야 할 때 여러 개별 정책을 업데이트하는 대신 권한 세트를 사용해 사용자의 그룹 멤버십 또는 속성을 한 번 변경하여 중앙에서 액세스를 관리할 수 있습니다. AWS IAM Identity Center(IAM Identity Center)를 사용하여 사용자 그룹 및 속성을 관리할 수 있습니다. IAM Identity Center는 가장 보편적으로 사용되는 속성을 지원합니다. 사용자 생성 중에 수동으로 입력한 것이든, 동기화 엔진을 사용하여 자동으로 프로비저닝된 것(예: System for Cross-Domain Identity Management(SCIM) 사양에 정의된 대로)이든 마찬가지입니다.

공통 보안 요구 사항이 있는 사용자들을 자격 증명 공급자가 정의한 그룹에 배치하고, 액세스 제어에 사용할 수 있는 사용자 속성(예: 부서 또는 위치)이 정확하게 업데이트되었는지 확인하는 메커니즘을

적절히 설정합니다. 개별 사용자가 아닌 이러한 그룹 및 속성을 사용하여 액세스를 제어합니다. 이를 통해 사용자의 액세스 권한을 변경해야 할 때 여러 개별 정책을 업데이트하는 대신 사용자의 그룹 멤버십 또는 속성을 한 번 변경하여 중앙에서 액세스를 관리할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- AWS IAM Identity Center(IAM Identity Center)를 사용하는 경우 그룹 구성: IAM Identity Center는 사용자 그룹을 구성하고 그룹에 원하는 수준의 권한을 할당할 수 있는 기능을 제공합니다.
  - AWS Single Sign-On - 자격 증명 관리
- 속성 기반 액세스 제어(ABAC) 자세히 알아보기: ABAC는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다.
  - AWS용 ABAC란 무엇입니까?
  - 실습: EC2에 대한 IAM 태그 기반 액세스 제어

리소스

관련 문서:

- AWS Secrets Manager 시작하기
- IAM 모범 사례
- 자격 증명 공급자 및 연동
- AWS 계정 루트 사용자

관련 동영상:

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale(대규모 보안 암호 관리, 검색, 교체 모범 사례)
- Managing user permissions at scale with AWS IAM Identity Center(AWS SSO를 사용하여 대규모로 사용자 권한 관리)
- Mastering identity at every layer of the cake

관련 예시:

- 실습: EC2에 대한 IAM 태그 기반 액세스 제어

SEC 3 사람과 시스템에 대한 권한은 어떻게 관리합니까?

AWS 및 워크로드에 액세스해야 하는 사람 및 시스템 자격 증명에 대한 액세스를 제어하는 권한을 관리합니다. 권한은 누가 어떤 조건에서 무엇에 액세스할 수 있는지를 제어합니다.

모범 사례

- SEC03-BP01 액세스 요구 사항 정의
- SEC03-BP02 최소 권한 액세스 부여
- SEC03-BP03 긴급 액세스 프로세스 설정
- SEC03-BP04 지속적으로 권한 축소
- SEC03-BP05 조직에 대한 권한 가드레일 정의
- SEC03-BP06 수명 주기에 따라 액세스 관리
- SEC03-BP07 퍼블릭 및 크로스 계정 액세스 분석
- SEC03-BP08 안전하게 조직과 리소스 공유
- SEC03-BP09 안전하게 제3자와 리소스 공유

SEC03-BP01 액세스 요구 사항 정의

관리자, 최종 사용자 또는 기타 구성 요소는 워크로드의 각 구성 요소 또는 리소스에 액세스해야 합니다. 누가 혹은 무엇이 각 구성 요소에 액세스할 수 있는지 명확하게 정의한 다음 적절한 자격 증명 유형과 인증 및 권한 부여 방법을 선택해야 합니다.

일반적인 안티 패턴:

- 애플리케이션에 보안 암호를 하드 코딩 또는 저장합니다.
- 각 사용자에 대한 사용자 지정 권한을 부여합니다.
- 수명이 긴 보안 인증을 사용합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

관리자, 최종 사용자 또는 기타 구성 요소는 워크로드의 각 구성 요소 또는 리소스에 액세스해야 합니다. 누가 혹은 무엇이 각 구성 요소에 액세스할 수 있는지 명확하게 정의한 다음 적절한 자격 증명 유형과 인증 및 권한 부여 방법을 선택해야 합니다.

조직 내 AWS 계정에 대한 정기적인 액세스를 제공하려면 페더레이션 액세스 또는 중앙 집중식 자격 증명 공급자를 사용해야 합니다. 또한 자격 증명 관리를 중앙 집중화하고, 직원 액세스 수명 주기에 대한 AWS 액세스를 통합하기 위해 정립된 사례가 있는지 확인해야 합니다. 예를 들어, 직원이 다른 액세스 수준의 직무로 변경할 경우 해당 그룹 멤버십 또한 변경하여 새로운 액세스 요구 사항을 반영해야 합니다.

비 인적 자격 증명에 대한 액세스 요구 사항을 정의할 경우 어떤 애플리케이션 및 구성 요소가 액세스해야 하는지, 그리고 어떻게 권한이 부여되는지 결정합니다. 권장되는 접근 방식은 최소 권한 액세스 모델을 통해 구축된 IAM 역할을 사용하는 것입니다. AWS 관리형 정책은 대부분의 일반적인 사용 사례를 다루는 사전 정의된 IAM 정책을 제공합니다.

AWS 서비스, 즉 AWS Secrets Manager 및 AWS Systems Manager Parameter Store는IAM 역할 사용이 실현 불가능한 경우 애플리케이션 또는 워크로드로부터 보안 암호를 안전하게 분리할 수 있도록 지원합니다. Secrets Manager에서 보안 인증에 대한 자동 교체를 설정할 수 있습니다. Systems Manager를 사용하여 스크립트, 명령, SSM 문서, 구성 및 자동화 워크플로의 파라미터를 참조할 수 있으며 이 경우 파라미터를 생성할 때 지정한 고유한 이름을 사용합니다.

AWS Identity and Access Management Roles Anywhere를 사용하여 IAM 외부에서 실행되는 워크로드에 대한 AWS의 임시 보안 자격 증명을 얻을 수 있습니다. 워크로드에서 사용하는 IAM 정책 및 IAM 역할은 AWS 애플리케이션에서 AWS 리소스에 액세스하는 데 사용하는 것과 동일한 것일 수 있습니다.

가능한 경우, 장기적이고 정적인 보안 인증보다는 단기적이고 임시적인 보안 인증을 사용하는 것이 좋습니다. 프로그래밍 방식 액세스 및 장기 보안 인증을 사용하는 IAM 사용자가 필요한 경우 마지막으로 사용된 액세스 키 정보를 사용하여 액세스 키를 교체 및 제거합니다.

리소스

관련 문서:

- 속성 기반 액세스 제어(ABAC)
- AWS IAM Identity Center
- IAM Roles Anywhere
- IAM Identity Center의 AWS 관리형 정책
- AWS IAM 정책 조건
- IAM 사용 사례
- 불필요한 보안 인증 제거

- 정책 사용
- AWS 계정, OU 또는 조직을 기준으로 AWS 리소스에 대한 액세스를 제어하는 방법
- AWS Secrets Manager의 향상된 검색을 사용하여 보안 암호를 쉽게 식별, 정렬 및 관리

관련 동영상:

- 60분 이내에 IAM 정책 마스터하기
- 업무 분리, 최소 권한, 위임 및 CI/CD
- 혁신을 위한 자격 증명 및 액세스 관리 간소화

SEC03-BP02 최소 권한 액세스 부여

구체적인 조건에서 특정 리소스에 대해 일정한 작업을 수행하기 위해 자격 증명이 필요한 액세스 권한만 부여하는 것이 좋습니다. 개별 사용자에 대한 권한을 정의하는 대신, 그룹 및 자격 증명 속성을 사용하여 대규모로 권한을 동적으로 설정합니다. 예를 들어 한 개발자 그룹에 자체 프로젝트에 대한 리소스만 관리하도록 액세스 권한을 허용할 수 있습니다. 이렇게 하면 특정 개발자가 프로젝트에서 빠지게 될 경우 기본 액세스 정책을 변경하지 않고도 해당 개발자의 액세스 권한이 자동으로 해지됩니다.

원하는 결과: 사용자는 작업을 수행하는 데 필요한 권한만 가지고 있어야 합니다. 사용자는 제한된 시간 안에 특정 작업을 수행할 수 있는 프로덕션 환경에 액세스할 권한만 부여받아야 하며, 해당 작업이 완료된 후에는 액세스 권한이 해지되어야 합니다. 사용자가 다른 프로젝트에 착수하거나 직무를 옮기는 등 액세스 권한이 더 이상 필요하지 않으면 권한은 해지되어야 합니다. 관리자 권한은 신뢰할 수 있는 소수의 관리자 그룹에만 주어져야 합니다. 권한을 정기적으로 검토하여 권한이 잘못 부여되어 있는 상황이 없도록 해야 합니다. 시스템 또는 시스템 계정에는 작업을 완료하는 데 필요한 최소 권한 집합이 부여되어야 합니다.

일반적인 안티 패턴:

- 사용자에게 기본적으로 관리자 권한을 부여합니다.
- 일상적인 활동에 루트 사용자를 이용합니다.
- 전체 관리자 권한은 아니지만 과도하게 허용적인 정책을 생성합니다.
- 최소 권한 액세스를 허용하는지를 확인하기 위해 권한을 검토하지 않습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

최소 권한의 원칙은 자격 증명을 부여할 때 특정 작업을 완수하는 데 필요한 최소한의 활동만 수행하도록 부여해야 한다고 규정합니다. 이를 통해 사용 편의성, 효율성 및 보안의 균형을 이룰 수 있습니다. 이 원칙에 따라 운영하면 의도하지 않은 액세스를 제한하고 누가 어떤 리소스에 액세스했는지 추적하는 데 도움이 됩니다. IAM 사용자 및 역할에는 기본적으로 권한이 없습니다. 루트 사용자는 기본적으로 전체 액세스 권한을 가지므로, 루트 사용자는 루트 액세스가 필요한 작업에만 엄격하게 제어, 모니터링 및 사용되어야 합니다.

IAM 정책은 IAM 역할 또는 특정 리소스에 대한 권한을 명시적으로 부여하는 데 사용됩니다. 예를 들어, 자격 증명 기반 정책은 IAM 그룹에 연결하는 한편 S3 버킷은 리소스 기반 정책으로 제어할 수 있습니다.

IAM 정책을 생성할 때 AWS에서 액세스를 허용하거나 거부하려면 '참'이어야 하는 서비스 작업, 리소스 및 조건을 지정할 수 있습니다. AWS에서는 액세스 범위를 줄일 수 있도록 다양한 조건을 지원합니다. 예를 들어, `PrincipalOrgID` 조건 키를 사용하면 요청자가 AWS 조직에 속하지 않는 경우 요청자의 작업을 거부할 수 있습니다.

또한 `CalledVia` 조건 키를 사용하여 AWS Lambda 함수를 생성하는 AWS CloudFormation(와)과 같이, AWS 서비스가 사용자를 대신하여 수행하는 요청을 제어할 수 있습니다. 다양한 정책 유형을 계층화하여 심층 방어를 설정하고 사용자의 권한 전반을 제한해야 합니다. 나아가 어떤 조건에서 어떤 권한을 허용할지도 제한할 수도 있습니다. 예를 들어 애플리케이션 팀이 구축하는 시스템에 대해 자체 IAM 정책을 만들도록 허용하되, 권한 경계를 적용하여 시스템이 수신할 수 있는 최대 권한을 제한할 수 있습니다.

구현 단계

- 최소 권한 정책 구현: IAM 그룹 및 역할에 최소 권한 액세스 정책을 적용하여 사용자별로 정의한 역할 또는 기능을 반영합니다.
  - API 사용에 대한 기본 정책: AWS CloudTrail 로그를 검토하여 필요한 권한을 결정합니다. 이 검토를 통해 사용자가 AWS 내에서 실제로 수행하는 작업에 맞게 조정된 권한을 만들 수 있습니다. IAM Access Analyzer는 활동을 기반으로 IAM 정책을 자동으로 생성할 수 있습니다. 조직 또는 계정 수준에서 IAM Access Advisor를 사용하여 특정 정책에 대해 마지막으로 액세스한 정보를 추적할 수 있습니다.
- 직무 역할에 따른 AWS 관리형 정책 사용을 고려해 보세요. 세분화된 권한 정책을 어디서부터 만들어야 할지 알기 어려울 수 있습니다. AWS에서는 결제 관리자, 데이터베이스 관리자 및 데이터 사이언티스트와 같은 일반적인 작업 역할에 대한 관리형 정책을 관리합니다. 이러한 정책은 최소 권한 정책을 구현하는 방법을 결정하는 동시에 사용자의 액세스 범위를 좁히는 데 도움이 될 수 있습니다.

- 불필요한 권한 제거: 필요하지 않은 권한을 제거하고 과도하게 허용적인 정책을 축소합니다. IAM Access Analyzer 정책 생성을 사용하면 권한 정책을 세밀하게 조정할 수 있습니다.
- 사용자의 프로덕션 환경 액세스 제한: 사용자는 유효한 사용 사례가 있는 프로덕션 환경에만 액세스할 수 있어야 합니다. 사용자가 프로덕션 액세스 권한이 필요한 특정 작업을 수행한 후에는 액세스 권한을 해지해야 합니다. 프로덕션 환경에 대한 액세스를 제한하면 예기치 않게 프로덕션에 영향을 미치는 이벤트를 방지하고 의도하지 않은 액세스의 영향 범위를 줄일 수 있습니다.
- 권한 경계 고려: 권한 경계는 자격 증명 기반 정책을 통해 IAM 엔터티에 부여할 수 있는 최대 권한을 설정하는 관리형 정책을 사용하는 기능입니다. 엔터티의 권한 경계는 자격 증명 기반 정책과 권한 경계 모두에서 허용되는 작업만 수행하도록 허용합니다.
- 권한 리소스 태그 고려: 리소스 태그를 사용하는 속성 기반 액세스 제어 모델을 활용하면 리소스 목적, 소유자, 환경 또는 기타 기준에 따라 액세스 권한을 부여할 수 있습니다. 예를 들어, 리소스 태그를 사용하여 개발 환경과 프로덕션 환경을 구분할 수 있습니다. 이러한 태그를 사용하여 개발자의 권한을 개발 환경으로 제한할 수 있습니다. 태그 지정 정책과 권한 정책을 결합하면 모든 직무에 대해 복잡한 사용자 지정 정책을 정의할 필요 없이 세분화된 리소스 액세스 제어가 가능합니다.
- AWS Organizations 서비스 제어 정책을 사용해 보세요. 서비스 제어 정책은 조직의 멤버 계정에 대해 사용 가능한 최대 권한을 중앙에서 제어합니다. 중요한 점은 서비스 제어 정책을 사용하여 멤버 계정의 루트 사용자 권한을 제한할 수 있다는 사실입니다. AWS Organizations(을)를 보다 풍부하게 활용할 수 있도록 권장 관리 제어 기능을 제공하는 AWS Control Tower(을)를 사용하는 것도 고려해 보세요. Control Tower 내에서 자체 제어 기능을 정의할 수도 있습니다.
- 조직의 사용자 수명 주기 정책 설정: 사용자 수명 주기 정책은 사용자가 AWS에 온보드되어 있을 때, 작업 역할 또는 범위를 변경했을 때, 또는 AWS에 더 이상 액세스할 필요가 없을 때 각각 수행할 작업을 정의합니다. 사용자 수명 주기의 각 단계에서 권한 검토를 수행하여 권한이 적절하게 제한되는지 확인하고 권한이 잘못 부여되어 있는 상황이 없도록 해야 합니다.
- 권한을 검토하고 불필요한 권한을 제거하기 위한 정기 예약 설정: 사용자 액세스를 정기적으로 검토하여 사용자에게 과도하게 허용되는 액세스 권한이 없는지 확인해야 합니다. AWS Config 및 IAM Access Analyzer는 사용자 권한을 감사할 때 유용합니다.
- 작업 역할 매트릭스 설정: 작업 역할 매트릭스는 AWS 기반 내에서 필요한 여러 역할 및 액세스 수준을 시각화합니다. 작업 역할 매트릭스를 바탕으로 조직 내 사용자 책임에 따라 권한을 정의하고 분리할 수 있습니다. 개별 사용자 또는 역할에 직접 권한을 적용하는 대신 그룹을 사용하세요.

리소스

관련 문서:

- 최소 권한 부여

- IAM 엔터티의 권한 경계
- 최소 권한 IAM 정책 작성 기법
- IAM Access Analyzer를 통해 액세스 활동에 기반한 IAM 정책을 생성하여 보다 쉽게 최소 권한을 구현
- IAM 권한 경계를 사용하여 개발자에게 권한 관리 위임
- 마지막 액세스 정보를 사용하여 권한 수정
- IAM 정책 유형과 사용 시기
- IAM 정책 시뮬레이터로 IAM 정책 테스트
- AWS Control Tower의 가드레일
- 제로 트러스트 아키텍처: AWS의 철학
- CloudFormation StackSets로 최소 권한의 원칙을 구현하는 방법
- 속성 기반 액세스 제어(ABAC)
- 사용자 활동 보기를 통한 정책 범위 축소
- 역할 액세스 보기
- 태그를 지정하여 환경 구성 및 책임 추진
- AWS 태그 지정 전략
- AWS 리소스 태그 지정

관련 동영상:

- 차세대 권한 관리
- 제로 트러스트: AWS의 철학
- 권한 경계를 사용하여 사용자 및 역할을 제한하고 권한 에스컬레이션을 방지하려면 어떻게 해야 하나요?

관련 예시:

- 실습: 역할 생성을 위임하는 IAM 권한 경계
- 실습: EC2에 대한 IAM 태그 기반 액세스 제어

SEC03-BP03 긴급 액세스 프로세스 설정

가능성은 낮지만 자동화된 프로세스 또는 파이프라인에 문제가 발생할 때 워크로드에 긴급 액세스할 수 있도록 하는 프로세스입니다. 이 긴급 액세스 프로세스를 통해 최소 권한 액세스를 사용할 수 있습니다. 그러나 사용자가 적절한 수준의 액세스 권한이 필요할 때는 해당 권한을 얻을 수 있습니다. 예를 들어, 관리자가 액세스를 위한 긴급 AWS 크로스 계정 역할 등의 요청을 확인하고 승인하는 프로세스 또는 관리자가 긴급 요청을 확인하고 승인하기 위해 따라야 하는 특정 프로세스를 설정합니다.

일반적인 안티 패턴:

- 기존 자격 증명 구성을 사용하여 중단으로부터 복구할 수 있는 긴급 프로세스가 없습니다.
- 승격된 장기 권한을 문제 해결 또는 복구 목적으로 부여합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

긴급 액세스 설정은 여러 가지 형태일 수 있으며 이에 대비해야 합니다. 첫 번째는 기본 자격 증명 공급자의 실패입니다. 이 경우 복구에 필요한 권한을 통해 액세스의 두 번째 방법을 사용해야 합니다. 이 방법은 백업 자격 증명 공급자 또는 IAM 사용자일 수 있습니다. 이 두 번째 방법을 사용할 경우 엄격한 제어, 모니터링 및 알림 이 필요합니다. 긴급 액세스 자격 증명은 이 목적에 맞는 계정에서 생성되어야 하며 복구를 위해 특별히 설계된 역할을 수행할 수 있는 권한만 가지고 있어야 합니다.

또한 승격된 임시 관리 액세스가 필요할 경우에 대비하여 긴급 액세스를 준비해야 합니다. 일반적인 시나리오는 변경 사항을 배포하는 데 사용하는 자동 프로세스로 권한 변경을 제한하는 것입니다. 이 프로세스에 문제가 발생할 경우 사용자는 기능 복원을 위해 승격된 권한을 요청해야 할 수 있습니다. 이 경우, 사용자가 승격된 액세스를 요청할 수 있고 관리자가 이를 확인 및 승인할 수 있는 프로세스를 설정합니다. 구현 계획에는 사전 프로비저닝 액세스 및 긴급 설정, 브레이크 글라스, 역할이 SEC10-BP05 액세스 권한 사전 프로비저닝의 일부로서 제공되며 이에 대한 모범 사례 지침이 자세히 설명되어 있습니다.

리소스

관련 문서:

- AWS 모니터링 및 알림
- 임시 승격된 액세스 관리

관련 동영상:

- 60분 이내에 IAM 정책 마스터하기

SEC03-BP04 지속적으로 권한 축소

팀에서 필요한 액세스 권한을 결정할 때 불필요한 권한을 제거하고 최소 권한을 부여하기 위한 검토 프로세스를 수립합니다. 인적 액세스와 시스템 액세스 모두에 대해 사용되지 않는 자격 증명과 권한을 지속적으로 모니터링하고 제거합니다.

원하는 결과: 권한 정책은 최소 권한 원칙을 준수해야 합니다. 직무와 역할이 더 잘 정의됨에 따라 권한 정책을 검토하여 불필요한 권한을 제거해야 합니다. 이 접근 방식은 보안 인증 정보가 의도치 않게 노출되거나 권한 부여 없이 액세스되는 경우 영향 범위를 줄입니다.

일반적인 안티 패턴:

- 기본적으로 사용자에게 관리자 권한을 부여합니다.
- 지나치게 관대하지만 전체 관리자 권한이 없는 정책을 생성합니다.
- 더 이상 필요하지 않은 권한 정책을 유지합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

팀과 프로젝트가 이제 막 시작되었으므로 허용 권한 정책을 사용하여 혁신과 민첩성을 확보할 수 있습니다. 예를 들어 개발 또는 테스트 환경에서 개발자에게 광범위한 AWS 서비스에 대한 액세스 권한을 부여할 수 있습니다. 액세스 권한을 지속적으로 평가하고 현재 작업을 완료하는 데 필요한 서비스 및 서비스 작업으로만 액세스 권한을 제한하는 것이 좋습니다. 인적 자격 증명과 시스템 자격 증명 모두에 대해 이 평가가 권장됩니다. 시스템 또는 서비스 계정이라고도 하는 시스템 자격 증명은 AWS에 애플리케이션 또는 서버에 대한 액세스 권한을 부여하는 자격 증명입니다. 지나친 허용 권한은 광범위한 영향을 미치고 잠재적으로 고객 데이터를 노출시킬 수 있으므로 이 액세스 권한은 프로덕션 환경에서 특히 중요합니다.

AWS는 사용되지 않는 사용자, 역할, 권한 및 보안 인증 정보를 식별하는 데 도움이 되는 여러 방법을 제공합니다. AWS는 또한 연결된 액세스 키와 Amazon S3 버킷의 객체와 같은 AWS 리소스에 대한 액세스 권한을 포함하여 IAM 사용자 및 역할의 액세스 활동을 분석하는 데 도움이 될 수 있습니다. AWS Identity and Access Management Access Analyzer 정책 생성은 보안 주체가 상호 작용하는 실제 서비스 및 작업을 기반으로 제한적 권한 정책을 생성하는 데 도움이 될 수 있습니다. 속성 기반 액세스 제어(ABAC)는 권한 정책을 각 사용자에게 직접 연결하는 대신 속성을 사용하여 사용자에게 권한을 제공할 수 있으므로 권한 관리를 간소화하는 데 도움이 됩니다.

구현 단계

- AWS Identity and Access Management Access Analyzer 사용: IAM Access Analyzer는 조직 및 계정에서 Amazon Simple Storage Service(Amazon S3) 버킷 또는 IAM 역할과 같이 외부 엔터티와 공유되는 리소스를 식별하는 데 도움이 됩니다.
- IAM Access Analyzer 정책 생성 사용: IAM Access Analyzer 정책 생성은 IAM 사용자 또는 역할의 액세스 활동을 기반으로 세분화된 권한 정책을 생성하는 데 도움이 됩니다.
- IAM 사용자 및 역할에 대해 허용되는 기간 및 사용 정책 결정: 마지막으로 액세스한 타임스탬프를 사용하여 사용되지 않는 사용자 및 역할을 식별하고 제거합니다. 마지막으로 액세스한 서비스 및 작업 정보를 검토하여 특정 사용자 및 역할에 대한 권한을 식별하고 범위를 지정합니다. 예를 들어 마지막으로 액세스한 정보를 사용하면 애플리케이션 역할에 필요한 특정 Amazon S3 작업을 식별하여 그러한 작업으로만 역할의 액세스 권한을 제한할 수 있습니다. 마지막으로 액세스한 정보 기능은 AWS Management Console에서 프로그램 방식으로 제공되므로 인프라 워크플로 및 자동화된 도구에 손쉽게 통합할 수 있습니다.
- AWS CloudTrail에서 데이터 이벤트 로깅 고려: 기본적으로 CloudTrail은 Amazon S3 객체 수준 활동(예: `GetObject` 및 `DeleteObject`) 또는 Amazon DynamoDB 테이블 활동(예: `PutItem` 및 `DeleteItem`)과 같은 데이터 이벤트를 로깅하지 않습니다. 특정 Amazon S3 객체 또는 DynamoDB 테이블 항목에 액세스해야 하는 사용자 및 역할을 결정하려면 이러한 이벤트에 대한 로깅을 활성화하는 것이 좋습니다.

리소스

관련 문서:

- 최소 권한 부여
- 불필요한 보안 인증 정보 삭제
- AWS CloudTrail란 무엇인가요?
- 정책 사용
- DynamoDB 로깅 및 모니터링
- Amazon S3 버킷 및 객체에 대해 CloudTrail 이벤트 로깅 활성화
- AWS 계정의 보안 인증 정보 보고서 가져오기

관련 동영상:

- Become an IAM Policy Master in 60 Minutes or Less(60분 이내에 IAM 정책 마스터하기)

- Separation of Duties, Least Privilege, Delegation, and CI/CD(업무 분리, 최소 권한, 위임 및 CI/CD)
- AWS re:Inforce 2022 - AWS Identity and Access Management (IAM) deep dive(AWS Identity and Access Management(IAM) 심층 분석)

SEC03-BP05 조직에 대한 권한 가드레일 정의

조직의 모든 자격 증명에 대한 액세스를 제한하는 공통 제어를 설정합니다. 예를 들어, 특정 AWS 리전에 대한 액세스를 제한하거나 중앙 보안 팀에 사용되는 IAM 역할과 같은 공통 리소스를 운영자가 삭제하지 못하게 할 수 있습니다.

일반적인 안티 패턴:

- 조직의 관리자 계정에서 워크로드를 실행합니다.
- 프로덕션과 비 프로덕션 워크로드를 동일한 계정에서 실행합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

AWS에서 추가 워크로드를 확장하고 관리하면서, 계정을 사용하여 이러한 워크로드를 분리하고 AWS Organizations를 사용하여 해당 계정을 관리해야 합니다. 이 경우 조직 내 모든 자격 증명에 대한 액세스를 제한하는 공통 권한 가드레일을 설정하는 것이 좋습니다. 예를 들어, 특정 AWS 리전에 대한 액세스를 제한하거나 중앙 보안팀이 사용하는 IAM 역할과 같은 공통 리소스를 팀에서 삭제하지 못하게 할 수 있습니다.

사용자가 주요 서비스를 비활성화하지 못하도록 방지하는 등 서비스 제어 정책 예시를 구현하는 것부터 시작할 수 있습니다. SCP는 IAM 정책 언어를 사용하여 모든 IAM 보안 주체(사용자 및 역할)가 준수하는 제어를 설정할 수 있습니다. 특정 서비스 작업과 리소스에 대한 액세스를 제한하거나, 조직의 액세스 제어 요구 사항에 부합하도록 특정 조건을 기반으로 액세스를 제한할 수 있습니다. 필요한 경우, 가드레일에 예외를 정의할 수 있습니다. 예를 들어, 주어진 계정에서 특정 관리자 역할을 제외한 모든 IAM 엔터티에 대해 서비스 작업을 제한할 수 있습니다.

관리 계정에서 워크로드를 실행하지 않는 것이 좋습니다. 관리 계정은 멤버 계정에 영향을 미치는 보안 가드레일을 관리 및 배포하는 데 사용해야 합니다. 일부 AWS 서비스는 위임된 관리자 계정의 사용을 지원합니다. 가능한 경우, 이 위임된 계정을 관리 계정 대신 사용해야 합니다. 조직의 관리자 계정에 대한 액세스는 강력하게 제한해야 합니다.

다중 계정 전략을 사용하면 워크로드에 가드레일을 훨씬 더 유연하게 적용할 수 있습니다. AWS Security Reference Architecture는 계정 구조 설계 방법에 대한 권장 가이드를 제공합니다. AWS

Control Tower와 같은 AWS 서비스는 조직 전체에 대한 사전 예방 제어와 탐지 제어 모두를 중앙에서 관리할 수 있는 기능을 제공합니다. 조직 내 각 계정 또는 OU에 대한 명확한 목적을 정의하고 해당 목적에 따라 제어를 제한합니다.

리소스

관련 문서:

- AWS Organizations
- 서비스 제어 정책(SCP)
- 다중 계정 환경에서 서비스 제어 정책 최대한 활용
- AWS Security Reference Architecture(AWS SRA)

관련 동영상:

- 서비스 제어 정책을 사용한 예방적 가드레일 적용
- AWS Control Tower를 통해 대규모 거버넌스 구축
- AWS Identity and Access Management 심층 분석

SEC03-BP06 수명 주기에 따라 액세스 관리

액세스 제어를 운영자 및 애플리케이션 수명 주기/중앙 집중식 페더레이션 공급자와 통합합니다. 예를 들어 조직에서 나가거나 역할이 변경될 때 사용자의 액세스 권한을 제거합니다.

워크로드를 관리할 때 별도의 여러 계정을 사용하다 보면, 해당 계정 간에 리소스를 공유해야 하는 경우가 있습니다. 리소스를 공유할 때는 AWS Resource Access Manager(AWS RAM)를 사용하는 것이 좋습니다.. 이 서비스를 사용하면 AWS Organizations 및 조직 단위 내에서 AWS 리소스를 쉽고 안전하게 공유할 수 있습니다. AWS RAM을 사용하면 리소스를 공유하는 조직이나 조직 단위에서의 계정 포함 여부에 따라 공유 리소스에 대한 액세스 권한이 자동으로 부여되거나 취소됩니다. 이렇게 하면 리소스를 원하는 계정끼리만 공유하도록 할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

사용자 액세스 수명 주기 현재 사용자만 액세스할 수 있도록 시설의 신규 참여 사용자, 직무 기능 변경, 퇴거 사용자에 적용할 사용자 액세스 수명 주기 정책을 구현합니다.

리소스

관련 문서:

- ABAC(속성 기반 액세스 제어)
- 최소 권한 부여
- IAM Access Analyzer
- 불필요한 자격 증명 삭제
- 정책 사용

관련 동영상:

- 60분 이내에 IAM 정책 마스터하기
- 업무 분리, 최소 권한, 위임 및 CI/CD

SEC03-BP07 퍼블릭 및 크로스 계정 액세스 분석

퍼블릭 및 크로스 계정 액세스를 강조하는 조사 결과를 지속적으로 모니터링합니다. 이 액세스 권한이 필요한 특정 리소스에 대해서만 퍼블릭 액세스 및 크로스 계정 액세스를 줄입니다.

원하는 결과: AWS 리소스 중 어떤 리소스가 누구와 공유되는지 파악합니다. 공유 리소스를 지속적으로 모니터링하고 감사하여 권한이 부여된 보안 주체와만 공유되는지 확인합니다.

일반적인 안티 패턴:

- 공유 리소스의 인벤토리를 유지하지 않습니다.
- 크로스 계정 또는 리소스에 대한 퍼블릭 액세스를 승인하는 프로세스를 따르지 않습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 낮음

구현 가이드

계정이 AWS Organizations에 있는 경우 전체 조직, 특정 조직 단위 또는 개별 계정에 리소스에 대한 액세스 권한을 부여할 수 있습니다. 계정이 조직의 멤버가 아닌 경우 개별 계정과 리소스를 공유할 수 있습니다. 리소스 기반 정책(예: Amazon Simple Storage Service(Amazon S3) 버킷 정책)을 사용하거나 다른 계정의 보안 주체가 사용자 계정의 IAM 역할을 수임하도록 허용하여 직접 크로스 계정 액세스 권한을 부여할 수 있습니다. 리소스 정책을 사용할 때 권한이 부여된 보안 주체에게만 액세스 권한이 부

여되는지 확인합니다. 공개적으로 사용 가능해야 하는 모든 리소스를 승인하는 프로세스를 정의합니다.

AWS Identity and Access Management Access Analyzer는 증명 가능한 보안을 사용하여 계정 외부의 리소스에 대한 모든 액세스 경로를 식별합니다. 리소스 정책을 지속적으로 검토하고, 퍼블릭 또는 크로스 계정 액세스의 조사 결과를 보고하여 잠재적으로 광범위한 액세스를 간단하게 분석할 수 있습니다. 모든 계정에 대한 가시성을 확보했는지 확인하려면 AWS Organizations로 IAM Access Analyzer를 구성하는 것이 좋습니다. IAM Access Analyzer를 사용하면 리소스 권한을 배포하기 전에 결과를 미리 볼 수 있습니다. 따라서 정책 변경 사항이 리소스에 대해 의도한 퍼블릭 및 크로스 계정 액세스만 부여하는지 확인할 수 있습니다. 다중 계정 액세스를 위해 설계할 때 신뢰 정책을 사용하여 역할을 수임할 수 있는 경우를 제어할 수 있습니다. 예를 들어 `PrincipalOrgId` 조건 키를 사용하여 AWS Organizations 외부에서 역할을 수임하려는 시도를 거부할 수 있습니다.

AWS Config는 잘못 구성된 리소스를 보고할 수 있으며 AWS Config 정책 확인을 통해 퍼블릭 액세스가 구성된 리소스를 감지할 수 있습니다. AWS Control Tower 및 AWS Security Hub 같은 서비스는 공개적으로 노출된 리소스를 식별하고 개선하기 위해 AWS Organizations 전체에 탐지 제어 및 가드레일 배포를 간소화합니다. 예를 들어 AWS Control Tower에는 Amazon EBS 스냅샷이 AWS 계정에 의해 복원 가능한지 감지할 수 있는 관리형 가드레일이 있습니다.

구현 단계

- AWS Organizations에 대해 AWS Config 활성화 고려: AWS Config를 사용하면 AWS Organizations 내의 여러 계정에서 찾은 조사 결과를 위임된 관리자 계정으로 집계할 수 있습니다. 이는 포괄적인 보기를 제공하고 계정 전체에 AWS Config 규칙를 배포하여 공개적으로 액세스 가능한 리소스를 감지할 수 있습니다.
- AWS Identity and Access Management Access Analyzer 구성: IAM Access Analyzer는 조직 및 계정에서 Amazon S3 버킷 또는 IAM 역할과 같이 외부 엔터티와 공유되는 리소스를 식별하는 데 도움이 됩니다.
- AWS Config에서 자동 개선조치를 사용하여 Amazon S3 버킷의 퍼블릭 액세스 구성 변경에 대응: Amazon S3 버킷에 대한 퍼블릭 액세스 차단 설정을 자동으로 재활성화할 수 있습니다.
- 모니터링 및 알림을 구현하여 Amazon S3 버킷이 공개되었는지 확인: Amazon S3 퍼블릭 액세스 차단이 비활성화된 시점과 Amazon S3 버킷이 공개되었는지 확인하기 위해 모니터링 및 알림을 구현해야 합니다. 또한 AWS Organizations를 사용하는 경우 Amazon S3 퍼블릭 액세스 정책의 변경을 방지하는 서비스 제어 정책을 생성할 수 있습니다. AWS Trusted Advisor는 퍼블릭 액세스 권한이 있는 Amazon S3 버킷을 확인합니다. 모든 사용자에게 업로드 또는 삭제 액세스 권한을 부여하는 버킷 권한은 모든 사용자가 버킷 항목을 추가하거나, 수정하거나, 제거할 수 있도록 허용하여 잠재적 보안 문제가 발생하는 원인이 됩니다. Trusted Advisor 점검 항목은 명시적인 버킷 권한뿐 아니라 버킷 권한을 재정의할 수 있는 관련 버킷 정책도 확인합니다. 또한 AWS Config를 사용하여 퍼블

릭 액세스를 위해 Amazon S3 버킷을 모니터링할 수 있습니다. 자세한 내용은 퍼블릭 액세스를 허용하는 Amazon S3 버킷을 모니터링하고 대응하기 위해 AWS Config를 사용하는 방법을 참조하세요. 액세스 권한을 검토하는 동안 Amazon S3 버킷에 포함된 데이터 유형을 고려하는 것이 중요합니다. Amazon Macie은 개인 키 또는 AWS 키와 같은 보안 인증 정보, PII, PHI와 같은 민감한 데이터를 검색하고 보호하는 데 도움이 됩니다.

리소스

관련 문서:

- AWS Identity and Access Management Access Analyzer 사용
- AWS Control Tower 제어 라이브러리
- AWS 기초 보안 모범 사례 표준
- AWS Config 관리형 규칙
- AWS Trusted Advisor 점검 참조
- Amazon EventBridge를 사용하여 AWS Trusted Advisor 점검 결과 모니터링
- 조직의 모든 계정에서 AWS Config 규칙 관리
- AWS Config 및 AWS Organizations

관련 동영상:

- Best Practices for securing your multi-account environment(다중 계정 환경 보안 모범 사례)
- Dive Deep into IAM Access Analyzer(IAM Access Analyzer 심층 분석)

SEC03-BP08 안전하게 조직과 리소스 공유

워크로드 수가 증가함에 따라 해당 워크로드의 리소스에 대한 액세스 권한을 공유하거나 여러 계정에서 리소스를 여러 번 프로비저닝해야 할 수 있습니다. 개발, 테스트 및 프로덕션 환경과 같이 환경을 분류하는 구성이 있을 수 있습니다. 그러나 분리 구성이 있다고 해서 안전하게 공유하는 것이 제한되지는 않습니다. 겹치는 구성 요소를 공유하면 운영 오버헤드를 줄일 수 있고 동일한 리소스를 여러 번 생성하는 동안 누락된 부분을 추측하지 않고도 일관된 경험을 제공할 수 있습니다.

원하는 결과: 안전한 방법을 사용하여 조직 내에서 리소스를 공유하고 데이터 손실 방지 이니셔티브를 지원하여 의도치 않은 액세스를 최소화합니다. 개별 구성 요소를 관리하는 것에 비해 운영 오버헤드를 줄이고 동일한 구성 요소를 수동으로 여러 번 생성할 때 발생하는 오류를 줄이며 워크로드의 확장성을

높입니다. 다중 장애 지점 시나리오에서 해결 시간을 단축할 수 있고 구성 요소가 더 이상 필요하지 않은 시기를 결정할 때 신뢰성을 높일 수 있습니다. 외부에서 공유되는 리소스 분석에 대한 권장 가이드는 SEC03-BP07 퍼블릭 및 크로스 계정 액세스 분석를 참조하세요.

일반적인 안티 패턴:

- 예상치 못한 외부 공유를 지속적으로 모니터링하고 자동으로 알리는 프로세스가 부족합니다.
- 공유해야 할 것과 공유하지 말아야 할 것에 대한 기준이 부족합니다.
- 필요할 때 명시적으로 공유하는 대신 광범위한 공개 정책을 기본으로 설정합니다.
- 필요할 때 겹치는 기본 리소스를 수동으로 생성합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

액세스 제어 및 패턴을 설계하여 공유 리소스의 소비를 신뢰할 수 있는 엔터티로만 안전하게 관리합니다. 공유 리소스를 모니터링하고 공유 리소스 액세스를 지속적으로 검토하고 부적절하거나 예상치 못한 공유에 대한 알림을 받습니다. 퍼블릭 및 크로스 계정 액세스 분석을 검토하면 필요한 리소스에 대한 외부 액세스 권한을 줄이도록 거버넌스를 설정하고 지속적으로 모니터링하고 자동으로 알리는 프로세스를 설정하는 데 도움이 됩니다.

AWS Organizations 내 크로스 계정 공유는 AWS Security Hub, Amazon GuardDuty 및 AWS Backup과 같은 여러 AWS 서비스에서 지원됩니다. 이러한 서비스를 통해 데이터를 중앙 계정과 공유하거나, 중앙 계정에서 액세스하거나, 중앙 계정에서 리소스 및 데이터를 관리할 수 있습니다. 예를 들어 AWS Security Hub는 조사 결과를 개별 계정에서 모든 조사 결과를 볼 수 있는 중앙 계정으로 전송할 수 있습니다. AWS Backup은 리소스를 백업하고 계정 간에 공유할 수 있습니다. AWS Resource Access Manager(AWS RAM)를 사용하여 VPC 서브넷 및 Transit Gateway 첨부 파일, AWS Network Firewall, Amazon SageMaker 파이프라인과 같은 다른 공통 리소스를 공유할 수 있습니다.

조직 내에서만 리소스를 공유하도록 계정을 제한하려면 서비스 제어 정책(SCP)을 사용하여 외부 보안 주체에 대한 액세스 권한을 방지합니다. 리소스를 공유할 때 자격 증명 기반 제어와 네트워크 제어를 결합하여 조직의 데이터 경계를 생성하여 의도치 않은 액세스로부터 보호합니다. 데이터 경계는 신뢰할 수 있는 자격 증명만 예상 네트워크의 신뢰할 수 있는 리소스에 액세스하고 있는지 확인하는 데 도움이 되는 예방 가드레일입니다. 이러한 제어를 통해 어떤 리소스를 공유할 수 있는지에 대한 적절한 제한을 설정하고, 리소스의 허용되지 않은 공유 또는 노출을 방지할 수 있습니다. 예를 들어 데이터 경계의 일부로 VPC 엔드포인트 정책과 `AWS:PrincipalOrgId` 조건을 사용하여 Amazon S3 버킷에 액세스하는 자격 증명이 조직에 속하는지 확인할 수 있습니다. SCP는 서비스 연결 역할(LSR) 또는 AWS 서비스 보안 주체에 적용되지 않는다는 점에 유의해야 합니다.

Amazon S3를 사용하는 경우 Amazon S3 버킷에 대한 ACL을 비활성화하고 IAM 정책을 사용하여 액세스 제어를 정의합니다. Amazon CloudFront에서 Amazon S3 오리진에 대한 액세스 권한을 제한하려면 오리진 액세스 ID(OAI)에서 AWS Key Management Service를 통한 서버 측 암호화를 비롯한 추가 기능을 지원하는 오리진 액세스 제어(OAC)로 마이그레이션합니다.

경우에 따라 조직 외부에서 리소스 공유를 허용하거나 리소스에 대한 타사 액세스 권한을 부여할 수 있습니다. 외부에서 리소스를 공유하기 위한 권한 관리에 대한 권장 가이드는 권한 관리를 참조하세요.

구현 단계

1. AWS Organizations를 사용합니다.

   AWS Organizations는 여러 AWS 계정를 사용자가 생성하고 중앙에서 관리하는 조직으로 통합할 수 있는 계정 관리 서비스입니다. 계정을 조직 단위(OU)로 그룹화하고 각 OU에 서로 다른 정책을 연결하여 예산, 보안 및 규정 준수 요구 사항을 충족할 수 있습니다. 또한 AWS 인공 지능(AI) 및 기계 학습(ML) 서비스가 데이터를 수집 및 저장하는 방법을 제어하고 Organizations와 통합된 AWS 서비스의 다중 계정 관리를 사용할 수 있습니다.
2. AWS Organizations를 AWS 서비스와 통합합니다.

   조직의 멤버 계정에서 사용자를 대신하여 작업을 수행하도록 AWS서비스를 활성화하면 AWS Organizations는 각 멤버 계정에서 해당 서비스에 대한 IAM 서비스 연결 역할을 생성합니다. AWS Management Console, AWS API 또는 AWS CLI를 사용하여 신뢰할 수 있는 액세스를 관리해야 합니다. 신뢰할 수 있는 액세스 활성화에 대한 권장 가이드는 다른 AWS 서비스와 함께 AWS Organizations 사용 및 Organizations와 함께 사용할 수 있는 AWS 서비스를 참조하세요.
3. 데이터 경계를 설정합니다.

   AWS 경계는 일반적으로 AWS Organizations에서 관리하는 조직으로 표시됩니다. 온프레미스 네트워크 및 시스템과 함께 AWS 리소스에 액세스하는 것은 내 AWS의 경계로 간주되는 경우가 많습니다. 경계의 목표는 자격 증명을 신뢰할 수 있고 리소스를 신뢰할 수 있으며 네트워크가 예상되는 경우 액세스가 허용되는지 확인하는 것입니다.

   a. 경계를 정의하고 구현합니다.

      각 권한 부여 조건은 AWS에 경계 구축 백서의 경계 구현에 설명된 단계를 따르세요. 네트워크 계층 보호에 대한 권장 가이드는 네트워크 보호를 참조하세요.

   b. 지속적으로 모니터링하고 알립니다.

      AWS Identity and Access Management Access Analyzer는 조직 및 계정에서 외부 엔터티와 공유되는 리소스를 식별하는 데 도움이 됩니다. IAM Access Analyzer를 AWS Security Hub와 통합하여 IAM Access Analyzer에서 Security Hub로 리소스에 대한 조사 결과를 전송 및 집계하여 환

경의 보안 태세를 분석할 수 있습니다. 통합을 활성화하려면 각 계정의 각 리전에서 IAM Access Analyzer 및 Security Hub를 모두 활성화합니다. 또한 AWS Config 규칙을 사용하여 구성을 감사하고 AWS Chatbot을 AWS Security Hub와 함께 사용하여 적절한 당사자에게 알릴 수 있습니다. 그런 다음 AWS Systems Manager 자동화 문서를 사용하여 규정 미준수 리소스를 개선할 수 있습니다.

c. 외부에서 공유되는 리소스에 대한 지속적인 모니터링 및 알림에 대한 권장 가이드는 퍼블릭 및 크로스 계정 액세스 분석을 참조하세요.

4. AWS 서비스에서 리소스 공유를 사용하고 그에 따라 제한합니다.

   Amazon Machine Image(AMI) 및 AWS Resource Access Manager(AWS RAM)와 같은 많은 AWS 서비스를 통해 다른 계정과 리소스를 공유하거나 다른 계정의 리소스를 대상으로 지정할 수 있습니다. AMI를 공유할 신뢰할 수 있는 계정을 지정하도록 `ModifyImageAttribute` API를 제한합니다. 신뢰할 수 없는 자격 증명의 액세스를 방지하기 위해 AWS RAM을 사용할 때 `ram:RequestedAllowsExternalPrincipals` 조건을 지정하여 조직으로만 공유를 제한합니다. 권장 가이드 및 고려 사항은 리소스 공유 및 외부 대상을 참조하세요.

5. AWS RAM을 사용하여 계정에서 또는 다른 AWS 계정와 안전하게 공유합니다.

   AWS RAM은 사용자가 생성한 리소스를 계정의 역할 및 사용자와 다른 AWS 계정와 안전하게 공유하는 데 도움이 됩니다. 다중 계정 환경에서 AWS RAM을 사용하면 리소스를 한 번 생성하여 다른 계정과 공유할 수 있습니다. 이 접근 방식은 크로스 계정 액세스를 사용할 때는 받지 못하는 Amazon CloudWatch 및 AWS CloudTrail과의 통합을 통해 일관성, 가시성 및 감사 가능성을 제공하는 동시에 운영 오버헤드를 줄이는 데 도움이 됩니다.

   이전에 리소스 기반 정책을 사용하여 공유한 리소스가 있는 경우 `PromoteResourceShareCreatedFromPolicy` API 또는 이에 상응하는 것을 사용하여 리소스 공유를 전체 AWS RAM 리소스 공유로 승격할 수 있습니다.

   경우에 따라 리소스를 공유하기 위해 추가 단계를 수행해야 할 수도 있습니다. 예를 들어 암호화된 스냅샷을 공유하려면 AWS KMS 키를 공유해야 합니다.

리소스

관련 모범 사례:

- SEC03-BP07 퍼블릭 및 크로스 계정 액세스 분석
- SEC03-BP09 안전하게 제3자와 리소스 공유
- SEC05-BP01 네트워크 계층 생성

관련 문서:

- 버킷 소유자가 소유하지 않은 객체에 크로스 계정 권한 부여
- IAM을 통한 신뢰 정책 사용 방법
- AWS에 데이터 경계 구축
- AWS 리소스에 대한 타사 액세스 권한을 부여할 때 외부 자격 증명을 사용하는 방법
- AWS Organizations과 함께 사용할 수 있는 AWS 서비스
- AWS에서 데이터 경계 설정: 신뢰할 수 있는 자격 증명만 회사 데이터에 액세스할 수 있도록 허용

관련 동영상:

- Granular Access with AWS Resource Access Manager(AWS Resource Access Manager를 통한 세분화된 액세스)
- Securing your data perimeter with VPC endpoints(VPC 엔드포인트를 통한 데이터 경계 보호)
- Establishing a data perimeter on AWS(AWS에 데이터 경계 설정)

관련 도구:

- 데이터 경계 정책 예시

SEC03-BP09 안전하게 제3자와 리소스 공유

클라우드 환경의 보안은 조직에 국한되지 않습니다. 조직은 타사를 이용하여 데이터의 일부를 관리할 수 있습니다. 타사 관리형 시스템에 대한 권한 관리는 임시 보안 인증 정보로 최소 권한 원칙을 사용하여 적시 액세스 방식을 따라야 합니다. 타사와 긴밀히 협력하면 영향 범위와 의도치 않은 액세스의 위험을 함께 줄일 수 있습니다.

원하는 결과: 사용자와 연결된 장기 AWS Identity and Access Management(IAM) 보안 인증 정보, IAM 액세스 키 및 보안 암호 키는 보안 인증 정보가 유효하고 활성 상태라면 누구나 사용할 수 있습니다. IAM 역할 및 임시 보안 인증 정보를 사용하면 이러한 민감한 세부 정보의 관리 및 운영 오버헤드를 포함하여 장기 보안 인증 정보를 유지하기 위한 노력을 줄여 전반적인 보안 태세를 개선할 수 있습니다. IAM 신뢰 정책의 외부 자격 증명에 대해 UUID(Universally Unique Identifier)를 사용하고 IAM 역할에 연결된 IAM 정책을 제어하면 타사에 부여된 액세스 권한이 너무 많이 허용되지 않았는지 감사하고 확인할 수 있습니다. 외부에서 공유되는 리소스 분석에 대한 권장 가이드는 SEC03-BP07 퍼블릭 및 크로스 계정 액세스 분석를 참조하세요.

일반적인 안티 패턴:

- 조건 없이 기본 IAM 신뢰 정책을 사용합니다.
- 장기 IAM 보안 인증 정보 및 액세스 키를 사용합니다.
- 외부 ID를 재사용합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

AWS Organizations 외부에서 리소스 공유를 허용하거나 타사에 계정에 대한 액세스 권한을 부여할 수 있습니다. 예를 들어, 타사가 계정 내 리소스에 액세스해야 하는 모니터링 솔루션을 제공할 수 있습니다. 이 경우, 타사에게만 필요한 권한이 포함된 IAM 크로스 계정 역할을 생성합니다. 또한 외부 ID 조건을 사용하여 신뢰 정책을 정의합니다. 외부 ID를 사용하는 경우 사용자 또는 타사는 각 고객, 타사 또는 테넌시용으로 고유한 ID를 생성할 수 있습니다. 고유한 ID는 생성된 후에는 사용자 외에는 누구도 제어할 수 없습니다. 타사는 안전하고 감사 가능하며 재현 가능한 방식으로 외부 ID를 고객과 연결하는 프로세스를 구현해야 합니다.

또한 IAM Roles Anywhere를 사용하여 AWS API를 사용하는 AWS 외부의 애플리케이션에 대한 IAM 역할을 관리할 수 있습니다.

타사가 더 이상 환경에 액세스할 필요가 없으면 역할을 제거합니다. 장기 보안 인증 정보를 타사에 제공하지 않아야 합니다. 공유를 지원하는 다른 AWS 서비스를 계속 인지하고 있어야 합니다. 예를 들어 AWS Well-Architected Tool을 사용하면 다른 AWS 계정와 워크로드를 공유할 수 있고 AWS Resource Access Manager를 사용하면 소유한 AWS 리소스를 다른 계정과 안전하게 공유할 수 있습니다.

구현 단계

1. 크로스 계정 역할을 사용하여 외부 계정에 대한 액세스 권한을 제공합니다.

   크로스 계정 역할은 고객에게 서비스를 제공하기 위해 외부 계정 및 타사에서 저장하는 민감한 정보의 양을 줄입니다. 크로스 계정 역할을 사용하면 AWS Partner 또는 조직의 다른 계정과 같은 타사에 계정의 AWS 리소스에 대한 액세스 권한을 안전하게 부여하는 동시에 해당 액세스를 관리 및 감사하는 기능을 유지할 수 있습니다.

   타사는 하이브리드 인프라에서 서비스를 제공하거나 오프사이트 위치로 데이터를 가져올 수 있습니다. IAM Roles Anywhere는 타사 워크로드가 AWS 워크로드와 안전하게 상호 작용하고 장기 보안 인증 정보의 필요성을 줄이는 데 도움이 됩니다.

외부 계정 액세스를 제공하기 위해 장기 보안 인증 정보 또는 사용자와 연결된 액세스 키를 사용해서는 안 됩니다. 대신 크로스 계정 역할을 사용하여 크로스 계정 액세스를 제공합니다.

2. 타사의 외부 ID를 사용합니다.

   외부 ID를 사용하면 IAM 신뢰 정책에서 역할을 수임할 수 있는 사람을 지정할 수 있습니다. 신뢰 정책은 역할을 수임하는 사용자가 작동 중인 조건 및 대상을 어설션하도록 요구할 수 있습니다. 또한 계정 소유자가 특정 상황에서만 역할을 수임하도록 허용할 수도 있습니다. 외부 ID의 기본 기능은 혼동된 대리인(confused deputy) 문제를 해결하고 방지하는 것입니다.

   AWS 계정 소유자이고 사용자 외에 다른 AWS 계정에 액세스하는 타사의 역할을 구성했거나 다른 고객을 대신하여 역할을 수임하는 위치에 있는 경우 외부 ID를 사용합니다. 타사 또는 AWS Partner와 협력하여 IAM 신뢰 정책에 포함할 외부 ID 조건을 설정합니다.

3. 범용적으로 고유한 외부 ID를 사용합니다.

   UUID(Universally Unique Identifier)와 같은 외부 ID에 대해 임의의 고유한 값을 생성하는 프로세스를 구현합니다. 고객 A가 중복된 외부 ID와 함께 고객 B의 역할 ARN을 사용하여 고객 B의 데이터를 볼 수 있으므로, 서로 다른 고객 간에 외부 ID를 재사용하는 타사의 경우 혼동된 대리인(confused deputy) 문제가 해결되지 않습니다. 타사가 서로 다른 AWS 계정로 여러 고객을 지원하는 다중 테넌트 환경에서 타사는 각 AWS 계정에 대한 외부 ID로 서로 다른 고유한 ID를 사용해야 합니다. 타사는 중복된 외부 ID를 감지하고 각 고객을 해당 외부 ID에 안전하게 매핑하는 업무를 담당합니다. 타사는 외부 ID를 지정할 때만 역할을 수임할 수 있는지 확인하기 위해 테스트해야 합니다. 타사는 외부 ID가 필요하기 전에는 고객 역할 ARN 및 외부 ID를 저장하지 않아야 합니다.

   외부 ID는 보안 암호로 취급되지는 않지만 전화번호, 이름, 계정 ID와 같이 쉽게 추측할 수 있는 값이 아니어야 합니다. 설정을 가장할 목적으로 외부 ID를 변경할 수 없도록 외부 ID를 읽기 전용 필드로 만듭니다.

   사용자 또는 타사가 외부 ID를 생성할 수 있습니다. ID 생성 담당자를 결정하는 프로세스를 정의합니다. 외부 ID를 생성하는 엔터티에 관계없이 타사는 고객 간에 고유성과 형식을 일관되게 적용합니다.

4. 고객이 제공한 장기 보안 인증 정보를 지원 중단합니다.

   장기 보안 인증 정보 사용을 중단하고 크로스 계정 역할 또는 IAM Roles Anywhere를 사용합니다. 장기 보안 인증 정보를 사용해야 하는 경우 역할 기반 액세스로의 마이그레이션 계획을 수립합니다. 키 관리에 대한 자세한 내용은 자격 증명 관리를 참조하세요. 또한 AWS 계정 팀 및 타사와 협력하여 위험 완화 런북을 수립합니다. 보안 인시던트의 잠재적 영향에 대한 대응 및 완화에 대한 권장 가이드는 인시던트 대응을 참조하세요.

5. 설정에 권장 가이드가 있거나 자동화되어 있는지 확인합니다.

   계정의 크로스 계정 액세스를 위해 생성된 정책은 최소 권한 원칙을 따라야 합니다. 타사는 역할 정책 문서를 제공하거나 AWS CloudFormation 템플릿 또는 이에 상응하는 템플릿을 사용하는 자동화된 설정 메커니즘을 제공해야 합니다. 이는 수동 정책 생성과 관련된 오류가 발생할 가능성을 줄이고 감사 가능한 트레일을 제공합니다. AWS CloudFormation 템플릿을 사용하여 크로스 계정 역할을 생성하는 방법에 대한 자세한 내용은 크로스 계정 역할을 참조하세요.

   타사는 자동화되고 감사 가능한 설정 메커니즘을 제공해야 합니다. 그러나 필요한 액세스를 설명하는 역할 정책 문서를 사용하여 역할 설정을 자동화해야 합니다. AWS CloudFormation 템플릿 또는 이에 상응하는 템플릿을 사용하여 감사 관행의 일환으로 드리프트 감지를 사용하여 변경 사항을 모니터링해야 합니다.

6. 변경 사항을 설명합니다.

   계정 구조, 타사에 대한 요구 사항 또는 제공되는 서비스 오퍼링이 변경될 수 있습니다. 변경 사항과 장애를 예상하고 적절한 인력, 프로세스 및 기술을 사용하여 그에 따라 계획을 수립해야 합니다. 사용자가 제공하는 액세스 수준을 정기적으로 감사하고 감지 방법을 구현하여 예상치 못한 변경 사항을 알립니다. 외부 ID의 역할 및 데이터 스토어의 사용을 모니터링하고 감사합니다. 예상치 못한 변경 사항 또는 액세스 패턴의 결과로 일시적으로 또는 영구적으로 타사 액세스를 취소할 준비가 되어 있어야 합니다. 또한 수행하는 데 걸리는 시간, 관련된 담당자, 비용, 다른 리소스에 미치는 영향을 포함하여 취소 작업에 미치는 영향을 측정합니다.

   감지 방법에 대한 권장 가이드는 감지 모범 사례를 참조하세요.

리소스

관련 모범 사례:

- SEC02-BP02 임시 보안 인증 정보 사용
- SEC03-BP05 조직에 대한 권한 가드레일 정의
- SEC03-BP06 수명 주기에 따라 액세스 관리
- SEC03-BP07 퍼블릭 및 크로스 계정 액세스 분석
- SEC04 감지

관련 문서:

- 버킷 소유자가 소유하지 않은 객체에 크로스 계정 권한 부여

- IAM 역할을 통한 신뢰 정책 사용 방법
- IAM 역할을 사용하여 AWS 계정 간에 액세스 위임
- IAM을 사용하여 다른 AWS 계정의 리소스에 액세스하려면 어떻게 해야 하나요?
- IAM의 보안 모범 사례
- 크로스 계정 정책 평가 로직
- AWS 리소스에 액세스 권한을 타사에 부여할 때 외부 ID를 사용하는 방법
- 사용자 지정 리소스를 사용하여 외부 계정에서 생성된 AWS CloudFormation 리소스에서 정보 수집
- 다른 사용자가 소유한 AWS 계정에 액세스하기 위한 외부 ID를 안전하게 사용
- IAM Roles Anywhere를 사용하여 IAM 외부의 워크로드로 IAM 역할 확장

관련 동영상:

- How do I allow users or roles in a separate AWS 계정 access to my AWS 계정?(내 AWS 계정에 대한 별도의 AWS 계정 액세스에서 사용자 또는 역할을 허용하려면 어떻게 해야 하나요?)
- AWS re:Invent 2018: Become an IAM Policy Master in 60 Minutes or Less(AWS re:Invent 2018: 60분 이내에 IAM 정책 마스터하기)
- AWS Knowledge Center Live: IAM Best Practices and Design Decisions(AWS 지식 센터 라이브: IAM 모범 사례 및 설계 결정)

관련 예시:

- Well-Architected Lab - Lambda cross account IAM role assumption (Level 300)(Lambda 크로스 계정 IAM 역할 수임(레벨 300))
- Configure cross-account access to Amazon DynamoDB(Amazon DynamoDB에 대한 크로스 계정 액세스 구성)
- AWS STS Network Query Tool(AWS STS 네트워크 쿼리 도구)

탐지

질문

- SEC 4 보안 관련 이벤트를 어떻게 감지하나요?

SEC 4 보안 관련 이벤트를 어떻게 감지하나요?

이벤트는 로그와 지표에서 캡처하고 분석하는 방식으로 파악할 수 있습니다. 보안 이벤트 및 잠재적 위협에 대한 조치를 취하면 워크로드를 보호할 수 있습니다.

모범 사례

- SEC04-BP01 서비스 및 애플리케이션 로깅 구성
- SEC04-BP02 로그, 결과 및 지표를 중앙에서 분석
- SEC04-BP03 이벤트 대응 자동화
- SEC04-BP04 조치 가능한 보안 이벤트 구현

SEC04-BP01 서비스 및 애플리케이션 로깅 구성

서비스 및 애플리케이션의 보안 이벤트 로그를 유지합니다. 이는 감사, 조사 및 운영 사용 사례에 대한 보안의 기본 원칙이며 거버넌스, 위험 및 규정 준수(GRC) 표준, 정책 및 절차를 기반으로 하는 공통 보안 요구 사항입니다.

원하는 결과: 조직은 AWS 서비스 및 애플리케이션에서 보안 인시던트 대응과 같은 내부 프로세스 또는 의무를 이행해야 할 때 적시에 안정적이고 일관되게 보안 이벤트 로그를 검색할 수 있어야 합니다. 더 나은 운영 결과를 위해 로그를 중앙 집중화하는 것이 좋습니다.

일반적인 안티 패턴:

- 로그는 영구적으로 저장되거나 너무 빨리 삭제됩니다.
- 누구나 로그에 액세스할 수 있습니다.
- 로그 거버넌스 및 사용을 위해 수동 프로세스에 전적으로 의존합니다.
- 필요한 경우를 대비하여 모든 유형의 로그를 저장합니다.
- 필요한 경우에만 로그 무결성을 확인합니다.

이 모범 사례 확립의 이점: 보안 인시던트에 대한 근본 원인 분석(RCA) 메커니즘과 거버넌스, 위험 및 규정 준수 의무에 대한 증거 소스를 구현합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

요구 사항에 따른 보안 조사 또는 기타 사용 사례 중에 관련 로그를 검토하여 인시던트의 전체 범위와 타임라인을 기록하고 이해할 수 있어야 합니다. 관심 있는 특정 작업이 발생했음을 나타내는 알림 생성

에도 로그가 필요합니다. 쿼리 및 검색 메커니즘과 알림을 선택, 활성화, 저장 및 설정하는 것이 중요합니다.

구현 단계

- 로그 소스를 선택하고 활성화합니다. 보안 조사에 앞서 관련 로그를 캡처하여 AWS 계정의 활동을 소급하여 재구성해야 합니다. 워크로드와 관련된 로그 소스를 선택하고 활성화합니다.

  로그 소스 선택 기준은 비즈니스에 필요한 사용 사례를 기반으로 해야 합니다. AWS CloudTrail 또는 AWS Organizations 트레일을 사용하여 각 AWS 계정에 대한 트레일을 설정하고 이에 대한 Amazon S3 버킷을 구성합니다.

  AWS CloudTrail은 AWS 서비스 활동을 캡처하는 AWS 계정에 대해 수행된 API 호출을 추적하는 로깅 서비스입니다. AWS Management Console, AWS CLI 또는 AWS SDK를 사용하여 CloudTrail 이벤트 기록을 통해 검색할 수 있도록 기본적으로 관리 이벤트의 90일 보존으로 활성화됩니다. 데이터 이벤트를 더 오래 보존하고 가시성을 확보하려면 CloudTrail 트레일을 생성하고 이를 Amazon S3 버킷과 연결하고 선택적으로 Amazon CloudWatch 로그 그룹과 연결합니다. 또는 최대 7년 동안 CloudTrail 로그를 유지하고 SQL 기반 쿼리 기능을 제공하는 CloudTrail Lake를 생성할 수 있습니다.

  AWS는 VPC를 사용하는 고객이 각각 VPC 흐름 로그 및 Amazon Route 53 확인자 쿼리 로그를 사용하여 네트워크 트래픽 및 DNS 로그를 활성화하고 Amazon S3 버킷 또는 CloudWatch 로그 그룹으로 스트리밍할 것을 권장합니다. VPC, 서브넷 또는 네트워크 인터페이스에 대한 VPC 흐름 로그를 생성할 수 있습니다. VPC 흐름 로그의 경우 흐름 로그를 사용하는 방법과 위치를 선택하여 비용을 절감할 수 있습니다.

  AWS CloudTrail 로그, VPC 흐름 로그 및 Route 53 확인자 쿼리 로그는 AWS에서 보안 조사를 지원하는 기본 로깅 소스입니다. 또한 Amazon Security Lake를 사용하여 쿼리할 준비가 된 Apache Parquet 형식 및 OCSF(Open Cybersecurity Schema Framework)로 이 로그 데이터를 수집, 정규화 및 저장할 수 있습니다. Security Lake는 다른 AWS 로그와 타사 소스의 로그도 지원합니다.

  AWS 서비스는 Elastic Load Balancing 로그, AWS WAF 로그, AWS Config 레코더 로그, Amazon GuardDuty 조사 결과, Amazon Elastic Kubernetes Service(Amazon EKS) 감사 로그, Amazon EC2 인스턴스 운영 체제 및 애플리케이션 로그와 같은 기본 로그 소스에서 캡처하지 않은 로그를 생성할 수 있습니다. 로깅 및 모니터링 옵션의 전체 목록은 AWS 보안 인시던트 대응 가이드의 부록 A: 클라우드 기능 정의 - 로깅 및 이벤트를 참조하세요.
- 각 AWS 서비스 및 애플리케이션에 대한 로깅 기능 연구: 각 AWS 서비스 및 애플리케이션은 각각 고유한 보존 및 수명 주기 기능이 있는 로그 스토리지 옵션을 제공합니다. 가장 일반적인 두 가지 로그 스토리지 서비스는 Amazon Simple Storage Service(Amazon S3) 및 Amazon CloudWatch입니다. 보존 기간이 긴 경우 비용 효율성과 유연한 수명 주기 기능을 위해 Amazon S3를 사용하는 것이 좋

습니다. 기본 로깅 옵션이 Amazon CloudWatch 로그인 경우 액세스 빈도가 낮은 로그를 Amazon S3에 아카이브하는 것이 좋습니다.

- 로그 스토리지 선택: 로그 스토리지 선택은 일반적으로 사용하는 쿼리 도구, 보존 기능, 친숙도, 비용과 관련이 있습니다. 로그 스토리지의 기본 옵션은 Amazon S3 버킷 또는 CloudWatch 로그 그룹입니다.

  Amazon S3 버킷은 선택적 수명 주기 정책을 통해 비용 효율적이고 내구성이 뛰어난 스토리지를 제공합니다. Amazon S3 버킷에 저장된 로그는 Amazon Athena와 같은 서비스를 사용하여 쿼리할 수 있습니다.

  CloudWatch 로그 그룹은 CloudWatch Logs Insights를 통해 내구성이 뛰어난 스토리지와 기본 제공 쿼리 기능을 제공합니다.

- 적절한 로그 보존 파악: Amazon S3 버킷 또는 CloudWatch 로그 그룹을 사용하여 로그를 저장하는 경우 각 로그 소스에 적절한 수명 주기를 설정하여 저장 및 검색 비용을 최적화해야 합니다. 고객은 일반적으로 3개월에서 1년 사이의 로그를 쉽게 쿼리할 수 있으며 최대 7년 동안 보존할 수 있습니다. 가용성 및 보존에 대한 선택은 보안 요구 사항과 법적, 규제 및 비즈니스 의무의 조합과 일치해야 합니다.
- 적절한 보존 및 수명 주기 정책으로 각 AWS 서비스 및 애플리케이션에 대한 로깅 활성화: 조직의 각 AWS 서비스 또는 애플리케이션에 대해 특정 로깅 구성 지침을 찾습니다.
  - AWS CloudTrail 트레일 구성
  - VPC Flow Logs 구성
  - Amazon GuardDuty 조사 결과 내보내기 구성
  - AWS Config 기록 구성
  - AWS WAF 웹 ACL 트래픽 구성
  - AWS Network Firewall 네트워크 트래픽 로그 구성
  - Elastic Load Balancing 액세스 로그 구성
  - Amazon Route 53 확인자 쿼리 로그 구성
  - Amazon RDS 로그 구성
  - Amazon EKS 컨트롤 플레인 로그 구성
  - Amazon EC2 인스턴스 및 온프레미스 서버에 대한 Amazon CloudWatch 에이전트 구성
- 로그에 대한 쿼리 메커니즘 선택 및 구현: 로그 쿼리의 경우 CloudWatch 로그 그룹에 저장된 데이터에는 CloudWatch Logs Insight를 사용할 수 있고 Amazon S3에 저장된 데이터에는 Amazon Athena 및 Amazon OpenSearch Service를 사용할 수 있습니다. 보안 정보 및 이벤트 관리(SIEM) 서비스와 같은 타사 쿼리 도구를 사용할 수도 있습니다.

로그 쿼리 도구를 선택하는 프로세스는 보안 작업의 인력, 프로세스 및 기술 측면을 고려해야 합니다. 운영, 비즈니스 및 보안 요구 사항을 충족하고 장기적으로 액세스 및 유지 관리 가능한 도구를 선택합니다. 로그 쿼리 도구는 스캔할 로그 수가 도구의 한도 내에서 유지될 때 최적으로 작동합니다. 비용이나 기술적 제약으로 인해 여러 쿼리 도구를 사용하는 것이 일반적입니다.

예를 들어 타사 보안 정보 및 이벤트 관리(SIEM) 도구를 사용하여 지난 90일 데이터에 대한 쿼리를 수행할 수 있지만, SIEM의 로그 수집 비용으로 인해 90일 이후 데이터에 대한 쿼리를 수행할 때는 Athena를 사용합니다. 구현에 관계없이, 특히 보안 이벤트 조사 중에 운영 효율성을 극대화하는 데 필요한 도구의 수를 최소화하는 접근 방식인지 확인합니다.

- 알림에 로그 사용: AWS는 여러 보안 서비스를 통해 알림을 제공합니다.
  - AWS Config는 AWS 리소스 구성을 모니터링 및 기록하며, 원하는 구성을 기준으로 평가 및 개선 조치를 자동화할 수 있습니다.
  - Amazon GuardDuty는 AWS 계정 및 워크로드를 보호하기 위해 악의적인 활동 및 무단 동작을 지속적으로 모니터링하는 위협 탐지 서비스입니다. GuardDuty는 AWS CloudTrail 관리 및 데이터 이벤트, DNS 로그, VPC 흐름 로그 및 Amazon EKS 감사 로그와 같은 소스에서 정보를 수집, 집계 및 분석합니다. GuardDuty는 CloudTrail, VPC 흐름 로그, DNS 쿼리 로그 및 Amazon EKS에서 직접 독립적인 데이터 스트림을 가져옵니다. Amazon S3 버킷 정책을 관리하거나 로그를 수집하고 저장하는 방식을 수정할 필요가 없습니다. 자체 조사 및 규정 준수 목적으로 이러한 로그를 보관하는 것이 좋습니다.
  - AWS Security Hub는 여러 AWS 서비스 및 타사 제품(선택 사항)의 보안 알림 또는 조사 결과를 집계하고 정리하고 우선순위를 지정함으로써 보안 알림 및 규정 준수 상태를 종합적으로 파악할 수 있는 단일 장소를 제공합니다.

  이러한 서비스에서 다루지 않는 보안 알림 또는 환경과 관련된 특정 알림에 대해 사용자 지정 알림 생성 엔진을 사용할 수도 있습니다. 이러한 알림 및 감지 구축에 대한 자세한 내용은 AWS 보안 인시던트 대응 탐지 가이드를 참조하세요.

리소스

관련 모범 사례:

- SEC04-BP02 로그, 결과 및 지표를 중앙에서 분석
- SEC07-BP04 데이터 수명 주기 관리 정의
- SEC10-BP06 도구 사전 배포

관련 문서:

- AWS 보안 인시던트 대응 가이드
- Amazon Security Lake 시작하기
- 시작하기: Amazon CloudWatch Logs
- 보안 파트너 솔루션: 로깅 및 모니터링

관련 동영상:

- AWS re:Invent 2022 - Introducing Amazon Security Lake(AWS re:Invent 2022 - Amazon Security Lake 소개)

관련 예시:

- Assisted Log Enabler for AWS(AWS의 지원 로그 인에이블러)
- AWS Security Hub Findings Historical Export(AWS Security Hub 조사 결과 기록 내보내기)

관련 도구:

- Snowflake 사이버 보안

SEC04-BP02 로그, 결과 및 지표를 중앙에서 분석

보안 운영팀은 로그를 수집하고 검색 도구를 사용하여 발생 가능한 관심 이벤트(무단 활동 또는 의도하지 않은 변경을 나타낼 수 있음)를 검색할 수 있습니다. 하지만 수집된 데이터를 분석하고 정보를 수동으로 처리하는 것만으로는 복잡한 아키텍처에서 유입되는 대량의 정보를 파악하기가 어렵습니다. 분석 및 보고만 수행하면 이벤트를 처리하는 데 적합한 리소스를 제때 원활하게 할당할 수 없습니다.

완성된 보안 운영팀을 구축하기 위한 모범 사례는 보안 이벤트 흐름 및 이벤트에서 확인된 정보를 알림 및 워크플로 시스템(예: 티켓팅 시스템, 버그 또는 문제 시스템 또는 기타 보안 정보 및 이벤트 관리(SIEM) 시스템)에 심층적으로 통합하는 것입니다. 이렇게 하면 이메일 및 정적 보고서가 아닌 효율적 방식으로 워크플로를 파악할 수 있으며 이벤트나 이벤트를 통해 확인된 정보를 라우팅, 에스컬레이션 및 관리할 수 있습니다. 대부분의 조직은 보안 알림도 채팅 또는 협업 및 개발자 생산성 플랫폼에 통합하고 있습니다. 자동화에 착수하는 조직의 경우, API 기반의 지연 시간이 짧은 티켓팅 시스템은 먼저 자동화할 대상을 계획할 때 상당한 유연성을 제공합니다.

이러한 모범 사례는 사용자 활동이나 네트워크 이벤트를 보여 주는 로그 메시지에서 생성된 보안 이벤트뿐 아니라 인프라 자체에서 감지된 변경 사항에도 적용됩니다. 어느 정도의 변화를 받아들일 것인지에 대한 기준이 명확하지 않기 때문에 AWS Identity and Access Management(IAM) 및 AWS Organizations 구성의 조합을 통해 변경 사항이 실행되는 것을 방지할 수 없는 경우에는 변경을 감지하고 변경 사항이 적절한지 판단한 다음 해당 정보를 올바른 수정 워크플로로 라우팅하는 능력이 보안 아키텍처를 유지 관리하고 검증하는 데 필수적입니다.

Amazon GuardDuty 및 AWS Security Hub는 다른 AWS 서비스를 통해서도 사용할 수 있는 로그 레코드에 대한 집계, 중복 제거, 분석 메커니즘을 제공합니다. GuardDuty는 AWS CloudTrail 관리 및 데이터 이벤트, VPC DNS 로그, VPC 흐름 로그와 같은 소스에서의 정보를 수집, 집계, 분석합니다. Security Hub는 GuardDuty, AWS Config, Amazon Inspector, Amazon Macie, AWS Firewall Manager 및 AWS Marketplace에서 사용할 수 있는 수많은 서드 파티 보안 제품에서의 출력은 물론 적절히 구축하는 경우 사용자 자체 코드에서의 출력을 수집, 집계, 분석할 수 있습니다. GuardDuty 및 Security Hub 모두 여러 계정의 결과와 분석 정보를 집계할 수 있는 관리자-멤버 모델을 보유하고 있으며, Security Hub는 온프레미스 SIEM을 AWS 측 로그 및 알림 프리프로세서와 어그리게이터로 사용하는 고객들이 주로 사용합니다. 이러한 고객들은 AWS Lambda 기반 프로세서와 전달자를 통해 Amazon EventBridge에 수집할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 로그 처리 기능 평가: 로그 처리에 사용할 수 있는 옵션을 평가합니다.
  - Amazon OpenSearch Service를 사용하여 거의 모든 것을 로깅 및 모니터링하기
  - 로깅 및 모니터링 솔루션 전문 파트너 찾기
- CloudTrail 로그 분석을 시작할 때 Amazon Athena를 테스트합니다.
  - Athena를 구성하여 CloudTrail 로그 분석
- AWS에서 중앙 집중식 로깅 구현: 다음 AWS 예시 솔루션을 통해 여러 소스에서 로깅을 중앙 집중화하는 방법을 알아보십시오.
  - 로깅 솔루션 중앙 집중화
- 파트너와 중앙 집중식 로깅 구현: APN 파트너는 중앙에서 로그를 분석하는 데 도움이 되는 솔루션을 보유하고 있습니다.
  - 로깅 및 모니터링

리소스

관련 문서:

- AWS Answers: 중앙 집중식 로깅
- AWS Security Hub
- Amazon CloudWatch
- Amazon EventBridge
- 시작하기: Amazon CloudWatch Logs
- 보안 파트너 솔루션: 로깅 및 모니터링

관련 동영상:

- 리소스 구성 및 규정 준수를 중앙에서 모니터링
- Amazon GuardDuty 및 AWS Security Hub 결과 수정
- 클라우드에서 위협 관리: Amazon GuardDuty 및 AWS Security Hub

SEC04-BP03 이벤트 대응 자동화

자동화 기능을 사용하여 이벤트를 조사하고 해결하면 수작업 부담과 인적 오류가 줄어들고 조사 역량을 확대할 수 있습니다. 정기적인 검토는 자동화 도구를 튜닝하고 지속적으로 반복하는 데 도움이 됩니다.

AWS에서는 Amazon EventBridge를 사용하여 관심 있는 이벤트와 자동화된 워크플로에 대해 예기치 않은 잠재적 변경 사항에 대한 정보를 조사할 수 있습니다. 이 서비스는 AWS CloudTrail 이벤트 등의 기본 AWS 이벤트 형식과 애플리케이션에서 생성 가능한 사용자 지정 이벤트를 중개하도록 설계된 확장 가능 규칙 엔진을 제공합니다. 또한 Amazon GuardDuty를 사용하면 인시던트 대응 시스템(AWS Step Functions)을 구축하는 워크플로 시스템 또는 중앙 보안 계정에 이벤트를 라우팅하거나, 추가 분석을 위해 버킷에 이벤트를 라우팅할 수 있습니다.

AWS Config 규칙 및 규정 준수 팩을사용하여 변경 사항을 감지하고 이 정보를 올바른 워크플로로 라우팅할 수도 있습니다. AWS Config는 범위 내 서비스의 변경 사항을 감지(EventBridge보다 지연 시간이 김)한 다음 롤백/규정 준수 정책 적용/변경 관리 플랫폼 및 운영 티켓팅 시스템 등으로 정보 전달을 위해, AWS Config 규칙 규칙을 사용하여 구문 분석할 수 있는 이벤트를 생성합니다. 자체 Lambda 함수를 작성하여 AWS Config 이벤트에 응답하는 것은 물론 AWS Config 규칙 개발 키트 및 오픈 소스 AWS Config 규칙도 활용할 수 있습니다. 규정 준수 팩은 YAML 템플릿으로 작성된 단일 엔터티로 배포하는 AWS Config 규칙 및 해결 조치의 모음입니다. 샘플 규정 준수 팩 템플릿은 AWS Well-Architected 보안 원칙에서 사용 가능합니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- GuardDuty로 자동화된 알림 구현: GuardDuty는 악성 활동 및 무단 행위를 지속적으로 모니터링하여 AWS 계정 및 워크로드를 보호하는 위협 탐지 서비스입니다. GuardDuty를 활성화하고 자동 알림을 구성합니다.
- 조사 프로세스 자동화: 이벤트를 조사하여 관리자가 시간을 절약할 수 있도록 정보를 보고하는 자동화된 프로세스를 개발합니다.
  - 실습: Amazon GuardDuty 체험

리소스

관련 문서:

- AWS Answers: 중앙 집중식 로깅
- AWS Security Hub
- Amazon CloudWatch
- Amazon EventBridge
- 시작하기: Amazon CloudWatch Logs
- 보안 파트너 솔루션: 로깅 및 모니터링
- Amazon GuardDuty 설정

관련 동영상:

- 리소스 구성 및 규정 준수를 중앙에서 모니터링
- Amazon GuardDuty 및 AWS Security Hub 결과 수정
- 클라우드에서 위협 관리: Amazon GuardDuty 및 AWS Security Hub

관련 예시:

- 실습: 탐지 제어 자동 배포

SEC04-BP04 조치 가능한 보안 이벤트 구현

팀에게 전송되고 팀에서 조치를 취할 수 있는 알림을 생성합니다. 팀에서 조치를 취하는 데 필요한 관련 정보가 알림에 포함되도록 합니다. 보유한 각 탐지 메커니즘에 대해 런북 또는 플레이북형태의 조

사 프로세스도 있어야 합니다. 예를 들어 Amazon GuardDuty를 활성화하면 서로 다른 결과가 생성됩니다.. 각 결과 유형에 대한 런북 항목이 있어야 합니다. 예를 들어 트로이 목마 를 발견한 경우에는 누군가에게 조사 및 수정을 지시하는 간단한 지침이 런북 안에 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- AWS 서비스에서 사용할 수 있는 지표 파악: 사용 중인 서비스에 대해 Amazon CloudWatch를 통해 사용할 수 있는 지표를 검색합니다.
  - AWS 서비스 설명서
  - Amazon CloudWatch 지표 사용
- Amazon CloudWatch 경보를 구성합니다.
  - Amazon CloudWatch 경보 사용

리소스

관련 문서:

- Amazon CloudWatch
- Amazon EventBridge
- 보안 파트너 솔루션: 로깅 및 모니터링

관련 동영상:

- 리소스 구성 및 규정 준수를 중앙에서 모니터링
- Amazon GuardDuty 및 AWS Security Hub 결과 수정
- 클라우드에서 위협 관리: Amazon GuardDuty 및 AWS Security Hub

인프라 보호

질문

- SEC 5 네트워크 리소스는 어떻게 보호합니까?
- SEC 6 컴퓨팅 리소스를 어떻게 보호합니까?

SEC 5 네트워크 리소스는 어떻게 보호합니까?

인터넷이든 프라이빗 네트워크이든 상관없이 어떤 형태든 네트워크 연결이 있는 워크로드에는 외부 및 내부 네트워크 기반 위협으로부터 보호하기 위한 다중 방어 계층이 필요합니다.

모범 사례

- SEC05-BP01 네트워크 계층 생성
- SEC05-BP02 모든 계층에서 트래픽 제어
- SEC05-BP03 네트워크 보호 자동화
- SEC05-BP04 검사 및 보호 구현

SEC05-BP01 네트워크 계층 생성

민감도 요구 사항을 공유하는 구성 요소를 계층으로 그룹화하여 무단 액세스의 잠재적 영향 범위를 최소화합니다. 예를 들어 인터넷에 액세스할 필요가 없는 Virtual Private Cloud(VPC)의 데이터베이스 클러스터는 인터넷에 연결되는 경로가 없는 서브넷에 배치해야 합니다. 트래픽은 인접해 있는 다음으로 덜 민감한 리소스에서만 전달되어야 합니다. 로드 밸런서 뒤에 있는 웹 애플리케이션을 고려합니다. 데이터베이스는 로드 밸런서에서 직접 액세스할 수 없어야 합니다. 비즈니스 로직 또는 웹 서버만 데이터베이스에 직접 액세스할 수 있어야 합니다.

원하는 결과: 계층화된 네트워크를 생성합니다. 계층화된 네트워크는 유사한 네트워킹 구성 요소를 논리적으로 그룹화하는 데 도움이 됩니다. 또한 무단 네트워크 액세스의 잠재적 영향 범위를 축소합니다. 적절하게 계층화된 네트워크는 권한이 없는 사용자가 AWS 환경 내에서 추가 리소스로 전환하는 것을 더 어렵게 만듭니다. 내부 네트워크 경로를 보호하는 것 외에도 웹 애플리케이션 및 API 엔드포인트와 같은 네트워크 엣지도 보호해야 합니다.

일반적인 안티 패턴:

- 단일 VPC 또는 서브넷에서 모든 리소스를 생성합니다.
- 지나치게 관대한 보안 그룹을 사용합니다.
- 서브넷을 사용하지 못합니다.
- 데이터베이스와 같은 데이터 스토어에 직접 액세스할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

연결성 요구 사항을 공유하는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스, Amazon Relational Database Service(Amazon RDS) 데이터베이스 클러스터, AWS Lambda 함수와 같은 구성 요소를 서브넷으로 구성된 계층으로 분할할 수 있습니다. VPC 내 또는 Amazon API Gateway 뒤에 Lambda 함수와 같은 서버리스 워크로드를 배포하는 것이 좋습니다. 인터넷에 액세스할 필요가 없는 AWS Fargate 작업은 인터넷에 연결되는 경로가 없는 서브넷에 배치해야 합니다. 이러한 계층화된 접근 방식은 의도치 않은 액세스를 허용할 수 있는 단일 계층 구성 오류로 인한 영향을 완화합니다. AWS Lambda의 경우, VPC에서 함수를 실행하여 VPC 기반 제어를 활용할 수 있습니다.

수천 개의 VPC, AWS 계정 및 온프레미스 네트워크를 포함할 수 있는 네트워크 연결의 경우 AWS Transit Gateway를 사용해야 합니다. Transit Gateway는 스포크처럼 작동하는 연결된 모든 네트워크 간에 트래픽이 라우팅되는 방식을 제어하는 허브 역할을 합니다. Amazon Virtual Private Cloud(Amazon VPC)와 Transit Gateway 간의 트래픽은 AWS 프라이빗 네트워크에 남아 있어 권한이 없는 사용자 및 잠재적인 보안 문제에 대한 외부 노출을 줄입니다. 또한 Transit Gateway 리전 간 피어링은 단일 장애 지점이나 대역폭 병목 없이 리전 간 트래픽을 암호화합니다.

구현 단계

- Reachability Analyzer를 사용하여 구성에 따라 소스와 대상 사이의 경로를 분석: Reachability Analyzer를 사용하면 VPC 연결 리소스와의 연결 확인을 자동화할 수 있습니다. 이 분석은 구성을 검토하여 수행됩니다(분석을 수행할 때 네트워크 패킷이 전송되지 않음).
- Amazon VPC Network Access Analyzer를 사용하여 리소스에 대한 의도치 않은 네트워크 액세스 식별: Amazon VPC Network Access Analyzer를 사용하면 네트워크 액세스 요구 사항을 지정하고 잠재적인 네트워크 경로를 식별할 수 있습니다.
- 리소스가 퍼블릭 서브넷에 있어야 하는지 고려: 리소스가 퍼블릭 소스에서 인바운드 네트워크 트래픽을 반드시 수신해야 하는 경우가 아니라면 VPC의 퍼블릭 서브넷에 리소스를 배치하지 않아야 합니다.
- VPC에 서브넷 생성: 각 네트워크 계층(여러 가용 영역을 포함하는 그룹)에 대한 서브넷을 생성하여 마이크로 세분화를 개선합니다. 또한 라우팅 및 인터넷 연결을 제어하기 위해 올바른 라우팅 테이블을 서브넷과 연결했는지 확인합니다.
- AWS Firewall Manager를 사용하여 VPC 보안 그룹 관리: AWS Firewall Manager는 여러 보안 그룹을 사용하는 관리 부담을 줄이는 데 도움이 됩니다.
- AWS WAF를 사용하여 일반적인 웹 취약성으로부터 보호: AWS WAF는 트래픽에서 SQL 명령어 삽입과 같은 일반적인 웹 취약성을 검사하여 엣지 보안을 강화하는 데 도움이 됩니다. 또한 특정 국가 또는 지리적 위치에서 발생하는 IP 주소의 트래픽을 제한할 수 있습니다.

- Amazon CloudFront를 콘텐츠 배포 네트워크(CDN)로 사용: Amazon CloudFront는 데이터를 사용자에게 더 가깝게 저장하여 웹 애플리케이션 속도를 높이는 데 도움이 됩니다. 또한 HTTPS를 시행하고, 지리적 영역에 대한 액세스 권한을 제한하고, 네트워크 트래픽이 CloudFront를 통해 라우팅될 때만 리소스에 액세스할 수 있도록 하여 엣지 보안을 개선할 수 있습니다.
- 애플리케이션 프로그래밍 인터페이스(API) 생성 시 Amazon API Gateway 사용: Amazon API Gateway는 REST, HTTPS 및 WebSocket API를 게시, 모니터링 및 보호하는 데 도움이 됩니다.

리소스

관련 문서:

- AWS Firewall Manager
- Amazon Inspector
- Amazon VPC 보안
- Reachability Analyzer
- Amazon VPC Network Access Analyzer

관련 동영상:

- AWS Transit Gateway reference architectures for many VPCs(여러 VPC를 위한 AWS Tranit Gateway 참조 아키텍처)
- Application Acceleration and Protection with Amazon CloudFront, AWS WAF, and AWS Shield(Amazon CloudFront, AWS WAF 및 AWS Shield를 사용한 애플리케이션 가속화 및 보호)
- AWS re:Inforce 2022 - Validate effective network access controls on AWS(AWS re:Inforce 2022 - AWS에서 효과적인 네트워크 액세스 검증)
- AWS re:Inforce 2022 - Advanced protections against bots using AWS WAF(AWS re:Inforce 2022 - AWS WAF를 사용하여 봇에 대한 고급 보호)

관련 예시:

- Well-Architected Lab - Automated Deployment of VPC(VPC 자동 배포)
- 워크숍: Amazon VPC Network Access Analyzer

SEC05-BP02 모든 계층에서 트래픽 제어

네트워크 토폴로지를 설계할 때 각 구성 요소의 연결 요구 사항을 조사해야 합니다. 예를 들어 구성 요소에 인터넷 액세스(인바운드 및 아웃바운드), VPC 연결, 엣지 서비스, 외부 데이터 센터가 필요한지 조사해야 합니다.

VPC를 사용하면 설정한 프라이빗 IPv4 주소 범위 또는 AWS에서 선택한 IPv6 주소 범위를 사용하여 AWS 리전 전반의 네트워크 토폴로지를 정의할 수 있습니다. 보안 그룹(상태 저장 검사 방화벽), 네트워크 ACL, 서브넷, 라우팅 테이블을 사용하는 등 인바운드 및 아웃바운드 트래픽 모두에 대해 심층적인 방어 접근 방식을 갖춘 여러 제어를 적용해야 합니다. VPC 내의 가용 영역에서 서브넷을 생성할 수 있습니다. 각 서브넷에는 서브넷 내의 트래픽이 전송되는 경로 관리를 위한 라우팅 규칙을 정의하는 연결된 경로 테이블이 있을 수 있습니다. VPC에 연결된 인터넷 또는 NAT 게이트웨이로 이동하거나 다른 VPC를 통해 이동하는 경로를 설정하면 인터넷 라우팅 가능한 서브넷을 정의할 수 있습니다.

VPC 내에서 시작되는 인스턴스, Amazon Relational Database Service(Amazon RDS) 데이터베이스 또는 기타 서비스에는 네트워크 인터페이스별로 자체 보안 그룹이 있습니다. 이 방화벽은 운영 체제 계층 외부에 있으며, 허용되는 인바운드 및 아웃바운드 트래픽용 규칙을 정의하는 데 사용할 수 있습니다. 보안 그룹 간의 관계를 정의할 수도 있습니다. 예를 들어 데이터베이스 계층 보안 그룹 내의 인스턴스는 관련 인스턴스에 적용된 보안 그룹을 참조하여 애플리케이션 계층 내의 인스턴스에서 전송하는 트래픽만 수락합니다. 비TCP 프로토콜을 사용하지 않는 한, 로드 밸런서 또는 다음 서비스 없이 인터넷에서 직접 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스에 액세스할 필요가 없습니다(보안 그룹에 의해 제한된 포트를 사용하는 경우에도). CloudFront. 이것은 운영 체제 또는 애플리케이션 문제를 통해 이루어지는 무단 침입으로부터 보호하는 데 도움이 됩니다. 서브넷은 상태 비저장 방화벽 역할을 하는 네트워크 ACL을 연결할 수도 있습니다. 계층 간에 허용되는 트래픽 범위를 좁히도록 네트워크 ACL을 구성해야 합니다. 이때 인바운드 규칙과 아웃바운드 규칙을 모두 정의해야 합니다.

일부 AWS 서비스에서는 API 호출을 위해 AWS API 엔드포인트가 위치한 인터넷에 구성 요소가 액세스해야 합니다. 다른 AWS 서비스에서는 VPC 엔드포인트 를 Amazon VPC 내에서 사용합니다. Amazon S3 및 Amazon DynamoDB를 비롯한 여러 AWS 서비스가 VPC 엔드포인트를 지원하며, 이 기술은 다음에서 일반화되었습니다. AWS PrivateLink. 이 접근법을 사용하여 AWS 서비스, 서드 파티 서비스, 다른 VPC에 호스팅되는 사용자의 자체 서비스에 안전하게 액세스하는 것을 권장합니다. AWS PrivateLink의 모든 트래픽은 글로벌 AWS 백본에 유지되며 인터넷을 통해 이동하지 않습니다. 연결은 서비스의 제공업체가 아닌 서비스의 소비자만 시작할 수 있습니다. 외부 서비스 액세스에 AWS PrivateLink를 사용하면 인터넷 액세스 없이 에어 갭 VPC를 생성할 수 있으며 VPC를 외부 위협 벡터로부터 보호하는 데 도움이 됩니다. 서드 파티 서비스는 AWS PrivateLink를 사용하여 고객이 VPC에서 프라이빗 IP 주소를 통해 서비스에 연결하도록 할 수 있습니다. 인터넷에 아웃바운드 연결해야 하는 VPC 자산의 경우 AWS 관리형 NAT 게이트웨이, 아웃바운드 전용 인터넷 게이트웨이 또는 사용자가 생성하고 관리하는 웹 프록시를 통해 아웃바운드 전용(단방향)으로 연결할 수 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- VPC에서 네트워크 트래픽 제어: VPC 모범 사례를 구현하여 트래픽을 제어합니다.
  - Amazon VPC 보안
  - VPC 엔드포인트
  - Amazon VPC 보안 그룹
  - 네트워크 ACL
- 엣지에서 트래픽 제어: Amazon CloudFront와 같은 엣지 서비스를 구현하여 추가 보호 계층과 기타 기능을 제공합니다.
  - Amazon CloudFront 사용 사례
  - AWS Global Accelerator
  - AWS Web Application Firewall(AWS WAF)
  - Amazon Route 53
  - Amazon VPC 인그레스 라우팅
- 프라이빗 네트워크 트래픽 제어: 워크로드에 대한 프라이빗 트래픽을 보호하는 서비스를 구현합니다.
  - Amazon VPC 피어링
  - Amazon VPC 엔드포인트 서비스(AWS PrivateLink)
  - Amazon VPC Transit Gateway
  - AWS Direct Connect
  - AWS Site-to-Site VPN
  - AWS 클라이언트 VPN
  - Amazon S3 액세스 포인트

리소스

관련 문서:

- AWS Firewall Manager
- Amazon Inspector
- AWS WAF 시작하기

관련 동영상:

- AWS Transit Gateway reference architectures for many VPCs(여러 VPC를 위한 AWS Transit Gateway 참조 아키텍처)
- Amazon CloudFront, AWS WAF, AWS Shield를 사용한 애플리케이션 가속화 및 보호

관련 예시:

- 실습: VPC 자동 배포

SEC05-BP03 네트워크 보호 자동화

위협 정보 및 이상 상태 감지 결과에 따라 자체 방어 네트워크를 제공하는 보호 메커니즘을 자동화합니다. 예를 들어 최신 위협에 적응하고 위협의 영향을 줄일 수 있는 침입 탐지 및 방지 도구가 있습니다. 웹 애플리케이션 방화벽은 네트워크 보호를 자동화할 수 있는 곳의 일례입니다. 예를 들어 AWS WAF Security Automations 솔루션(https://github.com/awslabs/aws-waf-security-automations)을 사용하여 알려진 위협 요소와 연결된 IP 주소에서 시작되는 요청을 자동으로 차단할 수 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 웹 기반 트래픽에 대한 보호 자동화: AWS는 일반적인 웹 기반 공격을 필터링하도록 설계된 AWS WAF 규칙 세트를 AWS CloudFormation을 사용하여 자동으로 배포하는 솔루션을 제공합니다. 사용자는 AWS WAF 웹 액세스 제어 목록(웹 ACL)에 포함된 규칙을 정의하도록 사전 구성된 다양한 보호 기능 중에서 선택할 수 있습니다.
  - AWS WAF 보안 자동화
- AWS Partner 솔루션 고려: AWS 파트너는 고객 온프레미스 환경의 기존 제어 솔루션과 동등한 수준이거나, 동일하거나 통합된 업계 최고 수준의 수백 가지 제품을 제공합니다. 이러한 제품은 기존 AWS 서비스를 보완하여 클라우드 및 온프레미스 환경에 포괄적인 보안 아키텍처와 보다 원활한 환경을 배포할 수 있도록 해줍니다.
  - 인프라 보안

리소스

관련 문서:

- AWS Firewall Manager

- Amazon Inspector
- Amazon VPC 보안
- AWS WAF 시작하기

관련 동영상:

- AWS Transit Gateway reference architectures for many VPCs(여러 VPC를 위한 AWS Transit Gateway 참조 아키텍처)
- Amazon CloudFront, AWS WAF, AWS Shield를 사용한 애플리케이션 가속화 및 보호

관련 예시:

- 실습: VPC 자동 배포

SEC05-BP04 검사 및 보호 구현

각 계층에서 트래픽을 검사하고 필터링합니다. 사용자는 VPC Network Access Analyzer를 사용하여 잠재적인 의도치 않은 액세스에 대해 VPC 구성을 검사할 수 있습니다. 네트워크 액세스 요구 사항을 지정하고 이 요구 사항을 충족하지 않는 잠재적인 네트워크 경로를 찾을 수 있습니다. HTTP 기반 프로토콜을 통해 트랜잭션되는 구성 요소의 경우, 웹 애플리케이션 방화벽이 일반 공격으로부터 보호하는 데 도움을 줄 수 있습니다. AWS WAF 는 Amazon API Gateway API, Amazon CloudFront 또는 Application Load Balancer로 전달되는 구성 가능한 규칙과 일치하는 HTTP(s) 요청을 모니터링하고 차단할 수 있는 웹 애플리케이션 방화벽입니다. AWS WAF를 시작하려면 AWS Managed Rules 를 자체 규칙과 함께 사용하거나 기존 파트너 통합을 사용할 수 있습니다..

AWS Organizations 전반에서 AWS WAF, AWS Shield Advanced 보호, Amazon VPC 보안 그룹을 관리하기 위해 AWS Firewall Manager를 사용할 수 있습니다. 그러면 여러 계정과 애플리케이션의 방화벽 규칙을 중앙에서 구성하고 관리할 수 있으므로 일반 규칙 적용을 좀 더 쉽게 확장할 수 있습니다. 공격에 신속하게 대응하기 위해 AWS Shield Advanced나 웹 애플리케이션에 대한 원치 않는 요청을 자동으로 차단할 수 있는 솔루션 을 사용할 수도 있습니다. Firewall Manager는 AWS Network Firewall과도 연동됩니다. AWS Network Firewall은 규칙 엔진을 사용하여 스테이트풀 및 스테이트리스 네트워크를 모두 세세하게 제어할 수 있는 관리형 서비스입니다. 이 서비스는 워크로드 보호에 도움이 되도록 규칙에 Suricata 호환 오픈 소스 침입 예방 시스템(IPS) 사양을 지원합니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- Amazon GuardDuty 구성: GuardDuty는 악성 활동 및 무단 행위를 지속적으로 모니터링하여 AWS 계정 및 워크로드를 보호하는 위협 탐지 서비스입니다. GuardDuty를 활성화하고 자동 알림을 구성합니다.
  - Amazon GuardDuty
  - 실습: 탐지 제어 자동 배포
- Virtual Private Cloud(VPC) 흐름 로그 구성: VPC 흐름 로그는 VPC의 네트워크 인터페이스에서 전송되고 수신되는 IP 트래픽에 대한 정보를 캡처하는 데 사용할 수 있는 기능입니다. 흐름 로그 데이터를 Amazon CloudWatch Logs 및 Amazon Simple Storage Service(Amazon S3)에 게시할 수 있습니다. 흐름 로그를 생성한 후에는 선택한 대상에서 해당 데이터를 검색하여 확인할 수 있습니다.
- VPC 트래픽 미러링 고려: 트래픽 미러링은 콘텐츠 검사, 위협 모니터링 및 문제 해결을 위해 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스의 탄력적 네트워크 인터페이스에서 네트워크 트래픽을 복사한 다음 대역 외 보안 및 모니터링 어플라이언스로 전송하는 데 사용할 수 있는 Amazon VPC 기능입니다.
  - VPC 트래픽 미러링

리소스

관련 문서:

- AWS Firewall Manager
- Amazon Inspector
- Amazon VPC 보안
- AWS WAF 시작하기

관련 동영상:

- 여러 VPC를 위한 AWS Transit Gateway 참조 아키텍처
- Amazon CloudFront, AWS WAF, AWS Shield를 사용한 애플리케이션 가속화 및 보호

관련 예시:

- 실습: VPC 자동 배포

SEC 6 컴퓨팅 리소스를 어떻게 보호합니까?

워크로드의 컴퓨팅 리소스는 다계층 방어를 통해 내/외부 위협으로부터 보호해야 합니다. 컴퓨팅 리소스에는 EC2 인스턴스, 컨테이너, AWS Lambda 함수, 데이터베이스 서비스, IoT 디바이스 등이 포함됩니다.

모범 사례

- SEC06-BP01 취약성 관리 수행
- SEC06-BP02 공격 표면 축소
- SEC06-BP03 관리형 서비스 구현
- SEC06-BP04 컴퓨팅 보호 자동화
- SEC06-BP05 사용자가 원격으로 작업을 수행할 수 있도록 지원
- SEC06-BP06 소프트웨어 무결성 검증

SEC06-BP01 취약성 관리 수행

코드, 종속성 및 인프라에 취약성이 있는지 자주 스캔하고 패치를 적용하여 새로운 위협으로부터 보호합니다.

원하는 결과: 취약성 관리 프로그램을 생성하고 유지합니다. Amazon EC2 인스턴스, Amazon Elastic Container Service(Amazon ECS) 컨테이너 및 Amazon Elastic Kubernetes Service(Amazon EKS) 워크로드와 같은 리소스를 정기적으로 스캔하고 패치를 적용합니다. Amazon Relational Database Service(Amazon RDS) 데이터베이스와 같은 AWS 관리형 리소스에 대한 유지 관리 기간을 구성합니다. 정적 코드 스캔을 사용하여 일반적인 문제에 대한 애플리케이션 소스 코드를 검사합니다. 조직에 필요한 기술이 있거나 외부 지원을 고용할 수 있는 경우 웹 애플리케이션 침투 테스트를 고려합니다.

일반적인 안티 패턴:

- 취약성 관리 프로그램이 없습니다.
- 심각도나 위험 회피를 고려하지 않고 시스템 패치 적용을 수행합니다.
- 공급업체에서 제공한 수명 종료(EOL) 날짜가 지난 소프트웨어를 사용합니다.
- 보안 문제를 분석하기 전에 코드를 프로덕션 환경에 배포합니다.

이 모범 사례 확립의 이점:

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

취약성 관리 프로그램에는 문제 해결의 일환으로 보안 평가, 문제 식별, 우선순위 지정 및 패치 작업 수행이 포함됩니다. 자동화는 워크로드를 지속적으로 스캔하여 문제 및 의도치 않은 네트워크 노출이 있는지 확인하고 문제 해결 수행이 핵심입니다. 리소스 생성 및 업데이트를 자동화하면 시간이 절약되고 추가 문제를 야기하는 구성 오류의 위험이 줄어듭니다. 잘 설계된 취약성 관리 프로그램은 소프트웨어 수명 주기의 개발 및 배포 단계에서 취약성 테스트도 고려해야 합니다. 개발 및 배포 중에 취약성 관리를 구현하면 취약성이 프로덕션 환경에 침투할 가능성을 줄이는 데 도움이 됩니다.

취약성 관리 프로그램을 구현하려면 AWS 공동 책임 모델과 이 모델이 특정 워크로드와 어떻게 관련되어 있는지 파악하고 있어야 합니다. 공동 책임 모델에 따라 AWS는 AWS 클라우드의 인프라를 보호하는 업무를 담당합니다. 이 인프라는 AWS 클라우드 서비스를 실행하는 하드웨어, 소프트웨어, 네트워킹 및 시설로 구성됩니다. 사용자는 Amazon EC2 인스턴스의 실제 데이터, 보안 구성, 관리 작업과 같은 클라우드의 보안을 담당하고 Amazon S3 객체가 적절하게 분류 및 구성되었는지 확인해야 합니다. 취약성 관리에 대한 접근 방식은 사용하는 서비스에 따라 달라질 수 있습니다. 예를 들어 AWS은 관리되는 관계형 데이터베이스 서비스인 Amazon RDS에 대한 패치 적용을 관리하지만 셀프 호스팅 데이터베이스에 패치 적용은 사용자가 담당합니다.

AWS에는 취약성 관리 프로그램을 지원하는 다양한 서비스가 있습니다. Amazon Inspector는 AWS 워크로드를 지속적으로 스캔하여 소프트웨어 문제 및 의도치 않은 네트워크 액세스가 있는지 확인합니다. AWS Systems Manager Patch Manager는 Amazon EC2 인스턴스 전반에 대한 패치 적용을 관리하는 데 도움이 됩니다. Amazon Inspector 및 Systems Manager는 AWS 보안 점검을 자동화하고 보안 알림을 중앙 집중화하는 데 도움이 되는 클라우드 보안 상태 관리 서비스인 AWS Security Hub에서 볼 수 있습니다.

Amazon CodeGuru는 정적 코드 분석을 사용하여 Java 및 Python 애플리케이션에서 잠재적인 문제를 식별하는 데 도움이 될 수 있습니다.

구현 단계

- Amazon Inspector 구성: Amazon Inspector는 새로 실행된 Amazon EC2 인스턴스, Lambda 함수 및 Amazon ECR에 푸시된 적격한 컨테이너 이미지를 자동으로 감지하고 즉시 스캔하여 소프트웨어 문제, 잠재적 결함 및 의도치 않은 네트워크 노출이 있는지 확인합니다.
- 소스 코드 스캔: 라이브러리 및 종속성을 스캔하여 문제 및 결함이 있는지 확인합니다. Amazon CodeGuru는 Java 및 Python 애플리케이션을 모두 스캔하여 일반적인 보안 문제가 있는지 확인하고 이를 해결하기 위한 권장 사항을 제공할 수 있습니다. OWASP Foundation은 소스 코드 분석 도구(SAST 도구라고도 함) 목록을 게시합니다.
- CI/CD 파이프라인 빌드 프로세스의 일환으로 스캔할 뿐만 아니라 기존 환경을 스캔하고 패치를 적용하는 메커니즘 구현: 보호를 위해 종속성 및 운영 체제의 문제를 스캔하고 패치를 적용하는 메커

니즘을 구현하여 새로운 위협으로부터 보호합니다. 해당 메커니즘을 정기적으로 실행합니다. 소프트웨어 취약성 관리는 패치를 적용하거나 소프트웨어 문제를 해결해야 하는 위치를 파악하는 데 필수적입니다. 취약성 평가를 지속적 통합/지속적 전달(CI/CD) 파이프라인에 조기에 포함하여 잠재적 보안 문제의 해결 우선순위를 지정합니다. 사용 중인 AWS 서비스에 따라 접근 방식이 달라질 수 있습니다. Amazon EC2 인스턴스에서 실행 중인 소프트웨어의 잠재적 문제를 확인하려면 Amazon Inspector를 파이프라인에 추가하여 문제 또는 잠재적 결함이 감지되는 경우 이를 알리고 빌드 프로세스를 중지합니다. Amazon Inspector는 지속적으로 리소스를 모니터링합니다. OWASP Dependency-Check, Snyk, OpenVAS와 같은 오픈 소스 제품, 패키지 관리자, 취약성 관리를 위한 AWS Partner 도구를 사용할 수도 있습니다.

- AWS Systems Manager 사용: AWS(Amazon Elastic Compute Cloud) 인스턴스, Amazon Machine Image(AMI), 기타 컴퓨팅 리소스를 비롯한 Amazon EC2 리소스에 대한 패치 관리에 대한 책임은 사용자에게 있습니다. AWS Systems Manager Patch Manager는 보안 관련 업데이트와 기타 유형의 업데이트를 모두 사용하여 관리형 인스턴스에 패치를 적용하는 프로세스를 자동화합니다. Patch Manager는 Microsoft 애플리케이션, Windows 서비스 팩, Linux 기반 인스턴스용 마이너 버전 업그레이드를 포함하여 운영 체제 및 애플리케이션 모두에 대해 Amazon EC2 인스턴스에 패치를 적용하는 데 사용할 수 있습니다. Amazon EC2 외에도 Patch Manager를 사용하면 온프레미스 서버에 패치를 적용할 수도 있습니다.

  지원되는 운영 체제 목록은 Systems Manager 사용자 가이드에서 지원되는 운영 체제를 참조하세요. 인스턴스를 스캔하여 누락된 패치 보고서만 확인하거나, 스캔하고 누락된 모든 패치를 자동으로 설치할 수 있습니다.
- AWS Security Hub 사용: Security Hub는 AWS의 보안 상태에 대한 포괄적인 보기를 제공합니다. 여러 AWS 서비스에서 보안 데이터를 수집하고 이러한 조사 결과를 표준화된 형식으로 제공하므로 AWS 서비스에서 보안 조사 결과의 우선순위를 지정할 수 있습니다.
- AWS CloudFormation 사용: AWS CloudFormation은 여러 계정 및 환경에서 리소스 배포를 자동화하고 리소스 아키텍처를 표준화하여 취약성 관리에 도움을 줄 수 있는 코드형 인프라(IaC) 서비스입니다.

리소스

관련 문서:

- AWS Systems Manager
- AWS Lambda 보안 개요
- Amazon CodeGuru
- 새로운 Amazon Inspector를 사용하여 클라우드 워크로드를 위한 개선되고 자동화된 취약성 관리

- Amazon Inspector 및 AWS Systems Manager를 사용하여 AWS에서 취약성 관리 및 개선조치 자동화 - 1부

관련 동영상:

- Securing Serverless and Container Services(서버리스 및 컨테이너 서비스 보호)
- Security best practices for the Amazon EC2 instance metadata service(Amazon EC2 인스턴스 메타데이터 서비스에 대한 보안 모범 사례)

SEC06-BP02 공격 표면 축소

운영 체제를 강화하고 사용 중인 구성 요소, 라이브러리 및 외부 사용 서비스를 최소화하여 의도치 않은 액세스에 대한 노출을 줄입니다. 운영 체제 패키지 또는 애플리케이션이나(Amazon Elastic Compute Cloud(Amazon EC2) 기반 워크로드의 경우) 코드의 외부 소프트웨어 모듈(모든 워크로드의 경우)에서 사용하지 않는 구성 요소를 줄이는 것으로 시작합니다. 일반적인 운영 체제 및 서버 소프트웨어에 대한 여러 가지 강화 및 보안 구성 가이드를 찾아볼 수 있습니다. 예를 들면 Center for Internet Security 로 시작하고 반복하면 됩니다.

Amazon EC2에서는 자체적으로 패치하고 강화한 Amazon Machine Image(AMI)를 생성하여 조직의 특정 보안 요구 사항을 충족할 수 있습니다. AMI에 적용하는 패치 및 기타 보안 제어 조치는 생성 시점에 발효되며 시작한 후 AWS Systems Manager 등을 사용하여 수정하지 않는 이상 동적으로 변경되지 않습니다.

EC2 Image Builder를 사용하여 보안 AMI 구축 프로세스를 간소화할 수 있습니다. EC2 Image Builder는 자동화를 작성하고 유지 관리할 필요 없이 골든 이미지를 생성하여 유지 관리하는 데 필요한 노력을 크게 절감해 줍니다. 소프트웨어 업데이트가 가능하면 사용자가 이미지 빌드를 시작할 필요 없이 Image Builder가 자동으로 새로운 이미지를 생성합니다. EC2 Image Builder를 사용하면 이미지를 프로덕션에 사용하기 전에 AWS에서 제공하는 테스트와 사용자의 자체 테스트를 사용하여 이미지의 기능과 보안을 쉽게 검증할 수 있습니다. 또한 AWS에서 제공하는 보안 설정을 적용하여 이미지 보안을 강화함으로써 내부 보안 기준을 충족할 수 있습니다. 예를 들면 AWS에서 제공하는 템플릿을 사용하여 Security Technical Implementation Guide(STIG) 표준을 준수하는 이미지를 생성할 수 있습니다.

서드 파티 정적 코드 분석 도구를 사용하여 확인되지 않은 함수 입력 범위와 해당하는 일반 취약성 및 노출(CVE)와 같은 일반적인 보안 문제를 식별합니다. 전용 인프라에서 Amazon CodeGuru 를 지원되는 언어에 대해 사용할 수 있습니다. 또한 종속성 확인 도구를 사용하여 코드가 링크된 라이브러리가 최신 버전인지, 해당 라이브러리에 CVE가 없는지, 소프트웨어 정책 요구 사항에 부합하는 라이선스 조건이 있는지를 확인할 수 있습니다.

Amazon Inspector를 사용하면 인스턴스에 대해 알려진 CVE를 확인하는 구성 평가를 수행하고, 보안 벤치마크를 기준으로 평가하고, 결함 알림을 자동화할 수 있습니다. 프로덕션 인스턴스 또는 빌드 파이프라인에서 실행되는 Amazon Inspector는 확인된 정보가 있으면 개발자와 엔지니어에게 알림을 보냅니다. 프로그래밍 방식으로 확인된 정보에 접근할 수 있으며, 팀에게 백로그 및 버그 추적 시스템에 접근 권한을 제공할 수 있습니다. EC2 Image Builder 는 자동화된 패치 적용, AWS에서 제공하는 보안 정책 적용 및 기타 사용자 지정을 통해 서버 이미지(AMI)를 유지 관리하는 데 사용될 수 있습니다. 컨테이너를 사용할 때는 빌드 파이프라인에서 이미지 리포지토리에 대해 정기적으로 ECR Image Scanning 을 구현하여 컨테이너에서 CVE를 찾습니다.

Amazon Inspector 및 기타 도구는 존재하는 구성 및 CVE를 식별하는 데 효과적이지만, 애플리케이션 수준에서 워크로드를 테스트하려면 다른 방법이 필요합니다. Fuzzing 은 자동화를 사용하여 잘못된 형식의 데이터를 입력 필드 및 애플리케이션의 기타 영역에 주입함으로써 버그를 찾는 유명한 방법입니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 운영 체제 강화: 모범 사례에 맞춰 운영 체제를 구성합니다.
  - Amazon Linux 보호
  - Microsoft Windows Server 보호
- 컨테이너식 리소스 강화: 보안 모범 사례에 맞춰 컨테이너식 리소스를 구성합니다.
- AWS Lambda 모범 사례를 구현합니다.
  - AWS Lambda 모범 사례

리소스

관련 문서:

- AWS Systems Manager
- Amazon EC2 Systems Manager로 Bastion 호스트 대체
- AWS Lambda 보안 개요

관련 동영상:

- Amazon EKS에서 고도의 보안 워크로드 실행
- Securing Serverless and Container Services

- Amazon EC2 인스턴스 메타데이터 서비스에 대한 보안 모범 사례

관련 예시:

- 실습: 웹 애플리케이션 방화벽 자동 배포

SEC06-BP03 관리형 서비스 구현

공유 책임 모델의 일환으로 보안 유지 관리 태스크를 줄일 수 있도록 Amazon Relational Database Service(Amazon RDS), AWS Lambda, Amazon Elastic Container Service(Amazon ECS) 등 리소스를 관리하는 서비스를 구현합니다. 예를 들어 Amazon RDS는 관계형 데이터베이스를 설정, 운영, 확장하는 데 도움을 주고 하드웨어 프로비저닝, 데이터베이스 설정, 패치 적용, 백업 등의 관리 작업을 자동화합니다. 즉, AWS Well-Architected Framework에 설명된 다른 방법으로 애플리케이션을 보호하는 데 더 많은 시간을 할애할 수 있습니다. Lambda를 사용하면 서버를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있으므로 인프라나 운영 체제가 아니라 코드 수준의 연결, 호출, 보안에만 집중하면 됩니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 사용 가능한 서비스 탐색: Amazon RDS, AWS Lambda, Amazon ECS 등 리소스를 관리하는 서비스를 탐색, 테스트 및 구현합니다.

리소스

관련 문서:

- AWS 웹 사이트
- AWS Systems Manager
- Replacing a Bastion Host with Amazon EC2 Systems Manager(Amazon EC2 Systems Manager로 Bastion 호스트 대체)
- AWS Lambda 보안 개요

관련 동영상:

- Running high-security workloads on Amazon EKS(Amazon EKS에서 고보안 워크로드 실행)
- Securing Serverless and Container Services

- Security best practices for the Amazon EC2 instance metadata service(Amazon EC2 인스턴스 메타데이터 서비스에 대한 보안 모범 사례)

관련 예시:

- 실습: AWS Certificate Manager 퍼블릭 인증서 요청

SEC06-BP04 컴퓨팅 보호 자동화

취약성 관리, 공격 대상 영역 축소, 리소스 관리 등 컴퓨팅 보호 메커니즘을 자동화합니다. 자동화를 사용하면 워크로드의 다른 측면을 보호하는 데 시간을 투자하고 인적 오류의 위험을 줄일 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 구성 관리 자동화: 구성 관리 서비스 또는 도구를 사용하여 보안 구성을 자동으로 적용하고 확보합니다.
  - AWS Systems Manager
  - AWS CloudFormation
  - 실습: VPC 자동 배포
  - 실습: EC2 웹 애플리케이션 자동 배포

- Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스 자동 패칭: AWS Systems Manager Patch Manager는 보안 관련 업데이트와 기타 유형의 업데이트를 모두 사용하여 관리형 인스턴스를 패치하는 프로세스를 자동화합니다. Patch Manager를 사용하여 운영 체제와 애플리케이션 모두에 패치를 적용할 수 있습니다.
  - AWS Systems Manager Patch Manager
  - AWS Systems Manager Automation으로 다중 계정 및 다중 리전 패치 작업을 중앙 집중화

- 침입 탐지 및 차단 구현: 침입 탐지 및 차단 도구를 구현하여 인스턴스에서 악의적인 활동을 모니터링하고 중지합니다.
- AWS Partner 솔루션 고려: AWS 파트너는 고객 온프레미스 환경의 기존 제어 솔루션과 동등한 수준이거나, 동일하거나 통합된 업계 최고 수준의 수백 가지 제품을 제공합니다. 이러한 제품은 기존 AWS 서비스를 보완하여 클라우드 및 온프레미스 환경에 포괄적인 보안 아키텍처와 보다 원활한 환경을 배포할 수 있도록 해줍니다.

- 인프라 보안

리소스

관련 문서:

- AWS CloudFormation
- AWS Systems Manager
- AWS Systems Manager Patch Manager
- AWS Systems Manager Automation으로 다중 계정 및 다중 리전 패치 작업을 중앙 집중화
- 인프라 보안
- Amazon EC2 Systems Manager로 Bastion 호스트 대체
- AWS Lambda 보안 개요

관련 동영상:

- Amazon EKS에서 고도의 보안 워크로드 실행
- Securing Serverless and Container Services
- Amazon EC2 인스턴스 메타데이터 서비스에 대한 보안 모범 사례

관련 예시:

- 실습: 웹 애플리케이션 방화벽 자동 배포
- 실습: EC2 웹 애플리케이션 자동 배포

SEC06-BP05 사용자가 원격으로 작업을 수행할 수 있도록 지원

대화형 액세스 기능을 제거하면 인적 오류의 위험과 수동 구성 또는 관리의 필요성을 줄일 수 있습니다. 예를 들어, 변경 관리 워크플로를 사용하여 코드형 인프라를 사용하는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 배포한 후 직접 액세스를 허용하는 대신 AWS Systems Manager와 같은 도구를 사용하거나 Bastion 호스트를 통해 Amazon EC2 인스턴스를 관리합니다. AWS Systems Manager는 자동화 워크플로, 문서 (플레이북) 및 Run Command등의 기능을 사용하여 다양한 유지 관리 및 배포 작업을 자동화할 수 있습니다. AWS CloudFormation 스택은 파이프라인에서 구축되며 AWS Management Console 또는 API를 직접 사용하지 않고도 인프라 배포 및 관리 작업을 자동화할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 콘솔 액세스 대체: AWS Systems Manager Run Command로 인스턴스에 대한 콘솔 액세스(SSH 또는 RDP)를 교체하여 관리 태스크를 자동화합니다.

- AWS Systems Manager Run Command

리소스

관련 문서:

- AWS Systems Manager
- AWS Systems Manager Run Command
- Replacing a Bastion Host with Amazon EC2 Systems Manager(Amazon EC2 Systems Manager로 Bastion 호스트 대체)
- AWS Lambda 보안 개요

관련 동영상:

- Running high-security workloads on Amazon EKS(Amazon EKS에서 고보안 워크로드 실행)
- Securing Serverless and Container Services
- Security best practices for the Amazon EC2 instance metadata service(Amazon EC2 인스턴스 메타데이터 서비스에 대한 보안 모범 사례)

관련 예시:

- 실습: 웹 애플리케이션 방화벽 자동 배포

SEC06-BP06 소프트웨어 무결성 검증

워크로드에 사용되는 소프트웨어, 코드, 라이브러리가 신뢰할 수 있는 소스에서 온 것이며 변조되지 않았는지 검증하는 메커니즘(예: 코드 서명)을 구현합니다. 예를 들어 바이너리 및 스크립트의 코드 서명 인증서를 검토하여 작성자를 확인하고 작성자가 생성한 후에 변조되지 않았는지 확인해야 합니다. AWS Signer 는 인증서 서명과 퍼블릭 및 프라이빗 키 등의 코드 서명 수명 주기를 중앙에서 관리하여

코드의 신뢰성과 무결성을 보장하는 데 도움이 됩니다. 코드 서명에 대한 고급 패턴과 모범 사례를 사용하는 방법은 다음에서 확인할 수 있습니다. AWS Lambda. 또한 다운로드한 소프트웨어의 체크섬을 공급자의 체크섬과 비교하면 변조되지 않았는지를 확인하는 데 도움이 될 수 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 메커니즘 조사: 코드 서명은 소프트웨어 무결성을 검증하는 데 사용할 수 있는 메커니즘입니다.
  - NIST: 코드 서명을 위한 보안 고려 사항

리소스

관련 문서:

- AWS Signer
- New - Code Signing, a Trust and Integrity Control for AWS Lambda(신규 - 코드 서명, AWS Lambda의 신뢰 및 무결성 제어)

데이터 보호

질문

- SEC 7 데이터는 어떻게 분류합니까?
- SEC 8 저장된 데이터는 어떻게 보호합니까?
- SEC 9 전송 중인 데이터는 어떻게 보호합니까?

SEC 7 데이터는 어떻게 분류합니까?

분류는 적절한 보호 및 보존 제어 수준을 결정하는 데 도움이 되도록 중요도와 민감도를 기준으로 데이터를 분류하는 방법을 제공합니다.

모범 사례

- SEC07-BP01 워크로드 안에서 데이터 식별
- SEC07-BP02 데이터 보호 제어 정의
- SEC07-BP03 식별 및 분류 자동화
- SEC07-BP04 데이터 수명 주기 관리 정의

SEC07-BP01 워크로드 안에서 데이터 식별

워크로드가 처리하는 데이터의 유형과 분류, 관련 비즈니스 프로세스, 데이터가 저장되는 위치, 데이터 소유자를 이해하는 것이 중요합니다. 또한 워크로드의 해당 법률 및 규정 준수 요구 사항과 적용해야 하는 데이터 제어를 이해해야 합니다. 데이터 식별은 데이터 분류 여정의 첫 번째 단계입니다.

이 모범 사례 확립의 이점:

데이터 분류를 통해 워크로드 소유자는 민감한 데이터를 저장하는 위치를 식별하고 해당 데이터에 액세스하고 공유하는 방법을 결정할 수 있습니다.

데이터 분류는 다음 질문에 답변하는 것을 목표로 합니다.

- 어떤 유형의 데이터가 있나요?

  다음과 같은 데이터가 있을 수 있습니다.
  - 영업 비밀, 특허, 계약과 같은 지적 재산(IP)
  - 개인과 연결된 병력 정보가 포함된 의료 기록과 같은 보호 대상 건강 정보(PHI)
  - 이름, 주소, 생년월일, 국가 ID, 등록 번호와 같은 개인 식별 정보(PII)
  - 기본 계정 번호(PAN), 카드 소유자 이름, 만료 날짜, 서비스 코드 번호와 같은 신용 카드 데이터
  - 민감한 데이터는 어디에 저장되나요?
  - 누가 데이터에 액세스하여 수정 및 삭제할 수 있나요?
  - 데이터를 잘못 취급하지 못하도록 방지하려면 사용자 권한을 이해하는 것이 필수적입니다.
- 생성, 읽기, 업데이트, 삭제(CRUD) 작업은 누가 수행할 수 있나요?
  - 누가 데이터에 대한 권한을 관리할 수 있는지 파악하여 권한 상승 가능성을 고려합니다.
- 데이터가 의도치 않게 공개, 변경, 삭제되면 비즈니스에 어떤 영향을 미칠 수 있나요?
  - 데이터가 의도치 않게 수정, 삭제, 공개되는 경우의 위험 결과를 이해합니다.

이러한 질문에 대한 답변을 알면 다음과 같은 조치를 취할 수 있습니다.

- 민감한 데이터 범위(예: 민감한 데이터 위치 수)를 줄이고 민감한 데이터에 대한 액세스 권한을 승인된 사용자로만 제한합니다.
- 암호화, 데이터 손실 방지, 자격 증명 및 액세스 관리와 같은 적절한 데이터 보호 메커니즘 및 기술을 구현할 수 있도록 다양한 데이터 유형을 파악합니다.
- 데이터에 대한 올바른 제어 목표를 제공하여 비용을 최적화합니다.

- 데이터 유형과 양, 민감도가 다른 데이터를 서로 분리하는 방법에 대한 규제 기관 및 감사 담당자의 질문에 자신 있게 답변합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

데이터 분류는 데이터의 민감도를 식별하는 행위입니다. 데이터를 쉽게 검색하고 추적할 수 있도록 태그 지정이 포함될 수 있습니다. 또한 데이터 분류는 데이터 중복을 줄여 검색 프로세스 속도를 높이면서 스토리지 및 백업 비용을 줄이는 데 도움이 될 수 있습니다.

Amazon Macie과 같은 서비스를 사용하여 민감한 데이터의 검색 및 분류를 대규모로 자동화합니다. Amazon EventBridge 및 AWS Config와 같은 다른 서비스를 사용하면 암호화되지 않은 Amazon Simple Storage Service(Amazon S3) 버킷 및 Amazon EC2 EBS 볼륨 또는 태그가 지정되지 않은 데이터 리소스와 같은 데이터 보안 문제에 대한 개선조치를 자동화할 수 있습니다. AWS 서비스 통합의 전체 목록은 EventBridge 설명서를 참조하세요.

고객 이메일, 지원 티켓, 제품 리뷰, 소셜 미디어와 같은 비정형 데이터에서 PII 감지는 자연어 처리(NLP) 서비스인 Amazon Comprehend를 사용하면 됩니다. 이 서비스는 기계 학습(ML)을 사용하여 비정형 텍스트에서 사람, 장소, 감정, 주제와 같은 인사이트와 관계를 찾습니다. 데이터 식별을 지원할 수 있는 AWS 서비스 목록은 AWS 서비스를 사용하여 PHI 및 PII 데이터를 감지하는 일반 기술을 참조하세요.

데이터 분류 및 보호를 지원하는 또 다른 방법은 AWS 리소스 태그 지정입니다. 태그 지정을 사용하면 리소스를 관리, 식별, 구성, 검색 및 필터링하는 데 사용할 수 있는 메타데이터를 AWS 리소스에 할당할 수 있습니다.

경우에 따라 전체 리소스(예: S3 버킷)에 태그를 지정하도록 선택할 수 있습니다. 특히 특정 워크로드 또는 서비스가 이미 알려진 데이터 분류의 프로세스 또는 전송을 저장해야 하는 경우에 더욱 그렇습니다.

적절한 경우 관리 및 보안 유지 관리를 쉽게 하기 위해 개별 객체 대신 S3 버킷에 태그를 지정할 수 있습니다.

구현 단계

Amazon S3 내의 민감한 데이터 감지:

1. 시작하기 전에 Amazon Macie 콘솔 및 API 작업에 액세스할 수 있는 적절한 권한이 있는지 확인합니다. 자세한 내용은 Amazon Macie 시작하기를 참조하세요.

2. 민감한 데이터가 Amazon S3에 있는 경우 Amazon Macie를 사용하여 자동화된 데이터 검색을 수행합니다.
   - Amazon Macie 시작하기 가이드를 사용하여 민감한 데이터 검색 결과에 대한 리포지토리를 구성하고 민감한 데이터에 대한 검색 작업을 생성합니다.
   - Amazon Macie를 사용하여 S3 버킷에서 민감한 정보를 미리 보는 방법

     기본적으로 Macie는 자동화된 민감한 데이터 검색을 위해 권장되는 관리형 데이터 식별자를 사용하여 객체를 분석합니다. 계정 또는 조직에 대해 자동화된 민감한 데이터 검색을 수행할 때 특정 관리형 데이터 식별자, 사용자 지정 데이터 식별자 및 허용 목록을 사용하도록 Macie를 구성하여 분석을 맞춤 조정할 수 있습니다. 특정 버킷(예: 일반적으로 AWS 로깅 데이터를 저장하는 S3 버킷)을 제외하여 분석 범위를 조정할 수 있습니다.
3. 자동화된 민감한 데이터 검색을 구성하고 사용하려면 Performing automated sensitive data discovery with Amazon Macie(Amazon Macie를 사용하여 자동화된 민감한 데이터 검색 수행)를 참조하세요.
4. Amazon Macie에 대한 자동 데이터 검색을 고려할 수도 있습니다.

Amazon RDS 내의 민감한 데이터 감지:

Amazon Relational Database Service(Amazon RDS) 데이터베이스의 데이터 검색에 대한 자세한 내용은 Macie를 사용하여 Amazon RDS 데이터베이스에 대한 데이터 분류 활성화를 참조하세요.

DynamoDB 내의 민감한 데이터 감지:

- Macie를 사용하여 DynamoDB에서 민감한 데이터 감지는 Amazon Macie를 사용하여 스캔을 위해 데이터를 Amazon S3로 내보내서 Amazon DynamoDB 테이블에서 민감한 데이터를 감지하는 방법을 설명합니다.

AWS 파트너 솔루션:

- 광범위한 AWS Partner Network 사용을 고려합니다. AWS 파트너는 AWS 서비스와 직접 통합되는 광범위한 도구 및 규정 준수 프레임워크를 보유하고 있습니다. 파트너는 조직의 요구 사항을 충족하는 데 도움이 되는 맞춤형 거버넌스 및 규정 준수 솔루션을 제공할 수 있습니다.
- 데이터 분류의 맞춤형 솔루션은 Data governance in the age of regulation and compliance requirements(규제 및 규정 준수 요구 사항 시대의 데이터 거버넌스)를 참조하세요.

AWS Organizations를 사용하여 정책을 생성하고 배포하여 조직에서 채택하는 태그 지정 표준을 자동으로 적용할 수 있습니다. 태그 정책을 사용하면 유효한 키 이름과 각 키에 유효한 값을 정의하는 규칙을 지정할 수 있습니다. 모니터링만 선택하면 기존 태그를 평가하고 정리할 수 있습니다. 태그가 선택한 표준을 준수하면 태그 정책에서 적용을 사용 설정하여 규정 미준수 태그가 생성되는 것을 방지할 수 있습니다. 자세한 내용은 AWS Organizations에서 서비스 제어 정책을 사용하여 권한 부여에 사용되는 리소스 태그 보호 및 권한이 부여된 보안 주체에 의한 것을 제외한 태그 수정 방지에 대한 예시 정책을 참조하세요.

- AWS Organizations에서 태그 정책 사용을 시작하려면 고급 태그 정책으로 이동하기 전에 태그 정책 시작하기의 워크플로를 따르는 것이 좋습니다. 전체 조직 단위(OU) 또는 조직으로 확장하기 전에 단일 계정에 간단한 태그 정책을 연결하는 효과를 이해하면 태그 정책을 준수하기 전에 태그 정책의 효과를 볼 수 있습니다. 태그 정책 시작하기에서는 고급 정책 관련 작업에 대한 지침 링크를 제공합니다.
- 데이터 분류 백서에 나열된 데이터 분류를 지원하는 다른 AWS 서비스 및 기능을 평가하는 것이 좋습니다.

리소스

관련 문서:

- Amazon Macie 시작하기
- Amazon Macie을 사용한 자동화된 데이터 검색
- 태그 정책 시작하기
- PII 엔터티 감지

관련 블로그:

- Amazon Macie를 사용하여 S3 버킷에서 민감한 정보를 미리 보는 방법
- Performing automated sensitive data discovery with Amazon Macie(Amazon Macie를 사용하여 자동화된 민감한 데이터 검색 수행)
- Common techniques to detect PHI and PII data using AWS Services(AWS 서비스를 사용하여 PHI 및 PII 데이터를 감지하는 일반 기술)
- Detecting and redacting PII using Amazon Comprehend(Amazon Comprehend를 사용하여 PII 감지 및 교정)

- Securing resource tags used for authorization using a service control policy in AWS Organizations(AWS Organizations에서 서비스 제어 정책을 사용하여 권한 부여에 사용되는 리소스 태그 보호)
- Enabling data classification for Amazon RDS database with Macie(Macie를 사용하여 Amazon RDS 데이터베이스에 대한 데이터 분류 활성화)
- Detecting sensitive data in DynamoDB with Macie(Macie를 사용하여 DynamoDB에서 민감한 데이터 감지)
-

관련 동영상:

- Event-driven data security using Amazon Macie(Amazon Macie를 사용한 이벤트 기반 데이터 보안)
- Amazon Macie for data protection and governance(데이터 보호 및 거버넌스를 위한 Amazon Macie)
- Fine-tune sensitive data findings with allow lists(허용 목록을 사용하여 민감한 데이터 조사 결과 미세 조정)

SEC07-BP02 데이터 보호 제어 정의

분류 수준에 따라 데이터를 보호합니다. 예를 들어 관련 권장 사항을 사용하여 공개용으로 분류된 데이터를 보호하면서, 추가 제어 기능을 통해 민감한 데이터를 보호합니다.

리소스 태그, 중요도별(각 주의, 영역, 커뮤니티별로도 가능) 개별 AWS 계정, IAM 정책, AWS Organizations SCP, AWS Key Management Service(AWS KMS), AWS CloudHSM을 사용함으로써 데이터 분류 및 암호화를 통한 보호를 위한 정책을 정의하고 구현할 수 있습니다. 예를 들어 기밀 데이터를 처리하는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스 또는 매우 중요한 데이터가 포함된 S3 버킷을 사용하는 프로젝트가 있는 경우 `Project=ABC` 태그를 지정할 수 있습니다. 직속 팀만이 프로젝트 코드의 의미를 알고 있으므로 속성 기반 액세스 제어를 사용하는 것이 가능합니다. 적절한 서비스만 보안 메커니즘을 통해 중요한 콘텐츠에 액세스할 수 있도록 키 정책 및 부여를 통해 AWS KMS 암호화 키 액세스 수준을 정의할 수 있습니다. 태그를 기반으로 권한 부여 결정을 내리는 경우, 태그에 대한 권한이 AWS Organizations의 태그 정책을 사용하여 적절하게 정의되었는지 확인해야 합니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 데이터 식별 및 분류 스키마 정의: 데이터 식별 및 분류를 수행하여 저장한 데이터 유형과 잠재적 영향, 그리고 이 데이터에 액세스할 수 있는 사용자를 평가합니다.
  - AWS 설명서
- 사용 가능한 AWS 제어 기능 파악: 기존에 사용 중이거나 앞으로 사용하려는 AWS 서비스에 대한 보안 제어 옵션을 알아봅니다. 서비스의 설명서에 보안 섹션이 있는 경우가 많습니다.
  - AWS 설명서
- AWS 규정 준수 리소스 파악: AWS에서 제공하는 리소스를 파악합니다.
  - https://aws.amazon.com/compliance/

리소스

관련 문서:

- AWS 설명서
- 데이터 분류 백서
- Amazon Macie 시작하기
- 누락된 텍스트

관련 동영상:

- 새로운 Amazon Macie 소개

SEC07-BP03 식별 및 분류 자동화

데이터 식별 및 분류를 자동화하면 올바른 제어를 구현하는 데 도움이 될 수 있습니다. 사람이 직접 액세스하도록 하는 대신 자동화를 사용하면 인적 오류와 노출의 위험이 줄어듭니다. 기계 학습을 사용하여 AWS에서 민감한 데이터를 자동으로 검색, 분류 및 보호하는 Amazon Macie와 같은 도구를 고려해 보아야 합니다. Amazon Macie는 개인 식별 정보(PII) 또는 지적 재산과 같은 민감한 데이터를 인식하고, 이러한 데이터가 어떻게 액세스되고 이동되는지 파악할 수 있는 대시보드 및 알림을 제공합니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- Amazon Simple Storage Service(Amazon S3) 인벤토리 사용: Amazon S3 인벤토리는 객체의 복제 및 암호화 상태를 감사하고 보고하는 데 사용할 수 있는 도구 중 하나입니다.
  - Amazon S3 인벤토리
- Amazon Macie 고려: Amazon Macie는 기계 학습을 사용하여 Amazon S3에 저장된 데이터를 자동으로 검색하고 분류합니다.
  - Amazon Macie

리소스

관련 문서:

- Amazon Macie
- Amazon S3 인벤토리
- 데이터 분류 백서
- Amazon Macie 시작하기

관련 동영상:

- 새로운 Amazon Macie 소개

SEC07-BP04 데이터 수명 주기 관리 정의

정의된 수명 주기 전략은 중요도는 물론 법률 및 조직 요구 사항을 기반으로 해야 합니다. 데이터 보존 기간, 데이터 폐기 프로세스, 데이터 액세스 관리, 데이터 변환, 데이터 공유 등의 측면을 고려해야 합니다. 데이터 분류 방법론을 선택할 때는 사용 가능성과 액세스 권한을 적절하게 절충해야 합니다. 또한 여러 액세스 수준, 그리고 각 수준에 대해 안전하면서도 쉽게 사용할 수 있는 방식을 구현하기 위한 여러 가지 방법도 고려해야 합니다. 항상 심층 방어 방식을 사용하고 데이터 그리고 데이터 변환, 삭제 또는 복사 메커니즘에 사람이 접근하는 것을 줄입니다. 예를 들어 사용자에게 애플리케이션에 대한 강력한 인증을 요구하고, 필요한 액세스 권한을 사용자보다는 애플리케이션에 부여함으로써 '한 발 떨어져서 작업'을 수행하도록 합니다. 또한 사용자가 신뢰할 수 있는 네트워크 경로에서 애플리케이션에 액세스하며, 암호 해독 키 액세스 권한이 있어야 하도록 설정합니다. 사용자에게 데이터 직접 액세스 권한을 제공하기보다는 대시보드 및 자동화된 보고와 같은 도구를 사용하여 데이터의 정보를 제공하는 것이 좋습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 데이터 유형 파악: 워크로드에서 저장하거나 처리하는 데이터 유형을 식별합니다. 이러한 데이터로는 텍스트, 이미지, 이진 데이터베이스 등이 있습니다.

리소스

관련 문서:

- 데이터 분류 백서
- Amazon Macie 시작하기

관련 동영상:

- 새로운 Amazon Macie 소개

SEC 8 저장된 데이터는 어떻게 보호합니까?

무단 액세스 또는 취급 부주의의 위험을 줄이기 위해 여러 제어 기능을 구현하여 저장된 데이터를 보호합니다.

모범 사례

- SEC08-BP01 보안 키 관리 구현
- SEC08-BP02 저장 시 암호화 적용
- SEC08-BP03 저장 데이터 보호 자동화
- SEC08-BP04 액세스 제어 적용
- SEC08-BP05 사람들이 데이터에 쉽게 액세스할 수 없도록 하는 메커니즘 사용

SEC08-BP01 보안 키 관리 구현

키의 저장, 교체, 액세스 제어를 포함하는 암호화 방식을 정의함으로써 권한이 없는 사용자로부터 콘텐츠를 보호하고 권한이 있는 사용자에게도 불필요한 콘텐츠 노출이 발생하는 것을 방지할 수 있습니다. AWS Key Management Service(AWS KMS)를 사용하면 암호화 키를 관리하고 다양한 AWS 서비스와 통합할 수 있습니다. 이 서비스는 AWS KMS 키에 내구성과 보안성이 뛰어난 중복 스토리지를 제공합니다. 키 별칭과 키 수준 정책을 정의할 수 있습니다. 정책을 사용하면 키 관리자와 키 사용자를 정의할 수 있습니다. 또한 AWS CloudHSM은 AWS 클라우드에서 고유한 암호화 키를 쉽게 생성하여 사용할

수 있는 클라우드 기반 하드웨어 보안 모듈(HSM)입니다. HSM을 사용하면 FIPS 140-2 수준 3 확인된 HSM을 통해 데이터 보안 관련 회사, 계약 및 규정 준수 요구 사항을 충족할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- AWS KMS 구현: AWS KMS를 사용하면 키 생성과 관리가 쉬워지며 광범위한 AWS 서비스 및 애플리케이션 내에서 암호화 사용을 제어할 수 있습니다. AWS KMS는 키 보호를 위해 FIPS 140-2 인증 하드웨어 보안 모듈을 사용하는 안전하고 복원력이 높은 서비스입니다.
  - 시작하기: AWS Key Management Service(AWS KMS)
- AWS Encryption SDK 고려: 애플리케이션이 클라이언트 측 데이터를 암호화해야 하는 경우 AWS KMS가 통합된 AWS Encryption SDK를 사용합니다.
  - AWS Encryption SDK

리소스

관련 문서:

- AWS Key Management Service
- AWS 암호화 서비스 및 도구
- 시작하기: AWS Key Management Service(AWS KMS)
- 암호화를 사용하여 Amazon S3 데이터 보호

관련 동영상:

- AWS에서 암호화가 작동하는 방식
- AWS에서 블록 스토리지 보호

SEC08-BP02 저장 시 암호화 적용

저장 데이터에 대한 암호화 사용을 적용해야 합니다. 암호화는 무단 액세스 또는 우발적 공개의 경우 민감한 데이터의 기밀성을 유지합니다.

원하는 결과: 프라이빗 데이터는 저장 시 기본적으로 암호화되어야 합니다. 암호화는 데이터의 기밀성을 유지하는 데 도움이 되며 의도적이거나 의도치 않은 데이터 공개 또는 유출에 대한 추가 보호 계층

을 제공합니다. 암호화된 데이터는 먼저 데이터의 암호화를 해제하지 않고는 읽거나 액세스할 수 없습니다. 암호화되지 않은 상태로 저장된 모든 데이터는 인벤토리를 만들고 제어해야 합니다.

일반적인 안티 패턴:

- 기본적으로 암호화 구성을 사용하지 않습니다.
- 복호화 키에 지나치게 관대한 액세스를 제공합니다.
- 암호화 및 복호화 키의 사용을 모니터링하지 않습니다.
- 데이터를 암호화되지 않은 상태로 저장합니다.
- 데이터 용도, 유형 및 분류에 관계없이 모든 데이터에 동일한 암호화 키를 사용합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

워크로드 내의 데이터 분류에 암호화 키를 매핑합니다. 이 접근 방식은 데이터에 단일 암호화 키 또는 매우 적은 수의 암호화 키를 사용할 때 지나치게 관대한 액세스로부터 데이터를 보호하는 데 도움이 됩니다(SEC07-BP01 워크로드 안에서 데이터 식별 참조).

AWS Key Management Service(AWS KMS)는 많은 AWS 서비스와 통합되어 저장 데이터를 보다 쉽게 암호화할 수 있습니다. 예를 들어 Amazon Simple Storage Service(Amazon S3)의 경우 버킷에 기본 암호화를 설정하여 새 객체가 자동으로 암호화되도록 하면 됩니다. AWS KMS를 사용할 때 데이터를 얼마나 엄격하게 제한해야 하는지 고려합니다. 기본 및 서비스 제어 AWS KMS 키는 사용자를 대신하여 AWS에서 관리하고 사용합니다. 기본 암호화 키에 대한 세분화된 액세스 권한이 필요한 민감한 데이터의 경우 고객 관리형 키(CMK)를 고려합니다. 키 정책을 사용하여 교체 및 액세스 관리를 포함하여 CMK를 완전히 제어할 수 있습니다.

또한 Amazon Elastic Compute Cloud(Amazon EC2) 및 Amazon S3는 기본 암호화 설정을 통해 암호화를 기본적으로 적용하도록 지원합니다. AWS Config 규칙를 사용하여 Amazon Elastic Block Store(Amazon EBS) 볼륨, Amazon Relational Database Service(Amazon RDS) 인스턴스 및 Amazon S3 버킷에 암호화를 사용하고 있는지 자동으로 확인할 수 있습니다.

AWS는 또한 클라이언트측 암호화 옵션을 제공하므로 데이터를 클라우드에 업로드하기 전에 암호화할 수 있습니다. AWS Encryption SDK는 봉투 암호화를 사용하여 데이터를 암호화하는 방법을 제공합니다. 래핑 키를 제공하면 AWS Encryption SDK가 암호화하는 각 데이터 객체에 대해 고유한 데이터 키를 생성합니다. 관리형 단일 테넌트 하드웨어 보안 모듈(HSM)이 필요한 경우 AWS CloudHSM을 고려합니다. AWS CloudHSM을 사용하면 FIPS 140-2 레벨 3 검증 HSM에서 암호화 키를 생성, 가져오기 및 관리할 수 있습니다. AWS CloudHSM의 일부 사용 사례에는 인증 기관(CA) 발급을 위한 프라이빗

키 보호와 Oracle 데이터베이스용 투명한 데이터 암호화(TDE) 활성화가 포함됩니다. AWS CloudHSM 클라이언트 SDK는 데이터를 AWS에 업로드하기 전에 AWS CloudHSM에 저장된 키를 사용하여 데이터 클라이언트측을 암호화할 수 있는 소프트웨어를 제공합니다. Amazon DynamoDB Encryption Client를 사용하면 DynamoDB 테이블에 업로드하기 전에 항목을 암호화하고 서명할 수도 있습니다.

구현 단계

- Amazon S3에 저장 시 암호화 적용: Amazon S3 버킷 기본 암호화를 구현합니다.

  새 Amazon EBS 볼륨에 기본 암호화 구성: Amazon EBS에서 제공하는 기본 키 또는 사용자가 생성한 키를 사용하는 옵션을 통해 새로 생성되는 모든 AWS 볼륨이 암호화된 형식으로 생성되도록 지정합니다.

  암호화된 Amazon Machine Image(AMI) 구성: 암호화가 활성화된 기존 AMI를 복사하면 루트 볼륨과 스냅샷이 자동으로 암호화됩니다.

  Amazon RDS 암호화 구성: 암호화 옵션을 사용하여 저장 시 Amazon RDS 데이터베이스 클러스터 및 스냅샷에 대한 암호화를 구성합니다.

  각 데이터 분류를 위해 적절한 보안 주체에 대한 액세스를 제한하는 정책으로 AWS KMS 키 생성 및 구성: 예를 들어 프로덕션 데이터 암호화를 위한 하나의 AWS KMS 키와 개발 또는 테스트 데이터 암호화를 위한 다른 키를 생성합니다. 다른 AWS 계정에 키 액세스를 제공할 수도 있습니다. 개발 및 프로덕션 환경에 대해 서로 다른 계정을 사용하는 것이 좋습니다. 프로덕션 환경에서 개발 계정의 아티팩트를 복호화해야 하는 경우 개발 아티팩트를 암호화하는 데 사용되는 CMK 정책을 편집하여 프로덕션 계정에 해당 아티팩트를 복호화할 수 있는 기능을 제공할 수 있습니다. 그러면 프로덕션 환경에서 프로덕션에 사용하기 위해 복호화된 데이터를 수집할 수 있습니다.

  추가 AWS 서비스에서 암호화 구성: 사용하는 다른 AWS 서비스의 경우 해당 서비스의 보안 문서를 검토하여 서비스의 암호화 옵션을 결정합니다.

리소스

관련 문서:

- AWS 암호화 도구
- AWS 설명서
- AWS Encryption SDK
- AWS KMS 암호화 세부 정보 백서
- AWS Key Management Service

- AWS 암호화 서비스 및 도구
- Amazon EBS 암호화
- Amazon EBS 볼륨의 기본 암호화
- Amazon RDS 리소스 암호화
- Amazon S3 버킷에 기본 암호화를 사용하려면 어떻게 해야 하나요?
- 암호화를 사용하여 Amazon S3 데이터 보호

관련 동영상:

- How Encryption Works in AWS(AWS에서 암호화가 작동하는 방식)
- Securing Your Block Storage on AWS(AWS에서 블록 스토리지 보호)

SEC08-BP03 저장 데이터 보호 자동화

자동화된 도구를 사용하여 저장된 데이터 제어를 지속적으로 검증하고 적용합니다. 예를 들어 암호화된 스토리지 리소스만 있는지 확인합니다. 다음을 사용하여 모든 EBS 볼륨이 암호화되었는지 검증을 자동화 할 수 있습니다. AWS Config 규칙참조. AWS Security Hub 를 사용하여 보안 표준을 기준으로 자동화된 검사를 통해 몇 가지 제어의 유효성을 확인할 수도 있습니다. 또한 AWS Config 규칙는 자동으로 규정 미준수 리소스를 수정할 수 있습니다..

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

저장된 데이터 란 워크로드의 어느 기간에서든지 비휘발성 스토리지에 지속되는 모든 데이터를 의미합니다. 여기에는 블록 스토리지, 객체 스토리지, 데이터베이스, 아카이브, IoT 디바이스 그리고 데이터가 지속되는 모든 기타 스토리지 미디어가 포함됩니다. 저장된 데이터를 보호하여 암호화 및 적절한 액세스 제어가 구현될 경우 무단 액세스 위험이 감소합니다.

저장 시 암호화 적용: 데이터를 저장할 때 반드시 암호화를 사용하도록 해야 합니다. AWS KMS는 여러 AWS 서비스와 원활하게 통합되므로 모든 저장 데이터를 쉽게 암호화할 수 있습니다. 예를 들어 Amazon Simple Storage Service(Amazon S3)의 경우 버킷에 기본 암호화 를 설정하여 새 객체가 모두 자동으로 암호화되도록 하면 됩니다. 또한 Amazon EC2 및 Amazon S3 는 기본 암호화 설정을 통해 암호화를 기본적으로 적용하도록 지원합니다. 전용 인프라에서 AWS Managed Config Rules 를 사용하여 예를 들면 다음에 암호화를 사용하고 있는지 자동으로 검사할 수 있습니다. EBS 볼륨, Amazon Relational Database Service(Amazon RDS) 인스턴스및 Amazon S3 버킷.

리소스

관련 문서:

- AWS 암호화 도구
- AWS Encryption SDK

관련 동영상:

- AWS에서 암호화가 작동하는 방식
- AWS에서 블록 스토리지 보호

SEC08-BP04 액세스 제어 적용

저장 데이터를 보호하려면 격리 및 버전 관리와 같은 메커니즘을 사용하여 액세스 제어를 적용하고 최소 권한 원칙을 적용합니다. 데이터에 대한 퍼블릭 액세스 권한 부여를 방지합니다.

원하는 결과: 권한이 부여된 사용자만 알아야 할 필요가 있을 때 데이터에 액세스할 수 있는지 확인합니다. 정기적인 백업 및 버전 관리를 통해 데이터를 보호하여 의도적이거나 우발적인 데이터 수정 또는 삭제를 방지합니다. 중요한 데이터를 다른 데이터와 분리하여 기밀성과 데이터 무결성을 보호합니다.

일반적인 안티 패턴:

- 민감도 요구 사항이 다르거나 분류가 다른 데이터를 함께 저장합니다.
- 복호화 키에 지나치게 관대한 권한을 사용합니다.
- 데이터를 잘못 분류합니다.
- 중요한 데이터의 자세한 백업을 유지하지 않습니다.
- 프로덕션 데이터에 대한 지속적인 액세스를 제공합니다.
- 데이터 액세스를 감사하거나 정기적으로 권한을 검토하지 않습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 낮음

구현 가이드

액세스(최소 권한 사용), 격리, 버전 관리를 포함한 여러 제어를 사용하여 저장 데이터를 보호할 수 있습니다. 데이터에 대한 액세스는 AWS CloudTrail과 같은 탐지 메커니즘과 Amazon Simple Storage

Service(Amazon S3) 액세스 로그와 같은 서비스 수준 로그를 사용하여 감사해야 합니다. 공개적으로 액세스할 수 있는 데이터의 인벤토리를 만들고 시간이 지남에 따라 공개적으로 사용 가능한 데이터의 양을 줄이기 위한 계획을 수립해야 합니다.

Amazon S3 Glacier 저장소 잠금 및 Amazon S3 객체 잠금은 Amazon S3의 객체에 대한 필수 액세스 제어를 제공합니다. 규정 준수 옵션으로 저장소 정책을 잠그면 루트 사용자도 잠금이 만료되기 전까지는 변경할 수 없습니다.

구현 단계

- 액세스 제어 적용: 암호화 키 액세스를 포함하여 최소 권한을 사용하는 액세스 제어를 적용합니다.
- 다양한 분류 수준에 따라 데이터 분리: 데이터 분류 수준에 서로 다른 AWS 계정를 사용하고, AWS Organizations를 사용하여 해당 계정을 관리합니다.
- AWS Key Management Service(AWS KMS) 정책 검토: AWS KMS 정책에서 부여된 액세스 수준을 검토합니다.
- Amazon S3 버킷 및 객체 권한 검토: S3 버킷 정책에서 부여된 액세스 수준을 주기적으로 검토합니다. 공개적으로 읽을 수 있는 버킷이나 쓸 수 있는 버킷을 사용하지 않는 것이 모범 사례입니다. AWS Config를 사용하여 공개적으로 사용 가능한 버킷을 감지하고 Amazon CloudFront를 사용하여 Amazon S3에서 콘텐츠를 제공하는 것이 좋습니다. 퍼블릭 액세스를 허용하면 안 되는 버킷은 퍼블릭 액세스가 되지 않도록 적절히 구성되어 있는지 확인합니다. 기본적으로 모든 S3 버킷은 프라이빗 버킷이며 명시적으로 액세스 권한이 부여된 사용자만 액세스할 수 있습니다.
- AWS IAM Access Analyzer 사용: IAM Access Analyzer는 Amazon S3 버킷을 분석하고 S3 정책이 외부 엔터티에 대한 액세스 권한을 부여할 때 조사 결과를 생성합니다.
- 적절한 경우 Amazon S3 버전 관리 및 객체 잠금을 활성화합니다.
- Amazon S3 인벤토리 사용: Amazon S3 인벤토리를 사용하면 S3 객체의 복제 및 암호화 상태를 감사하고 보고할 수 있습니다.
- Amazon EBS 및 AMI 공유 권한 검토: 권한을 공유하면 워크로드 외부의 AWS 계정과 이미지 및 볼륨을 공유할 수 있습니다.
- AWS Resource Access Manager 공유를 주기적으로 검토하여 리소스를 계속 공유해야 하는지 여부를 결정합니다. Resource Access Manager를 사용하면 Amazon VPC 내에서 AWS 네트워크 방화벽 정책, Amazon Route 53 확인자 규칙, 서브넷과 같은 리소스를 공유할 수 있습니다. 공유 리소스를 정기적으로 감사하고 더 이상 공유할 필요가 없는 리소스 공유를 중지합니다.

리소스

관련 모범 사례:

- SEC03-BP01 액세스 요구 사항 정의
- SEC03-BP02 최소 권한 액세스 부여

관련 문서:

- AWS KMS 암호화 세부 정보 백서
- Amazon S3 리소스에 대한 액세스 권한 관리 소개
- AWS KMS 리소스에 대한 액세스 권한 관리 개요
- AWS Config 규칙
- Amazon S3 + Amazon CloudFront: 클라우드 최적의 조합
- 버전 관리 사용
- Amazon S3 객체 잠금을 사용한 객체 잠금
- Amazon EBS 스냅샷 공유
- 공유 AMI
- Amazon S3에서 단일 페이지 애플리케이션 호스팅

관련 동영상:

- Securing Your Block Storage on AWS(AWS에서 블록 스토리지 보호)

SEC08-BP05 사람들이 데이터에 쉽게 액세스할 수 없도록 하는 메커니즘 사용

정상적인 운영 상황에서 모든 사용자가 민감한 데이터와 시스템에 직접 액세스하지 못하도록 합니다. 예를 들어 변경 관리 워크플로를 사용하여 직접 액세스를 허용하는 대신 도구나 Bastion 호스트를 사용하여 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 관리할 수 있습니다. 이렇게 하려면 AWS Systems Manager Automation을 사용할 수 있으며, 이 서비스는 작업을 수행할 때 사용하는 단계가 포함된 자동화 문서 를 사용합니다. 이러한 문서는 소스 제어에 저장되고, 실행하기 전에 피어 검토를 받고, 철저한 테스트를 받아 셸 액세스와 비교해 위험을 최소화할 수 있습니다. 비즈니스 사용자에게는 데이터 스토어에 대한 직접적인 액세스 대신 대시보드를 제공하여 쿼리를 실행하게 할 수 있습니다. CI/CD 파이프라인이 사용되지 않는 경우, 정상적으로 비활성화된 브레이크-글라스 액세스 메커니즘을 적절하게 제공하기 위해 필요한 제어 및 프로세스를 결정합니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 사람들이 데이터에 쉽게 액세스할 수 없도록 하는 메커니즘 구현: 이러한 메커니즘에는 직접 쿼리하는 대신 Amazon QuickSight와 같은 대시보드를 사용하여 사용자에게 데이터를 표시하는 방식이 포함됩니다.
  - Amazon QuickSight
- 구성 관리 자동화: 구성 관리 서비스 또는 도구를 사용하여 원격으로 작업을 수행하고 보안 구성을 자동으로 적용하고 확인합니다. 배스천 호스트를 사용하거나 EC2 인스턴스에 직접 액세스하지 않습니다.
  - AWS Systems Manager
  - AWS CloudFormation
  - AWS 기반 AWS CloudFormation 템플릿용 CI/CD 파이프라인

리소스

관련 문서:

- AWS KMS 암호화 세부 정보 백서

관련 동영상:

- How Encryption Works in AWS(AWS에서 암호화가 작동하는 방식)
- Securing Your Block Storage on AWS(AWS에서 블록 스토리지 보호)

SEC 9 전송 중인 데이터는 어떻게 보호합니까?

무단 액세스 또는 손실의 위험을 줄이기 위해 여러 제어 기능을 구현하여 전송 중인 데이터를 보호합니다.

모범 사례

- SEC09-BP01 보안 키 및 인증서 관리 구현
- SEC09-BP02 전송 중 데이터 암호화 적용
- SEC09-BP03 무단 데이터 침입 탐지 자동화
- SEC09-BP04 네트워크 통신 인증

SEC09-BP01 보안 키 및 인증서 관리 구현

암호화 키와 인증서를 안전하게 저장하고 엄격하게 액세스 제어를 통해 적절한 간격으로 교체합니다. 이를 위한 가장 좋은 방법은 AWS Certificate Manager(ACM) 같은 관리형 서비스를 사용하는 것입니다. AWS 서비스 및 연결된 내부 리소스에 사용할 공인 및 사설 TLS(Transport Layer Security) 인증서를 손쉽게 프로비저닝, 관리 및 배포할 수 있습니다. TLS 인증서는 네트워크 통신을 보호하고 인터넷에서는 웹 사이트 그리고 프라이빗 네트워크에서 리소스의 ID를 설정하기 위해 사용됩니다. ACM은 Elastic Load Balancer(ELB), AWS 배포, API Gateway의 API 등 AWS 리소스와 통합되며 자동 인증서 갱신도 처리합니다. ACM을 사용하여 사설 루트 CA를 배포하는 경우 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스, 컨테이너 등에서 사용할 수 있도록 인증서와 비공개 키가 제공될 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 보안 키 및 인증서 관리 구현: 정의된 보안 키 및 인증서 관리 솔루션을 구현합니다.
  - AWS Certificate Manager
  - AWS에서 전체 프라이빗 인증서 인프라를 호스팅하고 관리하는 방법
- 보안 프로토콜 구현: 데이터 변조나 손실 위험을 줄일 수 있도록 전송 계층 보안(TLS) 또는 IPsec 등 인증 및 기밀성 유지 기능을 제공하는 보안 프로토콜을 사용합니다. 사용 중인 서비스와 관련한 프로토콜 및 보안은 AWS 설명서를 참조하세요.

리소스

관련 문서:

- AWS 설명서

SEC09-BP02 전송 중 데이터 암호화 적용

조직, 법률 및 규정 준수 요구 사항을 충족할 수 있도록 조직의 정책, 규제 의무 및 표준에 따라 정의된 암호화 요구 사항을 적용합니다. 민감한 데이터를 Virtual Private Cloud(VPC) 외부로 전송할 때 암호화된 프로토콜만 사용합니다. 암호화는 데이터가 신뢰할 수 없는 네트워크로 전송되는 경우에도 데이터 기밀성을 유지하는 데 도움이 됩니다.

원하는 결과: 모든 데이터는 보안 TLS 프로토콜 및 암호 그룹을 사용하여 전송 중 암호화되어야 합니다. 데이터에 대한 무단 액세스를 완화하려면 리소스와 인터넷 간의 네트워크 트래픽을 암호화해야 합

니다. 가능한 경우 내부 AWS 환경 내의 네트워크 트래픽은 TLS를 사용하여 암호화해야 합니다. AWS 내부 네트워크는 기본적으로 암호화되며 권한이 없는 당사자가 트래픽을 생성하는 모든 리소스(예: Amazon EC2 인스턴스 및 Amazon ECS 컨테이너)에 대한 액세스 권한을 획득하지 않는 한 VPC 내의 네트워크 트래픽을 스푸핑하거나 스니핑할 수 없습니다. IPsec 가상 프라이빗 네트워크(VPN)를 사용하여 네트워크 간 트래픽을 보호하는 것이 좋습니다.

일반적인 안티 패턴:

- 사용 중단된 버전의 SSL, TLS 및 암호 그룹 구성 요소(예: SSL v3.0, 1024비트 RSA 키 및 RC4 암호)를 사용합니다.
- 퍼블릭 리소스에서 암호화되지 않은(HTTP) 트래픽을 허용합니다.
- 만료되기 전에 X.509 인증서를 모니터링하고 교체하지 않습니다.
- TLS에 자체 서명된 X.509 인증서를 사용합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

AWS 서비스는 통신에 TLS를 사용하는 HTTPS 엔드포인트를 제공하여 AWS API와 통신할 때 전송 중 암호화 기능을 제공합니다. HTTP와 같은 안전하지 않은 프로토콜은 보안 그룹을 사용하여 VPC에서 감사 및 차단할 수 있습니다. 또한 HTTP 요청은 Amazon CloudFront 또는 Application Load Balancer에서 HTTPS로 자동 리디렉션될 수 있습니다. 컴퓨팅 리소스를 완전하게 제어하여 서비스 간에 전송 중 암호화를 구현할 수 있습니다. 또한 외부 네트워크 또는 AWS Direct Connect로부터 특정 VPC로의 VPN 연결을 사용하여 트래픽을 쉽게 암호화할 수도 있습니다. 2023년 6월에 AWS에서 TLS 1.0 및 1.1 사용을 중단하므로 클라이언트가 TLS 1.2 이상을 사용하여 AWS API를 호출하는지 확인합니다. 특별한 요구 사항이 있는 경우 AWS Marketplace에서 타사 솔루션을 사용할 수 있습니다.

구현 단계

- 전송 중 암호화 적용: 정의된 암호화 요구 사항은 최신 표준 및 모범 사례를 토대로 하고 보안 프로토콜만 허용해야 합니다. 예를 들어 Application Load Balancer 또는 Amazon EC2 인스턴스로의 HTTPS 프로토콜을 허용하는 보안 그룹만 구성합니다.
- 엣지 서비스에서 보안 프로토콜 구성: Amazon CloudFront로 HTTPS를 구성하고 보안 태세 및 사용 사례에 적합한 보안 프로필을 사용합니다.
- 외부 연결에 VPN 사용: 데이터 프라이버시와 무결성을 모두 제공할 수 있도록 지점 간 또는 네트워크 간 연결에 IPsec VPN을 사용하는 것이 좋습니다.

- 로드 밸런서에서 보안 프로토콜 구성: 리스너에 연결할 클라이언트가 지원하는 가장 강력한 암호 그룹을 제공하는 보안 정책을 선택합니다. Application Load Balancer에 대한 HTTPS 리스너를 생성합니다..
- Amazon Redshift에서 보안 프로토콜 구성: 클러스터가 보안 소켓 계층(SSL) 또는 전송 계층 보안(TLS) 연결을 요구하도록 구성합니다.
- 보안 프로토콜 구성: AWS 서비스 설명서를 검토하여 전송 중 암호화 기능을 확인합니다.
- Amazon S3 버킷에 업로드할 때 보안 액세스 구성: Amazon S3 버킷 정책 제어를 사용하여 데이터에 대한 보안 액세스를 적용합니다.
- AWS Certificate Manager 사용 고려: ACM을 사용하면 AWS 서비스와 함께 사용할 퍼블릭 TLS 인증서를 프로비저닝, 관리 및 배포할 수 있습니다.
- 프라이빗 PKI 요구 사항에 AWS Private Certificate Authority 사용 고려: AWS Private CA를 사용하면 프라이빗 인증 기관(CA) 계층 구조를 생성하여 암호화된 TLS 채널을 생성하는 데 사용할 수 있는 최종 엔터티 X.509 인증서를 발급할 수 있습니다.

리소스

관련 문서:

- AWS 설명서
- CloudFront와 함께 HTTPS 사용
- AWS Virtual Private Network를 사용하여 원격 네트워크에 VPC 연결
- Application Load Balancer에 대한 HTTPS 리스너 생성
- 자습서: Amazon Linux 2에서 SSL/TLS 구성
- SSL/TLS를 사용하여 DB 인스턴스에 대한 연결 암호화
- 연결을 위한 보안 옵션 구성

SEC09-BP03 무단 데이터 침입 탐지 자동화

Amazon GuardDuty 등의 도구를 사용하여 의심스러운 활동 또는 정의된 경계 외부로 데이터를 이전하려는 시도를 자동으로 감지합니다. 예를 들어, GuardDuty는 다음을 사용하여 일반적이지 않은 Amazon Simple Storage Service(Amazon S3) 읽기 활동을 감지할 수 있습니다. Exfiltration:S3/AnomalousBehavior 결과. GuardDuty 외에도 네트워크 트래픽 정보를 캡처하는 Amazon VPC 흐름 로그를 Amazon EventBridge와 함께 사용하여 비정상적 연결(성공한 연결과 거부된 연결 모두)을 탐지할 수 있습니다. Amazon S3 Access Analyzer 는 Amazon S3 버킷에서 누가 어떤 데이터에 액세스할 수 있는지를 평가하는 데 도움이 될 수 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 무단 데이터 탐지 자동화: 도구 또는 감지 메커니즘을 사용하여 정의된 경계 외부로 데이터를 이동하려는 시도를 직접 감지합니다. 예를 들어 알 수 없는 호스트로 데이터를 복사하는 데이터베이스 시스템을 감지할 수 있습니다.
  - VPC 흐름 로그
- Amazon Macie 고려: Amazon Macie는 기계 학습 및 패턴 일치를 사용하여 AWS에서 민감한 데이터를 검색하고 보호하는 완전관리형 데이터 보안 및 데이터 개인 정보 보호 서비스입니다.
  - Amazon Macie

리소스

관련 문서:

- VPC 흐름 로그
- Amazon Macie

SEC09-BP04 네트워크 통신 인증

TLS(전송 계층 보안) 또는 IPsec과 같은 인증을 지원하는 프로토콜을 사용하여 통신의 자격 증명을 확인합니다.

인증을 지원하는 네트워크 프로토콜을 사용하여 당사자 간 신뢰를 맺을 수 있습니다. 이것이 프로토콜에서 사용되는 암호화에 추가되어 통신이 변조되거나 가로채질 위험을 줄입니다. 인증을 구현하는 일반적인 프로토콜에는 많은 AWS 서비스에서 사용되는 TLS(Transport Layer Security)와 다음에서 사용되는 IPsec가 포함됩니다. AWS Virtual Private Network(AWS VPN).

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 보안 프로토콜 구현: 데이터 변조나 손실 위험을 줄일 수 있도록 TLS 또는 IPsec 등 인증 및 기밀성 유지 기능을 제공하는 보안 프로토콜을 사용합니다. 사용하는 서비스와 관련한 프로토콜 및 보안은 AWS 설명서 를 참조하세요.

리소스

관련 문서:

- AWS 설명서

사고 대응

질문

- SEC 10 인시던트를 어떻게 예상하고 대응하며 어떻게 사후 복구합니까?

SEC 10 인시던트를 어떻게 예상하고 대응하며 어떻게 사후 복구합니까?

조직의 업무 중단을 최소화할 수 있도록 보안 인시던트를 제때 효고적으로 조사 및 대응하고 사후 복구하려면 철저한 준비가 필요합니다.

모범 사례

- SEC10-BP01 주요 직원과 외부 리소스 파악
- SEC10-BP02 인시던트 관리 계획 개발
- SEC10-BP03 포렌식 역량 확보
- SEC10-BP04 억제 기능 자동화
- SEC10-BP05 액세스 권한 사전 프로비저닝
- SEC10-BP06 도구 사전 배포
- SEC10-BP07 게임 데이 진행

SEC10-BP01 주요 직원과 외부 리소스 파악

조직이 인시던트에 대응하는 데 도움이 될 수 있는 내/외부 직원, 리소스, 법적 의무를 파악합니다.

클라우드에서 인시던트 대응 방식을 다른 팀(예: 법률 자문, 리더십, 비즈니스 이해관계자, AWS Support Services 등)과 함께 정의할 때는 주요 직원, 이해관계자 및 관련 연락처를 파악해야 합니다. 종속성을 줄이고 응답 시간을 단축하려면 사용하는 서비스에 대해 팀, 전문 보안팀, 응답자를 교육하고 실습 기회를 제공해야 합니다.

외부의 전문 지식 그리고 대응 능력을 강화할 수 있는 다른 관점을 제공할 수 있는 외부 AWS 보안 파트너를 찾는 것이 좋습니다. 신뢰할 수 있는 보안 파트너는 익숙하지 않은 잠재적 위험 또는 위협을 식별하는 데 도움을 줄 수 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 조직의 주요 직원 파악: 인시던트 대응 및 인시던트 이후의 복구에 참여해야 하는 조직 내의 직원 연락처 목록을 유지 관리합니다.
- 외부 파트너 파악: 필요한 경우 인시던트 대응 및 인시던트 이후의 복구를 지원할 수 있는 외부 파트너와 협력합니다.

리소스

관련 문서:

- AWS 인시던트 대응 안내서

관련 동영상:

- AWS 환경에서 보안 인시던트 준비 및 대응

관련 예시:

SEC10-BP02 인시던트 관리 계획 개발

인시던트 대응을 위해 작성해야 할 첫 번째 문서는 인시던트 대응 계획입니다. 인시던트 대응 계획은 인시던트 대응 프로그램 및 전략의 기초가 되도록 설계되었습니다.

이 모범 사례 확립의 이점: 철저하고 명확하게 정의된 인시던트 대응 프로세스를 개발하는 것은 성공적이고 확장 가능한 인시던트 대응 프로그램의 핵심입니다. 보안 이벤트가 발생하면 명확한 단계 및 워크플로가 적시에 대응하는 데 도움이 됩니다. 기존 인시던트 대응 프로세스가 이미 있을 수 있습니다. 현재 상태에 관계없이 인시던트 대응 프로세스를 정기적으로 업데이트, 반복, 테스트하는 것이 중요합니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 높음

구현 가이드

인시던트 관리 계획은 보안 인시던트의 잠재적 영향에 대한 대응, 완화 및 복구에 매우 중요합니다. 인시던트 관리 계획은 보안 인시던트를 적시에 파악하고 해결 및 대응하기 위한 구조화된 프로세스입니다.

클라우드에는 온프레미스 환경에서 볼 수 있는 수많은 동일한 운영 역할과 요구 사항이 있습니다. 인시던트 관리 계획을 수립할 때는 비즈니스 성과와 규정 준수 요구 사항에 가장 잘 맞는 대응 및 복구 전략을 고려하는 것이 중요합니다. 예를 들어, 미국 내 FedRAMP 규정을 준수하는 AWS에서 워크로드를 운영하는 경우 NIST SP 800-61 Computer Security Handling Guide(컴퓨터 보안 처리 안내서). 이와 유사하게 유럽 개인 식별 정보(PII) 데이터가 있는 워크로드를 운영하는 경우 유럽 연합 일반 데이터 보호 규정(GDPR)에 명시된 데이터 레지던스 관련 문제를 보호하고 대응할 수 있는 방법과 같은 시나리오를 고려합니다.

AWS에서 운영하는 워크로드에 대한 인시던트 관리 계획을 구축하는 경우, 인시던트 대응에 대한 심층 방어 방식을 구축하기 위해 AWS 공동 책임 모델로 시작합니다. 이 모델에서 AWS는 클라우드 자체의 보안을 관리하지만 클라우드 내에서 보안을 유지하는 것은 고객의 책임입니다. 즉, 고객은 구현을 선택하는 보안 제어에 대한 제어 권한을 보유하며 이에 대한 책임이 있습니다. 클라우드 중심 인시던트 관리 계획 구축에 대한 핵심 개념 및 기본 지침은 AWS 보안 인시던트 대응 안내서 에서 자세히 설명하고 있습니다.

효과적인 인시던트 관리 계획은 클라우드 운영 목표와 함께 끊임없이 반복되고 항상 최신 상태를 유지해야 합니다. 인시던트 관리 계획을 수립 및 발전시킬 때 아래에서 자세히 설명하는 구현 계획의 사용을 고려해 볼 수 있습니다.

구현 단계

역할과 책임 정의

보안 이벤트를 처리하려면 조직 간 규율과 행동 성향이 필요합니다. 조직 구조 내에는 인사(HR) 담당자, 경영진, 법무 담당자와 같이 인시던트 발생 시 책임이 있거나(Responsible) 책임을 지거나(Accountable) 자문을 받거나(Consulted) 최신 정보를 제공받는(Informed) 사람들이 많이 있어야 합니다. 이러한 역할 및 책임과 제3자가 개입해야 하는지 여부를 고려하십시오. 많은 지역에는 해야 할 일과 하지 말아야 할 일을 규정하는 현지 법률이 있습니다. 보안 대응 계획을 위해 Responsible, Accountable, Consulted, Informed(RACI) 차트를 작성하는 것이 불필요해 보일 수 있지만, 그렇게 하면 신속하고 직접적인 커뮤니케이션이 촉진되고 이벤트의 여러 단계에서 리더십의 윤곽을 명확하게 파악할 수 있습니다.

인시던트 발생 시 영향을 받는 애플리케이션 및 리소스의 소유자와 개발자를 포함하는 것이 중요합니다. 이들은 영향을 측정하는 데 도움이 되는 정보와 컨텍스트를 제공할 수 있는 주제 전문가(SME)이기 때문입니다. 개발자 및 애플리케이션 소유자의 인시던트 대응 전문 지식에 의존하기 전에 이들과 함께 연습하고 관계를 구축해야 합니다. 클라우드 관리자 또는 엔지니어와 같은 애플리케이션 소유자 또는 SME는 환경이 익숙하지 않거나 복잡하거나 대응자가 액세스할 수 없는 상황에서 조치를 취해야 할 수 있습니다.

마지막으로 신뢰할 수 있는 파트너는 추가적인 전문 지식과 가치 있는 조사를 제공할 수 있으므로 조사 또는 대응에 참여할 수 있습니다. 팀에 이러한 기술이 없다면 외부 담당자를 고용하여 도움을 받는 것이 좋습니다.

AWS 대응 팀 및 지원 이해

- AWS Support
  - AWS Support는 AWS 솔루션의 성공 및 운영 상태를 지원하는 도구와 전문 지식을 이용할 수 있는 다양한 플랜을 제공합니다. AWS 환경을 계획, 배포, 최적화하는 데 도움이 되는 기술 지원 및 추가 리소스가 필요한 경우 AWS 사용 사례에 가장 적합한 Support 플랜을 선택할 수 있습니다.
  - AWS 리소스에 영향을 미치는 문제에 대한 지원을 받으려면 AWS Management Console 관리 콘솔(로그인 필요)의 지원 센터를 중앙 연락 창구로 고려하십시오. AWS Support에 대한 액세스는 AWS Identity and Access Management으로 제어됩니다. AWS Support 기능에 액세스하는 방법에 대한 자세한 내용은 AWS Support 시작하기를참조하십시오.
- AWS 고객 인시던트 대응 팀(CIRT)
  - AWS 고객 인시던트 대응 팀(CIRT)은 24/7로 운영되는 전문 글로벌 AWS 팀으로 AWS 공동 책임 모델의고객 측에서 보안 이벤트가 진행되는 동안 고객을 지원합니다.
  - 고객을 지원할 때 AWS CIRT는 AWS에서의 활성 보안 이벤트 분류 및 복구를 지원합니다. 팀은 AWS 서비스 로그를 사용하여 근본 원인 분석을 지원하고 복구를 위한 권장 사항을 제공할 수 있습니다. 또한 향후 보안 이벤트를 방지하는 데 도움이 되는 보안 권장 사항 및 모범 사례를 제공할 수 있습니다.
  - AWS 고객은 AWS Support 사례를통해 AWS CIRT의 지원을 요청할 수 있습니다.
- DDoS 대응 지원
  - AWS는 AWS Shield를제공합니다. 이 서비스는 AWS에서 실행 중인 웹 애플리케이션을 보호하는 관리형 분산 서비스 거부(DDoS) 보호 서비스를 제공합니다. Shield는 애플리케이션 가동 중지 시간과 지연 시간을 최소화할 수 있는 상시 탐지 및 자동 인라인 방어 기능을 제공하므로 DDoS 보호 혜택을 받기 위해 AWS Support의 지원을 요청할 필요가 없습니다. Shield에는 다음 두 가지 계층이 있습니다. AWS Shield Standard 및 AWS Shield Advanced. 이 두 계층의 차이점에 대해 알아보려면 Shield 기능 설명서를참조하십시오.
- AWS Managed Services(AMS)
  - AWS Managed Services(AMS)에서 AWS 인프라를 지속적으로 관리하므로 사용자는 애플리케이션에 집중할 수 있습니다. AMS는 인프라를 유지 관리하기 위한 모범 사례를 구현함으로써 운영 오버헤드 및 위험을 줄이도록 지원합니다. AMS는 변경 요청, 모니터링, 패치 관리, 보안, 백업 서비스 등과 같은 일반적인 활동을 자동화하고 인프라를 프로비저닝, 운영 및 지원하기 위한 전체 수명 주기 서비스를 제공합니다.

- AMS는 일련의 보안 탐지 제어를 배포하고 경고에 대한 일차 대응을 연중무휴로 제공합니다. 경고가 시작되면 AMS는 일련의 표준 자동 및 수동 플레이북에 따라 일관된 응답을 확인합니다. 이러한 플레이북은 온보딩 중에 AMS 고객과 공유되므로 고객이 AMS를 통해 대응 방안을 개발하고 조정할 수 있습니다.

인시던트 대응 계획 개발

인시던트 대응 계획은 인시던트 대응 프로그램 및 전략의 기초가 되도록 설계되었습니다. 인시던트 대응 계획은 공식 문서에 포함되어야 합니다. 인시던트 대응 계획에는 일반적으로 다음 섹션이 포함됩니다.

- 인시던트 대응 팀 개요: 인시던트 대응 팀의 목표 및 기능을 간략하게 설명합니다.
- 역할 및 책임: 인시던트 대응 이해 관계자를 나열하고 인시던트 발생 시 해당 이해 관계자의 역할을 자세히 설명합니다.
- 커뮤니케이션 계획: 연락처 정보 및 인시던트 발생 시 커뮤니케이션 방법을 자세히 설명합니다.
- 백업 커뮤니케이션 방법: 인시던트 커뮤니케이션의 백업으로 대역 외 통신을 사용하는 것이 가장 좋습니다. 안전한 대역 외 통신 채널을 제공하는 애플리케이션의 예는 AWS Wickr입니다.
- 인시던트 대응 단계 및 취해야 할 조치: 인시던트 대응의 단계(예: 탐지, 분석, 제거, 억제, 복구)를 열거하며, 여기에는 해당 단계 내에서 취해야 할 상위 수준 조치가 포함됩니다.
- 인시던트 심각도 및 우선순위 정의: 인시던트의 심각도를 분류하는 방법, 인시던트의 우선순위를 지정하는 방법, 심각도 정의가 에스컬레이션 절차에 미치는 영향을 자세히 설명합니다.

이러한 섹션은 규모 및 업종이 다른 회사 간에 공통적으로 사용되지만 각 조직의 인시던트 대응 계획은 고유합니다. 조직에 가장 적합한 인시던트 대응 계획을 수립해야 합니다.

리소스

관련 모범 사례:

- SEC04(보안 관련 이벤트를 어떻게 탐지하나요?)

관련 문서:

- AWS 보안 인시던트 대응 안내서
- NIST: 컴퓨터 보안 인시던트 처리 안내서

SEC10-BP03 포렌식 역량 확보

인스던트 대응 담당자가 대응 계획의 어느 시점에 어떻게 포렌식 조사를 실행할지 이해하는 것이 중요합니다. 조직에서는 이 프로세스에서 어떤 증거가 수집되고 어떤 도구가 사용되는지 정의해야 합니다. 외부 전문가, 도구 및 자동화를 비롯하여 적합한 포렌식 역량을 파악하고 확보합니다. 미리 결정해야 할 중요한 사항은 라이브 시스템에서의 데이터 수집 여부입니다. 휘발성 메모리 콘텐츠 또는 액티브 네트워크 연결과 같은 일부 데이터는 시스템의 전원이 꺼지거나 재부팅되면 손실됩니다.

대응 팀에서는 AWS Systems Manager, Amazon EventBridge, AWS Lambda 등의 도구를 결합하여 운영 체제 및 VPC 트래픽 미러링 내에서 자동으로 포렌식 도구를 실행하여 네트워크 패킷 캡처를 확보함으로써 비영구적인 증거를 수집할 수 있습니다. 맞춤화된 포렌식 워크스테이션과 대응 담당자가 액세스 가능한 도구를 사용할 수 있는 전담 보안 계정에서 로그 분석 또는 디스크 이미지 분석과 같은 다른 활동을 수행합니다.

높은 내구성과 무결성을 제공하는 데이터 스토어에 관련 로그를 주기적으로 전송합니다. 대응 담당자는 이런 로그에 액세스할 수 있어야 합니다. AWS는 로그 조사를 용이하게 하는 Amazon Athena, Amazon OpenSearch Service(OpenSearch Service), Amazon CloudWatch Logs Insights와 같은 도구를 제공합니다. 또한 Amazon Simple Storage Service(Amazon S3) 객체 잠금을 사용하여 증거를 안전하게 보존합니다. 이 서비스는 WORM(Write-Once-Read-Many) 모델을 따르며 일정 기간 동안 객체가 삭제되거나 덮어써지지 않도록 합니다. 포렌식 조사 기법에는 전문가 교육이 필요하므로 외부 전문가의 참여가 필요할 수도 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 포렌식 기능 파악: 조직의 포렌식 조사 역량, 사용 가능한 도구 및 외부 전문가를 조사합니다.
- 인시던트 대응 및 포렌식 자동화

리소스

관련 문서:

- How to automate forensic disk collection in AWS(AWS에서 포렌식 디스크 수집을 자동화하는 방법)

SEC10-BP04 억제 기능 자동화

대응 시간과 조직에 대한 영향을 줄일 수 있도록 인시던트 억제 및 복구를 자동화합니다.

플레이북에서 프로세스와 도구를 생성하고 연습한 후에는 로직을 코드 기반 솔루션으로 해체할 수 있습니다. 그리고 이것은 많은 대응 인력들이 조치를 자동화하고 대응 인력의 편차 또는 추측을 없애기 위한 도구로 사용할 수 있습니다. 이렇게 하면 대응 수명 주기를 가속화할 수 있습니다. 다음 목표는 사람 응답자에 의해서가 아니라 알림 또는 이벤트 자체에서 이 코드가 호출되도록 함으로써 완벽히 자동으로 이루어지는 이벤트 중심의 대응을 생성하는 것입니다. 이러한 프로세스는 보안 시스템에 관련 데이터를 자동으로 추가해야 합니다. 예를 들어, 원치 않는 IP 주소에서 트래픽이 발생한 인시던트의 경우 AWS WAF 차단 목록 또는 Network Firewall 규칙 그룹이 자동으로 채워져 향후 활동을 차단할 수 있습니다.

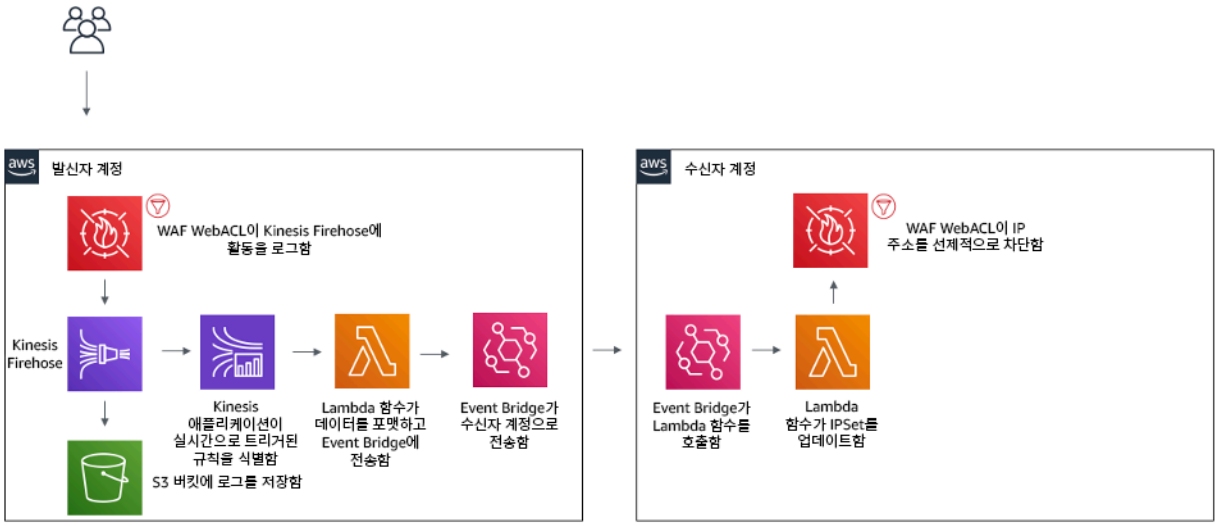


그림 3: 알려진 악성 IP 주소 차단을 자동화하는 AWS WAF

이벤트 중심의 대응 시스템을 사용하면 탐지 메커니즘이 대응 메커니즘을 트리거하여 이벤트를 자동으로 해결합니다. 이벤트 중심의 대응 기능을 사용하여 탐지 메커니즘과 대응 메커니즘 간의 시간을 단축할 수 있습니다. 이러한 이벤트 중심의 아키텍처를 생성하기 위해 이벤트에 대한 응답으로 코드를 실행하고 기본 컴퓨팅 리소스를 자동으로 관리하는 서버리스 컴퓨팅 서비스인 AWS Lambda를 사용할 수 있습니다. 예를 들어 AWS CloudTrail 서비스가 활성화된 AWS 계정이 있다고 가정해 보겠습니다. AWS CloudTrail이 비활성화된 경우(`cloudtrail:StopLogging` API 호출을 통해 Amazon EventBridge를 사용하여 특정 `cloudtrail:StopLogging` 이벤트를 모니터링하고 AWS Lambda 함수를 호출하여 `cloudtrail:StartLogging` 을 호출함으로써 로깅을 다시 시작할 수 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

억제 기능을 자동화합니다.

리소스

관련 문서:

- AWS 인시던트 대응 안내서

관련 동영상:

- AWS 환경에서 보안 인시던트 준비 및 대응

SEC10-BP05 액세스 권한 사전 프로비저닝

인시던트 응답자에게 AWS에 사전 프로비저닝된 올바른 액세스 권한이 있는지 확인하여 조사 및 복구 시간을 단축할 수 있도록 합니다.

일반적인 안티 패턴:

- 인시던트 대응을 위해 루트 계정을 사용합니다.
- 기존 사용자 계정을 변경합니다.
- 적시 권한 승격을 제공할 때 IAM 권한을 직접 조작합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

AWS는 가능한 경우 장기 보안 인증에 대한 의존도를 줄이거나 제거할 것을 권장합니다. 그 대신, 임시 보안 인증 및 적시 권한 승격 메커니즘을 사용하는 것이 좋습니다. 장기 보안 인증은 보안 위험에 노출되기 쉽고, 운영 오버헤드가 증가합니다. 따라서 대부분의 관리 작업에서 인시던트 대응 작업 뿐만 아니라 아이덴티티 페더레이션 과 함께 관리 액세스를 위한 임시 승격을 구현하는 것을 권장합니다. 이 모델에서는 사용자가 더 높은 수준의 권한(인시던트 대응 역할 등)을 요청하고, 사용자가 권한 승격에 적합한 경우 요청이 승인자에게 전송됩니다. 요청이 승인되면 사용자는 일련의 임시 AWS 보안 인증을 받아 자신의 작업을 완료하는 데 사용합니다. 이러한 보안 인증이 만료되면 사용자는 새로운 승격 요청을 제출해야 합니다.

대부분의 인시던트 대응 시나리오에서는 임시 권한 승격을 사용하는 것이 좋습니다. 이를 위한 올바른 방법은 AWS Security Token Service 및 세션 정책 을 사용하여 액세스의 범위를 지정하는 것입니다.

페더레이션형 ID를 사용할 수 없는 시나리오는 다음과 같습니다.

- ID 제공업체(idP)의 침해로 인한 중단

- 잘못된 구성 또는 인적 오류로 인한 페더레이션 액세스 관리 시스템의 손상
- DDoS(분산 서비스 거부) 이벤트 배포 또는 시스템을 사용할 수 없도록 렌더링하는 등의 악의적인 활동

위와 같은 사례의 경우, 긴급 브레이크 글라스 액세스를 구성하여 인시던트에 대한 조사 및 적시 개선이 이루어지도록 해야 합니다. 또한 적절한 권한이 있는 IAM 사용자 를 사용하여 작업을 수행하고 AWS 리소스에 액세스하는 것이 좋습니다. 루트 보안 인증은 루트 사용자 액세스가 필요한 작업에 사용합니다. 인시던트 응답자가 AWS 및 기타 관련 시스템에 대한 올바른 수준의 액세스 권한이 있는지 확인할 수 있도록, 전용 사용자 계정을 사전 프로비저닝하는 것이 좋습니다. 사용자 계정에는 권한이 있는 액세스가 필요하며, 엄격하게 제어 및 모니터링해야 합니다. 계정은 필요한 작업을 수행하기 위한 가장 최소한의 권한만으로 구축해야 하며, 액세스의 수준은 인시던트 관리 계획의 일부로 생성된 플레이북을 기준으로 해야 합니다.

모범 사례는 목적별 전용 사용자 및 역할을 사용하는 것입니다. IAM 정책 추가를 통해 사용자나 역할 액세스 권한을 임시 승격할 경우 인시던트가 발생하는 동안 사용자의 액세스 대상이 불명확해질 뿐만 아니라 승격된 권한이 취소되지 않는 위험이 발생합니다.

최대한 많은 수의 실패 시나리오에서 액세스를 얻을 수 있는지 확인할 수 있도록 가능한 많은 종속성을 제거하는 것이 중요합니다. 이를 지원하기 위해, 인시던트 대응 담당자가 전용 보안 계정에서 AWS Identity and Access Management 사용자로 생성되었는지, 그리고 기존 페더레이션 또는 Single Sign-On(SSO) 솔루션을 통해 관리되고 있지 않은지 확인할 수 있는 플레이북을 생성합니다. 각 개별 대응 담당자는 자신만의 명명된 계정을 가지고 있어야 합니다. 계정 구성은 강력한 암호 정책 및 다중 인증(MFA)을 적용해야 합니다. 인시던트 대응 플레이북에서 AWS Management Console에 대한 액세스 권한만 요구할 경우, 사용자는 구성된 액세스 키를 가지고 있지 않아야 하며 액세스 키 생성이 명시적으로 허용되지 않아야 합니다. 이것은 IAM 정책 또는 서비스 제어 정책(SCP)을 통해서만 구성해야 하며, AWS 보안 모범 사례(AWS Organizations SCP)를 따를 것을 권장합니다. 사용자는 다른 계정에서 인시던트 대응 역할을 수임할 수 있는 기능 외에 다른 권한이 없어야 합니다.

인시던트 과정에서 조사, 개선 조치 또는 복구 활동을 지원하기 위해 기타 내부 또는 외부 인력에게 액세스 권한을 부여해야 할 수 있습니다. 이 경우, 이전에 언급한 플레이북 메커니즘을 사용해야 하며, 인시던트가 완료된 후 모든 추가 액세스 권한이 즉시 취소되었는지 확인할 수 있는 프로세스가 반드시 있어야 합니다.

인시던트 대응 역할의 사용이 적절히 모니터링 및 감사되고 있는지 확인할 수 있도록, 이 목적을 위해 생성된 IAM 사용자 계정이 인력 간에 공유되지 않도록 하고, AWS 계정 루트 사용자가 특정 작업에 필요하지 않다면사용되지 않습니다. 루트 사용자가 필요한 경우(예를 들어, 특정 계정에 대한 IAM 액세스를 사용할 수 없는 경우), 사용 가능한 플레이북을 통해 별도의 프로세스를 사용하여 루트 사용자 암호 및 MFA 토큰의 가용성을 확인해야 합니다.

인시던트 대응 역할을 위한 IAM 정책을 구성하기 위해 IAM Access Analyzer 를 사용하여 AWS CloudTrail 로그를 기반으로 정책을 생성하는 것을 고려해 볼 수 있습니다. 이를 위해서는 비 프로덕션 계정에서 인시던트 대응 역할에 대한 관리자 액세스 권한을 부여하고 플레이북에 따라 실행해야 합니다. 완료되면 수행된 작업만 허용하는 정책을 생성할 수 있습니다. 그 후 이 정책은 모든 계정의 모든 인시던트 대응 역할에 적용할 수 있습니다. 더욱 쉬운 관리 및 감사를 허용할 수 있도록 각 플레이북에 대한 별도의 IAM 정책을 생성할 수 있습니다. 플레이북에 포함할 수 있는 예로는 랜섬웨어, 데이터 침해, 프로덕션 액세스의 손실 및 기타 시나리오에 대한 대응 계획이 있을 수 있습니다.

인시던트 대응 사용자 계정을 사용하여 전용 인시던트 대응 IAM 역할(다른 AWS 계정 계정 내)을 수임합니다. 이러한 역할은 보안 계정의 사용자만 수임할 수 있도록 구성되어야 하며, 신뢰 관계에서는 호출하는 보안 주체가 MFA를 사용하여 인증해야 합니다. 역할은 범위가 좁은 IAM 정책을 사용하여 액세스를 제어해야 합니다. 이러한 역할에 대한 모든 `AssumeRole` 요청은 CloudTrail에 로깅되고 알림이 생성되며, 이러한 역할을 사용하여 수행된 모든 작업이 로깅되도록 합니다.

IAM 사용자 계정과 IAM 역할의 이름을 모두 명확하게 지정하여 CloudTrail 로그에서 쉽게 찾을 수 있도록 하는 것이 좋습니다. 그 예로는 IAM 계정 *<USER_ID>*-BREAK-GLASS 및 IAM 역할 BREAK-GLASS-ROLE이 있습니다.

CloudTrail 은 AWS 계정의 API 활동을 기록하는 데 사용되며 인시던트 대응 역할의 사용에 대한 알림을 구성하는 데 사용해야 합니다. 루트 키 사용 시 알림 구성에 대한 블로그 게시물을 참조하세요. 지침을 수정하여 Amazon CloudWatch 지표 필터를 필터로 구성할 수 있으며 이 경우 `AssumeRole` 이벤트에 구성되며 인시던트 대응 IAM 역할과 관련됩니다.

```
{ $.eventName = "AssumeRole" && $.requestParameters.roleArn =
 "<INCIDENT_RESPONSE_ROLE_ARN>" && $.userIdentity.invokedBy NOT EXISTS && $.eventType !
= "AwsServiceEvent" }
```

인시던트 대응 역할은 높은 수준의 액세스 권한을 가질 가능성이 높기 때문에, 이러한 알림은 광범위한 그룹에 전달되고 신속하게 조치되는 것이 중요합니다.

인시던트 발생 시 응답자는 IAM에 의해 직접 보호되지 않는 시스템에 대한 액세스가 필요할 수 있습니다. 여기에는 Amazon Elastic Compute Cloud 인스턴스, Amazon Relational Database Service 데이터베이스 또는 서비스형 소프트웨어(SaaS) 플랫폼이 포함됩니다. SSH 또는 RDP와 같은 기본 프로토콜을 사용하는 것보다는 AWS Systems Manager Session Manager 를 Amazon EC2 인스턴스에 대한 모든 관리 액세스에 사용하는 것이 좋습니다. 이 액세스는 보안 및 감사 기능이 있는 IAM을 사용하여 제어할 수 있습니다. 또한 AWS Systems Manager Run Command 문서를 사용하여 플레이북의 일부를 자동화할 수 있으며, 이를 통해 사용자 오류를 줄이고 복구 시간을 개선할 수 있습니다. 데이터베이스 및 서드 파티 도구에 대한 액세스를 위해, AWS Secrets Manager에 액세스 보안 인증을 저장하고 인시던트 응답자 역할에 액세스 권한을 부여하는 것이 좋습니다.

마지막으로, 인시던트 대응 IAM 사용자 계정의 관리를 입사, 전근 및 퇴사 프로세스 에 추가하고 정기적으로 검토 및 테스트하여 대상 액세스만 허용되는지 확인해야 합니다.

리소스

관련 문서:

- AWS 환경에 대한 임시 승격된 액세스 관리
- AWS 보안 인시던트 대응 안내서
- AWS Elastic Disaster Recovery
- AWS Systems Manager Incident Manager
- IAM 사용자의 계정 암호 정책 설정
- AWS에서 다중 인증(MFA) 사용
- MFA를 통한 크로스 계정 액세스 구성
- IAM Access Analyzer를 사용하여 IAM 정책 생성
- 다중 계정 환경에서의 AWS Organizations 서비스 제어 정책 모범 사례
- AWS 계정의 루트 액세스 키가 사용될 때 알림을 받는 방법
- IAM 관리형 정책을 사용하여 세분화된 세션 권한 생성

관련 동영상:

- AWS의 인시던트 대응 및 포렌식 자동화
- 런북, 인시던트 보고서, 인시던트 대응에 대한 DIY 가이드
- AWS 환경에서 보안 인시던트 준비 및 대응

관련 예시:

- 실습: AWS 계정 설정 및 루트 사용자
- 실습: AWS 콘솔 및 CLI를 사용한 인시던트 대응

SEC10-BP06 도구 사전 배포

AWS에 사전 배포된 올바른 도구를 보안 담당자에게 제공함으로써 조사부터 복구까지 걸리는 시간을 단축할 수 있도록 합니다.

보안 엔지니어링 및 운영 기능을 자동화하기 위해 AWS의 포괄적인 API 및 도구 세트를 사용할 수 있습니다. 자격 증명 관리, 네트워크 보안, 데이터 보호, 모니터링 기능을 완전히 자동화하고 이미 사용하고 있는 대중적인 소프트웨어 개발 방법을 사용하여 제공할 수 있습니다. 보안 자동화를 구축하면 직원이 보안 상태를 모니터링하면서 수동으로 이벤트에 대응하는 것이 아니라 시스템이 모니터링 및 검토하고 대응을 시작할 수 있습니다. AWS 서비스 전체에서 검색 가능하며 관련성 있는 로그 데이터를 인시던트 대응 담당자에게 자동으로 제공하는 효과적인 방법 중 하나는 다음을 사용하는 것입니다. Amazon Detective.

인시던트 대응팀은 같은 방식으로 계속 알림에 대응할 경우 알림에 대한 피로감을 느낄 위험이 있습니다. 시간이 지남에 따라 팀이 알림에 무감각한 상태가 되어 일상적인 상황을 처리하는 데 실수하거나 비정상적인 알림을 놓칠 수 있습니다. 자동화는 반복적이고 일상적인 알림을 처리하는 기능을 사용함으로써 알림에 대한 피로감을 방지하며, 중요하고 특별한 인시던트만 사람이 직접 처리하도록 합니다. Amazon GuardDuty, AWS CloudTrail Insights, Amazon CloudWatch Anomaly Detection과 같은 이상 탐지 시스템을 통합하면 일반적인 임계값 기반 알림의 부담을 줄일 수 있습니다.

프로세스의 단계를 프로그래밍 방식으로 자동화하여 수동 프로세스를 개선할 수 있습니다. 이벤트에 대한 수정 패턴을 정의한 후 해당 패턴을 실행 가능한 로직으로 분해하고 코드를 작성하여 해당 로직을 수행할 수 있습니다. 그런 다음, 응답자가 해당 코드를 실행하여 문제를 해결할 수 있습니다. 시간이 지남에 따라 점점 더 많은 단계를 자동화할 수 있으며, 궁극적으로 일반적인 인시던트의 전체 클래스를 자동으로 처리할 수 있습니다.

Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스의 운영 체제 내에서 실행되는 도구의 경우, Amazon EC2 인스턴스 운영 체제에 설치한 에이전트를 사용하여 원격으로 안전하게 인스턴스를 관리할 수 있도록 해주는 AWS Systems Manager Run Command를 사용하여 평가해야 합니다. 많은 Amazon Machine Image(AMI)에 기본적으로 설치된 Systems Manager Agent(SSM Agent)가 필요합니다. 하지만 일단 인스턴스가 손상되었으면 해당 인스턴스에서 실행 중인 도구 또는 에이전트의 응답은 신뢰할 수 있는 것으로 간주해서는 안 됩니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 도구 사전 배포: 인시던트에 적절하게 대응할 수 있도록 보안 담당자가 적절한 도구를 AWS에 사전 배포하도록 합니다.
  - 실습: AWS Management Console 및 CLI를 사용한 인시던트 대응
  - Jupyter를 사용한 인시던트 대응 플레이북 - AWS IAM
  - AWS 보안 자동화

- 리소스 태깅 구현: 인시던트 발생 시 리소스를 식별할 수 있도록 조사 중인 리소스의 코드와 같은 정보로 리소스를 태깅합니다.
  - AWS 태그 지정 전략

리소스

관련 문서:

- AWS 인시던트 대응 안내서

관련 동영상:

- 런북, 인시던트 보고서, 인시던트 대응에 대한 DIY 가이드

SEC10-BP07 게임 데이 진행

시간이 지나면서 조직이 성장하고 발전함에 따라 위협 환경도 변화하므로 인시던트 대응 능력을 지속적으로 검토하는 것이 중요합니다. 이 평가를 수행하는 데 사용할 수 있는 한 가지 방법은 게임 데이 또는 시뮬레이션을 실행하는 것입니다. 시뮬레이션은 위협 행위자의 전술, 기술 및 절차(TTP)를 모방하도록 설계된 실제 보안 이벤트 시나리오를 사용하며, 이를 통해 조직은 이러한 모의 사이버 이벤트에 실제 상황과 같이 대응하여 인시던트 대응 능력을 발휘하고 평가할 수 있습니다.

이 모범 사례 확립의 이점: 시뮬레이션에는 다음과 같은 다양한 이점이 있습니다.

- 사이버 대비 상태를 검증하고 인시던트 대응자의 자신감을 높입니다.
- 도구 및 워크플로의 정확성과 효율성을 테스트합니다.
- 인시던트 대응 계획에 맞춰 커뮤니케이션 및 에스컬레이션 방법을 개선합니다.
- 덜 일반적인 벡터에 대응할 수 있는 기회를 제공합니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 보통

구현 가이드

시뮬레이션에는 다음과 같은 세 가지 주요 유형이 있습니다.

- 탁상 연습: 시뮬레이션에 대한 탁상 접근 방식은 다양한 인시던트 대응 이해 관계자가 참여하여 책임진 역할을 연습하고 확립된 커뮤니케이션 도구과 플레이북을 사용하는 토론 기반 세션입니다. 연습

은 일반적으로 가상 장소, 실제 장소 또는 이들 장소의 조합에서 하루 종일 수행할 수 있어 언제든 촉진시킬 수 있습니다. 토론을 기반으로 하기 때문에 탁상 연습은 프로세스, 사람, 협업에 중점을 둡니다. 기술은 토론의 핵심 부분이지만 인시던트 대응 도구 또는 스크립트의 실제 사용은 일반적으로 탁상 연습의 일부가 아닙니다.

- 퍼플 팀 연습: 퍼플 팀 연습은 인시던트 대응 담당자(블루 팀)와 시뮬레이션된 위협 행위자(레드 팀) 간의 협업 수준을 높입니다. 블루 팀은 보안 운영 센터(SOC)의 직원으로 구성되지만 실제 사이버 이벤트 중에 관여하게 될 다른 이해 관계자들도 포함될 수 있습니다. 레드 팀은 보안 공격 교육을 받은 침투 테스트 팀 또는 주요 이해 관계자로 구성됩니다. 레드 팀은 시나리오를 설계할 때 연습 진행자와 협력하여 시나리오가 정확하고 실현 가능한지 확인합니다. 퍼플 팀 연습에서는 인시던트 대응 작업을 지원하는 탐지 메커니즘, 도구 및 표준 운영 절차(SOP)에 주로 초점을 맞춥니다.
- 레드 팀 연습: 레드 팀 연습 중에 공격 팀(레드 팀)은 미리 정해진 범위에서 특정 목표 또는 일련의 목표를 달성하기 위해 시뮬레이션을 수행합니다. 방어 팀(블루 팀)은 훈련의 범위와 기간을 꼭 알 필요가 없습니다. 이를 모르면 실제 인시던트에 어떻게 대응하는지에 대한 더 현실적인 평가를 받을 수 있습니다. 레드 팀 연습은 침습적 테스트일 수 있으므로 주의가 필요하고 해당 연습이 환경에 실제로 해를 끼치지 않는지 확인하기 위한 관리 조치를 취해야 합니다.

정기적으로 사이버 시뮬레이션을 진행하는 것이 좋습니다. 각 연습 유형에는 참가자와 조직 전체에 대한 고유한 이점이 있으므로 덜 복잡한 시뮬레이션 유형(예: 탁상 연습)에서 시작하여 더 복잡한 시뮬레이션 유형(레드 팀 연습)으로 진행할 수 있습니다. 보안 성숙도, 리소스, 원하는 결과에 따라 시뮬레이션 유형을 선택해야 합니다. 일부 고객은 복잡성과 비용 때문에 레드 팀 연습을 선택하지 않을 수 있습니다.

구현 단계

선택한 유형에 관계없이 시뮬레이션은 일반적으로 다음 구현 단계를 따릅니다.

1. 핵심 연습 요소 정의: 시뮬레이션의 시나리오와 목표를 정의합니다. 이 두 가지 모두 리더의 승인을 받아야 합니다.
2. 주요 이해 관계자 식별: 연습에는 최소한 연습 진행자와 참가자가 필요합니다. 시나리오에 따라 법무, 커뮤니케이션 또는 경영진과 같은 추가 이해 관계자가 참여할 수 있습니다.
3. 시나리오 구축 및 테스트: 특정 요소가 실현 가능하지 않은 경우 구축 중인 시나리오를 재정의해야 할 수 있습니다. 이 단계의 결과로 최종 시나리오가 도출될 것으로 예상됩니다.
4. 시뮬레이션 촉진: 시뮬레이션 유형에 따라 어떤 방법으로 촉진시킬지 결정됩니다(종이를 사용한 시나리오 또는 고도로 기술적인 시뮬레이션 시나리오). 진행자는 연습 목표에 맞게 촉진 전략을 조정해야 하며 가능한 한 모든 연습 참가자를 참여시켜 최대한의 이점을 확보해야 합니다.

5. 사후 조치 보고서(AAR) 작성: 잘 운영된 영역, 개선이 필요한 영역, 잠재적인 격차를 파악합니다. AAR은 시뮬레이션의 효과와 시뮬레이션된 이벤트에 대한 팀의 반응을 측정하여 향후 시뮬레이션을 통해 시간의 흐름에 따른 진행 상황을 추적할 수 있도록 해야 합니다.

리소스

관련 문서:

- AWS 인시던트 대응 안내서

관련 동영상:

- AWS GameDay - Security Edition(보안 에디션)

애플리케이션 보안

질문

- SEC 11 설계, 개발 및 배포 수명 주기 전반에 걸쳐 애플리케이션의 보안 속성을 어떻게 통합하고 검증합니까?

SEC 11 설계, 개발 및 배포 수명 주기 전반에 걸쳐 애플리케이션의 보안 속성을 어떻게 통합하고 검증합니까?

인력 교육, 자동화를 사용한 테스트, 종속성 이해, 도구 및 애플리케이션의 보안 속성 검증은 프로덕션 워크로드에서 발생할 수 있는 보안 문제를 줄이는 데 도움이 됩니다.

모범 사례

- SEC11-BP01 애플리케이션 보안 교육
- SEC11-BP02 개발 및 릴리스 수명 주기를 통한 테스트 자동화
- SEC11-BP03 정기적인 침투 테스트 시행
- SEC11-BP04 수동 코드 검토
- SEC11-BP05 패키지 및 종속성 서비스의 중앙 집중화
- SEC11-BP06 프로그래밍 방식으로 소프트웨어 배포
- SEC11-BP07 정기적으로 파이프라인의 보안 속성 평가
- SEC11-BP08 보안 소유권이 워크로드 팀에 귀속되도록 프로그램 구축

SEC11-BP01 애플리케이션 보안 교육

애플리케이션의 안전한 개발 및 운영을 위해 조직 내 빌더에게 일반적인 사례를 교육합니다. 보안 중점 개발 관행을 도입하면 보안 검토 단계에서만 탐지되는 문제의 발생 가능성을 줄이는 데 도움이 됩니다.

원하는 결과: 소프트웨어는 보안을 염두에 두어 설계되고 구축되어야 합니다. 조직의 빌더가 위협 모델로 시작하는 보안 개발 관행에 대해 교육을 받으면 제작되는 소프트웨어의 전반적인 품질과 보안이 향상됩니다. 이 접근 방식은 보안 검토 단계 이후에 재작업의 필요성을 덜어주어 소프트웨어나 기능 납품 시간을 줄일 수 있습니다.

이 모범 사례에서 보안 개발은 작성 중인 소프트웨어와 소프트웨어 개발 수명 주기(SDLC)를 지원하는 도구 또는 시스템을 의미합니다.

일반적인 안티 패턴:

- 보안 검토가 끝날 때까지 기다린 다음 시스템의 보안 속성을 고려합니다.
- 보안 팀에 모든 보안 결정을 맡깁니다.
- SDLC에서 내린 결정이 조직의 전반적인 보안 기대치 또는 정책과 어떤 관련이 있는지를 전달하지 못합니다.
- 보안 검토 프로세스에 너무 늦게 참여합니다.

이 모범 사례 확립의 이점:

- 개발 주기 초기에 조직의 보안 요구 사항을 보다 효과적으로 이해합니다.
- 잠재적인 보안 문제를 보다 신속하게 식별하고 해결하여 기능을 발 빠르게 제공할 수 있습니다.
- 소프트웨어 및 시스템의 품질이 향상됩니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

조직의 빌더를 교육합니다. 위협 모델링 과정부터 시작하면 보안 교육의 굳건한 토대를 마련할 수 있습니다. 이상적으로는 빌더가 워크로드와 관련된 정보에 자체적으로 액세스할 수 있어야 합니다. 이러한 액세스를 통해 구축하는 시스템의 보안 속성에 대해 다른 팀에 문의할 필요 없이 정보에 기반한 의사 결정을 내릴 수 있습니다. 검토에 보안 팀을 참여시키는 프로세스가 명확하게 정의되어야 하며, 이를 간단하게 따를 수 있어야 합니다. 검토 프로세스의 단계는 보안 교육에 포함되어야 합니다. 알려진 구현 패턴 또는 템플릿을 사용할 수 있는 경우 찾기가 쉬워야 하며 전반적인 보안 요구 사항과 연결되어

야 합니다. 직접 맞춤 구성할 필요성을 줄이려면 AWS CloudFormation, AWS Cloud Development Kit (AWS CDK) Constructs, Service Catalog 또는 기타 템플릿 도구 사용을 고려해 보세요.

구현 단계

- 빌더를 대상으로 위협 모델링 과정을 교육하면 좋은 토대를 마련하고 보안을 대하는 방식을 교육할 수 있습니다.
- AWS 교육 and Certification, 산업 또는 AWS 파트너 교육에 대한 액세스 권한을 제공합니다.
- 보안 팀, 워크로드 팀 및 기타 이해 관계자 간의 책임 분담을 명확히 하는 조직의 보안 검토 프로세스를 교육합니다.
- 가능한 경우 코드 예시 및 템플릿을 포함하여 보안 요구 사항을 충족하는 방법을 다루는 자습형 지침을 게시합니다.
- 보안 검토 프로세스 및 교육을 받은 빌더 팀의 경험에 대해 정기적으로 피드백을 얻고, 피드백을 바탕으로 개선합니다.
- 게임 데이 또는 버그 배쉬 캠페인을 사용하여 문제 수를 줄이고 빌더의 기술을 강화합니다.

리소스

관련 모범 사례:

- SEC11-BP08 보안 소유권이 워크로드 팀에 귀속되도록 프로그램 구축

관련 문서:

- AWS 교육 and Certification
- How to think about cloud security governance(클라우드 보안 거버넌스를 대하는 방식)
- How to approach threat modeling(위협 모델링 접근 방식)
- Accelerating training – The AWS Skills Guild(교육 가속화 - AWS Skills Guild)

관련 동영상:

- Proactive security: Considerations and approaches(사전 예방적 보안: 고려 사항 및 접근 방법)

관련 예시:

- Workshop on threat modeling(위협 모델링 워크숍)

- Industry awareness for developers(개발자를 위한 업계 인지도)

관련 서비스:

- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK)(AWS CDK) Constructs
- Service Catalog
- AWS BugBust

SEC11-BP02 개발 및 릴리스 수명 주기를 통한 테스트 자동화

개발 및 릴리스 수명 주기 전반에 걸쳐 보안 속성 테스트를 자동화하세요. 자동화를 구현하면 릴리스에 앞서 소프트웨어의 잠재적인 문제를 일관되고 반복적으로 손쉽게 식별할 수 있어 소프트웨어 제공 중에 보안 문제가 발생할 위험이 줄어듭니다.

원하는 결과: 자동화된 테스트의 목표는 개발 수명 주기 전반에 걸쳐 잠재적인 문제를 조기에 자주 탐지할 수 있는 프로그래밍 방식을 제공하는 것입니다. 회귀 테스트를 자동화하면 기능 테스트 및 비기능 테스트를 다시 실행하여 이전에 테스트한 소프트웨어가 변경 후에도 예상대로 작동하는지 확인할 수 있습니다. 보안 장치 테스트를 정의하여 인증 정보 손상 또는 누락과 같은 일반적인 구성 오류를 확인하면 이러한 문제를 개발 프로세스 초기에 식별하고 해결할 수 있게 됩니다.

테스트 자동화는 애플리케이션의 요구 사항과 원하는 기능을 기반으로 애플리케이션 검증을 위해 특별히 제작된 테스트 사례를 사용합니다. 자동화된 테스트 결과는 생성된 테스트 출력을 각각의 예상 출력과 비교하여 전체 테스트 수명 주기를 가속화합니다. 회귀 테스트 및 장치 테스트 세트와 같은 테스트 방법론이 자동화에 가장 적합합니다. 보안 속성 테스트를 자동화하면 빌더가 보안 검토를 기다리지 않고도 자동으로 피드백을 받을 수 있습니다. 정적 또는 동적 코드 분석의 형태로 자동화된 테스트는 코드 품질을 개선하고 개발 수명 주기 초기에 잠재적인 소프트웨어 문제를 탐지하는 데 도움이 됩니다.

일반적인 안티 패턴:

- 자동화된 테스트의 테스트 사례 및 테스트 결과를 전달하지 않습니다.
- 릴리스 직전에만 자동화된 테스트를 수행합니다.
- 요구 사항이 자주 변경되는 테스트 사례를 자동화합니다.
- 보안 테스트 결과를 처리하는 방법에 대한 지침을 제공하지 못합니다.

이 모범 사례 확립의 이점:

- 시스템의 보안 속성을 평가하는 사용자에 대한 의존도를 낮춥니다.
- 여러 작업 흐름에서 일정한 결과를 도출하여 일관성이 향상됩니다.
- 프로덕션 소프트웨어에 보안 문제가 발생할 가능성이 줄어듭니다.
- 소프트웨어 문제를 조기에 발견하여 탐지부터 해결에 걸리는 시간이 단축됩니다.
- 여러 작업 흐름에 걸쳐 체계적이거나 반복적인 동작에 대한 가시성이 향상되어 조직 전체의 개선을 추진하는 데 유용합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

소프트웨어를 구축할 때 다양한 소프트웨어 테스트 메커니즘을 채택하여 애플리케이션의 비즈니스 논리에 바탕을 둔 기능적 요구 사항과 애플리케이션 신뢰성, 성능, 보안에 중점을 둔 비기능적 요구 사항을 기반으로 애플리케이션을 테스트합니다.

정적 애플리케이션 보안 테스트(SAST)는 비정상적인 보안 패턴에 대한 소스 코드를 분석하고 결함이 발생하기 쉬운 코드를 표시해 줍니다. SAST는 문서(요구 사항 사양, 설계 설명서, 설계 사양)와 같은 정적 입력 및 애플리케이션 소스 코드를 전적으로 사용하여 알려진 여러 보안 문제를 테스트합니다. 정적 코드 분석기는 대량의 코드를 신속하게 분석하는 데 도움이 됩니다. NIST Quality Group은 바이트 코드 스캐너 및 이진 코드 스캐너용 오픈 소스 도구를 포함하는 소스 코드 보안 분석기의 비교 정보를 제공합니다.

실행 중인 애플리케이션을 테스트하여 잠재적으로 예상치 못한 동작을 식별하는 동적 분석 보안 테스트(DAST) 방법론으로 정적 테스트를 보완합니다. 동적 테스트를 사용하면 정적 분석을 통해 탐지할 수 없는 잠재적 문제를 감지할 수 있습니다. 코드 리포지토리, 구축 및 파이프라인 단계에서 테스트하면 코드 입력 시 발생 가능한 여러 유형의 잠재적 문제를 확인할 수 있습니다. Amazon CodeWhisperer는 빌더의 IDE에서 보안 검색을 포함한 코드 권장 사항을 제공합니다. Amazon CodeGuru Reviewer는 애플리케이션 개발 중 중대한 문제, 보안 문제 및 찾기 어려운 버그를 식별할 수 있으며 코드 품질을 개선하기 위한 권장 사항을 제공합니다.

Security for Developers 워크숍에서는 AWS CodeBuild, AWS CodeCommit, AWS CodePipeline 등의 AWS 개발자 도구를 사용하여 SAST 및 DAST 테스트 방법론이 포함된 릴리스 파이프라인 자동화를 수행합니다.

SDLC를 진행하면서 보안 팀과 함께 정기적인 애플리케이션 검토를 포함하는 반복 프로세스를 수립하세요. 이러한 보안 검토에서 수집된 피드백은 릴리스 준비 상태 검토 과정에서 해결하고 검증해야 합니다. 검토를 통해 강력한 애플리케이션 보안 태세를 확립하고, 빌더에게 잠재적인 문제를 해결하는 데 도움이 되는 실용적인 피드백을 제공할 수 있습니다.

구현 단계

- 보안 테스트가 포함된 IDE, 코드 검토 및 CI/CD 도구를 일관성 있게 구현합니다.
- 문제를 해결해야 한다고 빌더에게 통보하는 대신 SDLC의 어느 지점에서 파이프라인을 차단하는 것이 적절한지 고려해 보세요.
- Security for Developers 워크숍에서는 정적 및 동적 테스트를 릴리스 파이프라인에 통합하는 예를 제공합니다.
- 개발자 IDE와 통합된 Amazon CodeWhisperer 및 커밋 시점에 코드를 스캔하는 Amazon CodeGuru Reviewer와 같은 자동화된 도구를 사용하여 테스트 또는 코드 분석을 수행하면 빌더가 적시에 피드백을 받을 수 있습니다.
- AWS Lambda를 사용하여 구축하면 Amazon Inspector를 통해 함수의 애플리케이션 코드를 스캔할 수 있습니다.
- AWS CI/CD 워크숍은 AWS에서 CI/CD 파이프라인을 구축하기 위한 시작점을 제공합니다.
- CI/CD 파이프라인에 자동화된 테스트가 포함된 경우 티켓팅 시스템을 사용하여 소프트웨어 문제의 알림 및 해결 방법을 추적해야 합니다.
- 결과를 생성할 수 있는 보안 테스트의 경우 해결 지침에 연결하면 빌더가 코드 품질을 개선하는 데 도움이 됩니다.
- 자동화된 도구의 결과를 정기적으로 분석하여 다음 자동화, 빌더 교육 또는 인식 캠페인의 우선순위를 지정합니다.

리소스

관련 문서:

- 지속적 전달 및 지속적 배포

- AWS DevOps 컴피턴시 파트너
- AWS 보안 컴피턴시 파트너(애플리케이션 보안용)
- Choosing a Well-Architected CI/CD approach(Well-Architected CI/CD 접근 방법 선택)
- Monitoring CodeCommit events in Amazon EventBridge and Amazon CloudWatch Events(Amazon EventBridge와 Amazon CloudWatch Events에서 CodeCommit 이벤트 모니터링)
- Secrets detection in Amazon CodeGuru Review(Amazon CodeGuru 검토의 비밀 탐지)
- Accelerate deployments on AWS with effective governance(효과적인 거버넌스를 통한 AWS의 배포 가속화)

- How AWS approaches automating safe, hands-off deployments(AWS 접근 방식으로 안전하게 자동 배포를 자동화하는 방법)

관련 동영상:

- Hands-off: Automating continuous delivery pipelines at Amazon(자동: Amazon에서 지속적 전달 파이프라인 자동화)
- Automating cross-account CI/CD pipelines(크로스 계정 CI/CD 파이프라인 자동화)

관련 예시:

- Industry awareness for developers(개발자를 위한 업계 인지도)
- AWS CodePipeline 거버넌스(GitHub)
- Security for Developers workshop(Security for Developers 워크숍)
- AWS CI/CD Workshop(AWS CI/CD 워크숍)

SEC11-BP03 정기적인 침투 테스트 시행

정기적으로 소프트웨어 침투 테스트를 시행하세요. 이러한 메커니즘은 자동화된 테스트나 수동 코드 검토로 탐지할 수 없는 잠재적인 소프트웨어 문제를 식별하는 데 도움이 됩니다. 또한 탐지 컨트롤의 효율성을 이해하는 데 도움이 될 수 있습니다. 침투 테스트를 통해 보호해야 하는 데이터를 노출하거나 예상보다 더 광범위한 권한을 부여하는 등 소프트웨어가 예기치 않은 방식으로 작동할 가능성이 있는지 확인해야 합니다.

원하는 결과: 침투 테스트는 애플리케이션의 보안 속성을 탐지, 문제 해결 및 검증하는 데 사용됩니다. 소프트웨어 개발 수명 주기(SDLC)의 일부로 일정을 정해 정기적인 침투 테스트를 수행해야 합니다. 침투 테스트로 인해 발견한 결과는 소프트웨어 출시 전에 해결해야 합니다. 침투 테스트의 결과를 분석하여 자동화를 통해 찾을 수 있는 문제가 있는지 확인해야 합니다. 능동적 피드백 메커니즘을 포함하는 정기적이고 반복 가능한 침투 테스트 프로세스를 통해 빌더에게 지침을 제공하고 소프트웨어 품질을 개선할 수 있습니다.

일반적인 안티 패턴:

- 알려진 보안 문제 또는 일반적인 보안 문제에 대한 침투 테스트만 수행합니다.
- 종속된 타사 도구 및 라이브러리가 없는 애플리케이션에 대한 침투 테스트만 수행합니다.
- 패키지 보안 문제에 대한 침투 테스트만 수행하고 구현된 비즈니스 논리는 평가하지 않습니다.

이 모범 사례 확립의 이점:

- 릴리스 전에 소프트웨어의 보안 속성에 대한 신뢰도가 높아집니다.
- 선호하는 애플리케이션 패턴을 식별할 수 있는 기회를 제공하여 소프트웨어 품질을 개선합니다.
- 피드백 루프를 통해 자동화 또는 추가 교육으로 소프트웨어의 보안 속성을 개선할 여지가 있는 개발 주기의 초기 단계를 식별할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

침투 테스트는 계획된 보안 위반 시나리오를 실행하여 보안 제어 기능을 탐지, 문제 해결 및 검증하는 체계적인 보안 테스트 활동입니다. 침투 테스트는 정찰부터 시작되는데, 이때 애플리케이션의 현재 설계와 종속성을 기반으로 데이터가 수집됩니다. 보안 관련 테스트 시나리오의 선별 목록도 작성되고 실행됩니다. 침투 테스트의 주요 목적은 환경이나 데이터에 무단으로 액세스할 권한을 얻는 데 악용될 수 있는 애플리케이션의 보안 문제를 파악하는 것입니다. 새 기능을 출시할 때 또는 애플리케이션 기능이나 기술 구현이 크게 변경될 때마다 침투 테스트를 수행해야 합니다.

침투 테스트를 수행하려면 개발 수명 주기에서 가장 적합한 단계를 식별해야 합니다. 시스템의 기능이 원하는 릴리스 상태에 근접했을 정도로 늦은 시점에 수행하되, 이때 문제를 해결할 시간이 충분히 남아 있어야 합니다.

구현 단계

- 침투 테스트의 범위를 파악하는 체계적인 프로세스를 수립해야 합니다. 위협 모델을 기반으로 이 프로세스를 수행하면 컨텍스트를 유지할 수 있습니다.
- 침투 테스트를 수행하기 위한 개발 주기의 적절한 시점을 파악합니다. 애플리케이션에 예상되는 변경이 최소한만 남아 있는 동시에 문제를 해결하기에 시간이 충분한 시점이어야 합니다.
- 빌더에게 침투 테스트 결과에서 기대할 수 있는 정보를 알려주고 문제 해결 정보를 얻는 방법을 교육합니다.
- 도구를 통해 공통 또는 반복 가능한 테스트를 자동화하여 침투 테스트 프로세스를 가속화합니다.
- 침투 테스트 결과를 분석하여 시스템 보안 문제를 식별하고, 이 데이터를 바탕으로 자동화된 테스트를 추가로 수행하고 빌더를 꾸준히 교육합니다.

리소스

관련 모범 사례:

- SEC11-BP01 애플리케이션 보안 교육
- SEC11-BP02 개발 및 릴리스 수명 주기를 통한 테스트 자동화

관련 문서:

- AWS 침투 테스트(AWS의 침투 테스트에 대한 자세한 지침 제공)
- Accelerate deployments on AWS with effective governance(효과적인 거버넌스를 통한 AWS의 배포 가속화)
- AWS 보안 컴피턴시 파트너
- Modernize your penetration testing architecture on AWS Fargate(AWS Fargate의 침투 테스트 아키텍처 현대화)
- AWS Fault injection Simulator

관련 예시:

- AWS CodePipeline을 통한 API 테스트 자동화(GitHub)
- 자동화된 보안 헬퍼(GitHub)

SEC11-BP04 수동 코드 검토

개발할 소프트웨어의 코드를 수동으로 검토하세요. 이 프로세스를 통해 코드를 작성한 개발자 이외의 사람이 코드 품질을 검사하게 됩니다.

원하는 결과: 개발 중에 수동 코드 검토 단계를 포함하면 작성 중인 소프트웨어의 품질이 높아지고, 경험이 부족한 팀원의 기술이 향상되며, 자동화를 사용할 수 있는 부분을 식별할 기회를 얻을 수 있습니다. 수동 코드 검토는 자동화된 도구 및 테스트를 통해 지원됩니다.

일반적인 안티 패턴:

- 배포 전에 코드 검토를 수행하지 않습니다.
- 같은 사람이 코드를 작성하고 검토하도록 합니다.
- 코드 검토를 지원하거나 조율하는 데 자동화를 사용하지 않습니다.
- 빌더가 코드를 검토하기 전에 애플리케이션 보안 교육을 받지 않습니다.

이 모범 사례 확립의 이점:

- 코드 품질이 향상됩니다.
- 공통된 접근 방식을 재사용할 수 있어 코드 개발의 일관성이 높아집니다.
- 침투 테스트 및 이후 단계에서 발견되는 문제의 수가 줄어듭니다.
- 팀 내 지식 전달이 개선됩니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

검토 단계는 전체 코드 관리 흐름의 일부로 구현되어야 합니다. 자세한 내용은 분기, 풀 요청 및 병합에 사용되는 전략에 따라 달라집니다. AWS CodeCommit 또는 GitHub, GitLab, Bitbucket 등 타사 솔루션을 사용하고 있을 수 있습니다. 어떤 방법을 사용하든 프로덕션 환경에 배포하는 데 앞서 프로세스에 코드 검토가 필요한지 확인하는 것이 중요합니다. Amazon CodeGuru Reviewer와 같은 도구를 사용하면 코드 검토 프로세스를 보다 쉽게 조율할 수 있습니다.

구현 단계

- 코드 관리 흐름의 일부로 수동 검토 단계를 구현하고, 계속하기 전에 검토를 수행합니다.
- 코드 검토를 관리하고 지원하는 Amazon CodeGuru Reviewer 사용을 고려해 보세요.
- 코드를 다음 단계로 진행하려면 코드 검토를 완료해야 하는 승인 흐름을 구현합니다.
- 수동 코드 검토 중에 자동으로 탐지 가능한 문제를 식별하는 프로세스가 있는지 확인합니다.
- 코드 개발 관행에 맞게 수동 코드 검토 단계를 통합합니다.

리소스

관련 모범 사례:

- SEC11-BP02 개발 및 릴리스 수명 주기를 통한 테스트 자동화

관련 문서:

- Working with pull requests in AWS CodeCommit repositories(AWS CodeCommit 리포지토리에서 풀 요청 작업)
- Working with approval rule templates in AWS CodeCommit(AWS CodeCommit에서 승인 규칙 템플릿 작업)
- GitHub의 풀 요청 정보

- Automate code reviews with Amazon CodeGuru Reviewer(Amazon CodeGuru Reviewer를 사용한 코드 검토 자동화)
- Automating detection of security vulnerabilities and bugs in CI/CD pipelines using Amazon CodeGuru Reviewer CLI(Amazon CodeGuru Reviewer CLI를 사용하여 CI/CD 파이프라인의 보안 취약성 및 버그 탐지 자동화)

관련 동영상:

- Continuous improvement of code quality with Amazon CodeGuru(Amazon CodeGuru를 통한 코드 품질의 지속적인 개선)

관련 예시:

- Security for Developers workshop(Security for Developers 워크숍)

SEC11-BP05 패키지 및 종속성 서비스의 중앙 집중화

빌더 팀이 소프트웨어 패키지 및 기타 종속성을 확보할 수 있도록 서비스를 중앙 집중화하세요. 서비스를 중앙 집중화하면 작성하는 소프트웨어에 포함하기 전에 패키지를 검증할 수 있습니다. 또한 조직에서 사용 중인 소프트웨어 분석에 쓰일 데이터를 하나의 소스에서 얻을 수 있습니다.

원하는 결과: 소프트웨어는 작성 중인 코드 외에 다양한 소프트웨어 패키지 세트로 구성됩니다. 따라서 JSON 구문 분석기 또는 암호화 라이브러리와 같이 반복적으로 사용되는 기능 구현을 간편하게 사용할 수 있습니다. 이러한 패키지 및 종속성에 대한 소스를 논리적으로 중앙 집중화하면 패키지를 사용하기 전에 패키지의 속성을 검증하는 메커니즘을 보안 팀에 제공할 수 있습니다. 또한 이러한 전략은 기존 패키지의 변경이나 빌더 팀(예: 인터넷에서 바로 임의 패키지 다운로드)으로 인해 예상치 못한 문제가 발생할 위험을 줄여줍니다. 수동 및 자동 테스트 흐름과 함께 이 전략을 사용하면 개발 중인 소프트웨어의 품질에 대한 신뢰도를 높일 수 있습니다.

일반적인 안티 패턴:

- 인터넷의 임의 리포지토리에서 패키지를 가져옵니다.
- 빌더에게 새 패키지를 제공하기 전에 테스트하지 않습니다.

이 모범 사례 확립의 이점:

- 구축 중인 소프트웨어에서 어떤 패키지가 사용되고 있는지 효과적으로 이해할 수 있습니다.

- 누가 무엇을 사용하고 있는지 파악한 후에 패키지를 업데이트해야 할 때 워크로드 팀에 알릴 수 있습니다.
- 소프트웨어에 문제가 포함된 패키지의 위험을 줄입니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

패키지 및 종속성에 대한 중앙 집중식 서비스를 빌더가 쉽게 사용할 수 있는 방식으로 제공합니다. 중앙 집중식 서비스는 단일 시스템으로 구현되기보다는 논리적으로 중앙에 배치될 수 있습니다. 이러한 접근 방식을 통해 빌더의 요구를 충족하는 방향으로 서비스를 제공할 수 있습니다. 업데이트가 발생하거나 새로운 요구 사항이 나타날 때 패키지를 리포지토리에 추가하는 효율적인 방법을 구현해야 합니다. AWS CodeArtifact와 같은 AWS 서비스 또는 이와 유사한 AWS 파트너 솔루션은 이러한 기능을 제공하는 방법을 안내합니다.

구현 단계:

- 소프트웨어가 개발되는 모든 환경에서 사용할 수 있는 논리적으로 중앙 집중화된 리포지토리 서비스를 구현합니다.
- 리포지토리에 대한 액세스를 AWS 계정 벤딩 프로세스의 일부로 포함합니다.
- 패키지를 리포지토리에 게시하기 전에 테스트하는 자동화를 구축합니다.
- 가장 일반적으로 사용되는 패키지, 언어 및 변경 사항이 제일 많은 팀의 지표를 유지 관리합니다.
- 빌더 팀이 새 패키지를 요청하고 피드백을 줄 수 있도록 자동화된 메커니즘을 제공합니다.
- 리포지토리의 패키지를 정기적으로 스캔하여 새로 발견된 문제의 잠재적 영향을 식별합니다.

리소스

관련 모범 사례:

- SEC11-BP02 개발 및 릴리스 수명 주기를 통한 테스트 자동화

관련 문서:

- Accelerate deployments on AWS with effective governance(효과적인 거버넌스를 통한 AWS의 배포 가속화)
- Tighten your package security with CodeArtifact Package Origin Control toolkit(CodeArtifact Package Origin Control 도구 키트로 패키지 보안 강화)

- Detecting security issues in logging with Amazon CodeGuru Reviewer(Amazon CodeGuru Reviewer를 사용한 로깅 내 보안 문제 탐지)
- Supply chain Levels for Software Artifacts (SLSA)(소프트웨어 아티팩트의 공급망 수준(SLSA))

관련 동영상:

- Proactive security: Considerations and approaches(사전 예방적 보안: 고려 사항 및 접근 방법)
- The AWS Philosophy of Security (re:Invent 2017)(AWS 보안 철학(re:Invent 2017))
- When security, safety, and urgency all matter: Handling Log4Shell(보안, 안전 및 긴급성이 모두 중요한 경우: Log4Shell 처리)

관련 예시:

- 다중 리전 패키지 게시 파이프라인(GitHub)
- AWS CodePipeline을 사용하여 AWS CodeArtifact에 Node.js 모듈 게시(GitHub)
- AWS CDK Java CodeArtifact 파이프라인 샘플(GitHub)
- AWS CodeArtifact를 사용한 프라이빗 .NET NuGet 패키지 배포(GitHub)

SEC11-BP06 프로그래밍 방식으로 소프트웨어 배포

가능한 한 프로그래밍 방식으로 소프트웨어를 배포하세요. 이 접근 방식을 통해 인적 오류로 배포 실패나 예기치 않은 문제가 발생할 가능성을 줄일 수 있습니다.

원하는 결과: AWS 클라우드에서의 안전한 구축을 위해서는 데이터에 대한 사람의 접근을 막는 것이 핵심 원칙입니다. 이러한 원칙에는 소프트웨어 배포 방법이 포함됩니다.

인력에 의존하여 소프트웨어를 배포하지 않으면 테스트한 것이 배포되고 매번 일관성 있게 배포가 이루어진다는 장점이 있습니다. 다른 환경에서 작동되도록 소프트웨어를 변경할 필요가 없습니다. 12가지 요소로 구성된 애플리케이션 개발의 원칙, 특히 구성의 외부화를 적용하면 변경하지 않고 동일한 코드를 여러 환경에 배포할 수 있습니다. 암호화된 서명 소프트웨어 패키지는 서로 다른 환경 간에 변경된 내용이 없음을 확인하는 좋은 방법입니다. 이 접근 방식을 통해 변경 프로세스의 위험을 줄이고 소프트웨어 릴리스의 일관성을 개선하는 결과를 얻을 수 있습니다.

일반적인 안티 패턴:

- 프로덕션에 소프트웨어를 수동으로 배포합니다.
- 다양한 환경에 맞게 소프트웨어를 수동으로 변경합니다.

이 모범 사례 확립의 이점:

- 소프트웨어 릴리스 프로세스에 대한 신뢰도가 높아집니다.
- 비즈니스 기능에 영향을 미치는 변경 실패 위험을 줄여줍니다.
- 변경 위험 감소로 인해 릴리스 주기가 길어집니다.
- 배포 중 예기치 않은 이벤트에 대한 자동 롤백 기능이 지원됩니다.
- 테스트된 소프트웨어가 배포된 소프트웨어임을 암호화하여 증명하는 기능이 제공됩니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

AWS 계정 구조를 구성하여 환경에서 영구적인 인적 액세스를 제거하고 CI/CD 도구를 사용하여 배포를 수행합니다. AWS Systems Manager Parameter Store와 같은 외부 소스에서 환경별 구성 데이터를 가져오도록 애플리케이션을 설계합니다. 패키지를 테스트한 후 서명하고, 배포하는 동안 서명을 검증합니다. 애플리케이션 코드를 푸시하도록 CI/CD 파이프라인을 구성하고 Canary를 사용하여 배포 성공을 확인합니다. AWS CloudFormation 또는 AWS CDK와 같은 도구를 사용하여 인프라를 정의한 다음, AWS CodeBuild 및 AWS CodePipeline을 사용하여 CI/CD 작업을 수행합니다.

구현 단계

- 효과적으로 정의된 CI/CD 파이프라인을 구축하여 배포 프로세스를 간소화합니다.
- AWS CodeBuild 및 AWS Code Pipeline을 사용하여 CI/CD 기능을 제공하면 보안 테스트를 파이프라인에 쉽게 통합할 수 있습니다.
- 여러 계정을 사용하여 AWS 환경 구성 백서의 환경 분리 지침을 따릅니다.
- 프로덕션 워크로드가 실행 중인 환경에 대한 영구적인 인적 액세스 권한이 없는지 확인합니다.
- 구성 데이터의 외부화를 지원하도록 애플리케이션을 설계합니다.
- 블루/그린 배포 모델을 사용한 배포를 고려합니다.
- Canary를 구현하여 소프트웨어의 배포 성공을 검증합니다.
- AWS Signer 또는 AWS Key Management Service(AWS KMS)와 같은 암호화 도구를 사용하여 배포 중인 소프트웨어 패키지에 서명하고 확인합니다.

리소스

관련 모범 사례:

- SEC11-BP02 개발 및 릴리스 수명 주기를 통한 테스트 자동화

관련 문서:

- AWS CI/CD Workshop(AWS CI/CD 워크숍)
- Accelerate deployments on AWS with effective governance(효과적인 거버넌스를 통한 AWS의 배포 가속화)
- 안전한 자동 배포 자동화
- Code signing using AWS Certificate Manager Private CA and AWS Key Management Service asymmetric keys(AWS Certificate Manager Private CA 및 AWS Key Management Service 비대칭 키를 사용한 코드 서명)
- Code Signing, a Trust and Integrity Control for AWS Lambda(코드 서명, AWS Lambda의 신뢰 및 무결성 제어)

관련 동영상:

- Hands-off: Automating continuous delivery pipelines at Amazon(자동: Amazon에서 지속적 전달 파이프라인 자동화)

관련 예시:

- Blue/Green deployments with AWS Fargate(AWS Fargate를 사용한 블루/그린 배포)

SEC11-BP07 정기적으로 파이프라인의 보안 속성 평가

특히 권한 분리에 주의를 기울여 Well-Architected 보안 원칙을 파이프라인에 적용하세요. 파이프라인 인프라의 보안 속성을 정기적으로 평가합니다. 파이프라인의 보안을 효율적으로 관리하면 파이프라인을 통과하는 소프트웨어의 보안을 보장할 수 있습니다.

원하는 결과: 소프트웨어를 구축하고 배포하는 데 사용되는 파이프라인은 환경의 다른 워크로드와 동일한 권장 사례를 따라야 합니다. 파이프라인에서 구현되는 테스트는 이를 사용하는 빌더가 편집할 수 없어야 합니다. 파이프라인은 빌더가 수행 중인 배포에 필요한 권한만 보유해야 하며, 잘못된 환경에 배포되지 않도록 안전 조치를 구현해야 합니다. 파이프라인은 장기 보안 인증 정보에 의존하지 않아야 하며, 구축 환경의 무결성을 확인할 수 있게 상태를 전송하도록 구성해야 합니다.

일반적인 안티 패턴:

- 빌더가 보안 테스트를 우회할 수 있습니다.
- 배포 파이프라인에 대한 권한이 지나치게 광범위합니다.
- 입력을 검증하도록 파이프라인을 구성하지 않습니다.
- CI/CD 인프라와 관련된 권한을 정기적으로 검토하지 않습니다.
- 장기 또는 하드코딩된 보안 인증 정보를 사용합니다.

이 모범 사례 확립의 이점:

- 파이프라인을 통해 구축 및 배포되는 소프트웨어의 무결성에 대한 신뢰도가 높아집니다.
- 의심스러운 활동이 있을 때 배포를 중지할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

IAM 역할을 지원하는 관리형 CI/CD 서비스로 시작하면 보안 인증 정보 유출 위험이 줄어듭니다. CI/CD 파이프라인 인프라에 보안 원칙을 적용하면 보안을 개선할 수 있는 부분을 가려내는 데 도움이 됩니다. CI/CD 환경을 구축하기 시작할 때 AWS 배포 파이프라인 참조 아키텍처를 따르는 것이 좋습니다. 파이프라인 구현을 정기적으로 검토하고 예기치 않은 동작에 대한 로그를 분석하면 소프트웨어 배포에 사용되는 파이프라인의 사용 패턴을 이해하는 데 유용합니다.

구현 단계

- 먼저 AWS 배포 파이프라인 참조 아키텍처부터 시작합니다.
- 파이프라인에 대한 최소 권한 IAM 정책을 프로그래밍 방식으로 생성하려면 AWS IAM Access Analyzer를 사용하는 것이 좋습니다.
- 파이프라인을 모니터링 및 알림과 통합하여 예기치 않거나 비정상적인 활동이 발생할 경우 알림을 받을 수 있습니다. AWS 관리형 서비스의 경우 Amazon EventBridge를 사용하면 AWS Lambda 또는 Amazon Simple Notification Service(Amazon SNS)와 같은 대상으로 데이터를 라우팅할 수 있습니다.

리소스

관련 문서:

- AWS 배포 파이프라인 참조 아키텍처

- Monitoring AWS CodePipeline(AWS CodePipeline 모니터링)
- Security best practices for AWS CodePipeline(AWS CodePipeline 보안 모범 사례)

관련 예시:

- DevOps 모니터링 대시보드(GitHub)

SEC11-BP08 보안 소유권이 워크로드 팀에 귀속되도록 프로그램 구축

빌더 팀이 개발하는 소프트웨어의 보안을 결정할 수 있도록 지원하는 프로그램 또는 메커니즘을 구축합니다. 보안 팀에서 검토 시 이러한 결정을 다시금 검증해야 하지만, 빌더 팀이 보안 소유권을 가지면 더욱 신속하고 안전하게 워크로드를 구축할 수 있습니다. 또한 이 메커니즘은 구축하는 시스템의 운영에 좋은 영향을 미치는 주인 의식 문화를 강화합니다.

원하는 결과: 보안 소유권과 의사 결정을 빌더 팀에 귀속하려면 빌더에게 보안을 대하는 방식을 교육하면 됩니다. 빌더 팀에 속하거나 연계된 보안 담당자를 활용하여 빌더 대상 교육을 강화할 수도 있습니다. 두 전략 모두 유효하며, 이를 통해 빌더 팀이 개발 주기 초기에 더 현명한 보안 결정을 내리도록 지원할 수 있습니다. 이 소유권 모델은 애플리케이션 보안 교육을 기반으로 합니다. 특정 워크로드에 대한 위협 모델부터 시작하면 적절한 컨텍스트에 초점을 맞춰 설계 사고를 집중할 수 있습니다. 보안에 주력하는 빌더 커뮤니티 또는 빌더 팀과 협력하는 보안 엔지니어 그룹이 있으면 소프트웨어가 작성되는 방식을 보다 깊이 이해할 수 있다는 이점도 있습니다. 이러한 이해도를 바탕으로 자동화 기능의 다음 개선 영역을 결정할 수 있습니다.

일반적인 안티 패턴:

- 보안 팀에 모든 보안 설계 결정을 맡깁니다.
- 개발 프로세스에서 보안 요구 사항을 조기에 해결하지 못합니다.
- 빌더 및 보안 담당자로부터 프로그램 운영 피드백을 받지 못합니다.

이 모범 사례 확립의 이점:

- 보안 검토를 빠르게 완료할 수 있습니다.
- 보안 검토 단계에서만 탐지되는 보안 문제가 감소합니다.
- 작성 중인 소프트웨어의 전반적인 품질이 향상됩니다.
- 체계 문제 또는 유의미한 개선 영역을 식별하고 이해할 수 있습니다.
- 보안 검토 결과로 인해 필요한 재작업의 양이 줄어듭니다.

- 보안 기능에 대한 인식이 개선됩니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

SEC11-BP01 애플리케이션 보안 교육의 지침부터 시작합니다. 그런 다음 조직에 가장 적합하다고 생각되는 프로그램의 운영 모델을 파악합니다. 2가지 주요 패턴은 빌더를 교육하거나 빌더 팀에 보안 인력을 투입하는 것입니다. 초기 접근 방식을 정한 후에는 단일 또는 소규모 워크로드 팀과 함께 시범 운영하여 해당 모델이 조직에 적합한지 입증해야 합니다. 조직의 빌더 및 보안 부문에서 리더십의 지원이 있으면 프로그램의 제공과 성공에 도움이 됩니다. 이 프로그램을 개발할 때 프로그램의 가치를 보여주는 데 사용할 수 있는 지표를 선택해야 합니다. AWS가 이러한 문제에 어떻게 접근했는지 알아보면 큰 도움이 됩니다. 이 모범 사례는 조직의 변화와 문화를 집중적으로 조명합니다. 빌더와 보안 커뮤니티 간의 협업을 지원하는 도구를 사용해야 합니다.

구현 단계

- 먼저 빌더에게 애플리케이션 보안 교육을 제공합니다.
- 빌더를 교육하기 위한 커뮤니티와 온보딩 프로그램을 개발합니다.
- 프로그램 이름을 선택합니다. Guardians, Champions, Advocates가 주로 사용됩니다.
- 빌더를 교육하거나, 보안 엔지니어를 합류시키거나, 보안 담당자를 연계하는 등 사용할 모델을 결정합니다.
- 보안, 빌더 및 기타 관련 그룹의 프로젝트 후원 주체를 결정합니다.
- 프로그램에 참여한 인원 수, 검토에 소요된 시간, 빌더 및 보안 인력의 피드백 지표를 추적합니다. 이러한 지표를 바탕으로 개선합니다.

리소스

관련 모범 사례:

- SEC11-BP01 애플리케이션 보안 교육
- SEC11-BP02 개발 및 릴리스 수명 주기를 통한 테스트 자동화

관련 문서:

- How to approach threat modeling(위협 모델링 접근 방식)
- How to think about cloud security governance(클라우드 보안 거버넌스를 대하는 방식)

관련 동영상:

- Proactive security: Considerations and approaches(사전 예방적 보안: 고려 사항 및 접근 방법)

신뢰성

안정성 원칙에서는 워크로드의 기능이 필요한 때에 기능을 정확하고 일관되게 수행하는 역량에 대해 다룹니다. 구현 방법에 대한 권장 가이드는 신뢰성 원칙 백서에서 확인할 수 있습니다.

모범 사례 영역

- 기반
- 워크로드 아키텍처
- 변경 사항 관리
- 장애 관리

기반

질문

- REL 1 서비스 할당량과 제약 조건은 어떻게 관리합니까?
- REL 2 네트워크 토폴로지는 어떻게 계획합니까?

REL 1 서비스 할당량과 제약 조건은 어떻게 관리합니까?

클라우드 기반 워크로드 아키텍처에는 서비스 할당량(서비스 한도라고도 함)이라는 것이 있습니다. 이러한 할당량은 실수로 필요한 것보다 많은 리소스를 프로비저닝하는 것을 방지하고 API 작업에 대한 요청 속도를 제한하여 서비스를 침해로부터 보호하기 위해 존재합니다. 광섬유 케이블을 통해 비트를 전송할 수 있는 속도나 물리적 디스크의 스토리지 양과 같은 리소스 제약 조건도 있습니다.

모범 사례

- REL01-BP01 Service Quotas 및 제약 조건 인식
- REL01-BP02 계정 및 리전 전체에서 서비스 할당량 관리
- REL01-BP03 아키텍처를 통해 고정된 서비스 할당량 및 제약 조건 수용
- REL01-BP04 할당량 모니터링 및 관리
- REL01-BP05 할당량 관리 자동화

- REL01-BP06 현재의 할당량과 최대 사용량 간에 장애 조치를 수용할 만큼 여유가 충분히 있는지 확인

REL01-BP01 Service Quotas 및 제약 조건 인식

워크로드 아키텍처에 대한 기본 할당량을 파악하고 할당량 증가 요청을 관리합니다. 디스크 또는 네트워크 등 어떤 클라우드 리소스 제약 조건이 잠재적인 영향을 미치는지 파악합니다.

원하는 결과: 고객은 서비스 성능 저하 또는 중단을 일으킬 수 있는 서비스 할당량 및 제약 조건에 도달했는지 확인하기 위해 주요 지표, 인프라 검토 및 자동화 개선 조치 단계를 모니터링하기 위한 적절한 지침을 구현하여 AWS 계정에서 서비스 성능 저하 또는 중단을 방지할 수 있습니다.

일반적인 안티 패턴:

- 하드 또는 소프트 할당량과 사용되는 서비스에 대한 한도를 파악하지 않고 워크로드를 배포합니다.
- 필요한 할당량을 분석 및 재구성하거나 미리 지원 팀에 연락하지 않고 대체 워크로드를 배포합니다.
- 클라우드 서비스에는 한도가 없고 속도, 한도, 횟수, 수량 등을 고려하지 않고 서비스를 사용할 수 있다고 가정합니다.
- 할당량이 자동으로 증가할 것이라고 가정합니다.
- 할당량 요청의 프로세스 및 타임라인을 모릅니다.
- 기본 클라우드 서비스 할당량이 리전 간에 비교되는 모든 서비스에 대해 동일하다고 가정합니다.
- 서비스 제약 조건은 위반할 수 있고 시스템에서 리소스의 제약 조건을 벗어나 자동으로 스케일링하거나 한도 증가를 추가한다고 가정합니다.
- 리소스 사용률을 강조하기 위해 피크 트래픽에서 애플리케이션을 테스트하지 않습니다.
- 필요한 리소스 크기를 분석하지 않고 리소스를 프로비저닝합니다.
- 실제 필요 또는 예상 피크를 잘 벗어나는 리소스 유형을 선택하여 용량을 오버프로비저닝합니다.
- 새로운 고객 이벤트 또는 새로운 기술 배포에 앞서 새로운 수준의 트래픽에 대한 용량 요구 사항을 미리 평가하지 않습니다.

이 모범 사례 확립의 이점: 서비스 할당량 및 리소스 제약 조건을 모니터링하고 관리를 자동화하면 사전에 실패를 줄일 수 있습니다. 모범 사례를 따르지 않으면 고객 서비스에 대한 트래픽 패턴의 변화로 인해 서비스 중단 또는 성능 저하가 발생할 수 있습니다. 모든 리전과 계정에서 이러한 값을 모니터링 및 관리하여 애플리케이션이 불리하거나 계획되지 않은 이벤트 발생 시 복원력을 개선할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

Service Quotas는 250개 이상의 AWS 서비스에 대한 할당량을 단일 위치에서 관리할 수 있는 AWS 서비스입니다. 할당량 값을 조회하는 것에 더해 Service Quotas 콘솔 또는 AWS SDK를 사용하여 할당량 증가를 요청하고 추적할 수 있습니다. AWS Trusted Advisor는 일부 서비스의 특정 측면에 대한 사용량 및 할당량을 표시하는 서비스 할당량 확인 기능을 제공합니다. 각 서비스의 기본 서비스 할당량은 해당하는 서비스의 AWS 설명서에도 문서화되어 있습니다. 예를 들어 Amazon VPC 할당량을 참조하세요.

일부 서비스 한도(예: 조절된 API에 대한 속도 제한)은 Amazon API Gateway 자체에서 사용량 계획을 구성하는 방법으로 설정됩니다. 구성을 통해 해당하는 서비스에 설정되는 기타 한도로는 프로비저닝된 IOPS, 할당된 Amazon RDS 스토리지 및 Amazon EBS 볼륨 할당이 있습니다. Amazon Elastic Compute Cloud에는 인스턴스, Amazon Elastic Block Store 및 탄력적 IP 주소 제한을 관리하는 데 도움이 되는 자체 서비스 한도 대시보드가 있습니다. 서비스 할당량이 애플리케이션 성능에 영향을 미치고 요구 사항에 맞춰 조정할 수 없는 사용 사례가 있는 경우 AWS Support에 문의하여 완화 방법이 있는지 확인하세요.

서비스 할당량은 리전에 따라 다를 수 있으며 특성상 전역적일 수도 있습니다. 할당량에 도달한 AWS 서비스를 사용하면 정상적인 사용 시에도 예상대로 작동하지 않을 수 있으며 서비스 중단 또는 성능 저하를 일으킬 수 있습니다. 예를 들어, 서비스 할당량은 하나의 리전에서 사용되는 DL Amazon EC2 수를 제한하며, Auto Scaling 그룹(ASG)을 사용하여 트래픽 스케일링 중에 이러한 수량 제한에 도달할 수 있습니다.

각 계정에 대한 서비스 할당량은 정기적으로 사용에 대해 평가해 해당 계정에 대해 적절한 서비스 한도를 확인해야 합니다. 서비스 할당량은 필요한 것보다 더 많은 리소스를 실수로 프로비저닝하지 않기 위한 운영 가드레일로 존재합니다. 또한 서비스의 남용을 방지하기 위해 API 작업에 대한 요청 속도를 제한하기도 합니다.

서비스 제약 조건은 서비스 할당량과 다릅니다. 서비스 제약 조건은 서비스 유형에 따라 정의된 특정 리소스 한도를 나타냅니다. 여기에는 스토리지 용량(예: gp2에는 1GB~16TB의 크기 제한이 있음) 또는 디스크 처리량(10,000 IOPS)이 있을 수 있습니다. 한도에 도달할 수 있는 사용량에 대해서는 리소스 유형의 제약 조건을 엔지니어링하고 지속적으로 평가해야 합니다. 예기치 않게 제약 조건에 도달한 경우 계정의 애플리케이션 또는 서비스가 성능이 저하되거나 중단될 수 있습니다.

서비스 할당량이 애플리케이션 성능에 영향을 미치는데 요구 사항에 맞춰 조정할 수 없는 사용 사례가 있는 경우 AWS Support에 문의하여 완화 방법이 있는지 확인하세요. 고정 할당량 조정에 대한 자세한 내용은 REL01-BP03 아키텍처를 통해 고정된 서비스 할당량 및 제약 조건 수용의 내용을 참조하세요.

Service Quotas 모니터링 및 관리를 지원하기 위한 여러 가지 AWS 서비스 및 도구가 있습니다. 할당량 수준을 자동으로 또는 수동으로 확인하려면 이러한 서비스 및 도구를 활용해야 합니다.

- AWS Trusted Advisor는 일부 서비스의 특정 측면에 대한 사용량 및 할당량을 표시하는 서비스 할당량 확인 기능을 제공합니다. 따라서 거의 할당량에 도달한 서비스를 식별하는 데 도움이 됩니다.
- AWS Management Console에서는 서비스 할당량 값을 표시하고, 새로운 할당량을 관리 및 요청하고, 할당량 요청 상태를 모니터링하고, 할당량 이력을 표시하는 방법을 제공합니다.
- AWS CLI 및 CDK에서는 서비스 할당량 수준과 사용을 자동으로 관리 및 모니터링하기 위한 프로그래밍 방식을 제공합니다.

구현 단계

Service Quotas의 경우:

- AWS Service Quotas를 검토합니다.
- 기존 서비스 할당량을 파악하려면 사용되는 서비스(예: IAM Access Analyzer)를 확인합니다. 서비스 할당량으로 제어되는 AWS 서비스는 약 250개가 있습니다. 그런 다음, 각 계정 및 리전 내에서 사용 가능한 특정 서비스 할당량 이름을 확인합니다. 리전당 약 3천개의 서비스 할당량 이름이 있습니다.
- AWS 계정에서 사용되는 AWS 리소스를 모두 찾기 위해 AWS Config을 사용하여 할당량 분석을 강화합니다.
- AWS CloudFormation 데이터를 사용하여 사용된 AWS 리소스를 확인합니다. AWS Management Console에서 또는 `list-stack-resources` AWS CLI 명령을 통해 생성된 리소스를 확인합니다. 템플릿 자체에 배포하도록 구성된 리소스를 볼 수도 있습니다.
- 배포 코드를 확인하여 워크로드에 필요한 모든 서비스를 결정합니다.
- 적용되는 서비스 할당량을 확인합니다. Trusted Advisor 및 Service Quotas에서 프로그래밍 방식으로 액세스되는 정보를 활용합니다.
- 서비스 할당량이 한계에 근접하거나 도달하면 알리도록 자동화된 모니터링 방법을 설정합니다(REL01-BP02 계정 및 리전 전체에서 서비스 할당량 관리 및 REL01-BP04 할당량 모니터링 및 관리 참조).
- 동일한 계정 내에서 서비스 할당량이 한 리전에서는 변경되었지만 다른 리전에서는 변경되지 않은 경우를 확인하도록 자동화된 프로그래밍 방식을 설정합니다(REL01-BP02 계정 및 리전 전체에서 서비스 할당량 관리 및 REL01-BP04 할당량 모니터링 및 관리 참조).
- 할당량 또는 서비스 제약 조건 오류가 있는지 확인하도록 애플리케이션 로그 및 지표 검사를 자동화합니다. 이러한 오류가 있는 경우 모니터링 시스템에 알림을 보냅니다.
- 특정 서비스에 더 큰 할당량이 필요한 것이 확인되면 필요한 할당량 변경을 계산하기 위한 엔지니어링 절차를 수립합니다(REL01-BP05 할당량 관리 자동화).
- 서비스 할당량 변경을 요청하기 위한 프로비저닝 및 승인 워크플로를 생성합니다. 여기에는 요청 거부 또는 부분 승인 시 예외 워크플로를 포함해야 합니다.

- 프로덕션 또는 로드된 환경으로 롤아웃하기 전에 새로운 AWS 서비스를 프로비저닝하고 사용하기에 앞서 서비스 할당량을 검토하는 엔지니어링 방법을 생성합니다(예: 로드 테스트 계정).

서비스 제약 조건:

- 리소스 제약 조건에 근접한 리소스에 대해 경고하는 모니터링 및 지표 방법을 설정합니다. CloudWatch를 지표 또는 로그 모니터링에 적절하게 활용합니다.
- 애플리케이션 또는 시스템에 의미 있는 제약 조건이 있는 각 리소스에 대해 알림 임계값을 설정합니다.
- 제약 조건에 거의 도달하면 리소스 유형을 변경하는 워크플로 및 인프라 관리 절차를 생성합니다. 이 워크플로에는 새 유형이 새로운 제약 조건이 있는 올바른 리소스 유형인지 확인하기 위한 모범 사례로 로드 테스트를 포함해야 합니다.
- 기존 절차 및 프로세스를 사용하여 식별된 리소스를 권장되는 새 리소스 유형으로 마이그레이션합니다.

리소스

관련 모범 사례:

- REL01-BP02 계정 및 리전 전체에서 서비스 할당량 관리
- REL01-BP03 아키텍처를 통해 고정된 서비스 할당량 및 제약 조건 수용
- REL01-BP04 할당량 모니터링 및 관리
- REL01-BP05 할당량 관리 자동화
- REL01-BP06 현재의 할당량과 최대 사용량 간에 장애 조치를 수용할 만큼 여유가 충분히 있는지 확인
- REL03-BP01 워크로드를 세그먼트화하는 방법 선택
- REL10-BP01 워크로드를 여러 위치에 배포
- REL11-BP01 워크로드의 모든 구성 요소를 모니터링하여 장애 감지
- REL11-BP03 모든 계층에서 복구 자동화
- REL12-BP05 카오스 엔지니어링을 이용한 복원력 테스트

관련 문서:

- AWS Well-Architected Framework의 신뢰성 원칙: 가용성

- AWS Service Quotas(이전 명칭: 서비스 한도)
- AWS Trusted Advisor 모범 사례 점검 항목('서비스 한도' 섹션 참조)
- AWS Answers의 AWS Limit Monitor
- Amazon EC2 서비스 한도
- Service Quotas란 무엇인가요?
- 할당량 증가 요청 방법
- 서비스 엔드포인트 및 할당량
- Service Quotas 사용 설명서
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries(AWS 장애 격리 경계)
- Availability with redundancy(중복성이 적용된 가용성)
- AWS for Data
- 지속적 통합이란 무엇인가요?
- 지속적 전달이란 무엇인가요?
- APN 파트너: 구성 관리를 지원할 수 있는 파트너
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS(AWS의 테넌트당 계정 SaaS 환경에서 계정 수명 주기 관리)
- 워크로드에서 API 제한 관리 및 모니터링
- View AWS Trusted Advisor recommendations at scale with AWS Organizations(AWS Organizations를 사용하여 대규모로 AWS Trusted Advisor 권장 사항 보기)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower(AWS Control Tower를 사용하여 서비스 한도 증가 및 엔터프라이즈 지원 자동화)

관련 동영상:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas(Service Quotas를 사용하여 AWS 서비스에 대한 할당량 보기 및 관리)
- AWS IAM 할당량 데모

관련 도구:

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP02 계정 및 리전 전체에서 서비스 할당량 관리

여러 계정 또는 리전을 사용하는 경우 프로덕션 워크로드가 실행되는 모든 환경에서 적절한 할당량을 요청해야 합니다.

원하는 결과: 서비스 및 애플리케이션은 여러 계정 또는 리전에 적용되는 구성 또는 영역, 리전 또는 계정 장애 조치를 사용하는 탄력적인 설계의 구성으로 인한 서비스 할당량 소진의 영향을 받지 않아야 합니다.

일반적인 안티 패턴:

- 다른 격리 영역에서 용량을 유지하는 메커니즘 없이 한 격리 리전의 리소스 사용량을 확장하도록 허용합니다.
- 격리 리전에서 모든 할당량을 독립적으로 수동으로 설정합니다.
- 기본 리전이 아닌 리전에서 성능이 저하되는 동안 향후 필요한 할당량에 복원력 아키텍처(액티브 또는 패시브)의 영향을 고려하지 않습니다.
- 할당량을 정기적으로 평가하지 않고 워크로드가 실행되는 모든 리전 및 계정에서 필요한 변경을 수행하지 않습니다.
- 여러 리전 및 계정 간에 증가를 요청하는 데 할당량 요청 템플릿을 사용하지 않습니다.
- 할당량 증가가 컴퓨팅 예약 요청과 같이 비용에 영향을 미친다고 잘못 생각하여 서비스 할당량을 업데이트하지 않습니다.

이 모범 사례 확립의 이점: 리전별 서비스를 사용할 수 없는 경우 보조 리전 또는 계정에서 현재 로드를 처리할 수 있는지 확인합니다. 이는 리전 손실 중 발생하는 오류 수 또는 성능 저하 수준을 줄이는 데 도움이 됩니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

서비스 할당량은 계정별로 추적됩니다. 다른 언급이 없는 한, 각 할당량은 AWS 리전별로 다릅니다. 프로덕션 환경에 더해 적용 가능한 모든 비 프로덕션 환경에서도 할당량을 관리하여 테스트 및 개발에 방해가 되지 않도록 합니다. 높은 수준의 복원력을 유지하려면 지속적으로 서비스 할당량을 (자동 또는 수동으로) 평가해야 합니다.

액티브/액티브, 액티브/패시브 – 핫, 액티브/패시브-콜드 및 액티브/패시브-파일럿 라이트 접근 방식을 사용하는 설계의 구현으로 인해 여러 리전에 걸쳐 워크로드가 증가하는 경우 모든 리전 및 계정 할당량 수준을 파악해야 합니다. 서비스 할당량이 올바르게 설정되어 있더라도 과거 트래픽 패턴이 항상 좋은 지표는 아닙니다.

서비스 할당량 이름 제한이 모든 리전에 대해 항상 같은 것도 아닙니다. 한 리전에서 이 값은 5일 수 있으며 다른 리전에서는 10일 수 있습니다. 로드 발생 시 일정한 복원력을 제공하려면 이러한 할당량 관리는 동일한 서비스, 계정, 리전을 모두 포함해야 합니다.

여러 리전(액티브 리전 또는 패시브 리전) 간에 모든 서비스 할당량 차이를 조정하고 이러한 차이를 지속적으로 조정하기 위한 프로세스를 생성합니다. 패시브 리전 장애 조치의 테스트 계획은 피크 액티브 용량으로 확장되는 경우가 거의 없습니다. 즉, 게임 데이 또는 탁상 훈련(TTX) 방식은 리전 간 서비스 할당량의 차이를 찾지 못할 수 있고 올바른 한도를 유지하지 못할 수 있습니다.

지정된 특정 할당량에 대한 서비스 할당량 제한이 모든 리전이 아니라 한 리전에서 변경되는 조건인 서비스 할당량 드리프트는 추적 및 평가해야 하는 매우 중요한 조건입니다. 트래픽이 있는 리전 또는 트래픽이 발생할 수 있는 리전에서는 할당량 변경을 고려해야 합니다.

- 서비스 요구 사항, 지연 시간, 규정, 재해 복구(DR) 요구 사항을 기준으로 관련 계정 및 리전을 선택합니다.
- 모든 관련 계정, 리전 및 가용 영역의 서비스 할당량을 확인합니다. 한도는 계정 및 리전별로 관리됩니다. 이러한 값은 차이가 있는지 비교해야 합니다.

구현 단계

- 사용 위험 수준을 벗어나 위반될 수 있는 Service Quotas 값을 검토합니다. AWS Trusted Advisor에서는 80% 및 90% 임계값 위반에 대한 알림을 제공합니다.

- (액티브/패시브 설계인 경우) 모든 패시브 리전에서 서비스 할당량에 대한 값을 검토합니다. 기본 리전에서 장애 발생 시 보조 리전에서 로드가 성공적으로 실행되는지 확인합니다.
- 동일한 계정 내 리전 간에 서비스 할당량 드리프트가 발생했는지 여부 평가를 자동화하고 한도 변경에 적절하게 대응합니다.
- 고객 조직 단위(OU)가 지원되는 방식으로 구성되어 있으면 여러 리전 및 계정에 적용해야 하는 모든 할당량의 변화를 반영하도록 서비스 할당량 템플릿을 업데이트해야 합니다.
  - 템플릿을 생성하고 할당량 변경에 리전을 연결합니다.
  - 필요한 모든 변경(리전, 한도 및 계정)에 대한 기존 서비스 할당량 템플릿을 모두 검토합니다.

리소스

관련 모범 사례:

- REL01-BP01 Service Quotas 및 제약 조건 인식
- REL01-BP03 아키텍처를 통해 고정된 서비스 할당량 및 제약 조건 수용
- REL01-BP04 할당량 모니터링 및 관리
- REL01-BP05 할당량 관리 자동화
- REL01-BP06 현재의 할당량과 최대 사용량 간에 장애 조치를 수용할 만큼 여유가 충분히 있는지 확인
- REL03-BP01 워크로드를 세그먼트화하는 방법 선택
- REL10-BP01 워크로드를 여러 위치에 배포
- REL11-BP01 워크로드의 모든 구성 요소를 모니터링하여 장애 감지
- REL11-BP03 모든 계층에서 복구 자동화
- REL12-BP05 카오스 엔지니어링을 이용한 복원력 테스트

관련 문서:

- AWS Well-Architected Framework의 신뢰성 원칙: 가용성
- AWS Service Quotas(이전 명칭: 서비스 한도)
- AWS Trusted Advisor 모범 사례 점검 항목('서비스 한도' 섹션 참조)
- AWS Answers의 AWS Limit Monitor
- Amazon EC2 서비스 한도

- Service Quotas란 무엇인가요?
- 할당량 증가 요청 방법
- 서비스 엔드포인트 및 할당량
- Service Quotas 사용 설명서
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries(AWS 장애 격리 경계)
- Availability with redundancy(중복성이 적용된 가용성)
- AWS for Data
- 지속적 통합이란 무엇인가요?
- 지속적 전달이란 무엇인가요?
- APN 파트너: 구성 관리를 지원할 수 있는 파트너
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS(AWS의 테넌트당 계정 SaaS 환경에서 계정 수명 주기 관리)
- 워크로드에서 API 제한 관리 및 모니터링
- View AWS Trusted Advisor recommendations at scale with AWS Organizations(AWS Organizations를 사용하여 대규모로 AWS Trusted Advisor 권장 사항 보기)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower(AWS Control Tower를 사용하여 서비스 한도 증가 및 엔터프라이즈 지원 자동화)

관련 동영상:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas(Service Quotas를 사용하여 AWS 서비스에 대한 할당량 보기 및 관리)
- AWS IAM 할당량 데모

관련 서비스:

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail

- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP03 아키텍처를 통해 고정된 서비스 할당량 및 제약 조건 수용

변경할 수 없는 서비스 할당량, 서비스 제약 조건 및 물리적 리소스 한도를 알고 있어야 합니다. 이러한 한도가 신뢰성에 영향을 미치지 않도록 애플리케이션 및 서비스의 아키텍처를 설계합니다.

네트워크 대역폭, 서버리스 기능 호출 페이로드 크기, API 게이트웨이의 스로틀 버스트 속도 및 데이터베이스에 대한 동시 사용자 연결 등이 여기에 포함됩니다.

원하는 결과: 애플리케이션 또는 서비스는 정상 조건 및 트래픽이 많은 조건에서 예상대로 수행됩니다. 해당 리소스의 고정 제약 조건 또는 서비스 할당량의 한계 내에서 작동하도록 설계되었습니다.

일반적인 안티 패턴:

- 확장 시 해당 설계에서 장애를 유발하는 설계 제약 조건이 있다는 사실을 인지하지 못한 채 하나의 서비스 리소스를 사용하는 설계 방식을 선택합니다.
- 비현실적이며 테스트 중에 고정된 서비스 할당량에 도달하는 벤치마킹을 수행합니다. 예를 들어, 버스트 한도에서 테스트를 실행하면서, 오랜 시간 실행합니다.
- 고정된 서비스 할당량을 초과하는 경우 확장할 수 없거나 수정할 수 없는 설계를 선택합니다. 예를 들어, SQS 페이로드 크기는 256kb입니다.
- 높은 트래픽 이벤트 동안 위험에 처할 수 있는 서비스 할당량의 임계값을 모니터링하고 경고하도록 관측성이 설계 및 구현되지 않았습니다.

이 모범 사례 확립의 이점: 애플리케이션이 중단이나 성능 저하 없이 예상되는 모든 서비스 부하 수준에서 실행되는지 확인합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

더 큰 용량 단위로 대체되는 소프트 서비스 할당량 또는 리소스와 달리 AWS 서비스의 고정 할당량은 변경할 수 없습니다. 따라서 이러한 모든 유형의 AWS 서비스는 애플리케이션 설계에 사용될 때 잠재적인 하드 용량 한도에 대해 평가되어야 합니다.

하드 한도는 Service Quotas 콘솔에 표시됩니다. 열에 `ADJUSTABLE = No`가 표시되면 서비스에 하드 한도가 있는 것입니다. 하드 한도는 일부 리소스 구성 페이지에도 표시됩니다. 예를 들어 Lambda에는 조정할 수 없는 특정 하드 한도가 있습니다.

예를 들어 Python 애플리케이션을 Lambda 함수에서 실행하도록 설계할 때 Lambda가 15분 이상 실행될 가능성이 있는지 확인하기 위해 애플리케이션을 평가해야 합니다. 코드가 이 서비스 할당량 한도보다 더 오래 실행될 수 있는 경우 대체 기술 또는 설계를 고려해야 합니다. 프로덕션 배포 후 이 한도에 도달하면 애플리케이션은 문제를 해결할 수 있을 때까지 성능 저하 및 중단이 발생합니다. 소프트 할당량과 달리 긴급 심각도 1 이벤트에서도 이러한 한도를 변경할 수 있는 방법이 없습니다.

애플리케이션이 테스트 환경에 배포되면 하드 한도에 도달할 수 있는지 확인하기 위한 전략을 사용해야 합니다. 스트레스 테스트, 부하 테스트 및 카오스 테스트는 도입 테스트 계획의 일부여야 합니다.

구현 단계

- 애플리케이션 설계 단계에서 사용할 수 있는 AWS 서비스의 전체 목록을 검토합니다.
- 이러한 모든 서비스에 대한 소프트 할당량 한도 및 하드 할당량 한도를 검토합니다. 모든 한도가 Service Quotas 콘솔에 표시되는 것은 아닙니다. 일부 서비스는 대체 위치에서 이러한 한도를 설명합니다.
- 애플리케이션을 설계할 때 비즈니스 결과, 사용 사례, 종속 시스템, 가용성 목표 및 재해 복구 개체와 같은 워크로드의 비즈니스 및 기술 동인을 검토합니다. 비즈니스 및 기술 동인이 워크로드에 적합한 분산 시스템을 식별하는 프로세스를 안내하도록 합니다.
- 리전 및 계정 전체에서 서비스 부하를 분석합니다. 서비스의 많은 하드 한도가 리전을 기반으로 합니다. 그러나 일부 한도는 계정을 기반으로 합니다.
- 영역 장애 및 리전 장애 시 리소스 사용량에 대한 복원력 아키텍처를 분석합니다. 액티브/액티브, 액티브/패시브 - 핫, 액티브/패시브 - 콜드 및 액티브/패시브 - 파일럿 라이트 접근 방식을 사용하는 다중 리전 설계의 진행에서 이러한 실패 사례는 더 높은 사용량을 야기할 것입니다. 이는 하드 한도에 도달할 수 있는 잠재적 사용 사례를 생성합니다.

리소스

관련 모범 사례:

- REL01-BP01 Service Quotas 및 제약 조건 인식
- REL01-BP02 계정 및 리전 전체에서 서비스 할당량 관리
- REL01-BP04 할당량 모니터링 및 관리
- REL01-BP05 할당량 관리 자동화
- REL01-BP06 현재의 할당량과 최대 사용량 간에 장애 조치를 수용할 만큼 여유가 충분히 있는지 확인
- REL03-BP01 워크로드를 세그먼트화하는 방법 선택
- REL10-BP01 워크로드를 여러 위치에 배포
- REL11-BP01 워크로드의 모든 구성 요소를 모니터링하여 장애 감지
- REL11-BP03 모든 계층에서 복구 자동화
- REL12-BP05 카오스 엔지니어링을 이용한 복원력 테스트

관련 문서:

- AWS Well-Architected Framework의 신뢰성 원칙: 가용성
- AWS Service Quotas(이전 명칭: 서비스 한도)
- AWS Trusted Advisor 모범 사례 점검 항목('서비스 한도' 섹션 참조)
- AWS Answers의 AWS Limit Monitor
- Amazon EC2 서비스 한도
- Service Quotas란 무엇인가요?
- 할당량 증가 요청 방법
- 서비스 엔드포인트 및 할당량
- Service Quotas 사용 설명서
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries(AWS 장애 격리 경계)
- Availability with redundancy(중복성이 적용된 가용성)
- AWS for Data
- 지속적 통합이란 무엇인가요?
- 지속적 전달이란 무엇인가요?
- APN 파트너: 구성 관리를 지원할 수 있는 파트너

- Managing the account lifecycle in account-per-tenant SaaS environments on AWS(AWS의 테넌트당 계정 SaaS 환경에서 계정 수명 주기 관리)
- 워크로드에서 API 제한 관리 및 모니터링
- View AWS Trusted Advisor recommendations at scale with AWS Organizations(AWS Organizations를 사용하여 대규모로 AWS Trusted Advisor 권장 사항 보기)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower(AWS Control Tower를 사용하여 서비스 한도 증가 및 엔터프라이즈 지원 자동화)
- Service Quotas에 사용되는 작업, 리소스 및 조건 키

관련 동영상:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas(Service Quotas를 사용하여 AWS 서비스에 대한 할당량 보기 및 관리)
- AWS IAM 할당량 데모
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small(루프 닫기 및 마음 열기: 규모에 상관없이 시스템을 제어하는 방법)

관련 도구:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP04 할당량 모니터링 및 관리

잠재적 사용량을 평가하고 할당량을 적절히 늘려 사용량 증가를 계획합니다.

원하는 결과: 관리 및 모니터링하는 액티브 자동화된 시스템이 배치되었습니다. 이러한 운영 솔루션은 할당량 사용량 임계값에 거의 도달하고 있는지 확인합니다. 이러한 문제는 요청된 할당량 변경에 의해 사전에 해결됩니다.

일반적인 안티 패턴:

- 서비스 할당량 임계값을 확인하도록 모니터링을 구성하지 않습니다.
- 해당 값을 변경할 수 없는 경우에도 하드 한도에 대한 모니터링을 구성하지 않습니다.
- 소프트 할당량 변경을 요청하고 확보하는 데 필요한 시간이 즉각적이거나 짧은 기간이라고 가정합니다.
- 서비스 할당량에 근접할 경우 알리는 경보를 구성하지만, 알림에 응답하는 프로세스를 갖추지 않습니다.
- AWS Service Quotas에서 지원하는 서비스에 대해서만 경보를 구성하고 다른 AWS 서비스는 모니터링하지 않습니다.
- 액티브/액티브, 액티브/패시브 - 핫, 액티브/패시브 - 콜드 및 액티브/패시브 - 파일럿 라이트 접근 방식과 같은 여러 리전 복원력 설계에 대한 할당량 관리를 고려하지 않습니다.
- 리전 간 할당량 차이를 평가하지 않습니다.
- 특정 할당량 증가 요청에 대해 모든 리전의 요구 사항을 평가하지 않습니다.
- 다중 리전 할당량 관리를 위한 템플릿을 활용하지 않습니다.

이 모범 사례 확립의 이점: AWS Service Quotas를 자동으로 추적하고 이러한 할당량을 기준으로 사용량을 모니터링하면 할당량 한도에 근접할 경우 이를 알 수 있습니다. 이 모니터링 데이터를 사용하여 할당량 소진으로 인한 성능 저하를 제한할 수도 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

지원되는 서비스의 경우 경고 또는 알람을 평가하고 보낼 수 있는 다양한 서비스를 구성하여 할당량을 모니터링할 수 있습니다. 사용량을 모니터링하는 데 도움이 될 수 있으며 할당량에 근접하면 알림을 받을 수 있습니다. 이러한 경보는 AWS Config, Lambda 함수, Amazon CloudWatch 또는 AWS Trusted Advisor에서 트리거될 수 있습니다. CloudWatch Logs의 지표 필터를 사용하여 로그의 패턴을 검색하고 추출하여 사용량이 할당량 임계값에 근접하는지 여부를 확인할 수도 있습니다.

구현 단계

모니터링:

- 현재 리소스 사용 내역(버킷, 인스턴스)을 파악합니다. Amazon EC2 `DescribeInstances` API와 같은 서비스 API 작업을 사용하여 현재 리소스 사용량 내역을 수집합니다.
- 다음을 사용하여 서비스에 필수적이며 적용 가능한 현재 할당량을 캡처합니다.
  - AWS Service Quotas
  - AWS Trusted Advisor
  - AWS 설명서
  - AWS 서비스별 페이지
  - AWS Command Line Interface(AWS CLI)
  - AWS Cloud Development Kit (AWS CDK)
- 250개 이상의 AWS 서비스에 대한 할당량을 단일 위치에서 관리할 수 있는 AWS 서비스인 AWS Service Quotas를 사용합니다.
- Trusted Advisor 서비스 한도를 사용하여 다양한 임계값에서 현재 서비스 한도를 모니터링합니다.
- 서비스 할당량 기록(콘솔 또는 AWS CLI)을 사용하여 리전별 증가를 확인합니다.
- 필요한 경우 각 리전 및 각 계정의 서비스 할당량 변경 사항을 비교하여 동등성을 생성합니다.

관리:

- 자동: AWS Config 사용자 지정 규칙을 설정하여 리전 간 서비스 할당량을 스캔하고 차이점을 비교합니다.
- 자동: 예약된 Lambda 함수를 설정하여 리전 간 서비스 할당량을 스캔하고 차이점을 비교합니다.
- 수동: AWS CLI, API 또는 AWS 콘솔을 통해 서비스 할당량을 스캔하여 리전 간 서비스 할당량을 스캔하고 차이점을 비교합니다. 차이점을 보고합니다.
- 리전 간에 할당량 차이가 확인되면 필요한 경우 할당량 변경을 요청합니다.
- 모든 요청 결과를 검토합니다.

리소스

관련 모범 사례:

- REL01-BP01 Service Quotas 및 제약 조건 인식

- REL01-BP02 계정 및 리전 전체에서 서비스 할당량 관리
- REL01-BP03 아키텍처를 통해 고정된 서비스 할당량 및 제약 조건 수용
- REL01-BP05 할당량 관리 자동화
- REL01-BP06 현재의 할당량과 최대 사용량 간에 장애 조치를 수용할 만큼 여유가 충분히 있는지 확인
- REL03-BP01 워크로드를 세그먼트화하는 방법 선택
- REL10-BP01 워크로드를 여러 위치에 배포
- REL11-BP01 워크로드의 모든 구성 요소를 모니터링하여 장애 감지
- REL11-BP03 모든 계층에서 복구 자동화
- REL12-BP05 카오스 엔지니어링을 이용한 복원력 테스트

관련 문서:

- AWS Well-Architected Framework의 신뢰성 원칙: 가용성
- AWS Service Quotas(이전 명칭: 서비스 한도)
- AWS Trusted Advisor 모범 사례 점검 항목('서비스 한도' 섹션 참조)
- AWS Answers의 AWS Limit Monitor
- Amazon EC2 서비스 한도
- Service Quotas란 무엇인가요?
- 할당량 증가 요청 방법
- 서비스 엔드포인트 및 할당량
- Service Quotas 사용 설명서
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries(AWS 장애 격리 경계)
- Availability with redundancy(중복성이 적용된 가용성)
- AWS for Data
- 지속적 통합이란 무엇인가요?
- 지속적 전달이란 무엇인가요?
- APN 파트너: 구성 관리를 지원할 수 있는 파트너
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS(AWS의 테넌트당 계정 SaaS 환경에서 계정 수명 주기 관리)

- 워크로드에서 API 제한 관리 및 모니터링
- View AWS Trusted Advisor recommendations at scale with AWS Organizations(AWS Organizations를 사용하여 대규모로 AWS Trusted Advisor 권장 사항 보기)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower(AWS Control Tower를 사용하여 서비스 한도 증가 및 엔터프라이즈 지원 자동화)
- Service Quotas에 사용되는 작업, 리소스 및 조건 키

관련 동영상:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas(Service Quotas를 사용하여 AWS 서비스에 대한 할당량 보기 및 관리)
- AWS IAM 할당량 데모
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small(루프 닫기 및 마음 열기: 규모에 상관없이 시스템을 제어하는 방법)

관련 도구:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP05 할당량 관리 자동화

임계값에 근접했을 때 알림을 받을 수 있는 도구를 구현합니다. AWS Service Quotas API를 사용하여 할당량 증가 요청을 자동화할 수 있습니다.

CMDB(구성 관리 데이터베이스) 또는 티켓팅 시스템을 Service Quotas와 통합하면 할당량 증가 요청 및 현재 할당량 추적을 자동화할 수 있습니다. Service Quotas는 AWS SDK 외에도 AWS Command Line Interface(AWS CLI)를 사용한 자동화를 제공합니다.

일반적인 안티 패턴:

- 스프레드시트에서 할당량 및 사용량 추적
- 일별, 주별 또는 월별 사용량에 대한 보고서를 실행한 다음 사용량을 할당량과 비교

이 모범 사례 정립의 이점: AWS 서비스 할당량을 자동으로 추적하고 해당 할당량을 기준으로 사용량을 모니터링하면 할당량에 근접할 경우 이를 알 수 있습니다. 자동화를 설정하여 필요할 때 할당량 증가를 요청할 수 있습니다. 사용량이 반대로 감소 추세를 보일 경우 할당량을 일부 낮추면 위험 감소(자격 증명이 침해당한 경우) 및 비용 절감의 이점을 실현할 수 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 자동화된 모니터링 설정 SDK를 사용하여 임계값에 근접할 때 알림을 제공하는 도구를 구현합니다.
  - Service Quotas를 사용하고 AWS Limit Monitor 또는 AWS Marketplace에서 제공하는 것과 같은 자동 할당량 모니터링 솔루션으로 이 서비스를 보강합니다.
    - Service Quotas란 무엇입니까?
    - AWS에서의 할당량 모니터링 - AWS 솔루션
  - Amazon SNS 및 AWS Service Quotas API를 사용하여 할당량 임계값을 기준으로 트리거되는 응답을 설정합니다.
  - 테스트를 자동화합니다.
    - 한도 임계값을 구성합니다.
    - AWS Config, 배포 파이프라인, Amazon EventBridge 또는 타사의 변경 이벤트와 통합합니다.
    - 응답을 테스트하기 위해 인위적으로 할당량 임계값을 설정합니다.
    - 알림 발생 시에 적절한 조치를 취하고 필요한 경우 AWS Support에 연락하도록 트리거를 설정합니다.
    - 변경 이벤트를 수동으로 트리거합니다.
    - 게임 데이를 실행하여 할당량 증가 변경 프로세스를 테스트합니다.

리소스

관련 문서:

- APN 파트너: 구성 관리를 지원할 수 있는 파트너
- AWS Marketplace: 한도 추적에 유용한 구성 관리 데이터베이스(CMDB) 제품
- AWS Service Quotas(이전의 '서비스 한도')
- AWS Trusted Advisor 모범 사례 점검 항목('서비스 한도' 섹션 참조)
- AWS에서의 할당량 모니터링 - AWS 솔루션
- Amazon EC2 서비스 한도
- Service Quotas란 무엇입니까?

관련 동영상:

- AWS Live re:Inforce 2019 - Service Quotas

REL01-BP06 현재의 할당량과 최대 사용량 간에 장애 조치를 수용할 만큼 여유가 충분히 있는지 확인

리소스는 실패하거나 액세스할 수 없는 경우에도 해당 리소스가 성공적으로 종료될 때까지 계속 할당량 계산에 포함될 수 있습니다. 장애가 있거나 액세스할 수 없는 리소스와 대체 리소스가 중복되는 부분이 할당량에 반영되는지 확인합니다. 이 차이를 계산할 때 네트워크 장애, 가용 영역 장애 또는 리전 장애와 같은 사용 사례를 고려해야 합니다.

원하는 결과: 리소스 또는 리소스 액세스 가능성의 작거나 큰 장애는 현재 서비스 임계값 내에서 처리될 수 있습니다. 영역 장애, 네트워크 장애 또는 리전 장애도 리소스 계획에서 고려되었습니다.

일반적인 안티 패턴:

- 장애 조치 시나리오를 고려하지 않고 현재의 수요를 기준으로 서비스 할당량을 설정합니다.
- 서비스의 최대 할당량을 계산할 때 정적 안정성 원칙을 고려하지 않습니다.
- 각 리전에 필요한 총 할당량을 계산할 때 액세스할 수 없는 리소스의 가능성을 고려하지 않습니다.
- 일부 서비스에 대한 AWS 서비스 장애 격리 경계 및 잠재적인 비정상적인 사용 패턴을 고려하지 않습니다.

이 모범 사례 확립의 이점: 서비스 중단 이벤트가 애플리케이션 가용성에 영향을 미치는 경우 클라우드를 통해 이러한 이벤트를 완화하거나 복구하는 전략을 구현할 수 있습니다. 그러한 전략에는 장애가 발

생하거나 액세스할 수 없는 리소스를 대체할 추가 리소스를 생성하는 작업이 포함되는 경우가 많습니다. 할당량 전략은 이러한 장애 조치 조건을 수용하고 서비스 한도 소진으로 인한 추가 저하를 계층화하지 않습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

할당량 한도를 평가할 때 일부 성능 저하로 인해 발생할 수 있는 장애 조치 사례를 고려합니다. 다음 유형의 장애 조치 사례를 고려해야 합니다.

- 중단되었거나 액세스할 수 없는 VPC
- 액세스할 수 없는 서브넷
- 가용 영역이 많은 리소스의 액세스 가능성에 영향을 줄 만큼 충분히 저하되었습니다.
- 다양한 네트워킹 경로 또는 수신 및 송신 지점이 차단되거나 변경됩니다.
- 많은 리소스의 액세스 가능성에 영향을 줄 정도로 리전의 성능이 충분히 저하되었습니다.
- 여러 리소스가 있지만 모든 리소스가 리전 또는 가용 영역의 장애로 인해 영향을 받는 것은 아닙니다.

위 목록과 같은 실패는 장애 조치 이벤트를 시작하는 트리거가 될 수 있습니다. 장애 조치를 하기로 내린 결정은 비즈니스에 미치는 영향이 크게 다를 수 있으므로 상황과 고객마다 다릅니다. 그러나 애플리케이션 또는 서비스를 장애 조치하기로 운영상 결정할 때 장애 조치 위치에 있는 리소스의 용량 계획 및 관련 할당량은 이벤트 시작 전에 해결되어야 합니다.

발생할 수 있는 정상 사용량보다 높은 사용량을 고려하여 각 서비스에 대한 서비스 할당량을 검토합니다. 이러한 사용량은 네트워킹 또는 권한으로 인해 도달할 수 있지만 여전히 액티브 상태인 리소스와 관련이 있을 수 있습니다. 종료되지 않은 액티브 리소스는 여전히 서비스 할당량 제한에 포함됩니다.

구현 단계

- 서비스 할당량과 최대 사용량 간에 장애 조치와 접근성 손실을 수용할 만큼 여유가 충분히 있는지 확인합니다.
- 배치 패턴, 가용성 요청 사항, 서비스 사용량 증가를 고려해 서비스 할당량을 결정합니다.
- 필요한 경우 할당량 증가를 요청합니다. 할당량 증가 요청이 반영될 시간을 고려합니다.
- 신뢰성 요구 사항("9의 개수"라고도 함)을 확인합니다.
- 장애 시나리오(예: 구성 요소, 가용 영역 또는 리전 손실)를 설정합니다.

- 배포 방법(Canary, 블루/그린, 레드/블랙 또는 롤링 등)을 설정합니다.
- 현재 한도에 적절한 버퍼(예: 15%)가 포함되어야 합니다.
- 적절한 경우 정적 안정성(영역 및 리전)에 대한 계산을 포함합니다.
- 사용량 증가 계획(사용 추세 모니터링 등)을 수립합니다.
- 가장 중요한 워크로드에 대한 정적 안정성의 영향을 고려합니다. 모든 리전 및 가용 영역에서 정적으로 안정적인 시스템을 준수하는 리소스를 평가합니다.
- 온디맨드 용량 예약을 사용하여 장애 조치 전에 용량을 예약하는 것을 고려합니다. 이는 장애 조치 중에 적절한 양과 유형의 리소스를 확보하여 가장 중요한 비즈니스 일정 중에 잠재적 위험을 줄일 수 있는 유용한 전략이 될 수 있습니다.

리소스

관련 모범 사례:

- REL01-BP01 Service Quotas 및 제약 조건 인식
- REL01-BP02 계정 및 리전 전체에서 서비스 할당량 관리
- REL01-BP03 아키텍처를 통해 고정된 서비스 할당량 및 제약 조건 수용
- REL01-BP04 할당량 모니터링 및 관리
- REL01-BP05 할당량 관리 자동화
- REL03-BP01 워크로드를 세그먼트화하는 방법 선택
- REL10-BP01 워크로드를 여러 위치에 배포
- REL11-BP01 워크로드의 모든 구성 요소를 모니터링하여 장애 감지
- REL11-BP03 모든 계층에서 복구 자동화
- REL12-BP05 카오스 엔지니어링을 이용한 복원력 테스트

관련 문서:

- AWS Well-Architected Framework의 신뢰성 원칙: 가용성
- AWS Service Quotas(이전 명칭: 서비스 한도)
- AWS Trusted Advisor 모범 사례 점검 항목('서비스 한도' 섹션 참조)
- AWS Answers의 AWS Limit Monitor
- Amazon EC2 서비스 한도

- Service Quotas란 무엇인가요?
- 할당량 증가 요청 방법
- 서비스 엔드포인트 및 할당량
- Service Quotas 사용 설명서
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries(AWS 장애 격리 경계)
- Availability with redundancy(중복성이 적용된 가용성)
- AWS for Data
- 지속적 통합이란 무엇인가요?
- 지속적 전달이란 무엇인가요?
- APN 파트너: 구성 관리를 지원할 수 있는 파트너
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS(AWS의 테넌트당 계정 SaaS 환경에서 계정 수명 주기 관리)
- 워크로드에서 API 제한 관리 및 모니터링
- View AWS Trusted Advisor recommendations at scale with AWS Organizations(AWS Organizations를 사용하여 대규모로 AWS Trusted Advisor 권장 사항 보기)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower(AWS Control Tower를 사용하여 서비스 한도 증가 및 엔터프라이즈 지원 자동화)
- Service Quotas에 사용되는 작업, 리소스 및 조건 키

관련 동영상:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas(Service Quotas를 사용하여 AWS 서비스에 대한 할당량 보기 및 관리)
- AWS IAM 할당량 데모
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small(루프 닫기 및 마음 열기: 규모에 상관없이 시스템을 제어하는 방법)

관련 도구:

- AWS CodeDeploy

- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL 2 네트워크 토폴로지는 어떻게 계획합니까?

워크로드는 여러 환경에 존재할 수 있습니다. 여기에는 다중 클라우드 환경(퍼블릭 액세스 및 프라이빗)과 기존 데이터 센터 인프라가 포함됩니다. 따라서 시스템 내부 및 시스템 간 연결, 퍼블릭 IP 주소 관리, 프라이빗 IP 주소 관리 및 도메인 이름 확인과 같은 네트워크 고려 사항을 계획에 포함해야 합니다.

모범 사례

- REL02-BP01 워크로드 퍼블릭 엔드포인트에 고가용성 네트워크 연결 사용
- REL02-BP02 클라우드와 온프레미스 환경의 프라이빗 네트워크 간에 이중화된 연결 프로비저닝
- REL02-BP03 확장 및 가용성을 위한 IP 서브넷 할당 계정 확인
- REL02-BP04 다대다 메시보다 허브 앤드 스포크 토폴로지 선호
- REL02-BP05 연결된 모든 프라이빗 주소 공간에서 겹치지 않는 프라이빗 IP 주소 범위 적용

REL02-BP01 워크로드 퍼블릭 엔드포인트에 고가용성 네트워크 연결 사용

워크로드의 퍼블릭 엔드포인트에 대한 고가용성 네트워크 연결을 구축하면 연결 손실로 인한 가동 중지 시간을 줄이고 워크로드의 가용성 및 SLA를 개선하는 데 도움이 될 수 있습니다. 이러한 고가용성을 달성하려면 고가용성 DNS, 콘텐츠 전송 네트워크(CDN), API Gateway, 로드 밸런싱 또는 역방향 프록시를 사용합니다.

원하는 결과: 퍼블릭 엔드포인트에 대한 고가용성 네트워크 연결을 계획, 구축 및 운영하는 것이 중요합니다. 연결 손실로 인해 워크로드에 연결할 수 없게 되면 워크로드가 실행 중이고 사용 가능한 경우에도 고객은 시스템이 다운된 것으로 보게 됩니다. 워크로드 자체에 대한 복원력 있는 아키텍처와 함께

워크로드의 퍼블릭 엔드포인트를 위한 고가용성 및 복원력 있는 네트워크 연결을 결합하여 고객에게 가능한 최상의 가용성과 서비스 수준을 제공할 수 있습니다.

AWS Global Accelerator, Amazon CloudFront, Amazon API Gateway, AWS Lambda 함수 URL, AWS AppSync API 및 Elastic Load Balancing(ELB)은 모두 고가용성 퍼블릭 엔드포인트를 제공합니다.Amazon Route 53은 퍼블릭 엔드포인트 주소를 확인할 수 있는지 확인하기 위해 도메인 이름 확인을 위한 고가용성 DNS 서비스를 제공합니다.

로드 밸런싱 및 프록싱을 위해 AWS Marketplace 소프트웨어 어플라이언스를 평가할 수도 있습니다.

일반적인 안티 패턴:

- 고가용성을 위한 DNS 및 네트워크 연결을 계획하지 않고 고가용성 워크로드를 설계합니다.
- 개별 인스턴스 또는 컨테이너에서 퍼블릭 인터넷 주소를 사용하고 DNS를 통해 이러한 주소에 대한 연결을 관리합니다.
- 서비스를 찾기 위해 도메인 이름 대신 IP 주소를 사용합니다.
- 퍼블릭 엔드포인트에 대한 연결이 끊어지는 시나리오를 테스트하지 않습니다.
- 네트워크 처리량 요구 사항 및 배포 패턴을 분석하지 않습니다.
- 워크로드의 퍼블릭 엔드포인트에 대한 인터넷 네트워크 연결이 중단될 수 있는 시나리오를 테스트하고 계획하지 않습니다.
- 콘텐츠 전송 네트워크를 사용하지 않고 대규모 지리적 영역에 콘텐츠(예: 웹 페이지, 정적 자산, 미디어 파일)를 제공합니다.
- DDoS(Distributed Denial Of Service) 공격에 대한 계획이 없습니다. DDoS 공격은 합법적인 트래픽을 차단하고 사용자의 가용성을 낮출 위험이 있습니다.

이 모범 사례 확립의 이점: 가용성이 높고 복원력이 뛰어난 네트워크 연결을 설계하면 사용자가 워크로드에 액세스하고 사용할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

퍼블릭 엔드포인트에 대한 고가용성 네트워크 연결 구축의 핵심은 트래픽 라우팅입니다. 트래픽이 엔드포인트에 도달할 수 있는지 확인하려면 DNS가 도메인 이름을 해당 IP 주소로 확인할 수 있어야 합니다. Amazon Route 53과 같이 가용성과 확장성이 뛰어난 도메인 이름 시스템(DNS)을 사용하여 도메인의 DNS 레코드를 관리합니다. Amazon Route 53에서 제공하는 상태 확인을 사용할 수도 있습니다. 상태 확인은 애플리케이션이 도달 가능하고 사용 가능하며 작동하는지 확인하며, 웹 페이지 또는 특

정 URL 요청과 같은 사용자 행동을 모방하는 방식으로 설정할 수 있습니다. 장애가 발생하면 Amazon Route 53은 DNS 확인 요청에 응답하고 트래픽을 상태 엔드포인트로만 보냅니다. Amazon Route 53에서 제공하는 지리적 DNS 및 지연 시간 기반 라우팅 기능을 사용할 수도 있습니다.

워크로드 자체의 가용성이 높은지 확인하려면 Elastic Load Balancing(ELB)을 사용합니다. Amazon Route 53은 대상 컴퓨팅 인스턴스에 트래픽을 분산하는 ELB로 트래픽을 라우팅하는 데 사용할 수 있습니다. 서버리스 솔루션을 위해 AWS Lambda와 함께 Amazon API Gateway를 사용할 수도 있습니다. 고객은 여러 AWS 리전에서 워크로드를 실행할 수도 있습니다. 다중 사이트 액티브/액티브 패턴을 사용하면 워크로드가 여러 리전의 트래픽을 처리할 수 있습니다. 다중 사이트 액티브/패시브 패턴을 사용하면 워크로드가 액티브 리전의 트래픽을 처리하는 동시에 데이터가 보조 리전으로 복제되고 기본 리전에서 장애가 발생할 경우 활성화됩니다. 그런 다음 Route 53 상태 확인을 사용하여 기본 리전의 모든 엔드포인트에서 보조 리전의 엔드포인트로의 DNS 장애 조치를 제어하여 사용자가 워크로드에 도달하고 사용할 수 있는지 확인할 수 있습니다.

Amazon CloudFront는 전 세계 엣지 로케이션 네트워크를 사용하여 요청을 처리함으로써 짧은 지연 시간과 높은 데이터 전송 속도로 콘텐츠를 배포하기 위한 간단한 API를 제공합니다. 콘텐츠 전송 네트워크(CDN)는 사용자와 가까운 위치에 있거나 캐시된 콘텐츠를 제공하여 고객에게 서비스를 제공합니다. 또한 콘텐츠에 대한 로드가 서버에서 CloudFront의 엣지 로케이션으로 이동되므로 애플리케이션의 가용성도 향상됩니다. 엣지 로케이션 및 리전 엣지 캐시는 콘텐츠의 캐시된 복사본을 사용자와 가까이에 유지하여 빠른 검색과 워크로드의 도달 가능성 및 가용성을 높입니다.

사용자가 지리적으로 분산된 워크로드의 경우 AWS Global Accelerator는 애플리케이션의 가용성과 성능을 개선하는 데 도움이 됩니다. AWS Global Accelerator는 하나 이상의 AWS 리전에서 호스팅되는 애플리케이션에 대한 고정 진입점 역할을 하는 애니캐스트 정적 IP 주소를 제공합니다. 이렇게 하면 트래픽이 가능한 한 사용자와 가까운 AWS 글로벌 네트워크로 유입되어 워크로드의 도달 가능성과 가용성이 향상됩니다. AWS Global Accelerator는 또한 TCP, HTTP 및 HTTPS 상태 확인을 사용하여 애플리케이션 엔드포인트의 상태를 모니터링합니다. 엔드포인트의 상태 또는 구성이 변경되면 사용자 트래픽이 정상 엔드포인트로 리디렉션되어 사용자에게 최상의 성능과 가용성을 제공합니다. 또한 AWS Global Accelerator에는 독립 네트워크 영역에서 서비스하는 두 개의 정적 IPv4 주소를 사용하여 애플리케이션의 가용성을 높이는 장애 격리 설계가 있습니다.

DDoS 공격으로부터 고객을 보호하기 위해 AWS는 AWS Shield Standard를 제공합니다. Shield Standard는 자동으로 활성화되며 SYN/UDP 플러드 및 반사 공격과 같은 일반적인 인프라(계층 3 및 4) 공격으로부터 보호하여 AWS에서 애플리케이션의 고가용성을 지원합니다. 더 정교하고 더 큰 공격(예: UDP 플러드)에 대한 추가 보호를 위해 상태 고갈 공격(예: TCP SYN 플러드)을 방지하고 Amazon Elastic Compute Cloud(Amazon EC2), Elastic Load Balancing(ELB), Amazon CloudFront, AWS Global Accelerator 및 Route 53에서 실행되는 애플리케이션을 보호하기 위해 AWS Shield Advanced 사용을 고려할 수 있습니다. HTTP POST 또는 GET 플러드와 같은 애플리케이션 계층 공격으로부

터 보호하려면 AWS WAF를 사용합니다. AWS WAF는 IP 주소, HTTP 헤더, HTTP 본문, URI 문자열, SQL 명령어 삽입 및 교차 사이트 스크립팅 조건을 사용하여 요청을 차단할지 또는 허용할지 결정할 수 있습니다.

구현 단계

1. 고가용성 DNS 설정: Amazon Route 53은 가용성과 확장성이 뛰어난 도메인 이름 시스템(DNS) 웹 서비스입니다. Route 53은 사용자 요청을 AWS 또는 온프레미스에서 실행되는 인터넷 애플리케이션에 연결합니다. 자세한 내용은 Amazon Route 53을 DNS 서비스로 구성을 참조하세요.
2. 상태 확인 설정: Route 53을 사용할 때 정상 대상만 확인할 수 있는지 확인합니다. Route 53 상태 확인을 생성하고 DNS 장애 조치를 구성하여 시작합니다. 상태 확인을 설정할 때 다음 측면을 고려해야 합니다.
   a. Amazon Route 53이 상태 확인이 정상인지 확인하는 방법
   b. 상태 확인 생성, 업데이트 및 삭제
   c. 상태 확인 상태 모니터링 및 알림 받기
   d. Amazon Route 53 DNS 모범 사례
3. DNS 서비스를 엔드포인트에 연결합니다.
   a. Elastic Load Balancing을 트래픽의 대상으로 사용하는 경우 로드 밸런서의 리전 엔드포인트를 가리키는 Amazon Route 53을 사용하여 별칭 레코드를 생성합니다. 별칭 레코드를 생성하는 동안 대상 상태 평가 옵션을 예로 설정합니다.
   b. API Gateway가 사용되는 서버리스 워크로드 또는 프라이빗 API의 경우 Route 53을 사용하여 트래픽을 API Gateway로 보냅니다.
4. 콘텐츠 전송 네트워크를 결정합니다.
   a. 사용자에게 더 가까운 엣지 로케이션을 사용하여 콘텐츠를 제공하려면 CloudFront가 콘텐츠를 제공하는 방법을 이해하는 것부터 시작합니다.
   b. 간단한 CloudFront 배포로 시작합니다. 그런 다음 CloudFront는 콘텐츠를 제공할 위치와 콘텐츠 제공을 추적 및 관리하는 방법에 대한 세부 정보를 알고 있습니다. CloudFront 배포를 설정할 때 다음 측면을 이해하고 고려해야 합니다.
      i. 캐싱이 CloudFront 엣지 로케이션에서 작동하는 방식
      ii. CloudFront 캐시에서 직접 제공되는 요청 비율 증가(캐시 적중률)
      iii. Amazon CloudFront Origin Shield 사용
      iv. CloudFront 오리진 장애 조치로 고가용성 최적화
5. 애플리케이션 계층 보호 설정: AWS WAF는 가용성에 영향을 미치거나 보안을 손상시키거나 과도한 리소스를 소비할 수 있는 일반적인 웹 악용 및 봇으로부터 보호하는 데 도움이 됩니다. 더 깊이 이

해하려면 AWS WAF의 작동 방식을 검토하고 애플리케이션 계층 HTTP POST 및 GET 플러드로부터 보호를 구현할 준비가 되면 AWS WAF 시작하기를 검토하세요. CloudFront와 함께 AWS WAF를 사용할 수도 있습니다. AWS WAF가 Amazon CloudFront 기능과 작동하는 방식에 대한 설명서를 참조하세요.

6. 추가 DDoS 보호 설정: 기본적으로 모든 AWS 고객은 AWS Shield Standard를 사용하여 웹 사이트 또는 애플리케이션을 대상으로 하는 일반적이고 가장 자주 발생하는 네트워크 및 전송 계층 DDoS 공격으로부터 추가 비용 없이 보호를 받습니다. Amazon EC2, Elastic Load Balancing, Amazon CloudFront, AWS Global Accelerator 및 Amazon Route 53에서 실행되는 인터넷 연결 애플리케이션을 추가로 보호하기 위해 AWS Shield Advanced를 고려하고 DDoS 복원력이 있는 아키텍처의 예를 검토할 수 있습니다. DDoS 공격으로부터 워크로드와 퍼블릭 엔드포인트를 보호하려면 AWS Shield Advanced 시작하기를 검토하세요.

리소스

관련 모범 사례:

- REL10-BP01 워크로드를 여러 위치에 배포
- REL10-BP02 다중 위치 배포에 적합한 위치 선택
- REL11-BP04 복구 중 컨트롤 플레인이 아닌 데이터 영역 사용
- REL11-BP06 이벤트가 가용성에 영향을 미치는 경우 알림 전송

관련 문서:

- APN 파트너: 네트워킹 계획을 지원할 수 있는 파트너
- AWS Marketplace for Network Infrastructure(네트워크 인프라에 대한 AWS Marketplace)
- AWS Global Accelerator란 무엇입니까?
- Amazon CloudFront란 무엇입니까?
- Amazon Route 53란 무엇입니까?
- Elastic Load Balancing란 무엇입니까?
- Network Connectivity capability - Establishing Your Cloud Foundations(네트워크 연결 기능 - 클라우드 기반 구축)
- What is Amazon API Gateway?(Amazon API Gateway란 무엇입니까?)
- What are AWS WAF, AWS Shield, and AWS Firewall Manager?(AWS WAF, AWS Shield 및 AWS Firewall Manager란 무엇입니까?)

- What is Amazon Route 53 Application Recovery Controller?(Amazon Route53 Application Recovery Controller란 무엇입니까?)
- Configure custom health checks for DNS failover(DNS 장애 조치를 위한 사용자 지정 상태 확인 구성)

관련 동영상:

- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator(AWS re:Invent 2022 - AWS Global Accelerator로 성능 및 가용성 향상)
- AWS re:Invent 2020: Global traffic management with Amazon Route 53(AWS re:Invent 2020: Amazon Route 53을 사용한 글로벌 트래픽 관리)
- AWS re:Invent 2022 - Operating highly available Multi-AZ applications(AWS re:Invent 2022 - 고가용성 다중 AZ 애플리케이션 운영)
- AWS re:Invent 2022 - Dive deep on AWS networking infrastructure(AWS re:Invent 2022 - AWS 네트워킹 인프라에 대해 자세히 알아보기)
- AWS re:Invent 2022 - Building resilient networks(AWS re:Invent 2022 - 복원력이 뛰어난 네트워크 구축)

관련 예시:

- Amazon Route 53 Application Recovery Controller(Route53 ARC)를 사용한 재해 복구
- 복원력 워크숍
- AWS Global Accelerator 워크숍

REL02-BP02 클라우드와 온프레미스 환경의 프라이빗 네트워크 간에 이중화된 연결 프로비저닝

별도로 배포된 프라이빗 네트워크 간에 여러 AWS Direct Connect 연결 또는 VPN 터널을 사용합니다. 고가용성을 위해 여러 Direct Connect 위치를 사용합니다. 여러 AWS 리전을 사용하는 경우 2개 이상의 AWS 리전에 이중화되도록 해야 합니다. VPN을 종료하는 AWS Marketplace 어플라이언스를 평가해야 할 수 있습니다. AWS Marketplace 어플라이언스를 사용하는 경우 다른 가용 영역에서 고가용성을 위해 중복 인스턴스를 배포합니다.

AWS Direct Connect는 온프레미스 환경에서 AWS로의 전용 네트워크 연결을 쉽게 설정할 수 있게 지원하는 클라우드 서비스입니다. Direct Connect Gateway를 사용하면 온프레미스 데이터 센터를 여러 AWS 리전에 분산된 다수의 AWS VPC에 연결할 수 있습니다.

이러한 중복성은 연결 복원력에 영향을 주는 잠재적 장애를 해결합니다.

- 토폴로지에서 장애가 발생하는 경우 복원할 방법
- 특정 항목을 잘못 구성하여 연결을 제거하는 경우의 결과
- 예기치 않은 트래픽 또는 서비스 사용량 증가 처리 가능 여부
- DDoS(분산 서비스 거부) 공격 시도에서 정상 상태 유지 가능 여부

VPN을 통해 온프레미스 데이터 센터에 VPC를 연결하는 경우 어플라이언스를 실행해야 하는 인스턴스 크기 및 공급업체를 선택할 때 필요한 복원력 및 대역폭 요구 사항을 고려해야 합니다. 해당 구현에서 복원되지 않는 VPN 어플라이언스를 사용하는 경우에는 두 번째 어플라이언스를 통한 중복 연결을 설정해야 합니다. 이러한 모든 시나리오에서는 허용되는 복구 시간을 정의하고 테스트를 진행하여 해당 요구 사항을 충족할 수 있는지를 확인해야 합니다.

Direct Connect 연결을 사용하여 VPC를 데이터 센터에 연결하기로 선택한 경우 이 연결이 고가용성이어야 한다면 각 데이터 센터에서 중복 Direct Connect 연결을 사용하세요. 중복 연결에는 첫 번째 연결과 다른 위치의 두 번째 Direct Connect 연결이 사용되어야 합니다. 데이터 센터가 다수인 경우 연결이 각기 다른 위치에서 종료되는지 확인합니다. 연결은 Direct Connect 복원 도구 키트 를 사용하여 설정할 수 있습니다.

AWS VPN을 사용하여 인터넷을 통해 VPN으로 장애 조치하려는 경우, VPN 터널당 최대 1.25Gbps의 처리량이 지원되기는 하지만 여러 AWS 관리형 VPN 터널이 같은 VGW에서 종료되는 경우에는 아웃바운드 트래픽용 ECMP(Equal Cost Multi Path)가 지원되지 않는다는 점을 알아야 합니다. 장애 조치 중 1Gbps 미만의 속도가 허용되는 경우가 아니라면 AWS 관리형 VPN을 Direct Connect 연결의 백업으로 사용하지 않는 것이 좋습니다.

또한 VPC 엔드포인트를 사용하여 퍼블릭 인터넷을 통하지 않고 비공개로 VPC를 지원되는 AWS 서비스와 AWS PrivateLink 기반 VPC 엔드포인트 서비스에 연결할 수 있습니다. 엔드포인트는 가상 디바이스입니다. 수평적으로 확장되고 이중화된 고가용성의 VPC 구성 요소입니다. 엔드포인트를 사용하면 네트워크 트래픽에 미치는 가용성 위험이나 대역폭 제약 없이 VPC의 인스턴스와 서비스 간에 통신할 수 있습니다.

일반적인 안티 패턴:

- 온사이트 네트워크와 AWS 간에 연결 공급자를 하나만 구성
- AWS Direct Connect 연결의 연결 기능을 사용하지만 연결을 하나만 구성
- VPN 연결을 위한 경로를 하나만 구성

이 모범 사례 수립의 이점: 클라우드 환경과 기업 또는 온프레미스 환경 간에 이중화된 연결을 구현하면 두 환경 간의 종속 서비스가 서로 안정적으로 통신할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- AWS와 온프레미스 환경 간 고가용성 연결을 구현해야 합니다. 별도로 배포된 프라이빗 네트워크 간에 여러 AWS Direct Connect 연결 또는 VPN 터널을 사용합니다. 고가용성을 위해 여러 Direct Connect 위치를 사용합니다. 여러 AWS 리전을 사용하는 경우 2개 이상의 AWS 리전에 이중화되도록 해야 합니다. VPN을 종료하는 AWS Marketplace 어플라이언스를 평가해야 할 수 있습니다. AWS Marketplace 어플라이언스를 사용하는 경우 다른 가용 영역에서 고가용성을 위해 중복 인스턴스를 배포합니다.
  - 온프레미스 환경에 중복 연결된 상태인지 확인합니다. 가용성 요구 사항을 충족하기 위해서는 여러 AWS 리전에 중복 연결이 필요할 수 있습니다.
    - AWS Direct Connect 복원력 권장 사항
    - 이중화 Site-to-Site VPN 연결을 사용하여 장애 조치 제공
      - 서비스 API 작업을 사용하여 Direct Connect 회선이 올바르게 사용되고 있는지 확인합니다.
        - DescribeConnections
        - DescribeConnectionsOnInterconnect
        - DescribeDirectConnectGatewayAssociations
        - DescribeDirectConnectGatewayAttachments
        - DescribeDirectConnectGateways
        - DescribeHostedConnections
        - DescribeInterconnects
      - Direct Connect 연결이 하나만 있거나 하나도 없는 경우 가상 프라이빗 게이트웨이에 이중화 VPN 터널을 설정합니다.
        - AWS Site-to-Site VPN이란 무엇입니까?
  - 현재 연결(예: Direct Connect, 가상 프라이빗 게이트웨이, AWS Marketplace 어플라이언스)을 파악합니다.
    - 서비스 API 작업을 사용하여 Direct Connect 연결의 구성을 쿼리합니다.
      - DescribeConnections
      - DescribeConnectionsOnInterconnect
      - DescribeDirectConnectGatewayAssociations

    - DescribeDirectConnectGatewayAttachments
    - DescribeDirectConnectGateways
    - DescribeHostedConnections
    - DescribeInterconnects
- 서비스 API 작업을 사용하여 라우팅 테이블에서 사용하는 가상 프라이빗 게이트웨이를 수집합니다.
    - DescribeVpnGateways
    - DescribeRouteTables
- 서비스 API 작업을 사용하여 라우팅 테이블에서 사용하는 AWS Marketplace 애플리케이션을 수집합니다.
    - DescribeRouteTables

리소스

관련 문서:

- APN 파트너: 네트워킹 계획을 지원할 수 있는 파트너
- AWS Direct Connect 복원력 권장 사항
- 네트워크 인프라에 대한 AWS Marketplace
- Amazon Virtual Private Cloud 연결 옵션 백서
- 여러 데이터 센터 고가용 네트워크 연결
- 이중화 Site-to-Site VPN 연결을 사용하여 장애 조치 제공
- Using the Direct Connect Resiliency Toolkit to get started(Direct Connect 복원 도구 키트를 사용하여 시작)
- VPC 엔드포인트 및 VPC 엔드포인트 서비스(AWS PrivateLink)
- Amazon VPC란 무엇입니까?
- Transit Gateway란 무엇일까요?
- AWS Site-to-Site VPN이란 무엇입니까?
- Working with Direct Connect Gateways(Direct Connect 게이트웨이 작업)

관련 동영상:

- AWS re:Invent 2018: Amazon VPC의 고급 VPC 설계 및 새로운 기능(NET303)

- AWS re:Invent 2019: 여러 VPC를 위한 AWS Transit Gateway 참조 아키텍처(NET406-R1)

REL02-BP03 확장 및 가용성을 위한 IP 서브넷 할당 계정 확인

Amazon VPC IP 주소 범위는 가용 영역의 서브넷에 IP 주소를 할당하고 추후 확장을 고려하는 등 워크로드의 요구 사항을 수용할 수 있도록 충분히 커야 합니다. 여기에는 로드 밸런서, EC2 인스턴스 및 컨테이너 기반 애플리케이션이 포함됩니다.

네트워크 토폴로지를 계획할 때는 첫 단계로 IP 주소 공간 자체를 정의합니다. 각 VPC에는 RFC 1918 지침에 따라 프라이빗 IP 주소 범위를 할당해야 합니다. 이 프로세스의 일부로 다음 요구 사항을 준수하십시오.

- 리전당 두 개 이상의 VPC에 대한 IP 주소 공간을 허용합니다.
- VPC 내에서 여러 가용 영역에 걸쳐 있는 여러 서브넷에 공간을 허용합니다.
- 사용되지 않은 CIDR 블록 공간은 향후 확장을 위해 항상 VPC 내에 남겨둡니다.
- 기계 학습용 스팟 플릿, Amazon EMR 클러스터 또는 Amazon Redshift 클러스터 등 사용할 수 있는 임시 EC2 인스턴스 플릿의 요구 사항을 충족할 IP 주소 공간이 있는지 확인합니다.
- 참고로 각 서브넷 CIDR 블록에서 처음 4개의 IP 주소와 마지막 IP 주소는 예약되므로 사용할 수 없습니다.
- 대규모 VPC CIDR 블록 배포를 계획해야 합니다. VPC에 할당된 초기 VPC CIDR 블록은 변경 또는 삭제가 불가능하지만 중첩되지 않은 추가 CIDR 블록을 VPC에 추가할 수 있습니다. 서브넷 IPv4 CIDR은 변경할 수 없지만 IPv6 CIDR은 변경할 수 있습니다. 가능한 최대 규모(/16)의 VPC를 배포하면 IP 주소 수가 65,000개를 초과하게 됩니다. 기본 10.x.x.x IP 주소 공간에만 255개의 VPC를 프로비저닝할 수 있습니다. 따라서 규모를 너무 작게 하는 것보다 지나치다 싶을 만큼 큰 규모로 배포해야 VPC를 더 쉽게 관리할 수 있습니다.

일반적인 안티 패턴:

- 크기가 작은 VPC 생성
- 작은 서브넷을 생성한 다음, 확장 시에 서브넷을 구성에 추가
- Elastic Load Balancer가 사용할 수 있는 IP 주소 수를 잘못 추정
- 트래픽이 많은 여러 로드 밸런서를 동일한 서브넷에 배포

이 모범 사례 정립의 이점: 이렇게 하면 워크로드의 증가를 수용하고 확장 시 가용성을 계속 제공할 수 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 성장, 규정 준수 및 다른 제품과 통합을 수용할 수 있도록 네트워크 계획을 수립합니다. 성장은 과소평가될 수 있고 규정 준수는 변경될 수 있으며 적절한 계획 없이는 프라이빗 네트워크 연결을 구현하기가 어려울 수 있습니다.
  - 서비스 요구 사항, 지연 시간, 규정, DR(재해 복구) 요구 사항을 기준으로 관련 AWS 계정 및 리전을 선택합니다.
  - 지역 VPC 배포에 대한 요구 사항을 파악합니다.
  - VPC 규모를 파악합니다.
    - 다중 VPC 연결을 배포할 것인지 여부를 결정합니다.
      - Transit Gateway란 무엇일까요?
      - 단일 리전 다중 VPC 연결
    - 규정 요구 사항에 따라 분리된 네트워킹이 필요한지 결정합니다.
    - VPC를 최대한 큰 규모로 만듭니다. VPC에 할당된 초기 VPC CIDR 블록은 변경 또는 삭제가 불가능하지만 중첩되지 않은 추가 CIDR 블록을 VPC에 추가할 수 있습니다. 그러나 이 경우 주소 범위를 분할될 수 있습니다.
    - VPC를 최대한 큰 규모로 만듭니다. VPC에 할당된 초기 VPC CIDR 블록은 변경 또는 삭제가 불가능하지만 중첩되지 않은 추가 CIDR 블록을 VPC에 추가할 수 있습니다. 그러나 이 경우 주소 범위를 분할될 수 있습니다.

리소스

관련 문서:

- APN 파트너: 네트워킹 계획을 지원할 수 있는 파트너
- 네트워크 인프라에 대한 AWS Marketplace
- Amazon Virtual Private Cloud 연결 옵션 백서
- 여러 데이터 센터 고가용 네트워크 연결
- 단일 리전 다중 VPC 연결
- Amazon VPC란 무엇입니까?

관련 동영상:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC(Amazon VPC에 대한 VPC 설계 및 새로운 기능)(NET303)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs(여러 VPC를 위한 AWS Transit Gateway 참조 아키텍처)(NET406-R1)

REL02-BP04 다대다 메시보다 허브 앤드 스포크 토폴로지 선호

셋 이상의 네트워크 주소 공간(예: VPC와 온프레미스 네트워크)이 VPC 피어링, AWS Direct Connect 또는 VPN을 통해 연결되는 경우 AWS Transit Gateway가 제공하는 것과 같은 허브 앤드 스포크 모델을 사용합니다.

이러한 네트워크가 두 개 뿐인 경우에는 서로 연결하면 되지만 네트워크 수가 증가하면 이 메쉬 기반 연결의 복잡성이 크게 증가합니다. AWS Transit Gateway는 유지 관리가 간편한 허브 앤드 스포크 모델을 제공하므로 여러 네트워크에 걸쳐 트래픽을 라우팅할 수 있습니다.

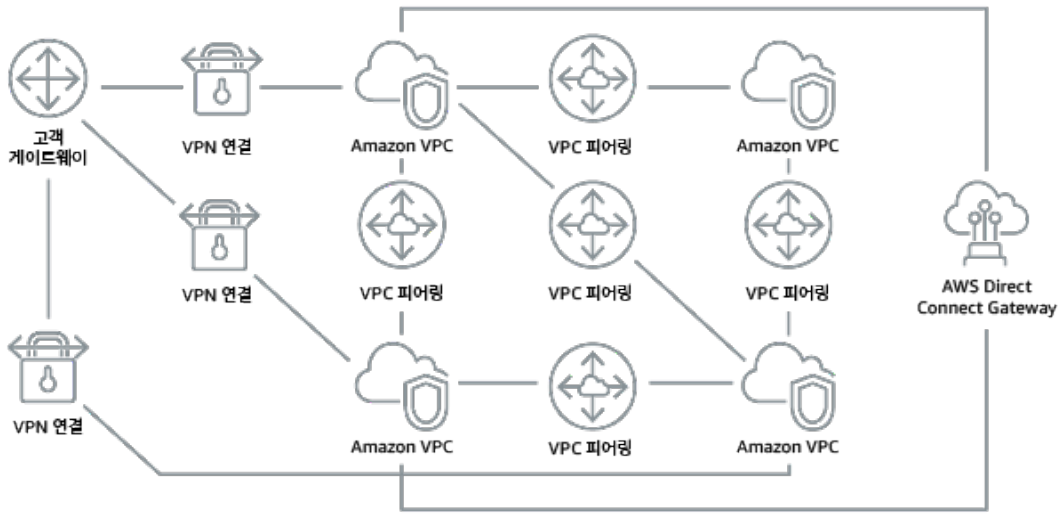


그림 1: AWS Transit Gateway가 없는 경우: VPN 연결을 사용하여 각 Amazon VPC를 서로 피어링하고 각 온사이트 위치에 피어링해야 합니다. 이 경우 확장에 따라 복잡성이 증가할 수 있습니다.

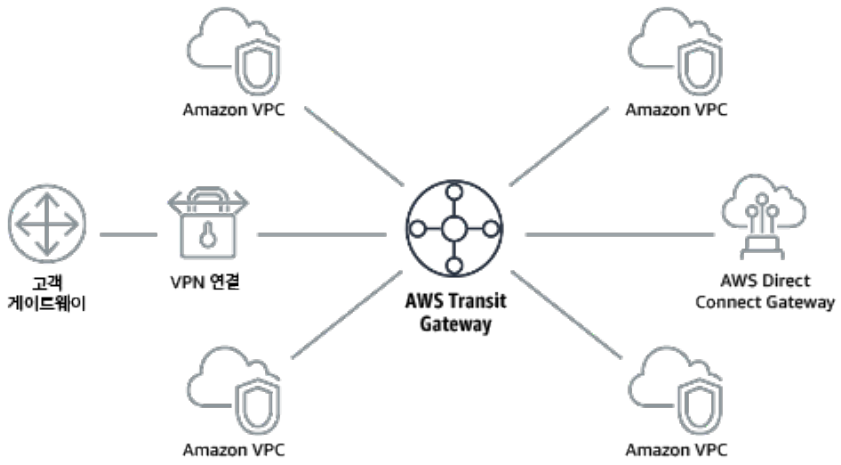


그림 2: AWS Transit Gateway를 사용하는 경우: 각 Amazon VPC 또는 VPN을 AWS Transit Gateway에 연결하기만 하면 각 VPC 또는 VPN 간에 트래픽이 라우팅됩니다.

일반적인 안티 패턴:

- VPC 피어링을 사용하여 2개 이상의 VPC 연결
- 각 VPC마다 여러 BGP 세션을 설정하여 여러 AWS 리전에 분산된 Virtual Private Cloud(VPC)에 걸쳐 있는 연결 설정

이 모범 사례 정립의 이점: 네트워크 수가 증가하면 이러한 메시 기반 연결의 복잡성이 크게 증가합니다. AWS Transit Gateway는 유지 관리가 간편한 허브 앤드 스포크 모델을 제공하므로 여러 네트워크에 트래픽을 라우팅할 수 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 다대다 메시보다 허브 앤드 스포크 토폴로지를 선호합니다. 셋 이상의 네트워크 주소 공간(VPC, 온프레미스 네트워크)이 VPC 피어링, AWS Direct Connect 또는 VPN을 통해 연결되는 경우 AWS Transit Gateway가 제공하는 것과 같은 허브 앤드 스포크 모델을 사용합니다.
  - 이러한 네트워크가 두 개뿐인 경우에는 서로 연결하면 되지만 네트워크 수가 증가하면 이 메시 기반 연결의 복잡성이 크게 증가합니다. AWS Transit Gateway는 유지 관리가 간편한 허브 앤드 스포크 모델을 제공하므로 여러 네트워크에 걸쳐 트래픽을 라우팅할 수 있습니다.
    - Transit Gateway란 무엇일까요?

리소스

관련 문서:

- APN 파트너: 네트워킹 계획을 지원할 수 있는 파트너
- 네트워크 인프라에 대한 AWS Marketplace
- 여러 데이터 센터 고가용 네트워크 연결
- VPC 엔드포인트 및 VPC 엔드포인트 서비스(AWS PrivateLink)
- Amazon VPC란 무엇입니까?
- Transit Gateway란 무엇일까요?

관련 동영상:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC(Amazon VPC에 대한 VPC 설계 및 새로운 기능)(NET303)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs(여러 VPC를 위한 AWS Transit Gateway 참조 아키텍처)(NET406-R1)

REL02-BP05 연결된 모든 프라이빗 주소 공간에서 겹치지 않는 프라이빗 IP 주소 범위 적용

VPN을 통해 피어링되거나 연결된 경우 각 VPC의 IP 주소 범위가 겹치지 않아야 합니다. 마찬가지로, VPC와 온프레미스 환경 간의 IP 주소 충돌 또는 사용하는 다른 클라우드 공급자와의 IP 주소 충돌을 방지해야 합니다. 필요한 경우 프라이빗 IP 주소 범위를 할당할 수 있어야 합니다.

IPAM(IP 주소 관리) 시스템을 사용하는 것이 도움이 될 수 있습니다. AWS Marketplace에서 여러 IPAM을 사용할 수 있습니다.

일반적인 안티 패턴:

- VPC에서 온프레미스 또는 회사 네트워크와 동일한 IP 범위 사용
- 워크로드를 배포하는 데 사용되는 VPC의 IP 범위를 추적하지 않음

이 모범 사례 수립의 이점: 네트워크를 능동적으로 계획하면 상호 연결된 네트워크에서 동일한 IP 주소가 여러 번 사용되는 것을 방지할 수 있습니다. 이렇게 하면 다른 애플리케이션을 사용하는 워크로드의 일부에서 라우팅 문제가 발생하는 것을 방지할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- CIDR 사용을 모니터링 및 관리합니다. AWS에서 잠재적인 사용량을 평가하고, 기존 VPC에 CIDR 범위를 추가하고, VPC를 생성하여 사용량을 계획적으로 늘릴 수 있습니다.
  - 현재 CIDR 사용량을 파악합니다(예: VPC, 서브넷).
    - 서비스 API 작업을 사용하여 현재 CIDR 사용량을 파악합니다.
  - 현재 서브넷 사용량을 파악합니다.
    - 서비스 API 작업을 사용하여 각 리전의 VPC당 서브넷 정보를 수집합니다.
      - DescribeSubnets
    - 현재 사용량을 기록합니다.
    - 중첩되지 않는 IP 범위를 생성했는지 판단합니다.
    - 여유 용량을 계산합니다.
    - 중첩된 IP 범위를 파악합니다. 중첩되는 범위를 연결해야 하는 경우 새 주소 범위로 마이그레이션하거나 AWS Marketplace에서 제공하는 Network and Port Translation(NAT) 어플라이언스를 사용할 수 있습니다.

리소스

관련 문서:

- APN 파트너: 네트워킹 계획을 지원할 수 있는 파트너
- 네트워크 인프라에 대한 AWS Marketplace
- Amazon Virtual Private Cloud 연결 옵션 백서
- 여러 데이터 센터 고가용 네트워크 연결
- Amazon VPC란 무엇입니까?
- IPAM이란?

관련 동영상:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC(Amazon VPC에 대한 고급 VPC 설계 및 새로운 기능)(NET303)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs(여러 VPC를 위한 AWS Transit Gateway 참조 아키텍처)(NET406-R1)

워크로드 아키텍처

질문

- REL 3 워크로드 서비스 아키텍처는 어떻게 설계합니까?
- REL 4 분산 시스템에서 장애 방지를 위한 상호 작용은 어떻게 설계합니까?
- REL 5 분산 시스템에서 장애 완화 또는 극복을 위한 상호 작용은 어떻게 설계합니까?

REL 3 워크로드 서비스 아키텍처는 어떻게 설계합니까?

SOA(서비스 지향 아키텍처) 또는 마이크로서비스 아키텍처를 사용하여 확장성과 안정성이 뛰어난 워크로드를 구축합니다. SOA(서비스 지향 아키텍처)는 서비스 인터페이스를 통해 소프트웨어 구성 요소를 재사용 가능하게 만드는 방식입니다. 마이크로서비스 아키텍처는 구성 요소를 더 작고 간단하게 만듭니다.

모범 사례

- REL03-BP01 워크로드를 세그먼트화하는 방법 선택
- REL03-BP02 특정 비즈니스 도메인 및 기능을 중심으로 서비스 구축
- REL03-BP03 API별로 서비스 계약 제공

REL03-BP01 워크로드를 세그먼트화하는 방법 선택

워크로드 세그먼트화는 애플리케이션의 복원력 요구 사항을 결정할 때 중요합니다. 되도록 모놀리식 아키텍처는 피해야 합니다. 대신 어떤 애플리케이션 구성 요소를 마이크로 서비스로 나눌 수 있을지 신중하게 고려합니다. 가능한 경우 애플리케이션 요구 사항에 따라 서비스 지향 아키텍처(SOA)와 마이크로 서비스의 결합으로 마무리될 수도 있습니다. 상태 비저장일 수 있는 워크로드는 마이크로 서비스로 배포할 수 있습니다.

원하는 결과: 워크로드는 지원 가능하고 확장 가능하며 가능한 한 느슨하게 결합되어 있어야 합니다.

워크로드를 세그먼트화하는 방법을 선택할 때 복잡성 대비 이점의 균형을 고려합니다. 신제품의 첫 출시에 필요한 것과 처음부터 워크로드를 확장할 때 필요한 것은 다릅니다. 기존 모놀리식을 리팩터링할 때 애플리케이션이 상태 비저장을 향한 해체를 얼마나 잘 지원하는지 고려해야 합니다. 서비스를 더 작은 부분으로 나누면 잘 정의된 소규모 팀에서 이러한 부분을 개발하고 관리할 수 있습니다. 그러나 크기가 작은 서비스는 복잡성을 불러올 수 있고 여기에는 지연 시간 증가, 디버깅 복잡성, 운영 부담 증가가 포함됩니다.

일반적인 안티 패턴:

- 마이크로서비스 Death Star 는 원자성 구성 요소의 상호 의존성이 커져 한 구성 요소의 장애가 훨씬 더 큰 장애로 이어져 구성 요소가 모놀리식처럼 경직되고 취약해질 수 있습니다.

이 모범 사례 정립의 이점:

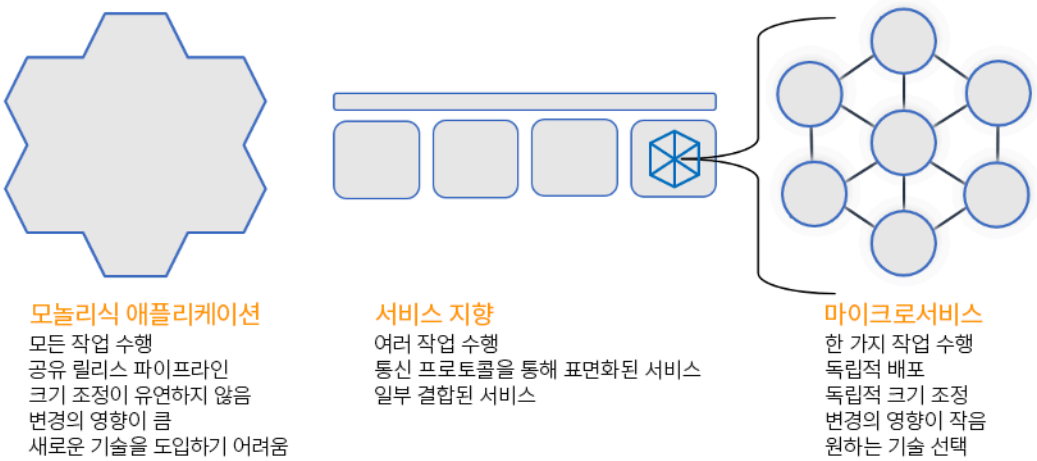
- 보다 구체적인 세그먼트를 사용하면 민첩성, 조직의 유연성 및 확장성이 향상됩니다.
- 서비스 중단의 영향이 감소합니다.
- 애플리케이션 구성 요소가 원자성이 더 큰 세그먼트화로 지원할 수 있는 여러 가능성 요구 사항을 가질 수 있습니다.
- 워크로드를 지원하는 팀의 책임이 잘 정의됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

워크로드를 분할하는 방법에 따라 아키텍처 유형 선택 SOA 또는 마이크로서비스 아키텍처(또는 드문 경우 모놀리식 아키텍처)를 선택합니다. 처음에 모놀리식 아키텍처를 선택하더라도, 사용자 채택에 따라 제품을 확장할 때 SOA 또는 마이크로서비스로 변경할 수 있는 모듈식 아키텍처인지 확인해야 합니다. SOA와 마이크로서비스는 더 작게 분할할 수 있기 때문에 현대의 확장 가능하고 안정적인 아키텍처로 선호되지만 특히 마이크로서비스 아키텍처를 배포하는 경우 절충을 고려해야 합니다.

한 가지 주요 절충은 분산 컴퓨팅 아키텍처가 구축되므로 사용자 지연 시간 요구 사항을 충족하기가 더 어려워지며, 사용자 상호 작용을 디버그하고 추적하는 과정이 더 복잡해진다는 것입니다. AWS X-Ray을(를) 사용하면 이 문제를 해결할 수 있습니다. 관리 중인 애플리케이션의 수가 증가함에 따라 운영 복잡성이 증가하므로 다수의 독립 구성 요소를 배포해야 한다는 점도 고려해야 합니다.



모놀리식, 서비스 지향, 마이크로서비스 아키텍처

구현 단계

- 애플리케이션을 리팩터링 또는 구축하기에 적절한 아키텍처를 결정합니다. SOA 및 마이크로서비스는 상태적으로 더 세분화된 조각화를 제공하며, 이는 확장 가능하고 안정적인 최신 아키텍처로서 선호됩니다. SOA는 마이크로서비스의 복잡성을 어느 정도 피하면서 더 세분화된 조각화를 실현하기에 좋은 절충안이 될 수 있습니다. 자세한 내용은 Microservice Trade-Offs를 참조하세요.
- 이를 워크로드에 적용할 수 있고 조직에서 지원할 수 있는 경우, 마이크로서비스 아키텍처를 사용하여 최고의 민첩성과 안정성을 실현해야 합니다. 자세한 내용은 AWS에서 마이크로서비스 구현을 참조하세요.
- 모놀리식을 더 작은 구성 요소로 리팩터링하려면 스트랭글러 피그 패턴 을 따라 고려하세요. 여기에는 특정 애플리케이션을 새 애플리케이션 및 서비스로 점차적으로 교체하는 작업이 포함됩니다. AWS Migration Hub Refactor Spaces 은(는) 점진적 리팩터링을 위한 시작점 역할을 합니다. 자세한 내용은 스트랭글러 패턴을 사용하여 원활하게 온프레미스 레거시 워크로드 마이그레이션을 참조하세요.
- 마이크로 서비스를 구현하려면 분산된 서비스가 서로 통신하기 위한 서비스 검색 메커니즘이 필요할 수 있습니다. AWS App Mesh 을(를) 서비스 지향 아키텍처와 함께 사용하여 안정적인 서비스 검색 및 액세스를 제공할 수 있습니다. AWS Cloud Map 은(는) 동적 DNS 기반 서비스 검색에도 사용할 수 있습니다.
- 모놀리식에서 SOA로 마이그레이션하는 경우 Amazon MQ 은(는) 클라우드에서 레거시 애플리케이션을 다시 설계하는 경우 서비스 버스로 격차를 해소하는 데 도움이 될 수 있습니다.
- 공유 데이터베이스 하나를 사용하는 기존 모놀리식의 경우 데이터를 더 작은 세그먼트로 재구성하는 방법을 선택합니다. 사업부, 액세스 패턴 또는 데이터 구조별로 선택할 수 있습니다. 리팩터링 프로세스 중 이 지점에서 데이터베이스의 관계형 또는 비관계형(NoSQL) 유형을 사용하여 진행할지 선택해야 합니다. 자세한 내용은 SQL에서 NoSQL로를 참조하세요.

구현 계획의 작업 수준: 높음

리소스

관련 모범 사례:

- REL03-BP02 특정 비즈니스 도메인 및 기능을 중심으로 서비스 구축

관련 문서:

- Amazon API Gateway: OpenAPI를 사용하여 REST API 구성
- 서비스 지향 아키텍처란 무엇인가요?
- 경계 컨텍스트(도메인 주도 설계의 주요 패턴)
- AWS에서 마이크로서비스 구현
- Microservice Trade-Offs
- Microservices - a definition of this new architectural term
- AWS 마이크로서비스
- AWS App Mesh란 무엇인가요?

관련 예시:

- 반복적 앱 데이터베이스 현대화 워크숍

관련 동영상:

- Delivering Excellence with Microservices on AWS(AWS에서 마이크로서비스를 사용하여 우수성 제공)

REL03-BP02 특정 비즈니스 도메인 및 기능을 중심으로 서비스 구축

서비스 지향 아키텍처(SOA)는 비즈니스 요구 사항에 따라 명확하게 정의된 기능으로 서비스를 정의합니다. 마이크로서비스는 도메인 모델과 경계 컨텍스트를 사용하여 비즈니스 컨텍스트 경계를 따라 서비스 경계를 그립니다. 비즈니스 도메인 및 기능에 초점을 맞추면 팀이 서비스에 대한 독립적인 신뢰성 요구 사항을 정의하는 데 도움이 됩니다. 경계 컨텍스트는 비즈니스 로직을 분리하고 캡슐화하므로 팀이 장애 처리 방법을 더 잘 판단할 수 있습니다.

원하는 결과: 엔지니어와 비즈니스 이해 관계자가 공동으로 경계 컨텍스트를 정의하고 이를 사용하여 특정 비즈니스 기능을 충족하는 서비스로 시스템을 설계합니다. 이러한 팀은 이벤트 스토밍과 같은 확립된 관행을 사용하여 요구 사항을 정의합니다. 새로운 애플리케이션은 잘 정의된 경계와 느슨한 결합이 가능하도록 설계됩니다. 기존 모놀리스는 경계 컨텍스트로 분해되고 시스템 설계는 SOA 또는 마이크로서비스 아키텍처로 이동합니다. 모놀리스를 리팩터링하는 경우에는 버블 컨텍스트 및 모놀리스 분해 패턴과 같은 확립된 접근 방식이 적용됩니다.

도메인 지향 서비스는 상태를 공유하지 않는 하나 이상의 프로세스로 실행됩니다. 이들은 수요 변동에 독립적으로 대응하고 도메인별 요구 사항을 고려하여 장애 시나리오를 처리합니다.

일반적인 안티 패턴:

- 팀이 특정 비즈니스 도메인 대신 UI 및 UX, 미들웨어 또는 데이터베이스와 같은 특정 기술 도메인을 중심으로 구성됩니다.
- 애플리케이션이 여러 도메인 책임에 걸쳐 있습니다. 여러 경계 컨텍스트에 걸쳐 있는 서비스는 유지 관리가 더 어렵고 더 많은 테스트 작업이 필요하며 소프트웨어 업데이트에 여러 도메인 팀이 참여해야 할 수 있습니다.
- 도메인 엔터티 라이브러리와 같은 도메인 종속성은 서비스 간에 공유되므로 한 서비스 도메인을 변경하려면 다른 서비스 도메인도 변경해야 합니다.
- 서비스 계약과 비즈니스 로직은 엔터티를 공통적이고 일관된 도메인 언어로 표현하지 않으므로 변환 계층이 발생하여 시스템이 복잡해지고 디버깅 작업이 늘어납니다.

이 모범 사례 확립의 이점: 애플리케이션이 비즈니스 도메인을 기반으로 하는 독립적인 서비스로 설계되며 공통 비즈니스 언어를 사용합니다. 서비스를 독립적으로 테스트 및 배포할 수 있습니다. 서비스가 구현된 도메인에 대한 도메인별 복원력 요구 사항을 충족합니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 높음

구현 가이드

도메인 중심 의사 결정(DDD)은 비즈니스 도메인을 중심으로 소프트웨어를 설계하고 구축하는 기본 접근 방식입니다. 비즈니스 도메인에 초점을 맞춘 서비스를 구축할 때는 기존 프레임워크를 사용하는 것이 좋습니다. 기존 모놀리식 애플리케이션에서 작업할 때 애플리케이션을 서비스로 현대화하는 확립된 기술을 제공하는 분해 패턴을 활용할 수 있습니다.

도메인 중심 의사 결정

구현 단계

- 팀은 이벤트 스토밍 워크숍을 통해 이벤트, 명령, 집계, 도메인을 가벼운 스티커 노트 형식으로 빠르게 파악할 수 있습니다.
- 도메인 컨텍스트에서 도메인 엔터티 및 기능이 형성되면 유사한 기능 및 특성을 공유하는 엔터티가 함께 그룹화되는 경계 컨텍스트를사용하여 도메인을 서비스로 분할할 수 있습니다. 모델을 컨텍스트로 나누면 마이크로서비스의 경계를 지정하는 방법에 대한 템플릿을 사용할 수 있게 됩니다.

  - 예를 들어 Amazon.com 웹 사이트 엔터티에는 패키지, 배송, 일정, 가격, 할인 및 통화가 포함될 수 있습니다.
  - 패키지, 배송, 일정은 배송 컨텍스트로 그룹화되고 가격, 할인, 통화는 가격 컨텍스트로 그룹화됩니다.
- 모놀리스를 마이크로서비스로 분해 는 마이크로서비스 리팩터링 패턴을 설명합니다. 비즈니스 역량, 하위 도메인 또는 트랜잭션별 분해 패턴을 사용하는 것은 도메인 중심 접근 방식과 부합됩니다.
- 예를 들어 버블 컨텍스트 등 전술적 기술을 사용하면 사전 재작성 및 DDD에 대한 완전한 약정 없이 기존 또는 레거시 애플리케이션에 DDD를 도입할 수 있습니다. 버블 컨텍스트 접근 방식에서는 새로 정의된 도메인 모델을 외부 영향으로부터 보호하는 손상 방지 계층또는 서비스 매핑 및 조정을 사용하여 작은 경계 컨텍스트가 설정됩니다.

팀이 도메인 분석을 수행하고 엔티티 및 서비스 계약을 정의한 후에는 AWS 서비스를 활용하여 도메인 중심 설계를 클라우드 기반 서비스로 구현할 수 있습니다.

- 도메인의 비즈니스 규칙을 실행하는 테스트를 정의하여 개발을 시작하십시오. 테스트 기반 개발(TDD)과 동작 기반 개발(BDD)은 팀이 서비스가 비즈니스 문제 해결에 초점을 맞출 수 있도록 도와줍니다.
- 비즈니스 도메인 요구 사항을 가장 잘 충족하는 AWS 서비스 및 마이크로서비스 아키텍처를선택합니다.
  - AWS Serverless를 사용하면 팀이 서버 및 인프라를 관리하는 대신 특정 도메인 로직에 집중할 수 있습니다.
  - AWS의 컨테이너는 인프라 관리를 간소화하므로 도메인 요구 사항에 집중할 수 있습니다.
  - 목적별 데이터베이스를 통해 도메인 요구 사항을 가장 적합한 데이터베이스 유형에 맞출 수 있습니다.
- AWS에서 육각형 아키텍처 구축에서는 기능적 요구 사항을 충족한 다음 통합 어댑터를 연결하기 위해 비즈니스 도메인에서 역방향으로 작업하여 서비스에 비즈니스 로직을 구축하는 프레임워크를 설명합니다. AWS 서비스를 통해 인터페이스 세부 정보를 비즈니스 로직과 분리하는 패턴은 팀이 도메인 기능에 집중하고 소프트웨어 품질을 개선하는 데 도움이 됩니다.

리소스

관련 모범 사례:

- REL03-BP01 워크로드를 세그먼트화하는 방법 선택
- REL03-BP03 API별로 서비스 계약 제공

관련 문서:

- AWS 마이크로서비스
- AWS에서 마이크로서비스 구현
- 모놀리식 유형을 마이크로서비스로 분할하는 방법
- 레거시 시스템으로 둘러싸여 있을 때 DDD 시작
- 도메인 중심 설계: 소프트웨어 중심부의 복잡성 해결
- AWS에서 육각형 아키텍처 구축에서는
- 모놀리스를 마이크로서비스로 분해
- 이벤트 스토밍
- 경계 컨텍스트 간 메시지
- 마이크로서비스
- 테스트 기반 개발
- 동작 기반 개발

관련 예시:

- 엔터프라이즈 클라우드 네이티브 워크숍
- AWS에서 클라우드 네이티브 마이크로서비스 설계(DDD/EventStormingWorkshop 중)

관련 도구:

- AWS 클라우드 데이터베이스
- AWS의 서버리스
- AWS의 컨테이너는

REL03-BP03 API별로 서비스 계약 제공

서비스 계약은 컴퓨터가 인식할 수 있는 API 정의에 정의된 API 생산자 및 소비자 간의 문서화된 계약입니다. 계약 버전 관리 전략을 사용하면 소비자가 기존 API를 계속 사용하면서 준비가 될 때 애플리케이션을 최신 API로 마이그레이션할 수 있습니다. 생산자 배포는 계약을 준수하는 한 언제든지 가능합니다. 서비스 팀은 원하는 기술 스택을 사용하여 API 계약을 충족할 수 있습니다.

원하는 결과:

일반적인 안티 패턴: 서비스 지향 또는 마이크로서비스 아키텍처로 구축된 애플리케이션은 통합 런타임 종속성을 유지하면서 독립적으로 작동할 수 있습니다. API 소비자 또는 생산자에게 배포되는 변경 사항은 양측이 공통 API 계약을 준수하는 경우 전체 시스템의 안정성을 방해하지 않습니다. 서비스 API를 통해 통신하는 구성 요소는 독립적인 기능 릴리스를 수행하거나 런타임 종속성을 업그레이드하거나 서로 거의 또는 전혀 영향을 주지 않고 재해 복구(DR) 사이트로 장애 조치할 수 있습니다. 또한 개별 서비스는 다른 서비스를 함께 확장할 필요 없이 독립적으로 확장하여 리소스 수요를 흡수할 수 있습니다.

- 강력한 형식의 스키마 없이 서비스 API를 생성합니다. 이를 통해 프로그래밍 방식으로 검증할 수 없는 API 바인딩 및 페이로드를 생성하는 데 사용할 수 없는 API가 생성됩니다.
- 서비스 계약이 발전할 때 API 소비자가 업데이트 및 릴리스하거나 실패하도록 하는 버전 관리 전략을 채택하지 않습니다.
- 도메인 컨텍스트 및 언어에서의 통합 실패를 설명하는 대신 기본 서비스 구현의 세부 정보를 유출하는 오류 메시지.
- 서비스 구성 요소를 독립적으로 테스트할 수 있도록 API 계약을 사용하여 테스트 사례 및 모의 API 구현을 개발하지 않습니다.

이 모범 사례 확립의 이점: API 서비스 계약을 통해 통신하는 구성 요소로 구성된 분산 시스템은 신뢰성을 향상시킬 수 있습니다. 개발자는 컴파일 중에 형식 검사를 통해 개발 프로세스 초기에 잠재적 문제를 포착하여 요청 및 응답이 API 계약을 준수하고 필수 필드가 있는지 확인할 수 있습니다. API 계약은 API에 대한 명확한 자체 문서화 인터페이스를 제공하고 서로 다른 시스템과 프로그래밍 언어 간에 상호 운용성을 개선합니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 보통

구현 가이드

비즈니스 도메인을 식별하고 워크로드 세분화를 결정한 후에는 서비스 API를 개발할 수 있습니다. 먼저 컴퓨터가 인식할 수 있는 API 서비스 계약을 정의한 다음 API 버전 관리 전략을 구현합니다. REST, GraphQL 또는 비동기 이벤트와 같은 일반 프로토콜을 통해 서비스를 통합할 준비가 되면 AWS 서비스를 아키텍처에 통합하여 구성 요소를 강력한 형식의 API 계약과 통합할 수 있습니다.

서비스 API 계약을 위한 AWS 서비스

AWS 서비스(예: Amazon API Gateway, AWS AppSync 및 Amazon EventBridge)를 아키텍처에 통합하여 애플리케이션에서 API 서비스 계약을 사용할 수 있습니다. Amazon API Gateway는 네이티브 AWS 서비스 및 기타 웹 서비스와 직접 통합할 수 있도록 도와줍니다. API Gateway는 OpenAPI 사양 및 버전 관리를 지원합니다. AWS AppSync는 관리형 GraphQL 엔드포인트로, 쿼리, 변형, 구독을 위한

서비스 인터페이스를 정의하는 GraphQL 스키마를 정의하여 구성합니다. Amazon EventBridge는 이벤트 스키마를 사용하여 이벤트를 정의하고 이벤트에 대한 코드 바인딩을 생성합니다.

구현 단계

- 먼저 API에 대한 계약을 정의합니다. 계약은 API의 기능을 표현할 뿐만 아니라 API 입출력에 대해 강력한 형식의 데이터 객체와 필드를 정의합니다.
- API Gateway에서 API를 구성할 때 엔드포인트의 OpenAPI 사양을 가져오고 내보낼 수 있습니다.
  - OpenAPI 정의 가져오기 API 생성을 단순화하고 AWS Serverless Application Model 및 AWS Cloud Development Kit (AWS CDK)같은 AWS 코드형 인프라 도구와 통합될 수 있습니다.
  - API 정의 내보내기 API 테스트 도구와의 통합을 단순화하고 서비스 소비자에게 통합 사양을 제공합니다.
- AWS AppSync로 GraphQL API를 정의하고 관리할 수 있습니다. 이를 위해 GraphQL 스키마 파일을 정의하여 계약 인터페이스를 생성하고 복잡한 REST 모델, 여러 데이터베이스 테이블 또는 레거시 서비스와의 상호 작용을 단순화합니다.
- AWS Amplify 프로젝트는 AWS AppSync와 통합되어 애플리케이션에서 사용할 강력한 형식의 JavaScript 쿼리 파일과 Amazon DynamoDB 테이블용 AWS AppSync AppSync GraphQL 클라이언트 라이브러리를 생성합니다.
- Amazon EventBridge에서 서비스 이벤트를 사용하는 경우 이벤트는 스키마 레지스트리에 이미 있거나 OpenAPI Spec으로 정의한 스키마를 준수합니다. 레지스트리에 정의된 스키마를 사용하면 스키마 계약에서 클라이언트 바인딩을 생성하여 코드를 이벤트와 통합할 수도 있습니다.
- API 확장 또는 버전 관리. 선택적 필드 또는 필수 필드의 기본값으로 구성할 수 있는 필드를 추가할 때 더 간단한 옵션은 API를 확장하는 것입니다.
  - REST 및 GraphQL과 같은 프로토콜에 대한 JSON 기반 계약은 계약 확장에 적합할 수 있습니다.
  - SOAP와 같은 프로토콜에 대한 XML 기반 계약은 서비스 소비자와 함께 테스트하여 계약 연장 가능성을 결정해야 합니다.
- API 버전을 관리할 때는 단일 코드베이스에서 로직을 유지할 수 있도록 파사드를 사용하여 버전을 지원하는 프록시 버전 관리를 구현하는 것이 좋습니다.
  - API Gateway에서는 요청 및 응답 매핑을 사용하여 새 필드에 기본값을 제공하거나 요청 또는 응답에서 제거된 필드를 제거하는 파사드를 설정하여 계약 변경 사항을 간단하게 흡수할 수 있습니다. 이 접근 방식을 사용하면 기본 서비스가 단일 코드베이스를 유지할 수 있습니다.

리소스

관련 모범 사례:

- REL03-BP01 워크로드를 세그먼트화하는 방법 선택
- REL03-BP02 특정 비즈니스 도메인 및 기능을 중심으로 서비스 구축
- REL04-BP02 느슨하게 결합된 종속성 구현
- REL05-BP03 재시도 호출 제어 및 제한
- REL05-BP05 클라이언트 시간 제한 설정

관련 문서:

- API(애플리케이션 프로그래밍 인터페이스)란?
- AWS에서 마이크로서비스 구현
- Microservice Trade-Offs
- Microservices - a definition of this new architectural term
- AWS 마이크로서비스
- OpenAPI에 대한 API Gateway 확장 프로그램 작업
- OpenAPI 사양
- GraphQL: 스키마 및 형식
- Amazon EventBridge 코드 바인딩

관련 예시:

- Amazon API Gateway: OpenAPI를 사용하여 REST API 구성
- OpenAPI를 사용한 Amazon API Gateway-Amazon DynamoDB CRUD 애플리케이션
- 서버리스 시대의 현대적 애플리케이션 통합 패턴: API Gateway 서비스 통합
- Amazon CloudFront을 사용하여 헤더 기반 API Gateway 버전 관리 구현
- AWS AppSync: 클라이언트 애플리케이션 구축

관련 동영상:

- AWS SAM에서 OpenAPI를 사용하여 API Gateway 관리

관련 도구:

- Amazon API Gateway

- AWS AppSync
- Amazon EventBridge

REL 4 분산 시스템에서 장애 방지를 위한 상호 작용은 어떻게 설계합니까?

분산 시스템에서 구성 요소(예: 서버 또는 서비스)는 통신 네트워크를 사용하여 상호 연결됩니다. 워크로드는 이러한 네트워크에서 데이터 손실 또는 지연 시간이 발생하더라도 안정적으로 작동해야 합니다. 분산 시스템의 구성 요소는 다른 구성 요소나 워크로드에 부정적인 영향을 미치지 않는 방식으로 작동해야 합니다. 여기에 나온 모범 사례는 장애를 방지하고 MTBF(평균 장애 간격)를 개선합니다.

모범 사례

- REL04-BP01 필요한 분산 시스템의 종류 식별
- REL04-BP02 느슨하게 결합된 종속성 구현
- REL04-BP03 일정한 작업 수행
- REL04-BP04 모든 응답의 멱등성 유지

REL04-BP01 필요한 분산 시스템의 종류 식별

하드 실시간 분산 시스템에서는 응답을 동기식으로 신속하게 제공해야 하지만, 소프트 실시간 시스템에서는 응답하기 위한 몇 분 이상의 여유가 주어집니다. 오프라인 시스템은 배치 또는 비동기식 처리를 통해 응답을 처리합니다. 하드 실시간 분산 시스템은 안정성 요구 사항이 가장 엄격합니다.

요청/응답 서비스로 알려진 높은 수준의 실시간 분산 시스템과 관련된 문제가 분산 시스템의 가장 어려운 문제라고 할 수 있습니다. 이 문제가 어려운 이유는 언제 도착할지 예상할 수 없는 요청에 대해 신속하게 응답을 제공해야 하기 때문입니다(예: 응답이 올 때까지 앞에서 대기하는 고객). 예를 들어 프런트엔드 웹 서버, 주문 파이프라인, 신용 카드 거래, 모든 AWS API 및 전화 통신이 여기에 포함됩니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 필요한 분산 시스템의 종류를 식별합니다. 분산 시스템에는 지연 시간, 확장 및 축소, 네트워킹 API에 대한 이해, 데이터 마샬링 및 마샬링 취소와 알고리즘(예: Paxos)의 복잡성과 같은 당면 과제가 있었습니다. 시스템이 커지고 분산화가 확대되자, 이론적으로는 극단적 사례에 해당했던 문제들이 주기적으로 발생하기 시작했습니다.
  - Amazon Builders' Library: 분산 시스템의 도전 과제

- 하드 실시간 분산 시스템에서는 응답이 동기식으로 신속하게 제공되어야 합니다.
- 소프트 실시간 시스템은 상대적으로 응답 시간의 여유가 있어 분 단위 이상의 응답 시간이 허용됩니다.
- 오프라인 시스템은 배치 또는 비동기식 처리를 통해 응답을 처리합니다.
- 하드 실시간 분산 시스템은 안정성 요구 사항이 가장 엄격합니다.

리소스

관련 문서:

- Amazon EC2: Ensuring Idempotency(멱등성 보장)
- Amazon Builders' Library: 분산 시스템의 도전 과제
- Amazon Builders' Library: (안정성, 지속적인 작업 및 맛 좋은 한 잔의 커피)
- Amazon EventBridge란 무엇입니까?
- Amazon Simple Queue Service란 무엇입니까?

관련 동영상:

- AWS New York Summit 2019: Intro to Event-driven Architectures(이벤트 중심 아카텍처 및 Amazon EventBridge 소개) 및 Amazon EventBridge(MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small(루프 닫기 및 마음 열기: 규모에 상관없이 시스템을 제어하는 방법) ARC337(느슨한 결합, 일정한 작업, 정적 안정성 포함)
- AWS re:Invent 2019: Moving to event-driven architectures(이벤트 중심 아키텍처로 전환)(SVS308)

REL04-BP02 느슨하게 결합된 종속성 구현

대기열 처리 시스템, 스트리밍 시스템, 워크플로 및 로드 밸런서와 같은 종속성은 약결합됩니다. 느슨한 결합은 한 구성 요소의 동작을 다른 종속 구성 요소에서 분리하여 복원력 및 민첩성을 높이는 데 도움이 됩니다.

한 구성 요소에 대한 변경이 다른 종속 구성 요소의 변경을 강제하는 경우 이러한 구성 요소는 강하게 결합된 것입니다. 느슨한 결합에서는 이 종속성이 분리되므로 종속 구성 요소에서는 버전이 지정되고 게시된 인터페이스만 알면 됩니다. 종속성 간에 느스한 결합을 구현하면 한 구성 요소의 장애가 다른 구성 요소에 영향을 미치지 않도록 분리됩니다.

느슨한 결합을 사용하면 종속 구성 요소에 미치는 위험을 최소화하면서 추가 코드 또는 기능을 추가할 수 있습니다. 또한 종속성의 기본 구현을 스케일 아웃하거나 변경할 수 있으므로 확장성이 개선됩니다.

느슨한 결합을 통해 복원력을 추가로 개선하려면 가능한 경우 구성 요소가 비동기식으로 상호 작용하도록 합니다. 이 모델은 즉각적인 응답이 필요하지 않고 요청이 등록되었다는 확인으로 충분한 상호 작용에 적합합니다. 이러한 상호 작용에는 이벤트를 생성하는 구성 요소와 이벤트를 사용하는 구성 요소가 포함됩니다. 두 구성 요소는 직접적인 지점 간 상호 작용을 통해 통합되지 않으며 일반적으로 내구성이 있는 중간 스토리지 계층(예: SQS 대기열 또는 Amazon Kinesis 또는 AWS Step Functions와 같은 스트리밍 데이터 플랫폼)을 통해 통합됩니다.

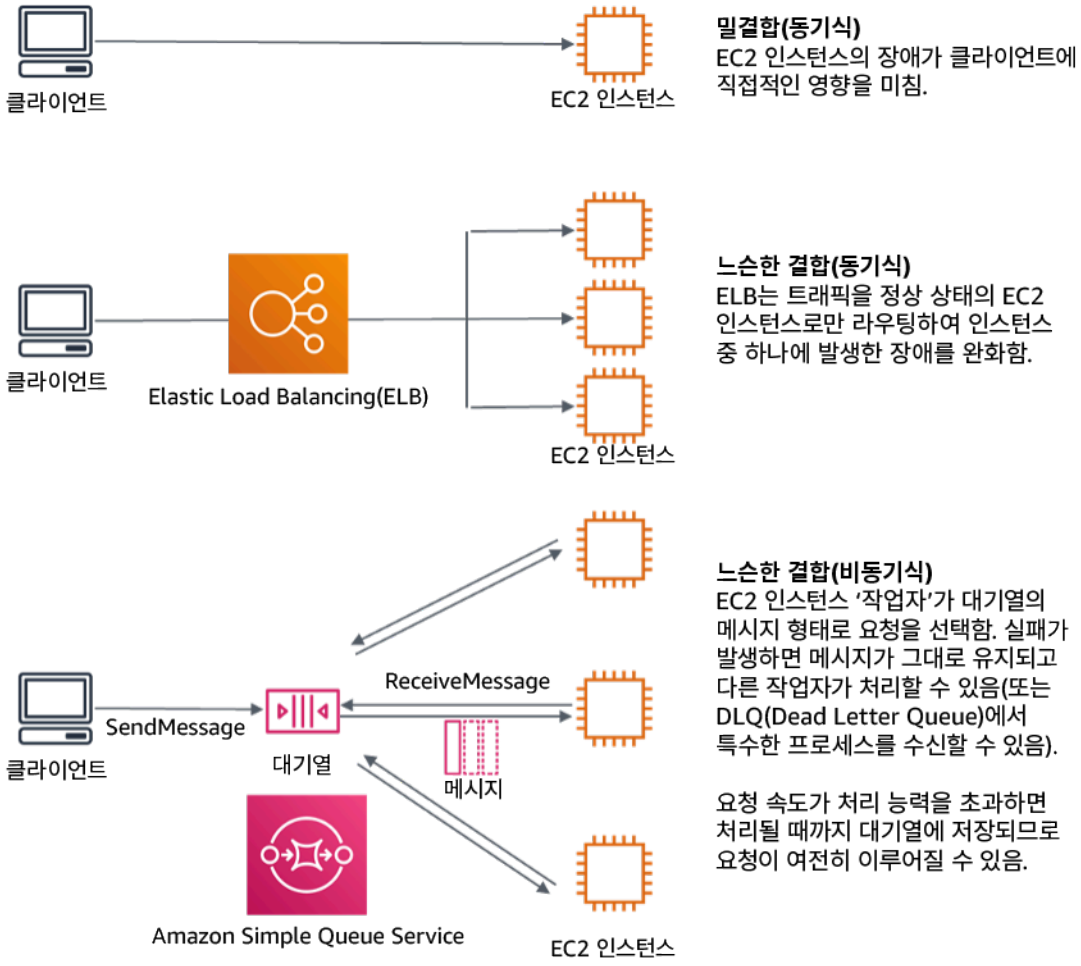


그림 4: 대기열 처리 시스템 및 로드 밸런서와 같은 종속성은 느슨하게 결합됨

Amazon SQS 대기열과 Elastic Load Balancer는 느슨한 결합을 위한 중간 계층을 추가할 수 있는 두 가지 방법의 예입니다. AWS 클라우드에서는 Amazon EventBridge를 사용하여 이벤트 기반 아키텍처를 구축할 수도 있습니다. Amazon EventBridge는 클라이언트(이벤트 소비자)가 사용하는 서비스에서 클라이언트(이벤트 생산자)를 추상화할 수 있습니다. Amazon Simple Notification Service(Amazon SNS)는 높은 처리량의 푸시 기반 다대다 메시징이 필요할 때 효과적인 솔루션입니다. Amazon SNS 주

제를 사용하면 게시자 시스템에서 다수의 구독자 엔드포인트로 메시지를 팬아웃하여 병렬 처리를 수행할 수 있습니다.

대기열은 다수의 장점을 제공하지만 대부분의 강성 실시간 시스템에서 임계 시간(주로 초 단위)을 초과한 요청은 무효한 요청(클라이언트가 포기하여 더 이상 응답을 기다리지 않는 요청)으로 간주되어 처리되지 않습니다. 이렇게 하면 오래된 요청 대신 여전히 유효한 요청일 가능성이 큰 최근 요청을 처리할 수 있습니다.

일반적인 안티 패턴:

- 워크로드의 일부로 싱글톤 배포
- 장애 조치 또는 비동기식 요청 처리 기능 없이 워크로드 티어 간에 API를 직접 호출

이 모범 사례 수립의 이점: 느슨한 결합은 한 구성 요소의 동작을 다른 종속 구성 요소에서 분리하여 복원력 및 민첩성을 높이는 데 도움이 됩니다. 한 구성 요소에서 발생한 장애는 다른 구성 요소로부터 격리됩니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 느슨하게 결합된 종속성을 구현합니다. 대기열 처리 시스템, 스트리밍 시스템, 워크플로 및 로드 밸런서와 같은 종속성은 약결합됩니다. 느슨한 결합은 한 구성 요소의 동작을 다른 종속 구성 요소에서 분리하여 복원력 및 민첩성을 높이는 데 도움이 됩니다.
  - AWS re:Invent 2019: Moving to event-driven architectures(이벤트 중심 아키텍처로 전환)(SVS308)
  - Amazon EventBridge란 무엇입니까?
  - Amazon Simple Queue Service란 무엇입니까?
    - Amazon EventBridge를 사용하면 느슨하게 결합되고 분산된 이벤트 중심의 아키텍처를 구축할 수 있습니다.
      - AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge(이벤트 중심 아카텍처 및 Amazon EventBridge 소개)(MAD205)
    - 한 구성 요소에 대한 변경이 다른 종속 구성 요소의 변경을 강제하는 경우 이러한 구성 요소는 강하게 결합된 것입니다. 느슨한 결합에서는 이 종속성이 분리되므로 종속 구성 요소에서는 버전이 지정되고 게시된 인터페이스만 알면 됩니다.
    - 가능한 경우 구성 요소 상호 작용을 비동기식으로 만듭니다. 이 모델은 즉각적인 응답이 필요하지 않고 요청이 등록되었다는 확인으로 충분한 상호 작용에 적합합니다.

- AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda(Amazon SQS 및 Lambda를 사용하는 확장 가능한 서버리스 이벤트 중심 애플리케이션)(API304)

리소스

관련 문서:

- AWS re:Invent 2019: Moving to event-driven architectures(이벤트 중심 아키텍처로 전환)(SVS308)
- Amazon EC2: Ensuring Idempotency(멱등성 보장)
- Amazon Builders' Library: 분산 시스템의 도전 과제
- Amazon Builders' Library: Reliability, constant work, and a good cup of coffee(안정성, 지속적인 작업 및 맛 좋은 한 잔의 커피)
- Amazon EventBridge란 무엇입니까?
- Amazon Simple Queue Service란 무엇입니까?

관련 동영상:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge(이벤트 중심 아카텍처 및 Amazon EventBridge 소개)(MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small(루프 닫기 및 마음 열기: 규모에 상관없이 시스템을 제어하는 방법) ARC337(느슨한 결합, 일정한 작업, 정적 안정성 포함)
- AWS re:Invent 2019: Moving to event-driven architectures(이벤트 중심 아키텍처로 전환)(SVS308)
- AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda(Amazon SQS 및 Lambda를 사용하는 확장 가능한 서버리스 이벤트 중심 애플리케이션)(API304)

REL04-BP03 일정한 작업 수행

대규모 로드가 급속도로 변경되면 시스템에서 장애가 발생할 수 있습니다. 예를 들어 서버 수천 대의 상태를 모니터링하는 상태 확인을 수행하는 워크로드의 경우 매번 동일한 크기의 페이로드(현재 상태의 전체 스냅샷)를 전송해야 합니다. 장애가 전혀 발생하지 않는 경우나, 또는 모든 서버에서 발생하는 경우와 무관하게 상태 확인 시스템은 대규모의 급속한 변경 없이 일정한 작업을 처리합니다.

예를 들어 상태 확인 시스템이 100,000개의 서버를 모니터링하는 경우 서버 장애율이 정상적으로 낮을 때는 서버에 가해지는 로드가 작습니다. 그러나 중대한 이벤트로 인해 이러한 서버의 절반이 비정상 상태가 될 때 상태 확인 시스템에서 알림 시스템을 업데이트하고 클라이언트로 상태를 전달하려면 상태 확인 시스템이 과부하가 될 수 있습니다. 그렇기 때문에 상태 확인 시스템에서는 매번 현재 상태의 전체 스냅샷을 전송하는 것이 낫습니다. 100,000개의 서버 상태(각각 비트로 표시됨)는 12.5KB 페이로드에 불과합니다. 장애가 발생한 서버가 없든 모든 서버에서 장애가 발생하든 상태 확인 시스템은 일정한 작업을 처리하므로 대규모의 급속한 변경이 시스템 안정성에 위협이 되지 않습니다. 이 방법으로 Amazon Route 53이 엔드포인트(예: IP 주소)에 대한 상태 확인을 수행하여 최종 사용자가 어떻게 엔드포인트에 라우팅되는지 판단합니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

- 로드에 대규모의 급격한 변화가 있을 때 시스템에서 장애가 발생하지 않도록 일정하게 작업을 수행합니다.
- 느슨한 결합 종속성을 구현합니다. 대기열 처리 시스템, 스트리밍 시스템, 워크플로 및 로드 밸런서와 같은 종속성은 약결합됩니다. 느슨한 결합은 한 구성 요소의 동작을 다른 종속 구성 요소에서 분리하여 복원력 및 민첩성을 높이는 데 도움이 됩니다.
  - Amazon Builders' Library: (안정성, 지속적인 작업 및 맛 좋은 한 잔의 커피)
  - AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small(루프 닫기 및 마음 열기: 규모에 상관없이 시스템을 제어하는 방법) ARC337(일정한 작업 포함)
    - 10만 개의 서버를 모니터링하는 상태 확인 시스템의 예에서 성공 또는 실패 횟수와 관계없이 페이로드 크기가 일정하게 유지되도록 워크로드를 엔지니어링합니다.

리소스

관련 문서:

- Amazon EC2: Ensuring Idempotency(멱등성 보장)
- Amazon Builders' Library: 분산 시스템의 도전 과제
- Amazon Builders' Library: (안정성, 지속적인 작업 및 맛 좋은 한 잔의 커피)

관련 동영상:

- AWS New York Summit 2019: Intro to Event-driven Architectures(이벤트 중심 아카텍처 및 Amazon EventBridge 소개) 및 Amazon EventBridge(MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small(루프 닫기 및 마음 열기: 규모에 상관없이 시스템을 제어하는 방법) ARC337(일정한 작업 포함)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small(루프 닫기 및 마음 열기: 규모에 상관없이 시스템을 제어하는 방법) ARC337(느슨한 결합, 일정한 작업, 정적 안정성 포함)
- AWS re:Invent 2019: Moving to event-driven architectures(이벤트 중심 아키텍처로 전환)(SVS308)

REL04-BP04 모든 응답의 멱등성 유지

멱등성이 있는 서비스는 각 요청이 정확히 한 번만 완료되도록 합니다. 이렇게 하면 다수의 동일한 요청에서 단일 요청과 동일한 결과가 나옵니다. 멱등성이 있는 서비스를 사용하면 클라이언트가 요청이 오류로 여러 번 처리될 것이라는 염려 없이 재시도를 시행할 수 있습니다. 이를 위해 클라이언트는 멱등성 토큰을 사용하여 API 요청을 실행할 수 있으며 요청이 반복될 때마다 동일한 토큰이 사용됩니다. 멱등성이 있는 서비스 API는 토큰을 사용하여 요청이 처음 완료되었을 때 반환된 응답과 동일한 응답을 반환합니다.

분산 시스템에서 작업을 최대 한 번(클라이언트가 한 번만 요청) 또는 최소 한 번(클라이언트가 성공 확인을 수신할 때까지 계속 요청) 수행하기는 쉽습니다. 그러나 작업의 멱등성을 보장하기는 어렵습니다. 즉 작업을 정확히 한 번 수행하여 다수의 동일한 요청에서 단일 요청과 동일한 결과를 얻기는 어렵습니다. API에서 멱등성 토큰을 사용하면 서비스가 중복 레코드를 생성하지 않고 부작용 없이 변경 요청을 한 번 이상 수신할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 모든 응답의 멱등성을 유지합니다. 멱등성이 있는 서비스는 각 요청이 정확히 한 번만 완료되도록 합니다. 이렇게 하면 다수의 동일한 요청에서 단일 요청과 동일한 결과가 나옵니다.
  - 클라이언트는 멱등성 토큰을 사용하여 API 요청을 실행할 수 있으며 요청이 반복될 때마다 동일한 토큰이 사용됩니다. 멱등성이 있는 서비스 API는 토큰을 사용하여 요청이 처음 완료되었을 때 반환된 응답과 동일한 응답을 반환합니다.
    - Amazon EC2: Ensuring Idempotency(멱등성 보장)

리소스

관련 문서:

- Amazon EC2: Ensuring Idempotency(멱등성 보장)
- Amazon Builders' Library: 분산 시스템의 도전 과제
- Amazon Builders' Library: Reliability, constant work, and a good cup of coffee(안정성, 지속적인 작업 및 맛 좋은 한 잔의 커피)

관련 동영상:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge(이벤트 중심 아카텍처 및 Amazon EventBridge 소개)(MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small(루프 닫기 및 마음 열기: 규모에 상관없이 시스템을 제어하는 방법) ARC337(느슨한 결합, 일정한 작업, 정적 안정성 포함)
- AWS re:Invent 2019: Moving to event-driven architectures(이벤트 중심 아키텍처로 전환)(SVS308)

REL 5 분산 시스템에서 장애 완화 또는 극복을 위한 상호 작용은 어떻게 설계합니까?

분산 시스템에서 구성 요소(예: 서버 또는 서비스)는 통신 네트워크를 사용하여 상호 연결됩니다. 워크로드는 이러한 네트워크에서 데이터 손실 또는 지연 시간이 발생하더라도 안정적으로 작동해야 합니다. 분산 시스템의 구성 요소는 다른 구성 요소나 워크로드에 부정적인 영향을 미치지 않는 방식으로 작동해야 합니다. 이러한 모범 사례를 준수하면 워크로드가 스트레스 또는 장애를 견디고, 더 빠르게 이를 복구하며, 이러한 장애의 영향을 완화할 수 있습니다. 그러면 결과적으로 MTTR(평균 복구 시간)이 개선됩니다.

모범 사례

- REL05-BP01 관련 하드 종속성을 소프트 종속성으로 변환하는 단계적 성능 저하 구현
- REL05-BP02 요청 제한
- REL05-BP03 재시도 호출 제어 및 제한
- REL05-BP04 빠른 실패 및 대기열 제한
- REL05-BP05 클라이언트 시간 제한 설정
- REL05-BP06 가능한 경우 서비스를 스테이트리스로 설계
- REL05-BP07 비상 레버 구현

REL05-BP01 관련 하드 종속성을 소프트 종속성으로 변환하는 단계적 성능 저하 구현

애플리케이션 구성 요소는 종속성을 사용할 수 없게 되더라도 핵심 기능을 계속 수행해야 합니다. 약간 오래된 데이터, 대체 데이터를 제공하거나 데이터를 제공하지 않을 수도 있습니다. 이를 통해 핵심 비즈니스 가치를 제공하는 동시에 지역화된 장애로 인한 전체 시스템 기능의 방해를 최소한으로 줄일 수 있습니다.

원하는 결과: 구성 요소의 종속성이 비정상 상태인 경우에도 구성 요소 자체가 성능이 저하된 방식으로 작동할 수 있습니다. 구성 요소의 장애 모드는 정상 작동으로 간주되어야 합니다. 워크플로는 이러한 장애가 완전한 장애로 이어지지 않거나 최소한 예측 가능하고 복구 가능한 상태로 이어지도록 설계되어야 합니다.

일반적인 안티 패턴:

- 필요한 핵심 비즈니스 기능을 식별하지 못합니다. 종속성 실패 시에도 구성 요소가 작동하는지 테스트하지 않습니다.
- 오류가 발생 시 또는 여러 종속성 중 하나만 사용할 수 없고 일부 결과가 여전히 반환될 수 있는 경우 데이터를 제공하지 않습니다.
- 트랜잭션이 부분적으로 실패하면 일관되지 않은 상태가 발생합니다.
- 중앙 파라미터 스토어에 액세스할 수 있는 다른 방법이 없습니다.
- 새로 고침 실패로 인한 결과를 고려하지 않고 로컬 상태를 무효화하거나 비웁니다.

이 모범 사례 확립의 이점: 단계적 성능 저하를 통해 시스템 전체의 가용성이 향상되고 장애가 발생하더라도 가장 중요한 기능의 작동을 유지할 수 있습니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 높음

구현 가이드

단계적 성능 저하를 구현하면 종속성 장애가 구성 요소 기능에 미치는 영향을 최소화하는 데 도움이 됩니다. 구성 요소가 종속성 장애를 감지하고 다른 구성 요소 또는 고객에게 미치는 영향을 최소화하는 방식으로 문제를 해결하는 것이 가장 좋습니다.

단계적 성능 저하를 고려하는 설계란 종속성 설계 시 잠재적인 장애 모드를 고려하는 것을 의미합니다. 각 장애 모드에서 구성 요소의 가장 중요한 기능을 대부분 또는 최소한 호출자 또는 고객에게 제공할 수 있는 방법을 마련하십시오. 이러한 고려 사항은 테스트 및 검증이 가능한 추가 요구 사항이 될 수 있습니다. 이상적으로는 구성 요소가 하나 이상의 종속성이 실패하더라도 적절한 방식으로 핵심 기능을 수행할 수 있어야 합니다.

이것은 기술적 논의인 만큼이나 비즈니스 논의이기도 합니다. 모든 비즈니스 요구 사항은 중요하며 가능하면 충족되어야 합니다. 그러나 모든 요구 사항이 충족되지 않을 때 어떤 일이 일어날지 묻는 것은 여전히 의미가 있습니다. 시스템은 가용성 및 일관성을 유지하도록 설계될 수 있지만 한 가지 요구 사항을 이루지 못하는 상황에서 어느 것이 더 중요할까요? 결제 처리의 경우 일관성이 필요할 수 있습니다. 실시간 애플리케이션의 경우 가용성이 필요할 수 있습니다. 고객 대상 웹 사이트의 경우 고객의 기대에 따라 답이 달라질 수 있습니다.

즉, 구성 요소의 요구 사항과 그 핵심 기능으로 간주되어야 하는 요소에 따라 달라진다는 의미입니다. 예를 들면 다음과 같습니다.

- 전자 상거래 웹 사이트는 맞춤형 추천, 최고 순위 제품, 고객 주문 상태 등 다양한 시스템의 데이터를 랜딩 페이지에 표시할 수 있습니다. 한 업스트림 시스템에 장애가 발생하더라도 고객에게 오류 페이지를 표시하는 대신 다른 모든 데이터를 표시하는 것이 좋습니다.
- 배치 쓰기를 수행하는 구성 요소는 개별 작업 중 하나가 실패하더라도 배치를 계속 처리할 수 있습니다. 재시도 메커니즘을 구현하는 것은 간단해야 합니다. 이렇게 하려면 어떤 작업이 성공했는지, 어떤 작업이 실패했는지, 왜 실패했는지에 대한 정보를 호출자에게 반환하거나 실패한 요청을 DLQ(Dead Letter Queue)에 넣어 비동기 재시도를 구현하면 됩니다. 실패한 작업에 대한 정보도 기록해야 합니다.
- 트랜잭션을 처리하는 시스템은 개별 업데이트가 모두 실행되었는지 또는 전혀 실행되지 않았는지 확인해야 합니다. 분산 트랜잭션의 경우 동일한 트랜잭션의 이후 작업이 실패할 경우 Saga 패턴을 사용하여 이전 작업을 롤백할 수 있습니다. 여기서 핵심은 일관성을 유지하는 것입니다.
- 시간이 중요한 시스템은 적시에 응답하지 않는 종속성을 처리할 수 있어야 합니다. 이러한 경우 회로 차단기 패턴을 사용할 수 있습니다. 종속성의 응답이 시간 제한하기 시작하면 시스템은 추가 호출이 이루어지지 않는 폐쇄 상태로 전환될 수 있습니다.
- 애플리케이션은 파라미터 스토어에서 파라미터를 읽을 수 있습니다. 기본 파라미터 세트를 사용하여 컨테이너 이미지를 생성하고 파라미터 스토어를 사용할 수 없는 경우에 이러한 이미지를 사용하는 것이 유용할 수 있습니다.

구성 요소 장애 시 사용되는 경로는 테스트가 필요하며 기본 경로보다 훨씬 간단해야 합니다. 일반적으로 폴백 전략은 피해야 합니다.같은 AWS 코드형 인프라 도구와 통합될 수 있습니다.

구현 단계

외부 및 내부 종속성을 식별합니다. 종속성에서 어떤 종류의 장애가 발생할 수 있는지 고려합니다. 이러한 장애 발생 시 업스트림 및 다운스트림 시스템과 고객에게 미치는 부정적인 영향을 최소화할 수 있는 방법을 강구합니다.

다음은 종속성 목록 및 장애 발생 시 단계적으로 성능을 저하시키는 방법입니다.

1. 종속성의 부분적 실패: 구성 요소가 다운스트림 시스템에 여러 요청을 보낼 수 있습니다. 이때 한 시스템에 여러 요청을 보내거나 여러 시스템에 한 번 요청할 수 있습니다. 비즈니스 상황에 따라 이를 처리하는 여러 방법이 적절할 수 있습니다(자세한 내용은 구현 지침의 이전 예시 참조).
2. 다운스트림 시스템이 높은 부하로 인해 요청을 처리할 수 없습니다. 다운스트림 시스템에 대한 요청이 지속적으로 실패하는 경우 계속 재시도하는 것은 의미가 없습니다. 이로 인해 이미 과부하된 시스템에 추가 부하가 발생하고 복구가 더 어려워질 수 있습니다. 여기서 회로 차단기 패턴을 활용하여 다운스트림 시스템에 대한 호출 실패를 모니터링할 수 있습니다. 많은 수의 호출이 실패하면 다운스트림 시스템으로의 추가 요청 전송이 중단되고 다운스트림 시스템을 다시 사용할 수 있는지 여부를 테스트하기 위한 호출이 가끔 전송됩니다.
3. 파라미터 스토어를 사용할 수 없습니다. 파라미터 스토어를 변환하기 위해 컨테이너 또는 머신 이미지에 포함된 소프트 종속성 캐싱 또는 정상적인 기본값을 사용할 수 있습니다. 참고로 이러한 기본값은 최신 상태로 유지되어야 하며 테스트 모음에 포함되어야 합니다.
4. 모니터링 서비스 또는 기타 비기능 종속성을 사용할 수 없습니다. 구성 요소가 간헐적으로 로그, 지표 또는 추적을 중앙 모니터링 서비스로 보낼 수 없는 경우에도 비즈니스 기능을 평소처럼 실행하는 것이 가장 좋은 경우가 많습니다. 오랫동안 자동으로 지표를 로깅 또는 푸시하지 않는 것은 허용되지 않는 경우가 많습니다. 또한 일부 사용 사례에서는 규정 준수 요구 사항을 충족하기 위해 완전한 감사 항목이 필요할 수 있습니다.
5. 관계형 데이터베이스의 프라이머리 인스턴스를 사용하지 못할 수 있습니다. 거의 모든 관계형 데이터베이스와 마찬가지로 Amazon Relational Database Service는 프라이머리 라이터 인스턴스를 하나만 가질 수 있습니다. 이로 인해 쓰기 워크로드에 단일 장애 지점이 생기고 규모 조정이 더 어려워집니다. 고가용성을 위한 다중 AZ 구성을 사용하거나 더 나은 규모 조정을 위해 Amazon Aurora 서버리스를 사용하면 이러한 문제를 부분적으로 완화할 수 있습니다. 고가용성 요구 사항이 매우 높은 경우 프라이머리 라이터에게 전혀 의존하지 않는 것이 좋습니다. 읽기만 하는 쿼리의 경우 읽기 전용 복제본을 사용할 수 있습니다. 이 복제본은 중복성과 스케일 업뿐만 아니라 스케일 아웃도 가능합니다. 예를 들어 Amazon Simple Queue Service 대기열에 쓰기를 버퍼링하여 프라이머리를 일시적으로 사용할 수 없는 경우에도 고객의 쓰기 요청을 계속 수락할 수 있습니다.

리소스

관련 문서:

- Amazon API Gateway: 처리량 향상을 위해 API 요청 제한
- CircuitBreaker('Release It!' 도서의 회로 차단기 요약)
- AWS에서의 오류 재시도 및 지수 백오프
- Michael Nygard 'Release It! Design and Deploy Production-Ready Software(Release It! 프로덕션 지원 소프트웨어 설계 및 배포)'

- Amazon Builders' Library: 분산 시스템의 폴백 방지
- Amazon Builders' Library: 심각한 수준의 대기열 백로그 방지
- Amazon Builders' Library: 캐싱 관련 당면 과제 및 전략
- Amazon Builders' Library: 시간 제한, 재시도 및 지터를 사용한 백오프

관련 동영상:

- 재시도, 백오프 및 지터: AWS re:Invent 2019: Introducing The Amazon Builders' Library(Amazon Builders' Library 소개)(DOP328)

관련 예시:

- Well-Architected lab: Level 300: Implementing Health Checks and Managing Dependencies to Improve Reliability(Well-Architected 실습: 레벨 300: 상태 확인 구현 및 종속성 관리를 통한 안정성 개선)

REL05-BP02 요청 제한

예상치 못한 수요 증가로 인한 리소스 고갈을 완화하기 위해 요청을 제한합니다. 제한 속도 이하의 요청은 처리되지만 정의된 한도를 초과한 요청은 거부되고 요청이 거부되었음을 알리는 메시지가 표시됩니다.

원하는 결과: 갑작스러운 고객 트래픽 증가, 플러딩 공격 또는 대량 재시도로 인한 대규모 볼륨 급증이 요청 제한을 통해 완화되므로 워크로드가 지원되는 요청 볼륨을 정상적으로 계속 처리할 수 있습니다.

일반적인 안티 패턴:

- API 엔드포인트 제한이 구현되지 않거나 예상 볼륨을 고려하지 않고 기본값으로 유지됩니다.
- API 엔드포인트가 로드 테스트되지 않거나 제한 한도가 테스트되지 않습니다.
- 요청 크기 또는 복잡성을 고려하지 않고 요청 속도를 제한합니다.
- 최대 요청 속도 또는 최대 요청 크기를 테스트하지만 둘 다 테스트하지는 않습니다.
- 리소스가 테스트에서 설정된 한도대로 프로비저닝되지 않습니다.
- 사용 계획이 애플리케이션 간(A2A) API 소비자를 위해 구성되거나 고려되지 않습니다.
- 수평적으로 조정되는 대기열 소비자에게는 최대 동시성 설정이 구성되어 있지 않습니다.
- IP 주소별 속도 제한이 구현되지 않았습니다.

이 모범 사례 확립의 이점: 제한 한도를 설정한 워크로드는 예상치 못한 볼륨 급증 상황에서도 정상적으로 작동하고 수락된 요청 로드를 성공적으로 처리할 수 있습니다. API 및 대기열에 대한 요청이 갑자기 또는 지속적으로 급증하면 요청이 제한되어 요청 처리 리소스가 소진되지 않습니다. 속도 제한은 개별 요청자를 제한하므로 단일 IP 주소 또는 API 소비자로부터 오는 대량의 트래픽이 리소스를 소진하여 다른 소비자에게 영향을 미치는 일이 없습니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 높음

구현 가이드

서비스는 알려진 용량의 요청을 처리하도록 설계되어야 합니다. 이 용량은 부하 테스트를 통해 설정할 수 있습니다. 요청 도착 속도가 한도를 초과할 경우 적절한 응답은 요청이 제한되었다는 신호를 보냅니다. 그러면 소비자가 오류를 처리하고 나중에 다시 시도할 수 있습니다.

서비스에 제한 구현이 필요한 경우 요청마다 토큰이 계산되는 토큰 버킷 알고리즘을 구현하는 것이 좋습니다. 토큰은 초당 제한 속도로 다시 충전되고 요청당 토큰 1개씩 비동기적으로 사용됩니다.

토큰 버킷 알고리즘.

Amazon API Gateway는 계정 및 리전 한도에 따라 토큰 버킷 알고리즘을 구현하며 사용 계획을 통해 클라이언트별로 구성할 수 있습니다. 또한 Amazon Simple Queue Service(Amazon SQS) 및 Amazon Kinesis는 요청을 버퍼링하여 요청 속도를 낮추고 처리 가능한 요청에 대해 더 높은 제한 속도를 허용할 수 있습니다. 마지막으로 AWS WAF를 사용하여 비정상적으로 높은 부하를 유발하는 특정 API 소비자를 제한하는 속도 제한을 구현할 수 있습니다.

구현 단계

API Gateway에서 API에 대한 제한 한도를 구성하고 제한이 초과되면 `429 Too Many Requests` 오류를 반환할 수 있습니다. AWS WAF를 AWS AppSync 및 API Gateway 엔드포인트와 함께 사용하여 IP 주소별 속도 제한을 활성화할 수 있습니다. 또한 시스템에서 비동기 처리를 허용할 수 있는 경우 메시지를 대기열 또는 스트림에 배치하여 서비스 클라이언트에 대한 응답 속도를 높일 수 있으며, 이를 통해 제한 속도를 높일 수 있습니다.

비동기 처리에서는 Amazon SQS를 AWS Lambda용 이벤트 소스로 구성한 경우 최대 동시성을 구성하여 높은 이벤트 속도가 워크로드 또는 계정의 다른 서비스에 필요한 사용 가능한 계정 동시 실행 할당량을 소모하지 않도록 할 수 있습니다.

API Gateway는 토큰 버킷의 관리형 구현을 제공하지만 API Gateway를 사용할 수 없는 경우 서비스용 토큰 버킷의 언어별 오픈 소스 구현(리소스의 관련 예제 참조)을 활용할 수 있습니다.

- 리전별 계정 수준, 단계별 API 및 사용 계획 수준별 API 키에서 API Gateway 제한 한도를 이해하고 구성합니다.
- 플러드로부터 보호하고 악성 IP를 차단하기 위해 AWS WAF 속도 제한 규칙을 API Gateway 및 AWS AppSync 엔드포인트에 적용합니다. A2A 소비자를 위한 AWS AppSync API 키에도 속도 제한 규칙을 구성할 수 있습니다.
- AWS AppSync API에 속도 제한보다 많은 제한 제어가 필요한지 고려하고 필요한 경우 AWS AppSync 엔드포인트 앞에 API Gateway를 구성합니다.
- Amazon SQS 대기열이 Lambda 대기열 소비자를 위한 트리거로 설정된 경우 최대 동시성을 서비스 수준 목표를 충족할 만큼 충분히 처리하지만 다른 Lambda 함수에 영향을 미치는 동시성 한도를 소비하지 않는 값으로 설정합니다. Lambda에서 대기열을 사용할 때는 동일한 계정 및 리전의 다른 Lambda 함수에 예약된 동시성을 설정하는 것이 좋습니다.
- Amazon SQS 또는 Kinesis에 대한 기본 서비스 통합과 함께 API Gateway를 사용하여 요청을 버퍼링합니다.
- API Gateway를 사용할 수 없는 경우 언어별 라이브러리를 참조하여 워크로드에 토큰 버킷 알고리즘을 구현합니다. 예제 섹션을 확인하고 직접 조사하여 적합한 라이브러리를 찾으십시오.
- 설정하려는 제한 또는 증가를 허용하려는 제한을 테스트하고 테스트된 제한을 문서화합니다.
- 테스트에서 설정한 범위를 초과하여 제한을 늘리지 마십시오. 제한을 늘릴 때는 증가를 적용하기 전에 프로비저닝된 리소스가 이미 테스트 시나리오의 리소스와 동일하거나 더 큰지 확인합니다.

리소스

관련 모범 사례:

- REL04-BP03 일정한 작업 수행
- REL05-BP03 재시도 호출 제어 및 제한

관련 문서:

- Amazon API Gateway: 처리량 향상을 위해 API 요청 제한
- AWS WAF: 속도 기반 규칙 문
- Amazon SQS를 이벤트 소스로 사용하는 경우 AWS Lambda의 최대 동시성 소개
- AWS Lambda: 최대 동시성

관련 예시:

- 가장 중요한 세 가지 AWS WAF 속도 기반 규칙
- Java Bucket4j
- Python 토큰 버킷
- Node 토큰 버킷
- .NET 시스템 스레딩 속도 제한

관련 동영상:

- AWS AppSync를 사용하여 GraphQL API 보안 모범 사례 구현

관련 도구:

- Amazon API Gateway
- AWS AppSync
- Amazon SQS
- Amazon Kinesis
- AWS WAF

REL05-BP03 재시도 호출 제어 및 제한

지수 백오프를 사용하여 각 재시도 간에 점진적으로 더 긴 간격으로 요청을 다시 시도합니다. 재시도 사이에 지터를 도입하여 재시도 간격을 무작위로 지정합니다. 최대 재시도 횟수를 제한합니다.

원하는 결과: 분산 소프트웨어 시스템의 일반적인 구성 요소로는 서버, 로드 밸런서, 데이터베이스, DNS 서버가 있습니다. 이러한 구성 요소는 정상 작동 중에 일시적 또는 제한적인 오류로 또한 재시도에 관계없이 지속되는 오류로 요청에 응답할 수 있습니다. 클라이언트가 서비스에 요청을 전송할 때 요청은 메모리, 스레드, 연결, 포트 또는 기타 제한된 리소스를 포함하여 리소스를 소비합니다. 재시도를 제어하고 제한하는 것은 부하가 걸리는 시스템 구성 요소가 과부하되지 않도록 리소스 소비를 줄이고 최소화하기 위한 전략입니다.

클라이언트는 시간이 초과되거나 오류 응답을 수신하면 재시도 여부를 결정해야 합니다. 재시도할 경우 지터 및 최대 재시도 값이 포함된 지수 백오프가 발생합니다. 그 결과 백엔드 서비스 및 프로세스에서 부하가 감소하고 자가 복구 시간이 단축되어 복구 속도가 빨라지고 요청 처리가 성공적으로 이루어질 수 있습니다.

일반적인 안티 패턴:

- 지수 백오프, 지터, 최대 재시도 값을 추가하지 않고 재시도를 구현합니다. 백오프 및 지터는 의도치 않게 공통 간격으로 조정된 재시도로 인한 인위적인 트래픽 급증을 방지하는 데 도움이 됩니다.
- 효과를 테스트하지 않고 재시도를 구현하거나 재시도 시나리오를 테스트하지 않고 SDK에 재시도가 이미 내장되어 있다고 가정합니다.
- 종속성에서 게시된 오류 코드를 이해하지 못하여 권한 부족을 나타내는 명확한 오류, 구성 오류 또는 수동 개입 없이는 해결되지 않을 것으로 예측되는 기타 조건을 포함하여 모든 오류를 재시도합니다.
- 반복되는 서비스 장애에 대한 모니터링 및 경고를 포함하여 근본적인 문제를 파악하고 해결할 수 있도록 하는 관찰성 관행을 다루지 않습니다.
- 기본 제공 또는 타사 재시도 기능이 충분할 때 사용자 지정 재시도 메커니즘을 개발합니다.
- 재시도를 복잡하게 만드는 방식으로 애플리케이션 스택의 여러 계층에서 재시도하여 대량 재시도로 리소스가 더 소모됩니다. 이러한 오류가 애플리케이션의 종속성에 어떤 영향을 미치는지 파악한 다음 한 수준에서만 재시도를 구현해야 합니다.
- 멱등성이 아닌 서비스 호출을 재시도하여 중복 결과 등 예상치 못한 부작용을 초래합니다.

이 모범 사례 확립의 이점: 재시도는 요청이 실패했을 때 클라이언트가 원하는 결과를 얻을 수 있도록 도와주지만, 원하는 응답을 받는 데 서버 시간을 더 많이 소비하기도 합니다. 오류가 드물거나 일시적인 경우 재시도가 유용합니다. 리소스 과부하로 인해 장애가 발생한 경우 재시도는 상황을 악화시킬 수 있습니다. 클라이언트 재시도 시 지터와 함께 지수 백오프를 추가하면 리소스 과부하로 인한 장애 발생 시 서버를 복구할 수 있습니다. 지터는 요청이 집중되어 급증하는 것을 방지하고 백오프는 일반 요청 부하에 재시도를 추가하여 발생하는 부하 에스컬레이션을 줄입니다. 마지막으로, 준안정 장애를 초래하는 백로그가 생성되지 않도록 최대 재시도 횟수 또는 경과 시간을 구성하는 것이 중요합니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 높음

구현 가이드

재시도 호출을 제어하고 제한합니다. 지수 백오프를 사용하여 점진적으로 더 긴 간격 후에 다시 시도합니다. 지터를 도입하여 재시도 간격을 무작위로 지정하고 최대 재시도 횟수를 제한합니다.

일부 AWS SDK는 기본적으로 재시도와 지수 백오프를 구현합니다. 해당하는 경우 워크로드에 이러한 기본 제공 AWS 구현을 사용합니다. 멱등성이 있는 서비스를 호출하고 재시도가 클라이언트 가용성을 향상시키는 경우 워크로드에 유사한 로직을 구현합니다. 사용 사례를 기반으로 시간 제한 대상 및 재시도 중단 시점을 결정합니다. 이러한 재시도 사용 사례에 대한 테스트 시나리오를 구축하고 실행합니다.

구현 단계

- 애플리케이션 스택에서 애플리케이션이 의존하는 서비스에 대한 재시도를 구현하기 위한 최적의 계층을 결정합니다.
- 선택한 언어에 대해 지수 백오프 및 지터가 포함된 검증된 재시도 전략을 구현하는 기존 SDK를 파악하고 직접 재시도 구현을 작성하는 것보다 이러한 전략을 우선적으로 사용합니다.
- 재시도를 구현하기 전에 서비스가 멱등성인지 확인합니다. 재시도를 구현한 후에는 반드시 테스트를 거치고 프로덕션 환경에서 정기적으로 실행해야 합니다.
- AWS 서비스 API를 호출할 때는 AWS SDK 및 AWS CLI를 사용하고 재시도 구성 옵션을 이해합니다. 기본값이 사용 사례에 맞는지 확인하고 필요에 따라 테스트하고 조정합니다.

리소스

관련 모범 사례:

- ???
- REL05-BP02 요청 제한
- REL05-BP04 빠른 실패 및 대기열 제한
- REL05-BP05 클라이언트 시간 제한 설정
- REL11-BP01 워크로드의 모든 구성 요소를 모니터링하여 장애 감지

관련 문서:

- AWS에서의 오류 재시도 및 지수 백오프
- Amazon Builders' Library: 시간 제한, 재시도 및 지터를 사용한 백오프
- Exponential Backoff and Jitter
- 멱등성 API로 안전한 재시도

관련 예시:

- Spring 재시도
- Resilience4j 재시도

관련 동영상:

- 재시도, 백오프 및 지터: AWS re:Invent 2019: Introducing The Amazon Builders' Library(Amazon Builders' Library 소개)(DOP328)

관련 도구:

- AWS SDK 및 도구: 재시도 동작
- AWS Command Line Interface: AWS CLI 재시도

REL05-BP04 빠른 실패 및 대기열 제한

서비스가 요청에 제대로 응답하지 못하면 빠르게 실패합니다. 이렇게 하면 요청에 연결된 리소스를 해제할 수 있고 리소스가 부족한 경우 서비스 복구를 허용할 수 있습니다. 빠른 실패는 클라우드에서 매우 안정적인 워크로드를 구축하기 위해 활용할 수 있는 잘 정립된 소프트웨어 설계 패턴입니다. 대기열도 잘 정립된 엔터프라이즈 통합 패턴으로, 이를 통해 로드를 원활하게 수행하고 비동기 처리가 허용될 때 클라이언트가 리소스를 해제할 수 있습니다. 서비스가 정상 상태에서는 제대로 응답할 수 있지만 요청 속도가 너무 높아서 실패하는 경우 대기열을 사용하여 요청을 버퍼링하십시오. 하지만 긴 대기열 백로그가 쌓이도록 두지 마십시오. 그러면 클라이언트가 이미 포기한 무효 요청을 처리할 수 있습니다.

원하는 결과: 시스템에서 리소스 경합, 시간 제한, 예외 또는 회색 실패가 발생하여 서비스 수준 목표를 달성할 수 없는 경우 빠른 실패 전략을 통해 시스템 복구 시간을 단축할 수 있습니다. 트래픽 스파이크를 흡수해야 하고 비동기 처리를 수용할 수 있는 시스템은 클라이언트가 대기열을 사용하여 요청을 백엔드 서비스에 버퍼링함으로써 요청을 신속하게 해제할 수 있도록 하여 신뢰성을 높일 수 있습니다. 요청을 대기열로 버퍼링할 때 대처할 수 없는 백로그를 방지하기 위해 대기열 관리 전략이 구현됩니다.

일반적인 안티 패턴:

- 메시지 대기열을 구현하지만 DLQ(Dead Letter Queue) 볼륨에 DLQ 또는 경보를 구성하여 시스템 장애 시점을 감지하지는 않습니다.
- 대기열 소비자가 지연되거나 오류로 인해 재시도되는 시기를 파악하기 위해 대기열에 있는 메시지의 대기 기간을 측정하는 것이 아니라 지연 시간을 측정합니다.

- 업무상 더 이상 필요하지 않아 이러한 메시지를 처리할 가치가 없는 경우 대기열에서 백로깅된 메시지를 지우지 않습니다.
- 후입선출(LIFO) 대기열이 클라이언트 요구 사항을 더 잘 충족할 때 선입선출(FIFO) 대기열을 구성합니다. 예를 들어 엄격한 순서가 필요하지 않고 백로그 처리가 모든 신규 요청과 시간이 중요한 요청을 지연시켜 모든 클라이언트가 서비스 수준 위반을 경험하는 경우입니다.
- 작업 접수를 관리하고 요청을 내부 대기열에 배치하는 API를 노출하는 대신 내부 대기열을 클라이언트에 노출합니다.
- 아주 많은 작업 요청 유형을 단일 대기열에 결합합니다. 그러면 리소스 수요가 요청 유형 전체에 분산되어 백로그 상태가 악화될 수 있습니다.
- 서로 다른 모니터링, 시간 제한, 리소스 할당이 필요함에도 복잡한 요청과 단순한 요청을 동일한 대기열에서 처리합니다.
- 소프트웨어에서 오류를 정상적으로 처리할 수 있는 상위 수준 구성 요소에 예외를 발생시키는 빠른 실패 메커니즘을 구현하기 위해 입력의 유효성을 확인하거나 어설션을 사용하지 않습니다.
- 장애 리소스를 요청 라우팅에서 제거하지 않습니다(특히 장애가 충돌 및 다시 시작, 간헐적인 종속성 장애, 용량 감소 또는 네트워크 패킷 손실로 인한 실패 및 성공을 모두 표시하는 회색인 경우).

이 모범 사례 확립의 이점: 빠르게 실패하는 시스템은 디버깅과 수정이 더 쉬우며 릴리스가 프로덕션 환경에 게시되기 전에 코딩과 구성 문제를 드러내는 경우가 많습니다. 효과적인 대기열 전략이 통합된 시스템은 트래픽 급증 및 간헐적인 시스템 장애 상태에 대한 복원력과 신뢰성을 높입니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 높음

구현 가이드

빠른 실패 전략은 소프트웨어 솔루션에 코딩할 수 있을 뿐만 아니라 인프라에 구성할 수도 있습니다. 대기열은 빠른 실패뿐만 아니라 시스템 구성 요소를 분리하여 부하를 원활하게 하는 간단하면서도 강력한 아키텍처 기법입니다. Amazon CloudWatch는 장애를 모니터링하고 경보하는 기능을 제공합니다. 시스템에 장애가 발생한 것으로 확인되면 손상된 리소스를 제거하는 등 완화 전략을 실행할 수 있습니다. 시스템이 원활한 로드를 위해 Amazon SQS 및 기타 대기열 기술로 대기열을 구현할 때 대기열 백로그와 메시지 소비 실패를 관리하는 방법을 고려해야 합니다.

구현 단계

- 소프트웨어에 프로그래밍 방식 어설션 또는 특정 메트릭을 구현하고 이를 사용하여 시스템 문제에 대해 명시적으로 경고합니다. Amazon CloudWatch는 애플리케이션 로그 패턴과 SDK 계측을 기반으로 지표 및 경보를 생성하는 데 도움이 됩니다.

- CloudWatch 지표 및 경보를 사용하여 처리 지연 시간을 늘리거나 반복적으로 요청 처리를 실패하는 손상된 리소스를 차단합니다.
- 백엔드 대기열 소비자가 요청을 처리하는 동안 클라이언트가 리소스를 해제하고 다른 작업을 계속할 수 있도록 요청을 수락하고 Amazon SQS를 사용하여 내부 대기열에 요청을 추가한 다음 메시지 생성 클라이언트에 성공 메시지로 응답하도록 API를 설계하여 비동기 처리를 사용합니다.
- 현재 시간과 메시지 타임스탬프를 비교해 대기열에서 메시지를 제거할 때마다 CloudWatch 지표를 생성하여 대기열 처리 대기 시간을 측정하고 모니터링합니다.
- 장애가 발생하여 메시지 처리가 실패했거나 서비스 수준 계약 내에서 처리할 수 없는 트래픽 스파이크가 발생하는 경우 오래된 트래픽이나 초과 트래픽을 스필오버 대기열로 넘깁니다. 이를 통해 새 작업을 우선적으로 처리하고 용량이 사용 가능한 경우 이전 작업을 우선적으로 처리할 수 있습니다. 이 기법은 LIFO 처리와 유사하며 모든 새 작업에 대해 정상적인 시스템 처리가 가능합니다.
- DLQ(Dead Letter Queue) 또는 재구동 대기열을 사용하여 백로그에서 처리할 수 없는 메시지를 나중에 조사하고 해결할 수 있는 위치로 옮깁니다.
- 이전 메시지를 재시도하거나 허용되는 경우 현재 시간을 메시지 타임스탬프와 비교하고 더 이상 요청 클라이언트와 관련이 없는 메시지를 삭제합니다.

리소스

관련 모범 사례:

- REL04-BP02 느슨하게 결합된 종속성 구현
- REL05-BP02 요청 제한
- REL05-BP03 재시도 호출 제어 및 제한
- REL06-BP02 지표 정의 및 계산(집계)
- REL06-BP07 시스템을 통한 요청의 엔드 투 엔드 추적 모니터링

관련 문서:

- 심각한 수준의 대기열 백로그 방지
- 빠른 실패
- 내 Amazon SQS 대기열에서 메시지 백로그가 증가하는 것을 방지하려면 어떻게 해야 합니까?
- Elastic Load Balancing: 영역 전환
- Amazon Route 53 Application Recovery Controller: 트래픽 장애 조치를 위한 라우팅 제어

관련 예시:

- 엔터프라이즈 통합 패턴: 배달 못한 편지 채널

관련 동영상:

- AWS re:Invent 2022 - Operating highly available Multi-AZ applications(고가용성 다중 AZ 애플리케이션 운영)

관련 도구:

- Amazon SQS
- Amazon MQ
- AWS IoT Core
- Amazon CloudWatch

REL05-BP05 클라이언트 시간 제한 설정

연결 및 요청에 대한 시간 제한을 적절하게 설정하고 체계적으로 확인합니다. 워크로드 세부 사항을 인식하지 못하므로 기본값에 의존하지 마십시오.

원하는 결과: 클라이언트 시간 제한은 완료에 비정상적으로 많은 시간이 걸리는 요청 대기와 관련된 클라이언트, 서버, 워크로드의 비용을 고려해야 합니다. 시간 제한의 정확한 원인을 알 수 없기 때문에 클라이언트는 서비스에 대한 지식을 활용하여 생각되는 원인과 적절한 시간 제한에 대한 기대치를 세워야 합니다.

구성된 값에 따라 클라이언트 연결이 시간 제한됩니다. 시간 제한이 발생한 후 클라이언트는 작업을 중단하고 재시도 또는 회로 차단기개방을 결정합니다. 이러한 패턴에서는 근본적인 오류 상태를 악화시킬 수 있는 요청 발행이 방지됩니다.

일반적인 안티 패턴:

- 시스템 시간 제한 또는 기본 시간 제한을 인식하지 못합니다.
- 정상적인 요청 완료 타이밍을 인식하지 못합니다.
- 요청을 완료하는 데 비정상적으로 오래 걸리는 원인 또는 이러한 완료를 기다리는 데 따른 클라이언트, 서비스 또는 워크로드 성능의 비용을 인식하지 못합니다.

- 네트워크가 손상되어 시간 제한에 도달한 후에만 요청이 실패할 확률과 시간 제한을 더 짧게 설정하지 않아 클라이언트와 워크로드에 발생할 수 있는 비용을 인식하지 못합니다.
- 연결 및 요청 모두에 대한 시간 제한 시나리오를 테스트하지 않습니다.
- 시간 제한을 매우 높게 설정합니다. 그러면 지연 시간이 길어지고 리소스 사용률이 증가할 수 있습니다.
- 시간 제한을 매우 낮게 설정합니다. 그러면 인위적인 오류가 발생합니다.
- 회로 차단기 및 재시도와 같은 원격 호출의 시간 제한 오류를 처리하기 위한 패턴을 간과합니다.
- 서비스 호출 오류율, 지연 시간에 대한 서비스 수준 목표, 지연 시간 이상치에 대한 모니터링을 고려하지 않습니다. 이러한 지표를 통해 공격적이거나 허용되는 시간 제한의 인사이트를 얻을 수 있습니다.

이 모범 사례 확립의 이점: 원격 호출 시간 제한이 구성되고 시스템이 시간 제한을 정상적으로 처리하도록 설계되어 원격 호출이 비정상적으로 느리게 응답하고 서비스 클라이언트에서 시간 제한 오류를 정상적으로 처리할 때 리소스가 절약됩니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 높음

구현 가이드

서비스 종속성 호출과 일반적으로 프로세스 전체의 모든 호출에 연결 시간 제한과 요청 시간 제한을 모두 설정합니다. 많은 프레임워크가 시간 제한 기능을 기본 제공하지만 일부 프레임워크에는 서비스 목표에 허용되는 것보다 높거나 무한한 기본값이 있으므로 주의해야 합니다. 값이 너무 높으면 클라이언트가 시간 제한이 발생할 때까지 대기하는 동안 리소스가 계속 소비되기 때문에 시간 제한의 유용성이 감소합니다. 값이 너무 낮으면 백엔드에서 트래픽이 증가하고 너무 많은 요청이 재시도되므로 지연 시간이 증가할 수 있습니다. 일부 경우에는 모든 요청이 재시도되기 때문에 이로 인해 전체가 중단될 수 있습니다.

시간 제한 전략을 결정할 때는 다음 사항을 고려하십시오.

- 요청의 내용, 대상 서비스의 장애 또는 네트워킹 파티션 장애로 인해 요청을 처리하는 데 평소보다 시간이 오래 걸릴 수 있습니다.
- 비정상적으로 비용이 많이 드는 콘텐츠를 요청하면 불필요한 서버 및 클라이언트 리소스가 소모될 수 있습니다. 이 경우 이러한 요청을 시간 초과시키고 재시도하지 않는 것이 리소스를 보존할 수 있습니다. 또한 서비스는 스로틀 및 서버 측 시간 제한으로 비정상적으로 비용이 많이 드는 콘텐츠로부터 스스로를 보호해야 합니다.
- 서비스 장애로 인해 비정상적으로 오래 걸리는 요청은 시간 제한 후 재시도할 수 있습니다. 요청 및 재시도에 따른 서비스 비용을 고려해야 하지만, 원인이 국소적인 장애인 경우 재시도는 비용이 많이

들지 않으며 클라이언트 리소스 소비를 줄일 수 있습니다. 장애의 특성에 따라 시간 제한으로 인해 서버 리소스가 해제될 수도 있습니다.

- 네트워크에서 요청 또는 응답을 전달하지 못해 완료하는 데 시간이 오래 걸리는 요청은 시간 초과 후 재시도할 수 있습니다. 요청 또는 응답이 전달되지 않았으므로 시간 제한의 길이와 상관없이 실패했을 것입니다. 이 경우 시간 초과로 인해 서버 리소스가 해제되지는 않지만 클라이언트 리소스가 해제되고 워크로드 성능이 향상됩니다.

재시도 및 회로 차단기와 같이 잘 정립된 설계 패턴을 활용하여 시간 제한을 원활하게 처리하고 빠른 실패 접근 방식을 지원할 수 있습니다. AWS SDK 및 AWS CLI는 연결 및 요청 시간 제한을 모두 구성하고 지수 백오프 및 지터를 통한 재시도를 구성할 수 있습니다. AWS Lambda 함수는 시간 제한 구성을 지원하고 AWS Step Functions에서는AWS 서비스 및 SDK와의 사전 구축된 통합을 활용하는 로우코드 회로 차단기를 구축할 수 있습니다. AWS App Mesh Envoy는 시간 제한 및 회로 차단기 기능을 제공합니다.

구현 단계

- 원격 서비스 호출에 대한 시간 제한을 구성하고 기본 제공되는 언어 시간 제한 기능 또는 오픈 소스 시간 제한 라이브러리를 활용합니다.
- 워크로드가 AWS SDK를 사용하여 호출하는 경우 설명서에서 언어별 시간 제한 구성을 검토하십시오.
  - Python
  - PHP
  - .NET
  - Ruby
  - Java
  - Go
  - Node.js
  - C++
- 워크로드에서 AWS SDK 또는 AWS CLI 명령을 사용하는 경우 `connectTimeoutInMillis` 및 `tlsNegotiationTimeoutInMillis`에 대한 AWS 구성 기본값을 설정하여 기본 시간 제한을 구성합니다.
- 명령줄 옵션 `cli-connect-timeout` 및 `cli-read-timeout#` 적용하여 AWS 서비스에 대한 일회성 AWS CLI 명령을 제어합니다.

- 오류 시나리오를 사전에 처리할 수 있도록 원격 서비스 호출의 시간 제한을 모니터링하고 지속적인 오류에 대한 경고를 설정합니다.
- 호출 오류율, 대기 시간에 대한 서비스 수준 목표, 대기 시간 이상값에 대한 CloudWatch 지표 및 CloudWatch 이상 탐지를 구현하여 과도하게 공격적이거나 허용적인 시간 제한 관리의 인사이트를 얻습니다.
- 시간 제한을 Lambda 함수에구성합니다.
- API Gateway 클라이언트는 시간 제한을 처리할 때 자체 재시도를 구현해야 합니다. API Gateway는 다운스트림 통합에 대해 50밀리초~29초의 통합 시간 제한을 지원하고 통합 요청 시간 제한 시 재시도하지 않습니다.
- 원격 호출이 시간 제한하는 경우 원격 호출을 방지하는 회로 차단기 패턴을 구현합니다. 호출이 실패하지 않도록 회로를 개방하고 호출이 정상적으로 응답할 때 회로를 폐쇄합니다.
- 컨테이너 기반 워크로드의 경우 App Mesh Envoy 기능을 검토하여 기본 제공 시간 제한 및 회로 차단기를 활용합니다.
- 특히 AWS 네이티브 SDK 및 지원되는 Step Functions 통합을 호출하여 워크로드를 간소화하는 경우 AWS Step Functions를 사용하여 원격 서비스 호출을 위한 로우 코드 회로 차단기를 구축합니다.

리소스

관련 모범 사례:

- REL05-BP03 재시도 호출 제어 및 제한
- REL05-BP04 빠른 실패 및 대기열 제한
- REL06-BP07 시스템을 통한 요청의 엔드 투 엔드 추적 모니터링

관련 문서:

- AWS SDK: 재시도 및 시간 제한
- Amazon Builders' Library: 시간 제한, 재시도 및 지터를 사용한 백오프
- Amazon API Gateway 할당량 및 중요 참고 사항
- AWS Command Line Interface: 명령줄 옵션
- AWS SDK for Java 2.x: API 시간 제한 구성
- 구성 객체 및 구성 참조를 사용하는 AWS Botocore
- AWS SDK for .NET: 재시도 및 시간 제한
- AWS Lambda: Lambda 함수 옵션 구성

관련 예시:

- AWS Step Functions 및 Amazon DynamoDB와 함께 회로 차단기 패턴 사용
- Martin Fowler: CircuitBreaker

관련 도구:

- AWS SDK
- AWS Lambda
- Amazon SQS
- AWS Step Functions
- AWS Command Line Interface

REL05-BP06 가능한 경우 서비스를 스테이트리스로 설계

서비스는 상태를 필요로 하지 않거나 디스크 및 메모리의 로컬로 저장된 데이터에 종속성이 없도록 서로 다른 클라이언트 요청 간에 상태를 오프로드해야 합니다. 이를 통해 가용성에 영향을 주지 않고 서버를 원하는 방식으로 교체할 수 있습니다. Amazon ElastiCache 또는 Amazon DynamoDB는 오프로드된 상태에 적합한 대상입니다.

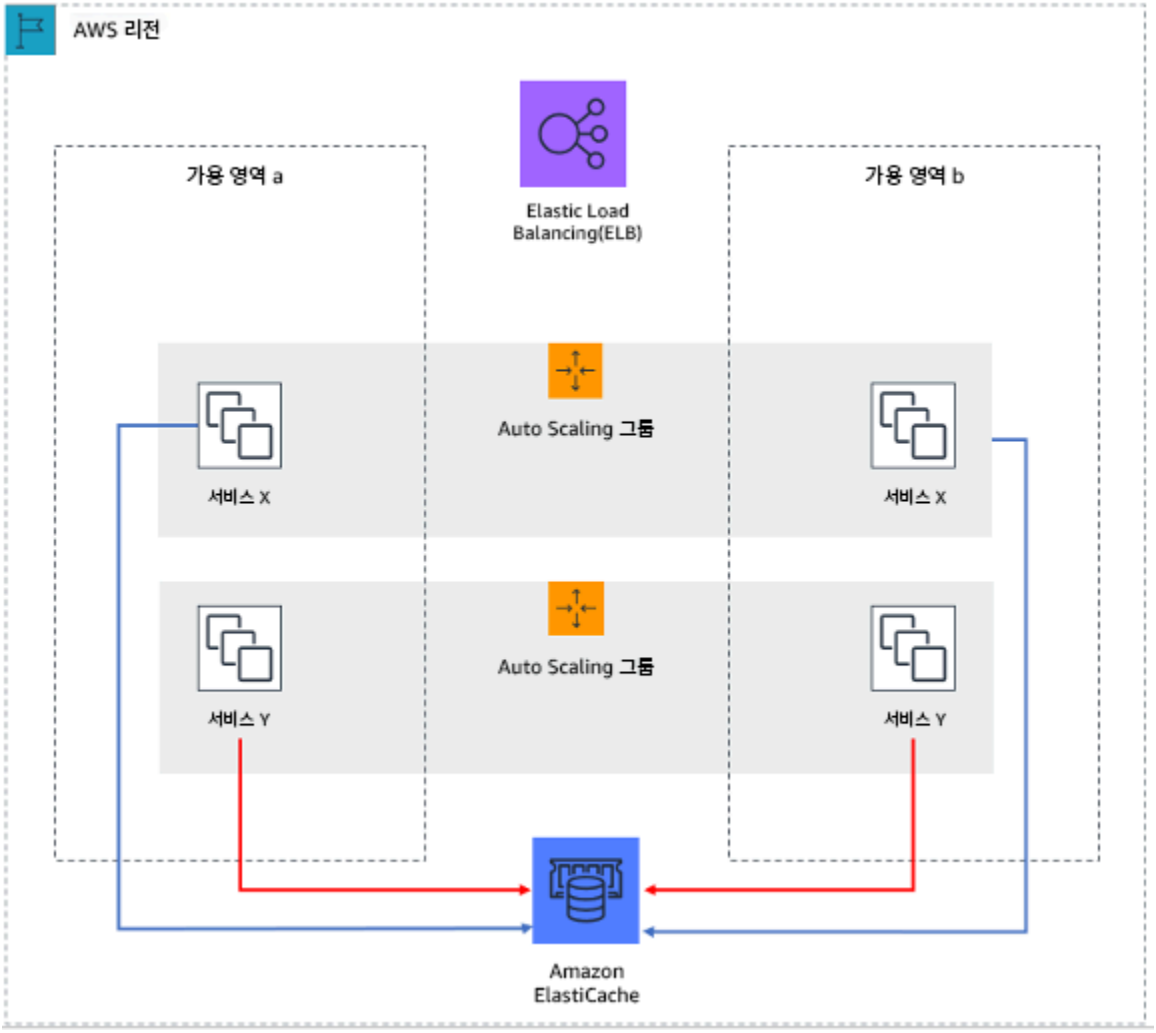


그림 7: 이 상태 비저장 웹 애플리케이션에서는 세션 상태가 Amazon ElastiCache로 오프로드됩니다.

사용자 또는 서비스는 애플리케이션과 상호 작용할 때 세션을 구성하는 일련의 상호 작용을 수행하는 경우가 많습니다. 세션은 애플리케이션을 사용하는 동안 요청 간에 유지되는 사용자의 고유한 데이터입니다. 상태 비저장 애플리케이션은 이전 상호 작용에 대한 지식을 필요로 하지 않으며 세션 정보를 저장하지 않는 애플리케이션입니다.

스테이스리스로 설계된 경우 AWS Lambda 또는 AWS Fargate와 같은 서버리스 컴퓨팅 서비스를 사용할 수 있습니다.

서버 대체 외에 상태 비저장 애플리케이션의 또 다른 이점은 사용 가능한 컴퓨팅 리소스(예: EC2 인스턴스 및 AWS Lambda 함수)로 모든 요청을 처리할 수 있기 때문에 수평적으로 확장할 수 있다는 것입니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 애플리케이션을 스테이스리스로 생성합니다. 무상태 애플리케이션은 수평 확장을 가능하게 하며, 개별 노드의 장애에 대한 내결함성이 있습니다.
  - 요청 파라미터에 실제로 저장될 수 있는 상태를 제거합니다.
  - 상태가 필요한지 여부를 조사한 후 상태 추적을 Amazon ElastiCache, Amazon RDS, Amazon DynamoDB 또는 서드 파티 분산 데이터 솔루션과 같은 복원성이 있는 다중 영역 캐시 데이터 스토어로 옮깁니다. 복원성이 있는 데이터 스토어로 옮길 수 없는 상태를 저장합니다.
    - 일부 데이터(쿠키 등)는 헤더 또는 쿼리 파라미터로 전달될 수 있습니다.
    - 리팩터링을 통해 요청에 빠르게 전달될 수 있는 상태를 제거합니다.
    - 일부 데이터는 요청별로 필요하지 않으며 요청에 따라 검색 시 검색될 수 있습니다.
    - 비동기식으로 검색할 수 있는 데이터를 제거합니다.
    - 필수 상태 요구 사항을 충족하는 데이터 스토어를 결정합니다.
    - 비관계형 데이터를 위한 NoSQL 데이터베이스를 고려합니다.

리소스

관련 문서:

- Amazon Builders' Library: 분산 시스템의 폴백 방지
- Amazon Builders' Library: 심각한 수준의 대기열 백로그 방지
- Amazon Builders' Library: 캐싱 관련 당면 과제 및 전략

REL05-BP07 비상 레버 구현

비상 레버는 워크로드의 가용성에 미치는 영향을 신속하게 완화할 수 있는 프로세스입니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 비상 레버를 구현합니다. 이 프로세스는 워크로드의 가용성에 미치는 영향을 신속하게 완화할 수 있는 프로세스입니다. 이 프로세스는 근본 원인이 없는 상태에서 작동할 수 있습니다. 이상적인 비상 레버는 완전히 결정적인 활성화 및 비활성화 기준을 제공하여 확인자에 대한 인지 부담을 0으로 줄여줍니다. 레버는 수동인 경우가 많지만 자동화할 수도 있습니다.
  - 예시 레버:

    - 모든 로봇 트래픽 차단
    - 동적 페이지 대신 정적 페이지 제공
    - 종속성으로의 호출 빈도 축소
    - 종속성으로부터의 호출 제한
- 비상 레버 구현 및 사용을 위한 팁
    - 레버가 활성화되면 더 적은 수의 작업을 수행
    - 단순하게 유지하고 바이모달 동작을 방지
    - 레버를 정기적으로 테스트
- 다음은 비상 레버가 아닌 작업의 예입니다.
    - 용량 추가
    - 서비스를 사용하는 클라이언트의 서비스 소유자를 호출하고 호출을 줄이도록 요청
    - 코드 변경 및 릴리스

변경 사항 관리

질문

- REL 6 워크로드 리소스는 어떻게 모니터링합니까?
- REL 7 수요 변경에 따라 조정되도록 워크로드를 설계하려면 어떻게 해야 합니까?
- REL 8 변경 사항은 어떻게 적용합니까?

REL 6 워크로드 리소스는 어떻게 모니터링합니까?

로그와 지표는 워크로드의 상태를 파악할 수 있는 유용한 도구입니다. 로그 및 지표를 모니터링하여 임계값을 초과하거나 중요한 이벤트가 발생하면 알림을 보내도록 워크로드를 구성할 수 있습니다. 모니터링을 수행하면 워크로드가 저성능 임계값을 초과하거나 장애가 발생할 때를 인식하고 이에 대응하여 자동으로 복구할 수 있습니다.

모범 사례

- REL06-BP01 워크로드의 모든 구성 요소 모니터링(생성)
- REL06-BP02 지표 정의 및 계산(집계)
- REL06-BP03 알림 전송(실시간 처리 및 경보)
- REL06-BP04 응답 자동화(실시간 처리 및 경보)

- REL06-BP05 분석
- REL06-BP06 정기적인 검토 시행
- REL06-BP07 시스템을 통한 요청의 엔드 투 엔드 추적 모니터링

REL06-BP01 워크로드의 모든 구성 요소 모니터링(생성)

Amazon CloudWatch 또는 서드파티 도구를 사용하여 워크로드의 구성 요소를 모니터링합니다. AWS Health 대시보드를 사용하여 AWS 서비스를 모니터링합니다.

프런트엔드, 비즈니스 로직 및 스토리지 계층을 포함하여 워크로드의 모든 구성 요소를 모니터링해야 합니다. 필요한 경우 주요 지표를 정의하고 로그에서 지표를 추출하는 방법을 설명하고 해당 경보 이벤트를 트리거하는 임계값을 설정합니다. 지표가 워크로드의 핵심 성과 지표(KPI)와 연관이 있도록 해야 하며 지표와 로그를 사용하여 서비스 성능 저하의 조기 지표를 파악합니다. 예를 들어, 분당 성공적으로 처리된 주문 수와 같은 비즈니스 성과와 관련된 지표는 CPU 사용량과 같은 기술적 지표보다 워크로드 문제를 더 빠르게 알려줍니다. AWS Health 대시보드를 사용하여 AWS 리소스의 기반이 되는 AWS 서비스의 성능 및 가용성에 대한 맞춤형 보기를 제공합니다.

클라우드에서 모니터링은 새로운 기회를 제공합니다. 대부분의 클라우드 공급업체는 사용자 지정 가능한 후크를 개발했으며 여러 계층의 워크로드를 모니터링하는 데 도움이 되는 인사이트를 제공할 수 있습니다. Amazon CloudWatch 등의 AWS 서비스는 통계 및 기계 학습 알고리즘을 적용하여 시스템 및 애플리케이션의 지표를 지속적으로 분석하고, 일반적인 기준을 결정하며, 사용자의 개입을 최소화하면서 이상 현상을 알립니다. 이상 탐지 알고리즘은 지표의 계절성과 추세 변화를 설명합니다.

AWS는 사용할 수 있는 모니터링 및 로그 정보를 풍부하게 제공하며 사용자는 워크로드별 지표를 정의하고, 수요가 있는 프로세스를 변경하고, ML 전문성과 상관없이 기계 학습 기법을 도입하는 데 이 정보를 사용할 수 있습니다.

또한 모든 외부 엔드포인트를 모니터링하여 기본 구현과 독립되어 있는지 확인합니다. 이 능동 모니터링은 사용자 Canary라고도 하는가상 트랜잭션에도 수행할 수 있지만 카나리 배포와 혼동해서는 안 됩니다. 가상 트랜잭션은 워크로드의 클라이언트가 수행하는 일반적인 다수의 태스크 매칭 작업을 주기적으로 실행합니다. 이 태스크의 기간은 짧아야 하며 테스트 중에 워크로드에 과부하가 발생하지 않아야 합니다. Amazon CloudWatch Synthetics를 사용하면 엔드포인트 및 API 모니터링을 위한 가상 Canary를 생성 할 수 있습니다. 가상 Canary 클라이언트 노드를 AWS X-Ray 콘솔과 함께 사용하여 선택한 기간에 오류, 장애 또는 조절 속도 문제를 경험하는 가상 Canary를 식별할 수도 있습니다.

원하는 결과:

워크로드의 모든 구성 요소로부터 핵심적인 지표를 수집하고 사용하여 워크로드 안정성과 최적의 사용자 경험을 보장합니다. 워크로드가 비즈니스 성과를 달성하지 못하고 있음을 탐지하면 재해 상황임을 빠르게 선언하고 인시던트에서 복구할 수 있습니다.

일반적인 안티 패턴:

- 워크로드에 대한 외부 인터페이스만 모니터링
- 워크로드별 지표를 생성하지 않으며 워크로드에서 사용하는 AWS 서비스에서 제공되는 지표에만 의존함
- 워크로드에서 기술적인 지표만 사용하며 워크로드가 기여하는 비기술적 KPI와 관련한 지표는 모니터링하지 않음
- 프로덕션 트래픽 및 단순한 상태 확인에 의존하여 워크로드 상태를 모니터링하고 평가함

이 모범 사례 정립의 이점: 워크로드의 모든 티어에서 모니터링할 경우 워크로드를 구성하는 구성 요소의 문제를 보다 신속하게 예측하고 해결할 수 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

1. 가능한 경우 로깅을 활성화합니다. 모니터링 데이터는 워크로드의 모든 구성 요소로부터 수집해야 합니다. S3 Access Logs 등의 추가 로깅을 활성화하고 워크로드가 워크로드별 데이터를 로깅하도록 합니다. Amazon ECS, Amazon EKS, Amazon EC2, Elastic Load Balancing, AWS Auto Scaling, Amazon EMR 등의 서비스로부터 CPU, 네트워크 I/O, 디스크 I/O 평균 지표를 수집합니다. 참조 CloudWatch 지표를 게시하는 AWS 서비스 에서 CloudWatch에 지표를 게시하는 AWS 서비스의 목록을 참조합니다.
2. 모든 기본 지표를 검토하고 데이터 수집에 간극이 있는지 확인합니다. 모든 서비스에서는 기본 지표를 생성합니다. 기본 지표를 수집하면 워크로드 구성 요소 간의 종속성을 이해하고 구성 요소 안정성과 성능이 워크로드에 어떤 영향을 미치는지 파악할 수 있습니다. 직접 지표를 생성하고 지표를 CloudWatch에 게시할 수 있습니다. AWS CLI 또는 API를 사용하면 됩니다. 이
3. 모든 지표를 평가하여 워크로드의 각 AWS 서비스에 어떤 지표를 알릴지 결정합니다. 워크로드 안정성에 큰 영향을 미치는 지표의 하위 집합을 선택할 수도 있습니다. 핵심 지표와 임계값에 집중하면 알림 의 수를 정리하고 허위 양성을 최소화할 수 있습니다.
4. 알림과 알림이 트리거된 후 워크로드의 복구 프로세스를 정의합니다. 알림을 정의하면 인시던트로부터 복구하는 데 필요한 단계를 빠르게 알리고, 에스컬레이션하고, 단계를 따라 사전에 정해진 Recovery Time Objective(RTO)를 달성할 수 있습니다. 전용 인프라에서 Amazon CloudWatch 경보를 사용하여 자동화된 워크플로를 호출하고 정의된 임계값에 따라 복구 절차를 시작할 수 있습니다.

5. 워크로드 상태에 대한 관련 데이터를 수집하기 위해 가상 트랜잭션을 사용하는 방법을 알아보세요. 가상 모니터링은 고객과 같은 경로를 따르고 같은 작업을 수행하므로 워크로드에 고객 트래픽이 없더라도 고객 경험을 지속적으로 확인할 수 있습니다. 이렇게 가상 트랜잭션을사용함으로써 고객보다 먼저 문제를 발견할 수 있습니다.

리소스

관련 모범 사례:

- REL11-BP03 모든 계층에서 복구 자동화

관련 문서:

- AWS Health 대시보드 시작하기 - 계정 상태
- CloudWatch 지표를 게시하는 AWS 서비스
- Network Load Balancer에 대한 액세스 로그
- Application Load Balancer에 대한 액세스 로그
- AWS Lambda의 Amazon CloudWatch Logs 액세스
- Amazon S3 서버 액세스 로깅
- Classic Load Balancer에 대한 액세스 로그 활성화(Classic Load Balancer에 대한 액세스 로그 활성화)
- Amazon S3로 로그 데이터 내보내기
- Amazon EC2 인스턴스에 CloudWatch 에이전트 설치
- 사용자 지정 지표 게시
- Amazon CloudWatch 대시보드 사용
- Amazon CloudWatch 지표 사용
- Canary 사용(Amazon CloudWatch Synthetics)
- Amazon CloudWatch Logs란 무엇입니까?

사용 설명서:

- 추적 생성
- Amazon EC2 Linux 인스턴스의 메모리 및 디스크 지표 모니터링
- 컨테이너 인스턴스와 CloudWatch Logs 사용

- VPC 흐름 로그
- Amazon DevOps Guru란 무엇입니까?
- AWS X-Ray란 무엇입니까?

관련 블로그:

- Amazon CloudWatch Synthetics 및 AWS X-Ray를 사용한 디버깅

관련 예시 및 워크숍:

- AWS Well-Architected 실습: 운영 우수성 - 종속성 모니터링
- Amazon Builders' Library: 운영 가시성을 위한 분산 시스템 계측
- 관찰 가능성 워크숍

REL06-BP02 지표 정의 및 계산(집계)

로그 데이터를 저장하고 필요한 경우 필터를 적용하여 특정 로그 이벤트 수 또는 로그 이벤트 타임스탬프에서 계산된 지연 시간과 같은 지표를 계산합니다.

Amazon CloudWatch 및 Amazon S3는 기본 집계 및 스토리지 계층으로 사용됩니다. AWS Auto Scaling 및 Elastic Load Balancing와 같은 일부 서비스에서는 클러스터나 인스턴스 전반에 걸쳐 CPU 로드 또는 평균 요청 지연 시간 관련 기본 지표가 기본적으로 제공됩니다. VPC Flow Logs 및 AWS CloudTrail과 같은 스트리밍 서비스의 경우에는 이벤트 데이터가 CloudWatch Logs로 전달되며, 이벤트 데이터에서 지표를 추출하려면 지표 필터를 정의하고 적용해야 합니다. 이렇게 하면 시계열 데이터가 나오며 알림을 트리거하도록 정의한 CloudWatch 경보에 대한 입력으로 이 데이터를 사용할 수 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 지표를 정의 및 계산(집계)합니다. 로그 데이터를 저장하고 필요한 경우 필터를 적용하여 특정 로그 이벤트 수 또는 로그 이벤트 타임스탬프에서 계산된 지연 시간과 같은 지표를 계산합니다.
  - 지표 필터는 로그 데이터가 CloudWatch Logs에 전송될 때 찾아야 하는 용어와 패턴을 정의합니다. CloudWatch Logs는 이 지표 필터를 사용하여 로그 데이터를 숫자 형식의 CloudWatch 지표로 변환하여 사용자가 그래프를 작성하거나 경보를 설정할 수 있도록 합니다.
    - 로그 데이터 검색 및 필터링

- 신뢰할 수 있는 서드 파티 도구를 사용하여 로그를 집계합니다.
  - 해당 서드 파티의 지침을 따릅니다. 대부분의 서드 파티 제품은 CloudWatch 및 Amazon S3와 통합됩니다.
- 일부 AWS 서비스는 로그를 Amazon S3에 직접 게시할 수 있습니다. Amazon S3에 저장하는 것이 로그의 주요 요구 사항인 경우, 추가 인프라를 설정하지 않고도 로그를 생성하는 서비스가 로그를 Amazon S3로 직접 전송하도록 할 수 있습니다.
  - Amazon S3로 로그를 직접 전송

리소스

관련 문서:

- Amazon CloudWatch Logs Insights 샘플 쿼리
- Amazon CloudWatch Synthetics 및 AWS X-Ray를 사용한 디버깅
- One Observability Workshop
- 로그 데이터 검색 및 필터링
- Amazon S3로 로그를 직접 전송
- Amazon Builders' Library: 운영 가시성을 위한 분산 시스템 계측

REL06-BP03 알림 전송(실시간 처리 및 경보)

중요한 이벤트가 발생할 때 알아야 하는 조직에 알림이 전송됩니다.

Amazon Simple Notification Service(Amazon SNS) 주제로 알림을 전송한 다음 원하는 수의 구독자에게 푸시할 수 있습니다. 예를 들어 Amazon SNS는 기술 직원이 응답할 수 있도록 특정 이메일 별칭으로 알림을 전달할 수 있습니다.

일반적인 안티 패턴:

- 임계값을 너무 낮게 구성하여 알림이 너무 많이 전송되도록 함
- 향후 조사할 수 있도록 경보를 보관하지 않음

이 모범 사례 정립의 이점: 이벤트에 대한 알림(응답할 수 있고 자동으로 해결할 수 있는 알림 포함)을 통해 이벤트 기록을 확보할 수 있으며 향후에 다른 방식으로 이벤트를 처리할 수 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 실시간 처리 및 경보 설정을 수행합니다. 중요한 이벤트가 발생할 때 알아야 하는 조직에 알림이 전송됩니다.
  - Amazon CloudWatch 대시보드는 CloudWatch 콘솔의 맞춤형 홈페이지로, 다른 리전에 분산되어 있는 리소스까지 포함하여 모든 리소스를 단일 보기에서 모니터링하는 데 이용할 수 있습니다.
    - Amazon CloudWatch 대시보드 사용
  - 지표가 한도를 초과하는 경우 경보를 생성합니다.
    - Amazon CloudWatch 경보 사용

리소스

관련 문서:

- One Observability Workshop
- Amazon Builders' Library: 운영 가시성을 위한 분산 시스템 계측
- Amazon CloudWatch 경보 사용
- Amazon CloudWatch 대시보드 사용
- Amazon CloudWatch 지표 사용

REL06-BP04 응답 자동화(실시간 처리 및 경보)

: 이벤트가 감지되면 자동화를 사용하여 실패한 구성 요소를 대체하는 등의 조치를 취합니다.

알림을 통해 AWS Auto Scaling 이벤트를 트리거할 수 있으며, 그러면 클러스터가 수요 변경에 대응할 수 있습니다. 서드 파티 티켓 시스템용 통합 지점으로 사용 가능한 Amazon Simple Queue Service(Amazon SQS)로 알림을 전송할 수도 있습니다. AWS Lambda에서도 알림을 구독하여 변경에 동적으로 대응하는 비동기 서버리스 모델을 사용자에게 제공할 수 있습니다. AWS Config는 AWS 리소스 구성을 지속적으로 모니터링하고 기록하며 AWS Systems Manager Automation 을 트리거하여 문제를 해결할 수 있습니다.

Amazon DevOps Guru는 애플리케이션 리소스가 비정상적으로 작동하는지를 자동으로 모니터링하고 표적화된 권장 사항을 제공하여 문제 파악 및 해결 시간을 단축합니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- Amazon DevOps Guru를 사용하여 자동화된 작업을 수행합니다. Amazon DevOps Guru는 애플리케이션 리소스가 비정상적으로 작동하는지를 자동으로 모니터링하고 표적화된 권장 사항을 제공하여 문제 파악 및 해결 시간을 단축합니다.
  - Amazon DevOps Guru란 무엇입니까?
- AWS Systems Manager를 사용하여 자동화된 작업을 수행합니다. AWS Config는 AWS 리소스 구성을 지속적으로 모니터링하고 기록하며 AWS Systems Manager Automation을 트리거하여 문제를 해결할 수 있습니다.
  - AWS Systems Manager Automation
    - Systems Manager Automation 문서를 생성하고 사용합니다. 이는 자동화 프로세스를 실행할 때 Systems Manager가 관리형 인스턴스 및 기타 AWS 리소스에서 수행하는 작업을 정의합니다.
    - 자동화 문서 작업(플레이북)
- Amazon CloudWatch가 경보 상태 변화 이벤트를 Amazon EventBridge에 전송합니다. 응답을 자동화하는 EventBridge 규칙을 생성합니다.
  - AWS 리소스의 이벤트에서 트리거되는 EventBridge 규칙 생성
- 응답 자동화를 위한 계획을 수립하고 실행합니다.
  - 모든 알림 응답 절차의 인벤토리를 작성합니다. 작업 순위를 정하기 전에 알림 응답을 계획해야 합니다.
  - 수행해야 할 특정 작업이 있는 모든 작업의 인벤토리를 작성합니다. 이러한 작업은 대부분 런북에 문서화됩니다. 예기치 않은 이벤트의 알림에 대한 플레이북도 있어야 합니다.
  - 런북 및 플레이북을 조사하여 자동화 가능한 작업을 모두 파악합니다. 일반적으로, 정의할 수 있는 작업은 대부분 자동화할 수 있습니다.
  - 오류가 발생하기 쉽거나 시간이 많이 걸리는 활동을 최우선으로 합니다. 오류의 원인을 제거하고 해결 시간을 단축하는 데 따른 효과가 가장 크기 때문입니다.
  - 자동화를 수행하기 위한 계획을 수립합니다. 자동화 및 업데이트를 위한 계획을 활성 상태로 유지합니다.
  - 수작업 요구 사항을 검토하여 자동화할 여지가 없는지 확인합니다. 수작업 프로세스를 검토하여 자동화 기회를 확인합니다.

리소스

관련 문서:

- AWS Systems Manager Automation
- AWS 리소스의 이벤트에서 트리거되는 EventBridge 규칙 생성
- One Observability Workshop
- Amazon Builders' Library: 운영 가시성을 위한 분산 시스템 계측
- Amazon DevOps Guru란 무엇입니까?
- 자동화 문서 작업(플레이북)

REL06-BP05 분석

로그 파일 및 지표 기록을 수집하고 이를 분석하여 더 광범위한 추세 및 워크로드 인사이트를 확보합니다.

Amazon CloudWatch Logs Insights는 로그 데이터를 분석하는 데 사용할 수 있는 단순하지만 강력한 쿼리 언어 를 지원합니다. Amazon CloudWatch Logs 또한 구독을 지원하므로 데이터를 Amazon S3로 원활하게 보내 여기서 데이터를 사용하거나 Amazon Athena로 보내 데이터를 쿼리할 수 있습니다. 다양한 형식의 쿼리도 지원됩니다. 참조 지원되는 SerDes 및 데이터 형식 (Amazon Athena 사용 설명서에 있음)에서 자세한 내용을 참조하세요. 방대한 로그 파일 세트를 분석하려면 Amazon EMR 클러스터를 실행하여 페타바이트 규모의 분석을 실행할 수 있습니다.

AWS 파트너와 서드파티에서 제공하는 다양한 도구를 집계, 처리, 저장 및 분석에 사용할 수 있습니다. 이러한 도구에는 New Relic, Splunk, Loggly, Logstash, CloudHealth 및 Nagios가 포함됩니다. 그러나 시스템과 애플리케이션 외부에서 생성되는 로그는 각 클라우드 공급자별로 다르며 각 서비스별로 다른 경우도 많습니다.

모니터링 프로세스에서 간과되는 경우가 많은 작업 중 하나로 데이터 관리를 들 수 있습니다. 데이터 모니터링을 위한 보존 요구 사항을 확인한 후 그에 따라 수명 주기 정책을 적용해야 합니다. Amazon S3는 S3 버킷 수준에서 수명 주기 관리를 지원합니다. 버킷의 각 경로에 이 수명 주기 관리 기능을 각기 다르게 적용할 수 있습니다. 수명 주기 종료가 가까워지면 장기 저장을 위해 데이터를 Amazon S3 Glacier로 전환한 다음 보존 기간이 종료되면 데이터를 만료 처리할 수 있습니다. S3 Intelligent-Tiering 스토리지 클래스는 성능 영향이나 운영 오버헤드 없이 데이터를 가장 비용 효율적인 티어로 자동으로 이동하여 비용을 최적화하도록 설계되었습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- CloudWatch Logs Insights를 사용하면 Amazon CloudWatch Logs의 로그 데이터를 대화식으로 검색하고 분석할 수 있습니다.

  - CloudWatch Logs Insights를 사용한 로그 데이터 분석
  - Amazon CloudWatch Logs Insights 샘플 쿼리
- Amazon CloudWatch Logs를 사용하여 사용할 수 있는 Amazon S3에 로그를 전송하거나 Amazon Athena에 전송하여 데이터를 쿼리합니다.
  - Athena를 사용하여 Amazon S3 서버 액세스 로그를 분석하려면 어떻게 해야 합니까?
    - 서버 액세스 로그 버킷에 대한 S3 수명 주기 정책을 생성합니다. 로그 파일을 주기적으로 제거하도록 수명 주기 정책을 구성합니다. 이러한 정책을 구성하면 Athena에서 각 쿼리에 대해 분석되는 데이터의 양이 줄어듭니다.
      - S3 버킷에 대한 수명 주기 정책을 생성하려면 어떻게 해야 하나요?

리소스

관련 문서:

- Amazon CloudWatch Logs Insights 샘플 쿼리
- CloudWatch Logs Insights를 사용한 로그 데이터 분석
- Amazon CloudWatch Synthetics 및 AWS X-Ray를 사용한 디버깅
- S3 버킷에 대한 수명 주기 정책을 생성하려면 어떻게 해야 하나요?
- Athena를 사용하여 Amazon S3 서버 액세스 로그를 분석하려면 어떻게 해야 합니까?
- One Observability Workshop
- Amazon Builders' Library: 운영 가시성을 위한 분산 시스템 계측

REL06-BP06 정기적인 검토 시행

워크로드 모니터링이 구현되는 방식을 자주 검토하고 중요한 이벤트 및 변경 사항에 따라 업데이트합니다.

효과적인 모니터링의 기반은 주요 비즈니스 지표입니다. 비즈니스 우선 순위가 변경됨에 따라 이러한 지표가 워크로드에 반영되는지 확인하십시오.

모니터링을 감사하면 애플리케이션이 가용성 목표를 달성하는 시기를 확인하는 데 도움이 됩니다. 근본 원인 분석을 수행하려면 장애 발생 시에 수행된 작업을 검색하는 기능이 필요합니다. AWS는 인시던트 중에 서비스의 상태를 추적할 수 있는 서비스를 제공합니다.

- Amazon CloudWatch Logs: 로그를 저장하고 해당 내용을 검사할 수 있는 서비스입니다.

- Amazon CloudWatch Logs Insights: 대량 로그를 몇 초 만에 분석할 수 있는 완전관리형 서비스입니다. 이 서비스는 빠른 대화형 쿼리 및 시각화를 제공합니다.
- AWS Config: 다양한 시점에서 사용된 AWS 인프라를 확인할 수 있습니다.
- AWS CloudTrail: 특정 시간에 호출된 AWS API 및 기준으로 사용된 원칙을 확인할 수 있는 서비스입니다.

AWS에서는 주간 회의를 개최하여 운영 성과를 검토하고 알게 된 내용을 팀 간에 공유합니다. AWS에는 많은 팀이 있기 때문에 검토할 워크로드를 무작위로 선택하는 The Wheel 을 만들었습니다. 운영 성능 검토 및 지식 공유를 위한 정기 케이던스를 설정하면 운영 팀의 성과를 개선하는 역량을 발전시킬 수 있습니다.

일반적인 안티 패턴:

- 기본 지표만 수집
- 모니터링 전략을 설정한 후 다시 검토하지 않음
- 주요 변경 사항이 배포될 때 모니터링에 대해 논의하지 않음

이 모범 사례 수립의 이점: 모니터링을 정기적으로 검토하면 예상된 문제가 실제로 발생할 때 알림에 대응하는 것이 아니라 잠재적인 문제를 미리 예측할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 워크로드에 대해 여러 대시보드를 생성합니다. 주요 비즈니스 지표는 물론, 다양한 사용량에서 예상되는 워크로드의 상태와 가장 관련성이 높은 것으로 확인된 기술 지표도 포함된 최상위 대시보드가 있어야 합니다. 또한 검사할 수 있는 다양한 애플리케이션 티어와 종속성에 대한 대시보드도 필요합니다.
  - Amazon CloudWatch 대시보드 사용
- 워크로드 대시보드에 대한 정기적인 검토 일정을 예약하고 검토를 수행합니다. 대시보드를 정기적으로 검사합니다. 검사하는 세부 수준을 나타내는 다양한 케이던스를 구성할 수 있습니다.
  - 지표에서 추세를 검사합니다. 지표 값을 과거 값과 비교하여 조사해야 할 문제가 있음을 시사하는 추세가 나타나는지 확인합니다. 이러한 예로는 지연 시간 증가, 주요 비즈니스 기능 감소, 장애 응답 증가 등이 있습니다.
  - 지표에서 특이값/이상 항목을 검사합니다. 평균 또는 중간값은 특이값 및 이상 항목을 감출 수 있습니다. 해당 기간 동안의 가장 높은 값과 가장 낮은 값을 살펴보고 극단적인 값의 원인을 조사합

니다. 이러한 원인을 계속 제거하면서 극단성을 낮추면 워크로드 성능의 일관성을 지속적으로 개선할 수 있습니다.

- 동작의 급격한 변화를 찾습니다. 지표의 수량 또는 방향이 갑자기 바뀔 경우, 애플리케이션이 변경되었거나 외부 요인이 발생한 것일 수 있습니다. 이 같은 외부 요인으로 인해 추적할 지표를 추가해야 할 수 있습니다.

리소스

관련 문서:

- Amazon CloudWatch Logs Insights 샘플 쿼리
- Amazon CloudWatch Synthetics 및 AWS X-Ray를 사용한 디버깅
- One Observability Workshop
- Amazon Builders' Library: 운영 가시성을 위한 분산 시스템 계측
- Amazon CloudWatch 대시보드 사용

REL06-BP07 시스템을 통한 요청의 엔드 투 엔드 추적 모니터링

제품 팀이 문제를 더 쉽게 분석 및 디버그하고 성능을 개선할 수 있도록 서비스 구성 요소를 통해 처리되는 요청을 추적합니다.

원하는 결과: 모든 구성 요소를 포괄적으로 추적할 수 있는 워크로드는 디버그하기 쉬우므로 근본 원인 찾기를 단순화하여 오류의 평균 해결 시간(MTTR) 및 지연 시간이 개선됩니다. 엔드 투 엔드 추적은 영향을 받는 구성 요소를 찾고 오류 또는 지연 시간의 근본 원인을 자세히 조사하는 데 걸리는 시간을 줄여줍니다.

일반적인 안티 패턴:

- 추적이 모든 구성 요소가 아니라 일부 구성 요소에서만 사용됩니다. 예를 들어 AWS Lambda를 추적하지 않으면 팀이 급증하는 워크로드에서 콜드 스타트로 인한 지연 시간을 명확하게 이해하지 못할 수 있습니다.
- Synthetic canary 또는 실제 사용자 모니터링(RUM)이 추적이 가능하도록 구성되지 않습니다. Canary 또는 RUM이 없으면 추적 분석에서 클라이언트 상호 작용 원격 측정이 생략되어 불완전한 성능 프로파일이 생성됩니다.
- 하이브리드 워크로드에 클라우드 네이티브 및 타사 추적 도구가 모두 포함되지만 단일 추적 솔루션을 선택하고 완전히 통합하는 단계는 아직 수행하지 않았습니다. 선택한 추적 솔루션에 따라 클라우

드 네이티브 추적 SDK를 사용하여 클라우드 네이티브가 아닌 구성 요소를 측정하거나 타사 도구를 클라우드 네이티브 추적 텔레메트리를 수집하도록 구성해야 합니다.

이 모범 사례 확립의 이점: 개발 팀은 문제에 대한 경고를 받으면 구성 요소별 로깅, 성능, 장애와의 상관 관계를 포함하여 시스템 구성 요소 상호 작용을 종합적으로 파악할 수 있습니다. 추적을 통해 근본 원인을 시각적으로 쉽게 식별할 수 있으므로 근본 원인을 조사하는 데 소요되는 시간이 줄어듭니다. 구성 요소 상호 작용을 자세히 이해하는 팀은 문제를 해결할 때 더 현명하고 빠른 의사 결정을 내릴 수 있습니다. 시스템 추적을 분석하면 재해 복구(DR) 장애 조치를 언제 실행할지, 자가 복구 전략을 가장 잘 구현할 수 있는 위치 등을 더 제대로 결정할 수 있어 궁극적으로 서비스에 대한 고객 만족도를 향상시킬 수 있습니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 보통

구현 가이드

분산된 애플리케이션을 운영하는 팀은 추적 도구를 사용하여 상관 관계 식별자를 설정하고 요청 추적을 수집하며 연결된 구성 요소의 서비스 맵을 구축할 수 있습니다. 서비스 클라이언트, 미들웨어 게이트웨이 및 이벤트 버스, 컴퓨팅 구성 요소, 키 값 저장소 및 데이터베이스가 포함된 스토리지 등 모든 애플리케이션 구성 요소가 요청 추적에 포함되어야 합니다. 서비스 수준 계약과 목표에 대해 시스템 성능을 정확하게 평가할 수 있도록 엔드 투 엔드 추적 구성에 Synthetic canary와 실제 사용자 모니터링을 포함하여 원격 클라이언트 상호 작용과 지연 시간을 측정합니다.

이때 AWS X-Ray 및 Amazon CloudWatch 애플리케이션 모니터링 계측 서비스를 사용하여 요청이 애플리케이션을 통과하는 동안 요청에 대한 전체 보기를 제공할 수 있습니다. X-Ray를 통해 애플리케이션 텔레메트리를 수집하여 페이로드, 함수, 추적, 서비스, API에서 이를 시각화하고 필터링할 수 있으며 노코드 또는 로우 코드 시스템 구성 요소에 대해 활성화할 수 있습니다. CloudWatch 애플리케이션 모니터링에는 추적을 지표, 로그, 경보와 통합하는 ServiceLens가 포함됩니다. 또한 CloudWatch 애플리케이션 모니터링에는 엔드포인트와 API를 모니터링하기 위한 Synthetics와 웹 애플리케이션 클라이언트를 측정하기 위한 실제 사용자 모니터링도 포함됩니다.

구현 단계

- AWS X-Ray를 모든 지원되는 네이티브 서비스(예: Amazon S3, AWS Lambda, Amazon API Gateway)에서 사용합니다. 이러한 AWS 서비스에서는 코드형 인프라, AWS SDK 또는 AWS Management Console을 사용하여 구성 토글을 통해 X-Ray가 활성화됩니다.
- 애플리케이션 계측 AWS Distro for Open Telemetry 및 X-Ray 또는 타사 수집 에이전트.

- 프로그래밍 언어별 구현은 AWS X-Ray 개발자 가이드를 참조하십시오. 이러한 설명서 섹션에서는 애플리케이션 프로그래밍 언어와 관련된 HTTP 요청, SQL 쿼리 및 기타 프로세스의 계측 방법을 자세히 설명합니다.
- X-Ray 추적을 Amazon CloudWatch Synthetic Canary 및 Amazon CloudWatch RUM 에 사용하여 최종 사용자 클라이언트에서 다운스트림 AWS 인프라를 통한 요청 경로를 분석합니다.
- 팀이 문제에 대해 빠르게 경고를 받은 후 ServiceLens를 사용하여 추적 및 서비스 맵을 심층적으로 분석할 수 있도록 리소스 상태와 canary 텔레메트리를 기반으로 CloudWatch 추적 및 경보를 구성합니다.
- 기본 추적 솔루션에 타사 도구를 사용하는 경우 Datadog, New Relic또는 Dynatrace 와 같은 타사 추적 도구에 대한 X-Ray 통합을 활성화합니다.

리소스

관련 모범 사례:

- REL06-BP01 워크로드의 모든 구성 요소 모니터링(생성)
- REL11-BP01 워크로드의 모든 구성 요소를 모니터링하여 장애 감지

관련 문서:

- AWS X-Ray란 무엇인가요?
- Amazon CloudWatch: 애플리케이션 모니터링
- Amazon CloudWatch Synthetics 및 AWS X-Ray를 사용한 디버깅
- Amazon Builders' Library: 운영 가시성을 위한 분산 시스템 계측
- AWS X-Ray를 다른 AWS 서비스와 통합
- AWS Distro for OpenTelemetry 및 AWS X-Ray
- Amazon CloudWatch: 합성 모니터링 사용
- Amazon CloudWatch: CloudWatch RUM 사용
- Amazon CloudWatch synthetics canary 및 Amazon CloudWatch 경보 설정
- 가용성과 그 이후: AWS에 배포된 시스템의 복원력 파악 및 개선

관련 예시:

- One Observability Workshop

관련 동영상:

- AWS re:Invent 2022 - How to monitor applications across multiple accounts(여러 계정의 애플리케이션을 모니터링하는 방법)
- AWS 애플리케이션을 모니터링하는 방법

관련 도구:

- AWS X-Ray
- Amazon CloudWatch
- Amazon Route 53

REL 7 수요 변경에 따라 조정되도록 워크로드를 설계하려면 어떻게 해야 합니까?

확장 가능한 워크로드는 리소스를 자동으로 추가하거나 제거하여 특정 시기의 수요에 리소스 공급을 맞출 수 있는 탄력성을 제공합니다.

모범 사례

- REL07-BP01 리소스를 확보하거나 조정할 때 자동화 사용
- REL07-BP02 워크로드 장애 감지 시 리소스 확보
- REL07-BP03 워크로드에 더 많은 리소스가 필요한 것으로 감지되면 리소스 확보
- REL07-BP04 워크로드 로드 테스트

REL07-BP01 리소스를 확보하거나 조정할 때 자동화 사용

손상된 리소스를 교체하거나 워크로드 크기를 조정할 때 Amazon S3 및 AWS Auto Scaling과 같은 관리형 AWS 서비스를 사용하여 프로세스를 자동화합니다. 서드 파티 도구 및 AWS SDK를 사용하여 크기를 자동 조정할 수도 있습니다.

관리형 AWS 서비스에는 Amazon S3, Amazon CloudFront, AWS Auto Scaling, AWS Lambda, Amazon DynamoDB, AWS Fargate, Amazon Route 53가 있습니다.

AWS Auto Scaling을 사용하면 손상된 인스턴스를 감지하고 교체할 수 있습니다. 또한 Amazon EC2 인스턴스 및 스팟 플릿, Amazon ECS 태스크, Amazon DynamoDB 테이블 및 인덱스와 Amazon Aurora 복제본에 대한 조정 계획을 수립할 수도 있습니다.

EC2 인스턴스의 크기를 조정할 때는 여러 가용 영역(3개 이상 권장)을 사용하고 용량을 추가하거나 제거하여 이러한 가용 영역 간의 균형을 유지해야 합니다. ECS 태스크 또는 Kubernetes 포드(Amazon Elastic Kubernetes Service를 사용하는 경우) 역시 여러 개의 가용 영역에 분산되어야 합니다.

AWS Lambda를 사용하는 경우에는 인스턴스가 자동으로 조정됩니다. 함수에 대한 이벤트 알림이 수신될 때마다 AWS Lambda가 컴퓨팅 플릿 내에서 여유 용량을 신속하게 찾고 할당된 동시성까지 코드를 실행합니다. 사용자는 필요한 동시성이 특정 Lambda와 Service Quotas에 구성되어 있는지 확인해야 합니다.

Amazon S3는 높은 요청 속도를 처리하도록 자동으로 확장됩니다. 예를 들어 애플리케이션은 버킷에서 접두사마다 초당 3,500개 이상의 PUT/COPY/POST/DELETE 및 5,500개의 GET/HEAD 요청을 전송할 수 있습니다. 버킷의 접두사 수에는 제한이 없습니다. 읽기를 병렬화하여 읽기 또는 쓰기 성능을 높일 수 있습니다. 예를 들어 Amazon S3 버킷에서 10개의 접두사를 생성하여 읽기를 병렬화하는 경우 읽기 성능을 초당 55,000개의 읽기 요청으로 확장할 수 있습니다.

Amazon CloudFront 또는 신뢰할 수 있는 콘텐츠 전송 네트워크(CDN)를 구성하고 사용합니다. CDN은 더 빠른 최종 사용자 응답 시간을 제공할 수 있으며 콘텐츠 요청을 캐시에서 처리하므로 워크로드의 크기를 확대할 필요가 줄어듭니다.

일반적인 안티 패턴:

- 자동 복구를 위해 Auto Scaling 그룹을 구현하지만 탄력성은 구현하지 않음
- Auto Scaling을 사용하여 트래픽의 대규모 증가에 대응
- 고도의 상태 저장 애플리케이션을 배포하여 탄력성 옵션을 없앰

이 모범 사례 수립의 이점: 자동화는 리소스를 배포하고 폐기할 때 수작업으로 인한 오류가 발생할 가능성을 없앱니다. 자동화는 배포 또는 폐기 요구 사항에 대한 느린 대응으로 인해 비용이 초과하거나 서비스가 거부될 위험성을 없앱니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- AWS Auto Scaling을 구성하고 사용합니다. 애플리케이션을 모니터링하고 용량을 자동으로 조정하여 최저 비용으로 안정적이고 예측 가능한 성능을 유지할 수 있습니다. AWS Auto Scaling을 사용하면 여러 서비스에 걸쳐 여러 리소스에 대한 애플리케이션 크기 조정을 설정할 수 있습니다.
  - AWS Auto Scaling이란 무엇입니까?

    - Amazon EC2 인스턴스, 스팟 플릿, Amazon ECS 태스크, Amazon DynamoDB 테이블 및 인덱스, Amazon Aurora 복제본, AWS Marketplace 어플라이언스에 적절하게 Auto Scaling을 구성합니다.
        - DynamoDB Auto Scaling을 통한 처리량 용량 자동 관리
            - 서비스 API 작업을 사용하여 경보, 확장 정책, 가동 준비 시간 및 중단 시간을 지정합니다.
- Elastic Load Balancing을 사용합니다. 로드 밸런서는 경로별 또는 네트워크 연결별로 로드를 분산할 수 있습니다.
    - Elastic Load Balancing이란 무엇입니까?
        - Application Load Balancers는 경로별로 로드를 배포할 수 있습니다.
            - Application Load Balancer란 무엇입니까?
                - 도메인 이름의 하위 경로를 기준으로 다른 워크로드에 트래픽을 분산하도록 Application Load Balancer를 구성합니다.
                - Application Load Balancers를 사용해 AWS Auto Scaling과 통합하는 방식으로 수요를 관리해 로드를 분산할 수 있습니다.
                    - 오토 스케일링 그룹과 함께 로드 밸런서 사용
        - Network Load Balancer는 연결별로 로드를 분산합니다.
            - Network Load Balancer란 무엇입니까?
                - TCP를 사용하여 다른 워크로드로 트래픽을 분산하도록 Network Load Balancer를 구성하거나 워크로드가 일정한 IP 주소 집합을 갖도록 합니다.
                - Network Load Balancer를 사용해 AWS Auto Scaling과 통합하는 방식으로 수요를 관리해 로드를 분산할 수 있습니다.
- 가용성 높은 DNS 공급자를 사용합니다. DNS 이름을 사용하면 사용자가 IP 주소 대신 이름을 입력하여 워크로드에 액세스하고 이 정보를 정의된 범위, 즉 일반적으로 워크로드 사용자의 경우 전역적으로 배포할 수 있습니다.
    - Amazon Route 53 또는 신뢰할 수 있는 DNS 제공업체를 사용합니다.
        - Amazon Route 53란 무엇입니까?
    - Route 53를 사용하여 CloudFront 배포 및 로드 밸런서를 관리합니다.
        - 관리할 도메인 및 하위 도메인을 결정하십시오.
        - ALIAS 또는 CNAME 레코드를 사용하여 적절한 레코드 세트를 생성합니다.
            - 레코드 작업

- AWS 글로벌 네트워크를 사용하여 사용자로부터 애플리케이션으로의 경로를 최적화합니다. AWS Global Accelerator는 30초 이내에 애플리케이션 엔드포인트의 상태를 지속적으로 모니터링하고 트래픽을 정상 엔드포인트로 리디렉션합니다.
  - AWSGlobal Accelerator는 로컬 또는 글로벌 사용자에 대해 애플리케이션의 가용성과 성능을 개선하는 서비스입니다. Application Load Balancers, Network Load Balancer 또는 Amazon EC2 인스턴스 등, 하나 또는 여러 AWS 리전에서 애플리케이션 엔드포인트에 고정된 진입점 역할을 하는 고정 IP 주소를 제공합니다.
    - AWS Global Accelerator란 무엇입니까?
- Amazon CloudFront 또는 신뢰할 수 있는 콘텐츠 전송 네트워크(CDN)를 구성하고 사용합니다. 콘텐츠 전송 네트워크를 사용하면 최종 사용자 입장에서는 응답 시간을 단축할 수 있으며 워크로드를 불필요하게 확장할 수 있는 콘텐츠 요청을 처리할 수 있습니다.
  - Amazon CloudFront란 무엇입니까?
    - 워크로드에 Amazon CloudFront 배포를 구성하거나 서드 파티 CDN을 사용합니다.
      - CloudFront의 IP 범위를 엔드포인트 보안 그룹 또는 액세스 정책에 사용함으로써 워크로드에 대한 액세스를 제한하여 CloudFront에서만 액세스할 수 있도록 할 수 있습니다.

리소스

관련 문서:

- APN 파트너: 자동화된 컴퓨팅 솔루션의 생성을 지원할 수 있는 파트너
- AWS Auto Scaling: 조정 계획 작동 방식
- AWS Marketplace: 오토 스케일링과 함께 사용할 수 있는 제품
- DynamoDB Auto Scaling을 통한 처리량 자동 관리
- 오토 스케일링 그룹과 함께 로드 밸런서 사용
- AWS Global Accelerator란 무엇입니까?
- Amazon EC2 Auto Scaling란 무엇입니까?
- AWS Auto Scaling이란 무엇입니까?
- Amazon CloudFront란 무엇입니까?
- Amazon Route 53란 무엇입니까?
- Elastic Load Balancing이란 무엇입니까?
- Network Load Balancer란 무엇입니까?
- Application Load Balancer란 무엇입니까?

- 레코드 작업

REL07-BP02 워크로드 장애 감지 시 리소스 확보

가용성이 영향을 받는 경우 필요에 따라 리소스를 사후에 확장하여 워크로드 가용성을 복원합니다.

먼저 상태 확인과 이러한 확인에 대한 기준을 구성하여 리소스 부족으로 인해 가용성이 영향을 받는 시기를 나타내야 합니다. 그런 다음 적절한 담당자에게 수동으로 리소스를 조정하도록 알리거나 자동화를 트리거하여 자동으로 리소스를 조정합니다.

워크로드에 맞게 적절하게 수동으로 규모를 조정할 수 있습니다. 예를 들어 AWS Management Console 또는 AWS CLI를 통해 Auto Scaling 그룹의 EC2 인스턴스 수를 변경하거나 DynamoDB 테이블의 처리량을 수정할 수 있습니다. 하지만 가능한 경우에는 항상 자동화를 사용해야 합니다(리소스를 확보하거나 조정할 때 자동화 사용)는 SAP NetWeaver Guide Finder 및 SAP NetWeaver Security Guide를 참조하세요.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 워크로드 장애 감지 시 리소스를 확보합니다. 가용성이 영향을 받는 경우 필요에 따라 리소스를 사후에 확장하여 워크로드 가용성을 복원합니다.
  - AWS Auto Scaling의 핵심 구성 요소인 크기 조정 계획을 사용하여 리소스 크기 조정을 위한 지침 집합을 구성합니다. AWS CloudFormation을 사용하거나 AWS 리소스에 태그를 추가하는 경우 애플리케이션마다 서로 다른 리소스 집합에 대해 크기 조정 계획을 설정할 수 있습니다. AWS Auto Scaling은 각 리소스에 맞춤화된 크기 조정 전략에 대한 권장 사항을 제공합니다. 크기 조정 계획을 생성하면 AWS Auto Scaling이 동적 조정과 예측 조정 방식을 결합하여 조정 전략을 지원합니다.
    - AWS Auto Scaling: 조정 계획 작동 방식
  - Amazon EC2 Auto Scaling을 사용하면 애플리케이션의 로드를 처리하는 데 사용할 수 있는 적절한 수의 Amazon EC2 인스턴스를 확보할 수 있습니다. Auto Scaling 그룹이라는 EC2 인스턴스 모음을 생성합니다. 각 Auto Scaling 그룹의 최소 인스턴스 수를 지정할 수 있으며, Amazon EC2 Auto Scaling은 사용하면 그룹의 크기가 이 값 아래로 내려가지 않게 합니다. 각 오토 스케일링 그룹의 최대 인스턴스 수를 지정할 수 있으며, Amazon EC2 Auto Scaling은 그룹의 크기가 이 값을 초과하지 않게 합니다.
    - Amazon EC2 Auto Scaling이란 무엇입니까?
  - Amazon DynamoDB Auto Scaling은 AWS Application Auto Scaling 서비스를 사용하여 사용자를 대신해 실제 트래픽 패턴에 따라 프로비저닝된 처리량 용량을 동적으로 조정합니다. 따라서 테이

블 또는 글로벌 보조 인덱스를 통해 프로비저닝된 읽기 및 쓰기 용량을 늘려 조절 없이 급증하는 트래픽을 처리할 수 있습니다.

- DynamoDB Auto Scaling을 통한 처리량 자동 관리

리소스

관련 문서:

- APN 파트너: 자동화된 컴퓨팅 솔루션의 생성을 지원할 수 있는 파트너
- AWS Auto Scaling: 조정 계획 작동 방식
- AWS Marketplace: 오토 스케일링과 함께 사용할 수 있는 제품
- DynamoDB Auto Scaling을 통한 처리량 자동 관리
- Amazon EC2 Auto Scaling이란 무엇입니까?

REL07-BP03 워크로드에 더 많은 리소스가 필요한 것으로 감지되면 리소스 확보

수요를 충족하고 가용성에 영향을 미치지 않도록 리소스를 사전에 확장합니다.

많은 AWS 서비스가 수요에 맞춰 자동으로 확장됩니다. Amazon EC2 인스턴스 또는 Amazon ECS 클러스터를 사용하는 경우 워크로드 수요에 해당하는 사용량 지표에 따라 이러한 자동 조정이 수행되도록 구성할 수 있습니다. Amazon EC2의 경우 평균 CPU 사용률, 로드 밸런서 요청 수 또는 네트워크 대역폭을 사용하여 EC2 인스턴스를 스케일아웃(또는 스케일인)할 수 있습니다. Amazon ECS의 경우 평균 CPU 사용률, 로드 밸런서 요청 수 및 메모리 사용률을 사용하여 ECS 작업을 스케일 아웃(또는 스케일 인)할 수 있습니다. AWS에서 Target Auto Scaling을 사용하면 Autoscaler가 가정용 온도 조절기처럼 작동하여 리소스를 추가하거나 제거함으로써 지정한 목표 값(예: 70%의 CPU 사용률)을 유지합니다.

또한 AWS Auto Scaling은 기계 학습을 사용하여 각 리소스의 기간별 워크로드를 분석하고 향후 2일간의 로드를 주기적으로 예측하는 Predictive Auto Scaling을 수행할 수도 있습니다.

리틀의 법칙은 필요한 컴퓨팅 인스턴스(EC2 인스턴스, 동시 Lambda 함수 등) 수를 계산하는 데 도움이 됩니다.

$L = \lambda W$

L = 인스턴스 수(또는 시스템의 평균 동시성)

λ = 요청이 도착하는 평균 속도(요청/초)

W = 시스템이 각 요청에 소비하는 평균 시간(초)

예를 들어 100rps에서 각 요청을 처리하는 데 0.5초가 걸리는 경우 수요를 따라가려면 50개의 인스턴스가 필요합니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 워크로드에 더 많은 리소스가 필요한 것으로 감지되면 리소스를 확보합니다. 수요를 충족하고 가용성에 영향을 미치지 않도록 리소스를 사전에 확장합니다.
  - 지정된 요청 속도를 처리하는 데 필요한 컴퓨팅 리소스(컴퓨팅 동시성)를 계산합니다.
    - Telling Stories About Little's Law
  - 사용량에 대한 과거 패턴이 있는 경우 Amazon EC2 Auto Scaling에 대한 예약된 조정을 설정합니다.
    - Amazon EC2 Auto Scaling의 예약된 조정
  - AWS 예측 크기 조정을 사용합니다.
    - Predictive Scaling for EC2, Powered by Machine Learning(기계 학습 기반 EC2용 예측 확장)

리소스

관련 문서:

- AWS Auto Scaling: 조정 계획 작동 방식
- AWS Marketplace: 오토 스케일링과 함께 사용할 수 있는 제품
- DynamoDB Auto Scaling을 통한 처리량 자동 관리
- Predictive Scaling for EC2, Powered by Machine Learning(기계 학습 기반 EC2용 예측 확장)
- Amazon EC2 Auto Scaling의 예약된 조정
- Telling Stories About Little's Law
- Amazon EC2 Auto Scaling란 무엇입니까?

REL07-BP04 워크로드 로드 테스트

확장 작업이 워크로드의 필요 사항을 충족하는지 측정하기 위한 로드 테스트 방식을 채택하십시오.

지속적인 로드 테스트를 수행하는 것이 중요합니다. 로드 테스트를 통해 한계점을 찾고 워크로드의 성능을 테스트할 수 있습니다. AWS를 사용하면 프로덕션 워크로드의 규모를 모델링하는 임시 테스트 환

경을 쉽게 설정할 수 있습니다. 클라우드에서는 온디맨드 방식으로 프로덕션 규모의 테스트 환경을 만들고, 테스트를 완료한 다음 해당 리소스를 폐기할 수 있습니다. 테스트 환경을 실행하는 동안에만 비용을 지불하면 되기 때문에 온프레미스 테스트 비용의 몇 분의 일에 불과한 가격으로 실제 환경을 시뮬레이션할 수 있습니다.

또한 프로덕션에서의 로드 테스트는 프로덕션 시스템에 스트레스가 가해지는 실전 연습의 일부로 간주되어야 하며 고객 사용량이 적은 시간에는 모든 직원이 결과를 해석하고 발생하는 문제를 해결해야 합니다.

일반적인 안티 패턴:

- 구성이 프로덕션 환경과 동일하지 않은 배포에 대해 로드 테스트 수행
- 전체 워크로드가 아니라 워크로드의 개별 부분에 대해서만 로드 테스트 수행
- 대표적인 실제 요청 세트가 아니라 요청의 하위 집합을 사용하여 로드 테스트 수행
- 예상 부하보다 작은 안전 계수로 부하 테스트 수행

이 모범 사례 수립의 이점: 로드 시 장애가 발생하는 아키텍처의 구성 요소를 알고 있으며, 해당 로드에 곧 도달할 것임을 나타내는 지표를 식별하고 조사하여 문제를 해결할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 로드 테스트를 수행하여 워크로드의 어느 측면에 용량을 추가 또는 제거해야 하는지 파악합니다. 로드 테스트에는 프로덕션 환경에서 수신하는 것과 유사한 대표 트래픽이 있어야 합니다. 계측한 지표를 모니터링하면서 로드를 늘려 리소스를 추가 또는 제거해야 하는 시점을 나타내는 지표를 결정합니다.
  - AWS에서의 분산 로드 테스트: 수천 명의 연결된 사용자 시뮬레이션
    - 요청의 조합을 식별합니다. 다양한 요청 조합이 있을 수 있으므로 트래픽 조합을 식별할 때 다양한 시간 프레임을 살펴봐야 합니다.
    - 로드 드라이버를 구현합니다. 사용자 지정 코드, 오픈 소스 또는 상용 소프트웨어를 사용하여 로드 드라이버를 구현할 수 있습니다.
    - 처음에는 작은 용량을 사용하여 로드 테스트를 수행합니다. 인스턴스 또는 컨테이너 하나 정도의 작은 용량으로 로드를 유도하면 즉각적인 효과가 나타납니다.
    - 더 큰 용량에 대해 로드 테스트를 수행합니다. 분산된 부하에서는 효과가 다르게 나타나므로 가능한 한 제품 환경에 가깝게 테스트해야 합니다.

리소스

관련 문서:

- AWS에서의 분산 로드 테스트: 수천 명의 연결된 사용자 시뮬레이션

REL 8 변경 사항은 어떻게 적용합니까?

새로운 기능을 배포하고 워크로드와 운영 환경에서 알려진 소프트웨어를 실행하고 예측 가능한 방식으로 패치 또는 교체할 수 있도록 하려면 변경 사항을 제어해야 합니다. 이러한 변경이 제어되지 않으면 변경의 영향을 예측하거나 변경으로 인해 발생하는 문제를 해결하기가 어려워집니다.

모범 사례

- REL08-BP01 배포와 같은 표준 활동에 런북 사용
- REL08-BP02 배포의 일부로 기능 테스트 통합
- REL08-BP03 배포의 일부로 복원력 테스트 통합
- REL08-BP04 변경 불가능한 인프라를 사용하여 배포
- REL08-BP05 자동화를 통한 변경 사항 배포

REL08-BP01 배포와 같은 표준 활동에 런북 사용

런북은 특정 결과를 달성하기 위한 미리 정의된 절차입니다. 수동 또는 자동으로 표준 활동을 수행할 때 런북을 사용합니다. 워크로드 배포, 워크로드 패치 적용 또는 DNS 수정과 같은 활동이 여기에 포함됩니다.

예를 들어 배포 중에 롤백이 안전한지 확인하는 프로세스를 준비합니다.. 서비스를 안정적으로 유지하려면 고객 중단 없이 배포를 롤백할 수 있는지 확인하는 것이 중요합니다.

런북 절차를 수행할 때는 유효하고 효과적인 수동 프로세스에서 시작하고, 이를 코드에 구현한 다음, 필요한 경우 자동으로 실행되도록 트리거합니다.

고도로 자동화된 정교한 워크로드의 경우에도 게임 데이를 실행하거나 엄격한 보고 및 감사 요구 사항을 충족할 때 런북을 유용하게 사용할 수 있습니다.

플레이북은 특정 인시던트에 대응하여 사용되며 런북은 특정 결과를 달성하기 위해 사용됩니다. 런북은 일상적인 활동에 대한 것이고, 플레이북은 일상적이지 않은 이벤트에 대응하는 데 사용되는 경우가 많습니다.

일반적인 안티 패턴:

- 프로덕션 환경에서 계획되지 않은 구성 변경 수행
- 더 빠르게 배포하기 위해 계획의 단계를 건너뛰고, 그 결과 배포에 실패
- 변경 사항 되돌리기를 테스트하지 않고 변경 수행

이 모범 사례 수립의 이점: 변경 계획을 효과적으로 수립하면 영향을 받는 모든 시스템을 파악할 수 있으므로 변경 사항을 성공적으로 실행할 수 있습니다. 테스트 환경에서 변경 사항을 검증하면 신뢰성을 높일 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 런북에 절차를 문서화하면 잘 알려진 이벤트에 일관된 방식으로 신속하게 대응할 수 있습니다.
  - AWS Well-Architected Framework: 개념: 런북
- 코드형 인프라의 원칙을 사용해 인프라를 정의합니다. AWS CloudFormation 또는 신뢰할 수 있는 서드 파티 제품을 사용하여 인프라를 정의하면 버전 제어 소프트웨어를 통한 변경 사항의 각 버전을 차례대로 확인할 수 있습니다.
  - AWS CloudFormation 또는 신뢰할 수 있는 서드 파티 제품을 사용하여 인프라를 정의합니다.
    - AWS CloudFormation이란 무엇인가요?
  - 우수한 소프트웨어 설계 원칙으로 분리된 단일 템플릿을 생성합니다.
    - 구현 권한, 템플릿 및 책임자를 결정합니다.
      - AWS Identity and Access Management로 액세스 제어
    - 버전을 제어하려면 AWS CodeCommit 또는 신뢰할 수 있는 서드 파티 도구와 같은 소스 제어를 사용합니다.
      - AWS CodeCommit이란 무엇입니까?

리소스

관련 문서:

- APN 파트너: 자동화된 배포 솔루션의 생성을 지원할 수 있는 파트너
- AWS Marketplace: 배포 자동화에 사용할 수 있는 제품
- AWS Well-Architected Framework: 개념: 런북
- AWS CloudFormation이란 무엇인가요?
- AWS CodeCommit이란 무엇입니까?

관련 예시:

- 플레이북 및 런북으로 운영 자동화

REL08-BP02 배포의 일부로 기능 테스트 통합

기능 테스트는 자동화된 배포의 일부로 실행됩니다. 성공 기준이 충족되지 않으면 파이프라인이 중지되거나 롤백됩니다.

이러한 테스트는 파이프라인에서 프로덕션 전에 준비되는 사전 프로덕션 환경에서 실행됩니다. 이 작업을 배포 파이프라인의 일부로 수행하는 것이 가장 좋습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 배포의 일부로 기능 테스트를 통합합니다. 기능 테스트는 자동화된 배포의 일부로 실행됩니다. 성공 기준이 충족되지 않으면 파이프라인이 중지되거나 롤백됩니다.
  - AWS CodePipeline에 모델링된 소프트웨어 릴리스 파이프라인의 '테스트 작업'을 실행하는 중에 AWS CodeBuild를 호출합니다. 이 기능을 사용하면 단위 테스트, 정적 코드 분석, 통합 테스트 등 코드에 대해 다양한 테스트를 손쉽게 실행할 수 있습니다.
    - AWS CodePipeline Adds Support for Unit and Custom Integration Testing with AWS CodeBuild(AWS CodePipeline, AWS CodeBuild를 사용한 단위 및 맞춤형 통합 테스트 관련 지원 추가)
  - AWS Marketplace 솔루션을 사용하여 소프트웨어 제공 파이프라인의 일부로 자동화된 테스트를 실행합니다.
    - 소프트웨어 테스트 자동화

리소스

관련 문서:

- AWS CodePipeline Adds Support for Unit and Custom Integration Testing with AWS CodeBuild(AWS CodePipeline, AWS CodeBuild를 사용한 단위 및 맞춤형 통합 테스트 관련 지원 추가)
- 소프트웨어 테스트 자동화
- AWS CodePipeline란 무엇입니까?

REL08-BP03 배포의 일부로 복원력 테스트 통합

복원력 테스트(카오스 엔지니어링의 원칙 사용)는 사전 프로덕션 환경에서 자동화된 배포 파이프라인에 포함되어 실행됩니다.

이러한 테스트는 프로덕션 전 환경에서 파이프라인에서 준비되고 실행됩니다. 또한 프로덕션 환경에서도 다음의 일부로 실행되어야 합니다. 게임 데이.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 배포의 일부로 복원력 테스트를 통합합니다. 프로덕션 환경에서 격변의 조건을 견딜 수 있는 워크로드의 기능에 대한 신뢰성을 확보하기 위해 시스템을 실험하는 분야인 카오스 엔지니어링을 사용합니다.
  - 복원력 테스트는 결함 또는 리소스 저하를 발생시켜 워크로드가 설계된 복원력으로 대응하는지 평가합니다.
    - Well-Architected lab: Level 300: Testing for Resiliency of EC2 RDS and S3(Well-Architected 실습: 레벨 300: EC2 RDS 및 S3의 복원력 테스트)
  - 이러한 테스트는 자동화된 배포 파이프라인의 사전 프로덕션 환경에서 정기적으로 실행할 수 있습니다.
  - 또한 예약된 게임 데이의 일부로 프로덕션에서 실행해야 합니다.
  - 카오스 엔지니어링 원칙을 사용하여 다양한 장애 하에서 워크로드의 성능이 어떻게 나타날지에 대한 가설을 제안한 다음 복원력 테스트를 사용하여 가설을 테스트합니다.
    - Principles of Chaos Engineering(카오스 엔지니어링의 원칙)

리소스

관련 문서:

- Principles of Chaos Engineering(카오스 엔지니어링의 원칙)
- AWS Fault Injection Simulator란 무엇입니까?

관련 예시:

- Well-Architected lab: Level 300: Testing for Resiliency of EC2 RDS and S3(Well-Architected 실습: 레벨 300: EC2 RDS 및 S3의 복원력 테스트)

REL08-BP04 변경 불가능한 인프라를 사용하여 배포

변경 불가능한 인프라는 프로덕션 워크로드의 현재 위치에서 업데이트, 보안 패치 또는 구성 변경이 발생하지 않도록 규정하는 모델입니다. 변경이 필요한 경우 아키텍처가 새 인프라에 구축되고 프로덕션 환경에 배포됩니다.

변경 불가능한 인프라 패러다임의 가장 일반적인 구현 형태는 변경 불가능한 서버입니다.. 즉, 서버에 업데이트 또는 수정이 필요한 경우 이미 사용 중인 서버를 업데이트하는 대신 새 서버가 배포됩니다. 따라서 SSH를 통해 서버에 로그인하고 소프트웨어 버전을 업데이트하는 대신 애플리케이션의 모든 변경은 코드 리포지토리에 소프트웨어를 푸시(예: git push)하는 것으로 시작됩니다. 변경이 불가능한 인프라에서는 변경이 허용되지 않으므로 배포된 시스템의 상태를 확신할 수 있습니다. 변경이 불가능한 인프라는 본질적으로 더 일관되고 안정적이며 예측 가능하며 소프트웨어 개발 및 운영의 여러 측면을 간소화합니다.

변경이 불가능한 인프라에서 애플리케이션을 배포할 때는 Canary 또는 블루/그린 배포를 사용합니다.

Canary 배포 는 일반적으로 단일 서비스 인스턴스에서 실행되는 새 버전(Canary)으로 소수의 사용자를 연결하는 방식입니다. 그런 다음 생성되는 동작 변경 또는 오류를 면밀히 조사합니다. 중대한 문제가 발생하여 사용자에게 이전 버전을 다시 제공해야 하는 경우에는 Canary에서 트래픽을 제거할 수 있습니다. 배포가 정상적으로 진행되면 배포가 완료될 때까지 변경 사항에서 오류를 모니터링하면서 원하는 속도로 배포를 계속 진행할 수 있습니다. Canary 배포를 지원하는 배포 구성을 사용하여 AWS CodeDeploy를 구성할 수 있습니다.

블루/그린 배포 는 Canary 배포와 비슷합니다. 단, 전체 애플리케이션 플릿이 병렬로 배포됩니다. 이 패턴에서는 블루와 그린의 두 스택에서 번갈아 가며 배포를 수행합니다. 이 패턴에서도 새 버전으로 트래픽을 전송한 다음 배포에 문제가 발생하면 이전 버전으로 장애 복구할 수 있습니다. 일반적으로 모든 트래픽은 한 번에 전환되지만, Amazon Route 53의 가중치 기반 DNS 라우팅 기능을 사용하면 각 버전에 대한 트래픽의 일부를 사용하여 새 버전의 채택을 유도할 수도 있습니다. 블루/그린 배포를 지원하는 배포 구성을 사용하여 AWS CodeDeploy와 AWS Elastic Beanstalk를 구성할 수 있습니다.

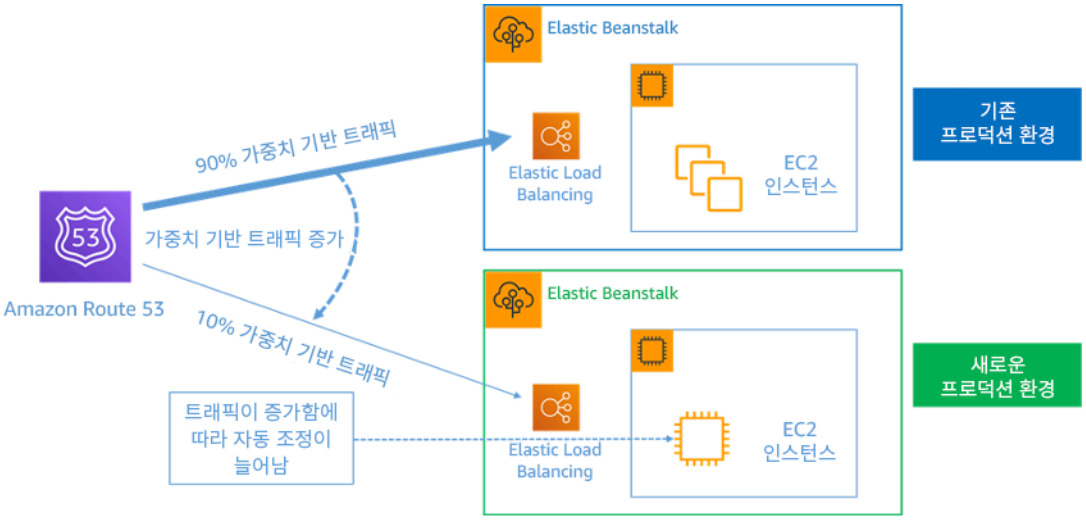


그림 8: AWS Elastic Beanstalk 및 Amazon Route 53를 사용한 블루/그린 배포

변경 불가능한 인프라의 이점:

- 구성 드리프트 감소: 알려진 기본 버전 제어 구성에서 서버를 자주 교체하면 인프라가 알려진 상태로 재설정되므로 구성 드리프트가 방지됩니다.
- 간소화된 배포: 업그레이드를 지원할 필요가 없으므로 배포가 간소화됩니다. 업그레이드는 단지 새로운 배포일 뿐입니다.
- 안정성 있는 원자 단위 배포: 배포가 성공적으로 완료되거나 아무 것도 변경되지 않습니다. 따라서 배포 프로세스의 신뢰도가 개선됩니다.
- 빠른 롤백 및 복구 프로세스를 통해 배포 안전성 개선: 이전 작업 버전이 변경되지 않으므로 배포가 더 안전합니다. 오류가 감지되면 롤백할 수 있습니다.
- 일관된 테스트 및 디버깅 환경: 모든 서버에 동일한 이미지가 사용되므로 환경 간에 차이가 없습니다. 하나의 빌드가 여러 환경에 배포됩니다. 또한 일관되지 않은 환경이 방지되고 테스트 및 디버깅이 간소화됩니다.
- 확장성 개선: 서버가 기본 이미지를 사용하고 일관적이며 반복 가능하므로 자동 조정이 간단합니다.
- 간소화된 도구 체인: 프로덕션 소프트웨어 업그레이드를 관리하는 구성 관리 도구를 제거할 수 있으므로 도구 체인이 간소화됩니다. 서버에 추가 도구 또는 에이전트가 설치되지 않습니다. 변경은 기본 이미지에 수행되고 테스트된 후 롤아웃됩니다.
- 보안 강화: 서버에 대한 모든 변경을 거부하면 인스턴스에서 SSH를 비활성화하고 키를 제거할 수 있습니다. 이렇게 하면 공격 벡터가 줄어들어 조직의 보안 태세를 개선할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 변경 불가능한 인프라를 사용하여 배포합니다. 변경 불가능한 인프라는 프로덕션 시스템의 현재 위치에서 업데이트, 보안 패치 또는 구성 변경이 발생하지 않도록 규정하는 모델입니다. 변경이 필요한 경우 새 버전의 아키텍처가 구축되고 프로덕션 환경에 배포됩니다.
  - 블루/그린 배포 개요
  - Deploying Serverless Applications Gradually(점진적으로 서버리스 애플리케이션 배포)
  - 변경 불가능한 인프라: 불변성을 통한 안정성, 일관성 및 신뢰성
  - CanaryRelease

리소스

관련 문서:

- CanaryRelease
- Deploying Serverless Applications Gradually(점진적으로 서버리스 애플리케이션 배포)
- 변경 불가능한 인프라: 불변성을 통한 안정성, 일관성 및 신뢰성
- 블루/그린 배포 개요
- Amazon Builders' Library: 배포 중 롤백 안전 보장

REL08-BP05 자동화를 통한 변경 사항 배포

배포 및 패치 적용이 자동화되므로 부정적인 영향이 제거됩니다.

프로덕션 시스템을 변경하는 것은 조직에서 가장 위험 부담이 큰 영역에 속합니다. 배포는 소프트웨어를 통해 해결할 수 있는 업무상의 문제와 더불어 해결해야 하는 가장 중요한 문제로 간주됩니다. 오늘날에는 배포 관련 문제를 해결하려면 운영 과정에서 해당하는 모든 영역(변경 사항 테스트/배포, 용량 추가/제거, 데이터 마이그레이션 포함)에 자동화를 사용해야 합니다. AWS CodePipeline을 사용하면 워크로드를 해제하는 데 필요한 단계를 관리할 수 있습니다. 여기에는 AWS CodeDeploy를 사용하여 Amazon EC2 인스턴스, 온프레미스 인스턴스, 서버리스 Lambda 함수 또는 Amazon ECS 서비스에 대한 애플리케이션 코드 배포를 자동화하는 배포 상태가 포함됩니다.

> 권장 사항
>
> 일반적인 통념으로는 가장 어려운 작업 절차에 사람을 투입하는 것이 좋다고 하지만 AWS에서는 바로 그런 이유로 가장 어려운 절차를 자동화할 것을 권장합니다.

일반적인 안티 패턴:

- 수동으로 변경 수행
- 비상 워크플로를 통해 자동화 단계를 건너뜀
- 계획을 따르지 않음

이 모범 사례 수립의 이점: 자동화를 사용하여 모든 변경 사항을 배포하면 인적 오류가 발생할 가능성이 사라지고 프로덕션을 변경하기 전에 테스트하여 계획이 완료되었는지 확인할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 배포 파이프라인을 자동화합니다. 배포 파이프라인을 사용하면 이상 징후의 자동 테스트 및 탐지를 실행하여 프로덕션 배포 전 특정 단계에서 파이프라인을 중지하거나 변경 사항을 자동으로 롤백할 수 있습니다.
  - Amazon Builders' Library: 배포 중 롤백 안전 보장
  - Amazon Builders' Library: 지속적 전달을 통한 신속한 배포
    - AWS CodePipeline 또는 신뢰할 수 있는 서드 파티 제품을 사용하여 파이프라인을 정의하고 실행합니다.
      - 코드 저장소에 변경 사항이 전달되고 나서 실행되도록 파이프라인을 구성합니다.
        - AWS CodePipeline이란 무엇입니까?
      - Amazon Simple Notification Service(Amazon SNS) 및 Amazon Simple Email Service(Amazon SES)를 사용하여 파이프라인의 문제에 대한 알림을 전송하거나 Amazon Chime과 같은 팀 채팅 도구에 통합합니다.
        - Amazon Simple Notification Service란 무엇입니까?
        - Amazon SES란 무엇입니까?
        - Amazon Chime이란 무엇일까요?
        - Automate chat messages with webhooks(Webhook으로 채팅 메시지 기능 자동화)

리소스

관련 문서:

- APN 파트너: 자동화된 배포 솔루션의 생성을 지원할 수 있는 파트너
- AWS Marketplace: 배포 자동화에 사용할 수 있는 제품

- Automate chat messages with webhooks(Webhook으로 채팅 메시지 기능 자동화)
- Amazon Builders' Library: 배포 중 롤백 안전 보장
- Amazon Builders' Library: 지속적 전달을 통한 신속한 배포
- AWS CodePipeline란 무엇입니까?
- CodeDeploy란 무엇입니까?
- AWS Systems Manager Patch Manager
- Amazon SES란 무엇입니까?
- Amazon Simple Notification Service란 무엇입니까?

관련 동영상:

- AWS Summit 2019: CI/CD on AWS(AWS 기반 CI/CD)

장애 관리

질문

- REL 9 데이터는 어떻게 백업합니까?
- REL 10 장애 격리를 사용하여 워크로드를 보호하려면 어떻게 해야 합니까?
- REL 11 구성 요소 장애를 견디도록 워크로드를 설계하려면 어떻게 해야 합니까?
- REL 12 안정성은 어떻게 테스트합니까?
- REL 13 DR(재해 복구)를 어떻게 계획합니까?

REL 9 데이터는 어떻게 백업합니까?

RTO(복구 시간 목표) 및 RPO(복구 시점 목표)에 대한 요구 사항을 충족하도록 데이터, 애플리케이션 및 구성을 백업합니다.

모범 사례

- REL09-BP01 백업해야 하는 모든 데이터 확인 및 백업 또는 소스에서 데이터 복제
- REL09-BP02 백업 보안 및 암호화
- REL09-BP03 자동으로 데이터 백업 수행
- REL09-BP04 백업 무결성 및 프로세스를 확인하기 위해 데이터의 주기적인 복구 수행

REL09-BP01 백업해야 하는 모든 데이터 확인 및 백업 또는 소스에서 데이터 복제

워크로드에 사용되는 데이터 서비스 및 리소스의 백업 기능을 숙지하고 사용합니다. 대부분의 서비스는 워크로드 데이터를 백업할 수 있는 기능을 제공합니다.

원하는 결과: 데이터 소스가 식별되고 중요도에 따라 분류됩니다. 그런 다음 RPO에 따라 데이터 복구 전략을 수립합니다. 이 전략에는 데이터 소스 백업 또는 다른 소스에서 데이터를 복제하는 기능이 포함됩니다. 데이터가 손실될 경우 구현된 전략에 따라 정의된 RPO 또는 RTO 내에 복구 또는 데이터 재현이 가능합니다.

클라우드 성숙도 단계: 기초

일반적인 안티 패턴:

- 워크로드의 모든 데이터 소스와 그 중요도를 알지 못합니다.
- 중요한 데이터 소스의 백업을 생성하지 않습니다.
- 중요도를 기준으로 사용하지 않고 일부 데이터 소스의 백업만 생성합니다.
- RPO가 정의되지 않았거나 백업 주기가 RPO에 부합하지 않습니다.
- 백업이 필요한지 또는 데이터를 다른 소스에서 복제할 수 있는지 평가하지 않습니다.

이 모범 사례 확립의 이점: 백업이 필요한 지점을 파악하고 백업 생성을 위한 메커니즘을 구현하거나 외부 소스에서 데이터를 복제할 수 있는 경우 중단 시 데이터를 복원하고 복구하는 역량이 향상됩니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

모든 AWS 데이터 스토어는 백업 기능을 제공합니다. Amazon RDS 및 Amazon DynamoDB 등의 서비스는 추가적으로 시점 복구(PITR)를 활성화하는 자동화된 백업을 지원합니다. 따라서 현재 시간으로부터 최대 5분 전까지 원하는 시점으로 백업을 복원할 수 있습니다. 많은 AWS 서비스는 백업을 다른 AWS 리전으로 복사할 수 있는 기능을 제공합니다. AWS Backup 도구는 AWS 서비스 전체에서 데이터 보호를 중앙 집중화하고 자동화하는 기능을 제공합니다. AWS Elastic Disaster Recovery를 사용하면 초 단위로 측정되는 Recovery Point Objective(RPO)를 사용하여 전체 서버 워크로드를 복사하고 온프레미스, 크로스 AZ 또는 크로스 리전에서 지속적인 데이터 보호를 유지할 수 있습니다.

Amazon S3는 자체 관리형 및 AWS 관리형 데이터 소스의 백업 목적지로 사용할 수 있습니다. Amazon EBS, Amazon RDS, Amazon DynamoDB 등의 AWS 서비스는 기본적으로 백업을 생성하는 기능이 내장되어 있습니다. 서드 파티 백업 소프트웨어를 사용할 수도 있습니다.

온프레미스 데이터는 AWS Storage Gateway 또는 AWS DataSync를 사용하여 AWS 클라우드에 백업할 수 있습니다. Amazon S3 버킷을 사용하여 이 데이터를 AWS에 저장할 수 있습니다. Amazon S3는 데이터 스토리지 비용을 줄이기 위해 Amazon S3 Glacier 또는 S3 Glacier Deep Archive와 같은 여러 스토리지 계층을 제공합니다.

다른 소스에서 데이터를 재현하여 데이터 복구 요구 사항을 충족할 수 있습니다. 예를 들어 Amazon ElastiCache 복제본 노드 또는 Amazon RDS 읽기 복제본은 기본 노드가 손실된 경우 데이터를 복제하는 데 사용할 수 있습니다. 이와 같은 소스를 사용하여 Recovery Point Objective(RPO) 및 Recovery Time Objective(RTO)를 충족할 수 있는 경우 백업이 필요하지 않을 수 있습니다. 또 다른 예로 Amazon EMR로 작업하는 경우 Amazon S3에서 Amazon EMR로 데이터를 복제할 수 있는 한 HDFS 데이터 스토어를 백업할 필요가 없을 수 있습니다.

백업 전략을 선택할 때는 데이터 복구에 걸리는 시간을 고려하시기 바랍니다. 데이터를 복구하는 데 필요한 시간은 백업의 유형(백업 전략의 경우) 또는 데이터 복제 메커니즘의 복잡성에 따라 달라집니다. 이 시간은 워크로드의 RTO 이내여야 합니다.

구현 단계

1. 워크로드의 모든 데이터 소스를 파악합니다. 데이터는 데이터베이스, 볼륨, 파일 시스템, 로깅 시스템 및 객체 스토리지와 같은 여러 리소스에 저장할 수 있습니다. 데이터가 저장된 다양한 AWS 서비스와 이러한 서비스가 제공하는 백업 기능에 대한 관련 문서를 찾으려면 리소스 섹션을 참조하세요.
2. 중요도에 따라 데이터 소스를 분류합니다. 데이터 세트마다 워크로드의 중요도가 다르므로 복원력에 대한 요구 사항도 달라집니다. 예를 들어, 어떤 데이터는 매우 중요하며 0에 가까운 RPO가 필요하지만 어떤 데이터는 덜 중요하며 더 높은 RPO와 일부 데이터 손실을 용인할 수 있습니다. 마찬가지로, 데이터 세트마다 RTO 요구 사항이 다릅니다.
3. AWS 또는 타사 서비스를 사용하여 데이터 백업을 만듭니다. AWS Backup은 AWS에서 다양한 데이터 소스의 백업을 생성할 수 있는 관리형 서비스입니다. AWS Elastic Disaster Recovery는 AWS 리전에 대한 자동화된 1초 미만의 데이터 복제를 처리합니다. 이런 AWS 서비스의 대부분은 백업을 생성하는 기능이 기본적으로 내장되어 있습니다. AWS Marketplace에도 이런 기능을 제공하는 솔루션이 많이 있습니다. 다양한 AWS 서비스에서 데이터 백업을 생성하는 방법에 대한 정보는 아래 나열된 리소스를 참조하세요.
4. 백업되지 않는 데이터의 경우 데이터 복제 메커니즘을 수립합니다. 여러 가지 이유로 다른 소스에서 복제될 수 있는 데이터는 백업하지 않기로 결정할 수 있습니다. 백업을 저장하는 데는 비용이 들기 때문에 백업을 생성하기보다는 필요할 때 다른 소스에서 데이터를 복제하는 것이 더 저렴한 상황이 있을 수도 있습니다. 또는 백업을 복원하는 작업이 다른 소스에서 데이터를 복제하는 것보다 더 오래 걸려 RTO를 준수할 수 없을 수도 있습니다. 그런 경우에는 장단점을 비교하여 데이터 복구가 필요할 때 다른 소스에서 데이터를 복제하는 방법에 대한 프로세스를 효과적으로 정의하여 수립해야 합니다. 예를 들면, 분석을 위해 Amazon S3의 데이터를 데이터 웨어하우스(예: Amazon Redshift)

또는 MapReduce 클러스터(예: Amazon EMR)로 로드한 경우 다른 소스에서 데이터를 복제한 예가 될 수 있습니다. 이러한 분석 결과가 어딘가에 저장되어 있거나 재현될 수만 있다면 데이터 웨어하우스 또는 MapReduce 클러스터의 장애로 인해 데이터 손실이 발생하지 않습니다. 소스에서 복제할 수 있는 다른 예로는 캐시(예: Amazon ElastiCache) 또는 RDS 읽기 전용 복제본이 있습니다.

5. 데이터 백업 주기를 설정합니다. 데이터 소스의 백업을 생성하는 일은 주기적으로 이루어져야 하며 빈도는 RPO에 따라 다릅니다.

구현 계획의 작업 수준: 보통

리소스

관련 모범 사례:

REL13-BP01 가동 중단 시간 및 데이터 손실 시의 복구 목표 정의

REL13-BP02 복구 목표 달성을 위해 정의된 복구 전략 사용

관련 문서:

- AWS Backup란 무엇입니까?
- AWS란 무엇입니까?
- Volume Gateway란 무엇입니까?
- APN 파트너: 백업을 지원할 수 있는 파트너
- AWS Marketplace: 백업에 사용할 수 있는 제품
- Amazon EBS 스냅샷
- Amazon EFS 백업
- Amazon FSx for Windows File Server 백업
- ElastiCache for Redis의 백업 및 복원
- Neptune에서 DB 클러스터 스냅샷 생성
- DB 스냅샷 생성
- Creating an EventBridge Rule That Triggers on a Schedule(일정에 따라 트리거되는 EventBridge 규칙 생성)
- Amazon S3로 리전 간 복제
- EFS-to-EFS AWS Backup
- Amazon S3로 로그 데이터 내보내기

- 객체 수명 주기 관리
- DynamoDB에 대한 온디맨드 백업 및 복원
- DynamoDB의 시점 복구
- Amazon OpenSearch Service 인덱스 스냅샷 작업
- AWS Elastic Disaster Recovery란 무엇입니까?

관련 동영상:

- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS(AWS re:Invent 2021 - AWS를 통한 백업, 재해 복구 및 랜섬웨어 보호)
- AWS Backup Demo: Cross-Account and Cross-Region Backup(AWS Backup 데모: 크로스 계정 및 크로스 리전 백업)
- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace(STG341)(AWS re:Invent 2019: AWS Backup 심층 분석, ft. Rackspace(STG341))

관련 예시:

- Well-Architected lab: Implementing Bi-Directional Cross-Region Replication (CRR) for Amazon S3(Well-Architected 실습: Amazon S3에 대한 양방향 크로스 리전 복제(CRR) 구현)
- Well-Architected lab: Testing Backup and Restore of Data(Well-Architected 실습: 데이터 백업 및 복원 테스트)
- Well-Architected Lab - Backup and Restore with Failback for Analytics Workload(Well-Architected 실습 - 분석 워크로드에 대한 백업 및 페일백으로 복원)
- Well-Architected Lab - Disaster Recovery - Backup and Restore(Well-Architected 실습 - 재해 복구 - 백업 및 복원)

REL09-BP02 백업 보안 및 암호화

인증 및 권한 부여를 사용하여 백업에 대한 액세스를 제어하고 감지합니다. 백업의 데이터 무결성이 침해되었을 경우 암호화 기법을 사용하여 예방 및 감지합니다.

일반적인 안티 패턴:

- 백업과 복원 자동화에 대한 액세스 권한을 데이터에 대한 액세스 권한과 동일하게 설정
- 백업을 암호화하지 않음

이 모범 사례 확립의 이점: 백업의 보안을 유지하면 데이터 변조가 방지되며, 데이터 암호화는 데이터가 실수로 노출될 경우 해당 데이터에 대한 액세스를 방지합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

AWS Identity and Access Management(IAM) 등의 인증 및 권한 부여를 사용하여 백업에 대한 액세스를 제어하고 감지합니다. 백업의 데이터 무결성이 침해되었을 경우 암호화 기법을 사용하여 예방 및 감지합니다.

Amazon S3는 유휴 데이터 암호화를 위한 다양한 방법을 지원합니다. Amazon S3는 서버 측 암호화를 사용하여 객체를 암호화되지 않은 데이터로 수락한 다음 저장 시 암호화합니다. 클라이언트측 암호화를 사용하면 데이터가 Amazon S3로 전송되기 전에 워크로드 애플리케이션에서 데이터가 암호화됩니다. 어느 방법을 사용하든 AWS Key Management Service(AWS KMS)를 사용하여 데이터 키를 생성 및 저장하거나 사용자가 관리하는 자체 키를 제공할 수 있습니다. AWS KMS를 사용하면 IAM을 사용하여 데이터 키와 해독된 데이터에 접근할 수 있는 사용자와 접근할 수 없는 사용자에 대한 정책을 설정할 수 있습니다.

Amazon RDS의 경우 데이터베이스를 암호화하도록 선택하면 백업도 암호화됩니다. DynamoDB 백업은 항상 암호화됩니다. AWS Elastic Disaster Recovery를 사용하면 전송 중이거나 저장된 모든 데이터가 암호화됩니다. Elastic Disaster Recovery를 사용하면 기본 Amazon EBS 암호화 볼륨 암호화 키 또는 사용자 지정 고객 관리형 키를 사용하여 저장 데이터를 암호화할 수 있습니다.

구현 단계

1. 각 데이터 스토어에 암호화를 사용합니다. 소스 데이터가 암호화되면 백업도 암호화됩니다.
    - Amazon RDS에서 암호화를 사용합니다. RDS 인스턴스를 생성할 때 AWS Key Management Service를 사용하여 저장 데이터의 암호화를 구성할 수 있습니다.
    - Amazon EBS 볼륨에서 암호화를 사용합니다. 볼륨 생성 시 기본 암호화를 구성하거나 고유한 키를 지정할 수 있습니다.
    - 필수 Amazon DynamoDB 암호화를 사용합니다. DynamoDB는 모든 저장 데이터를 암호화합니다. 계정에 저장된 키를 지정하여 AWS 소유 AWS KMS 키 또는 AWS 관리형 KMS 키를 사용할 수 있습니다.
    - Amazon EFS에 저장된 데이터를 암호화합니다. 파일 시스템을 생성할 때 암호화를 구성합니다.
    - 원본 및 대상 리전에 암호화를 구성합니다. KMS에 저장된 키를 사용하여 Amazon S3에서 저장 데이터 암호화를 구성할 수 있지만 키는 리전별로 다릅니다. 복제를 구성할 때 대상 키를 지정할 수 있습니다.

- 기본 또는 사용자 지정 Amazon EBS 암호화를 Elastic Disaster Recovery에 대해 사용할지 여부를 선택합니다. 이 옵션은 스테이징 영역 서브넷 디스크와 복제된 디스크에 복제된 저장 데이터를 암호화합니다.

2. 백업에 액세스할 수 있는 최소 권한을 구현합니다. 보안 모범 사례에 따라 백업, 스냅샷 및 복제본에 대한 액세스를 제한합니다.

리소스

관련 문서:

- AWS Marketplace: 백업에 사용할 수 있는 제품
- Amazon EBS Encryption(Amazon EBS 암호화)
- Amazon S3: Protecting Data Using Encryption(Amazon S3: 암호화를 사용하여 데이터 보호)
- CRR Additional Configuration: Replicating Objects Created with Server-Side Encryption (SSE) Using Encryption Keys stored in AWS KMS(CRR 추가 구성: AWS KMS에 저장된 암호화 키를 사용하여 서버 측 암호화(SSE)로 생성된 객체 복제)
- DynamoDB Encryption at Rest(DynamoDB 저장 시 암호화)
- Encrypting Amazon RDS Resources(Amazon RDS 리소스 암호화)
- Encrypting Data and Metadata in Amazon EFS(EFS의 데이터 및 메타데이터 암호화)
- Encryption for Backups in AWS(AWS의 백업 암호화)
- 암호화된 테이블 관리
- Security Pillar - AWS Well-Architected Framework(보안 원칙 - AWS Well-Architected Framework)
- AWS Elastic Disaster Recovery란 무엇입니까?

관련 예시:

- Well-Architected lab: Implementing Bi-Directional Cross-Region Replication (CRR) for Amazon S3(Well-Architected 실습: Amazon S3에 대한 양방향 크로스 리전 복제(CRR) 구현)

REL09-BP03 자동으로 데이터 백업 수행

Recovery Point Objective(RPO)에 정의된 정기적인 일정 또는 데이터 세트의 변경에 따라 백업이 자동으로 생성되도록 구성합니다. 적은 데이터 손실을 요구하는 중요한 데이터 세트는 자주 자동으로 백업되어야 합니다. 반면 일부 손실은 용인하는 중요도가 상대적으로 낮은 데이터의 경우 더 낮은 빈도로 백업할 수 있습니다.

원하는 결과: 정해진 주기로 데이터 소스의 백업을 생성하는 자동화된 프로세스

일반적인 안티 패턴:

- 수동으로 백업 수행
- 백업을 지원하는 리소스를 사용하지만 자동화 대상에 백업을 포함하지 않음

이 모범 사례 확립의 이점: 백업을 자동화하면 RPO를 기준으로 주기적으로 백업이 생성되며, 생성되지 않은 경우 알림을 보냅니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

AWS Backup을 사용하여 다양한 AWS 데이터 소스의 자동화된 데이터 백업을 생성할 수 있습니다. Amazon RDS 인스턴스는 5분마다 거의 지속적으로 백업할 수 있으며 Amazon S3 객체는 15분마다 거의 지속적으로 백업될 수 있으므로 백업 기록에서 특정 시점으로 시점 복구(PITR)가 가능합니다. Amazon EBS 볼륨, Amazon DynamoDB 테이블 또는 Amazon FSx 파일 시스템과 같은 다른 AWS 데이터 소스의 경우 AWS Backup은 최대 빈도 1시간 간격으로 자동화된 백업을 실행할 수 있습니다. 이러한 서비스는 기본 백업 기능도 제공합니다. 시점 복구와 함께 자동 백업을 제공하는 AWS 서비스에는 Amazon DynamoDB, Amazon RDS 및 Amazon Keyspaces(Apache Cassandra용)가 포함되며, 백업 기록 내의 특정 시점으로 복원할 수 있습니다. 다른 AWS 데이터 스토리지 서비스는 대부분 최대 빈도 1시간 간격으로 주기적 백업 일정을 수립하는 기능을 제공합니다.

Amazon RDS 및 Amazon DynamoDB는 시점 복구를 통해 지속적인 백업을 제공합니다. Amazon S3 버전 관리는 일단 활성화되면 자동으로 이루어집니다. Amazon Data Lifecycle Manager를 사용하여 Amazon EBS 스냅샷의 생성, 복사 및 삭제를 자동화할 수 있습니다. 또한 Amazon EBS 기반 Amazon Machine Image(AMI) 및 기본 Amazon EBS 스냅샷의 생성, 복사, 사용 중단 및 등록 취소도 자동화할 수 있습니다.

AWS Elastic Disaster Recovery는 소스 환경(온프레미스 또는 AWS)에서 대상 복구 리전으로 지속적인 블록 수준 복제를 제공합니다. 특정 시점 Amazon EBS 스냅샷은 서비스에서 자동으로 생성 및 관리됩니다.

AWS Backup은 백업 자동화 및 기록을 중앙 집중식으로 볼 수 있는 완전관리형 정책 기반 백업 솔루션을 제공합니다. AWS Storage Gateway를 사용하여 클라우드뿐 아니라 온프레미스의 여러 AWS 서비스에 걸쳐 데이터 백업을 중앙 집중화하고 자동화합니다.

Amazon S3는 버전 관리에 더해 복제 기능도 제공합니다. 전체 S3 버킷을 같거나 다른 AWS 리전의 다른 버킷에 자동으로 복제할 수 있습니다.

구현 단계

1. 현재 수동으로 백업 중인 데이터 소스를 식별합니다. 자세한 내용은 REL09-BP01 백업해야 하는 모든 데이터 확인 및 백업 또는 소스에서 데이터 복제의 내용을 참조하세요.
2. 워크로드에 대한 RPO를 결정합니다. 자세한 내용은 REL13-BP01 가동 중단 시간 및 데이터 손실 시의 복구 목표 정의단축할 수 있습니다.
3. 자동화된 백업 솔루션 또는 관리형 서비스를 사용합니다. AWS Backup은 클라우드 및 온프레미스에서 AWS 서비스 전반에 걸쳐 데이터 보호를 쉽게 중앙 집중화하고 자동화할 수 있는 완전관리형 서비스입니다. AWS Backup에서 백업 계획을 사용하여 백업할 리소스와 이러한 백업을 생성해야 하는 빈도를 정의하는 규칙을 만듭니다. 이 빈도는 2단계에서 설정한 RPO를 기반으로 해야 합니다. AWS Backup을 사용하여 자동 백업을 생성하는 방법에 대한 실습 지침은 데이터 백업 및 복원 테스트를 참조하세요. 데이터를 저장하는 AWS 서비스에서는 대부분 기본적으로 백업 기능을 제공합니다. 예를 들어, RDS를 사용하여 시점 복구(PITR) 기능이 있는 자동화된 백업을 생성할 수 있습니다.
4. 자동화된 백업 솔루션 또는 관리형 서비스에서 지원되지 않는 데이터 소스의 경우 신뢰할 수 있는 서드 파티 솔루션을 사용하여 자동화된 백업을 생성하는 것을 고려하세요. 아니면 AWS CLI 또는 SDK를 사용하여 백업 생성을 위한 자동화를 생성할 수 있습니다. AWS Lambda 함수 또는 AWS Step Functions를 사용하여 데이터 백업 생성의 로직을 정의하고, Amazon EventBridge를 사용하여 RPO를 기반으로 정해진 빈도에 백업을 실행합니다.

구현 계획의 작업 수준: 낮음.

리소스

관련 문서:

- APN 파트너: 백업을 지원할 수 있는 파트너
- AWS Marketplace: 백업에 사용할 수 있는 제품
- Creating an EventBridge Rule That Triggers on a Schedule(일정에 따라 트리거되는 EventBridge 규칙 생성)
- AWS Backup란 무엇입니까?
- AWS Step Functions란 무엇인가요?
- AWS Elastic Disaster Recovery란 무엇입니까?

관련 동영상:

- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace(STG341)

관련 예시:

- Well-Architected lab: Testing Backup and Restore of Data(Well-Architected 실습: 데이터 백업 및 복원 테스트)

REL09-BP04 백업 무결성 및 프로세스를 확인하기 위해 데이터의 주기적인 복구 수행

복구 테스트를 수행하여 백업 프로세스 구현이 Recovery Time Objective(RTO) 및 Recovery Point Objective(RPO)를 충족하는지 검증합니다.

원하는 결과: 효과적으로 정의된 메커니즘을 사용하여 백업의 데이터가 주기적으로 복구되어 워크로드에 정해진 Recovery Time Objective(RTO) 내에 복구가 가능하도록 합니다. 원본 데이터가 손상되거나 액세스가 불가능하지 않고 Recovery Point Objective(RPO)에서 허용되는 데이터 손실만 이루어진 채로 백업이 원본 데이터가 포함된 리소스로 복원되는지 확인합니다.

일반적인 안티 패턴:

- 백업을 복원하지만 복원이 사용 가능한 상태인지 확인하기 위해 데이터를 쿼리하거나 검색하지 않습니다.
- 백업이 존재한다고 가정합니다.
- 시스템의 백업이 완전히 작동하며 시스템에서 데이터를 복구할 수 있다고 가정합니다.
- 복원 시점 또는 백업에서의 데이터 복구가 워크로드의 RTO 이내에 이루어진다고 가정합니다.
- 백업에 포함된 데이터가 워크로드의 RPO에 부합한다고 가정합니다.
- 런북을 사용하지 않거나 설정된 자동화 절차를 벗어나 필요에 따라 복원합니다.

이 모범 사례 확립의 이점: 백업의 복구를 테스트하면 데이터가 손실되거나 손상되는 것을 걱정하지 않고 필요할 때 데이터를 복원할 수 있으며, 복원 및 복구가 워크로드의 RTO 내에 이루어지도록 하며, 데이터 손실이 있는 경우 워크로드의 RPO에 부합하도록 할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

백업 및 복원 기능을 테스트하면 중단 시 백업 및 복원 수행의 역량에 대한 신뢰도를 높일 수 있습니다. 주기적으로 백업을 새로운 위치에 복원하고 테스트를 실행하여 데이터의 무결성을 확인합니다. 수행해야 하는 몇 가지 일반적인 테스트는 모든 데이터가 사용 가능한지, 손상되지 않았는지, 액세스 가능한지, 모든 데이터 손실이 워크로드에 대한 RPO 내에 속하는지 확인하는 것입니다. 이러한 테스트는 복구 메커니즘의 속도가 워크로드의 RTO에 부합할 만큼 빠른지 확인하는 데도 도움이 됩니다.

AWS를 사용하면 테스트 환경을 구축하고 백업을 복원하여 RTO 및 RPO 기능을 평가하고 데이터 콘텐츠와 무결성에 대한 테스트를 실행할 수 있습니다.

또한 Amazon RDS와 Amazon DynamoDB는 PITR(특정 시점 복구)을 지원합니다. 연속 백업을 사용하면 데이터 세트를 지정된 날짜 및 시간의 상태로 복원할 수 있습니다.

모든 데이터를 사용 가능한지, 데이터가 손상되지 않았는지, 모든 데이터에 액세스할 수 있는지, 데이터 손실이 있는 경우 워크로드의 RPO에 부합하는지 확인하는 작업이 있습니다. 이러한 테스트는 복구 메커니즘의 속도가 워크로드의 RTO에 부합할 만큼 빠른지 확인하는 데도 도움이 됩니다.

AWS Elastic Disaster Recovery는 Amazon EBS 볼륨의 지속적인 특정 시점 복구 스냅샷을 제공합니다. 소스 서버가 복제되면 구성된 정책에 따라 특정 시점 상태가 시간 경과에 따라 기록됩니다. Elastic Disaster Recovery는 트래픽을 리디렉션하지 않고 테스트 및 드릴 목적으로 인스턴스를 시작하여 이러한 스냅샷의 무결성을 확인하는 데 도움이 됩니다.

구현 단계

1. 현재 백업되는 데이터 소스를 식별하고 이 백업이 어디에 저장되는지 파악합니다. 구현 가이드는 REL09-BP01 백업해야 하는 모든 데이터 확인 및 백업 또는 소스에서 데이터 복제의 내용을 참조하세요.
2. 각 데이터 소스에 대한 데이터 유효성 검사 기준을 설정합니다. 다양한 유형의 데이터에는 다양한 유효성 검사 메커니즘이 필요할 수 있는 다양한 속성이 있습니다. 확신을 갖고 프로덕션에 사용하기 전에 이 데이터가 어떻게 검증될 수 있는지 고려하세요. 데이터를 검증하는 몇 가지 일반적인 방법은 데이터 유형, 형식, 체크섬, 크기 등의 데이터 및 백업 속성을 사용하거나 이러한 속성과 사용자 지정 검증 로직을 결합하는 것입니다. 예를 들어, 복원된 리소스와 백업이 생성된 당시의 데이터 소스의 체크섬 값을 비교해 볼 수 있습니다.
3. 데이터 중요도에 따라 데이터 복구에 대한 RTO와 RPO를 설정합니다. 구현 가이드는 REL13-BP01 가동 중단 시간 및 데이터 손실 시의 복구 목표 정의단축할 수 있습니다.
4. 복구 기능을 평가합니다. 백업 및 복원 전략을 검토하여 RTO 및 RPO를 충족하는지 파악하고 필요에 따라 전략을 조정합니다. AWS Resilience Hub를 사용하여 워크로드 평가를 실행할 수 있습니다. 이 평가를 통해 애플리케이션 구성을 복원력 정책과 비교하고 RTO 및 RPO 목표가 충족될 수 있는지 보고합니다.
5. 데이터 복원을 위해 프로덕션에서 사용되는 현재 설정된 프로세스를 사용하여 테스트 복원을 수행합니다. 이 프로세스는 원본 데이터 소스가 백업되는 방법, 백업 자체의 형식 및 스토리지 위치 또는 다른 소스에서의 데이터 복제 여부에 따라 달라집니다. 예를 들어 AWS Backup과 같은 관리형 서비스를 사용하는 경우 백업을 새 리소스로 복원하는 것만큼 간단할 수 있습니다. AWS Elastic Disaster Recoveryy를 사용한 경우 복구 훈련을 시작할 수 있습니다

6. 데이터 유효성 검사를 위해 이전에 설정한 기준에 따라 복원된 리소스에서 데이터 복구 유효성을 검사합니다. 복원되고 복구된 데이터에 백업 시 가장 최신 레코드 또는 항목이 포함되어 있습니까? 이 데이터가 워크로드의 RPO에 부합합니까?
7. 복원 및 복구에 필요한 시간을 측정하고 기존 RTO와 비교합니다. 프로세스가 워크로드의 RTO에 부합합니까? 예를 들어, 복원 프로세스가 시작됐을 때의 타임스탬프와 복구 검증이 완료됐을 때의 타임스탬프를 비교하여 프로세스에 걸린 시간을 계산합니다. 모든 AWS API 호출에는 타임스탬프가 지정되며 이 정보는 AWS CloudTrail에서 사용할 수 있습니다. 이 정보가 복원 프로세스가 시작된 시간에 대한 세부 정보를 제공하지만 검증이 완료된 종료 타임스탬프는 검증 로직에서 기록되어야 합니다. 자동화된 프로세스를 사용하는 경우 Amazon DynamoDB와 같은 서비스를 사용하여 이 정보를 저장할 수 있습니다. 또한 많은 AWS 서비스에서 특정 작업이 발생한 시점의 타임스탬프가 기록된 정보가 표시되는 이벤트 기록을 제공합니다. AWS Backup 내에서 백업 및 복원 작업을 작업이라고 하며 이러한 작업에는 복원 및 복구에 필요한 시간을 측정하는 데 사용할 수 있는 메타데이터의 일부로 타임스탬프 정보가 포함되어 있습니다.
8. 데이터 검증이 실패하거나 복원 및 복구에 필요한 시간이 워크로드에 정해진 RTO를 초과하는 경우 이해 관계자에게 알립니다. 이 실습과 같이 자동화를 구현하는 경우 Amazon Simple Notification Service(Amazon SNS)와 같은 서비스를 사용하여 이해 관계자에게 이메일 또는 SMS와 같은 푸시 알림을 보낼 수 있습니다. 이러한 메시지는 Amazon Chime, Slack 또는 Microsoft Teams와 같은 메시징 애플리케이션에 게시하거나 AWS Systems Manager OpsCenter를 사용하여 작업을 OpsItems로 생성하는 데 사용할 수도 있습니다.
9. 주기적으로 실행되도록 프로세스를 자동화합니다. 예를 들어, AWS Lambda 또는 AWS Step Functions의 State Machine과 같은 서비스를 사용하면 복원 및 복구 프로세스를 자동화할 수 있으며, Amazon EventBridge를 사용하여 아래 아키텍처 다이어그램에서 보이는 것처럼 이 자동 워크플로를 주기적으로 트리거할 수 있습니다. Automate data recovery validation with AWS Backup(AWS Backup으로 데이터 복구 검증 자동화)하는 방법을 알아보세요. 또한 이 Well-Architected 실습은 여기에 있는 여러 단계에 대해 자동화를 수행하는 한 가지 방법에 대한 실습 경험을 제공합니다.

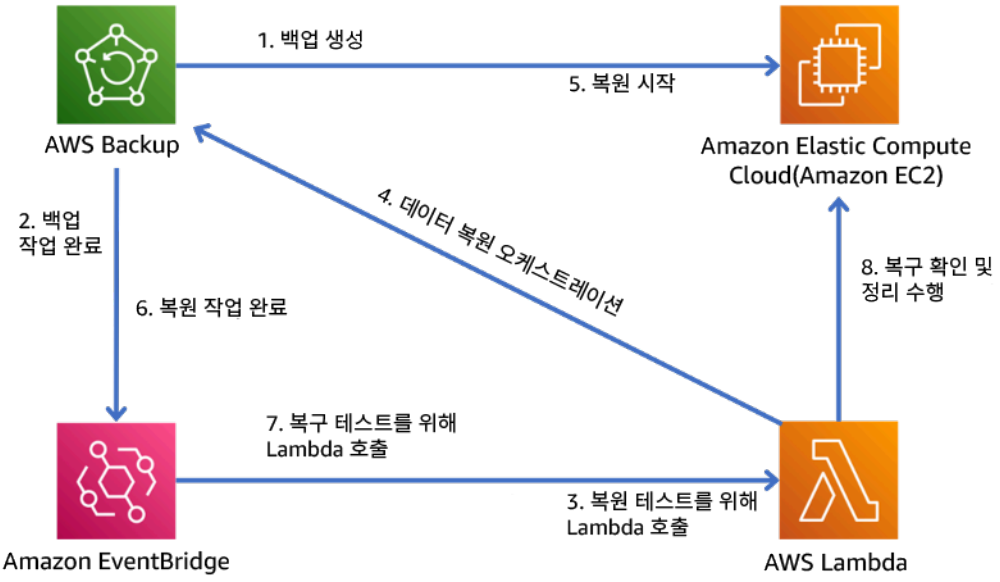


그림 9. 자동화된 백업 및 복원 프로세스

구현 계획의 작업 수준: 검증 기준의 복잡성에 따라 중간에서 높음.

리소스

관련 문서:

- Automate data recovery validation with AWS Backup(AWS Backup으로 데이터 복구 검증 자동화)
- APN 파트너: 백업을 지원할 수 있는 파트너
- AWS Marketplace: 백업에 사용할 수 있는 제품
- Creating an EventBridge Rule That Triggers on a Schedule(일정에 따라 트리거되는 EventBridge 규칙 생성)
- DynamoDB에 대한 온디맨드 백업 및 복원
- AWS Backup란 무엇입니까?
- AWS Step Functions란 무엇인가요?
- AWS Elastic Disaster Recovery란 무엇입니까?
- AWS Elastic Disaster Recovery

관련 예시:

- Well-Architected lab: Testing Backup and Restore of Data(Well-Architected 실습: 데이터 백업 및 복원 테스트)

REL 10 장애 격리를 사용하여 워크로드를 보호하려면 어떻게 해야 합니까?

장애 격리 경계는 워크로드 내부 장애의 영향을 제한된 수의 구성 요소로 제한합니다. 경계 외부의 구성 요소는 장애가 발생하더라도 영향을 받지 않습니다. 다수의 장애 격리 경계를 사용하여 워크로드에 미치는 영향을 제한할 수 있습니다.

모범 사례

- REL10-BP01 워크로드를 여러 위치에 배포
- REL10-BP02 다중 위치 배포에 적합한 위치 선택
- REL10-BP03 단일 위치로 제약된 구성 요소의 복구 자동화
- REL10-BP04 격벽 아키텍처를 사용하여 영향 범위 제한

REL10-BP01 워크로드를 여러 위치에 배포

워크로드 데이터와 리소스를 여러 가용 영역에 분산하거나 필요한 경우 AWS 리전 전체에 분산합니다. 필요에 따라 다양한 위치를 사용할 수 있습니다.

AWS의 서비스 설계 관련 기본 원칙 중 하나는 기본 물리적 인프라에서 단일 장애 지점이 없어야 한다는 것입니다. 이 원칙을 준수하려면 가용 영역 여러 개를 사용하며 단일 영역에서 장애가 발생해도 복원이 가능한 소프트웨어와 시스템을 구축해야 합니다. 마찬가지로, 시스템은 단일 컴퓨팅 노드, 단일 스토리지 볼륨 또는 단일 데이터베이스 인스턴스에서 장애가 발생하더라도 복원 가능하도록 구축됩니다. 중복 구성 요소를 사용하는 시스템을 구축할 때는 구성 요소가 독립적(AWS 리전의 경우 자율적)으로 작동해야 합니다. 중복 구성 요소를 사용한 이론적 가용성 계산에서 얻을 수 있는 이점은 이것이 사실인 경우에만 유효합니다.

가용 영역(AZ)

AWS 리전은 서로 독립적으로 설계된 여러 개의 가용 영역으로 구성됩니다. 화재, 홍수, 태풍 등의 자연재해로 인한 상관 장애 시나리오를 방지하기 위해 각 가용 영역은 물리적으로 유의미하게 떨어져 있도록 분리됩니다. 또한 각 가용 영역에는 독립된 물리적 인프라(다목적 전원에 대한 전용 연결, 대기 백업 전원, 독립 기계 서비스, 가용 영역 내/외부의 독립 네트워크 연결)가 포함되어 있습니다. 이 설계는 이 시스템 중 하나에서 발생한 장애가 영향을 받은 하나의 가용 영역에만 국한되도록 합니다. 가용 영역은 지리적으로는 분리되지만 같은 지역에 배치되어 높은 처리량과 낮은 지연 시간의 네트워킹이 가능합니다. 동기식으로 데이터를 복제하는 등(예: 데이터베이스 간에) 전체 AWS 리전(여러 개의 물리적으로 독립적인 데이터 센터로 구성된 모든 가용 영역의 AWS 리전)을 워크로드의 하나의 논리적 배포 대상으로 취급할 수 있습니다. 그러면 액티브/액티브 또는 액티브/대기 구성에서 가용 영역을 사용할 수 있습니다.

가용 영역은 서로 독립되어 있으므로 워크로드가 여러 영역을 사용하도록 아키텍팅된 경우 워크로드 가용성이 높아집니다. 일부 AWS 서비스(Amazon EC2 인스턴스 데이터 영역 포함)는 해당 서비스가 위치한 가용 영역과 운명을 같이하는 엄격한 영역별 서비스로 배포됩니다. 그러나 다른 가용 영역에 있는 Amazon EC2 인스턴스는 영향을 받지 않고 계속해서 작동합니다. 마찬가지로, 한 가용 영역에서의 장애로 인해 Amazon Aurora 데이터베이스에 장애가 발생하면 영향을 받지 않은 가용 영역에 있는 읽기 전용 복제본 Aurora 인스턴스가 자동으로 기본으로 승격될 수 있습니다. 반면 Amazon DynamoDB 등의 리전별 AWS 서비스는 사용자가 가용 영역 배치를 구성할 필요 없이 해당 서비스에 설정된 가용성 설계 목표를 달성하기 위해 내부적으로 액티브/액티브 구성에서 여러 가용 영역을 사용합니다.

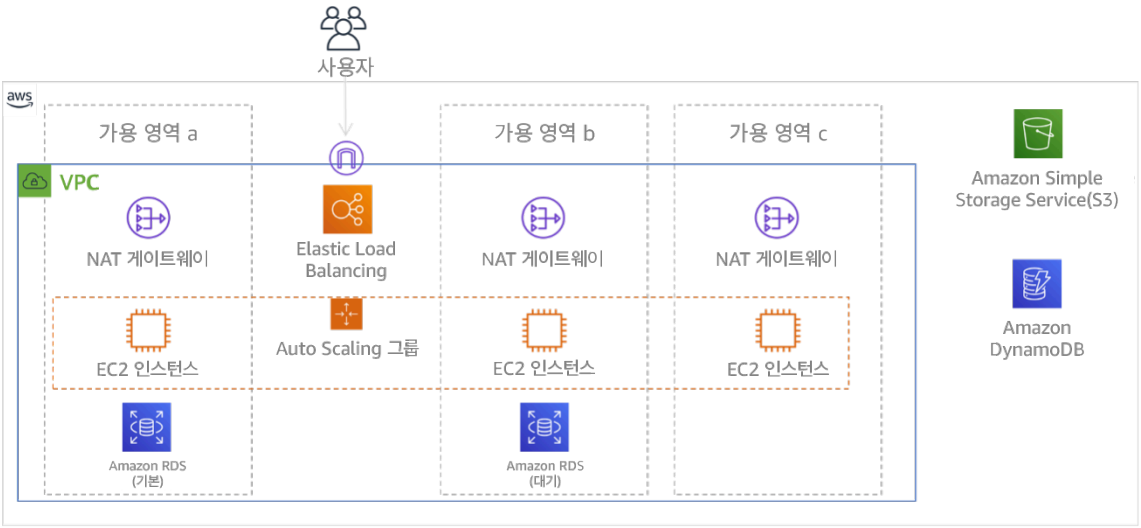


그림 9: 3개의 가용 영역에 배포된 다중 계층 아키텍처. Amazon S3와 Amazon DynamoDB는 항상 자동으로 다중 AZ에 유지됩니다. 또한 ELB는 3개 영역 모두에 배포됩니다.

AWS 컨트롤 플레인에서는 대개 전체 리전(여러 가용 영역) 내의 리소스를 관리하는 기능이 제공되지만, Amazon EC2, Amazon EBS 등의 특정 컨트롤 플레인에는 단일 가용 영역으로 결과를 필터링하는 기능도 있습니다. 이처럼 결과가 필터링되면 요청은 지정된 가용 영역에서만 처리되므로 다른 가용 영역이 중단될 가능성이 낮아집니다. 이 AWS CLI 예시는 us-east-2c 가용 영역에서만 Amazon EC2 인스턴스 정보를 가져오는 것을 보여줍니다.

```
AWS ec2 describe-instances --filters Name=availability-zone,Values=us-east-2c
```

AWS 로컬 영역

AWS 로컬 영역은 서브넷 및 EC2 인스턴스와 같은 영역 단위 AWS 리소스에 대한 배치 위치로 선택될 수 있다는 점에서 해당 AWS 리전 내의 가용 영역과 유사하게 작동합니다. 이러한 로컬 영역은 연결된 AWS 리전이 아니라 현재 AWS 리전이 없는 대규모 인구, 산업 및 IT 센터에 가까운 곳에 위치한다는 점에서 특별합니다. 그럼에도 불구하고 로컬 영역의 로컬 워크로드와 AWS 리전에서 실행 중인 워크로

드 간에는 고대역폭의 안전한 연결이 유지됩니다. 지연 시간이 짧아야 하는 요구 사항을 충족하기 위해 사용자에게 더 가까운 위치에 워크로드를 배포하려면 AWS 로컬 영역을 사용해야 합니다.

Amazon 글로벌 엣지 네트워크

Amazon 글로벌 엣지 네트워크는 전 세계 도시의 엣지 로케이션으로 구성됩니다. Amazon CloudFront는 이 네트워크를 사용하여 최종 사용자에게 더 짧은 지연 시간으로 콘텐츠를 전송합니다. AWS Global Accelerator를 사용하면 이러한 엣지 로케이션에 워크로드 엔드포인트를 생성하여 사용자와 가까운 AWS 글로벌 네트워크로의 온보딩을 제공할 수 있습니다. Amazon API Gateway는 CloudFront 배포를 사용하여 엣지 최적화 API 엔드포인트를 활성화함으로써 가장 가까운 엣지 로케이션을 통한 클라이언트 접근을 가속화합니다.

AWS 리전

AWS 리전은 자율적으로 작동하도록 설계되므로 다중 리전 접근 방식을 사용하려면 각 리전에 서비스의 전용 복사본을 배포해야 합니다.

다중 리전 접근법은 일회성의 대규모 이벤트가 발생할 때 복구 목표를 달성하기 위한 재해 복구 전략에 자주 사용됩니다. 이러한 전략에 대한 자세한 내용은 DR(재해 복구) 계획 을 참조하십시오. 그러나 여기에서는 시간 경과에 따라 평균 업타임 목표를 달성하는 가용성에 집중합니다. 목표가 고가용성인 경우 다중 리전 아키텍처가 일반적으로 액티브/액티브로 설계됩니다. 여기에서 각 서비스 복사본은 각각의 리전에서 액티브(요청을 서비스함)입니다.

> **추천**
>
> 단일 AWS 리전 안에서 다중 AZ 전략을 사용하여 대부분의 워크로드에 대한 가용성 목표를 충족할 수 있습니다. 워크로드에 매우 높은 가용성 요구 사항이 있는 경우 또는 다중 리전 아키텍처를 요구하는 다른 비즈니스 목표가 있는 경우에만 다중 리전 아키텍처를 고려하십시오.

AWS는 교차 리전에서 서비스를 운영할 수 있는 기능을 제공합니다. 예를 들어, AWS는 Amazon Simple Storage Service(Amazon S3) 복제본, Amazon RDS 읽기 전용 복제본(Aurora 읽기 전용 복제본 포함), Amazon DynamoDB 글로벌 테이블을 사용하여 지속적인 비동기식 데이터 복제를 제공합니다. 지속적인 복제본을 통해 각 액티브 리전에서 거의 즉시 데이터의 버전을 사용할 수 있습니다.

AWS CloudFormation을 사용하면 인프라를 정의하고 AWS 계정 및 AWS 리전 전체에 일관적으로 배포할 수 있습니다. 또한 AWS CloudFormation StackSets는 한 번의 작업으로 여러 계정과 리전에 AWS CloudFormation 스택을 생성, 업데이트, 삭제할 수 있도록 이 기능을 확장합니다. Amazon EC2 인스턴스 배포의 경우 AMI(Amazon Machine Image)를 사용하여 하드웨어 구성 및 설치된 소프트웨어

같은 정보를 제공합니다. 필요한 AMI를 생성하고 이것을 액티브 리전에 복사하는 Amazon EC2 Image Builder 파이프라인을 구현할 수 있습니다. 그렇게 하면 이 Golden AMI에 새로운 각 리전에 워크로드를 배포하고 스케일 아웃하는 데 필요한 모든 것이 포함됩니다.

트래픽을 라우팅하기 위해 Amazon Route 53 및 AWS Global Accelerator에서 모두 어떤 사용자가 어떤 액티브 리전 엔드포인트로 이동하는지를 결정하는 정책을 정의할 수 있습니다. Global Accelerator를 사용하면 트래픽 다이얼을 설정하여 각 애플리케이션 엔드포인트로 이동하는 트래픽의 비율을 제어할 수 있습니다. Route 53는 이 비율 접근법과 함께 지리 근접 및 지연 시간 기반 접근법 등 여러 정책을 지원합니다. Global Accelerator는 AWS 엣지 서버의 광범위한 네트워크를 자동으로 활용하여 트래픽이 가능한 한 빨리 AWS 네트워크 백본을 사용하도록 온보딩함으로써 요청 지연 시간을 단축합니다.

이런 기능은 모두 각 리전의 자율성을 보존하도록 작동합니다. 이 방식의 예외는 Amazon CloudFront 및 Amazon Route 53와 같이 글로벌 엣지 제공 기능을 제공하는 서비스와 AWS Identity and Access Management(IAM) 서비스용 컨트롤 플레인 등의 몇 가지뿐입니다. 대부분의 서비스는 전적으로 단일 리전 내에서 작동합니다.

온프레미스 데이터 센터

온프레미스 데이터 센터에서 실행되는 워크로드의 경우 가능하면 하이브리드 환경을 설계합니다. AWS Direct Connect는 온프레미스에서 AWS로 연결되는 전용 네트워크 연결을 제공하므로 두 환경에서 모두 워크로드를 실행할 수 있습니다.

또 다른 옵션은 AWS Outposts를 사용하여 온프레미스에서 AWS 인프라 및 서비스를 실행하는 것입니다. AWS Outposts는 AWS 인프라, AWS 서비스, API 및 도구를 온프레미스 데이터 센터로 확장하는 완전관리형 서비스입니다. AWS 클라우드에서 사용되는 것과 동일한 하드웨어 인프라가 데이터 센터에 설치됩니다. AWS Outposts는 가장 가까운 AWS 리전에 연결됩니다. 그런 다음에는 AWS Outposts를 사용하여 지연 시간이 짧아야 하거나 로컬 데이터 처리 요구 사항이 있는 워크로드를 지원할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 다중 가용 영역 및 AWS 리전을 선택합니다. 워크로드 데이터와 리소스를 여러 가용 영역에 분산하거나 필요한 경우 AWS 리전 전체에 분산합니다. 필요에 따라 다양한 위치를 사용할 수 있습니다.
  - 리전별 서비스는 가용 영역 전체에 배포됩니다.
    - 여기에는 Amazon S3, Amazon DynamoDB, AWS Lambda(VPC에 연결되지 않은 경우)가 포함됩니다.

- 컨테이너, 인스턴스 및 함수 기반 워크로드를 여러 가용 영역에 배포합니다. 캐시를 비롯한 다중 영역 데이터 스토어를 사용합니다. VPC에서 실행할 때 EC2 Auto Scaling, ECS 태스크 배치, AWS Lambda 함수 구성의 기능 및 ElastiCache 클러스터를 사용합니다.
  - Auto Scaling 그룹을 배포할 때는 서로 다른 가용 영역에 있는 서브넷을 사용합니다.
    - 예: 가용 영역 전반에 인스턴스 분산
    - Amazon ECS 태스크 배치 전략
    - Amazon VPC에서 리소스에 액세스하도록 AWS Lambda 함수 구성
    - 리전 및 가용 영역 선택
  - Auto Scaling 그룹을 배포할 때는 서로 다른 가용 영역에 있는 서브넷을 사용합니다.
    - 예: 가용 영역 전반에 인스턴스 분산
  - ECS 작업 배치 파라미터를 사용하여 DB 서브넷 그룹을 지정합니다.
    - Amazon ECS 태스크 배치 전략
  - VPC에서 실행할 기능을 구성할 때 여러 가용 영역의 서브넷을 사용합니다.
    - Amazon VPC에서 리소스에 액세스하도록 AWS Lambda 함수 구성
  - ElastiCache 클러스터를 통해 여러 가용 영역을 사용합니다.
    - 리전 및 가용 영역 선택
- 워크로드를 여러 리전에 배포해야 하는 경우 다중 리전 전략을 선택합니다. 단일 AWS 리전 내에서 다중 가용 영역 전략을 사용하여 대부분의 신뢰성 요구 사항을 충족할 수 있습니다. 비즈니스 요구 사항을 충족하는 데 필요한 경우 다중 리전 전략을 사용합니다.
  - AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications(다중 리전 액티브-액티브 애플리케이션을 위한 아키텍처 패턴)(ARC209-R2)
    - 다른 AWS 리전으로 백업하면 필요할 때 데이터를 사용할 수 있다는 보장이 추가됩니다.
    - 일부 워크로드의 경우 규제 요건에 따라 다중 리전 전략을 사용해야 합니다.
- 워크로드의 AWS Outposts를 평가합니다. 온프레미스 데이터 센터에 대한 지연 시간이 짧아야 하거나 로컬 데이터 처리 요구 사항을 충족해야 하는 워크로드의 경우 AWS Outposts를 사용하여 온프레미스에서 AWS 인프라와 서비스를 실행합니다.
  - AWS Outposts란 무엇입니까?
- AWS 로컬 영역이 사용자에게 서비스를 제공하는 데 도움이 되는지 확인합니다. 짧은 지연 시간에 대한 요구 사항이 있는 경우 AWS 로컬 영역이 사용자 근처에 있는지 확인합니다. 그렇다면 이를 사용하여 해당 사용자에게 더 가까운 위치에 워크로드를 배포합니다.
  - AWS 로컬 영역 FAQ

리소스

관련 문서:

- AWS 글로벌 인프라
- AWS 로컬 영역 FAQ
- Amazon ECS 태스크 배치 전략
- 리전 및 가용 영역 선택
- 예: 가용 영역 전반에 인스턴스 분산
- 전역 테이블: DynamoDB를 사용한 다중 리전 복제
- Amazon Aurora 글로벌 데이터베이스 사용
- Creating a Multi-Region Application with AWS Services(AWS 서비스로 다중 리전 애플리케이션 생성) 블로그 시리즈
- AWS Outposts란 무엇입니까?

관련 동영상:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications(다중 리전 액티브-액티브 애플리케이션을 위한 아키텍처 패턴)(ARC209-R2)
- AWS re:Invent 2019: Innovation and operation of the AWS global network infrastructure(AWS 글로벌 네트워크 인프라 혁신 및 운영)(NET339)

REL10-BP02 다중 위치 배포에 적합한 위치 선택

원하는 결과

고가용성을 위해서는 그림 10에 나온 것과 같이 가능하면 언제나 워크로드 구성 요소를 여러 개의 가용 영역(AZ)에 배포하세요. 복원력에 대한 요구 사항이 매우 높은 워크로드의 경우 다중 리전 아키텍처에 대한 옵션을 신중하게 평가하세요.

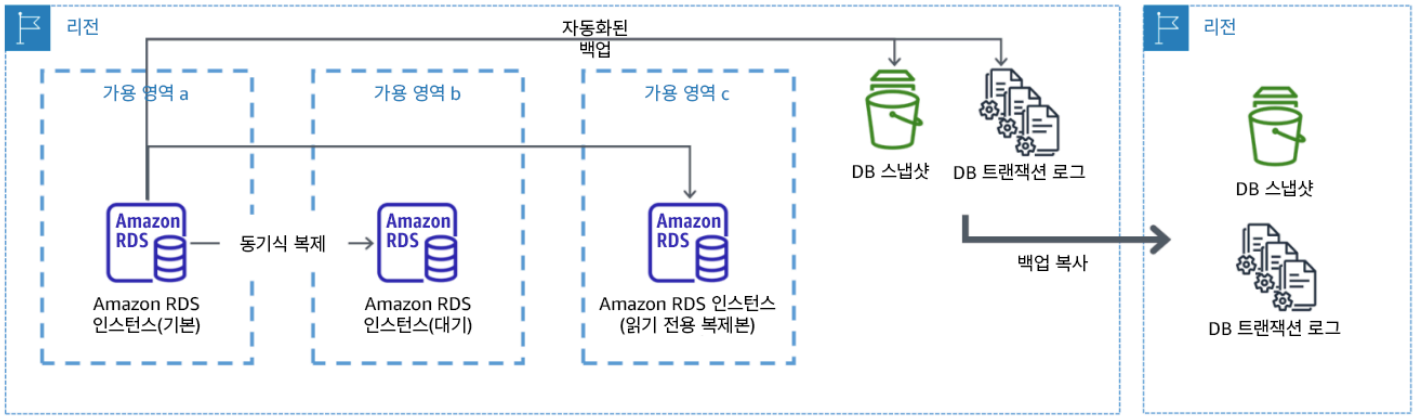


그림 10: 다른 AWS 리전에 백업되고 복원력이 있는 다중 AZ 데이터베이스 배포

일반적인 안티 패턴

- 다중 AZ 아키텍처로 요구 사항을 충족할 수 있는데 다중 리전 아키텍처로 설계함
- 복원력과 다중 위치 요구 사항이 구성 요소 간에 다를 경우 애플리케이션 구성 요소 간의 종속성을 고려하지 않음

이 모범 사례 수립의 이점

복원력을 위해서는 방어 계층을 구축하는 접근 방식을 사용해야 합니다. 하나의 계층은 다중 AZ를 사용하여 가용성이 높은 아키텍처를 구축함으로써 더 작고 더 일반적인 장애를 예방하고 또 다른 방어 계층은 만연한 자연 재해와 리전 수준의 장애와 같은 드문 이벤트를 예방하도록 설계합니다. 이 두 번째 계층에는 애플리케이션이 여러 AWS 리전에 분산되도록 하는 아키텍팅이 포함됩니다.

- 99.5% 가용성과 99.99% 가용성의 차이는 월간 3.5시간이 넘습니다. 워크로드가 여러 가용 영역에 있는 경우 워크로드의 예상 가용성은 최대 99.99%에 그치게 됩니다.
- 워크로드를 여러 가용 영역에서 실행하면 전력, 냉각기, 네트워킹에서의 장애와 화재나 홍수와 같은 대부분의 자연 재해로 인한 장애를 격리할 수 있습니다.
- 워크로드에 다중 리전 전략을 구현하면 한 나라의 광범위한 지역에 영향을 미치는 만연한 자연 재해나 리전 전체에 발생한 기술적 장애로부터 워크로드를 보호할 수 있습니다. 다중 리전 아키텍처를 구현하는 일은 매우 복잡할 수 있으며 대개 대부분의 워크로드에는 필요하지 않다는 점을 참고하시기 바랍니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

가용 영역 한 곳의 중단이나 일부 소실에 따른 재해 이벤트의 경우 단일 AWS 리전 내 다중 가용 영역에 고가용성 워크로드를 구현하면 자연 재해와 기술 재해의 위험을 완화하는 데 도움이 됩니다. 각 AWS 리전은 여러 가용 영역으로 구성되며, 각 가용 영역은 상당히 멀리 떨어진 다른 영역에 발생한 장애의 영향을 받지 않습니다. 그러나 서로 상당히 멀리 떨어진 여러 가용 영역 구성 요소가 소실될 위험이 있는 재해 이벤트의 경우 리전 전체에 영향을 주는 장애를 완화하는 재해 복구 옵션을 구현해야 합니다. 매우 높은 복원력을 요구하는 워크로드(핵심 인프라, 건강 관련 애플리케이션, 금융 시스템 인프라 등)의 경우 다중 리전 전략이 필요할 수 있습니다.

구현 단계

1. 워크로드를 평가하고 복원력에 대한 요구 사항을 다중 가용 영역 접근법(단일 AWS 리전)으로 충족할 수 있는지, 아니면 다중 리전 접근법이 필요한지 결정합니다. 이런 요구 사항을 충족하기 위해 다중 리전 아키텍처를 구현하면 복잡성이 가중되므로 사용 사례와 요구 사항을 신중하게 고려해야 합니다. 복원력 요구 사항은 단일 AWS 리전을 사용하여 대부분 충족할 수 있습니다. 여러 리전을 사용해야 하는지 결정할 때는 다음과 같은 요구 사항을 고려하세요.
   a. 재해 복구(DR): 가용 영역 한 곳의 중단이나 일부 소실에 따른 재해 이벤트의 경우 단일 AWS 리전 내 다중 가용 영역에 고가용성 워크로드를 구현하면 자연 재해와 기술 재해의 위험을 완화하는 데 도움이 됩니다. 서로 상당히 멀리 떨어진 여러 가용 영역 구성 요소가 소실될 위험이 있는 재해 이벤트의 경우, 자연 재해를 완화하거나 리전 전체에 영향을 주는 기술적 장애를 완화하는 여러 리전에 걸친 재해 복구를 구현해야 합니다.
   b. 고가용성(HA): 다중 리전 아키텍처(각 리전에 여러 개의 가용 영역 사용)를 사용하여 99.99% 이상의 가용성을 달성할 수 있습니다.
   c. 스택 현지화: 전 세계 사용자를 대상으로 워크로드를 배포할 때는 다양한 AWS 리전에 현지화된 스택을 배포하여 해당 리전의 사용자에게 제공할 수 있습니다. 언어, 통화, 저장되는 데이터의 유형이 현지화의 대상이 될 수 있습니다.
   d. 사용자에 대한 근접성: 전 세계 사용자에게 워크로드를 배포할 때는 최종 사용자에게 가까운 AWS 리전에 스택을 배포하여 지연 시간을 단축할 수 있습니다.
   e. 데이터 레지던시: 일부 워크로드에는 특정 사용자의 데이터가 특정 국가의 국경 내에 남아 있어야 한다는 데이터 레지던시 요구 사항이 적용됩니다. 해당 규제에 따라 해당 국경 내의 AWS 리전에 전체 스택을 배포할지, 데이터만 배포할지 선택할 수 있습니다.
2. AWS 서비스에서 제공하는 다중 AZ 기능의 예는 다음과 같습니다.
   a. EC2 또는 ECS를 사용하여 워크로드를 보호하기 위해 컴퓨팅 리소스 앞에 Elastic Load Balancer를 배포합니다. 그런 다음 Elastic Load Balancing가 비정상 영역의 인스턴스를 탐지하고 정상 영역으로 트래픽을 라우팅하는 솔루션을 제공합니다.

   i. Application Load Balancers 시작하기

   ii. Network Load Balancer 시작하기

b. 로드 밸런싱을 지원하지 않는 상용 기성품 소프트웨어를 실행하는 EC2 인스턴스의 경우 다중 AZ 재해 복구 방법론을 구현하여 일종의 내결함성을 달성할 수 있습니다.

   i. the section called “REL13-BP02 복구 목표 달성을 위해 정의된 복구 전략 사용”

c. Amazon ECS 태스크의 경우 세 개의 가용 영역에 균등하게 서비스를 배포하여 가용성과 비용의 균형을 달성합니다.

   i. Amazon ECS 가용성 모범 사례 | 컨테이너

d. Aurora Amazon RDS가 아닌 경우 구성 옵션으로 다중 AZ를 선택할 수 있습니다. 기본 데이터베이스 인스턴스가 실패하면 Amazon RDS가 자동으로 대기 데이터베이스를 승격하여 다른 가용 영역의 트래픽을 수신하도록 합니다. 다중 리전 읽기 전용 복제본도 복원력을 향상하기 위해 생성할 수 있습니다.

   i. Amazon RDS 다중 AZ 배포

   ii. 다른 AWS 리전에 읽기 전용 복제본 생성

3. AWS 서비스에서 제공하는 다중 리전 기능의 예는 다음과 같습니다.

a. 서비스에서 자동으로 다중 AZ 가용성이 제공되는 Amazon S3 워크로드의 경우 다중 리전 배포가 필요하다면 다중 리전 액세스 포인트를 고려하세요.

   i. Amazon S3에서의 다중 리전 액세스 포인트

b. 서비스에서 자동으로 다중 AZ 가용성이 제공되는 DynamoDB 테이블의 경우 기존 테이블을 글로벌 테이블로 쉽게 변환하여 다중 리전의 이점을 누릴 수 있습니다.

   i. 단일 리전 Amazon DynamoDB 테이블을 글로벌 테이블로 변환

c. 워크로드에 Application Load Balancers 또는 Network Load Balancer가 선행된다면 AWS Global Accelerator를 사용하여 정상 엔드포인트가 포함된 다중 리전으로 트래픽을 이동시킴으로써 애플리케이션의 가용성을 향상합니다.

   i. AWS Global Accelerator의 표준 액셀러레이터 엔드포인트 - AWS Global Accelerator(amazon.com)

d. AWS EventBridge를 사용하는 애플리케이션의 경우 교차 리전 버스를 사용하여 이벤트를 선택한 다른 리전에 전달하는 방법을 고려해 보세요.

   i. AWS 리전 간에 Amazon EventBridge 이벤트 전송 및 수신

e. Amazon Aurora 데이터베이스의 경우 여러 AWS 리전에 걸친 Aurora 글로벌 데이터베이스를 고려하세요. 기존 클러스터를 수정하여 새로운 리전을 추가할 수도 있습니다.

   i. Amazon Aurora 글로벌 데이터베이스 시작하기

f. 워크로드에 AWS Key Management Service(AWS KMS) 암호화 키가 포함되어 있는 경우 다중 리전 키가 애플리케이션에 적합한지 고려하세요.
   i. AWS KMS의 다중 리전 키
g. 다른 AWS 서비스 기능의 경우 AWS 서비스로 다중 리전 애플리케이션 생성에 대한 블로그 시리즈를 참고하세요.

구현 계획의 작업 수준: 보통에서 높음

리소스

관련 문서:

- AWS 서비스로 다중 리전 애플리케이션 생성 시리즈
- AWS의 재해 복구(DR) 아키텍처, 파트 IV: 다중 사이트 액티브/액티브
- AWS 글로벌 인프라
- AWS 로컬 영역 FAQ
- AWS의 재해 복구(DR) 아키텍처, 파트 I: 클라우드에서의 복구 전략
- 클라우드마다 다른 재해 복구
- 전역 테이블: DynamoDB를 사용한 다중 리전 복제

관련 동영상:

- AWS re:Invent 2018: 다중 리전 액티브-액티브 애플리케이션을 위한 아키텍처 패턴(ARC209-R2)
- Auth0: 자동 장애 조치를 통해 매월 15억 건 이상의 로그인으로 확장할 수 있는 다중 리전 고가용성 아키텍처 사용

관련 예시:

- AWS의 재해 복구(DR) 아키텍처, 파트 I: 클라우드에서의 복구 전략
- DTCC, 온프레미스에서 수행할 수 있는 수준 이상의 복원력 달성
- Expedia Group, 다중 리전 다중 가용 영역 아키텍처를 전용 DNS 서비스와 함께 사용하여 애플리케이션의 복원력 강화
- Uber: 다중 리전 Kafka용 재해 복구
- Netflix: 다중 리전 복원력을 위한 액티브/액티브 전략 사용

- Atlassian 클라우드를 위한 데이터 레지던시를 구축하는 방법
- Intuit TurboTax, 두 개의 리전에서 실행

REL10-BP03 단일 위치로 제약된 구성 요소의 복구 자동화

워크로드의 구성 요소를 단일 가용 영역 또는 온프레미스 데이터 센터에서만 실행해야 하는 경우 정의된 복구 목표 내에서 워크로드를 완전히 재구축할 수 있는 기능을 구현합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

기술적 제약으로 인해 워크로드를 여러 위치에 배포하는 모범 사례를 따를 수 없다면 복원력을 달성할 수 있는 대체 경로를 구현해야 합니다. 이러한 경우를 위해 필요한 인프라를 다시 생성하고, 애플리케이션을 다시 배포하고, 필요한 데이터를 다시 생성하는 기능을 자동화해야 합니다.

예를 들어 Amazon EMR은 지정된 클러스터의 모든 노드를 동일한 가용 영역에서 시작합니다. 동일한 영역에서 클러스터를 실행하면 데이터 접근 속도가 빨라져 작업 흐름의 성능이 개선되기 때문입니다. 워크로드 복원력에 이 구성 요소가 필요한 경우 클러스터와 해당 데이터를 다시 배포할 수 있어야 합니다. 또한 Amazon EMR의 경우 다중 AZ를 사용하는 것 이외의 방법으로 중복성을 프로비저닝해야 합니다. 다중 노드를 프로비저닝할 수 있습니다. EMRFS(EMR 파일 시스템)를 사용하면 EMR의 데이터를 Amazon S3에 저장할 수 있으며, 이는 다시 여러 가용 영역 또는 AWS 리전 리전에 걸쳐 복제할 수 있습니다.

마찬가지로 Amazon Redshift의 경우 기본적으로 사용자가 선택한 AWS 리전 내에서 임의로 선택된 가용 영역에 클러스터를 프로비저닝합니다. 모든 클러스터 노드는 동일한 영역에 프로비저닝됩니다.

온프레미스 데이터 센터에 배포된 상태 저장 서버 기반 워크로드의 경우 AWS Elastic Disaster Recovery를 사용하여 AWS에서 워크로드를 보호할 수 있습니다. 이미 AWS에서 호스팅되는 경우 Elastic Disaster Recovery를 사용하여 워크로드를 대체 가용 영역 또는 리전으로 보호할 수 있습니다. Elastic Disaster Recovery는 경량 스테이징 영역에 대한 지속적인 블록 수준 복제를 사용하여 온프레미스 및 클라우드 기반 애플리케이션의 빠르고 안정적인 복구를 제공합니다.

구현 단계

1. 자가 복구를 구현합니다. 가능한 경우 자동 크기 조정을 사용하여 인스턴스 또는 컨테이너를 배포합니다. 자동 크기 조정을 사용할 수 없는 경우 EC2 인스턴스에 대한 자동 복구를 사용하거나 Amazon EC2 또는 ECS 컨테이너 수명 주기 이벤트를 기반으로 자가 복구 자동화를 구현합니다.
   - 단일 인스턴스 IP 주소, 프라이빗 IP 주소, 탄력적 IP 주소 및 인스턴스 메타데이터가 필요하지 않은 인스턴스 및 컨테이너 워크로드에 Amazon EC2 Auto Scaling 그룹을 사용합니다.

    - 시작 템플릿 사용자 데이터는 대부분의 워크로드를 자가 복구할 수 있는 자동화를 구현하는 데 사용할 수 있습니다.
- 단일 인스턴스 ID 주소, 프라이빗 IP 주소, 탄력적 IP 주소 및 인스턴스 메타데이터가 필요한 워크로드에 Amazon EC2 인스턴스 자동 복구를 사용합니다.
    - 자동 복구는 인스턴스 장애가 감지될 때 SNS 주제로 복구 상태 알림을 전송합니다.
- 자동 크기 조정 또는 EC2 복구를 사용할 수 없는 경우 Amazon EC2 인스턴스 수명 주기 이벤트 또는 Amazon ECS 이벤트를 사용하여 자가 복구를 자동화합니다.
    - 이벤트를 사용하여 필요한 프로세스 로직에 따라 구성 요소를 복구하는 자동화를 호출합니다.
- AWS Elastic Disaster Recovery을 사용하여 단일 위치로 제한된 상태 저장 워크로드를 보호합니다.

리소스

관련 문서:

- Amazon ECS 이벤트
- Amazon EC2 Auto Scaling 수명 주기 후크
- 인스턴스 복구
- 서비스 자동 크기 조정
- Amazon EC2 Auto Scaling란 무엇입니까?
- AWS Elastic Disaster Recovery

REL10-BP04 격벽 아키텍처를 사용하여 영향 범위 제한

격벽 아키텍처(셀 기반 아키텍처라고도 함)를 구현하여 워크로드 내 장애의 영향을 제한된 수의 구성 요소로 제한합니다.

원하는 결과: 셀 기반 아키텍처는 워크로드의 여러 격리된 인스턴스를 사용하며 여기에서 각 인스턴스를 셀이라고 합니다. 각 셀은 독립적이고 다른 셀과 상태를 공유하지 않으며 전체 워크로드 요청의 하위 집합을 처리합니다. 이렇게 하면 잘못된 소프트웨어 업데이트와 같은 오류의 잠재적 영향이 개별 셀과 처리 중인 요청으로 축소됩니다. 워크로드가 10개의 셀을 사용하여 100개의 요청을 처리하는 경우 장애가 발생하면 전체 요청의 90%는 장애의 영향을 받지 않습니다.

일반적인 안티 패턴:

- 셀이 경계 없이 성장할 수 있도록 합니다.

- 동시에 모든 셀에 코드 업데이트 또는 배포를 적용합니다.
- 라우터 계층을 제외하고 셀 간에 상태 또는 구성 요소를 공유합니다.
- 복잡한 비즈니스 또는 라우팅 로직을 라우터 계층에 추가합니다.
- 셀 간 상호 작용을 최소화하지 않습니다.

이 모범 사례 확립의 이점: 셀 기반 아키텍처를 사용하면 많은 일반적인 유형의 오류가 셀 자체 내에 억제되어 추가적인 장애 격리를 제공합니다. 이러한 결함 경계는 실패한 코드 배포 또는 손상되거나 포이즌 필 요청이라고도 하는 특정 실패 모드를 트리거하는 요청과 같이 억제하기 어려운 실패 유형에 대한 복원력을 제공할 수 있습니다.

구현 가이드

선박의 격벽은 선체 파손이 선체의 한 섹션 내에 억제되도록 합니다. 복잡한 시스템에서 이 패턴은 장애 격리를 활성화하기 위해 종종 복제됩니다. 장애 격리 경계는 워크로드 내부 장애의 영향을 제한된 수의 구성 요소로 제한합니다. 경계 외부의 구성 요소는 장애가 발생하더라도 영향을 받지 않습니다. 다수의 장애 격리 경계를 사용하여 워크로드에 미치는 영향을 제한할 수 있습니다. AWS에서 고객은 여러 가용 영역 및 리전을 사용하여 장애 격리를 제공할 수 있지만 장애 격리의 개념은 워크로드의 아키텍처로도 확장될 수 있습니다.

전체 워크로드는 파티션 키로 분할된 셀입니다. 이 키는 서비스의 세부 수준 또는 최소한의 크로스 셀 상호 작용으로 서비스의 워크로드를 세분화할 수 있는 자연스러운 방식에 맞춰 조정되어야 합니다. 파티션 키로는 고객 ID, 리소스 ID 또는 대부분의 API 호출에서 쉽게 액세스할 수 있는 기타 파라미터를 예로 들 수 있습니다. 셀 라우팅 계층은 파티션 키를 기반으로 개별 셀에 요청을 배포하고 클라이언트에 단일 엔드포인트를 제공합니다.

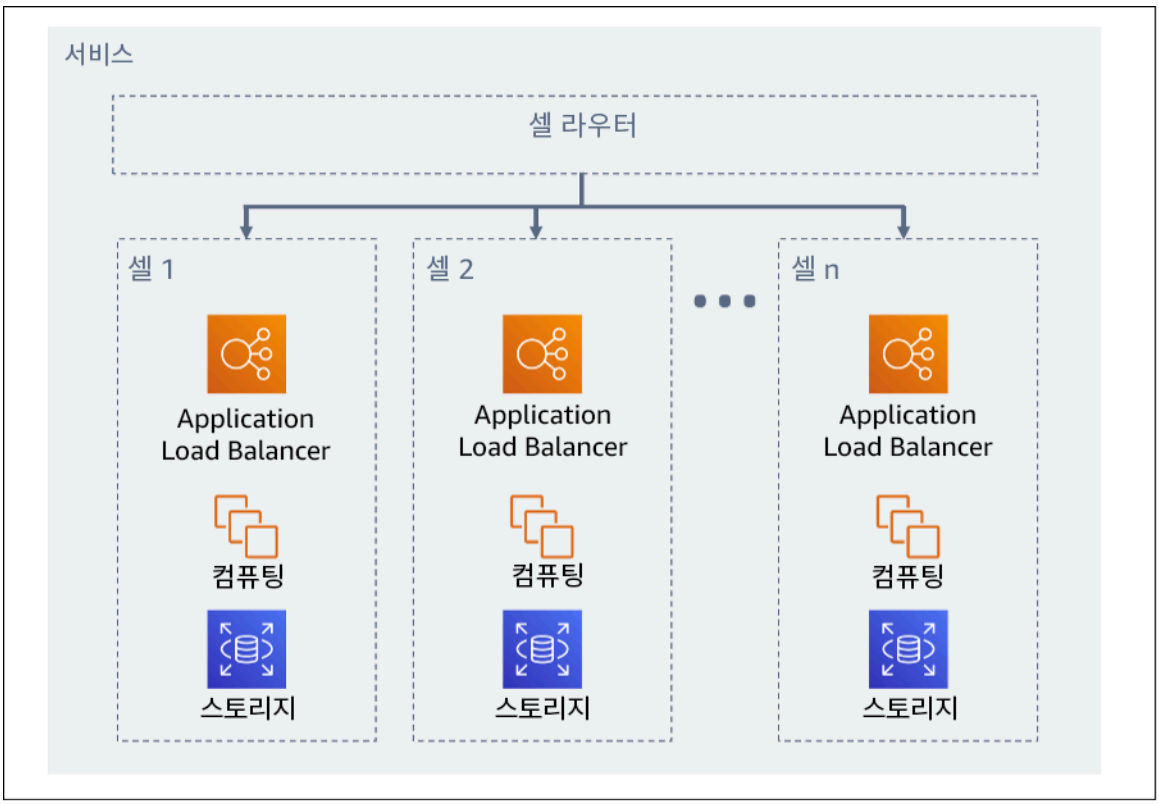


그림 11: 셀 기반 아키텍처

구현 단계

셀 기반 아키텍처를 설계할 때 고려해야 할 몇 가지 설계 고려 사항이 있습니다.

1. 파티션 키: 파티션 키를 선택할 때 다음을 특별히 고려해야 합니다.
   - 이 키는 서비스의 세부 수준 또는 최소한의 크로스 셀 상호 작용으로 서비스의 워크로드를 세분화할 수 있는 자연스러운 방식에 맞춰 조정되어야 합니다. 예를 들어 `## ID` 또는 `### ID`등이 있습니다.
   - 파티션 키는 모든 요청에서 직접 또는 다른 파라미터에 의해 결정론적으로 쉽게 추론될 수 있는 방식으로 사용 가능해야 합니다.
2. 영구 셀 매핑: 업스트림 서비스는 해당 리소스의 수명 주기 동안 단일 셀과만 상호 작용해야 합니다.
   - 워크로드에 따라 한 셀에서 다른 셀로 데이터를 마이그레이션하는 데 셀 마이그레이션 전략이 필요할 수 있습니다. 셀 마이그레이션이 필요할 수 있는 가능한 시나리오는 워크로드의 특정 사용자 또는 리소스가 너무 커져 전용 셀이 필요한 경우입니다.
   - 셀은 셀 간에 상태 또는 구성 요소를 공유해서는 안 됩니다.
   - 결과적으로 셀 간 상호 작용은 피하거나 최소로 유지해야 합니다. 이러한 상호 작용은 셀 간의 종속성을 생성하여 장애 격리 개선을 감소시키기 때문입니다.

3. 라우터 계층: 라우터 계층은 셀 간에 공유되는 구성 요소이므로 셀과 동일한 구획화 전략을 따를 수 없습니다.
    - 라우터 계층은 파티션 키를 셀에 매핑하기 위해 암호화 해시 함수와 모듈식 산술을 결합하는 것과 같이 컴퓨팅적으로 효율적인 방식으로 파티션 매핑 알고리즘을 사용하여 요청을 개별 셀에 배포하는 것이 좋습니다.
    - 다중 셀 영향을 방지하려면 라우팅 계층을 가능한 한 단순하고 수평적으로 확장할 수 있어야 하므로 이 계층 내에서 복잡한 비즈니스 로직을 피해야 합니다. 그러면 예상되는 동작을 항상 쉽게 이해할 수 있게 하여 철저한 테스트 가능성을 허용하는 추가적인 이점이 있습니다. Colm MacCárthaigh가 신뢰성, 지속적인 작업, 좋은 커피 한 잔에서 설명한 것처럼 단순한 디자인과 지속적인 작업 패턴은 신뢰할 수 있는 시스템을 생성하고 안티프래질을 줄입니다.
4. 세포 크기: 세포에는 최대 크기가 있어야 하며 그 이상으로 자라지 않아야 합니다.
    - 중단점에 도달하고 안전한 작동 마진이 설정될 때까지 철저한 테스트를 수행하여 최대 크기를 식별해야 합니다. 테스트 사례를 구현하는 방법에 대한 자세한 내용은 다음을 참조하세요. REL07-BP04 워크로드 로드 테스트
    - 전체 워크로드는 셀 추가를 통해 증가하므로, 수요 증가에 따라 워크로드를 확장할 수 있습니다.
5. 다중 AZ 또는 다중 리전 전략: 다양한 장애 도메인으로부터 보호하기 위해 여러 계층의 복원력을 활용해야 합니다.
    - 복원력을 위해서는 방어 계층을 구축하는 접근 방식을 사용해야 합니다. 하나의 계층은 다중 AZ를 사용하여 가용성이 높은 아키텍처를 구축함으로써 더 작고 더 일반적인 장애를 예방하고 또 다른 방어 계층은 만연한 자연 재해와 리전 수준의 장애와 같은 드문 이벤트를 예방하도록 설계합니다. 이 두 번째 계층에는 애플리케이션이 여러 AWS 리전에 분산되도록 하는 아키텍팅이 포함됩니다. 워크로드에 다중 리전 전략을 구현하면 한 나라의 광범위한 리전에 영향을 미치는 만연한 자연 재해나 리전 전체에 발생한 기술적 장애로부터 워크로드를 보호할 수 있습니다. 다중 리전 아키텍처를 구현하는 일은 매우 복잡할 수 있으며 대개 대부분의 워크로드에는 필요하지 않다는 점을 참고하시기 바랍니다. 자세한 내용은 REL10-BP02 다중 위치 배포에 적합한 위치 선택단축할 수 있습니다.
6. 코드 배포: 동시에 모든 셀에 코드 변경 사항을 배포하는 것보다 시차를 둔 코드 배포 전략을 선호해야 합니다.
    - 이렇게 하면 잘못된 배포 또는 사람의 실수로 인해 여러 셀에 미치는 잠재적인 장애를 최소화하는 데 도움이 됩니다. 자세한 내용은 안전한 핸드오프 배포 자동화를 참조하세요.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

리소스

관련 모범 사례:

- REL07-BP04 워크로드 로드 테스트
- REL10-BP02 다중 위치 배포에 적합한 위치 선택

관련 문서:

- Reliability, constant work, and a good cup of coffee(안정성, 지속적인 작업 및 맛 좋은 한 잔의 커피)
- AWS and Compartmentalization(AWS 및 구획화)
- 셔플 샤딩을 사용한 워크로드 격리
- 안전한 핸드오프 배포 자동화

관련 동영상:

- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small(루프 닫기 및 마음 열기: 규모에 상관없이 시스템을 제어하는 방법)
- AWS re:Invent 2018: How AWS Minimizes the Blast Radius of Failures(ARC338)(AWS가 장애의 영향 범위를 최소화하는 방법(ARC338))
- 셔플 샤딩: AWS re:Invent 2019: Introducing The Amazon Builders' Library(DOP328)(Amazon Builders' Library 소개(DOP328))
- AWS Summit ANZ 2021 - Everything fails, all the time: Designing for resilience(AWS Summit ANZ 2021 - 항상 실패하는 모든 것: 복원력을 위한 설계)

관련 예시:

- Well-Architected lab: Fault isolation with shuffle sharding(Well-Architected 실습: 셔플 샤딩으로 장애 격리)

REL 11 구성 요소 장애를 견디도록 워크로드를 설계하려면 어떻게 해야 합니까?

고가용성 및 낮은 MTTR(평균 복구 시간)이 요구되는 워크로드는 복원력을 고려하여 설계해야 합니다.

모범 사례

- REL11-BP01 워크로드의 모든 구성 요소를 모니터링하여 장애 감지

- REL11-BP02 정상 리소스로 장애 조치
- REL11-BP03 모든 계층에서 복구 자동화
- REL11-BP04 복구 중 컨트롤 플레인이 아닌 데이터 영역 사용
- REL11-BP05 정적 안정성을 사용하여 바이모달 동작 방지
- REL11-BP06 이벤트가 가용성에 영향을 미치는 경우 알림 전송
- REL11-BP07 가용성 목표 및 가동 시간 서비스 수준에 관한 계약(SLA)을 충족하도록 제품 설계

REL11-BP01 워크로드의 모든 구성 요소를 모니터링하여 장애 감지

워크로드 상태를 지속적으로 모니터링하여 성능 저하 또는 장애가 발생하는 즉시 수동 및 자동화된 시스템을 통해 이를 인식할 수 있도록 합니다. 비즈니스 가치를 기반으로 KPI(핵심 성능 지표)를 모니터링합니다.

모든 복구 메커니즘은 문제를 신속하게 탐지하는 기능에서 시작되어야 합니다. 기술적 장애를 먼저 감지하여 해결합니다. 그러나 가용성은 비즈니스 가치를 제공하는 워크로드의 기능에 따라 결정되므로 이 요구 사항을 측정하는 핵심 성과 지표(KPI)를 탐지 및 수정 전략의 핵심 척도로 사용해야 합니다.

일반적인 안티 패턴:

- 경보가 구성되지 않았기 때문에 알림 없이 중단이 발생합니다.
- 경보가 존재하지만 대응 시간이 충분하지 않은 임계치에 있습니다.
- 지표는 RTO(복구 시간 목표)를 충족하기에 충분한 지표가 수집되지 않는 경우가 많습니다.
- 워크로드의 고객용 계층만 능동적으로 모니터링됩니다.
- 기술 지표만 수집하며 비즈니스 기능 지표는 수집하지 않습니다.
- 워크로드의 사용자 경험을 측정하는 지표가 없습니다.

이 모범 사례 수립의 이점: 모든 계층에서 적절한 모니터링을 사용하면 감지 시간을 단축하여 복구 시간을 줄일 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 복구 목표에 따라 구성 요소의 수집 간격을 결정합니다.
  - 모니터링 간격은 필요한 복구 속도에 따라 달라집니다. 복구 시간은 복구에 걸리는 시간에 따라 결정되므로 이 시간과 RTO(복구 시간 목표)를 고려하여 수집 빈도를 결정해야 합니다.

- 구성 요소에 대한 세부 모니터링을 구성합니다.
  - EC2 인스턴스 및 Auto Scaling에 대한 세부 모니터링이 필요한지 결정합니다. 세부 모니터링은 1분 간격 지표를 제공하며 기본 모니터링은 5분 간격 지표를 제공합니다.
    - 인스턴스에 대한 세부 모니터링 활성화 또는 비활성화
    - Amazon CloudWatch를 사용하여 오토 스케일링 및 인스턴스 모니터링
  - RDS에 대한 향상된 모니터링이 필요한지 결정합니다. 향상된 모니터링은 RDS 인스턴스의 에이전트를 사용하여 RDS 인스턴스의 여러 프로세스 또는 스레드에 대한 유용한 정보를 가져옵니다.
    - Enhanced Monitoring
- 비즈니스 핵심 성과 지표(KPI)를 측정하는 사용자 지정 지표를 생성합니다. 워크로드는 주요 비즈니스 기능을 구현합니다. 이러한 기능은 간접 문제가 발생하는 시기를 식별하는 데 도움이 되는 KPI로 사용되어야 합니다.
  - 사용자 지정 지표 게시
- 사용자 canary를 사용하여 사용자 경험의 장애를 모니터링합니다. 가장 중요한 테스트 중 하나는 고객 행동을 실행하고 시뮬레이션할 수 있는 가상 트랜잭션 테스트('Canary 테스트'라고도 하지만 카나리 배포와는 다름)입니다. 다양한 원격 위치에서 워크로드 엔드포인트에 대해 이러한 테스트를 지속적으로 실행합니다.
  - Amazon CloudWatch Synthetics로 사용자 Canary 생성 지원
- 사용자 경험을 추적하는 사용자 지정 지표를 생성합니다. 고객의 경험을 계측할 수 있으면 소비자 경험이 저하되는 시기를 결정할 수 있습니다.
  - 사용자 지정 지표 게시
- 워크로드의 일부가 제대로 작동하지 않는 시기를 감지하고 리소스의 크기를 자동 조정해야 하는 시점을 알려주는 경보를 설정합니다. 경보를 사용하면 대시보드에 경보를 시각적으로 표시하고, Amazon SNS 또는 이메일을 통해 알림을 전송하며, Auto Scaling을 통해 워크로드의 리소스를 스케일 업 또는 다운할 수 있습니다.
  - Amazon CloudWatch 경보 사용
- 지표를 시각화하는 대시보드를 생성합니다. 대시보드를 사용하면 추세, 이상값 및 기타 잠재적 문제의 지표를 시각적으로 표시하거나, 조사가 필요할 수 있는 문제를 표시할 수 있습니다.
  - CloudWatch 대시보드 사용

리소스

관련 문서:

- Amazon CloudWatch Synthetics로 사용자 Canary 생성 지원

- 인스턴스에 대한 세부 모니터링 활성화 또는 비활성화
- Enhanced Monitoring
- Amazon CloudWatch를 사용하여 오토 스케일링 및 인스턴스 모니터링
- 사용자 지정 지표 게시
- Amazon CloudWatch 경보 사용
- CloudWatch 대시보드 사용

관련 예시:

- Well-Architected lab: Level 300: Implementing Health Checks and Managing Dependencies to Improve Reliability(Well-Architected 실습: 레벨 300: 상태 확인 구현 및 종속성 관리를 통한 안정성 개선)

REL11-BP02 정상 리소스로 장애 조치

리소스 장애가 발생할 경우 정상 리소스가 계속해서 요청을 처리할 수 있는지 확인합니다. 위치 장애(예: 가용 영역 또는 AWS 리전)의 경우, 손상되지 않은 위치의 정상 리소스로 장애 조치할 수 있는 시스템을 갖추고 있어야 합니다.

Elastic Load Balancing, AWS Auto Scaling 등의 AWS 서비스는 리소스와 가용 영역에 걸쳐 로드를 분산하도록 도와줍니다. 따라서 남아 있는 상태가 양호한 리소스로 트래픽을 이동시켜 개별 리소스(예: EC2 인스턴스)의 장애 또는 가용 영역의 장애를 완화할 수 있습니다. 다중 리전 워크로드의 경우 이 작업이 더 복잡합니다. 예를 들어 리전 간 읽기 전용 복제본을 사용하여 데이터를 여러 AWS 리전에 배포할 수 있지만, 장애 조치가 발생할 경우 읽기 전용 복제본을 기본으로 승격하고 트래픽을 해당 위치로 지정해야 합니다. Amazon Route 53 및 AWS Global Accelerator는 AWS 리전 전체에 트래픽을 라우팅하도록 도와줍니다.

워크로드에서 Amazon S3 또는 Amazon DynamoDB와 같은 AWS 서비스를 사용하는 경우 이러한 서비스는 여러 가용 영역에 자동으로 배포됩니다. 장애가 발생하면 AWS 컨트롤 플레인이 자동으로 정상적인 위치로 트래픽을 라우팅합니다. 데이터가 여러 가용 영역에 중복으로 저장되고 가용성이 유지됩니다. Amazon RDS의 경우 구성 옵션으로 다중 AZ를 선택해야 합니다. 그러면 장애 시 AWS가 자동으로 트래픽을 정상 인스턴스로 보냅니다. Amazon EC2 인스턴스, Amazon ECS 태스크 또는 Amazon EKS 포드의 경우에는 배포할 가용 영역을 선택합니다. 그러면 Elastic Load Balancing이 비정상 영역에서 인스턴스를 감지하고 정상적인 영역으로 트래픽을 라우팅합니다. Elastic Load Balancing을 사용하는 경우 온프레미스 데이터 센터의 구성 요소로 트래픽을 라우팅할 수도 있습니다.

다중 리전 접근 방식(온프레미스 데이터 센터도 포함될 수 있음)의 경우 Amazon Route 53를 사용하여 인터넷 도메인을 정의하고, 트래픽이 정상적인 리전으로 라우팅되도록 상태 확인을 포함하는 라우팅 정책을 할당할 수 있습니다. 또는 AWS Global Accelerator가 제공하는 정적 IP 주소를 애플리케이션의 고정된 진입점으로 사용하고 인터넷 대신 AWS 글로벌 네트워크를 사용하여 선택한 AWS 리전의 엔드포인트로 라우팅하여 성능과 신뢰성을 개선할 수 있습니다.

AWS에서는 장애 복구를 고려한 서비스 설계 방식이 사용됩니다. 즉, 장애에서 복구하는 시간과 데이터에 대한 영향을 최소화하는 방식으로 서비스가 설계됩니다. AWS 서비스는 기본적으로 한 리전 내의 여러 복제본에 저장된 요청만 승인하는 데이터 스토어를 사용합니다. 이러한 서비스와 리소스에는 Amazon Aurora, Amazon Relational Database Service(Amazon RDS) Multi-AZ DB 인스턴스, Amazon S3, Amazon DynamoDB, Amazon Simple Queue Service(Amazon SQS), Amazon Elastic File System(Amazon EFS) 등이 있습니다. 이러한 서비스/리소스는 셀 기반 격리와 가용 영역에서 제공되는 장애 격리 기능을 사용하도록 구성됩니다. AWS의 운영 절차에서는 자동화가 광범위하게 사용됩니다. 또한 서비스 중단 시 빠르게 복구할 수 있도록 교체 및 다시 시작 기능도 최적화됩니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 정상 리소스로 장애 조치합니다. 리소스 장애가 발생할 경우 정상 리소스가 계속해서 요청을 처리할 수 있는지 확인합니다. 위치 장애(예: 가용 영역 또는 AWS 리전)의 경우, 손상되지 않은 위치의 정상 리소스로 장애 조치할 수 있는 시스템을 갖추고 있어야 합니다.
  - 워크로드에서 Amazon S3 또는 Amazon DynamoDB와 같은 AWS 서비스를 사용하는 경우 이러한 서비스는 여러 가용 영역에 자동으로 배포됩니다. 장애가 발생하면 AWS 컨트롤 플레인이 자동으로 정상적인 위치로 트래픽을 라우팅합니다.
  - Amazon RDS의 경우 구성 옵션으로 다중 AZ를 선택해야 합니다. 그러면 장애 시 AWS가 자동으로 트래픽을 정상 인스턴스로 보냅니다.
    - Amazon RDS를 위한 고가용성(다중 AZ)
  - Amazon EC2 인스턴스 또는 Amazon ECS 태스크의 경우에는 배포할 가용 영역을 선택합니다. 그러면 Elastic Load Balancing이 비정상 영역에서 인스턴스를 감지하고 정상적인 영역으로 트래픽을 라우팅합니다. Elastic Load Balancing을 사용하는 경우 온프레미스 데이터 센터의 구성 요소로 트래픽을 라우팅할 수도 있습니다.
  - 다중 리전 접근 방식(온프레미스 데이터 센터도 포함될 수 있음)의 경우 정상적인 위치의 데이터와 리소스가 계속해서 요청을 처리할 수 있는지 확인합니다.
    - 예를 들어 리전 간 읽기 전용 복제본을 사용하여 데이터를 여러 AWS 리전에 배포할 수 있지만, 기본 위치에 장애가 발생할 경우 읽기 전용 복제본을 마스터로 승격하고 트래픽을 해당 위치로 지정해야 합니다.

    - Amazon RDS 읽기 전용 복제본 개요
- Amazon Route 53를 사용하여 인터넷 도메인을 정의하고 트래픽이 정상적인 리전으로 라우팅되도록 상태를 확인하는 등 라우팅 정책을 할당할 수 있습니다. 또는 AWS Global Accelerator가 제공하는 정적 IP 주소를 애플리케이션의 고정된 진입점으로 사용하고 퍼블릭 인터넷 대신 AWS 글로벌 네트워크를 사용하여 선택한 AWS 리전의 엔드포인트로 라우팅하여 성능과 신뢰성을 개선할 수 있습니다.
    - Amazon Route 53: 라우팅 정책 선택
    - AWS Global Accelerator란 무엇입니까?

리소스

관련 문서:

- APN 파트너: 내결함성 자동화를 지원할 수 있는 파트너
- AWS Marketplace: 내결함성에 사용할 수 있는 제품
- AWS OpsWorks: Using Auto Healing to Replace Failed Instances(AWS OpsWorks: 자동 복구를 사용하여 실패한 인스턴스 대체)
- Amazon Route 53: 라우팅 정책 선택
- Amazon RDS를 위한 고가용성(다중 AZ)
- Amazon RDS 읽기 전용 복제본 개요
- Amazon ECS 태스크 배치 전략
- Creating Kubernetes Auto Scaling Groups for Multiple Availability Zones(여러 가용 영역에 Kubernetes Auto Scaling 그룹 생성)
- AWS Global Accelerator란 무엇입니까?

관련 예시:

- Well-Architected lab: Level 300: Implementing Health Checks and Managing Dependencies to Improve Reliability(Well-Architected 실습: 레벨 300: 상태 확인 구현 및 종속성 관리를 통한 안정성 개선)

REL11-BP03 모든 계층에서 복구 자동화

장애가 감지되면 자동화된 기능을 사용하여 수정 작업을 수행합니다.

재시작 기능 은 장애를 해결하는 데 사용할 수 있는 중요한 도구입니다. 분산 시스템에서 모범 사례는 앞서 설명한 것처럼 가능한 경우 서비스를 상태 비저장으로 만드는 것입니다. 이렇게 하면 재시작 시 데이터 손실 또는 가용성이 손실되는 것을 방지할 수 있습니다. 클라우드에서는 재시작의 일부로 전체 리소스(예: EC2 인스턴스 또는 Lambda 함수)를 대체할 수 있으며 이러한 대체는 일반적으로 필수적입니다. 재시작은 그 자체로 장애를 복구할 수 있는 단순하면서도 안정적인 방법입니다. 워크로드에는 다양한 유형의 장애가 발생합니다. 장애는 하드웨어, 소프트웨어, 통신 및 작업 과정에서 발생할 수 있습니다. 서로 다른 유형의 장애를 각각 격리, 식별 및 수정하는 새로운 메커니즘을 구성하는 대신 여러 범주의 장애를 동일한 복구 전략에 매핑하는 것이 좋습니다. 인스턴스는 하드웨어 결함, 운영 체제 버그, 메모리 누수 또는 기타 원인으로 인해 장애를 경험할 수 있습니다. 이러한 장애가 발생할 경우 각 상황에 맞춰진 수정 조치를 구축하는 대신 인스턴스 장애로 처리하십시오. 인스턴스를 종료하고 AWS Auto Scaling을 통해 인스턴스를 교체합니다. 나중에 환경 외부에서 장애 발생 리소스 분석을 수행할 수 있습니다.

또 다른 예는 네트워크 요청을 다시 시작하는 기능입니다. 네트워크 시간 제한 장애와 종속성 장애(종속성이 오류를 반환함)에 대해 같은 복구 방식이 적용됩니다. 두 이벤트는 모두 시스템에 비슷한 영향을 주므로, 한 이벤트를 “특수 사례”로 처리하는 대신 지수 백오프 및 지터를 통해 제한적으로 재시도하는 비슷한 전략이 적용됩니다.

재시작 기능 은 복구 중심 컴퓨팅 및 고가용성 클러스터 아키텍처에 포함된 복구 메커니즘입니다.

Amazon EventBridge를 사용하면 CloudWatch 경보 또는 다른 AWS 서비스의 상태 변경과 같은 이벤트를 모니터링하고 필터링할 수 있습니다. 그런 다음 이벤트 정보를 기반으로 AWS Lambda, AWS Systems Manager Automation 또는 다른 대상을 트리거하여 워크로드에 대한 맞춤형 수정 로직을 실행할 수 있습니다.

EC2 인스턴스 상태를 확인하도록 Amazon EC2 Auto Scaling을 구성할 수 있습니다. 인스턴스가 실행 중 이외의 상태이거나 시스템 상태가 손상된 경우 Amazon EC2 Auto Scaling은 해당 인스턴스를 비정상으로 간주하고 대체 인스턴스를 시작합니다. AWS OpsWorks를 사용하는 경우 OpsWorks 계층 수준에서 EC2 인스턴스의 자동 복구 기능을 구성할 수 있습니다.

대규모 교체(예: 전체 가용 영역이 손실됨)의 경우 한 번에 여러 개의 새 리소스를 확보하는 대신 정적 안정성을 통해 고가용성을 유지하는 것이 좋습니다.

일반적인 안티 패턴:

- 인스턴스 또는 컨테이너에 개별적으로 애플리케이션 배포.
- 자동 복구를 사용하지 않고 여러 위치에 배포할 수 없는 애플리케이션을 배포.
- 자동 크기 조정 및 자동 복구로 복구하지 못한 애플리케이션을 수동으로 복구.

이 모범 사례 수립의 이점: 자동 복구는 워크로드를 한 번에 한 위치에만 배포할 수 있는 경우에도 평균 복구 시간을 단축하고 워크로드의 가용성을 보장합니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- Auto Scaling 그룹을 사용하여 워크로드에 계층을 배포합니다. Auto Scaling은 무상태 애플리케이션에서 자가 복구를 수행하고 용량을 추가 및 제거할 수 있습니다.
  - AWS Auto Scaling 작동 방식
- 여러 위치에 배포할 수 없고 장애 발생 시 재부팅이 허용되는 애플리케이션이 배포되어 있는 EC2 인스턴스에 자동 복구를 구현합니다. 애플리케이션을 여러 위치에 배포할 수 없는 경우 자동 복구를 사용하여 장애가 발생한 하드웨어를 교체하고 인스턴스를 다시 시작할 수 있습니다. 인스턴스 메타데이터 및 관련 IP 주소는 물론 Amazon EBS 볼륨과 Elastic File System 및 Lustre/Windows용 파일 시스템의 탑재 지점도 유지됩니다.
  - Amazon EC2 자동 복구
  - Amazon Elastic Block Store(Amazon EBS)
  - Amazon Elastic File System(Amazon EFS)
  - Amazon FSx for Lustre란 무엇입니까?
  - Amazon FSx for Windows File Server란 무엇입니까?
    - AWS OpsWorks를 사용하는 경우 계층 수준에서 EC2 인스턴스의 자동 복구 기능을 구성할 수 있습니다.
      - AWS OpsWorks: Using Auto Healing to Replace Failed Instances(AWS OpsWorks: 자동 복구를 사용하여 실패한 인스턴스 대체)
- 자동 크기 조정 또는 자동 복구를 사용할 수 없거나 자동 복구가 실패할 경우 AWS Step Functions 및 AWS Lambda를 사용하여 자동 복구를 구현합니다. 자동 크기 조정을 사용할 수 없고, 자동 복구를 사용할 수 없거나 자동 복구가 실패하는 경우 AWS Step Functions 및 AWS Lambda를 사용하여 복구를 자동화할 수 있습니다.
  - AWS Step Functions란 무엇입니까?
  - AWS Lambda란 무엇입니까?
    - Amazon EventBridge를 사용하면 CloudWatch 경보 또는 다른 AWS 서비스의 상태 변경과 같은 이벤트를 모니터링하고 필터링할 수 있습니다. 그런 다음 이벤트 정보를 기반으로 AWS Lambda(또는 다른 대상)를 트리거하여 워크로드에 대한 사용자 지정 수정 로직을 실행할 수 있습니다.
      - Amazon EventBridge란 무엇입니까?

- Amazon CloudWatch 경보 사용

리소스

관련 문서:

- APN 파트너: 내결함성 자동화를 지원할 수 있는 파트너
- AWS Marketplace: 내결함성에 사용할 수 있는 제품
- AWS OpsWorks: Using Auto Healing to Replace Failed Instances(AWS OpsWorks: 자동 복구를 사용하여 실패한 인스턴스 대체)
- Amazon EC2 자동 복구
- Amazon Elastic Block Store(Amazon EBS)
- Amazon Elastic File System(Amazon EFS)
- AWS Auto Scaling 작동 방식
- Amazon CloudWatch 경보 사용
- Amazon EventBridge란 무엇입니까?
- AWS Lambda란 무엇입니까?
- AWS Systems Manager Automation
- AWS Step Functions란 무엇입니까?
- Amazon FSx for Lustre란 무엇입니까?
- Amazon FSx for Windows File Server란 무엇입니까?

관련 동영상:

- AWS의 정적 안정성: AWS re:Invent 2019: Introducing The Amazon Builders' Library(Amazon Builders' Library 소개)(DOP328)

관련 예시:

- Well-Architected lab: Level 300: Implementing Health Checks and Managing Dependencies to Improve Reliability(Well-Architected 실습: 레벨 300: 상태 확인 구현 및 종속성 관리를 통한 안정성 개선)

REL11-BP04 복구 중 컨트롤 플레인이 아닌 데이터 영역 사용

제어 영역은 리소스를 구성하는 데 사용되며, 데이터 영역은 서비스 제공에 사용됩니다. 일반적으로 데이터 영역은 제어 영역보다 가용성 설계 목표가 더 높으며, 일반적으로 덜 복잡합니다. 복원력에 영향을 미칠 가능성이 있는 이벤트에 대해 복구 또는 완화 대응을 구현할 때 제어 영역 작업을 사용하면 아키텍처의 전반적인 복원력이 감소될 수 있습니다. 예를 들어, Amazon Route 53 데이터 영역을 사용하면 상태 확인을 기반으로 DNS 쿼리를 안정적으로 라우팅할 수 있지만, Route 53 라우팅 정책을 업데이트할 때 컨트롤 플레인을 사용하므로 복구에 사용할 수 없습니다.

Route 53 데이터 영역은 DNS 쿼리에 응답하고 상태 확인을 수행 및 평가합니다. Route 53 데이터 영역은 전역에 배포되며 100% 가용성 서비스 수준에 관한 계약(SLA)을 위해 설계됩니다. Route 53 리소스를 생성, 업데이트, 삭제하는 데 사용하는 Route 53 관리 API 및 콘솔은 DNS를 관리할 때 필요한 강력한 일관성과 내구성을 우선시하도록 설계된 컨트롤 플레인에서 실행됩니다. 이를 위해 컨트롤 플레인은 하나의 리전(US East (N. Virginia))에 위치합니다. 두 시스템 모두 안정성이 높게 구축되었지만 컨트롤 플레인은 SLA에 포함되지 않습니다. 데이터 영역의 복원력 높은 설계가 컨트롤 플레인과 달리 가용성을 유지하는 상황이 드물게 있을 수 있습니다. 재해 복구 및 장애 조치 메커니즘에는 데이터 영역 기능을 사용하여 가능한 한 최고 수준의 신뢰성을 제공하십시오.

데이터 영역, 컨트롤 플레인, 고가용성 목표를 충족하기 위해 AWS에서 서비스를 구축하는 방법에 관한 자세한 내용은 가용 영역을 사용한 정적 안정성 문서와 Amazon Builders' Library를 참조하십시오.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 재해 복구에 Amazon Route 53를 사용할 때 컨트롤 플레인이 아닌 데이터 영역을 사용합니다. Route 53 Application Recovery Controller를 사용하면 준비 상태 확인 및 라우팅 제어를 통해 장애 조치를 관리하고 조정할 수 있습니다. 이러한 기능은 애플리케이션의 장애 복구 성능을 지속적으로 모니터링하며, 이를 통해 여러 AWS 리전, 가용 영역 및 온프레미스에서 애플리케이션 복구를 제어할 수 있습니다.
  - What is Route 53 Application Recovery Controller?(Amazon Route 53 Application Recovery Controller란 무엇입니까?)
  - Creating Disaster Recovery Mechanisms Using Amazon Route 53(Amazon Route 53를 사용하여 재해 복구 메커니즘 생성)
  - Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 1: Single-Region stack(Amazon Route 53 Application Recovery Controller를 사용하여 복원력이 높은 애플리케이션 구축, 파트 1: 단일 리전 스택)

    - Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 2: Multi-Region stack(Amazon Route 53 Application Recovery Controller를 사용하여 복원력이 높은 애플리케이션 구축, 파트 2: 다중 리전 스택)
- 데이터 영역을 사용할 작업과 컨트롤 플레인을 사용할 작업을 파악합니다.
    - Amazon Builders' Library: Avoiding overload in distributed systems by putting the smaller service in control(소규모 서비스 제어를 통한 분산 시스템 오버로드 방지)
    - Amazon DynamoDB API(컨트롤 플레인 및 데이터 영역)
    - AWS Lambda 실행 (컨트롤 플레인 및 데이터 영역으로 분할)
    - AWS Lambda 실행 (컨트롤 플레인 및 데이터 영역으로 분할)

리소스

관련 문서:

- APN 파트너: 내결함성 자동화를 지원할 수 있는 파트너
- AWS Marketplace: 내결함성에 사용할 수 있는 제품
- Amazon Builders' Library: Avoiding overload in distributed systems by putting the smaller service in control(소규모 서비스 제어를 통한 분산 시스템 오버로드 방지)
- Amazon DynamoDB API(컨트롤 플레인 및 데이터 영역)
- AWS Lambda 실행 (컨트롤 플레인 및 데이터 영역으로 분할)
- AWS Elemental MediaStore Data Plane(AWS Elemental MediaStore 데이터 영역)
- Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 1: Single-Region stack(Amazon Route 53 Application Recovery Controller를 사용하여 복원력이 높은 애플리케이션 구축, 파트 1: 단일 리전 스택)
- Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 2: Multi-Region stack(Amazon Route 53 Application Recovery Controller를 사용하여 복원력이 높은 애플리케이션 구축, 파트 2: 다중 리전 스택)
- Creating Disaster Recovery Mechanisms Using Amazon Route 53(Amazon Route 53를 사용하여 재해 복구 메커니즘 생성)
- What is Route 53 Application Recovery Controller?(Amazon Route 53 Application Recovery Controller란 무엇입니까?)

관련 예시:

- Amazon Route 53 Application Recovery Controller 소개

REL11-BP05 정적 안정성을 사용하여 바이모달 동작 방지

바이모달 동작은 워크로드가 정상 모드와 장애 모드에서 다른 동작을 보이는 것을 말합니다. 예를 들어 가용 영역에 장애가 발생할 경우 새 인스턴스를 시작하는 방법을 사용할 수 있습니다. 그러나 이 방법 대신 정적으로 안정적이며 한 모드에서만 작동하는 워크로드를 구축해야 합니다. 한 AZ가 제거된 경우 제거된 영역의 워크로드 로드를 처리하기에 충분한 인스턴스를 각 가용 영역에 프로비저닝한 다음 Elastic Load Balancing 또는 Amazon Route 53 상태 확인을 사용하여 손상된 인스턴스에서 로드를 이동합니다.

컴퓨팅 배포(예: EC2 인스턴스 또는 컨테이너)에서 정적 안정성은 최고의 안정성을 제공합니다. 정적 안정성은 비용 문제와 비교하여 검토되어야 합니다. 컴퓨팅 파워를 적게 프로비저닝하고 장애 발생 시 새 인스턴스를 시작하는 방법이 비용은 더 저렴합니다. 그러나 대규모 장애(예: 가용 영역 장애)의 경우 이 접근 방식은 장애가 발생하기 전에 대비하는 것이 아니라 장애가 발생할 때 대응하는 방법에 의존하기 때문에 효율성이 떨어집니다. 따라서 워크로드의 안정성 요구 사항과 비용 요구 사항을 비교해야 합니다. 사용하는 가용 영역이 많을수록 정적 안정성을 유지하는 데 필요한 추가 컴퓨팅 용량은 줄어듭니다.

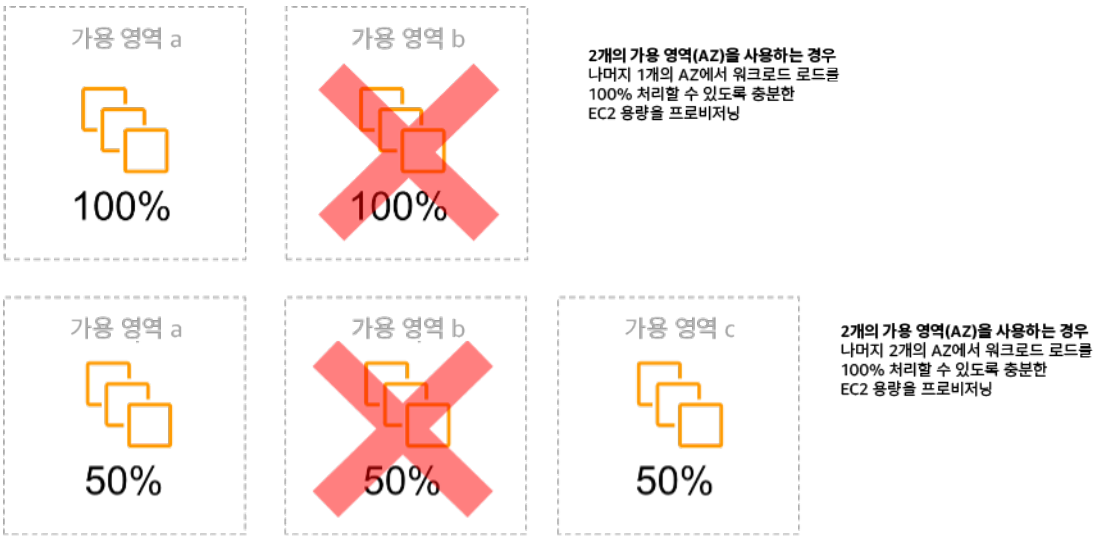


그림 14: 가용 영역에 걸친 EC2 인스턴스의 정적 안정성

트래픽이 전환된 후에는 AWS Auto Scaling을 사용하여 장애가 발생한 영역의 인스턴스를 비동기식으로 교체하고 정상 영역에서 인스턴스를 시작합니다.

바이모달 동작의 또 다른 예는 시스템이 전체 시스템의 구성 상태를 새로 고치려고 시도할 수 있는 네트워크 시간 제한입니다. 그러면 다른 구성 요소에 예기치 않은 로드가 더해져 해당 구성 요소에 장애가 발생하고 또 다른 예기치 않은 결과가 야기될 수 있습니다. 이 부정적인 피드백 루프는 워크로드의

가용성에 영향을 미칩니다. 대신 정적으로 안정적이며 한 모드에서만 작동하는 시스템을 구축해야 합니다. 일정한 작업을 수행하고 항상 고정된 케이던스에서 구성 상태를 새로 고치는 것이 정적으로 안정된 설계의 하나일 수 있습니다. 호출이 실패하면 워크로드는 이전에 캐시된 값을 사용하고 경보를 트리거합니다.

바이모달 동작의 또 다른 예로 장애 발생 시 클라이언트에서 워크로드 캐시를 우회하는 것을 허용하는 동작이 있습니다. 이는 클라이언트 요구 사항을 수용한 솔루션처럼 보이지만 워크로드의 수요가 크게 변경되고 장애를 초래할 가능성이 높으므로 허용해서는 안 됩니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 정적 안정성을 사용하여 바이모달 동작을 방지합니다. 바이모달 동작은 워크로드가 정상 모드와 장애 모드에서 다른 동작을 보이는 것을 말합니다. 예를 들어 가용 영역에 장애가 발생할 경우 새 인스턴스를 시작하는 방법을 사용할 수 있습니다.
  - Minimizing Dependencies in a Disaster Recovery Plan(재해 복구 계획의 종속성 최소화)
  - Amazon Builders' Library: 가용 영역을 사용한 정적 안정성
  - AWS의 정적 안정성: AWS re:Invent 2019: Amazon Builders’ Library 소개(DOP328)
    - 그러나 이 방법 대신 정적으로 안정적이고 한 모드에서만 작동하는 시스템을 구축해야 합니다. 한 가용 영역이 제거된 경우 제거된 영역의 워크로드 로드를 처리하기에 충분한 인스턴스를 각 영역에 프로비저닝한 다음 Elastic Load Balancing 또는 Amazon Route 53 상태 확인을 사용하여 손상된 인스턴스에서 로드를 이동합니다.
    - 바이모달 동작의 또 다른 예로 장애 발생 시 클라이언트에서 워크로드 캐시를 우회하는 것을 허용하는 동작이 있습니다. 이는 클라이언트 요구 사항을 수용하는 솔루션처럼 보이지만 워크로드의 수요가 크게 변경되고 장애를 초래할 가능성이 높으므로 허용해서는 안 됩니다.

리소스

관련 문서:

- Minimizing Dependencies in a Disaster Recovery Plan(재해 복구 계획의 종속성 최소화)
- Amazon Builders' Library: 가용 영역을 사용한 정적 안정성

관련 동영상:

- AWS의 정적 안정성: AWS re:Invent 2019: Amazon Builders’ Library 소개(DOP328)

REL11-BP06 이벤트가 가용성에 영향을 미치는 경우 알림 전송

중대한 이벤트가 감지되면 이벤트로 인해 야기된 문제가 자동으로 해결된 경우에도 알림이 전송됩니다.

자동 복구를 사용하면 워크로드의 안정성을 유지할 수 있습니다. 그러나 자동 복구로 인해 해결해야 할 근본적인 문제가 가려질 수도 있습니다. 적절한 모니터링 및 이벤트를 구현하면 자동 복구로 해결된 문제를 포함한 문제의 패턴을 감지하여 근본 원인 문제를 해결할 수 있습니다. 장애 발생을 기준으로 Amazon CloudWatch Alarms를 트리거할 수 있으며 자동화된 복구 작업의 실행을 기준으로 트리거할 수도 있습니다. 이메일을 보내거나 Amazon SNS 통합을 사용하여 서드파티 인시던트 추적 시스템에 인시던트를 기록하도록 CloudWatch Alarms를 구성할 수 있습니다.

일반적인 안티 패턴:

- 누구도 조치를 취하지 않는 경보 전송
- 자동 복구 자동화를 수행하지만 복구가 필요한 것을 알리지 않음.

이 모범 사례 수립의 이점: 복구 이벤트에 대한 알림이 전송되면 자주 발생하는 문제가 무시되지 않습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 비즈니스 핵심 성과 지표(KPI)가 낮은 임계값을 초과할 때 이러한 지표에 대한 경보를 전송합니다. 비즈니스 KPI에 대해 낮은 임계값 경보를 설정하면 워크로드를 사용할 수 없거나 작동하지 않는 시기를 파악하는 데 도움이 됩니다.
  - 정적 임계값을 기반으로 CloudWatch 경보 생성
- 복구 자동화를 호출하는 이벤트에 대한 경보 SNS API를 직접 호출하여 생성한 자동화를 통해 알림을 보낼 수 있습니다.
  - Amazon Simple Notification Service란 무엇입니까?

리소스

관련 문서:

- 정적 임계값을 기반으로 CloudWatch 경보 생성
- Amazon EventBridge란 무엇입니까?

- Amazon Simple Notification Service란 무엇입니까?

REL11-BP07 가용성 목표 및 가동 시간 서비스 수준에 관한 계약(SLA)을 충족하도록 제품 설계

가용성 목표 및 가동 시간 서비스 수준에 관한 계약(SLA)를 충족하도록 제품을 설계합니다. 가용성 목표 또는 가동 시간 SLA를 게시하거나 비공개로 동의하는 경우 아키텍처 및 운영 프로세스가 이를 지원하도록 설계되었는지 확인합니다.

원하는 결과: 각 애플리케이션에는 비즈니스 성과를 달성하기 위해 모니터링 및 유지 관리할 수 있는 성능 지표에 대해 정의된 가용성 및 SLA 목표가 있습니다.

일반적인 안티 패턴:

- SLA를 설정하지 않고 워크로드를 설계 및 배포합니다.
- SLA 지표는 근거나 비즈니스 요구 사항 없이 높게 설정됩니다.
- 종속성 및 기본 SLA를 고려하지 않고 SLA를 설정합니다.
- 애플리케이션 설계는 복원력에 대한 공동 책임 모델을 고려하지 않고 생성됩니다.

이 모범 사례 확립의 이점: 주요 복원력 목표를 기반으로 애플리케이션을 설계하면 비즈니스 목표와 고객 기대치를 충족하는 데 도움이 됩니다. 이러한 목표는 다양한 기술을 평가하고 다양한 장단점을 고려하는 애플리케이션 설계 프로세스를 추진하는 데 도움이 됩니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

애플리케이션 설계는 비즈니스, 운영 및 재무 목표에서 파생된 다양한 요구 사항 집합을 고려해야 합니다. 운영 요구 사항 내에서 워크로드는 적절하게 모니터링되고 지원될 수 있도록 특정 복원력 지표 대상을 가져야 합니다. 복원력 지표는 워크로드를 배포한 후에 설정하거나 파생해서는 안 됩니다. 설계 단계에서 정의해야 하며 다양한 결정과 장단점을 안내하는 데 도움이 됩니다.

- 모든 워크로드에는 고유한 복원력 지표 세트가 있어야 합니다. 이러한 지표는 다른 비즈니스 애플리케이션과 다를 수 있습니다.
- 종속성을 줄이면 가용성에 긍정적인 영향을 미칠 수 있습니다. 각 워크로드는 종속성과 해당 SLA를 고려해야 합니다. 일반적으로 가용성 목표가 워크로드 목표 이상인 종속성을 선택합니다.
- 가능한 경우 종속성 손상에도 불구하고 워크로드가 올바르게 작동할 수 있도록 느슨하게 결합된 설계를 고려하세요.

- 특히 복구 또는 성능 저하 중에 컨트롤 플레인 종속성을 줄입니다. 미션 크리티컬 워크로드에 대해 정적으로 안정적인 설계를 평가합니다. 리소스 스페어링을 사용하여 워크로드에서 이러한 종속성의 가용성을 높입니다.
- 평균 탐지 시간(MTTD 및 평균 복구 시간(MTTR)을 줄임으로써 SLA를 달성하기 위해서는 관측성과 계측이 중요합니다.
- 장애 빈도 감소(MTBF 증가), 장애 탐지 시간 단축(MTTD 감소) 및 복구 시간 단축(MTTR 감소)은 분산 시스템에서 가용성을 개선하는 데 사용되는 세 가지 요소입니다.
- 워크로드에 대한 복원력 지표를 설정하고 충족하는 것은 모든 효과적인 설계의 기초입니다. 이러한 설계는 설계 복잡성, 서비스 종속성, 성능, 확장성 및 비용의 균형을 고려해야 합니다.

구현 단계

- 다음 질문을 고려하여 워크로드 설계를 검토하고 문서화합니다.
  - 워크로드에서 컨트롤 플레인은 어디에 사용됩니까?
  - 워크로드는 내결함성을 어떻게 구현합니까?
  - 확장, 자동 확장, 중복성 및 고가용성 구성 요소에 대한 디자인 패턴은 무엇입니까?
  - 데이터 일관성 및 가용성에 대한 요구 사항은 무엇입니까?
  - 리소스 절약 또는 리소스 정적 안정성에 대한 고려 사항이 있습니까?
  - 서비스 종속성은 무엇입니까?
- 이해관계자와 협력하면서 워크로드 아키텍처를 기반으로 SLA 지표를 정의합니다. 워크로드에서 사용하는 모든 종속성의 SLA를 고려합니다.
- SLA 목표가 설정되면 SLA를 충족하도록 아키텍처를 최적화합니다.
- SLA를 충족하는 설계가 설정되면 운영 변경, 프로세스 자동화 및 MTTD 및 MTTR 감소에 중점을 둔 런북을 구현합니다.
- 배포되면 SLA를 모니터링하고 보고합니다.

리소스

관련 모범 사례:

- REL03-BP01 워크로드를 세그먼트화하는 방법 선택
- REL10-BP01 워크로드를 여러 위치에 배포
- REL11-BP01 워크로드의 모든 구성 요소를 모니터링하여 장애 감지
- REL11-BP03 모든 계층에서 복구 자동화

- REL12-BP05 카오스 엔지니어링을 이용한 복원력 테스트
- REL13-BP01 가동 중단 시간 및 데이터 손실 시의 복구 목표 정의
- 워크로드 상태 파악 파악

관련 문서:

- Availability with redundancy(중복성이 적용된 가용성)
- 신뢰성 원칙 백서 - 가용성
- Measuring availability(가용성 측정)
- AWS Fault Isolation Boundaries(AWS 장애 격리 경계)
- Shared Responsibility Model for Resiliency (복원력을 위한 공동 책임 모델)
- 가용 영역을 사용한 정적 안정성
- AWS 서비스 수준에 관한 계약(SLA)
- Guidance for Cell-based Architecture on AWS(AWS의 셀 기반 아키텍처에 대한 지침)
- AWS infrastructure(AWS 인프라)
- Advanced Multi-AZ Resiliance Patterns whitepaper(고급 다중 AZ 복원력 패턴 백서)

관련 서비스:

- Amazon CloudWatch
- AWS Config
- AWS Trusted Advisor

REL 12 안정성은 어떻게 테스트합니까?

프로덕션 환경의 스트레스에 대한 복원력을 가지도록 워크로드를 설계한 후 설계대로 작동하고 예상한 복원력을 제공하는지 확인할 수 있는 유일한 방법은 테스트입니다.

모범 사례

- REL12-BP01 플레이북을 사용하여 장애 조사
- REL12-BP02 인시던트 사후 분석 수행
- REL12-BP03 기능 요구 사항 테스트
- REL12-BP04 확장 및 성능 요구 사항 테스트

- REL12-BP05 카오스 엔지니어링을 이용한 복원력 테스트
- REL12-BP06 정기적으로 게임 데이 진행

REL12-BP01 플레이북을 사용하여 장애 조사

잘 알려지지 않은 장애 시나리오에 일관되고 신속하게 대응할 수 있도록 플레이북에 조사 프로세스를 문서화합니다. 플레이북은 장애 시나리오에 영향을 미치는 요인을 식별하기 위해 수행되는 미리 정의된 단계입니다. 문제가 확인되거나 에스컬레이션될 때까지 각 프로세스 단계의 결과를 사용하여 다음에 수행할 단계를 결정합니다.

플레이북은 사후 대응적 조치를 효과적으로 수행하기 위해 반드시 수행해야 하는 사전 예방적 계획입니다. 플레이북에 포함되지 않은 장애 시나리오가 프로덕션 환경에서 발생할 경우 이 문제를 먼저 해결합니다(화재 진압). 그런 다음 돌아가서 문제를 해결하기 위해 취한 단계를 살펴보고 이를 사용하여 플레이북에 새 항목을 추가합니다.

플레이북은 특정 인시던트에 대응하여 사용되며 런북은 특정 결과를 달성하기 위해 사용됩니다. 런북은 일상적인 활동에 대해 사용되고, 플레이북은 일상적이지 않은 이벤트에 대응하는 데 사용되는 경우가 많습니다.

일반적인 안티 패턴:

- 문제를 진단하거나 인시던트에 대응하는 프로세스를 모른 상태에서 워크로드 배포를 계획합니다.
- 이벤트를 조사할 때 로그 및 지표를 수집할 시스템에 대한 무계획적 결정
- 데이터를 검색할 수 있을 만큼 오래 지표 및 이벤트를 유지하지 않음

이 모범 사례 수립의 이점: 플레이북을 캡처하면 프로세스를 일관되게 따를 수 있습니다. 플레이북을 코드화하면 수작업으로 인한 오류 발생을 최소화할 수 있습니다. 플레이북을 자동화하면 팀원의 개입 요구 사항을 없애거나 개입을 시작할 때 추가 정보를 제공함으로써 이벤트에 대응하는 시간을 단축할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 플레이북을 사용하여 문제를 파악합니다. 플레이북은 문제 조사를 위한 문서화된 프로세스입니다. 플레이북에 프로세스를 문서화하면 장애 발생 시나리오에 일관되고 빠르게 대응할 수 있습니다. 플레이북은 적절한 기술을 보유한 팀원이 해당하는 정보를 수집하고, 장애의 잠재적 출처를 확인하고, 결함 위치를 구분하고, 발생 요인을 확인(인시던트 사후 분석 수행)하는 데 필요한 정보와 지침을 포함해야 합니다.

- 코드로 플레이북을 구현합니다. 플레이북을 스크립트로 작성하여 작업을 코드로 수행하면 일관성을 유지하고 수동 프로세스에서 발생하는 오류를 최소화할 수 있습니다. 플레이북은 문제의 발생 요인을 식별하는 데 필요할 수 있는 다양한 단계를 나타내는 여러 스크립트로 구성할 수 있습니다. 플레이북 활동의 일부분으로 런북 활동을 트리거하거나 수행할 수도 있고, 확인된 이벤트 대응 과정에서 플레이북 실행 여부를 묻는 메시지를 표시할 수도 있습니다.
  - AWS Systems Manager를 사용하여 운영 플레이북 자동화
  - AWS Systems Manager Run Command
  - AWS Systems Manager Automation
  - AWS Lambda란 무엇입니까?
  - Amazon EventBridge란 무엇입니까?
  - Amazon CloudWatch 경보 사용

리소스

관련 문서:

- AWS Systems Manager Automation
- AWS Systems Manager Run Command
- AWS Systems Manager를 사용하여 운영 플레이북 자동화
- Amazon CloudWatch 경보 사용
- Canary 사용(Amazon CloudWatch Synthetics)
- Amazon EventBridge란 무엇입니까?
- AWS Lambda란 무엇입니까?

관련 예시:

- 플레이북 및 런북으로 운영 자동화

REL12-BP02 인시던트 사후 분석 수행

고객에게 영향을 주는 이벤트를 검토하고 발생 요인과 예방 조치 항목을 식별합니다. 이 정보를 사용하여 재발을 제한하거나 방지하는 완화 기능을 개발합니다. 신속하고 효과적인 대응을 위한 절차를 개발합니다. 목표 대상에 맞게 적절히 발생 요인과 수정 조치를 전달합니다. 필요한 경우 다른 관계자들에게 이러한 원인을 전달하는 방법을 마련합니다.

기존 테스트에서 문제가 발견되지 않은 이유를 평가합니다. 테스트가 아직 없는 경우 이 사례에 대한 테스트를 추가합니다.

일반적인 안티 패턴:

- 발생 요인을 찾지만, 다른 잠재적 문제와 해결 방법을 계속 자세히 살펴보지는 않음
- 인적 오류 원인만 식별하며, 인적 오류를 예방할 수 있는 교육이나 자동화는 제공하지 않음

이 모범 사례 정립의 이점: 인시던트 사후 분석을 수행하고 결과를 공유하면 다른 워크로드에서 동일한 발생 요인이 나타날 경우 위험을 완화할 수 있으며 인시던트가 발생하기 전에 해결하거나 자동 복구를 구현할 수 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 인시던트 사후 분석을 위한 표준을 수립합니다. 우수한 인시던트 사후 분석은 시스템의 다른 위치에서 사용되는 아키텍처 패턴이 있는 문제에 대해 공통 솔루션을 제안할 수 있는 기회를 제공합니다.
  - 발생 요인을 솔직하게 밝히면서도 책임을 따지지 않도록 합니다.
  - 문제를 문서화하지 않으면 문제를 시정할 수 없습니다.
    - 인시던트 사후 분석에서 책임을 따지지 않으면서 제안된 시정 조치를 공정하게 평가하도록 하고 애플리케이션 팀의 솔직한 자체 평가와 협업을 유도합니다.
- 발생 요인을 확인하는 프로세스를 사용합니다. 재발을 제한하거나 방지하기 위한 완화책을 개발하고 빠르고 효과적인 대응을 위한 절차를 개발할 수 있도록 이벤트의 발생 요인을 식별하고 문서화하는 프로세스를 마련합니다. 목표 대상에 맞게 적절히 발생 요인을 알립니다.
  - 로그 분석이란 무엇일까요?

리소스

관련 문서:

- 로그 분석이란 무엇일까요?
- 오류 수정(COE)을 개발해야 하는 이유

REL12-BP03 기능 요구 사항 테스트

단위 테스트와 필수 기능을 검증하는 통합 테스트 등의 기법을 사용합니다.

이러한 테스트는 구축 및 배포 작업의 일부로 자동으로 실행될 때 최상의 결과를 제공합니다. 예를 들어 개발자가 AWS CodePipeline을 사용하여 소스 리포지토리에 변경 사항을 커밋하면 CodePipeline이 변경 사항을 자동으로 감지합니다. 이러한 변경 사항이 구축되고 테스트가 실행됩니다. 테스트가 완료되면 테스트를 위해 구축된 코드가 스테이징 서버에 배포됩니다. CodePipeline은 스테이징 서버에서 통합 또는 로드 테스트와 같은 더 많은 테스트를 실행합니다. 이러한 테스트가 성공적으로 완료되면 CodePipeline은 테스트되고 승인된 코드를 프로덕션 인스턴스에 배포합니다.

또한 가장 중요한 테스트 중 하나는 고객 행동을 실행하고 시뮬레이션할 수 있는 가상 트랜잭션 테스트(Canary 테스트라고도 하지만카나리 배포와는 다름)입니다. 다양한 원격 위치에서 워크로드 엔드포인트에 대해 이러한 테스트를 지속적으로 실행합니다. Amazon CloudWatch Synthetics를 사용하면 엔드포인트 및 API 모니터링을 위한 Canary를 생성 할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 기능 요구 사항을 테스트합니다. 여기에는 단위 테스트와 필수 기능을 검증하는 통합 테스트가 포함됩니다.
  - CodePipeline를 AWS CodeBuild와 함께 사용하여 코드 테스트 및 빌드 실행
  - AWS CodePipeline Adds Support for Unit and Custom Integration Testing with AWS CodeBuild(AWS CodePipeline, AWS CodeBuild를 사용한 단위 및 맞춤형 통합 테스트 관련 지원 추가)
  - Continuous Delivery and Continuous Integration(지속적 전달 및 지속적 통합)
  - Canary 사용(Amazon CloudWatch Synthetics)
  - 소프트웨어 테스트 자동화

리소스

관련 문서:

- APN 파트너: 지속적 통합 파이프라인 구현을 지원할 수 있는 파트너
- AWS CodePipeline Adds Support for Unit and Custom Integration Testing with AWS CodeBuild(AWS CodePipeline, AWS CodeBuild를 사용한 단위 및 맞춤형 통합 테스트 관련 지원 추가)
- AWS Marketplace: 지속적인 통합에 사용할 수 있는 제품
- Continuous Delivery and Continuous Integration(지속적 전달 및 지속적 통합)
- 소프트웨어 테스트 자동화

- CodePipeline를 AWS CodeBuild와 함께 사용하여 코드 테스트 및 빌드 실행
- Canary 사용(Amazon CloudWatch Synthetics)

REL12-BP04 확장 및 성능 요구 사항 테스트

워크로드가 조정 및 성능 요구 사항을 충족하는지 확인하기 위한 로드 테스트 등의 기법을 사용합니다.

클라우드에서는 워크로드에 대한 프로덕션 규모의 테스트 환경을 필요할 때 생성할 수 있습니다. 축소된 인프라에서 이러한 테스트를 실행하는 경우 관찰된 결과를 프로덕션 환경에서 발생할 것으로 생각되는 범위까지 확장해야 합니다. 실제 사용자에게 영향을 주지 않도록 주의하고 실제 사용자 데이터 및 손상된 사용 통계 또는 프로덕션 보고서와 충돌하지 않도록 테스트 데이터에 태그를 지정하면 프로덕션에서도 로드 및 성능 테스트를 수행할 수 있습니다.

테스트를 통해 기본 리소스, 조정 설정, 서비스 할당량 및 복원력 설계가 로드 조건에서 예상대로 작동하는지 확인합니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

- 크기 조정 및 성능 요구 사항을 테스트합니다. 로드 테스트를 수행하여 워크로드가 크기 조정 및 성능 요구 사항을 충족하는지 확인합니다.
  - AWS에서의 분산 로드 테스트: 수천 명의 연결된 사용자 시뮬레이션
  - Apache JMeter
    - 프로덕션 환경과 동일한 환경에 애플리케이션을 배포하고 로드 테스트를 수행합니다.
      - 코드형 인프라를 사용하여 프로덕션 환경과 최대한 유사한 환경을 구축합니다.

리소스

관련 문서:

- AWS에서의 분산 로드 테스트: 수천 명의 연결된 사용자 시뮬레이션
- Apache JMeter

REL12-BP05 카오스 엔지니어링을 이용한 복원력 테스트

시스템이 불리한 조건에서 어떻게 반응하는지 파악하려면 프로덕션 내 환경 또는 프로덕션과 최대한 가까운 환경에서 카오스 실험을 정기적으로 실행합니다.

원하는 결과:

워크로드의 복원력은 이벤트 중 워크로드의 알려진 예상 동작을 확인하는 복원력 테스트 이외에 장애 주입 실험 또는 예기치 않은 부하 발생의 형태로 카오스 엔지니어링을 적용하여 정기적으로 확인합니다. 카오스 엔지니어링과 복원력 테스트를 결합하여 구성 요소 장애 시 워크로드가 중단되지 않고, 예기치 않은 중단이 발생한 경우에도 영향을 최소화하거나 아무런 영향 없이 복구할 수 있다는 확신을 얻을 수 있습니다.

일반적인 안티 패턴:

- 복원력을 뛰어나게 설계했으나 장애 발생 시 워크로드가 전체적으로 작동하는 방식을 확인하지 않습니다.
- 실제 조건 및 예상되는 부하에서 절대 실험하지 않습니다.
- 실험을 코드로 처리하지 않고 개발 주기 전체에서 유지 관리하지 않습니다.
- 카오스 실험을 CI/CD 파이프라인의 일부로 또한 배포 범위 외부에서도 실행하지 않습니다.
- 어떤 결함을 실험할지 결정할 때 과거의 인시던트 후 분석 사용에 소홀합니다.

이 모범 사례 확립의 이점: 워크로드의 복원력을 확인하기 위해 장애를 도입하면 탄력적인 설계의 복구 절차가 실제 장애 발생 시에도 잘 작동할 것으로 확신할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

카오스 엔지니어링은 서비스 공급자, 인프라, 워크로드 및 구성 요소 수준에서 고객에게 미치는 영향을 최소화하거나 아무런 영향을 미치지 않으면서 제어되는 방식으로 실제 중단(시뮬레이션)을 지속적으로 도입할 수 있는 역량을 팀에 제공합니다. 이렇게 하면 팀이 장애로부터 배우고 워크로드의 복원력을 관찰, 측정 및 개선하고 이벤트 발생 시 알림이 전달되고 팀이 알림을 받는지 확인할 수 있습니다.

지속적으로 수행하면 카오스 엔지니어링은 해결하지 않고 두면 가용성과 운영에 부정적인 영향을 미칠 수 있는 결함을 강조해서 보여줄 수 있습니다.

> **Note**
>
> 카오스 엔지니어링은 프로덕션 환경의 급변하는 조건을 견딜 수 있는 시스템의 역량을 확신하기 위해 시스템에 대해 수행하는 실험 분야입니다. Principles of Chaos Engineering(카오스 엔지니어링의 원칙)

시스템이 중단을 견딜 수 있는 경우 카오스 엔지니어링은 자동화되어 회귀 테스트로 유지 관리해야 합니다. 이러한 방식으로 카오스 실험은 시스템 개발 수명 주기(SDLC) 및 CI/CD 파이프라인의 일부로 수행해야 합니다.

구성 요소 장애 발생 시 워크로드가 중단되지 않는지 확인하기 위해 실험의 일부로 실제 이벤트를 도입합니다. 예를 들어, Amazon EC2 인스턴스 손실 또는 기본 Amazon RDS 데이터베이스 인스턴스 장애 조치로 실험하고 워크로드가 영향을 받지 않는지(또는 최소한의 영향만 받는지) 확인합니다. 구성 요소 장애를 결합하여 가용 영역에서 중단으로 인해 발생할 수 있는 이벤트를 시뮬레이션합니다.

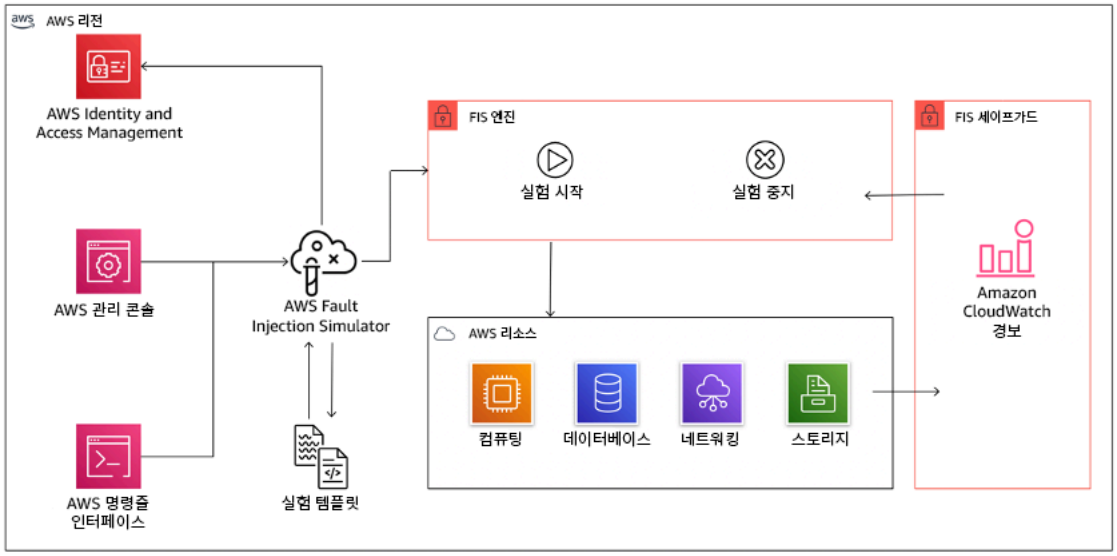
애플리케이션 수준 장애(예: 충돌)의 경우 메모리 및 CPU 소진 등과 같은 원인으로 시작할 수 있습니다.

간헐적 네트워크 중단으로 인한 외부 종속에 대한 폴백 또는 장애 조치 메커니즘 을 확인하려면 구성 요소가 몇 초에서 몇 시간까지 지속될 수 있는 지정된 기간에 서드파티 제공업체에 대한 액세스를 차단하여 해당 이벤트를 시뮬레이션해야 합니다.

그 외의 모드에서 성능이 저하되면 기능이 저하되고 응답 속도가 느려져서 서비스가 일시적으로 중단되는 경우가 많습니다. 이러한 성능 저하는 대개 중요 서비스의 지연 시간 증가 및 불안정한 네트워크 연결(패킷이 삭제됨)로 인해 발생합니다. 지연 시간, 삭제된 메시지 및 DNS 장애 등과 같은 네트워킹 효과를 비롯한 장애를 사용한 실험에는 이름 확인 불가, DNS 서비스에 연결 불가 또는 종속적 서비스에 연결 설정 불가가 포함될 수 있습니다.

카오스 엔지니어링 도구:

AWS Fault Injection Service(AWS FIS)은(는) CD 파이프라인의 일부로 또는 파이프라인 외부에서 사용할 수 있는 장애 주입 실험을 실행하는 데 사용되는 완전관리형 서비스입니다. AWS FIS은(는) 카오스 엔지니어링 게임 데이 중 사용하기 좋습니다. Amazon EC2, Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service(Amazon EKS) 및 Amazon RDS을(를) 비롯한 여러 유형의 리소스 간에 동시 장애 발생을 지원합니다. 이러한 장애에는 리소스 종료, 강제 장애 조치, CPU 또는 메모리에 스트레스 유발, 제한, 지연 시간 및 패킷 손실이 포함됩니다. Amazon CloudWatch 경보와 통합되므로 테스트가 예상치 못한 영향을 미치는 경우 실험을 롤백하기 위해 중지 조건을 가드레일로 설정할 수 있습니다.



AWS Fault Injection Service은(는) 워크로드에 장애 주입 실험을 실행할 수 있도록 AWS 리소스와 통합됩니다.

장애 주입 실험을 위한 서드파티 옵션도 몇 가지 있습니다. 여기에는 오픈 소스 도구(예: Chaos Toolkit, Chaos Mesh 및 Litmus Chaos)와 상용 옵션(예: Gremlin)이 포함됩니다. AWS에 삽입할 수 있는 장애 범위를 확장하기 위해 AWS FIS이(가) Chaos Mesh 및 Litmus Chaos와 통합되어 여러 도구 간에 장애 주입 워크플로를 조정할 수 있습니다. 예를 들어, 임의로 선택된 비율의 클러스터 노드를 AWS FIS 장애 작업을 사용하여 종료하는 동안 Chaos Mesh 또는 Litmus 결함을 사용하여 포드의 CPU에 대한 스트레스 테스트를 실행할 수 있습니다.

구현 단계

- 실험에 사용할 장애를 결정합니다.

  워크로드 설계의 복원력을 평가합니다. 해당 설계(Well-Architected Framework의 모범 사례를 사용하여 생성됨)는 중요한 종속성, 과거 이벤트, 알려진 문제 및 규정 준수 요구 사항을 기반으로 위험을 설명합니다. 복원력을 유지하기 위한 설계 요소와 완화하도록 설계된 장애를 나열합니다. 이러한 목록 작성에 대한 자세한 내용은 이전 인시던트 재발을 방지하기 위한 프로세스 생성 방법을 안내하는 운영 준비 검토 백서 를 참조하세요. 장애 모드 및 영향 분석(FMEA) 프로세스는 장애의 구성 요소 수준 및 장애가 워크로드에 미치는 영향을 분석하기 위한 프레임워크를 제공합니다. FMEA는 Adrian Cockcroft가 장애 모드 및 지속적인 복원력에서 자세히 설명합니다.
- 각 장애에 우선순위를 할당합니다.

  처음에는 높음, 보통 또는 낮음 등과 같이 대략적인 분류로 시작합니다. 우선순위를 평가하려면 장애 빈도와 장애가 전체 워크로드에 미치는 영향을 고려해야 합니다.

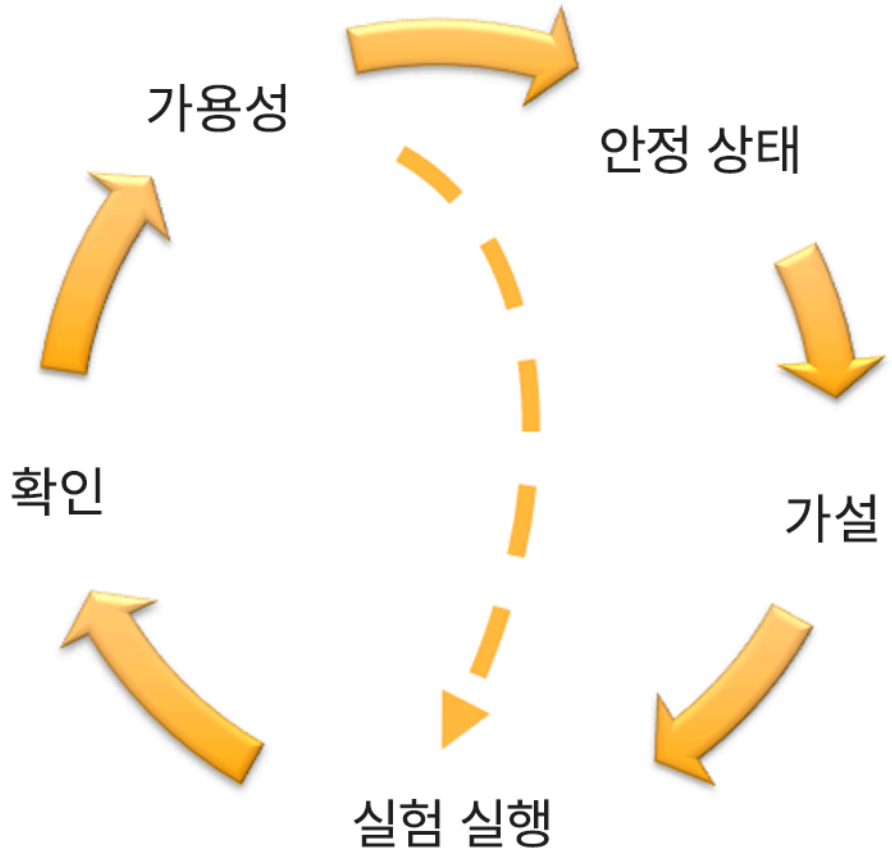
주어진 장애의 빈도를 고려할 때 확인 가능한 경우 이 워크로드에 대한 이전 데이터를 분석합니다. 확인할 수 없는 경우에는 유사한 환경에서 실행되는 다른 워크로드의 데이터를 사용합니다.

주어진 장애의 영향을 고려할 때 장애의 범위가 클수록 일반적으로 영향도 커집니다. 또한 워크로드 설계 및 용도도 고려합니다. 예를 들어, 소스 데이터 스토어에 액세스하는 기능은 데이터 변환 및 분석을 수행하는 워크로드에 중요합니다. 이러한 경우 액세스 장애와 제한된 액세스 및 지연 시간 삽입에 대한 실험의 우선순위를 지정할 수 있습니다.

인시던트 후 분석은 장애 모드의 빈도 및 영향을 파악하기 위한 좋은 데이터 출처입니다.

할당된 우선순위를 사용하여 어떤 장애를 먼저 실험할지, 새로운 장애 주입 실험을 개발할 순서를 결정합니다.

- 수행하는 각 실험에 대해 카오스 애플리케이션 및 지속적인 복원력 플라이휠을 따릅니다.



Adrian Hornsby의 과학적 방법을 사용하는 카오스 엔지니어링 및 연속 복원력 플라이휠

- 안정 상태를 정상 동작을 나타내는 워크로드의 측정 가능한 출력으로 정의합니다.

예상한 대로 안정적으로 작동하면 워크로드가 안정 상태를 보이는 것입니다. 따라서 안정 상태를 정의하기 전에 워크로드가 정상인지 확인합니다. 특정 비율의 장애는 허용 가능한 한도 내에서 발생할 수 있기 때문에 안정 상태라고 해서 장애 발생 시 워크로드에 미치는 영향이 반드시 없는 것은 아닙니다. 안정 상태는 실험 중 관찰하는 기준선으로, 다음 단계에서 정의하는 가설이 예상대로 나타나지 않는 경우 이상이 있음을 강조합니다.

예를 들어, 결제 시스템의 안정 상태를 성공률 99%, 왕복 시간 500ms의 300 TPS 처리로 정의할 수 있습니다.

- 워크로드가 장애에 반응할 방법에 대한 가설을 작성합니다.

  올바른 가설은 안정 상태 유지를 위해 장애 완화에 기대되는 작동 방식을 기반으로 합니다. 가설은 워크로드가 특정 완화를 수행하도록 설계되었기 때문에 특정 유형의 장애 발생 시 시스템 또는 워크로드가 계속해서 안정 상태를 유지할 것이라고 가정합니다. 특정 유형의 장애 및 완화가 가설에 명시되어 있어야 합니다.

  가설에는 다음 템플릿을 사용할 수 있습니다(하지만 다르게 표현할 수도 있음).

  > **Note**
  >
  > 만약 ## ## 가 발생하면 #### ## 워크로드가 #### ## ## ### ## 을 유지하기 위한 ## ## ### #####.

  예를 들면 다음과 같습니다.

  - Amazon EKS 노드 그룹 중 20%의 노드가 종료되면 트랜잭션 생성 API가 100ms에서 요청의 99%를 계속해서 제공합니다(안정 상태). Amazon EKS 노드는 5분 이내에 복구되고 포드는 실험 시작 후 8분 이내에 트래픽을 예약하고 처리합니다. 3분 이내에 알림이 전송됩니다.
  - 단일 Amazon EC2 인스턴스 장애가 발생하면 주문 시스템의Elastic Load Balancing 상태 확인 기능이 Elastic Load Balancing에 나머지 정상 인스턴스에만 요청을 보내도록 하고 동시에 Amazon EC2 Auto Scaling에서는 장애가 발생한 인스턴스를 교체하여 서버 측(5xx) 오류의 증가율을 0.01% 미만으로 유지합니다(안정 상태).
  - 1차 Amazon RDS 데이터베이스 인스턴스에서 장애가 발생하면 공급망 데이터 컬렉션 워크로드는 장애 조치를 취하고 대기Amazon RDS 데이터베이스 인스턴스에 연결하여 데이터베이스 읽기 또는 쓰기 오류를 1분 미만으로 유지합니다(안정 상태).
- 장애를 도입하여 실험을 실행합니다.

실험은 기본적으로 장애 발생 시 안전해야 하며 워크로드에서 허용해야 합니다. 워크로드에서 장애가 발생할 것임을 아는 경우 실험을 실행하지 마세요. known-unknowns 또는 unknown-unknowns를 찾으려면 카오스 엔지니어링을 사용해야 합니다. Known-unknowns 는 알고 있지만 완전히 파악하지 못한 항목이며, unknown-unknowns 는 알지 못할 뿐만 아니라 완전히 파악하지 못한 항목입니다. 실패할 것을 알고 있는 워크로드에 대한 실험에서는 새로운 인사이트를 얻을 수 없습니다. 실험은 주의해서 계획해야 하고 영향 범위가 명확해야 하며 예기치 못한 변동 발생 시 적용할 수 있는 롤백 메커니즘을 제공해야 합니다. 실사에서 워크로드가 실험을 통과해야 한다고 나타나면 실험을 계속 진행합니다. 장애 주입을 위한 옵션은 여러 가지가 있습니다. AWS에서의 워크로드의 경우 AWS FIS 에서는 작업이라 부르는 미리 정의된 다양한 장애 시뮬레이션을 해당 알림에 대한 작업이. 또한 다음 문서를 사용하여 AWS FIS에서 실행하는 사용자 지정 작업을 정의할 수도 있습니다. AWS Systems Manager 문서.

스크립트에 워크로드의 현재 상태를 파악하는 기능이 없고, 스크립트가 로그를 내보낼 수 없고, 가능한 경우 롤백 메커니즘 및 중지 조건을 제공할 수 없는 경우에는 카오스 실험에 사용자 지정 스크립트를 사용하지 않는 것이 좋습니다.

카오스 엔지니어링을 지원하는 효율적인 프레임워크 또는 도구 세트는 실험의 현재 상태를 추적하고, 로그를 내보내고, 실험의 제어되는 실행을 지원하기 위한 롤백 메커니즘을 제공해야 합니다. 실험 시 예기치 못한 변동이 발생하는 경우 실험을 롤백하는 안전 메커니즘과 분명하게 정의된 범위가 있는 실험을 수행하도록 허용하는 AWS FIS 등과 같은 확립된 서비스로 시작합니다. AWS FIS을(를) 사용한 다양한 실험에 대해 알아보려면 Resilient and Well-Architected Apps with Chaos Engineering lab(카오스 엔지니어링을 사용해 잘 설계된 복원력이 뛰어난 앱 실습)을 참조하세요. 또한 AWS Resilience Hub 은(는) 워크로드를 분석하여 AWS FIS에서 구현 및 실행하도록 선택할 수 있는 실험을 생성합니다.

> **Note**
>
> 모든 실험에 대해 범위와 그 영향을 명확하게 이해해야 합니다. 프로덕션 환경에서 실행하기 전에 비프로덕션 환경에서 먼저 장애를 시뮬레이션하는 것이 좋습니다.

실험은 가능한 경우 제어 및 실험적 시스템 배포를 둘 다 실행하는 카나리 배포 를 사용하여 실제 로드를 받는 프로덕션 환경에서 실행해야 합니다. 프로덕션 환경에서 처음으로 실험하는 경우 잠재적인 영향을 완화하기 위해 사용량이 적은 시간에 실험을 실행하는 것이 좋습니다. 또한 실제 고객 트래픽 사용의 위험이 너무 큰 경우에는 프로덕션 인프라에서 제어 및 실험적 배포에 대해 가상 트래픽을 사용하여 실험을 실행할 수 있습니다. 프로덕션 환경을 사용할 수 없는 경우 가급적 프로덕션 환경과 유사한 프로덕션 전 환경에서 실험을 실행합니다.

실험이 허용 가능한 제한을 벗어나 프로덕션 트래픽 또는 다른 시스템에 영향을 미치지 않도록 가드레일을 설정하고 모니터링해야 합니다. 가드레일 지표에 대해 정의한 임곗값에 도달한 경우 실험을 중지하기 위한 중지 조건을 설정합니다. 중지 조건에는 워크로드의 안정 상태에 대한 지표와 장애를 도입하는 구성 요소에 대한 지표를 포함해야 합니다. 또한 종합 모니터링 (사용자 canary라고도 함)은 일반적으로 사용자 프록시로 포함해야 하는 한 가지 지표입니다. AWS FIS에 대한 중지 조건 은 실험 템플릿의 일부로 지원되며 템플릿당 최대 5개의 중지 조건을 사용할 수 있습니다.

카오스 원칙 중 한 가지는 실험 범위 및 그 영향을 최소화하는 것입니다.

단기적인 부정적 영향 중 일부를 허용해야 하지만 실험의 여파를 최소화하고 제한하는 것은 카오스 엔지니어의 책임이자 의무입니다.

실험 범위 및 잠재적인 영향을 확인하기 위한 방법은 먼저 비프로덕션 환경에서 실험을 수행하여 실험 중 중지 조건의 임곗값이 예상대로 활성화되고 프로덕션 환경에서 직접 실험하지 않고 관찰을 통해 예외를 포착할 수 있는지 확인하는 것입니다.

장애 주입 실험을 실행할 때 책임 당사자 모두가 알림을 잘 받았는지 확인합니다. 운영 팀, 서비스 신뢰성 팀, 고객 지원 팀 등과 같은 적절한 팀과 소통하여 실험 실행 시점과 기대하는 결과에 대해 알립니다. 이러한 팀에 통신 도구를 제공하여 부정적인 영향을 확인한 경우 실험을 실행하는 팀에 알릴 수 있도록 합니다.

워크로드와 기본 시스템을 원래 정상 상태로 복원해야 합니다. 보통, 워크로드의 탄력적인 설계는 스스로 복구됩니다. 하지만 일부 장애 설계 또는 장애가 발생한 실험에서는 워크로드를 예기치 않은 장애 상태로 둘 수 있습니다. 따라서 실험을 마치면 이 점을 인식하고 워크로드 및 시스템을 복원해야 합니다. AWS FIS를 사용하면 작업 파라미터 내에서 롤백 구성을 설정할 수 있습니다(후속 작업이라고도 함). 후속 작업은 대상을 작업 실행 전의 상태로 되돌립니다. 자동 작업(예: AWS FIS 사용)이든 수동 작업이든 상관 없이 후속 작업은 장애 감지 및 처리 방법을 설명하는 플레이북의 일부여야 합니다.

- 가설을 확인합니다.

  Principles of Chaos Engineering(카오스 엔지니어링의 원칙) 은 워크로드의 안정 상태를 확인하는 방법에 대한 다음 지침을 제공합니다.

  시스템의 내부 특성보다는 시스템의 측정 가능한 결과에 중점을 둡니다. 단기간의 결과에 대한 측정값은 시스템의 안정 상태에 대한 프록시를 구성합니다. 전체 시스템의 처리량, 오류 발생률 및 지연 시간 백분위수는 모두 안정 상태 동작을 나타내는 관심 지표일 수 있습니다. 실험 중 체계적인 동작 패턴에 집중하여 카오스 엔지니어링은 시스템 작동 방식 확인이 아니라 시스템이 잘 작동하는지를 확인합니다.

이전 두 가지 예에서는 서버측(5xx) 오류 증가를 0.01% 미만으로, 데이터베이스 읽기 및 쓰기 오류를 1분 미만으로 지정한 안정 상태 지표를 포함합니다.

5xx 오류는 워크로드의 클라이언트가 직접 경험하는 장애 모드의 결과이기 때문에 좋은 지표입니다. 데이터베이스 오류 측정은 장애의 직접적인 결과이기 때문에 적절하긴 하지만 실패한 고객 요청 수 또는 고객에게 표시된 오류 수 등과 같은 클라이언트 영향 측정으로 보충해야 합니다. 또한 워크로드의 클라이언트가 직접 액세스할 수 있는 API 또는 URI에 종합 모니터(사용자 canary라고도 함)를 포함합니다.

- 복원력을 위해 워크로드 설계를 개선합니다.

  안정 상태가 유지되지 않는 경우 장애를 완화하기 위해 워크로드 설계를 어떻게 개선할 수 있는지 조사하여 AWS Well-Architected 신뢰성 원칙에 대한 모범 사례를 적용합니다. 추가 지침 및 리소스는 AWS 빌더 라이브러리에서 찾을 수 있습니다. 이 라이브러리는 특히, 상태 확인 개선 방법 또는 애플리케이션 코드에서 백오프를 사용하여 재시도 채택 방법에 대한 문서를 호스트합니다.

  이러한 변경 사항을 구현한 후에는 실험을 다시 실행하여(카오스 엔지니어링 프라이휠에서 점선으로 표시됨) 변경 사항의 영향을 확인합니다. 확인 단계에서 가설이 참으로 입증되면 워크로드가 안정 상태이므로 주기가 계속됩니다.

- 실험을 정기적으로 실행합니다.

  카오스 실험은 주기이고 카오스 엔지니어링의 일부로 주기적으로 실행해야 합니다. 워크로드가 실험의 가설을 충족하면 CI/CD 파이프라인의 회귀 부분으로 계속해서 실행되도록 실험을 자동화해야 합니다. 이렇게 하는 방법을 알아보려면 AWS CodePipeline을(를) 사용하여 AWS FIS 실험을 실행하는 방법에 관한 블로그를 참조하세요. 이 실습은 CI/CD 파이프라인에서 반복 AWS FIS 실험 에 관한 내용으로, 직접 적용해 볼 수 있습니다.

  장애 주입 실험은 게임 데이의 일부이기도 합니다(REL12-BP06 정기적으로 게임 데이 진행참조). 게임 데이에서는 장애나 이벤트를 시뮬레이션하여 시스템, 프로세스 및 팀 대응을 확인합니다. 게임 데이의 목적은 이례적인 이벤트 발생 시 팀이 수행해야 할 작업을 실제로 수행해보는 것입니다.

- 실험 결과를 캡처하고 저장합니다.

  장애 주입 실험의 결과는 캡처한 다음 지속해야 합니다. 나중에 실험 결과 및 추세를 분석할 수 있도록 필요한 데이터(예: 시간, 워크로드 및 조건)를 모두 포함합니다. 결과의 예에는 대시보드 스냅샷, 지표 데이터베이스의 CSV 덤프 또는 이벤트에 대해 손으로 작성한 기록, 실험에서 관찰한 내용 등이 있을 수 있습니다. AWS FIS(으)로 기록한 실험은 이러한 데이터 캡처의 일부일 수 있습니다.

리소스

관련 모범 사례:

- REL08-BP03 배포의 일부로 복원력 테스트 통합
- REL13-BP03 재해 복구 구현을 테스트하여 구현 확인

관련 문서:

- AWS Fault Injection Service란 무엇인가요?
- AWS Resilience Hub란 무엇인가요?
- Principles of Chaos Engineering(카오스 엔지니어링의 원칙)
- Chaos Engineering: Planning your first experiment(카오스 엔지니어링: 첫 번째 실험 계획)
- Resilience Engineering: Learning to Embrace Failure(복원력 엔지니어링: 실패를 수용하는 방법 학습)
- 카오스 엔지니어링 스토리
- 분산 시스템의 폴백 방지
- 카오스 실험을 위한 카나리 배포

관련 동영상:

- AWS re:Invent 2020: Testing resiliency using chaos engineering(카오스 엔지니어링을 사용하여 복원력 테스트)(ARC316)
- AWS re:Invent 2019: Improving resiliency with chaos engineering(카오스 엔지니어링을 사용하여 복원력 개선)(DOP309-R1)
- AWS re:Invent 2019: Performing chaos engineering in a serverless world(서버리스 환경에서 카오스 엔지니어링 수행)(CMY301)

관련 예시:

- Well-Architected lab: Level 300: Testing for Resiliency of Amazon EC2, Amazon RDS, and Amazon S3(Amazon EC2, Amazon RDS 및 Amazon S3의 복원력 테스트)
- Chaos Engineering on AWS lab(AWS에서 카오스 엔지니어링 실습)
- Resilient and Well-Architected Apps with Chaos Engineering lab(카오스 엔지니어링을 사용해 잘 설계된 복원력이 뛰어난 앱 실습)을 참조하세요

- Serverless Chaos lab(서버리스 카오스 실습)
- Measure and Improve Your Application Resilience with AWS Resilience Hub lab(AWS Resilience Hub를 사용하여 애플리케이션 복원력 측정 및 개선 실습)

관련 도구:

- AWS Fault Injection Service
- AWS Marketplace: Gremlin Chaos Engineering Platform
- Chaos Toolkit
- Chaos Mesh
- Litmus

REL12-BP06 정기적으로 게임 데이 진행

실제 장애 시나리오에 영향을 받을 직원들과 함께 게임 데이를 정기적으로 수행하여 프로덕션에 최대한 근접한 장애 및 이벤트 대응 절차(프로덕션 환경의 장애 절차 포함)를 연습합니다. 게임 데이에서는 프로덕션 이벤트가 사용자에게 영향을 미치지 않도록 하는 조치가 시행됩니다.

게임 데이에서는 장애나 이벤트를 시뮬레이션하여 시스템, 프로세스 및 팀 대응을 테스트합니다. 게임 데이의 목적은 이례적인 이벤트 발생 시 팀이 수행해야 할 작업을 실제로 수행해보는 것입니다. 그러면 어느 분야에서 개선이 필요한지 파악하고, 조직이 이벤트에 대처하는 경험을 쌓도록 도울 수 있습니다. 팀이 대응 방법을 체득할 수 있도록 게임 데이를 정기적으로 진행해야 합니다.

복원력을 위한 설계가 마련되고 비 프로덕션 환경에서 이 설계를 테스트한 후에는 실전 연습을 통해 모든 구성 요소가 프로덕션에서 예상대로 작동하는지 확인합니다. 실전 연습, 특히 첫 번째 실전 연습에서는 엔지니어와 운영 팀이 모두 모여 실전 연습의 일정과 내용에 대한 정보를 숙지합니다. 런북이 준비되어 있습니다. 프로덕션 시스템에서 미리 정해진 방식으로 발생할 수 있는 장애 이벤트를 비롯한 시뮬레이션 이벤트가 실행되고 영향이 평가됩니다. 모든 시스템이 설계대로 작동할 경우 감지 및 자가 복구가 수행됩니다. 영향은 거의 없습니다. 그러나 부정적인 영향이 관찰되면 테스트가 롤백되고 워크로드 문제가 해결됩니다. 필요한 경우 런북을 사용하여 수동으로 문제를 해결합니다. 게임 데이는 프로덕션에서 수행되는 경우가 많으므로 고객의 가용성에 영향을 미치는 일이 없도록 모든 예방 조치를 취해야 합니다.

일반적인 안티 패턴:

- 절차를 문서화하지만 결코 연습하지 않음
- 테스트 연습에 비즈니스 의사 결정권자가 참여하지 않음

이 모범 사례 수립의 이점: 게임 데이를 정기적으로 실시하면 실제 인시던트가 발생할 때 모든 직원이 정책과 절차를 따르도록 하고 이러한 정책과 절차가 적절한지 검증할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 런북 및 플레이북을 정기적으로 실행하도록 게임 데이 수행 게임 데이에는 비즈니스 소유자, 개발 직원, 운영 직원 및 인시던트 대응 팀 등 프로덕션 이벤트에 관련된 모든 사람이 참여해야 합니다.
  - 로드 또는 성능 테스트를 실행한 다음 장애 주입을 실행합니다.
  - 런북에서 이상이 있는지 찾고 플레이북을 연습할 기회가 있는지 살펴봅니다.
    - 런북에서 벗어난 경우 해당 런북을 구체화하거나 동작을 수정합니다. 플레이북을 실행하는 경우, 사용되었어야 하는 런북을 식별하거나 새로운 런북을 만듭니다.

리소스

관련 문서:

- AWS 게임 데이란 무엇입니까?

관련 동영상:

- AWS re:Invent 2019: Improving resiliency with chaos engineering(카오스 엔지니어링을 사용하여 복원력 개선)(DOP309-R1)

  관련 예시:

- AWS Well-Architected 실습 - 복원력 테스트

REL 13 DR(재해 복구)를 어떻게 계획합니까?

DR 전략의 시작은 백업 및 이중화 워크로드 구성 요소를 갖추는 것입니다. RTO 및 RPO는 워크로드 복원을 위한 목표입니다. 비즈니스 요구 사항에 따라 이러한 목표를 설정합니다. 워크로드 리소스 및 데이터의 위치와 기능을 고려하여 이러한 목표를 충족하는 전략을 구현합니다. 중단 가능성과 복구 비용도 워크로드에 대한 재해 복구 옵션을 갖추는 것이 지니는 비즈니스 가치를 파악하는 데 도움이 되는 주요 요소입니다.

모범 사례

REL13-BP01 가동 중단 시간 및 데이터 손실 시의 복구 목표 정의

워크로드에는 RTO(복구 시간 목표) 및 RPO(복구 시점 목표)가 있습니다.

RTO(복구 시간 목표) 는 서비스 중단 시점과 서비스 복원 시점 간에 허용되는 최대 지연 시간으로, 서비스를 사용할 수 없는 상태로 허용되는 기간을 결정합니다.

RPO(복구 시점 목표) 는 마지막 데이터 복구 시점 이후 허용되는 최대 시간으로, 마지막 복구 시점과 서비스 중단 시점 사이에 허용되는 데이터 손실량을 결정합니다.

워크로드에 적절한 재해 복구(DR) 전략을 선택할 때 RTO 및 RPO 값을 고려하는 것이 중요합니다. 이 두 가지 목표는 비즈니스 차원에서 결정하고 기술 팀에서 DR 전략을 선택하여 구현할 때 사용합니다.

원하는 결과:

워크로드마다 비즈니스 영향에 따라 정의된 RTO 및 RPO가 할당됩니다. 워크로드는 관련 RTO와 RPO로 서비스 가용성과 용인 가능한 데이터 손실을 정의하는 사전에 정의된 티어에 할당됩니다. 그러한 티어 할당이 불가능하면 나중에 티어를 생성할 의도로 워크로드마다 맞춤형으로 할당될 수 있습니다. RTO 및 RPO는 워크로드의 재해 복구 전략 구현을 선택하기 위한 기본적인 고려 사항 중 하나로 사용됩니다. DR 전략을 선택할 때 추가로 고려해야 할 사항은 비용 제약, 워크로드 종속성, 운영 요구 사항입니다.

RTO의 경우 중단의 기간에 따른 영향을 파악합니다. 영향이 선형적인지, 비선형적인 영향이 있는지 (예: 4시간 후에 다음 교대가 시작될 때까지 제조 라인을 중단시킴) 파악해야 합니다.

다음과 같은 재해 복구 매트릭스는 워크로드 중요도가 복구 목표와 어떤 연관이 있는지 파악하는 데 도움이 됩니다. (참고로 X 축과 Y 축의 실제 값은 조직의 요구 사항에 따라 맞춤화해야 합니다.)

재해 복구 매트릭스						
		복구 시점 목표				
		< 1분	< 1시간	< 6시간	< 1일	1일 이상
복구 시간 목표	< 10분	심각	심각	높음	보통	보통
	< 2시간	심각	높음	보통	보통	낮음
	< 8시간	높음	보통	보통	낮음	낮음
	< 24시간	보통	보통	낮음	낮음	낮음
	24시간 이상	보통	낮음	낮음	낮음	낮음

그림 16: 재해 복구 매트릭스

일반적인 안티 패턴:

- 복구 목표가 정의되지 않음
- 임의의 복구 목표 선택
- 너무 관대하고 비즈니스 목표를 충족하지 못하는 복구 목표 선택
- 가동 중단 시간 및 데이터 손실의 영향을 파악하지 않음
- 워크로드 구성에서 달성할 수 없는 즉각 복구 또는 데이터 무손실과 같이 비현실적인 복구 목표 선택
- 실제 비즈니스 목표보다 더 엄격한 복구 목표 선택. 이로 인해 워크로드에 필요한 수준 이상으로 DR 구현의 비용이 높아지고 DR 구현이 복잡해집니다.
- 종속 워크로드와 호환되지 않는 복구 목표 선택
- 규제 요구 사항을 고려하지 않은 복구 목표
- 워크로드에 대한 RTO 및 RPO를 정의했으나 테스트하지 않음

이 모범 사례 수립의 이점: 재해 복구를 구현하는 데 기준이 될 시간 및 데이터 손실에 대한 복구 목표가 필요합니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

주어진 워크로드에서 가동 중단 시간 및 데이터 손실이 비즈니스에 미치는 영향을 파악해야 합니다. 가동 중단 시간이나 데이터 손실이 커질수록 일반적으로 영향이 커지지만, 영향이 어떻게 커지는지는 워

크로드 유형에 따라 다를 수 있습니다. 예를 들어, 영향이 적을 때는 가동 중단 시간을 최대 1시간까지 용인할 수 있지만 그 후에는 영향이 빠르게 증가할 수 있습니다. 비즈니스에 미치는 영향은 비용 손실(수익 손실), 고객 신뢰(평판에 미치는 영향), 운영 문제(급여 누락 또는 생산성 저하), 규제 위험 등 다양한 형태로 나타날 수 있습니다. 다음 단계를 따라 이러한 영향을 이해하고 워크로드에 RTO 및 RPO를 설정하세요.

구현 단계

1. 이 워크로드의 비즈니스 이해 관계자를 파악하고 이해 관계자를 관여시켜 이 단계를 구현합니다. 워크로드의 복구 목표는 비즈니스 차원의 의사 결정 사항입니다. 그런 다음 기술 팀에서 비즈니스 이해 관계자와 협력하여 이 목표를 사용해 DR 전략을 선택합니다.

   > **Note**
   >
   > 2단계와 3단계에서는 다음을 사용합니다. the section called “구현 워크시트”.

2. 아래 질문에 답하여 의사 결정을 내리는 데 필요한 정보를 수집합니다.
3. 조직에 워크로드 영향에 대한 중요도 범주나 티어가 있습니까?
   a. 있다면 이 워크로드를 범주에 할당합니다.
   b. 없다면 범주를 만듭니다. 5개 이하의 범주를 생성하고 각각의 복구 시간 목표 범위를 구체화합니다. 범주 예시로는 ‘매우 중요’, ‘높음’, ‘보통’, ‘낮음’이 있습니다. 워크로드가 각각의 범주에 어떻게 해당하는지 파악하려면 워크로드가 미션에 필수적인지, 비즈니스 차원에서 중요한지 아니면 비즈니스 추진과 큰 관련이 없는지 고려합니다.
   c. 범주에 따라 워크로드 RTO와 RPO를 설정합니다. 언제나 이 단계에 들어설 때 계산한 원래의 값보다 더 엄격하게(더 낮은 RTO 및 RPO) 범주를 선택하세요. 이로 인해 값이 부적절하게 크게 변경된다면 새로운 범주를 만드는 것을 고려하세요.
4. 이 답변에 따라 RTO 및 RPO 값을 워크로드에 할당합니다. 직접 할당하거나 워크로드를 사전에 정의된 서비스 티어에 할당하면 됩니다.
5. 이 워크로드의 재해 복구 계획(DRP)을 문서화합니다. 이는 조직의 비즈니스 연속성 계획(BCP)에포함되며, 워크로드 팀 및 이해 관계자가 액세스할 수 있는 위치에 저장합니다.
   a. RTO 및 RPO와 이러한 값을 결정하는 데 사용된 정보를 기록합니다. 비즈니스에 미치는 워크로드의 영향을 평가하는 데 사용한 전략을 포함합니다.
   b. 재해 복구 목표에서 추적하고 있거나 추적하려는 RTO 및 RPO 외의 다른 지표를 기록합니다.
   c. 이 계획에 DR 전략 및 런북의 세부 정보를 추가합니다.
6. 그림 15와 같은 매트릭스에서 워크로드 중요도를 찾아보면 조직을 위해 사전 정의된 서비스 티어를 설정할 수 있습니다.

7. 에 따라 DR 전략(또는 DR 전략의 개념 증명)을 구현하고 나면the section called “REL13-BP02 복구 목표 달성을 위해 정의된 복구 전략 사용”이 전략을 테스트하여 워크로드의 실제 RTC(복구 시간 역량) 및 RPC(복구 시점 역량)를 파악합니다. 이것이 복구 목표에 부합하지 않으면 비즈니스 이해 관계자와 협력하여 목표를 조정하거나 목표에 부합하도록 DR 전략을 변경합니다.

기본 질문

1. 비즈니스에 심각한 영향이 미치기 전에 워크로드가 중단되어도 되는 최대 시간은 얼마입니까?
   a. 워크로드가 중단되었을 때 분당 비즈니스에 발생하는 금전적 비용(직접적인 금전적 영향)을 파악합니다.
   b. 이 영향은 항상 선형적이지 않다는 점을 고려합니다. 처음에는 영향이 제한적이지만 특정 시점을 지나면 영향이 빠르게 증가할 수 있습니다.
2. 비즈니스에 심각한 영향이 미치기 전에 손실되어도 되는 데이터의 최대 양은 얼마입니까?
   a. 가장 중요한 데이터 스토어에 대해 이 값을 고려합니다. 다른 데이터 스토어에 대해 각각의 중요도를 파악합니다.
   b. 워크로드 데이터가 손실되면 재생성할 수 있습니까? 백업 후 복원하는 것보다 재생성이 운영상 더 용이하면 워크로드 데이터를 재생성하는 데 사용되는 소스 데이터의 중요도에 따라 RPO를 선택합니다.
3. 이것이 의존하는 워크로드(다운스트림) 또는 이것에 의존하는 워크로드(업스트림)의 복구 목표 및 가용성 기대치는 얼마입니까?
   a. 이 워크로드가 업스트림 종속성의 요구 사항을 충족하도록 하는 복구 목표를 선택합니다.
   b. 다운스트림 종속성의 복구 기능에 따라 달성할 수 있는 복구 목표를 선택합니다. 중요하지 않은 다운스트림 종속성(다른 해결책이 있는 것)은 제외할 수 있습니다. 아니면 중요한 다운스트림 종속성으로 필요하다면 복구 기능을 개선합니다.

추가 질문

아래 질문을 고려하고 이 워크로드에 어떻게 적용되는지 생각하세요.

4. 중단의 유형(예: 리전, 가용 영역 등)에 따라 다른 RTO 및 RPO가 있습니까?
5. RTO/RPO가 변경될 수 있는 특정 시기(계절, 세일 이벤트, 제품 출시)가 있습니까? 그렇다면 다른 측정 방식과 시간 경계가 무엇입니까?
6. 워크로드가 중단되면 얼마나 많은 고객이 영향을 받습니까?
7. 워크로드가 중단될 경우 평판에 미치는 영향은 무엇입니까?

8. 워크로드가 중단될 경우 발생할 수 있는 운영상의 다른 영향에는 어떤 것이 있습니까? 예를 들어,이메일 시스템을 사용할 수 없게 되거나 급여 시스템에서 트랜잭션을 제출할 수 없게 되면 직원 생산성에 영향을 미칩니다.
9. 워크로드 RTO 및 RPO가 사업부 및 조직 DR 전략과 어떻게 연계됩니까?
10. 서비스 제공에 내부적으로 계약상의 의무가 있습니까? 그 의무를 이행하지 못하면 불이익이 있습니까?
11. 데이터에 대한 규제 또는 규정상의 제약은 무엇입니까?

구현 워크시트

구현의 2 및 3단계에 이 워크시트를 사용하세요. 필요에 따라 질문을 추가하는 등 이 워크시트를 조정할 수 있습니다.

2단계: 기본 질문	워크로드에 적용 여부	워크로드 RTO	워크로드 RPO	조정된 RTO	조정된 RPO	지침
[1] 워크로드가 다운되어도 괜찮은 최대 시간						중단 시작부터 복구까지 측정된 시간
[2] 손실되어도 되는 최대한의 데이터						마지막으로 알려진 복원 가능한 정상 데이터 세트 이후로 측정된 시간
[3a] 업스트림 종속성						가장 엄격한 업스트림 복구 목표 입력
[3b] 다운스트림 종속성						가장 덜 엄격한 다운스트림 복구 목표 입력
[3a] 조정된 업스트림 종속성						업스트림 값이 현재 값보다 적고 다운스트림 값이 더 크면 종속성을 조정하고
[3b] 조정된 다운스트림 종속성						조정된 값을 여기에 입력
[3] 종속성						업스트림 종속성을 충족하도록 값을 줄이거나 다운스트림 종속성 기능에 따라 값 늘리기
2단계: 추가 질문						질문이 적용되면 표기. 질문이 적용되지 않으면 건너뜀
기본 RTO/RPO						위의 RTO 및 RPO 값을 여기로 가져오기
[4] 중단의 유형	[ ]Y / [ ]N					요구 사항이 가장 엄격한 이벤트 유형에 대한 복구 목표 입력
[5] 구체적인 시간 목표	[ ]Y / [ ]N					요구 사항이 가장 엄격한 시기에 대한 복구 목표 입력
[6] 중단된 고객	[ ]Y / [ ]N					가동 중지 시간 또는 데이터 손실을 기준으로 영향을 받은 고객의 그래프 작성. 이 그래프를 사용하여 고객 영향에 따라 수용 가능한 최대 RTO 및 RPO 입력
[7] 평판에 미치는 영향	[ ]Y / [ ]N					비즈니스와 조율하여 평판에 미치는 영향에 따라 최대 RTO 및 RPO 결정
[8] 운영에 미치는 영향	[ ]Y / [ ]N					운영에 미치는 영향에 따라 최대 RTO 및 RPO 입력
[9] 조직 차원의 조율	[ ]Y / [ ]N					LOB 및 조직 요구 사항에 따라 이 워크로드 유형의 최대 RTO 및 RPO 입력
[10] 계약상 의무	[ ]Y / [ ]N					계약상의 의무에 따라 최대 RTO 및 RPO 입력
[11] 규제 준수	[ ]Y / [ ]N					해당하는 규제 준수 요구 사항에 따라 최대 RTO 및 RPO 입력
추가 질문에 따른 타겟						4~11번 질문의 최솟값(더 엄격한 값)을 여기에 입력
조정된 타겟						위의 목표를 달성할 수 없는 경우 이해 관계자들과 조율하여 제약을 완화하거나 새로운 최솟값을 여기에 입력
조정된 RTO/RPO						기본 RPO/RTO 값 또는 조정된 타겟 중 더 낮은 값 입력
3단계						
사전 정의된 범주 또는 티어에 매핑						정의된 티어 중 가장 근접한 티어와 일치하도록 두 값을 아래로(더 엄격하게) 조정

워크시트

구현 계획의 작업 수준: 낮음

리소스

관련 모범 사례:

- the section called “REL09-BP04 백업 무결성 및 프로세스를 확인하기 위해 데이터의 주기적인 복구 수행”
- the section called “REL13-BP02 복구 목표 달성을 위해 정의된 복구 전략 사용”
- the section called “REL13-BP03 재해 복구 구현을 테스트하여 구현 확인”

관련 문서:

- AWS 아키텍처 블로그: 재해 복구 시리즈
- AWS에서 워크로드의 재해 복구: 클라우드에서의 복구(AWS 백서)
- AWS 복원력 허브로 복원력 정책 관리
- APN 파트너: 재해 복구를 지원할 수 있는 파트너
- AWS Marketplace: 재해 복구에 사용할 수 있는 제품

관련 동영상:

- AWS re:Invent 2018: 다중 리전 액티브-액티브 애플리케이션을 위한 아키텍처 패턴(ARC209-R2)
- AWS에서 워크로드의 재해 복구

REL13-BP02 복구 목표 달성을 위해 정의된 복구 전략 사용

워크로드의 복구 목표에 부합하는 재해 복구(DR) 전략을 정의합니다. 백업 및 복원, 대기(액티브/패시브), 액티브/액티브 등의 전략을 선택합니다.

원하는 결과: 각 워크로드에 대해 워크로드가 DR 목표를 달성하도록 하는 DR 전략이 정의되고 구현되어 있습니다. 워크로드 간의 DR 전략은 재사용 가능한 패턴(예: 이전에 설명한 전략)을 활용합니다.

일반적인 안티 패턴:

- DR 목표가 유사한 워크로드에 일관적이지 않은 복구 절차를 구현합니다.
- 재해가 발생했을 때 DR 전략이 임시로 구현되도록 합니다.
- 재해 복구 계획을 마련하지 않았습니다.
- 복구 시 컨트롤 플레인 작업에 의존합니다.

이 모범 사례 확립의 이점:

- 정의된 복구 전략을 사용하면 공통적인 도구 및 테스트 절차를 사용할 수 있습니다.
- 정의된 복구 전략을 사용하면 팀 간의 지식 공유와 팀이 소유한 워크로드에 대한 DR 구현이 향상됩니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음 DR 전략을 계획, 구현, 테스트하지 않으면 재해 발생 시 복구 목표를 달성하지 못할 가능성이 큽니다.

구현 가이드

DR 전략은 기본 위치에서 워크로드를 실행할 수 없게 되었을 때 복구 사이트에서 워크로드를 실행하는 능력에 달려 있습니다. 가장 흔한 복구 목표는 다음에서 논의한 RTO 와 RPO입니다 REL13-BP01 가동 중단 시간 및 데이터 손실 시의 복구 목표 정의단축할 수 있습니다.

하나의 AWS 리전 내에서 여러 가용 영역(AZ)에 걸친 DR 전략은 화재, 홍수, 대규모의 정전과 같은 재해 이벤트 시 피해를 완화해 줍니다. 워크로드를 특정 AWS 리전에서 실행할 수 없게 되는 흔치 않은 이벤트에 대한 예방 조치를 구현하는 것이 요구 사항이라면 여러 리전을 사용하는 DR 전략을 사용할 수 있습니다.

여러 리전에 걸쳐 DR 전략을 아키텍팅할 때 다음 전략 중 하나를 사용해야 합니다. 전략은 복잡성과 비용이 증가하고 RTO 및 RPO가 감소하는 순서로 나열되어 있습니다. 복구 리전이란 워크로드에 사용한 기본 리전이 아닌 다른 AWS 리전을 말합니다.

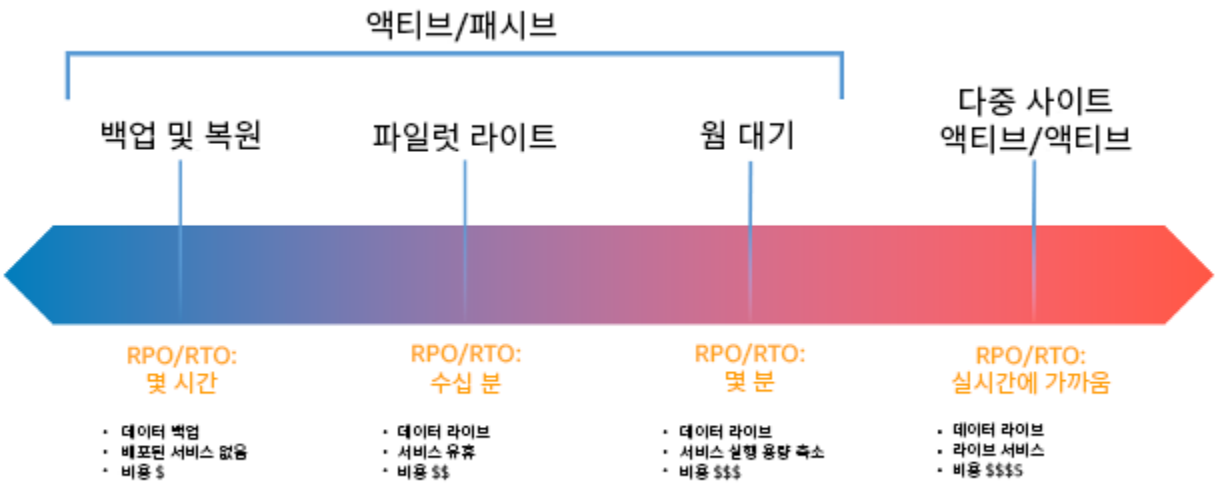


그림 17: 재해 복구(DR) 전략

- 백업 및 복원(시간 단위 RPO, 24시간 이하의 RTO): 데이터와 애플리케이션을 DR 리전에 백업합니다. 자동화된 백업 또는 지속적인 백업을 사용하면 특정 시점 복구가 가능하여 경우에 따라서는

RPO를 5분까지 줄일 수 있습니다. 재해 이벤트 시 인프라를 배포하고(RTO를 단축하기 위해 코드형 인프라 사용), 코드를 배포하고, 백업 데이터를 복원하여 복구 리전에서 재해로부터 복구합니다.

- 파일럿 라이트(분 단위 RPO, 10분 단위 RTO): 코어 워크로드의 인프라 복사본을 복구 리전에 프로비저닝합니다. 데이터를 복구 리전에 복제하고 복구 리전에서 백업을 생성합니다. 데이터베이스 및 객체 스토리지 등 데이터 복제 및 백업을 지원하는 데 필요한 리소스가 항상 실행됩니다. 애플리케이션 서버 또는 서버리스 컴퓨팅과 같은 기타 요소는 배포되지 않지만 필요에 따라 필수 구성 및 애플리케이션 코드로 생성될 수 있습니다.
- 웜 대기(초 단위 RPO, 분 단위 RTO): 모든 기능을 갖추었지만 축소된 버전의 워크로드를 복구 리전에 항상 실행되는 상태로 유지합니다. 비즈니스 크리티컬 시스템은 완전히 복제되고 항상 실행되지만 플릿은 축소됩니다. 데이터가 복구 리전에 복제되며 실행됩니다. 복구 시기가 되면 시스템은 프로덕션 로드를 처리하기 위해 신속하게 확장됩니다. 웜 대기가 확장될수록 RTO 및 컨트롤 플레인 의존도는 낮아집니다. 완전히 확장되면 이것을 상시 대기 방식이라고 합니다.
- 다중 리전(다중 사이트) 액티브/액티브(0에 가까운 RPO, 0일 수 있는 RTO): 워크로드가 여러 AWS 리전에 배포되고 능동적으로 트래픽을 처리합니다. 이 전략을 사용하려면 리전 전체에서 데이터를 동기화해야 합니다. 서로 다른 두 개 리전에 있는 복제본에 같은 레코드를 쓸 때 나타날 수 있는 충돌을 피하거나 처리해야 하는데, 그 방법이 복잡할 수 있습니다. 데이터 복제는 데이터 동기화에 유용하며 일부 유형의 재해로부터 보호해 주지만, 솔루션에 특정 시점 복구 옵션이 포함되지 않은 이상 데이터 손상 또는 중단으로부터 보호해 주지는 않습니다.

> **Note**
>
> 파일럿 라이트와 웜 대기 간의 차이를 이해하기 어려울 수 있습니다. 둘 다 복구 리전에 속한 환경과 기본 리전 자산의 복사본을 포함합니다. 차이점은 파일럿 라이트의 경우 먼저 추가 조치를 취하지 않으면 요청을 처리할 수 없지만 웜 대기는 축소된 용량 수준으로 트래픽을 즉시 처리할 수 있다는 점입니다. 파일럿 라이트를 사용하려면 서버를 켜야 하고 코어 인프라가 아닌 인프라를 추가로 배포해야 할 수 있으며 스케일 업해야 합니다. 반면 웜 대기를 사용하려면 스케일 업만 하면 됩니다. 다른 것은 이미 모두 배포되고 실행되는 상태입니다. RTO 및 RPO 요구 사항에 따라 두 전략 중에서 선택하십시오.
>
> 비용이 문제이고 웜 대기 전략에 정의된 것과 유사한 RPO 및 RTO 목표를 달성하려는 경우 파일럿 라이트 접근 방식을 취하고 향상된 RPO 및 RTO 목표를 제공하는 AWS Elastic Disaster Recovery와 같은 클라우드 네이티브 솔루션을 고려할 수 있습니다

구현 단계

1. 이 워크로드의 복구 요구 사항을 충족하는 DR 전략을 결정합니다.

DR 전략을 선택할 때는 가동 중단 시간과 데이터 손실을 줄이는 것(RTO 및 RPO)과 전략 구현의 비용과 복잡성을 줄이는 것 사이에서 절충해야 합니다. 필요 이상으로 엄중한 전략을 구현하지 말아야 합니다. 불필요한 비용이 발생하기 때문입니다.

예를 들어, 다음 다이어그램에서 비즈니스는 허용 가능한 최대 RTO와 서비스 복원 전략에 지출할 수 있는 비용의 한계를 결정했습니다. 비즈니스의 목표를 감안할 때 파일럿 라이트 또는 웜 대기 DR 전략이 RTO와 비용 기준을 둘 다 만족시킵니다.

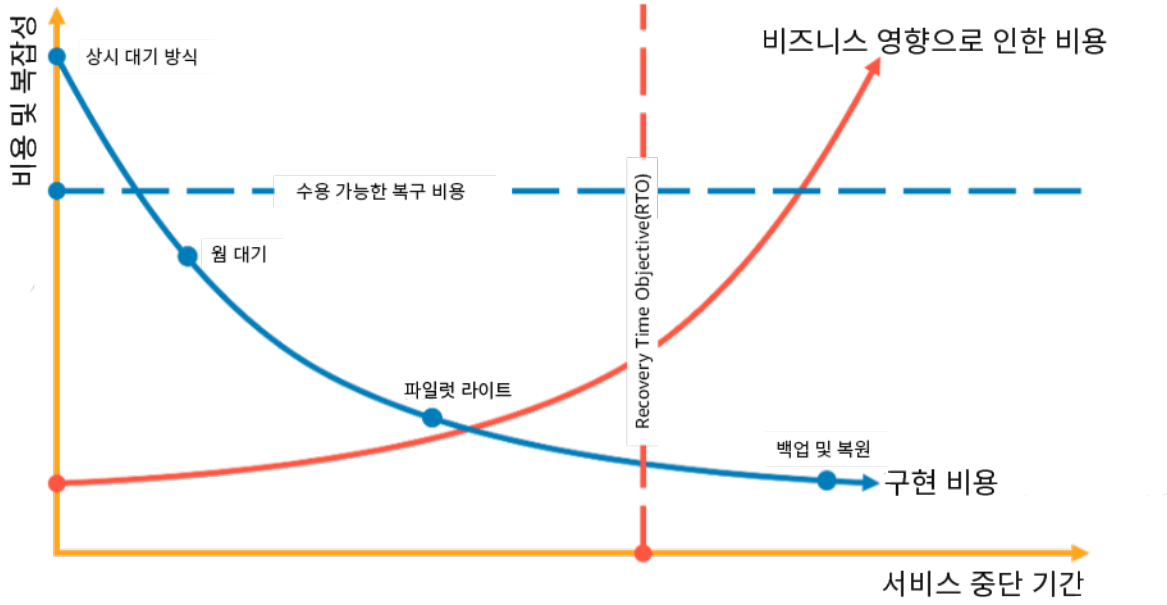


그림 18: RTO와 비용에 따라 DR 전략 선택

자세한 내용은 비즈니스 연속성 계획(BCP)을 참조하세요.

2. 선택한 DR 전략이 구현될 수 있는 패턴을 검토합니다.

이 단계는 선택한 전략을 구현하는 방법을 파악하기 위한 것입니다. 전략은 AWS 리전을 기본 사이트 및 복구 사이트로 사용하여 설명되어 있습니다. 그러나 하나의 리전 내에서 DR 전략으로 가용 영역을 선택할 수도 있습니다. 그런 경우 이런 전략 중 여러 개를 활용하게 됩니다.

다음 단계에서는 특정 워크로드에 전략을 적용할 수 있습니다.

백업 및 복구

백업 및 복구는 구현하기에 가장 덜 복잡하지만, 워크로드를 복원하는 데 더 많은 시간과 노력이 필요하므로 RTO와 RPO가 높아지는 전략입니다. 항상 데이터의 백업을 만들어 다른 사이트(예: 다른 AWS 리전)에 복사해 두는 것이 좋습니다.

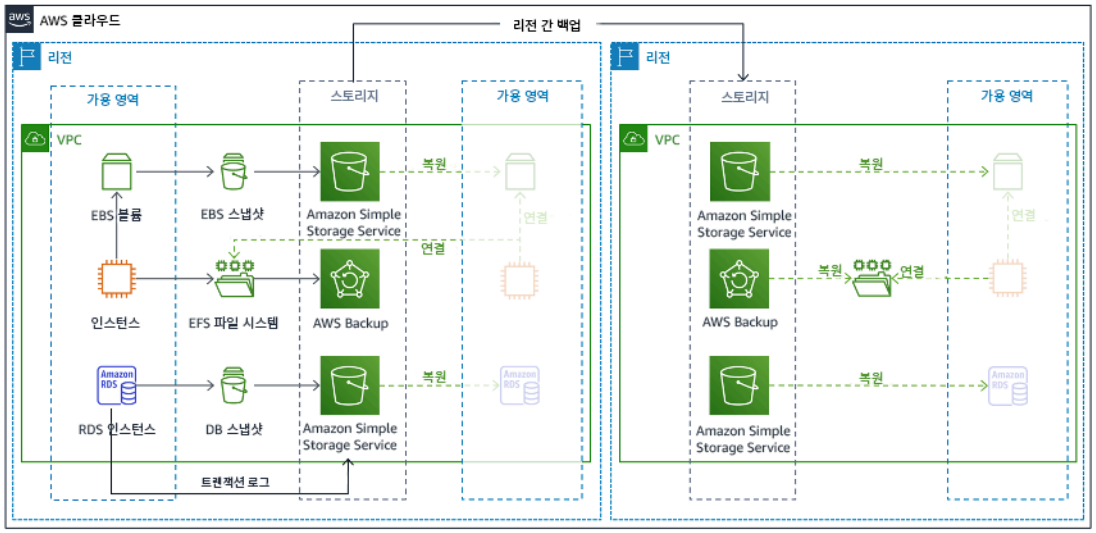


그림 19: 백업 및 복원 아키텍처

자세한 내용은 Disaster Recovery (DR) Architecture on AWS, Part II: Backup and Restore with Rapid Recovery(AWS에서의 재해 복구(DR) 아키텍처, 파트 II: 빠른 복구와 백업 및 복원)를 참조하세요

파일럿 라이트

파일럿 라이트 접근 방식에서는 기본 리전에서 복구 리전으로 데이터를 복제합니다. 워크로드 인프라에 사용되는 코어 리소스가 복구 리전에 배포되지만 기능하는 스택이 되려면 기타 리소스 및 그 종속성이 여전히 필요합니다. 예를 들어 그림 20에서는 컴퓨팅 인스턴스가 배포되지 않았습니다.

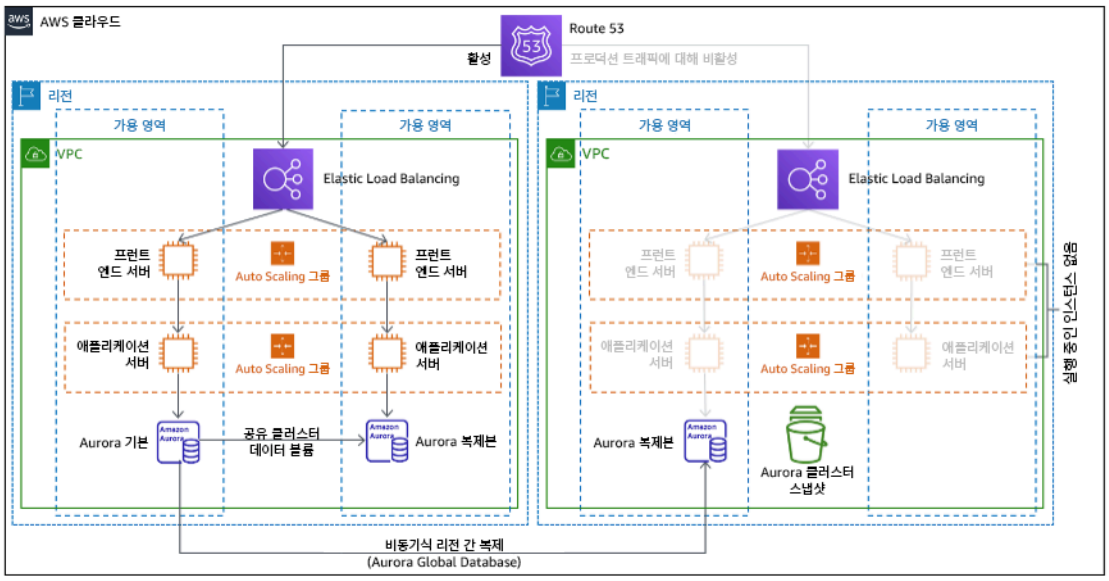


그림 20: 파일럿 라이트 아키텍처

자세한 내용은 Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby(AWS에서의 재해 복구(DR) 아키텍처, 파트 III: 파일럿 라이트 및 웜 대기)를 참조하세요

웜 대기

웜 대기 접근법에서는 스케일 다운되었지만 완전히 기능하는 프로덕션 환경의 복사본이 다른 리전에 복사됩니다. 이 접근법은 파일럿 라이트의 개념을 확대하고 복구 시간을 단축합니다. 워크로드가 다른 리전에서 상시 실행되기 때문입니다. 복구 리전이 완전한 용량으로 배포되면 이것을 상시 대기 방식이라고 합니다.

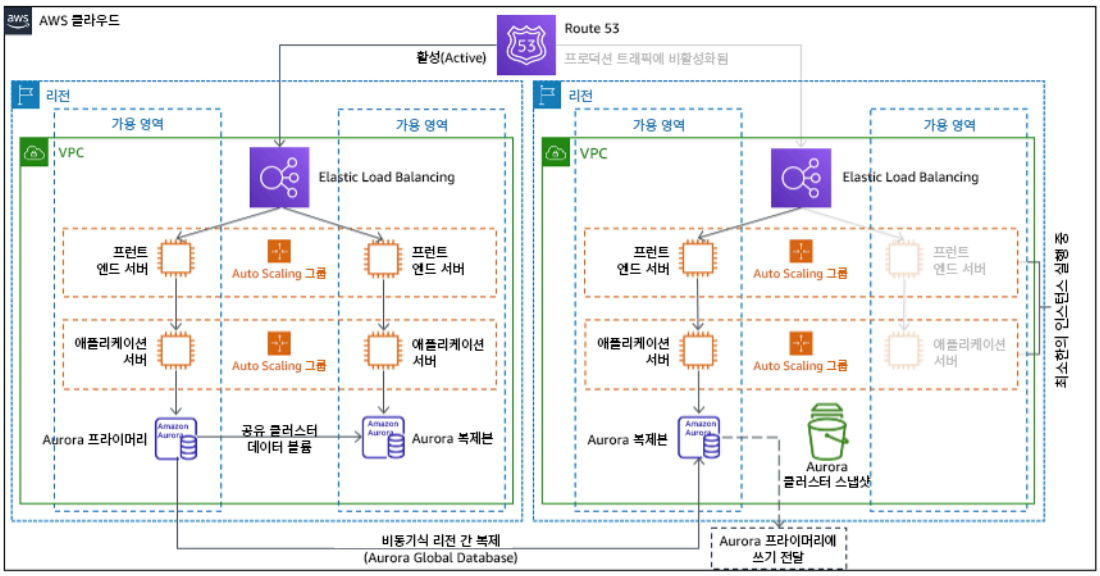


그림 21: 웜 대기 아키텍처

웜 대기 또는 파일럿 라이트를 사용하려면 복구 리전에서 리소스를 스케일 업해야 합니다. 필요할 때 용량을 사용할 수 있는지 확인하려면 EC2 인스턴스에 대한 용량 예약을 사용하세요. AWS Lambda를 사용하는 경우 프로비저닝된 동시성은 함수 호출에 즉시 응답할 준비가 되도록 실행 환경을 제공할 수 있습니다.

자세한 내용은 Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby(AWS에서의 재해 복구(DR) 아키텍처, 파트 III: 파일럿 라이트 및 웜 대기)를 참조하세요

다중 사이트 액티브/액티브

다중 사이트 액티브/액티브 전략의 일부로 여러 리전에서 워크로드를 동시에 실행할 수 있습니다. 다중 사이트 액티브/액티브는 배포된 모든 리전에서 트래픽을 처리합니다. 고객은 DR과 다른 이유를 이 전략을 선택할 수 있습니다. 이 전략은 가용성을 높이기 위해서 또는 글로벌 사용자에게 워크로드를 배포하는 경우(사용자에게 엔드포인트를 가까이 가져가거나 해당 리전의 사용자에게 현지화된 스택을 배

포하기 위해) 사용될 수 있습니다. DR 전략으로, 워크로드가 배포된 AWS 리전 중 하나에서 지원되지 않는다면 해당 리전이 철수되며 가용성을 유지하는 데 나머지 리전이 사용됩니다. 다중 사이트 액티브/액티브는 DR 전략 중 운영 측면에서 가장 복잡하며 비즈니스 요구 사항에 따라 필요한 경우에만 선택해야 합니다.

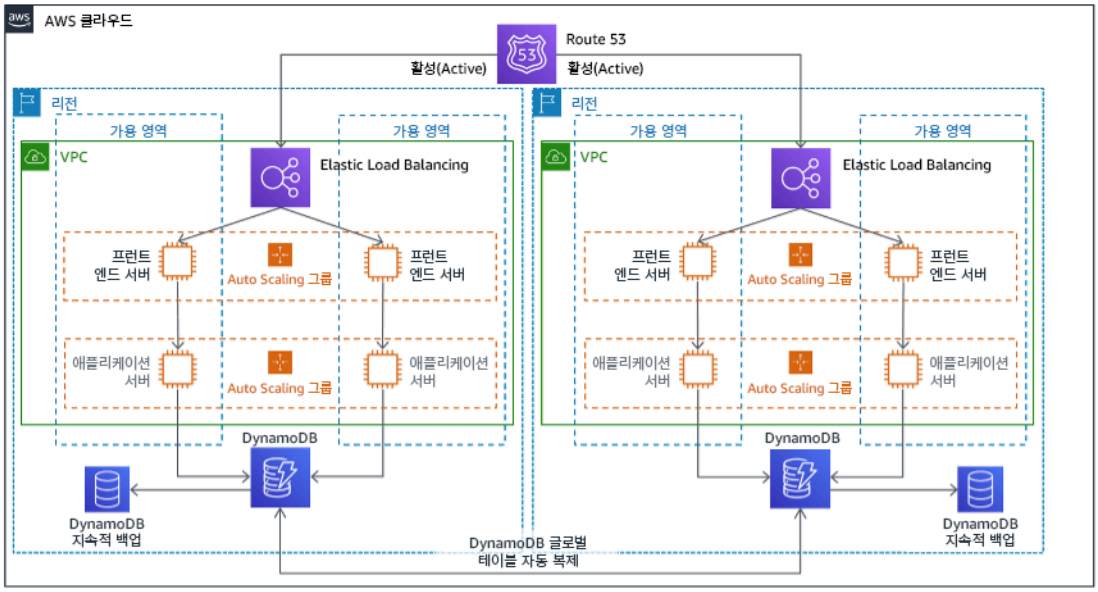


그림 22: 다중 사이트 액티브/액티브 아키텍처

이 전략에 대한 자세한 내용은 Disaster Recovery (DR) Architecture on AWS, Part IV: Multi-site Active/Active(AWS에서의 재해 복구(DR) 아키텍처, 파트 IV: 다중 사이트 액티브/액티브 단원)를 참조하세요

AWS Elastic Disaster Recovery

재해 복구를 위해 파일럿 라이트 또는 웜 대기 전략을 고려하고 있다면, AWS Elastic Disaster Recovery는 이점이 개선된 대체 접근 방식을 제공할 수 있습니다. Elastic Disaster Recovery는 웜 대기와 유사한 RPO 및 RTO 목표를 제공할 수 있지만, 파일럿 라이트의 저비용 접근 방식을 유지합니다. Elastic Disaster Recovery는 초 단위의 RPO와 분 단위의 RTO를 달성하기 위해 지속적인 데이터 보호를 사용하여 기본 리전에서 복구 리전으로 데이터를 복제합니다. 데이터를 복제하는 데 필요한 리소스만 복구 영역에 배포되므로 파일럿 라이트 전략과 유사하게 비용이 절감됩니다. Elastic Disaster Recovery를 사용할 때 서비스는 장애 조치 또는 복구 드릴의 일부로 시작될 때 컴퓨팅 리소스의 복구를 조정하고 오케스트레이션합니다.

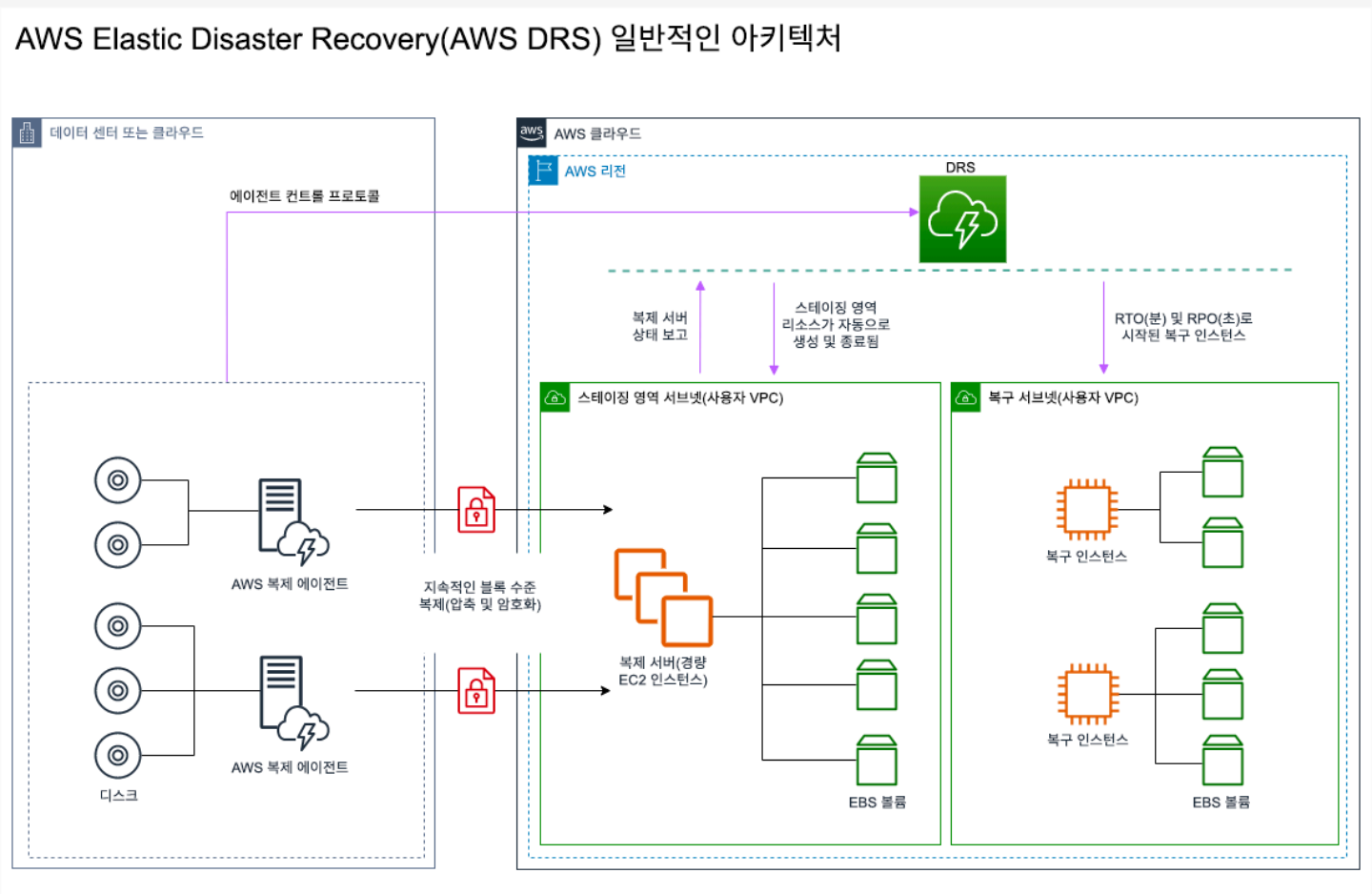


그림 23: AWS Elastic Disaster Recovery 아키텍처

데이터 보호를 위한 기타 관행

모든 전략에서 데이터 재해를 완화해야 합니다. 지속적인 데이터 복제는 일부 유형의 재해로부터 보호해 주지만, 전략에 저장된 데이터의 버전 관리 또는 특정 시점 복구 옵션이 포함되지 않은 이상 데이터 손상 또는 중단으로부터 보호해 주지는 않습니다. 복구 사이트에서 복제된 데이터를 백업하여 복제본 외에도 특정 시점 백업을 생성해야 합니다.

하나의 AWS 리전에서 여러 가용 영역(AZ) 사용

하나의 리전에서 여러 개의 AZ를 사용하면 DR 구현에서 위 전략 중 여러 요소를 사용하게 됩니다. 먼저, 그림 23에서처럼 여러 개의 AZ를 사용하여 고가용성(HA) 아키텍처를 생성해야 합니다. 이 아키텍처는 Amazon EC2 인스턴스와 Elastic Load Balancer가 여러 AZ에 리소스를 배포하여 적극적으로 요청을 처리하므로 다중 사이트 활성/활성 접근 방식을 사용합니다. 이 아키텍처는 또한 기본Amazon RDS 인스턴스에 장애가 발생하면(또는 AZ 자체에 장애가 발생하면) 대기 인스턴스가 기본 인스턴스로 승격되는 상시 대기 방식도 보여줍니다.

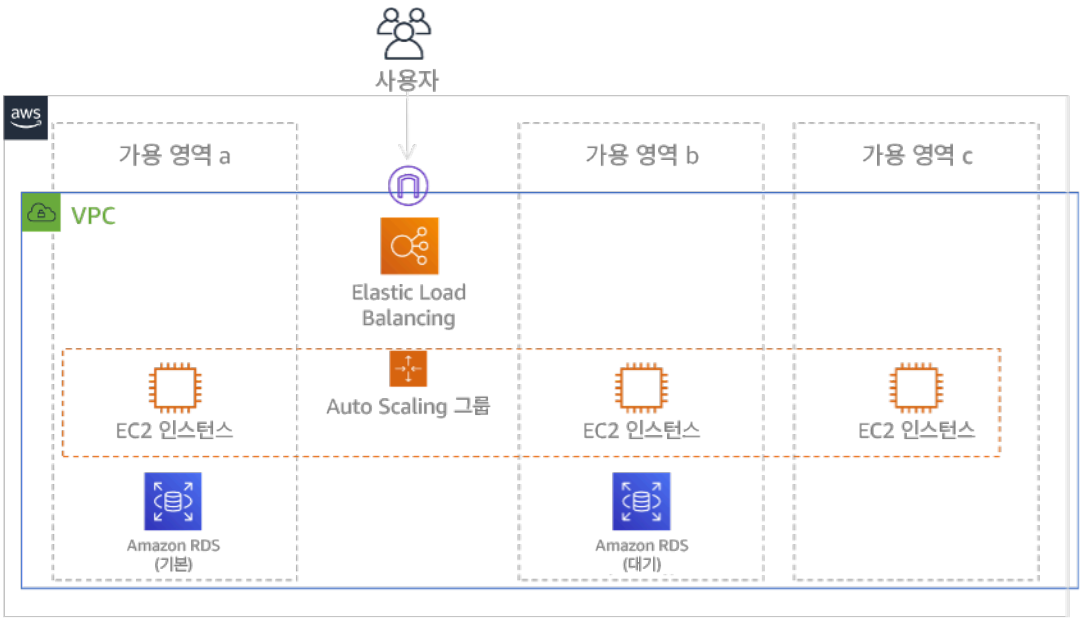


그림 24: 다중 AZ 아키텍처

이 HA 아키텍처 외에도 워크로드를 실행하는 데 필요한 모든 데이터의 백업을 추가해야 합니다. 이는 Amazon EBS 볼륨 또는 Amazon Redshift 클러스터와 같은 단일 영역으로 제한된 데이터에 특히 중요합니다. AZ에 장애가 발생하면 이 데이터를 다른 AZ에 복원해야 합니다. 가능하다면 추가적인 보호 조치로 데이터 백업을 다른 AWS 리전에 복사해야 합니다.

단일 리전에 대한 덜 일반적인 대안인 다중 AZ DR은 블로그 게시물, Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 1: Single-Region stack(Amazon Route53 Application Recovery Controller를 사용하여 복원력이 뛰어난 애플리케이션 구축, 파트 1: 단일 리전 스택)에 설명되어 있습니다. 여기에 나오는 전략은 리전의 작동 방식처럼 AZ 간에 가능한 한 많은 격리를 유지하기 위한 것입니다. 이 대안을 사용하면 액티브/액티브 또는 액티브/패시브 접근법 중에서 선택할 수 있습니다.

> **Note**
>
> 일부 워크로드에는 규제 데이터 상주 요구 사항이 적용됩니다. 현재 하나의 AWS 리전만 있는 근처 워크로드에 상주 요구 사항이 적용되는 경우 복수 리전이 비즈니스 요구 사항에 적합하지 않을 수 있습니다. 다중 AZ 전략은 대부분의 재해로부터 안전하게 보호해 줍니다.

3. 워크로드의 리소스를 평가하고 장애 조치 전에(정상 작동 시) 복구 리전에서 워크로드 리소스의 구성을 평가합니다.

인프라 및 AWS 리소스의 경우 AWS CloudFormation과 같은 코드형 인프라 또는 Hashicorp Terraform과 같은 타사 도구를 사용합니다. 한 번의 작업으로 여러 계정 및 리전에 배포하려면 AWS CloudFormation StackSets를 사용할 수 있습니다. 다중 사이트 액티브/액티브 및 상시 대기 방식 전략의 경우 복구 리전에 배포된 인프라의 리소스는 기본 리전의 리소스와 같습니다. 파일럿 라이트 및 웜 대기 전략의 경우 배포된 인프라가 프로덕션으로 사용되려면 추가 조치가 필요합니다. CloudFormation 파라미터와 조건부 로직을 사용하면 단일 템플릿으로 배포된 스택이 액티브 상태인지 대기 상태인지 제어할 수 있습니다. Elastic Disaster Recovery를 사용할 때 서비스는 애플리케이션 구성 및 컴퓨팅 리소스의 복원을 복제하고 오케스트레이션합니다.

모든 DR 전략에서는 데이터 소스가 AWS 리전 내에서 백업된 다음 해당 백업이 복구 리전으로 복사되어야 합니다. AWS Backup은 이러한 리소스에 대한 백업을 구성, 예약 및 모니터링할 수 있는 중앙 집중식 보기를 제공합니다. 파일럿 라이트, 웜 대기 및 다중 사이트 액티브/액티브의 경우 기본 리전의 데이터를 Amazon Relational Database Service(Amazon RDS) DB 인스턴스 또는 Amazon DynamoDB 테이블과 같은 복구 리전의 데이터 리소스로 복제해야 합니다. 그래야 이런 데이터 리소스가 복구 리전에서 실행되고 요청을 처리할 준비가 됩니다.

여러 리전에서 AWS 서비스가 작동하는 방식에 대해 자세히 알아보려면 AWS 서비스로 다중 리전 애플리케이션 생성에 대한 이 블로그 시리즈를 참조하세요.

4. 필요할 경우 장애 조치 시(재해 이벤트 시) 복구 리전을 어떻게 준비할지 결정하고 구현합니다.

다중 사이트 액티브/액티브의 경우 장애 조치는 한 리전을 철수하고 나머지 액티브 리전에 의존하는 것입니다. 일반적으로 이러한 리전은 트래픽을 받을 준비가 되어 있습니다. 파일럿 라이트 및 웜 대기 전략의 경우 복구 조치로 그림 20의 EC2 인스턴스와 같은 누락된 리소스와 기타 누락된 리소스를 배포해야 합니다.

위에 설명된 모든 전략에서 데이터베이스의 읽기 전용 인스턴스를 기본 읽기/쓰기 인스턴스로 승격해야 합니다.

백업 및 복원의 경우 백업에서 데이터를 복원하면 해당 데이터에 대해 EBS 볼륨, RDS DB 인스턴스, DynamoDB 테이블 등의 리소스가 생성됩니다. 또한 인프라와 배포 코드를 복원해야 합니다. AWS Backup을 사용하여 복구 리전에서 데이터를 복원할 수 있습니다. 참조 REL09-BP01 백업해야 하는 모든 데이터 확인 및 백업 또는 소스에서 데이터 복제 를 참조하세요. 인프라 재구축에는 필요한 Amazon Virtual Private Cloud(Amazon VPC), 서브넷 및 보안 그룹 외에 EC2 인스턴스와 같은 리소스 생성이 포함됩니다. 이러한 복원 작업의 대부분은 자동화할 수 있습니다. 방법을 알아보려면 이 블로그 게시물을 참조하세요.

5. 필요할 경우(재해 이벤트 시) 트래픽을 장애 조치로 어떻게 다시 라우팅할지 결정하고 구현합니다.

이 장애 조치 작업은 자동 또는 수동으로 시작할 수 있습니다. 상태 확인 또는 경보를 기반으로 자동으로 시작된 장애 조치는 신중하게 사용해야 합니다. 불필요한 장애 조치(거짓 경보)는 비가용성 및 데이터 손실과 같은 비용을 발생시키기 때문입니다. 따라서 수동으로 시작된 장애 조치를 자주 사용합니다. 이 경우에도 여전히 장애 조치 단계는 자동화하여 수동 시작은 버튼을 누르는 것 정도가 되도록 해야 합니다.

AWS 서비스를 사용할 때는 몇 가지 트래픽 관리 옵션이 있습니다. 옵션 중 하나는 Amazon Route 53을 사용하는 것입니다. Amazon Route 53을 사용하면 하나 또는 그 이상의 AWS 리전에서 여러 개의 IP 엔드포인트를 하나의 Route 53 도메인 이름과 연결할 수 있습니다. 수동으로 시작되는 장애 조치를 구현하려면 트래픽을 복구 리전으로 다시 라우팅하는 고가용성 데이터 영역 API를 제공하는 Amazon Route 53 Application Recovery Controller를 사용할 수 있습니다. 장애 조치를 구현할 때 데이터 영역 작업을 사용하고 REL11-BP04 복구 중 컨트롤 플레인이 아닌 데이터 영역 사용에 설명된 대로 컨트롤 플레인 작업은 피하세요.

이것과 다른 옵션에 대해 자세히 알아보려면 재해 복구 백서의 이 섹션을 참조하세요.

6. 워크로드의 페일백을 위한 계획을 수립합니다.

페일백은 재해 이벤트가 수그러든 후 워크로드 작업을 기본 리전으로 돌려놓는 것입니다. 인프라 및 코드를 기본 리전에 프로비저닝하는 작업은 일반적으로 처음 사용된 것과 같은 단계를 따르며 코드형 인프라 및 코드 배포 파이프라인을 사용합니다. 페일백의 어려운 점은 데이터 스토어 복원과 작동하는 복구 리전에서 그 일관성을 확보하는 것입니다.

장애 조치된 상태에서는 복구 리전에서 데이터베이스가 실행되고 복구 리전에 최신 데이터가 있게 됩니다. 이때 목표는 복구 리전에서 기본 리전으로 재동기화하여 최신 상태를 확보하는 것입니다.

일부 AWS 서비스에서는 이 작업이 자동으로 이루어집니다. Amazon DynamoDB 글로벌 테이블을 사용한다면 기본 리전의 테이블을 사용할 수 없게 되어도 온라인 상태로 돌아오면 DynamoDB가 보류 중인 쓰기 작업의 전파를 재개합니다. Amazon Aurora 글로벌 데이터베이스를 사용하고 계획된 관리형 장애 조치를 사용하는 경우 Aurora 글로벌 데이터베이스의 기존 복제 토폴로지가 유지됩니다. 따라서 기본 리전에 있는 이전의 읽기/쓰기 인스턴스가 복제본이 되고 복구 리전에서 업데이트를 수신합니다.

이 작업이 자동화되지 않는 경우 기본 리전에서 복구 리전의 데이터베이스의 복제본으로 데이터베이스를 다시 구축해야 합니다. 이때 이전의 기본 데이터베이스가 삭제되고 새로운 복제본이 생성되는 경우가 많습니다. 예를 들어 계획되지 않은 장애 조치를 가정한 Amazon Aurora 글로벌 데이터베이스로 이 작업을 수행하는 방법에 대한 지침은 이 실습: Fail Back a Global Database(글로벌 데이터베이스 페일백)를 참조하세요.

장애 조치 후 복구 리전에서 계속 실행할 수 있는 경우 이 리전을 새로운 기본 리전으로 만드는 것이 좋습니다. 이전의 기본 리전을 복구 리전으로 만들려면 위의 단계를 모두 따르면 됩니다. 일부 조직에서는 예약된 교체를 수행하여 기본 및 복구 리전을 주기적으로(예: 3개월마다) 교체합니다.

장애 조치 및 장애 복구가 필요한 모든 단계는 팀의 모든 멤버가 사용할 수 있는 플레이북에 유지 관리해야 하며 주기적으로 검토해야 합니다.

Elastic Disaster Recovery를 사용할 때 서비스는 페일백 프로세스를 오케스트레이션하고 자동화하는 데 도움이 됩니다. 자세한 내용은 페일백 수행을 참조하세요.

구현 계획의 작업 수준: 높음

리소스

관련 모범 사례:

- the section called “REL09-BP01 백업해야 하는 모든 데이터 확인 및 백업 또는 소스에서 데이터 복제”
- the section called “REL11-BP04 복구 중 컨트롤 플레인이 아닌 데이터 영역 사용”
- the section called “REL13-BP01 가동 중단 시간 및 데이터 손실 시의 복구 목표 정의”

관련 문서:

- AWS 아키텍처 블로그: 재해 복구 시리즈
- AWS에서 워크로드의 재해 복구: 클라우드에서의 복구(AWS 백서)
- 클라우드에서의 재해 복구 옵션
- Build a serverless multi-region, active-active backend solution in an hour(한 시간 내에 서버리스 다중 리전, 활성/활성 백엔드 솔루션 구축)
- Multi-region serverless backend — reloaded(다중 리전 서버리스 백엔드 - 다시 로드됨)
- RDS: 리전 간 읽기 전용 복제본 복제
- Route 53: DNS 장애 조치 구성
- S3: 리전 간 복제
- AWS Backup란 무엇입니까?
- What Route 53 is Application Recovery Controller?(Amazon Route 53 Application Recovery Controller란 무엇입니까?)
- AWS Elastic Disaster Recovery

- HashiCorp Terraform: 시작하기 - AWS
- APN 파트너: 재해 복구를 지원할 수 있는 파트너
- AWS Marketplace: 재해 복구에 사용할 수 있는 제품

관련 동영상:

- Disaster Recovery of Workloads on AWS(AWS에서 워크로드의 재해 복구)
- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications(ARC209-R2)(다중 리전 액티브-액티브 애플리케이션을 위한 아키텍처 패턴)(ARC209-R2)
- AWS Elastic Disaster Recovery 시작하기 | Amazon Web Services

관련 예시:

- Well-Architected 실습 - 재해 복구 - DR 전략을 설명하는 일련의 워크숍

REL13-BP03 재해 복구 구현을 테스트하여 구현 확인

복구 사이트에 대한 장애 조치를 정기적으로 테스트하여 제대로 작동하고 RTO 및 RPO가 충족되는지 확인합니다.

일반적인 안티 패턴:

- 프로덕션 환경에서 장애 조치를 테스트하지 않음

이 모범 사례 확립의 이점: 재해 복구 계획을 정기적으로 테스트하면 필요할 때 제대로 작동하도록 보장하고 팀이 전략 실행 방법을 숙지하고 있는지 확인할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

거의 사용되지 않는 복구 경로를 개발하는 것은 피해야 할 패턴입니다. 읽기 전용 쿼리에 사용되는 보조 데이터 스토어를 예로 들 수 있습니다. 데이터 스토어에 데이터를 쓸 때 기본 스토어에서 장애가 발생하면 보조 데이터 스토어로 장애 조치를 진행할 수 있습니다. 이 장애 조치를 자주 테스트하지 않으면 보조 데이터 스토어의 기능에 대한 가정이 잘못될 수 있습니다. 예를 들어 마지막으로 테스트했을 때는 보조 용량이 충분했지만 이 시나리오에서는 더 이상 로드를 모두 처리하지 못할 수도 있습니다. 경험에 따르면 자주 테스트하는 경로만이 유일하게 작동하는 오류 복구 방법입니다. 이러한 이유로 인해 복구 경로를 적게 갖는 것이 가장 좋습니다. 복구 패턴을 설정하고 정기적으로 테스트할 수 있습니

다. 복잡하거나 중요한 복구 경로가 있는 경우 해당 복구 경로의 작동을 확신하기 위해 프로덕션 환경에서 해당 장애를 정기적으로 연습해야 합니다. 앞에서 설명한 예의 경우에는 필요 여부에 관계없이 대기 스토어로 정기 장애 조치를 수행해야 합니다.

구현 단계

1. 복구가 가능하도록 워크로드를 설계합니다. 복구 경로를 정기적으로 테스트합니다. 복구 지향 컴퓨팅은 복구를 향상시키는 시스템의 특성을 식별합니다. 이러한 특성으로는 격리 및 중복성, 변경 사항을 롤백하는 시스템 전체 기능, 상태 모니터링 및 결정 기능, 진단 제공 기능, 자동 복구, 모듈식 설계 및 재시작 기능 등이 있습니다. 지정된 시간에 지정한 상태로 복구를 수행할 수 있도록 복구 경로에 대해 연습하세요. 이 복구 과정에 런북을 사용하여 문제를 문서화하고 다음 테스트 전에 해결 방법을 찾습니다.
2. Amazon EC2 기반 워크로드의 경우 AWS Elastic Disaster Recovery를 사용하여 DR 전략을 위한 드릴 인스턴스를 구현하고 시작합니다. AWS Elastic Disaster Recovery는 훈련을 효율적으로 실행할 수 있는 기능을 제공하여 장애 조치 이벤트를 준비하는 데 도움이 됩니다. 트래픽을 리디렉션하지 않고 테스트 및 드릴 목적으로 Elastic Disaster Recovery를 사용하여 인스턴스를 자주 시작할 수도 있습니다.

리소스

관련 문서:

- APN 파트너: 재해 복구를 지원할 수 있는 파트너
- AWS 아키텍처 블로그: 재해 복구 시리즈
- AWS Marketplace: 재해 복구에 사용할 수 있는 제품
- AWS Elastic Disaster Recovery
- AWS에서 워크로드의 재해 복구: 클라우드에서의 복구(AWS 백서)
- AWS Elastic Disaster Recovery 장애 조치 준비
- The Berkeley/Stanford recovery-oriented computing project(Berkeley/Stanford 복구 중심 컴퓨팅 프로젝트)
- AWSFault Injection Simulator란 무엇입니까?

관련 동영상:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications(다중 리전 액티브-액티브 애플리케이션을 위한 아키텍처 패턴)

- AWS re:Invent 2019: Backup-and-restore and disaster-recovery solutions with AWS(AWS의 백업 및 복원과 재해 복구 솔루션)

관련 예시:

- Well-Architected 실습 - 복원력 테스트

REL13-BP04 DR 사이트 또는 리전에서 구성 드리프트 관리

DR 사이트 또는 리전에 필요한 인프라, 데이터 및 구성이 갖추어져 있는지 확인합니다. 예를 들어 AMI와 Service Quotas가 최신 상태인지 확인합니다.

AWS Config는 AWS 리소스 구성을 지속적으로 모니터링하고 기록합니다. 이 서비스를 사용하면 드리프트를 감지하고, AWS Systems Manager Automation 을 트리거하여 드리프트를 수정한 후, 경보를 생성할 수 있습니다. AWS CloudFormation은 배포된 스택의 드리프트를 추가로 감지할 수 있습니다.

일반적인 안티 패턴:

- 기본 위치에서 구성을 생성하거나 인프라를 변경할 때 복구 위치에서 업데이트에 실패함
- 기본 및 복구 위치에서 잠재적 제한(예: 서비스의 차이)을 고려하지 않음

이 모범 사례 수립의 이점: DR 환경이 기존 환경과 일치하는지 확인하면 완전한 복구를 보장할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 배포 파이프라인이 기본 사이트와 백업 사이트 모두에 제공되는지 확인합니다. 애플리케이션을 프로덕션에 배포하기 위한 배포 파이프라인은 개발 및 테스트 환경을 포함하여 지정된 모든 재해 복구 전략 위치에 배포해야 합니다.
- AWS Config를 활성화하여 잠재적 드리프트 위치를 추적합니다. AWS Config 규칙을 사용하여 재해 복구 전략을 실행하고 드리프트가 감지되면 알림을 생성하는 시스템을 구축합니다.
  - AWS Config 규칙에 따른 규정 미준수 AWS 리소스 문제 해결
  - AWS Systems Manager Automation
- AWS CloudFormation을 사용하여 인프라를 배포합니다. AWS CloudFormation은 CloudFormation 템플릿에 명시된 내용과 실제로 배포된 항목 사이의 드리프트를 감지할 수 있습니다.

- AWS CloudFormation: 전체 CloudFormation 스택의 드리프트 감지

리소스

관련 문서:

- APN 파트너: 재해 복구를 지원할 수 있는 파트너
- AWS 아키텍처 블로그: 재해 복구 시리즈
- AWS CloudFormation: 전체 CloudFormation 스택의 드리프트 감지
- AWS Marketplace: 재해 복구에 사용할 수 있는 제품
- AWS Systems Manager Automation
- AWS에서 워크로드의 재해 복구: 클라우드에서의 복구(AWS 백서)
- AWS에서 인프라 구성 관리 솔루션을 구현하려면 어떻게 해야 합니까?
- AWS Config 규칙에 따른 규정 미준수 AWS 리소스 문제 해결

관련 동영상:

- AWS re:Invent 2018: 다중 리전 액티브-액티브 애플리케이션을 위한 아키텍처 패턴(ARC209-R2)

REL13-BP05 복구 자동화

AWS 또는 서드 파티 도구를 사용하여 시스템 복구를 자동화하고 트래픽을 DR 사이트 또는 리전으로 라우팅합니다.

구성된 상태 확인에 따라 Elastic Load Balancing 및 AWS Auto Scaling과 같은 AWS 서비스를 사용하여 정상 상태인 가용 영역에 로드를 분산하고, Amazon Route 53 및 AWS Global Accelerator와 같은 서비스를 사용하여 정상 상태인 AWS 리전으로 로드를 라우팅할 수 있습니다. Amazon Route 53 Application Recovery Controller를 사용하면 준비 상태 확인 및 라우팅 제어 기능을 통해 장애 조치를 관리하고 조정할 수 있습니다. 이러한 기능은 애플리케이션의 장애 복구 성능을 지속적으로 모니터링하므로 이를 통해 여러 AWS 리전, 가용 영역 및 온프레미스에서 애플리케이션 복구를 제어할 수 있습니다.

기존의 물리적 또는 가상 데이터 센터 또는 프라이빗 클라우드의 워크로드의 경우 AWS Elastic Disaster Recovery를(AWS Marketplace를 통해 제공) 사용하면 조직에서 AWS에 자동화된 재해 복구 전략을 설정할 수 있습니다. CloudEndure는 AWS에서 교차 리전/교차 AZ 재해 복구도 지원합니다.

일반적인 안티 패턴:

- 자동 장애 조치와 자동 장애 복구를 동일하게 구현하면 장애가 발생할 때 플래핑을 유발할 수 있습니다.

이 모범 사례 수립의 이점: 자동 복구는 수작업으로 인한 오류 발생 가능성을 없앰으로써 복구 시간을 단축합니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

- 복구 경로를 자동화합니다. 짧은 복구 시간이 요구되는 고가용성 시나리오에서는 인간의 판단 및 수작업을 사용할 수 없습니다. 모든 상황에서 시스템이 자동으로 복구되어야 합니다.
  - 자동 장애 조치 및 장애 복구를 위해 CloudEndure Disaster Recovery를 사용합니다. CloudEndure Disaster Recovery는 시스템(운영 체제, 시스템 상태 구성, 데이터베이스, 애플리케이션 및 파일 포함)을 대상 AWS 계정과 원하는 리전의 저렴한 스테이징 영역으로 지속적으로 복제합니다. 재해가 발생할 경우 몇 분 만에 수천 대의 시스템을 자동으로 완전히 프로비저닝된 상태로 시작하도록 CloudEndure Disaster Recovery에 지시할 수 있습니다.
    - Performing a Disaster Recovery Failover and Failback(재해 복구를 위한 조치 및 절차 수행)
    - CloudEndure Disaster Recovery

리소스

관련 문서:

- APN 파트너: 재해 복구를 지원할 수 있는 파트너
- AWS 아키텍처 블로그: 재해 복구 시리즈
- AWS Marketplace: 재해 복구에 사용할 수 있는 제품
- AWS Systems Manager Automation
- AWS로의 CloudEndure Disaster Recovery
- Disaster Recovery of Workloads on AWS: Recovery in the Cloud(AWS에서 워크로드의 재해 복구: 클라우드에서의 복구)(AWS 백서)

관련 동영상:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications(다중 리전 액티브-액티브 애플리케이션을 위한 아키텍처 패턴)(ARC209-R2)

성능 효율성

성능 효율성 원칙은 컴퓨팅 리소스를 시스템 요구 사항에 맞게 효율적으로 사용하고, 수요 변화 및 기술 진화에 발맞춰 그러한 효율성을 유지할 수 있는 역량을 포함합니다. 구현 방법에 대한 권장 가이드는 성능 효율성 원칙 백서에서 확인할 수 있습니다.

모범 사례 영역

- 선택
- 검토(Review)
- 모니터링
- 절충

선택

질문

- PERF 1 최고의 성능을 제공하는 아키텍처를 선택하려면 어떻게 해야 합니까?
- PERF 2 컴퓨팅 솔루션을 어떻게 선택합니까?
- PERF 3 스토리지 솔루션을 어떻게 선택합니까?
- PERF 4 데이터베이스 솔루션을 어떻게 선택합니까?
- PERF 5 네트워킹 솔루션을 구성하려면 어떻게 해야 합니까?

PERF 1 최고의 성능을 제공하는 아키텍처를 선택하려면 어떻게 해야 합니까?

워크로드에서 최적의 성능을 얻으려면 여러 접근 방식을 취해야 합니다. Well-Architected 시스템은 다수의 솔루션 및 기능을 사용하여 성능을 개선합니다.

모범 사례

- PERF01-BP01 사용 가능한 서비스 및 리소스 파악
- PERF01-BP02 아키텍처를 선택하는 프로세스 정의
- PERF01-BP03 비용 요구 사항을 고려한 의사 결정
- PERF01-BP04 정책 또는 참조 아키텍처 사용
- PERF01-BP05 클라우드 공급자 또는 해당하는 파트너의 지침 사용
- PERF01-BP06 기존 워크로드 벤치마크

- PERF01-BP07 워크로드 로드 테스트

PERF01-BP01 사용 가능한 서비스 및 리소스 파악

클라우드에서 사용 가능한 방대한 서비스와 리소스에 대해 알아보고 이해합니다. 워크로드와 관련이 있는 서비스 및 구성 옵션을 확인하고, 성능을 최적화하는 방법을 파악합니다.

기존 워크로드를 평가하는 경우에는 워크로드에 사용되는 다양한 서비스 리소스의 인벤토리를 생성해야 합니다. 인벤토리는 관리형 서비스 및 최신 기술로 교체 가능한 구성 요소를 평가하는 데 도움이 됩니다.

일반적인 안티 패턴:

- 클라우드를 코로케이션된 데이터 센터로 사용합니다.
- 영구 스토리지를 필요로 하는 모든 것에 공유 스토리지를 사용합니다.
- 자동 조정을 사용하지 않습니다.
- 현재 표준에 가장 근접한 인스턴스 유형을 사용하지만 필요한 경우 더 큰 인스턴스 유형을 사용합니다.
- 관리형 서비스로 사용할 수 있는 기술을 배포하고 관리합니다.

이 모범 사례 수립의 이점: 잘 모르는 서비스를 고려할 경우 인프라 비용과 서비스 유지 관리에 필요한 노력이 크게 줄어들 가능성이 있습니다. 새로운 서비스와 기능을 배포하여 출시 시간을 단축할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

관련 서비스에 대한 워크로드 소프트웨어 및 아키텍처 인벤토리 생성: 워크로드의 인벤토리를 수집하고 자세히 알아볼 제품 범주를 결정합니다. 관리형 서비스로 대체하여 성능을 개선하고 운영 복잡성을 줄일 수 있는 워크로드 구성 요소를 식별합니다.

리소스

관련 문서:

- AWS 아키텍처 센터
- AWS Partner Network
- AWS 솔루션 라이브러리

- AWS 지식 센터

관련 동영상:

- Amazon Builders' Library 소개(DOP328)
- This is my Architecture

관련 예시:

- AWS 샘플
- AWS SDK 예시

PERF01-BP02 아키텍처를 선택하는 프로세스 정의

클라우드에 대한 내부 경험과 지식을 사용하거나 게시된 사용 사례, 관련 설명서 또는 백서 등의 외부 리소스를 사용하여 리소스 및 서비스를 선택하는 프로세스를 정의합니다. 워크로드에 사용될 수 있는 서비스는 테스트하고 벤치마킹할 수 있도록 장려하는 프로세스를 정의해야 합니다.

아키텍처에 대한 중요한 사용자 사례를 작성할 때는 각 중요 사례를 얼마나 빠르게 실행했는지 명시하는 등 성능 요구 사항을 포함해야 합니다. 이러한 중요 사례에서는 이런 사례들이 요구 사항을 어떻게 만족시키는지 가시성을 확보하기 위한 추가적인 스크립트들이 구현되어야 합니다.

일반적인 안티 패턴:

- 시간이 지나면 현재 아키텍처가 정적 아키텍처가 되고 업데이트되지 않는다고 가정합니다.
- 시간이 지나면 타당한 이유 없이 아키텍처 변경을 도입합니다.

이 모범 사례 정립의 이점: 아키텍처 변경을 위한 프로세스를 정의하면 수집된 데이터를 사용하여 워크로드 설계에 지속적으로 영향을 줄 수 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

아키텍처 접근 방식 선택: 성능 요구 사항을 충족하는 아키텍처의 종류를 식별합니다. 전송 미디어(데스크톱, 웹, 모바일, IoT), 기존 요구 사항 및 통합 등의 제약 조건을 식별합니다. 리팩터링을 포함한 재사용 기회를 식별합니다. AWS 솔루션스 아키텍트, AWS 참조 아키텍처 및 AWS 파트너와 같은 다른 팀, 아키텍처 다이어그램 및 리소스를 참조하면 아키텍처 선택에 도움을 받을 수 있습니다.

성능 요구 사항 정의: 고객 경험을 바탕으로 가장 중요한 지표를 식별합니다. 각 지표에 대해 목표, 측정 방식 및 우선 순위를 정합니다. 고객 경험을 정의합니다. 고객이 요구하는 성능 경험 및 고객이 워크로드의 성능을 판단하는 근거를 문서화합니다. 중요한 사용자 사례에 대한 경험 문제를 우선적으로 처리합니다. 성능 요구 사항을 포함하고 스크립트로 작성된 사용자 여정을 구현하여 이러한 사례의 성능이 요구 사항에 부합하는지 확인합니다.

리소스

관련 문서:

- AWS 아키텍처 센터
- AWS Partner Network
- AWS Solutions Library
- AWS 지식 센터

관련 동영상:

- Amazon Builders' Library 소개(DOP328)
- This is my Architecture

관련 예시:

- AWS 샘플
- AWS SDK 예시

PERF01-BP03 비용 요구 사항을 고려한 의사 결정

워크로드에는 작업에 대한 비용 요구 사항이 있는 경우가 많습니다. 내부 비용 제어 기능을 사용하여 예상되는 리소스 요구 사항에 적합한 유형 및 크기의 리소스를 선택하십시오.

관리형 데이터베이스, 인 메모리 캐시 및 ETL 서비스와 같은 완전관리형 서비스로 대체할 수 있는 워크로드 구성 요소를 결정하십시오. 운영 워크로드를 줄이면 비즈니스 결과에 리소스를 집중시킬 수 있습니다.

비용 요구 사항 모범 사례는 다음 백서에서 비용 효율적인 리소스 섹션을 참조하십시오. 비용 최적화 원칙 백서.

일반적인 안티 패턴:

- 인스턴스 패밀리는 하나만 사용합니다.
- 라이선스가 부여된 솔루션과 오픈 소스 솔루션을 비교 평가하지 않습니다.
- 블록 스토리지만 사용합니다.
- 관리형 서비스로 사용 가능한 EC2 인스턴스 및 Amazon EBS 또는 휘발성 볼륨에 공통 소프트웨어를 배포합니다.

이 모범 사례 정립의 이점: 선택할 때 비용을 고려하면 다른 투자가 가능해집니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

워크로드 구성 요소를 최적화하여 비용 절감: 워크로드 구성 요소를 적정 크기로 조정하고 탄력성을 활성화하여 비용을 절감하고 구성 요소 효율성을 극대화합니다. 해당하는 경우 관리형 데이터베이스, 인메모리 캐시 및 역방향 프록시와 같은 관리형 서비스로 대체할 수 있는 워크로드 구성 요소를 결정합니다.

리소스

관련 문서:

- AWS 아키텍처 센터
- AWS Partner Network
- AWS Solutions Library
- AWS 지식 센터
- AWS Compute Optimizer

관련 동영상:

- Amazon Builders' Library 소개(DOP328)
- This is my Architecture
- Optimize performance and cost for your AWS compute(CMP323-R1)

관련 예시:

- AWS 샘플
- AWS SDK 예시

- Compute Optimizer 및 메모리 활용 지원을 통해 적절한 규모 결정
- AWS Compute Optimizer 데모 코드

PERF01-BP04 정책 또는 참조 아키텍처 사용

내부 정책 및 기존 참조 아키텍처를 평가하고 분석을 사용하여 워크로드에 사용할 서비스 및 구성을 선택하면 성능 및 효율성을 극대화할 수 있습니다.

일반적인 안티 패턴:

- 회사의 관리 오버헤드에 영향을 줄 수 있는 기술 선택의 광범위한 사용을 허용합니다.

이 모범 사례 수립의 이점: 아키텍처, 기술 및 공급업체 선택에 대한 정책을 수립하면 신속하게 의사 결정을 내릴 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

기존 정책 또는 참조 아키텍처를 사용하여 워크로드 배포: 서비스를 클라우드 배포에 통합한 다음 성능 테스트를 사용하여 성능 요구 사항을 계속해서 충족할 수 있는지 확인합니다.

리소스

관련 문서:

- AWS 아키텍처 센터
- AWS Partner Network
- AWS 솔루션 라이브러리
- AWS 지식 센터

관련 동영상:

- Amazon Builders' Library 소개(DOP328)
- This is my Architecture

관련 예시:

- AWS 샘플

- AWS SDK 예시

PERF01-BP05 클라우드 공급자 또는 해당하는 파트너의 지침 사용

클라우드 회사 리소스(예: 솔루션스 아키텍트, 전문 서비스 또는 적절한 파트너)를 의사 결정의 지침으로 사용하십시오. 이러한 리소스는 최적의 성능을 위해 아키텍처를 검토하고 개선하는 데 도움이 될 수 있습니다.

추가 지침 또는 제품 정보가 필요한 경우 AWS에 문의하여 지원을 받으십시오. AWS 솔루션스 아키텍트 및 AWS Professional Services 는 솔루션 구현에 대한 지침을 제공합니다. AWS 파트너 는 비즈니스 민첩성 및 혁신을 달성하는 데 도움이 되는 AWS 전문 지식을 제공합니다.

일반적인 안티 패턴:

- AWS를 일반적인 데이터 센터 공급자로 사용합니다.
- AWS 서비스를 설계 의도와 다른 방식으로 사용합니다.

이 모범 사례 수립의 이점: 공급자 또는 파트너와 상담하면 의사 결정에서 확신을 얻을 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

AWS 리소스에 문의하여 지원 요청: AWS 솔루션스 아키텍트 및 Professional Services는 솔루션 구현에 대한 지침을 제공합니다. APN 파트너는 비즈니스 민첩성 및 혁신을 달성하는 데 도움이 되는 AWS 전문 지식을 제공합니다.

리소스

관련 문서:

- AWS 아키텍처 센터
- AWS Partner Network
- AWS Solutions Library
- AWS 지식 센터

관련 동영상:

- Amazon Builders' Library 소개(DOP328)

- This is my Architecture

관련 예시:

- AWS 샘플
- AWS SDK 예시

PERF01-BP06 기존 워크로드 벤치마크

기존 워크로드의 성능을 벤치마크하여 클라우드에서의 성능을 파악합니다. 이러한 벤치마크에서 수집된 데이터를 사용하면 아키텍처를 원활하게 결정할 수 있습니다.

통합 테스트 및 실제 사용자 모니터링과 함께 벤치마킹을 활용하여 워크로드 구성 요소의 성능에 대한 데이터를 생성합니다. 벤치마킹은 대개 로드 테스트보다 빠르게 설정할 수 있으며, 특정 구성 요소의 기술을 평가할 때 사용됩니다. 벤치마킹은 새 프로젝트를 시작할 때 로드 테스트를 위한 완전한 솔루션이 부족한 경우 종종 사용됩니다.

사용자 지정 벤치마크 테스트를 직접 작성할 수도 있고 TPC-DS 등의 업계 표준 테스트를 사용하여 데이터 웨어하우징 워크로드를 벤치마킹할 수도 있습니다. 산업 벤치마크는 여러 환경을 비교할 때 유용합니다. 사용자 지정 벤치마크는 아키텍처 내에서 수행될 것으로 예상되는 특정 작업 유형을 타게팅하는 데 유용합니다.

벤치마킹을 사용할 때는 유효한 결과를 얻을 수 있도록 테스트 환경을 사전 준비하는 것이 중요합니다. 동일한 벤치마크를 여러 번 실행하여 시간대별 차이를 파악하십시오.

벤치마크는 대개 로드 테스트보다 더 빠르게 실행되므로 배포 파이프라인 초기에 성능 편차 관련 피드백을 더 빠르게 제공하려는 경우에 사용할 수 있습니다. 구성 요소나 서비스의 큰 변화를 평가할 때 벤치마킹을 수행하면 변경 작업의 타당성을 빠르게 확인할 수 있습니다. 벤치마킹은 로드 테스트와 함께 사용해야 합니다. 로드 테스트에서는 프로덕션 환경의 워크로드 성능에 대한 정보를 얻을 수 있기 때문입니다.

일반적인 안티 패턴:

- 워크로드 특성을 나타내지 않는 일반적인 벤치마크를 사용합니다.
- 고객의 피드백과 관점을 유일한 벤치마크로 삼습니다.

이 모범 사례 수립의 이점: 현재 구현을 벤치마킹하면 성능 개선을 측정할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

개발 중 성능 모니터링: 워크로드가 변경될 때의 성능을 파악할 수 있는 프로세스를 구현합니다.

전송 파이프라인에 통합: 전송 파이프라인에서 로드 테스트를 자동으로 실행합니다. 미리 정의된 KPI(핵심 성능 지표) 및 임계값을 기준으로 테스트 결과를 비교하여 성능 요구 사항이 계속 충족되는지 확인합니다.

사용자 여정 테스트: 로드 테스트에는 프로덕션 데이터의 통합 또는 제거 버전(민감한 정보 또는 식별 정보 제거)을 사용하십시오. 대규모 애플리케이션에서 재생 또는 사전 프로그래밍된 사용자 여정을 사용하여 전체 아키텍처를 실행합니다.

실제 사용자 모니터링: CloudWatch RUM을 사용하면 애플리케이션 성능에 대한 클라이언트 측 데이터를 수집하고 확인하는 데 도움이 됩니다. 이 데이터를 활용하여 실제 사용자 성능 벤치마크를 설정할 수 있습니다.

리소스

관련 문서:

- AWS 아키텍처 센터
- AWS Partner Network
- AWS 솔루션 라이브러리
- AWS 지식 센터
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics

관련 동영상:

- Amazon Builders' Library 소개(DOP328)
- This is my Architecture
- Amazon CloudWatch RUM을 통한 애플리케이션 최적화
- Amazon CloudWatch Synthetics 데모

관련 예시:

- AWS 샘플

- AWS SDK 예시
- 분산 로드 테스트
- Amazon CloudWatch Synthetics를 활용한 페이지 로드 시간 측정
- Amazon CloudWatch RUM 웹 클라이언트

PERF01-BP07 워크로드 로드 테스트

다양한 리소스 유형 및 크기의 최신 워크로드 아키텍처를 클라우드에 배포합니다. 배포를 모니터링하여 병목 현상 또는 초과 용량을 식별하는 성능 지표를 캡처합니다. 이 성능 정보를 사용하여 아키텍처 및 리소스 선택을 설계하거나 개선할 수 있습니다.

로드 테스트에는 실제 워크로드가 사용되므로 프로덕션 환경에서 솔루션의 성능을 확인할 수 있습니다. 로드 테스트는 프로덕션 데이터의 통합 또는 제거 버전(민감한 정보 또는 식별 정보 제거)을 사용하여 실행되어야 합니다. 또한 대규모 워크로드를 통해 전체 아키텍처에서 진행되는 재생/사전 프로그래밍 방식의 사용자 작업 과정을 사용합니다. 전송 파이프라인의 일부로 로드 테스트를 자동으로 수행하고 결과를 미리 정의된 KPI 및 임계값과 비교합니다. 이렇게 하면 필요한 성능을 계속해서 달성할 수 있습니다.

일반적인 안티 패턴:

- 전체 워크로드가 아니라 워크로드의 개별 부분에 대해 로드 테스트를 수행합니다.
- 프로덕션 환경과 동일하지 않은 인프라에서 로드 테스트를 수행합니다.
- 향후 문제가 발생할 수 있는 위치를 예측할 때 예상 로드에 대해서만 로드 테스트를 수행합니다.
- AWS Support에 알리지 않은 채로 로드 테스트를 수행하고, 서비스 거부 이벤트처럼 보여 테스트에 실패한 것으로 처리합니다.

이 모범 사례 수립의 이점: 로드 테스트에서 성능을 측정하면 로드 증가의 영향을 받게 될 위치를 알 수 있습니다. 이렇게 하면 워크로드에 영향을 미치기 전에 필요한 변경 사항을 예상할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

로드 테스트를 수행하여 선택한 접근 방식 검증: 개념 증명에 대한 로드 테스트를 수행하여 성능 요구 사항이 충족되는지 확인합니다. AWS 서비스를 사용하면 프로덕션 규모의 환경을 실행하여 아키텍처를 테스트할 수 있습니다. 테스트 환경이 필요할 때에만 비용을 지불하므로 온프레미스 환경을 사용하는 데 드는 비용의 일부로 필요한 모든 테스트를 수행할 수 있습니다.

지표 모니터링: Amazon CloudWatch는 아키텍처의 리소스 전반에서 지표를 수집할 수 있습니다. 또한 사용자 지정 지표를 수집하고 게시하여 비즈니스 또는 파생 지표를 파악할 수도 있습니다. CloudWatch 또는 타사 솔루션을 사용하여 임계값 위반 시점을 나타내는 경보를 설정합니다.

대규모 데스트: 로드 테스트에는 실제 워크로드가 사용되므로 프로덕션 환경에서 솔루션의 성능을 확인할 수 있습니다. AWS 서비스를 사용하면 프로덕션 규모의 환경을 실행하여 아키텍처를 테스트할 수 있습니다. 테스트 환경이 필요할 때만 비용을 지불하므로 온프레미스 환경을 사용할 때보다 저렴한 비용으로 전체 테스트를 실행할 수 있습니다. AWS 클라우드를 활용하여 워크로드를 테스트하면 확장에 실패한 위치 또는 비선형 방식으로 확장되는 위치를 찾을 수 있습니다. 예를 들어, 스팟 인스턴스를 사용하여 저렴한 비용으로 로드를 생성하고 프로덕션 환경에서 발생하기 전에 병목 현상을 발견할 수 있습니다.

리소스

관련 문서:

- AWS CloudFormation
- CloudFormer를 사용한 AWS CloudFormation 템플릿 구축
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- AWS에서의 분산 로드 테스트

관련 동영상:

- Amazon Builders' Library 소개(DOP328)
- Amazon CloudWatch RUM을 통한 애플리케이션 최적화
- Amazon CloudWatch Synthetics 데모

관련 예시:

- AWS에서의 분산 로드 테스트

PERF 2 컴퓨팅 솔루션을 어떻게 선택합니까?

워크로드에 가장 적합한 컴퓨팅 솔루션은 애플리케이션 설계, 사용량 패턴 및 구성 설정에 따라 다릅니다. 아키텍처는 다양한 구성 요소에 대해 서로 다른 컴퓨팅 솔루션을 사용하고 다양한 기능을 활성화하

여 성능을 개선할 수 있습니다. 아키텍처에 대해 잘못된 컴퓨팅 솔루션을 선택하면 성능 효율성 저하로 이어질 수 있습니다.

모범 사례

- PERF02-BP01 사용 가능한 컴퓨팅 옵션 평가
- PERF02-BP02 사용 가능한 컴퓨팅 구성 옵션 파악
- PERF02-BP03 컴퓨팅 관련 지표 수집
- PERF02-BP04 적정 크기 조정으로 필요한 구성 확인
- PERF02-BP05 사용 가능한 리소스 탄력성 사용
- PERF02-BP06 지표를 기준으로 컴퓨팅 요구 사항의 지속적 평가

PERF02-BP01 사용 가능한 컴퓨팅 옵션 평가

워크로드가 인스턴스, 컨테이너 및 함수와 같은 다양한 컴퓨팅 옵션을 사용하여 어떻게 이점을 얻을 수 있는지 알아봅니다.

원하는 결과: 사용 가능한 모든 컴퓨팅 옵션을 이해하여 성능을 개선하고, 불필요한 인프라 비용을 절감하며, 워크로드를 유지하는 데 필요한 운영 노력을 줄일 기회를 파악할 수 있습니다. 또한 새로운 서비스와 기능을 배포할 때 출시 시간을 단축할 수 있습니다.

일반적인 안티 패턴:

- 마이그레이션 후 워크로드에 온프레미스에서 사용하던 것과 동일한 컴퓨팅 솔루션을 활용합니다.
- 클라우드 컴퓨팅 솔루션과 이러한 솔루션이 컴퓨팅 성능을 어떻게 개선할 수 있는지를 잘 모르고 있습니다.
- 대체 컴퓨팅 솔루션이 워크로드 특성에 보다 적절한데도 확장 또는 성능 요구 사항을 충족하려고 기존 컴퓨팅 솔루션의 규모를 과도하게 키웁니다.

이 모범 사례 정립의 이점: 비즈니스 이해관계자와 엔지니어링 팀이 컴퓨팅 요구 사항을 파악하고 사용 가능한 컴퓨팅 솔루션을 평가하여 선택한 컴퓨팅 솔루션을 사용할 경우의 이점과 한계를 이해할 수 있습니다. 선택한 컴퓨팅 솔루션은 워크로드 성능 기준에 적합해야 합니다. 주요 기준에는 처리 요구 사항, 트래픽 패턴, 데이터 액세스 패턴, 확장 요구 사항 및 지연 시간 요구 사항이 포함됩니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

워크로드에 도움이 되고 성능 요구 사항을 충족할 수 있는 가상화, 컨테이너화 및 관리 솔루션을 이해해야 합니다. 워크로드에는 여러 유형의 컴퓨팅 솔루션이 포함될 수 있는데 컴퓨팅 솔루션마다 특성이 다릅니다. 원하는 워크로드 규모와 컴퓨팅 요건에 따라 컴퓨팅 솔루션을 선택하고 구성하여 요구 사항을 충족할 수 있습니다. 클라우드 아키텍트는 인스턴스, 컨테이너 및 함수의 장점과 단점을 숙지해야 합니다. 다음 단계에서는 워크로드 특성 및 성능 요구 사항에 적합한 컴퓨팅 솔루션을 선택하는 방법에 대해 설명합니다.

유형	서버	컨테이너	함수
AWS 서비스	Amazon Elastic Compute Cloud(Amazon EC2)	Amazon Elastic Container Service(Amazon ECS), Amazon Elastic Kubernetes Service(Amazon EKS)	AWS Lambda
주요 특징	하드웨어 라이선스 요구 사항, 배치 옵션 및 컴퓨팅 지표를 기반으로 다양하게 선택할 수 있는 인스턴스 패밀리 전용 옵션이 있음	간편한 배포, 일관된 환경, EC2 인스턴스 위에서 실행, 확장성	짧은 런타임(15분 이하), 최대 메모리 및 CPU가 다른 서비스보다 높지 않음, 관리형 하드웨어 계층, 수백만 개의 동시 요청으로 확장 가능
일반 사용 사례	리프트 앤드 시프트 마이그레이션, 단일 애플리케이션, 하이브리드 환경, 엔터프라이즈 애플리케이션	마이크로서비스, 하이브리드 환경	마이크로서비스, 이벤트 기반 애플리케이션

구현 단계:

1. 섹션을 평가하여 컴퓨팅 솔루션이 상주해야 하는 위치를 선택합니다 the section called “PERF05-BP06 네트워크 요구 사항에 따라 워크로드의 위치 선택”. 이 위치에서는 사용 가능한 컴퓨팅 솔루션 유형이 제한됩니다.
2. 위치 및 애플리케이션 요구 사항에 맞는 컴퓨팅 솔루션 유형을 식별합니다.

a. Amazon Elastic Compute Cloud(Amazon EC2) 가상 서버 인스턴스는 폭넓은 패밀리와 크기로 제공됩니다. 이러한 인스턴스는 솔리드 스테이트 드라이브(SSD) 및 그래픽 처리 장치(GPU)를 비롯하여 다양한 기능을 제공합니다. EC2 인스턴스는 인스턴스를 선택할 때 가장 큰 유연성을 제공합니다. EC2 인스턴스를 시작할 때 지정한 인스턴스 유형에 따라 인스턴스의 하드웨어가 정해집니다. 각 인스턴스 유형은 서로 다른 컴퓨팅, 메모리 및 스토리지 기능을 제공합니다. 인스턴스 유형은 이러한 기능에 따라 인스턴스 패밀리로 그룹화됩니다. 대표적인 사용 사례로는 엔터프라이즈 애플리케이션 실행, 고성능 컴퓨팅(HPC), 기계 학습 애플리케이션 훈련 및 배포, 클라우드 네이티브 애플리케이션 실행 등이 있습니다.

b. Amazon Elastic Container Service(Amazon ECS) 는 AWS Fargate를 사용하여 EC2 인스턴스 또는 서버리스 인스턴스의 클러스터에서 컨테이너를 자동으로 실행하고 관리할 수 있는 완전 관리형 컨테이너 오케스트레이션 서비스입니다. Amazon ECS를 Amazon Route 53, Secrets Manager, AWS Identity and Access Management(IAM), Amazon CloudWatch 등 다른 서비스와 함께 사용할 수 있습니다. 애플리케이션이 컨테이너화되어 있고 엔지니어링 팀이 도커 컨테이너를 선호하는 경우 Amazon ECS를 사용하는 것이 좋습니다.

c. Amazon Elastic Kubernetes Service(Amazon EKS) 는 완전관리형 Kubernetes 서비스입니다. AWS Fargate를 사용하여 EKS 클러스터를 실행하도록 선택하면 서버를 프로비저닝하고 관리할 필요가 없습니다. Amazon EKS는 Amazon CloudWatch, 오토 스케일링, AWS Identity and Access Management(IAM), Amazon Virtual Private Cloud(VPC)와 같은 AWS 서비스에 통합되어 관리하기가 편리합니다. 컨테이너를 사용할 때는 계산 지표를 활용하여 EC2 또는 AWS Fargate 인스턴스 유형을 선택하는 방식과 마찬가지로 계산 지표를 통해 워크로드에 가장 적합한 유형을 선택해야 합니다. 애플리케이션이 컨테이너화되어 있고 엔지니어링 팀이 도커 컨테이너보다 Kubernetes를 선호하는 경우 Amazon EKS를 선택하는 것이 좋습니다.

d. 기존 파이프라인과 함께 AWS Lambda 를 사용하여 허용된 런타임, 메모리 및 CPU 옵션을 지원하는 코드를 실행할 수 있습니다. 코드를 업로드하기만 하면 AWS Lambda가 해당 코드를 실행하고 확장하는 데 필요한 모든 것을 관리합니다. 다른 AWS 서비스에서 자동으로 트리거하거나 직접 호출하도록 코드를 설정할 수 있습니다. Lambda는 클라우드용으로 개발된 단기 실행 마이크로서비스 아키텍처에 권장됩니다.

3. 새로운 컴퓨팅 솔루션을 실험한 후 마이그레이션을 계획하고 성능 지표를 검증합니다. 이는 한 번으로 끝나는 과정이 아닙니다. 다음 섹션을 참조하십시오 the section called “PERF02-BP04 적정 크기 조정으로 필요한 구성 확인”.

구현 계획의 작업 수준: 워크로드가 한 컴퓨팅 솔루션에서 다른 컴퓨팅 솔루션으로 이동하는 경우 애플리케이션을 리팩터링하는 데 보통 수준의 노력이 필요할 수 있습니다.

리소스

관련 문서:

- AWS로 클라우드 컴퓨팅
- EC2 인스턴스 유형
- EC2 인스턴스에 대한 프로세서 상태 제어
- EKS 컨테이너: EKS 워커 노드
- Amazon ECS 컨테이너: Amazon ECS 컨테이너 인스턴스
- 함수: Lambda 함수 구성
- 컨테이너 권장 가이드
- 서버리스 권장 가이드

관련 동영상:

- How to choose compute option for startups
- Optimize performance and cost for your AWS compute(CMP323-R1)
- Amazon EC2 foundations(CMP211-R2)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- Deliver high-performance ML inference with AWS Inferentia(CMP324-R1)
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2(CMP202-R1)

관련 예시:

- 웹 애플리케이션을 컨테이너로 마이그레이션
- 서버리스 Hello World 실행

PERF02-BP02 사용 가능한 컴퓨팅 구성 옵션 파악

각 컴퓨팅 솔루션에는 워크로드 특성을 지원하는 옵션과 구성이 있습니다. 다양한 옵션을 통해 워크로드를 보완할 수 있는 방식과 애플리케이션에 가장 적합한 구성 옵션을 알아봅니다. 이러한 옵션의 예로는 인스턴스 패밀리, 크기, 기능(GPU, I/O), 버스팅, 시간 초과, 함수 크기, 컨테이너 인스턴스 및 동시성이 있습니다.

원하는 결과: CPU, 메모리, 네트워크 처리량(throughput), GPU, IOPS, 트래픽 패턴, 데이터 액세스 패턴을 비롯한 워크로드 특성이 문서화되어 있어 워크로드 특성에 맞는 컴퓨팅 솔루션을 구성하는 데 사용됩니다. 이러한 각 지표와 워크로드별 사용자 지정 지표를 기록하여 모니터링한 후 요구 사항을 효과적으로 충족하도록 컴퓨팅 구성을 최적화하는 데 사용합니다.

일반적인 안티 패턴:

- 온프레미스에서 사용하던 컴퓨팅 솔루션을 그대로 활용합니다.
- 워크로드 특성에 맞게 컴퓨팅 옵션 또는 인스턴스 패밀리를 검토하지 않습니다.
- 버스팅 기능을 보장하기 위해 컴퓨팅 규모를 과도하게 키웁니다.
- 동일한 워크로드에 여러 컴퓨팅 관리 플랫폼을 사용합니다.

이 모범 사례 수립의 이점: 각 워크로드에 적합한 솔루션을 결정하려면 AWS 컴퓨팅 오퍼링에 대해 잘 알아야 합니다. 워크로드에 사용할 컴퓨팅 오퍼링을 선택한 후에는 이러한 컴퓨팅 오퍼링을 신속하게 실험하여 워크로드 요구 사항이 제대로 충족되는지 확인할 수 있습니다. 워크로드 특성에 맞게 최적화된 컴퓨팅 솔루션을 사용하면 성능이 향상되고 ,비용이 줄어들며, 신뢰성이 높아집니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

워크로드에서 4주 이상 동일한 컴퓨팅 옵션을 사용하고 있으며 앞으로도 특성이 변하지 않을 것으로 예상되는 경우 AWS Compute Optimizer 를 사용하여 컴퓨팅 특성에 따라 권장 사항을 알아볼 수 있습니다. 지표 부족, 지원되지 않는 인스턴스 유형 또는 예측 가능한 특성 변경으로 인해 AWS Compute Optimizer를 사용할 수 없는 경우 로드 테스트 및 실험을 기반으로 지표를 예측해야 합니다.

구현 단계:

1. EC2 시작 유형을 사용하여 EC2 인스턴스 또는 컨테이너에서 실행 중입니까?
    a. 워크로드에서 GPU를 사용하여 성능을 높일 수 있습니까?
        i. 가속화된 컴퓨팅 인스턴스는 기계 학습 훈련, 추론 및 고성능 컴퓨팅에 최상의 성능을 제공하는 GPU 기반 인스턴스입니다.
    b. 워크로드에서 기계 학습 추론 애플리케이션을 실행합니까?
        i. AWS Inferentia(Inf1) - Inf1 인스턴스는 기계 학습 추론 애플리케이션을 지원하도록 구축되었습니다. Inf1 인스턴스를 사용하는 고객은 이미지 인식, 음성 인식, 자연어 처리, 개인화 및 사기 탐지와 같은 대규모 기계 학습 추론 애플리케이션을 실행할 수 있습니다. TensorFlow, PyTorch 또는 MXNet과 같이 널리 사용되는 기계 학습 프레임워크 중 하나에 모델을 구축하고 GPU 인스턴스를 사용하여 모델을 훈련할 수 있습니다. 요구 사항을 충족하도록 기계 학습 모

델을 훈련한 후에는 Inferentia 칩의 기계 학습 추론 성능을 최적화하는 컴파일러, 런타임 및 프로파일링 도구로 구성된 특수 SDK(소프트웨어 개발 키트)인 AWS Neuron을 사용하여 Inf1 인스턴스에 모델을 배포할 수 있습니다.

c. 워크로드를 낮은 수준의 하드웨어와 통합하여 성능을 개선합니까?

   i. FPGA(Field Programmable Gate Array) - FPGA를 사용하면 매우 까다로운 워크로드에 사용자 지정 하드웨어 가속 실행 기능을 사용함으로써 워크로드를 최적화할 수 있습니다. C, Go 등의 지원되는 일반 프로그래밍 언어나 Verilog, VHDL 등의 하드웨어 중심 언어를 활용해 알고리즘을 정의할 수 있습니다.

d. 최소 4주간의 지표를 보유하고 있으며 향후 트래픽 패턴과 지표가 흡사하게 유지될 것으로 예측 가능합니까?

   i. 사용 Compute Optimizer 를 활용하여 사용자의 컴퓨팅 특성에 가장 알맞은 컴퓨팅 구성에 대한 기계 학습 권장 사항을 알아봅니다.

e. CPU 지표로 인해 워크로드 성능이 제한됩니까?

   i. 컴퓨팅 최적화 인스턴스는 고성능 프로세서가 필요한 워크로드에 이상적입니다.

f. 메모리 지표로 인해 워크로드 성능이 제한됩니까?

   i. 메모리 최적화 인스턴스는 상당한 메모리를 제공하여 메모리를 많이 사용하는 워크로드를 지원합니다.

g. IOPS로 인해 워크로드 성능이 제한됩니까?

   i. 스토리지 최적화 인스턴스는 로컬 스토리지에 대한 높은 순차 읽기 및 쓰기 액세스(IOPS)가 필요한 워크로드에 적합하도록 설계되었습니다.

h. 워크로드 특성에 모든 지표에 걸친 균형 잡힌 요구 사항이 반영되었습니까?

   i. 트래픽 급증을 처리하려면 워크로드 CPU를 버스트해야 합니까?

      A. 버스트 가능 성능 인스턴스는 컴퓨팅 최적화 인스턴스와 유사하지만, 컴퓨팅에 최적화된 인스턴스에서 식별된 고정 CPU 기준을 초과하여 버스트할 수 있습니다.

   ii. 범용 인스턴스는 모든 특성을 균형 있게 제공하여 다양한 워크로드를 지원합니다.

i. 컴퓨팅 인스턴스가 Linux에서 실행 중이고 네트워크 인터페이스 카드의 네트워크 처리량(throughput)으로 인해 제한을 받습니까?

   i. 검토 '성능 질문 5, 모범 사례 2: 사용 가능한 네트워킹 기능 평가'를 검토하여 성능 요구 사항을 충족하는 올바른 인스턴스 유형 및 패밀리를 찾아봅니다.

j. 워크로드에 1년 동안 사용할 수 있는 특정 가용 영역의 일관되고 예측 가능한 인스턴스가 필요합니까?

   i. 예약 인스턴스 는 특정 가용 영역에서 용량 예약을 확정합니다. 예약 인스턴스는 특정 가용 영역에서 컴퓨팅 기능이 필요한 경우에 이상적입니다.

k. 워크로드에 전용 하드웨어가 필요한 라이선스가 있습니까?

   i. 전용 호스트 는 기존 소프트웨어 라이선스를 지원하고 규정 준수 요구 사항을 충족할 수 있도록 합니다.

l. 컴퓨팅 솔루션을 버스트해야 하고 동기식 처리가 필요합니까?

   i. 온디맨드 인스턴스 를 사용하면 장기 약정 없이 시간 또는 초 단위로 컴퓨팅 용량을 이용할 수 있습니다. 이러한 인스턴스는 성능 기준 요구를 넘어서 버스팅하는 데 도움이 됩니다.

m. 스테이스리스, 내결함성 및 비동기식인 컴퓨팅 솔루션을 사용합니까?

   i. 스팟 인스턴스 를 사용하면 내결함성을 갖춘 스테이스리스 워크로드를 위해 미사용 인스턴스 용량을 활용할 수 있습니다.

2. 컨테이너를 Fargate에서 실행하고 있습니까?

   a. 메모리 또는 CPU로 인해 작업 성능이 제한됩니까?

      i. SAP 환경의 보안 관련 작업에 대한 지침은 태스크 크기 를 사용하여 메모리 또는 CPU를 조정할 수 있습니다.

   b. 성능이 트래픽 패턴 버스트의 영향을 받고 있습니까?

      i. SAP 환경의 보안 관련 작업에 대한 지침은 Auto Scaling 구성을 사용하여 트래픽 패턴을 일치시킬 수 있습니다.

3. 컴퓨팅 솔루션이 Lambda에 있습니까?

   a. 최소 4주간의 지표를 보유하고 있으며 향후 트래픽 패턴과 지표가 흡사하게 유지될 것으로 예측 가능합니까?

      i. 사용 Compute Optimizer 를 활용하여 사용자의 컴퓨팅 특성에 가장 알맞은 컴퓨팅 구성에 대한 기계 학습 권장 사항을 알아봅니다.

   b. AWS Compute Optimizer를 사용하는 데 필요한 지표가 충분하지 않습니까?

      i. Compute Optimizer를 사용하는 데 활용할 지표가 없는 경우 AWS Lambda 파워 튜닝 을 통해 최적의 구성을 선택할 수 있습니다.

   c. 메모리 또는 CPU로 인해 함수 성능이 제한됩니까?

      i. 성능 요구 지표를 충족하도록 Lambda 메모리 를 구성합니다.

   d. 함수 실행 시 시간이 초과되었습니까?

      i. 시간 초과 설정을 변경합니다.

   e. 함수 성능이 활동 및 동시성 버스트로 인해 제한됩니까?

      i. 성능 요구 사항을 충족하도록 동시성 설정 을 구성합니다.

   f. 함수가 비동기적으로 실행되며 재시도 시 실패합니까?

      i. 비동기 구성 설정에서 이벤트의 최대 사용 기간과 최대 재시도 제한을 구성합니다.

구현 계획의 작업 수준:

이 모범 사례를 적용하려면 현재 컴퓨팅 특성과 지표를 알고 있어야 합니다. 이러한 지표를 수집하고 기준선을 설정한 다음 해당 지표를 사용하여 이상적인 컴퓨팅 옵션을 식별하는 작업은 낮음 수준부터 보통 수준의 노력을 투자하여 수행됩니다. 이는 로드 테스트 및 실험을 통해 가장 효과적으로 검증됩니다.

리소스

관련 문서:

- AWS 클라우드 컴퓨팅
- AWS Compute Optimizer
- EC2 인스턴스 유형
- EC2 인스턴스에 대한 프로세서 상태 제어
- EKS 컨테이너: EKS 워커 노드
- Amazon ECS 컨테이너: Amazon ECS 컨테이너 인스턴스
- 함수: Lambda 함수 구성

관련 동영상:

- Amazon EC2 foundations(CMP211-R2)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- AWS 컴퓨팅의 성능 및 비용 최적화(CMP323-R1)

관련 예시:

- Compute Optimizer 및 메모리 사용률을 활성화하여 적절한 크기 지정
- AWS Compute Optimizer 데모 코드

PERF02-BP03 컴퓨팅 관련 지표 수집

계산 리소스의 성능을 이해하려면 다양한 시스템의 활용률을 기록하고 추적해야 합니다. 이 데이터를 사용하면 리소스 요구 사항을 보다 정확하게 파악할 수 있습니다.

워크로드를 통해 지표, 로그 및 이벤트와 같은 대량의 데이터가 생성될 수 있습니다. 기존 스토리지, 모니터링 및 관찰성 서비스가 이렇게 만들어진 데이터를 관리할 수 있는지를 결정해야 합니다. 리소스 활

용률을 반영하는 지표를 식별하고, 단일 플랫폼에서 수집 및 집계되고 상관 관계가 지정될 수 있는 지표를 파악하십시오. 이러한 지표는 모든 워크로드 리소스, 애플리케이션 및 서비스를 반영하여 시스템 전체를 한눈에 살펴보고 성능 향상 기회 및 문제를 신속하게 식별할 수 있도록 지원합니다.

원하는 결과: 컴퓨팅 관련 리소스에 대한 모든 지표는 보존 기능이 구현된 단일 플랫폼에서 식별, 수집 및 집계되고 상관 관계가 지정되어 비용 및 운영 목표를 지원합니다.

일반적인 안티 패턴:

- 지표에 대해 수동 로그 파일 검색만 사용합니다.
- 지표를 내부 도구에만 게시합니다.
- 선택한 모니터링 소프트웨어에서 기록한 기본 지표만 사용합니다.
- 문제가 발생한 경우에만 지표를 검토합니다.

이 모범 사례 정립의 이점: 워크로드의 성능을 모니터링하려면 일정 기간에 걸쳐 여러 성능 지표를 기록해야 합니다. 이러한 지표를 바탕으로 성능 이상을 감지할 수 있습니다. 비즈니스 지표를 기준으로 성능을 측정하여 워크로드 요구 사항을 충족하는지 확인할 수도 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

컴퓨팅 관련 지표를 식별, 수집 및 집계하고 상관 관계를 지정하십시오. Amazon CloudWatch와 같은 서비스를 사용하면 구현을 보다 빠르고 편리하게 유지 관리할 수 있습니다. 기록된 기본 지표 외에도 워크로드 내에서 추가 시스템 수준 지표를 식별하고 추적해야 합니다. CPU 활용률, 메모리, 디스크 I/O, 네트워크 인바운드 및 아웃바운드 지표와 같은 데이터를 기록하면 활용률 수준이나 병목 현상을 파악할 수 있습니다. 이와 같은 데이터는 워크로드의 성능과 컴퓨팅 솔루션의 활용 방법을 이해하는 데 매우 중요합니다. 데이터 기반 접근 방식의 일환으로 이 지표를 사용하면 워크로드 리소스를 능동적으로 튜닝하고 최적화할 수 있습니다.

구현 단계:

1. 어떤 컴퓨팅 솔루션 지표를 추적해야 합니까?
    a. EC2 기본 지표
    b. Amazon ECS 기본 지표
    c. EKS 기본 지표
    d. Lambda 기본 지표

   e. EC2 메모리 및 디스크 지표
2. 현재 승인되어 사용 중인 로깅 및 모니터링 솔루션이 있습니까?
   a. Amazon CloudWatch
   b. AWS Distro for OpenTelemetry
   c. Amazon Managed Service for Prometheus
3. 보안 및 운영 목표에 맞게 데이터 보존 정책을 식별하고 구성했습니까?
   a. CloudWatch 지표용 기본 데이터 보존
   b. CloudWatch Logs용 기본 데이터 보존
4. 지표 및 로그 집계 에이전트를 어떻게 배포합니까?
   a. AWS Systems Manager 자동화
   b. OpenTelemetry Collector

구현 계획의 작업 수준: 보통 수준의 노력을 들여 모든 컴퓨팅 리소스에서 지표를 식별, 추적, 수집 및 집계하고 상관 관계를 지정할 수 있습니다.

리소스

관련 문서:

- Amazon CloudWatch 설명서
- CloudWatch 에이전트를 사용하여 Amazon EC2 인스턴스 및 온프레미스 서버에서 지표 및 로그 수집
- AWS Lambda의 Amazon CloudWatch Logs 액세스
- 컨테이너 인스턴스와 CloudWatch Logs 사용
- 사용자 지정 지표 게시
- AWS Answers: 중앙 집중식 로깅
- CloudWatch 지표를 게시하는 AWS 서비스
- AWS Fargate에서의 Amazon EKS 모니터링

관련 동영상:

- AWS의 애플리케이션 성능 관리

- 모니터링 플랜 세우기

관련 예시:

- 레벨 100: CloudWatch 대시보드를 통한 모니터링
- 레벨 100: CloudWatch 대시보드를 통한 Windows EC2 인스턴스 모니터링
- 레벨 100: CloudWatch 대시보드를 통한 Amazon Linux EC2 인스턴스 모니터링

PERF02-BP04 적정 크기 조정으로 필요한 구성 확인

워크로드의 다양한 성능 특성, 그리고 이러한 특성과 메모리, 네트워크, I/O 및 CPU 사용량 간의 관계를 분석합니다. 이 데이터를 사용하면 워크로드 프로필에 가장 적합한 리소스를 선택할 수 있습니다. 예를 들어 데이터베이스와 같은 메모리 집약적 워크로드는 코어당 메모리 비율이 높을수록 유리할 수 있습니다. 하지만 컴퓨팅 집약적인 워크로드에는 더 많은 코어 수와 더 높은 주파수가 필요할 수 있지만 코어당 더 적은 양의 메모리로 만족할 수 있습니다.

일반적인 안티 패턴:

- 모든 워크로드에 사용할 수 있는 모든 성능 특성에서 가장 큰 값을 가진 인스턴스를 선택합니다.
- 관리 용이성을 위해 모든 인스턴스 유형을 한 가지 유형으로 표준화합니다.
- 특정 워크로드의 실제 요구 사항을 검증하지 않고 종합 표준 벤치마크에 대해 최적화합니다.
- 새로운 오퍼링을 재평가하고 통합하지 않고도 오랜 기간 동안 동일한 인프라를 유지합니다.

이 모범 사례 수립의 이점: 워크로드의 요구 사항을 숙지하면 이러한 요구 사항을 사용 가능한 컴퓨팅 오퍼링과 비교하고 신속하게 실험하여 워크로드 요구 사항을 가장 효율적으로 충족하는 오퍼링을 결정할 수 있습니다. 이를 통해 필요하지 않은 리소스 비용을 초과 지불하지 않고도 최적의 성능을 얻을 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

적정 크기 조정으로 워크로드 구성을 수정합니다. 성능, 전반적인 효율성 및 비용 효율성을 최적화하려면 먼저 워크로드에 필요한 리소스를 결정해야 합니다. 데이터베이스와 같은 메모리 집약적 워크로드의 경우 인스턴스의 R-패밀리와 같은 메모리 최적화 인스턴스를 선택합니다. 더 높은 컴퓨팅 용량이 필요한 워크로드의 경우 인스턴스의 C-패밀리를 선택하거나 코어 수가 더 많거나 코어 주파수가 더 높은

인스턴스를 선택합니다. 종합 표준 벤치마크와 비교하는 대신 워크로드의 요구 사항에 따라 I/O 성능을 선택합니다. I/O 성능을 높이려면 인스턴스의 I-패밀리에서 인스턴스를 선택하거나 I/O 최적화 Amazon EBS 볼륨을 선택하거나 인스턴스 스토어가 있는 인스턴스를 선택합니다. 특정 인스턴스 유형에 대한 자세한 내용은 Amazon EC2 인스턴스 유형을 참조하세요.

적정 크기 조정을 통해 불필요한 리소스에 과도한 비용을 지불하지 않으면서 워크로드가 가능한 한 잘 수행되도록합니다.

구현 단계

- 워크로드를 파악하거나 리소스 요구 사항을 분석합니다.
- 워크로드를 별도로 평가합니다. AWS 클라우드 클라우드는 어느 한쪽을 포기할 필요 없이 자체적으로 각 워크로드의 크기를 적절하게 조정할 수 있는 유연성과 민첩성을 제공합니다.
- 테스트 환경을 생성하여 워크로드에 가장 적합한 컴퓨팅 오퍼링을 찾습니다.
- 새로운 컴퓨팅 오퍼링을 지속적으로 재평가하고 워크로드 요구 사항과 비교합니다.
- 더 나은 가격 대비 성능을 위해 새로운 서비스 오퍼링을 정기적으로 검토합니다.
- Well-Architected Framework를 정기적으로 검토합니다.

리소스

관련 모범 사례:

- PERF02-BP03 컴퓨팅 관련 지표 수집
- PERF02-BP06 지표를 기준으로 컴퓨팅 요구 사항의 지속적 평가

관련 문서:

- AWS Compute Optimizer
- AWS 클라우드 컴퓨팅
- Amazon EC2 인스턴스 유형
- Amazon ECS 컨테이너: Amazon ECS 컨테이너 인스턴스
- Amazon EKS 컨테이너: Amazon EKS 워커 노드
- 함수: Lambda 함수 구성

관련 동영상:

- Amazon EC2 foundations (CMP211-R2)(Amazon EC2 기본 사항(CMP211-R2))
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2(더 정확하고, 더 빠르고, 더 저렴한 컴퓨팅: Amazon EC2 비용 최적화)(CMP202-R1)
- Deliver high performance ML inference with AWS Inferentia (CMP324-R1)(AWS Inferentia로 고성능 ML 추론 제공(CMP324-R1))
- Optimize performance and cost for your AWS compute (CMP323-R1)(AWS 컴퓨팅의 성능 및 비용 최적화(CMP323-R1))
- Powering next-gen Amazon EC2: Deep dive into the Nitro system(차세대 Amazon EC2 지원: Nitro 시스템 심층 분석)
- How to choose compute option for startups(스타트업을 위한 컴퓨팅 옵션을 선택하는 방법)
- Optimize performance and cost for your AWS compute (CMP323-R1)(AWS 컴퓨팅의 성능 및 비용 최적화(CMP323-R1))

관련 예시:

- Compute Optimizer 및 메모리 사용률을 활성화하여 적정 크기 조정
- AWS Compute Optimizer데모 코드

PERF02-BP05 사용 가능한 리소스 탄력성 사용

클라우드는 수요 변화에 맞춰 다양한 메커니즘을 통해 리소스를 동적으로 확장 및 축소할 수 있는 유연성을 제공합니다. 이 탄력성과 컴퓨팅 관련 지표를 함께 활용하면 워크로드는 필요한 리소스만 사용하도록 변화에 자동으로 대응할 수 있습니다.

일반적인 안티 패턴:

- 급격한 증가도 처리할 수 있도록 초과 프로비저닝합니다.
- 용량을 수동으로 늘려 경보에 대응합니다.
- 프로비저닝 시간을 고려하지 않고 용량을 늘립니다.
- 조정 이벤트 후에 다시 축소하는 대신 증가된 용량을 그대로 둡니다.
- 워크로드의 실제 요구 사항을 직접 반영하지 않는 지표를 모니터링합니다.

이 모범 사례 확립의 이점: 수요는 고정되거나 가변적일 수 있고 패턴을 따르거나 급증할 수 있습니다. 공급과 수요를 일치시키면 워크로드 비용이 최소화됩니다. 워크로드 탄력성을 모니터링, 테스트 및 구

성하면 성능이 최적화되고 비용이 절감되며 사용량 요구 사항이 변경됨에 따라 안정성이 향상됩니다. 이에 대한 수동 접근이 가능하지만 더 큰 규모에서는 비실용적입니다. 자동화된 지표 기반 접근 방식은 리소스가 언제든지 요구 사항을 충족하도록 보장합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

지표 기반 자동화는 워크로드에 필요한 리소스 수요와 일치하는 리소스 공급을 목표로, 탄력성을 활용하는 데 사용해야 합니다. 예를 들어 Amazon CloudWatch 지표를 사용하여 리소스를 모니터링하거나 Auto Scaling 그룹에 Amazon CloudWatch 지표를 사용할 수 있습니다.

컴퓨팅 관련 지표를 함께 활용하는 경우 워크로드가 변화에 자동으로 대응하여 목표를 달성하는 데 가장 적합한 리소스 세트를 활용할 수 있습니다. 또한 프로비저닝 시간 및 잠재적 리소스 장애에 대한 계획을 세워야 합니다.

인스턴스, 컨테이너 및 함수는 Application Auto Scaling의 형태로 또는 Amazon EC2 Auto Scaling과 함께 서비스의 기능으로 탄력성을 위한 메커니즘을 제공합니다. 아키텍처에서 탄력성을 사용하여 다양한 사용 규모에 대한 성능 요구 사항을 충족하기에 충분한 용량이 있는지 확인합니다.

배포하는 워크로드 유형에 대해 탄력적인 리소스를 스케일 업 또는 스케일 다운할 수 있는 지표를 검증합니다. 예를 들어 동영상 트랜스코딩 애플리케이션을 배포하는 경우 100%의 CPU 활용률이 예상되므로, 기본 지표로 사용해서는 안 됩니다. 또는 대기 중인 트랜스코딩 작업의 대기열 깊이를 기준으로 측정하여 인스턴스 유형을 조정할 수 있습니다.

워크로드 배포에서 스케일 업 및 스케일 다운 이벤트를 모두 처리해야 합니다. 워크로드 구성 요소를 안전하게 축소하는 것은 수요에 따라 리소스를 확장하는 것만큼 중요합니다.

워크로드가 예상대로 작동하는지 확인하기 위해 이벤트 크기 조정을 위한 테스트 시나리오를 만듭니다.

구현 단계

- 기록 데이터를 활용하여 시간별 워크로드의 리소스 수요를 분석합니다. 다음과 같은 구체적인 질문을 합니다.
  - 워크로드가 시간이 지남에 따라 일정하게 알려진 속도로 증가하고 있습니까?
  - 계절에 따라 반복 가능한 패턴으로 워크로드가 증가하거나 감소합니까?
  - 워크로드가 급증합니까? 급증을 예상하거나 예측할 수 있습니까?
- 모니터링 서비스와 기록 데이터를 최대한 활용합니다.

- 리소스에 태그를 지정하면 모니터링에 도움이 될 수 있습니다. 태그를 사용할 때 태그 지정 모범 사례를 참조합니다. 태그는 리소스를 관리, 식별 및 구성할 때도 도움이 될 수 있습니다.
- AWS에서는 다양한 방식을 사용하여 수요와 공급을 일치시킬 수 있습니다. 비용 최적화 원칙 모범 사례(COST09-BP01 ~ COST09-03)는 다음과 같은 비용 접근 방식을 사용하는 방법을 설명합니다.
  - COST09-BP01 워크로드 수요 분석 수행
  - COST09-BP02 수요 관리를 위한 버퍼 또는 스로틀 구현
  - COST09-BP03 동적으로 리소스 공급
- 스케일 다운 이벤트에 대한 테스트 시나리오를 생성하여 워크로드가 예상대로 작동하는지 확인합니다.
- 대부분의 비 프로덕션 인스턴스는 사용되지 않을 때 중지되어야 합니다.
- Amazon Elastic Block Store(Amazon EBS)를 사용할 때 필요한 스토리지의 경우 볼륨 기반 탄력성을 활용합니다.
- Amazon Elastic Compute Cloud(Amazon EC2)의 경우 수요 급증 시 컴퓨팅 인스턴스 수를 자동으로 늘리고 수요 감소 시 용량을 줄여 성능과 비용을 최적화할 수 있는 Auto Scaling 그룹을 사용하는 것이 좋습니다.

리소스

관련 모범 사례:

- PERF02-BP03 컴퓨팅 관련 지표 수집
- PERF02-BP04 적정 크기 조정으로 필요한 구성 확인
- PERF02-BP06 지표를 기준으로 컴퓨팅 요구 사항의 지속적 평가

관련 문서:

- AWS 클라우드 컴퓨팅
- Amazon EC2 인스턴스 유형
- Amazon ECS 컨테이너: Amazon ECS 컨테이너 인스턴스
- Amazon EKS 컨테이너: Amazon EKS 워커 노드
- 함수: Lambda 함수 구성

관련 동영상:

- Amazon EC2 foundations (CMP211-R2)(Amazon EC2 기본 사항(CMP211-R2))
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2(더 정확하고, 더 빠르고, 더 저렴한 컴퓨팅: Amazon EC2 비용 최적화)(CMP202-R1)
- Deliver high performance ML inference with AWS Inferentia (CMP324-R1)(AWS Inferentia로 고성능 ML 추론 제공(CMP324-R1))
- Optimize performance and cost for your AWS compute (CMP323-R1)(AWS 컴퓨팅의 성능 및 비용 최적화(CMP323-R1))
- Powering next-gen Amazon EC2: Deep dive into the Nitro system(차세대 Amazon EC2 지원: Nitro 시스템 심층 분석)

관련 예시:

- Amazon EC2 Auto Scaling 그룹 예
- Amazon EFS 자습서

PERF02-BP06 지표를 기준으로 컴퓨팅 요구 사항의 지속적 평가

데이터 기반 접근 방식을 사용하여 시간이 지남에 따라 워크로드에 적합한 컴퓨팅 리소스를 지속적으로 평가하고 최적화합니다.

원하는 결과: 시스템 수준 지표를 사용하여 시간별 워크로드 동작 및 요구 사항을 적극적으로 모니터링합니다. 수집된 데이터를 기반으로 사용 가능한 리소스에 대해 워크로드의 요구 사항을 평가하고 워크로드 프로필에 가장 적합하도록 컴퓨팅 환경을 변경합니다 예를 들어 워크로드는 계속 사용하다 보면 초기에 지정된 것보다 메모리가 더 많이 사용될 수 있습니다. 이 경우 다른 인스턴스 패밀리나 크기로 전환하면 성능과 효율성이 모두 개선될 수 있습니다.

일반적인 안티 패턴:

- 컴퓨팅 요구 사항을 재평가하지 않고 워크로드에 대한 인사이트를 얻기 위해 시스템 수준 지표를 모니터링합니다.
- 피크 워크로드 요구 사항에 따라 컴퓨팅 요구 사항을 설계합니다.
- 워크로드 특성에 보다 효율적으로 맞는 대체 컴퓨팅 솔루션으로 이동할 때 확장 또는 성능 요구 사항을 충족하도록 기존 컴퓨팅 솔루션의 규모를 과도하게 키웁니다.

이 모범 사례 수립의 이점: 실제 데이터 및 원하는 비용과 성능의 균형을 기반으로 컴퓨팅 리소스를 최적화합니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

데이터 기반 접근 방식을 사용하여 관찰된 워크로드 동작을 기반으로 컴퓨팅 리소스를 최적화합니다. 성능 및 효율성을 극대화하려면 워크로드에서 시간대별로 수집된 데이터를 사용하여 리소스를 지속적으로 튜닝하고 최적화합니다. 워크로드의 현재 리소스 사용량 추세를 파악하고, 워크로드의 요구에 더 적합하게 리소스 사용 방식을 변경할 수 있는 영역을 확인합니다. 리소스가 너무 많이 커밋되면 시스템 성능이 저하되고 리소스가 적절하게 사용되지 않으면 시스템 사용 효율성이 낮아지고 비용이 높아집니다.

성능 및 리소스 사용률을 최적화하려면 통합된 운영 보기, 세분화된 실시간 데이터 및 기간별 참조가 필요합니다. 자동화된 대시보드를 생성하여 이 데이터를 시각화하고 운영 및 사용률에 관한 인사이트를 얻을 수 있습니다.

구현 단계

1. 시간 경과에 따른 컴퓨팅 관련 지표를 수집합니다.
2. 선택한 컴퓨팅 솔루션에서 사용 가능한 리소스와 워크로드 지표를 비교합니다.
3. 기존 솔루션의 크기를 조정하거나 대체 컴퓨팅 솔루션을 평가하여 필요한 구성 변경을 결정합니다.

리소스

관련 모범 사례:

- PERF02-BP01 사용 가능한 컴퓨팅 옵션 평가
- PERF02-BP02 사용 가능한 컴퓨팅 구성 옵션 파악
- PERF02-BP03 컴퓨팅 관련 지표 수집
- PERF02-BP04 적정 크기 조정으로 필요한 구성 확인

관련 문서:

- AWS 클라우드 컴퓨팅
- AWS Compute Optimizer
- EC2 인스턴스 유형
- Amazon ECS 컨테이너: Amazon ECS 컨테이너 인스턴스
- Amazon EKS 컨테이너: Amazon EKS 워커 노드

- AWS Lambda 함수 작업의 모범 사례

관련 동영상:

- Amazon EC2 foundations (CMP211-R2)(Amazon EC2 기본 사항(CMP211-R2))
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2(더 정확하고, 더 빠르고, 더 저렴한 컴퓨팅: Amazon EC2 비용 최적화)(CMP202-R1)
- Deliver high performance ML inference with AWS Inferentia (CMP324-R1)(AWS Inferentia로 고성능 ML 추론 제공(CMP324-R1))
- Optimize performance and cost for your AWS compute (CMP323-R1)(AWS 컴퓨팅의 성능 및 비용 최적화(CMP323-R1))
- Powering next-gen Amazon EC2: Deep dive into the Nitro system(차세대 Amazon EC2 지원: Nitro 시스템 심층 분석)
- Selecting and optimizing Amazon EC2 instances(Amazon EC2 인스턴스 선택 및 최적화)

관련 예시:

- Compute Optimizer 및 메모리 사용률을 활성화하여 적정 크기 조정
- AWS Compute Optimizer데모 코드

PERF 3 스토리지 솔루션을 어떻게 선택합니까?

시스템에 대한 최적의 스토리지 솔루션은 액세스 방식 종류(블록, 파일, 객체), 액세스 패턴(랜덤 또는 순차), 필요한 처리량, 액세스 빈도(온라인, 오프라인, 보관), 업데이트 빈도(WORM, 동적) 및 가용성과 내구성 제약 사항에 따라 다릅니다. 설계가 잘된 시스템은 여러 스토리지 솔루션을 사용하며, 다양한 기능을 통해 성능을 개선하고 리소스를 효율적으로 사용할 수 있도록 지원합니다.

모범 사례

- PERF03-BP01 스토리지 특성 및 요구 사항 파악
- PERF03-BP02 사용 가능한 구성 옵션 평가
- PERF03-BP03 액세스 패턴과 지표를 기준으로 결정

PERF03-BP01 스토리지 특성 및 요구 사항 파악

워크로드 스토리지의 요구 사항 파악하여 문서화하고 각 위치의 스토리지 특성을 정의합니다. 스토리지 특성 예시에는 공유 가능한 액세스, 파일 크기, 성장률, IOPS, 지연 시간, 액세스 패턴 및 데이터 지속성 등이 있습니다. 이러한 특성을 사용하여 스토리지의 요구 사항에 대해 블록, 파일, 객체 또는 인스턴스 스토리지 서비스가 가장 효율적인 솔루션인지 평가합니다.

원하는 결과: 스토리지 요구 사항을 파악하여 스토리지의 요구 사항별로 문서화하고 사용 가능한 스토리지 솔루션을 평가합니다. 주요 스토리지 특성을 바탕으로 팀에서는 선택한 스토리지 서비스가 워크로드 성능에 어떻게 도움이 될지 이해합니다. 주요 기준에는 데이터 액세스 패턴, 성장률, 스케일링 요구 사항 및 지연 시간 요구 사항이 포함됩니다.

일반적인 안티 패턴:

- 모든 워크로드에 Amazon Elastic Block Store(Amazon EBS)와 같은 하나의 스토리지 유형만 사용합니다.
- 모든 워크로드의 스토리지 액세스 성능 요구 사항이 비슷하다고 가정합니다.

이 모범 사례 확립의 이점: 파악한 특성과 필요한 특성을 바탕으로 스토리지 솔루션을 선택하면 워크로드 성능을 개선하고 비용을 절감하며 워크로드 유지 관리를 위한 운영 작업을 줄일 수 있습니다. 워크로드 성능은 스토리지 서비스의 솔루션, 구성 및 위치에 따라 향상됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

벤치마킹 또는 로드 테스트를 사용하여 워크로드의 가장 중요한 스토리지 성능 지표를 식별하고 데이터 기반 접근 방식의 일부로 개선 사항을 구현합니다. 이 데이터를 사용하여 스토리지 솔루션이 제한되는 위치를 식별하고, 솔루션을 개선할 수 있는 구성 옵션을 검사합니다. 워크로드의 예상 증가율을 확인하고 해당 속도에 맞는 스토리지 솔루션을 선택합니다. AWS 스토리지 서비스를 조사하여 다양한 워크로드에 적합한 스토리지 솔루션을 결정하세요. AWS의 프로비저닝 스토리지 솔루션은 스토리지 서비스를 테스트하고 이것이 워크로드의 요구 사항에 적합한지 확인할 수 있는 기회를 늘립니다.

AWS 서비스	주요 특징	일반 사용 사례
Amazon S3	99.999999999% 내구성, 무제한 성장, 어디에서나 액세스 가능, 액세스 및 복원력에 기반한 여러 가지 비용 모델	클라우드 네이티브 애플리케이션 데이터, 데이터 아카이빙 및 백업, 분석, 데이터 레이크, 정

AWS 서비스	주요 특징	일반 사용 사례
		적 웹 사이트 호스팅, IoT 데이터
Amazon S3 Glacier	몇 초에서 몇 시간에 이르는 지연 시간, 무제한 성장, 최저 가격, 장기 스토리지	데이터 아카이빙, 미디어 아카이브, 장기 백업 보존
Amazon EBS	스토리지 크기에 관리 및 모니터링, 낮은 지연 시간, 지속적인 스토리지, 99.8%~99.9% 내구성이 필요하며, 대부분의 볼륨 유형은 하나의 EC2 인스턴스에서만 액세스 가능	COTS 애플리케이션, I/O 집약적 애플리케이션, 관계형 및 NoSQL 데이터베이스, 백업 및 복구
EC2 인스턴스 스토어	미리 정의된 스토리지 크기, 가장 짧은 지연 시간, 지속성 없음, EC2 인스턴스에서만 액세스 가능	COTS 애플리케이션, I/O 집약적 애플리케이션, 인 메모리 데이터 저장소
Amazon EFS	99.999999999% 내구성, 무제한 성장, 여러 컴퓨팅 서비스에서 액세스 가능	여러 컴퓨팅 서비스 간에 현대화된 애플리케이션 파일 공유, 콘텐츠 관리 시스템 스케일링을 위한 파일 스토리지
Amazon FSx	네 가지 파일 시스템(NetApp, OpenZFS, Windows File Server 및 Amazon FSx for Lustre) 지원, 여러 파일별 시스템에 사용할 수 있는 스토리지, 여러 컴퓨팅 서비스에서 액세스 가능	클라우드 네이티브 워크로드, 프라이빗 클라우드 버스팅, 특정 파일 시스템이 필요한 마이그레이션된 워크로드, VMC, ERP 시스템, 온프레미스 파일 스토리지 및 백업
Snow 제품군	휴대용 디바이스, 256비트 암호화, NFS 엔드포인트, 온보드 컴퓨팅, TB 스토리지	클라우드로 데이터 마이그레이션, 스토리지, 극한의 온프레미스 조건에서 컴퓨팅, 재해 복구, 원격 데이터 수집

AWS 서비스	주요 특징	일반 사용 사례
AWS Storage Gateway	클라우드 지원 스토리지에 대해 지연 시간이 짧은 온프레미스 액세스 제공, 완전 관리형 온프레미스 캐시	클라우드 마이그레이션에 대한 온프레미스 데이터, 온프레미스 소스에서 클라우드 데이터 레이크 채우기, 현대화된 파일 공유

구현 단계:

1. 벤치 마킹 또는 로드 테스트를 사용하여 스토리지 필요의 주요 특성을 수집합니다. 주요 특징은 다음과 같습니다.
   a. 공유 가능(구성 요소가 이 스토리지에 액세스할 수 있음)
   b. 성장률
   c. 처리량
   d. 지연 시간
   e. I/O 크기
   f. 내구성
   g. 액세스 패턴(일기와 쓰기, 빈도, 급증하는지 또는 일관적인지)
2. 스토리지 특성을 지원하는 스토리지 솔루션 유형을 파악합니다.
   a. Amazon S3 은(는) 무제한 확장성, 높은 가용성과 액세스 가능성을 위한 여러 옵션을 갖춘 객체 스토리지 서비스입니다. Amazon S3 안팎에서 객체 전송 및 액세스는 Transfer Acceleration 또는 액세스 포인트 등과 같은 서비스를 사용하여 위치, 보안 필요 및 액세스 패턴을 지원할 수 있습니다. Amazon S3의 성능 지침 을 사용하여 워크로드 성능의 요구 사항을 충족하도록 Amazon S3 구성을 최적화할 수 있습니다.
   b. Amazon S3 Glacier 은(는) 데이터 아카이빙을 위해 구축된 Amazon S3의 스토리지 클래스입니다. 밀리초에서 5~12시간까지 액세스 시간이 다양하고 비용 및 보안 옵션이 제각각인 세 가지 아카이빙 솔루션 중에서 선택할 수 있습니다. Amazon S3 Glacier은(는) 비즈니스 요구 사항 및 데이터 특성을 지원하는 데이터 수명 주기를 구현하여 성능 요구 사항을 충족하도록 돕습니다.
   c. Amazon Elastic Block Store(Amazon EBS) 은(는) Amazon Elastic Compute Cloud(Amazon EC2)을(를) 위해 설계된 고성능 블록 스토리지 서비스입니다. IOPS 또는 처리량에 대한 우선 순위를 지정하는 특성이 다른 SSD 기반 또는 HDD 기반 솔루션 중에서 선택할 수 있습니다. EBS 볼륨은 고성능 워크로드, 파일 시스템을 위한 기본 스토리지, 데이터베이스 또는 연결된 스테이지 시스템에만 액세스할 수 있는 애플리케이션에 적합합니다.

d. Amazon EC2 인스턴스 스토어 는 Amazon EC2 인스턴스에 연결되므로 Amazon EBS과(와) 유사하지만 이상적으로 버퍼, 캐시 또는 기타 임시 콘텐츠로 사용해야 하는 임시 스토리지에 불과합니다. 인스턴스 스토어는 분리할 수 없으며 인스턴스가 종료되면 모든 데이터가 손실됩니다. 인스턴스 스토어는 데이터를 지속할 필요가 없는 높은 I/O 성능 및 짧은 지연 시간 사용 사례에 사용할 수 있습니다.

e. Amazon Elastic File System(Amazon EFS) 은(는) 여러 유형의 컴퓨팅 솔루션에서 액세스할 수 있는 탑재 가능한 파일 시스템입니다. Amazon EFS은(는) 스토리지를 자동으로 늘리고 줄이며 대기 시간을 일정하게 짧게 유지하기 위해 성능이 최적화됩니다. EFS에는 두 가지 성능 구성 모드인 범용 모드와 최대 I/O 모드가 있습니다. 범용은 읽기 지연 시간이 1밀리초 미만이며, 쓰기 지연 시간은 한 자릿수 밀리초입니다. 최대 I/O 기능은 공유 파일 시스템이 필요한 수천 개의 컴퓨팅 인스턴스를 지원할 수 있습니다. Amazon EFS은(는) 두 가지 처리량 모드즉, 버스팅과 프로비저닝을 지원합니다. 갑작스러운 액세스 급증 패턴이 나타나는 워크로드는 버스팅 처리량 모드에서 이점을 얻을 수 있는 동시에 일정하게 액세스가 높은 워크로드는 프로비저닝 처리량 모드에서 잘 작동할 수 있습니다.

f. Amazon FSx 은(는) 일반적으로 사용되는 네 가지 파일 시스템인 NetApp ONTAP, OpenZFS, Windows File Server 및 Lustre를 지원하는 최신 AWS 컴퓨팅 솔루션을 기반으로 구축되었습니다. Amazon FSx 지연 시간, 처리량 및 IOPS 는 파일 시스템에 따라 달라지며 워크로드의 요구 사항에 적합한 파일 시스템을 선택할 때 고려해야 합니다.

g. AWS Snow Family 은(는) 클라우드과 데이터 스토리지로 온라인 및 오프라인 데이터 마이그레이션과 온프레미스 컴퓨팅을 지원하는 스토리지 및 컴퓨팅 디바이스입니다. AWS Snow 디바이스는 많은 양의 온프레미스 데이터 수집, 해당 데이터의 처리 및 클라우드로 해당 데이터 이동을 지원합니다. 파일 수, 파일 크기 및 압축과 관련하여 여러 가지 문서화된 성능 모범 사례 가 있습니다.

h. AWS Storage Gateway 은(는) 클라우드 기반 스토리지에 대한 온프레미스 애플리케이션 액세스를 제공합니다. AWS Storage Gateway는 Amazon S3, Amazon S3 Glacier, Amazon FSx 및 Amazon EBS 등과 같은 여러 클라우드 스토리지 서비스를 지원하고 iSCSI, SMB 및 NFS 등과 같은 여러 프로토콜을 지원합니다. 자주 액세스하는 데이터를 온프레미스에 캐싱하여 지연 시간이 짧은 성능을 제공하고 변경된 데이터와 압축된 데이터만 AWS(으)로 전송합니다.

3. 새로운 스토리지 솔루션을 실험하고 최적의 구성을 파악한 후 마이그레이션을 계획하고 성능 지표를 검증합니다. 이는 지속적인 프로세스이며 주요 특성이 변경되거나 사용 가능한 서비스 또는 옵션이 변경되면 다시 평가해야 합니다.

구현 계획의 작업 수준: 워크로드가 한 스토리지 솔루션에서 다른 컴퓨팅 솔루션으로 이동하는 경우 애플리케이션을 리팩터링하는 데 중간 수준의 작업이 필요할 수 있습니다.

리소스

관련 문서:

- Amazon EBS 볼륨 유형
- Amazon EC2 스토리지
- Amazon EFS: Amazon EFS 성능
- Amazon FSx for Lustre 성능
- Amazon FSx for Windows File Server 성능
- Amazon FSx for NetApp ONTAP 성능
- Amazon FSx for OpenZFS 성능
- Amazon S3 Glacier: Amazon S3 Glacier 설명서
- Amazon S3: 요청 속도 및 성능 고려 사항
- AWS의 클라우드 스토리지
- AWS Snow Family
- EBS I/O 특성

관련 동영상:

- Deep dive on Amazon EBS(STG303-R1)
- Optimize your storage performance with Amazon S3(STG343)

관련 예시:

- Amazon EFS CSI 드라이버
- Amazon EBS CSI 드라이버
- Amazon EFS 유틸리티
- Amazon EBS 오토 스케일링
- Amazon S3 예시
- Amazon FSx for Lustre Container Storage Interface(CSI) 드라이버

PERF03-BP02 사용 가능한 구성 옵션 평가

다양한 특성 및 구성 옵션과 스토리지와의 관련성을 평가합니다. 프로비저닝된 IOPS, SSD, 마그네틱 스토리지, 객체 스토리지, 아카이브 스토리지 또는 휘발성 스토리지를 사용하는 위치와 방법을 파악하여 워크로드의 성능과 스토리지 공간을 최적화합니다.

Amazon EBS 는 워크로드에 맞게 스토리지 성능과 비용을 최적화할 수 있는 광범위한 옵션을 제공합니다. 이러한 옵션은 두 가지 주요 범주로 구분됩니다. 그중 하나는 데이터베이스 및 부트 볼륨과 같은 트랜잭션 워크로드용 SSD-지원 스토리지이고(주로 IOPS에 따라 성능이 달라짐), 다른 하나는 MapReduce 및 로그 처리와 같은 처리량이 높은 워크로드용 HDD 지원 스토리지입니다(주로 초당 MB에 따라 성능이 달라짐).

SSD 지원 볼륨에는 지연 시간에 따라 달라지는 트랜잭션 워크로드용으로 최상의 성능을 제공하는 프로비저닝된 IOPS SSD와 다양한 트랜잭션 데이터용으로 사용 가능하도록 가격과 성능이 적절하게 절충된 범용 SSD가 포함됩니다.

Amazon S3 Transfer Acceleration 을 사용하면 클라이언트와 S3 버킷 간의 장거리 파일 전송을 빠르게 수행할 수 있습니다. Transfer Acceleration은 전 세계에 분산된 Amazon CloudFront의 엣지 로케이션을 활용하여 최적화된 네트워크 경로를 통해 데이터를 라우팅합니다. GET 요청을 많이 수행하는 S3 버킷의 워크로드에는 CloudFront가 포함된 Amazon S3를 사용합니다. 큰 파일을 업로드할 때는 여러 부분이 동시에 업로드되는 멀티 파트 업로드를 사용하면 네트워크 처리량을 극대화할 수 있습니다.

Amazon Elastic File System(Amazon EFS) 은 AWS 클라우드 서비스 및 온프레미스 리소스에 사용할 수 있는 간편하고 확장 가능하며 탄력적인 완전관리형 NFS 파일 시스템을 제공합니다. Amazon EFS는 범용 성능 모드와 최대 I/O 성능 모드라는 두 성능 모드를 제공하여 다양한 클라우드 스토리지 워크로드를 지원합니다. 또한 버스팅 처리량(throughput) 및 프로비저닝된 처리량(throughput)이라는 두 처리량(throughput) 모드 중에서 파일 시스템에 적합한 모드를 선택할 수 있습니다. 워크로드에 사용할 설정을 결정하려면 Amazon EFS 사용 설명서를 참조하십시오.

Amazon FSx 는 엔터프라이즈 워크로드용 Amazon FSx for Windows File Server, 고성능 워크로드용 Amazon FSx for Lustre, NetApps 주요 ONTAP 파일 시스템용 Amazon FSx for NetApp ONTAP, Linux 기반 파일 서버용 Amazon FSx for OpenZFS 등 4가지 파일 시스템 중에서 선택할 수 있는 옵션을 제공합니다. FSx는 SSD를 지원하며, 빠르고 예측 가능하며 확장 가능하고 일관된 성능을 제공하도록 설계되었습니다. Amazon FSx 파일 시스템은 지속적으로 높은 읽기 및 쓰기 속도와 일관되게 지연 시간이 짧은 데이터 액세스를 제공합니다. 워크로드의 요구 사항에 따라 필요한 처리량 수준을 선택할 수 있습니다.

일반적인 안티 패턴:

- 모든 워크로드에 Amazon EBS와 같은 하나의 스토리지 유형만 사용합니다.

- 모든 스토리지 계층을 기준으로 한 실제 테스트 없이 워크로드 전체에 프로비저닝된 IOPS를 사용합니다.
- 모든 워크로드의 스토리지 액세스 성능 요구 사항이 비슷하다고 가정합니다.

이 모범 사례 수립의 이점: 모든 스토리지 서비스 옵션을 평가하면 인프라 비용과 워크로드를 유지 관리하는 데 필요한 노력을 줄일 수 있습니다. 새로운 서비스와 기능의 배포에 대한 출시 시간을 가속화할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

스토리지 특성 결정: 스토리지 솔루션을 평가할 때는 공유 기능, 파일 크기, 캐시 크기, 지연 시간, 처리량(throughput) 및 데이터 지속성 등 필요한 스토리지 특성이 무엇인지 결정합니다. 그런 다음 요구 사항에 가장 적합한 AWS 서비스에 요구 사항을 맞춥니다.

리소스

관련 문서:

- AWS의 클라우드 스토리지
- Amazon EBS 볼륨 유형
- Amazon EC2 스토리지
- Amazon EFS: Amazon EFS 성능
- Amazon FSx for Lustre 성능
- Amazon FSx for Windows File Server 성능
- Amazon Glacier: Amazon Glacier 설명서
- Amazon S3: 요청 속도 및 성능 고려 사항
- AWS의 클라우드 스토리지
- AWS의 클라우드 스토리지
- EBS I/O 특성

관련 동영상:

- Deep dive on Amazon EBS(STG303-R1)
- Optimize your storage performance with Amazon S3(STG343)

관련 예시:

- Amazon EFS CSI 드라이버
- Amazon EBS CSI 드라이버
- Amazon EFS 유틸리티
- Amazon EBS 자동 확장
- Amazon S3 예시

PERF03-BP03 액세스 패턴과 지표를 기준으로 결정

워크로드의 액세스 패턴을 기준으로 스토리지 시스템을 선택하고, 워크로드에서 데이터에 액세스하는 방법을 결정하여 이러한 스토리지 시스템을 구성합니다. 블록 스토리지 대신 객체 스토리지를 선택하여 스토리지 효율성을 높이십시오. 선택한 스토리지 옵션을 데이터 접근 패턴과 일치하도록 구성합니다.

데이터에 액세스하는 방식은 스토리지 솔루션의 성능에 영향을 줍니다. 따라서 성능을 극대화하려면 접근 패턴에 가장 적합한 스토리지 솔루션을 선택하거나, 스토리지 솔루션에 따라 접근 패턴을 변경하는 것이 좋습니다.

RAID 0 어레이를 생성하면 단일 볼륨에서 프로비저닝할 때보다 파일 시스템의 성능 수준을 더 높일 수 있습니다. 내결함성보다 I/O 성능이 더 중요할 때는 RAID 0를 사용하는 것이 좋습니다. 예를 들어 데이터 복제가 이미 별도로 설정되어 있으며 사용 빈도가 매우 높은 데이터베이스의 경우에는 RAID 0를 사용해야 합니다.

워크로드에 사용된 모든 스토리지 옵션에서 워크로드에 적절한 스토리지 지표를 선택합니다. 버스트 크레딧을 이용하는 파일 시스템을 사용하는 경우에는 이러한 크레딧 제한에 근접할 때 알림을 제공하는 경보를 생성합니다. 전체 워크로드 스토리지 상태를 보여주는 스토리지 대시보드를 생성해야 합니다.

Amazon EBS 또는 Amazon FSx와 같이 크기가 고정된 스토리지 시스템의 경우 전체 스토리지 크기 대비 사용된 스토리지의 양을 모니터링해야 하며, 임계값에 도달할 때 스토리지 크기를 늘릴 수 있는 자동화 기능을 생성해야 합니다.

일반적인 안티 패턴:

- 고객이 불만을 제기하지 않으면 스토리지 성능이 적절한 것이라고 가정합니다.
- 모든 워크로드가 해당 계층 내에서 적합하다고 가정하고 하나의 스토리지 계층만 사용합니다.

이 모범 사례 수립의 이점: 성능 및 리소스 사용률을 최적화하려면 통합된 운영 보기, 세분화된 실시간 데이터 및 기간별 참조가 필요합니다. 1초의 세분성을 포함하는 자동 대시보드와 데이터를 생성하여 데이터에 대한 지표 산술을 수행하고 스토리지 요구 사항에 대한 운영 및 사용률 인사이트를 도출할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

스토리지 사용량 및 액세스 패턴 최적화: 워크로드 액세스 패턴 및 사용 가능한 스토리지 옵션의 특성에 따라 스토리지 시스템을 선택합니다. 오버헤드를 줄이면서 요구 사항을 충족할 수 있는 최적의 데이터 저장 위치를 결정합니다. 스토리지의 특성에 따라 데이터를 구성하고 데이터와 상호 작용할 때는 성능 최적화 및 액세스 패턴을 사용합니다(예: 볼륨 스트라이핑 또는 데이터 분할).

스토리지 옵션에 적합한 지표 선택: 워크로드에 적합한 스토리지 지표를 선택해야 합니다. 각 스토리지 옵션은 워크로드의 시간대별 성능을 추적할 수 있는 다양한 지표를 제공합니다. 스토리지 버스트 지표를 기준으로 측정해야 합니다(예: Amazon EFS의 버스트 크레딧 모니터링). Amazon Elastic Block Store 또는 Amazon FSx와 같이 크기가 고정된 스토리지 시스템의 경우 사용된 스토리지의 양과 전체 스토리지 크기를 비교하여 모니터링해야 합니다. 임계값에 도달하면 가능한 경우 스토리지 크기를 늘리는 자동화를 생성합니다.

지표 모니터링: Amazon CloudWatch는 아키텍처의 리소스 전반에서 지표를 수집할 수 있습니다. 또한 사용자 지정 지표를 수집하고 게시하여 비즈니스 또는 파생 지표를 파악할 수도 있습니다. CloudWatch 또는 타사 솔루션을 사용하여 임계값 위반 시점을 나타내는 경보를 설정합니다.

리소스

관련 문서:

- Amazon EBS 볼륨 유형
- Amazon EC2 스토리지
- Amazon EFS: Amazon EFS 성능
- Amazon FSx for Lustre 성능
- Amazon FSx for Windows File Server 성능
- Amazon Glacier: Amazon Glacier 설명서
- Amazon S3: 요청 속도 및 성능 고려 사항
- AWS의 클라우드 스토리지
- EBS I/O 특성

- Amazon CloudWatch를 사용한 Amazon EBS 성능 모니터링 및 이해

관련 동영상:

- Deep dive on Amazon EBS(STG303-R1)
- Optimize your storage performance with Amazon S3(STG343)

관련 예시:

- Amazon EFS CSI 드라이버
- Amazon EBS CSI 드라이버
- Amazon EFS 유틸리티
- Amazon EBS 자동 확장
- Amazon S3 예시

PERF 4 데이터베이스 솔루션을 어떻게 선택합니까?

시스템에 대한 최적의 데이터베이스 솔루션은 가용성, 일관성, 파티션 허용 오차, 지연 시간, 내구성, 확장성, 쿼리 기능에 대한 요구 사항에 따라 다릅니다. 여러 시스템은 다양한 하위 시스템에 서로 다른 데이터베이스 솔루션을 사용하고 다양한 기능을 활성화하여 성능을 개선할 수 있습니다. 시스템에 대해 잘못된 데이터베이스 솔루션 및 기능을 선택하면 성능 효율성이 저하될 수 있습니다.

모범 사례

- PERF04-BP01 데이터 특성 파악
- PERF04-BP02 사용 가능한 옵션 평가
- PERF04-BP03 데이터베이스 성능 지표 수집 및 기록
- PERF04-BP04 액세스 패턴을 기준으로 데이터 스토리지 선택
- PERF04-BP05 액세스 패턴 및 지표를 기준으로 데이터 스토리지 최적화

PERF04-BP01 데이터 특성 파악

워크로드 데이터 세트의 특성, 액세스 패턴 및 요구 사항에 최적으로 일치하는 데이터 관리 솔루션을 선택합니다. 데이터 관리 솔루션을 선택하고 구현할 때는 쿼리, 확장 및 스토리지 특성이 워크로드 데이터 요구 사항을 지원하는지 확인해야 합니다. 다양한 데이터베이스 옵션이 데이터 모델과 어떻게 일치하는지, 어떤 구성 옵션이 사용 사례에 가장 적합한지 알아보십시오.

AWS에서는 관계형, 키 값, 문서, 인 메모리, 그래프, 시계열 및 원장 데이터베이스를 포함한 수많은 데이터베이스 엔진을 제공합니다. 각 데이터 관리 솔루션에는 사용 사례 및 데이터 모델을 지원하는 옵션과 구성이 있습니다. 워크로드에서 데이터 특성에 따라 여러 가지 데이터베이스 솔루션을 사용할 수 있습니다. 특정 문제에 가장 적합한 데이터베이스 솔루션을 선택하면, 모든 상황에 한 가지 접근 방식을 사용하는 제한적인 모놀리식 데이터베이스에서 탈피하여 고객의 요구를 충족하는 데이터 관리에 집중할 수 있습니다.

원하는 결과: 지원 데이터베이스 솔루션의 선택과 구성을 용이하게 하고 잠재적인 대안에 대한 인사이트를 제공하도록 충분한 세부 정보를 담아 워크로드 데이터 특성을 문서화합니다.

일반적인 안티 패턴:

- 대규모 데이터 세트를 유사한 특성을 가진 더 작은 데이터 모음으로 분할하는 방법을 고려하지 않아 데이터 및 성장 특성에 보다 적합한 목적별 데이터베이스를 더 많이 사용할 기회를 놓칩니다.
- 데이터 액세스 패턴을 사전에 식별하지 않아 나중에 많은 비용이 들고 복잡한 재작업을 해야 합니다.
- 필요에 따라 신속하게 확장되지 않는 데이터 스토리지 전략을 사용하여 성장을 제한합니다.
- 모든 워크로드에 하나의 데이터베이스 유형과 공급업체를 선택합니다.
- 특정 데이터베이스 솔루션 유형에 대한 내부 경험과 지식이 있어 하나의 데이터베이스 솔루션만 고수합니다.
- 온프레미스 환경에서 잘 작동했다는 이유로 같은 데이터베이스 솔루션을 그대로 사용합니다.

이 모범 사례 수립의 이점: 다양한 워크로드에 적합한 데이터베이스 솔루션을 결정하려면 모든 AWS 데이터베이스 솔루션에 대해 잘 알아야 합니다. 워크로드에 사용할 데이터베이스 솔루션을 선택한 후에는 이러한 각 데이터베이스 오퍼링을 신속하게 실험하여 워크로드 요구 사항이 계속해서 충족되는지 확인할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

- 잠재적인 비용 절감 효과가 나타나지 않을 수 있습니다.
- 데이터가 필요한 수준으로 보호되지 않을 수 있습니다.
- 데이터 액세스 및 스토리지 성능이 최적의 상태가 아닐 수 있습니다.

구현 가이드

워크로드의 데이터 특성과 액세스 패턴을 정의하십시오. 사용 가능한 모든 데이터베이스 솔루션을 검토하여 데이터 요건을 지원하는 솔루션을 식별해야 합니다. 주어진 워크로드 내에서 다양한 데이터베이스를 선택할 수 있습니다. 각 서비스 또는 서비스 그룹을 평가하고 개별적으로 검토하십시오. 데이터

의 일부 또는 전체에 대해 사용할 수 있는 대체 데이터 관리 솔루션이 확인된 경우 비용, 보안, 성능, 신뢰성 측면에서 이점을 제공할 수 있는 대체 구현을 시험해 보면 좋습니다. 새로운 데이터 관리 전략을 채택할 경우 기존 문서를 업데이트합니다.

유형	AWS 서비스	주요 특징	일반 사용 사례
관계형	Amazon RDS, Amazon Aurora	참조 무결성, ACID 트랜잭션, 쓰기 스키마	ERP, CRM, 상용 소프트웨어
키-값	Amazon DynamoDB	높은 처리량(throughput), 짧은 지연 시간, 무한에 가까운 확장성	장바구니(전자 상거래), 제품 카탈로그, 채팅 애플리케이션
문서	Amazon DocumentDB	JSON 문서를 저장하고 모든 속성 쿼리	콘텐츠 관리(CMS), 고객 프로파일, 모바일 애플리케이션
인 메모리	Amazon ElastiCache, Amazon MemoryDB	마이크로초 단위의 지연 시간	캐싱, 게임 리더보드
그래프	Amazon Neptune	데이터 간의 관계가 의미를 갖는 고도의 관계형 데이터	소셜 네트워크, 개인화 엔진, 사기 탐지
시계열	Amazon Timestream	기본 차원이 시간인 데이터	DevOps, IoT, 모니터링
와이드 컬럼	Amazon Keyspaces	Cassandra 워크로드	산업 장비 유지보수, 경로 최적화
원장	Amazon QLDB	변경 불가능하고 암호로 확인할 수 있는 원장 변경 사항	기록 시스템, 의료, 공급망, 금융 기관

구현 단계

1. 데이터가 어떻게 구조화되었습니까?(예: 비정형, 키 값, 반정형, 관계형)

   a. 데이터가 비정형인 경우 Amazon S3 와 같은 객체 스토어 또는 Amazon DocumentDB와 같은 NoSQL 데이터베이스를 고려합니다.

   b. 키 값 데이터의 경우 DynamoDB, ElastiCache for Redis 또는 MemoryDB를 고려합니다.

   c. 데이터가 관계형 구조를 가진다면 어느 정도의 참조 무결성이 요구됩니까?

      i. 외래 키 제약 조건의 경우 Amazon RDS 및 Aurora 와 같은 관계형 데이터베이스가 원하는 무결성 수준을 제공할 수 있습니다.

      ii. 일반적으로 NoSQL 데이터 모델 내에서는 문서나 테이블을 조인하는 대신 단일 요청으로 검색하도록 데이터를 단일 문서 또는 문서 모음으로 비정규화합니다.

2. 원자성, 일관성, 격리, 내구성(ACID) 규정 준수가 필요합니까?

   a. 관계형 데이터베이스와 연결된 ACID 속성이 필요한 경우 Amazon RDS 및 Aurora와 같은 관계형 데이터베이스를 고려합니다.

3. 어떤 정합성 모델이 필요합니까?

   a. 애플리케이션에서 최종 일관성을 허용할 수 있다면 NoSQL 구현을 고려합니다. 다른 특성을 검토하면 가장 적합한 NoSQL 데이터베이스를 선택하는 데 도움이 됩니다.

   b. 철저한 일관성이 필요한 경우 DynamoDB 또는 Amazon RDS와 같은 관계형 데이터베이스로 강력히 일관된 읽기를 사용할 수 있습니다.

4. 어떤 쿼리 및 결과 형식을 지원해야 합니까?(예: SQL, CSV, Parque, Avro, JSON 등)

5. 어떤 데이터 유형, 필드 크기 및 전체 용량을 사용합니까?(예: 텍스트, 숫자, 공간, 시계열 계산, 바이너리 또는 블롭, 문서)

6. 시간이 흐르면서 스토리지 요구 사항이 어떻게 변경되며, 이러한 변화가 확장성에 어떤 영향을 미칩니까?

   a. 서버리스 데이터베이스(DynamoDB 및 Amazon Quantum Ledger Database)는 무한에 가까운 스토리지로 동적으로 확장됩니다.

   b. 관계형 데이터베이스는 프로비저닝된 스토리지에 대한 상한선이 있으며, 한도에 도달하면 샤딩과 같은 메커니즘을 통해 수평으로 분할해야 하는 경우가 많습니다.

7. 쓰기 쿼리 대비 읽기 쿼리의 비율은 얼마입니까? 캐싱이 성능을 향상시킬 가능성이 있습니까?

   a. 읽기 집약적인 워크로드는 캐싱 계층을 통해 이점을 얻을 수 있습니다. 이는 ElastiCache 또는 DAX일 수 있습니다 (데이터베이스가 DynamoDB인 경우).

   b. 읽기는 관계형 데이터베이스 (예: Amazon RDS)가 있는 읽기 전용 복제본으로 오프로드될 수 있습니다.

8. 저장 및 수정(OLTP - 온라인 트랜잭션 처리) 또는 검색 및 보고(OLAP - 온라인 분석 처리)가 높은 우선순위를 가집니까?

   a. 처리량(throughput)이 많은 트랜잭션 처리의 경우 DynamoDB 또는 Amazon DocumentDB와 같은 NoSQL 데이터베이스를 고려합니다.

   b. 분석 쿼리의 경우 Amazon Redshift 와 같은 열 형식 데이터베이스를 사용하거나 데이터를 Amazon S3로 내보내고 Athena 또는 QuickSight를 사용하여 분석을 수행하는 것을 고려합니다.

9. 이 데이터의 중요도와 필요한 보호 및 암호화 수준은 어느 정도입니까?

   a. Amazon RDS 및 Aurora 엔진은 AWS KMS를 사용하여 데이터 암호화를 지원합니다. 또한 Microsoft SQL Server 및 Oracle은 Amazon RDS를 사용할 때 기본 투명한 데이터 암호화(TDE)를 지원합니다.

   b. DynamoDB의 경우 IAM 과 함께 세분화된 액세스 제어를 통해 누가 키 수준에서 어떤 데이터에 액세스할지 제어할 수 있습니다.

10. 데이터에 필요한 내구성은 어느 정도입니까?

   a. Aurora는 리전 내 3개의 가용 영역에서 데이터를 자동으로 복제하므로, 데이터 손실 가능성이 적으면서도 데이터의 내구성이 매우 높아집니다.

   b. DynamoDB는 여러 가용 영역에 걸쳐 자동으로 복제되므로, 높은 가용성과 데이터 내구성을 제공합니다.

   c. Amazon S3는 99.999999999%의 내구성을 지원합니다. Amazon RDS 및 DynamoDB와 같은 많은 데이터베이스 서비스는 장기 보존 및 아카이브를 위해 Amazon S3로 데이터 내보내기를 지원합니다.

11. 할 것 Recovery Time Objective(RTO) 또는 Recovery Point Objective(RPO) 요건이 솔루션에 영향을 미칩니까?

   a. Amazon RDS, Aurora, DynamoDB, Amazon DocumentDB, Neptune은 모두 시점 복구와 온디맨드 백업 및 복원을 지원합니다.

   b. 고가용성 요구 사항을 위해 DynamoDB 테이블은 글로벌 테이블 기능을 사용하여 전역적으로 복제할 수 있으며, Aurora 클러스터는 글로벌 데이터베이스 기능을 사용하여 여러 리전에 걸쳐 복제할 수 있습니다. 나아가 S3 버킷은 교차 리전 복제를 사용하여 AWS 리전에 걸쳐 복제 가능합니다.

12. 상용 데이터베이스 엔진/라이선싱 비용에 대한 걱정을 덜고 싶습니까?

   a. Amazon RDS 또는 Aurora에서 PostgreSQL 및 MySQL과 같은 오픈 소스 엔진을 고려합니다.

   b. 그리고 AWS DMS 및 AWS SCT 를 활용하여 상용 데이터베이스 엔진에서 오픈 소스로 마이그레이션을 수행합니다.

13. 데이터베이스에 대한 운영상의 기대치는 어떠합니까? 관리형 서비스로 전환하는 것이 주요 관심사입니까?

a. Amazon EC2 대신 Amazon RDS를 활용하고 자체 호스팅한 NoSQL 데이터베이스 대신 DynamoDB 또는 Amazon DocumentDB를 활용하여 운영 오버헤드를 줄일 수 있습니다.

14.현재 데이터베이스에 어떻게 액세스합니까? 애플리케이션 액세스만 가능합니까? 아니면 비즈니스 인텔리전스(BI) 사용자와 기타 연결된 상용 애플리케이션이 있습니까?

a. 외부 도구에 의존하는 경우 해당 도구가 지원하는 데이터베이스와의 호환성을 유지해야 할 수도 있습니다. Amazon RDS는 Microsoft SQL Server, Oracle, MySQL, PostgreSQL 등 지원하는 다양한 엔진 버전과 완벽하게 호환됩니다.

15.잠재적인 데이터 관리 서비스 목록과 이러한 서비스를 가장 효과적으로 사용할 수 있는 위치는 다음과 같습니다.

a. 관계형 데이터베이스는 미리 정의된 스키마와 스키마 간의 관계를 사용하여 데이터를 저장합니다. 이러한 데이터베이스는 ACID(원자성, 일관성, 격리, 내구성) 트랜잭션을 지원하고 참조 무결성과 강력한 데이터 일관성을 유지하도록 설계되었습니다. 많은 기존 애플리케이션, 엔터프라이즈 리소스 계획(ERP), 고객 관계 관리(CRM) 및 전자 상거래의 데이터가 관계형 데이터베이스를 사용하여 저장됩니다. 이러한 데이터베이스 엔진의 다수를 Amazon EC2에서 실행하거나 AWS 관리형 데이터베이스 서비스: Amazon Aurora, Amazon RDS 및 Amazon Redshift중에서 선택할 수 있습니다.

b. 키-값 데이터베이스는 대개 대량의 데이터를 저장 및 검색하는 일반적인 액세스 패턴에 최적화되어 있습니다. 이 데이터베이스는 동시 요청의 양이 매우 많은 경우에도 빠른 응답 시간을 제공합니다. 트래픽이 많은 웹 앱, 전자 상거래 시스템 및 게임 애플리케이션은 키 값 데이터베이스의 일반적인 사용 사례입니다. AWS에서는 Amazon DynamoDB를 활용할 수 있습니다. Amazon DynamoDB는 인터넷 규모의 애플리케이션을 위한 보안, 백업 및 복원, 인 메모리 캐싱 기능이 내장된 탁월한 내구성을 지닌 완전관리형 다중 리전, 다중 마스터 데이터베이스입니다.

c. 인 메모리 데이터베이스는 데이터에 대한 실시간 액세스, 가장 짧은 지연 시간 및 가장 높은 처리량(throughput)이 필요한 애플리케이션에 사용됩니다. 이 데이터베이스는 데이터를 메모리에 직접 저장함으로써 밀리초 단위의 지연 시간도 허용되지 않는 애플리케이션에 마이크로초 단위의 지연 시간을 제공합니다. 애플리케이션 캐싱, 세션 관리, 게임 순위표 및 지리 공간 애플리케이션에 인메모리 데이터베이스를 사용할 수 있습니다. Amazon ElastiCache 는 다음과 호환되는 완전관리형 인메모리 데이터 스토어입니다. Redis 또는 Memcached. 애플리케이션의 내구성 요건이 더 높은 경우 Amazon MemoryDB for Redis 에서는 이를 조합하여 놀랍도록 빠르고 내구성이 뛰어난 인 메모리 데이터베이스 서비스를 제공합니다.

d. 도큐먼트 데이터베이스는 반정형 데이터를 JSON 유사 문서로 저장하도록 설계되었습니다. 이러한 데이터베이스는 개발자가 콘텐츠 관리, 카탈로그 및 사용자 프로필과 같은 애플리케이션을 신속하게 구축하고 업데이트하는 데 도움이 됩니다. Amazon DocumentDB 는 MongoDB 워크로드를 지원하는 완전관리형 문서 데이터베이스 서비스로, 탁월한 속도, 확장성 및 고가용성을 제공합니다.

e. 와이드 컬럼 스토어는 NoSQL 데이터베이스의 한 유형입니다. 테이블, 행 및 열을 사용하지만 관계형 데이터베이스와 달리 열의 이름과 형식은 동일한 테이블에서 행마다 다를 수 있습니다. 일반적으로 와이드 컬럼 스토어는 대규모 산업 앱에서 장비 유지 관리, 플릿 관리 및 라우팅 최적화를 위해 사용됩니다. Amazon Keyspaces(Apache Cassandra용) 는 와이드 컬럼 확장성 및 고가용성을 갖춘 관리형 Apache Cassandra 호환 데이터베이스 서비스입니다.

f. 그래프 데이터베이스는 밀접한 관계가 있는 그래프 데이터 세트 간에 밀리초 단위의 지연 시간으로 수백만 개의 관계를 탐색하고 쿼리해야 하는 애플리케이션을 위한 솔루션입니다. 많은 기업에서 사기 탐지, 소셜 네트워킹 및 추천 엔진에 그래프 데이터베이스를 사용합니다. Amazon Neptune 은 빠르고 안정적인 완전 관리형 그래프 데이터베이스 서비스로, 상호 연결성이 높은 데이터 세트를 활용하는 애플리케이션을 쉽게 구축하고 실행할 수 있습니다.

g. 시계열 데이터베이스는 시간이 지남에 따라 변화하는 데이터에서 효율적으로 분석 정보를 수집, 통합 및 도출합니다. IoT 애플리케이션, DevOps 및 산업용 텔레메트리에 시계열 데이터베이스를 활용할 수 있습니다. Amazon Timestream 은 IoT 및 운영 애플리케이션을 위한 고속의 확장 가능한 완전관리형 시계열 데이터베이스 서비스입니다. 이 서비스를 사용하면 하루에 수조 건의 이벤트를 손쉽게 저장하고 분석할 수 있습니다.

h. 원장 데이터베이스는 모든 애플리케이션에 대해 확장 가능하고 변경 불가능하며 암호화 방식으로 확인 가능한 트랜잭션 레코드를 유지하는 신뢰할 수 있는 중앙 집중식 권한을 제공합니다. 레코드 시스템, 공급망, 등록 및 심지어 은행 거래 시스템에 원장 데이터베이스가 사용되는 것을 볼 수 있습니다. Amazon Quantum Ledger Database(Amazon QLDB) 는 신뢰할 수 있는 중앙 기관이 소유한 투명하고 변경 불가능하며 암호화 방식으로 확인 가능한 트랜잭션 로그를 제공하는 완전관리형 원장 데이터베이스입니다. Amazon QLDB는 모든 애플리케이션 데이터 변경 사항을 추적하고 시간 경과에 따른 완전하고 확인 가능한 변경 기록을 유지합니다.

구현 계획의 작업 수준: 워크로드가 한 데이터베이스 솔루션에서 다른 데이터베이스 솔루션으로 이동하는 경우 데이터 및 애플리케이션을 리팩터링하는 데 많은 노력이 필요할 수 있습니다.

리소스

관련 문서:

- AWS 클라우드 데이터베이스
- AWS 데이터베이스 캐싱
- Amazon DynamoDB Accelerator
- Amazon Aurora 모범 사례
- Amazon Redshift 성능
- Amazon Athena 10가지 성능 향상 팁

- Amazon Redshift Spectrum 모범 사례
- Amazon DynamoDB 모범 사례
- EC2 및 Amazon RDS 간 선택
- Amazon ElastiCache 구현 모범 사례

관련 동영상:

- AWS 목적별 데이터베이스(DAT209-L)
- Amazon Aurora 스토리지 상세 설명: 작동 방식(DAT309-R)
- Amazon DynamoDB deep dive: Advanced design patterns(DAT403-R1)

관련 예시:

- Amazon Redshift 데이터 공유를 사용하여 데이터 패턴 최적화
- 데이터베이스 마이그레이션
- MS SQL Server - AWS Database Migration Service(DMS) 복제 데모
- 데이터베이스 현대화 실습 워크숍
- Amazon Neptune 샘플

PERF04-BP02 사용 가능한 옵션 평가

데이터 관리 솔루션을 선택하기 전에 사용 가능한 데이터베이스 옵션과 데이터베이스 옵션을 통해 성능을 최적화하는 방법을 이해해야 합니다. 로드 테스트를 통해 워크로드에 중요한 데이터베이스 지표를 식별하십시오. 데이터베이스 옵션을 탐색하면서 파라미터 그룹, 스토리지 옵션, 메모리, 컴퓨팅, 읽기 전용 복제본, 최종 일관성, 연결 풀링 및 캐싱 옵션과 같은 다양한 측면을 고려해야 합니다. 지표를 개선하려면 다음과 같은 다양한 구성 옵션을 사용해 보십시오.

원하는 결과: 워크로드에서 데이터 유형에 따라 하나 이상의 데이터베이스 솔루션을 사용할 수 있습니다. 데이터베이스 기능과 이점이 데이터 특성, 액세스 패턴 및 워크로드 요구 사항과 최적으로 맞아떨어집니다. 데이터베이스 성능과 비용을 최적화하려면 데이터 액세스 패턴을 평가하여 적절한 데이터베이스 옵션을 결정해야 합니다. 허용 가능한 쿼리 시간을 평가하여 선택한 데이터베이스 옵션이 요구 사항을 충족할 수 있는지 확인합니다.

일반적인 안티 패턴:

- 데이터 액세스 패턴을 식별하지 않습니다.

- 선택한 데이터 관리 솔루션의 구성 옵션을 알지 못합니다.
- 다른 사용 가능한 구성 옵션을 고려하지 않고 인스턴스 크기만 늘립니다.
- 선택한 솔루션의 확장 특성을 테스트하지 않습니다.

이 모범 사례 수립의 이점: 데이터베이스 옵션을 탐색하고 실험하여 인프라 비용을 절감하고, 성능 및 확장성을 향상하고, 워크로드를 유지하는 데 들여야 하는 수고를 줄일 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

- 불필요하게 타협하며 모든 데이터베이스에 적합하게 하나로 최적화해야 할 필요는 없습니다.
- 트래픽 패턴과 일치하도록 데이터베이스 솔루션을 구성하지 않으면 비용이 증가합니다.
- 확장 문제로 인해 운영 문제가 발생할 수 있습니다.
- 데이터가 필요한 수준으로 보호되지 않을 수 있습니다.

구현 가이드

데이터베이스 옵션을 구성할 수 있도록 워크로드 데이터 특성을 파악하십시오. 로드 테스트를 실행하면 주요 성능 지표와 병목 현상을 식별할 수 있습니다. 이러한 특성과 지표를 사용하여 데이터베이스 옵션을 평가하고 다른 구성을 실험합니다.

AWS 서비스	Amazon RDS, Amazon Aurora	Amazon DynamoD	Amazon Document B	Amazon ElastiCache	Amazon Neptune	Amazon Timestream	Amazon Keyspace	Amazon QLDB
컴퓨팅 규모 조정	인스턴스 크기 증가, 로드 변화에 따라 Aurora 서버리스 인스턴스 자동 확장	온디맨드 용량 모드로 자동 읽기/쓰기 확장 또는 프로비저닝된 용량 모드에	인스턴스 크기 증가	인스턴스 크기 증가, 클러스터에 노드 추가	인스턴스 크기 증가	용량을 조정하기 위해 자동 확장	온디맨드 용량 모드로 자동 읽기/쓰기 확장 또는 프로비저닝된 용량 모드에	용량을 조정하기 위해 자동 확장

AWS 서비스	Amazon RDS, Amazon Aurora	Amazon DynamoD	Amazon Document B	Amazon ElastiCache	Amazon Neptune	Amazon Timestream	Amazon Keyspace	Amazon QLDB
		서 프로비저닝된 읽기/쓰기 용량 자동 확장					서 프로비저닝된 읽기/쓰기 용량 자동 확장	
읽기 스케일 아웃	모든 엔진에서 읽기 전용 복제본 지원, Aurora에서 읽기 전용 복제본 인스턴스의 자동 확장 지원	프로비저닝된 읽기 용량 단위 증대	읽기 전용 복제본	읽기 전용 복제본	읽기 전용 복제본, 읽기 전용 복제본 인스턴스의 자동 확장 지원	자동 확장	프로비저닝된 읽기 용량 단위 증대	문서화된 동시성 제한까지 자동 스케일 업

AWS 서비스	Amazon RDS, Amazon Aurora	Amazon DynamoD	Amazon Document B	Amazon ElastiCache	Amazon Neptune	Amazon Timestream	Amazon Keyspace	Amazon QLDB
쓰기 스케일 아웃	인스턴스 크기 증가, 애플리케이션에서 쓰기를 일괄 처리하거나 데이터베이스 앞에 대기열 추가. 여러 인스턴스에 걸쳐 애플리케이션 수준 샤딩을 통한 수평 확장	프로비저닝된 쓰기 용량 단위 증대 파티션 수준 쓰기 제한을 방지하기 위해 최적의 파티션 키 보장	기본 인스턴스 크기 증가	클러스터 모드에서 Redis를 사용하여 쓰기를 여러 샤드로 분산	인스턴스 크기 증가	크기를 조정하는 동안 쓰기 요청이 제한될 수 있음. 제한 예외가 발생하는 경우 동일하거나 더 높은 처리량(throughput)으로 데이터를 계속 전송하여 자동 크기 조정 동시 쓰기 요청을 줄이기 위한 일괄 쓰기	프로비저닝된 쓰기 용량 단위 증대 파티션 수준 쓰기 제한을 방지하기 위해 최적의 파티션 키 보장	문서화된 동시성 제한까지 자동 스케일 업
엔진 구성	파라미터 그룹	해당 사항 없음	파라미터 그룹	파라미터 그룹	파라미터 그룹	해당 사항 없음	해당 사항 없음	해당 사항 없음

AWS 서비스	Amazon RDS, Amazon Aurora	Amazon DynamoD	Amazon Document B	Amazon ElastiCache	Amazon Neptune	Amazon Timestream	Amazon Keyspace	Amazon QLDB
캐싱	인 메모리 캐싱, 파라미터 그룹을 통해 구성 가능. ElastiCache for Redis와 같은 전용 캐시와 결합하여 일반적으로 액세스하는 항목에 대한 요청 오프로드	DAX(DAX 완전관리형 캐시 지원	인 메모리 캐싱. 필요에 따라 ElastiCache for Redis와 같은 전용 캐시와 결합하여 일반적으로 액세스하는 항목에 대한 요청 오프로드	기본 함수가 캐싱됨	쿼리 결과 캐시를 사용하여 읽기 전용 쿼리의 결과 캐시	Timestream에는 고성능 인 메모리 계층을 포함한 2개의 스토리지 계층 존재	ElastiCache for Redis와 같은 별도의 전용 캐시를 배포하여 일반적으로 액세스하는 항목에 대한 요청 오프로드	해당 사항 없음

AWS 서비스	Amazon RDS, Amazon Aurora	Amazon DynamoD	Amazon Document B	Amazon ElastiCache	Amazon Neptune	Amazon Timestream	Amazon Keyspace	Amazon QLDB
고가용성/재해 복구	프로덕션 워크로드에 대한 권장 구성으로 두 번째 가용 영역에서 대기 인스턴스를 실행하여 리전 내에서 복원력 제공. 리전 간 복원력의 경우 Aurora 글로벌 데이터베이스 사용 가능	리전 내에서 고가용성 제공. DynanoDB 글로벌 테이블을 사용하여 리전에 걸쳐 테이블 복제 가능	가용성을 위해 여러 가용 영역에 걸쳐 다수의 인스턴스 생성. 스냅샷은 리전 간에 공유할 수 있으며 클러스터는 DMS를 통해 복제하여 크로스 리전 복제/재해 복구 제공	프로덕션 클러스터 구성의 경우 보조 가용 영역에 노드를 하나 이상 생성할 것을 권장. ElastiCache 글로벌 데이터 스토어를 사용하여 여러 리전에서 클러스터 복제 가능	다른 가용 영역의 읽기 전용 복제본이 장애 조치 대상 역할을 함. 스냅샷은 리전 간에 공유할 수 있으며, Neptune 스트림으로 클러스터를 복제하여 2개의 서로 다른 리전에 있는 두 클러스터 간에 데이터 복제 가능	리전 내에서 고가용성 제공. 크로스 리전 복제를 수행하려면 Timestream SDK로 사용자 지정 애플리케이션을 개발해야 함	리전 내에서 고가용성 제공. 크로스 리전 복제에는 사용자 지정 애플리케이션 로직 또는 타사 도구 필요	리전 내에서 고가용성 제공. 리전 간에 복제하려면 Amazon QLDB 분개장의 콘텐츠를 S3 버킷으로 내보내고 크로스 리전 복제를 위한 버킷을 구성

구현 단계

1. 선택한 데이터베이스에 사용할 수 있는 구성 옵션은 무엇입니까?
   a. Amazon RDS 및 Aurora용 파라미터 그룹을 사용하면 캐시에 할당된 메모리나 데이터베이스의 시간대를 조정하는 것처럼 일반적인 데이터베이스 엔진 수준 설정을 조정할 수 있습니다.
   b. Amazon RDS, Aurora, Neptune, Amazon DocumentDB 등의 프로비저닝된 데이터베이스 서비스와 Amazon EC2에 배포된 데이터베이스 서비스의 경우 인스턴스 유형과 프로비저닝된 스토리지를 변경하고 읽기 전용 복제본을 추가할 수 있습니다.
   c. DynamoDB를 사용하면 온디맨드 및 프로비저닝이라는 두 용량 모드를 지정할 수 있습니다. 다양한 워크로드를 고려하여 언제든지 모드를 전환하고 프로비저닝 모드에서 할당된 용량을 늘릴 수 있습니다.
2. 워크로드에 읽기 또는 쓰기 작업이 많습니까?
   a. 읽기 오프로드(읽기 전용 복제본, 캐싱 등)에 사용할 수 있는 솔루션은 무엇입니까?
      i. DynamoDB 테이블의 경우 캐싱을 위해 DAX를 사용하여 읽기를 오프로드할 수 있습니다.
      ii. 관계형 데이터베이스의 경우 ElastiCache for Redis 클러스터를 생성하고 캐시에서 먼저 읽으며 요청한 항목이 없으면 데이터베이스로 폴백하도록 애플리케이션을 구성할 수 있습니다.
      iii. Amazon RDS, Aurora 등 관계형 데이터베이스와 Neptune, Amazon DocumentDB 등 프로비저닝된 NoSQL 데이터베이스는 모두 읽기 전용 복제본을 추가하여 워크로드의 읽기 부분을 오프로드할 수 있도록 지원합니다.
      iv. DynamoDB와 같은 서버리스 데이터베이스는 자동으로 크기가 조정됩니다. 워크로드를 처리하기에 충분한 읽기 용량 단위(RCU)가 프로비저닝되었는지 확인합니다.
   b. 쓰기 확장(파티션 키 샤딩, 대기열 추가 등)에 사용할 수 있는 솔루션은 무엇입니까?
      i. 관계형 데이터베이스의 경우 인스턴스 크기를 늘려 확장된 워크로드를 수용하거나 프로비저닝된 IOPS를 늘려 기본 스토리지에 대한 처리량(throughput)을 증대할 수 있습니다.
         - 데이터베이스에 직접 쓰는 대신 데이터베이스 앞에 대기열을 적용할 수도 있습니다. 이 패턴을 사용하면 데이터베이스에서 수집 정보를 분리하고 흐름 속도를 제어하여 데이터베이스가 과부하되지 않도록 할 수 있습니다.
         - 단기간 트랜잭션을 많이 만들지 않고 쓰기 요청을 일괄 처리하면 쓰기 볼륨이 많은 관계형 데이터베이스의 처리량(throughput)을 향상시킬 수 있습니다.
      ii. DynamoDB와 같은 서버리스 데이터베이스는 자동으로 쓰기 처리량(throughput)을 확장하거나 용량 모드에 따라 프로비저닝된 쓰기 용량 단위(WCU)를 조정하여 확장할 수 있습니다.
         - 그러나 지정된 파티션 키에 대한 처리량 한계에 도달한 경우에도 핫 파티션과 관련된 문제가 발생할 수 있습니다. 이 문제는 보다 고르게 분산된 파티션 키를 선택하거나 파티션 키를 쓰기 샤딩하여 해결할 수 있습니다.

3. 현재 또는 예상되는 초당 최대 트랜잭션(TPS)은 얼마입니까? 이 트래픽 볼륨과 이 볼륨 +X%를 통해 테스트하여 확장 특성을 이해합니다.
   a. PostgreSQL용 pg_bench와 같은 기본 도구를 사용하여 데이터베이스를 스트레스 테스트하고 병목 현상 및 확장 특성을 이해할 수 있습니다.
   b. 프로덕션과 유사한 트래픽을 캡처하여 실제 상황뿐만 아니라 합성 워크로드를 시뮬레이션할 수 있도록 반복해야 합니다.
4. 서버리스 또는 탄력적으로 확장 가능한 컴퓨팅을 사용하는 경우 이 확장이 데이터베이스에 미치는 영향을 테스트합니다. 필요한 경우 연결 관리 또는 풀링을 도입하여 데이터베이스에 미치는 영향을 줄이십시오.
   a. RDS 프록시는 Amazon RDS 및 Aurora와 함께 사용하여 데이터베이스에 대한 연결을 관리할 수 있습니다.
   b. DynamoDB와 같은 서버리스 데이터베이스에는 연계된 연결이 없으니, 프로비저닝된 용량 및 자동 확장 정책을 고려하여 로드 급증을 처리합니다.
5. 로드를 예측할 수 있습니까? 로드가 급증하는 경우가 있거나 비활성 기간이 있습니까?
   a. 비활성 기간이 있는 경우 해당 기간에 프로비저닝된 용량 또는 인스턴스 크기를 축소하는 것이 좋습니다. Aurora Serverless V2는 로드를 기반으로 자동 스케일 업 및 스케일 다운됩니다.
   b. 비 프로덕션 인스턴스의 경우 업무 시간 외에 이러한 인스턴스를 일시 중지하거나 정지하는 것을 고려합니다.
6. 액세스 패턴과 데이터 특성에 따라 데이터 모델을 세분화하고 분리해야 합니까?
   a. AWS DMS 또는 AWS SCT를 사용하여 데이터를 다른 서비스로 옮기는 것을 고려합니다.

구현 계획의 작업 수준:

이 모범 사례를 적용하려면 현재 데이터 특성과 지표를 알고 있어야 합니다. 이러한 지표를 수집하고 기준선을 설정한 다음 해당 지표를 사용하여 이상적인 데이터베이스 구성 옵션을 식별하는 작업에는 낮은 수준부터 중간 수준의 노력이 필요합니다. 이는 로드 테스트 및 실험을 통해 가장 효과적으로 검증됩니다.

리소스

관련 문서:

- AWS를 사용한 클라우드 데이터베이스
- AWS 데이터베이스 캐싱
- Amazon DynamoDB Accelerator

- Amazon Aurora 모범 사례
- Amazon Redshift 성능
- Amazon Athena 10가지 성능 향상 팁
- Amazon Redshift Spectrum 모범 사례
- Amazon DynamoDB 모범 사례

관련 동영상:

- AWS purpose-built databases(DAT209-L)
- Amazon Aurora storage demystified: How it all works(DAT309-R)
- Amazon DynamoDB deep dive: Advanced design patterns(DAT403-R1)

관련 예시:

- Amazon DynamoDB 예시
- AWS 데이터베이스 마이그레이션 샘플
- 데이터베이스 현대화 워크숍
- Amazon RDS for Postgress DB에서 파라미터를 사용한 작업

PERF04-BP03 데이터베이스 성능 지표 수집 및 기록

데이터 관리 시스템의 성능을 이해하려면 관련 지표를 추적해야 합니다. 이러한 지표를 통해 데이터 관리 리소스를 최적화하고, 워크로드 요구 사항을 충족하며, 워크로드의 성능을 명확하게 한눈에 파악할 수 있습니다. 데이터베이스 성능과 관련된 성능 측정값을 기록하는 도구, 라이브러리 및 시스템을 사용합니다.

데이터베이스가 호스팅되는 시스템과 관련된 지표(예: CPU, 스토리지, 메모리, IOPS)와 데이터 자체에 액세스하기 위한 지표(예: 초당 트랜잭션, 쿼리 속도, 응답 시간, 오류)가 있습니다. 모든 지원 또는 운영 직원이 이러한 지표에 쉽게 액세스할 수 있어야 하며, 지표에 추세와 이상 징후, 병목 현상을 식별할 수 있는 충분한 기록이 남겨져 있어야 합니다.

원하는 결과: 데이터베이스 워크로드의 성능을 모니터링하려면 일정 기간에 걸쳐 여러 성능 지표를 기록해야 합니다. 이를 통해 이상 징후를 탐지할 수 있을뿐더러 비즈니스 지표를 기준으로 성능을 측정하여 워크로드 요구 사항을 충족하는지 확인할 수 있습니다.

일반적인 안티 패턴:

- 지표에 대해 수동 로그 파일 검색만 사용합니다.
- 팀에서 사용하는 내부 도구에만 지표를 게시하고 워크로드를 종합적으로 바라보고 있지 않습니다.
- 선택한 모니터링 소프트웨어에서 기록한 기본 지표만 사용합니다.
- 문제가 발생한 경우에만 지표를 검토합니다.
- 시스템 수준 지표만 모니터링하며 데이터 액세스나 사용량 지표는 캡처하지 않습니다.

이 모범 사례 수립의 이점: 성능 기준선을 설정하면 워크로드의 정상적인 동작과 요구 사항을 이해하는 데 도움이 됩니다. 비정상적인 패턴을 더 빨리 식별하고 디버깅할 수 있어 데이터베이스의 성능과 신뢰성이 향상됩니다. 성능 저하 없이 최적의 비용을 보장하도록 데이터베이스 용량을 구성할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

- 정상 상태와 정상 성능 수준을 구분하지 못하면 문제를 식별하고 의사 결정을 내리는 데 어려움을 겪을 수 있습니다.
- 잠재적인 비용 절감 효과가 나타나지 않을 수 있습니다.
- 신뢰성 또는 성능 저하를 초래할 수 있는 성장 패턴이 식별되지 않습니다.

구현 가이드

데이터베이스 관련 지표를 식별, 수집 및 집계하고 상관 관계를 지정하십시오. 지표에는 데이터베이스를 지원하는 기본 시스템과 데이터베이스 지표가 모두 포함되어야 합니다. 기본 시스템 지표에는 CPU 활용률, 메모리, 사용 가능한 디스크 스토리지, 디스크 I/O, 네트워크 인바운드 및 아웃바운드 지표가 포함될 수 있습니다. 데이터베이스 지표는 초당 트랜잭션, 상위 쿼리, 평균 쿼리 속도, 응답 시간, 인덱스 사용량, 테이블 잠금, 쿼리 시간 초과 및 열린 연결 수로 구성 가능합니다. 이 데이터는 워크로드의 성능과 데이터베이스 솔루션의 사용 방법을 이해하는 데 중요합니다. 이러한 지표를 데이터 중심 전략에 포함시켜 워크로드의 리소스를 조정하고 최적화할 수 있습니다.

구현 단계:

1. 어떤 데이터베이스 지표를 추적해야 합니까?
    a. Amazon RDS 모니터링 지표
    b. 성능 개선 도우미를 사용한 모니터링
    c. 향상된 모니터링
    d. DynamoDB 지표

   e. DynamoDB DAX 모니터링
   f. MemoryDB 모니터링
   g. Amazon Redshift 모니터링
   h. 시계열 지표 및 차원
   i. Aurora용 클러스터 수준 지표
   j. Amazon Keyspaces 모니터링
   k. Amazon Neptune 모니터링
2. 데이터베이스 모니터링을 통해 운영 성능 이상 문제를 감지하는 기계 학습 솔루션의 이점을 얻을 수 있습니까?
   a. Amazon DevOps Guru for Amazon RDS 에서는 성능 문제에 대한 가시성을 제공하고 권장 수정 조치를 제안합니다.
3. SQL 사용과 관련해서 애플리케이션 수준 세부 정보가 필요합니까?
   a. AWS X-Ray 에서는 애플리케이션으로 계측하여 인사이트를 얻고, 단일 쿼리의 모든 데이터 포인트를 정리할 수 있습니다.
4. 현재 승인되어 사용 중인 로깅 및 모니터링 솔루션이 있습니까?
   a. Amazon CloudWatch 는 아키텍처의 리소스 전반에서 지표를 수집할 수 있습니다. 또한 사용자 지정 지표를 수집하고 게시하여 비즈니스 또는 파생 지표를 파악할 수도 있습니다. CloudWatch 또는 타사 솔루션을 사용하여 임계값 위반 시점을 나타내는 경보를 설정합니다.
5. 보안 및 운영 목표에 맞게 데이터 보존 정책을 식별하고 구성했습니까?
   a. CloudWatch 지표용 기본 데이터 보존
   b. CloudWatch Logs용 기본 데이터 보존

구현 계획의 작업 수준: 이 경우에는 보통 수준의 노력을 들여 모든 데이터베이스 리소스에서 지표를 식별, 추적, 수집 및 집계하고 상관 관계를 지정할 수 있습니다.

리소스

관련 문서:

- AWS 데이터베이스 캐싱
- Amazon Athena 10가지 성능 향상 팁
- Amazon Aurora 모범 사례
- Amazon DynamoDB Accelerator

- Amazon DynamoDB 모범 사례
- Amazon Redshift Spectrum 모범 사례
- Amazon Redshift 성능
- AWS를 사용한 클라우드 데이터베이스
- Amazon RDS 성능 개선 도우미

관련 동영상:

- AWS purpose-built databases(DAT209-L)
- Amazon Aurora storage demystified: How it all works(DAT309-R)
- Amazon DynamoDB deep dive: Advanced design patterns(DAT403-R1)

관련 예시:

- 레벨 100: CloudWatch 대시보드를 통한 모니터링
- AWS 데이터 모으기 지표 수집 프레임워크
- Amazon RDS 모니터링 워크숍

PERF04-BP04 액세스 패턴을 기준으로 데이터 스토리지 선택

워크로드의 액세스 패턴과 애플리케이션의 요구 사항을 바탕으로 사용할 최적의 데이터 서비스와 기술을 결정합니다.

원하는 결과: 식별되어 문서화된 데이터 액세스 패턴을 기반으로 데이터 스토리지를 선택했습니다. 여기에는 가장 일반적인 읽기, 쓰기 및 삭제 쿼리, 필요에 따른 계산 및 집계 필요성, 데이터 복잡성, 데이터 상호 종속성 및 필수 일관성 요구 사항이 포함될 수 있습니다.

일반적인 안티 패턴:

- 운영 관리를 간소화하기 위해 데이터베이스 엔진을 하나만 선택합니다.
- 시간이 지나면 데이터 액세스 패턴이 일관되게 유지될 것이라고 가정합니다.
- 애플리케이션에 복잡한 트랜잭션, 롤백 및 일관성 로직을 구현합니다.
- 데이터베이스가 높은 트래픽 버스트를 지원하도록 구성되어 있어 데이터베이스 리소스가 대부분의 시간 동안 유휴 상태로 유지됩니다.
- 트랜잭션 및 분석 용도로 공유 데이터베이스를 사용합니다.

이 모범 사례 수립의 이점: 액세스 패턴을 기반으로 데이터 스토리지를 선택하고 최적화하면 개발 복잡성을 줄이고 성능 기회를 최적화하는 데 도움이 됩니다. 읽기 전용 복제본, 글로벌 테이블, 데이터 파티셔닝 및 캐싱을 사용해야 하는 시기를 파악하면 워크로드 요구 사항에 따라 운영 오버헤드를 줄이고 규모를 확장할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험의 수준: 중간

구현 가이드

데이터 액세스 패턴을 식별하고 평가하여 올바른 스토리지 구성을 선택하세요. 각 데이터베이스 솔루션에는 스토리지 솔루션을 구성하고 최적화하는 데 사용하는 옵션이 있습니다. 수집된 지표와 로그로 옵션을 실험하여 최적의 구성을 찾을 수 있습니다. 다음 테이블에서 데이터베이스 서비스별 스토리지 옵션을 검토하세요.

AWS Services	Amazon RDS	Amazon Aurora	Amazon Dynamol	Amazon Docume B	Amazon ElastiCa he	Amazon Neptune	Amazon Timestre m	Amazon Keyspac	Amazon QLDB
Scaling Storage	Storage can be scaled up manually or configure d to scale automati ally to a maximur of 64 TiB based on engine types. Provisior	Storage scales automati ally up to maximur of 128 TiB and decrease when data is removed Maximur storage size also depends upon specific	Storage automati ally scales. Tables are unconstr ined in terms of size.	Storage scales automati ally up to maximur of 64 TiB. Starting Amazon Docume B 4.0 storage can decrease by compara e amounts for	Storage is in-memory , tied to instance type or count.	Storage scales automati ally can grow up to 128 TiB (or 64 TiB in few Regions) Upon data removal from, total allocated space remains	Organize your time series data to optimize query processi g and reduce storage costs. Retentio period can be configure d through in-memory	Scales table storage up and down automati ally as your applicati on writes, updates, and deletes data.	Storage automatic ally scales. Tables are unconstra ined in terms of size.

AWS Services	Amazon RDS	Amazon Aurora	Amazon Dynamol	Amazon Docume B	Amazon ElastiCa he	Amazon Neptune	Amazon Timestre m	Amazon Keyspac	Amazon QLDB
	ed storage cannot be decrease .	Aurora MySQL or Aurora Postgres engine versions.		data removal through dropping a collectio n or index. With Amazon Docume B 3.6 allocated space remains same and free space is reused when data volume increase .		same and is reused in the future.	and magnetic tiers.		

구현 단계:

1. 트랜잭션, 원자성, 일관성, 격리, 내구성(ACID) 규정 준수 및 일관된 읽기 요구 사항을 이해합니다. 모든 데이터베이스가 이를 지원하지는 않으며, 대부분의 NoSQL 데이터베이스는 최종 정합성 모델을 제공합니다.

2. 최적의 스토리지 솔루션을 식별하려면 글로벌 분산 애플리케이션의 트래픽 패턴, 지연 시간 및 액세스 요구 사항을 고려해야 합니다.
3. 쿼리 패턴, 무작위 액세스 패턴 및 일회성 쿼리를 분석합니다. 텍스트 및 자연어 처리, 시계열 및 그래프에 대한 고도로 전문화된 쿼리 기능을 중심으로 한 고려 사항도 염두에 두어야 합니다.
4. 예상되는 데이터 및 트래픽 증가를 식별하고 문서화하세요.
    a. Amazon RDS 및 Aurora에서는 문서화된 한도까지 스토리지 자동 확장을 지원합니다. 이 한도를 넘어서는 경우 데이터를 아카이브하고, 분석을 위해 과거 데이터를 집계하거나, 샤딩을 통해 수평으로 확장할 수 있도록 오래된 데이터는 Amazon S3(으)로 이전하는 것을 고려하세요.
    b. DynamoDB 및 Amazon S3에서는 거의 무제한에 가까운 스토리지 볼륨으로 자동 확장됩니다.
    c. EC2에서 실행 중인 Amazon RDS 인스턴스와 데이터베이스의 크기는 수동으로 조정할 수 있으며, EC2 인스턴스에는 나중에 새 EBS 볼륨을 추가하여 추가 스토리지로 활용할 수 있습니다.
    d. 인스턴스 유형은 활동 변화에 따라 변경될 수 있습니다. 예를 들어, 테스트 과정에서 더 작은 인스턴스로 시작했다가 서비스에 대한 프로덕션 트래픽을 수신하기 시작할 때 인스턴스를 확장하는 것도 가능합니다. Aurora Serverless V2는 로드 변화에 따라 자동으로 확장됩니다.
5. 정상 성능과 최고 성능(초당 트랜잭션(TPS) 및 초당 쿼리(QPS)), 일관성(ACID 및 최종 일관성)에 대한 요구 사항 기준치를 정합니다.
6. 솔루션 배포 측면과 데이터베이스 액세스 요구 사항(글로벌 복제본, 다중 AZ, 읽기 전용 복제본, 다중 쓰기 노드 등)을 문서화합니다.

구현 계획의 작업 수준: 낮음. 데이터 관리 솔루션에 대한 로그나 지표가 없는 경우 데이터 액세스 패턴을 식별하고 문서화하기 전에 이를 생성해야 합니다. 데이터 액세스 패턴을 파악하면 큰 어려움 없이 데이터 스토리지를 선택하고 구성할 수 있습니다.

리소스

관련 문서:

- AWS의 클라우드 데이터베이스
- Amazon RDS DB 인스턴스의 스토리지를 사용한 작업
- Amazon DocumentDB 스토리지
- AWS 데이터베이스 캐싱
- Amazon Timestream 스토리지
- Amazon Keyspaces의 스토리지
- Amazon ElastiCache FAQ

- Amazon Neptune 스토리지, 신뢰성 및 가용성
- Amazon Aurora 모범 사례
- Amazon DynamoDB Accelerator
- Amazon DynamoDB 모범 사례
- Amazon RDS 스토리지 유형
- Amazon RDS 인스턴스 클래스의 하드웨어 사양
- Aurora 스토리지 한도

관련 동영상:

- AWS 목적별 데이터베이스(DAT209-L)
- Amazon Aurora 스토리지 상세 설명: 작동 방식(DAT309-R)
- Amazon DynamoDB 심층 분석: 고급 설계 패턴(DAT403-R1)

관련 예시:

- AWS에서의 분산 부하 테스트로 실험 및 테스트

PERF04-BP05 액세스 패턴 및 지표를 기준으로 데이터 스토리지 최적화

성능을 최대한 높이려면 데이터 저장 또는 쿼리 방식을 최적화하는 성능 특성과 액세스 패턴을 사용합니다. 인덱싱, 키 분산, 데이터 웨어하우스 설계, 캐싱 전략 등의 최적화가 시스템 성능이나 전반적인 효율성에 미치는 영향을 측정합니다.

일반적인 안티 패턴:

- 지표에 대해 수동 로그 파일 검색만 사용합니다.
- 지표를 내부 도구에만 게시합니다.

이 모범 사례 정립의 이점: 워크로드에 필요한 지표가 충족되는지 확인하려면 읽기 및 쓰기와 관련된 데이터베이스 성능 지표를 모니터링해야 합니다. 이 데이터를 사용하여 데이터 스토리지 계층에 대한 읽기 및 쓰기 모두에 대한 새로운 최적화를 추가할 수 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

지표 및 패턴을 기준으로 데이터 스토리지 최적화: 보고된 지표를 사용하여 워크로드에서 성능이 낮은 영역을 식별하고 데이터베이스 구성 요소를 최적화합니다. 각 데이터베이스 시스템에서 평가해야 하는 성능 관련 특성(예: 데이터를 캐싱/인덱싱하거나 여러 시스템으로 분산하는 방법)은 서로 다릅니다. 최적화의 영향을 측정합니다.

리소스

관련 문서:

- AWS 데이터베이스 캐싱
- Amazon Athena 10가지 성능 향상 팁
- Amazon Aurora 모범 사례
- Amazon DynamoDB Accelerator
- Amazon DynamoDB 모범 사례
- Amazon Redshift Spectrum 모범 사례
- Amazon Redshift 성능
- AWS를 사용한 클라우드 데이터베이스
- DevOps Guru for RDS로 성능 이상 분석
- DynamoDB용 읽기/쓰기 용량 모드

관련 동영상:

- AWS purpose-built databases(DAT209-L)
- Amazon Aurora storage demystified: How it all works(DAT309-R)
- Amazon DynamoDB deep dive: Advanced design patterns(DAT403-R1)

관련 예시:

- Amazon DynamoDB 실습

PERF 5 네트워킹 솔루션을 구성하려면 어떻게 해야 합니까?

워크로드에 대한 최적의 네트워크 솔루션은 지연 시간, 처리량 요구 사항, 지터 및 대역폭에 따라 다릅니다. 위치 옵션은 사용자 또는 온프레미스 리소스와 같은 물리적 제약에 따라 결정됩니다. 엣지 로케이션 또는 리소스 배치를 통해 이러한 제약을 상쇄할 수 있습니다.

모범 사례

- PERF05-BP01 네트워킹이 성능에 미치는 영향 파악
- PERF05-BP02 사용 가능한 네트워킹 기능 평가
- PERF05-BP03 하이브리드 워크로드에 적절한 규모의 전용 연결 또는 VPN 선택
- PERF05-BP04 로드 밸런싱 및 암호화 오프로딩 활용
- PERF05-BP05 성능을 개선할 수 있는 네트워크 프로토콜 선택
- PERF05-BP06 네트워크 요구 사항에 따라 워크로드의 위치 선택
- PERF05-BP07 지표를 기준으로 네트워크 구성 최적화

PERF05-BP01 네트워킹이 성능에 미치는 영향 파악

네트워크 관련 결정 사항이 워크로드 성능에 영향을 주는 방식을 분석하고 파악합니다. 네트워크는 애플리케이션 구성 요소, 클라우드 서비스, 엣지 네트워크 및 온프레미스 데이터 간의 연결을 담당하므로, 워크로드 성능에 큰 영향을 미칠 수 있습니다. 사용자 경험은 워크로드 성능 외에 네트워크 지연 시간, 대역폭, 프로토콜, 위치, 네트워크 정체, 지터, 처리량(throughput) 및 라우팅 규칙에도 영향을 받습니다.

원하는 결과: 지연 시간, 패킷 크기, 라우팅 규칙, 프로토콜 및 지원 트래픽 패턴을 포함한 워크로드의 네트워킹 요구 사항 목록을 문서화합니다. 사용 가능한 네트워킹 솔루션을 검토하고 워크로드 네트워킹 특성을 충족하는 서비스를 파악합니다. 클라우드 기반 네트워크는 빠르게 재구축될 수 있으므로 성능 효율성을 개선하려면 네트워크 아키텍처를 지속적으로 변경해야 합니다.

일반적인 안티 패턴:

- 모든 트래픽이 기존 데이터 센터를 통과합니다.
- 실제 사용 요구 사항을 파악하지 않고 Direct Connect 세션을 초과 구축합니다.
- 네트워킹 솔루션을 정의할 때 워크로드 특성과 암호화 오버헤드를 고려하지 않습니다.
- 클라우드의 네트워킹 솔루션에 온프레미스 개념과 전략을 적용합니다.

이 모범 사례 수립의 이점: 네트워킹이 워크로드 성능에 미치는 영향을 이해하면 잠재적인 병목 현상을 식별하고, 사용자 경험을 개선하고, 신뢰성을 높이고, 워크로드 변화에 따라 운영 유지 관리 작업을 줄이는 데 도움이 됩니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

워크로드의 중요한 네트워크 성능 지표를 식별하고 해당 네트워킹 특성을 파악합니다. 벤치마킹 또는 로드 테스트를 사용하여 데이터 기반 접근 방식의 일부로 요구 사항을 정의하고 문서화합니다. 이 데이터를 사용하여 네트워킹 솔루션이 제한되는 위치를 식별하고, 워크로드를 개선하는 구성 옵션을 검사합니다. 사용 가능한 클라우드 네이티브 네트워킹 기능과 옵션 및 이러한 기능과 옵션이 요구 사항을 기준으로 워크로드 성능에 미치는 영향을 파악합니다. 각 네트워킹 기능에는 장단점이 있으며, 워크로드 특성에 맞게 구성하고 필요에 따라 확장할 수 있습니다.

구현 단계:

1. 네트워킹 성능 요구 사항을 정의하고 문서화합니다.
    a. 네트워크 지연 시간, 대역폭, 프로토콜, 위치, 트래픽 패턴(급증 및 빈도), 처리량(throughput), 암호화, 검사 및 라우팅 규칙과 같은 지표가 포함됩니다.
2. 기본 네트워킹 특성을 파악합니다.
    a. VPC 흐름 로그
    b. AWS Transit Gateway 지표
    c. AWS PrivateLink 지표
3. 애플리케이션 네트워킹 특성을 파악합니다.
    a. 탄력적 네트워크 어댑터
    b. AWS App Mesh 지표
    c. Amazon API Gateway 지표
4. 엣지 네트워킹 특성을 파악합니다.
    a. Amazon CloudFront 지표
    b. Amazon Route 53 지표
    c. AWS Global Accelerator 지표
5. 하이브리드 네트워킹 특성을 파악합니다.
    a. Direct Connect 지표
    b. AWS Site-to-Site VPN 지표

   c. AWS Client VPN 지표
   d. AWS 클라우드 WAN 지표
6. 보안 네트워킹 특성을 캡처합니다.
   a. AWS Shield, WAF, Network Firewall 지표
7. 추적 도구로 엔드 투 엔드 성능 지표를 파악합니다.
   a. AWS X-Ray
   b. Amazon CloudWatch RUM
8. 네트워크 성능을 벤치마크하고 테스트합니다.
   a. 벤치마크 네트워크 처리량(throughput): 인스턴스가 동일한 VPC에 있을 때 EC2 네트워크 성능에 영향을 미칠 수 있는 몇 가지 요인입니다. 동일한 VPC에 있는 EC2 Linux 인스턴스 간의 네트워크 대역폭을 측정합니다.
   b. 로드 테스트를 수행하여 네트워킹 솔루션 및 옵션을 실험합니다.

구현 계획의 작업 수준: 보통 수준의 노력을 들여 워크로드 네트워킹 요구 사항, 옵션 및 사용 가능한 솔루션을 문서화합니다.

리소스

관련 문서:

- Application Load Balancer
- Linux 기반 EC2 향상된 네트워킹
- Windows의 EC2 향상된 네트워킹
- EC2 배치 그룹
- Linux 인스턴스에서 ENA(Elastic Network Adapter)를 사용하여 향상된 네트워킹 활성화
- Network Load Balancer
- AWS의 네트워킹 제품
- Transit Gateway
- Amazon Route 53에서 지연 시간 기반 라우팅으로 전환
- VPC 엔드포인트
- VPC 흐름 로그

관련 동영상:

- AWS 및 하이브리드 AWS 네트워크 아키텍처에 대한 연결성(NET317-R1)
- Optimizing Network Performance for Amazon EC2 Instances(CMP308-R1)
- 애플리케이션의 글로벌 네트워크 성능 향상
- EC2 인스턴스 및 성능 최적화 모범 사례
- Amazon EC2 인스턴스의 네트워크 성능 최적화
- Well-Architected Framework의 네트워킹 모범 사례 및 팁
- 대규모 마이그레이션의 AWS 네트워킹 모범 사례

관련 예시:

- AWS Transit Gateway 및 확장 가능한 보안 솔루션
- AWS 네트워킹 워크숍

PERF05-BP02 사용 가능한 네트워킹 기능 평가

클라우드에서 성능을 높일 수 있는 네트워킹 기능을 평가합니다. 테스트, 지표 및 분석을 통해 이러한 기능의 영향을 측정할 수 있습니다. 예를 들어 지연 시간, 패킷 손실 또는 지터를 줄이는 데 사용할 수 있는 네트워크 수준 기능을 활용합니다.

많은 서비스가 성능 개선을 위해 개발되며 다른 서비스는 일반적으로 네트워크 성능을 최적화하기 위한 기능을 제공합니다. AWS Global Accelerator 및 Amazon CloudFront 등과 같은 서비스는 성능을 개선하기 위해 개발된 반면에 대부분의 다른 서비스는 네트워크 트래픽을 최적화하기 위한 제품 기능을 갖추고 있습니다. 그러므로 워크로드 성능을 개선하는 서비스 기능(예: EC2 인스턴스 네트워크 기능, 강화된 네트워킹 인스턴스 유형, Amazon EBS 최적화 인스턴스, Amazon S3 Transfer Acceleration, CloudFront)을 고려해야 합니다.

원하는 결과: 워크로드 내 구성 요소 인벤토리를 문서화했으며 구성 요소별로 어떤 네트워킹 구성이 성능 요구 사항을 충족하도록 돕는지 파악했습니다. 네트워킹 기능을 평가한 후 성능 지표를 실험 및 측정하여 사용 가능한 기능을 사용하는 방법을 파악했습니다.

일반적인 안티 패턴:

- 모든 워크로드를 최종 사용자에게 가까운 AWS 리전이 아니라 본사에 가장 가까운 AWS 리전에 저장합니다.
- 워크로드 성능 벤치마크에 실패했는데, 실패한 벤치마크를 기준으로 계속해서 워크로드 성능을 평가하고 있습니다.
- 성능 개선 옵션을 확인하기 위해 서비스 구성을 검토하지 않습니다.

이 모범 사례 확립의 이점: 모든 서비스 기능 및 옵션을 평가하면 워크로드 성능을 개선하고 인프라 비용을 줄이고 워크로드를 유지 관리하는 데 필요한 작업을 줄이며 전반적인 보안 상태를 개선할 수 있습니다. 전 세계에 분산된 AWS 백본을 사용하여 고객에게 최상의 네트워킹 환경을 제공할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

사용 가능한 네트워크 관련 구성 옵션과 이러한 옵션이 워크로드에 미치는 영향을 검토합니다. 성능 최적화에서는 이러한 옵션이 아키텍처와 상호 작용하는 방식과 측정된 성능과 사용자의 체감 성능에 미치는 영향을 파악하는 것이 중요합니다.

구현 단계:

1. 워크로드 구성 요소 목록을 만듭니다.
   a. AWS 클라우드 WAN을 사용하여 조직 네트워크를 구축, 관리 및 모니터링합니다.
   b. Network Manager를 사용하여 네트워크의 상태를 파악합니다. 기존 구성 관리 데이터베이스(CMDB) 도구를 사용하거나 AWS Config 등과 같은 도구를 사용하여 워크로드 인벤토리 또는 구성 방식을 생성합니다.
2. 기존 워크로드인 경우에는 성능 지표의 벤치마크를 식별하고 문서화하여 병목 현상과 개선해야 할 부분에 집중적으로 살펴봅니다. 성능 관련 네트워킹 지표는 비즈니스 요구 사항 및 워크로드 특성을 기준으로 워크로드에 따라 다릅니다. 무엇보다 대역폭, 지연 시간, 패킷 손실, 지터 및 재전송 등과 같은 지표가 워크로드 검토에 중요할 수 있습니다.
3. 새로운 워크로드인 경우 로드 테스트 를 수행하여 성능 병목 현상을 파악합니다.
4. 파악한 성능 병목 현상에 대해 솔루션의 구성 옵션을 검토하여 성능 개선 기회를 파악합니다.
5. 네트워크 경로 또는 루트를 모르는 경우 Network Access Analyzer 를 사용하여 파악합니다.
6. 지연 시간을 더욱 단축할 수 있는 네트워크 프로토콜을 검토합니다.
   - PERF05-BP05 성능을 개선할 수 있는 네트워크 프로토콜 선택
7. 여러 위치에서 AWS Site-to-Site VPN을 사용하여 AWS 리전에 연결하는 경우에는 가속화된 Site-to-Site VPN 연결 을 살펴보고 네트워킹 성능을 개선할 수 있는 기회가 있는지 검토합니다.
8. 워크로드 트래픽이 여러 계정에 분산된 경우 네트워크 토폴로지 및 서비스를 평가하여 지연 시간을 단축합니다.
   - 여러 계정을 연결할 때 VPC 피어링 및 AWS Transit Gateway 사이 운영 및 성능 균형을 평가합니다. AWS Transit Gateway에서는 AWS Site-to-Site VPN 처리량이 단일 IPsec 최대 한도 를 초과한 경우 다중 경로를 사용하여 지원합니다. Amazon VPC과(와) AWS Transit Gateway 간 트래픽은 프라이빗 AWS 네트워크에서 유지되어 인터넷에 노출되지 않습니다. AWS Transit Gateway

은(는) 수천 개의 AWS 계정 간에 걸쳐 있고 온프레미스 네트워크에 연결되는 모든 VPC를 상호 연결하는 방식을 간소화합니다. AWS Transit Gateway은(는) Resource Access Manager를 사용하여 여러 계정 간에 공유합니다. 글로벌 네트워크 트래픽의 상태를 파악하기 위해 Network Manager 를 사용하여 네트워크 지표를 중앙에서 확인할 수 있습니다.

9. 위치를 검토하고 사용자와 워크로드 간 거리를 최소화합니다.

   a. AWS Global Accelerator 은(는) Amazon Web Services 글로벌 네트워크 인프라를 사용하여 사용자 트래픽의 성능을 최대 60%까지 개선하는 네트워킹 서비스입니다. 인터넷 연결이 혼잡한 경우 AWS Global Accelerator은(는) 애플리케이션 경로를 최적화하여 패킷 손실, 지터 및 지연 시간을 지속적으로 줄입니다. DNS 구성을 업데이트하거나 클라이언트용 애플리케이션을 변경하지 않고도 가용 영역 또는 AWS 리전 간에 엔드포인트 이동을 간소화하는 고정 IP 주소를 제공합니다.

   b. Amazon CloudFront 은(는) 워크로드 콘텐츠 전송 성능 및 지연 시간을 전역적으로 개선할 수 있습니다. CloudFront은(는) 콘텐츠를 캐시하고 최종 사용자에 대한 지연 시간을 단축할 수 있는 접속 지점을 전 세계에 410개 이상 보유하고 있습니다.

   c. Amazon Route 53은(는) 지연 시간 기반 라우팅, 지리적 위치 라우팅, 지리 근접 라우팅및 IP 기반 라우팅 옵션을 제공하여 전 세계 고객을 대상으로 워크로드 성능을 개선할 수 있습니다. 워크로드 트래픽 및 사용자 위치를 검토하여 어떤 라우팅 옵션이 워크로드 성능을 최적화하는지 파악합니다.

10스토리지 IOP를 개선하기 위한 추가 Amazon S3 기능을 평가합니다.

   a. Amazon S3 Transfer Acceleration 은 외부 사용자가 CloudFront의 네트워킹 최적화 이점을 활용하여 Amazon S3에 데이터를 업로드하도록 하는 기능입니다. 이렇게 하면 AWS 클라우드 전용 연결을 사용할 수 없는 원격 위치에서 대량의 데이터를 전송할 수 있습니다.

   b. Amazon S3 다중 리전 액세스 포인트 는 콘텐츠를 여러 리전으로 복제하고 액세스 포인트 하나를 제공하여 워크로드를 간소화합니다. 다중 리전 액세스 포인트를 사용하는 경우 가장 낮은 지연 시간 버킷을 식별하는 서비스로 데이터를 요청하거나 Amazon S3에 데이터를 쓸 수 있습니다.

11.컴퓨팅 리소스 네트워크 대역폭을 검토합니다.

   a. EC2 인스턴스, 컨테이너 및 Lambda 함수에서 사용하는 탄력적 네트워크 인터페이스(ENI)는 흐름 기준으로 제한됩니다. 배치 그룹을 검토하여 EC2 네트워킹 처리량을 최적화할 수 있습니다.. 흐름 기준에서 병목 현상을 방지하려면 여러 흐름을 사용하도록 애플리케이션을 설계합니다. 컴퓨팅 관련 네트워크 지표를 모니터링하고 이러한 지표에 대한 가시성을 얻으려면 CloudWatch 지표 및 `ethtool`을 사용합니다. `ethtool` 은 ENA 드라이버에 포함되어 있으며 사용자 지정 지표로 CloudWatch에 게시할 수 있는 추가 네트워크 관련 지표를 노출할 수 있습니다.

   b. 최신 버전의 EC2 인스턴스는 강화된 네트워킹 기능을 활용할 수 있습니다. N-시리즈 EC2 인스턴스(예: `M5n` 및 `M5dn`)는 4세대 사용자 지정 Nitro card를 활용하여 단일 인스턴스에 최대 100Gbps

의 네트워크 처리량을 전달합니다. 이러한 인스턴스는 기본 M5 인스턴스에 비해 4배 더 높은 네트워크 대역폭과 패킷 프로세스를 제공하며 네트워크 집약적 애플리케이션에 적합합니다.

c. Amazon Elastic Network Adapter (ENA)는 클러스터 배치 그룹 내에서 인스턴스에 대해 더 나은 처리량을 제공하여 한층 더 최적화합니다.

d. Elastic Fabric Adapter (EFA)는 AWS에서 높은 수준의 대규모 노드 간 통신이 필요한 워크로드를 실행할 때 사용할 수 있는 Amazon EC2 인스턴스용 네트워크 인터페이스입니다. EFA를 사용하면 MPI(메시지 전달 인터페이스)를 사용하는 HPC(고성능 컴퓨팅) 애플리케이션과 NCCL(NVIDIA Collective Communications Library)을 사용하는 ML(기계 학습) 애플리케이션을 수천 개 CPU 또는 GPU로 확장할 수 있습니다.

e. Amazon EBS 최적화 인스턴스는 최적화된 구성 스택을 사용하고 Amazon EBS I/O를 늘리기 위해 전용 용량을 추가로 제공합니다. 이 최적화를 수행하면 Amazon EBS I/O와 인스턴스의 기타 트래픽 간에 경합이 최소화되므로 EBS 볼륨의 성능을 최대한 높일 수 있습니다.

구현 계획의 작업 수준:

이 모범 사례를 확립하려면 네트워크 성능에 영향을 미치는 현재 워크로드 구성 요소 옵션을 알아야 합니다. 구성 요소 수집, 네트워크 개선 옵션 평가, 개선 사항 실험, 구현 및 문서화는 작업 수준이 낮음 에서 중간 입니다.

리소스

관련 문서:

- Amazon EBS - 최적화 인스턴스
- Application Load Balancer
- Amazon EC2 인스턴스 네트워크 대역폭
- Linux 기반 EC2 향상된 네트워킹
- Windows의 EC2 향상된 네트워킹
- EC2 배치 그룹
- Linux 인스턴스에서 ENA(Elastic Network Adapter)를 사용하여 향상된 네트워킹 활성화
- Network Load Balancer
- AWS의 네트워킹 제품
- AWS Transit Gateway
- Amazon Route 53에서 지연 시간 기반 라우팅으로 전환

- VPC 엔드포인트
- VPC 흐름 로그
- 클라우드 CMDB 구축
- AWS Transit Gateway을(를) 사용하여 VPN 처리량 스케일링

관련 동영상:

- Connectivity to AWS and hybrid AWS network architectures(AWS 및 하이브리드 AWS 네트워크 아키텍처에 대한 연결성)(NET317-R1)
- Optimizing Network Performance for Amazon EC2 Instances(CMP308-R1)
- AWS Global Accelerator

관련 예시:

- AWS Transit Gateway 및 확장 가능한 보안 솔루션
- AWS 네트워킹 워크숍

PERF05-BP03 하이브리드 워크로드에 적절한 규모의 전용 연결 또는 VPN 선택

AWS에서 온프레미스 및 클라우드 리소스를 연결하는 데 공용 네트워크가 필요한 경우 성능 요구 사항을 충족할 수 있는 충분한 대역폭을 확보해야 합니다. 하이브리드 워크로드에 대한 대역폭 및 지연 시간 요구 사항을 예측하세요. 연결 옵션에 대한 크기 요구 사항은 이러한 수치를 바탕으로 결정됩니다.

원하는 결과: 하이브리드 네트워킹이 필요한 워크로드를 배포할 때 전용 연결 또는 가상 프라이빗 네트워크(VPN)와 같은 여러 연결 구성 옵션을 사용할 수 있습니다. 사용자의 위치와 클라우드 간에 적절한 대역폭 및 암호화 요구 사항이 있는지 확인하면서 각 워크로드에 적합한 연결 유형을 선택합니다.

일반적인 안티 패턴:

- 모든 워크로드 요구 사항(대역폭, 지연 시간, 지터, 암호화 및 트래픽 요구 사항)을 이해하거나 식별하지 못합니다.
- 백업 또는 병렬 연결 옵션은 평가하지 않습니다.

이 모범 사례 확립의 이점: 적절한 크기의 하이브리드 네트워크 솔루션을 선택하고 구성하면 워크로드의 신뢰성이 향상되고 성능을 높일 기회가 극대화됩니다. 워크로드 요구 사항을 파악하고, 미리 계획하

고, 하이브리드 솔루션을 평가하여 비용이 많이 드는 물리적 네트워크 변경과 운영 오버헤드를 최소화하는 동시에 출시 시간을 단축할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

대역폭 요구 사항을 기반으로 하이브리드 네트워킹 아키텍처를 개발합니다. 하이브리드 애플리케이션의 대역폭 및 지연 시간 요구 사항을 추정합니다. 전용 네트워크 연결 또는 인터넷 기반 VPN 사용 간에 적절한 연결 옵션을 고려합니다.

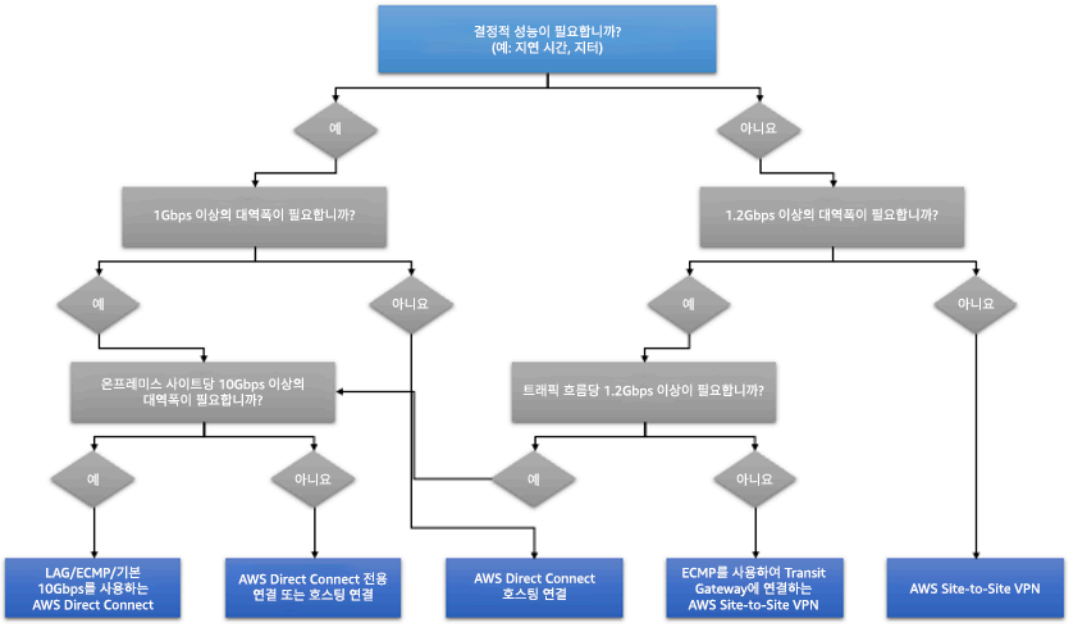
전용 연결은 사설 회선을 통해 네트워크 연결을 설정합니다. 일관된 성능을 달성하면서 고대역폭, 짧은 지연 시간이 필요할 때 적합합니다. VPN 연결은 인터넷을 통해 보안 연결을 설정합니다. 기존 인터넷 연결을 사용하여 암호화된 연결이 필요할 때 적합합니다.

대역폭 요구 사항에 따라 단일 VPN 또는 전용 연결로는 충분하지 않을 수 있으며, 여러 연결에 걸쳐 트래픽 로드 밸런싱을 활성화하도록 하이브리드 설정을 설계해야 합니다.

구현 단계

1. 하이브리드 애플리케이션의 대역폭 및 지연 시간 요구 사항을 추정합니다.
   a. AWS로 이동하는 기존 앱의 경우 내부 네트워크 모니터링 시스템의 데이터를 활용합니다.
   b. 모니터링 데이터가 없는 새로운 앱 또는 기존 앱의 경우 제품 소유자에게 문의하여 적절한 성능 지표를 도출하고 우수한 사용자 경험을 제공합니다.
2. 연결 옵션으로 전용 연결 또는 VPN을 선택합니다. 모든 워크로드 요구 사항(암호화, 대역폭 및 트래픽 요구)을 기반으로 AWS Direct Connect 또는 AWS Site-to-Site VPN(또는 둘 다)을 선택할 수 있습니다. 다음 다이어그램은 적절한 연결 유형을 선택하는 데 도움이 됩니다.
   a. 전용 연결을 사용하려면 AWS Direct Connect가 필요할 수 있으며, 이는 프라이빗 네트워크 연결이기 때문에 보다 예측 가능하고 일관된 성능을 제공합니다. AWS Direct Connect는 전용 연결 또는 호스팅된 연결을 사용하여 50Mbps에서 100Gbps까지 AWS 환경에 대한 전용 연결을 제공합니다. 이렇게 하면 지연 시간을 관리/제어하고 대역폭을 프로비저닝할 수 있으므로 워크로드를 다른 환경에 효율적으로 연결할 수 있습니다. AWS Direct Connect 파트너를 사용하면 여러 환경에 엔드 투 엔드로 연결할 수 있으며 일관된 성능을 갖춘 확장 네트워크를 제공할 수 있습니다. AWS는 기본 100Gbps, LAG(Link Aggregation Group) 또는 BGP Electocost Multipath(ECMP)를 사용하여 스케일링 Direct Connect 연결 대역폭을 제공합니다.
   b. VPN 연결을 생각한다면 AWS 관리형 VPN을 선택하는 것이 좋습니다. AWS Site-to-Site VPN은 IPsec(인터넷 프로토콜 보안) 프로토콜을 지원하는 관리형 VPN 서비스를 제공합니다. VPN 연결이 생성되면 각 VPN 연결에는 고가용성을 위해 두 개의 터널이 포함됩니다. AWS Transit

Gateway를 사용하면 여러 VPC 간의 연결성을 단순화하고 단일 VPN 연결을 사용하여 AWS Transit Gateway에 연결된 VPC에 연결할 수 있습니다. 또한 AWS Transit Gateway를 사용하면 여러 VPN 터널에서 ECMP(Equal Cost Multi-Path) 라우팅 지원을 활성화하여 1.25Gbps IPsec VPN 처리량 제한 이상으로 확장할 수 있습니다.



결정론적 성능 흐름도

구현 계획의 작업 수준: 높음. 하이브리드 네트워크에 대한 워크로드 요구 사항을 평가하고 하이브리드 네트워킹 솔루션을 구현하는 데 상당한 노력이 필요합니다.

리소스

관련 문서:

- Network Load Balancer
- AWS의 네트워킹 제품
- AWS Transit Gateway
- Amazon Route 53에서 지연 시간 기반 라우팅으로 전환
- VPC 엔드포인트
- VPC 흐름 로그
- AWS Site-to-Site VPN

- Building a Scalable and Secure Multi-VPC AWS Network Infrastructure(확장 가능하고 안전한 멀티 VPC AWS 네트워크 인프라 구축)
- AWS Direct Connect
- Client VPN

관련 동영상:

- Connectivity to AWS and hybrid AWS network architectures(NET317-R1)(AWS 및 하이브리드 AWS 네트워크 아키텍처에 대한 연결(NET317-R1))
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1)(Amazon EC2 인스턴스의 네트워크 성능 최적화(CMP308-R1))
- AWS Global Accelerator
- AWS Direct Connect
- Transit Gateway Connect
- VPN Solutions(VPN 솔루션)
- Security with VPN Solutions(VPN 솔루션을 사용한 보안)

관련 예시:

- AWS Transit Gateway 및 확장 가능한 보안 솔루션
- AWS 네트워킹 워크숍

PERF05-BP04 로드 밸런싱 및 암호화 오프로딩 활용

로드 밸런서를 사용하여 대상 리소스의 성능 효율성을 최적화하고 시스템의 응답성을 개선합니다.

원하는 결과: 트래픽을 처리할 컴퓨팅 리소스의 수를 줄입니다. 대상에서 리소스 사용 불균형이 발생하지 않도록 합니다. 컴퓨팅 집약적인 태스크를 로드 밸런서로 오프로드합니다. 클라우드 탄력성과 유연성을 활용하여 성능을 개선하고 아키텍처를 최적화합니다.

일반적인 안티 패턴:

- 로드 밸런서 유형을 선택할 때 워크로드 요구 사항을 고려하지 않습니다.
- 성능 최적화 시 로드 밸런서 기능을 활용하지 않습니다.
- 워크로드는 로드 밸런서 없이 인터넷에 직접 노출됩니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

로드 밸런서는 워크로드의 진입점 역할을 하며 여기에서 컴퓨팅 인스턴스 또는 컨테이너와 같은 백엔드 대상으로 트래픽을 분산합니다. 아키텍처를 최적화하는 첫 번째 단계는 적절한 로드 밸런서 유형을 선택하는 것입니다.

먼저 프로토콜(예: TCP, HTTP, TLS 또는 WebSockets), 대상 유형(예: 인스턴스, 컨테이너 또는 서버리스), 애플리케이션 요구 사항(장기간 실행되는 연결, 사용자 인증 또는 고정성 등) 및 배치(예: 리전, 로컬 영역, Outpost 또는 영역 격리)와 같은 워크로드 특성을 나열해 봅니다.

적절한 로드 밸런서를 선택한 후 해당 기능을 활용하면 백엔드가 트래픽을 처리하는 데 필요한 노력을 줄일 수 있습니다.

예를 들어 Application Load Balancer(ALB)와 Network Load Balancer(NLB)를 모두 사용하면 SSL/TLS 암호화 오프로딩을 수행할 수 있습니다. 이를 통해 대상에서 완료되는 CPU 집약적 TLS 핸드셰이크를 방지하고 인증서 관리를 개선할 수 있습니다.

로드 밸런서에서 SSL/TLS 오프로딩을 구성하면 백엔드에 암호화되지 않은 트래픽을 전달하고 백엔드 리소스를 확보하고 클라이언트에 대한 응답 시간을 개선하는 동시에 클라이언트에서 들어오고 나가는 트래픽의 암호화를 담당하게 됩니다.

Application Load Balancer도 대상에서 지원할 필요 없이 HTTP2 트래픽을 처리할 수 있습니다. HTTP2가 TCP 연결을 보다 효율적으로 사용하므로 이렇게 간단한 결정이 애플리케이션 응답 시간을 개선할 수 있습니다.

로드 밸런서를 활용하면 컨테이너 및 서버리스와 같은 다양한 백엔드 유형에 트래픽을 분산하여 아키텍처를 보다 유연하게 만들 수도 있습니다. 예를 들어 Application Load Balancer는 헤더, 메서드 또는 패턴과 같은 요청 파라미터를 기반으로 트래픽을 다른 대상 그룹으로 전달하는 리스너 규칙으로 구성할 수 있습니다.

아키텍처를 정의할 때 워크로드 지연 시간 요구 사항도 고려해야 합니다. 예를 들어 지연 시간에 민감한 애플리케이션이 있는 경우 지연 시간이 매우 짧은 Network Load Balancer를 사용하기로 결정할 수 있습니다. 또는 AWS 로컬 영역 또는AWS Outposts에서 Application Load Balancer를 활용하여 워크로드를 고객에게 더 가까이 가져오기로 결정할 수 있습니다.

지연 시간에 민감한 워크로드에 대한 또 다른 대응책은 교차 영역 로드 밸런싱입니다. 교차 영역 로드 밸런싱을 활용하면 각 로드 밸런서 노드를 사용하도록 설정된 모든 가용 영역의 등록된 대상에 트래픽

을 분산합니다. 이렇게 하면 왕복 지연 시간에 한 자릿수 밀리초가 추가될 수 있지만 가용성이 향상됩니다.

마지막으로 ALB와 NLB는 모두 로그 및 지표와 같은 모니터링 리소스를 제공합니다. 올바르게 모니터링을 설정하면 애플리케이션의 성능 인사이트를 수집하는 데 도움이 될 수 있습니다. 예를 들어 ALB 액세스 로그를 사용하면 응답하는 데 더 오래 걸리는 요청이나 성능 문제를 일으키는 백엔드 대상을 찾을 수 있습니다.

구현 단계

1. 워크로드에 적합한 로드 밸런서를 선택합니다.
   a. HTTP/HTTPS 워크로드에 Application Load Balancer를 사용합니다.
   b. TCP 또는 UDP에서 실행되는 비 HTTP 워크로드에 Network Load Balancer를 사용합니다.
   c. 두 제품의 기능을 모두 활용하려면 두 제품의 조합(NLB의 대상인 ALB)을 사용합니다. 예를 들어 ALB의 HTTP 헤더 기반 라우팅과 함께 NLB의 고정 IP를 사용하려는 경우 또는 HTTP 워크로드를 AWS PrivateLink에 노출하려는 경우 이를 수행할 수 있습니다.
   d. 로드 밸런서를 비교한 전체 내용을 보려면 ELB 제품 비교를 참조하세요.
2. SSL/TLS 오프로딩을 사용합니다.
   a. AWS Certificate Manager와 통합된 Application Load Balancer 및 Network Load Balancer로 HTTPS/TLS 리스너를 구성합니다.
   b. 일부 워크로드는 규정 준수상의 이유로 엔드 투 엔드 암호화가 필요할 수 있습니다. 이 경우 대상에서 암호화를 사용하도록 설정해야 합니다.
   c. 보안 모범 사례는 SEC09-BP02 전송 중 데이터 암호화 적용을 참조하세요.
3. 적절한 라우팅 알고리즘을 선택합니다.
   a. 라우팅 알고리즘에 따라 백엔드 대상에서의 활용도 및 성능에 미치는 영향에 차이를 만들 수 있습니다. 예를 들어 ALB는 라우팅 알고리즘에 대해 다음 두 가지 옵션을 제공합니다.
   b. 처리되지 않은 요청 최소화: 애플리케이션에 대한 요청의 복잡성이 다양하거나 대상의 처리 능력이 다양한 경우 백엔드 대상에 로드를 더 효율적으로 분산하기 위해 사용합니다.
   c. 라운드 로빈: 요청과 대상이 유사하거나 대상 간에 요청을 균등하게 분산해야 하는 경우에 사용합니다.
4. 교차 영역 또는 영역 격리를 고려합니다.
   a. 지연 시간 개선 및 영역 장애 도메인을 위해 교차 영역 끔(영역 격리)을 사용합니다. NLB에서는 기본적으로 비활성화되어 있으며 ALB에서는 대상 그룹별로 비활성화할 수 있습니다.
   b. 가용성과 유연성 향상을 위해 교차 영역을 사용합니다. 기본적으로 교차 영역은 ALB에 대해 활성화되어 있으며 NLB에서는 대상 그룹별로 비활성화할 수 있습니다.

5. HTTP 워크로드에 대해 HTTP 연결 유지를 활성화합니다.
    a. HTTP 워크로드의 경우 백엔드 대상에 대한 웹 서버 설정에서 HTTP 연결 유지를 활성화합니다. 이 기능을 사용하면 로드 밸런서는 연결 유지 제한 시간이 만료될 때까지 백엔드 연결을 재사용하여 HTTP 요청 및 응답 시간을 개선하고 백엔드 대상의 리소스 사용률을 줄일 수 있습니다. Apache 및 Nginx에 대해 이 작업을 수행하는 방법에 대한 자세한 내용은 Apache 또는 NGINX를 ELB의 백엔드 서버로 사용하기 위한 최적의 설정은 무엇인가요?를 참조하세요.
6. 컴퓨팅 리소스의 더 나은 오케스트레이션을 위해 Elastic Load Balancing 통합을 사용합니다.
    a. 로드 밸런서와 통합된 Auto Scaling을 사용합니다. 성능 효율적인 시스템의 주요 측면 중 하나는 백엔드 리소스의 크기를 적절하게 조정하는 것과 관련이 있습니다. 이를 위해서는 백엔드 대상 리소스에 대한 로드 밸런서 통합을 활용할 수 있습니다. Auto Scaling 그룹과 로드 밸런서 통합을 사용하면 수신 트래픽에 대한 응답으로 필요에 따라 로드 밸런서에서 대상이 추가되거나 제거됩니다.
    b. 로드 밸런서는 컨테이너화된 워크로드를 위해 Amazon ECS 및 Amazon EKS와 통합할 수도 있습니다.
        - Elastic Load Balancing을 사용하여 Auto Scaling Scaling 그룹의 인스턴스 간에 트래픽 분산
        - Amazon ECS - 서비스 로드 밸런싱
        - Amazon EKS에서 애플리케이션 로드 밸런싱
        - Amazon EKS에서 네트워크 로드 밸런싱
7. 로드 밸런서를 모니터링하여 성능 병목 현상을 찾습니다.
    a. Application Load Balancer 및 Network Load Balancer에 대한 액세스 로그를 활성화합니다.
    b. ALB에 대해 고려해야 할 주요 필드는 `request_processing_time`, `request_processing_time`및 `response_processing_time`입니다.
    c. NLB에 대해 고려해야 할 주요 필드는 `connection_time` 및 `tls_handshake_time`입니다.
    d. 필요할 때 로그를 쿼리할 준비합니다. Amazon Athena를 사용하여 ALB 로그와 NLB 로그를 모두 쿼리할 수 있습니다.
    e. 성능 관련 지표에 대한 경보(예: ALB의`TargetResponseTime`)를 생성합니다.

리소스

관련 모범 사례:

- SEC09-BP02 전송 중 데이터 암호화 적용

관련 문서:

- ELB 제품 비교
- AWS 글로벌 인프라
- 가용 영역 선호도를 사용하여 성능 개선 및 비용 절감
- Amazon Athena를 사용한 단계별 로그 분석
- Application Load Balancer 로그 쿼리
- Application Load Balancer 모니터링
- Network Load Balancer 모니터링

관련 동영상:

- AWS re:Invent 2018: [REPEAT 1] Elastic Load Balancing: Deep Dive and Best Practices (NET404-R1)(AWS re:Invent 2018: [REPEAT 1] Elastic Load Balancing: 심층 분석 및 모범 사례(NET404-R1))
- AWS re:Invent 2021 - How to choose the right load balancer for your AWS workloads(AWS re:Invent 2021 - AWS 워크로드에 적합한 로드 밸런서를 선택하는 방법)
- AWS re:Inforce 2022 - How to use Elastic Load Balancing to enhance your security posture at scale (NIS203)(AWS re:Inforce 2022 - Elastic Load Balancing을 사용하여 대규모로 보안 태세를 강화하는 방법(NIS203))
- AWS re:Invent 2019: Get the most from Elastic Load Balancing for different workloads (NET407-R2)(AWS re:Invent 2019: 다양한 워크로드에 Elastic Load Balancing을 최대한 활용)

관련 예시:

- Amazon Athena를 사용한 로그 분석의 CDK 및 CloudFormation 샘플

PERF05-BP05 성능을 개선할 수 있는 네트워크 프로토콜 선택

워크로드의 성능 요구 사항을 평가하고 워크로드의 전반적인 성능을 최적화하는 네트워크 프로토콜을 선택합니다.

원하는 처리량을 달성하려면 지연 시간과 대역폭 간의 관계를 고려해야 합니다. 예를 들어 파일 전송이 전송 제어 프로토콜(TCP)을 사용하는 경우 지연 시간이 길수록 전체 처리량이 줄어듭니다. 이 문제는 TCP 튜닝 및 최적화된 전송 프로토콜을 사용하여 해결되며 경우에 따라 사용자 데이터그램 프로토콜(UDP)을 사용하기도 합니다.

Scalable Reliable Datagram(SRD) 프로토콜은 신뢰할 수 있는 데이터그램 전달을 제공하는 AWS for Elastic Fabric Adapter에서 구축한 네트워크 전송 프로토콜입니다. TCP 프로토콜과 달리 SRD는 패킷을 재정렬하여 순서와 관계 없이 전달할 수 있습니다. SRD의 순서와 관계 없이 전달 메커니즘은 대체 경로를 통해 패킷을 병렬로 전송하므로 처리량이 증가합니다.

일반적인 안티 패턴:

- 성능 요구 사항과 관계없이 모든 워크로드에 TCP를 사용합니다.

이 모범 사례 확립의 이점:

- 워크로드 구성 요소 간 통신에 적합한 프로토콜을 선택하면 해당 워크로드에 대한 성능을 극대화할 수 있습니다.
- 사용자와 워크로드 구성 요소 간의 통신에 적절한 프로토콜이 사용되는지 확인하면 애플리케이션의 전반적인 사용자 경험을 개선하는 데 도움이 됩니다. 예를 들어, TCP 및 UDP를 함께 사용하면 VDI 워크로드는 중요한 데이터에 대한 TCP의 신뢰성과 실시간 데이터에 대한 UDP의 속도를 활용할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간(부적절한 네트워크 프로토콜을 사용하면 느린 응답 시간, 긴 대기 시간 및 낮은 확장성과 같은 성능 저하가 발생)

구현 가이드

워크로드의 성능을 향상하기 위한 주요 고려 사항은 지연 시간 및 처리량 요구 사항을 이해한 다음 성능을 최적화하는 네트워크 프로토콜을 선택하는 것입니다.

TCP 사용을 고려해야 할 시기

TCP는 신뢰할 수 있는 데이터 전달을 제공하며 신뢰성과 보장된 데이터 전달이 중요한 워크로드 구성 요소 간의 통신에 사용될 수 있습니다. 많은 웹 기반 애플리케이션은 HTTP 및 HTTPS와 같은 TCP 기반 프로토콜을 사용하여 AWS 서버와 통신하기 위한 TCP 소켓을 열어줍니다. 이메일 및 파일 데이터 전송은 데이터 교환 속도 및 네트워크 정체를 제어할 수 있는 TCP의 기능으로 인해 TCP를 사용하는 일반적인 애플리케이션입니다. TCP와 함께 TLS를 사용하면 통신에 약간의 오버헤드가 추가되어 지연 시간이 증가하고 처리량이 줄어들 수 있습니다. 오버헤드는 주로 완료하는 데 여러 번의 왕복이 필요할 수 있는 핸드셰이크 프로세스의 추가 오버헤드에서 발생합니다. 핸드셰이크가 완료되면 데이터 암호화 및 복호화의 오버헤드는 비교적 작습니다.

UDP 사용을 고려해야 할 시기

UDP는 비연결 지향 프로토콜이므로 로그, 모니터링 및 VOIP 데이터와 같이 빠르고 효율적인 전송이 필요한 애플리케이션에 적합합니다. 또한 워크로드의 최적 성능을 보장하기 위해 많은 클라이언트의 작은 쿼리에 응답하는 워크로드 구성 요소가 있는 경우 UDP를 사용하는 것이 좋습니다. Datagram Transport Layer Security(DTLS)는 TLS의 UDP에 해당합니다. UDP와 함께 DTLS를 사용하면 핸드셰이크 프로세스가 간소화되므로 오버헤드가가 데이터를 암호화하고 복호화할 때 발생합니다. DTLS는 또한 보안 파라미터를 나타내고 변조를 감지하기 위한 추가 필드를 포함하기 때문에 UDP 패킷에 소량의 오버헤드를 추가합니다.

SRD 사용을 고려해야 할 시기

Scalable Reliable Datagram(SRD)은 여러 경로에 걸쳐 트래픽을 로드 밸런싱할 수 있어 고처리량 워크로드에 최적화된 네트워크 전송 프로토콜이며 패킷 감소 또는 연결 장애에서 빠르게 복구합니다. 따라서 SRD는 컴퓨팅 노드 간에 처리량이 많고 지연 시간이 짧은 통신이 필요한 고성능 컴퓨팅(HPC) 워크로드에 가장 효과적입니다. 여기에는 노드 간에 대량의 데이터 전송을 수반하는 시뮬레이션, 모델링 및 데이터 분석과 같은 병렬 처리 작업이 포함될 수 있습니다.

구현 단계

1. AWS Global Accelerator 및 AWS Transfer Family 서비스를 사용하여 온라인 파일 전송 애플리케이션의 처리량을 향상합니다. AWS Global Accelerator 서비스를 사용하면 클라이언트 디바이스와 AWS 워크로드 간에 지연 시간을 단축하는 데 도움이 됩니다. AWS Transfer Family를 사용하면 Secure Shell File Transfer Protocol(SFTP) 및 File Transfer Protocol over SSL(FTPS)과 같은 TCP 기반 프로토콜을 사용하여 파일 전송을 AWS 스토리지 서비스로 안전하게 확장하고 관리할 수 있습니다.
2. 네트워크 지연 시간을 사용하여 TCP가 워크로드 구성 요소 간의 통신에 적합한지 확인합니다. 클라이언트 애플리케이션과 서버 간의 네트워크 지연 시간이 길면 TCP 3방향 핸드셰이크에 시간이 걸릴 수 있습니다. 그렇게 되면 애플리케이션의 응답성에 영향을 줄 수 있습니다. 첫 번째 바이트까지의 시간(TTFB) 및 왕복 시간(RTT)과 같은 지표를 사용하여 네트워크 지연 시간을 측정할 수 있습니다. 워크로드가 사용자에게 동적 콘텐츠를 제공하는 경우, 동적 콘텐츠의 각 오리진에 영구 연결을 설정하는 Amazon CloudFront를 사용하면 각 클라이언트 요청을 지연시킬 수 있는 연결 설정 시간을 제거할 수 있습니다.
3. TCP 또는 UDP와 함께 TLS를 사용하면 암호화 및 복호화의 영향으로 인해 지연 시간이 증가하고 워크로드에 대한 처리량이 감소합니다. 이러한 워크로드의 경우 Elastic Load Balancing에서 SSL/TLS 오프로딩을 고려하여 백엔드 인스턴스 대신 로드 밸런서가 SSL/TLS 암호화 및 복호화 프로세스를 처리하도록 하여 워크로드 성능을 개선합니다. 이를 통해 백엔드 인스턴스의 CPU 사용률을 줄여 성능을 향상시키고 용량을 늘릴 수 있습니다.
4. Network Load Balancer(NLB)를 사용하여 인증 및 승인, 로깅, DNS, IoT 및 스트리밍 미디어와 같은 UDP 프로토콜을 이용하는 서비스를 배포하여 워크로드의 성능 및 신뢰성을 향상시킵니다. NLB는

여러 대상에 걸쳐 수신되는 UDP 트래픽을 배포하여 워크로드를 수평 확장하고 용량을 늘리고 단일 대상의 오버헤드를 줄일 수 있습니다.

5. 고성능 컴퓨팅(HPC) 워크로드의 경우 SRD 프로토콜을 사용하는 Elastic Network Adapter(ENA) Express 기능을 사용하여 EC2 인스턴스 간의 네트워크 트래픽에 대해 더 높은 단일 흐름 대역폭(25Gbps)과 더 짧은 꼬리 지연 시간(99.9 백분위수)을 제공하여 네트워크 성능을 개선하는 것이 좋습니다.
6. Application Load Balancer(ALB)를 사용하여 워크로드 구성 요소 간에 또는 gRPC 지원 클라이언트와 서비스 간에 gRPC(원격 프로시저 호출) 트래픽을 라우팅하고 부하를 분산합니다. gRPC는 전송에 TCP 기반 HTTP/2 프로토콜을 사용하며 더 가벼운 네트워크 공간, 압축, 효율적인 이진 직렬화, 다양한 언어 지원, 양방향 스트리밍과 같은 성능상의 이점을 제공합니다.

리소스

관련 문서:

- Amazon EBS - 최적화 인스턴스
- Application Load Balancer
- Linux 기반 EC2 향상된 네트워킹
- Windows의 EC2 향상된 네트워킹
- EC2 배치 그룹
- Linux 인스턴스에서 ENA(Elastic Network Adapter)를 사용하여 향상된 네트워킹 활성화
- Network Load Balancer
- AWS의 네트워킹 제품
- Transit Gateway
- Amazon Route 53에서 지연 시간 기반 라우팅으로 전환
- VPC 엔드포인트
- VPC 흐름 로그

관련 동영상:

- Connectivity to AWS and hybrid AWS network architectures (NET317-R1)(AWS 및 하이브리드 AWS 네트워크 아키텍처에 대한 연결(NET317-R1))
- Optimizing Network Performance for Amazon EC2 Instances(CMP308-R1)(Amazon EC2 인스턴스의 네트워크 성능 최적화(CMP308-R1))

- Tuning Your Cloud: Improve Global Network Performance for Application(클라우드 튜닝: 애플리케이션의 글로벌 네트워크 성능 향상)
- Application Scaling with EFA and SRD(EFA 및 SRD를 통한 애플리케이션 확장)

관련 예시:

- AWS Transit Gateway 및 확장 가능한 보안 솔루션
- AWS 네트워킹 워크숍

PERF05-BP06 네트워크 요구 사항에 따라 워크로드의 위치 선택

리소스 배치 옵션을 평가하여 네트워크 지연 시간을 줄이고 처리량을 향상시켜 페이지 로드 및 데이터 전송 시간을 줄임으로써 최적의 사용자 경험을 제공합니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

Amazon EC2 인스턴스와 같은 리소스는 AWS 리전, AWS 로컬 영역, AWS Outposts 또는 AWS Wavelength 영역에 배치됩니다. 이 위치를 선택하면 주어진 사용자 위치의 네트워크 지연 시간 및 처리량에 영향을 미칩니다. Amazon CloudFront 및 AWS Global Accelerator와 같은 엣지 서비스는 엣지 로케이션의 콘텐츠를 캐싱하거나 AWS 글로벌 네트워크를 통해 워크로드에 대한 최적의 경로를 사용자에게 제공하여 네트워크 성능을 향상시키는 데 사용될 수 있습니다.

구현 단계

1. 다음과 같은 주요 요소를 토대로 하여 배포용으로 적절한 AWS 리전을 하나 이상 선택합니다.
   a. 사용자의 위치: 워크로드 사용자 근처의 리전을 선택하면 사용자가 워크로드를 사용할 때 지연 시간이 단축됩니다.
   b. 데이터의 위치: 데이터를 많이 사용하는 애플리케이션에서는 데이터 전송 시 지연 시간 병목 현상이 가장 많이 발생합니다. 따라서 애플리케이션 코드는 최대한 데이터와 가까운 위치에서 실행되어야 합니다.
   c. 기타 제약 조건: 보안 및 규정 준수 등의 제약(예: 데이터 상주 요구 사항)을 고려해야 합니다.
2. 주어진 워크로드의 경우, 구성 요소가 짧은 지연 시간이 필요한 상호 의존적인 Amazon EC2 인스턴스 그룹으로 구성되었다면 클러스터 배치 그룹을 사용하여 워크로드의 요구 사항을 충족할 수 있도록 해당 인스턴스의 배치에 영향을 주는 것이 좋습니다. 동일한 클러스터 배치 그룹의 인스턴스는 TCP/IP 트래픽에 더 높은 흐름당 처리량 제한을 제공하며 네트워크의 동일한 높은 이분 대역폭 세

그먼트에 배치됩니다. 네트워크 지연 시간이 짧거나 처리량이 높은 경우 또는 두 조건을 모두 충족하는 경우 성능이 개선되는 애플리케이션에는 클러스터 배치 그룹을 사용하는 것이 좋습니다.

3. 예를 들어 짧은 지연 시간이나 데이터 상주 요구 사항으로 인해 위치에 민감한 워크로드의 경우 AWS 로컬 영역 또는 AWS Outposts를 검토합니다.
   a. AWS 로컬 영역은 컴퓨팅, 스토리지, 데이터베이스 및 기타 선별된 AWS 서비스를 인구가 많은 산업 센터와 가까운 곳에 배치하는 인프라 배포 유형입니다.
   b. AWS Outposts는 AWS 인프라와 서비스를 거의 모든 온프레미스 또는 엣지 로케이션에 제공하여 진정으로 일관된 하이브리드 경험을 구현하는 완전관리형 솔루션 제품군입니다.
4. 고해상도 라이브 비디오 스트리밍, 고음질 오디오 및 증강 현실/가상 현실(AR/VR)과 같은 5G 디바이스용 애플리케이션은 지연 시간이 매우 짧아야 합니다. 이러한 애플리케이션의 경우 AWS Wavelength을 고려합니다. AWS Wavelength는 5G 네트워크 내에 AWS 컴퓨팅 및 스토리지 서비스를 포함하여 지연 시간이 매우 짧은 애플리케이션을 개발, 배포 및 확장하기 위한 모바일 엣지 컴퓨팅 인프라를 제공합니다.
5. 지리적으로 분산된 사용자가 있는 경우 콘텐츠 배포 네트워크(CDN)를 사용하여 전 세계적으로 분산된 접속 지점(POP)을 통해 데이터를 전달함으로써 정적 및 동적 웹 콘텐츠를 빠르게 배포할 수 있습니다. CDN은 일반적으로 엣지 컴퓨팅 기능을 제공하며 HTTP 헤더 조작 및 URL 재작성 및 엣지에서 대규모로 리디렉션과 같은 지연 시간에 민감한 작업을 수행합니다. Amazon CloudFront는 정적 및 동적 웹 콘텐츠의 배포 속도를 높이는 웹 서비스입니다. CloudFront의 사용 사례에는 정적 웹 사이트 콘텐츠 전송 가속화 및 온디맨드 비디오 제공 또는 라이브 스트리밍 비디오가 포함됩니다. CloudFront를 통해 지연 시간을 줄이면서 시청자의 콘텐츠와 경험을 사용자 지정할 수도 있습니다.
6. 일부 애플리케이션은 첫 번째 바이트까지의 지연 시간과 지터를 줄이고 처리량을 늘려 고정 진입 지점 또는 그 이상의 성능이 필요합니다. 이러한 애플리케이션은 정적 애니캐스트 IP 주소와 엣지 로케이션에서 TCP 종료를 제공하는 네트워킹 서비스의 혜택을 누릴 수 있습니다. AWS Global Accelerator은 애플리케이션의 성능을 최대 60%까지 향상시키고 다중 리전 아키텍처에 빠른 장애 조치를 제공합니다. AWS Global Accelerator는 하나 이상의 AWS 리전에서 호스팅되는 애플리케이션의 고정 진입 지점으로 사용되는 정적 애니캐스트 IP 주소를 제공합니다. 이 IP 주소를 사용하면 가능한 한 사용자와 가까운 AWS 글로벌 네트워크으로 트래픽이 유입될 수 있습니다. AWS Global Accelerator는 클라이언트와 클라이언트와 가장 가까운 AWS 엣지 로케이션 간에 TCP 연결을 설정하여 초기 연결 설정 시간을 줄입니다. AWS Global Accelerator를 검토하여 TCP/UDP 워크로드의 성능을 향상시키고 다중 리전 아키텍처에 빠른 장애 조치를 제공합니다.
7. 애플리케이션 또는 사용자가 온프레미스 상태인 경우 네트워크와 클라우드 간에 전용 네트워크 연결을 사용하면 혜택을 볼 수 있습니다. 전용 네트워크 연결은 혼잡이 발생하거나 예기치 않은 지연 시간이 증가할 가능성을 줄일 수 있습니다. AWS Direct Connect는 네트워크를 AWS에 직접 연결하고 퍼블릭 인터넷을 우회하여 애플리케이션 성능을 향상시킬 수 있습니다. 새 연결을 만들 때 AWS Direct Connect 제공 파트너가 제공하는 호스팅된 연결을 선택하거나 AWS에서 전용 연결을 선택하

고 전 세계 100개 이상의 AWS Direct Connect 위치에 배포할 수 있습니다. 또한 AWS에서 낮은 데이터 전송 속도로 네트워킹 비용을 줄이고 선택적으로 장애 조치를 위해 Site-to-Site VPN을 구성할 수 있습니다.

8. AWS 내의 리소스에 연결하기 위해 Site-to-Site VPN을 구성하면 필요에 따라 가속화할 수 있습니다. 가속화된 Site-to-Site VPN 연결은 AWS Global Accelerator를 사용하여 온프레미스 네트워크에서 고객 게이트웨이 디바이스에 가장 가까운 AWS 엣지 로케이션으로 트래픽을 라우팅합니다.
9. 워크로드 트래픽 및 사용자 위치를 검토하여 워크로드 성능을 최적화할 DNS 라우팅 옵션을 식별합니다. Amazon Route 53은 지연 시간 기반 라우팅, 지리적 위치 라우팅, 지리 근접 라우팅 및 IP 기반 라우팅 옵션을 제공하여 전 세계 연결 대상을 위한 워크로드 성능을 향상시키는 데 도움이 됩니다.
    a. Route 53은 또한 최종 사용자에게 짧은 쿼리 지연 시간을 제공합니다. Route 53은 전 세계의 DNS 서버의 글로벌 애니캐스트 네트워크를 사용하여 네트워크 조건에 따라 최적의 위치에서 쿼리에 자동으로 응답하도록 설계되었습니다.

리소스

관련 모범 사례:

- COST07-BP02 비용을 기준으로 리전 구현
- COST08-BP03 데이터 전송 비용을 줄이기 위한 서비스 구현
- REL10-BP01 워크로드를 여러 위치에 배포
- REL10-BP02 다중 위치 배포에 적합한 위치 선택
- SUS01-BP01 Amazon 재생 에너지 프로젝트 근처의 리전 및 그리드의 탄소 집약도가 다른 위치(또는 리전)보다 낮은 리전 선택
- SUS02-BP04 사용자 위치에 맞게 워크로드의 지리적 배치 최적화
- SUS04-BP07 네트워크 간 데이터 이동 최소화

관련 문서:

- AWS 글로벌 인프라
- AWS Local Zones and AWS Outposts, choosing the right technology for your edge workload(AWS 로컬 영역 및 AWS Outposts, 엣지 워크로드에 적합한 기술 선택)
- 배치 그룹
- AWS 로컬 영역
- AWS Outposts

- AWS Wavelength
- Amazon CloudFront
- AWS Global Accelerator
- AWS Direct Connect
- Site-to-Site VPN
- Amazon Route 53

관련 동영상:

- AWS Local Zones Explainer Video(AWS 로컬 영역 설명 비디오)
- AWS Outposts: Overview and How It Works(AWS Outposts: 개요 및 작동 방식)
- AWS re:Invent 2021 - AWS Outposts: Bringing the AWS experience on premises(AWS re:Invent 2021 - AWS Outposts: 온프레미스로 AWS 경험 가져오기)
- AWS re:Invent 2020: AWS Wavelength: Run apps with ultra-low latency at 5G edge(AWS re:Invent 2020: AWS Wavelength: 5G 엣지에서 매우 짧은 지연 시간으로 앱 실행)
- AWS re:Invent 2022 - AWS Local Zones: Building applications for a distributed edge(AWS re:Invent 2022 - AWS 로컬 영역: 분산된 엣지를 위한 애플리케이션 구축)
- AWS re:Invent 2021 - Building low-latency websites with Amazon CloudFront(AWS re:Invent 2021 - Amazon Cloudfront를 사용하여 지연 시간이 짧은 웹 사이트 구축)
- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator(AWS re:Invent 2022 - AWS Global Accelerator로 성능 및 가용성 향상)
- AWS re:Invent 2022 - Build your global wide area network using AWS(AWS re:Invent 2022 - AWS를 사용하여 글로벌 광역 네트워크 구축)
- AWS re:Invent 2020: Global traffic management with Amazon Route 53(AWS re:Invent 2020: Amazon Route 53을 사용한 글로벌 트래픽 관리)

관련 예시:

- AWS Global Accelerator 워크숍
- 에지 함수를 사용하여 다시 작성 및 리디렉션 처리

PERF05-BP07 지표를 기준으로 네트워크 구성 최적화

부적절하게 네트워크를 구성하면 네트워크 성능, 효율성 및 비용에 영향을 미치는 경우가 많습니다. 일반적인 네트워크 환경에서는 초기 단계에서 배포를 신속하게 완료하기 위해 네트워크 성능 측면의 적절한 네트워크 구성을 제대로 고려하지 않습니다. 네트워크 구성을 최적화하려면 먼저 네트워크 환경에 대한 가시성과 데이터가 있어야 합니다.

네트워크 리소스의 수행 방식을 이해하려면 데이터를 수집 및 분석하여 정보를 바탕으로 네트워크 구성 최적화 관련 결정을 내립니다. 이러한 변경의 영향을 측정한 다음 영향 측정값을 활용해 향후 결정을 내립니다.

원하는 결과: 지표 및 네트워크 모니터링 도구를 사용하여 워크로드의 변화에 따라 네트워크 구성을 최적화합니다. 클라우드 기반 네트워크는 빠르게 최적화할 수 있으므로 성능 효율성을 유지하려면 네트워크 아키텍처를 지속적으로 변경해야 합니다.

일반적인 안티 패턴:

- 모든 성능 관련 문제가 애플리케이션 관련 문제라고 가정합니다.
- 워크로드를 배포한 위치와 가까운 위치에서만 네트워크 성능을 테스트합니다.
- 모든 네트워크 서비스에 기본 구성을 사용합니다.
- 충분한 용량을 제공하려고 네트워크 리소스를 과도하게 프로비저닝합니다.

이 모범 사례 확립의 이점: AWS 네트워크와 관련된 필수 지표를 수집하고 네트워크 모니터링 도구 구현을 통해 네트워크 성능을 이해하고 네트워크 구성을 최적화할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

VPC, 서브넷 또는 네트워크 인터페이스를 오가는 트래픽을 모니터링하는 것은 AWS 네트워크 리소스를 활용하는 방법과 네트워크 구성을 최적화하는 방법을 이해하는 데 중요합니다. 다음 도구를 사용하면 트래픽 사용, 네트워크 액세스 및 로그에 대한 정보를 추가로 검사할 수 있습니다.

구현 단계

1. Amazon VPC IP Address Manager를 사용합니다. IPAM을 사용하여 AWS 및 온프레미스 워크로드의 IP 주소를 계획, 추적 및 모니터링할 수 있습니다. 이는 IP 주소 사용 및 할당을 최적화하는 가장 좋은 방법입니다.
2. VPC 흐름 로그를 켭니다. VPC 흐름 로그를 사용하여 VPC의 네트워크 인터페이스에서 전송되고 수신되는 IP 트래픽에 대한 정보를 캡처합니다. VPC 흐름 로그를 사용하면 지나치게 제한적이거나 허

용되는 보안 그룹 규칙을 진단하고 네트워크 인터페이스와의 트래픽 방향을 결정할 수 있습니다. 흐름 로그를 게시할 때 Vended 로그에 대한 데이터 수집 및 아카이빙 요금이 적용됩니다.

3. DNS 쿼리 로깅을 켭니다. Route 53이 수신하는 퍼블릭 또는 프라이빗 DNS 쿼리에 대한 정보를 기록하도록 Amazon Route 53을 구성할 수 있습니다. DNS 로그를 사용하면 요청된 도메인 또는 하위 도메인 또는 DNS 쿼리에 응답한 Route 53 엣지 로케이션을 이해하여 DNS 구성을 최적화할 수 있습니다.
4. Reachability Analyzer를 사용하여 네트워크 도달 가능성을 분석하고 디버깅합니다. Reachability Analyzer는 VPC의 소스 리소스와 대상 리소스 간의 연결을 테스트할 수 있는 구성 분석 도구입니다. 이 도구를 사용하면 네트워크 구성이 의도한 연결과 일치하는지 확인하는 데 도움이 됩니다.
5. Network Access Analyzer를 사용하여 리소스에 대한 네트워크 액세스를 파악합니다. Network Access Analyzer를 사용하면 네트워크 액세스 요구 사항을 지정하고 지정된 요구 사항을 충족하지 않는 네트워크 경로를 식별할 수 있습니다. 해당 네트워크 구성을 최적화하면 네트워크 상태를 이해하고 확인하며 AWS의 네트워크가 규정 준수 요구 사항을 충족하는지 입증할 수 있습니다.
6. Amazon CloudWatch를 사용하고 네트워크 옵션에 적절한 지표를 활성화합니다. 워크로드에 적합한 네트워크 지표를 선택해야 합니다. 예를 들어 VPC 네트워크 주소 사용량, VPC NAT 게이트웨이, AWS Transit Gateway, VPN 터널, AWS Network Firewall, Elastic Load Balancing 및 AWS Direct Connect에 대한 지표를 활성화할 수 있습니다. 지표를 지속적으로 모니터링하는 것은 네트워크 상태와 사용량을 관찰하고 파악하는 좋은 방법이며 관찰을 기반으로 네트워크 구성을 최적화하는 데 도움이 됩니다.

구현 계획의 작업 수준: 중간

리소스

관련 문서:

- VPC 흐름 로그
- 퍼블릭 DNS 쿼리 로깅
- IPAM이란 무엇입니까?
- Reachability Analyzer란 무엇입니까?
- What is Network Access Analyzer?(Network Access Analyzer란 무엇입니까?)
- CloudWatch metrics for your VPCs(VPC의 CloudWatch 지표)
- Optimize performance and reduce costs for network analytics with VPC Flow Logs in Apache Parquet format(Apache Parquet 형식의 VPC 흐름 로그를 사용하여 네트워크 분석을 위한 성능 최적화 및 비용 절감)

- Monitoring your global and core networks with Amazon Cloudwatch metrics(Amazon Cloudwatch 지표를 사용하여 글로벌 및 핵심 네트워크 모니터링)
- Continuously monitor network traffic and resources(지속적인 네트워크 트래픽 및 리소스 모니터링)

관련 동영상:

- Networking best practices and tips with the Well-Architected Framework(Well-Architected Framework의 네트워킹 모범 사례 및 팁)
- Monitoring and troubleshooting network traffic(네트워크 트래픽 모니터링 및 문제 해결)

관련 예시:

- AWS 네트워킹 워크숍
- AWS 네트워크 모니터링

검토(Review)

질문

- PERF 6 새 릴리스를 활용하기 위해 워크로드를 어떻게 발전시킵니까?

PERF 6 새 릴리스를 활용하기 위해 워크로드를 어떻게 발전시킵니까?

워크로드 설계 시 선택할 수 있는 옵션은 한정되어 있습니다. 그러나 시간이 지나면 워크로드의 성능을 개선할 수 있는 새로운 기술과 접근 방식을 사용할 수 있게 됩니다.

모범 사례

- PERF06-BP01 새 리소스 및 서비스에 대한 최신 정보 숙지
- PERF06-BP02 워크로드 성능 개선을 위한 프로세스 정의
- PERF06-BP03 장기적인 워크로드 성능 개선

PERF06-BP01 새 리소스 및 서비스에 대한 최신 정보 숙지

새로운 서비스, 설계 패턴 및 제품 오퍼링이 제공되면 성능을 개선할 방법을 평가합니다. 평가, 내부 논의 또는 외부 분석을 통해 워크로드의 효율성을 높이고 성능을 개선할 수 있는 방법을 결정합니다.

워크로드 관련 업데이트, 새 기능 및 서비스를 평가하는 프로세스를 정의합니다. 새 기술을 사용하는 개념 증명 작성, 내부 그룹과의 상담 등을 그 예로 들 수 있습니다. 새 아이디어나 서비스를 시도할 때는 성능 테스트를 실행하여 해당 아이디어나 서비스가 워크로드의 성능에 주는 영향을 측정합니다. 코드형 인프라(IaC) 및 DevOps 문화를 사용함으로써 비용 또는 위험을 최소화하면서 역량을 활용하여 새 아이디어나 기술을 자주 테스트할 수 있습니다.

원하는 결과: 구성 요소, 설계 패턴, 워크로드 특성의 인벤토리를 문서화했습니다. 이러한 문서를 사용하여 서비스 업데이트, 기능 및 신제품을 팀에게 알리는 구독 목록을 만들 수 있습니다. 새 릴리스를 평가하고 비즈니스에 미치는 영향과 우선 순위에 대한 권장 사항을 제공할 구성 요소 이해 관계자를 파악했습니다.

일반적인 안티 패턴:

- 워크로드가 성능 요구 사항을 충족하지 못할 때만 새 옵션 및 서비스를 검토합니다.
- 모든 신제품 오퍼링이 워크로드에 유용하지 않을 것이라고 가정합니다.
- 워크로드를 개선할 때 항상 구매가 아닌 구축을 선택합니다.

이 모범 사례 확립의 이점: 새로운 서비스나 제품 오퍼링을 고려함으로써, 워크로드의 성능과 효율성을 향상하고, 인프라의 비용을 줄이며 서비스 유지 관리에 필요한 노력을 절감할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

AWS의 업데이트, 새 기능 및 서비스를 평가하는 프로세스를 정의합니다. 예를 들어, 새로운 기술을 사용하는 개념 증명을 작성할 수 있습니다. 새 아이디어나 서비스를 시도할 때는 성능 테스트를 실행하여 해당 아이디어나 서비스가 워크로드의 효율성 또는 성능에 미치는 영향을 측정합니다. AWS에서 제공되는 유연한 기능을 활용하면 비용이나 위험을 최소화하면서 새 아이디어나 기술을 자주 테스트할 수 있습니다.

구현 단계

1. 워크로드 솔루션을 문서화합니다. 구성 관리 데이터베이스(CMDB) 솔루션을 사용하여 인벤토리를 문서화하고 서비스 및 종속성을 분류합니다. 도구, 즉 AWS Config 등을 사용하여 워크로드에서 사용되는 AWS의 모든 서비스 목록을 확보합니다.
2. 각 워크로드 구성 요소 및 범주의 소유자를 문서화하는 태깅 전략 을 사용합니다. 예를 들어, 현재 Amazon RDS를 데이터베이스 솔루션으로 사용 중인 경우, 데이터베이스 관리자(DBA)가 새로운 서비스 및 업데이트를 평가하고 연구하기 위해 소유자로 지정되고 문서화되도록 합니다.

3. 워크로드 구성 요소 관련 뉴스 및 업데이트 소스를 파악합니다. 이전에 언급한 Amazon RDS 예시에서 범주 소유자는 해당 워크로드 구성 요소와 일치하는 제품에 대한 '새로운 소식' AWS 블로그 를 구독해야 합니다. RSS 피드를 구독하거나 이메일 구독을 관리할 수 있습니다. 사용하는 Amazon RDS 데이터베이스, 소개된 기능, 릴리스된 인스턴스 및 Amazon Aurora Serverless와 같은 신제품에 대한 업그레이드를 모니터링합니다. 구성 요소에 사용되는 업계 블로그, 제품 및 공급업체를 모니터링합니다.
4. 업데이트 및 새로운 서비스를 평가하기 위한 프로세스를 문서화합니다. 업데이트 및 새로운 서비스를 연구, 테스트, 실험 및 검증하는 데 필요한 시간과 공간을 범주 소유자에게 제공합니다. 문서화된 비즈니스 요구 사항 및 KPI를 다시 참조하여 비즈니스에 긍정적인 영향을 줄 업데이트에 대한 우선순위를 지정할 수 있습니다.

구현 계획의 작업 수준: 이 모범 사례를 확립하려면 현재의 워크로드 구성 요소를 인지하고, 범주 소유자를 파악하고 서비스 업데이트의 소스를 파악해야 합니다. 시작하기 위한 작업 수준은 낮지만 시간이 지남에 따라 발전되고 개선할 수 있는 지속적인 프로세스입니다.

리소스

관련 문서:

- AWS 블로그
- AWS의 새로운 소식

관련 동영상:

- AWS Events YouTube 채널
- AWS Online Tech Talks YouTube 채널
- Amazon Web Services YouTube 채널

관련 예시:

- AWS Github
- AWS Skill Builder

PERF06-BP02 워크로드 성능 개선을 위한 프로세스 정의

새 서비스, 설계 패턴, 리소스 유형 및 구성을 사용할 수 있게 되면 평가를 위한 프로세스를 정의해야 합니다. 예를 들어 새로운 인스턴스 오퍼링에서 기존 성능 테스트를 실행하여 워크로드 개선 가능성을 결정합니다.

워크로드의 성능에는 몇 가지 주요 제약 사항이 있습니다. 워크로드의 성능을 향상시킬 수 있는 혁신이 어떤 것인지 파악할 수 있도록 이러한 내용을 문서화합니다. 새 서비스나 기술을 사용할 수 있게 되면 해당 서비스/기술을 습득할 때 이 정보를 사용하여 제약 조건이나 병목 현상을 완화하는 방법을 파악합니다.

일반적인 안티 패턴:

- 시간이 지나면 현재 아키텍처가 정적 아키텍처가 되고 절대 업데이트되지 않는다고 가정합니다.
- 시간이 지나면 타당한 지표 없이 아키텍처 변경을 도입합니다.

이 모범 사례 수립의 이점: 아키텍처 변경을 위한 프로세스를 정의하면 수집된 데이터를 사용하여 워크로드 설계에 지속적으로 영향을 줄 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

워크로드에 대한 주요 성능 제약 식별: 워크로드의 성능 제약을 문서화하면 어떤 종류의 혁신이 워크로드의 성능을 개선할 수 있는지 알 수 있습니다.

리소스

관련 문서:

- AWS 블로그
- AWS의 새로운 소식

관련 동영상:

- AWS Events YouTube 채널
- AWS Online Tech Talks YouTube 채널
- Amazon Web Services YouTube 채널

관련 예시:

- AWS Github
- AWS Skill Builder

PERF06-BP03 장기적인 워크로드 성능 개선

조직에서 평가 프로세스를 통해 수집된 정보를 사용하여 제공되는 새 서비스나 리소스를 적극적으로 도입합니다.

새 서비스나 기술을 평가할 때 수집한 정보를 사용하여 변경을 진행합니다. 비즈니스 또는 워크로드가 변경되면 성능도 변경되어야 합니다. 워크로드 지표에서 수집한 데이터를 사용해 효율성이나 성능을 가장 많이 개선할 수 있는 영역을 평가하고, 수요를 충족할 수 있도록 새 서비스와 기술을 사전에 도입합니다.

일반적인 안티 패턴:

- 시간이 지나면 현재 아키텍처가 정적 아키텍처가 되고 절대 업데이트되지 않는다고 가정합니다.
- 시간이 지나면 타당한 지표 없이 아키텍처 변경을 도입합니다.
- 업계의 모두가 사용하고 있다는 이유 때문에 아키텍처를 변경합니다.

이 모범 사례 수립의 이점: 워크로드 성능과 비용을 최적화하려면 사용 가능한 모든 소프트웨어와 서비스를 평가하여 워크로드에 적합한 소프트웨어와 서비스를 결정해야 합니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

시간 경과에 따른 워크로드 변경: 새 서비스나 기술을 평가할 때 수집한 정보를 사용하여 변경을 진행합니다. 비즈니스 또는 워크로드가 변경되면 성능도 변경되어야 합니다. 워크로드 지표에서 수집한 데이터를 사용하여 효율성 또는 성능을 가장 많이 개선할 수 있는 영역을 평가하고, 새 서비스와 기술을 미리 도입하여 요구 사항에 부응합니다.

리소스

관련 문서:

- AWS 블로그
- AWS의 새로운 소식

관련 동영상:

- AWS Events YouTube 채널
- AWS Online Tech Talks YouTube 채널
- Amazon Web Services YouTube 채널

관련 예시:

- AWS Github
- AWS Skill Builder

모니터링

질문

- PERF 7 리소스 성능을 모니터링하려면 어떻게 해야 합니까?

PERF 7 리소스 성능을 모니터링하려면 어떻게 해야 합니까?

시스템 성능은 시간이 지남에 따라 저하될 수 있습니다. 시스템 성능을 모니터링하여 성능 저하 상태를 식별하고 운영 체제 또는 애플리케이션 로드와 같은 내부 또는 외부 요인을 해결합니다.

모범 사례

- PERF07-BP01 성능 관련 지표 기록
- PERF07-BP02 이벤트 또는 인시던트 발생 시의 지표 분석
- PERF07-BP03 워크로드 성능을 측정하는 핵심 성능 지표(KPI) 설정
- PERF07-BP04 모니터링을 사용하여 경보 기반 알림 생성
- PERF07-BP05 정기적인 간격으로 지표 검토
- PERF07-BP06 사전 모니터링 및 경보 생성

PERF07-BP01 성능 관련 지표 기록

모니터링 및 관찰 서비스를 사용하여 성능 관련 지표를 기록합니다. 지표의 예로는 레코드 데이터베이스 트랜잭션, 속도가 느린 쿼리, I/O 지연 시간, HTTP 요청 처리량(throughput), 서비스 지연 시간 또는 기타 주요 데이터가 있습니다.

워크로드에 중요한 성능 지표를 확인하여 기록합니다. 워크로드의 전반적인 성능이나 효율성에 영향을 미치는 구성 요소를 파악하려면 이 데이터가 필요합니다.

고객 경험을 바탕으로 중요한 지표를 식별하십시오. 각 지표에 대해 목표, 측정 방식 및 우선 순위를 정합니다. 이러한 지표를 사용하여 성능 관련 문제를 사전에 해결할 수 있도록 경보와 알림을 작성합니다.

일반적인 안티 패턴:

- 운영 체제 수준 지표만 모니터링하여 워크로드에 대한 인사이트를 얻습니다.
- 피크 워크로드 요구 사항에 따라 컴퓨팅 요구 사항을 설계합니다.

이 모범 사례 정립의 이점: 성능 및 리소스 사용률을 최적화하려면 주요 성능 지표에 대한 통합된 운영 보기가 필요합니다. 대시보드를 생성하고 데이터에 대한 지표 산술을 수행하여 운영 및 사용률에 대한 인사이트를 도출할 수 있습니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

워크로드에 관련된 성능 지표를 식별하고 기록합니다. 이 데이터는 워크로드의 전체 성능 또는 효율성에 영향을 미치는 구성 요소를 식별하는 데 도움이 됩니다.

성능 지표 식별: 고객 경험을 바탕으로 가장 중요한 지표를 식별합니다. 각 지표에 대해 목표, 측정 방식 및 우선 순위를 정합니다. 이러한 데이터 포인트를 사용하여 성능 관련 문제를 사전에 해결하기 위한 경보와 알림을 작성합니다.

리소스

관련 문서:

- CloudWatch 설명서
- CloudWatch 에이전트를 사용하여 Amazon EC2 인스턴스 및 온프레미스 서버에서 지표 및 로그 수집
- 사용자 지정 지표 게시
- 모니터링, 로깅 및 성능 APN 파트너
- X-Ray 설명서
- Amazon CloudWatch RUM

관련 동영상:

- Cut through the chaos: Gain operational visibility and insight(MGT301-R1)
- AWS의 애플리케이션 성능 관리
- 모니터링 플랜 세우기

관련 예시:

- 레벨 100: CloudWatch 대시보드를 통한 모니터링
- 레벨 100: CloudWatch 대시보드를 통한 Windows EC2 인스턴스 모니터링
- 레벨 100: CloudWatch 대시보드를 통한 Amazon Linux EC2 인스턴스 모니터링

PERF07-BP02 이벤트 또는 인시던트 발생 시의 지표 분석

이벤트나 인시던트에 대응하는 과정에서 모니터링 대시보드나 보고서를 사용해 이벤트/인시던트의 영향을 파악하고 진단합니다. 이러한 대시보드나 보고서에서는 예상 성능을 제공하지 못하는 워크로드의 부분을 파악할 수 있습니다.

아키텍처에 중요한 사용자 사례를 작성할 때는 중요한 각 사례에 필요한 실행 속도를 지정하는 등의 성능 요구 사항을 포함합니다. 이러한 중요 사례의 경우 스크립트로 작성된 사용자 여정을 추가로 구현하여 해당 사례의 성능이 요구 사항에 부합하는지 확인합니다.

일반적인 안티 패턴:

- 성능 이벤트는 한 번 발생하고 마는 문제이며, 이상 징후와 관련된 것일 뿐이라고 가정합니다.
- 성능 이벤트에 대응할 때 기존 성능 지표만 평가합니다.

이 모범 사례 수립의 이점: 워크로드가 예상 수준에서 작동하는지 확인하려면 분석에 사용할 추가 지표 데이터를 수집하여 성능 이벤트에 대응해야 합니다. 이 데이터는 성능 이벤트의 영향을 이해하고 워크로드 성능을 개선하기 위한 변경을 제안하는 데 사용됩니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

중요한 사용자 사례에 대한 경험 문제를 우선적으로 처리: 아키텍처에 중요한 사용자 사례를 작성할 때는 중요한 각 사례에 필요한 실행 속도를 지정하는 등의 성능 요구 사항을 포함합니다. 이러한 중요 사

례의 경우 스크립트로 작성된 사용자 여정을 구현하여 이러한 사례의 성능이 요구 사항에 부합하는지 확인합니다.

리소스

관련 문서:

- CloudWatch 설명서
- Amazon CloudWatch Synthetics
- 모니터링, 로깅 및 성능 APN 파트너
- X-Ray 설명서

관련 동영상:

- Cut through the chaos: Gain operational visibility and insight(MGT301-R1)
- Amazon CloudWatch RUM을 통한 애플리케이션 최적화
- Amazon CloudWatch Synthetics 데모

관련 예시:

- Amazon CloudWatch Synthetics를 활용한 페이지 로드 시간 측정
- Amazon CloudWatch RUM 웹 클라이언트

PERF07-BP03 워크로드 성능을 측정하는 핵심 성능 지표(KPI) 설정

워크로드 성능을 양적 및 질적으로 측정하는 KPI를 식별하십시오. KPI는 비즈니스 목표와 관련된 워크로드의 상태를 측정하는 데 도움이 됩니다. KPI를 통해 비즈니스 및 엔지니어링 팀은 목표 측정값을 전략에 맞추고, 이를 조합하여 비즈니스 성과를 도출하는 방법을 파악할 수 있습니다. 비즈니스 목표, 전략 또는 최종 사용자 요구 사항이 변경되면 KPI를 다시 검토해야 합니다.

예를 들어, 웹 사이트 워크로드에는 전체 성능을 나타내는 지표로 페이지 로드 시간을 사용할 수 있습니다. 이 지표는 최종 사용자 경험을 측정하는 여러 데이터 포인트 중 하나입니다. 페이지 로드 시간 임계값을 파악하는 것 말고도 성능이 충족되지 않을 경우 예상되는 결과나 비즈니스 위험도 문서화해야 합니다. 페이지 로드 시간이 길면 최종 사용자에게 직접적인 영향을 주고, 사용자 경험 수준이 떨어져 고객이 이탈하는 결과가 발생할 수 있습니다. KPI 임계값을 정의할 때는 업계 벤치마크와 최종 사용자 기대치를 모두 고려해야 합니다. 가령 현재 업계 벤치마크에 따르면 웹 페이지를 2초 안에 로드하면 되지만, 최종 사용자는 웹 페이지가 1초 안에 로드될 것으로 기대한다면 이러한 데이터 포인트를 전부 고

려해서 KPI를 설정해야 합니다. KPI의 또 다른 예는 내부 성능 요구 사항을 충족하는 데 초점을 맞출 수 있습니다. 프로덕션 데이터가 생성된 후 1영업일 안에 영업 보고서를 작성할 때 KPI 임계값이 설정될 수 있습니다. 이러한 보고서는 매일의 의사 결정과 비즈니스 성과에 직접적인 영향을 미칠 수 있습니다.

원하는 결과: 다양한 부서와 이해관계자가 참여하여 KPI를 수립합니다. 팀은 참조용으로 실시간 세분화된 데이터와 기록 데이터를 사용하여 워크로드 KPI를 평가하고, KPI 데이터에 대한 지표 산술을 수행하여 운영 및 활용률 인사이트를 도출하는 대시보드를 만들어야 합니다. 합의된 KPI 및 임계값을 설명하는 KPI를 문서화해야 합니다. 이러한 KPI와 임계값은 모니터링되는 지표에 매핑되어 비즈니스 목표와 전략을 지원합니다. KPI는 성능 요구 사항을 파악하는 데 활용할 수 있으며, 이를 의도적으로 검토하고 모든 팀과 자주 공유해서 정보를 파악해야 합니다. 위험과 절충안을 명확하게 식별하고 KPI 임계값이 충족되지 않으면 비즈니스에 어떤 영향이 있는지 이해해야 합니다.

일반적인 안티 패턴:

- 시스템 수준 지표를 모니터링하여 워크로드에 대한 인사이트를 얻고, 해당 지표에 대한 비즈니스 영향을 이해하지 못합니다.
- KPI가 이미 표준 지표 데이터로 게시 및 공유되고 있다고 가정합니다.
- KPI를 정의하지만, 모든 팀과 공유하지는 않습니다.
- 정량화되어 측정 가능한 KPI를 정의하지 않습니다.
- KPI를 비즈니스 목표나 전략에 맞추지 않습니다.

이 모범 사례 정립의 이점: 워크로드 상태를 나타내는 특정 지표를 식별하면 우선순위에 따라 팀을 조율하고 성공적인 비즈니스 성과를 정의할 수 있습니다. 이러한 지표를 모든 부서와 공유하면 임계값, 기대치 및 비즈니스에 미치는 영향을 파악하고, 이에 따른 조정이 가능해집니다.

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

워크로드의 상태에 영향을 받는 모든 부서 및 비즈니스 팀은 KPI를 정의하는 데 기여해야 합니다. 한 사람이 조직의 KPI와 관련된 협업을 주도하고 일정, 문서, 정보를 주관해야 합니다. 이 단일 스레드 소유자는 비즈니스 목표와 전략을 공유하고 각 부서에서 KPI를 생성하는 비즈니스 이해관계자에게 작업을 할당하는 경우가 많습니다. KPI가 정의되면 운영 팀은 주로 다양한 KPI의 성공을 지원하고, 이를 알리는 지표를 정의하는 데 도움을 줍니다. KPI는 워크로드를 지원하는 모든 팀원이 KPI를 알고 있는 경우에만 효과가 있습니다.

구현 단계

1. 비즈니스 이해관계자를 식별하고 문서화합니다.
2. 회사의 목표와 전략을 파악합니다.
3. 회사의 목표와 전략에 부합하는 일반적인 업계 KPI를 검토합니다.
4. 워크로드에 대한 최종 사용자의 기대치를 검토합니다.
5. 회사의 목표와 전략을 지원하는 KPI를 정의하고 문서화합니다.
6. KPI를 충족하는 데 도움이 되는 승인 절충 전략을 파악하고 문서화합니다.
7. KPI에 정보를 제공할 지표를 식별하고 문서화합니다.
8. 심각도 또는 경보 수준에 대한 KPI 임계값을 식별하고 문서화합니다.
9. KPI가 충족되지 않을 경우의 위험과 영향을 파악하고 문서화합니다.
10. KPI당 검토 빈도를 파악합니다.
11. 워크로드를 지원하는 모든 팀과 KPI 문서를 공유합니다.

구현 지침의 작업 수준: KPI를 정의하고 공유하는 데는 낮은 수준의 작업량이 필요합니다. 이 작업은 일반적으로 몇 주간 비즈니스 이해관계자를 만나 목표, 전략 및 워크로드 지표를 검토하는 방식으로 수행 가능합니다.

리소스

관련 문서:

- CloudWatch 설명서
- 모니터링, 로깅 및 성능 APN 파트너
- X-Ray 설명서
- Amazon CloudWatch 대시보드 사용
- Amazon QuickSight KPI

관련 동영상:

- AWS re:Invent 2019: Scaling up to your first 10 million users(ARC211-R)
- Cut through the chaos: Gain operational visibility and insight(MGT301-R1)
- 모니터링 플랜 세우기

관련 예시:

- Amazon QuickSight로 대시보드 생성

PERF07-BP04 모니터링을 사용하여 경보 기반 알림 생성

정의한 성능 관련 KPI를 사용하여 측정값이 예상 경계를 벗어나는 경우 경보를 자동으로 생성하는 모니터링 시스템을 사용합니다.

Amazon CloudWatch는 아키텍처의 리소스 전반에서 지표를 수집할 수 있습니다. 또한 사용자 지정 지표를 수집하고 게시하여 비즈니스 또는 파생 지표를 파악할 수도 있습니다. CloudWatch 또는 타사 모니터링 서비스를 사용하여 임계값이 초과되었음을 나타내는 경보를 설정합니다. 이 경보는 지표가 필요한 경계를 벗어났음을 나타냅니다.

일반적인 안티 패턴:

- 직원을 통해서만 지표를 살피고 문제가 발생할 경우 대응하도록 합니다.
- 서버리스 워크플로를 트리거하여 동일한 작업을 수행할 수 있음에도 불구하고 운영 런북만 사용합니다.

이 모범 사례 정립의 이점: 미리 정의된 임계값 또는 지표에서 이상 동작을 식별하는 기계 학습 알고리즘을 기반으로 경보를 설정하고 작업을 자동화할 수 있습니다. 이 동일한 경보를 사용하여 서버리스 워크플로를 트리거한 다음 워크로드의 성능 특성을 수정할 수 있습니다(예: 컴퓨팅 용량 증가, 데이터베이스 구성 변경).

이 모범 사례를 정립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

지표 모니터링: Amazon CloudWatch는 아키텍처의 리소스 전반에서 지표를 수집할 수 있습니다. 사용자 지정 지표를 수집하고 게시하여 비즈니스 또는 파생 지표를 파악할 수 있습니다. CloudWatch 또는 타사 모니터링 서비스를 사용하여 임계값 위반 시점을 나타내는 경보를 설정합니다.

리소스

관련 문서:

- CloudWatch 설명서
- 모니터링, 로깅 및 성능 APN 파트너
- X-Ray 설명서

- CloudWatch에서 경보 및 경보 작업 사용

관련 동영상:

- AWS re:Invent 2019: Scaling up to your first 10 million users(ARC211-R)
- Cut through the chaos: Gain operational visibility and insight(MGT301-R1)
- 모니터링 플랜 세우기
- Amazon CloudWatch Events에서 AWS Lambda 사용

관련 예시:

- Cloudwatch Logs 사용자 지정 경보

PERF07-BP05 정기적인 간격으로 지표 검토

주기적인 유지 관리의 일환으로 또는 이벤트나 인시던트 대응 과정에서 수집된 지표를 검토합니다. 이러한 검토를 수행하면 문제를 해결하는 데 반드시 필요했던 지표와 문제를 확인/해결/방지하는 데 도움이 되었던 지표(추적한 경우)를 추가로 파악할 수 있습니다.

인시던트나 이벤트 대응의 일환으로 문제를 해결하는 데 도움이 되었던 지표와, 현재는 추적 중이지 않지만 도움이 되었을 수 있는 지표를 평가합니다. 이 평가 결과를 토대로 하여 수집한 지표의 품질을 개선하면 이후 인시던트를 예방하거나 더 빨리 해결할 수 있습니다.

일반적인 안티 패턴:

- 지표가 장기간 경보 상태로 유지되는 것을 허용합니다.
- 자동화 시스템으로 수행할 수 없는 경보를 생성합니다.

이 모범 사례 수립의 이점: 수집 중인 지표를 지속적으로 검토하여 문제가 올바르게 식별, 해결 또는 방지되는지 확인합니다. 지표를 장기간 경보 상태로 유지할 경우에도 지표가 부실해질 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

지속적으로 지표 수집 및 모니터링 과정 개선: 인시던트나 이벤트 대응의 일환으로 문제를 해결하는 데 도움이 되었던 지표와 현재는 추적 중이지 않지만 도움이 되었을 수 있는 지표를 평가합니다. 이 방법을 사용하여 수집한 지표의 품질을 개선하면 사후 인시던트를 예방하거나 더 빨리 해결할 수 있습니다.

리소스

관련 문서:

- CloudWatch 설명서
- CloudWatch 에이전트를 사용하여 Amazon EC2 인스턴스 및 온프레미스 서버에서 지표 및 로그 수집
- 모니터링, 로깅 및 성능 APN 파트너
- X-Ray 설명서

관련 동영상:

- Cut through the chaos: Gain operational visibility and insight(MGT301-R1)
- AWS의 애플리케이션 성능 관리
- 모니터링 플랜 세우기

관련 예시:

- Amazon QuickSight로 대시보드 생성
- 레벨 100: CloudWatch 대시보드를 통한 모니터링

PERF07-BP06 사전 모니터링 및 경보 생성

KPI(핵심 성능 지표)를 모니터링 및 경보 시스템과 함께 사용하여 성능 관련 문제를 선제적으로 해결합니다. 경보를 사용하여 가능한 경우 문제를 해결하는 자동화 작업을 트리거합니다. 자동 대응이 불가능한 경우 대응을 수행할 수 있는 담당자에게 경보를 에스컬레이션합니다. 예를 들어 필요한 KPI(핵심 성과 지표) 값을 예측하고 해당 값이 특정 임계값을 초과하는 경우 경보를 생성할 수 있는 시스템이나, KPI가 필요한 값의 범위를 벗어나는 경우 배포를 자동으로 중지하거나 롤백할 수 있는 도구가 있습니다.

워크로드가 실행 중일 때 성능을 확인할 수 있는 프로세스를 구현합니다. 워크로드가 최적의 상태로 작동하고 있는지를 확인할 수 있도록 성능 기대치 관련 기준을 설정하고 모니터링 대시보드를 구축합니다.

일반적인 안티 패턴:

- 워크로드에 대한 운영 변경을 수행할 수 있는 기능을 운영 직원에게만 허용합니다.

- 사전 조치 없이 모든 경보를 운영 팀으로 필터링합니다.

이 모범 사례 수립의 이점: 경보 작업을 사전에 해결하면 지원 직원이 자동으로 실행할 수 없는 항목에 집중할 수 있습니다. 이렇게 하면 운영 직원은 모든 경보를 처리해야 하는 부담 없이 중요한 경보에만 집중할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

운영 중 성능 모니터링: 워크로드가 실행될 때의 성능을 파악할 수 있는 프로세스를 구현합니다. 모니터링 대시보드를 구축하고 성능 기대치에 대한 기준을 설정합니다.

리소스

관련 문서:

- CloudWatch 설명서
- 모니터링, 로깅 및 성능 APN 파트너
- X-Ray 설명서
- CloudWatch에서 경보 및 경보 작업 사용

관련 동영상:

- Cut through the chaos: Gain operational visibility and insight(MGT301-R1)
- AWS의 애플리케이션 성능 관리
- 모니터링 플랜 세우기
- Amazon CloudWatch Events에서 AWS Lambda 사용

관련 예시:

- Cloudwatch Logs 사용자 지정 경보

절충

질문

- PERF 8 절충을 통해 성능을 개선하려면 어떻게 해야 합니까?

PERF 8 절충을 통해 성능을 개선하려면 어떻게 해야 합니까?

솔루션을 설계할 때 절충이 필요한 영역을 결정하면 최적의 접근 방식을 선택할 수 있습니다. 일관성, 내구성 및 공간을 포기하는 대신, 시간 및 지연 시간을 선택하여 성능을 개선할 수 있는 경우가 많습니다.

모범 사례

- PERF08-BP01 성능이 가장 중요한 영역 파악
- PERF08-BP02 설계 패턴 및 서비스 파악
- PERF08-BP03 절충이 고객 및 효율성에 주는 영향 파악
- PERF08-BP04 성능 개선의 영향 측정
- PERF08-BP05 다양한 성능 관련 전략 사용

PERF08-BP01 성능이 가장 중요한 영역 파악

워크로드 성능을 개선하여 효율성을 높이고 고객 환경을 개선할 수 있는 영역을 파악합니다. 예를 들어, 많은 양의 고객 상호 작용이 수행되는 웹 사이트에서는 엣지 서비스를 사용하여 콘텐츠 전송 위치를 고객과 더 가까운 곳으로 이동하는 방법으로 성능을 개선할 수 있습니다.

원하는 결과: 아키텍처, 트래픽 패턴 및 데이터 액세스 패턴을 이해하여 성능 효율성이 향상되고 지연 시간 및 처리 시간을 파악합니다. 워크로드가 증가하면서 고객 환경에 영향을 미칠 수 있는 잠재적 병목 현상을 파악합니다. 이러한 영역을 파악할 때는 성능 문제를 제거하기 위해 배포할 수 있는 솔루션을 살펴보는 것이 좋습니다.

일반적인 안티 패턴:

- 표준 컴퓨팅 지표, 즉 `CPUUtilization` 또는 메모리 압력이 성능 문제를 포착하기에 충분한 것으로 가정합니다.
- 선택한 모니터링 소프트웨어에서 기록한 기본 지표만 사용합니다.
- 문제가 발생한 경우에만 지표를 검토합니다.

이 모범 사례 확립의 이점: 성능의 중요 영역을 이해함으로써 워크로드 소유자가 KPI를 모니터링하고 큰 영향을 미치는 개선에 우선순위를 지정할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

트래픽 패턴, 지연 시간 및 중요한 성능 영역을 파악할 수 있는 엔드 투 엔드 추적을 설정합니다. 데이터 액세스 패턴을 모니터링하여 쿼리 속도가 느리거나 데이터 조각이 잘못되거나 잘못 분할된 데이터를 찾습니다. 로드 테스트 또는 모니터링을 사용하여 워크로드의 제한된 영역을 파악합니다.

구현 단계

1. 엔드 투 엔드 모니터링을 설정하여 모든 워크로드 구성 요소 및 지표를 캡처합니다.
    - 실제 사용자 클라이언트 측 및 프런트 엔드 세션에서 애플리케이션 성능 지표를 캡처할 수 있도록 Amazon CloudWatch RUM(실제 사용자 모니터링) 을 사용합니다.
    - 애플리케이션 레이어를 통해 트래픽을 추적하고 구성 요소와 종속 요소 간 지연 시간을 파악할 수 있도록 AWS X-Ray 를 설정합니다. X-Ray 서비스 맵을 사용하여 워크로드 구성 요소 간 관계 및 지연 시간을 확인합니다.
    - 데이터베이스 성능 지표를 확인하고 성능 개선을 파악할 수 있도록 Amazon Relational Database Service 성능 개선 도우미 를 사용합니다.
    - 데이터베이스 OS 성능 지표를 파악할 수 있도록 Amazon RDS 향상된 모니터링 을 사용합니다.
    - 워크로드 구성 요소 및 서비스별 CloudWatch 지표 를 수집하고 어떤 지표가 성능 효율성에 영향을 미치는지 파악합니다.
    - 애플리케이션 레이어를 통해 트래픽을 추적하고 구성 요소와 종속 요소 간 지연 시간을 파악할 수 있도록 Amazon DevOps Guru 를 설정합니다.
2. 지표를 생성하고 트래픽 패턴, 병목 현상 및 중요한 성능 영역을 파악하기 위한 테스트를 수행합니다.
    - 애플리케이션 레이어를 통해 트래픽을 추적하고 구성 요소와 종속 요소 간 지연 시간을 파악할 수 있도록 CloudWatch Synthetic Canary 를 설정하고 이때 `cron` 작업 또는 속도 표현식을 사용하여 시간에 따라 일관된 지표를 생성하도록 합니다.
    - 최대 트래픽을 생성하거나 예상 증가율로 워크로드를 테스트할 수 있도록 AWS 분산 로드 테스트 솔루션을 사용합니다.
3. 지표 및 텔레메트리를 평가하여 중요한 성능 영역을 파악합니다. 팀과 함께 이러한 영역을 검토하여 병목 현상을 방지할 수 있는 모니터링 및 솔루션을 논의합니다.
4. 성능 개선을 실험하고 데이터로 이러한 변경 사항을 측정합니다.
    - 워크로드의 새로운 개선 및 워크로드에 영향을 미치는 성능을 테스트하도록 CloudWatch Evidently 를 사용합니다.

구현 계획의 작업 수준: 이 모범 사례를 확립하려면 엔드 투 엔드 지표를 검토하고 현재 워크로드 성능을 인지해야 합니다. 엔드 투 엔드 모니터링을 설정하고 중요한 성능 영역을 파악하는 것은 중간 수준의 작업입니다.

리소스

관련 문서:

- Amazon Builders' Library
- X-Ray 설명서
- Amazon CloudWatch RUM
- Amazon DevOps Guru
- CloudWatch RUM 및 X-Ray

관련 동영상:

- Amazon Builders' Library 소개(DOP328)
- Amazon CloudWatch Synthetics 데모

관련 예시:

- Amazon CloudWatch Synthetics를 활용한 페이지 로드 시간 측정
- Amazon CloudWatch RUM 웹 클라이언트
- X-Ray SDK for Node.js
- X-Ray SDK for Python
- X-Ray SDK for Java
- X-Ray SDK for .Net
- X-Ray SDK for Ruby
- X-Ray 대몬(daemon)
- AWS에서의 분산 로드 테스트

PERF08-BP02 설계 패턴 및 서비스 파악

워크로드 성능 개선에 도움이 되는 다양한 설계 패턴과 서비스를 조사하고 파악합니다. 분석을 수행하는 동안 성능 개선을 위해 트레이드-오프할 수 있는 요소를 파악합니다. 예를 들어, 캐시 서비스를 사용

하면 데이터베이스 시스템에 가해지는 로드를 줄일 수 있습니다. 그러나 캐싱은 궁극적으로 일관성을 가져올 수 있으며 비즈니스 요구 사항 및 고객 기대치 내에서 구현하기 위해 엔지니어링 작업이 필요합니다.

원하는 결과: 설계 패턴을 조사하면 성능이 가장 우수한 시스템을 지원하는 아키텍처 설계를 선택할 수 있습니다. 사용 가능한 성능 구성 옵션 및 이러한 옵션이 워크로드에 영향을 미치는 방식을 확인합니다. 이러한 옵션이 아키텍처와 상호 작용하는 방식, 그리고 측정된 성능과 최종 사용자의 체감 성능에 주는 영향을 파악해야 워크로드 성능을 최적화할 수 있습니다.

일반적인 안티 패턴:

- 기존의 모든 IT 워크로드 성능 전략이 클라우드 워크로드에 가장 적합하다고 가정합니다.
- 관리형 서비스를 사용하는 대신 캐싱 솔루션을 구축하고 관리합니다.
- 어떤 패턴이 위크로드 성능을 개선하는지 평가하지 않고 모든 워크로드에 대해 동일한 설계 패턴을 사용합니다.

이 모범 사례 확립의 이점: 워크로드에 적합한 설계 패턴 및 서비스를 선택하면 성능을 최적화하고 운영 우수성을 개선하며 신뢰성을 개선할 수 있습니다. 올바른 설계 패턴이 현재 워크로드 특성을 충족하고 향후 성장 또는 변경에 맞춰 확장할 수 있도록 돕습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

사용 가능한 성능 구성 옵션과 이러한 옵션이 워크로드에 미치는 영향을 확인합니다. 이러한 옵션이 아키텍처와 상호 작용하는 방식과 측정된 성능과 사용자의 체감 성능에 미치는 영향을 파악해야 워크로드 성능을 최적화할 수 있습니다.

구현 단계:

1. 워크로드 성능을 개선하는 설계 패턴을 평가 및 검토합니다.
    a. Amazon Builders' Library 는 Amazon의 기술 구축 및 운영 방식에 관한 상세한 설명을 제공합니다. 이러한 문서는 Amazon의 선임 엔지니어가 작성하며 아키텍처, 소프트웨어 전송 및 운영 전반에 걸친 주제를 다룹니다.
    b. AWS 솔루션 라이브러리 는 서비스, 코드 및 구성을 모아 놓은 바로 배포 가능한 솔루션 컬렉션입니다. 이러한 솔루션은 일반적인 사용 사례와 산업 또는 워크로드 유형별로 그룹화된 설계 패턴을 기반으로 AWS 및 AWS 파트너가 생성했습니다. 예를 들어 워크로드에 대해 분산 로드 테스트 솔루션 을 설정할 수 있습니다.

   c. AWS 아키텍처 센터 는 설계 패턴, 콘텐츠 유형 및 기술별로 그룹화된 참조 아키텍처 다이어그램을 제공합니다.

   d. AWS 샘플 은 실습 예제로 가득 찬 GitHub 리포지토리로, 일반적인 아키텍처 패턴, 솔루션 및 서비스를 살펴보는 데 도움이 됩니다. 최신 서비스 및 예제로 자주 업데이트됩니다.

2. 선택한 설계 패턴을 모델링하도록 워크로드를 개선하고 서비스 및 서비스 구성 옵션을 사용하여 워크로드 성능을 개선합니다.

   a. 다음 위치에서 사용할 수 있는 리소스로 내부 팀을 교육할 수 있습니다. AWS Skills Guild.

   b. AWS Partner Network 을(를) 사용하여 전문 지식을 빠르게 제공하고 개선 역량을 확장할 수 있습니다.

구현 계획의 작업 수준: 이 모범 사례를 확립하려면 워크로드 성능을 개선하는 데 도움이 될 수 있는 설계 패턴 및 서비스를 파악해야 합니다. 설계 패턴을 평가한 후 설계 패턴 구현의 작업 수준은 높음 입니다.

리소스

관련 문서:

- AWS 아키텍처 센터
- AWS Partner Network
- AWS 솔루션 라이브러리
- AWS 지식 센터
- Amazon Builders' Library
- 로드 셰딩을 사용하여 오버로드 방지
- 캐싱 관련 당면 과제 및 전략

관련 동영상:

- Amazon Builders’ Library 소개(DOP328)
- This is My Architecture

관련 예시:

- AWS 샘플

- AWS SDK 예시

PERF08-BP03 절충이 고객 및 효율성에 주는 영향 파악

성능 관련 개선 사항을 평가할 때는 고객 및 워크로드 효율성에 영향을 미치는 옵션을 결정합니다. 예를 들어 키–값 데이터 스토어를 사용하여 시스템 성능이 개선되는 경우, 이 옵션의 지속되는 특성이 결과적으로 고객에 미치는 영향을 평가하는 것이 중요합니다.

지표와 모니터링을 통해 시스템에서 성능 수준이 낮은 영역을 파악합니다. 성능을 개선할 수 있는 방법과 해당 개선 과정에서 트레이드-오프해야 하는 요소와 성능 개선 작업이 시스템과 사용자 환경에 미치는 영향을 확인합니다. 예를 들어 데이터 캐싱 구현 시에는 성능을 크게 개선할 수 있지만, 캐시된 데이터를 업데이트하거나 무효화할 방법 및 시기와 관련된 명확한 전략을 마련해야 잘못된 시스템 동작을 방지할 수 있습니다.

일반적인 안티 패턴:

- 최종 일관성과 같은 구현에 대한 절충점이 있더라도 모든 성능 이점을 구현해야 한다고 가정합니다.
- 성능 문제가 심각한 지점에 도달했을 때만 워크로드 변경을 평가합니다.

이 모범 사례 수립의 이점: 잠재적인 성능 관련 개선 사항을 평가할 때는 변경에 대한 절충이 워크로드 요구 사항과 일치하는지 여부를 결정해야 합니다. 경우에 따라 절충을 보상하기 위해 추가 제어를 구현해야 할 수도 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

절충 파악: 지표와 모니터링을 사용하여 시스템에서 성능 수준이 낮은 영역을 파악합니다. 개선 방법과 절충이 시스템과 사용자 환경에 미치는 영향을 결정합니다. 예를 들어 데이터 캐싱 구현 시에는 성능을 크게 개선할 수 있지만, 잘못된 시스템 동작을 방지하기 위해 캐시된 데이터를 업데이트하거나 무효화할 방법 및 시기와 관련된 명확한 전략을 마련해야 합니다.

리소스

관련 문서:

- Amazon Builders' Library
- Amazon QuickSight KPI

- Amazon CloudWatch RUM
- X-Ray 설명서

관련 동영상:

- Amazon Builders' Library 소개(DOP328)
- 모니터링 플랜 세우기
- Amazon CloudWatch RUM을 통한 애플리케이션 최적화
- Amazon CloudWatch Synthetics 데모

관련 예시:

- Amazon CloudWatch Synthetics를 활용한 페이지 로드 시간 측정
- Amazon CloudWatch RUM 웹 클라이언트

PERF08-BP04 성능 개선의 영향 측정

성능 개선을 위해 변경이 수행된 경우 수집된 지표와 데이터를 평가합니다. 이 정보를 사용하여 성능 개선이 워크로드, 워크로드의 구성 요소 및 고객에게 미치는 영향을 확인합니다. 이 측정을 수행하면 트레이드-오프를 통한 성능 개선을 파악할 수 있으며 부정적인 부작용 발생 여부를 확인할 수 있습니다.

잘 설계된 시스템은 다양한 성능 관련 전략을 조합하여 활용합니다. 따라서 지정된 핫스팟이나 병목 현상을 가장 많이 개선할 수 있는 전략을 결정해야 합니다. 예를 들어 여러 관계형 데이터베이스 시스템에서 데이터를 샤딩하면 전반적인 처리량이 높아지는 동시에 트랜잭션이 계속 지원됩니다. 각 샤드 내에서 캐싱을 수행하면 로드를 줄일 수 있습니다.

일반적인 안티 패턴:

- 관리형 서비스로 제공되는 기술을 수동으로 배포하고 관리합니다.
- 여러 구성 요소를 사용하여 워크로드의 성능을 높일 수 있는 경우 네트워킹과 같은 하나의 구성 요소에만 집중합니다.
- 고객의 피드백과 관점을 유일한 벤치마크로 삼습니다.

이 모범 사례 수립의 이점: 성능 전략을 구현할 때는 여러 서비스와 기능을 선택해야 합니다. 이렇게 하면 성능에 대한 워크로드 요구 사항을 충족할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

Well-Architected 시스템은 다양한 성능 관련 전략을 조합하여 활용합니다. 따라서 지정된 핫스팟이나 병목 현상을 가장 많이 개선할 수 있는 전략을 결정해야 합니다. 예를 들어 여러 관계형 데이터베이스 시스템에서 데이터를 샤딩하면 전반적인 처리량이 높아지는 동시에 트랜잭션이 계속 지원됩니다. 각 샤드 내에서 캐싱을 수행하면 로드를 줄일 수 있습니다.

리소스

관련 문서:

- Amazon Builders' Library
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- AWS에서의 분산 로드 테스트

관련 동영상:

- Amazon Builders’ Library 소개(DOP328)
- Amazon CloudWatch RUM을 통한 애플리케이션 최적화
- Amazon CloudWatch Synthetics 데모

관련 예시:

- Amazon CloudWatch Synthetics를 활용한 페이지 로드 시간 측정
- Amazon CloudWatch RUM 웹 클라이언트
- AWS에서의 분산 로드 테스트

PERF08-BP05 다양한 성능 관련 전략 사용

해당하는 경우 다수의 전략을 활용하여 성능을 개선합니다. 예를 들어 데이터 캐싱 등의 전략을 사용해 과도한 네트워크 또는 데이터베이스 호출을 방지하고, 데이터베이스 엔진용 읽기 전용 복제본을 사용해 읽기 속도를 높이고, 가능한 경우 데이터 샤딩/압축을 수행하여 데이터 볼륨을 줄이고, 제공되는 결과를 버퍼링/스트리밍하여 차단을 방지하는 등의 전략을 사용할 수 있습니다.

워크로드를 변경할 때는 지표를 수집 및 평가하여 변경의 영향을 확인합니다. 시스템 및 최종 사용자에 대한 영향을 모두 측정하여 절충 작업이 워크로드에 미치는 영향을 파악합니다. 로드 테스트 등의 체계적인 방식을 사용해 개별 요소 트레이드-오프 시, 성능을 개선할 수 있는지 여부를 파악합니다.

일반적인 안티 패턴:

- 고객이 불만을 제기하지 않으면 워크로드 성능이 적절한 것이라고 가정합니다.
- 성능 관련 변경을 수행한 후에만 성능에 대한 데이터를 수집합니다.

이 모범 사례 수립의 이점: 성능 및 리소스 사용률을 최적화하려면 통합된 운영 보기, 세분화된 실시간 데이터 및 기간별 참조가 필요합니다. 대시보드를 생성하고 데이터에 대한 지표 산술을 수행하여 워크로드 운영 및 사용률의 시간대별 변화에 대한 인사이트를 도출할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

데이터 기반 접근 방식을 통해 아키텍처 발전 도모: 워크로드를 변경할 때는 지표를 수집 및 평가하여 변경의 영향을 확인합니다. 시스템 및 최종 사용자에 대한 영향을 모두 측정하여 절충 작업이 워크로드에 미치는 영향을 파악합니다. 로드 테스트 등의 체계적인 방식을 사용해 개별 요소 트레이드-오프 시, 성능을 개선할 수 있는지 여부를 파악합니다.

리소스

관련 문서:

- Amazon Builders' Library
- Amazon ElastiCache 구현 모범 사례
- AWS 데이터베이스 캐싱
- Amazon CloudWatch RUM
- AWS에서의 분산 로드 테스트

관련 동영상:

- Amazon Builders’ Library 소개(DOP328)
- AWS purpose-built databases(DAT209-L)
- Amazon CloudWatch RUM을 통한 애플리케이션 최적화

관련 예시:

- Amazon CloudWatch Synthetics를 활용한 페이지 로드 시간 측정
- Amazon CloudWatch RUM 웹 클라이언트
- AWS에서의 분산 로드 테스트

비용 최적화

비용 최적화 원칙은 시스템을 실행하여 최저 가격으로 비즈니스 가치를 제공할 수 있는 역량을 포함합니다. 구현 방법에 대한 권장 가이드는 비용 최적화 원칙 백서에서 확인할 수 있습니다.

모범 사례 영역

- 클라우드 재무 관리 시행
- 지출 및 사용량 인식
- 비용 효율적인 리소스
- 수요 관리 및 리소스 공급
- 시간 경과에 따른 최적화

클라우드 재무 관리 시행

질문

- COST 1 클라우드 재무 관리를 어떻게 구현합니까?

COST 1 클라우드 재무 관리를 어떻게 구현합니까?

조직은 클라우드 재무 관리를 시행하여 비용과 사용량을 최적화하고 AWS에서 규모를 확대하면서 비즈니스 가치를 실현하고 재정적 성공을 거둘 수 있습니다.

모범 사례

- COST01-BP01 비용 최적화 역할 설정
- COST01-BP02 재무 팀과 기술 팀 간의 파트너십 수립
- COST01-BP03 클라우드 예산 및 예측 수립
- COST01-BP04 조직의 프로세스에서 비용 인식 구현
- COST01-BP05 비용 최적화 보고 및 알림

- COST01-BP06 사전 예방적 비용 모니터링
- COST01-BP07 새로운 서비스 릴리스로 최신 상태 유지
- COST01-BP08 비용 인식 문화 조성
- COST01-BP09 비용 최적화의 비즈니스 가치 정량화

COST01-BP01 비용 최적화 역할 설정

조직 전체에서 비용 인식을 확립하고 유지를 담당하는 팀(클라우드 비즈니스 오피스 또는 클라우드 혁신 센터)을 구성합니다. 조직 전체에서 재무, 기술 및 비즈니스 역할을 수행하는 인력이 팀에 필요합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

클라우드 컴퓨팅에서 비용 인식 문화를 확립하고 유지를 담당하는 클라우드 비즈니스 오피스(CBO) 또는 클라우드 혁신 센터(CCOE) 팀을 구성합니다. 기존의 개인, 조직 내부의 팀 또는 조직 전반의 핵심 재무, 기술 및 조직 이해관계자로 구성된 신규 팀이 이 역할을 맡을 수 있습니다.

이 역할(개인 또는 팀)은 업무 시간의 필요한 비율을 비용 관리 및 비용 최적화 활동에 우선적으로 할애합니다. 소규모 조직의 경우 이 역할이 소비하는 시간의 비율은 대기업의 정규직 역할보다 작을 수 있습니다.

이 역할은 프로젝트 관리, 데이터 과학, 재무 분석 및 소프트웨어 또는 인프라 개발 역량을 통해 다방면에서 접근해야 합니다. 이 역할은 다음 세 가지 다른 부문 내에서 비용 최적화를 실행하여 워크로드의 효율성을 개선할 수 있습니다.

- 중앙화: 고객은 재무 운영, 비용 최적화, CBO 또는 CCOE 등과 같은 지정된 팀을 통해 거버넌스 메커니즘을 설계 및 구현하고 회사 전체에서 모범 사례를 추진할 수 있습니다.
- 탈중앙화: 최적화를 실행할 수 있도록 기술 팀에 영향을 미칩니다.
- 하이브리드: 중앙화 팀과 탈중앙화 팀이 함께 비용 최적화를 실행하기 위해 협력할 수 있습니다.

이 역할의 성과는 비용 최적화 목표를 실행하고 제공할 수 있는 능력을 기준으로 측정될 수 있습니다(예: 워크로드 효율성 지표).

변화를 만들기 위해 이 역할에 대한 경영진 차원의 지원을 아끼지 않는 것이 성공의 핵심 요소입니다. 스폰서는 비용 효율적인 클라우드 사용의 옹호자로 간주되며 이 역할에 대한 에스컬레이션 지원을 제

공하여 비용 최적화 활동이 조직이 정의한 우선 순위 수준에 따라 처리될 수 있도록 합니다. 그러지 않으면 지침이 무시되고 비용 절감 기회가 우선되지 않습니다. 스폰서와 이 역할의 담당자는 함께 조직에서 클라우드가 효율적으로 사용되는지 확인하고 지속적으로 비즈니스 가치를 전달합니다.

Business, Enterprise-On-Ramp 또는 Enterprise Support 플랜을 사용 중으로 이러한 팀 또는 역할을 구축하는 데 도움이 필요한 경우 계정 팀을 통해 클라우드 재무 관리(CFM) 전문가에게 문의하세요.

구현 단계

- 주요 구성원 정의: 조직의 모든 관련 부분이 비용 관리에 기여하고 있으며 이해 관계가 있는지 확인해야 합니다. 조직 내의 공통 팀으로는 일반적으로 재무, 애플리케이션 또는 제품 소유자, 관리, 기술 팀(DevOps) 등이 있습니다. 일부는 정규직(재무, 기술)이며 일부는 필요에 따라 주기적으로 근무합니다. CFM을 수행하는 개인 또는 팀에는 일반적으로 다음이 필요합니다.
  - 소프트웨어 개발 기술 - 스크립트 및 자동화가 구축되는 경우
  - 인프라 엔지니어링 기술 - 스크립트 또는 자동화를 배포하고 서비스 또는 리소스 프로비저닝 방법을 파악하기 위한 경우
  - 운영 감각 - CFM은 클라우드의 효율적인 사용을 측정, 모니터링, 수정, 계획 및 스케일링하여 클라우드를 효율적으로 운영할 수 있는 솔루션
- 목표 및 지표 정의: 이 역할은 조직에 가치를 다양한 방법으로 전달해야 합니다. 역할 목표는 정의되며 조직이 발전함에 따라 계속해서 변화합니다. 공통 활동으로는 조직 전체의 비용 최적화에 대한 교육 프로그램의 생성과 실행, 비용 최적화를 위한 모니터링과 보고 등 조직 차원의 표준 개발, 최적화 실행을 위한 워크로드 목표 설정 등이 있습니다. 또한 이 역할은 조직의 비용 최적화 역량에 대해 조직에 정기적으로 보고해야 합니다.

  가치 기반 주요 성과 지표(KPI)를 정의할 수 있습니다. KPI는 비용 기반 또는 가치 기반일 수 있습니다. KPI를 정의할 때 효율성과 예상되는 비즈니스의 성과 측면에서 예상 비용을 계산할 수 있습니다. 가치 기반 KPI는 비용과 사용 지표를 비즈니스 가치 요인에 묶고 AWS 지출의 변화를 합리화하도록 돕습니다. 가치 기반 KPI를 이끌어 내기 위한 첫 번째 단계는 표준 KPI 세트를 선정하고 합의하기 위해 조직 전체에서 협력하는 것입니다.
- 정기적인 주기 설정: 그룹(재무, 기술 및 비즈니스 팀)은 정기적으로 함께 모여 목표와 지표를 검토해야 합니다. 일반적인 주기는 조직의 상태를 검토하고 현재 실행 중인 프로그램을 검토한 후 전반적인 재무 및 최적화 지표를 검토하는 것입니다. 그런 다음 주요 워크로드가 더 자세히 보고됩니다.

  이러한 정기적인 회의 중 워크로드 효율성(비용) 및 비즈니스 성과를 검토할 수 있습니다. 예를 들어, 워크로드에 대한 20%의 비용 증가는 고객 사용량 증가와 일치할 수 있습니다. 이 경우 20%라는 비용 증가는 투자로 해석할 수 있습니다. 이처럼 정기적인 주기의 회의는 팀이 전체 조직에 의미를 줄 수 있는 가치 기반 KPI를 식별하는 데 도움이 될 수 있습니다.

리소스

관련 문서:

- AWS CCOE 블로그
- 클라우드 비즈니스 오피스 생성
- CCOE - 클라우드 혁신 센터

관련 동영상:

- Vanguard CCOE 성공 사례

관련 예시:

- 클라우드 혁신 센터(CCOE)를 사용하여 전체 기업 혁신
- 전체 기업 혁신을 위한 CCOE 구축
- CCOE 구축 시 피해야 할 7가지 위험

COST01-BP02 재무 팀과 기술 팀 간의 파트너십 수립

클라우드 여정의 모든 단계에서 비용 및 사용량 논의에 재무 및 기술 팀이 참여하도록 하십시오. 팀은 정기적으로 만나 조직의 목적과 목표, 비용 및 사용량의 현재 상태, 재무 및 회계 실무와 같은 주제에 대해 논의합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

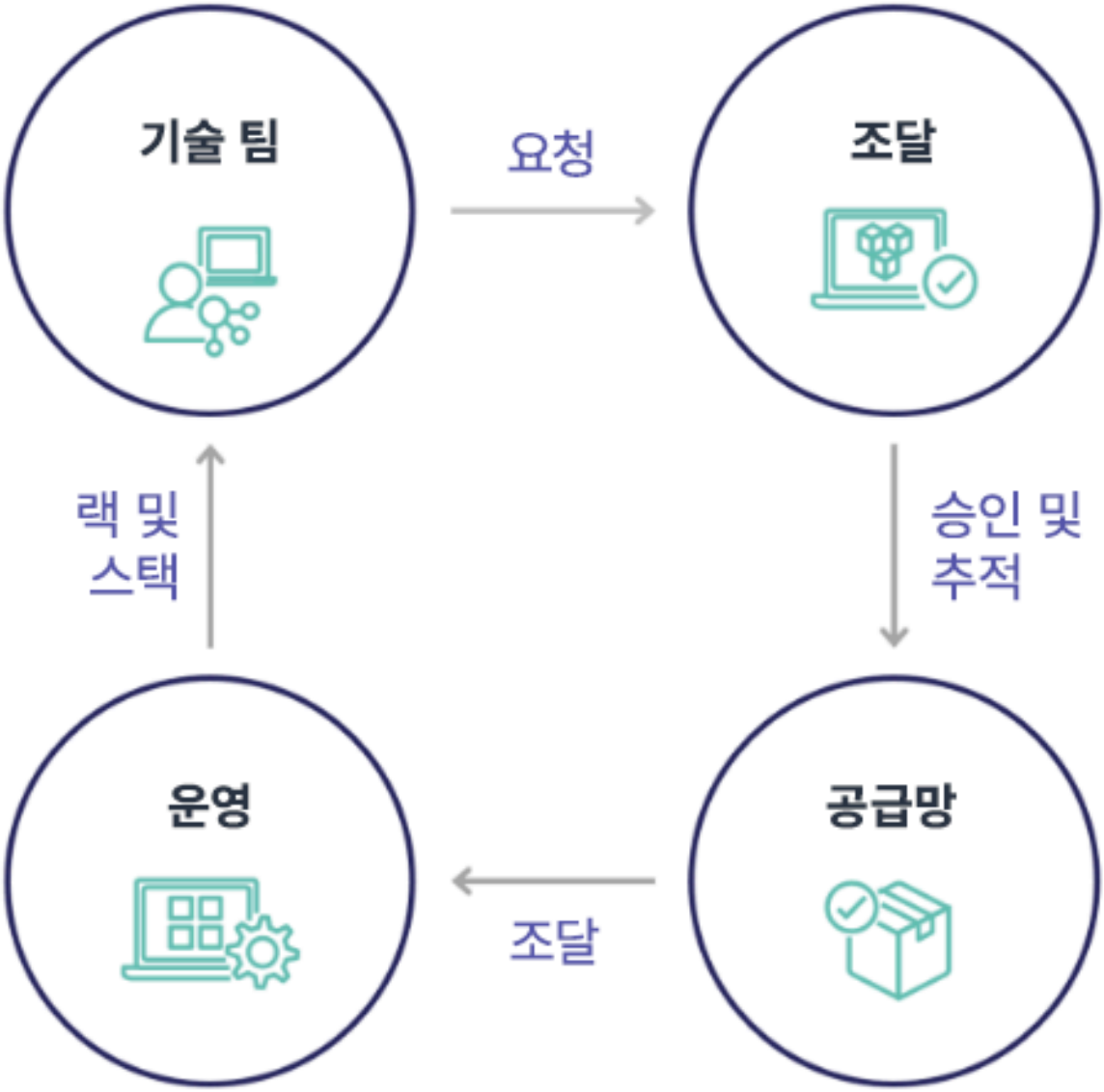
구현 가이드

기술 팀이 클라우드에서 더 빠른 혁신을 달성할 수 있는 이유는 승인, 구매 및 인프라 배포 주기가 짧기 때문입니다. 이는 이전에 데이터 센터와 온프레미스 환경에서 자본을 조달 및 배포하고 프로젝트 승인 시에만 비용을 할당하기 위해 시간 소모적이고 리소스 집약적인 프로세스를 실행했던 재무 조직에 있어서 하나의 조정이 될 수 있습니다.

재무 및 조달 조직 관점에서 자본 예산 책정, 자본 요청, 승인, 조달 및 물리적 인프라 설치 프로세스는 수십 년 동안 학습하고 표준화하고 있는 분야입니다.

- 엔지니어링 또는 IT 팀은 일반적으로 요청자입니다.
- 다양한 재무 팀은 승인자 및 조달자의 역할을 합니다.

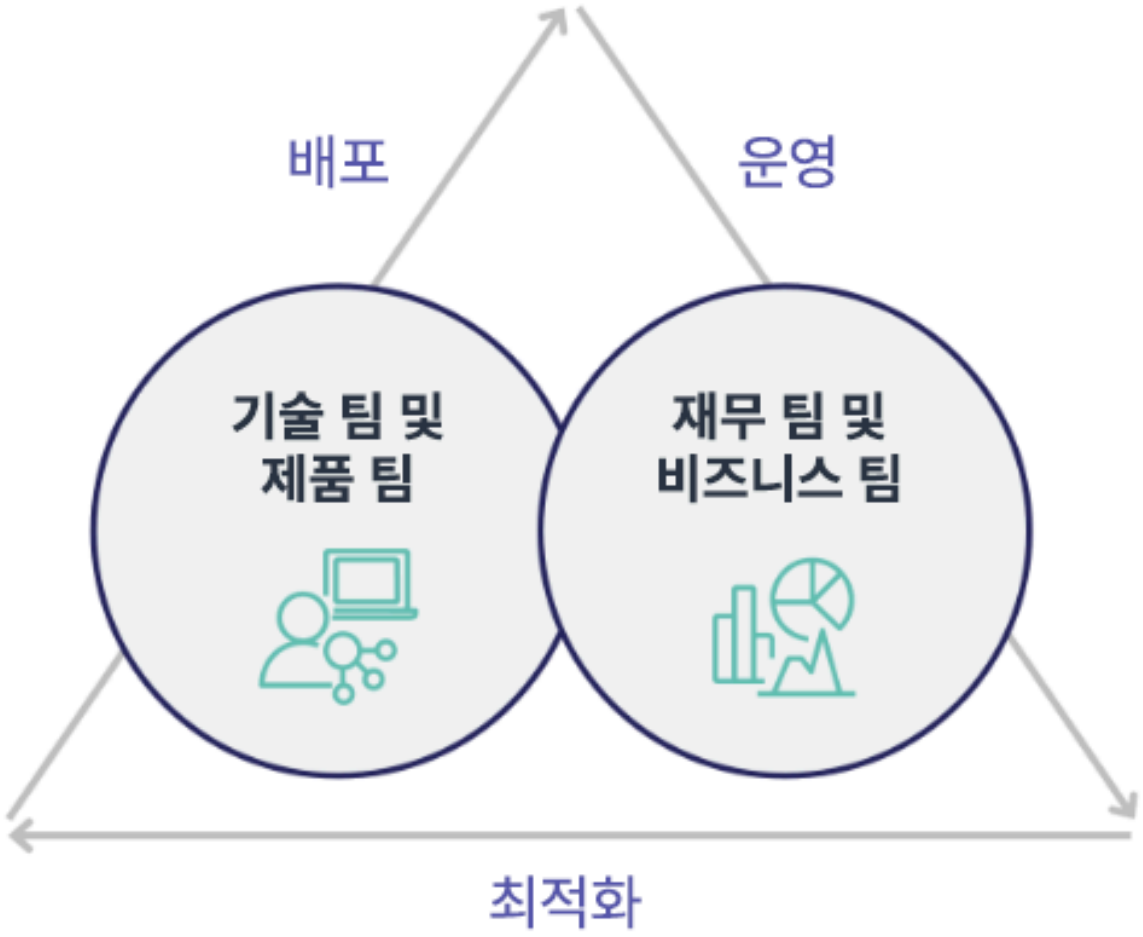
- 운영 팀은 바로 사용할 수 있는 인프라를 래킹, 스태킹 및 전달합니다.



클라우드를 도입하면 인프라 조달 및 사용 시 더 이상 종속성의 사슬에 얽매이지 않아도 됩니다. 클라우드 모델에서 기술 팀과 제품 팀은 구축자가 아니라 제품의 운영자와 소유자가 되며, 조달과 배포를 비롯하여 과거에는 재무 팀과 운영 팀과 관련이 있던 작업 대부분을 담당합니다.

클라우드 리소스를 프로비저닝하는 데는 사용자 계정과 올바른 권한 세트만 있으면 됩니다. 또한 IT 및 재무 위험을 줄여줍니다. 즉, 팀에서 몇 번만 클릭하거나 API를 호출하기만 하면 유휴 또는 필요 없는 클라우드 리소스를 종료할 수 있습니다. 게다가 기술 팀이 더 빠르게 혁신할 수 있도록 하여, 민첩하게 실험을 시작하고 분석하는 능력이 향상됩니다. 클라우드 사용 시 다양한 측면이 자본 예산과 예측 관점

에서 예측 가능성에 영향을 미칠 수 있지만 클라우드를 이용하여 조직은 과다 프로비저닝 비용을 줄이고 보수적인 과소 프로비저닝과 관련된 기회 비용을 절감할 수 있습니다.



주요 재무 관계자와 기술 관계자 간의 파트너십을 수립하면 조직의 목표를 함께 이해하고 클라우드 컴퓨팅의 가변 지출 모델에서 재정적 성공을 지원하는 메커니즘을 개발할 수 있습니다. 다음을 포함한 조직 내부의 관련 팀이 클라우드 여정의 모든 단계에서 비용 및 사용 논의에 참여해야 합니다.

- 재무 책임자: CFO, 재무 관리자, 재무 계획자, 비즈니스 분석가, 구매, 소싱 및 지급 계정 부서는 클라우드 모델 소비, 구매 옵션 및 월별 인보이스 프로세스를 이해해야 합니다. 재무 팀은 IT 가치 스토리를 생성하고 널리 공유하기 위해 기술 팀과 협력해 어떻게 기술 비용이 비즈니스 성과와 연결되는지 비즈니스 팀이 이해할 수 있도록 도와야 합니다. 이런 관점에서 기술에 대한 지출은 비용이 아니라 투자로 인식됩니다. 클라우드와 온프레미스 운영 사이에는 근본적인 차이(예: 사용량 변경 속도, 종량과금제, 계층화된 요금, 요금 모델, 세부 결제 및 사용량 정보)가 있으므로 재무 조직은 클라우드 사용이 구매 프로세스, 인센티브 추적, 비용 할당 및 재무제표 등의 비즈니스 측면에 미치는 영향을 이해해야 합니다.

- 기술 책임자: 기술 책임자(제품 및 애플리케이션 소유자 포함)는 재무 요구 사항(예: 예산 제약)과 비즈니스 요구 사항(예: 서비스 수준 계약)을 알고 있어야 합니다. 그러면 적절한 조직 목표를 달성하기 위한 워크로드를 구현할 수 있습니다.

재무와 기술의 파트너십은 다음과 같은 이점을 제공합니다.

- 재무 팀과 기술 팀이 비용과 사용량을 거의 실시간으로 파악할 수 있습니다.
- 재무 팀과 기술 팀이 클라우드 지출 차이를 처리하기 위한 표준 운영 절차를 설정할 수 있습니다.
- 재무 관계자는 자본을 투입하여 구매 시 약정을 통해 할인을 받는 방법(예: 예약 인스턴스 또는 AWS 절감형 플랜)과 클라우드를 사용하여 조직 성장을 지원하는 방법에 대한 전략적 자문 역할을 할 수 있습니다.
- 클라우드에서 기존의 지급 계정 및 구매 프로세스를 함께 사용할 수 있습니다.
- 재무 팀과 기술 팀이 공동으로 향후 AWS 비용과 사용량을 예측하여 조직 예산을 조정하고 작성할 수 있습니다.
- 공동의 언어를 통해 조직 간 커뮤니케이션과 재무 개념에 대한 일반적인 이해를 개선할 수 있습니다.

조직 내에서 비용 및 사용량 논의에 관여해야 하는 추가 이해관계자는 다음과 같습니다.

- 사업부 책임자: 사업부 책임자는 사업부와 전체 회사에 지침을 제공할 수 있도록 클라우드 비즈니스 모델을 파악해야 합니다. 확장 및 워크로드 사용량을 예측해야 할 때와 예약 인스턴스 또는 Savings Plans 등의 장기 구매 옵션을 평가할 때는 이러한 클라우드 관련 지식이 반드시 필요합니다.
- 엔지니어링 팀: 재무 팀과 기술 팀 간의 파트너십 확립은 엔지니어가 클라우드 재무 관리(CFM)에 대한 조치를 취할 수 있도록 권장하는 비용 인식 문화를 구축하는 데 반드시 필요합니다. CFM 또는 재무 운영 실무자와 재무 팀에 발생하는 일반적인 문제 중 하나는 엔지니어가 클라우드의 전체 비즈니스를 이해하고 모범 사례를 따르며 권장 조치를 취하도록 하는 것입니다.
- 서드파티: 조직에서 서드파티(예: 컨설턴트 또는 도구)를 사용하는 경우 이러한 업체가 재무 목표에 부합하는지 확인해야 하며 참여 모델 및 ROI(투자 수익률)를 통해 이러한 일치를 증명할 수 있습니다. 서드파티는 일반적으로 관리 대상 워크로드 보고와 분석을 수행하며 설계 대상 워크로드의 비용을 분석합니다.

CFM을 구현하고 성공을 달성하려면 재무, 기술 및 비즈니스 팀 간의 협업과 조직 전반에서 클라우드 비용이 전달되고 평가되는 방식의 변화가 필요합니다. 엔지니어링 팀을 포함시켜 이들이 모든 단계에서 이러한 비용과 사용 논의에 참여하고 모범 사례를 따라 합의된 조치를 취할 수 있도록 장려합니다.

구현 단계

- 주요 구성원 정의: 재무 팀과 기술 팀의 모든 관련 멤버가 파트너십에 참여하는지 확인하세요. 재무 팀의 관련 멤버는 클라우드 청구서와 관련하여 업무를 수행하는 사람들입니다. 대개 CFO, 재무 관리자, 재무 계획자, 비즈니스 분석가, 구매, 소싱 담당자가 여기에 해당합니다. 기술 멤버는 대개 제품 및 애플리케이션 소유자, 기술 관리자 및 클라우드에 구축된 모든 팀의 담당자가 됩니다. 다른 멤버로는 제품 사용에 영향을 주는 마케팅과 같은 사업부 소유자와 목표 및 메커니즘에 부합하도록 하고 보고를 지원하는 컨설턴트와 같은 서드파티 등이 있을 수 있습니다.
- 논의 주제 정의: 팀 간에 공통된 주제나 공동의 이해가 필요한 주제를 정의합니다. 발생한 시점부터 청구서가 지불될 때까지 비용을 추적합니다. 관련된 모든 멤버와 적용해야 하는 조직의 프로세스를 기록합니다. 각 단계 또는 이들이 거치는 프로세스와 사용 가능한 요금 모델, 계층화된 요금, 할인 모델, 예산 책정, 재무 요구 사항 등의 관련 정보를 파악합니다.
- 정기적인 주기 설정: 재무 팀과 기술 팀 간의 파트너십을 구축하려면 일치된 방향을 수립하고 유지하기 위해 정기적인 소통 주기를 확립해야 합니다. 그룹은 목표 및 지표에 따라 정기적으로 모여야 합니다. 일반적인 주기는 조직의 상태를 검토하고 현재 실행 중인 프로그램을 검토한 후 전반적인 재무 및 최적화 지표를 검토하는 것입니다. 그런 다음 주요 워크로드가 더 자세히 보고됩니다.

리소스

관련 문서:

- AWS 뉴스 블로그

COST01-BP03 클라우드 예산 및 예측 수립

클라우드 비용 및 사용량의 매우 가변적인 특성에 맞게 기존 조직의 예산 책정 및 예측 프로세스를 조정합니다. 프로세스는 추세나 비즈니스 동인을 기반으로 하는 알고리즘 또는 둘의 조합을 사용하여 동적으로 수행되어야 합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

고객은 효율성, 속도 및 민첩성을 달성하기 위해 클라우드를 사용합니다. 이러한 특성으로 인해 클라우드의 비용 및 사용량은 매우 가변적입니다. 워크로드 효율성이 증가하거나 새 워크로드 및 기능이 배포되는 경우 비용이 증가할 수 있습니다. 워크로드 효율성이 증가하는 경우 또는 새로운 워크로드와 기능이 배포되는 경우 비용 증가가 발생할 수 있습니다. 또는 더 많은 고객을 지원하기 위해 워크로드를 확장할 경우에도 사용량 및 비용이 증가합니다. 이제 그 어느 때보다 쉽게 리소스를 이용할 수 있습니다. 또한 클라우드의 탄력성 덕분에 비용 및 예측의 탄력성 역시 보장됩니다. 따라서 이러한 가변성을 포함하도록 기존 조직의 예산 편성 프로세스를 수정해야 합니다.

추세 기반 알고리즘(기간별 비용을 입력으로 사용)을 사용하거나, 비즈니스 동인 기반 알고리즘(예: 신제품 출시 또는 리전별 확장)을 사용하거나, 또는 추세 기반 알고리즘과 비즈니스 동인 기반 알고리즘의 조합을 사용하여 기존의 예산 책정과 예측 프로세스를 보다 동적으로 수행하세요.

AWS Budgets 을(를) 사용하면 기간, 반복 또는 금액(고정 또는 가변)을 지정하고 서비스, AWS 리전 및 태그 등과 같은 필터를 추가하여 사용자 지정 예산을 세부 수준에서 설정할 수 있습니다. 기존 예산의 성과에 대한 정보를 계속해서 파악하기 위해 AWS Budgets 보고서를 생성한 다음 여러분과 이해관계자에게 해당 보고서가 이메일로 정기적으로 전송되도록 설정할 수 있습니다. 또한 대응적 조치인 AWS Budgets 알림을 실제 비용을 기반으로 생성할 수 있습니다. 예측 비용을 기반으로 생성하면 잠재적인 비용 초과를 완화할 조치를 취할 시간을 벌 수 있습니다. 비용이나 사용량이 초과되거나 예산으로 책정된 비용이 초과될 것으로 예측되면 알림이 전송됩니다.

AWS을(를) 통해 동적 예측 및 예산 책정 프로세스를 구축할 수 있는 유연성을 확보할 수 있습니다. 따라서 비용이 예산 한도를 초과하지 않는지, 아니면 초과하는지에 관한 정보를 파악할 수 있습니다.

AWS Cost Explorer 을(를) 사용하면 정의된 미래의 시간 범위에서 과거 지출을 기준으로 비용을 예측할 수 있습니다. AWS Cost Explorer의 예측 엔진은 비용 유형(예: 예약된 인스턴스)을 기준으로 과거 데이터를 구분하고 기계 학습과 규칙 기반 모델을 결합하여 모든 비용 유형 전반에서 비용을 개별적으로 예측합니다. AWS Cost Explorer 을(를) 사용하여 기간별 비용(추세 기반)에 적용된 기계 학습 알고리즘을 기반으로 일 단위(최대 3개월) 또는 월 단위(최대 12개월) 클라우드 비용을 예측할 수 있습니다.

Cost Explorer을(를) 사용하여 추세 기반 예측을 확인한 다음 AWS Pricing Calculator 을(를) 사용하여 예상 사용량(트래픽, 초당 요청 수, 필요한 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스 등)을 기반으로 AWS 사용 사례와 향후 비용을 예측합니다. 또한 AWS을(를) 사용하는 경우에도 이를 사용하여 지출 방식을 계획하고 비용 절감 기회를 찾고 정보에 기반한 결정을 내릴 수 있습니다.

AWS Cost Anomaly Detection 을(를) 사용하여 예상치 못한 비용을 줄이고 혁신 속도를 늦추지 않고 제어를 강화할 수 있습니다. AWS Cost Anomaly Detection은(는) 고급 기계 학습 기술을 사용하여 이례적인 지출과 근본 원인을 파악하기 때문에 신속하게 조치를 취할 수 있습니다. 간단한 3단계를 수행하면맥락에 맞는 고유한 모니터링 시스템을 구축하고 이례적인 비용이 감지된 경우 알림을 받을 수 있습니다. 빌더가 이를 구축하여 AWS Cost Anomaly Detection(으)로 비용을 모니터링하여 예상치 못한 청구 위험을 줄일 수 있도록 하세요.

또한 잘 구축된 비용 최적화 원칙의 재무 팀과 기술 팀 간의 파트너십 단원에서 언급한 것처럼 일관성을 위해 모두 동일한 도구 또는 프로세스를 사용하도록 하려면 IT 팀, 재무 팀 및 기타 이해관계자 간에 파트너십을 확립하고 주기적으로 소통하는 것이 중요합니다. 예산을 변경해야 하는 경우 소통 주기를 늘리면 보다 신속하게 변화에 대응할 수 있습니다.

구현 단계

- 기존 예산 및 예측 프로세스 업데이트: 예산 책정 및 예측 프로세스에서 추세 기반, 비즈니스 동인 기반 또는 이 둘의 조합을 구현합니다.
- 알림 구성: AWS Budgets 알림 및 Cost Anomaly Detection을(를) 사용합니다.
- 주요 이해관계자와 함께 정기 검토 수행: 예를 들어, IT, 재무, 플랫폼 및 기타 비즈니스 영역의 이해관계자가 비즈니스 방향과 사용의 변화에 맞춰 조정할 수 있습니다.

리소스

관련 문서:

- AWS Cost Explorer
- AWS Budgets
- AWS Pricing Calculator
- AWS Cost Anomaly Detection
- AWS License Manager

관련 예시:

- 출시: 이제 AWS Cost Explorer에서 사용 기반 예측 사용 가능
- AWS Well-Architected 실습 - 비용 및 사용 거버넌스

COST01-BP04 조직의 프로세스에서 비용 인식 구현

비용 인식을 구현하고 사용량에 영향을 미치는 신규 또는 기존 프로세스에 투명성과 책임을 더하고 비용 인식에 기존 프로세스를 활용합니다. 직원 교육에 비용 인식을 구현합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

신규 및 기존 조직 프로세스 안에 비용 인식을 구현해야 합니다. 이는 다른 모범 사례를 구현하기 위한 기본 전제 조건 중 하나입니다. 가능한 경우 기존 프로세스를 재사용하고 수정하는 것이 좋습니다. 이렇게 하면 민첩성과 속도에 미치는 영향을 최소화할 수 있습니다. 재무 및 비즈니스 이해관계자를 위해 비용 인식을 높이고 효율성 핵심 성과 지표(KPI)를 정립하기 위해 비즈니스 및 재무 팀의 의사 결정권자와 기술 팀에 클라우드 비용을 보고합니다. 다음은 워크로드에 비용 인식을 구현하는 데 도움이 되는 권장 사항입니다.

- 변경 관리에 변경이 재정에 미치는 영향을 정량화하는 비용 측정이 포함되어 있는지 확인하세요. 비용 측정을 포함하면 비용 관련 문제를 사전에 해결하고 비용 절감을 강조할 수 있습니다.
- 비용 최적화가 운영 역량의 핵심 구성 요소인지 확인하세요. 예를 들어 기존 인시던트 관리 프로세스를 활용하여 비용 및 사용량 이상 또는 비용 초과의 근본 원인을 조사하고 식별할 수 있습니다.
- 자동화 또는 도구를 사용하여 비용 절감 및 비즈니스 가치 실현을 가속화하세요. 구현 비용을 고려할 때는 투자 수익(ROI) 구성 요소를 포함하도록 대화를 구성하여 시간 또는 금전 투자의 당위성을 설명합니다.
- 약정 기반 구입 옵션, 공유 서비스 및 마켓플레이스 구입 비용을 비롯하여 클라우드 비용에 대한 내역 확인(Showback) 또는 결제 처리를 구현하여 클라우드 비용을 할당해 비용 인식이 최대한 반영된 클라우드 사용을 촉진합니다.
- 조직 전체에서 비용 인식 교육을 포함하도록 기존 교육 및 개발 프로그램을 확대하세요. 여기에 지속적인 교육 및 자격증을 포함하는 것이 좋습니다. 이렇게 하면 조직에서 비용 및 사용량을 자체적으로 관리할 수 있는 역량을 갖출 수 있습니다.
- 무료로 제공되는 AWS 기본 도구(예: AWS Cost Anomaly Detection, AWS Budgets및 AWS Budgets 보고서)를 활용하세요..

조직에서 일관되게 클라우드 재무 관리 (CFM) 사례를 도입하면 이러한 행동이 작업 및 의사 결정 과정에 정착하게 됩니다. 그 결과, 새로운 클라우드 애플리케이션을 설계하는 개발자부터 새로운 클라우드 투자에 대한 ROI를 분석하는 재무 관리자까지 비용 인식 문화가 더 잘 자리 잡게 됩니다.

구현 단계

- 관련 조직의 프로세스 파악: 각 조직 단위에서 프로세스를 검토하고 비용 및 사용량에 영향을 미치는 프로세스를 파악합니다. 리소스가 생성 또는 종료되게 하는 모든 프로세스는 검토를 위해 포함해야 합니다. 인시던트 관리 및 교육과 같이 비즈니스에서 비용 인식을 지원할 수 있는 프로세스를 찾습니다.
- 스스로 지속 가능한 비용 인식 문화 확립: 관련된 모든 이해관계자가 변경의 원인 및 영향을 비용으로 인식하도록 하여 클라우드 비용을 이해할 수 있도록 합니다. 그러면 조직에서는 혁신을 위해 스스로 지속 가능한 비용 인식 문화를 확립할 수 있습니다.
- 비용 인식으로 프로세스 업데이트: 각 프로세스는 비용을 인식하도록 수정됩니다. 이 프로세스에서는 비용이 미치는 영향의 평가와 같은 추가 사전 점검이나 비용과 사용량에서 예상한 변화가 발생했는지 검증하는 사후 점검이 필요할 수 있습니다. 비용 및 사용량에 대한 항목을 포함하도록 교육 및 인시던트 관리와 같은 지원 프로세스를 확장할 수 있습니다.

도움을 받으려면 계정 팀을 통해 CFM 전문가에게 문의하거나 아래 리소스 및 관련 문서를 살펴보세요.

리소스

관련 문서:

- AWS 클라우드 재무 관리

관련 예시:

- 효율적인 클라우드 재무 관리를 위한 전략
- 비용 관리 블로그 시리즈 #3: 비용 충격을 처리하는 방법
- AWS Cost Management 초보자 가이드

COST01-BP05 비용 최적화 보고 및 알림

목표를 기준으로 비용 및 사용량 알림을 제공하도록 AWS Budgets 및 AWS Cost Anomaly Detection을(를) 구성합니다. 정기 회의를 통해 워크로드의 비용 효율성을 분석하고 비용 인식 문화를 장려합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

조직 내의 비용 및 사용량 최적화를 정기적으로 보고해야 합니다. 비용 최적화를 위한 전용 세션을 구현하거나 워크로드에 대한 정기적인 운영 보고 주기에 비용 최적화를 포함할 수 있습니다. 서비스와 도구를 사용하여 비용 절감 기회를 파악하여 구현합니다. AWS Cost Explorer 는 대시보드 및 보고서를 제공합니다. 다음을 사용하면 구성된 예산을 기준으로 비용 및 사용량의 진행 상태를 추적할 수 있습니다. AWS Budgets 보고서)를 활용하세요..

AWS Budgets 을(를) 사용하면 사용자 지정 예산을 설정하여 임곗값을 초과한 경우 비용 및 사용량을 추적하고 이메일로 받은 알림 또는 Amazon Simple Notification Service(Amazon SNS) 알림에 신속하게 대응할 수 있습니다. 기본 예산 기간을 매일, 매월, 매분기 또는 매년으로 설정하고 특정 예산 한도를 생성하여 실제 또는 예측 비용 및 사용량이 예산 임곗값을 향해 어떻게 진행되고 있는지 알림을 받을 수 있습니다. 또한 알림 및 해당 알림에 대한 작업이 자동으로 실행되거나 예산 목표가 초과된 경우에는 승인 프로세스를 통해 실행되도록 설정할 수도 있습니다.

예기치 못한 비용 및 사용량의 변화에 신속하게 대응할 수 있도록 비용 및 사용량에 대한 알림을 구현하세요. AWS Cost Anomaly Detection 을(를) 사용하면 갑작스럽게 발생하는 비용을 줄이고 혁신 속도

를 늦추지 않고 제어 기능을 개선할 수 있습니다. AWS Cost Anomaly Detection에서는 비정상적인 비용 및 근본 원인을 파악하여 갑작스러운 비용 청구 위험을 줄여줍니다. 간단한 3단계를 수행하면 맥락에 맞는 고유한 모니터링 시스템을 구축하고 이례적인 비용이 감지된 경우 알림을 받을 수 있습니다.

또한 Amazon QuickSight 을(를) AWS Cost and Usage Report(CUR) 데이터와 함께 사용하면 더욱 세분화된 데이터가 포함된 사용자 지정 보고서를 제공할 수 있습니다. Amazon QuickSight에서는 보고서를 예약하고 과거 비용 및 사용량 또는 비용 절감 기회에 대한 비용 보고서를 이메일을 통해 정기적으로 받을 수 있습니다.

AWS Trusted Advisor을(를) 사용합니다. 이는 프로비저닝된 리소스가 비용 최적화를 위한 AWS 모범 사례에 부합하는지 여부를 확인하는 지침을 제공합니다.

안정 상태 워크로드, 유휴 리소스 및 사용률이 낮은 리소스와 관련된 비용의 절감을 위한 절감형 플랜, 예약 인스턴스 및 AWS Cost Explorer의 Amazon Elastic Compute Cloud(Amazon EC2) 적정 크기 조정 권장 사항이 포함된 보고서를 주기적으로 생성합니다. 배포되는 리소스에 대한 클라우드 낭비와 관련된 비용을 파악하여 회수합니다. 크기가 잘못 지정된 리소스를 생성하거나 예상한 패턴 대신 다른 사용 패턴이 관찰되는 경우 클라우드 낭비가 발생합니다. AWS 모범 사례에 따라 클라우드 낭비를 줄이고 클라우드 비용을 최적화 및 절감 하세요.

더 나은 리소스 구매 옵션을 선택할 수 있도록 정기적으로 보고서를 생성하여 워크로드에 대한 단위 비용을 절감하세요. 절감형 플랜, 예약된 인스턴스 또는 Amazon EC2 스팟 인스턴스 등과 같은 구입 옵션은 내결함성 워크로드에서 가장 크게 비용을 절감할 수 있고 이해관계자(비즈니스 소유자, 재무 팀 및 기술 팀)가 이러한 약정 논의에 참여할 수 있습니다.

클라우드의 총 소유 비용(TCO)을 절감하는 데 도움이 될 수 있는 기회 또는 새로운 릴리스 발표 내용이 포함된 보고서를 제공합니다. 추가 비용 절감을 위해 새로운 서비스, 리전, 기능, 솔루션 또는 새로운 방법을 채택합니다.

구현 단계

- AWS Budgets 구성: 워크로드의 모든 계정에서 AWS Budgets을(를) 구성합니다. 태그를 사용하여 전체 계정 지출에 대한 예산과 워크로드에 대한 예산을 설정합니다.
  - Well-Architected 실습: 비용 및 사용 거버넌스
- 비용 최적화 보고: 워크로드의 효율성을 논의하고 분석하기 위한 규칙적인 주기를 설정합니다. 설정된 지표를 사용하여 달성된 지표와 지표 달성에 든 비용을 보고합니다. 부정적인 추세가 있다면 찾아서 수정하고 조직 전체에 장려할 수 있는 긍정적인 추세를 찾습니다. 보고에는 애플리케이션 팀 및 소유자, 재무 및 경영진의 담당자가 포함되어야 합니다.
  - Well-Architected 실습: 시각화

리소스

관련 문서:

- AWS Cost Explorer
- AWS Trusted Advisor
- AWS Budgets
- AWS Budgets 모범 사례
- Amazon CloudWatch
- AWS CloudTrail
- Amazon S3 분석
- AWS Cost and Usage Report

관련 예시:

- Well-Architected 실습: 비용 및 사용 거버넌스
- Well-Architected 실습: 시각화
- AWS 클라우드 비용 최적화를 시작하는 주요 방법

COST01-BP06 사전 예방적 비용 모니터링

도구 및 대시보드를 구현하여 워크로드에 대한 비용을 사전에 모니터링합니다. 알림을 받을 때만 비용 및 범주를 살펴보지 말고 구성된 도구 또는 바로 사용 가능한 도구와 관련된 비용을 정기적으로 검토합니다. 비용을 사전에 모니터링하고 분석하면 긍정적인 추세를 파악하여 조직 전체에서 해당 추세를 촉진할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

예외 또는 이상이 있을 뿐만이 아니라 조직 내에서 선제적으로 비용 및 사용량을 모니터링하는 것이 좋습니다. 사무실이나 업무 환경 전체에 가시성이 높은 대시보드를 구축하면 핵심 인력이 필요한 정보에 접근할 수 있을 뿐 아니라 조직이 비용 최적화에 중점을 두고 있음을 보여줄 수 있습니다. 가시성이 높은 대시보드를 사용하면 성공적인 결과를 적극적으로 홍보하고 조직 전체에 이를 구현할 수 있습니다.

따라서 AWS Cost Explorer 또는 기타 대시보드(예: Amazon QuickSight)를 사용하는 매일 또는 잦은 루틴을 만들어 사전에 비용을 살펴보고 분석할 수 있습니다. AWS 계정 수준, 워크로드 수준 또는 특

정 AWS 서비스 수준에서 그룹화와 필터링을 사용하여 AWS 서비스 사용량과 비용을 분석하고 해당 사용량과 비용이 예상된 것인지 확인합니다. 시간 수준 및 리소스 수준 세분화와 태그를 사용하여 가장 많이 사용하는 리소스에서 발생하는 비용을 필터링하고 식별할 수 있습니다. 또한 Cost Intelligence Dashboard(Amazon QuickSight 솔루션, AWS Solutions Architects로 구축)를 사용하여 고유한 보고서를 작성하고 실제 비용 및 사용량과 예산을 비교할 수 있습니다.

구현 단계

- 비용 최적화 보고: 워크로드의 효율성을 논의하고 분석하기 위한 규칙적인 주기를 설정합니다. 설정된 지표를 사용하여 달성된 지표와 지표 달성에 든 비용을 보고합니다. 부정적인 추세가 있다면 찾아서 수정하고 조직 전체에 장려할 수 있는 긍정적인 추세를 찾습니다. 보고에는 애플리케이션 팀 및 소유자, 재무 및 경영진의 담당자가 포함되어야 합니다.
- 비용 초과 가능성을 예방하기 위해 적시에 조치를 취할 수 있도록 비용과 사용량에 대해 일 단위 AWS Budgets 생성 및 사용: AWS Budgets을(를) 사용하면 알림을 구성할 수 있어 예산 유형이 미리 구성된 임곗값을 벗어난 경우 알림을 받을 수 있습니다. AWS Budgets을(를) 제대로 활용하려면 기대 비용과 사용량을 한도로 설정해 두는 것이 좋습니다. 그러면 예산을 초과하는 모든 항목을 초과 지출로 간주할 수 있습니다.
- 비용 모니터링을 위한 AWS Cost Anomaly Detection 생성: AWS Cost Anomaly Detection 은(는) 고급 기계 학습 기술을 사용하여 비정상적인 지출과 근본 원인을 식별하므로 빠르게 조치를 취할 수 있습니다. 또한 평가하고 싶은 지출 세그먼트(예: 개별 AWS 서비스, 멤버 계정, 비용 할당 태그 및 비용 범주)를 정의하는 비용 모니터링을 구성할 수 있고 알림을 받는 경우와 위치 그리고 방법을 설정할 수 있습니다. 각 모니터링에 이름, 비용 영향 임곗값, 각 구독에 대한 알림 빈도(개별 알림, 일일 요약, 주별 요약)를 비롯하여 비즈니스 소유자와 기술 팀에 대한 여러 알림 구독을 첨부할 수 있습니다.
- AWS Cost Explorer을(를) 사용하거나 AWS Cost and Usage Report(CUR) 데이터를 Amazon QuickSight 대시보드와 통합하여 조직의 비용 시각화: AWS Cost Explorer에는 시간에 따라 AWS 비용과 사용량을 시각화하고 이해하며 관리할 수 있는 사용하기 쉬운 인터페이스가 있습니다. Cost Intelligence Dashboard 는 사용자 지정 및 액세스 가능한 대시보드로, 고유한 비용 관리 및 최적화 도구의 기반을 만드는 데 유용합니다.

리소스

관련 문서:

- AWS Budgets
- AWS Cost Explorer
- 일일 비용 및 사용 예산
- AWS Cost Anomaly Detection

관련 예시:

- Well-Architected 실습: 시각화
- Well-Architected 실습: 고급 시각화
- Well-Architected 실습: Cloud Intelligence 대시보드
- Well-Architected 실습: 비용 시각화
- 슬랙을 통한 AWS Cost Anomaly Detection 알림

COST01-BP07 새로운 서비스 릴리스로 최신 상태 유지

정기적으로 전문가 또는 AWS 파트너의 상담을 받아 더 저렴한 가격을 제공하는 서비스와 기능을 고려합니다. AWS 블로그와 기타 정보 출처를 검토합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

AWS은(는) 새로운 기능을 지속적으로 추가하고 있습니다. 따라서 최신 기술을 활용하여 더욱 신속하게 실험하고 혁신할 수 있습니다. 새로운 AWS 서비스와 기능을 구현하여 워크로드의 비용 효율성을 높일 수 있습니다. 주기적으로 AWS 비용 관리, AWS 뉴스 블로그, AWS 비용 관리 블로그및 AWS의 새로운 소식 을 검토하여 새로운 서비스 및 기능 릴리스에 대한 정보를 확인하십시오. 새로운 소식 게시물은 모든 AWS 서비스, 기능 및 리전 확장이 발표되면 해당 내용을 간략하게 안내합니다.

구현 단계

- 블로그 구독: AWS 블로그 페이지를 방문하고 새로운 소식 블로그와 기타 관련 블로그를 구독합니다. 이메일 주소를 사용하여 커뮤니케이션 기본 설정 페이지에 등록할 수 있습니다.
- AWS 뉴스 구독: 주기적으로 AWS 뉴스 블로그 및 AWS의 새로운 소식 을 검토하여 새로운 서비스 및 기능 릴리스에 대한 정보를 확인하십시오. RSS 피드를 구독하거나 이메일을 사용하여 발표 및 릴리스 소식을 팔로우할 수 있습니다.
- AWS 가격 인하 팔로우: 스케일 조정에서 얻은 경제적 효율성을 고객에게 되돌려 드리기 위해 AWS에서 실시하는 모든 서비스의 정기적인 가격 인하는 표준으로 자리를 잡았습니다. 2022년 4월을 기준으로 AWS은(는) 2006년 처음 운영을 시작한 이래 총 115회 가격을 인하했습니다. 가격 때문에 결정을 보류하고 있다면 가격 인하와 새로운 서비스 통합을 적용한 후 다시 검토할 수 있습니다. Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 비롯하여 이전부터 지속된 가격 인하를 위한 노력은 AWS 뉴스 블로그의 가격 인하 카테고리에서 살펴볼 수 있습니다..

- AWS 이벤트 및 모임: 현지 AWS 서밋과 해당 지역의 다른 조직과의 모든 현지 회의에 참석합니다. 직접 참석할 수 없는 경우 가상 이벤트에 참석하여 AWS 전문가와 기타 고객의 비즈니스 사례에 대해 자세히 들어보세요.
- 계정 팀과의 만남: 계정 팀과 정기적인 회의를 예약하고 이들과 만나 업계 동향과 AWS 서비스에 대해 논의합니다. 계정 관리자, 솔루션 아키텍트 및 지원 팀과 이야기합니다.

리소스

관련 문서:

- AWS 비용 관리
- AWS의 새로운 소식
- AWS 뉴스 블로그

관련 예시:

- Amazon EC2 – IT 비용 최적화 및 절감을 위한 15년
- AWS 뉴스 블로그 - 가격 인하

COST01-BP08 비용 인식 문화 조성

조직 전체에 비용 인식 문화를 조성하기 위한 변화를 주도하거나 이러한 프로그램을 구현합니다. 소규모로 시작한 후 역량이 커지고 조직의 클라우드 사용이 증가함에 따라 대규모 프로그램을 구현하는 것이 좋습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

비용 인식 문화가 조성되면 조직 전체에서 유기적이고 분산된 방식으로 수행되는 모범 사례를 통해 비용 최적화 및 클라우드 비용 관리(재무 운영, 클라우드 혁신 센터, 클라우드 운영 팀 등)를 확장할 수 있습니다. 비용 인식을 통해 엄격한 하향식의 중앙 집중식 접근 방식에 비해 최소한의 노력으로 조직 전체의 역량을 강화할 수 있습니다.

클라우드 컴퓨팅에서 특히, 클라우드 컴퓨팅의 1차 비용 요인에 대한 비용 인식을 통해 팀에서는 모든 변경의 예상 결과를 비용 관점에서 이해할 수 있습니다. 클라우드 환경에 액세스하는 팀은 가격 모델과 전통적인 온프레미스 데이터 센터와 클라우드 컴퓨팅 간의 차이를 알고 있어야 합니다.

비용 인식 문화의 주요 이점은 기술 팀이 상황이 발생한 후 필요에 따라 비용 최적화를 대응적으로 수행하는 대신 지속적으로 사전에 비용을 최적화한다는 점입니다(예: 비용은 새로운 워크로드를 설계하거나 기존 워크로드를 변경할 때 기술적 요구 사항이 아닌 것으로 간주됨).

문화의 작은 변화는 현재 및 향후 워크로드의 효율성에 큰 영향을 미칠 수 있습니다. 예를 들면 다음과 같습니다.

- 비용 관점에서 역할과 미치는 영향을 파악할 수 있도록 엔지니어링 팀 내 가시성 확보 및 인식 확립
- 조직 전체 비용 및 사용량의 게임화. 공개적으로 확인 가능한 대시보드 또는 팀 전체의 정규화된 비용 및 사용량을 비교하는 보고서(예: 워크로드당 비용, 트랜잭션당 비용)를 사용하여 게임화할 수 있습니다.
- 비용 효율성 인식. 자발적 또는 임의의 비용 최적화 성과를 공개적으로 또는 비공개적으로 보상하고, 실수를 통한 교훈으로 향후 재발을 방지합니다.
- 워크로드에 대한 조직의 하향식 요구 사항을 생성하여 미리 정의된 예산을 실행합니다.
- 필요한 만큼만 지불할 수 있도록 변경에 대한 비즈니스 요구 사항과 아키텍처 인프라 또는 워크로드 구성에 대해 요청된 변경 사항이 미치는 비용 영향에 대해 질문하세요.
- 변경 계획자가 비용에 영향을 미치는 예상 변경 사항을 파악하고 비즈니스 결과를 비용 효율적으로 전달할 수 있도록 이해관계자가 확인하도록 하세요.

구현 단계

- 기술 팀에 클라우드 비용 보고: 재무 및 비즈니스 이해관계자를 위해 비용 인식을 높이고 효율성 KPI를 확립하기 위함입니다.
- 이해관계자 또는 팀원에게 계획된 변경에 대해 알리기: 주간 변경 회의 중 계획된 변경 및 워크로드에 대한 비용 이점에 대해 논의할 수 있도록 소개 항목을 생성합니다.
- 계정 팀과의 만남: 계정 팀과 정기적인 회의 주기를 확립하고 업계 동향과 AWS 서비스에 대해 논의합니다. 계정 관리자, 아키텍트 및 지원 팀과 이야기합니다.
- 성공 사례 공유: 긍정적인 태도 및 비용 최적화 권장을 위해 워크로드, AWS 계정 또는 조직에 대한 비용 절감 성공 사례를 공유합니다.
- 교육: 기술 팀 또는 팀원이 AWS 클라우드에 대한 리소스 비용 인식의 교육을 받도록 합니다.
- AWS 이벤트 및 모임: 현지 AWS 서밋과 해당 지역의 다른 조직과의 모든 현지 회의에 참석합니다.
- 블로그 구독: AWS 블로그 페이지를 방문하여 ‘새로운 소식’ 블로그 및 기타 관련 블로그를 구독하여 AWS에서 공유한 새로운 릴리스, 구현, 예제, 변경 사항을 팔로우해 보세요.

리소스

관련 문서:

- AWS 블로그
- AWS 비용 관리
- AWS 뉴스 블로그

관련 예시:

- AWS 클라우드 재무 관리
- AWS Well-Architected 실습: 클라우드 재무 관리

COST01-BP09 비용 최적화의 비즈니스 가치 정량화

비용 최적화의 비즈니스 가치를 정량화하면 조직에 대한 전체적인 이점을 이해할 수 있습니다. 비용 최적화는 필요한 투자이므로 비즈니스 가치를 정량화하면 이해관계자에게 투자 수익률을 설명할 수 있습니다. 비즈니스 가치를 정량화하면 향후 비용 최적화 투자에 대해 이해관계자로부터 더 많은 지지를 얻는 데 도움이 되며, 조직의 비용 최적화 활동에 대한 결과를 측정하는 프레임워크로 사용될 수 있습니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

비용 최적화의 절감 효과를 보고하는 것에 더해 제공된 추가 가치를 정량화하는 것이 좋습니다. 비용 최적화의 이점은 일반적으로 각 비즈니스 결과에 대한 비용 절감이라는 측면에서 정량화됩니다. 예를 들어 저렴한 비용으로 워크로드 결과 수준을 유지할 수 있는 절감형 플랜을 구매하는 경우 온디맨드 Amazon Elastic Compute Cloud(Amazon EC2) 비용 절감을 정량화할 수 있습니다. 유휴 Amazon EC2 인스턴스가 종료되거나 연결되지 않은 Amazon Elastic Block Store(Amazon EBS) 볼륨이 삭제될 때 AWS 지출의 비용 감소를 정량화할 수 있습니다.

그러나 비용 최적화는 비용 절감 또는 회피 이상의 이점을 제공합니다. 효율성 개선 및 비즈니스 가치를 측정하는 추가 데이터를 캡처하는 것이 좋습니다.

구현 단계

- 비용 최적화 모범 사례 실행: 예를 들어 리소스 수명 주기를 관리하면 인프라 및 운영 비용이 절감되며 실험을 위한 시간 및 예상 밖의 예산이 창출됩니다. 이렇게 하면 조직의 민첩성이 개선되고 수익 창출을 위한 새로운 기회가 열립니다.
- 자동화 구현: 예를 들어 Auto Scaling은 최소한의 노력으로 탄력성을 보장하고 수동 용량 계획 작업을 제거하여 직원의 생산성을 높여줍니다. 운영 복원력에 대한 자세한 내용은 다음 백서를 참조하십시오. Well-Architected 안정성 원칙 백서.
- 향후 AWS 비용 예측: 재무 관계자는 예측을 통해 다른 내부 및 외부 조직의 이해관계자의 기대치를 설정하고 조직의 재무 예측 기능을 개선할 수 있습니다. AWS Cost Explorer를 사용하여 비용 및 사용량을 예측할 수 있습니다.

리소스

관련 문서:

- AWS 블로그
- AWS 비용 관리
- AWS 뉴스 블로그
- Well-Architected 안정성 원칙 백서
- AWS Cost Explorer

지출 및 사용량 인식

질문

- COST 2 사용량을 어떻게 관리합니까?
- COST 3 사용량과 비용을 어떻게 모니터링합니까?
- COST 4 리소스를 어떻게 폐기합니까?

COST 2 사용량을 어떻게 관리합니까?

목표 달성 과정에서 발생하는 비용을 적정 수준으로 유지하는 정책과 메커니즘을 설정합니다. '견제와 균형' 방식을 도입하면 비용을 과도하게 지출하지 않고 혁신을 이룰 수 있습니다.

모범 사례

- COST02-BP01 조직 요구 사항에 따라 정책 개발
- COST02-BP02 목표 및 타겟 이행

- COST02-BP03 계정 구조 구현
- COST02-BP04 그룹 및 역할 만들기
- COST02-BP05 비용 제어 기능 구현
- COST02-BP06 프로젝트 수명 주기 추적

COST02-BP01 조직 요구 사항에 따라 정책 개발

조직에서 리소스를 관리하는 방법을 정의하는 정책을 개발하고 정기적으로 검사합니다. 정책에서는 리소스 수명 주기 동안의 생성, 수정, 폐기를 포함한 리소스 및 워크로드의 비용 측면을 다루어야 합니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 높음

구현 가이드

비용 및 사용량을 효과적으로 관리하고 비용 절감 기회를 파악하려면 조직의 비용 및 동인을 파악하는 것이 중요합니다. 조직에서는 일반적으로 여러 팀이 운영하는 여러 워크로드를 운영합니다. 이러한 팀은 매출원이 각기 다른 개별 조직 단위 소속일 수 있습니다. 워크로드, 개별 조직 또는 제품 책임자에게 리소스 비용을 귀속하는 기능은 효율적인 사용 행동으로 이어지고 낭비되는 요소를 줄여줍니다. 정확한 비용 및 사용량 모니터링을 통해 워크로드가 얼마나 최적화되었는지, 조직 단위 및 제품의 수익성이 어느 정도인지 이해할 수 있습니다. 이를 통해 조직 내에서 리소스를 할당할 위치에 대해 더 많은 정보를 활용하여 의사 결정을 내릴 수 있습니다. 비용은 사용량에 따라 변경되므로 비용 변경을 주도하려면 조직의 모든 수준에서 사용량을 인식하는 것이 필수적입니다. 사용량 및 지출을 적절하게 파악하려면 다각적 방식을 사용하는 것이 좋습니다.

거버넌스를 수행하는 첫 번째 단계는 조직의 요구 사항을 활용하여 클라우드 사용에 대한 정책을 개발하는 것입니다. 이러한 정책은 조직에서 클라우드를 사용하는 방법과 리소스를 관리하는 방법을 정의합니다. 정책에서는 리소스 수명 주기 동안의 생성, 수정, 폐기를 비롯하여 비용 또는 사용량과 관련된 워크로드의 모든 측면을 다루어야 합니다. 정책과 절차를 준수하고 클라우드 환경의 모든 변화에 대비하여 구현되었는지 확인합니다. IT 변경 관리 회의에서 계획된 변경 사항의 비용 영향, 즉 증가 및 감소 여부, 비즈니스 타당성, 예상 결과에 대해 질문합니다.

정책은 조직 전체에서 쉽게 이해할 수 있고 효과적으로 구현할 수 있을 만큼 단순해야 합니다. 또한 정책은 준수 및 해석하기 쉬워야 하고(그래야 정책이 사용될 수 있음) 구체적이어야 합니다(그래야 팀 간에 오해가 없음). 또한 당사의 메커니즘과 같이 정기적으로 점검하고 고객의 비즈니스 상황 또는 우선 순위가 바뀌면 정책이 상황에 맞지 않을 수 있으므로 업데이트해야 합니다.

사용할 지리적 리전, 리소스를 실행해야 하는 하루 중 시간 등 넓은 범위의 상위 수준 정책부터 시작합니다. 그런 다음 다양한 조직 단위 및 워크로드에 대한 정책을 점진적으로 구체화합니다. 가장 많이 사

용하는 정책에는 사용할 수 있는 서비스와 기능(예: 테스트 및 개발 환경의 비교적 성능이 낮은 스토리지), 각 그룹에서 사용할 수 있는 리소스 유형(예: 개발 계정에서 사용할 수 있는 가장 큰 리소스 크기는 중간 규모), 리소스 사용 시간(임시, 단기 또는 특정 기간)이 있습니다.

정책 예제

다음은 비용 최적화에 중점을 둔 자체 클라우드 거버넌스 정책을 작성하기 위해 검토할 수 있는 샘플 정책입니다. 조직의 요구 사항과 이해 관계자의 요청에 따라 정책을 조정해야 합니다.

- 정책 이름: 리소스 최적화 및 비용 절감 정책과 같은 명확한 정책 이름을 정의합니다.
- 목적: 이 정책을 사용해야 하는 이유와 예상되는 결과를 설명합니다. 이 정책의 목적은 비즈니스 요구 사항을 충족하기 위해 원하는 워크로드를 배포하고 실행하는 데 필요한 최소 비용이 있는지 확인하는 것입니다.
- 범위: 이 정책을 누가 언제 사용해야 하는지 명확하게 정의합니다. 예를 들어 DevOps X Team은 X 환경(프로덕션 또는 비프로덕션)에서 미국 동부 고객에게 이 정책을 사용해야 합니다.

정책 문

1. 워크로드의 환경 및 비즈니스 요구 사항(개발, 사용자 승인 테스트, 사전 프로덕션 또는 프로덕션)에 따라 us-east-1 또는 여러 개의 미국 동부 리전을 선택합니다.
2. 오전 6시~오후 8시(동부 표준시(EST)) 사이에 실행되도록 Amazon EC2 및 Amazon RDS 인스턴스를 예약합니다.
3. 8시간 동안 활동이 없으면 사용하지 않는 모든 Amazon EC2 인스턴스를 중지하고 24시간 동안 활동이 없으면 사용하지 않는 Amazon RDS 인스턴스를 중지합니다.
4. 비프로덕션 환경에서는 24시간 동안 활동이 없으면 사용하지 않는 모든 Amazon EC2 인스턴스를 종료합니다. Amazon EC2 인스턴스 소유자(태그 기준)에게 프로덕션 환경에서 중지된 Amazon EC2 인스턴스를 검토하도록 요청하고 사용하지 않을 경우 72시간 이내에 해당 Amazon EC2 인스턴스가 종료될 것임을 알립니다.
5. 일반 인스턴스 패밀리 및 크기(예: m5.large)를 사용한 다음 AWS Compute Optimizer를 사용하여 CPU 및 메모리 사용률에 따라 인스턴스 크기를 조정합니다.
6. Auto Scaling을 사용하여 우선순위를 지정해 트래픽에 따라 실행 중인 인스턴스 수를 동적으로 조정합니다.
7. 중요하지 않은 워크로드에는 스팟 인스턴스를 사용합니다.
8. 용량 요구 사항을 검토하여 예측 가능한 워크로드를 위한 절감형 플랜 또는 예약 인스턴스를 약정하고 클라우드 재무 관리 팀에 알립니다.

9. Amazon S3 수명 주기 정책을 사용하여 자주 액세스하지 않는 데이터를 저렴한 스토리지 계층으로 이동합니다. 보존 정책이 정의되지 않은 경우 Amazon S3 Intelligent Tiering을 사용하여 객체를 자동으로 아카이브 계층으로 이동합니다.
10. Amazon CloudWatch를 사용하여 리소스 사용률을 모니터링하고 경보를 설정하여 규모 조정 이벤트를 트리거합니다.
11. 각 AWS 계정에 대해 AWS Budgets를 사용하여 비용 센터 및 사업부를 기준으로 계정의 비용 및 사용 예산을 설정합니다.
12. 계정의 비용 및 사용 예산을 설정하는 데 AWS Budgets를 사용하면 지출을 확실히 파악하고 예상치 못한 청구서를 피할 수 있으므로 비용을 더 잘 관리할 수 있습니다.

절차: 이 정책을 구현하기 위한 세부 절차를 제공하거나 각 정책 문을 구현하는 방법을 설명하는 다른 문서를 소개합니다. 이 섹션에서는 정책 요구 사항을 수행하기 위한 단계별 지침을 제공해야 합니다.

이 정책을 구현하려면 다양한 타사 도구 또는 AWS Config 규칙을 사용하여 정책 문 준수 여부를 확인하고 AWS Lambda 함수를 사용하여 자동화된 수정 작업을 트리거할 수 있습니다. AWS Organizations를 사용하여 정책을 적용할 수도 있습니다. 또한 정기적으로 리소스 사용량을 검토하고 필요에 따라 정책을 조정하여 비즈니스 요구 사항을 계속 충족하는지 확인해야 합니다.

구현 단계

- 이해 관계자와 회의: 정책을 개발하려면 조직 내 이해 관계자(클라우드 비즈니스 오피스, 엔지니어 또는 정책 시행을 위한 기능적 의사 결정권자)에게 요구 사항을 지정하고 문서화하도록 요청합니다. 광범위하게 시작하여 반복적인 접근 방식을 취하고 각 단계에서 가장 작은 단위까지 계속 세분화합니다. 팀원에는 조직 단위 또는 애플리케이션 소유자와 같이 워크로드에 직접적인 관심이 있는 멤버뿐만 아니라 보안 및 재무 팀과 같은 지원 그룹이 포함됩니다.
- 확인: 각 팀이 AWS 클라우드에 액세스하고 배포할 수 있는 정책에 동의하는지 확인합니다. 각 팀이 조직 정책을 준수하고, 리소스 생성이 합의된 정책 및 절차에 적합한지 확인합니다.
- 온보딩 교육 세션 만들기: 새로운 조직 구성원에게 온보딩 교육 과정을 이수하도록 하여 비용과 조직 요구 사항에 대해 인식하도록 합니다. 이전의 경험과 정책이 다를 것으로 가정하거나, 전혀 생각하고 있지 않을 수 있습니다.
- 워크로드 위치 정의: 국가와 국가 내 지역을 포함한 워크로드의 작동 위치를 정의합니다. 이 정보는 AWS 리전 및 가용 영역에 매핑하는 데 사용됩니다.
- 서비스와 리소스 정의 및 그룹화: 워크로드에 필요한 서비스를 정의합니다. 서비스마다 필요한 리소스 유형, 크기 및 수를 지정합니다. 애플리케이션 서버나 데이터베이스 스토리지와 같은 기능별로 리소스 그룹을 정의합니다. 리소스는 여러 그룹에 속할 수 있습니다.

- 역할별로 사용자 정의 및 그룹화: 누구인지 또는 조직에서 어떤 직책을 맡고 있는지가 아니라 무슨 일을 하며 워크로드를 어떻게 사용하는지에 초점을 두고 워크로드와 밀접한 일을 하는 사용자를 정의합니다. 유사한 사용자나 역할을 함께 그룹화합니다. AWS 관리형 정책을 가이드로 사용할 수 있습니다.
- 작업 정의: 이전에 식별된 위치, 리소스, 사용자를 사용하여 수명 주기(개발, 운영 및 폐기) 동안 각자 워크로드 성과를 거두는 데 필요한 작업을 정의합니다. 각 위치에서 그룹의 개별 요소가 아니라 그룹을 기반으로 작업을 식별합니다. 읽기 또는 쓰기로 광범위하게 시작한 다음 각 서비스에 대한 특정 작업까지 세분화합니다.
- 검토 기간 정의: 워크로드와 조직 요구 사항은 시간이 지나면서 바뀔 수 있습니다. 조직의 우선 순위와 일치하도록 워크로드 검토 일정을 정의합니다.
- 정책 문서화: 정의된 정책에 조직의 필요에 따라 액세스할 수 있는지 확인합니다. 이러한 정책은 환경의 액세스를 구현, 유지 관리, 감사하는 데 사용됩니다.

리소스

관련 문서:

- 클라우드에서 변경 관리
- 직무 역할에 대한 AWS 관리형 정책
- AWS 다중 계정 결제 전략
- AWS 서비스에 사용되는 작업, 리소스 및 조건 키
- AWS 관리 및 거버넌스
- IAM 정책을 사용하여 AWS 리전에 대한 액세스 제어
- 글로벌 인프라 리전 및 AZ

관련 동영상:

- 대규모 AWS 관리 및 거버넌스

관련 예시:

- VMware - 클라우드 정책이란 무엇일까요?

COST02-BP02 목표 및 타겟 이행

워크로드에 대한 비용 및 사용량 목표와 타겟을 모두 이행합니다. 목표는 예상 결과에 대한 조직의 방향성을 제공하고, 타겟은 워크로드에 대해 달성할 수 있는 구체적인 측정 가능한 결과를 제공합니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 높음

구현 가이드

조직의 비용과 사용량에 대한 목표와 타겟을 설정합니다. AWS에서 성장하는 조직으로서 비용 최적화를 위한 목표를 설정하고 추적하는 것이 중요합니다. 이러한 목표 또는 핵심 성과 지표(KPI) 에는 온디맨드 지출 비율이나 AWS Graviton 인스턴스 또는 gp3 EBS 볼륨 유형과 같은 최적화된 서비스의 채택 등이 포함될 수 있습니다. 측정 가능하고 달성 가능한 목표를 설정하면 지속적인 비즈니스 운영에 있어 중요한 효율성 개선이 얼마나 이루어졌는지 계속해서 측정하는 데 도움이 될 수 있습니다. 목표를 통해 조직은 원하는 성과를 달성하는데 필요한 지침을 얻고 나아갈 방향을 알 수 있습니다. 타겟을 통해 달성해야 할 구체적이고 측정 가능한 결과를 파악할 수 있습니다. 간단히 말해 목표는 나아가고자 하는 방향이고 타겟은 그 방향과의 거리와 목표 달성 시점에 해당합니다(SMART(구체성, 측정 가능성, 할당성, 현실성, 적시성) 지침 활용). 예를 들어, 목표는 비용은 조금(비선형) 늘리면서 플랫폼 사용량은 크게 늘리는 것입니다. 타겟은 플랫폼 사용량을 20% 늘리고 비용 증가를 5% 미만으로 유지하는 것입니다. 워크로드 효율성을 6개월마다 개선하는 것은 일반적인 목표의 또 다른 예입니다. 이에 따른 타겟은 비즈니스당 비용 지표를 6개월마다 5%씩 줄이는 것이 될 수 있습니다.

비용 최적화의 목표는 워크로드 효율성을 높이는 것이며, 이는 시간이 지남에 따라 워크로드의 비즈니스 결과당 비용을 줄이는 것을 의미합니다. 모든 워크로드에 대해 이 목표를 구현하고 6~12개월마다 효율성 5% 증가와 같은 타겟을 설정하는 것이 좋습니다. 클라우드에서는 비용 최적화 관련 기능을 구축하고 새로운 서비스 및 기능을 릴리스하여 이를 달성할 수 있습니다.

KPI 및 관련 비용 절감 기회를 거의 실시간으로 파악하고 시간 경과에 따른 진행 상황을 추적하는 것이 중요합니다. KPI 목표 정의 및 추적을 시작하려면 클라우드 인텔리전스 대시보드(CID) 프레임워크의 KPI 대시보드를 사용하는 것이 좋습니다. AWS Cost and Usage Report의 데이터를 기반으로 KPI 대시보드는 맞춤형 목표를 설정하고 시간 경과에 따른 진행 상황을 추적할 수 있는 일련의 권장 비용 최적화 KPI를 제공합니다.

KPI 목표를 설정하고 추적할 수 있는 다른 솔루션이 있다면 조직의 모든 클라우드 재무 관리 이해 관계자가 이를 채택하도록 해야 합니다.

구현 단계

- 예상 사용량 수준 정의: 먼저 사용량 수준에 초점을 맞추십시오. 애플리케이션 소유자, 마케팅 팀, 규모가 큰 비즈니스 팀과 협력하여 워크로드의 예상 사용량 수준을 파악합니다. 시간 경과에 따라 고객

수요는 어떻게 바뀔 것이며, 계절에 따른 증가나 마케팅 캠페인으로 인한 변화가 있을지도 알아봅니다.

- 워크로드 리소싱 및 비용 정의 사용량 수준을 정의한 후 이러한 사용량 수준을 충족하는 데 필요한 워크로드 리소스의 변경 사항을 수량화합니다. 워크로드 구성 요소의 리소스 크기 또는 수를 늘리거나, 데이터 전송을 늘리거나, 워크로드 구성 요소를 특정 수준의 다른 서비스로 변경해야 할 수 있습니다. 이러한 각 주요 지점에서 비용은 어떻게 될 것이며, 사용량에 변화가 있을 때 비용은 어떻게 바뀔지 지정합니다.
- 비즈니스 목표 정의: 예상 사용량과 비용 변화의 결과를 예상되는 기술 변화나 실행 중인 모든 프로그램과 결합하고 워크로드에 대한 목표를 설정합니다. 목표에서는 사용량과 비용, 그 둘의 관계를 다루어야 합니다. 목표는 간단하고 간략하며, 기업에서 어떤 결과를 원하는지 이해하는 데 도움이 되어야 합니다(예: 사용하지 않은 리소스는 특정 비용 수준 미만으로 유지). 각각의 사용하지 않은 리소스 유형에 대해 목표를 정의하거나 목표와 타겟에 손실을 초래하는 비용을 정의할 필요가 없습니다. 사용량 변화 없이 비용 변화만 예상되는 경우, 조직 프로그램(예: 훈련 및 교육을 통한 역량 쌓기)이 있는지 확인합니다.
- 타겟 정의: 정의된 각 목표에 대해 측정 가능한 타겟을 지정합니다. 워크로드의 효율성을 높이는 것이 목표라면, 타겟은 개선 수치(일반적으로 소비한 USD당 비즈니스 성과)와 달성 시점을 정량화합니다. 예를 들어 과도한 프로비저닝으로 인해 발생하는 낭비를 최소화하는 목표를 세운 경우, 프로덕션 워크로드의 첫 번째 티어에서 과도한 컴퓨팅 프로비저닝으로 인한 낭비가 티어 컴퓨팅 비용의 10%를 초과하지 않도록 하고 프로덕션 워크로드의 두 번째 티어에서 발생하는 과도한 컴퓨팅 프로비저닝이 티어 컴퓨팅 비용의 5%를 초과하지 않도록 하는 것을 타겟으로 지정할 수 있습니다.

리소스

관련 문서:

- 직무 역할에 대한 AWS 관리형 정책
- AWS Control Tower 랜딩 존에 대한 AWS 다중 계정 전략
- IAM 정책을 사용하여 AWS 리전에 대한 액세스 제어
- SMART 목표

관련 동영상:

- Well-Architected 실습: 목표 및 타겟(레벨 100)

관련 예시:

- Well-Architected 실습: 리소스 폐기(목표 및 타겟)
- Well-Architected 실습: 리소스 유형, 규모 및 숫자(목표 및 타겟)

COST02-BP03 계정 구조 구현

조직에 적합한 계정 구조를 구현합니다. 이를 통해 조직 전체에서 비용을 쉽게 할당하고 관리할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

AWS Organizations를 사용하여 다수의 AWS 계정을 생성할 수 있으며, 이를 통해 AWS 기반 워크로드가 확장함에 따라 환경을 중앙에서 통제할 수 있습니다. 조직 단위(OU) 구조로 AWS 계정을 그룹화하고 각 OU에 다수의 AWS 계정을 생성하여 조직 계층 구조를 모델링할 수 있습니다. 계정 구조를 생성하려면 어떤 AWS 계정이 관리 계정인지 먼저 결정해야 합니다. 그 후 관리 계정 모범 사례 및 멤버 계정 모범 사례를 따라 설계된 계정 구조를 기반으로 새 AWS 계정을 생성하거나 기존 계정을 멤버 계정으로 선택할 수 있습니다.

이때 조직의 규모나 사용량에 관계없이 하나의 멤버 계정이 연결된 하나 이상의 관리 계정을 항상 보유하는 것이 좋습니다. 모든 워크로드 리소스는 멤버 계정 내에만 상주해야 하며 관리 계정에는 리소스를 생성하면 안 됩니다. 필요한 AWS 계정 수는 경우에 따라 다릅니다. 현재 및 향후의 운영 모델과 비용 모델을 평가하여 AWS 계정 구조가 조직의 목표를 반영하는지 확인해야 합니다. 일부 회사에서는 비즈니스 이유로 다음과 같은 여러 AWS 계정를 생성합니다.

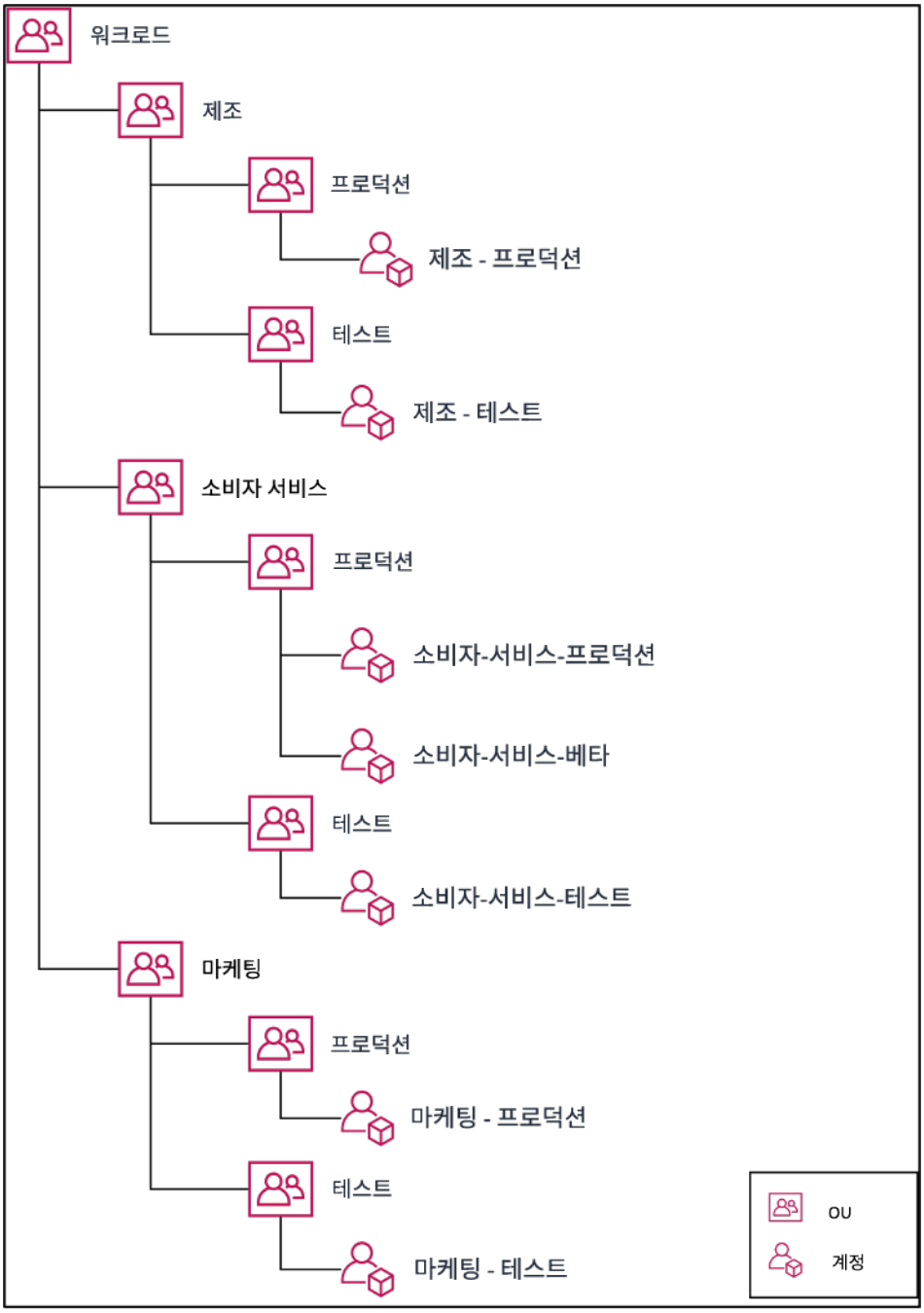
- 조직 단위, 비용 센터 또는 특정 워크로드 간에 관리 또는 재무 및 결제 작업을 분리해야 하는 경우
- 특정 워크로드별로 AWS 서비스 한도가 설정되어 있는 경우
- 워크로드와 리소스 간에 격리 및 분리에 대한 요구 사항이 있는 경우

AWS Organizations에서 통합 결제를 사용하면 하나 이상의 멤버 계정과 관리 계정 간의 구성이 생성됩니다. 멤버 계정을 사용하면 비용과 사용량을 그룹으로 분리하고 구별할 수 있습니다. 각 조직 단위(재무, 마케팅, 영업 등) 또는 각 환경 수명 주기(개발, 테스트, 프로덕션 등) 또는 각 워크로드(워크로드 a, b, c)용으로 별도의 멤버 계정을 생성한 다음 통합 결제를 사용하여 이러한 연결 계정을 집계하는 방식이 흔히 사용됩니다.

통합 결제에서는 여러 멤버 AWS 계정의 결제를 하나의 관리 계정에 통합할 수 있으며, 각 연결 계정의 활동은 계속 확인할 수 있습니다. 관리 계정에 비용 및 사용량이 집계되므로 서비스 대량 구매 할인율

을 극대화하고 약정 할인(절감형 플랜 및 예약 인스턴스)을 최대한 활용하여 가장 높은 할인을 받을 수 있습니다.

다음 다이어그램은 조직 단위(OU)로 AWS Organizations를 사용하여 다양한 계정을 그룹화하고 각 OU에 다양한 AWS 계정을 배치하는 방법을 보여줍니다. 계정 구성을 위한 패턴을 제공하는 다양한 사용 사례 및 워크로드에는 OU를 사용하는 것이 좋습니다.



조직 단위에 다양한 AWS 계정를 그룹화하는 예

AWS Control Tower를 사용하면 여러 AWS 계정을 빠르게 설정하고 구성하여 조직의 요구 사항에 부합하는 거버넌스를 구현할 수 있습니다.

구현 단계

- 분리 요구 사항 정의: 분리에 대한 요구 사항은 보안, 신뢰성, 재무 구조와 같은 여러 요인을 결합한 것입니다. 각 요인을 순서대로 살펴보고 워크로드 또는 워크로드 환경이 다른 워크로드와 분리되어야 하는지 여부를 지정합니다. 보안은 액세스 및 데이터 요구 사항에 대한 준수를 촉진합니다. 신뢰성은 한도를 관리하여 환경과 워크로드가 다른 요인에 영향을 미치지 않도록 합니다. Well-Architected 프레임워크의 보안과 신뢰성 원칙을 주기적으로 검토하고 제공된 모범 사례를 따릅니다. 재무 구조는 재무를 엄격하게 분리합니다(각기 다른 비용 센터, 워크로드 소유권 및 책임). 분리의 일반적인 예로는 별도의 계정에서 실행 중인 프로덕션 및 테스트 워크로드, 또는 인보이스 및 결제 데이터를 계정을 소유한 조직의 개별 사업부나 부서 또는 이해 관계자에 제공하기 위한 별도의 계정을 사용하는 등이 있습니다.
- 그룹화 요구 사항 정의: 그룹화 요구 사항은 분리 요구 사항에 우선하지 않지만 관리를 지원하는 데 사용됩니다. 분리할 필요가 없는 유사한 환경 또는 워크로드를 함께 그룹화합니다. 예를 들어, 하나 이상의 워크로드에 속하는 여러 테스트 또는 개발 환경을 함께 그룹화합니다.
- 계정 구조 정의: 이러한 분리와 그룹화를 사용하여 각 그룹에 대한 계정을 지정하고 분리 요구 사항을 유지합니다. 이러한 계정은 멤버 또는 연결된 계정입니다. 이러한 멤버 계정을 하나의 관리 또는 지급인 계정으로 그룹화하면 사용량이 결합되어 모든 계정에서 더 큰 대량 구매 할인을 받을 수 있고 모든 계정에 대해 단일 청구서가 제공됩니다. 결제 데이터를 분리하고 각 멤버 계정에 결제 데이터의 개별 보기를 제공할 수 있습니다. 멤버 계정의 사용 내역 또는 결제 데이터가 다른 계정에 표시되지 않아야 하거나 AWS와 별도의 청구서가 필요한 경우 여러 관리 또는 지급인 계정을 정의합니다. 이 경우 각 멤버 계정마다 고유의 관리 또는 지급인 계정이 있습니다. 리소스는 항상 멤버 또는 연결 계정에 배치되어야 합니다. 관리 또는 지급인 계정은 관리에만 사용해야 합니다.

리소스

관련 문서:

- 비용 할당 태그 사용
- 직무 역할에 대한 AWS 관리형 정책
- AWS 다중 계정 결제 전략
- IAM 정책을 사용하여 AWS 리전에 대한 액세스 제어
- AWS Control Tower
- AWS Organizations

- 관리 계정 및 멤버 계정 모범 사례
- 여러 계정을 사용하여 AWS 환경 구성
- 예약 인스턴스 및 절감형 플랜 할인 공유 활성화
- 통합 결제
- 통합 결제

관련 예시:

- CUR 분할 및 액세스 공유

관련 동영상:

- AWS Organizations 소개
- AWS Organizations의 모범 사례를 사용하는 다중 계정 AWS 환경 설정

관련 예시:

- Well-Architected 실습: AWS 조직 생성(레벨 100)
- AWS Cost and Usage Report 분할 및 액세스 공유
- 통신 회사를 위한 AWS 다중 계정 전략 정의
- AWS 계정 최적화 모범 사례
- AWS Organizations의 조직 단위에 대한 모범 사례

COST02-BP04 그룹 및 역할 만들기

정책에 따라 그룹과 역할을 만들고 각 그룹에서 인스턴스와 리소스를 생성, 수정 또는 폐기할 수 있는 사용자를 제어합니다. 예를 들어 개발, 테스트 및 프로덕션 그룹을 만듭니다. 이는 AWS 서비스와 타사 솔루션에 적용됩니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

정책을 개발한 후에는 조직 내부 사용자의 논리적 그룹 및 역할을 생성하여 권한을 할당하고 사용량을 제어할 수 있습니다. 개괄적인 수준에서 사람을 그룹화하는 것으로 시작합니다. 일반적으로 조직 단위 및 직무 역할(예: IT 부서의 시스템 관리자 또는 재무 관리자)이 여기에 해당합니다. 그룹에는 유사한 작

업을 수행하고 유사한 접근 권한이 필요한 사람이 포함됩니다. 역할은 그룹이 수행해야 할 작업을 정의합니다. 예를 들어 IT의 시스템 관리자는 모든 리소스를 생성할 수 있는 접근 권한이 필요하지만 분석 팀원은 분석 리소스만 생성하면 됩니다.

구현 단계

- 그룹 만들기: 조직 정책에 정의된 사용자 그룹을 사용하여 필요한 경우 해당 그룹을 만듭니다. 사용자, 그룹 및 인증에 대한 모범 사례는 보안 원칙을 참조하십시오.
- 역할 및 정책 만들기: 조직 정책에 정의된 작업을 사용하여 필요한 역할과 액세스 정책을 생성합니다. 역할 및 정책에 대한 모범 사례는 보안 원칙을 참조하십시오.

리소스

관련 문서:

- 직무에 관한 AWS 관리형 정책
- AWS 다중 계정 결제 전략
- IAM 정책을 사용하여 AWS 리전에 대한 액세스 제어
- Well-Architected 보안 원칙

관련 예시:

- Well-Architected 실습: 기본 자격 증명 및 액세스

COST02-BP05 비용 제어 기능 구현

조직 정책 및 정의된 그룹과 역할을 기준으로 제어 기능을 구현합니다. 이렇게 하면 조직 요구 사항에 따라 정의된 비용만 발생합니다. 예를 들어 리전 또는 리소스 유형 액세스를 제어할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

비용 제어를 구현하기 위한 일반적인 첫 번째 단계는 비용 또는 사용량 이벤트가 정책 범위를 벗어날 때 알림을 설정하는 것입니다. 워크로드 또는 새로운 활동에 대한 제한 또는 부정적인 영향 없이 신속하게 조치를 취하고 교정 조치가 필요한지 여부를 확인할 수 있습니다. 워크로드 및 환경 제한을 파악한 후에는 거버넌스를 시행할 수 있습니다. AWS Budgets을 사용하면 AWS 비용, 사용량 및 약정 할인(절감형 플랜 및 예약 인스턴스)에 대한 알림을 설정하고 월간 예산을 정의할 수 있습니다. 집계 비용

수준(예: 모든 비용)에서 예산을 생성하거나 연결 계정, 서비스, 태그 또는 가용 영역과 같은 특정 차원만 포함하는 보다 세분화된 수준에서 예산을 생성할 수 있습니다.

AWS Budgets을 통해 예산 한도를 설정한 후 AWS Cost Anomaly Detection을 사용하여 예상치 못한 비용을 줄일 수 있습니다. AWS Cost Anomaly Detection은 기계 학습을 사용하여 비용과 사용량을 지속적으로 모니터링함으로써 비정상적인 지출을 탐지하는 비용 관리 서비스입니다. 이를 통해 비정상적인 지출과 근본 원인을 식별하여 빠르게 조치를 취할 수 있습니다. 먼저 AWS Cost Anomaly Detection에서 비용 모니터링을 생성한 후 달러 임계값을 설정하여 알림 기본 설정을 선택합니다(예: 1,000 USD 이상의 영향이 있는 이상을 알림). 알림을 수신하고 나면 이상과 비용의 영향에 대한 근본 원인을 분석할 수 있습니다. 또한 AWS Cost Explorer에서 자체적인 이상 분석을 모니터링 및 수행할 수도 있습니다.

AWS Identity and Access Management 및 AWS Organizations 서비스 제어 정책(SCP)을 통해 AWS에 거버넌스 정책을 시행합니다. IAM을 사용하면 AWS 서비스 및 리소스에 대한 액세스를 안전하게 관리할 수 있습니다. IAM을 사용하면 AWS 리소스를 생성 및 관리할 수 있는 사용자, 생성할 수 있는 리소스 유형 및 리소스 생성 위치를 제어할 수 있습니다. 이렇게 하면 정의된 정책 외부에 리소스가 생성될 가능성이 최소화됩니다. 이전에 생성한 역할 및 그룹을 사용하고 IAM 정책을 할당하여 올바른 사용을 시행할 수 있습니다. SCP를 사용하면 조직 내 모든 계정에 대해 사용 가능한 최대 권한을 중앙에서 제어하여 계정이 액세스 제어 지침을 계속 준수할 수 있습니다. SCP는 모든 기능이 활성화된 조직에서만 사용할 수 있으며, 기본적으로 멤버 계정에 대한 작업을 거부하거나 허용하도록 SCP를 구성할 수 있습니다. 액세스 관리 구현에 대한 자세한 내용은 Well-Architected 보안 원칙 백서를 참조하세요.

AWS Service Quotas 관리를 통해 거버넌스를 구현할 수도 있습니다. 오버헤드를 최소화하는 방식으로 서비스 할당량을 설정하고 정확하게 유지 관리하면 조직의 요구 사항에 포함되지 않는 리소스 생성을 최소화할 수 있습니다. 이렇게 하려면 요구 사항 변경 속도와 진행 중인 프로젝트(리소스 생성 및 폐기)를 파악하고 할당량 변경을 구현할 수 있는 속도를 고려해야 합니다. 필요한 경우 서비스 할당량을 사용하여 할당량을 늘릴 수 있습니다.

구현 단계

- 지출에 대한 알림 구현: 정의된 조직 정책으로 AWS Budgets을 생성하여 지출이 정책을 벗어날 때 이를 알립니다. 전체 계정 지출에 대해 알리는 비용 예산을 계정당 하나씩 여러 개 구성합니다. 각 계정 내에서 해당 계정 내의 더 작은 단위에 대한 추가 비용 예산을 구성합니다. 이러한 단위는 계정 구조에 따라 다릅니다. 일반적인 예로 AWS 리전, 워크로드(태그 사용) 또는 AWS 서비스가 있습니다. 개인의 이메일 계정이 아닌 이메일 배포 목록을 알림에 대한 수신자로 구성합니다. 금액을 초과할 때에 대한 실제 예산을 구성하거나 예상 예산을 사용하여 예상 사용량에 대해 알릴 수 있습니다. 또한 특정 IAM 또는 SCP 정책을 시행하거나, 대상 Amazon EC2 또는 Amazon RDS 인스턴스를 중지할 수 있는 AWS 예산 활동을 사전 구성할 수 있습니다. 예산 활동은 자동으로 실행하거나 워크플로 승인이 필요할 수 있습니다.

- 비정상적인 지출에 대한 알림 구현: AWS Cost Anomaly Detection을 사용하여 조직 내 뜻밖의 비용을 줄이고 비정상적인 지출 가능성의 근본 원인을 분석할 수 있습니다. 비용 모니터링을 생성하여 지정된 세분화에서 비정상적인 지출을 식별하고 AWS Cost Anomaly Detection에서 알림을 구성하면, 비정상적인 지출이 탐지되었을 때 알림이 전송됩니다. 이를 통해 이상의 근본 원인을 분석하고 비용에 대한 영향을 이해할 수 있습니다. AWS Cost Categories를 사용하는 동시에 AWS Cost Anomaly Detection을 구성하여 어떤 프로젝트 팀이나 사업부에서 예상치 못한 비용의 근본 원인을 분석하고, 시기 적절하며 필요한 조치를 취할지 식별할 수 있습니다.
- 사용량에 대한 제어 구현: 정의된 조직 정책으로 IAM 정책 및 역할을 구현하여 사용자가 수행할 수 있는 작업과 수행할 수 없는 작업을 지정합니다. AWS 정책에 여러 조직 정책이 포함될 수 있습니다. 정책을 정의한 것과 동일한 방식으로 광범위하게 시작한 다음 각 단계에서 보다 세분화된 제어를 적용합니다. 서비스 한도도 효과적인 사용량 제어 방식입니다. 모든 계정에 올바른 서비스 한도를 설정합니다.

리소스

관련 문서:

- 직무 역할에 대한 AWS 관리형 정책
- AWS 다중 계정 결제 전략
- IAM 정책을 사용하여 AWS 리전에 대한 액세스 제어
- AWS Budgets
- AWS Cost Anomaly Detection
- AWS 비용 제어

관련 동영상:

- AWS Budgets를 사용하여 지출 및 사용량을 추적하는 방법

관련 예시:

- IAM 액세스 관리 정책 예시
- 서비스 제어 정책 예시
- AWS 예산 활동
- 태그를 사용하여 Amazon EC2 리소스에 대한 액세스를 제어하는 IAM 정책 생성
- 특정 Amazon EC2 리소스에 대한 IAM Identity의 액세스 제한

- 패밀리별로 Amazon EC2 사용량을 제한하는 IAM 정책 생성
- Well-Architected 실습: 비용 및 사용 거버넌스(레벨 100)
- Well-Architected 실습: 비용 및 사용 거버넌스(레벨 200)
- AWS Chatbot을 사용하여 Cost Anomaly Detection의 Slack 통합

COST02-BP06 프로젝트 수명 주기 추적

프로젝트, 팀 및 환경의 수명 주기를 추적, 측정 및 감사하여 불필요한 리소스 사용 및 이에 따른 비용 지출을 막으십시오.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

워크로드의 전체 수명 주기를 추적해야 합니다. 수명 주기를 추적하면 워크로드 또는 워크로드 구성 요소가 더 이상 필요하지 않을 때 이를 폐기하거나 수정할 수 있습니다. 이 기능은 새로운 서비스 또는 기능을 릴리스할 때 특히 유용합니다. 사용 중인 것처럼 보일 수 있는 기존 워크로드 및 구성 요소를 폐기하여 고객을 새 서비스로 리디렉션해야 합니다. 워크로드의 이전 단계를 확인하십시오. 워크로드가 프로덕션 단계로 전환된 후에는 이전 환경을 폐기하거나 다시 필요할 때까지 용량을 크게 줄일 수 있습니다.

AWS는 엔터티 수명 주기 추적에 사용할 수 있는 다양한 관리 및 거버넌스 서비스를 제공합니다. 여러분은 AWS Config 또는 AWS Systems Manager 를 사용하여 AWS 리소스 및 구성에 대한 상세한 인벤토리를 제공할 수 있습니다. 추적 기능을 기존 프로젝트 또는 자산 관리 시스템에 통합하여 조직 내의 진행 중인 프로젝트와 제품을 추적하는 것이 좋습니다. AWS에서 제공하는 다양한 이벤트 및 지표 집합과 현재 시스템을 통합하면 중요한 수명 주기 이벤트 뷰(view)를 작성하고 리소스를 사전에 관리하여 불필요한 비용을 줄일 수 있습니다.

엔터티 수명 주기 추적 구현에 대한 자세한 내용은 Well-Architected 운영 우수성 원칙 백서 를 참조하십시오.

구현 단계

- 워크로드 검토 수행: 조직 정책에 정의된 대로 기존 프로젝트를 감사합니다. 감사에 드는 노력은 조직의 대략적인 위험, 가치 또는 비용에 비례해야 합니다. 감사에 포함할 주요 영역은 조직의 인시던트 또는 가동 중단 위험, 가치 또는 조직에 대한 기여도(수익 또는 브랜드 평판으로 측정), 워크로드 비용(총 리소스 비용 및 운영 비용으로 측정), 워크로드 사용량(단위 시간당 조직 성과 수로 측정)입니다. 수명 주기 동안 이러한 영역이 변경되면 전체 또는 부분 폐기와 같은 워크로드 조정이 필요합니다.

리소스

관련 문서:

- AWS Config
- AWS Systems Manager
- 직무에 관한 AWS 관리형 정책
- AWS 다중 계정 결제 전략
- IAM 정책을 사용하여 AWS 리전에 대한 액세스 제어

COST 3 사용량과 비용을 어떻게 모니터링합니까?

비용을 모니터링하고 적절하게 할당하기 위한 정책 및 절차를 구성합니다. 이렇게 하면 이 워크로드의 비용 효율성을 측정하고 개선할 수 있습니다.

모범 사례

- COST03-BP01 세부 정보 소스 구성
- COST03-BP02 비용 및 사용량에 조직 정보 추가
- COST03-BP03 비용 귀속 범주 식별
- COST03-BP04 조직 지표 설정
- COST03-BP05 결제 및 비용 관리 도구 구성
- COST03-BP06 워크로드 지표를 기준으로 비용 할당

COST03-BP01 세부 정보 소스 구성

AWS Cost and Usage Report와 Cost Explorer에서 시간 단위의 세분 수준을 구성하여 자세한 비용 및 사용량 정보를 제공합니다. 전송된 모든 비즈니스 성과에 대한 로그 항목을 생성하도록 워크로드를 구성합니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

AWS Cost Explorer에서 시간 단위 세분화를 활성화하고 AWS Cost and Usage Report(CUR)를 생성합니다. 이러한 데이터 원본은 전체 조직의 비용 및 사용량을 가장 정확하게 보여줍니다. CUR은 모든 청구 가능한 AWS 서비스에 대해 일 또는 시간 단위 사용량 세분화, 요금, 비용 및 사용량 속성을 제공

합니다. 태그 지정, 위치, 리소스 속성 및 계정 ID를 포함하여 가능한 모든 차원을 CUR에서 사용할 수 있습니다.

다음과 같은 사용자 지정을 사용하여 CUR을 구성합니다.

- 리소스 ID 포함
- CUR 자동 새로 고침
- 시간 단위 세분화
- 버전 관리: 기존 보고서 덮어쓰기
- 데이터 통합: Amazon Athena(Parquet 형식 및 압축)

참고로, AWS Glue 를 사용하여 분석에 사용할 데이터를 준비하고 Amazon Athena 를 사용하여 데이터 분석을 수행하며 SQL을 사용하여 데이터를 쿼리합니다. 또한 Amazon QuickSight 를 사용하여 사용자 지정된 복합적인 시각화를 작성하고 조직 전체에 배포할 수도 있습니다.

구현 단계

- 비용 및 사용 보고서 구성: 결제 콘솔을 사용하여 하나 이상의 비용 및 사용 보고서를 구성합니다. 모든 식별자 및 리소스 ID를 포함하는 시간 단위의 세분 수준으로 보고서를 구성합니다. 다른 세부 수준의 다른 보고서를 생성하여 더 개략적인 요약 정보를 제공할 수도 있습니다.
- Cost Explorer에서 시간 단위의 세분 수준 구성: 결제 콘솔을 사용하여 시간 단위 및 리소스 수준 데이터를 사용합니다.

> **Note**
>
> 이 기능 사용에 따른 비용이 발생합니다. 자세한 내용은 요금을 참조하십시오.

- 애플리케이션 로깅 구성: 애플리케이션에서 제공하는 각 비즈니스 성과를 로깅하여 추적하고 측정할 수 있는지 확인합니다. 비용 및 사용량 데이터와 일치하도록 이 데이터의 세부 수준이 시간 단위 이상인지 확인합니다. 로깅 및 모니터링에 대한 자세한 내용은 AWS Well-Architected 운영 우수성 원칙 을 참조하십시오.

리소스

관련 문서:

- AWS 계정 설정

- AWS Cost and Usage Report(CUR)
- AWS Glue
- Amazon QuickSight
- AWS 비용 관리 요금
- AWS 리소스 태그 지정
- AWS Budgets를 통해 비용 분석
- Cost Explorer를 사용한 비용 분석
- AWS Cost and Usage Report 관리
- AWS Well-Architected 운영 우수성 원칙

관련 예시:

- AWS 계정 설정

COST03-BP02 비용 및 사용량에 조직 정보 추가

조직, 워크로드 속성, 비용 할당 범주에 따라 태그 지정 스키마를 정의하여, 리소스를 필터링하고 검색하거나 비용 관리 도구에서 비용과 사용량을 모니터링할 수 있습니다. 가능하면 모든 리소스에서 목적, 팀, 환경, 비즈니스와 관련이 있는 기타 기준에 따라 일관적인 태그 지정을 구현합니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

AWS에서 태그 지정을 구현하여 조직 정보를 리소스에 추가할 수 있습니다. 이러한 조직 정보는 다시 비용 및 사용량 정보에 추가됩니다. 태그는 키–값 페어입니다. 키는 정의되며 조직 전체에서 고유해야 하고 값은 리소스 그룹에 고유해야 합니다. 예를 들어 키가 `Environment`이고 값은 `Production`인 키-값 페어가 있을 수 있습니다. 프로덕션 환경의 모든 리소스에는 이 키–값 페어가 있습니다. 태그 지정을 통해 의미가 있는 관련된 조직 정보를 사용하여 비용을 분류하고 추적할 수 있습니다. 조직 범주(예: 비용 센터, 애플리케이션 이름, 프로젝트 또는 소유자)를 나타내는 태그를 적용하고 워크로드 및 워크로드의 특성(예: 테스트 또는 프로덕션)을 식별하여 조직 전체의 비용 및 사용량을 해당하는 개체에 귀속할 수 있습니다.

AWS 리소스(예: Amazon Elastic Compute Cloud 인스턴스 또는 Amazon Simple Storage Service 버킷)에 태그를 적용하고 활성화하면 AWS가 비용 및 사용 보고서에 이 정보를 추가합니다. 태그가 지정

된 리소스와 지정되지 않은 리소스에 대해 보고서를 실행하고 분석을 수행하면 내부 비용 관리 정책에 대한 규정 준수를 개선하고 정확한 귀속을 보장할 수 있습니다.

조직 전체 계정에 적용되는 AWS 태그 지정 표준을 생성하고 구현하면 통일성 있는 일관된 방식으로 AWS 환경을 관리하고 제어할 수 있습니다. AWS Organizations에서 태그 정책을 사용하여 AWS Organizations의 계정에 포함된 AWS 리소스에 태그를 사용하는 방법에 대한 규칙을 정의할 수 있습니다. 태그 정책을 사용하면 AWS 리소스 태그 지정에 대한 표준화된 접근 방식을 쉽게 도입할 수 있습니다.

AWS Tag Editor를 사용하면 여러 리소스의 태그를 추가, 삭제 및 관리할 수 있습니다. Tag Editor에서는 태그를 지정하려는 리소스를 검색하고, 검색 결과에 나온 리소스에 대한 태그를 관리할 수 있습니다.

AWS Cost Categories를 사용하면 리소스에 태그를 지정할 필요 없이 조직의 의미를 비용에 지정할 수 있습니다. 비용 및 사용량 정보를 고유한 내부 조직 구조에 매핑할 수도 있습니다. 계정 및 태그와 같은 결제 차원을 사용하여 비용을 매핑하고 분류하는 범주 규칙을 정의하면 태그 지정에 더해 또 다른 수준의 관리 기능을 제공할 수 있습니다. 특정 계정과 태그를 여러 프로젝트에 매핑할 수도 있습니다.

구현 단계

- 태그 지정 스키마 정의: 비즈니스 전반의 모든 이해관계자를 모아 스키마를 정의합니다. 여기에는 대개 기술직, 재무직 및 경영진이 포함됩니다. 모든 리소스가 보유해야 하는 태그 목록과 리소스가 보유해야 하는 태그 목록을 정의합니다. 조직 전체에 걸쳐 태그 이름과 값이 일치하는지 확인합니다.
- 태그 리소스: 정의된 비용 속성 범주를 사용하여 범주에 따라 워크로드의 모든 리소스에 태그를 지정합니다. CLI, Tag Editor, AWS Systems Manager 등의 도구를 사용하여 효율성을 높입니다.
- AWS Cost Categories 구현: 태그 지정을 구현하지 않고 Cost Categories를 만들 수 있습니다. Cost categories는 기존의 비용 및 사용량 규모를 사용합니다. 스키마에서 범주 규칙을 생성하고, Cost Categories에 구현합니다.
- 태그 지정 자동화: 모든 리소스에서 높은 수준의 태그 지정을 유지하는지 확인하려면 리소스가 생성될 때 자동으로 태그가 지정되도록 자동화합니다. AWS CloudFormation 등의 서비스를 사용하여 리소스를 생성할 때 태그가 지정되는지 확인합니다. 또한, Lambda 함수를 사용하여 태그를 자동으로 지정하는 맞춤형 솔루션을 만들거나, 워크로드를 주기적으로 스캔하고 태그가 지정되지 않은 리소스를 제거하는 마이크로서비스를 사용할 수 있습니다. 이는 테스트 및 개발 환경에서 매우 유용합니다.
- 태그 지정 모니터링 및 보고: 조직 전체에서 높은 수준의 태그 지정을 유지하는지 확인하기 위해 워크로드 전체의 태그를 보고하고 모니터링합니다. AWS Cost Explorer를 사용하여 태그가 지정되거나 태그가 지정되지 않은 리소스의 비용을 확인하거나 Tag Editor 등의 서비스를 사용할 수 있습니

다. 태그가 지정되지 않은 리소스의 수를 정기적으로 검토하고 원하는 태그 지정 수준에 도달할 때까지 태그를 추가하는 작업을 수행합니다.

리소스

관련 문서:

- 태그 지정 모범 사례
- AWS CloudFormation 리소스 태그
- AWS Cost Categories
- AWS 리소스 태그 지정
- AWS Budgets을 통한 비용 분석
- Cost Explorer를 사용한 비용 분석
- AWS 비용 및 사용 보고서 관리

관련 동영상:

- 비용 센터나 프로젝트별로 청구서를 분리하려면 AWS 리소스에 태그를 어떻게 지정해야 하나요?
- AWS 리소스 태그 지정

관련 예시:

- 식별 정보나 역할에 따라 새로운 AWS 리소스에 자동으로 태그 지정

COST03-BP03 비용 귀속 범주 식별

조직 내 비용을 내부 소비 주체에 할당하는 데 활용할 수 있는 조직 범주(예: 사업부, 부서, 프로젝트)를 파악하여 지출에 대한 책임을 강화하고 소비 행동을 효과적으로 주도하십시오.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 높음

구현 가이드

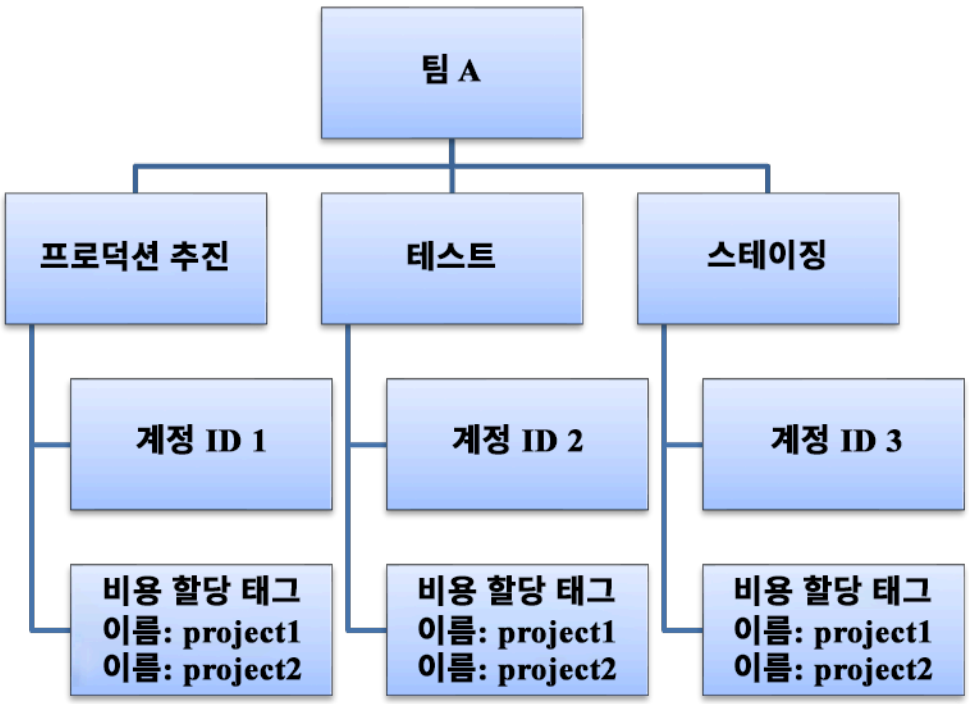
비용 범주화 프로세스는 예산 책정, 회계, 재무 보고, 의사 결정, 벤치마킹, 프로젝트 관리에 매우 중요합니다. 비용을 분류하고 범주화하면 팀에서 클라우드 여정 전반에 걸쳐 발생할 비용 유형을 더 잘 이해할 수 있어 정보에 따라 결정을 내리고 예산을 효과적으로 관리할 수 있습니다.

클라우드 지출 책임에 따라 수요와 비용을 엄격히 관리하는 것은 강력한 인센티브로를 이어집니다 그 결과 클라우드 지출의 대부분을 소비 사업부 또는 팀에 할당하는 조직에서 클라우드 비용을 크게 절감할 수 있습니다.

재무 팀 및 기타 이해관계자와 협력하여 조직 내 비용 할당 방식 관련 요구 사항을 파악합니다. 워크로드 비용은 개발, 테스트, 프로덕션 및 폐기를 포함한 전체 수명 주기에 걸쳐 할당되어야 합니다. 조직에서 학습, 직원 개발 및 아이디어 창출에 대해 발생한 비용의 귀속 방법을 파악합니다. 그러면 이 목적으로 사용되는 계정을 일반적인 IT 비용 예산 대신 교육 및 개발 예산에 올바르게 할당할 수 있습니다.

조직 내 이해관계자와 함께 비용 기여 범주를 정의한 후 AWS Cost Categories를 사용하여 비용 및 사용 정보를 특정 프로젝트 또는 부서/사업부 AWS 계정과 같은 AWS 클라우드의 의미 있는 범주로 그룹화합니다. 계정, 태그, 서비스, 요금 유형 및 기타 비용 범주와 같은 다양한 차원을 사용하여 정의한 규칙에 따라 사용자 지정 범주를 생성하고 비용과 사용량 정보를 이러한 범주에 매핑할 수 있습니다. Cost Categories가 설정되면 범주별로 비용과 사용량 정보를 볼 수 있으므로 조직에서 더 나은 전략 결정과 구매 결정을 내릴 수 있습니다. 이들 범주는 AWS Cost Explorer, AWS Budgets, AWS Cost and Usage Report에서 표시됩니다.

예를 들어, 다음 다이어그램에서는 여러 환경(규칙)이 있는 여러 팀(Cost Category)과 여러 리소스 또는 자산(차원)이 있는 각 환경 등 조직의 비용과 사용량 정보를 그룹화하는 방법을 보여 줍니다.



비용 및 사용량 조직도.

구현 단계

- 조직 범주 정의: 이해 관계자를 만나 조직의 구조 및 요구 사항을 반영하는 범주를 정의합니다. 이들은 사업부, 예산, 비용 센터 또는 부서와 같은 기존 재무 범주의 구조에 직접 매핑됩니다. 교육과 같이 클라우드가 비즈니스에 제공하는 성과를 살펴보십시오. 이들은 조직 범주이기도 합니다. 한 리소스에 여러 범주를 할당할 수 있으며 리소스는 서로 다른 여러 범주에 있을 수 있으므로 필요한 만큼 범주를 정의합니다.
- 역할 범주 정의: 이해 관계자를 만나 비즈니스 내에서 자신의 역할을 반영하는 범주를 정의합니다. 역할 범주는 워크로드 또는 애플리케이션 이름과 프로덕션, 테스트, 개발 등의 환경 유형일 수 있습니다. 한 리소스에 여러 범주를 할당할 수 있으며 리소스는 서로 다른 여러 범주에 있을 수 있으므로 AWS Cost Categories를 사용해 범주화된 구조 내에서 비용을 관리 할 수 있도록 필요한 만큼 범주를 정의합니다.
- AWS Cost Categories 정의: 비용 및 사용량 정보를 구성하기 위해 비용 범주를 생성할 수 있습니다. 또한 AWS Cost Categories를 사용하여 AWS 비용과 사용량을 의미 있는 범주로 매핑할 수 있습니다. Cost Categories를 사용하면 규칙 기반 엔진을 사용하여 비용을 정리할 수 있습니다. 구성하는 규칙에 따라 비용이 범주로 구성됩니다. 이들 규칙 내에서 각 범주에 대한 여러 차원(예: 특정 AWS 계정, 특정 AWS 서비스, 특정 요금 유형)을 사용하여 필터링할 수 있습니다. 그런 다음, AWS Billing and Cost Management 콘솔의여러 제품에서 이들 범주를 사용할 수 있습니다. AWS Cost Explorer, AWS Budgets, AWS Cost and Usage Report, AWS Cost Anomaly Detection를 포함합니다. Cost Categories를 사용하여 비용 그룹을 만들 수도 있습니다. Cost Categories를 만들면(Cost Category를 만들고 사용량 레코드에 값이 업데이트될 때까지 최대 24시간 소요) AWS Cost Explorer, AWS Budgets, AWS Cost and Usage Report 및 AWS Cost Anomaly Detection에 표시됩니다. 예를 들어, 사업부(DevOps Team)에 대한 Cost Categories를 생성하고 각 범주 아래에서 정의된 그룹을 기반으로 여러 차원(AWS 계정, 비용 할당 태그, 서비스 또는 요금 유형)을 가진 여러 규칙(각 하위 범주에 대한 규칙)을 생성합니다. AWS Cost Explorer 및 AWS Budgets에서 Cost Category가 추가적 청구 차원으로 표시됩니다. 이를 사용하여 특정 Cost Category 값을 필터링하거나 Cost Category를 기준으로 그룹화할 수 있습니다.

리소스

관련 문서:

- AWS 리소스 태그 지정
- 비용 할당 태그 사용
- AWS Budgets를 통한 비용 분석
- Cost Explorer를 사용한 비용 분석

- AWS Cost and Usage Reports 관리
- AWS Cost Categories
- AWS Cost Categories로 비용 관리
- Cost Categories 만들기
- Cost Categories 태그 지정
- Cost Categories 내에서 요금 분할
- AWS Cost Categories 기능

관련 예시:

- AWS Cost Categories로 비용 및 사용량 데이터 구성
- AWS Cost Categories로 비용 관리

COST03-BP04 조직 지표 설정

이 워크로드에 필요한 조직 지표를 설정합니다. 워크로드 지표의 예로는 생성된 고객 보고서 또는 고객에게 제공된 웹 페이지가 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

비즈니스 성공을 기준으로 워크로드의 결과를 측정하는 방법을 파악합니다. 일반적으로 각 워크로드에는 성능을 나타내는 소규모의 주요 결과 세트가 있습니다. 구성 요소가 많은 복잡한 워크로드가 있는 경우 목록의 우선 순위를 지정하거나 각 구성 요소에 대한 지표를 정의하고 추적할 수 있습니다. 팀과 협력하여 사용할 지표를 파악하십시오. 이 단위는 워크로드의 효율성 또는 각 비즈니스 결과의 비용을 파악하는 데 사용됩니다.

구현 단계

- 워크로드 결과 정의: 비즈니스 이해 관계자를 만나 워크로드에 대한 성과를 정의합니다. 이는 고객 사용량의 기본 척도이며 기술 지표가 아니라 비즈니스 지표여야 합니다. 워크로드당 거시 지표 수가 적어야 합니다(5개 미만). 워크로드가 서로 다른 사용 사례에 대해 여러 성과를 생성하는 경우 이들을 단일 지표로 그룹화합니다.
- 워크로드 구성 요소 결과 정의: 크고 복잡한 워크로드가 있거나 잘 정의된 입력 및 출력을 사용하여 워크로드를 마이크로서비스 등의 구성 요소로 쉽게 나눌 수 있는 경우 각 구성 요소에 대한 지표를

정의합니다. 노력은 구성 요소의 가치와 비용을 반영해야 합니다. 가장 큰 구성 요소로 시작하여 더 작은 구성 요소로 진행합니다.

리소스

관련 문서:

- AWS 리소스 태그 지정
- AWS Budgets을 통한 비용 분석
- Cost Explorer를 사용한 비용 분석
- AWS 비용 및 사용 보고서 관리

COST03-BP05 결제 및 비용 관리 도구 구성

조직 정책에 따라 비용 관리 도구를 구성하여 클라우드 지출을 관리하고 최적화합니다. 여기에는 비용과 사용 데이터를 구성하고 추적하고 통합 과금과 액세스 권한을 통해 제어 기능을 강화하며 예산 책정과 예측을 통해 계획을 개선하고 알림 또는 경고를 수신하며 리소스와 가격 최적화를 통해 비용을 더욱 절감하는 서비스, 툴, 리소스가 포함됩니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 높음

구현 가이드

강한 책임감을 갖게 하려면 비용 배분 전략의 일환으로 계정 전략을 먼저 고려해야 합니다. 이를 제대로 한다면 그 이상은 필요 없을 수도 있습니다. 그렇지 않으면 인식이 결여되고 안 좋은 상황이 더 생길 수 있습니다.

클라우드 지출에 대한 책임감을 높이려면 사용자가 비용과 사용량을 파악할 수 있는 도구에 액세스할 수 있어야 합니다. 모든 워크로드와 팀은 다음과 같은 세부 정보 및 목적에 맞게 도구를 구성하는 것이 좋습니다.

- 구성: 자체 태그 전략 및 분류법을 사용하여 비용 할당 및 거버넌스 기준을 설정합니다. 지원되는 AWS 리소스에 태그를 지정하고 조직 구조(사업부, 부서 또는 프로젝트)에 따라 의미 있게 분류합니다. 특정 비용 센터의 계정 이름에 태그를 지정하고 AWS Cost Categories와 매핑하여 해당 비용 센터의 특정 사업부 계정을 그룹화하여 사업부 소유자가 한 곳에서 여러 계정의 소비 내역을 볼 수 있습니다.
- 액세스: 조직 전체의 청구 정보를 통합 결제에서 추적하고 올바른 이해 관계자 및 비즈니스 소유자가 액세스할 수 있는지 확인합니다.

- 통제: 적절한 가드레일로 효과적인 거버넌스 메커니즘을 구축하여 SCP, 태그 정책, 예산 알림을 사용할 때 예상치 못한 시나리오를 방지합니다. 예를 들어 효과적인 제어 메커니즘을 사용하면 팀이 지원되지 않는 리전에서 리소스를 생성하는 것을 방지할 수 있습니다.
- 현재 상태: 현재 비용 및 사용량 수준을 보여주는 대시보드를 구성합니다. 대시보드는 작업 환경 내의 가시성이 높은 위치에서 사용할 수 있어야 합니다(운영 대시보드와 유사함). 이때 클라우드 인텔리전스 대시보드(CID) 또는 기타 지원되는 제품을 통해 이러한 가시성을 확보할 수 있습니다.
- 알림: 비용 또는 사용량이 정의된 한도를 벗어나거나 AWS Budgets 또는 AWS Cost Anomaly Detection에서 이상 현상이 발생할 경우 알림을 보냅니다.
- 보고서: 모든 비용 및 사용량 정보를 요약하고 할당 가능한 비용의 상세한 데이터를 통해 클라우드 지출에 대한 인식과 책임 의식을 높입니다. 보고서는 해당 보고서를 사용하는 팀과 관련이 있어야 하며 이상적으로는 권장 사항을 포함해야 합니다.
- 추적: 구성된 목표 또는 타겟을 기준으로 현재 비용 및 사용량을 보여줍니다.
- 분석: 팀원이 가능한 모든 차원에서 시간 단위와 같은 세부 수준까지 사용자 지정 심층 분석을 수행할 수 있는 기능을 제공합니다.
- 검사: 리소스 배포 및 비용 최적화 기회를 최신 상태로 유지할 수 있습니다. 조직 수준에서 리소스 배포에 대한 알림을 받고(Amazon CloudWatch, Amazon SNS 또는 Amazon SES 사용) 비용 최적화 권장 사항(예: AWS Compute Optimizer 또는AWS Trusted Advisor)을 검토합니다.
- 트렌드: 필요한 기간의 비용 및 사용량 변동을 보여주는 기능을 필요한 세부 수준으로 제공합니다.
- 예측: 향후 예상 비용을 표시하고 리소스 사용량을 예측하며 생성하는 예측 대시보드에 대한 지출을 표시합니다.

필수 항목에 AWS 도구, 예를 들어 AWS Cost Explorer, AWS Billing또는 AWS Budgets를 사용하거나 더 자세히 보기 위해 CUR 데이터를 Amazon Athena 및 Amazon QuickSight와 통합하여 이 기능을 제공할 수 있습니다. 조직에 필수적인 기술 또는 역량이 없다면 AWS ProServ, AWS Managed Services(AMS)또는 AWS Partner과 협력하여 그들의 도구를 사용할 수 있습니다. 타사 도구를 사용할 수도 있지만, 먼저 그 비용이 조직에 도움이 되는지 확인하십시오.

구현 단계

- 팀 기반 도구 액세스 허용: 계정을 구성하고 도구 사용에 필수인 비용과 사용량 보고서에 대한 액세스 권한이 있는 그룹을 만들어 AWS Identity and Access Management를 통해 도구에 대한 액세스를 관리하는 데 사용합니다(AWS Cost Explorer 등). 이 그룹에는 애플리케이션을 소유하거나 관리하는 모든 팀의 담당자가 포함되어야 합니다. 그래야 모든 팀이 비용 및 사용량 정보에 액세스하여 사용량을 추적할 수 있습니다.

- AWS Budgets 구성: 워크로드의 모든 계정에서 AWS Budgets를 구성합니다. 태그를 사용하여 전체 계정 지출에 대한 예산과 워크로드에 대한 예산을 설정합니다. AWS Budgets에서 알림을 구성하여 책정된 예산을 초과하거나 예상 비용이 예산을 초과하는 경우 알림을 받습니다.
- AWS Cost Explorer 구성: 워크로드 및 계정에 대해 AWS Cost Explorer을 구성하여 앞으로의 분석에 대한 비용 데이터를 시각화합니다. 전체 지출, 워크로드의 주요 사용량 지표 및 과거 비용 데이터를 기반으로 미래의 비용을 예측하는 워크로드 대시보드를 만듭니다.
- AWS Cost Anomaly Detection 구성: 계정, 핵심 서비스 또는 비용 및 사용량을 모니터링하기 위해 만든 Cost Categories에 대해 AWS Cost Anomaly Detection을 사용하여 비정상적인 지출을 감지합니다. 집계된 보고서에서 개별적으로 알림을 수신하고, 이메일이나 Amazon Simple Notification Service 주제로 알림을 수신하면 이상이 발생한 근본 원인을 분석하여 알아내고 비용을 높이는 요소를 찾아낼 수 있습니다.
- 고급 도구 구성: 선택적으로 조직을 위해 추가 세부 정보와 세부 수준을 제공하는 사용자 지정 도구를 생성할 수도 있습니다. 고급 분석 기능을 구현하는 데 Amazon Athena를사용하고 대시보드를 구성하는 데 Amazon QuickSight를사용할 수 있습니다. 사전 구성된 고급 대시보드에는 클라우드 인텔리전스 대시보드(CID)를 사용하는 것이 좋습니다. 또한 AWS 파트너와 협력하여 클라우드 관리 솔루션을 도입한다면 클라우드 청구 요금 모니터링과 최적화를 하나의 편리한 위치에서 활성화할 수 있습니다.

리소스

관련 문서:

- AWS 비용 관리
- AWS 리소스 태그 지정
- AWS Budgets를 통한 비용 분석
- Cost Explorer를 사용한 비용 분석
- AWS Cost and Usage Reports 관리
- AWS Cost Categories
- AWS를 통한 클라우드 재무 관리
- AWS APN 파트너 - 비용 관리

관련 동영상:

- 클라우드 인텔리전스 대시보드 배포

- FinOps 또는 비용 최적화 지표, KPI에 대한 알림 수신

관련 예시:

- Well-Architected 실습 - AWS 계정 설정
- Well-Architected 실습: 결제 시각화
- Well-Architected 실습: 비용 및 사용 거버넌스
- Well-Architected 실습: 비용 및 사용량 분석
- Well-Architected 실습: 비용 및 사용량 시각화
- Well-Architected 실습: Cloud Intelligence 대시보드

COST03-BP06 워크로드 지표를 기준으로 비용 할당

사용량 지표 또는 비즈니스 성과에 따라 워크로드의 비용을 할당하여 워크로드 비용 효율성을 측정합니다. 분석 서비스를 통해 비용 및 사용량 데이터를 분석하는 프로세스를 구현하면 인사이트와 차지백 기능을 제공할 수 있습니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 낮음

구현 가이드

비용 최적화는 최저 가격으로 비즈니스 성과를 거두는 것입니다. 이를 위해서는 워크로드 지표(워크로드 효율성으로 측정됨)를 기준으로 워크로드 비용을 할당해야 합니다. 로그 파일 또는 기타 애플리케이션 모니터링을 통해 정의된 워크로드 지표를 모니터링하십시오. 이 데이터를 워크로드 비용과 결합합니다. 워크로드 비용은 특정 태그 값 또는 계정 ID에 연결된 비용을 조회하여 확인할 수 있습니다. 이 분석은 시간 단위 수준으로 수행하는 것이 좋습니다. 요청 속도가 다양한 정적 비용 구성 요소(예: 백엔드 데이터베이스를 영구적으로 실행)가 있는 경우 일반적으로 효율성이 달라집니다(예: 사용량이 아침 9시에서 저녁 5시에 최고조에 달하고 야간에는 요청이 거의 없음). 정적 비용과 가변 비용 간의 관계를 이해하면 최적화 활동에 집중하는 데 도움이 됩니다.

공유 리소스에 대한 워크로드 지표를 생성하는 것은 Amazon Elastic Container Service(Amazon ECS) 및 Amazon API Gateway의 컨테이너화된 애플리케이션과 같은 리소스에 비해 어려울 수 있습니다. 하지만 사용량을 분류하고 비용을 추적할 수 있는 몇 가지 방법이 있습니다. Amazon ECS 및 AWS Batch 공유 리소스를 추적해야 하는 경우 AWS Cost Explorer에서 분할 비용 할당 데이터를 활성화할 수 있습니다. 분할 비용 할당 데이터를 사용하면 컨테이너화된 애플리케이션의 비용 및 사용량을 이해하여 최적화하고 공유 컴퓨팅 및 메모리 리소스가 소비되는 방식에 따라 개별 비즈니스 엔터티에 애플리케

이션 비용을 다시 할당할 수 있습니다. 공유 API Gateway 및 AWS Lambda 함수 사용량이 있는 경우 AWS Application Cost Profiler를 사용하여 ### ID 또는 ## ID#따라 소비량을 분류할 수 있습니다.

구현 단계

- 워크로드 지표에 비용 할당: 정의된 지표와 구성된 태그를 사용하여 워크로드 출력과 워크로드 비용을 결합하는 지표를 생성합니다. Amazon Athena 및 Amazon QuickSight와 같은 분석 서비스를 사용하여 전체 워크로드와 모든 구성 요소에 대한 효율성 대시보드를 생성합니다.

리소스

관련 문서:

- AWS 리소스 태그 지정
- AWS Budgets를 통한 비용 분석
- Cost Explorer를 사용한 비용 분석
- AWS Cost and Usage Reports 관리

관련 예시:

- AWS 분할 비용 할당 데이터를 사용하여 Amazon ECS 및 AWS Batch의 비용 가시성 향상

COST 4 리소스를 어떻게 폐기합니까?

프로젝트 시작부터 마지막까지의 전체 과정에서 변경 제어 및 리소스 관리를 구현합니다. 그러면 사용되지 않은 리소스를 종료하여 낭비되는 리소스를 줄일 수 있습니다.

모범 사례

- COST04-BP01 수명 주기에 걸친 리소스 추적
- COST04-BP02 폐기 프로세스 구현
- COST04-BP03 리소스 폐기
- COST04-BP04 리소스 자동 폐기
- COST04-BP05 데이터 보존 정책 적용

COST04-BP01 수명 주기에 걸친 리소스 추적

수명 주기 동안 리소스 및 리소스와 시스템의 관련성을 추적하는 방법을 정의하고 구현합니다. 태그 지정 기능을 사용하여 리소스의 워크로드나 기능을 파악할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

더 이상 필요하지 않은 워크로드 리소스를 폐기합니다. 일반적으로 테스트에 사용되는 리소스가 그 예입니다. 테스트가 완료된 후에는 리소스를 제거해도 됩니다. 태그를 사용하여 리소스를 추적(및 해당 태그에서 보고서를 실행)하면 사용되지 않거나 해당 라이선스가 만료될 예정이므로 폐기할 자산을 식별할 수 있습니다. 태그를 사용하면 리소스에 기능별 레이블을 지정하거나 폐기할 수 있는 알려진 날짜를 지정하여 리소스를 효과적으로 추적할 수 있습니다. 그런 다음 이러한 태그에 대해 보고를 실행할 수 있습니다. 예를 들어 기능 태그를 지정할 때는 워크로드 수명 주기 측면에서 리소스의 목적을 식별하는 `feature-X testing`과 같은 값을 지정할 수 있습니다. 또 다른 예는 리소스에 대해 `LifeSpan` 또는 TTL을 사용하는 것으로, 폐기 기간이나 특정 시간을 정의하는 to-be-deleted 태그 키 이름 및 값이 이에 해당합니다.

구현 단계

- 태그 지정 체계 구현: 리소스가 속한 워크로드를 식별하는 태그 지정 체계를 구현하여 워크로드 내의 모든 리소스에 태그가 적절히 지정되는지 확인합니다. 태그 지정을 사용하면 목적, 팀, 환경 또는 비즈니스에 맞는 다른 기준에 따라 리소스를 분류할 수 있습니다. 태그 지정 사용 사례, 전략 및 기법에 대한 자세한 내용은 AWS 태그 지정 모범 사례를 참조하세요.
- 워크로드 처리량 또는 출력 모니터링 구현: 워크로드 처리량 모니터링 또는 경보를 구현하여 입력 요청 또는 출력 완료 시 트리거합니다. 워크로드 요청 또는 출력이 0으로 떨어질 때 즉, 워크로드 리소스가 더 이상 사용되지 않을 때 알림을 제공하도록 구성합니다. 워크로드가 정상 조건에서 주기적으로 0으로 떨어질 경우 시간 계수를 통합합니다. 사용되지 않거나 사용률이 낮은 리소스에 대한 자세한 내용은 AWS Trusted Advisor 비용 최적화 점검을 참조하세요.
- AWS 리소스 그룹화: AWS 리소스의 그룹을 생성합니다. AWS Resource Groups을 사용하여 동일한 AWS 리전에 있는 AWS 리소스를 구성하고 관리할 수 있습니다. 조직 내 리소스를 식별하고 정렬하는 데 도움이 되도록 대부분의 리소스에 태그를 추가할 수 있습니다. Tag Editor를 사용하여 지원되는 리소스에 태그를 일괄적으로 추가할 수 있습니다. AWS Service Catalog을 사용하여 최종 사용자에게 승인된 제품의 포트폴리오를 생성, 관리 및 배포하고 제품 수명 주기를 관리할 수 있습니다.

리소스

관련 문서:

- AWS Auto Scaling
- AWS Trusted Advisor
- AWS Trusted Advisor 비용 최적화 점검
- AWS 리소스 태그 지정
- 사용자 지정 지표 게시

관련 동영상:

- AWS Trusted Advisor 사용을 통한 비용 최적화 방법

관련 예시:

- AWS 리소스 구성
- AWS Trusted Advisor 사용을 통한 비용 최적화

COST04-BP02 폐기 프로세스 구현

미사용 리소스를 식별하고 폐기하는 프로세스를 구현합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

조직 전체에서 미사용 리소스를 식별하고 제거하는 표준화된 프로세스를 구현합니다. 이 프로세스에서는 검색 수행 빈도와 리소스를 제거하여 모든 조직 요구 사항이 충족되는지 확인하는 프로세스를 정의해야 합니다.

구현 단계

- 폐기 프로세스 생성 및 구현: 워크로드 개발자 및 소유자와 협력하여 워크로드와 해당 리소스에 대한 폐기 프로세스를 구축합니다. 이 프로세스에서는 워크로드가 사용 중인지, 그리고 각 워크로드 리소스가 사용 중인지 확인하는 방법을 다룹니다. 규제 요구 사항을 준수하면서 리소스를 폐기하고 리소스를 서비스에서 제거하는 데 필요한 단계를 자세히 알아봅니다. 라이선스 또는 연결된 스토리지와 같은 관련 리소스도 다룹니다. 워크로드 소유자에게 폐기 프로세스가 실행되었음을 알립니다.

  다음 폐기 단계를 사용하여 프로세스의 일부로 무엇을 확인해야 하는지 알 수 있습니다.

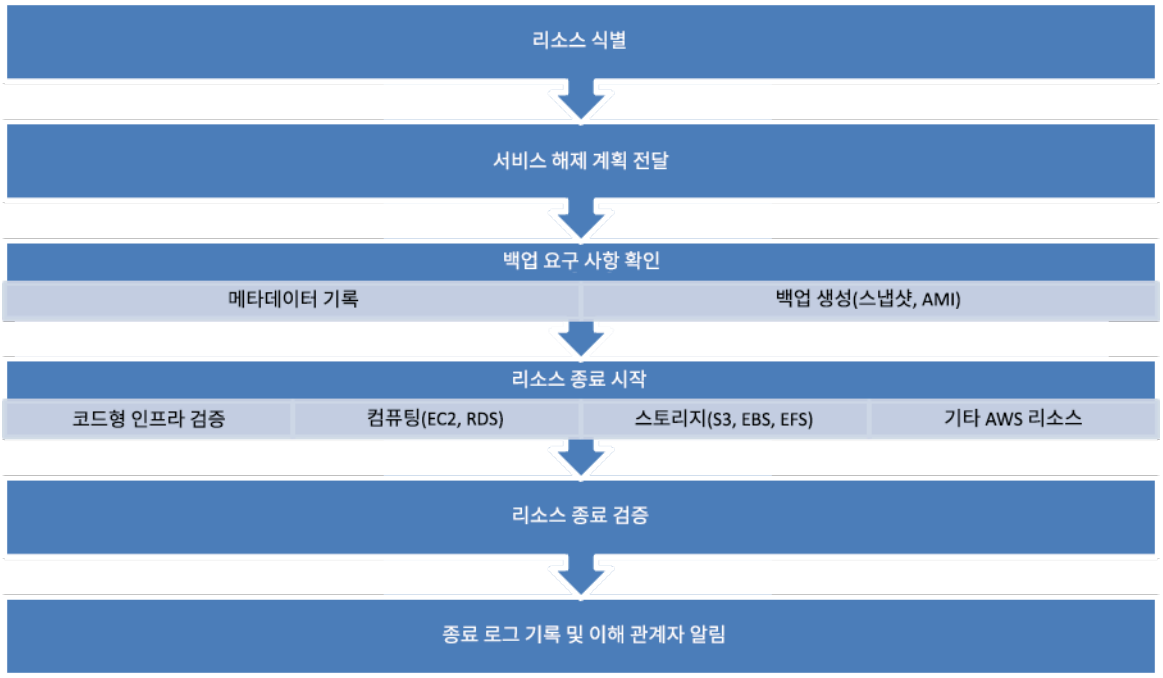
- 폐기해야 할 리소스 식별: AWS 클라우드에서 폐기 대상인 리소스를 식별합니다. 필요한 모든 정보를 기록하고 폐기를 예약합니다. 타임라인에서 프로세스 중 예상치 못한 문제가 발생했는지 여부와 시기를 반드시 설명합니다.
- 조정 및 의사 소통: 워크로드 소유자와 협력하여 폐기할 리소스를 확인합니다.
- 메타데이터 기록 및 백업 생성: 프로덕션 환경의 리소스에 필요하거나 중요한 리소스인 경우 메타데이터(예: 퍼블릭 IP, 리전, AZ, VPC, 서브넷 및 보안 그룹)를 기록하고 백업(예: Amazon Elastic Block Store 스냅샷 및 AMI, 키 내보내기 및 인증서 내보내기 실행)을 생성합니다.
- 코드형 인프라 검증: 리소스가 AWS CloudFormation, Terraform, AWS Cloud Development Kit (AWS CDK) 또는 기타 코드형 인프라와 함께 배포되어 필요한 경우 다시 배포할 수 있도록 할지 여부를 결정합니다.
- 액세스 방지: 일정 기간 동안 제한적 제어를 적용하여 리소스가 필요한지 여부를 결정하는 동안 리소스를 사용하지 못하도록 합니다. 필요한 경우 리소스 환경을 원래 상태로 되돌릴 수 있는지 확인합니다.
- 내부 폐기 프로세스 준수: 조직의 관리 작업 및 폐기 프로세스를 따릅니다. 여기에는 조직 도메인에서 리소스 제거, DNS 레코드 제거, 그리고 구성 관리 도구, 모니터링 도구, 자동화 도구 및 보안 도구에서 리소스 제거 등이 있습니다.

리소스가 Amazon EC2 인스턴스인 경우 다음 목록을 참조합니다. 자세한 내용은 Amazon EC2 리소스를 삭제하거나 종료하려면 어떻게 해야 하나요?를 참조하세요.

- 모든 Amazon EC2 인스턴스 및 로드 밸런서를 중지하거나 종료합니다. Amazon EC2 인스턴스는 종료된 후 잠시 콘솔에 표시됩니다. 실행 중 상태가 아닌 인스턴스에 대해서는 요금이 부과되지 않습니다.
- Auto Scaling 인프라를 삭제합니다.
- 모든 전용 호스트를 해제합니다.
- 모든 Amazon EBS 볼륨 및 Amazon EBS 스냅샷을 삭제합니다.
- 모든 탄력적 IP 주소를 해제합니다.
- 모든 Amazon Machine Image(AMI) 등록을 취소합니다.
- 모든 AWS Elastic Beanstalk 환경을 종료합니다.

리소스가 Amazon S3 Glacier 스토리지의 객체이고 최소 스토리지 기간을 충족하기 전에 아카이브를 삭제하면 비례 할당으로 계산된 조기 삭제 요금이 청구됩니다. Amazon S3 Glacier의 최소 스토리지 기간은 사용된 스토리지 클래스에 따라 다릅니다. 각 스토리지 클래스의 최소 스토리지 기간에 대한 요약은 Amazon S3 스토리지 클래스 전반에 걸친 성능을 참조하세요. 조기 삭제 요금의 계산 방법에 대한 자세한 내용은 Amazon S3 요금을 참조하세요.

다음 간단한 폐기 프로세스 순서도는 폐기 단계를 간략히 보여줍니다. 리소스를 폐기하기 전에 폐기 대상으로 식별된 리소스가 조직에서 사용하지 않는 것이 맞는지 확인해야 합니다.



리소스 폐기 흐름.

리소스

관련 문서:

- AWS Auto Scaling
- AWS Trusted Advisor
- AWS CloudTrail

관련 동영상:

- CloudFormation 스택을 삭제하고 일부 리소스 유지
- Amazon EC2 인스턴스를 시작한 사용자 확인

관련 예시:

- Amazon EC2 리소스 삭제 또는 종료
- Amazon EC2 인스턴스를 시작한 사용자 확인

COST04-BP03 리소스 폐기

정기적 감사 또는 사용량 변화와 같은 이벤트에 의해 트리거된 리소스를 폐기합니다. 폐기는 일반적으로 주기적으로 수행되며 수동 또는 자동으로 수행할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

사용되지 않는 리소스를 검색하는 빈도 및 작업은 잠재적 절감을 고려하여 결정해야 합니다. 따라서 비용이 높은 계정은 비용이 낮은 계정보다 더 자주 분석되어야 합니다. 워크로드의 상태 변경(예: 제품 EOL 또는 교체)을 통해 검색 및 폐기 이벤트를 트리거할 수 있습니다. 검색 및 폐기 이벤트는 시장 상황의 변화나 제품 종료와 같은 외부 이벤트에 의해 트리거될 수도 있습니다.

구현 단계

- 리소스 폐기: 더 이상 필요하지 않거나 라이선스 계약이 종료되는 AWS 리소스의 감가상각 단계입니다. 스냅샷 생성 또는 백업과 같은 원치 않는 중단이 발생하지 않도록, 폐기 단계나 리소스 폐기로 진행하기 전에 최종 확인을 완료해야 합니다. 폐기 프로세스를 사용하여, 사용되지 않는 것으로 식별된 각 리소스를 폐기합니다.

리소스

관련 문서:

- AWS Auto Scaling
- AWS Trusted Advisor

관련 예시:

- Well-Architected 실습: 리소스 폐기(레벨 100)

COST04-BP04 리소스 자동 폐기

중요하지 않은 리소스, 필수가 아닌 리소스 또는 사용률이 낮은 리소스를 파악하고 폐기하는 과정에서 워크로드가 리소스 종료를 정상적으로 처리하도록 설계합니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

자동화를 사용하여 폐기 프로세스와 관련된 비용을 줄이거나 제거합니다. 자동 폐기를 수행하도록 워크로드를 설계하면 수명 주기 동안 전체 워크로드 비용을 절감할 수 있습니다. AWS Auto Scaling을 사용하여 폐기 프로세스를 수행할 수 있습니다. API 또는 SDK를 사용하여 워크로드 리소스를 자동으로 폐기하는 사용자 지정 코드를 구현할 수도 있습니다.

현대적 애플리케이션은 서버리스 우선으로 구축되며, 이는 서버리스 서비스 도입을 우선시하는 전략입니다. AWS는 컴퓨팅, 통합 및 데이터 스토어라는 3개의 모든 스택 계층에 대해 서버리스 서비스를 개발했습니다. 서버리스 아키텍처를 사용하면 자동 확장 및 축소를 통해 트래픽이 적은 기간 동안 비용을 절감할 수 있습니다.

구현 단계

- AWS Auto Scaling 구현: 지원되는 리소스의 경우 AWS Auto Scaling을 사용하여 리소스를 구성합니다. AWS Auto Scaling은 AWS 서비스 사용 시 사용률과 비용 효율을 최적화할 수 있도록 지원합니다. 수요가 낮아지면 AWS Auto Scaling은 과도한 지출을 방지할 수 있도록 모든 초과 리소스 용량을 자동으로 제거합니다.
- CloudWatch를 구성하여 인스턴스 종료: CloudWatch 경보를 사용하여 인스턴스를 종료하도록 구성할 수 있습니다. 폐기 프로세스의 지표를 사용하여 Amazon Elastic Compute Cloud 작업으로 경보를 구현합니다. 롤아웃하기 전에 비 프로덕션 환경에서 작업을 확인합니다.
- 워크로드 내에서 코드 구현: AWS SDK 또는 AWS CLI를 사용하여 워크로드 리소스를 폐기할 수 있습니다. 애플리케이션 내에서 AWS와 통합되고, 더 이상 사용되지 않는 리소스를 종료하거나 제거하는 코드를 구현합니다.
- 서버리스 서비스 사용: 애플리케이션을 구축하고 실행하기 위해 AWS에 서버리스 아키텍처와 이벤트 중심 아키텍처를 구축하는 우선순위를 지정합니다. AWS는 자동으로 최적화된 리소스 사용률과 자동 폐기(스케일 인 및 스케일 아웃)를 본질적으로 제공하는 여러 서버리스 기술 서비스를 제공합니다. 서버리스 애플리케이션을 통해 리소스 사용률이 자동으로 최적화되고 과다 프로비저닝에 대한 비용을 지불할 필요가 없습니다.

리소스

관련 문서:

- AWS Auto Scaling
- AWS Trusted Advisor
- AWS의 서버리스

- 인스턴스를 중지, 종료, 재부팅 또는 복구하는 경보 생성
- Amazon EC2 Auto Scaling 시작하기
- Amazon CloudWatch 경보에 종료 작업 추가

관련 예시:

- AWS CloudFormation 스택 자동 삭제 일정 관리
- Well-Architected 실습 – 리소스 자동 폐기(레벨 100)
- Servian AWS 자동 정리

COST04-BP05 데이터 보존 정책 적용

조직의 요구 사항에 따라 객체 삭제를 처리할 수 있도록 지원되는 리소스에 대한 데이터 보존 정책을 정의합니다. 더 이상 필요 없는 불필요하거나 분리된 리소스와 객체를 파악하여 삭제합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

데이터 보존 정책 및 수명 주기 정책을 사용하여 폐기 프로세스 관련 비용과 식별된 리소스의 스토리지 비용을 줄입니다. 자동화된 스토리지 클래스 마이그레이션 및 삭제를 수행하기 위한 데이터 보존 정책 및 수명 주기 정책을 정의하면 수명 주기 중 전반적인 스토리지 비용을 줄일 수 있습니다. Amazon Data Lifecycle Manager를 사용하여 Amazon Elastic Block Store 스냅샷 및 Amazon EBS 지원 Amazon Machine Image(AMI)의 생성 및 삭제를 자동화하고, Amazon S3 Intelligent-Tiering 또는 Amazon S3 수명 주기 구성을 사용하여 Amazon S3 객체의 수명 주기를 관리할 수 있습니다. 또한 API 또는 SDK를 사용하여 사용자 지정 코드를 구현해 객체가 자동으로 삭제되도록 수명 주기 정책 및 정책 규칙을 생성할 수 있습니다.

구현 단계

- Amazon Data Lifecycle Manager 사용: Amazon Data Lifecycle Manager에 대한 수명 주기 정책을 사용하여 Amazon EBS 스냅샷 및 Amazon EBS 지원 AMI의 삭제를 자동화합니다.
- 버킷에 대한 수명 주기 구성 설정: 버킷에 대한 Amazon S3 수명 주기 구성을 사용하여 객체 수명 주기 중 Amazon S3에서 취할 조치를 정의하고 비즈니스 요구 사항에 따라 객체의 수명 주기 종료 시 삭제하도록 정의할 수 있습니다.

리소스

관련 문서:

- AWS Trusted Advisor
- Amazon Data Lifecycle Manager
- Amazon S3 버킷에서 수명 주기 구성 설정 방법

관련 동영상:

- Automate Amazon EBS Snapshots with Amazon Data Lifecycle Manager(DLM으로 Amazon EBS 스냅샷 자동화)
- Empty an Amazon S3 bucket using a lifecycle configuration rule(수명 주기 구성 규칙을 사용하여 S3 버킷 비우기)

관련 예시:

- Empty an Amazon S3 bucket using a lifecycle configuration rule(수명 주기 구성 규칙을 사용하여 S3 버킷 비우기)
- Well-Architected 실습: 리소스 자동 폐기(레벨 100)

비용 효율적인 리소스

질문

- COST 5 서비스를 선택할 때 비용을 어떻게 평가합니까?
- COST 6 리소스 유형, 크기 및 수 선택을 통해 비용 목표를 어떻게 달성합니까?
- COST 7 비용 절감을 위해 가격 책정 모델을 어떻게 사용합니까?
- COST 8 데이터 전송 요금을 위한 계획은 어떻게 합니까?

COST 5 서비스를 선택할 때 비용을 어떻게 평가합니까?

Amazon EC2, Amazon EBS 및 Amazon S3는 기본 구성 AWS 서비스입니다. Amazon RDS 및 Amazon DynamoDB와 같은 관리형 서비스는 더 높은 수준이거나 애플리케이션 수준의 AWS 서비스입니다. 기본 구성 서비스와 관리형 서비스를 적절히 선택하여 이 워크로드의 비용을 최적화할 수 있습니다. 예를 들어 관리형 서비스를 사용하면 관리 및 운영 고정 비용을 상당 부분 줄이고, 응용 프로그램 및 비즈니스 관련 활동에 집중할 수 있습니다.

모범 사례

- COST05-BP01 조직의 비용 요구 사항 파악
- COST05-BP02 이 워크로드의 모든 구성 요소 분석
- COST05-BP03 각 구성 요소의 철저한 분석 수행
- COST05-BP04 비용 효율적인 라이선스가 포함된 소프트웨어 선택
- COST05-BP05 조직의 우선순위에 따라 비용을 최적화할 이 워크로드의 구성 요소 선택
- COST05-BP06 시간별로 사용량이 달라지는 경우 비용 분석 수행

COST05-BP01 조직의 비용 요구 사항 파악

팀원과 협의하여 이 워크로드의 비용 최적화와 기타 원칙(예: 성능, 안정성)의 적절한 절충 수준을 정의합니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

워크로드에 사용할 서비스를 선택할 때는 조직의 우선 순위를 이해하는 것이 중요합니다. 비용 및 기타 Well-Architected 원칙(예: 성능 및 안정성) 간에 균형을 유지해야 합니다. 완전한 비용 최적화 워크로드는 조직의 요구 사항과 가장 일치하는 솔루션을 의미하며 꼭 비용이 가장 낮은 솔루션이 아닐 수 있습니다. 조직 내 모든 팀과 회의를 통해 제품, 비즈니스, 기술, 재무 등의 정보를 수집하세요.

구현 단계

- 조직의 비용 요구 사항 파악: 제품 관리 팀, 애플리케이션 소유자, 개발 및 운영 팀, 관리 및 재무 팀 등 조직의 팀원을 만나 보세요. 이 워크로드와 해당 구성 요소에 대해 Well-Architected 원칙의 우선 순위를 정합니다. 원칙 목록을 순서대로 나열합니다. 각 원칙에 가중치를 더할 수도 있습니다. 이를 통해 원칙에 얼마나 더 많은 초점이 맞춰져 있는지 또는 두 원칙 사이에 초점이 얼마나 비슷한지 알 수 있습니다.

리소스

관련 문서:

- AWS 총 소유 비용(TCO) 계산기
- Amazon S3 스토리지 클래스
- 클라우드 제품

COST05-BP02 이 워크로드의 모든 구성 요소 분석

현재 크기나 비용과 관계없이 모든 워크로드 구성 요소가 분석되도록 해야 합니다. 검토 작업은 현재 비용과 예상 비용 등의 제공될 수 있는 이점을 반영해야 합니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

워크로드의 모든 구성 요소에 대해 철저한 분석을 수행합니다. 분석 비용과 수명 주기 동안 워크로드의 잠재적 절감액이 균형을 이루는지 확인하십시오. 구성 요소의 현재 영향과 향후 잠재적 영향을 파악해야 합니다. 예를 들어 제안된 리소스의 비용이 월 10 USD이고 예상 로드가 월 15 USD를 초과하지 않는 경우, 시스템 수명 전체에서 얻을 수 있는 잠재적 이점보다 비용을 50%(월 5 USD) 절감하기 위한 하루 동안의 노력이 더 들 수 있습니다. 더 빠르고 효율적인 데이터 기반 추정을 사용하면 이 구성 요소에 대해 전반적으로 가장 좋은 결과를 얻을 수 있습니다.

워크로드는 시간이 지남에 따라 변경될 수 있으며 워크로드 아키텍처 또는 사용량이 변경되면 워크로드에 가장 적합한 서비스 세트도 변경될 수 있습니다. 서비스 선택을 분석할 때는 현재 및 향후 워크로드 상태 및 사용량 수준을 포함해야 합니다. 향후 워크로드 상태 또는 사용량에 대한 서비스를 구현하면 향후 변경에 필요한 노력을 줄이거나 제거하여 전반적인 비용을 절감할 수 있습니다.

AWS Cost Explorer 및 AWS Cost and Usage Report (CUR)을 사용하여 PoC(개념 증명) 또는 환경 실행 비용을 분석할 수 있습니다. 또한 AWS Pricing Calculator 로 워크로드 비용을 추정할 수도 있습니다.

구현 단계

- 워크로드 구성 요소 나열: 모든 워크로드 구성 요소의 목록을 만듭니다. 이 목록은 각 구성 요소가 분석되었는지 확인하는 데 사용됩니다. 여기에 드는 노력은 조직의 우선순위에 정의된 워크로드에 대한 중요도를 반영해야 합니다. 리소스를 그룹화하면 데이터베이스가 여러 개인 경우 프로덕션 데이터베이스 스토리지와 같이 효율성이 향상됩니다.
- 구성 요소 목록의 우선순위 지정: 구성 요소 목록을 가져와서 작업량순으로 우선순위를 지정합니다. 이는 일반적으로 구성 요소의 비용(가장 비싼 것부터 가장 싼 것까지) 또는 조직 우선순위에 정의된 중요도를 따릅니다.
- 분석 수행: 목록의 각 구성 요소에 대해 사용 가능한 옵션과 서비스를 검토하고 조직의 우선순위에 가장 잘 맞는 옵션을 선택합니다.

리소스

관련 문서:

- AWS Pricing Calculator
- AWS Cost Explorer
- Amazon S3 스토리지 클래스
- 클라우드 제품

COST05-BP03 각 구성 요소의 철저한 분석 수행

조직에서 발생하는 각 구성 요소의 전반적인 비용을 확인합니다. 그런 다음 운영 및 관리 비용을 감안하여 총 소유 비용을 계산합니다(특히, 클라우드 제공업체에서 관리형 서비스를 사용하는 경우). 검토 작업은 잠재적 이점을 반영해야 합니다(예를 들어 분석에 소요된 시간이 구성 요소의 비용에 비례).

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

팀에서는 이렇게 절약된 시간을 기술적 부채 청산, 혁신 및 부가 가치 기능과 비즈니스를 차별화할 수 있는 요인을 만드는 데 집중할 수 있다는 점을 고려하세요. 예를 들어 온프레미스 환경에서 클라우드로 데이터베이스를 최대한 빠르게 리프트 앤드 시프트(리호스팅이라고도 함)하고 최적화는 나중에 수행해야 할 수 있습니다. AWS에서 라이선스 비용이 거의 발생하지 않거나, 전혀 발생하지 않을 수 있는 관리형 서비스를 사용하여 실현한 비용 절감 이점을 파악하는 것이 좋습니다. AWS의 관리형 서비스는 서비스 유지 관리를 위한 운영 및 관리 부담(예: OS 패치 적용 또는 업그레이드)을 없애 혁신 및 비즈니스에 집중할 수 있도록 합니다.

관리형 서비스는 클라우드 규모로 운영되므로 거래 또는 서비스당 비용을 절감할 수 있습니다. 애플리케이션의 핵심 아키텍처를 변경하지 않고 실질적인 이점을 달성하기 위해 잠재적인 최적화를 수행할 수 있습니다. 예를 들어, Amazon Relational Database Service(Amazon RDS) 등과 같은 서비스형 데이터베이스 플랫폼으로 마이그레이션하거나 AWS Elastic Beanstalk 등과 같은 완전관리형 플랫폼으로 애플리케이션을 마이그레이션하여 데이터베이스 인스턴스를 관리하는 데 소요되는 시간을 단축할 수 있습니다.

관리형 서비스에는 대개 충분한 용량을 보장하기 위해 설정할 수 있는 속성이 있습니다. 초과 용량을 최소한으로 유지하면서 성능은 극대화할 수 있도록 이러한 속성을 설정하고 모니터링해야 합니다. AWS Management Console 또는 AWS API 및 SDK를 사용해 AWS Managed Services의 속성을 수정하여 변화하는 수요에 맞게 리소스 요구를 조정할 수 있습니다. 예를 들어, Amazon EMR 클러스터(또는 Amazon Redshift 클러스터)의 노드 수를 늘리거나 줄여서 클러스터 규모를 스케일 아웃하거나 스케일 인할 수 있습니다.

또한 AWS 리소스의 여러 인스턴스를 압축하여 리소스 사용 밀도를 높일 수도 있습니다. 예를 들어 단일 Amazon Relational Database Service(Amazon RDS) 데이터베이스 인스턴스에 소형 데이터베이스

여러 개를 프로비저닝할 수 있습니다. 그리고 사용량이 증가하면 스냅샷 및 복원 프로세스를 사용하여 데이터베이스 중 하나를 전용 Amazon RDS 데이터베이스 인스턴스로 마이그레이션할 수 있습니다.

관리형 서비스에서 워크로드를 프로비저닝할 때는 서비스 용량 조정 요구 사항을 파악해야 합니다. 일반적으로 이러한 요구 사항은 시간, 작업량 및 정상 워크로드 작동에 미치는 영향을 의미합니다. 리소스를 프로비저닝할 때는 변경이 발생하기까지 소요되는 시간을 고려하여 이 시간을 허용하는 데 필요한 오버헤드를 프로비저닝해야 합니다. Amazon CloudWatch 등의 시스템 및 모니터링 도구와 통합된 API와 SDK를 사용하면 서비스를 수정하는 데 필요한 지속적인 작업을 사실상 수행하지 않아도 됩니다.

Amazon RDS, Amazon Redshift 및 Amazon ElastiCache에서는 관리형 데이터베이스 서비스를 제공합니다. Amazon Athena, Amazon EMR 및 Amazon OpenSearch Service에서는 관리형 분석 서비스를 제공합니다.

AMS는 엔터프라이즈 고객 및 파트너를 대신하여 AWS 인프라를 운영하는 서비스입니다. 이 서비스를 사용하면 규정을 준수하는 안전한 환경에 워크로드를 배포할 수 있습니다. AMS는 자동화가 포함된 엔터프라이즈 클라우드 운영 모델을 사용하므로 고객은 조직 요구 사항을 충족하면서 클라우드로 더 빠르게 이전하고 지속적인 관리 비용을 절감할 수 있습니다.

구현 단계

- 철저한 분석 수행: 구성 요소 목록을 사용하여 가장 높은 우선순위부터 가장 낮은 우선순위까지 각 구성 요소를 살펴봅니다. 우선순위가 높고 비용이 많이 드는 구성 요소의 경우 추가 분석을 수행하고 사용 가능한 모든 옵션과 장기적인 영향을 평가합니다. 우선순위가 낮은 구성 요소의 경우 사용량 변화로 인해 구성 요소의 우선순위가 변경되는지 평가한 다음 적절한 작업에 대한 분석을 수행합니다.
- 관리형 및 비관리형 리소스 비교: 관리하는 리소스의 운영 비용을 고려하고 AWS 관리형 리소스와 비교합니다. 예를 들어, Amazon EC2 인스턴스에서 실행 중인 데이터베이스를 검토한 다음 Amazon EC2에서 Apache Spark를 실행하는 것에 비해 Amazon RDS 옵션(AWS 관리형 서비스) 또는 Amazon EMR와 비교합니다. 자체 관리형 워크로드에서 AWS 완전관리형 워크로드로 이전하는 경우 옵션을 주의해서 살펴보세요. 여기서 고려해야 할 가장 중요한 세 가지 요소는 사용하려는 관리형 서비스의 유형, 데이터 마이그레이션에 사용할 프로세스 및 AWS 공동 책임 모델에 대한 이해도입니다.

리소스

관련 문서:

- AWS 총 소유 비용(TCO) 계산기

- Amazon S3 스토리지 클래스
- AWS 클라우드 제품
- AWS 공동 책임 모델

관련 동영상:

- Why move to a managed database?(관리형 데이터베이스로 이전하는 이유)
- What is Amazon EMR and how can I use it for processing data?(Amazon EMR이란 무엇이며, 데이터 처리에 어떻게 사용할 수 있나요?)

관련 예시:

- 관리형 데이터베이스 서비스로 이전하는 이유
- AWS DMS를 사용하여 동일한 SQL Server 데이터베이스의 데이터를 단일 Amazon RDS for SQL Server 데이터베이스로 통합
- 데이터를 Amazon Managed Streaming for Apache Kafka(Amazon MSK)에 대규모로 제공
- ASP.NET 웹 애플리케이션을 AWS Elastic Beanstalk에 마이그레이션

COST05-BP04 비용 효율적인 라이선스가 포함된 소프트웨어 선택

오픈 소스 소프트웨어는 워크로드 비용에서 상당한 부분을 차지할 수 있는 소프트웨어 라이선스 비용을 없앱니다. 라이선스가 부여된 소프트웨어가 필요한 경우 CPU와 같은 임의의 속성에 바인딩된 라이선스를 피하고 결과 또는 성과에 바인딩된 라이선스를 찾으십시오. 이러한 라이선스의 비용은 해당 라이선스가 제공하는 혜택에 더 근접하게 조정됩니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

오픈 소스 소프트웨어를 사용하면 소프트웨어 라이선스 비용을 줄일 수 있습니다. 라이선스 비용은 워크로드 규모가 확장됨에 따라 워크로드 비용에 상당한 영향을 미칠 수 있습니다. 총 비용을 기준으로 라이선스가 부여된 소프트웨어의 이점을 측정하면 워크로드 효율성을 최대한 최적화할 수 있습니다. 라이선스 변경 사항과 이러한 변경 사항이 워크로드 비용에 미치는 영향을 모델링하십시오. 공급업체가 데이터베이스 라이선스 비용을 변경하는 경우 이 비용이 워크로드의 전반적인 효율성에 어떤 영향을 미치는지 조사하십시오. 공급업체의 기간별 요금 발표를 고려하여 제품 전반의 라이선스 변경 추세를 파악하십시오. 또한 라이선스 비용은 하드웨어에 따라 확장되는 라이선스(CPU 바인딩 라이선스)와

같이 처리량이나 사용량과 관계없이 확장될 수 있습니다. 이러한 라이선스는 해당하는 결과 없이 비용이 빠르게 증가할 수 있으므로 피해야 합니다.

구현 단계

- 라이선스 옵션 분석: 사용 가능한 소프트웨어의 라이선스 약관을 검토합니다. 필요한 기능을 갖춘 오픈 소스 버전을 찾고 라이선스가 부여된 소프트웨어의 혜택이 비용보다 큰지 확인하세요. 좋은 조건은 소프트웨어에서 제공하는 혜택에 소프트웨어 비용을 맞춥니다.
- 소프트웨어 공급자 분석: 공급자의 과거 요금 또는 라이선스 변경을 검토합니다. 특정 공급자의 하드웨어 또는 플랫폼에서 실행에 대한 징벌적 조건과 같이 성과에 부합하지 않는 변경 사항을 찾아보십시오. 또한 감사를 수행하는 방법과 부과될 수 있는 페널티를 찾아보세요.

리소스

관련 문서:

- AWS 총 소유 비용(TCO) 계산기
- Amazon S3 스토리지 클래스
- 클라우드 제품

COST05-BP05 조직의 우선순위에 따라 비용을 최적화할 이 워크로드의 구성 요소 선택

워크로드에 대한 모든 구성 요소를 선택할 때는 비용을 고려해야 합니다. 여기에는 애플리케이션 수준 및 관리형 서비스 또는 서버리스, 컨테이너 또는 이벤트 기반 아키텍처를 사용한 전반적인 비용 절감이 포함됩니다. 오픈 소스 소프트웨어, 라이선스 요금이 없는 소프트웨어 또는 비용 절감을 위한 대안을 사용하여 라이선스 비용을 최소화합니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

모든 구성 요소를 선택할 때 서비스 및 옵션 비용을 고려합니다. 이 과정에서는 Amazon Relational Database Service(Amazon RDS), Amazon DynamoDB, Amazon Simple Notification Service(Amazon SNS) 및 Amazon Simple Email Service(Amazon SES) 등의 애플리케이션 수준 서비스와 관리형 서비스를 사용하여 전체적인 조직 비용을 절감할 수 있습니다. 컴퓨팅 구성 요소의 경우에는 서버리스 서비스와 컨테이너를 사용합니다(예: AWS Lambda 및 정적 웹사이트용 Amazon Simple Storage Service(Amazon S3)). 가능한 경우 애플리케이션을 컨테이너화하고 AWS 관리형 컨테이너 서비스(예: Amazon Elastic Container Service(Amazon ECS) 또는 Amazon Elastic Kubernetes Service(Amazon

EKS))를 사용합니다. 오픈 소스 소프트웨어 또는 라이선스 요금이 없는 소프트웨어를 사용하여 라이선스 비용을 최소화합니다(예: 컴퓨팅 워크로드용 Amazon Linux 또는 Amazon Aurora(으)로 데이터베이스 마이그레이션).

AWS Lambda, Amazon Simple Queue Service(Amazon SQS), Amazon SNS 및 Amazon SES 등과 같은 서버리스 또는 애플리케이션 수준 서비스를 사용할 수 있습니다. 이러한 서비스를 사용하면 리소스를 관리할 필요가 없으며 코드 실행, 대기열 서비스 및 메시지 전송 기능을 제공합니다. 또 다른 이점은 사용량에 따라 성능과 비용이 확장되므로 효율적인 비용 할당 및 귀속이 가능하다는 것입니다.

서버리스 서비스와 함께 이벤트 기반 아키텍처(EDA)도 사용할 수 있습니다. 이벤트 기반 아키텍처는 푸시 기반이므로 이벤트가 라우터에 나타날 때 모든 것이 온디맨드 방식으로 발생합니다. 이렇게 하면 이벤트가 있는지 확인하기 위한 지속적인 폴링 비용을 지불하지 않습니다. 즉, 네트워크 대역폭 소비, CPU 사용률, 유휴 폴릿 용량, SSL/TLS 핸드셰이크가 줄어듭니다.

서버리스에 대한 자세한 내용은 Well-Architected 서버리스 애플리케이션 렌즈 백서를 참조하세요.

구현 단계

- 비용을 최적화할 각 서비스 선택: 우선순위가 지정된 목록 및 분석을 사용하여 조직의 우선순위와 가장 잘 일치하는 각 옵션을 선택합니다. 수요를 충족하기 위해 용량을 늘리는 대신 더 저렴한 비용으로 더 뛰어난 성능을 선사할 수 있는 다른 옵션을 고려해 보세요. 예를 들어, AWS에서 데이터베이스에 대해 예상되는 트래픽을 검토한 다음 인스턴스 크기를 늘리거나 Amazon ElastiCache 서비스(Redis 또는 Memcached)를 사용하여 데이터베이스에 캐시된 메커니즘을 제공할지 검토해야 합니다.
- 이벤트 기반 아키텍처 평가: 서버리스 아키텍처를 사용하면 분산된 마이크로서비스 기반 애플리케이션을 위한 이벤트 기반 아키텍처를 구축할 수도 있습니다. 그러면 확장 가능하고 복원력이 있으며 민첩하고 비용 효과적인 솔루션을 구축하는 데 도움이 됩니다.

리소스

관련 문서:

- AWS 총 소유 비용(TCO) 계산기
- AWS 서버리스
- 이벤트 기반 아키텍처란 무엇인가요?
- Amazon S3 스토리지 클래스
- 클라우드 제품
- Amazon ElastiCache (Redis OSS)

관련 예시:

- 이벤트 기반 아키텍처 시작하기
- 이벤트 기반 아키텍처란 무엇인가요?
- Amazon ElastiCache (Redis OSS)를 사용하여 Statsig가 100배 더 비용 효과적으로 실행하는 방법
- AWS Lambda 함수 작업의 모범 사례

COST05-BP06 시간별로 사용량이 달라지는 경우 비용 분석 수행

워크로드는 시간이 지남에 따라 바뀔 수 있습니다. 일부 서비스 또는 기능은 다양한 사용 수준에서 더 비용 효율적입니다. 예상 사용량에 따라 시간별로 각 구성 요소 분석을 수행하면 수명 주기 동안 워크로드를 비용 효과적으로 유지할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

AWS에서 새로운 서비스와 기능이 릴리스되면 워크로드에 대한 최적의 서비스가 변경될 수 있습니다. 필요한 노력에는 잠재적 이점이 반영되어야 합니다. 워크로드 검토 빈도는 조직의 요구 사항에 따라 다릅니다. 비용이 높은 워크로드인 경우 새로운 서비스를 빨리 구현할수록 비용 절감이 극대화되므로 검토를 자주 수행하는 것이 좋습니다. 사용량 패턴이 변경되는 경우에도 검토를 시행해야 합니다. 사용량의 큰 변화는 대체 서비스가 더 적합하다는 의미일 수 있습니다.

데이터를 AWS 클라우드로 이전해야 하는 경우 데이터 세트가 파일, 데이터베이스, 머신 이미지, 블록 볼륨 또는 테이프 백업이든 상관없이 AWS에서 제공하는 광범위한 서비스 및 파트너 도구를 사용하여 데이터 세트를 마이그레이션할 수 있습니다. 예를 들어, 많은 양의 데이터를 AWS 안팎으로 이동하거나 엣지에서 데이터를 처리하기 위해 AWS 목적별 디바이스 중 하나를 사용하여 페타바이트 단위의 데이터를 오프라인에서 비용 효율적으로 이동할 수 있습니다. 또 다른 예로, 더 빠른 데이터 전송 속도의 경우 직접 연결 서비스가 비즈니스에 필요한 일관된 연결을 제공하는 VPN보다 더 저렴할 수 있습니다.

시간의 흐름에 따라 달라지는 사용량에 대한 비용 분석을 기반으로 규모 조정 활동을 검토합니다. 결과를 분석하여 여러 인스턴스 유형 및 구매 옵션으로 인스턴스를 추가하도록 규모 조정 정책을 조정할 수 있는지 알아봅니다. 설정을 검토하여 플릿 크기가 더 작은 사용자 요청을 처리하기 위해 최소값을 줄일 수 있는지 확인하고 더 많은 리소스를 추가하여 예상되는 높은 수요를 충족합니다.

조직의 이해관계자와 논의하여 시간의 흐름에 따라 달라지는 사용량에 대한 비용 분석을 수행하고 AWS Cost Explorer의 예측 기능을 사용하여 서비스 변경이 미치는 잠재적인 영향을 예측합니다. AWS Budgets, CloudWatch 결제 경보 및 AWS Cost Anomaly Detection을(를) 사용하여 사용 수준 트리거를 모니터링해 가장 비용 효과적인 서비스를 식별하여 더 빠르게 구현합니다.

구현 단계

- 예측된 사용 패턴 정의: 마케팅 및 제품 소유자와 같은 조직과 협력하여 워크로드의 예상 및 예측 사용 패턴을 문서화합니다. 비즈니스 이해 관계자와 과거 비용 및 예측 비용과 사용량 증가에 대해 논의하여 비즈니스 요구 사항에 따라 증가되도록 합니다. 더 많은 사용자가 AWS 리소스를 사용할 것으로 예상되는 일, 주 또는 월을 식별합니다. 리소스 사용이 늘어난다는 것은 기존 리소스 용량을 증가하거나 추가 서비스를 채택하여 비용을 줄이고 성능을 개선해야 함을 나타냅니다.
- 예측 사용량에 대한 비용 분석 수행: 정의된 사용 패턴을 사용하여 이러한 각 지점에서 분석을 수행합니다. 분석 작업은 잠재적 성과를 반영해야 합니다. 예를 들어 사용량 변화가 큰 경우 비용과 변경 사항을 확인하기 위해 철저한 분석을 수행해야 합니다. 다시 말해, 비용이 높아지면 비즈니스를 위한 사용량도 따라서 증가해야 합니다.

리소스

관련 문서:

- AWS 총 소유 비용(TCO) 계산기
- Amazon S3 스토리지 클래스
- 클라우드 제품
- Amazon EC2 Auto Scaling
- 클라우드 데이터 마이그레이션
- AWS Snow Family

관련 동영상:

- AWS OpsHub for Snow Family

COST 6 리소스 유형, 크기 및 수 선택을 통해 비용 목표를 어떻게 달성합니까?

진행 중인 태스크에 대해 적절한 리소스 크기와 리소스 수를 선택해야 합니다. 가장 비용 효율적인 유형, 크기 및 수를 선택하여 리소스 낭비를 최소화할 수 있습니다.

모범 사례

- COST06-BP01 비용 모델링 수행
- COST06-BP02 데이터를 기준으로 리소스 유형, 크기, 개수 선택
- COST06-BP03 지표를 기준으로 리소스 유형, 크기, 개수 자동 선택

COST06-BP01 비용 모델링 수행

조직 요구 사항(예: 비즈니스 요구 사항 및 기존 약정)을 파악하고 워크로드 및 각 구성 요소의 비용 모델링(전반적인 비용)을 수행합니다. 그리고 예상되는 다양한 부하에서 워크로드의 벤치마크 활동을 수행하여 비용을 비교합니다. 모델링 작업은 잠재적 이점을 반영해야 합니다. 예를 들면 구성 요소의 비용과 비례하여 시간을 소비해야 합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

워크로드와 각 구성 요소에 대한 비용 모델링을 수행하여 리소스 간의 균형을 파악하고 특정 성능 수준을 고려하여 워크로드의 각 리소스에 대해 적합한 크기를 찾습니다. 비용 고려 사항을 이해하면 계획된 워크로드 배포에 대한 가치 실현 성과를 평가할 때 조직의 비즈니스 사례 및 의사 결정 프로세스를 알릴 수 있습니다.

그리고 예상되는 다양한 부하에서 워크로드의 벤치마크 활동을 수행하여 비용을 비교합니다. 모델링 작업은 잠재적 이점을 반영해야 합니다(예: 소요 시간이 구성 요소 비용 또는 예상 절감액에 비례). 모범 사례는 AWS Well-Architected 프레임워크의 성능 효율성 원칙 섹션 검토를 참조하세요.

예를 들어, 컴퓨팅 리소스로 구성된 워크로드에 대한 비용 모델링을 생성하기 위해 AWS Compute Optimizer에서 워크로드 실행을 위한 비용 모델링을 지원할 수 있습니다. 이 서비스는 사용량 기록을 기준으로 컴퓨팅 리소스에 적합한 크기 권장 사항을 제공합니다. AWS Compute Optimizer 내에서 보다 정확한 권장 사항을 제공하는 데 도움이 되는 메모리 지표를 수집하도록 CloudWatch 에이전트가 Amazon EC2 인스턴스에 배포되었는지 확인합니다. 기계 학습을 활용하여 위험 수준에 따라 여러 권장 사항을 제시하는 무료 서비스이므로 컴퓨팅 리소스에 사용하기에 적합한 데이터 소스입니다.

다른 서비스 및 워크로드 구성 요소(예: AWS Trusted Advisor, Amazon CloudWatch 및 Amazon CloudWatch Logs)에 대한 작업 크기를 올바르게 조정하기 위해 사용자 지정 로그와 함께 데이터 소스로 사용할 수 있는 서비스가 여러 개 있습니다. AWS Trusted Advisor는 리소스를 확인하여 사용률이 낮은 리소스에 플래그를 지정합니다. 그러면 리소스의 크기를 올바르게 조정하고 비용 모델링을 생성할 수 있습니다.

다음은 비용 모델링 데이터 및 지표에 대한 권장 사항입니다.

- 모니터링은 최종 사용자 환경을 정확하게 반영해야 합니다. 기간의 정확한 세부 수준을 선택하고, 평균이 아닌 최대값이나 99번째 백분위수를 적절하게 선택합니다.
- 모든 워크로드 주기를 포함하는 데 필요한 분석 기간의 정확한 세부 수준을 선택합니다. 예를 들어 분석을 2주 동안 수행하는 경우 사용률이 높은 월 단위 주기를 분석하지 못하여 리소스가 너무 적게 프로비저닝될 수 있습니다.

- 기존 약정, 다른 워크로드에 대해 선택한 요금 모델, 더 빠르게 혁신하는 기능을 고려하여 계획된 워크로드에 대해 올바른 AWS 서비스를 선택하고 핵심 비즈니스 가치에 집중합니다.

구현 단계

- 리소스에 대해 비용 모델링 수행: 테스트할 특정 리소스 유형 및 크기의 별도 계정에 워크로드 또는 개념 증명을 배포합니다. 테스트 데이터로 워크로드를 실행하고 테스트 실행 시간의 비용 데이터와 함께 출력 결과를 기록합니다. 그런 다음 워크로드를 다시 배포하거나 리소스 유형 및 크기를 변경하고 테스트를 다시 실행합니다. 비용 모델링을 생성하는 동안 이러한 리소스와 함께 사용할 수 있는 제품의 라이선스 비용과 이러한 리소스를 배포하고 관리하기 위한 예상 운영(노동력 또는 엔지니어링) 비용을 포함합니다. 기간(시간별, 일별, 월별, 연간 또는 3년)에 따른 비용 모델링을 고려합니다.

리소스

관련 문서:

- AWS Auto Scaling
- 적정 크기 조정에 대한 기회 식별
- Amazon CloudWatch 기능
- 비용 최적화: Amazon EC2 적정 크기 조정
- AWS Compute Optimizer
- AWS 요금 계산기

관련 예시:

- 데이터 기반 비용 모델링 수행
- 계획된 AWS 리소스 구성의 비용 예측
- 적절한 AWS 도구 선택

COST06-BP02 데이터를 기준으로 리소스 유형, 크기, 개수 선택

워크로드 및 리소스 특성(예: 컴퓨팅, 메모리, 처리량, 쓰기 집약형)에 대한 데이터를 기준으로 리소스 크기나 유형을 선택합니다. 일반적으로는 워크로드의 이전(온프레미스) 버전, 설명서 또는 워크로드와 관련된 기타 정보 출처를 사용해 리소스 사용량을 선택합니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

워크로드 및 리소스 특성(예: 컴퓨팅, 메모리, 처리량, 쓰기 집약형)을 기준으로 리소스 크기나 유형을 선택합니다. 일반적으로는 비용 모델링, 온프레미스 버전과 같은 워크로드의 이전 버전, 설명서 또는 워크로드와 관련된 기타 정보 출처(백서, 게시된 솔루션)를 사용하여 선택합니다.

구현 단계

- 데이터를 기준으로 리소스 선택 비용 모델링 데이터를 사용하여 예상 워크로드 사용량 수준을 선택한 다음 지정된 리소스 유형과 크기를 선택합니다.

리소스

관련 문서:

- AWS Auto Scaling
- Amazon CloudWatch 기능
- 비용 최적화: EC2 적정 크기 조정

COST06-BP03 지표를 기준으로 리소스 유형, 크기, 개수 자동 선택

현재 실행 중인 워크로드의 지표를 사용하여 비용을 최적화하기에 적합한 크기와 유형을 선택합니다. 컴퓨팅, 스토리지, 데이터 및 네트워킹 서비스의 처리량, 크기 및 스토리지를 적절하게 프로비저닝합니다. 자동 조정 등의 피드백 루프나 워크로드의 사용자 지정 코드로 이 프로비저닝을 수행할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

실행 중인 워크로드에서 나오는 활성 지표를 사용하여 해당 워크로드를 변경하는 피드백 루프를 워크로드 내에 생성합니다. AWS Auto Scaling과 같은 관리형 서비스를 사용하여 규모 조정 작업을 대신 수행하도록 서비스를 구성할 수 있습니다. 또한 AWS는 최소한의 노력으로 리소스를 수정할 수 있는 API, SDK 및 기능을 제공합니다. Amazon EC2 인스턴스를 중지하고 시작하도록 워크로드를 프로그래밍하면 인스턴스 크기 또는 인스턴스 유형을 변경할 수 있습니다. 이렇게 하면 변경에 필요한 거의 모든 운영 비용을 절감하면서 규모 조정의 이점을 실현할 수 있습니다.

일부 AWS 서비스에는 Amazon Simple Storage Service Intelligent-Tiering과 같은 자동 유형 또는 크기 선택 기능이 기본적으로 포함되어 있습니다. Amazon S3 Intelligent-Tiering은 사용 패턴에 따라 자주 접근하고 자주 접근하지 않는 두 개의 접근 계층 간에 자동으로 데이터를 이동합니다.

구현 단계

- 워크로드 지표를 구성하여 관측성 높이기: 워크로드의 핵심 지표를 캡처합니다. 이러한 지표는 워크로드 출력과 같은 고객 경험을 나타내며 CPU 및 메모리 사용량과 같은 리소스 유형 및 크기 간의 차이에 부합합니다. 컴퓨팅 리소스의 경우 성능 데이터를 분석하여 Amazon EC2 인스턴스 크기를 적절하게 조정합니다. 유휴 인스턴스와 사용률이 낮은 인스턴스를 식별합니다. 확인해야 할 핵심 지표는 CPU 사용량 및 메모리 사용률(예를 들어, AWS Compute Optimizer 및 메모리 사용률을 활성화하여 적절한 크기 지정에서 설명한 대로 90%의 시간에 40%의 CPU 사용률)입니다. 4주 동안 최대 CPU 사용량과 메모리 사용률이 40% 미만인 인스턴스를 식별합니다. 이것이 비용을 절감하기 위해 적정하게 크기를 조정하는 인스턴스입니다. Amazon S3와 같은 스토리지 리소스의 경우 Amazon S3 스토리지 렌즈를 사용할 수 있으며, 이를 통해 기본적으로 버킷 수준에서 다양한 범주의 28개 지표와, 대시보드에서 14일의 기록 데이터를 확인할 수 있습니다. 요약 및 비용 최적화 또는 이벤트를 기준으로 Amazon S3 스토리지 렌즈 대시보드를 필터링하여 특정 지표를 분석할 수 있습니다.
- 적정 크기 조정 권장 사항 보기: 비용 관리 콘솔에서 AWS Compute Optimizer 및 Amazon EC2 적정 크기 조정 도구의 적정 크기 조정 권장 사항을 사용하거나, 리소스의 AWS Trusted Advisor 적정 크기 조정을 검토하여 워크로드를 조정합니다. 서로 다른 리소스의 크기를 적절하게 조정할 때는 올바른 도구를 사용하고 Amazon EC2 인스턴스, AWS 스토리지 클래스 또는 Amazon RDS 인스턴스 유형에 관계없이 올바른 크기 조정 지침을 따르는 것이 중요합니다. 스토리지 리소스의 경우 객체 스토리지 사용량, 활동 추세에 대한 가시성을 제공하고, 비용 최적화를 위한 실행 가능한 권장 사항을 제시하며 데이터 보호 모범 사례를 적용할 수 있는 Amazon S3 스토리지 렌즈를 사용할 수 있습니다. Amazon S3 스토리지 렌즈가 조직 전체의 지표 분석에서 도출한 상황별 권장 사항을 사용하여 스토리지를 최적화하는 즉각적인 단계를 수행할 수 있습니다.
- 지표를 기준으로 리소스 유형 및 크기 자동 선택: 워크로드 지표를 사용하여 워크로드 리소스를 수동 또는 자동으로 선택합니다. 컴퓨팅 리소스의 경우, AWS Auto Scaling을 구성하거나 애플리케이션 내에서 코드를 구현하면 자주 변경해야 하는 경우 필요한 작업량을 줄이고 수동 프로세스보다 빨리 변경 사항을 구현할 수 있습니다. 단일 Auto Scaling 그룹 내에서 온디맨드 인스턴스 및 스팟 인스턴스 플릿을 시작하고 자동으로 확장할 수 있습니다. 스팟 인스턴스 사용에 대한 할인을 받는 것은 물론, 예약 인스턴스 또는 절감형 플랜을 사용하여 정규 온디맨드 인스턴스 요금을 할인 받을 수 있습니다. 이 모든 요소를 결합하여 Amazon EC2 인스턴스의 비용 절감을 최적화하고 애플리케이션에 대해 원하는 규모와 성능을 결정할 수 있습니다. 또한 Auto Scaling Groups(ASG)의 속성 기반 인스턴스 유형 선택(ABS) 전략을 사용하여 vCPU, 메모리 및 스토리지와 같은 일련의 속성으로서의 인스턴스 요구 사항을 표현할 수 있습니다. 신세대 인스턴스 유형이 릴리스되면 이를 자동으로 사용하고 Amazon EC2 스팟 인스턴스를 통해 더 광범위한 용량에 액세스할 수 있습니다. Amazon EC2 플릿과 Amazon EC2 Auto Scaling는 지정된 속성에 맞는 인스턴스를 선택하고 시작하여 인스턴스 유형을 수동으로 선택할 필요성이 사라집니다. 스토리지 리소스의 경우, 데이터 액세스 패턴이 변경되면 성능 영향이나 운영 오버헤드 없이 자동으로 스토리지 비용이 절감되는 스토리지 클래스를 자동으

로 선택할 수 있는 Amazon S3 Intelligent Tiering 및 Amazon EFS Infrequent Access 기능을 사용할 수 있습니다.

리소스

관련 문서:

- AWS Auto Scaling
- AWS 적정 크기 조정
- AWS Compute Optimizer
- Amazon CloudWatch 기능
- CloudWatch 설정
- CloudWatch 사용자 지정 지표 게시
- Amazon EC2 Auto Scaling 시작하기
- Amazon S3 스토리지 렌즈
- Amazon S3 Intelligent-Tiering
- Amazon EFS Infrequent Access
- SDK를 사용하여 Amazon EC2 인스턴스 시작

관련 동영상:

- 서비스의 적정 크기 조정

관련 예시:

- Amazon EC2 플릿을 위한 Auto Scaling에 대한 속성 기반 인스턴스 유형 선택
- 예약된 조정을 사용하여 비용을 위한 Amazon Elastic Container Service 최적화
- Amazon EC2 Auto Scaling으로 예측 스케일링
- Amazon S3 스토리지 렌즈를 사용한 비용 최적화 및 사용량에 대한 가시성 확보
- Well-Architected 실습: 적정 크기 조정 권장 사항(레벨 100)
- Well-Architected 실습: AWS Compute Optimizer 및 메모리 사용률을 활성화하여 적정 크기 조정(레벨 200)

COST 7 비용 절감을 위해 가격 책정 모델을 어떻게 사용합니까?

해당 리소스에 대해 비용을 최소화하는 데 가장 적합한 요금 모델을 사용합니다.

모범 사례

- COST07-BP01 요금 모델 분석 수행
- COST07-BP02 비용을 기준으로 리전 구현
- COST07-BP03 비용 효율적인 조건을 갖춘 서드 파티 계약 선택
- COST07-BP04 워크로드의 모든 구성 요소용 요금 모델 구현
- COST07-BP05 관리 계정 수준에서 요금 모델 분석 수행

COST07-BP01 요금 모델 분석 수행

워크로드의 각 구성 요소를 분석합니다. 구성 요소와 리소스가 장기간 실행되는지(약정 할인의 경우) 아니면 동적으로 단기간 실행되는지(스팟 또는 온디맨드의 경우) 결정합니다. 비용 관리 도구의 권장 사항을 바탕으로 워크로드를 분석하고 높은 수익을 달성할 수 있도록 해당 권장 사항에 비즈니스 규칙을 적용합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

AWS는 조직의 요구 사항에 적합하며 제품에 맞게 가장 비용 효율적인 방법으로 리소스 요금을 지불할 수 있는 다수의 요금 모델을 제공합니다. 팀과 협력하여 가장 적합한 요금 모델을 결정하세요. 요금 모델은 가용성에 따라 결정되는 여러 옵션의 조합으로 구성되는 경우가 많습니다.

온디맨드 인스턴스를 사용하면 실행 중인 인스턴스에 따라 시간별 또는 초 단위(최소 60초)로 컴퓨팅 또는 데이터베이스 용량에 대해 사용한 비용을 지불할 수 있습니다. 장기 약정이나 선불 비용 결제는 필요하지 않습니다.

Savings Plans은 1년 또는 3년 기간 동안 일정한 사용량(시간당 달러로 측정)을 약정하는 대가로, Amazon EC2, Lambda 및 AWS Fargate를 낮은 가격으로 사용할 수 있도록 지원하는 유연한 요금 모델입니다.

스팟 인스턴스는 사전 약정 없이 시간당 온디맨드 가격의 최대 90% 할인된 가격으로 예비 컴퓨팅 용량을 요청할 수 있는 Amazon EC2 요금 메커니즘입니다.

예약형 인스턴스는 최대 75%까지 할인된 가격을 선불로 결제하여 용량을 예약할 수 있도록 지원합니다. 자세한 내용은 예약을 통한 비용 최적화를 참조하세요.

프로덕션 , 품질 및 개발 환경과 관련된 리소스에 대해 Savings Plan을 포함하도록 선택할 수 있습니다. 아니면 샌드박스 리소스의 전원은 필요한 경우에만 켜지므로, 해당 환경의 리소스에 대해 온디맨드 모델을 선택할 수도 있습니다. Amazon 스팟 인스턴스를 사용하여 Amazon EC2 비용을 절감하거나 컴퓨팅 Savings Plans을 사용하여 Amazon EC2, Fargate, Lambda 비용을 절약하세요. AWS Cost Explorer 권장 사항 도구는 절감형 플랜과 함께 약정 할인을 받을 수 있는 기회를 제공합니다.

이전에 Amazon EC2용 예약형 인스턴스를 구매했거나 조직 내에서 비용 할당 관행을 마련한 경우에는 당분간 Amazon EC2 예약형 인스턴스를 계속 사용할 수 있습니다. 그러나 향후 보다 유연한 비용 절감 메커니즘으로 Savings Plans을 사용할 전략을 수립할 것을 권장합니다. AWS Cost Management에서 Savings Plans(SP) 권장 사항을 새로 고쳐 언제든지 최신 절감형 플랜 권장 사항을 생성할 수 있습니다. 예약형 인스턴스(RI)를 사용하여 Amazon RDS, Amazon Redshift, Amazon ElastiCache 및 Amazon OpenSearch Service 비용을 절감할 수 있습니다. 절감형 플랜 및 예약형 인스턴스는 전액 선결제, 부분 선결제, 선결제 없음이라는 3가지 옵션으로 제공됩니다. AWS Cost Explorer RI 및 SP 구매 권장 사항에 나와 있는 권장 사항을 참조하세요.

스팟 워크로드 기회를 찾으려면 전체 사용량을 시간대별로 확인하여 사용량 또는 탄력성이 주기적으로 변경되는 기간을 찾아보세요. 스팟 인스턴스를 내결함성 및 유연성을 갖춘 다양한 애플리케이션에 사용할 수 있습니다. 예를 들어, 상태 비저장 웹 서버, API 엔드포인트, 빅 데이터 및 분석 애플리케이션, 컨테이너형 워크로드, CI/CD 및 기타 유연한 워크로드 등이 있습니다.

사용하지 않을 때(업무 시간 이후 및 주말) Amazon EC2 및 Amazon RDS 인스턴스를 해제할 수 있는지를 분석하세요. 이 전략을 활용하면 연중무휴로 사용할 때보다 비용을 70% 이상 절감할 수 있습니다. 특정 시간에만 사용해야 하는 Amazon Redshift 클러스터가 있는 경우 클러스터를 일시 중지했다가 나중에 다시 시작하면 됩니다. Amazon Redshift 클러스터 또는 Amazon EC2 및 Amazon RDS 인스턴스가 중지되면 컴퓨팅 과금 청구가 정지되고 스토리지 요금만 부과됩니다.

온디맨드 용량 예약(ODCR)은 요금 할인이 아닙니다. 용량 예약은 예약된 용량으로 인스턴스를 실행하는지와 관계없이 동일한 온디맨드 요율로 청구됩니다. 실행하려는 리소스에 용량을 충분히 제공해야 하는 경우 이를 고려해야 합니다. ODCR은 더 이상 필요하지 않을 때 취소할 수 있기 때문에 장기 약정에 얽매이지 않으면서 Savings Plans 또는 예약형 인스턴스가 제공하는 할인 혜택도 누릴 수 있습니다.

구현 단계

- 워크로드 탄력성 분석: Cost Explorer 또는 사용자 지정 대시보드에서 시간 단위의 세부 수준을 사용하여 워크로드 탄력성을 분석합니다. 실행 중인 인스턴스 개수의 정기적인 변경 사항을 확인합니다. 기간이 짧은 인스턴스가 스팟 인스턴스 또는 스팟 플릿에 적합합니다.
  - Well-Architected 실습: Cost Explorer
  - Well-Architected 실습: 비용 시각화

- 기존 요금 계약 검토: 장기적인 요구 사항에 대해 현재 계약 또는 약정을 검토합니다. 현재 사용 중인 약정과 이러한 약정을 얼마나 사용하고 있는지 분석합니다. 기존의 계약상 할인 또는 엔터프라이즈 계약을 활용합니다. 엔터프라이즈 계약은 고객의 요구에 가장 적합한 계약을 맞춤 설정할 수 있는 옵션을 제공합니다. 장기 약정의 경우 특정 인스턴스 유형, 인스턴스 제품군, AWS 리전, 가용 영역에 대해 예약 요금 할인, 예약형 인스턴스 또는 Savings Plans을 고려하세요.
- 약정 할인 분석 수행: 계정에서 Cost Explorer를 사용하여 Savings Plans 및 예약형 인스턴스 권장 사항을 검토합니다. 필요한 할인 및 위험과 함께 올바른 권장 사항을 구현하려면 Well-Architected 실습을 따르세요.

리소스

관련 문서:

- 예약형 인스턴스 권장 사항에 접근
- 인스턴스 구매 옵션
- AWS 엔터프라이즈

관련 동영상:

- Save up to 90% and run production workloads on Spot(최대 90% 절감 및 스팟에서 프로덕션 워크로드 실행)

관련 예시:

- Well-Architected 실습: Cost Explorer
- Well-Architected 실습: 비용 시각화
- Well-Architected 실습: 요금 모델

COST07-BP02 비용을 기준으로 리전 구현

리소스 요금은 리전별로 다를 수 있습니다. 리전별 비용 차이를 식별하고 지연 시간, 데이터 상주 및 데이터 주권 요구 사항을 충족하기 위해 필요한 경우에만 비용이 더 높은 지역에서 배포합니다. 따라서 리전 비용을 고려하면 이 워크로드의 전체 가격을 최저 수준으로 낮출 수 있습니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 보통

구현 가이드

글로벌한 AWS 클라우드 인프라는 전 세계 여러 위치에서 호스트되며AWS 리전, 가용 영역, 로컬 영역, AWS Outposts, Wavelength 영역을 중심으로 구축되었습니다. 리전은 전 세계에 있는 실제 위치이고 각 리전은 AWS에 가용 영역이 여러 개 있는 개별 지리적 영역입니다. 각 리전 내의 여러 분리된 위치인 가용 영역은 이중화된 전력, 네트워킹 및 연결 기능을 갖추고 있는 하나 이상의 개별 데이터 센터로 구성됩니다.

각 AWS 리전은 현지 시장 조건 내에서 운영되며 예를 들어, 리소스 요금은 토지, 광섬유, 전기 및 세금 비용의 차이 때문에 리전마다 다릅니다. 전 세계에서 사용 가능한 최저 가격으로 리소스를 실행할 수 있도록 솔루션의 한 구성 요소나 전체 솔루션을 운영할 특정 리전을 선택합니다. 또한 AWS 계산기를 사용하여 위치 유형(리전, Wavelength 영역 및 로컬 영역) 및 리전별로 서비스를 검색해 다양한 리전에서 워크로드 비용을 예측합니다.

솔루션 설계 시의 모범 사례는 사용자와 더 가까운 위치에 컴퓨팅 리소스를 배치하여 지연 시간을 줄이고 데이터 주권을 강화하는 것입니다. 비즈니스, 개인정보 처리방침, 성능, 보안 요구 사항에 따라 지리적 위치를 선택합니다. 전 세계에 최종 사용자가 분포하는 애플리케이션의 경우 여러 위치를 사용합니다.

개인정보 처리방침, 보안 및 비즈니스 요구 사항에 대한 의무가 없는 경우 AWS 서비스에 대해 더 저렴한 요금을 제공하는 리전을 사용하여 워크로드를 배포합니다. 예를 들어, 기본 리전이 ap-southeast-2(시드니)이고 다른 리전 사용에 대한 제한 사항(예: 개인정보 처리방침, 보안)이 없는 경우, 개발이나 테스트처럼 중요하지 않은 Amazon EC2 인스턴스는 north-east-1(버지니아 북부) 리전에 배포하면 비용을 줄일 수 있습니다.

	규정 준수	지연 시간	비용	서비스/기능
리전 1	✓	15ms	$$	✓
리전 2	✓	20ms	$$$	X
리전 3	✓	80ms	$	✓
리전 4	✓	15ms	$$	✓
리전 5	✓	20ms	$$$	X
리전 6	✓	15ms	$	✓
리전 7	✓	80ms	$	✓
리전 8	✓	15ms	$	X

리전별 기능 표

위의 표는 리전 4가 다른 리전에 비해 지연 시간이 짧고 서비스를 이용할 수 있으며 비용이 가장 저렴한 리전이기 때문에 이 시나리오에 가장 적합한 옵션임을 보여줍니다.

구현 단계

- AWS 리전 가격 검토: 현재 리전의 워크로드 비용을 분석합니다. 서비스 및 사용 유형별로 가장 높은 비용부터 시작하여 사용 가능한 다른 리전의 비용을 계산합니다. 예상 절감액이 구성 요소 또는 워크로드 이동 비용보다 큰 경우 새 리전으로 마이그레이션합니다.
- 다중 리전 배포를 위한 요구 사항 검토: 비즈니스 요구 사항 및 의무(개인정보 처리방침, 보안 또는 성능)를 분석하여 여러 리전을 사용하면 안 되는 제한 사항이 있는지 확인합니다. 단일 리전을 사용하도록 제한하는 의무가 없는 경우 다중 리전을 사용하십시오.
- 필요한 데이터 전송 분석: 리전을 선택할 때 데이터 전송 비용을 고려합니다. 데이터는 고객 및 리소스 가까운 곳에 두십시오. 데이터가 흐르고 데이터 전송이 최소화된 더 저렴한 AWS 리전을 선택합니다. 데이터 전송에 대한 비즈니스 요구 사항에 따라 Amazon CloudFront, AWS PrivateLink, AWS Direct Connect 및 AWS Virtual Private Network을 사용하여 네트워킹 비용을 줄이고 성능을 개선하며 보안을 강화할 수 있습니다.

리소스

관련 문서:

- 예약 인스턴스 권장 사항에 접근
- Amazon EC2 요금
- 인스턴스 구매 옵션
- 리전 표

관련 동영상:

- Save up to 90% and run production workloads on Spot(최대 90% 절감 및 스팟에서 프로덕션 워크로드 실행)

관련 예시:

- 공통 아키텍처에 대한 데이터 전송 비용 개요
- 글로벌 배포를 위한 비용 고려 사항
- 보안 및 규정 준수 등의 제약을 고려해야 합니다
- Well-Architected 실습: 리전별로 서비스 사용량 제한(레벨 200)

COST07-BP03 비용 효율적인 조건을 갖춘 서드 파티 계약 선택

비용 효율적인 계약과 조건은 이러한 서비스의 비용이 제공하는 혜택에 따라 늘어나도록 보장합니다. 조직에 추가적인 혜택을 제공할 때 조정되는 계약 및 요금을 선택하십시오.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

클라우드에서 서드파티 솔루션 또는 서비스를 활용하는 경우 요금 구조는 비용 최적화 결과에 부합해야 합니다. 요금은 제공하는 결과와 가치에 따라 조정되어야 합니다. 예를 들어 제공하는 절감액의 일정 비율을 가져와서 절감액(결과)이 커질수록 더 많은 요금을 부과하는 소프트웨어가 있습니다. 청구 금액에 따라 규모가 조정되는 계약은 특정 청구 금액의 일정 부분에 대해 결과를 제공하지 않는 한 일반적으로 비용 최적화에 맞지 않습니다. 예를 들어 Amazon Elastic Compute Cloud(Amazon EC2)에 대한 권장 사항을 제공하고 전체 청구 금액의 일정 비율을 부과하는 솔루션을 사용하는 경우 이 권장

사항이 어떤 이득도 되지 않는 다른 서비스를 함께 사용하면 솔루션 요금이 증가하게 됩니다. 또 다른 예로는 관리되는 리소스 비용의 일정 비율로 요금을 부과하는 관리형 서비스가 있습니다. 인스턴스 크기가 크다고 해서 반드시 더 많은 관리 노력이 필요한 것은 아니지만 부과되는 요금은 증가합니다. 따라서 이러한 서비스 요금 방식에 서비스의 효율성을 증진하는 비용 최적화 프로그램 또는 기능이 포함되어 있는지 확인해야 합니다.

구현 단계

- 서드 파티 계약 및 조건 분석: 타사 계약의 요금을 검토합니다. 다양한 사용량에 대해 모델링을 수행하고, 새로운 서비스 사용량과 같은 새로운 비용 또는 워크로드 증가로 인한 현재 서비스의 증가를 고려합니다. 추가 비용이 비즈니스에 필요한 이점을 제공하는지 여부를 파악합니다.

리소스

관련 문서:

- 예약형 인스턴스 권장 사항에 접근
- 인스턴스 구매 옵션

관련 동영상:

- 최대 90% 절감 및 스팟에서 프로덕션 워크로드 실행

COST07-BP04 워크로드의 모든 구성 요소용 요금 모델 구현

영구 실행되는 리소스는 Savings Plans 또는 예약형 인스턴스와 같은 예약 용량을 활용해야 합니다. 단기간 용량은 스팟 인스턴스나 스팟 플릿을 사용하도록 구성됩니다. 온디맨드 인스턴스는 예약 용량을 사용할 만큼 충분히 길게 실행되지 않으며(리소스 유형에 따라 1년 중 대개 25%~75%에 해당하는 기간에 실행됨) 중단할 수 없는 단기 워크로드에만 사용됩니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 낮음

구현 가이드

워크로드 구성 요소의 요구 사항을 고려하고 가능한 요금 모델을 파악합니다. 구성 요소의 가용성 요구 사항을 정의합니다. 워크로드의 기능이 다수의 독립된 리소스를 통해 수행되는지 여부와 워크로드의 시간대별 요구 사항을 파악합니다. 기본 온디맨드 요금 모델 및 기타 적용 가능한 모델을 사용하여 리소스 비용을 비교합니다. 리소스 또는 워크로드 구성 요소의 잠재적 변경을 고려합니다.

구현 단계

- 요금 모델 구현: 분석 결과를 사용하여 Savings Plans(SP), 예약형 인스턴스(RI)를 구입하거나 스팟 인스턴스를 구현합니다. 첫 번째 RI 구매인 경우 목록에서 상위 5개 또는 10개 권장 사항을 선택한 후 다음 달 또는 2개월 동안 결과를 모니터링하고 분석합니다. 예를 들어 2주마다 또는 매월 약정 할인을 정기적으로 소량 구매합니다. 중단될 수 있거나 상태 정보를 저장하지 않는 워크로드에 대해 스팟 인스턴스를 구현합니다.
- 워크로드 검토 주기: 요금 모델 적용 범위를 구체적으로 분석하는 워크로드에 대한 검토 주기를 구현합니다. 워크로드가 필요한 적용 범위에 도달하면 2~4주마다 또는 조직 사용량이 바뀔 때 추가 약정 할인을 구매하십시오.

리소스

관련 문서:

- 예약형 인스턴스 권장 사항에 접근
- EC2 집합
- 예약 인스턴스 구매 방법
- 인스턴스 구매 옵션
- 스팟 인스턴스

관련 동영상:

- 최대 90% 절감 및 스팟에서 프로덕션 워크로드 실행

COST07-BP05 관리 계정 수준에서 요금 모델 분석 수행

과금 정보 및 비용 관리 도구를 확인하고 관리 계정 수준에서 정기적인 분석을 수행하기 위해 권장되는 할인 혜택과 약정, 예약을 확인하십시오.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 낮음

구현 가이드

정기적인 비용 모델링을 수행하면 여러 워크로드에 걸친 최적화 기회를 구현할 수 있습니다. 예를 들어, 여러 워크로드에 온디맨드 인스턴스를 사용하는 경우 집계 수준에서 변경 위험이 낮으므로 약정 기반 할인을 구현하여 전반적인 비용을 절감할 수 있습니다. 2주에서 1개월의 정기적인 주기로 분석을 수

행하는 것이 좋습니다. 이렇게 하면 구매를 조금씩 조정할 수 있으므로 워크로드 및 워크로드 구성 요소의 변경에 따라 요금 모델의 적용 범위를 점진적으로 변경할 수 있습니다.

여러분은 AWS Cost Explorer 권장 사항 도구를 사용하여 관리 계정에서 약정 할인의 기회를 찾을 수 있습니다. 관리 계정 수준의 권장 사항은 계정 간 절감액을 극대화하는 약정을 추천할 수 있도록 예약 인스턴스 또는 절감형 플랜(SP) 할인 공유를 활성화한 AWS 조직 내 모든 계정의 사용량을 고려하여 측정됩니다.

관리 계정 수준에서 구매하면 대부분의 경우 최대한 비용을 절감할 수 있지만 특정 연결 계정의 사용량에 할인을 먼저 적용하려는 경우 등 연결 계정 수준에서 SP를 구매하는 것을 고려해야 하는 상황도 있을 수 있습니다. 멤버 계정 권장 사항은 개별 계정 수준에서 측정되어 분리된 각 계정에 대한 절감 효과를 극대화합니다. 계정에 RI 약정과 SP 약정이 둘 다 있는 경우 다음 순서로 적용됩니다.

영역 RI > 표준 RI > 컨버터블 RI > 인스턴스 절감형 플랜 > 컴퓨팅 절감형 플랜

관리 계정 수준에서 SP를 구매하면 절감액이 가장 높은 할인율에서 가장 낮은 할인율 순으로 적용됩니다. 관리 계정 수준의 SP는 연결된 모든 계정을 살펴보고 할인율이 가장 높은 곳에 절감액을 적용합니다. 할인 적용 대상을 제한하려는 경우 연결 계정 수준에서 절감형 플랜을 구매하면 해당 계정에서 적격 컴퓨팅 서비스를 실행할 때마다 할인이 먼저 적용됩니다. 해당 계정에서 적합한 컴퓨팅 서비스를 실행하지 않는 경우 할인은 동일한 관리 계정 내의 다른 연결된 계정과 공유됩니다. 할인 공유는 기본적으로 활성화되어 있지만 필요한 경우 비활성화할 수 있습니다.

통합 결제 패밀리에서는 절감형 플랜이 먼저 소유자 계정의 사용량에 적용된 다음 다른 계정의 사용량에 적용됩니다. 이는 공유를 활성화한 경우에만 해당됩니다. 절감형 플랜은 가장 높은 절감율에 먼저 적용됩니다. 절감률이 동일한 사용량이 여러 개 있는 경우 절감형 플랜 요금이 가장 낮은 첫 번째 사용량에 절감형 플랜이 적용됩니다. 절감형 플랜은 남은 사용량이 더 이상 없거나 약정이 소진될 때까지 계속 적용됩니다. 나머지 사용량은 온디맨드 요금으로 청구됩니다. AWS Cost Management에서 절감형 플랜 권장 사항을 새로 고쳐 언제든지 최신 절감형 플랜 권장 사항을 생성할 수 있습니다.

인스턴스의 유연성을 분석한 후 권장 사항에 따라 약정할 수 있습니다. 다양한 잠재 리소스 옵션을 사용하여 워크로드의 단기 비용을 분석하고 AWS 요금 모델을 분석하며 이를 비즈니스 요구 사항에 맞춰 조정하여 총 소유 비용 및 비용 최적화 기회를 파악할 수 있는 비용 모델링을 생성합니다.

구현 단계

- 약정 할인 분석 수행: 계정에서 Cost Explorer를 사용하여 Savings Plans 및 예약 인스턴스 권장 사항을 검토합니다. 절감형 플랜 권장 사항을 이해하고, 월간 지출액을 추정하고, 월간 절감액을 추산해야 합니다. 계정 간 절감액을 극대화하기 위해 예약 인스턴스 또는 Savings Plans 할인 공유를 활성화한 AWS 조직 내 모든 멤버 계정의 사용량을 고려하여 측정되는 관리 계정 수준의 권장 사항을 검

토합니다. 필요한 할인 및 위험과 함께 올바른 권장 사항을 구현하려면 Well-Architected 실습을 따르면 됩니다.

리소스

관련 문서:

- AWS 요금은 어떻게 적용됩니까?
- 인스턴스 구매 옵션
- 절감형 플랜 개요
- 절감형 플랜 권장 사항
- 예약 인스턴스 권장 사항에 접근
- AWS 사용량에 Savings Plans을 적용하는 방법
- 통합 결제를 통한 Savings Plans
- 예약 인스턴스 및 Savings Plans 할인 공유 활성화

관련 동영상:

- Save up to 90% and run production workloads on Spot(최대 90% 절감 및 스팟에서 프로덕션 워크로드 실행)

관련 예시:

- Well-Architected 실습: 요금 모델(레벨 200)
- Well-Architected 실습: 요금 모델 분석(레벨 200)
- Savings Plan을 구매하기 전에 고려해야 할 사항은 무엇입니까?
- 단계적 Savings Plans을 사용하여 약정 위험을 줄이는 방법은?
- When to Use Spot Instances(스팟 인스턴스 사용 시기)

COST 8 데이터 전송 요금을 위한 계획은 어떻게 합니까?

비용 최소화를 위한 아키텍처 관련 사항을 결정할 수 있도록 데이터 전송 요금을 계획하고 모니터링해야 합니다. 아키텍처를 약간이라도 효율적으로 변경하면 장기적으로 운영 비용을 크게 줄일 수 있습니다.

모범 사례

- COST08-BP01 데이터 전송 모델링 수행
- COST08-BP02 데이터 전송 비용을 최적화할 구성 요소 선택
- COST08-BP03 데이터 전송 비용을 줄이기 위한 서비스 구현

COST08-BP01 데이터 전송 모델링 수행

조직 요구 사항을 수집하고 워크로드 및 각 워크로드 구성 요소의 데이터 전송 모델링을 수행합니다. 그러면 현재 데이터 전송 요구 사항을 충족할 수 있는 최저 비용을 파악할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

워크로드에서 데이터 전송이 발생하는 위치, 전송 비용 및 관련된 이점을 파악하십시오. 그러면 정보를 바탕으로 결정을 내리고 아키텍처 의사 결정을 수정하거나 수락할 수 있습니다. 예를 들어 가용 영역 간에 데이터를 복제하는 다중 가용 영역 구성이 있는 경우 구조의 비용을 모델링하고 필요한 안정성 및 복원력을 달성하기에 허용 가능한 비용(두 가용 영역의 컴퓨팅 및 스토리지 비용을 지불하는 것과 유사함)인지 결정합니다.

다양한 사용 수준에 걸쳐 비용을 모델링합니다. 워크로드 사용량은 시간이 지남에 따라 변경될 수 있으며 여러 수준에서 다양한 서비스를 사용하는 것이 더 비용 효율적일 수 있습니다.

사용 AWS Cost Explorer 또는 AWS Cost and Usage Report (CUR)을 사용하여 데이터 전송 비용을 파악하고 모델링하세요. PoC(개념 증명)를 구성하거나 워크로드를 테스트하고 사실적으로 시뮬레이션된 로드로 테스트를 실행합니다. 다양한 워크로드 수요에서 비용을 모델링할 수 있습니다.

구현 단계

- 데이터 전송 비용 계산: SAP 환경의 보안 관련 작업에 대한 지침은 AWS 요금 페이지 를 사용하고 워크로드에 대한 데이터 전송 비용을 계산합니다. 워크로드 사용량의 증가 및 감소 모두에 대해 여러 사용 수준의 데이터 전송 비용을 계산합니다. 워크로드 아키텍처에 대한 여러 옵션이 있는 경우 비교를 위해 각 옵션의 비용을 계산합니다.
- 성과에 비용 연결: 발생한 각 데이터 전송 비용에 대해 워크로드에 대해 달성하는 성과를 지정합니다. 구성 요소 간 전송인 경우 분리를 위한 것일 수 있으며, 가용 영역 간 전송인 경우 이중화를 위한 것일 수 있습니다.

리소스

관련 문서:

- AWS 캐싱 솔루션
- AWS 요금
- Amazon EC2 요금
- Amazon VPC 요금
- Amazon CloudFront로 더 빠르게 콘텐츠 전송

COST08-BP02 데이터 전송 비용을 최적화할 구성 요소 선택

모든 구성 요소를 선택해야 하며, 데이터 전송 비용을 줄이도록 아키텍처를 설계해야 합니다. 이 과정에서는 광역 네트워크(WAN) 최적화, 다중 가용 영역(AZ) 구성 등의 구성 요소를 사용할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

데이터 전송 최적화를 고려하여 설계를 수행하면 데이터 전송 비용을 최소화할 수 있습니다. 이 설계 과정에서는 콘텐츠 전송 네트워크를 사용해 사용자와 더 가까운 위치에 데이터를 배치하거나, 온프레미스에서 AWS로의 전용 네트워크 링크를 사용합니다. 또한 WAN 최적화 및 애플리케이션 최적화를 사용하여 구성 요소 간에 전송되는 데이터의 양을 줄일 수도 있습니다.

구현 단계

- 데이터 전송을 위한 구성 요소 선택: 데이터 전송 모델링을 사용하여 데이터 전송 비용이 가장 큰 영역이나 워크로드 사용량이 변경되는 경우 데이터 전송 비용이 발생하는 영역을 집중적으로 살펴봅니다. 데이터 전송 필요를 없애거나 비용을 낮추는 대체 아키텍처나 추가 구성 요소를 찾습니다.

리소스

관련 문서:

- AWS 캐싱 솔루션
- Amazon CloudFront로 더 빠르게 콘텐츠 전송

COST08-BP03 데이터 전송 비용을 줄이기 위한 서비스 구현

데이터 전송을 줄이기 위한 서비스를 구현합니다. 예를 들어 Amazon CloudFront 등의 콘텐츠 전송 네트워크(CDN)를 사용해 최종 사용자에게 콘텐츠를 전송하거나, Amazon ElastiCache를 사용하여 계층을 캐시하거나, VPN 대신 AWS Direct Connect를 사용해 AWS에 연결할 수 있습니다.

이 모범 사례가 수립되지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

Amazon CloudFront 는 지연 시간이 짧고 전송 속도가 빠른 데이터 전송용 글로벌 콘텐츠 전송 네트워크로, 전 세계의 엣지 로케이션에서 데이터를 캐시하므로 리소스에 대한 로드가 감소합니다. CloudFront를 사용하면 전 세계의 많은 사용자에게 콘텐츠를 전송하는 과정의 관리 작업을 줄이고 지연 시간을 최소화할 수 있습니다.

AWS Direct Connect 를 사용하면 AWS에 대한 전용 네트워크 연결을 설정할 수 있습니다. 이렇게 하면 네트워크 비용을 줄이고 대역폭을 늘릴 수 있으며 인터넷 기반 연결에 비해 더 일관성이 높은 네트워크 환경을 제공할 수 있습니다.

AWS VPN 을 사용하면 프라이빗 네트워크와 AWS 글로벌 네트워크 간에 안전한 프라이빗 연결을 설정할 수 있습니다. 탄력적인 완전관리형 서비스로서 빠르고 간편한 연결을 제공하므로 소규모 사무실 또는 비즈니스 파트너에 적합합니다.

VPC 엔드포인트 를 사용하면 프라이빗 네트워킹을 통해 여러 AWS 서비스를 연결할 수 있으며, 퍼블릭 데이터 전송 및 NAT 게이트웨이 비용을 줄일 수 있습니다. 게이트웨이 VPC 엔드포인트 는 시간당 비용이 없으며 Amazon Simple Storage Service(Amazon S3) 및 Amazon DynamoDB를 지원합니다. 인터페이스 VPC 엔드포인트는 AWS PrivateLink 를 통해 제공되며 시간당 요금과 GB당 사용 비용이 있습니다.

구현 단계

- 서비스 구현: 데이터 전송 모델링을 사용하여 비용이 가장 크고 볼륨 흐름이 가장 높은 영역을 확인합니다. AWS 서비스를 검토하고 전송을 줄이거나 제거하는 서비스(특히 네트워킹 및 콘텐츠 전송)가 있는지 평가합니다. 또한 데이터 또는 대량의 데이터에 대한 액세스가 반복되는 캐싱 서비스를 찾습니다.

리소스

관련 문서:

- AWS Direct Connect
- AWS 제품 둘러보기
- AWS 캐싱 솔루션
- Amazon CloudFront
- Amazon CloudFront를 사용하여 콘텐츠를 더 빠르게 전송

수요 관리 및 리소스 공급

질문

- COST 9 수요와 리소스 공급은 어떻게 관리합니까?

COST 9 수요와 리소스 공급은 어떻게 관리합니까?

비용과 성능을 적절하게 절충한 워크로드에서는 비용을 결제한 모든 리소스가 사용되는지 확인하고, 사용률이 매우 낮은 인스턴스가 없도록 해야 합니다. 사용률 지표가 매우 높거나 낮으면 조직의 운영 비용(사용률이 너무 높아 성능이 저하됨)이 늘어나거나 과도한 프로비저닝으로 AWS 지출 금액이 낭비되는 등 조직에 악영향을 미칩니다.

모범 사례

- COST09-BP01 워크로드 수요 분석 수행:
- COST09-BP02 수요 관리를 위한 버퍼 또는 제한 구현
- COST09-BP03 동적으로 리소스 공급

COST09-BP01 워크로드 수요 분석 수행:

시간별 워크로드 수요를 분석합니다. 분석에서 시기별 추세를 파악하고 전체 워크로드 수명 주기 동안의 작동 상태를 정확하게 반영하는지 확인합니다. 분석 작업은 소요되는 시간 대비 워크로드 비용 등의 제공될 수 있는 이점을 반영해야 합니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

워크로드의 요구 사항을 파악합니다. 조직의 요구 사항에는 요청에 대한 워크로드 응답 시간이 나타나야 합니다. 응답 시간을 사용하면 수요가 관리되는지 여부 또는 리소스 공급이 수요에 따라 변경되는지 여부를 확인할 수 있습니다.

분석에는 수요의 예측 가능성 및 반복 가능성, 수요 변경의 속도 및 수요 변경의 규모가 포함되어야 합니다. 분석은 월말 처리 또는 휴가철 피크와 같은 계절적 변동을 포함하기에 충분히 긴 기간에 걸쳐 수행되어야 합니다.

분석 작업에 확장 구현의 잠재적 이점이 반영되는지 확인하십시오. 구성 요소의 예상 총 비용을 찾아보고 워크로드 수명에 걸쳐 사용량 및 비용이 증가하거나 감소하는지 살펴봅니다.

여러분은 AWS Cost Explorer 또는 Amazon QuickSight 에서 AWS Cost and Usage Report(CUR) 또는 애플리케이션 로그를 사용하여 워크로드 수요를 시각적으로 분석할 수 있습니다.

구현 단계

- 기존 워크로드 데이터 분석: 기존 워크로드, 이전 버전의 워크로드 또는 예측된 사용 패턴의 데이터를 분석합니다. 로그 파일과 모니터링 데이터를 사용하여 고객이 워크로드를 어떻게 사용하는지에 대한 인사이트를 얻을 수 있습니다. 일반적인 지표는 실제 수요(초당 요청 수), 수요 비율이 바뀌는 시간 또는 서로 다른 수준에 있는 시간, 수요 변경률 등이 있습니다. 워크로드의 전체 주기를 분석하여 월말 또는 연말 이벤트와 같은 주기적 변경 사항에 대한 데이터를 수집해야 합니다. 분석에 반영되는 작업량은 워크로드 특성을 반영해야 합니다. 수요 변화가 가장 많은 고가치 워크로드에 가장 많은 작업량을 배치해야 합니다. 수요 변화가 가장 적은 저가치 워크로드에 가장 적은 작업량을 배치해야 합니다. 일반적인 가치 지표는 위험, 브랜드 인식, 수익 또는 워크로드 비용입니다.
- 외부 영향 예측: 워크로드의 수요에 영향을 주거나 변화를 줄 수 있는 조직 전체의 팀원과 만나십시오. 일반적인 팀은 영업, 마케팅 또는 비즈니스 개발입니다. 이들 팀과 협력하여 작업 주기를 확인하고 워크로드 수요를 바꾸는 이벤트가 있는지 확인합니다. 이 데이터로 워크로드 수요를 예측합니다.

리소스

관련 문서:

- AWS Auto Scaling
- AWS 인스턴스 스케줄러
- Amazon SQS 시작하기
- AWS Cost Explorer
- Amazon QuickSight

COST09-BP02 수요 관리를 위한 버퍼 또는 제한 구현

버퍼링 및 조절은 워크로드의 수요를 수정하여 평준화합니다. 클라이언트가 재시도를 수행할 때 조절을 구현합니다. 요청을 저장하고 나중에 처리하도록 버퍼링을 구현합니다. 클라이언트가 필요한 시간에 응답을 수신하도록 제한 및 버퍼가 설계되었는지 확인합니다.

이 모범 사례를 정립하지 않을 경우 노출되는 위험의 수준: 보통

구현 가이드

제한 수요의 소스에 재시도 기능이 있는 경우 조절을 구현할 수 있습니다. 조절은 소스에 현재 요청을 처리할 수 없는 경우 나중에 다시 시도해야 함을 알려줍니다. 소스는 일정 시간 동안 기다린 후 요청을 다시 시도합니다. 조절을 구현하면 워크로드의 최대 리소스 양과 비용을 제한할 수 있는 장점이 있습니다. AWS에서는 Amazon API Gateway 를 사용하여 조절을 구현할 수 있습니다. 접근 관리 구현에 대한 자세한 내용은 Well-Architected 안정성 원칙 백서 를 참조하십시오.

버퍼 기반: 조절과 비슷하게, 버퍼는 서로 다른 속도로 실행되는 애플리케이션이 효과적으로 통신할 수 있도록 요청 처리를 연기합니다. 버퍼링 기반 접근 방식에서는 대기열을 사용해 생산자의 메시지(작업 단위)를 수락합니다. 메시지는 소비자가 읽은 후 처리되므로 소비자의 비즈니스 요구 사항을 충족하는 속도로 메시지를 실행할 수 있습니다. 생산자는 데이터 내구성 및 백 프레셔(소비자 실행 속도가 느려서 생산자의 속도도 느려지는 현상)와 같은 조절 관련 문제를 처리할 필요가 없습니다.

AWS에서는 여러 서비스 중에서 버퍼링 방식을 구현하는 데 적합한 서비스를 선택할 수 있습니다. Amazon Simple Queue Service(Amazon SQS) 는 단일 소비자가 개별 메시지를 읽을 수 있는 대기열을 제공하는 관리형 서비스입니다. Amazon Kinesis 에서는 여러 소비자가 같은 메시지를 읽을 수 있는 스트림을 제공합니다.

버퍼 기반 접근 방식으로 아키텍처를 설계할 때는 필요한 시간에 요청을 처리하도록 워크로드를 설계해야 하며 작업에 대한 중복 요청을 처리할 수 있는 기능이 있어야 합니다.

구현 단계

- 클라이언트 요구 사항 분석: 클라이언트 요청을 분석하여 재시도를 수행할 수 있는지 확인합니다. 재시도를 수행할 수 없는 클라이언트의 경우 버퍼를 구현해야 합니다. 전체 수요, 변경률 및 필요한 응답 시간을 분석하여 필요한 조절 또는 버퍼의 크기를 결정합니다.
- 버퍼 또는 제한 구현: 워크로드에서 버퍼 또는 조절을 구현합니다. Amazon Simple Queue Service(Amazon SQS)와 같은 큐는 워크로드 구성 요소에 버퍼를 제공할 수 있습니다. Amazon API Gateway는 워크로드 구성 요소에 제한 기능을 제공할 수 있습니다.

리소스

관련 문서:

- AWS Auto Scaling
- AWS 인스턴스 스케줄러
- Amazon API Gateway
- Amazon Simple Queue Service
- Amazon SQS 시작하기
- Amazon Kinesis

COST09-BP03 동적으로 리소스 공급

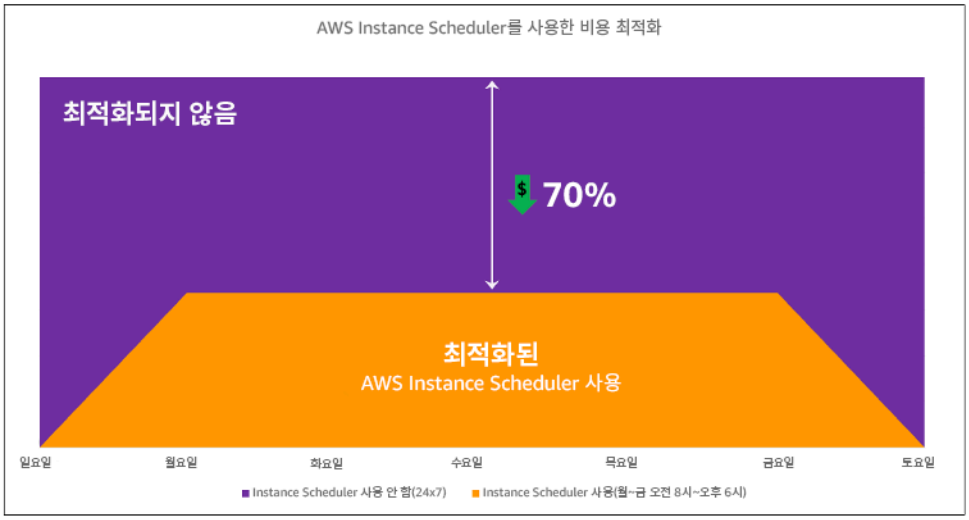
리소스가 계획된 방식으로 프로비저닝됩니다. 자동 크기 조정과 같은 수요 기반이거나, 수요를 예측할 수 있고 리소스가 시간을 기준으로 제공되는 시간 기반일 수 있습니다. 이러한 방법을 사용하면 과다 프로비저닝 또는 과소 프로비저닝을 최소화할 수 있습니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 낮음

구현 가이드

AWS 고객이 애플리케이션에 사용할 수 있는 리소스를 늘리고 수요를 충족하는 리소스를 공급할 수 있는 몇 가지 방법이 있습니다. 이러한 옵션 중 하나는 Amazon Elastic Compute Cloud(Amazon EC2) 및 Amazon Relational Database Service(Amazon RDS) 인스턴스의 시작/중지를 자동화하는 AWS Instance Scheduler를 사용하는 것입니다. 다른 옵션은 AWS Auto Scaling을 사용하는 것입니다. 이 옵션을 사용하면 애플리케이션 또는 서비스의 요구에 따라 컴퓨팅 리소스를 자동으로 조정할 수 있습니다. 수요에 따라 리소스를 공급하면 사용한 리소스에 대해서만 비용을 지불하고 필요할 때 리소스를 해제하여 비용을 절감하며 필요하지 않을 때는 리소스를 종료할 수 있습니다.

AWS Instance Scheduler를 사용하면 정의된 시간에 Amazon EC2 및 Amazon RDS 인스턴스의 중지와 시작을 구성할 수 있습니다. 그러면 예를 들어 저녁 6시 이후에는 필요 없는 Amazon EC2 인스턴스에 매일 오전 8시에 액세스하는 등의 일정한 시간 패턴을 통해 동일한 리소스의 수요를 충족할 수 있습니다. 이 솔루션은 사용하지 않는 리소스는 중지하고 필요할 때 시작하여 운영 비용을 줄여줍니다.



AWS Instance Scheduler를 통한 비용 최적화.

또한 AWS Systems Manager 빠른 설정을 사용하여 간단한 사용자 인터페이스(UI)로 계정 및 리전 전체에서 Amazon EC2 인스턴스의 일정을 쉽게 구성할 수 있습니다. AWS Instance Scheduler로 Amazon EC2 또는 Amazon RDS 인스턴스를 예약하고 기존 인스턴스를 중지 및 시작할 수 있습니다. 그러나 Auto Scaling 그룹(ASG)의 일부이거나 Amazon Redshift 또는 Amazon OpenSearch Service 등과 같은 서비스를 관리하는 인스턴스는 중지 및 시작할 수 없습니다. Auto Scaling 그룹에는 그룹 내 인스턴스에 대한 고유한 일정이 있으며 이러한 인스턴스가 생성됩니다.

AWS Auto Scaling은 용량을 조정하여 최대한 저렴한 비용으로 변화하는 수요를 충족하기 위한 안정적이고 예측 가능한 성능을 유지하는 데 도움이 됩니다. 애플리케이션의 용량을 조정하기 위한 이 완전관리형 무료 서비스는 Amazon EC2 인스턴스, 스팟 플릿, Amazon ECS, Amazon DynamoDB, Amazon Aurora과(와) 통합됩니다. Auto Scaling이 제공하는 자동 리소스 검색 기능을 사용하면 워크로드에서 구성 가능한 리소스를 쉽게 찾을 수 있습니다. 또한 기본적으로 포함되어 있는 규모 조정 전략을 통해 성능, 비용 또는 둘 사이의 균형을 최적화할 수 있으며 예측 규모 조정 기능을 통해 주기적으로 발생하는 스파이크를 지원할 수 있습니다.

Auto Scaling 그룹을 조정하는 데 사용할 수 있는 다양한 조정 옵션이 있습니다.

- 항상 현재 인스턴스 수준을 유지
- 수동으로 규모 조정
- 일정에 따라 규모 조정
- 수요에 따라 규모 조정
- 예측 규모 조정을 사용

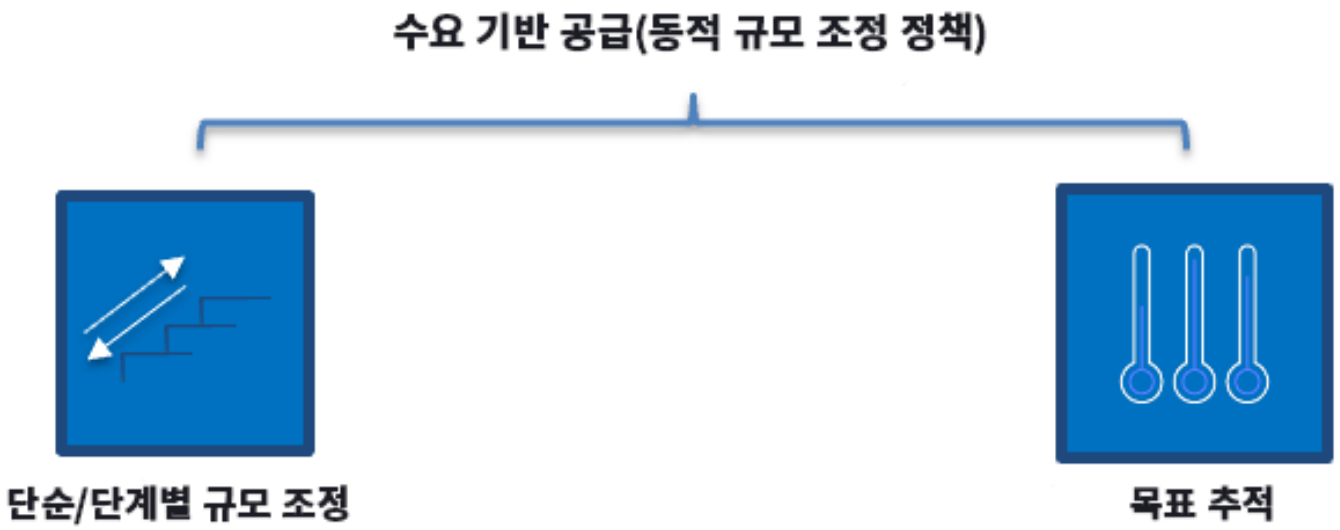
Auto Scaling 정책은 서로 다르며 동적 및 예약 규모 조정 정책으로 분류될 수 있습니다. 동적 정책은 수동 또는 동적 규모 조정으로, 예약 또는 예측 규모 조정입니다. 동적, 예약 및 예측 스케일링에 규모 조정 정책을 사용할 수 있습니다. 또한 Amazon CloudWatch의 지표 및 경보를 사용하여 워크로드에 대한 조정 이벤트를 트리거할 수도 있습니다. 최신 기능 및 개선 사항에 액세스할 수 있는 시작 템플릿을사용하는 것이 좋습니다. 시작 구성을 사용할 때 모든 Auto Scaling 기능을 사용할 수 있는 것은 아닙니다. 예를 들어 스팟 및 온디맨드 인스턴스를 모두 시작하거나 여러 인스턴스 유형을 지정하는 Auto Scaling 그룹은 생성할 수 없습니다. 시작 템플릿을 사용하여 이러한 기능을 구성해야 합니다. 시작 템플릿을 사용할 때는 각 템플릿의 버전을 지정하는 것이 좋습니다. 시작 템플릿의 버전 관리를 통해 전체 파라미터 집합의 하위 집합을 생성할 수 있습니다. 그런 다음 이를 재사용하여 동일한 시작 템플릿의 다른 버전을 생성할 수 있습니다.

AWS Auto Scaling을 사용하거나 AWS API 또는 SDK를사용하여 코드에 규모 조정을 포함할 수 있습니다. 이렇게 하면 환경을 수동으로 변경하는 데 따른 운영 비용이 제거되므로 전반적인 워크로드 비용이 절감되며 변경을 훨씬 더 빠르게 수행할 수 있습니다. 이는 또한 언제든지 수요에 맞춰 워크로드 리소스를 조정합니다. 이 모범 사례를 따르고 조직에 리소스를 동적으로 공급하려면 AWS 클라우드의 수평 및 수직 규모 조정과 Amazon EC2 인스턴스에서 실행 중인 애플리케이션의 특성을 이해해야 합니다. 이 모범 사례를 따르기 위해 클라우드 재무 관리 팀이 기술 팀과 협력하는 것이 좋습니다.

Elastic Load Balancing(Elastic Load Balancing)는 여러 리소스에 걸쳐 수요를 분산하여 규모를 조정하는 데 도움이 됩니다. ASG 및 Elastic Load Balancing를 사용하면 Auto Scaling 그룹 내에서 어떤 인스턴스에도 부하가 걸리지 않도록 트래픽을 최적으로 라우팅하여 들어오는 요청을 관리할 수 있습니다. 요청은 용량 또는 사용률을 고려하지 않고 대상 그룹의 모든 대상에 라운드 로빈 방식으로 분산됩니다.

일반적인 지표로는 CPU 사용률, 네트워크 처리량 및 Elastic Load Balancing에서 관찰된 요청 및 응답 지연 시간과 같은 표준 Amazon EC2 지표가 있습니다. 가능한 경우 고객 경험을 나타내는 지표를 사용해야 합니다. 일반적으로는 워크로드 내 애플리케이션 코드에서 생성될 수 있는 사용자 지정 지표입니다. 이 문서에서는 수요를 동적으로 충족하는 방법을 자세히 설명하기 위해 Auto Scaling을 수요 기반 공급 모델과 시간 기반 공급 모델의 두 범주로 분류하고 각 모델을 심층적으로 살펴보겠습니다.

수요 기반 공급: 클라우드의 탄력성을 활용하여 거의 실시간에 가까운 수요 상태를 기반으로 변화하는 수요를 충족할 리소스를 공급합니다. 수요 기반 공급에서는 API 또는 서비스 기능을 사용하여 프로그래밍 방식을 통해 아키텍처의 클라우드 리소스 양을 변경합니다. 이렇게 하면 아키텍처 구성 요소의 규모를 조정할 수 있으며, 수요 급증 기간 동안에는 리소스 수를 늘려 성능을 유지하고 수요 감소 기간에는 용량을 줄여 비용을 절감할 수 있습니다.

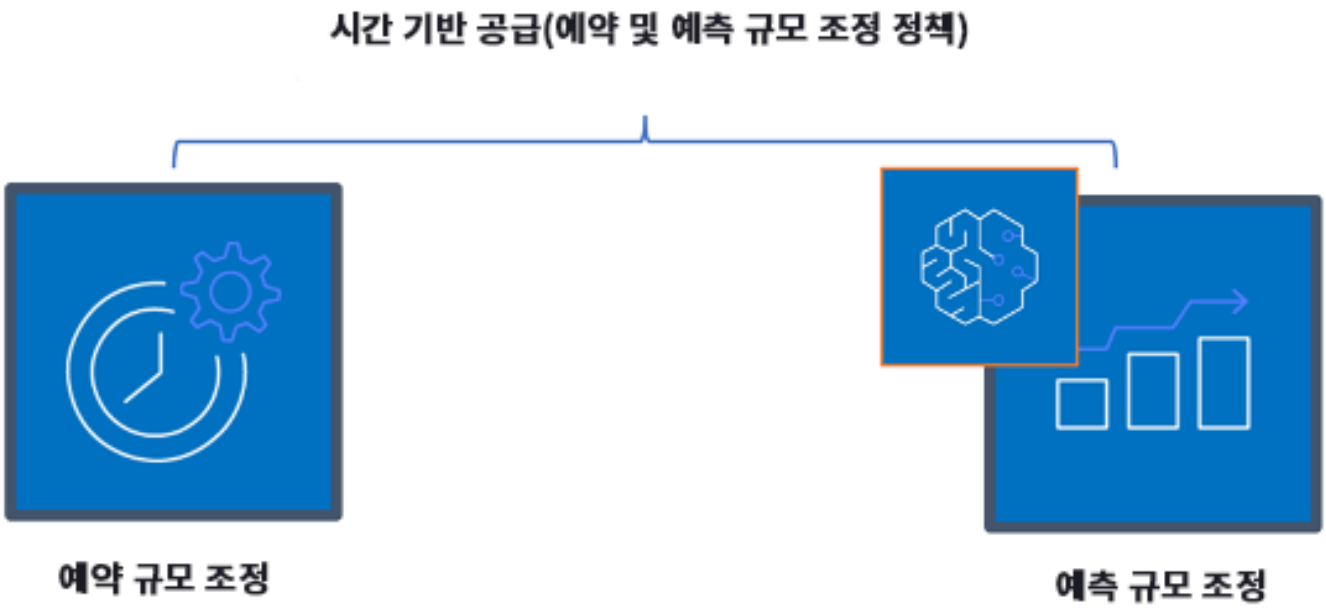


수요 기반 동적 규모 조정 정책

- 단순/단계별 규모 조정: 고객이 수동으로 정의한 단계에 따라 지표를 모니터링하고 인스턴스를 추가/제거합니다.
- 목표 추적: 지표를 사용자가 정의한 목표로 유지하기 위해 인스턴스를 자동으로 추가 또는 제거하는 온도 조절기와 유사한 제어 메커니즘입니다.

수요 기반 방식을 사용하여 설계할 때는 두 가지 주요 사항을 고려해야 합니다. 먼저 새 리소스를 프로비저닝해야 하는 속도를 파악해야 합니다. 그리고 수요와 공급 간의 차이 규모는 변화한다는 점을 이해해야 합니다. 따라서 수요 변화 속도에 맞게 공급 속도를 변경할 수 있도록 준비하는 동시에 리소스 장애에도 대비해야 합니다.

시간 기반 공급: 시간 기반 방식에서는 시간별로 예측 가능하거나 적절하게 정의되는 수요에 맞게 리소스 용량을 조정합니다. 이 방식에서는 일반적으로 리소스 용량이 리소스 사용률 수준에 따라 달라지지 않습니다. 시간 기반 방식을 사용하면 필요한 특정 시간에 리소스를 사용할 수 있으며, 시작 절차 및 시스템 또는 일관성 검사로 인한 지연 없이 리소스를 제공할 수 있습니다. 또한 사용량이 많은 기간 동안 추가 리소스를 제공하거나 용량을 늘릴 수 있습니다.



시간 기반 규모 조정 정책

예약 또는 예측 자동 규모 조정을 사용하여 시간 기반 접근 방식을 구현할 수 있습니다. 사용자 도달 또는 수요 증가 시 리소스를 사용할 수 있도록 정의된 시간(예: 업무 시간 시작 시)에 워크로드 스케일 아웃/인을 예약할 수 있습니다. 예측 크기 조정은 패턴을 사용하여 스케일 아웃하는 반면에 예약된 크기 조정은 미리 정의된 시간을 사용하여 스케일 아웃합니다. 또한 그룹에서 속성 기반 인스턴스 유형 선택(ABS) 전략을 사용하여 vCPU, 메모리 및 스토리지와 같은 일련의 속성으로서의 인스턴스 요구 사항을 표현할 수도 있습니다. 또한 신세대 인스턴스 유형이 릴리스되면 이를 자동으로 사용하고 Amazon EC2 스팟 인스턴스를 통해 더 광범위한 용량에 액세스할 수 있습니다. Amazon EC2 플릿과 Amazon EC2 Auto Scaling는 지정된 속성에 맞는 인스턴스를 선택하고 시작하여 인스턴스 유형을 수동으로 선택할 필요성이 사라집니다.

또한 AWS API, SDK 및 AWS CloudFormation 을 활용하면 전체 환경을 필요할 때 자동으로 프로비저닝하고 폐기할 수 있습니다. 이 방식은 정의된 업무 시간이나 일정 기간 동안에만 실행되는 개발 또는 테스트 환경에 적합합니다. API를 사용해 환경 내에서 리소스 규모를 조정할 수 있습니다(수직 확장). 예를 들어 인스턴스 크기나 클래스를 변경하여 프로덕션 워크로드의 규모를 확장할 수 있습니다. 이렇게 하려면 인스턴스를 중지했다가 시작한 후 다른 인스턴스 크기나 클래스를 선택합니다. 크기를 늘리거나, 성능(IOPS)을 조정하거나, 사용 중에 볼륨 유형을 변경하도록 수정할 수 있는 Amazon EBS 탄력적 볼륨 등의 다른 리소스에도 이 기술을 적용할 수 있습니다.

시간 기반 방식을 사용하여 설계할 때는 두 가지 주요 사항을 고려해야 합니다. 먼저 사용 패턴의 일관성 정도를 파악해야 합니다. 그리고 패턴 변경 시의 영향을 고려해야 합니다. 워크로드를 모니터링하고 비즈니스 인텔리전스를 사용하면 예측 정확도를 높일 수 있습니다. 사용 패턴이 크게 변경되는 경우에는 패턴이 변경된 기간이 포함되도록 시간을 조정할 수 있습니다.

구현 단계

- 예약 규모 조정 구성: 예측 가능한 수요 변화를 위해 시간 기반 조정은 적시에 올바른 개수의 리소스를 제공할 수 있습니다. 리소스 생성 및 구성이 수요 변화에 대응할 만큼 충분히 빠르지 않은 경우에도 유용합니다. 워크로드 분석으로 AWS Auto Scaling을 사용하여 예약된 크기 조정을 구성합니다. 시간 기반 일정을 구성하기 위해 예약된 크기 조정의 예측 크기 조정을 사용하여 예상되는 또는 예측 가능한 로드 변화에 따라 Auto Scaling 그룹 내 Amazon EC2 인스턴스 수를 미리 늘릴 수 있습니다.
- 예측 규모 조정 구성: 예측 규모 조정을 사용하면 트래픽 흐름의 일별 및 주별 패턴에 앞서 Auto Scaling 그룹 내 Amazon EC2 인스턴스 수를 늘릴 수 있습니다. 시작하는 데 오래 걸리는 애플리케이션이 있고 정기적으로 트래픽이 급증하는 경우 예측 크기 조정 사용을 고려해야 합니다. 예측 크기 조정은 예측한 로드가 발생 전에 용량을 초기화함으로써 반응적인 속성의 동적 크기 조정을 단독으로 사용하는 것과 비교하여 더 빠르게 확장할 수 있도록 합니다. 예를 들어, 사용자가 업무 시간 시작과 함께 워크로드를 사용하기 시작하고 업무 시간이 지나면 사용하지 않는 경우, 예측 크기 조정은 업무 시간 전에 용량을 추가할 수 있습니다. 그러면 변화하는 트래픽에 대응하기 위한 동적 크기 조정의 지연이 사라집니다.
- 동적 자동 규모 조정 구성: 활성 워크로드 지표를 기반으로 규모 조정을 구성하려면 Auto Scaling을 사용합니다. 분석을 사용하여 올바른 리소스 수준에서 시작하도록 Auto Scaling을 구성하고 워크로드가 필요한 시간 내에 조정되도록 합니다. 단일 Auto Scaling 그룹 내에서 온디맨드 인스턴스 및 스팟 인스턴스 플릿을 시작하고 자동으로 확장할 수 있습니다. 스팟 인스턴스 사용에 대한 할인을 받는 것은 물론, 예약 인스턴스 또는 절감형 플랜을 사용하여 정규 온디맨드 인스턴스 요금을 할인받을 수 있습니다. 이 모든 요소를 결합하여 Amazon EC2 인스턴스의 비용 절감을 최적화하고 애플리케이션에 대해 원하는 규모와 성능을 얻도록 지원할 수 있습니다.

리소스

관련 문서:

- AWS Auto Scaling
- AWS Instance Scheduler
- Auto Scaling 그룹의 크기 확장
- Amazon EC2 Auto Scaling 시작하기
- Amazon SQS 시작하기

- Amazon EC2 Auto Scaling의 예약 규모 조정
- Amazon EC2 Auto Scaling의 예측 규모 조정

관련 동영상:

- Target Tracking Scaling Policies for Auto Scaling(Auto Scaling에 대한 대상 추적 규모 조정 정책)
- AWS Instance Scheduler

관련 예시:

- Amazon EC2 플릿의 Auto Scaling을 위한 속성 기반 인스턴스 유형 선택
- 예약 규모 조정을 사용하여 Amazon Elastic Container Service 비용 최적화
- Amazon EC2 Auto Scaling으로 예측 스케일링
- Instance Scheduler를 AWS CloudFormation과 함께 사용하여 Amazon EC2 인스턴스를 예약하려면 어떻게 해야 하나요?

시간 경과에 따른 최적화

질문

- COST 10 새로운 서비스를 어떻게 평가합니까?
- COST 11 작업 비용을 어떻게 평가합니까?

COST 10 새로운 서비스를 어떻게 평가합니까?

AWS에서 신규 서비스와 기능이 출시되면 기존에 결정한 아키텍처 관련 사항을 검토하여 비용 측면에서 여전히 가장 효율적인 결정인지 확인하는 것이 좋습니다.

모범 사례

- COST10-BP01 워크로드 검토 프로세스 개발
- COST10-BP02 정기적으로 워크로드 검토 및 분석

COST10-BP01 워크로드 검토 프로세스 개발

워크로드 검토 기준과 프로세스를 정의하는 프로세스를 개발합니다. 검토 작업에는 잠재적 이점이 반영되어야 합니다. 예를 들어 핵심 워크로드 또는 비용이 청구 금액의 10%보다 많은 워크로드는 분기별로 또는 연 2회 검토하고, 비용이 청구 금액의 10%보다 적은 워크로드는 연 1회 검토할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

가장 비용 효율적인 워크로드를 유지하려면 워크로드를 정기적으로 검토하여 새로운 서비스, 기능 및 구성 요소를 구현할 기회가 있는지 확인해야 합니다. 전반적인 비용 절감을 실현하려면 검토 프로세스가 잠재적인 절감액에 비례해야 합니다. 예를 들어 전체 지출의 50%를 차지하는 워크로드는 전체 지출의 5%를 차지하는 워크로드보다 더 정기적으로, 더 철저히 검토되어야 합니다. 외부 요인 또는 변동성 고려. 워크로드가 특정 지역 또는 시장 부문에 서비스를 제공하고 있고 해당 영역의 변화가 예측되는 경우에는 자주 검토하여 비용을 절감할 수 있습니다. 검토에서 고려할 또 다른 요인은 변경 구현에 들어가는 노력입니다. 변경 사항을 테스트하고 검증하는 데 상당한 비용이 발생한다면 검토 빈도를 줄여야 합니다.

오래된 레거시 구성 요소 및 리소스를 유지 관리하는 데 드는 장기적인 비용과 여기에 새로운 기능을 구현할 수 없다는 점을 고려하세요. 현재의 테스트 및 검증 비용이 제안된 이점을 상회할 수 있습니다. 그러나 시간이 지남에 따라 워크로드와 현재 기술 간의 격차가 증가하면 변경 비용이 크게 증가하여 비용이 훨씬 높아질 수 있습니다. 예를 들어 새로운 프로그래밍 언어로 이동하는 비용은 현재로서 비용 효율적이지 않을 수 있습니다. 하지만 5년 안에는 해당 언어에 숙련된 인력을 구하는 비용이 증가할 수 있으며, 워크로드 증가로 인해 더 큰 시스템을 새로운 언어로 전환하는 데 이전보다 더 많은 노력이 필요하게 될 것입니다.

워크로드를 구성 요소로 나누고 구성 요소의 비용을 할당한 다음(추정치로 충분함) 각 구성 요소 옆에 요인(예: 작업에 들어가는 노력 및 외부 시장 요인)을 나열하세요. 이러한 지표를 사용하여 각 워크로드에 대한 검토 빈도를 결정합니다. 예를 들어 높은 비용, 낮은 변경 노력 및 높은 외부 요인으로 분류되는 웹 서버의 경우 검토 빈도가 높을 수 있습니다. 중앙 데이터베이스는 중간 비용, 높은 변화 노력, 낮은 외부 요인으로 분류되므로 검토 빈도는 중간일 수 있습니다.

새 서비스, 설계 패턴, 리소스 유형 및 구성을 사용할 수 있게 되면 워크로드 비용을 최적화하기 위해 평가를 위한 프로세스를 정의합니다. 성능 원칙 검토 및 신뢰성 원칙 검토 프로세스와 마찬가지로 최적화 및 개선 작업과 문제 해결을 식별 및 검증하고 이에 대한 우선순위를 지정한 다음 백로그에 통합합니다.

구현 단계

- 검토 빈도 정의: 워크로드 및 해당 구성 요소를 검토해야 하는 빈도를 정의합니다. 지속적인 개선 및 검토 빈도를 위해 시간과 리소스를 할당하여 워크로드의 효율성 및 최적화를 개선합니다. 이는 여러 가지 요소를 조합한 것으로 조직 내 워크로드마다 다를 수 있으며 워크로드의 구성 요소 간에 다를 수 있습니다. 일반적인 요인으로는 수익 또는 브랜드 측면에서 측정한 조직에 대한 중요성, 워크로드의 총 실행 비용(운영 및 리소스 비용 포함), 워크로드의 복잡성, 변경 실행 용이성, 소프트웨어 라이선스 계약, 변경으로 인해 징벌적 라이선스에 의한 라이선스 비용이 크게 증가하는지 여부 등이 있습니다. 기능적 또는 기술적으로 구성 요소를 정의할 수 있습니다(예: 웹 서버 및 데이터베이스, 컴퓨팅 및 스토리지 리소스). 요인을 적절히 절충하고 워크로드와 해당 구성 요소에 대한 기간을 정하세요. 18개월마다 전체 워크로드를 검토하고, 6개월마다 웹 서버를 검토하고, 12개월마다 데이터베이스를 검토하고, 6개월마다 컴퓨팅 및 단기 스토리지를 검토하고, 12개월마다 장기 스토리지를 검토하기로 결정할 수 있습니다.
- 검토 완전성 정의: 워크로드 또는 워크로드 구성 요소를 검토하는 데 얼마나 많은 노력이 드는지 정의합니다. 검토 빈도와 마찬가지로 여러 가지 요소를 비교 검토하여 절충해야 합니다. 개선 기회를 평가하고 우선순위를 지정해 가장 큰 이점이 제공되는 영역에서 작업을 중점적으로 수행하고 동시에 이러한 작업에 어느 정도의 노력이 필요한지 예측합니다. 예상한 성과가 목표에 미치지 못하고 더 많은 노력이 필요하다면 다른 대안을 찾아서 해당 과정을 반복합니다. 개선 가능한 운영 프로세스를 지속적으로 개선하기 위해서는 검토 프로세스에 전담 리소스와 시간을 포함해야 합니다. 예를 들어, 데이터베이스 구성 요소 분석에 1주일, 컴퓨팅 리소스 분석에 1주일, 스토리지 검토에 4시간을 할애하기로 결정할 수 있습니다.

리소스

관련 문서:

- AWS 뉴스 블로그
- 클라우드 컴퓨팅 유형
- AWS의 새로운 소식

관련 예시:

- AWS Support 사전 예방적 서비스
- SAP 워크로드에 대한 정기 워크로드 검토

COST10-BP02 정기적으로 워크로드 검토 및 분석

지정한 각 프로세스를 기준으로 기존 워크로드를 정기적으로 검토하여 새로운 서비스를 채택할 수 있는지, 기존 서비스를 교체할 수 있는지, 워크로드를 재설계할 수 있는지를 확인하세요.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

AWS는 최신 기술로 보다 빠르게 실험하고 혁신할 수 있도록 새로운 기능을 계속해서 추가하고 있습니다. AWS 새로운 소식에서는 AWS가 이를 수행하는 방법을 자세히 설명하고 AWS 서비스, 기능 및 리전 확장이 출시될 때 간략히 소식을 발표합니다. 공개된 출시 서비스 및 기능을 자세히 살펴보고, 기존 워크로드를 검토하고 분석하는 데 사용할 수 있습니다. 새로운 AWS 서비스 및 기능의 이점을 실현하려면 워크로드에 대한 검토를 실행하고 필요에 따라 새로운 서비스와 기능을 구현해야 합니다. 즉, 워크로드에 사용하는 기존 서비스를 교체하거나 워크로드를 현대화하여 새로운 AWS 서비스를 채택해야 할 수 있습니다. 예를 들어, 워크로드를 검토한 후 메시징 구성 요소를 Amazon Simple Email Service로 대체할 수 있습니다. 이렇게 하면 여러 인스턴스의 운영 및 유지 관리 비용을 없애면서 모든 기능을 더 저렴한 비용으로 제공할 수 있습니다.

워크로드를 분석하고 잠재적인 기회를 강조하려면 최신 서비스뿐만 아니라 새로운 솔루션 구축 방법도 고려해야 합니다. 다른 여러 고객의 아키텍처 설계, 당면 과제 및 솔루션에 대해 알아보려면 AWS의 This is My Architecture 동영상을 살펴보세요. All-In 시리즈를 확인하여 고객 사례와 AWS 서비스의 적용 실례를 살펴보세요. 기본 클라우드 아키텍처 패턴 모범 사례를 설명, 검토 및 분석하는 Back to Basics 동영상 시리즈도 시청할 수 있습니다. 또 다른 소스는 How to Build This 동영상입니다. 이 동영상은 AWS 서비스를 사용하여 최소 기능 제품(MVP)을 실현하는 방법에 대한 아이디어가 있는 사람을 돕고자 마련되었습니다. 이를 통해 아이디어로 똘똘 뭉친 전 세계 빌더가 경험이 풍부한 AWS 솔루션스 아키텍트의 아키텍트 지도를 받을 수 있습니다. 마지막으로, 단계별 자습서로 이루어진 시작하기 리소스 자료를 검토할 수 있습니다.

검토 프로세스를 실행하기 전에 동의한 검토 프로세스를 따르면서 특정 서비스 또는 리전과 성능 요구 사항을 참조하려면 워크로드, 보안 및 데이터 개인 정보 보호 요건에 대한 비즈니스 요구 사항을 준수하세요.

구현 단계

- 정기적으로 워크로드 검토: 정의된 프로세스를 사용하여 지정된 빈도로 검토를 수행합니다. 각 구성 요소에 대해 적절한 노력을 기울였는지 확인합니다. 이 프로세스는 비용 최적화를 위해 서비스를 선택한 초기 설계 프로세스와 유사합니다. 장기적 이점뿐만 아니라 서비스와 서비스가 제공하는 이점인 변경 비용의 이 시간 요소를 분석합니다.

- 새로운 서비스 구현: 분석 결과가 변경 실행인 경우 먼저 워크로드의 기준을 수행하여 결과별로 현재 비용을 파악합니다. 변경을 실행한 다음 분석을 수행하여 결과별로 새로운 비용을 확인합니다.

리소스

관련 문서:

- AWS 뉴스 블로그
- AWS의 새로운 소식
- AWS 설명서
- AWS 시작하기
- AWS 일반 리소스

관련 동영상:

- AWS - This is My Architecture
- AWS - Back to Basics
- AWS - All-In 시리즈
- How to Build This

COST 11 작업 비용을 어떻게 평가합니까?

모범 사례

- COST11-BP01 운영 자동화 실행

COST11-BP01 운영 자동화 실행

클라우드에서의 운영 작업 비용을 평가합니다. 자동화를 사용하여 관리 태스크, 배포 및 기타 운영 작업에 들인 시간과 노력이 줄어든 정도를 정량화할 수 있습니다. 운영 작업에 필요한 시간과 비용을 평가하고 관리 태스크를 자동화하여 수동 작업 비율을 최대한 줄이세요.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

운영을 자동화하면 일관성과 확장성이 향상되고, 가시성, 신뢰성 및 유연성이 높아지며, 비용이 절감되고, 인적 자원을 확보하고 지표를 개선하여 혁신을 가속화할 수 있습니다. 이를 통해 워크로드를 배포, 관리 또는 운영할 때 일관되고 안정적인 환경을 제공할 수 있어 수동 작업의 빈도를 줄이고, 효율성을

개선하며, 결과적으로 기업에 이익을 안겨줍니다. 수동 운영 태스크에 인프라 리소스를 투입하지 않고, 대신 가치가 높은 태스크와 혁신에 투자할 수 있으니 비즈니스 성과가 개선됩니다. 기업은 클라우드에서 워크로드를 관리할 수 있는 검증된 방법을 필요로 합니다. 이러한 솔루션은 위험을 최소화하고 신뢰성을 극대화하면서도 안전하고 빠르며 비용 효율적이어야 합니다.

먼저 클라우드의 전반적인 운영 비용을 살펴보며 필요한 작업을 기준으로 운영의 우선순위를 정하는 일부터 시작하세요. 가령 클라우드에 새 리소스를 배포하거나, 기존 리소스를 최적화하여 변경하거나, 필요한 구성을 구현하는 데 시간이 얼마나 걸리나요? 운영 및 관리 비용을 고려하여 수동 작업으로 인한 총 비용을 살펴봅니다. 관리 태스크에 대한 자동화 우선순위를 지정하여 수동 작업의 비율을 줄이세요. 검토 작업에는 잠재적 이점이 반영되어야 합니다. 자동이 아닌 수동으로 태스크를 수행하는 데 소요된 시간을 예로 들 수 있습니다. 반복적이고 가치가 높은 활동을 자동화하는 것에 우선순위를 두세요. 작업자가 실수할 위험이 높은 활동은 보통 자동화하면 좋습니다. 이러한 위험으로 인해 종종 원치 않는 추가 운영 비용이 발생하기 때문입니다(예: 운영 팀의 초과 근무).

AWS 서비스, 도구 또는 타사 제품을 사용하여 특정 요구 사항에 맞게 구현 및 사용자 지정할 AWS 자동화를 선택할 수 있습니다. 다음 표에는 관리 및 운영을 자동화하는 AWS 서비스를 통해 달성할 수 있는 몇 가지 핵심 운영 기능과 특징이 나와 있습니다.

- AWS Audit Manager: AWS 사용 현황을 꾸준히 감사하여 위험 및 규정 준수 평가를 간소화합니다.
- AWS Backup: 데이터 보호를 중앙에서 관리하고 자동화합니다.
- AWS Config: 컴퓨팅 리소스를 구성하고, 구성 및 리소스 인벤토리를 검증, 감사 및 평가합니다.
- AWS CloudFormation: 인프라를 코드로 사용하여 고가용성 리소스를 실행합니다.
- AWS CloudTrail: IT 변경 관리, 규정 준수 및 제어를 지원합니다.
- Amazon EventBridge: 이벤트를 예약하고 AWS Lambda를 트리거하여 실행합니다.
- AWS Lambda: 반복 프로세스를 이벤트로 트리거하거나 Amazon EventBridge로 확정 일정에 따라 실행하여 자동화합니다.
- AWS Systems Manager: 워크로드 시작 및 중지, 운영 체제 패치, 자동 구성 및 지속적인 관리를 수행합니다.
- AWS Step Functions: 작업을 예약하고 워크플로를 자동화합니다.
- AWS Service Catalog: 소비 및 코드형 인프라를 규정 준수 및 제어로 템플릿을 지정합니다.

팀은 이렇게 절약한 시간을 기술적 부채 청산, 혁신 및 부가 가치 기능에 집중할 수 있습니다. 예를 들어, 온프레미스 환경을 최대한 빠르게 클라우드로 리프트 앤드 시프트하고 최적화는 나중에 수행하는 경우입니다. Amazon Relational Database Service, Amazon EMR, Amazon WorkSpaces, Amazon SageMaker 등 라이선스 비용이 거의 발생하지 않거나 전혀 발생하지 않는 AWS 완전관리형 서비스를

사용하여 실현할 수 있는 비용 절감 이점을 파악하는 것이 좋습니다. 관리형 서비스를 사용하면 서비스 유지 관리를 위한 운영 및 관리 부담이 제거되므로 혁신에 집중할 수 있습니다. 또한 관리형 서비스는 클라우드 규모로 운영되기 때문에 트랜잭션 또는 서비스당 비용을 줄일 수 있습니다.

AWS 제품 및 서비스를 사용하여 즉시 자동화를 채택하고 싶지만, 조직에 기술이 부족한 경우 AWS Managed Services(AMS), AWS Professional Services 또는 AWS 파트너에 문의하여 자동화 채택률을 높이고 클라우드의 운영 효율성을 개선하세요.

AWS Managed Services(AMS)는 엔터프라이즈 고객 및 파트너를 대신하여 AWS 인프라를 운영하는 서비스입니다. 이 서비스를 사용하면 규정을 준수하는 안전한 환경에 워크로드를 배포할 수 있습니다. AMS는 자동화가 포함된 엔터프라이즈 클라우드 운영 모델을 사용하므로 고객은 조직 요구 사항을 충족하면서 클라우드로 더 빠르게 이전하고 지속적인 관리 비용을 절감할 수 있습니다.

AWS Professional Services는 AWS를 통해 원하는 비즈니스 성과를 달성하고 운영을 자동화하도록 도움을 줄 수 있습니다. AWS Professional Services는 엔터프라이즈 클라우드 컴퓨팅의 중심 영역에서 비즈니스 활동을 지원하는 글로벌 전문 사례를 제공합니다. 전문 사례는 솔루션, 기술 및 산업 주제 영역에 걸쳐 모범 사례, 프레임워크, 도구 및 서비스를 통해 맞춤형 지침을 제공합니다. 이를 통해 고객은 클라우드 센터에 최적화된 자동화되고 강력하며 민첩한 IT 운영 및 거버넌스 기능을 배포할 수 있습니다.

구현 단계

- 한 번 구축으로 대규모 배포: AWS CloudFormation, AWS SDK, AWS Command Line Interface(AWS CLI)와 같은 코드형 인프라를 사용하여 한 번 배포하고 동일한 환경이나 재해 복구 시나리오에 여러 번 사용할 수 있습니다. 배포 중 태그를 지정하여 다른 모범 사례에 나와 있는 대로 소비량을 추적합니다. AWS Launch Wizard를 사용하여 널리 사용되는 엔터프라이즈 워크로드를 배포하는 시간을 줄일 수 있습니다. AWS Launch Wizard에서는 AWS 모범 사례에 따라 엔터프라이즈 워크로드의 크기 조정, 구성 및 배포 과정을 안내합니다. AWS Service Catalog를 사용하면 누구나 승인된 셀프 서비스 클라우드 리소스를 검색할 수 있도록 AWS에서 사용 가능한 코드형 인프라 승인 템플릿을 생성하고 관리할 수 있습니다.
- 운영 자동화: 사람의 개입 없이 일상적인 운영 작업을 자동으로 실행합니다. AWS 서비스와 도구를 사용하여 특정 요구 사항에 맞게 구현 및 사용자 지정할 AWS 자동화를 선택할 수 있습니다. 예를 들어, EC2 Image Builder를 사용하여 AWS 또는 온프레미스에서 사용할 가상 시스템 및 컨테이너 이미지를 빌드, 테스트 및 배포할 수 있습니다. 원하는 작업을 AWS 서비스로 수행할 수 없거나 리소스 필터링을 통해 더 복잡한 작업이 필요한 경우 AWS CLI 또는 AWS SDK 도구를 사용하여 운영 작업을 자동화하세요. AWS CLI는 AWS Console을 사용하지 않고 스크립트를 통해 AWS 서비스를 제어하고 관리하는 전체 프로세스를 자동화하는 기능을 제공합니다. AWS 서비스와 상호 작용할 기본 AWS SDK를 선택합니다. 다른 코드 예는 AWS SDK 코드 예제 리포지토리에서 확인하세요.

리소스

관련 문서:

- AWS 클라우드에서 운영 현대화
- AWS Services for Automation(자동화를 위한 AWS 서비스)
- AWS Systems Manager Automation
- AWS automations for SAP administration and operations(SAP 관리 및 운영을 위한 AWS 자동화)
- AWS Managed Services
- AWS Professional Services
- 인프라 및 자동화

관련 예시:

- Reinventing automated operations (Part I)(자동화된 운영 혁신(파트 I))
- Reinventing automated operations (Part II)(자동화된 운영 혁신(파트 II))
- AWS automations for SAP administration and operations(SAP 관리 및 운영을 위한 AWS 자동화)
- IT Automations with AWS Lambda(AWS Lambda를 통한 IT 자동화)
- AWS 코드 예제 리포지토리
- AWS 샘플

지속 가능성

지속 가능성 원칙에는 클라우드 워크로드를 구축할 때 사용된 서비스의 영향을 이해하고, 전체 워크로드 수명 주기에 걸쳐 영향을 정량화하고, 이러한 영향을 줄이기 위한 설계 원칙과 모범 사례를 적용하는 작업이 포함됩니다. 구현 방법에 대한 권장 가이드는 지속 가능성 원칙 백서에서 확인할 수 있습니다.

모범 사례 영역

- 에 설명되어 있습니다
- 수요에 맞춘 조정
- 소프트웨어 및 아키텍처
- 데이터
- 하드웨어 및 서비스

- 프로세스 및 문화

에 설명되어 있습니다

질문

- SUS 1 워크로드에 적합한 리전을 선택하려면 어떻게 해야 합니까?

SUS 1 워크로드에 적합한 리전을 선택하려면 어떻게 해야 합니까?

워크로드 리전의 선택은 성능, 비용 및 탄소 배출량을 포함한 KPI에 큰 영향을 미칩니다. 이러한 KPI를 효과적으로 개선하려면 비즈니스 요구 사항과 지속 가능성 목표를 기준으로 워크로드의 리전을 선택해야 합니다.

모범 사례

- SUS01-BP01 비즈니스 요구 사항과 지속 가능성 목표를 기준으로 리전 선택

SUS01-BP01 비즈니스 요구 사항과 지속 가능성 목표를 기준으로 리전 선택

비즈니스 요구 사항과 지속 가능성 목표를 기준으로 워크로드의 리전을 선택하여 성능, 비용 및 탄소 배출량을 비롯한 KPI를 최적화할 수 있습니다.

일반적인 안티 패턴:

- 자신의 고유한 위치를 기준으로 워크로드의 리전을 선택합니다.
- 모든 워크로드 리소스를 하나의 지리적 위치로 통합합니다.

이 모범 사례 수립의 이점: Amazon 재생 에너지 프로젝트와 가까운 곳이나 탄소 집약도가 낮은 리전에 워크로드를 배치하면 클라우드 워크로드의 탄소 배출량을 줄일 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

글로벌 네트워크 인프라가 함께 연결됨에 따라 AWS 클라우드는 리전 네트워크와 접속 지점(PoP)을 지속적으로 확장하고 있습니다. 워크로드 리전의 선택은 성능, 비용 및 탄소 배출량을 포함한 KPI에 큰 영향을 미칩니다. 이러한 KPI를 효과적으로 개선하려면 비즈니스 요구 사항과 지속 가능성 목표를 기준으로 워크로드의 리전을 선택해야 합니다.

구현 단계

- 규정 준수, 사용 가능한 기능, 비용 및 지연 시간을 비롯한 비즈니스 요구 사항을 기준으로 다음 단계를 따라 워크로드의 잠재적 리전을 평가하고 후보 명단에 추가할 수 있습니다.
  - 필수 지역 규정에 따라 해당 리전이 이를 준수하는지 확인합니다.
  - AWS 리전 서비스 목록을 사용하여 해당 리전에 워크로드를 실행하는 데 필요한 서비스와 기능이 있는지 확인합니다.
  - AWS Pricing Calculator를 사용하여 각 리전의 워크로드 비용을 계산합니다.
  - 최종 사용자 위치와 각 AWS 리전 사이의 네트워크 지연 시간을 테스트합니다.
- Amazon 재생 에너지 프로젝트 근처의 리전 및 그리드의 탄소 집약도가 다른 위치(또는 리전)보다 낮은 리전을 선택합니다.
  - 관련 지속 가능성 지침을 파악하여 온실 가스 협약(시장 기반 및 위치 기반 방법)을 기준으로 매년 탄소 배출량을 추적 및 비교합니다.
  - 탄소 배출량을 추적하는 데 사용하는 방법을 기준으로 리전을 선택합니다. 지속 가능성 지침에 따른 리전 선택에 대한 자세한 내용은 지속 가능성 목표를 기준으로 워크로드의 리전을 선택하는 방법을 참조하세요.

리소스

관련 문서:

- 탄소 배출량 추정치의 이해
- 세계 속의 Amazon
- 재생 에너지 방법론
- 워크로드의 리전을 선택할 때 고려해야 할 사항

관련 동영상:

- 지속 가능성 설계와 AWS 탄소 배출량 절감

수요에 맞춘 조정

질문

- SUS 2 클라우드 리소스를 비즈니스 요구에 어떻게 맞추시겠습니까?

SUS 2 클라우드 리소스를 비즈니스 요구에 어떻게 맞추시겠습니까?

사용자 및 애플리케이션이 워크로드 및 기타 리소스를 사용하는 방식을 통해 지속 가능성 목표를 달성하기 위한 개선 사항을 식별할 수 있습니다. 인프라를 지속적으로 확장하여 수요를 충족하고 사용자를 지원하는 데 필요한 최소 리소스만 활용하는지 확인합니다. 고객 요구 사항에 맞게 서비스 수준을 조정합니다. 사용자 및 애플리케이션이 리소스를 소비하는 데 필요한 네트워크를 제한하도록 리소스를 배치합니다. 사용되지 않는 자산을 제거합니다. 팀원에게 지속 가능성에 미치는 영향을 최소화하면서 요구 사항을 지원하는 디바이스를 제공합니다.

모범 사례

- SUS02-BP01 워크로드 인프라를 동적으로 확장
- SUS02-BP02 SLA를 지속 가능성 목표에 맞게 조정
- SUS02-BP03 미사용 자산의 생성 및 유지 관리 중지
- SUS02-BP04 네트워킹 요구 사항에 따라 워크로드의 지리적 배치 최적화
- SUS02-BP05 수행된 활동에 대한 팀원 리소스 최적화
- SUS02-BP06 버퍼링 또는 제한 개선으로 수요 곡선 완화

SUS02-BP01 워크로드 인프라를 동적으로 확장

클라우드의 탄력성을 활용하고 인프라를 동적으로 확장하여, 클라우드 리소스 공급을 수요에 맞게 조정하고 워크로드의 용량 초과 프로비저닝을 방지할 수 있습니다.

일반적인 안티 패턴:

- 사용자 로드에 따라 인프라를 확장하지 않습니다.
- 항상 인프라를 수동으로 확장합니다.
- 조정 이벤트 후에 다시 축소하는 대신 증가된 용량을 그대로 둡니다.

이 모범 사례 실천의 이점: 워크로드 탄력성을 구성하고 테스트하면 클라우드 리소스 공급을 수요에 효율적으로 일치시키고 용량 초과 프로비저닝을 방지할 수 있습니다. 클라우드의 탄력성을 활용하여 수요가 급증하는 도중과 그 이후에 용량을 자동으로 확장하여 비즈니스 요구 사항을 충족하는 데 필요한 리소스만큼만 사용할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

클라우드는 수요 변화에 맞춰 다양한 메커니즘을 통해 리소스를 동적으로 확장 또는 축소할 수 있는 유연성을 제공합니다. 공급과 수요를 최적으로 일치시키면 워크로드 환경에 미치는 영향이 최소화됩니다.

수요는 고정되거나 가변적일 수 있으므로 관리에 부담이 되지 않도록 측정 기준과 자동화가 필요합니다. 애플리케이션은 인스턴스 크기를 수정하여 수직(위로 또는 아래로)으로 확장하거나 인스턴스 수를 수정하여 수평(안 또는 밖으로)으로 확장하거나 둘을 모두 조합하여 확장할 수 있습니다.

다양한 접근 방식을 사용하여 리소스 공급과 수요를 일치시킬 수 있습니다.

- 타겟 추적 접근법: 스케일링 지표를 모니터링하고 필요에 따라 용량을 자동으로 늘리거나 줄입니다.
- 예측 확장: 일일 및 주간 추세를 예상하여 확장할 수 있습니다.
- 일정 기반 접근법: 예측 가능한 부하 변화에 따라 직접 스케일링 일정을 설정합니다.
- 서비스 스케일링: 기본적으로 설계별로 확장되는 서버리스와 같은 서비스를 선택하거나 자동 스케일링 기능을 제공합니다.

활용률이 낮거나 없는 기간을 식별하고 리소스의 크기를 조정하여 초과 용량을 제거하고 효율성을 개선합니다.

구현 단계

- 탄력성은 보유한 리소스의 공급을 해당 리소스의 수요에 맞춥니다. 인스턴스, 컨테이너 및 함수는 자동 스케일링과 함께, 또는 서비스의 기능을 사용하여 탄력성을 지원하는 메커니즘을 제공합니다. AWS는 사용자 로드가 적은 기간에 워크로드를 빠르고 쉽게 축소할 수 있도록 다양한 자동 스케일링 메커니즘을 제공합니다. 자동 스케일링 메커니즘의 예시는 다음과 같습니다.

Auto scaling mechanism	Where to use
Amazon EC2 Auto Scaling	애플리케이션의 사용자 로드를 처리하는 데 사용할 수 있는 적절한 수의 Amazon EC2 인스턴스가 있는지 확인하는 데 사용합니다.
Application Auto Scaling	Amazon EC2를 벗어난 개별 AWS 서비스(예: Lambda 기능 또는 Amazon Elastic Container Service (Amazon ECS) 서비스)에 대한 리소스를 자동으로 확장하는 데 사용합니다.

Auto scaling mechanism	Where to use
Kubernetes Cluster Autoscaler	AWS에서 Kubernetes 클러스터를 자동으로 확장하는 데 사용합니다.

- 확장은 대개 Amazon EC2 인스턴스나 AWS Lambda 함수 등의 컴퓨팅 서비스와 관련하여 설명하는 경우가 많습니다. Amazon DynamoDB 읽기/쓰기 용량 단위나 Amazon Kinesis Data Streams 샤드와 같은 컴퓨팅 외의 서비스를 구성할 때는 수요에 일치시키는 것이 좋습니다.
- 스케일 업 또는 스케일 다운 대한 지표가 배포 중인 워크로드 유형에 대해 검증되었는지 확인합니다. 동영상 트랜스코딩 애플리케이션을 배포하는 경우 100%의 CPU 활용률이 예상되므로, 기본 지표로 사용해서는 안 됩니다. 필요한 경우 맞춤형 지표(예: 메모리 사용률)을 스케일링 정책에 사용할 수 있습니다. 올바른 지표를 선택하려면 Amazon EC2에 대한 다음 지침을 고려하세요.
  - 지표는 유효한 사용률 지표여야 하며 인스턴스가 얼마나 많이 사용되는지를 설명해야 합니다.
  - 지표 값은 Auto Scaling 그룹 내 인스턴스 수에 비례하여 늘거나 줄어야 합니다.
- Auto Scaling 그룹에는 수동 스케일링 대신 동적 스케일링을 사용합니다. 또한, 동적 스케일링에는 타겟 추적 스케일링 정책을 사용할 것을 권합니다.
- 워크로드 배포에서 확장 및 축소 이벤트를 모두 처리할 수 있는지 확인합니다. 축소 이벤트에 대한 테스트 시나리오를 생성하여 워크로드가 예상대로 작동하고 사용자 환경에 영향(예: 스티키 세션 손실)을 미치지 않는지 확인합니다. 활동 기록을 사용하여 Auto Scaling 그룹의 스케일링 활동을 확인할 수 있습니다.
- 워크로드의 예측 가능한 패턴을 평가하고 예측 및 계획된 수요 변화에 따라 사전 예방적으로 확장합니다. 예측 스케일링에서는 용량을 과도하게 프로비저닝할 필요가 없습니다. 자세한 내용은 Amazon EC2 Auto Scaling을 사용한 예측 스케일링을 참조하세요.

리소스

관련 문서:

- Amazon EC2 Auto Scaling 시작하기
- Predictive Scaling for EC2, Powered by Machine Learning(기계 학습 기반 EC2용 예측 확장)
- Analyze user behavior using Amazon OpenSearch Service, Amazon Data Firehose and Kibana(Amazon OpenSearch Service, Amazon Kinesis Data Firehose 및 Kibana를 사용하여 사용자 행동 분석)
- What is Amazon CloudWatch?(CloudWatch란 무엇인가요?)

- Monitoring DB load with Performance Insights on Amazon RDS(Amazon RDS의 성능 개선 도우미로 DB 로드 모니터링)
- Introducing Native Support for Predictive Scaling with Amazon EC2 Auto Scaling(Amazon EC2 Auto Scaling을 사용한 예측 확장에 대한 기본 지원 소개)
- Introducing Karpenter - An Open-Source, High-Performance Kubernetes Cluster Autoscaler(Karpenter - 고성능 오픈 소스 Kubernetes Cluster Autoscaler 소개)
- Deep Dive on Amazon ECS Cluster Auto Scaling(Elastic Container Service 클러스터 자동 스케일링 심층 분석)

관련 동영상:

- Build a cost-, energy-, and resource-efficient compute environment(비용, 에너지, 리소스 효율적인 컴퓨팅 환경 구축)
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2(더 정확하고, 더 빠르고, 더 저렴한 컴퓨팅: Amazon EC2 비용 최적화)(CMP202-R1)

관련 예시:

- 실습: Amazon EC2 Auto Scaling 그룹 예시
- 실습: Karpenter를 사용하여 오토스케일링 구현

SUS02-BP02 SLA를 지속 가능성 목표에 맞게 조정

지속 가능성 목표를 기준으로 서비스 수준에 관한 계약(SLA)을 검토 및 최적화하여 계속해서 비즈니스 필요를 충족하면서 워크로드를 지원하는 데 필요한 리소스를 최소화합니다.

일반적인 안티 패턴:

- 워크로드 SLA가 알려져 있지 않거나 모호합니다.
- 가용성 및 성능에 대해서만 SLA를 정의합니다.
- 모든 워크로드에 대해 동일한 설계 패턴(예: 다중 AZ 아키텍처)을 사용합니다.

이 모범 사례 확립의 이점: 지속 가능성 목표에 맞춰 SLA를 조정하면 비즈니스 요구 사항을 충족하면서 리소스 사용을 최적화할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

SLA는 클라우드 워크로드에서 예상되는 서비스 수준(예: 응답 시간, 가용성, 데이터 보존 등)을 정의합니다. SLA는 클라우드 워크로드의 아키텍처, 리소스 사용 및 환경 영향에 영향을 미칩니다. 주기적으로 SLA를 검토하여 허용 가능한 수준으로 서비스를 줄여 리소스 사용을 크게 줄이는 절충안을 제시합니다.

구현 단계

- 비즈니스 요구 사항을 충족하되 초과하지는 않으면서 지속 가능성 목표를 지원하는 SLA를 정의 또는 재설계합니다.
- 허용 가능한 수준으로 서비스를 줄여 지속 가능성에 미치는 영향을 크게 줄이는 절충안을 제시합니다.
  - 지속 가능성 및 신뢰성: 가용성이 뛰어난 워크로드는 리소스를 더 많이 사용하는 경향이 있습니다.
  - 지속 가능성 및 성능: 성능을 높이기 위해 리소스를 더 많이 사용하면 환경에 미치는 영향이 커질 수 있습니다.
  - 지속 가능성 및 보안: 지나치게 안전한 워크로드는 환경에 미치는 영향이 커질 수 있습니다.
- 비즈니스에 중요한 기능에 우선순위를 두고 중요하지 않은 기능에 대해 더 낮은 서비스 수준(예: 응답 시간 또는 복구 시간 목표)을 허용하는 설계 패턴(예: AWS의 마이크로서비스)을 사용합니다.

리소스

관련 문서:

- AWS 서비스 수준에 관한 계약(SLA)
- SaaS 공급자의 서비스 수준 계약 중요성

관련 동영상:

- Delivering sustainable, high-performing architectures(지속 가능한 고성능 아키텍처 제공)
- Build a cost-, energy-, and resource-efficient compute environment(비용, 에너지, 리소스 효율적인 컴퓨팅 환경 구축)

SUS02-BP03 미사용 자산의 생성 및 유지 관리 중지

워크로드에서 사용되지 않는 자산을 폐기하여 수요를 지원하는 데 필요한 클라우드 리소스 수를 줄이고 낭비를 최소화합니다.

일반적인 안티 패턴:

- 중복되거나 더 이상 필요하지 않은 자산에 대해 애플리케이션을 분석하지 않습니다.
- 중복되거나 더 이상 필요하지 않은 자산을 제거하지 않습니다.

이 모범 사례 확립의 이점: 사용되지 않는 자산을 제거하면 리소스가 확보되고 워크로드의 전체 효율성이 향상됩니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

사용되지 않는 자산은 스토리지 공간 및 컴퓨팅 전력과 같은 클라우드 리소스를 소비합니다. 이러한 자산을 식별하고 제거함으로써 리소스를 확보하여 클라우드 아키텍처의 효율성을 높일 수 있습니다. 애플리케이션 자산(예: 사전 컴파일된 보고서, 데이터 세트, 정적 이미지 등)과 자산 액세스 패턴을 정기적으로 분석하여 중복된 자산, 활용률이 낮은 자산 및 잠재적 폐기 대상을 식별합니다. 이러한 중복 자산을 제거하여 워크로드의 리소스 낭비를 줄이세요.

구현 단계

- 모니터링 도구를 사용하여 더 이상 필요하지 않은 정적 자산을 식별합니다.
- 자산을 제거하기 전에 제거로 인해 아키텍처가 받는 영향을 평가합니다.
- 플랜을 세우고 더 이상 필요하지 않은 자산을 제거합니다.
- 중복 생성 자산을 통합하여 중복 처리를 제거합니다.
- 애플리케이션을 업데이트하여 더 이상 필요 없는 자산을 생성하고 저장하지 않습니다.
- 더 이상 필요하지 않은 관리 자산의 생산 및 저장을 중단하도록 제3자에게 지시합니다.
- 생성된 중복 자산을 통합하도록 제3자에게 지시합니다.
- 워크로드를 정기적으로 검토하여 사용되지 않는 자산을 식별하고 제거합니다.

리소스

관련 문서:

- Optimizing your AWS Infrastructure for Sustainability, Part II: Storage(지속 가능성을 위한 AWS 인프라 최적화, 파트 II: 스토리지)
- AWS 계정에서 더 이상 필요하지 않은 활성 리소스를 종료하려면 어떻게 해야 하나요?

관련 동영상:

- How do I check for and then remove active resources that I no longer need on my AWS 계정?(AWS 계정에서 더 이상 필요하지 않은 활성 리소스를 확인하고 제거하려면 어떻게 해야 하나요?)

SUS02-BP04 네트워킹 요구 사항에 따라 워크로드의 지리적 배치 최적화

네트워크 트래픽이 이동해야 하는 거리를 단축하고 워크로드를 지원하는 데 필요한 총 네트워크 리소스를 줄일 수 있는 워크로드의 클라우드 위치 및 서비스를 선택합니다.

일반적인 안티 패턴:

- 자신의 고유한 위치를 기준으로 워크로드의 리전을 선택합니다.
- 모든 워크로드 리소스를 하나의 지리적 위치로 통합합니다.
- 모든 트래픽이 기존 데이터 센터를 통과합니다.

이 모범 사례 확립의 이점: 사용자에게 가깝게 워크로드를 배치하면 지연 시간을 최대한 단축할 수 있고 동시에 네트워크 간 데이터 이동을 줄이고 환경에 미치는 영향을 줄일 수 있습니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 보통

구현 가이드

AWS 클라우드 인프라는 리전, 가용 영역, 배치 그룹, 엣지 로케이션(예: AWS Outposts 및 AWS 로컬 영역)과같은 위치 옵션을 중심으로 구축됩니다. 이러한 위치 옵션은 애플리케이션 구성 요소, 클라우드 서비스, 엣지 네트워크, 온프레미스 데이터 센터 간의 연결을 유지합니다.

워크로드의 네트워크 액세스 패턴을 분석하여 이러한 클라우드 위치 옵션을 사용하는 방법을 식별하고 네트워크 트래픽이 이동해야 하는 거리를 줄이십시오.

구현 단계

- 워크로드의 네트워크 액세스 패턴을 분석하여 사용자가 애플리케이션을 사용하는 방법을 식별합니다.
  - 예를 들어 Amazon CloudWatch 및 AWS CloudTrail같은 모니터링 도구를 사용하여 네트워크 활동에 대한 데이터를 수집합니다.
  - 데이터를 분석하여 네트워크 액세스 패턴을 식별합니다.
- 다음과 같은 주요 요소를 토대로 하여 워크로드 배포용 리전을 선택합니다.

  - 지속 가능성 목표: 리전 선택 에 설명되어 있습니다.
  - 데이터 위치: 데이터를 많이 사용하는 애플리케이션의 경우(예: 빅 데이터 및 기계 학습) 애플리케이션 코드는 최대한 데이터와 가까운 위치에서 실행되어야 합니다.
  - 사용자의 위치: 사용자가 직접 사용하는 애플리케이션의 경우 워크로드의 사용자와 가까운 리전(또는 리전들)을 선택합니다.
  - 기타 제약 조건: AWS 아키텍처 블로그 워크로드의 리전을 선택할 때 고려해야 할 사항에서 설명한 것처럼 비용 및 규정 준수 등의 제약을 고려해야 합니다.
- 자주 사용되는 자산에 로컬 캐싱 또는 AWS 캐싱 솔루션을 사용하여 성능을 개선하고 데이터 이동을 줄이며 환경에 미치는 영향을 줄입니다.

서비스	사용 시기
Amazon CloudFront	이미지, 스크립트, 동영상 등의 정적 콘텐츠와 API 응답 또는 웹 애플리케이션 등의 동적 콘텐츠를 캐시하는 데 사용합니다.
Amazon ElastiCache	웹 애플리케이션의 콘텐츠를 캐시하는 데 사용합니다.
DynamoDB Accelerator	DynamoDB 테이블에 인 메모리 가속화를 추가하는 데 사용합니다.

- 워크로드 사용자에게 더 가까운 위치에서 코드를 실행할 수 있는 서비스를 사용합니다.

서비스	사용 시기
Lambda@Edge	객체가 캐시에 없는 경우 시작되는 컴퓨팅 집약적 작업에 사용합니다.
Amazon CloudFront 함수	HTTP(s) 요청 또는 응답 조작 등과 같이 단기 실행 함수에 의해 시작될 수 있는 간단한 사용 사례에 사용합니다.
AWS IoT Greengrass	커넥티드 디바이스를 위한 로컬 컴퓨팅, 메시징 및 데이터 캐시를 실행하는 데 사용합니다.

- 연결 풀을 사용하여 연결을 재사용하고 필요한 리소스를 줄입니다.

- 지속적 연결 및 동기식 업데이트에 의존하지 않는 분산 데이터 스토어를 사용하여 리전별 사용자 집단을 일관되게 지원합니다.
- 사전 프로비저닝된 정적 네트워크 용량을 공유 동적 용량으로 교체하고 네트워크 용량의 지속 가능성에 미치는 영향을 다른 구독자와 공유합니다.

리소스

관련 문서:

- Optimizing your AWS Infrastructure for Sustainability, Part III: Networking(지속 가능성을 위한 AWS 인프라 최적화, 파트 III: 네트워킹)
- Amazon ElastiCache 설명서
- Amazon CloudFront란 무엇인가요?
- Amazon CloudFront 주요 기능

관련 동영상:

- Demystifying data transfer on AWS(AWS에서의 데이터 전송 알아보기)
- 차세대 Amazon EC2 인스턴스의 네트워크 성능 확장

관련 예시:

- AWS 네트워킹 워크숍
- 지속 가능성을 위한 설계 - 네트워크 간 데이터 이동 최소화

SUS02-BP05 수행된 활동에 대한 팀원 리소스 최적화

팀원에게 제공되는 리소스를 최적화하여 팀원에게 필요한 지원을 충분히 제공하면서도 환경 지속 가능성에 미치는 영향을 최소화합니다.

일반적인 안티 패턴:

- 팀원이 사용하는 디바이스가 클라우드 애플리케이션의 전반적인 효율성에 미치는 영향을 무시합니다.
- 팀원이 사용하는 리소스를 수동으로 관리하고 업데이트합니다.

이 모범 사례 확립의 이점: 팀원 리소스를 최적화하면 클라우드 지원 애플리케이션의 전반적인 효율성이 향상됩니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

팀원이 서비스를 사용하는 데 활용하는 리소스, 예상 수명 주기, 재무 및 지속 가능성에 미치는 영향을 이해합니다. 이러한 리소스를 최적화하기 위한 전략을 구현합니다. 예를 들어, 활용률이 낮은 고성능 단일 사용자 시스템 대신 활용률이 높은 확장 가능한 인프라에서 렌더링 및 컴파일과 같은 복잡한 작업을 수행합니다.

구현 단계

- 사용 방식에 맞게 워크스테이션 및 기타 디바이스를 프로비저닝합니다.
- 가상 데스크톱 및 애플리케이션 스트리밍을 사용하여 업그레이드 및 디바이스 요구 사항을 제한합니다.
- 프로세서 또는 메모리 집약적인 태스크를 클라우드로 옮겨 탄력성을 활용합니다.
- 디바이스 수명 주기에 대한 프로세스 및 시스템의 영향을 평가하고 비즈니스 요구 사항을 충족하면서 디바이스 교체 요구 사항을 최소화하는 솔루션을 선택합니다.
- 디바이스에 대한 원격 관리를 구현하여 필요한 출장을 줄입니다.
  - AWS Systems Manager Fleet Manager는 AWS 또는 온프레미스에서 실행 중인 노드를 원격으로 관리하는 데 도움이 되는 통합 사용자 인터페이스(UI) 환경입니다.

리소스

관련 문서:

- Amazon WorkSpaces란 무엇인가요?
- Amazon WorkSpaces용 Cost Optimizer
- Amazon AppStream 2.0 설명서
- NICE DCV

관련 동영상:

- Managing cost for Amazon WorkSpaces on AWS(AWS에서 Amazon Workspaces의 비용 관리)

SUS02-BP06 버퍼링 또는 제한 개선으로 수요 곡선 완화

버퍼링 및 제한은 수요 곡선을 완화하고 워크로드에 필요한 프로비저닝 용량을 줄입니다.

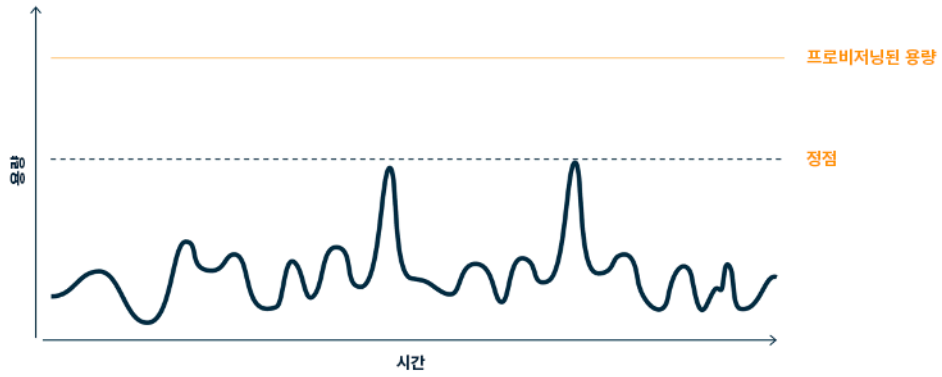
일반적인 안티 패턴:

- 클라이언트 요청은 필요하지 않아도 즉시 처리합니다.
- 클라이언트 요청에 대한 요구 사항을 분석하지 않습니다.

이 모범 사례 확립의 이점: 수요 곡선을 완화하면 워크로드에 필요한 프로비저닝 용량이 줄어듭니다. 프로비저닝 용량을 줄이면 에너지 소비와 환경에 미치는 영향도 줄어듭니다.

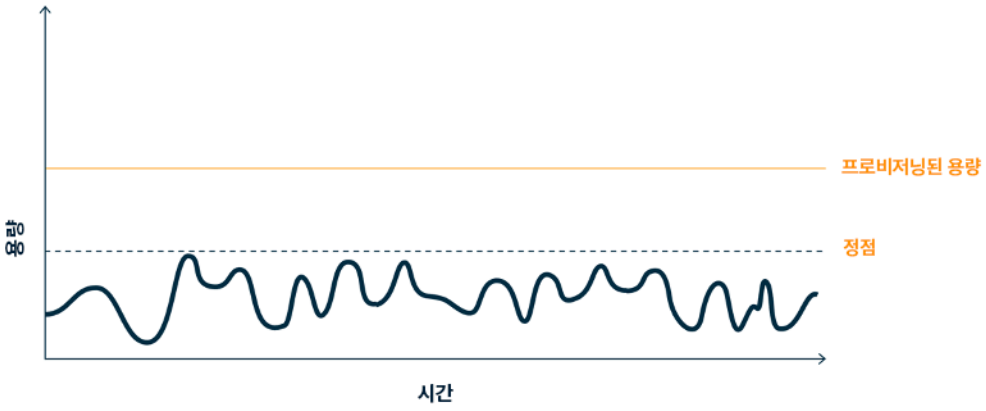
이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

워크로드 수요 곡선을 완화하면 워크로드에 프로비저닝된 용량을 줄이고 환경에 미치는 영향도 줄일 수 있습니다. 아래 그림에 표시된 수요 곡선을 바탕으로 워크로드를 가정합니다. 이 워크로드에는 2개의 정점이 있으며, 이러한 정점을 처리하기 위해 주황색 선으로 표시된 리소스 용량이 프로비저닝됩니다. 이 워크로드에 사용되는 리소스와 에너지는 수요 곡선 아래의 영역이 아니라 프로비저닝된 용량 선 아래의 영역으로 표시됩니다. 이 2가지 정점을 처리하려면 프로비저닝된 용량이 필요하기 때문입니다.



높은 프로비저닝 용량이 필요한 2개의 고유한 정점이 존재하는 수요 곡선입니다.

버퍼링 또는 제한을 사용하여 수요 곡선을 수정하고 정점을 완화할 수 있습니다. 이렇게 되면 프로비저닝된 용량과 소비되는 에너지가 줄어듭니다. 클라이언트가 재시도를 수행할 때 제한을 구현합니다. 요청을 저장하고 나중에 처리하도록 버퍼링을 구현합니다.



수요 곡선 및 프로비저닝된 용량에 대한 제한의 효과입니다.

구현 단계

- 클라이언트 요청을 분석하여 응답 방법을 결정합니다. 고려해야 할 질문은 다음과 같습니다.
  - 이 요청을 비동기식으로 처리할 수 있는가?
  - 클라이언트에 재시도 기능이 있는가?
- 클라이언트에 재시도 기능이 있는 경우 현재 요청을 처리할 수 없으면 나중에 다시 시도해야 함을 소스에 알려주는 제한 기능을 구현할 수 있습니다.
  - Amazon API Gateway를 사용하여 제한을 구현할 수 있습니다.
- 재시도를 수행할 수 없는 클라이언트의 경우 수요 곡선을 완화하려면 버퍼를 구현해야 합니다. 버퍼는 서로 다른 속도로 실행되는 애플리케이션이 효과적으로 통신할 수 있도록 요청 처리를 연기합니다. 버퍼 기반 접근 방식은 대기열 또는 스트림을 사용하여 생산자의 메시지를 수락합니다. 메시지는 소비자가 읽은 후 처리되므로 소비자의 비즈니스 요구 사항을 충족하는 속도로 메시지를 실행할 수 있습니다.
  - Amazon Simple Queue Service(Amazon SQS)는 단일 소비자가 개별 메시지를 읽을 수 있는 대기열을 제공하는 관리형 서비스입니다.
  - Amazon Kinesis에서는 여러 소비자가 같은 메시지를 읽을 수 있는 스트림을 제공합니다.
- 전체 수요, 변경률 및 필수 응답 시간을 분석하여 필요한 제한 또는 버퍼의 크기를 적절하게 조정합니다.

리소스

관련 문서:

- Amazon SQS 시작하기
- Application integration Using Queues and Messages(대기열 및 메시지를 사용한 애플리케이션 통합)

관련 동영상:

- Choosing the Right Messaging Service for Your Distributed App(분산형 앱에 적합한 메시징 서비스 선택)

소프트웨어 및 아키텍처

질문

- SUS 3 소프트웨어 및 아키텍처 패턴을 활용하여 지속 가능성 목표를 지원하려면 어떻게 해야 합니까?

SUS 3 소프트웨어 및 아키텍처 패턴을 활용하여 지속 가능성 목표를 지원하려면 어떻게 해야 합니까?

로드 평준화를 수행하고 배포된 리소스의 높은 활용률을 일관되게 유지하여 소비되는 리소스를 최소화하기 위한 패턴을 구현합니다. 구성 요소는 시간 경과에 따른 사용자 행동의 변화로 인해 사용 부족으로 인해 유휴 상태가 될 수 있습니다. 패턴과 아키텍처를 수정하여 활용률이 낮은 구성 요소를 통합함으로써 전체 활용률을 높입니다. 더 이상 필요하지 않은 구성 요소를 폐기합니다. 워크로드 구성 요소의 성능을 이해하고 리소스를 가장 많이 사용하는 구성 요소를 최적화합니다. 고객이 서비스에 액세스하고 패턴을 구현하는 데 사용하는 디바이스를 숙지하여 디바이스 업그레이드 필요성을 최소화합니다.

모범 사례

- SUS03-BP01 비동기식 및 예약된 작업을 위한 소프트웨어 및 아키텍처 최적화
- SUS03-BP02 사용 빈도가 낮거나 전혀 없는 워크로드 구성 요소 제거 또는 리팩터링
- SUS03-BP03 가장 많은 시간 또는 리소스를 소모하는 코드 영역 최적화
- SUS03-BP04 디바이스 및 장비에 대한 영향 최적화
- SUS03-BP05 데이터 액세스 및 스토리지 패턴을 가장 잘 지원하는 소프트웨어 패턴 및 아키텍처 사용

SUS03-BP01 비동기식 및 예약된 작업을 위한 소프트웨어 및 아키텍처 최적화

배포된 리소스의 일관되고 높은 사용률을 유지할 수 있도록 대기열 기반과 같은 효율적인 소프트웨어 및 아키텍처 패턴을 사용합니다.

일반적인 안티 패턴:

- 클라우드 워크로드의 리소스를 과다하게 프로비저닝하여 예상치 못한 수요 급증이 발생합니다.
- 아키텍처가 메시징 구성 요소에 의한 비동기식 메시지의 발신자와 수신자를 분리하지 않습니다.

이 모범 사례 확립의 이점:

- 효율적인 소프트웨어 및 아키텍처 패턴이 워크로드의 사용되지 않는 리소스를 최소화하고 전체적인 효율성을 개선합니다.
- 비동기식 메시지 수신과 관계없이 처리를 확장할 수 있습니다.
- 메시징 구성 요소를 통해 가용성 요구 사항이 완화되어 더 적은 리소스로 이를 충족할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

이벤트 기반 아키텍처와 같은 효율적인 아키텍처 패턴을 사용하면 구성 요소의 사용률이 균등해지고 워크로드의 과다한 프로비저닝을 최소화할 수 있습니다. 효율적인 아키텍처 패턴을 사용하면 시간 경과에 따른 수요 변화로 인해 사용하지 않는 유휴 리소스를 최소화할 수 있습니다.

워크로드 구성 요소의 요구 사항을 이해하고 리소스의 전체 사용률을 높이는 아키텍처 패턴을 도입합니다. 더 이상 필요하지 않은 구성 요소를 폐기합니다.

구현 단계

- 워크로드에 대한 수요를 분석하여 이에 대한 대응 방법을 결정합니다.
- 동기식 응답이 필요하지 않은 요청이나 작업의 경우 대기열 기반 아키텍처 및 오토 스케일링 작업자를 사용하여 사용률을 극대화합니다. 대기열 기반 아키텍처를 고려해야 하는 몇 가지 예는 다음과 같습니다.

Queuing mechanism	Description
AWS Batch 작업 대기열	AWS Batch 작업은 컴퓨팅 환경에서 실행되도록 예약될 때까지 작업 대기열로 제출됩니다.
Amazon Simple Queue Service 및 Amazon EC2 스팟 인스턴스	Amazon SQS와 스팟 인스턴스를 페어링하여 내결함성이 있고 효율적인 아키텍처를 구축합니다.

- 언제든 처리할 수 있는 요청이나 작업의 경우 더 높은 효율을 위해 예약 메커니즘을 사용하여 작업을 일괄 처리합니다. AWS의 예약 메커니즘의 예는 다음과 같습니다.

Scheduling mechanism	Description
Amazon EventBridge 스케줄러	Amazon EventBridge의 기능을 통해 대규모로 예약된 작업을 생성, 실행 및 관리할 수 있습니다.
AWS Glue 시간 기반 일정	AWS Glue의 크롤러와 작업에 대한 시간 기반 일정을 정의합니다.
Amazon Elastic Container Service (Amazon ECS) 예약된 작업	Amazon ECS는 예약된 작업 생성을 지원합니다. 예약된 작업은 Amazon EventBridge 규칙을 사용하여 예약에 따라 또는 EventBridge 이벤트에 대한 응답으로 작업을 실행합니다.
Instance Scheduler	Amazon EC2 및 Amazon Relational Database Service 인스턴스의 시작 및 종료 예약을 구성합니다.

- 아키텍처에서 폴링과 웹후크 메커니즘을 사용하는 경우 이를 이벤트로 바꿉니다. 이벤트 기반 아키텍처를 사용하여 고효율 워크로드를 구축합니다.
- AWS의 서버리스를 활용하여 과다하게 프로비저닝된 인프라를 제거합니다.
- 입력 대기 중인 유휴 리소스를 방지하기 위해 아키텍처의 개별 구성 요소의 적절한 크기를 지정합니다.

리소스

관련 문서:

- Amazon Simple Queue Service란 무엇인가요?
- Amazon MQ란 무엇인가요?
- Amazon SQS 기반 크기 조정
- AWS Step Functions란 무엇인가요?
- AWS Lambda란 무엇인가요?
- Amazon SQS에서 AWS Lambda 사용
- Amazon EventBridge란 무엇인가요?

관련 동영상:

- 이벤트 기반 아키텍처로의 전환

SUS03-BP02 사용 빈도가 낮거나 전혀 없는 워크로드 구성 요소 제거 또는 리팩터링

사용되지 않아 더 이상 필요하지 않은 구성 요소를 제거하고 활용률이 낮은 구성 요소를 리팩터링하여 워크로드에서 낭비되는 리소스를 최소화합니다.

일반적인 안티 패턴:

- 워크로드의 개별 구성 요소 사용률 수준을 정기적으로 확인하지 않습니다.
- AWS 적정 크기 조정 도구(예: AWS Compute Optimizer)에서 권장 사항을 확인하고 분석하지 않습니다.

이 모범 사례 확립의 이점: 사용되지 않는 구성 요소를 제거하면 낭비를 최소화하고 클라우드 워크로드의 전반적인 효율성을 향상할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

워크로드를 검토하여 유휴 상태이거나 사용하지 않는 구성 요소를 식별합니다. 이는 수요 변화 또는 새로운 클라우드 서비스 출시로 촉발될 수 있는 반복적인 개선 프로세스입니다. 예를 들어, AWS Lambda 함수 실행 시간의 급격한 저하는 메모리 크기를 줄여야 하는 필요성을 나타내는 지표가 될 수

있습니다. 또한, AWS에서 새로운 서비스와 기능을 출시함에 따라 워크로드에 맞는 최적의 서비스와 아키텍처가 변경될 수 있습니다.

워크로드 활동을 지속적으로 모니터링하고 개별 구성 요소의 활용률 수준을 개선할 수 있는 기회를 찾아보세요. 유휴 상태인 구성 요소를 제거하고 적절한 크기 조정 작업을 수행하면 클라우드 리소스를 최소화하여 비즈니스 요구 사항을 충족할 수 있습니다.

구현 단계

- 워크로드의 중요한 구성 요소(예: Amazon CloudWatch 지표의 CPU 활용률, 메모리 활용률 또는 네트워크 처리량)에 대한 활용률 지표를 모니터링하고 캡처합니다.
- 안정적인 워크로드를 위해 정기적으로 AWS 적정 크기 조정 도구(예: AWS Compute Optimizer)를 확인하여 유휴 상태이거나, 사용되지 않거나, 활용도가 낮은 구성 요소를 식별합니다.
- 일시적인 워크로드의 경우 활용률 지표를 평가하여 유휴 상태이거나, 사용되지 않거나, 활용도가 낮은 구성 요소를 식별합니다.
- 더 이상 필요하지 않은 구성 요소 및 관련 자산(예: Amazon ECR 이미지)을 사용 중지합니다.
- 사용률이 낮은 구성 요소를 리팩터링하거나 다른 리소스와 통합하여 사용 효율성을 개선합니다. 예를 들어, 사용률이 낮은 개별 인스턴스에서 데이터베이스를 실행하는 대신 단일 Amazon RDS 데이터베이스 인스턴스에 여러 개의 소규모 데이터베이스를 프로비저닝할 수 있습니다.
- 작업 단위를 완료하기 위해 워크로드에서 프로비저닝한 리소스를 파악합니다.

리소스

관련 문서:

- AWS Trusted Advisor
- What is Amazon CloudWatch?(CloudWatch란 무엇인가요?)
- Amazon ECR에서 사용되지 않는 이미지 자동 정리

관련 예시:

- Well-Architected 실습 - AWS Compute Optimizer로 적정 크기 조정
- Well-Architected 실습 - 하드웨어 패턴 최적화 및 지속 가능성 KPI 관찰

SUS03-BP03 가장 많은 시간 또는 리소스를 소모하는 코드 영역 최적화

아키텍처의 여러 구성 요소 내에서 실행되는 코드를 최적화하여 리소스 사용을 최소화하고 성능을 극대화할 수 있습니다.

일반적인 안티 패턴:

- 리소스 사용에 대한 코드 최적화를 무시합니다.
- 일반적으로 리소스를 늘리는 방법으로 성능 문제에 대응합니다.
- 코드 검토 및 개발 프로세스에서 성능 변경을 추적하지 않습니다.

이 모범 사례 확립의 이점: 효율적인 코드를 활용하면 리소스 사용을 최소화하고 성능을 향상할 수 있습니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 보통

구현 가이드

리소스 사용 및 성능을 최적화하려면 클라우드 아키텍처 기반 애플리케이션의 코드를 포함한 모든 기능 영역을 검사해야 합니다. 빌드 환경 및 프로덕션에서 워크로드의 성능을 지속적으로 모니터링하고 리소스 사용량이 특히 높은 코드 스니펫을 개선할 기회를 식별합니다. 코드 내에서 리소스를 비효율적으로 사용하는 버그 또는 안티 패턴을 식별하기 위해 정기적인 검토 프로세스를 채택합니다. 사용 사례에 대해 동일한 결과를 생성하는 간단하고 효율적인 알고리즘을 활용합니다.

구현 단계

- 워크로드를 개발하는 동안 자동화된 코드 검토 프로세스를 채택하여 품질을 개선하고 버그와 안티 패턴을 식별합니다.
  - Amazon CodeGuru Reviewer를 사용한 코드 검토 자동화
  - Amazon CodeGuru를 통한 동시성 버그 탐지
  - Amazon CodeGuru를 사용한 Python 애플리케이션의 코드 품질 향상
- 워크로드를 실행할 때 리소스를 모니터링하여 작업 단위당 리소스 요구 사항이 높은 구성 요소를 코드 검토 대상으로 구분합니다.
- 코드 검토의 경우 코드 프로파일러를 사용하여 가장 많은 시간 또는 리소스를 사용하는 코드 영역을 최적화 대상으로 파악합니다.
  - Amazon CodeGuru Profiler를 사용한 조직의 탄소 배출량 감소
  - Amazon CodeGuru Profiler를 통한 Java 애플리케이션 메모리 사용량 이해

  - Amazon CodeGuru Profiler를 사용한 고객 경험 개선 및 비용 절감
- 워크로드에 가장 효율적인 운영 체제 및 프로그래밍 언어를 사용합니다. 에너지 효율적인 프로그래밍 언어(Rust 포함)에 대한 자세한 내용은 Rust를 통한 지속 가능성을참조하십시오.
- 컴퓨팅 집약적인 알고리즘을 동일한 결과를 내는 보다 단순하고 효율적인 버전으로 대체합니다.
- 정렬 및 서식 지정과 같은 불필요한 코드를 제거합니다.

리소스

관련 문서:

- Amazon CodeGuru Profiler란 무엇인가요?
- FPGA 인스턴스
- AWS에서의 구축을 위한 도구의 AWS SDK

관련 동영상:

- Improve Code Efficiency Using Amazon CodeGuru Profiler(Amazon CodeGuru Profiler를 사용하여 코드 효율성 향상)
- Automate Code Reviews and Application Performance Recommendations with Amazon CodeGuru(Amazon CodeGuru를 사용하여 코드 검토 및 애플리케이션 성능 권장 사항 자동화)

SUS03-BP04 디바이스 및 장비에 대한 영향 최적화

아키텍처에 사용되는 디바이스와 장비를 이해하고 전략을 바탕으로 사용량을 줄입니다. 이를 통해 클라우드 워크로드의 전반적인 환경 영향을 최소화할 수 있습니다.

일반적인 안티 패턴:

- 고객이 사용하는 디바이스가 환경에 미치는 영향을 무시합니다.
- 고객이 사용하는 리소스를 수동으로 관리하고 업데이트합니다.

이 모범 사례 확립의 이점: 고객 디바이스에 최적화된 소프트웨어 패턴 및 기능을 구현하면 클라우드 워크로드의 전반적인 환경 영향을 줄일 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

고객 디바이스에 최적화된 소프트웨어 패턴 및 기능을 구현하면 다음과 같은 여러 가지 방법으로 환경에 미치는 영향을 줄일 수 있습니다.

- 이전 버전과 호환되는 새 기능을 구현하면 하드웨어 교체 횟수를 줄일 수 있습니다.
- 디바이스에서 효율적으로 실행되도록 애플리케이션을 최적화하면 에너지 소비를 줄이고 배터리 수명을 연장할 수 있습니다(배터리로 작동하는 경우).
- 디바이스의 애플리케이션을 최적화하면 네트워크를 통한 데이터 전송도 줄일 수 있습니다.

아키텍처에 사용되는 디바이스와 장비, 예상 수명 주기 및 이러한 구성 요소 교체의 영향을 이해합니다. 디바이스 에너지 소비와 고객이 디바이스를 교체해야 하는 필요성, 수동 업그레이드를 최소화하는 소프트웨어 패턴과 기능을 구현합니다.

구현 단계

- 아키텍처에 사용되는 디바이스의 인벤토리를 구성합니다. 디바이스는 모바일, 태블릿, IOT 디바이스, 스마트 라이트 또는 공장의 스마트 디바이스일 수 있습니다.
- 디바이스에서 실행 중인 애플리케이션을 최적화합니다.
  - 백그라운드에서 태스크를 실행하는 것과 같은 전략을 사용하여 에너지 소비를 줄입니다.
  - 페이로드를 구축할 때 네트워크 대역폭과 지연 시간을 고려하고 애플리케이션이 지연 시간이 긴 저대역폭 링크에서 잘 작동하도록 지원하는 기능을 구현합니다.
  - 페이로드 및 파일을 디바이스에 필요한 최적화된 형식으로 변환합니다. 예를 들어, Amazon Elastic Transcoder 또는 AWS Elemental MediaConvert를 사용하여 대용량의 고품질 디지털 미디어 파일을 사용자가 모바일 디바이스, 태블릿, 웹 브라우저 및 연결된 TV에서 재생할 수 있는 형식으로 변환합니다.
  - 서버 측에서 계산 집약적인 활동(예: 이미지 렌더링)을 수행하거나 애플리케이션 스트리밍을 사용하여 구형 디바이스에서 사용자 경험을 개선합니다.
  - 페이로드를 관리하고 로컬 스토리지 요구 사항을 제한하기 위해 특히 대화형 세션의 경우 출력을 분할하고 페이지 번호를 매깁니다.
- 자동화된 무선 업데이트(OTA) 메커니즘을 사용하여 하나 이상의 디바이스에 업데이트를 배포합니다.
  - CI/CD 파이프라인을 사용하여 모바일 애플리케이션을 업데이트할 수 있습니다.
  - AWS IoT Device Management를 사용하여 연결된 디바이스를 규모에 맞게 원격으로 관리할 수 있습니다.

- 새로운 기능 및 업데이트를 테스트하려면 대표적인 하드웨어 집합과 함께 관리형 Device Farm을 사용하고 개발을 반복하여 지원되는 디바이스를 최대화하세요. 자세한 내용은 SUS06-BP04 테스트에 관리형 Device Farm 사용 페이지를 참조하세요.

리소스

관련 문서:

- AWS Device Farm이란 무엇인가요?
- Amazon AppStream 2.0 설명서
- NICE DCV
- OTA tutorial for updating firmware on devices running FreeRTOS(FreeRTOS를 실행하는 디바이스에서 펌웨어를 업데이트하기 위한 OTA 튜토리얼)

관련 동영상:

- Introduction to AWS Device Farm(AWS Device Farm 소개)

SUS03-BP05 데이터 액세스 및 스토리지 패턴을 가장 잘 지원하는 소프트웨어 패턴 및 아키텍처 사용

데이터가 워크로드 내에서 사용되고, 사용자가 소비하고, 전송 및 저장되는 방식을 이해합니다. 데이터 액세스 및 스토리지를 가장 잘 지원하는 소프트웨어 패턴 및 아키텍처를 사용하여 워크로드를 지원하는 데 필요한 컴퓨팅, 네트워킹 및 스토리지 리소스를 최소화합니다.

일반적인 안티 패턴:

- 모든 워크로드의 데이터 스토리지 및 액세스 패턴이 비슷하다고 가정합니다.
- 모든 워크로드가 해당 계층 내에서 적합하다고 가정하고 하나의 스토리지 계층만 사용합니다.
- 시간이 지나면 데이터 액세스 패턴이 일관되게 유지될 것이라고 가정합니다.
- 아키텍처는 높을 가능성이 있는 데이터 액세스 버스트를 지원하므로, 리소스가 대부분 유휴 상태로 유지됩니다.

이 모범 사례 확립의 이점: 데이터 액세스 및 스토리지 패턴을 기반으로 아키텍처를 선택하고 최적화하면 개발 복잡성을 줄이고 전체 활용률을 높일 수 있습니다. 글로벌 테이블, 데이터 파티셔닝 및 캐싱을 사용해야 하는 시기를 파악하면 워크로드 요구 사항에 따라 운영 오버헤드를 줄이고 규모를 확장할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

데이터 특성 및 액세스 패턴에 가장 적합한 소프트웨어 및 아키텍처 패턴을 사용합니다. 예를 들어, AWS 기반 최신 데이터 아키텍처를 통해 고유한 분석 사용 사례에 최적화된 목적별 서비스를 사용할 수 있습니다. 이러한 아키텍처 패턴은 효율적인 데이터 처리를 가능하게 하고 리소스 사용을 줄여줍니다.

구현 단계

- 데이터 특성 및 액세스 패턴을 분석하여 클라우드 리소스에 적합한 구성을 식별합니다. 고려해야 할 주요 특성은 다음과 같습니다.
  - 데이터 형식: 정형, 반정형 및 비정형
  - 데이터 증가: 제한, 무제한
  - 데이터 내구성: 영구, 임시, 일시적
  - 액세스 패턴(읽기 또는 쓰기, 업데이트 빈도, 급증 또는 일관성 여부)
- 데이터 액세스 및 스토리지 패턴을 가장 잘 지원하는 아키텍처 패턴을 사용합니다.
  - 아키텍트를 시작하겠습니다! 현대적 데이터 아키텍처
  - Databases on AWS: The Right Tool for the Right Job(AWS 기반 데이터베이스: 작업에 맞는 적절한 도구)
- 기본적으로 압축된 데이터와 함께 작동하는 기술을 사용합니다.
- 아키텍처의 데이터 처리에 목적별 분석 서비스를 사용합니다.
- 가장 많이 나타나는 쿼리 패턴을 가장 효과적으로 지원하는 데이터베이스 엔진을 사용합니다. 데이터베이스 인덱스를 관리하여 효율적인 쿼리 실행을 보장합니다. 자세한 내용은 AWS 데이터베이스를 참조하세요.
- 아키텍처에 사용되는 네트워크 용량을 줄이는 네트워크 프로토콜을 선택합니다.

리소스

관련 문서:

- Athena 압축 지원 파일 형식
- Amazon Redshift를 사용한 열 기반 데이터 형식에서 COPY 명령
- Converting Your Input Record Format in Firehose(Kinesis Data Firehose Firehose에서 입력 레코드 형식 변환)

- AWS Glue에서 ETL 입력 및 출력의 포맷 옵션
- 열 형식으로 변환하여 Amazon Athena에서 쿼리 성능 향상
- Amazon Redshift를 사용하여 Amazon S3에서 압축된 데이터 파일 로드
- Monitoring DB load with Performance Insights on Amazon Aurora(Amazon Aurora의 성능 개선 도우미로 DB 로드 모니터링)
- Monitoring DB load with Performance Insights on Amazon RDS(Amazon RDS의 성능 개선 도우미로 DB 로드 모니터링)
- Amazon S3 Intelligent-Tiering storage class(Amazon S3 Intelligent-Tiering 스토리지 클래스)

관련 동영상:

- Building modern data architectures on AWS(AWS에서 현대적 데이터 아키텍처 구축)

데이터

질문

- SUS 4 데이터 관리 정책 및 패턴을 활용하여 지속 가능성 목표를 지원하려면 어떻게 해야 합니까?

SUS 4 데이터 관리 정책 및 패턴을 활용하여 지속 가능성 목표를 지원하려면 어떻게 해야 합니까?

데이터 관리 원칙을 구현하여 워크로드를 지원하는 데 필요한 프로비저닝된 스토리지와 이를 사용하는 데 필요한 리소스를 줄입니다. 데이터를 이해하고 데이터의 비즈니스 가치와 데이터 사용 방식을 가장 잘 지원하는 스토리지 기술과 구성을 사용합니다. 요구 사항이 감소하면 데이터를 더 효율적이고 성능이 낮은 스토리지로 수명 주기를 변경하고 더 이상 필요하지 않은 데이터는 삭제합니다.

모범 사례

- SUS04-BP01 데이터 분류 정책 구현
- SUS04-BP02 데이터 액세스 및 스토리지 패턴을 지원하는 기술 사용
- SUS04-BP03 정책을 사용하여 데이터 세트의 수명 주기 관리
- SUS04-BP04 탄력성 및 자동화 기능을 사용하여 블록 스토리지 또는 파일 시스템 확장
- SUS04-BP05 불필요하거나 중복된 데이터 제거
- SUS04-BP06 공유 파일 시스템 또는 스토리지를 사용하여 공용 데이터에 액세스
- SUS04-BP07 네트워크 간 데이터 이동 최소화

- SUS04-BP08 다시 생성하기 어려운 경우에만 데이터 백업

SUS04-BP01 데이터 분류 정책 구현

데이터를 분류하여 비즈니스 성과에 대한 중요도를 파악하고 데이터를 저장할 에너지 효율적인 적절한 스토리지 티어를 선택합니다.

일반적인 안티 패턴:

- 처리 중이거나 저장된 데이터 자산 중 특성(예: 민감도, 비즈니스 중요도 또는 규제 요구 사항)이 유사한 데이터 자산을 식별하지 않습니다.
- 데이터 자산의 인벤토리 등록을 위한 데이터 카탈로그를 구현하지 않았습니다.

이 모범 사례 확립의 이점: 데이터 분류 정책을 구현하면 가장 에너지 효율성이 높은 데이터 스토리지 티어를 확인할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

데이터 분류에는 처리 중인 데이터 유형과 조직에서 소유하거나 운영하는 정보 시스템에 저장되는 데이터 유형을 식별하는 작업이 포함됩니다. 또한 데이터의 중요도와 데이터 손상, 손실 또는 남용의 영향을 확인하는 작업도 포함됩니다.

데이터의 상황별 사용에서부터 시작하여 거꾸로 작업하고 조직 운영에 대한 제공된 데이터 세트의 중요도 수준을 고려하는 분류 체계를 만들어 데이터 분류 정책을 구현합니다.

구현 단계

- 워크로드에 존재하는 다양한 데이터 유형의 인벤토리를 수행합니다.
  - 데이터 분류 범주에 대한 자세한 내용은 데이터 분류 백서를 참조하세요.
- 조직에 대한 위험을 기준으로 데이터의 중요도, 기밀성, 무결성 및 가용성을 확인합니다. 이러한 요구 사항을 확인하여 채택한 데이터 분류 티어 중 하나로 데이터를 그룹화합니다.
  - 예시는 Four simple steps to classify your data and secure your startup(데이터를 분류하고 스타트업을 보호하기 위한 네 가지 간단한 단계)을 참조하세요.
- 환경에 태그가 지정되지 않은 데이터와 분류되지 않은 데이터가 있는지 주기적으로 감사하고 데이터를 적절하게 분류하고 태그를 지정합니다.
  - 예시는 AWS Glue의 데이터 카탈로그 및 크롤러를 참조하세요.

- 감사 및 거버넌스 기능을 제공하는 데이터 카탈로그를 설정합니다.
- 각 데이터 클래스의 처리 절차를 결정하여 문서화합니다.
- 태그가 지정되지 않은 데이터와 분류되지 않은 데이터를 식별하여 데이터를 적절하게 분류하고 태그를 지정하기 위해 자동화를 사용하여 환경을 지속적으로 감사합니다.

리소스

관련 문서:

- Leveraging AWS 클라우드 to Support Data Classification(AWS 클라우드를 활용하여 데이터 분류 지원)
- AWS Organizations의 태그 정책

관련 동영상:

- Enabling agility with data governance on AWS(AWS에 대한 데이터 거버넌스를 사용하여 민첩성 지원)

SUS04-BP02 데이터 액세스 및 스토리지 패턴을 지원하는 기술 사용

데이터 액세스 및 저장 방법을 가장 잘 지원하는 스토리지 기술을 사용하여 워크로드를 지원하면서 프로비저닝된 리소스를 최소화합니다.

일반적인 안티 패턴:

- 모든 워크로드의 데이터 스토리지 및 액세스 패턴이 비슷하다고 가정합니다.
- 모든 워크로드가 해당 계층 내에서 적합하다고 가정하고 하나의 스토리지 계층만 사용합니다.
- 시간이 지나면 데이터 액세스 패턴이 일관되게 유지될 것이라고 가정합니다.

이 모범 사례 확립의 이점: 데이터 액세스 및 스토리지 패턴을 기반으로 스토리지 기술을 선택하고 최적화하면 비즈니스 요구 사항을 충족하는 데 필요한 클라우드 리소스를 줄이고 클라우드 워크로드의 전반적인 효율성을 향상시킬 수 있습니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 낮음

구현 가이드

따라서 성능 효율성을 극대화하려면 액세스 패턴에 가장 적합한 스토리지 솔루션을 선택하거나, 스토리지 솔루션에 따라 액세스 패턴을 변경하는 것이 좋습니다.

- 데이터 특성 및 액세스 패턴을 평가하여 스토리지 요구 사항의 주요 특성을 수집합니다. 고려해야 할 주요 특성은 다음과 같습니다.
  - 데이터 유형: 정형, 반정형 및 비정형
  - 데이터 증가: 제한, 무제한
  - 데이터 내구성: 영구, 임시, 일시적
  - 액세스 패턴: 읽기 또는 쓰기, 빈도, 급증하는지 또는 일관적인지
- 데이터 특성 및 액세스 패턴을 지원하는 적절한 스토리지 기술로 데이터를 마이그레이션합니다. AWS 스토리지 기술의 몇 가지 예와 주요 특성은 다음과 같습니다.

유형	기술	주요 특징
객체 스토리지	Amazon S3	무제한 확장성, 높은 가용성과 액세스 가능성을 위한 여러 옵션을 갖춘 객체 스토리지 서비스입니다. Amazon S3 안팎에서 객체 전송 및 액세스는 Transfer Acceleration 또는 액세스 포인트등과 같은 서비스를 사용하여 위치, 보안 요구, 액세스 패턴을 지원할 수 있습니다.
아카이빙 스토리지	Amazon S3 Glacier	데이터 아카이빙을 위해 구축된 Amazon S3의 스토리지 클래스입니다.
공유 파일 시스템	Amazon Elastic File System(Amazon EFS)	여러 유형의 컴퓨팅 솔루션에서 액세스할 수 있는 탑재 가능한 파일 시스템입니다. Amazon EFS는 스토리지를 자동으로 늘리고 줄이며 성능

유형	기술	주요 특징
		이 최적화되어 일관되게 지연 시간이 짧습니다.
공유 파일 시스템	Amazon FSx	최신 AWS 컴퓨팅 솔루션을 기반으로 구축되어 일반적으로 사용되는 네 가지 파일 시스템인 NetApp ONTAP, OpenZFS, Windows 파일 서버, Lustre를 지원합니다. Amazon FSx 지연 시간, 처리량, IOPS는 파일 시스템에 따라 달라지며 워크로드의 요구 사항에 적합한 파일 시스템을 선택할 때 고려해야 합니다.
블록 스토리지	Amazon Elastic Block Store(Amazon EBS)	Amazon Elastic Compute Cloud(Amazon EC2)를 위해 설계된 확장 가능한 고성능 블록 스토리지 서비스입니다. Amazon EBS에는 IOPS 집약적 트랜잭션 워크로드를 위한 SSD 지원 스토리지와 처리량 집약적 워크로드를 위한 HDD 지원 스토리지가 포함됩니다.
관계형 데이터베이스	Amazon Aurora, Amazon RDS, Amazon Redshift	ACID(원자성, 일관성, 격리, 내구성) 트랜잭션을 지원하고 참조 무결성과 강력한 데이터 일관성을 유지하도록 설계되었습니다. 많은 기존 애플리케이션, 엔터프라이즈 리소스 계획(ERP), 고객 관계 관리(CRM), 전자 상거래 시스템의 데이터가 관계형 데이터베이스를 사용하여 저장됩니다.

유형	기술	주요 특징
키-값 데이터베이스	Amazon DynamoDB	대개 대량의 데이터를 저장 및 검색하는 일반적인 접근 패턴에 최적화되어 있습니다. 트래픽이 많은 웹 앱, 전자 상거래 시스템, 게임 애플리케이션은 키 값 데이터베이스의 일반적인 사용 사례입니다.

- 크기가 고정된 스토리지 시스템(예: Amazon EBS 또는 Amazon FSx)의 경우 사용 가능한 스토리지 공간을 모니터링하고 임계값에 도달 시 스토리지 할당을 자동화합니다. Amazon CloudWatch를 활용하여 Amazon EBS 및 Amazon FSx에대한 다양한 지표를 수집 및 분석할 수 있습니다.
- Amazon S3 스토리지 클래스는 객체 수준에서 구성할 수 있으며 단일 버킷에는 모든 스토리지 클래스 전체에 걸쳐 저장된 객체를 포함할 수 있습니다.
- 또한 Amazon S3 수명 주기 정책을 사용하여 애플리케이션 변경 없이 스토리지 클래스 간에 객체를 자동으로 전환하거나 데이터를 제거할 수 있습니다. 이러한 스토리지 메커니즘을 고려할 때 리소스 효율성, 액세스 지연 시간 및 신뢰성 간에 절충해야 합니다.

리소스

관련 문서:

- Amazon EBS 볼륨 유형
- Amazon EC2 인스턴스 스토어
- Amazon S3 Intelligent-Tiering
- Amazon EBS I/O 특성
- Amazon S3 스토리지 클래스 사용
- Amazon S3 Glacier란 무엇인가요?

관련 동영상:

- AWS의 데이터 레이크용 아키텍처 패턴
- Amazon EBS 심층 분석(STG303-R1)
- Amazon S3를 사용하여 스토리지 성능 최적화(STG343)

- AWS에서 현대적 데이터 아키텍처 구축

관련 예시:

- Amazon EFS CSI 드라이버
- Amazon EBS CSI 드라이버
- Amazon EFS 유틸리티
- Amazon EBS 오토 스케일링
- Amazon S3 예시

SUS04-BP03 정책을 사용하여 데이터 세트의 수명 주기 관리

모든 데이터의 수명 주기를 관리하고 삭제를 자동으로 적용하여 워크로드에 필요한 총 스토리지를 최소화합니다.

일반적인 안티 패턴:

- 데이터를 수동으로 삭제합니다.
- 워크로드 데이터를 삭제하지 않습니다.
- 보존 및 액세스 요구 사항에 따라 데이터를 더 에너지 효율적인 스토리지로 이동하지 않습니다.

이 모범 사례 확립의 이점: 데이터 수명 주기 정책을 사용하여 워크로드에서 데이터에 효율적으로 액세스하고 보존할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

데이터 세트는 일반적으로 수명 주기 동안 보존 및 액세스 요구 사항이 각각 다릅니다. 예를 들어, 애플리케이션이 한정된 기간 동안 일부 데이터 세트에 자주 액세스해야 할 수 있습니다. 그 후 해당 데이터 세트에 자주 액세스하지 않습니다.

전체 수명 주기 동안 데이터 세트를 효율적으로 관리하려면 수명 주기 정책을 구성해야 하며, 이것은 데이터 세트를 처리하는 방법을 정의하는 규칙입니다.

수명 주기 구성 규칙을 통해, 데이터 세트를 보다 에너지 효율적인 스토리지 계층으로 전환하고 아카이브하거나 삭제하도록 특정 스토리지 서비스에 요청할 수 있습니다.

구현 단계

- 워크로드의 데이터 세트를 분류합니다.
- 각 데이터 클래스의 처리 절차를 정의합니다.
- 수명 주기 규칙을 적용하는 자동화된 수명 주기 정책을 설정합니다. 다양한 AWS 스토리지 서비스에 대한 자동화된 수명 주기 정책을 설정하는 몇 가지 예는 다음과 같습니다.

Storage service	How to set automated lifecycle policies
Amazon S3	Amazon S3 수명 주기를 사용하여 전체 수명 주기 동안 객체를 관리할 수 있습니다. 액세스 패턴을 알 수 없거나 변화하거나 예측할 수 없는 경우 Amazon S3 Intelligent-Tiering을 사용할 수 있으며, 이를 통해 액세스 패턴을 모니터링하고, 액세스하지 않은 객체를 더 저렴한 액세스 계층으로 자동으로 이동할 수 있습니다. Amazon S3 스토리지 렌즈 지표를 활용하여 수명 주기 관리의 최적화 기회와 격차를 파악할 수 있습니다.
Amazon Elastic Block Store	Amazon Data Lifecycle Manager를 사용하여 Amazon EBS 스냅샷 및 Amazon EBS 지원 AMI의 생성, 보존 및 삭제를 자동화할 수 있습니다.
Amazon Elastic File System	Amazon EFS 수명 주기 관리는 파일 시스템의 파일 스토리지를 자동으로 관리합니다.
Amazon Elastic Container Registry	Amazon ECR 수명 주기 정책은 수명 또는 개수를 기준으로 만료되는 이미지에 의한 컨테이너 이미지 정리를 자동화합니다.
AWS Elemental MediaStore	MediaStore 컨테이너에 객체를 얼마 동안 저장해야 하는지 제어하는 객체 수명 주기 정책을 사용할 수 있습니다.

- 사용되지 않은 볼륨, 스냅샷 및 보존 기간이 지난 데이터를 삭제합니다. 삭제할 Amazon DynamoDB Time To Live 또는 Amazon CloudWatch 로그 보존과 같은 네이티브 서비스 기능을 활용합니다.

- 수명 주기 규칙에 따라 관련 데이터를 집계 및 압축합니다.

리소스

관련 문서:

- Amazon S3 스토리지 클래스 분석을 통해 Amazon S3 수명 주기 규칙 최적화
- AWS Config 규칙으로 리소스 평가

관련 동영상:

- Amazon S3 수명 주기로 데이터 수명 주기 간소화 및 스토리지 비용 최적화
- Amazon S3 스토리지 렌즈를 사용한 스토리지 비용 절감

SUS04-BP04 탄력성 및 자동화 기능을 사용하여 블록 스토리지 또는 파일 시스템 확장

데이터가 늘어나면서 블록 스토리지 또는 파일 시스템을 확장하여 프로비저닝된 총 스토리지를 최소화하는 탄력성 및 자동화 기능을 사용합니다.

일반적인 안티 패턴:

- 향후 필요에 따라 대규모 블록 스토리지 또는 파일 시스템을 조달합니다.
- 파일 시스템의 초당 입출력 작업 처리량(IOPS)을 초과 프로비저닝합니다.
- 데이터 볼륨의 사용률을 모니터링하지 않습니다.

이 모범 사례 확립의 이점: 스토리지 시스템에 대한 초과 프로비저닝을 최소화하면 유휴 리소스가 줄어들고 워크로드의 전반적인 효율성이 향상됩니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

워크로드에 적합한 크기 할당, 처리량 및 지연 시간으로 블록 스토리지 및 파일 시스템을 생성합니다. 이러한 스토리지 서비스를 과도하게 프로비저닝하지 않고도 데이터 증가에 따라 블록 스토리지 또는 파일 시스템을 확장할 수 있는 탄력성 및 자동화 기능을 사용합니다.

구현 단계

- Amazon EBS와 같이 크기가 고정된 스토리지의 경우 전체 스토리지 크기 대비 사용된 스토리지의 양을 모니터링해야 하며, 가능하다면 임계값에 도달할 때 스토리지 크기를 늘릴 수 있는 자동화 기능을 생성해야 합니다.
- 탄력적 볼륨 및 관리형 블록 데이터 서비스를 사용하여 영구 데이터의 증가에 따른 추가 스토리지 할당을 자동화합니다. 예를 들어, Amazon EBS 탄력적 볼륨을 사용하면 볼륨 크기와 볼륨 유형을 변경하거나 Amazon EBS 볼륨의 성능을 조정할 수 있습니다.
- 파일 시스템에 적합한 스토리지 클래스, 성능 모드 및 처리량 모드를 선택하여 비즈니스 요구 사항을 충족하세요. 초과할 필요는 없습니다.
  - Amazon EFS 성능
  - Amazon EBS volume performance on Linux instances(Linux 인스턴스의 Amazon EBS 볼륨 성능)
- 데이터 볼륨의 목표 사용률 수준을 설정하고 예상 범위를 벗어나는 볼륨 크기를 조정합니다.
- 데이터에 적합하도록 읽기 전용 볼륨의 크기를 알맞게 조정합니다.
- 블록 스토리지의 고정 볼륨 크기로 초과 용량을 프로비저닝하지 않도록 데이터를 객체 스토어로 마이그레이션합니다.
- 탄력적인 볼륨 및 파일 시스템을 정기적으로 검토하여 유휴 볼륨을 종료하고 과도하게 프로비저닝된 리소스를 현재 데이터 크기에 맞게 축소합니다.

리소스

관련 문서:

- Amazon FSx 설명서
- What is Amazon Elastic File System?(Amazon Elastic File System이란 무엇일까요?)

관련 동영상:

- Deep Dive on Amazon EBS Elastic Volumes(Amazon EBS 탄력적 볼륨 심층 분석)
- Amazon EBS and Snapshot Optimization Strategies for Better Performance and Cost Savings(성능 향상 및 비용 절감을 위한 Amazon EBS 및 스냅샷 최적화 전략)
- Optimizing Amazon EFS for cost and performance, using best practices(모범 사례를 사용하여 Amazon EFS의 비용 및 성능 최적화)

SUS04-BP05 불필요하거나 중복된 데이터 제거

불필요하거나 중복된 데이터를 제거하여 데이터 세트를 저장하는 데 필요한 스토리지 리소스를 최소화합니다.

일반적인 안티 패턴:

- 쉽게 얻을 수 있거나 다시 생성할 수 있는 데이터를 중복합니다.
- 데이터의 중요도를 고려하지 않고 모든 데이터를 백업합니다.
- 데이터를 불규칙하게 또는 운영 이벤트에만 삭제하거나 전혀 삭제하지 않습니다.
- 스토리지 서비스의 내구성에 관계없이 데이터를 중복 저장합니다.
- 업무상 타당한 이유 없이 Amazon S3 버전 관리를 활성화합니다.

이 모범 사례 확립의 이점: 불필요한 데이터를 제거하면 워크로드 및 워크로드의 환경 영향에 필요한 스토리지 크기를 줄일 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

필요하지 않은 데이터를 저장하지 않습니다. 불필요한 데이터의 삭제를 자동화합니다. 파일 및 블록 수준에서 데이터 중복을 제거하는 기술을 사용합니다. 서비스의 네이티브 데이터 복제 및 중복성 기능을 활용합니다.

구현 단계

- AWS Data Exchange의 기존 공개 데이터 세트 및 AWS의 개방형 데이터를 사용하여 데이터 저장을 방지할 수 있는지 여부를 평가합니다.
- 블록 및 객체 수준에서 데이터 중복을 제거할 수 있는 메커니즘을 사용합니다. 다음은 AWS의 데이터 중복을 제거하는 방법의 몇 가지 예입니다.

Storage service	Deduplication mechanism
Amazon S3	새로운 FindMatches ML Transform을 사용하여 데이터 세트 전체(식별자가 없는 것 포함)에서 일치하는 레코드를 찾으려면 AWS Lake Formation FindMatches를 사용합니다.

Storage service	Deduplication mechanism
Amazon FSx	Amazon FSx for Windows에서 데이터 중복 제거를 활성화합니다.
Amazon Elastic Block Store 스냅샷	스냅샷은 증분식 백업입니다. 즉, 가장 최근 스냅샷 이후에 변경된 디바이스의 블록만 저장됩니다.

- 데이터 액세스를 분석하여 불필요한 데이터를 식별합니다. 수명 주기 정책을 자동화합니다. 삭제할 Amazon DynamoDB Time To Live, Amazon S3 수명 주기 또는 Amazon CloudWatch 로그 보존과 같은 네이티브 서비스 기능을 활용합니다.
- AWS의 데이터 가상화 기능을 사용하여 소스의 데이터를 유지 관리하고 데이터 중복을 방지합니다.
  - AWS의 클라우드 네이티브 데이터 가상화
  - 실습: Amazon Redshift 데이터 공유를 사용하여 데이터 패턴 최적화
- 증분식 백업을 만들 수 있는 백업 기술을 사용합니다.
- Amazon S3의 내구성 및 Amazon EBS의 복제를 활용하여 자체 관리형 기술(예: 독립 디스크의 이중화 어레이(RAID)) 대신 내구성 목표를 달성합니다.
- 로그 및 추적 데이터를 중앙 집중화하고, 동일한 로그 항목을 중복 제거하며, 필요에 따라 세부적으로 조정하는 메커니즘을 설정합니다.
- 적절한 경우에만 캐시를 미리 채웁니다.
- 캐시 모니터링 및 자동화를 설정하여 그에 따라 캐시 크기를 조정합니다.
- 새 버전의 워크로드를 푸시할 때 객체 스토어 및 엣지 캐시에서 오래된 배포 및 자산을 제거합니다.

리소스

관련 문서:

- CloudWatch Logs에서 로그 데이터 보존 변경
- Amazon FSx for Windows File Server에서 데이터 중복 제거
- 데이터 중복 제거를 포함한 Amazon FSx for ONTAP의 기능
- Amazon CloudFront에서의 파일 무효화
- AWS Backup을 사용하여 Amazon EFS 파일 시스템 백업 및 복구
- Amazon CloudWatch Logs란 무엇인가요?

- Amazon RDS의 백업 작업

관련 동영상:

- AWS Lake Formation의 ML 변환을 통한 퍼지 매칭 및 데이터 중복 제거

관련 예시:

- Amazon Athena를 사용하여 Amazon S3 서버 액세스 로그를 분석하려면 어떻게 해야 합니까?

SUS04-BP06 공유 파일 시스템 또는 스토리지를 사용하여 공용 데이터에 액세스

공유 파일 시스템 또는 스토리지를 채택하여 데이터 중복을 방지하고 워크로드를 위한 보다 효율적인 인프라를 지원합니다.

일반적인 안티 패턴:

- 각 개별 클라이언트에 대해 스토리지를 프로비저닝합니다.
- 비활성 클라이언트에서 데이터 볼륨을 분리하지 않습니다.
- 플랫폼 및 시스템 전반에 걸쳐 스토리지에 액세스할 권한을 제공하지 않습니다.

이 모범 사례 확립의 이점: 공유 파일 시스템 또는 스토리지를 사용하면 데이터를 복사하지 않고도 1명 이상의 소비자에게 데이터를 공유할 수 있습니다. 이를 통해 워크로드에 필요한 스토리지 리소스를 줄일 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

여러 사용자 또는 애플리케이션이 동일한 데이터 세트에 액세스하는 경우 워크로드에 효율적인 인프라를 지원하려면 공유 스토리지 기술을 사용해야 합니다. 공유 스토리지 기술은 데이터 세트를 저장 및 관리하고 데이터 중복을 방지할 수 있는 중앙 위치를 제공합니다. 또한 여러 시스템에 걸쳐 데이터의 일관성을 적용합니다. 나아가 공유 스토리지 기술을 사용하면 여러 컴퓨팅 리소스가 동시에 데이터에 액세스하고 이를 처리할 수 있으므로 컴퓨팅 성능을 보다 효율적으로 사용할 수 있습니다.

이러한 공유 스토리지 서비스에서 필요한 경우에만 데이터를 가져오고 사용하지 않는 볼륨을 분리하여 리소스를 확보하세요.

구현 단계

- 데이터의 소비자가 다수인 경우 데이터를 공유 스토리지로 마이그레이션합니다. AWS 기반 공유 스토리지 기술의 몇 가지 예는 다음과 같습니다.

Storage option	When to use
Amazon EBS 다중 연결	Amazon EBS 다중 연결을 사용하면 하나의 프로비저닝된 IOPS SSD(io1 또는 io2) 볼륨을 동일한 가용 영역에 있는 여러 인스턴스에 연결할 수 있습니다.
Amazon EFS	When to Choose Amazon EFS(Amazon EFS 선택 시점)를 참조하세요.
Amazon FSx	Amazon FSx 파일 시스템 선택을 참조하세요.
Amazon S3	파일 시스템 구조가 필요하지 않고 객체 스토리지와 함께 작동하도록 설계된 애플리케이션은 Amazon S3를 대규모로 확장 가능하고 내구성이 뛰어난 저비용 객체 스토리지 솔루션으로 사용할 수 있습니다.

- 필요한 경우에만 공유 파일 시스템에 데이터를 복사하거나 데이터를 가져옵니다. 예를 들어, Amazon S3를 바탕으로 하는 Amazon FSx for Lustre 파일 시스템을 생성하고 작업을 처리하는 데 필요한 데이터의 하위 집합만 Amazon FSx로 로드할 수 있습니다.
- SUS04-BP03 정책을 사용하여 데이터 세트의 수명 주기 관리에 나와 있는 사용 패턴에 따라 데이터를 삭제합니다.
- 자주 사용하지 않는 클라이언트에서 볼륨을 분리합니다.

리소스

관련 문서:

- Linking your file system to an Amazon S3 bucket(파일 시스템을 S3 버킷에 연결)
- Using Amazon EFS for AWS Lambda in your serverless applications(서버리스 애플리케이션에서 AWS Lambda용 Amazon EFS 사용)
- Amazon EFS Intelligent-Tiering Optimizes Costs for Workloads with Changing Access Patterns(Amazon EFS Intelligent-Tiering을 통해 액세스 패턴 변화에 따른 워크로드 비용 최적화)

- Using Amazon FSx with your on-premises data repository(온프레미스 데이터 리포지토리에 Amazon FSx 사용)

관련 동영상:

- Storage cost optimization with Amazon EFS(Amazon EFS를 사용하여 스토리지 비용 최적화)

SUS04-BP07 네트워크 간 데이터 이동 최소화

공유 파일 시스템 또는 객체 스토리지를 사용하여 공통 데이터에 액세스하고 워크로드의 데이터 이동을 지원하는 데 필요한 총 네트워킹 리소스를 최소화합니다.

일반적인 안티 패턴:

- 데이터 사용자의 위치에 관계없이 모든 데이터를 동일한 AWS 리전에 저장합니다.
- 네트워크를 통해 데이터를 이동하기 전에 데이터 크기와 형식을 최적화하지 않습니다.

이 모범 사례 확립의 이점: 네트워크 간 데이터 이동을 최적화하면 워크로드에 필요한 총 네트워킹 리소스가 줄어들고 환경에 미치는 영향도 줄어듭니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 보통

구현 가이드

조직 전체에서 데이터를 옮기려면 컴퓨팅, 네트워킹 및 스토리지 리소스가 필요합니다. 기술을 사용하여 데이터 이동을 최소화하고 워크로드의 전반적인 효율성을 개선합니다.

구현 단계

- 데이터 또는 사용자와의 근접성을 워크로드 리전 선택 시결정 요소로 고려하십시오.
- 리전별 사용 서비스를 분할하여 해당 리전별 데이터가 사용되는 리전 내에 저장되도록 합니다.
- 네트워크를 통해 데이터를 이동하기 전에 효율적인 파일 형식(예: Parquet 또는 ORC)을 사용하고 데이터를 압축합니다.
- 사용하지 않는 데이터는 옮기지 마십시오. 사용하지 않는 데이터의 이동을 방지하는 데 도움이 되는 몇 가지 예는 다음과 같습니다.
  - 관련 데이터에 대해서만 API 응답을 줄입니다.
  - 상세한 경우 데이터를 집계합니다(레코드 수준 정보는 필요하지 않음).

  - 자세한 내용은 Well-Architected 실습 - Amazon Redshift 데이터 공유를 사용하여 데이터 패턴 최적화를 참조하십시오.
  - 교차 계정 데이터 공유를 AWS Lake Formation에서고려하십시오.
- 워크로드 사용자에게 더 가까운 위치에서 코드를 실행할 수 있는 서비스를 사용합니다.

서비스	사용 시기
Lambda@Edge	객체가 캐시에 없는 경우 실행되는 컴퓨팅 집약적 작업에 사용합니다.
CloudFront 함수	HTTP(s) 요청/응답 조작 등과 같이 단기 실행 함수에 의해 시작될 수 있는 간단한 사용 사례에 사용합니다.
AWS IoT Greengrass	커넥티드 디바이스를 위한 로컬 컴퓨팅, 메시징 및 데이터 캐시를 실행합니다.

리소스

관련 문서:

- Optimizing your AWS Infrastructure for Sustainability, Part III: Networking(지속 가능성을 위한 AWS 인프라 최적화, 파트 III: 네트워킹)
- AWS 글로벌 인프라
- CloudFront 글로벌 엣지 네트워크를 포함한 Amazon CloudFront 주요 기능
- Amazon OpenSearch Service에서 HTTP 요청 압축
- Amazon EMR을 사용한 중간 데이터 압축
- Amazon S3에서 Amazon Redshift로 압축 데이터 파일 로딩
- Amazon CloudFront를 사용하여 압축된 파일 제공

관련 동영상:

- Demystifying data transfer on AWS(AWS에서의 데이터 전송 알아보기)

관련 예시:

- 지속 가능성을 위한 설계 - 네트워크 간 데이터 이동 최소화

SUS04-BP08 다시 생성하기 어려운 경우에만 데이터 백업

비즈니스 가치가 없는 데이터는 백업하지 않으면서 워크로드의 스토리지 리소스 요구 사항을 최소화합니다.

일반적인 안티 패턴:

- 데이터에 대한 백업 전략이 없습니다.
- 쉽게 다시 생성할 수 있는 데이터를 백업합니다.

이 모범 사례 확립의 이점: 중요하지 않은 데이터를 백업하지 않으면 워크로드에 필요한 스토리지 리소스를 줄이고 환경에 미치는 영향도 줄일 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

불필요한 데이터를 백업하지 않으면 비용을 절감하고 워크로드에 사용되는 스토리지 리소스를 줄일 수 있습니다. 비즈니스 가치가 있거나 규정 준수 요구 사항을 충족하는 데 필요한 데이터만 백업합니다. 백업 정책을 검토하고 복구 시나리오에서 가치를 제공하지 않는 임시 스토리지는 제외합니다.

구현 단계

- SUS04-BP01 데이터 분류 정책 구현에 나와 있는 대로 데이터 분류 정책을 구현합니다.
- 데이터 분류 중요도를 바탕으로 Recovery Time Objective(RTO) 및 Recovery Point Objective(RPO) 기반 백업 전략을 설계합니다. 중요하지 않은 데이터는 백업하지 마세요.
  - 쉽게 다시 생성할 수 있는 데이터는 제외합니다.
  - 백업 대상에서 임시 데이터를 제외합니다.
  - 공용 위치에서 데이터를 복원하는 데 필요한 시간이 서비스 수준에 관한 계약(SLA)을 초과하지 않는 한, 데이터의 로컬 사본을 제외합니다.
- 자동화된 솔루션 또는 관리형 서비스를 사용하여 비즈니스 크리티컬 데이터를 백업합니다.
  - AWS Backup은 완전관리형 서비스로, AWS 서비스 전반, 클라우드 및 온프레미스에서 데이터 보호를 쉽게 중앙 집중화하고 자동화할 수 있습니다. AWS Backup을 사용하여 자동화된 백업을 생성하는 방법에 대한 실습 지침은 Well-Architected Labs - Testing Backup and Restore of Data(Well-Architected 실습 - 데이터 백업 및 복원 테스트)를 참조하세요.

- AWS Backup을 사용하여 Amazon EFS용 백업을 자동화하고 백업 비용을 최적화합니다.

리소스

관련 모범 사례:

- REL09-BP01 백업해야 하는 모든 데이터 확인 및 백업 또는 소스에서 데이터 재현
- REL09-BP03 자동으로 데이터 백업 수행
- REL13-BP02 복구 목표 달성을 위해 정의된 복구 전략 사용

관련 문서:

- AWS Backup을 사용하여 Amazon EFS 파일 시스템 백업 및 복구
- Amazon EBS 스냅샷
- Amazon Relational Database Service의 백업 작업
- APN 파트너: 백업을 지원할 수 있는 파트너
- AWS Marketplace: 백업에 사용할 수 있는 제품
- Amazon EFS 백업
- Backing Up Amazon FSx for Windows File Server(Amazon FSx for Windows File Server 백업)
- Amazon ElastiCache (Redis OSS) 백업 및 복구

관련 동영상:

- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS(AWS re:Invent 2021 - AWS를 통한 백업, 재해 복구, 랜섬웨어 보호)
- AWS Backup Demo: Cross-Account and Cross-Region Backup(AWS Backup 데모: 계정 간 및 리전 간 백업)
- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341)(AWS re:Invent 2019: AWS Backup에 대한 심층 분석(Rackspace 소개))

관련 예시:

- Well-Architected Lab - Testing Backup and Restore of Data(Well-Architected 실습 - 데이터 백업 및 복원 테스트)

- Well-Architected Lab - Backup and Restore with Failback for Analytics Workload(Well-Architected 실습 - 분석 워크로드에 대한 백업 및 페일백으로 복원)
- Well-Architected Lab - Disaster Recovery - Backup and Restore(Well-Architected 실습 - 재해 복구 - 백업 및 복원)

하드웨어 및 서비스

질문

- SUS 5 지속 가능성 목표를 지원하기 위해 아키텍처에서 클라우드 하드웨어 및 서비스를 어떻게 선택하고 사용합니까?

SUS 5 지속 가능성 목표를 지원하기 위해 아키텍처에서 클라우드 하드웨어 및 서비스를 어떻게 선택하고 사용합니까?

하드웨어 관리 방식을 변경하여 지속 가능성에 미치는 워크로드의 영향을 줄일 수 있는 기회를 모색합니다. 프로비저닝 및 배포에 필요한 하드웨어의 양을 최소화하고 개별 워크로드에 가장 효율적인 하드웨어 및 서비스를 선택합니다.

모범 사례

- SUS05-BP01 요구 사항을 충족하는 데 필요한 최소한의 하드웨어 사용
- SUS05-BP02 영향이 가장 적은 인스턴스 유형 사용
- SUS05-BP03 관리형 서비스 사용
- SUS05-BP04 하드웨어 기반 컴퓨팅 액셀러레이터의 사용 최적화

SUS05-BP01 요구 사항을 충족하는 데 필요한 최소한의 하드웨어 사용

비즈니스 요구 사항을 효율적으로 충족하기 위해 워크로드에 필요한 최소한의 하드웨어를 사용합니다.

일반적인 안티 패턴:

- 리소스 사용률을 모니터링하지 않습니다.
- 아키텍처의 사용률 수준이 낮은 리소스가 있습니다.
- 크기 조정이 필요한지 여부를 결정하기 위해 정적 하드웨어의 사용률 검토를 수행하지 않습니다.
- 비즈니스 KPI를 기반으로 컴퓨팅 인프라의 하드웨어 사용률 목표를 설정하지 않습니다.

이 모범 사례 확립의 이점: 클라우드 리소스의 크기를 적절하게 조정하면 워크로드의 환경 영향을 줄이고, 비용을 절감하며, 성능 벤치마크를 유지하는 데 도움이 됩니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 보통

구현 가이드

워크로드에 필요한 총 하드웨어 수를 최적으로 선택하여 전반적인 효율성을 개선합니다. AWS 클라우드는 다양한 메커니즘(예: AWS Auto Scaling)을 통해 리소스 수를 동적으로 확장하거나 줄일 수 있는 유연성을 제공하고 수요 변화에 대응합니다. 또한, 최소한의 노력으로 리소스를 수정할 수 있는 API 및 SDK도 제공합니다. 이러한 기능을 사용하여 워크로드 구현을 자주 변경할 수 있습니다. 또한 AWS 도구의 적정 크기 조정 지침을 바탕으로 클라우드 리소스를 효율적으로 운영하고 비즈니스 요구 사항을 충족할 수 있습니다.

구현 단계

- 필요에 가장 적합한 인스턴스 유형을 선택합니다.
  - 워크로드에 적합한 Amazon EC2 인스턴스 유형을 선택하려면 어떻게 해야 하나요?
  - Amazon EC2 플릿에 대한 속성 기반 인스턴스 유형 선택
  - 속성 기반 인스턴스 유형 선택을 사용하여 Auto Scaling 그룹 생성
- 가변 워크로드의 경우 작은 증분 단위로 크기를 조정합니다.
- 여러 컴퓨팅 구매 옵션을 사용하여 인스턴스 유연성, 확장성 및 비용 절감의 균형을 맞춥니다.
  - 온디맨드 인스턴스는 인스턴스 유형, 위치 또는 시간을 유연하게 지정할 수 없는 최신 상태 저장 급증 워크로드에 가장 적합합니다.
  - 스팟 인스턴스는 내결함성 및 유연성이 뛰어난 애플리케이션을 위한 다른 옵션을 보완하는 데 유용한 방법입니다.
  - 안정적인 상태의 워크로드에는 컴퓨팅 절감형 플랜을 활용하여 변화하는 요구 사항(예: AZ, 리전, 인스턴스 제품군 또는 인스턴스 유형)에 유연성을 제공합니다.
- 인스턴스 및 가용 영역 다양성을 활용하여 애플리케이션 가용성을 극대화하고 가능한 경우 초과 용량을 활용할 수 있습니다.
- AWS 도구에서 적정 크기 조정 권장 사항을 사용하여 워크로드를 조정합니다.
  - AWS Compute Optimizer
  - AWS Trusted Advisor
- 자동화가 대체 리소스를 배포하는 동안 일시적으로 용량을 줄일 수 있는 서비스 수준에 관한 계약(SLA)을 협상합니다.

리소스

관련 문서:

- Optimizing your AWS Infrastructure for Sustainability, Part I: Compute(지속 가능성을 위한 AWS 인프라 최적화, 파트 I: 컴퓨팅)
- Amazon EC2 플릿의 Auto Scaling을 위한 속성 기반 인스턴스 유형 선택
- AWS Compute Optimizer 설명서
- Operating Lambda: Performance optimization(Lambda 운영: 성능 최적화)
- Auto Scaling 설명서

관련 동영상:

- Build a cost-, energy-, and resource-efficient compute environment(비용, 에너지, 리소스 효율적인 컴퓨팅 환경 구축)

관련 예시:

- Well-Architected 실습 - AWS Compute Optimizer 및 메모리 사용률을 활성화하여 적정 크기 조정(레벨 200)

SUS05-BP02 영향이 가장 적은 인스턴스 유형 사용

새로운 인스턴스 유형을 지속적으로 모니터링하고 사용하여 에너지 효율 개선을 활용합니다.

일반적인 안티 패턴:

- 인스턴스 패밀리는 하나만 사용합니다.
- x86 인스턴스만 사용합니다.
- Amazon EC2 Auto Scaling 구성에서 인스턴스 유형을 하나만 지정합니다.
- AWS 인스턴스를 설계되지 않은 방식으로 사용합니다(예: 컴퓨팅에 최적화된 인스턴스를 메모리 집약적 워크로드에 사용).
- 새 인스턴스 유형을 정기적으로 평가하지 않습니다.
- AWS 적정 크기 조정 도구(예: AWS Compute Optimizer)의 권장 사항을 확인하지 않습니다.

이 모범 사례 확립의 이점: 적정 크기로 조정된 에너지 효율적인 인스턴스를 사용하면 환경 영향 및 워크로드 비용을 크게 줄일 수 있습니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 보통

구현 가이드

클라우드 워크로드에서 효율적인 인스턴스를 사용하는 것은 리소스 사용량을 줄이고 비용 효율성을 높이는 데 매우 중요합니다. 새로운 인스턴스 유형의 릴리스를 지속적으로 모니터링하고 기계 학습 훈련 및 추론, 동영상 트랜스코딩과 같은 특정 워크로드를 지원하도록 설계된 인스턴스 유형을 포함하여 에너지 효율성 개선의 이점을 활용합니다.

구현 단계

- 워크로드의 환경 영향을 줄일 수 있는 인스턴스 유형을 알아보고 탐색합니다.
  - 최신 AWS의 새로운 소식 을 구독하여 최신 AWS 기술 및 인스턴스 소식을 파악합니다.
  - 다양한 AWS 인스턴스 유형에 대해 알아봅니다.
  - 다음 동영상을 보고 Amazon EC2에서 와트당 에너지 사용 대비 최고 성능을 제공하는 AWS Graviton 기반 인스턴스에 대해 살펴봅니다. re:Invent 2020 - Deep dive on AWS Graviton2 processor-powered Amazon EC2 instances(AWS Graviton2 프로세서 기반 Amazon EC2 인스턴스에 대해 자세히 알아보기) 및 Deep dive into AWS Graviton3 and Amazon EC2 C7g instances(AWS Graviton3 및 Amazon C7g 인스턴스에 대해 자세히 알아보기).
- 워크로드를 영향이 가장 적은 인스턴스 유형으로 전환하도록 계획합니다.
  - 워크로드를 위한 새로운 기능 또는 인스턴스를 평가하기 위한 프로세스를 정의합니다. 클라우드에서 민첩성을 활용하여 새 인스턴스 유형이 워크로드 환경 지속 가능성을 어떻게 개선할 수 있는지 신속하게 테스트합니다. 프록시 지표를 사용하여 작업 단위를 완료하는 데 필요한 리소스를 측정합니다.
  - 가능한 경우 다양한 vCPU 수와 다양한 메모리 용량으로 작동하도록 워크로드를 수정하여 인스턴스 유형 선택의 폭을 극대화합니다.
  - 워크로드의 성능 효율성을 개선하려면 워크로드를 Graviton 기반 인스턴스로 전환할 것을 고려합니다.
    - AWS Graviton Fast Start
    - AWS Graviton 기반 Amazon Elastic Compute Cloud 인스턴스로 워크로드를 전환할 때의 고려 사항
    - AWS Graviton2 for ISVs
  - AWS Graviton 옵션 선택을 AWS 관리형 서비스 사용에 고려하세요.

  - 지속 가능성에 미치는 영향이 가장 적고 비즈니스 요구 사항을 충족하는 인스턴스를 제공하는 리전으로 워크로드를 마이그레이션합니다.
  - 기계 학습 워크로드의 경우 AWS Trainium, AWS Inferentia및 Amazon EC2 DL1과 같이 워크로드에 따라 다른 특별히 구축된 하드웨어의 장점을 활용합니다. Inf2 인스턴스와 같은 AWS Inferentia 인스턴스는 동급 Amazon EC2 인스턴스에 비해 최대 50% 더 우수한 와트당 성능을 제공합니다.
  - 또한 Amazon SageMaker Inference Recommender 를 사용하여 ML 추론 엔드포인트의 크기를 적정하게 조정합니다.
  - 급증하는 워크로드의 경우(추가 용량에 대한 요구 사항이 적은 워크로드) 버스트 가능한 성능 인스턴스를 사용합니다.
  - 상태 비저장 또는 내결함성 워크로드의 경우 Amazon EC2 스팟 인스턴스 를 사용하여 클라우드의 전반적인 활용률을 높이고 사용하지 않는 리소스가 지속 가능성에 미치는 영향을 줄입니다.
- 워크로드 인스턴스를 운영 및 최적화합니다.
  - 임시 워크로드의 경우 인스턴스 Amazon CloudWatch 지표 (예: `CPUUtilization`)를 사용하여 인스턴스가 유휴 상태이거나 사용률이 적은지 파악합니다.
  - 안정적인 워크로드의 경우 AWS 적정 크기 조정 도구(예: AWS Compute Optimizer)를 정기적으로 사용하여 인스턴스를 최적화하고 크기를 적정하게 조정할 기회를 파악합니다.
    - Well-Architected 실습 - 적정 크기 조정 권장 사항
    - Well-Architected 실습 - Compute Optimizer로 적정 크기 조정
    - Well-Architected 실습 - 하드웨어 패턴 최적화 및 지속 가능성 KPI 관찰

리소스

관련 문서:

- Optimizing your AWS Infrastructure for Sustainability, Part I: Compute(지속 가능성을 위한 AWS 인프라 최적화, 파트 I: 컴퓨팅)
- AWS Graviton
- Amazon EC2 DL1
- Amazon EC2 용량 예약 플릿
- Amazon EC2 스팟 플릿
- 함수: Lambda 함수 구성
- Amazon EC2 플릿에 대한 속성 기반 인스턴스 유형 선택
- AWS에서 지속 가능하고 효율적이며 비용 최적화된 애플리케이션 구축

- Contino 지속 가능성 대시보드가 고객의 탄소 발자국 줄이기를 지원하는 방법

관련 동영상:

- Deep dive on AWS Graviton2 processor-powered Amazon EC2 instances(AWS Graviton2 프로세서 기반 Amazon EC2 인스턴스에 대해 자세히 알아보기)
- Deep dive into AWS Graviton3 and Amazon EC2 C7g instances(AWS Graviton3 및 Amazon C7g 인스턴스에 대해 자세히 알아보기)
- Build a cost-, energy-, and resource-efficient compute environment(비용, 에너지, 리소스 효율적인 컴퓨팅 환경 구축)

관련 예시:

- 솔루션: AWS에서 지속 가능성을 위해 딥 러닝 워크로드를 최적화하기 위한 지침
- Well-Architected 실습 - 적정 크기 조정 권장 사항
- Well-Architected 실습 - Compute Optimizer로 적정 크기 조정
- Well-Architected 실습 - 하드웨어 패턴 최적화 및 지속 가능성 KPI 관찰
- Well-Architected Lab - Migrating Services to Graviton(Well-Architected 실습 - Graviton으로 서비스 마이그레이션)

SUS05-BP03 관리형 서비스 사용

관리형 서비스를 사용하여 클라우드에서 보다 효율적으로 운영합니다.

일반적인 안티 패턴:

- 활용률이 낮은 Amazon EC2 인스턴스를 사용하여 애플리케이션을 실행합니다.
- 사내 팀이 혁신이나 단순화에 집중할 시간 없이 워크로드만 관리합니다.
- 관리형 서비스에서 보다 효율적으로 실행할 수 있는 태스크를 위한 기술을 배포하고 유지 관리합니다.

이 모범 사례 확립의 이점:

- 관리형 서비스를 사용하면 새로운 혁신과 효율성을 추진하는 데 도움이 될 수 있는 수백만 명의 고객 인사이트를 보유한 AWS에 책임을 맡길 수 있습니다.

- 관리형 서비스는 멀티 테넌트 컨트롤 플레인으로 인해 많은 사용자에게 서비스의 환경 영향을 분산시킵니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

관리형 서비스는 배포된 하드웨어의 높은 사용률과 지속 가능성 최적화를 유지하는 책임을 AWS로 이전합니다. 또한 관리형 서비스를 사용하면 서비스를 유지 관리해야 하는 운영 및 관리 부담이 줄어들기 때문에 팀이 혁신에 더 많은 시간을 할애하고 집중할 수 있습니다.

워크로드를 검토하여 AWS 관리형 서비스로 대체할 수 있는 구성 요소를 식별합니다. 예를 들어, Amazon RDS, Amazon Redshift, Amazon ElastiCache에서는 관리형 데이터베이스 서비스를 제공합니다. Amazon Athena, Amazon EMR, Amazon OpenSearch Service에서는 관리형 분석 서비스를 제공합니다.

구현 단계

1. 서비스 및 구성 요소에 대한 워크로드 인벤토리를 작성합니다.
2. 관리형 서비스로 대체할 수 있는 구성 요소를 평가하고 식별합니다. 관리형 서비스 사용을 고려할 수 있는 몇 가지 예는 다음과 같습니다.

Task	What to use on AWS
데이터베이스 호스팅	Amazon Elastic Compute Cloud(Amazon EC2)에서 자체 Amazon RDS 인스턴스를 유지 관리하는 대신 관리형 Amazon Relational Database Service(Amazon RDS) 인스턴스를 사용합니다.
컨테이너 워크로드 호스팅	자체 컨테이너 인프라를 구현하는 대신 AWS Fargate를 사용합니다.
웹 앱 호스팅	AWS Amplify 호스팅을 정적 웹 사이트 및 서버 측 렌더링 웹 앱을 위한 완전관리형 CI/CD 및 호스팅 서비스로 사용합니다.

3. 종속성을 식별하고 마이그레이션 플랜을 생성합니다. 그에 따라 런북과 플레이북을 업데이트합니다.

- AWS Application Discovery Service는 애플리케이션 종속성 및 활용에 대한 세부적인 정보를 자동으로 수집하고 제공하여 마이그레이션을 계획할 때 보다 정확한 결정을 내릴 수 있도록 합니다.

4. 관리형 서비스로 마이그레이션하기 전에 서비스를 테스트합니다.
5. 마이그레이션 플랜을 바탕으로 자체 호스팅 서비스를 관리형 서비스로 교체합니다.
6. 마이그레이션이 완료된 후 서비스를 지속적으로 모니터링하여 필요에 따라 조정하고 서비스를 최적화합니다.

리소스

관련 문서:

- AWS 클라우드 제품
- AWS 총 소유 비용(TCO) 계산기
- Amazon DocumentDB
- Amazon Elastic Kubernetes Service(EKS)
- Amazon Managed Streaming for Apache Kafka(Amazon MSK)

관련 동영상:

- Cloud operations at scale with AWS Managed Services(AWS Managed Services를 사용하여 적절한 규모의 클라우드 운영)

SUS05-BP04 하드웨어 기반 컴퓨팅 액셀러레이터의 사용 최적화

가속 컴퓨팅 인스턴스의 사용을 최적화하여 워크로드의 물리적 인프라 요구를 줄입니다.

일반적인 안티 패턴:

- GPU 사용을 모니터링하지 않습니다.
- 특별히 구축된 인스턴스가 더 높은 성능, 더 낮은 비용 및 더 나은 와트당 성능을 제공함에도 불구하고 워크로드에 범용 인스턴스를 사용합니다.
- CPU 기반 대안을 사용하는 것이 더 효율적인 작업에 하드웨어 기반 컴퓨팅 액셀러레이터를 사용합니다.

이 모범 사례 확립의 이점: 하드웨어 기반 액셀러레이터의 사용을 최적화하여 워크로드의 물리적 인프라 요구를 줄일 수 있습니다.

이 모범 사례가 확립되지 않았을 경우의 위험 수준: 보통

구현 가이드

높은 처리 용량이 필요한 경우, 그래픽 처리 장치(GPU) 및 필드 프로그래밍 가능 게이트 어레이(FPGA)와 같은 하드웨어 기반 컴퓨팅 액셀러레이터에 대한 액세스를 제공하는 가속 컴퓨팅 인스턴스 사용의 이점을 활용할 수 있습니다. 이러한 하드웨어 액셀러레이터는 CPU 기반 대안보다 더 효율적인 그래픽 처리 또는 데이터 패턴 일치와 같은 특정 기능을 수행합니다. 렌더링, 트랜스코딩, 기계 학습 등 많은 가속 워크로드는 리소스 사용 면에서 매우 가변적입니다. 필요한 시간 동안만 이 하드웨어를 실행하고 필요하지 않은 경우 자동화를 통해 이를 폐기하여 리소스 사용을 최소화합니다.

구현 단계

- 요구 사항을 해결할 수 있는 가속 컴퓨팅 인스턴스를 식별합니다.
- 기계 학습 워크로드의 경우 AWS Trainium, AWS Inferentia및 Amazon EC2 DL1같이 워크로드에 따라 특별히 구축된 다른 하드웨어의 장점을 활용합니다. Inf2 인스턴스와 같은 AWS Inferentia 인스턴스는 동급 Amazon EC2 인스턴스에 비해 최대 50% 더 우수한 와트당 성능을제공합니다.
- 가속 컴퓨팅 인스턴스의 사용량 지표를 수집합니다. 예를 들어 GPU의 경우 CloudWatch 에이전트를 사용하여 `utilization_gpu` 및 `utilization_memory` 같은 지표를 수집할 수 있습니다(Amazon CloudWatch를 사용하여 NVIDIA GPU 지표 수집참조).
- 코드, 네트워크 운영, 하드웨어 액셀러레이터 설정을 최적화하여 기본 하드웨어가 반드시 제대로 활용되도록 해야 합니다.
  - GPU 설정 최적화
  - 딥 러닝 AMI에서 GPU 모니터링 및 최적화
  - Amazon SageMaker에서 딥 러닝 훈련의 GPU 성능 튜닝을 위한 I/O 최적화
- 최신 고성능 라이브러리 및 GPU 드라이버를 사용합니다.
- 자동화를 사용하여 사용하지 않는 GPU 인스턴스 사용을 해제합니다.

리소스

관련 문서:

- 가속화된 컴퓨팅
- 아키텍트를 시작하겠습니다! 맞춤형 칩과 액셀러레이터를 사용한 아키텍팅
- 워크로드에 적합한 Amazon EC2 인스턴스 유형을 선택하려면 어떻게 해야 하나요?
- Amazon EC2 VT1 인스턴스

- Amazon Elastic Graphics
- Amazon SageMaker을 통해 컴퓨터 비전 추론을 위한 최고의 AI 액셀러레이터 및 모델 컴파일 선택

관련 동영상:

- 딥 러닝을 위한 Amazon EC2 GPU 인스턴스 선택 방법
- Amazon EC2 Elastic GPU 심층 분석
- 비용 효과적인 딥 러닝 추론 배포

프로세스 및 문화

질문

- SUS 6 조직의 프로세스가 지속 가능성 목표를 어떻게 지원하고 있습니까?

SUS 6 조직의 프로세스가 지속 가능성 목표를 어떻게 지원하고 있습니까?

개발, 테스트 및 배포 방식을 변경하여 지속 가능성에 미치는 영향을 줄일 수 있는 기회를 모색합니다.

모범 사례

- SUS06-BP01 지속 가능성 개선을 신속하게 도입할 수 있는 방법 채택
- SUS06-BP02 워크로드를 최신 상태로 유지
- SUS06-BP03 구축 환경의 사용률 제고
- SUS06-BP04 테스트에 관리형 Device Farm 사용

SUS06-BP01 지속 가능성 개선을 신속하게 도입할 수 있는 방법 채택

잠재적인 개선 사항을 검증하고, 테스트 비용을 최소화하며, 경미한 개선 사항을 제공하는 방법과 프로세스를 채택합니다.

일반적인 안티 패턴:

- 지속 가능성을 위한 애플리케이션 검토는 프로젝트 시작 시 1번만 수행합니다.
- 릴리스 프로세스가 너무 번거로워 리소스 효율성을 위해 사소한 변경 사항을 적용할 수 없어 워크로드가 오래되었습니다.
- 지속 가능성을 위한 워크로드를 개선할 수 있는 메커니즘이 없습니다.

이 모범 사례 확립의 이점: 지속 가능성 개선 사항을 도입하고 추적하기 위한 프로세스를 수립함으로써 새로운 기능을 꾸준히 채택하고, 문제를 제거하고, 워크로드 효율성을 개선할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

잠재적인 지속 가능성 개선 사항을 프로덕션에 배포하기 전에 테스트하고 검증합니다. 개선 사항으로 실현될 미래의 잠재적 이익을 계산할 때 테스트 비용을 고려합니다. 적은 비용으로 경미한 개선 사항을 적용할 수 있는 테스트 방법을 개발합니다.

구현 단계

- 지속 가능성 개선 사항을 위한 요구 사항을 개발 백로그에 추가합니다.
- 반복적인 개선 프로세스를 사용하여 이러한 개선 사항을 식별, 평가, 우선순위 지정, 테스트 및 배포합니다.
- 개발 프로세스를 지속적으로 개선하고 간소화합니다. 예를 들어, 지속적 통합(CI) 및 지속적 전달(CD) 파이프라인을 통해 소프트웨어 전송 프로세스를 자동화함으로써 잠재적인 개선 사항을 테스트하고 배포하여 작업량을 줄이고 수동 프로세스로 인한 오류를 제한합니다.
- 테스트 비용을 줄이기 위해 최소 기능 구성 요소를 사용하여 잠재적인 개선 사항을 개발하고 테스트합니다.
- 개선의 영향을 지속적으로 평가하고 필요에 따라 조정합니다.

리소스

관련 문서:

- AWS가 지원하는 지속 가능성 솔루션
- Scalable agile development practices based on AWS CodeCommit(AWS CodeCommit에 기반한 확장 가능한 민첩한 개발 관행)

관련 동영상:

- Delivering sustainable, high-performing architectures(지속 가능한 고성능 아키텍처 제공)

관련 예시:

- Well-Architected Lab - Turning cost & usage reports into efficiency reports(Well-Architected 실습 - 비용 및 사용량 보고서를 효율성 보고서로 전환)

SUS06-BP02 워크로드를 최신 상태로 유지

워크로드를 최신 상태로 유지하여 효율적인 기능을 채택하고, 문제를 제거하며, 워크로드의 전반적인 효율성을 개선합니다.

일반적인 안티 패턴:

- 시간이 지나면 현재 아키텍처가 정적 아키텍처가 되고 업데이트되지 않는다고 가정합니다.
- 업데이트된 소프트웨어 및 패키지가 워크로드와 호환되는지 평가하는 시스템 또는 정기적인 주기가 없습니다.

이 모범 사례 확립의 이점: 워크로드를 최신 상태로 유지하기 위한 프로세스를 확립하면 새로운 기능을 도입하고 문제를 해결하며 워크로드 효율성을 개선할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

최신 운영 체제, 런타임, 미들웨어, 라이브러리 및 애플리케이션을 사용하면 워크로드 효율성을 개선하고 보다 효율적인 기술을 쉽게 채택할 수 있습니다. 공급업체가 자체적인 지속 가능성 목표를 충족할 수 있는 기능을 제공함에 따라, 최신 소프트웨어에는 워크로드의 지속 가능성에 미치는 영향을 보다 정확하게 측정하는 기능이 포함될 수도 있습니다. 정기적인 주기로 최신 기능 및 릴리스와 함께 워크로드를 최신 상태로 유지합니다.

구현 단계

- 워크로드를 위한 새로운 기능 또는 인스턴스를 평가하기 위한 프로세스 및 일정을 정의합니다. 클라우드에서 민첩성을 활용하여 새 기능이 다음 작업을 수행하도록 워크로드를 어떻게 개선할 수 있는지 신속하게 테스트합니다.
  - 지속 가능성 영향 줄이기
  - 성능 효율성을 높이기
  - 계획된 개선 작업의 장애 요인 제거
  - 지속 가능성에 미치는 영향을 측정 및 관리할 수 있는 능력 증진
- 워크로드 소프트웨어 및 아키텍처를 조사하여 업데이트하는 데 필요한 구성 요소를 식별합니다.

  - AWS Systems Manager 인벤토리를 사용하여 Amazon EC2 인스턴스에서 운영 체제(OS), 애플리케이션, 인스턴스 메타데이터를 수집하고 소프트웨어를 실행 중인 인스턴스, 소프트웨어 정책에 필요한 구성, 업데이트해야 할 인스턴스를 신속하게 파악합니다.
- 워크로드 구성 요소의 업데이트 방법을 파악합니다.

Workload component	How to update
머신 이미지	EC2 Image Builder를 사용하여 Linux 또는 Windows 서버 이미지용 Amazon Machine Image(AMI) 업데이트를 관리합니다.
컨테이너 이미지	Amazon Elastic Container Registry (Amazon ECR)를 기존 파이프라인과 함께 사용하여 Amazon Elastic Container Service (Amazon ECS) 이미지를 관리합니다.
AWS Lambda	AWS Lambda에는 버전 관리 기능이 있습니다.

- 업데이트 프로세스에 자동화를 사용하여 새 기능 배포에 필요한 작업 수준을 줄이고 수동 프로세스로 인한 오류를 제한합니다.
  - CI/CD를 사용하면 클라우드 애플리케이션과 관련된 AMI, 컨테이너 이미지 및 기타 아티팩트를 자동으로 업데이트할 수 있습니다.
  - AWS Systems Manager Patch Manager와 같은 도구를 사용하여 시스템 업데이트 프로세스를 자동화하고, AWS Systems Manager Maintenance Windows를 사용하여 활동을 예약할 수 있습니다.

리소스

관련 문서:

- AWS 아키텍처 센터
- AWS의 새로운 소식
- AWS 개발자 도구

관련 예시:

- Well-Architected 실습 - 인벤토리 및 패치 관리

- 실습: AWS Systems Manager

SUS06-BP03 구축 환경의 사용률 제고

워크로드를 개발, 테스트 및 구축하기 위한 리소스 활용도를 높입니다.

일반적인 안티 패턴:

- 구축 환경을 수동으로 프로비저닝하거나 종료합니다.
- 테스트, 구축 또는 릴리스 작업과 별개로 구축 환경을 계속 실행 중인 상태로 유지합니다(예: 개발 팀원이 근무 시간 외에 환경을 실행).
- 구축 환경에 리소스를 과도하게 프로비저닝합니다.

이 모범 사례 확립의 이점: 빌드 환경의 활용률을 높임으로써 클라우드 워크로드의 전반적인 효율성을 개선하는 동시에 효과적으로 개발, 테스트 및 구축 작업을 수행하도록 빌더에 리소스를 할당할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

자동화와 코드형 인프라를 사용하여 필요 시 빌드 환경을 가동하고 사용하지 않을 때는 해당 환경을 종료합니다. 일반적인 패턴은 개발 담당 팀원의 근무 시간과 일치하는 가용 기간을 예약하는 것입니다. 테스트 환경은 프로덕션 구성과 매우 유사해야 합니다. 그러나 버스트 용량, Amazon EC2 스팟 인스턴스, 자동 확장 데이터베이스 서비스, 컨테이너 및 서버리스 기술과 함께 인스턴스 유형을 사용하여 개발 및 테스트 용량을 용도에 맞게 조정할 수 있는 기회를 찾아야 합니다. 테스트 요구 사항만 충족하도록 데이터 볼륨을 제한합니다. 테스트에서 프로덕션 데이터를 사용하는 경우 프로덕션 데이터를 공유하고, 데이터를 이동하지 않을 수 있는지 알아봅니다.

구현 단계

- 코드형 인프라를 사용하여 구축 환경을 프로비저닝합니다.
- 자동화를 사용하여 개발 및 테스트 환경의 수명 주기를 관리하고 구축 리소스의 효율성을 극대화합니다.
- 전략을 사용하여 개발 및 테스트 환경의 활용도를 극대화합니다.
  - 현실적인 최소한의 재현 환경을 사용하여 잠재적 개선 사항을 개발 및 테스트합니다.
  - 가능한 경우 서버리스 기술을 사용합니다.
  - 온디맨드 인스턴스를 사용하여 개발자 디바이스를 보완합니다.

- 버스트 용량이 포함된 인스턴스 유형, 스팟 인스턴스 및 기타 기술을 사용하여 사용량에 맞게 구축 용량을 조정합니다.
- 배스천 호스트 플릿을 배포하는 대신 네이티브 클라우드 서비스를 도입하여 보안 인스턴스 셸에 액세스합니다.
- 구축 작업에 따라 구축 리소스를 자동으로 확장합니다.

리소스

관련 문서:

- AWS Systems Manager Session Manager
- Amazon EC2 버스트 가능 성능 인스턴스
- AWS CloudFormation이란 무엇인가요?
- AWS CodeBuild란 무엇인가요?
- Instance Scheduler on AWS

관련 동영상:

- Continuous Integration Best Practices(지속적 통합 모범 사례)

SUS06-BP04 테스트에 관리형 Device Farm 사용

관리형 Device Farm을 사용하여 대표적인 하드웨어 집합에서 새 기능을 효율적으로 테스트합니다.

일반적인 안티 패턴:

- 물리적 개별 디바이스에서 애플리케이션을 수동으로 테스트하고 배포합니다.
- 앱 테스트 서비스를 사용하여 실제 물리적 디바이스에서 앱(예: Android, iOS 및 웹 앱)을 테스트하고 상호 작용하지 않습니다.

이 모범 사례 확립의 이점: 관리형 Device Farm을 사용하여 클라우드 지원 애플리케이션을 테스트하면 다음과 같은 여러 이점을 얻을 수 있습니다.

- 여기에는 다양한 디바이스에서 애플리케이션을 테스트할 수 있는 보다 효율적인 기능이 포함되어 있습니다.
- 테스트를 위한 사내 인프라가 필요하지 않습니다.

- 비교적 널리 사용되지 않는 구형 하드웨어를 포함하여 다양한 디바이스 유형을 제공하므로 불필요한 디바이스 업그레이드가 필요하지 않습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

관리형 Device Farm을 사용하면 대표적인 하드웨어 집합에서 새 기능에 대한 테스트 프로세스를 간소화할 수 있습니다. 관리형 Device Farm은 사용 빈도가 낮은 오래된 하드웨어를 포함하여 다양한 디바이스 유형을 제공하며, 불필요한 디바이스 업그레이드로 인해 고객의 지속 가능성이 영향을 받지 않도록 합니다.

구현 단계

- 테스트 요구 사항 및 플랜(예: 테스트 유형, 운영 체제 및 테스트 일정)을 정의합니다.
  - Amazon CloudWatch RUM을 사용하여 클라이언트 측 데이터를 수집 및 분석하고 테스트 플랜을 구체화할 수 있습니다.
- 테스트 요구 사항을 지원할 수 있는 관리형 Device Farm을 선택합니다. 예를 들어, AWS Device Farm을 사용하여 변경 사항이 대표적인 하드웨어 집합에 미치는 영향을 테스트하고 이해할 수 있습니다.
- 지속적 통합(CI) 및 지속적 전달(CD)을 사용하여 테스트를 예약하고 실행합니다.
  - Integrating AWS Device Farm with your CI/CD pipeline to run cross-browser Selenium tests(CI/CD 파이프라인과 AWS Device Farm을 통합하여 교차 브라우저 Selenium 테스트 실행)
  - Building and testing iOS and iPadOS apps with AWS DevOps and mobile services(AWS DevOps 및 모바일 서비스로 iOS 및 iPadOS 앱 구축 및 테스트)
- 테스트 결과를 지속적으로 검토하고 필요한 개선을 수행합니다.

리소스

관련 문서:

- AWS Device Farm device list(AWS Device Farm 디바이스 목록)
- CloudWatch RUM 대시보드 보기

관련 예시:

- Android용 AWS Device Farm 샘플 앱

- iOS용 AWS Device Farm 샘플 앱
- AWS Device Farm용 Appium Web 테스트

관련 동영상:

- Optimize applications through end user insights with Amazon CloudWatch RUM(Amazon CloudWatch RUM을 통해 최종 사용자 인사이트로 애플리케이션 최적화)

고지 사항

고객은 본 문서에 포함된 정보를 독자적으로 평가할 책임이 있습니다. 본 문서는 (a) 정보 제공만을 위한 것이며, (b) 사전 고지 없이 변경될 수 있는 현재의 AWS 제품 제공 서비스 및 사례를 보여 주며, (c) AWS 및 자회사, 공급업체 또는 라이선스 제공자로부터 어떠한 약정 또는 보증도 하지 않습니다. AWS 제품 또는 서비스는 명시적이든 묵시적이든 어떠한 종류의 보증, 진술 또는 조건 없이 '있는 그대로' 제공됩니다. 고객에 대한 AWS의 책임과 법적 책임은 AWS 계약서에 준하며 본 문서는 AWS와 고객 간의 계약에 포함되지 않고 계약을 변경하지도 않습니다.